Dongbu Robot

DR Visual Logic

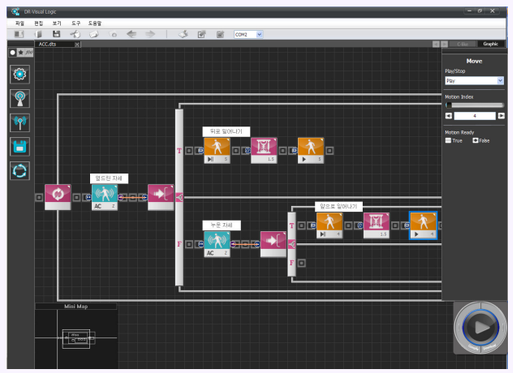
for HOVIS Genie

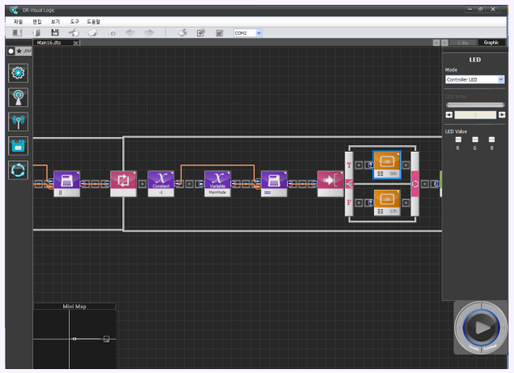
- DR-Visual Logic 소개
- DR-Visual Logic 으로 HOVIS Genie 프로그래밍 시작
- DR-Visual Logic 사용자 인터페이스
- DR-Visual Logic (HOVIS Genie) 모듈별 예제 프로그래밍

www.dongburobot.com

DR Visual Logic
For Hovis Genie

CONTENTS





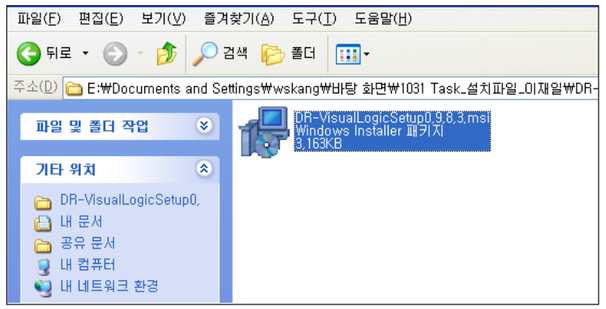
01 DR-Visual Logic 소개

DR-Visual Logic 은 동부로봇에서 개발한 고유의 로봇 프로그래밍 언어를 Graphic 모듈화한 로봇 프로그래밍 툴입니다.
마우스만을 활용하여 드래그 앤 드랍 방식으로 초보자도 쉽게 프로그래밍 할 수 있는 구조이며, C-like 탭을 제공하여, 그래픽 프로그램의 소스코드를 바로 확인 할 수 있습니다. C 문법 과 유사한 코드이기 때문에 프로그래밍 입문자가 C문법을 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 언어입니다.
앞으로 업그레이드된 제어기 관련 모듈 추가 및 로봇 모션의 모듈화 및 시뮬레이션 결합 등으로 다채로운 기능을 발휘하는 프로그램으로 업그레이드 할 예정입니다.

■ 시스템 요건

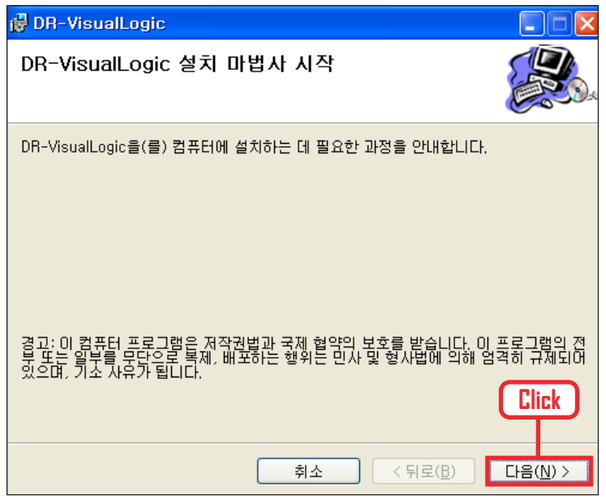
- 최소 인텔 팬티엄 800 Mhz
- Windows XP/Vista/7
- 최소 256 MB RAM
- 하드디스크 설치 공간 약 20 MB 필요
- USB Port
- Wi-Fi 환경 (무선 AP 필요)

1.1 설치하기 / 설치부터 실행까지 따라해보세요



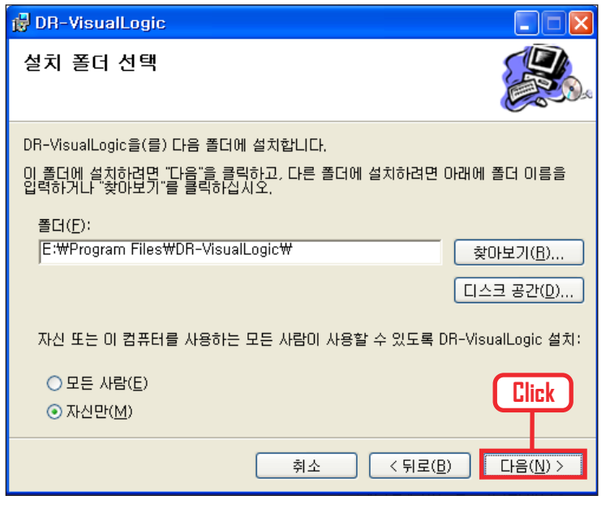
01 설치파일

Hovis Genie 전용 DR-Visual Logic 설치파일을 클릭하여 시작합니다.



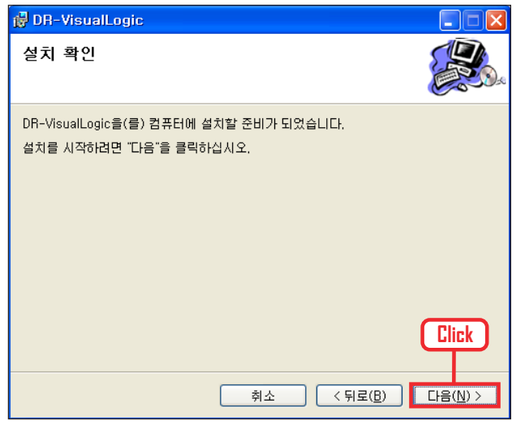
02 설치 마법사 시작

"다음" 버튼을 클릭합니다.



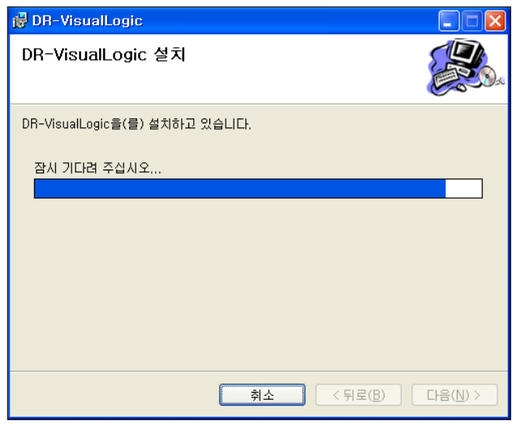
03 설치 폴더를 선택

"다음" 버튼을 클릭합니다.



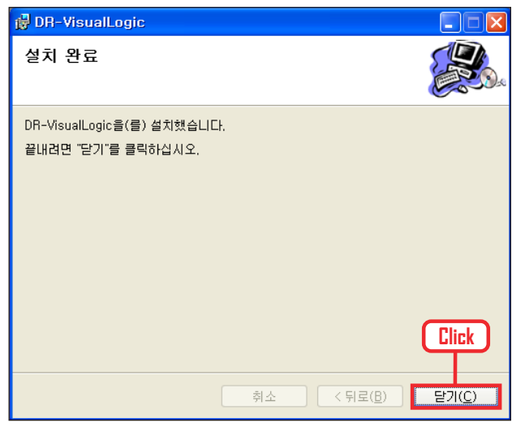
04 설치 확인

"다음" 버튼을 클릭합니다.



05 설치 시작

설치 시작 합니다.
프로그레스 바가 끝날때 까지 기다려 주세요.



06 설치 완료 선택

"닫기" 를 클릭하세요
소프트웨어 설치가 완료되었습니다.

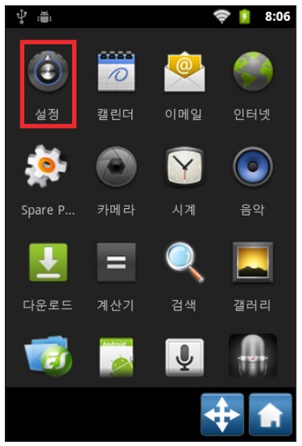
07 실행파일 확인

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

DR-Visual Logic 으로 HOVIS Genie 프로그래밍 시작 02

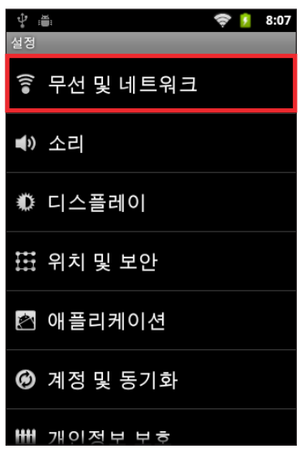
Wi-Fi 연결하기

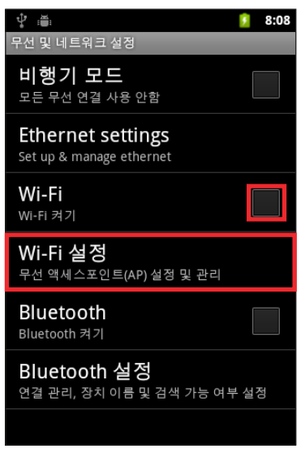
DR-Visual Logic의 첫 번째 프로그램을 작성하기에 앞서, DR-Visual Logic과 로봇의 MID를 Wi-Fi(무선 AP)를 통해 연결합니다.
MID를 무선 AP에 연결하는 방법은 다음과 같습니다.



01 MID 전원 ON

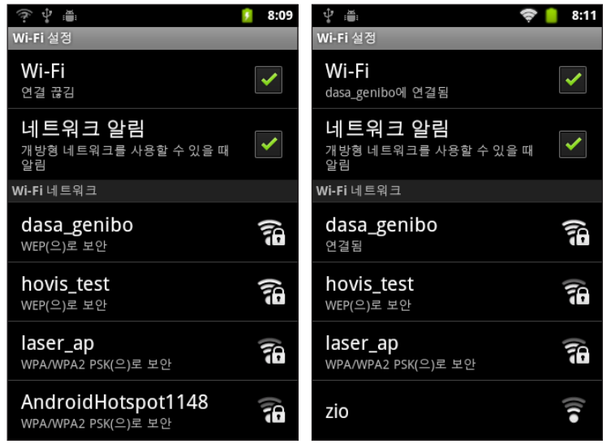
MID의 전원을 킵니다. 전원버튼을 약 3초 정도 누른 후 떼면 MID의 전원이 켜집니다. 잠시 후, MID가 부팅이 완료되면 화면 하단 오른쪽 메뉴버튼을 누릅니다.





02 무선 및 네트워크 설정

설정 〉 무선 및 네트워크 에서 Wi-Fi를 클릭하여 Wi-Fi를 킵니다. 그리고 Wi-Fi 설정을 클릭합니다.

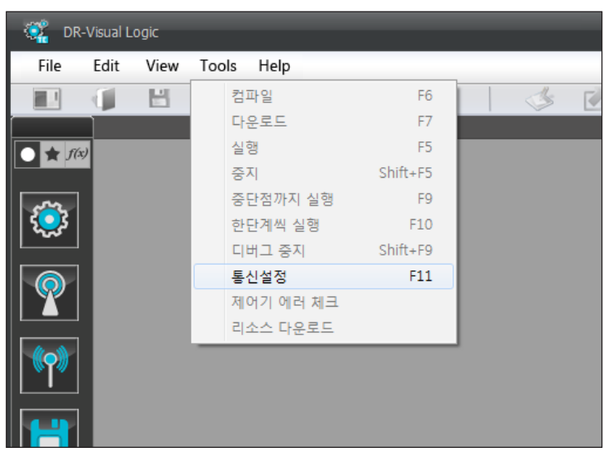


03 PC와 연결된 AP에 MID 연결

PC를 무선 AP에 연결합니다. 그리고 MID에서 PC와 연결 된 무선 AP를 선택합니다. (이 예제에서 PC 와 연결 된 AP는 dasa_genibo) 오른쪽 화면과 같이 dasa_genibo에 "연결됨"이 표시되고, 화면 상단에 Wi-Fi가 표시되면 성공적으로 연결된 것입니다.

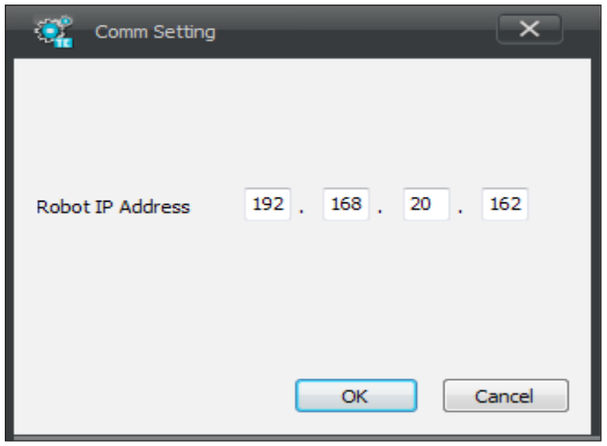
04 IP 확인

03에서 연결한 무선 AP인 dasa_genibo를 다시 선택하여 IP를 확인합니다.
(IP 주소 192.168.20.162)



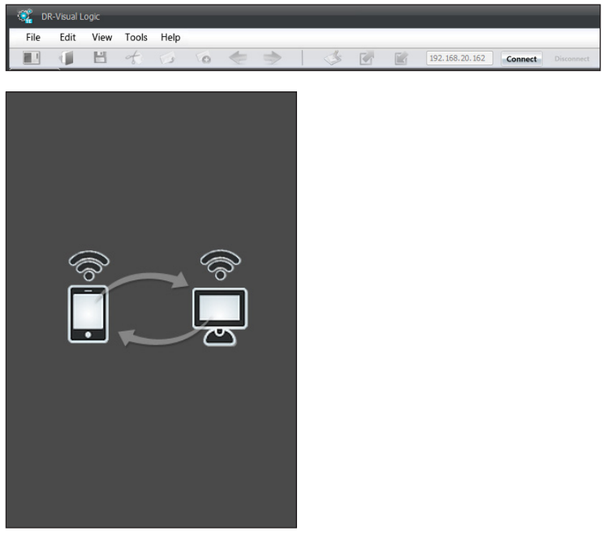
05 DR-Visual Logic에서 통신 설정

DR-Visual Logic에서 Tool 〉 통신설정 (단축키 F11)을 선택합니다.



06 DR-Visual Logic에서 통신 설정

04에서 확인 한 IP를 입력한 후, OK 버튼을 클릭합니다.



07 DR-Visual Logic과 MID의 연결

화면 오른쪽 상단에 있는 Connect 버튼을 클릭합니다.
MID의 화면이 다음과 같이 변경되고, DR-Visual Logic과 연결되었다는 안내멘트가 나옵니다. 이로써 PC와 MID의 Wi-Fi 연결이 완료되었습니다.

– 연결이 되지 않는다면
1. PC가 무선 AP에 연결되었는지 확인
2. MID의 DR-Visual Logic Player 앱을 다시 실행

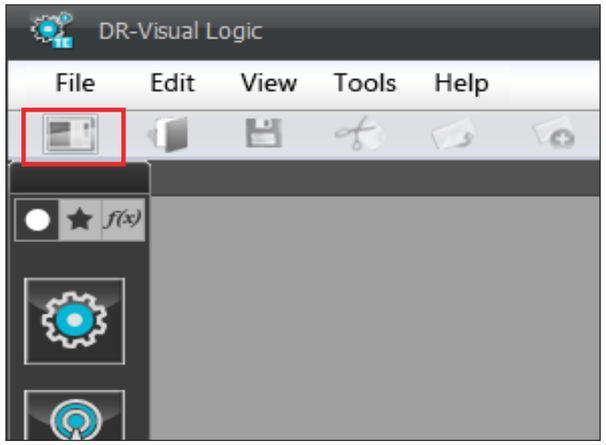
2.1 첫 번째 프로그램 따라하기

[예제설명]

첫 번째 프로그램 예제입니다. DR-Visual Logic에서 작성한 프로그램을 Hovis Genie에서 실행해봅니다. 프로그램을 실행하면, 로봇이 "안녕하세요. 저는 호비스 지니입니다."를 말하며, 잠시 동안 좌우로 회전합니다.

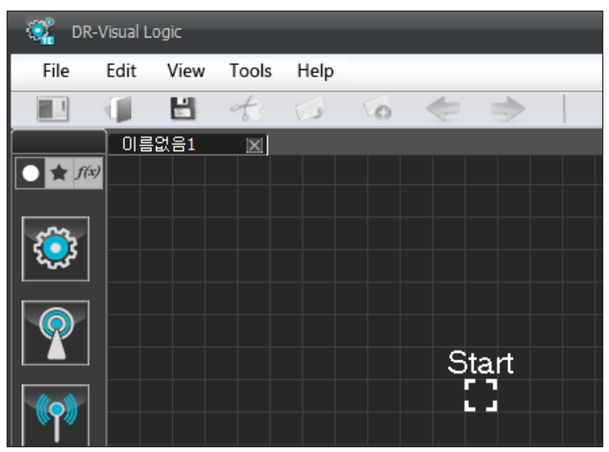
Tip. DR-Visual Logic 화면 컨트롤

- 마우스 휠 업 (화면 크게 보기)
- 마우스 휠 다운 (화면 작게 보기)
- Shift + 마우스 좌클릭 후 이동 (화면 평행 이동)



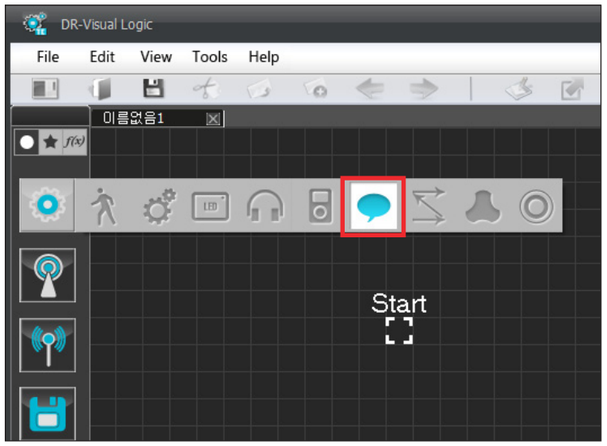
01 새로 만들기

도구 모음의 가장 왼쪽 아이콘을 클릭하여(단축키 Ctrl+N) 새로운 dts 파일을 만듭니다.



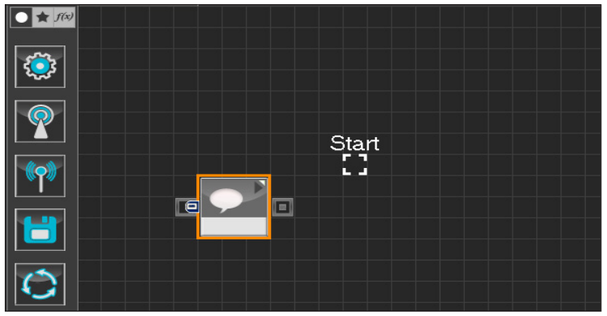
02 새창

새로운 파일이 생성되었습니다.



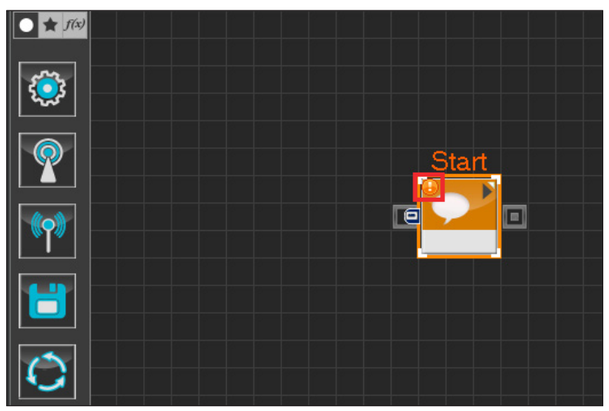
03 모듈 선택

모듈을 배치하기 위해서는 왼쪽 모듈바에서 모듈을 클릭해야 합니다.
Motion 〉 TTS 모듈을 클릭합니다.



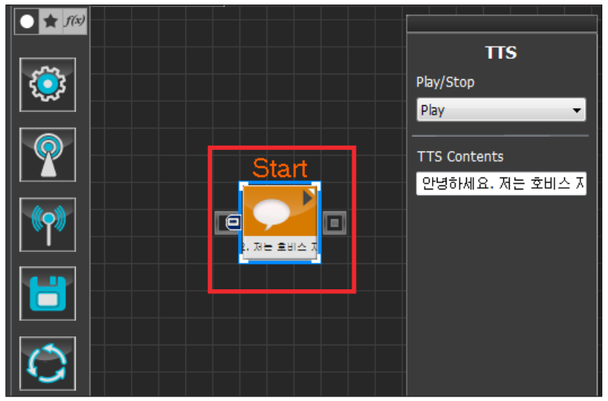
04 모듈 배치하기

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



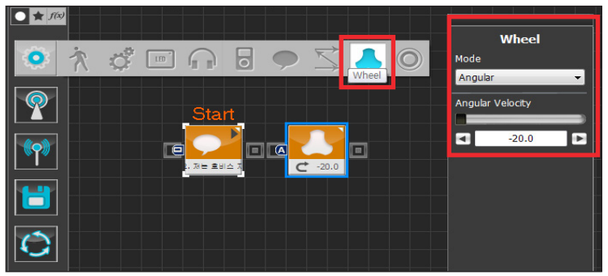
05 프로그래밍 시작

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



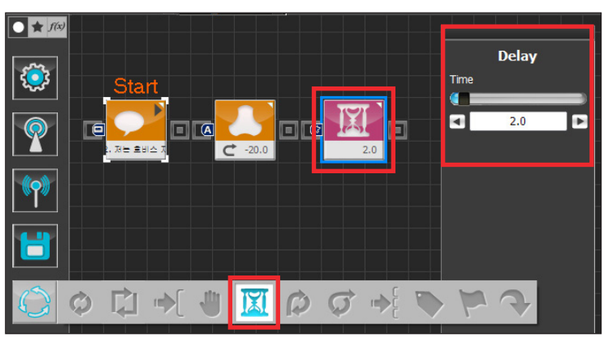
06 Warning의 해결

모듈에 해결되지 않은 Warning이 있는 경우, 모듈 왼쪽 상단에 작은 느낌표 아이콘이 생깁니다. 마우스를 오버하면 해당 Warning에 대한 내용을 볼 수 있습니다. 여기에서는 TTS 내용을 입력하지 않아 Warning이 발생하였습니다. 다음과 같이 TTS 내용을 입력합니다.
"안녕하세요. 저는 호비스 지니입니다."



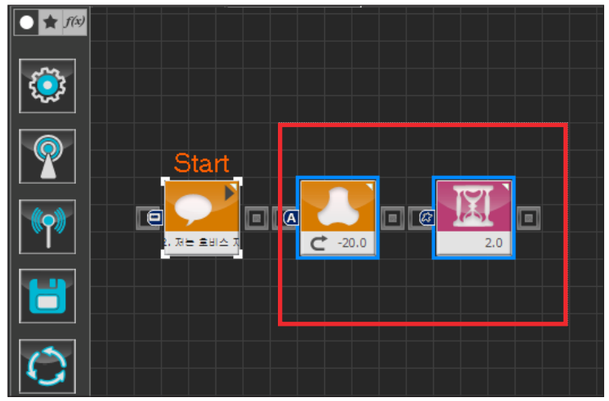
07 Wheel 제어

Motion 〉 Wheel 모듈을 선택합니다. 그리고 그림과 같이 Wheel 모듈이 활성화되도록 배치합니다. 배치된 Wheel 모듈을 선택하여 Mode를 Angular, Angular Velocity를 –20으로 선택합니다. 여기서 Angular Mode는 로봇을 제자리 회전하기 위해서 지정하며, Angular Velocity가 –이면 우회전, +이면 좌회전입니다.



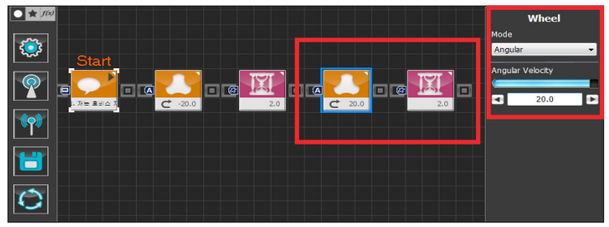
08 Delay

다음 명령을 수행하기 전 2초간 기다립니다.현재 Wheel에 지정된 제자리 우회전 명령을 2초간 유지하는 효과가 있습니다.



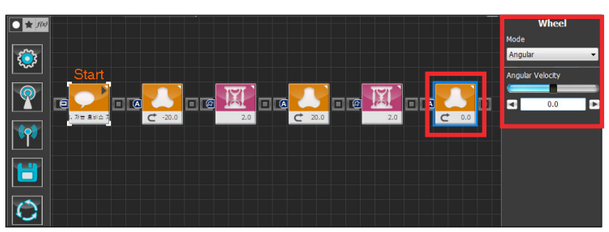
09 모듈의 복사

다음은 제자리 좌회전 명령을 2초간 유지하도록 합니다. 위의 07–08에서 작성한 두 모듈을 복사하여 붙여 넣고, 복사된 Wheel 모듈에 Angular Velocity의 값만 변경하면 됩니다. 그림과 같이 마우스로 두개의 모듈을 드래그 하여 Ctrl+C를 누릅니다.



10 모듈의 붙여 넣기

09에서 복사한 모듈을 Ctrl+V를 눌러 붙여 넣습니다. 복사된 Wheel 모듈의 Angular Velocity를 +20으로 변경합니다.



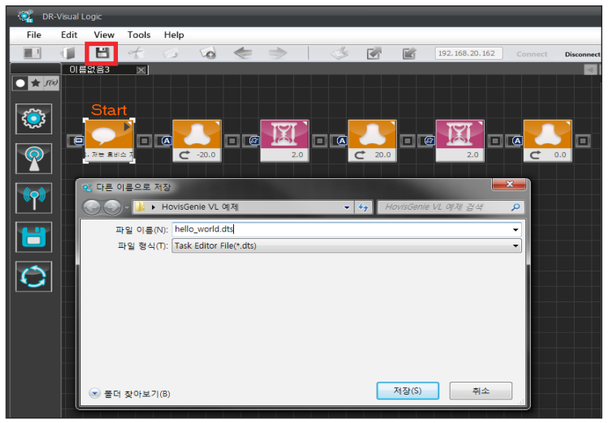
11 Wheel 멈춤

Wheel 모듈의 Angular Velocity를 0으로 지정하여 Wheel의 동작을 멈춥니다.

```
1  void main()
2  {
3          mid_tts_play( "안녕하세요. 저는 호비스 지니입니다." )
4          mid_wheel_angular( -20.0 )
5          delay( 2000 )
6          mid_wheel_angular( 20.0 )
7          delay( 2000 )
8          mid_wheel_angular( 0.0 )
9  }
```

12 C-like 보기

프로그램 작성이 완료되었습니다. 오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 화면이 나옵니다.
C와 유사한 문법 구조를 가지고 있으므로 C문법 선행학습 효과가 있습니다.
각 모듈 별로 클릭하면 커서가 따라서 움직이므로 모듈이 Text로 어떻게 변환하는지 확인할 수 있습니다.

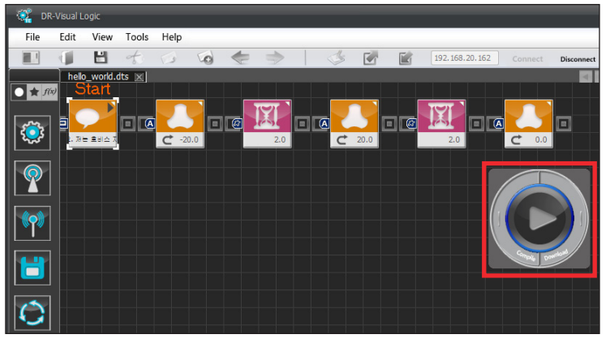


13 파일 저장하기

도구 모음의 세 번째 아이콘을 클릭 (단축키 Ctrl+S)하여 파일을 저장합니다.
(hello_world.tts)

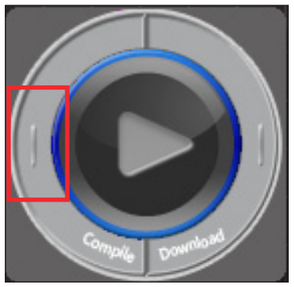
2.2 프로그램 컴파일, 다운로드 및 실행

지금까지 PC(DR-Visual Logic)와 MID를 무선 AP를 통해 연결하고, 첫 번째 프로그램 예제를 작성하였습니다. 여기에서는 작성한 첫 번째 예제 프로그램을 컴파일하여 MID에 다운로드 하고 실행해봅니다.



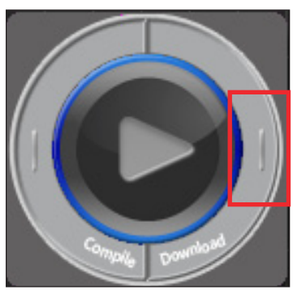
01 컴파일, 다운로드 및 실행

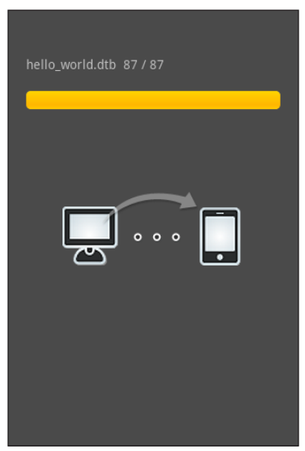
프로그래밍 후 컴파일 -> 다운로드 -> 실행의 과정을 거칩니다.
컴파일, 다운로드 및 실행을 하기 위해서 리모콘 윈도우를 사용합니다.



02 컴파일

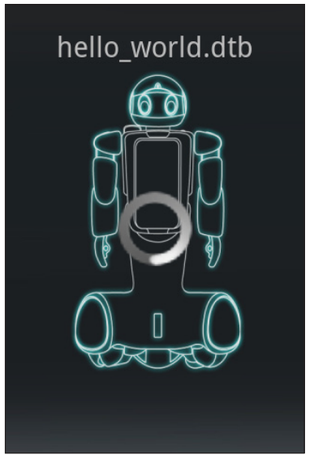
Compile 을 클릭하면 컴파일이 됩니다. 에러가 없으면 성공적으로 컴파일이 되며, 에러 발생시 메시지를 확인하시고 적절한 조치를 취하세요.
(컴파일은 dts 파일을 실행가능한 dtb 파일로 변경합니다)





03 다운로드

Download를 클릭하면 컴파일 된 파일을 MID에 다운로드합니다. PC와 MID 연결에 이상이 없다면, MID가 다음화면으로 바뀌며 다운로드가 됩니다.
(다운로드 클릭 시 자동으로 컴파일이 되므로, 02의 과정을 생략하실 수 있습니다.)



04 실행

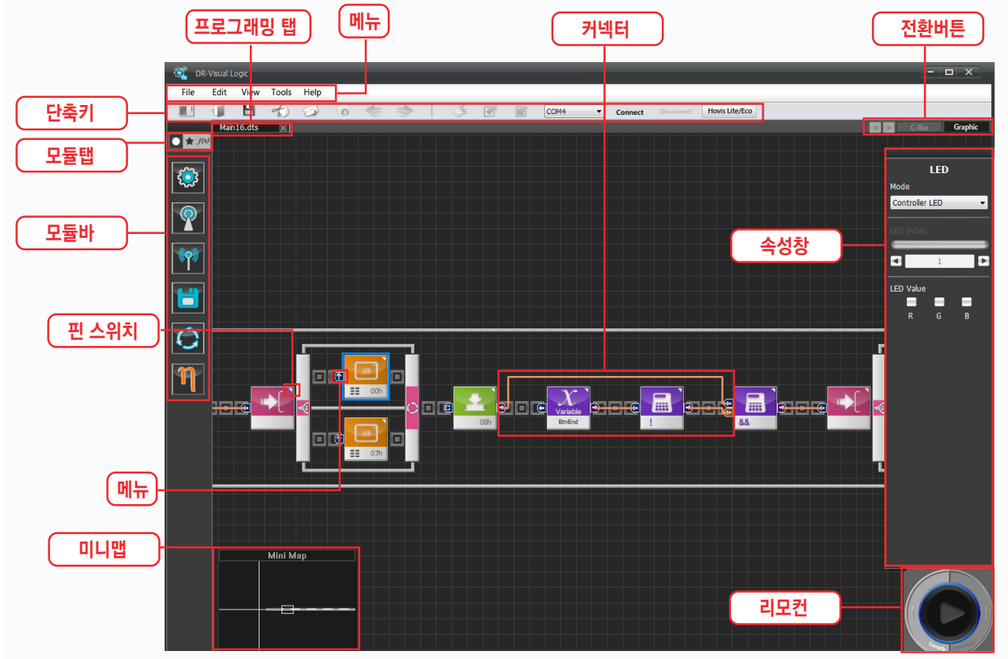
다운로드가 완료되면, 가운데 실행버튼을 눌러 다운로드 한 파일을 로봇에서 실행할 수 있습니다.

실행버튼을 클릭하면, MID 화면이 다음과 같이 변경되며 프로그램을 실행합니다.

※주의사항
– 실행버튼을 누르기 전, 반드시 다운로드 과정을 거쳐야 합니다.
– 실행버튼을 눌러도 실행이 되지 않는다면, 로봇 전원이 켜져 있는지 확인하세요.

DR–Visual Logic 사용자 인터페이스

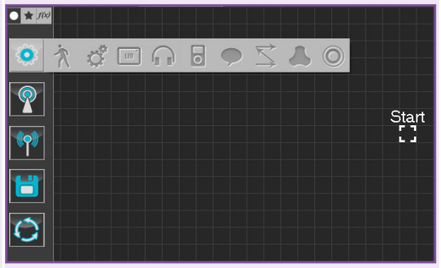
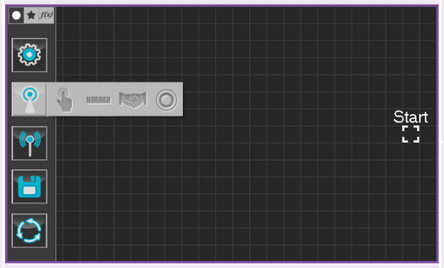
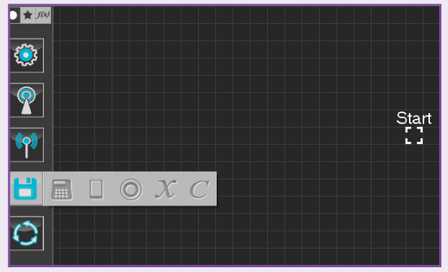
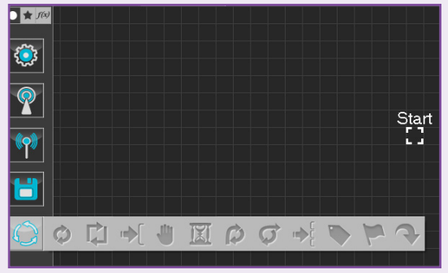
03



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.

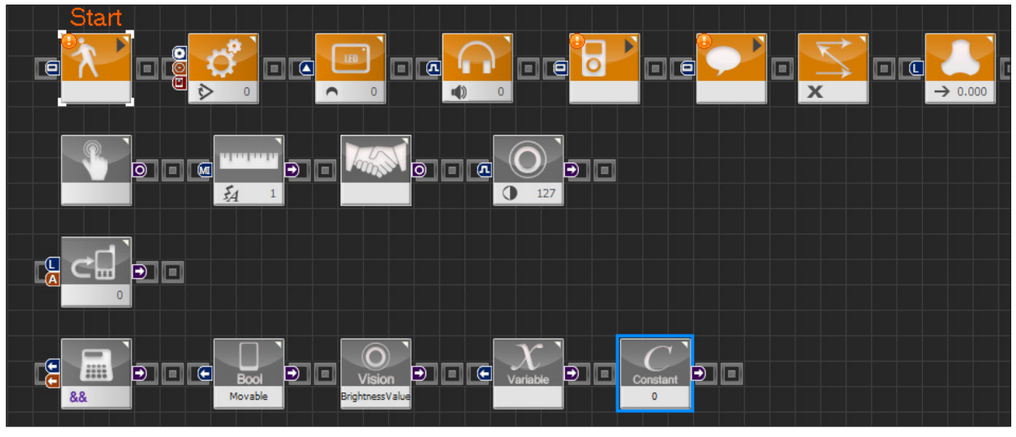
3.1 프로그래밍 모듈

DR-Visual Logic은 아래와 같은 모듈로 구성됩니다.
모듈 팩에는 프로그램을 만들기 위해 필요한 모든 프로그래밍 모듈이 포함되어 있습니다. DR-Visual Logic의 HOVIS Genie의 경우 MID가 지원해주는 기능으로 구성되어 있습니다.

Module Pack	Picture	Module	Description
Motion (모션)		Move	저장된 로봇 모션을 실행함
		Motor	모터별로 위치 / 속도제어
		LED	Head LED – 머리의 눈, 귀, 입, 이마 LED 제어
		Sound	지정한 멜로디 실행
		MP3	지정한 MP3 파일 재생
		TTS	Text를 음성으로 변환.
		Navigation	2차원 평면상에 기준점을 설정하고, x, y 좌표를 추가하여 로봇을 이동
		Wheel	로봇의 직진(Linear) 및 회전(Angular) 속도를 지정하여 로봇을 구동
		Vision Capture	MID의 전면 카메라의 영상을 캡쳐
Sensor (센서)		Touch Sensor	머리 모듈의 터치 여부
		Distance Sensor	거리 측정 구동부 전면에 5개 구동부 하단에 3개
		Hand Touch Sensor	양손바닥의 택트 스위치 눌림 여부
		Vision Sensor	캡쳐 된 화면에 대한 영상처리
Communication (통신)		IR Receive	적외선 리모컨 데이터 인식
Data (데이터)		Operator	연산자(논리/산술/비교/비트/증감)
		MID RAM	MID에서 사용하는 지정 변수를 읽거나 쓸 때 사용
		Vision Result	캡쳐 된 화면에 대한 영상처리 값을 읽을 때 사용
		Variable	사용자 변수를 쓸 때 사용
		Constant	상수 값을 입력할 때 사용
Flow (흐름 제어)		Loop	무한 반복/for문
		While	while문(특정 조건이 참이면 반복)
		If-else	if-else문(제어 분기)
		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

3.2 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.

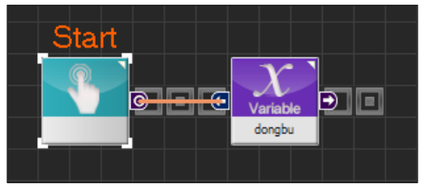
3.3 일반형 모듈 사용하기

Motion 모듈 팩의 모듈은 출력이 없으며, 단독으로 있어도 소스로 변환됩니다.


```
1  void main()
2  {
3          mid_motion( "dongbu.dmt", 0, 0 )
4          jog( 512, 0, 0, 60 )
5          mid_led_brow( 0 )
6          mid_sound( 0 )
7          mid_mp3_play( "dongbu.mp3" )
8          mid_tts_play( "안녕하세요" )
9          mid_navi_init()
10         mid_wheel_linear( 0.000, 0, false )
11 }
```


Sensor 모듈, Communication 모듈, 그리고 Data의 Constant는 단독으로 소스로 변환되지 않습니다.
이 모듈들은 출력 핀이 다른 입력 핀과 연결 되어야만 의미를 가집니다. 단독으로 소스로 변환되지 못할 때, 모듈의 왼쪽 상단에 오류가 표시됩니다.

이런 모듈들은 출력 핀을 다른 입력 핀으로 적절히 연결하면 오류가 사라지며, C-like 소스코드로 변환됩니다.
※ Data 모듈 팩에 속한 모듈들은 단독으로 소스로 변환되는 경우도 있습니다.



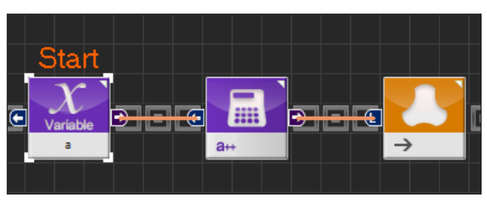

```
1  int dongbu
2  void main()
3  {
4          dongbu=( MPSU_TouchStatus )
5  }
```

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

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          a++
5  }
```

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




```
1  int a
2  void main()
3  {
4          mid_wheel_linear( ( a++ ), 0, false )
5  }
```

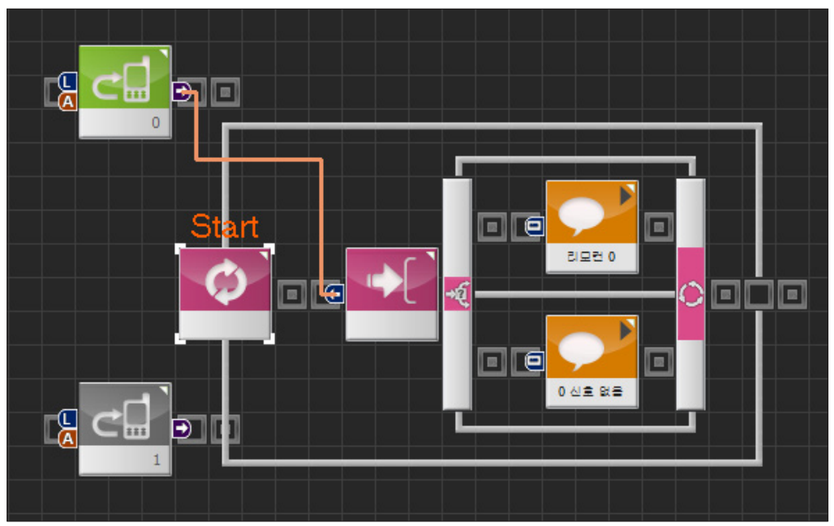
그 외에 MID RAM, Vision Result, Variable은 단독으로 소스로 변환되지 않습니다.

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


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

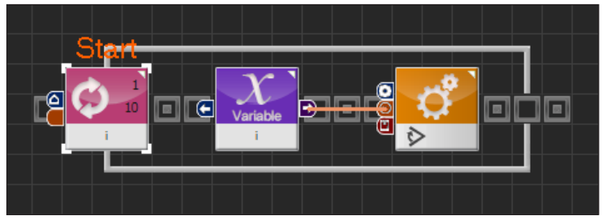
한편, MID RAM, Vision Result 중에는 값을 읽을 수만 있는 값(Read-only)이 있습니다.
이 경우 입력 핀이 없어지며 대입이 불가능합니다.

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.



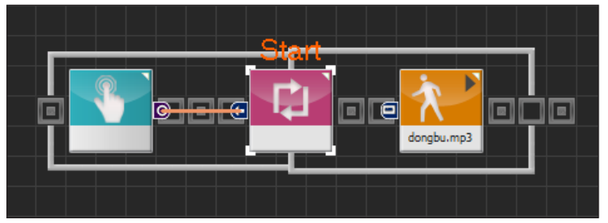
3.4 프로그래밍 모듈 〉 Flow 형 모듈

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.



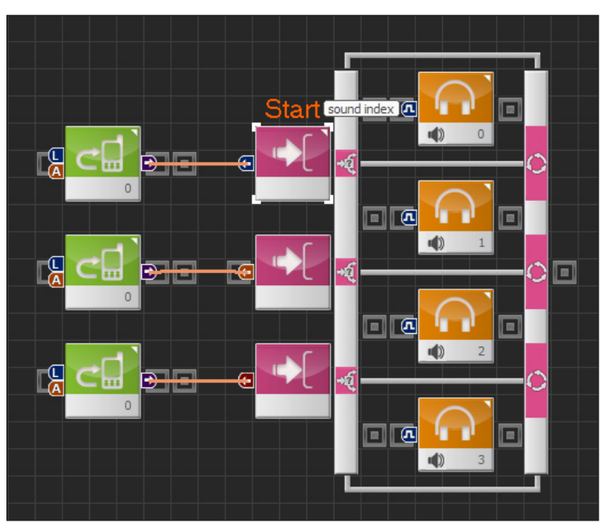
Loop

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



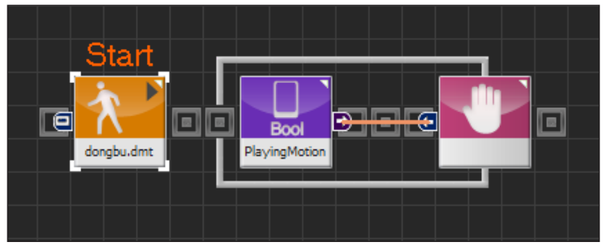
While

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.

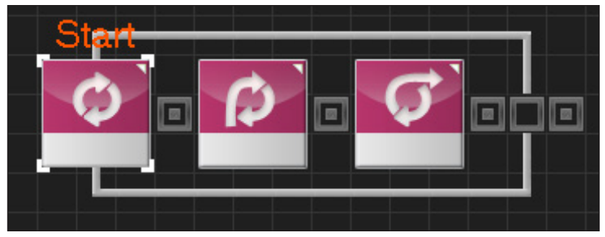


Wait

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

Delay

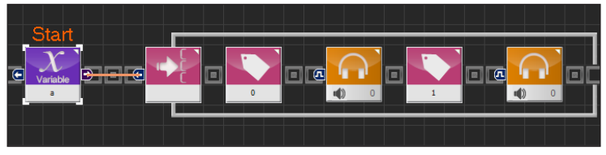
일정 시간 동안 프로그램의 실행을 지연시킵니다.



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-case문을 구성합니다. switch 모듈의 입력에 따라서, 일치하는 case문 이후의 것을 실행합니다.

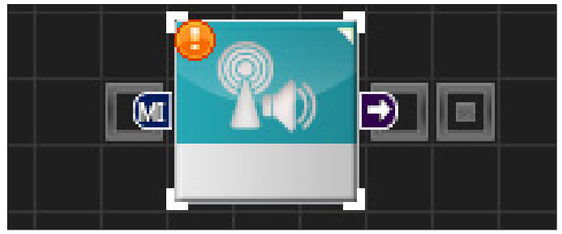
Label, Goto

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

3.5 프로그래밍 모듈 〉 핀

모듈에 따라 입력 값과 출력 값을 갖는 모듈이 있습니다.

출력 핀의 결과 값은 다른 모듈의 입력 핀과 연결되어 다른 모듈의 입력 값으로 동작합니다.



핀

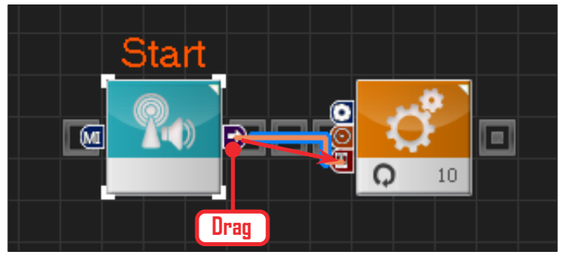
입출력 값이 있는 모듈은 모듈상에 좌우측에 핀이 나와 있습니다. 왼쪽은 입력 핀, 오른쪽은 출력 핀입니다.

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.

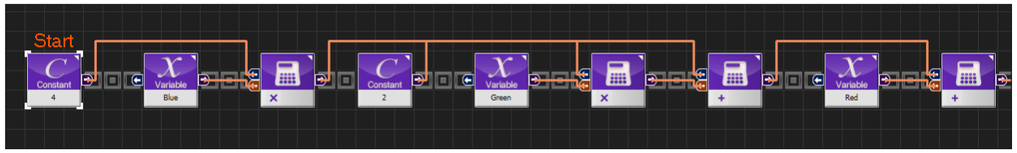


핀 연결

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

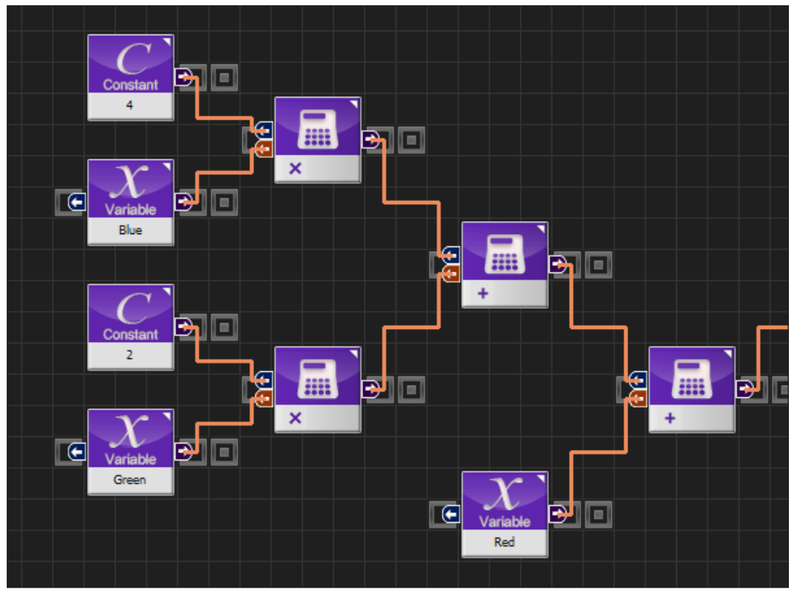
3.6 프로그래밍 모듈 〉 연결형 타입

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.

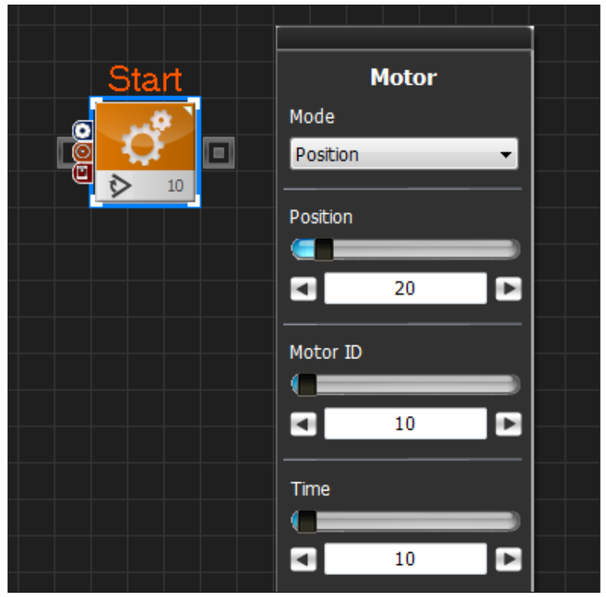


병렬형 연결

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

3.7 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

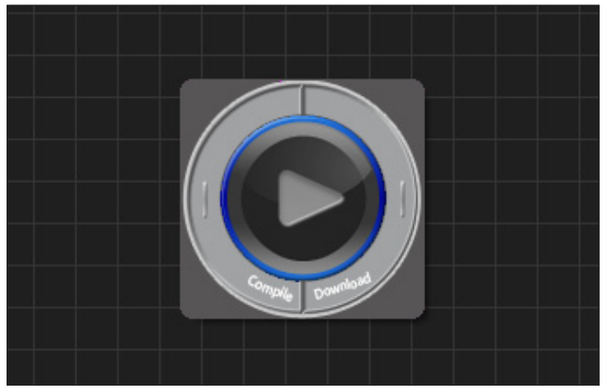
모터모듈을 클릭했을 때 우측과 같은 속성창이 나타납니다.

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.

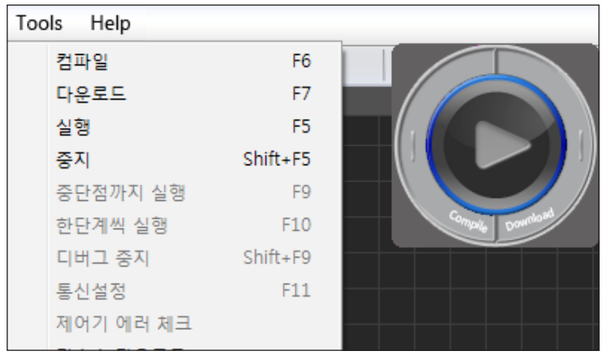
3.8 컴파일/다운로드 하기

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 : 컴파일 한 프로그램을 다운로드 합니다.

실행 : 다운로드 한 프로그램을 실행합니다.

중지 : 프로그램을 중지시킵니다.

중단점까지 실행 : 차기 버전에서 지원 예정입니다.

한단계씩 실행 : 차기 버전에서 지원예정입니다.

리소스 다운로드 : 프로그램에서 사용되는 리소스를 다운로드 합니다. (DMT, MP3)

3.9 다양한 기능

메뉴 기능

DR-Visual Logic에서 제공하는 메뉴의 기능은 아래 표와 같습니다.

분 류	항 목	설 명
File	새창/新規/New Window	새로운 dts 파일을 만듭니다.
	불러오기/開く/Load	저장된 dts 파일을 불러옵니다.
	창닫기/閉じる/Close	현재 열린 dts 파일을 닫습니다.
	저장/保存/Save	현재 dts 파일을 저장합니다.
	다른이름저장/名前を付けて保存/Save As	현재 dts 파일을 다른 이름으로 저장합니다.
	인쇄미리보기/印刷プレビュー/Print Preview	인쇄하기 전에 화면상으로 미리 봅니다.
	인쇄/印刷/Print	인쇄합니다.
	종료/終了/Quit	프로그램을 종료합니다.
Edit	전단계/元に戻す/Undo	직전의 작업을 취소합니다.
	앞복원/やり直し/Redo	취소한 작업을 다시 복원합니다.
	잘라내기/切り取り/Cut	현재 선택된 부분을 클립보드로 잘라냅니다.
	복사하기/コピー/Copy	현재 선택된 부분을 클립보드로 복사합니다.
	붙여넣기/貼り付け/Paste	클립보드의 내용을 붙여넣습니다.
	삭제/削除/Delete	현재 선택된 부분을 삭제합니다.
View	기본보기/基本表示/Default View	확대/축소 상태를 기본 상태로 만듭니다.
	화면확대/拡大/Zoom In	화면을 확대합니다.
	화면축소/縮小/Zoom Out	화면을 축소합니다.
	모듈을 중앙으로/モジュールを中央に/Center This Module	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.

분 류	항 목	설 명
Tools	컴파일/コンパイル/Compile	작성한 프로그램을 컴파일합니다.
	다운로드/ダウンロード/Download	컴파일한 프로그램을 다운로드 합니다.
	실행/実行/Run	다운로드 한 프로그램을 실행합니다.
	중지/停止/Stop	프로그램을 중지시킵니다.
	중단점까지 실행/ブレークポイントまで実行 /Run to Breakpoint	C-like 창에서 소스에 중단점을 지정하면 그곳까지 실행 후 일시 중단합니다. (차기 버전 지원 예정)
	한단계씩 실행/一段階ずつ実行 /Run Step by Step	C-like 창에서 소스 한 줄 씩 실행합니다. (차기 버전 지원 예정)
	디버그 중지/デバッグの停止 /Stop Debugging	디버깅을 중지시킵니다. (차기 버전 지원 예정)
	통신설정/通信設定/Comm Setting	Wi-Fi 통신을 설정합니다.
	제어기 에러 체크/コントローラー エラー チェック/Controller Error Check	MID의 에러 상태를 체크합니다. (차기 버전 지원 예정)
Help	내용색인/内容索引/Index	도움말 파일을 엽니다.
	온라인지원/オンラインサポート/Online Help	동부로봇 홈페이지를 엽니다.
	소프트웨어업데이트/ソフトアップデート/S/W Update	소프트웨어가 최신 버전인지 확인합니다.
	펌웨어업데이트/ファームウェアアップデート/F/W Update	제어기의 펌웨어를 업데이트합니다. (Hovis Genie는 해당 사항 없음)
	펌웨어복구/ファームウエア復旧/ F/W Restore	제어기에 다른 펌웨어가 쓰여졌거나 제어기가 고장난 경우 펌웨어를 복구합니다. (Hovis Genie는 해당 사항 없음)
	프로그램정보/プログラム情報/About	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.

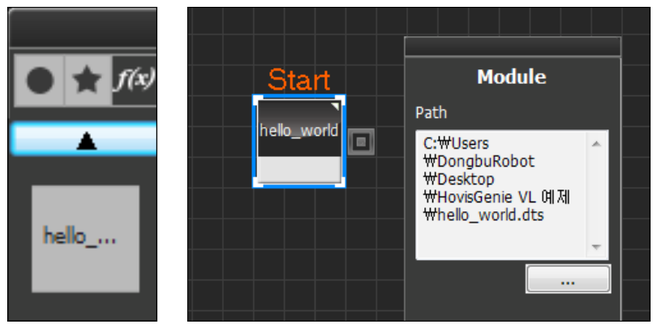
Wheel Up : 화면 크게 보기

Wheel Down : 화면 작게 보기

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, 추가 할 dts 파일을 선택할 수 있습니다.



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

```
1  void hello_world()
2  {
3          mid_tts_play( "안녕하세요. 저는 호비스 지니입니다
4          mid_wheel_angular( -20.0 )
5          delay( 2000 )
6          mid_wheel_angular( 20.0 )
7          delay( 2000 )
8          mid_wheel_angular( 0.0 )
9  }
10 void main()
11 {
12         hello_world()
13 }
```

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

04 DR-Visual Logic (HOVIS Genie)

모듈 별 기본 예제 프로그래밍

DR-Visual Logic에서 Hovis Genie를 제어하기 위하여 각 모듈 별 기본 예제 프로그램입니다.

예제 명	해당 모듈	예제 설명
로봇 모션	Motion 〉 Move	태권도 모션 파일을 무한반복 하는 프로그램을 작성합니다. 이 프로그램은 로봇 전시를 위해 활용할 수 있습니다. (파일명: exam_motion.dts, tekwuando.dmt)
상체 모터 개별 제어	Motion 〉 Motor	각각의 상체 모터를 제어하는 프로그램입니다. 간단한 모션을 제작할 때 모션 파일 대신에 활용할 수 있습니다. (파일명: exam_motor.dts)
머리 LED 제어	Motion 〉 LED	머리의 눈, 귀, 입, 이마 LED를 제어하는 프로그램입니다. 로봇의 LED를 이용하여 다양한 효과를 줄 수 있습니다. (파일명: exam_led.dts)
전방 장애물 탐지	Sensor〉 Distance Sensor Motion 〉 TTS	로봇이 전진을 하면서, 전방 장애물 탐지 시 로봇 음성을 출력하고 멈추는 예제입니다. 이 예제를 응용하면 장애물 회피를 통한 자율주행 프로그램을 제작할 수 있습니다. (파일명: exam_distance_tts.dts)
터치 센서와 사운드 출력	Sensor〉 Touch Sensor Sensor 〉 Hand Touch Sensor Motion 〉 Sound Motion 〉 Mp3	로봇 머리와 양손 터치 센서를 이용하여 MP3 및 효과 사운드를 출력하는 프로그램입니다. 로봇의 센서와 멀티미디어를 이용하면 HRI(Human Robot Interaction) 효과를 줄 수 있습니다 (파일명: exam_touch_sound.dts, song_001.mp3)
네비게이션 주행	Sensor〉 Touch Sensor Sensor 〉 Hand Touch Sensor Motion 〉 Sound Motion 〉 Mp3	로봇 이동경로를 지정하며 네비게이션을 수행하는 프로그램을 작성합니다. (파일명: exam_navigation.dts)
리모컨 구동 제어	Communication 〉 IR Receive Motion 〉 Wheel	리모콘을 이용하여 로봇 구동부를 제어하는 프로그램입니다. (파일명: exam_remocon.dts)
비전 응용	Motion 〉 Vision Capture Sensor 〉 Vision Sensor	Color Tracking 로봇 MID에 장착 된 카메라를 이용한 비전 응용 프로그램입니다. 로봇이 빨간 물체를 인식하여 트랙킹하는 예제입니다. (파일명: exam_vision_color_tracking.dts) – 기타 응용 예제 (예제 파일만 제공) 비전 응용 프로그램은 강력한 센싱기능을 제공하며, 기타 여러 가지 응용 프로그램을 제작할 수 있습니다. 라인트레이서 (파일명: exam_vision_line_tracer.dts) Motion Detection Game (파일명: exam_vision_motion_game.dts)

4.1 로봇 모션

예제설명

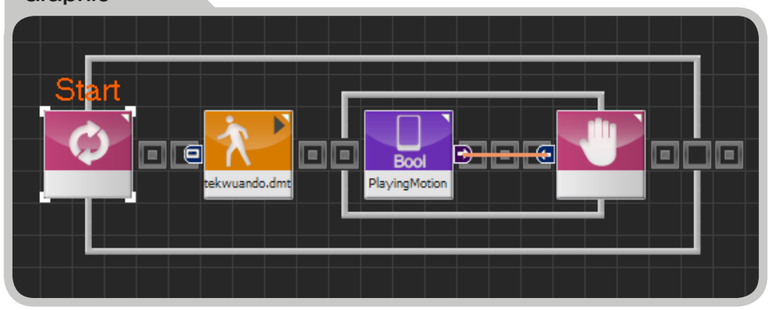
로봇 모션은 각각의 모터를 제어하여 하나의 종합적인 동작을 만드는 것입니다. 그러므로 모션을 만들기 위해 각각의 모터를 일일히 제어하여야 합니다. 하지만, 이런 방식은 모션 제작의 복잡성을 증가시킵니다. 많은 경우 모션 에디터 툴을 이용하여 모션을 쉽게 제작하는 것이 보편적입니다.

이번 예제는 모션 에디터 DR-SIM을 이용하여 제작한 모션파일(DMT)을 로봇에 다운로드하고 무한 재생합니다.
본 예제에서는 태권도 모션인 tekwuando.dmt를 사용합니다. 해당 모션 파일은 웹에서 예제와 함께 제공됩니다.

전체 프로그램

1. Loop – 무한루프
2. Move – 태권도 모션 실행
3. MID RAM (PlayingMotion) – 모션이 수행 중이며 TRUE, 아니면 FALSE를 출력
4. Wait –입력 핀의 값이 TRUE이며 해당 루프를 반복

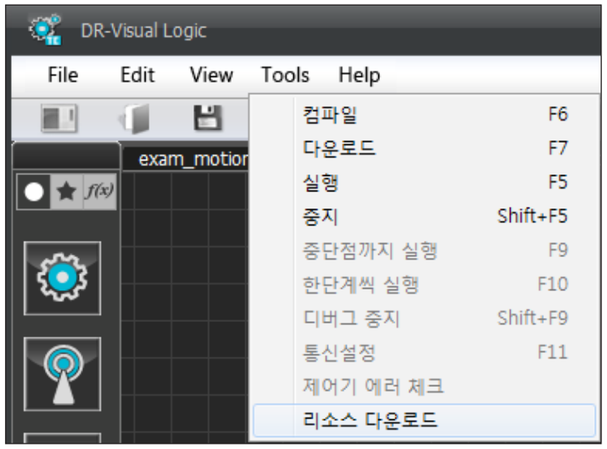
Graphic

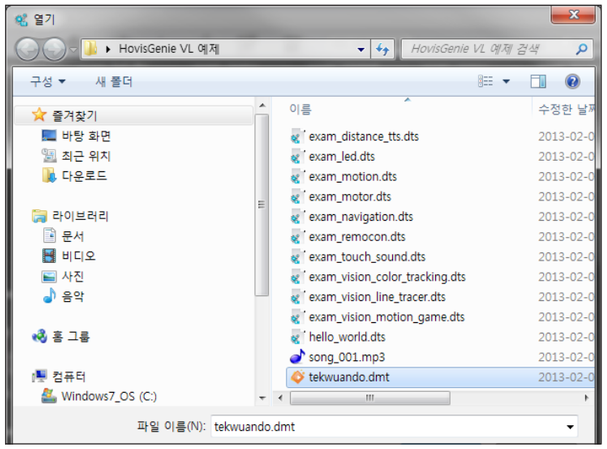


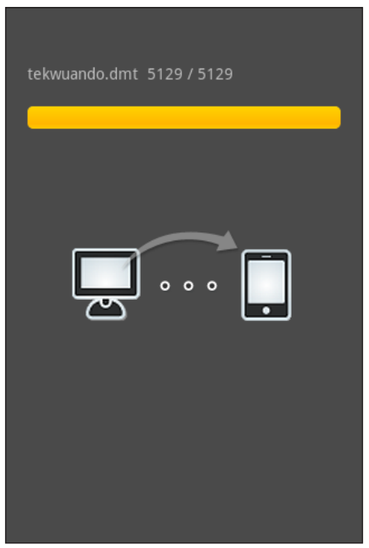
C-Like

```
void main()
{
        while( true )
        {
                mid_motion( "tekwuando.dmt", 0, 0 )
                waitwhile( MID_PlayingMotion )
        }
}
```

따라하기



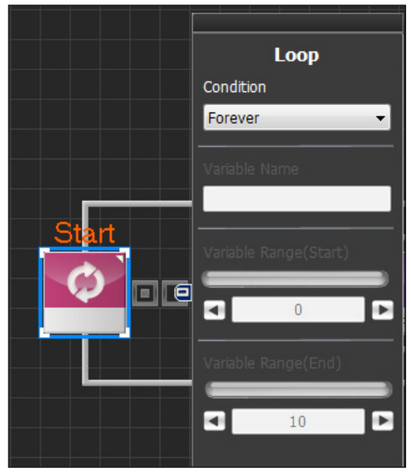




01 사전작업

DR-Visual Logic과 MID를 연결하고, 태권도 모션 파일을 다운로드합니다.

- 메뉴 〉 Tool 〉 리소스 다운로드 클릭
- 태권도 모션 파일을 선택하여 다운로드 시작



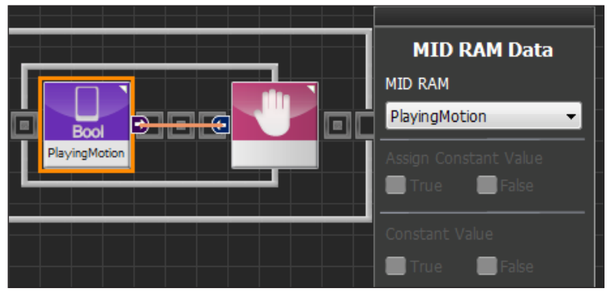
02 Loop

Loop 모듈을 선택하여 하여, Start Point에 도킹합니다.
태권도 모션을 무한 반복하므로 속성창의 Condition을 Forever로 지정합니다.

03 Move

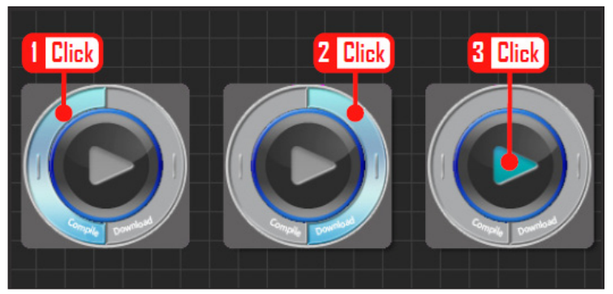
Move 모듈을 다음과 같이 배치합니다.

속성 창에 Play/Stop은 Play로 지정하고 재생할 모션 파일 이름을 지정합니다.



04 Wait 및 MID RAM Data

Wait 모듈을 이어서 배치합니다.
MID RAM Data에 Wait 모듈 안에 배치하고, 속성 창에서 PlayingMotion을 선택합니다.
PlayingMotion은 모션이 수행 중이면 True, 그렇지 않으면 False를 출력합니다. 이 출력값을 Wait 모듈 입력 핀으로 연결합니다.



05 컴파일, 다운로드, 실행

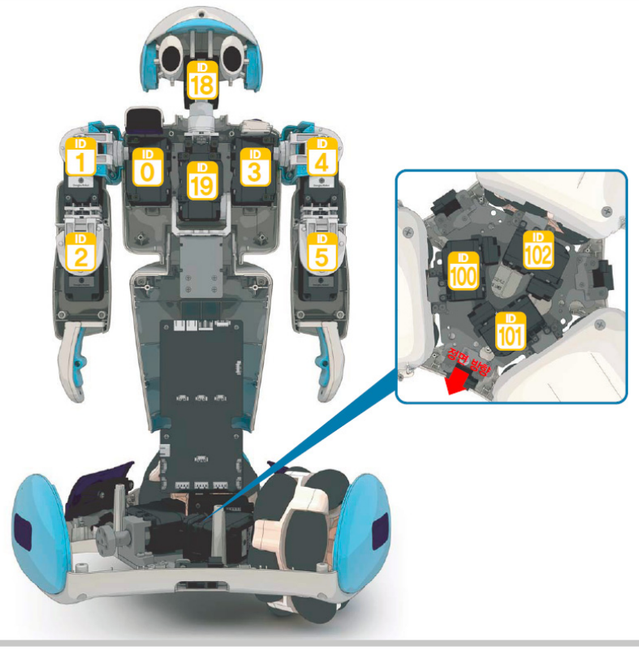
왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

4.2 상체 모터 개별 제어

예제설명

로봇에 모션을 재생하기 위해서는 모션파일을 만들어서 로봇에 수행하는 방법이 효율적입니다. 그러나 하나의 모터만 제어한다거나, 간단한 모션을 만드는 경우 모션파일을 만드는 경우보다 직접 모터를 제어하는 것이 효율적 일 수도 있습니다. 이번 예제에서는 모터를 직접 제어하여 로봇을 움직여 봅니다.

Hovis Genie에는 상체에 DRS-0101 서보 모터 8개가 있으며, 옴니휠 구동부에 DRS-0102 서보 모터 3개가 탑재되어 있습니다. 각 서보 모터의 ID 배치는 다음 그림과 같습니다.

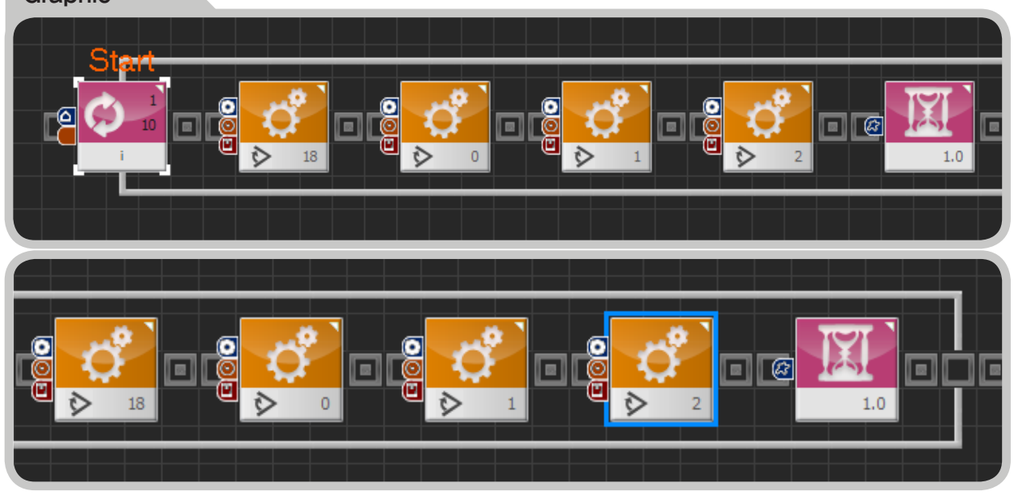


각각의 서보 모터는 위치 및 속도 제어를 통해 구동할 수 있습니다. 구동부의 100~102번 모터를 제외한 나머지 서보 모터의 경우 로봇의 관절을 구성하므로 속도제어가 무의미합니다. 그러므로 속도제어는 사용하지 않습니다. 이번 예제는 위치제어만 사용하며, 로봇이 오른손을 차렷 자세에서 펴는 자세를 10번 반복하는 예제입니다.

전체 프로그램

1. Loop – For 루프 (1~10까지 10번 반복)
2. Motor – 오른팔과 머리 서보 모터의 위치 제어 (ID: 0, 1, 2, 18) – 펴는 자세
3. Delay – 1초 딜레이
4. Motor – 오른팔과 머리 서보 모터의 위치 제어 (ID: 0, 1, 2, 18) – 차렷 자세

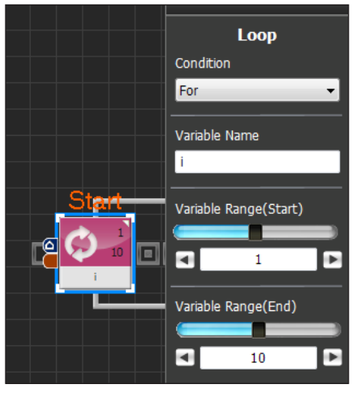
Graphic



C-Like

```
short i
void main()
{
        for( i = 1 ~ 10 )
        {
                jog( 412, 0, 18, 60 )
                jog( 512, 0, 0, 60 )
                jog( 412, 0, 1, 60 )
                jog( 512, 0, 2, 60 )
                delay( 1000 )
                jog( 512, 0, 18, 60 )
                jog( 235, 0, 0, 60 )
                jog( 235, 0, 1, 60 )
                jog( 512, 0, 2, 60 )
                delay( 1000 )
        }
}
```

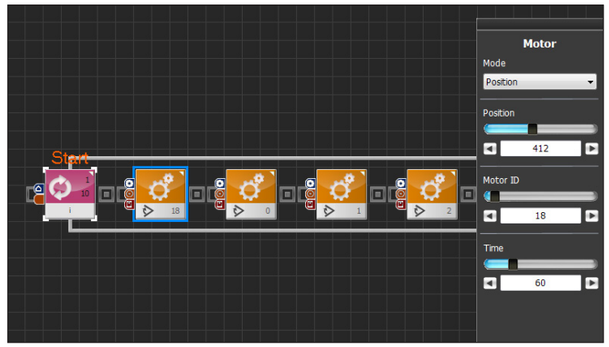
따라하기



01 Loop

Loop 모듈을 선택하고 Start Point에 도킹합니다. Loop의 모듈 안에서는 모터 제어를 통해 차렷 자세와 오른팔을 펴는 동작을 10번 반복하므로, Loop의 속성 창을 다음 그림과 같이 지정합니다.

Condition : For
Variable Name : i
Variable Range(Start) : 1
Variable Range(End) : 10



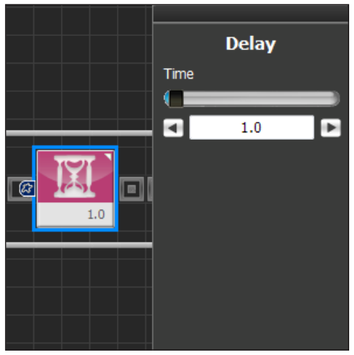
02 Motor

모터의 위치제어를 통해 오른팔을 펴는 자세를 취하도록 합니다. 오른팔 모터의 ID는 각각 0번, 1번, 2번이며, 추가적으로 18번 머리모터도 함께 제어합니다.

Motor 모듈을 그림과 같이 추가하고 각 Motor 모듈의 속상 창에서 다음과 같이 값을 지정합니다.

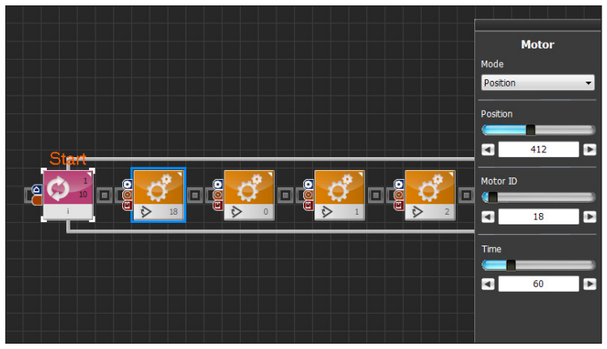
Mode : 위치 및 속도 제어를 선택
Position : 위치 값 (0~1023)
　　가운데 영점 (512)
Motor ID : 모터의 ID
Time : 1 = 11.2ms
ex) 60　11.2ms = 672ms

Mode	Pos	Pos	Pos	Pos
Position	412	512	412	512
ID	18	0	1	2
Time	60	60	60	60



03 Delay

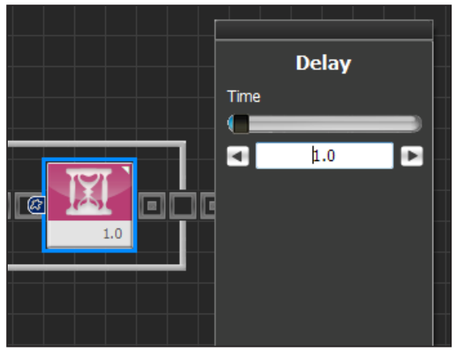
1초동안 프로그램을 멈춥니다.



04 Motor

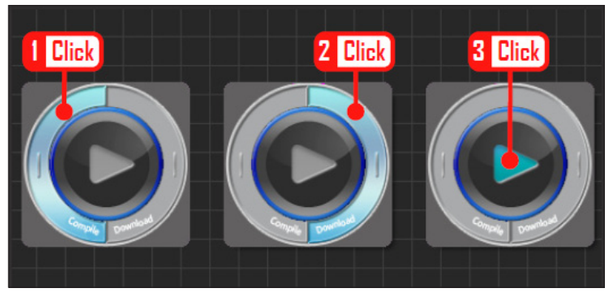
모터의 위치제어를 통해 오른팔을 차렷 자세를 취하도록 합니다. 오른팔의 모터의 ID는 각각 0번, 1번, 2번이며, 추가적으로 18번 머리모터도 함께 제어합니다.

Mode	Pos	Pos	Pos	Pos
Position	512	235	235	512
ID	18	0	1	2
Time	60	60	60	60



05 Delay

1초동안 프로그램을 멈춥니다.



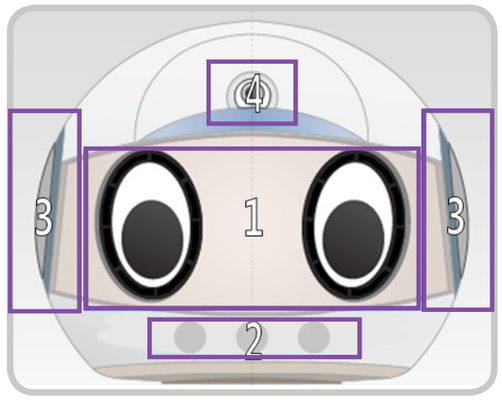
06 컴파일, 다운로드, 실행

왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

4.3 머리 LED 제어

예제설명

Hovis Genie의 머리에는 LED를 제어할 수 있는 보드가 있어 눈, 입, 귀, 그리고 이마의 LED를 키고 끌 수 있습니다. 구체적으로 LED의 구성은 다음과 같습니다.



1번 위치는 눈 LED입니다. 한쪽 눈에는 빨간색과 파란색 LED가 각 8개씩 배치되어 16개의 LED를 가지며, 양쪽 눈에는 총 32개의 LED가 달려있습니다. 눈 LED의 경우 파란색, 빨간색을 각각 설정할 수 있으며, 파란색과 빨간색을 동시에 켜서 보라색 효과를 낼 수 있습니다.
2번 위치는 입 LED입니다. 입 LED는 총 3개로 구성되어 있습니다. 3번은 귀 LED이며 양쪽에 있습니다. 귀의 경우 한쪽 귀에 각 4개씩 LED를 가지나, 모두 키거나 끄는 등 일괄적인 제어만 가능합니다. 4번 위치는 이마의 휘도 LED로 밝기의 강도를 조절할 수 있는 LED입니다.

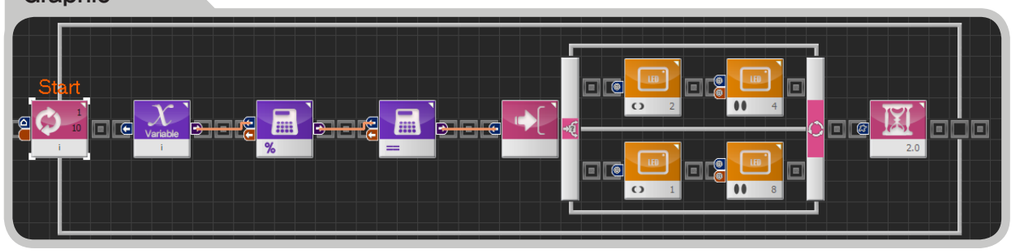
LED 모듈은 눈, 귀, 입, 그리고 이마의 LED를 효과적으로 제어하기 위한 기능을 제공합니다. LED 모듈은 로봇의 감정 표현 및 시각 효과에 유용하게 사용할 수 있습니다.

이번 예제 프로그램에서는 Hovis Genie의 머리 LED 제어를 합니다. 이 프로그램은 루프를 10번 반복하면서 눈과 귀의 LED를 번갈아 키고 끄는 프로그램입니다.

전체 프로그램

1. Loop – For 루프 i (1~10까지 10번 반복)
2. If-Else – i 값을 2로 나눈 나머지 값이 0이면 TRUE, 아니면 FALSE
3. LED – 눈과 귀의 LED를 번갈아 가며 제어
4. Delay – 2초간 딜레이

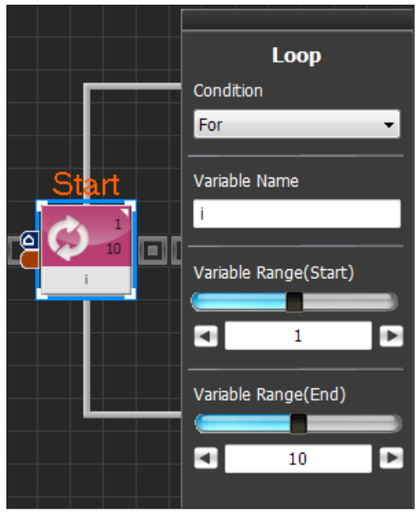
Graphic



C-Like

```
short i
void main()
{
        for( i = 1 ~ 10 )
        {
                if( ( ( i % 2 ) == 0 ) )
                {
                        mid_led_ear( 2 )
                        mid_led_eye( 4, 2 )
                }
                else
                {
                        mid_led_ear( 1 )
                        mid_led_eye( 8, 2 )
                }
                delay( 2000 )
        }
}
```

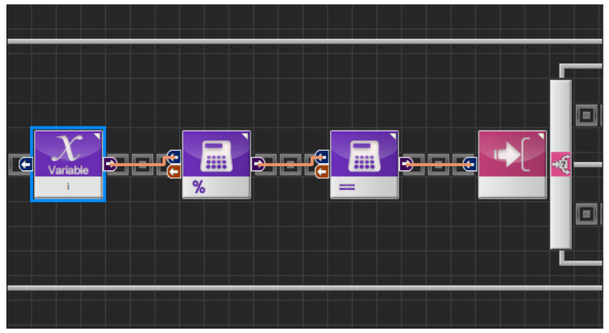

따라하기



01 Loop

Loop 모듈을 선택하고 Start Point에 도킹합니다. Loop는 10번 반복하므로, Loop의 속성 창을 다음 그림과 같이 지정합니다.

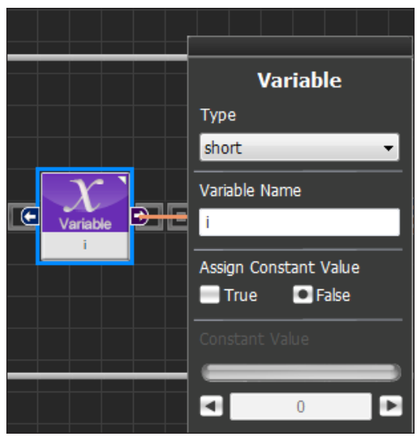
Condition : For
Variable Name : i
Variable Range(Start) : 1
Variable Range(End) : 10



02 If-Else

If-Else 모듈은 True 또는 False의 값을 입력 핀을 통해 받아들입니다.

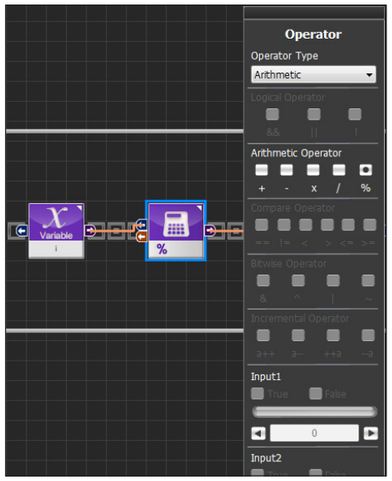
01의 Loop가 10번을 반복하며 증가하는 변수 값 i를 이용하여 2로 나눈 값이 0인지 비교하여 If-Else의 입력 핀에 연결합니다.



02-1 If-Else 조건문 1

Variable 모듈
Type: short
Variable Name: i
Assign Constant Value: False

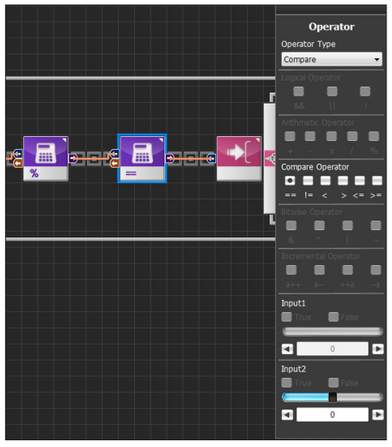
01에서 지정한 Loop의 변수 i가 short형이므로 Type을 short으로 지정합니다. Loop의 변수 i를 사용하는 것이므로 Variable Name은 i입니다. Assign Constant Value가 True이면 지정하는 값이 대입되므로 False로 지정합니다.



02-2 If-Else 조건문 2

Operator 모듈
Operator Type: Arithmetic
Arithmetic Operator: %
Input2: 2

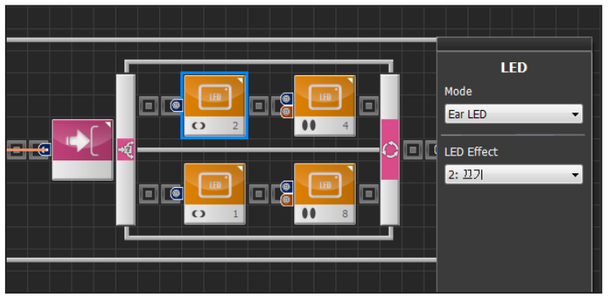
02-1의 i값을 Input1로 입력하고, Input2를 2로 지정합니다. Arithmetic Operator를 %로 지정하여 Input 1의 값을 Input 2로 나눈 나머지값을 출력 핀을 통해 다른 모듈의 입력 핀으로 연결합니다.



02-3 If-Else 조건문 3

Operator 모듈
Operator Type: Compare
Arithmetic Operator: ==
Input2: 0

02-2의 연산 값을 Input1로 받고, Input2는 0으로 지정합니다. 두 수를 비교하여 값이 같으면 True, 틀리면 False를 출력합니다. 이 출력 값은 최종적으로 If-Else의 입력 핀으로 연결됩니다.



03 LED

LED 모듈을 통해 눈과 귀의 LED를 번갈아 킵니다.

Mode: 눈(Eye), 귀(Ear), 입(Mouth), 또는 이마(Brow) 선택
LED Effect: 각 모드에 따라 효과 선택

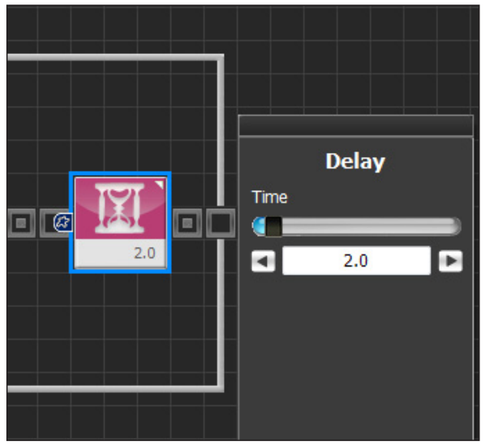
LED 모듈 각각의 속성 창 설정은 다음과 같습니다.

[TRUE]

	첫 번째	두 번째
Mode	Ear LED	Eye LED
LED Effect	2:끄기	4:전체 켜기

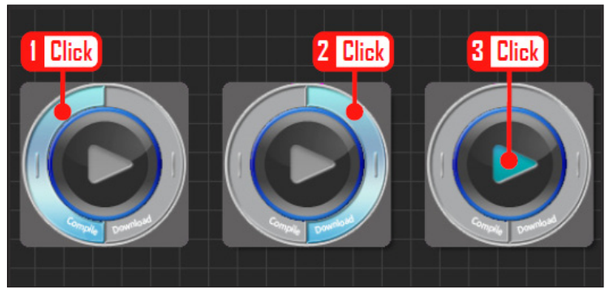
[FALSE]

	첫 번째	두 번째
Mode	Ear LED	Eye LED
LED Effect	1:켜기	8:끄기



04 Delay

2초간 프로그램을 멈춥니다. LED를 2초마다 한 번씩 번갈아 켜주는 효과를 주기 위함입니다.



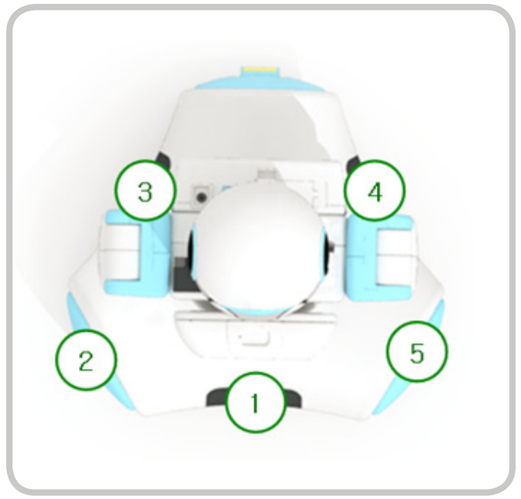
05 컴파일, 다운로드, 실행

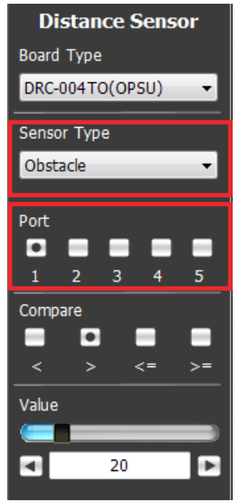
왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

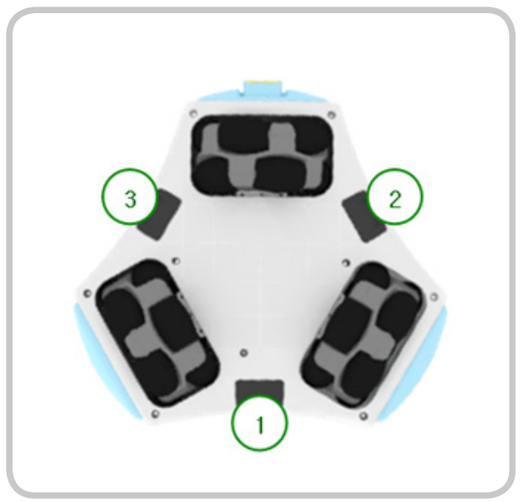
4.4 전방 장애물 탐지

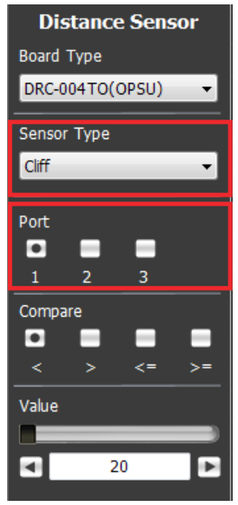
예제설명

Hovis Genie는 주변환경을 인지하기 위해 8개의 PSD 센서가 있으며 5개는 거리 탐지를 위해 사용되며, 3개는 하단 감지를 위해 사용됩니다. DR-Visual Logic의 Distance Sensor 모듈의 속성 창에서 각 센서의 지정은 다음과 같습니다.









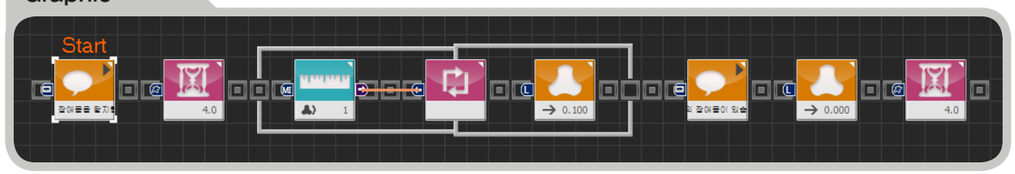
PSD 센서는 주변의 거리 정보를 최대 80cm까지 획득하여 장애물을 인지합니다. 로봇은 PSD 센서를 통해 장애물을 회피하거나 여러 주행 전략을 수립할 수 있습니다.

이번 예제는 로봇이 전방을 향해 전진을 하다가 장애물을 탐지하면 탐지 사실을 로봇 음성으로 이야기하고 멈춰서는 프로그램입니다.

전체 프로그램

1. TTS – 장애물 탐지 시작 안내 멘트
2. Delay – 4초간 딜레이
3. While – 입력 핀이 TRUE이면 루프 무한 반복
   조건 : Distance Sensor – Obstacle(1번)이 20cm보다 크면 TRUE, 작으면 FALSE
   - TRUE 시 : Wheel – 로봇 전진
   - FALSE 시 : 루프를 벗어남
4. TTS – 장애물 탐지 안내 멘트
5. Wheel – 구동 바퀴 정지
6. Delay – 4초간 딜레이

Graphic



C-Like

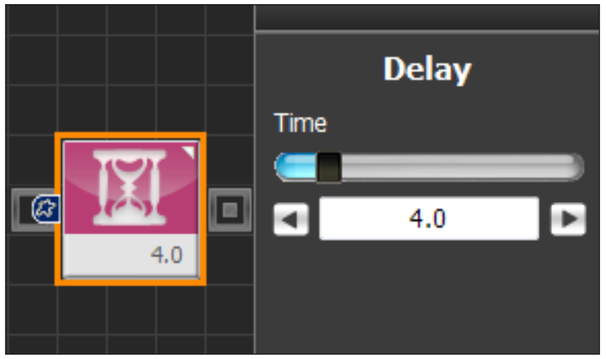
```
void main()
{
        mid_tts_play( "전방의 장애물을 탐지합니다." )
        delay( 4000 )
        while( ( OPSU_ObstaclePSD1 > 20 ) )
        {
                mid_wheel_linear( 0.100, 0, false )
        }
        mid_tts_play( "전방의 장애물이 있습니다." )
        mid_wheel_linear( 0.000, 0, false )
        delay( 4000 )
}
```

따라하기

01 TTS

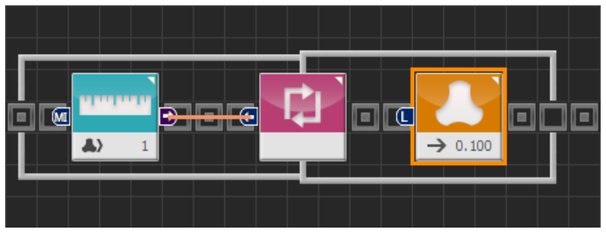
TTS로 전방 장애물 탐지 안내 메시지를 출력합니다.

– Play/Stop : Play
– TTS Contents
"전방의 장애물을 탐지합니다."



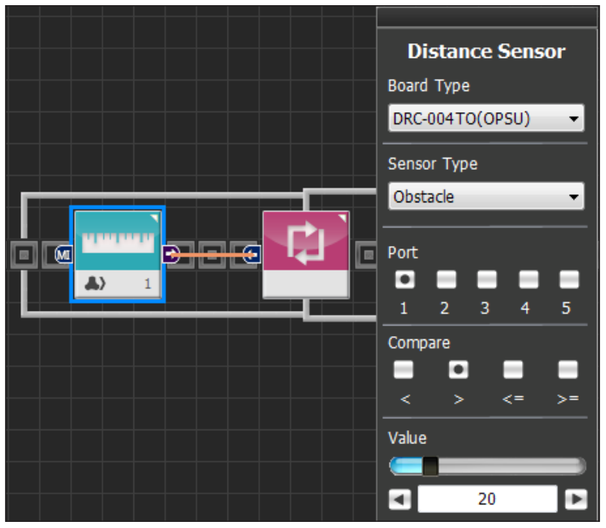
02 Delay

TTS 모듈은 로봇 음성이 끝날 때까지 기다리지 않습니다.
즉, 로봇 음성이 출력되고 있는 상황에서도 다음 모듈들의 명령을 수행하므로 로봇 음성을 충분히 들을 수 있는 시간만큼 프로그램을 딜레이 합니다.



03 While

While 모듈은 입력 핀이 TRUE일 경우 루프를 반복하고, FASLE이면 루프를 벗어납니다.
Distance Sensor 모듈이 입력 핀에 값을 주며, 그 값이 TRUE이면 로봇을 전진시킵니다.

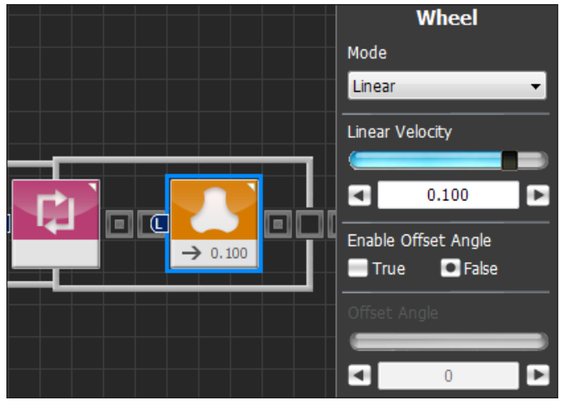


03–1 While 모듈 조건

Distance Sensor는 지정한 PSD 센서의 값을 읽어오는 모듈입니다.

– Board Type: DRC–004TO(OPSU)
– Sensor Type: Obstacle
– Port : 1
– Compare : 〉
– Value : 20

이 모듈은 전방의 PSD 센서 값을 읽어 그 거리 값이 20cm 보다 크면 TRUE, 작으면 FALSE를 출력 핀으로 내보냅니다.

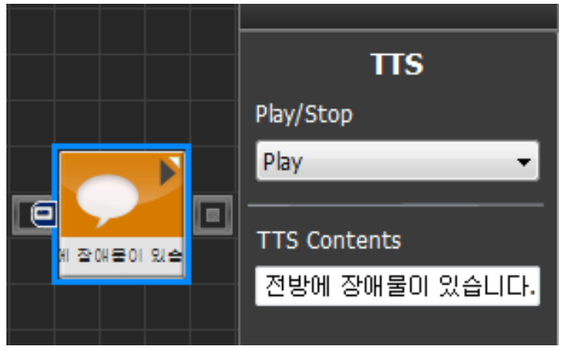


03–2 While 모듈 루프

While 모듈 루프 안의 Wheel 모듈은 03–1의 조건이 FALSE가 아닐 때까지 반복적으로 수행됩니다.

이 Wheel 모듈은 로봇을 전방을 향해 전진시킵니다. Wheel 모듈의 속성 창의 값을 다음과 같이 지정합니다.

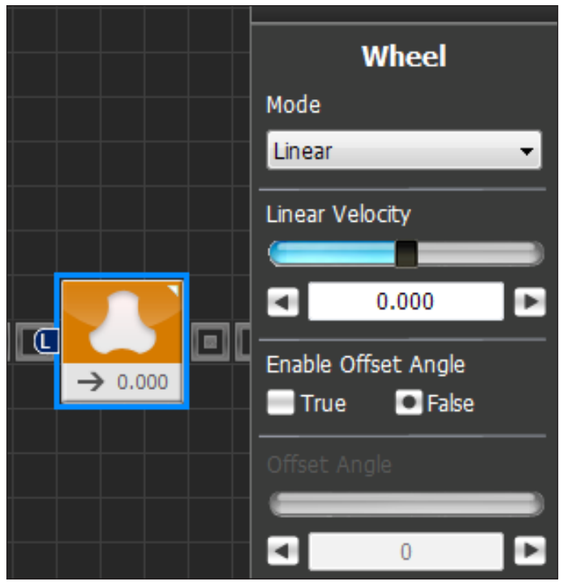
– Mode : Linear
– Linear Velocity: 0.100
– Enable Offset Angle : False



04 TTS

TTS로 전방 장애물 탐지 사실을 출력합니다.

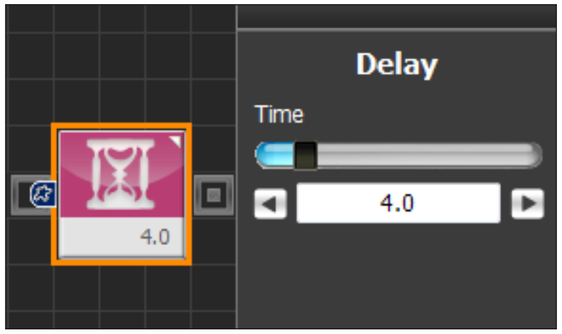
– Play/Stop : Play
– TTS Contents
“전방에 장애물을 탐지합니다.”



05 Wheel

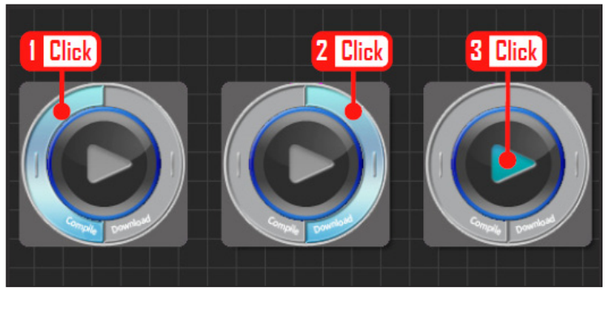
이 Wheel 모듈은 로봇의 구동부를 정지시킵니다. Wheel 모듈의 속성 창의 값을 다음과 같이 지정합니다.

– Mode : Linear
– Linear Velocity: 0.000
– Enable Offset Angle : False



06 Delay

TTS 모듈은 로봇 음성이 끝날 때까지 기다렸다가 다음 모듈을 진행하지 않습니다. 즉, 로봇 음성이 출력되고 있는 상황에서도 다음 모듈들의 명령을 수행하므로 로봇 음성을 충분히 들을 수 있는 시간만큼 프로그램을 딜레이 합니다.



07 컴파일, 다운로드, 실행

왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

4.5 터치 센서와 사운드 출력

예제설명

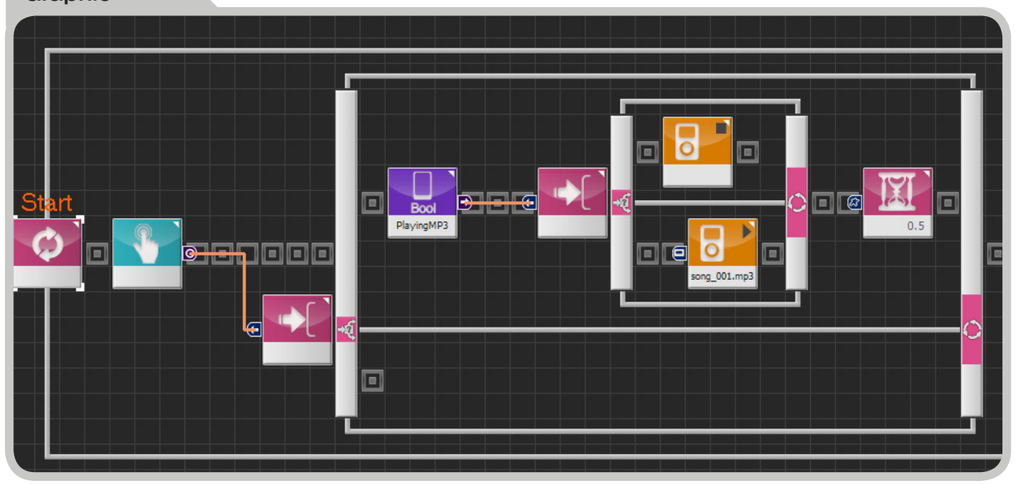
터치 센서는 로봇과 유저와의 상호작용을 위하여 활용할 수 있습니다. 로봇의 터치 센서는 총 3개가 있으며, 센서의 위치는 양 손바닥 그리고 머리입니다.

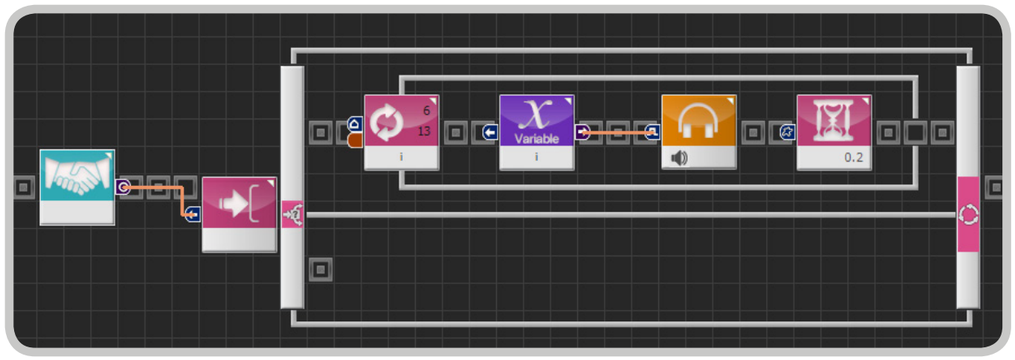
이번 예제는 HOVIS Genie의 3개의 터치 센서를 이용하여 사운드 출력을 하는 프로그램입니다. 머리를 터치하면 MP3를 재생을 시작하고 이미 재생 중이면 재생을 멈춥니다. 그리고 양손을 터치하면 “도레미파솔라시도”와 “도시라솔파미레도”를 재생합니다.

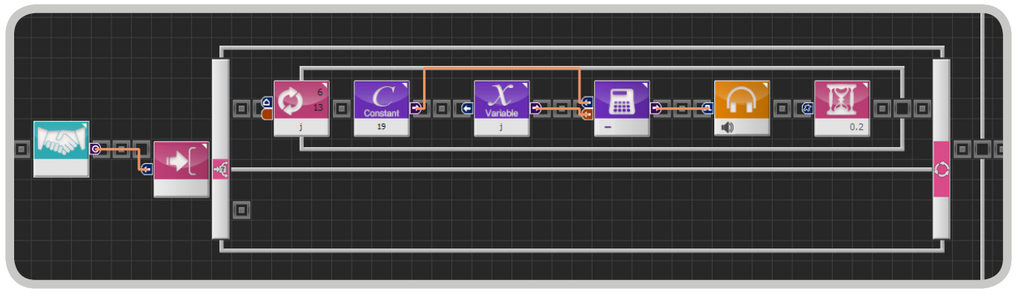
전체 프로그램

1. Loop – 무한루프
2. Touch Sensor – 머리 터치 여부 검사
   - TRUE 이면: MID RAM(PlayingMP3) 변수로 MP3 재생 여부 검사
   - PlayingMP3가 TRUE 시: MP3 재생 중지
   - PlayingMP3가 FALSE 시: MP3 재생 시작
3. Hand Touch Sensor(Right) – 오른손 터치 여부 검사
   - TRUE 이면: Sound 모듈로 “도레미파솔라시도” 재생
4. Hand Touch Sensor(Left) – 왼손 터치 여부 검사
   - TRUE 이면: Sound 모듈로 “도시라솔파미레도” 재생

Graphic



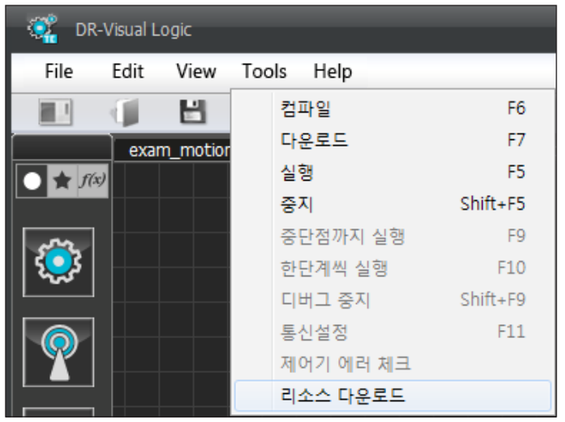




C-Like

```
short i, j
void main()
{
	while( true )
	{
		if( ( MPSU_TouchStatus ) )
		{
			if( MID_PlayingMP3 )
			{
				mid_mp3_stop()
			}
			else
			{
				mid_mp3_play( "song_001.mp3" )
			}
			delay( 500 )
		}
		else
		{
		}
		if( ( SERVO_GPIO1[2]==0 ) )
		{
			for( i = 6 ~ 13 )
			{
				mid_sound( i )
				delay( 200 )
			}
		}
		else
		{
		}
		if( ( SERVO_GPIO1[5]==0 ) )
		{
			for( j = 6 ~ 13 )
			{
				mid_sound( ( 19 - j ) )
				delay( 200 )
			}
		}
		else
		{
		}
	}
}
```

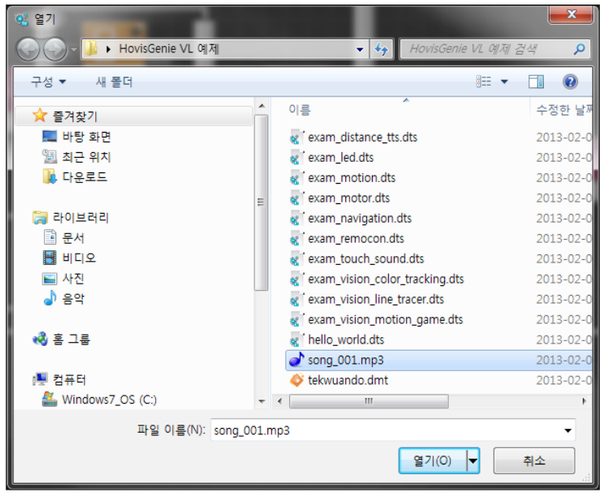
따라하기

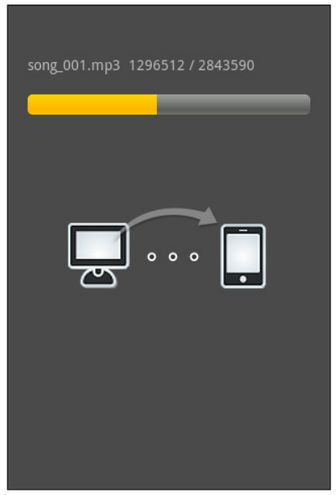


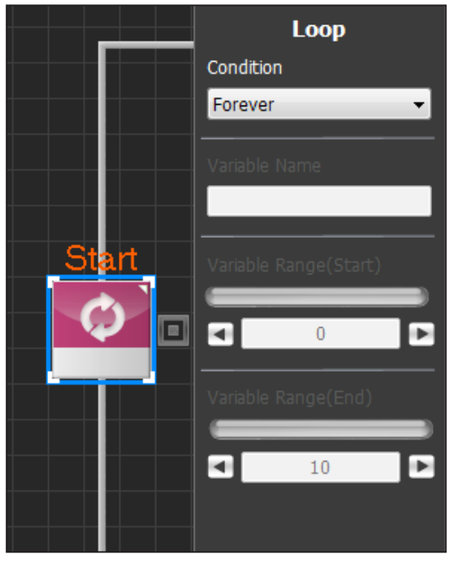
01 사전작업

DR-Visual Logic과 MID를 연결하고, 재생할 MP3 파일을 MID로 다운로드합니다.

- 메뉴 〉 Tool 〉 리소스 다운로드 클릭
- 재생할 MP3 파일을 선택하여 다운로드 시작

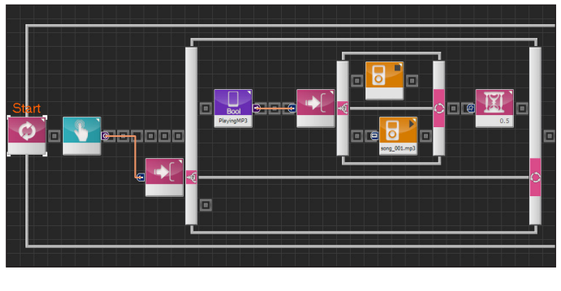






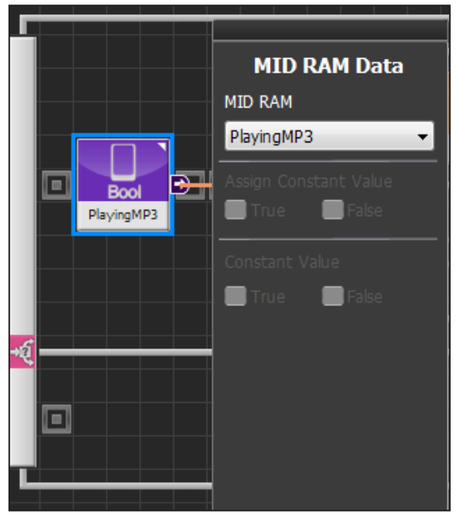
02 Loop

Loop 모듈을 선택하여 하여, Start Point에 도킹합니다
무한반복하며 머리와 양손터치 여부를 검사하므로 속성창의 Condition을 Forever로 지정합니다.



03 Touch Sensor (머리 터치 여부 검사)

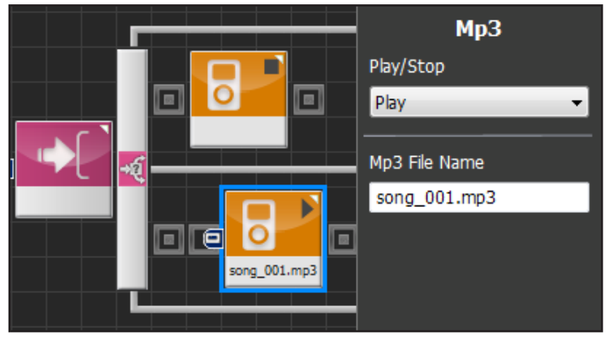
Touch Sensor 모듈을 배치하고, Touch Sensor 출력 핀과 If-Else 모듈의 출력 핀을 연결 선을 통해 연결합니다.



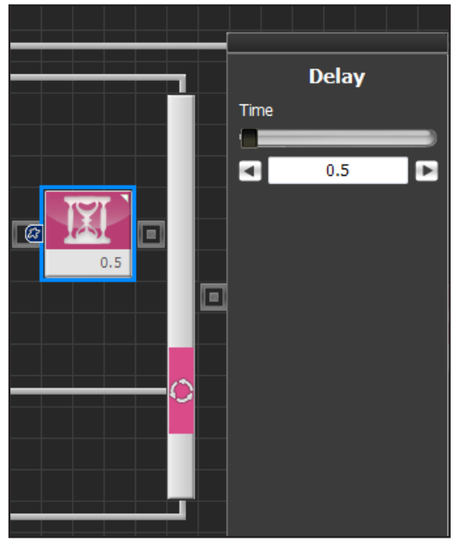
MID RAM Data 모듈의 PlayingMP3 변수는 MP3가 재생 중이면 TRUE를, 아니면 FALSE를 출력합니다

If-Else 모듈을 배치하고 MP3 모듈을 배치합니다.

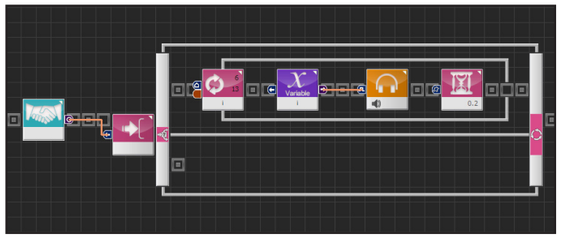
TRUE 부분의 MP3 모듈에는 Play/Stop속성을 Stop으로 지정합니다. Stop은 MP3 파일에 관계없이 현재 재생되는 MP3파일을 중지합니다.

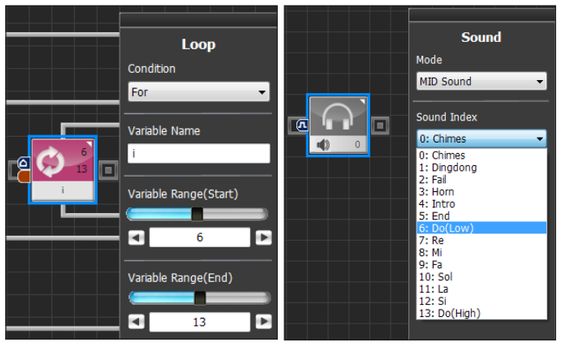


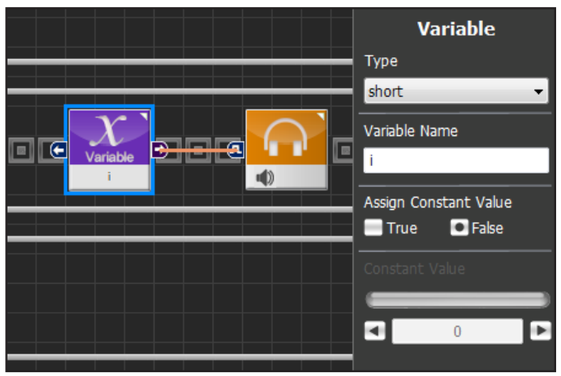
FALSE 부분의 MP3 모듈은 재생할 MP3파일의 이름을 지정합니다.
- Play/Stop : Play
- Mp3 File Name : song_001.mp3

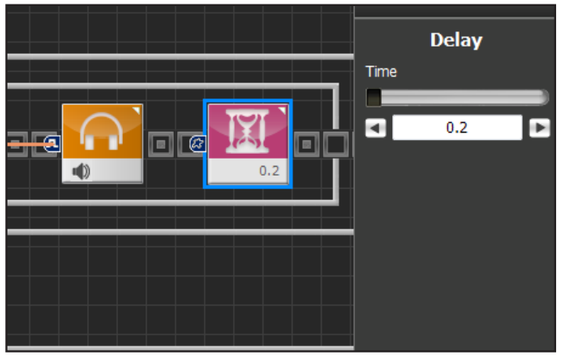


마지막으로, Delay 모듈을 배치합니다. Delay가 없으면 유저가 한번 터치하여도, 머리를 두 번 터치 한 것처럼 인식하여 MP3 재생을 시작하자마자 중지될 수 있습니다.
이는 Touch Sensor 모듈이 Loop 모듈 안에 배치되어 있기 때문입니다.









04 Hand Touch Sensor (오른손)

Hand Touch Sensor 모듈을 배치하고 속성 창에서 Right를 지정하여 오른손 터치 여부를 읽습니다.

이 Hand Touch Sesnor는 오른손 스위치를 누르고 있을 때 TRUE를, 그렇지 않으면 FALSE를 출력합니다.

Hand Touch Sensor 출력 핀과 If-Else 모듈의 출력 핀을 연결 선을 통해 연결합니다.

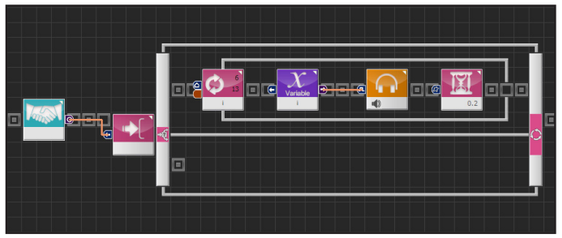
오른손이 터치되면 피아노 "도레미파솔라시도"를 연주하기 위하여 루프가 필요합니다.

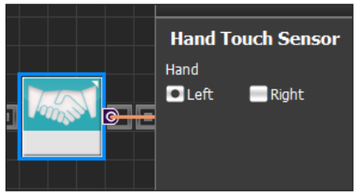
Loop 모듈을 추가합니다.
- Condition: For
- Variable Name: i
- Variable Range(Start): 6
- Variable Range(End): 13

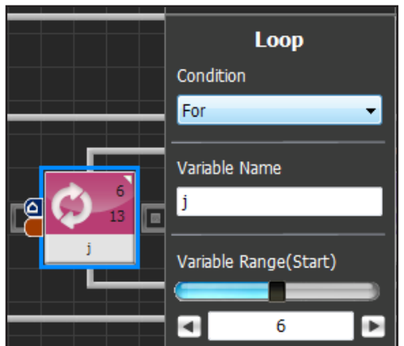
범위가 6~13인 이유는 Sound 모듈의 피아노 음의 Sound Index가 6~13이기 때문입니다.

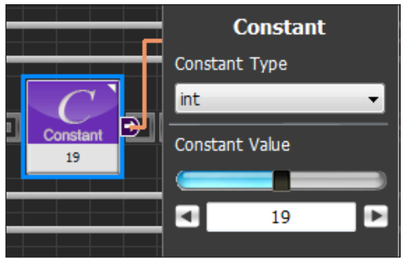
Variable 모듈을 이용하여 Loop 모듈에서 지정한 i의 값을 이용합니다. Loop 모듈에서 사용한 i변수는 short형 이므로 Type을 short으로 지정합니다.

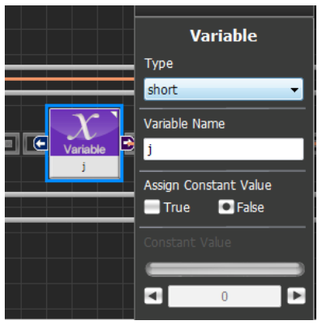
Delay 모듈을 이용하여 피아노 음을 재생할 때 재생 간격을 0.2초로 지정합니다. 이 딜레이를 주지 않게 되면 거의 하나의 소리가 재생되는 것처럼 들리므로 반드시 딜레이를 주도록 합니다.











05 Hand Touch Sensor (왼손)

Hand Touch Sensor 모듈을 배치하고 속성 창에서 Left를 지정하여 왼손 터치 여부를 읽습니다.

이 Hand Touch Sesnor는 왼손 스위치를 누르고 있을 때 TRUE를, 그렇지 않으면 FALSE를 출력합니다.

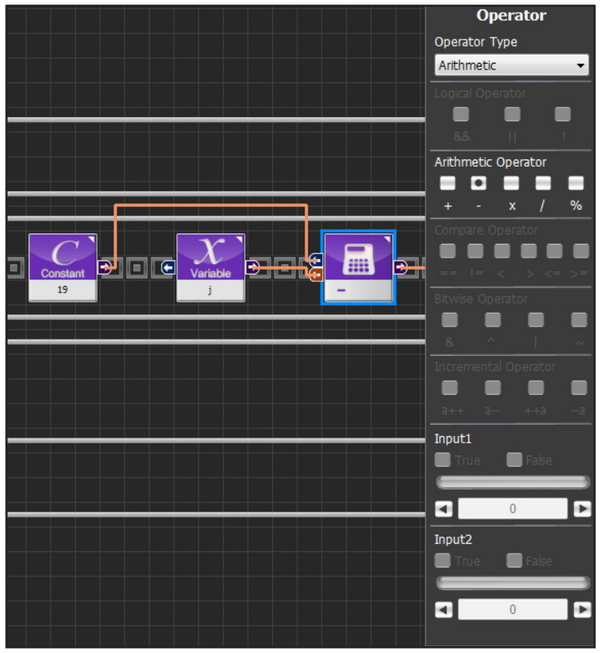
Hand Touch Sensor 출력 핀과 If-Else 모듈의 출력 핀을 연결 선을 통해 연결합니다.

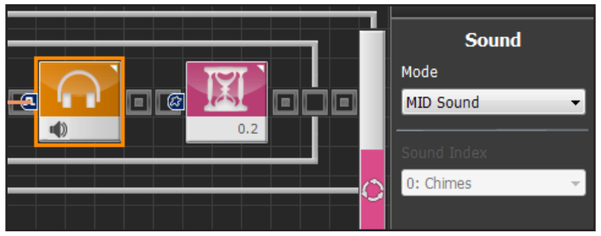
왼손이 터치되면 피아노 "도시라솔파미레도"를 연주하기 위하여 루프가 필요합니다.

Loop 모듈을 추가합니다.
- Condition: For
- Variable Name: j
- Variable Range(Start): 6
- Variable Range(End): 13

오른손 Sound 모듈의 경우 6부터 13까지의 값이 루프를 돌며 Sound 모듈에 입력되었지만, 왼손의 경우는 반대로 13부터 6까지 Sound 모듈에 입력되어야 합니다.

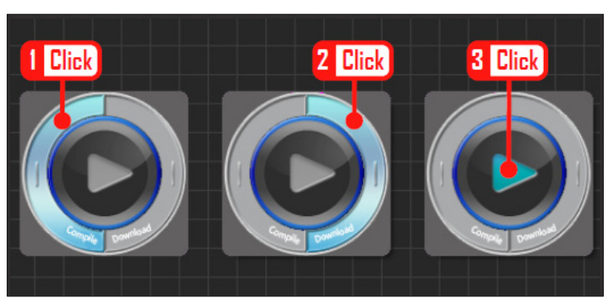
6~13을 13~6으로 변환하기 위해서는 19에서 Loop 모듈에서 사용한 j변수를 빼주면 됩니다. j = 6일 때, 19-6=13이며, j=13일 때, 19-13=6 입니다.





계산 된 값을 Sound 모듈에 연결선을 통해 입력합니다.

마지막으로, Delay 모듈을 이용하여 피아노 음을 재생할 때 재생 간격을 0.2초로 지정합니다. 이 딜레이를 주지 않게 되면 거의 하나의 소리가 재생되는 것처럼 들리므로 반드시 딜레이를 주도록 합니다.



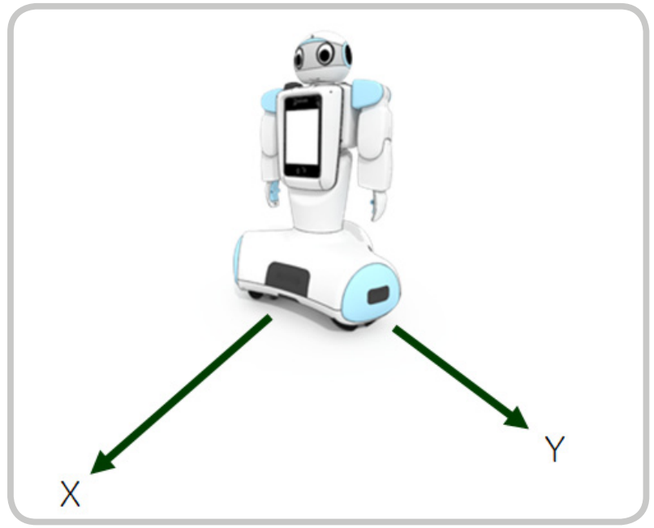
06 컴파일, 다운로드, 실행

왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

4.6 네비게이션 주행

예제설명

Hovis Genie의 네비게이션 주행은 로봇 중심부를 기준으로 로봇의 정면 방향을 X축으로 하는 2차원 평면상에서 좌표를 통해 로봇을 주행하는 것을 말합니다. 로봇 좌표의 단위는 m이며, 로봇 좌표 축을 그림으로 나타내면 다음과 같습니다.



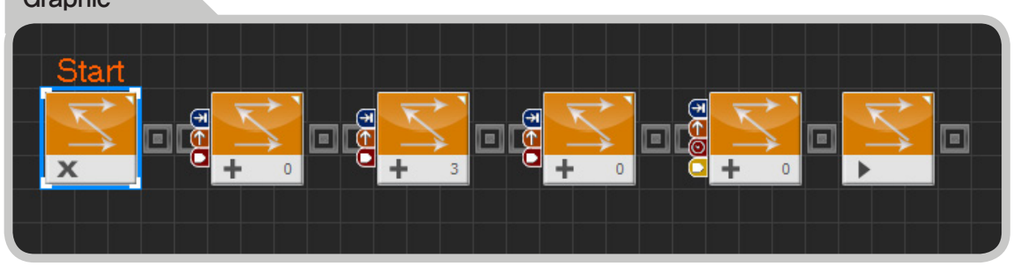
네비게이션 주행은 바퀴의 회전량을 이용하여 로봇의 좌표를 계산하므로 오차가 있으며, 구동하면 할수록 그 오차가 누적됩니다. 이러한 오차로 인하여, DR-Visual Logic의 네비게이션 모듈은 개략적인 네비게이션을 구현하는데 응용할 수 있습니다. 참고로 로봇 네비게이션을 정밀하게 하기 위해 로봇이 스스로 위치를 보정하는 기술은 활발히 연구되는 분야입니다.

이번 예제는 로봇이 위치한 현 위치를 기준으로 다이아몬드 주행을 수행하는 프로그램입니다.

전체 프로그램

1. Navigation – 초기화
2. Navigation – 방문 지점 추가
3. Navigation – 네비게이션 시작

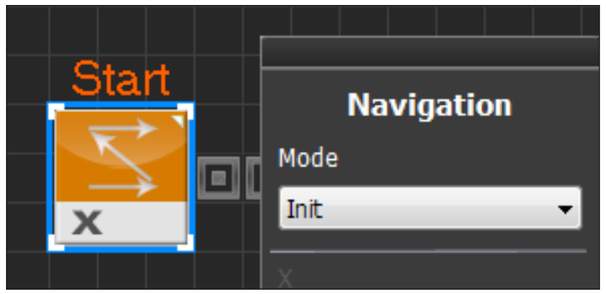
Graphic



C-Like

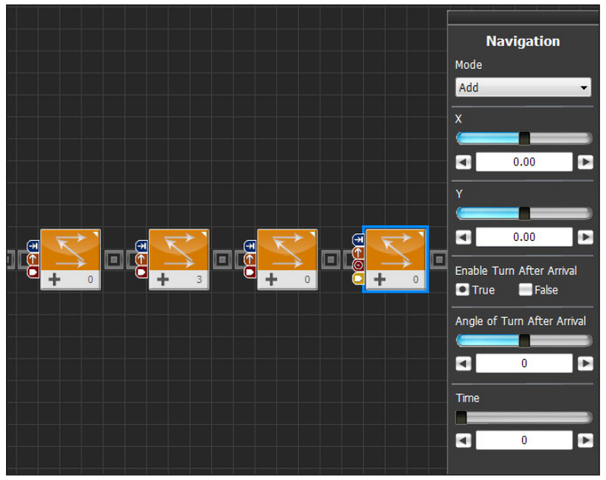
```
void main()
{
        mid_navi_init()
        mid_navi_add( 0.50, 0.50, 0, false, 0 )
        mid_navi_add( 1.00, 0.00, 0, false, 3 )
        mid_navi_add( 0.50, -0.50, 0, false, 0 )
        mid_navi_add( 0.00, 0.00, 0, true, 0 )
        mid_navi_start()
}
```

따라하기



01 네비게이션 초기화

Navigation 모듈을 추가하고 속성 창에서 Mode를 Init으로 설정합니다. 네비게이션 초기화를 하면 로봇이 현 위치를 기준으로 정면을 X축으로 하는 가상 좌표축을 구축합니다. 즉 현 위치를 (0, 0)으로 설정합니다.



02 네비게이션 지점 추가

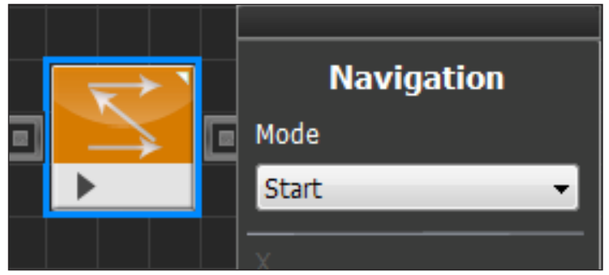
로봇이 다이아몬드를 그리며 주행하기 위해서 로봇이 방문할 지점을 추가합니다.

4개의 Navigation 모듈을 추가하고 각 모듈에 대한 설정을 합니다.

	Mode	X	Y	E.	A.	Time
1	Add	0.5	0.5	F	–	0
2	Add	1.0	0.0	F	–	3
3	Add	0.5	–0.5	F	–	0
4	Add	0.0	0.0	T	0	0

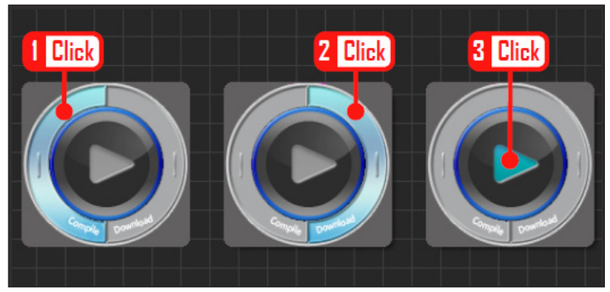
– E. (Enable Turn After Arrival)
– A. (Angle of Turn After Arrival)

Navigation 속성 창
– X: X좌표 (단위:m)
– Y: Y좌표 (단위:m)
– Enable Turn After Arrival
  로봇이 방문 지점 도착 후 바라보고 있는 방향을 지정할 지 여부
– Angle of Turn After Arrival
  로봇이 방문 지점 도착 후 바라볼 방향을 지정 (–180~180도)
– Time
  로봇이 방문 지점 도착 후 대기시간



03 네비게이션 시작

마지막으로 Navigation 모듈을 추가하고 Mode를 Start로 지정합니다.



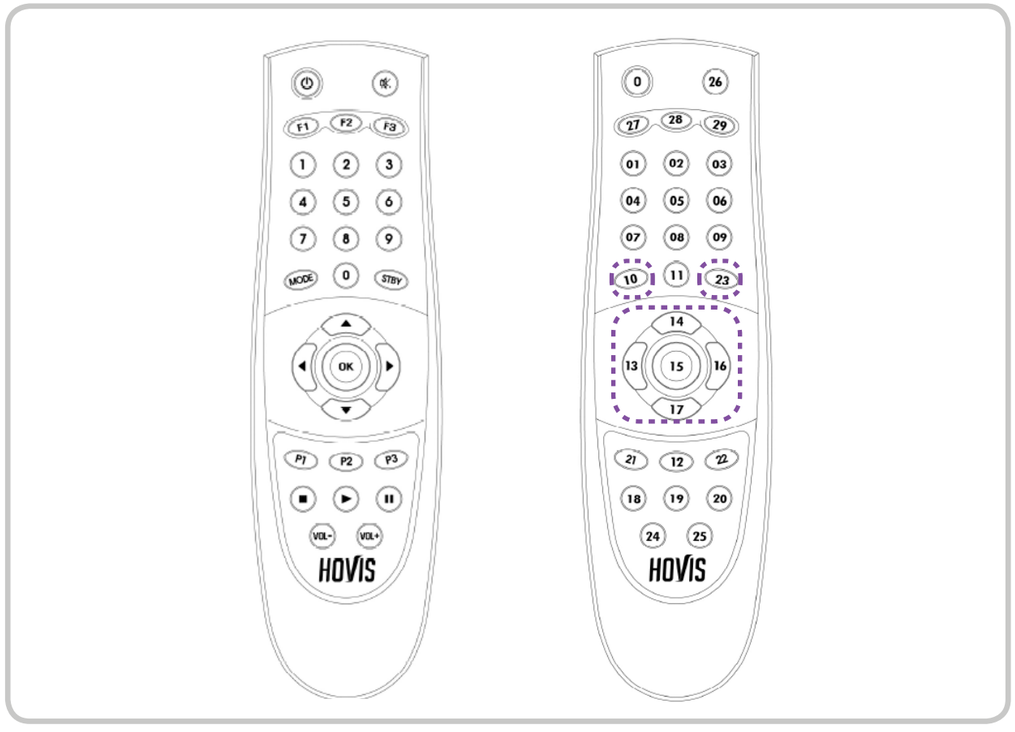
04 컴파일, 다운로드, 실행

왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

4.7 리모컨 구동 제어

예제설명

DR-Visual Logic의 IRReceive 모듈을 이용하면 Hovis Genie의 액세서리로 제공되는 리모컨을 사용하여 여러 가지 프로그램을 작성할 수 있습니다. 리모컨의 각각 버튼에 할당 된 번호는 다음과 같습니다.



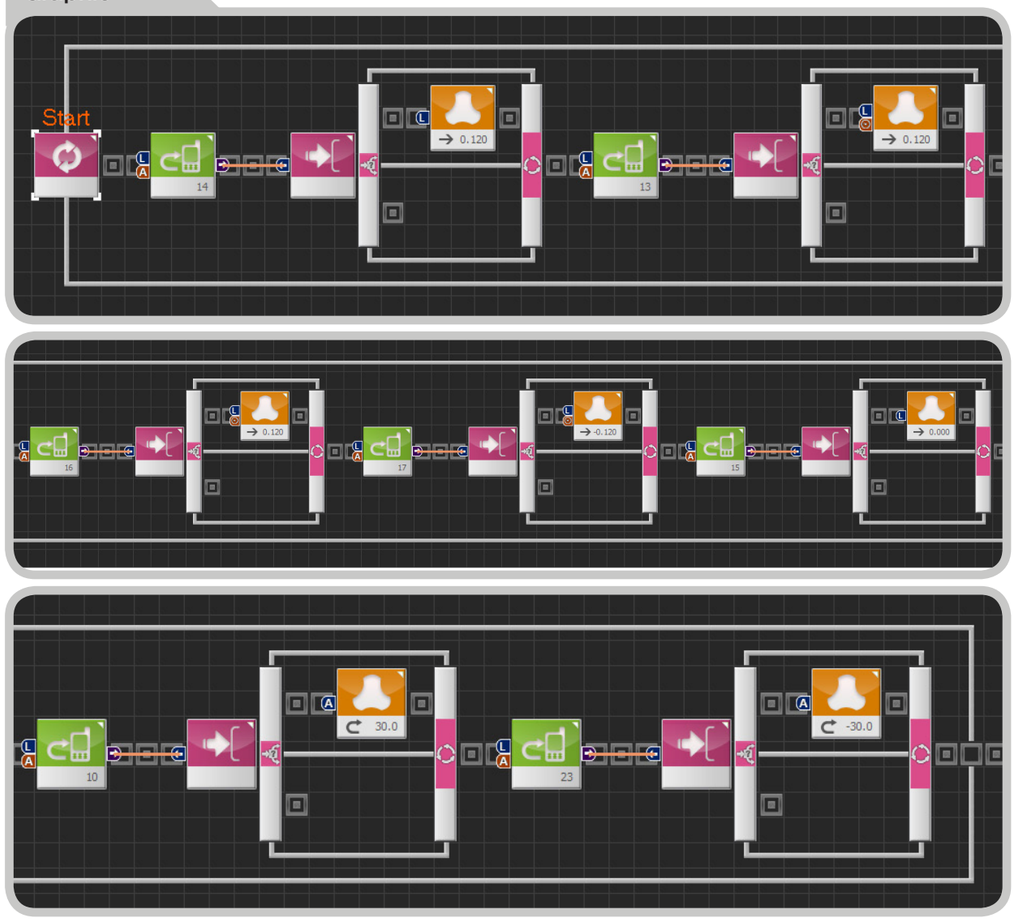
이번 예제는 리모컨을 통해 로봇의 구동부를 제어하는 프로그램입니다.

전체 프로그램

1. Loop – 무한 반복
2. IRReceive – 해당하는 리모컨 데이터에 대한 수신 확인
3. Wheel – 해당 리모컨 데이터에 알맞게 로봇 구동부 제어

리모컨 데이터	구동부
14	전진
13	좌로 평행이동
16	우로 평행이동
17	후진
10	제자리에서 좌로 회전
23	제자리에서 우로 회전
15	정지

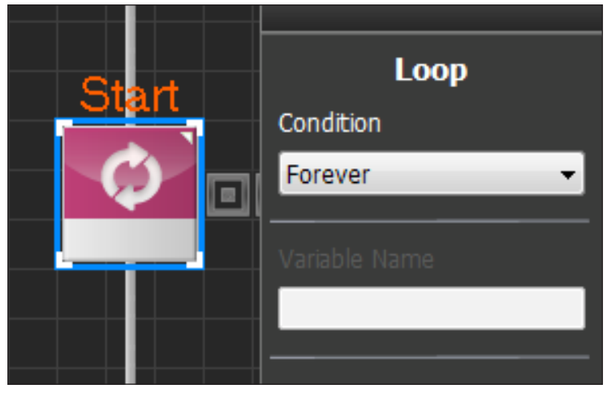
Graphic



C-Like

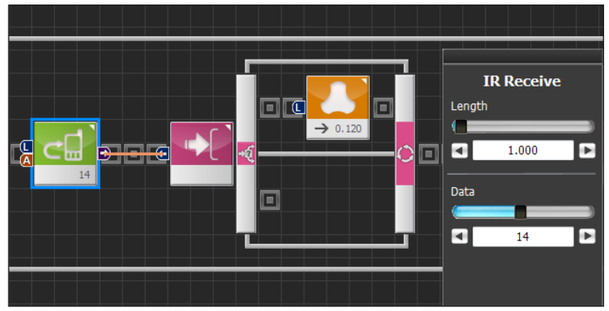
```
void main()
{
	while( true )
	{
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 14 ) )
		{
			mid_wheel_linear( 0.120, 0, false )
		}
		else
		{
		}
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 13 ) )
		{
			mid_wheel_linear( 0.120, 90, true )
		}
		else
		{
		}
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 16 ) )
		{
			mid_wheel_linear( 0.120, -90, true )
		}
		else
		{
		}
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 17 ) )
		{
			mid_wheel_linear( -0.120, 0, true )
		}
		else
		{
		}
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 15 ) )
		{
			mid_wheel_linear( 0.000, 0, false )
		}
		else
		{
		}
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 10 ) )
		{
			mid_wheel_angular( 30.0 )
		}
		else
		{
		}
		if( ( MPSU_RmcLength >= 8 && MPSU_RmcData == 23 ) )
		{
			mid_wheel_angular( -30.0 )
		}
		else
		{
		}
	}
}
```

따라하기



01 Loop

Loop 모듈을 선택하여 하여, Start Point에 도킹합니다.
리모컨 신호가 들어왔는지 지속적으로 확인하므로, 속성창의 Condition을 Forever로 지정합니다.



IRReceive 속성	Wheel 속성
Length: 1.0 Data: 13	좌로 평행이동 Mode: Linear Linear Velocity: 0.12 Enable Offset Angle: True Offset Angle: 90
Length: 1.0 Data: 16	우로 평행이동 Mode: Linear Linear Velocity: 0.12 Enable Offset Angle: True Offset Angle: -90
Length: 1.0 Data: 17	후진 Mode: Linear Linear Velocity: -0.12 Enable Offset Angle: False
Length: 1.0 Data: 15	정지 Mode: Linear Linear Velocity: 0.0 Enable Offset Angle: False
Length: 1.0 Data: 10	제자리에서 좌로 회전 Mode: Angular Angular Velocity: 30.0
Length: 1.0 Data: 23	제자리에서 우로 회전 Mode: Angular Angular Velocity: -30.0

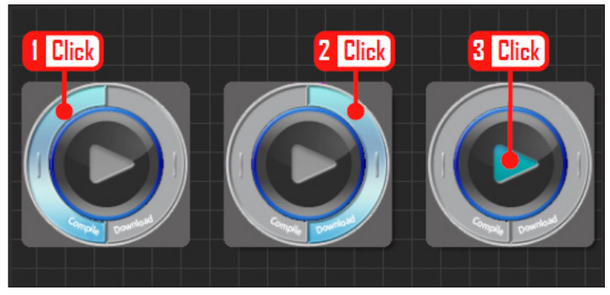
02 리모컨 버튼

IRReceive 모듈을 배치하고 속성 창을 설정합니다. IRReceive 속성 창의 Length는 리모컨 버튼이 눌린 시간이며, Data는 리모컨에 할당된 번호입니다.
(구동 제어 예제설명 그림 참조)
- Length: 1.0 (단위:초)
- Data : 14

14에 해당하는 리모컨 버튼을 1초 이상 누르면 IRReceive 모듈이 TRUE를 출력합니다.

TRUE이면 로봇이 전진 할 수 있도록 Wheel 모듈을 배치합니다.
- Mode: Linear
- Linear Velocity: 0.120
- Enable Offset Angle: False

이어서, 다른 버튼에 대해서도 다음 표를 참조하여 작성합니다.



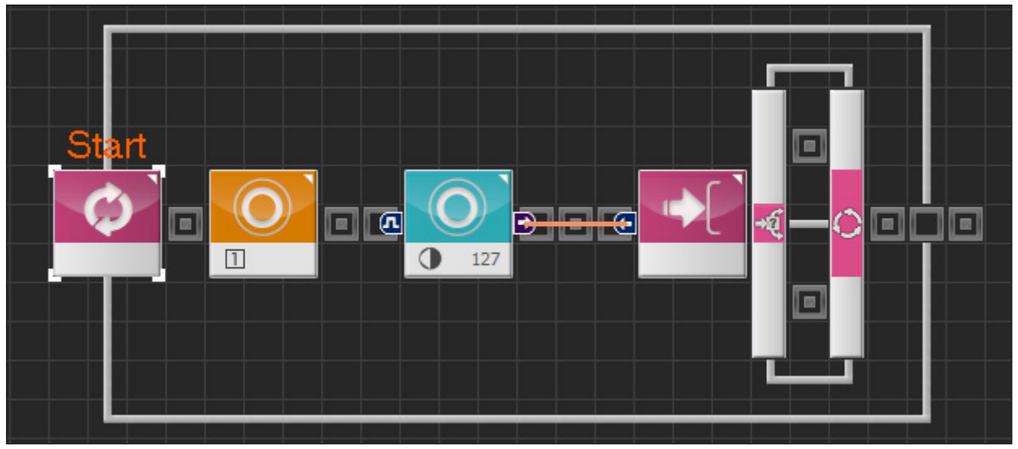
03 컴파일, 다운로드, 실행

왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.

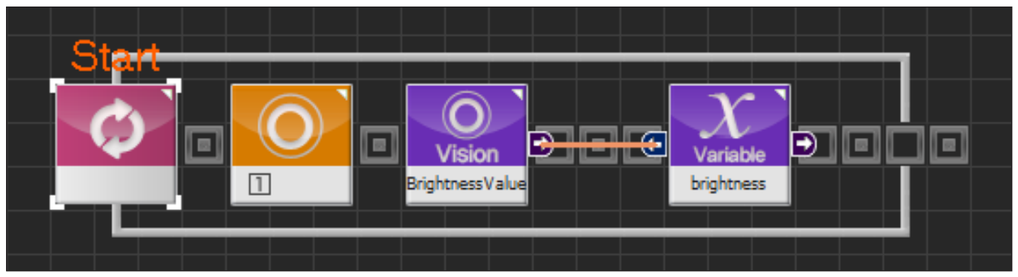
4.8 비전 응용

사람의 감각기관중 가장 많은 정보를 얻을 수 있는 기관이 눈이듯이, 로봇이 주변환경에서 가장 많은 정보를 얻을 수 있는 센서가 비전 카메라입니다.
로봇은 MID에 탑재된 카메라를 눈으로 사용합니다. 그리고 DR-Visual Logic을 이용하면 영상의 Brightness(밝기), Motion(모션 감지), Color(색상 감지)등의 비전 처리와 결합하여 다양한 응용 프로그램을 제작할 수 있습니다.

DR-Visual Logic에서는 영상 획득 모듈과 처리 모듈을 구분합니다. 영상 획득 모듈은 Vision Capture 모듈이며, 영상 처리 모듈은 Vision Sensor와 Vision Result 모듈입니다. 영상 처리 모듈인 Vision Sensor와 Vision Result는 Vision Capture 모듈에 의해 캡쳐된 영상을 이용합니다. 다음의 두 그림은 Vision Capture 모듈과 Vision Sensor 그리고 Vision Result 모듈을 나란히 배치하여, DR-Visual Logic에서 영상을 처리하는 일반적인 흐름을 나타냅니다. (Loop 모듈은 이러한 일련의 영상처리 과정을 반복적으로 수행하기 위하여 필요합니다. 만약, 영상을 한번 만 처리한다면 Loop 모듈은 필요 없습니다.)



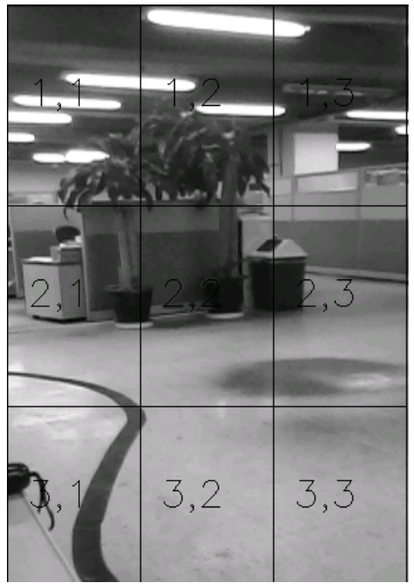
[Vision Capture – Vision Sensor]



[Vision Capture – Vision Result]

실제 비전 처리가 이루어지는 Vision Sensor와 Vision Result 모듈의 경우 공통적으로 속성창에서 Grid와 Region을 지정하여야 합니다. Grid란 Vision Capture 모듈에 의해 획득한 영상의 영역을 구분하기 위하여 가로가 row 개이고 세로가 column 개인 네모 공간으로 표한 하는 것을 말합니다 그리고 Region은 네모로 구분한 영역을 (1,1) ~ (row, col)의 각 네모 공간을 말합니다.

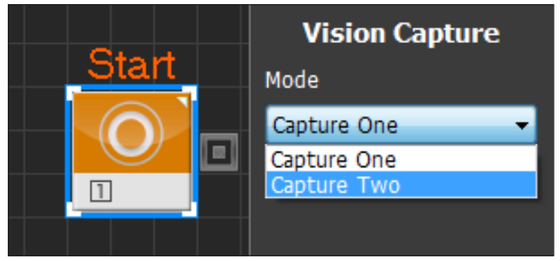
만약 획득한 영상의 Grid Size (Row x Column)를 3 x 3으로 지정한다면, 아래의 그림처럼 획득한 영상이 총 9개의 영역으로 구분됩니다.



Region은 영상을 처리하고자 하는 대상이 되는 영역입니다. 위와 같이 Grid Size가 3 x 3인 영역에서 영상을 처리 할 관심 영역이 가장 중심이 되는 가운데 영역이라면, Region을 (2, 2)로 지정하면 됩니다.

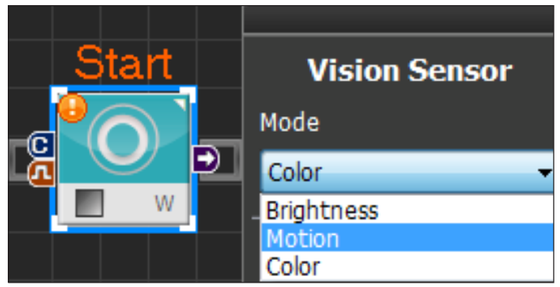
아래그림은 DR-Visual Logic에서 영상처리 관련 모듈의 속성 창 내역입니다.

Vision Capture 모듈



Mode 〉 Capture One: 영상을 한 번 캡쳐
Mode 〉 Capture Two: 영상을 두 번 캡쳐
Interval: 영상을 캡쳐 한 후, 두 번째 영상 캡쳐까지의 시간 (1~1000ms) 모션 감지시 사용

Vision Sensor와 Vision Result 모듈



(1) Vision Sensor

Vision Sensor 모듈은 Vision Capture 모듈에 의해 캡쳐된 영상을 처리하여 TRUE 또는 FALSE를 출력하는 모듈입니다.

Mode 〉 Brightness

영역의 이미지 밝기 평균값이 Threshold 값보다 작거나 같으면 TRUE Brightness Threshold : 밝기에 대한 임계 값 (0~255)

Mode 〉 Motion:

영역의 움직임에 대한 Threshold 값보다 크거나 같으면 TRUE. 두 장의 캡쳐가 필요.
Motion Threshold : 움직임에 대한 임계 값 (0~255)

Mode 〉Color

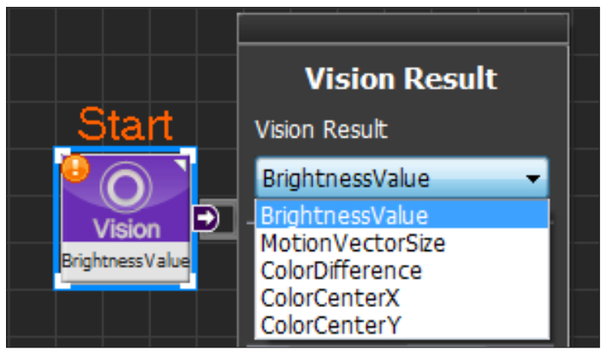
지정된 색에 대한 Threshold 값보다 작거나 같으면 TRUE.

Color : White, Red, Green, Blue, Yellow, Black 중 택일

Color Threshold : 지정된 Color에 대한 임계 값 (0~255)

(2) Vision Result

Vision Result 모듈은 Vision Capture 모듈에 의해 캡쳐된 영상을 처리하여 해당 값을 출력하는 모듈입니다.



Vision Result 〉BrightnessValue

영역의 이미지 밝기 평균값을 출력합니다.

Vision Result 〉MotionVectorSize

영역의 움직임에 대한 크기를 출력합니다.

Vision Result 〉ColorDifference

지정한 색과 영역에서 탐지 된 색의 차이를 출력합니다.

Color : White, Red, Green, Blue, Yellow, Black 중 택일

Vision Result 〉ColorCenterX

Vision Result 〉ColorCenterY

영역내의 지정한 색으로 탐지된 물체가 있으면, 물체의 가운데 x좌표 또는 y좌표를 출력합니다. (출력된 좌표는 MID의 화면 크기 320X480에 기준함)

Color : White, Red, Green, Blue, Yellow, Black 중 택일

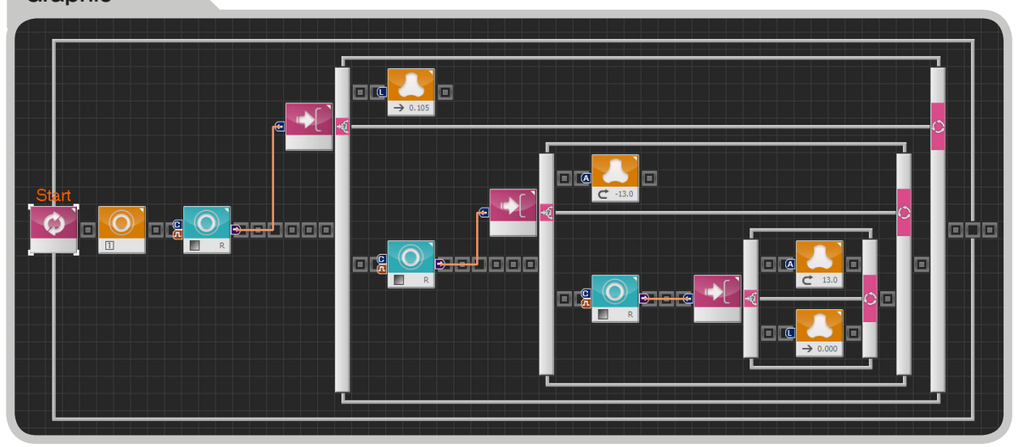
예제설명

이번 예제는 로봇이 빨간 색 물체를 탐지하여 그 물체를 따라가는 예제입니다. 이 예제는 MID 카메라의 영상을 캡쳐하고 Grid Size를 3X3으로 나눕니다. 그리고 Region (2,2)에 빨간 물체가 있다면 로봇을 전진시키고, (2,1)과 (2,3)에 있으면 로봇을 우측과 좌측으로 제자리 회전하게 하게 하여 물체를 따라갑니다.

전체 프로그램

1. Loop – 무한루프
2. Vision Capture – MID 카메라로 영상 캡쳐
3. Vision Sensor – Grid Size (3X3) Region (2,2)에서 빨간 물체 감지 시 전진
4. Vision Sensor – Grid Size (3X3) Region (2,1)에서 빨간 물체 우측으로 회전
5. Vision Sensor – Grid Size (3X3) Region (2,3)에서 빨간 물체 좌측으로 회전

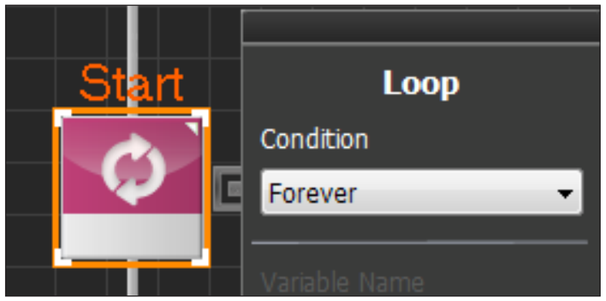
Graphic



C-Like

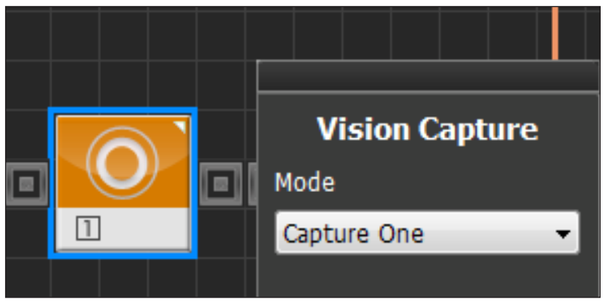
```
void main()
{
	while( true )
	{
		mid_vision_capture_one()
		if( ( VISION_ColorDifference[ 3, 3, 2, 2, 1 ] <= 30 ) )
		{
			mid_wheel_linear( 0.105, 0, false )
		}
		else
		{
			if( ( VISION_ColorDifference[ 3, 3, 2, 1, 1 ] <= 30 ) )
			{
				mid_wheel_angular( -13.0 )
			}
			else
			{
				if( ( VISION_ColorDifference[ 3, 3, 2, 3, 1 ] <= 30 ) )
				{
					mid_wheel_angular( 13.0 )
				}
				else
				{
					mid_wheel_linear( 0.000, 0, false )
				}
			}
		}
	}
}
```

따라하기



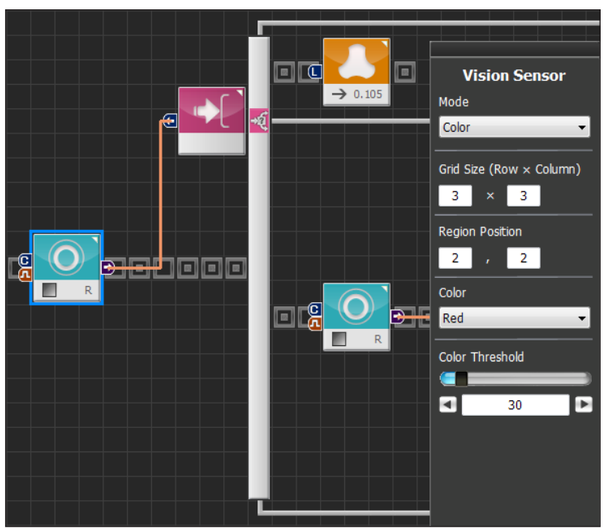
01 Loop

반복적으로 영상처리를 하기 위해서 Condition을 Forever로 지정한 루프 모듈을 배치합니다.



02 Vision Capture

영상을 획득합니다. Mode를 Capture One으로 지정하여 캡쳐 시 한 장의 사진을 캡쳐하도록 합니다.

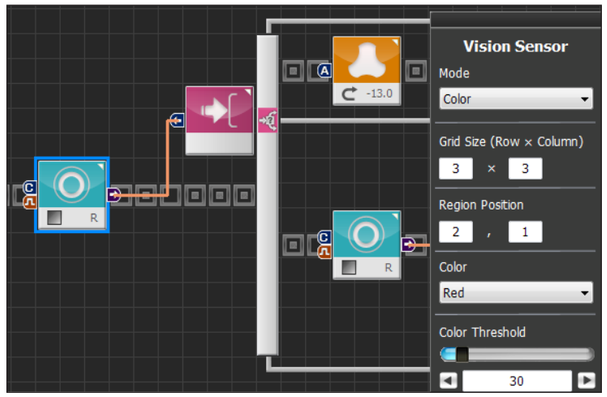


03 Vision Sensor (로봇 전진 조건)

Vision Sensor 모듈을 배치하고, 가운데 빨간 물체를 탐지하면 로봇을 전진, 그렇지 않으면 다른 조건을 배치합니다. Vision Sensor 모듈의 속성 창을 다음과 같이 지정합니다.

– Mode: Color
– Grid Size (Row x Column): 3 x 3
– Region Position: 2, 2
– Color: Red
– Color Threshold: 30

※Vision Sensor 모듈은 빨간색으로부터 색조가 30정도의 차이가 있는 색이 감지되면 빨간색을 찾았다고 인식합니다. Color Threshold 값을 증가시키면 빨간색을 찾는 조건을 완화시킬 수 있습니다.

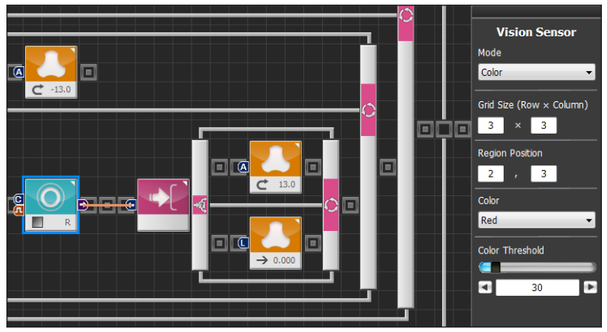


04 Vision Sensor (로봇 우회전 조건)

03 의 Vision Sensor 모듈이 FALSE일 경우, Vision Sensor 모듈을 통해 우측에 빨간색 물체가 있는 지 검사합니다. 우측에 빨간색 물체가 있으면 로봇이 우측으로 회전하게 됩니다.

Vision Sensor 모듈의 속성 창을 다음과 같이 지정합니다.

– Mode: Color
– Grid Size (Row x Column): 3 x 3
– Region Position: 2, 1
– Color: Red
– Color Threshold: 30

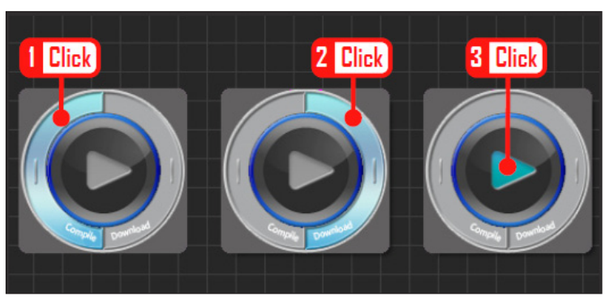


05 Vision Sensor (로봇 좌회전 및 멈춤)

04 의 Vision Sensor 모듈이 FALSE일 경우, Vision Sensor 모듈을 통해 좌측에 빨간색 물체가 있는 지 검사합니다. 좌측에 빨간색 물체가 있으면 로봇이 좌측으로 회전하게 됩니다.

Vision Sensor 모듈의 속성 창을 다음과 같이 지정합니다.

– Mode: Color
– Grid Size (Row x Column): 3 x 3
– Region Position: 2, 3
– Color: Red
– Color Threshold: 30



06 컴파일, 다운로드, 실행

왼쪽 컴파일 버튼을 클릭하여 프로그램을 컴파일 합니다. 에러가 없으면 오른쪽 다운로드 버튼을 클릭하여 프로그램을 MID에 다운로드 합니다. 다운로드가 완료되면, 실행버튼을 클릭하여 프로그램을 실행시킵니다.