소프트웨어 : DR-Visual Logic

모듈별 프로그래밍 : Button, LED

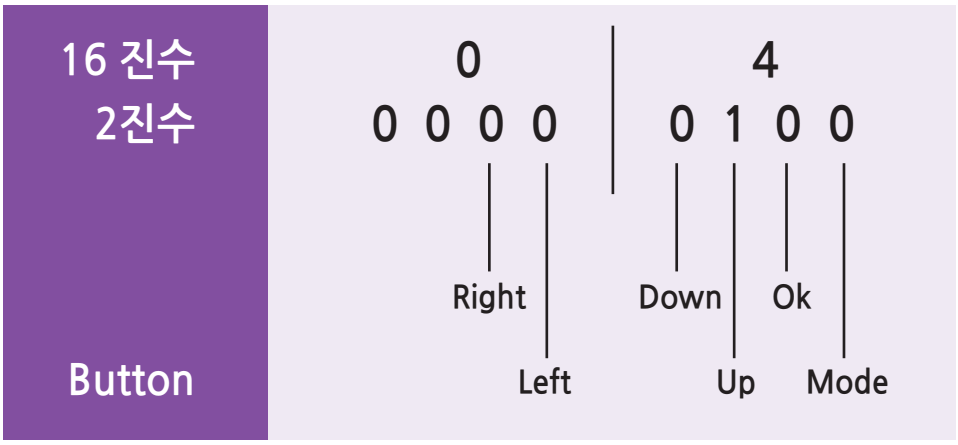
Button, LED 예제 따라하기

예제설명

DRC 제어기에 있는 Button 을 이용하여 LED 를 켰다 끄는 프로그래밍을 합니다.

Button 과 LED 프로그래밍을 하기위해선 Button 과 LED 가 켜지는 과정을 먼저 알아야 합니다.

$16 = 2^4$



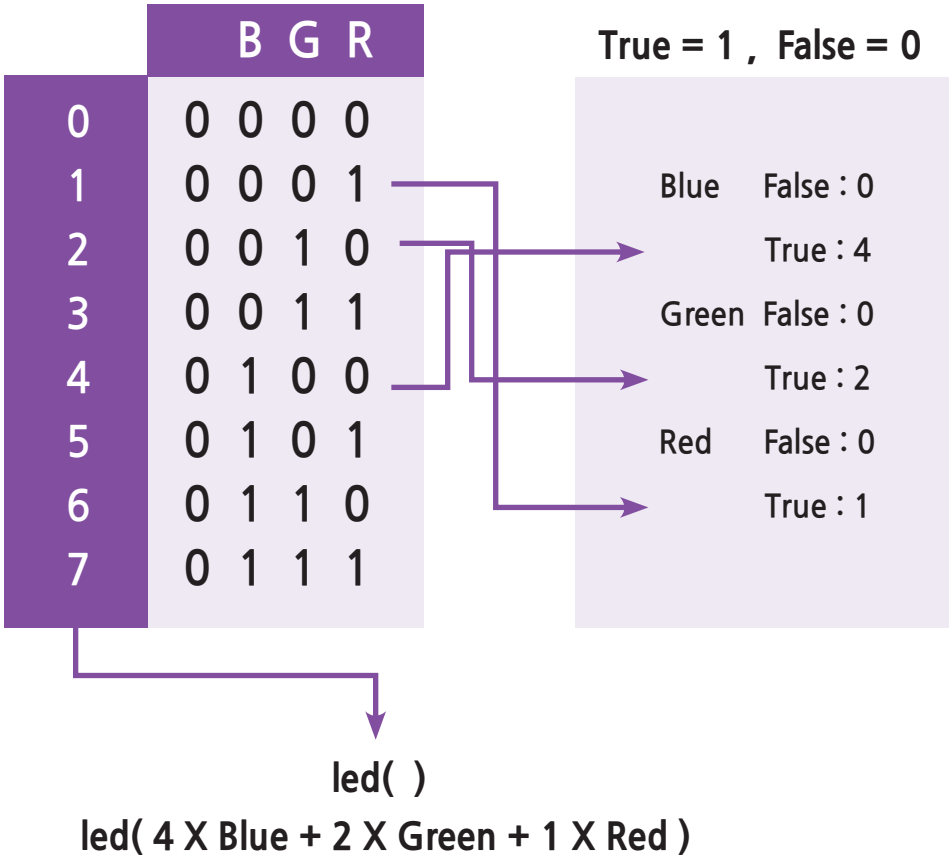
EX)

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

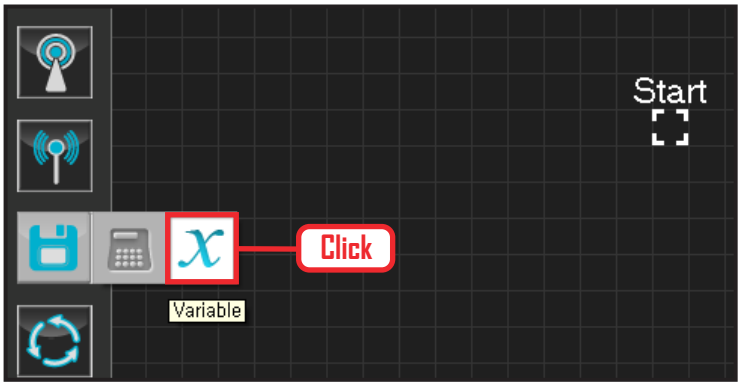
DRC에는 7개의 LED가 있지만, Task 모드에서 제어 가능한 것은 3개입니다. Red, Green, Blue의 LED 3개가 켜진 상태와 꺼진 상태를 표시하기 위해서는 3Bit가 필요합니다. 위 그림에서 보는 바와 같이, 각 LED는 바이트의 최하위 비트부터 Red, Green, Blue의 순서로 매칭되어 있습니다.

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.

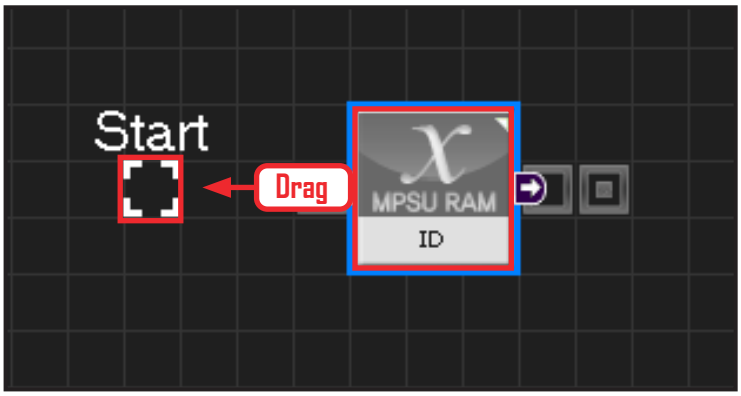
예를 들어서 Blue, Green이 true이고 Red가 false일 때 4 x Blue + 2 x Green + 1 x Red = 6이 됩니다. 6은 2진수로 표시하면 00000110이므로 이것을 LED 모듈의 입력 값으로 넣으면 Green, Blue LED가 켜지는 것입니다.

위 배경지식을 기반으로 Button 과 LED 프로그래밍을 해봅니다



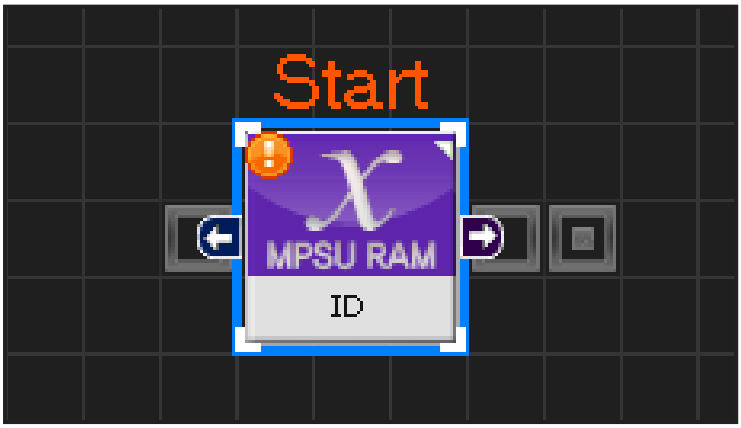
01 변수 지정

Data 〉 Variable 모듈을 클릭합니다.



02 시작

모듈의 왼쪽 연결선을 Start Point 에 드래그하여 정확히 도킹을 시킵니다.

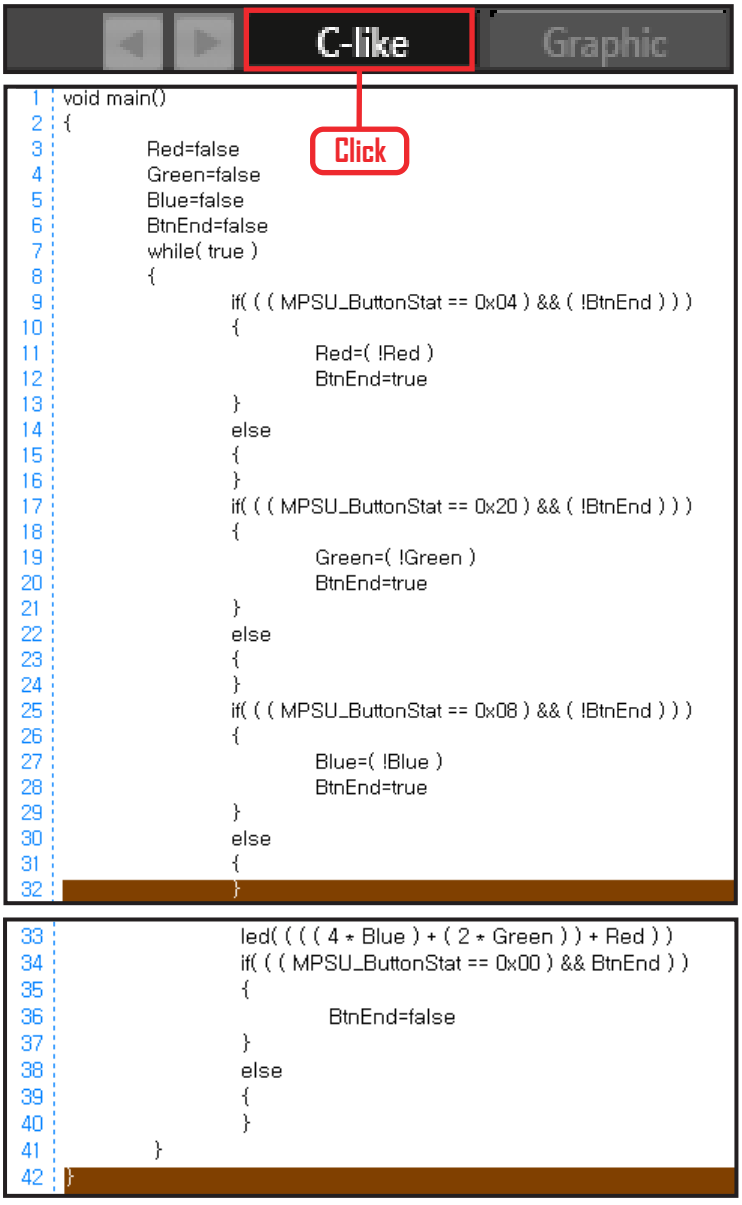


03 프로그래밍 시작

모듈과 Start Point 가 정확히 도킹하면 왼쪽과 같이 활성화된 칼라 이미지 모듈로 변합니다.
그럼 프로그래밍이 시작되었다는 의미입니다.

04 전체 프로그래밍

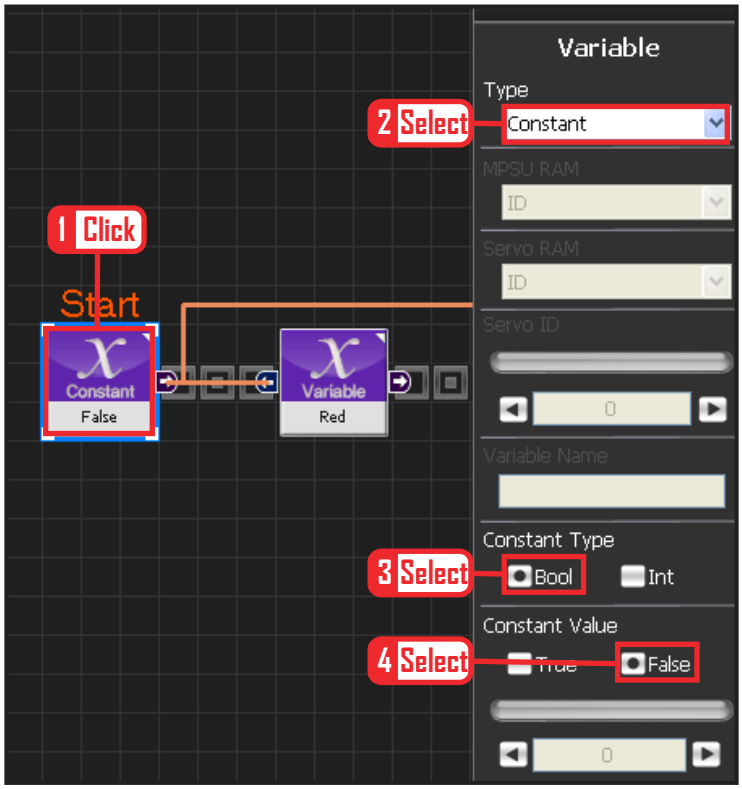
버튼과 LED를 이용한 전체 프로그래밍입니다.




```
1  void main()
2  {
3          Red=false
4          Green=false
5          Blue=false
6          BtnEnd=false
7          while( true )
8          {
9                  if( ( ( MPSU_ButtonStat == 0x04 ) && ( !BtnEnd ) ) )
10                 {
11                         Red=( !Red )
12                         BtnEnd=true
13                 }
14                 else
15                 {
16                 }
17                 if( ( ( MPSU_ButtonStat == 0x20 ) && ( !BtnEnd ) ) )
18                 {
19                         Green=( !Green )
20                         BtnEnd=true
21                 }
22                 else
23                 {
24                 }
25                 if( ( ( MPSU_ButtonStat == 0x08 ) && ( !BtnEnd ) ) )
26                 {
27                         Blue=( !Blue )
28                         BtnEnd=true
29                 }
30                 else
31                 {
32                 }
33                 led( ( ( ( 4 * Blue ) + ( 2 * Green ) ) + Red ) )
34                 if( ( ( MPSU_ButtonStat == 0x00 ) && BtnEnd ) )
35                 {
36                         BtnEnd=false
37                 }
38                 else
39                 {
40                 }
41         }
42 }
```

05 C-Like 보기

오른쪽 상단의 Graphic 탭에서 C-like 탭을 클릭하면 왼쪽과 같은 Task 프로그래밍 화면이 나옵니다.
Button과 LED를 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.

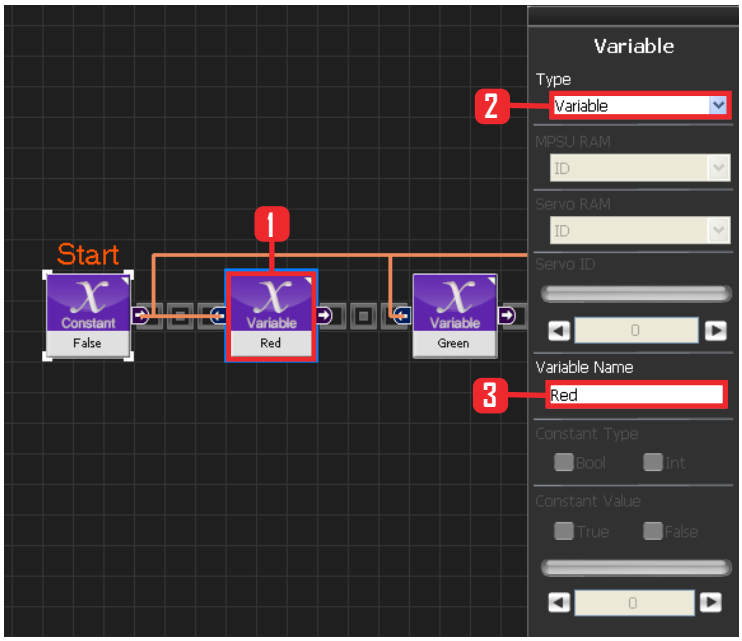


06 False 로 초기화

모든 LED 는 False 꺼진 상태로 초기화합니다.

Data 〉 Variable 모듈을 선택합니다.
Type : Contant 를 선택합니다.
Constant Type 은 Bool 로 설정합니다. Bool 은 참과 거짓을 나타내는 자료형입니다.
Constant Value : False 를 선택합니다

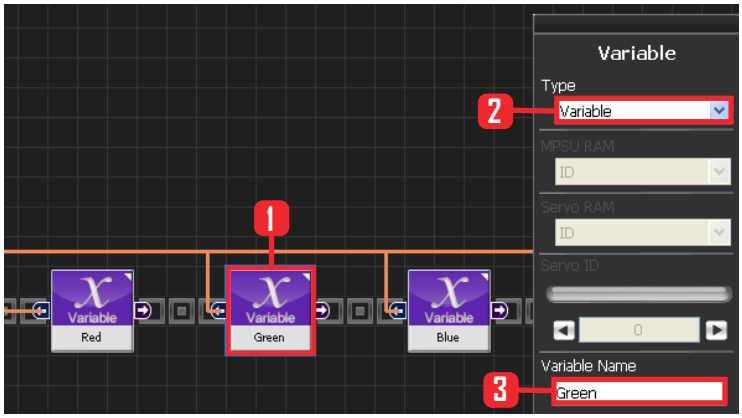
이후 커넥터를 이용하여 False를 각 변수에 연결합니다.



07 Red 변수

Data 〉 Variable 을 선택합니다.
Type : Variable 로 선택합니다.
Variable Name : Red 로 입력합니다.

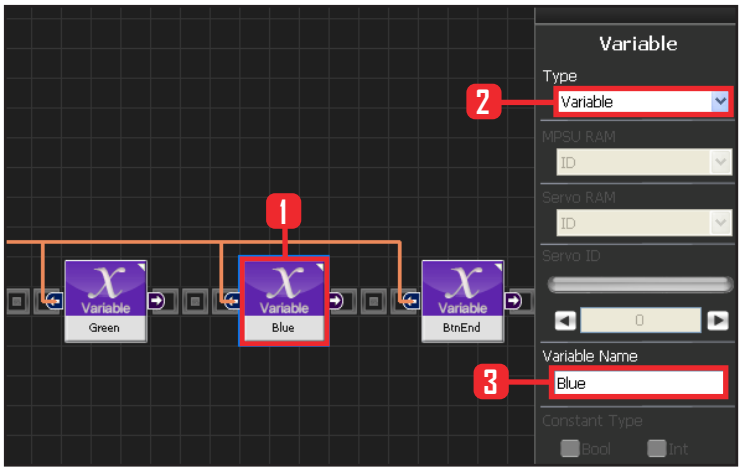
False 면 빨간색 LED가 꺼지고, True 면 켜집니다.



08 Green 변수

Data 〉 Variable 을 선택합니다.
Type : Variable 로 선택합니다.
Variable Name : Green 으로 입력합니다.

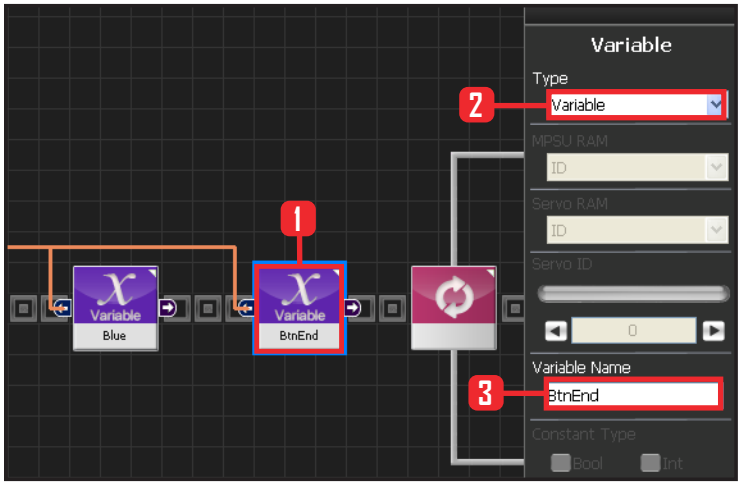
False 면 초록색 LED가 꺼지고, True 면 켜집니다.



09 Blue 변수

Data 〉 Variable 을 선택합니다.
Type : Variable 로 선택합니다.
Variable Name : Blue 로 입력합니다.

False 면 파란색 LED가 꺼지고, True 면 켜집니다.



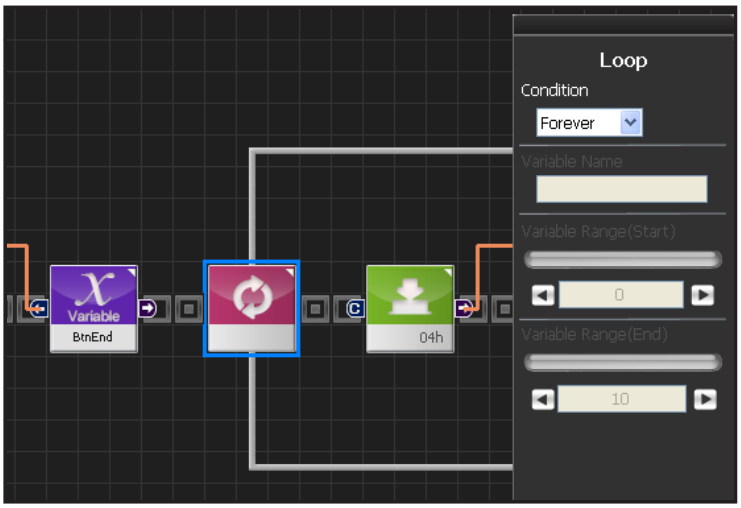
10 BtnEnd 변수

버튼이 안 눌렸을 때 False로 유지되다가
버튼이 눌리고 원하는 동작이 끝나는 순간
False –〉 True로 바뀌는 변수입니다.
또한 버튼을 떼는 순간 True 에서 False로 바뀌는 변수입니다.

Data 〉 Variable 을 선택합니다.
Type : Variable 로 선택합니다.
Variable Name : BtnEnd 로 입력합니다.

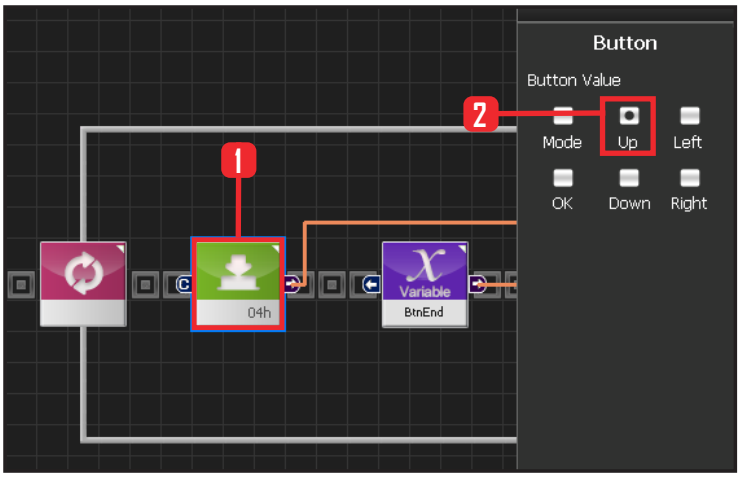
11 변수 선언

Red, Green, Blue, BtnEnd 4개 모두 초기값을 False 로 설정합니다.



12 Loop

Forever 무한 반복시킵니다.

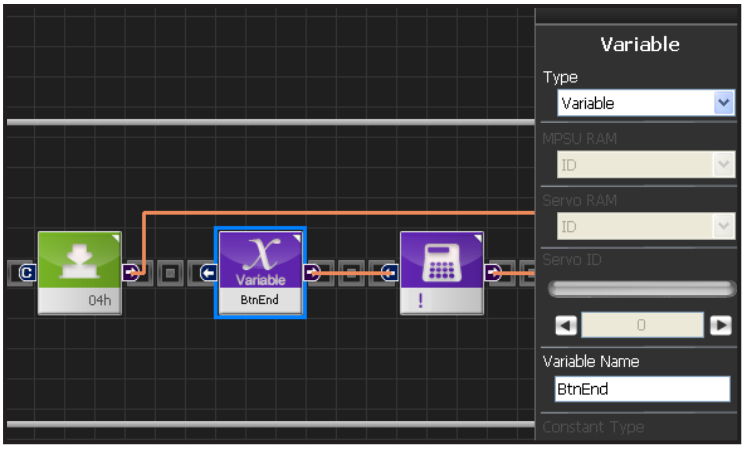


13 Up 버튼

버튼모듈을 만듭니다. 이모듈은 선택한 버튼이 눌렸을 경우 True가 되고 그 외에는 False가 됩니다. Up Button 을 선택시, Up Button이 눌린경우 true, 그외에 false 가 됩니다.

Communication 〉 Button 모듈을 선택합니다.
Button Value 를 Up 으로 선택합니다.

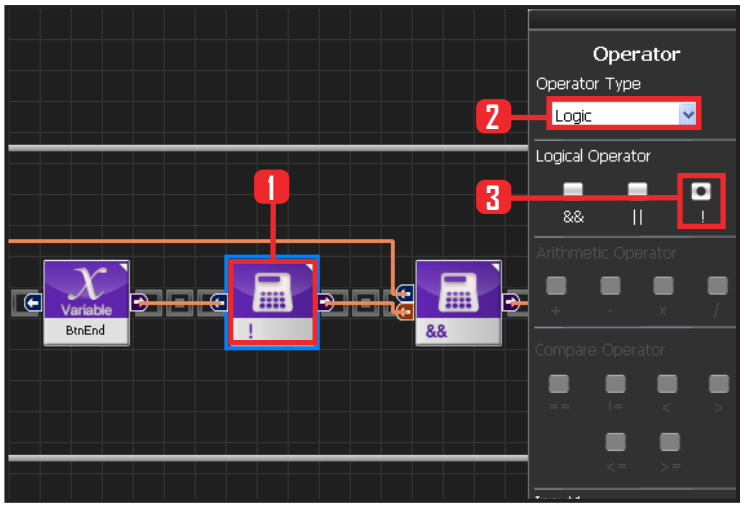
16진수로 04h 이므로 모듈 그림에 04h가 표기됩니다.



14 BtnEnd

BtnEnd 값은 false 로 초기화 되어있습니다. 뒤에 not 연산이 붙어서 True 로 변합니다.

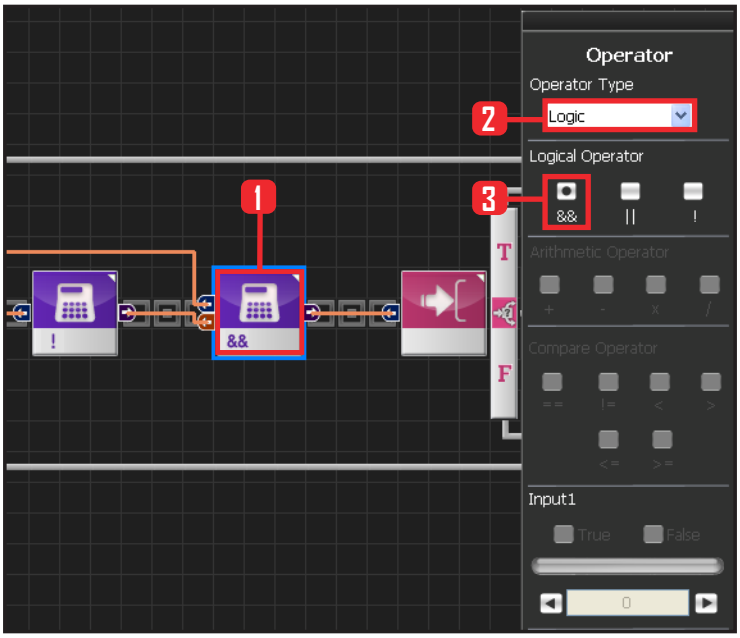
앞의 btnEnd 변수를 복사하여 붙입니다.



15 ! 연산

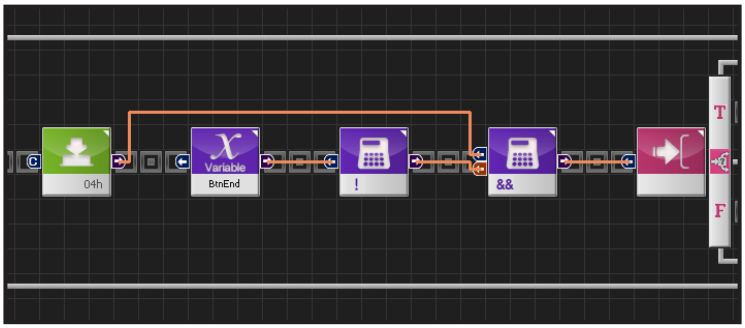
! 연산을 적용하여 BtnEnd 값을 반대로 바꿉니다.

Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Logic으로 선택합니다.
Logical Operator : ! 로 선택합니다.



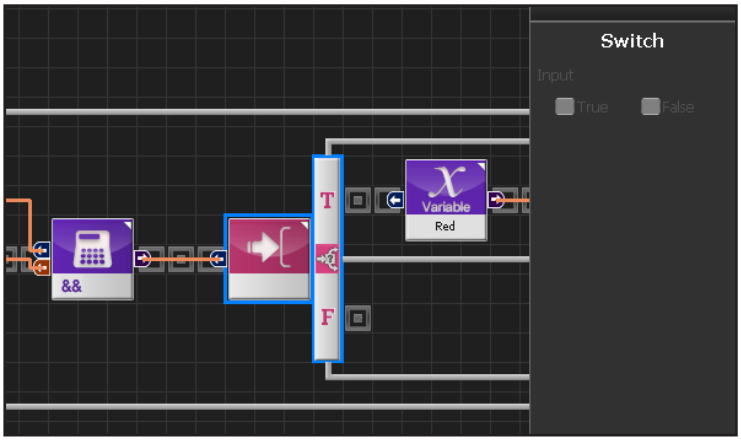
16 And 연산

Up 버튼이 눌러졌고, BtnEnd가 false (! 적용하여 True) 일때 True가 되어, 뒤의 조건문을 실행하도록 합니다.
Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Logic으로 선택합니다.
Logical Operator : && 로 선택합니다.



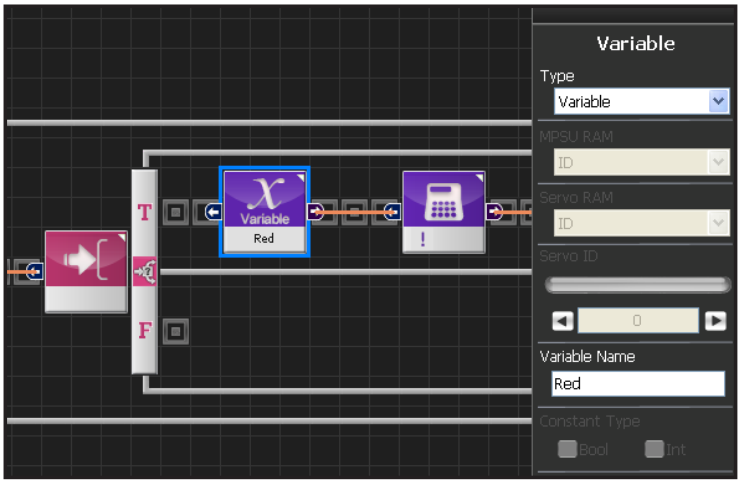
17 Up 버튼이 눌렸을 때

Up 버튼이 눌려졌고 BtnEnd가 false일때, 뒤 조건을 수행합니다.



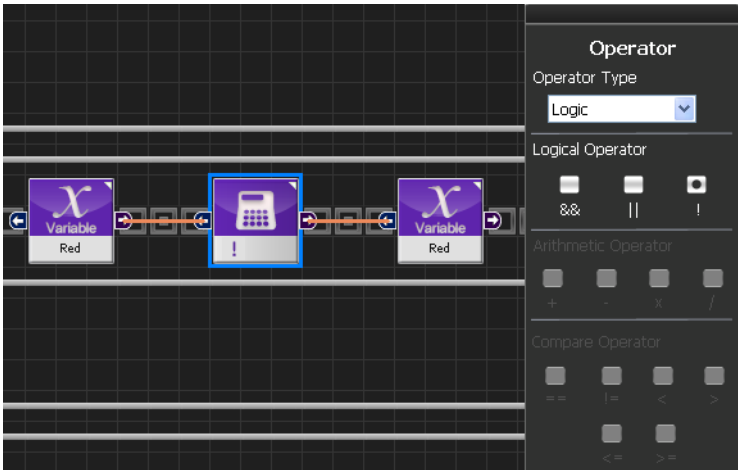
18 If 분기문

True 일 때 윗 부분을 수행합니다.



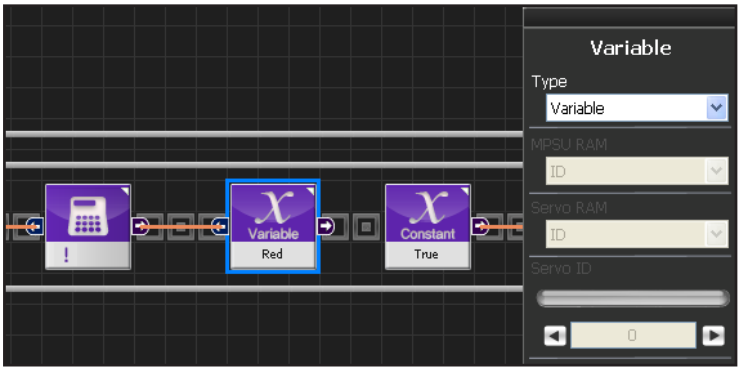
19 Red 출력

앞의 Red변수를 복사하여 붙입니다.



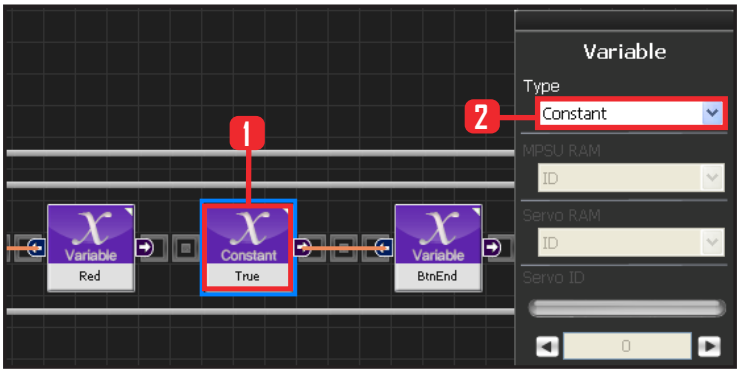
20 ! 연산

Red 가 True 이면 False 로 False 이면 True 로 변합니다.



21 Red 입력

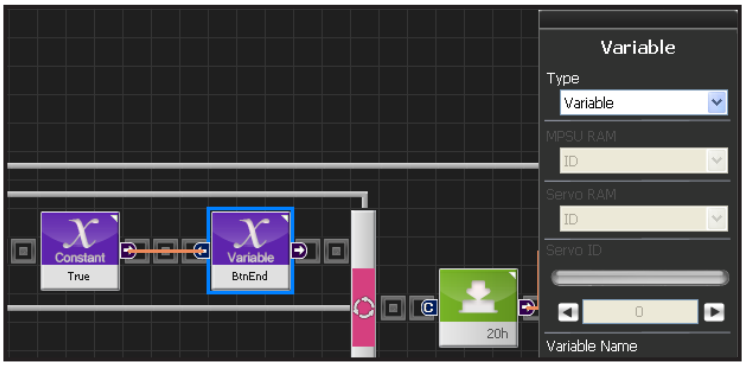
앞의 Red변수의 값이 true 일땐 false로, false일땐 true로 바뀌어 저장됩니다.



22 True 설정

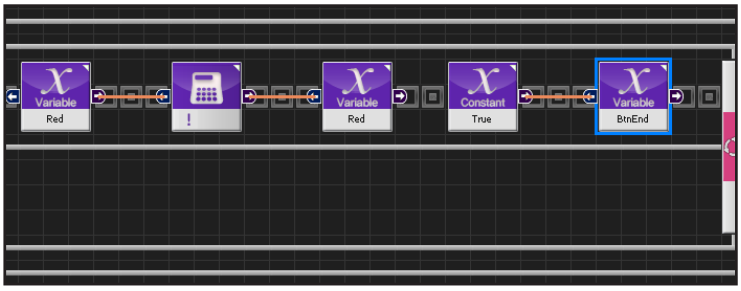
버튼이 눌리고 해야할 동작이 끝났으므로
BtnEnd가 false 에서 True로 바뀌어야 합니다.

Data 〉 Variable 모듈을 선택합니다.
Type : Contant 를 선택합니다.
Constant Type 은 Bool 로 설정합니다. Bool 은 참과 거짓을 나타내는 자료형입니다.
Constant Value : True 를 선택합니다



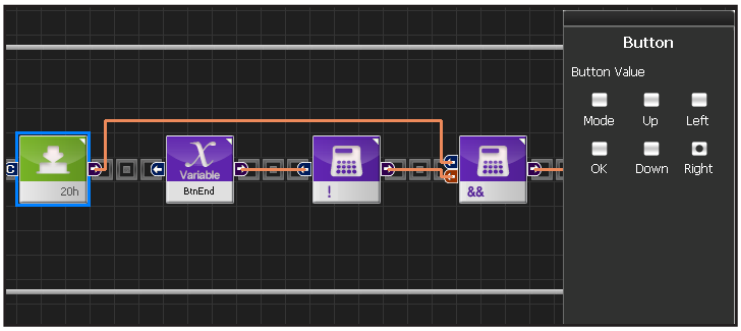
23 BtnEnd를 true로

True 값이 BtnEnd 에 입력됩니다.
BtnEnd가 true가 되면, loop를 반복할 때 up버튼이 계속 늘려 있더라도 조건문을 만족시키지 않으므로 Red변수의 값이 더이상 변하지 않게 됩니다.



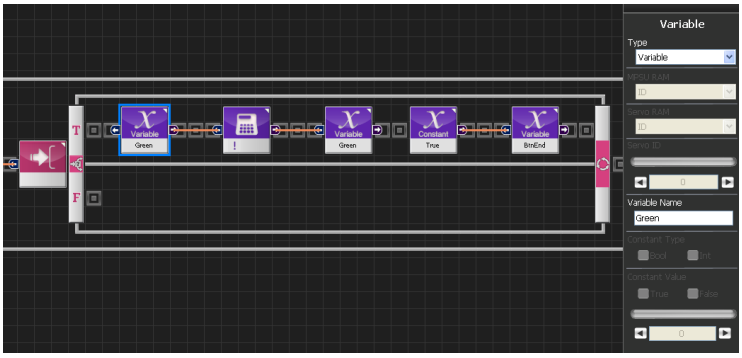
24 Red LED

Up버튼을 한번 누르면 Red가 켜지고, 다시 한번 누르면 Red 가 꺼집니다.



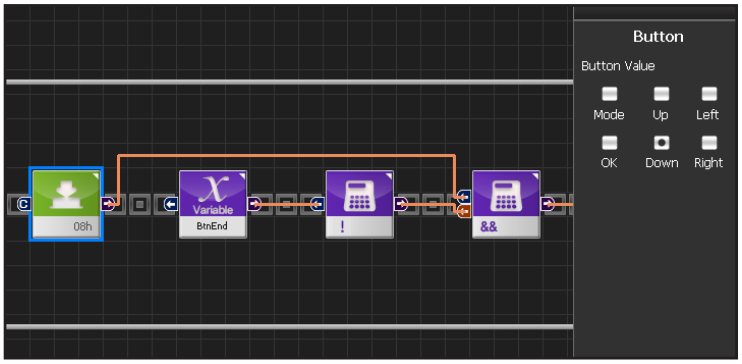
25 Right 버튼

Right 버튼을 눌렀을 때입니다.



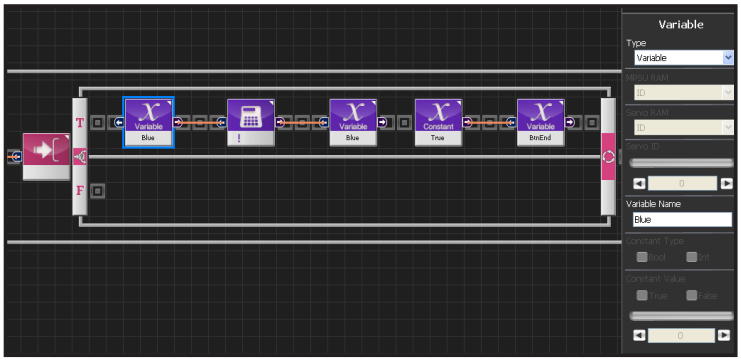
26 Green LED

Right 버튼을 한번 누르면 Green 이 켜졌다가, 한번 더 누르면 Green 이 꺼집니다.



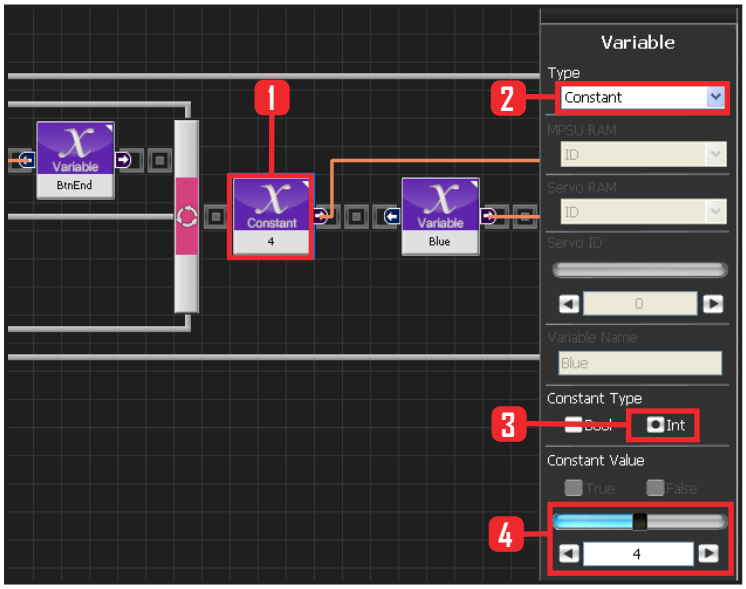
27 Down 버튼

Down 버튼을 눌렀을 때입니다.



28 Blue LED

Down 버튼을 한번 누르면 Blue 가 켜졌다가, 한번 더 누르면 Blue 가 꺼집니다.



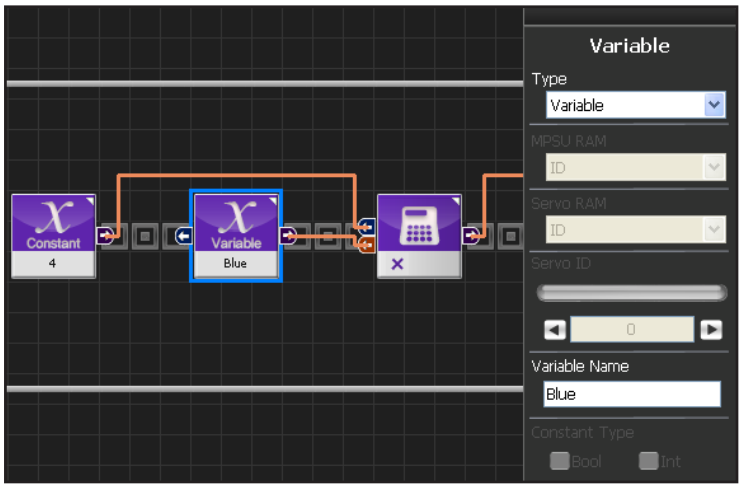
29 LED 값

위에서 설명했듯이 LED 모듈에 입력값이 들어가면, 그 값에 따라 LED가 켜집니다. 그 식은 (4 x Blue + 2 x Green + 1 x Red) 입니다.
위 식을 모듈별로 연결하여 표현한 것이 좌측 그림입니다.

상수 4 를 설정합니다.

(4 x Blue + 2 x Green + 1 x Red)

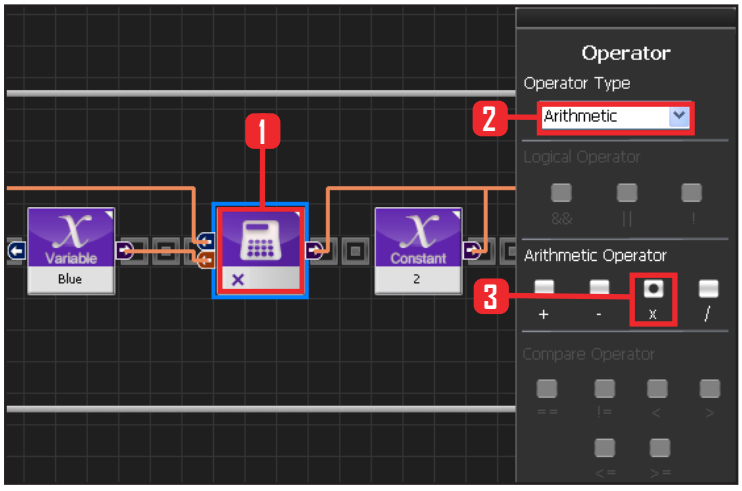
Data 〉 Variable 모듈을 선택합니다.
Type : Contant 를 선택합니다.
Constant Type 은 int 로 설정합니다.
Constant Value : 4로 설정합니다.



30 Blue

(4 x Blue + 2 x Green + 1 x Red)

앞의 Blue 변수를 복사합니다.

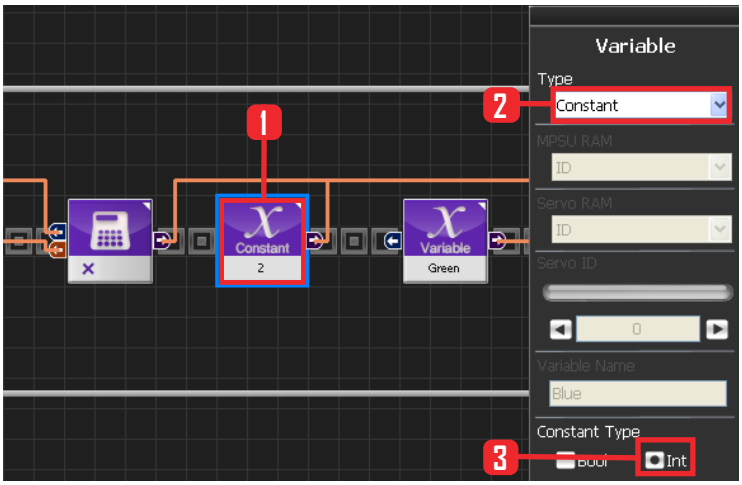


31 곱하기

(4 x Blue + 2 x Green + 1 x Red)

Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Arithmetic으로 선택합니다.
Arithmetic Operator : X 로 선택합니다.

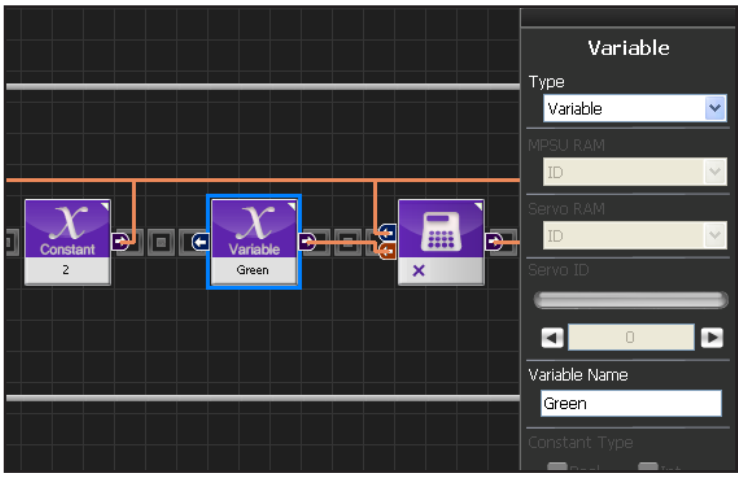
곱하기 모듈의 두 입력 커넥터에 상수 4와 Blue 변수 모듈을 이어 붙입니다.



32 상수2

(4 x Blue + 2 x Green + 1 x Red)

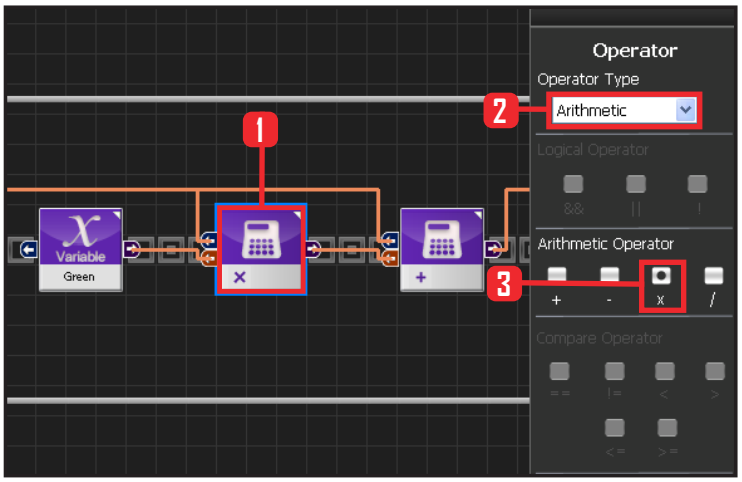
Data 〉 Variable 모듈을 선택합니다.
Type : Contant 를 선택합니다.
Constant Type 은 int 로 설정합니다.
Constant Value : 2로 설정합니다.



33 Green

(4 x Blue + 2 x Green + 1 x Red)

앞의 Green 변수를 복사합니다.

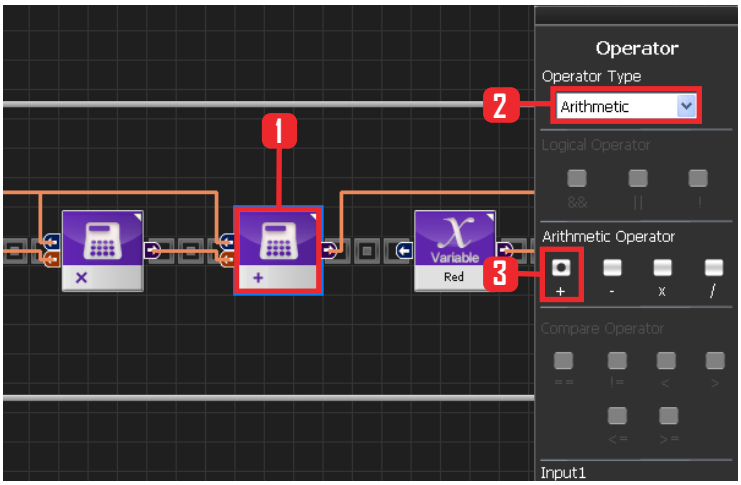


34 곱하기

(4 x Blue + 2 x Green + 1 x Red)

Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Arithmetic으로 선택합니다.
Arithmetic Operator : X 로 선택합니다.

곱하기 모듈의 두 입력 커넥터에 상수 2와 Green 변수 모듈을 이어 붙입니다.

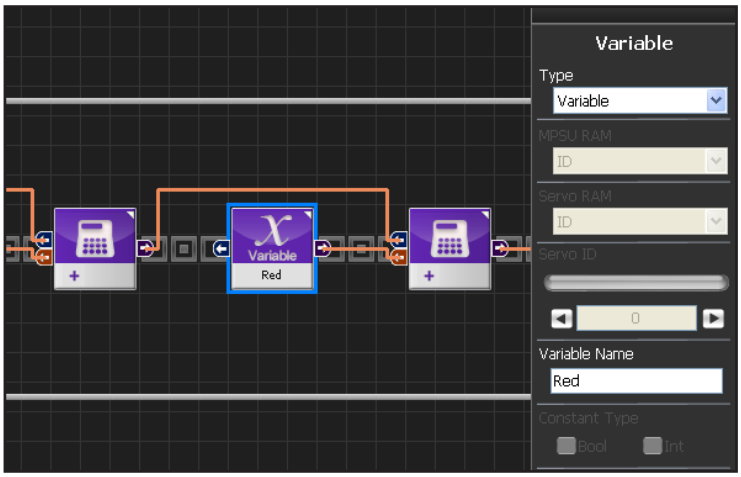


35 더하기

(4 x Blue + 2 x Green + 1 x Red)

Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Arithmetic으로 선택합니다.
Arithmetic Operator : + 로 선택합니다.

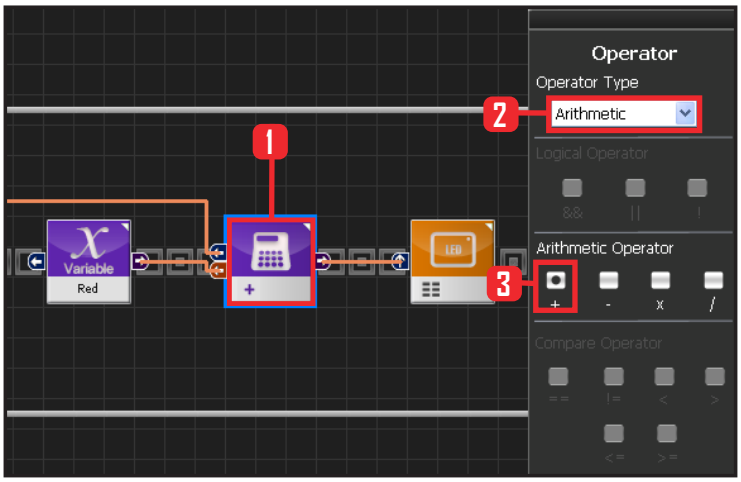
앞서 31, 34번에서 나온 두 곱하기 모듈의 출력을 더하기 모듈의 두 입력 커넥터에 연결합니다.



36 Red

(4 x Blue + 2 x Green + 1 x Red)

앞의 Red 변수를 복사해 옵니다.

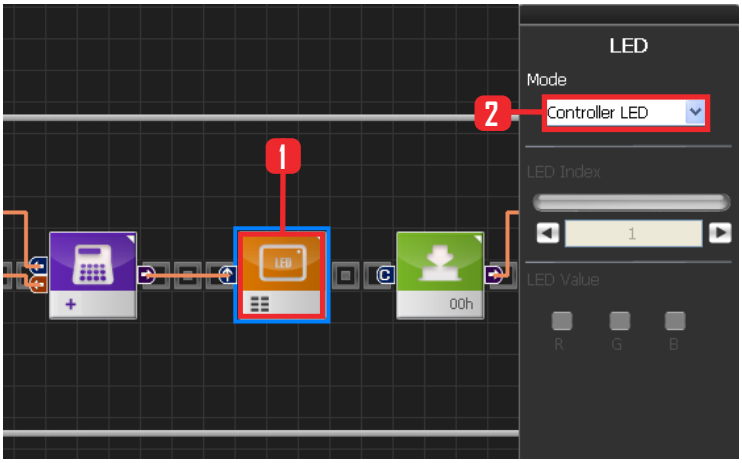


37 더하기

(4 x Blue + 2 x Green + 1 x Red)

Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Arithmetic으로 선택합니다.
Arithmetic Operator : + 로 선택합니다.

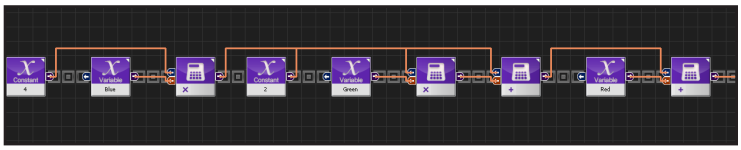
35번의 더하기 모듈과 Red 변수 모듈의 출력을 더하기 모듈의 두 입력 커넥터에 연결합니다.



38 LED

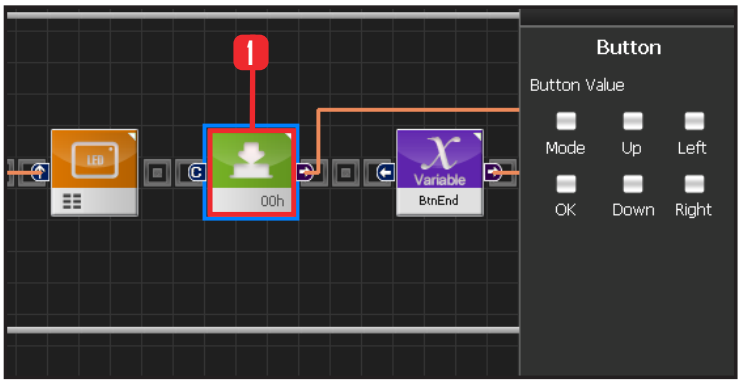
Motion 〉 LED 모듈을 선택합니다.
Mode : Controller LED 를 선택합니다.

앞에서 연산한 값을 LED Value에 넣어서 각각의 LED 가 켜지게 합니다.



39 LED 값 산출 연산

(4 x Blue + 2 x Green + 1 x Red)
을 모듈로 표현한 것입니다.

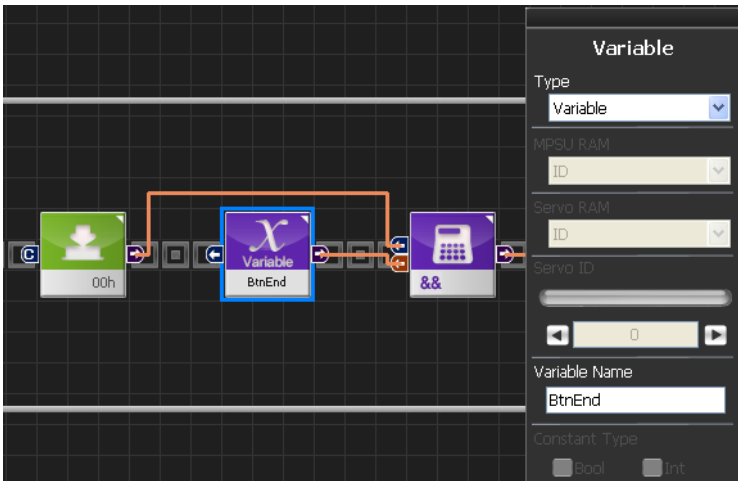


40 버튼이 안눌린 상태

버튼이 눌려서 원하는 동작을 마친후 BtnEnd 변수는 True로 바뀝니다. BtnEnd 변수가 True가 되면 버튼에 따른 동작을 더이상 하지 않으므로 한번 누를 때 원하는 동작이 한번만 실행되게 할 수 있습니다. 버튼을 떼면, BtnEnd를 false로 초기화 시켜야 합니다. 지금부터는 초기화 과정을 프로그래밍 합니다.

Motion 〉 Button 모듈을 선택합니다.
Button Value : 아무것도 선택하지 않습니다.

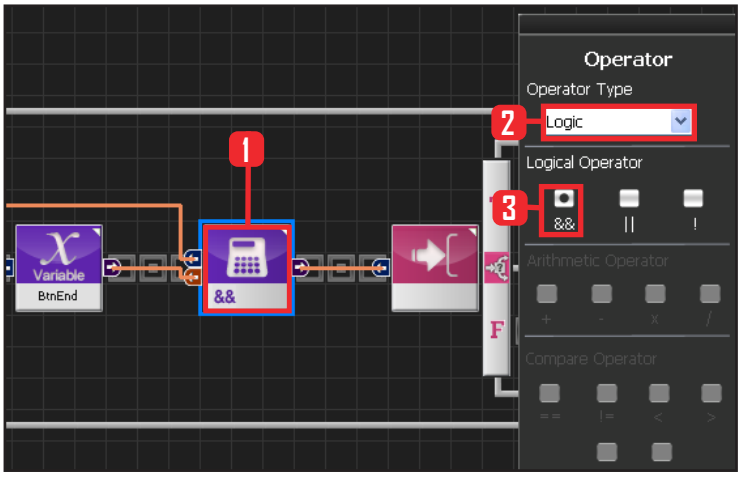
버튼이 안눌린 상태를 의미합니다.



41 BtnEnd 가 True

BtnEnd가 True 일때입니다.

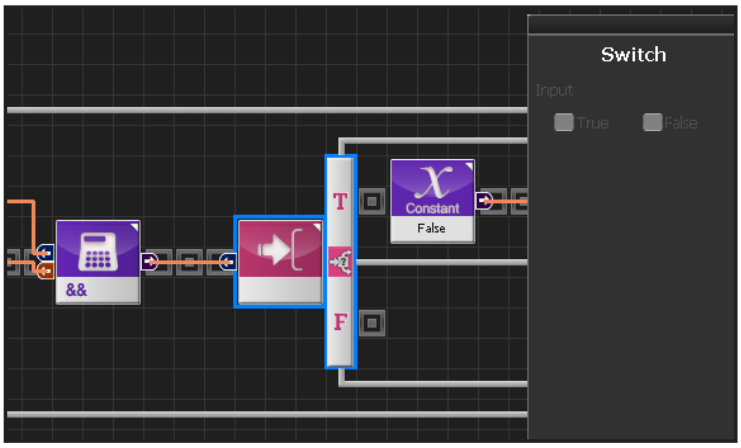
앞의 BtnEnd 변수를 복사합니다.



42 && 연산자

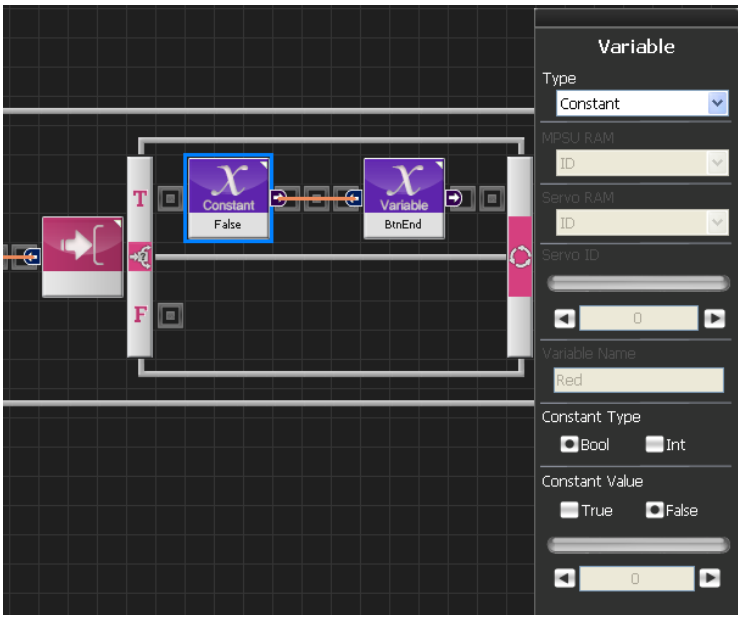
버튼이 안눌린 상태와 BtnEnd 가 True 를 동시에 만족하는 상태는 방금 버튼을 뗀 상태를 의미합니다.

Data 〉 Operator 모듈을 선택합니다.
Operartor Type : Logic으로 선택합니다.
Logical Operator : && 로 선택합니다



43 If 분기문

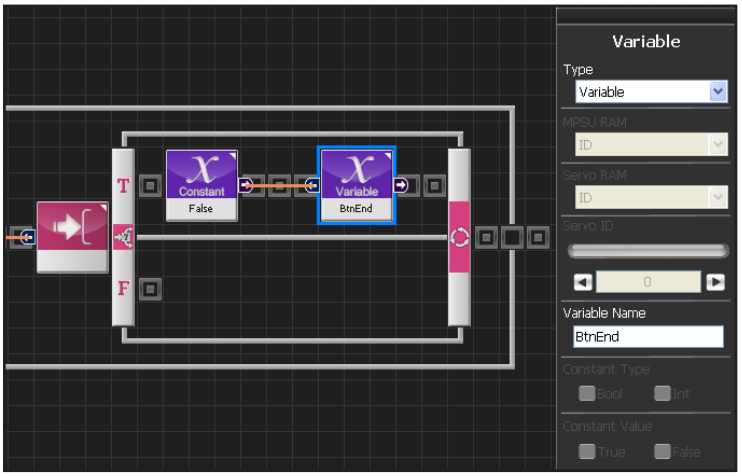
방금 버튼을 눌렀다가 뗀 상태가 True 이면 수행합니다.



44 False 값

BtnEnd 를 True 에서 False 로 만듭니다.

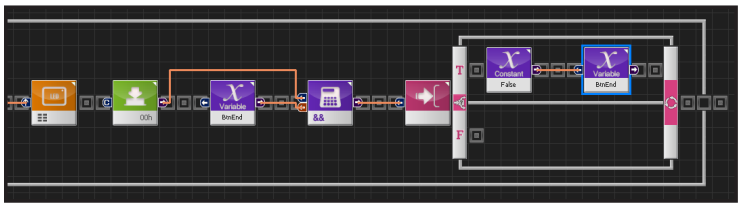
Data 〉 Variable 모듈을 선택합니다.
Type : Contant 를 선택합니다.
Constant Type 은 Bool 로 설정합니다. Bool 은 참과 거짓을 나타내는 자료형입니다.
Constant Value : False 를 선택합니다.



45 BtnEnd를 False로

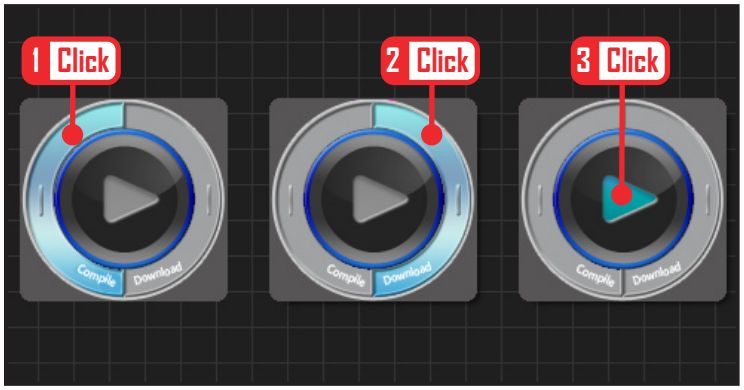
BtnEnd 에 False 값을 입력합니다.

앞의 BtnEnd 변수를 복사합니다.



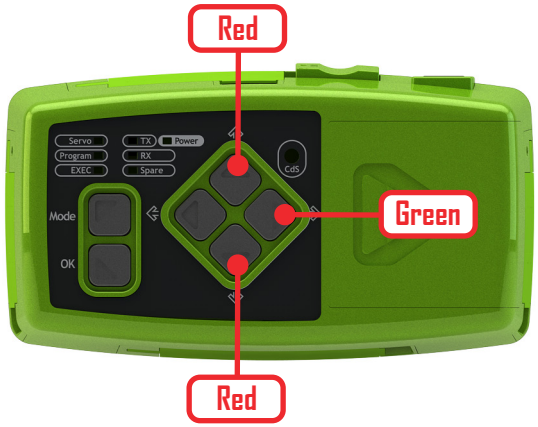
46 버튼 뗐을 때 초기화

방금 버튼을 뗀 상태에 BtnEnd를 false 로 초기화 시킵니다.



47 컴파일, 다운로드, 실행

왼쪽 클릭하여 컴파일 시킵니다. 에러가 없으면 오른쪽 클릭하여 로봇에 다운로드 시킵니다. 다운로드 완료되면 가운데 화살표 실행버튼을 눌러 로봇에서 실행시킵니다.



48 로봇동작

Up 버튼에 Red
Right 버튼에 Green
Down 버튼에 Blue 가 켜졌다 다시 한번 누르면 꺼집니다.