Software : DR-Visual Logic

HOVIS

Programming Individual Module : Button, LED

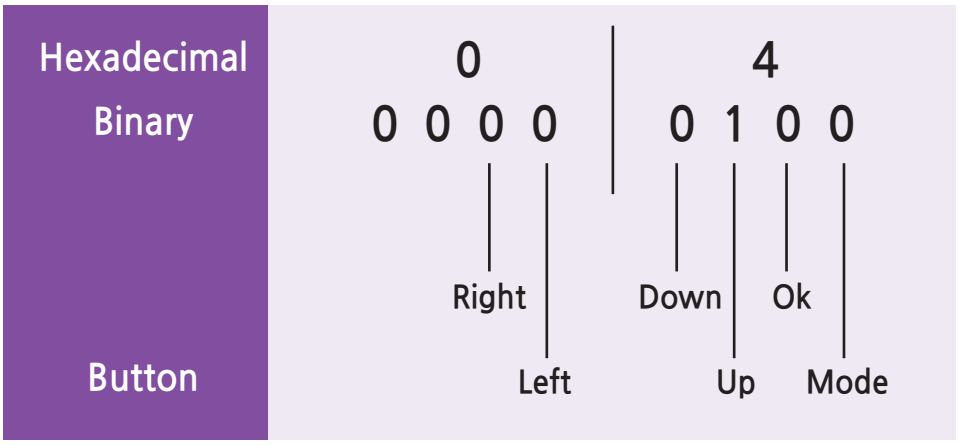
Button, LED Example Step by Step

Example Description

This example uses the buttons on DRC controller to turn on/off LED.

In order to program the button and the LED, user should have an understanding of how the button and the LED work.

$16 = 2^4$



EX) Right

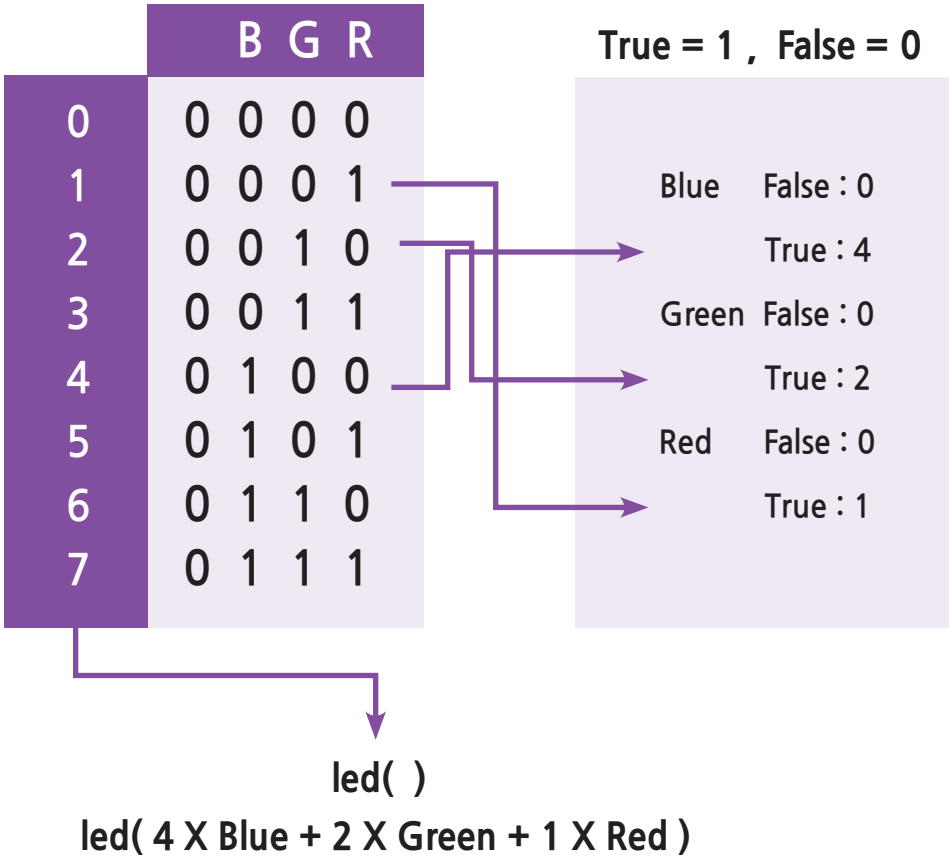
2	0 h
0 0 1 0	0 0 0 0

Down

0	8 h
0 0 0 0	1 0 0 0

Button

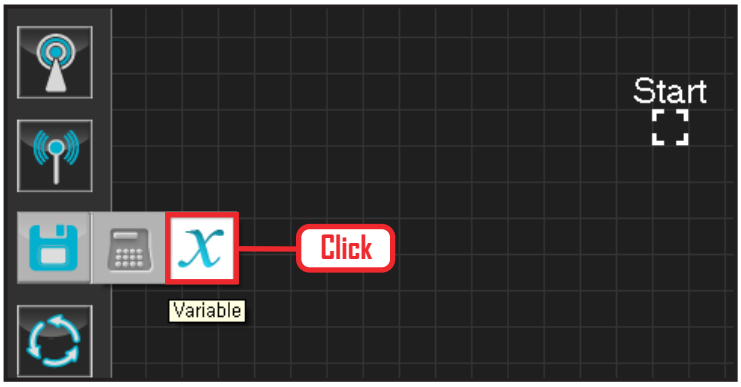
DRC has 6 buttons and pressed button is expressed by a 1 Byte. 1 Byte is made up of 8 Bits so 1Byte is able to carry 8 1s and 0s. 6 bits are required to express pressed (1) and released (0) status of 6 DRC buttons. As shown in the diagram above, each button is matched with a single bit. Pressed button is expressed in 1s and 0s and it is shown in hexadecimal format at bottom right side of the button module. Pressed 'right'; button has value of 00100000 or 20h when converted to hexadecimal format (h refers to hexadecimal). Pressed 'down' button has value or 00001000 or 08h in hexadecimal format. For something more complicated, pressed 'up+down' button has value of 00001100 or 0Ch in hexadecimal format.



LED

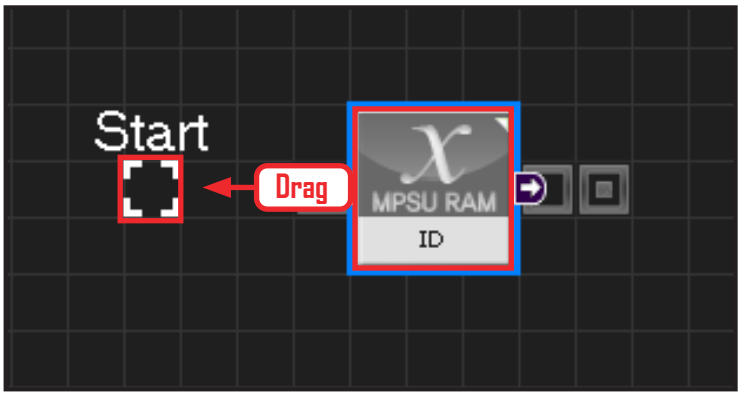
DRC has seven LEDs but only three can be controlled by the task mode. Three bits are required to express on/off status of the three LEDs; Red, Green, Blue. As shown in the diagram above, each LED (Red, Green Blue) is matched with a bit in an ascending order from the lowest bit of the byte to the highest. LED lights up when the LED value is used as an input of the LED module. All LEDs are turned off when the input value is 0(00000000) and they are turned on when the input value is 7(00000111). Blue in binary format is 4, Green 2, and Red 1. When on/off state of each individual LED is determined by the value (true, false) of the variables Blue, Green, and Red, it is possible to control the LEDs by their variable names using 4 x Blue + 2 x Green + 1 x Red as the input of the LED module. For example, when Blue and Green is 'true' and Red 'false, it becomes 4 x Blue + 2 x Green + 1 x Red = 6. 6 in binary format is 00000110. Green and Blue LED will light up when this value is used as an input of the LED module.

Use the basic principals from above to program the Buttons and LEDs.



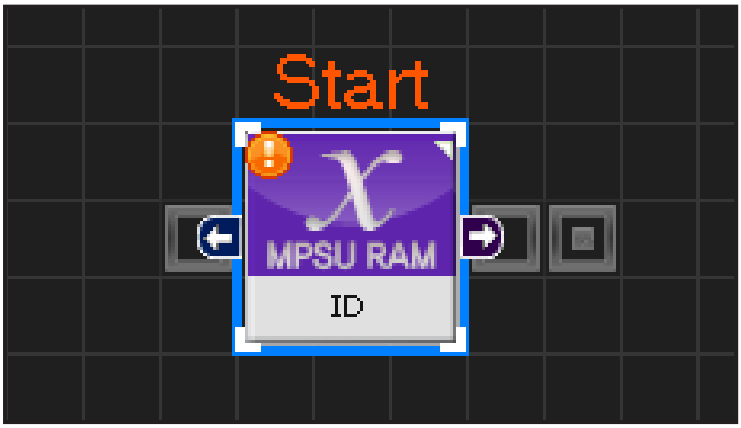
01 Assign Variable

Click Data 〉 Variable module.



02 Start

Click and drag the connecting line located at left side of the module to the Start Point and dock.

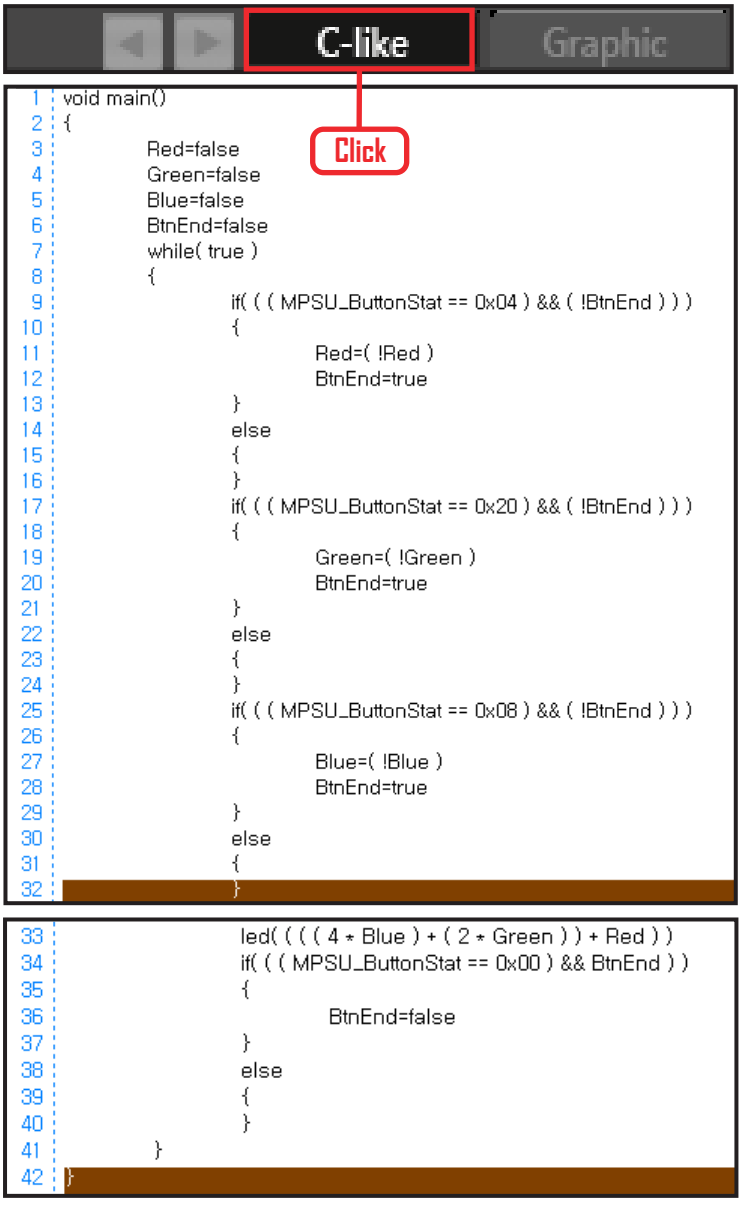


03 Start Programming

When the module and the Start Point is docked properly, module will become active and change color as seen in the photo to the left.
This means programming has started.

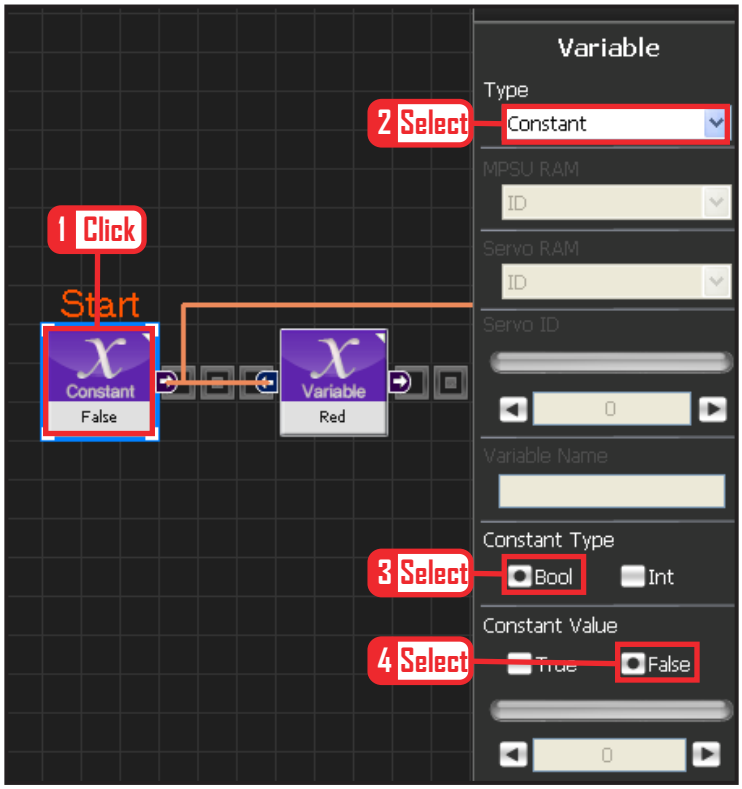
04 Entire Program

Entire program using the buttons and LED.



05 Viewing C-Like

Click the 'C–like' tab near the top right and task programming window will open as shown in the photo to the left. This is the task window of the entire program. Codes are very similar to the C language structure so studying the codes will help the user become familiar with the C language structure. Cursor will jump follwing the clicked module, making it easy to see the module changing to text.

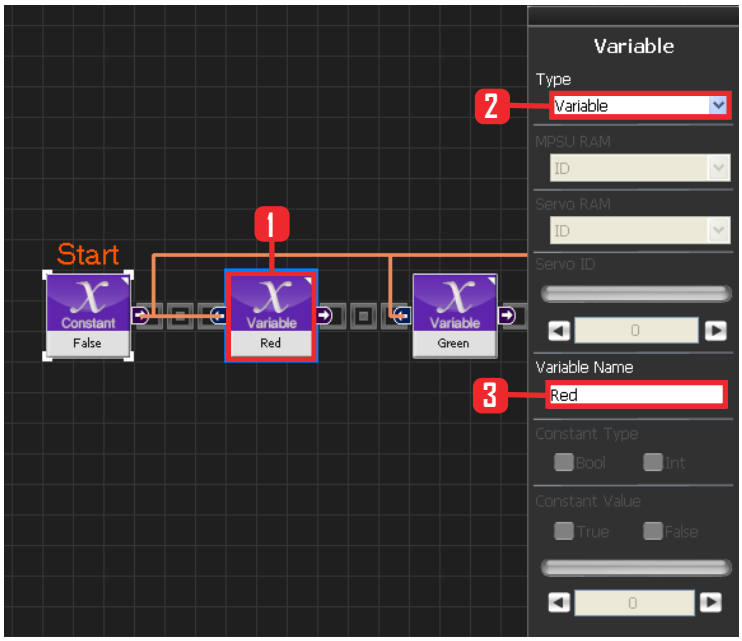


06 Initialize as False

All LEDs are initialized False (Off).

Select Data 〉Variable .
Select Type : Contant .
Select Constant Type Bool . True or False data type.
Select Constant Value : False

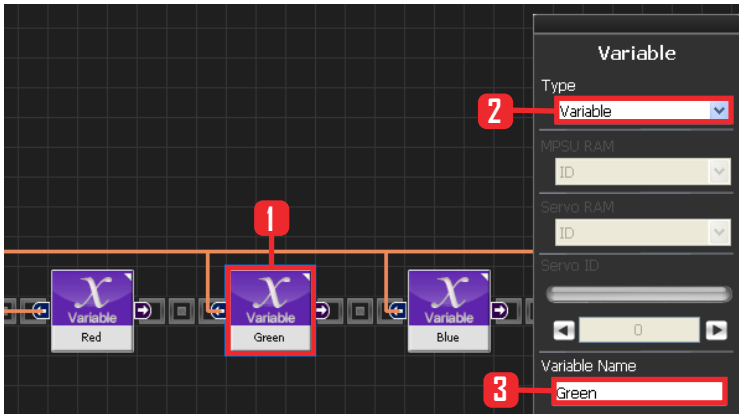
Use the connector to connect False to the variables.



07 Red Variable

Select Data 〉Variable .
Select Type : Variable .
Variable Name : Red .

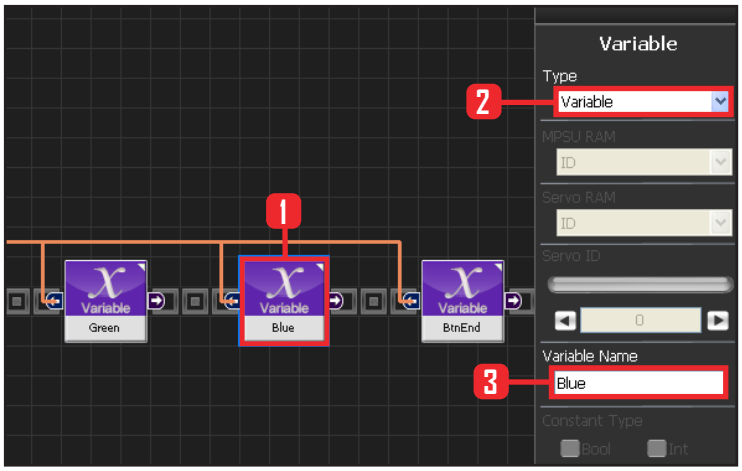
Red LED off when False, on when True.



08 Green Variable

Select Data 〉Variable .
Select Type : Variable .
Variable Name : Green .

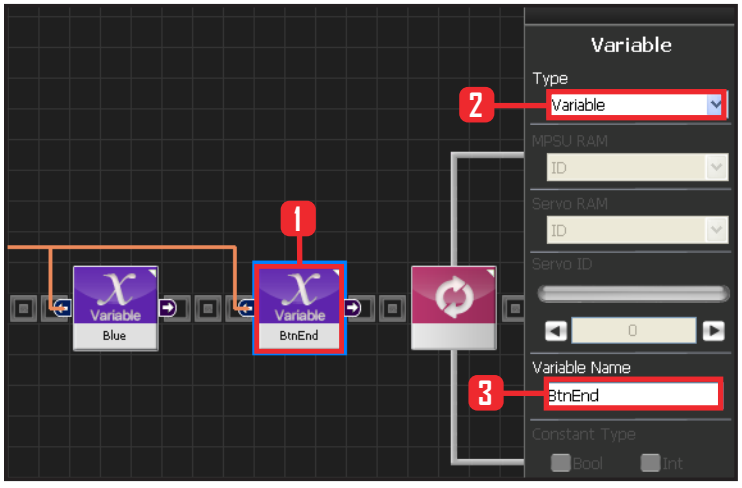
Green LED off when False, on when True



09 Blue Variable

Select Data 〉Variable .
Select Type : Variable .
Variable Name : Blue .

Blue LED off when False, on when True



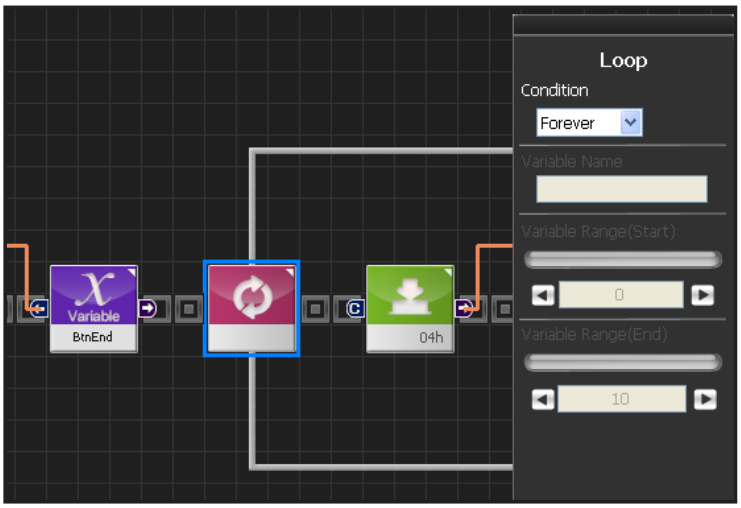
10 BtnEnd Variable

BtnEnd Variable maintains 'False' value while the button remains released but changes from False 〉True as soon as the button is pressed and the motion ends.
Value changes back from 'True' to 'False' as soon as the button is released.

Select Data 〉Variable .
SelectType : Variable .
Variable Name : BtnEnd .

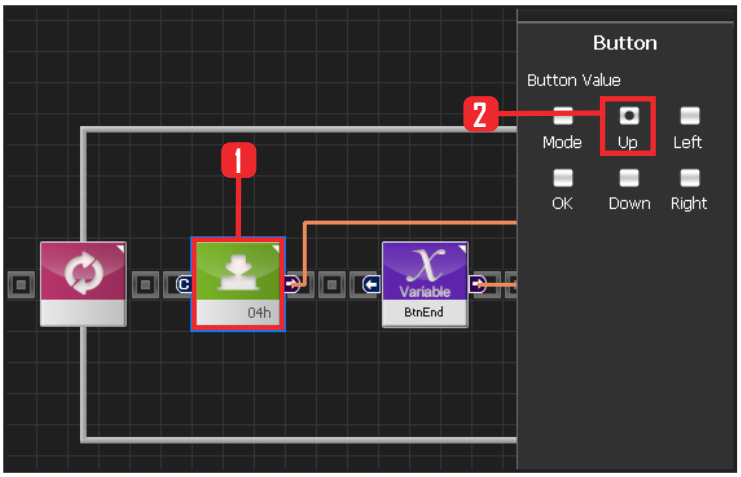
11 Assign Variable

Assign False as the initial vlaue of Red, Green, Blue, BtnEnd.



12 Loop

Forever infinite repetition.

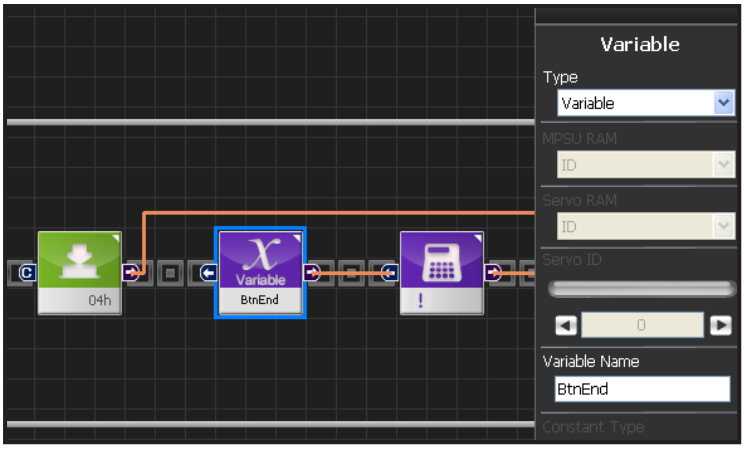


13 Up Button

Create a button module. This module becomes 'True' when the selected button is pressed and 'False' under other conditions. When Up Button is selected, 'True' when Up Button is pressed and 'False' under other conditions.

Select Communication 〉Button module.
Set Button Value : Up .

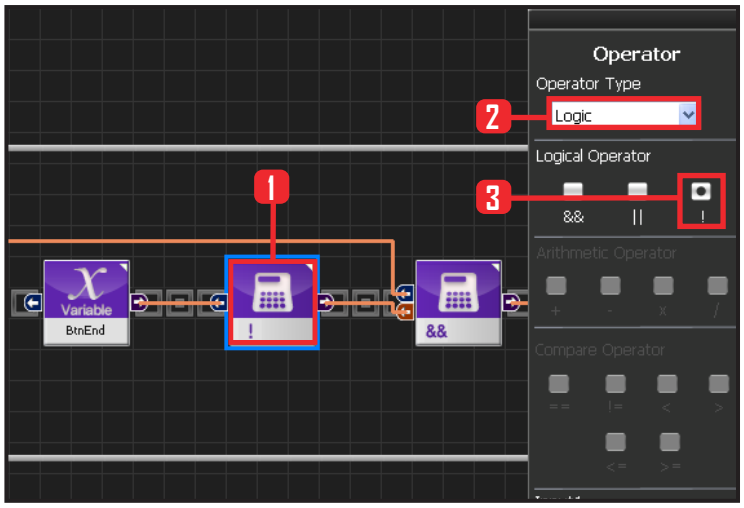
Value is 04h in hexadecimal format. 04h will be shown in the module.



14 BtnEnd

BtnEnd value is intialized as false. It beocmes True with 'not' opertaor attached to the back.

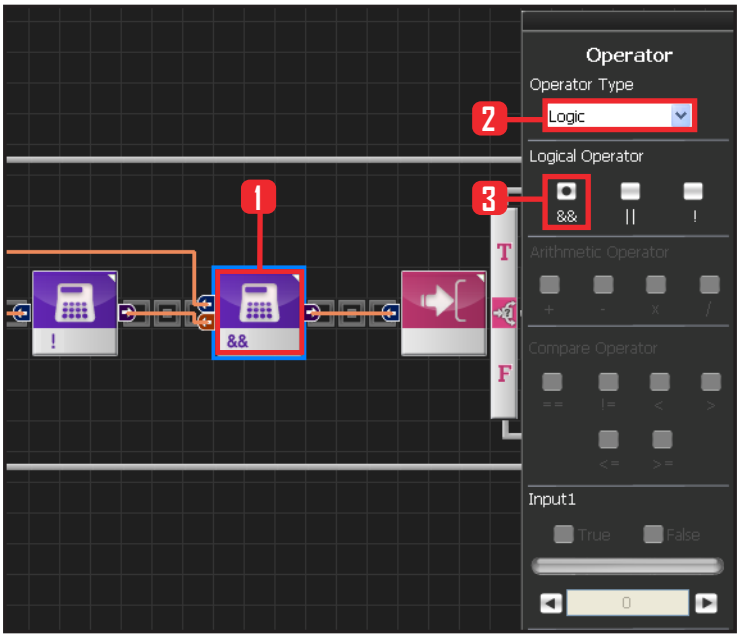
Copy and paste the btnEnd variiablef from the front.



15 ! Operator

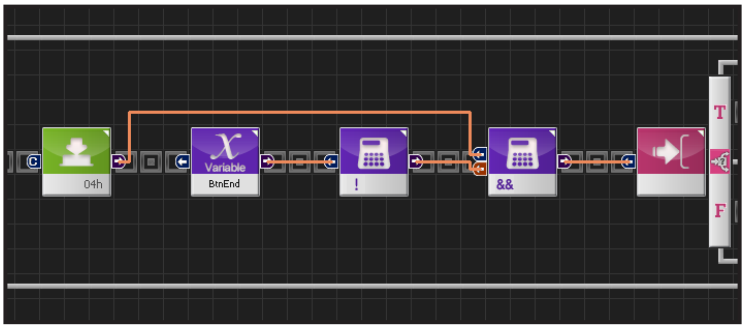
Use ! operator to change the BtnEnd value to the opposite.

Select Data 〉Operator module.
Select Operartor Type : Logic.
Select Logical Operator : ! .



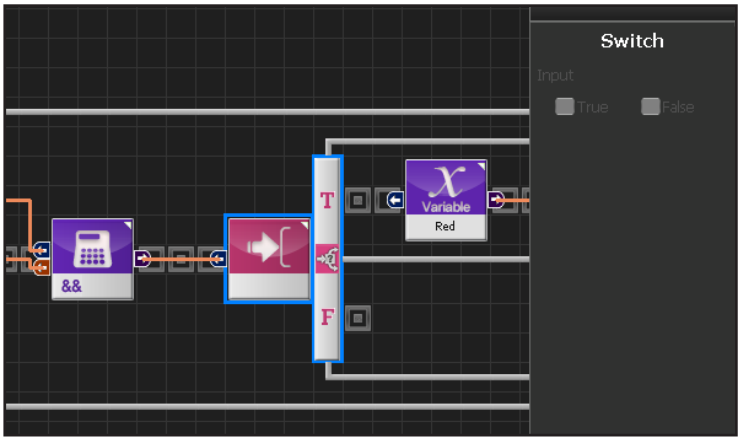
16 And Operator

When Up button is pressed, BtnEnd false (Becomes True by applying !) becomes True and executes the conditional statement behind.
Select Data 〉Operator module.
Select Operartor Type : Logic.
Select Logical Operator : && .



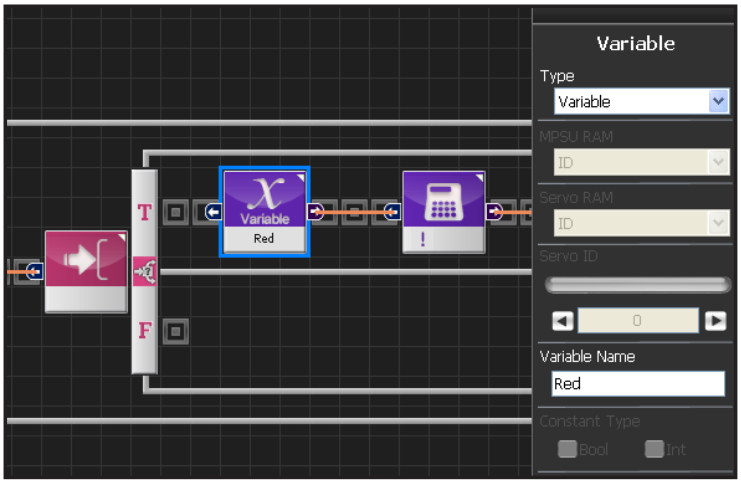
17 Up Button Pressed

When Up button is pressed and BtnEnd is false, condition behind is executed.



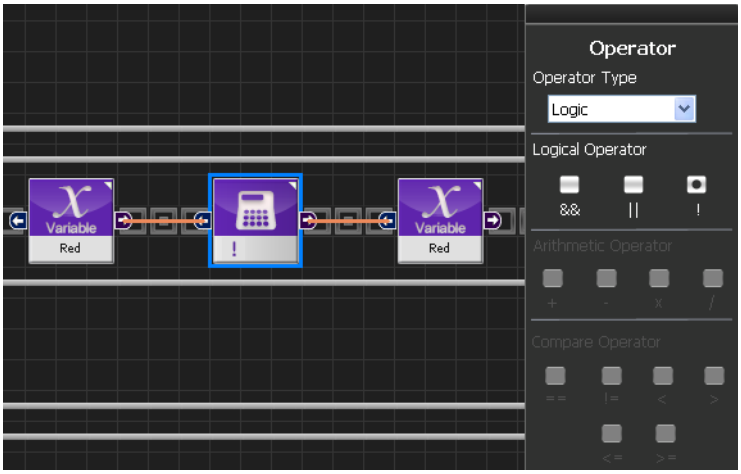
18 If Switch

Runs the upper part when True.



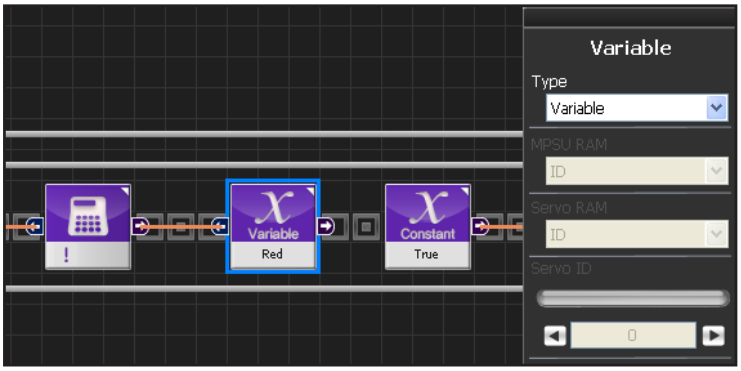
19 Red Output

Copy and paste Red variable from front.



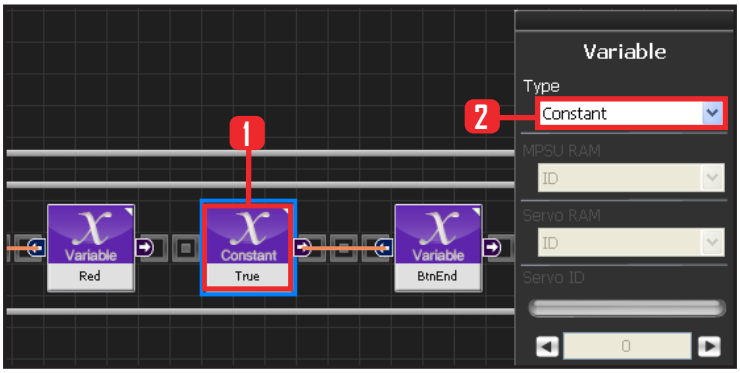
20 ! Operator

When Red is True it becomes False and vice versa.



21 Red Input

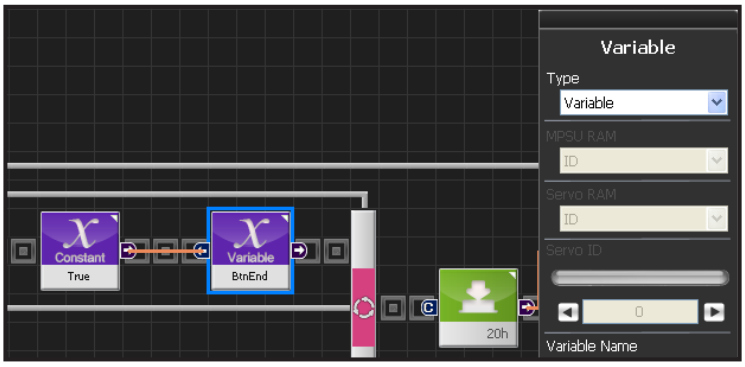
When Red variable value is true it becomes false and vice versa. Changed value is saved.



22 True Setup

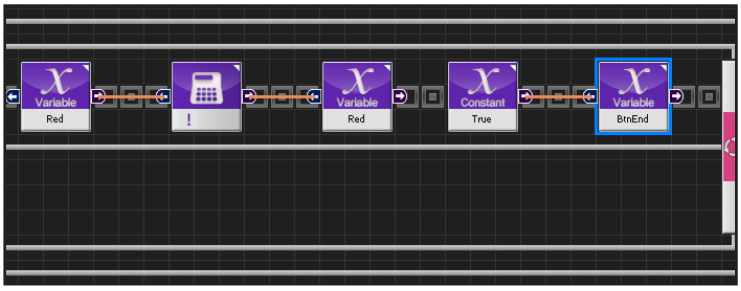
With the programmed motion been finished after press button, the BtnEnd should be changed from false to true.

Select Data 〉Variable module.
Select Type : Contant.
Select Constant Type: Bool
Bool: True or False data type.
Select Constant Value : True



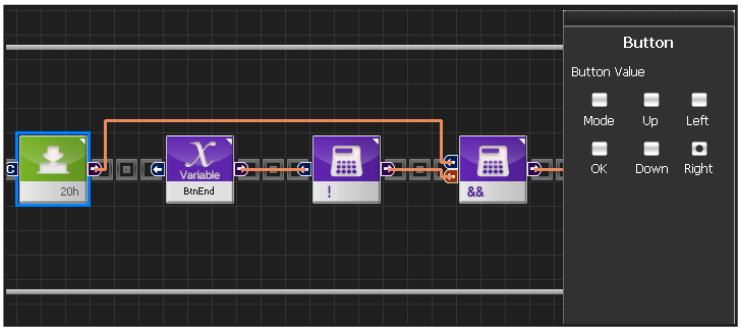
23 BtnEnd to True

Input True value in the BtnEnd .
When BtnEnd value is true and loop is running, pressed up button will not satisfy the conditional statement and Red variable value will not change further.



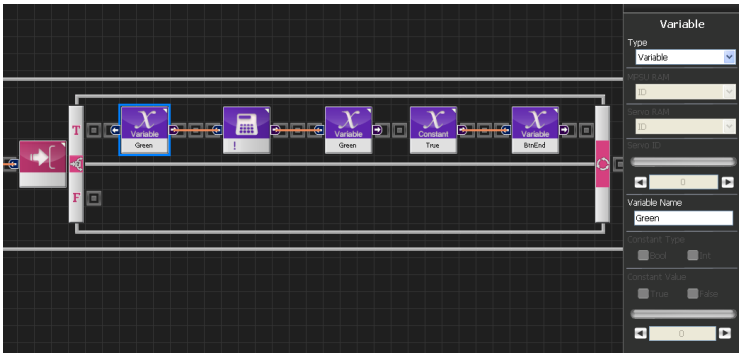
24 Red LED

Red LED will light when up button is pressed once and go off when it is pressed once more.



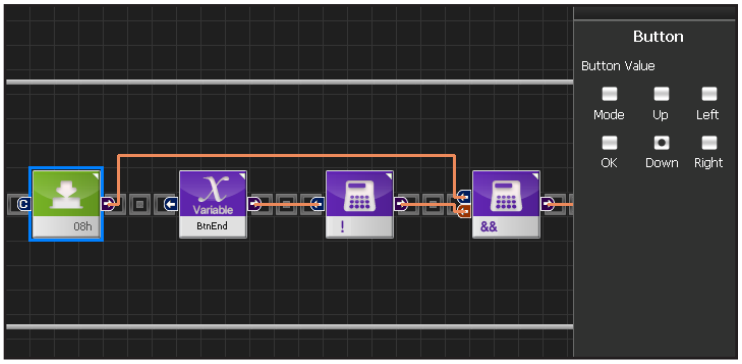
25 Right Button

When Right is pressed.



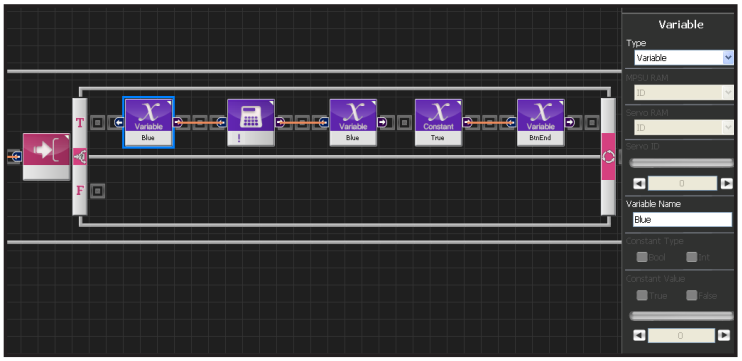
26 Green LED

Green LED will light when right button is pressed once and go off when it is pressed once more.



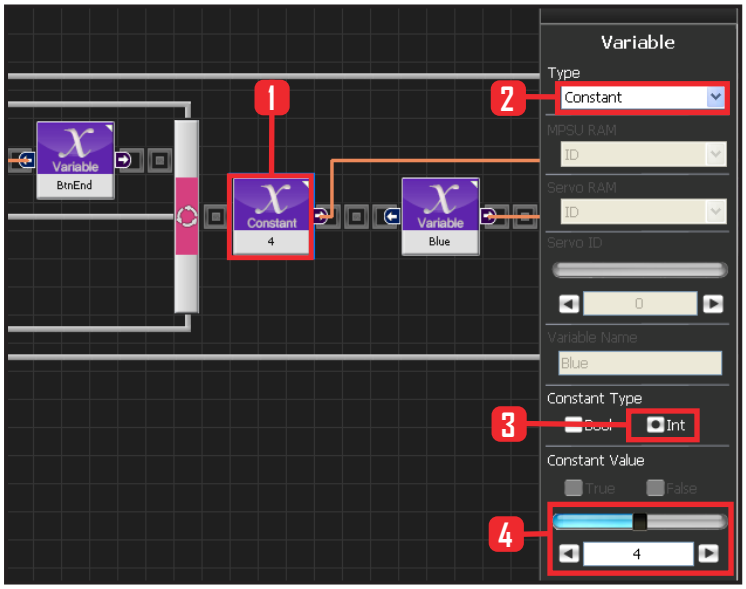
27 Down Button

When down button is pressed.



28 Blue LED

Blue LED will light when right button is pressed once and go off when it is pressed once more.



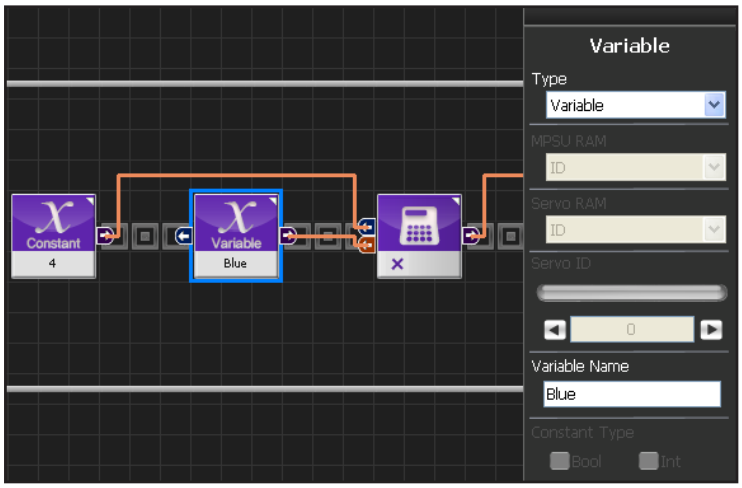
29 LED Value

As explained above, LED lights up depending on the input value of the LED module. Diagram on the left shows the connected modules according to the input formula (4 x Blue + 2 x Green + 1 x Red)

Set constant value : 4 .

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

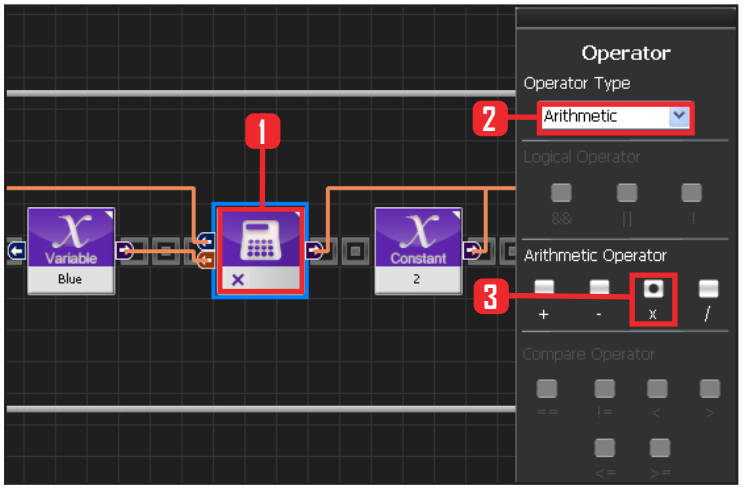
Select Data 〉Variable module.
Select Type : Contant .
Select Constant Type: int .
Set Constant Value : 4.



30 Blue

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

Copy the Blue variable from front.

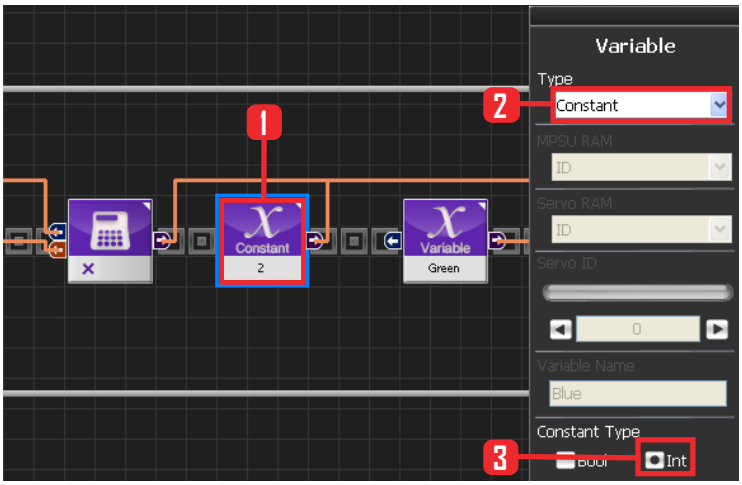


31 Multiplication

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

Slect Data 〉Operator module.
Select Operartor Type : Arithmetic.
Select Arithmetic Operator : X .

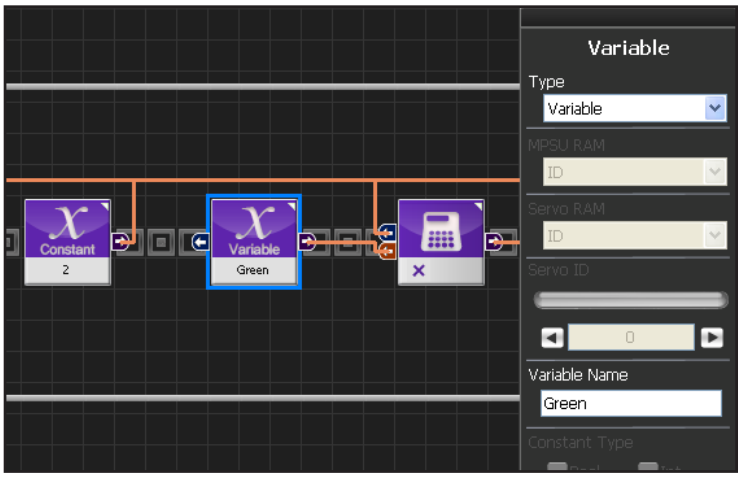
Connect constant 4 and Blue variable module to the two input connectors of the multiplication module.



32 Constant 2

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

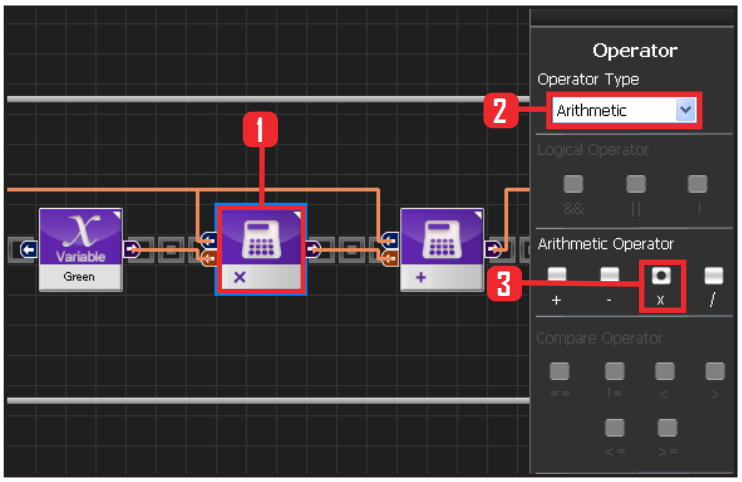
Select Data 〉Variable module.
Select Type : Contant .
Select Constant Type: int .
Set Constant Value : 2.



33 Green

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

Copy the Green variable from front.

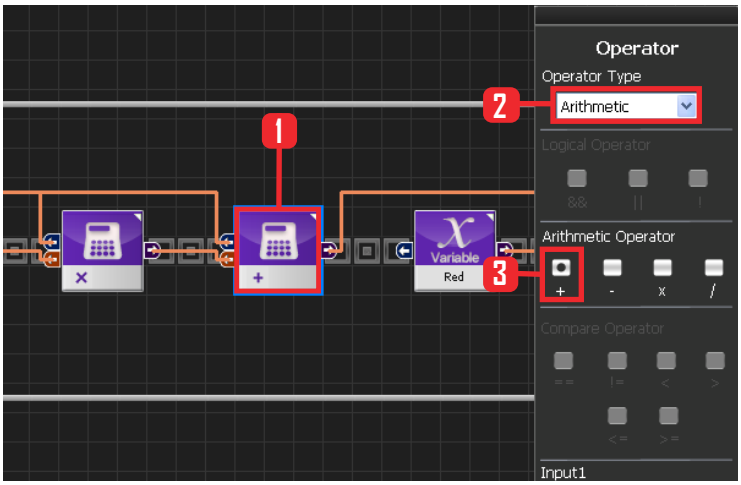


34 Multiplication

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

Slect Data 〉Operator module.
Select Operartor Type : Arithmetic.
Select Arithmetic Operator : X .

Connect constant 2 and Green variable module to the two input connectors of the multiplication module.

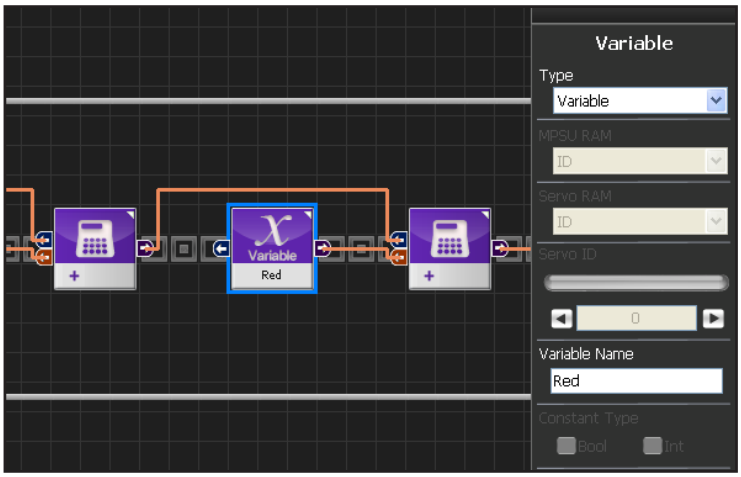


35 Addition

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

Select Data 〉Operator module.
Select Operartor Type : Arithmetic.
Select Arithmetic Operator : + .

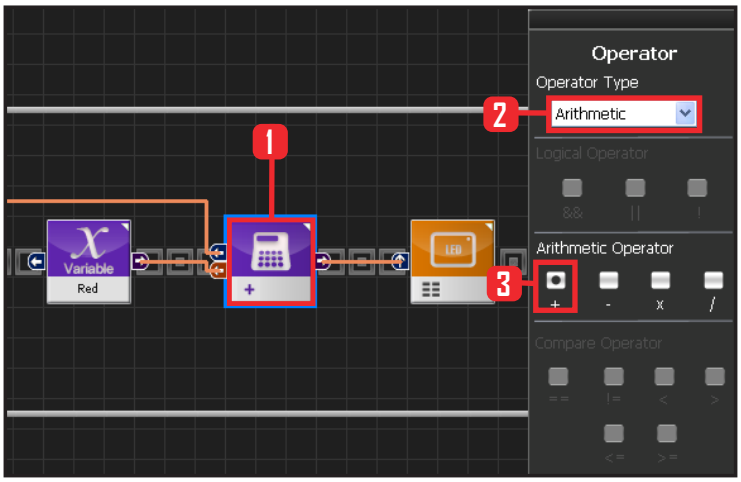
Connect the output from the multiplication modules in #31 & 34 to the two input connectors of the addition module.



36 Red

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

Copy Red variable from front.

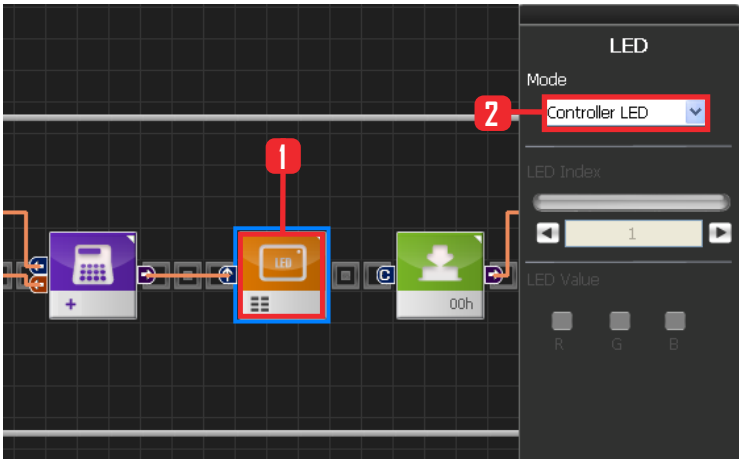


37 Addition

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

Slect Data 〉Operator module.
Select Operartor Type : Arithmetic.
Select Arithmetic Operator : X .

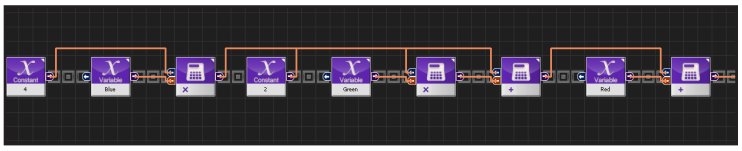
Connect output from the addition module in #35 and Red variable module to the two input connectors of the addition module.



38 LED

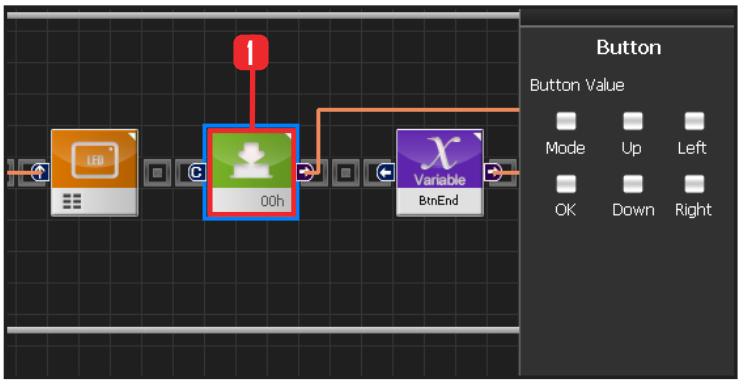
Select Motion 〉LED module.
Select Mode : Controller LED .

Input the values from the previous calculations into the LED value to turn on each individual LED.



39 LED Value Output Calculation

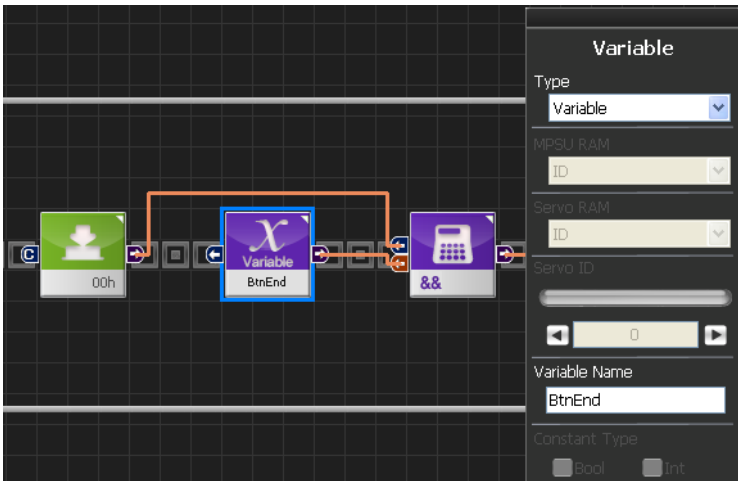
(4 x Blue + 2 x Green + 1 x Red)
Shown as connected modules.



40 Button Released State

When the motion associated with the button press ends, BtnEnd variable changes to True. Since the motion associated with the button press does not run when BtnEnd variable is True, it is possible to make single button press run the associated motion only once. BitEnd variable has to be initialized to False when the button is released. Following program shows how to ini–tialize the button.
Select Motion 〉Button module.
Button Value : None.

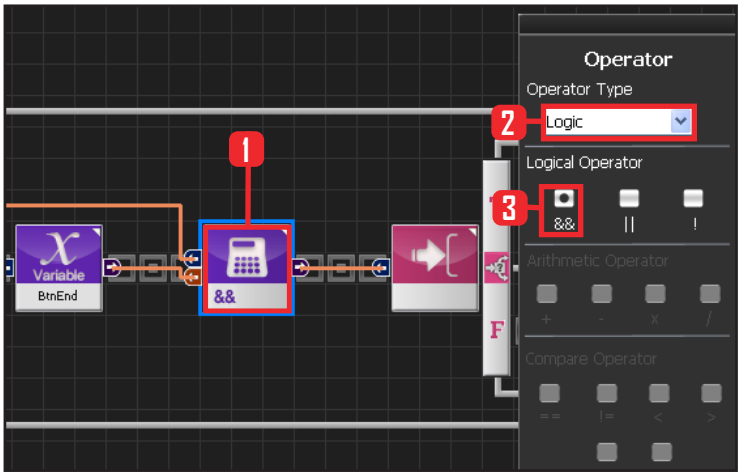
Released button state.



41 BtnEnd is True

When BtnEnd is True .

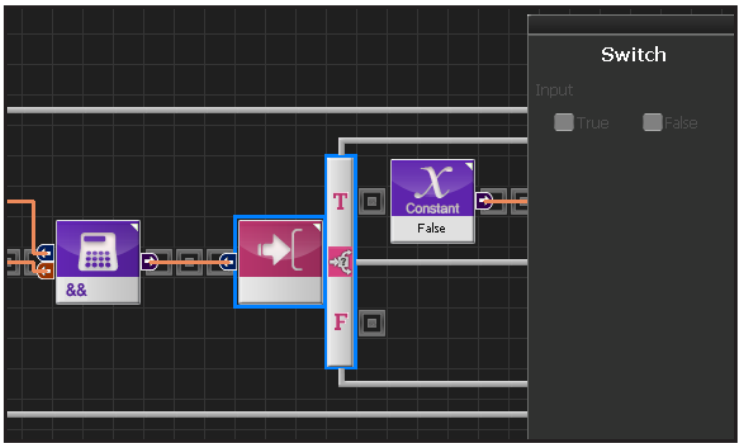
Copy BtnEnd variable from front.



42 && Operator

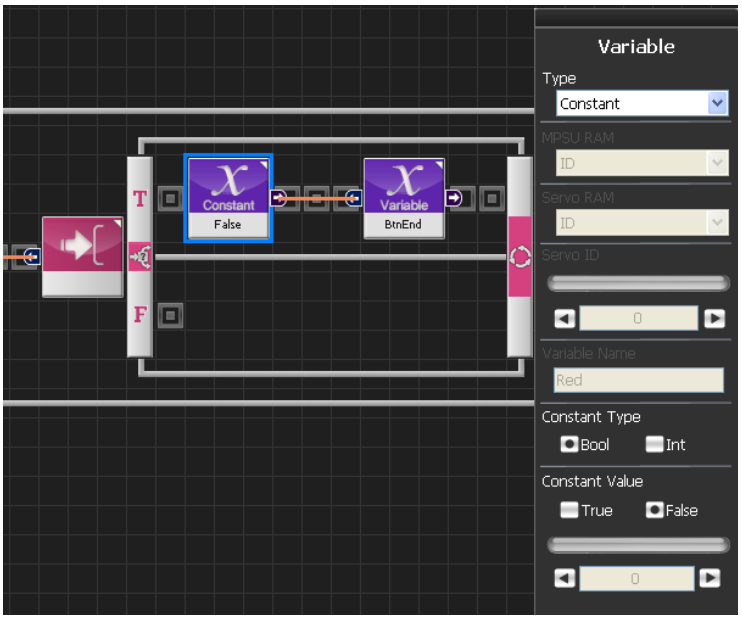
Just released button state satisfies both released button state and BtnEnd True .

Select Data 〉Operator module.
Select Operartor Type : Logic.
Select Logical Operator : &&



43 If Conditional Statement

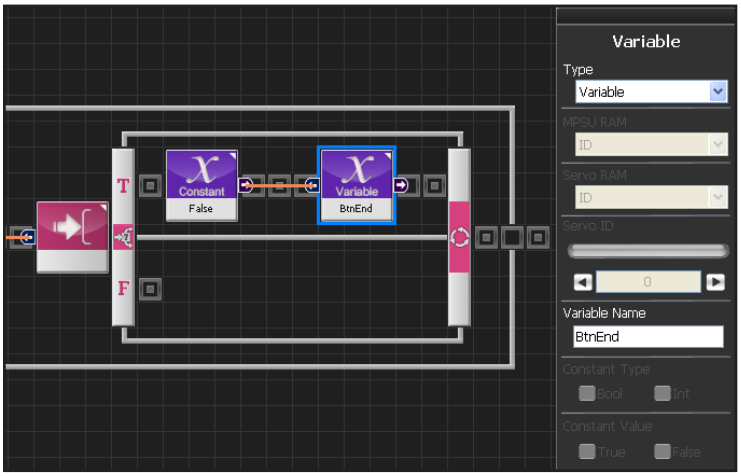
Run if just released button state is True.



44 False Value

Change BtnEnd from True to False.

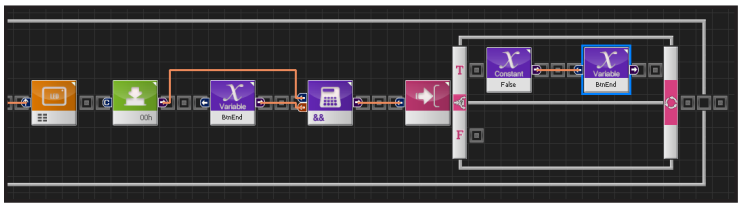
Select Data 〉Variable module.
Select Type : Contant .
Select Constant Type : Bool
Bool: True or False data type.
Constant Value : False.



45 Change BtnEnd to False

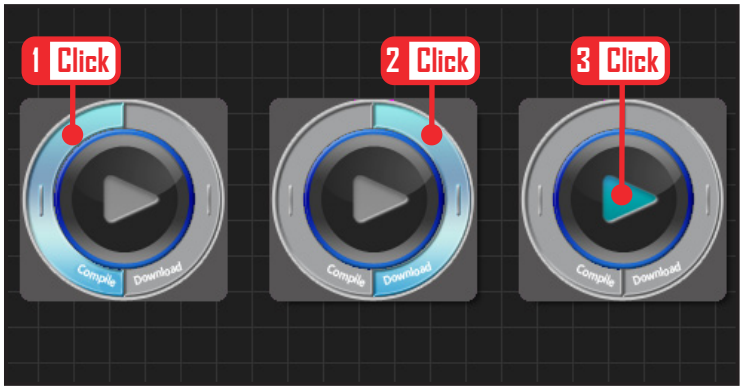
Input False value to BtnEnd .

Copy BtnEnd variable from front.



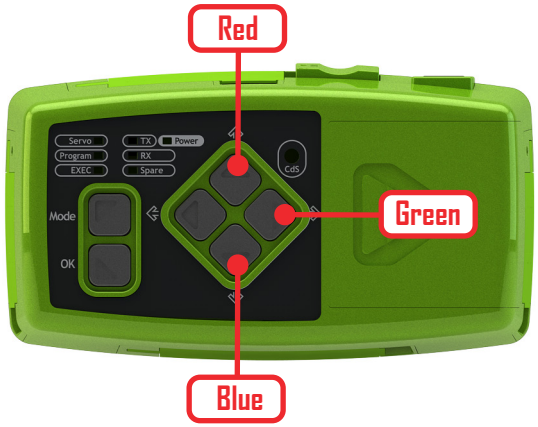
46 Initialize Button When Released

From the just releaed button state, initialize Bt–nEnd to false .



47 Compile, Downolad, Run

Click 'Compile'. Click 'download' on the right if there is no compilation error. Download to ro-bot. Click 'Run' button (Arrow button) after the download.



48 Robot Motion

Up button: Red
Right button: Green
Down button: Blue
LEDs will light up when pressed and go off when pressed once more.