Easy Learning Using Icons

HOVIS DRC & Visual Logic Robot Programming

- Easy programming even for the novices using Drag & Drop method
- Learn C language grammar using C-Like

Dongbu Robot Archive

(Download manuals & applications.)

www.dongburobot.com

Dongbu Robot

Dongbu Robot

Easy Learning with Icons
Visual Logic
Robot Programming

1st Edition published date :
Publisher : Dongbu Robot Co., Ltd
Address : 11th Fl Bucheon Techno Prk Bldg, 401, Yakdae-dong, Wonmi-gu, Bucheon-si, Gyeonggi-do, S Korea
Customer Inquiry : 032-329-5551
FAX : 032-329-5569

Contents

PART 01

Donbu Robot DRC & Humanoid

Chapter 01

Introouduction

Chapter 02

Controller

Chapter 03

Parts List

Chapter 04

Assembly Diagram

PART 02

DR–Visual Logic Programming

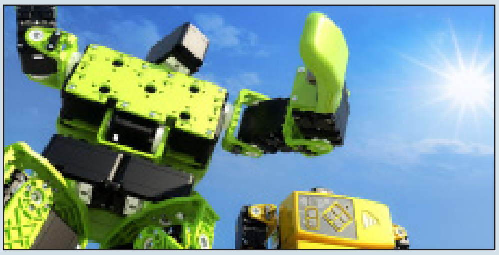
Appendix

PART 01

Donbu Robot
DRC & Humanoid

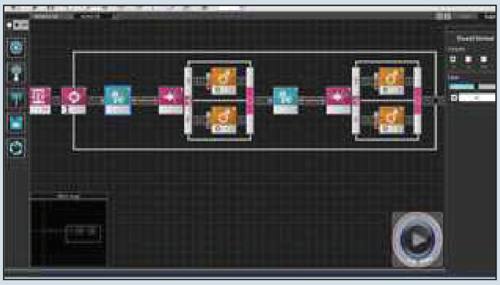
Hovis Lite Introduction

Introducing Hovis Lite.



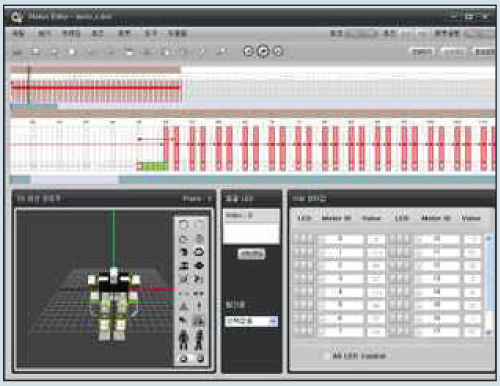
Choice of Colors

Use four different colored brackets to create your own unique humanoid robot. Hovis Lite is the first robot in the world that can be upgraded with external body case, omni wheel, and android terminal.



DR-Visual Logic (Task Editor)

To program the robot based on the controller (DRC), Hovis Lite is supplied with 24 modules and a graphic programming language tool DR-Visual Logic that uses drag & drop method. Even the novice users without any prior knowledge of programming language would find DR-Visual Logic easy to use.



DR-SIM (Motion Editor)

DR-SIM is a robot motion editor that incorporates 'time frame' feature found in the video editors. DR-SIM allows the user to create robot motions on screen, to capture motions from the robot, view user created motion simulations on screen, and to download and apply the created motion to the robot for execution.

- Choice of four different colored brackets
- Assemble up to 27 different types of robot
- Upgradable by external body case type
- Android terminal and programming interface included
- Source supplied
- Curriculum supplied

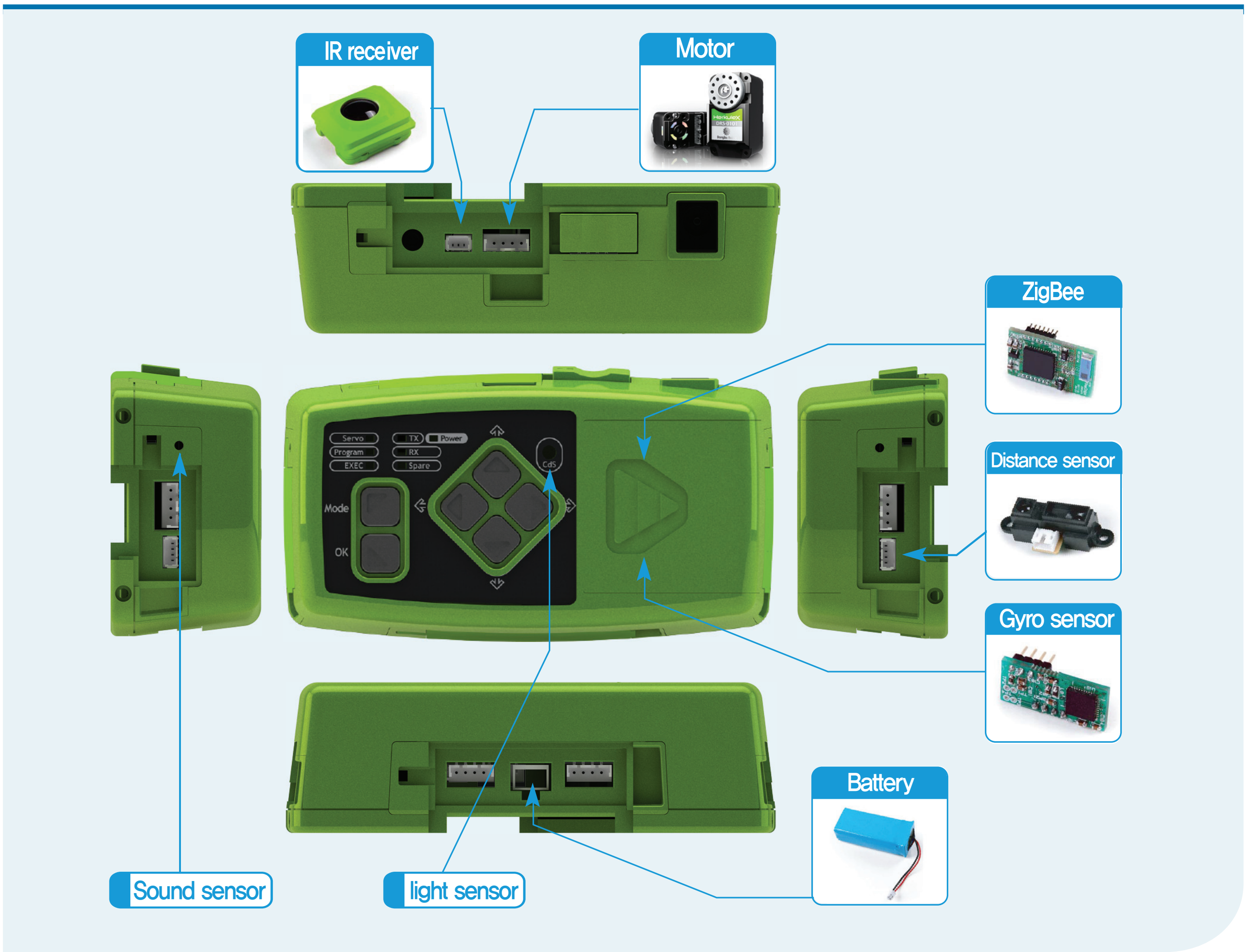
Related Site

Download Program/Manual :
www.hovis.co.kr/guide
www.dongburobot.com → English → Customer support → For Services → Archives
www.dongburobot.com → Quick Menu → For Service → Archives

PART 01 Controller

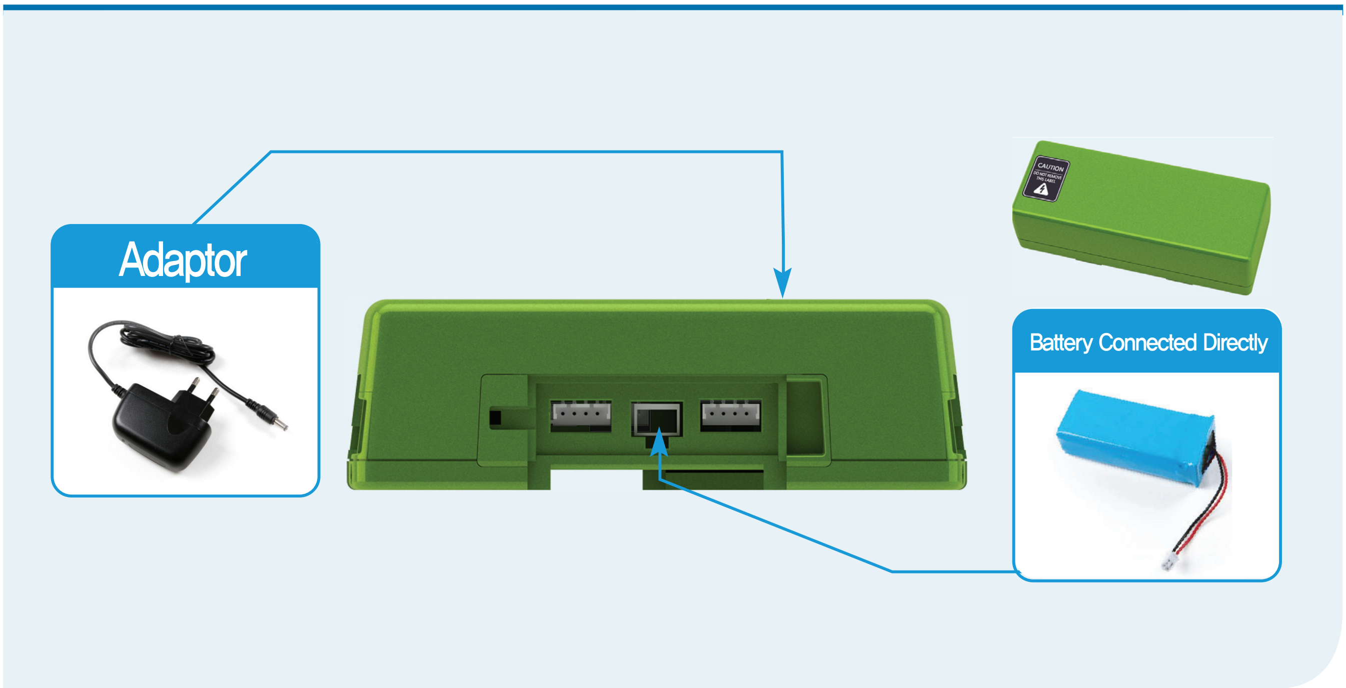
Overview

DRC Controller is the main component and brain of the Hovis Lite. Controller has variety of connectors and interfaces including 6 servo motor ports, 2 PSD sensor ports, Gyro sensor, and ZigBee interface. Light sensor and the sound sensor in built-in to the controller. DR-Visual Logic program is a visual robot programming language that uses DRC functions to program the robot. Various sensors and 1~32 motors can be programmed and tested.



CPU	ATMega 128
Size, Weight	108 x 58.5 x 33 (mm), 82 g
Operating Voltage	Tolerance Range : 6.5V ~ 10V, Recommended Voltage : 7.4V
Serial Speed	115,200 bps ~ 666,667 bps
Consumed Current	When IDLE : 50mA, Overall Max Current : 3A (PTC Fuse)
Interface	Button : 6ea, MIC : 2ea, LED : 7ea
External I/O	Servo Motor : 6ea, PSD Sensor : 2ea
Back Cover I/O	ZigBee : 1ea, Gyro Sensor : 1ea
Internal I/O	Sound Sensor : 2ea, Light Sensor : 1ea

DRC Battery Installation

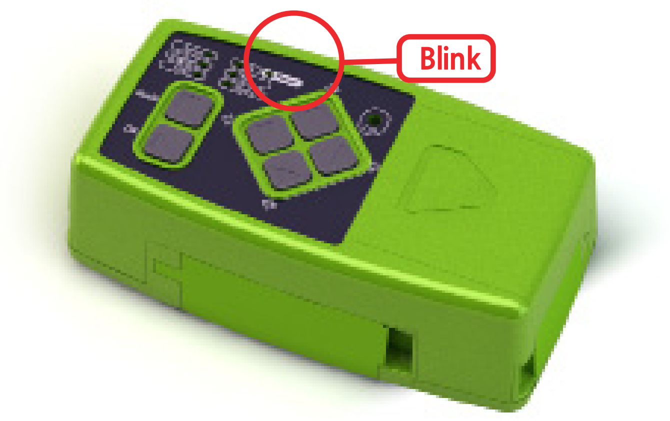


Battery

To supply power to the DRC, connect the battery to controller by the power connector found at bottom.

Battery Charging

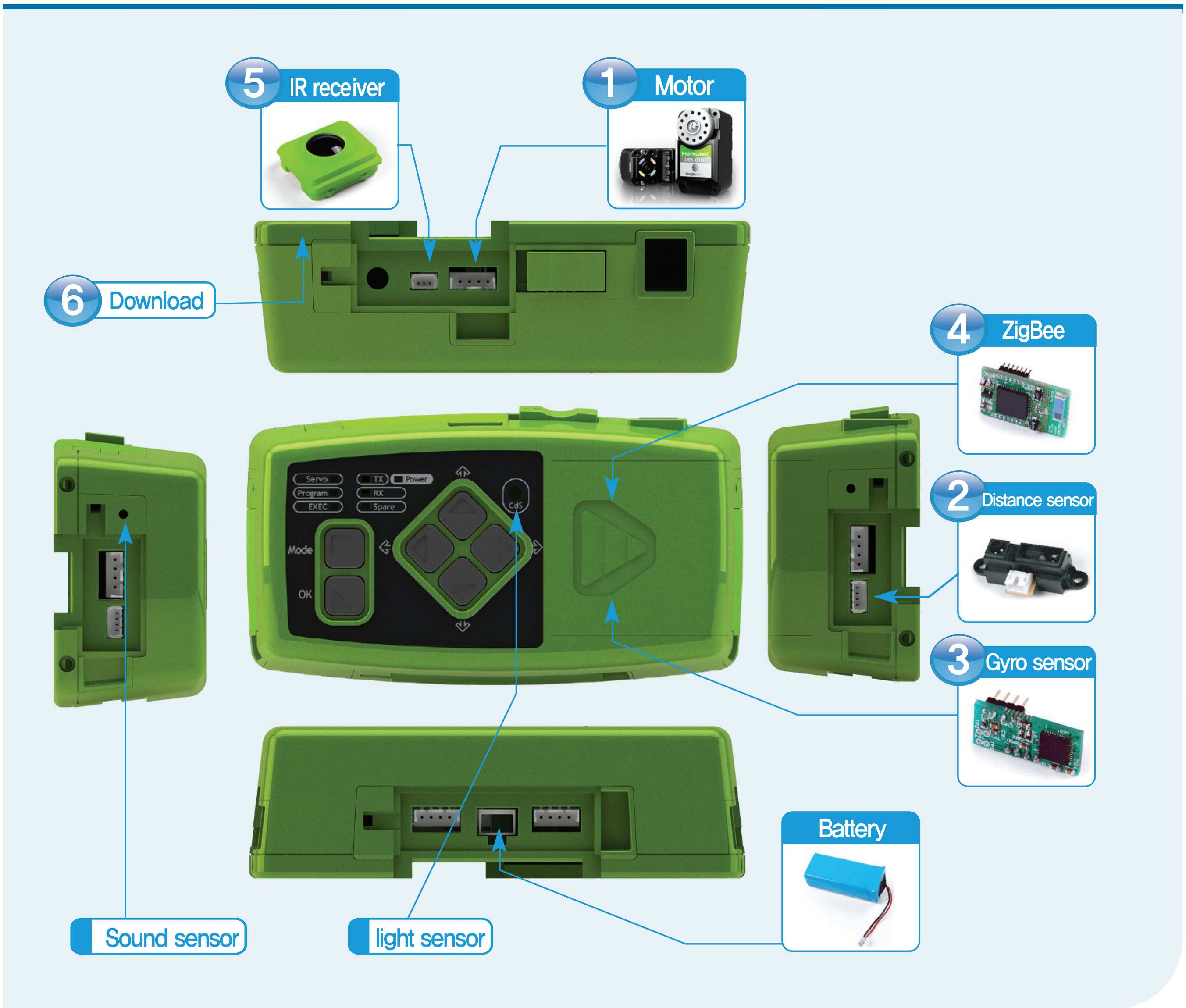
Battery can be charged directly using the battery cable and the adapter. Another charging method is to connect the adapter to the adapter connector found at top of the controller. Battery installed on the controller will start to charge automatically when the adapter is connected to the controller.



Low Battery

Power LED will start to blink when the battery level falls below 20%.

DRC Connection



1 Motor Connection

There are five ports around the controller

2 Distance Sensor Connection

There are two ports, one on each side.

3 Gyro Sensor Connection

Open the controller cover and install internally.

4 ZigBee Connection

Open the controller cover and install internally.

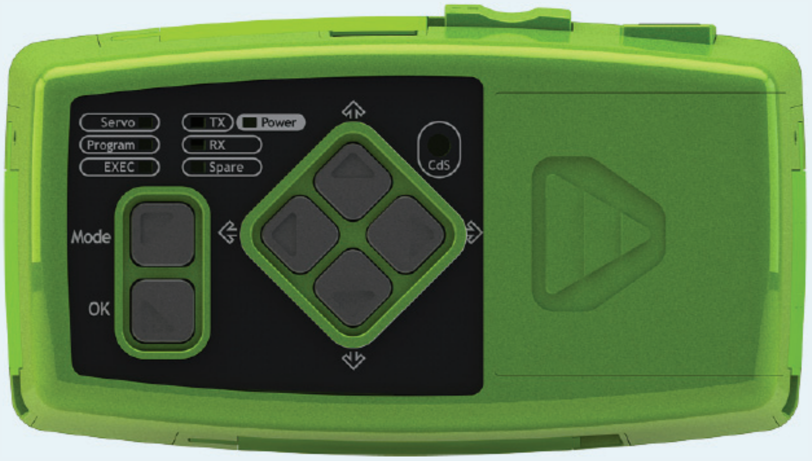
5 IR Receiver Connection

1 port, used for receiving remote control signal.

6 Download Connection

Ear phone Jack connection used to downolad program from PC to the robot.

DRC Interface



DRC controller has Input/Output buttons and LEDs at the front and motor and sensor ports at the back. Interface buttons at the front are used to give input commands the LEDs are used to verify data output.

	Name	Short Cut	Standard Task Mode
Main Button	Mode	Run Task	
	Ok	Confirm	
Navi Key	(Left)	Battery Level	Check Mode
	(Up)	DRC Self Test	Autonomous Movement Mode
	(Right)	Switch wired/wireless com	Remote Control Mode
	(Down)	Motor ID Scan	Sound Demo Mode
LED	Servo	HerkuleX Running	
	Program	DR-SIM/Visual Logic Running	
	EXEC	Task Running	
	TX	Data Transmit	
	RX	Data Receive	
	Spare	User Defined	
	Power	Power	
Sensor	Cds	Light Sensor	

DRC Register Map

Register

DRC has a registers which contains current controller state, settings, and various sensor related data.

For example, number of motors connected to the robot and their ID, error status, and current error codes are all part of current controller state. Controller settings include such data as Min/Max input voltage, Ack Policy, and etc. Sensor readings such as luminosity detected by the light sensor and location of the detected sound are part of sensor data.

Controller register is divided into (Non-Volatile, EEPROM) register and (Volatile, RAM) register. Non-Volatile registers retain data even when the power has been turned off and contain basic sestup values pertaining to the controller operation. Values in the Non-Volatile registers are copied to Volatile registers as soon as the power is turned on. Volatile registers contain controller settings, state, and sensor values. Data in the Volatile registers have direct effect on the operation of the controller.

Knowing the content of the the registers and how the content changes allow the user to write more refined robot motion progam using DR-Visual program. Knowledge about registers also help the user to read the the controller status and to change the operatonal settings, making robot operation more convenient.

Protocol

Protocol is a predefined format or rules for commands that are given to read or write to registers. Protocol is defined not only for read/write commands but also for other commands such as run commands for running saved tasks or sounds , reboot command for rebooting the controller, and host of other commands.

Communication between the PC and the controller use such predefined protocols to send and receive packets. DR-SIM and DR-Visual prgrams provided by Dongbu Robot were also created using such protocols. User should become familiar with the protocols in order to controll the DRC using their custom made programs.

Refer to DRC Regisers and Protocols section in the manual for more information.

PART 01

Controller

DRC Funcitons

1 Program Overview

Firmware : Internal program that cannot be modified by the user.

Task : User defined prgram that can be modified using the Task Editor (DR-Visual Logic).

At the time of release, basic humanoid Task Program is defined. Program can be modified by the user.

2 Operating Method

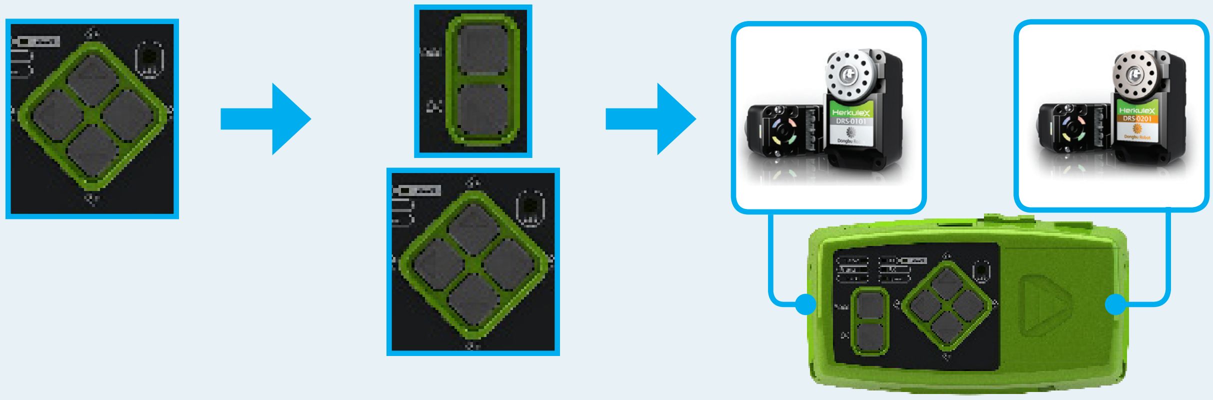
Basic functions in the firmware will start to operate when power is turned on and Navi key pressed. Pressing the Mode button will run the saved task. From running a basic task, press the Navi key and OK after the task to select which mode to go into

Firmware (Cannot be changed by the user)

1 Navi button click

2 Mode or Navi click

3 Controller self test



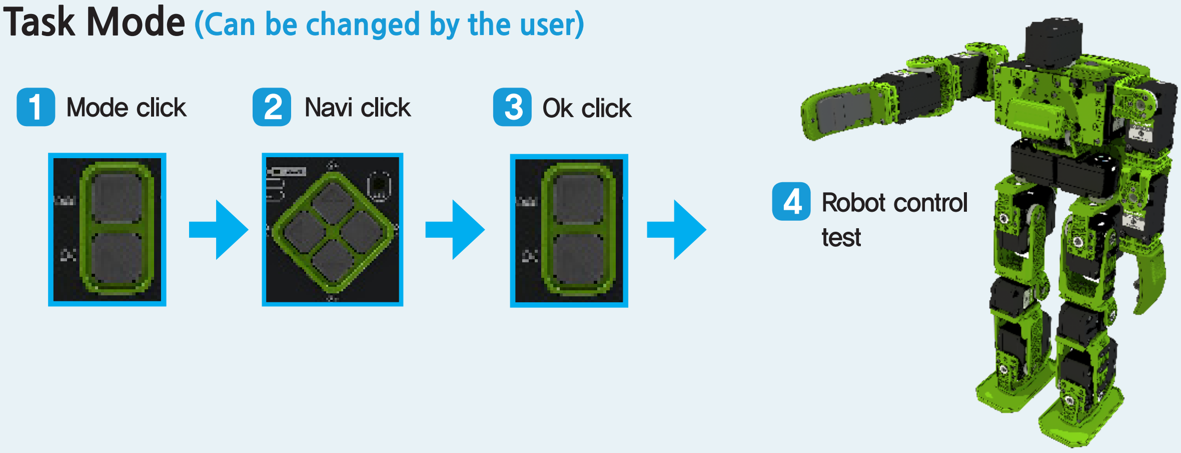
Task Mode (Can be changed by the user)

1 Mode click

2 Navi click

3 Ok click

4 Robot control test



DRC Functions

3 Operation & Functions

	Name	Description	Operation & Functions
Main Button	◩ Mode	Mode Change	Start Task For standard Task, Mode→ Navi key → Ok (to select operating mode)
	⬕ OK	Confirm button	
Navi Key (Firmware)	⬖ (L)	Battery level check	Battery level check → Shown by 3 left LED, low 1개 LED, medium 2 LED, high 3 LED
	⬘ (Up)	Test	Motor & sensor test using the controller Method : (Up) → Button → Test according to motor response from sensor Test : Motor/Light/Sound/Distance/Accel/Gyro
	⬗ (R)	Wired/Wireless	Wired using Ear phone jack / Wireless using ZigBee
	⬙ (Down)	Motor ID Scan	Rescan connnected motor ID
Navi Key (Basic Task for Humanoid)	⬖ (L)	Check Mode	Mode → (L) → Ok : Check Mode : Individual motors, Arm/Leg module connection and assembly check.
	⬘ (Up)	Autonomous Mode	Mode → (Up) → Ok : Autonomous Mode : Robot operates by itself.
	⬗ (R)	Remote Control Mode	Mode → (R) → Ok : Remote Control Mode: Run predefined motion saved in the remote control.
	⬙ (Down)	Sound Dem Mode	Mode → (Down) → Ok : Sound Demo Mode : Run motions based on sound input.
LED_mode	Servo	HerkuleX running	Blinks when HerkuleX Manager is in operation
	Program	DR-SIM/Visual Logic running	Blinks when DR-SIM / Visual Logic is being used for editing. LED on when downloading data or firmware
	EXEC	Mode chage/Task	On while the task is running when Task mode is entered using the mode button.
	TX	Data Transmit	Blinks when transmitting data, User spae when task in operation.
	RX	Data Receive	Blinks when receiving data, user spae when task in operation.
	Spare	User Defined	
	LED Blink	Error	3 right side LEDs will blink when in error
LED_Power	Power	Power level	Blinks when battery level is below 20%
Sensor	CdS	Light Sensor	Light sensor

DRC Basic Test

DRC is capable of running basic tests through the test mode even when the robot is not assembled. Proceed with the mo–tor and sensor test by turning on the power and pressing the (up) button. Sensor test is performed by checking the motor response from the motors ID1 and 2 attached to left and right. Tests can be performed for motor, light sensor, sound sensor, distance sensor, and gyro sensor. Testing methods are as follows.

Light sensor and sound sensor tests are done by menu and OK as they are built in to the controller
For PSD, Acc/Gyro, pressing the (down) button will check to see if only one of the sensor is connected. Test will proceed if only one sensor is connected and stop if more than one sensor is connected.

1 Motor : Connect motors to the left and right(No 1 on left, No 2 on right)
(Up) → (L) : left motor will move, (R) right motor will move, (Up) both motors will move .

2 Lighte Sensor : (Up) → (Menu) : Light sensor in operation, Both motors will move when Cds blocked

3 Sound Sensor : (Up) → (OK) : Sound Sensor in operation,
Clap from left, left motor will move,
Clap from right, right motor will move

Functions below will operate when (Down) button is pressed after connecting the sensor. Only the sensor to be tested should be connected as testing will not work if more than one sensor is connected.

4 PSD Digital : (Up) → Connect digital distance sensor→ (Down) PSD in operation
When object moves within 10cm → Both motors will move
When object moves beyond 10cm (cliff detection) → Both motors will stop
→ Cliff detection

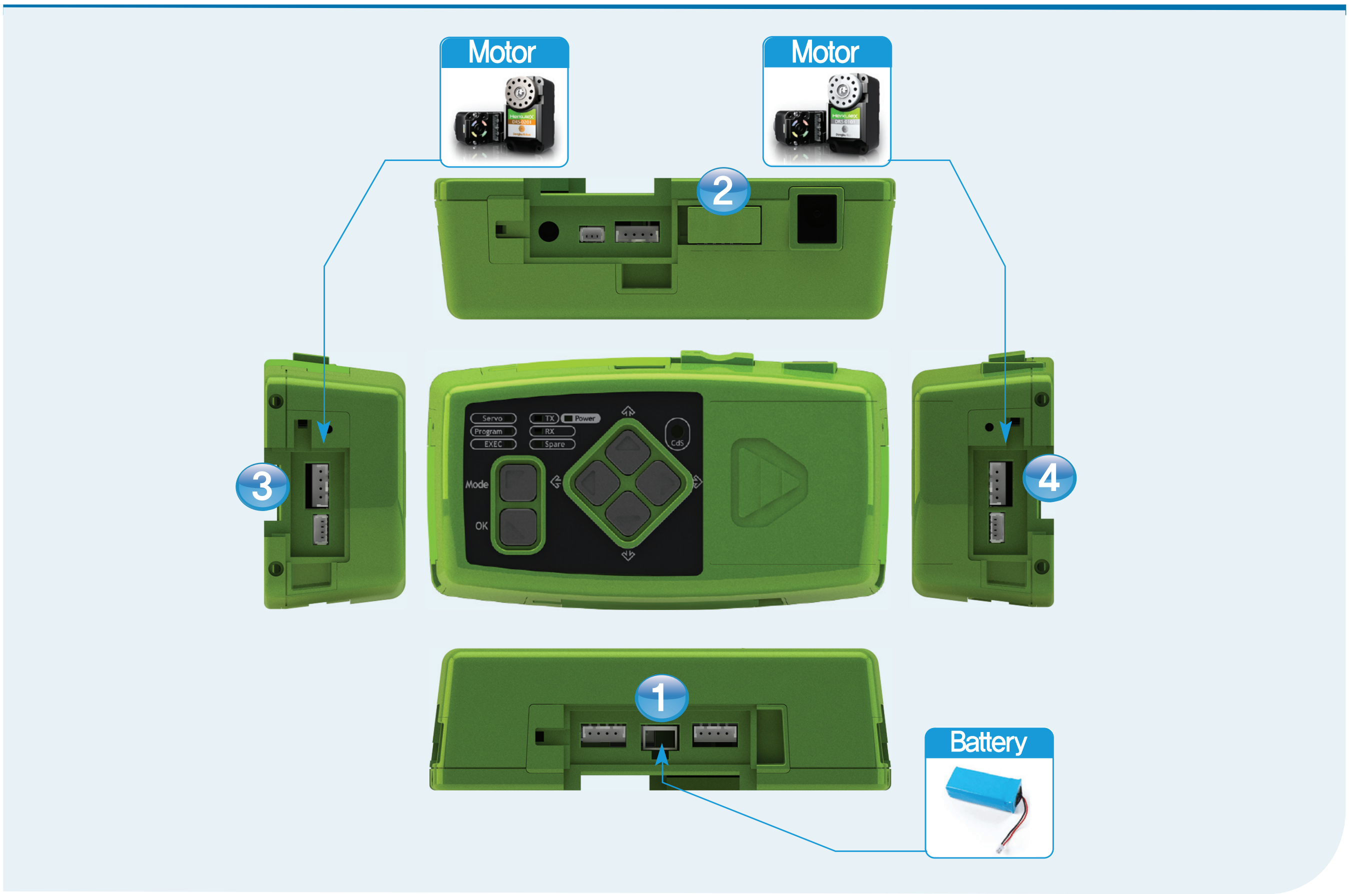
5 PSD Analog : (Up) → Connect analog PSD sensor → (Down) PSD in operation
Both motors will turn in same direction. Farther the object, faster the motor movement
Movement will slow when object comes closer → When object is <10 cm, motors will move in opposite direction.
→ Collision avoidance after wall/object detection

6 Acc : (Up) → Connect Acc/Gyro → (Down) Acc in operation
Motors stopped when the controller angle is same as when the robot is standing straight.
Motor speed will vary depending on the angle. The greater the angle faster the movement.

7 Gyro : (Up) → Connect Acc/Gyro → (Down) Acc in operation → (Down) Gyro in operation
No motore movement when the controller is not moving.
Motor moves at approximately the same speed as the revolving controller.

Follow the detailed test instructions below.

DRC Basic Test : Servo Motor



1 Connect the battery

2 Turn on the power

3 Connect left motor : Make sure to connect Motor ID 1 .(Other motors will not operate.)

4 Connect right motor : Make sure to connect Motor ID 2 .(Other motors will not operate.)

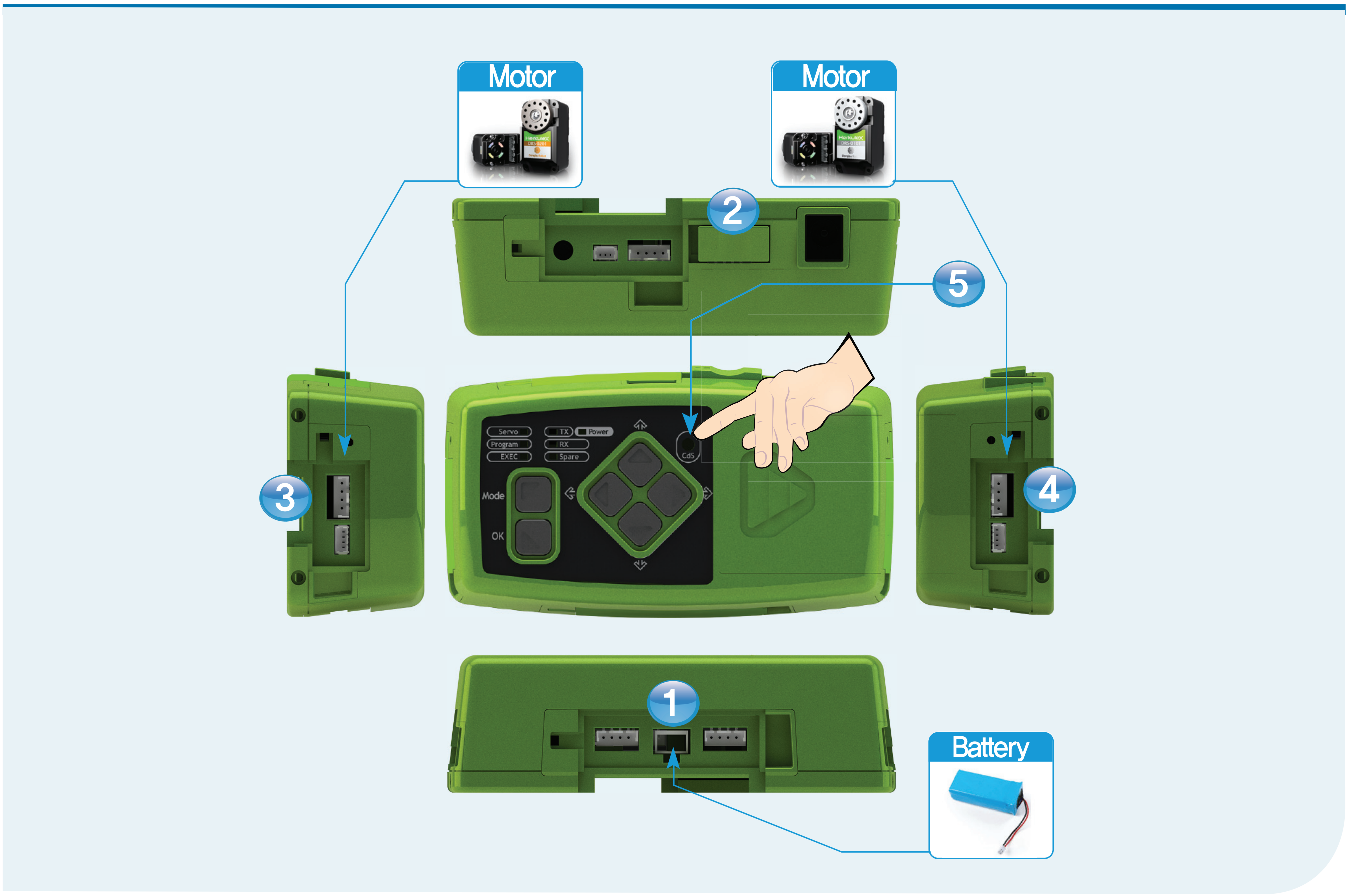
Place the motor outward to test. Simulates wheels turning.

→ Test Process

- Turn on power, Press (Up) button to enter Test Mode
- Press Navi Key (L) button. → Left motor will turn.
- Press Navi Key 의 (R) button. → Right motor will turn.
- Press Navi Key (Up) button. → Both motors will turn in forward direction.

Motors are operating without error if they worked according to the directions above, Results of all following tests will be shown by how the two motors behave. Do not disconnect the motors and continue on with sensor tests.

DRC Basic Tes : Light Sensor



1 Connect Battery

2 Turn on the power

3 Connect left motor : Make sue to connect Motor ID 1. (Other motors will not operate.)

4 Connect right motor : Make sure to connect Motor ID 2 .(Other motors will not work.)

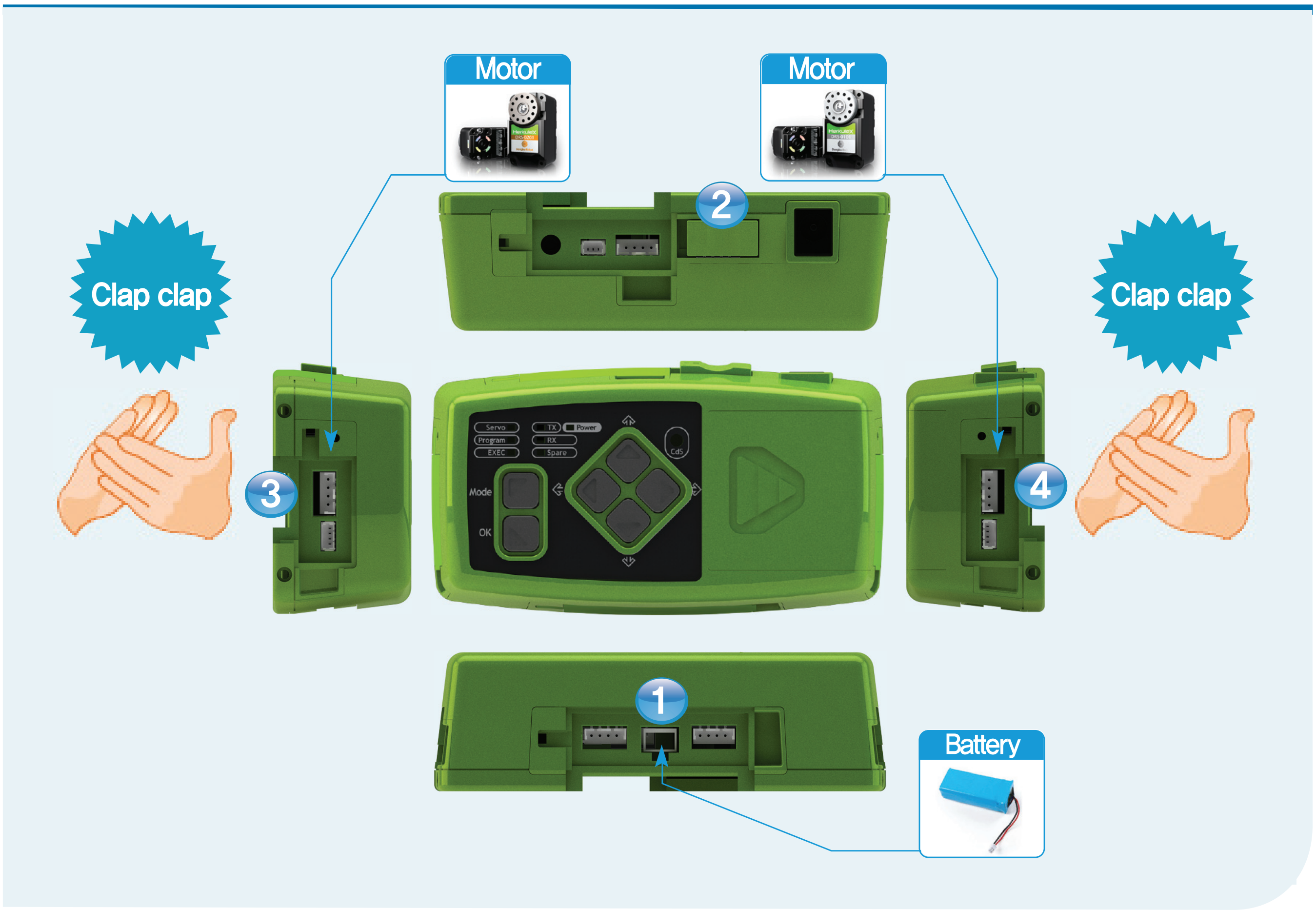
This test simulates robot arms grabbing the air when the light disappears.

→ Test Process

- Turn on power, Press (Up) button to enter Test Mode
- Cover the Cds window with hand. → both motors will turn at the same time.

Light sensor is operating without error if the motors turned accordingly.End light sensor test.

DRC Basic Test : Sound Sensor



1 Connect Battery

2 Turn on the power

3 Connect left motor : Make sure to connect Motor ID 1 .(Other motors will not operate)

4 Connect right motor : Make sure to connect Motor ID 2 .(Other motors will not operate)

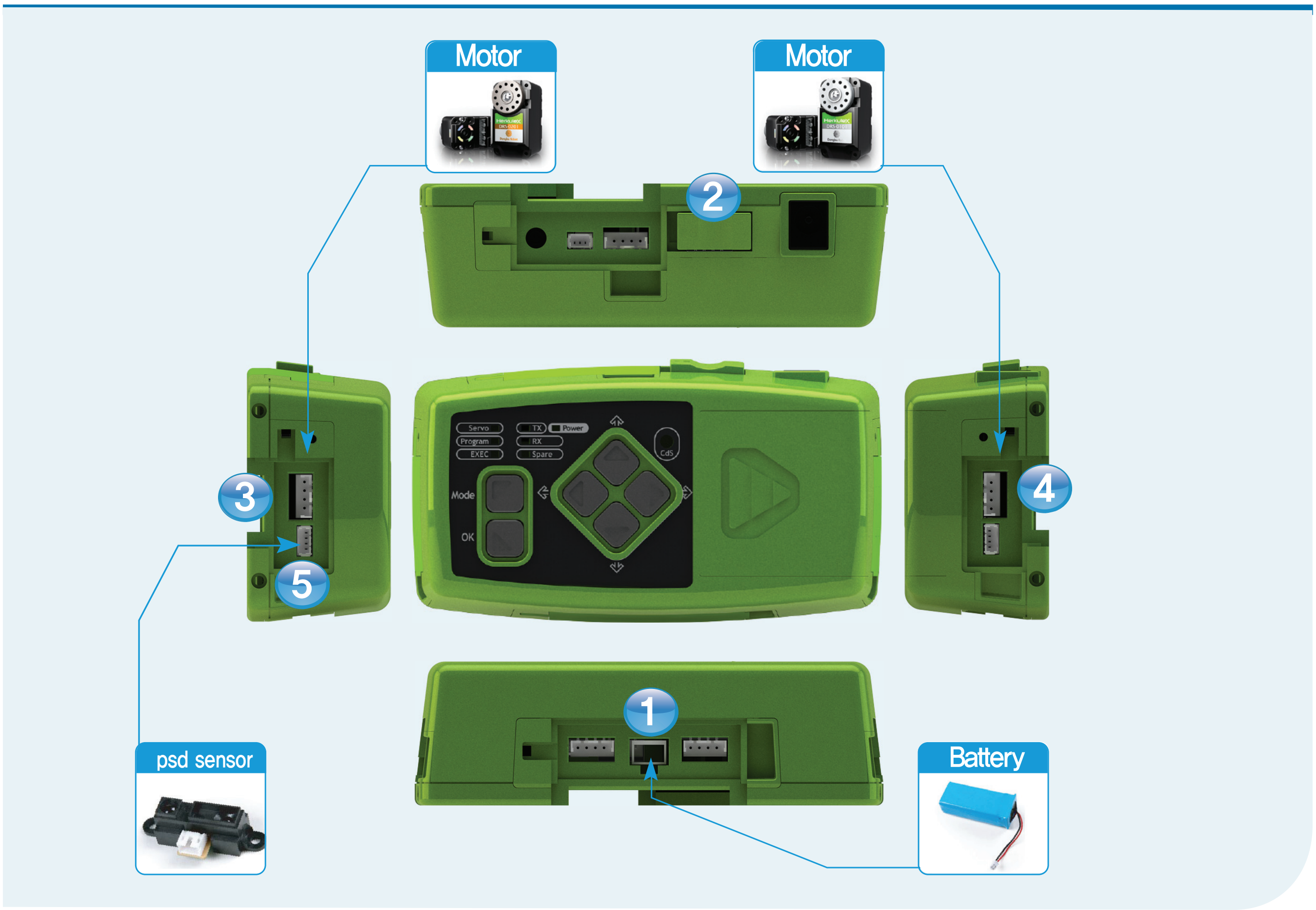
Motor near the direction of the clapping sound will turn.

→ Test Process

- Turn on power, Press (Up) button to enter Test Mode .
- Press (OK) button. → Sound Sensor in operation.
- Clap from left side. → Left motor will turn.
- Clap from right side. → Right motr will turn.

Sound sensor is operating without error if the motors turned accordingly. End sound sensor test.

DRC Basic Test : PSD Digital Distance Sensor



1 **Connect Battery**

2 **Turn on the power**

3 **Connect left motor :** Make sure to connect Motor ID 1 .(Other motors will not operate)

4 **Connect right motor :** Make sure to connect Motor ID 2 .(Other motors will not operate)

5 **Connect PSD Digital Sensor**

PSD Digital sensor uses certain distance as a base of measure and checks to see how far or near it is. It is normally used to check the depth of the ground to detect steep drop (cliff) and stop the robot.

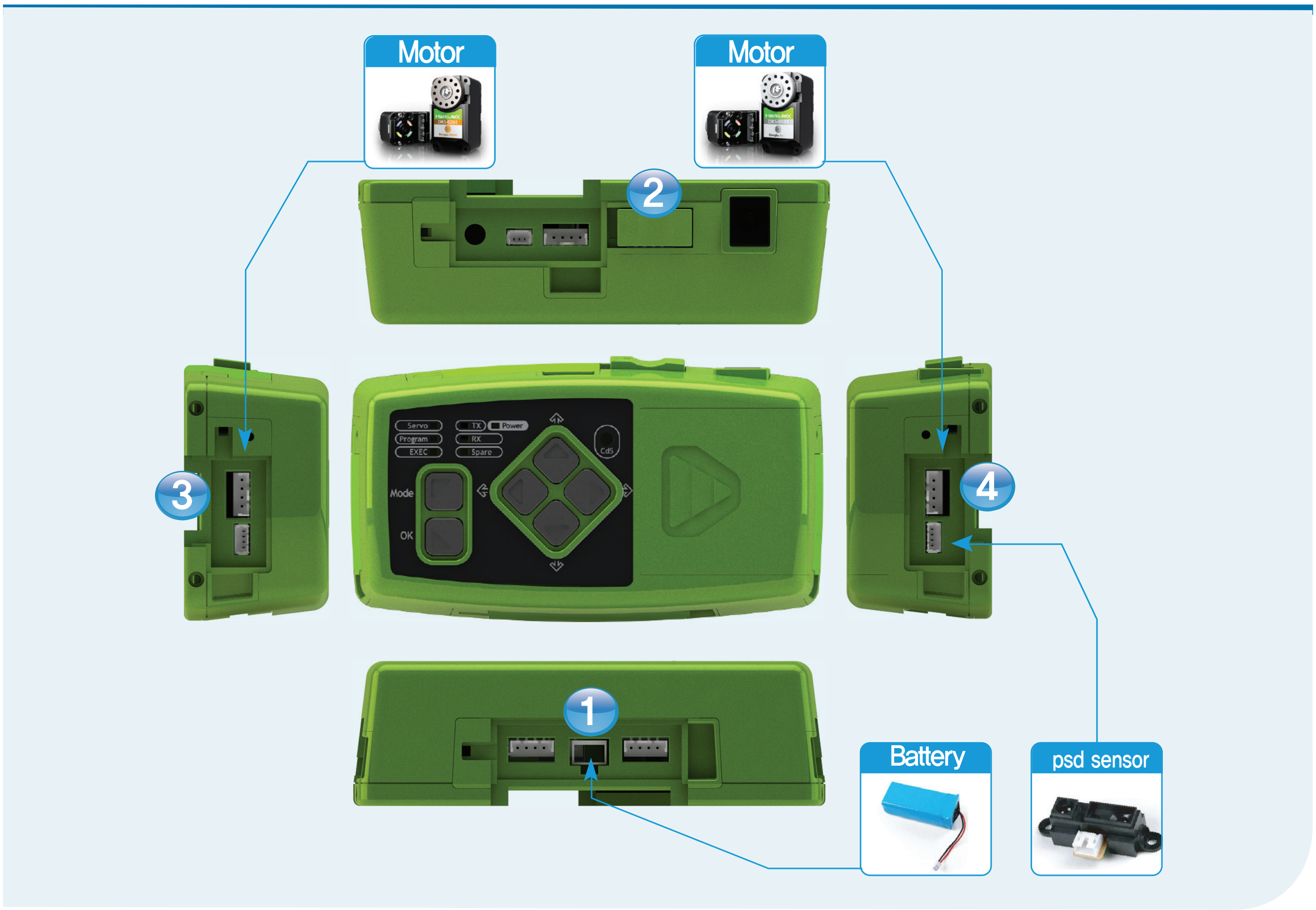
→ Test Process

- Turn on power, Press (Up) button to enter Test Mode.
- Connect the PSD line and press (Down) button. → PSD Digital Sensor in operation.
- Move your hand or an object within 10cm from the senson. → Both motors will turn.
- Move your had or an object away from the sensor to the distance > 10cm . → Both motors will stop.

* PSD Digital has only On/Off mode with certain distance as a base of measure.

PSD Digital Sensor is operating without error if the motors turned accordingly. End PSD Digital Sensor test.

DRC Basic Test : PSD Analog Distance Sensor



1 Connect Battery

2 Turn on the power

3 Connect left motor : Make sure to connect Motor ID 1 .(Other motors will not operate)

4 Connect right motor : Make sure to connect Motor ID 2 .(Other motors will not operate)

5 Connect PSD Analog Sensor

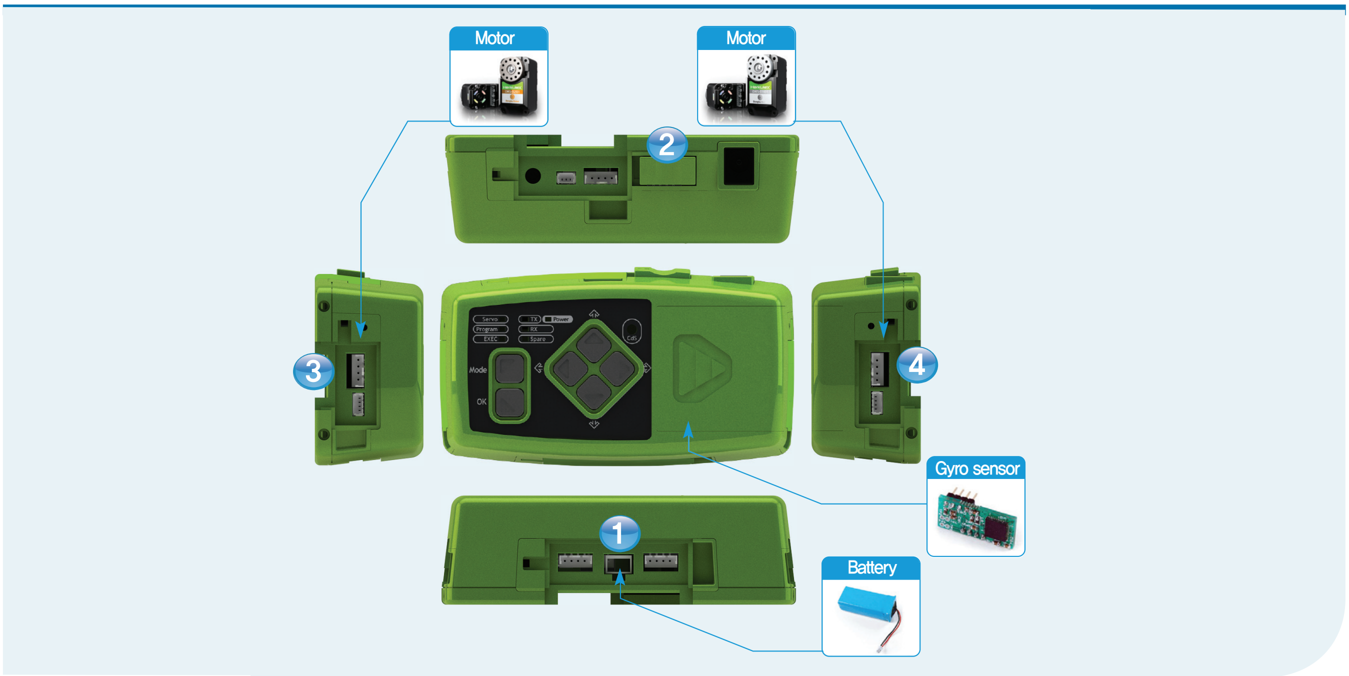
PSD Analog Sensor is able to measure the distance in realtime and control the motor speed according to the distance from an object. Normally used to avoid obstacles by slowing down and chaning direction.

→ Test Process

- Turn on power, press (Up) button to enter Test Mode. Connect PSD line and press (Down)button.
  → PSD Digital Sensor in operation. → Both motors will turn in same direction.
- Place a hand or an object near the sensor and mover away. → Farther the object faster the motor movement.
  → Nearer the object the slower the motor movement.
- When the object is less than 5cm away from the sensor, motors will turn in opposite direction.

PSD Analog Sensor is operating without error if the motors turned accordingly. End PSD Analog Sensor .

DRC Basic Test : Acc/Gyro Sensor



1 **Connect Battery**

2 **Turon on power**

3 **Connect left motor :** Make sure to connect Motor ID 1 .(Other motors will not operate)

4 **Connect right motor :** Make sure to connect Motor ID 2 .(Other motors will not operate)

Place the motors outward to simulate wheels turning.

5 **Connect Acc/Gyro Sensor :** Open the controller cover and connect the Acc/Gyro Sensor.

→ Test Process : ACC

- Turn power on and press (Up) button to enter Test Mode. Connect Acc/Gyro and press (Down) button.
  → Acc sensor is in operation.
- Motors stopped when the controller angle is same as when it is attached to the robot standing up straight.
- Tilt the controller slowly. → Speed of the motor will vary with the angle of the tilt. Greater the angle the faster the motr will turn.

Acc sensor is operating without error if the motors turned accordingly.

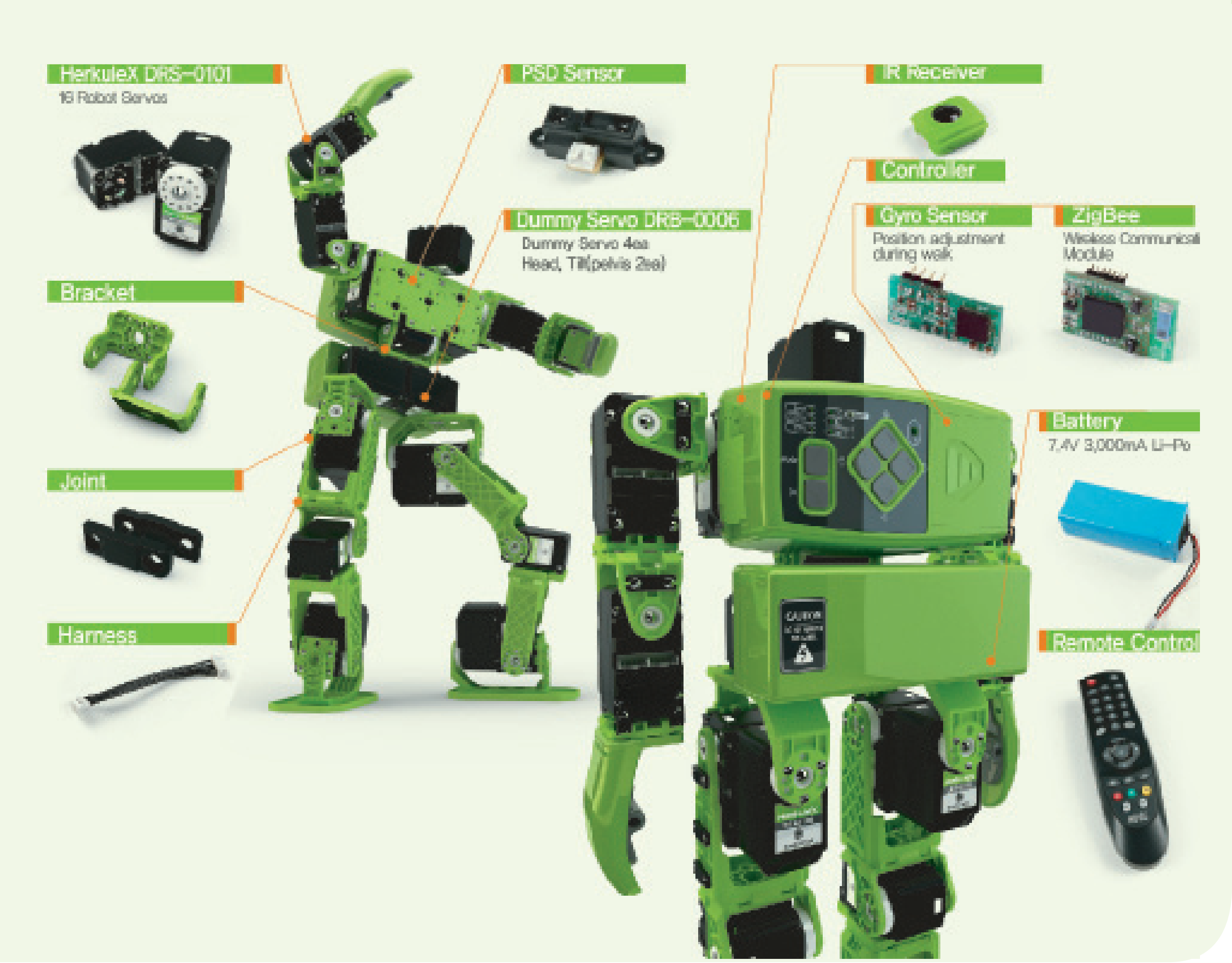
→ Test Process : Gyro

- Press (Down) button one more time from the ACC test mode. → Gyro sensor is in operation.
- Motors stopped when the controller is not moving.
- Move the controller. → Motors will turn at approximately similar speed to the moving controller.

Acc/Gyro sensor is operating without error if the motors tunred accordingly. End Acc/Gyro sensor test. Both the Acc and Gyro sensor is on a single chip board.

Material

Hovis Lite Componets Diagram



1 Servo Motr : HerculeX DRS-0101

2 Bracket : Acts as connecting joint between servo motors

3 Joint

4 Harness : Cable

5 PSD Sensor : Measures Distance

6 IR Receiver : Remote Control Receiver

7 Controller DRC

8 Gyro Sensor : Adjusts position while walking

9 ZigBee : Communications Module

10 Battery

11 Remote Control

Hovis Lite Parts List

Quantity Per Item.

Main	Controller	1ea	□
	7.4V Li-po Battery	1ea	□
	AC adoptor	1ea	□
	HerkuleX DRS-0101	16ea	□
Bracket	Front / Back	4ea	□
	Hand	4ea	□
	Foot	4ea	□
	Universal Plate	40ea	□
	Ankle Plate	8ea	□
	Dummy Servot	4ea	□
	U1-type Bracket	6ea	□
	U2-type Bracket	8ea	□
	U3-type Bracket	6ea	□
	U4-type Bracket	8ea	□
Accessory	Serial Cable (DSUB 9Pin/3p Audio Jack)	1ea	□
	USB to Serial Gender	1ea	□
	Wheel (White, Ø60)	4ea	□
	Horn(Plastic)	5ea	□

Joint	L-type Joint (Single Nut)	70ea	□
	L-type Joint (Double Nut)	10ea	□
	L-type Joint (Hole Only)	40ea	□
	I-type Joint (10.9mm,Hole only)	10ea	□
	I-type Joint (12.5mm,Double Nut)	60ea	□
	I-type Joint (16.0mm,Hole only)	20ea	□
	V-type Joint (12.0mm,Single Nut)	20ea	□
	V-type Joint (12.0mm,Double Nut)	20ea	□
	Bushing Set	30ea	□
Bolt & Nut & Harness	Nut (M2)	40ea	□
	Nut (M3)	10ea	□
	Bolt (PH/T 2.0X13)	25ea	□
	Bolt (PH/M 2.0X4)	50ea	□
	Bolt (PH/M 2.0X5)	20ea	□
	Bolt (PH/M 2.0X6)	150ea	□
	Bolt (PH/M 2.0X8)	20ea	□
	Bolt (PH/M 3.0X6)	10ea	□
	Bolt (PH/M 3.0X8)	20ea	□
	Bolt (PH/T 2.0X4)	40ea	□
	Bolt (PH/T 2.0X5)	60ea	□
	Harness (75mm)	4ea	□
	Harness (100mm)	4ea	□
	Harness (200mm)	6ea	□
	Harness (300mm)	2ea	□
	Harness Clamp	20ea	□

Material

HOVIS

Hovis Lite Parts List

Main

DRC-005T 1ea
Controller

DRL-0728 1ea
7.4V Li-Po Battery
(3,000mA)

DRQ-0002/3/4/5
AC Adaptor 1ea
(US/EU/UK/AU)

DRS-0101 16ea
HerkuleX Smart Servo

Bolt & Nut & Harness

DRA-0002 4ea
Harness (75mm)

DRA-0003 4ea
Harness (100mm)

DRA-0006 6ea
Harness (200mm)

DRA-0007 2ea
Harness (300mm)

DRA-0051 40ea
Nut (M2)

DRA-0052 10ea
Nut (M3)

DRA-0054 25ea
Bolt (PH/T 2.0X13)

DRA-0056 50ea
Bolt (PH/M 2.0X4)

DRA-0057 20ea
Bolt (PH/M 2.0X5)

DRA-0058 150ea
Bolt (PH/M 2.0X6)

DRA-0059 20ea
Bolt (PH/M 2.0X8)

DRA-0061 10ea
Bolt (PH/M 3.0X6)

DRA-0062 20ea
Bolt (PH/M 3.0X8)

DRA-0063 40ea
Bolt (PH/T 2.0X4)

DRA-0064 60ea
Bolt (PH/T 2.0X5)

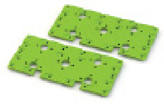
DRJ-0010 20ea
Harness Clamp

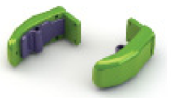
Material

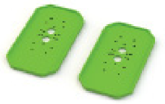
Hovis Lite Parts List

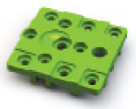
Photo Of The Parts.

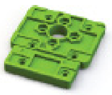
Bracket

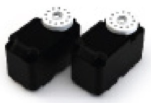

DRB-0001 4ea
Front / Back

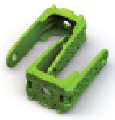

DRB-0002 4ea
Hand

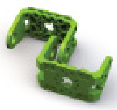

DRB-0003 4ea
Foot

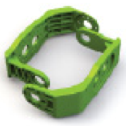

DRB-0004 40ea
Universal Plate

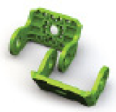

DRB-0005 8ea
Ankle Plate


DRB-0006 4ea
Dummy Servo

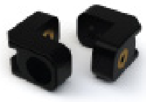

DRB-0008 6ea
U1-type Bracket

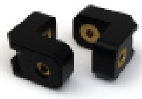

DRB-0009 8ea
U2-type Bracket

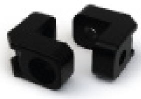

DRB-0010 6ea
U3-type Bracket

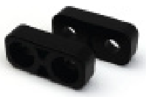

DRB-0011 8ea
U4-type Bracket

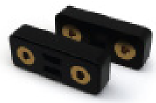
Joint

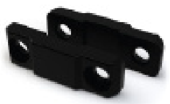

DRJ-0001 70ea
L-type Joint (Single Nut)

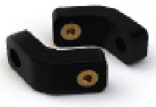

DRJ-0002 10ea
L-type Joint (Double Nut)

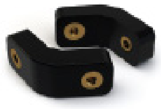

DRJ-0003 40ea
L-type Joint (Hole only)

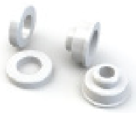

DRJ-0004 10ea
I-type Joint
(10.9mm, Hole only)


DRJ-0006 60ea
I-type Joint
(12.5mm, Double Nut)

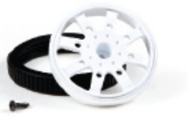

DRJ-0007 20ea
I-type Joint
(16.0mm, Hole only)

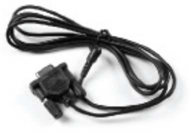

DRJ-0008 20ea
V-type Joint
(12.0mm, Single Nut)

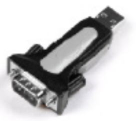

DRJ-0009 20ea
V-type Joint
(12.0mm, Double Nut)

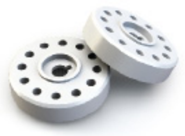

DRJ-0011 30ea
Bushing Set

Etc


DCW-0001 4ea
Wheel (White, Ø60)


DRI-0002 1ea
Serial Cable
(DSUB 9Pin-3P Audio Jack)


DRI-0003 1ea
USB to Serial Gender


DRH-1001 5ea
Horn (Plastic)

Humanoid Assembly Diagram

Servo motors in our humanoid robot are released with an ID number on each motor. When assembling the robot, make sure the servos are assembled at the right location by referring to the ID placement diagram.

Robot will not operate properly when servo motors are placed incorrectly. Motor ID numbers are based on 20 axis robot.

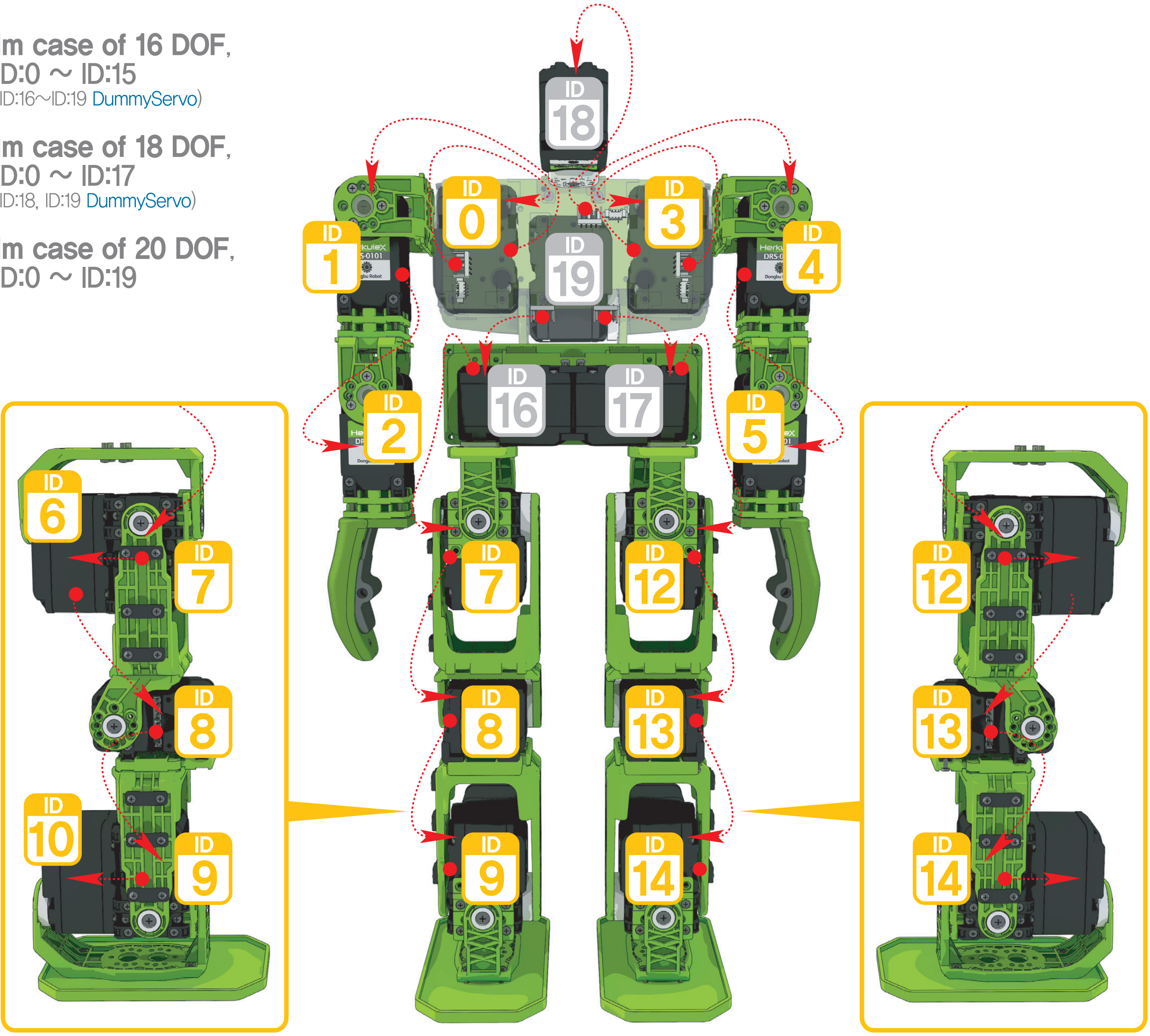
Motors numbered 16~19 have the last ID numbers as they dummy motors and replacements.

Wiring is the most difficult part in assembling the robot. Please read the manual carefully to fully grasp the wiring concept and try assembling one at a time. Refer to the humanoid robot assembly diagram for wiring details.

Im case of 16 DOF,
ID:0 ~ ID:15
(ID:16~ID:19 DummyServo)

Im case of 18 DOF,
ID:0 ~ ID:17
(ID:18, ID:19 DummyServo)

Im case of 20 DOF,
ID:0 ~ ID:19

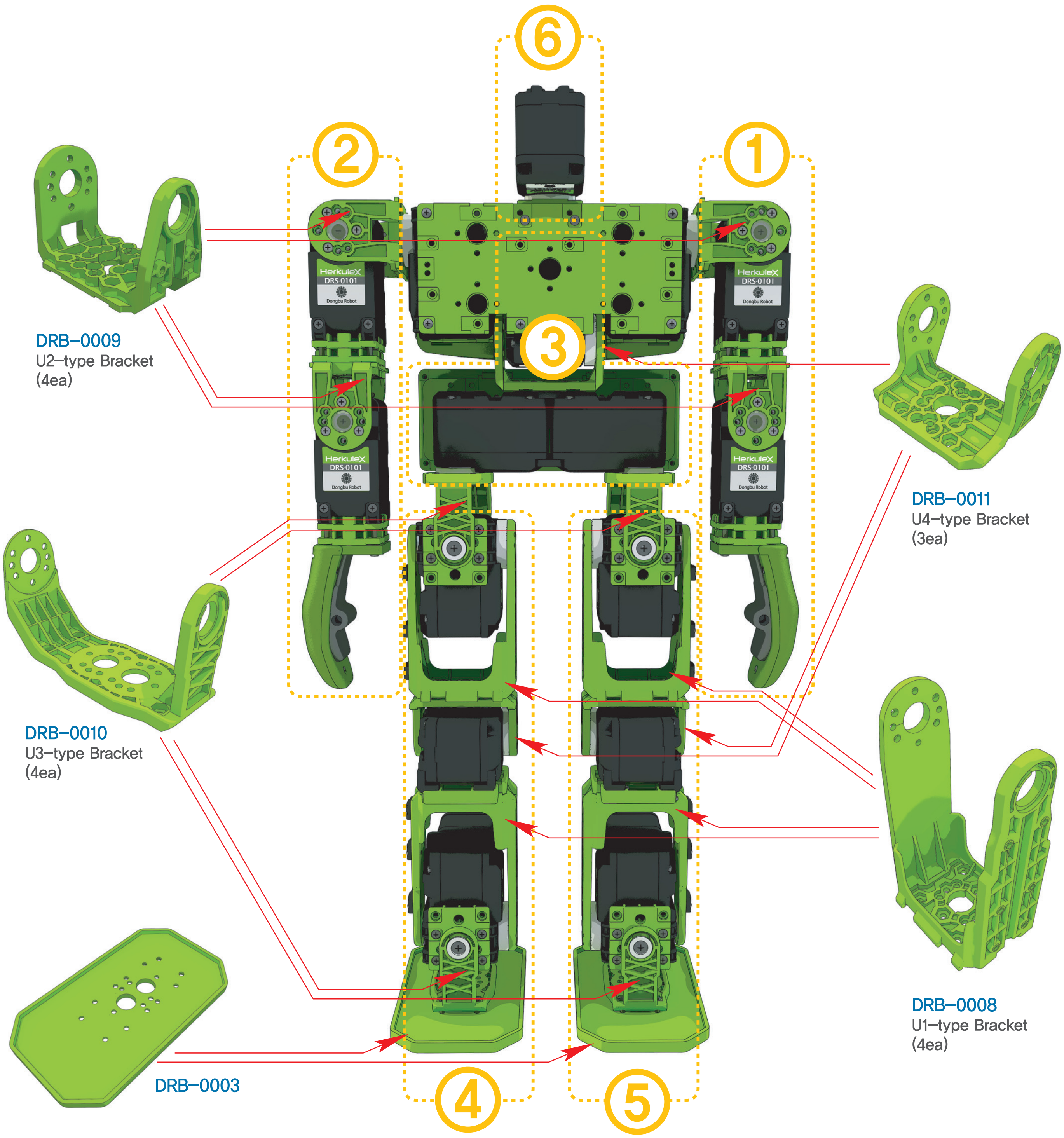


Humanoid Assembly Diagram

Brackets act as joints and as connection beween the servos.

Brackets make up the shoulder, waist, hips, knees, and feet. Study the bracket assembly diagram before assembly.

Assembly sequence is as follows ①Left Arm, ②Right Arm, ③Waist, ④Right Leg, ⑤Left Leg, ⑥Head, ⑦Controller, ⑧Battery



Humanoid DRC Function Instructions

Turning on the controller

Power	Blinks when battery level falls below 20%
Batter Level Check	Turn on the power and press (L) button to check the battery → Shown by left 3 LEDs low 1 LED, medium 2 LED, high 3 LED
Entering Task	Press mode btton to enter basic task Navi key → Ok button, select desired mode

Robot Operation

Assembly midpoint check mode	Mode → (Down) → OK : Check assembly at various midpoints. right arm, left arm, right leg, left leg, sensor
Motor check mode	Mode → (L) → OK : Motor check mode – motor torque released when checking, motor selected one at time. Selected motor LED blinks. – (Up) Motor ID ascending order, (Down) Motor ID descending order – Warning alarm sounds if motor ID does not exist.
Autonomous Mode	Mode → (Up) → OK : Autonomous mode : robot moves by itself – Clap, and the robot will move towards the direction of the clap for number or claps (Clapping sound during the movement will be ignored) – Robot will start basic movements if does not receive particular re– sponse in 5s – Basic movements : sit, stand,move forward,backward, change direction – Obstacle avoidance(PSD sensor required) – When fall, automatically stand up by oneself. (Gyro Sensor Needed)
Remote Control Mode	Mode → (R) → OK : Remote Control Mode Predefined movements in number keys (0~9)and in direction keys(up,down,L, R, stop)

Program Download

	Motion of LED	Declared LED
HerkuleX connection	Blinks when HerkuleX Manager is running	Servo
DR-SIM/ Visual Logic Connection	Blinks when DR-SIM / Visual Logic is being used to edit Lit when dowonloading data or firmware	Program
Apply Task to Robot	LED will stay lit when the task is running	EXEC
Data Transmit	Blinks when transmitting data, when task is running User Spare area used	TX
Data Receive	Blinks when receiving data, when task is running User Spare area used	RX
User Defined		Spare
Error	When error detected, all LEDs blink with alarm.	All LED blinks

Robot Motion

Check Mode	**Mode → (L) → OK Enter Check Mode** **When green RX LED comes on, press (L) or (R) button to select item to check.** **(L) : Motor check mode** – Selects each individual motor and checks connection status and assembly. – Red TX LED comes on when motor check mode entered. – When Green LED on selected motor comes on, motor turns to center position (512). Rest of the motors go into Torque Off state when LED goes off. – Press (Up) & (Down) to select the motor ID(0~15). Check the conntection status and location of the motors. First selected ID is 0(Right Shoulder). – Press (Up) button to increase ID by 1, (Down) button to decrease by 1. – Warning buzzer will sound if selected motor ID does not exist. **(R) : Midpoint Check Mode.** – Checks assembly state of arms, legs, and other parts by testing individual modules. – Spare LED comes on when Midpoint Check Mode is entered. – Motors in the selected module makes slow repeated movements to simulate straight and bent posture. – Press (Up) & (Down) button to select the arm and the leg. – Sequence: Left arm, Right arm, Left leg, right arm. Left arm is the first selected module. – If motor ID is missing from the selected module, buzzer will make same number of sounds as the numbr of missing motor IDs.
Autonomous Mode	**Mode → (Up) → OK, enter autonomous mode.** – Robot makes autonomous movements without user intervention. – Robot will select from the following movements in random; forward, front roll, left turn, right turn. – In forward movement, robot will select from 10/20/30 steps in random. In left/right turn, random selection from 12/24/36 steps. – Robot will pause for brief time after completing the randomly selected movement before starting next random movement. – Robot will be able to avoid obstacles if PSD sensor is installed in the ADC prot 1. – If robot detects an obstacle, it will randomly select one of the following movements; backward & left turn, left turn, back roll & left turn. – backward roll & left turn is only possible if the robot detects an obstacle after moving at least 10 steps forward. – If an obstacle cannot be avoided even after making umber of left turs, robot will try backeward & left turn. – If acceleration sensor is installed, robot will get back up after falling. – If robot detects a fall, it will stop current motion and switch to getting up mode.

Using Humanoid DRC Functions

Robot Motion

Remote Control Mode	**Mode –> (R) –> OK, enter remote control model** – Controls the robot by remote control. Remote control receiver must be installed for this mode to function. – Up : Forward – Down : Backward – L : Left turn – R : Right turn – OK : Stop – 1 : Roll forward – 2 : Roll backward – 3 : Push–up – 4 : Boxing – 5 : forward get up(Acceleration sensor must be installed, Only possible from supine position) – 6 : backward get up(Acceleration sensor must be installed, Only possible from prone position)
Sound Dem Mode	**Mode –> (Down) –> OK, enter sound demo mode.** – Robot reacts to the number and direction of the sound – Sound detection on, when controller TX, RX, Spare LED is on – During sound detection, – Single sound detected : Random motion from roll forward, roll backward, push–up, boxing. – Sound deteced twice : robot will lift the arm in the direction of the detected sound and wave. If the sound was detected once from the left and once from the right, robot will wave the left arm first and then the right arm. – Three sounds detected : Robot will turn to the direction of the sound and walk 10 steps forward. If the sound was from the left, robot will turn left and then walk forward. **※ Sound detection may not work 100% all the time due to background noise, echo from the wall, and other environmental factors. Robot will detect loud and short sounds like hand clapping more easily.**

PART 02

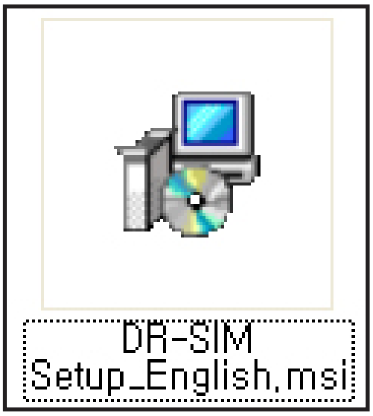
DR-Visual Logic Programming

DR–SIM & DR–Visual Logic

DR–SIM Introduction

DR–SIM, also called 'motion editor' is an easy to use robot motion editing software tool. In addition to motion creation, editing, and capturing actual robot motion, DR–SIM supports powerful simulation function that allows the user to simulate the motion prior to applying it to the robot. DR–SIM also incorporates timeline feature similar to the ones found in video editing software. Timeline allows the user to create motion based on time and to add multimedia effect to the motion by adding LED lighting effect and sound in the timeline.

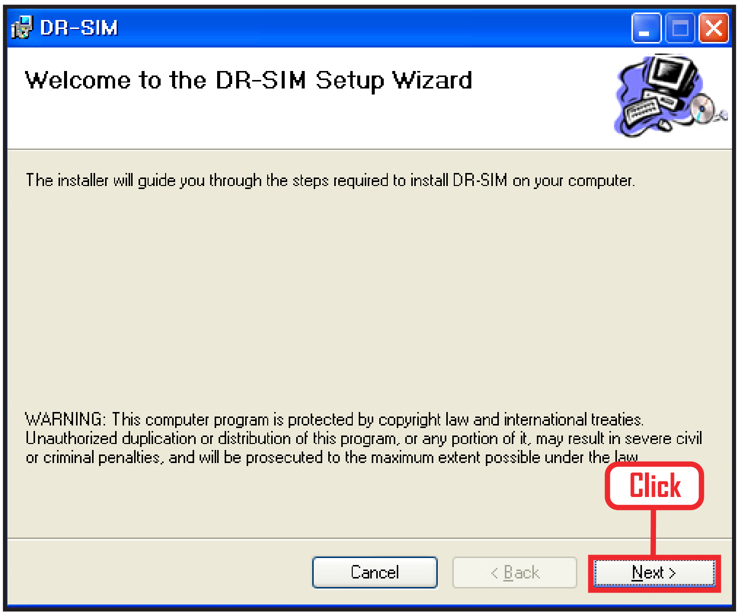
- **System requirement**
- **Minimum Intel Pentium 800 Mhz**
- **Windows XP, Windows Vista, Windows 7**
- **Minimum 256 MB RAM**
- **Hard Disk Space 300 MB required**
- **USB Port**
- **Macintosh(under development)**



DR-SIM Setup_English.msi

01 Installation File

Click on installation file.



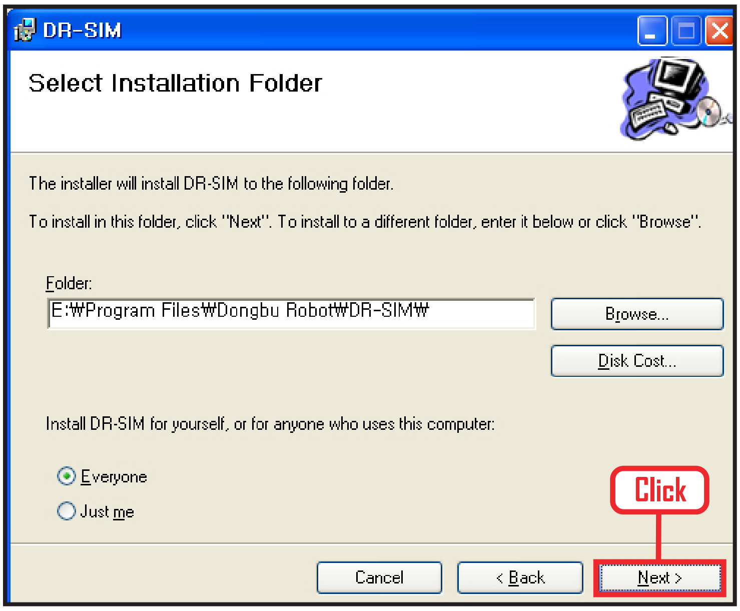
Welcome to the DR-SIM Setup Wizard

The installer will guide you through the steps required to install DR-SIM on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

02 Start installation wizard

Click **"Next"** button.



Select Installation Folder

The installer will install DR-SIM to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

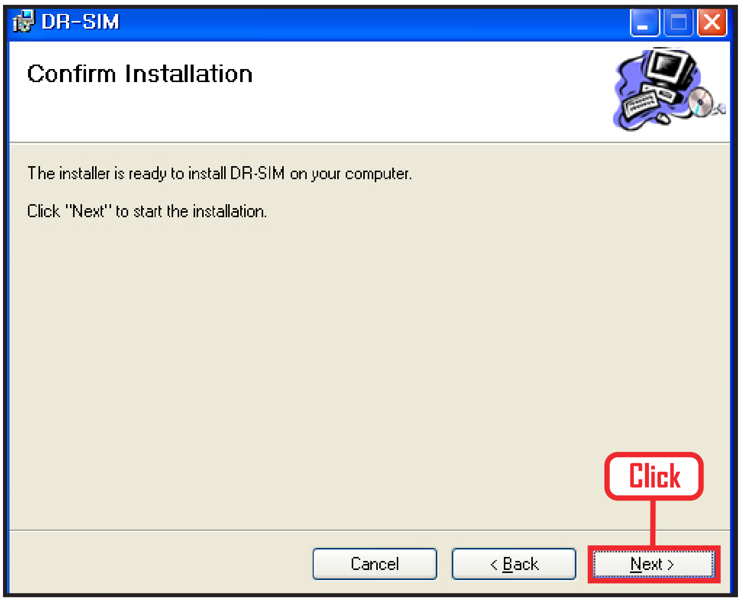
Folder:
E:₩Program Files₩Dongbu Robot₩DR-SIM₩

Install DR-SIM for yourself, or for anyone who uses this computer:

● Everyone
○ Just me

03 Select installation folder

Click "Next" button.



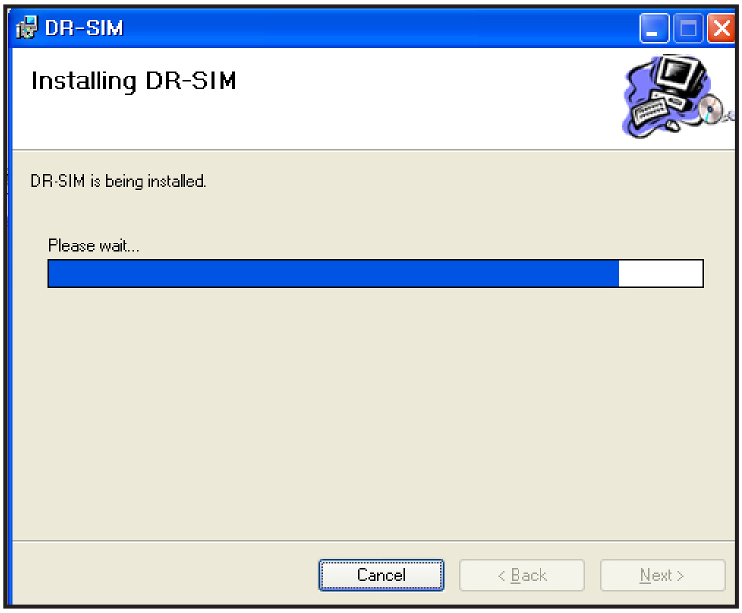
Confirm Installation

The installer is ready to install DR-SIM on your computer.

Click "Next" to start the installation.

04 Confirm installation

Click "Next" button.



Installing DR-SIM

DR-SIM is being installed.

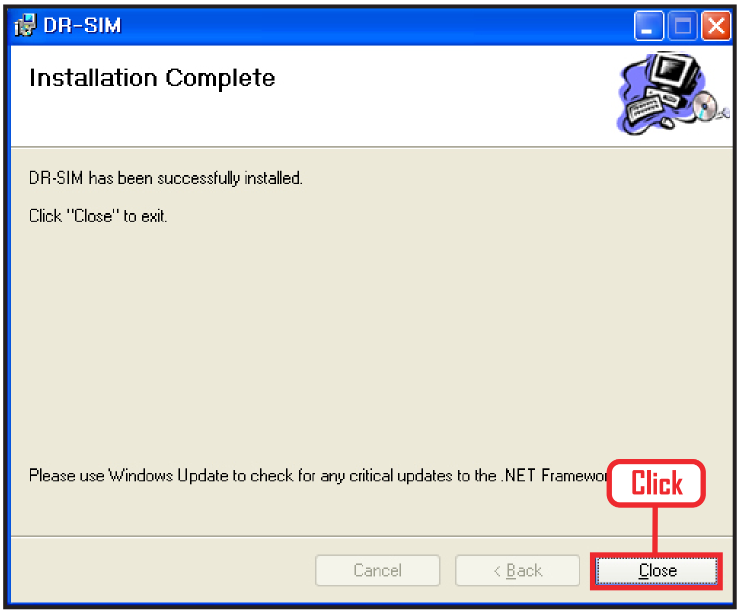
Please wait...

05 Start installation

Starting installation. Wait untill the installation bar ends.

06 Confirm installation

Click "Close" button
Software installation complete.



DR-SIM

Installation Complete

DR-SIM has been successfully installed.

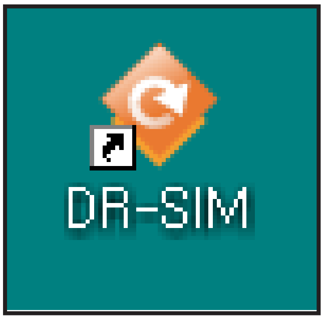
Click "Close" to exit.

Please use Windows Update to check for any critical updates to the .NET Framewor

07 Check executable file

Check for the executable file, desktop shortcut icon and from Windows Start 〉 All Programs 〉 Dongbu Robot 〉 DR–SIM.
Click on the executable file to run the program.

If the porgram did not install properly, install the Microsoft, Net Frame work 3.5 and try again.



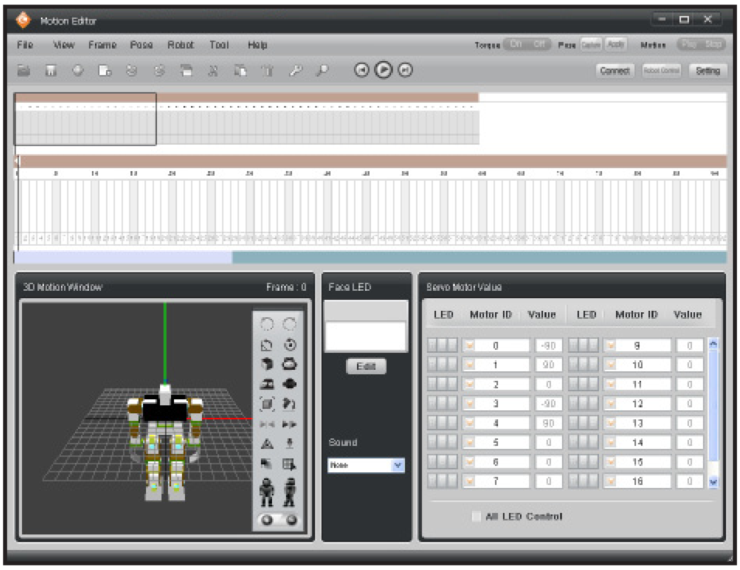
Hello DR-SIM

First example of creating motion. Use DR-SIM tor create simple motion and run the motion simulation.

Connect to the robot and download the created motion file and then check the motion being applied by the robot.

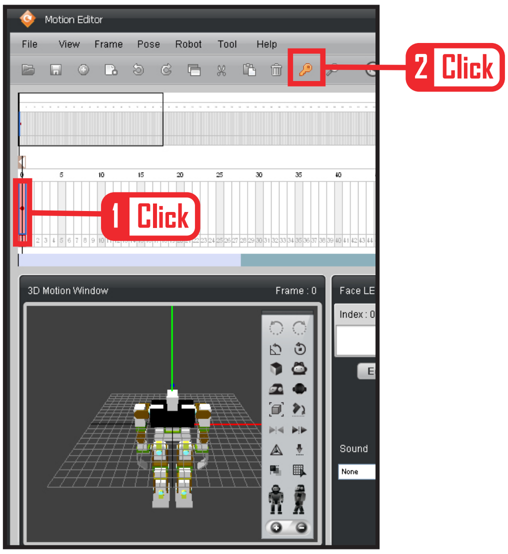
00 Run program

Click on DR-SIM icon and run the program.



01 Full Screen

DR-SIM Full Screen.
Timeline is in the middle and motion editior at botom. Motion is usually created or edited using the timeline and 3D motion window.

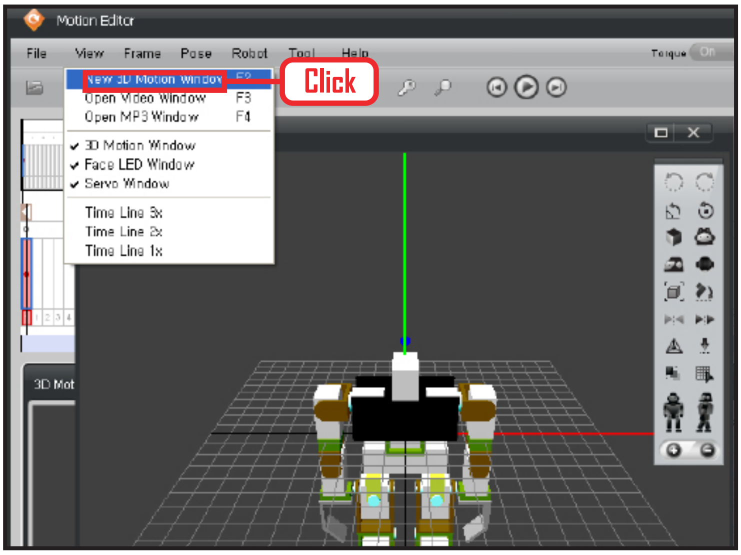


02 Basic Posture

Insert the basic posture in the robot motion starting point or in the first frame.

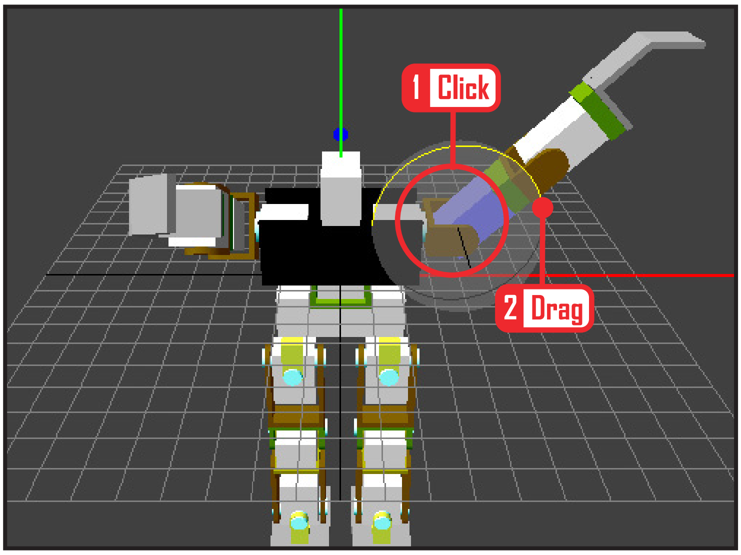
Place the posture in the 3D motion window as basic posture →Click first frame → Insert the key frame at the top. Click on (Key icon)
Basic posture has been inserted in the key frame.

If the posture in the 3D motion window is not a basic posture, select it as basic posture from the tool bar on the right.(Shortcut Alt + I)



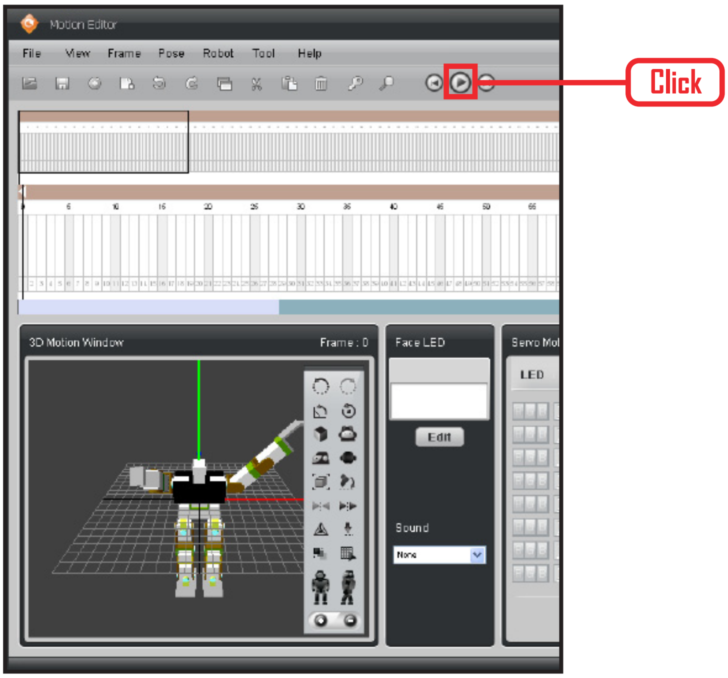
03 3D Window

To enlarge the 3D window, **Menu 〉 View 〉 Click on ‘New 3D Window’.**
Click and drag to enlarge the new 3D window.



04 Motion Edit

Click on the robot joint and thin yellow joint movement line will appear. Click and hold left mouse button on the line and drag.
Lift the left hand up left and the right hand up front.

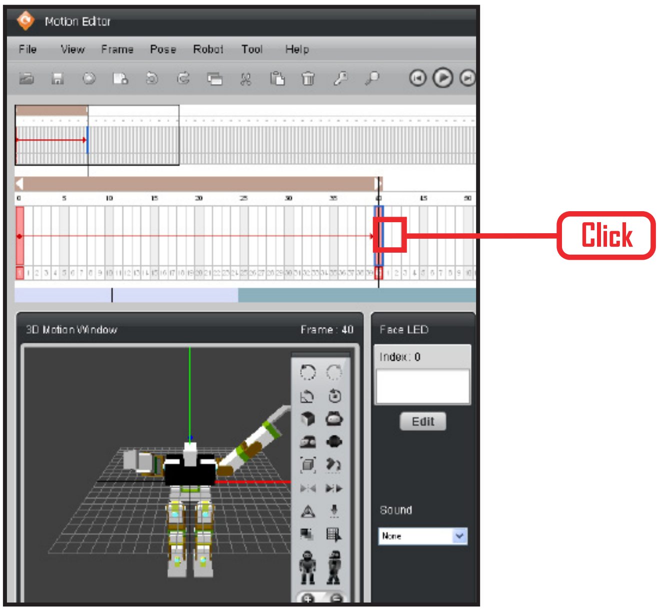


05 Insert Edited Motion

Insert the motion edited in 3D window into desired timeframe. After inserting the motion, click on the “ ⊙ “ at the top to view the simulation in the 3D window.

06 Inserted Frame Midpoint Check

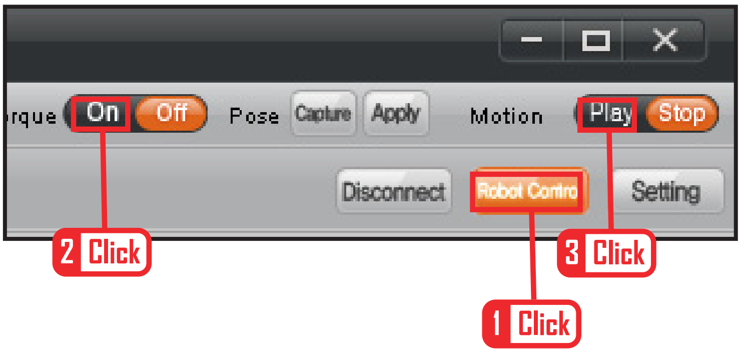
To view the motion between two frames, click on the timeframe section between the basic motion and the edited motion.



07 Connecting to Robot

Use the USB to Serial converter cable to connect the robot to the USB port of PC or notebook computer. **Click "Connect"** icon to make the connection. Check the Com port if the connection does not occur. **Click "Torque On"** button and try moving the robot arm or the leg by hand. Torque is on if the robot does not move. Click on **"Robot Play"** button and robot will move following the created motion,

This ends the first lesson on creating robot motion and play.

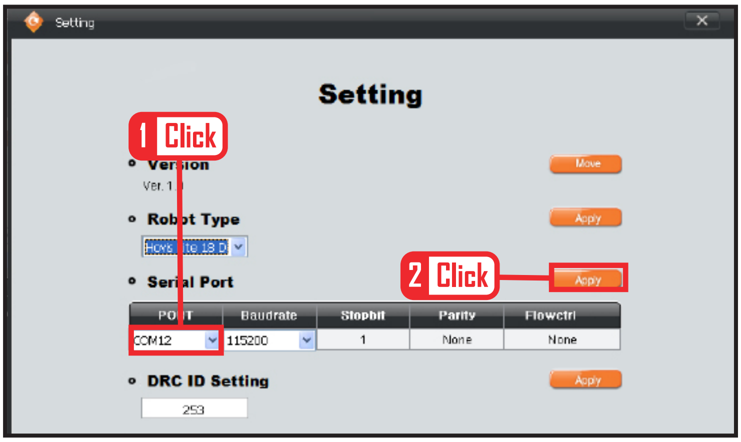


# Reference : COM Port Setting

If the connection to the robot does not occur, it is most likely due to wrong Com port settings. Right click on "My Computer", click on "Properties" to open "System Properties" window, click on "Hardware" tab to open the device manager. Click Com port to view the list of configurable Com ports. Select COM2 connected to the USB and save. Com port connected to the robot should now be open.

09 Reference : COM Port Setting

Select COM2 and save.
Robot and PC software should now connect.



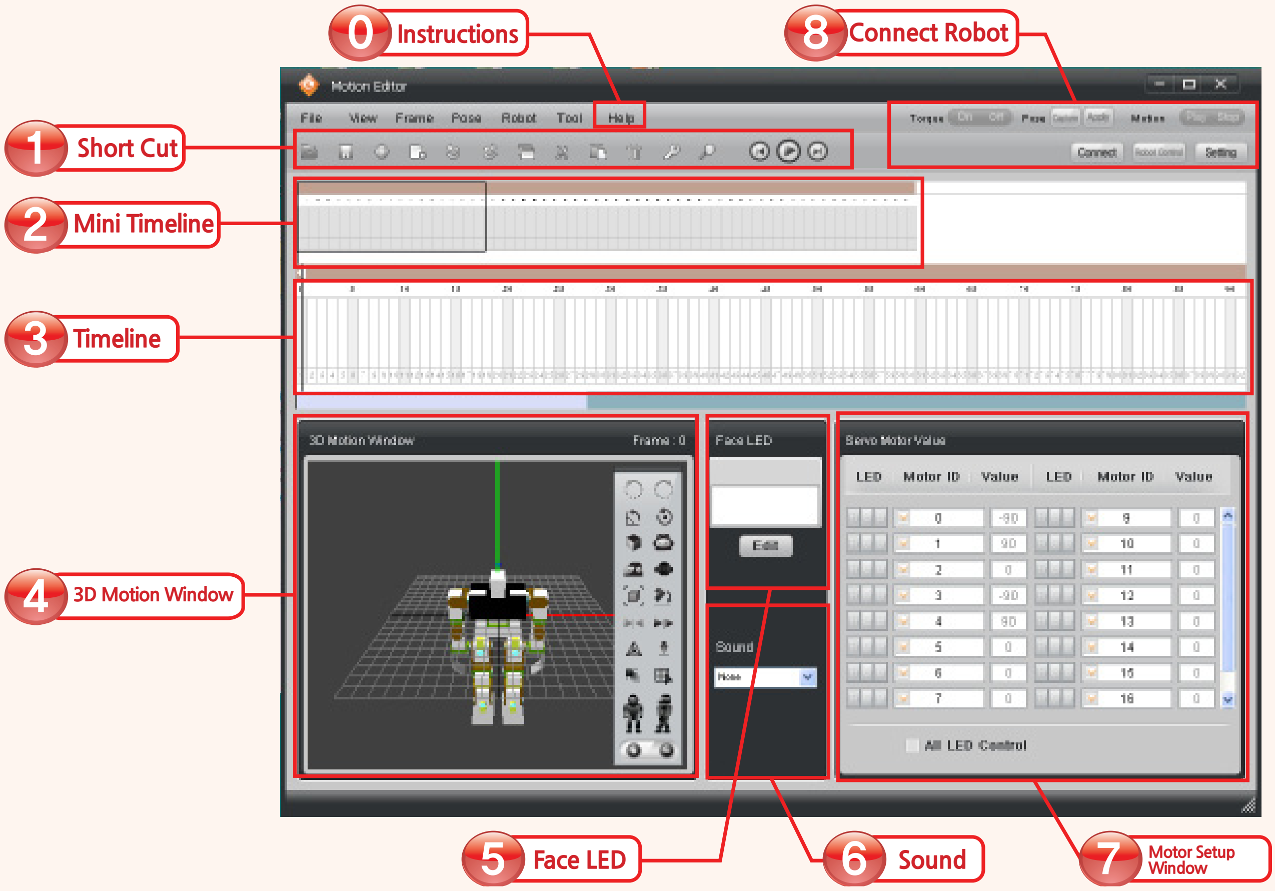
Setting

- Version
  Ver. 1.0
- Robot Type
  Hovis Lite 18 D
- Serial Port

PORT	Baudrate	Stopbit	Parity	Flowctrl
COM12	115200	1	None	None

- DRC ID Setting
  253

User Interface



Servo Motor Value

LED	Motor ID	Value	LED	Motor ID	Value
	0	-90		9	0
	1	90		10	0
	2	0		11	0
	3	-90		12	0
	4	90		13	0
	5	0		14	0
	6	0		15	0
	7	0		16	0

□ All LED Control

0 **Instructions :** DR-SIM detailed user instructions. Press F1 to view Help .

1 **Short Cut :** Collection of frequently used menus. Simulation Play, Insert Keyframe, and etc.

2 **Mini Timeline :** Shows the outline of whole timeframe.

3 **Timeline :** Created or edited motion can be placed by time.

4 **3D Motion Window :** Edit robot motion or view the motion simulation.

5 **Face LED :** Enables user to edit Face LED. Insert into timeframe after editing.

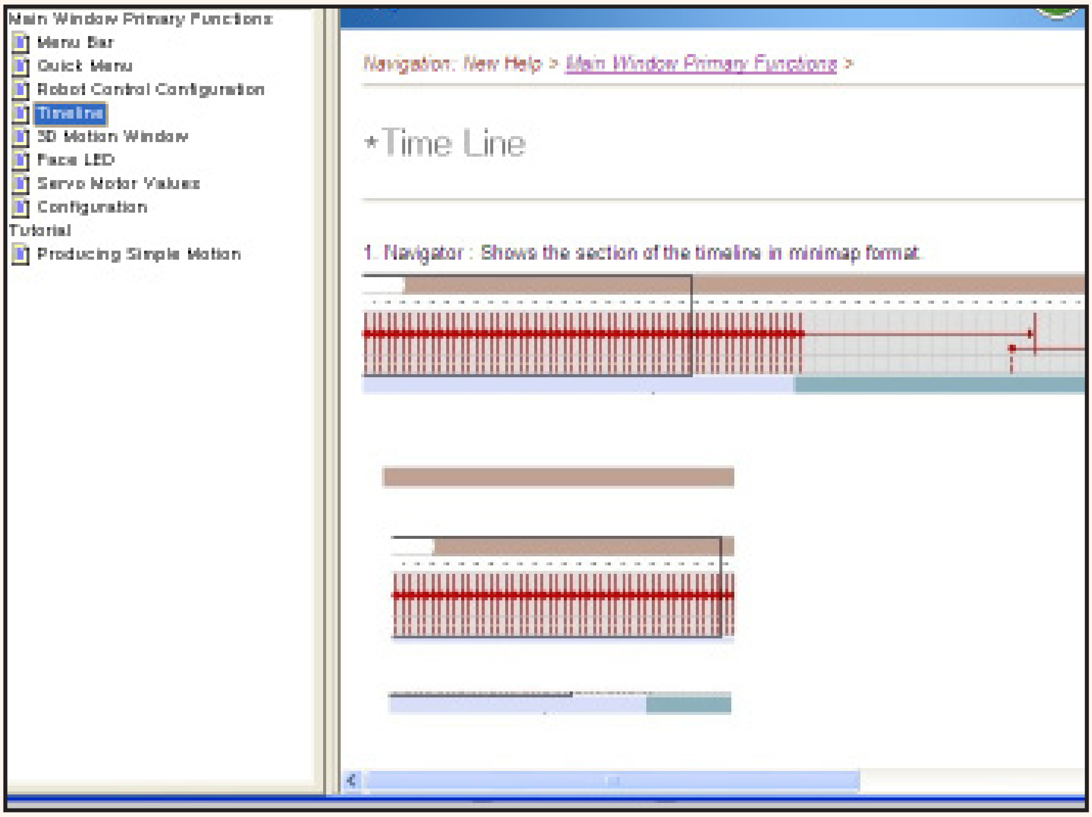
6 **Sound :** Select saved sound. Insert into timeframe after selcection.

7 **Motor Setup Window :** Configure values and LED settings by ID for all motors used in the robot.

8 **Connect Robot:** Shortcut for connecting to the robot. Used to download motion file to the robot or to capture actual robot motion.

Help

Click 'Help' on the menu bar to popup the help window. We recommend reading the Help files prior to using the DR-SIM program. **(Click 'Help' 〉 Click'Index' 〉 Click 'Timeline' on left menu → Window shown below will open up**



Navigation: New Help > Main Window Primary Functions >

Time Line

1. Navigator : Shows the section of the timeline in minimap format.

- **Com Setting :** Instruction on setting up the COM port.
- **Main Window Major Functions :** Instructions on how to use program functions.

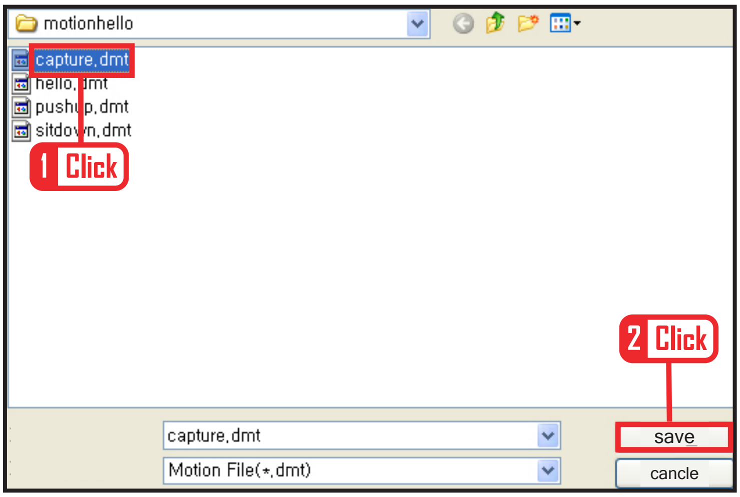
Menu Bar
Quick Menu
Robot Control Setting
Timeline
3D Motion Window
Face LED
Servo Motor Values
Environment Setting

- **Tutorial**

Creating Simple Motion : Explanation about sample motion creation.
Check our website for more motion samples.

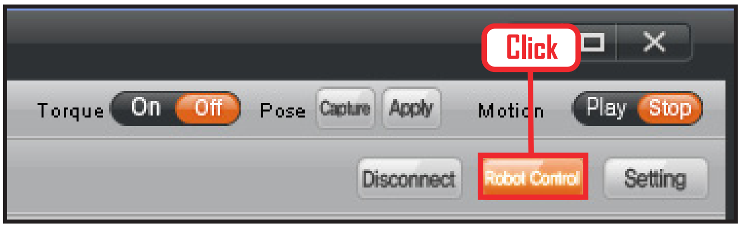
Download

Edited motions are saved as a file. Saved motions files can be batch downloaded to the DRC controller (Existing files in the DRC will be deleted). Downloaded files are ginve a number according to the order of download which then can be loaded and used by DR-Visual Logic(Task Editor).



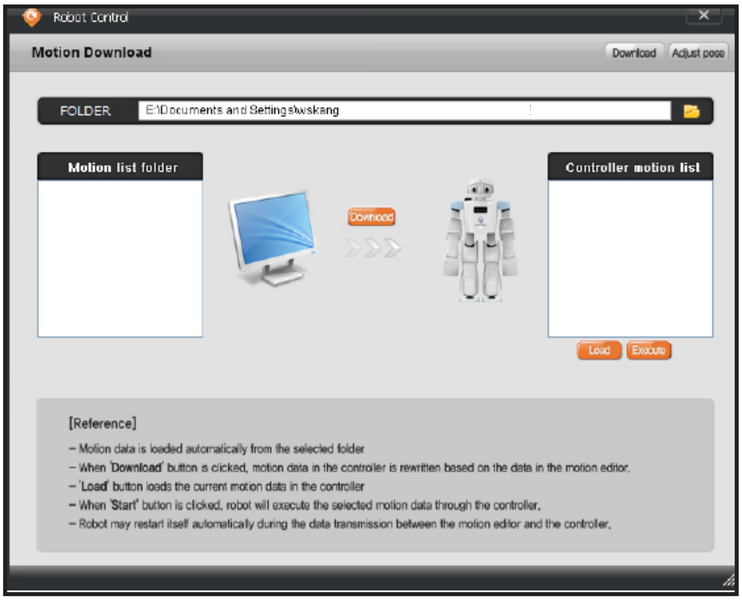
Save Robot Motion

To save the robot motion, **File 〉Save As 〉 insert file name and save.**



01 Robot Control

With the robot and DR-SIM connected, click on **"Robot Control"** icon.



[Reference]

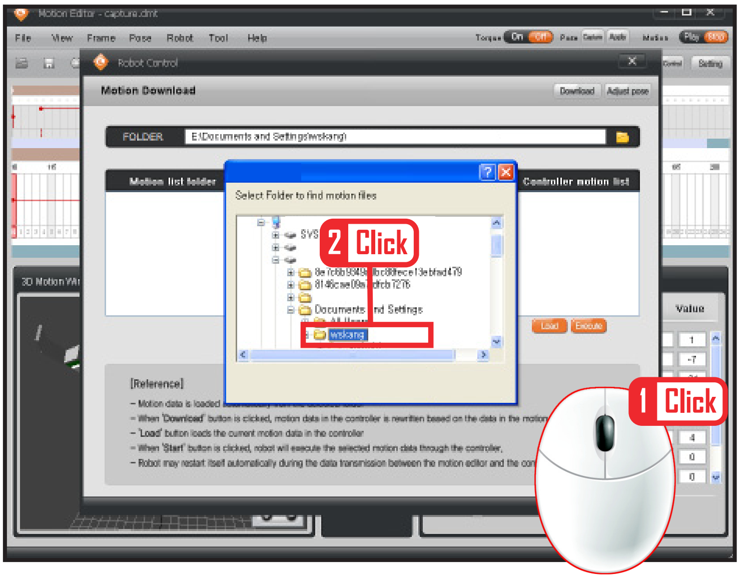
- Motion data is loaded automatically from the selected folder
- When 'Download' button is clicked, motion data in the controller is rewritten based on the data in the motion editor.
- 'Load' button loads the current motion data in the controller
- When 'Start' button is clicked, robot will execute the selected motion data through the controller.
- Robot may restart itself automatically during the data transmission between the motion editor and the controller.

02 Robot Control Window

Motion information download popup opens up. Top section shows the directoy of saved motion files. Left window shows motion list saved in the PC. Files downloaded to the DRC will be listed by number on the right window.

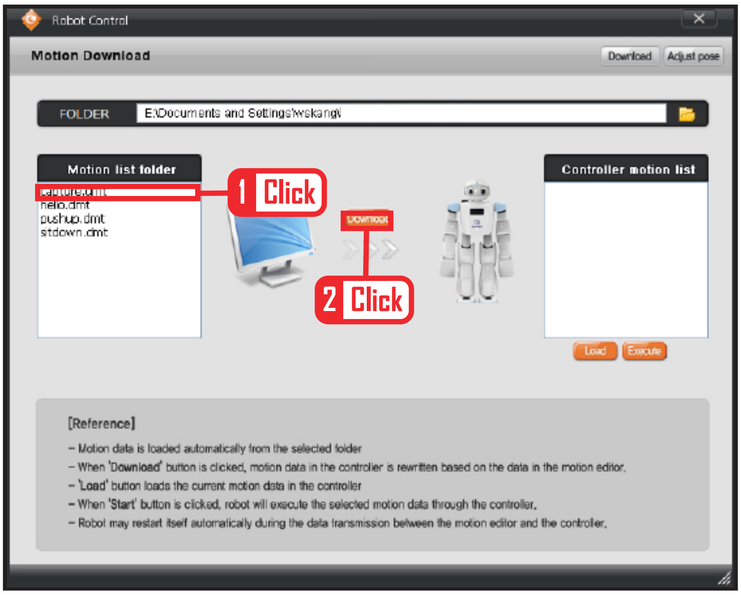
03 Open Saved File Folder

Click on the folder icon at the right side of the folder directory to open up the folder search popup window. Click to select the folder where the motion files are saved.



04 Motion List

Left window shows the list of motions in the selected folder. Place the cursor on the list and click the download icon.



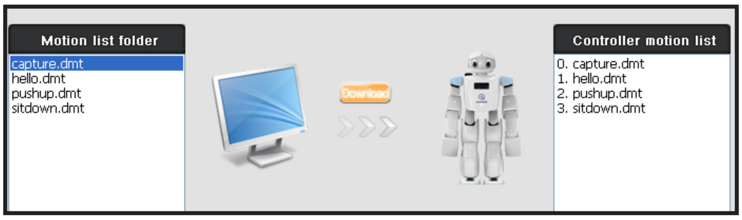
FOLDER: E:\Documents and Settings\wskang\

Motion list folder:
capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt

[Reference]
– Motion data is loaded automatically from the selected folder
– When 'Download' button is clicked, motion data in the controller is rewritten based on the data in the motion editor.
– 'Load' button loads the current motion data in the controller
– When 'Start' button is clicked, robot will execute the selected motion data through the controller.
– Robot may restart itself automatically during the data transmission between the motion editor and the controller.

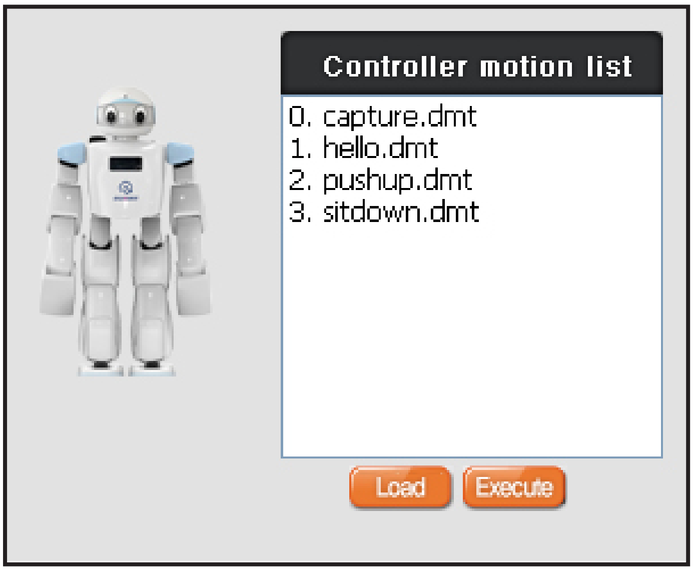
05 Download

Robot motions will be downloaded one at a time.



Motion list folder:
capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt

Controller motion list:
0. capture.dmt
1. hello.dmt
2. pushup.dmt
3. sitdown.dmt



Controller motion list

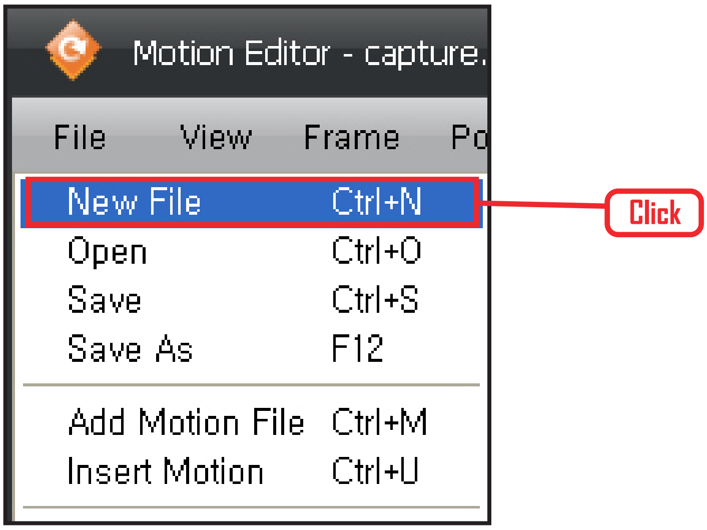
0. capture.dmt
1. hello.dmt
2. pushup.dmt
3. sitdown.dmt

06 Controller Motion List

Once all the motions are downloaded, motions will be listed from number 0 in the controller motion list window on the right side.
Numbers can be called up by index when programming with DR-Visual Logic (Task Editor).
This ends the lesson in robot motion download.

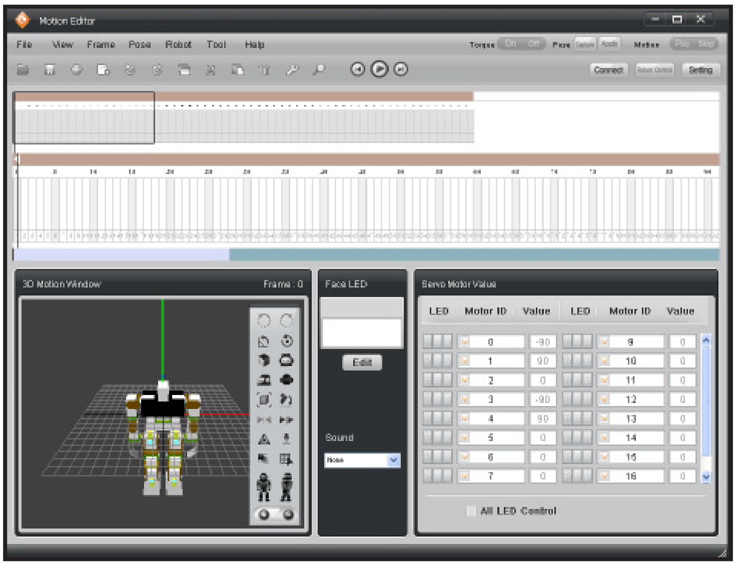
Creating Motion – Step by Step

There are two different methods of creating motion. One method is to use the 3D motion window, motion can be created by clicking on robot joints and using the motion lines. Another way is to capture the motion from the robot. Following lesson will show how to create motion by using both methods.



01 New File

File 〉 Click on 'New File'.



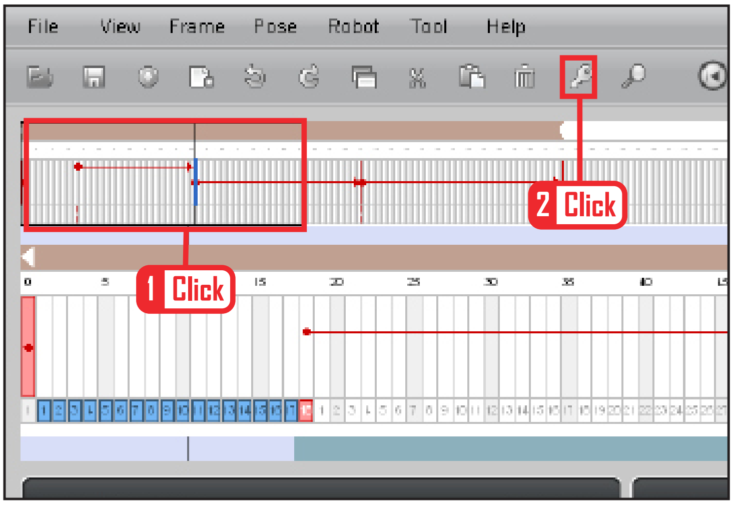
Servo Motor Value

LED	Motor ID	Value	LED	Motor ID	Value
	0	-90		9	0
	1	90		10	0
	2	0		11	0
	3	-90		12	0
	4	90		13	0
	5	0		14	0
	6	0		15	0
	7	0		16	0

□ All LED Control

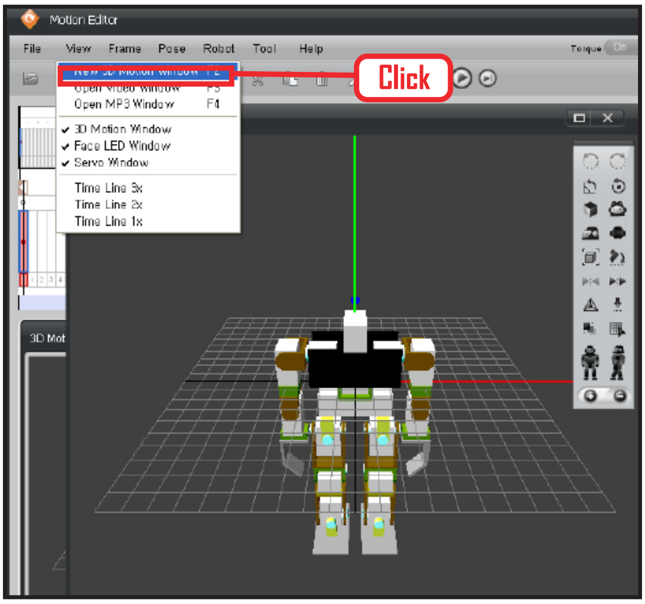
02 New Motion Window

Previous motion window will disappear and new robot 3D motion window will open.



03 First Frame

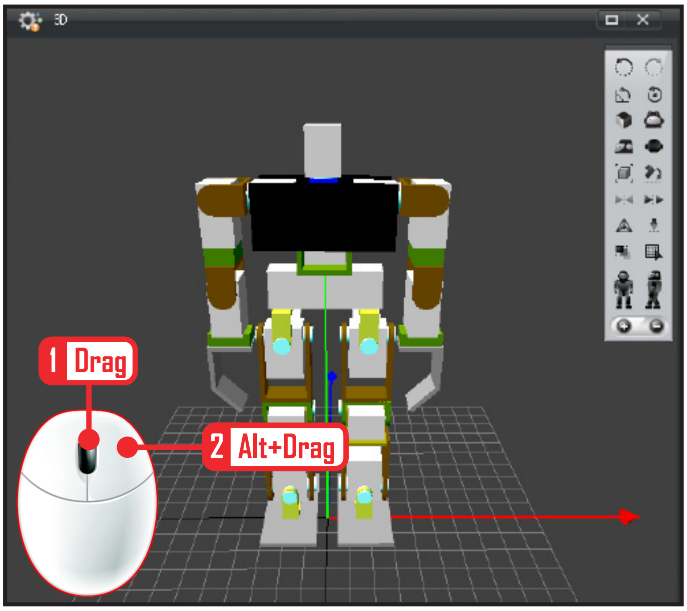
Insert teh basic posture into the first frame. Click on the first frame and then click on the key frame insert.



04 New Motion Window

Motion window can be opened up separtely and enlarged to conveniently edit motion on screen. Total of three 3D motion windows can be opened at same time and placed side by side or top and bottom to be used for editing.

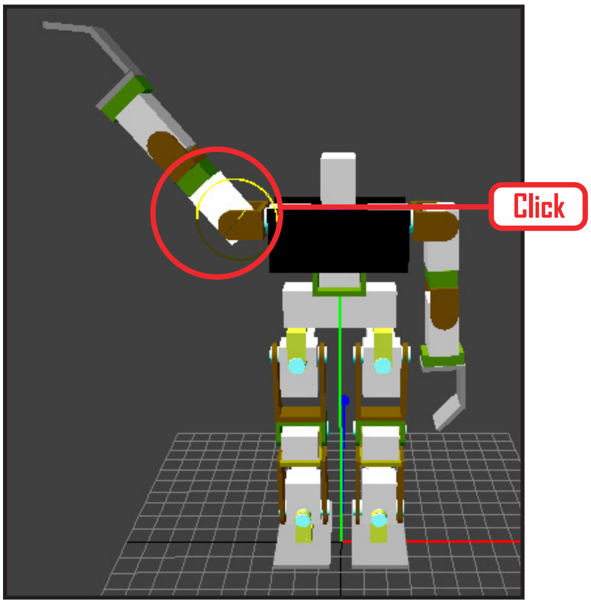
View 〉 Click on 'New 3D Motion Window'.
Use the mouse to drag and elarge the newly opened 3D popup window.



05 Enlarging Robot.

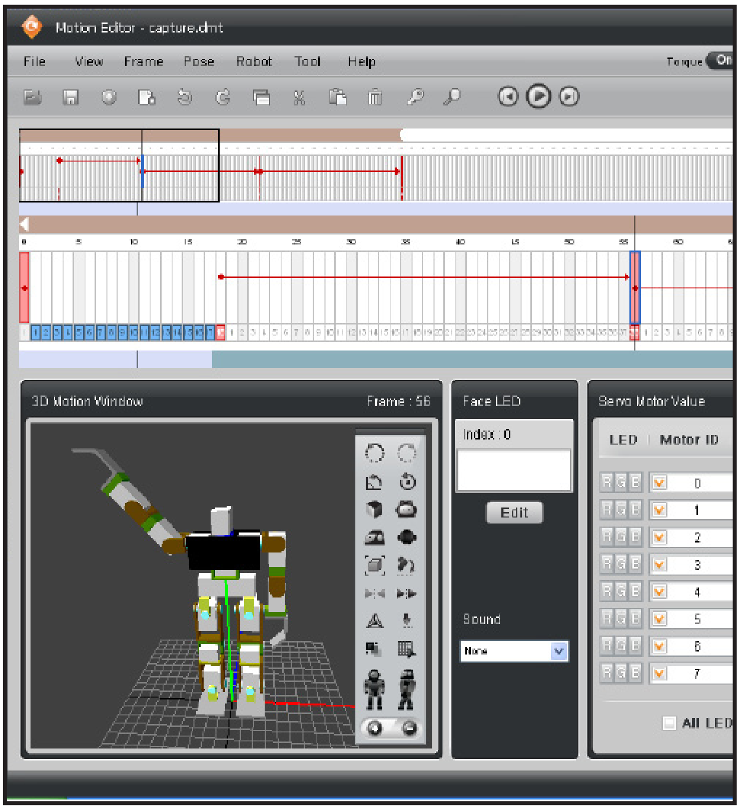
Robot in the edit window can be enlarged. Click on an empty space and use the mouse wheel to zoome in or out.

To change the angle, press and hold the right mouse button and drag. To change the robot position, press shift + press and hold right mouse button and drag.



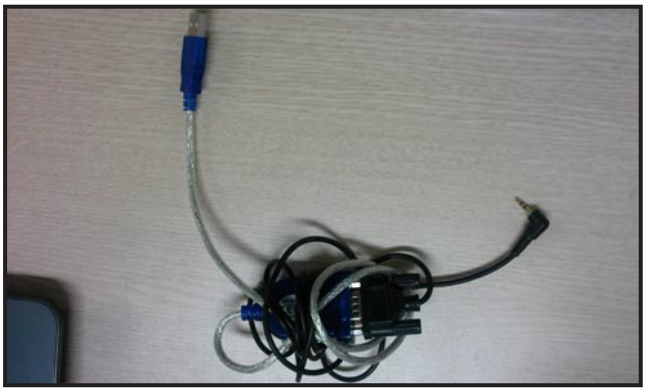
06 Edit Arm Motion

Lifting the arm. Click on the shoulder area and yellow motion line will appear. Click and drag along the motion line to lift the arm.



07 Insert Key Frame

Insert the lifted arm motion into the frame. Click on the desired frame and then click on 'key' icon to insert the motion.

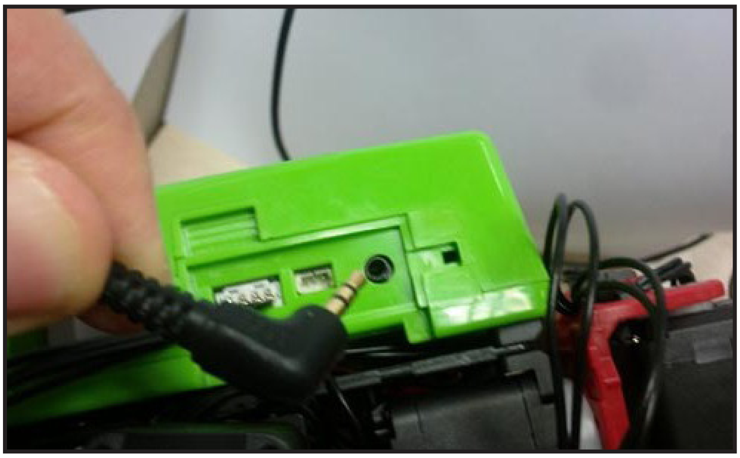


08-1 USB to Serial Converter

Start connection to the robot.
USB to Serial conversion cable that connects robot to the PC/Notebook USB port.

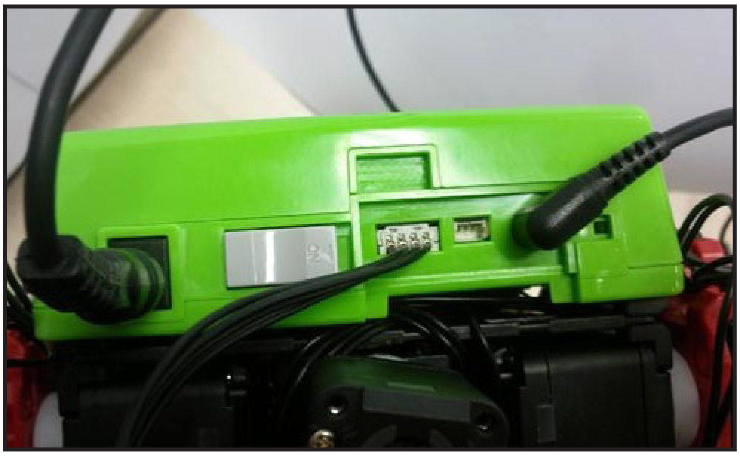
08-2 USB port

Connection to the PCs with Serial Port in the back can made using the serial cable but connection to notebook computers without the serial port requires USB to Serial converter.



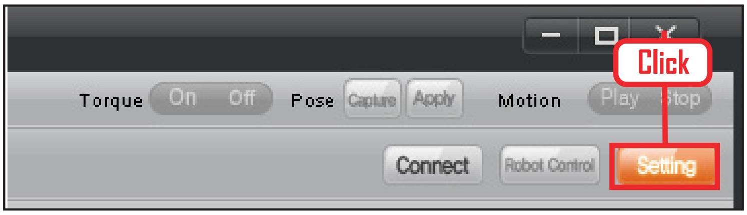
08-3 Connecting to Robot

Connect the RS232c audio jack to the robot.



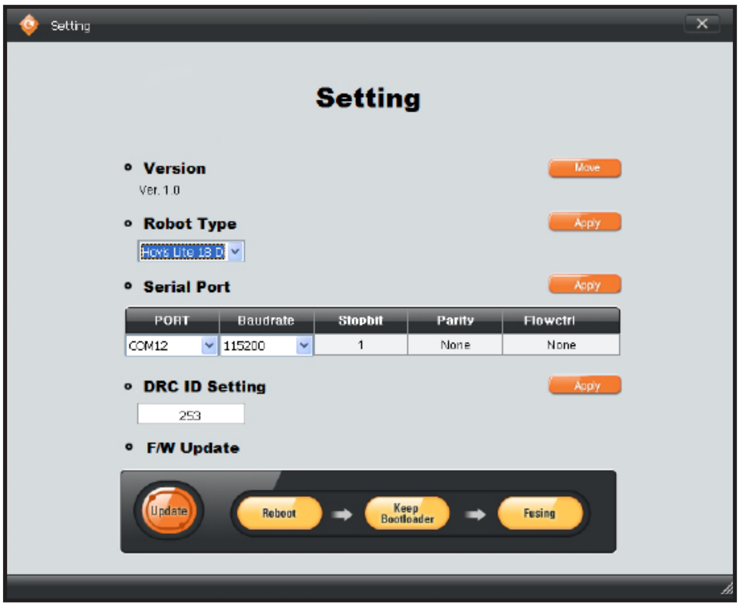
08-4 Robot Port

Looking at top of the DRC controller, you will find serial port connecting to the PC, head side servo motor port, and power port.
Photo shows all three ports connected.



08-5 Robot Connection Button

Menu for making connection to the robot is located at top right of DR-SIM window.
Click on 'Environment Setup' button to configure the COM Port.



Setting

- Version: Ver. 1.0
- Robot Type: Hovis Lite 18 D

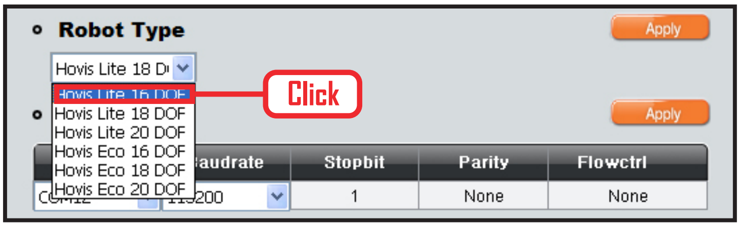
Serial Port

PORT	Baudrate	Stopbit	Parity	Flowctrl
COM12	115200	1	None	None

- DRC ID Setting: 253
- F/W Update

09 Environment Setup

Environment Setup window shows DR–SIM version, robot selection, and Communications setting.



Robot Type

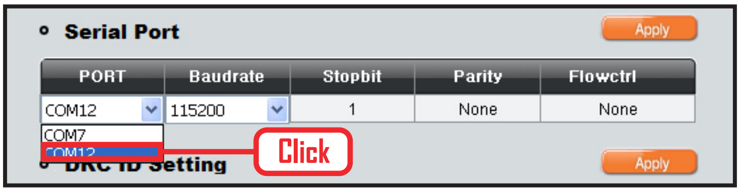
Hovis Lite 18 D

- Hovis Lite 16 DOF
- Hovis Lite 18 DOF
- Hovis Lite 20 DOF
- Hovis Eco 16 DOF
- Hovis Eco 18 DOF
- Hovis Eco 20 DOF

PORT	Baudrate	Stopbit	Parity	Flowctrl
COM12	115200	1	None	None

10 Robot Selection

DR–SIM provides total of 6 different types of humanoid robot. Most basic robot is the 16 axis humanoid.
Select the type of robot that was assembled.
Select Hovis Lite 16 DOF



Serial Port

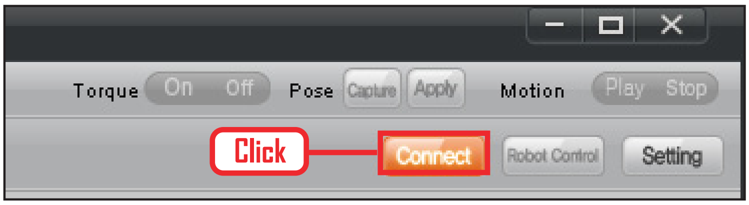
PORT	Baudrate	Stopbit	Parity	Flowctrl
COM12	115200	1	None	None

- COM7
- COM12

DRC ID Setting

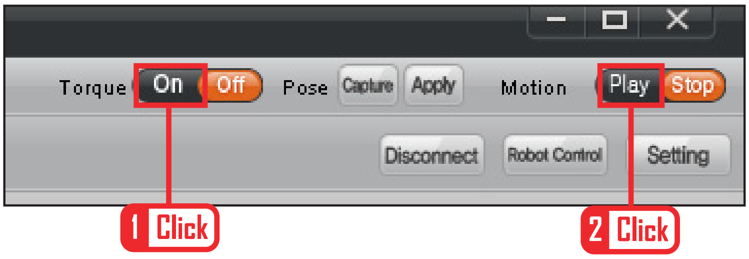
11 COM Port Selection

“PORT” shows the COM Port numbers that can be selected. Select one of the ports. If there is no connection, go to the hardware properties in windows and check the number of the COM port that can be used.



12 Connecting

Click on the ‘Connect’ icon.

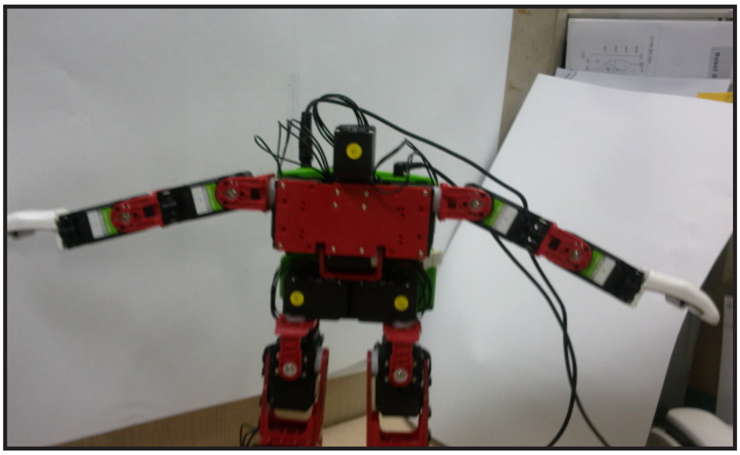


13 Connection

As shown in the left photo, Torque button becomes active when connection to the robot is made. Click “Torque On” button to operate the robot and click ‘Play’ button to play current motion.

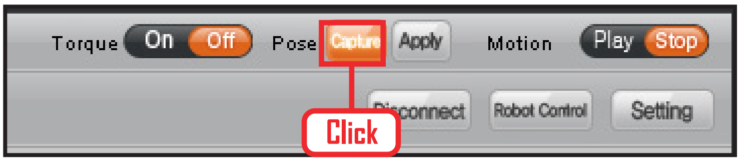
14 Robot Motion for Capture

This lesson will show how to caputre and edit motion from the robot.Click "Torque Off" button and then manually manipulate the robot to make desired motion.



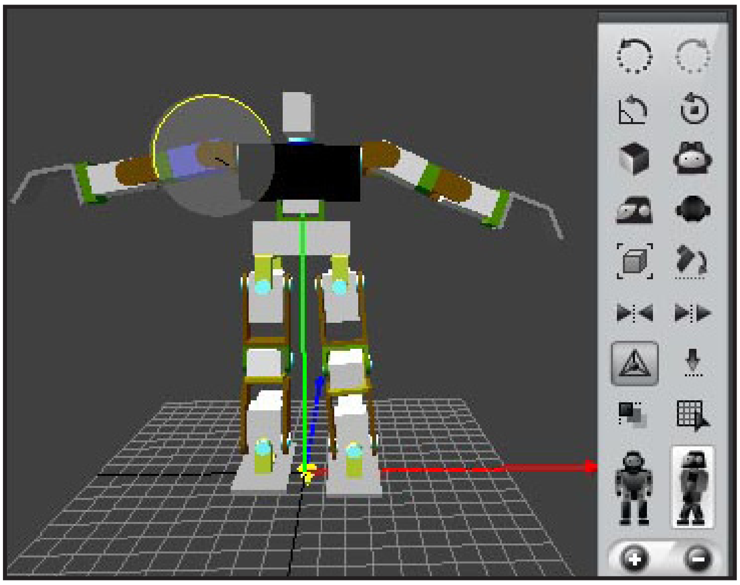
15-1 Capture

Click 'Capture' button.



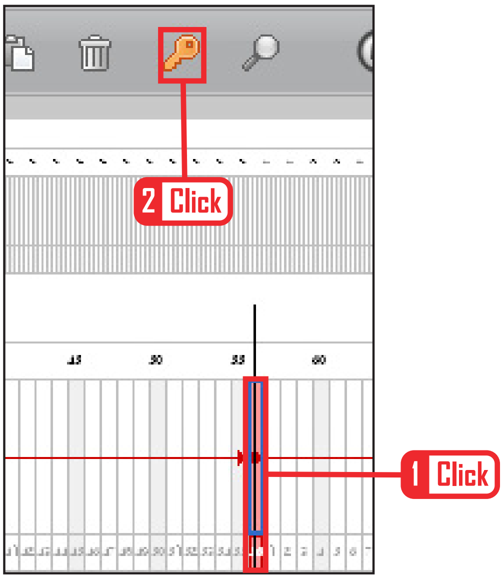
15-2 Show Captured Motion in 3D Window

Captured motion is shown in the 3D motion window as soon as capture button is clicked.



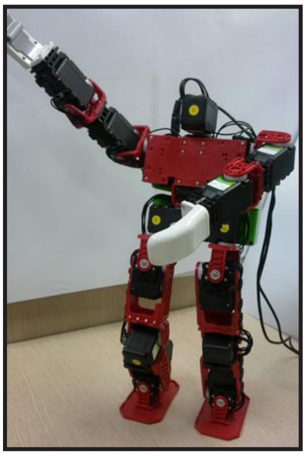
15-3 Insert Key Frame

Insert captured motion into the desired frame. Click on the frame first and then click on the key frame.



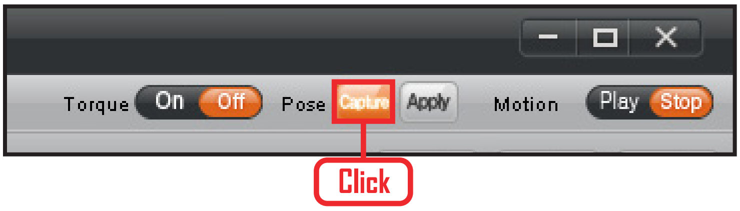
16-1 Different Motion

Manually make a different motion .



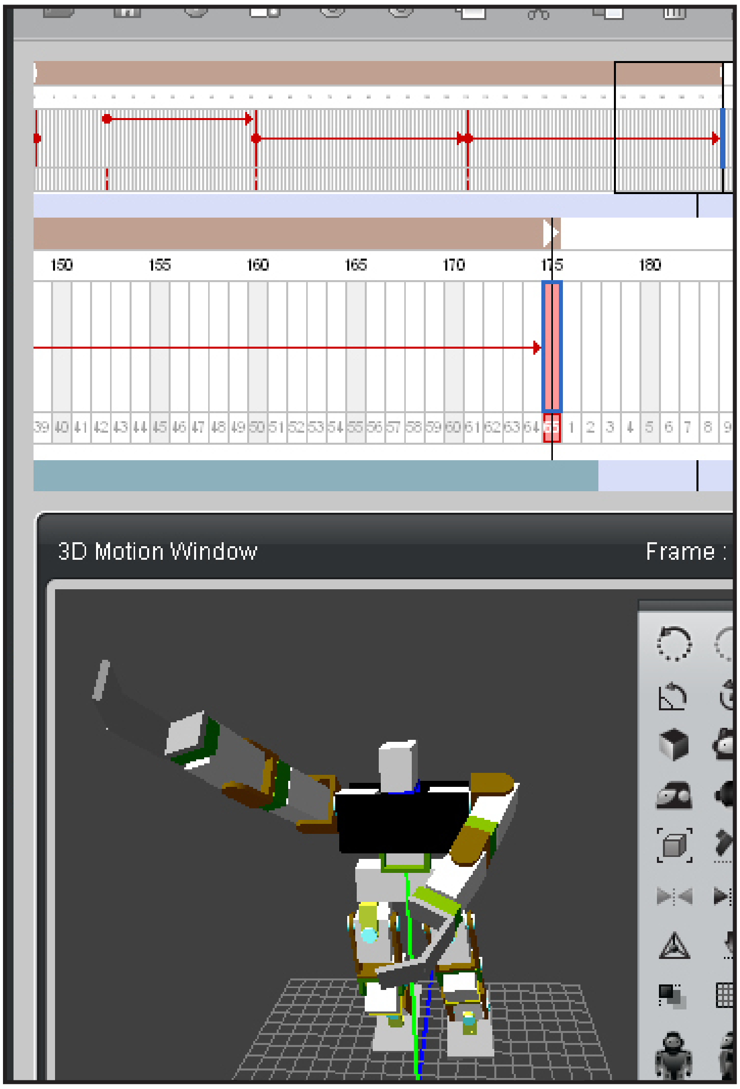
16-2 Capture

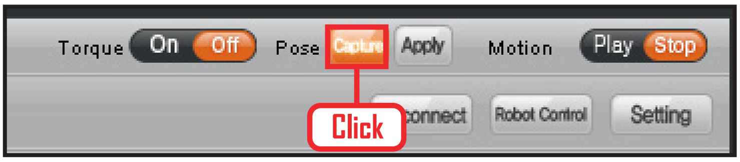
Capture.

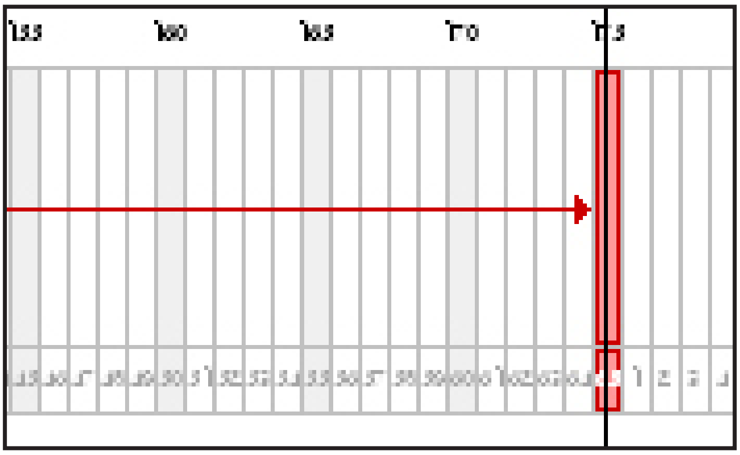


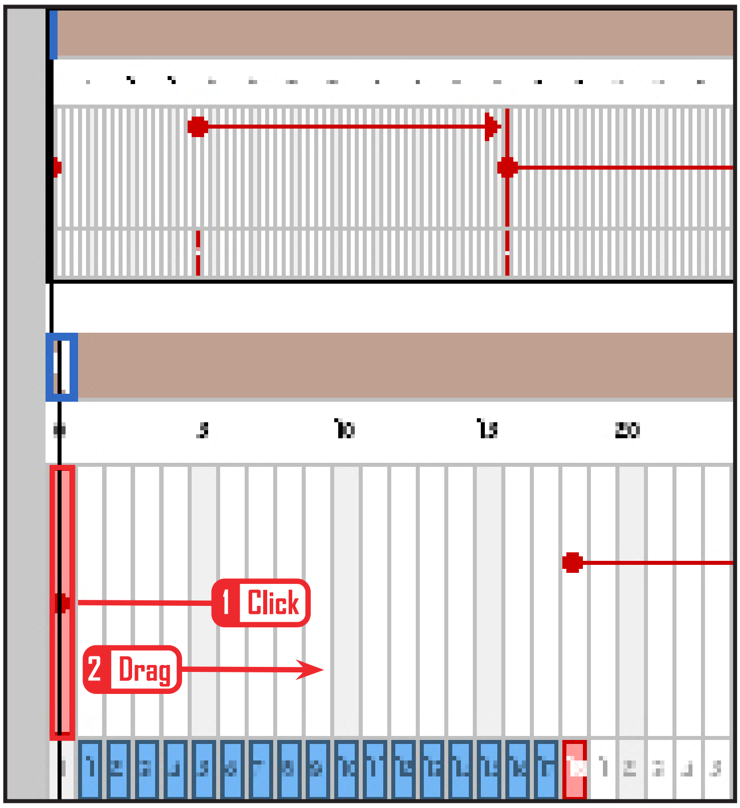
17 Check Motion

Compare the manually made motion with the motion in the 3D motion window.









18 Capturing

Capture the newly made motion

19 Insert Key Frame

Insert motion in the desired frame.

20 Delay Value

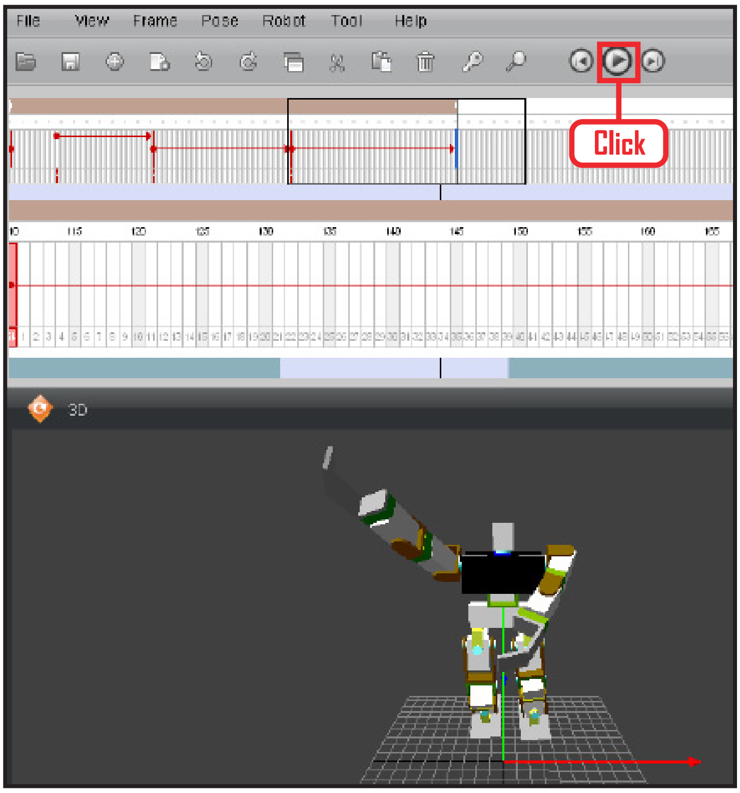
Robot may make a suden movement if there is a large motion difference between the first and the second motion key frame. To prevent such a sudden movement, there is a way slow down the first motion.

Click bottom of the frame and drag to the right with left mouse button pressed. Such an action will show up as photo on the left and Delay value will be created.

21 Screen Play

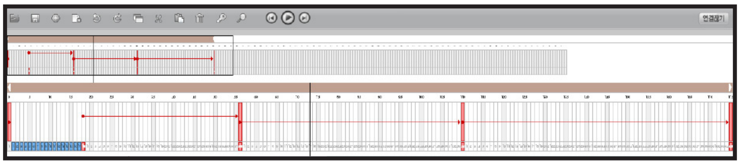
Play the created motion in the motion window.

Click 'Play' icon.



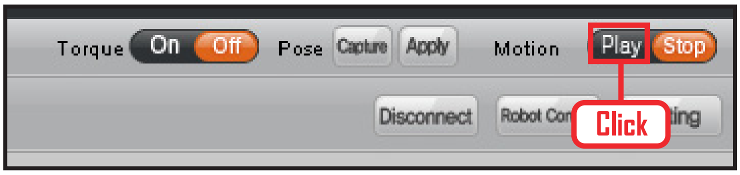
22 Play

Progress line shows the motion being played progressing on the time frame.



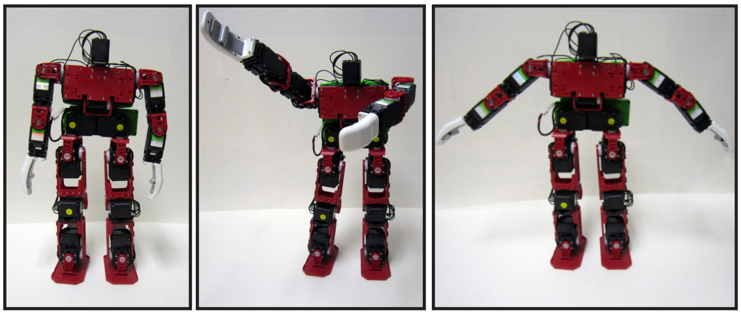
23 Play On Robot

Apply and play the motion on robot. Click 'Play' button located near top right.



24 Robot Motion

Left photo shows the motion created in 3D motion window. middle and right photos show captured motions.When played, motions will be played consecutively.

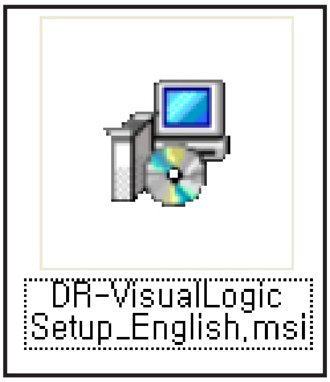


Installation

DR-Visual Logic Introduction

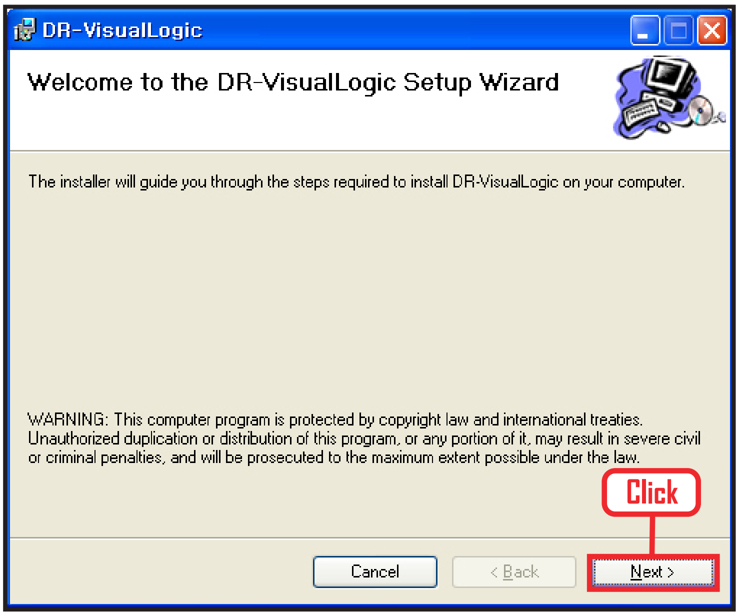
DR-Visual Logic is a Drag & Drop type graphic robot programming tool derived from the robot programming language developed by Dongbu Robot. DR-Visual Logic has been customized to work with Dongbu Robot DRC controller by modularizing the DRC functions. Drag & drop method using the mouse makes DR-Visual Logic easy to program even by the novices and by using the provided C-like tab, text codes converted from the graphic program can be viewed immediately. As the codes are similar to the C language, it will also help the novice programmers in learning the C language. DR-Visual Logic is one of the easiest and yet powerful programming tools in the market and its versatility makes it equally popular with novice and advanced users alike. Planned upgrade to the program to make it even more versatile and powerful includes upgraded DRC function modules, motion modules, and integrated simulation

- System Requirements
- Minimum Intel Pentium 800 Mhz
- Windows XP, Windows 7
- Minimum 256 MB RAM
- Hard disk space 300 MB required
- USB Port
- Macintosh (Under Development)



01 Installation File

Click on the installation file.


DR-VisualLogic

Welcome to the DR-VisualLogic Setup Wizard

The installer will guide you through the steps required to install DR-VisualLogic on your computer.

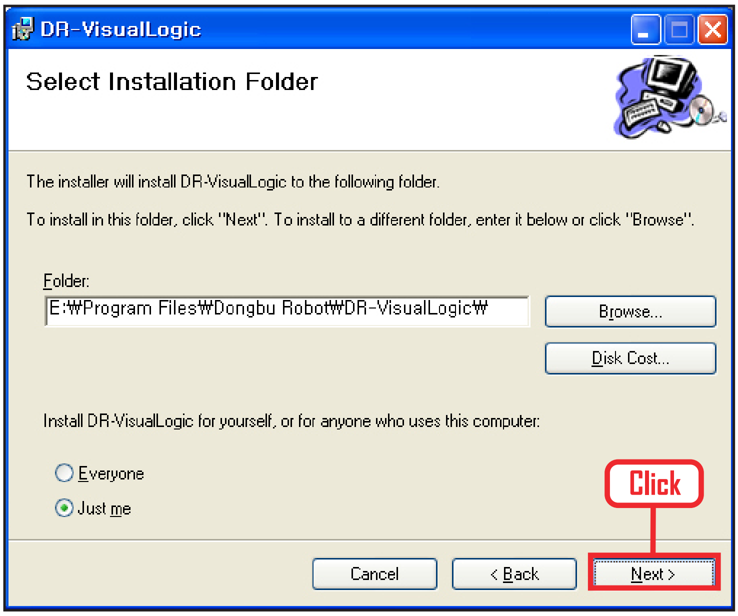
WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

02 Start installation Wizard

Click “Next” button.

03 Select Installation Folder

Click "Next" button.



DR-VisualLogic

Select Installation Folder

The installer will install DR-VisualLogic to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

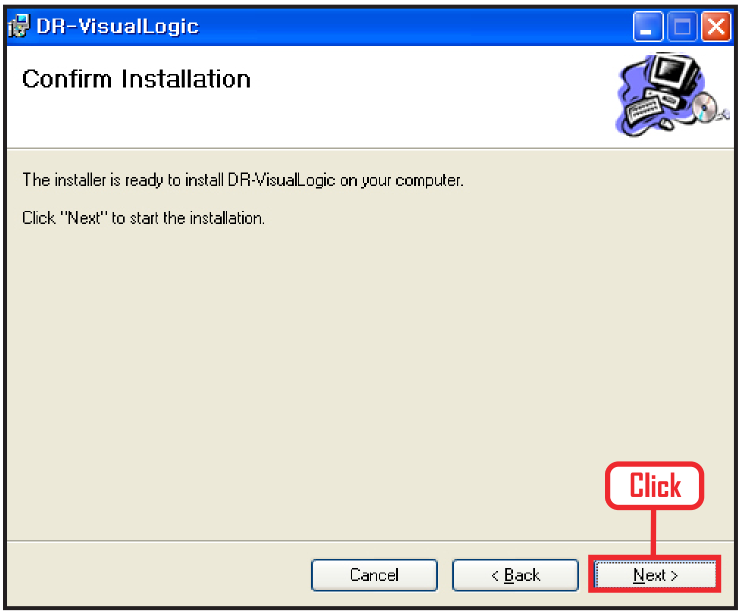
Folder:
E:\Program Files\Dongbu Robot\DR-VisualLogic\

Install DR-VisualLogic for yourself, or for anyone who uses this computer:

○ Everyone
● Just me

04 Confirm Installation

Click "Next" button.



DR-VisualLogic

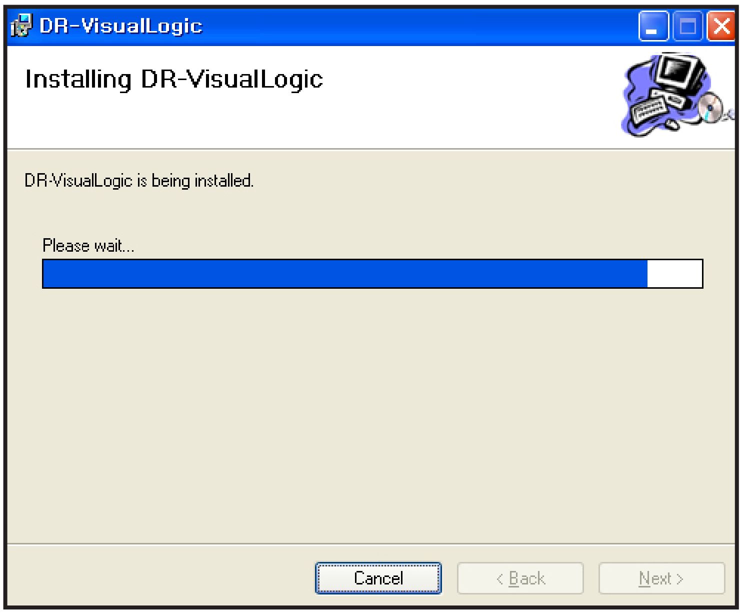
Confirm Installation

The installer is ready to install DR-VisualLogic on your computer.

Click "Next" to start the installation.

05 Start Installation

Starting installation. Wait for the progress bar to end.

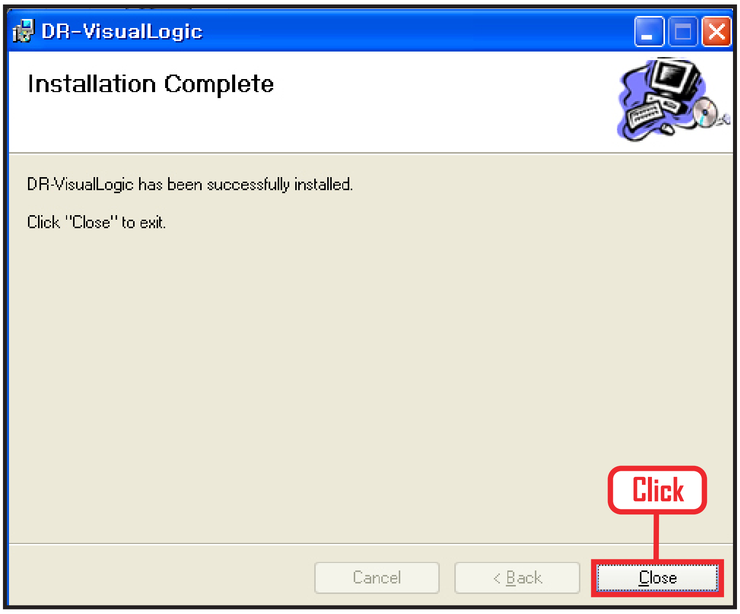


DR-VisualLogic

Installing DR-VisualLogic

DR-VisualLogic is being installed.

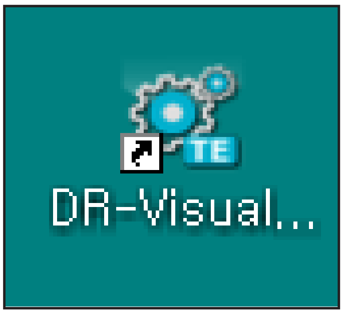
Please wait...



Installation Complete

DR-VisualLogic has been successfully installed.

Click "Close" to exit.



06 Finish Installation

Click “Close” button
Program installation complete.

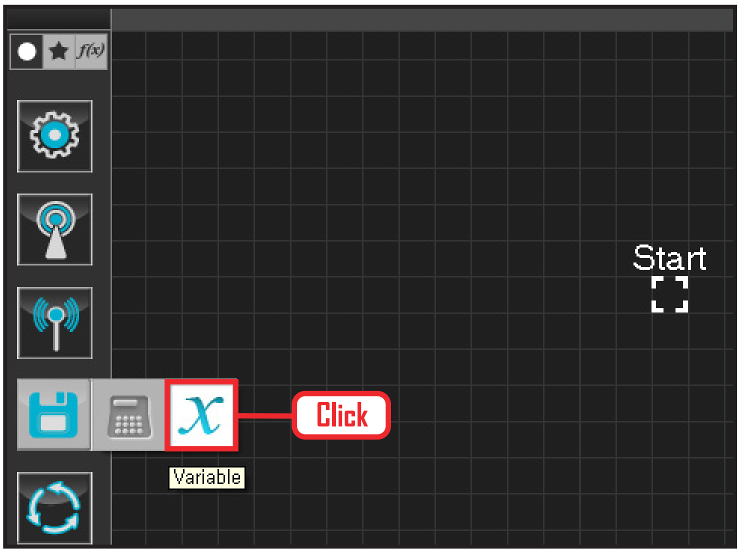
07 Check executable file

Check for the executable file, desktop shortcut icon and from Windows Start 〉All Programs 〉 Dongbu Robot 〉DR–VisualLogic.
Click on the executable file to run the program.

First Program Step by Step

Sample Progam Description

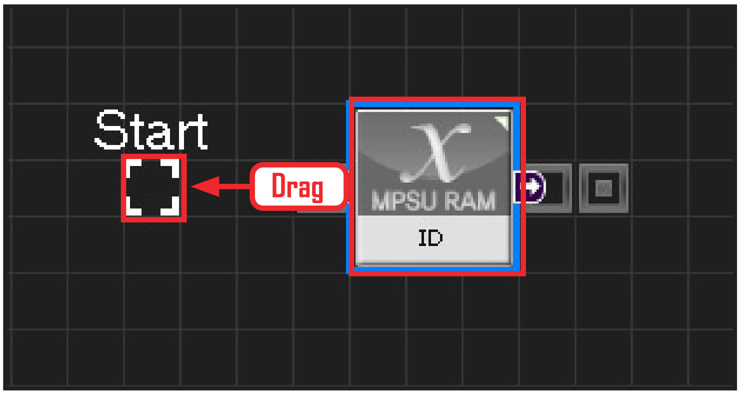
Robot has both arms spread out, lower one arm to the side of the body. 16 axis humanoid robot will spread out both arms when all motors are aligned in the center. one of the arm will be lowered to the side of the body.



01 Assign Variable

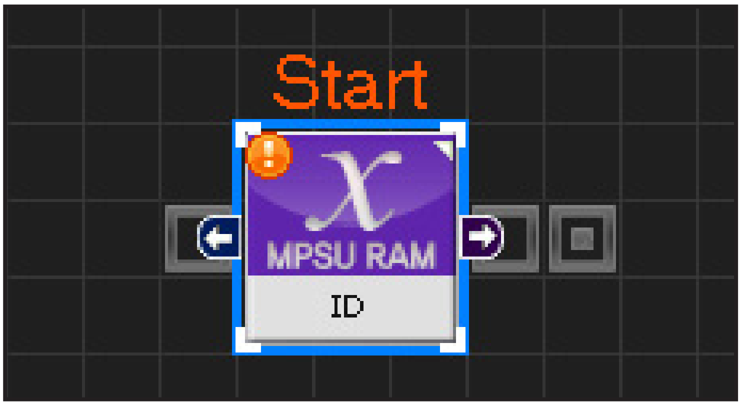
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

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.

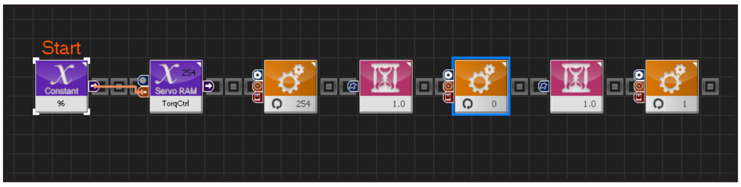


03 Start Programming

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.

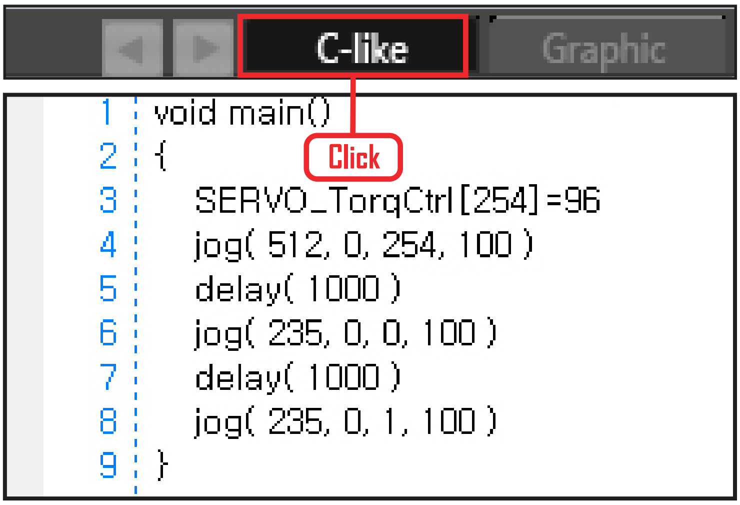
04 Entire Program

Photo to the left shows the entire progam lowering the robot arm by moving the motor.



05 Viewing C-Like

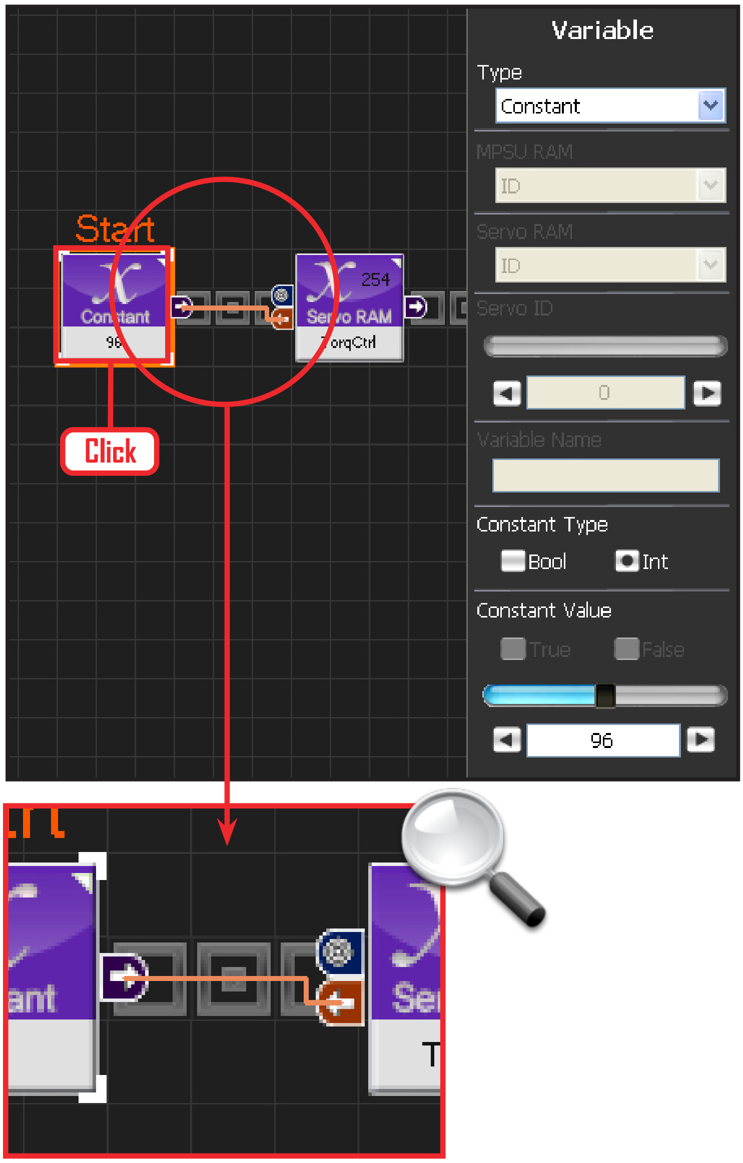
Click the 'C-like' tab near the top right and task programming window will open as shown in the photo to the left. This is the task window of the entire program. Codes are very similar to the C language structure so studying the codes will help the user become familiar with the C language structure.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



```
void main()
{
   SERVO_TorqCtrl[254]=96
   jog( 512, 0, 254, 100 )
   delay( 1000 )
   jog( 235, 0, 0, 100 )
   delay( 1000 )
   jog( 235, 0, 1, 100 )
}
```

06 Variable Setup

This section allows the servo motor to operate on it's own. Select Constant as the Variable Type. In properties, set constant value as 96. When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.



Variable

Type: Constant

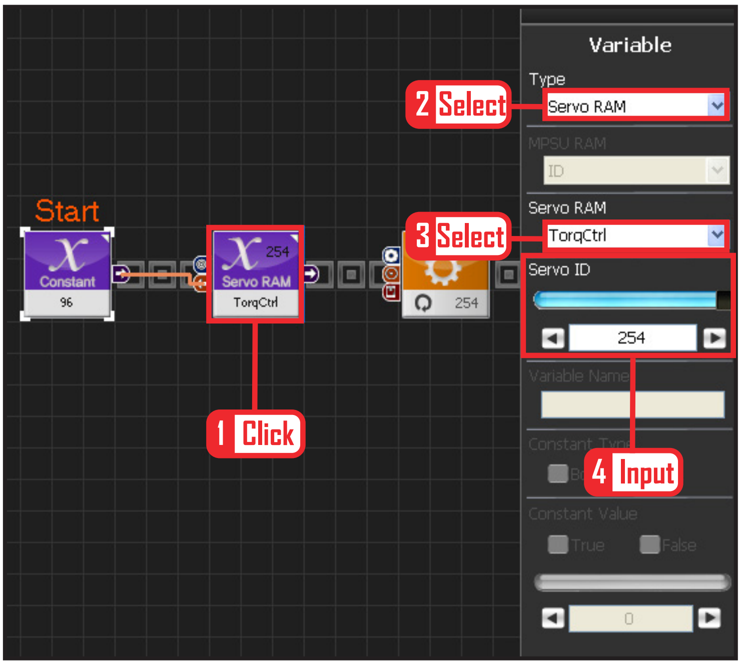
ID

ID

0

Constant Type: □ Bool ● Int

Constant Value: 96

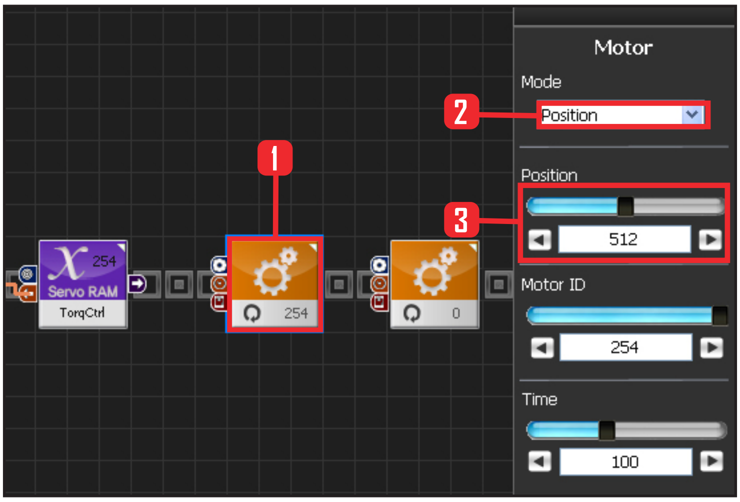


Type: Servo RAM
Servo RAM: TorqCtrl
Servo ID: 254

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉 Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos.

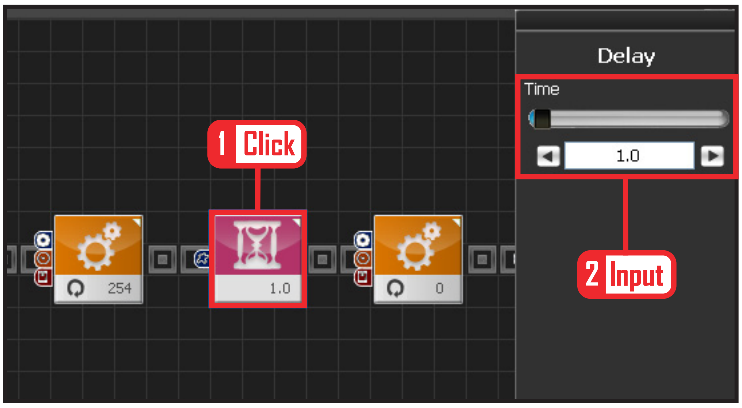


Mode: Position
Position: 512
Motor ID: 254
Time: 100

08 Set Angle to All Servos

This section sets all servo motor angles to the center.

Select Motion 〉 Motor.
Select Mode : Positon. adjust angle.
Set Position : 512 . 512 means motor will be sent to the center
Set Motor ID : 254 . 254 means it will be applied to all connected servos.
Set Time : 100 . 1 unit = 11.2ms. 100 units would be approximately 1.12s.
It means motors will be positioned at the desired angle for 1.12s.

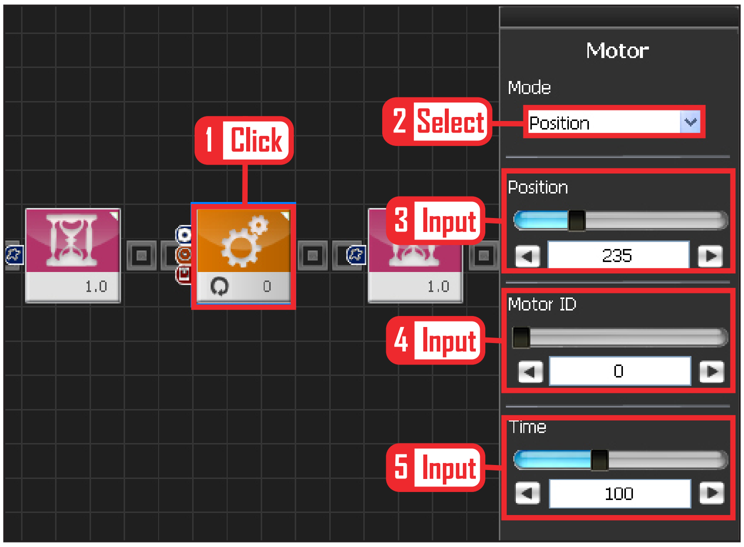


Time: 1.0

09 Delay

This section delays the motor for 1s before starting.

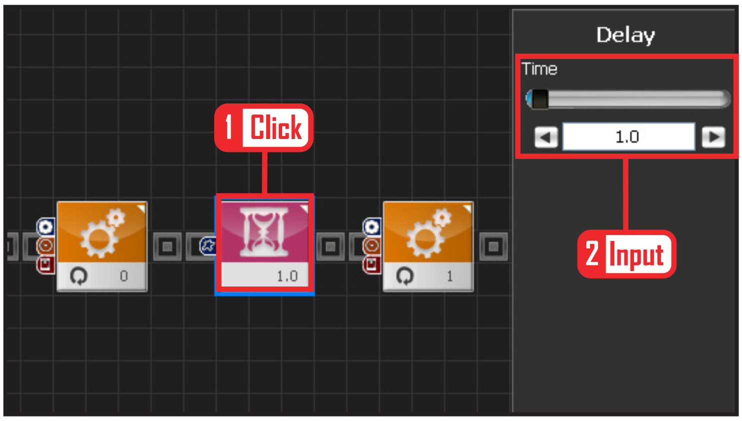
Select Flow 〉 Delay module.
Set Time : 1.0 . It means delay of approximately 1s.



10 Setup Motor ID 0 (Right Shoulder)

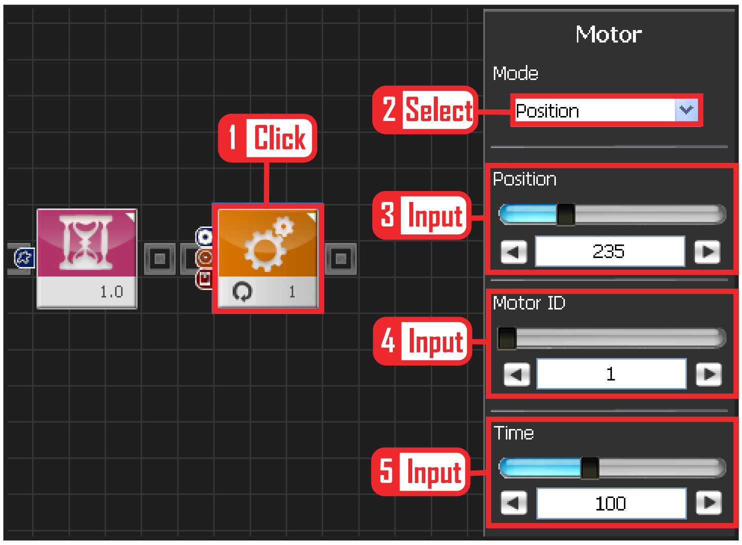
Creating attention posture (Basic Posture)
When all robot motors are aligned to the center, humanoid robot arms will be stretched out to the side. Setup below lowers one arm to the side of the body.

Select Motion 〉 Motor.
Select Mode : Position.
Set Position : 235. 235 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.
Set Motor ID : 0. Right shoulder motor has ID 0
Set Time : 100. Motor will turn to the desired angle in approximately 1.12s.



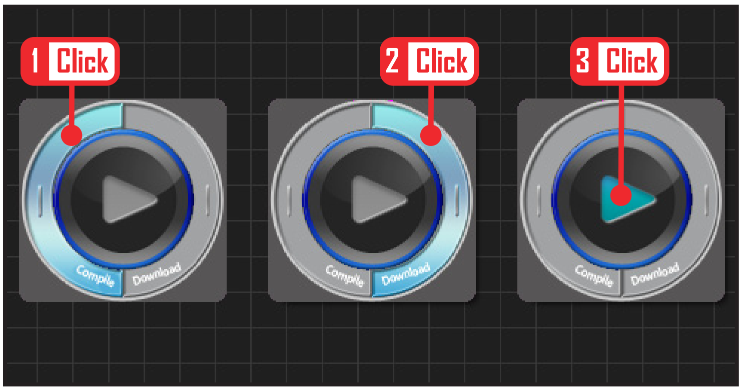
11 Delay

Setup below makes the motor wait for 1s before starting.
Select Flow 〉 Delay Module.
Set Time : 1.0 . Delay start by 1s.



12 Set Motor ID 1 (Right Arm)

Select Mode : Postion.
Set Position : 235. 235 lowers the horizonally stretched arm to vertical down position.
Set Motor ID : 1. Right upper arm motor connected to the should has motor ID 1.
Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s.

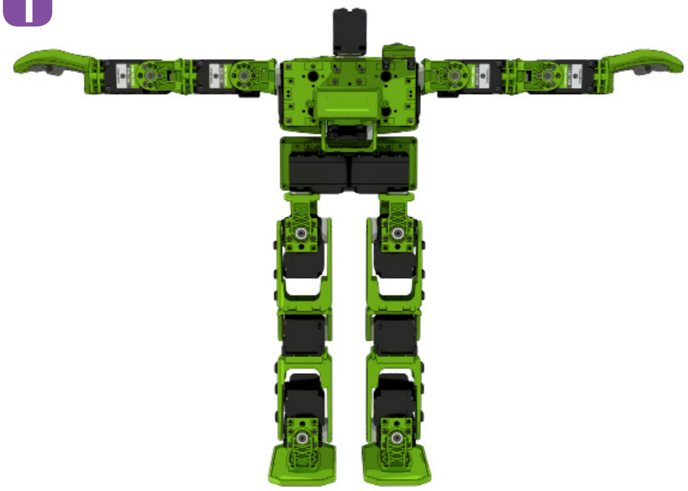


13 Download

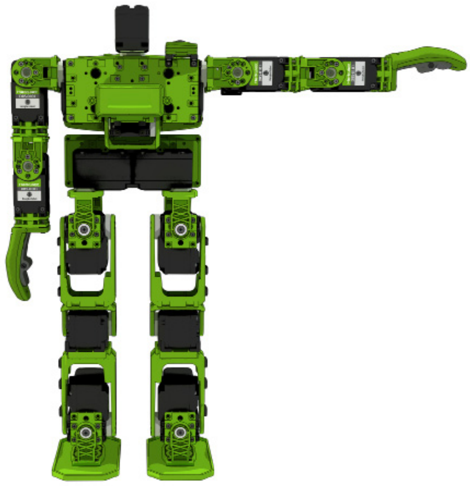
Compile after programming done → Download to robot → Run.

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

1



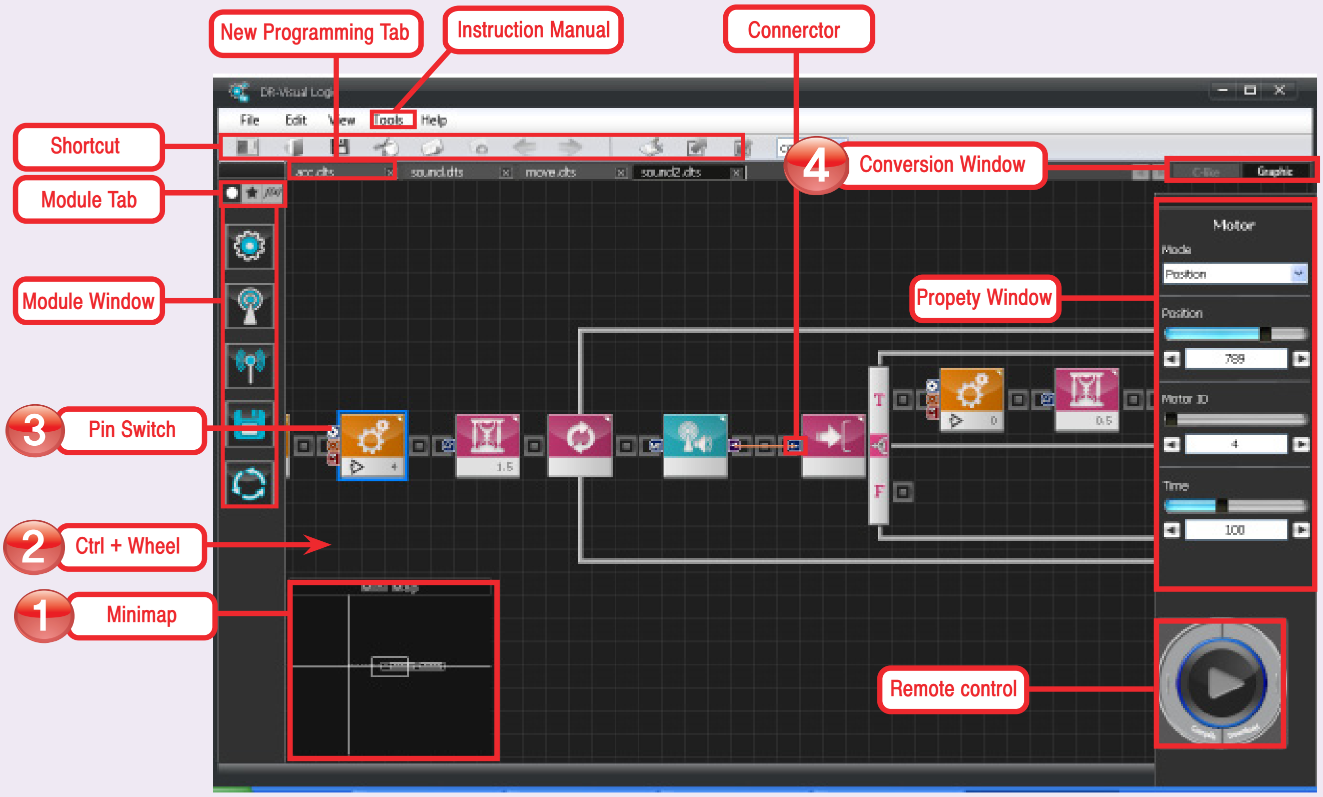
2



14 Robot Motion

Right arm will lower to the side from horizontally stretched out position.

User Interface



❶ Mini Map: Controlled by dragging, shows current position even in lengthy program, jump to any position.

❷ Wheel Up/Down : Screen zoom in/out

❸ Pin Switch : Shows pin names of the current module, disappears whenc clicked again.

❹ Conversion Window : From Graphic to Text. Converts graphic programming source to text source, Similar to C language structure.

0 **Shortcut :** Group of shortcut icons for frequently used commands.

1 **Module Tab :** All modules

2 **Module Window**

3 **New Programming Tab**

4 **Instruction Manual**

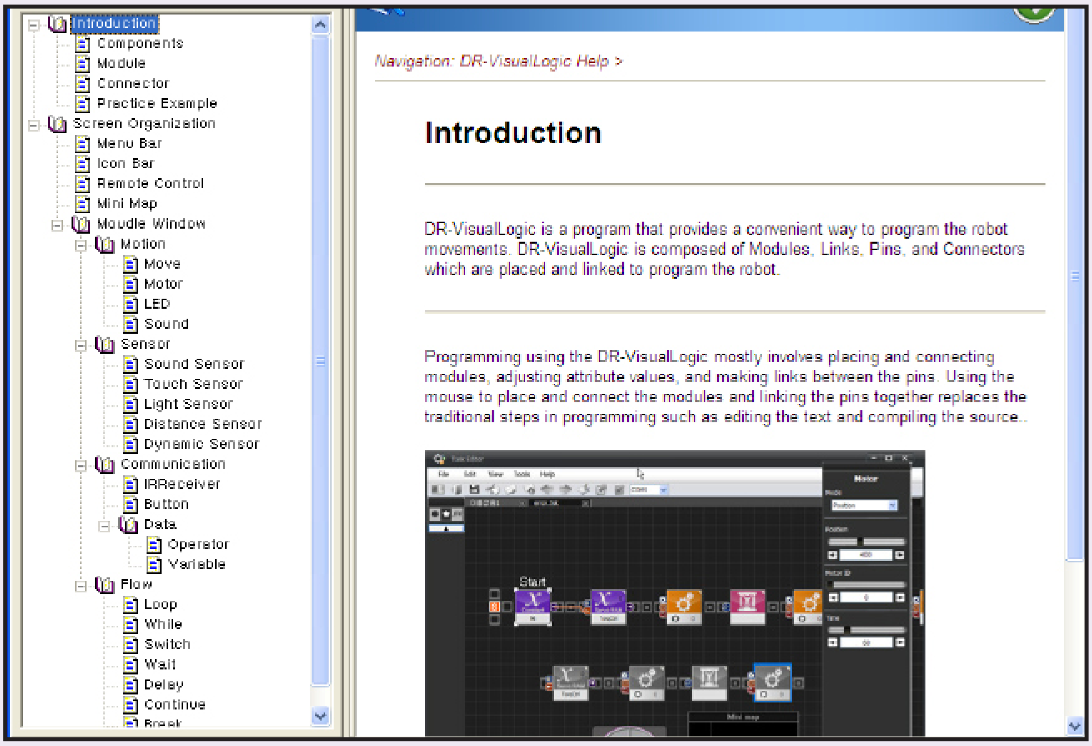
5 **Connector**

6 **Property Window**

7 **Downloader**

Help

From the menu, click Tools 〉Help, Help window will popup as shown belw. We recommend users to read the Help files prior to using the DR-Visual Logic. (**Click: Help 〉Index 〉Timeline** → Window below will open up)



Navigation: DR-VisualLogic Help >

Introduction

DR-VisualLogic is a program that provides a convenient way to program the robot movements. DR-VisualLogic is composed of Modules, Links, Pins, and Connectors which are placed and linked to program the robot.

Programming using the DR-VisualLogic mostly involves placing and connecting modules, adjusting attribute values, and making links between the pins. Using the mouse to place and connect the modules and linking the pins together replaces the traditional steps in programming such as editing the text and compiling the source..

Outline

Organization
Module
Connector
Practise

Screen Organization

Menu bar
Icon bar
Remote Control
Mini Map
Module Window

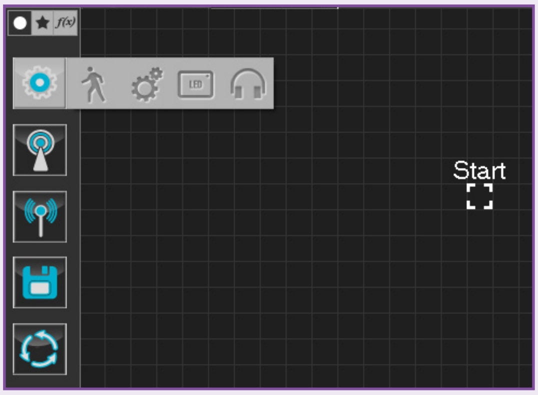
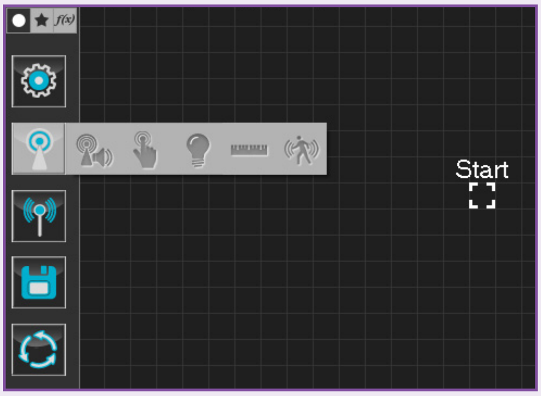
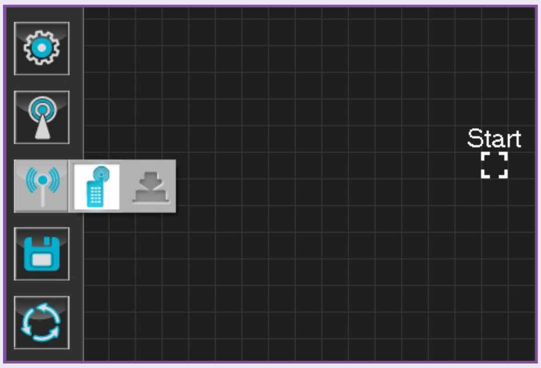
Module Window

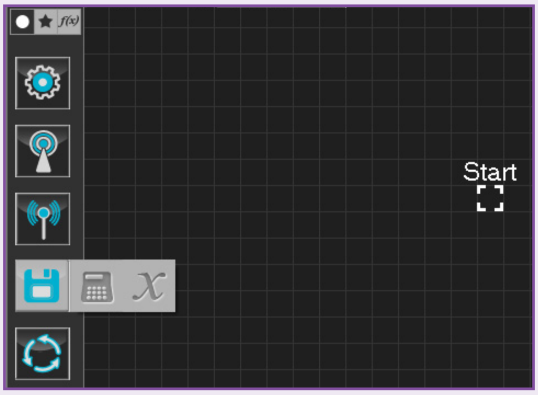
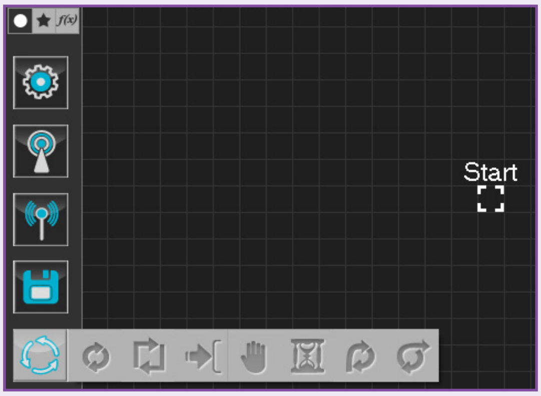
Motion : Move(Saved robot motion), Motor(servo motor), LED, Sound
Sensor : Sound Sensor, Touch Sensor, Light Sensor,
Distance Sensor(Distance Sensor, PSD Digital, PSD Analog),
Dynamics Sensor(Accermeter, Zyro sensor)
Communication : IRReceive, ZigBee, Button
Data : Operator, Variable
Flow : Loop, While, Switch, Wait, Delay, Continue, Break

Programming Module

DR–Visual Logic is comprised of follwing modules.

Module Pack contains all programming modules required to create a program. Each module is supported by the DRC controller function. 'Description': location of each part on humanoid robot and short description of the corresponding module.

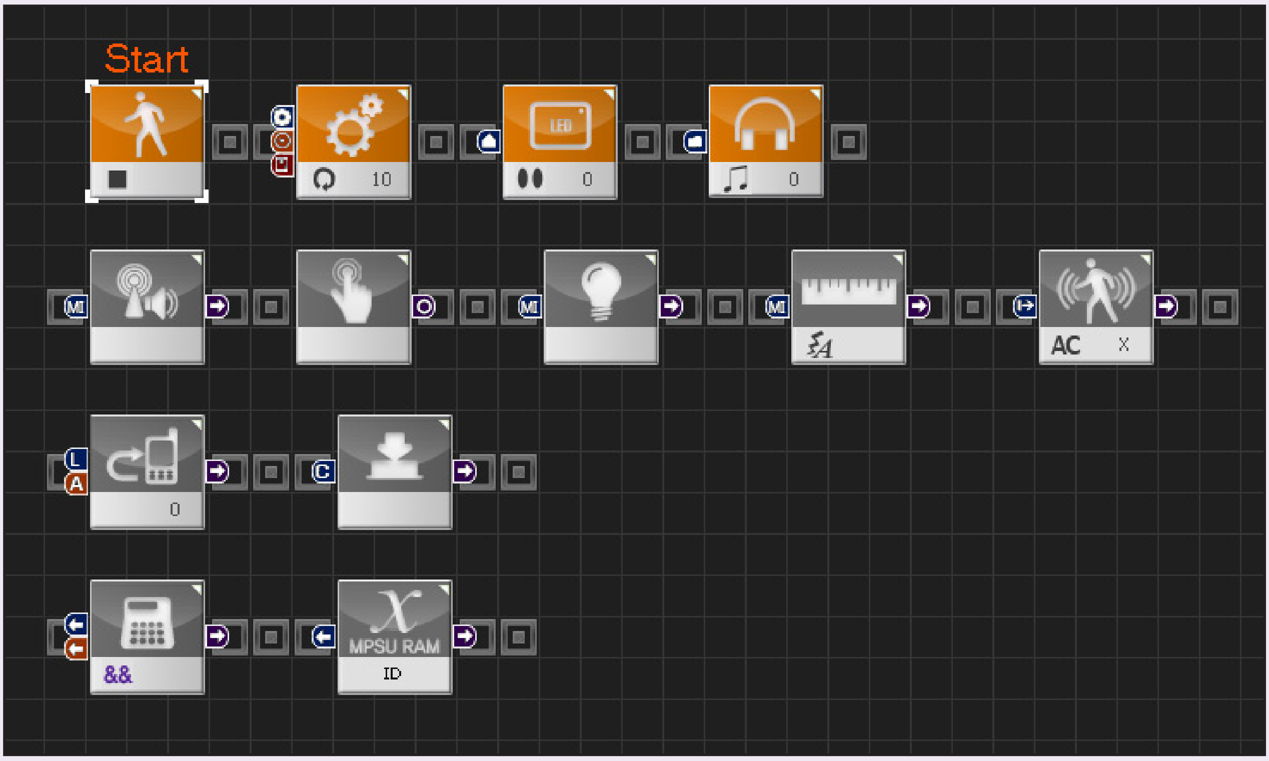
Module Pack	Picture	Module	Description
Motion		Move	Run saved robot motion
		Motor	Position/speed control by each motor
		LED	Head – run saved LED Back – Control LED on controller
		Sound	Sound Buzzer
Sensor		Sound Sensor	Internal, distinquishes Left & Right
		Touch	Recognize touch on head module
		Light	Internal, back, measures light
		Distance	Measures distance
		Dynamics	Dynamics, Measures acceleration and angular speed.
Communi cation		IRRciever	Recognize remote control data
		Button	Reconginze rear controller button.

Module Pack	Picture	Module	Description
Data		Operator	Operator
		Variable	Register data user declare variable/ constant
Flow		Loop	Endless loop/for statement
		While	Continue loop while condition met.
		Switch	Control branch, if–else
		Wait	Wait while condition met
		Delay	Delay for set time
		Continue	Return to beginning of loop
		Break	Exit loop

All DR-Visual Logic modules are either regular or flow modules.

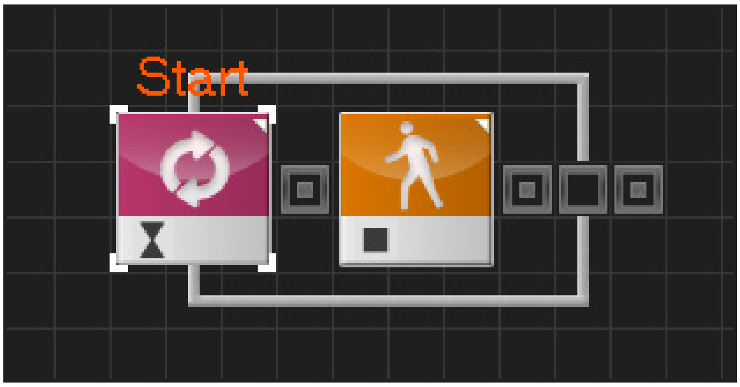
Regular modules are connected together and used sequentially.

All modules except for the flow modules are regular modules.



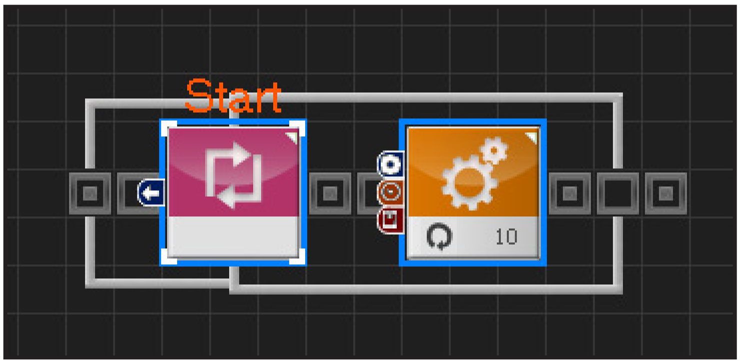
From the top, module icons represent Motion, Sensor, Communicaiton, and Data .

Flow modules connect to the regular modules and control the flow of the program with loop, switch, and etc. Unlike regular modules, outline appers around the flow modules when they are connected to the regular modules.



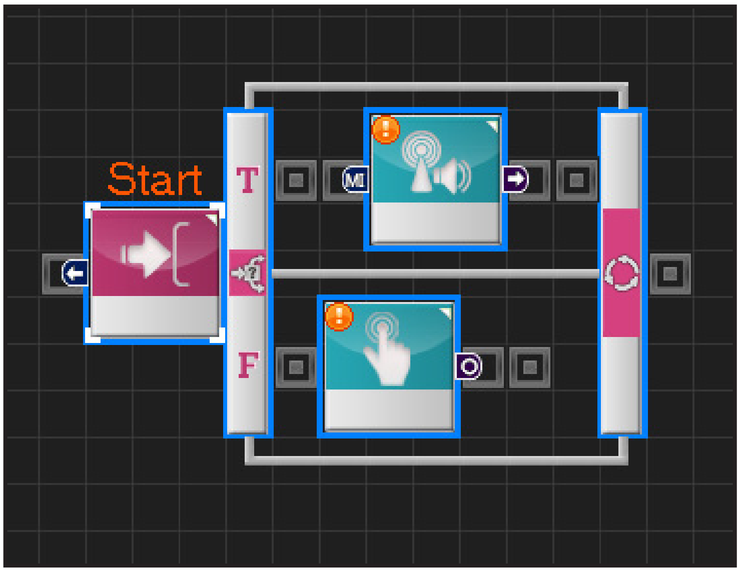
Loop

Loop module commands repeat of certain section. Loop with For statement would repeat certain number of times whereas Loop with Forever statement would repeat infinite times.



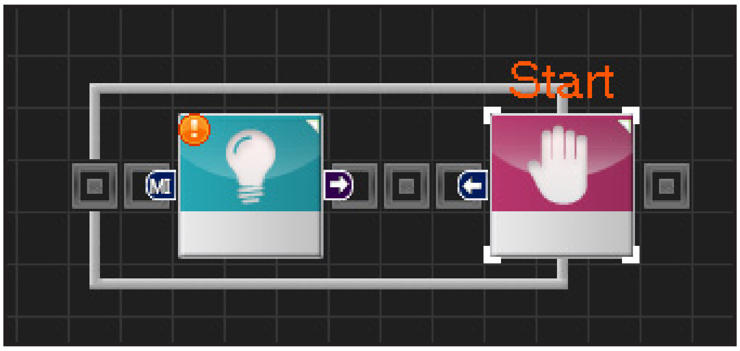
While

While module requires certain condition to be met before proceeding to the next step. It is a loop stateement with attached condition.



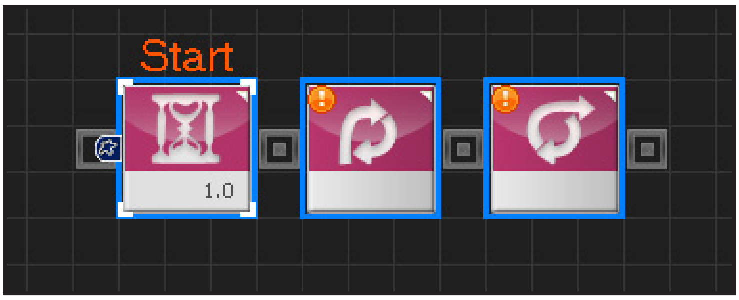
Switch

Switch module is similar to if–else statement. If the condition is True, it will perform the top task and if the condition is false it will perform the bottom task.



Wait

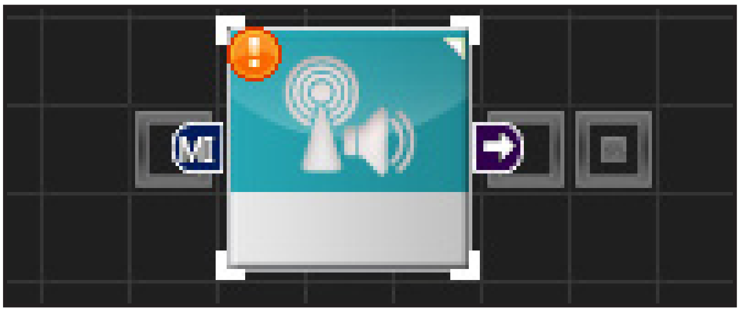
If input condition is True, halt program execution and wait. Program execution will continue if the the condition becomes False.



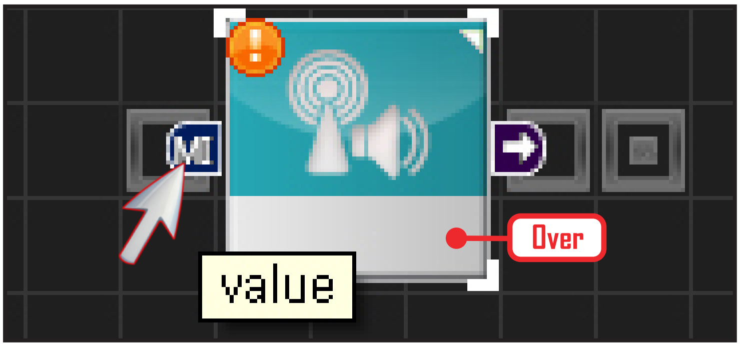
Delay, Continue, Break

Three modules in the left are arranged like regular modules without the graphic outline.
Delay module delays the program for certain period. Continue module sends the execution back to the beginning of the loop. Break module exits the program from the loop.

Some modules have Input and Output values. Resulting value of the output connector becomes the input value of the next connected module. Modules with both Input/Output values will have connectors on both left and right side of the module. Refer to below for example of connectors.

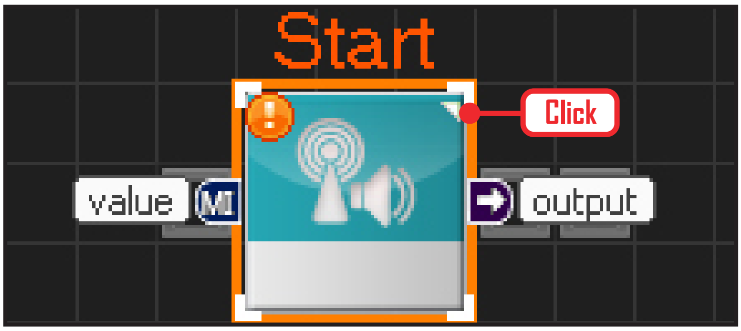


Connector



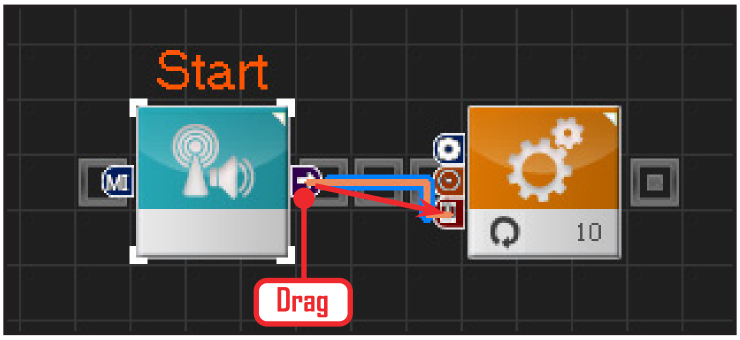
Help Balloon

It is difficult to distinquish the connector just by looking at the connector icon. To find out the function of the connector, place the mouse cursor on top of the connector and balloon will appear with the name of the connector.



Opening Help Balloons

To view the name of several connectors all at once, click on the triangle icon at top left corner of the module and connector names will appear beside each connector. Click one more time to close the balloons.

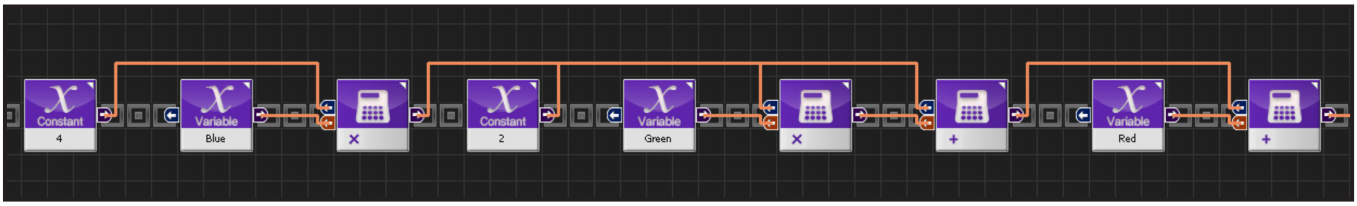


Connecting Connectors

To input the output value of the front module into the input value of the following module, use the mouse to drag and connect. Connecting line will appear as shown in the left photo.

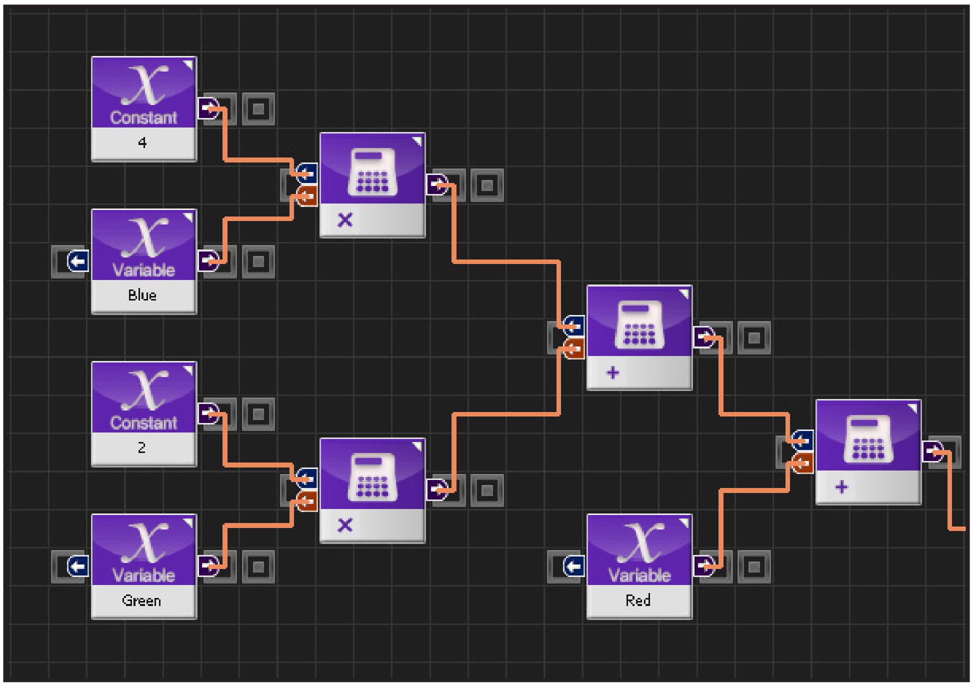
Connection Type

Module connections can be either serial type connection or row type connection.



Serial Type Connection

In serial type connection, modules are connected sequentially from left to right. The photo above shows arithmetic calculation program. ((4xBlue)+(2xGreen))+(1xRed) calcuation shown as serial type connection.

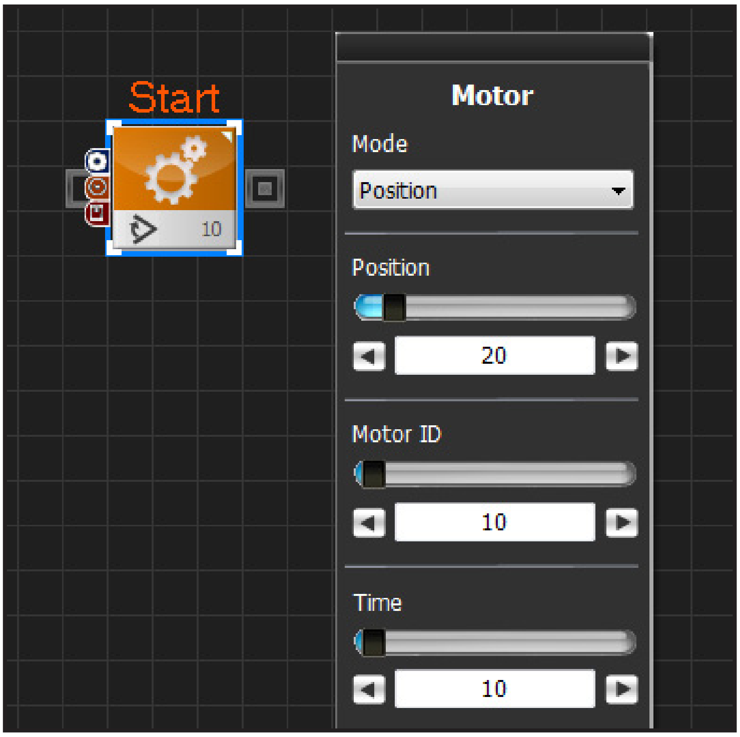


Row Type Connection

Row type connection, modules are connected in rows using vertical spacing. The 2nd photo with row type connection is same program as the 1st photo with serial type connection.

Property WIndow

Modules have their own properties and thses propeties must be given a value for program to work. UI in property window includes list popup, radio button, number setting, and etc. Refer to the Help file for details on properties for each module, property values, and limits.



Motor

Mode: Position

Position: 20

Motor ID: 10

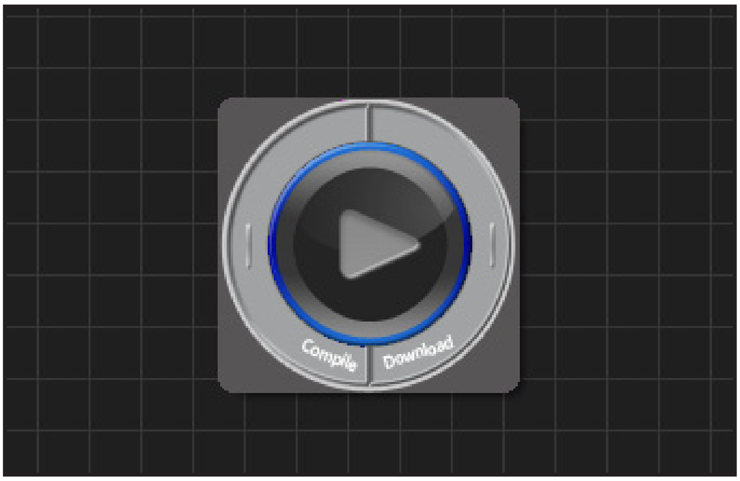
Time: 10

Property Window

When motor module is clicked, property window shows up on right side of the window. Motor has speed and position control properties. To control the position, select 'Position' in Mode selector. To control speed, select 'Velocity'. Position, Motor ID, Time values are adjusted in the detailed settings below the Mode Selector.

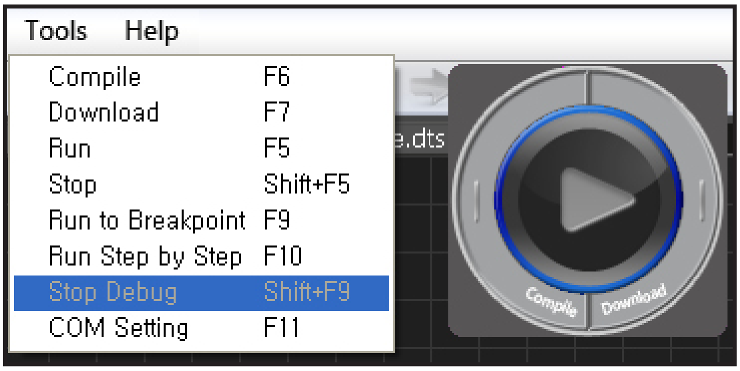
Compile/Download

Once the programming is complete, it is compiled, downloaded to the robot and run. Downloader is a large icon located at bottom left side of the programming window. More specific commands are found in the tools menu.



Downloader Icon

Downloader icon has three commands. Compile command on the left, download command on the right, and play command in the middle shown by arrow like icon.



Tools Menu

Tools menu contains more specific related commands.

Compile : Comile edited task..
Download : Download compiled task file.
Run : Run downloaded Task file.
Stop : Stops running the program.
Run to breakpoint : Program will run to designated breakpoint and stop.
Stop Debug: Stops debugging process
Run in single steps : Runs program by module.

Programming Individual Modules

Provided sample program is based on 16 axis humanoid robot with DRC controller platform. Sample program will require reprogramming if it is to be used for 18,20 axis humanoid robot or other variations with change in modules or motions. Before running the program, check the motor ID and robot sensor locations. Also, use the DR-SIM to check the saved motion list and apply correct index values. Provided sample program is as follows.

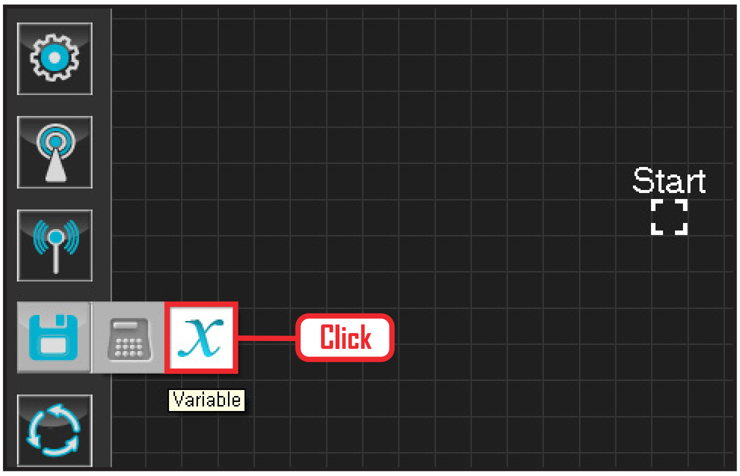
Module Pack	Module	Example
Motion	Move	Move module loads the robot motion saved in the DRC controller and applies it to the program. Robot motion can be loaded by the number, and if required, names can be checked from DR-SIM. This program will repeat running the motion creatd by DR-SIM on the robot indefinitely. This is a relatively complicated program useful for reliability test or for demostration purposes.
	Motor	This program creates dancing motion by controlling individual motors.
	LED	This program will turn on/off the LED by pressing the button on DRC Controller.
	Sound	This progam will output sound when input from remote control buttons(#1~8) is received.
Sensor	Sound Sensor	Sound sensors are located inside the DRC on both sides. This program will make the robot respond to the left clap by lifting the left arm and to the right clap by lifting the right arm (Sample # 2). Robot may have difficulty distinquishing the direction of the clap when there is lots of background noise. It may respond by lifting both arms to a single clap from one direction or respond erratically. More refined programming is required to make the robot to respond more reliably regardless of the background noise. Refining the program by forcing a DELAY after registering the first sound so that it will not receive anymore sound input will increase the reliability.
	Light	This program makes the motor respond to the external luminosity. When luminosity decreases, robot will lift up the left arm (Covering the CDS sensor at back of the controller will decrease the luminosity and robot will respond by lifting the left arm).
	Distance	PSD Digital(Distance Sensor) : This program makes the robot walk backwards, turn right, and then walk straight if it detecs a wall within certain distance. PSD Analog(Distance Sensor) : This progam makes the robot turn left to avoid the wall.
	Dynamics	Accerlateration : This program makes the robot stand if it falls forward, makes the robot stand if it falls backward.
Communi cation	IRRciever	This program assigns different sound notes to the remote control buttons 1~8 and makes the DRC controller play the sound.Buttons1~8 matches Do~Si. (With Sound)
	Button	LED at back of the DRC respond to the press of a button on DRC (with LED)

Move Example Step by Step

Example Description

Move module loads the robot motion saved in the DRC controller and applies it to the program. Robot motion can be loaded by the number, and if required, names can be checked from DR-SIM. This program will repeat running the motion creatd by DR-SIM on the robot indefinitely. This is a relatively complicated program useful for reliability test or for demostration purposes.

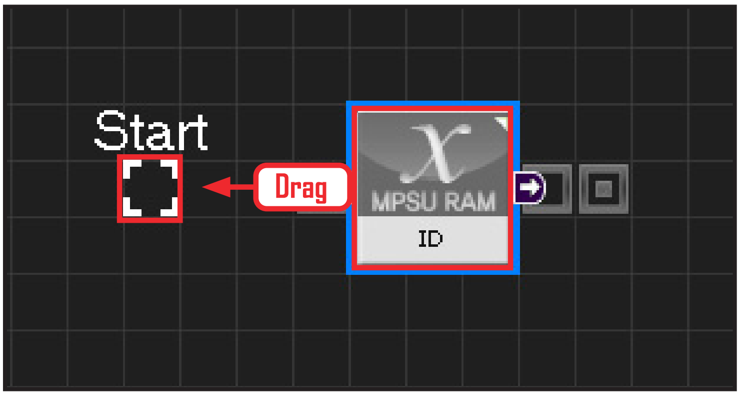
* Motions and Motion numbers used in this example are not same as the provided basic motion. This example assumes motion was created by DR-SIM and downloaded to the DRC. To download motion go to www.hovis.co.kr/guide



01 Assign Variable

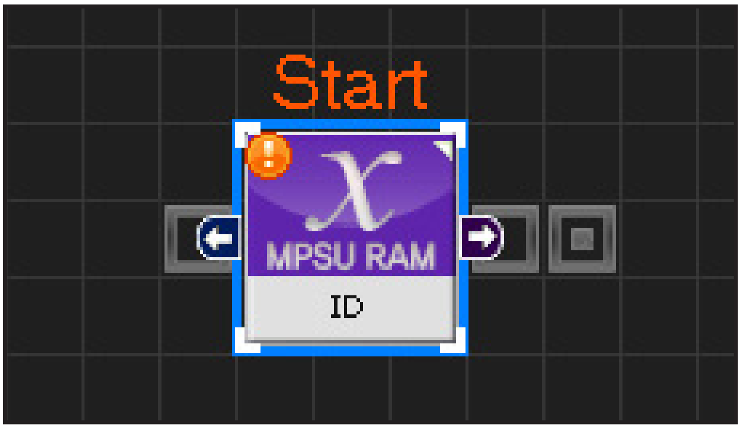
Operating the robot is same as operating the ro-bot servo motor. Value has to be assigned so that servo will be able to operate.

Click Data 〉 Variable module.



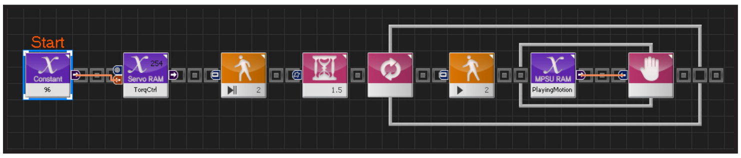
02 Start

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



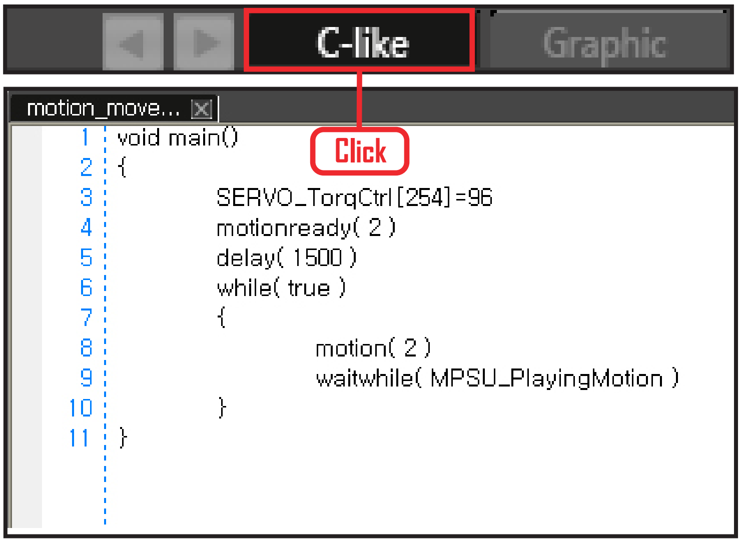
03 Start Programming

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

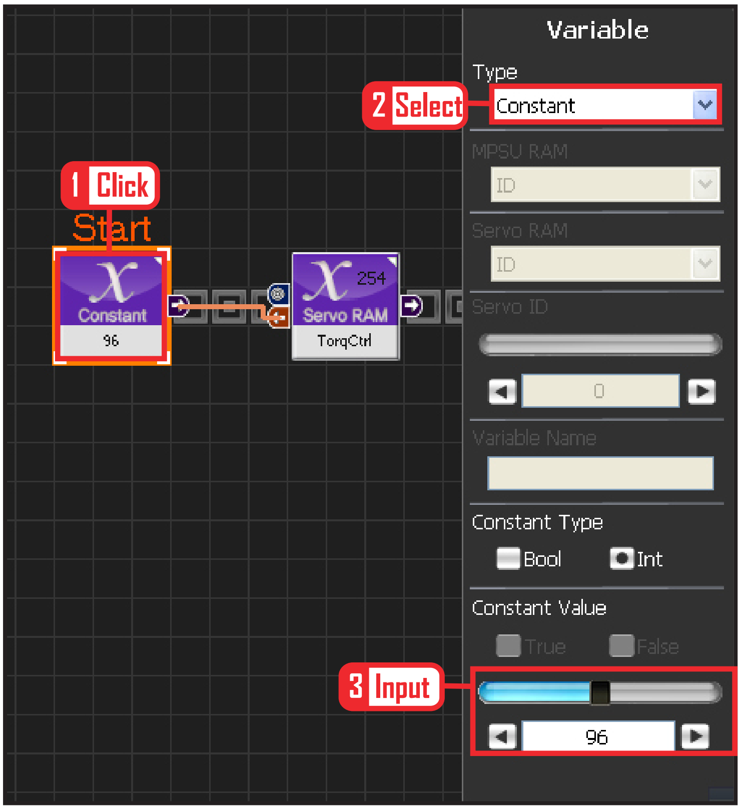
Loads the saved motion and duplicates 저장된 모션을 가져와 일정하게 반복 시키는 프로그래밍입니다. Motion ready 값에 주의합니다.



```
void main()
{
        SERVO_TorqCtrl[254]=96
        motionready( 2 )
        delay( 1500 )
        while( true )
        {
                motion( 2 )
                waitwhile( MPSU_PlayingMotion )
        }
}
```

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.



Constant 96

Type: Constant

MPSU RAM: ID

Servo RAM: ID

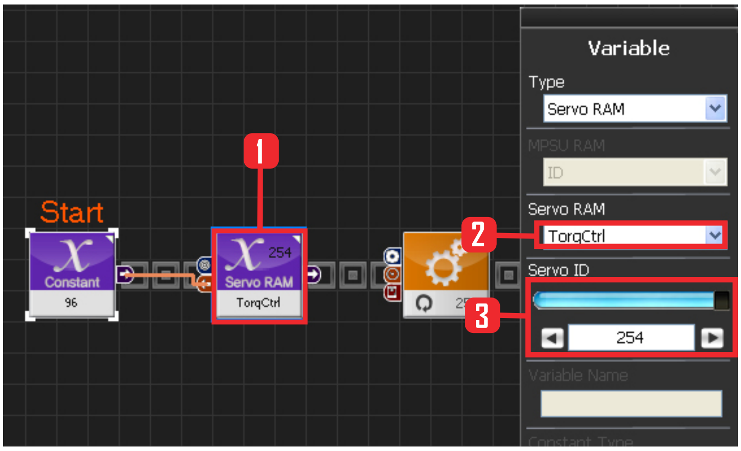
Servo ID: 0

Constant Type: ○ Bool ● Int

Constant Value: 96

06 Variable Setup

This section makes the servo motor to operate on it's own. Select Constant as the Variable Type. In properties, set constant value as 96.
When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

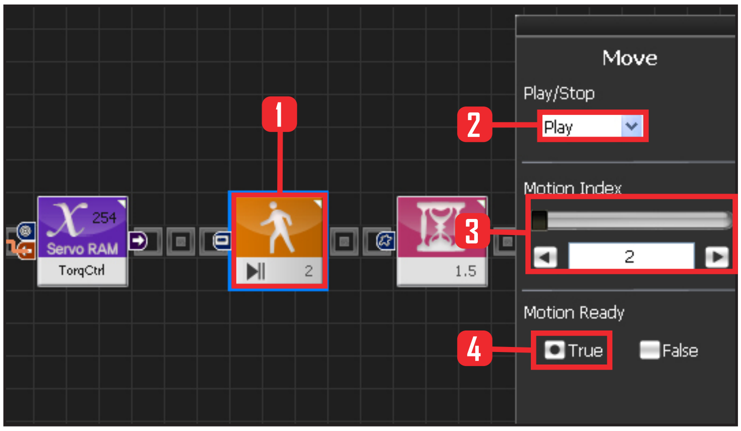
Variable Name

Constant Type

07 Apply to All Servos

This section applies contact value 96 to all ser-vos.

Select Variable 〉Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos..



Move

Play/Stop: Play

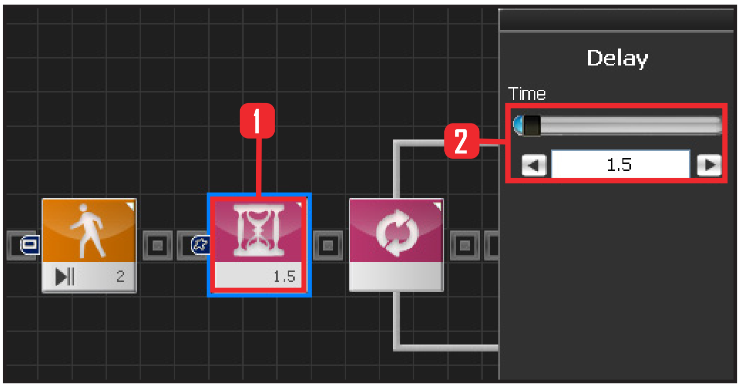
Motion Index: 2

Motion Ready: ● True ○ False

08 Motion Ready

When the motion is loaded, robot may make a sudden movement or motion change from the current position. If the difference between the current position and the start of the motion is drastically different, it may cause stress to the motors or pose danger to the user. To prevent such an occurence 'Motion Ready' is used to give time for motion to start.

Select Motion 〉Move .
Select Play/Stop : Play .
Select Motion Index : 2. Load motion No 2. As a reference, motion No 2 in this progam makes the r obot sit and stand. It does not necessarily have to be No 2. User can select another motion No to use.
Select Motion Ready : True . When True is selected, robot will slowly change to starting position of the moiton.



Delay

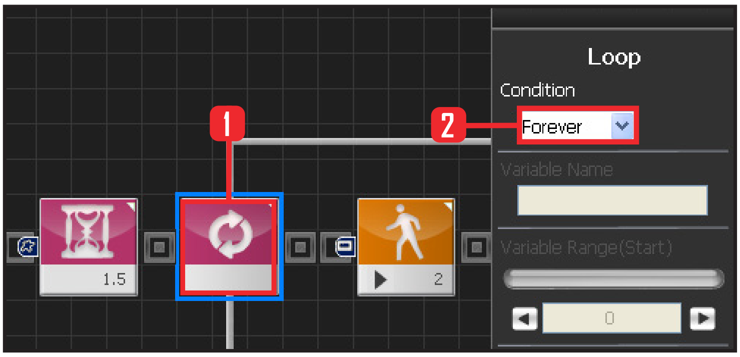
Time: 1.5

09 Delay

To prevent the motion from starting before Motion Ready ends, set Delay value to 1.5s.

10 Loop

Set Forever infinite loop.



Loop

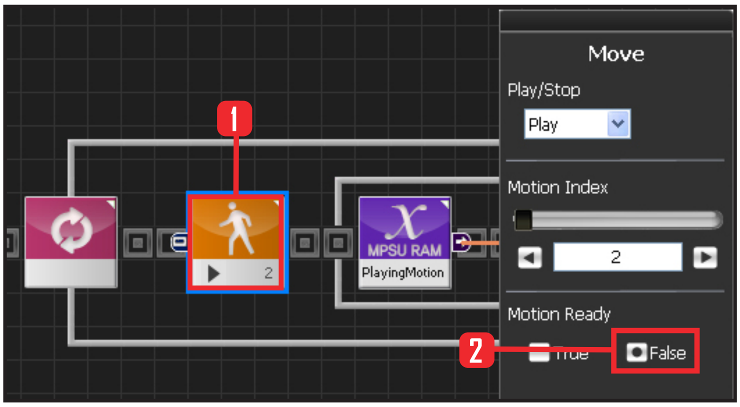
Condition: Forever

Variable Name

Variable Range(Start): 0

11 Motion Movement

If Motion Ready value is set to False, motion will run from start to finish.
Select Motion Ready : False .



Move

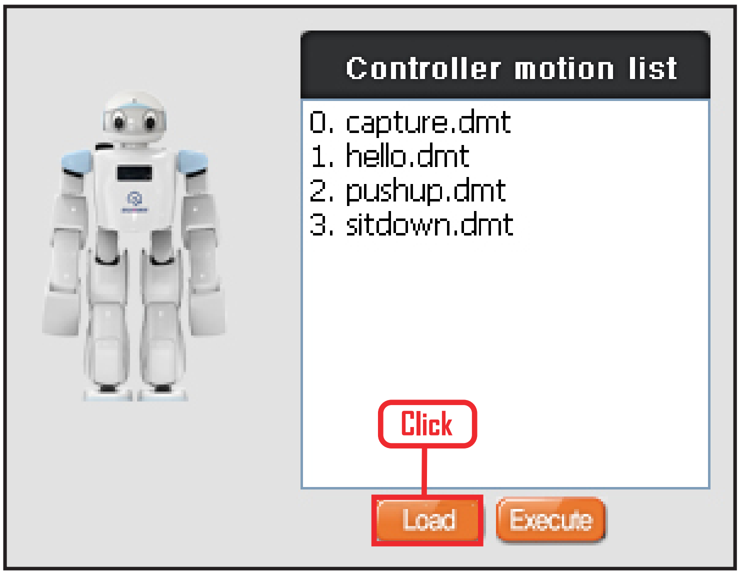
Play/Stop: Play

Motion Index: 2

Motion Ready: ○ True ● False

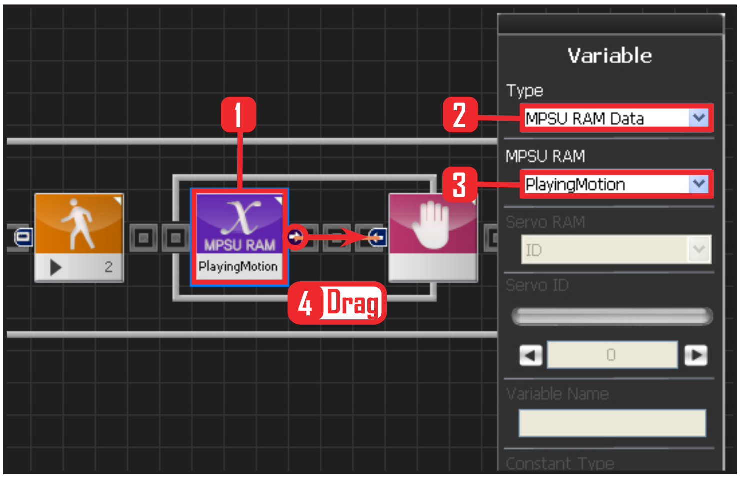
Reference: View Motion

To view the list of motions in the controller, connect to the robot and click robot setting from DR-SIM.
No 2 motion Robot sits and stands.



Controller motion list

0. capture.dmt
1. hello.dmt
2. pushup.dmt
3. sitdown.dmt

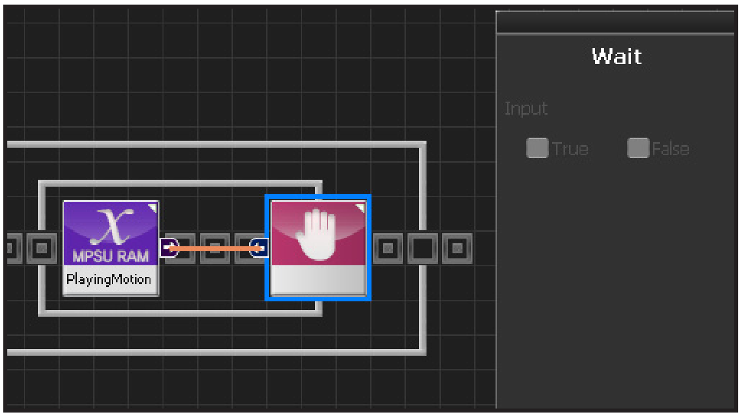


12 Check Motion Movement

Loop refers to continuous repetition. It takes time for the actual motion to complete after Move command has been issued, but loop with single move module will continue to run and give motion command even while the previous motion is still running. The lag in actual motion will result in difference between the number of motion commands given by the move module and the number of actual motions. To correct this difference, loop will need to wait for the motion to complete before repeating the process.'Playing Motion' is found within Variable 〉 MPSU RAM Data.

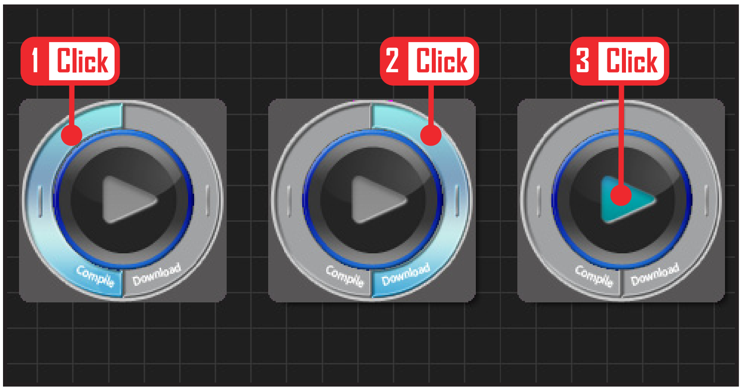
'Playing Motion' is a variable that checks whether the motion is in process. Loop will wait for the current motion to end if 'wait' is added to the 'Playing Motion'.

Select Data 〉 Variable Module.
Select Type : MPSU RAM Data
Select MPSU RAM : Playing Motion
Add Wait module to the output connector.



13 Wait

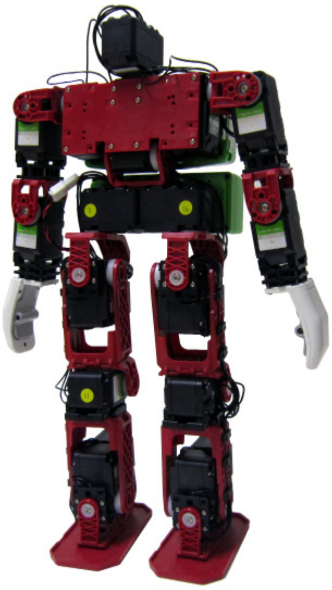
Wait untill the motion ends.
Go to the begining and repeat when motion ends.



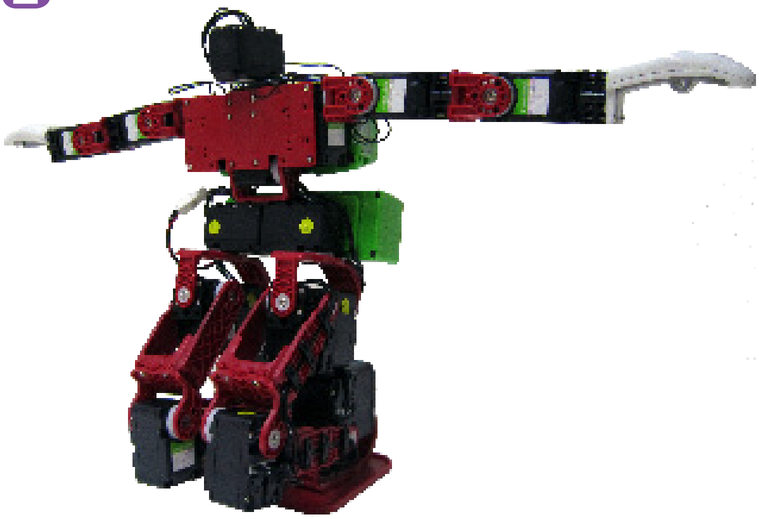
14 Compile, Download, Run

Click left icon to compile. If no compile error is found, click right icon and download to robot. After the downoad is complete, click the arrow like run button in the middle to apply the pro-gram to the robot.

1



2



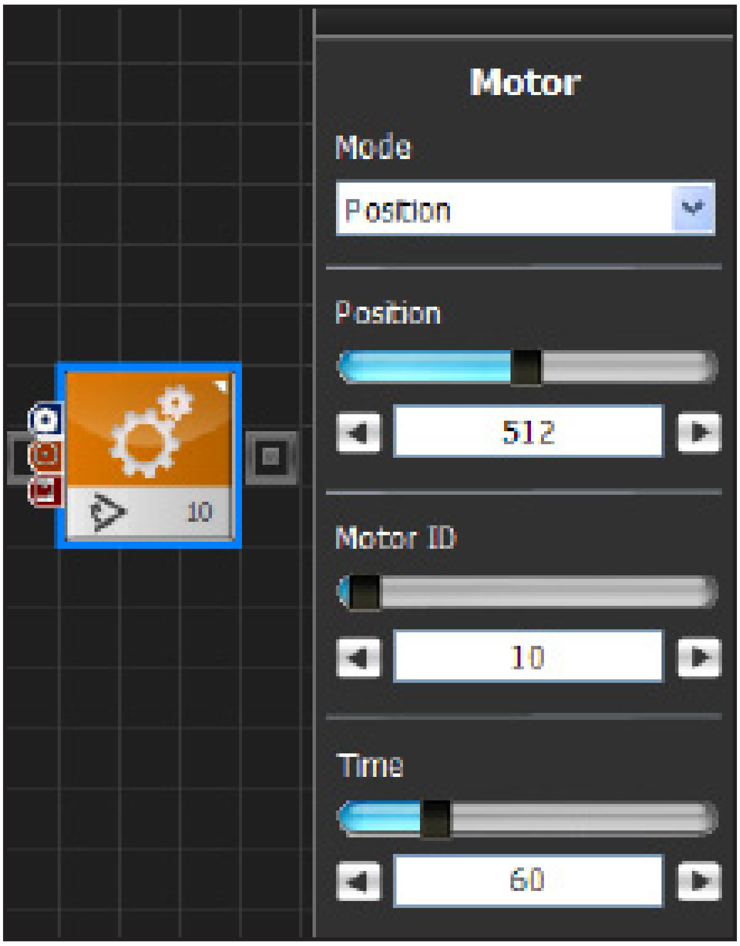
15 Robot Motion

Robot wiil continuously repeat sit and stand motion.

Programming Individual Module : Motion 〉 Motor

Motor Description

Motor module has two types of oerating modes. Positions control mode and Speed control model.



Motor

Mode

Position

Position

512

Motor ID

10

Time

60

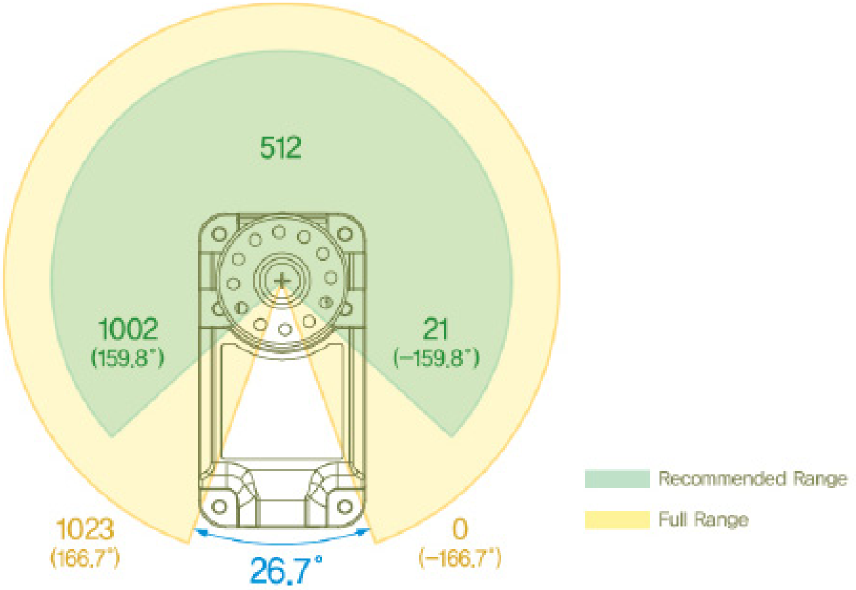
1 Position Control Mode

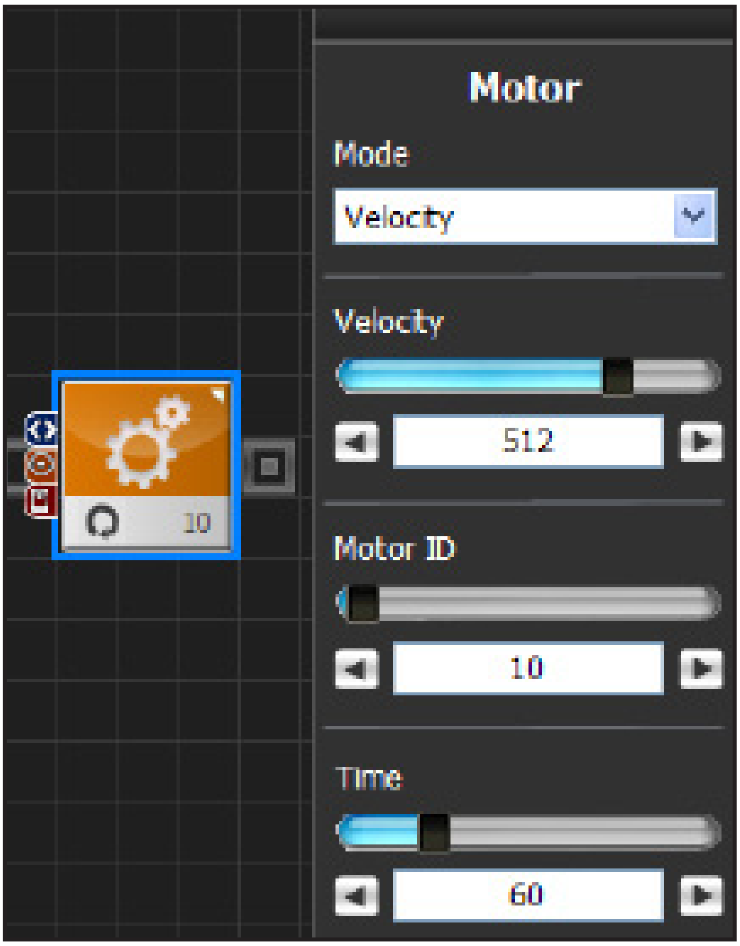
Position control mode changes the position of the selected motor to desired position.

Position has value range between -127~1152. Servos are released from the factory with default value range of 21~~1002. Values beyond the default range is possible with adjustment to min/max motor values and position adjustment values. Motor has regular position value of 512 which is used as a standard position value when assembling. When all Hovis motors have position value of 512, Hovis will be in standing position with both arms stretched out 90 degrees to the side. Refer to the diagram below to view position range and regular position.

Motor ID is the ID of the servo to be controlled.

Time refers to the time it takes for servo to reache the goal position. 1tick = 11.2ms. 100 tick would take the servo 1.12s to reach the goal position.





Motor

Mode: Velocity

Velocity: 512

Motor ID: 10

Time: 60

2 Speed(Velocity) Control Mode

Speed control mode puts the selected servo in continous roation with specific speed.

Velocity has value range of −1023~1023. Larger the value, larger the output with increased rotation speed. Sign of the value determines the direction of the rotation.

Motor ID is the ID of the servo to be controlled.

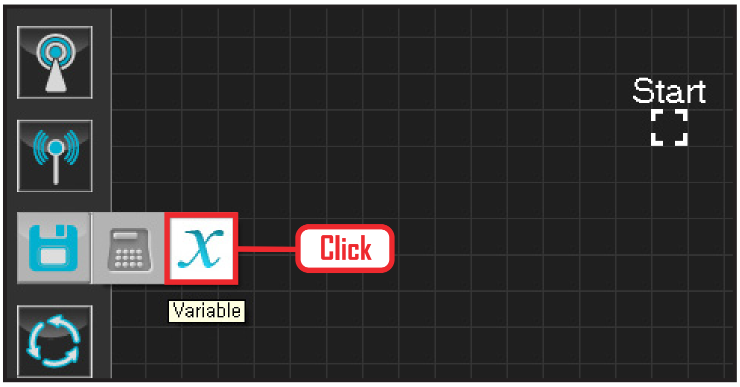
Time refers to the time it takes for servo to reache the goal position. 1tick = 11.2ms. When set to 100 tick, servo would take 1.12s to gradually reach the goal speed.

Example Step by Step

Example Description

Robot motions are usually made by controlling each individual servos and cosolidating their response. But, controlling each servo is a compliecated procedure which is why tools such as the DR-SIM (Motion Editior) is usually used.

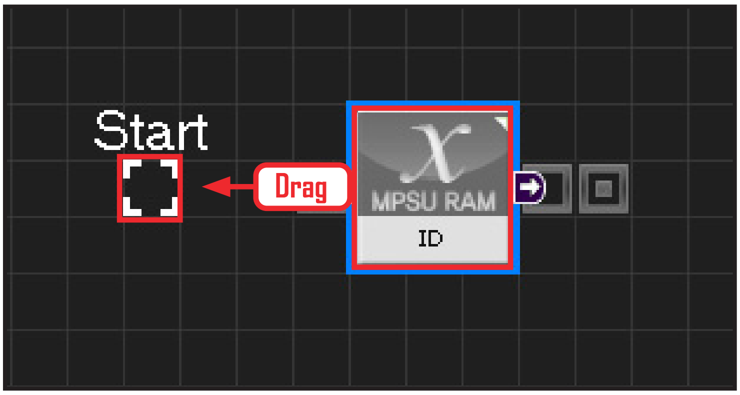
Instead of using the Motion Editor, this example will use the Task Editor to control each individual servos to produce a continuous motion. The end result of the program will be a very interesting wave dancing robot.



01 Assign Variable

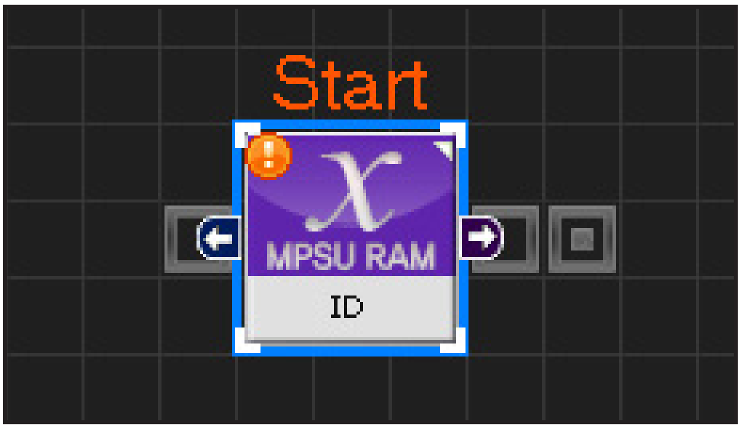
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

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.

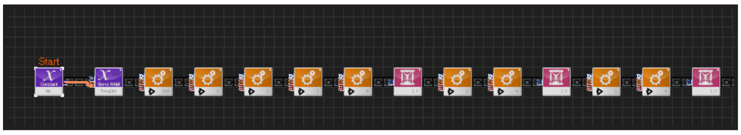


03 Start Programming

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.

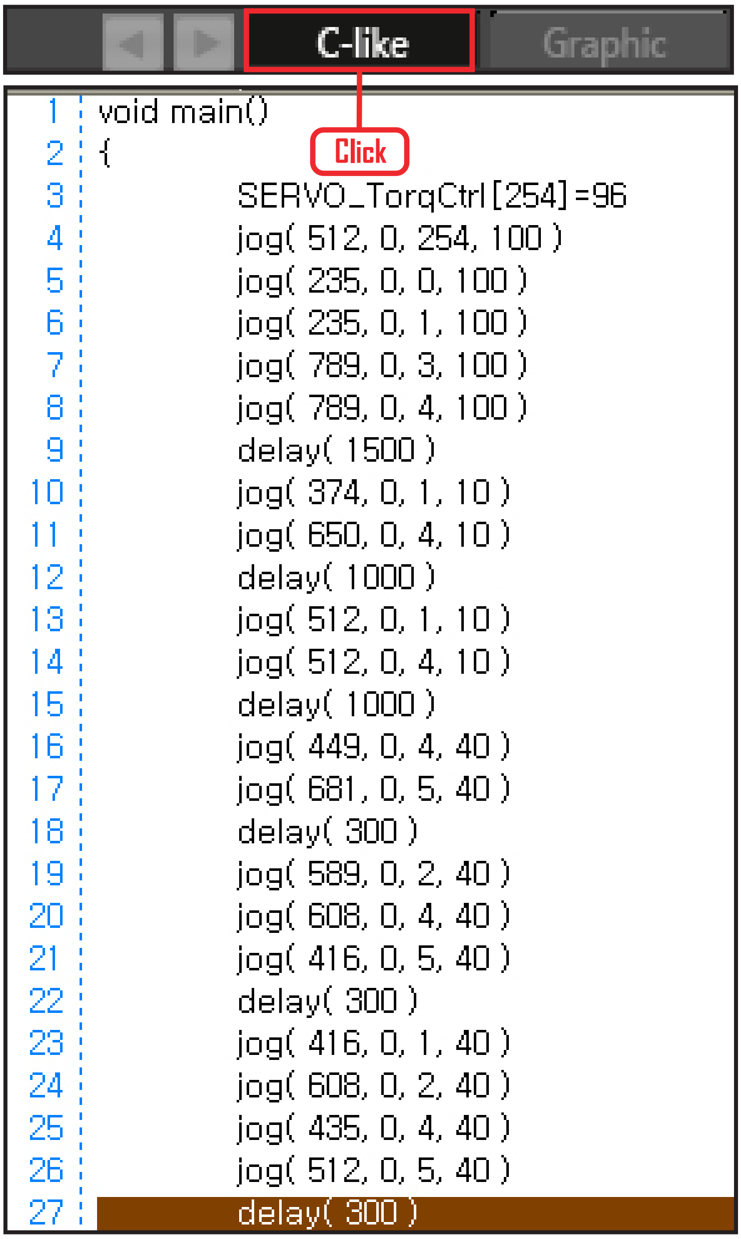
04 Entire Program

Entire program controlling the motors.

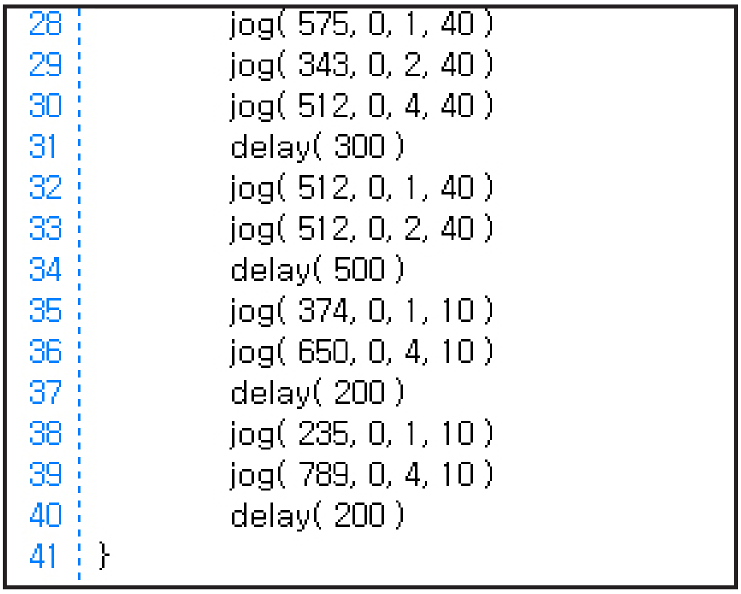


05 Viewing C-Like

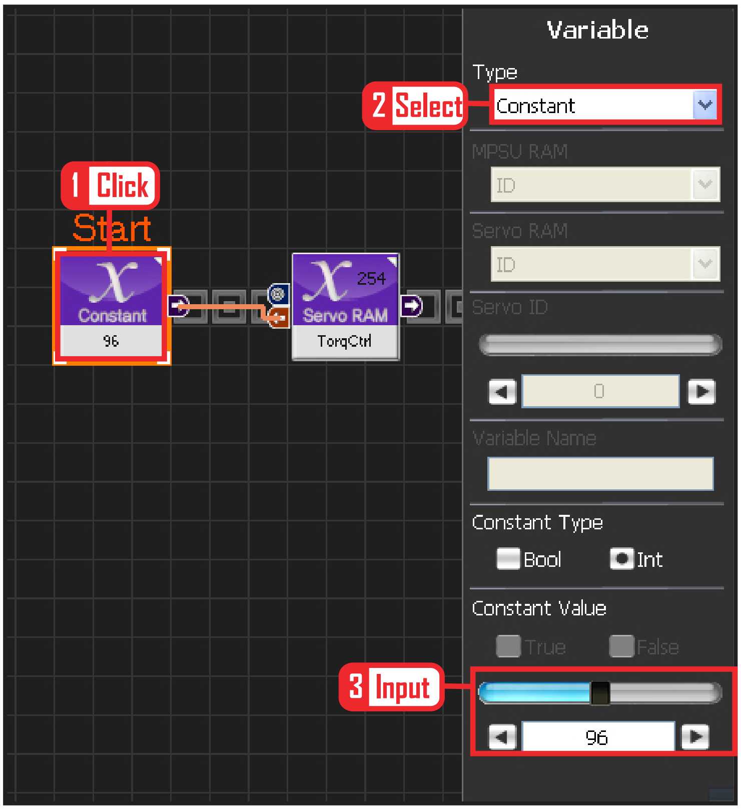
Click the 'C–like' tab near the top right and task programming window will open as shown in the photo to the left. This is the task window of the entire program. Codes are very similar to the C language structure so studying the codes will help the user become familiar with the C language structure.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



```
void main()
{
          SERVO_TorqCtrl[254]=96
          jog( 512, 0, 254, 100 )
          jog( 235, 0, 0, 100 )
          jog( 235, 0, 1, 100 )
          jog( 789, 0, 3, 100 )
          jog( 789, 0, 4, 100 )
          delay( 1500 )
          jog( 374, 0, 1, 10 )
          jog( 650, 0, 4, 10 )
          delay( 1000 )
          jog( 512, 0, 1, 10 )
          jog( 512, 0, 4, 10 )
          delay( 1000 )
          jog( 449, 0, 4, 40 )
          jog( 681, 0, 5, 40 )
          delay( 300 )
          jog( 589, 0, 2, 40 )
          jog( 608, 0, 4, 40 )
          jog( 416, 0, 5, 40 )
          delay( 300 )
          jog( 416, 0, 1, 40 )
          jog( 608, 0, 2, 40 )
          jog( 435, 0, 4, 40 )
          jog( 512, 0, 5, 40 )
          delay( 300 )
```



```
            jog( 575, 0, 1, 40 )
            jog( 343, 0, 2, 40 )
            jog( 512, 0, 4, 40 )
            delay( 300 )
            jog( 512, 0, 1, 40 )
            jog( 512, 0, 2, 40 )
            delay( 500 )
            jog( 374, 0, 1, 10 )
            jog( 650, 0, 4, 10 )
            delay( 200 )
            jog( 235, 0, 1, 10 )
            jog( 789, 0, 4, 10 )
            delay( 200 )
}
```



1 Click

2 Select

3 Input

Variable

Type: Constant

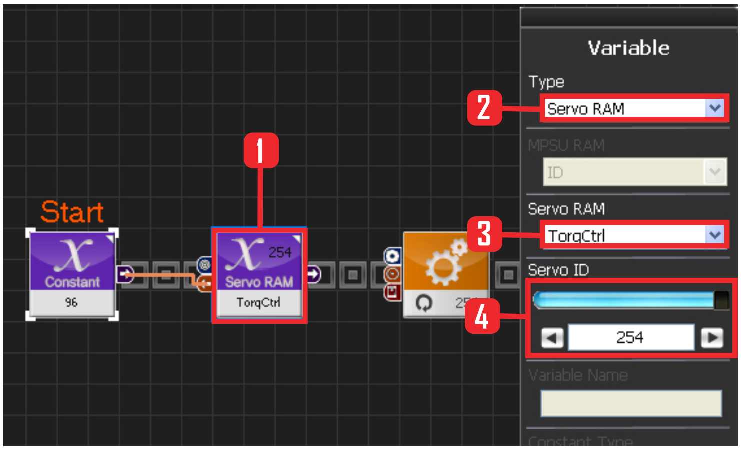
Constant Type: ☐ Bool ● Int

Constant Value: ☐ True ☐ False

96

06 Variable Setup

This section makes the servo motor to operate on it's own. Select Constant as the Variable Type. In properties, set constant value as 96. When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

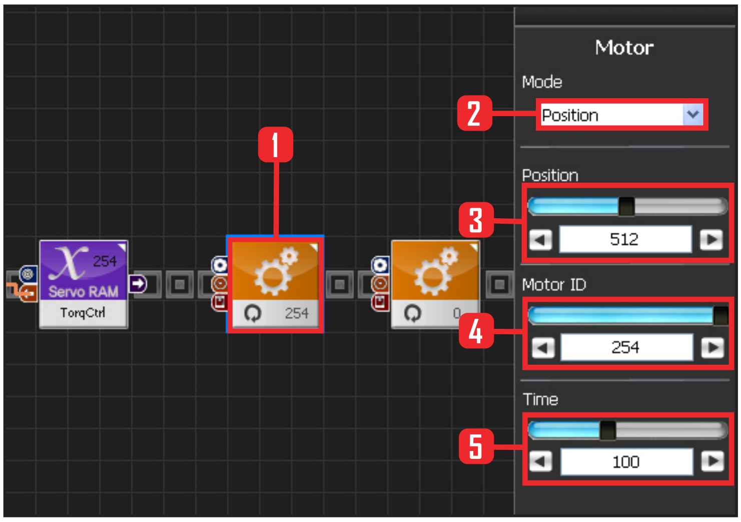
Variable Name

Constant Type

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉 Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos..



Motor

Mode: Position

Position: 512

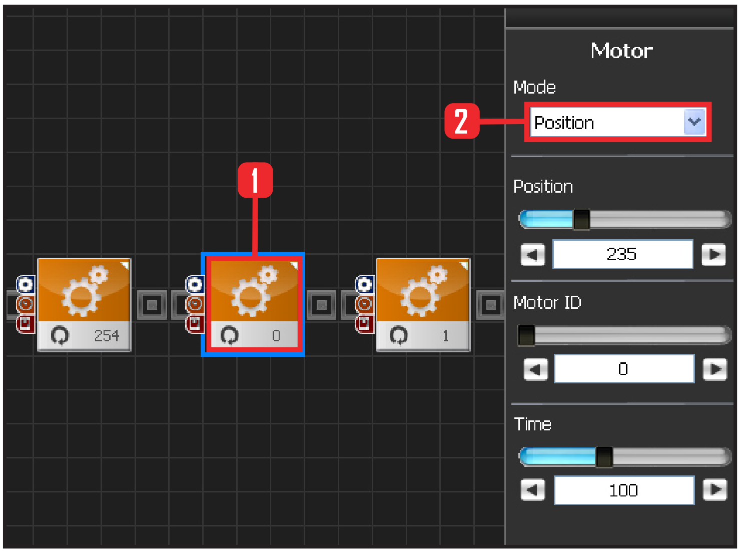
Motor ID: 254

Time: 100

08 Set Angle to All Servos

Set all servo motor angles to the center.

Select Motion 〉 Moter .
Select Mode : Positon . Set angle.
Set Position : 512 . 512 sets servo angle to the center
Set Motor ID : 254 . apply to all servos
Set Time : 100 . 1tick = 11.2ms, 100 tick = 1.12s.
Move to set angle for1.12s.



Motor

Mode: Position

Position: 235

Motor ID: 0

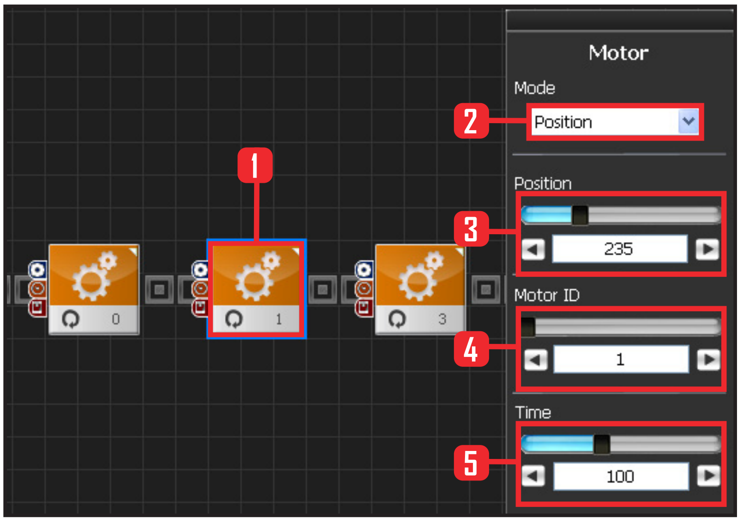
Time: 100

09 Motor ID 0 (Right Shoulder) Setup

1st stage : Initial position

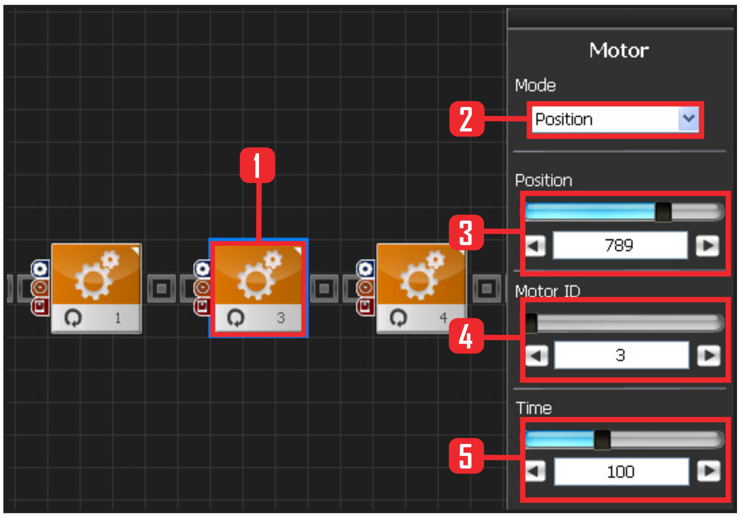
Make attention posture(Basic posture)
When all servo motors are aligned to the center, humanoid robot will be standing with both arms stretched out to the side. This stretched out arm posture need to be returend to the basic attention posture to make applying motion easier.

Select Motion 〉 Motor
Select Mode : Position
Set Position : 235 . 235 turns the the motor so that the right arm in horizontal position can be lowered to vertical position.
Set Motor ID : 0. Right shoulder motor has ID 0.
Set Time : 100. Motor will turn to set angle in 1.12s.



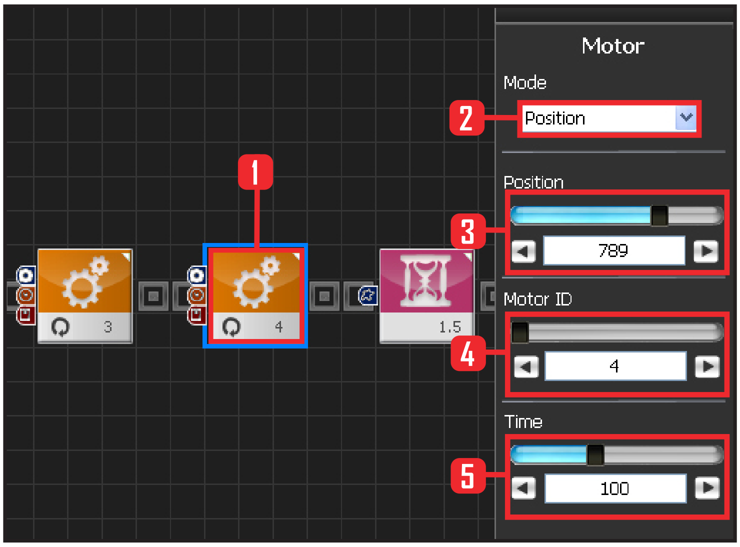
10 Motor ID 1 (Right Arm) Setup

Select Mode : Postion .
Set Position : 235 . 235 turn the horizontal arm to vertical position.
Set Motor ID : 1. Right upper arm motor connected to the shoulder has ID 1.
Set Time : 100 . Motor will turn to set angle in 1.12s.



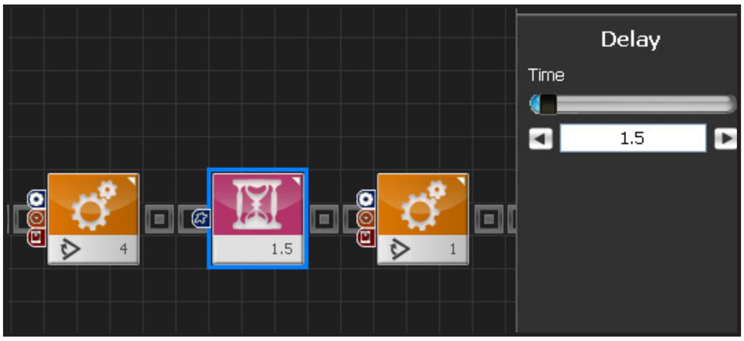
11 Motor ID 3(Left Shoulder) Setup

Select Mode : Position.
Set Position : 789. 789 turns the the motor so that the let arm in horizontal position can be lowered to vertical position.
Set Motor ID : 3. Left shoulder motor has ID 3.
Set Time : 100. Motor will turn to set angle in 1.12s.



12 Motor ID 4(Left Arm) Setup

Select Mode : Postion .
Set Position : 789 . 235 turn the horizontal arm to vertical position.
Set Motor ID : 4. Left upper arm motor connected to the shoulder has ID 4.
Set Time : 100 . Motor will turn to set angle in 1.12s.



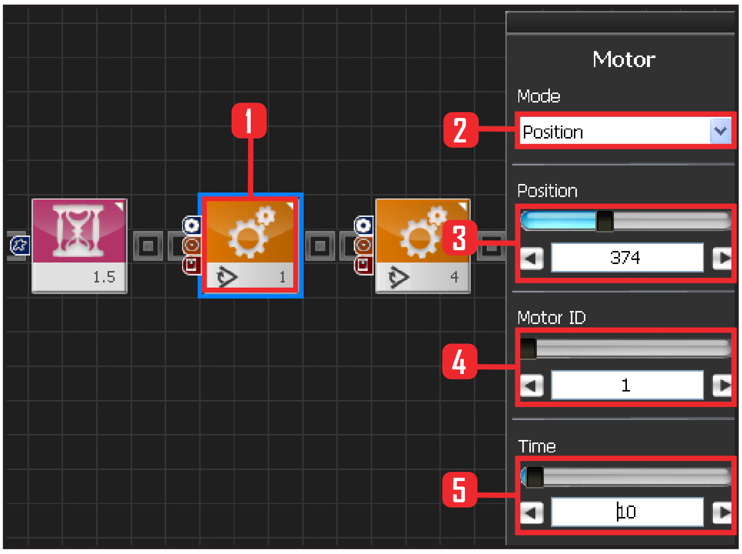
Delay

Time

1.5

13 Delay

Delay 1.5 s.



Motor

Mode

Position

Position

374

Motor ID

1

Time

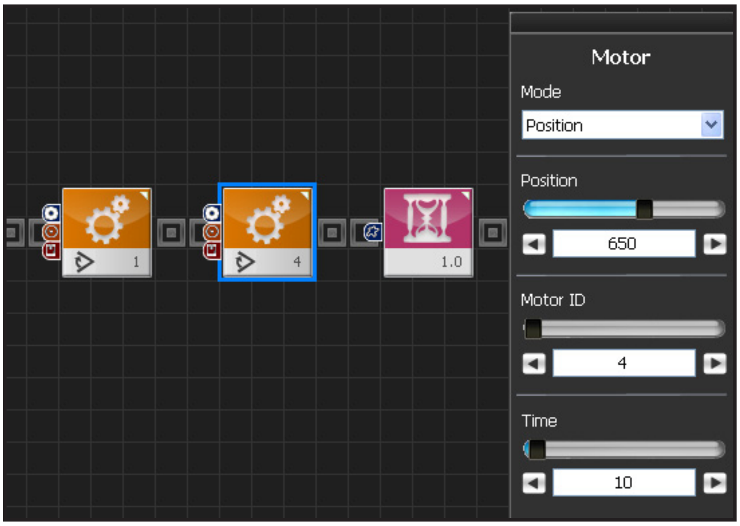
10

14 Motor ID1(Right Arm) Setup

2nd Stage : Set arm angle to 45 degrees.

Set arm angle to 45 degrees to prepare the robot for the dance.

Select Motion > Moter.
Select Mode : Position.
Set Position : 374 . 374 changes the right arm angle to 45 degrees.
Select Motor ID : 1. Right upper arm motor has ID 0.
Set Time : 10. Motor will turn to set angle in 0.112s.



Motor

Mode

Position

Position

650

Motor ID

4

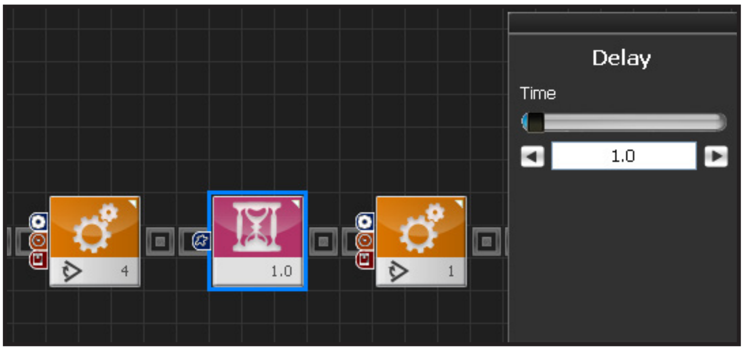
Time

10

15 Motor ID 4(Right Arm) Setup

Set left upper arm motor ID 4 to 45 degrees.

Select Motion > Moter.
Select Mode : Position.
Set Position : 650 . 650 changes the left arm angle to 45 degrees.
Select Motor ID : 4. left upper arm motor has ID 0.
Set Time : 10. Motor will turn to set angle in 0.112s.



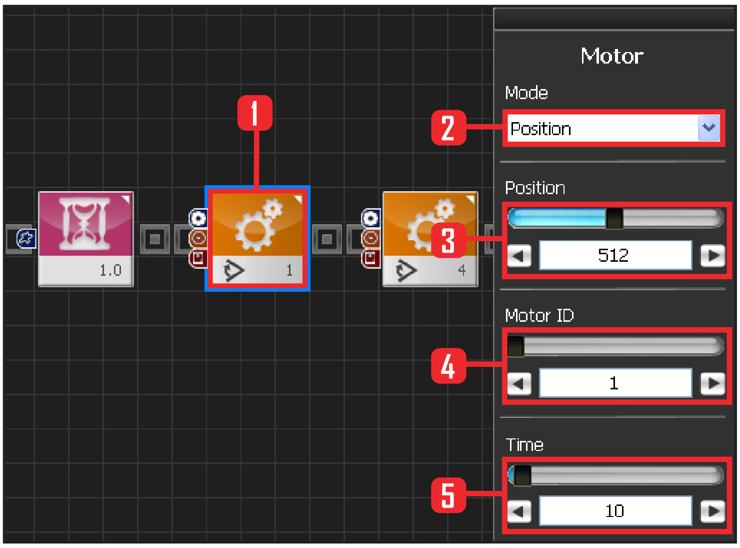
Delay

Time

1.0

16 Delay

Delay 1s.



Motor

Mode

Position

Position

512

Motor ID

1

Time

10

17 Motor ID 1(Right arm) Setup

3rd Stage : Set arm angle to 90 degreees.

Set arm angle to 90 degrees to start the robot on the wave dance.

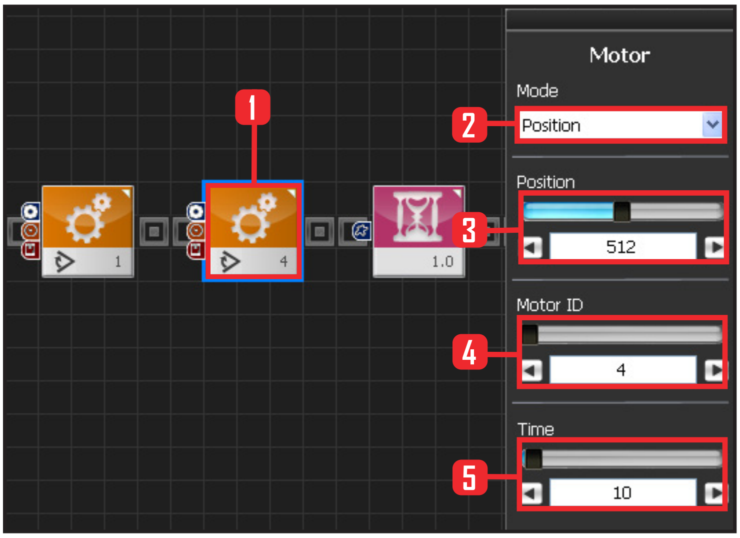
Setup Motion > Motor.

Select Mode : Position.

Set Position : 512. 512 50 changes the lright arm angle to 45 degrees. 512 is also the center position of the motor. When all motors are set to the center position, robot will stretch out both arms to the side.

Set Motor ID : 1. Right upper arm motor connected to the shoulder has ID 1

SetTime : 10. Motor will turn to set angle in 0.112s.



Motor

Mode

Position

Position

512

Motor ID

4

Time

10

18 Motor ID 4(Left Arm) Setup

Set arm angle to 90 degreees.

Set arm angle to 90 degrees to start the robot on the wave dance.

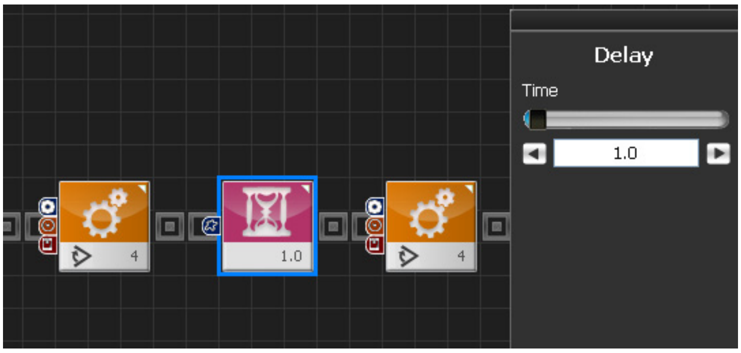
Setup Motion > Moter.

Select Mode : Position.

Set Position : 512. 512 50 changes the left arm angle to 45 degrees. 512 is also the center position of the motor. When all motors are set to the center position, robot will stretch out both arms to the side.

Set Motor ID : 1. Left upper arm motor connected to the shoulder has ID 4

SetTime : 10. Motor will turn to set angle in 0.112s.



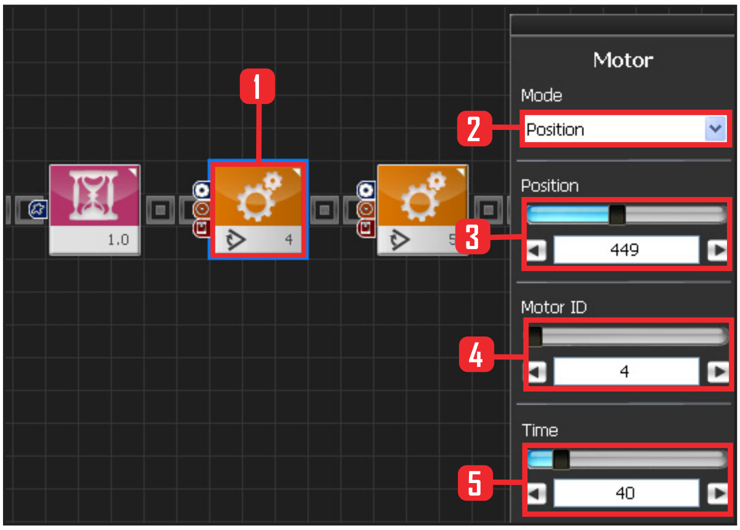
Delay

Time

1.0

19 Delay

Delay 1s.



Motor

Mode: Position

Position: 449

Motor ID: 4

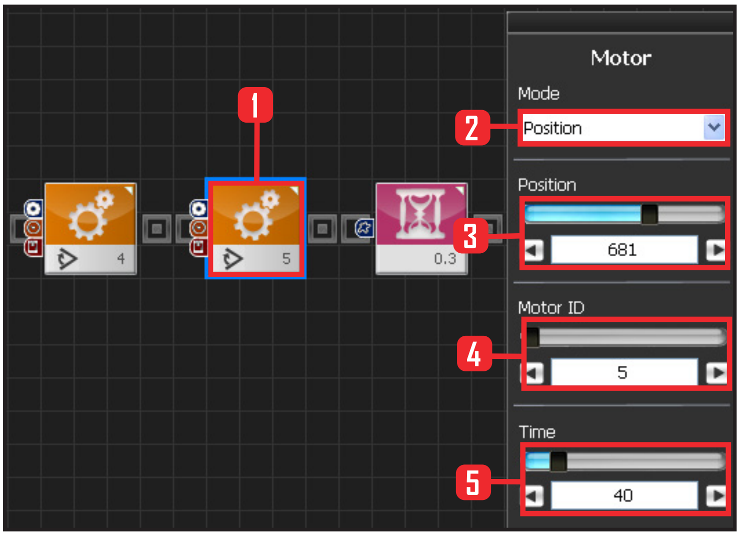
Time: 40

20 Motor ID 4(Left Arm) Setup

4th Stage : Wave 1 stage

Start the wave from the left arm.

Select Motion 〉 Moter.
Select Mode : Position.
Set Position : 449. 449 changes the left arm angle to the start of the wave dance.
Set Motor ID : Left upper arm motor connected to the shoulder has ID 4
Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode: Position

Position: 681

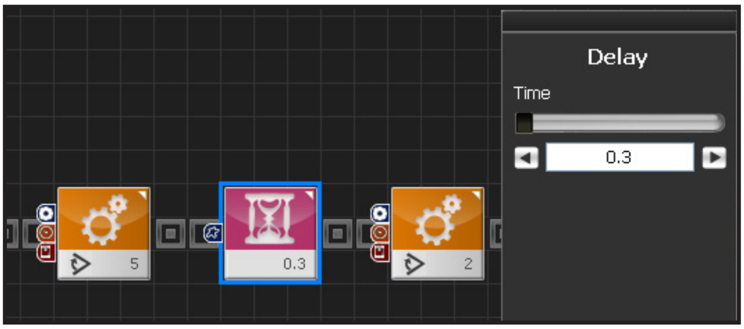
Motor ID: 5

Time: 40

21 Motor ID 5(Lower Left Arm) Setup

Lower left arm wave.

Select Motion 〉 Moter.
Select Mode : Position.
Set Position : 681.
Set Motor ID : 5. Lower left arm motor has ID 5.
Set Time : 40 . Motor will turn to set angle in 0.448s.



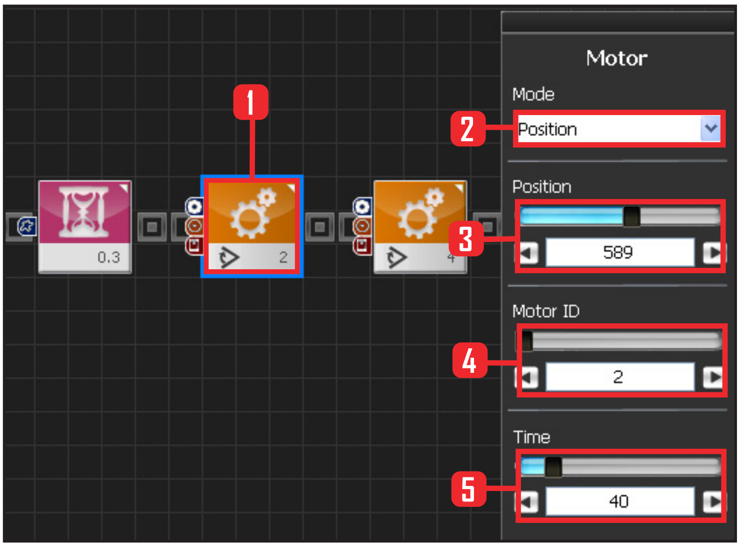
Delay

Time

0.3

22 Delay

Delay 0.3s.
Short delay as dance has started.



Motor

Mode: Position

Position: 589

Motor ID: 2

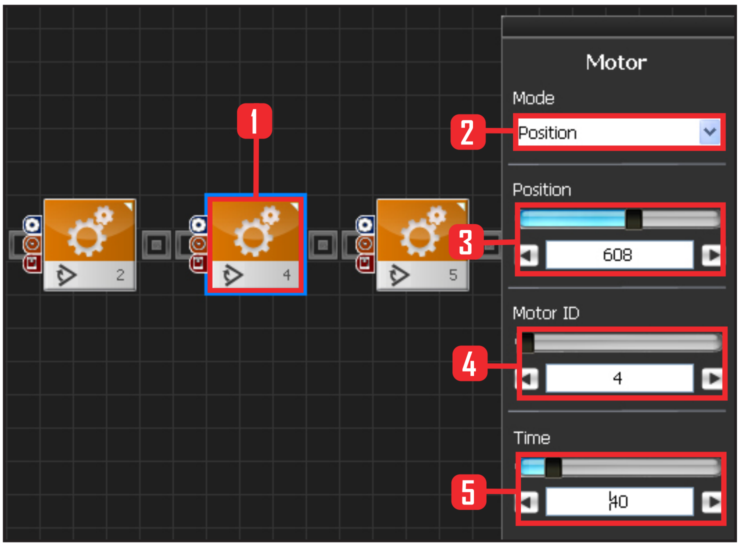
Time: 40

23 Motor ID 2(Lower Right Arm) Setup

5th Stage : Wave 2 Stage

Lower right arm wave.

Select Motion 〉 Motor.
Select Mode : Position .
Set Position : 589.
Set Motor ID : 2. Lower rigt arm motor has ID 2.
Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode: Position

Position: 608

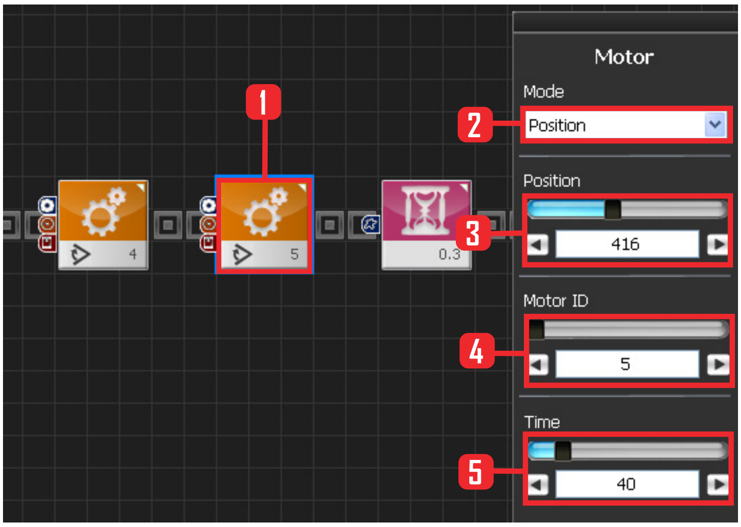
Motor ID: 4

Time: 40

24 Motor ID 4(Left Right Arm) Setup

Change the motor angle slightly to give wave effect.

Select Motion 〉 Motor.
Select Mode : Position .
Set Position : 608 .
Set Motor ID : 4 .
Set Time : 40 . Motor will turn to set angle in 0.448s.


Motor

Mode: Position

Position: 416

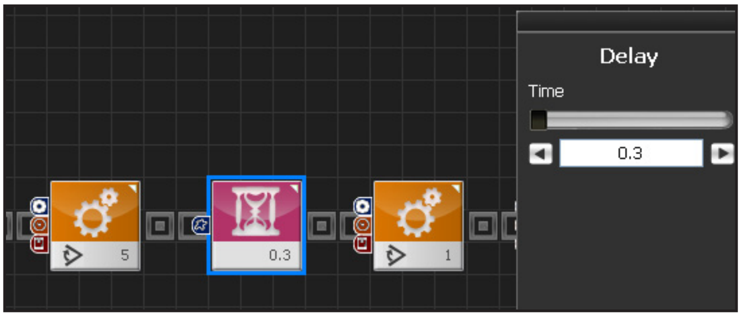
Motor ID: 5

Time: 40

25 Motor ID 5(Lower Left Arm) Setup

Change the motor angle slightly to give wave effect.

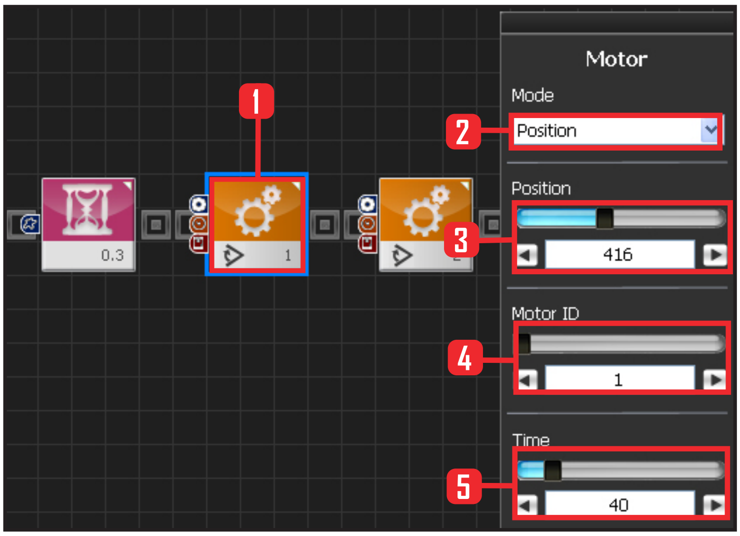
Select Motion > Motor.
Select Mode : Position .
Set Position : 416.
Set Motor ID : 5 .
Set Time : 40 . Motor will turn to set angle in 0.448s.


Delay

Time: 0.3

26 Delay

Delay 0.3s.
Short delay as dance has started.


Motor

Mode: Position

Position: 416

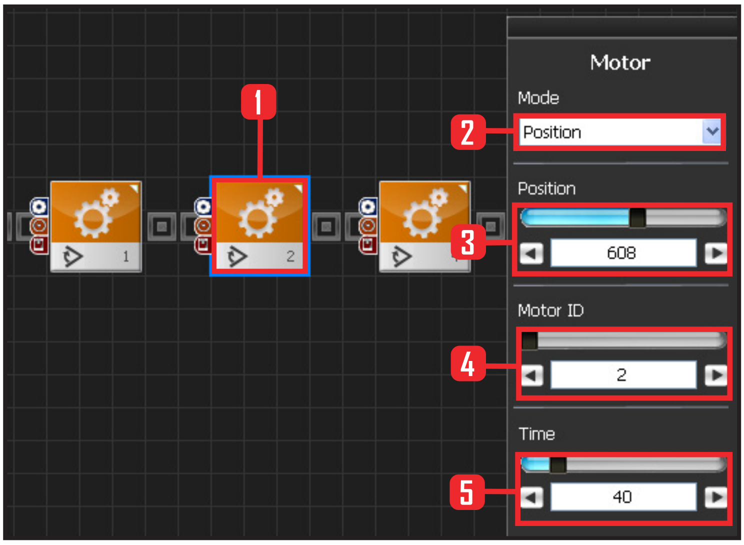
Motor ID: 1

Time: 40

27 Motor ID 1(Upper Right Arm) Setup

6th Stage : Wave 3 Stage
Return motor to original position.

Select Motion > Motor.
Select Mode : Position .
Set Position : 416.
Set Motor ID : 1 .
Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode: Position

Position: 608

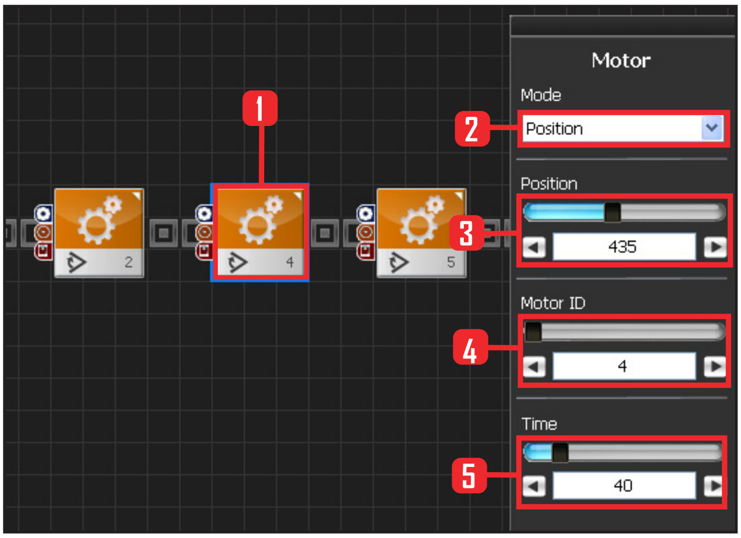
Motor ID: 2

Time: 40

28 Motor ID 2(Lower Right Arm) Setup

Return motor to original position.

Select Motion 〉Motor.
Select Mode : Position .
Set Position : 608.
Set Motor ID : 2.
Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode: Position

Position: 435

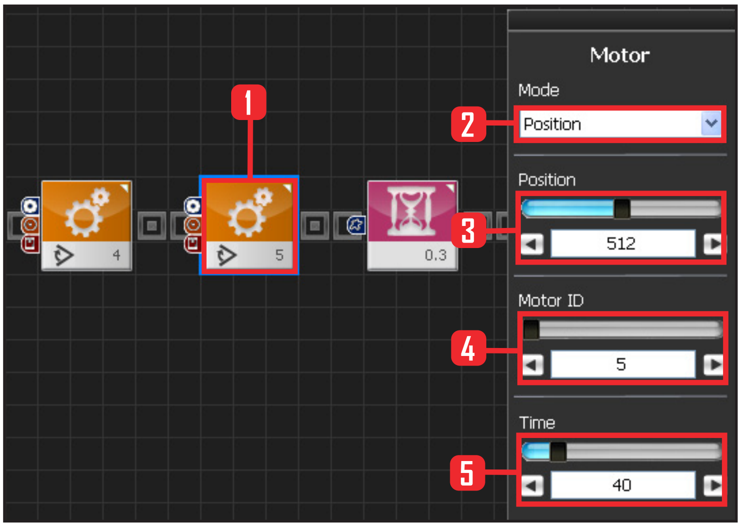
Motor ID: 4

Time: 40

29 Motor ID 4(Upper Left Arm) Setup

Return motor to original position.

Select Motion 〉Motor.
Select Mode : Position .
Set Position : 435.
Set Motor ID : 4.
Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode: Position

Position: 512

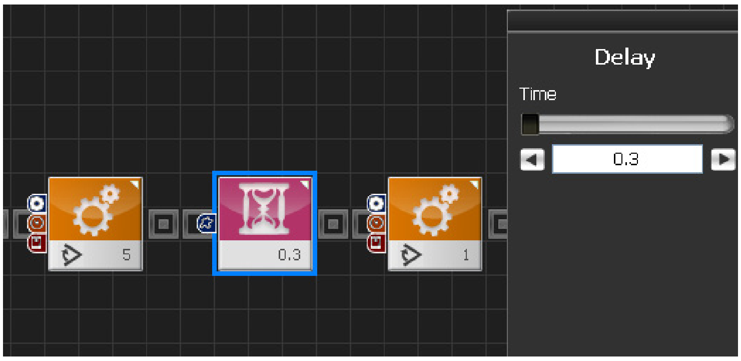
Motor ID: 5

Time: 40

30 Motor ID 5(Lower Left Arm) Setup

Return motor to original position.

Select Motion 〉Motor.
Select Mode : Position .
Set Position : 512.
Set Motor ID : 5.
Set Time : 40 . Motor will turn to set angle in 0.448s.


Delay

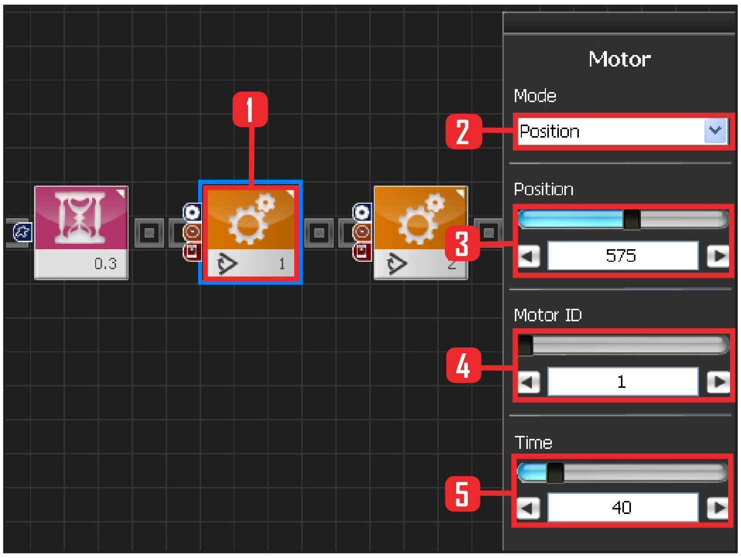
Time

0.3

31 Delay

Delay 0.3s.

Short delay as dance has started.


Motor

Mode: Position

Position: 575

Motor ID: 1

Time: 40

32 Motor ID 1(Upper Right Arm) Setup

7th Stage : Wave 4 Stage

End Wave.

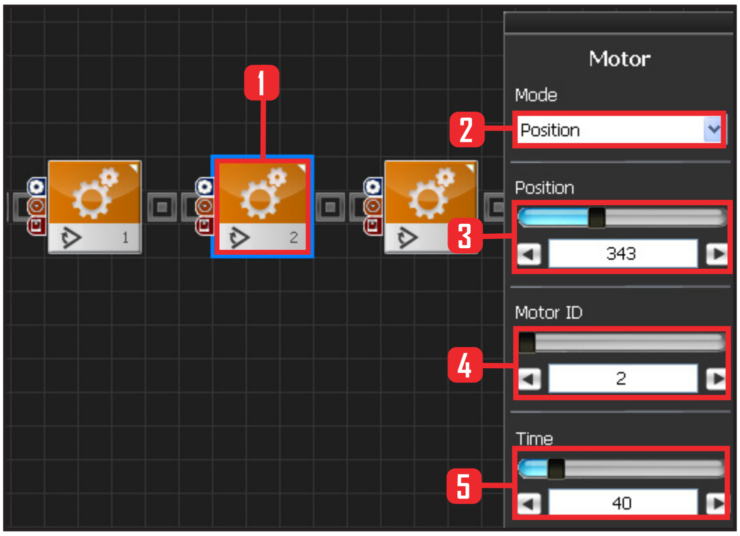
Select Motion 〉 Motor.

Select Mode : Position .

Set Position : 575.

Set Motor ID : 1.

Set Time : 40 . Motor will turn to set angle in 0.448s.


Motor

Mode: Position

Position: 343

Motor ID: 2

Time: 40

33 Motor ID 2(Lower Right Arm) Setup

End Wave.

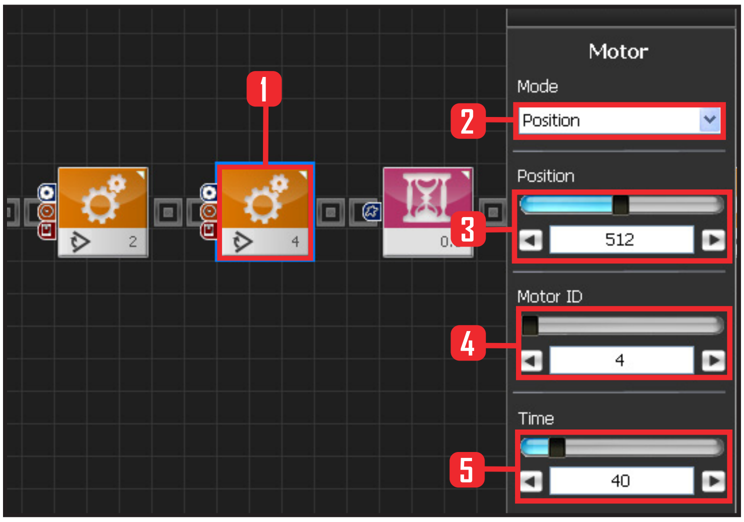
Select Motion 〉 Motor.

Select Mode : Position .

Set Position : 343.

Set Motor ID : 2.

Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode

Position

Position

512

Motor ID

4

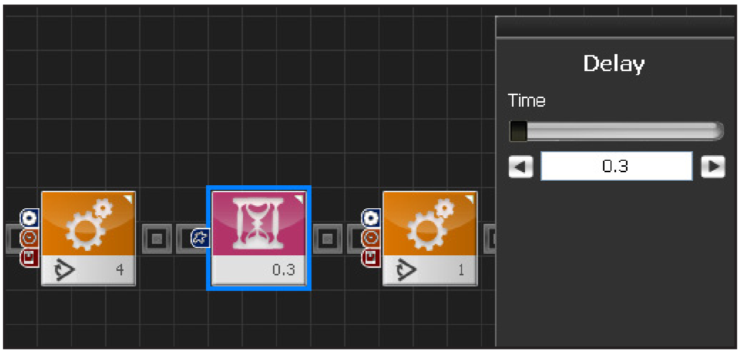
Time

40

34 Motor ID 4(Left Upper Arm) Setup

End Wave.

Select Motion 〉Motor.
Select Mode : Position .
Set Position : 512.
Set Motor ID : 4.
Set Time : 40 . Motor will turn to set angle in 0.448s.



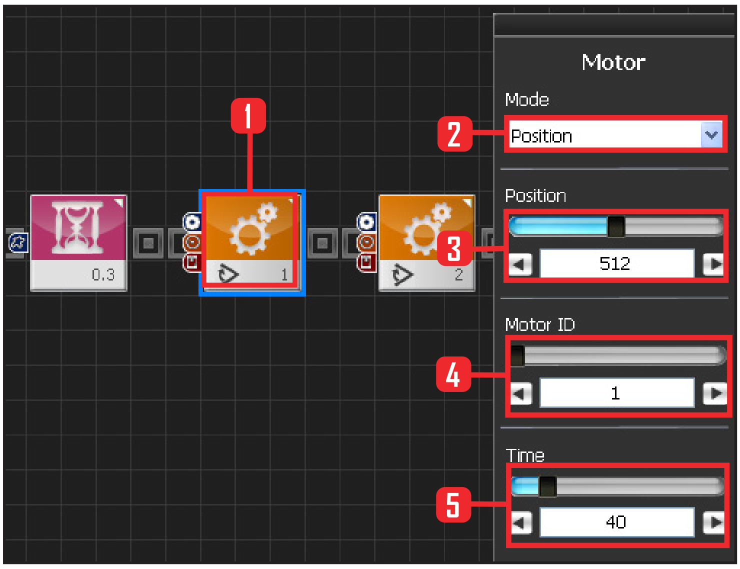
Delay

Time

0.3

35 Delay

Delay 0.3s.
Short delay as dance has started.



Motor

Mode

Position

Position

512

Motor ID

1

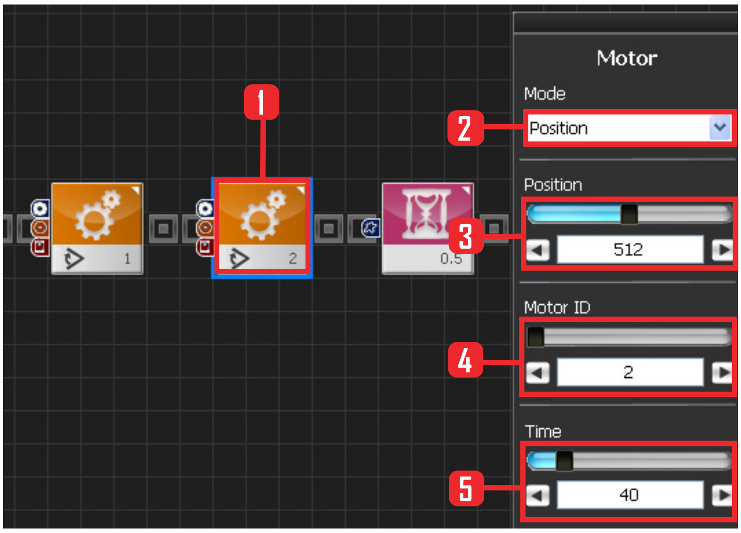
Time

40

36 Motor ID 1(Right Upper Amr) Setup

8th Stage : Wave 5 Stage
Extermd both arms to the side .

Select Motion 〉Motor.
Select Mode : Position .
Set Position : 512.
Set Motor ID : 1.
Set Time : 40 . Motor will turn to set angle in 0.448s.



Motor

Mode: Position

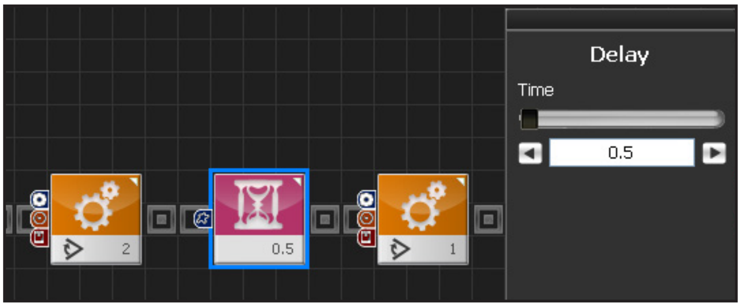
Position: 512

Motor ID: 2

Time: 40

37 Motor ID 2(Lower Right Am) Setup

Externd both arms to the side .
Select Motion 〉 Motor.
Select Mode : Position .
Set Position : 512.
Set Motor ID : 2.
Set Time : 40 . Motor will turn to set angle in 0.448s.

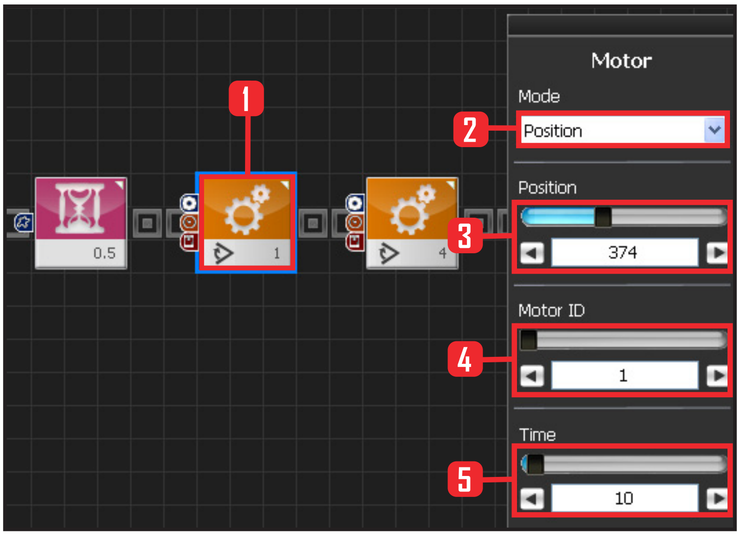


Delay

Time: 0.5

38 Delay

Delay 0.5s



Motor

Mode: Position

Position: 374

Motor ID: 1

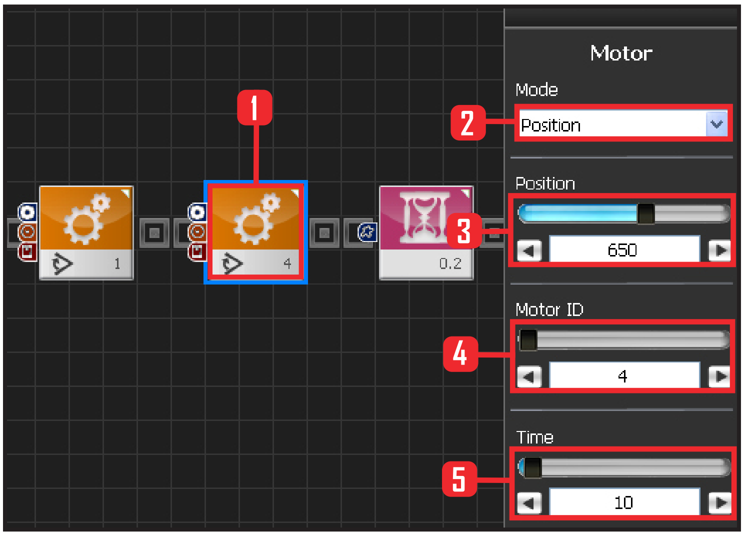
Time: 10

39 Motor ID 1(Upper Right Arm) Setup

9th Stage : Lower arm to 45 degrees

Return to attention posture, change arm angle to 45 degrees first.

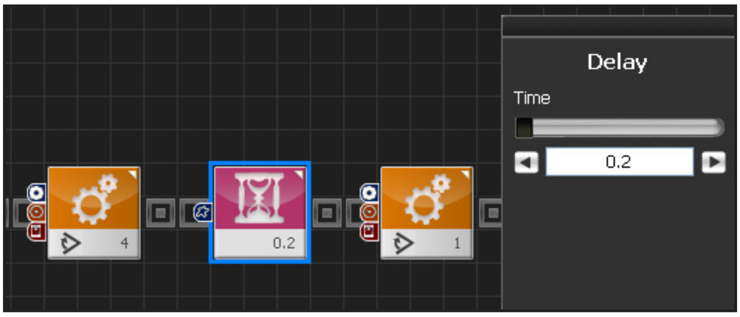
Select Motion 〉 Motr.
Select Mode : Position.
Set Position : 374.
Set Motor ID : 1.
Set Time : 10 .



40 Motor ID 4(Left Upper Arm) Setup

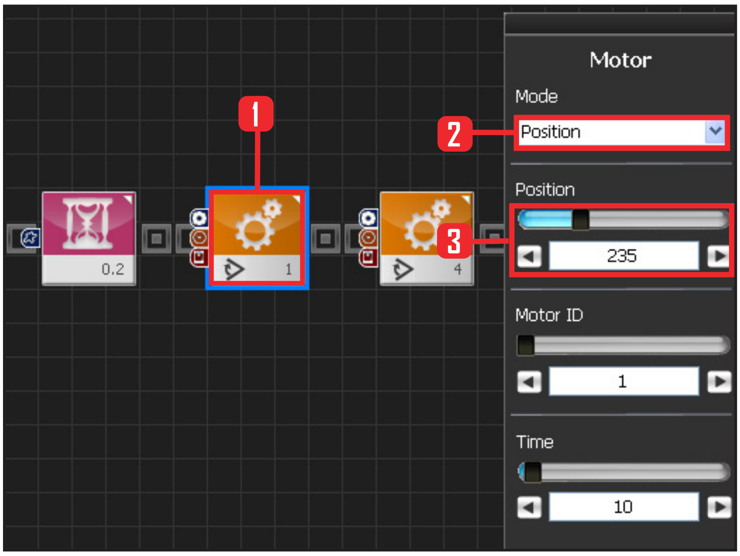
Return to attention posture, change arm angle to 45 degrees first.

Select Motion 〉 Motor.
Select Mode : Position.
Set Position : 650.
Set Motor ID : 4.
Set Time : 10 .



41 Delay

Delay 0.2s .



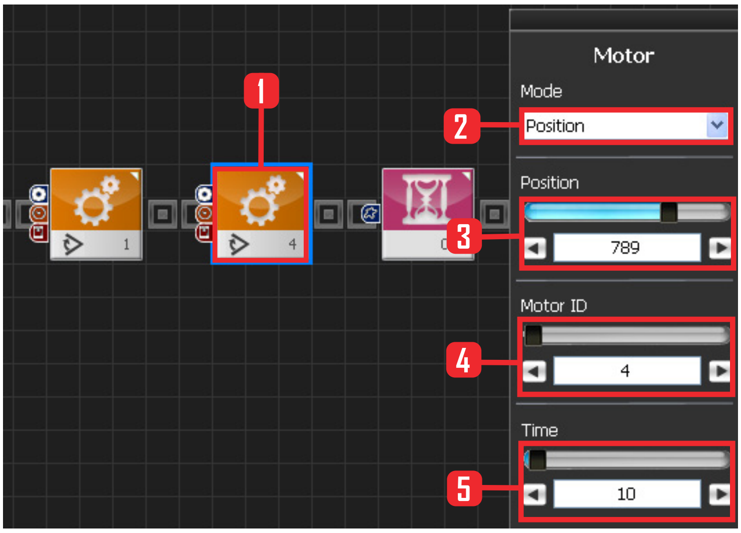
42 Motor ID 1(Upper Right Arm) Setup

10th Stage : Dance Complete

Return to attention posture.

Select Motion 〉 Motor.
Select Mode : Position.
Set Position : 235.
Set Motor ID : 1.
Set Time : 10 .

43 Motor ID 4(Upper Left Arm) Setup



Motor

Mode: Position

Position: 789

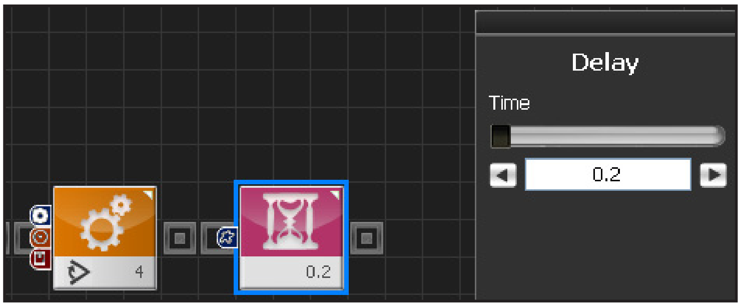
Motor ID: 4

Time: 10

Return to attention posture.

Select Motion 〉 Motor.
Select Mode : Position.
Set Position : 235.
Set Motor ID : 4.
Set Time : 10 .

44 Delay

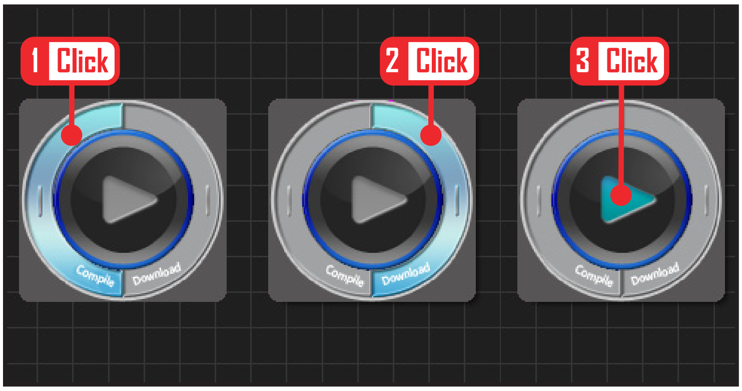


Delay

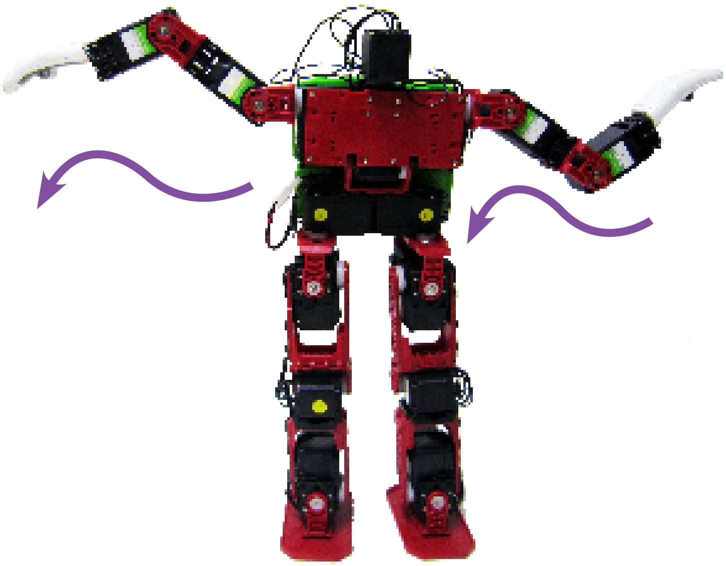
Time: 0.2

Delay 0.2s .

45 Compile, Download, Run



Click ‘Compile’. Click ‘download’ on the right if there is no compilation error. Download to robot. Click ‘Run’ button (Arrow button) after the download.



46 Robot Motion

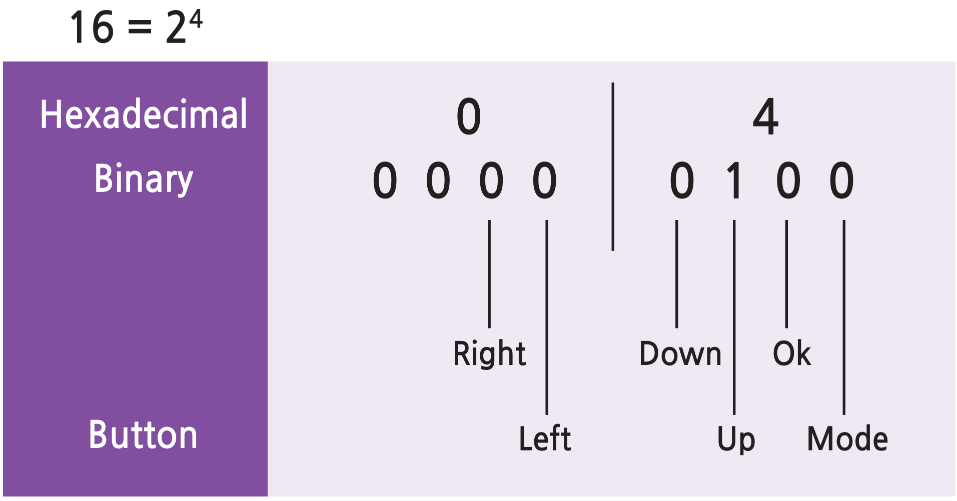
Wave dance will start from the left arm.

Button, LED Example Step by Step

Example Description

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

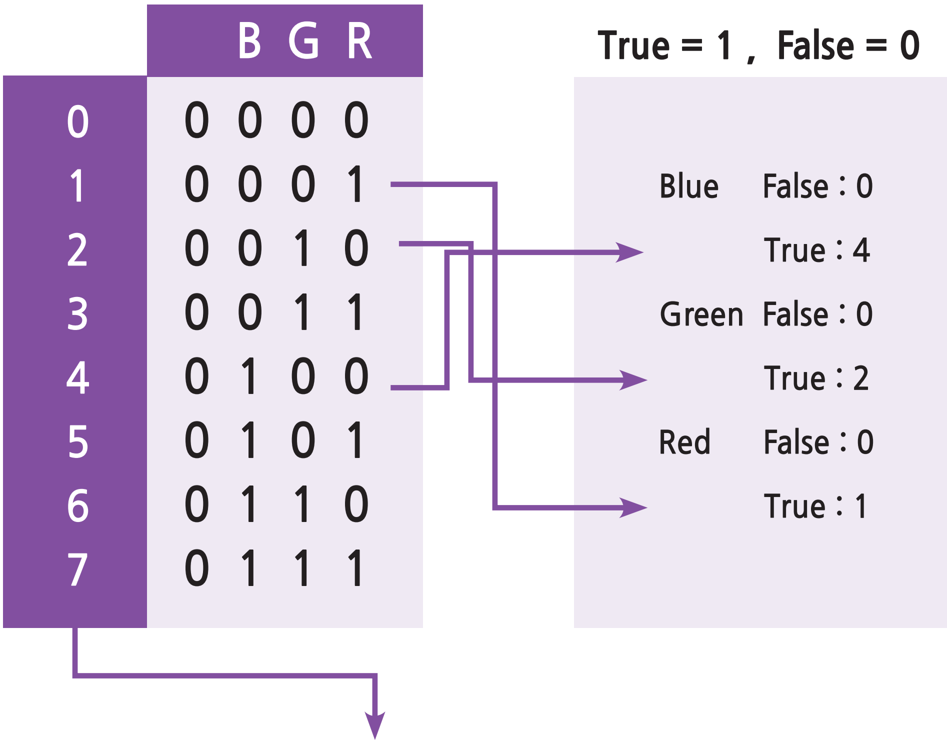
In order to program the button and the LED, user should have an understanding of how the button and the LED work.



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.



		B	G	R
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

True = 1 , False = 0

Blue	False : 0
	True : 4
Green	False : 0
	True : 2
Red	False : 0
	True : 1

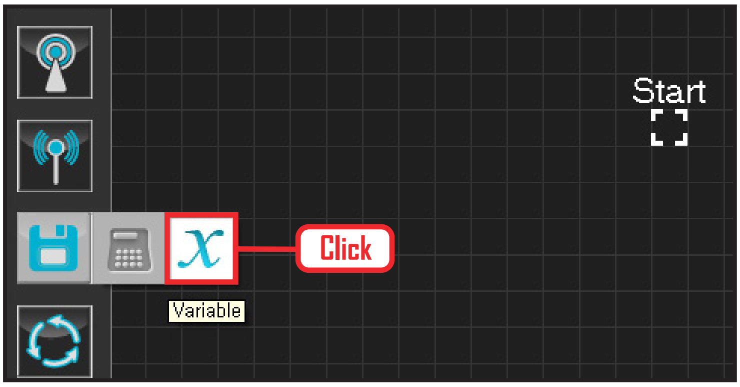
led( )

led(4 X Blue + 2 X Green + 1 X Red)

LED

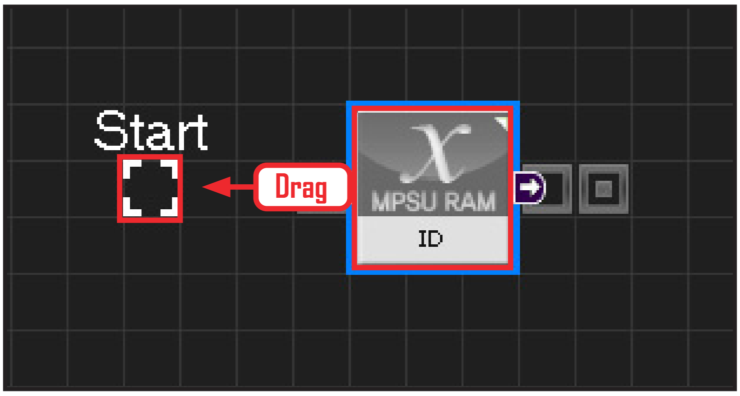
DRC has seven LEDs but only three can be controlled by the task mode. Three bits are required to express on/off status of the three LEDs; Red, Green, Blue. As shown in the diagram above, each LED (Red, Green Blue) is matched with a bit in an ascending order from the lowest bit of the byte to the highest. 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.



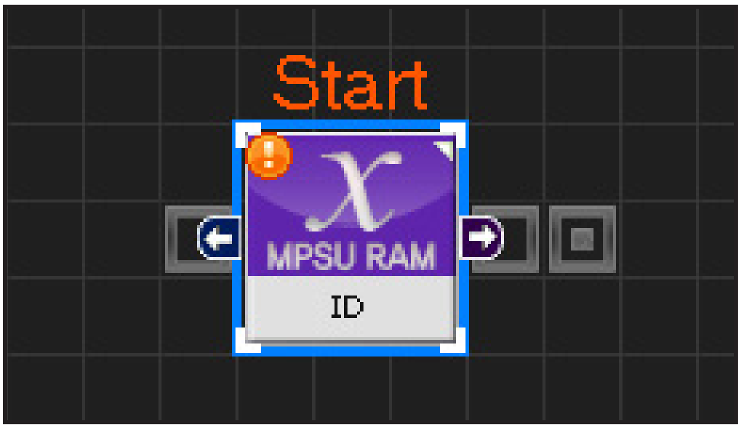
01 Assign Variable

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.

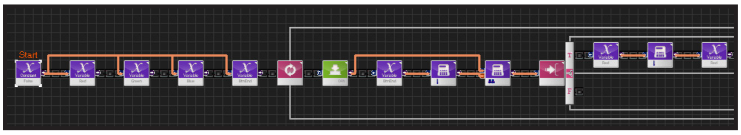


03 Start Programming

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.

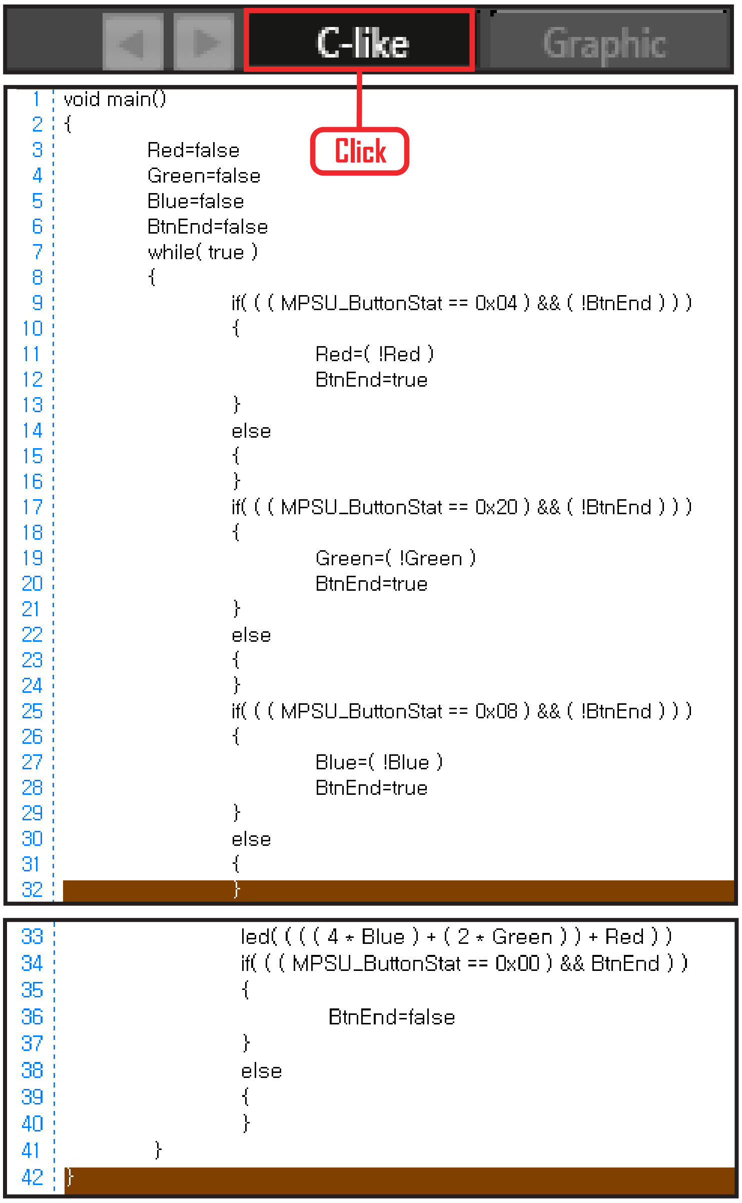
04 Entire Program

Entire program using the buttons and LED.

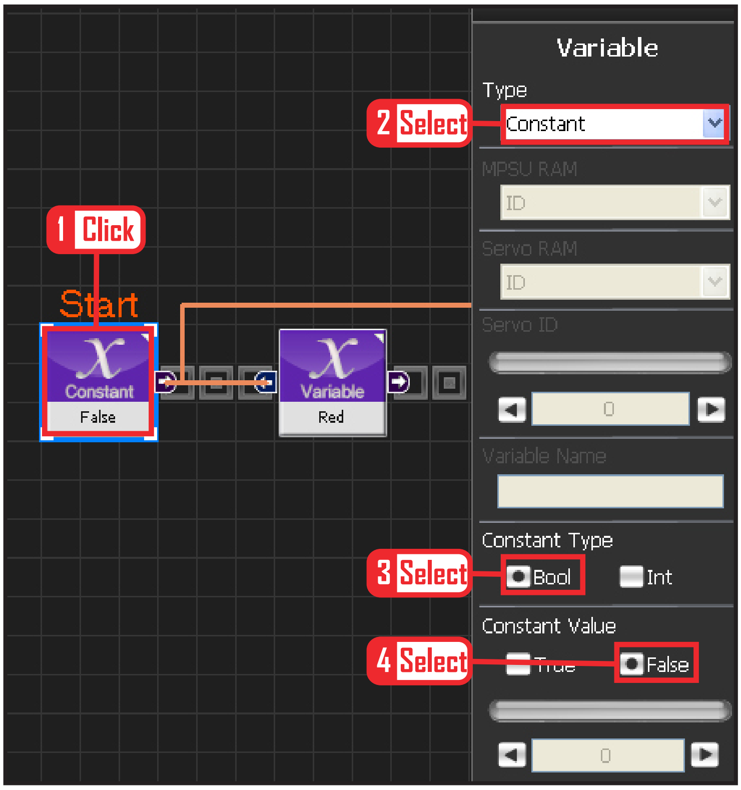


05 Viewing C-Like

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.



```
void main()
{
        Red=false
        Green=false
        Blue=false
        BtnEnd=false
        while( true )
        {
                if( ( ( MPSU_ButtonStat == 0x04 ) && ( !BtnEnd ) ) )
                {
                        Red=( !Red )
                        BtnEnd=true
                }
                else
                {
                }
                if( ( ( MPSU_ButtonStat == 0x20 ) && ( !BtnEnd ) ) )
                {
                        Green=( !Green )
                        BtnEnd=true
                }
                else
                {
                }
                if( ( ( MPSU_ButtonStat == 0x08 ) && ( !BtnEnd ) ) )
                {
                        Blue=( !Blue )
                        BtnEnd=true
                }
                else
                {
                }
                led( ( ( ( 4 * Blue ) + ( 2 * Green ) ) + Red ) )
                if( ( ( MPSU_ButtonStat == 0x00 ) && BtnEnd ) )
                {
                        BtnEnd=false
                }
                else
                {
                }
        }
}
```



1 Click

2 Select Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name:

3 Select Constant Type: ● Bool □ Int

4 Select Constant Value: □ True ● False

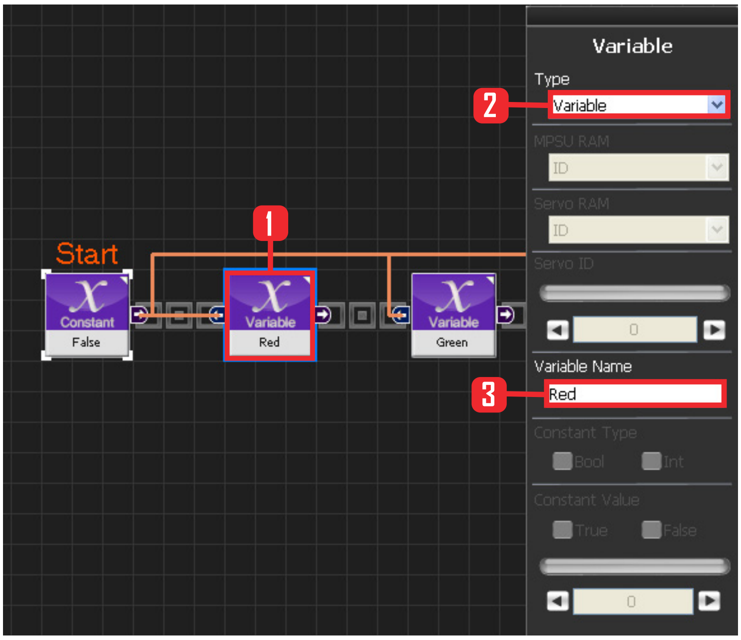
0

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.



1

2 Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

3 Variable Name: Red

Constant Type: □ Bool □ Int

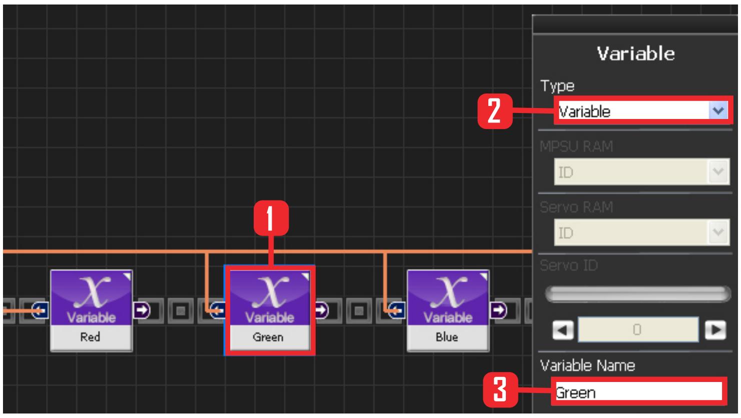
Constant Value: □ True □ False

0

07 Red Variable

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

Red LED off when False, on when True.



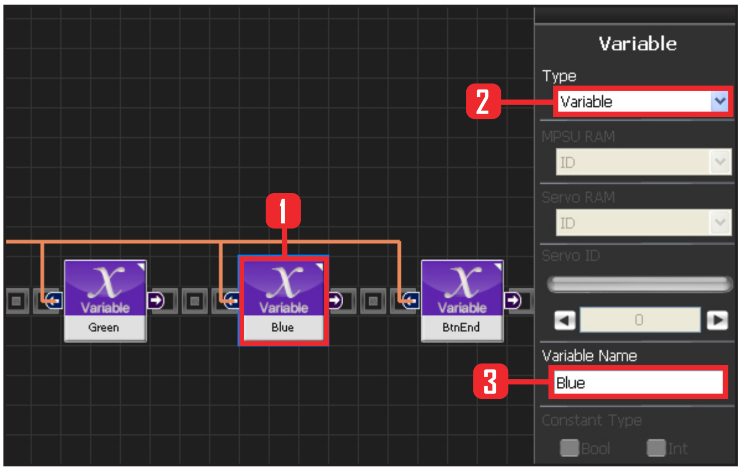
Type: Variable

Variable Name: Green

08 Green Variable

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

Green LED off when False, on when True



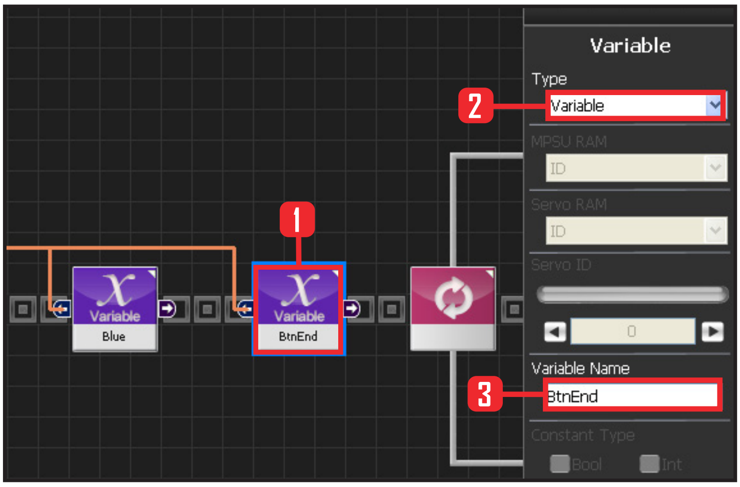
Type: Variable

Variable Name: Blue

09 Blue Variable

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

Blue LED off when False, on when True



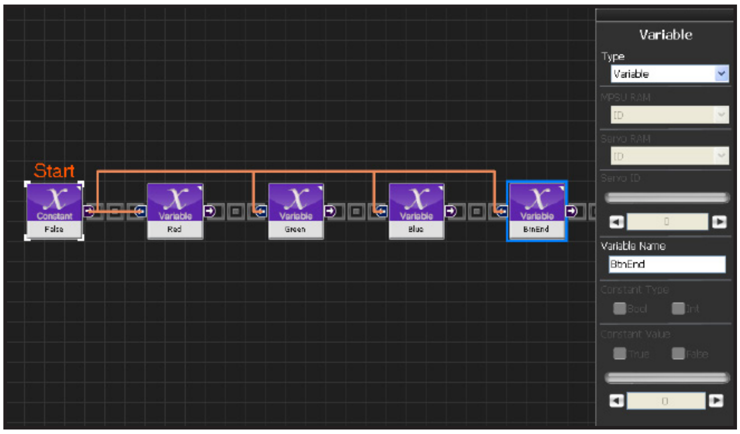
Type: Variable

Variable Name: BtnEnd

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 .



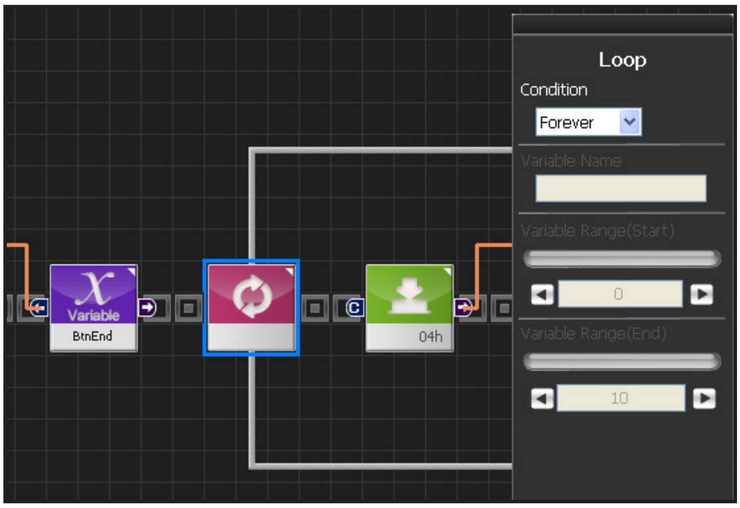
Variable

Type: Variable

Variable Name: BtnEnd

11 Assign Variable

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



Loop

Condition: Forever

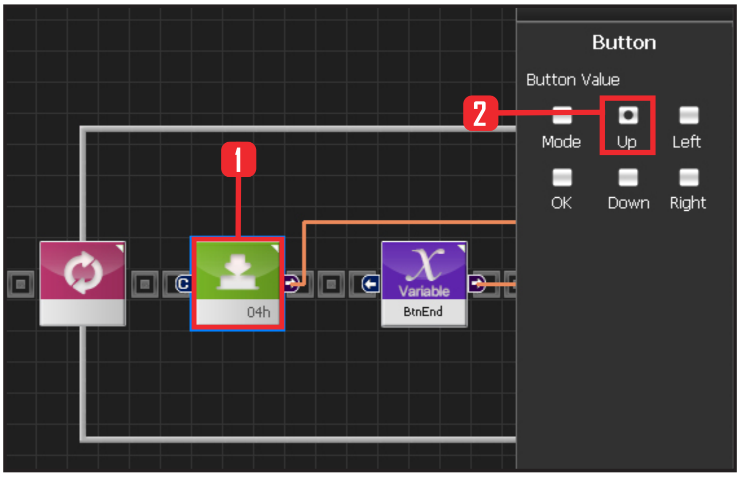
Variable Name

Variable Range(Start): 0

Variable Range(End): 10

12 Loop

Forever infinite repetition.



Button

Button Value

□ Mode ● Up □ Left

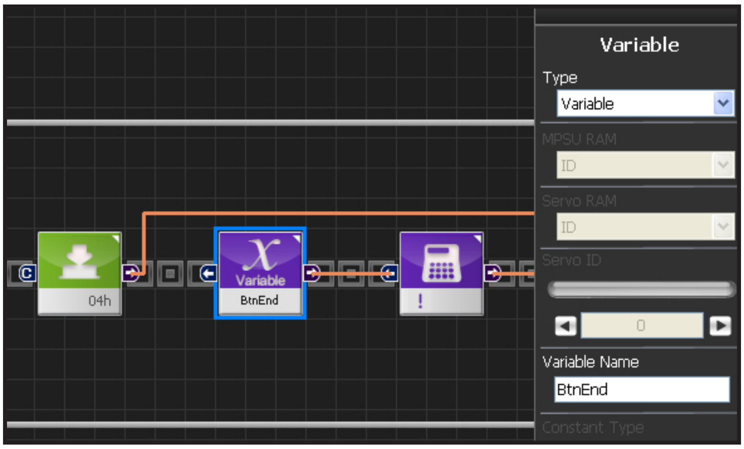
□ OK □ Down □ Right

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.



Variable

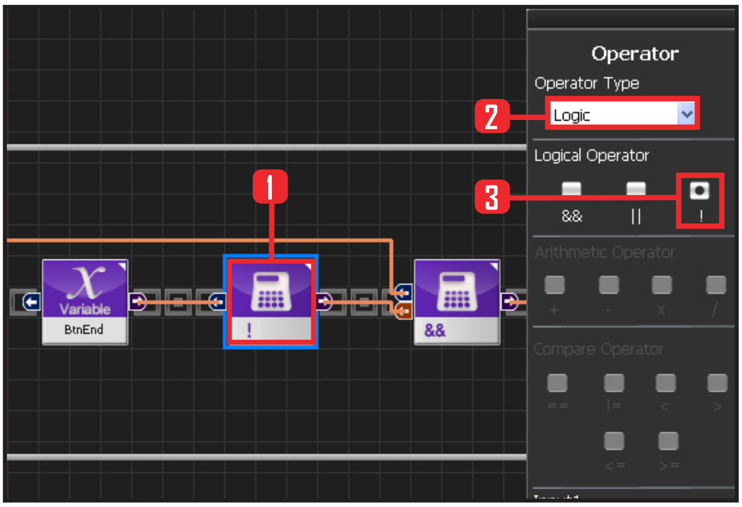
Type: Variable

Variable Name: BtnEnd

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.



Operator

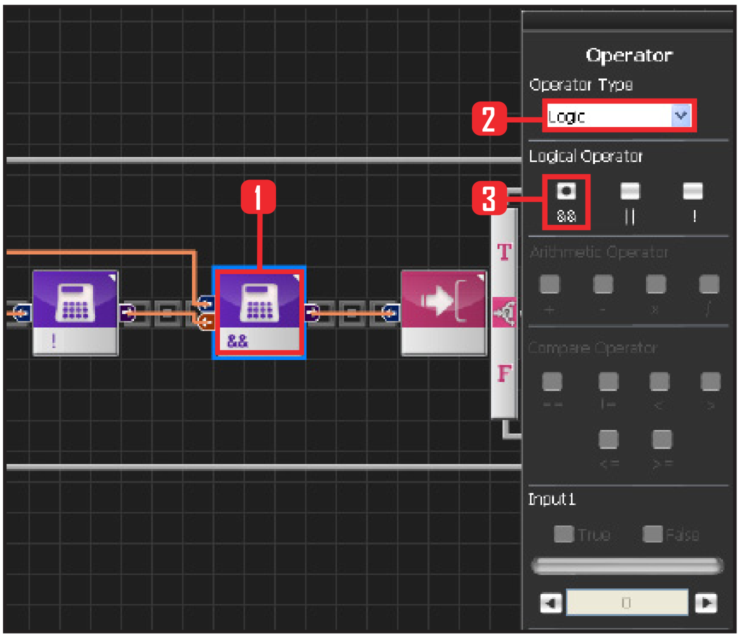
Operator Type: Logic

Logical Operator: ● !

15 ! Operator

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

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



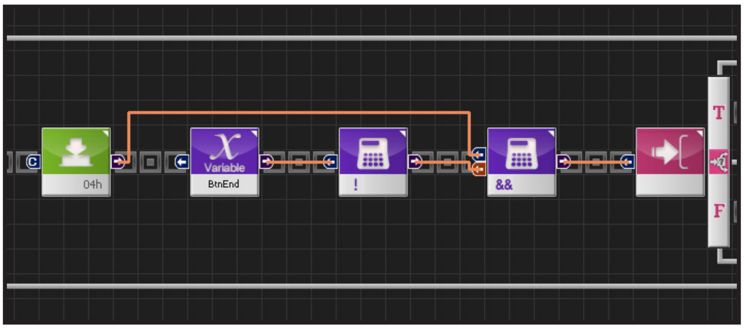
Operator

Operator Type: Logic

Logical Operator: ● &&

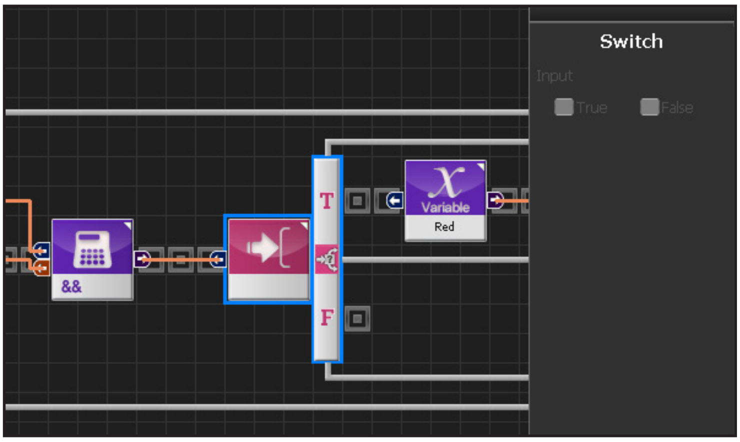
16 And Operator

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.



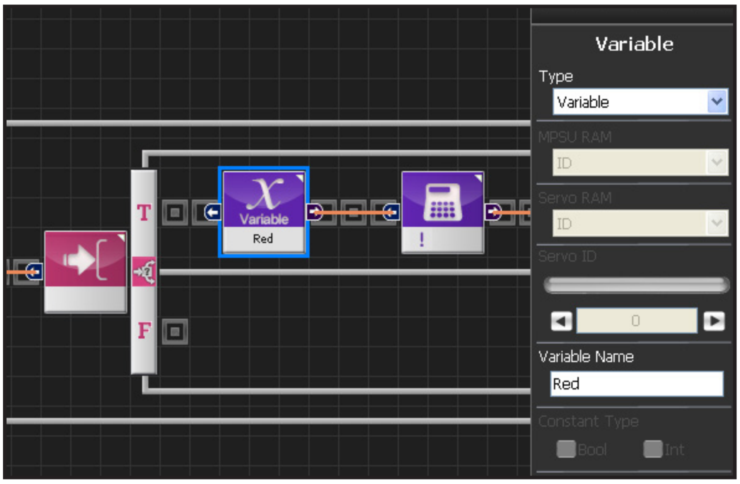
Switch

Input

□ True □ False

18 If Switch

Runs the upper part when True.



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

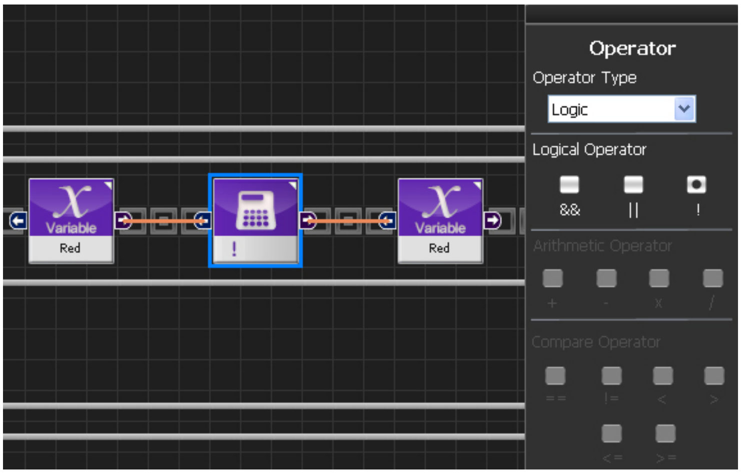
Servo ID: 0

Variable Name: Red

Constant Type: □ Bool □ Int

19 Red Output

Copy and paste Red variable from front.



Operator

Operator Type: Logic

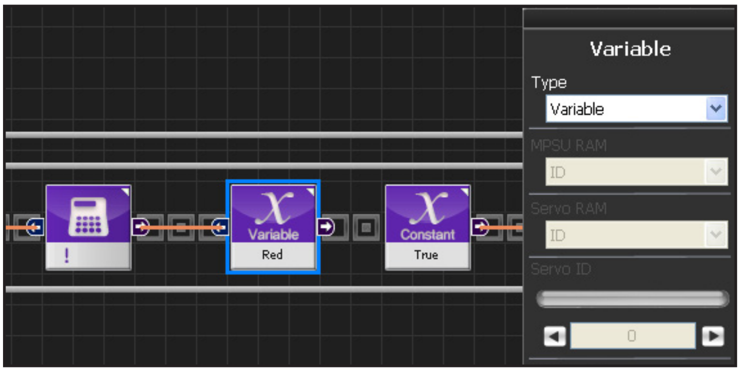
Logical Operator
□ && □ || ● !

Arithmetic Operator
□ + □ - □ x □ /

Compare Operator
□ == □ != □ < □ >
□ <= □ >=

20 ! Operator

When Red is True it becomes False and vice versa.



Variable

Type: Variable

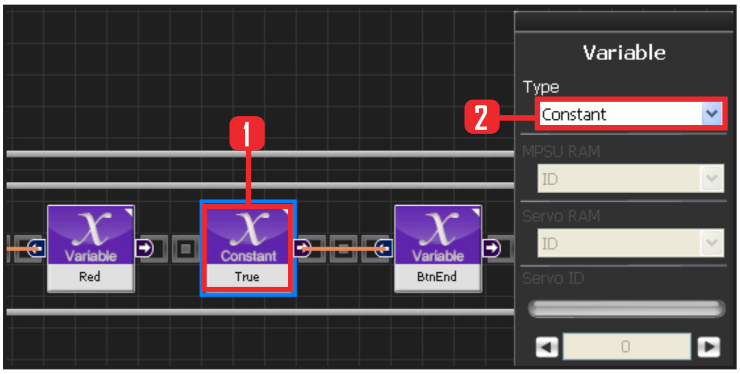
MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

21 Red Input

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



Variable

Type: Constant

MPSU RAM: ID

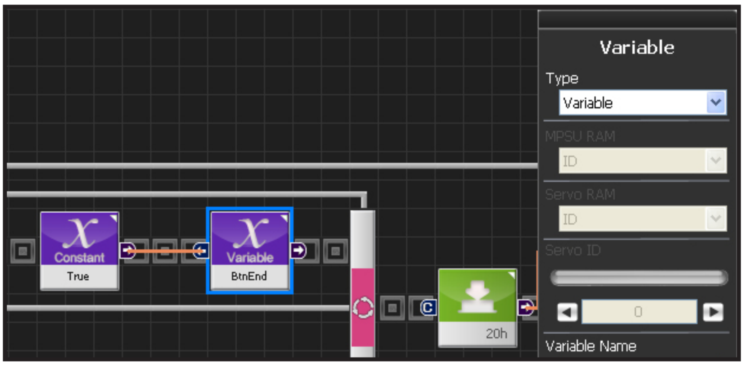
Servo RAM: ID

Servo ID: 0

22 True Setup

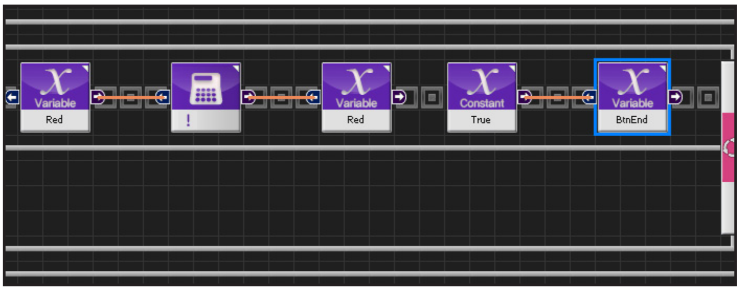
With the programmed motion been finished after press button, the BtnEnd should be changed from false to true.

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



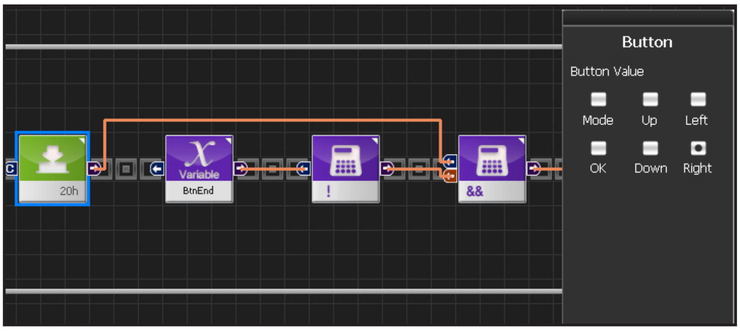
23 BtnEnd to True

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.



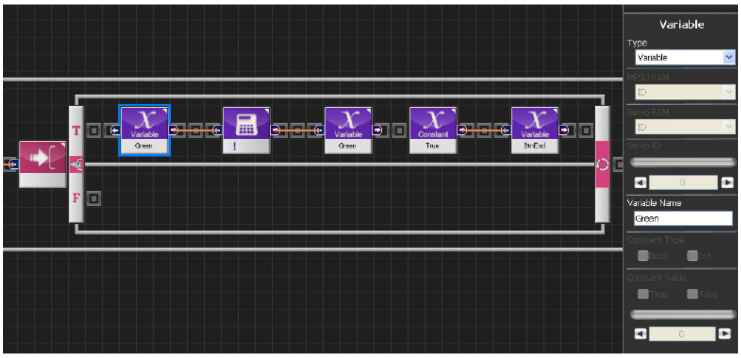
24 Red LED

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.



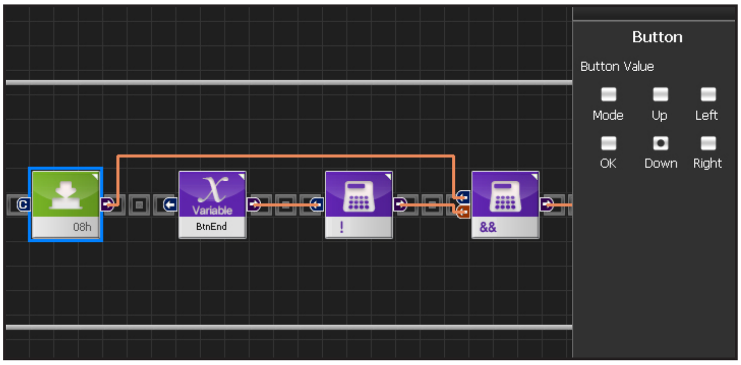
Variable

Type: Variable

Variable Name: Green

26 Green LED

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



Button

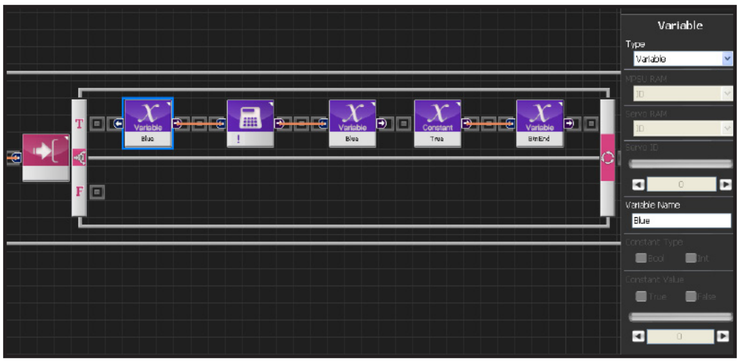
Button Value

□ Mode □ Up □ Left

□ OK ● Down □ Right

27 Down Button

When down button is pressed.



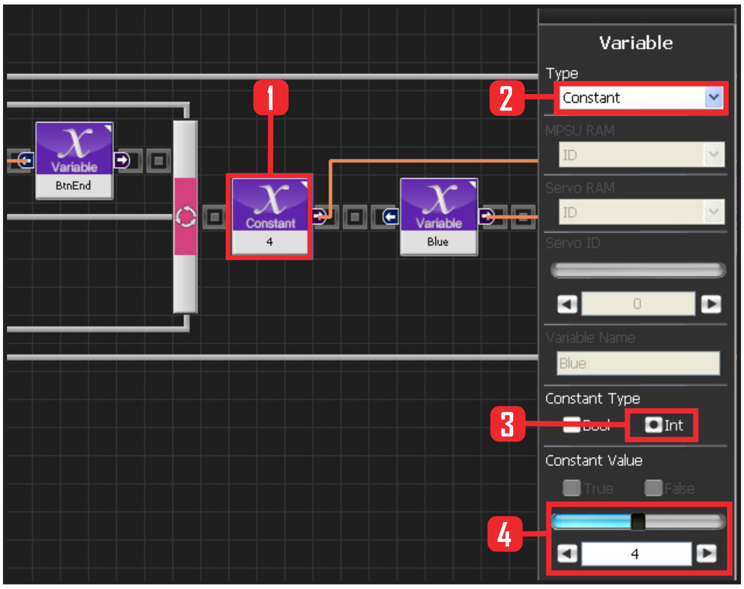
Variable

Type: Variable

Variable Name: Blue

28 Blue LED

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



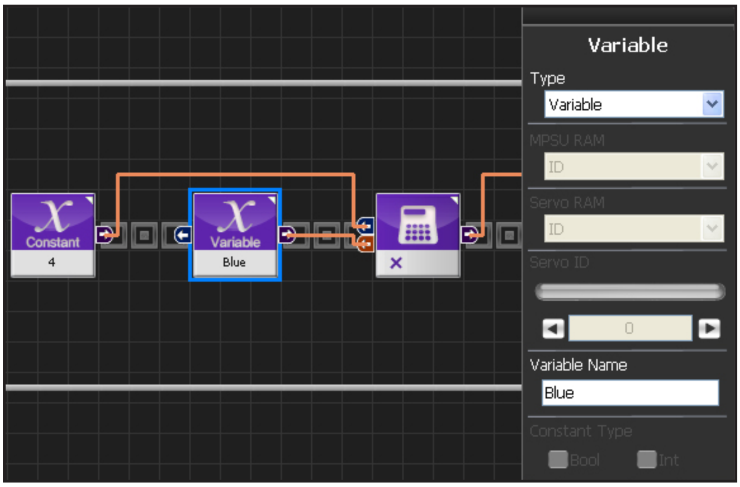
29 LED Value

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)

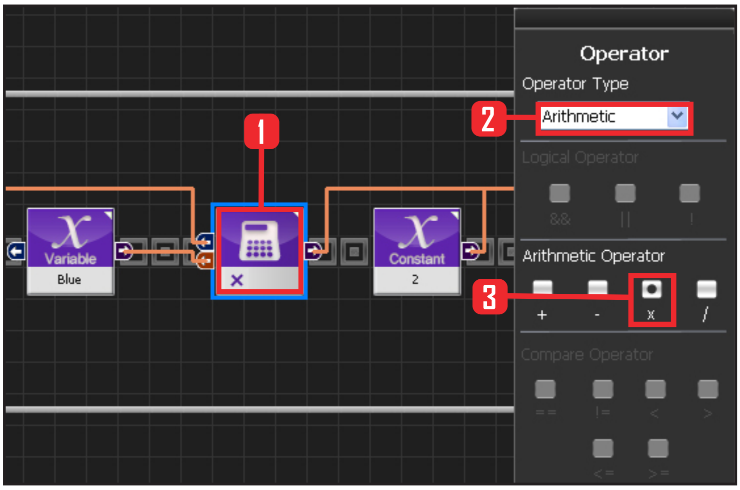
Select Data 〉Variable module.
Select Type : Contant .
Select Constant Type: int .
Set Constant Value : 4.



30 Blue

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

Copy the Blue variable from front.

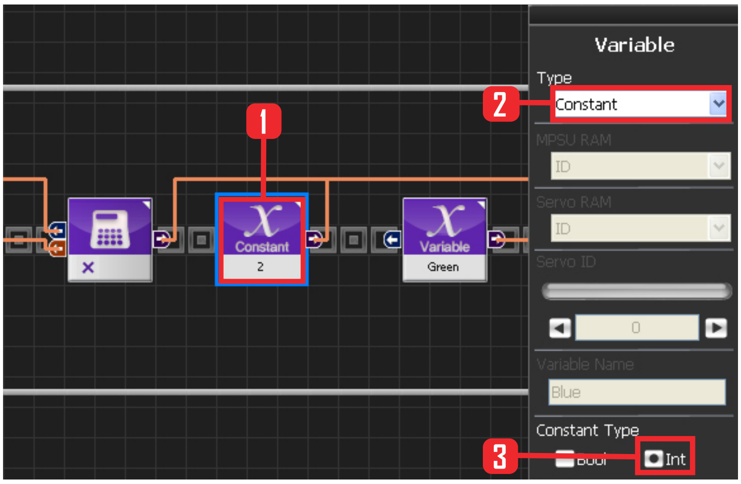


31 Multiplication

(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.



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

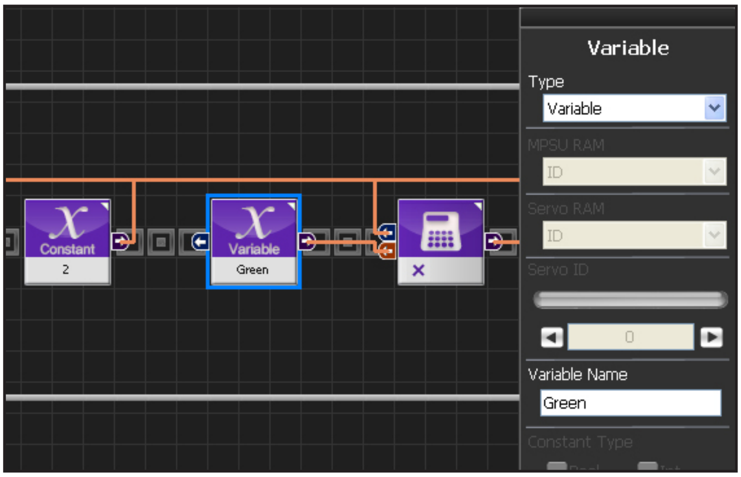
Variable Name: Blue

Constant Type: ○ Bool ● Int

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.



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

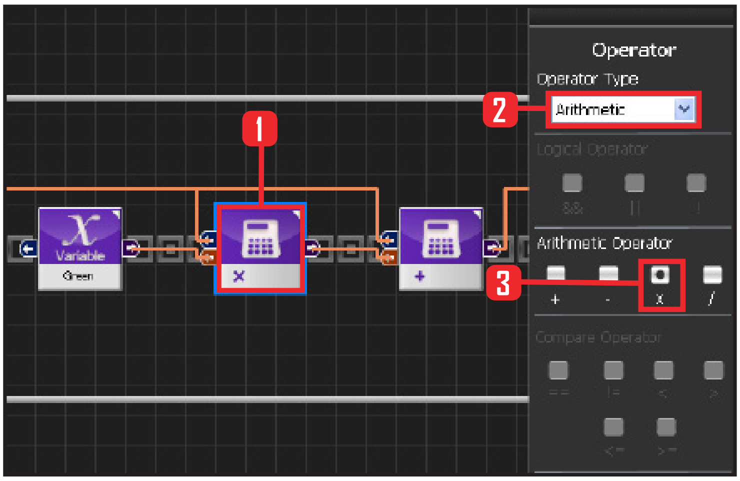
Variable Name: Green

Constant Type: ○ Bool ○ Int

33 Green

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

Copy the Green variable from front.



Operator

Operator Type: Arithmetic

Logical Operator: ○ && ○ || ○ !

Arithmetic Operator: ○ + ○ - ● x ○ /

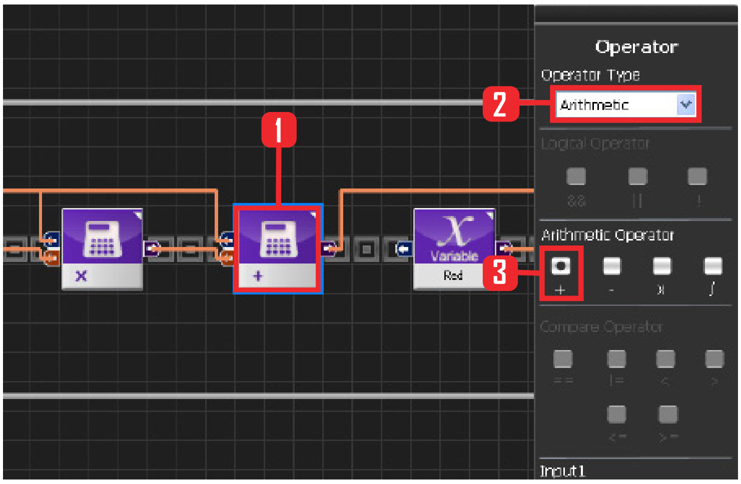
Compare Operator: ○ == ○ != ○ < ○ > ○ <= ○ >=

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.


Operator

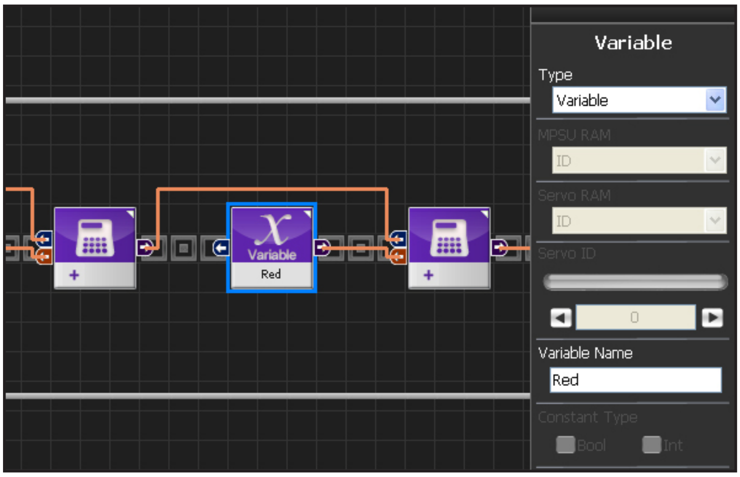
Operator Type: Arithmetic

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.


Variable

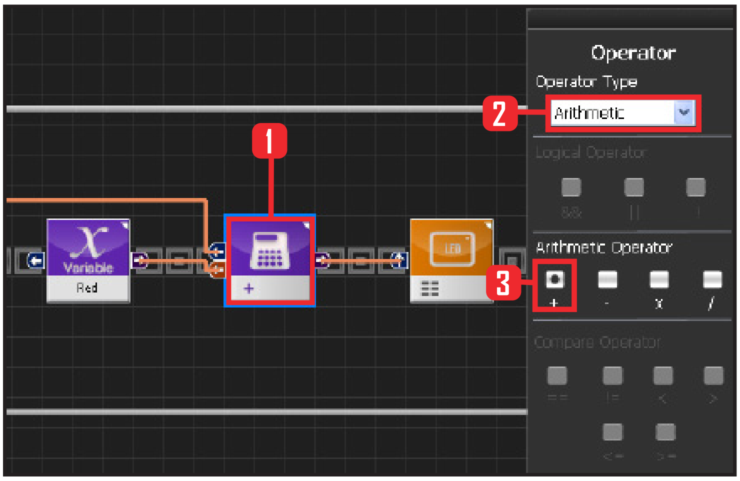
Type: Variable

Variable Name: Red

36 Red

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

Copy Red variable from front.


Operator

Operator Type: Arithmetic

37 Addition

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

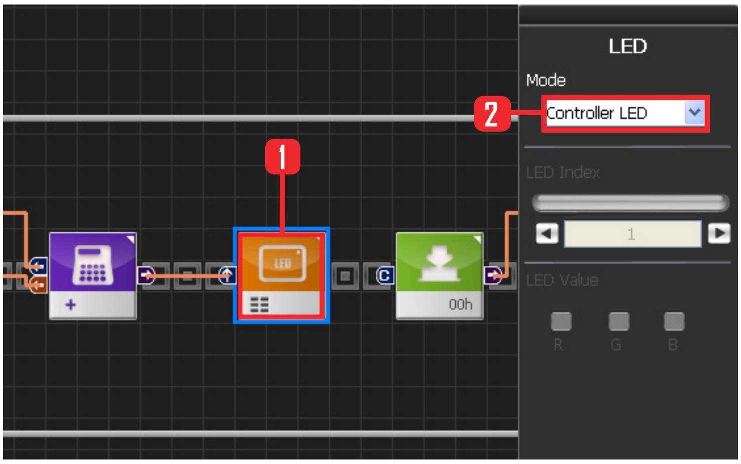
Slect Data 〉 Operator module.
Select Operartor Type : Arithmetic.
Select Arithmetic Operator : X .

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

38 LED

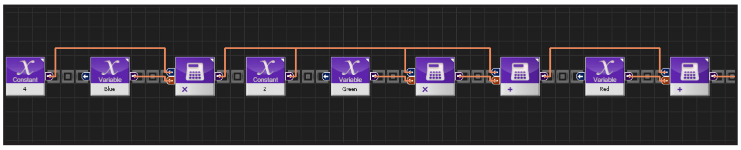
Select Motion 〉LED module.
Select Mode : Controller LED .

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



39 LED Value Output Calculation

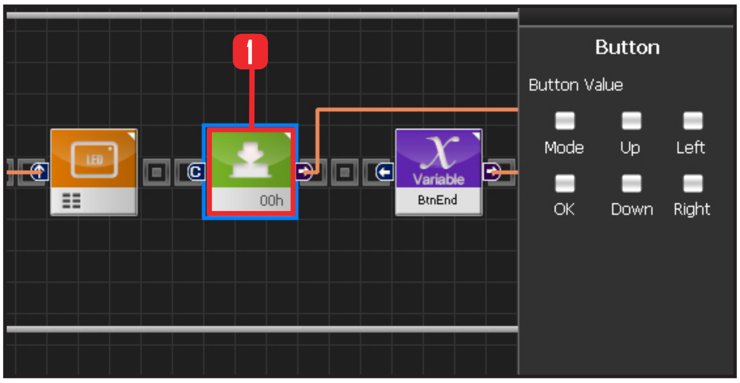
(4 x Blue + 2 x Green + 1 x Red)
Shown as connected modules.

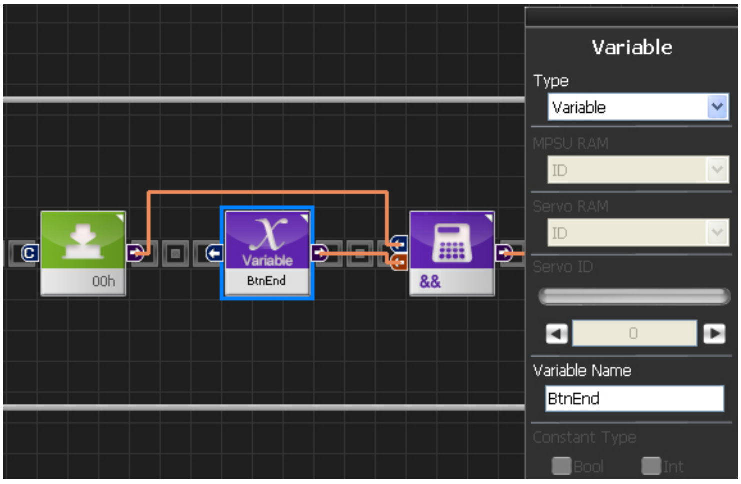


40 Button Released State

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 initialize the button.
Select Motion 〉Button module.
Button Value : None.

Released button state.





Variable

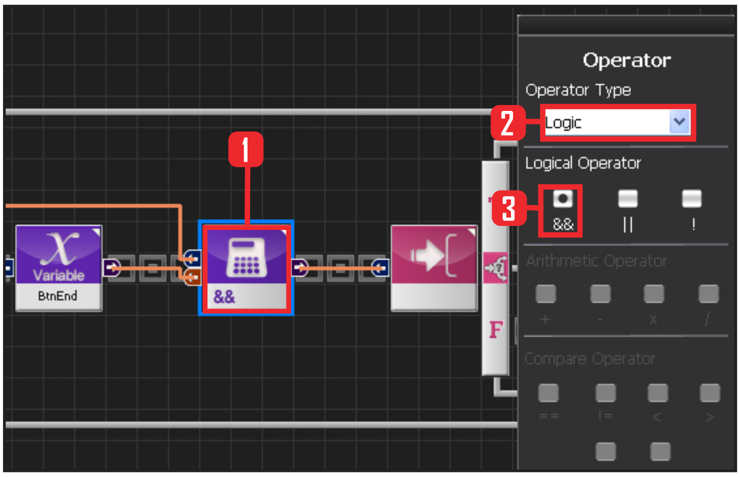
Type: Variable

Variable Name: BtnEnd

41 BtnEnd is True

When BtnEnd is True .

Copy BtnEnd variable from front.



Operator

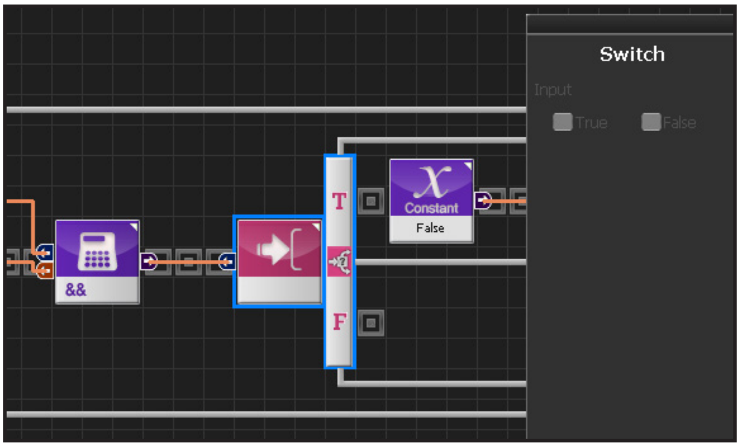
Operator Type: Logic

Logical Operator: &&

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 : &&



Switch

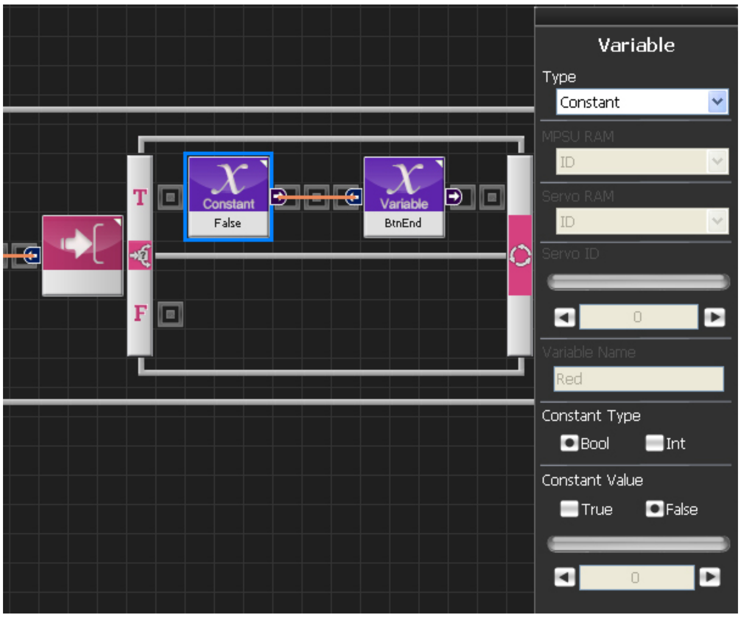
43 If Conditional Statement

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.



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: Red

Constant Type: ● Bool ○ Int

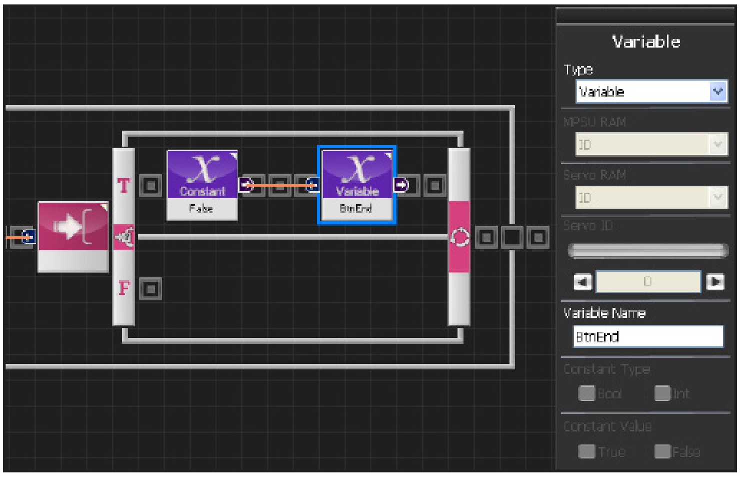
Constant Value: ○ True ● False

0

45 Change BtnEnd to False

Input False value to BtnEnd .

Copy BtnEnd variable from front.



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

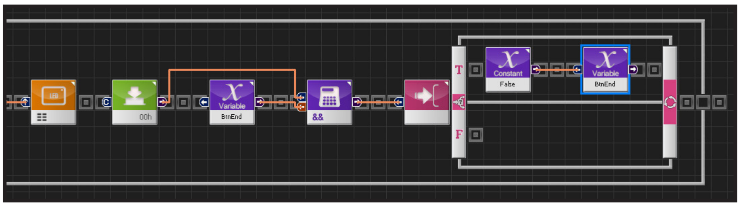
Variable Name: BtnEnd

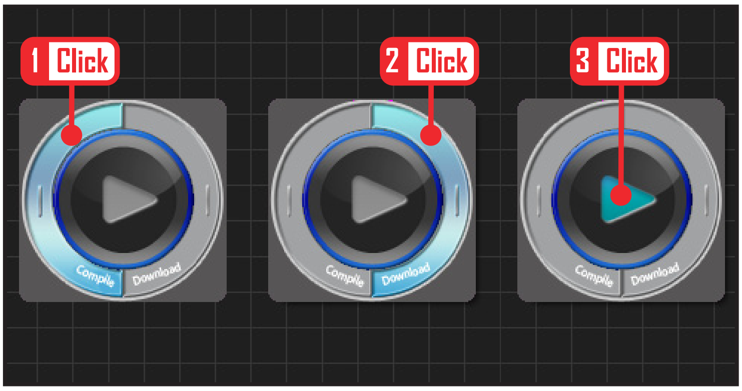
Constant Type: ○ Bool ○ Int

Constant Value: ○ True ○ False

46 Initialize Button When Released

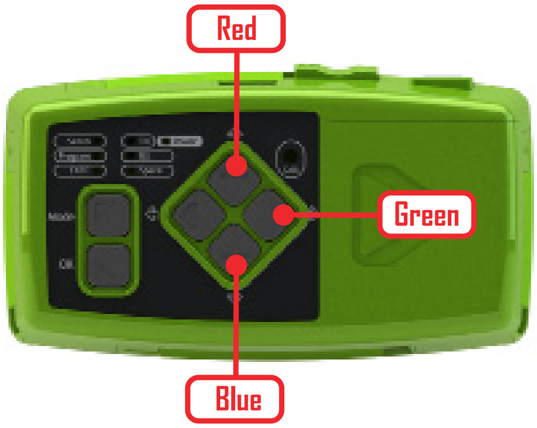
From the just releaed button state, initialize BtnEnd to false .





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

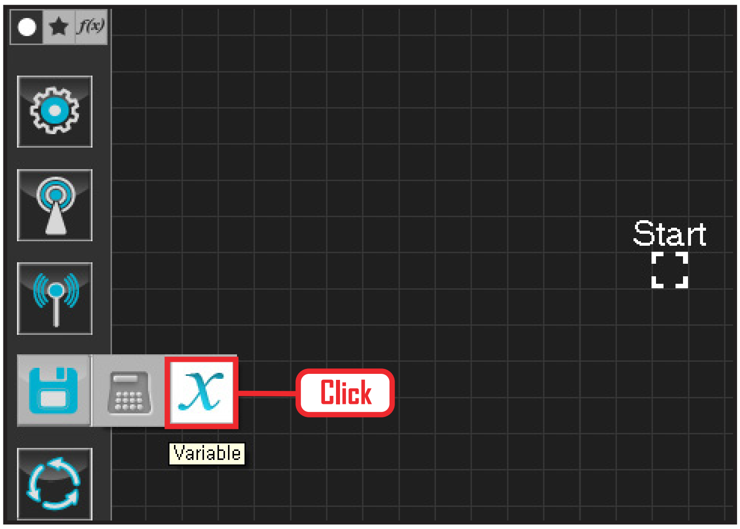
Up button: Red
Right button: Green
Down button: Blue
LEDs will light up when pressed and go off when pressed once more.

Light Example Step by Step

Example Description

This example uses the light luminosity to operate the robot motors.

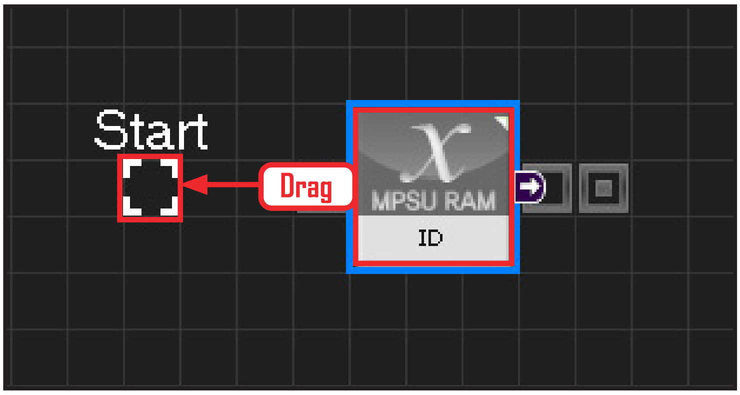
Robot will lift the left arm when luminosity decreases.



01 Assign Variable

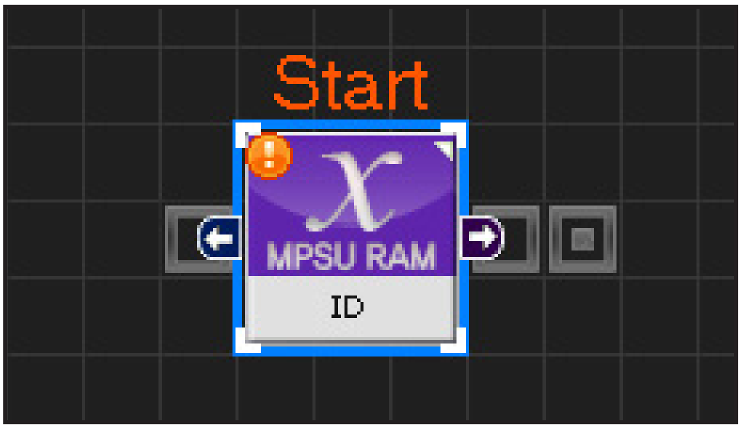
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

Click Data 〉 Variable module.



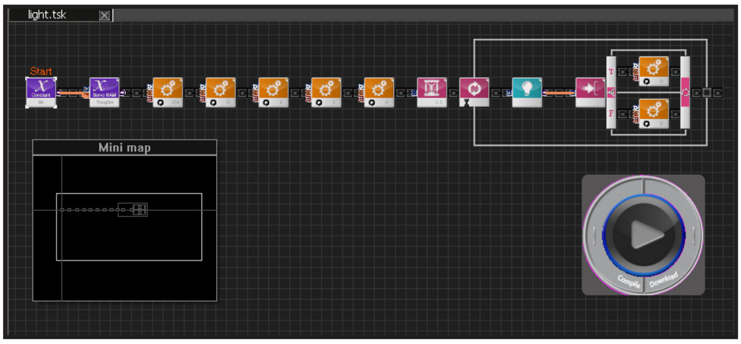
02 Start

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



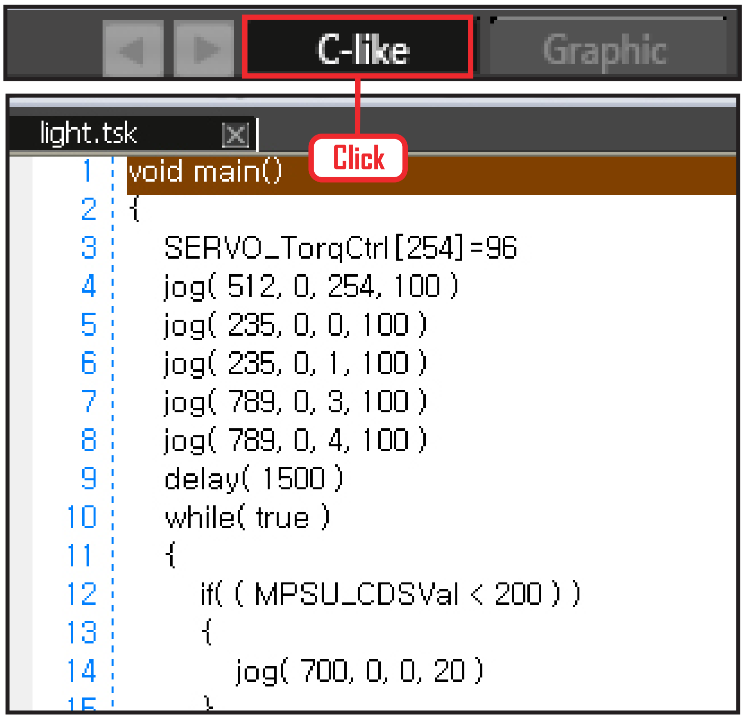
03 Start Programming

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

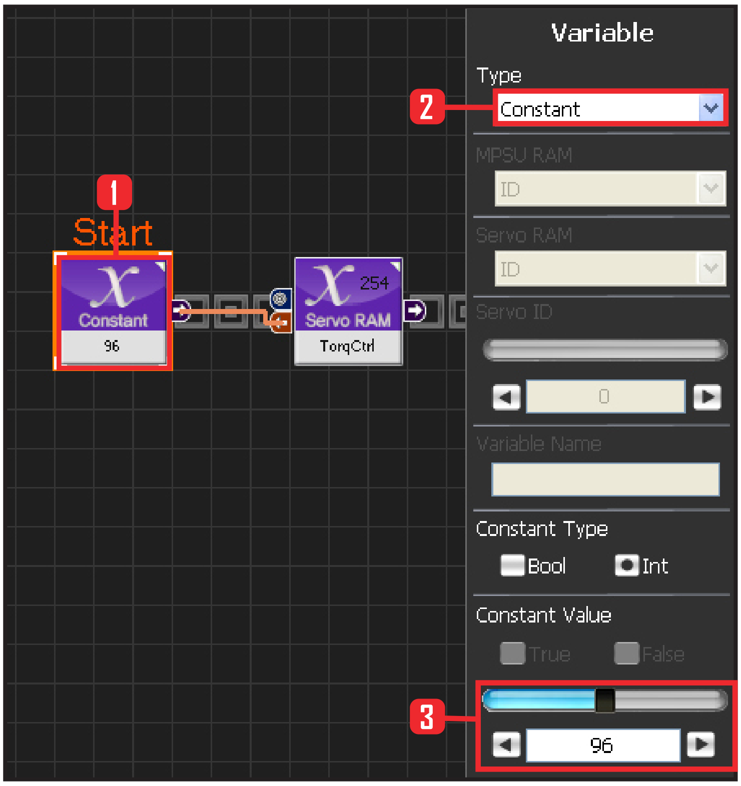
Entire program showing the light sensor to operate the robot motors.



```
void main()
{
   SERVO_TorqCtrl[254]=96
   jog( 512, 0, 254, 100 )
   jog( 235, 0, 0, 100 )
   jog( 235, 0, 1, 100 )
   jog( 789, 0, 3, 100 )
   jog( 789, 0, 4, 100 )
   delay( 1500 )
   while( true )
   {
      if( ( MPSU_CDSVal < 200 ) )
      {
         jog( 700, 0, 0, 20 )
      }
```

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.



Variable

Type: Constant

Constant Type: □ Bool ● Int

Constant Value: □ True □ False

96

06 Setup Constant

This section allows the servo motor to operate on it's own.

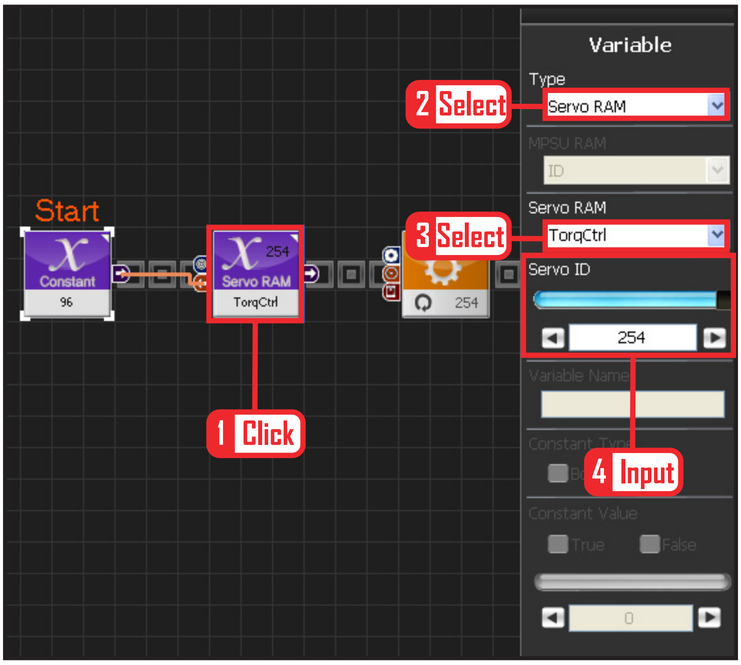
Select Constant as the Variable Type. In properties, set constant value as 96.

When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos.



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

Variable Name

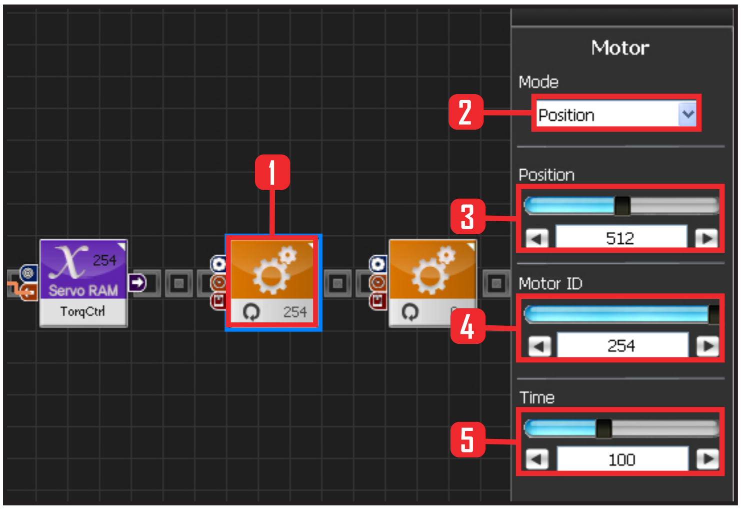
Constant Type

Constant Value
□ True □ False
0

08 Set Angle to All Servos

This section sets all servo motor angles to the center.

Select Motion 〉Motor.
Select Mode : Positon. adjust angle.
Set Position : 512 . 512 means motor will be sent to the center
Set Motor ID : 254 . 254 means it will be applied to all connected servos.
Set Time : 100 . 1 unit = 11.2ms. 100 units would be approximately 1.12s.
It means motors will be positioned at the desired angle for 1.12s.



Motor

Mode: Position

Position: 512

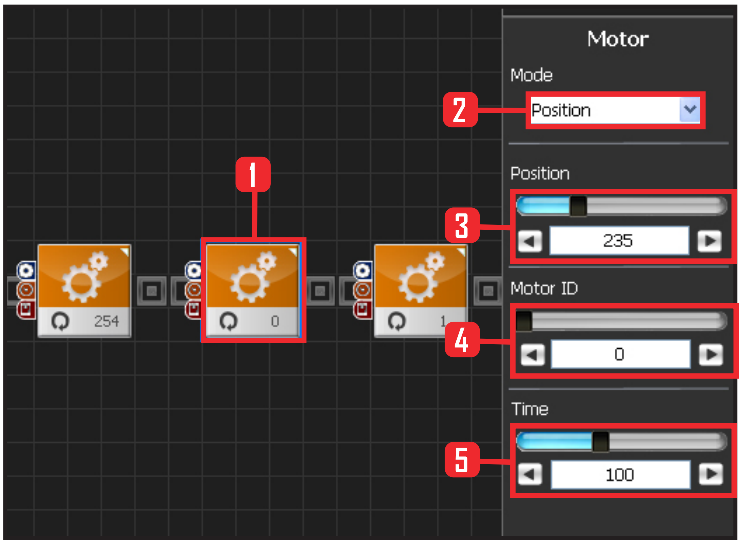
Motor ID: 254

Time: 100

09 Setup Motor ID 0 (Right Shoulder)

Creating attention posture (Basic Posture)
When all robot motors are aligned to the center, humanoid robot arms will be stretched out to the side. Setup below lowers one arm to the side of the body.

Select Motion 〉Motor .
Select Mode : Position.
Set Position : 235. 235 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.
Set Motor ID : 0. Right shoulder motor has ID 0
Set Time : 100. Motor will turn to the desired angle in approximately 1.12s.



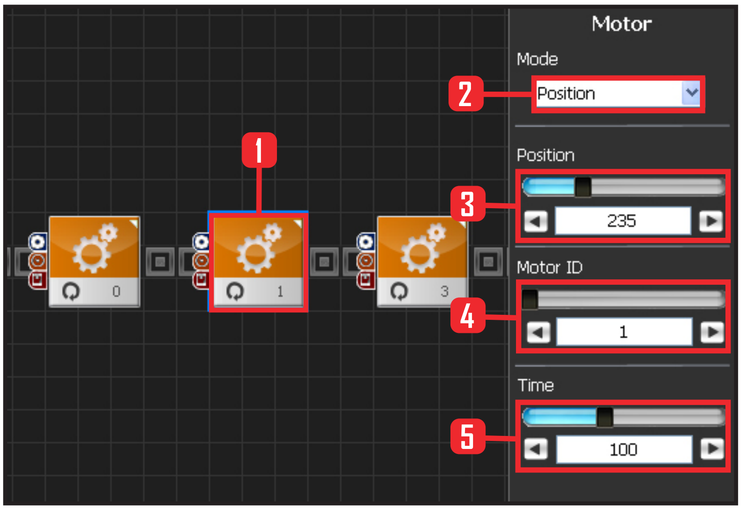
Motor

Mode: Position

Position: 235

Motor ID: 0

Time: 100



Motor

Mode: Position

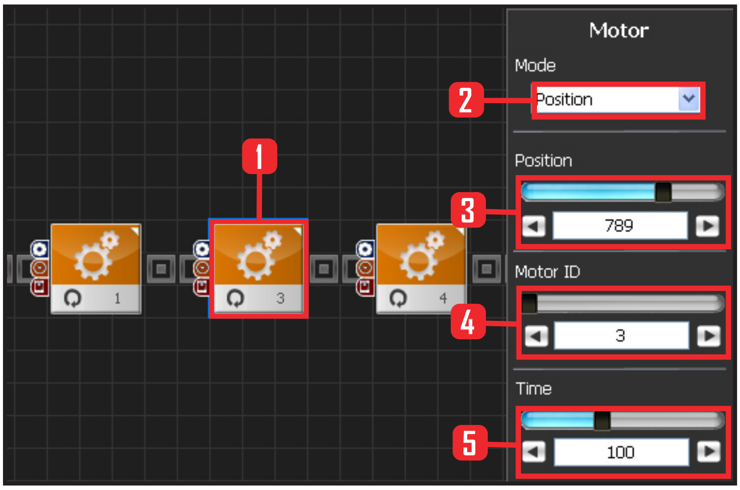
Position: 235

Motor ID: 1

Time: 100

10 Setup Motor ID 1 (Right Arm)

Select Mode : Postion.
Set Position : 235. 235 lowers the horizonally stretched arm to vertical down position.
Set Motor ID : 1. Right upper arm motor connected to the should has motor ID 1.
Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s.



Motor

Mode: Position

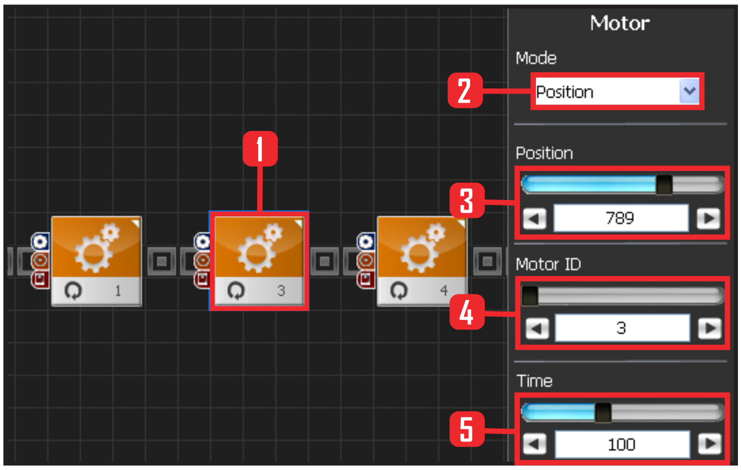
Position: 789

Motor ID: 3

Time: 100

11 Setup Motor ID 3(Left Shoulder)

Select Motion 〉 Motor.
Select Mode : Position.
Set Position : 789. 789 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.
Set Motor ID : 0. Left shoulder motor has ID 3
Set Time : 100. Motor will turn to the desired angle in approximately 1.12s.



Motor

Mode: Position

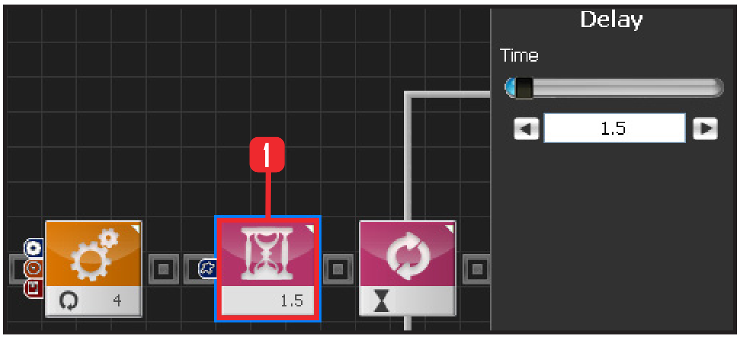
Position: 789

Motor ID: 3

Time: 100

12 Setup Motor ID 4 (Left Arm)

Select Mode : Postion.
Set Position : 789. 789 lowers the horizonally stretched arm to vertical down position.
Set Motor ID : 4. Right upper arm motor connected to the should has motor ID 4.
Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s.


Delay

Time: 1.5

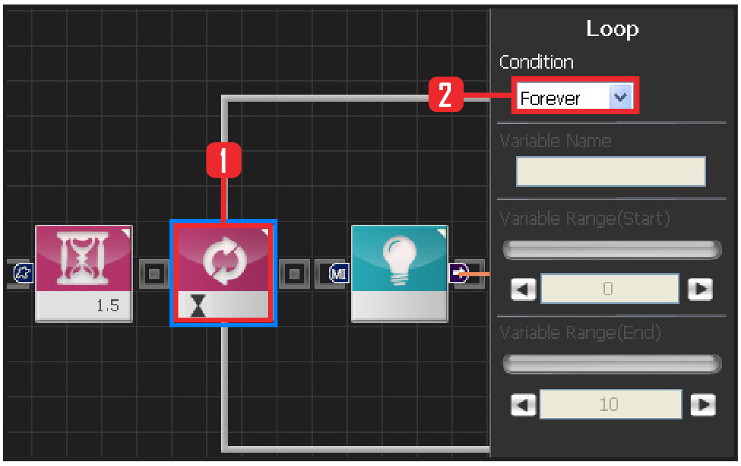
13 Delay

This section makes the robot wait untill the robot is at attention posture and ready to run the next module.

Select Flow 〉 Delay module.

Set Time : 1.5 . Unit is in seconds.

Delay 1.5s.


Loop

Condition: Forever

Variable Name

Variable Range(Start): 0

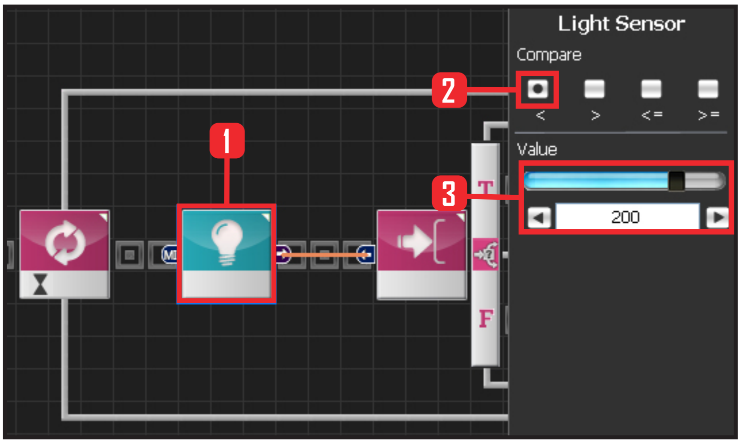
Variable Range(End): 10

14 Loop

Select Flow 〉 Loop module.

Select Condition: Forever.

Infinite loop.


Light Sensor

Compare: ● < □ > □ <= □ >=

Value: 200

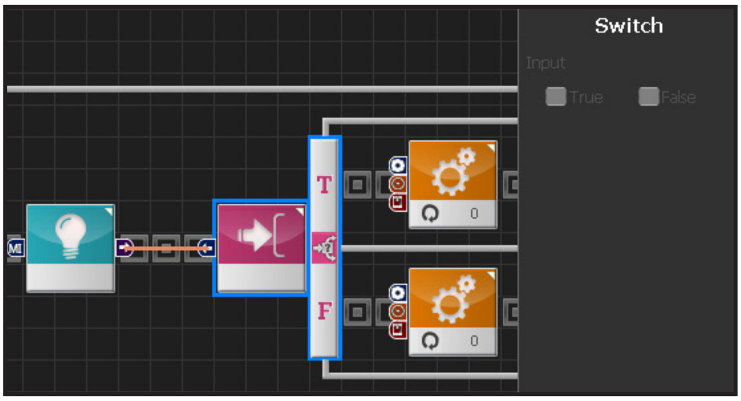
15 Light Sensor

Select Sensor 〉 Light module.

Select Compare : 〈 . Smaller than certain value.

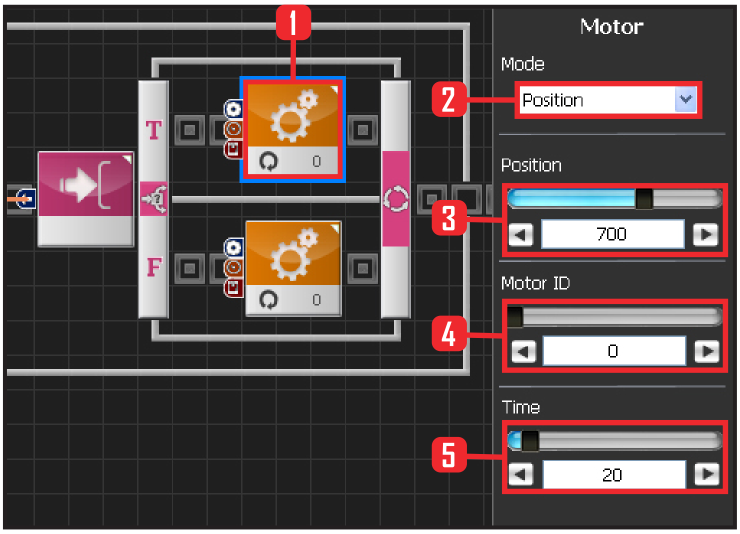
Set Value : 200. Luminosity 200.

Module output is True if the luminosity is smaller than 200 and False if larger than 200.



16 Switch IF Conditional Statement

Run applicable section depending on True or False value.

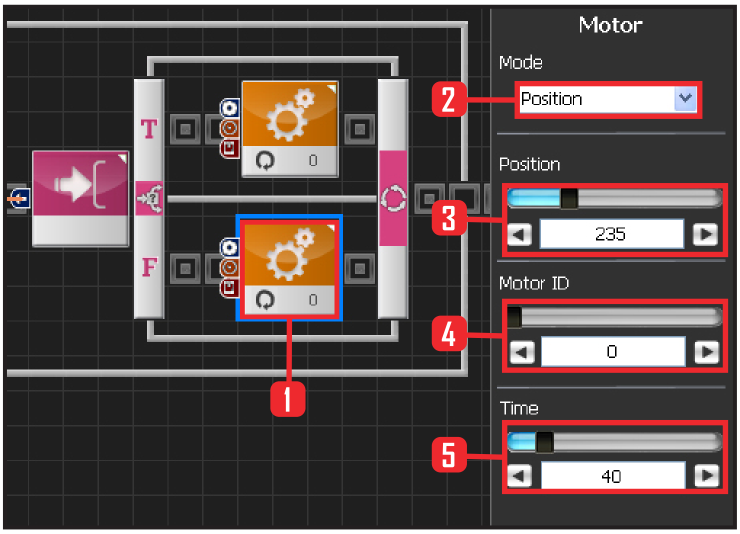


Motor
Mode: Position
Position: 700
Motor ID: 0
Time: 20

17 Setup Motor ID 0(Right Shoulder)

Lift right arm if the luminosity is less than 200(True), If the luminosity is greater than 200(False) keep current posture with the arm lowered.

Select Motion 〉 Motor.
Select Mode : Position .
Set Position : 700 . 700 lifts the arm.
Set Motor ID : 0 . Right shoulder moto ID is 0
Set Time : 20 .

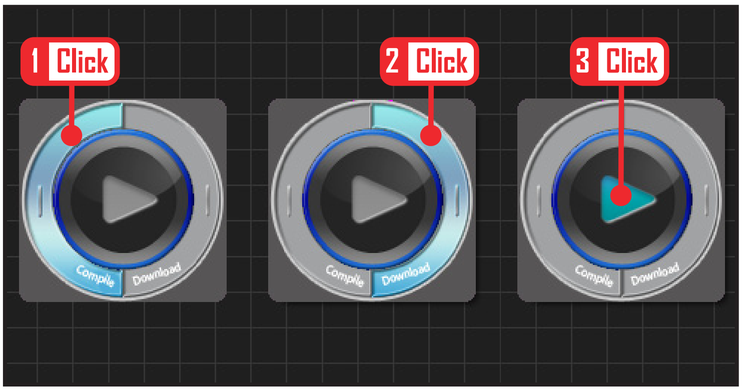


Motor
Mode: Position
Position: 235
Motor ID: 0
Time: 40

18 Set Motor ID 0(Shoulder)

Uncover the controller cds sensor and the robot will go back to the attention posture.

Select Motion〉Motor.
Select Mode : Position .
Set Position : 235 . 235 lowers the arm to the side.
Set Motor ID : 0 . Right shoulder motos has ID 0.
Set Time : 40. Arm comes down at slower pace than when it was going up.



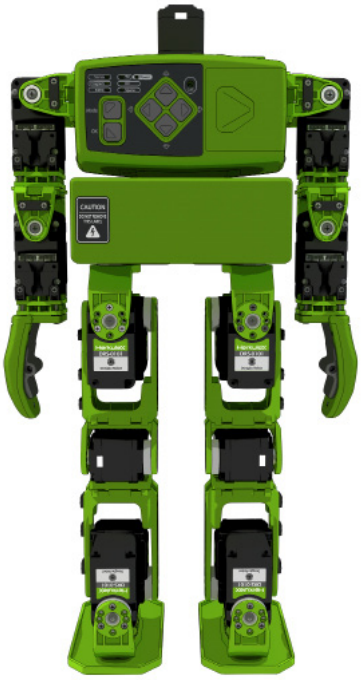
18 Download

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

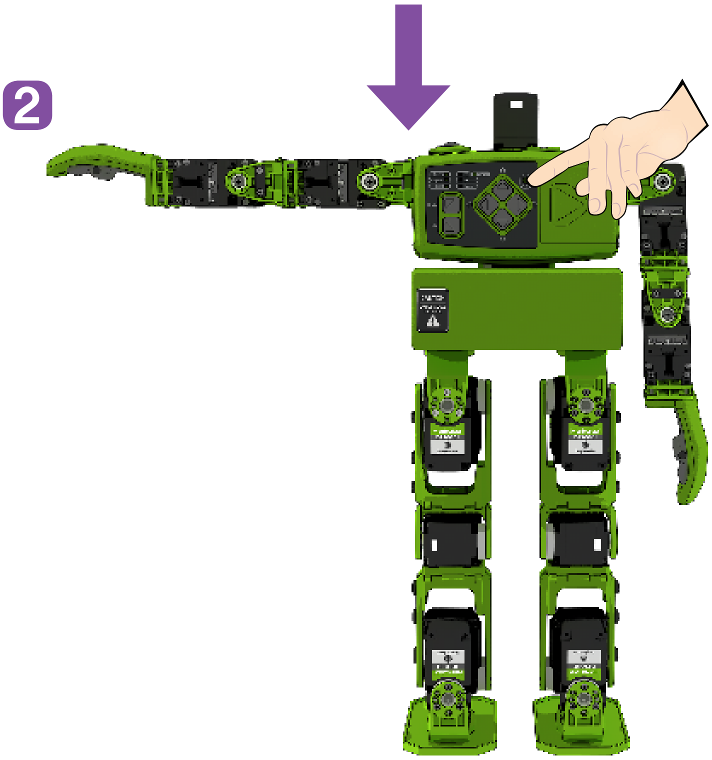
19 Robot Motion

Robot is at attention posture under the bright light.
Robot will lift the right arm when the controller cds is covered.
Robot will lower the arm when the cds is uncovered.

1



2

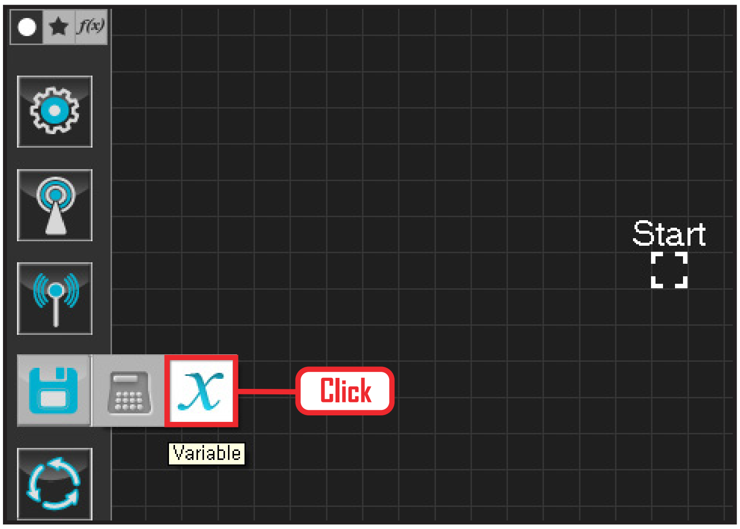


Sound Sensor Example Step by Step

Example Description

Sound Sensor is located inside the DRC controller on both sides.

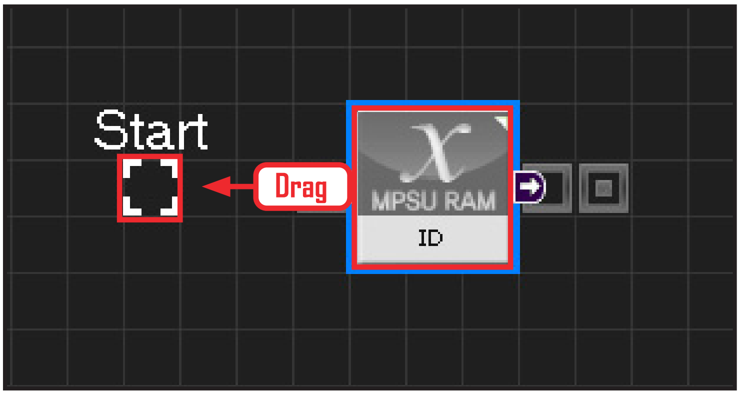
This example will make the robot lift the left arm with left side clap and right arm with the right side clap.



01 Assign Variable

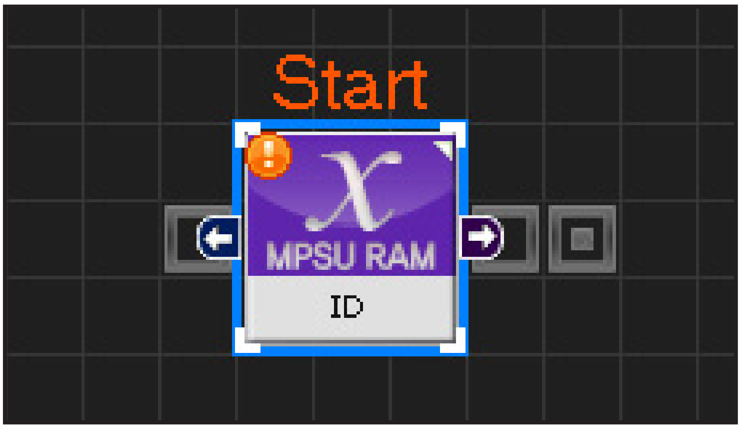
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

Click Data 〉 Variable module.



02 Start

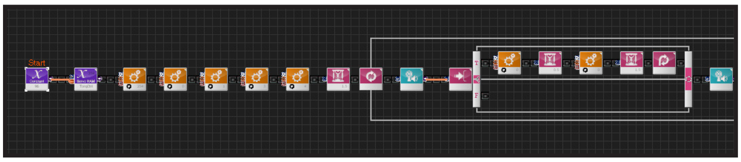
Click and drag the connecting line located at left side of the module to the Start Point and dock



03 Start Programming

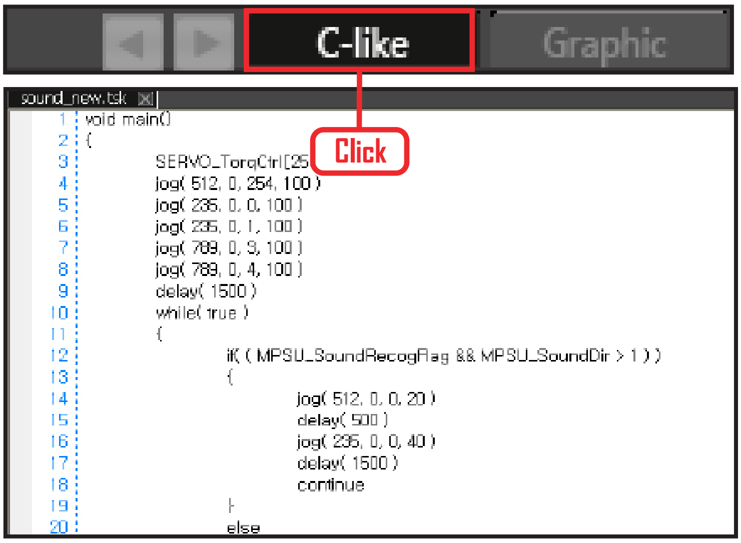
When the module and the Start Point is docked properly, module will become active and change color as seen in the photo to the left. This means programming has started..

04 Entire Program



Use the sound sensor to operate robot motors.

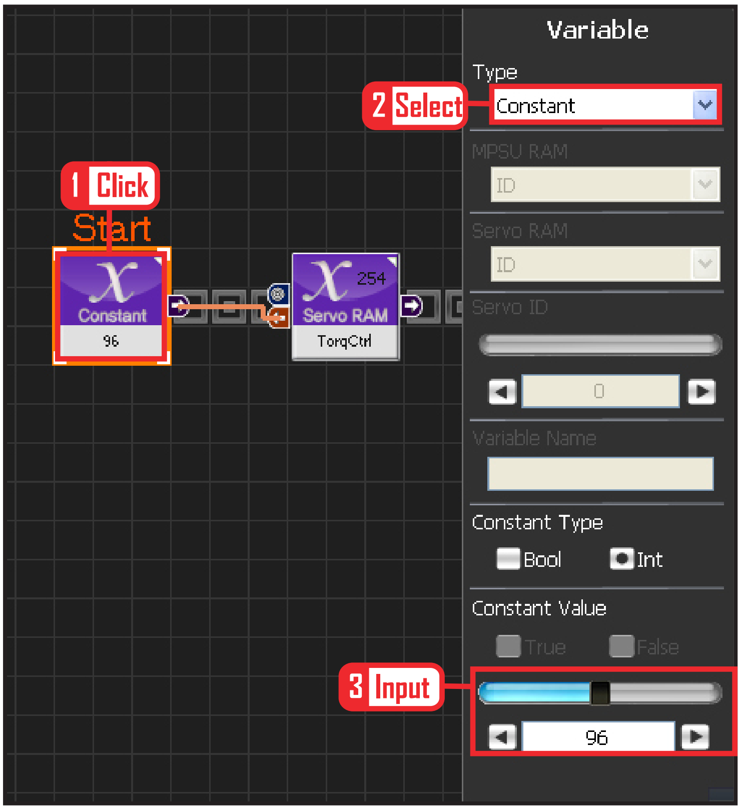
05 Viewing C-Like



```
void main()
{
        SERVO_TorqCtrl[25
        jog( 512, 0, 254, 100 )
        jog( 235, 0, 0, 100 )
        jog( 235, 0, 1, 100 )
        jog( 789, 0, 3, 100 )
        jog( 789, 0, 4, 100 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_SoundRecogFlag && MPSU_SoundDir > 1 ) )
                {
                        jog( 512, 0, 0, 20 )
                        delay( 500 )
                        jog( 235, 0, 0, 40 )
                        delay( 1500 )
                        continue
                }
                else
```

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 Setup Constant



Variable

Type: Constant

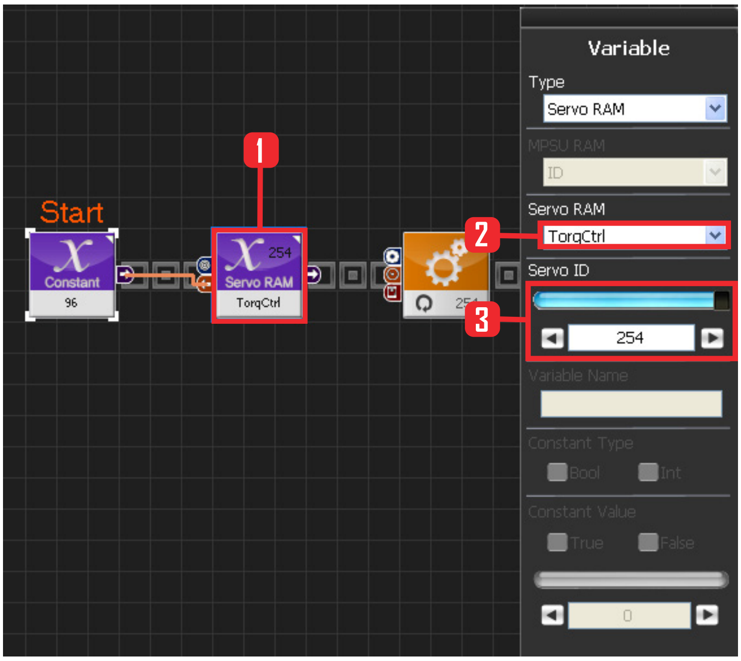
Constant Type: □ Bool ● Int

Constant Value: 96

This section allows the servo motor to operate on it's own.

Select Constant as the Variable Type. In properties, set constant value as 96.

When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next mo–duel through the output connector.



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

Variable Name

Constant Type: □ Bool □ Int

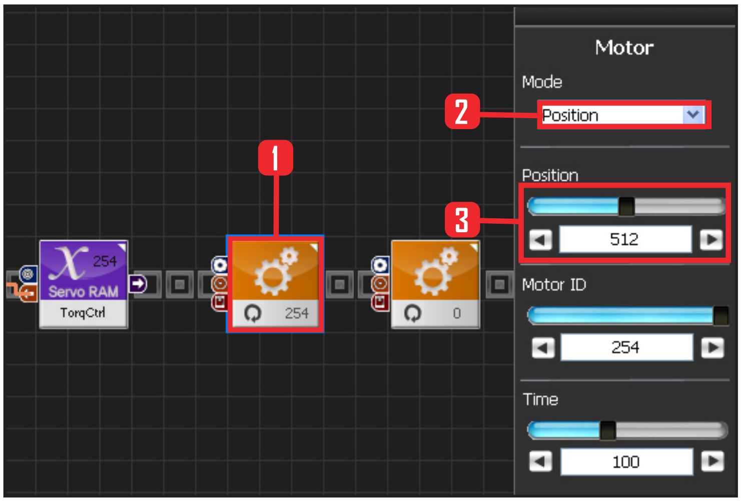
Constant Value: □ True □ False

0

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉 Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos.



Motor

Mode: Position

Position: 512

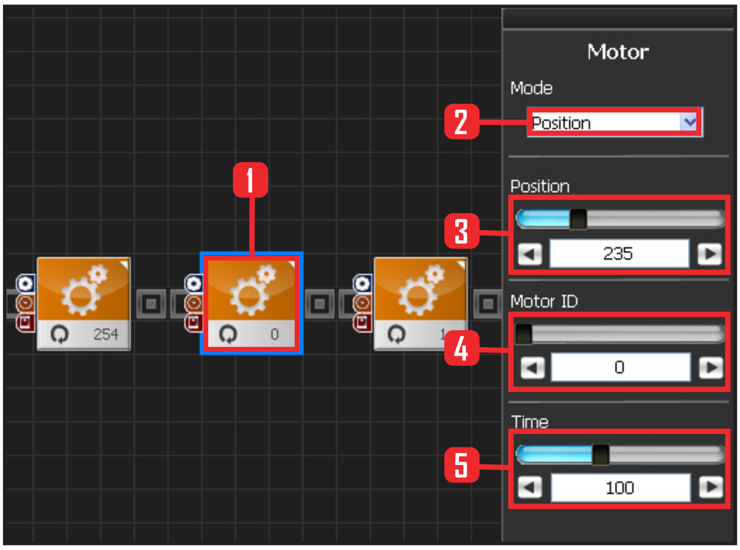
Motor ID: 254

Time: 100

08 Set Angle to All Servos

This section sets all servo motor angles to the center.

Select Motion 〉 Motor.
Select Mode : Positon. adjust angle.
Set Position : 512 . 512 means motor will be sent to the center
Set Motor ID : 254 . 254 means it will be applied to all connected servos.
Set Time : 100 . 1 unit = 11.2ms. 100 units would be approximately 1.12s.
It means motors will be positioned at the desired angle in 1.12s.



Motor

Mode: Position

Position: 235

Motor ID: 0

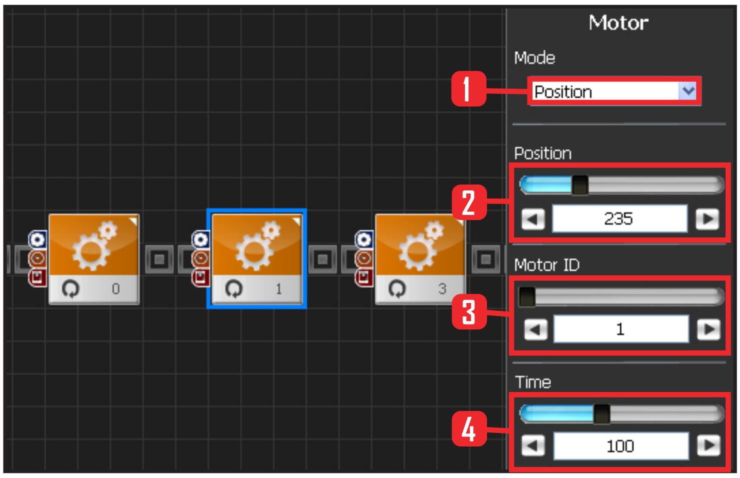
Time: 100

09 Setup Motor ID 0 (Right Shoulder)

Creating attention posture (Basic Posture)

When all robot motors are aligned to the center, humanoid robot arms will be stretched out to the side. Setup below lowers one arm to the side of the body.

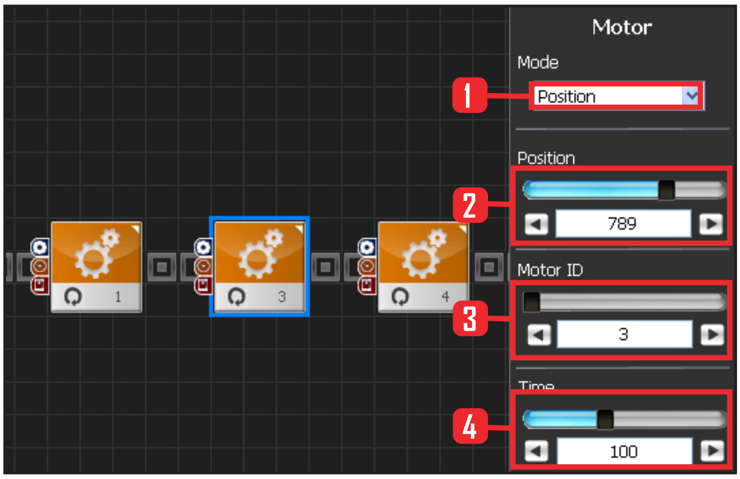
Select Motion 〉 Motor .
Select Mode : Position.
Set Position : 235. 235 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.
Set Motor ID : 0. Right shoulder motor has ID 0
Set Time : 100. Motor will turn to the desired angle in approximately 1.12s.



Motor
Mode: Position
Position: 235
Motor ID: 1
Time: 100

10 Setup Motor ID 1 (Right Arm)

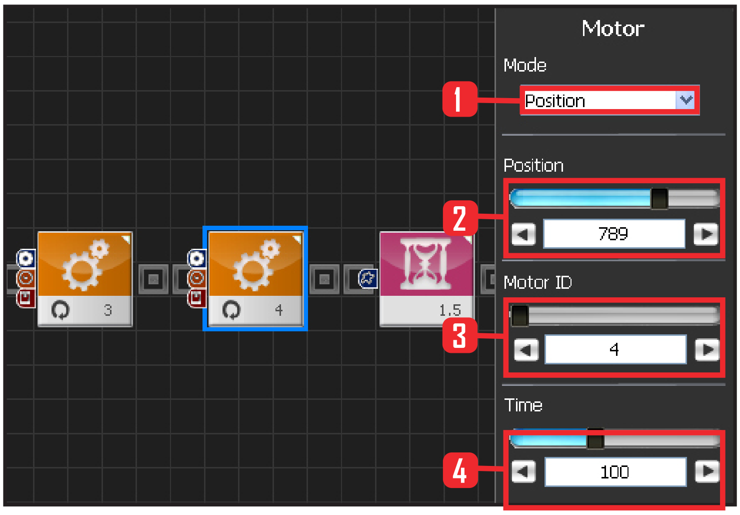
Select Mode : Postion.
Set Position : 235. 235 lowers the horizonally stretched arm to vertical down position.
Set Motor ID : 1. Right upper arm motor connected to the should has motor ID 1.
Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s..



Motor
Mode: Position
Position: 789
Motor ID: 3
Time: 100

11 Setup Motor ID 3(Left Shoulder)

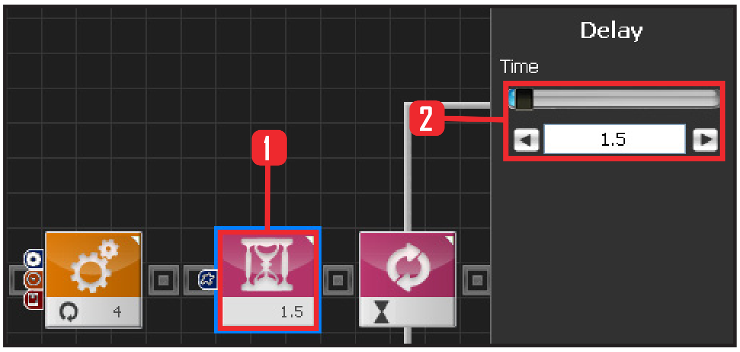
Select Motion 〉 Motor .
Select Mode : Position.
Set Position : 789. 789 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.
Set Motor ID : 0. Left shoulder motor has ID 3
Set Time : 100. Motor will turn to the desired angle in approximately 1.12s..



Motor
Mode: Position
Position: 789
Motor ID: 4
Time: 100

12 Setup Motor ID 4(Left Arm)

Select Mode : Postion.
Set Position : 789. 789 lowers the horizonally stretched arm to vertical down position.
Set Motor ID : 4. Right upper arm motor connected to the should has motor ID 4.
Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s.



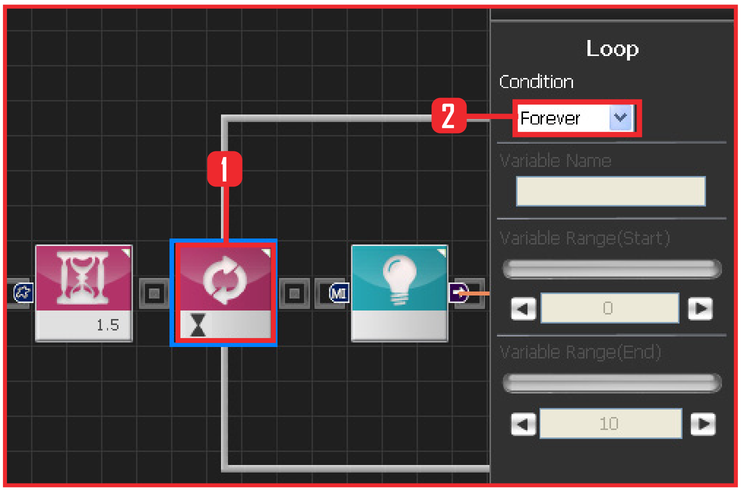
13 Delay

This section makes the robot wait untill the robot is at attention posture and ready to run the next module.

Select Flow 〉 Delay module.

Set Time : 1.5 . Unit is in seconds.

Delay 1.5s.

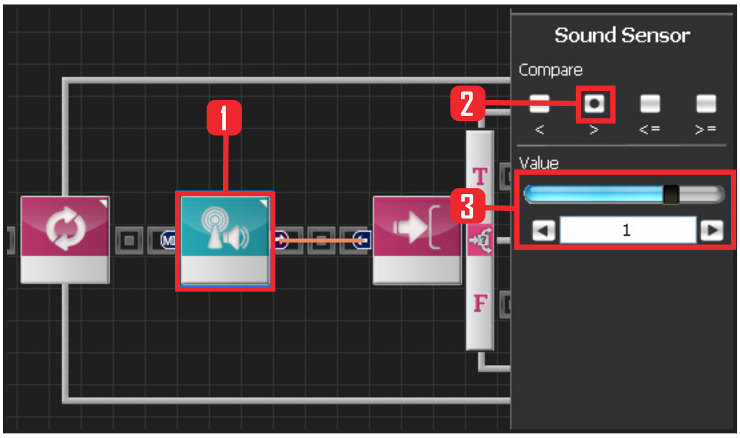


14 Loop

Select Flow 〉 Loop module.

Select Condition: Forever.

Infinite loop.



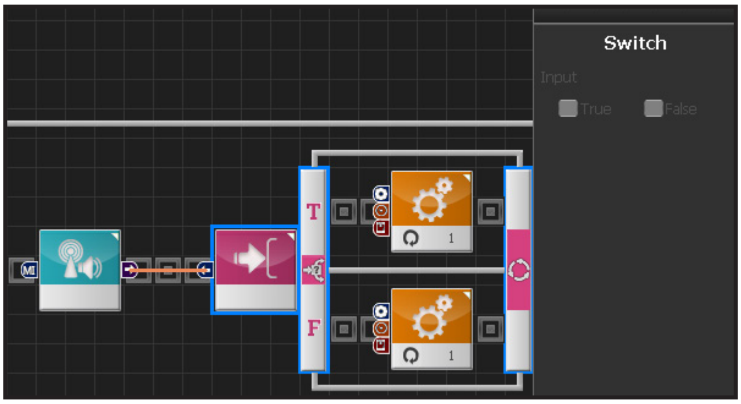
15 Sound Sensor

Select Sensor 〉 Sound Sensor module.

Select Compare : 〉. Larger than certain value.

Set Value : 0 . Range of the sound loaction is from –2 to 2. Negative number denotes sound is from the left and the Positive number from the right.

Value 〉 0 denotes that sound is from the right. If the detected sound is from the right side, Output is True or False otherwise.

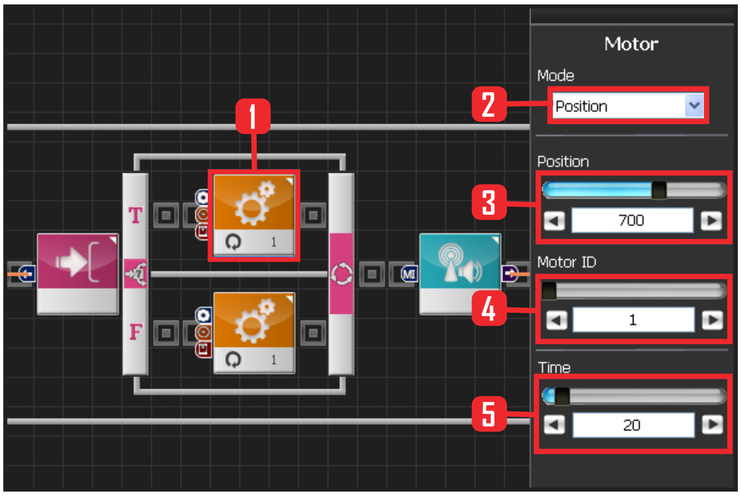

Switch

Input

□ True □ False

16 Switch IF Conditional Statement

Run applicable section depending on True or False value.
True if the sound is from the right or False otherwise.


Motor

Mode: Position

Position: 700

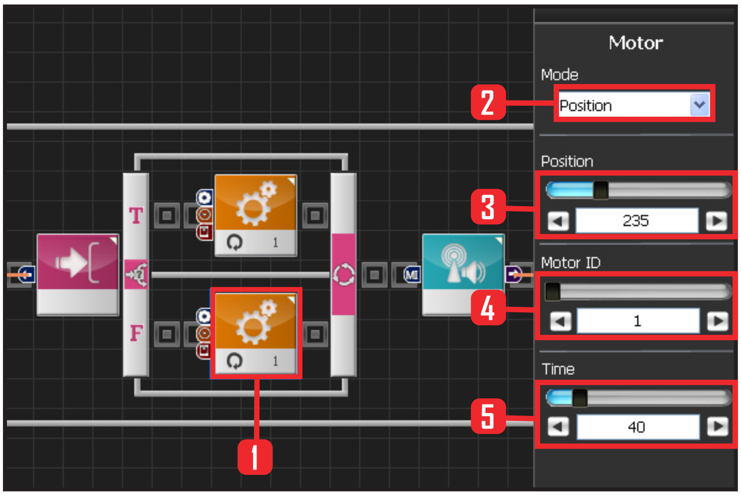
Motor ID: 1

Time: 20

17 Setup Motor ID 1(Right Arm)

True if sound heard from the right side. Robot will lift right arm.

Select Motion 〉Motor .
Select Mode : Position .
Set Position : 700 . 700 lifts the right arm.
Set Motor ID : 1 . Upper right arm motor ID is 1.
Set Time : 20 .


Motor

Mode: Position

Position: 235

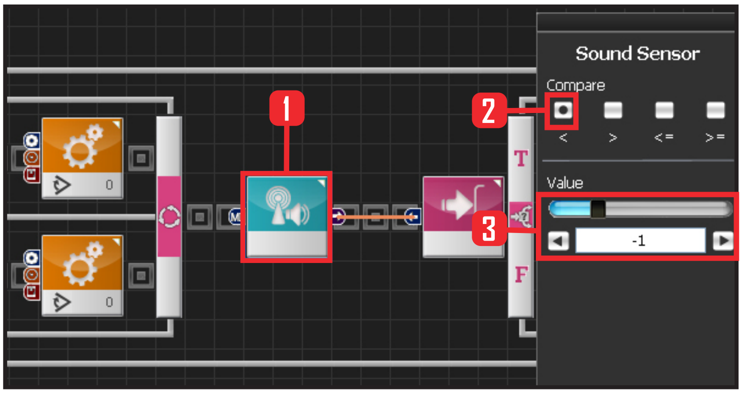
Motor ID: 1

Time: 40

18 Setup Motor ID 1(Right Arm)

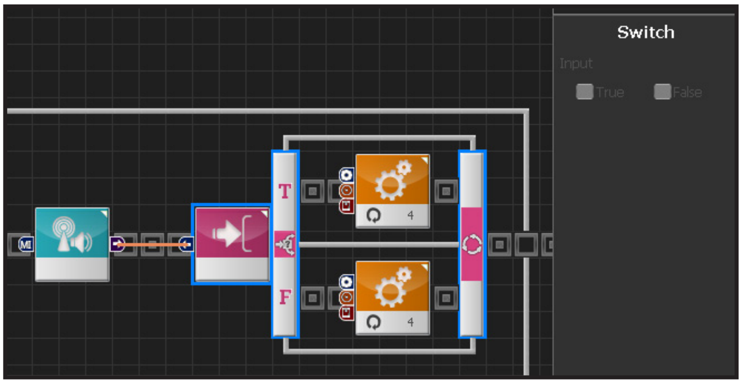
False if no sound is detected or if the sound is from different location. Maintain attention posture with arms lowered to the side.

Select Motion 〉Motor.
Select Mode : Position.
Set Position : 235 . 235 maintains attention pos–ture. Lowers the arm to the side if it was lifted up.
Set Motor ID : 1 . Upper right arm motor ID is 1.
Set Time : 40 . Arm comes down at slower pace than when it was going up.



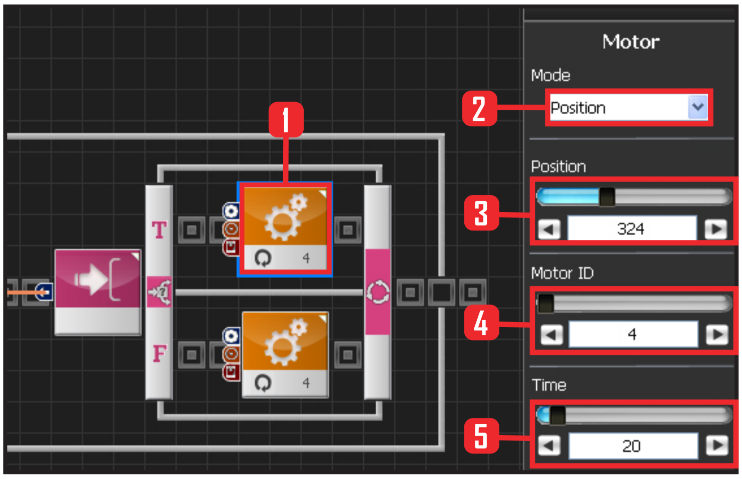
19 Sound Sensor

Select Sensor 〈 Sound Sensor module.
Select Compare : 〈. Larger than certain value.
Set Value : 0 . Range of the sound loaction is from –2 to 2. Negative number denotes sound is from the left and the Positive number from the right.
Value 〈 0 denotes that sound is from the left. If the detected sound is from the left side, Output is True or False otherwise.



20 Switch IF Conditional Statement

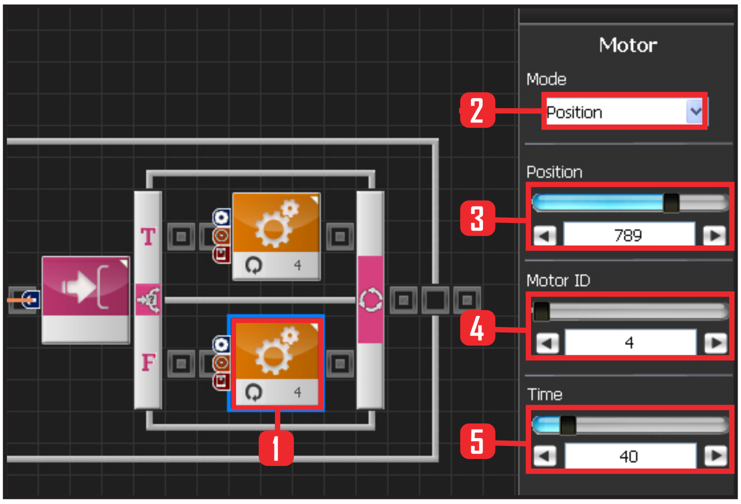
Run applicable section depending on True or False value.
True if the sound is from the left or False otherwise.



21 Setup Motor ID 4 (Left Arm)

True if sound heard from the left side. Robot will lift left arm.

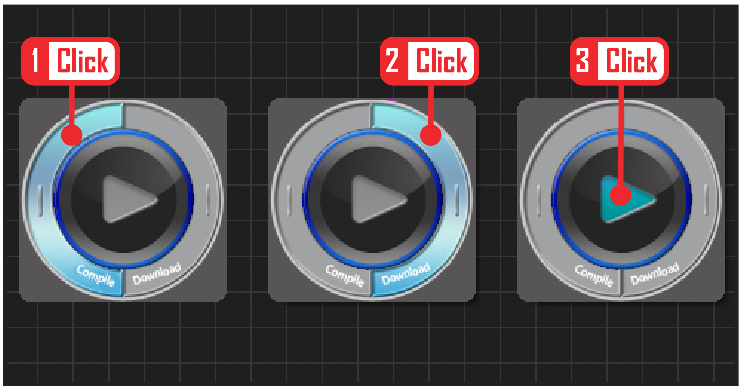
Select Motion 〉 Motor .
Select Mode : Position .
Set Position : 324 . 324 lifts the left arm.
Set Motor ID : 4 . Upper left arm motor ID is 4.
Set Time : 20 .



22 Setup Motor ID 4 (Left Arm)

False if no sound is detected or if the sound is from different location. Maintain attention posture with arms lowered to the side..

Select Motion 〉 Motor.
Select Mode : Position.
Set Position : 789 . 789 maintains attention posture. Lowers the arm to the side if it was lifted up.
Set Motor ID : 4. Upper left arm motor ID is 4.
Set Time : 40 . Arm comes down at slower pace than when it was going up.



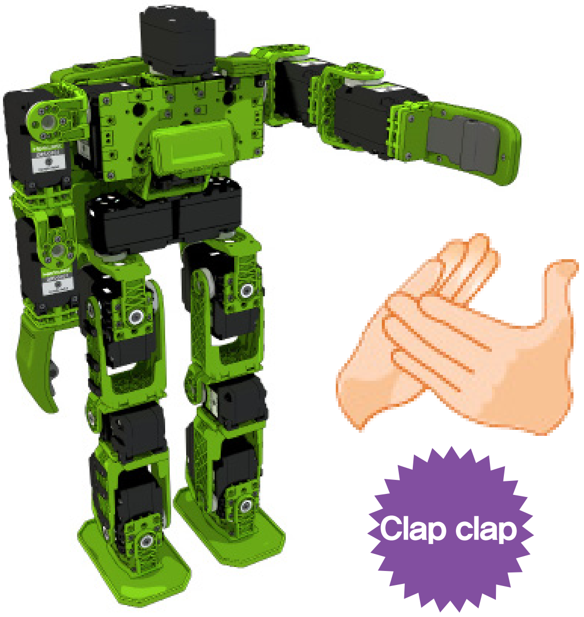
23 Compile, Download, Run

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

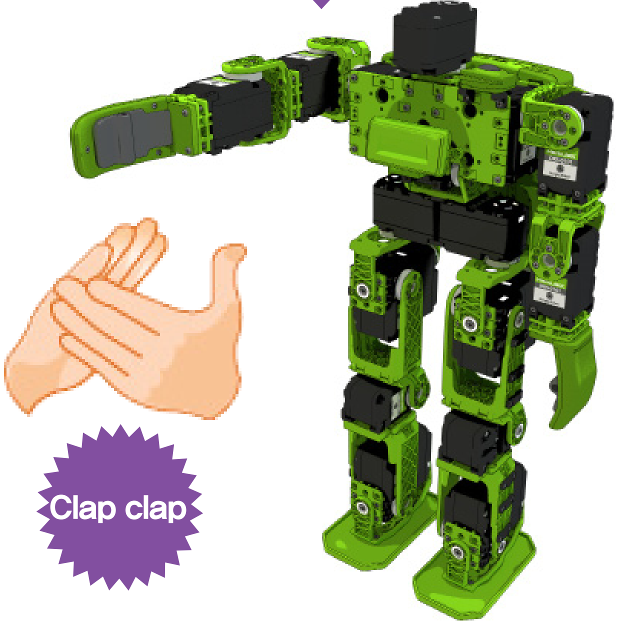
24 Robot Motion

robot will lift the left arm with left side clap and right arm with the right side clap.

1



2



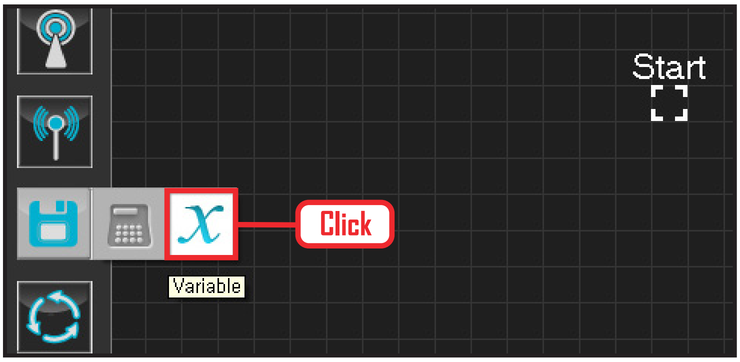
Sound Sensor(indepth) Example Step by Step

Example Description

Sound Sensor is located inside the DRC controller on both sides.

First sound program made the robot lift it's left or right arm in response to the location of the clapping sound.

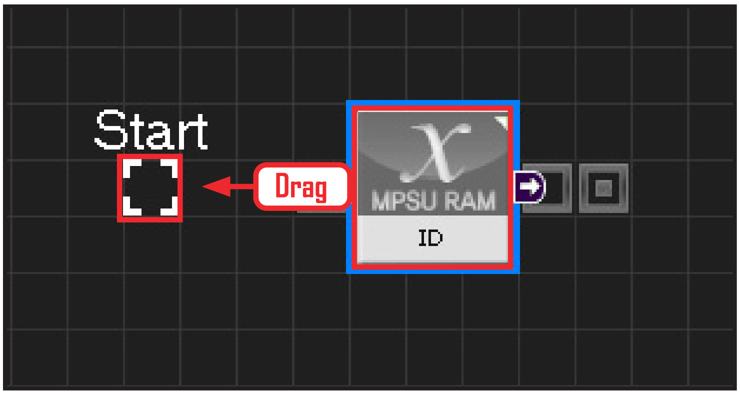
Robot may have difficulty distinquishing the direction of the clap when there is lots of background noise. It may respond by lifting both arms to a single clap from one direction or respond erratically. More refined programming is required to make the robot to respond more reliably regardless of the background noise. Refining the program by forcing a DELAY after registering the first sound so that it will not receive anymore sound input will increase the reliability.



01 Variable Setup

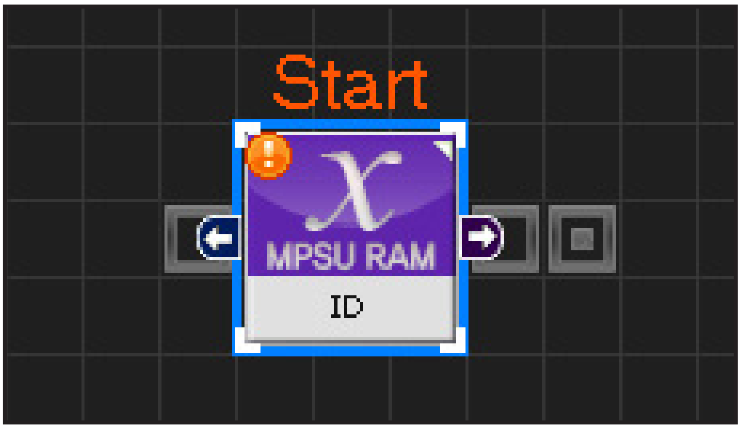
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

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

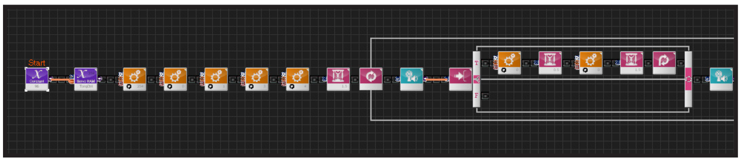


03 Start Programming

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..

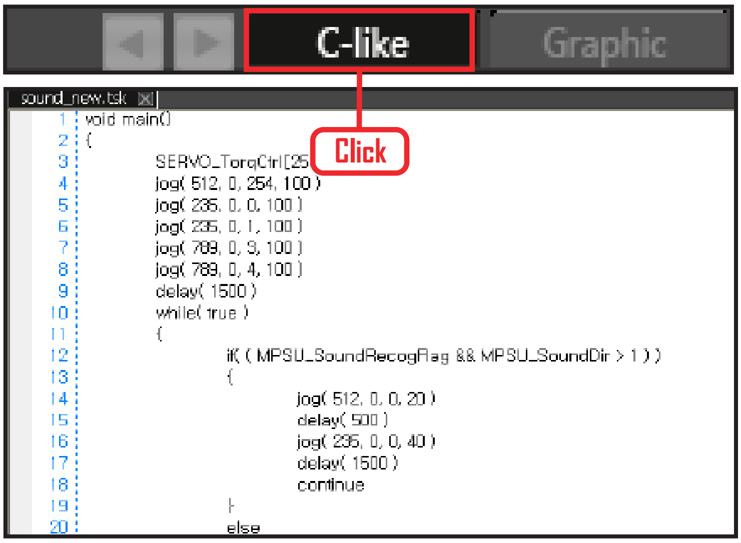
04 Entire Program

Program increases the sensitivity of the sound sensor to make the robot response more reliable.



05 Viewing C-Like

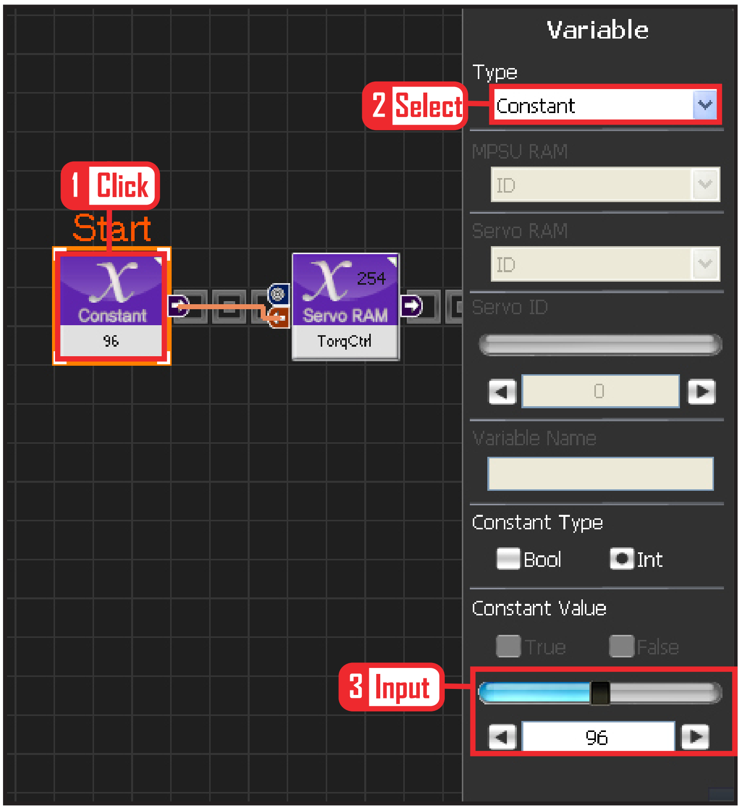
Click the ‘C–like’ tab near the top right and task programming window will open as shown in the photo to the left. This is the task window of the entire program. Codes are very similar to the C language structure so studying the codes will help the user become familiar with the C language structure.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



```
void main()
{
        SERVO_TorqCtrl[25
        jog( 512, 0, 254, 100 )
        jog( 235, 0, 0, 100 )
        jog( 235, 0, 1, 100 )
        jog( 789, 0, 3, 100 )
        jog( 789, 0, 4, 100 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_SoundRecogFlag && MPSU_SoundDir > 1 ) )
                {
                        jog( 512, 0, 0, 20 )
                        delay( 500 )
                        jog( 235, 0, 0, 40 )
                        delay( 1500 )
                        continue
                }
                else
```

06 Setup Constant

This section allows the servo motor to operate on it’s own.
Select Constant as the Variable Type. In properties, set constant value as 96.
When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.

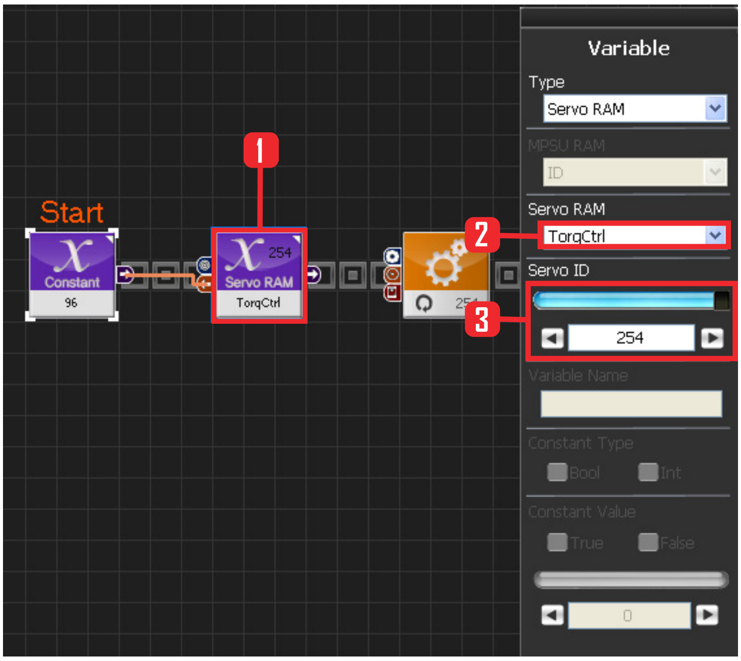


Variable

Type: Constant

Constant Type: ○ Bool ● Int

Constant Value: 96



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

Variable Name

Constant Type: □ Bool □ Int

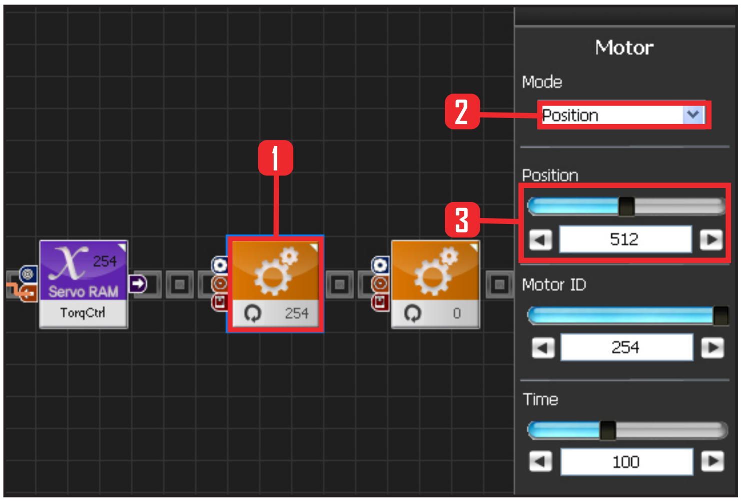
Constant Value: □ True □ False

0

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉 Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos.



Motor

Mode: Position

Position: 512

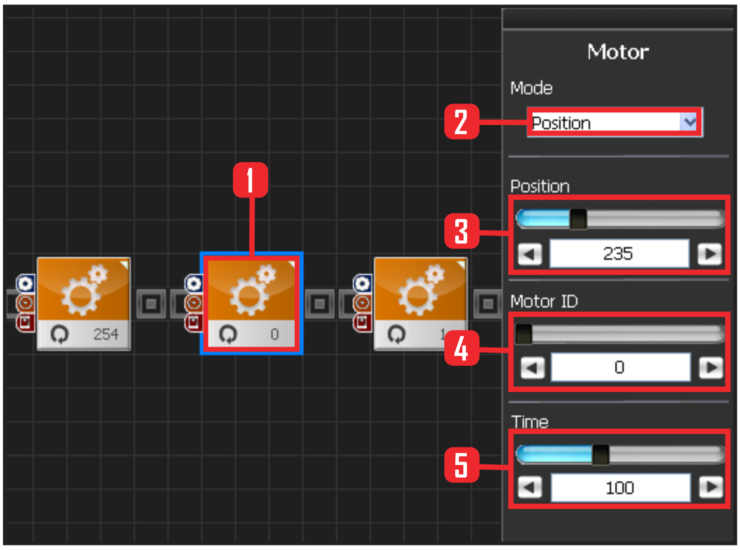
Motor ID: 254

Time: 100

08 Set Angle to All Servos

This section sets all servo motor angles to the center.

Select Motion 〉 Motor.
Select Mode : Positon. adjust angle.
Set Position : 512 . 512 means motor will be sent to the center
Set Motor ID : 254 . 254 means it will be applied to all connected servos.
Set Time : 100 . 1 unit = 11.2ms. 100 units would be approximately 1.12s.
It means motors will be positioned at the desired angle in 1.12s.



Motor

Mode: Position

Position: 235

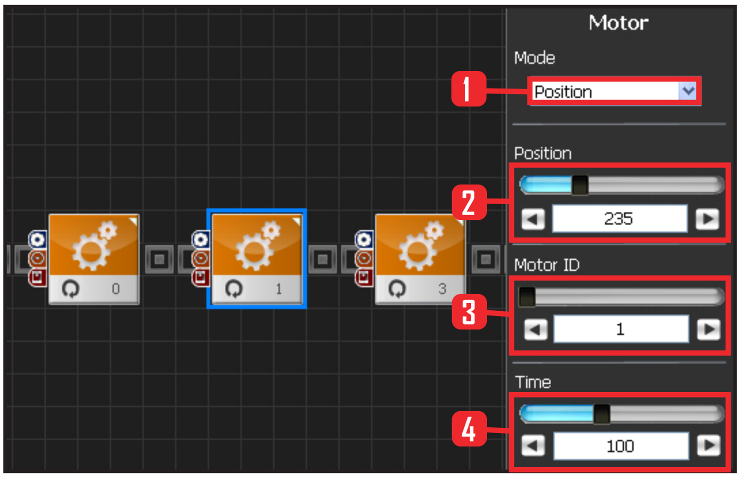
Motor ID: 0

Time: 100

09 Setup Motor ID 0 (Right Shoulder)

Creating attention posture (Basic Posture)
When all robot motors are aligned to the center, humanoid robot arms will be stretched out to the side. Setup below lowers one arm to the side of the body.

Select Motion 〉 Motor .
Select Mode : Position.
Set Position : 235. 235 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.
Set Motor ID : 0. Right shoulder motor has ID 0
Set Time : 100. Motor will turn to the desired angle in approximately 1.12s.



Motor

1. Mode: Position
2. Position: 235
3. Motor ID: 1
4. Time: 100

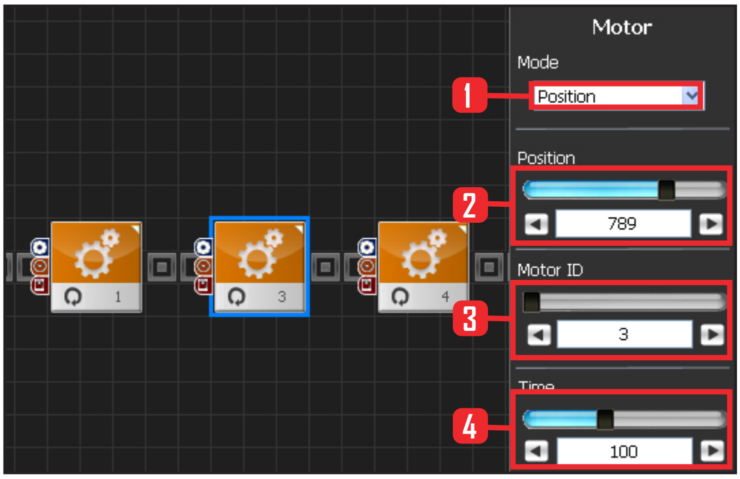
10 Setup Motor ID 1 (Right Arm)

Select Mode : Postion.

Set Position : 235. 235 lowers the horizonally stretched arm to vertical down position.

Set Motor ID : 1. Right upper arm motor connected to the should has motor ID 1.

Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s...



Motor

1. Mode: Position
2. Position: 789
3. Motor ID: 3
4. Time: 100

11 Setup Motor ID 3 (Left Shoulder)

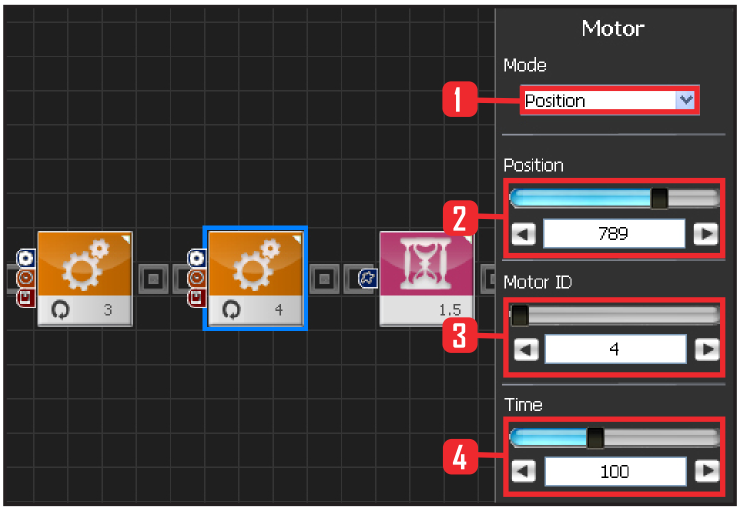
Select Motion 〉Motor .

Select Mode : Position.

Set Position : 789. 789 turns the motor so that that the arm stretched out horizontally will be lowered to vertical down position.

Set Motor ID : 0. Left shoulder motor has ID 3

Set Time : 100. Motor will turn to the desired angle in approximately 1.12s..



Motor

1. Mode: Position
2. Position: 789
3. Motor ID: 4
4. Time: 100

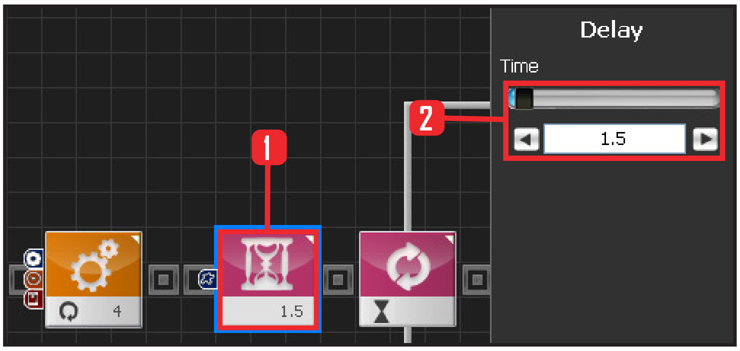
12 Setup Motor ID 4 (Left Arm)

Select Mode : Postion.

Set Position : 789. 789 lowers the horizonally stretched arm to vertical down position.

Set Motor ID : 4. Right upper arm motor connected to the should has motor ID 4.

Set Time : 100 . Motor will turn to the desired angle in apporoximately 1.12s..



Delay

Time

1.5

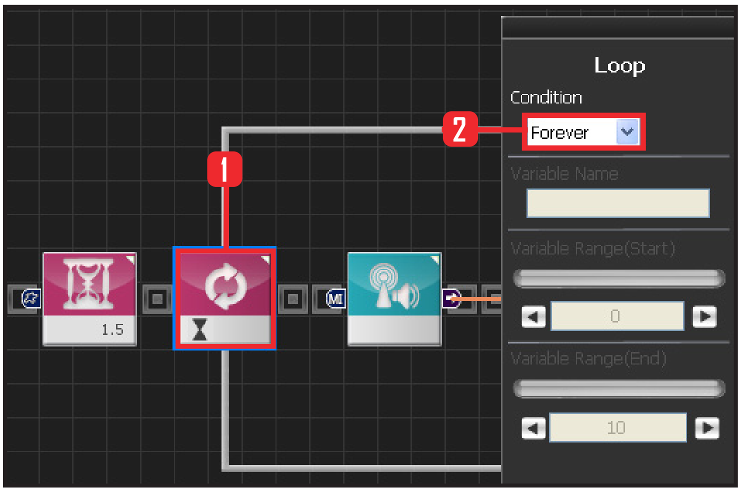
13 Delay

This section makes the robot wait untill the robot is at attention posture and ready to run the next module.

Select Flow 〉 Delay module.

Set Time : 1.5 . Unit is in seconds.

Delay 1.5s.



Loop

Condition

Forever

Variable Name

Variable Range(Start)

0

Variable Range(End)

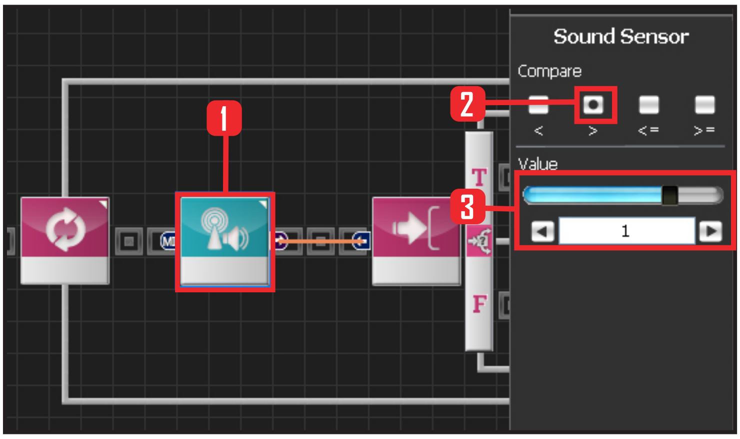
10

14 Loop 반복문

Select Flow 〉 Loop module.

Select Condition: Forever.

Infinite loop.



Sound Sensor

Compare

□ ■ □ □
< > <= >=

Value

1

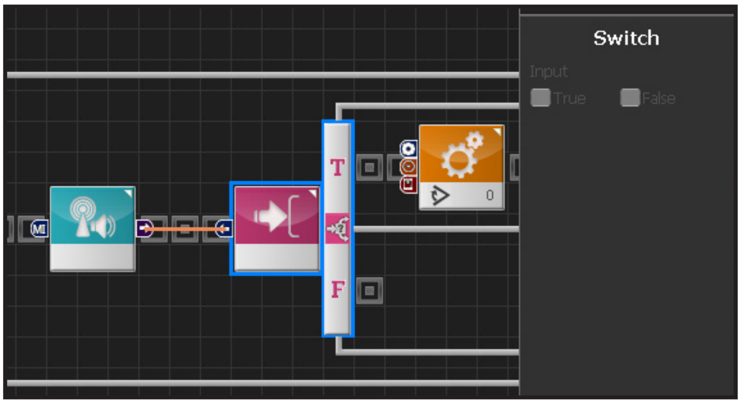
15 Sound Sensor

Select Sensor 〉 Sound Sensor module.

Select Compare : 〉.

Set Value : 1 .

Median sound value is 0. However, setting the value to 1 will decrease the sensitivity so that only the sound larger than 1 (loud noise from the right) will be registered. This will prevent the robot from responding to the background noise or lifting both arms.

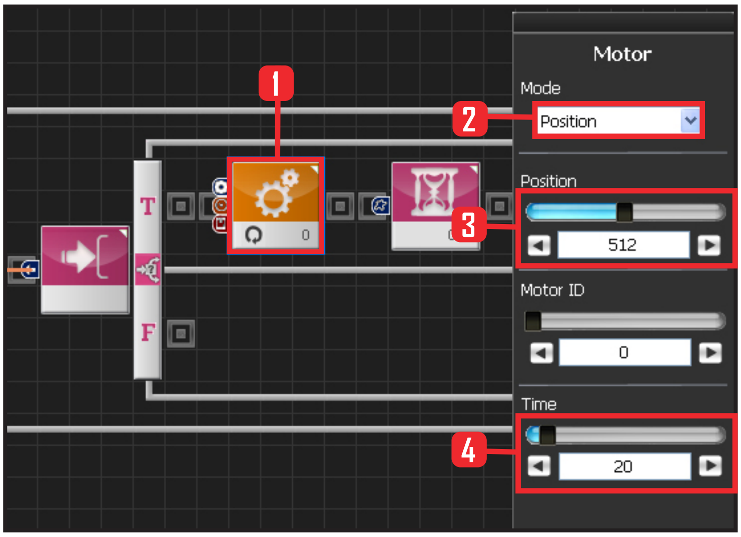

Switch

Input

□ True □ False

16 Switch IF Conditional Statement

Proceed only if the previous condition is True.


Motor

Mode: Position

Position: 512

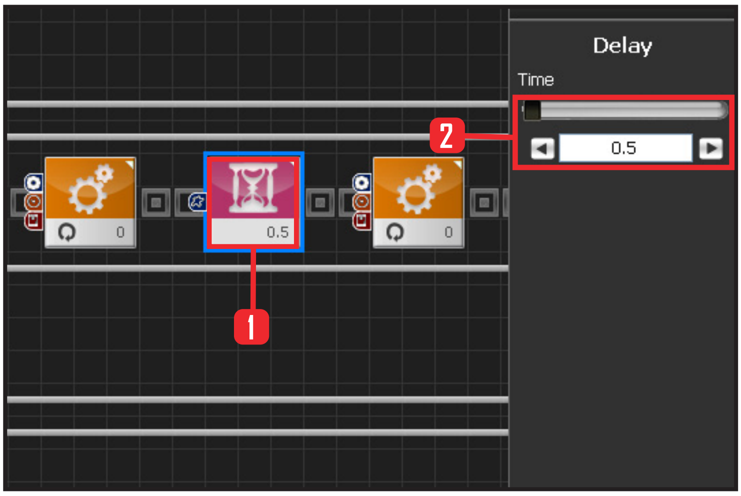
Motor ID: 0

Time: 20

17 Setup Motor ID 0 (Right Shoulder)

Lift right arm if True; the sound loation value is greater than 0. There are times when the other arm may start to move due to background or motor noise. This program prevents the other arm from moving when one arm is already in mototion.

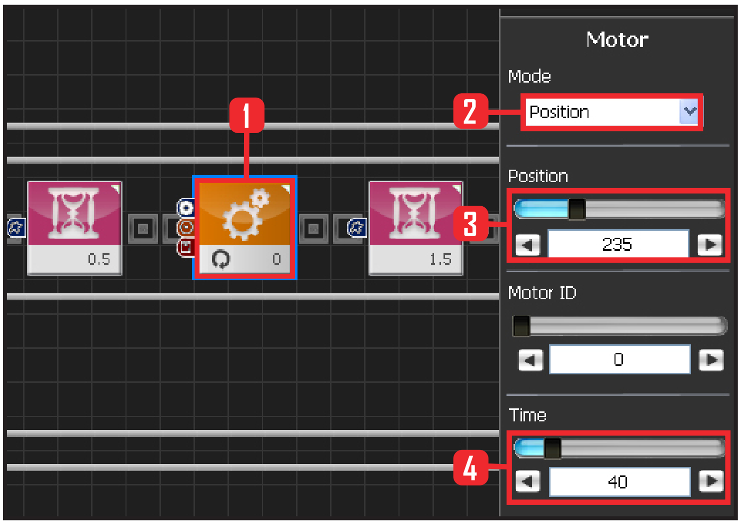
Select Motion > Motor module.
Select Mode : Position .
Set Position : 512. Both arms streched out.
Set Time : 20
Robot arms liift up to 90 degrees angle from the attention posture.


Delay

Time: 0.5

18 Delay

Whle the arm is moving, other arm may start to move or the moving arm may respond again to the background noise. Delay is added to prevent such occurences while the arm is in motion.
No other motion is allowed during the 0.5s of Delay except for the right arm.

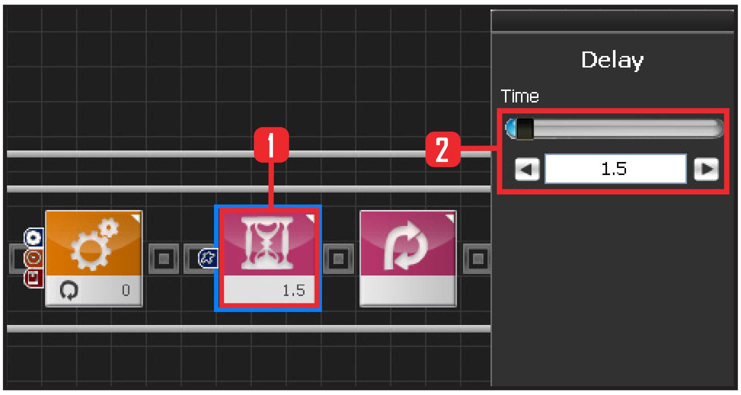


19 Motor ID 0 (Right Shoulder) Return to Attention Posture.

Lower the arm back to attention posture.

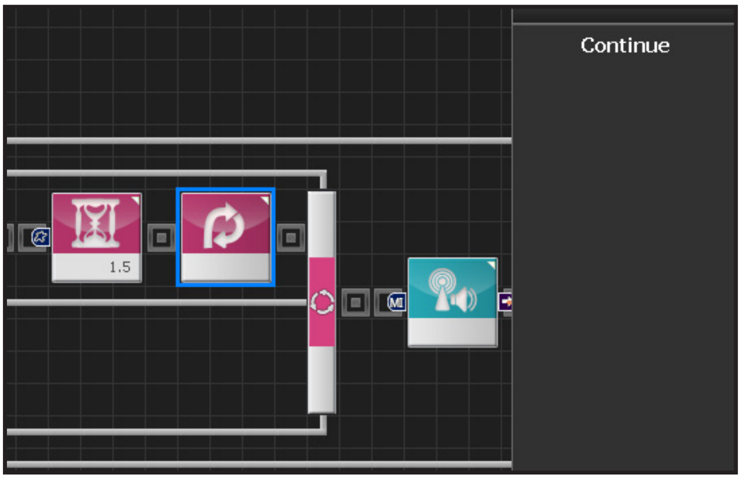
Select Motion 〉 Motor module.
Select Mode : Position .
Set Position :235. Return to attention posture
Set Time:40.

Return right arm to attention posture.



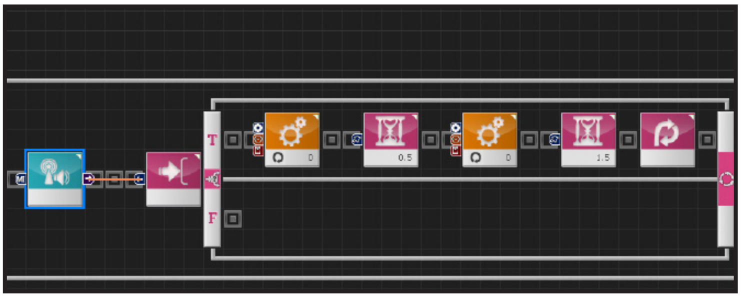
20 Delay

Add Delay to prevent any other motion after returning to attention posture.
When 1.5s Delay value is added. Robot will not move or register sound during the delay. Robot will respond to sound again after the Delay.



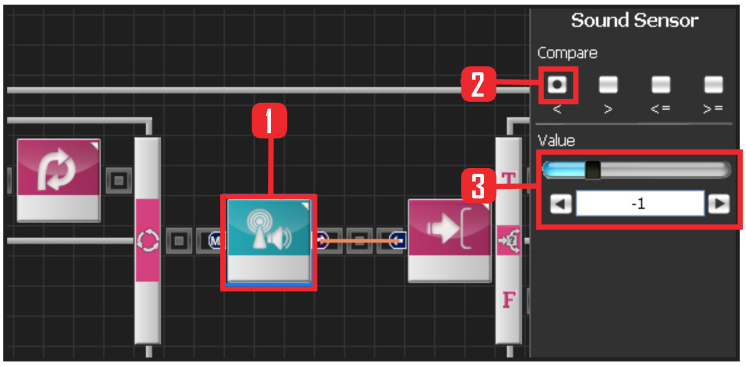
21 Continue

Return to the beginning of the loop after 1.5s Delay.



22 Summary

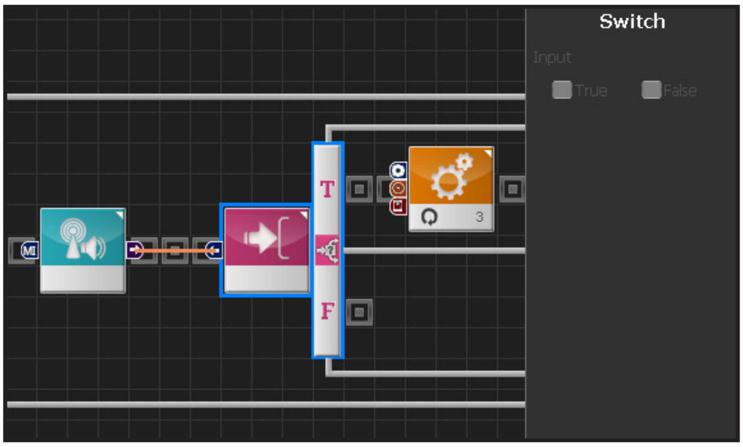
Just completed program blocked certain external stimulus from being registered by the robot. This increased the reliability of the robot response to the sound coming from the right direction.



23 Sound Sensor (2nd)

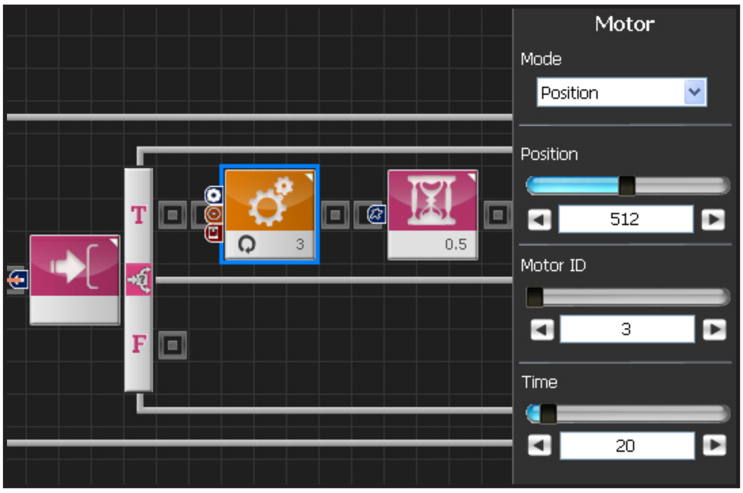
Setup second sound sensor. Left arm will respond to the sound coming from the left.

Select Sensor 〉 Sound Sensor module.
Compare : 〈 .
Value : –1 . Respond when smaller than –1.



24 Switch IF Conditional Statement

Proceed only if the previous condition is True.



Motor

Mode: Position

Position: 512

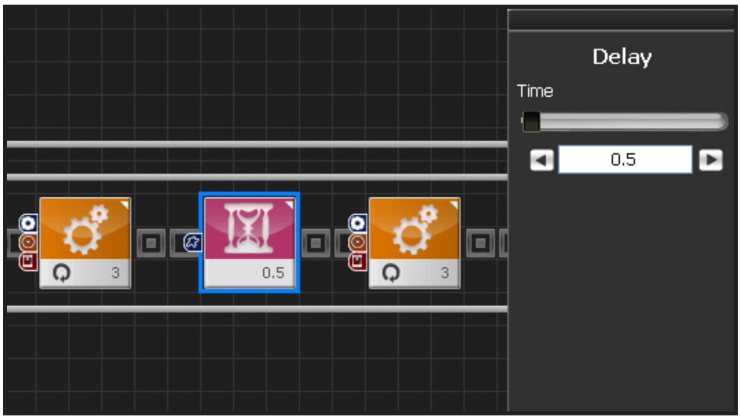
Motor ID: 3

Time: 20

25 Setup Motor ID 3 (Left Shoulder)

True if the sound location value is less than –1.Lift left arm to steched out position.

Select Motion>Motor module.
Set Mode : Position .
Set Position: 512 .
Shoulder angle is 789 when in attention posture. Arm becomes streched out to the side when the angle changes from 789 to 512.
Set Time: 20.

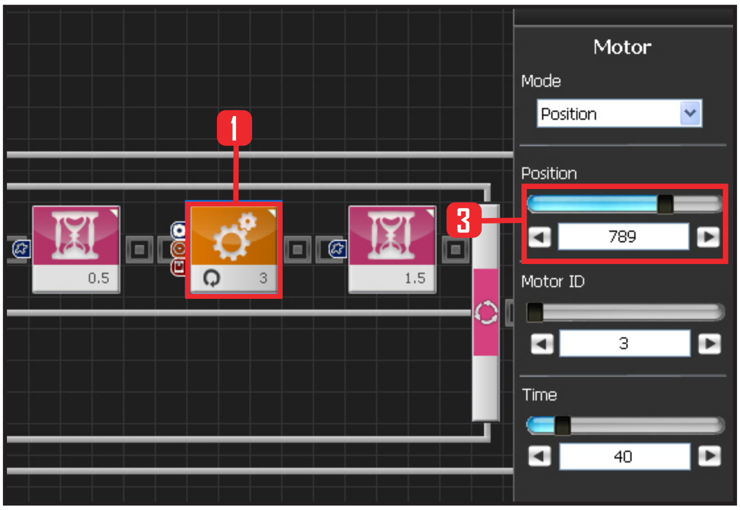


Delay

Time: 0.5

26 Delay

Whle the arm is moving, other arm may start to move or the moving arm may respond again to the background noise. Delay is added to prevent such occurences while the arm is in motion.
No other motion is allowed during the 0.5s of Delay except for the right arm.



Motor

Mode: Position

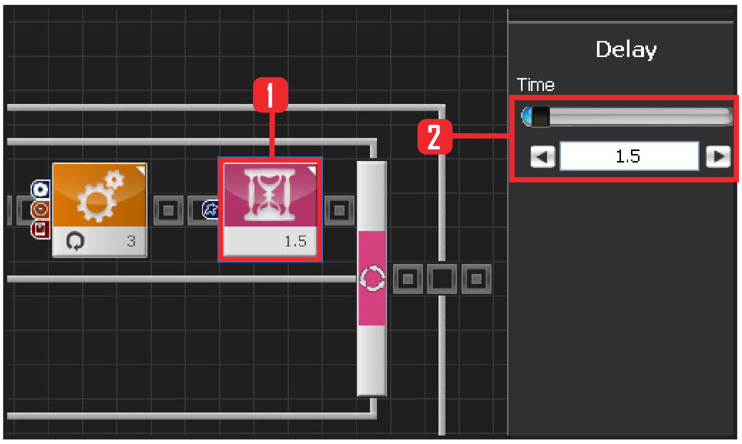
Position: 789

Motor ID: 3

Time: 40

27 Motor ID 3 (Left Shoulder) Return to Attention Posture.

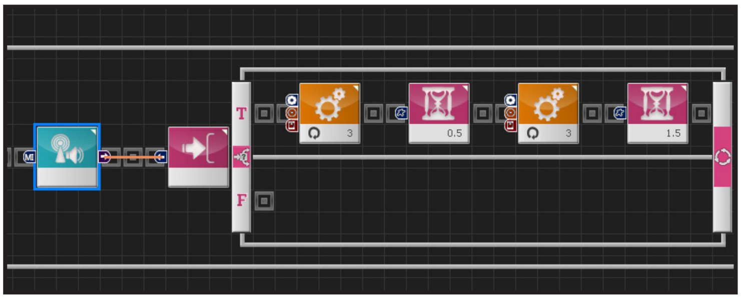
Set Motor ID 3 Position to 789 and return to attention posture.



28 Delay

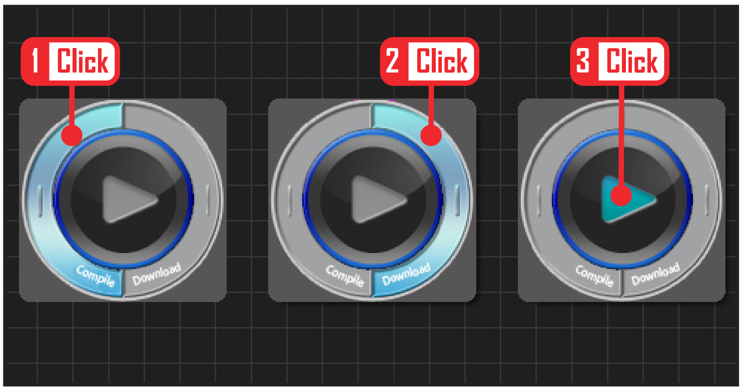
Add 1.5s Delay value to prevent other motions.

Motor ID 3 does not have Continue as Moto ID 0 since this is the end of the loop and progrm will automatically go back tothe beginning of the loop.



29 Left Arm Response

When robot registers a clap from the left, it will lift the left arm and then go back to the attention posture. Delay value makes the robot respond only to the first clap it registers. All other sounds all claps will be ignored. This refinement allows the robot to resopond more reliably in noisy environment.



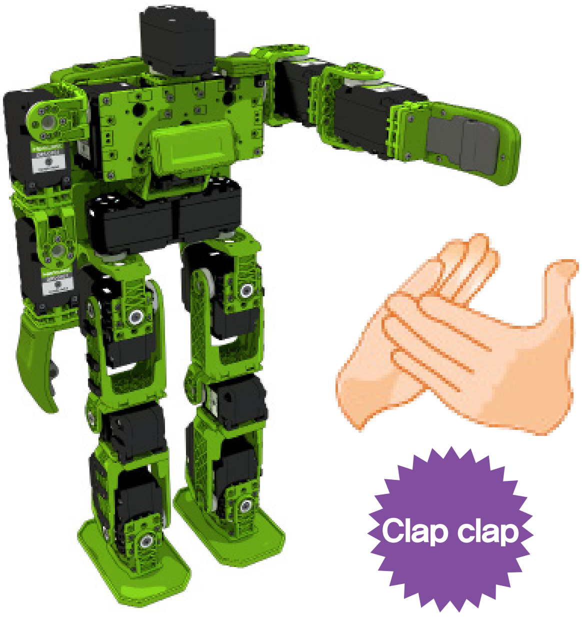
30 Compile, Download, Run

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

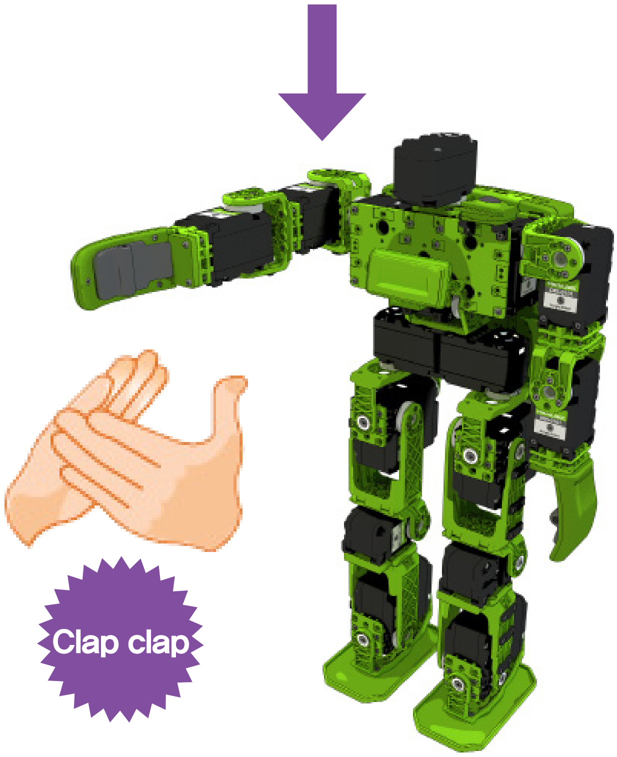
31 Robot Motion

robot will lift the left arm with left side clap and right arm with the right side clap.

1



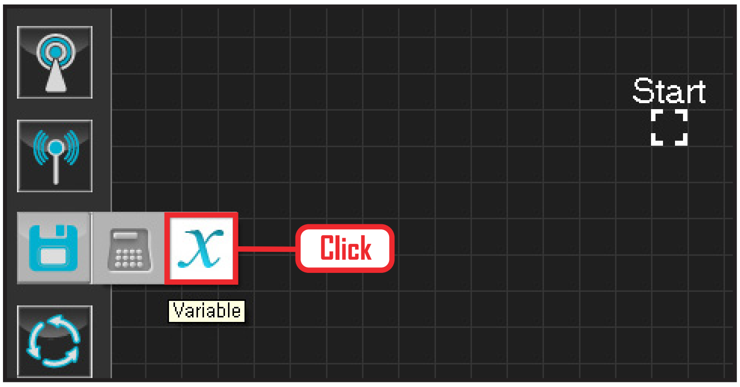
2



Digital Distance Sensor Example Step by Step

Example Description

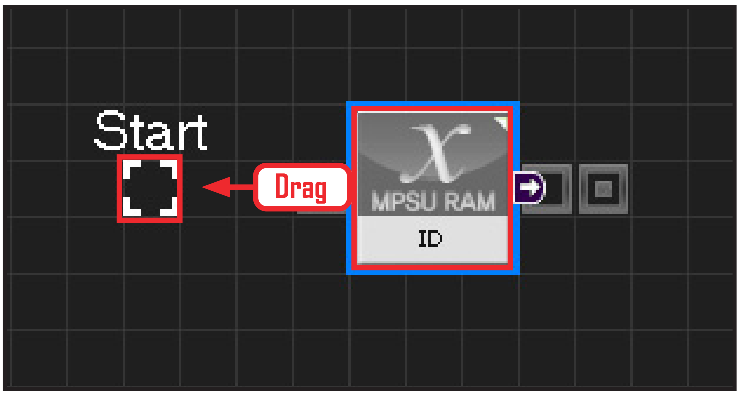
Analog sensor is capable of detecting the actual distance from an object whereas digital sensor uses specific distance as a reference to judge how far or near it is from the reference distance. Robots with wheels use the sensor for cliff detection more often than for object avoidance and humanoid robots with moving legs use the sensor for object avoidance rather than for cliff detection. This example will use the sensor for object detection and avoidance. Compare the program and the result with the analog sensor program. When the robot nears the wall, it will move backwards, change direction and move forward again. This example requires digital distance sensor to be installed at ADC port #1 (left).



01 Assign Variable

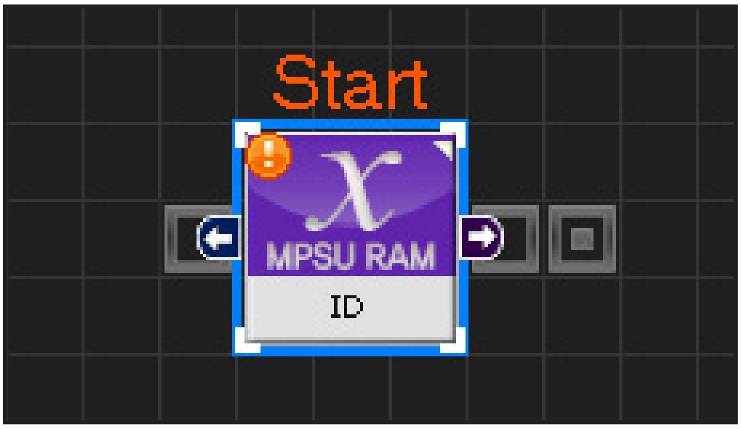
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

Click Data 〉 Variable module.



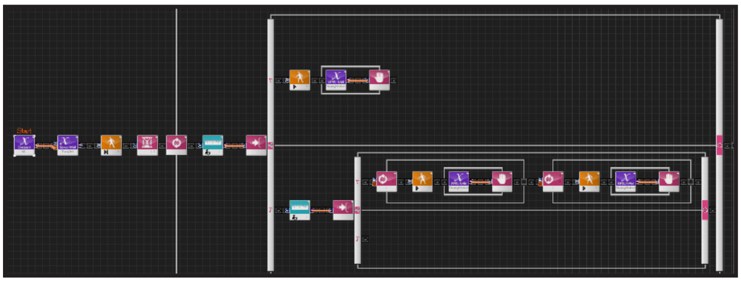
02 Start

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



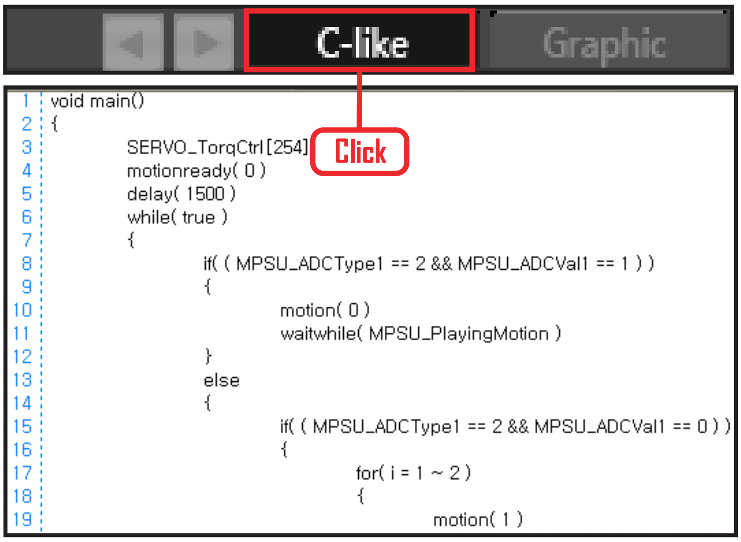
03 Start Programming

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

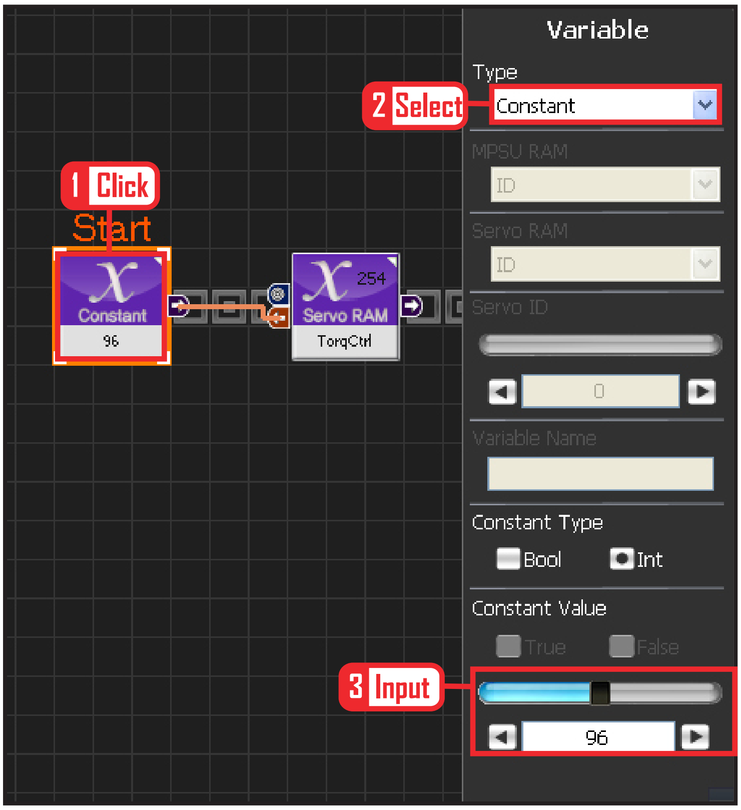
Entire program using the digital sensor.



```
void main()
{
        SERVO_TorqCtrl[254]
        motionready( 0 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 1 ) )
                {
                        motion( 0 )
                        waitwhile( MPSU_PlayingMotion )
                }
                else
                {
                        if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 0 ) )
                        {
                                for( i = 1 ~ 2 )
                                {
                                        motion( 1 )
```

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.



Variable

Type: Constant

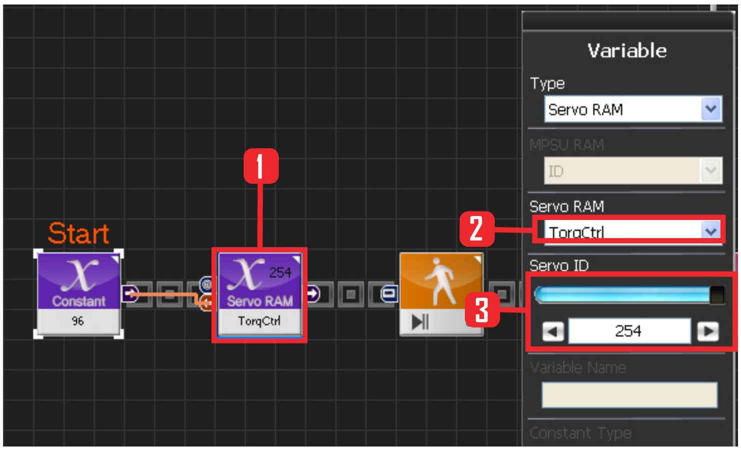
Constant Type: ○ Bool ● Int

Constant Value: □ True □ False

96

06 Setup Constant

This section allows the servo motor to operate on it’s own.
Select Constant as the Variable Type. In properties, set constant value as 96.
When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

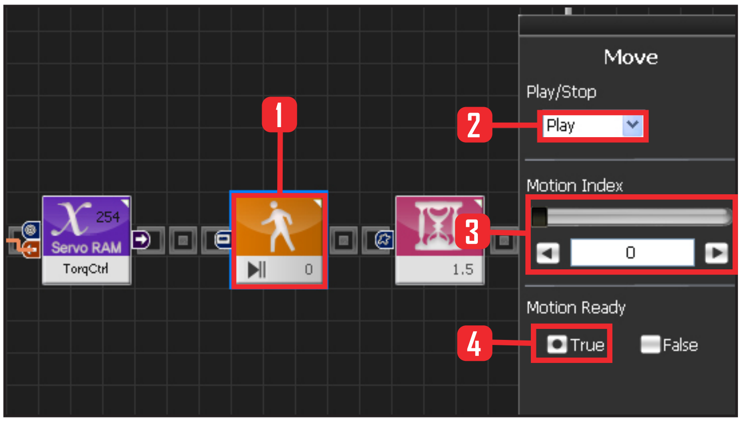
Variable Name

Constant Type

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉 Type : Servo RAM.
Select Servo RAM : TorqCtrl.
Set Servo ID : 254. 254 means it will be applied to all connected servos.



Move

Play/Stop: Play

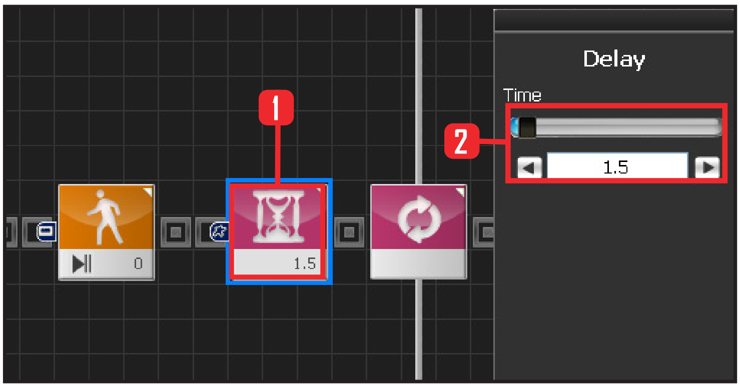
Motion Index: 0

Motion Ready: ● True ○ False

08 Motion Ready

Robot goes through a prepatory stage before starting the next motion. This prepatory stage allows the robot to move slowly to the the initial position of the motion to be run. This prevents stress or damage from sudden change in motion.
IF Motion Ready is True prepare for next motion. Run next motion if False.

Select Motion 〉 Move module.
Select Play/Stop : Play.
Set Motion Index : 0 . walk forward
Select Motion Ready : True.
Motion Ready Stage



Delay

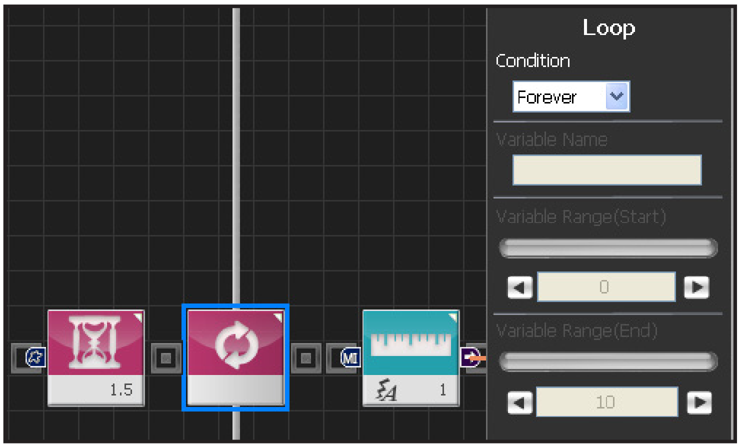
Time: 1.5

09 Delay

Set delay to 1.5s to prevent next step from staring before Motion Ready ends.

10 Loop

Select Loop: Forever
Infinite loop.

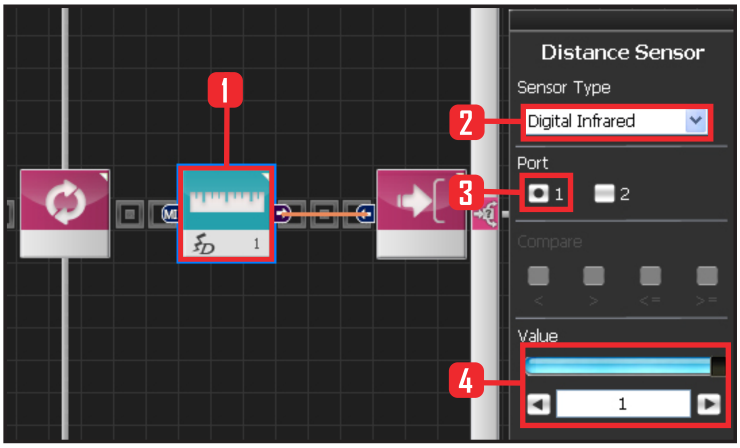


Loop
Condition: Forever
Variable Name
Variable Range(Start): 0
Variable Range(End): 10

11 Setup Digital Distance Sensor

Digital sensors have different measuing distance.
Setup with 20cm as standard.

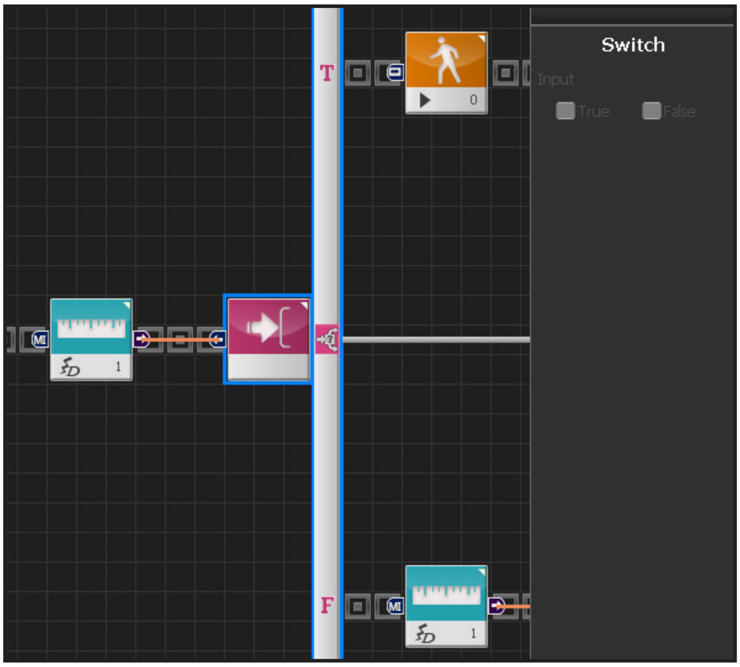
Select Sensor 〉Distance Sensor module.
Select Sensor Type : Digital Infrared.
Select Port : 1.
Set Value : 1 . farther than 10cm.



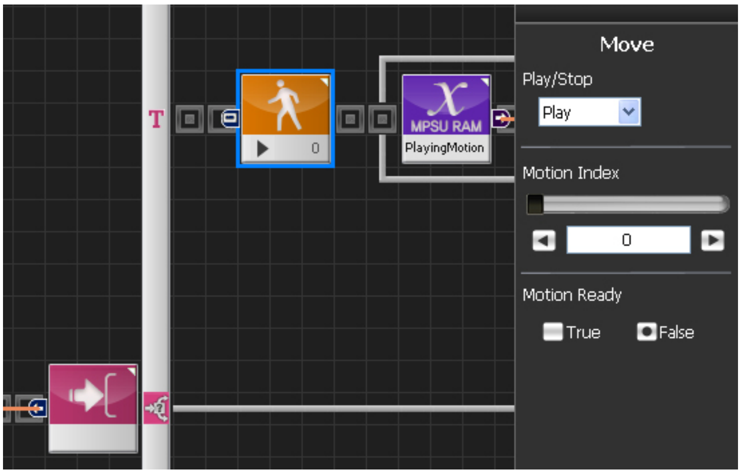
Distance Sensor
Sensor Type: Digital Infrared
Port: ● 1 □ 2
Compare: □ < □ > □ <= □ >=
Value: 1

12 If Conditional Statement

Proceed if True, go to next conditional statement if False.



Switch
Input: □ True □ False



Move

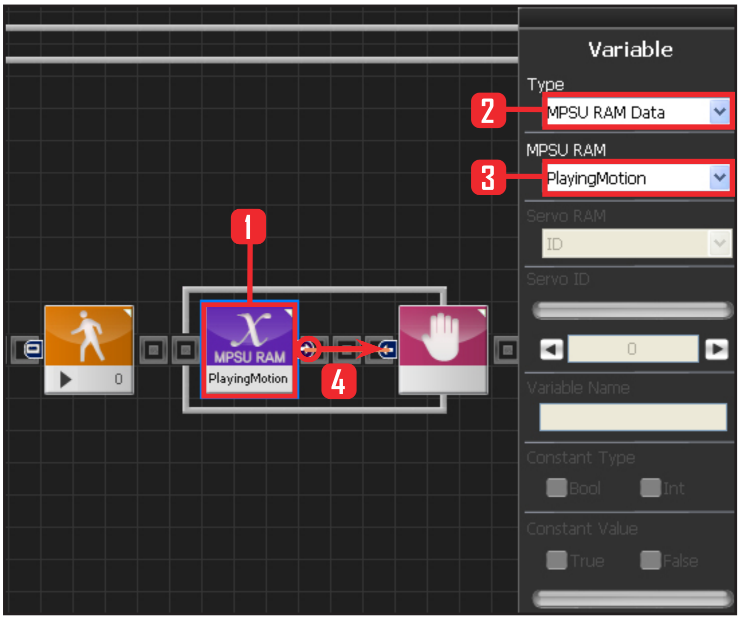
Play/Stop: Play

Motion Index: 0

Motion Ready: ○ True ● False

13 Forward

Robot will move forward since the distance is greater than10cm.
When False is selected as Motion Ready value, robot will proceed with forward motion.



Variable

Type: MPSU RAM Data

MPSU RAM: PlayingMotion

Servo RAM: ID

Servo ID: 0

Variable Name

Constant Type: □ Bool □ Int

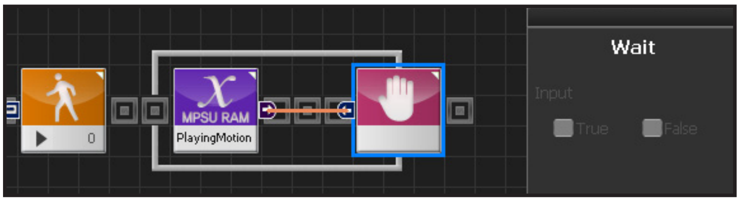
Constant Value: □ True □ False

14 Motion Movement Check

Loop refers to continuous repetition. It takes time for the actual motion to complete after Move command has been issued, but loop with single move module will continue to run and give motion command even while the previous motion is still running. The lag in actual motion will result in difference between the number of motion commands given by the move module and the number of actual motions. To correct this difference, loop will need to wait for the motion to complete before repeating the process.'Playing Motion' is found within Variable > MPSU RAM Data. 'Playing Motion' is a variable that checks whether the motion is in process. Loop will wait for the current motion to end if 'wait' is added to the 'Playing Motion'.

Select Data > Variable Module.
Select Type : MPSU RAM Data
Select MPSU RAM : Playing Motion
Add Wait module to the output connector.

Data > Variable.
Type : MPSU RAM Data.
MPSU RAM : Playing Motion.

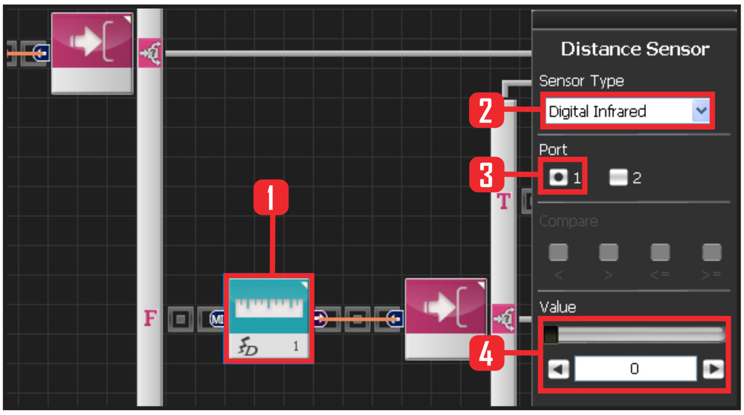


Wait

Input: □ True □ False

15 Wait

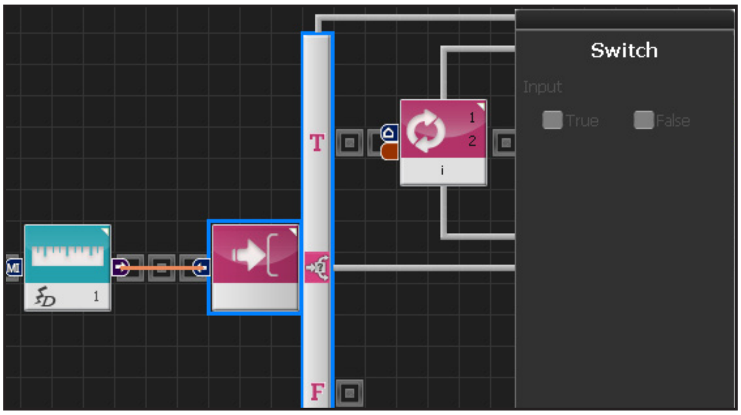
Wait untill the motion ends.
Go to the begining and repeat when motion ends.



16 Motion Near The Wall

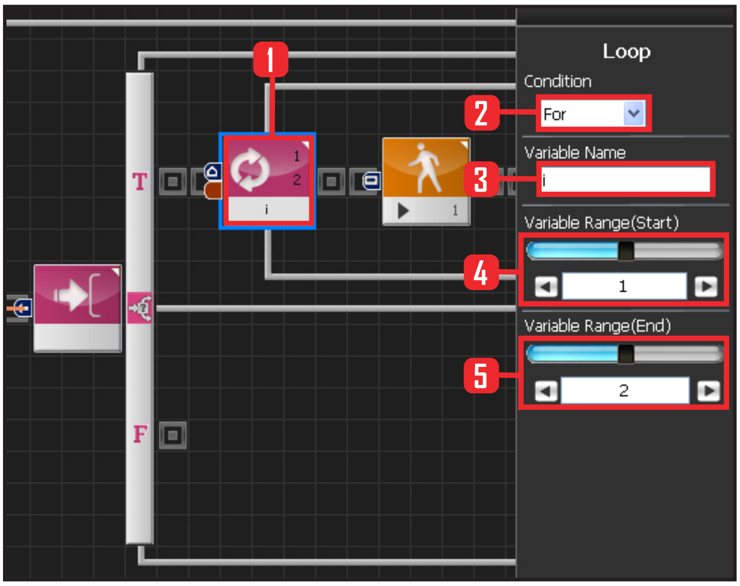
When the robot is less than 10cm from the wall, program will make the robot walk backwards and change direction.

Select Sensor 〉Distance Sensor module.
Select Sensor Type : Digital Infrared .
Select Port : 1.
Set Value : 0 . Within 10cm distance.



17 If Conditional Statement

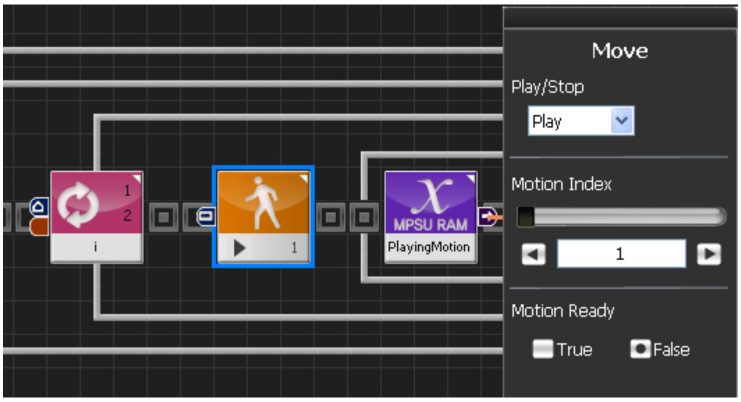
Run statement within True if less than 10cm from the wall.



18 For Loop

Repeat certain motion untill the condition is met. Motion #1 is a walk backwards motion. walk backwards motion makes the robot take one step backward each using left and rigt feet.
Robot can be moved to the desired location by adding For statement to the motion to repeat the motion desired number of times.

Select Flow 〉Loop module.
Select Condition : For .
Set Variable Name: i .
Set Variable Range(Start) 1 .
Set Variable Range(End) 2 .
Repeat motion twice.



Move

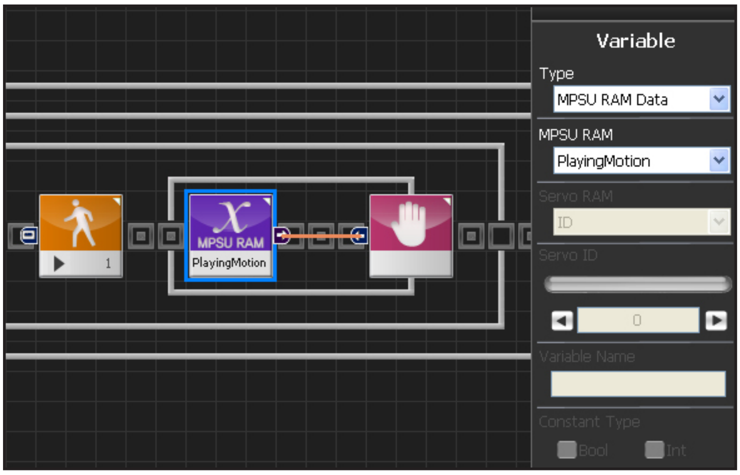
Play/Stop: Play

Motion Index: 1

Motion Ready: □ True ● False

19 Walk Backwards

#1is a walk backwards motion.
Robot will run the walk backwards motion if False is selected.



Variable

Type: MPSU RAM Data

MPSU RAM: PlayingMotion

Servo RAM: ID

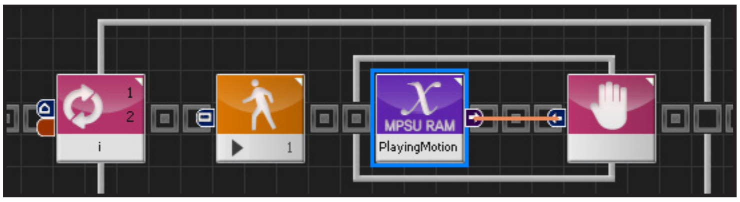
Servo ID: 0

Variable Name

Constant Type: □ Bool □ Int

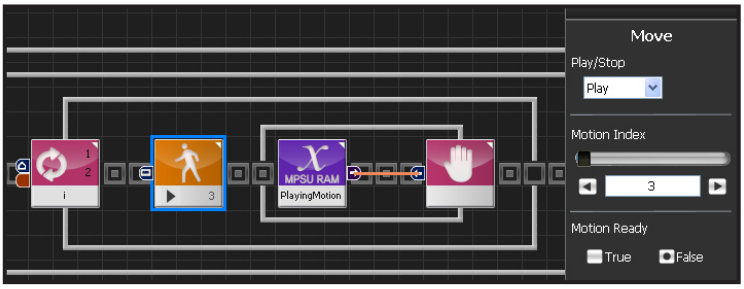
20 Check Motion

Use Playing Motion to check the robot motion. When the motion ends, return to the start of the For statement.



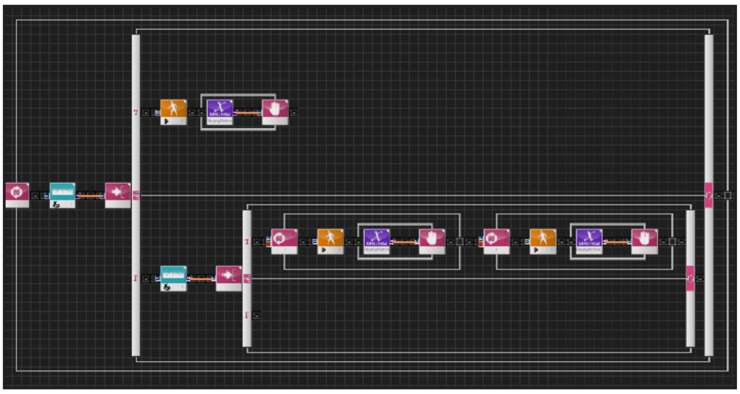
21 Repeat Backwards Motion Twice

Program makes the robot repeat the walk backwards motion twice.



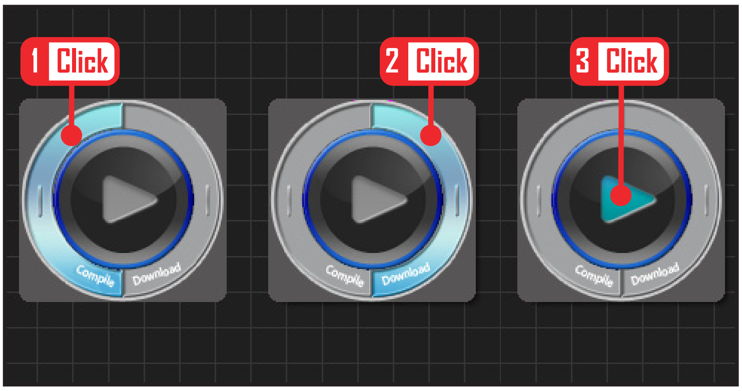
22 Right Turn

Robot motion #3 makes the robot change direction to the right. Right turn motion can be controlled by using the For statement . Seletct motion #3, set For statement from 1–3 and program as above.



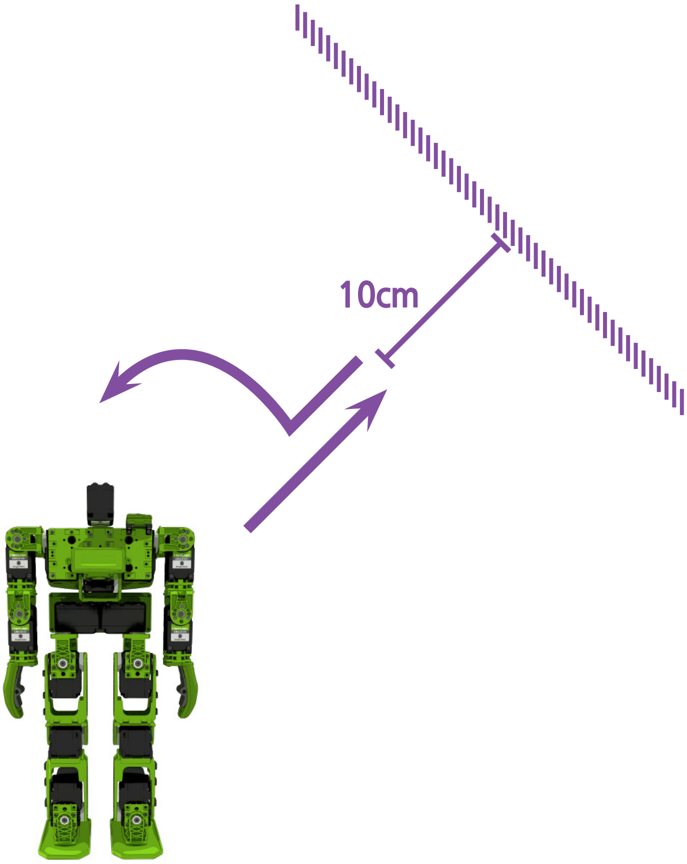
23 Entire Program

Program make the robot walk froward when the distance to the wall is greater than 10cm . If the distance is less than 10cm, robot will repeat the backward and right turn motion according to the For statement and avoid the obstacle.



24 Compile, Download, Run

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



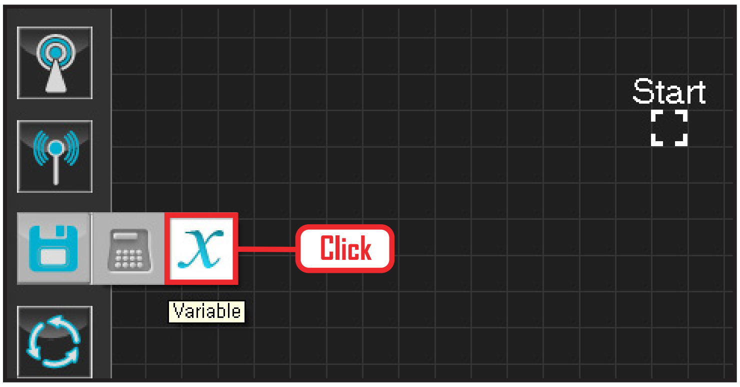
25 Robot Motion

When detects a wall within 10cm, it will walk backwards, change direction to the right and start walking forward again.

Analog Distance Sensor Example Step by Step

Example Description

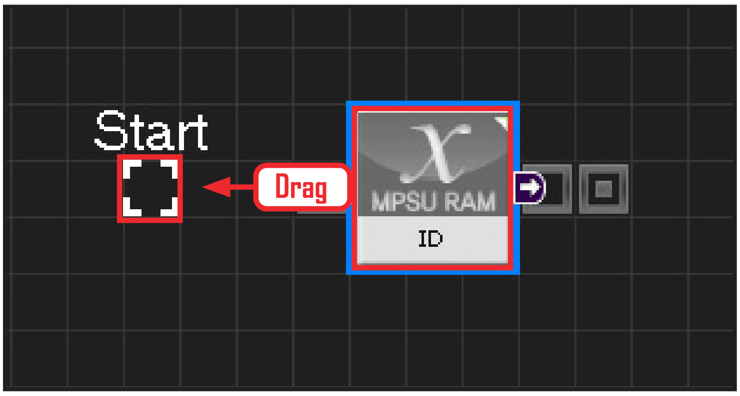
This example program is an obstacle avoidance program that uses analog sensor to make the robot avoid hitting an obstacle by turning to the left. Hovis Lite has two type of distance sensors, analog and digital. Digital sensor uses specific distance (10cm) as a reference and it can only determine if an object is within or beyond the reference range. On the other hand, analog sensor is capable of detecting an object within 6~80cm range. This example requires PSD sensor to be installed at ADC port #1 (left).



01 Assign Variable

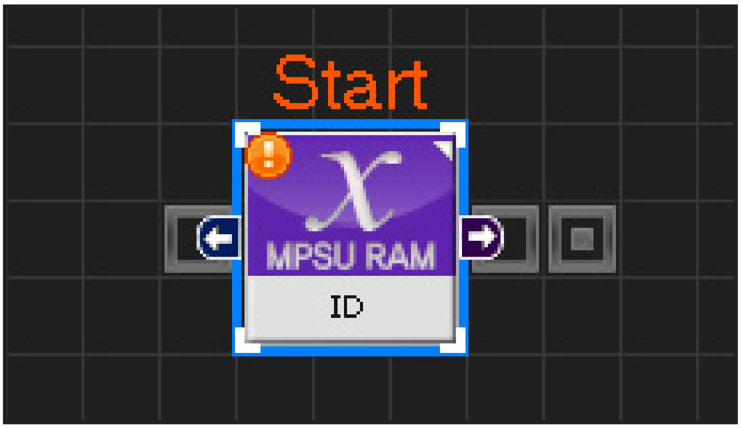
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

Click Data 〉 Variable module.



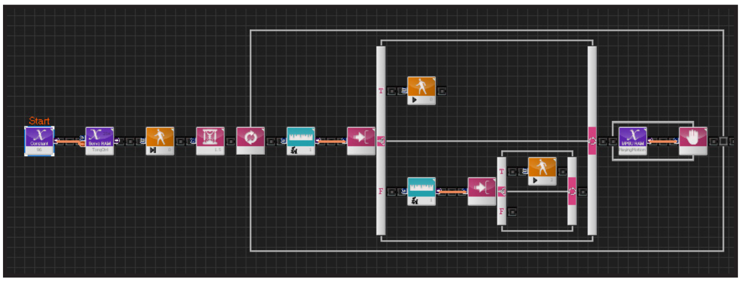
02 Start

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



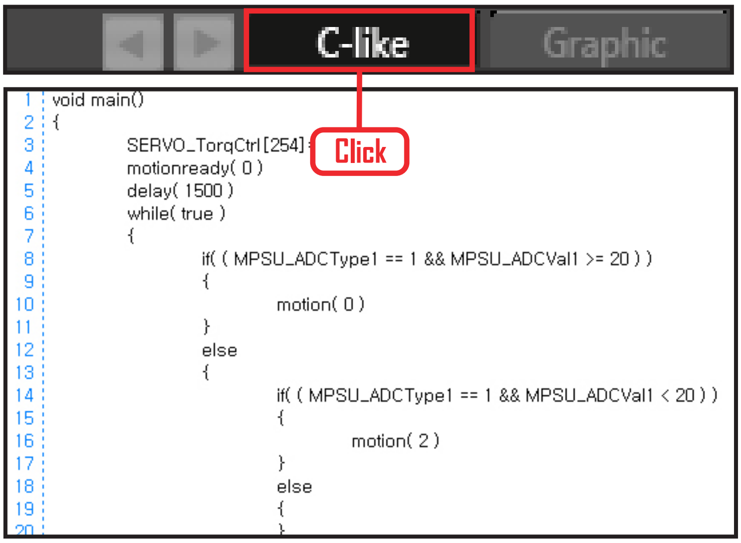
03 Start Programming

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 Progam

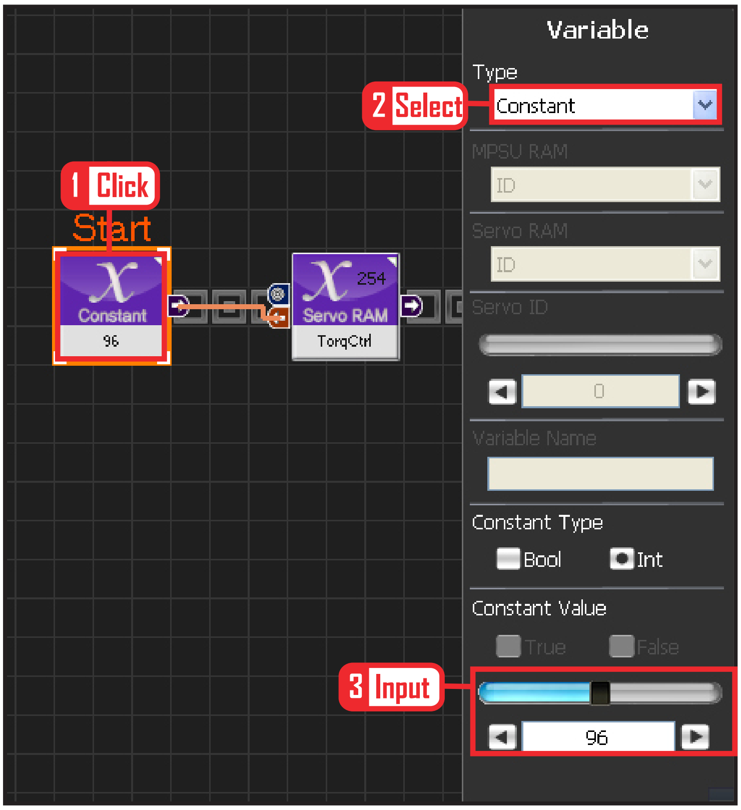
Entire program using the analog sensor to make the robot avoid hitting an obstacle.



```
void main()
{
        SERVO_TorqCtrl[254]
        motionready( 0 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_ADCType1 == 1 && MPSU_ADCVal1 >= 20 ) )
                {
                        motion( 0 )
                }
                else
                {
                        if( ( MPSU_ADCType1 == 1 && MPSU_ADCVal1 < 20 ) )
                        {
                                motion( 2 )
                        }
                        else
                        {
                        }
```

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.



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name

Constant Type: □ Bool ● Int

Constant Value: □ True □ False

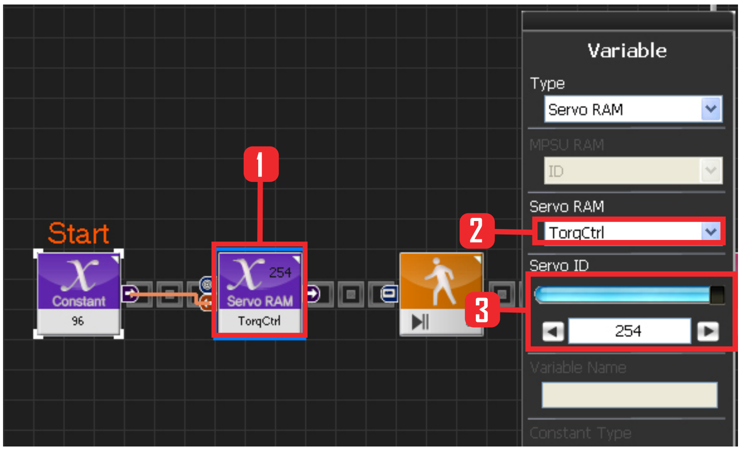
96

06 Setup Constant

This section allows the servo motor to operate on it's own.

Select Constant as the Variable Type. In properties, set constant value as 96.

When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.

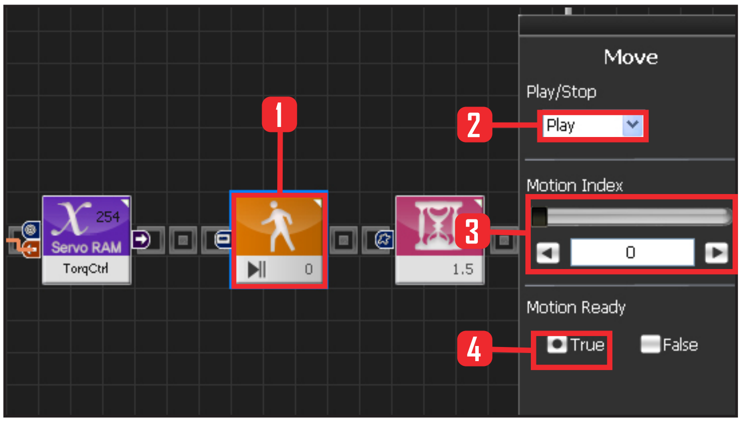


Variable
Type: Servo RAM
MPSU RAM: ID
Servo RAM: TorqCtrl
Servo ID: 254
Variable Name
Constant Type

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉 Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos.

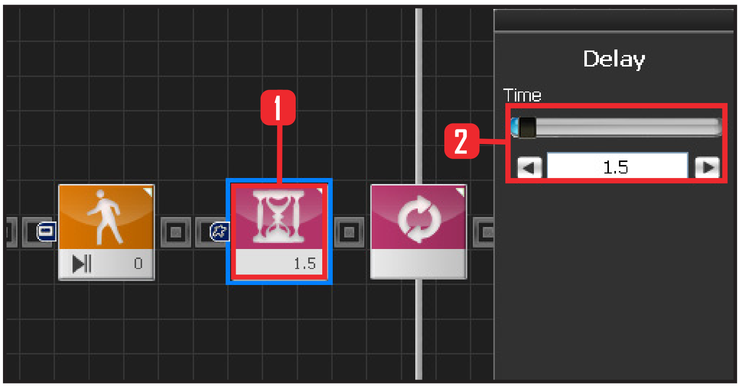


Move
Play/Stop: Play
Motion Index: 0
Motion Ready: ● True ○ False

08 Motion Ready

Robot goes through a prepatory stage before starting the next motion. This prepatory stage allows the robot to move slowly to the the initial position of the motion to be run. This prevents stress or damage from sudden change in motion. IF Motion Ready is True prepare for next motion. Run next motion if False.

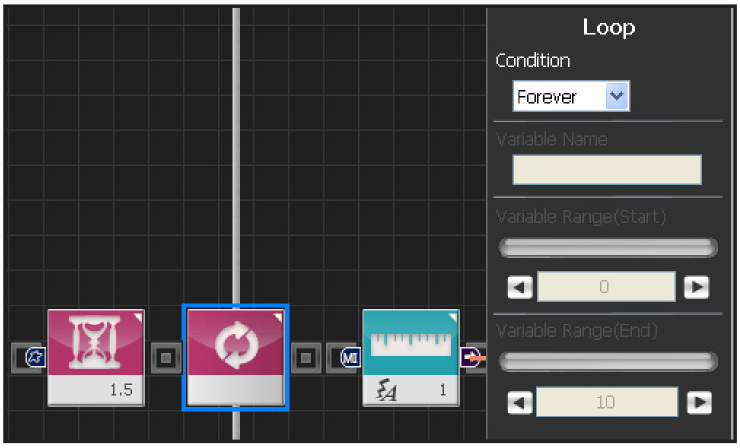
Select Motion 〉 Move module.
Select Play/Stop : Play .
Set Motion Index : 0 . walk forward
Select Motion Ready : True .
Motion Ready Stage



Delay
Time: 1.5

09 Delay

Set delay to 1.5s to prevent next step from staring before Motion Ready ends.



Loop

Condition

Forever

Variable Name

Variable Range(Start)

0

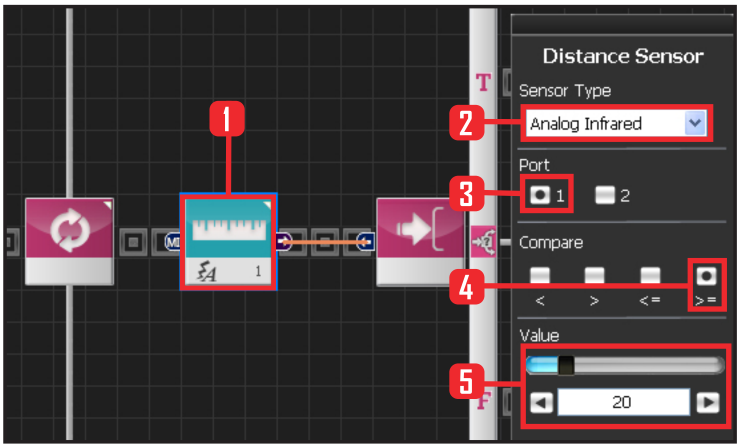
Variable Range(End)

10

10 Loop

Select Loop: Forever

Infinite loop.



Distance Sensor

Sensor Type

Analog Infrared

Port

● 1 □ 2

Compare

□ < □ > □ <= ● >=

Value

20

11 Setup Analog Sensor

Setup with 20cm as standard.

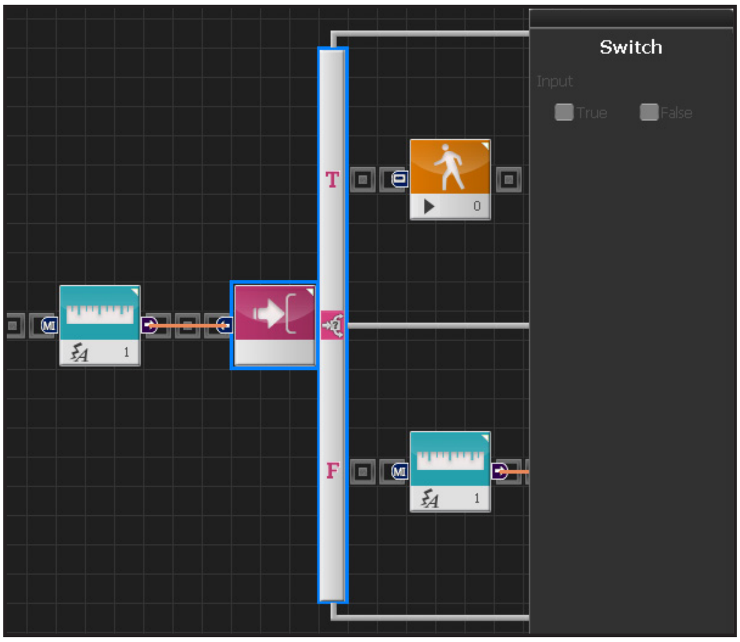
Select Sensor 〉Distance Sensor module.

Select Sensor Type : Analog Infrared

Select Port : 1.

Select Compare : 〉= . True if equal to or greater than standard.

Set Value : 20 . 20cm .



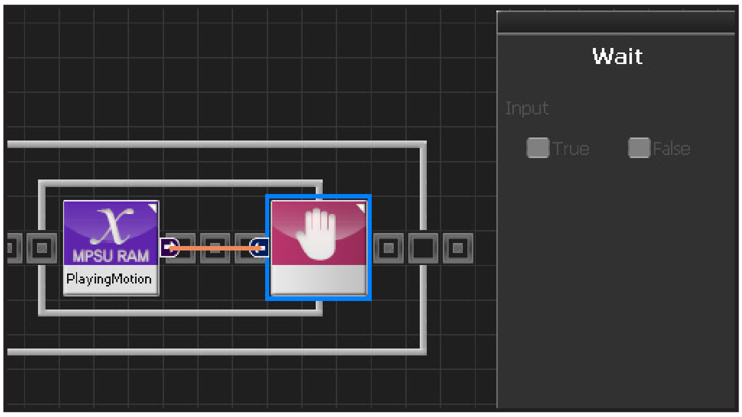
Switch

Input

□ True □ False

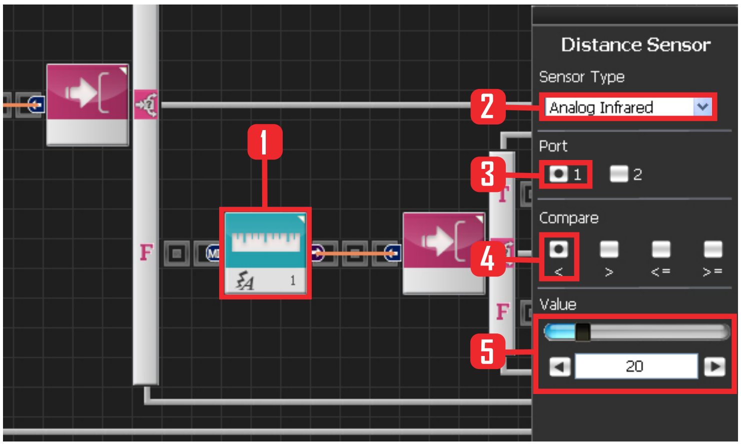
12 If Conditional Statement

Proceed if True or go to the next conditional statement if False.



13 Forward

Robot will move forward since the distance is farther than 20cm.
When False is selected as Motion Ready value, robot will proceed with forward motion.



Sensor Type: Analog Infrared

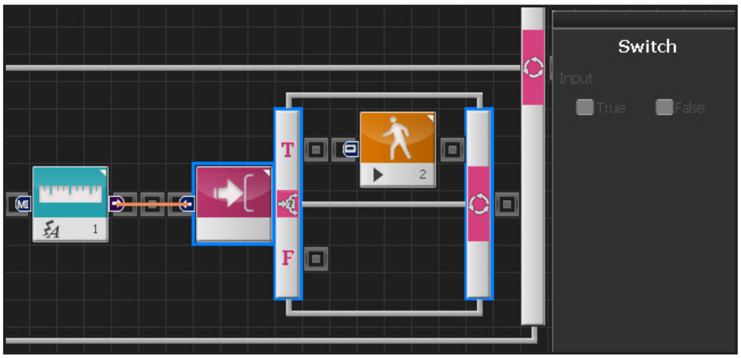
Port: ● 1 □ 2

Value: 20

14 Motion Near The Wall

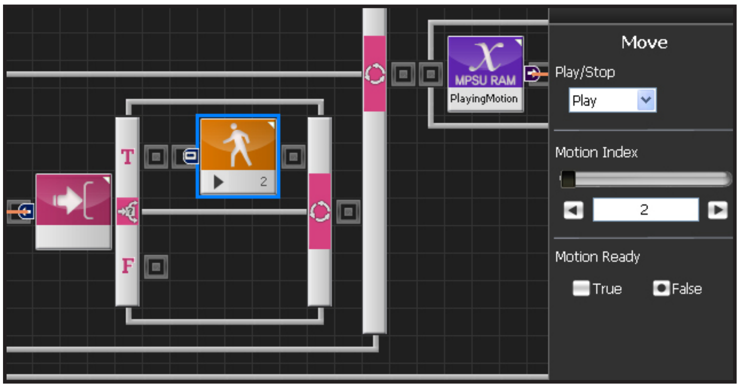
Robot will make a left turn if it detecs an obstacle within 20cm.

Select Sensor 〉 Distance Sensor module.
Select Sensor Type : Analog Infrared .
Select Port : 1.
Select Compare :〈 . True if less than standard.
Value : 20 . 20cm .



15 If Conditional Statement

Run statement within True if less than 20cm from the obstacle.



Move

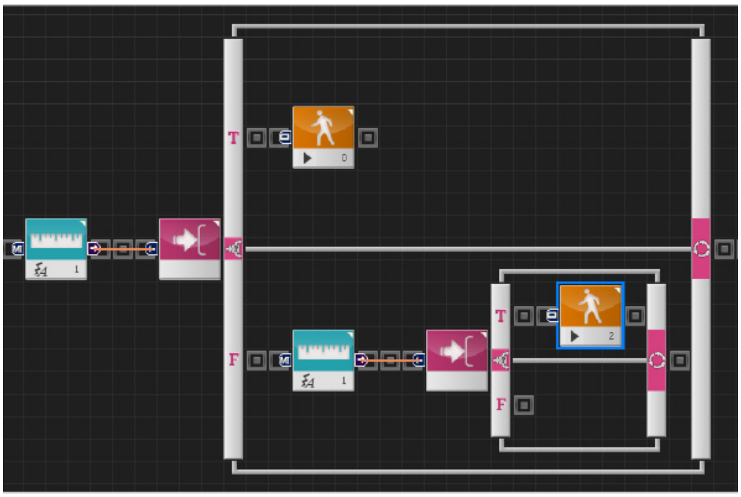
Play/Stop: Play

Motion Index: 2

Motion Ready: □ True ● False

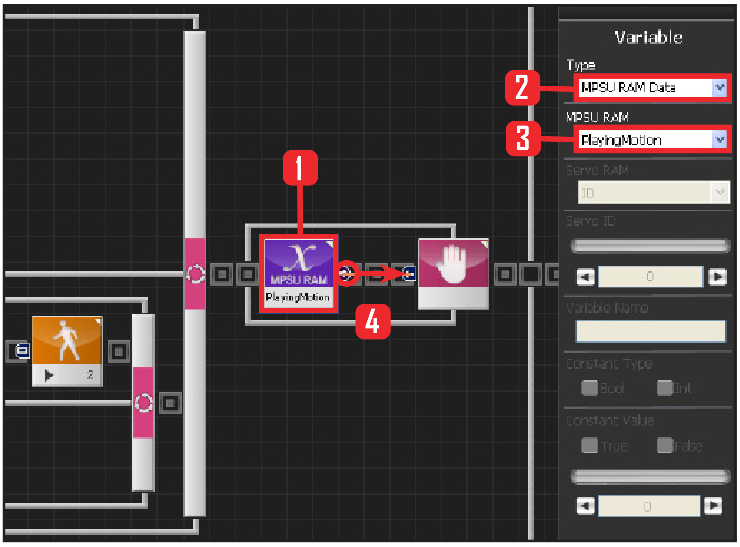
16 Left Turn

Robot motion #2makes the robot change direction to the left. Robot will run the left turn motion if the Motion Ready value is False.



17 Motion According to Distance

If the distance to the obstacle is greater than 20cm, robot will keep moving forward. If the distance is less than 20cm, robot will make a left turn.



Variable

Type: MPSU RAM Data

MPSU RAM: PlayingMotion

Servo RAM: ID

Servo ID: 0

Variable Name

Constant Type: □ Bool □ Int

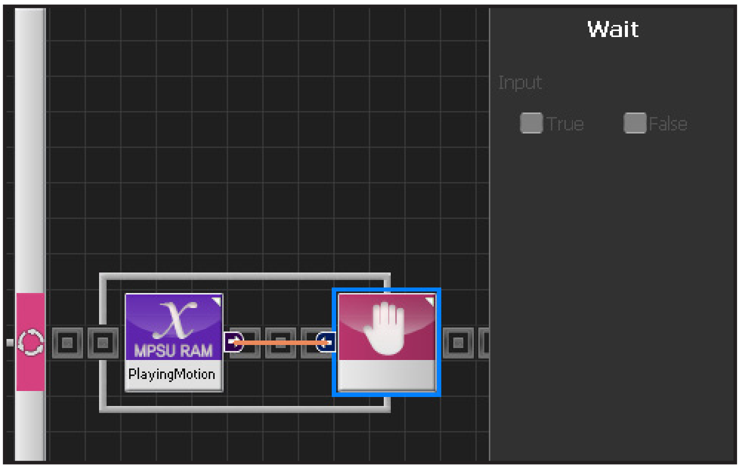
Constant Value: □ True □ False

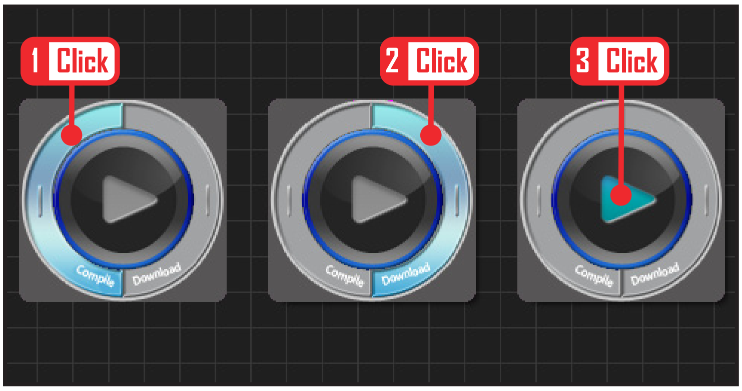
0

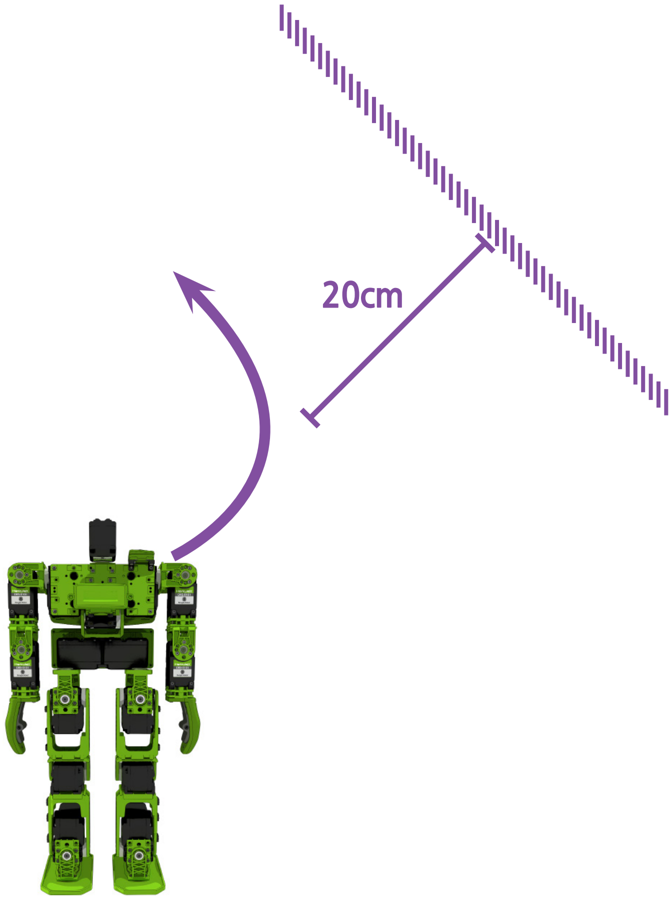
18 Motion Movement Check

Loop refers to continuous repetition. It takes time for the actual motion to complete after Move command has been issued, but loop with single move module will continue to run and give motion command even while the previous motion is still running. The lag in actual motion will result in difference between the number of motion commands given by the move module and the number of actual motions. To correct this difference, loop will need to wait for the motion to complete before repeating the process.'Playing Motion' is found within Variable 〉 MPSU RAM Data. 'Playing Motion' is a variable that checks whether the motion is in process. Loop will wait for the current motion to end if 'wait' is added to the 'Playing Motion'.

Select Data 〉 Variable Module.
Select Type : MPSU RAM Data
Select MPSU RAM : Playing Motion
Add Wait module to the output connector.







19 Wait

Wait untill the motion ends.
Go to the begining and repeat when motion ends.

20 Compile, Download, Run

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

21 Robot Motion

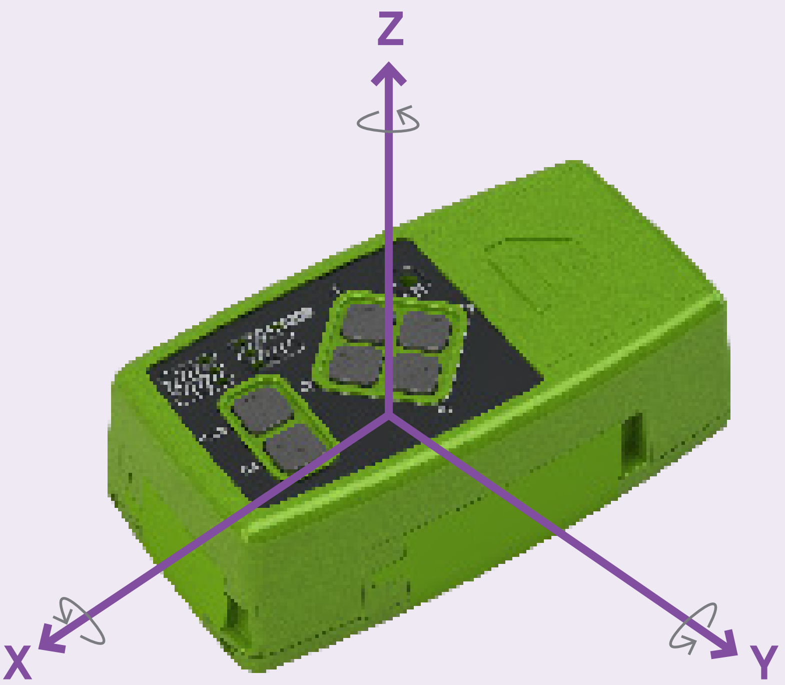
Robot will walk forward and then make a left turn if it detects an obstacle within 20cm distance.

Acceleration Sensor Example Step by Step

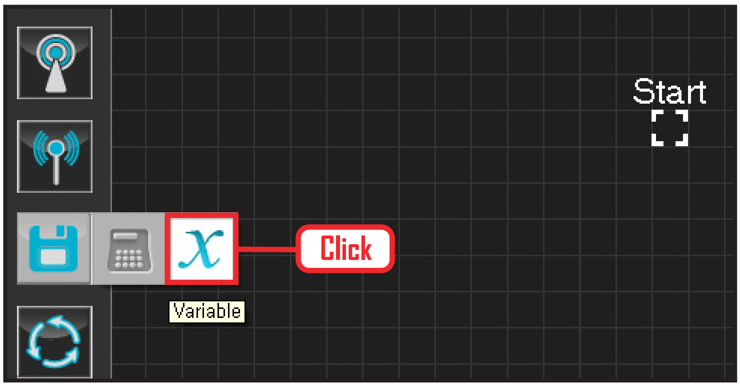
Example Description

Use the Acceleration sensor to make the robot stand when it falls forward or backward.

Acceleration sensor is attached to a module type board that also has Gyro sensor attached to it. Sensor module can be installed inside the controller by opening the controller back cover .



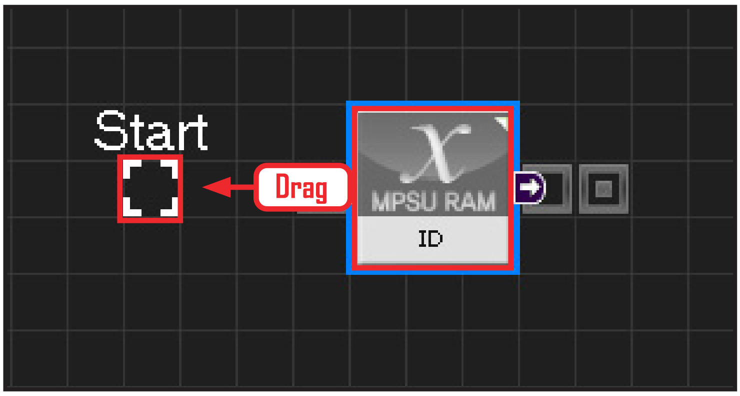
- When the robot is in prone position (lying face down), Z axis "-" acclerates and it's value is approximately - 4096.
- When the robot is in supine position (lying on the back), Z axix "+" accelerates and it's value is approximately 4096. (4096 is approximately 1g force of gravity value.)



01 Assign Variable

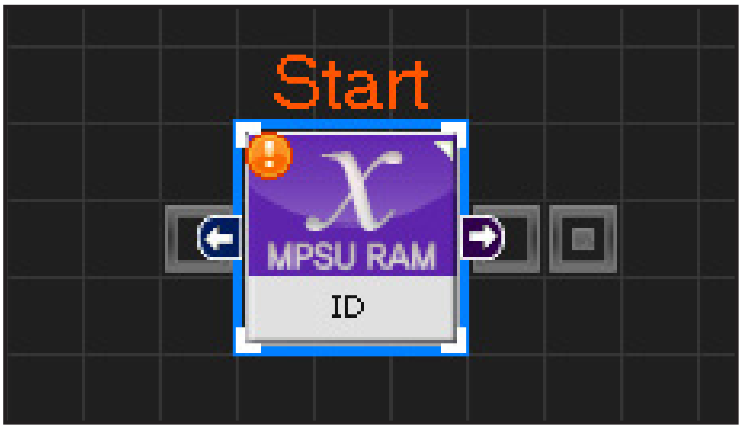
Operating the robot is same as operating the robot servo motor. Value has to be assigned so that servo will be able to operate.

Click Data 〉 Variable module.



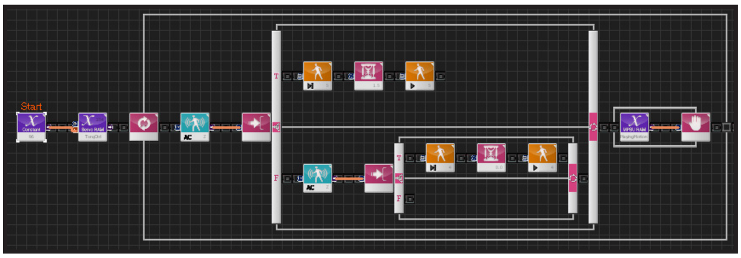
02 Start

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



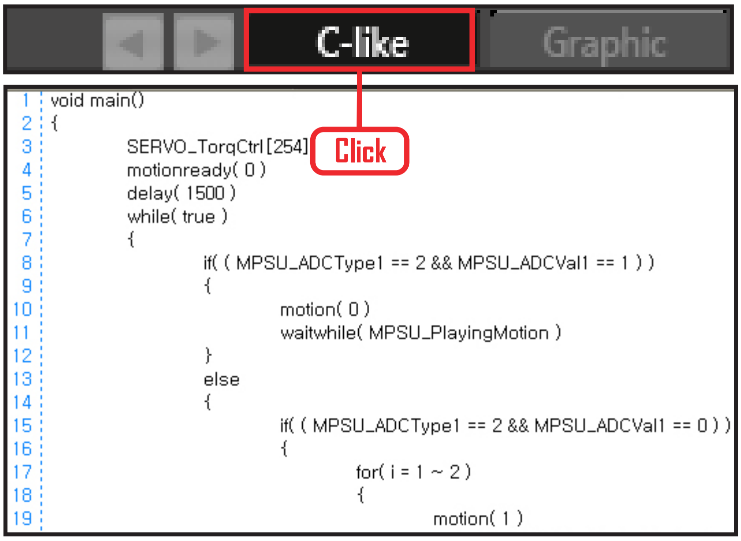
03 Start Programming

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

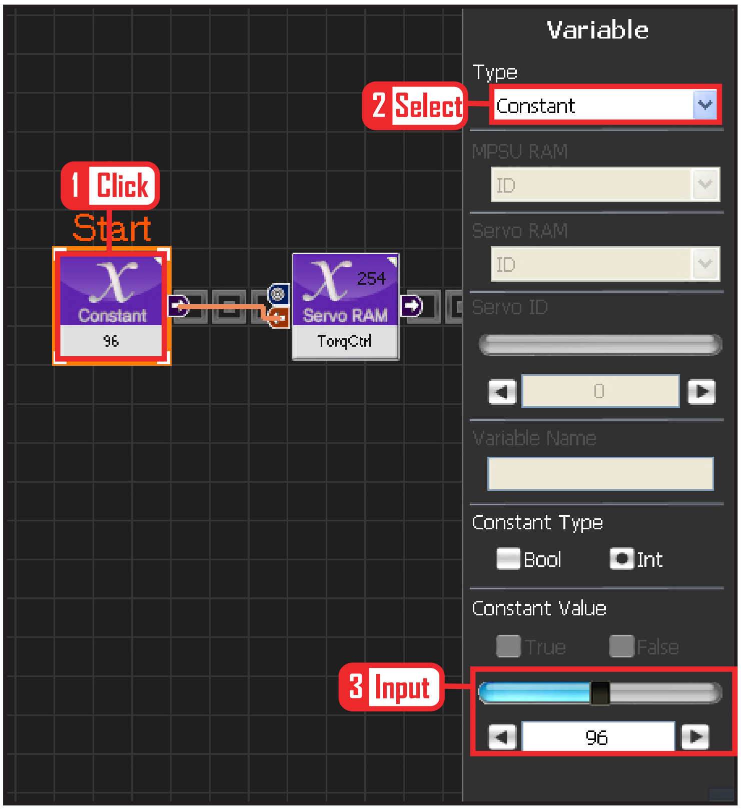
Entire program using the accleration sensor to make the robot stand after falling.



```
void main()
{
        SERVO_TorqCtrl[254]
        motionready( 0 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 1 ) )
                {
                        motion( 0 )
                        waitwhile( MPSU_PlayingMotion )
                }
                else
                {
                        if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 0 ) )
                        {
                                for( i = 1 ~ 2 )
                                {
                                        motion( 1 )
```

05 View 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.



Type: Constant

Constant Type: □ Bool ● Int

Constant Value: 96

06 Setup Constant

This section allows the servo motor to operate on it's own.

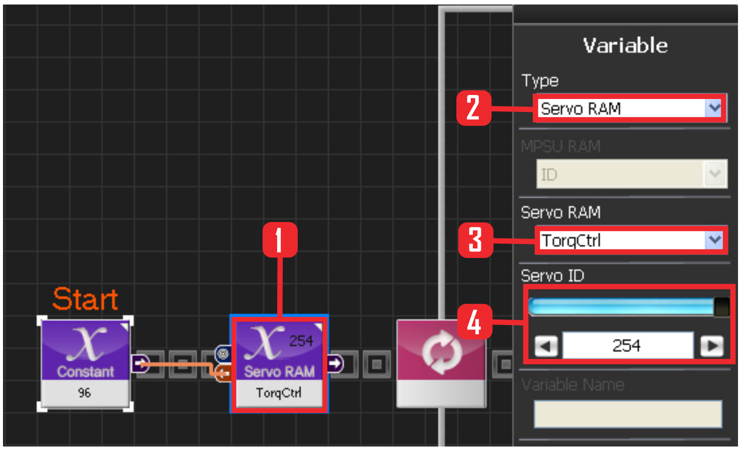
Select Constant as the Variable Type. In properties, set constant value as 96.

When 96(0x60) is entered in the servo TorgControl register, servo becomes ready to operate. This value is sent to the torque value of the next moduel through the output connector.

07 Apply to All Servos

This section applies contact value 96 to all servos.

Select Variable 〉Type : Servo RAM.
Select Servo RAM : TorqCtrl .
Set Servo ID : 254. 254 means it will be applied to all connected servos.



Variable

Type: Servo RAM

MPSU RAM: ID

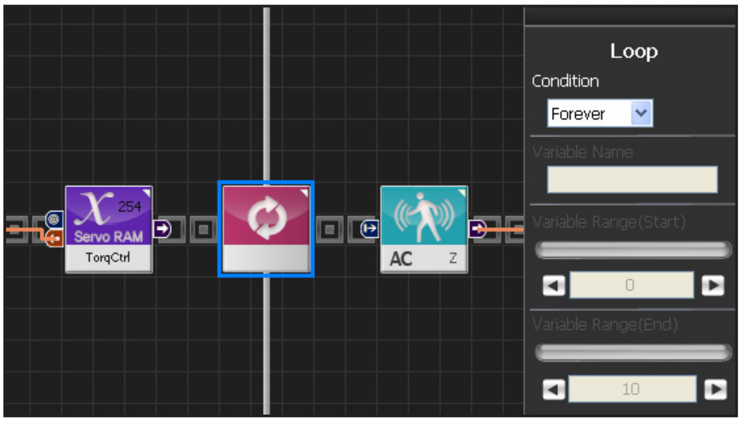
Servo RAM: TorqCtrl

Servo ID: 254

Variable Name

08 Loop

Select Flow 〉Loop module.
Select Condition: Forever.
Infinite loop.



Loop

Condition: Forever

Variable Name

Variable Range(Start): 0

Variable Range(End): 10

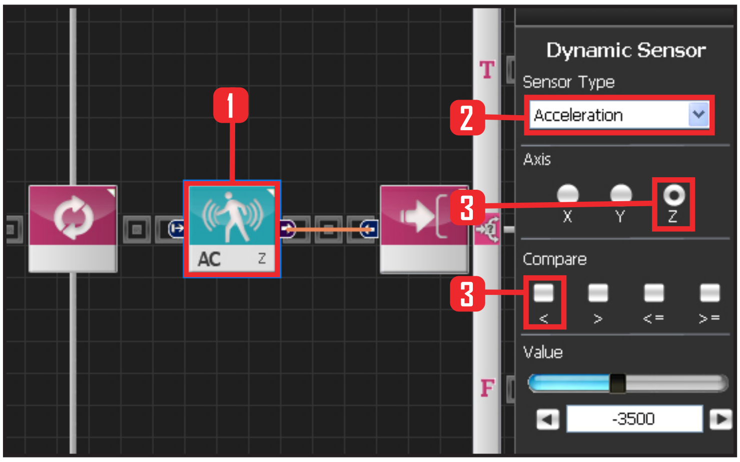
09 Acceleration Setup (Prone)

Acceleration has value of 0 when the robot is standing up straight.
When the robot is in prone position it has value of –4096 and +4096 when in supine position.
If the accleration value is near –4096, it can be assumed that the robot has fallen forward. Set –3500 as standard value.

if the value is less thant –3500, robot is assumed to have fallen forward.

Select Sensor 〉Dynamic Sensor module.

Select Sensor Type : Acceleration
Select Axis : Z .
Select Compass : 〈 .
Set Value : –3500 .



Dynamic Sensor

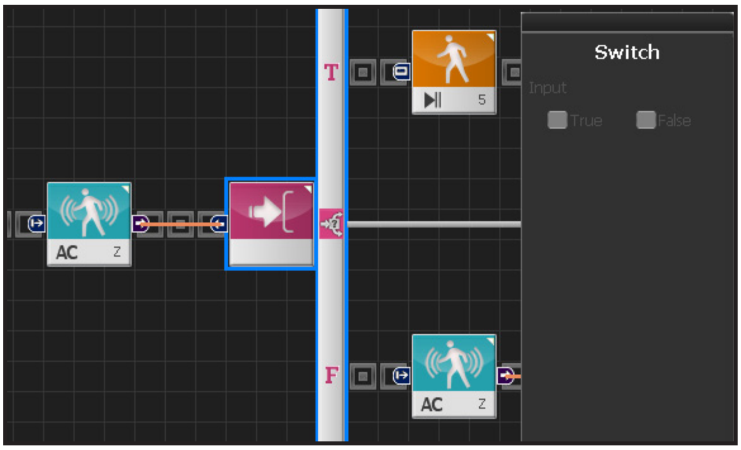
Sensor Type: Acceleration

Axis: ○ X ○ Y ● Z

Compare: □ < □ > □ <= □ >=

Value: -3500

캡쳐다시하기



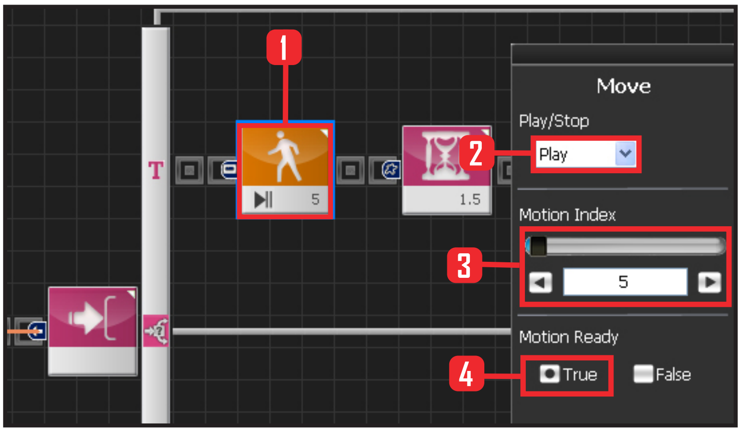
Switch

Input

□ True □ False

10 If Conditional Statement

Robot gets up backward when True, Proceed to next conditional statement if False.



Move

Play/Stop: Play

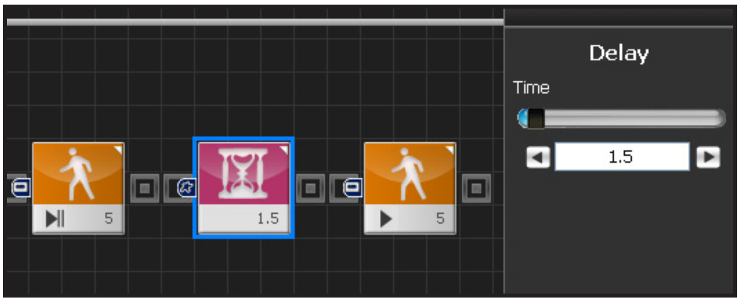
Motion Index: 5

Motion Ready: ● True □ False

11 Run Up Backwards Motion

Insert Up Backwards motion when the robot is in prone position.
Motion #5 is up backward motion.

Select Motion 〉 Move module.
Select Play/Stop : Play .
Set Motion Index : 5
Select Motion Ready : True
Prepatory stage for motion.



Delay

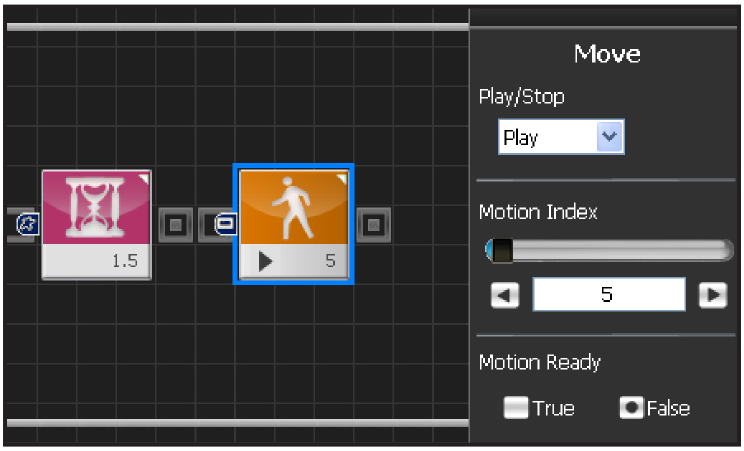
Time: 1.5

12 Delay

Set delay to 1.5s to prevent next step from staring before Motion Ready ends.

13 Run Up Backwards Motion

When False is selected as Motion Ready value, robot will run the up backwards motion.



Move

Play/Stop: Play

Motion Index: 5

Motion Ready: ☐ True ● False

14 Setup Gravity Acceleration (Supine Position)

Gravity acceleration has value of 0 when the robot is standing up straight.
When the robot is in prone position it has value of –4096 and +4096 when in supine position.
If the accleration value is near 4096, it can be assumed that the robot has fallen backward. Set 3500 as standard value.

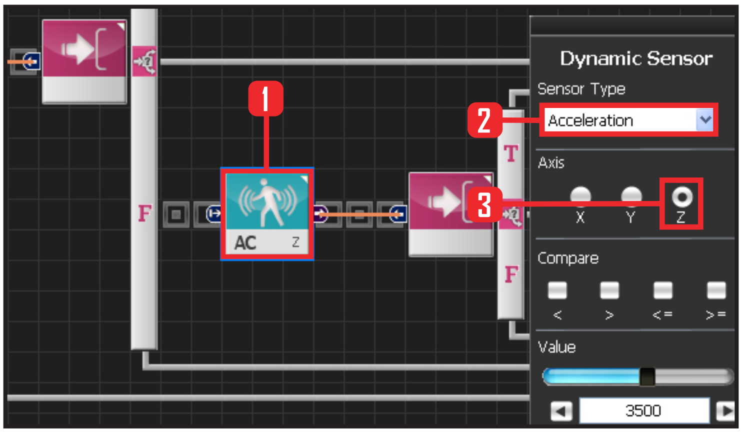
Select Sensor 〉 Dynamic Sensor module.

Select Sensor Type : Acceleration

Select Axis : Z .

Select Compass : 〉 .

Set Value : 3500 .



Dynamic Sensor

Sensor Type: Acceleration

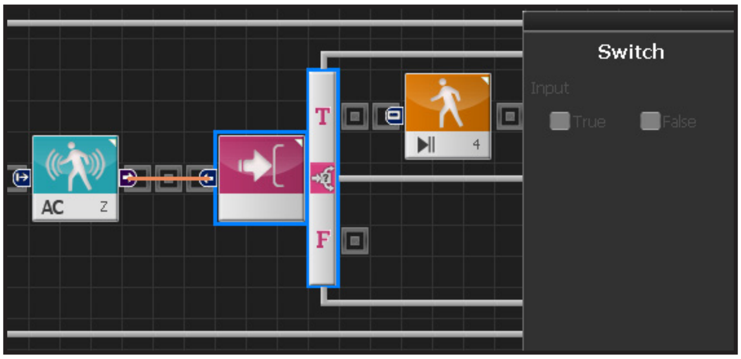
Axis: ○ X ○ Y ● Z

Compare: ☐ < ☐ > ☐ <= ☐ >=

Value: 3500

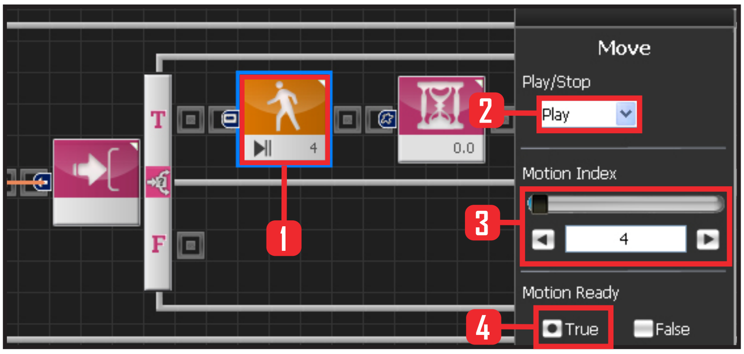
15 If Conditional Statement

Robot gets up if True.



Switch

Input: ☐ True ☐ False



Move

Play/Stop
Play

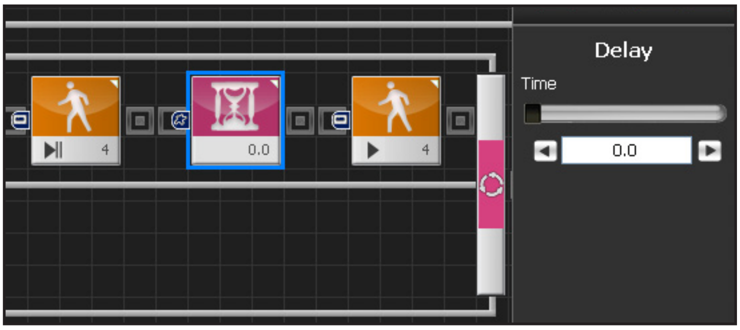
Motion Index
4

Motion Ready
● True □ False

16 Motion Ready

Robot goes through a prepatory stage before starting the next motion. This prepatory stage allows the robot to move slowly to the the initial position of the motion to be run. This prevents stress or damage from sudden change in motion. IF Motion Ready is True prepare for next motion. Run next motion if False

Select Motion 〉 Move module.
Select Play/Stop : Play .
Select Motion Index : 4 . Motion # 4, robot gets up forward.
Select Motion Ready : True .
Motion ready stage.

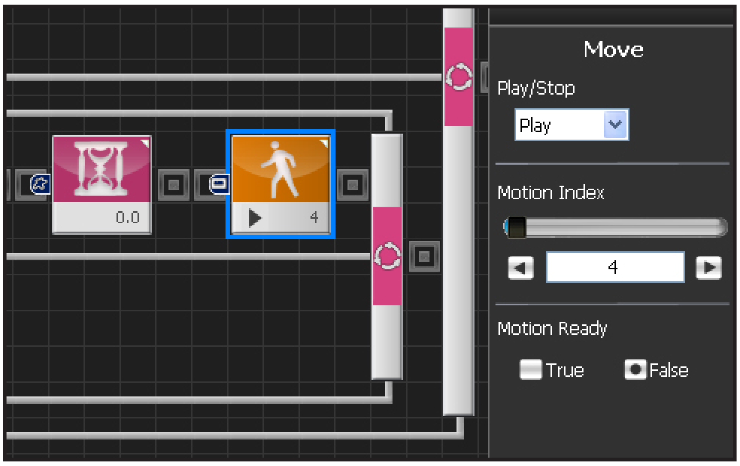


Delay

Time
0.0

17 Delay

Set delay to 1.5s to prevent next step from staring before Motion Ready ends.



Move

Play/Stop
Play

Motion Index
4

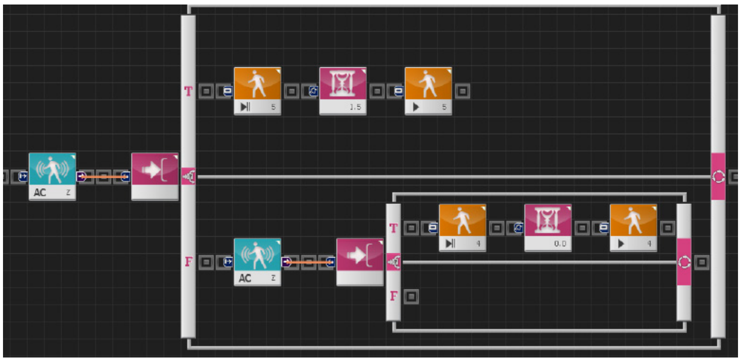
Motion Ready
□ True ● False

18 Run Up Forward Motion

When False is selected as Motion Ready value, robot will run the up forward motion.

19 Getting Back Up

Robot determines if it has fallen by referencing the Z axix acceleration value and runs the appropriate motion to get back up.



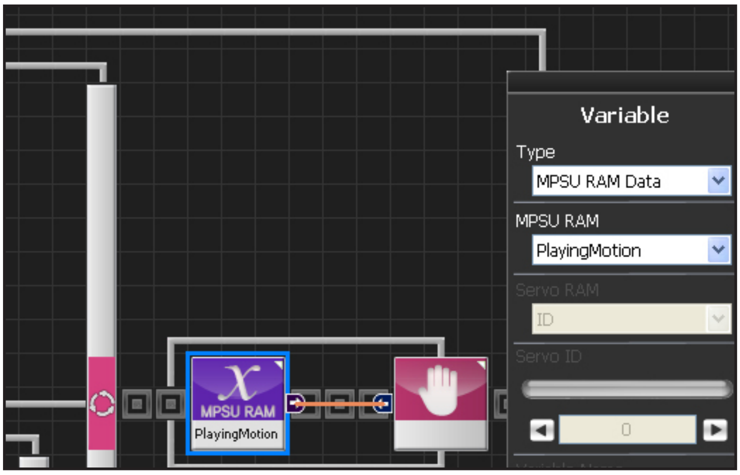
20 Motion Movement Check

Loop refers to continuous repetition. It takes time for the actual motion to complete after Move command has been issued, but loop with single move module will continue to run and give motion command even while the previous motion is still running. The lag in actual motion will result in difference between the number of motion

commands given by the move module and the number of actual motions. To correct this difference, loop will need to wait for the motion to complete before repeating the process.'Playing Motion' is found within Variable 〉 MPSU RAM Data.

'Playing Motion' is a variable that checks whether the motion is in process. Loop will wait for the current motion to end if 'wait' is added to the 'Playing Motion'.

Select Data 〉 Variable Module.
Select Type : MPSU RAM Data
Select MPSU RAM : Playing Motion
Add Wait module to the output connector.


Variable

Type: MPSU RAM Data

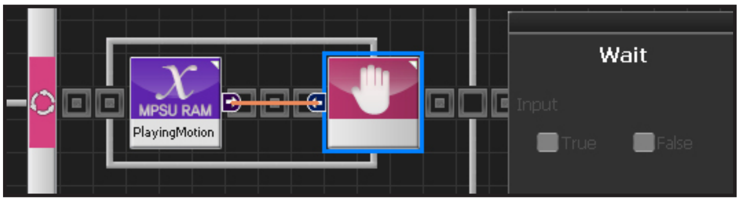
MPSU RAM: PlayingMotion

Servo RAM: ID

Servo ID: 0

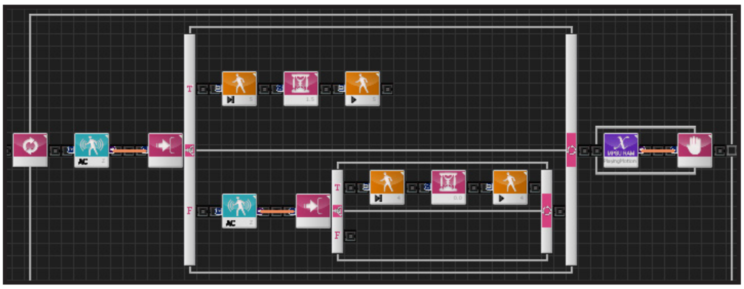
21 Wait

Wait untill the motion ends.
Go to the begining and repeat when motion ends.


Wait

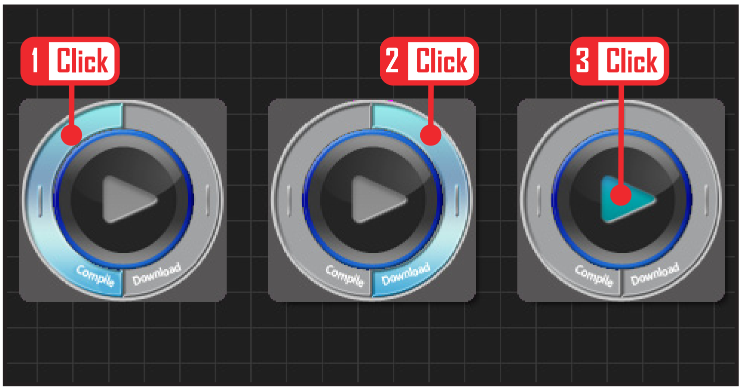
Input

□ True □ False



22 Entire Program

Robot detremines if it has fallen backwards or forward and runs the appropriate motion to get back up.

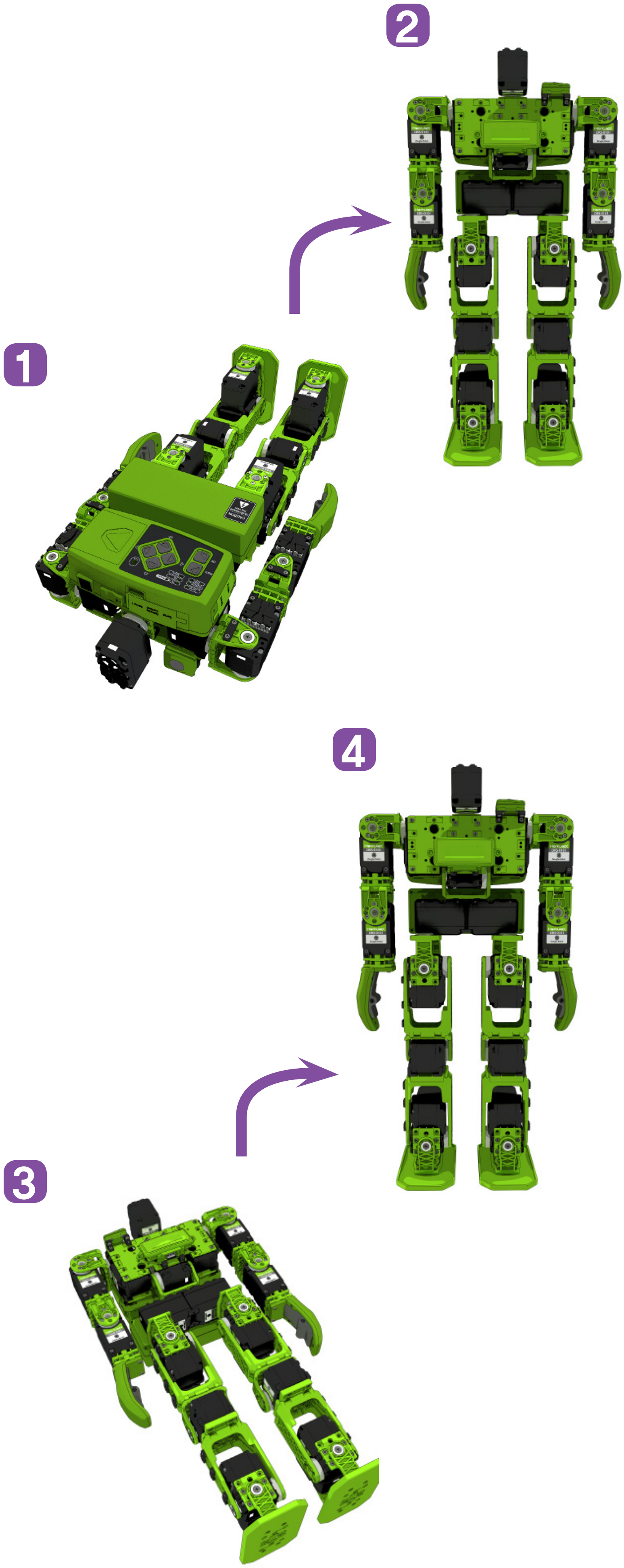


23 Compile, Download, Run

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

24 Robot Motion

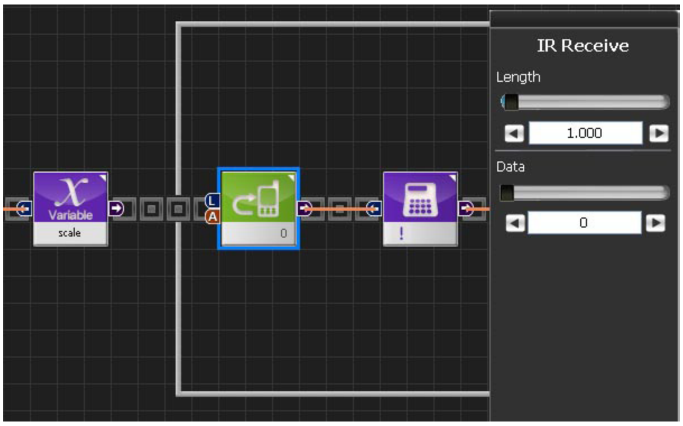
If the robot is in prone position, it gets back up backwards. If it is in supine position, it gets back up forward.



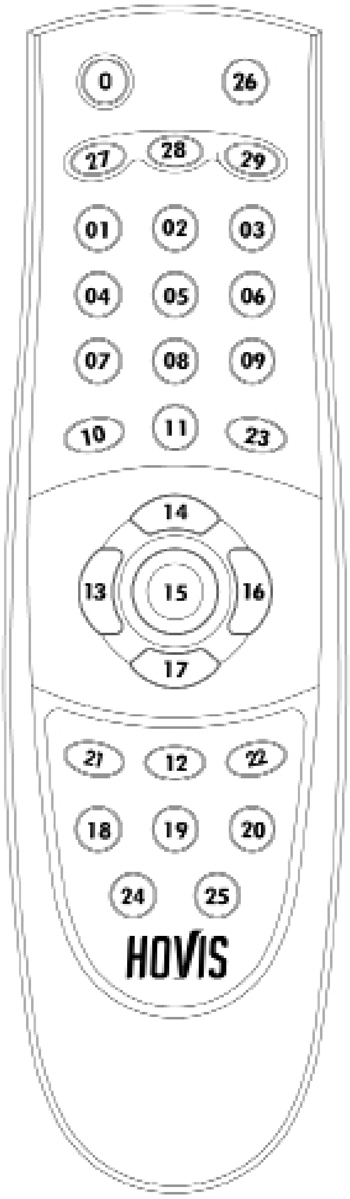
IRReceive, Sound & Motion Example Step by Step

(Explain by Sound examples, skip the explaination of motion examples, Data Match for Remote Controller)

Data figure from IR Recieve Module shall match the key from on the right side of remote controller.



Length
1.000
Data
0

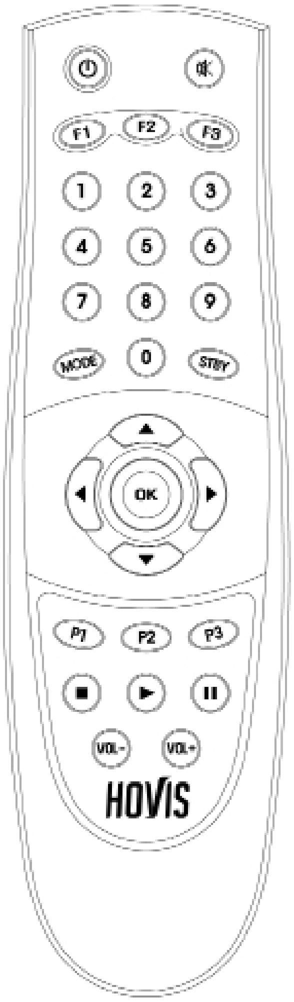


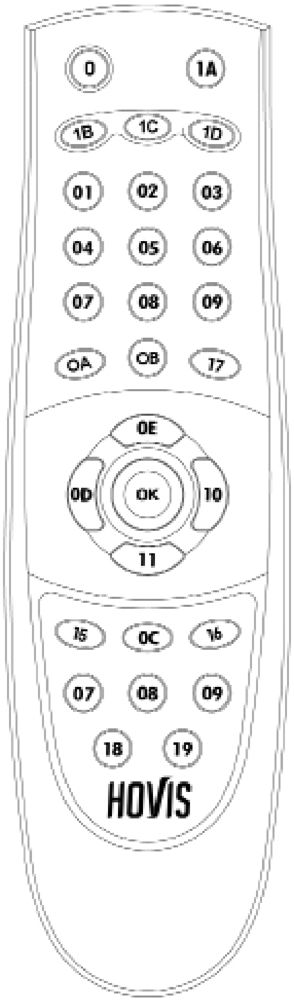
Hovis remote control keymap is as shown in the picture to the right. IR Receive module data values correspond to numbers in the right key.

For example, if the top right power button is pressed, Data 0 is received by the DRC. Robot can be programmed to take certain action when ever the power button is pressed by setting the Data to 0 in the IR RECEIVE module and connecting to Switch module input

Channel setting

Bothe the Remote control channel and DRC channel is user selectable but selected channel in DRC must match the remote controller channel in order for DRC to receive data from the remote control. Remote control channel can be selected by pressing 1~0 number + OK button simultaneously. DRC channel is selected by changing the RmcChannel value in MPSU Ram Data. RmcChannel values corresponding to remote control numbers are as follows.



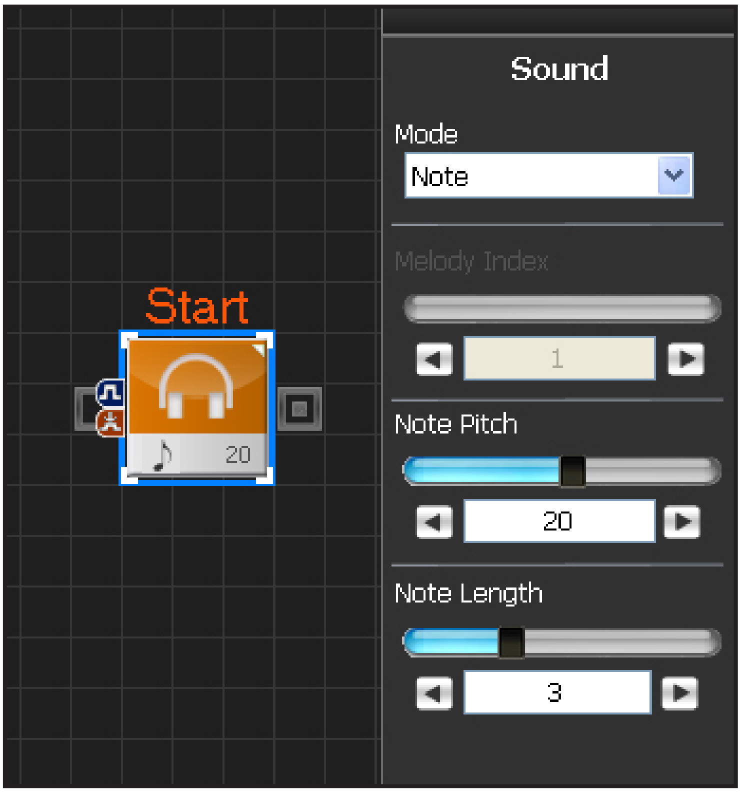


Remote Control Button	RmcChannel Value
0+OK	97(0x61)
1+OK	98(0x62)
2+OK	99(0x63)
3+OK	100(0x64)
4+OK	101(0x65)
5+OK	102(0x66)
6+OK	103(0x67)
7+OK	104(0x68)
8+OK	105(0x69)
9+OK	106(0x6A)

Example Description

This example associates remote control number button to a music note and outputs Do,Re,...Do (1~8) notes.

Note pictch is dependent on the value of the Note Pitch in Motion 〉Sound module. DRC controller has total of 38 pitches from 0~37 and it is able to ouptut total of 3 octaves.



Sound

Mode: Note

Melody Index: 1

Note Pitch: 20

Note Length: 3

00 Sound Property Window

Select Motion 〉Sound module.
Mode has Melody & Note. Melody selects and plays one of the saved edited notes.
Note Mode is selected to use the 36 note pitches.
Refer to the table below

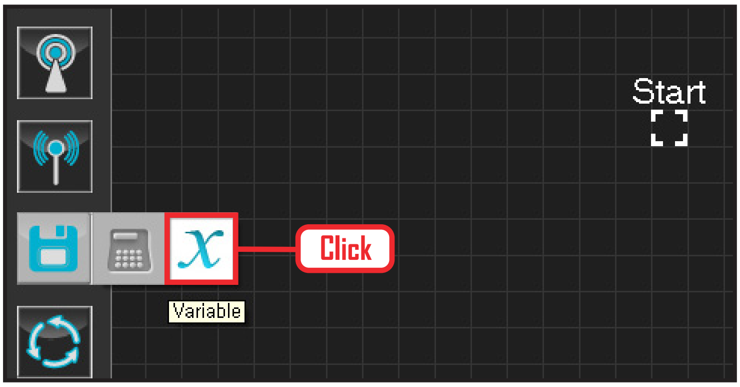
Note Pitch from 0~37 can be selected. Note pitches comprise total of of 3 octaves.
Note Length refers to the beat. Thirty-second note to the whole note can be selected.
Refer to the table below

Note Pitch

No.	0												
Note	NA												
No.	1	2	3	4	5	6	7	8	9	10	11	12	
Note	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si	
No.	13	14	15	16	17	18	19	20	21	22	23	24	
Note	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si	
No.	25	26	27	28	29	30	31	32	33	34	35	36	37
Note	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si	Do

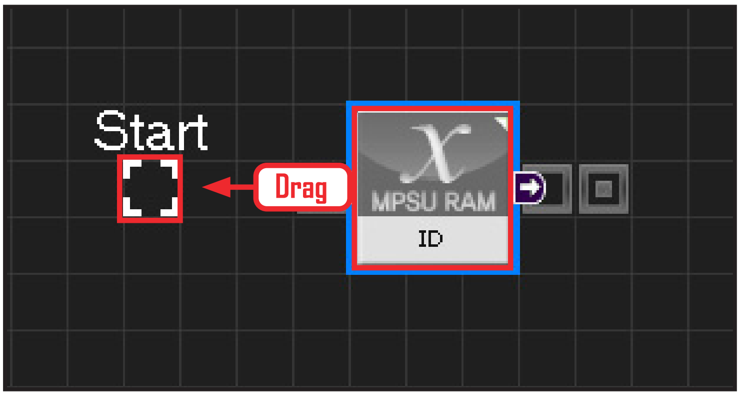
Note Length

No.	0	1	2	3	4	5	6	7	8	9
Raw Data	6	12	18	24	36	48	72	96	144	192
(ms)	38.4	76.8	115.2	153.6	230.4	307.2	460.8	614.4	921.6	1228.8
Note	32 note	16 note	16 dot note	8 note	8 dot note	4 note	4 dot note	2 note	2 dot note	Whole note



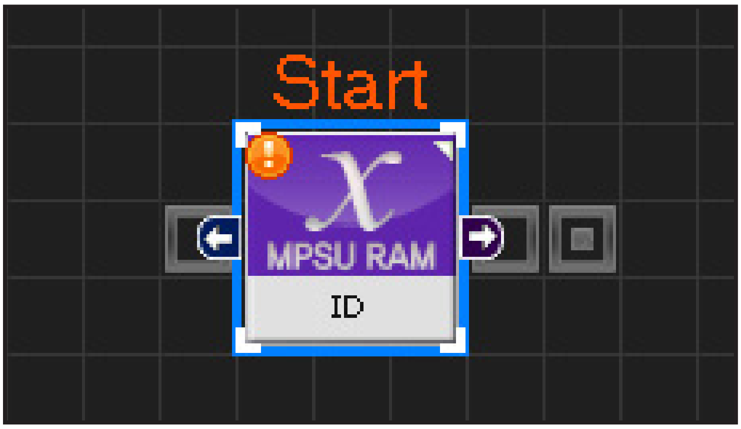
01 Assign Variable

Select Data 〉 Variable module.



02 Start

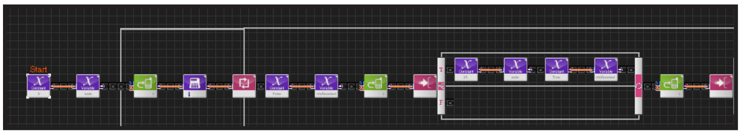
Click and drag the connecting line located at left side of the module to the Start Point and dock



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..

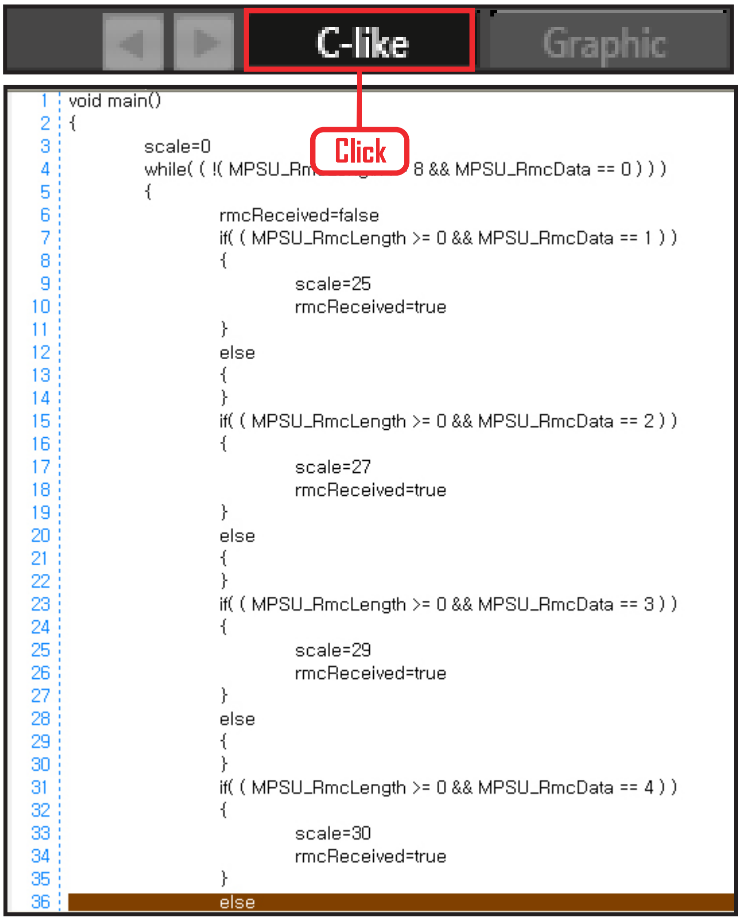
04 Entire Program



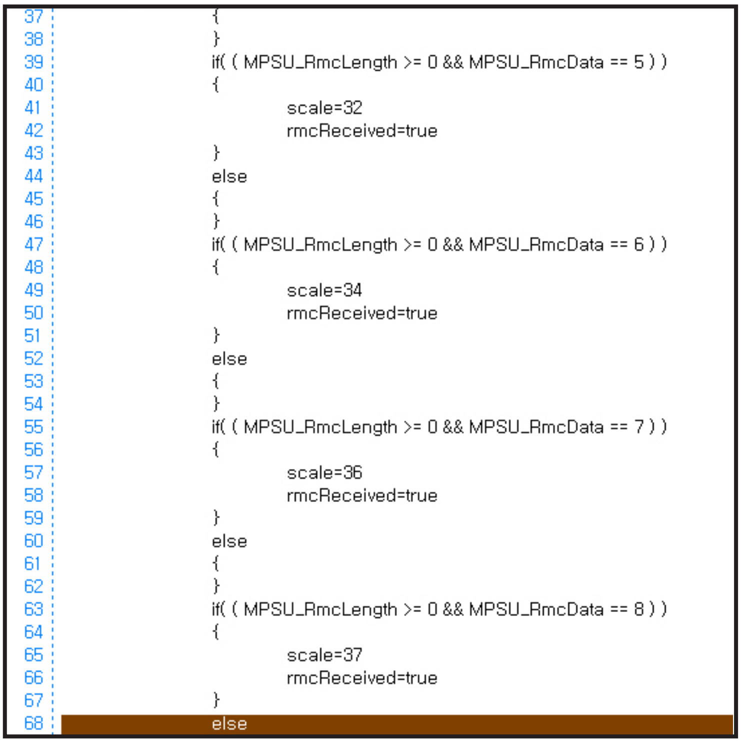
Entire program using the remote control and the buzzer.

05 Viewing C-Like

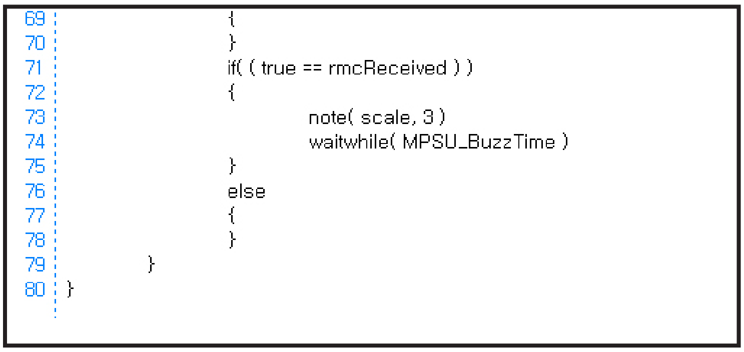
Click the 'C–like' tab near the top right and task programming window will open as shown in the photo to the left. This is the task window of the entire program. Codes are very similar to the C language structure so studying the codes will help the user become familiar with the C language structure.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



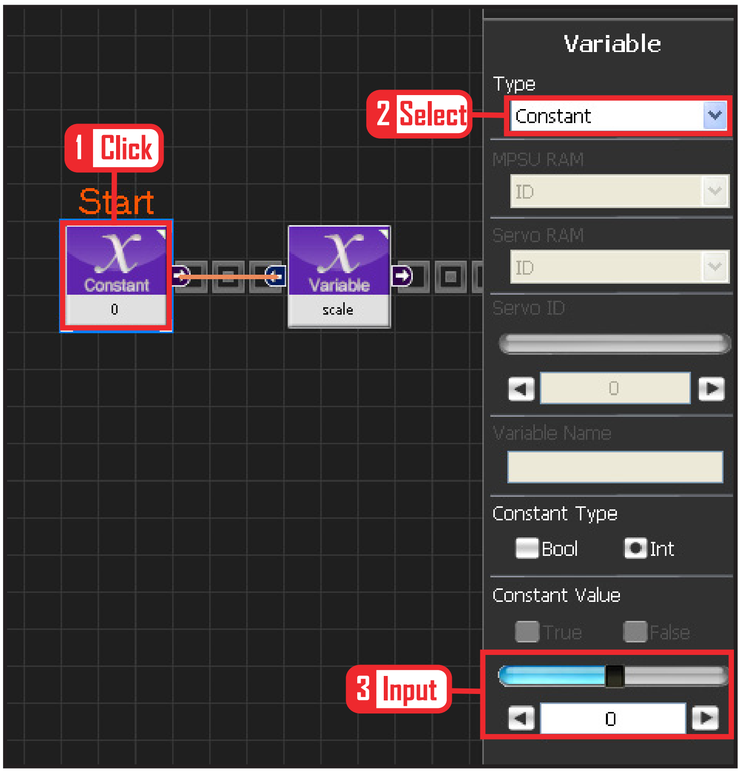
```
void main()
{
        scale=0
        while( ( !( MPSU_Rm          8 && MPSU_RmcData == 0 ) ) )
        {
                rmcReceived=false
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 1 ) )
                {
                        scale=25
                        rmcReceived=true
                }
                else
                {
                }
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 2 ) )
                {
                        scale=27
                        rmcReceived=true
                }
                else
                {
                }
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 3 ) )
                {
                        scale=29
                        rmcReceived=true
                }
                else
                {
                }
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 4 ) )
                {
                        scale=30
                        rmcReceived=true
                }
                else
```



```
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 5 ) )
        {
            scale=32
            rmcReceived=true
        }
        else
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 6 ) )
        {
            scale=34
            rmcReceived=true
        }
        else
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 7 ) )
        {
            scale=36
            rmcReceived=true
        }
        else
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 8 ) )
        {
            scale=37
            rmcReceived=true
        }
        else
```



```
        {
        }
        if( ( true == rmcReceived ) )
        {
            note( scale, 3 )
            waitwhile( MPSU_BuzzTime )
        }
        else
        {
        }
    }
}
```



Variable

Type: Constant

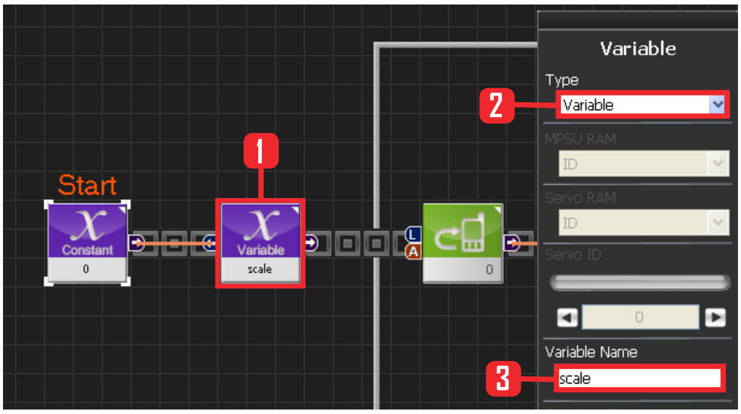
Constant Type: □ Bool ● Int

Constant Value: 0

06 Setup Constant

Declare variable of the scale to be played.

Select Data 〉 Variable .
Select Type : Constant .
Set Constant Value : 0 .



Variable

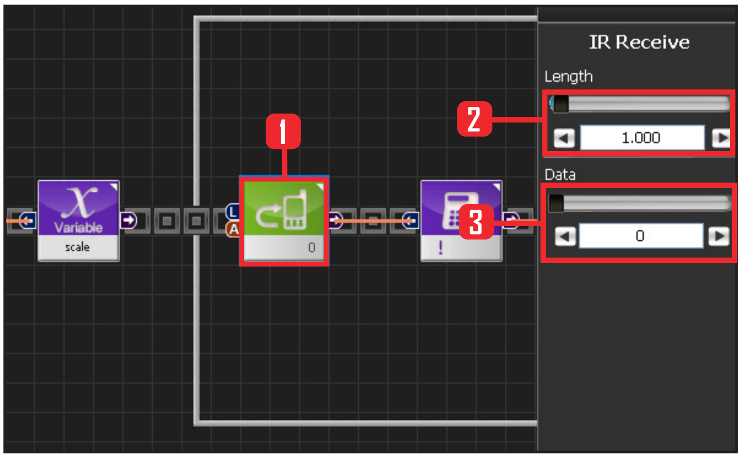
Type: Variable

Variable Name: scale

07 Variable Name

Declare the name of the scale variable to be played.

Select Data 〉 Variable .
Select Type : Variable .
Set Variable Name : scale .



IR Receive

Length: 1.000

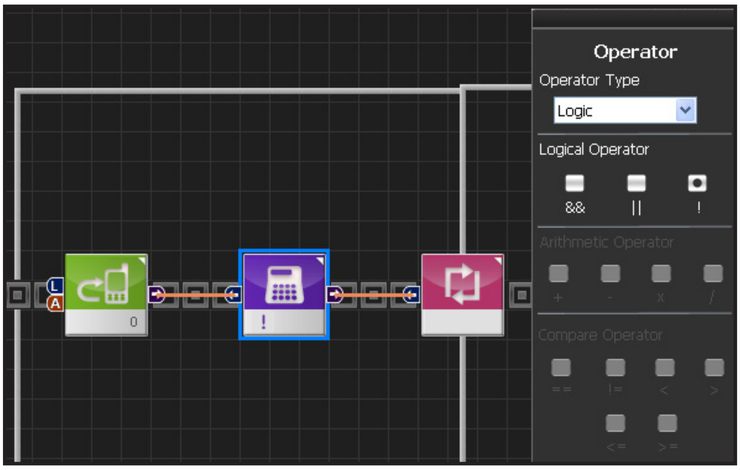
Data: 0

08 While Statement Exception

Exits if remote control button 0 is pressed loger than set time.

Select Communication 〉 IRReceive module.
Set Length : 1.000 . 1s button press.
Set Data : 0 . Power button press.

When the power button is pressed longer than 1s, output of the module is True. False if less than 1s.



Operator

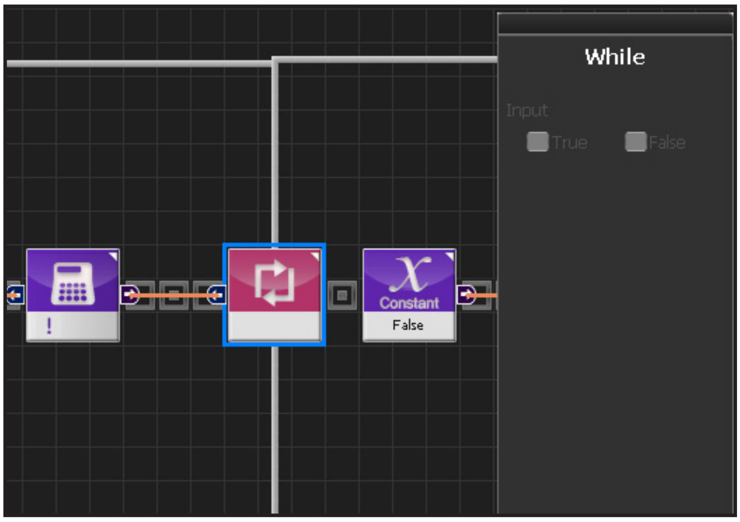
Operator Type: Logic

Logical Operator: ● !

09 Setup ! operator

! converts true / false value to opposite. Output value of IRReceive module is converted to opposite value and used as input value of the while statement.

10 While Loop



While

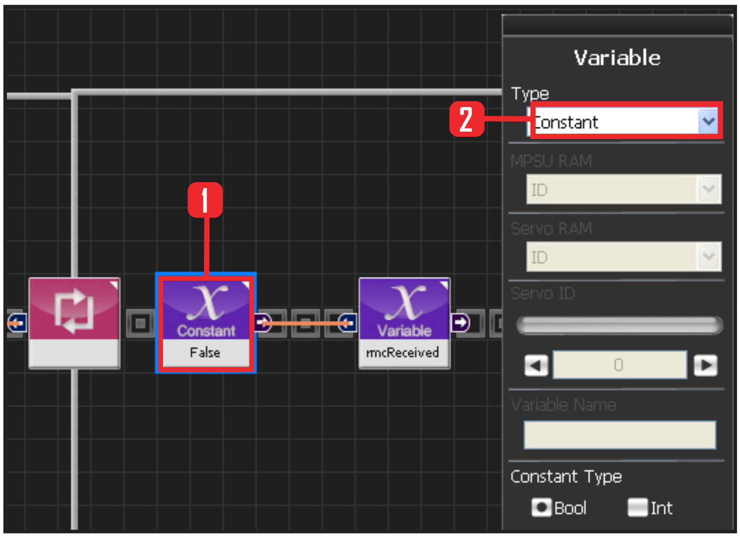
Input

□ True □ False

Repeat depending on previous condition.

If True, continue to repeat next step.By going through the ! operator, repeat if the ouput value of the IRReceive module is false, exit loop if true. Exit loop if the power button is presed longer than 1s.

11 Initialize Remote Control Input Variable.



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name:

Constant Type: ● Bool □ Int

Select variable showing that remote control input was received.

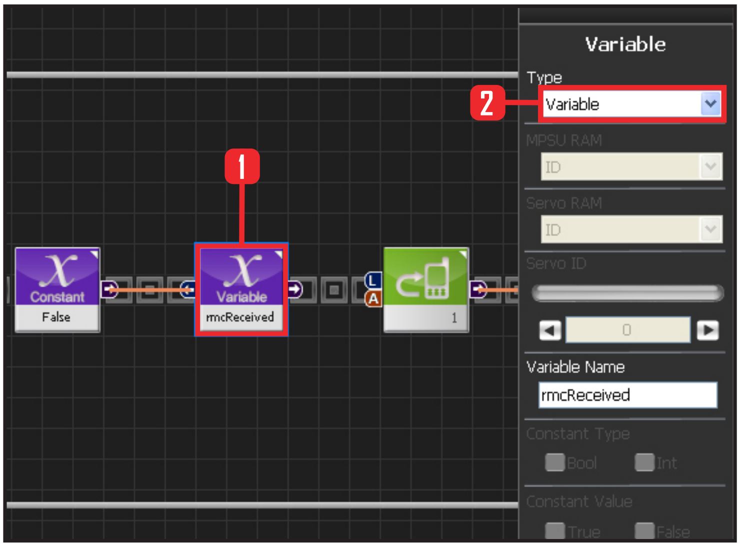
Select Data > Variable module.

Select Type : Contant .

Select Constant Type: Bool: True or False data type

Select Constant Value : False

12 Remote Control Input Initial Variable.



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: rmcReceived

Constant Type: □ Bool □ Int

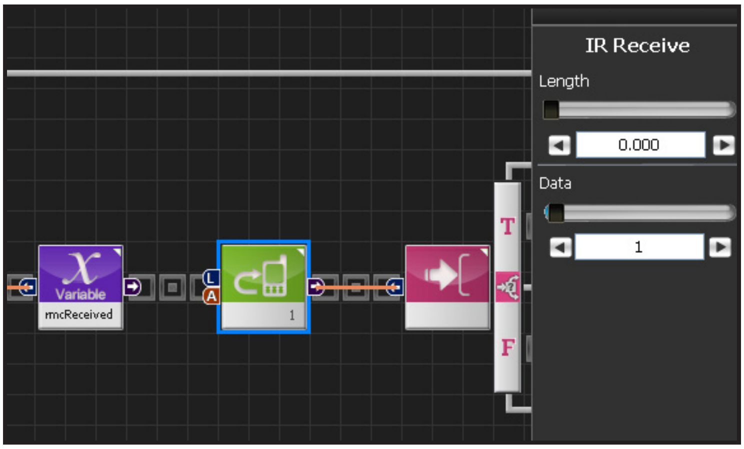
Constant Value: □ True □ False

Select Data > Variable .

Select Type : Variable .

Variable Name : rmcReceived

rmcReceived is a variable showing that remote control button 1~8 input was received within the loop. Intitalized as False at beginning of the loop. Play note if the checked value towards the end of the loop is True.



IR Receive

Length

0.000

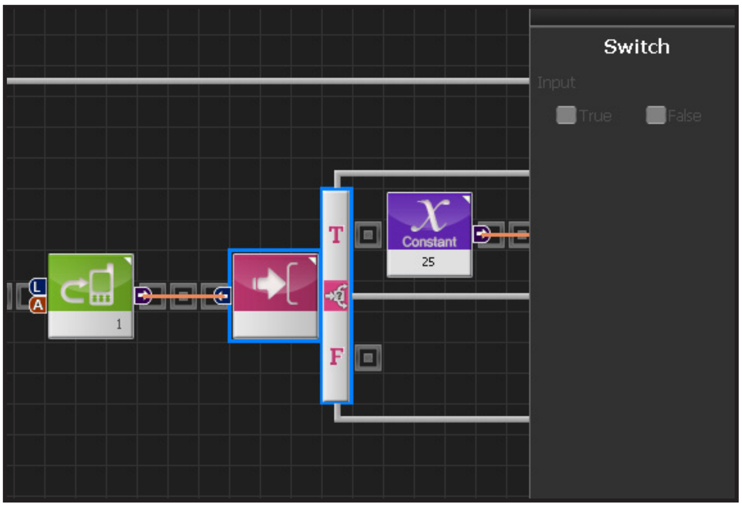
Data

1

13 Remote Control Button 1

Check if remote control button 1 was pressed.

Select Communication〉IRReceive module
Set Length : 0.000 .
Set Data : 1 . Refers to Button1.



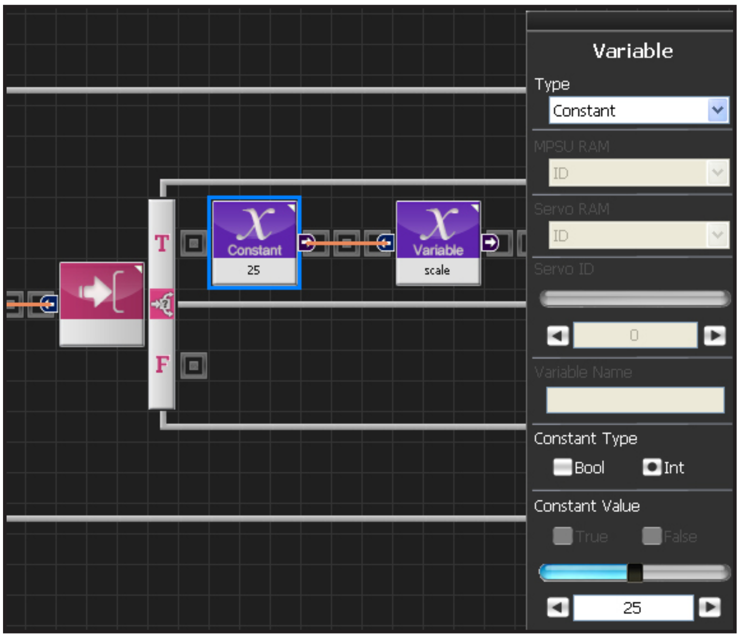
Switch

Input

□True □False

14 IF Conditional Statement

Run if True.



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name

Constant Type

□Bool ●Int

Constant Value

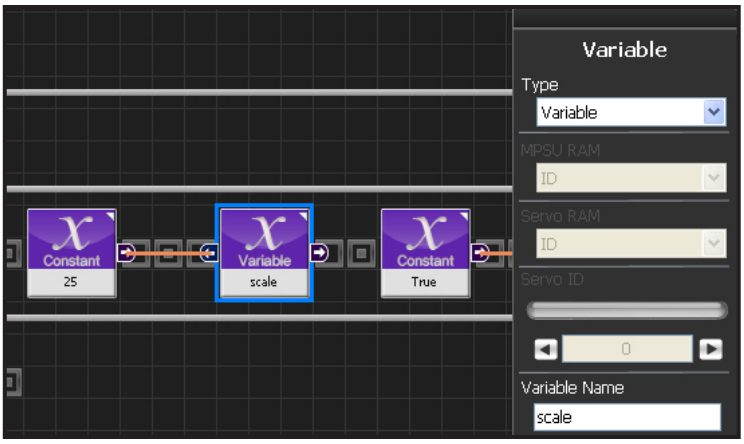
□True □False

25

15 Save "Do" Note

As explanined previously, Note Pitch (3 octaves) number 25 referst to 'Do' note.
Change the Scale value to 'Do'

Select Data〉Variable module.
Select Type : Contant
Select Constant Type: int.
Set Constant Value : 25 . 25 refers to "Do".



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

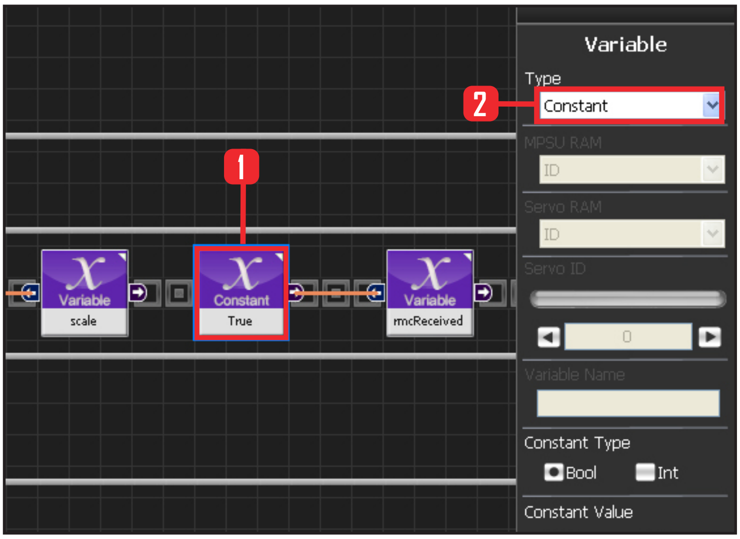
Variable Name: scale

16 scale

Declare variable name of the scale to be playes as Scale.

Select Data 〉 Variable .
Select Type : Variable .
Set Variable Name : scale .

Receive previous constant value 25 using input connecter .



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name

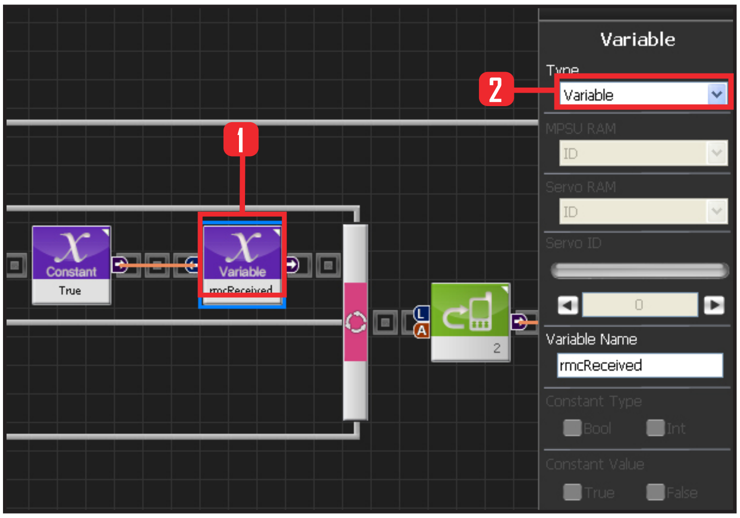
Constant Type: ● Bool □ Int

Constant Value

17 Save Remote Control Input Confirm Valule

If rmcRecieved value is True, it denotes one of the remote control button (1~8) was pressed.

Select Data 〉 Variable module.
Select Type : Contant .
Select Constant Type : Bool .
Select Constant Value : True .



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: rmcReceived

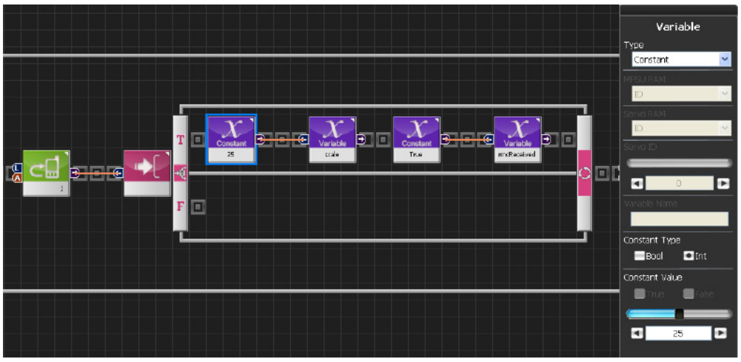
Constant Type: □ Bool □ Int

Constant Value: □ True □ False

18 Save Remote Control Input Confirm Value

Select Data 〉 Variable .
Select Type : Variable .
Set Variable Name : rmcReceived.

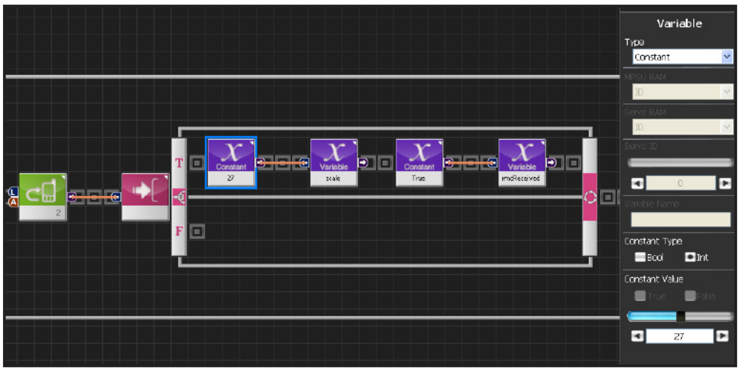
Receive previous connstant value True using input conntor.



Variable
Type: Constant
Constant Type: Bool ● Int
Constant Value: 25

19 1 -> “Do” Note

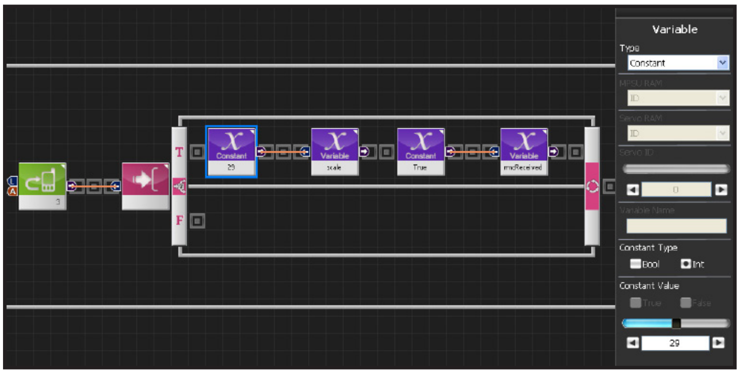
Program saves note ‘Do’ in the scale when reomote control button 1 is pressed.



Variable
Type: Constant
Constant Type: Bool ● Int
Constant Value: 27

20 2 -> “Re” Note

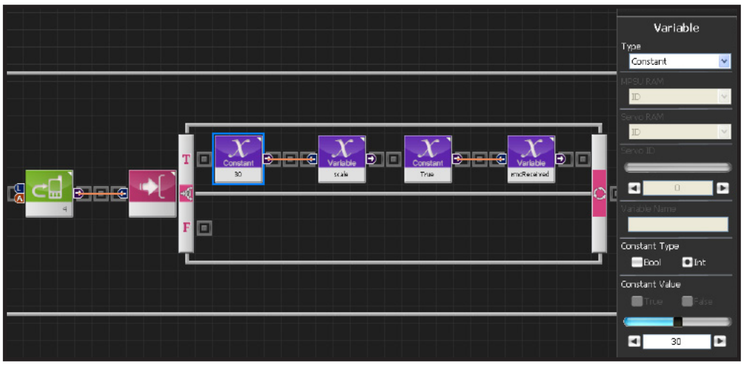
Program saves note ‘Re’ in the scale when reomote control button 2 is pressed.
Scale = No 27 is ‘Re’ note.



Variable
Type: Constant
Constant Type: Bool ● Int
Constant Value: 29

21 3 -> “Mi” Note

Program saves note ‘Mi’ in the scale when reomote control button 3 is pressed.
Scale = No 29 is ‘Mi’ note.



Variable

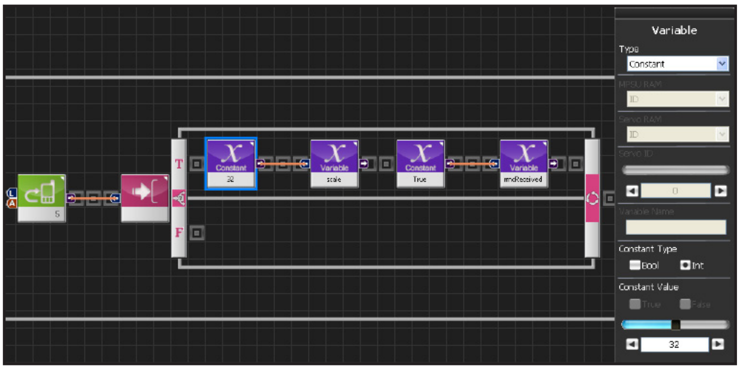
Type: Constant

Constant Type: Bool Int

Constant Value: 30

22 4 -> “Fa” note

Program saves note ‘Fa’ in the scale when reomote control button 4 is pressed.
Scale = No 30 is ‘Fa’ note.



Variable

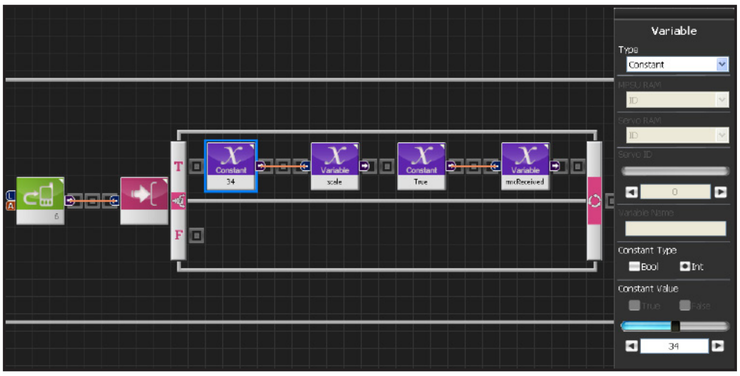
Type: Constant

Constant Type: Bool Int

Constant Value: 32

23 5 -> “Sol” Note

Program saves note ‘Sol’ in the scale when reomote control button 5 is pressed.
Scale = No 32 is ‘Sol’ note



Variable

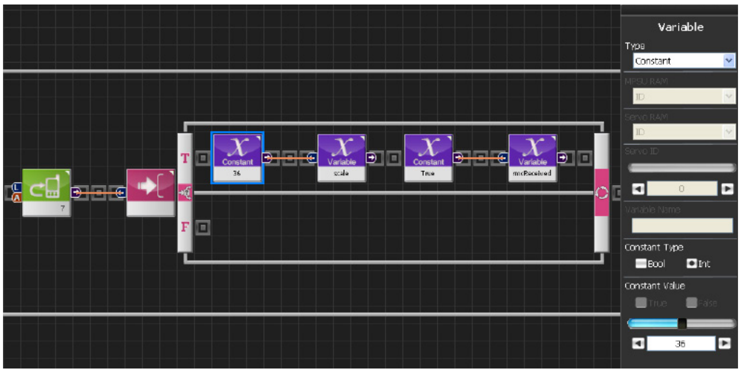
Type: Constant

Constant Type: Bool Int

Constant Value: 34

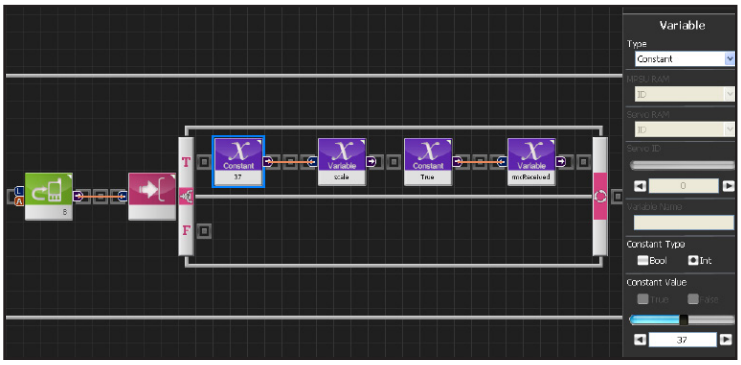
24 6 -> “Ra” Note

Program saves note ‘Ra’ in the scale when reomote control button 6 is pressed.
Scale = No 34 is ‘Ra’ note .



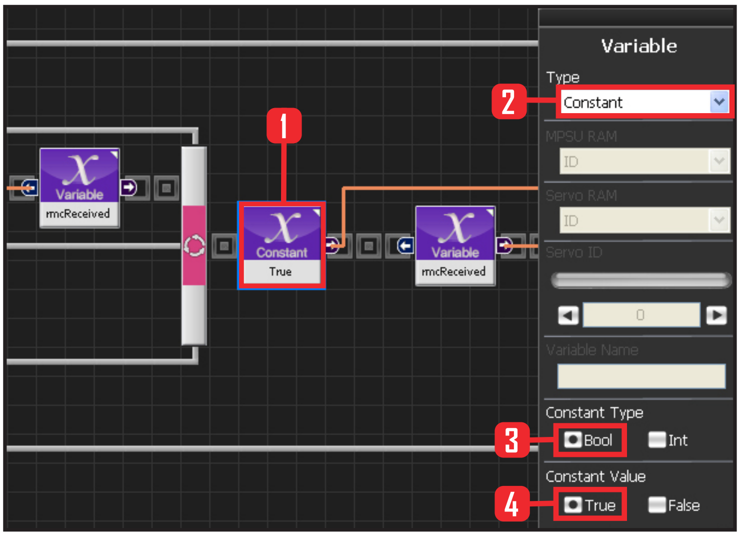
25 7 -> "Si" Note

Program saves note 'Si' in the scale when reomote control button 7 is pressed.
Scale = No 36 is 'Si' note.



26 8-> "Do" Note

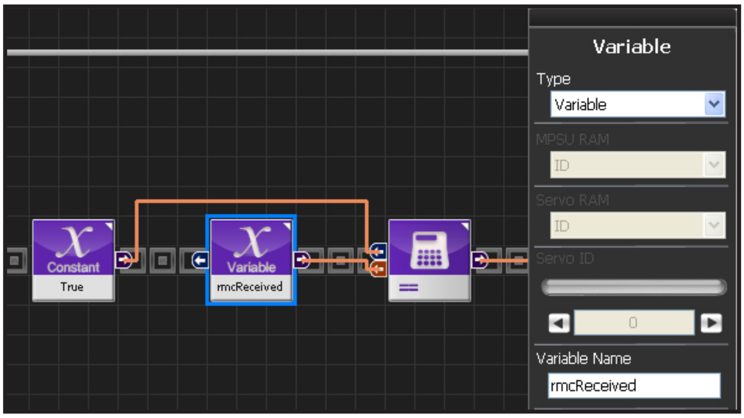
Program saves note 'Do' in the scale when reomote control button 8 is pressed.
Scale = No 37 is 'Do' note .



27 Whe rmcReceived is True

When rmcReceived is True, input saved scale value where pitch value was previously saved into note to ouput note.

Select Data 〉 Variable module.
Select Type : Contant .
Select Constant Type: Bool.
Select Constant Value : True.



Variable

Type: Variable

MPSU RAM: ID

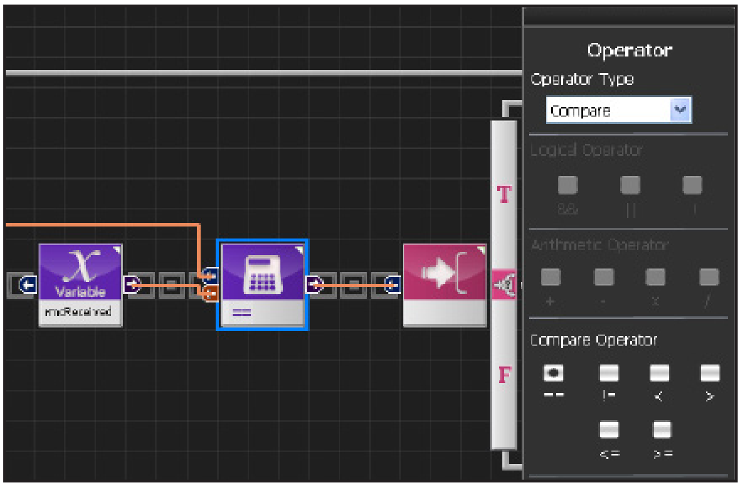
Servo RAM: ID

Servo ID: 0

Variable Name: rmcReceived

28 When rmcReceived is True

rmcReceived variable name is identical.



Operator

Operator Type: Compare

Logical Operator: && || !

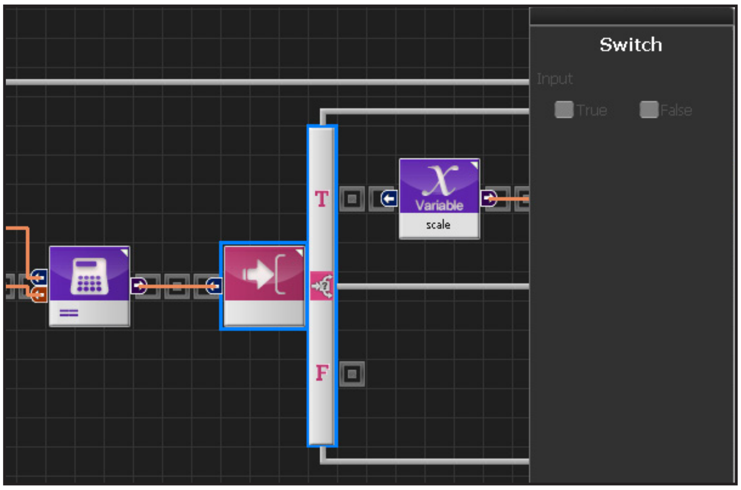
Arithmetic Operator: + - x /

Compare Operator: ● == □ != □ < □ > □ <= □ >=

29 Comparison Operator ==

Select Data 〉 Operator module
Select Operator Type : Compare .
Select Compare Operator: == .

rmcReceived = = refers to true ,
shows “rmcReceived is equal to true .

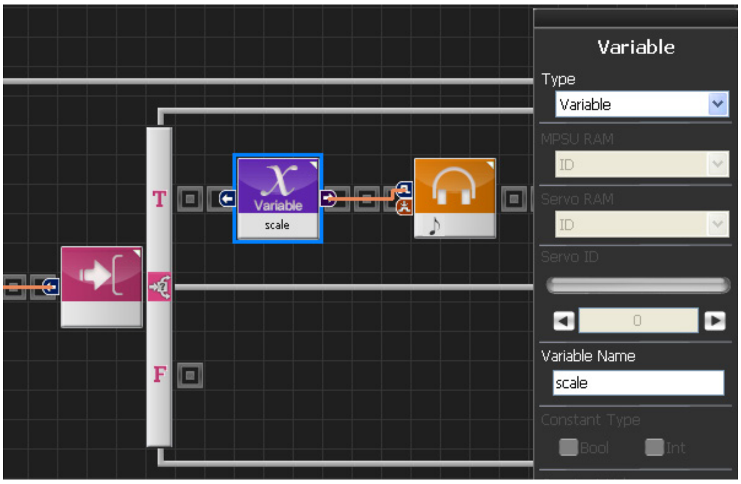


Switch

Input: □ True □ False

30 Switch IF Conditional Statement

Run if True.



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

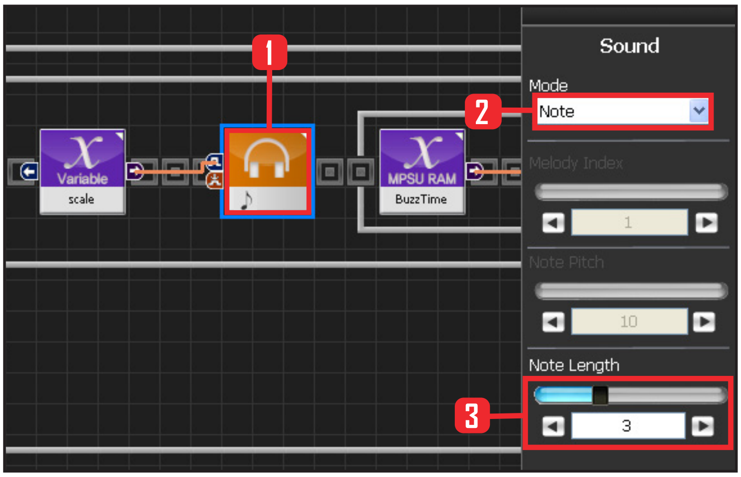
Variable Name: scale

Constant Type: Bool Int

31 scale -> note

Input Scale value into Note.

Make variable scale module.



Sound

Mode: Note

Melody Index: 1

Note Pitch: 10

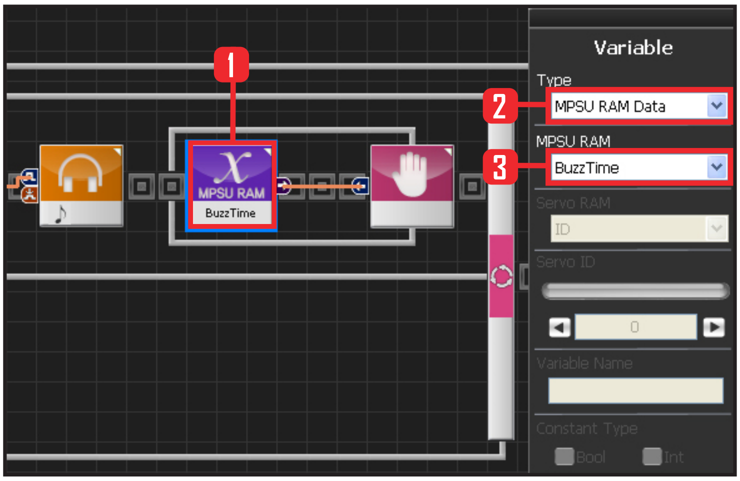
Note Length: 3

32 Sound Play

Input Scale value into note to play sound.

Select Motion 〉 Sound module.
Set Note Length : 3. detnotes eighth note. Lasts 153.6ms .

Different scale values were saved depending on the input from the remote control buttons. When the scale value is recevied by Note Pitch correspoding note will play.



Variable

Type: MPSU RAM Data

MPSU RAM: BuzzTime

Servo RAM: ID

Servo ID: 0

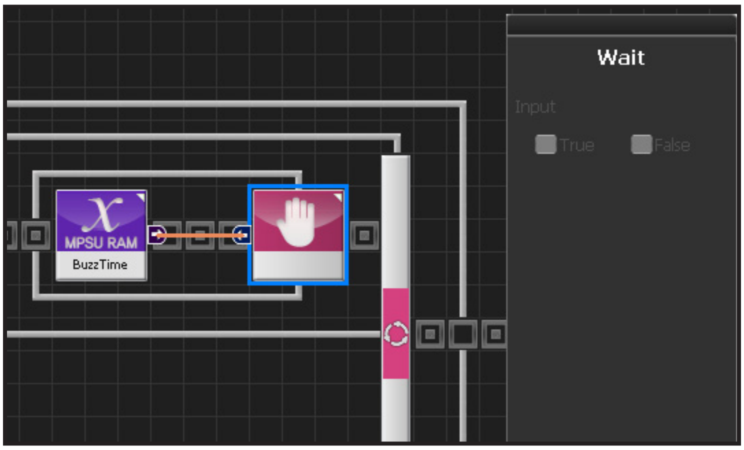
Variable Name

Constant Type: Bool Int

33 BuzzTime

Buzz Time in MPSU RAM Data decides if the note is playing and waits.
When buzzer starts to sound, BuzzTime acquires certain value which decreases by 1 every 6.4ms. If the value is other than 0, buzzer is still sounding and if the value is 0, buzzer has stopped. Refer to 'Raw Data' in note length table for initial BuzzTime values.

Select Data 〉 Variable .
Select Type : MPSU RAM Data .
Select MPSU RAM : BuzzTime .



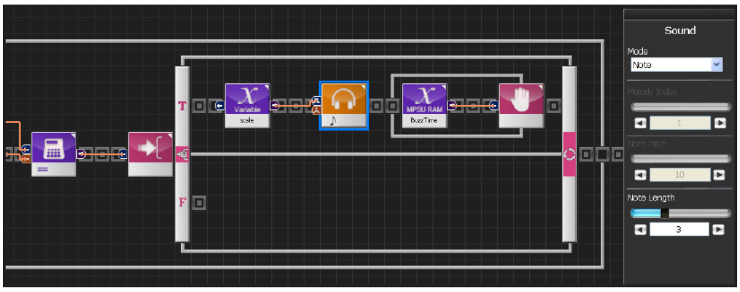
Wait

Input

□ True □ False

34 Wait

Wait untill Buzztime value becomes 0, In othe words, wait untill the sound ends.



Sound

Mode: Note

Melody Index: 1

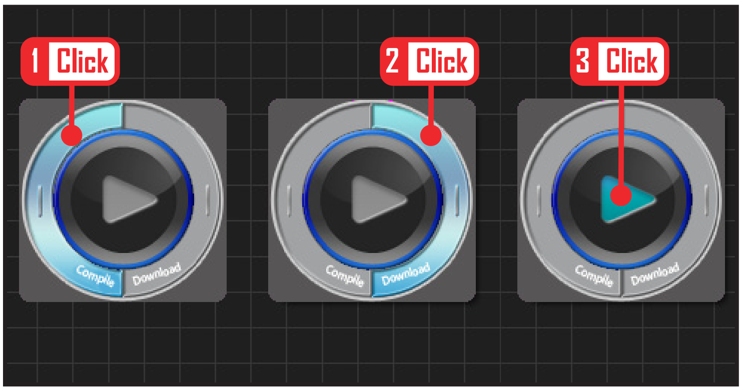
Note Pitch: 10

Note Length: 3

35 Note Output Process

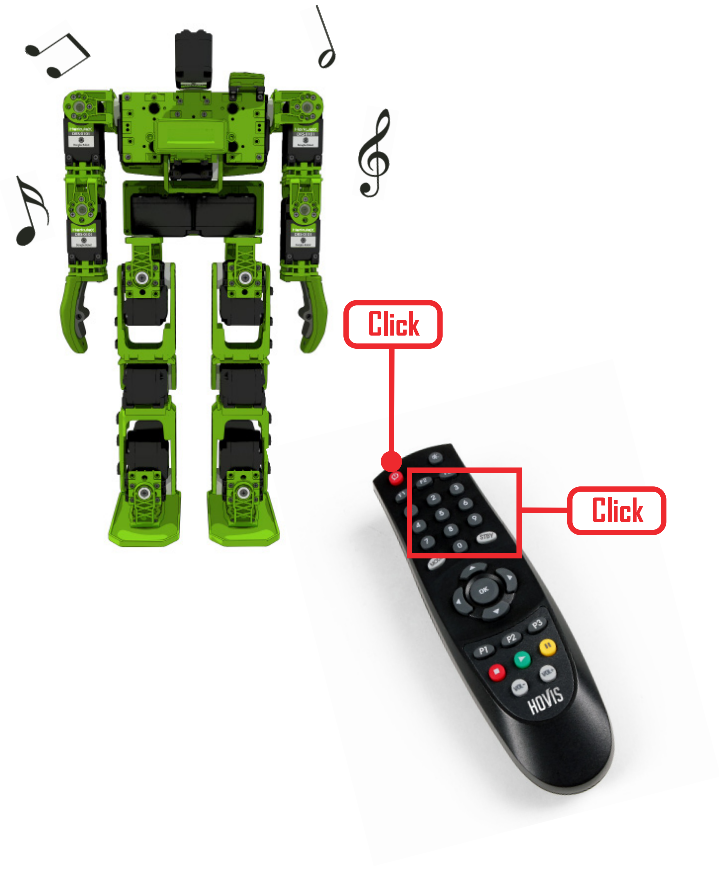
When rmcReceived is True, value saved in scale is used as input to Sound module which then outputs corresponding note.

BuzzTime to checks the end of the note and goes back to the begining.



36 Compile, Download, Run

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



37 Robot Motion

Press Remote control buttons(1~8) to play notes. End task by pressing the power button for more than 1s.

Appendix

Register

Non-Volatile Register Map

Controller registers contain current status and opertational data of the controller and it is comprised of Non-Volatile and Volatile registers. Reading or changing the register data using command protoclols eable the user to control the controller and use the DR-Visual Logic to program the robot.

Non-Volatile Register (EEP Register) Map

Non-Volatile registers retain data even when the controller power has been turned off and contain basic values pertaing to the controller operation. Values in the Non-Volatile registers are copied to the volatile registers as soon as the controller power is turned on. Any changes made to the Non-Volatile registers will not afffect the opertaion of the robot untill the changed values are copied to the Volatile registers after reboot or after power has been turned off and back on.

- **Address**

  Address refers to the register address. In order to read/write to the register, packet must contain the relevant register address.
- **Default**

  Factor default values. Rollback command is used to the change the cotents of the Non-Volatile regsters back to factory default values.
- **Valid Range**

  Valid data range register can have. Error will occur when the data is being copied to the the Volatile register if the input data exceeds valid rage and the data will be turncated to fit within the valid range of the volatile register.
- **RW**

  RO(Read Only) refers to registers where data can only be read from but not written to. Error will occur if an attempt is made to write to the RO registers. RO Registers contain such data as the controller model number, firmware version, and sensor data. RW registers can be read from and written to.

※ e(Reg_Name) : Refers to Reg_Name of Non-Volatile Register(EEP Register)

※ r(Reg_Name) : Refers to Reg_Name of Volatile Register(RAM Register).

Addr	Type	Bytes	Default	Valid Range	RW	Comments
0	Model No1	1	0x05	–	RO	Controller mode No.
1	Model No2	1	0x54	–	RO	
2	Version1	1	0x01	–	RO	Firmware version
3	Version2	1	0x22	–	RO	
4	Baud Rate	1	0x10	Refer 08page	RW	PC–Controller, Com speed between Controller–Servo
5	Special Function	1	0x00	0x00~0xF3	RW	Flag for using DRC for special function
6	Reserved	1	0x00	–	–	
7	ID	1	0xFD	0x00~0xFD	RW	Controller ID(0xFE: Can be used as Broadcasting ID) ID cannot be assigned)
8	Ack Policy	1	0x01	0x00~0x02	RW	Reply to packet according to policy
9	Torque Off Policy	1	0x03	0x00~0x7F	RW	Torque off according to policy
10	Alarm LED Policy	1	0x7F	0x00~0x7F	RW	Alarm LED blink accordking to policy
11	Status Check Policy	1	0x01	0x00~0x01	RW	Decide whether to check value of servo angle
12	Min. Voltage	1	0x5F	0x00~0xFE	RW	Minimum voltage(0x5F : 7.1V)
13	Max. Voltage	1	0x88	0x00~0xFE	RW	Maximum voltage(0x88 : 10.0V)
14	Max.Temperature	1	0xDF	0x00~0xFE	RW	Max temperature(0xDF : 85 ˚ C)
15	Remocon Channel	1	0x61	0x61~0x6A	RW	IR remote control channel code
16	Servo Ack Wait Tick	1	0x04	0x00~0xFE	RW	Minimum wait time for Servo Ack (0x04 : 6.4ms)
17	Zigbee Ack Wait Tick	1	0x50	0x00~0xFE	RW	Zigbee Ack wait time (0x50 : 128ms)
18	LED Blink Period	1	0xBB	0x00~0xFE	RW	Warning LED blink peirod(0xBB : 300ms)
19	ADC Fault Check Period	2	0x0138	0x0000 ~ 0x7FFF	RW	Temperature/Voltage Error Detection Period (0x0138 : About 500ms)
21	Packet Garbage Check Period	2	0x007D	0x0000 ~0x7FFF	RW	Packet Corruption Detection Period (0x7D : About 200ms)

Address 0–6 contains basic controller and communications data . Address 7–22 contains controller function data. Data in address 7–22 are copied to Volatile register when the controller is rebooted.

Volatile Register Map

Volatile Register(RAM Register MAP)

Volatile Register contains controller operation settings, controller status, and sensor data values. Data values contained in the Volatile registers have direct influence on operation of the controller. Rebooting the controller initizalizes the data in the Volatile register. Even if the register values were changed to change the controller settings, values in the Volatile registers will revert back to the initial setting when the controller is rebooted.

Addr	Type	Bytes	Valid Range	RW	Comments
0	ID	1	0x00~0xFD	RW	Data copied from non-volatile register when controller is booted.
1	Ack Policy	1	0x00~0x02	RW	
2	Torque Off Policy	1	0x00~0x7F	RW	
3	Alarm LED Policy	1	0x00~0x7F	RW	
4	Status Check Policy	1	0x00~0x01	RW	
5	Min. Voltage	1	0x00~0xFE	RW	
6	Max. Voltage	1	0x00~0xFE	RW	
7	Max. Temperature	1	0x00~0xFE	RW	
8	Remocon Channel	1	0x61~0x6A	RW	
9	Servo Ack Wait Tick	1	0x00~0xFE	RW	
10	Zigbee Ack Wait Tick	1	0x00~0xFE	RW	
11	LED Blink Period	1	0x00~0xFE	RW	
12	ADC Fault Check Period	2	0x0000~0x7FFF	RW	
14	Packet Garbage Check Period	2	0x0000~0x7FFF	RW	
16	Status Error	1	0x00~0x7F	RW	Status error, Refer to 11 page
17	Error Codes [0] ~ [4]	5	Refer to 52 page	RW	Most recent 5 error codes
22	LED Control	1	0x00~0x07	RW	LED value when running Task (0x01:Red, 0x02:Green, 0x04: Blue)
23	User Timer Tick	1	0x00~0xFF	RW	User configurable timer(100[ms]/tick)
24	Connected Program	1	0~3	RW	Currently connected PC program
25	Zigbee Channel	1	11~26	RW	Zigbee communication channel
26	Zigbee PANID	2	0x0000~0xFFFF	RW	WPAN ID ZigbBee belongs to
28	Zigbee SADDR	2	0x0000~0xFFFF	RW	Zigbee ID
30	Zigbee DSTADDR	2	0x0000~0xFFFF	RW	Matching Zigbee ID
32	Zigbee ACKREQ	1	0~2	RW	Decide whether to request ACK after RF communication
33	Zigbee BACKOFF	1	0~2	RW	Decide whether to apply Random delay after RF communication.
34	Servo Count	1	0~32	RO	Number of connected motors
35	Servo ID[0]~[32]	33	0x00~0xFE	RO	ID of each motor (0xFE means motor does not exist)

Addr	Type	Bytes	Valid Range	RW	Comments
68	Playing Motion	1	0x00~0x01	RO	Check whether Motion running
69	Playing Task	1	0, 1, 3	RO	Check whether Task running
70	Charger Connected	1	0~1	RO	Check whether charger connected
71	Buzzer Scale	1	0x00~0x25	RO	Buzzer ptich
72	Buzzer Time	1	0~192	RO	Buzzer sound time(6.4[ms]/tick)
73	Button Status	1	0x00~0x3F	RO	Button Status
74	Remocon Length	1	0~240	RO	Remote control button press ime(125[ms]/tick)
75	Remocon Data	1	0x00~0x1D,0xFE	RO	Remote control button number
76	Input Voltage Value	1	0x00~0xFE	RO	Input Voltage Raw Data, 8bit
77	Temperature Value	1	0x00~0xFE	RO	Current temperature Raw Data, 8bit
78	Light Sensor Value	1	0x00~0xFE	RO	Light sensor value Raw Data, 8bit
79	ADC Port 1 Sensor Type	1	0~2	RO	Sensor type connected to ADC port 1
80	ADC Port 2 Sensor Type	1	0~2	RO	Sensor type connected to ADC port 2
81	ADC Port 1 Sensor Value	2	0x0000~0xFFFF	RO	Sensor output value connected to ADC port 1
83	ADC Port 2 Sensor Value	2	0x0000~0xFFFF	RO	Sensor output value connected to ADC port 2
85	ACC/GYRO Connected	1	0~1	RO	Acc/Gyro sensor connection status
86	ACC X Value	2	-4096~4095	RO	Acc sensor X axis Raw Data, 13bit
88	ACC Y Value	2	-4096~4095	RO	Acc sensor Y axis Raw Data, 13bit
90	ACC Z Value	2	-4096~4095	RO	Acc sensor Z axis Raw Data, 13bit
92	GYRO X Value	2	-32768~32767	RO	Gyro sensor X axis Raw Data, 16bit
94	GYRO Y Value	2	-32768~32767	RO	Gyro sensor Y axis Raw Data, 16bit
96	GYRO Z Value	2	-32768~32767	RO	Gyro sensor Z axis Raw Data, 16bit
98	Sound Detection Flag	1	0~250	RO	Number of successive sound detections (Cleared after 1s)
99	Sound Direction	1	-2~2	RO	Direction of detected sound(- Left, + Right)
100	Reserved	1	-	-	Touch status value of connected buzzer module
101	Tick	2	0~60000	RO	System tick, 1.6[ms]/INT
103	DRT-HWW1 Connected	1	0~1	RO	DRT-HWWI connection status
104	DRC-004TO Connected	1	0~1	RO	DRC-004TO connection status
105	Reserved	1	-	-	Reserved
106	Servo Status Error & Detail [0]~[31]	64	0x00~0x80 * 64	RO	Status value of connected motor
170	Servo Position[0]~[31]	64	0x0000~0x7FFF	RO	Position value of connected motor
234	DRT-HWWI Status Error	1	0x00~0x80	RO	Status Error of DRT-HWW1 connected motor
235	DRT-HWWI Status Detail	1	0x00~0x7F	RO	Detailed Status DRT-HWW1 connected motor
236	DRT-004TO Status Error	1	0x00~0x80	RO	Status Error DRT-004TO connected motor
237	DRT-004TO Status Detail	1	0x00~0x7F	RO	Detailed Status DRT-004TO connected motor

Detailed Register Description

Model No 1, Model No 2(EEP Register 0, 1 Address)

DRC model name expressed in 2 byte binary format.Cannot be changed by the user.

Version 1, Version 2(EEP Register 2, 3 Address)

DRC firmaware version. If not the latest version, download and update from the website. Can not be changed by the user

Baud Rate(EEP Register Address #4)

Datat value determining the UART communication speed between the PC & DRC and DRC & DRS. Communication speed according to the data values are as follows. Communication speed will be set at default value of 115,200 bps if the data value entered is not in the value list below,

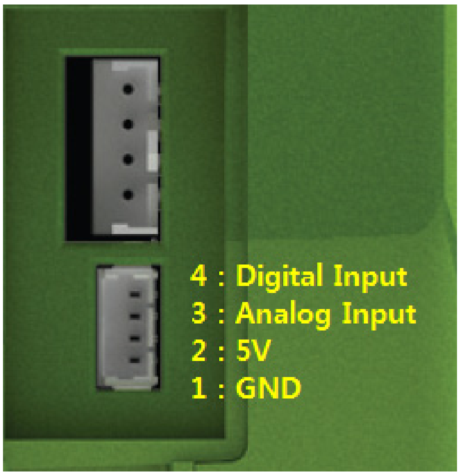
Baud Rate	Register Value
57,600	34
115,200	16
200,000	9
250,000	7
400,000	4
500,000	3
666,667	2

Special Function(EEP Register Address #5)

EEP Register Address #5 is used when DRC-005T is to be used for special function. Decision to use the special function is set by writing 1 or 0 to each bit. Default value is 0x00 (No functions used). Functions corresponding to each bit is shown below.

Bit	Value	Mode
0	0x01	Custom Sensor Mode
1	0x02	TTL Communication Mode
2	0x04	Reserved
3	0x08	Reserved
4	0x10	Reserved
5	0x20	Reserved
6	0x40	Reserved
7	0x80	Reserved

*** Custom Sensor Mode:** This mode is for using custom sensors with DRC-005T. DRC-005T has 4pin sensor ports on each side which can normally be used with only limited type of sensors. However, by using custom sensor mode, it is possible to connect other type of sensors to these ports providing sensors use 5V input power. Ports on each side can accept 1 analog and 1 digital sensor for total of 4 custom sensors (2 digital and 2 analog). Analog sensor values r(ADC Port 1 Sensor Value) and r(ADC Port 2 Sensor Value) are expressed in 10 bits (0~1023), digital sensor values r(ADC Port 1 Sensor Type) and r(ADC Port 2 Sensor Type) are expressed by 0~1. Sensor port pin map is as shown in the photo.



*** TTL Communication Mode:** PC and DRC-005T uses RS-232C ±5~10V communications level. However, it is possible to control the DRC-005T like a PC using 3.3V TTL level instead of RS-232C level by setting the DRC-005T communication mode to TTL communication mode. Zigbee connection pin is used to communicate with DRC-005T using TTL level. Zigbee connection pin is as shown below.

ID(EEP Register Address #7, RAM Register Address #0)

DRC ID. Default value is 253(0xFD). if several DRCs are given distinct IDs, it is possible to connect them to the same communications line and control them similar to controlling several DRSs. To prevent malfuction, each DRC connected to the same communications line should have distinct ID.

ACK Policy(EEP Register Address #8, RAM Register Address #1)

Data value determines whether to send ACK Packet when Request Packet is from PC to DRC.

- 0 : Do not send reply to any Request Packet.
- 1 : Send reply to only those Request Packets requesting reply such as Read Command and few others.
- 2 : Reply to all Request Packets.

※ When STAT Request Packet is received, send reply regardless of ACK Policy.

※ Do not reply to REMOCON Regardless of ACK Policy.

※ Do not reply when pID is 254(0xFE, Broadcast pID) with an exception of STAT command.

※ Refer to 22page for detailed explanation of response to individual ACK Packet ACK Policy.

Torque Off Policy(EEP Register Address #9, RAM Register Address #2)

Determines whether to release(off) the torque to the connected servo motors when error is detected.

- (r(Torque Off Policy) & r(Status Error)) is True, all connected servo motors will have the torque released(off). Servos with torque off will not be able to move.
- r(Status Error) Error state has to be cancelled first to turn the motors back to Torque On state.

※ & is a Bitwise AND operator.When peforming A & B operation, binary representation of A & B are compared and the result is shown as 1 only if both A and B has 1 in the binary format. Exampe) 00101110 & 10110110 = 00100110이

Alarm LED Policy(EEP Register Address #10, RAM Register Address #3)

Determines whether to blink warning LED when error detected.

- (r(Alarm LED Policy) & r(Status Error)) is True, TX, RX, Spare LED on controller will blink and the warning LED blinkd perid is determined by the r(LED Blink Period).
- Original function of the TX, RX, Spare LED will be ignored while the LEDs are blinking error warning.
- r(Status Error) has to be cancelled first in order for TX, RX, Spare LED to return to their normal function.

Status Check Policy(EEP Register Address #11, RAM Register Address #4)

Determines whether controller should continuouisly read the current servo position. When r(Status Check Policy) is set at 1, Controller will continuously read the current servo position and servo status and update the r(Servo Status Error & Status Detail[0]~[31]) and r(Servo Position[0]~[31]). Controller will not perform the update if r(Status Check Policy) is set at 0.

Minimum Voltage(EEP Register Address #12, RAM Register Address #5)

Refers to minimum input voltage Raw Data. If the DRC input voltage r(Input Voltage Value) is below r(Minimum Voltage), 0 bit "Exceed Input Voltage Limit" will be selected in the r(Status Error) and 0x01(Low Voltage) will be added to r(Error Codes[0]~[4]).

- Default value is 0x5F(App 7.1V). Refer to to the conversion chart (page 48) to see the relationship to actual voltage.

Maximum Voltage(EEP Register Address #13, RAM Register Address #6)

Refers to maximum input voltage. If the DRC input voltage r(Input Voltage Value) is above r(Maximum Voltage), 0 bit "Exceeded Input Voltage Limit" will be selected int the r(Status Error) and 0x02(High Voltage) will be added to r(Error Codes[0]~[4]).

- Default value is 0x88(App 10.0V). Refer to to the conversion chart (page 48) to see the relationship to actual voltage.

Maximum Temperature(EEP Register Address #14, RAM Register Address #7)

Refers to maximum operating temperature Raw Data. If DRC temperature r(Temperature Value) exceeds r(Maximum Temperature), 1 bit "Exceed Temperature Limit" will be selected in r(Status Error) and 0x03(Hight Temperature) will be added to r(Error Codes[0]~[4]).

- Default value is 0xDF(약 85℃). Refer to to the conversion chart (page 50) to see the relationship to actual temperature.

Remocon Channel(EEP Register Address #15, RAM Register Address #8)

Refers to remote control channel. Remote control has value range from 0x61 to 0x6A with 10 selectable channels. Actual remote control channel must match the r(Remocon Channel) for remote control commands to be recognized.

Servo Ack Wait Tick(EEP Register Address #16, RAM Register Address #9)

Wait to receive Servo Ack after sending cut request to the servo connected to the DRC. No reply received judgment is made if Servo Ack is not received by the DRC within the prescribed time based on the estimated size of the Servo Ack. Servo Ack wait Tick refers to the wait time for the shortest Servo Ack (9 byte) with the wait time increasing as the length of the Servo Ack increases. 1 tick is equal to 1.6ms and the default value is 0x04 (approximately 6.4ms).

Zigbee Ack Wait Tick (EEP Register Address #17, RAM Register Address #10)

Maximum waiting time for receiving reply packet (ACK Packet) from the Zugbee module connected to DRC. It the return packet (ACK Packet) is not received within the maximum waiting time, it is assumed no reply will be received. 1tick = 1.6ms, Default value is 0x50(약 128ms).

LED Blink Period(EEP Register Address #18, RAM Register Address #11)

Alarm LED blink rate when LED blinks according to the r(Alarm LED Policy) when error detected. LED will be on for r(LED Blink Period) and off for r(LED Blink Period) with continous repetition. 1tick = 1.6ms., Default value is 0xBB(Appx 300ms).

ADC Fault Check Period(EEP Register Address #19, RAM Register Address #12)

Input voltage and temerature check period. If input voltage and the temeratrure exceeds maximum limit for longer than r(ADC Fault Check Period), it is assumed that error has occured. 1tick= 1.6ms, Default value is 0x0138(500ms).

Packet Garbage Check Period(EEP Register Address #21, RAM Register Address #14)

Incomplete or garbage packet check period. If incomplete packet is received or if complete packet is not received within r(Packet Garbage Check Period), incomplete packet will be deleted and #2 bit "Invaild Pacekt" will be selectd in r(Status Error) . Depending on where the packet was coming from, 0x41(Zigbee module incomplete reply packet)or 0x51(Servo incomplete reply packet), or 0x61(PC incomplete request packet) will be added to r(Error Codes[0]~[4]).

Status Error(RAM Register Address #16)

Shows the controller error states. Total of 7 bits are used to show different error state values. r(Alarm LED Policy) and r(Torque Off Policy) also have the same error format as below. Alarm LED will start to blink if error state expressed by 1 bit in r(Alarm LED Policy) occurs. Torque will be released on all connected servos if error state expresedd by 1 bit in r(Torque Off Policy) occurs.

Bit	Value	Type
0	0x01	Exceed Input Voltage limit
1	0x02	Exceed Temperature limit
2	0x04	Invalid Packet
3	0x08	Servo Missing
4	0x10	EEP REG distorted
5	0x20	Servo Status Error
6	0x40	Flash Data Distorted
7	0x80	Reserved

Error Codes[0]~[4](RAM Register Address #17)

Shows the detailed error codes when error occurs. Total of 5 bytes are used to save most recent 5 error codes. When error occurs, error code is saved in [0] and previous error codes saved in [0]~[3] are pushed back 1 byte to [1]~[4]. For details, refer to error code list in (page 52).

LED Control(RAM Register Address #22)

Controls the LED whien running Task. Register can have values from 0x00~0x07, LED comes on when each bit is 1 and goes off whe each bit is 0. Table below shows the LED controlled by each bit. LED control has no meaning when Task is not running and the each bit is always 0.

Bit	Value	LED
0	0x01	TX(Red)
1	0x02	RX(Green)
2	0x04	Spare(Blue)

User Timer Tick(RAM Register Address #23)

Timer controlled by the user, if value other than 0 is used, number will decrease by 1 every 100ms. It is used to set the delay time when running Task.

Connected Program(RAM Register Address #24)

Register shows the program currently connected and communicating with the PC.

- 0 : Not connected to the program
- 1 : Connected to HerkuleX Manager
- 2 : Connected to DR-SIM
- 3 : Connected to DR-Visual Logic

Zigbee channel (RAM Registor Address #25)

Holds frequency channel Zgbee module is currently using to communicate with. Selectable channels are from 11~16 with 15 being the default factory value. Register value is 0 if Zigbee module is not connected.

Zigbee PANID(RAM Register Address #26)

Register shows ID of the WPAN (Wireless Personal Area Networt) Zigbee module is currently connected to. Zigbee module will have factory default value of 0xBADA when first connected to DRC. Register value will be 0xFFFF if Zigbee moodule is not connected.

Zigbee SADDR(RAM Register Address #28)

Zigbee module has Short Address of 2 bytes and Long Address of 8 bytes. DRC uses the Short Address for communicating and Short Address is also used to distinguish each individual Zigbee module connected to same WPAN. Zigbee module will have factory default value of 0xBEAD when first connected to DRC. Register value will be 0xFFFF if Zigbee moodule is not connected.

Zigbee DSTADDR(RAM Register Address #30)

Refers to Short Address of the Zigbee module receiving the packet when packet is sent to another module on the same WPAN. Zigbee module will have factory default value of 0xBEAD when first connected to DRC. Register value will be 0xFFFF if Zigbee moodule is not connected.

※ If packet is sent with register value of 0xFFFF, sent pacekt will be broadcasted and every Zigbee module connected to the same WPAN will receive the packet.

Zigbee ACKREQ(RAM Register Address #32)

WIreless commnunication maybe disrupted by another wireless equipment or an obstacle. When sending wireless signal from Zigbee module to another module. requesting ACK packet from the receiving module will increase the reliability by resending the packet if reply packet is not received. However, requesting ACK packet increases the communications time so it is not recommended when packets are being sent at lesss than 100ms intervals. Receive reply packets when r(Zigbee ACKREQ) is 1 and do not receive reply packets when r(Zigbee ACKREQ) is 0. Factory default value saved in Zigbee module is 1. Register will have value of 2 if Zigbee module is not connected.

Zigbee BACKOFF(RAM Register Address #33)

WIreless communication from Zigbee module to another module may not be possible while another equipment or Zigbee module is using the same wireless frequency. Setting r(Zigbee BACKOFF) to 1 will make the module wait for random amount of time before trying to establish commnication again. Similar to r(Zigbee ACKREQ), r(Zigbee BACKOFF) increases communication reliability as well as the communication time. Module will retry communication without waiting if r(Zigbee BACKOFF) is 0. Factory default value saved in Zigbee module is 1. Register will have value of 2 if Zigbee module is not connected.

Servo Count(RAM Register Address #34)

Shows the total number of servo motors with distinct ID connected to the cotroller. Maximum of 32 servo motors can be connected. If number of motors exceed 32, #5 bit "Servo Status Error" will be selected in r(Status Error) and 0x33 (Too Many Serovs Connected) will be added to r(Error Codes[0]~[4]).

Servo ID[0]~[32](RAM Register Address #35)

33 byte space containing ID of the currently connected servo motors. Total of r(Servo Count) byte contains servo motor ID from Servo ID[0] to ID[r(Servo Count)-1]. 0xFE(Broadcasting ID) is saved in the extra space. Even though 32 is the maximum number of servos allowed, 33 bytes are used to satisfy the rule of saving 0xFE in Servo ID[r(Servo Count)] even when r(Servo Count) is 32.

Playing Motion(RAM Register Address #68)

Flag showing whether the motion saved in the DRC is running. 1 = running, 0 = not running.

Playing Task(RAM Register Address #69)

Flag showing wether the task saved in the DRC is running. 1= running, 3= running in debug mode, 0 = not running.

Charger Connected(RAM Register Address #70)

Flag showing whether the battery charge is connected to the DRC by DC jack. 1= connected, 0 = not connected.

Buzzer Scale(RAM Register Address #71)

Shows the pitch of the note currently being played by the buzzer. 3 octaves of buzzer tones can be expressed in semi-tone units. Maintains 0 value when buzzer is not playing. # in front of the pitch denotes octave.

Value	Pitch	Value	Pitch	Value	Pitch	Value	Pitch
0	rest	10	3Ra	20	4Sol	30	5Fa
1	3Do	11	3Ra#	21	4Sol#	31	5Fa#
2	3Re#	12	3Si	22	4Ra	32	5Sol
3	3Re	13	4Do	23	4Ra#	33	5Sol#
4	3Re#	14	4Do#	24	4Si	34	5Ra
5	3Mi	15	4Re	25	5Do	35	5Ra#
6	3Fa	16	4Re#	26	5Do#	36	5Si
7	3Fa#	17	4Mi	27	5Re	37	6Do
8	3Sol	18	4Fa	28	5Re#		
9	3Sol#	19	4Fa#	29	5Mi		

Buzzer Time(RAM Register Address #72)

Shows the remaining play time of the buzzer note being played. 1tick = 6.4ms. There are total of 10 different note lengths that can be used to make buzzer melody or be used to play the note in the task. For example, to run 8 minute note, value of 24 is written in the Buzzer Time and this value will decrease by 1 every 6.4ms untill it becomes 00. Buzzer will sound for 153.6ms.

Button Status(RAM Register Address #73)

Shows the state of 6 buttons. State of each button is expressed by 1 bit, pressed button is 1, released button is 0. For example, when OK button and Left button is pressed simulatneously r(Button Status) is 0x12.

Bit	Value	Button
0	0x01	Mode
1	0x02	OK
2	0x04	Up
3	0x08	Down
4	0x10	Left
5	0x20	Right

Remocon Length(RAM Register Address #74)

Shows the length of time remote control button is being pressed. Once the button signal is received, normal value of 0 increases by 1 every 125ms . For example, r(Remocon Length) value of 3s button press is 24. Maximum r(Remocon Length) value of 240 allows up to 30s button press to be recognized.

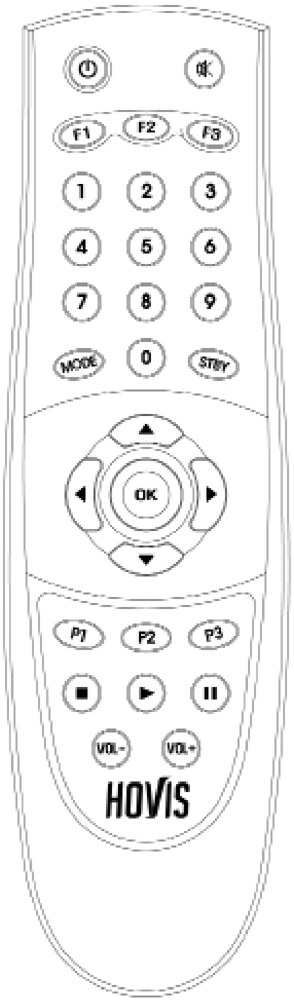
Remocon Data(RAM Register Address #75)

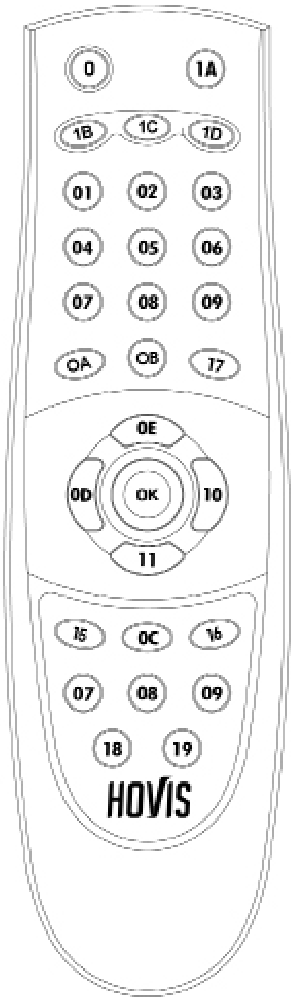
Key value of the pressed remoted control button. Each remote control button has distinct key value assigned. Key value is 254(0xFE) when there is no button signal.

Hovis remote control keymap is as shown in the picture to the left. IR Receive module data values correspond to numbers in the left key.

For example, if the top left power button is pressed, Data 0 is received by the DRC. Robot can be programmed to take certain action when ever the power button is pressed by setting the Data to 0 in the IR RECEIVE module and connecting to Switch module input.

Bothe the Remote control channel and DRC channel is user selectable but selected channel in DRC must match the remote controller channel in order for DRC to receive data from the remote control. Remote control channel can be selected by pressing 1~0 number + OK button simultaneously. DRC channel is selected by changing the RmcChannel value in MPSU Ram Data. RmcChannel values corresponding to remote control numbers are as follows.





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

Input Voltage Value(RAM Register Address #76)

Shows the ADC(Anlog-to-Digital Conversion) value of the input voltage in RAW DATA. Refer to the coversion chart in (page 48) to view the relationship to actual voltage value.

Temperature Value(RAM Register Address #77)

Shows the ADC(Anlog-to-Digital Conversion) value of the current temperature in Raw Data. Refer to the conversion chart in (page 50) to view the relationship to actual temperature.

Light Sensor Value(RAM Register Address #78)

Shows the amount of light coming into the light sensor attached to the DRC.The larger the r(Light Sensor Value) value, brighter the operating environment.

ADC Port 1 Sensor Type(RAM Register Address #79)

Shows the type of sensor attached to the ADC Port 1 .

- 0 : No sensor attached.
- 1 : Analog infrared distance sensor (PSD) attached.
- 2 : Digital distance sensor attached.
- 3 : Shows that DRX-0001M is connected.

ADC Port 2 Sensor Type(RAM Register Address #80)

Shows the type of sensor attached to ADC Port 2.

- 0 : No sensor attached.
- 1 : Analog infrared distance sensor (PSD) attached.
- 2 : Digital distance sensor attached.
- 3 : Shows that DRX-0001M is connected.

ADC Port 1 Sensor Value(RAM Register Address #81)

Shows the value of the sensor attached to ADC Port 1.

- When r(ADC Port 1 Sensor Type) is 0 : 0, no sensor attached.
- When r(ADC Port 1 Sensor Type) is 1 : Detected distance (cm unit) shown as value of 3~40.
- When r(ADC Port 1 Sensor Type) is 2 : Output of the digital distance sensor shown as 0 or 1. 1 if object is at distance of 〉10cm, 0 if 〈 10cm.
- When r(ADC Port 1 Sensor Type) is 3 : 0 as it is not a sensor.

ADC Port 2 Sensor Value(RAM Register Address #83)

Shows the value of the sensor attached to ADC Port 2.

- When r(ADC Port 2 Sensor Type) is 0 : 0, no sensor attached.
- When r(ADC Port 2 Sensor Type) is 1 : Detected distance (cm unit) shown as value of 3~40.
- When r(ADC Port 1 Sensor Type) is 2 : Output of the digital distance sensor shown as 0 or 1. 1 if object is at distance of 〉10cm, 0 if 〈 10cm.
- When r(ADC Port 2 Sensor Type) is 3 : 0 as it is not a sensor.

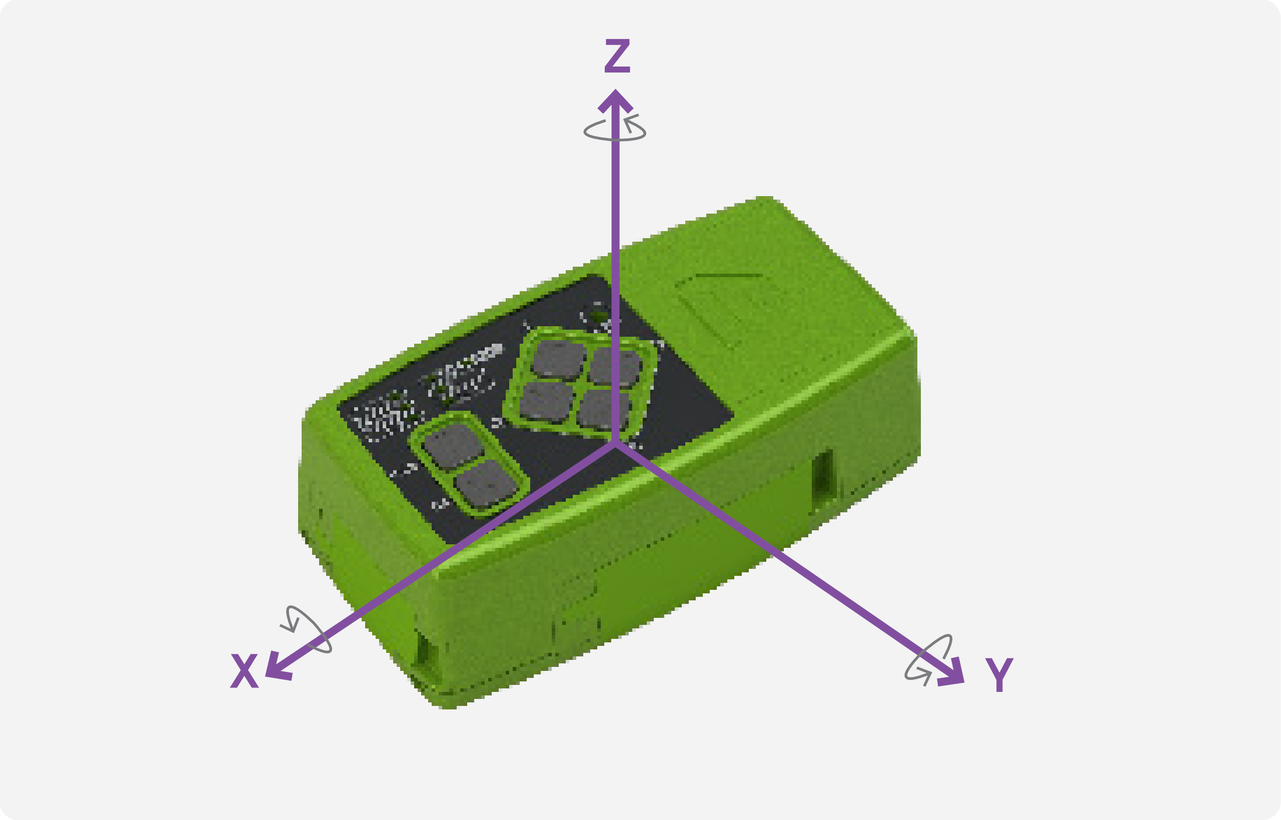
Acc/Gyro Connected(RAM Register Address #85)

Flag shows whether the Acc/Gyro sensor module is attached. 1 if attached. 0 if not attached.

Acc X Value, Acc Y Value, Acc Z Value(RAM Register Address #86,88,90)

Acc sensor X, Y, Z axis value in Raw Data. Acc sensor measures the accleration being applied to the controller. Direction of the acceleration sesor axis are shown below in the diagram. Each axis has value range from -4096~4095. 265 is 1g(Grvitational accleration, 9.8m/s2). Use the following formula to convert Raw Data to g unit.

- Accleration(g) = (Raw Data) / 265



Gyro X Value, Gyro Y Value, Gyro Z Value(RAM Register Address #92,94,96)

Gyro sensor X, Y, Z axis value in Raw Data. Gyro sensor measures the rotational speed of the controller. Point the right thumb towards the direction of the gyro sensor axis and fold the remaining figers into the palm to find the (+) direction of the axis rotation. In other words, (+) direction of the rotation is conuter clockwise direction when looking down the axis. Each axis of the Gyro senson has value range of -32768~32767. About 16.38 is 1˚/s(1˚ rotation per 1s). Use the following formula to convert Raw Data to ˚/s unit.

- Angle speed(˚/s) = (Raw Data) / 32768 X 2000

Sound Detection Flag(RAM Register의 Address #98)

Shows the number of successive sounds detected by DRC. r(Sound Detection Flag) is incremented by 1 for when sound is detected by DRC and then goes back to 0 if no more sound is detected in 1s. If another sound is detected in 1s, r(Sound Detection Flag) is again incremented by 1, and then DRC waits for another sound detection for 1s.

Sound Direction(RAM Register Address #99)

Shows the direction of the most recent sound detected when r(Sound Detection Flag) is greater than or equal to 1.

- -2 : Sound detected from 90˚ to the left.
- -1 : Sound detected from 45˚에 to the left.
- 0 : Sound detected from the middle.
- 1 : Sound detected from 45˚ to the right.
- 2 : Sound detected from 90˚ to the right.

Value is 0 when r(Sound Detection Flag) is 0.

Tick(RAM Register Address #101)

Timer Tick is the basic standard for all DRC operation related to time. Starts from 0 and then goes back to 0 after reaching 60000.

Servo Position[0]~[31](RAM Register Address #170)

2*32 byte space containing the position of the the connected servos. Position value of the servo with r(Servo ID[n]) ID is saved in the r(Servo Position[n]). Position value is updated continuously in real-time if r(Status Check Policy) is 1.Space above r(Servo Count) is filled with 0s.

Touch Status(RAM Register Address #100)

Shows the head touch status when head module DRT-HWW1 is connected. Value is 1 when head touch is detected and 0 otherwise. Value remains 0 when DRT-HWW1 is not connected.

Tick(RAM Register Address #101)

DRT-HWW1 Connected(RAM Register Address #103)

Shows DRT-HWW1 connection status. Value is 1 when DRT-HWW1 is connected and 0 when DET-HWW1 is not connected.

DRC-004TO Connected(RAM Register Address #104)

Shows DRC-004TO connection status. Value is 1 when DRC-004TO is connected and 0 when DET-HWW1 is not connected.

Servo Status Error & Detail[0]~[31](RAM Register Address #106)

2*32 byte storage containing status of currently connected servo motors. r(Status Error) and r(Status Detail) values in servo motor RAM Register of the servo corresponding to r(Servo ID[n]) are saved in r(Servo Status Error & Detail[n]). If r(Status Check Policy) value is 1, r(Servo Status Error & Detail[n]) values are updated continuously to show the current status of connected servos. Space above the number of connected servos r(Servo Count) are filled with 0s and if status cannot be updated due to lack of communication with the connected servo, value of the 1st byte (Status Error) of r(Servo Status Error & Detail[n]) changes to 0x80 to show communication error.

Servo Position[0]~[31](RAM Register Address #170)

2*32 byte storage containing position of currently connected servo motors. r(Calibrated Position) value in servo motor Ram Register of the servo corresponding to r(Servo ID[n]) is saved in r(Servo Position[n]). If r(Status Check Policy) value is 1, r(Servo Position[n]) value is updated continuously to show the current position value of the connected servo. if r(Status Check Policy) value is 1. Space above the number of connected servos r(Servo Count) are filled with 0s.

DRT-HWW1 Status Error(RAM Register Address #234)

Storage containing r(Status Error) value of DRT-HWW1. If r(Status Check Policy) value is 1, value is updated continuously to show error status of the connected DRT-HWW1. If r(DRT-HWW1 Connected) value is 0 (DRT-HWW1 disconnected), register value continues to remain as 0 and if status cannot be updated due to lack of c ommunication, value changes to 0x80 to show communication error.

DRT-HWW1 Status Detail(RAM Register의 Address #235)

Storage containing r(Status Detail) value of DRT-HWW1. If r(Status Check Policy) value is 1, value is updated continuously to show detailed status of the connected DRT-HWW1. If r(DRT-HWW1 Connected) value is 0 (DRT-HWW1 disconnected), register value continues to remain as 0.

DRC-004TO Status Error(RAM Register Address #236)

Storage containing r(Status Error) value of DRC-004TO. If r(Status Check Policy) value is 1, value is updated continuously to show error status of the connected DRC-004TO. If r(DRC-004TO Connected) value is 0 (DRC-HWW1 disconnected), register value continues to remain as 0 and if status cannot be updated due to lack of communication, value changes to 0x80 to show communication error

DRC-004TO Status Detail(RAM Register Address #237)

Storage containing r(Status Detail) value of DRC-004TO. If r(Status Check Policy) value is 1, value is updated continuously to show detailed status of the connected DRC-004TO. If r(DRC-HWW1 Connected) value is 0 (DRC-004TO disconnected), register value continues to remain as 0.

Protocol

Protocol Format

Overview

Packet controlling the DRC is divided into 'Request packet' used when communicating from PC to DRC and reply packet 'Ack Packet' from DRC to PC.

Setup

Commuications settings are as follows.

Baud Rate : 57,600 / 115,200 / 0.2M / 0.25M / 0.4M / 0.5M / 0.667M

Data Bit : 8

Stop Bit : 1

Parity : None

Flow Control : None

※ Communication speed of the Com port attached to the PC or USB to Serial Cable maybe limited by the hardware or the driver. Check the Baud Rate if there is problem in communication. Default DRC factory value is 115,200bps.

Packet Structure

Item	Header		Packet Size	pID	CMD	Check Sum1	Check Sum2	Optional Data
value	0xFF	0xFF	7~223	0~0xFE	Refer to details	Refer to details	Refer to details	Refer to details
bytes	1	1	1	1	1	1	1	MAX 216

1. Header(2 Byte)

Beginning of the packet. Composed of 2 bytes 0xFF & 0xFF.

2. Packet Size(1Byte)

Total byte size of the packet from Header to the Optional Data. Maximum Packet Size is 223. Packet Size exceeding 223 bytes will cause error.

3. pID(1Byte)

ID of the DRC to be controlled. Care is required When pID is 254(0xFE), as all DRC rceiving the packet becomes control target. pID larger than 254 will cause error.

※ To distinguis from register r(ID), ID within the packet will be shows as pID.

4. CMD(1Byte)

In the request packet, CMD refers to command to be peformed by DRC. In the reply packet, CMD refers to the command received by the DRC.There are total of 14 commands in the request packet and 13 in the reply packet. To distinguish the reply packet CMD from the CMD in the request packet, 0x40 Bitwise OR operation is performed on the request packet CMD. For example, 0x51 is the reply packet CMD to the request packet EEP_WRITE(0x11) CMD. Refer to the Command Set in page 20 for complete CMD list and page 22 to view detailed description of each CMD.

There are also 9 types of request packets that can be relayed to the servos connected to the DRC. DRC will check the request packets before relaying them to the servo motors and once reply is received from the servos, it will be relayed to the PC. Refer to the servo manual for more information on servo request and reply packets.

5. Check Sum1, Check Sum2(2 Byte)

Check Sum1, 2 is a 2 byte space used to check integrity of the transmitted data. When there is n byte of Optional Data, Check Sum is calculated as follows.

Check Sum1 = (Packet Size ^ pID ^ CMD ^ Data[0] ^ … ^ Data[n–1]) & 0xFE

Check Sum2 = (~(Packet Size ^ pID ^ CMD ^ Data[0] ^ … ^ Data[n–1])) & 0xFE

※ ~ is a Bitwise NOT operator, when ~A is performed, all bits in A are negated.. Example) ~(01101101)becomes 10010010.

※ ^ is a Bitwise AND operator, when A ^ B is performed, each bit of A and B are compared and only the same bits become 1.
Exampe) 00101110 ^ 10110110 becomes 01100111.

6. Optional Data(0~216Byte)

Optional data that changes according to the CMD type. Refer to the detailed command description in page 22 for more information on Optional Data.

Command Set

List of commands that go in the CMD section of the protocol. There are 14 types of CMDs in the (Request Packet) and 13 types of CMDs in the reply packet (ACK Packet). When Request Packet is sent from the PC to DRC, DRC will perform the task requested in the received packet and send the result or status back to the PC in the form of ACK Packet. Refer to the pag 22 to view more detailed information on Request Packet & ACK packet forms and formats.

1. Request Packet(PC to DRC)

Name	Cmd	Remark
EEP_WRITE	0x11	Change Length number of values in EEP Register Address
EEP_READ	0x12	Request Length number of values fromEEP Register Address
RAM_WRITE	0x13	Change Length number of values from RAM Register Address
RAM_READ	0x14	Request Length number of values from RAM Register Address Length
CON_CHECK	0x15	Scan to check the the ID of servos connected to the controller
PLAY_MOTION	0x16	Run saved Motion
PLAY_TASK	0x17	Run saved Task
PLAY_BUZZ	0x18	Run saved head LED & Buzzer
STAT	0x19	Request controller error status and most recent error code
ROLLBACK	0x1A	Rest all variables to factory default value Rest values will be applied after power is turned off and back on.
REBOOT	0x1B	Request reboot
ZIGBEE	0x1C	Send control command related to Zigbee connected to the controller
REMOCON	0x1D	Send Remote Control Data
SERVO_FW_UPDATE	0x1E	Enter Servo F/W update mode

2. ACK Packet (DRC to PC)

Name	Cmd	Remark
EEP_WRITE	0x51	Retun r(Status Error) & r(Status Error Codes[0]) Reply when r(Ack Policy) is All
EEP_READ	0x52	Return Len number of values from EEP Register Address r(Ack Policy) is Read Only, Reply when All
RAM_WRITE	0x53	Rturn r(Status Error) & r(Status Error Codes[0]) Reply when r(Ack Policy) is All
RAM_READ	0x54	Return Len number of values from RAM Register Address r(Ack Policy) is Read Only, Reply when All
CON_CHECK	0x55	Return servo IDs found by scan r(Ack Policy) is Read Only, Reply when All
PLAY_MOTION	0x56	Return r(Status Error) & r(Status Error Codes[0]) Reply when r(Ack Policy) is All
PLAY_TASK	0x57	Reply and reply format depends on Instruction (Refer to 34page)
PLAY_BUZZ	0x58	Return r(Status Error) & r(Status Error Codes[0]) Reply when r(Ack Policy) is All
STAT	0x59	Return r(Status Error) & r(Status Error Codes[0]) Always reply regardless of r(Ack Policy)
ROLLBACK	0x5A	Return r(Status Error) & r(Status Error Codes[0]) Reply when r(Ack Policy) is All
REBOOT	0x5B	Return r(Status Error)와 r(Status Error Codes[0]) Reply when r(Ack Policy) is All
ZIGBEE	0x5C	Reply and reply format depends on Instruction (Refer to 43page)
REMOCON	–	No reply packet.
SERVO_FW_ UPDATE	0x5E	Return r(Status Error) & r(Status Error Codes[0]) Reply when r(Ack Policy) is All

Detailed Command Description – EEP_WRITE

1-1. EEP_WRITE – Request Packet(0x11)

Item	Packet Size	pID	CMD	Data[0]	Data[1]	Data[2]	…	Data[Length+1]
Value	7+2+Length	0~0xFE	0x11	Address	Length	EEP Data[0]	…	EEP Data[Length-1]

Change Length number or values from EEP Register Address. Optional Data contains Address, Length, and Length number of data. Optional Datal length is (2+Length) byte. Total Packet size is standard 7byte + (2+Length)byte = (9+Length) byte. When DRC receives this particular packet, Values in Non-Volatile register address from Address to (Address+Length-1) are changed from EEP Data[0] to EEP Data[Length-1].

※ Any changes made to the Non-Volatile memory does not have direct affect on the operation of the DRC. Values changed by the EEP_WRITE will be copied to the Volatile register when the DRC is rebooted by the REBOOT CMD or when the power is turned off and back on.

Example

- Request Packet to change the e(Alarm LED Policy) of the DRC with r(ID)253 to 0x3

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]
Value	0xFF	0xFF	0x0A(10)	0xFD	0x11	0xD2	0x2C	0x0A	0x01	0x3F

※ CS1, CS2 is abbreviation of Check Sum1 & Check Sum2.
e(Alarm LED Policy) address is 10 and the data length is 1. EEP Data[0] is 0x3F. Packet Size is (9+Length)=10. Check Sum1 & Check Sum2 are calculated according to the formaula in page 19.

1-2. EEP_WRITE – Ack Packet(0x51)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x51	r(Status Error)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contins the r(ID) of the replying DRC, CMD becomes 0x51 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x11.

Reply Condition

EEP_WRITE reply is sent only when r(ACK Policy) is 2(Reply to all packets). Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent even if r(ACK Policy) is 2.

Example

- Reply Packet after receiving request packet to change the e(Alarm LED Policy) of the DRC with r(ID)253 to 0x3

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x51	0xA4	0x5A	0x00	0x00

Send current status and most recent error code.Both are 0x00 as there is no error.

2-1. EEP_READ – Request Packet(0x12)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	0~0xFE	0x12	Address	Length

Read Length number of values from EEP Register Address . Optional Data contains Address, Length, and Length number of data. Optional Datal length is (2+Length) byte. Total Packet size is standard 7bytes + (2+Length)byte = (9+Length) byte. When DRC receives this packet, values from Non-Volatile register address from Address to (Address+Length-1) are sent by the reply packet.

Example

- Request packet to read e(Min Voltage), e(Max Voltage), e(Max Temperature) values from DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x12	0xE8	0x16	0x0C	0x03

e(Min Voltage) address is 12, length is 3. Packet Size is 9.

Check Sum1 & Check Sum2 are calculated according to the formaula in page 19.

2-2. EEP_READ – ACK Packet(0x52)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2+Length+2	r(ID)	0x52	Address	Length

Item	Data[2]	…	Data[Length+1]	Data[Length+2]	Data[Length+3]
Value	EEP Data[0]	…	EEP Data[Length-1]	r(Status Error)	r(Error Codes[0])

Values in the Non-Volatile register address from Address to (Address+Length-1) are sent contained in EEP Data[0]to EEP Data[Length-1]. r(Status Error) & r(Error Codes[0]) values are sent as well. Address, Length, Length number of values, and r(Status Error) & r(Error Codes[0]) are contained in the Optional Data. Optional Data length is (2+Length+2) bytes. Total packet size is standard 7 bytes + (4+Length) = (11+Length) bytes. pID contins the r(ID) of the replying DRC. CMD becomes 0x52 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x12.

Reply Condition

EEP_READ reply is sent when r(ACK Policy) is 1(Reply to only Read command), 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to Request packet to read e(Min Voltage), e(Max Voltage), e(Max Temperature) values from DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x0E(14)	0xFD	0x52	0xA6	0x58	0x0C	0x03

Item	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]
Value	0x5F	0x88	0xDF	0x00	0x00

Send 3 bytes of data from Address 12 contained in Data[2]~Data[4]. e(Min Voltage) in Data[2], e(Max Voltage)in Data[3] , e(Max Temperature) in Data[4]. Send current status and the most recent error code contained in Data[5] and Data[6]. When there is no error, both Data[5] and [6] contain 0x00.

Detailed Command Description – RAM_WRITE

3-1. RAM_WRITE – Request Packet(0x13)

Format

item	Packet Size	pID	CMD	Data[0]	Data[1]	Data[2]	…	Data[Length+1]
Value	7+2+Length	0~0xFE	0x13	Address	Length	RAM Data[0]	…	RAM Data[Length-1]

Change Length number or values from RAM Register Address. Optional Data contains Address, Length, and Length number of data. Optional Datal length is (2+Length) byte. Total Packet size is standard 7byte + (2+Length)byte = (9+Length) byte. When DRC receives this particular packet, Values in Volatile register address from Address to (Address+Length-1) are changed from RAM Data[0] to RAM Data[Length-1].

Example

- Request Packet to change the r(Status Error) & r(Error Codes[0]~[4]) of the DRC with r(ID)253 to 0x00

item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x0F(15)	0xFD	0x13	0xF6	0x08	0x10	0x06

item	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
Value	0x00	0x00	0x00	0x00	0x00	0x00

r(Status Error) address is 16, As 6 bytes of data after the address has to be changed. Address is 16, Length is 6, and RAM Data[0]~RAM Data[5] is 0x00. Total Packet Size is (9+Length)= 15. Check Sum1 & Check Sum2 are calculated according to the formaula in page 19.

※ Both r(Status Error) & r(Error Codes[0]~[4]) are R/W registers but because registers contain current MPSU status, values cannot be changed arbitrarily .The choice of values for these 6 byte registers are to use current values or to change them all to 0x00. If any other values are used "Invalid Packet"(#2 bit) will be selected in r(Status Error) and 0x73(r(Status Error) and " Invalid write command" will be added to r(Error Codes[0]~[4]).

3-2. RAM_WRITE - ACK Packet(0x53)

Format

item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x14	r(Status Error)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contins the r(ID) of the replying DRC. CMD becomes 0x53 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x13.

Reply Condition

RAM_WRITE reply is sent only when r(ACK Policy) is 2(Reply to all packets). Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent even if r(ACK Policy) is 2.

Example

- Reply to Request Packet to change the r(Status Error) & r(Error Codes[0]~[4]) of the DRC with r(ID)253 to 0x00

item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x53	0xA6	0x58	0x00	0x00

Send current status and most recent error code.Both are 0x00 as there is no error.

Detailed Command Description – RAM_READ

4–1. RAM_READ – Request Packet(0x14)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	0~0xFE	0x14	Address	Length

Read Length number of values from RAM Register Address . Optional Data contains Address, Length, and Length number of data. Optional Datal length is (2+Length) byte. Total Packet size is standard 7bytes + (2+Length)byte = (9+Length) byte. When DRC receives this packet, values from Volatile register address from Address to (Address+Length–1) are sent by the reply packet.

Example

- Request packet to read e(Min Voltage), e(Max Voltage), e(Max Temperature) values from DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x14	0xE6	0x18	0x05	0x03

R(Min Voltage) address is 5, length is 3. Packet Size is 9.

Check Sum1 & Check Sum2 are calculated according to the formaula in page 00.

4–2. RAM_READ – ACK Packet(0x54)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2+Length+2	r(ID)	0x54	Address	Length

Item	Data[2]	…	Data[Length+1]	Data[Length+2]	Data[Length+3]
Value	RAM Data[0]	…	RAM Data[Length–1]	r(Status Error)	r(Error Codes[0])

Values in the Volatile register address from Address to (Address+Length–1) are sent contained in RAM Data[0]to RAM Data[Length–1]. r(Status Error) & r(Error Codes[0]) values are sent as well. Address, Length, Length number of values, and r(Status Error) & r(Error Codes[0]) are contained in the Optional Data. Optional Data length is (2+Length+2) bytes. Total packet size is standard 7 bytes + (4+Length) = (11+Length) bytes. pID contins the r(ID) of the replying DRC. CMD becomes 0x54 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x14.

RAM_READ reply is sent when r(ACK Policy) is 1(Reply to only Read command), 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to Request packet to read r(Min Voltage), r(Max Voltage), r(Max Temperature) values from DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x54	0xA8	0x56	0x05	0x03

Item	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]
Value	0x5F	0x88	0xDF	0x00	0x00

Send 3 bytes of data from Address 5 contained in Data[2]~Data[4]. r(Min Voltage) in Data[2], r(Max Voltage)in Data[3] , r(Max Temperature) in Data[4]. Send current status and the most recent error code contained in Data[5] and Data[6]. When there is no error, both Data[5] and [6] cotnain 0x00.

Detailed Command Description – CON_CHECK

5-1. CON_CHECK – Request Packet(0x15)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]	…	Data[Length]
Value	7+1+Length	0~0xFE	0x15	Length	ID[0]	…	ID[Length-1]

Checks to see if servos with ID of ID[0]~ID[Length-1] are connected to the DRC. Optional Data contains Length, Length number of ID. Optional Data length is (1+Length) bytes. Total Packet size is standard 7bytes + (1+Length)byte = (8+Length) byte. When DRC receives this packet, It initiates communication with the serovs with ID[0] to ID[Length-1]. Total number of sucessfully contacted servos and their IDs are sent back by the ACK packet.

※ When Length is 0, all IDs from 0~253 are scanned.

Example

- Request packet to check if servos with ID 0, 1, 2, 3, 4 are connected to DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x0D(13)	0xFD	0x15	0xE4	0x1A	0x05

Item	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
Value	0x00	0x01	0x02	0x03	0x04

There are 5 servos, therefor Length is 5. ID of the servos 0, 1, 2, 3, 4 are in Data[1] to Data[5]..Packet Size (8+Length) = 13. Check Sum1 & Check Sum2 are calculated according to the formaula in page 19.

5-2. CON_CHECK – ACK Packet(0x55)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]	…	Data[Length]	Data[Length+1]	Data[Length+2]
Value	7+1+Length+2	r(ID)	0x55	Length	ID[0]	…	ID[Length-1]	r(Status Error)	r(Error Codes[0])

After performing ID scan, number of sucessfully scanned servos are entered in Length, ID values entered in ID[0] to ID[Length-1] and sent back using reply packet together with r(Status Error) & r(Error Codes[0]) values. Address, Length, Length number of values, and r(Status Error) & r(Error Codes[0]) are contained in the Optional Data. Optional Data length is (1+Length+2) bytes. Total packet size is standard 7 bytes + (3+Length) = (10+Length) bytes. pID contins the r(ID) of the replying DRC, CMD becomes 0x55 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x15.

Reply Condition

CON_CHECK reply is sent when r(ACK Policy) is 1(Reply to only Read command), 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Request packet to check if servos with ID 0, 1, 2, 3, 4 are connected to DRC with r(ID) 253. Reply packet whenscan result shows only ID 0,1,2 are connected.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]
Value	0xFF	0xFF	0x0D(13)	0xFD	0x55	0xA4	0x5A	0x03	0x00	0x01	0x02	0x00	0x00

ID scan result shows only ID 0, 1, 2 are connected. Data[0] showing Length is 3 and IDs are entered sequentially in Data[1]~Data[3]. Send current status and the most recent error code contained in Data[4] and Data[5]. When there is no error, both Data[4] and [5] cotnain 0x00. Packet Size (10+Length)=13

Detailed Command Description – PLAY_MOTION

6-1. PLAY_MOTION – Request Packet(0x16)

Fromat

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	0~0xFE	0x16	Motion No.	Motion Ready Flag

When DR-SIM is used to save motion in DRC, saved motion receives a number between 0 to 127. PLAY_MOTION packet runs the saved motion in DRC, Motion No. refers to the saved motion number. Motion Ready Flag decides whether to take motion ready posture. When packet is sent with Motion Ready Flag set to 1, first frame of the motion will be played slowly. Damage to the motor or fall due to sudden movent can be prevented by sening a packet with Motion Ready Flag set to 1 and then another packet with Flag set to 0 little later. Also, current motion will stop if packet is sent with motion No. 254(0xFE).

See below for arrangement of the motion with Motion No. & Motion Ready Flag

Motion No	Motion Ready Flag	Motion
0~127	0	Run Motion
0~127	1	Run first frame of the motion
254	0~1	Stop Motion

Example

- Request packet to run Motion No 1 in DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x16	0xE2	0x1C	0x01	0x00	0x00

As motion being run is No.1, Motion No. is set to 1, Motion Ready Flag set to 0.

- Request packet to run first frame of Motion No 2 in DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x16	0xE0	0x1E	0x02	0x01

As motion being run is No.2, Motion No. is set to 2, Motion Ready Flag set to 1.

6-2. PLAY_MOTION – ACK Packet(0x56)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x56	r(Status Error)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contins the r(ID) of the replying DRC. CMD becomes 0x56 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x16.

Reply Condition

PLAY_MOTION reply is sent when r(ACK Policy) is 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to request packet to run Motion No 2 in DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x56	0xA2	0x5C	0x00	0x00

Send current status and most recent error code.Both are 0x00 as there is no error.

Detailed Command Description – PLAY_TASK

7–1. PLAY_TASK – Request Packet(0x17)

Format

Item	Packet Size	pID	CMD	Data[0]
Value	7+1	0～0xFE	0x17	Instruction

Use DR–Visual Logic to run the Task saved in DRC. Depending oh the instruction, PLAY_TASK is divided into 4 commands which perfom different function according to the Instruction.

- When Instruction is 0, runs the Task in normal mode.
- When Instruction is 1, runs Task in debugging mode.
- When Instruction is 2, rusn the fisrt stop of the Task and stops. This Instruction has meaning only when in debugging mode.
- When Instruction is 254, stops Task. Task stops regardless of whether it's in normal or debugging mode.

Example

- Request packet to run Task saved in DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x08(8)	0xFD	0x17	0xE2	0x1C	0x00

Task run, Instruction is 0.

- Request packet to run Task saved in DRC with r(ID) 253 in debugging mode

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x08(8)	0xFD	0x17	0xE2	0x1C	0x01

Instruction is 1 as Task is running in debugging mode.

- Packet running one step of the Task when DRC with r(ID) 253 is in debugging mode.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x08(8)	0xFD	0x17	0xE0	0x1E	0x02

Instruction is 2 as Task runs for single step in debugging mode.

7-2. PLAY_TASK - ACK Packet(0x57)

Format - Debuggin ACK Packet

Item	Packet Size	pID	CMD	Data[0]	Data[1]	Data[2]	Data[3]
Value	7+4	r(ID)	0x57	Program Counter L	Program Counter H	r(Status Error)	r(Error Codes[0])

Format - Status ACK Packet

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x57	r(Status Error)	r(Error Codes[0])

Depending on the Instruction, PLAY_TASK replay packet is divided into two types.

Debugging reply packet shows which section of the task is running in 2 bytes by using Progam Counter L and Program counter H. This information is used to find out which code is currently running when debugging Task in DR-Visual Logic. Debuggin reply packet also includes r(Status Error) & r(Error Codes[0]) values. As Optional Data length is fixed at 4 bytes, total packet size is 11 bytes. pID contins the r(ID) of the replying DRC, CMD becomes 0x57 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x17.

Status reply packet includes r(Status Error) & r(Error Codes[0]) values. As Optional Data length is fixed at 2 bytes, total packet size is fixed at 9 bytes. Debugging reply packet is used in circumstances related to debugging and status reply packet in other circumstances. Refer to below to view the type of reply packet being sent depnding on the Instruction & circumstances.

Instruction	Sucess	Fail
0	Status	Status
1	Debugging	Status
2	Debugging	Status
254	Status	Status

Instruction 0 (Task Running) and Instruction 254 (Task stop) are replied with status reply packet. Debuggin related instructions such as Instruction 1 (Run Task in debuggin mode) and Instruction 2 (Run one step) are replied with debugging reply packet. However, under the circustances when requested command cannot be performed as when Instruction 1 is sent while the Task is running or Instruction 2 is sent when Task is not running, reply will be with status reply packet.

Reply Condition

Status reply packet is sent when r(ACK Policy) is 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Debuggin reply packet is sent when r(ACK Policy) is 1(Reply to only Read command), 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to request packet to run Task saved in DRC with r(ID) 253

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x57	0xA2	0x5C	0x00	0x00

As request packet Instruction is 0, reply with current status and most recent error code. There is no error. both Data [0] & [1] wil have 0x00 values.

- Reply to request packet to run Task saved in DRC with r(ID) 253 in debugging mode.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]	Data[3]
Value	0xFF	0xFF	0x0B(11)	0xFD	0x57	0xAA	0x54	0x0B	0x00	0x00	0x00

As request packet instruction is 1, reply with degugging reply packet. Current program counter location is saved in Data[0] & Data[1]. Current location after starting debugging process is 0x000B. Current status and recent error code is saved in Data[2] & Data[3].

- Request to run one step of Task when DRC with r(ID) 253 is in debuggin mode.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]	Data[3]
Value	0xFF	0xFF	0x0B(11)	0xFD	0x57	0x86	0x78	0x26	0x00	0x00	0x00

As request packet instruction is 2, reply with degugging reply packet. Current program counter location saved in Data[0] & Data[1] is showing 0x0026. Current status and recent error code is saved in Data[2] & Data[3].

Detailed Command Description – PLAY_BUZZ

8–1. PLAY_BUZZ – Request Packet(0x18)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	0~0xFE	0x18	Reserved	Buzz No.

Run Buzzer saved in DRC. Buzzer can have number between 1 to 63. Send request packet with Buzzer number in Data[1] Buzz No. Enter 0 in Data[0] as this space is Reserved for other data.

Example

- Request packet to run Buzzer No. 5 in DRC with r(ID) 253.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x18	0xE8	0x16	0x00	0x05

Running Buzzer No. 5. Data[1] is 5.

8–2. PLAY_BUZZ – ACK Packet(0x58)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x58	r(Status Error)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contins the r(ID) of the replying DRC. CMD becomes 0x58 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x18.

Reply Condition

PLAY_ Buzz reply is sent when r(ACK Policy) is 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to request packet to run Buzzer No. 5 in DRC with r(ID) 253.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x58	0xAC	0x52	0x00	0x00

Send current status and most recent error code.Both are 0x00 as there is no error.

Detailed Command Description – STAT

9–1. STAT – Request Packet(0x19)

Format

Item	Packet Size	pID	CMD
Value	7+2	0~0xFE	0x18

Request current status of DRC. DRC sends reply packet with r(Status Error) & r(Error Codes[0]) values included.

Example

- Request packet to DRC with r(ID) 253 to perform STAT command.

Item	Header		Packet Size	pID	CMD	CS1	CS2
Value	0xFF	0xFF	0x07(7)	0xFD	0x19	0xE2	0x1C

9–2. STAT – ACK Packet(0x59)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x59	r(Status Er–ror)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contins the r(ID) of the replying DRC, CMD becomes 0x59 by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x19.

Reply Condition

Reply is sent to STAT request regardless of r(ACK Policy). Reply is sent even if the pID of request packet is 254(Broadcasting ID).

Exampe

- Reply to request packet to DRC with r(ID) 253 to perform STAT command.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x59	0xAC	0x52	0x00	0x00

Send current status and most recent error code.Both are 0x00 as there is no error.

Detailed Command Description – ROLLBACK

10–1. ROLLBACK – Request Packet(0x1A)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	0~0xFE	0x1A	ID Skip	Baud Skip

Initialize Non–Volatile register using the factory default values saved in DRC. Initialized Non–Volatile will affect the operation after DRC has been rebooted or power turned off and back on. ID Skip and Baud Skip in Data[0] & Data[1] determines whether e(ID) & e(Baud Rate) will be exempt from initialization.When ID Skip is 1, e(ID) will not be initialized and when Baud Skip is 1, e(Baud Rate) will not be initialized.

Example

- Request packet to DRC r(ID) 253 to initialize Non–Volatile registers except for e(ID).

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x1A	0xEE	0x10	0x01	0x00

Request packet will initialize the register with an exception of e(ID). ID Skip is 1, Baud Skip is 0.

- Request packet to DRC r(ID) 253 to initialize register with exceoption on e(ID) & e(Baud Rate).

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x1A	0xEE	0x10	0x01	0x01

Request packet will initialize the register with exception of e(ID) & e(Baud Rate). ID Skip is 1, Baud Skip is 1.

10-2. ROLLBACK – ACK Packet(0x5A)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x5A	r(Status Error)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contins the r(ID) of the replying DRC. CMD becomes 0x5A by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x1A.

Reply Condition

ROLLBACK reply is sent when r(ACK Policy) is 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to request packet to DRC r(ID) 253 to initialize Non-Volatile registers except for e(ID).

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x5A	0xAE	0x50	0x00	0x00

Send current status and most recent error code.Both are 0x00 as there is no error.

Detailed Command Description – REBOOT

11-1. REBOOT – Request Packet(0x1B)

Format

Item	Packet Size	pID	CMD
Value	7	0~0xFE	0x1B

Request packet to DRC requesting SW reset. When DRC receives this packet, it will reset itself and start initial booting sequence.

Example

Item	Header		Packet Size	pID	CMD	CS1	CS2
Value	0xFF	0xFF	0x07(7)	0xFD	0x1B	0xE0	0x1E

11-2. REBOOT – ACK Packet(0x5B)

Item	Packet Size	pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x5B	r(Status Error)	r(Error Codes[0])

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contains the r(ID) of the replying DRC. CMD becomes 0x5B by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x1B.

Reply Condition

REBOOT reply is sent when r(ACK Policy) is 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x5B	0xAE	0x50	0x00	0x00

Send current status and most recent error code. Both are 0x00 as there is no error.

Detailed Command Description – ZIGBEE

12–1. ZIGBEE – Request Packet(0x1C)

Format

Item	Packet Size	pID	CMD	Data[0]
Value	7+1	0~0xFE	0x1C	Instruction

Request packet with commands related to conrolling the Zigbee module attached to DRC. Depending oh the instruction, ZIGBEE is divided into 6 commands which perfom different function according to the Instruction.

There are 5 types (total 8 bytes) of Zigbee related registers in the Volatile register map, r(Zigbee PANID), r(Zigbee SADDR), r(Zigbee DSTADDR), r(Zigbee ACKREQ), r(Zigbee BACKOFF). Each register corresponds to the property values saved in the Zigbee module. Communication using Zigbee cannot be wired and wireless at the same time.

- When Instruction is 0, Zigbee module property values are read to the Volatile register.
- When Instruction is 1, Property values in Volatile register are replaced with property values in Zigbee module .
- When Instruction is 2, Proerty values in Zigbee module are intialized to factory default values.
- When Instruction is 3, Zigbee module is reset.
- When Instruction is 4, Change to wired communication mode (Using connection cable and COM PORT).
- Instruction is 5, Change to wireless communication mode (Wireless communication using Zigbee).

Example

- Request packet to read Zigbee property values from DRC with r(ID) 253.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x08(8)	0xFD	0x1C	0xE8	0x16	0x00

Instruction is 0; reading property values from the module to the RAM.

- Request packet to change the Zigbee module values to factory value from DRC with r(ID) 253.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x08(8)	0xFD	0x1C	0xEA	0x14	0x02

Instruction 2; Initialize Zigbee module property values to factory default.

- Request packet to change the DRC with r(ID) 253 to wireless communication mode.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]
Value	0xFF	0xFF	0x08(8)	0xFD	0x1C	0xEC	0x12	0x05

Instruction 5; Change communication mode to wireless

12-2. ZIGBEE – ACK Packet(0x5C)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]	Data[2]
Value	7+3	r(ID)	0x5C	Success	r(Status Error)	r(Error Codes[0])

ZIGBEE reply packet carries value of 'Success' field in Data[0]. 'Success' field in reply packet shows whether the command sent by the request packet was successfully carried out. Success value is 1 when the Zigbee coomand was successful, value is 0 if the command failed due to communication error or because Zigbee module was not installed. r(Status Error) & r(Error Codes[0]) values are included in the Optional Data. As Optional Data size is fixed at 3 bytes, total Packet Size is 10 bytes. pID conatins the r(ID) of the replying DRC, CMD becomes 0x5C by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x1C.

Reply Condition

CON_CHECK reply is sent when r(ACK Policy) is 1(Reply to only Read command), 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to request packet to read Zigbee prperty values from DRC with r(ID) 253 (Zibee installed).

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]
Value	0xFF	0xFF	0x0A(10)	0xFD	0x5C	0xAA	0x54	0x01	0x00	0x00

Success value is 1 since Zigbee was installed and communication was successful.

Example

- Reply to request packet to initialize Zigbee to factory defalut values from DRC with r(ID) 253 (Zigbee installed).

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]
Value	0xFF	0xFF	0x0A(10)	0xFD	0x5C	0xAA	0x54	0x01	0x00	0x00

Zigbee initialized to factory default values. Success value is 1.

- Reply to request packe to change DRC with r(ID) 253 to wireless mode (Zigbee not installed).

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]
Value	0xFF	0xFF	0x0A(10)	0xFD	0x5C	0xAA	0x54	0x00	0x00	0x00

Mode change failed since Zigbee is not installed. Success value is 0.

Detailed Command Description – REMOCON

13–1. REMOCON – Request Packet(0x1D)

Format

Item	Packet Size	pID	CMD	Data[0]	Data[1]	Data[2]
Value	7+3	0~0xFE	0x1D	Channel	Length	Data

IR remote control can be used to send control commands when IR receiver is attached to DRC. However, when IR remote control is not available or when in wireless communication mode using Zigbee, request packet with REMOCON command can be used control the DRC. Remote control Channel(0x61~0x6A) goes in Data[0], remote control button press Length (0~240, 1= 125ms) in Data [1], and remote control button key data in Data[2]. When DRC receives remote control value, Channel is compared with r(Remocon Channel). If they are found to match, r(Remocon Length) & r(Remocon Data) values are changed to Length & Data for 250ms. r(Remocon Length) & r(Remocon Data) values are changed back to 0 & 254 after 250ms. When using REMOCON request packet, it is recommended to increase the Length value by 1 every 125ms.

Example

- Request packet notifying all DRC(Broadcasting) button 0x21 using channel 0x61has been presse for 1s

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]	Data[2]
Value	0xFF	0xFF	0x0A(10)	0xFE	0x1D	0xA0	0x5E	0x61	0x08	0x21

pID is 0xFE since packet is being sent to all DRCs. Since channel is 0x61, Data[0] value is 0x61.Since 1unit=125ms, 1s = 8 units. Data[1] has value of 8 and Data [2] has remote control key value of 0x21.

13–2. REMOCON – ACK Packet(0x1D)

REMOCON command does not have reply packet.

Detailed Command Description – SERVO_FW_UPDATE

14-1. SERVO_FW_UPDATE – Request Packet(0x1E)

Format

Item	Packet Size	pID	CMD
Value	7	0~0xFE	0x1Z

Request packed used to update the servo (Firmware) connected to DRC. Since servo firmware update rquires special protocol, SERVO_FW_UPDATE request packet has to be sent to enter special update mode. While in special update mode, there is no communication between the PC and the DRC and unit behaves as if PC and the servos are connected directly.

Example

- Request packet to change the DRS with r(ID) 253 to servo firmware update mode.

Item	Header		Packet Size	pID	CMD	CS1	CS2
Value	0xFF	0xFF	0x09(9)	0xFD	0x1E	0xE4	0x1A

14-2. SERVO_FW_UPDATE – ACK Packet(0x5E)

Format

Item	Packet Size		pID	CMD	Data[0]	Data[1]
Value	7+2	r(ID)	0x5E	r(Status Error)	r(Error Codes[0])	0xA0

Send reply packet with r(Status Error) & r(Error Codes[0]) values included. With Optional Data length fixed at 2 bytes, total Packet size is fixed at 9 bytes. pID contains the r(ID) of the replying DRC, CMD becomes 0x5E by applying 0x40 Bitwise OR operation to the Request Packet CMD 0x1E.

Reply Condition

SERVO_FW_UPDATE reply is sent when r(ACK Policy) is 2(Reply to all packets).Exception to this rule is when pID of the request packet is 254(Broadcasting ID), in which case reply is not sent.

Example

- Reply to request packet to change the DRS with r(ID) 253 to servo firmware update mode.

Item	Header		Packet Size	pID	CMD	CS1	CS2	Data[0]	Data[1]
Value	0xFF	0xFF	0x09(9)	0xFD	0x5E	0xAA	0x54	0x00	0x00

Send current status and most recent error code. Both are 0x00 as there is no error.

Appendix

ADC Lookup Table – Voltage

ADC		VIN	ADC		VIN	ADC		VIN	ADC		VIN
Decimal	HEX		Decimal	HEX		Decimal	HEX		Decimal	HEX	
0	0	0.000	64	40	4.722	128	80	9.444	192	C0	14.167
1	1	0.074	65	41	4.796	129	81	9.518	193	C1	14.240
2	2	0.148	66	42	4.870	130	82	9.592	194	C2	14.314
3	3	0.221	67	43	4.944	131	83	9.666	195	C3	14.388
4	4	0.295	68	44	5.017	132	84	9.740	196	C4	14.462
5	5	0.369	69	45	5.091	133	85	9.813	197	C5	14.536
6	6	0.443	70	46	5.165	134	86	9.887	198	C6	14.609
7	7	0.516	71	47	5.239	135	87	9.961	199	C7	14.683
8	8	0.590	72	48	5.313	136	88	10.035	200	C8	14.757
9	9	0.664	73	49	5.386	137	89	10.109	201	C9	14.831
10	A	0.738	74	4A	5.460	138	8A	10.182	202	CA	14.905
11	B	0.812	75	4B	5.534	139	8B	10.256	203	CB	14.978
12	C	0.885	76	4C	5.608	140	8C	10.330	204	CC	15.052
13	D	0.959	77	4D	5.681	141	8D	10.404	205	CD	15.126
14	E	1.033	78	4E	5.755	142	8E	10.477	206	CE	15.200
15	F	1.107	79	4F	5.829	143	8F	10.551	207	CF	15.273
16	10	1.181	80	50	5.903	144	90	10.625	208	D0	15.347
17	11	1.254	81	51	5.977	145	91	10.699	209	D1	15.421
18	12	1.328	82	52	6.050	146	92	10.773	210	D2	15.495
19	13	1.402	83	53	6.124	147	93	10.846	211	D3	15.569
20	14	1.476	84	54	6.198	148	94	10.920	212	D4	15.642
21	15	1.549	85	55	6.272	149	95	10.994	213	D5	15.716
22	16	1.623	86	56	6.345	150	96	11.068	214	D6	15.790
23	17	1.697	87	57	6.419	151	97	11.141	215	D7	15.864
24	18	1.771	88	58	6.493	152	98	11.215	216	D8	15.938
25	19	1.845	89	59	6.567	153	99	11.289	217	D9	16.011
26	1A	1.918	90	5A	6.641	154	9A	11.363	218	DA	16.085
27	1B	1.992	91	5B	6.714	155	9B	11.437	219	DB	16.159
28	1C	2.066	92	5C	6.788	156	9C	11.510	220	DC	16.233
29	1D	2.140	93	5D	6.862	157	9D	11.584	221	DD	16.306
30	1E	2.214	94	5E	6.936	158	9E	11.658	222	DE	16.380
31	1F	2.287	95	5F	7.010	159	9F	11.732	223	DF	16.454
32	20	2.361	96	60	7.083	160	A0	11.806	224	E0	16.528
33	21	2.435	97	61	7.157	161	A1	11.879	225	E1	16.602
34	22	2.509	98	62	7.231	162	A2	11.953	226	E2	16.675
35	23	2.582	99	63	7.305	163	A3	12.027	227	E3	16.749
36	24	2.656	100	64	7.378	164	A4	12.101	228	E4	16.823
37	25	2.730	101	65	7.452	165	A5	12.174	229	E5	16.897
38	26	2.804	102	66	7.526	166	A6	12.248	230	E6	16.970
39	27	2.878	103	67	7.600	167	A7	12.322	231	E7	17.044
40	28	2.951	104	68	7.674	168	A8	12.396	232	E8	17.118
41	29	3.025	105	69	7.747	169	A9	12.470	233	E9	17.192
42	2A	3.099	106	6A	7.821	170	AA	12.543	234	EA	17.266
43	2B	3.173	107	6B	7.895	171	AB	12.617	235	EB	17.339
44	2C	3.247	108	6C	7.969	172	AC	12.691	236	EC	17.413
45	2D	3.320	109	6D	8.043	173	AD	12.765	237	ED	17.487
46	2E	3.394	110	6E	8.116	174	AE	12.839	238	EE	17.561
47	2F	3.468	111	6F	8.190	175	AF	12.912	239	EF	17.635
48	30	3.542	112	70	8.264	176	B0	12.986	240	F0	17.708
49	31	3.615	113	71	8.338	177	B1	13.060	241	F1	17.782
50	32	3.689	114	72	8.411	178	B2	13.134	242	F2	17.856
51	33	3.763	115	73	8.485	179	B3	13.207	243	F3	17.930
52	34	3.837	116	74	8.559	180	B4	13.281	244	F4	18.003
53	35	3.911	117	75	8.633	181	B5	13.355	245	F5	18.077
54	36	3.984	118	76	8.707	182	B6	13.429	246	F6	18.151
55	37	4.058	119	77	8.780	183	B7	13.503	247	F7	18.225
56	38	4.132	120	78	8.854	184	B8	13.576	248	F8	18.299
57	39	4.206	121	79	8.928	185	B9	13.650	249	F9	18.372
58	3A	4.280	122	7A	9.002	186	BA	13.724	250	FA	18.446
59	3B	4.353	123	7B	9.076	187	BB	13.798	251	FB	18.520
60	3C	4.427	124	7C	9.149	188	BC	13.872	252	FC	18.594
61	3D	4.501	125	7D	9.223	189	BD	13.945	253	FD	18.668
62	3E	4.575	126	7E	9.297	190	BE	14.019	254	FE	18.741
63	3F	4.648	127	7F	9.371	191	BF	14.093	255	FF	18.815

ADC Lookup Table – Temperature

ADC		VIN	ADC		VIN	ADC		VIN	ADC		VIN
Decimal	HEX		Decimal	HEX		Decimal	HEX		Decimal	HEX	
0	0	–80.57	64	40	–1.34	128	80	25.00	192	C0	56.99
1	1	–72.89	65	41	–0.89	129	81	25.41	193	C1	57.67
2	2	–64.26	66	42	–0.44	130	82	25.82	194	C2	58.36
3	3	–58.84	67	43	0.01	131	83	26.24	195	C3	59.05
4	4	–54.80	68	44	0.46	132	84	26.65	196	C4	59.76
5	5	–51.55	69	45	0.90	133	85	27.07	197	C5	60.48
6	6	–48.81	70	46	1.34	134	86	27.49	198	C6	61.21
7	7	–46.43	71	47	1.78	135	87	27.91	199	C7	61.96
8	8	–44.32	72	48	2.21	136	88	28.33	200	C8	62.71
9	9	–42.41	73	49	2.64	137	89	28.75	201	C9	63.48
10	A	–40.68	74	4A	3.07	138	8A	29.18	202	CA	64.27
11	B	–39.08	75	4B	3.50	139	8B	29.60	203	CB	65.06
12	C	–37.59	76	4C	3.93	140	8C	30.03	204	CC	65.88
13	D	–36.20	77	4D	4.35	141	8D	30.46	205	CD	66.71
14	E	–34.89	78	4E	4.77	142	8E	30.89	206	CE	67.55
15	F	–33.66	79	4F	5.19	143	8F	31.32	207	CF	68.41
16	10	–32.49	80	50	5.61	144	90	31.76	208	D0	69.29
17	11	–31.37	81	51	6.03	145	91	32.20	209	D1	70.19
18	12	–30.31	82	52	6.45	146	92	32.64	210	D2	71.11
19	13	–29.29	83	53	6.86	147	93	33.08	211	D3	72.05
20	14	–28.31	84	54	7.27	148	94	33.52	212	D4	73.01
21	15	–27.36	85	55	7.68	149	95	33.97	213	D5	74.00
22	16	–26.45	86	56	8.09	150	96	34.42	214	D6	75.01
23	17	–25.57	87	57	8.50	151	97	34.87	215	D7	76.04
24	18	–24.72	88	58	8.91	152	98	35.33	216	D8	77.10
25	19	–23.89	89	59	9.32	153	99	35.78	217	D9	78.19
26	1A	–23.09	90	5A	9.72	154	9A	36.24	218	DA	79.31
27	1B	–22.31	91	5B	10.13	155	9B	36.71	219	DB	80.46
28	1C	–21.54	92	5C	10.53	156	9C	37.17	220	DC	81.65
29	1D	–20.80	93	5D	10.94	157	9D	37.64	221	DD	82.87
30	1E	–20.08	94	5E	11.34	158	9E	38.11	222	DE	84.13
31	1F	–19.37	95	5F	11.74	159	9F	38.59	223	DF	85.44
32	20	–18.68	96	60	12.14	160	A0	39.07	224	E0	86.78
33	21	–18.00	97	61	12.55	161	A1	39.55	225	E1	88.17
34	22	–17.34	98	62	12.95	162	A2	40.04	226	E2	89.62
35	23	–16.69	99	63	13.35	163	A3	40.53	227	E3	91.12
36	24	–16.05	100	64	13.75	164	A4	41.02	228	E4	92.67
37	25	–15.42	101	65	14.15	165	A5	41.52	229	E5	94.29
38	26	–14.81	102	66	14.54	166	A6	42.02	230	E6	95.98
39	27	–14.20	103	67	14.94	167	A7	42.52	231	E7	97.75
40	28	–13.61	104	68	15.34	168	A8	43.03	232	E8	99.59
41	29	–13.02	105	69	15.74	169	A9	43.55	233	E9	101.53
42	2A	–12.45	106	6A	16.14	170	AA	44.07	234	EA	103.57
43	2B	–11.88	107	6B	16.54	171	AB	44.59	235	EB	105.71
44	2C	–11.32	108	6C	16.94	172	AC	45.12	236	EC	107.98
45	2D	–10.76	109	6D	17.34	173	AD	45.65	237	ED	110.38
46	2E	–10.22	110	6E	17.74	174	AE	46.19	238	EE	112.93
47	2F	–9.68	111	6F	18.13	175	AF	46.74	239	EF	115.65
48	30	–9.15	112	70	18.53	176	B0	47.29	240	F0	118.57
49	31	–8.62	113	71	18.93	177	B1	47.84	241	F1	121.72
50	32	–8.10	114	72	19.33	178	B2	48.40	242	F2	125.12
51	33	–7.59	115	73	19.73	179	B3	48.97	243	F3	128.83
52	34	–7.08	116	74	20.13	180	B4	49.54	244	F4	132.89
53	35	–6.58	117	75	20.54	181	B5	50.12	245	F5	137.38
54	36	–6.08	118	76	20.94	182	B6	50.71	246	F6	142.40
55	37	–5.59	119	77	21.34	183	B7	51.30	247	F7	148.06
56	38	–5.10	120	78	21.74	184	B8	51.90	248	F8	154.56
57	39	–4.62	121	79	22.15	185	B9	52.51	249	F9	162.13
58	3A	–4.14	122	7A	22.55	186	BA	53.13	250	FA	171.18
59	3B	–3.66	123	7B	22.96	187	BB	53.75	251	FB	182.34
60	3C	–3.19	124	7C	23.36	188	BC	54.38	252	FC	196.72
61	3D	–2.72	125	7D	23.77	189	BD	55.02	253	FD	216.58
62	3E	–2.26	126	7E	24.18	190	BE	55.67	254	FE	247.46
63	3F	–1.80	127	7F	24.59	191	BF	56.33	255	FF	310.08
63	3F	4.648	127	7F	9.371	191	BF	14.093	255	FF	18.815

Error Code Detailed Description

Status Error Flag	Error Code	Description
Exceed Input Voltage limit	0x01	Voltage too low
	0x02	Voltage too high
Exceed Temperature limit	0x03	Temperature too high
Servo Missing	0x11	No reply from servo while reading servo register during self check mode.
	0x12	No reply from servo while reading servo register during Task execution.
EEP REG distorted	0x21	Wrong model name in EEPROM
	0x22	Wrong EEPROM ID
	0x23	EEPROM data corrupt
Servo Status Error	0x31	Servo status error
	0x32	DRT–HWW1 status error
	0x33	Too many servos connected to DRC
	0x34	DRC–004T0 status error
Invalid Packet	0x41	Zigbee Ack not received properly or Noise interference received
	0x42	Check Sum Error in Zigbee Ack
	0x43	Unknown Command in Zigbee Ack
	0x44	Received Zigbee Ack but ID is not 0xFC
	0x45	Packet size received in Zigbee Ack too large
	0x46	Packet size received in Zigbee Ack incompatiable with command
	0x47	Zigbee Ack not received
	0x51	Packet received in Zigbee Ack incomplete or Noise interference received
	0x52	Check Sum Error in Servo Ack
	0x53	Unknown Command in Servo Ack
	0x54	Invalid ID packet received in Servo Ack
	0x55	DRT–HWW1 related command received from Servo Ack but ID is not 0XFB
	0x56	Packet size received in Servo Ack too large
	0x57	Packet size received in Servo Ack incompatiable with command
	0x58	UART Buffer receiving packet in Servo Ack is full
	0x59	Buffer for saving packet to be sent to Servo is full
	0x5A	SDRC–004TO related command received from Servo Ack but ID is not 0XFA
	0x61	Packet received by PC incomplete or Noise interference received
	0x62	Check Sum Error in packet received by PC
	0x63	Unkown Command in packet received by PC
	0x64	Invalid ID packet received by PC
	0x65	DRT–004TO related command received from PC packet but ID is not 0XFB
	0x66	Packet size received by PC too large
	0x67	Packet size received by PC incompatiable with command
	0x68	UART Buffer receiving packet by PC is full
	0x69	DRC–004TO related command received from PC packet but ID is not 0XFA
	0x71	EEP/RAM WRITE/READ command beyond register range
	0x72	Incorrect value used in RAM_WRITE
	0x73	Incorrect value used in RAM_WRITE Status
	0x74	Incorrect ID in CON_CHECK packet
	0x75	Incorrect motion number in PLAY_MOTION
	0x76	Incorrect instruction in PLAY_TASK
	0x77	Incorrect Channel or Length in REMOCON
	0x78	Incorrect instruction in ZIGBEE
	0x79	Incorrect buzzer number in PLAY_BUZZ

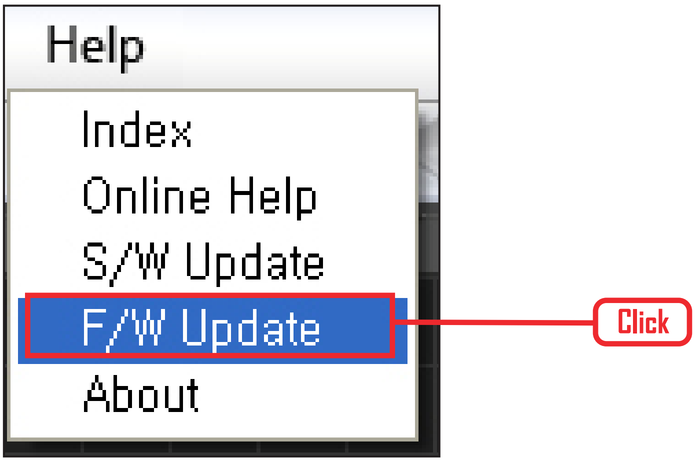
Error Code Detailed Description

Status Error Flag	Error Code	Description
Flash Data Distorted	0x81	Trying to run non existing Motion
	0x82	Problem with Motion data
	0x83	Number of axis in Motion data different than actual number of axis
	0x84	Frame with negative time to next frame
	0x85	Too many Repeat commands stacked (Maximum 3)
	0x91	Problem with Task data
	0x92	Error while performing arithmetic operation
	0x93	Program stack overflow
	0x94	Incorrect register address while loading MPSU RAM
	0x95	Incorrect register length while loading MPSU RAM
	0x96	Incorrect register address while loading Servo RAM
	0x97	Incorrect register length while loading Servo RAM
	0x98	Incorrect ID while loading Servo RAM
	0x99	Incorrect register length while reading MPSU RAM
	0x9A	Incorrect register length while reading Servo RAM
	0x9B	Incorrect ID while reading Servo RAM
	0xA1	Value in Motion command beyond range
	0xA2	Value in Motion Ready beyond range
	0xA3	Value in Servo control command beyond range
	0xA4	Head LED command value out of range
	0xA5	Value in DRC LED command beyond range
	0xA6	Vlaue in Buzzer melody command beyond range
	0xA7	Value in Buzzer note command beyond range
	0xB1	Trying to run non existing head LED
	0xB2	Trying to play non existing Buzzer

Troubleshooting

Firmware Update

Example Explanation



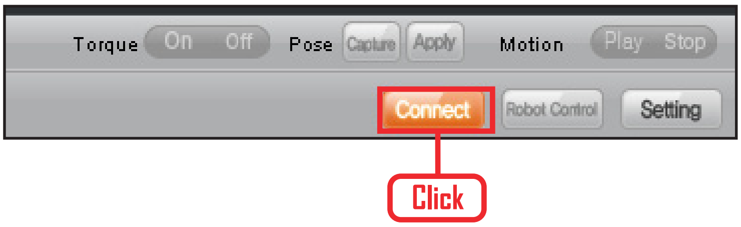
01 Help > Firmware Updae

Update controller firmware through DR-Visual Logic.
With the controller connected to the PC.
Help 〉Click firmware update.

Calibration (0 Point Adjustment)

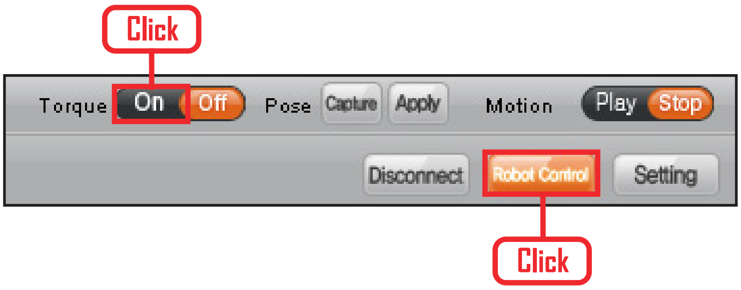
Checks robot to see if it was assembled correctly/exactly and makes adjustment if necessary. If the robot was not assembled correctly, it may cause error or unwanted movement.

Click 'Robot Control' in DR-SIM to adjust the position of the robot motors.



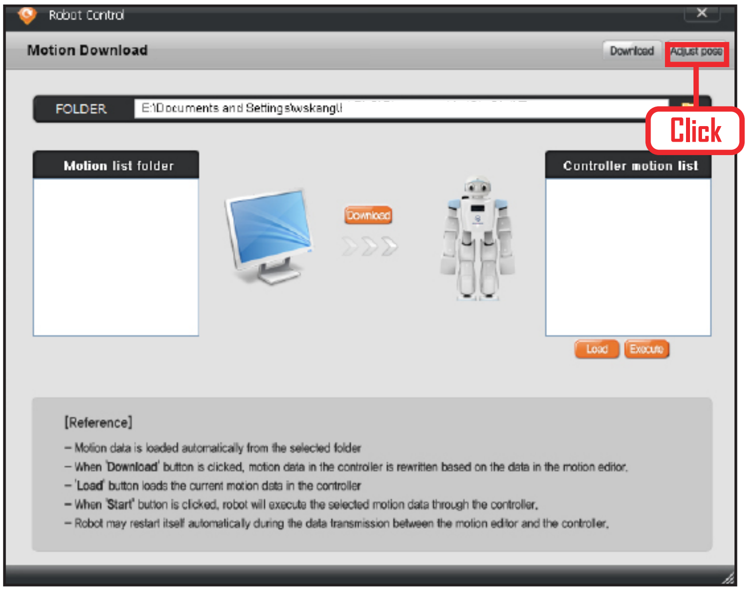
01 Connect

Connect to robot.
Click Connect.



02 Robot Control

Turn on power.
Click Torque On.
Calibration is done in Robot Control.
Click Robot Control.



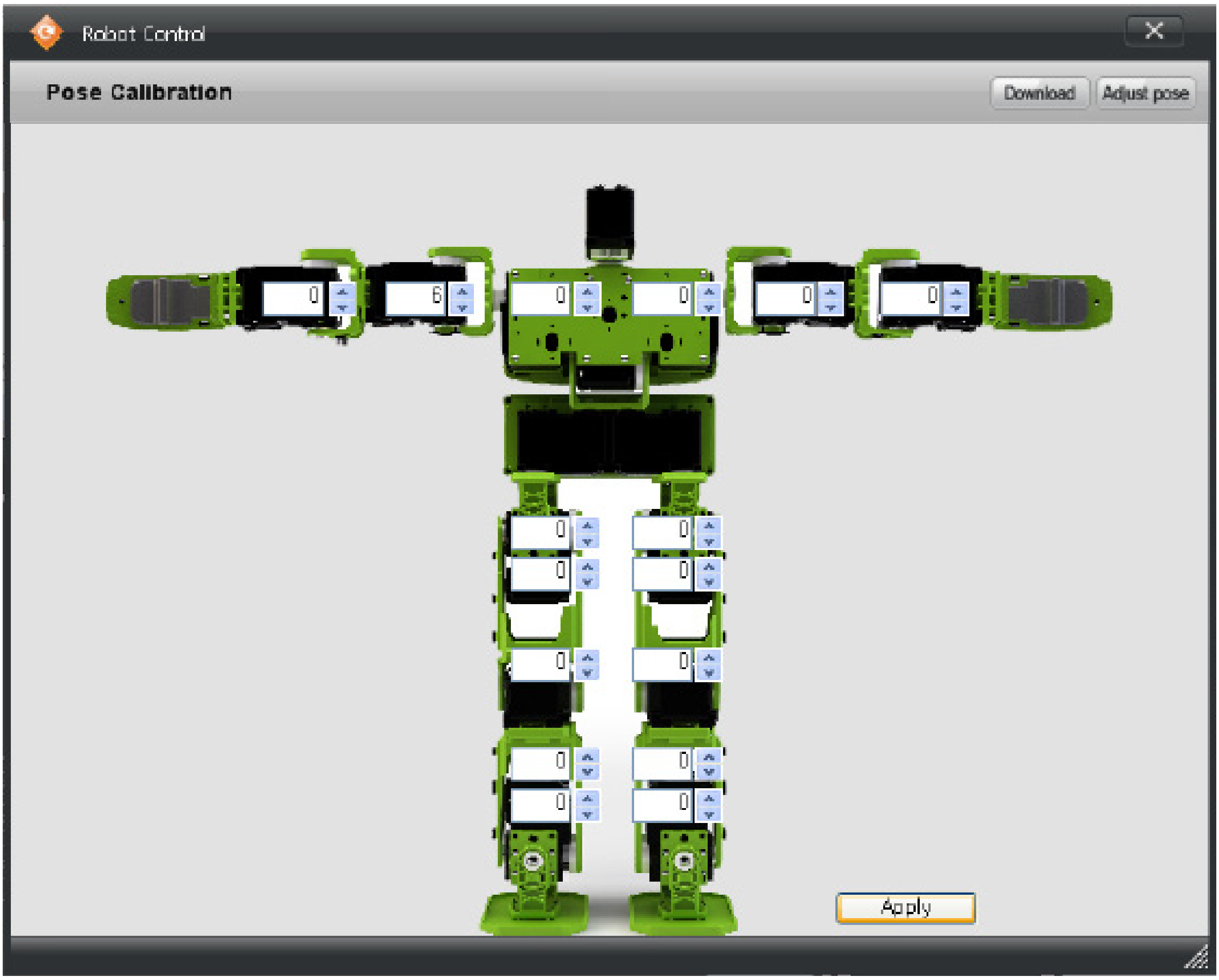
FOLDER E:\Documents and Settings\wskang\

[Reference]

- Motion data is loaded automatically from the selected folder
- When 'Download' button is clicked, motion data in the controller is rewritten based on the data in the motion editor.
- 'Load' button loads the current motion data in the controller
- When 'Start' button is clicked, robot will execute the selected motion data through the controller.
- Robot may restart itself automatically during the data transmission between the motion editor and the controller.

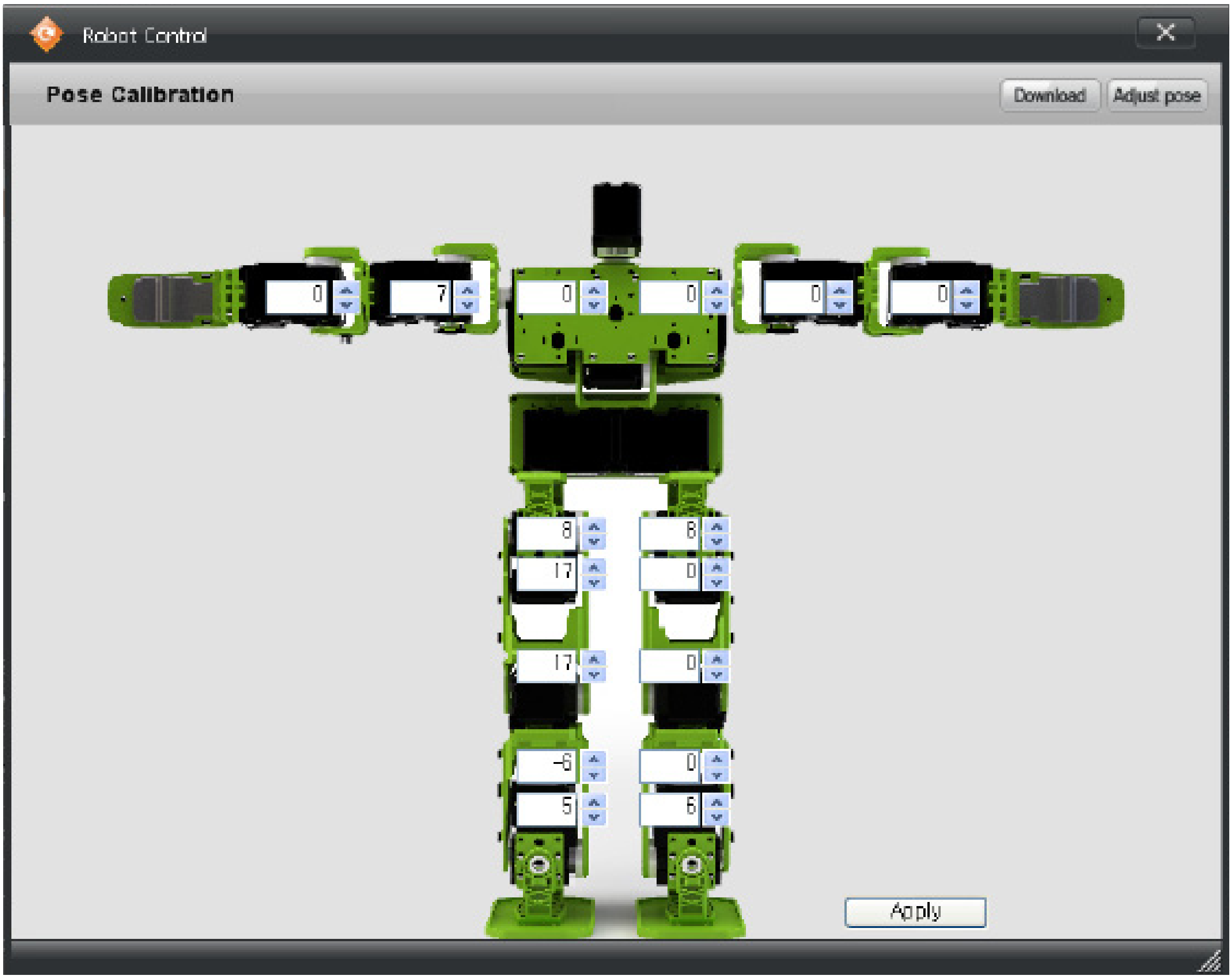
03 Posture Adjustment

Robot control window is divided into motion download and posture adjustment.
Click Posture Adjustment



Clicking Posture Adjustment will show current calibrantion values.
Compare with the actual robot and adjust the calibration values.

Calibration value range is from –128 ~ 127. Use the Up/Down button to change the values and notice the actual robot making slight movements.



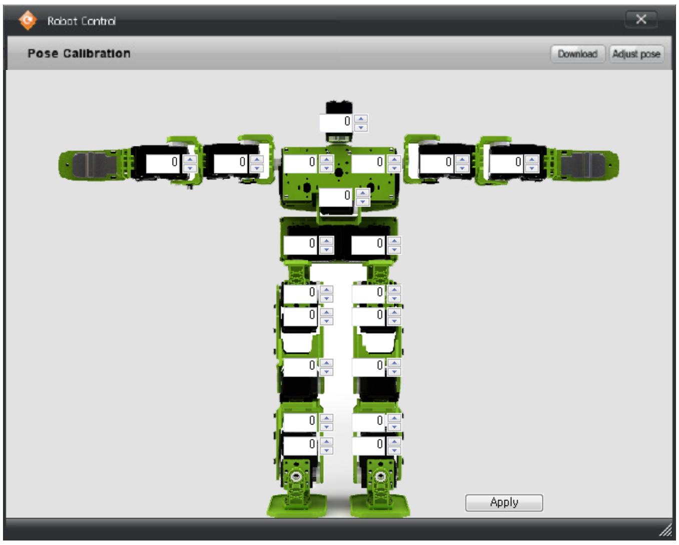
Check the robot to view the adjustments being made and click Apply when the correct setting is achieved.
Press Apply to save the adjustment to the robot. Robot will show adjusted values when connected.

Checks robot to see if it was assembled correctly/exactly and makes adjustment if necessary. If the robot was not assembled correctly, it may cause error or unwanted movement.

Click ‘Robot Control’ in DR-SIM to adjust the position of the robot motors.

01

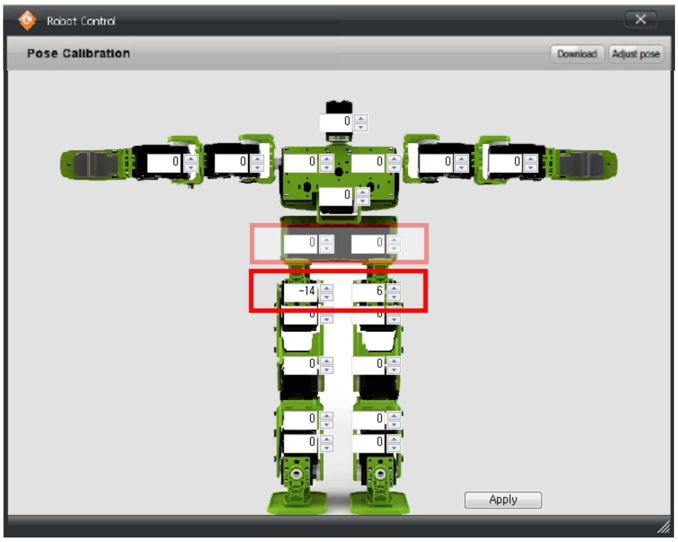
Click Robot Control 〉 Posture Adjustment button. When the posture adjustment window opens up, lift up the robot and check the assembly.

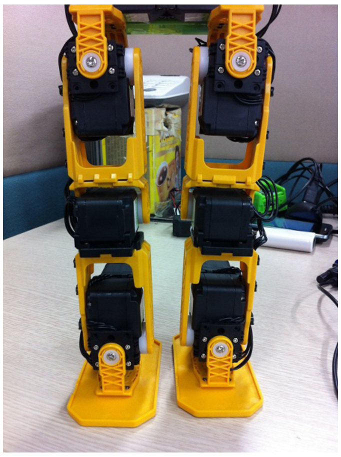


02

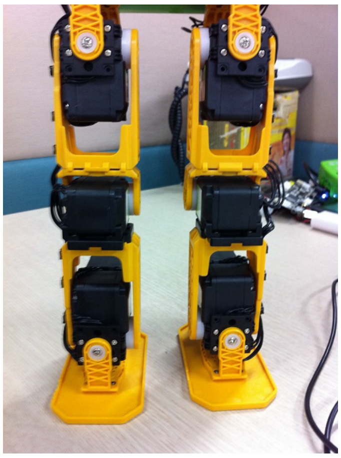
View the robot directely from the front and adjust the leg balance.

* Square boxes apply to 18 axis and 20 axis robots.





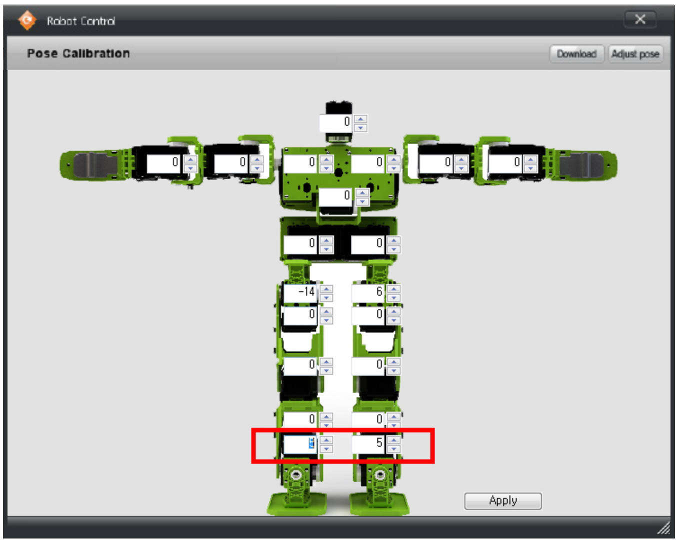
〈Before〉



〈After〉

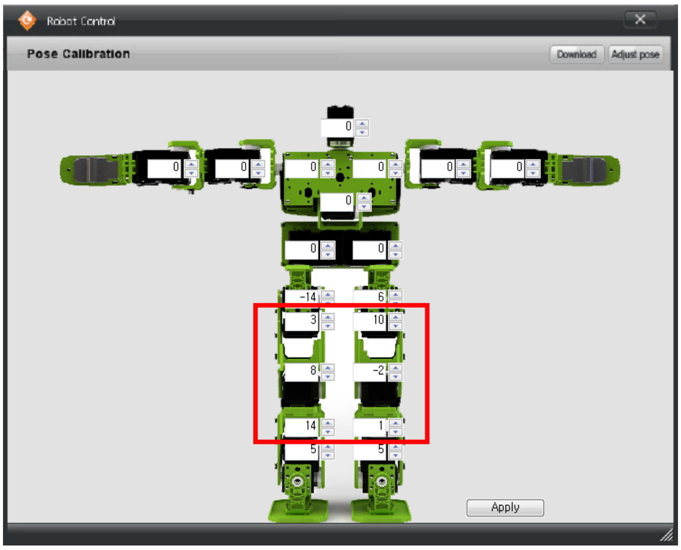
03

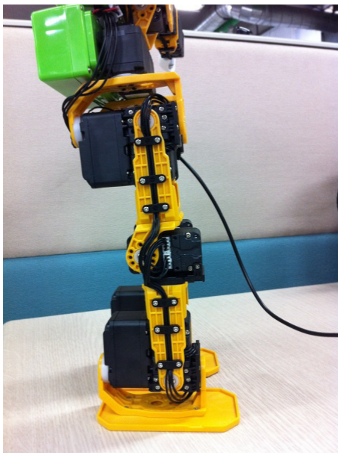
Make adjustments so that both feet are flat on the ground.



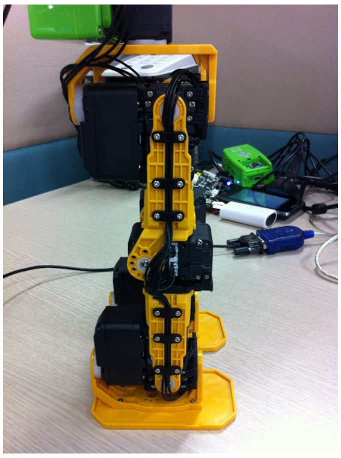
04

View the robot from the side and adjust the vertical angle.

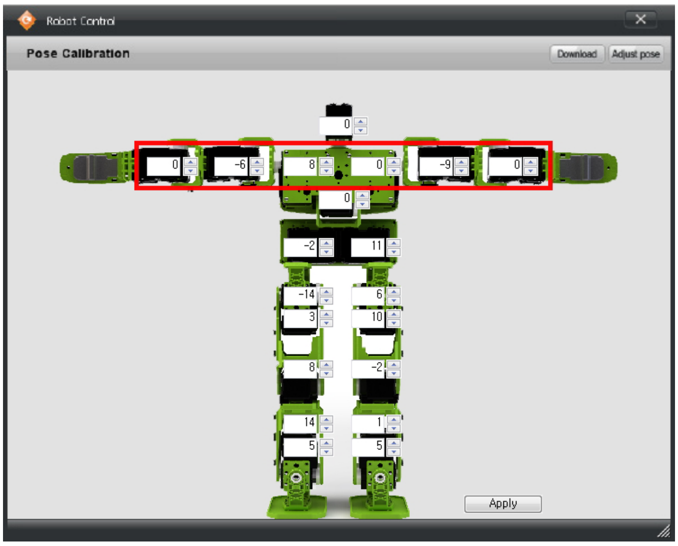




〈Before〉

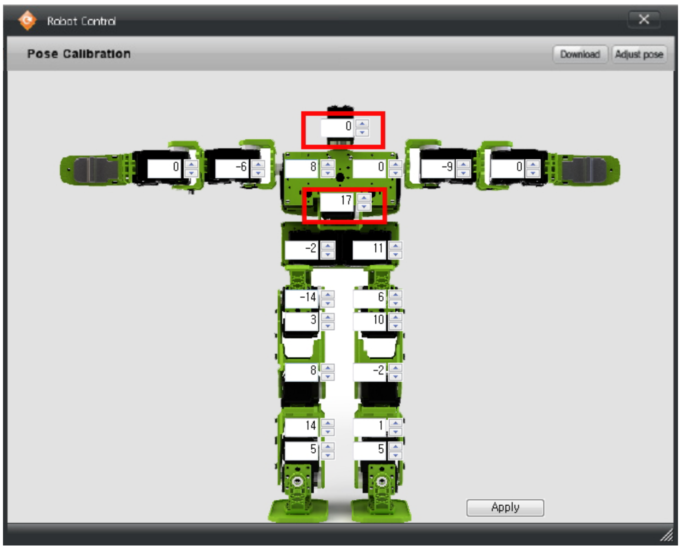


〈After〉



05

View the robot from the top and adjust the arms to form straight line.



06

Adjust the waist and head for 20 Axis robot.

07

Make further necessary adjustments and end the calibration.

Changing the Motor ID

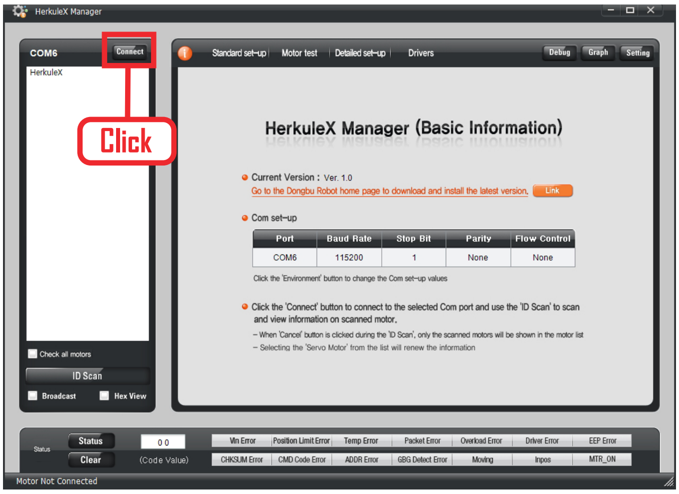
Since DRC identifies each motor by the motor ID number, it is important to place each motor in correct position according to the ID when assembling the robot. However, if the motor was incorrectly positioned or if the robot is being reassembled from 16 axis to 18 or 20 axis robot, motor ID change will be necessary.

- Make sure to change the motor ID prior to reassembling the robot from 16 axis to 18 or 20 axis.
- Follow the steps below to change the motor ID if the motors were positioned incorrectly during the assembly.

  ex) Position of the motors ID 9 and 10 were switched.

  ID 9 → ID100 (Motor ID 20 to 254 are spare ID.)
  ID 10 → ID 9
  ID 100 → ID 10

Example shown below uses HerkuleX Manager program to change the motor ID 253 to ID 15.
(HerkuleX Manager prgogram can be downloaded from Dongbu Robot website.)
http://www.dongburobot.com/jsp/cms/view.jsp?code=100122



HerkuleX Manager (Basic Information)

- Current Version : Ver. 1.0
  Go to the Dongbu Robot home page to download and install the latest version.
- Com set-up

Port	Baud Rate	Stop Bit	Parity	Flow Control
COM6	115200	1	None	None

Click the 'Environment' button to change the Com set-up values

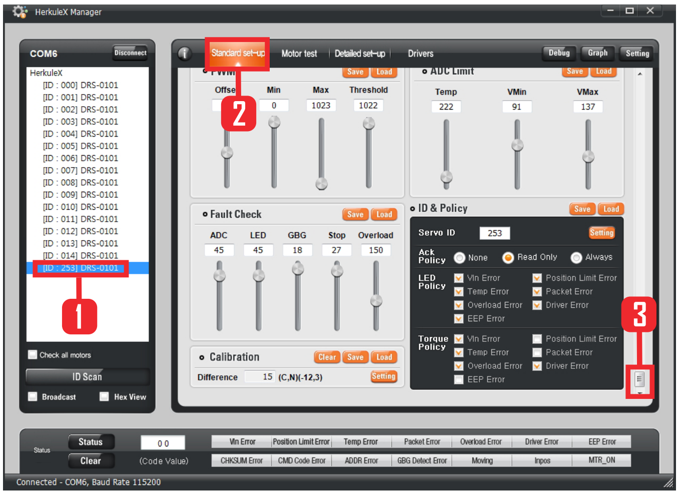
- Click the 'Connect' button to connect to the selected Com port and use the 'ID Scan' to scan and view information on scanned motor.
  - When 'Cancel' button is clicked during the 'ID Scan', only the scanned motors will be shown in the motor list
  - Selecting the 'Servo Motor' from the list will renew the information

0 0 (Code Value)

Motor Not Connected

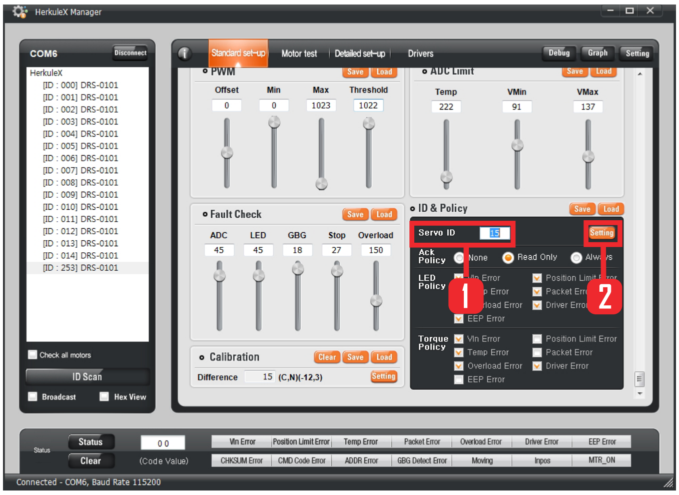
01

Connect the motor to the controller (DRC) and run the HerkuleX Manager program. Setup the COM Port and click Connect button.



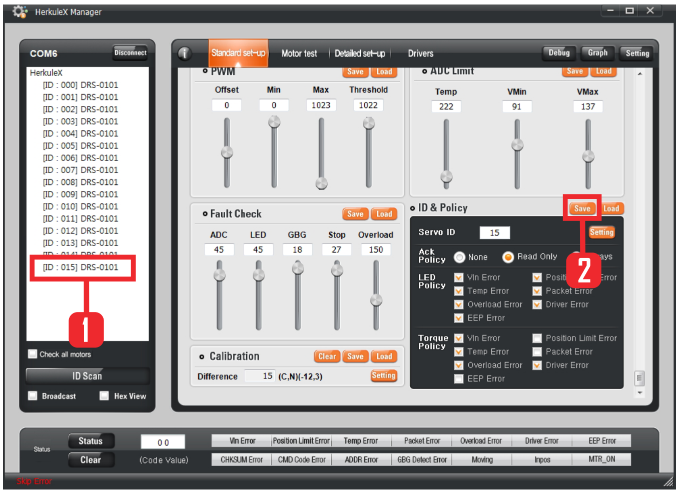
02

Motor connected to the controller (DRC) shows up in the left window when Connect button is clicked. Click [ID: 253] DRS-0101 to change the motor ID 253 to 15. Next, click on basic properties and then use the scroll bar to position the ID&Policy window so that it becomes visible.



03

Enter desired value in Servo ID (value is 15 in this example) and then click Setup. Motor ID scan will run automatically when Setup is clicked.



04

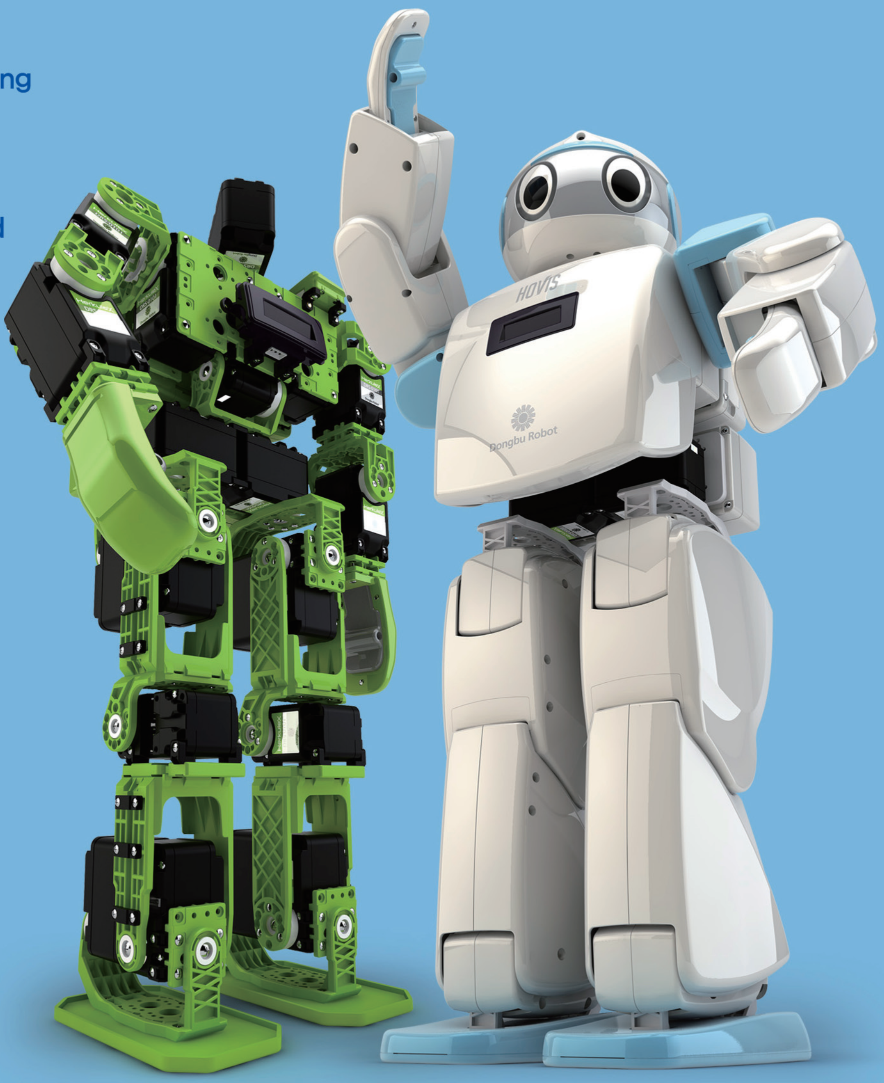
ID scan will show that Motor ID has changed from 253 to 15. As the last step, click [ID: 015]DRS-0101 and then click Save button to save the changed motor ID. (Changed motor ID shown by the ID Scan is from the changed RAM Register value which looses its data when power is turned off. Clicking the Save button will save the changed Motor ID in EEP Register which retains data even when the power is turned off.)

05

Disconnect and reconnect power on motor. Then run ID SCAN through HerkuleX Manager to varify the ID of motor.

HOVIS DRC & Visual Logic Robot Programming

Learn algorithm and robot control using graphic programming tool Visual Logic.



Dongbu Robot

Dongbu Robot Co.,Ltd www.dongburobot.com

HQ : 11th Fl Bucheon Techno Prk Bldg, 401, Yakdae–dong, Wonmi–gu, Bucheon–si, Gyeonggi–do, S Korea
TEL : 032–329–5551 (ext 112), FAX : 032–329–5569, E–MAIL : robotsales@dongbu.com
Factory : 6 Gil 27, 4 San dan, Jiksan–eup, Sebouk–gu, Cheonan–ski, Chungcheonnam–do, S Korea,
TEL : 041–590–1700, FAX : 041–590–1701

93550
9 788998 873103
ISBN 978-89-98873-10-3