

Operation Manual of Beauto Builder R

Software made specifically for Beauto Racer

Vstone Co., Ltd.



目次

1. Introduction	4
1-1. Installation of software	5
1-2. Connection of the robot to PC/Recognition	6
1-3. Explanation of hardware of the robot	9
2. How to create a sequence program	10
2-1. Explanation of window	10
2-2. Addition of action block	11
2-3. Creation of flowchart	12
2-4. Writing/Execution of program	13
2-5. Adjustment of motor speed	16
2-6. Addition of other action blocks	18
2-7. Setting of details of command	20
2-8. Save/Load of program	22
2-9. LED/Wait command	23
3. Creation of a program using the sensors	29
3-1. Check of sensor reaction	29
3-2. How to use Branch	31
3-3. Programming for Line Trace	38
3-4. Line Trace using the right and left sensors	43
3-5. RANDOM command	50
4. Creation of program using LOOP	51
5. Using Simulator	55
5-1. Running Simulator	55
5-2. Explanation of Simulator window	55
5-3. The operation of Simulator	56
5-4. Adjustment of the virtual robot motor speed	58
5-5. Checking the Sensor values	58
5-6. Menu and Toolbar in Simulator Windows	59
5-7. Select the course	60
5-7-1. trying sequence program course (course1/2)	61
5-7-2-7. Using sensors (course 3)	62

5-7-3. Line Trace using a sensor (course 4,5)	63
5-7-4. Line Trace using the sensors (course 6)	64
5-7-5. Figure of 8 line trace and the advanced Line trace.(Course7,8)	65
5-7-6. The original course (course 9)	65
6. Functions for advanced-level users	67
6-1. Wheel control with setting of speed and turning rate	67
6-2. Display of Memory Map	70
6-3. Use of Arithmetic block	71
6-4. Branch command for advanced-level users	74
6-5. Judgment for overlap of blocks	75
6-6. Check of LOOP connection	75
7. Convenient functions for programming	76
7-1. Move of action block	76
7-2. Delete of action block	77
7-3. Copy/Cut/Paste of action block	77
7-4. Selection of action block	78
7-5. Operation of arrow path	78
7-6. Explanation of menu/toolbar	79
7-7. Explanation of shortcut key	81
8. Others	82
8-1. Answers to the exercises	82
8-2. Load of program from the robot body	85
8-3. Update the firmware of the robot	86
8-4. Save screenshot of Program area	87
8-5. Q&A	88

1. Introduction

We thank you for purchasing "Beauto Racer" (hereinafter, referred to as the "**robot**" or "**robot body**"). Beauto Builder R (hereinafter, referred to as "this software") is software for making programs for the robot on PC connected to the robot. "Operation Manual of Beauto Builder R" (hereinafter, referred to as "this document") describes how to operate this software. This document explains the operation procedure with programming exercises and is designed on the assumption that the user practices, actually operating the robot. Therefore, it is recommended to read this document step by step from the beginning.

1-1. Installation of software

First, install this software on PC. This software operates on PC having the following environment:

- **OS: Windows2000/XP/Vista**
- **CPU: Pentium-III or higher (1 GHz or higher recommended)**
- **RAM: 128 MB**
- **Interface: USB port**
- **Screen size: XGA(1024×768) or more**
- **Simulator requires the installations of DirectX9.0b or newer**

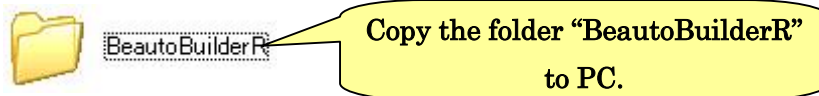
If you have the CD-ROM included in accessories of this product at hand, set the CD to PC and open the CD Drive under My Computer. The CD contains a folder with the name "Beauto Builder R". Copy the folder to any place (such as Desk Top) on PC. Now, the installation is completed.

If you don't have the CD-ROM, access to the User Support page below and download the latest version of the software.

● Beauto Racer Support Page URL

<http://www.vstone.co.jp/top/products/robot/beauto/rdownload.html>

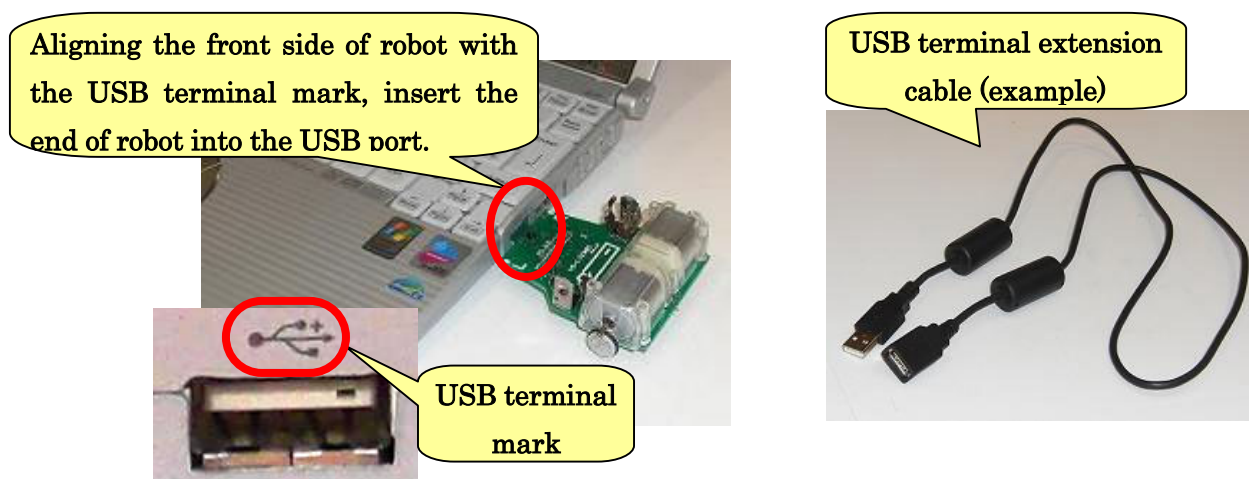
Since the downloaded file has a ZIP compression format, uncompress the file. After uncompression, a folder with the name "BeautoBuilderR" is created. Copy the folder to any place on PC (such as Desk Top).



1-2. Connection of the robot to PC/Recognition

After completing the installation of this software, let's connect the robot to PC and check that PC recognizes the robot. First, **turn OFF the power switch on the robot body**. Then, with the front side of robot facing to the **USB terminal mark** (see the figure below), insert the end of robot into the USB terminal of PC.

For your information, using a commercially available "USB terminal extension cable", the robot can be connected to PC via the cable without being directly connected.



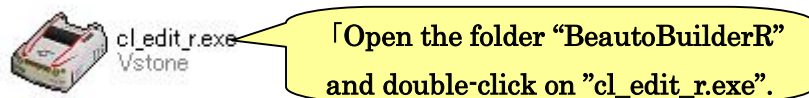
Before connecting the robot to PC, be sure to turn off the power switch on the robot. If the power switch is ON, interruption of communication may occur or the motor may run by itself.

When PC and the robot are connected for the first time, a dialogue balloon as shown below appears on the PC screen. If the message "A new hardware has been installed and PC is now ready for use." is displayed, PC has recognized the robot successfully.

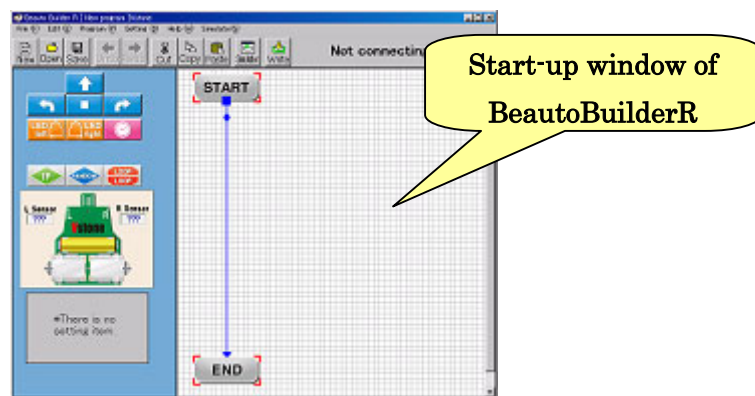


When PC and the robot are connected next time, PC automatically recognizes the robot without displaying the dialogue balloon.

Now, in order to make sure that the above-mentioned procedure was conducted properly, let's start up this software and communicate with the robot. First, open the folder "BeautoBuilderR" copied to PC. You will find the execution file "cl_edit_r.exe" in the folder. Double-click the file for execution.

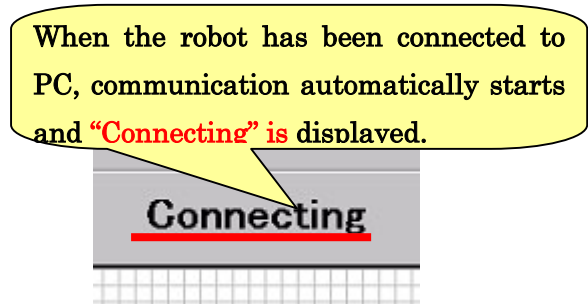
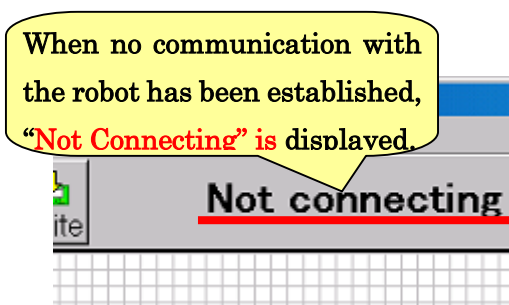


After the double-click, this software starts up and the window shown below appears on the screen.

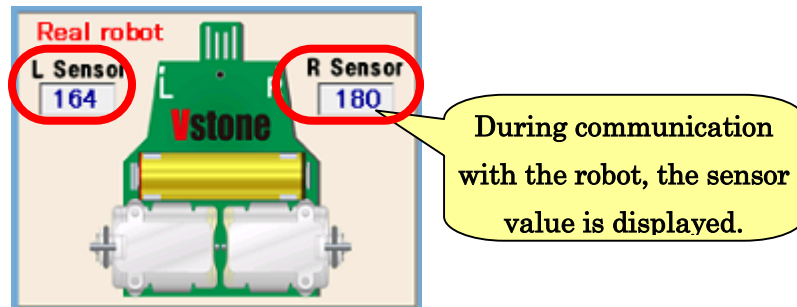


※ When the error message["Gdiplus.dll Not Found"] is displayed, Open the folder"gdipius" in the folder "BeautoBuilderR",copy the file "gdiplus.dll" and put it in the folder that contains the file "cl edit r.exe".

In the upper right of the window, the current status of communication with the robot is displayed in text. In a case where communication with the robot has not been established, such as in the case where the robot has not been connected to PC, **“Not Connected”** is displayed. When PC and the robot has been connected, communication automatically starts and **“Now connecting”** is displayed in the upper right of the window.



In the fields for “**Left sensor**” and “**Right sensor**” of the left part of the window, the current sensor values of the robot are displayed.

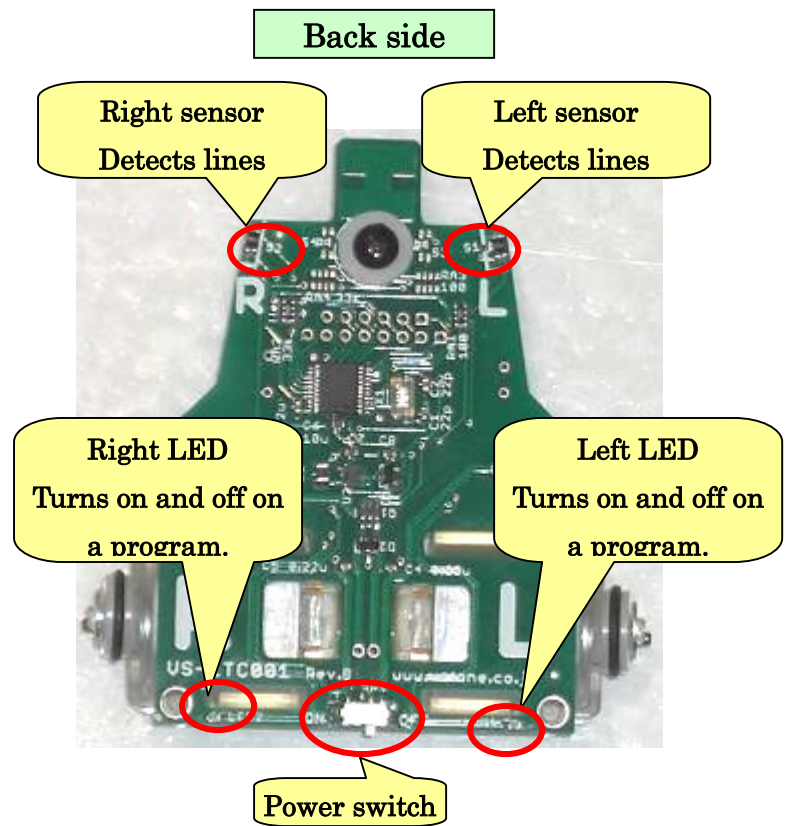
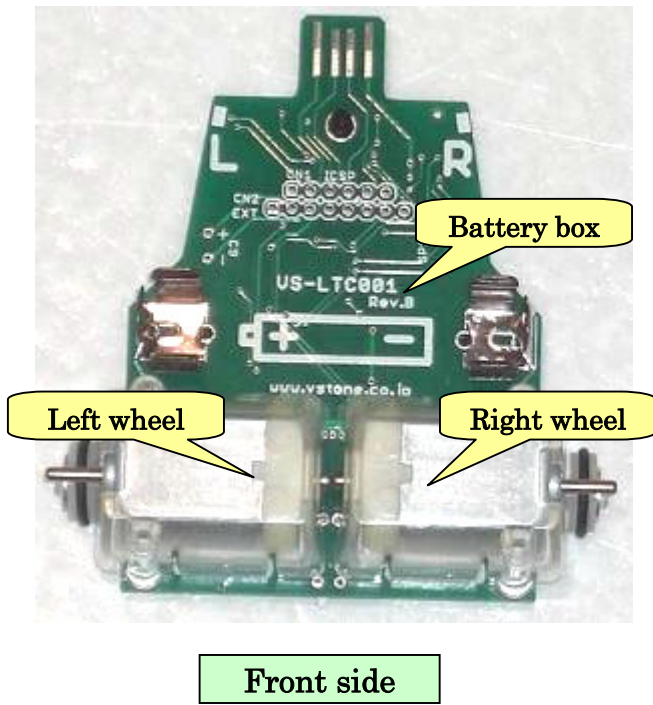


During communication with the robot, the sensor value is displayed.

- The robot has not been fully inserted into the USB terminal of PC as far as it goes. The robot has been inserted obliquely.
 - Insert the robot into the terminal straight as far as it will go so that PC and the robot is properly connected.
- The robot has been inserted with its front side and back side reversed.
 - With the front side and back side of the robot in the correct orientation, insert the robot into the USB terminal so that PC and the robot is properly connected.
- The robot has been inserted into and removed from PC many times.
 - If the robot has been inserted into and removed from PC many times, it may take a little longer time for PC to recognize the robot. In this state, communication is established properly, but if you feel uncomfortable with it, reboot the PC.

1-3. Explanation of hardware of the robot

The robot body is equipped with interfaces including motors, sensors and LEDs. The location and name of each equipment are as shown below.



2. How to create a sequence program

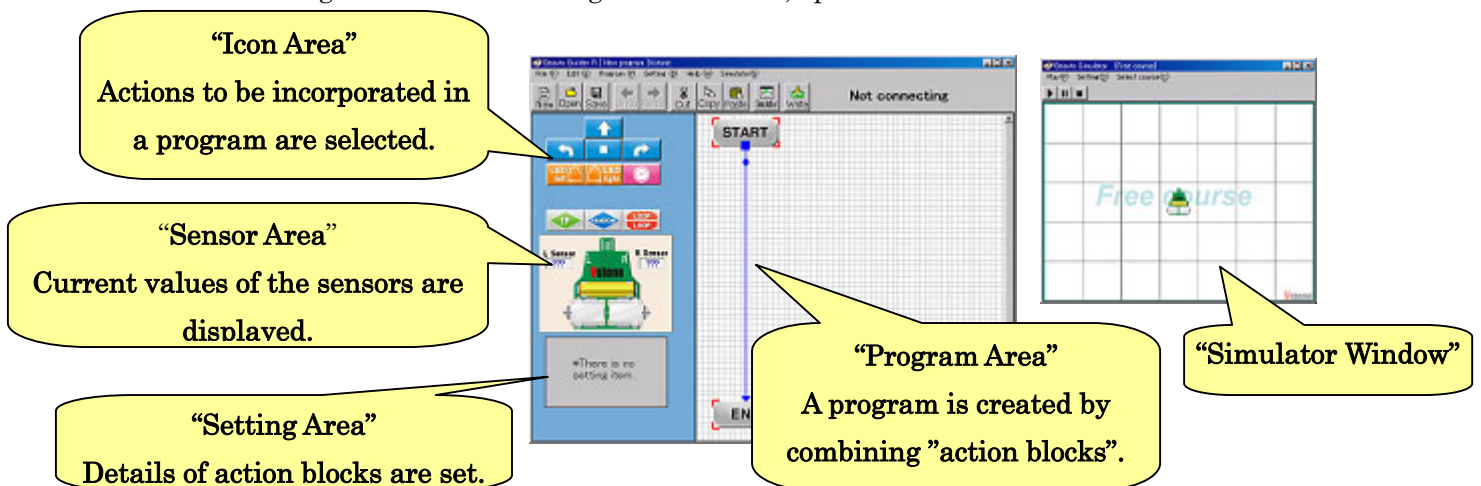
Hereinafter, programming using this software will be described. First, the basic points such as the basic operation of this software and usable commands will be explained, actually programming.

2-1. Explanation of window

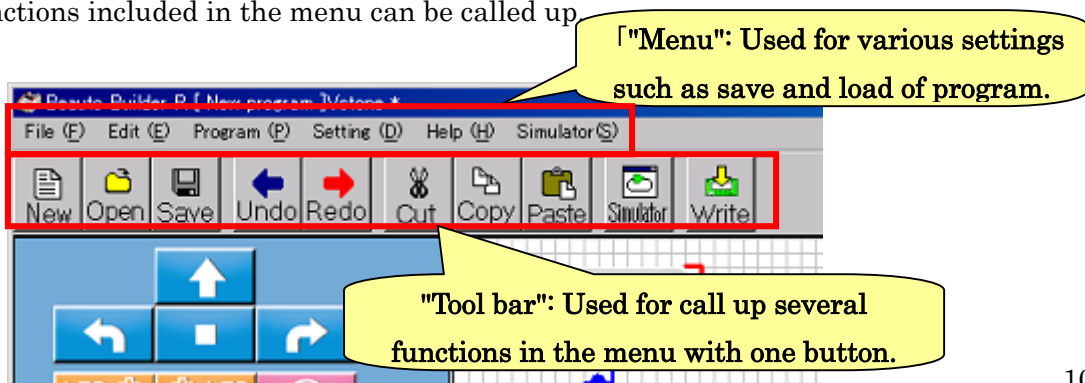
The window of this software is roughly divided into a left part and a right part. The left part is roughly into three areas.

The areas in the left part of the screen are called, in the order from the top, "Icon Area" in which "actions" (commands) to be used in a program are selected, "Sensor Area" in which the current sensor status of the robot is displayed, and "Setting Area" in which details of an action is set.

The right part of the screen is called "Program Area" in which a program is created by combining actions. When using the simulator, open Simulator Window.

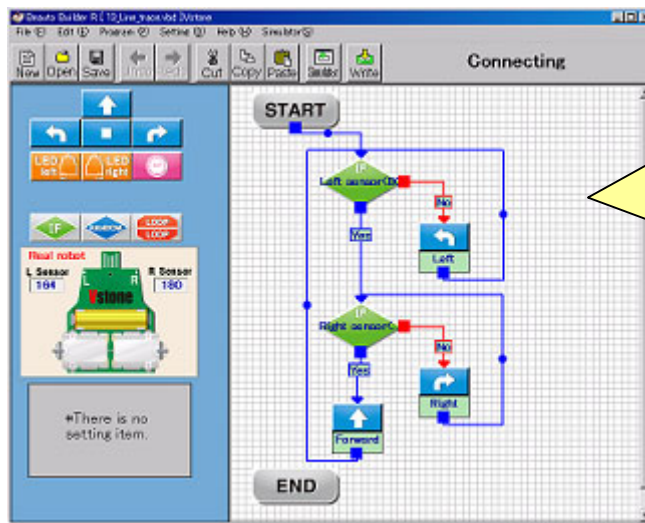


Furthermore, at the top of the window, the "menu" and "toolbar" are provided. The menu refers to the character strings directly below the window title, which is used for save and load of programs and setting for the robot. The toolbar refers to the buttons arranged side by side directly below the menu. With one button in the tool bar, some of functions included in the menu can be called up.



In the Program Area in the right part of the window, actions are displayed in the form of "action block" color-coded and symbolized according to the kind in a way easy to understand. Action blocks are connected with blue and red arrows, which indicate the sequence of execution of a program. A program always starts with the action block of "Start" and end with the action block of "Exit".

In this way, this software allows you to create a program in a flowchart form by connecting various kinds of action blocks to meet the purpose of the program.



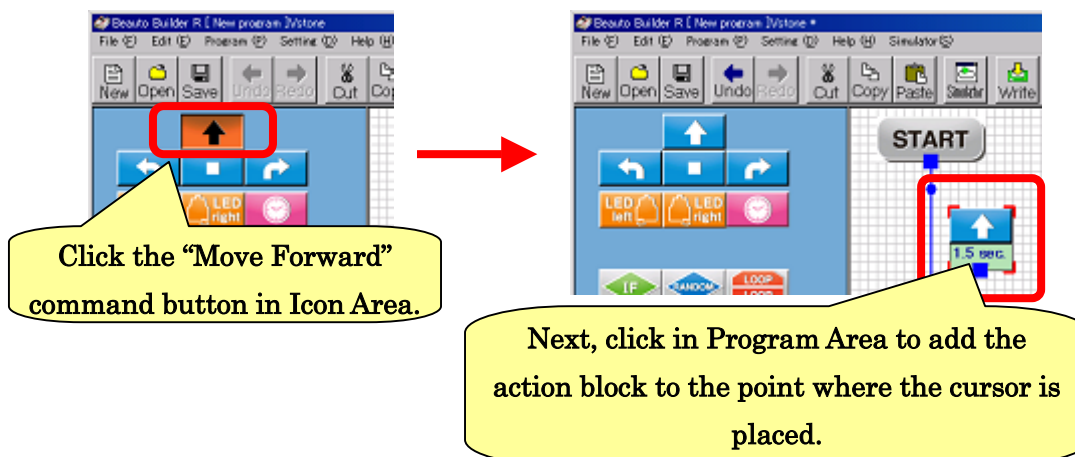
Example of a window where a program is being created. Just like making a flowchart, a program can be made by connecting "action blocks".

2-2. Addition of action block

Now, let's make a sequencst program where the robot just moves forward.

First, an action block (command) to be used in the program is selected from the Icon Area in the left part of the window, and then it is added to a place you like in the Program Area in the right part of the window.

Click the up-arrow button (see the figure below) from the Icon Area. This button is for the command "Move Forward". By clicking the button, its color changes. Next, click on a point you like in the Program Area with the mouse. With the click of the mouse, the command selected in the Icon Area are added to the point as the "action block".



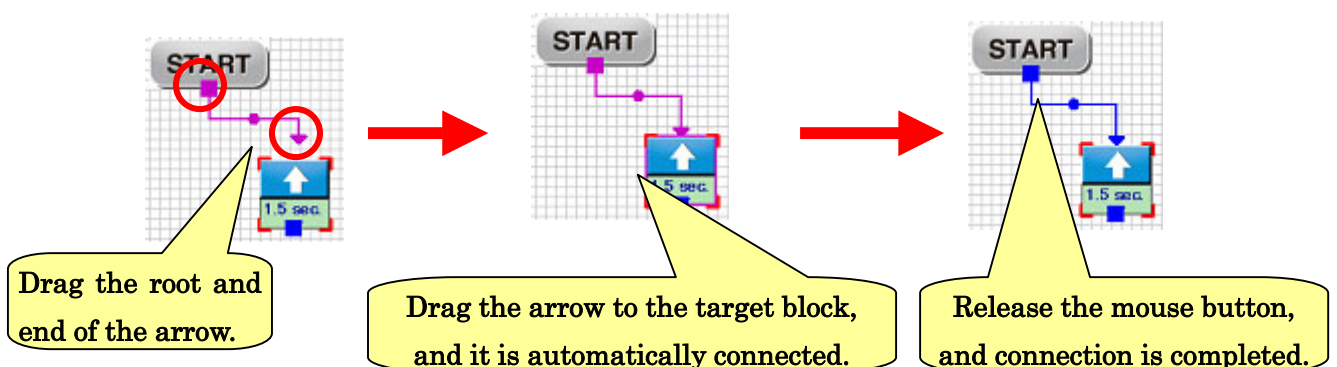
Thus, commands used in a program can be added by the procedure; ” **In the Icon Area, click the command to be used**” → ” **Click a point in the Program Area where the command is to be added**”

2-3. Creation of flowchart

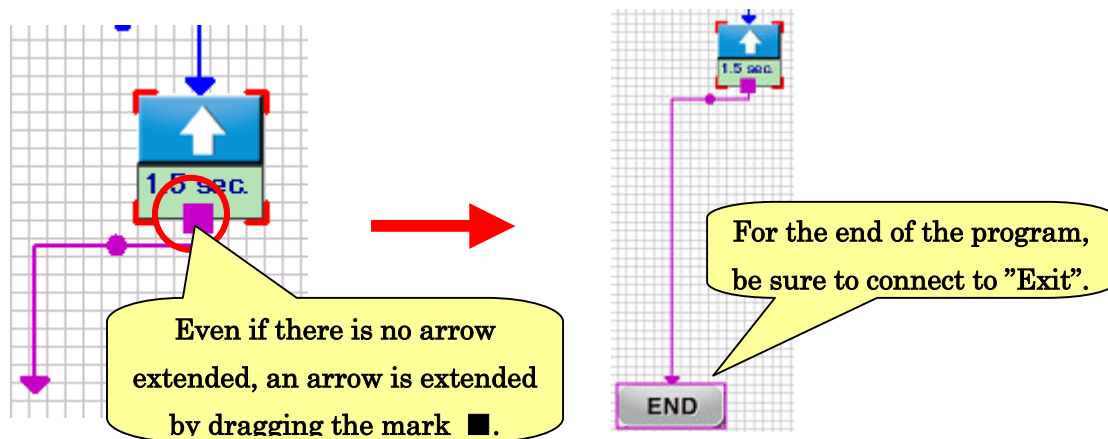
Now, the command to ”Move the robot forward” was added in the program. The robot, however, does not move as it is even if the program is executed, **because the added command “Move Forward” has not been incorporated between the ”Start” and ”Exit” arrows in the program.** In order to make the robot execute the added command, the arrows of the action blocks need to be re-connected to incorporate the added action block in the flowchart. Let’s re-connect the arrows as mentioned below.

To re-connect the arrow of action block, **click the root or end of arrow of the action block and drag.** By dragging the mark ■ at the root of arrow or the mark ▲ at the end of arrow, **the color of the arrow changes to purple and the tip of arrow can be moved freely.** By dragging the tip of arrow and putting the mouse cursor on another action block, the arrow is automatically connected to the action block. Then, by releasing the mouse button, the arrow is re-connected.

As shown in the figures below, connect the arrow of the action block ”Start” to the action block ”Move Forward”.




In the same way, connect the arrow of the action block “Move Forward” to the action block “Exit” . There is no arrow extended from the action block “Move Forward” , but by clicking the mark ■ at the root, an arrow is extended.

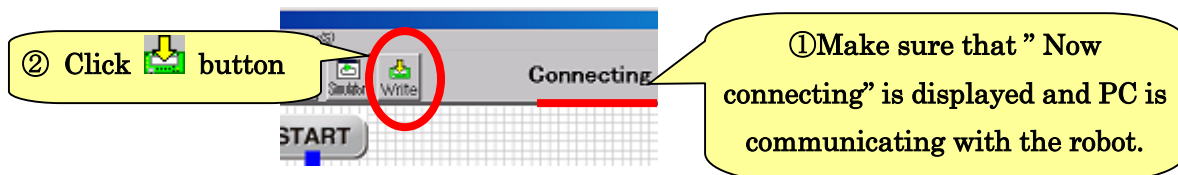



As described above, be sure to connect the arrow of the last command to the action block "Exit" in a program.

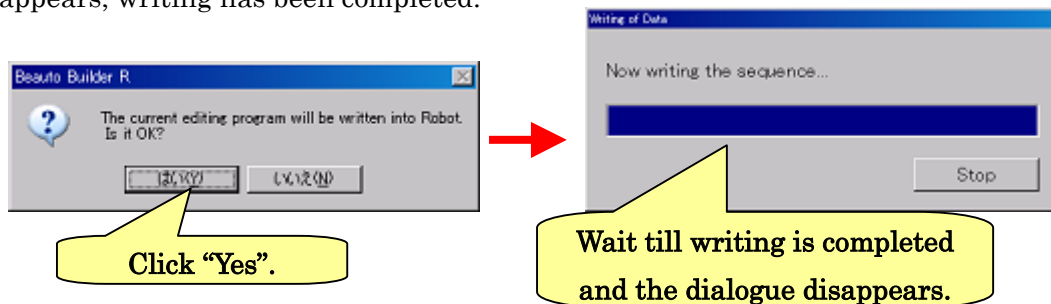
* Only to the action block “Start”, no arrow can be connected. It escapes from connection of an arrow.

2-4. Writing/Execution of program

Here, let's write the created program in the robot and execute it. First, connect the robot to PC and establish communication. After making sure that PC is communicating with the robot, click  button.

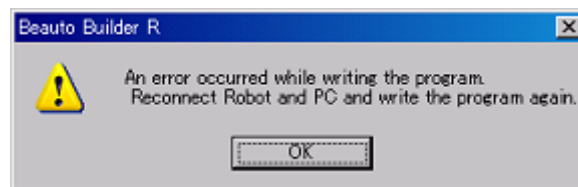


After clicking the  button, the confirmation message for execution of writing appears. Click "Yes". After click, the window showing the writing status of the program on the screen appears and writing starts. When the window showing the writing status disappears, writing has been completed.



*** During writing of a program, do not remove the robot from PC or do not touch or move the robot roughly.**

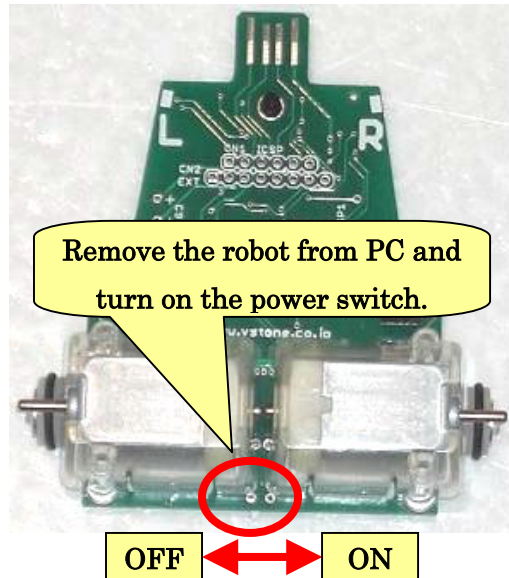
If a program cannot be normally written because of interruption of communication, etc. during writing, the following message is displayed. In this case, click "OK" to close the message and re-do writing according to the proper procedure.



One of the major causes of program writing error is "The robot was touched or moved during writing". If the robot is touched or moved during writing, communication may be interrupted, resulting in a writing error. **During writing, it is recommended to leave the robot without touching it till writing is completed.**

After completing the writing of the program, let's execute the program. By turning on the power of the robot with the robot removed from PC, the robot starts the program soon. First, remove the robot from PC. Then, place the robot on a safe place to prevent it from falling from a desk or hitting an object while it is moving. After it's ready, turn on the power switch and release it.

When the power switch is turned on, the robot moves forward for about 1.5 seconds according to the created program and stops. **After the program is executed till the end, the robot stops. If you want to start the program again, turn off the power switch and turn it on again. If you want to stop the program in midstream, turn off the power switch.**



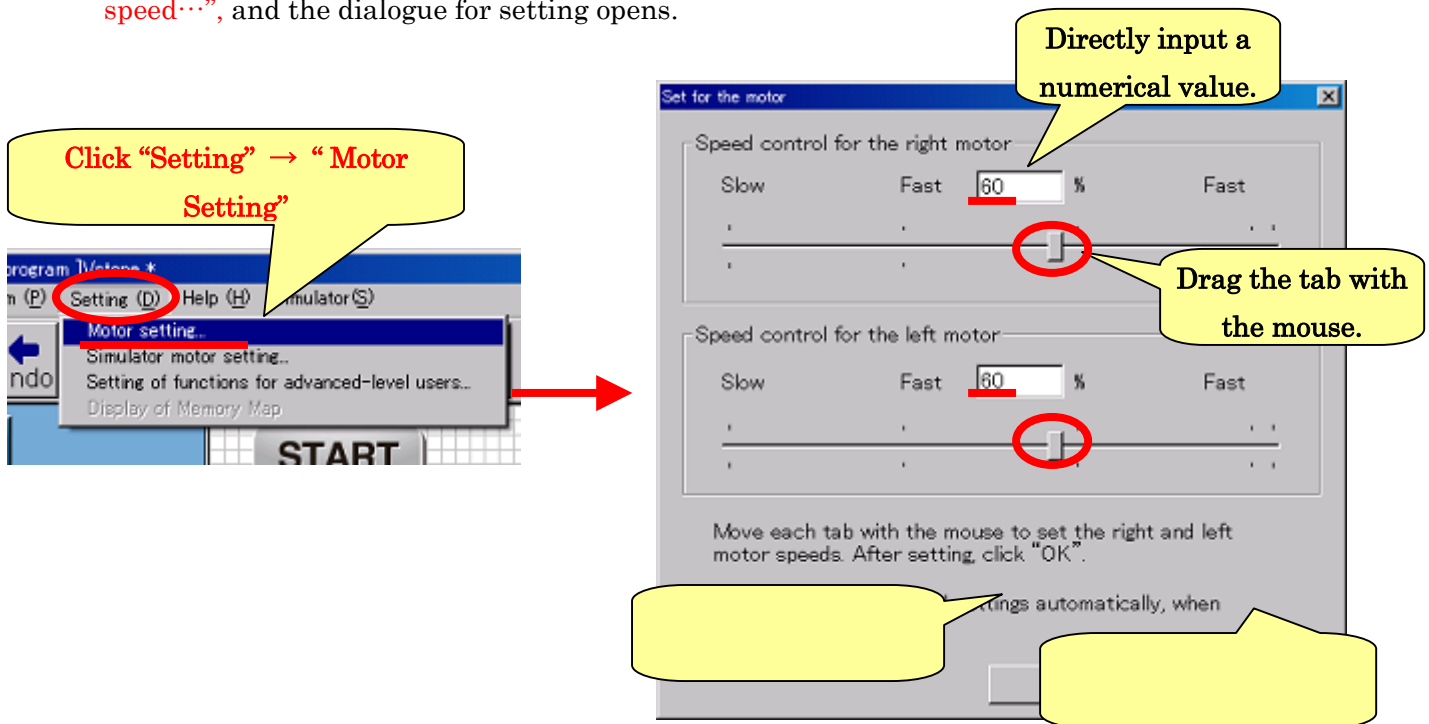
If the program is not executed normally, such as the case where one or both motors do not run, the assembly of the robot may have a problem. Re-check for the following points:

- **The motor is mounted in an inverted position.**
 - In a correct position, the brown fittings on the motor are contacted with the board of robot body. Re-assemble so that the motor is placed in a correct position.
- **The brown fittings of the motor protrude from the back holes.**
 - The fittings on the motor were raised too much during assembly. Lay down the fittings a little lower and re-assemble.
- **The battery is inserted with reversed polarity.**
 - As shown in the drawing on the board of robot body, insert the battery in a correct direction.
- **The motor does not run properly unless the motor holder is pushed in with a finger.**
 - The motor holder may not be sufficiently pressed in or the fittings on the motor may not be sufficiently raised. According to the Assembly Manual, raise the fittings on the motor up to 30 degrees or so and press the motor holder in sufficiently.

2-5. Adjustment of motor speed

When a program is executed, some users may experience such a problem that **”the robot will not move straight or it turns slightly right or left”**. This is caused by the individual difference of parts or the ground where the robot moves. In this program, ideally, the robot should run straight. Therefore, **let’s adjust the right and left motor speeds on this software to make the robot run straight.**

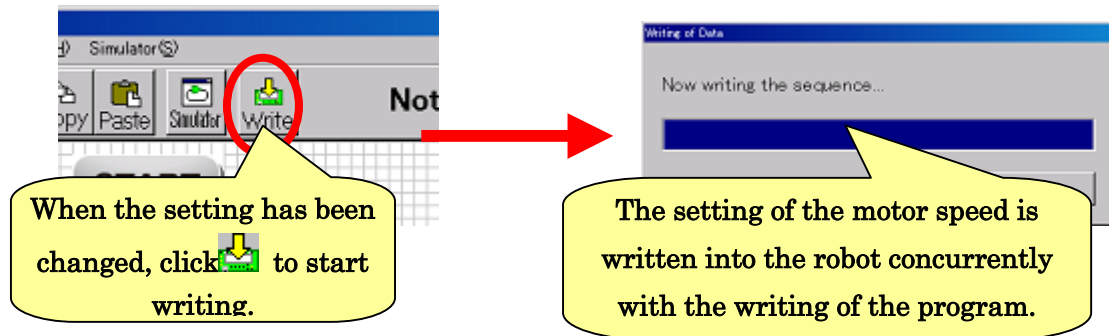
First, in the menu at the top of the window, click **”Setting”** → **”Setting of the Motor speed…”**, and the dialogue for setting opens.



The motor speed can be set individually for the right and left motors. The setting value is within 0 to 100%, and the larger the value is, the higher the motor speed becomes. The value can be set by dragging the slider tab with the mouse or directly entering the value on the keyboard. After changing the setting, click "OK" to confirm the entry. To stop the change, click "Close".

The setting of simulator robot speed is separate from this. Refer to " Simulator" for details.

To make the new motor speed take effect on the robot, the value must be written into the robot. The setting of the motor speed is written at the same time when the program is written into the robot. When the setting of the motor speed has been changed, write the program into the robot.

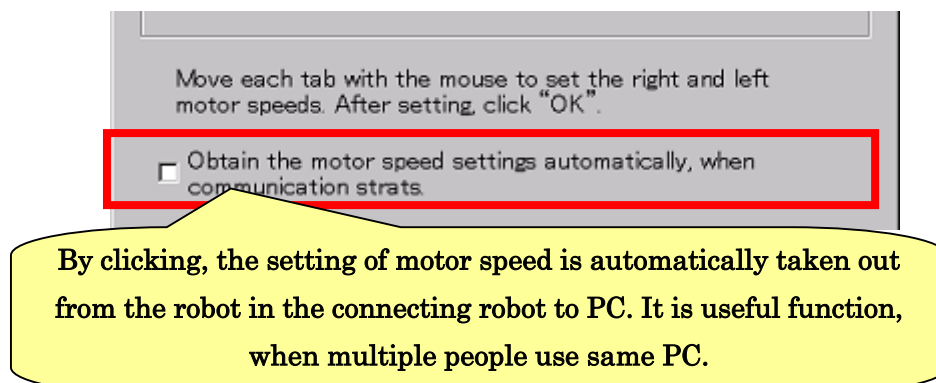


After completion of writing, re-execute the program and make sure that the motor speed has been adjusted properly. If it has not been adjusted properly, re-adjust it on this software and write it into the robot.

As explained at the beginning, **the motor speed changes depending on the individual difference of parts or the ground where the robot moves.** Therefore, if any part is replaced or the robot is operated in a different place, a deviation may occur again. In addition, **if the motor speed is too high or low, the robot may not move properly according to the program. In such cases, re-adjust the motor speed.**

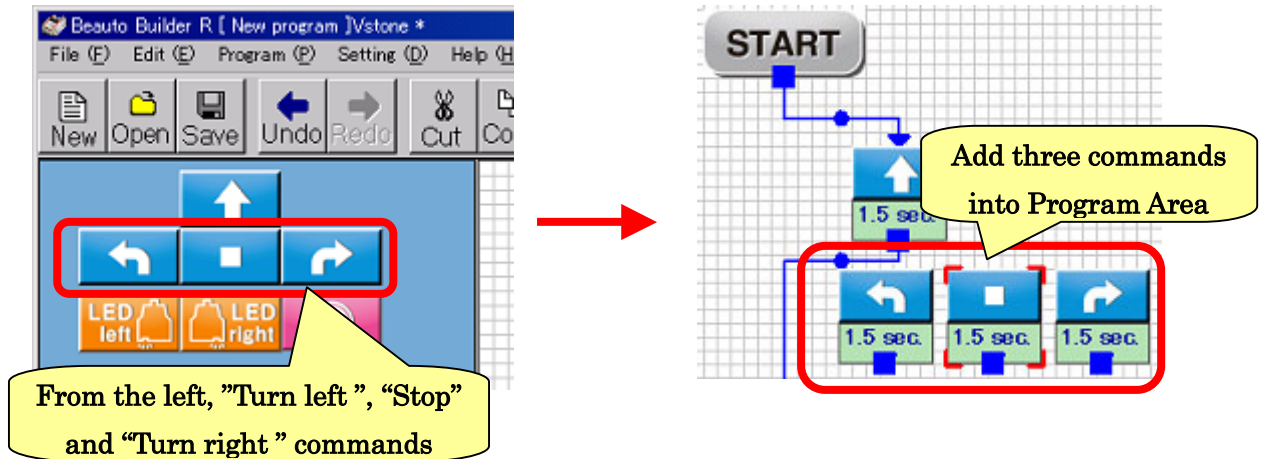
In the case where the robot moves straight as it is, adjustment is not needed. However, for the purpose of learning the adjustment procedure, it is recommended to change the setting of motor speed and check to see if the motion of the robot changes.

By clicking "Obtaining the motor speed setting automatically, when starting connection.", the setting of motor speed is automatically taken out from the robot in the connecting robot to PC.

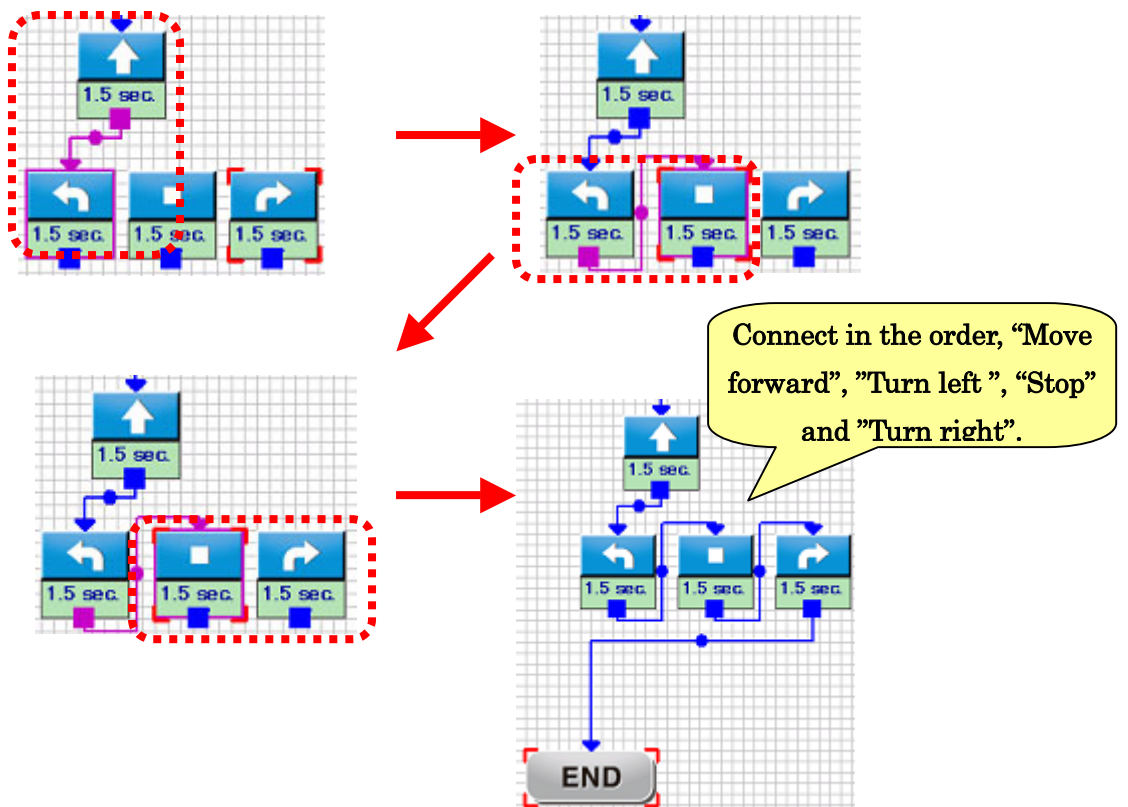


2-6. Addition of other action blocks

Here, let's create a program including multiple commands. Add the commands "Stop", "Turn right" and "Turn left" to the above program.

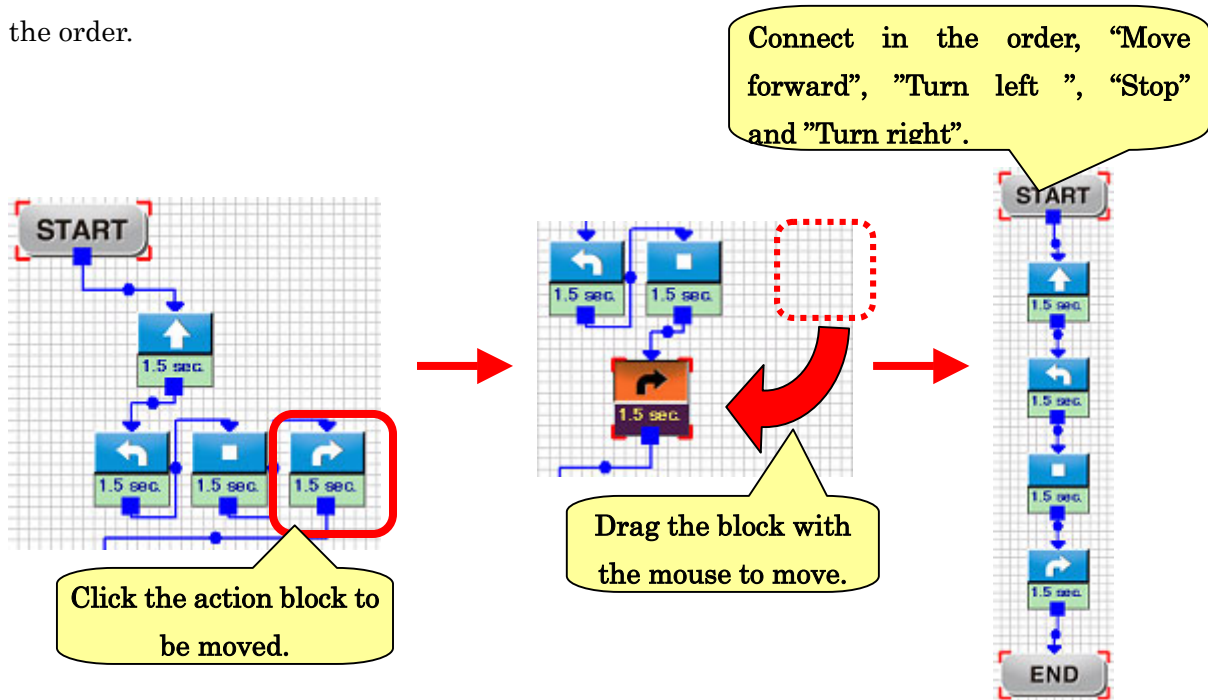


After adding the commands into the Program Area, the arrows of action blocks must be connected in the same way as previously to create a program. Now, connect the commands in the order of "Move forward" → "Turn left" → "Stop" → "Turn right".

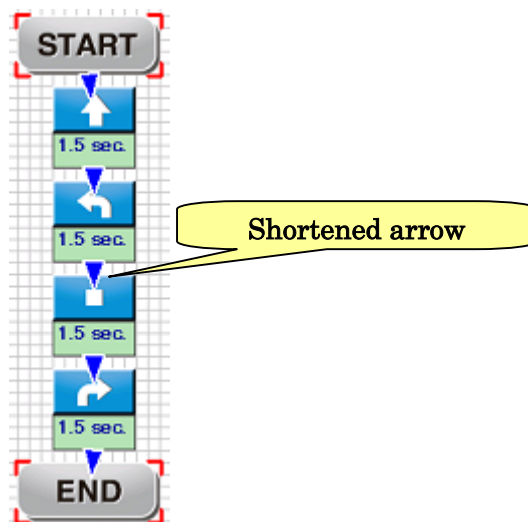


After creating the program, write it into the robot and execute it. The robot will be operated in the order of " Move forward" →" Turn left " →" Stop" →" Turn right "

When multiple action blocks are added in the Program Area, the program becomes hard to read. **Move the action blocks to appropriate positions by dragging them with the mouse so that the program becomes easy to read.** In a general flowchart, commands are arranged vertically along the flow of a program. Let's arrange the blocks according to the order.



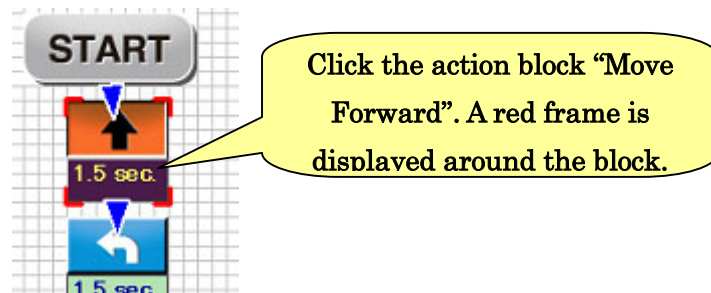
When the action blocks connected by arrows are put close to each other, the arrows will be shortened as shown in the figure below.



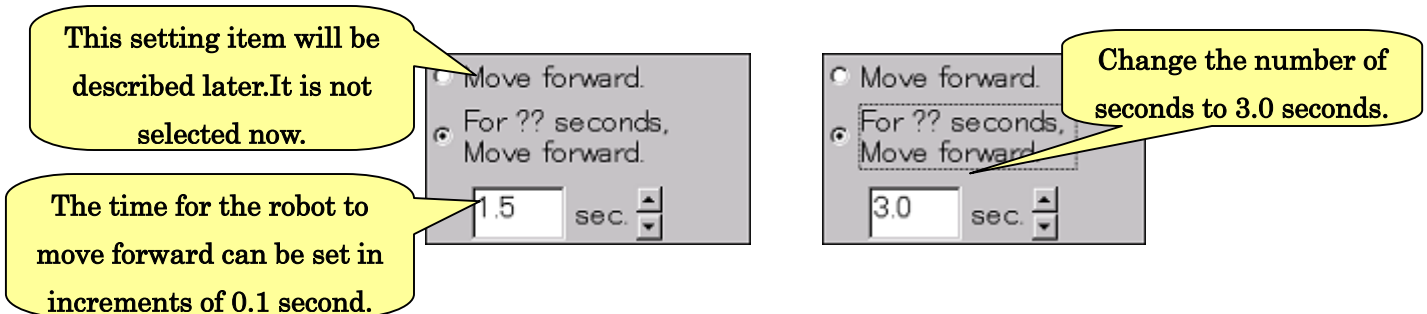
2-7. Setting of details of command

In creation of the program so far, the commands selected from the Icon Area were incorporated into the program as they were. **For each command, more detailed setting such as "Change the time of the wheel motion" can be made.** Now, let's change the settings of the commands added to the program individually.

First, click the action block "Move Forward" with the mouse. **The color of the block will change and at the same time, a red frame will be displayed around the block.**



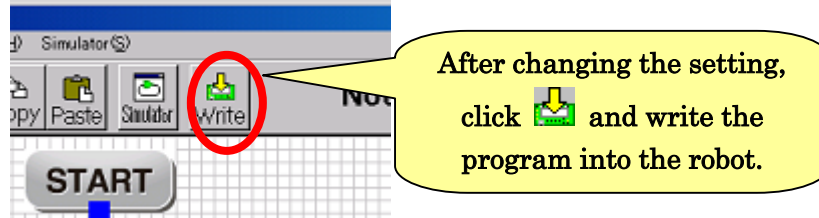
Next, check "Setting Area" in the lower left part of the window. The Setting Area is for changing the details of setting for a command. Click the action block "Move Forward", and the content of this Area changes as shown in the figure below.



The value "1.5 sec." displayed here **indicates how many seconds the robot is to move forward.** By changing this value, the robot is allowed to move forward for a longer (or shorter) time than the time set in the current program. The number of seconds can be set within 0.0 to 25.5 seconds in increments of 0.1 second. The value may be set by entering on the keyboard or clicking the spin button on the right side. Now, change the first set value "1.5 sec." to "3.0 sec.".

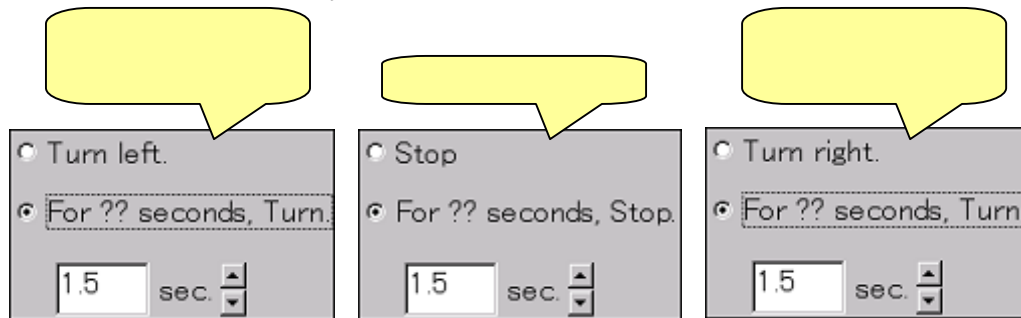
Concerning the item "Move forward", how to deal with it is difficult, and the explanation is not be made here. For detail, see the "[Creation of a program using the sensors](#)" described later.

After changing the value, re-write the program into the robot and execute it. Check to see if the time to move forward became longer.

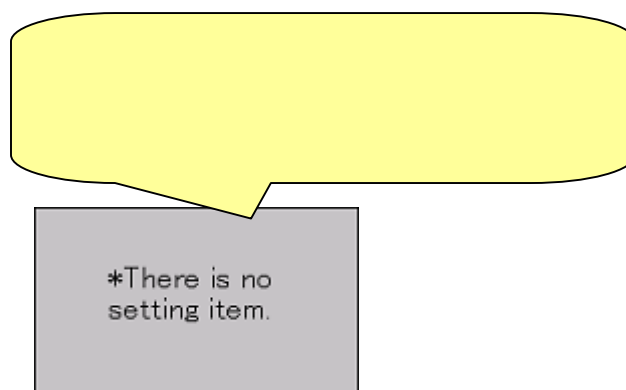


After executing the program and making sure that the time to "Move forward" has been changed, click the other commands such as "Turn right" and "Stop" in Program Area and check the respective content in the Setting Area.

For the commands "Turn right", "Turn left" and "Stop", the time for the motor to run (or stop) can be set in the same way as for the above-mentioned command "Move forward".



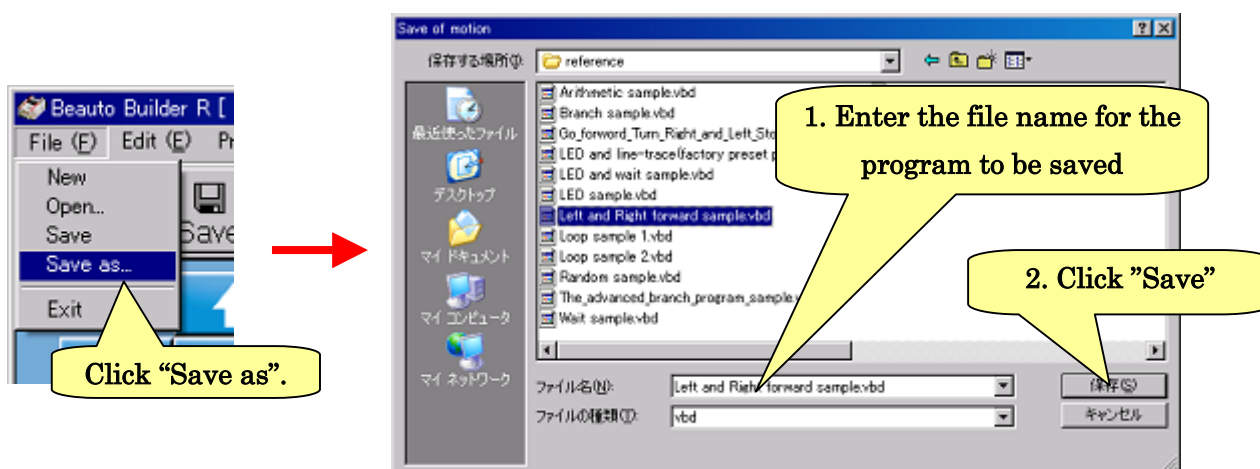
As for "Start" and "Exit", there is no setting item that can be changed. Therefore, the message as shown in the figure below is displayed in the Setting Area.



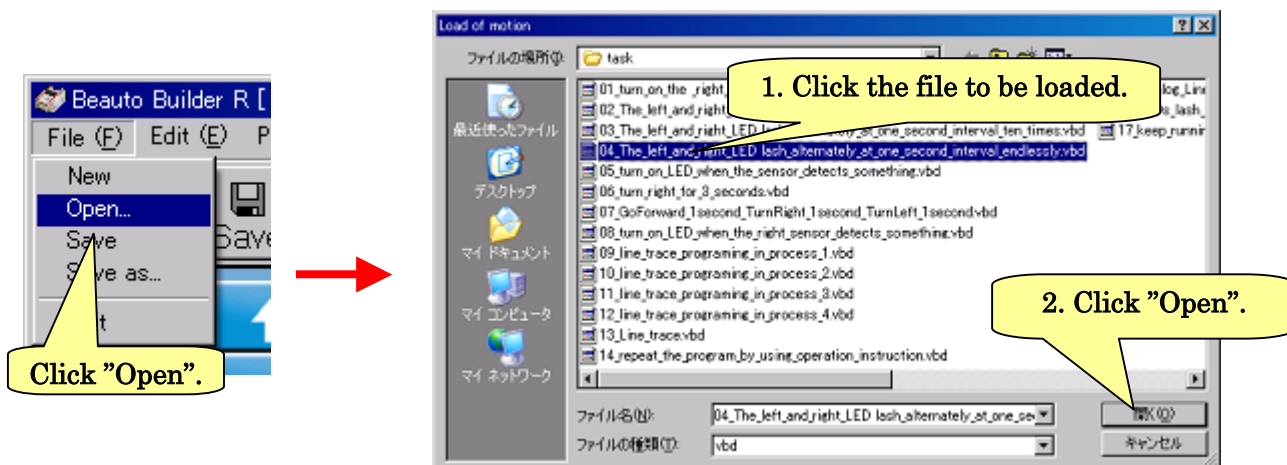
2-8. Save/Load of program



A program created with this software can be saved in a file and loaded from a file. To save a program in a file, follow the procedure below.

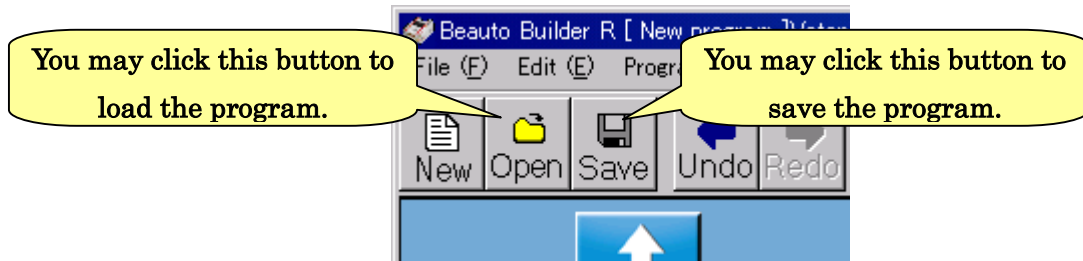
To save a program in a file, click "File" → "Save as" on the menu. The window as shown in the right figure below will appear. Enter the file name for the program and click "Save".



To load a program from a file, click "File" → "Open" on the menu. The window as shown in the right figure below will appear. Enter the file name for the program to be loaded and click "Open".

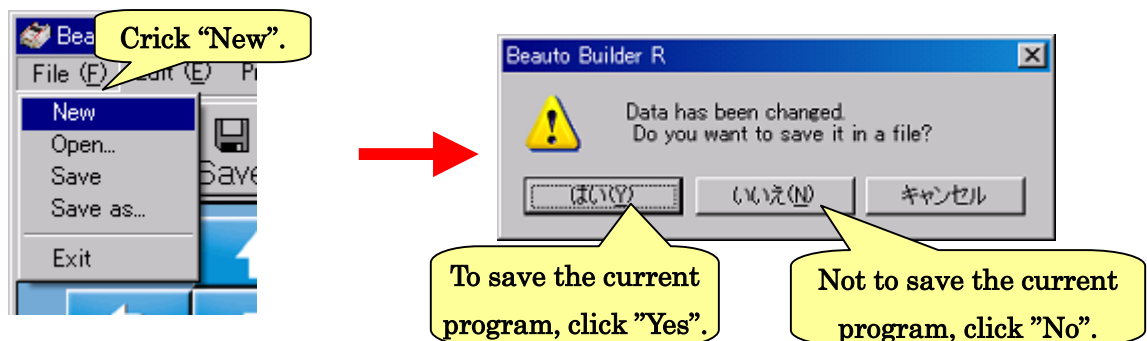



By clicking the  button on the toolbar at the top of the window, a file can be saved. By clicking the  button, the program can be loaded from the file.



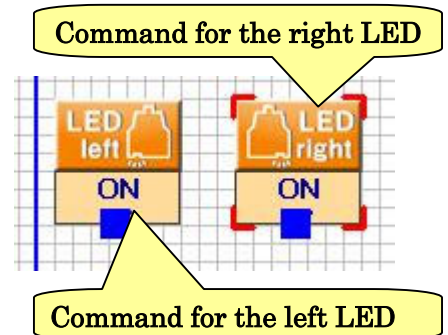
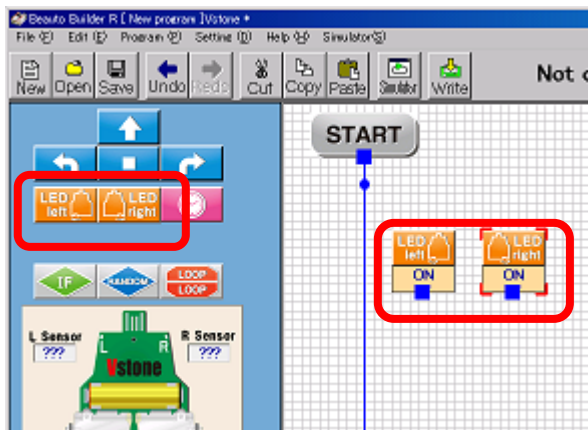
2-9. LED/Wait command

Here, let's use other commands than the commands for moving the motor. First of all, the program which is currently being created must be deleted and a new program will be created. Click "File" → "New creation" on the menu. At this time, if the program which is currently being created has not been saved in a file, the message in the right figure below is displayed. When you want to save the current program in a file, click "Yes", while when you don't want to save it, click "No".



To start creation of a new program, you may click the  button on the tool bar.

After completion of creation of a new program, add the two commands as shown in the figure below from the Icon Area to the Program Area. These are the commands for operating the LEDs on the robot body. "LED left" is for operating the left LED and "LED right" is for the right LED.



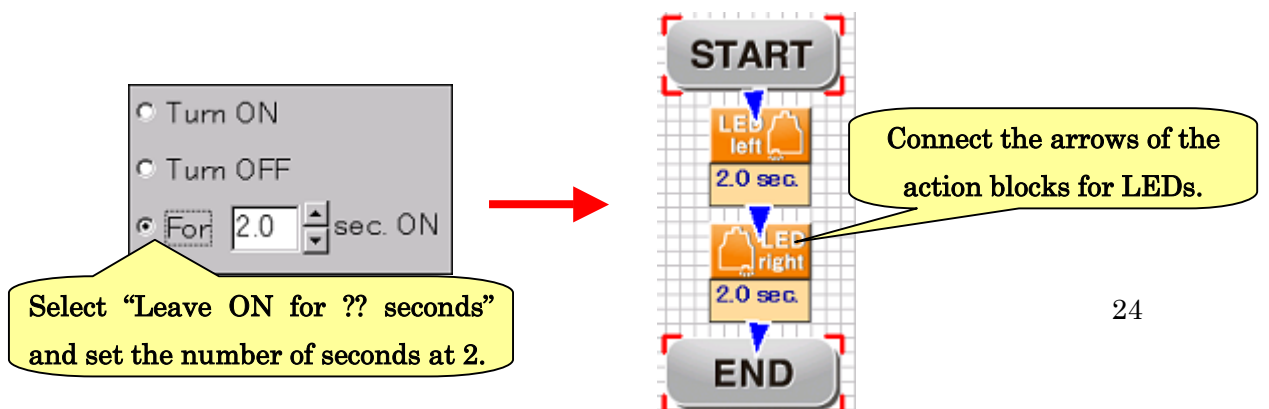
In the Setting Area, the LED command is displayed as shown in the figure below. In the Setting Area, there are three option items for setting of LED turning on or off. "Turn ON" is for turning on the LED, and "Turn OFF" is for turning off the LED. "Leave ON for ?? seconds" is for leaving the LED ON for the designated seconds and then turning off the LED.

The LED is turned ON. Proceed to the next command while leaving the LED ON.

The LED is turned OFF. Proceed to the next command while leaving the LED OFF.

Leave the LED ON for the designated seconds. After a lapse of the designated time, the LED is turned off and proceed to the next command.

Now, let's create a program in which the right and left LEDs are lighted individually for 2 seconds. For the two LED commands, change the current setting "Turn ON" to "Leave ON for ?? seconds", and set the number of seconds at 2. After completion of change in the Setting Area, connect the action blocks for the two LEDs in the Program Area.

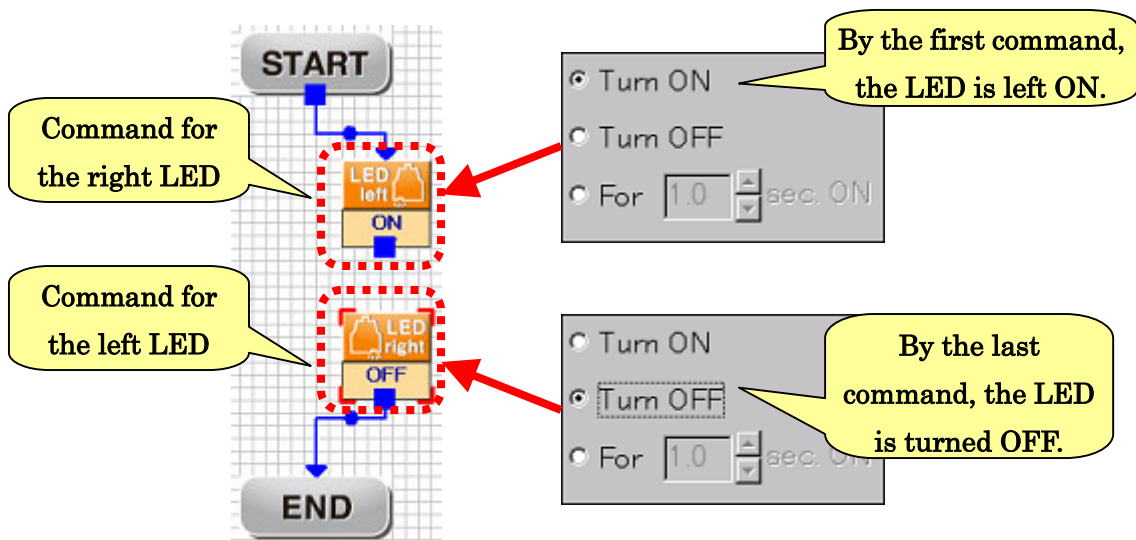


After completion of creation of the program, write it into the robot and execute it. During execution, turn over the robot and check to see if the LEDs operate properly.



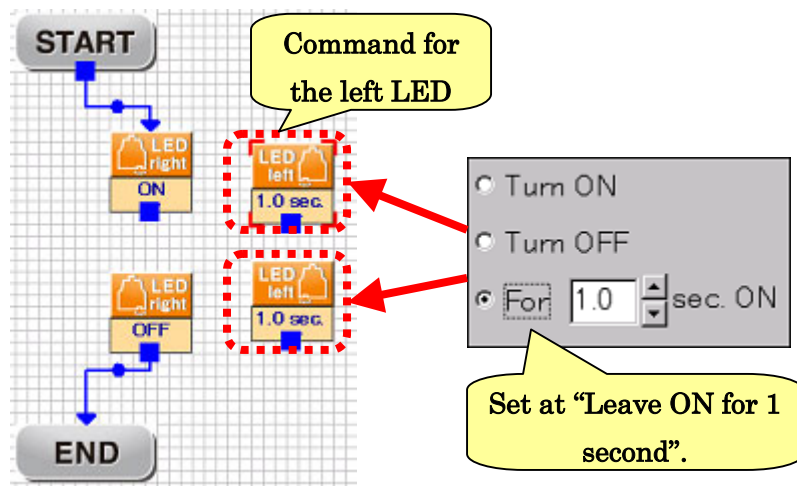
Next, let's create a program in which "The right LED is left ON and the left LED is left flashing at intervals of 1 second." Seemingly, this program is difficult, but you can create it easily if you understand the characteristics of LED commands.

As explained above about the Setting Area, "Turn ON" and "Turn OFF" mean to proceed to the next command "leaving the LED ON" and "leaving the LED OFF", respectively. By making good use of the commands, both LEDs can be turned on or either the right or left LED can be lighted up individually. Now, let's make the part of program for "the right LED is left ON after the program starts and the right LED is turned OFF when the program ends."

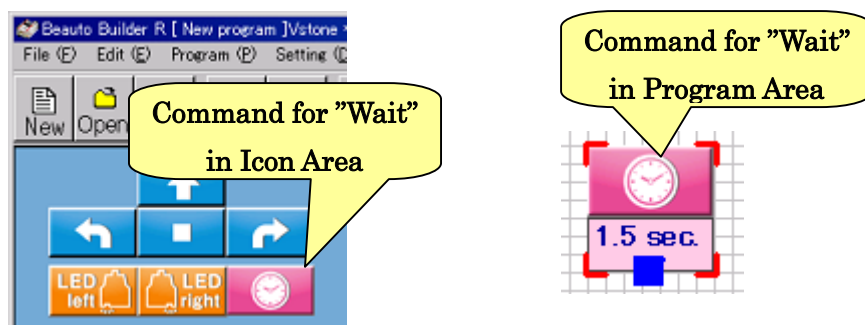


When this program is executed, the right LED will be left ON. (If a command having no arrow connected to any block is executed, the program will end in the same way as the case where "Exit" is clicked.)

Next, let's make the part of program for "the left LED is left flashing at intervals of 1 second." The part of "flashing at intervals of 1 second" is analyzed as repetition of two commands "Flash ON for 1 second" and "Flash OFF for 1 second". The command "Leave ON for 1 second" can be realized by the command "Flash ON for 1.0 second" just as the above created program. Now, add two commands for the left LED to the Program Area and set "Leave ON for 1 second" for both of them.

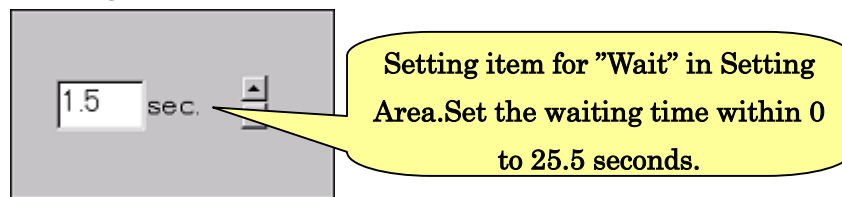


How can the command "Flash OFF for 1 second" be realized? If a setting of a command "Leave OFF for ?? seconds" could be exist for LED, it would be solved easily. Unfortunately, however, such a setting does not exist. Then, you can use a new command "Wait". The command for Wait is displayed in the Icon Area and the Program Area as shown below.

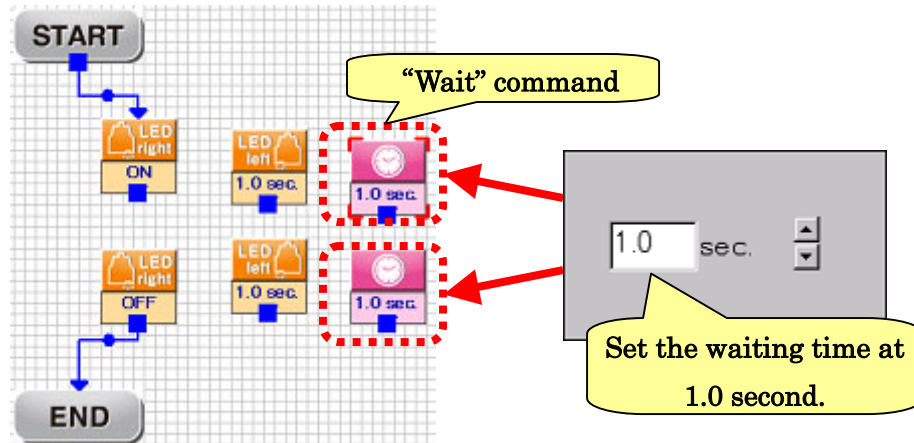


Wait is a command for "Wait for the next command while keeping the robot in the current state." When in the state where the motor of robot is running or the LED is lighted up, for example, Wait is executed, the robot will wait for the designated time keeping the motor or LED in the current state. It means that the state of "Leave OFF for 1 second" can be produced by executing Wait for one second in the state where "the right LED is ON and the left LED is OFF".

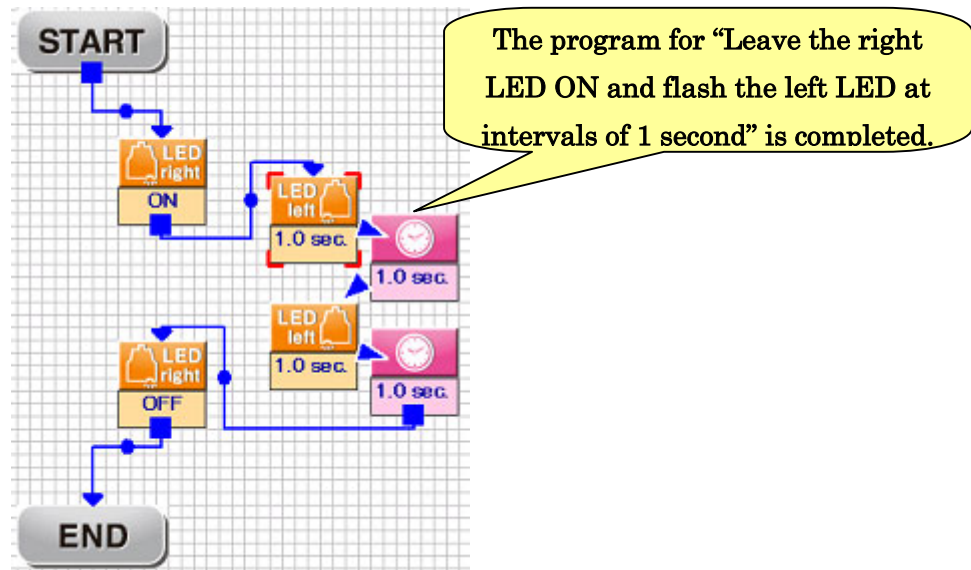
The setting item for the Wait command is only a waiting time. In the Setting Area, it is displayed as shown in the figure below, and the waiting time shall be set within 0 to 25.5 seconds. The setting method is the same as that for other commands.



Now, let's add two Wait commands to the Program Area and set the waiting time for both commands at 1.0 second.



Now, all commands required for the program are prepared. The arrows of action blocks must be connected. Let's connect the arrows as shown in the figure below, complete the program and then write it to the robot for execution.



According to the explanation above, you will be able to create a program for moving the motors and LEDs as you desire. However, according to the explanation above, only one-way program from "Start" to "Exit" can be made. In the following chapter, we will make a program for making the robot follow conditional branches using the sensors.

Before proceeding to the branch program, in order to review what you have learned so far, challenge the following exercises.

Program exercises

- Create a program in which the right LED flashes two times at intervals of 1 second and the left LED flashes three times at intervals of 1 second.
- Write a square having each side of 30 cm in length on paper, and create a program in which the robot traces around the square.

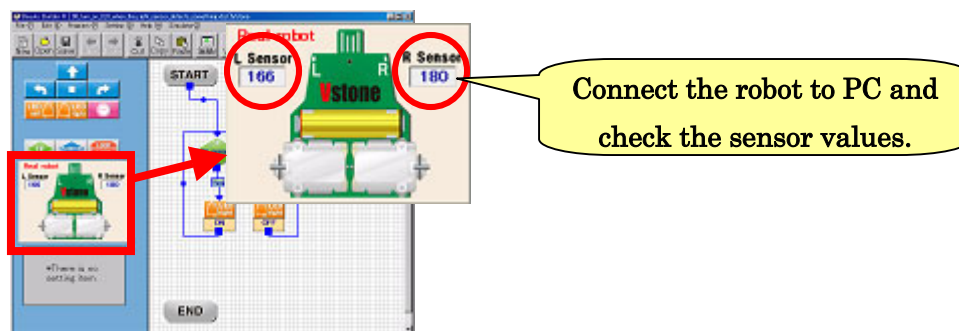
(Suggested answers are presented in "Answers to the exercises" at the end of this document.)

3. Creation of a program using the sensors

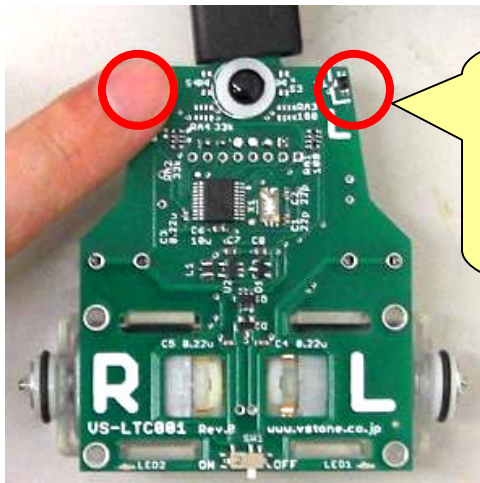
As mentioned at the beginning of this document, the robot is equipped two infrared ray sensors on the front back. **These infrared ray sensors can recognize the depth and brightness of a ground color, and a program for "Trace a black line drawn on a white ground" can be executed.** Such a motion of robot is called "**Line Trace**", which is widely adopted in exercises for robot programming learning and robot competitions. In this chapter, we will learn programming with the aim of creating a program for Line Trace with the robot.

3-1. Check of sensor reaction

Before starting programming, let's check the operation of the sensors on the robot. First, connect the robot to PC and see "Sensor Area" in the left part of the window. If you have a USB extension cable, use the cable to connect the robot. As briefly explained about installation and checking of operation in the Introduction, the current sensor values of the robot are displayed in the Sensor Area.



If a USB extension cable is used to connect the robot, **hold up the robot, cover the sensor with a finger and release the finger repeatedly.** At the same time, observe the value for the sensor in the **Sensor Area** and check to see how the values change by covering the sensor with a finger or releasing the finger from the sensor. It is expected that the sensor value decreases when the sensor is covered with a finger, while it increases when the finger is released from the sensor. Take notes on the approximate values in the respective states.




Cover the sensor with a finger and check to see if the value is changed. At this time, take notes on the change of values.

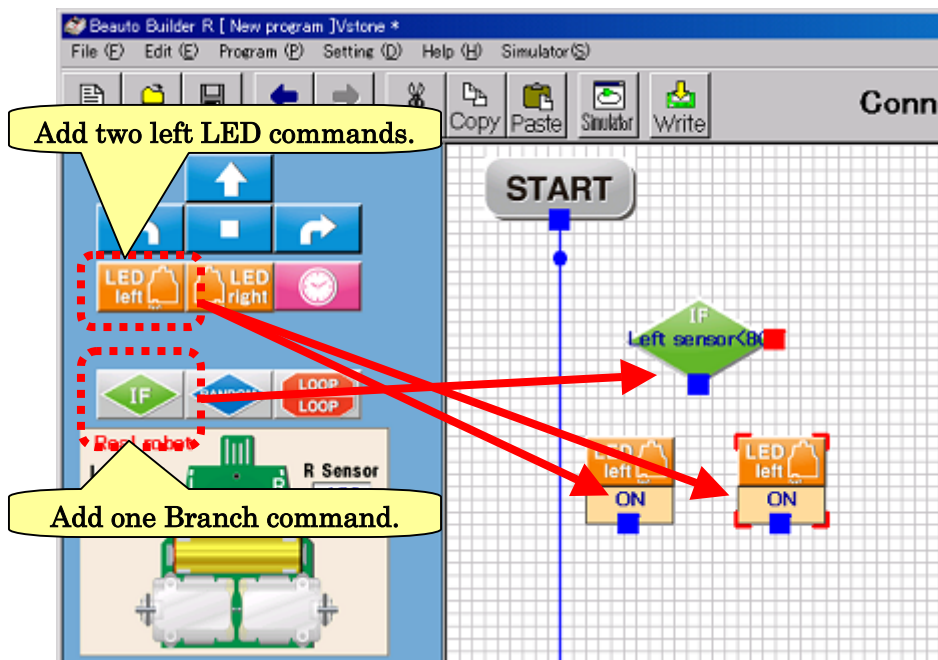
- Finger comes near the sensor: The value decreases.
- Finger gets away from the sensor: The value increases.

3-2. How to use Branch

Here, programming using branches will be explained. To begin with, as an exercise, let's create a program in which "the left LED is lighted up when the left sensor is covered with a finger and it goes off when the finger is released."

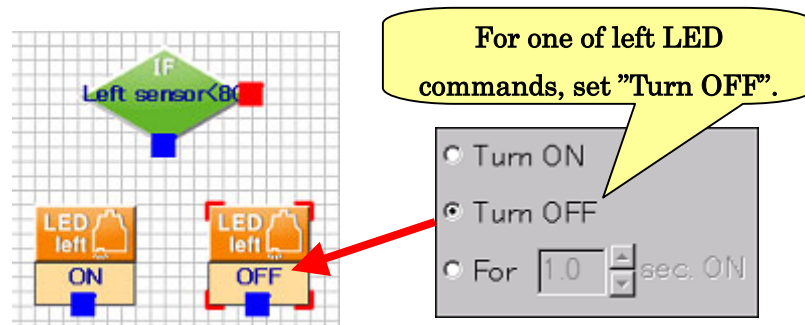
First, let's use the command for branch using the sensors. When a program is currently being created, click "File" → "New Creation" on the menu or click the  button on the tool bar to start creation of a new program.

Next, as shown in the figure below, add two "left LED" commands and one "Branch" command into the Program Area. The green rhombic block is for a "Branch" command.

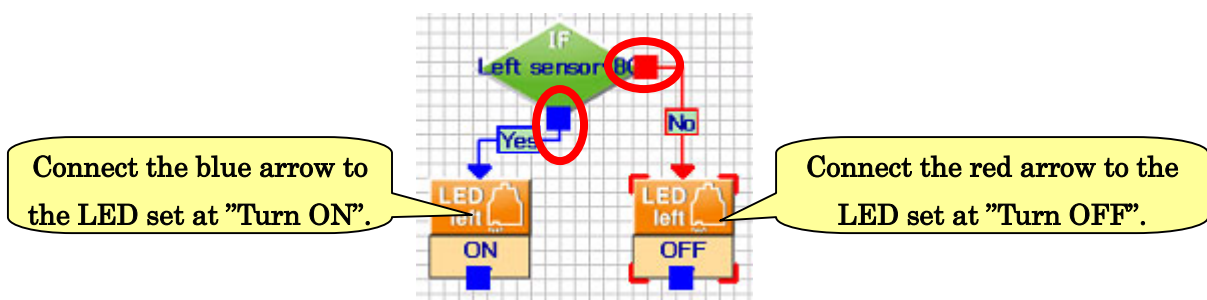


Setting of Branch command requires a long and detail explanation, and it will be explained in the next section. Here, setting of other commands and connection of arrows will be made.

Both of the left LED commands added to the Program Area are initially set at "Turn ON". In this program, the LEDs are turned on and off depending on the sensor response. Set "Turn OFF" for one of two left LEDs.

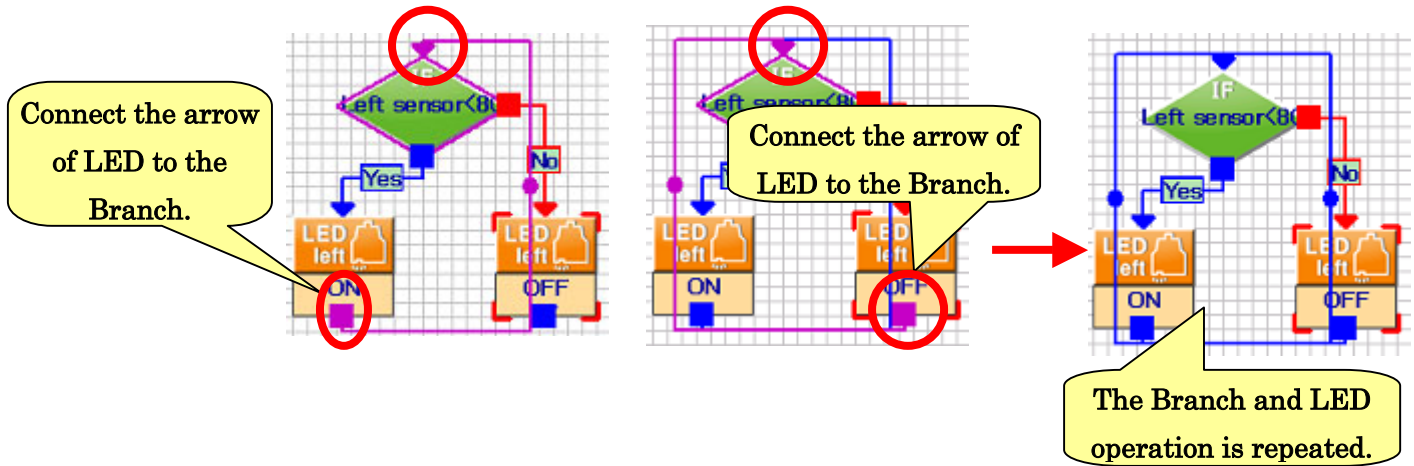


Next, connect the Branch arrow to the left LED command. All action blocks that were used in the explanation so far had each only one arrow, but a Branch block has two arrows. When a program is executed, the robot will determine which arrow should be followed depending according to the predetermined condition. Now, as shown in the figure below, connect the blue square arrow to the LED set at "Turn ON" and the red square arrow to the LED set at "Turn OFF", respectively.

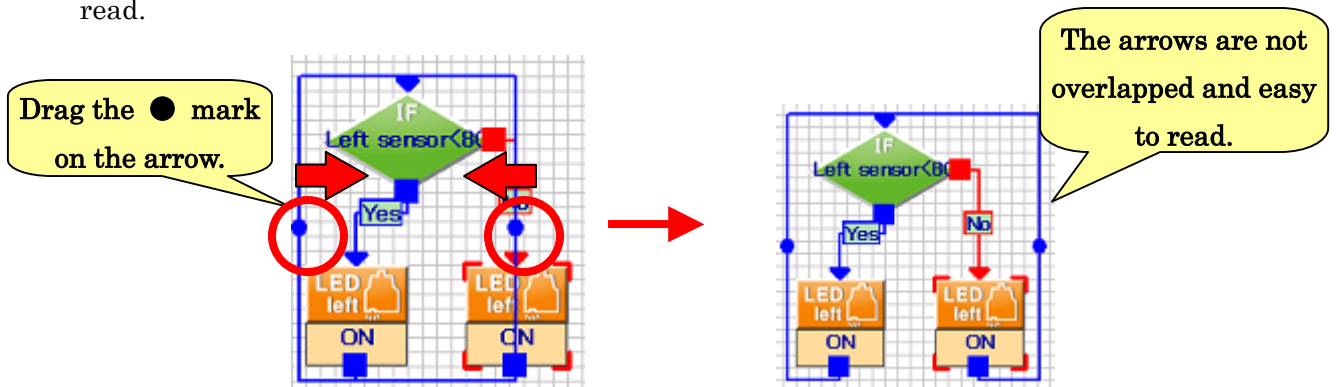


After that, the arrows of the left LEDs must be connected. In the program created in the previous chapter, the arrow was connected to "Exit". **In this program, the LEDs need to be turned on and off depending on change of the sensor status.** Therefore, **after proceeding from a branch to either command, it is necessary to return to the branch and re-check the sensor status.**

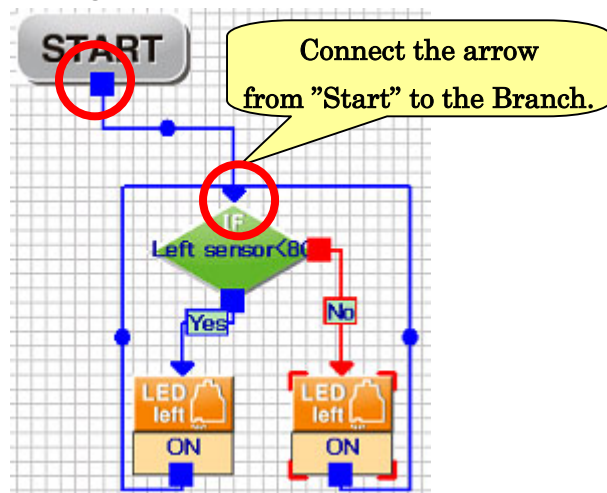
Now, connect the arrows of LED commands to the Branch commands as shown in the figure below.



When arrows are connected, they overlap each other and the program becomes very hard to read. Therefore, **drag the ● mark located at a midpoint on the arrow with the mouse to move the position of the arrow.** Thus, by moving the arrows so that they do not overlap each other as shown in the right figure below, the program becomes very easy to read.

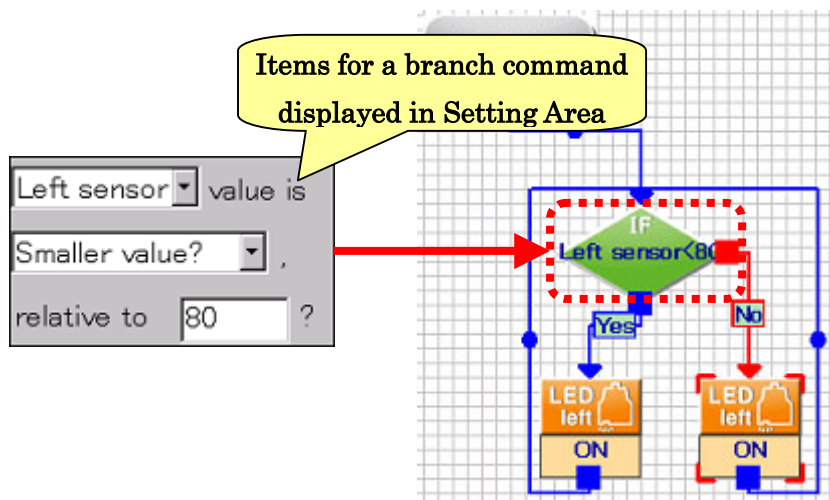


In addition, let's connect the arrow from "Start" to the Branch. Now, the arrows have been connected. Next, setting of the condition of the branch will be learned.



For the current program, **the condition that “in what state of the sensor the LED turns on or off” has not been set, which is most important.** If the condition is set, the program is completed. How to set the condition, together with how to set in the Setting Area for branch will be explained below.

First, click a command for branch in the Program Area and check the Setting Area. In the Setting Area, the items as shown below will be displayed.



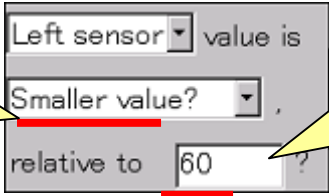
Setting items for a branch command are included in three lines. All the three items need to be set. **On the first line, select the “Sensor to be applied to the condition”.** In this program, the “Left sensor” should be selected, which has been set initially. If the right sensor is applied, select the “Right sensor”.

Select the sensor to be applied to the condition.
Since the “Left sensor” has been set initially, no selection is needed.

Left sensor value is
Smaller value?
relative to 80 ?

2 On the second and third lines, the "constant to be compared with the sensor value" and the "kind of condition" should be set, respectively. In this program, what should be input here are as follows: Set "60" on the second line and "smaller value?" on the third line, as shown in the figure below.

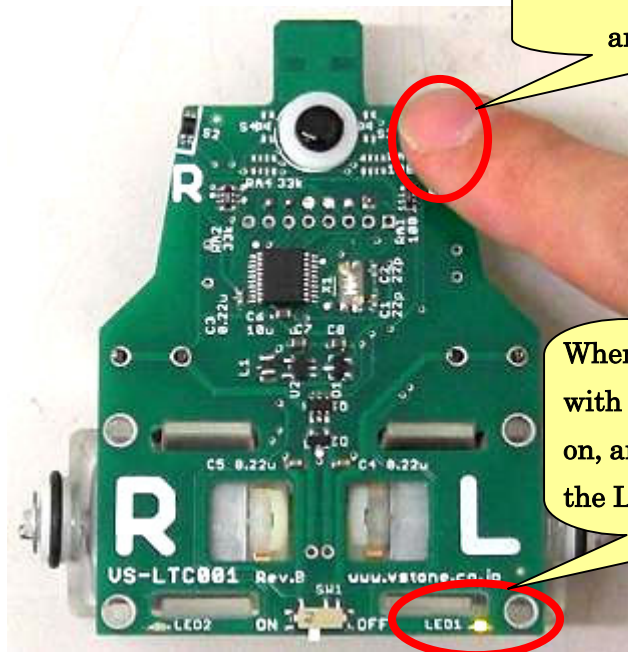
On the third line, set the kind of condition, which is "smaller value?". Leave as it is.



On the second line, input the constant to be compared with the sensor value.

Next, let's write the program with the condition into the robot and execute it to see if the condition set above will work properly. When the left sensor of the robot is covered with a finger, the left LED turns on, and when the finger is released, the LED goes off.

Cover the left sensor with a finger and release the finger.



When the left sensor is covered with a finger, the left LED turns on, and when the finger is released, the LED goes off.

From now, let's check how this program is executed in the robot sequentially. In the above-mentioned procedure, the sensor of the robot was covered with a finger and the value was observed. **The sensor value became very small when the sensor was covered with a finger, while it became large when the finger was released from the sensor.** On the assumption that the sensor value is "around 0 when the sensor is covered with a finger" and "around 160 when the finger was released", let's see the sequence of program in each case.

First, when the sensor value is around 0, " $0 \text{ (sensor value)} < 60$ " holds. Therefore, proceed to the "Yes" arrow. Ahead of the "Yes" arrow, the left LED ON command exists. In this way, the program in which "the LED turn on when the sensor is covered with a finger" is executed.

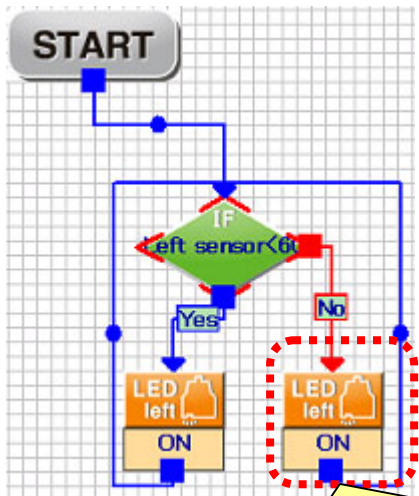
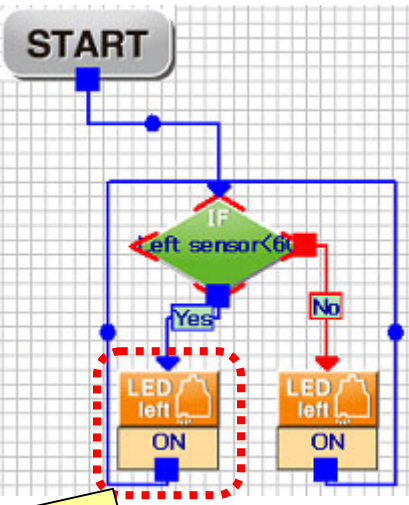
On the other hand, when the sensor value is around 160, " $160 \text{ (sensor value)} < 60$ " holds. Therefore, proceed to the "No" arrow. Ahead of the "No" arrow, the left LED OFF command exists. Thus, the program in which "the LED turn on when the sensor is covered with a finger" is executed. In this way, the program in which "the LED turn off when the finger is released from the sensor" is executed.

Furthermore, after the left LED is ON/OFF, the arrows are connected to the branch again. Therefore, the program will be repeated permanently, like reading the sensor value, proceeding to either arrow, and returning to the branch...

When the condition is "smaller value?" relative to "60":

Cover with a finger =
The sensor value is 0.

Release the finger.
= The sensor value is 160.



The condition holds. Proceed to "Yes".
The left LED turns "ON".

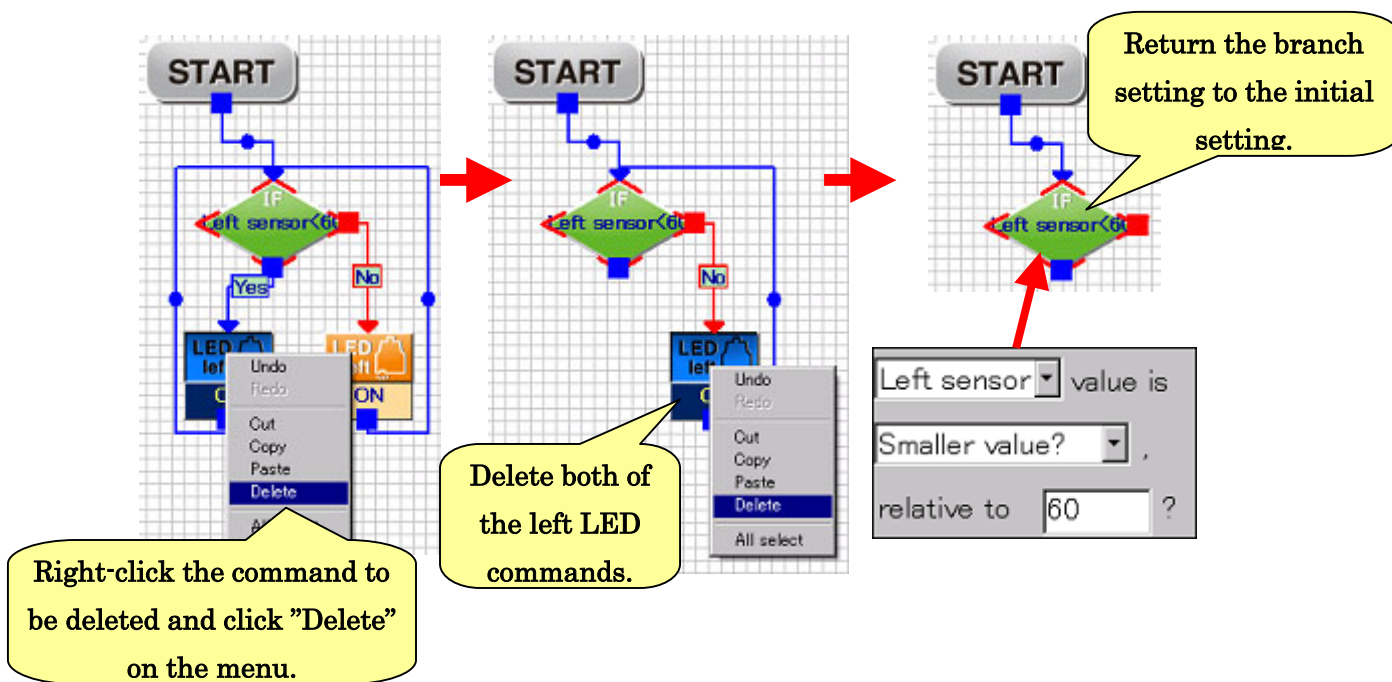
The condition does not hold. Proceed to "No".
The left LED turns "OFF".

Now, the program operation was checked and the mechanism was understood. Do the following exercises to check to see how the program works:

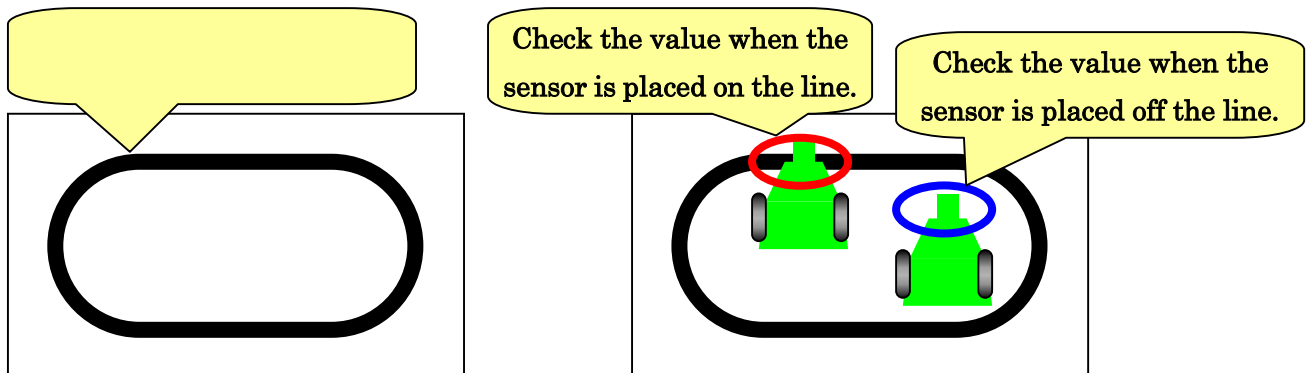
- Check to see how it works when the branch condition is changed to "larger value?".
- Check to see how it works when the arrows of "Yes" and "No" are connected in reverse.

3-3. Programming for Line Trace

Let's make a program for Line Trace using the program created so far. **First, delete both of the left LED commands. Place the mouse cursor on the command to be deleted and right-click.** The menu appears. Click "Delete" on the menu. Or, to delete a command, you may click the command to be deleted and press the Delete key on the keyboard. **Return the branch command setting to "the left sensor is, relative to 60, smaller value?"**



Place the course for Line Trace included in the product as shown in the figure below. After placing the course, when a USB extension cable is used for connection, **check the sensor values "when the robot sensor is placed directly on the line" and "when the robot sensor is placed off the line", respectively, in the same manner that the sensor values were initially checked.**



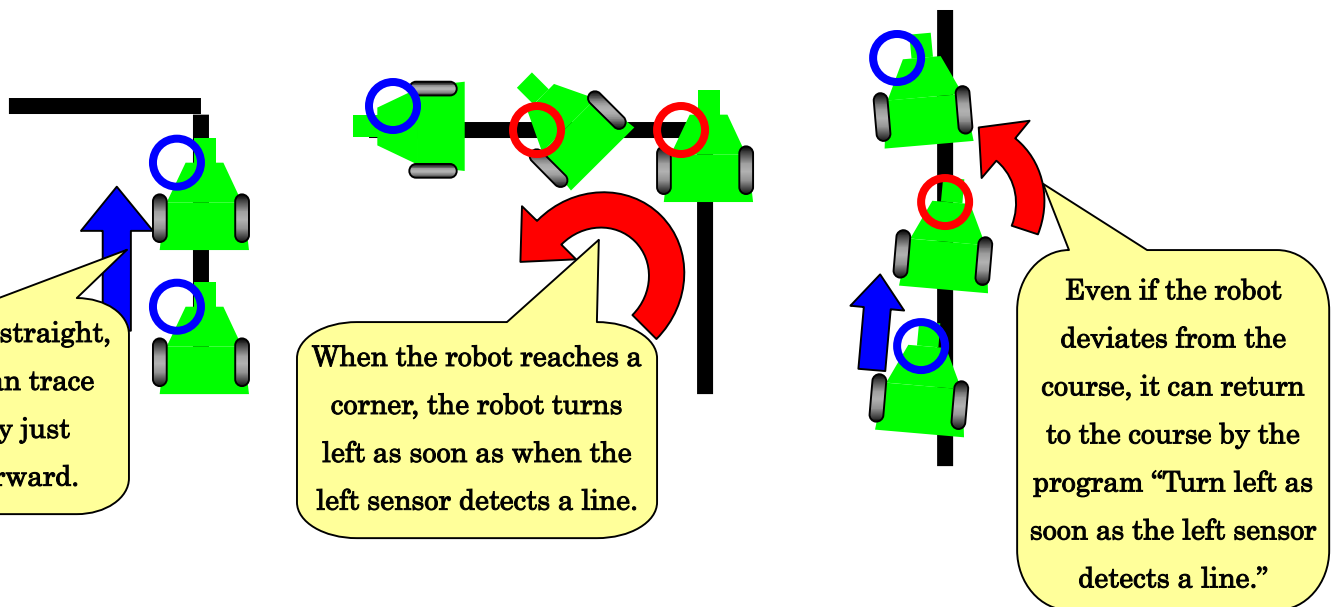
- Check to see if the sensor values become as follows:
- Sensor value when the line cannot be detected: Smaller
 - Sensor value when the line can be detected: Larger

Instead of the deleted left LED commands, let's add commands for the motor. There are four kinds of commands for the motor to operate. Which commands should be selected for this program? The principle of Line Trace will be explained below, and let's think about it.

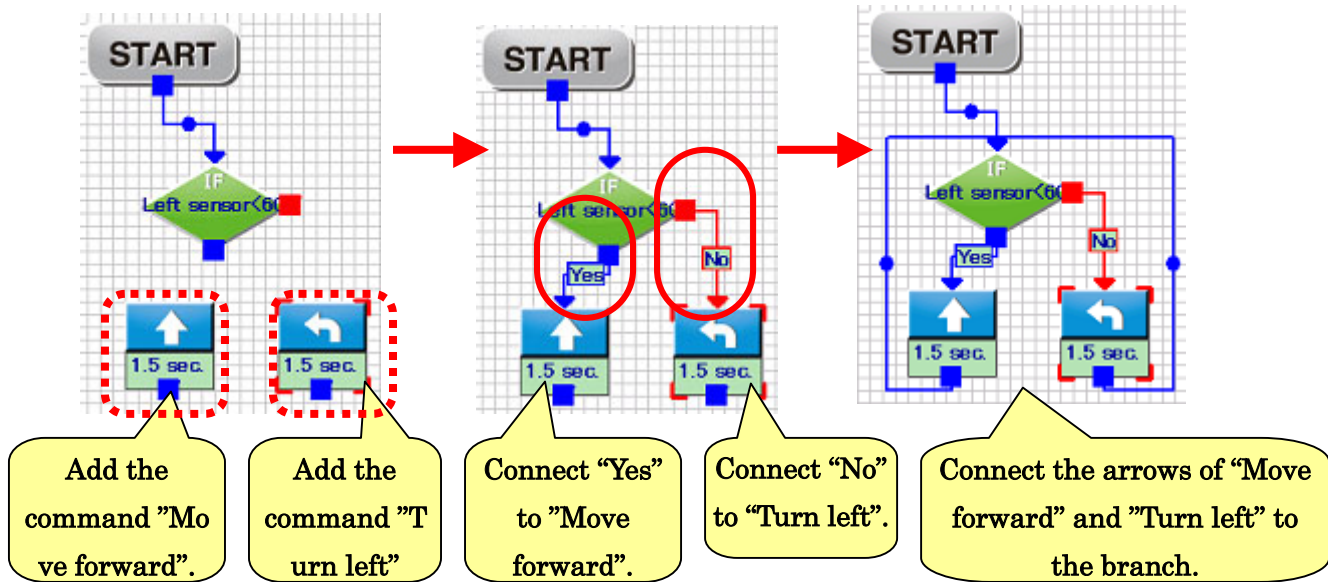
Line Trace is based on the premise that "the robot starts in the state where it is straddling over the line." In this state, "neither the right sensor nor the left sensor is detecting the line." If the line is straight as shown in the left figure below, you will see that the robot can follow the line just by moving straight.

How about if the robot reaches a corner of the course as shown in the center figure below? If the robot keeps going straight, it will deviate from the line. However, it never passes on the corner but the left sensor detects the line. Accordingly, by making a program "Turn left when the left sensor detects a line," the robot can move without deviating from the line.

As shown in the right figure below, if the robot turns slightly at the time of start, the robot will deviate from the line when it keeps going straight. Then, by making a program "Turn left as soon as the left sensor detects a line," the robot can return to the line.



Now, as explained above, add the commands "Move forward" and "Turn left" to the Program Area. Then, connect them as shown in the figures below.

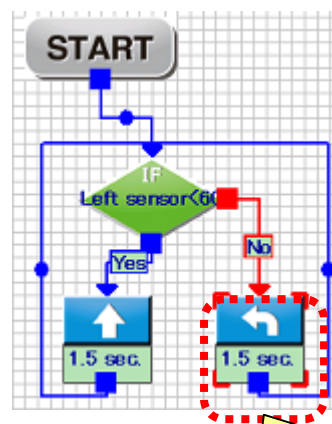
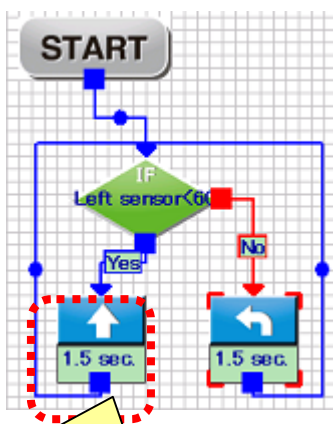


Let's check if the program created so far is correct before actually moving the robot. In the same manner as the previous program, check how the robot moves "when the line can be detected" and "when the line cannot be detected", respectively. If no problem is found, let's write the program into the robot and actually move the robot. Start moving "counterclockwise" on a course.

When the condition is, relative to "60", "smaller value?":

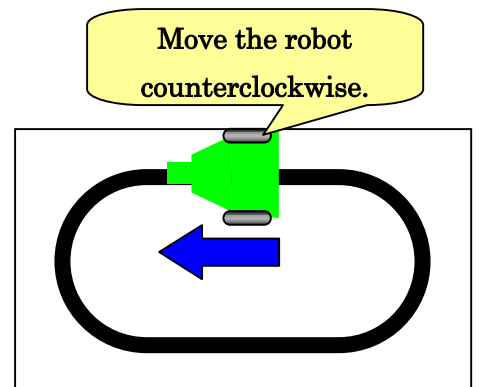
The line cannot be detected=
The sensor value is 10.

The line can be detected=
The sensor value is 100.



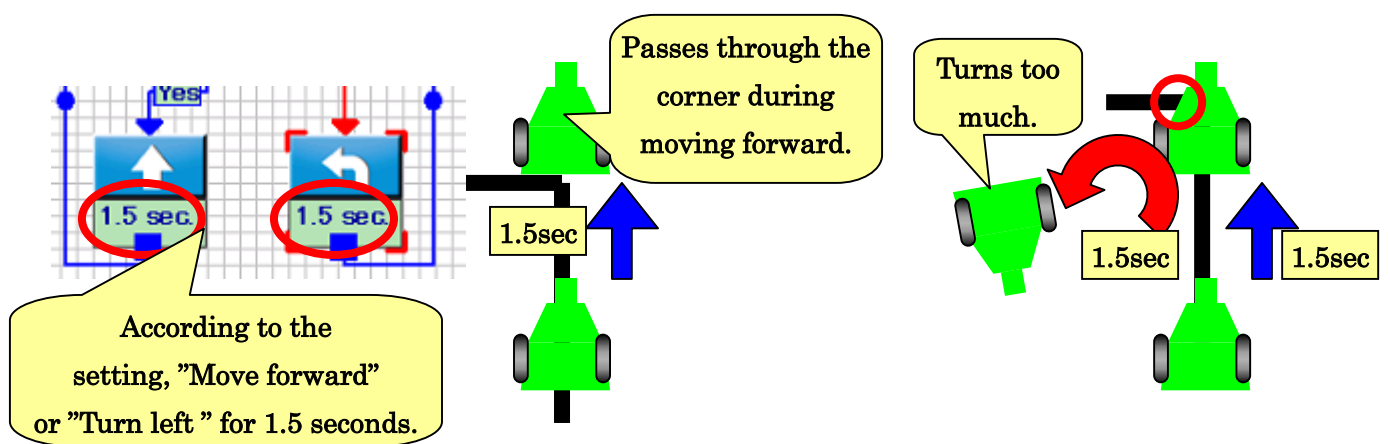
The condition is satisfied.
Proceed to "Yes".
The robot "Moves forward".

The condition is not satisfied. Proceed to "No".
The robot "Turns left".



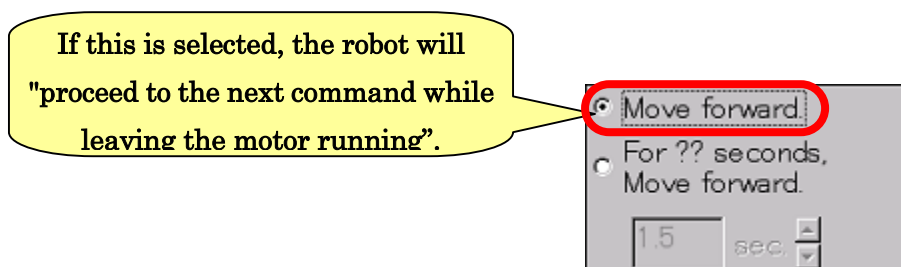
After the program is executed, the robot deviates from the line or turns too much and cannot do the Line Trace well. The flow of program is correct, but why doesn't it operate properly? In this case, there is a problem in the command setting for the motor.

The command for the motor explained so far is "Operate for the number of seconds set in the Setting Area". Looking over the created program again, **it is found that "Operate for 1.5 seconds" is set for both "Move forward" and "Turn left"**. In this program, **the motor operates for 1.5 seconds every time the sensor detects a line**. As a result, **the sensor overlooks a curve as shown in the center figure below or the robot turns left for 1.5 seconds after the line is detected as shown in the right figure below.**

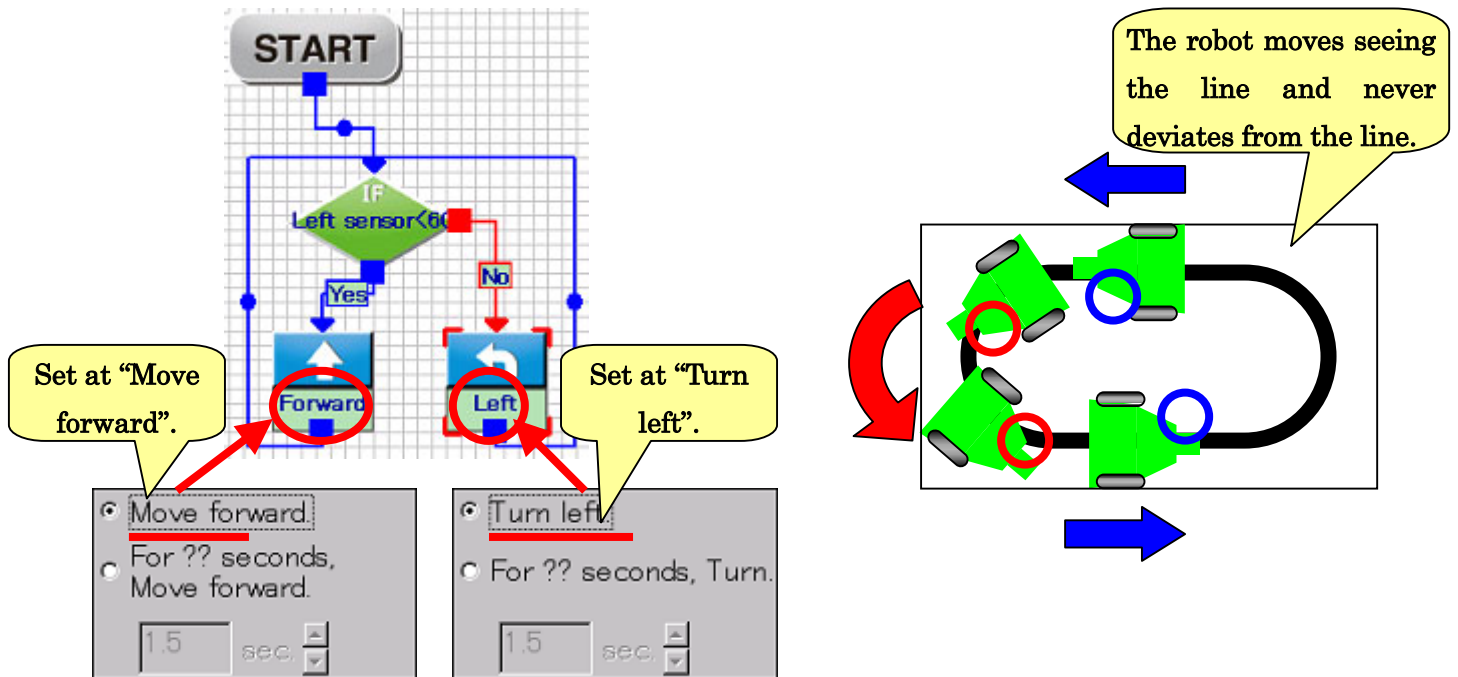


To solve the problem, we could take a measure of "shortening the setting time of 1.5 seconds", **but when the shortest time "0.0 seconds" is set, the robot will not move**. This is because **"Move forward or turn for 0.0 seconds" means "Do not move forward or do not turn"**. Then, when "0.1 seconds" is set, the motor will move but the sensor will be overlooked for 0.1 seconds.

The best solution is to **"check the sensor while running the motor"** without setting the time. For the setting procedure of this, we already learned about a similar procedure when learning the command settings for LED. At that time, when the commands "Turn ON" and "Turn OFF" for LED were selected, the robot could proceed to the next command with "leaving the LED ON" or "leaving the LED OFF". Similarly, for the motor, **there are commands to proceed to the next command while "leaving the motor running" and "leaving the motor off"**.



Change the commands Move forward for “1.5 seconds” and Turn left for “1.5 seconds” to Move “Forward” and Turn “Left” as shown in the left figure below. After change, write the program into the robot and execute it. Check to see if the robot can do Line Trace properly.



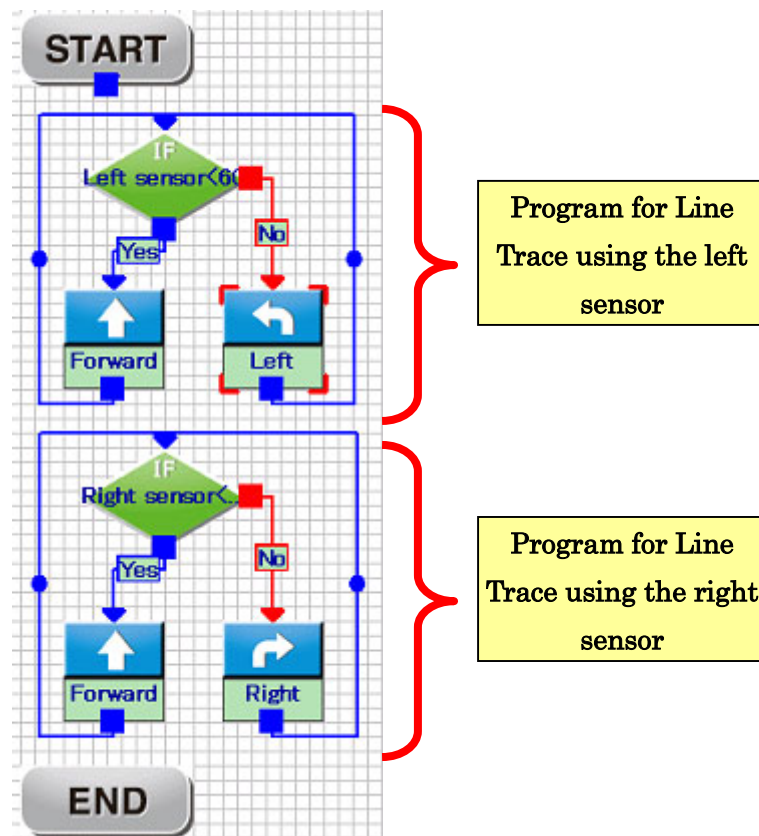
Now, the program operates properly. Then, let's make a program to do Line Trace using the right sensor. The program can be made by partly modifying the program for the left sensor. There are three hints: "To use the right sensor", "To reverse the turning direction" and "To run the robot clockwise on the course". (Suggested answer is presented in "Answers to the exercises" at the end of this document.)

In a case where the robot deviates from the line after the settings of the program are corrected, the motor speeds may be too high. In this case, reduce the speeds for both motors by the same value.

3-4. Line Trace using the right and left sensors

In the programs which were created so far, either the left sensor or the right sensor was used. Therefore, the robot moved properly only either clockwise or counterclockwise. By making a program in which both the right and left sensors are used at the same time, the robot can do the Line Trace moving in either direction on a course. Now, let's make a program in which both sensors are used.

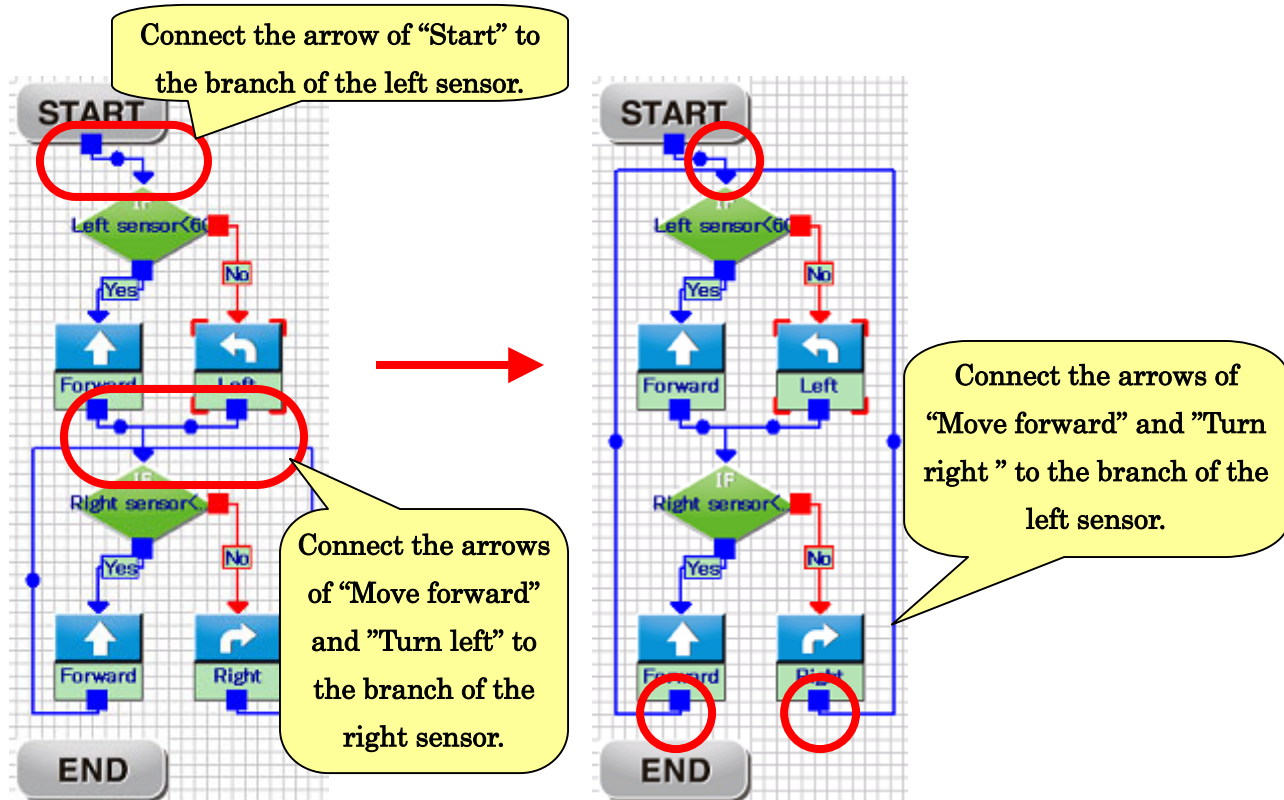
First, create the programs for Line Trace using the right and left sensors, respectively. The program for Line Trace using the left sensor is the same as the program made previously. To make a program for Line Trace using the right sensor, change the "Left sensor" at the branch to the "Right sensor" and the "Turn left" command to the "Turn right" command, respectively. For the "Move forward" and "Turn right or left" commands, do not set the number of seconds (do not select "...for ?? sec.").



Modify the above programs to make a program for Line Trace using both sensors.

The present program is structured so that the flow goes from one sensor (branch) to "Move forward" or "Turn" and vice versa. First, let's modify the program so that it goes "Check one sensor and proceed to "Move forward" or "Turn", and then proceed to the next sensor (branch)". Connect the arrows of "Move forward" and "Turn left" extending from the branch of the left sensor to the branch of the right sensor. At the same time, connect the arrow of "Start" to the branch of the left sensor.

By changing the connection of the arrows, the flow that "Check the left sensor, and then check the right sensor" has been made. For the right sensor, however, the present flow is structured so that the operation goes from the right sensor to "Move forward" or "Turn right" and vice versa. Connect the arrows of "Move forward" and "Turn right" extending to the branch of the right sensor to the branch of the left sensor.



Now, the program for checking both right and left sensors has been made. Write the program into the robot and execute it

After execution of the program, the robot will not move properly. **It moves forward without following the line even if it sees the line.** Apparently, the program is correct, but there seems to be a problem somewhere.

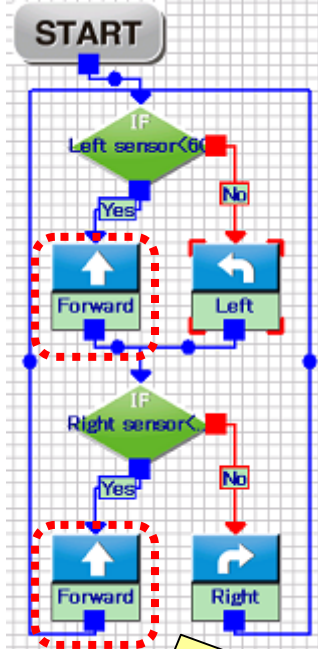
Let's check to see what is wrong with the program by following the sequence of the commands.

First, concerning the case where **both right and left sensors do not detect the line**, the first branch of the left sensor is the same as that of the correct program, and "Move forward" is selected correctly. The next branch of the right sensor is also the same as that of the correct program and "Move forward" is selected correctly. **At both branches, the same command "Move forward" is selected, the robot moves forward properly.**

Next, concerning the case **where the left sensor detects the line, at the branch of the left sensor, the command "Turn left" is correctly selected.** Then, **at the branch of the right sensor, since the right sensor does not detect the line, "Move forward" is selected.** At this time, **this "Move forward" command and the previous "Turn left" command conflict with each other.** In fact, this is the cause of the problem. **At the "Turn left" command, the robot tries to stop the left motor, but the motor will not stop immediately. In the meanwhile, the "Move forward" command is given and soon the robot will run the left motor which was to stop. As a result, the robot keeps moving forward.**

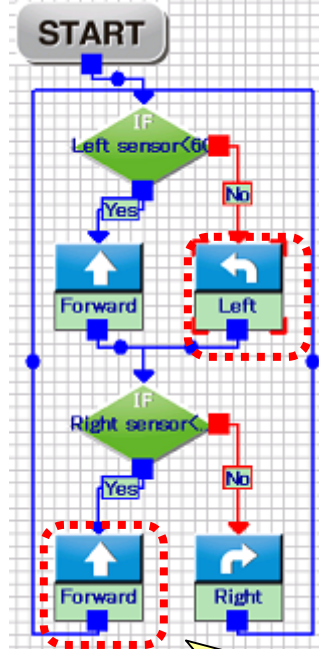
On the other hand, in the case **where the right sensor detects the line, at the branch of the right sensor, "Turn right" is correctly selected.** But **the left sensor does not detect the line, and therefore, the "Move forward" command selected at the previous branch of the left sensor conflicts with the "Turn right" command.** At the "Turn right" command, the robot tries to stop the right motor. But the program goes round and at the next branch of the left sensor, "Move forward" is selected. As a result, the right motor runs and the robot will keep moving forward.

When both right and left sensors do not detect the line:



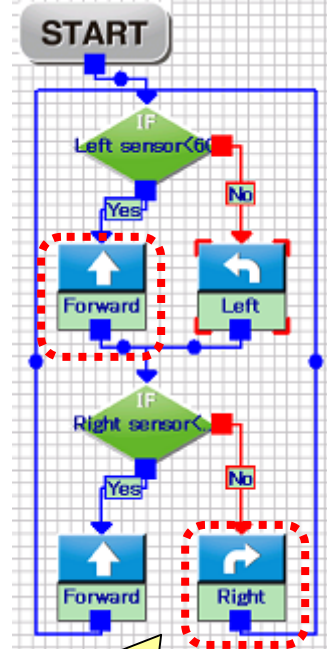
At both branches, "Move forward" is selected and the commands do not conflict.

When only the left sensor detects the line:



"Turn left" and "Move forward" commands conflict.

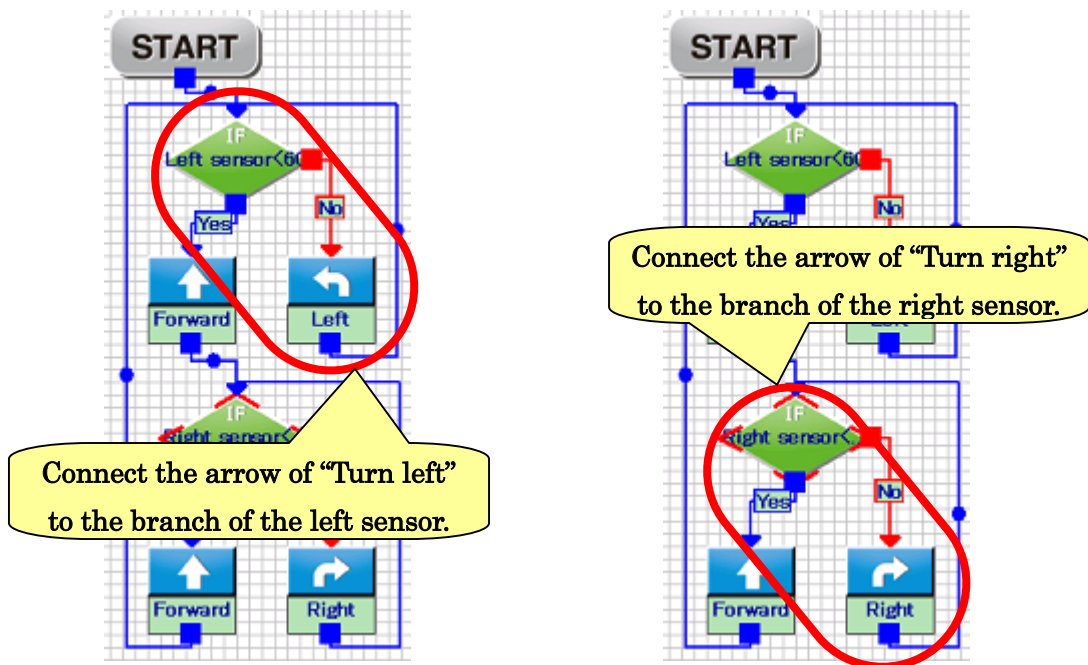
When only the right sensor detects the line:



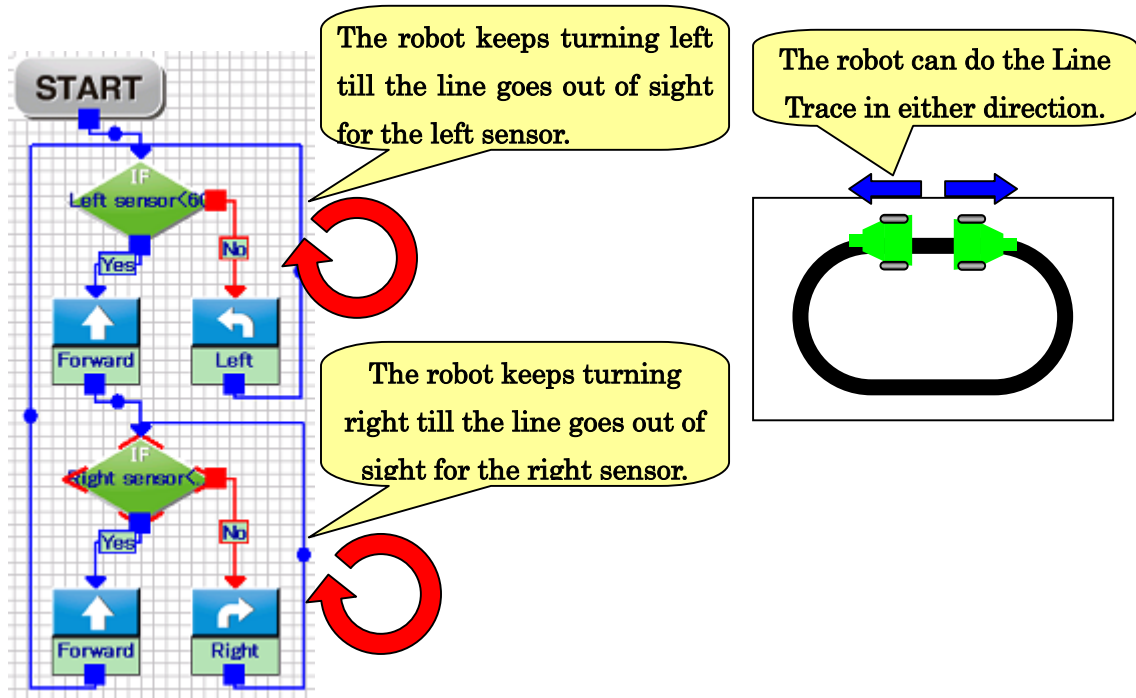
"Move forward" and "Turn right" commands conflict.

As a result of checking the sequence of the program, it was found that “if the commands at two branches conflict, a problem occurs. However, in order to check the right and left sensors, the operation must be programmed so that two branches are passed through. How can we create a program in which ”two commands do not conflict” and ”both sensors can be checked”? The answer is to make a program in which ”after the operation for one of the sensors =”Turn” command is completed, the robot proceeds to the next sensor”.

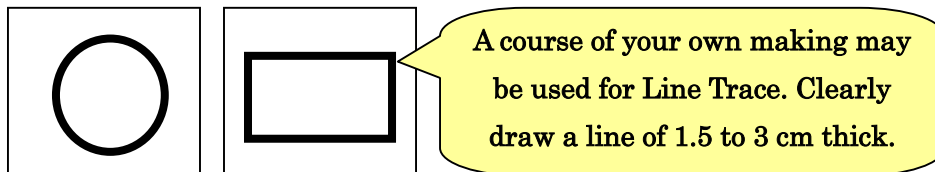
Now, let’s change the connection of the arrows in the present program as shown in the figure below. Connect the arrow of “Turn left” to the branch of the left sensor, and the arrow of “Turn right” to the branch of the right sensor, respectively. After that, execute the program.



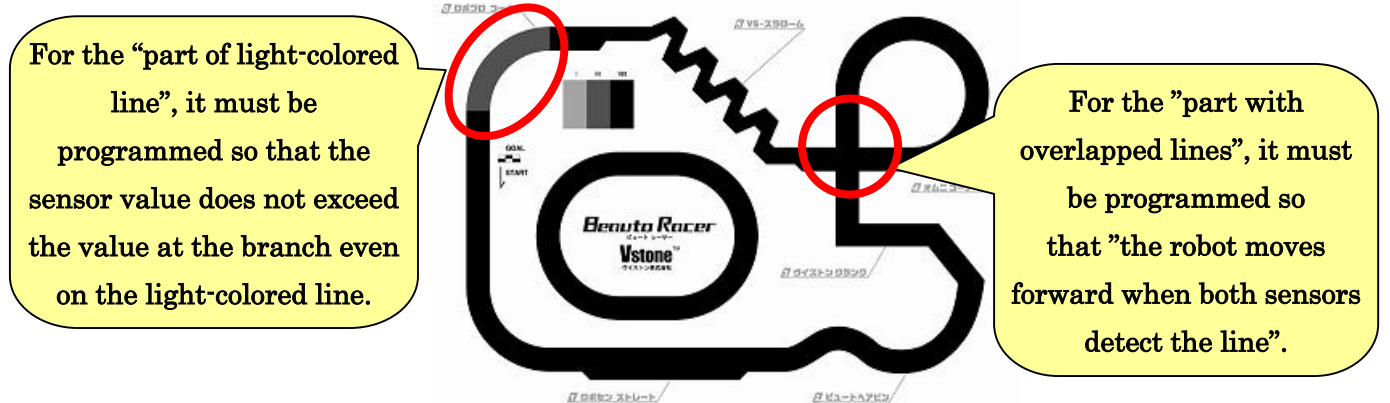
In this program, when the right and left sensors detect the line, the robot will "keep turning till the line goes out of sight for the sensor that detected the line". When the sensor is not detecting the line, the flow goes to the next sensor. Now, "Turn" and "Move forward" commands do not conflict and the robot can do the Line Trace properly.



With the program which has been created so far, the robot can do the Line Trace on a sequence course as shown in the figure below. Draw a line with a black magic marker on white paper or print a course drawn on PC and try running the robot on the course. It is important to draw a line as thick as possible (about 1.5 to 3 cm) deeply and clearly. If the motor speed is too high, the robot may deviate from the course soon. In such a case, adjust the motor speed to a lower rate.



This program can be used as a base for the Line Trace Course for advanced-level users which is contained in the CD provided with the product, but some parts require a little more complex programming. Especially, the “part of light-colored line” and ”part with overlapped lines” require more detailed programming.



For the “part of light-colored line”, check the sensor value at the part, and change the setting value at the branch command so that the part is detected as a line. For the ”part with overlapped lines”, make a program for ”the case where both sensors detects the lines”. In this case, the robot does not deviate the line if it ”Moves forward”. Therefore, make a program as such.

Check and change the setting value at the branch command so that the part of light-colored line is detected as a line.

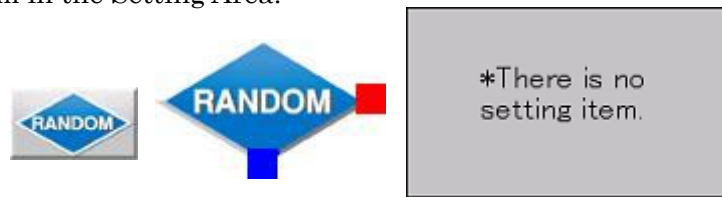
Neither of the sensors doesn't detect the line.

Only the left sensor detects the line.

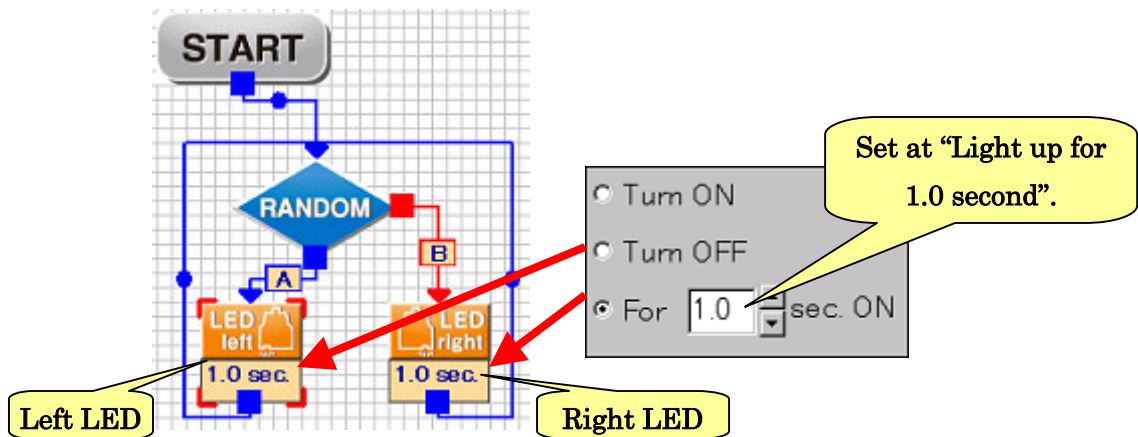
3-5. RANDOM command

Another command to branch a program is "RANDOM". Differently from a branch using sensors, no predetermined condition is set for a Random command. During execution of the program, the robot will determine by itself to which way it moves at the branch. A RANDOM command can be used in "roulette" where an accidental element is required or "exploration" and "Sumo" where incorporation of unintentional variety of motions are desired, for example.

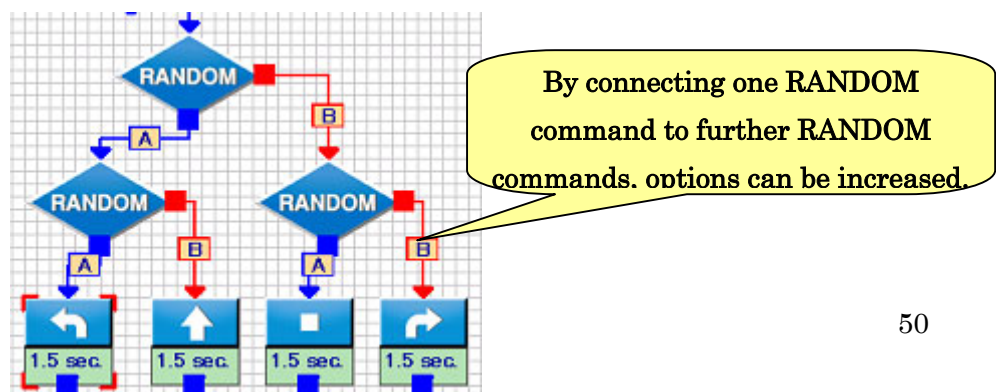
The RANDOM command is displayed in the Icon Area and Program Area as shown in the figure below, respectively. No condition is set for the RANDOM command, and there is no setting item in the Setting Area.



Now, let's create a program using RANDOM in which "the right and left LEDs are lighted up at RANDOM for 1 second, respectively," as a sample. Add one RANDOM command and each one of the right and left LED commands and connect the arrows as shown in the figure below. For the LED commands, set "Light up for 1.0 second", respectively. After the program is created, write the program into the robot and execute it.



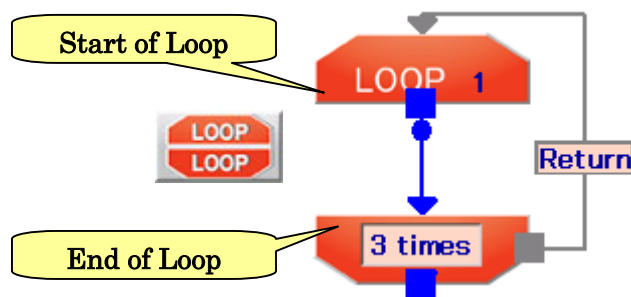
Basically, the RANDOM command has two options "A" and "B". However, by connecting a number of commands, options can be increased. For example, as shown in the figure below, connect one RANDOM to further RANDOM commands, resulting in four options.



4. Creation of program using LOOP

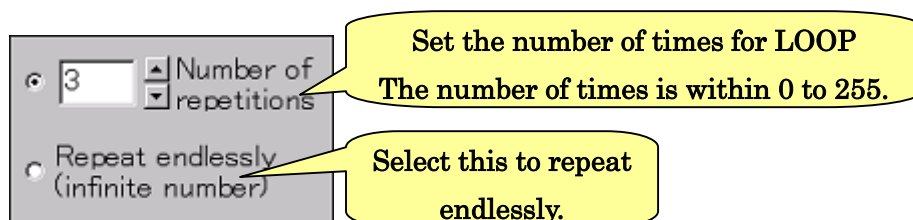
As an example, a “program to flash LED 100 times at intervals of 1 second” will be created. According to the method explained so far, two commands, “Leave the LED ON for 1 second” and “Wait for 1 second”, must be placed each by 100 pieces alternately. However, such a programming takes a lot of time and labor. Also, such a long program cannot be recorded into the robot body. Therefore, the “LOOP” command is used to repeat the same command over and over again.

The “LOOP” command is displayed in the Icon Area and the Program Area as shown in the figure below.

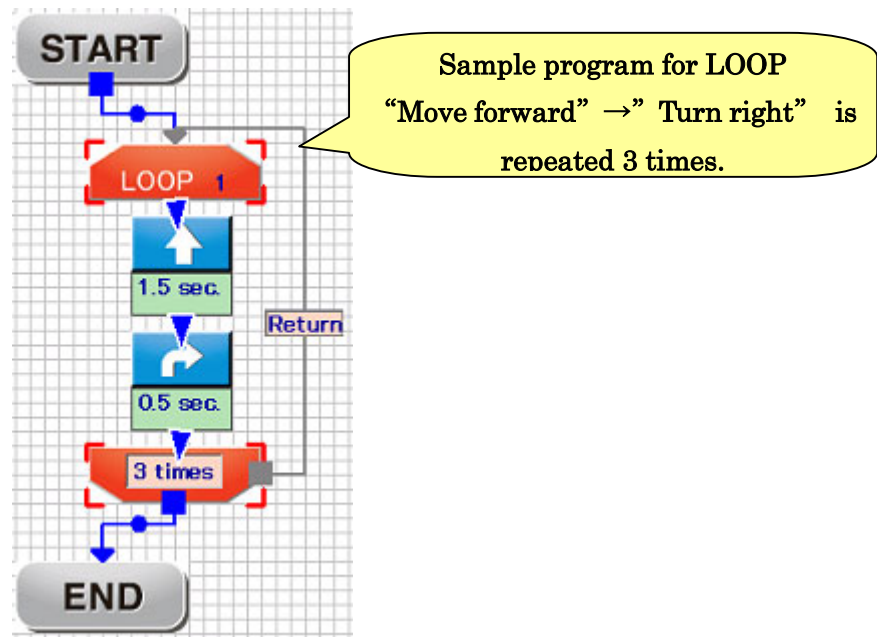


For the LOOP command, two action blocks are added at one time. They indicate “Start of LOOP” and “Exit of LOOP”, respectively. Just as a program is executed from “Start” to “Exit”, LOOP has a start and an end and the commands between them are repeated for the set number of times. If arrows for the Start of LOOP and End of LOOP are not be connected properly, the execution of commands are not repeated properly.

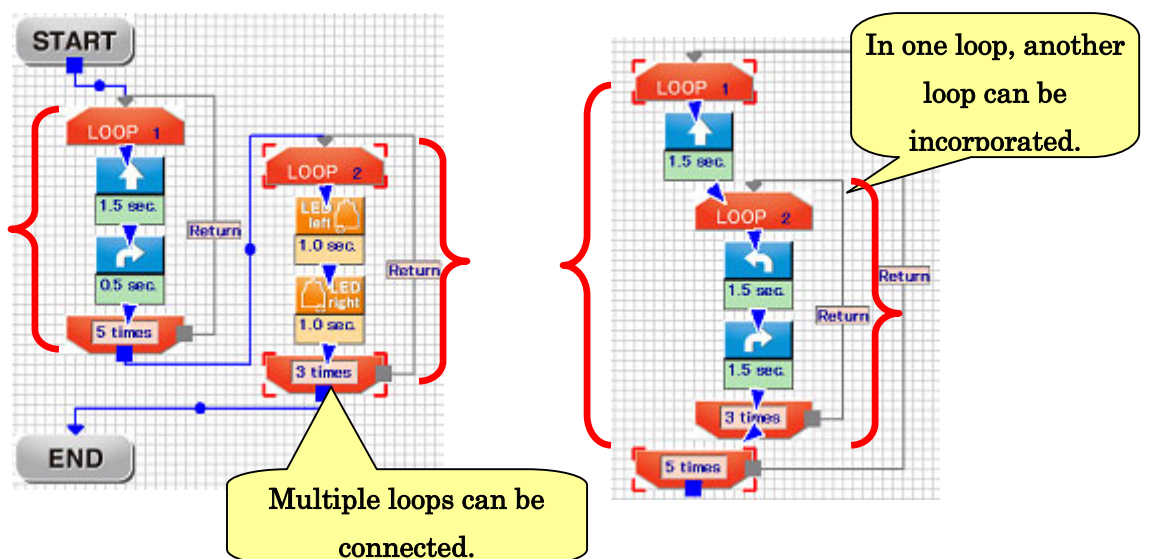
In the Setting Area, the display for LOOP is as shown in the figure below. The number of times can be set for LOOP, which is 0 to 255. If the number of times for LOOP is set at 0, the program will go ahead without executing the commands for LOOP at all. To repeat the program endlessly, select “Repeat endlessly (infinite number)”.



Now, let's create a program using the LOOP command. Create a program as shown in the figure below and execute it. **The robot will repeat the motion of " Move forward for 1.5 seconds" →" Turn right for 0.5 seconds" three times, and the program will end.** By changing the number of times for LOOP in the Setting Area, the motion will be repeated by the set number of times.

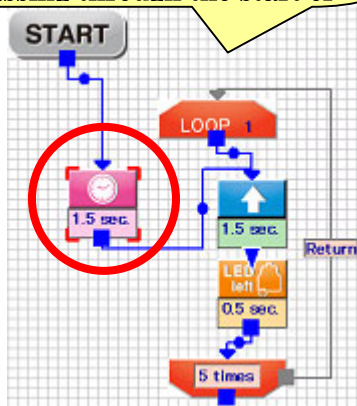


Up to seven LOOP commands can be used in one program. By combining multiple LOOP commands, more complex loops can be programmed. In order to execute the LOOP command properly, however, it is important to understand how to connect arrows well.

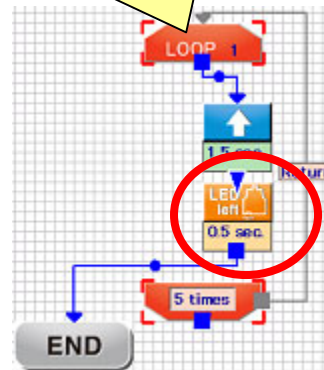


When a loop is connected as shown in the figures below, a warning message will be displayed when the program is written into the robot.

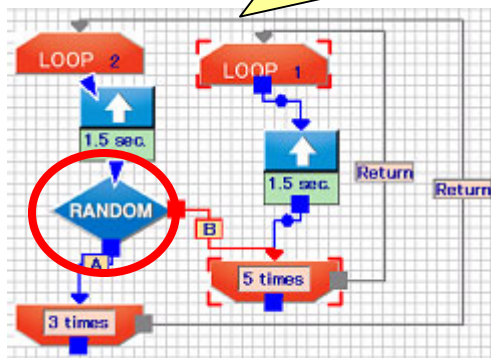
NG: The arrow is connected directly into the LOOP without passing through the start of



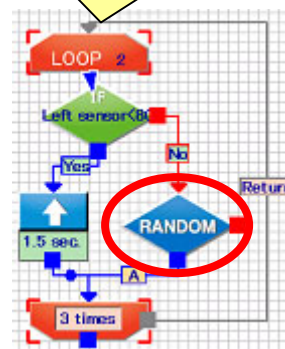
NG: The program ends without passing through the end of LOOP.



NG: At the midway of one loop, the arrow breaks into another loop.



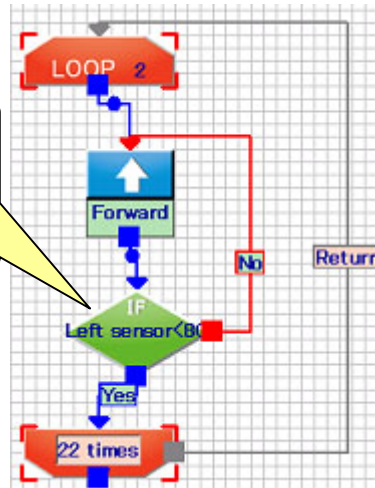
NG: An arrow which has not been connected at the midway of one loop exists.



In either case, the major causes are that "the program ends before the loop ends" or "the start and the end of the loop are connected in a wrong combination." If there is a problem in LOOP, the action blocks in the order from the start of the program to the point having the problem are displayed in different colors. Using it as a hint, solve the problem.

For reference, the connection of arrows with which “the program does not end in a loop” as shown in the figure below has no problem. With the program shown in the figure below, ”the program will end when the sensor reacts 5 times.”


It proceeds to the next block when the left sensor value exceeds 60. After proceeding 5 times, it goes out of the loop and ends.

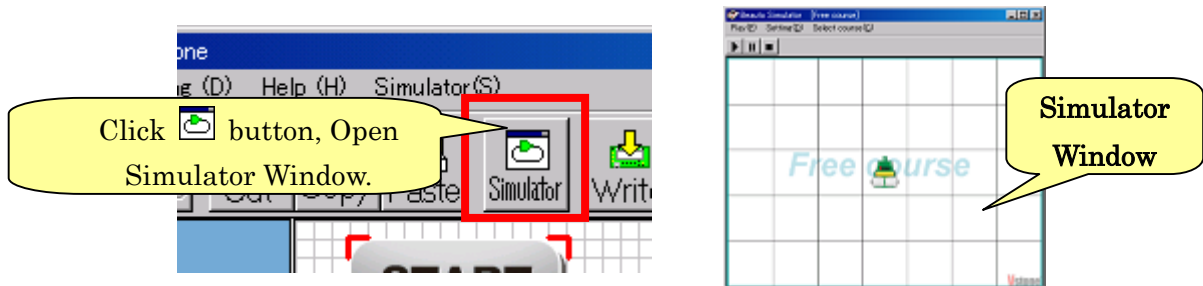


5. Using Simulator

With Simulator function it will be able to learn programming in case of no real machine. Please note that the virtual robot is different in motor speed and the sensitivity of sensors from the real one, so the performance is not always correspond with each other.

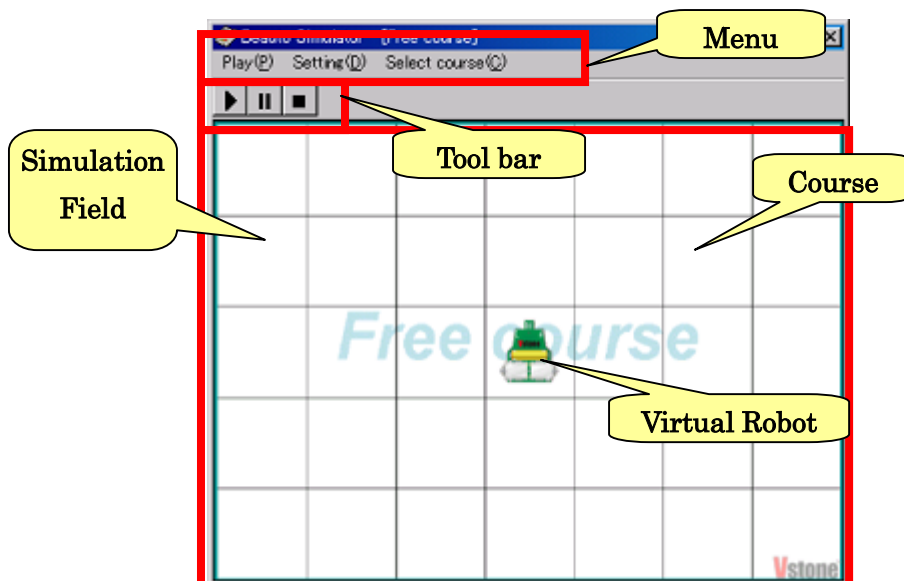
5-1. Running Simulator

Click tool bar  button to open Simulator Window. A program is created in Program area same as the real robot. Simulation works in Simulator Window

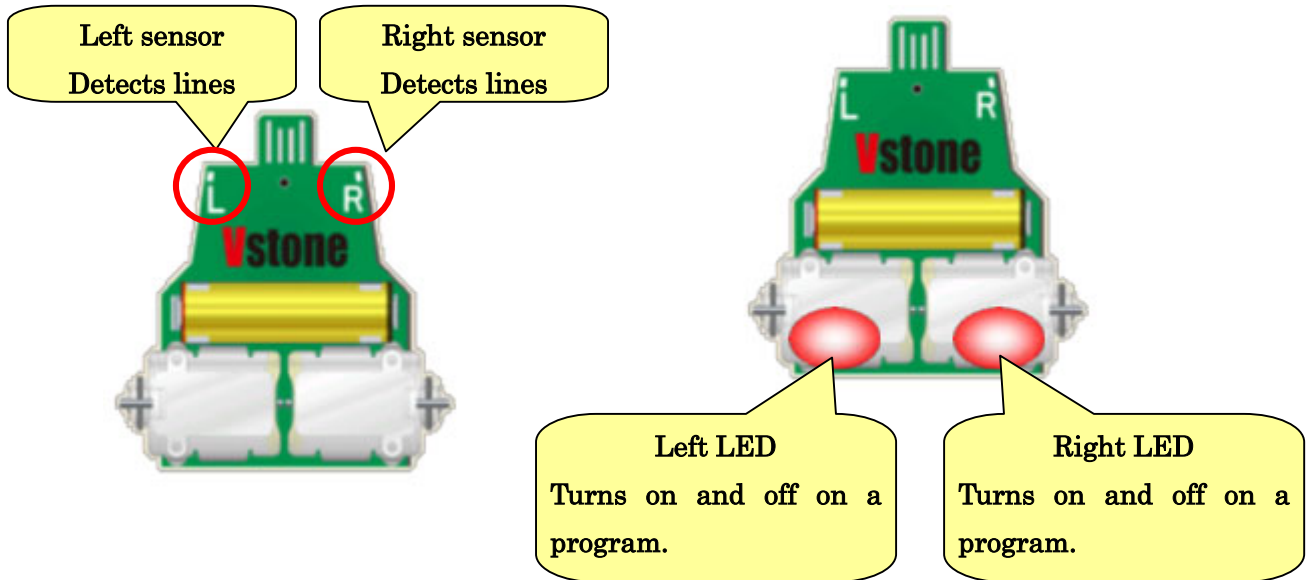


5-2. Explanation of Simulator window

The window of this software is roughly divided into an upper part and a lower part. At the top of the window, the "menu" and "toolbar" are provided. The menu refers to the character strings directly below the window title, which is used for starting simulation and selection simulation course. At the lower area it is virtual space, it appear virtual robot and course.



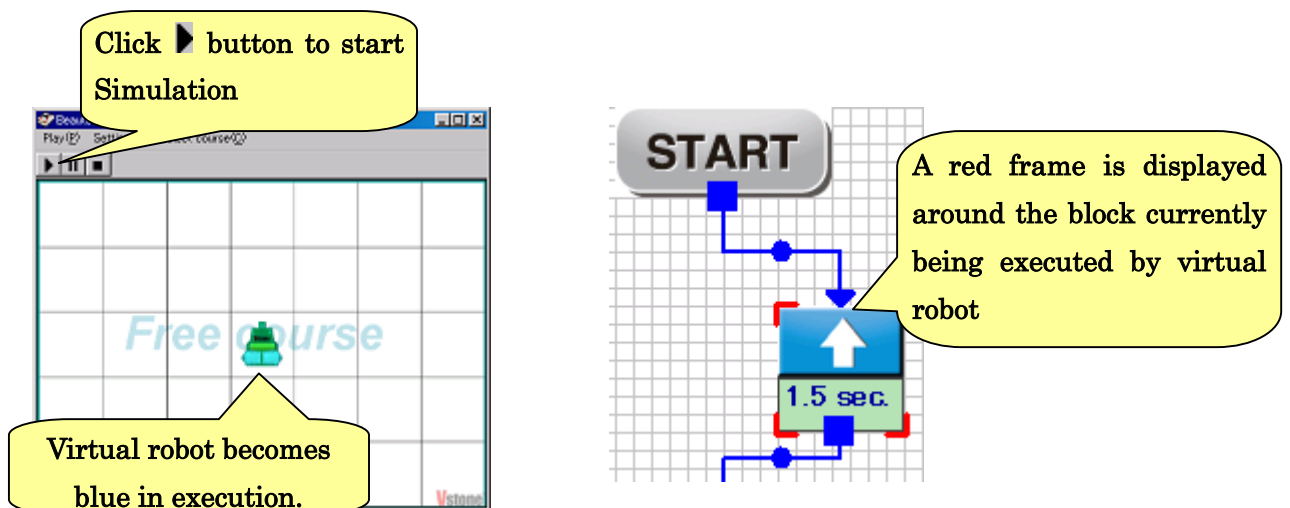
The Virtual robot body is equipped with interfaces including motors, sensors and LEDs same as BeautoRacer. The location and name of each equipment are as shown below.



5-3. The operation of Simulator

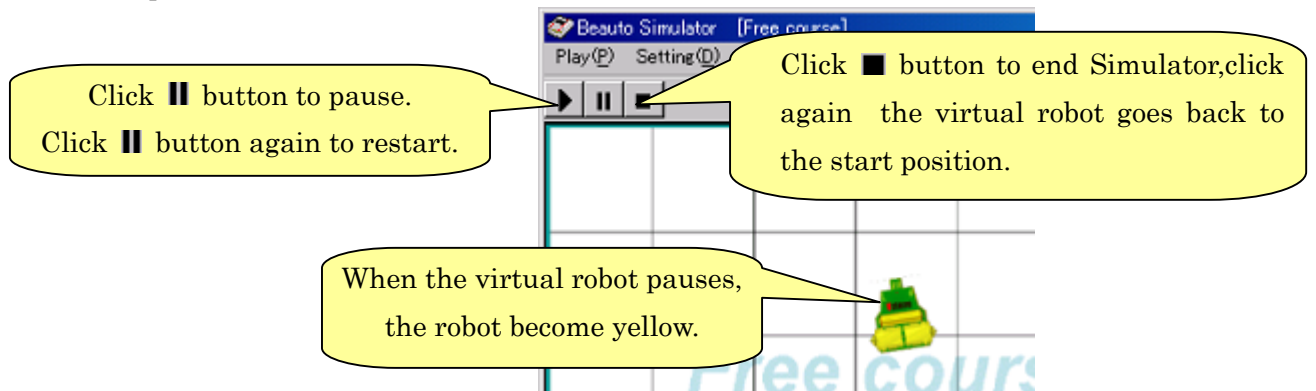
Click ► button to start Simulation, and the virtual robot become blue and start the created program. After the program is executed till the end, the robot stops.

In the program is excuted, **A red frame is displayed around the block** that the virtual robot currently executes.

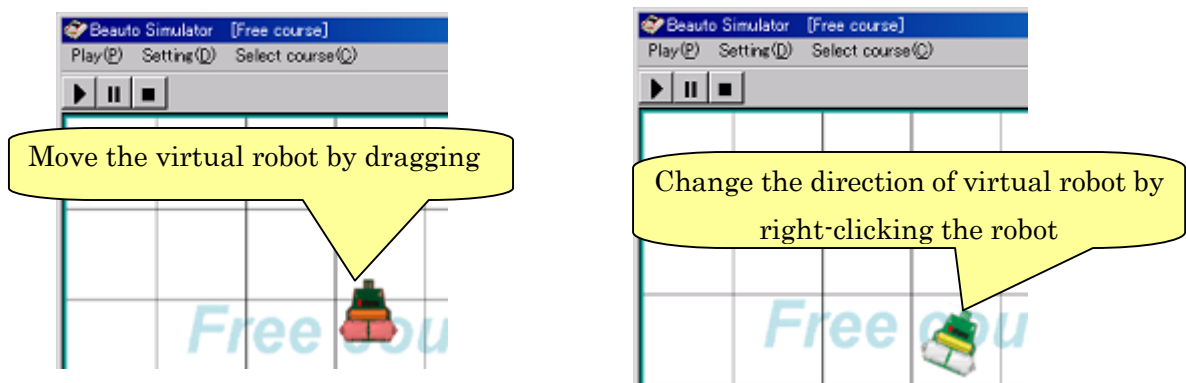


Click ■ button to end Simulator. Click || button to pause. When the virtual robot pauses, the robot become yellow. Click || button again to restart the program.

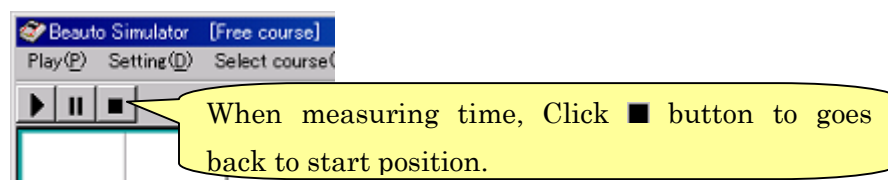
Clicking ■ button Before executing the program, the virtual robot goes back to the start position.



When the virtual robot does not become blue(not executes), you can move the virtual robot by dragging and change the direction of virtual robot by right-clicking the robot.

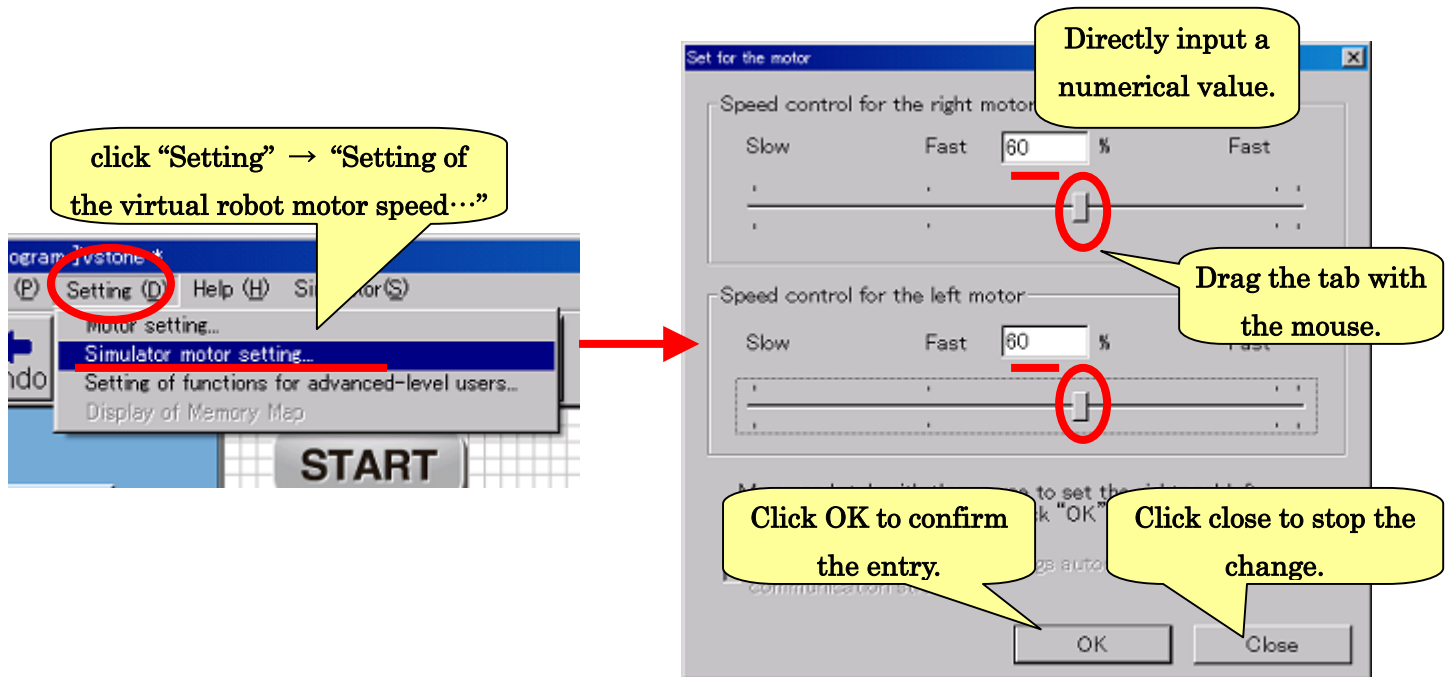


When changing the position and the direction of the virtual robot, it does not measure the time. Click ■ button to restart for measuring time.



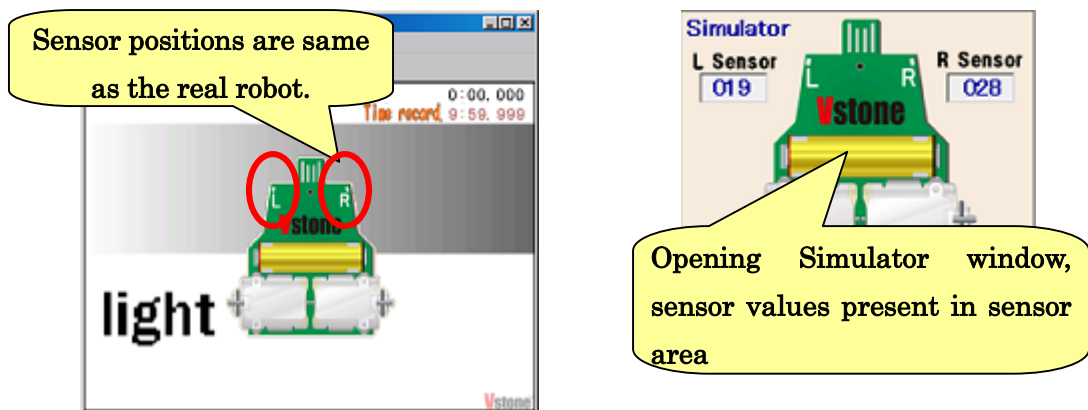
5-4. Adjustment of the virtual robot motor speed

The virtual robot motor speed can be set by the similar way of the real robot. In the menu at the top of the window, click “Setting” → “Setting of the virtual robot motor speed...”, and the dialogue for setting opens.



5-5. Checking the Sensor values

When opening Simulator window, sensor values present in sensor area. There are two sensors at the front “L” “R” of robot. That can sense the color shading. The Sensor value indicates 10 on the white area and 160 on the black area.






5-6.Menu and Toolbar in Simulator Windows

"Menu" and "toolbar" are provided in Simulator Window. The menu refers to the character strings directly below the window title, which is used for starting simulation and selection simulation course. Each functions are the followings.

Menu and Toolbar in Simulator Windows

Execution

- Start() = Executing the program
- Stop() = Stop the executing the program. When not executing the program, set the virtual robot at the initial position.
- Pause() = Pause the program. Click again and restart the execution.

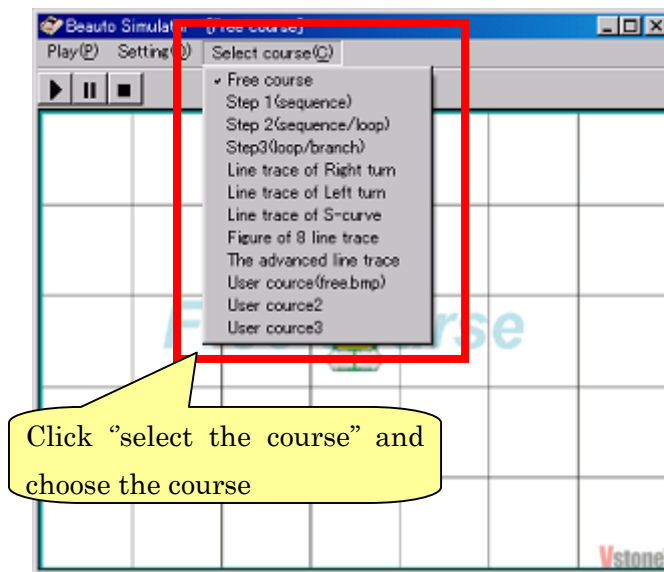
Settings

- Zoom out = Switch viewpoint to watch the whole course.
 - Zoom in(Twice) = Magnify an image twice
 - Zoom in (Four times) = Magnify an image four times
 - Standard Play = execute the program at the standard speed.
 - Play slowly= execute the program at the slow speed.
 - Play more slowly= execute the program at the much slower speed.
 - Check records = Present the fastest record.
-
- Select the course
 - Choose the course from registered courses.

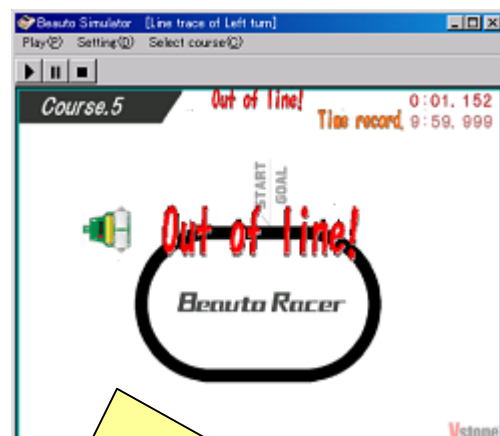
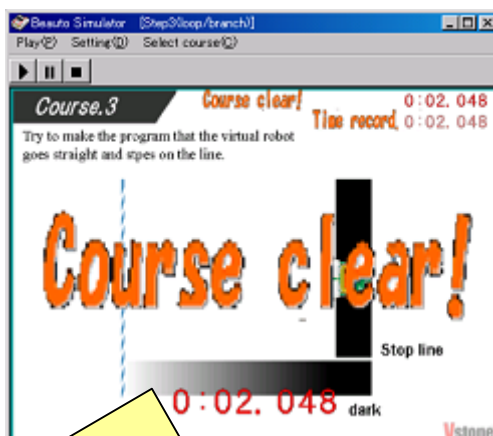
5-7. Select the course

It is possible to select the course from 10 registered courses and try the line trace program in your PC. This Simulator determines whether the branch block is used, whether the virtual robot run in the correct path and so on. You can record the your fastest time.

Changing the course, click “select the course” and choose the course that you want to try.

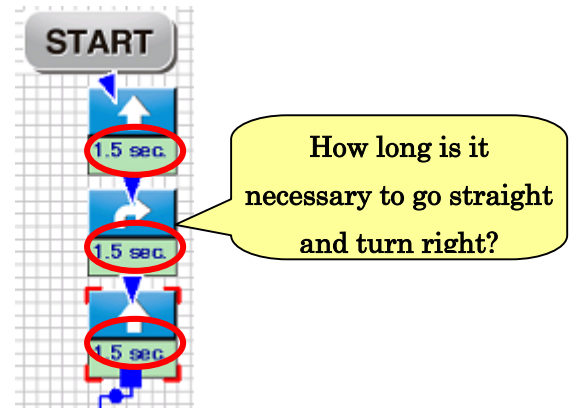
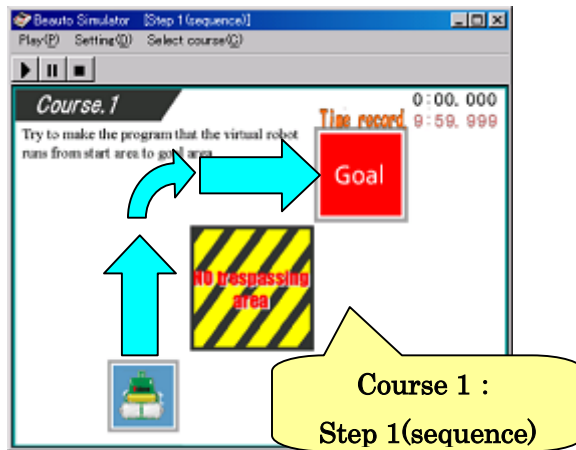


Each course has course-based task. It is possible to accomplish each course by referring “2. How to create a sequence program”, “3. Creation of a program using the sensors”, “4. Creation of program using LOOP”.

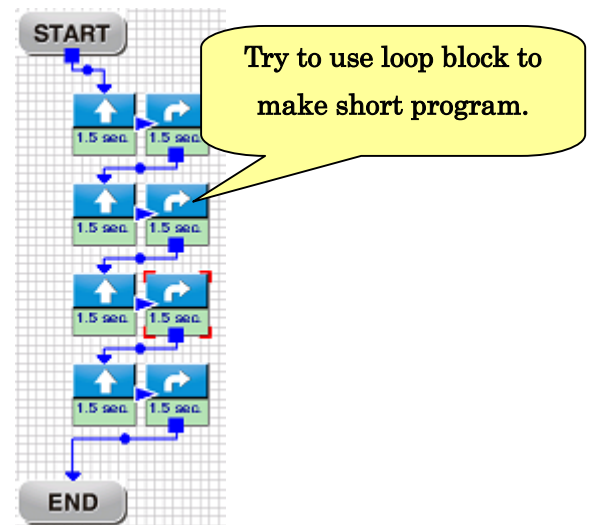
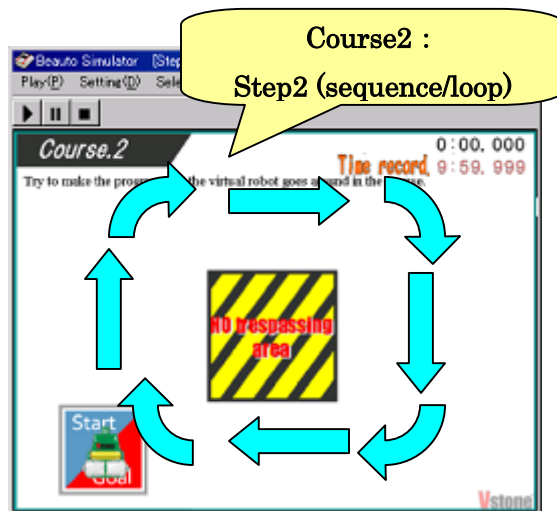


5-7-1. trying sequence program course (course1/2)

At the course1 and course2, a task is that the virtual robot runs from start area to goal area without entering NO Trespassing area.

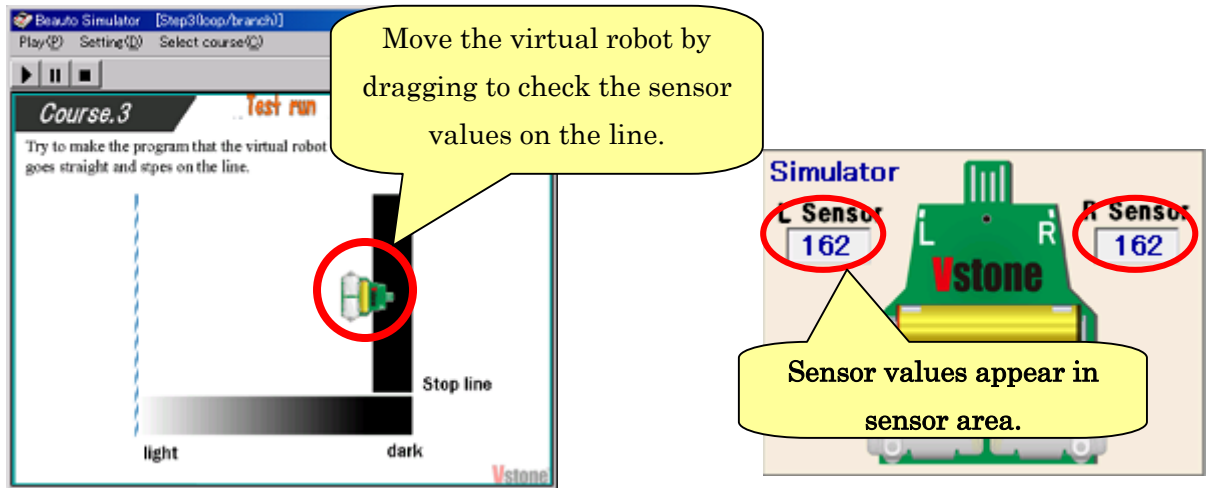


At the course 2, it is better to use loop block.

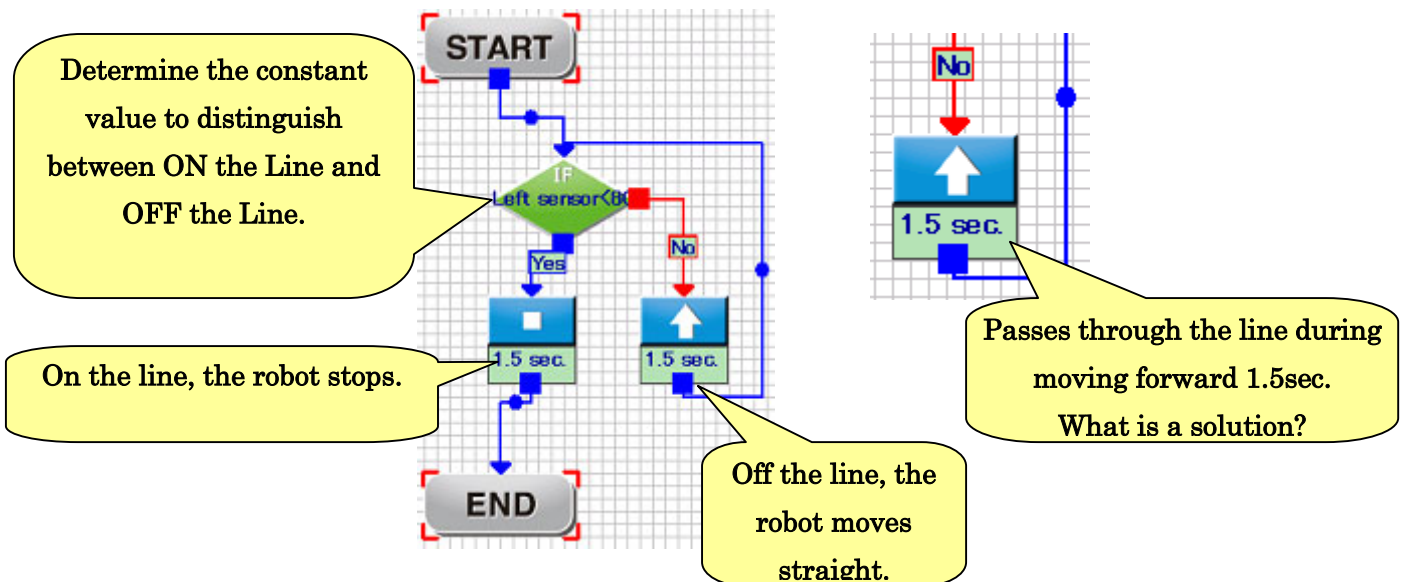


5-7-2. Using sensors (course 3)

To accomplish the course 3, check the sensor value on white area and black line.

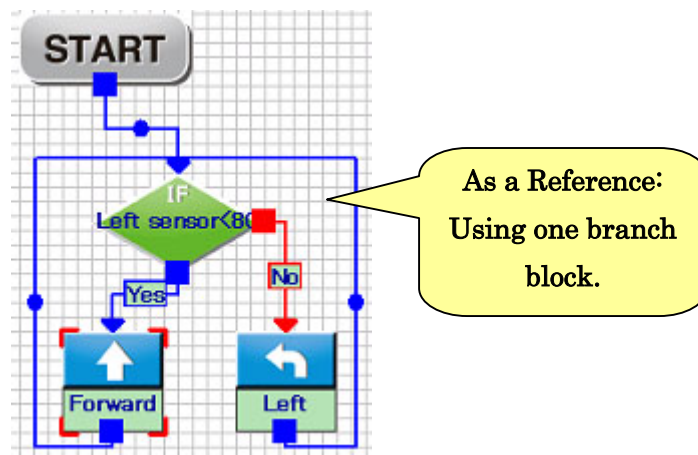
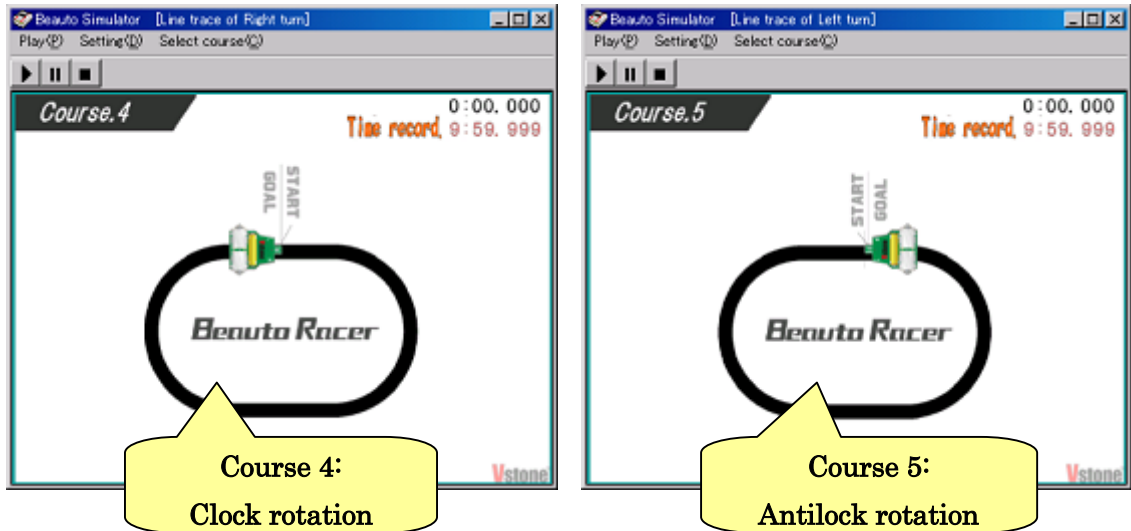


Based on the values you checked, Make a program that is "when the robot sensor is placed off the line, the robot moves straight." and "when the robot sensor is placed directly on the line, the robot stops." Refer the chapter "3-2. How to use Branch".



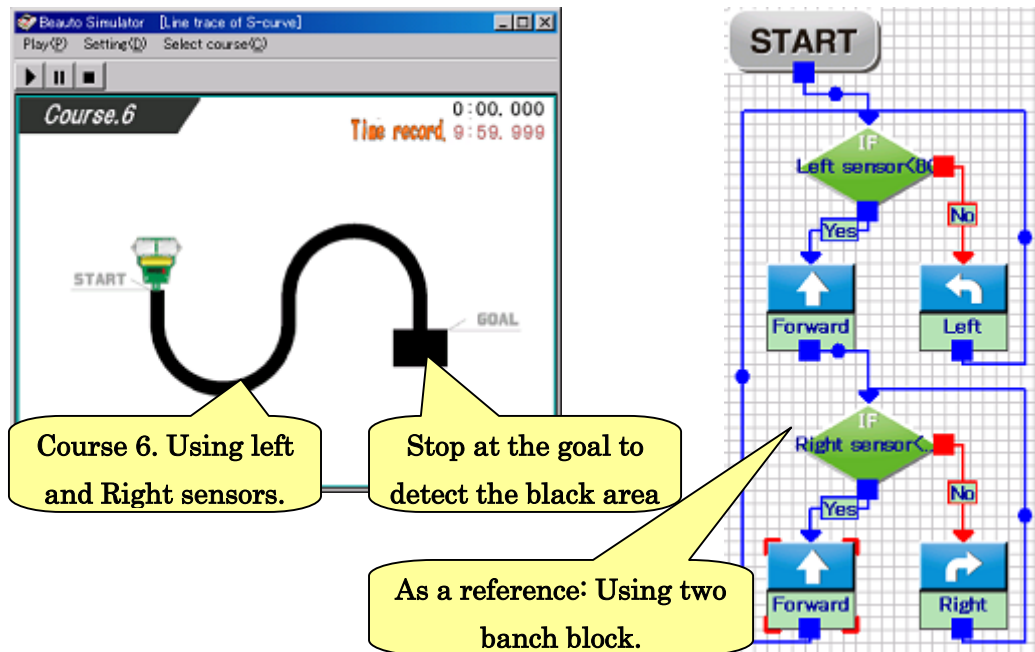
5-7-3. Line Trace using a sensor (course 4,5)

At the course4 and course5, a task is that the virtual robot traces the line by using a sensor. Course 4 is clock rotation. Course 5 is anticlock rotation. Refer the chapter “3-3. Programming for Line Trace”, decide which sensor should be used and make the program.

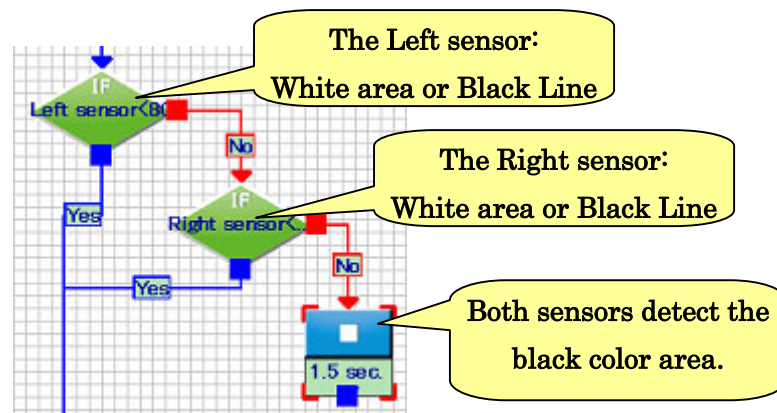


5-7-4. Line Trace using the sensors (course 6)

The course 6 is a S-curve line course. The task is that robot traces the line and stops the goal area. It is necessary to use left and right sensor, because the course has left and right turn. This task is not easy, the chapter “3-4. Line Trace using the right and left sensors” will be the useful reference.



For the goal area, make a program for “the case where both sensors detects the Black lines”. The one branch block enables to use one sensor. So use more than two branch block and link them

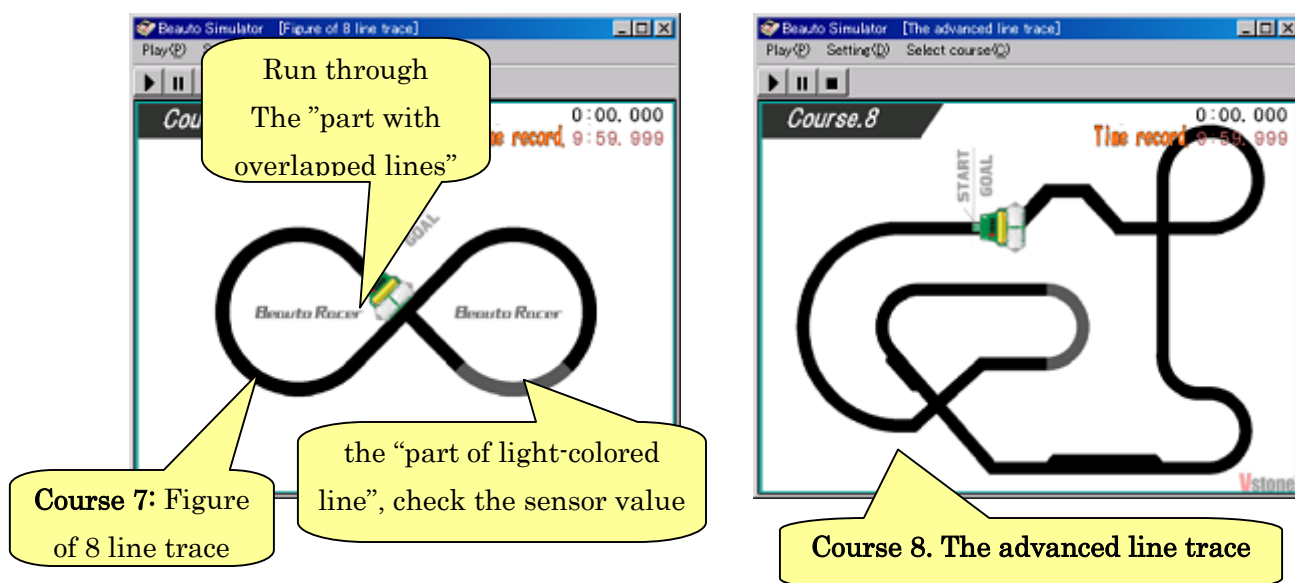


5-7-5 .Figure of 8 line trace and the advanced Line trace.(Course 7,8)

The rest of two courses are advanced and difficult. The "part with overlapped lines" and the "part of light-colored line" require more detailed programming.

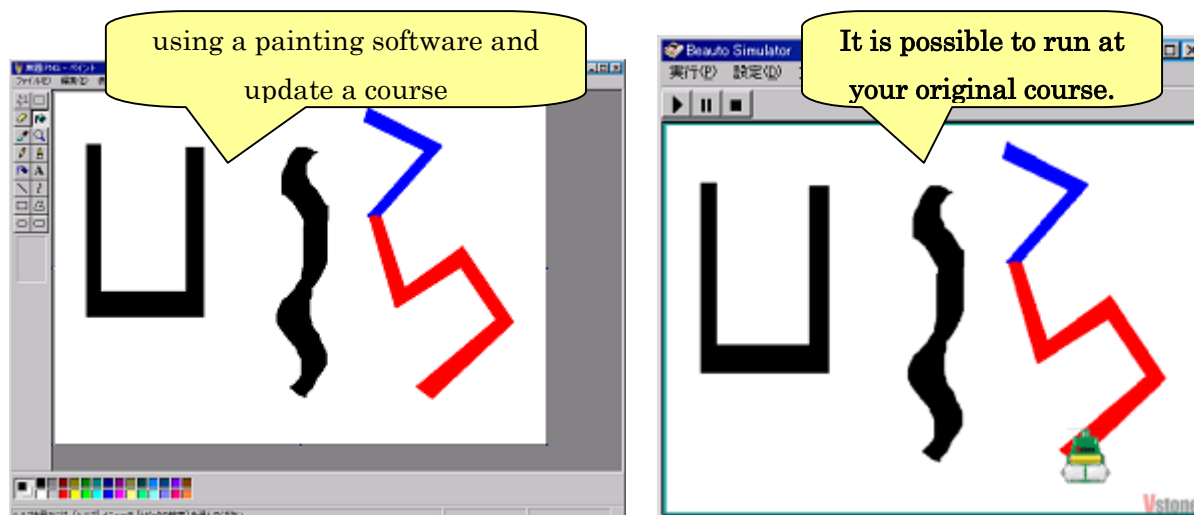
To run through The "part with overlapped lines", apply the course6 program and change it to "Go forward when the two sensors detect black line."

For the "part of light-colored line", check the sensor value at the part by dragging the virtual robot and change the setting value at the branch command so that the part is detected as a line.



5-7-6.The original course (course 9)

The last course "User setting (free.bmp)" is the plain field. It is possible to make an original course by using a painting software and update a course.



The image file of the course 9 is "free.bmp" file in the folder "texture" of this software. You can update this file by using painting software like Windows "Paint" and draw a course. (This original course can not detect your success and failure like other courses.)

Please note the below, when you change the image file and save. If failed, the simulator doesn't work well. Before update the file, make backup file.

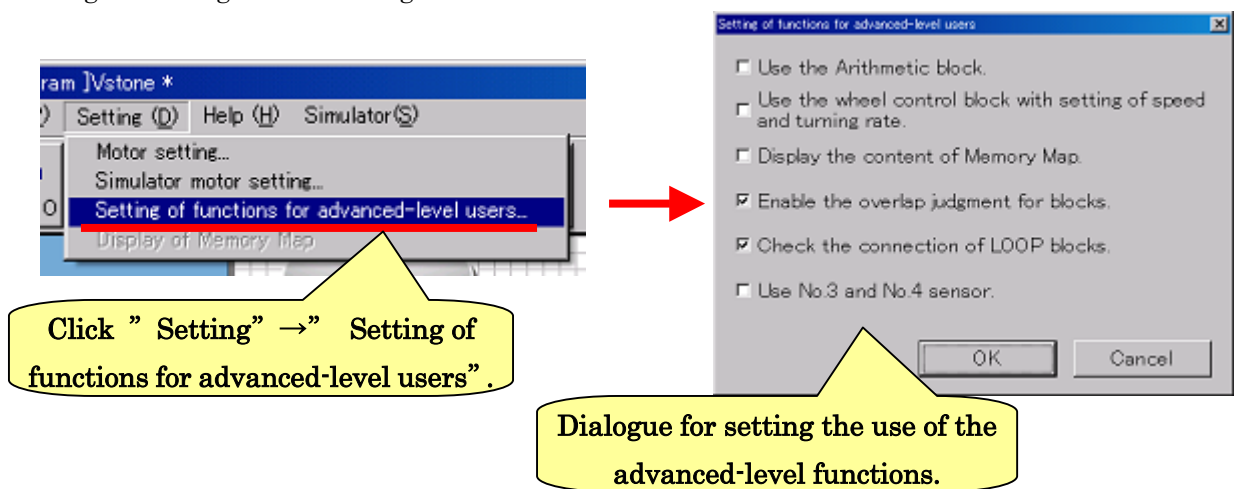
- **Do not change file name.**
- **Keep the file type "24 bit bitmap"**
- **Keep the size of this image file 700×500 pixel**

There is no limitation about colors and shapes. It is better to draw bold and deep black line for a line trace course. One pixel is equal to 1mm. So you can reproduce the course that you made the course on paper.

6. Functions for advanced-level users

This software has the functions for advanced-level programmers using variables and the Memory Map to allow for various programming. While a sequence program can be created satisfactorily according to the procedures explained so far, the functions for advanced-level users allow you to understand programming more deeply.

The functions for advanced-level users cannot be used only with this software installed. In order to use the functions, the setting needs to be changed from the setting dialogue. The setting dialogue for the functions for advanced-level users can be opened by clicking "Setting" → "Setting of functions for advanced-level users" on the menu.

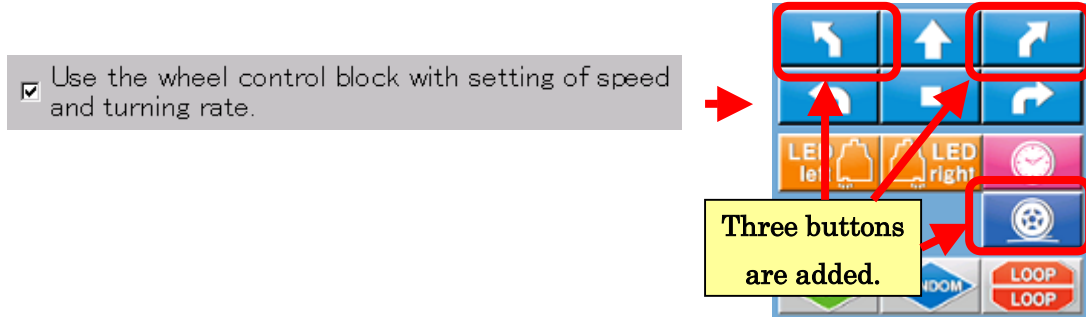


On the dialogue, 5 setting items are displayed. By clicking, place or remove the checkmark in the checkbox to change the setting. Click "OK" on the dialogue, and the new setting will be applied. Hereinafter, the details of each setting item will be explained.

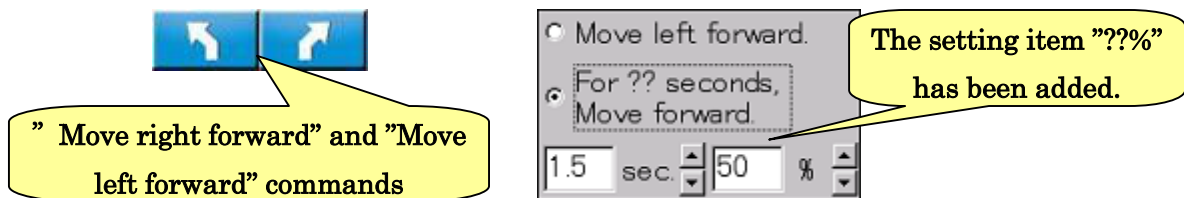
6-1. Wheel control with setting of speed and turning rate

The items which are easy to set will be explained first. By placing a checkmark in the box for "Use the wheel control block with setting of speed and turning rate", the command with which any motor speed can be set becomes enabled. The motor speed under the initially usable commands for the motor such as "Move forward" and "Turn" was always constant. At that time, the speed could be changed by setting the motor speed, but in one program, the robot always moved at the same speed. On the other hand, this function allows you to change the motor speed freely by command in one program.

By placing a checkmark in the box for "Use the wheel control block with setting of speed and turning rate" and clicking "OK" on the dialogue, the buttons as shown below will be added into the Icon Area.

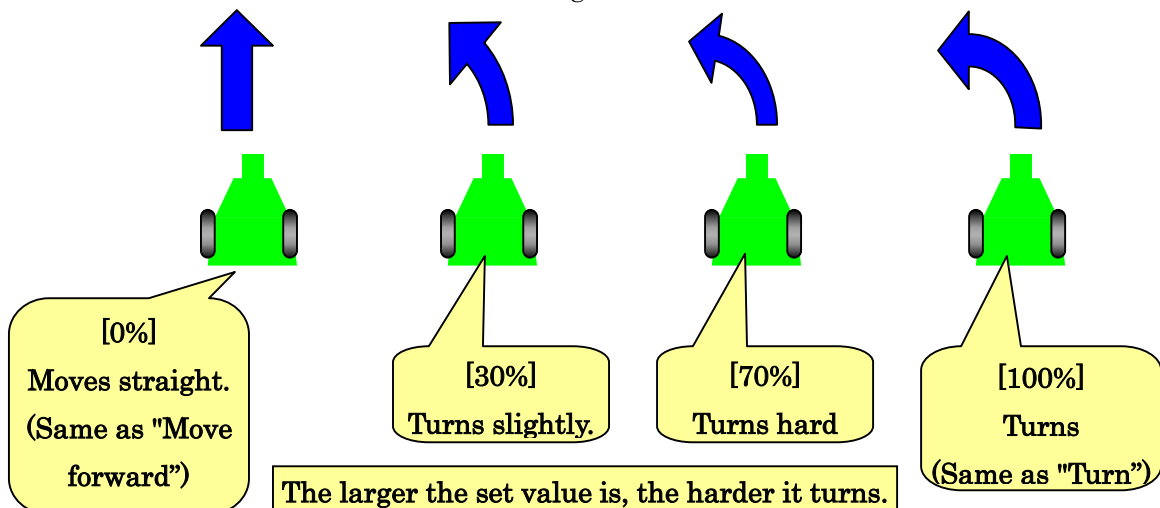


Two buttons shown in the left figure below are for the commands "Move right forward" and "Move left forward". As extensions of the "Turn right" and "Turn left" commands, these are used to set "at what rate the robot turns". The items to be set for these commands in the Setting Area are as shown in the right figure below. Most of the items are the same as those for the motor command previously explained, but the setting item "??%" has been added.

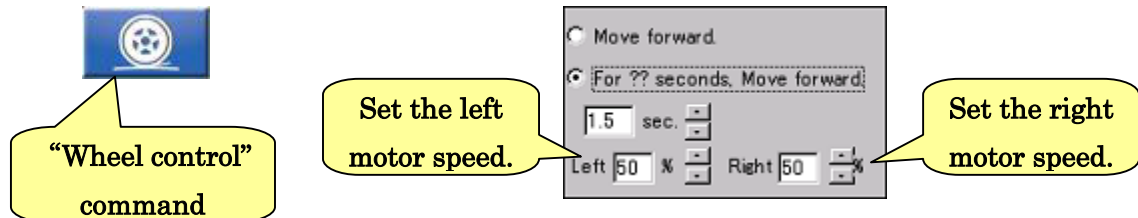


For "??%", enter a value of 0 to 100. "The smaller the value is, the more straight the robot moves" or "the larger the value is, the harder the robot turns."

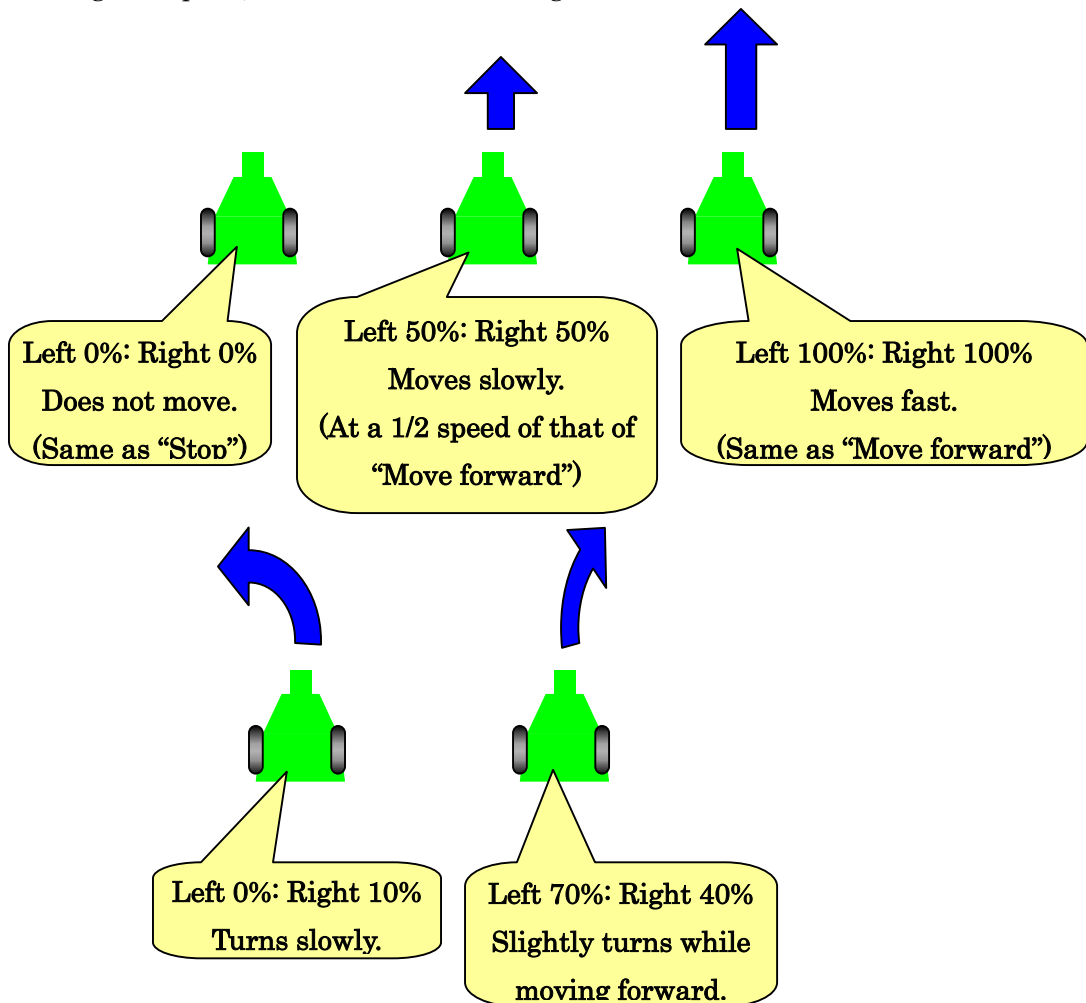
The other setting items are just the same as the previously explained motor command. When "Move for ?? seconds" is selected, the robot moves for the designated time. When "Move left frontward" or "Move right frontward" is selected, the robot proceeds to the next command with the motor running.



The button shown in the left figure below is for the "wheel control" command. As extensions of the "Turn right" and "Turn left" commands, **this command is used to set any given speeds for both right and left motors.** The setting items for this command in the Setting Area are as shown in the right figure below.



The setting items "??%" for the wheel control command are for the right and left motor speeds. For "??%", enter a value of 0 to 100. The larger the value is, **the higher the motor speed becomes.** By entering the same value for the right and left motor speeds, the robot can move forward at a given speed, "move right forward" or "move left forward" at a given speed, or turn at a flexible angle.



the motor.

“Gain of the motor” indicates the motor speed. **As in the case of the motor speed, the larger the value is, the faster the motor runs.** The value here is displayed within 0 to 255. **The value for this can be changed only from the ”Setting for motor speed” and it cannot be changed from the program.**

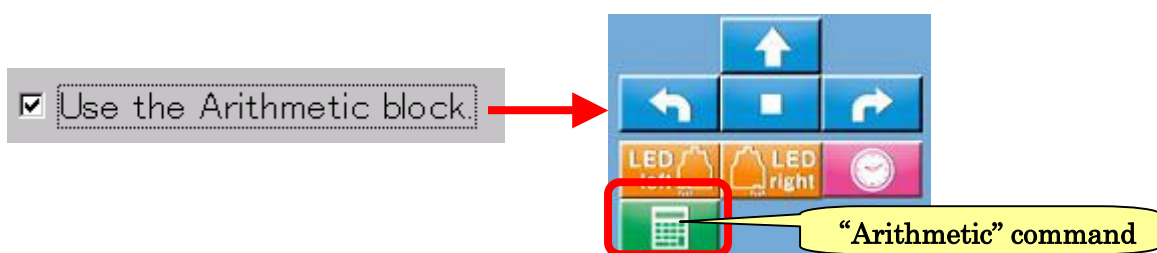
“Left sensor” , ” Right sensor” and ” Sensors ③ to ⑥” indicate the robot sensor values. The robot has only two sensors, but up to four more sensors can be added separately. In that case, the sensor values for the added ” Sensors ③ to ⑥” are displayed. Even if nothing is connected to the sensors, the values may change by themselves due to instability of input. Also, **since the values here are constantly updated by the robot, they cannot be written form the program.**

“LED” shows the ON/OFF status of the right and left LEDs. **The ON/OFF status is displayed by a value of 0 to 3.** “When it is 0, both of the right and left LEDs are OFF”, ”when it is 1, only the left LED is ON”, ”when it is 2, only the right LED is ON”, ”When it is 3, both of the right and left LEDs are ON”. The robot may change the values by itself and you may change it to any value you like on the program.

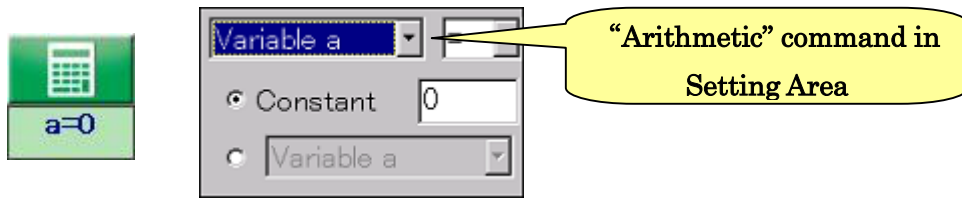
“Variable a to h” indicate variables which can be used in the program. **”Variable” is provided so that the user can freely write values to be used in the program, and the robot never change the values by itself.** By making a program that “Record the sensor values at intervals of 1 second in the order of variables a to h”, the sensor values for the last several seconds can be recorded in the robot.

6-3. Use of Arithmetic block

By placing a checkmark in the checkbox for “Use the Arithmetic block”, the ”Arithmetic” command for rewriting the Memory Map can be used. The button as shown in the figure below is added in the Icon Area.

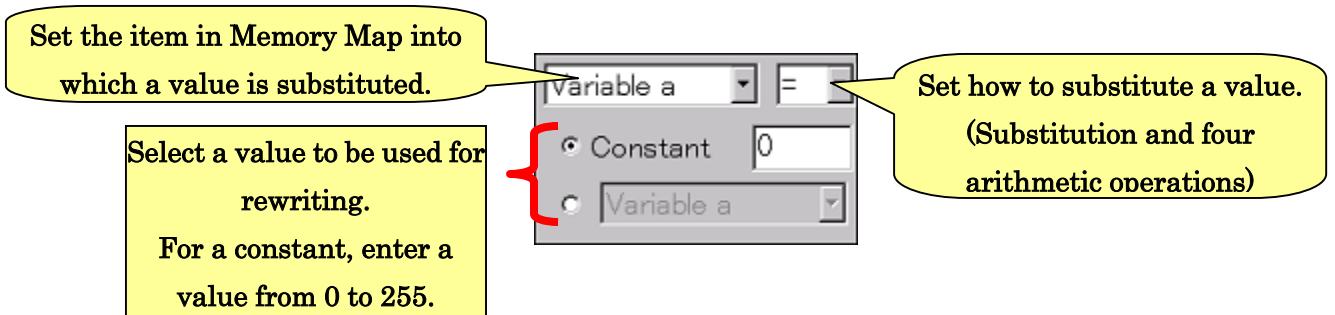


The "Arithmetic" command is displayed in the Program Area as shown in the figure below. In the Setting Area, it is displayed as shown in the right figure below.



A number of items are included in the Setting Area. Each of them will be explained. **On the first line, the item in the Memory Map into which a value is substituted is selected.** It can be selected from "Variable a to h", "LED" and "Right and left motor speeds".

At the left field on the second line, how to substitute a value into the Memory Map is selected. By the Arithmetic command, not only a simply predetermined value is set, but four operations of arithmetic, "Add", "Subtract", "Multiply" and "Divide", can be conducted for the current value displayed in the selected item of the Memory Map .



For the specific operation of the Arithmetic command, a program example will be described below. This program is to "calculate the average of the right and left sensor values." The average of the right and left sensor values" is obtained by the calculation of "(right sensor value + left sensor value) ÷ 2", but by the Arithmetic command, the whole calculation cannot be conducted at one time. Furthermore, as explained in the description about Memory Map, values in the items for "right and left sensors" cannot be changed. Therefore, **substitute the right and left sensor values into the Variable a and Variable b.** Then, **add the variable b to the variable a to obtain the value for (right sensor value + left sensor value)**". Finally, **divide the variable a by 2 and obtain the average.**

1. Substitute the left sensor value into Variable a.

Variable a =
 Constant 0
 Left sensor

2. Substitute the right sensor value into Variable b.

Variable b =
 Constant 0
 Right sensor

3. Add Variable b to Variable a.

Variable a +=
 Constant 0
 Variable b

4. Divide Variable a by 2.

Variable a /=
 Constant 2
 Left sensor



A value substituted into the item for Variable is an integer of "0 to 255" and a decimal or negative value cannot be substituted. When the calculation result is a decimal value, discard all digits to the right of the decimal point. When it is a negative value, such as -1 or smaller, it is corrected to 0. When it is 256 or larger, it is corrected to 255.

6-4. Branch command for advanced-level users

When a checkmark is placed in the checkbox for "Use the Arithmetic block", the setting items for the branch command will be slightly changed. With this function enabled, it is found that the setting items for the branch command in the Setting Area are as shown in the figure below.

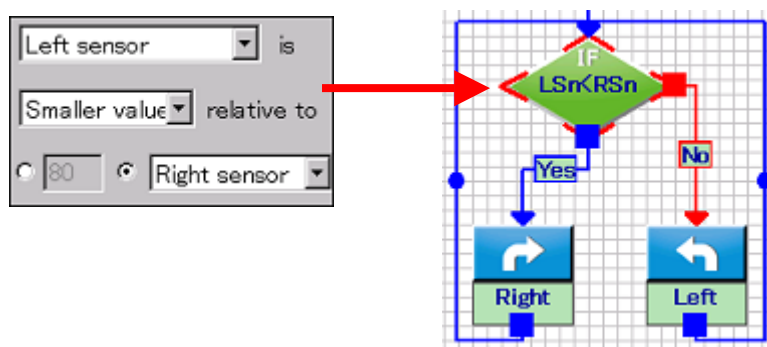
The major changes are that "on the first line, the items other than sensors can be selected" and "for a comparison value, the items in the Memory Map other than a value of 0 to 255 can be selected."

Set an item in Memory Map as a reference for comparison. A value other than a sensor value can be set.

Left sensor is
Smaller value relative to
80 Left sensor

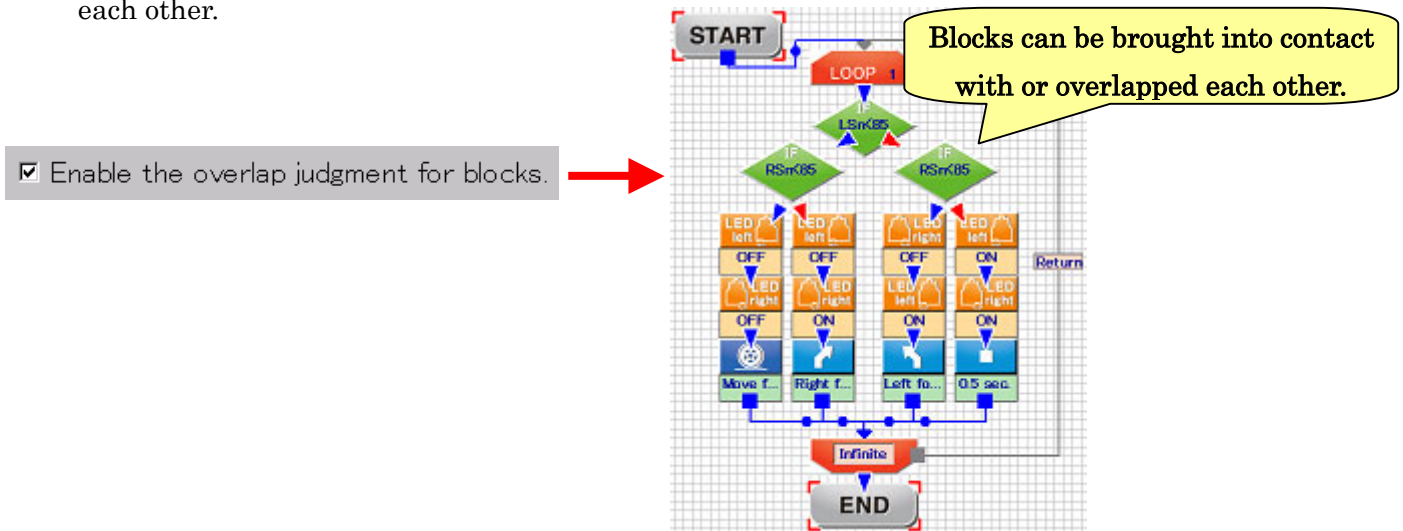
An item in Memory Map can be selected as a value to be compared.

In the case of a programs shown in the figure below, for example, comparison is made between the right and left sensor values and the motor having a larger sensor value will run.



6-5. Judgment for overlap of blocks

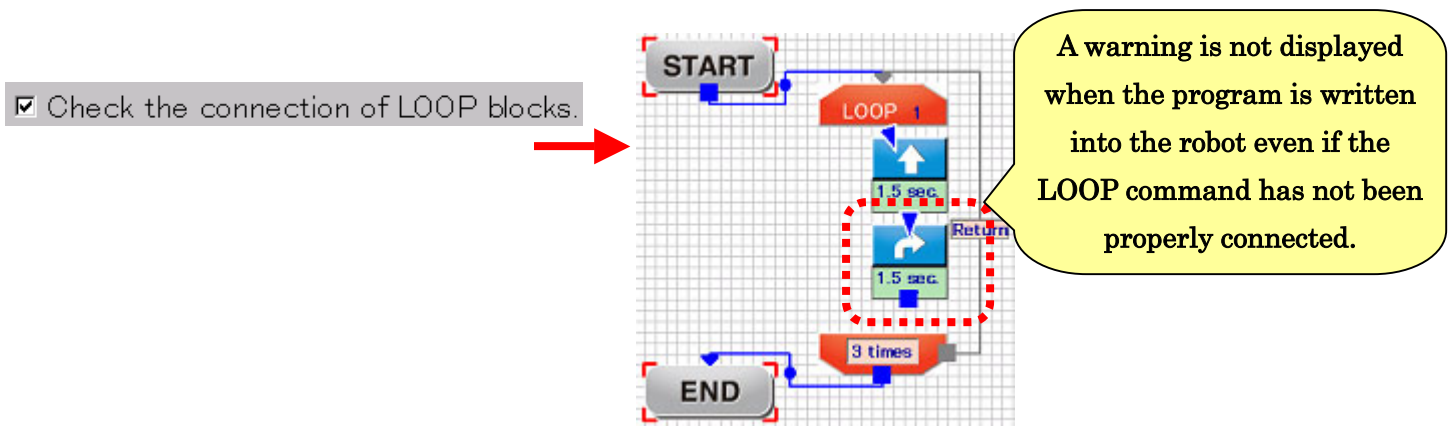
The item “Enable the overlap judgment for blocks” is selected to prevent action blocks in the Program Area for overlapping each other. In the checkbox for this item, a checkmark has been initially placed. By removing the checkmark, action blocks can be overlapped each other.



When a program is created using many action blocks, disable this function to facilitate programming.

6-6. Check of LOOP connection

The item “Check the connection of LOOP blocks” is selected to display a warning when a program to be written has wrong connection of arrows for LOOP action blocks in the Program Area. In the checkbox for this item, a checkmark has been initially placed. By removing the checkmark, a warning is not displayed when the program is written into the robot even if the arrows for LOOP command have not been properly connected.

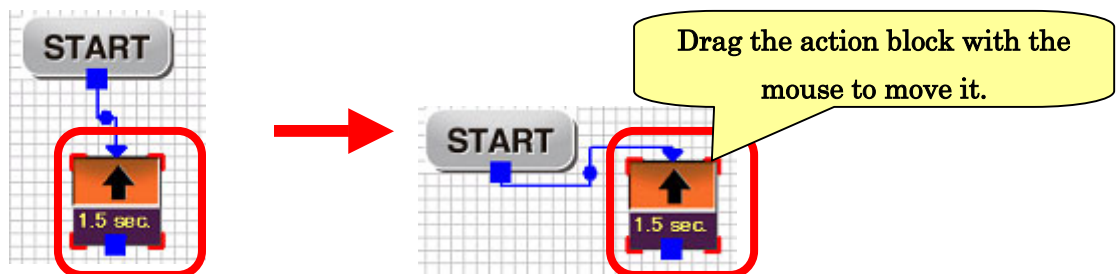


7. Convenient functions for programming

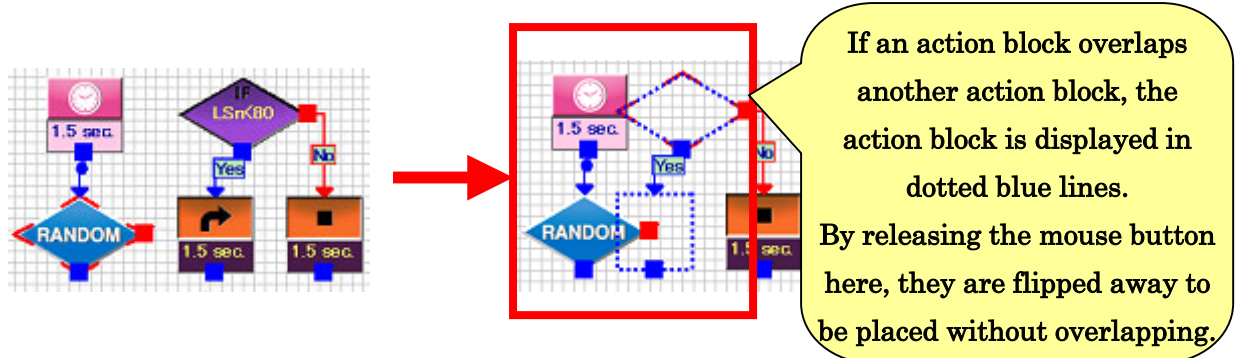
The operation procedures of this software such as move, delete and copy of action block as well as functions on the menu/toolbar will be explained below. Some of the functions were already explained, but it is recommended to review them here.

7-1. Move of action block

To move an action block, click the action block with the mouse and drag it. The action block moves along the grid on the background.

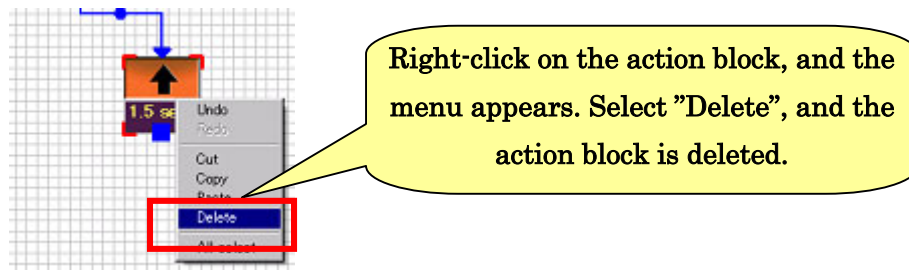


If an action block overlaps another action block during moving, the action block is displayed in dotted blue lines as shown in the figure below. Action blocks cannot be placed as they overlap each other. Therefore, when the mouse button is released, the blocks are flipped away and positioned so that they don't overlap each other. (However, on the edges of Program Area or if many action blocks exist in the Program Area, overlap of blocks may occur.)



7-2. Delete of action block

To delete an action block from the Program Area, put the mouse cursor on the action block and right-click. Then, the menu appears. From the menu, select "Delete".

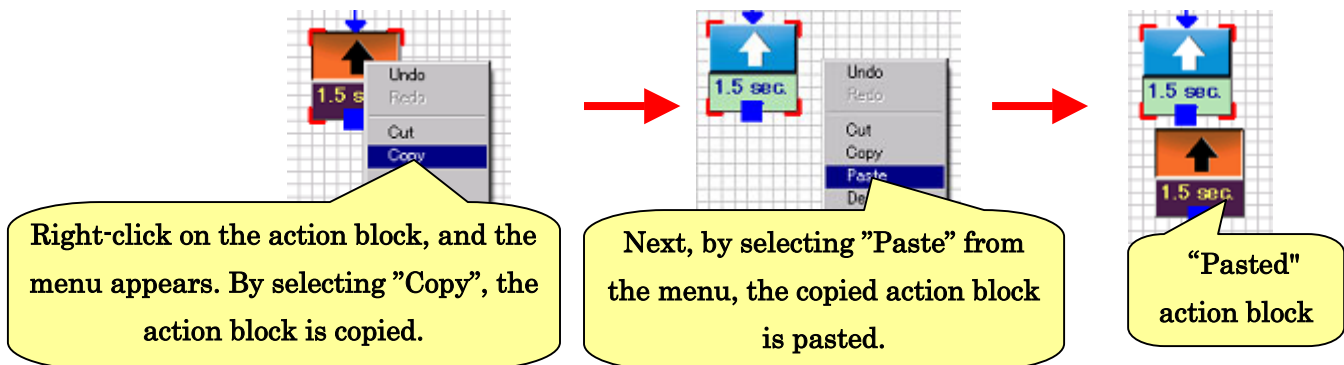


Or, you may delete an action block by clicking the action block to be deleted and pressing the Delete key on the keyboard.

7-3. Copy/Cut/Paste of action block

Right-click on an action block, and the menu appears. By selecting "Copy" from the menu, the action block can be copied. By selecting "Cut" from the menu, the action block can be deleted after copy.

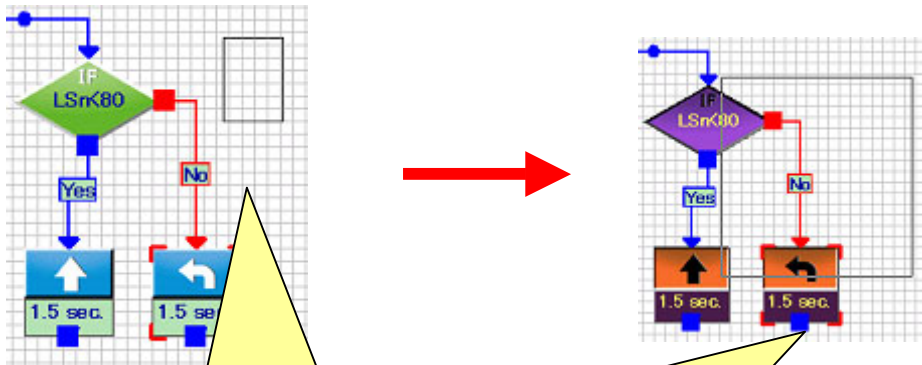
After copying an action block, right-click on the Program Area, and the menu appears. By selecting "Paste" from the menu, the copied action block can be pasted in the Program Area.



* For the action blocks "Start" and "Exit", Copy/Cut/Paste cannot be used. Up to 7 action blocks for LOOP can be pasted.

7-4. Selection of action block

By clicking the area without action block in the Program Area and dragging, a square line is displayed as shown in the figure below. When this square overlaps action blocks, the action blocks are displayed in different colors just as they are clicked. The state where an action block is displayed in a different color is called “in the state of being selected”. All action blocks in the state of being selected can be moved/deleted/copied/cut/pasted at one time.

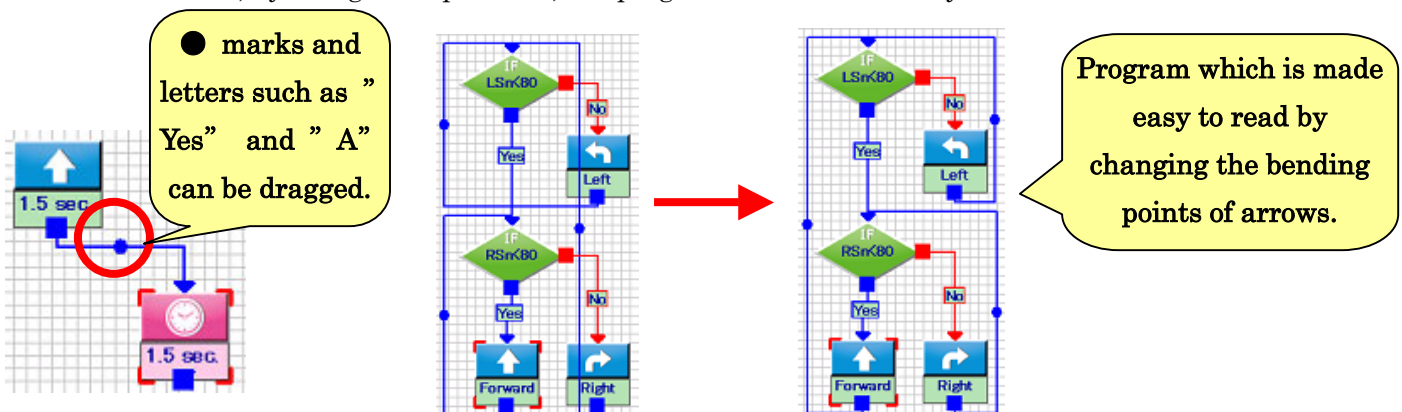


By clicking the area without action block in the Program Area and dragging, a square line is displayed.

When the square and action blocks are overlapped, the action block will be “in the state of being selected”. All action blocks in the state of being selected can be moved/deleted/copied/cut/pasted together at one time.

7-5. Operation of arrow path




By clicking a black circle at a midpoint or a center of the letters such as “Yes”, “No”, “A”, “B” and “Return” on an arrow path and dragging, the bending position of the arrow path can be changed. When arrows and action blocks overlap and the program is hard to read, by using this operation, the program can be made easy to read.








7-6. Explanation of menu/toolbar

Items in the menu and functions of the buttons in the toolbar of this software will be explained. For the items in the menu and buttons in the toolbar having the same functions, the icons in the toolbar are indicated in parentheses after the item names.


• File

- **New creation** () Abandon the currently created program and start creation of a new program.
- **Open** () Load the program saved in a file.
- **Save** () Save the currently created program. If the program has no name, give a name to the program and save it.
- **Save as** Save the currently created program with a different name.
- **Exit** End this software.

• Edit

- **Undo** () Return the currently created program to the last state.
- **Redo** () Bring the program subjected to "Undo" forward to the next state.
- **Cut** () Copy the currently selected action block in the Program Area and delete it.
- **Copy** () Copy the currently selected action block in the Program Area.
- **Paste** () Paste the action block cut by "Cut" and copied by "Copy" in the Program Area.
- **Delete** Delete the currently selected action block in the Program Area.
- **All select** Select all action blocks that are currently in the Program Area.

- **Program**

- **Program write** () . . . Write the currently created program in the robot.
- **Program load** Load the program recorded in the robot into the Program Area.

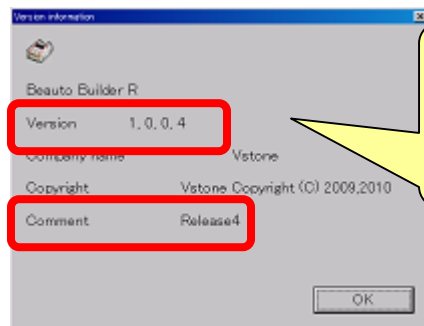
Save the screen shot of program area . . . Save the image of recent program area The image is JPEG format

- **Setting**

- **Setting of motor** Adjust the wheel speed of the robot.
- **Setting of motor(Simulator)** . . . Adjust the wheel speed of the simulator robot.
- **Setting of functions for advanced-level users** . . . Open the dialogue for setting on whether the functions for advanced-level users will be used.
- **Display of Memory Map** Display the Memory Map on the screen.

- **Help**

- **Version information** Copyright notice and version information about this software can be checked. By clicking this item, the dialogue shown below is displayed.



The number for "Version" and "Release No." for "Comment" indicate the version of this software.

- **Update of firmware** . . Update the program written into the robot body. A new version of firmware is released on the Product Support Page. Update of firmware rewrites the program in your robot to the new version.

7-7. Explanation of shortcut key

By conducting a specific key operation on the keyboard of PC, some items of the menu introduced in this document can be called up. This function is called a shortcut key. It should be noted that the operating procedure differs depending on which side of the screen the mouse cursor is placed when a shortcut key is entered, "Program Area" side or "Icon Area/Sensor Area/Setting Area" side. For details, refer to the description below.

●When the mouse cursor is placed on the Program Area side:

- Ctrl (control key) +A . . . Same as "Edit" on the menu → " Select All"
- Ctrl + C Same as "Edit" on the menu → " Copy"
- Ctrl + V Same as "Edit" on the menu → " Paste"
- Ctrl + X Same as "Edit" on the menu→" Cut"
- Delete キー Same as "Edit" on the menu" →" Delete"
- Ctrl + Q Same as " File" on the menu→" Exit"
- Ctrl + Z Same as "Edit" on the menu→" Undo"
- Ctrl + Y Same as "Edit" on the menu→" Redo"

●When the mouse cursor is placed on the Icon Area/Sensor Area/Setting Area side:

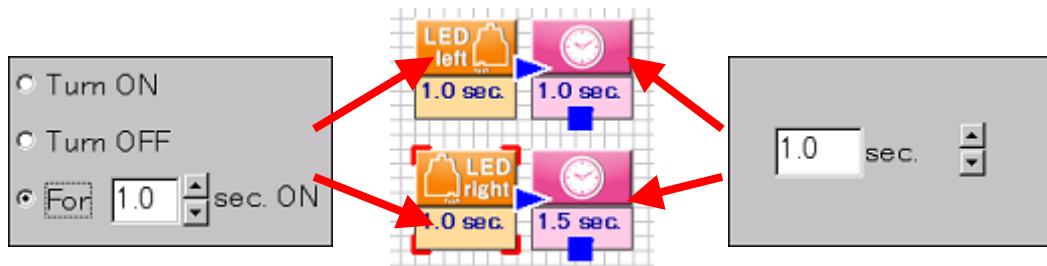
- Ctrl + A Select all texts for the value setting.
- Ctrl + C Select all the selected texts for the value setting.
- Ctrl + V Paste the copied text to the value setting item.
- Ctrl + X Cut the selected value setting text.
- Ctrl + Z Return the value setting text to the last status.

8. Others

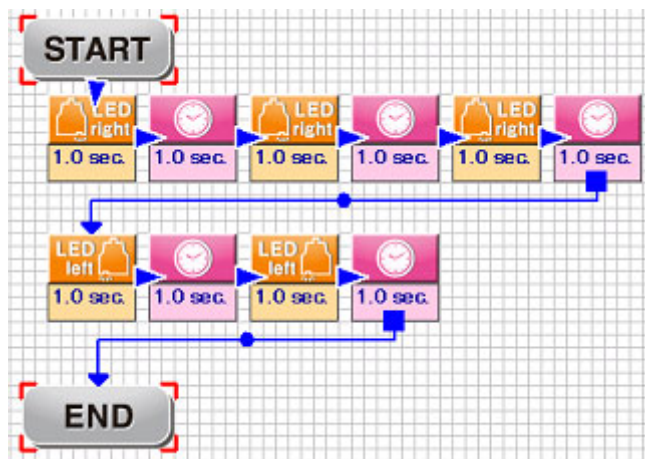
8-1. Answers to the exercises

- Program in which the right LED flashes twice at intervals of 1 second and the left LED flashes three times at intervals of 1 second.

First, create a program in which the right and left LEDs "flash once at intervals of 1 second". The commands to be used and settings are as shown in the figure below.

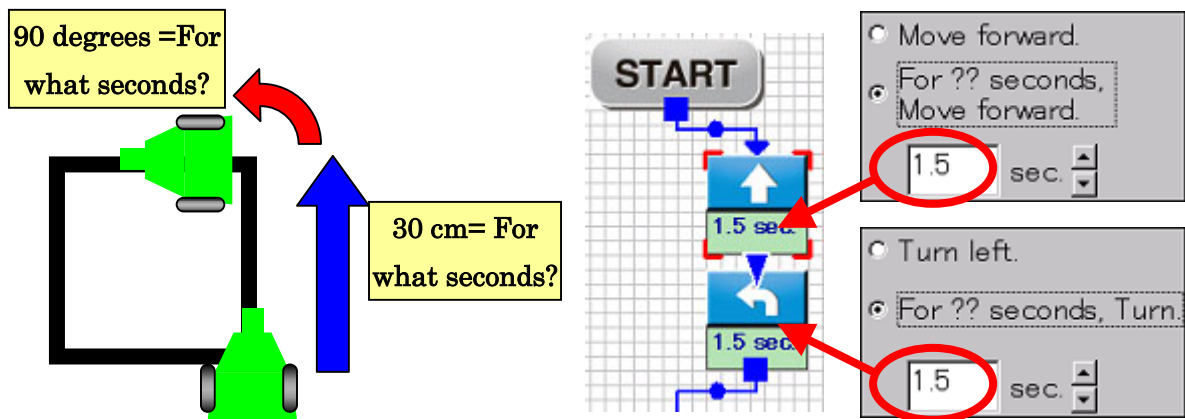


Next, make a program in which the right LED flashes three times at intervals of 1 second and make a program in which the left LED flashes twice. Then, connect the programs as shown in the figure below. Now, the program is completed.

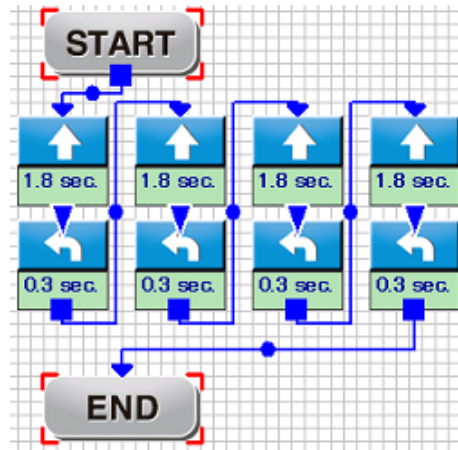


- Write a square having each side of 30 cm in length on paper, and create a program in which the robot traces around the square.

First, for robot currently in use, check to see "for what seconds it takes the robot to move forward by 30 cm" and "for what seconds it takes the robot to turn 90 degrees." Create a program as shown in the figure below. Then, change the number of seconds many times and execute the program to examine a correct number of seconds.

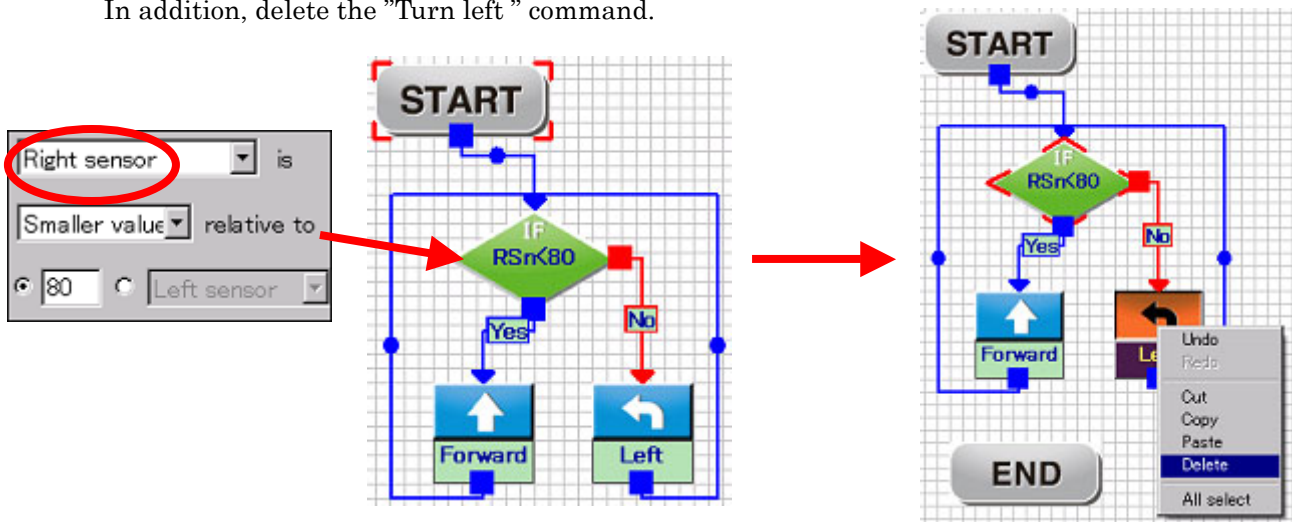


After the setting for moving forward by 30 cm and that for turning by 90 degrees are completed, place four action blocks side by side, respectively, and connect them as shown in the figure below. Now, the program is completed. With the program, however, the robot may not move properly due to a subtle error. In that case, adjust the number of seconds.

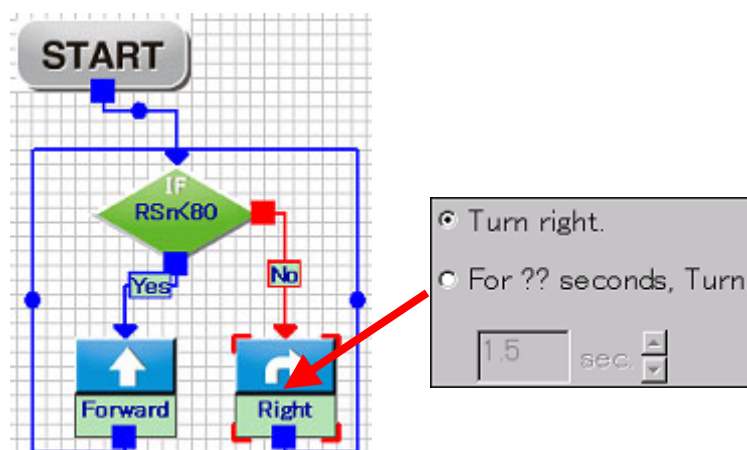


• **Program for Line Trace using the right sensor**

First, change the setting at the branch command for the "left sensor" to the "right sensor". In addition, delete the "Turn left" command.



Instead of the deleted "Turn left", add "Turn right" and connect as shown in the figure below. Now, the program is completed. For the "Turn right" command, select the setting item "Turn right".



8-2. Load of program from the robot body

On this software, the program written into the robot body can be loaded and displayed in the Program Area. To load the program from the robot body, click "Program" on the menu → "Program load". After click, a confirmation message that the currently edited program should be saved or loaded is displayed. If it may be loaded without any problem, click "Yes". After load of the program is completed, the loaded program is displayed in the Program Area. At the same time, the right and left wheel speeds are loaded and the setting is updated.

Concerning load of a program, it should be noted that completely the same content as the written content cannot be displayed. Other precautions are described below. Take due caution in execution of loading.

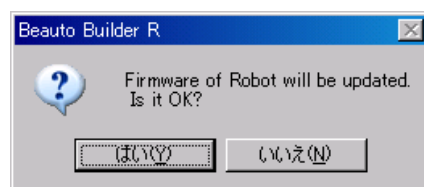
- **Coordinate data of action block is not recorded into the robot. Therefore, the coordinate of action block in the loaded program may be quite different from that in the written program.**
- **Action block having no arrow connected to any block will be automatically connected to "Exit".**
- **If time is not designated for an action block for Move or LED, setting of the number of seconds itself is not written into the robot, and the number of seconds may be different from that in the written program.**
- **For a branch or arithmetic action block, either a constant or Memory Map is selected on the second line. The data of the item not selected is not written into the robot and the setting of a constant or Memory Map may be different from that in the written program.**
- **For a LOOP action block, if "Repeat endlessly (infinite number)" has been set, "Repeat endlessly (infinite number)" is not set in the loaded program. However, after the LOOP action ends, the arrow will be automatically connected to the start of LOOP. (Actually, "Repeat endlessly (infinite number)" is set.)**
- **Depending on connection of action blocks or settings, multiple action blocks may be integrated as one action.**
- **Move action block for which speed or turning rate can be set may be replaced with "Move forward" or "Turn left or right".**
- **The above-mentioned differences exist, but when a loaded program is written back into the robot again, completely the same program is executed.**

8-3. Update the firmware of the robot

In the robot body, the most basic program called "firmware" has been written in order to communicate with PC or execute a created program. The firmware is not lost even if a program is written into the robot from this software. In the future, if the firmware is updated to improve debug or functions of the robot body, download a new firmware from the Support Page and write into the robot, thereby updating your robot, too.

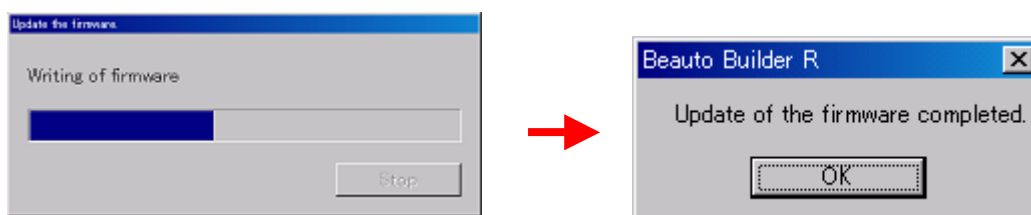
To update the firmware of the robot body, take the following procedure.

First, connect PC and the robot. Then, click " Help" →" Update of firmware" on the menu. The confirmation message for update appears. When you want to update, click " Yes" .



After clicking "Yes", the window for selecting a file for firmware is displayed. Select the file for the downloaded firmware and click "Open".

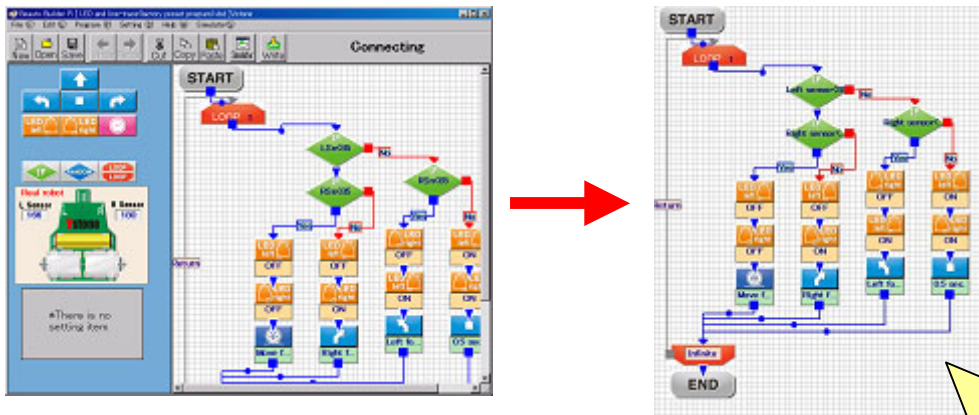
When the file is opened, update of the firmware automatically starts. Wait till update is completed just as writing of a program. When the message shown in the right figure below is displayed, update of the firmware is completed.



If update of the firmware ends in failure, re-execute update.

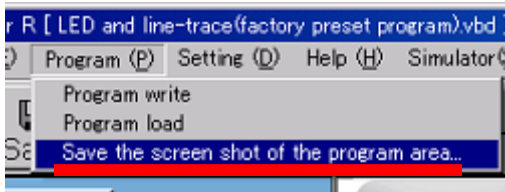
8-4. Save screenshot of Program area

Click " Program" →" Save screenshot of Program area" on the menu, you can save the image of recent program area The image is JPEG format. This function is useful, when making documentations about your program.



You can save the image of all program area.

Click " Save screenshot of Program area " on the menu. The file saving window will appear. Enter the file name for the image and click " Save"



Click " Save screenshot of Program area " on the menu. And Enter the file name for the image and click " Save"

The Saving image is exactly same to displayed view on PC. Please note that the arrow that is out of program area will not fall inside the image

8-5.Q&A

If any problem occurs during use of this software, refer to the following Q&A items to find the cause and a measure. If the problem is not solved even after the measure described below is taken or there is no symptom that corresponds to the problem in the following items, please contact the address indicated at the end of this document.

Q: The wheels of the robot body do not rotate.

A1: The battery of the robot may be weak. If the wheels rotate feebly, there is a high possibility that the battery is weak. Replace the battery with a new one and check to see whether the wheels rotate properly.

A2: While the robot and PC are connected, the program may not be executed properly. Disconnect the robot from PC and execute the program.

A3: There is a possibility that in the setting for the motor, the wheel speed has been set at a very low speed. Referring to “3. *Set for the motor*”, set a higher speed for the wheel speed.

A4: For the “Move” action blocks, the time has been set at a very short time. Therefore, the robot may stop before it is accelerated fully. Re-set the time at a longer time.

A5: If a heavy load is applied to one of the wheels such as a case where the wheel is stuck with something, the power of the whole robot may be reduced. As a result, the wheels may stop. Check the wheels for getting stuck.

A6: If the motor has been installed to the robot body in a wrong direction or the parts have not been inserted properly, power may not be supplied to the motor. Check those points.

Q: When the program is written into the robot, the message “Now reconnecting to Robot...” is continuously displayed and writing ends in failure.

A1: After the robot is connected to PC and disconnected from PC several times, it may take a long time for PC to recognize the robot depending on the PC environment. Disconnect the robot from PC and connect it again or restart up PC before writing.

A2: If communication with the robot is interrupted during writing of a program, the above symptom occurs. Re-connect the robot to PC completely, and execute writing without touching or moving the robot.

Q: Although a program is not being executed, the motor runs.

A: If the robot is connected to PC with the power switch ON, a program may be started suddenly or the motor may start running. Before connecting the robot to PC, be sure to turn off the power.

Q: When the robot body is operating, the power is shut off or reset frequently.

A1: The battery of the robot body may be weak. Replace the battery with a new one and check to see if the problem is resolved.

A2: A heavy load may be applied to the robot wheels, reducing the power of the whole robot body. Check the wheels for any problem such as getting stuck with something.

Q: Smoke, spark or odor was given out from the robot body.

A1: There may be a short in the wiring for the robot. Immediately, turn off the power switch, disconnect the communication cable and remove the battery. Then, contact at the address indicated at the end of this document.

Q: Even if the robot is connected to PC, no communication is established.

A1: If the robot has been connected with the robot body being reversed, it has not been inserted fully or it has been inserted obliquely, the robot may not properly get in contact with PC, resulting in communication error. Check to see if the robot has been connected properly.

A2: If writing of a program into the robot ended in failure or this software is forcefully terminated due to any cause, PC may not properly recognize the robot. In this case, disconnect the robot from PC and then, re-connect. Check to see if the problem is resolved.

Q: Communication between PC and the robot is sometimes shut off suddenly.

A1: If the robot is moved roughly while the robot is connected to PC, contact of the robot with PC may become unstable and communication may be shut off. While the robot is connected to PC, do not move the robot too roughly.

A2: If the robot has not been inserted fully or it has been inserted obliquely, contact of the robot with PC may become unstable and communication may be shut off. Check to see if the robot has been connected properly.

Q: When a program is written into the robot, the error message "Program data size is too large" is displayed and the program cannot be written.

A: If the data size of a created program exceeds the capacity of the robot, the program cannot be written into the robot body. Data of action blocks having no connection of arrow in the program are written into the robot body. Therefore, delete unnecessary blocks to reduce the program data size within the capacity.

Q: For an Arithmetic command, a value in the Memory Map is not changed to an intended value.

A1: Only an integer of 0 to 255 can be entered in each item of Memory Map. The resultant value of arithmetic is automatically corrected to an integer within the range. If the resultant value of arithmetic is 256 or larger, it is corrected to 255 and if it is a negative value, it is corrected to 0. If the resultant value of a division is a decimal value, all digits to the right of the decimal point are discarded.

Q: When activating software, the error message["Gdiplus.dll Not Found"] is displayed.

A: This error message will be displayed, when gdiplus.dll is not installed on Windows2000 PC or Old PC. In this case, open the folder "gdiplus" in the folder "BeautoBuilderR", copy the file "gdiplus.dll" and put it in the folder that contains the file "cl_edit_r.exe".

Q: Virtual robot doesn't move in the simulator window.

A3: Motor speed [simulator] might be too much slow. Change the motor speed settings.

A4: The time for the robot to move might be short. Please make the time long.

Q: Virtual robot goes through the goal, but it has not recognized as goal.

A: In some course, passing goal is not recognized as a goal. There would be no goal, virtual robot did not stop at the goal.

●Contact

Vstone Co., Ltd.

2-15-28, Mitejima, Nishiyodogawa-ku, Osaka-city, Osaka 555-0012 Japan

Tel:06-4808-8701 Fax:06-4808-8702

e-mail: infodesk@vstone.co.jp

Product Support URL:

http://www.vstone.co.jp/english/products/beauto_racer/download.html

URL: <http://www.vstone.co.jp/english/>

(2011/3/4)