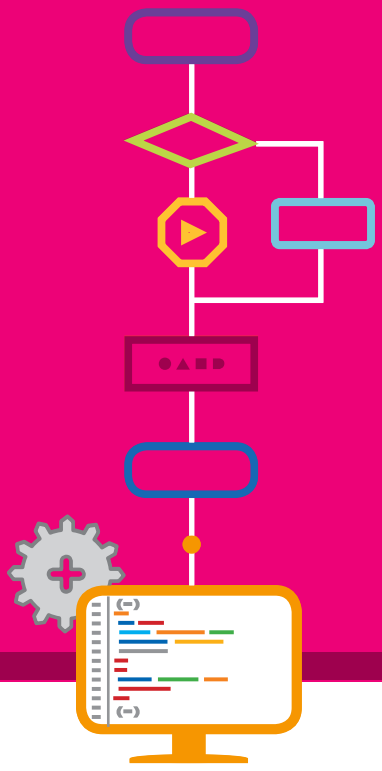
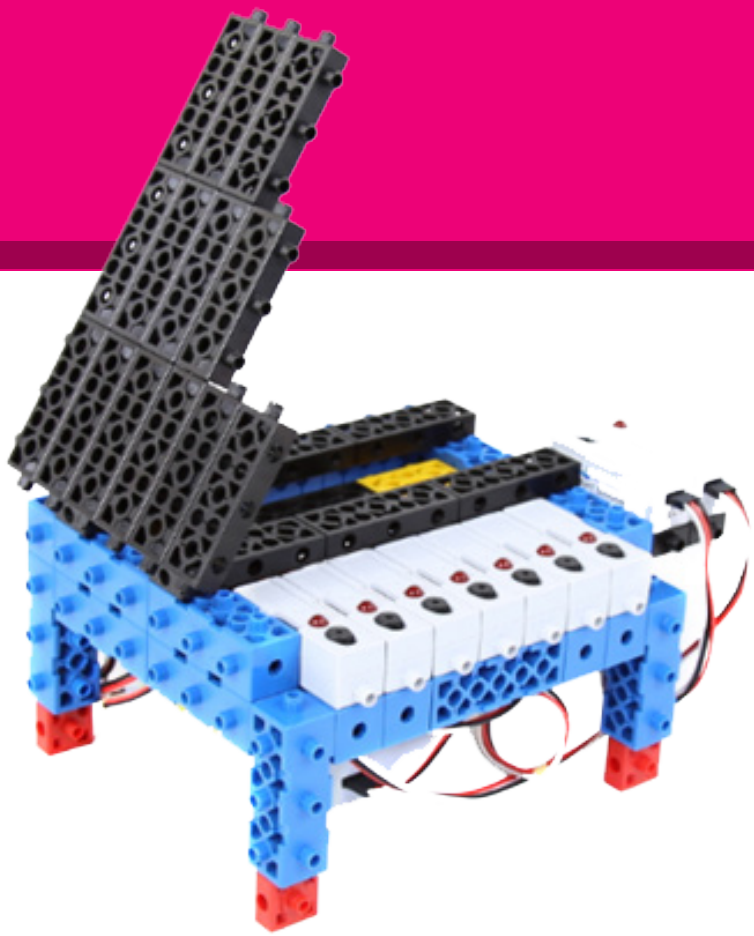
SCRATCH CODING KIT

Logic boost

CODING CLASS

CLASS 4





Educational Toy

ROBOTORI

CLASS 4

Before getting started

" The best programmers are not marginally better than merely good ones. They are an order-of-magnitude better, measured by whatever standard: conceptual creativity, speed, ingenuity of design, or problem-solving ability."

Randall E. Stross

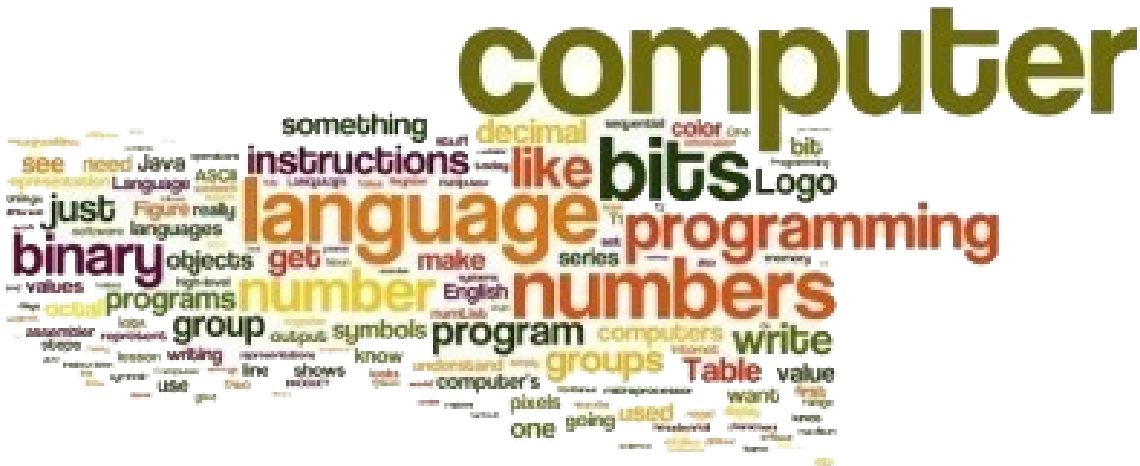


Table of contents

CLASS 4

To begin

Welcome to Class 4!

Now you have leared all contents of coding robotics from classes 1 through 4! Great job!
Now you will begin the intensive course on Scratch. If the previous courses were easy for you, now you will begin the relatively harder courses.

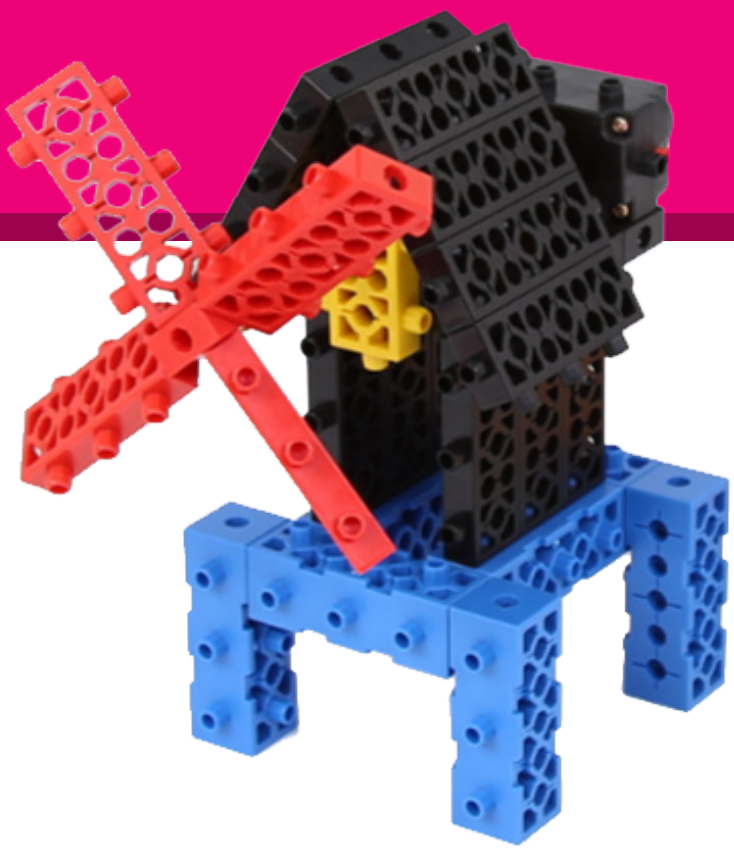
Feel free to express your creativities!

SCRATCH CODING KIT

Logic Boost

Wait until

LESSON 1



CLASS 4

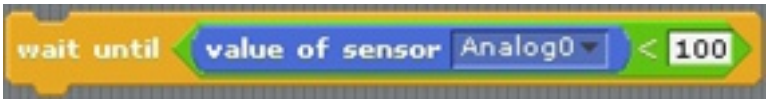
Introducing Blocks

introducing Waiting Block

The “waiting” block changes the progam by waiting by a certain number of seconds and makes it easier for coding.

The “waiting” block pauses the program until it satisfies the specific condition

The “waiting” block in the figure above makes the program wait until an infrared sensor connects to Digital INPUT Port 2 is pressed.



The “waiting” block from the figure above makes the program wait until the measurement of an infrared sensor is connected to Analog INPUT Port 0 is less than 100.

Tip : You can also use the “repeating” block with the function of the “waiting” block. Think about how you can use this!

Coding Activities

Creative Advanced Preparation Activity

Think about how you can use the "waiting" block to move the robot that has a DC motor.
On this page we will learn how to develop some creative thinking skills.

Just like the mini-windmill we made in Class 1, create a robot that uses only one DC motor and use the space below to simply draw out how you can move the robot around.

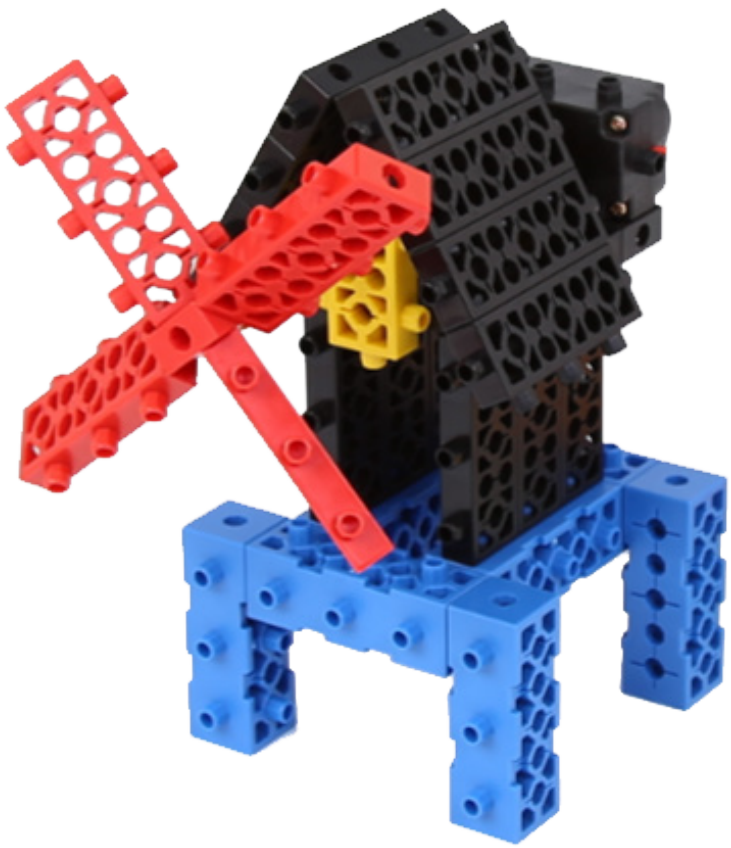
Making DC Motor Robot

CLASS 4

Coding Activities

How to Make a Mini-Windmill

First, we need to make our old friend, Mini-Windmill. Now that some time has passed, you should be able to make this really quickly!

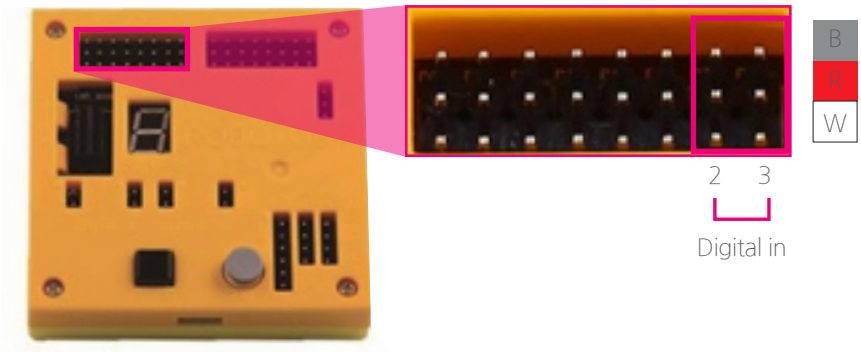


Coding Activities

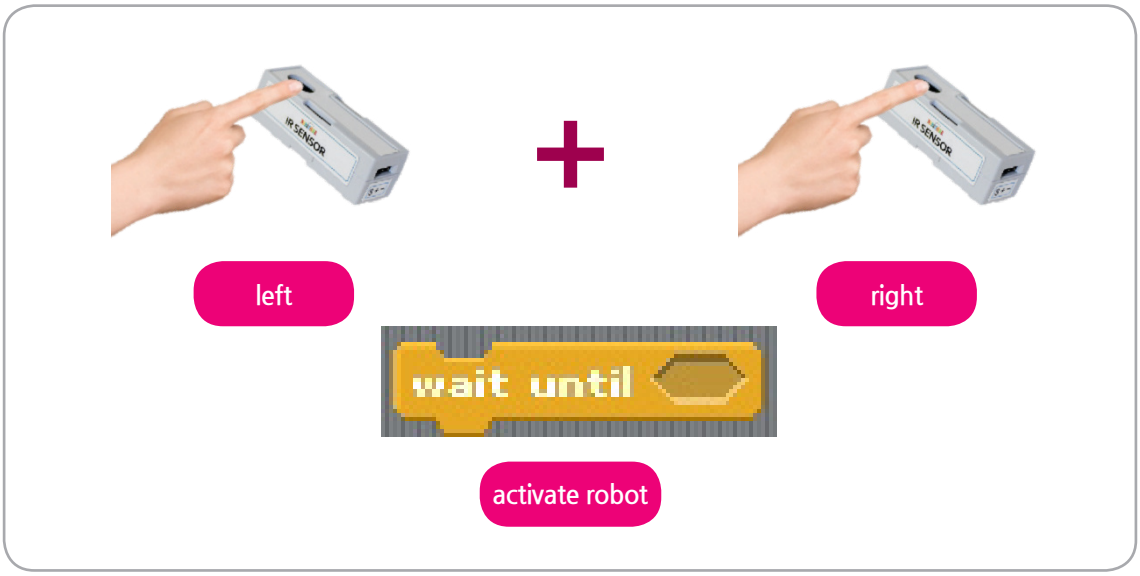
Using the “Waiting” block

Now you need two infrared sensors. Plug the infrared sensors into the digital INPUT port of your main cell. Then, plug in your model, or your mini-windmill, into the left port of the DC motor of the main cell.

Infrared sensors must be plugged into the 7th and 8th port of the main cell’s digital INPUT...



You are going to use the waiting block to make a robot that is only activated when both of the infrared sensors are pressed.



CLASS 4

Coding Activities

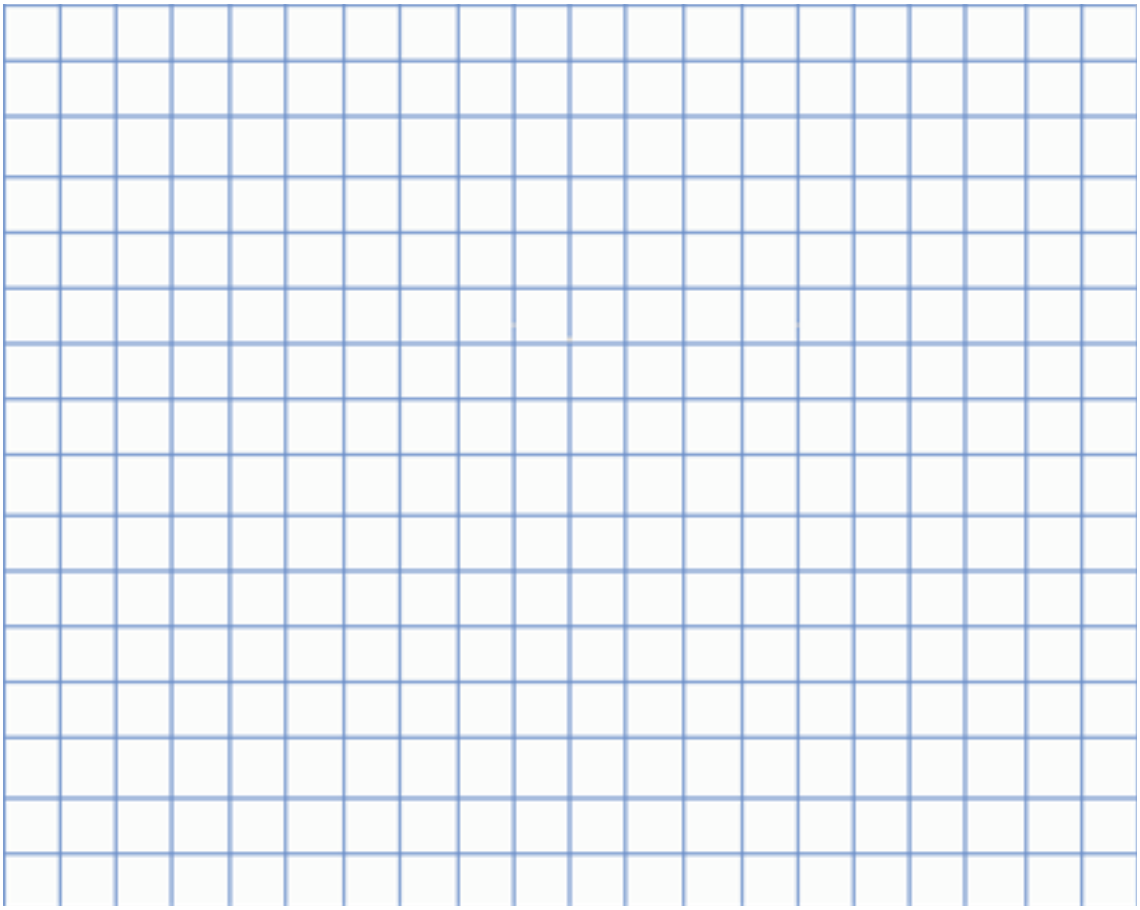
Using “Waiting” Block

Now let’s brainstorm several ideas.

Then, show how your program appears in Scratch on the flowchart. The most important thing to remember is that the goal of this program is to have a DC motor that only works when both infrared sensors are pressed.

Hint: Use the “waiting” block and the “if~ or” block.

Start Program

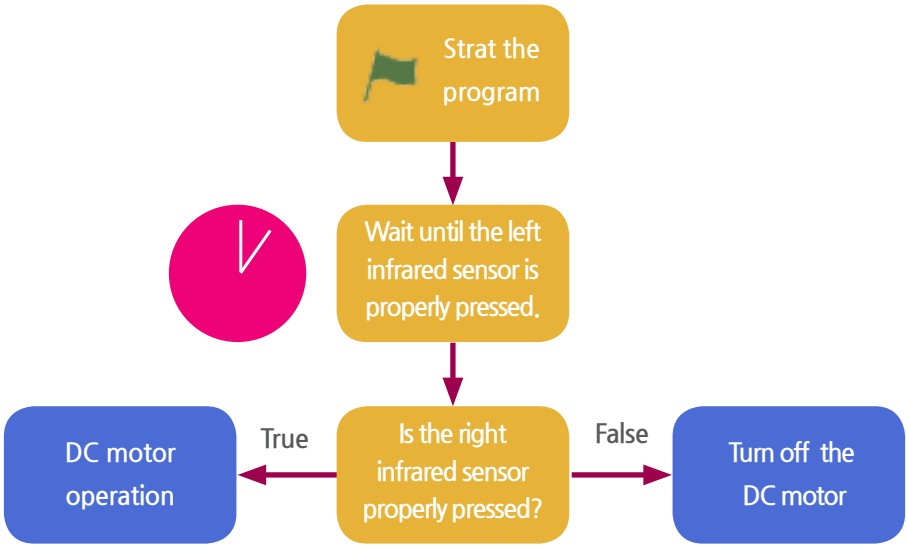


Coding Activities

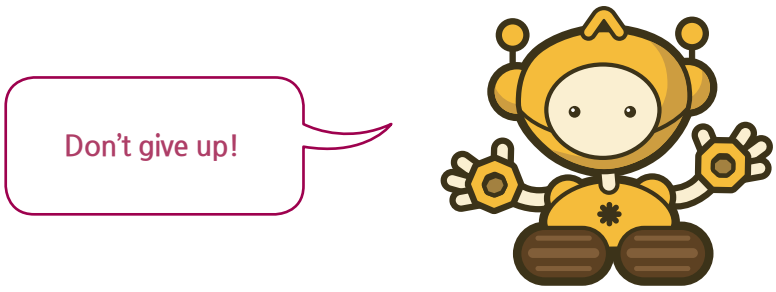
Using “Waiting” Block

Are you still having problems?

Don’t worry! Check out how the code you have to make appears. Let’s make it so that the system below repeats.



Within the program, the “waiting” block will not activate the program until the left infrared sensor is pressed. Therefore, if the left infrared sensor is not pressed, the “waiting” block will pause the program.

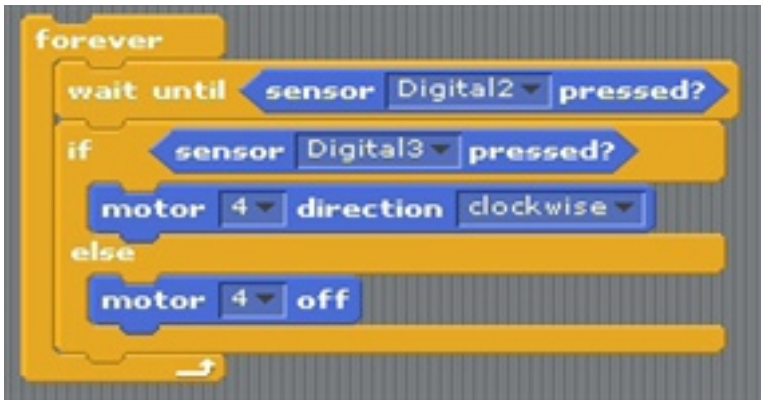


CLASS 4

Coding Activities

Using “Waiting” Block

Take a look at the scratch code below. The “waiting” block pauses the program until the left sensor is pressed. Once the sensor is pressed, the program will continuously run and move over to the “if~ or” block..



Topic

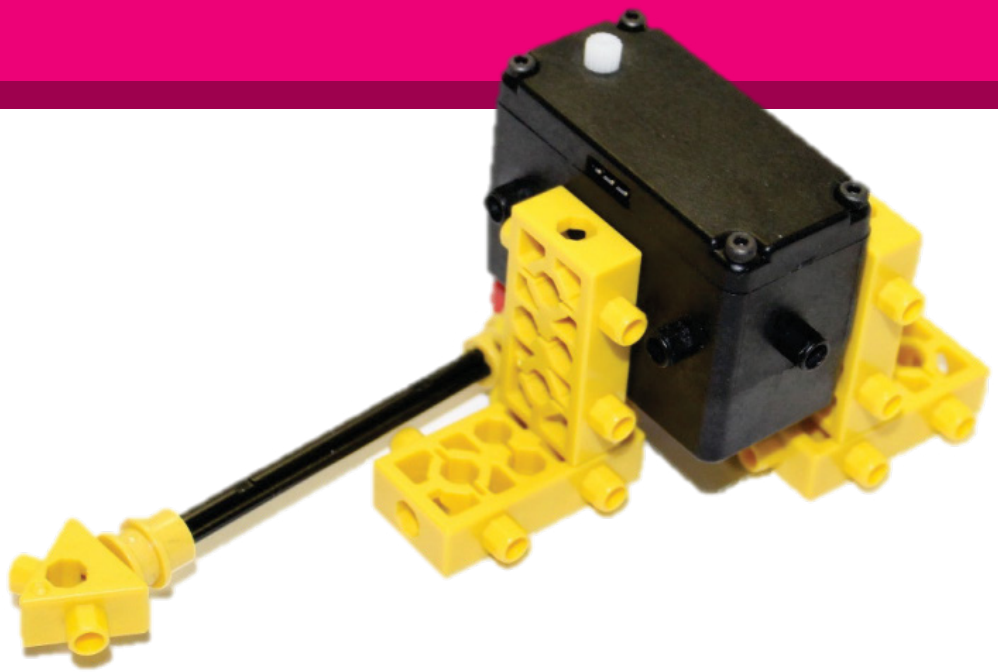
The “Waiting” block is a hard concept to understand. Think of it as a door that blocks the activation of the program. In the coding section, the “waiting” block obstructs the program from being activated before the infrared sensor is pressed. If the sensor is pressed, the guard opens the door and allows you to activate the “if~ or” block.

SCRATCH CODING KIT

Logic Boost

Basics of Variables

LESSON 2



CLASS 4

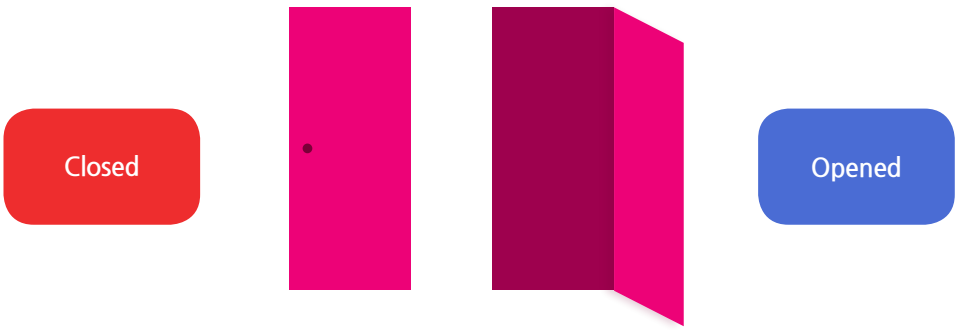
Introduction

What are variables?

Variables are used to save the information of certain numbers or values.

1. The value of analog & digital sensor

2. Character



Variables based coding activity

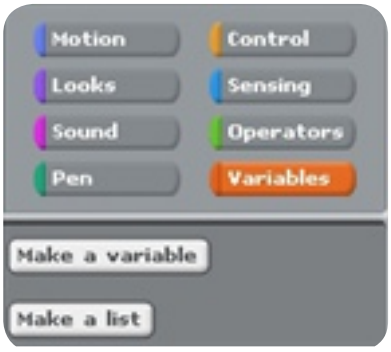
Why are variables important?

Without variables, there is no way for computers to 'remember' something.
If, you are not able to remember the simple things, then your life will be very limited. Giving your robot the ability to remember is more complicated, but also interactive. Most importantly, it opens up the opportunity for more interesting designs.

Now let's use variables in the scratch.

Making Variables

To make variables, you must click on the variable tab and click "Create Variables".



Then, you can see the pop-up window show up. This pop-up window is where you decide the name of the variable. Input your new variable name as "counter." Input "counter" in the input window and press (OK).

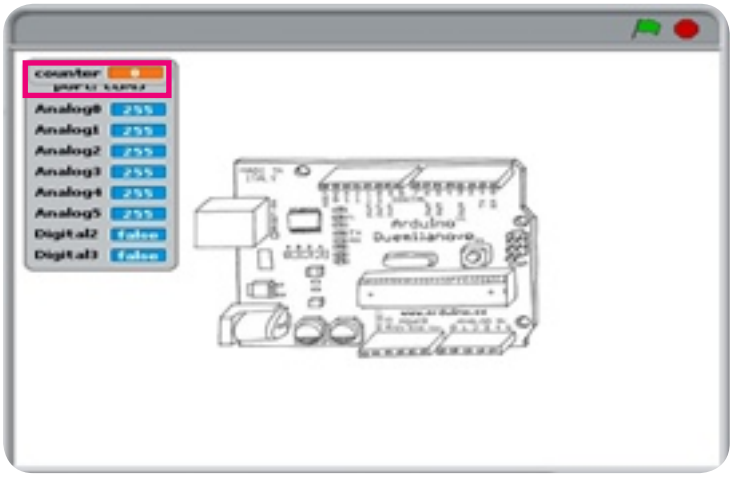
CLASS 4

Variables based coding activity

Making Variables

After you have finished making variables, you will see some changes in your scratch screen.
First, you will be able to see the new orange blocks related to the “counter” variables in your palette menu.

On the right side of your screen, the “counter” variable is shown on the stage window.

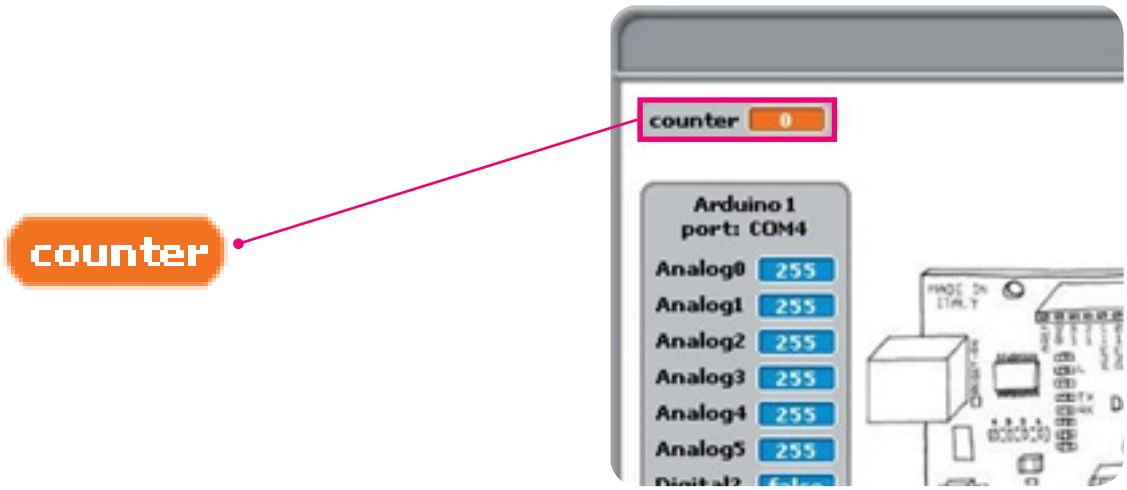


Variables based coding activity

Variable Blocks

Now we will learn about the new variable block we saw previously. First, look at the three blocks below.

The first block shows the actual value that is held in our variable, “counter.” The variable is displaying a certain number and that number can be found in the stage window. Currently, the variable “counter” is displaying “0.”



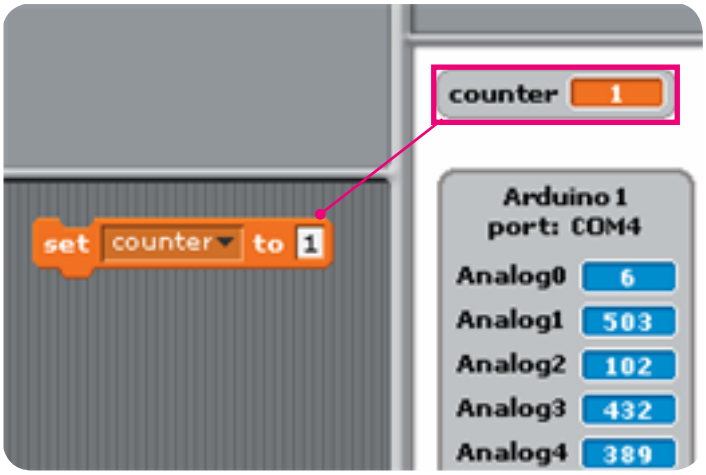
CLASS 4

Variables based coding activity

Variable blocks

The “Save 0 at counter” block changes the saved value to the value or character that the user desires.

Click on the “0” and change it to “1”



You will see that the variable value of “counter” has changed from “0” to “1”!

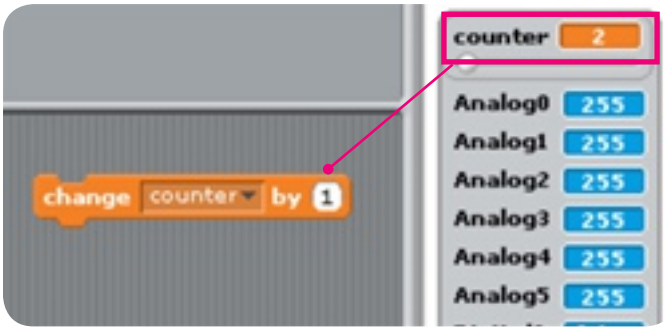
Also, you can change the values to letters. Double click the white area and input a word like “true” or “false.”
You will see that now your variable vlue has changed to “true” or “false”!

Variables based coding activity

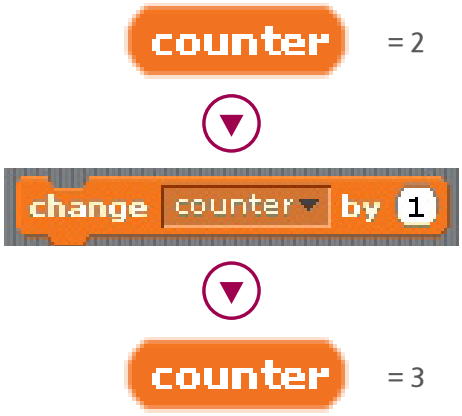
Variable blocks

"Accumulate 1 at counter" block causes the value of variabes to change. Double click the block show below!

In the white blank in the block above, input "1" and double click the block.



This block adds "1" to the variable value that was originally saved, and changed "1" to "2"!
If you continue to press on this block , you will be increasing the value by 1 continuously.



CLASS 4

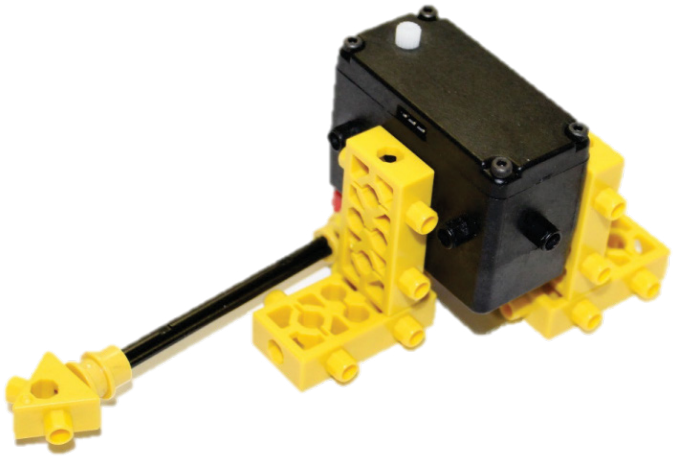
Variables based coding activity

Variable blocks

What should we do to increase the value of the"counter" variable by 2?

Double click the white area and input "2." Now you will see that the value of the variable will start increasing by 2 as you double click!

Do you remember this analog meter? Use the servo-motor that is used in this model to learn the functions of variables!



Material list

								
Servo x 1	Mini 2 x 4	Motor connector x 2	Triangle x 1	Link x 2	A 64 x 1	Servo motor x 1	Battery Case x 1	Mainboard x 1

Variables based coding activity

How to make an analog meter

1


x 2

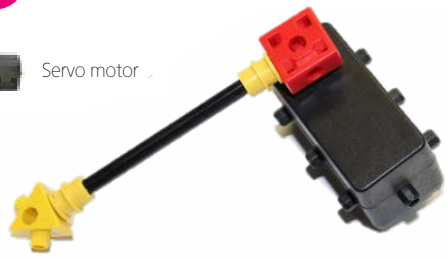

x 1


x 1


x 1

2 connect them at this angle!


Servo motor



3


x 4

4

Complete

CLASS 4

Time Watch Coding Activities

Time Watch

Let's make a time watch using the meter we made before. A time watch is a device that records time in seconds.

Time watches were used in sports-related events before the digital time watch was invented.
Time watches were also used for the aircraft speed measurement test.
The picture on the right: Pocket time watch made in the Sweden Mountain town La Chaux-de-Fonds

Now you will make a simple program that uses variables.
Make a program that uses variables to record how many seconds have passed and display what you have recorded with the analog meter.

Time Watch Coding Activities

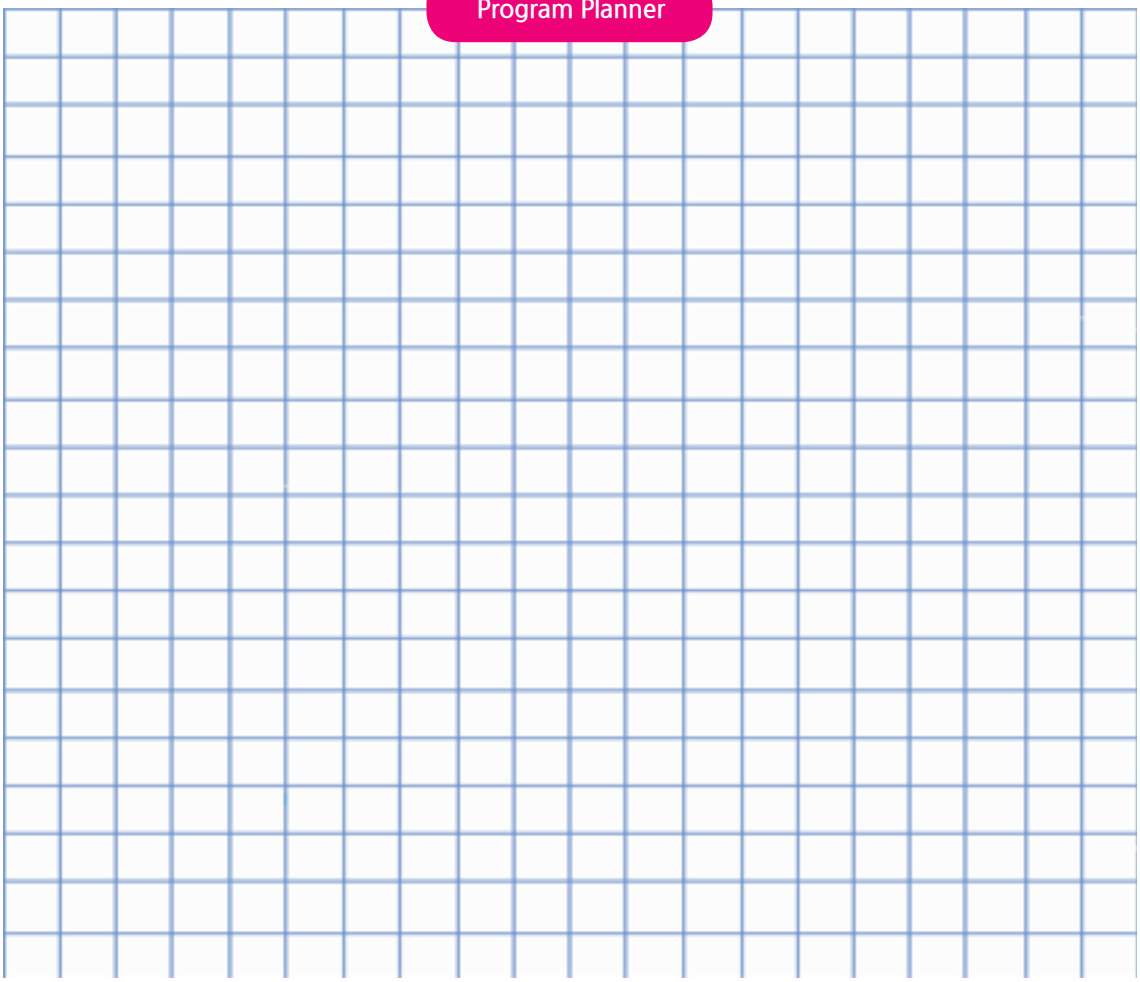
Time Watch

You have enough knowledge to make your own program!

Once again, You need to make a program that records time. Think about what variables and variable blocks you will have to use to make this program.

Use the space below to plan out how you will make your program!

Program Planner

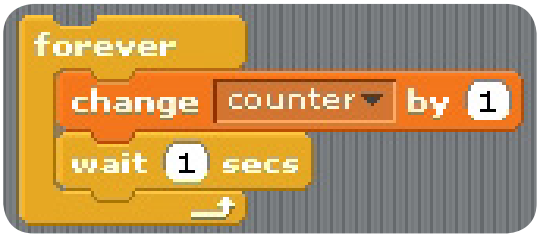


CLASS 4

Time Watch Coding Activities

Time Watch

Was it hard to make the last program? Now, we will divide it into a parts and work through it.
First, you need to make the variable increase by 1 for every second. Refer to the code below!

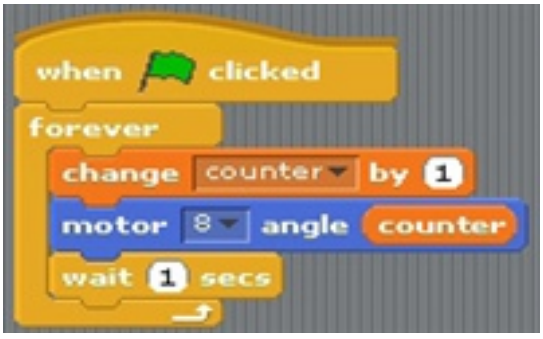


The code on the left adds "1" to the variable every second. Now, the variable you have made works as a timer! If you remember what we've learned previously, you know that you can replace the value of servo motor angle with the value that can be expressed as a number.

Replace the "counter" variable with the value of the servo motor angle.

Don't forget to do infinite repeat!

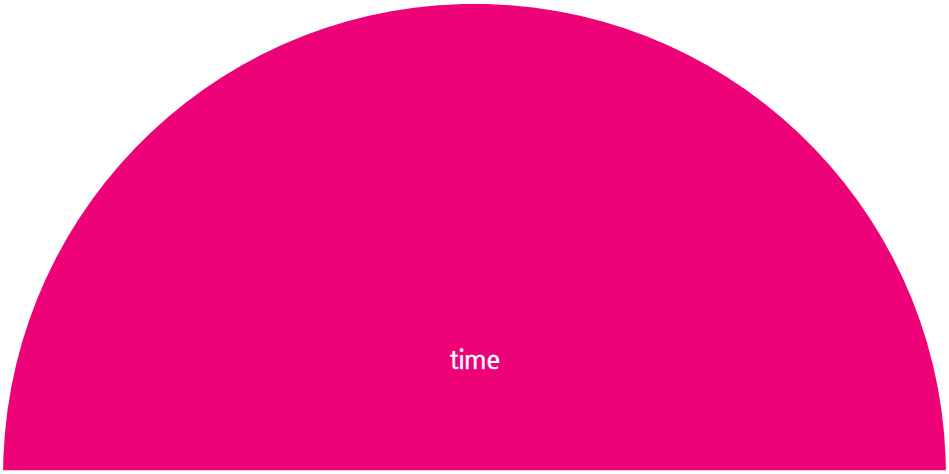
Combine these codes



Time Watch Coding Activities

Time Watch

Using the space shown below, line up the bottom line of the time watch you have made. Activate the program!



Line up the bottom line of the servo motor into the blank

The second hand of the time watch you have made should move towards the right very slowly.
If not, you need to adjust the direction of the servo motor. We learned how to adjust the servo motor in Class 3.

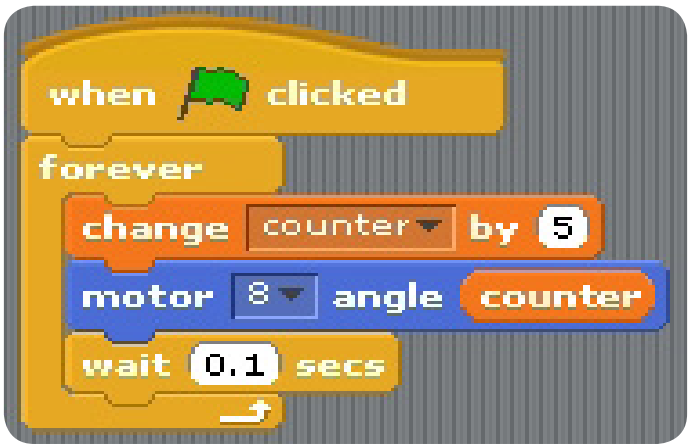
CLASS 4

Time Watch Coding Activities

Time Watch

If you activate the program you've learned previously, you will be able to see the servo motor moving slowly. Now let's think about it. How can you increase the range for the motion of the servo motor?

Change the accumulating value from "1" to "5."
Now, you will be able to see the "counter" variable moving by "5" per 0.1 second.



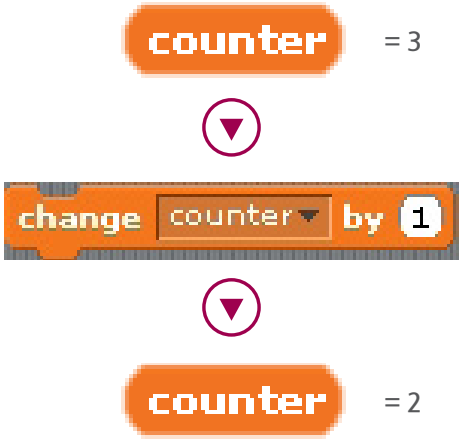
If you activate the program again, you can see the second hand moving with a wider width. Grab a pencil and mark where the second hand points at each second to see the difference in movement.

Time Watch Coding Activities

Challenge

How can you make the time watch to start at the beginning, hit the end and then go back to the beginning? Change the accumulating by "1" to accumulating by "-1" at "counter" variable.

Everytime you double click the "accumulating by (-1) to (counter)" block, you can see that the counter variable decrease by 1!

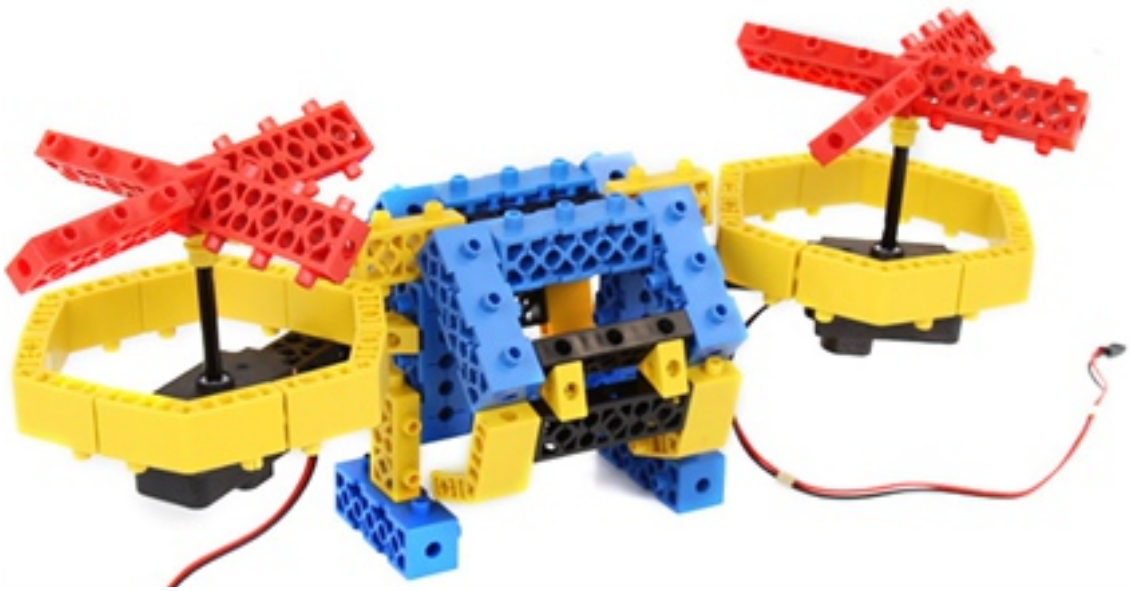


S C R A T C H C O D I N G K I T

Logic Boost

Using variables

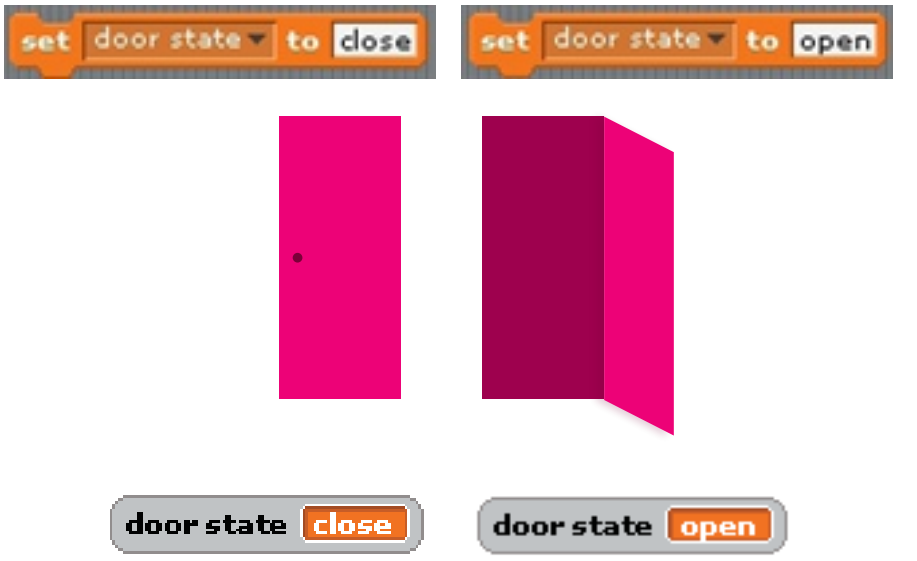
LESSON 3

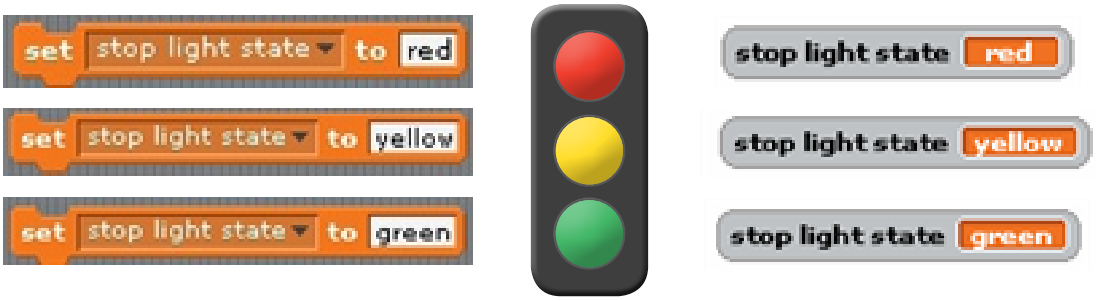


Status indication and variables

Status indicating variables

Previously you learned how the variables can express something numerical such as the sensor value and time. The variables, however, can also be used to save characters or word. With this, variables can mark the "status." "Status," in this case, means the expression of condition or feature of certain object.
For example, opening or closing a door or red, green, and yellow traffic lights are specific condition or feature.



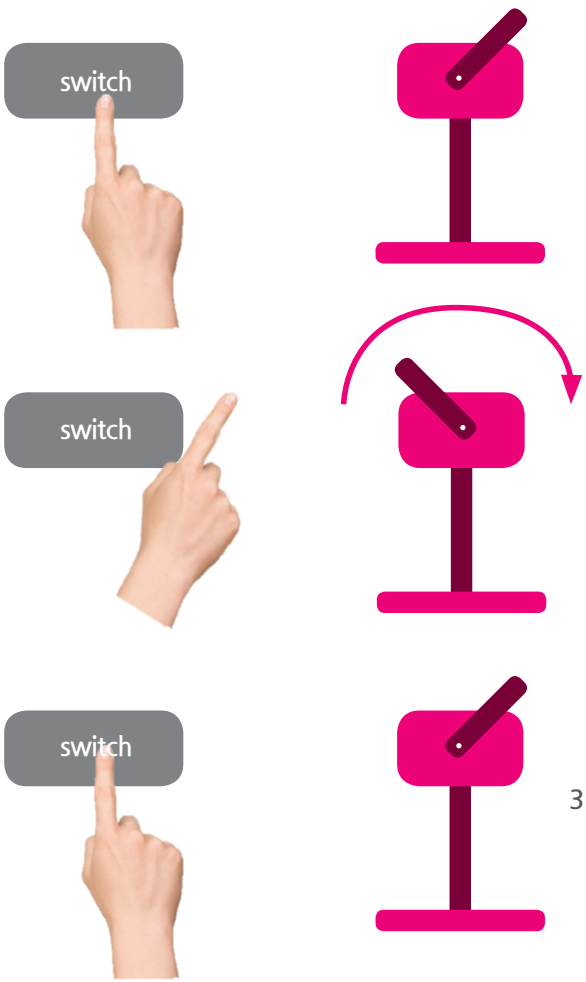


CLASS 4

Status indication and variables

Status indicating variables

Now let's learn how to use the variables to display the status. Please refer to the example below!
Imagine having an electric fan. The electric fan is turned on only if you press the button. If you let go of the button, it will still be on. It only turns on if you press the button again.



1. Press the switch button to turn on the fan

2. The fan continues to work

3. Press the switch button again to turn off the fan.

Making a Robot

Status indicating variables

Imagine yourself being a computer. What should you do to program the function of the fan on the previous page using the variable?
On Scratch, use variables to make a coding program that turns on and off the fan as you press.

Material list

					
Diamond H6 x 3	Diamond V6 x 8	Rubi 7 x 4	Rubi 6 x 4	Rubi 0 x 7	Mini 2 x 4
					
Mini 1 x 2	Curve x 16	Triangle x 6	Motor connector x 2	DC motor x 2	A 45 x 2

CLASS 4

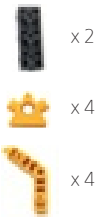
Making a Robot

1

2

Make two identical model

3



Make two identical model

4

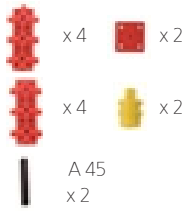
Assemble models you made in step 2 and 3

5

Assemble models you made in step 1 and 4

6

Make two identical model



Making a Robot

7


x 2


x 2


x 2

8 assemble models you made in step 6 and 7

9


x 12

Make two identical model

10

assemble models you made in step 8 and 9

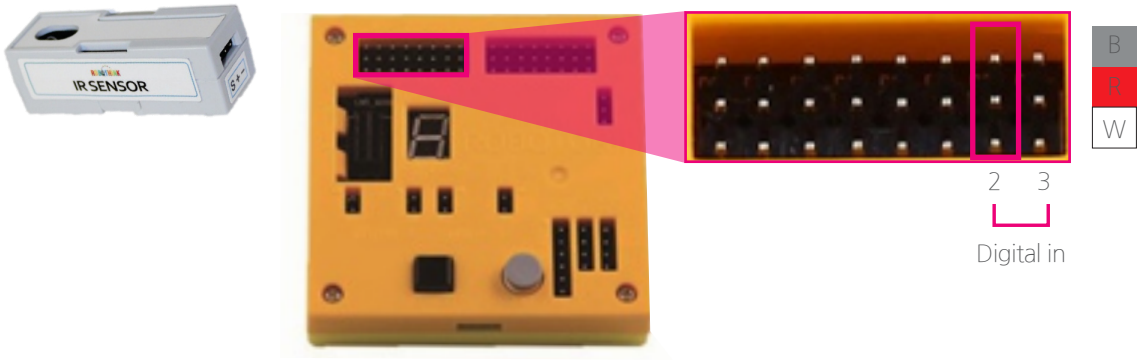
Complete

CLASS 4

Drone Coding Activity

Coding a Drone

Get out one infrared sensor and connect it with Digital INPUT Port 7 of the main cell.
Plug in the left and right DC motor properly to the port.



Think about how to make a program.
On the space provided below, write down several things you need to activate the program.
Hint: Make sure you use variables!

make a program

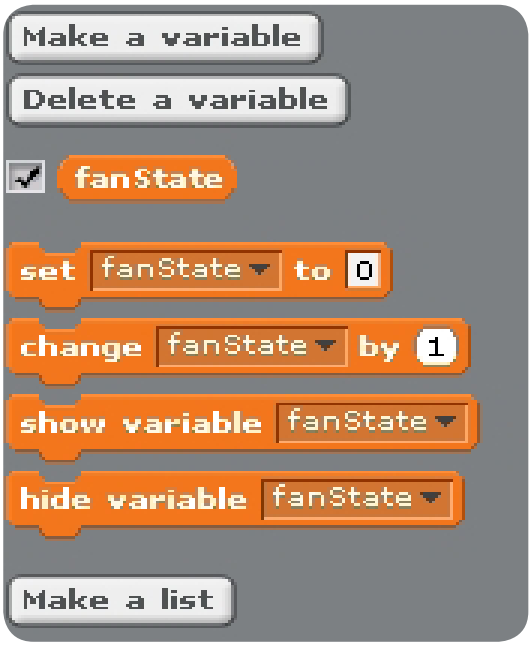
Drone Coding Activity

Coding a Drone

Compare the items below and your own!

- Make a variable that displays if the drone is on or off.
- Whenever an infrared sensor is recognized, a program must be programmed that changes the value of the variable representing the state of the drone to on and off.
- If the value of the variable indicating the state of the drone is on, the DC motor is activated. If it is off, the motor is not operated

Let's take a look at what could possibly be easier to create a variable. Let's make a variable called "fanState". Pay close attention at the upper-case representation of "State".
In computer programming, use uppercase letters when creating names in order to better describe the role of variables and make them easier to read.
Instead of spacing, it is preferred to use uppercase letters to distinguish.



Please note that is is easier to read "fanState" than "fanstate".

CLASS 4

Drone Coding Activity

Coding a Drone

Since we made the needed variables, we now need to learn about the item that has to be programed.

- **Whenever an infrared sensor is recognized, a program must be programmed to change the value of the variable representing the state of the drone to on and off.**

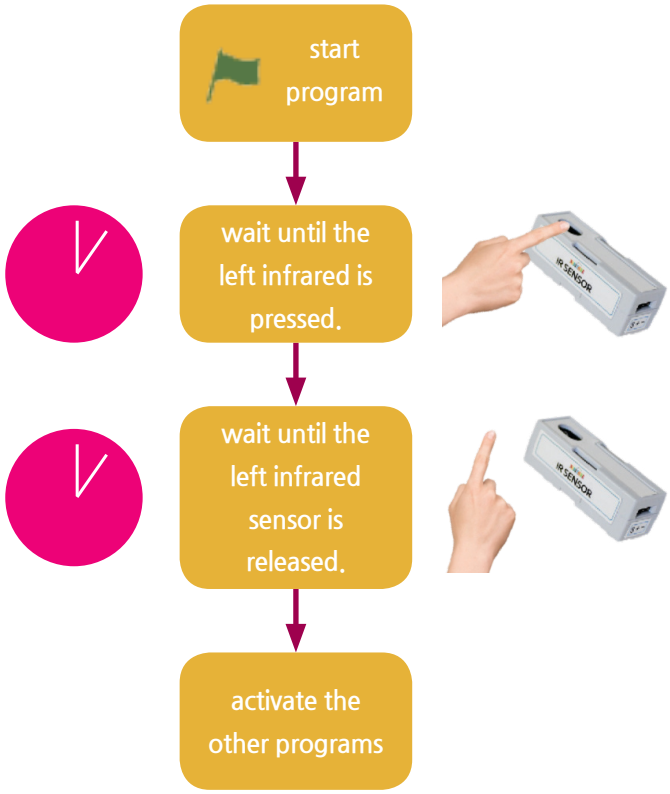
This might look simple but needs a complicated coding. Don't be disappointed even if you can not figure it out. But, please try to brainstorm using the space below on how you could make this program.
Hint: you need the "waiting' and "if~ or" block to make the program.

make a program

Drone Coding Activity

Coding a Drone

Now think of this code as two separate parts. Your program must be activated when the infrared sensor is pressed and released. The "waiting" block is used in this situation.



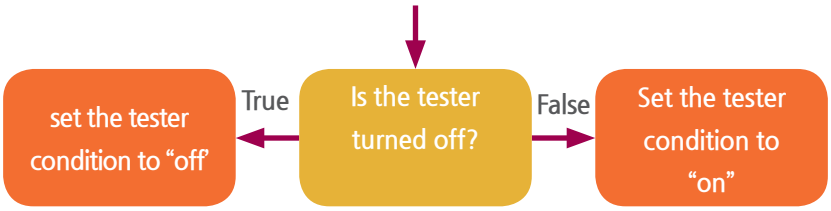
The first waiting block pauses the program until the infrared sensor is pressed. Once it is pressed, the first waiting block moves out of the way.
But, the second waiting block obstructs the program until your hand is released from the infrared sensor.
Check out the codes below!

CLASS 4

Drone Coding Activity

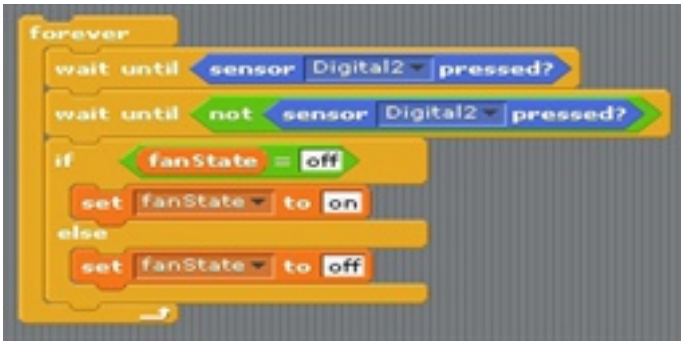
Coding a Drone

Now, you need to change the value of “fanState” to “on” and “off” whenever the infrared sensor is pressed.



As a review, the variable you made needs to have a value of either “on” or “off”. To display the flow chart above, conditional statement such as “if~ or” is necessary. If the condition of tester is already at “off”, you need to change the value of variable “fanState” to “on.” If not, you need to change the variable value to “off.” If you do this, you will be able to see the value saved in the variable “fanState” changing whenever the user presses the infrared sensor. Take a look at the code shown below.

Now combine the codes you made previously. Don’t forget the infinite repeat block!



Drone Coding Activity

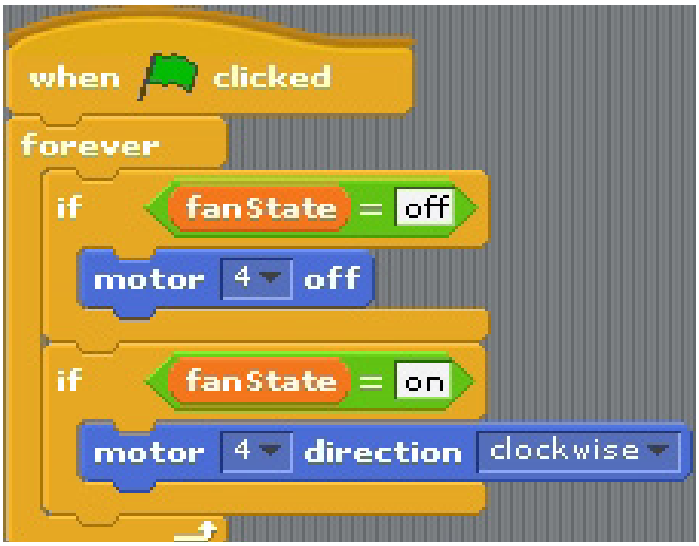
Coding a Drone

Now we are almost done. The last item of program you have to make is:

- If the value of the variable indicating the state of the drone is on, the DC motor is activated. If it is off, the motor is not operated.

This coding is relatively easy! Please go ahead and code the program by yourself.

Hint: Use the conditional statement "If ~ or".



The program above modifies the motor movement depending on whether the variable value of "fanState" is "on" or "off."

Activate the code you have made and watch what happens!

CLASS 4

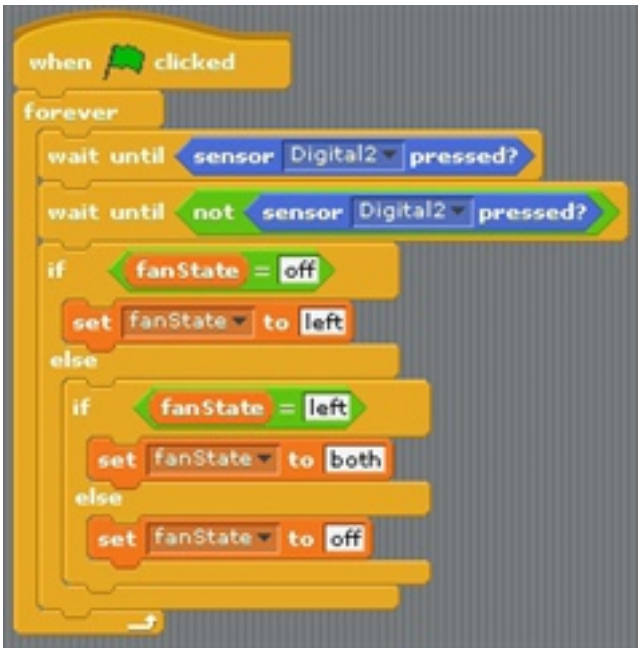
Drone Coding Activity

Challenge

Think of what you would do if you got two motors. What should you do to activate the left motor when the infrared sensor is first pressed? When you press the infrared sensor again, both left and right sensor must be turned on. When you press the sensor for the third time, both motors should pause their work.

Hint: You need to make the condition of the variables from "on" and "off" to "left", "both" and "off."

Also don't forget to use the "if~ or" statement.

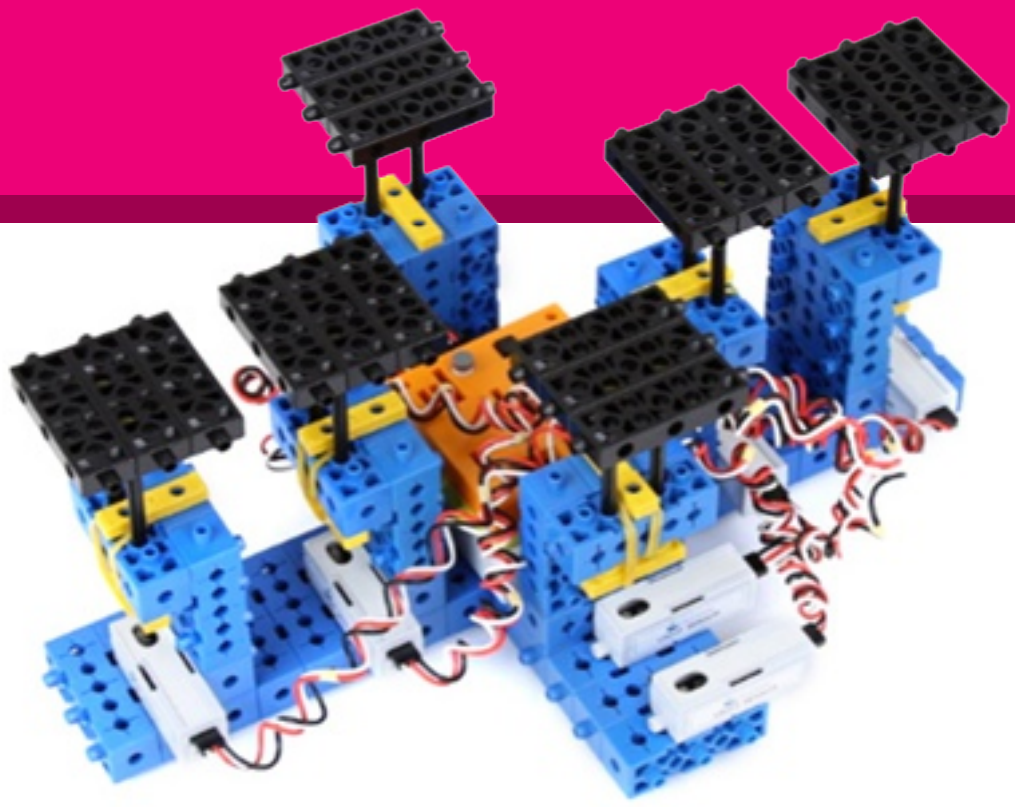


SCRATCH CODING KIT

Logic Boost

Mole Game

LESSON 4



CLASS 4

Introduction

Mole Game

The mole game is a game that allows the user to score by pressing the IR sensor (Infrared sensor) according to the number from the scratch program in the range of 1~6.
In the mole game, you need to use the 7 IR sensors we learned in previous lessons. One IR sensor has the job of activating the game, and the rest have numbers 1~6 assigned. Remember this as you start the coding.

Coding the mole game (Principle)

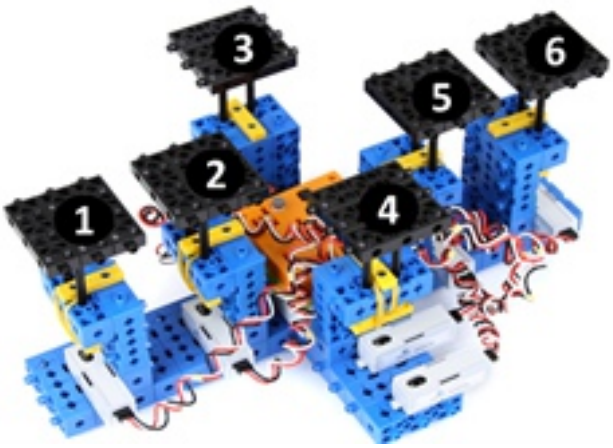
Before we go ahead and start coding, we will go over the things you need to keep in mind when coding. It is really important to have an overall guideline of what to express and how to express them before you start actual coding.

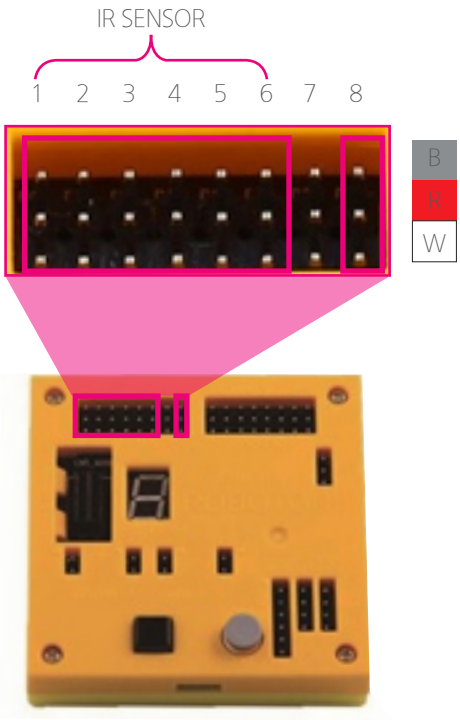
Let's take a quick look at the basic operations we need to implement.

1. Activate the game when the IR sensor plugged in the 8th port is pressed.
2. When the game begin, S4A program of the computer randomly calls a number from 1~6.
3. Score a point when the specific, assigned IR sensor is pressed before the set time. Then continues onto calling next number.
4. If the assigned IR sensor is not pressed, announce that the game is over and stop the script.

Now shall we start making the mole game?

Making a robot





Material list

						
Diamond H8 x 18	Diamond H6 x 6	Diamond V8 x 12	Diamond V6 x 22	Rubi 8 x 6	Rubi 2 x 12	Link x 12
						
A 64 x 12	Mainboard128 x 1	Battery Case x 1	Link x 8	IR Sensor x 7	Motor connector x 24	

CLASS 4

Making a Robot

1


x 12

x 16

2

Make two identical model


x 7


x 1


x 1


x 8

3


x 18


x 6


x 6

Make 6 identical models

4


x 6


x 12

x 12


x 24


x 6

Make 6 identical models and insert the rubber bands

5

Assemble model 3 and 4.

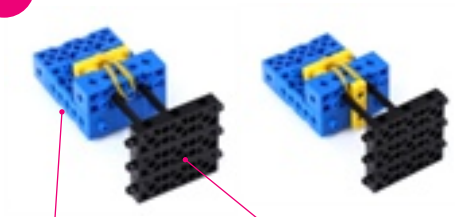


Figure 3

Figure 4

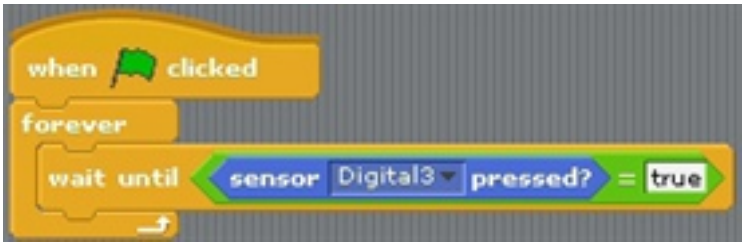

x 6

6

Mole game coding activity

Mole game activation

First, let's make a code that activates the game when the IR sensor connected to digital port 3 is pressed.
The initial sensor value of digital port is "false". The sensor value changes to "true" when the user presses the IR sensor.
Using this do the coding saying "when digital 3 sensor value = true, activate the game".



Mole game activation coding

Make a script by dragging the blocks as shown above.
The first block means that the script will be activated when the flag icon is clicked. The 'infinite repeat' block below that means that it will infinitely repeat to check the script in this block.

There is 'waiting' block inside the 'infinite repeat' block. As we know, this block waits until certain condition is met, and that condition will be "digital 3 sensor value = true" in this case. Which means, when the IR sensor of digital port 3 is pressed and digital 3 sensor value becomes true, it will move away from this block and go on to the next direction.

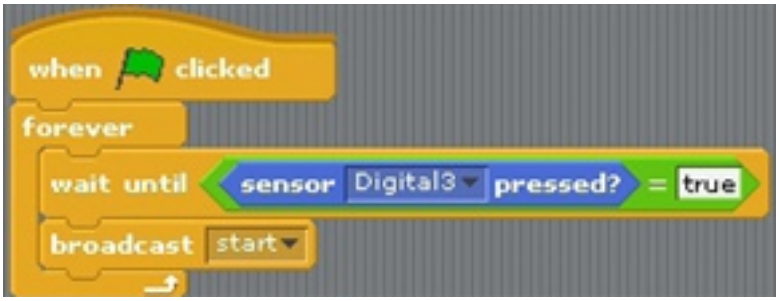
Then the coding above means "when the flag icon is pressed, I will wait until the IR sensor connected to the digital port 3 is pressed".

CLASS 4

Mole game coding activity

Mole game activation coding

Once the IR sensor connected to digital port 3 is pressed, the signal announcing the start of the game must be sent. At this moment, we use the “Broadcast” block. “Broadcast” in Scratch means it’s sending a signal. If you send a signal, you must receive it, right?



Broadcast the mole game

The figure above is the script with ‘broadcast’ block added. Now it waits until the digital 3 sensor value becomes true and then activate the ‘broadcast’ block.

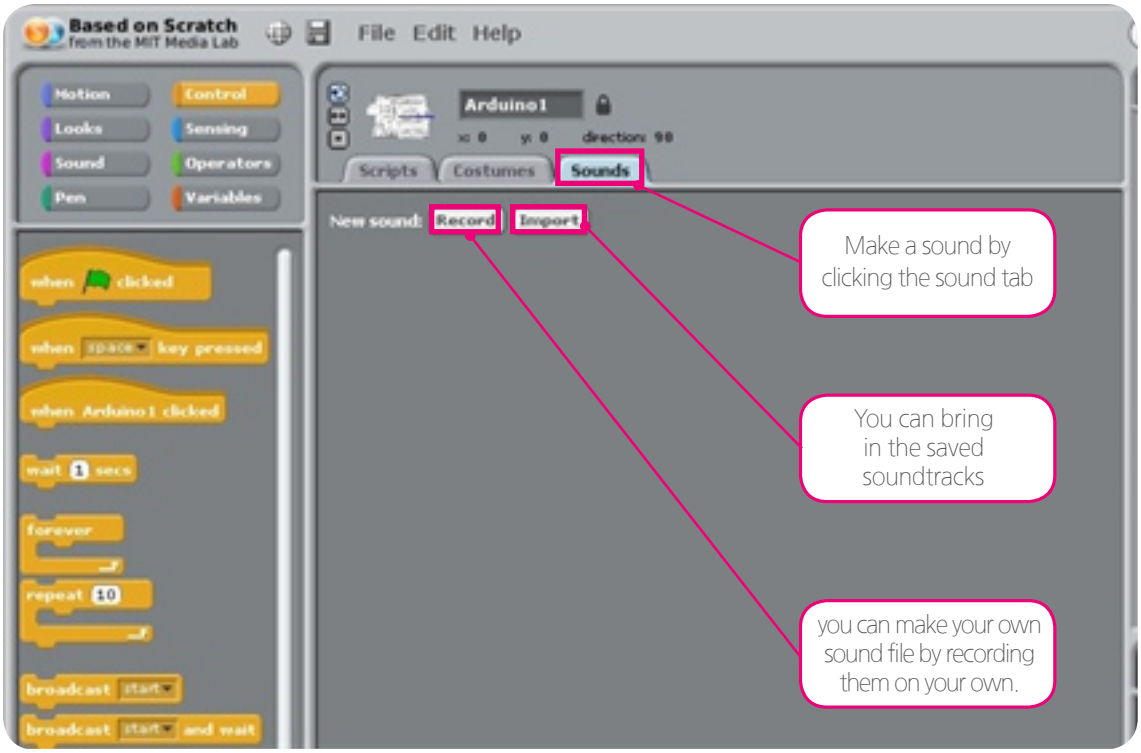
When “start” is broadcasted, it must be received. The figure above is the “when receiving start’ block. Below that is the script that must be activated.

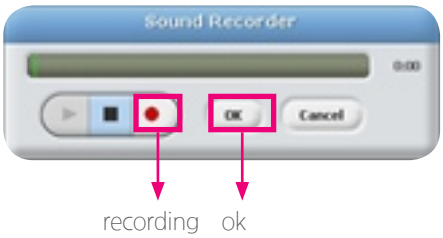
Therefore, if you put your hand on the digital 3 sensor, the ‘start’ is broadcasted, and it is received to activate the script.

Mole game coding activity

Make a sound for a mole game

Now let's make a script after receiving the 'start' broadcast.
We need a sound for announcing the start of the game, signaling number 1~6 and sound that announces the end of the game. You can record this sounds in Scratch! If you cannot record the sounds, you can simply open the soundtrack.





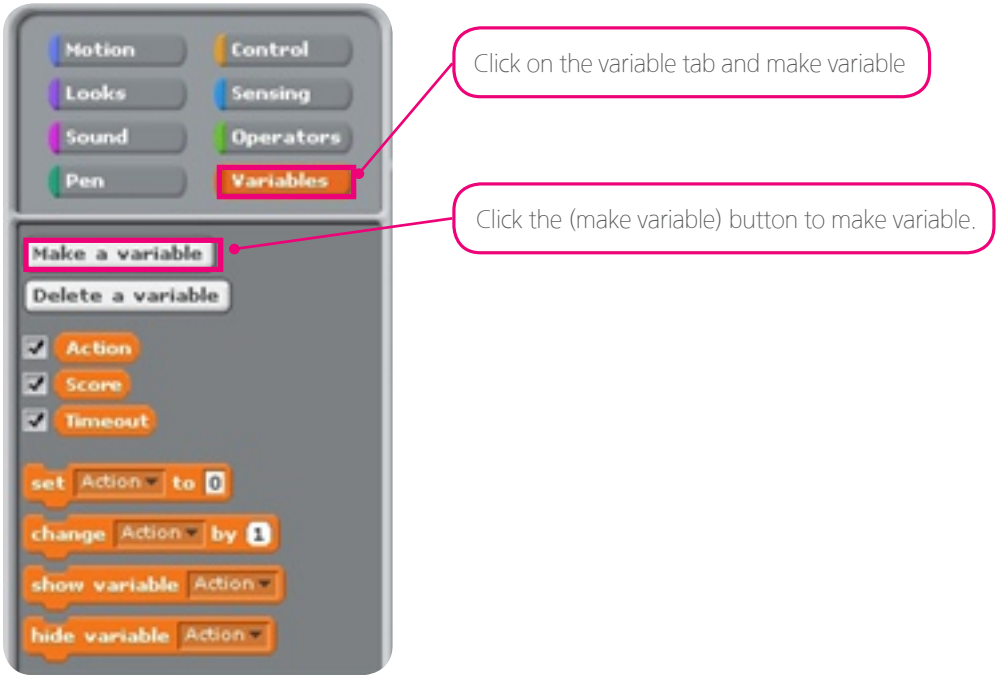
If you click the record button the recorder window pops up. If you press the red circle the recording begins, and if you press the pause button, the recording stops. Press the 'Ok' button after the recording, it will save the recording file.

CLASS 4

Mole game coding activity

Mole game variable coding

Did you make all the sound file necessary?
Then let's make a script for the activation of the game.
In the mole game, every time you press the infrared sensor, the 'score' will increase by one point. And each infrared sensor has to be pressed within a certain 'time.' Here we have to use variables, such as 'Run', "Score', 'Time'. You can simply think of variables as storage space.

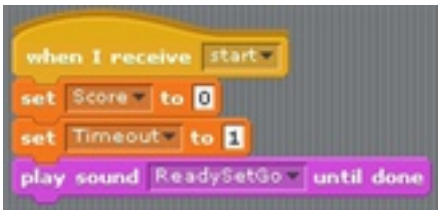


Mole game variable coding

Make a variable like the picture above. Here, 'Action' is activation, 'Score' is score, and 'Timeout' is responsible for representing time. If you made all the variables above, you have all materials you need! We are going to utilize these materials on the next page!

Mole game coding activity

Mole game activation coding



Drag the coding blocks and make a script like the one above. Does the work make sense?
First, receive a 'start' broadcast from the top. Then, the blocks below will be activated in order. Save 0 at 'Score' variable, and 1 at 'Timeout' variable. Right now, there is nothing in the storage space in variables. Therefore, we need to set the first value. After saving, play the sound file announcing the start. Drag block from the sound tab and add it to the script.

How to use the mole game block

Now you need to express "After the sound file announcing start is played, the computer will call up the numbers 1 to 6 and need to press the corresponding infrared sensor within the specified time."
Now let's think about how to call an arbitrary number from 1 to 6 and how to behave when each number is called.

Make a script like the one above. The block 'random number between 1~6' is a block that produces the number arbitrarily. Put in one of the produced arbitrary number into the 'Action' variable. After the number is saved in the 'Action' variable, connect it to the "If~" block. This block activates the script when certain situation , in this case "number saved in the 'Action' variable = number inserted in the blank", after the number is saved into the 'Action' variable.
For example, if the random number block has produced 1 and saved it to 'Action' variable, then the scripted inside the 'If 'Action=1" block will be activated.

CLASS 4

Mole game coding activity

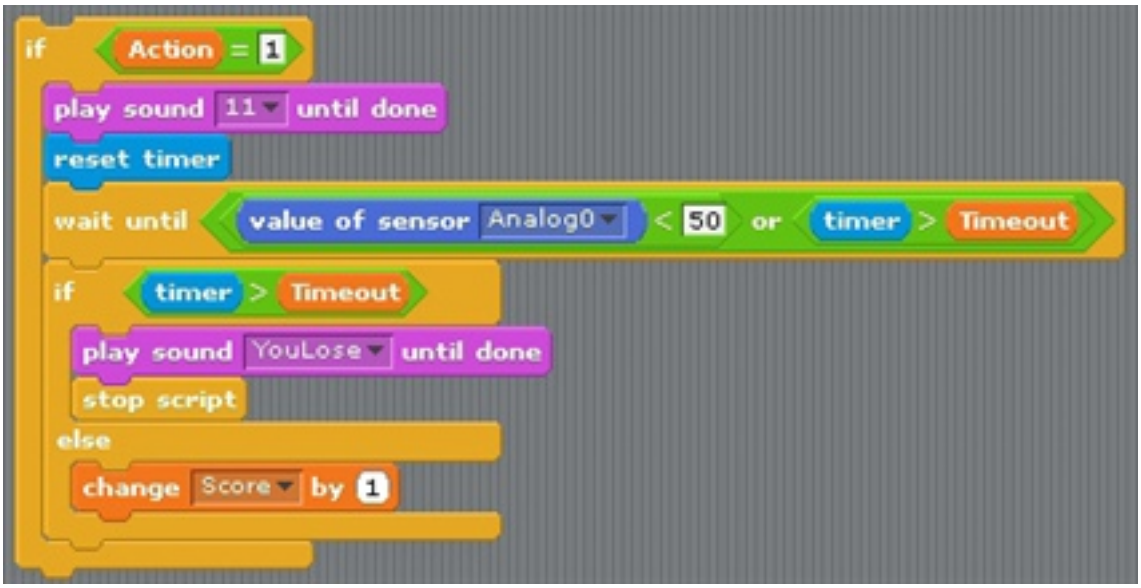
Mole game 'If~' block repetition

Do you remember the 'If~' block used in the previous page?
It was a block that activates the script when certain condition is satisfied. Then the 'If~' block for number 1 to 6 will be made.
Which means that there will be six 'If~' block connected. Then what should happen to the script inside the each 'If~' block?
They must be similar to each other. This is because as long as the 'Action = number input' condition is confirmed, the behavior that must be expressed is same.
Therefore, if you make a script inside "If 'Action=1'" block accurately, then the following tasks are relatively easy. Then let's learn what kind of script goes into the "If 'Action=1'".

Look at the script above. In the 'Action' variable, there is some variable between 1-6 stored, and this is a script that runs when the number stored is 1. Shall we take a look at what blocks are inside?
First, we need a sound to tell users to press '1.' Computer must sound '1' and initialize the timer. Timer, in this case, is a stopwatch processing inside the scratch program. If you initialize the timer, the time becomes 0 second. Then use the 'waiting' block to wait until the Analog 0 port IR sensor is pressed or until the timer exceeds the time saved in the 'Timeout' variable. So, if the user presses the analog 0 sensor before the time saved in the 'Timeout' variable they will escape the "waiting" block. Or, if they fail to press the correct sensor before the time saved in the 'Timeout' variable, they also escape the "waiting" block. The reason why we connect these two conditions with "Or" block is because you will escape the waiting block if only one of the condition is satisfied. If we had to satisfy both of the conditions, we would have used the 'and' block.

Mole game coding activity

How to use “If not ~’ block in the mole game.



With the script we used in the previous page make a script like the one above.

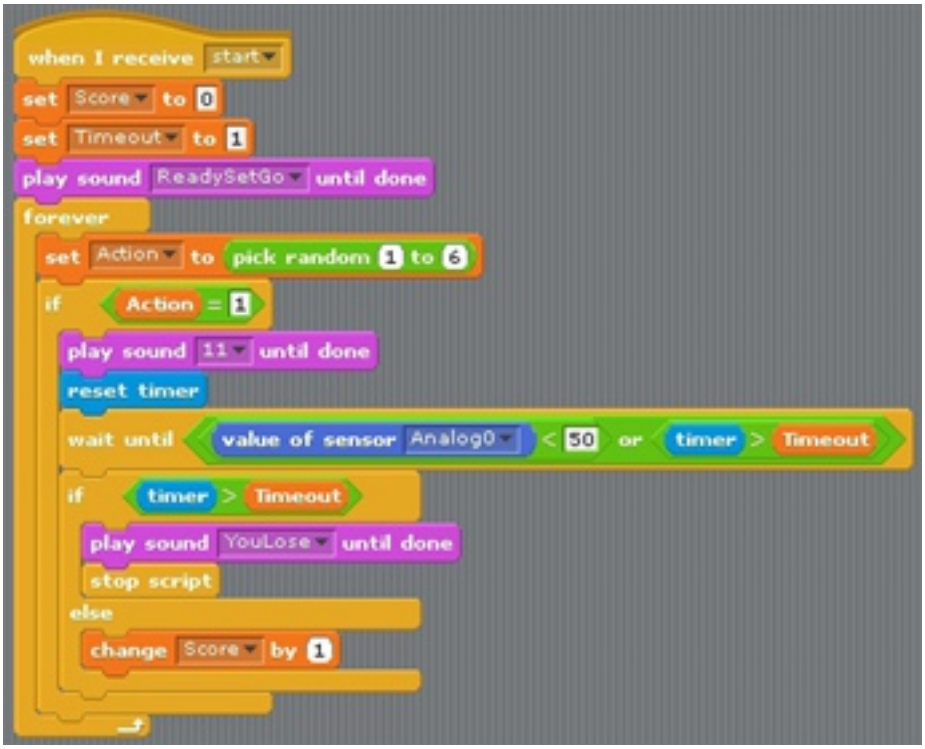
Previously, you expressed pressing the sensor or timer exceeding the time set in the ‘Timeout’ variable. Let’s think about what should happen next. If the timer exceeds ‘1 second’ the game is over. At this moment, computer must announce the end of the game and stop the script.
But if the timer did not exceed ‘1 second’, which means that the user pressed the IR sensor within a second, 1 point must be accumulated. We need to use ‘If not~’ block to express this. This block is similar to the ‘If~’ block we learned in previous lesson. We just added the “not” part to the ‘If~’ block.
Therefore, if you fail to satisfy ‘If~’ condition, then the script in ‘If not~’ will be activated. If you satisfy ‘If~’ condition then the script inside the ‘If~’ will be activated and the one saved inside ‘If not~’ will not. You must access inside the ‘If~’ to end the game and stop the script if the ‘timer > Timeout (1sec.)’ condition is satisfied in the ‘If not~’ block. If this is not the situation, you must add 1 point to ‘Score’ variable each time. Your score is increasing by 1.

CLASS 4

Mole game coding activity

Mole game coding

Now you've completed the script for the condition "'Action=1". So let's combine the scripts we have created so far.



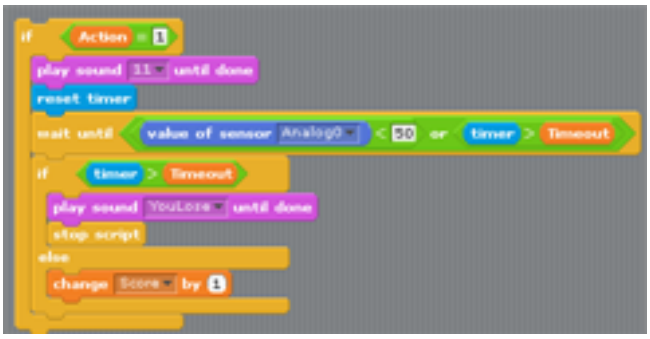
If you combine them it will look like the script above. There is one thing added here, the 'infinite repeat' block. This block means that it will check the internal script of the block repetively.
By using this block, the game will go through not just once but will continue until the script stops (until we have GAME OVER).

This completes the coding when the number to be called from the computer is '1'.
If so, how do you write a script when you call a number between 2 and 6? Try it yourself.

Mole game coding activity

Mole game 2~5 script coding

Utilize the script on the right and code script from 2 to 5. Be aware of change in value of 'Action' variable, each corresponding number of sounds and which port is the sensor plugged into.



CLASS 4

Mole game coding activity

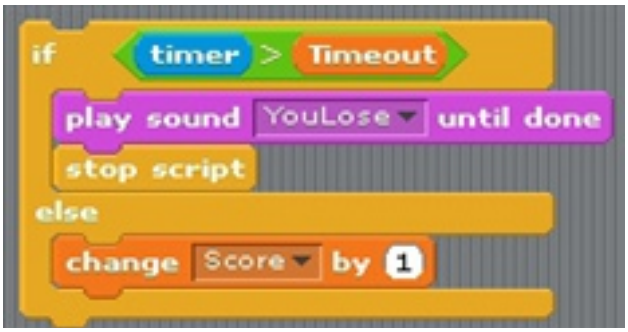
Mole game coding

The final script of the mole game will be shown as below!

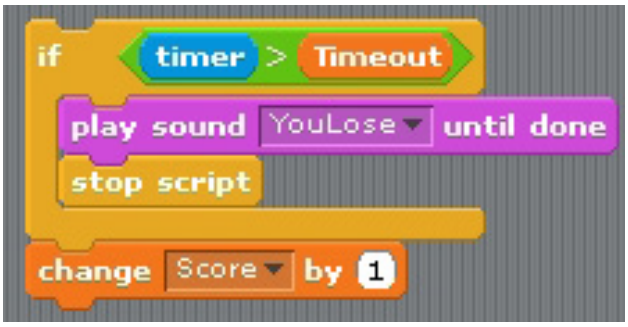
The whole script can be found at http://www.robotori.com/web_eng -> Moretips -> Manual ->EDU ->Logic boost CODING CLASS 4 ->download whole script

Mole game coding activity

Mole game efficient coding method



Above is the portion of the entire mole game script. There is a way to make this script with 'If not~' to be much simpler. Take a look at a content inside the "'not'. It is a process where you accumulate 1 point every time into the 'Score' variable. Do you think you can make this process easier by replacing the 'If not~' block with 'If~' block?



You can express the script as show above if you use the 'If~' block. It may be a small difference, but it is important to practice making everything as simple as possible when coding. The difference between the two is the presence of 'not'. You don't need to use 'not' to accumulate the point to 'Score' variable since if you fail to satisfy the condition of 'If~' block you automatically go on to the next block without activating the script.

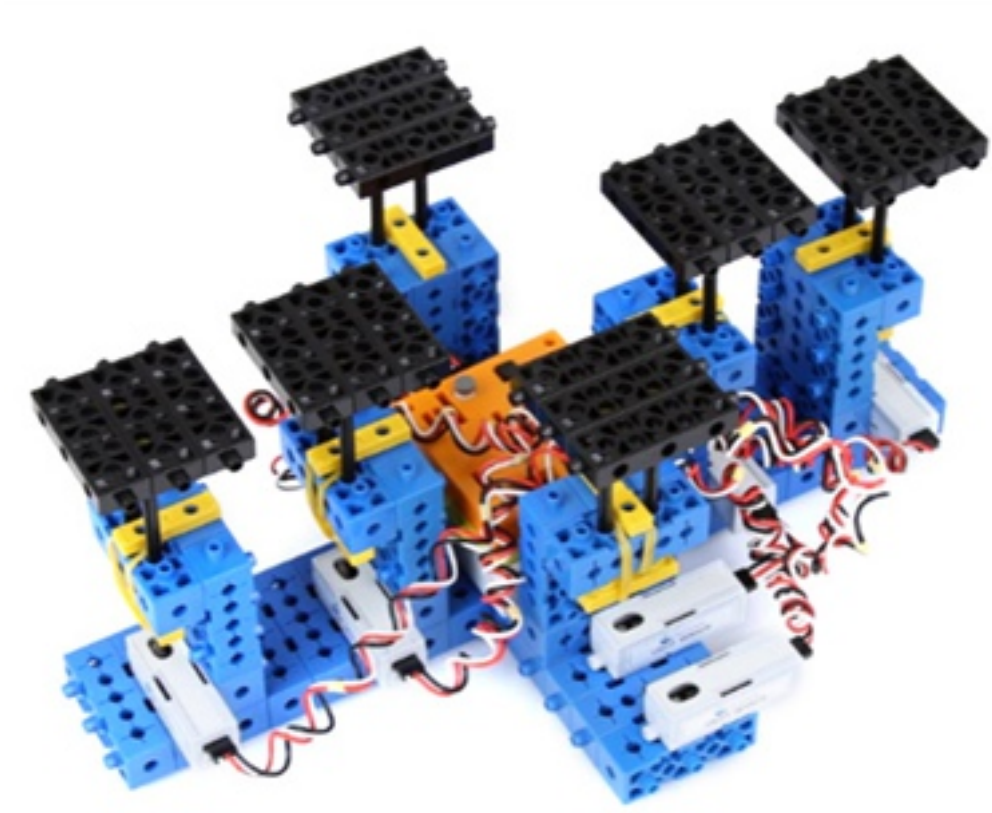
Like this, it is important to find more efficient ways and use less blocks.

CLASS 4

Mole game coding activity

Mole Game Play

Now please activate the mole game program you coded and if it does not function like it's supposed to, look through your book to check if you have made any mistakes.
So, shall we begin?



Mole game coding activity

Mole game Question 1

How was the game? Was it too easy for you?

Then code the program so your time decreases by 0.1 second every time 10 points are accumulated in your score.

CLASS 4

Commentary

Mole game question 1 commentary

Did you get it to decrease your time by 0.1 when you accumulate 10 points to 'Score'? was it activated correctly? If it did, just take this commentary as consultation and move on to the next page. If it didn't work out as you wanted, use the script below.

Combine the script above into the beginning of the 'infinite repeat' block and run the program. If the 'Score' becomes the multiple of 10, everything else becomes 0 when it is divided by ten. Therefore, every time 'Score' becomes 10, 20, 30,··· rest divided by 10 becomes 0 so the script inside the 'If~' block is activated. The script inside the block means that subtract 0.1 from the currently saved 'Timeout' variable value and save it again. Therefore, whenever the 'Score' becomes 10,20,30,··· the 'Timeout' will become 1, 0.9, 0.8,.. The numerical change of 'Time-out' variable is shown in the screen on the main cell on your right.

If you follow the script above, you will notice something odd. You will see the variable value of 'Timeout' will start at 0.9 instead of 1. This is because if the 'Score' starts with 0 and when divided by 10, it will give 0 again. Therefore, to satisfy the condition, the variable value of 'Timeout' starts with 0.9.

You can solve this problem using the script above. Connect the condition saying "'Score' must be greater than 0" with 'and' block so that the 'If~' block is only activated if both condition is satisfied. If you code it like this, you can avoid the first value being 0.9.

The whole script can be found at http://www.robotori.com/web_eng -> Moretips -> Manual ->EDU ->Logic boost CODING CLASS 4 ->download whole script

Activity

Mole game question 2

How do you feel after doing the mole game with shortened time?
Wasn't it bored to increase the number by 1? Now let's make a program where the computer continuously calls the numbers 1 or 2, and the user must input the number that is called at the very end.

CLASS 4

Commentary

Mole game question 2 commentary

Did you make the program where computer calls 1 or 2 continuously and the user have to input the last number that computer calls? Was it working well? If it did, just take this commentary as helpful advice and move on to the next page. If it didn't work out as you wanted, use the script below.

〈Generate 1 or 2 random and trick numbers〉

Since you have to select one or two numbers, use the random number block to find a random number between 1 and 2. If the generated random number is equal to 2, create a tricky random number, sound the number to be played by the trick first, and proceed to the code in the previous example.

The code above is a PlayTrick code with a reduced number of 'If~' blocks. Reducing the code is good for readability, so it's good to be able to codify coding to reduce the length of the code.

〈Play trick 3〉

The whole script can be found at http://www.robotori.com/web_eng -> Moretip
-> Manual ->EDU ->Logic boost CODING CLASS 4 ->download whole script

SCRATCH CODING KIT

Logic Boost

List

LESSON 5

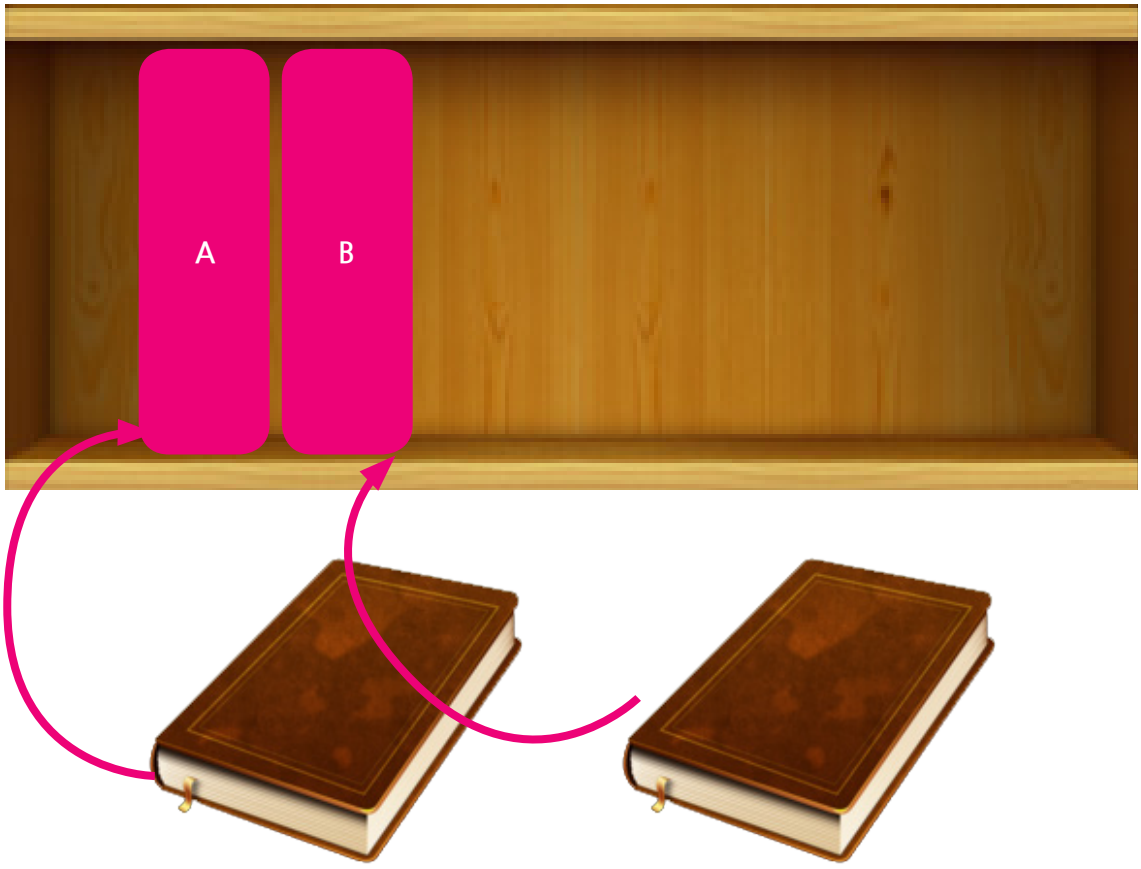
CLASS 4

Introduction

What is a list?

A list can be used to save several number or information of values. It is very similar to the variables; the only difference is that a list can save more than one value while the variables can save only one value. Think of the list as a book shelf that can contain many books.

list



Book hat has a value of "A"

Book hat has a value of “B"

List coding activities

Why do you need a list?

The variables we learned in the previous lesson can save only one information.
But, list can save more than one information. With list, you don't have to make several different variables, you can save several information at once. Furthermore, you can get the saved information anytime and use it.

Making a list

To make a list, click on the tab you used for variable, and click "Make a List"

Then you'll see the pop-up window showing up. This pop-up window is for assigning the names.
Name your new list as "List." Type in "list" in the input window and click "Ok".

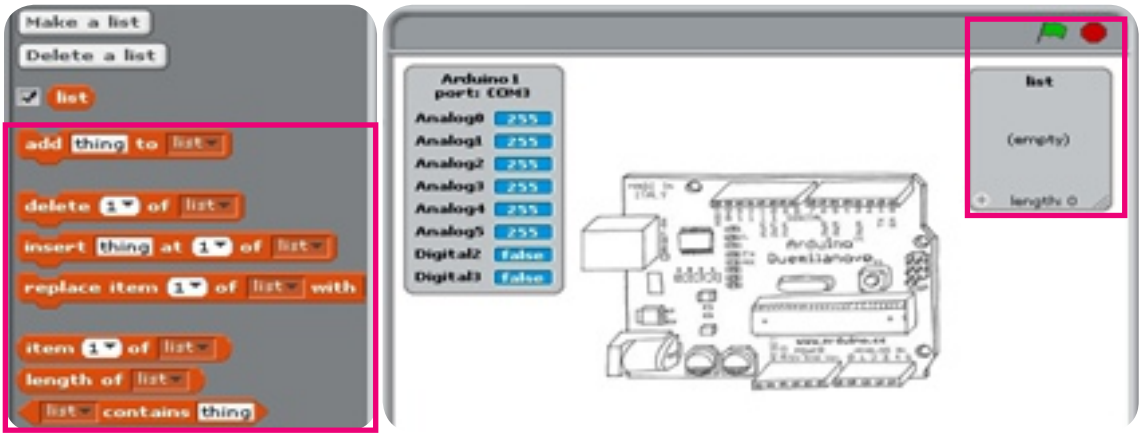
CLASS 4

List coding activities

Making a List

After making a list, you will see some differences in the scratch screen.
First, you will see new brown blocks related to “list” list in the palette menu.

On the right side of the screen, you will see the “list” list in the stage window.

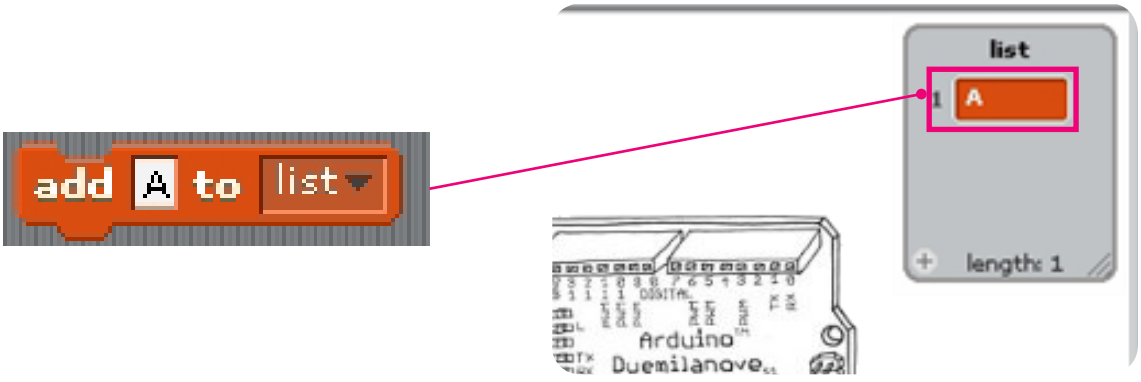


List coding activities

List Blocks

Now we will learn about the new blocks you need to know to use the list

The first block is a block that adds values to your “list” list in order. Just like adding a book on the shelf, you are adding values to the “list” list. Currently on the stage screen you “list” shows up as empty but you’ll be able to see the change in number and value as you add the value to the list.



CLASS 4

List coding activities

List blocks

The "Deleting an item at position 1 in the list" block deletes certain value or character that user chooses at specific place.

If you double click part (1) and change it to different value, delete all values using the black arrow, or delete the last item.

The "put ( ) in position (1) in the (list)" block puts a value or letter in the specific portion of the list. If you want to add the value you forgot to add as you were adding, you can use this block to fix it afterward.

As you can see above, there are 4 values (A, B, D, E) saved in the list at first. If you want to add 'C' between 'B' and 'D' , you can use the block that puts 'C' into the position 3. This is because the value 'C' has to go in between 'B' and 'D' and since 'B' I in the position 2, 'C' has to go after 'B' and has to be in the position of 3.

List coding activities

List blocks

The block “Replace item 1 in list with ~” changes the value of the item in the list.

As you can see above, the first list has C saved in the position 3. If you use the “ replace item 3 in the list with CC” block, you can see that the ‘CC’ has replace C in the list.

The “item at position (1) in list” block on the very left is the block that shows the value that I saved in the specific position. The value of the item at position (1) in the list will be ‘A’ for now. It’s a block hat you can use as a specific value when you’re coding. The “size of list” block in the idle is the block that shows how many items there are n the list. Currently value of this block will be 5.

This is because the list we are looking at has 5 items. The block you see on the right, “include () in list?”, is the block you can use to check if the list has certain value in it. The block “include (A) in list?” will show the value of ‘True,’ but the block “include (F) in the list?” will show the value of ‘False’.

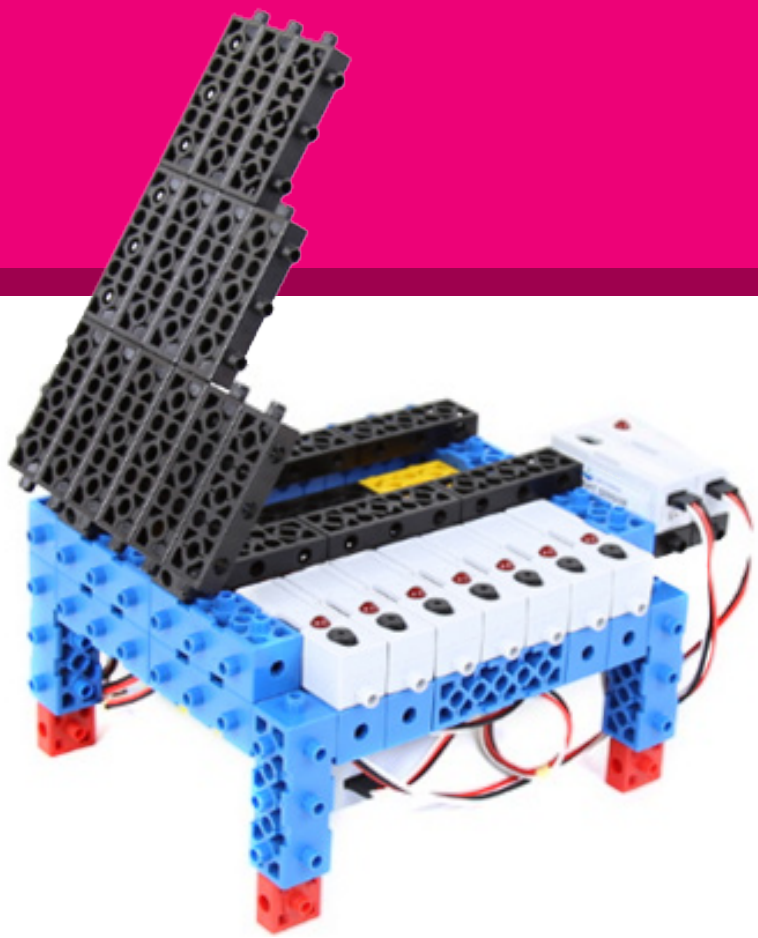
Now shall we take a look at a piano which is the example of usage of list?

SCRATCH CODING KIT

Logic Boost

TORI piano

LESSON 6



introduction

TORI piano

The TORI Piano is a model for recording and playback in Scratch, and we will code the program using the button sensor, the infrared sensor and the LED sensor.
The key of this coding is that if you press and hold the infrared sensor connected to the input digital port 3 for more than 2 seconds to record, and if you press and release it for less than 2 seconds, the existing recorded sound will be played back.
TORI Piano uses 7 button sensors, 1 infrared sensor, and 1 LED sensor.One infrared sensor plays the role of recording and playback, and the remaining seven button sensors are the sensors responsible for the sound of Do Re Mi Fa So La Ti. An LED sensor is a sensor to let you know that you are recording by making it light during recording.
Now keep these things in mind as you start coding.

Code TORI Piano (Principle)

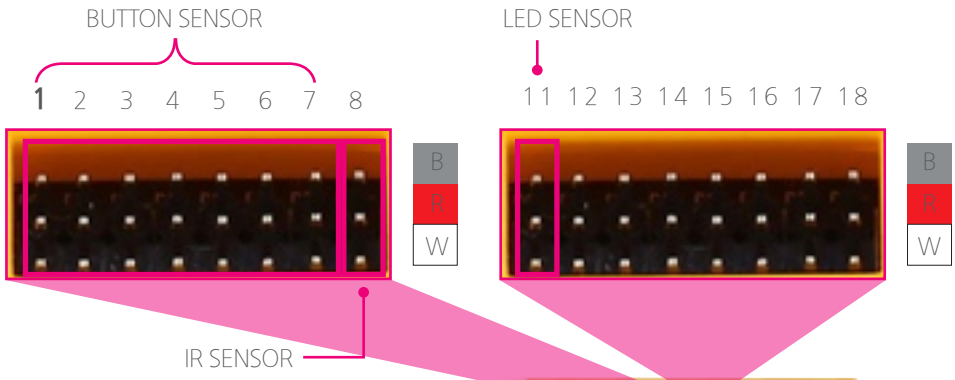
Let's go over the steps to take into consideration before coding the TORI piano in detail. The process of coding with an overall guideline on what actions we should express and how we can express them is very important.

Let's take a quick look at the basic behaviors we need to implement.
1. Let's take a quick look at the basic behaviors we need to implement.
2. When the recording starts, it saves the scale and beat you have pressed. Pressing the infrared sensor again will end the recording and play back the corded sound.
3. If you press the infrared sensor of port 8 for more than 2 seconds, the existing sound is deleted, and new recording is done. If you press and release it for less than 2 seconds, the previously recorded sound will be played back.
4. The button sensor for each scale must play the sound once if the button sensor was pressed once.

Should we start making the TORI Piano now?

CLASS 4

Making a robot



Material list

							
Diamond V8 x 12	Diamond V6 x 9	Rubi 8 x 3	Rubi 4 x 9	Rubi 2 x 6	Rubi 0 x 4	Mini 2 x 2	Mini 1 x 4
							
Triangle x 2	Mainboard 128 x 1	Battery Case x 1	IR SENSOR x 1	LED sensor x 1	button sensor x 7	Link x 10	