Coding the Ultrasonic Turtle

Coding the Ultrasonic Turtle

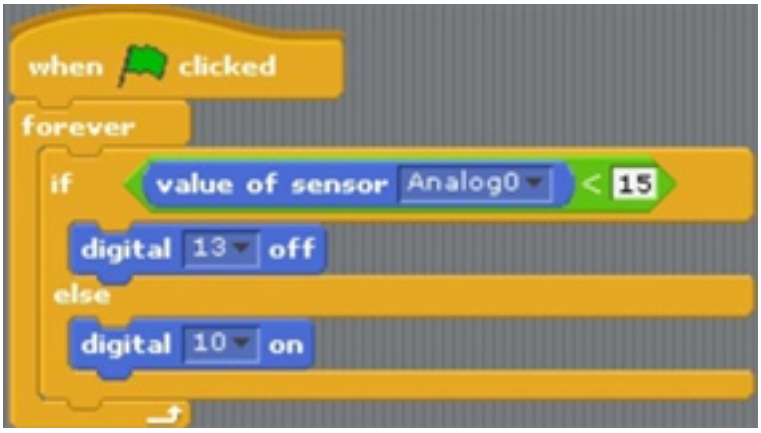
You have to turn on the LED sensor according to the distance of the ultrasonic sensor. There are three LED sensors on the ultrasonic turtle. When the tortoise is closest to the wall, the LED sensor is off.
After that, as the tortoise gets further and further away from the wall, the LED will turn on one by one. First, try coding the first LED sensor. If the distance between the tortoise and the wall is less than '15', the LED sensor connected to the digital 10 port will be turned off. If it is larger than '15', write the script below to turn on the LED sensor.

CLASS 3

Coding the Ultrasonic Turtle

Coding the Ultrasonic Turtle

Remember when you coded the mini windmill to work according to the distance of the ultrasonic sensor? If the LED sensor of the digital port 10 is off and the digital value of the digital port 10 is higher than that of the analog value of 15 (if the value is larger) You can code it so that the LED sensor is on.



Does it look like the script above?

Since the ultrasonic turtle is turning on the LED sensor when the distance is far away, the LED sensor should be off when the distance is close. The value of 'Analog 0 sensor' is smaller than '15', which means that the distance between the ultrasonic sensor and the recognition board should be closer to '15'.

At this time, the LED sensor must be turned off, so the 'output off of digital 10' block is executed by the condition. If not (if the wall is longer than the analog value of '15') then you have to turn on the LED sensor using the 'digital 10th output on' block.

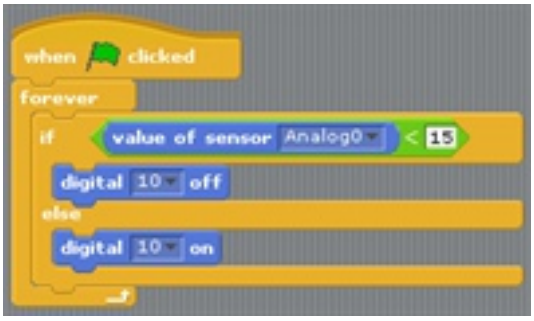
Coding the Ultrasonic Turtle

Codig the Ultrasonic Turtle

Did you check that the first LED sensor turns on and off with distance?

Now try coding for the second and third LED sensors. The standard for the analog value of the first LED sensor was '15'. Second, code the setting of the analog value of the third LED sensor as '20', '25'. Along with the analog value criteria, the number of digital ports will of course change. If you change the output of the digital port and the analog value slightly, you can easily create a script.

Please refer to the script you created earlier and code yourself in the space below!

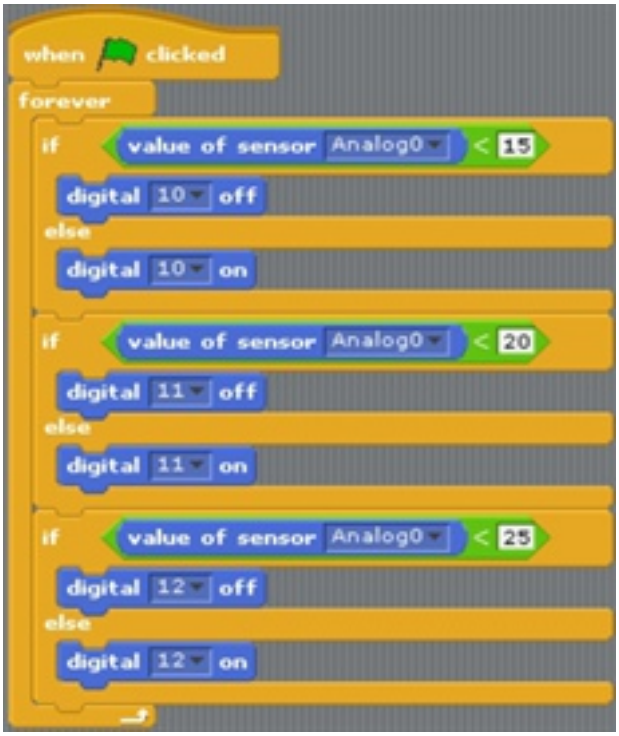


CLASS 3

Coding the Ultrasonic Turtle

Coding the Ultrasonic Turtle

Have you tried coding for the rest of the LED sensors? Now you can combine your scripts into one. If you followed this process well then you should have a script like this!



The script shown above now turns on the LED sensors one at a time as the distance from the wall increases. If the distance value is less than 15, all three LED sensors are turned off. If the distance value is greater than 15 and less than 20, one LED sensor is turned on (digital output 10 turns on) 'And less than' 25 ', two LED sensors are turned on, and if greater than' 25 ', three LED sensors are turned on.

You can see that several LED sensors are turned on depending on the distance from the ultrasonic sensor.

“IF” syntax utilization

Using ‘IF’ syntax

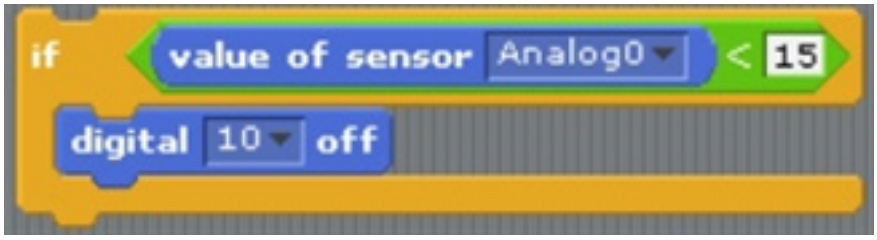
In the past we created a script using the 'if & if' syntax. If so, now 'if; Use the syntax to create a script for the Ultrasonic Distance Meter. I can make it very simple. And think about how to code the 'or' part. Now, changing the 'if & if' syntax to the 'if' syntax will be very easy for many departments. Keep in mind that it is still important to continue practicing. Let's create a script below.

CLASS 3

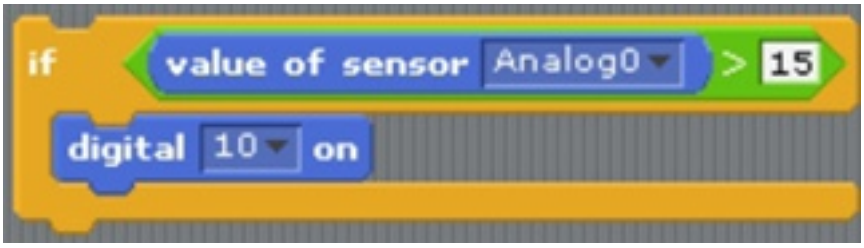
“IF” syntax utilization

Using ‘IF’ syntax

If you had previously controlled the digital output with the 'if & if' statement, Take control of the digital output with two phrases. First, when the ultrasonic sensor value is low (when the recognition board is approaching), the first LED sensor should be off. Create a script



The above script will turn off the LED sensor connected to the digital 10 port when the ultrasonic sensor value is less than '15'. Now, when the ultrasonic sensor value is greater than '15', you have to turn on the LED sensor of digital 10 port. Please create the script below.



We created a script when the analog reference value is '15' above. When the remaining baseline values are '20' and '25', you can make the scripts the same. If analog reference value is '20', digital output 11 should be turned off and on. If analog reference value is '25', digital output 12 should be turned off and on. Once you've created each script, you can create a loop that will continue to check the condition with an 'infinite loop' block

In the previous level, we have practiced how to make the 'if & not' syntax into the 'if' syntax through many examples. If you understand the terms of the 'if & not' or 'if' statements, then there is no problem in interpreting these two statements the same way. Be sure to always put the condition in the phrase carefully.

“IF” syntax utilization

Coding ‘IF’ syntax

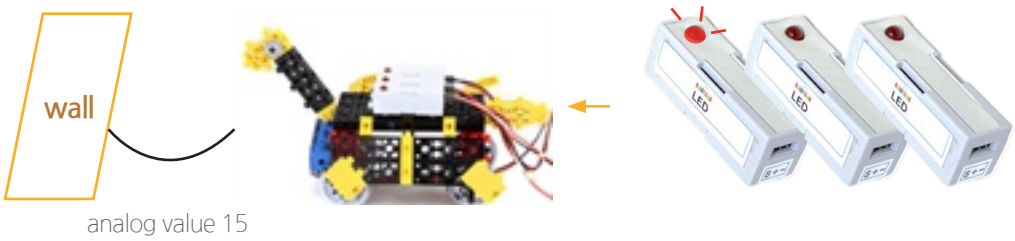
If you combine the script you made, it should look like the one below!

CLASS 3

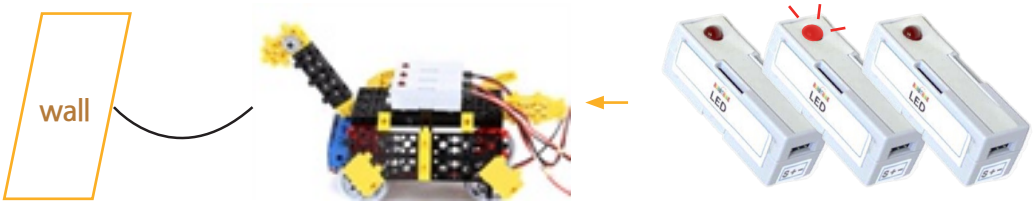
"IF" syntax utilization

Utilizing the Ultrasonic Turtle

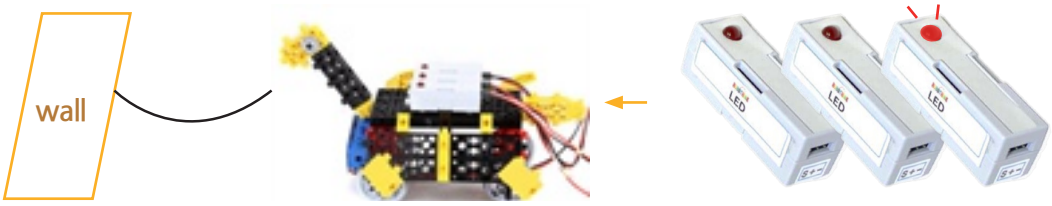
An ultrasonic turtle is a model that turns on multiple LED sensors as the distance from the wall increases. It was a model in which the LED sensor was turned on cumulatively every time the distance was distant. Now, let's create a script that turns on the LED sensors sequentially, rather than turning them on cumulatively.



analog value 15



analog value 20

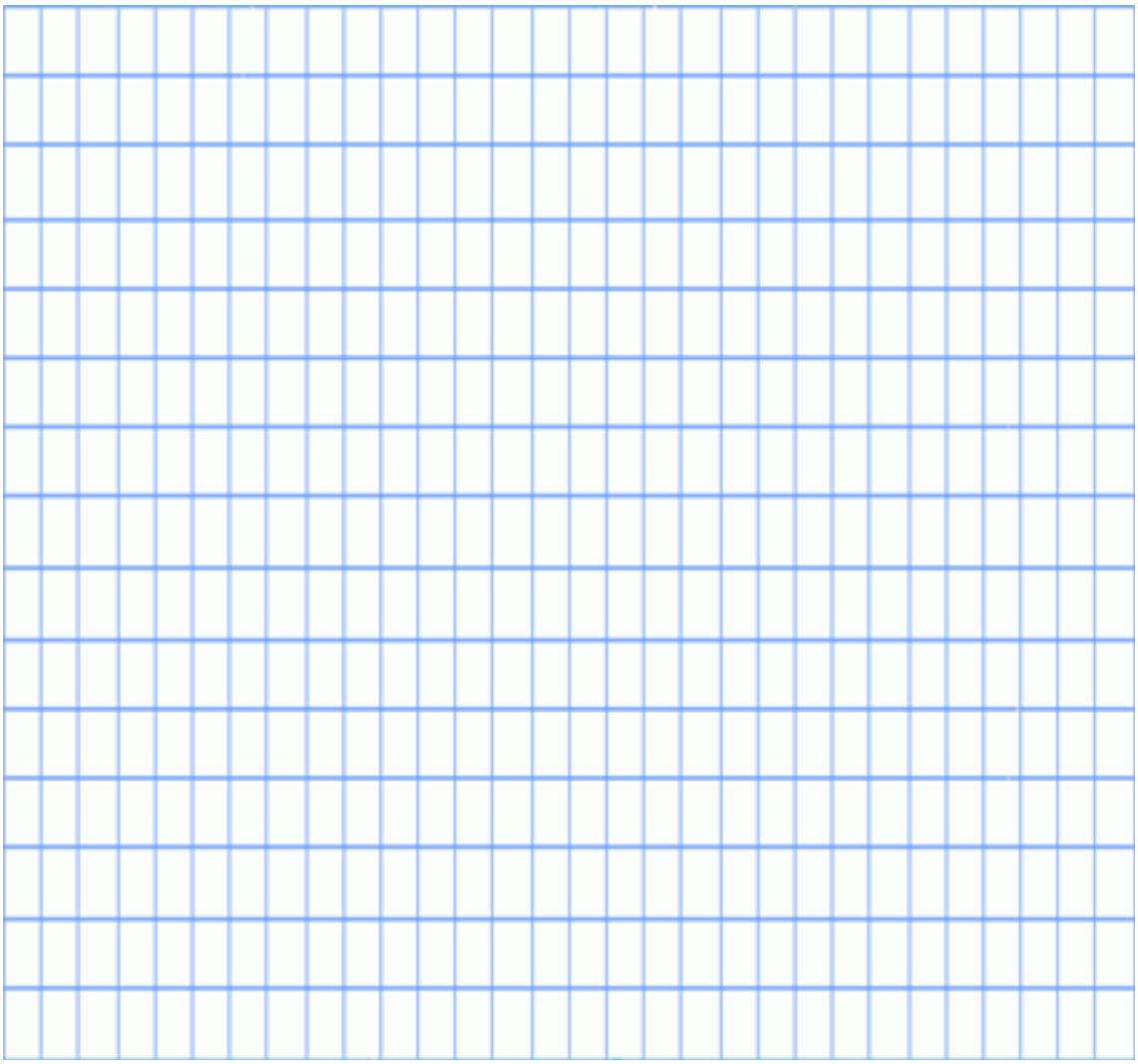


analog value 25

“IF” syntax utilization

Utilizing the Ultrasonic

Now make a script that modifies the script so that the LED sensors do not turn on cumulatively, but instead turn on sequentially. Please refer to the 'DJ Box' script you learned earlier. Use the 'and' operator to make the condition better. Let's create a script in the space below.



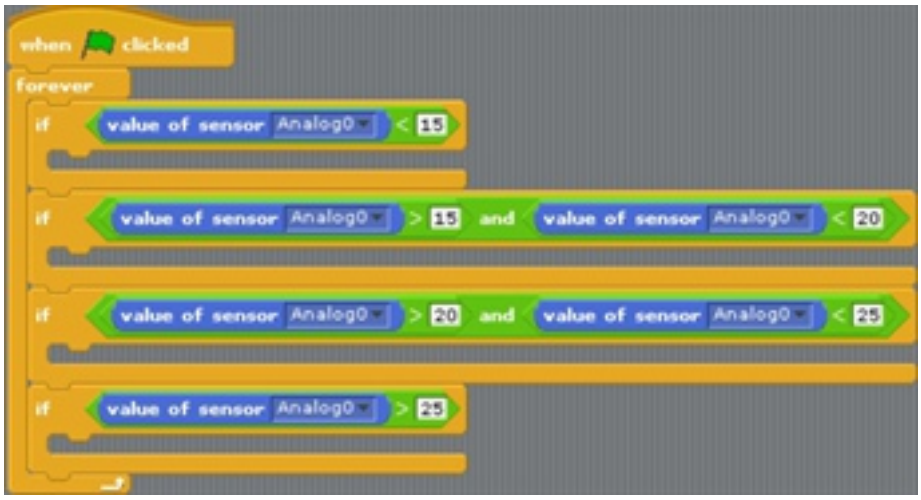
CLASS 3

"IF" syntax utilization

Utilize the Ultrasonic Turtle

In order to turn on the LED sensor sequentially, it is necessary to distinguish the sections first. The analog reference values you currently have are '15', '20', '25'. If so, a total of four sections will be created. , The output of the LED sensor is divided into four sections when it is smaller than 15, larger than 15, smaller than 20, larger than 20, smaller than 25, and larger than 25, you have to control the output of the LED sensor in each section.

First, divide the section between '15' and '20', '20' and '25' are expressed using the conditional operators 'and'.



Did you make a script above? We split up the section using 'and' operator. Now let's put a coding block on how each LED sensor works in each section.

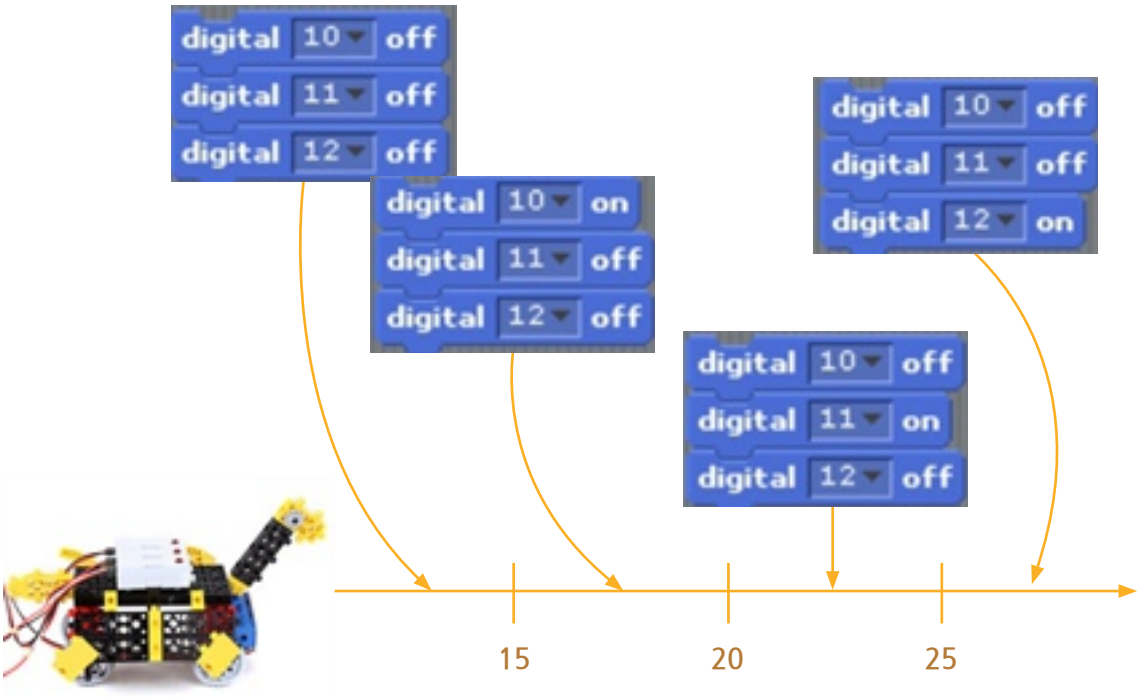
When it is less than the reference value of '15', all three LED sensors should be off. Turn on the first LED sensor between '15' and '20' (use digital output 10). When it is between '20' and '25', turn off the first LED sensor and turn on the second LED sensor (turn off digital 10 and turn on digital output 11). When it is greater than '25', turn off the second LED sensor and turn on the third LED sensor (turn off digital output 11 and turn on digital output 12). Create a script to do this.

“IF” syntax utilization

Utilize the Ultrasonic Turtle

First, it's easy to code three sets of LED sensors on and off. Combine the three digital coding blocks as shown below and adjust the values for each section.

With this combined coding block, you can adjust the value for each section. Remember that the LED sensor will turn on sequentially and complete the script.

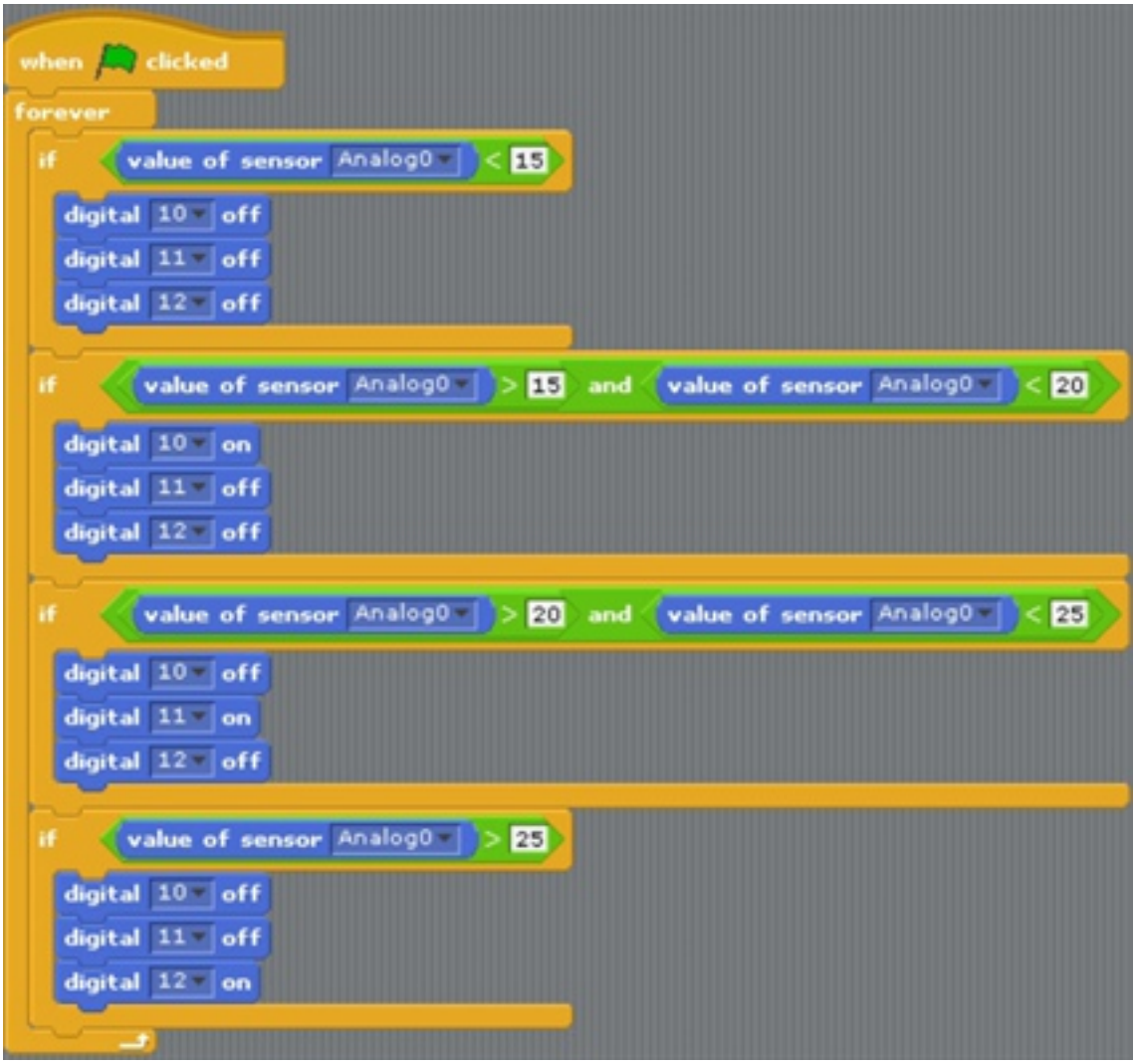


CLASS 3

"IF" syntax utilization

Utilizing the Ultrasonic Turtle

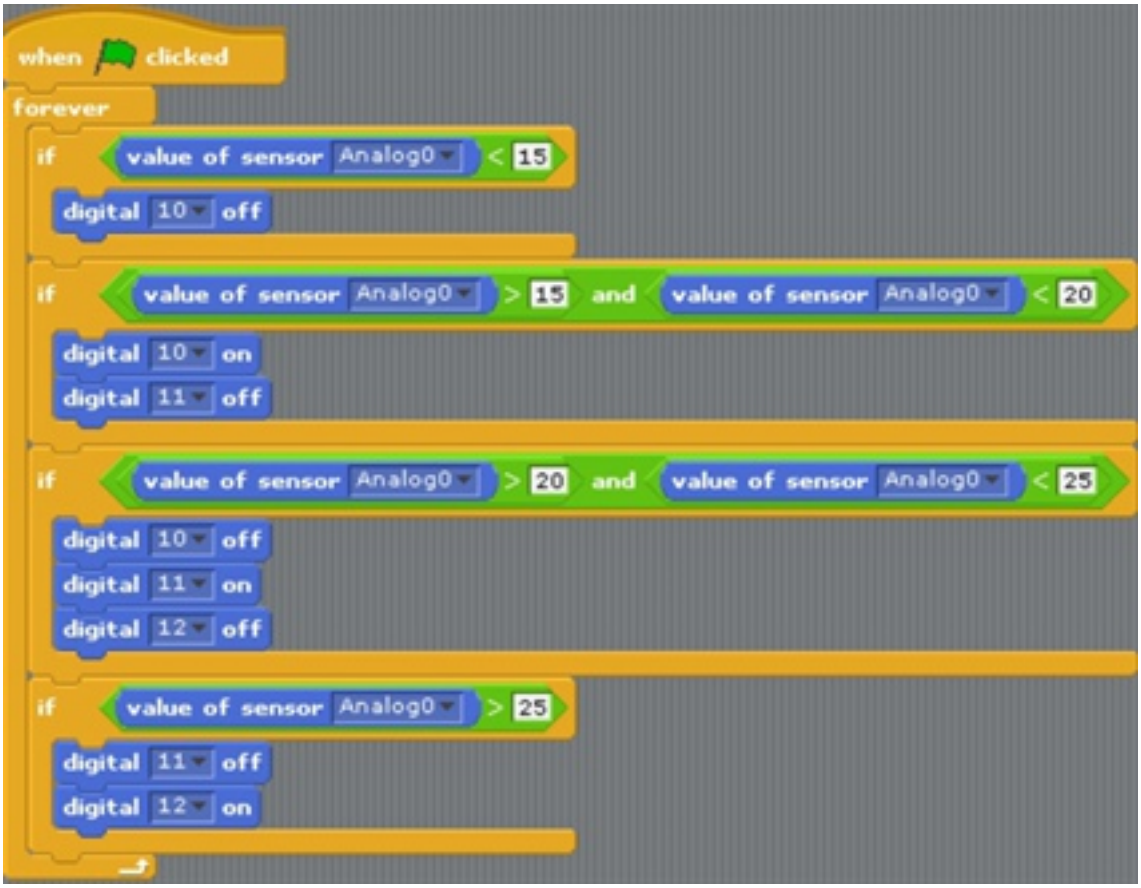
If you combine all the scripts you have made, it should look like the one below!



“IF” syntax utilization

Utilizing the Ultrasonic Turtle

By subtracting the unnecessary digital output blocks from each section, you can create shorter scripts. Try to reduce the length of the script by subtracting the unnecessary coding blocks. Please see below for reference.



CLASS 3

"IF" syntax utilization

Utilizing the Ultrasonic Turtle

Now you can use the ultrasonic turntable you just made in front of you and use the sound tap of the scratches and the ultrasonic sensor to play differently depending on the distance.

Now, instead of the LED sensor, we will use the sound tab to make the sound in accordance with the distance of the ultrasonic sensor in the scratch program.

Make an ultrasonic turtle and play your own one! Because it depends on the distance, you have to play the tortoise and play the tortoise. Remember to remember this. Create a script using nested 'if' conditional statements!

“IF” syntax utilization

Utilizing the Ultrasonic Turtle

Similar to the concept of an ultrasonic LED sensor, let's make a turtle that makes a piano sound according to distance. First, click on the Sounds tab and drag the block below.

Double-click the above coding block. Can you hear the sound? If you can not hear the sound, you need to connect your earphone or speakers to your computer to create an environment that can make a sound. Make sure your computer is muted. This coded block plays a role of playing back a 60-scale by 0.5 times. We are going to use this block to show the sound of a pianist.

If you press the arrow next to the '60' value, the piano keyboard will appear and you can press the desired key to make the corresponding sound. And please change the beat to one night. It takes at least one beat to get the sound to sound properly. Prepare a coding block so that the eight scales of the DoReMiFaSoLaTi are displayed.

do : 

re : 

mi : 

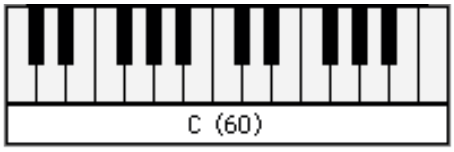
fa : 

sol : 

la : 

si : 

do : 



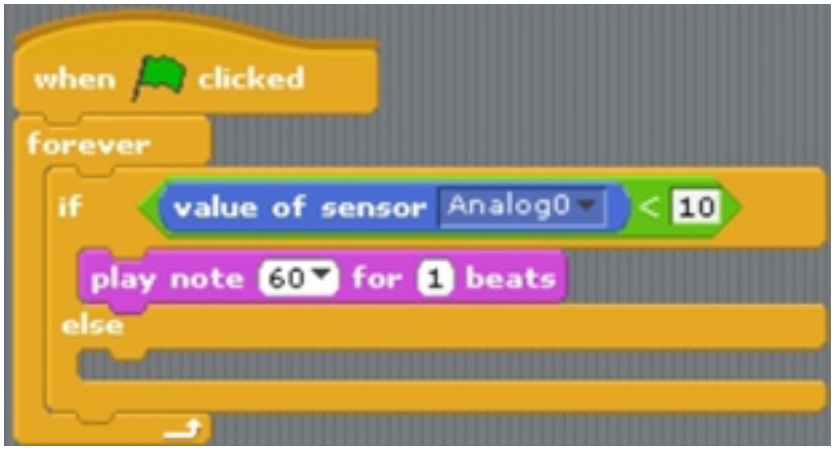
CLASS 3

"IF" syntax utilization

Ultrasonic Turtle 'Do'

Now, let's create a script in earnest. Let's start by setting the standard for distance. Consider the first distance reference as the analog value '10'. Just like I had the baseline when I turned on the LED sensor in the front.

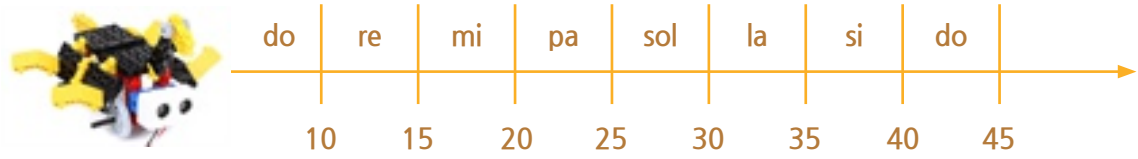
Now, start coding '10', assuming that the value is incremented by '5' so that the interval is fixed. First, let's create a script to play the Do when it is smaller than the analog value '10'. Create the script below!



The above script is a script that will play a scale of 60 beats per minute if the distance between the ultrasonic sensor and the wall is less than '10'. The reason for using the 'if & else' syntax is that you have to play a different note if you are farther away from the analog value of the turtle '10'.

Now we are going to create a script that will be nested so that the 'no' condition is followed by the 'if & else' syntax for the next scale. Do you understand?

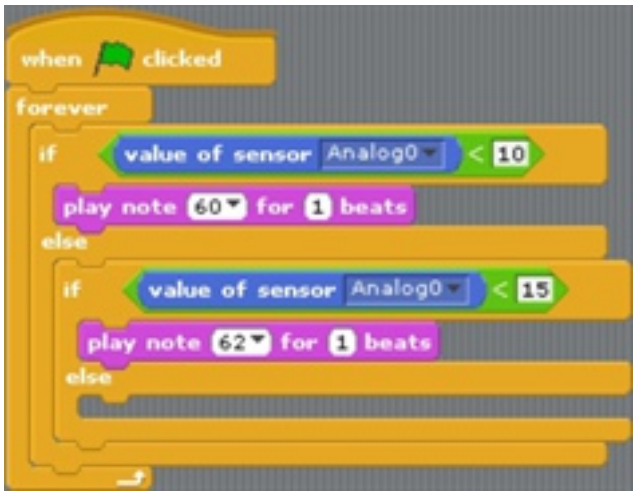
Below is a picture of how the distance depends on the distance. Please refer to future when deciding standards!



“IF” syntax utilization

Ultrasonic Turtle ‘Re’

Now let's create a script that gives the next scale. You have to run it when the analog reference value is less than '15'. In the script I created earlier, let's create a nested 'if & else' statement. Create the script below!



The above script is a script that tells you to play 62 scales in one beat if the distance between the ultrasonic turtle and the wall is less than '15'.

You have to use nested 'if & else' syntax in your existing scripts. The first condition is that if the value of Analog 0 sensor is less than '10', it plays 60 scales in one beat. 'Or' means that the value of analog 0 sensor is greater than or equal to 10. In other words, the first or 'if&else' condition is' if the value of the analog sensor is greater than or equal to '10'.

In addition, by using the fact that the value of the analog 0 sensor is smaller than '15' by overlapping the 'if &' statement, the actual condition is that when the value of the analog 0 sensor is greater than or equal to '10' That coding block is used. Therefore, when the value of the analog 0 sensor is greater than or equal to '10' and less than '15', the 62 scale is played in one beat. If so, how do you know how to create a script for the next scale?

Now, make your own scales!

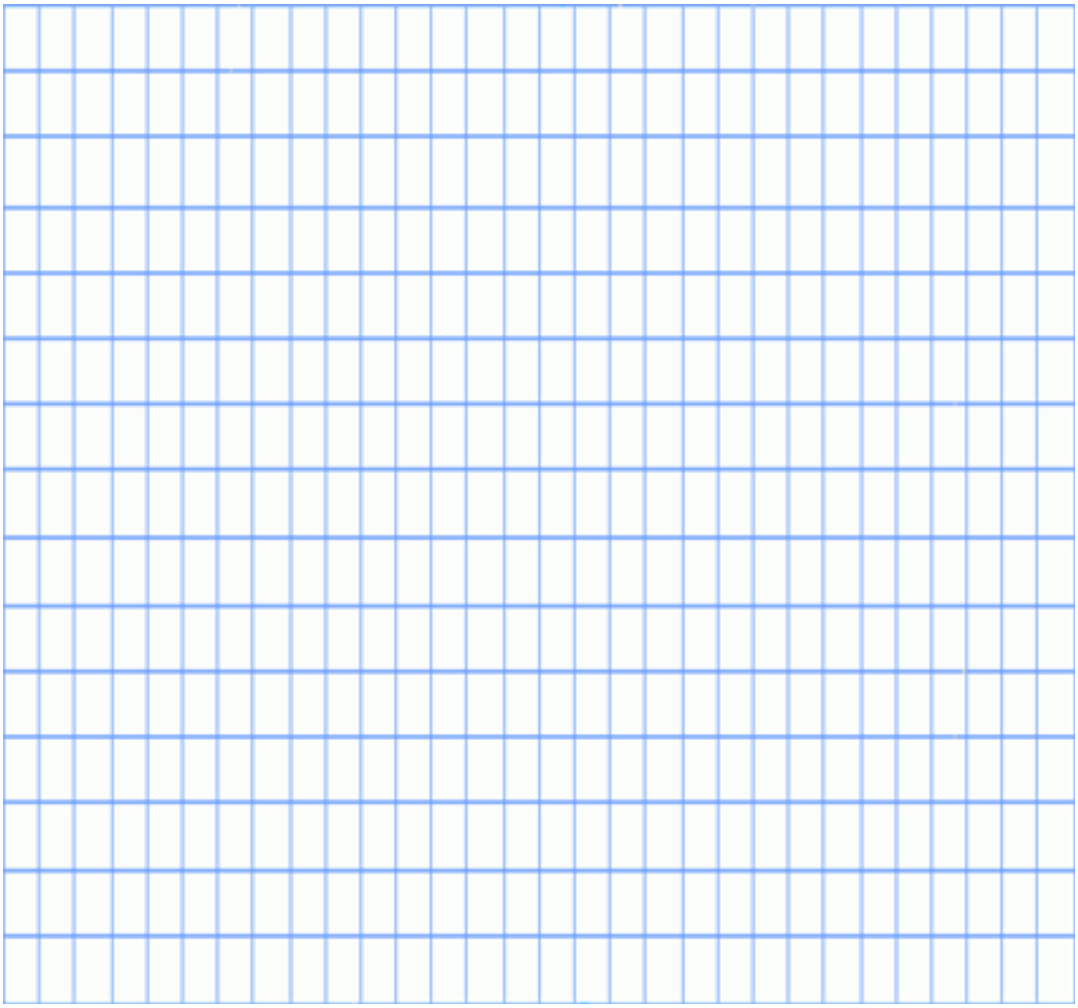
If you do not understand, try starting from scratch!

CLASS 3

“IF” syntax utilization

Ultrasonic Piano Playback

Now you can refer to the previous script and create the rest of the script! It will be very easy from now on. Please set the scale value corresponding to each order correctly. You can continue to nest the 'if & else' syntax.
Let's create a script in the space below.



“IF” syntax utilization

Ultrasonic Piano Playback

Have you created a script for each kind of script? Once you have created the script, it will look like the one below.

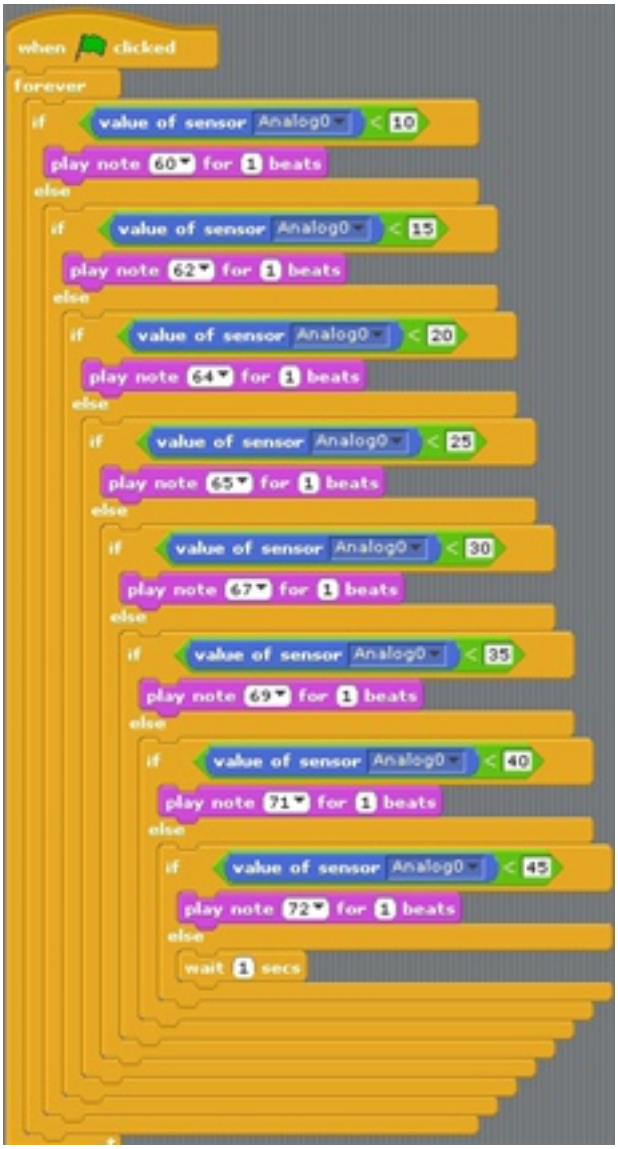
So what should I add to the last 'if & if' syntax? If the analog value is greater than '45', wait 1 second instead of playing the scale. Do you know what coding block to add? Complete the final script!

CLASS 3

"IF" syntax utilization

Ultrasonic Piano playback

The final script for the ultrasound piano will look like this!

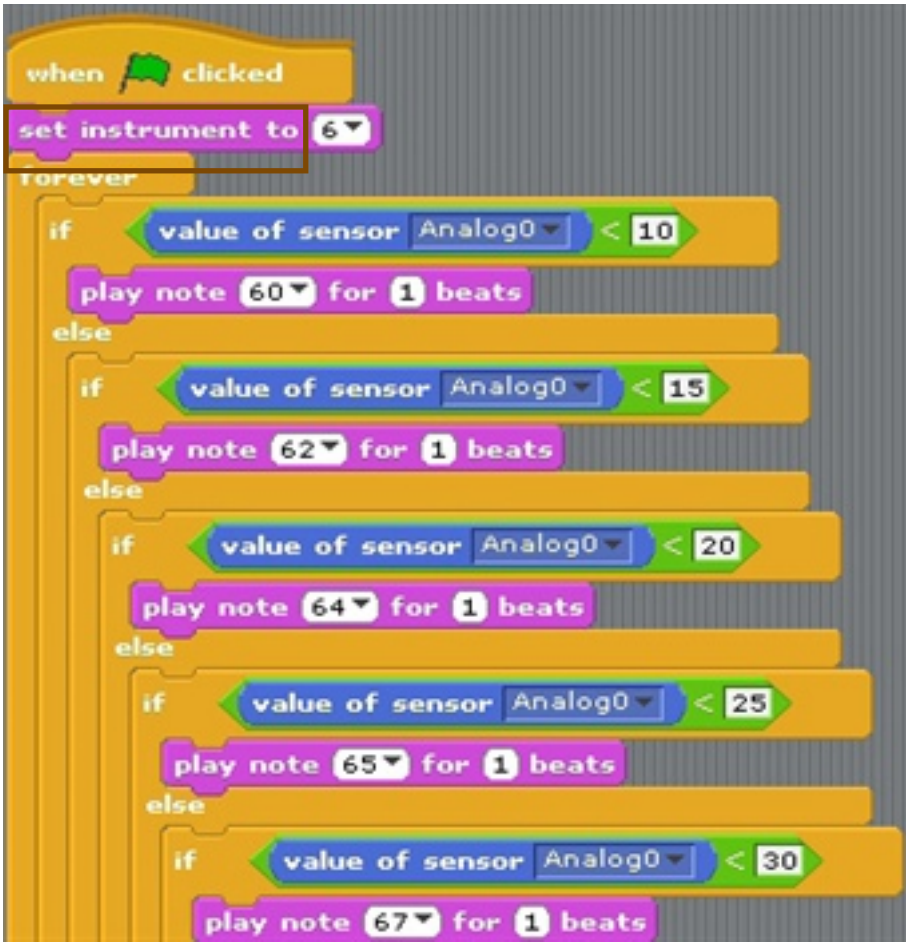


Replace the Ultrasonic Piano Instrument

Replace the Ultrasonic Piano Instrument

On the Sound tab, add the following block to your script!

Let's use another instrument instead of the piano in the above coding block. You can add this coding block just after the execution block of the script to make the sound of the instrument you want! It's not difficult, try it!



SCRATCH CODING KIT

Logic Boost

Line Tracer

LESSON 7

Activity

Line Tracer

Now you are going to make a comprehensive model using a variety of motors and sensors. Let's try to code the motion of the model by using infrared sensor, ultrasonic sensor and DC motor learned in the previous. The name of this model is the line tracer. A line tracer literally moves along a line. The line tracer you create moves along the black line.

We will try to express motion by combining two infrared sensors, an input sensor, and an ultrasonic sensor and two DC motors. The two infrared sensors will continue to search for the black line. One of the remaining ultrasonic sensors stops working every time there are obstacle blocks. And if you get completely out of the black line, it will stop all operation.

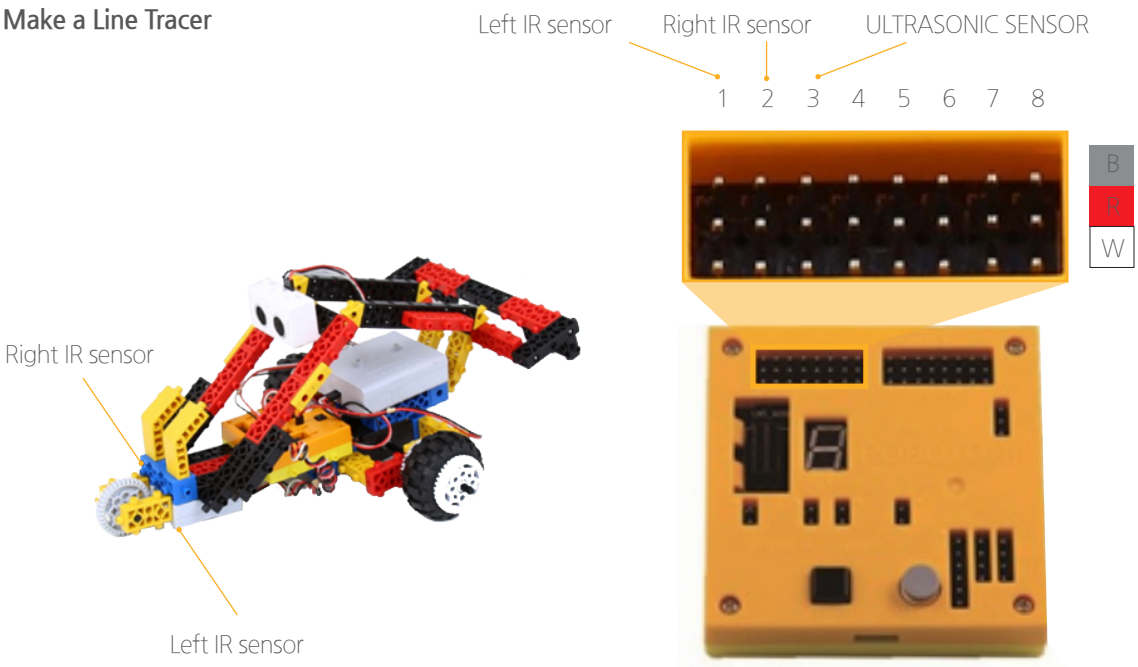
Can you imagine what kind of action it'll be doing?

CLASS 3

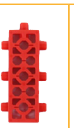
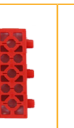
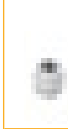
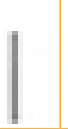
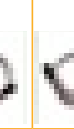
Activity

Line Tracer

Make a Line Tracer



Line Tracer materials

												
Diamond H8 x 1	Diamond V8 x 4	Diamond V6 x 2	Rubi 8 x 4	Rubi 7 x 6	Rubi 4 x 4	Rubi 6 x 9	Rubi 2 x 2	Rubi 0 x 7	Mini 2 x 12	Curve x 2	Link x 8	Triangle x 6
												
middle connector x 2	Sawtooth12 x 4	Sawtooth36 x 5	Off-road wheel x 2	A45 x 3	A64 x 2	Battery case x 1	DCmotor x 2	IR sensor x 2	Mainboard x 1	3pin connector x 3	4pin connector x 1	ULTRASONIC SENSOR x 1

Activity

1

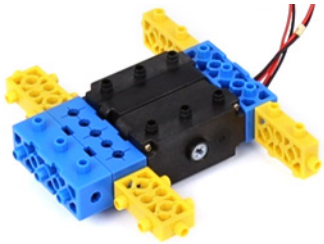
x 1

x 1

x 1

x 2

x 4



2

x 4

x 4

x 4

Make two of these

3

x 2

x 1

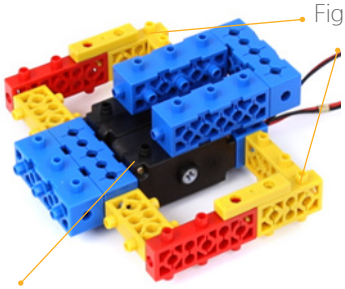


Figure 1

4

x 1

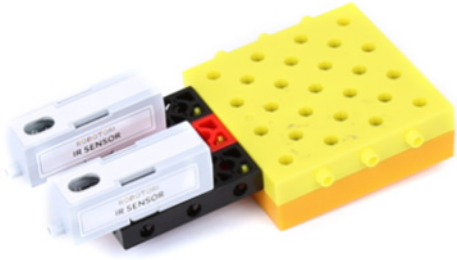
x 2

x 1

x 2

5

bottom



6

x 2

x 1

Figure 4

CLASS 3

Activity

7

x 2

x 1 A 45

x 1

x 2

8

Figure 7 Figure 8

9

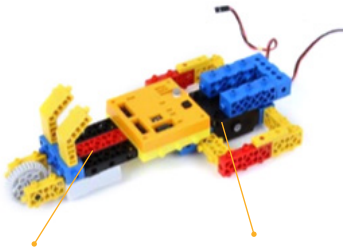
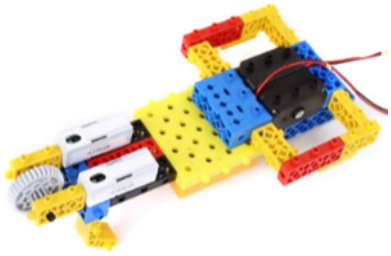


Figure 8 Figure 3

10



bottom

11

x 2
A 45

x 2
A 64

x 2

x 2

x 2

Figure 10 Figure 2

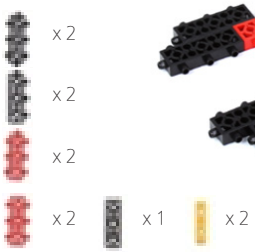
12

x 2

x 2

Activity

13



14

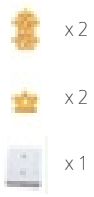
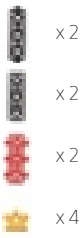


Figure 12

15

16



17

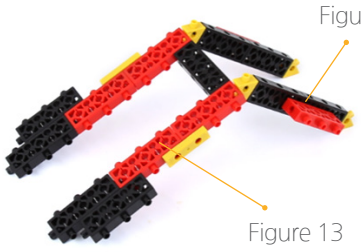


Figure 16

Figure 13

Complete

Figure 17

Figure 14

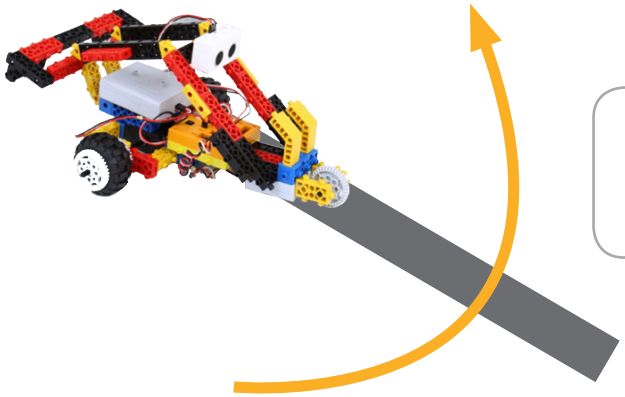
Figure 15

CLASS 3

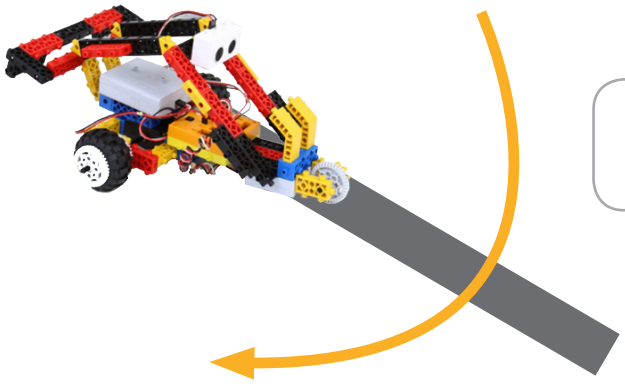
Activity

Coding the Line Tracer

Coding the line tracer. Do you remember the line follower model you learned earlier? Now, use two infrared sensors to code the line tracer on the black line. Each infrared sensor can tell if the car is on line or not. The program will use the following logic to move the car along the line.



1. If the left infrared sensor is on the black line and the right infrared sensor is not on the black line, we will activate the right motor and turn it to the left.

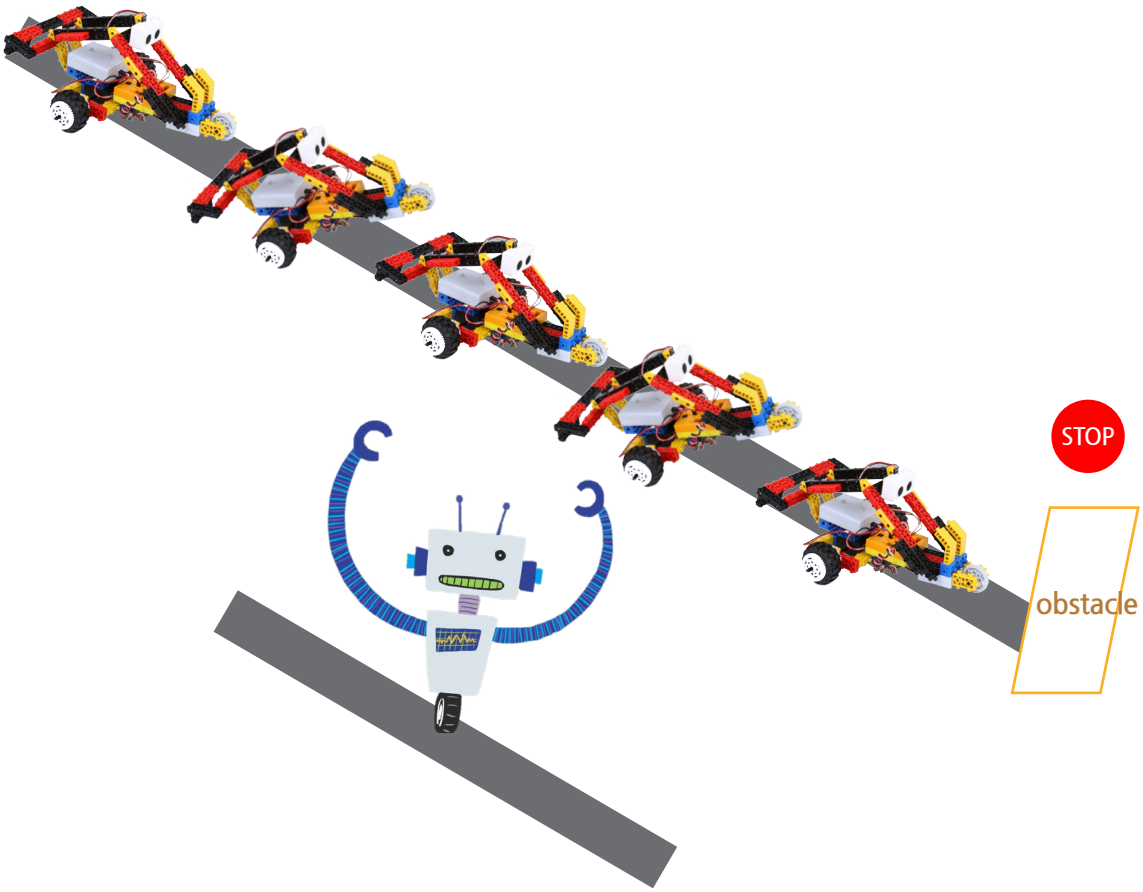


2. If the left infrared sensor is not on the black line and the right infrared sensor is on the black line, it will turn the left motor to the right.

Activity

Coding the Line Tracer

When the left infrared sensor detects the line (detects the absorption of light), the line tracer moves to the left, and when the infrared sensor on the right detects the line (detects the absorption of light), the line tracer will move to the right. If you detect a line (if there is less reflection of light in fact) you will continue to go straight. If the infrared sensor is no longer able to detect the line (if there is a lot of light reflection) or if it encounters an obstacle, it will stop.



CLASS 3

Activity

Coding the Line Tracer

Now you can code your own line tracer yourself. If the infrared sensor on the left of the two infrared sensors is white and the infrared sensor on the right is black, operate the left DC motor. If the infrared sensor on the left is black and the infrared sensor on the right is white, do. And, when the ultrasonic sensor detects the obstacle in the line tracer in progress, the operation should stop. All the movements will stop even if they leave the black line.

Use the 'if & else' conditional, sensor input and DC motor operation blocks to create your script!

Commentary

Coding the Line Tracer Ultrasonic sensor

Try coding when the infrared sensor on the left of the two infrared sensors is on the black servant and the infrared sensor on the right is not on the black line. If the infrared sensor is on a black subdivision, the analog value will be smaller, and if it is not over the black line, the analog value will be larger. I'll think of the threshold as '40' and try coding. And the action at this time should be to the left of the line tracer. You must stop motor # 4 and let motor # 7 forward.

When you create the above script, the infrared sensor on the left is on the black line. When the infrared sensor on the right is not on the black line, the motor stops at 4, the 7 motor goes forward and the line tracer goes to the left. Now let's see how the left infrared sensor is not above the black line and the right infrared sensor is on the black line. Please create the script below.

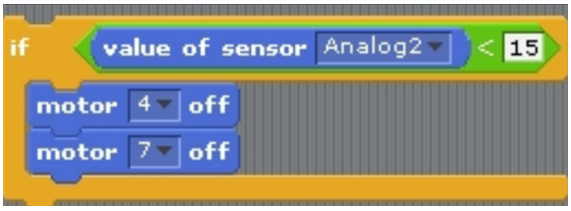
When you create the above script, the infrared sensor on the left is not on the black line. When the infrared sensor on the right is on the black line, motor 4 moves forward, stop motor 7, and let the line tracer go to the right. Now, let's create a script for an ultrasonic sensor. If the line tracer follows the black line and meets an obstacle, you must stop using the ultrasonic sensor.

CLASS 3

Commentary

Coding the Line Tracer Ultrasonic sensor

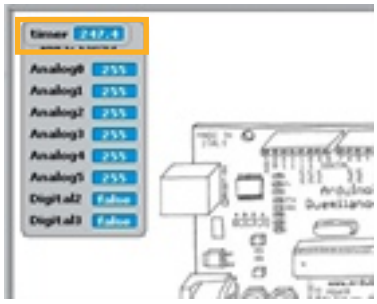
Now you're going to try coding for an ultrasonic sensor. The line tracer must stop when it encounters obstacles along the black line, which is very simple. When the distance of the object approaches the ultrasonic sensor, you can create a script that stops the two DC motors. It will be very simple.



If you create a script like the one above, the line tracer will work, and if it recognizes the obstacle through the ultrasonic sensor, it stops working. Very simple, isn't it?

So what if the line tracer is completely out of the black line? Unless both infrared sensors are on the black line. When coding, you have to consider a lot of situations. Because unexpected things can happen. If the infrared sensors are not all on the black line, the line tracer will have to stop. It should not continue to work.

If so, what criteria should you code for when you are not on this line? Here, we'll try to learn about timers. Click the Observation tab in the Scratch palette 

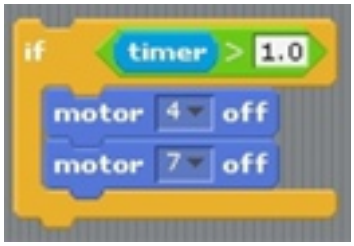


If you check 'Timer' on the Observation tab, you can see that time continues to flow on the stage screen. 'Timer' is literally a stopwatch inside the scratch. How do you code using this 'timer'?

Commentary

Coding Line Tracer Timer

Create a script that stops when the line tracer follows the black line and then goes off. It is very easy to use the timer I learned earlier.
If the line tracer leaves the line, the analog value will be lower for both infrared sensors. In other words, one is that the analog value is big and one is not the analog value is small. In this case, if you use the timer and the timer is waiting for 1 second, but the analog input of the infrared sensor is not correct, you have to stop all 2 motors.
Make a script below



If the timer is greater than one second, I will stop the motor 4 and the motor 7. That is, if the input values of two infrared sensors do not come in conditionally, the time of the timer will continue to flow, and if the time exceeds 1 second (if the input value of the infrared sensor that does not meet the condition is kept over 1 second) It is.

If so, if the line tracer works normally along the black line, then the timer's time should not flow. This is because the motor will stop when the time of the timer expires and it goes over a second. Therefore, if the line tracer is working properly, you need to continue to initialize the timer. At this time, we use the "timer initialization" block.

Drag the "Timer initialization" block onto the Observation tab and double-click on it. You can now see that the timer value on the stage is initialized to 0 seconds. Now we need to use the "Timer initialization" block in the script that runs the existing line tracer so that the timer does not run out of time. Try putting blocks in your previously created script.

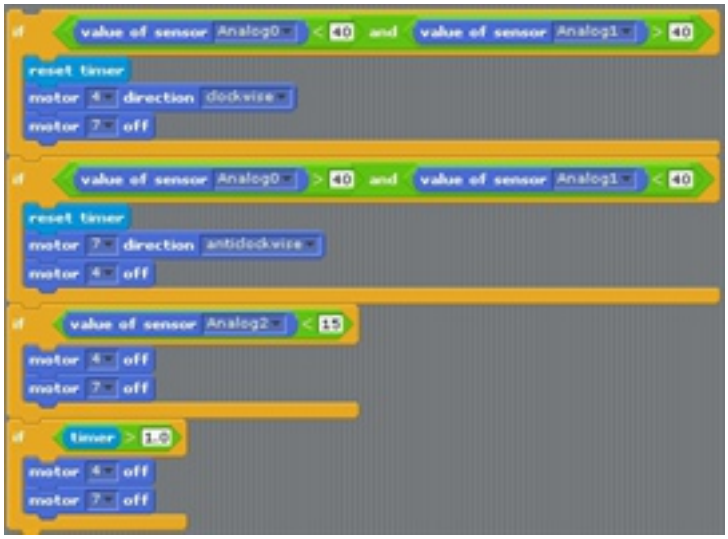
CLASS 3

Commentary

Coding Line Tracer

If you add a "timer initialization" block to your previously created script, it will look something like this:

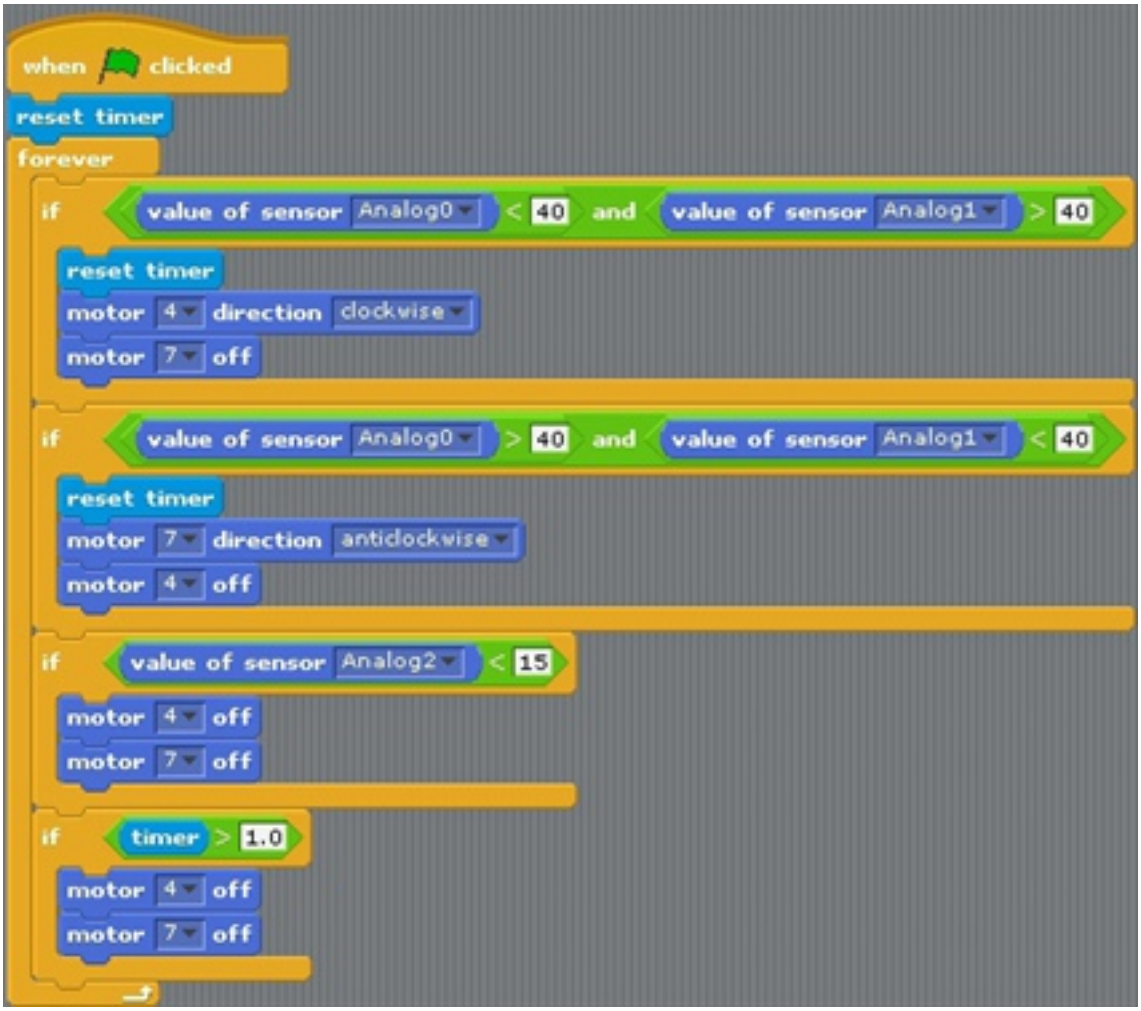
In this case, if the line tracer is working properly, the timer will be reset so that it will not exceed 1 second. It will not stop the motor because it does not exceed one second, right? Now combine your scripts. If you combine the scripts, it will look like this!



Commentary

Coding Line Tracer

If you combine all of the scripts created by various departments, you'll see something like this! It's a basic thing to celebrate endless repetition and execution blocks, so you can do it yourself. Just reset the timer one more time!

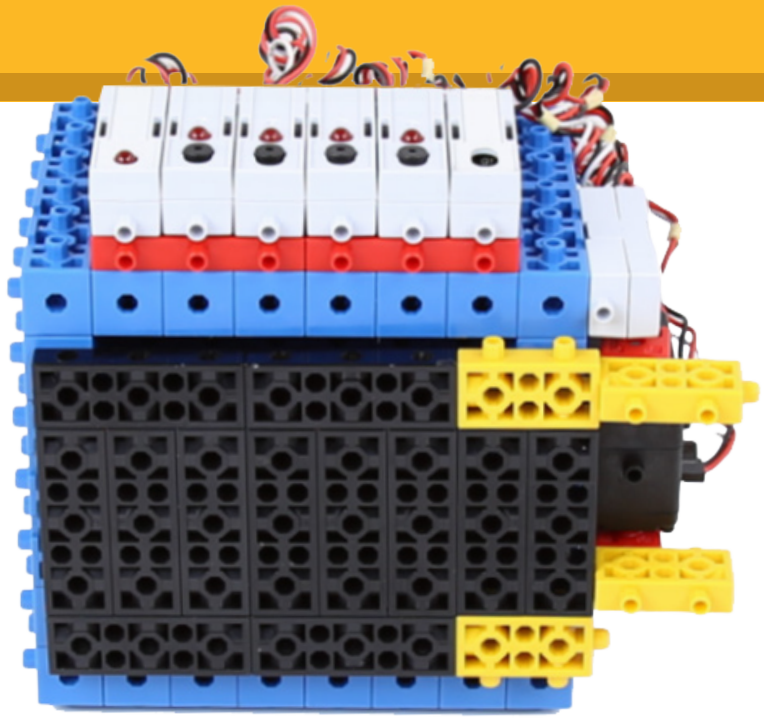


SCRATCH CODING KIT

Logic Boost

Tori Secret Safe

LESSON 8



Introduction

Tori Secret Safe

The Tori secret safe is a model that opens and closes the door of the safe only when you enter a number that matches the password specified in the scratch program. I will code the program using all four kinds of sensors (infrared sensor, button sensor, LED, sound sensor). At first, you can enter numbers from 1 to 4 using the button sensor connected to the ports 0 to 3 of the analyzer with the door open. When you press the infrared sensor after inputting the number, if the number matches the designated password, the sound sensor makes a sound and the door is closed. If the number does not match, the door does not work. The LED is blinking when the door is closed. When the door is closed, pressing the button sensor according to the specified password and inputting it to the infrared sensor, it is the key to this activity to make the door open with the sound of the door open and the LED continuously on.
Use one infrared sensor, four button sensors, one LED sensor and one sound sensor in the Tori secret safe. 1 The new infrared sensor opens and closes the safe door. The four button sensors are used to enter the passwords from 1 to 4, one LED sensor indicates whether the door is open or closed, and one sound sensor is used when the door is moved Everyone makes a sound. Remember this and try coding.

Because there are four different kinds of sensors, there are many things to consider when coding. In this model, you will learn a variety of coding skills.

Coding Tori Secret Safe (Principle)

I'll take a look at the actions you need to take into account before coding Tori secret safe in detail. Before we do this coding, it is very important to have a coding process with general guidelines on what actions we should express and how we can express them. Especially, if the code is complicated, the more complex it is, the more useful it will be when coding the actions that are expressed.

Let's put together the basic actions we need to implement on the next page.

CLASS 3

Introduction

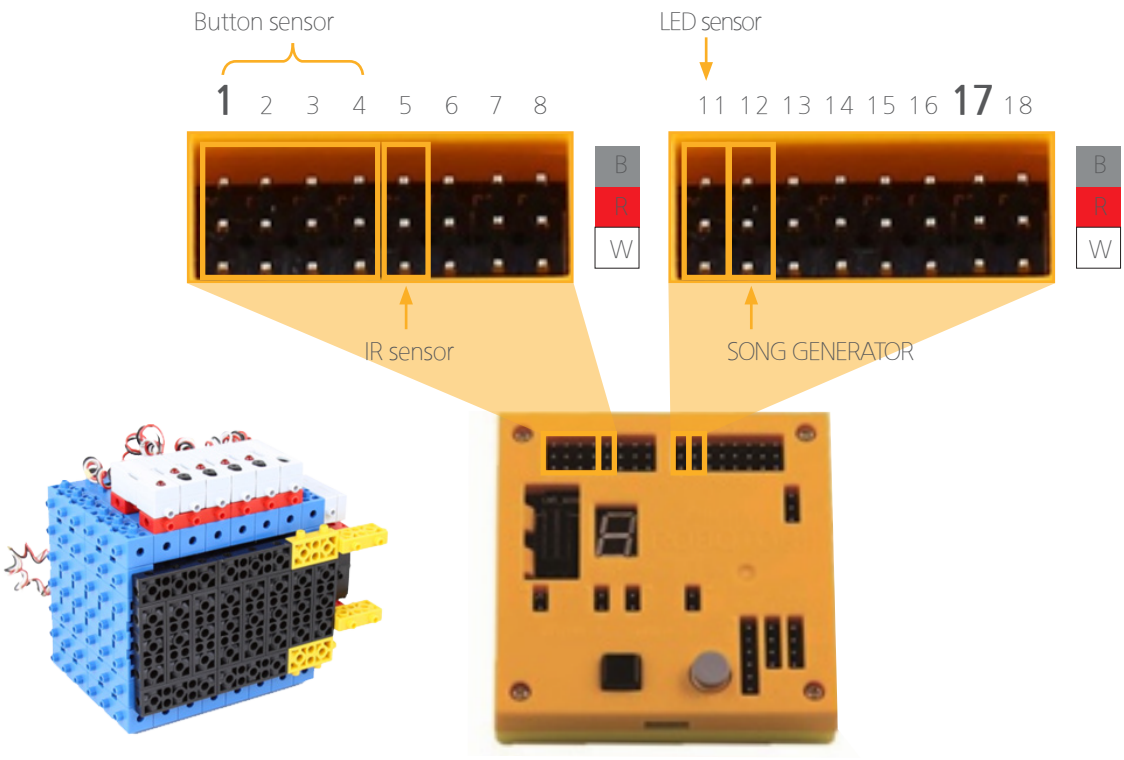
Coding the Tori Secret Safe (principle)

There are a lot of behaviors to express in the Tori secret safe. Since the program is not simple, we need to make sure the code is clean before doing the complicated coding. Let's try to arrange these actions once on this page.

1. With the door open, enter the numbers from 1 to 4 as the password when the button sensor connected to the analog input ports 0 ~ 3 is accessed. After inputting, press the infrared sensor connected to the analog 4 port to compare with the saved password. If the entered number matches the password, the door closes; otherwise, the door does not close.
2. When the door is closed, the sound sensor is plugged into the digital 11 input port and the LED sensor plugged into the digital 10 port blinks.
3. In the same way, when the door is closed, input the number matching the saved number with the sensor, press the infrared sensor, compare with the password, and the door will open.
4. If the number you entered does not match the password, the sound sensor sounds a short two times.
5. If the entered number matches the password, the door opens and the sound sensor makes a sound. The LED sensor is still on when the door is open.

Now, shall we start building our own secret safe?

Activity



Secret Safe materials

								
Diamond V8 x 40	Diamond V6 x 22	Rubi 7 x 2	Rubi 6 x 2	Rubi 4 x 6	Rubi 2 x 2	Rubi 0 x 4	Rubi 2 x 4	Servo x 2
								
Servo motor x 1	Mainboard 128 x 1	Battrtey case x 1	Battrtey case x 1	IR SENSOR x 1	LED SENSOR x 1	BUTTON SENSOR x 4	connector x 9	

CLASS 3

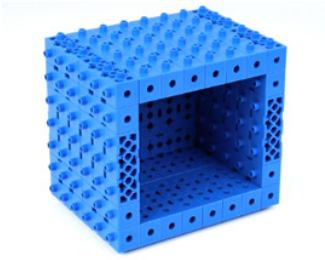
Activity

Making Secret Safe

1

x 40

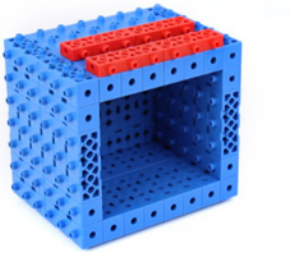
x 22



2

x 2

x 2



3

x 6

x 2

x 4

x 4 x 2 x 1 x 1

4

x 1

x 1

x 1

x 1 x 1 x 4 x 8

5

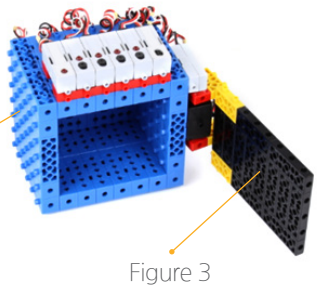
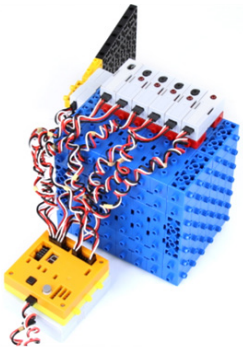


Figure 2

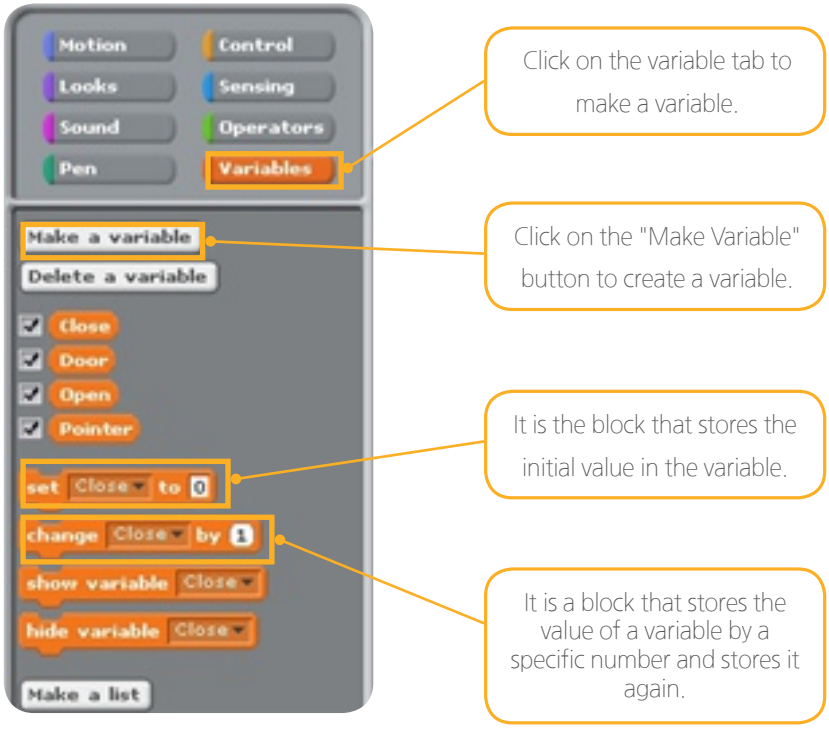
Figure 3



Activity

Tori secret safe variable coding

We need a lot of things to make a script for the Tori secret safe. The important thing is 'variable'. Do you remember? In scratch, you can think of variables as storage.
You name the variable (storage space) and put the value into that variable. In the Tori secret safe model, we have to use a total of four variables. If you have a lot of variables, you can improve your readability when coding by giving a proper name for the purpose of the variable.



Tori Secret Safe Variable Coding

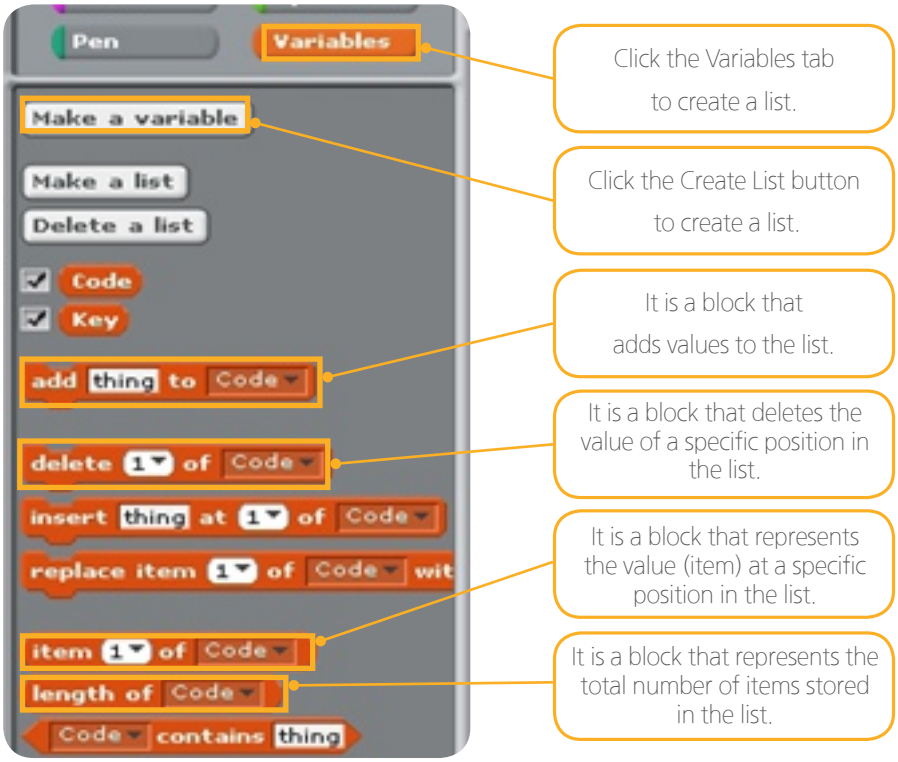
Create a variable as shown in the figure above. We will deal with the above variables while coding. The name of each variable is made considering the role, so please make it the same as the above.

CLASS 3

Activity

Tori secret safe variable coding

We need a lot of things to make a script for the Tori secret safe. The important thing is 'variable'. Do you remember? In scratch, you can think of variables as storage.
You name the variable (storage space) and put the value into that variable. In the Tori secret safe model, we have to use a total of four variables. If you have a lot of variables, you can improve your readability when coding by giving a proper name for the purpose of the variable.



Tori Secret Safe Variable Coding

Create a variable as shown in the figure above. We will deal with the above variables while coding. The name of each variable is made considering the role, so please make it the same as the above.

Activity

Tori Secret Safe List coding

We need to use 'list' to create the ability to store passwords for the Tori secret safe. A list is a function that allows you to save as many values as you want. The list you need is a list of 'input numbers' and a list of 'passwords'. The Tori secret safe must store the number of button sensors you press in the list for 'input number' in order and then store the pre-stored password in the list for 'password'.

Tori secret safe basic coding

Drag the blocks as shown in the picture above to create a script. The first block means that you will run the script when you hit the flag icon on the scratch screen. And the 'Delete all items in code' block, which is linked to the Mint, means to delete all items in the 'Code' list. It removes all of the previously entered numbers or any remaining items. The role of the 'Delete all items in the key' block also removes all items in the 'Key' list.

Now try setting the password. The list of passwords is on the 'Key' list. We'll put '1' in the first position, '2' in the second position, '3' in the third position, and '4' in the fourth position in the list. At this time, we use the 'insert ~ to ~position in Key' block. We put a specific value in a specific position in the 'Key' list using this block.

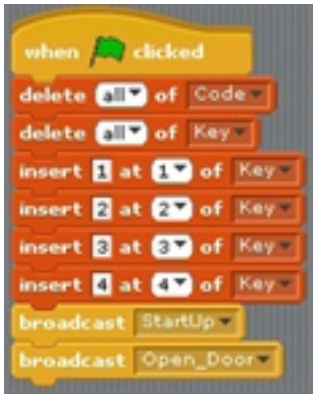
For example, the 'Put 1 in Key' block means to save the value of the first item in the 'Key' list as '1'. What you have coded above is that when you click the flag icon, you delete all the contents of the 'Code' and 'Key' lists and save the password '1234' in the 'Key' list.

CLASS 3

Activity

Tori Secret Safe Activation Coding

Once the password is set, you have to send a signal to start the full program. At this time, we use the 'broadcast' block. As we discussed earlier, 'broadcasting' means sending signals in scratch. If you do a broadcast, you have to get a broadcast.



Tori Secret Safe Broadcasting

The above picture is a script with the 'Broadcast' block added. After deleting all the list items and saving the password, 'Stratup Broadcast' and 'Open_Door Broadcast' blocks are executed.

When 'StartUp' and 'Open_Door' are broadcast, they have to be broadcasted. The above picture is 'When I get StartUp' and 'When I get Open_Door'. And below that is the script that will be run after that. In other words, if you delete all the items in the list and save the password as '1234', you will be broadcasting 'StartUp' and 'Open_Door'

Activity

Tori Secret safe basic setting coding

We'll create a script when we receive the broadcast, with the 'Broadcast' block we learned earlier. First, create a script when you receive the 'StartUp' broadcast. The role of 'when receiving StartUp' script is to set the initial value of the variable and to set the other operation of the LED sensor whether to open or close the door. Let's go ahead and make it!

Create a script like the one above. 'Close' is the angle of the servo motor when the door is closed, and 'Open' is the angle of the servo motor when the door is opened. After setting the initial values of these two variables, try coding the operation of the LED sensor. The LED sensor will flash every 1 second if the door is closed and stay on if the door is open. The coding of the operation of the LED sensor will be discussed in detail on the next page.

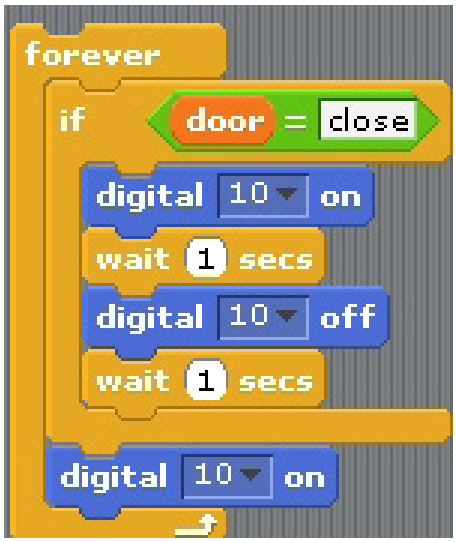
Do you remember the block that runs the servomotor? Try to assign multiple values to this block (0 ~ 180) and check the angle when the door is opened and closed. We need to store the value of the Closed variable in the 'Close' variable, and the value of the variable in the 'Open' variable when the statement is opened. Since the angle of the servomotor you have is different for each product, you need to check the angle. It is very important to check the figures first because it can cause errors in actual operation!

CLASS 3

Activity

Tori Secret Safe LED Coding

Let's take a closer look at the script for the LED sensor behavior in the 'StartUp' script we learned earlier. At the end of the 'StartUp' script, there is a big script to see how the LED sensor works. The role of this script is to keep the LED sensor on continuously when the door is open and to blink the LED sensor every 1 second if the door is closed.



Make a script like the one above.

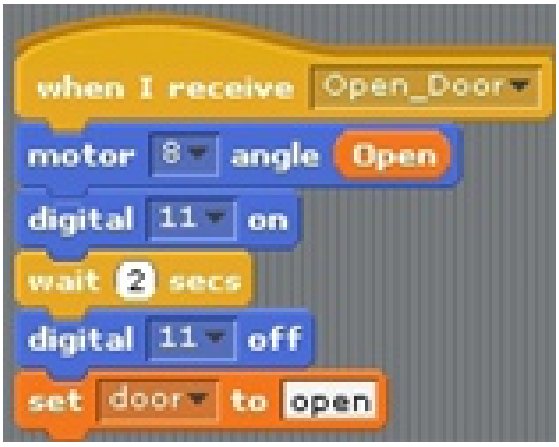
First, the 'infinite loop' block means that we will continue to check the script in this block indefinitely. Inside the 'infinite loop' block, there is a 'if' block, which executes the scripts contained inside when certain conditions are met, ie, the value stored in the 'door' variable = close, then activate the internal script. If the 'door = close' condition is met (if the door is closed), turn on Digital 10 output (turn on LED). Wait for 1 second with the LED on, then turn off Digital 10 output (turn off the LED). And wait another second. This will cause the LED sensor to flash at 1 second intervals when the door is closed. However, if you do not satisfy the 'door = close' condition, ie if the door is open, keep the Digital 10 output on. (The LED is still on).

So, we learned about the operation of the LED sensor. On the next page we will try to code the script when we receive the 'Open_door' broadcast (when the door is open).

Activity

Tori Secret Safe 'Open_door' coding

Now let's create a script when we receive the 'Open_Door' broadcast. The role of the 'Open_Door receiving' script is to open the door and play the sound sensor for 2 seconds. Let's make it



Create a script like the one above. First, open the door by activating the angle of motor 8 (servomotor) by the value stored in 'Open' variable. After that, turn on Digital 11. (Turn on the sound sensor.) Turn off Digital 11 output after playing for 2 seconds. (Turn off the sound sensor). And we store 'open' in the 'door' variable. The 'door' variable is a variable that tells you the current state of the statement.

Let 's try to organize the process so far again. First, delete all items in the 'Code' and 'Key' lists and save the password '1234' in the 'key' list. Then, set the angle when the door is opened and closed. If the door is open, the LED is on. If the door is closed, the LED will flash every 1 second. The first state starts with the door open. The servomotor operates at a preset angle to open the door, and the sound sensor is played.

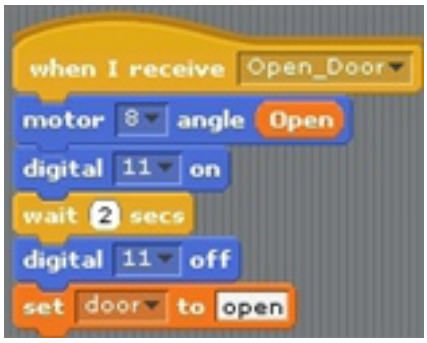
We created a script for the door when it opened. So, the script when the door is closed is very similar, right? On the next page, you can create a script when the door is closed.

CLASS 3

Activity

Tori Secret Safe ‘close_door’ Coding

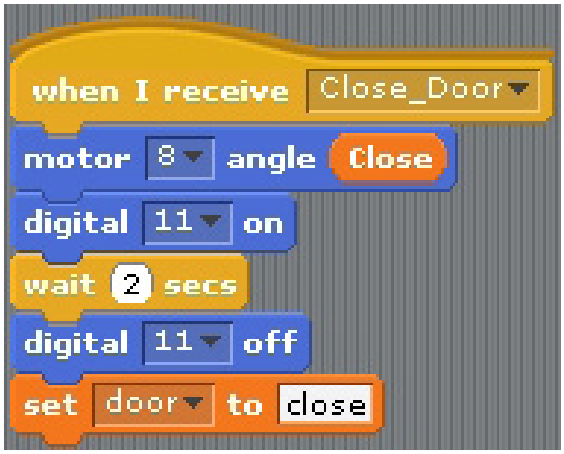
Use the script on the right to code the script when the door is closed. If 'Open_Door' was a broadcast when the door was opened, if it was closed, try coding it to start with 'Close_Door' broadcasting. Think about how you can change the word 'open'!



Activity

Tori secret safe 'Close_door' coding

You have learned that the broadcast that opens the door in the front is 'Open_Door'. Now create a script when you receive a broadcast that closes the door. The name of this broadcast is 'Close_Door' for associativity. The role of the 'Close_Door Receiving' script is to close the door and play the sound sensor for 2 seconds. Let's go ahead and make it.



reate a script like the one above. First, close the door by activating the angle of motor No. 8 (servo motor) by the value stored in the 'Close' variable. After that, turn on Digital 11. (Turn on the sound sensor). After playing it for 2 seconds, turn off digital output 11 (turn off the sound sensor) and then store 'close' in the 'door' variable. The 'door' variable is a variable that indicates the current state of the statement.

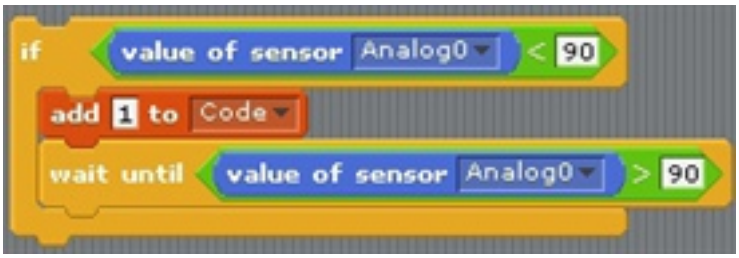
We created a script that closes the door. Now the script for the door opening and closing actions is ready. On the next page, we will learn coding by inputting numbers from 1 to 4 by using button sensor and infrared sensor in earnest.

CLASS 3

Activity

Enter the Tori Secret Safe Number

Now let's learn coding using the button sensor and the infrared sensor to enter the number. We'll add the number to the 'Code' list when we press the button sensor once.



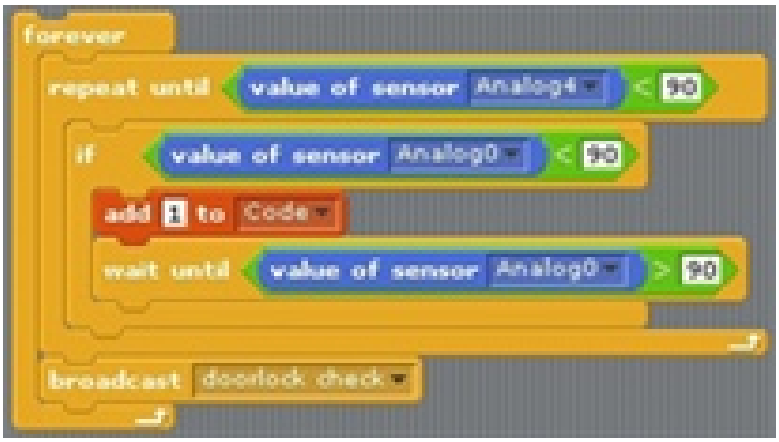
Make a script like the one above

First, use the 'if' block. It is a block that executes an internal script when certain conditions are met. The condition here is the value of the analog 0 sensor <90. There are four button sensors connected to the current input analog ports 0 ~ 3. Here we will create a script for the button sensor plugged into the input analog port 0. The button sensor has a smaller value when pressed. 'Analog 0 sensor value <90' can be satisfied when the button sensor connected to input analog port 0 is pressed. In other words, if the value of analogue 0 sensor is <90, the meaning of the block is 'if the button sensor connected to input analogue port 0 is pressed'.

If this condition is satisfied, add '1' to the 'Code' list. At this time, if the analog 0 button sensor is still pressed, it will add '1' to 'Code' list, so we use 'Wait' block. This block is a block waiting for a certain condition. If the user presses the analog 0 button sensor, it will wait until it is released before proceeding to the next script. At this time, the condition will be 'Analog 0 sensor value> 90'. That is, if the value of the analog 0 sensor is less than 90 (if the button sensor is pressed), add 1 to the Code list and wait until the value of the analog 0 sensor is greater than 90 (until the button sensor is released).

Activity

Enter the Tori Secret Safe Number



Create a script like the one above using the script you created in the previous page.
There is an 'iterate through' (repeat until) block in the 'infinite loop' block that keeps repeating the script inside. 'Repeat to' (repeat until) block is a block that executes internal scripts until a certain condition is reached. As shown above, the condition of the block 'to repeat' is 'Analog 4 sensor value <90'. Input Analog port 4 has an infrared sensor connected. In other words, it keeps running the internal script until the analog No.4 port infrared sensor is smaller than 90 (until the infrared sensor is pressed). I've covered the script inside the 'repeat to' block above.

If the Analog 4 port infrared sensor is pressed, it will exit the block until 'Repeat to'. After that, it will broadcast 'doolock check'. 'doorlock check' is a script that activates the door if the '1234' password is present, or not. I'll take a closer look at 'doorlock check'.

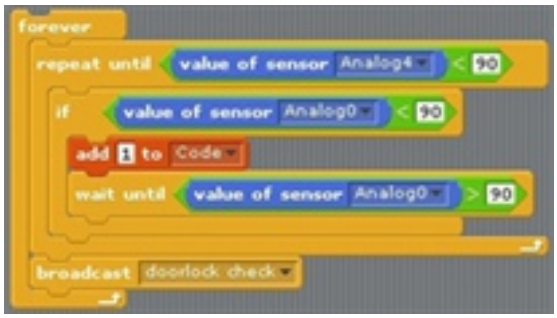
We have now created a script that will enter the number with the button sensor connected to the analog 0 port, inform the infrared sensor connected to the analog 4 port that the number input is finished, and compare it with the password. Now let's create a script that connects to ports 1 to 3 analog!

CLASS 3

Activity

Enter Tori Secret Safe Number

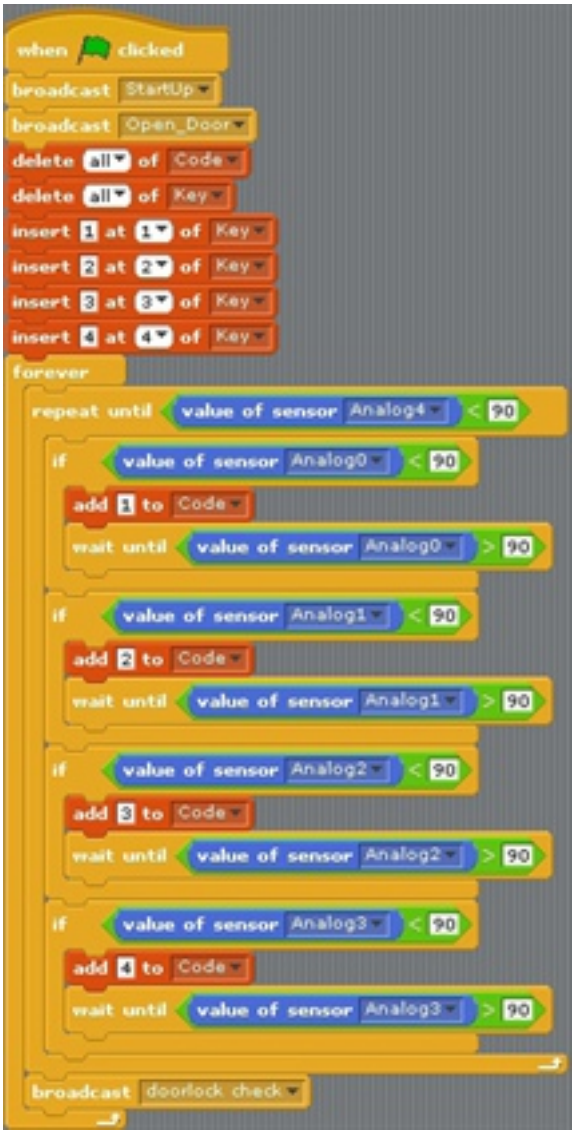
When using the script on the right and pressing the button sensors of analogue 1 ~ 3, try to save the numbers from '2' to '4' in the 'Code' list. Code the script when it's closed. Please note that the port number of the analog sensor changes and that the number to be added to the list changes!



Activity

Tori Secret Safe Activation Coding

The script in the executable part of the Tori secret safe will look like this:

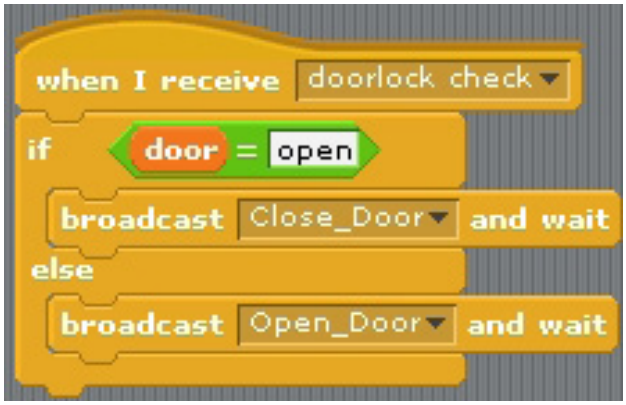


CLASS 3

Activity

Tori secret safe door motion coding

Now let's create a script when we receive the 'doorlock check' broadcast. The role of the 'when doorlock check is received' script is to broadcast a message that the door is closed if the number entered when the door is open matches the password. If the number entered when the door is closed matches the password, the door will open. Let's go ahead and make it!



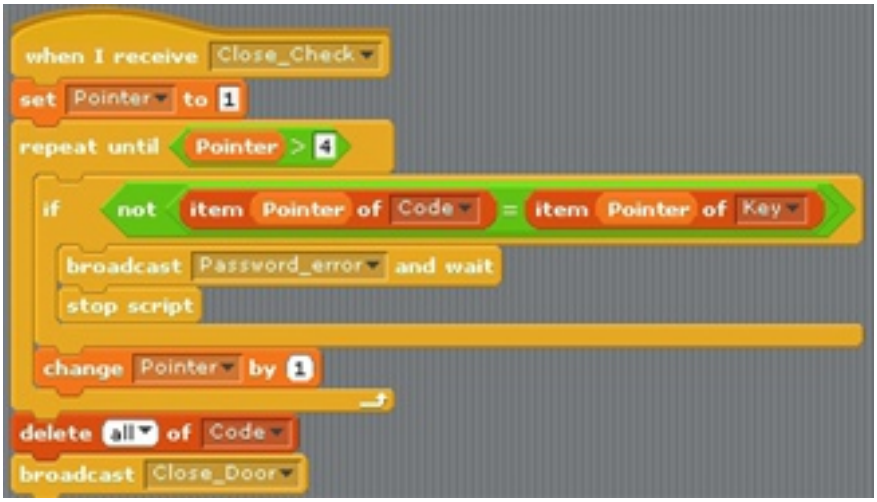
Create a script like the one above.
First, when you receive the "doorlock check" broadcast, this script is executed. In this case, we use 'if ~ or' blocks. This block can be found in the Controls tab. This block is similar to the 'if' block we learned earlier. Just the 'or~' part is added to the 'if' block. In other words, if it does not satisfy the condition of 'if' block, then the script contained inside 'or~' part will be executed. If it satisfies the condition of 'if' block, then the script inside the 'if' block will be activated and the script inside the 'or~' part will not.
If you meet the condition 'door = open' in the block or 'block' (if the door is open), you will go inside and broadcast 'Close_Check'. If you do not meet this condition, you can go inside and broadcast 'Open_Check'. 'Close_Check' is a script that closes a door if it is matched against a password and does not close a door if it is not. 'Open_Check' is a script that opens the door when matches the password but does not open the door if it does not match.

Now let's take a look at the two scripts that were broadcast on the next page.
The two scripts are very similar, so if you understand them well, you can easily code them.

Activity

Tori secret safe door close comparison coding

Now create a script when you receive the 'Close_Check' broadcast. The 'Close_Check Received' script's role is to compare the entered number against the password and close it if it matches, or 'Password_error' if it does not match. Let's make it once.



First, when you receive the broadcast, save '1' in the 'Pointer' variable. The reason for storing '1' is that this variable must compare the first item stored in the 'Code' list with the first item stored in the 'Key' list. If the first item matches, the variable is incremented by 1 and the second, third, and fourth items are matched. Save '1' and use 'repeat to block'. The condition is 'Pointer> 4'. If the 'Pointer = 4' condition is used, the 'Pointer' variable will become 4 and exit the 'Repeat to' block and compared with items up to item 3.

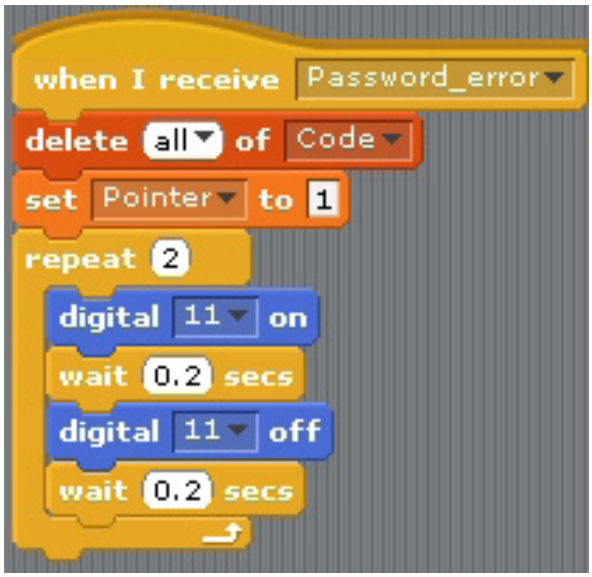
Inside the script use the 'if' block. The condition is "Item = 'Code Pointer' item = 'Key Pointer' is not item '. If the 'Code' list stored with the entered number and the 'Key' list stored with the password '1234' are compared, if they do not match, the condition of 'if' is satisfied, 'Password_error' is broadcast and the script will be stopped. However, as the 'Pointer' variable is accumulated one by one, the items in the 'Code' and 'Key' lists are the same, it exits 'repeat to~' block and deletes all items in 'Code' and broadcasts 'Close_Door'.

CLASS 3

Activity

Tori secret safe Password Inconsistent coding

Now let's create a script when we receive the 'Password_error' broadcast. The primary function of the 'Receive Password_error' script is to play the sound sensor twice at 0.2 second intervals when the entered number and password do not match. Let's go ahead and make them.



Make a script lie the one above.

When you receive a broadcast, delete all the items in the 'Code' list. It deletes all the input numbers so that they are not left. Then save the '1' back to the 'Pointer' variable. Think of these two blocks as initializing to re-enter numbers.

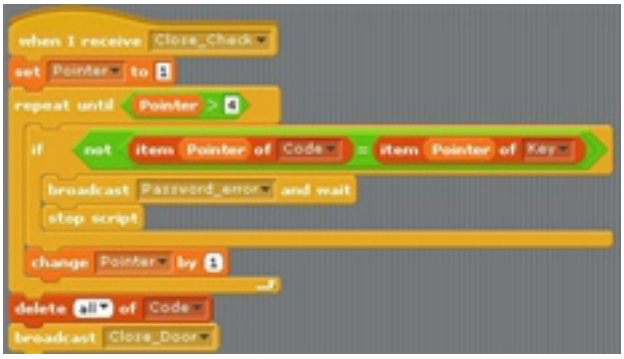
You should now play the sound sensor twice at 0.2 second intervals to let you know that the number entered does not match the password. In this case, let's use the block 'repeat ~ times'. It is a block that executes the internal script as many times as the condition. Turn on the digital output 11 and activate the sound sensor. Wait 0.2 seconds and turn on the Digital 11 output to stop the sound sensor. And wait another 0.2 seconds. Repeat this action twice and you'll hear a short 'beep beep' sound.
So, the script 'Password_error received' is a script that initializes the password when the input number and password do not match and tells the user if they have made a mistake.

Activity

Tori secret safe Door open Comparison Coding

Use a script on the right to code a script that compares the number and password you enter when the door opens.

Also, if 'Close_Check' was the broadcast when the door was opened, if it was closed, try coding it to start with 'Open_Check'

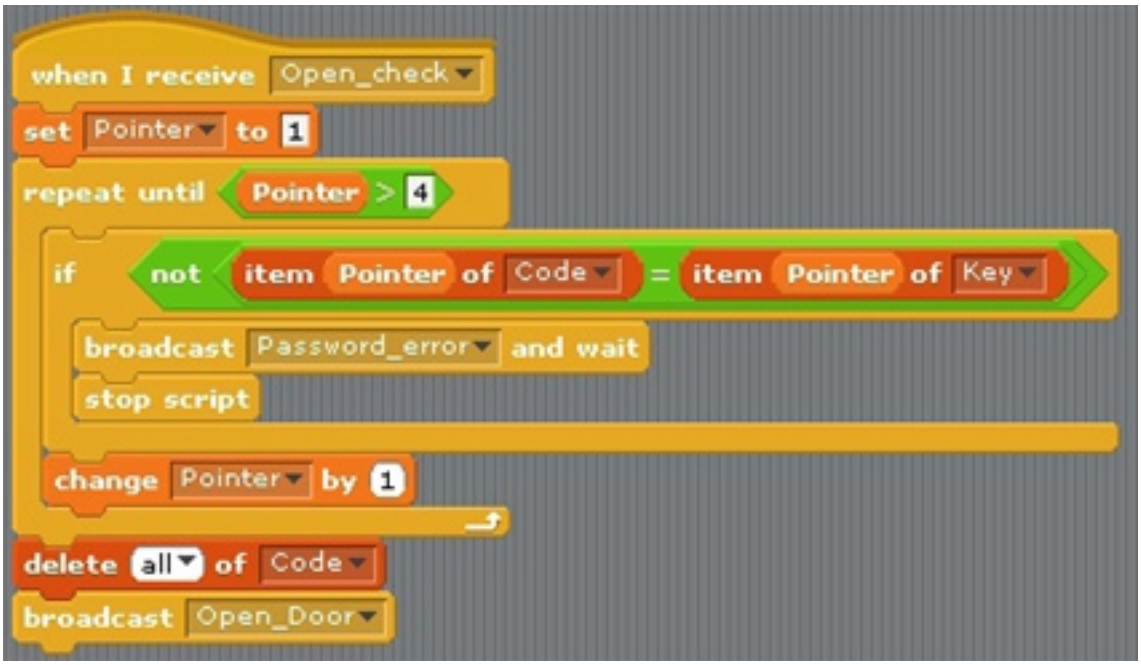


CLASS 3

Activity

Tori secret safe door open comparison coding

You should have learned that the broadcast you are comparing, when you close the door, against the password is 'Close_Check'. Now, create a script that compares passwords when you open the door. The name of this broadcast is 'Open_Check' for association. The role of the 'Receive Open_Check' script is to compare the entered numbers against the passwords to see if they match, and if it does not match, open 'Password_error'.

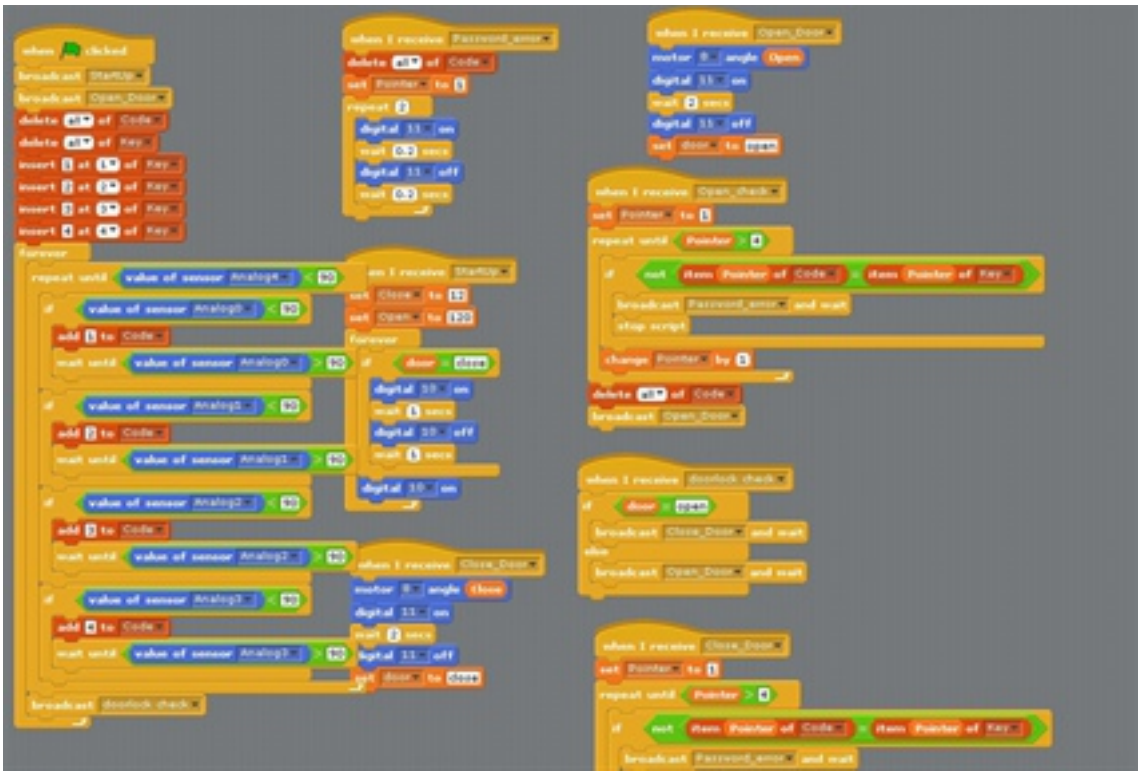


First, when you receive the broadcast, save '1' in the 'Pointer' variable. Save '1' and use 'repeat to block'. The condition is 'Pointer> 4'. If you use the 'Pointer = 4' condition, the 'Pointer' variable will be 4 when you click 'Repeat to block' because it exits the 'repeat to ~' block. Use 'If' block for internal script. The condition is "'item in Code Pointer position = item in Key Pointer condition' is not true". If the items in the 'Code' list where the entered number is stored and the 'Key' list in which the password '1234' is stored are the same, exit the 'repeat to~' block and delete all items in 'Code' and broadcast 'Open_Door'.

Activity

Tori secret safe entire coding

Now I have created all the scripts for the Tori secret safe. Do you understand well? If you do not understand well, please read it again and study. If you have been following so well, the entire script below will be complete!



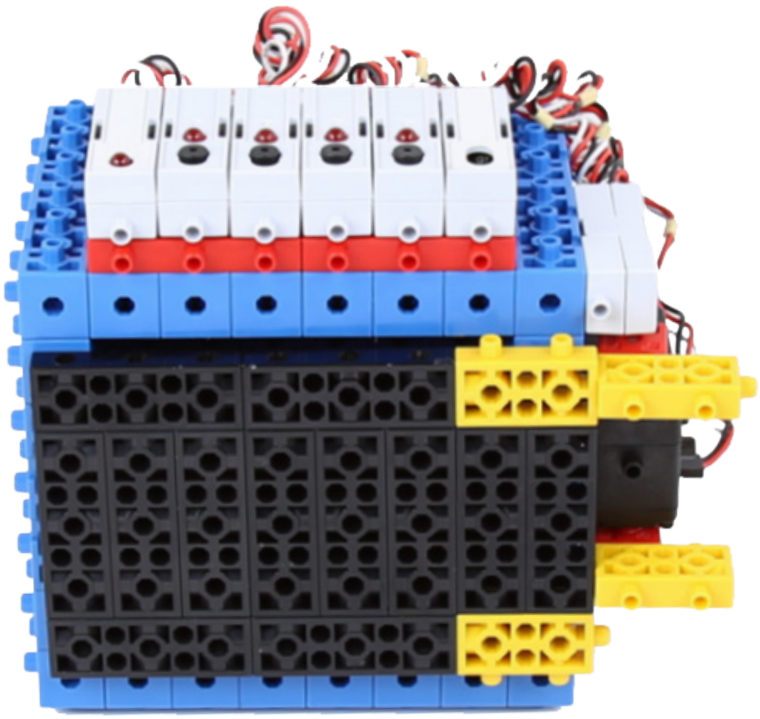
※ The whole script can be found at http://www.robotori.com/web_eng -> Moretips -> Manual ->EDU ->Logic boost CODING CLASS 3 ->download whole script

CLASS 3

Activity

Tori Secret Safe Play

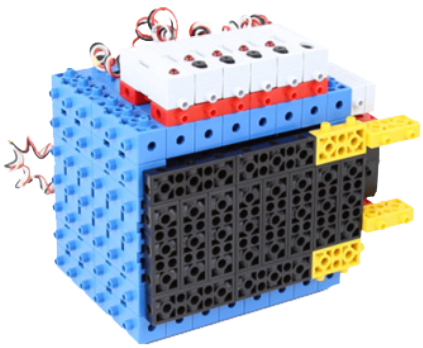
Now run the coded Tori secret safe program and see if it works. If it does not work, check to see if there is anything wrong starting from the beginning of the book. Let's unlock the code and open the safe.



Activity

Tori Secret Safe Problem 1

Does your coded safe work? If so, try a little application. Your current password is '1234'. Let's change the password now. Think carefully about what you learned earlier. Let's create the 4 digits of the desired password with numbers from 1 to 4.



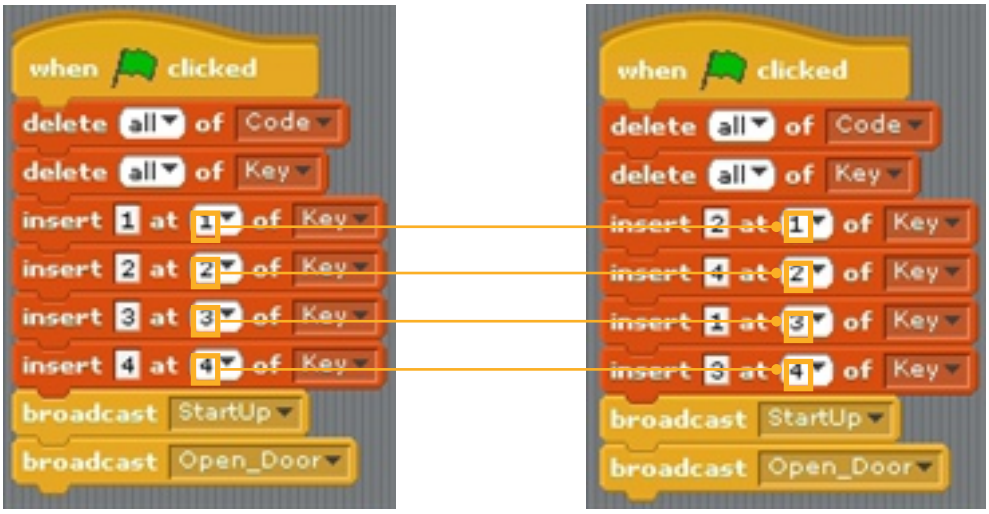
CLASS 3

Commentary

Tori Secret Safe Problem 1 commentary

Did you change the password?

Now you can enter 4 numbers from 1 to 4 as you have added '1234' in 'Key' list.

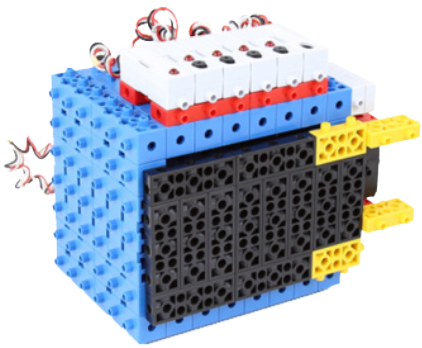


The above script puts '2', '4', '1', '3' in positions 1,2,3,4 of 'Key' list. If the existing items in the list are '1', '2', '3', or '4', just change the items in the list! Test that it works with the changed password!

Activity

Tori Secret Safe Problem 2

This time, we'll set the password to '1003'. Now we have a number from 1 to 4 and I have set and coded it. Please code how you would like to change the input number when other numbers are entered. Let's make it input when the password is set to '1003'.



CLASS 3

Commentary

Tori Secret Safe Problem 2 commentary

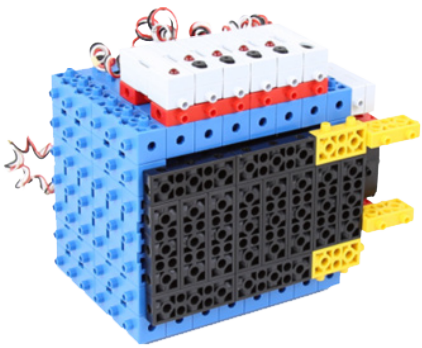
Change what you learned in question 1 to change the password to "1003" and change the input number from 0 to 3 instead of 1 to 4.

If you change only the numbers so that the input number is 0 ~ 3 instead of 1 ~ 4, the number of the button sensor changes from 0 ~ 3, so you can press '1003' to operate the door.

Activity

Tori Secret Safe Problem 3

Now let's set the password to 6 digits. Now we'll try to code from 1 to 4 so that our password is 6 digits. When you save your password in the 'Key' list, try to code 6 entries. Let's create a six-digit password.

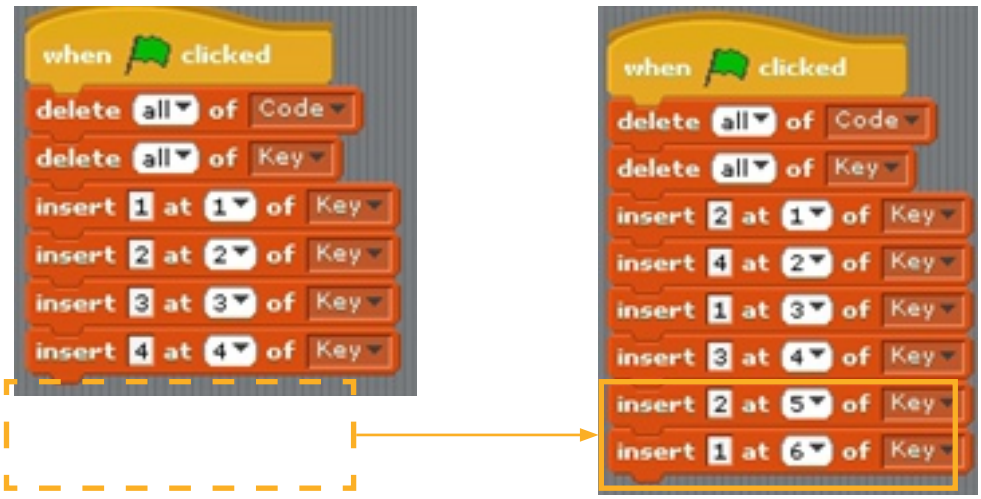


CLASS 3

Commentary

Tori Secret safe problem 3 commentary

Have you changed your password to 6 digits? Just add two more blocks to the script that saved the four-digit password stored in the 'Key' list and save a total of six passwords. You can enter 4 numbers from 1 to 4 as you have added 4 '1234'.

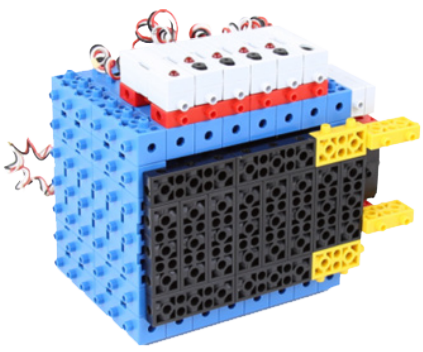


We can add two more blocks of 'input ~ to ~position of Key' to make my password six digits. You can pick 2 of the numbers from 1 to 4 and add them to positions 5,6 in the 'Key' list. Make your password six digits and see if the door of the safe works!

Activity

Tori Secret Safe problem 4

In this case, to open the safe, the user must input the button sensor and specify the password. Then, in order to close the door, the user must press the number corresponding to the password inputted by the button sensor, the door is opened. We need to create a variable that compares the number of digits in the password and a variable that stores the user's password. You will have to remove the part that sets the password first and broadcast it to save the password. Make it!

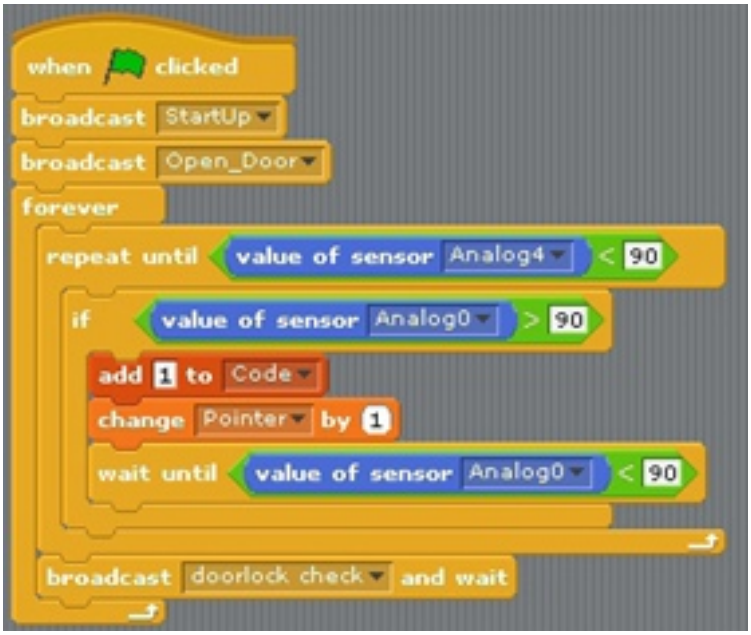


CLASS 3

Commentary

Tori Secret Safe Problem 4 commentary

Let's try to get rid of the part where we set the password in 'Key' list in the beginning because it is a way to input the password directly. The rest of the 'Broadcast' block is the same. We are only removing the part that sets the password.

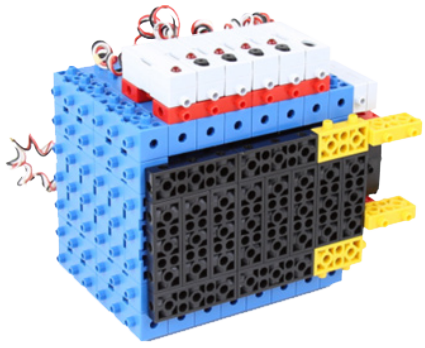


When you start the program, broadcast 'StartUp' and 'Open_Door'. And there is one thing that changed: when you entered the number with the button sensor, and every time you enter the number, it accumulates the 'Pointer' variable. The reason to increase the 'Pointer' variable every time you press the sensor is to store the 'Pointer' value in the number of digits.

Commentary

Tori Secret Safe Problem 4 Commentary

The activated script will look like below



CLASS 3

Commentary

Tori Secret Safe Problem 4 Commentary

Let's make a script for when you receive 'StartUp' broadcast.

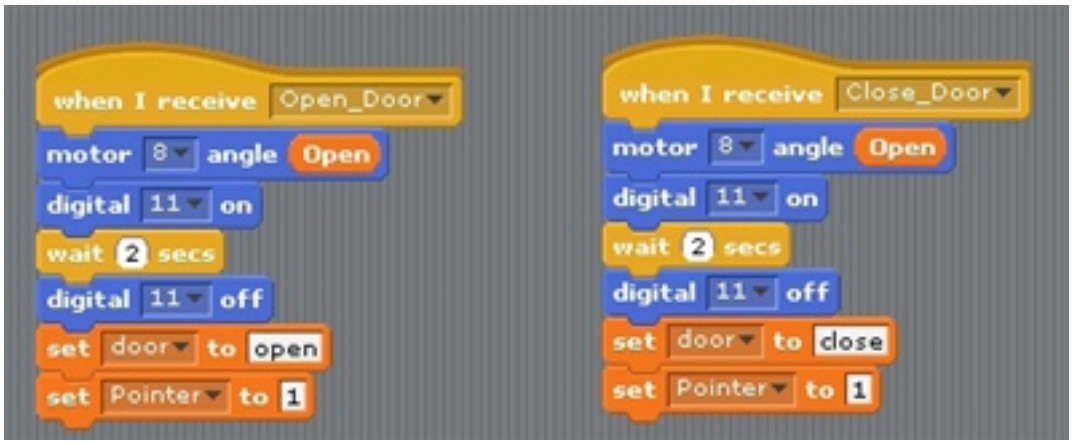
The role of the 'When received StartUp' script is to set the initial value of the variable, and to set the behavior of the LED sensor whether the door is open or closed.

Create a script like the one above. The 'keycnt' variable stores the number of digits. The 'index' variable stores the number entered in the 'Code' list into the 'Key' list. The 'Close' variable stores the angle of the servo motor when the door is closed, Open 'variable is the angle of the servomotor when the door is opened. Let's set the initial values of these four variables and then code the operation of the LED sensor. The LED sensor will flash every 1 second if the door is closed and stay on if the door is open. Please note that the coding of the operation of the LED sensor is covered in the previous section.

Commentary

Tori Secret Safe Problem 4 Commentary

Now let's create a script when we receive the 'Open_Door' and 'Close_Door' broadcasts. 'When you receive the Open_Door' script, it will open the door and play the sound sensor for 2 seconds. The role of the 'Close_Door Receiving' script is to close the door and play the sound sensor for 2 seconds. Shall we make it?



Create a script like the one above. It's very similar to the scripts you created earlier, 'Open_Door' and 'Close_Door'. The acting role is the same as the one we created earlier.

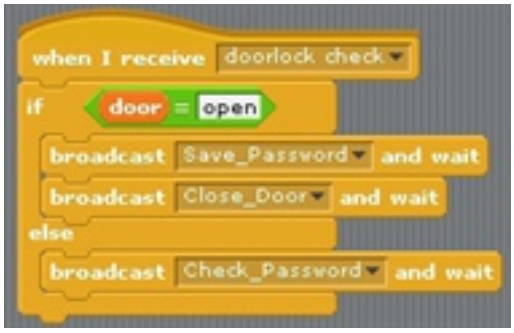
Now, at the end of the script, save 1 in the 'Pointer' variable and return it to its initial state. This is because once the door is opened or closed, it must be used to store the input number as a password, or to compare the stored password with the input number. The rest is the same.

CLASS 3

Commentary

Tori Secret Safe Problem 4 commentary

Now let's create a script when we receive the 'doorlock check' broadcast. The role of the 'when receive doorlock check' script is to save the entered number as a password when the door is open and close the door. When the door is opened, if the entered number matches the password, the close_door is broadcasted. When the door is closed, the entered number and password are compared. If they match, the open_door is broadcasted.
Shall we make it?



Create a script like the one above.
First, when you receive the "doorlock check" broadcast, this script is executed. In this case, we use 'if ~ or' blocks. This block can be found in the Controls tab. This block is similar to the 'if' block we learned earlier. 'or' block is added to 'if~'. In other words. If it does not meet the condition of 'if', then the script contained inside 'or' will be executed.

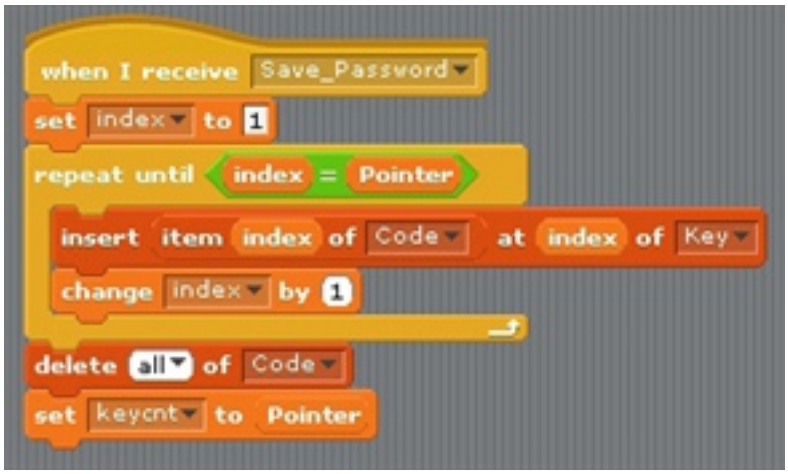
If the condition of 'if' is satisfied, the script contained in it is executed, and the script contained in 'or' is not executed. If you meet the condition of 'door = open' in block 'if' block (if door is open), go inside 'if' and broadcast 'Save_Password' and 'Close_Door'. If you do not meet this condition, go inside 'or' block and broadcast 'Check_Password'. 'Save_Password' is a script that will input number and save the passwordif the door is open. 'Close_Door' means to close the door. 'Check_Password' is a script to open the door if the door is closed if the number matches the password.

Now let's go on to the next page and learn about the broadcasted scripts.

Commentary

Tori Secret Safe Problem 4 commentary

Now let's create a script when we receive the 'Save_Password' broadcast. The main function of the 'Save_PasswordReceived' script is to store the entered number as a password. Here we use the variables 'index' and 'keycnt'.



Create a script like the one above. When receive the broadcast, it'll store 1 in the 'index' variable.

The 'Pointer' variable now has a cumulative number from 1 to the number entered by the user. Accumulate the value of the 'index' variable by 1 and save the number entered in the 'Code' list as a password in the 'Key' list in the order of the items until it becomes 'index = Pointer'. When 'index = Pointer', exit the block 'repeat to'. Delete all items in the 'Code' list. It deletes all the input numbers so that they are not left.

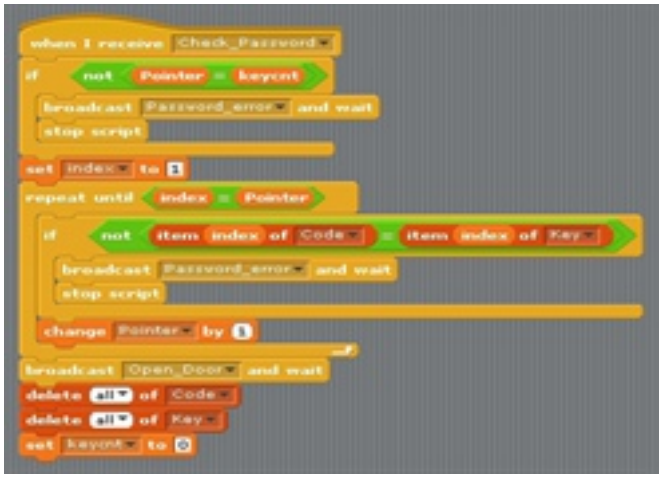
Then, save the value stored in the 'Pointer' variable in the 'keycnt' variable. 'Pointer' variables are accumulated from the number starting from 1, and they are stored in the variables that are responsible for the number of digits, so that it can compare digits when you input them again later.

CLASS 3

Commentary

Tori Secret Safe door close comparison coding

Finally, let's create a script when we receive the 'Check_Password' broadcast. When you receive the 'Check_Password' script, the role of the script compares the entered numbers against the passwords to see if they match, and if it does not match, it will broadcast 'Password_error'.
Let's make it.



Create a script like the one above. First, when we receive the broadcast, we compare the 'Pointer' variable, which is the number of digits entered by the user, and the 'keycnt' variable, which is the number of digits in the password. If the two values are not the same, broadcast 'Password_error' and stop the script. If the number of digits is not the same, the numbers entered in the list are not compared at first, and the password error notification is given. Then, store a '1' in the 'index' variable. Save '1' and use 'repeat to block'.

The condition is 'index = Pointer'. The current 'Pointer' variable stores the number of digits (number of items) that you enter. Increase the 'index' variable by 1 and compare the number 1 item in the list of 'Code' with the number of the item '1' in the list of 'Key' where the password is stored. If it does not match, broadcast 'Password_error' and stop the script. If the 'index' variable is incremented by 1, the two lists are compared. If all the items in all the positions are matched, exit the block 'Repeat to' to broadcast 'Open_Door' (open the door) After deleting all the items in the 'Key' list, save the initial value 0 in the 'keycnt' variable.

※ The whole script can be found at http://www.robotori.com/web_eng -> Moretips -> Manual ->EDU ->Logic boost CODING CLASS 3 ->download whole script

Summary

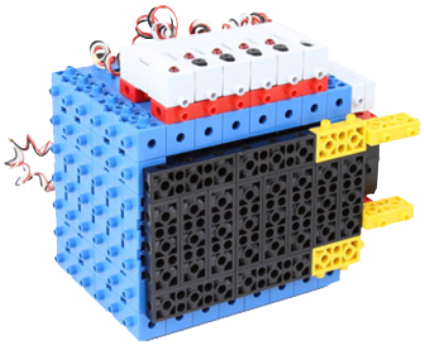
Tori Secret Safe Summary

How did you like making a secret safe in Tori? In addition to the four problems you have, there are some areas where you can do more applications depending on your imagination and creativity.

You can now handle all four sensors (button sensor, infrared sensor, sound sensor, LED sensor) freely. Think about various coding methods based on this Tori secret safe.

We have learned the various skills necessary for coding in this model, so I hope you will take one step further in the course of further and more detailed coding.

You can easily solve the problems that you have to solve in your life by applying the coding viewpoint. Please note that this coding is not limited to one field but can be used for various fields.

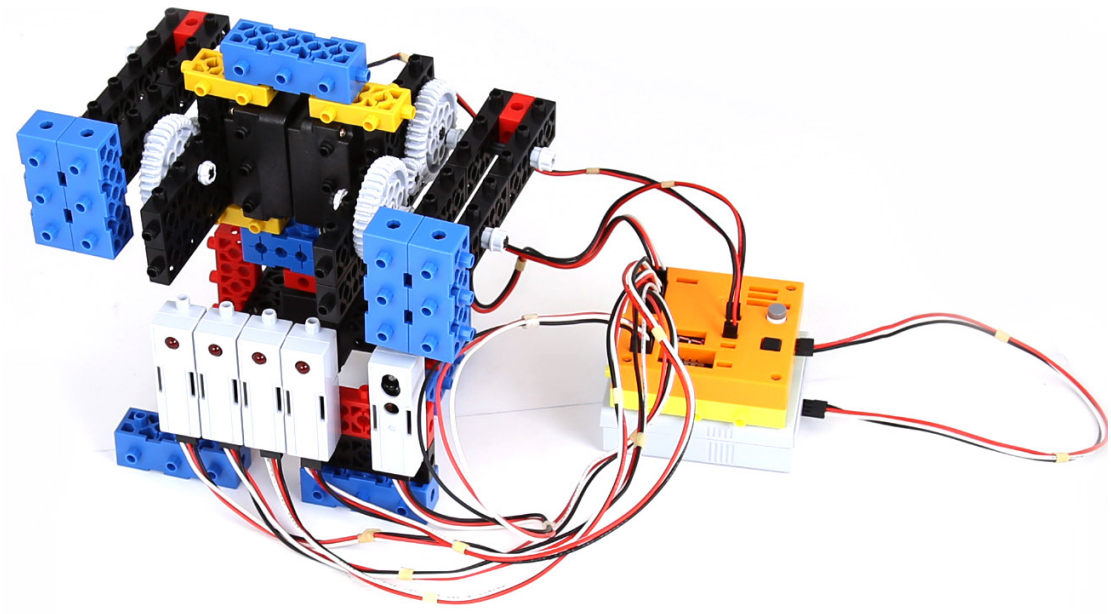


SCRATCH CODING KIT

Logic Boost

Dancing Robots

LESSON 9



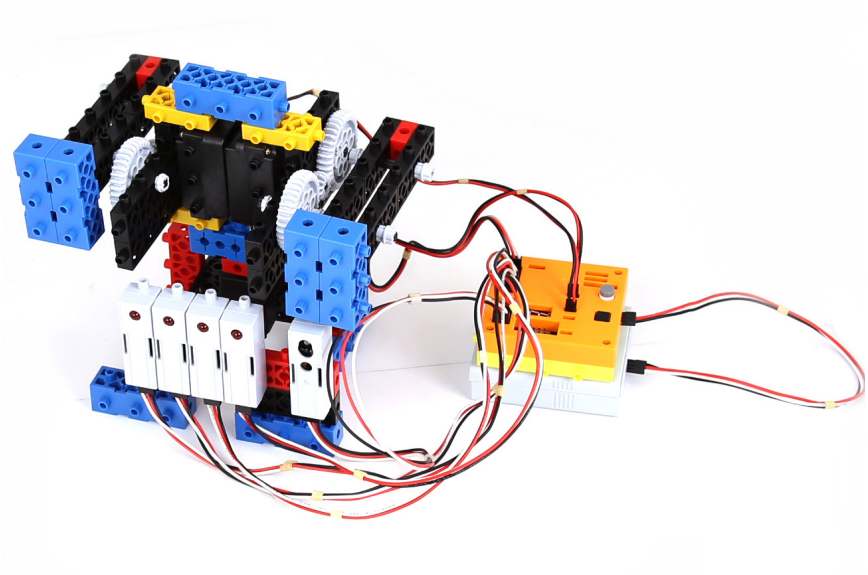
Making a Robot

Dancing Robots

Let's make a robot dance. You can use the scratch program to make a variety of expressions using infrared sensors and LED sensors.

There are 3 motors and 5 sensors in the dancing robot. We use two DC motors, one servo motor, one new infrared sensor and four LED sensors.
As you code your dancing robot, you will learn to 'modulate'. It is possible to classify various complicated actions to make the script simpler. When you cover the infrared sensor by hand, the LED sensor will flash in order, and the motor will run and let the robot dance. Just think about how you should code it.

Now we're going to make a dancing robot!

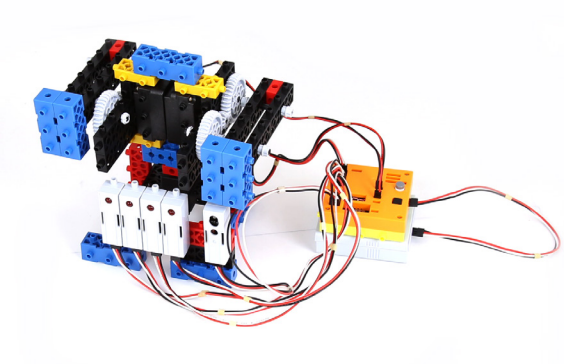


CLASS 3

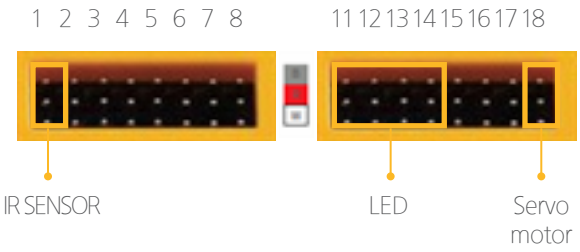
Making a Robot

Dancing Robots

Let's make a dancing robot.



Connecting Main Cell



Dancing robots Materials

							
Diamond H6 x 1	Diamond V8 x 3	Diamond V6 x 8	Rubi 8 x 4	Rubi 4 x 12	Rubi 2 x 2	Rubi 0 x 3	Rubi 7 x 1
							
Rubi 6 x 3	Mini 2 x 4	Mini 1 x 2	Servo x 1	Sawtooth 12 x 2	Sawtooth 36 x 4	Short connector x 4	Middle connector x 8
							
A23 x 6	A45 x 4	DC motor x 2	Servo motor x 1	IR SENSOR x 1	LED SENSOR x 4	connector x 6	

Making a Robot

Making a Robot

1

x 2

2

x 2 x 2

x 2 x 2 A23

x 2

3

x 2

x 1

x 1 A23

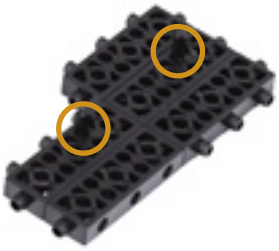
Four crossholes must be horizontal.

4

x 2 A45

x 2

5



x 2 x 1

x 2 x 2 A23

x 2

6

x 2

x 1

x 1 A23

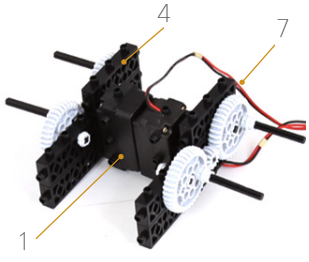
Four crossholes must be horizontal.

7

x 2 A45

x 2

8

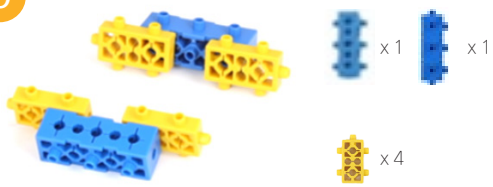


CLASS 3

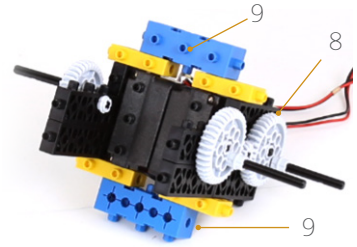
Making a Robot

Making a Robot

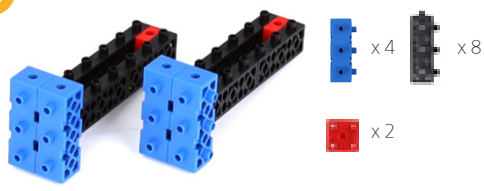
9



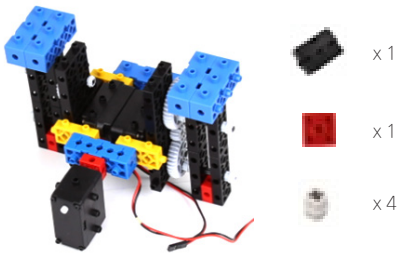
10



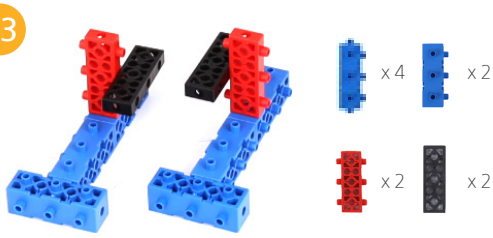
11



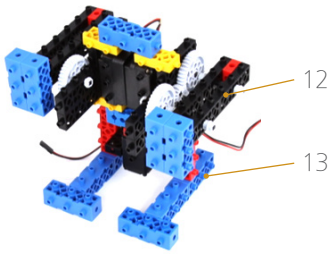
12



13

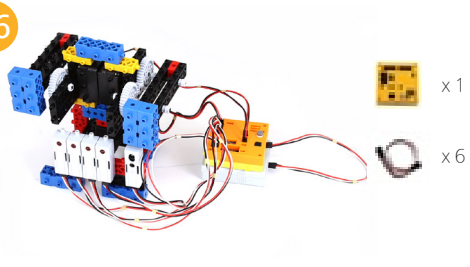


14



15

16



Making a Robot

Coding Dancing Robot s

Coding a dancing robot. The dancing robot will illuminate the LED sensor in sequence and move the DC motor and servomotor when the infrared sensor is turned on.

Infrared sensor Outputs the LED sensor through one input and is able to operate the motor. The program will work in the following order.

1.When you cover the infrared sensor by hand, the four LEDs turn on and off in sequence.

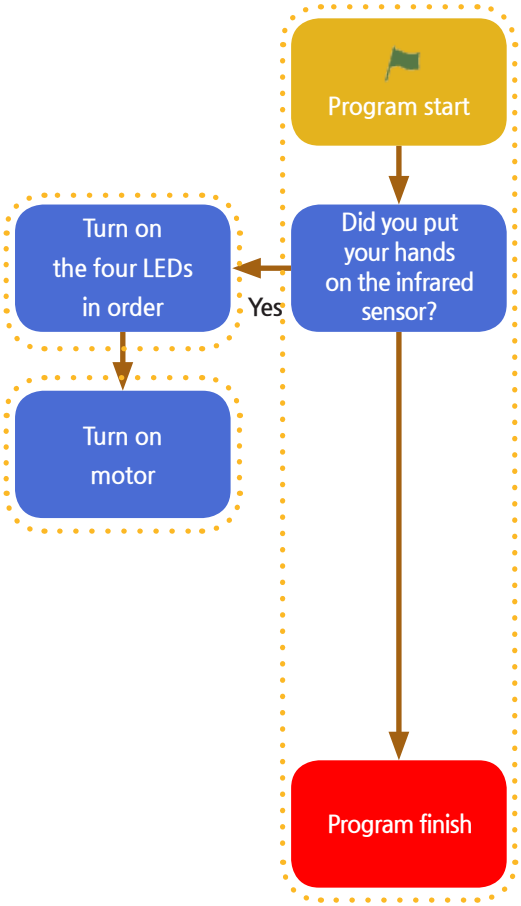
2. Once the four LED sensors are working, the servomotor and the DC motor work.

CLASS 3

Activity

Coding Dancing Robots

The dancing robot needs to create a total of three scripts. First, it consists of a script for starting and stopping the program and infrared sensor recognition, a script for turning on the four LED sensors in sequence, and a script for activating the motor. The program will check whether the infrared sensor is recognized or not and move accordingly. Check the flowchart below and think of the program as repeating. The dotted line will be the individual script.

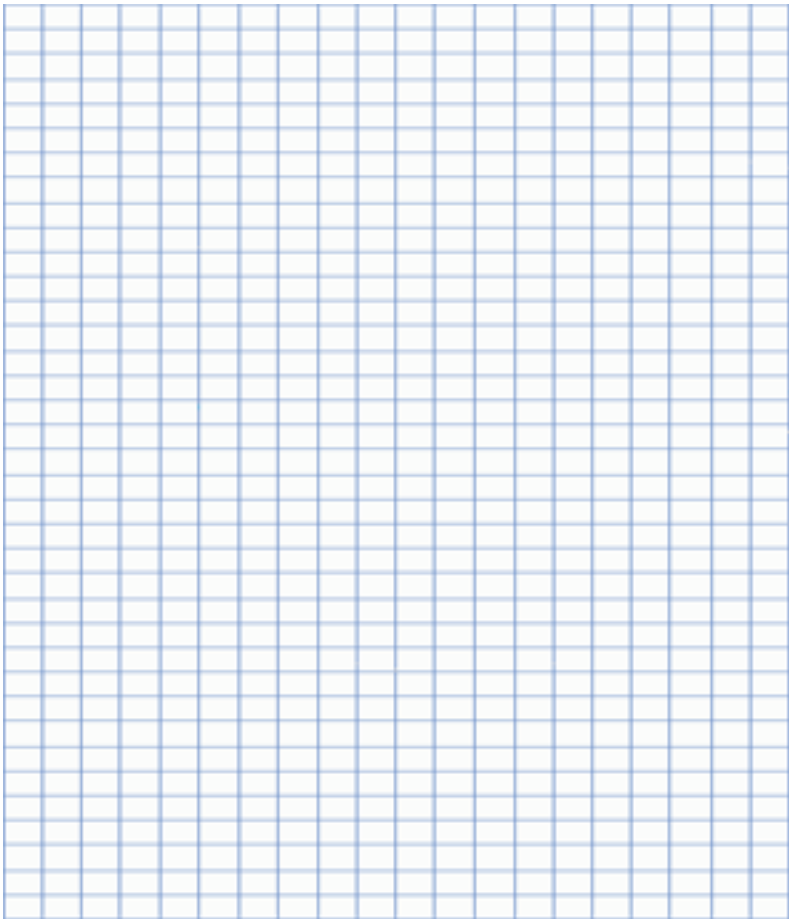


Activity

Coding Dancing Robots

Before, we decided to make three dancer robots scripts. But before that, try coding as you know it. Before we learn the concept of 'modularity' we can code a dancing robot that does the same thing in one script.

First, when the script is executed, set the angle of servo motor to 90. After that, when the infrared sensor is recognized, the four LED sensors turn on for 0.1 second each in turn, then the servo motor moves 180 degrees for 0.1 second, the 4th DC motor for 0.1 second and then the servo motor back to 90 degrees please. Code this by yourself.



CLASS 3

Commentary

Dancing Robots

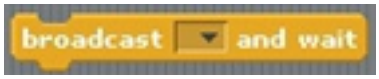
Did you create a script?
If it does not work, please refer to the script below. First, the script should start when you click on the flag icon and set the servo motor angle to 90 degrees. Put an 'infinite loop' block below it. Inside the 'Infinite Repeat' block, insert the block to turn on the LED sensor in 0.1 second increments. After turning on the LED sensors one after another, you have to operate the servomotor at 180 degrees for 0.1 second, for 4 seconds at .1 second, and then operate the servo motor at 90 degrees. Let's see the script below.

Activity

Modular coding for dancing robots

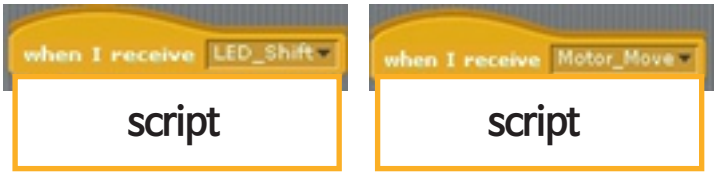
From now on, we will learn about modularization. You may feel that the script you learned earlier is somewhat complicated to code. Modularization of these scripts (in smaller units) makes it easier to understand when coding. First of all, let's divide the script into three and code it simply.

We need a 'make broadcast' block to split the script into three. In Scratch, 'broadcast' means to send a signal. If you do a broadcast, you have to get a broadcast. On the scratch 'control' tab, drag the 'Broadcast and Wait' blueprints.



Click on the black arrow in the 'Broadcast and Wait' block to bring up the window shown above. In the 'Name of message' box, type the name of the script that will receive the signal. Make 'LED_Shift' that will handle the operation of the LED sensor and 'Motor_Move', which will be responsible for the operation of the motor.

Did you make these two blocks? Now, on the 'Controls' tab, try pulling the 'when you receive' block.



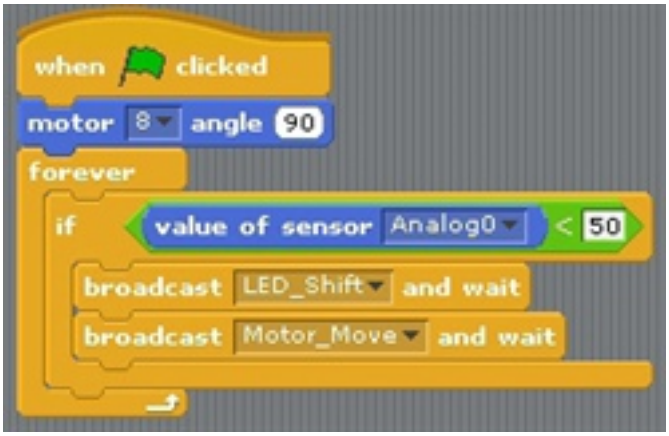
When 'LED_Shift' or 'Motor_Move' is broadcast, you have to receive the broadcast. The above picture is called 'When receiving LED_Shift' and 'When receiving Motor_Move'. And below that is the script that will be run after that. In other words, depending on which broadcast you have received, subsequent scripts will be executed. So we can create multiple scripts.

CLASS 3

Activity

Modular coding for dancing robots

Let's divide the entire script we created earlier into three scripts. First, let's create a script that runs when the flag icon is clicked. You can substitute the 'broadcast and wait' block instead of the LED and motor. Please see below.



Do you understand this script?

Set the servo motor to 90 degrees at first. Then, in the 'Infinite Repeat' block, 'If the value of Analog 0 Sensor is <50', put the block as before.

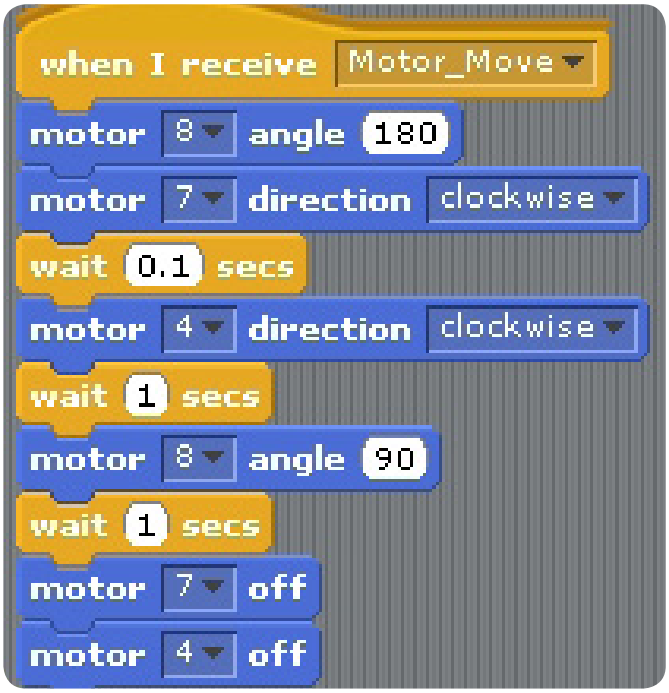
In this case, 'LED_Shift' and 'Motor_Move' scripts are executed if 'LED_Shift broadcast and wait' block and 'Motor_Move broadcast and wait' blocks are inserted into "If' syntax.

Now let's create the script 'LED_Shift' and 'Motor_Move'.

Activity

Modular coding for dancing robot

We broadcast 'LED_Shift' and 'Motor_Move' before. Now create a script for each broadcast. The script will start with the block 'when receiving LED_Shift' and 'when receiving Motor_Move'. Create the script below!



Does this script make sense? In the previous scripting with a single script, the length of the script was long and the distinction between each action was not clear. However, using the 'Broadcast' function makes it easier to create a script, and it makes good sense of the role of each script.

This division into smaller units is called 'modularization'. You can think of a large piece divided into pieces. This 'modularity' allows you to represent a single script that is difficult to code in multiple, split-level scripts and solve problems.

Do not forget to add to your coding skills, 'modular' learned through dancing robots!

SCRATCH CODING KIT

Logic Boost

Using a Wired Remote Control

LESSON 10

Making a Robot

Activity introduction

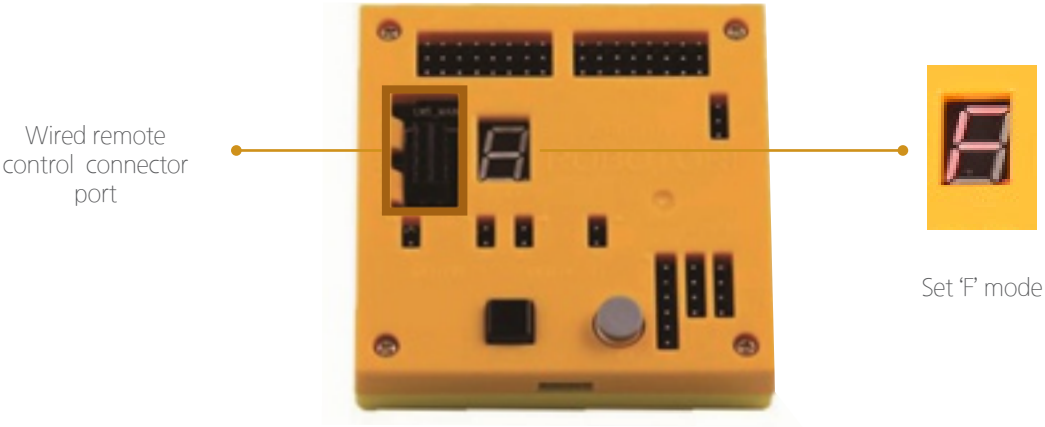
In this class, let's try activities using wired remote control. we will connect the remote control to the line follower we created earlier and move it.
First, let's create a model by referring to the line-follower assembly diagram we made earlier.

CLASS 3

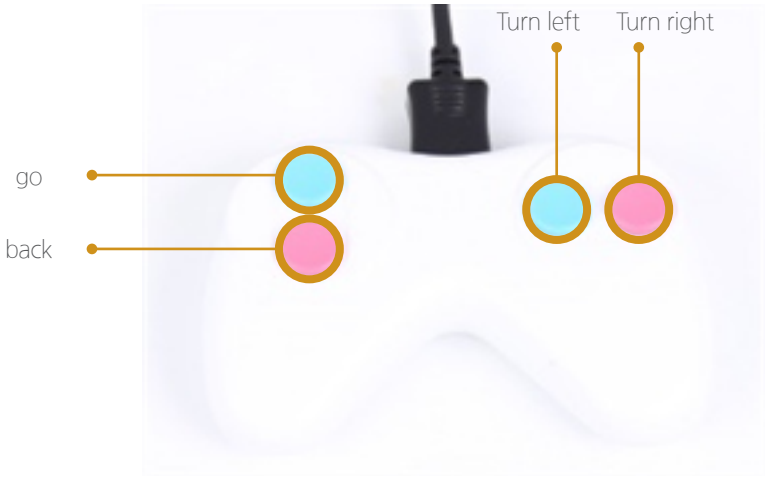
Wired Remote Control

Connecting a Wired Remote Control

Have you made all of your line followers? So, please refer to below and connect the wired remote to the main cell. The wired remote control can be plugged into the wired remote connector of the main cell. At this time, set the mode to 'F'.

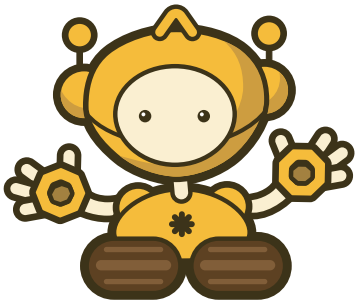


Each button on the wired remote control can be moved by using the following functions.



Once you're connected, move your line follower around!

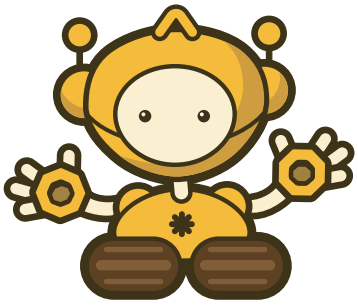
MEMO



CLASS 3

MEMO

MEMO



FCC Information to User

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Caution

Modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Compliance Information : This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment.