MAX:BOT
STARTER GUIDE



DFROBOT
DRIVE THE FUTURE

Contents

Contents **2**
Max:bot **6**
An introduction to Max:bot 6
How to define a robot? 6
Component list of Max:bot 7
An brief introduction to Micro:bit **8**
What is Micro:bit? 8
How to program Micro:bit 10
Makecode graphical programmable 10
Micropython code programming 11
How to graphical program Micro:bit through Makecode? 11
Open makecode 11
The programming interface 12
Download the program and upload to micro:bit 14
Chapter 1: Max:bot Run **16**
Goals: 16
Digital modules 16
1.1 Go forward 17
Key information 17
Program 17
Exercises 19
1.2 Go in circles 19
Key information: 19
Program 20
Exercise 23
1.3 Round in squares 23
Key information 23
Program 24
Exercise 27
Chapter 2: Smart Max:bot **27**
Goals: 28

Digital Modules 28
2.1 The "shy" Max:bot 28
Key information 29
Program 30
Exercise 34
2.2 Escape the maze 35
Key information 35
Program 37
Exercise 39
Chapter 3: Photoreceptive eyes 39
Goals 39
Digital modules 40
3.1 Adventure to brightness 40
Key information 41
Program 41
Exercise 42
3.2 The moth robot 43
Key information 43
Program 44
Exercise 45
Chapter 4 Ultrasonic, the secret weapon 46
Goals: 46
Digital modules 47
4.1 Ultrasonic measures distance 47
Key information 49
Program 49
Exercise 52
4.2 Car safeguard 53
Key information 53
Program 54
Exercise 57
Chapter 5 Max:bot Go! 58
Goals: 58

Digital modules 58
5.1 Max:bot wake up! 59
Key information 61
Program 61
Exercise 64
5.2 Trace the track 65
Key information 65
Program 65
Exercise 68
Chapter 6: Radio communication 69
Goals: 69
Digital modules 70
6.1 Say Hi 70
Key information 72
Program 73
Exercise 76
6.2 Motion sensing racing 76
Key information 79
Program 79
Exercise 84
Chapter 7: The sound and light show 85
Goals: 86
Digital modules 86
7.1 The rainbow light strip 86
Key information 87
Program 87
Exercise 90
7.2 Put the music on! 90
Key information 91
Program 91
Exercise 93
7.3 The sound and light show! 93
Key information 94

Program 95
Exercise 101

Max:bot

An introduction to Max:bot

Max:bot, an easy-to-use mobile platform with the best designed interfaces, can provide various functions for its users. It can be perfectly functioned when connected to Micro:bit. Besides, Max:bot carries nearly all fundamental capabilities of a robot. However, what may surprise you more is the all-metal integrated shiny enclosure which is easy to assemble.

- The best match of Micro:bit
- Set up quickly and easy to use
- Aluminium design, more solid and more durable
- Available to upgrade and connect to expansion boards. Support more than 100 Gravity digital modules.

How to define a robot?

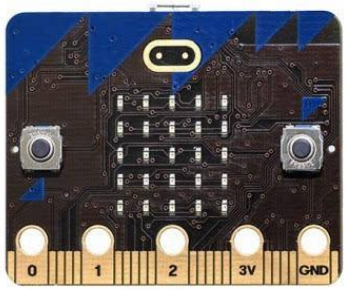
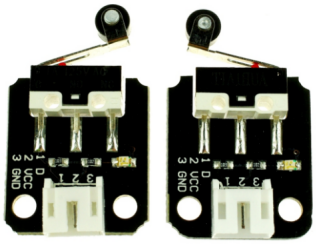
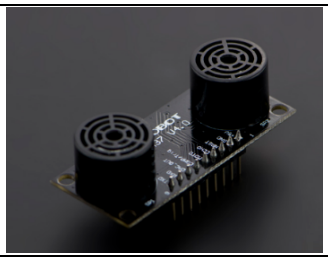
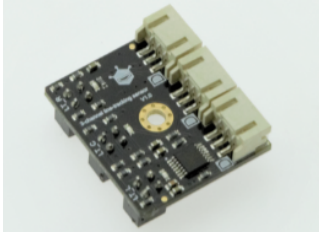
Generally speaking, robots are automated machines with a certain amount of computing power. Just like living creatures, robots too have their own "organs":

The brain: Micro:bit is the brain. With its own logical system, the board can process the received information, and send out commands.

The sensory system: A large number of sensors can be connected to Max:bot, thus we get the sensory system. For instance, a Smart Grayscale Sensor and a Digital Crash Sensor can be used as the eye and the hand to receive outside information.

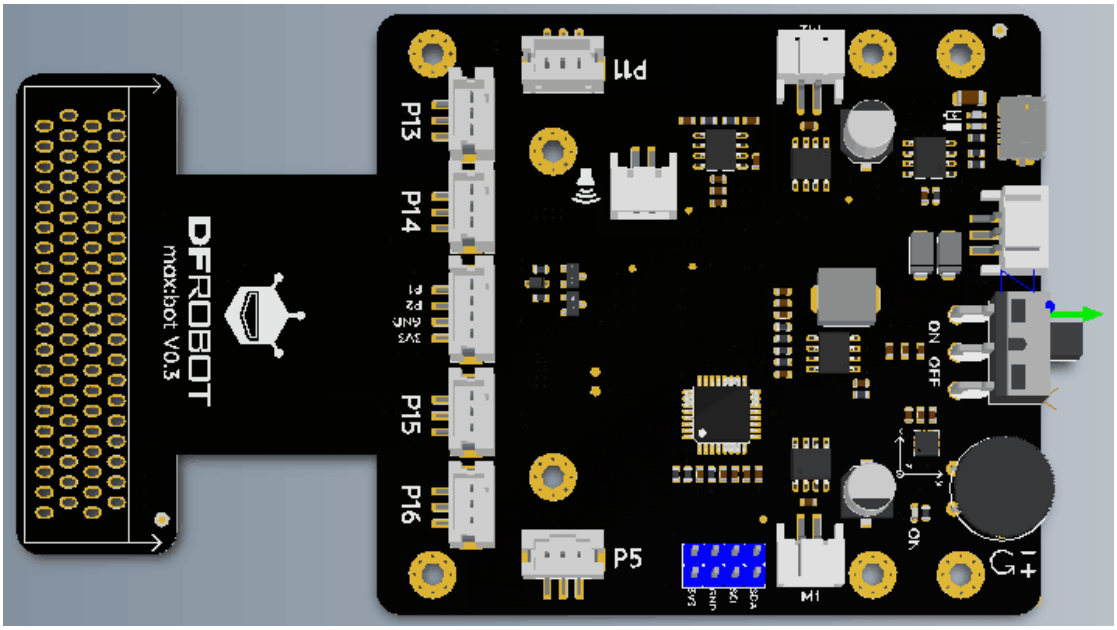
The executive system: This system can change our environment or send out messages to the outside world through gestures, languages, expressions, etc. LED strips and digital speaker modules can be connected to Max:bot for sending out massages. Additionally, there are another 25 red leds mounted in the Micro:bit which can be used to display messages and patterns.

Component list of Max:bot

Name(s)	Image(s)	Pin(s) to be connected
Micro:bit		
Digital Crash sensor(Left) Digital Crash sensor(Right)		P13、P16
URM37 V4.0 Ultrasonic Sensor For Arduino/ Raspberry Pi		P1、P2
Digital Line Tracking(Following) Sensor		P14、P15
Digital RGB LED strip		P5、P11

Motor		P8、P12

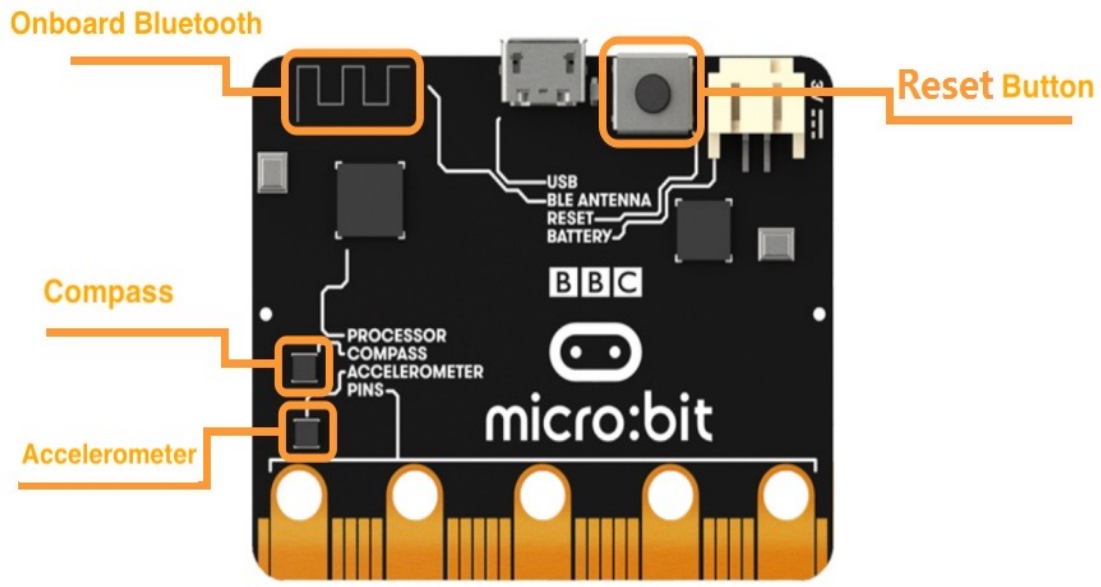
Connect all the components listed above to the board below, we can then have the body of Max:bot.

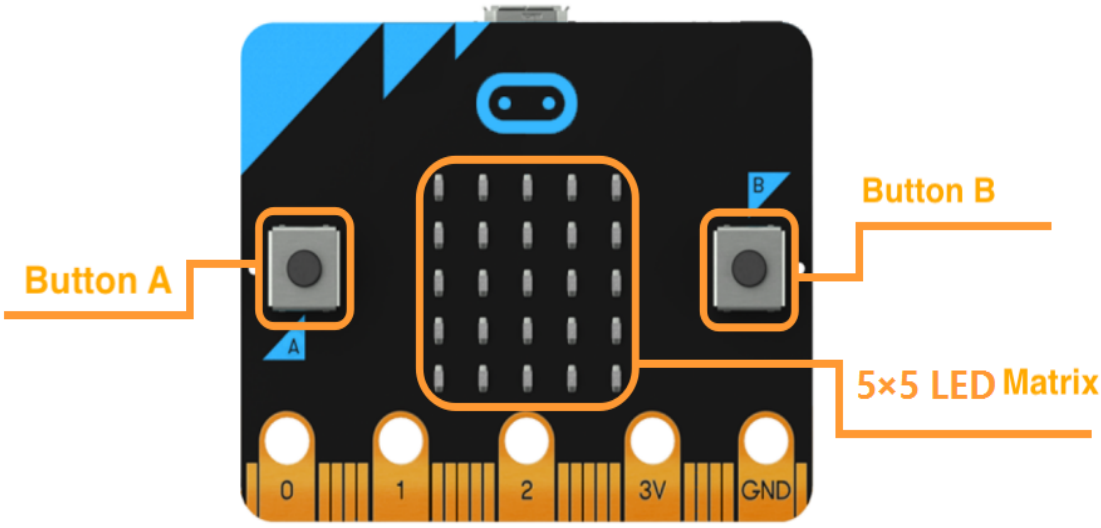


An brief introduction to Micro:bit

What is Micro:bit?

Micro:bit is an easy-to-use, powerful and cost effective pocket-sized microcontroller designed for kids and beginners learning how to program, letting them easily bring ideas into DIY digital games, interactive projects and robotics.





Thanks to its extensive I/O ports and hardware support, Micro:bit is well suited for various robot-related learning and development.

- A small board similar to the size of a credit card (4cm x 5cm)
- On-board modules, like: accelerometer, Compass and Bluetooth® Smart module
- A pocket-sized microcontroller
- A 5x5 LED matrix (also supports light detection)

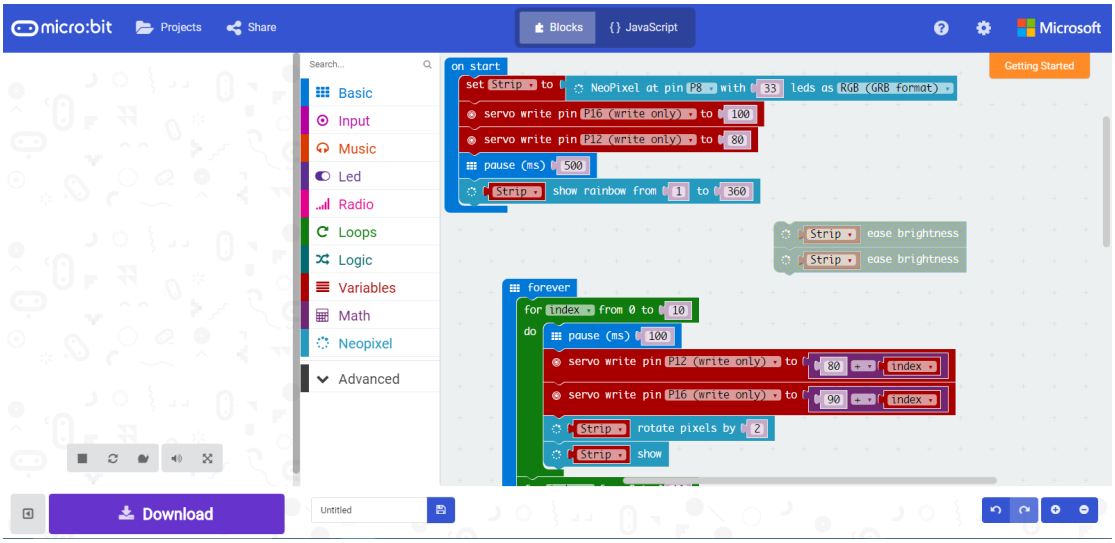
- Light and temperature sensors and other common sensing devices

Equipped with ARM's M0 processor, Micro:bit can execute most of the fundamental functions of a robot.

How to program Micro:bit

Makecode graphical programmable

For beginners, the merits of Micro:bit lies in its rich programming languages and inclusive interfaces.



Take Microsoft's makecode online programming platform as an example.

Users can drag and place the relevant modules in the webpage to read data of sensors, and to process the data quickly through certain logic functions including loop, judgement, etc. The development of micro:bit reduced the cost of programming greatly so as to ensure the majority learners to focus on robot control.

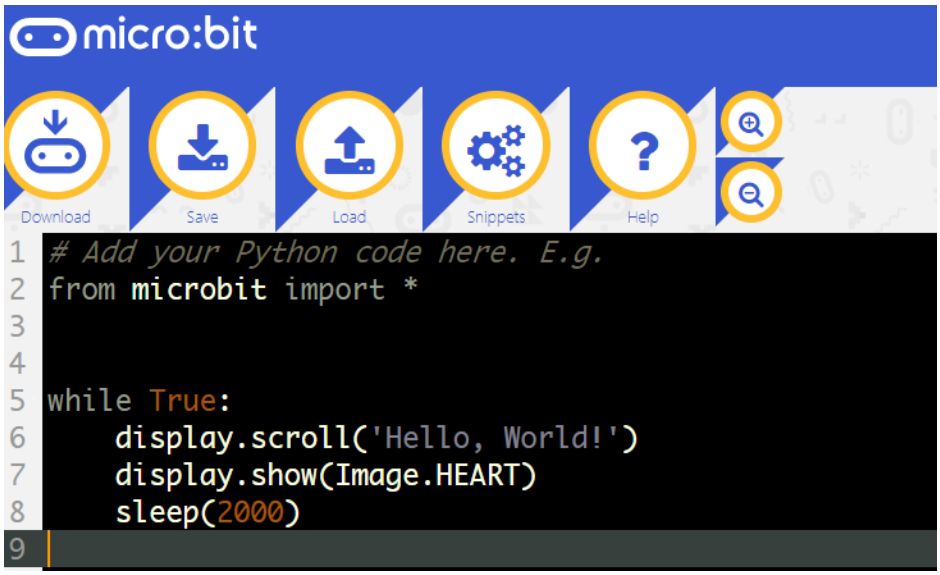
Click below links to start your online programming trip:

Http://microbit.dfrobot.com.cn/index.html or,

Https://makecode.microbit.org/#

Micropython code programming

Senior users can use Python to program Micro:bit, which is more powerful and more widely used. Micropython is developed based on Python language. It continues to use Python language but is more powerful in its functions.



In this tutorial, we will learn how to use the makecode graphical programming. The other starter guide about Python is to be continued.

How to graphical program Micro:bit through Makecode?

Open makecode

Open makecode from the following links:

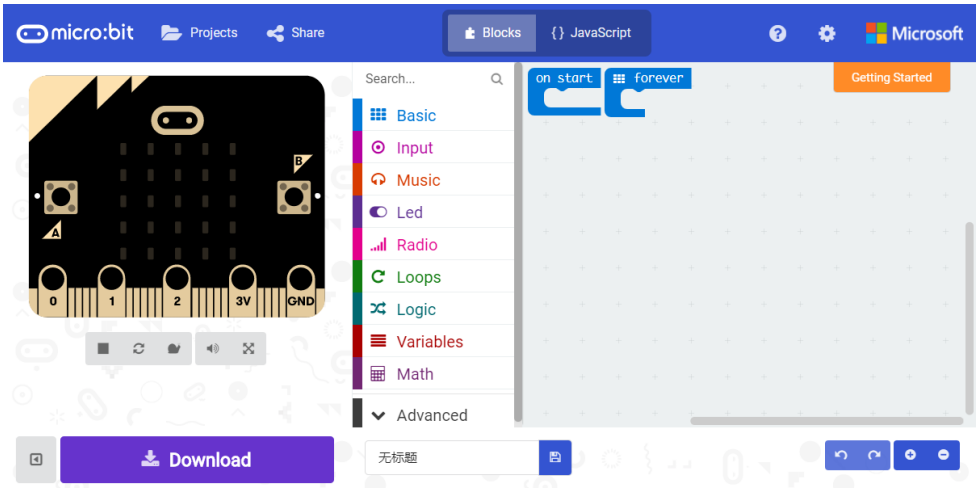
Please note: Keep your computer always connecting to the Internet.

Http://microbit.dfrobot.com.cn/index.html or,

Https://makecode.microbit.org/#

When using it for the first time, no program can be found on the interface.

The initial language is English.



Makecode has the memory function. So whenever you open it, you will find it maintains the last programed interface.

The programming interface

According to its functions, Makecode can be divided as below five parts:



Simulation window: Simulates the operating status of micro: bit. During the process of programming, you can always check how your program looks like the through the window.

Function area: Where you can find all the function blocks, including input, output, loop, logic, etc.

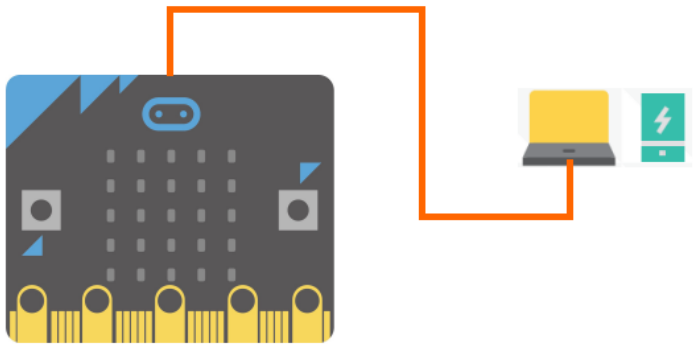
Programming area: Dragged the blocks from “Function area”, stack them up and build your program here.

Settings: You can select language here Language .

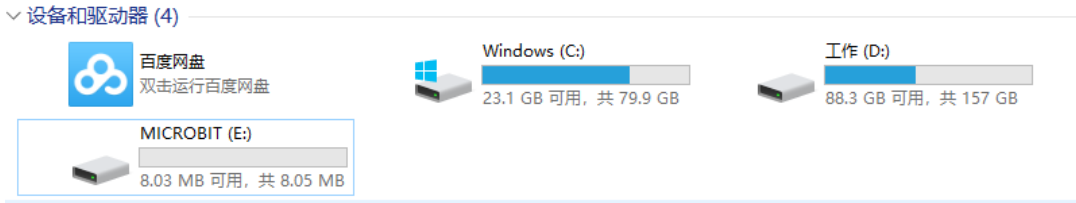
Naming area: You can name your project here. The default is “Untitled”.

Download the program and upload to micro:bit

1. Connect micro:bit to computer via the USB cable.



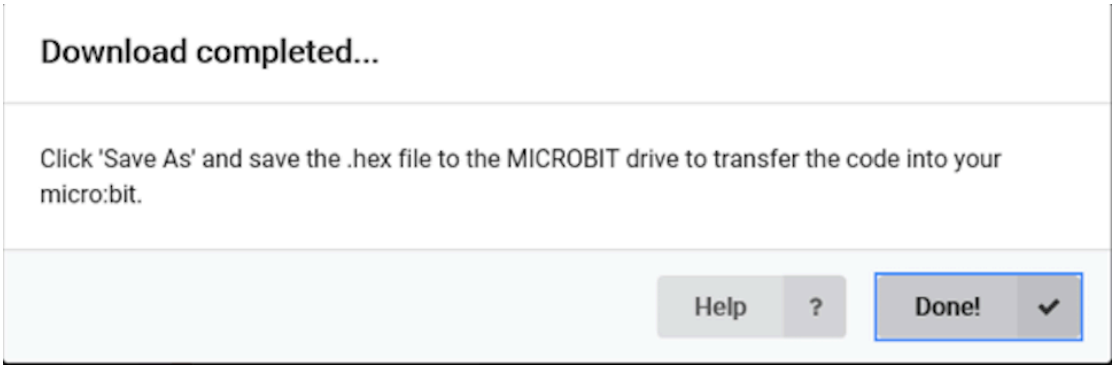
2. Before programming, we should make sure that the mainboard is recognized by the computer. When micro: bit is connected, a "MICROBIT" directory will show up in “My Computer”.



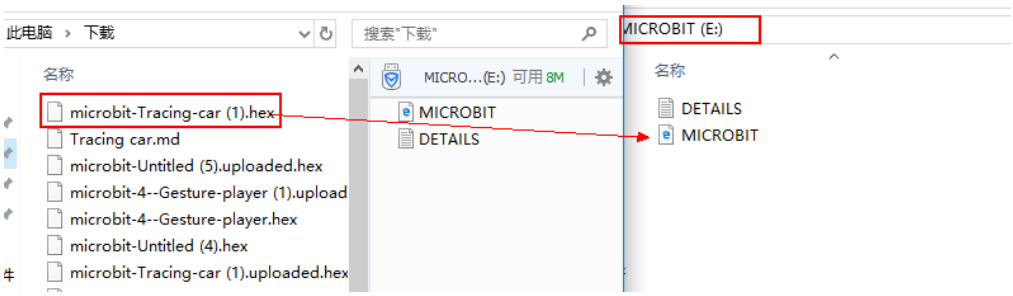
3. Click “download” and save the programed “.hex” file.



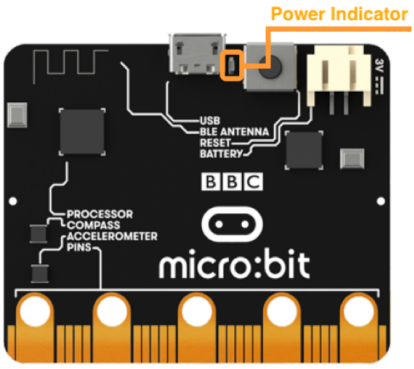
Then you will see the pop-up dialog box below:



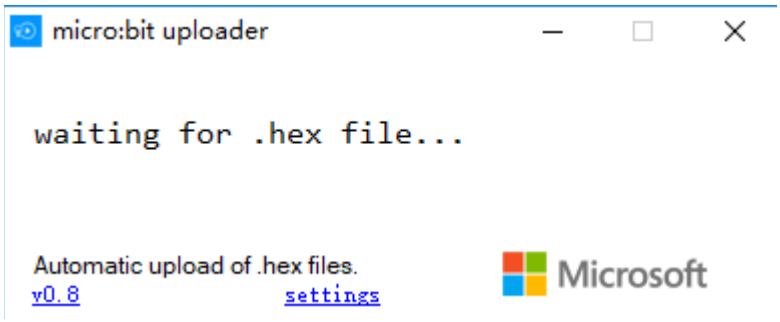
4. Find the saved “.hex” file and move it into “MICROBIT”.



During the process of downloading, the power indicator on the back of micro: bit will blink. When completed, it will stop flashing and keep on going.

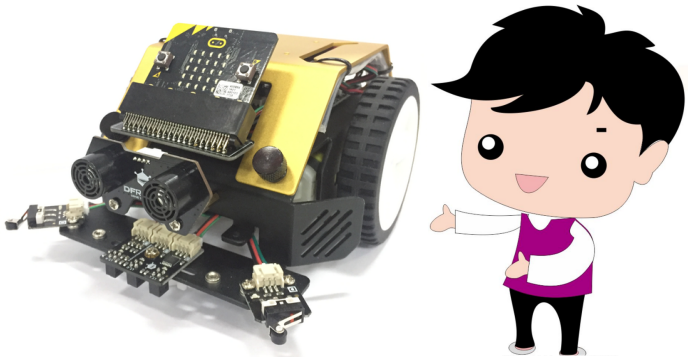


Tips: During programming, you can operate the Microbit.Uploader which can be used to upload the ".hex" file automatically to "MACROBIT".



Chapter 1: Max:bot Run

Hi there, he is Max:bot, your personal robot companion. I believe you have already been obsessed with his shiny skin. You must want to play with him now. But he is a newborn that even cannot make one simple move. God knows how much he has dreamt to jump and dance like you do. He really need your help.

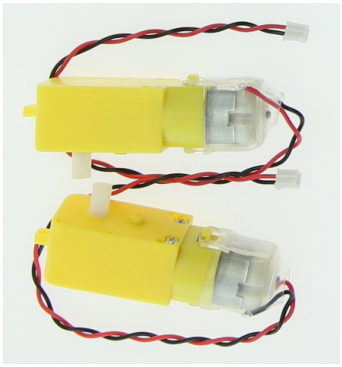


Goals:

1. How to run the wheels?
2. What is the differential steering principle?
3. How to build a sub-loop?

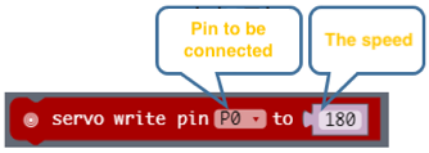
Digital modules

Digital modules	Images	Functions

Motor		The motors allow Max:bot to go forward and rotate.

1.1 Go forward

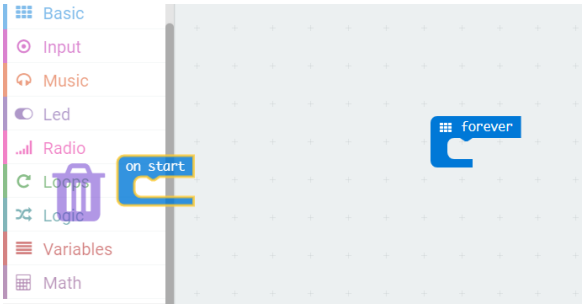
Key information

Function block(s)	Image(s)	Function(s)
"servo write"		The "servo write" function is used to control the speed and directions of the motors. 
"forever"		All the functions to be executed repetedly should be placed into it. The "forever" loop is the major loop.

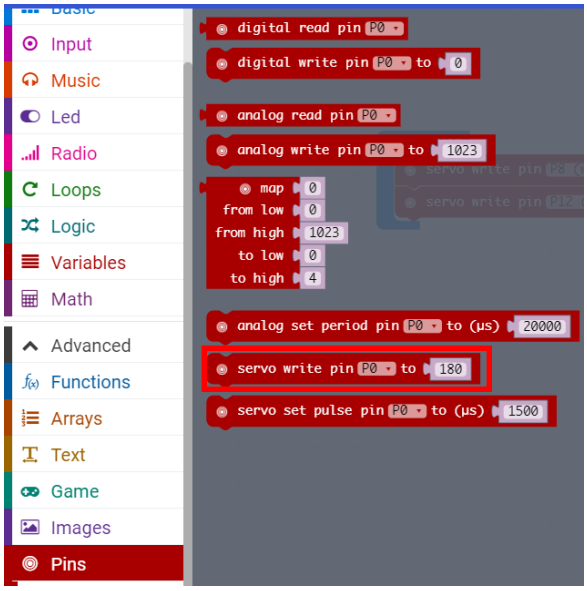
Program

（1） Open makecode: http://microbit.dfrobot.com.cn/index.html

（2） The "on start" module will not be used in this part. So we have to delete it. We just need to drag and drop it to the function area. It will then be deleted automatically. This method is also available for all other functions.

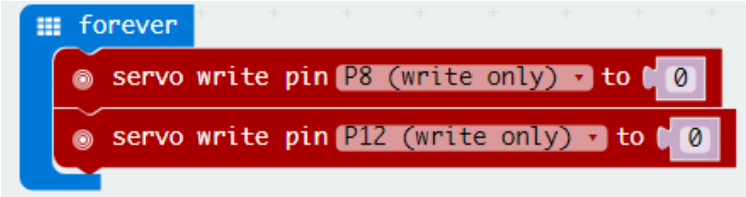


（3） Find the “servo write” function from “Pins” in the function area. This function controls the speed of motors.

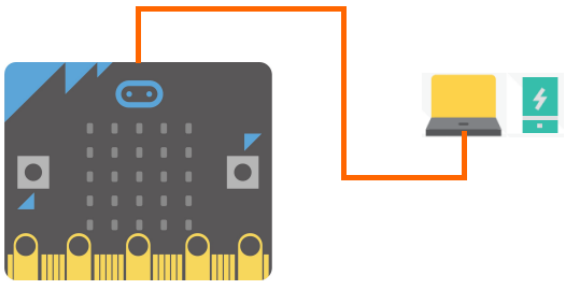


（4） Put the “servo write” function into the “forever” loop.

Please note: the left wheel should be connected to P8, and the right to P12.

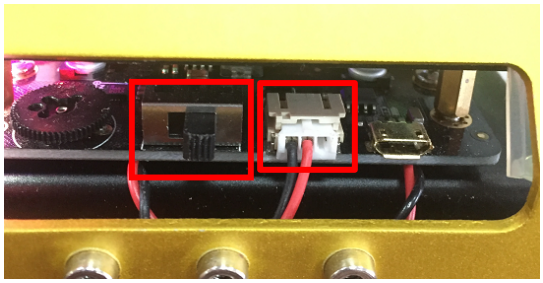


（5） Now please download it to Max:bot. Wow! Can you see that? Max:bot is going forward!



After downloading, you have to remove the USB cable from the computer. And the battery module should be connected to Max:bot.

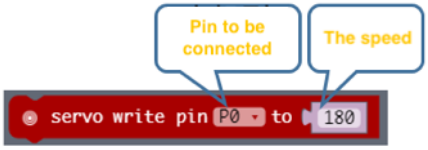
If Max:bot still not move, please check if you have switched it on or not.



Exercises

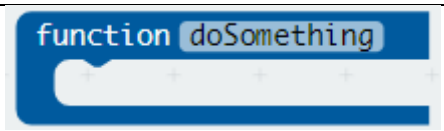
It is you that allow Max:bot to move forward. He really appreciates it. But he also wants to know how to go backward. Will you help him with this?

Tips: the number in the "servo write" function controls the speed.



1.2 Go in circles

Key information:

Function block(s)	Image(s)	Function(s)
"function"		Should continuous actions be executed repeatdly, we can use the "function" to replace the series of actions. Thus the major loop can

		be simplified. Please note: The "function" should not be placed into the "forever" loop. It is an independent sub-loop.
"call function"		If the actions programmed in the "founction" should be used in the "forever" loop. We will then find and place the "call function" into the "forever" loop to trigger the actions programmed in the "function". Please note: In the "forever" loop, the "call function" is a corresponding use of the "function".

Please note: the "function" and "call function" are all lie in "Advanced" => "Functions". But both of them are not listed in it directly. The way to find them goes like:

1. Way to find "function": "Advanced" => "Functions" => "Make a Function". Click "Make a Function", name it and click "OK". Then you will find this "function" block appears in the programming area automatically.
2. Way to find "call function": at the same time the "function" block appears in the programming area, the "call function" block appears in the "Functions" of function area.

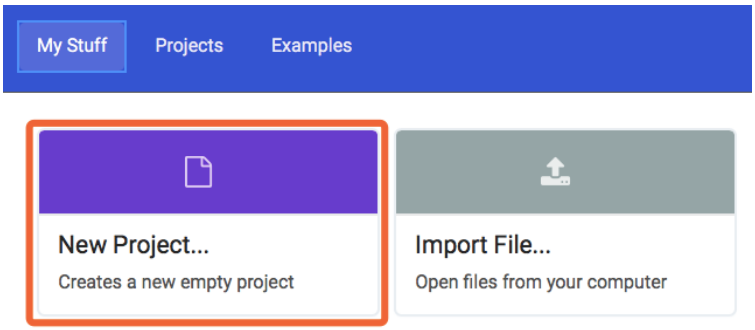
   Please remember that the "call function" appears only when the "function" is created.

Program

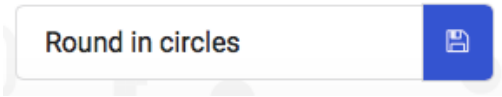
（1） Start a new project and name it as " Round in circles"

Click "Projects". 

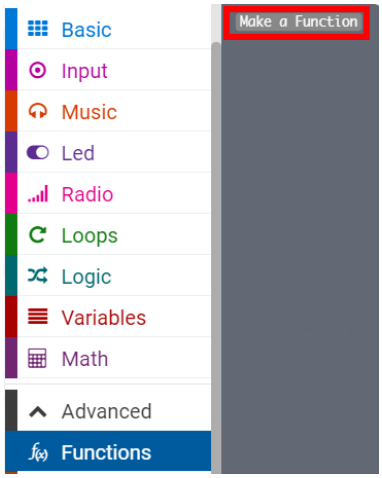
Please note: To avoid any confusion, we have to start a new project, otherwise makecode would save it directly into the last project file.



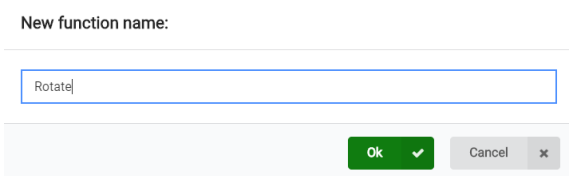
Name it: Round in circles



（2） Here we need to create a sub-loop. It goes as: “Advanced” => "Functions" => "Make a Function".



Click “Make a Function” and name it as: Rotate. Then you will find the “function” block appears in the programming area automatically. And at the same time it is created, the “call function” appears in the “Functions” of the function area.



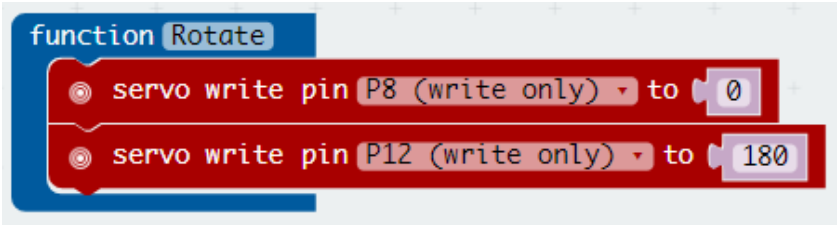
（3） How to make Maxbot turn?

Before programming, we will need to first get familiar with the “differential steering” principle.

In our daily life, what enables a car to turn is the different rotate speed of the driving wheels. Both the clutches that are mounted on the left and right axle shafts and the brake can be used to control the driving wheels to rotate in different speed. At present, nearly all tracked vehicles, like tanks, are abide by this principle.

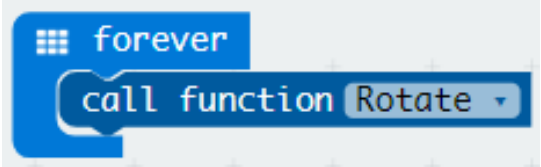
Therefore, the key to success is to rotate the two wheels in different speed.

Place the “servo write” functions into the “function” sub-loop.

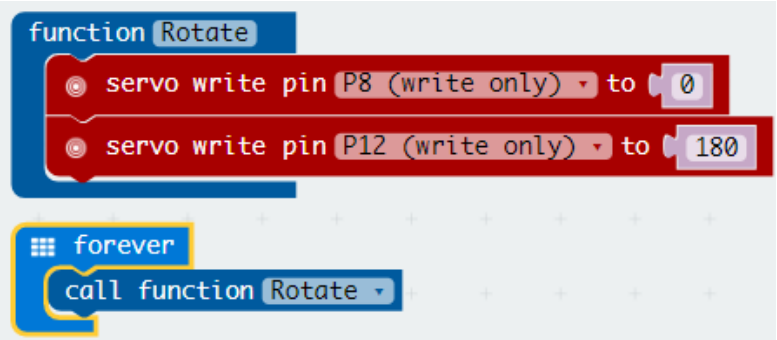


The “180” is the top clockwise rotation speed, whereas the “0” is the top anticlockwise rotation speed.

Above we have created the sub-loop named “Rotate”. We need now to place the “call function” from “Functions” into the “forever” loop. Thus the actions in the “function” will be executed by the “forever” loop.



（4） Combine all function blocks listed above together, we will have the final program as below:



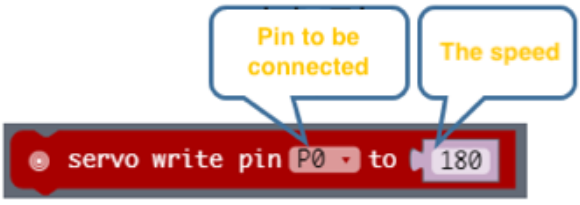
Download the program to Max:bot, it will then rotate in circles clockwise.

Do not forget to remove the USB cable and switch Max:bot on.

Exercise

We have learnt how to make Max:bot rotate clockwise. Can you make it rotate anticlockwise with the guidance of the differential steering principle?

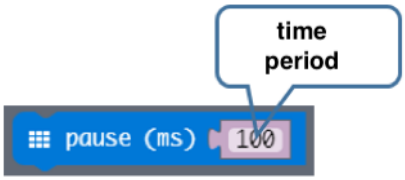
Tip: It is the value of “servo write” to control the speed.



1.3 Round in squares

Key information

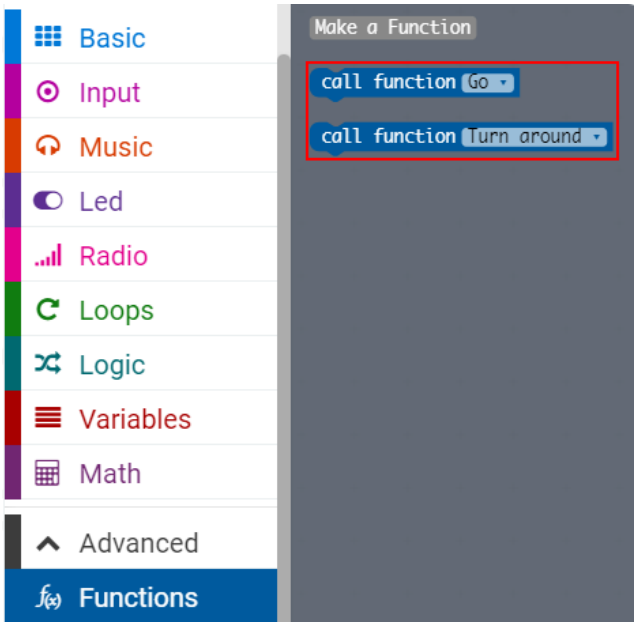
Function block(s)	Image(s)	Function(s)

"pause"		The "Pause" function maintains the same status lasting for a certain period of time.

Program

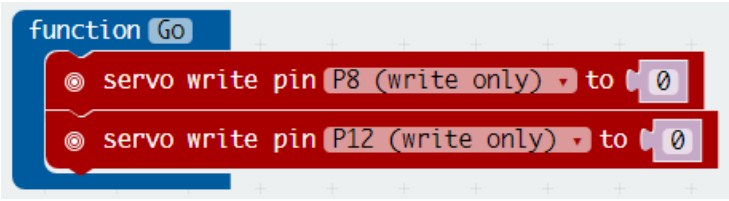
（1） Start a new project and name it “Round in squares”

（2） Create another two sub-loops and name them as “Go” and “Turn around”. Have you remembered how to create them? We will review it together: “Advanced”=>”Functions”=>”Make a Function”.

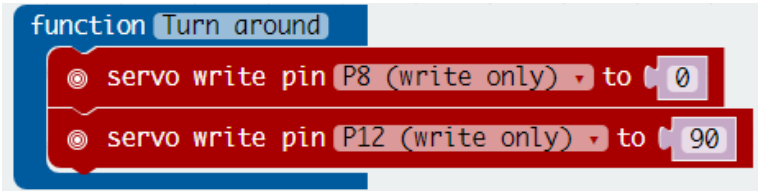


As we all know that if one would like to round in squares, he/she must be firstly walk straight for a while (like 1 minuet), then turn 90 degrees and keep walking forward for another minuet... These moves will be repeated until he/she finished a square.

（3） Therefore, we will need first to make it go straight. As have mentioned in previous parts that the “servo write” function can be used to control the speed of wheels and make Max:bot go forward. So we will use it in this part. Set both the values as “0” to keep the two wheels rotate in the same speed. And then put the “servo write” into “function(Go)”.



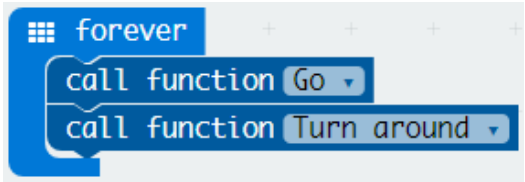
（4） We have also learnt the “differential steering” principle to make Max:bot rotate. But here, we have to adjust the values to make it turn 90 degrees only. Set P8(the left wheel) as “0”, and P12(the right wheel) as “90”. The value ”90” means the wheel is to be kept still.



Compared with the “function(Rotate)”, the values “0” and “90” set in the “function(Turn around)” means:

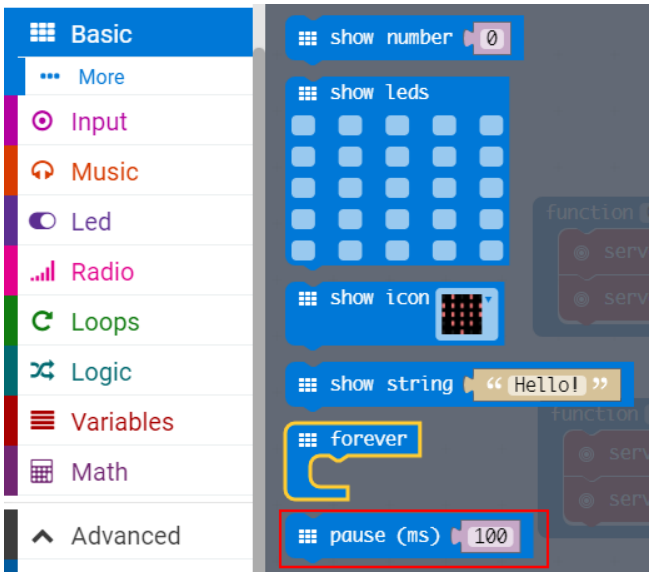
The “90” in P12 means the right wheel rotates 0 degree. The “0” in P8 means the left wheel rotates at the top anticlockwise speed. Thus, Max:bot turns right.

（5） Place both the “call function(Go)” and the “call function (Turn around)” into the “forever” loop. The final program goes as:

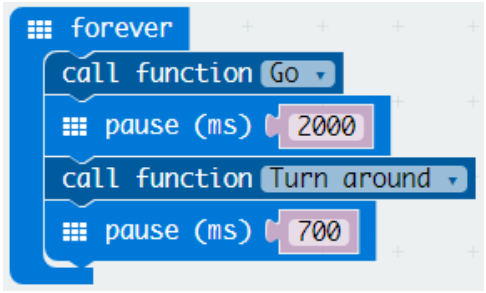


Does the Max:bot go in square now? I suppose no. We have to be more patient about it. No pain no gain.

To ensure Max:bot round in squares, we have to give him enough time to move forward and turn. We need the "pause" function here to set the time. You can find it in "Basic" of the function area. Place it into the "forever" loop.

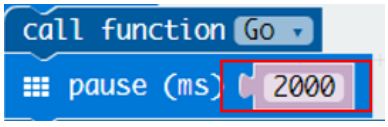


The program goes as below:

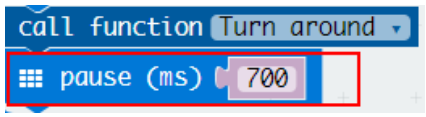


Please note:

a. The "2000" determines the side length of the square.



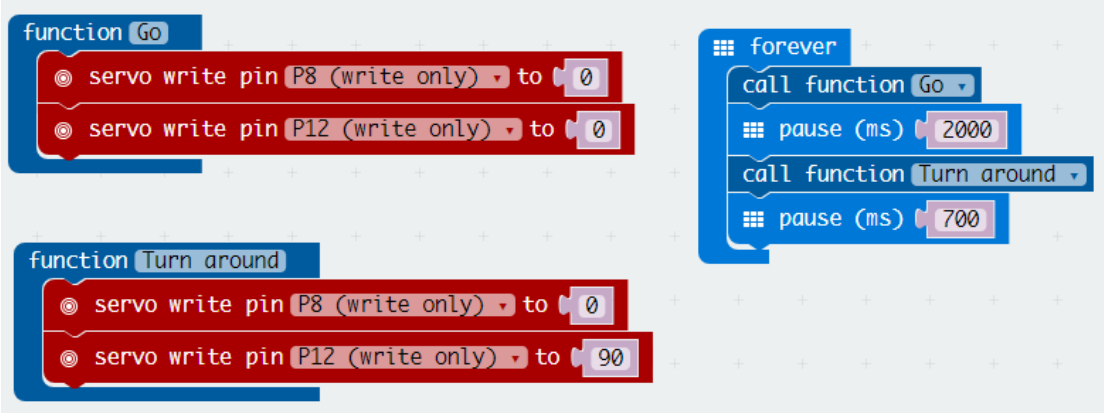
b. The "700" determines the rotation angle.



Different batteries output different voltages. So the "2000" and "700" here are just used for a reference. You have to try and adjust them based on

the actual situation. Remember what we have just said: No pain no gain. Just try it!

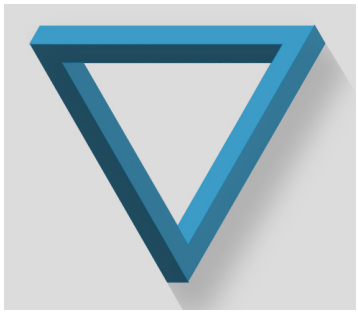
（6） Combine all the functions listed above together, the final program is as following. Make sure you have made every step correctly and then download it to Max:bit.



You did it! It now rounds in squares!

Exercise

Once you have made it round in squares, it is a piece of cake to round it in triangle. Try it!



Chapter 2: Smart Max:bot

You may now possess a sense of fulfilment because Max:bot can go to anywhere you want it to go. But have you feel a little bit more frustrated that

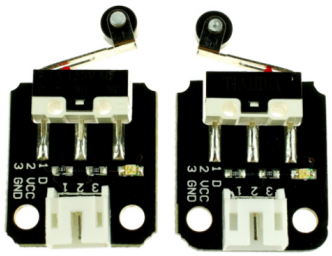
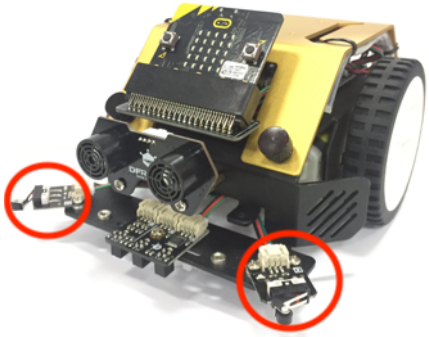
there always has something to block its way, like a desk or a closet. Don't worry, in this chapter, we will teach Max:bot how to avoid these obstacles.



Goals:

1. How to use the Digital Crash Sensor?
2. How to use the "if then" function?
3. Diagram aided programming

Digital Modules

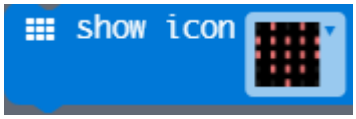
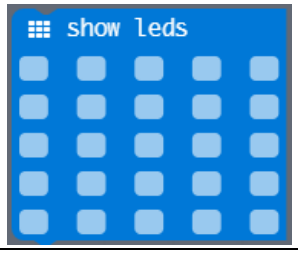
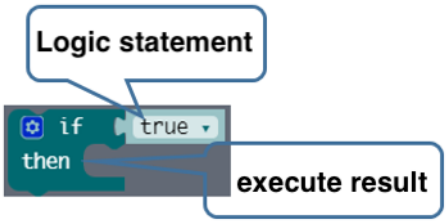
Digital Module(s)	Image(s)	Function(s)
The Digital crash sensor	 	The digital crash sensor can, like a button, detect obstacles. As the second image to the left shows that there are two digital crash sensors (circled in red) connected to Max:bot separately.

2.1 The "shy" Max:bot

Sometimes, when being touched, Max:bot will become cute and shy. Yes, he can feel you.



Key information

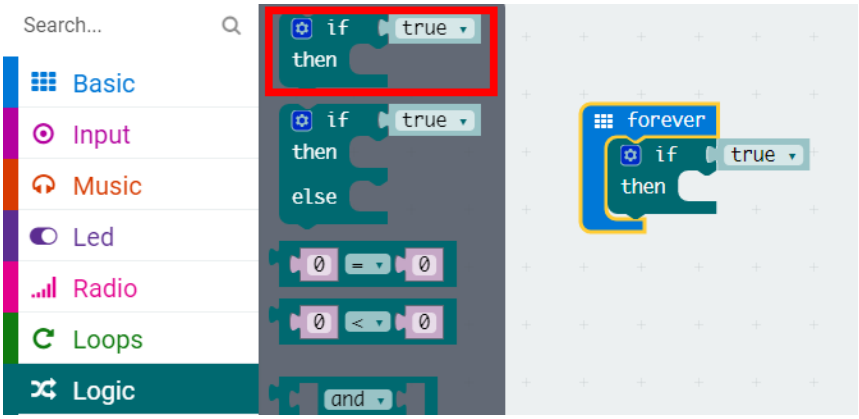
Function block(s)	Image(s)	Function(s)
"show icon"		There are many icons can be selected in the "show icon" function. You can use it to choose your favorite icons.
"show leds"		There are 25 light blue dots inside the block, you can click each of them to edit a pattern. These dots represent the 25 leds in Micro:bit.
"if then"		The "if then" is used to judge which function will be executed. Which means that if the logic statement to the right side of "if" is met, the code sentence to the right of "then" should be executed, whereas the next function should be executed if the logic statement is not met.
"="		The "=" opertator is used to control the range of a value. For example: As we can see from the image to

		the left that there are two light red dots. If we place "digital read(P1)" into the left dot and change the value of the right dot to "1", it will then means that the value of P1 equals to 1. Inside the "=" operator, there are many other alternatives, like ">", "<", etc.
"digital read"		"Digital read" is used to read the status of the Pin. There are two status: a high voltage (presented as 1), and a low voltage (presented as 0).

Program

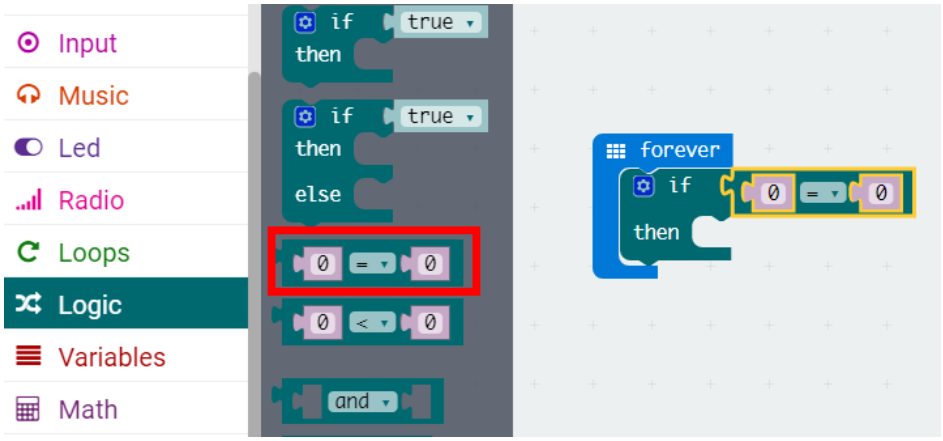
（1） Start a new project and name it as "the shy Max:bot"

（2） Find the "if then" function from "Logic" in the function area and place it into the "forever" loop.

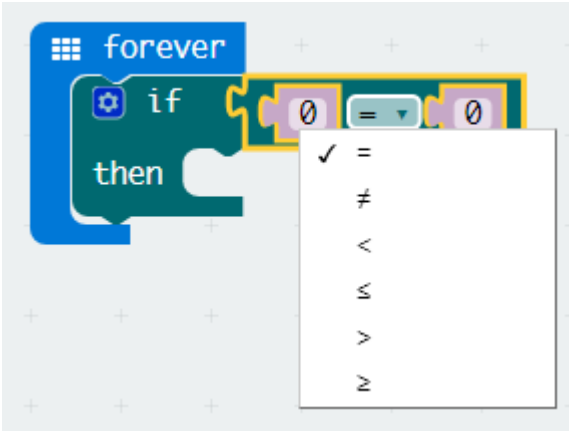


If the logic statement to the right side of "if" is met, the function to the right of "then" should be executed, whereas the next function should be executed if the logic statement is not met. This is an either or thing.

（3） The "=" operator lies in the "Logic".



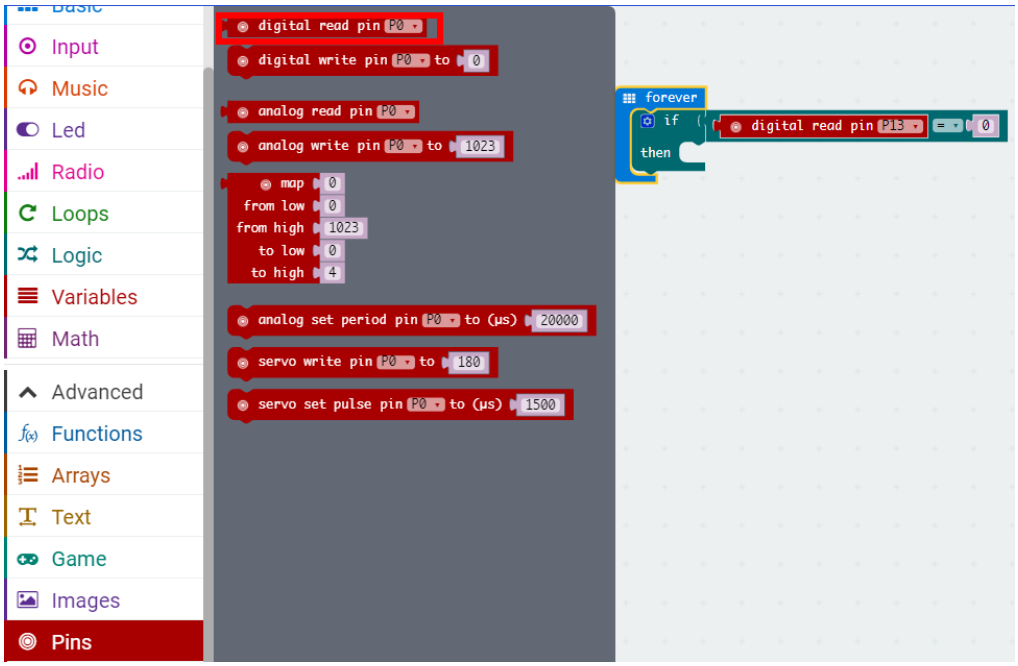
Besides "=", there are many other operators, like ">", "<", "≠", "≤", "≥", can be selected in the drop-down menu.



（4） How to control the digital crash sensor?

Anytime when Max:bot is touched by others, he will immediately become a shy robot. Why he becomes so sensitive? Because we have connected digital crash sensors to it.

In this part, the "digital read" function will be applied to determine whether the digital crash sensors are touched or not. Connect one of the digital crash sensors to Pin13.

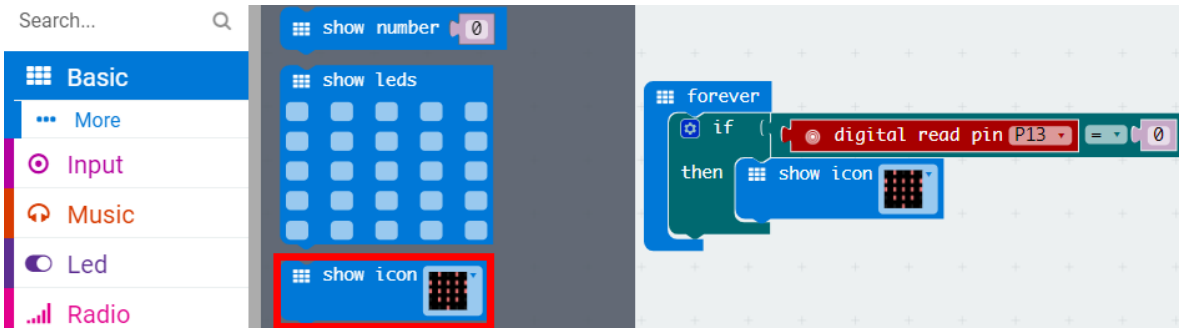


Do you have any idea about what response Max:bot is going to make? The table below can be used as a reference.

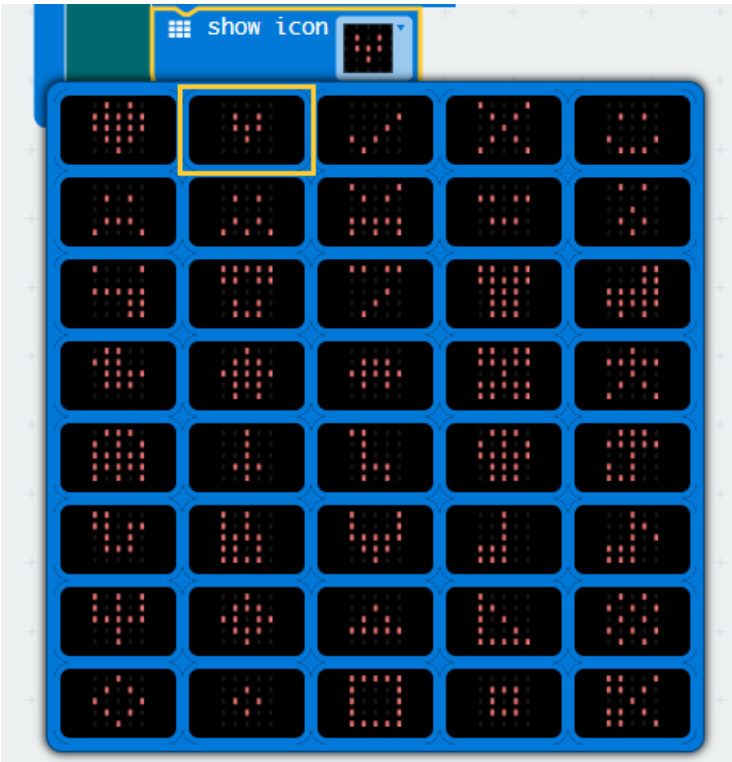
Status	Values
Being touched (the indicator turns on)	0
Untouched (the indicator turns off)	1

（5） Make the heart beat

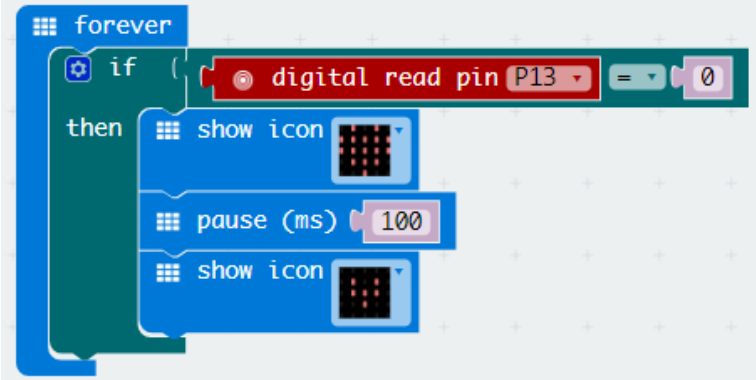
Once has being touched by others, Max:bot will immediately become shy and with his heart speeds up. But how can we make his heart really beat? The function “show icon” does the job. You can find it in “Basic” of the function area.



As has shown in below image that there is a micro triangle lies at the top right of the “show icon”. Click it and choose the smaller heart icon.

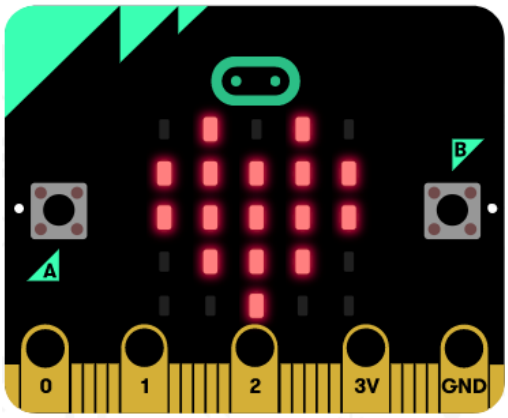


To make it more like a real heartbeat, we need to add the “pause” function and another bigger heart icon. By putting together all the functions listed above, we will have the following program.



（6） Download the program to Max:bot.

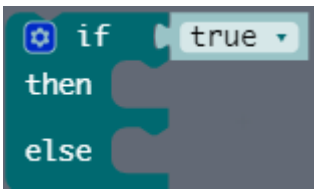
Now when you touch Max:bot, he will then show you his heartbeat. What a cute robot!



Exercise

Needless to say, Max:bot is a very nice companion. We have already created his heartbeat. Now we want to make him a smile face when he is untouched.

We will use the “if else” function to make him smile. You can also find it in “Logic” of the function area.



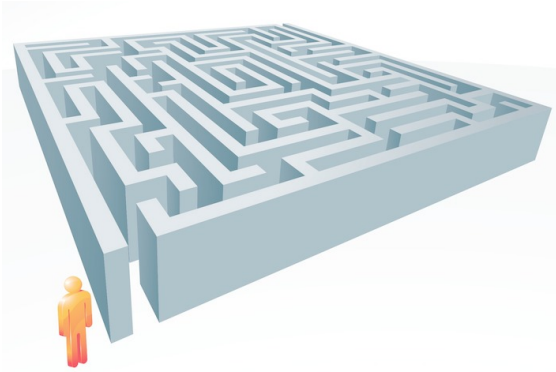
Compared with the “if then”, there is an additional “else” in the “if else”. What is it?

As we have learnt in previous part that, in the “if then” function, If the logic statement to the right side of “if” is met, the function to the right of "then" should be executed, whereas the next function should be executed if the logic statement is not met.

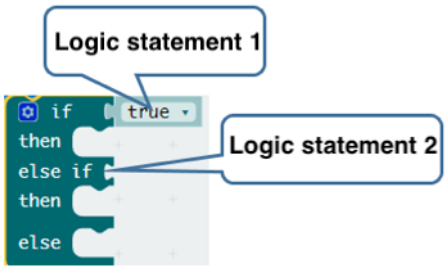
While in the “if else” function, if the logic statement on the right side of “if” is not met, the statement to the right of “else” should be executed.

2.2 escape the maze

Since the digital crash sensor grows Max:bot to a sensory robot, what if we place him in a maze? Will he himself find a way out?



Key information

Function block(s)	Image(s)	Function(s)
Multiple “if else”		Since the “if else” function is not new to us, the multiple “if else” function is but an overlaied “if else”. However, the multiple “if else” is not a ready to use function block, it evolves from

		the “if else” and serves the same function. The way to build the multiple “if else” function is as below: As we may see from the left image that there lies a blue gear-like icon at the top left corner of “else if” function. Click it and drag the “else if” sentence from the left side into the middle part between the “if else”. Then you will find the “if else” function is evolved into the multiple “if else” function. You can surely repeat above action to make a longer multiple “if else” function. But here, to illustrate its function, we will set the one time dragged multiple “if else” function as an example: As we can see from the left image that, in the multiple “if else” function, there are logic statement 1 and logic statement 2. Which means: There will be a judgement for the received information. If the logic statement 1 is met, the function

		to the right of the first "then" should be executed. Whereas the logic statement 2 is met, the function to the right of the second "then" should be executed.

Program

（1） Start a new program and name it as “escape the maze”

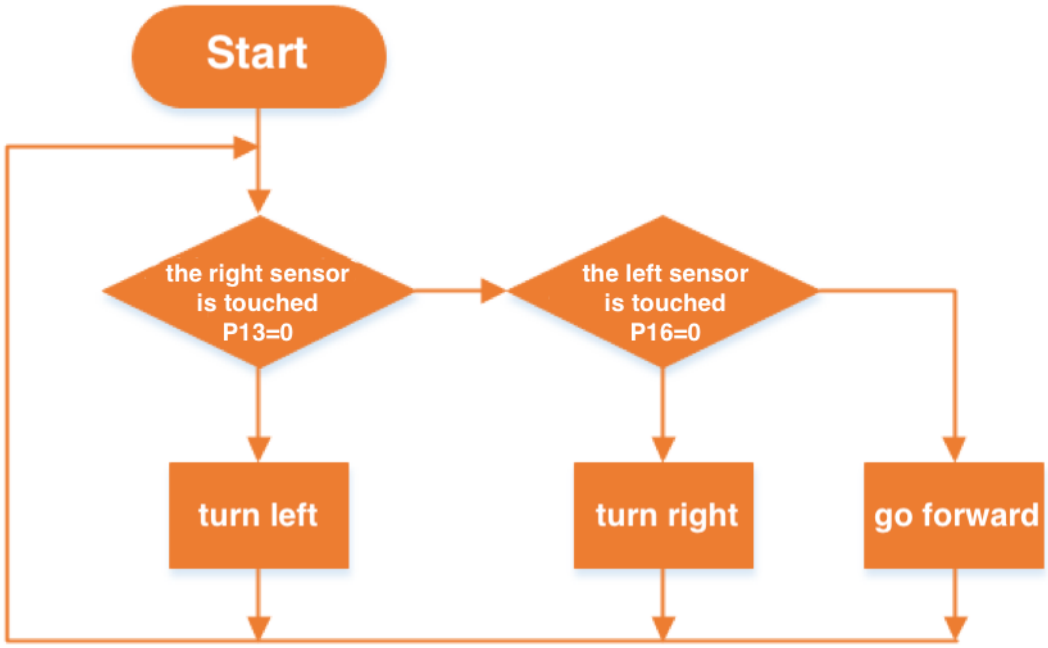
（2） Help Max:bot to escape the maze

Before programming, we should first get to know what Max:bot would do for avoiding obstacles.

If the left digital crash sensor is touched, Max:bot will then turn to the right;

If the right digital crash sensor is touched, Max:bot will then turn to the left.

The diagram below can be taking as a guidance:

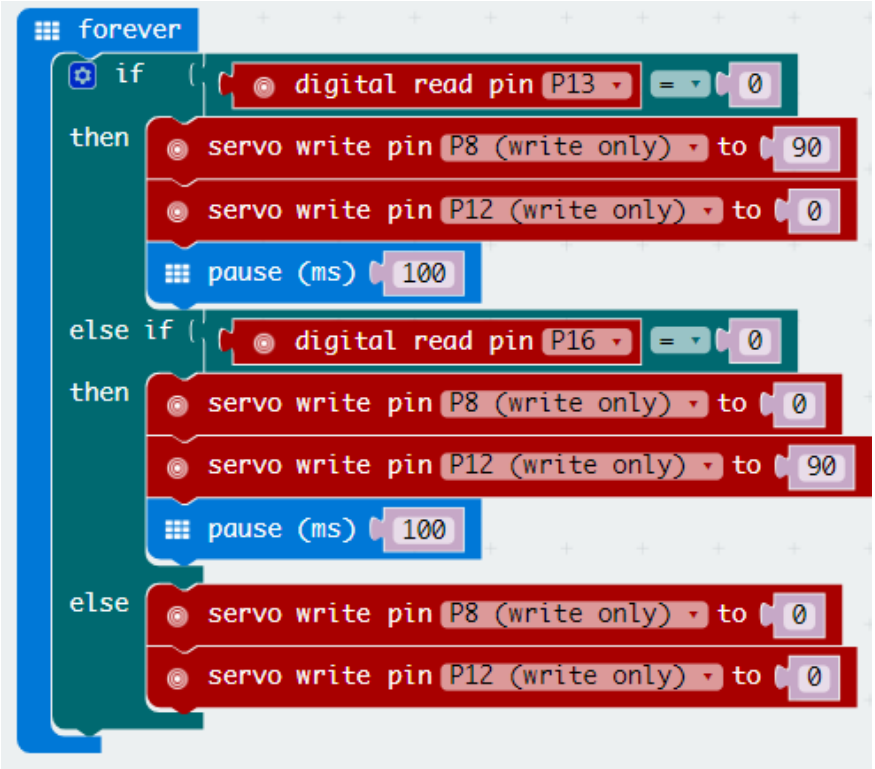




With the assistance of the diagram, please try to program it.

（3） Some above learnt programs will be combined in this part to help Max:bot escape the maze. Such as: turn left, turn right, go straight, etc. By putting all the programs mentioned above together, we have the final program as below:

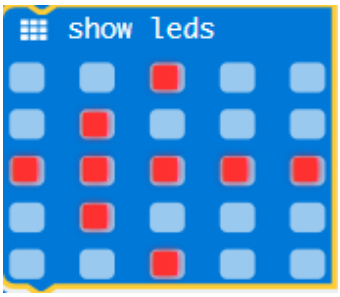
Please note: Different batteries output different voltages. So the value below in the “pause” function is just used for a reference. You have to try and adjust them based on the actual situation.



Exercise

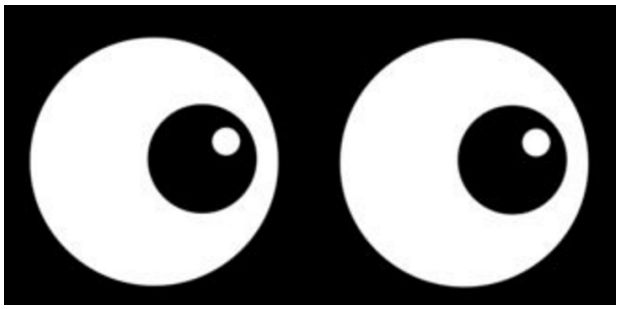
Max:bot, a really fast learner, now has made another step forward! Can you remember the "heartbeat" we made for him? How about to make him show us his direction when he is ready to turn.

Tips: You may need the help of the "show leds" function.



Chapter 3: Photoreceptive eyes

In previous chapters, since Max:bot is, like an alive creature with eyes, able to recognize and avoid obstacles, what else can we explore about his sensory system?

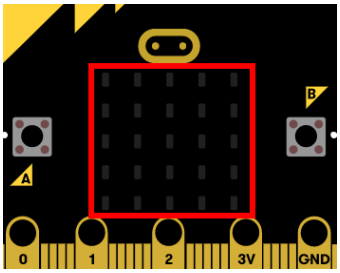


You will be surprised that Max:bot can even sense the brightness of light. In this chapter, we will work together to learn how to achieve this.

Goals

1. How to use the light sensor?
2. How to use the digital display module?
3. Diagram aided programming

Digital modules

Digital module(s)	Image(s)	Function(s)
The light sensor		The 5x5 LED matrix of Micro:bit is used to display patterns and detect light. It can, with an on-board light sensitive component, transfer the brightness of light into output values.

3.1 Adventure to brightness

When in surroundings with strong lights, people used to instinctively narrow their eyes for protection. Whereas being in surroundings with poor lights, they are not able to distinguish objects. Here come the questions: it is what that influences the light brightness. How the colors of lights influence its brightness?



To make above clear, we should ask Max:bot for help. As the saying goes: learn to walk before you run. So we will need first to use Micro:bit to transfer the brightness into output values.

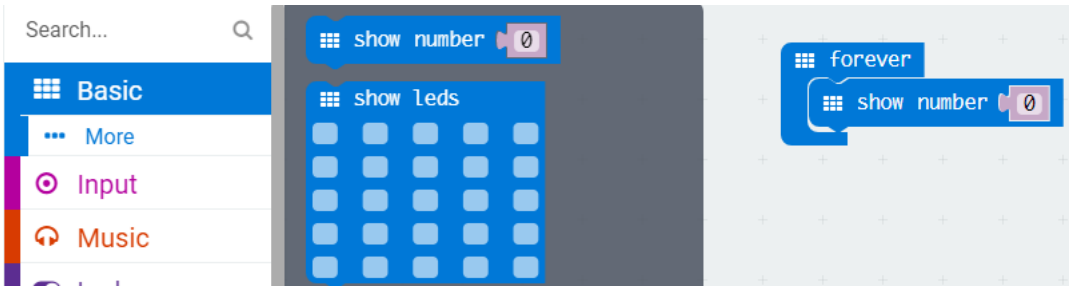
Key information

Function block(s)	Image(s)	Function(s)
"light level"		We can use the "light level" function to read the value of brightness. It ranges from 0 to 255, which means the brightness is from weak to strong.
"show number"		The "show number" function is used to show the received number.

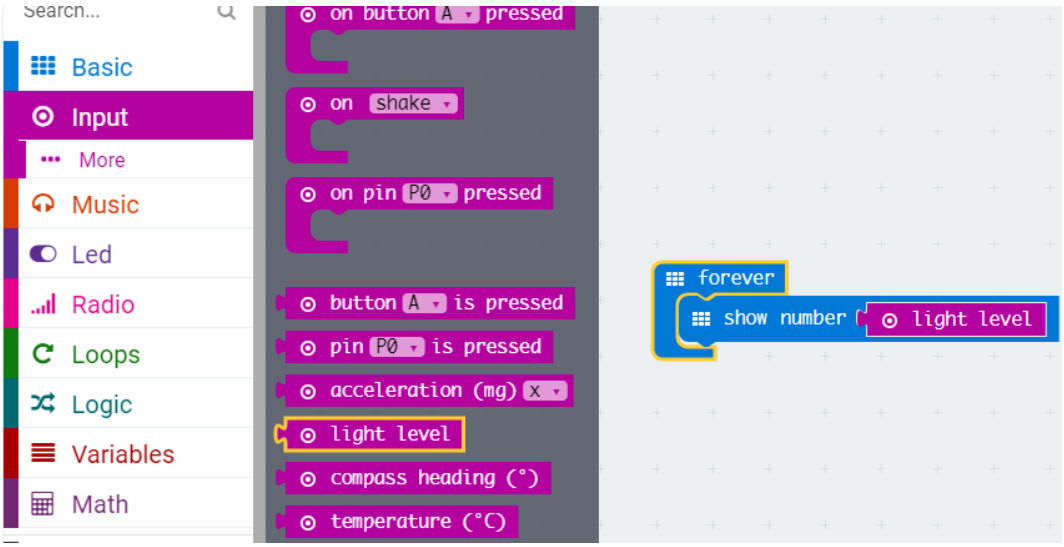
Program

（1） Start a new program and name it as "adventure to brightness".

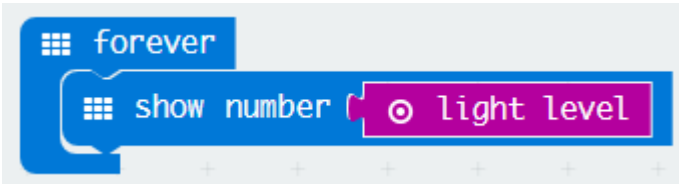
（2） Find the "show number" function from "Basic" and place it into the "forever" loop. The initial number is "0".

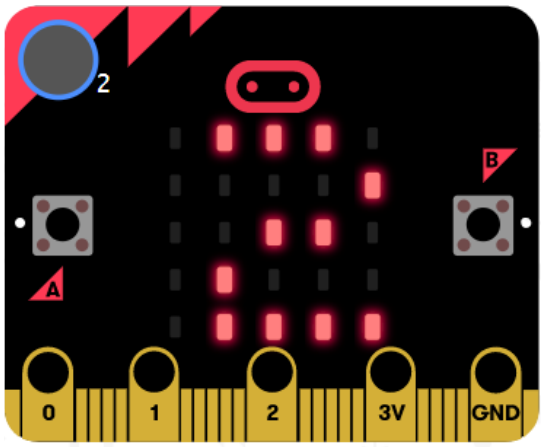


（3） As we have mentioned above that the "show number" function can be used to show the number that has been read by the "light level" function. So, in this step, we need first to find the "light level" function from the "Input" in the function area, and then place it to the "show number" function. Place them into the "forever" loop.



（4） Download the program to Max:bot. The detected brightness will be showed immediately on the 5x5 LED matrix of Micro:bit, then we can see the brightness with our own eyes.





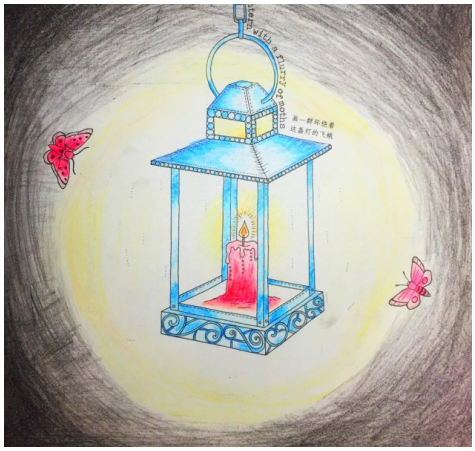
Exercise

There are different rooms with different lights in your house, why not go and detect their brightness? You can record them in the table below.

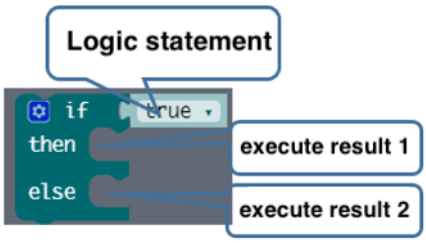
Surrounding(s)	Light brightness

3.2 The moth robot

Have you ever seen, at night, hundreds of moths flew against the streetlights? That is because moth is a kind of phototactic insect. Sounds very interesting, right? As have learnt the principle about how can Max:bot sense the brightness, we can now apply it into a moth robot.



Key information

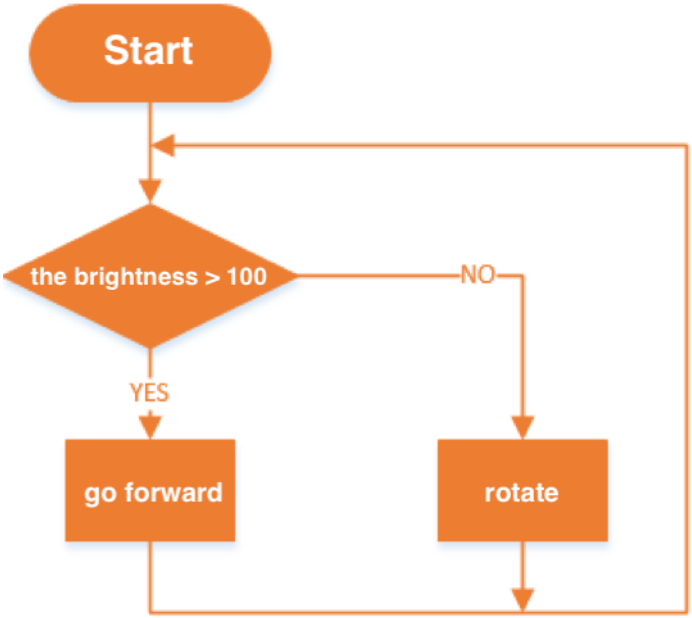
Function block(s)	Image(s)	Function(s)
"if else"		As we can see from the left image that, in the "if else" function, there lies the logic statement to the right of "if", the execute result 1 to the right of "then", and the execute result 2 to the right of "else". Which means: If the logic statement is met, the execute result 1 should be

		executed, whereas the execute result 2 should be executed if not met.

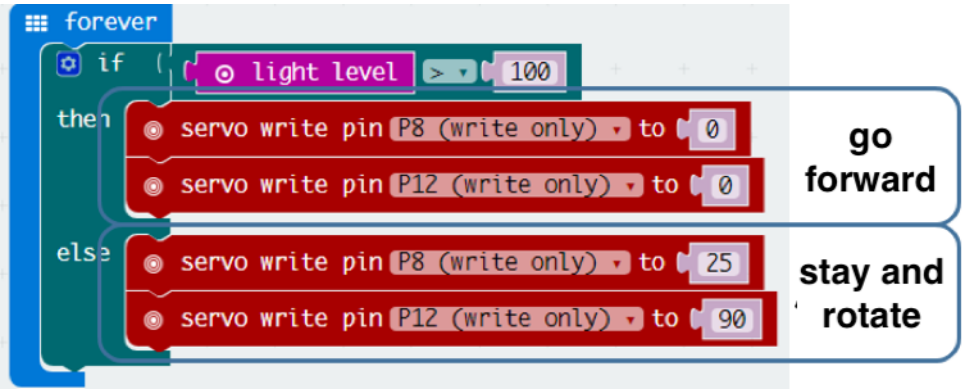
Program

（1） Start a new program and name it as “the moth robot”

（2） When the brightness reaches a certain range of value, the moth robot will go forward to the light. Whereas stay and rotate if the value is not reached. Below diagram can provide assistance to your program.



（3） In this part, the operator “=” will be used to judge if the brightness reaches the set value or not. And the “servo write” function is used to make the robot go forward or rotate. The “if else” function here is used to judge when should it go forward and when to stay and rotate. By putting all the functions listed above together, we will have the final program as below:



Please note: the value of brightness should neither be too large nor too small, it should be kept in a reasonable range.



If the value is too large, the moth robot will not go forward till a relatively strong light is given.

If the value is too small, the moth robot will then not likely to stop moving forward.

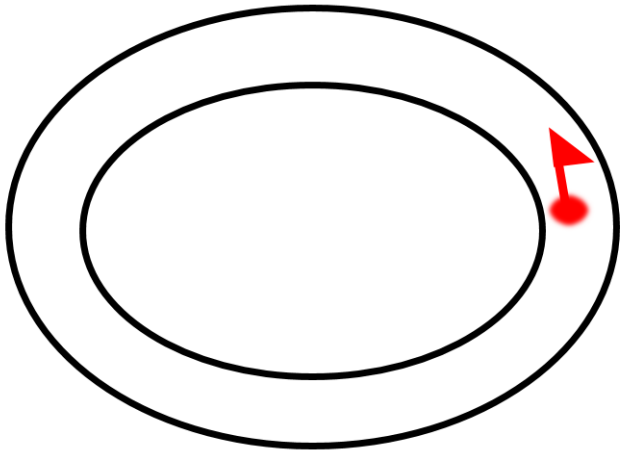
So you have to think twice and set a reasonable value.

（4） Download the final program to Max:bot. You can now play a light sensing game with this moth robot.

Exercise

Let's hold a race for the moth robot player.

You can invite your friends to join the race. Each of you will use the same electric torch to guide the moth robot move forward. The winner goes to the person that completes a lap in the shortest time. The starting line stands a red flag.



Tips: Maintain its speed within a reasonable range holds the key to success.

Chapter 4 Ultrasonic, the secret weapon

In the previous chapter, we have taught Max:bot how to use its "eyes" (the digital crash sensors and the light sensor) to avoid obstacles and sense the brightness of light. However, have you ever noticed that there are another pair of real eyes on his face. What can they do?

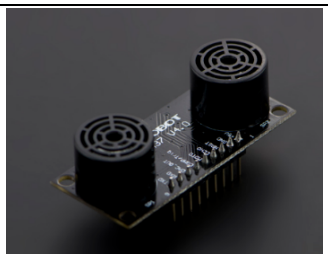
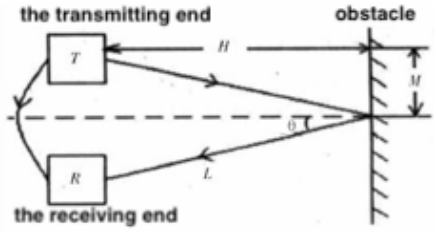


They are eyes with superpower. But you have to promise me to keep it as our secret. This pair of eyes are the secret weapon of Max:bot. Let's learn them together.

Goals:

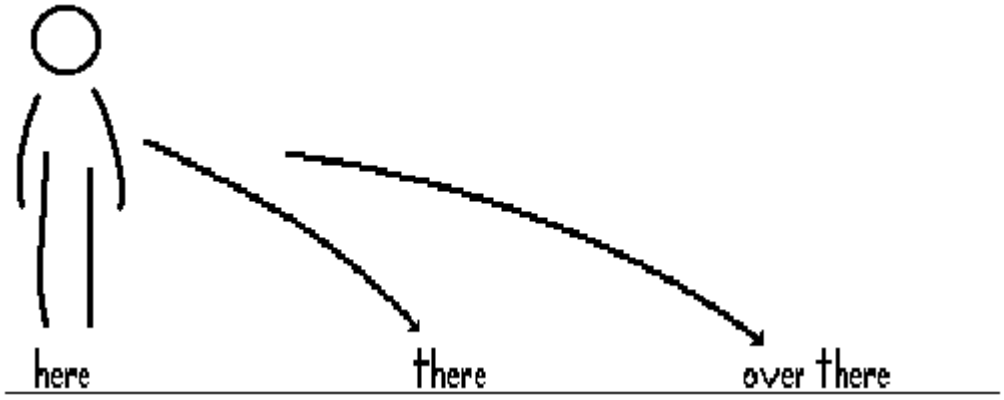
1. How the ultrasonic sensor works?
2. How to add package?
3. Diagram aided programming

Digital modules

Digital module(s)	Image(s)	Function(s)
The ultrasonic sensor		 The ultrasonic sensor can send out ultrasonic through the transmitting end. Once encountering obstacles, the sent out ultrasonic will be bounced back immediately and received by the receiving end. Thus Max:bot can detect obstacles.

4.1 ultrasonic measures distance

What may surprise you again is that Max:bot is more a reliable tool than a good companion. Once we have Max:bot, all other measuring tools, like rulers, tapes, etc., can be thrown away. In short, Max:bot can measure distance with ease.



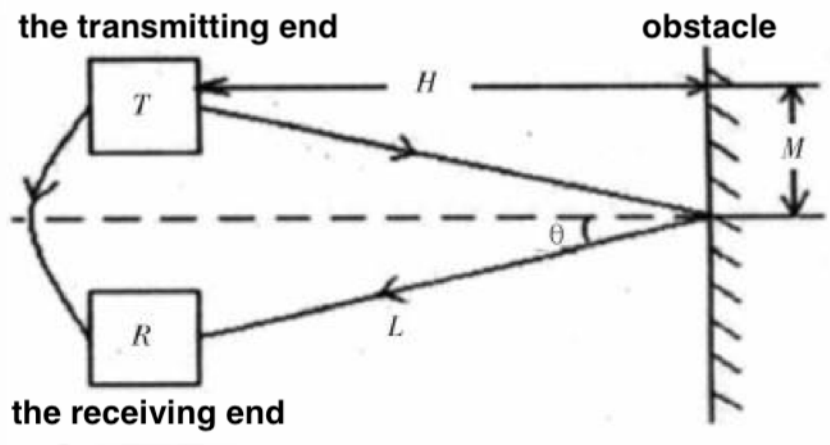
What is the ultrasonic?

As we may know, when vibrating, objects produce sound waves. Some of them can be heard by our ears, while the others cannot. Scientists named the vibrating times per second as the sound frequency with its unit named as Hertz. Almost every human is able to hear the sound frequencies ranging from 20 to 20000 Hertz.



However, most of us is not able to hear those sound frequencies that are higher than 20000 Hertz or lower that 20 Hertz. Therefore, for those sound frequencies that are higher than 20000 Hertz, scientists have another name for them, it is the "ultrasonic".

How can the ultrasonic sensor measure distance?

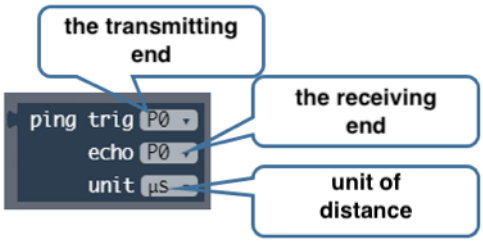


As we have learnt above that the ultrasonic sensor can, through the air, send out ultrasonic through the transmitting end. Once encountering obstacles,

the sent out ultrasonic will be bounced back immediately and received by the receiving end. Therefore, the ultrasonic sensor can calculate the distance through the formula: S=340t/2. Among which, "S" is the distance; "340" means ultrasonic travels at 340 meters per second in air; "t" is the total amount of time; "/2" means the time we need for the calculation is the time the ultrasonic travels to the obstacle. But the "t" (the total amount of time) is the time both for transmitting and receiving, so the "t" need to divide 2.

You may now have been convinced that Max:bot is a professional and powerful tool. He can even calculate the distance within one second! Let's try it!

Key information

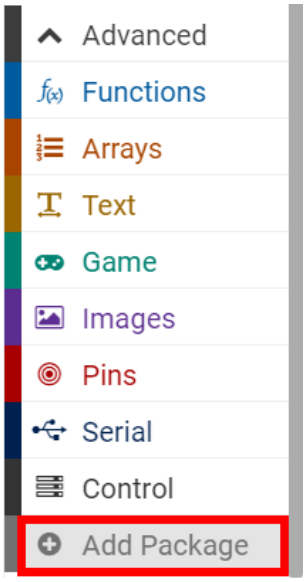
Function block(s)	Image(s)	Function(s)
"ping unit"		To enable the ultrasonic sensor calculate the distance, we need first to connect it correctly. As we can see from the image to the left that the transmitting end should be connected to the Pin behind "ping trig" and the receiving end to the Pin behind "echo". Besides, we need to select the correct unit (cm) to the right of "unit".

Program

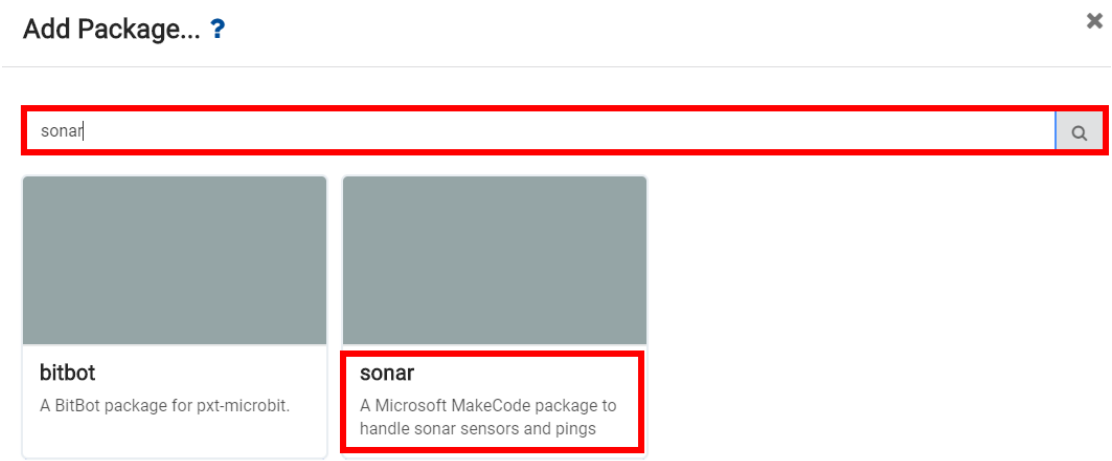
（1） Star a new program and name it as "ultrasonic measures distance"

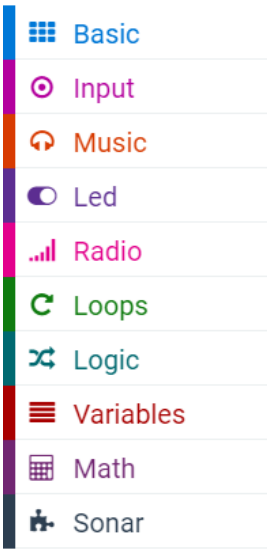
（2） Add package

The function area has listed some basic functions. However, for requirements of special functions, we have to use the "Add package". We need to add the "sonar" function in this part.

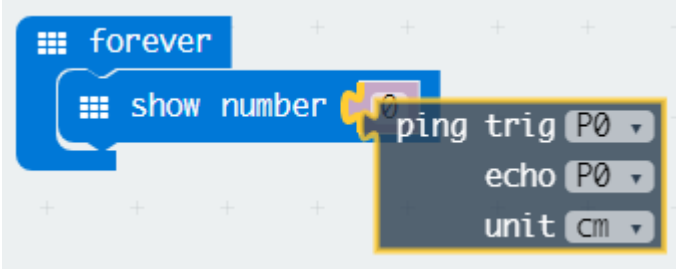


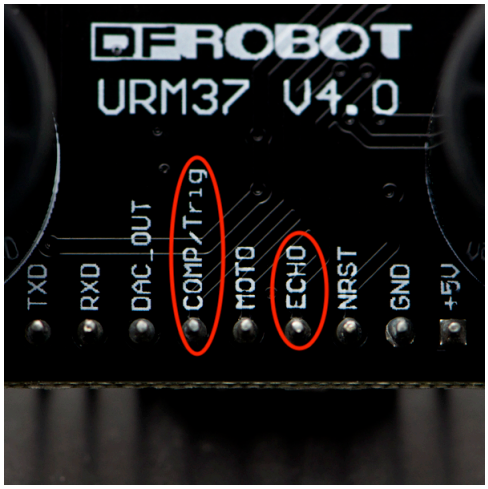
Click "Add package" and search "sonar". Then click the package named "sonar" and it will be listed in the function area automatically.





（3） We need first to place the “show number” function from “Basic” into the “forever” loop. Then put the “ping unit” function from “Sonar” in the “show number” function.

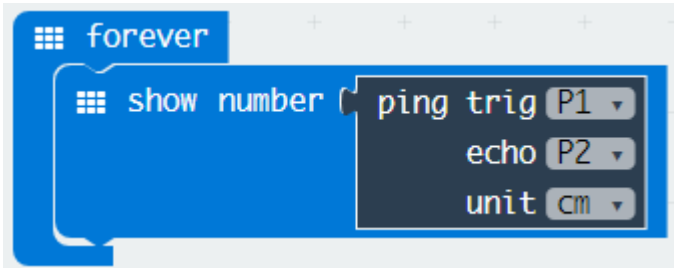




The image above is from the ultrasonic sensor. Among which, the “COMP/Trig” represents the transmitting end and should be connect to P1, and “ECHO” the receiving end to P2. While programming, in the

"ping unit" function, we should change the Pin of "trig" to P1, and the "echo" to P2.

（4） By putting all the functions listed above together, we will have the final program as below. Download it to Max:bot and the calculated distance between Max:bot and the obstacle will be displayed.



Please note: Keep in mind to swich Max:bot on. To make the calculation more acurate, it is better to maintain the distance between Max:bot and the obstacle within 5cm to 300cm.

Exercise

Above we have learnt how to use Max:bot to measure a distance. Do you have any idea about how to use it measure your height? Just try it!



Tips: To make the calculation more accurate, you may need to:

1. Pay attention to where and what direction the ultrasonic sensor locates.
2. To avoid major deviations, it is better to adjust Max:bot within 10cm.

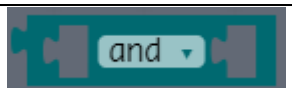
4.2 Car safeguard

Whether you believe or not, traffic accidences are, nowadays, taking place almost anywhere at any time. Cars make our life more convenient, but at the same time, put us in danger. How can we help drivers to be more sensitive about those potential dangerous? Max:bot does the job.



Key information

Function block(s)	Image(s)	Function(s)

“item”		The “item” function represents a variable. We can use it to realize a dynamic effect.
“set item to”		We can use the “set to” function to set a value for the variable ”item”. The image to the left means: the value of the variable named as “item” equals 0.
“and”		When the “and” function is used, it means that both its left and right parts should be met simultaneously.

Program

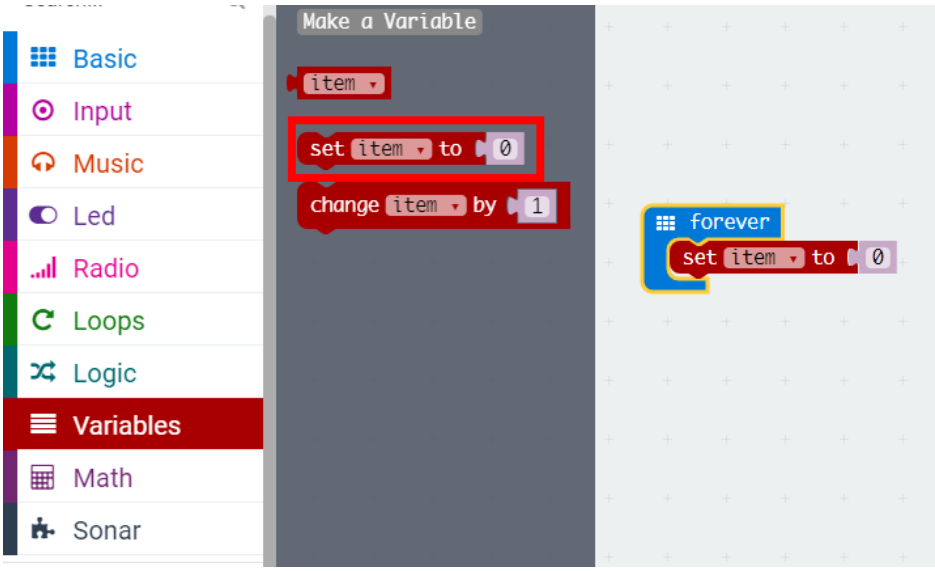
（1） Start a new program and name it as “car safeguard”

（2） Item

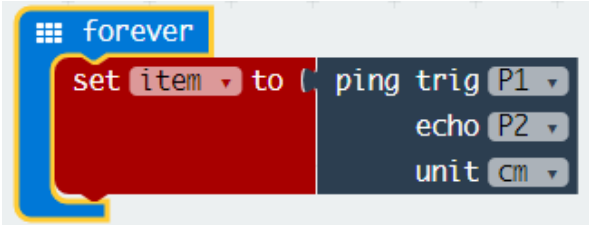
How to use the variable “item”?

Sensory system of Max:bot carry real-time detections to the changing surroundings. For instance, brightness may vary from light to light. Which means Max:bot requires a special function to sense and transform the variable brightness into a certain value. Thus we have the “set item to” function. Among which, the “item” in the middle is the detected variable. The whole function means: set the value of the variable named “item” to.

Find the “set item to” function from “Variables”, and place it into the “forever” loop.



In this part, the data that has been detected in real time is the distance that the ultrasonic travels. Therefore, we need to set the value of the variable according to the detected distance. We have learnt how to use the "ping unit" function to calculate the distance, so, in this part, the "ping unit" function should be placed in the "set item to" function.

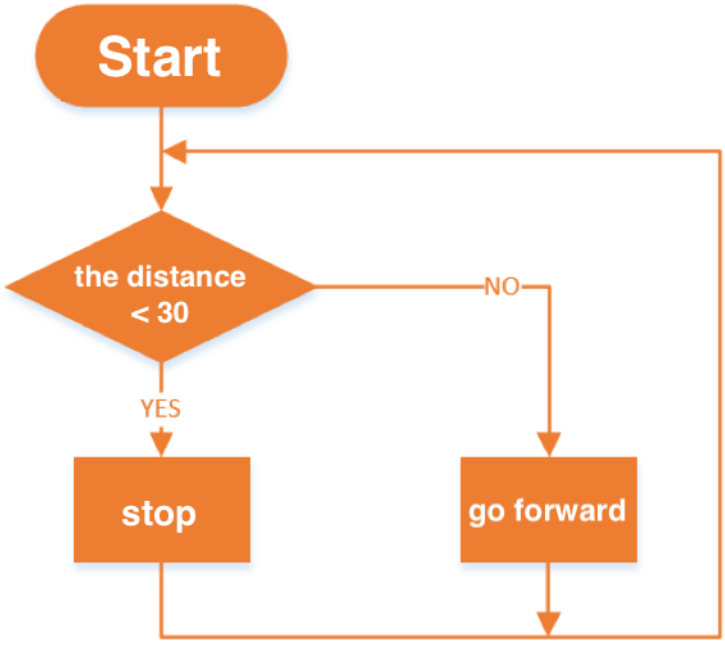


Since the value of the variable named "item" has been set, we can now use the variable "item" directly in later parts. You can find the variable "item" from the "Variables" in the function area.

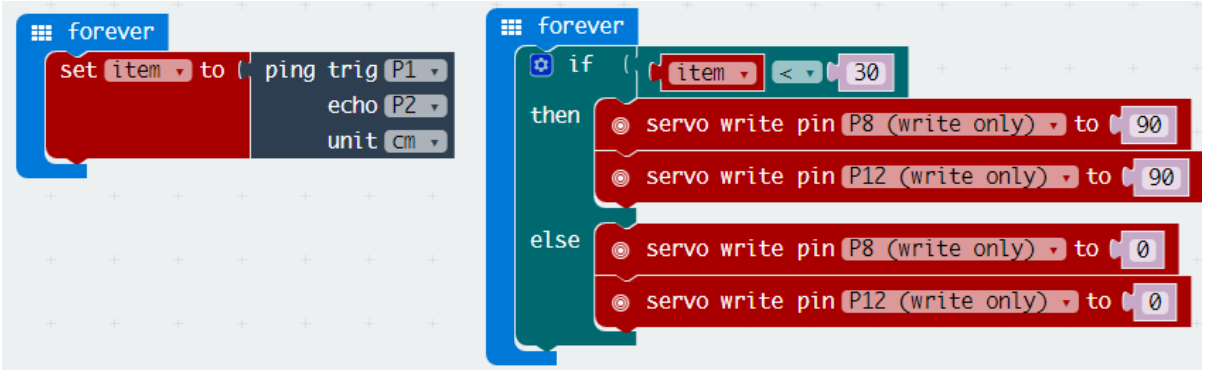


（3） Once the distance between the obstacle ahead and Max:bot is shorter than the preset value, Max:bot will then stop automatically to avoid rear-end collision.

It sounds like a very good idea, but how to achieve this? We can use the diagram below as a guidance.

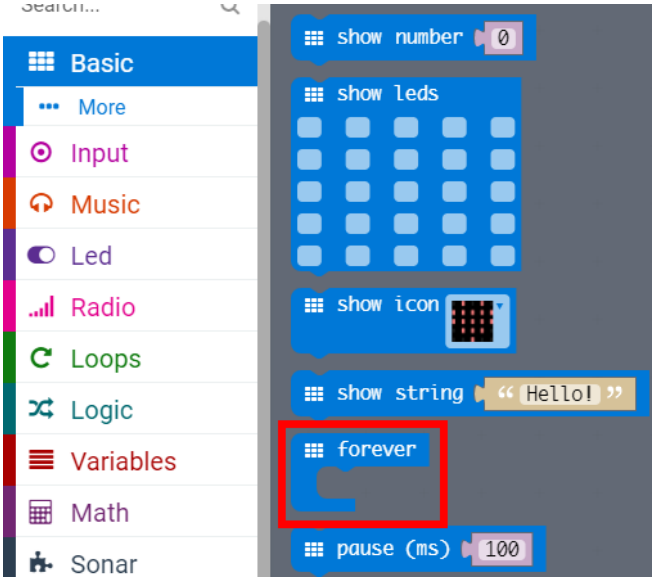


（4） Combine all the function blocks listed above together, the final program goes as below. Download it to Max:bot. Then you can enjoy a nice and safe road trip!



As you may have noticed that there are two “forever” loops. It is because that we need Max:bot to operate two programs simultaneously without influence each other. On the one hand, it should detect the value of the variable, and, on the other hand, it itself should determine when to stop and when to go forward. So we need

to use two “forever” loops in this part. You can find the “forever” loop from the “Basic” in the function area.



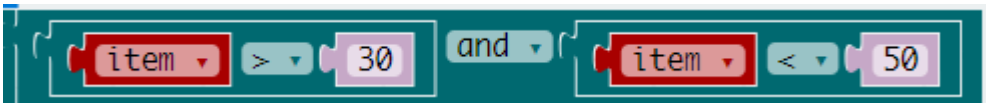
Exercise

When driving, the car will not stop completely but to gradually slow down. When the distance between it and the car ahead becomes closer and closer, its speed will be reduced correspondingly.

Let’s try it!



Tips: the “and” function can be used to judge different distances.



Chapter 5: Max:bot Go!

Can you remember how many eyes do Max:bot have? Firstly, he has the "eyes" (the digital crash sensors and the light sensor) to avoid obstacles and sense the brightness; then he has the eye on his face (the ultrasonic sensor) to measure the distance between it and the obstacle. You may wonder if he has any other eyes? The answer is absolutely yes. In this chapter, another eye will be introduced. Max:bot keeps this eye toward the ground to avoid falling down from a height. By the way, with the help of this eye, Max:bot can go along a fixed trace. Let's explore it!

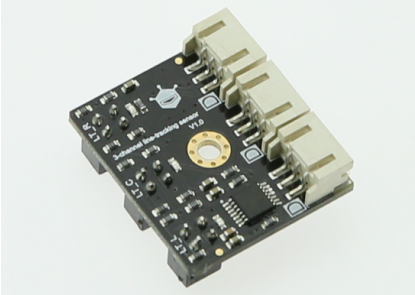


Goals:

1. How to use the line tracking(following) sensor?
2. How to use the "and" function?

Digital modules

Digital module(s)	Image(s)	Function(s)

The line tracking(following) sensor		The line tracking(following) sensor enables Max:bot to trace a white line in a black background, or a black line in a white background.

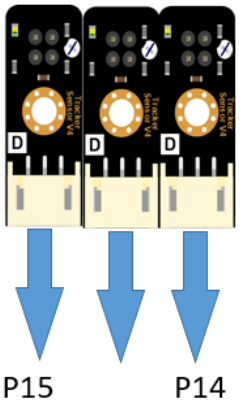
5.1 Max:bot wake up!

Watch out! There is a precipice ahead! We have to stop now! Hey, take it easy, Max:bot will never do such a stupid thing. Thanks to the line tracking(following) sensor, he can always wake up to the danger at last moment.



But how does the line tracking(following) sensor work?

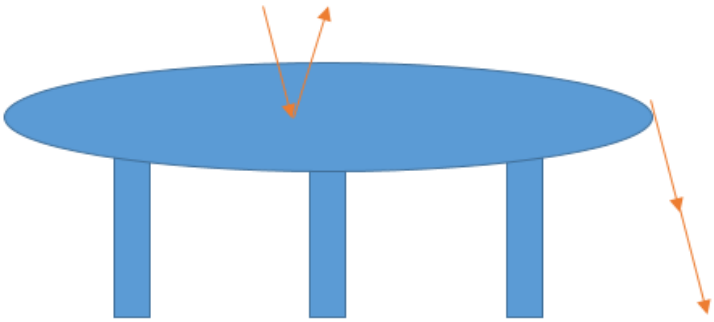
There are three probes in the line tracking(following) sensor. But in this part, we will use only two of them. Connect the left probe to P15 and the right to P14.



In addition, both the left and the right probes can be used as separated infrared sensors.



Which means that both the probes can transmit and receive infrared. The way the probe works is similar to the ultrasonic sensor's. However, it will not calculate the distance according to the received information but only to judge if there are obstacles or not.



How does the line tracking(following) sensor work?

When being putting on a table, both of the probes transmit infrared. The transmitted infrared encounters the obstacle (the table) and then be bounced back and receive by them. Thus, the indicator lights up and the Max:bot receives a high voltage (presented as 1)

Whereas when Max:bot goes to the edge of the desk, the transmitted infrared will not be bounced back. Thus, the indicator turns off and Max:bot receives a low voltage(presented as 0).

Key information

Function block(s)	Image(s)	Function(s)
"and"		When the "and" function is used, it means that both its left and right parts should be met simultaneously.

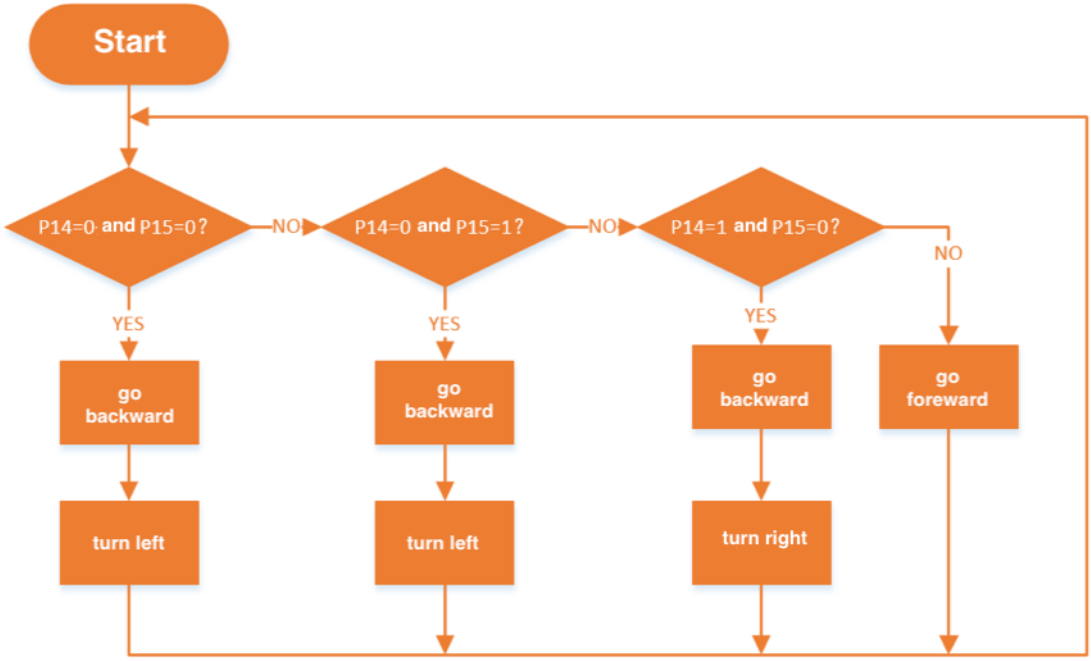
Program

（1） Start a new program and name it as "Max:bot wake up!".

（2） In this program, we will only use the left and the right probes of the tracking(following) sensor. Connect them separately to P15 and P14. Below four situations should be taken into consideration:

status	description	status of the probes
	both probes detect obstacles(the desk)	P14=1; P15=1;
	both probes detect nothing	P14=0; P15=0;
	the left probe detect an obstacle, and the right probe detect nothing	P14=0; P15=1;
	the left probe detect nothing, and the right probe detect an obstacle	P14=1; P15=0;

（3） According to the four potential situations, we can now draw a diagram to program Max:bot.

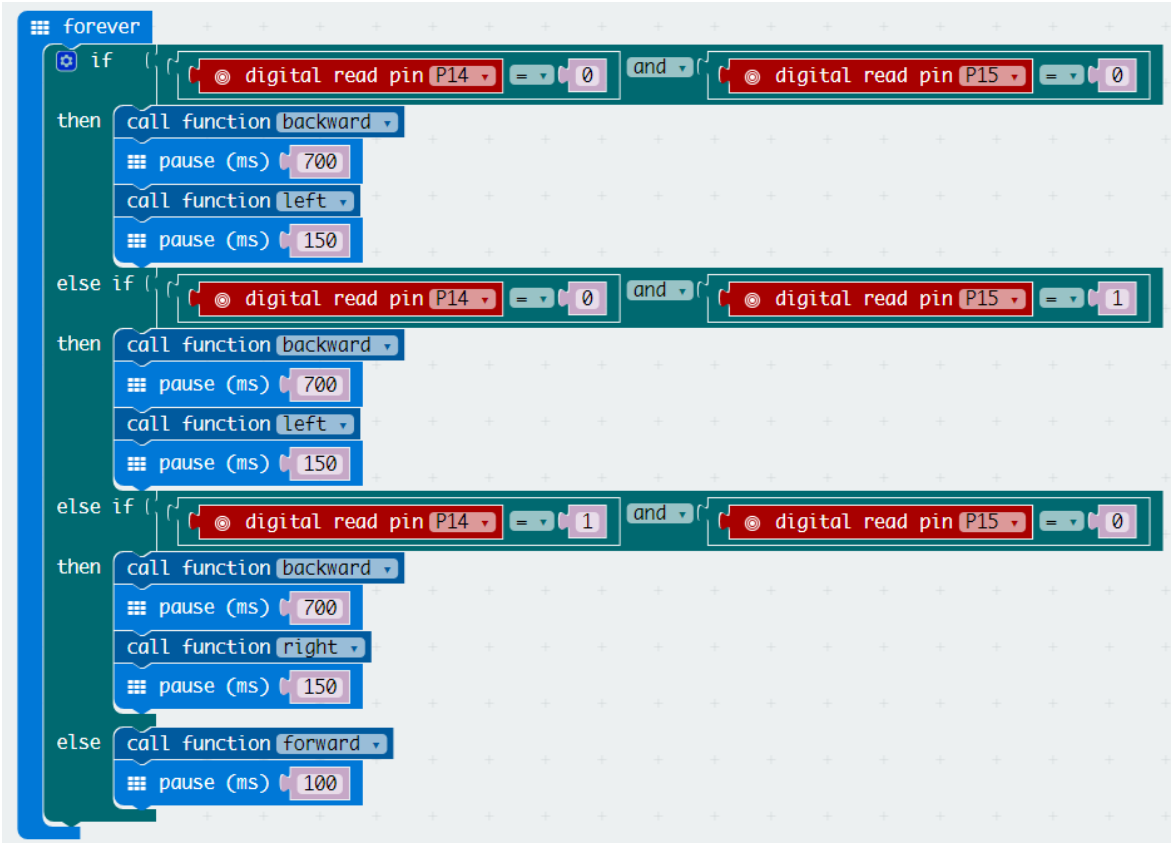


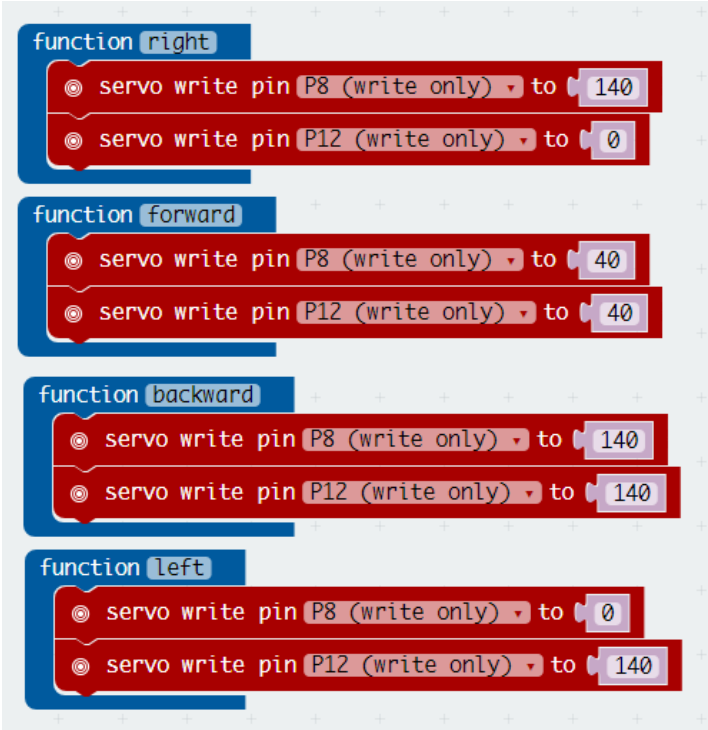
（4） Below we have applied so many functions that have been learnt before. Let's review them together.

In the first place, we have to use the sub-loop "function" to enable Max:bot turn right, go forward, turn left and go backward.

The multiple "if else" function here is used to judge which action Max:bot should take: The first "if then" means both the probes detect the precipice, Max:bot will then go backward and turn left; the second "if then" indicates that the right probe detects the precipice, Max:bot will then go backward and turn left; the third "if else" shows the left probe detects the precipice, Max:bot will then go backward and turn right. The last "else" means if none of the three "if then" is met, Max:bot will then keep going forward.

By putting all the functions listed above together, the final program goes as below. Download it to Max:bot. The smart robot now can freely dance on the table.


forever
if digital read pin P14 = 0 and digital read pin P15 = 0
then call function backward
pause (ms) 700
call function left
pause (ms) 150
else if digital read pin P14 = 0 and digital read pin P15 = 1
then call function backward
pause (ms) 700
call function left
pause (ms) 150
else if digital read pin P14 = 1 and digital read pin P15 = 0
then call function backward
pause (ms) 700
call function right
pause (ms) 150
else call function forward
pause (ms) 100


function right
servo write pin P8 (write only) to 140
servo write pin P12 (write only) to 0
function forward
servo write pin P8 (write only) to 40
servo write pin P12 (write only) to 40
function backward
servo write pin P8 (write only) to 140
servo write pin P12 (write only) to 140
function left
servo write pin P8 (write only) to 0
servo write pin P12 (write only) to 140

Tips： The speed of Max:bot should be maintained within a reasonable range to avoid falling from the table.

Exercise

Can you find out some other objects that should be avoided falling from a height?



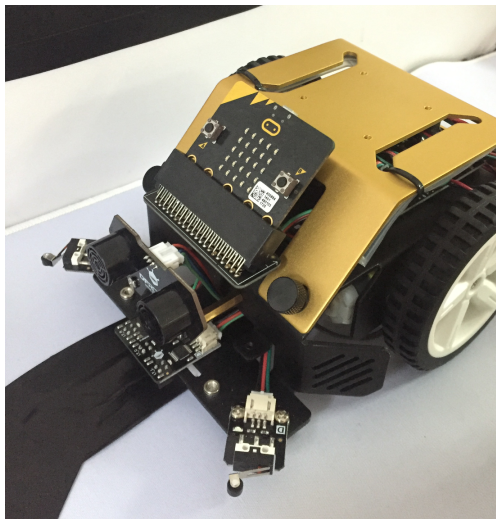
Nowadays, more and more advanced tools are embraced by human beings with opened arms. Such as the self-charging sweeping robot which can clean the floor on schedule. To avoid falling from stairs, it should be programmed with fall arrest function.

There lines an array of probes at the bottom edge of the sweeping robot to protect it from falling from stairs.

Once you thought of other objects that should be protected from falling down, you can note them down in the blank below. The more the better.

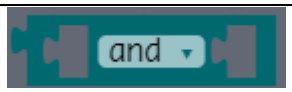
5.2 Trace the track

Give Max:bot a long enough track, he will trace it across the world.



Let's play the Treasure Hunt!

Key information

Function block(s)	Image(s)	Function(s)
"and"		When the "and" function is used, it means that both its left and the right parts should be met simultaneously.

Program

（1） Start a new project and name it as "trace the track"

（2） In this chapter, we need first to pave a black track by a black pen or a black tape. Then we should program to enable Max:bot to trace along it.



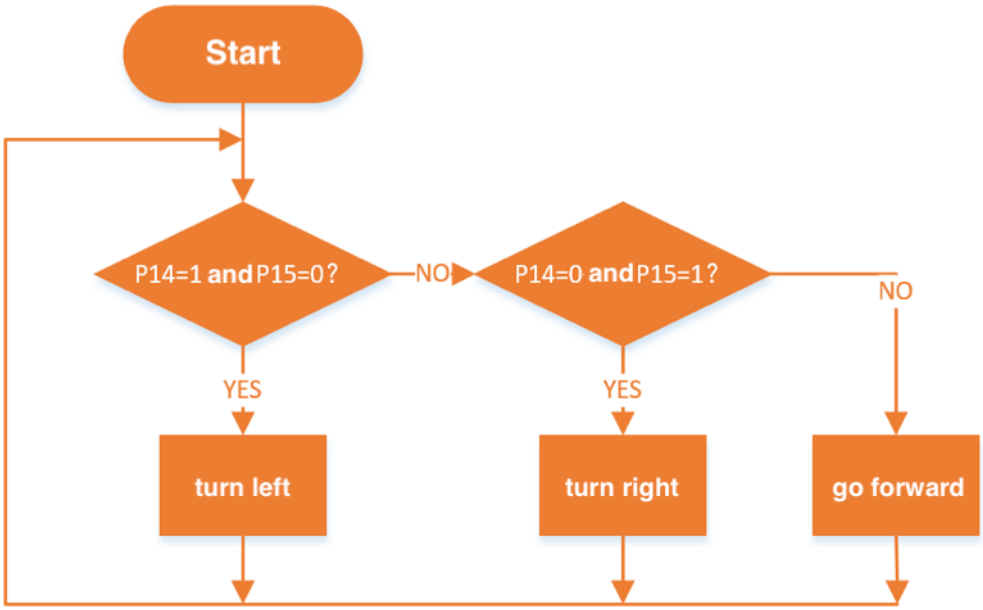
Please note: the black track should be wide enough that both the left (P14) and the right (P15) probes of the tracking(following) sensors can detect it simultaneously.

（3） Below situations should be taken into consideration:

status	description	status of the probes
	**both the probes detect the black track**	P14=0; P15=0;
	**the left probe detects the track, while the right detects nothing**	P14=0; P15=1;
	**the left probe detects nothing, while the right detects the track**	P14=1; P15=0;

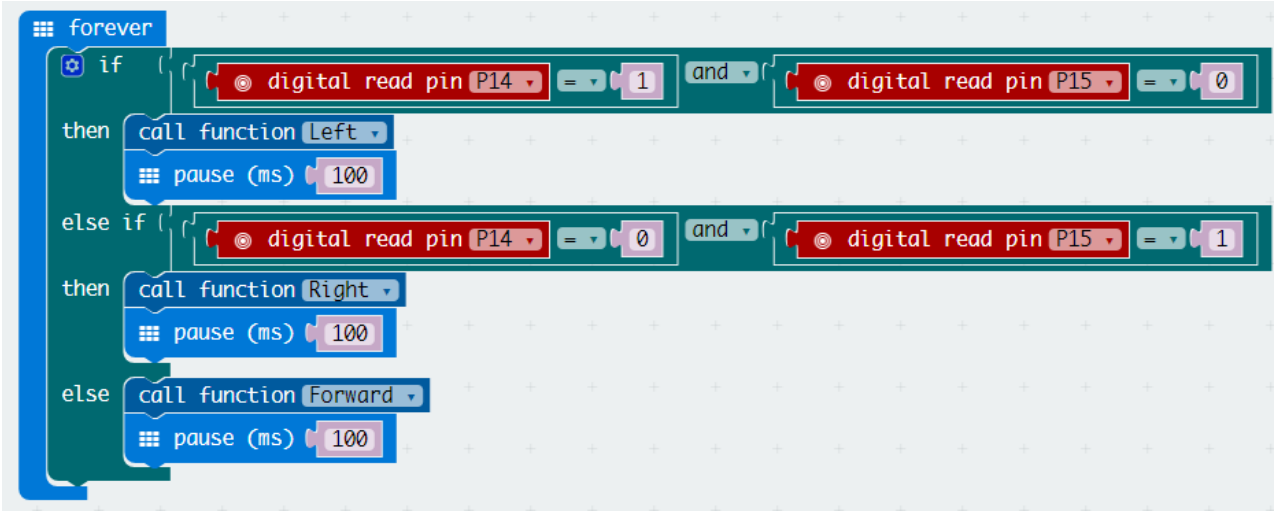
（4） As have analyzed above, we can now draw a diagram for programming.

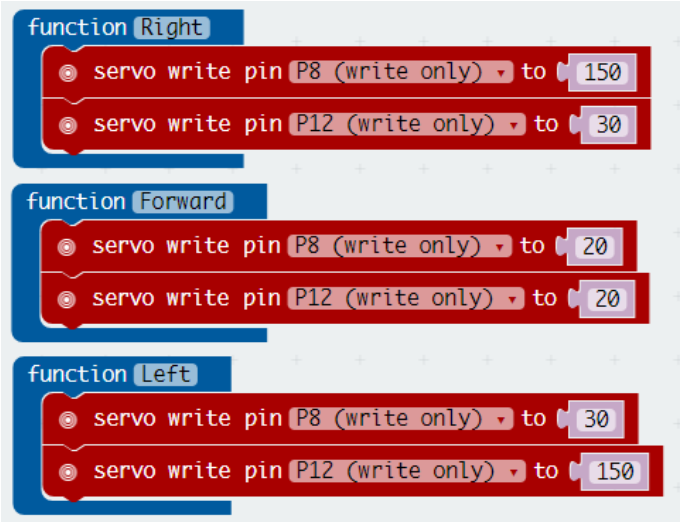




（5） In this program, like the final program of the last part, we use the "function" to enable Max:bot to turn right, go forward and turn left. The multiple "is else" function is applied to tell Max:bot when to turn left, when to turn right and when to keep going forward. Combine all the function blocks listed above together, we will have the final program as below.

Download the final program to Max:bit. It will then go and trace along the black track.

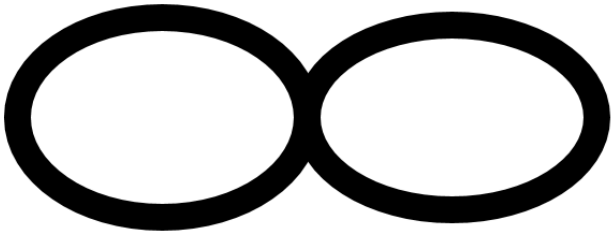




Tips: To keep Max:bot always tracing along the track, we need to maintain its speed and pause time within a reasonable range.

Exercise

Max:bot can now travel around the world by the black track! You may not believe it because some roads are quite complex. Are Max:bot able to trace along the complicated tracks like below?



In the course of programming, you may encountered some troubles. Note them down and discuss with your friends to find out the solution.

-
-
-

Chapter 6: Radio communication

With one Max:bot, you can almost control everything. What if you have two Max:bots in hand?

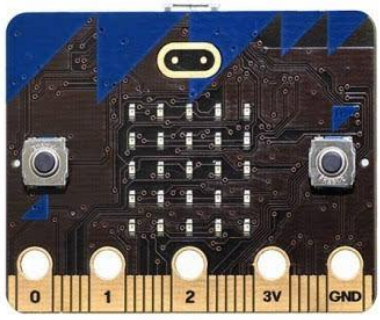


As we have learnt that Micro:bit is the brain of Max:bot. Other than the powerful functions we experienced in previous chapters, Micro:bit has another function which can realize the remote control. Let's explore it!

Goals:

1. How to use the radio communication function?
2. How to use the accelerometer?

Digital modules

Digital module(s)	Image(s)	Function(s)
Micro:bit		Micro:bit has mounted with the radio module and the accelerometer.

6.1 Say Hi

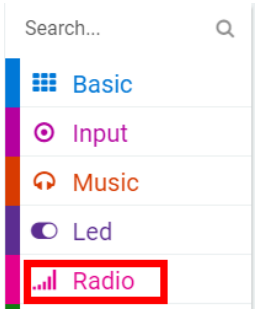
When two Max:bots meet up with each other, what would they do? It must be say Hi. But how to achieve this?



The radio communication function does the job. When two Max:bots come across each other, one of them will send out a signal via Micro:bit, while the other receives the signal and will respond to it.

How to use the "Radio" function?

We will need two micro: bits in this part, one sends out message (transmitting end), the other receives the signal (receiving end). The micro: bit communicates over radio.



We need first to get to know the three most commonly used functions under "Radio":

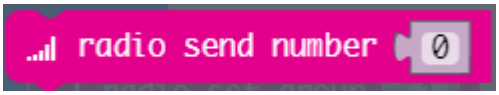
1、"set group":



Function description: There are a total of 255 different radio channels. The transmitter and the receiver should be set in the same group to be able to communicate.

Please note: different radio channels should be set for different players to avoid any confusion.

2、"send number" and "send string":

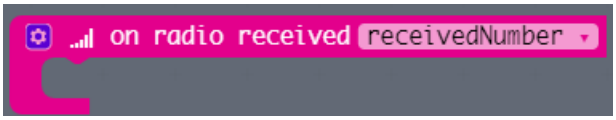


Function description: send out a number or a variable over radio.



Function description: send out a string over radio.

3、"on radio received number" and "on radio received string":

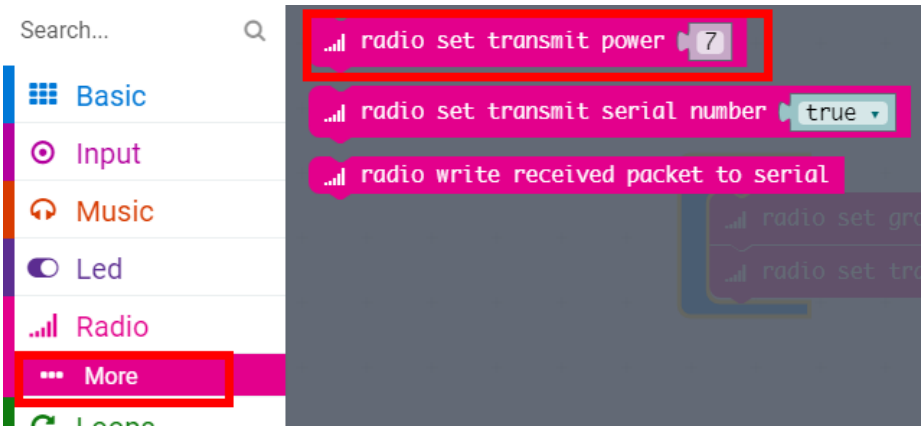


Function description: Upon a number or a variable is received, it will execute the program inside the "on radio received number" loop



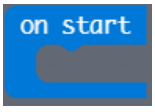
Function description: Upon a string is received, it will execute the program inside the "on radio received string" loop

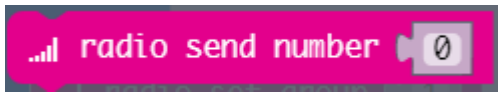
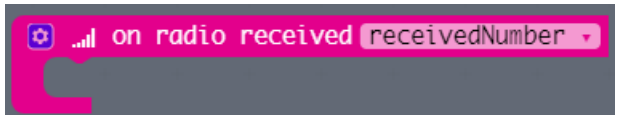
4、"set transmit power"



There are 7 grades can be selected for radio transmit power. The higher the grade is, the farther the power transmits.

Key information

Function block(s)	Image(s)	Function(s)
"on start"		Functions that are placed into the "on start" will be executed once only.

"set group"		Define the radio channel.
"radio send number"		"0" send out the number "0" over radio
"on radio received number"		Upon a number is received, it will execute the program inside the "on radio received" loop.
"set transmit power"		Set the grade for radio transmit power.

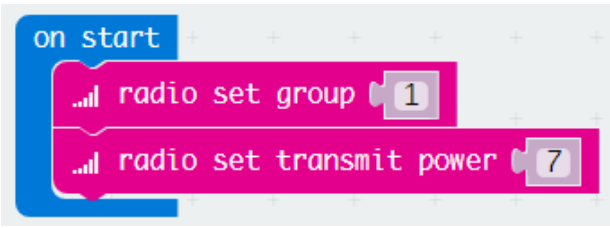
Program

The transmitting end

（1） Start a new project and name it as "say hi the transmitting end"

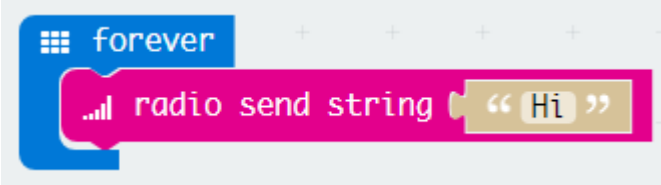
（2） Initial setting the transmitting end

Functions that are placed into the "on start" will be executed once only. Which means both the "set group" and the "set transmit power" will be executed only for one time after power up.

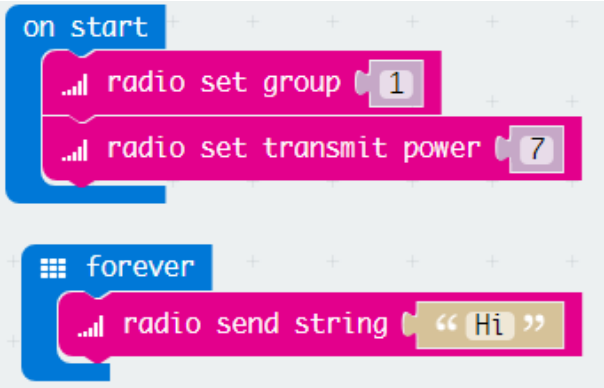


（3） Set the transmitting content

Since the Max:bot will say Hi to each other, so we need to use the "radio send string" function to send out the "Hi" over radio.



（4） Combine all the function blocks listed above together, the final program of the transmitting end goes as below. Download it to the transmitting end.



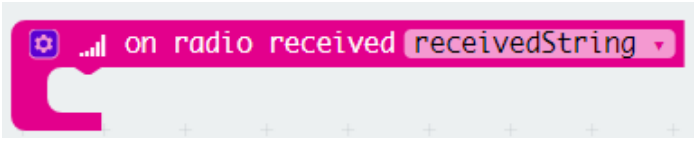
The receiving end

（1） Start a new project and name it as “say hi the receiving end”

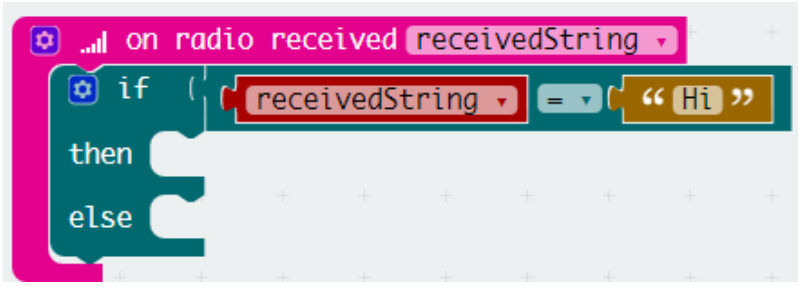
（2） Set the same radio channel as the transmitting end via “set group”. Place it into “on start”.



（3） The "on radio received string" function will be used to judge whether the transmitted information is received or not.



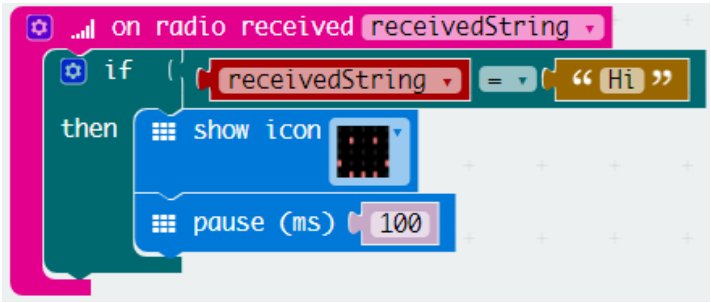
（4） Below we use the "if else" function to judge whether the received information is “Hi” or not.



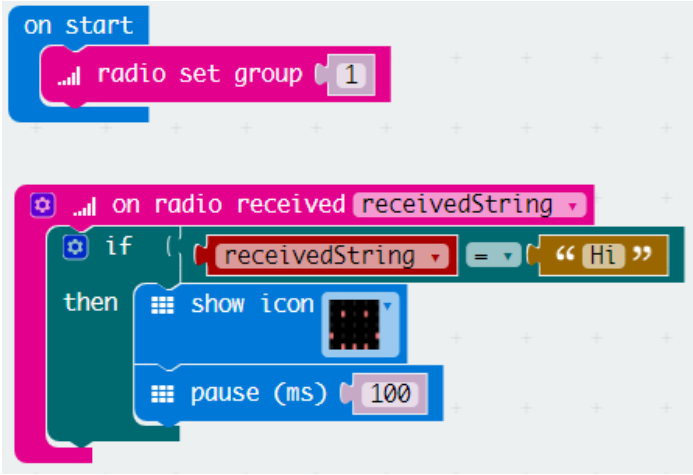
Please note: “Hi” is in the form of text. So the “” function will be used in this part to read the text. You can find it from the “Text” in the function area.



（5） If the “Hi” is received by Max:bot via radio, the robot will then display a smiling face. We will apply the “show icon” and the “pause” function to generate a smiling face.



（6） By putting all the functions listed above together, the final program goes as below. Download it to the receiving end.



Switch on both the receiving Max:bot and the transmitting Max:bot, the smiling face then can be seen in the receiving Max:bot.

Exercise

On some occasions, like spy films, we need to identify each other through a certain code. Now let's play a spy game!

One Max:bot sends out a code, like 1231; while when the other received the code, it will then display a corresponding code, as 1321. If failed, he will then be proved not one of yours.



6.2 Motion sensing racing

Have you ever played a motion sensing game, like racing? This kind of racing requires you to drive the car on the screen through real body move.



With Micro:bits, you can play the motion sensing game. The on-board accelerometer does the job.

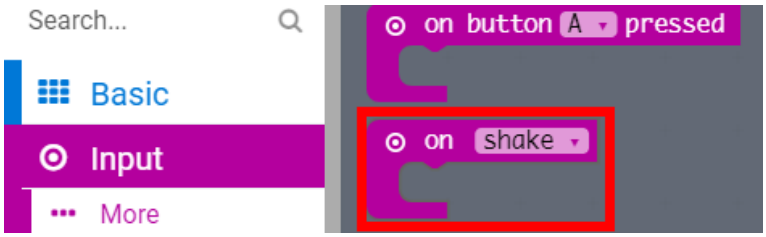
What is the accelertometer?

An accelerometer is a device that measures proper acceleration. The on-board accelerometer can be used in combination with other functions for telling Micor:bit's motions. Including its moving direction, angle, gesture, etc.

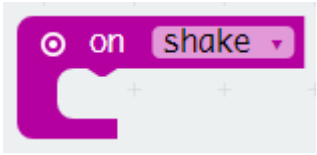
This kind of accelerometers can also be found in mobile phones, which enable their users to play motion sensing games. Moreover, the accelerometer, serves as one of the most important parts of the smart bracelet, can read motions to provide a step-counting function.



In the function area, we can use the "on" function from the "Input" to read the motions of Micro:bit.

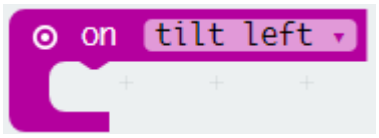


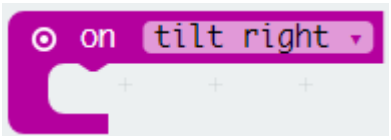
1、 “on shake”



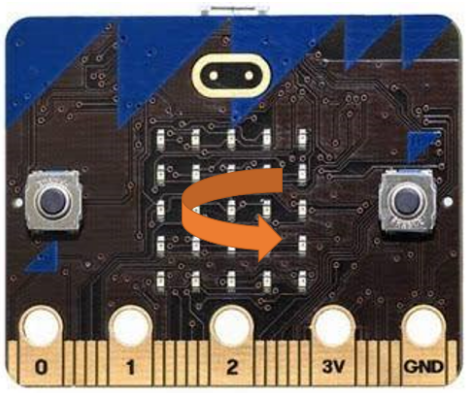
Function description: to judge whether Micro:bot is shaking or not. This function is similar to the step-counting function of a pedometer.

2、 ”on tilt left” and “on tilt right”

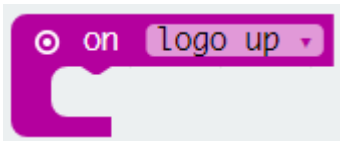




Function description: to judge whether Micro:bit is tilting to the left or to the right.

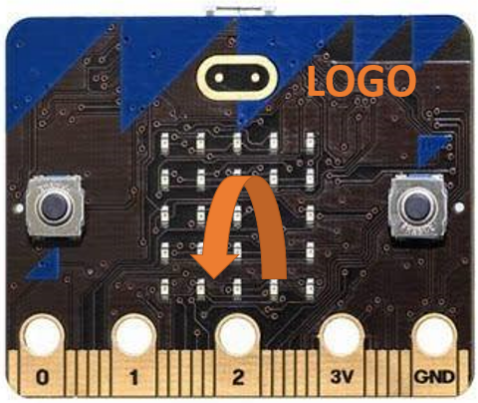


3、 “on logo up” and “on logo down”

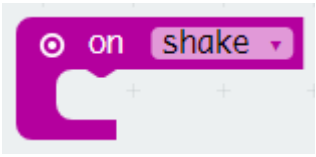
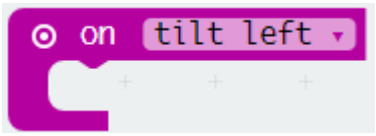




Function description: to judge whether Micro:bit is tilting towards the upside or towards the downside. Among which, the upside has a LOGO looks like: , whereas the downside has none.



Key information

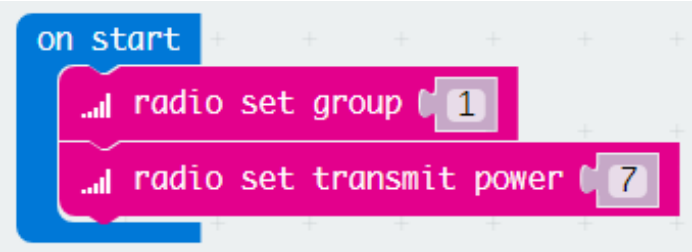
Function block(s)	Image(s)	Function(s)
"on shake"		To judge whether Micro:bit is shaking or not.
"on tilt left"		To judge whether Mirco:bit is tilting to the left or to the right.
"on logo up"		To judge whether Micro:bit is tilting to the LOGO side or not.
"show arrow"		To display an arrow on Micro:bit. The "North" means the arrow points to north. There are other seven directions can be selected.

Program

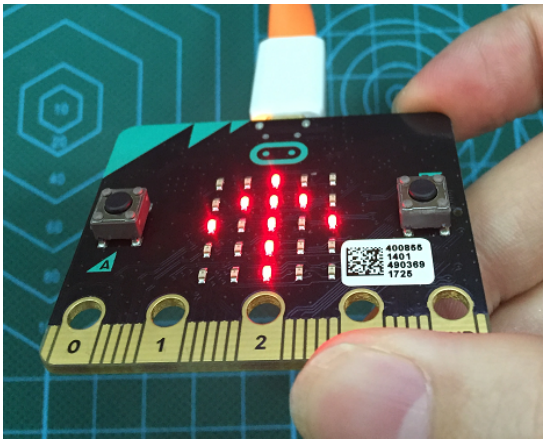
The transmitting end

（1） Start a new project and name it as "sensing racing the transmitting end"

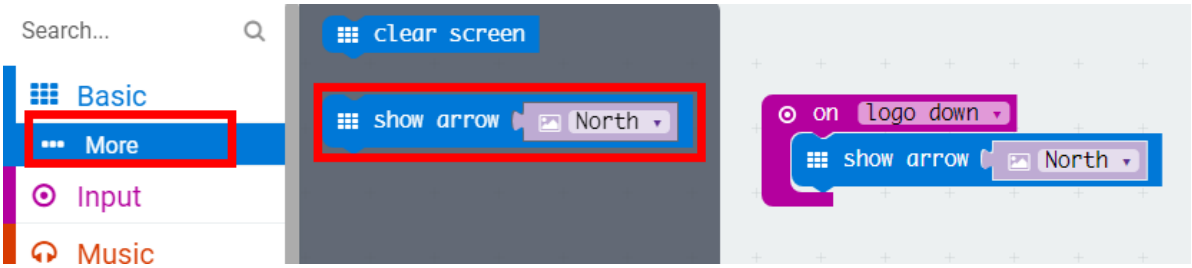
（2） Use the “on start” to initial set the transmitting end.



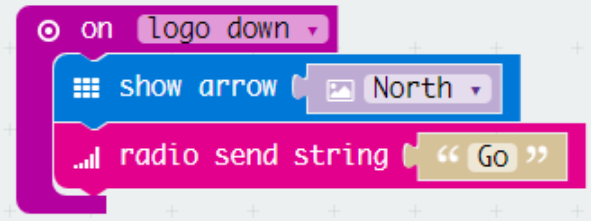
（3） When tilting to the downside (the side without LOGO), Micro:bit displays an arrow. And Max:bot goes toward the direction the arrow indicates.



Please note: the “show arrow” function cannot be found directly in the “Basic”. We should first click the “Basic”, there will then appears the “More” under the “Basic”. Click the “More”, and you will find the “show arrow” function. Place the “show arrow” function into the “on logo down”. Which means: at the time Micro:bit is tilting to the downside, it displays an arrow pointing to the north.



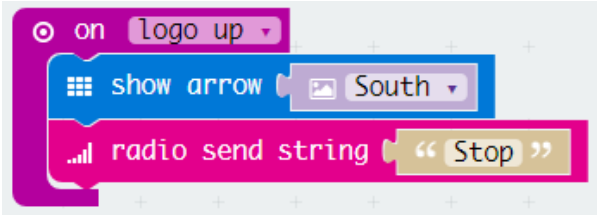
We need to use the “radio send string” function to send out the “Go” over radio. Place the “radio send string” into the “on logo down”.



（4） In addition, we have to program the other three statuses:

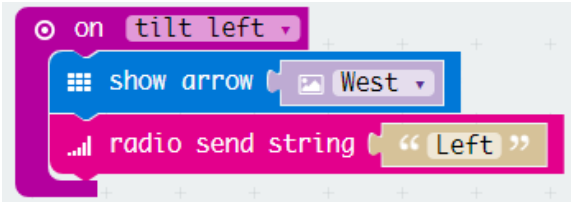
Stop

Place both the “show arrow” and the “radio send string” functions into the “on logo up”. Adjust the arrow pointing to the south and radio send the “Stop”.



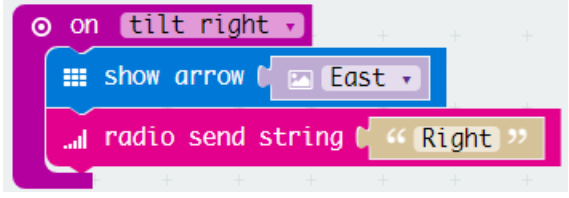
Turn to the left

Place both the “show arrow” and the “radio send string” functions into the “on tilt left”. Adjust the arrow pointing to the west and radio send the “Left”.



Turn to the right

Place both the “show arrow” and the “radio send string” functions into the “on tilt right”. Adjust the arrow pointing to the east and radio send the “Right”.



（5） By putting all above listed functions together, we will have the final program as below. Download it to the transmitting end.

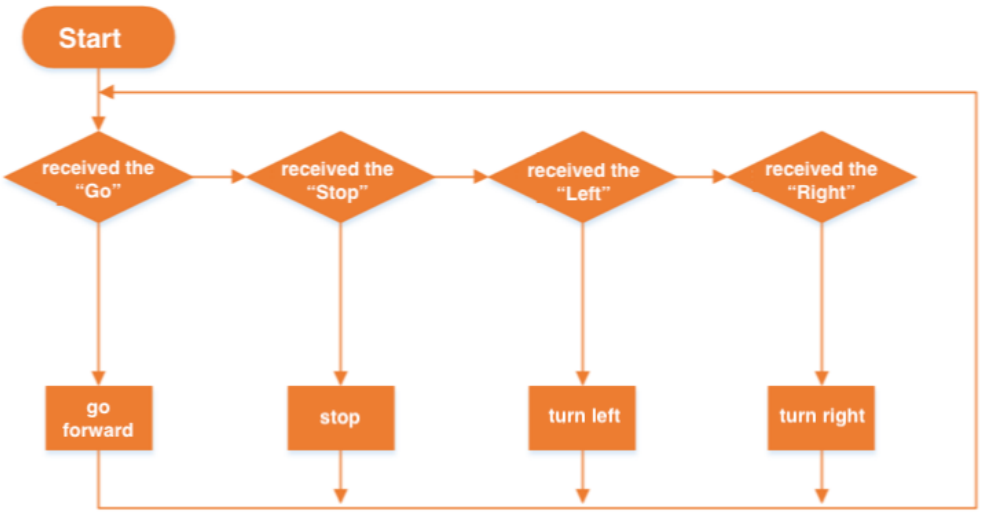


The receiving end

（1） Start a new project and name it as “sensing racing the receiving end”

（2） We need to program corresponding motions of Max:bot according to the massages received by the receiving end.

We can program the receiving end according to below diagram.



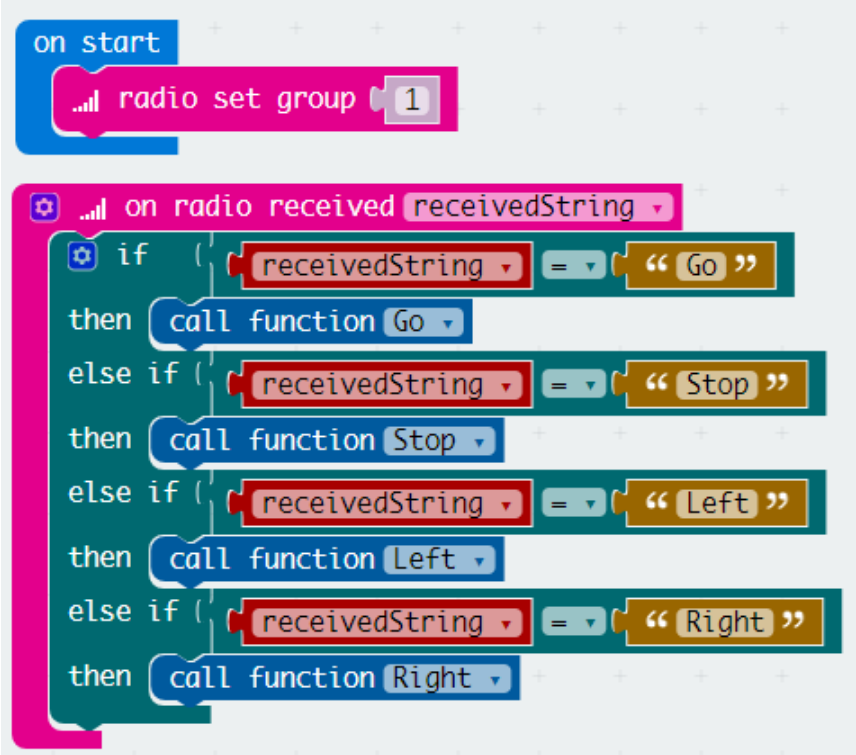
（3） Here we need first to use the “servo write” to control the speed of the wheels and realize the motions like: go forward, stop, turn left and turn right.

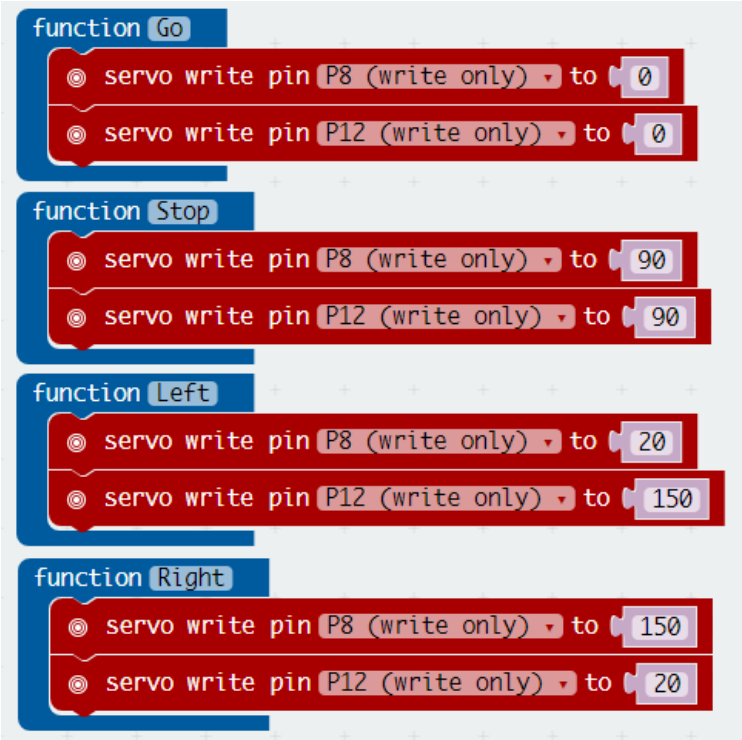
Then, the “receivedstring” function and the operator “=” are used to judge which words is received.

The multiple “if else” is applied to determine when should Max:bot to go forward, stop, turn left and turn right.

Combine all the function blocks listed below together, the final program goes as following.

Download it to Max:bot. Then we can invite our friends to play the racing game.



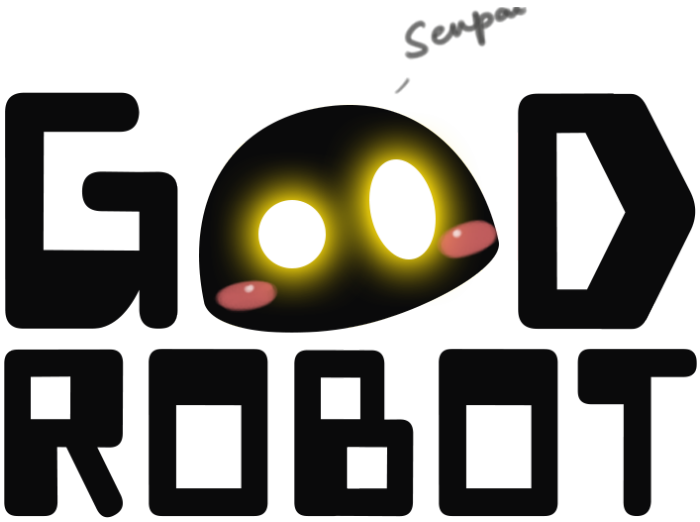


Exercise

When playing a motion sensing racing, the larger angle your hands turn, the larger angle the racing car on the screen turns. How to achieve this via Max:bot? Try to design your own diagram in the blank below.

Chapter 7: The sound and light show

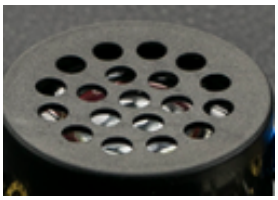
Max:bot, our reliable companion, has led us to escape the maze and win the motion sensing racing. You may still have a sense of loss because Max:bot never speak to you. But actually, he can even play the sound and light show! Let's explore more!



Goals:

1. How to use the LED strip module?
2. How to use the digital speaker module?
3. How to program a piece of music?
4. How to make a "Fluid LED Strip"?

Digital modules

Digital module(s)	Image(s)	Function(s)
A LED strip module		There are many leds lie on the strip. Through programming, different leds can be set with different colors, brightness and flash frequencies.
The digital speaker module		The digital speaker module is an output module which is used to play music.

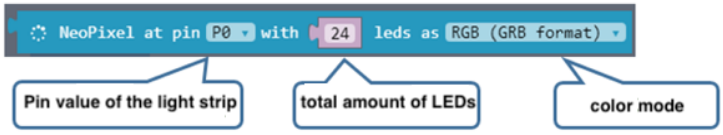
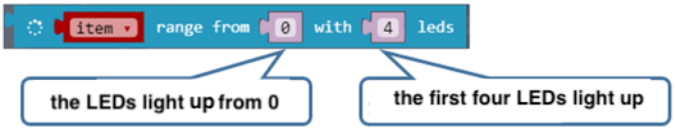
7.1 The rainbow light strip

On some special occasions, like your birthday party, various of festivals, etc., all of us would like to decorate our houses with colorful light strips.



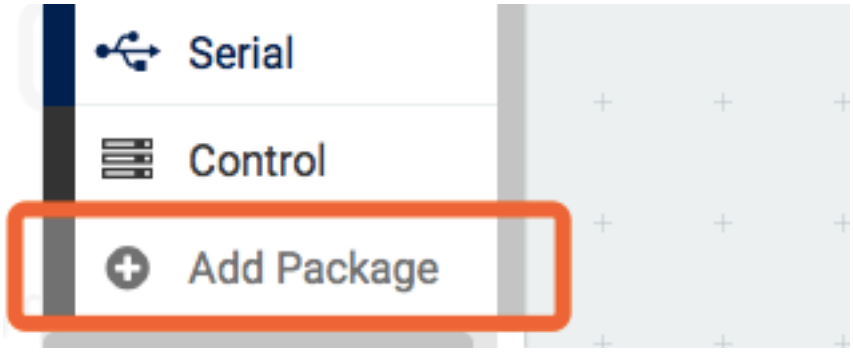
In this part, we will use the LED strip to decorate Max:bot.

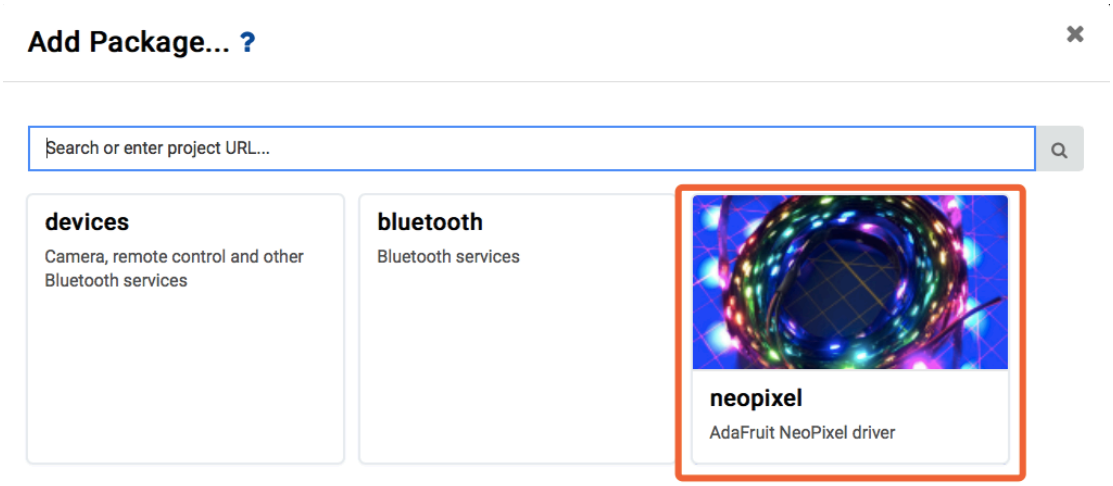
Key information

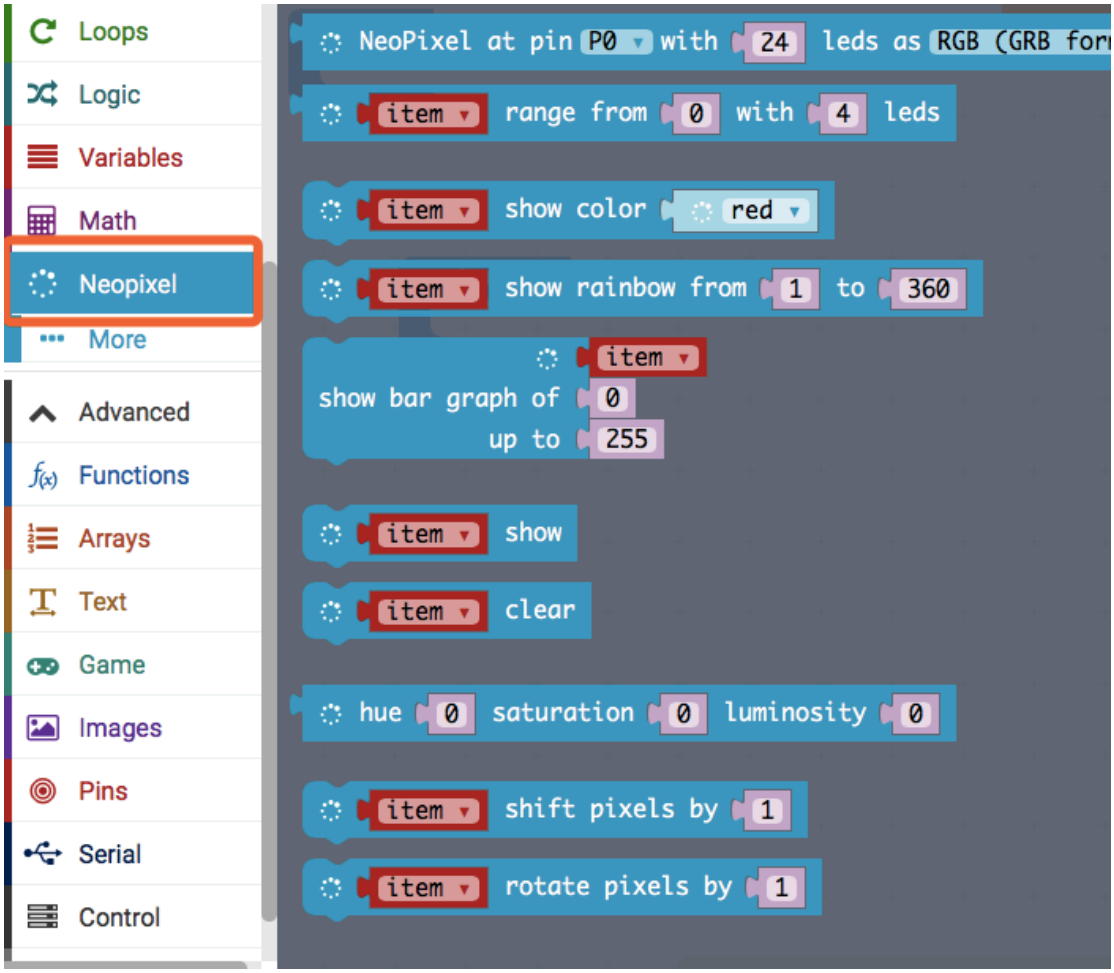
Function block(s)	Image(s)	Function(s)
"Neopixel"		The "Neopixel" function can be used to setup how many leds are included; which pin to be connected to and what colors will display.
"range leds"		The "range leds" is used to configure how many leds (out of the total number of leds that physically included in the strip) will be under control of the program
"show rainbow"		The "show rainbow" function is used to setup light effect.

Program

（1） The function block for controlling the RGB light strip is not listed in the tool box when we start a new project. To add this function, we need to manually import the "Neopixel" from the bottom part of the page. Click "Add Package" and to open the extension function manager then select "Neopixel". The Neopixel will show up in the function list.


Serial
Control
Add Package


Add Package... ?
Search or enter project URL...
devices
Camera, remote control and other Bluetooth services
bluetooth
Bluetooth services
neopixel
AdaFruit NeoPixel driver



（2） Program the rainbow light effect

To start using the "Neopixel" function, we will first setup how many leds are included in the light strip and which pin it connects to.

Then, use the "show rainbow" function to setup light effect. This effect will only apply to leds that we defined in our last step. The parameter inside the function block is used to set the gradient range of color, 1 represents red whereas 360 represents blue, the closer these two numbers are, the smaller the gradient changes:



Among which, the  from the  function bock is replaced by the "Nexpixel" function.

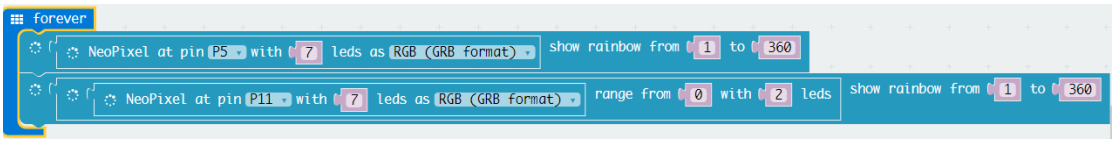
（3） Download the program to Max:bot.



The LED strip now can display the rainbow light effect.

Exercise

Connect two light strips separately to Pin5 and P11. Program the light strips as below.



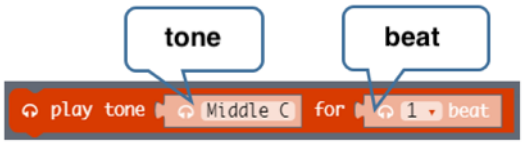
Can you tell us the differences between the two light strips? Record them and share with your friends!

7.2 Put the music on!

Listen to sweet music, and your foolish fears and pretty jealousies will pass away. What if we add a sound effect while playing games? For instance, a piece of lively song is played to celebrate your victory.



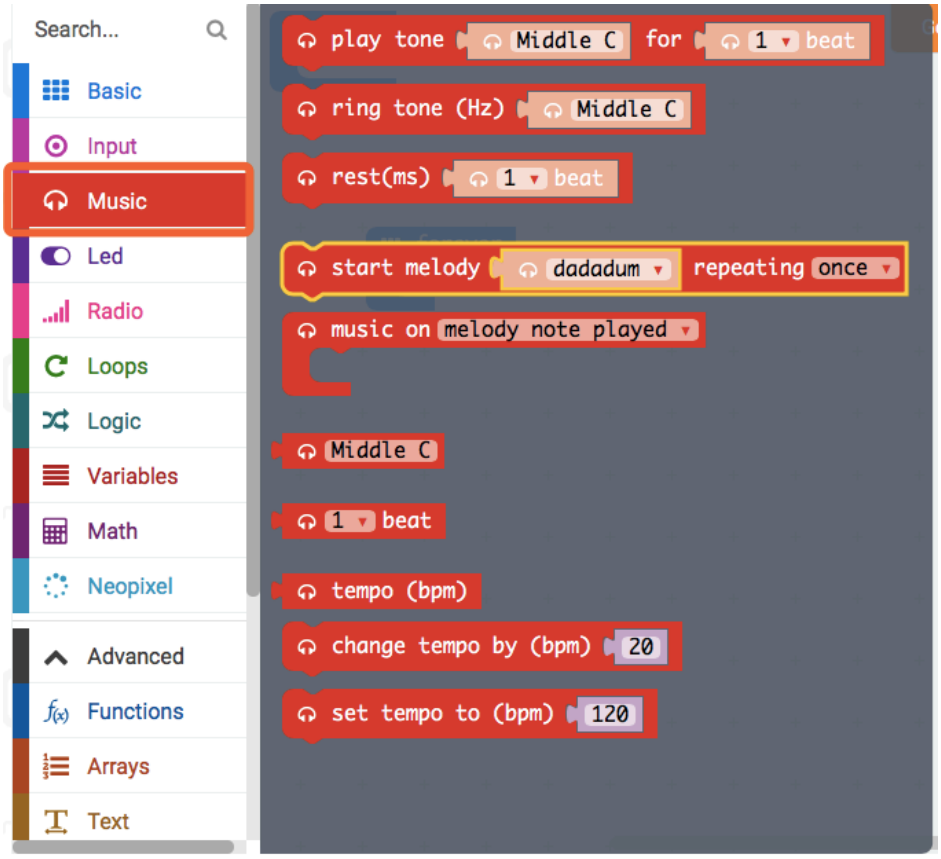
Key information

Function block(s)	Image(s)	Function(s)
"start melody"		The "start melody" function is used to select songs that listed in Makeode.
"play tone"		The "play tone" function is used to set the tone and beats.

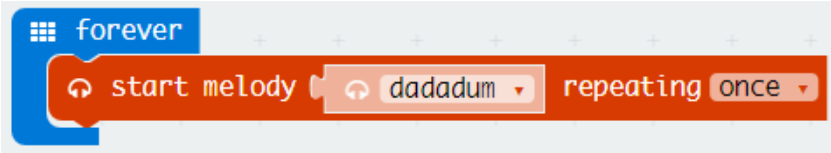
Program

（1） Start a new project and name it as "put the music on"

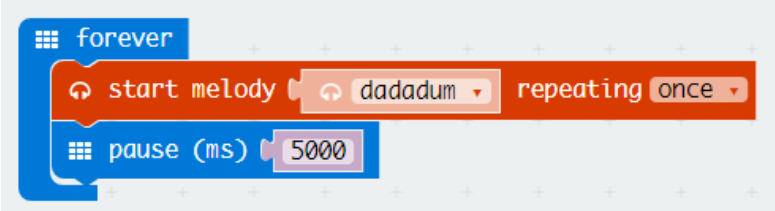
（2） Forever 中。The "start melody" function lies in the "Music" of the function area. Find it and place into the "forever" loop.



（3） There are many melodies can be selected from the drop-dpwn list of the “start melofy”. Here we choose the “dadadum” to repeat “once”.



（4） In this part, we will use the “pause” function to provide enough time for Max:bot to finish his song – the “dadadum”. Combine all the function blocks listed above together, our final program goes as following:



（5） Download it to Max:bot and you will hear his first song – the “dadadum”.

Exercise
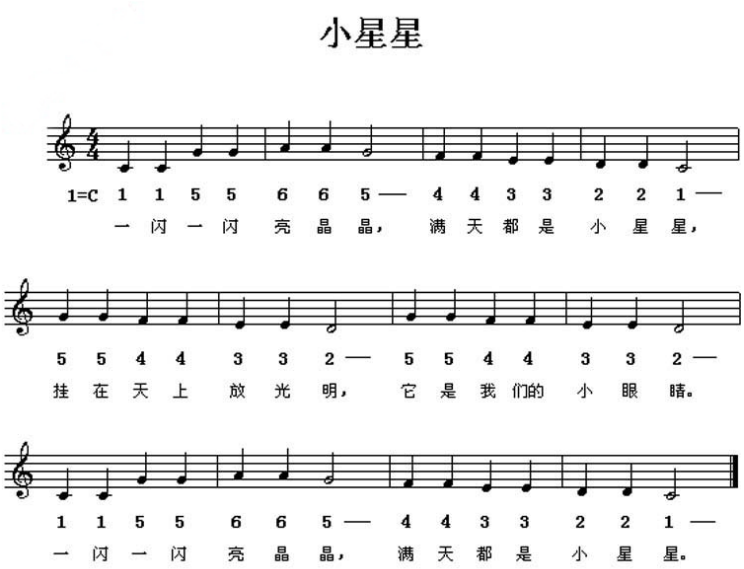
What if you want Max:bot to sing a song composed by you? He can make your dream come true. We will use the "play tone" function to set the tone and beats.



As has shown in the diagram below that each tone corresponds to a certain note.

Middle C	Middle D	Middle E	Middle F	Middle G	Middle A	Middle B
1(Do)	2(Re)	3(Mi)	4(Fa)	5(So)	6(La)	7(Ti)

By applying the "play tone" function, we can work together to edit the "Twinkle, Twinkle Little Star". Come and join us!

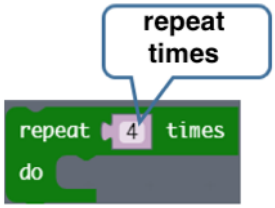
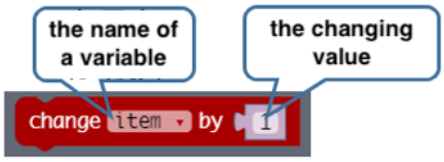


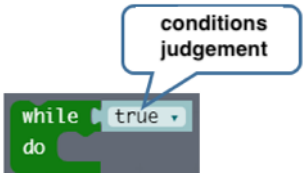
7.3 The sound and light show!

Is Christmas your favorite festival? Do you like to decorate the Christmas tree? In this chapter, we will work hand in hand to make a special Christmas tree robot who can play a sound and light show for your family and guests!



Key information

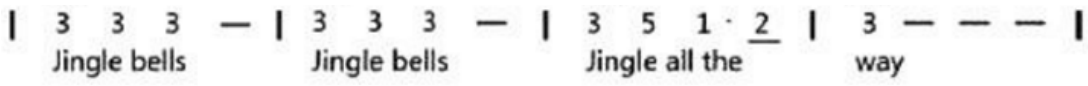
Function block(s)	Image(s)	Function(s)
"repeat"		We can first place a function into the "repeat" and then use it to determine how many times the function will be repeated.
"change"		The "change" is used to determine how a variable changes. For example: we name the variable as "item" and set the changing value as "1". Which means every time the "change" function is executed, the variable will add 1. As we have learnt before that the functions in the "forever" loop will be executed repeatedly. So the first time the "change" function is executed, the

		variable is “item”, while for the second time it is repeated, the “variable” will then become “item+1”. The variable will continue to change as below: Item → item+1 → (item+1)+1
”while do”		The “while do” loop is used to judge whether the condition to the right of the “while” is met or not. If yes, the function to the right of “do” will be executed.
”clear”		The “clear” function is used to turn off the light effect.
”show”		The “show” function is used to turn on the light effect.

Program

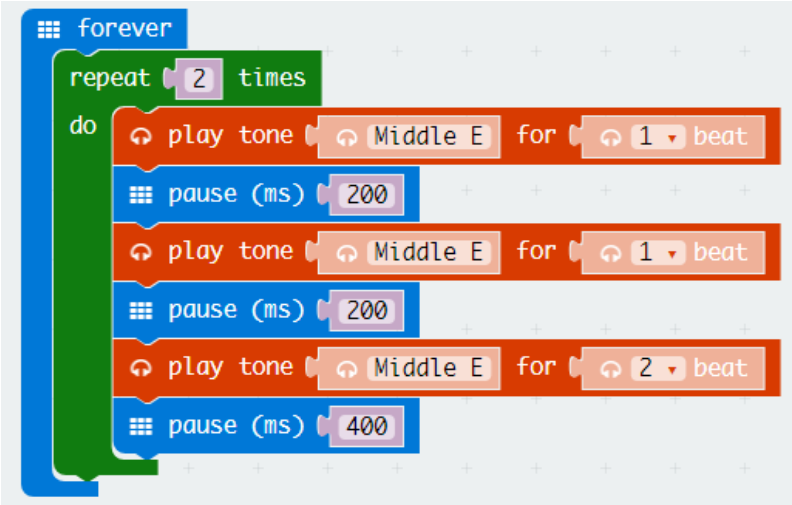
（1） Start a new project and name it as “the sound and light show”

（2） Use the “play tone” function to edit the “Jingle Bells”.



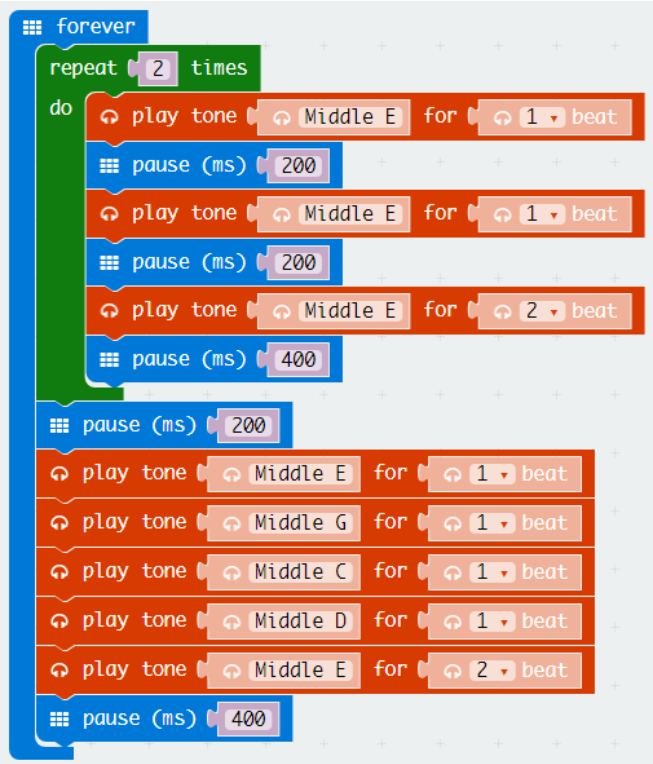
The first two phrases of “Jingle Bells” keep the same notes and beats. So we can apply the “repeat” function to do the job.

As we have learnt before that Middle E equals the note 3(Mi). So we can program the first two phrases as below:



You can add the “pause” function to make Max:bot more professional.

According to the phrases of "Jingle Bells", the complete music goes as following:



（3） Program the corresponding light effect

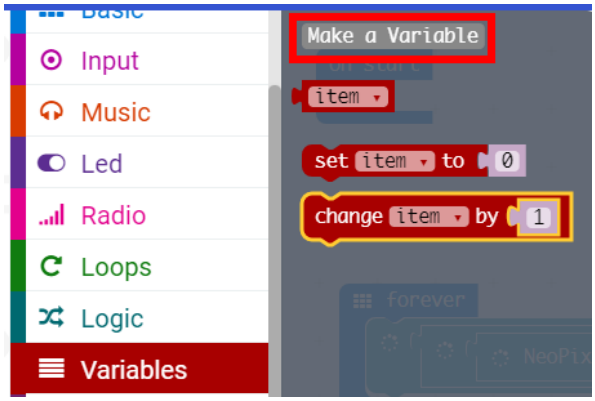
In this step, we will learn how to turn the two light strips into two “Fluid LED Strips”.

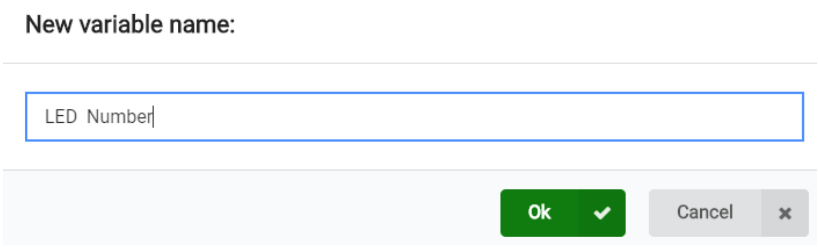
Before getting started, we should get to know what the Fluid LED Strip is. It means the leds on the strip will be lighted up one by one until all the leds are turned on.



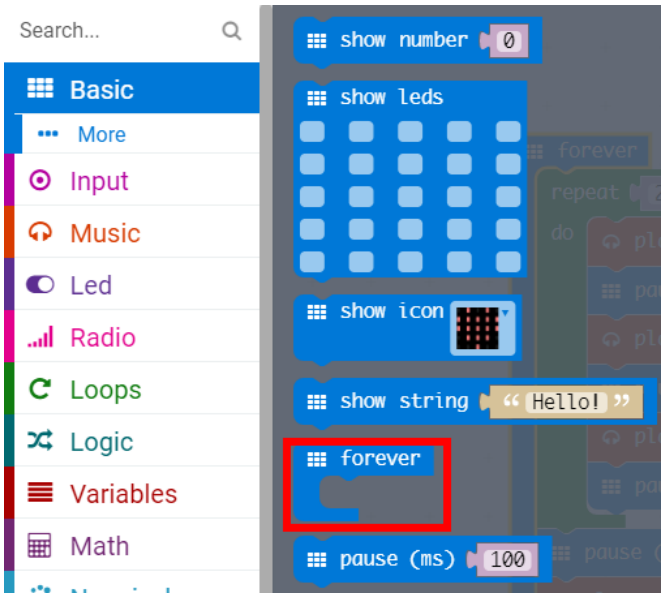
So the number of LED(s) to be lit should be a variable. Thus we need to create a variable in the first place.

Find the "Make a Variable" from the "Variables" in the function area. Click and name it as "LED Number". Click Ok.





Place another "forever" loop from the "Basic" into the program area.

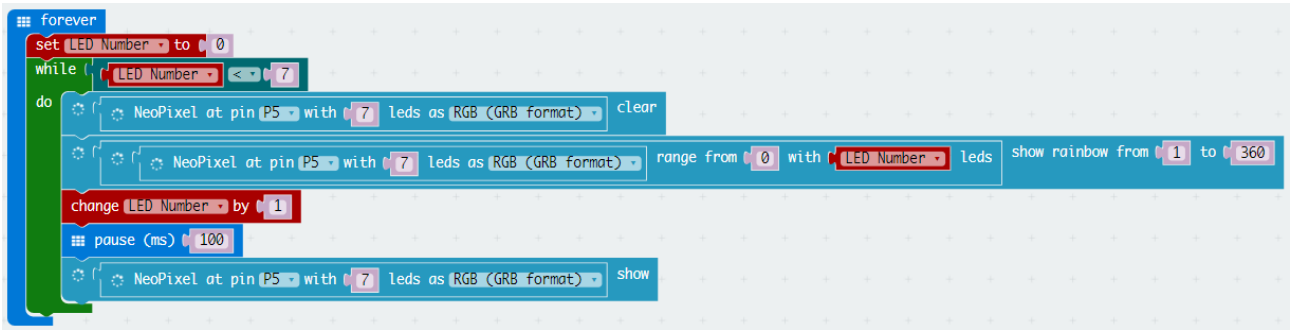


The pin P5 connected the left strip comes with 7 leds. To add a rainbow light effect to the strip, we will need to set the color range from 1-360 (red gradually changes to blue). The created variable "LED Numbers" should be placed into the "range from" function.



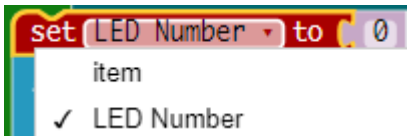
The variable "LED Number" is used to determine how many leds on the left strip to be lit.

Since there are 7 leds on the strip, the number of variable "LED Number" should be limited within 7. The initial value is 0.



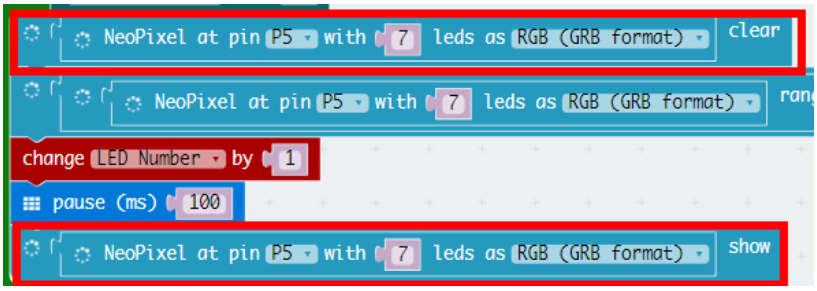
When programming, we should keep below notices in mind:

a. When using the “set to” function, we should click the drop-down menu(the tiny triangle) to select the variable “LED Number”.



b. If you are programming the left light strip, set the Pin as P5. Whereas for the right one, Pin 11.

c. To realize the fluid light effect, we should place both the “clear” and “show” functions into the “forever” loop.

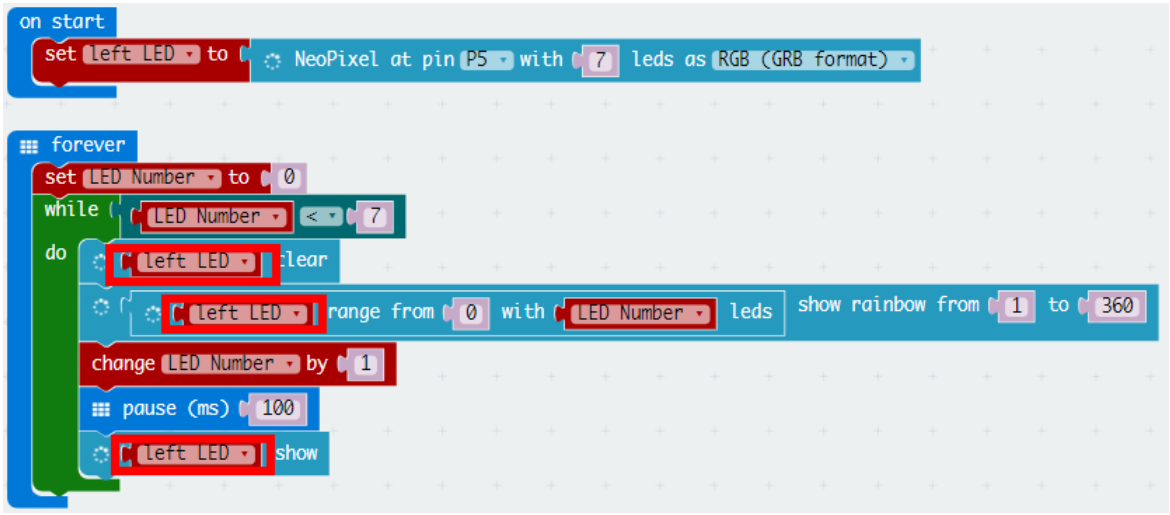


（4） As we can see from above that the “Neopixel” function which controls the light strip is used three time in the program. To simplify the major loop, we can set a variable “left LED” to replace the long “Neopixel” function.

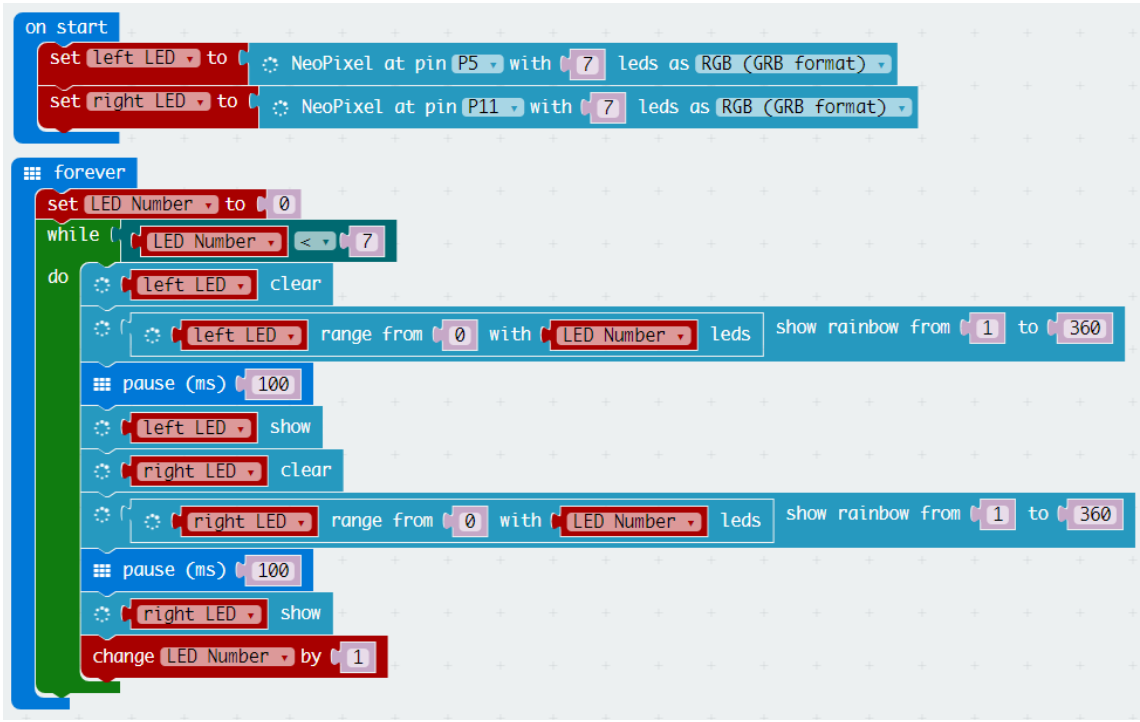
We need first to create a variable and name it as “left LED”.

Then place the long “Neopixel” to the right of the “set to” function. Which means that the whole “Neopixel” function will be replaced by the variable “left LED”. The “set to” function lies in the “Variables”.

Place the programmed functions into the “on start”.

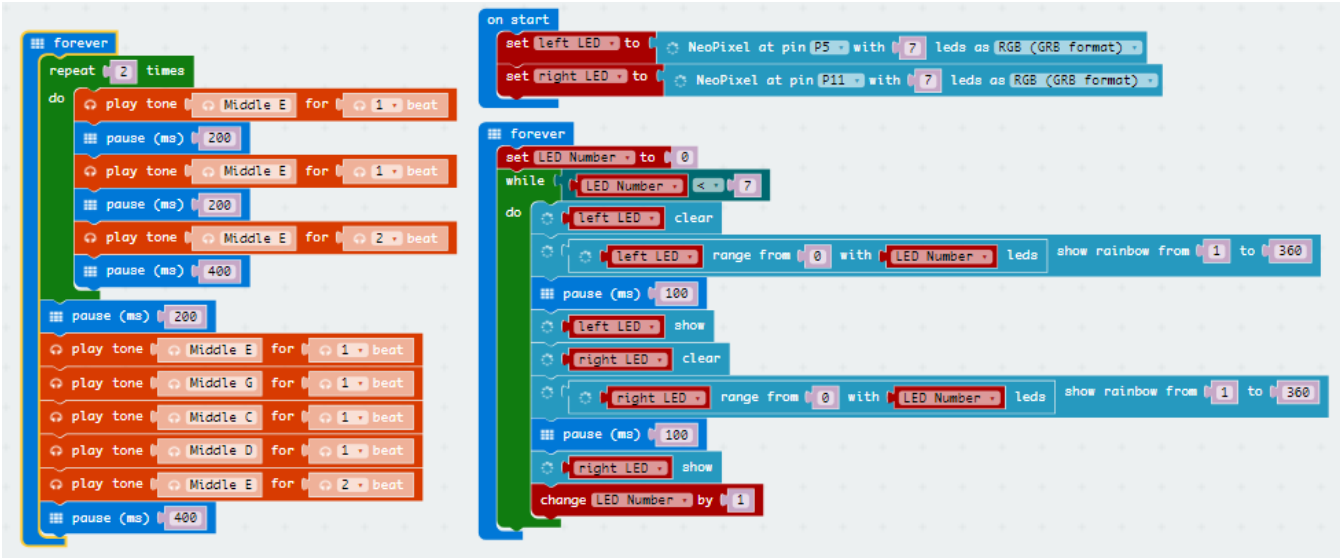


Thus, we can apply the same procedure to program the right light strip. By putting together all the functions listed above, the final program of fluid light effect goes as following:



（5） Above we have learnt how to play the “Jingle Bells” and how to display the fluid light effect. To enable Max:bot to perform the sound and light show, we have to combine all the functions listed in both programs together. The final program goes as below:

Download the final program to Max:bot. Now he is able to play the customized sound and light show for your incoming Christmas!



Exercise

Since your Christmas robot is able to play a sound and light show, why not asking him to go out and send presents for you?

In this starter guide, we just introduced some basic functions of Max:bot. Actually, it possesses magical abilities that are never exactly explored. But what we believe indeed is that, through your creative minds, all those mysteries will be eventually unfolded to all of us. Please visit the DFRobot blog and share your ideas with like-minded people!

Welcome to the DFRobot blog!

www.dfrobot.com/blog