A STARTER GUIDE

OF BOSON KIT

FOR MICRO:BIT

DFROBOT

DRIVE THE FUTURE

Contents

Remember to check out the DFRobot blog for more tutorials, projects and latest trends. The link can be found on the top left corner of the page.

Chapter 1: MakeCode and micro:bit

An Introduction to MakeCode

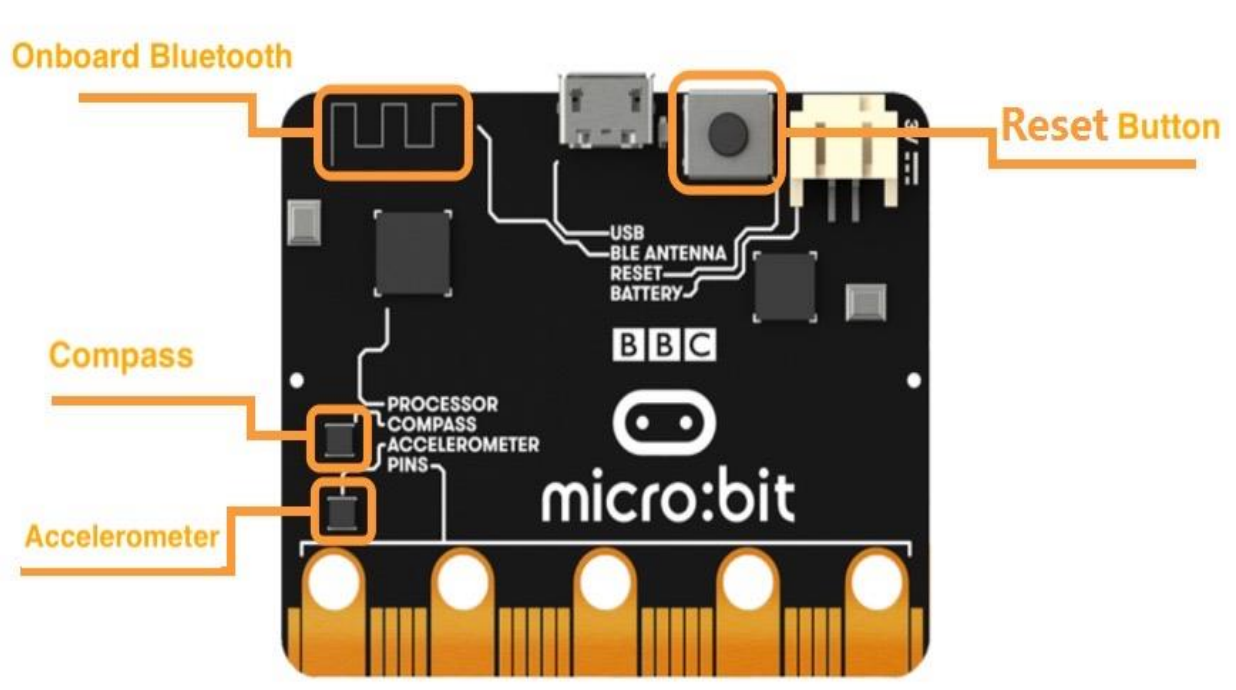
MakeCode for micro: bit is one of the most widely used graphical programming environment from the micro: bit website. It is an open source project developed by Microsoft.

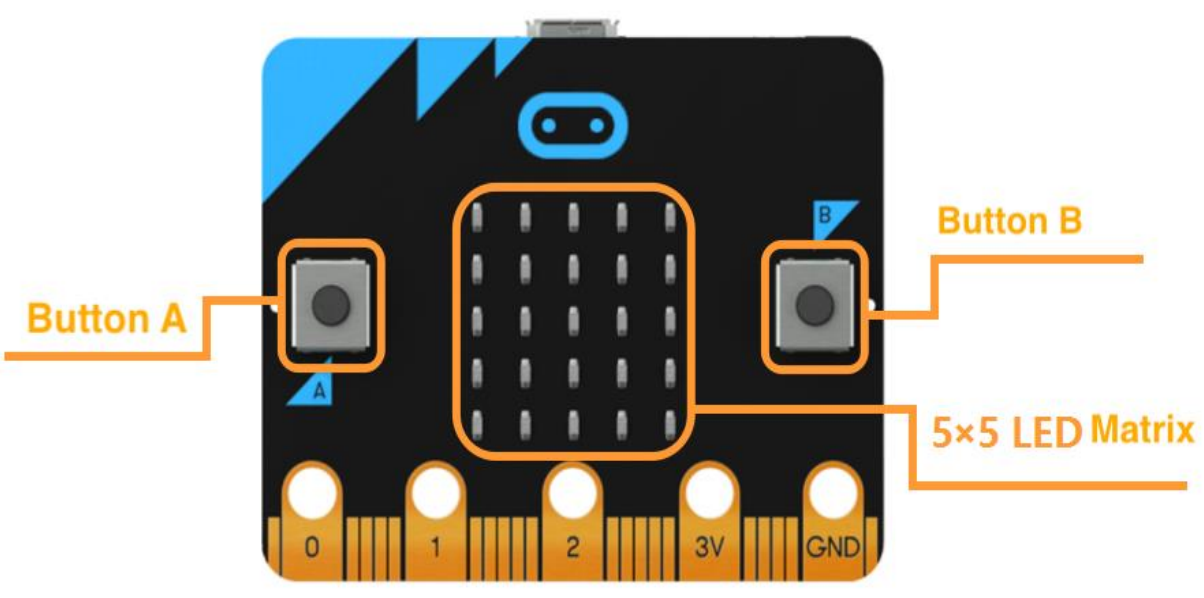
A Brief Introduction to micro: bit

micro:bit is a 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.

micro:bit comes with a variety of on-board modules, including a 5x5 LED matrix (also supports light detection), 2 programmable buttons, motion detector, Compass and Bluetooth® Smart module. Additionally, you may attach more modules such as a servo motor, RGB LED lights through 5 I/O rings or 20 edge connectors.

The micro:bit can be used to realize many cool ideas. Everything you can imagine, a robot, electric instrument, or even a home automation system. The possibilities are endless! The micro: bit holds a host of innovative features such as 25 red LEDs to display messages and two programmable buttons to control game or control music. It can detect motion, recognize gesture, and be interconnected with other devices or the Internet via Bluetooth connection.





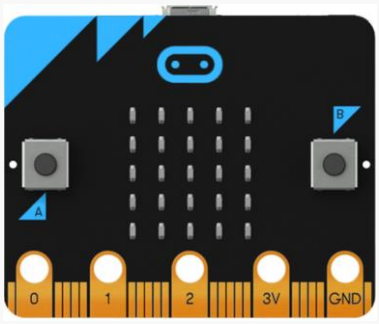
The micro: bit is equipped with light and temperature sensors and other common sensing devices, which means that it itself can also produce a lot of common smart products used in our daily lives.

How to Use micro:bit

If you are new to micro: bit, you can start with the online programming platform- The MakeCode Editor, to learn about how to program the micro: bit.

Before getting started, please make sure that the following items are ready by your hand. In addition, you will also need a computer running operating systems such as Windows 10 / Mac OS / Linux and with Internet connectivity.

COMPONENTS LIST:



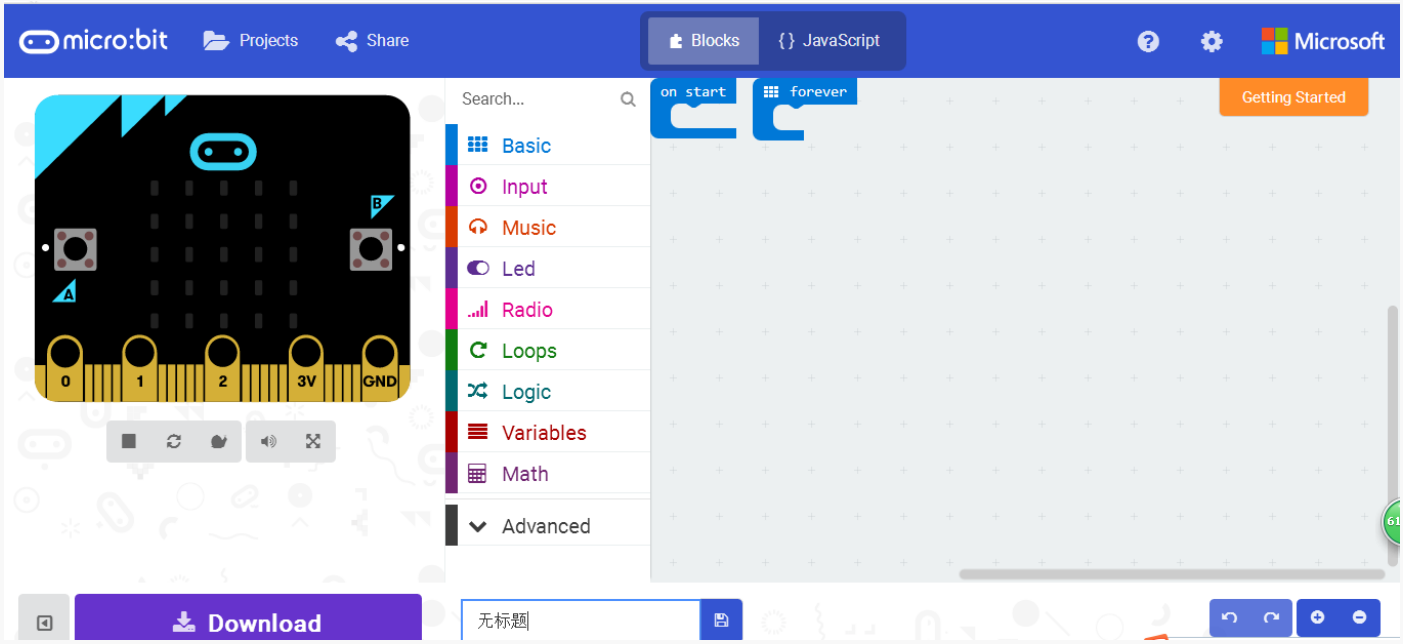
Micro:bit Mainboard

USB cable

The following steps are based on Windows 10 OS. It can be used as a reference for other operating systems.

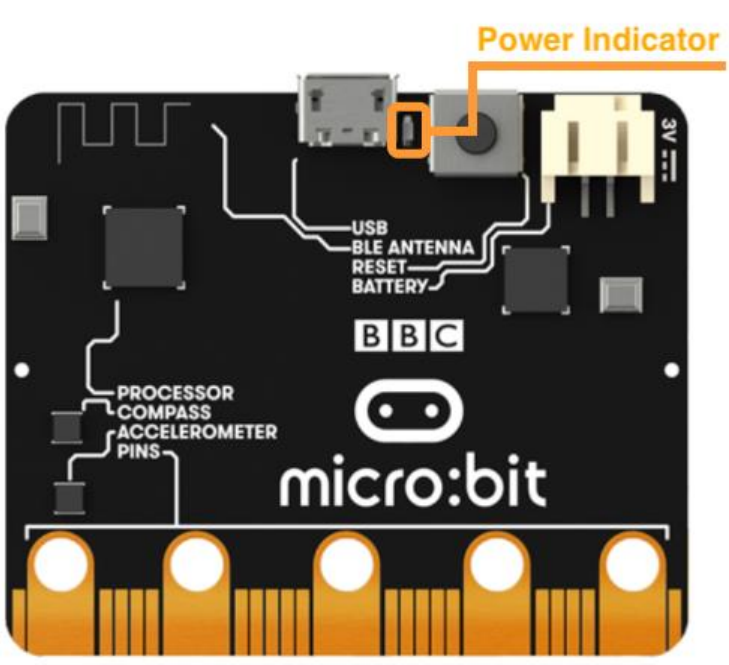
STEP 1: Open MakeCode

Visit the MakeCode page from the following link: https://makecode.microbit.org/

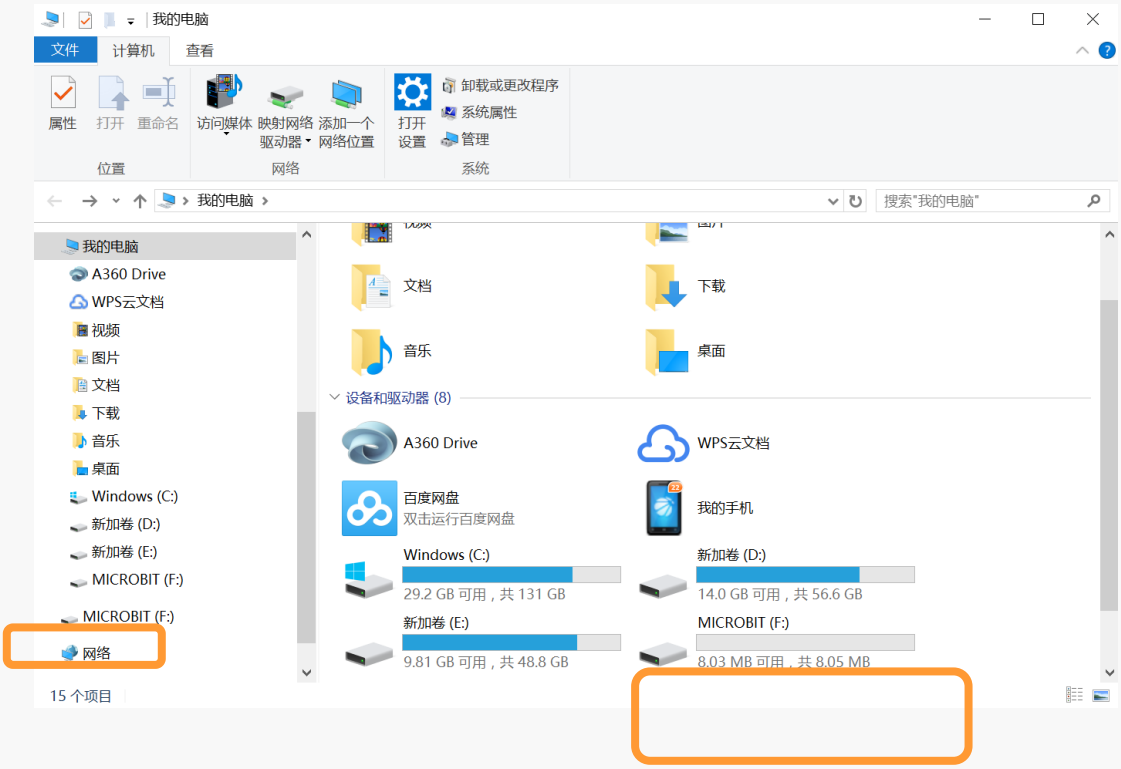


STEP 2: Connect micro: bit to computer

Micro: bit connects to computer via the USB cable. The power indicator on the back side of the micro: bit will light up when connected.



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”.



STEP 3: Start a New Project

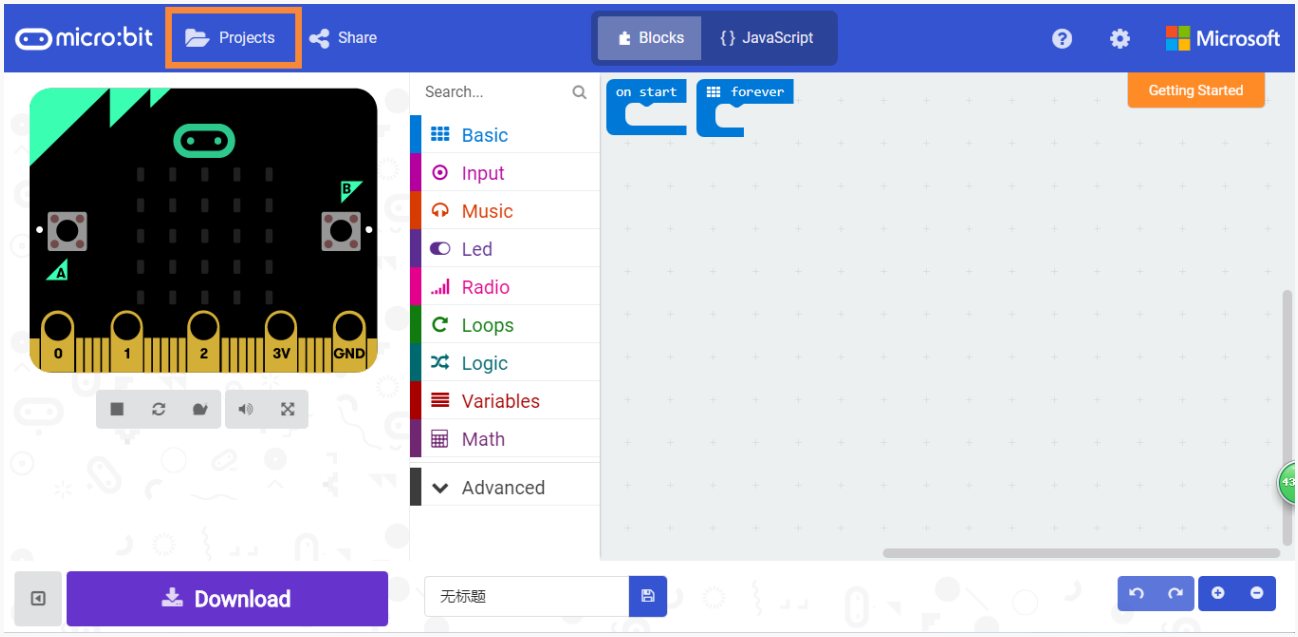
Before starting a new project, we will need to first get familiar with the programming interface.

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.

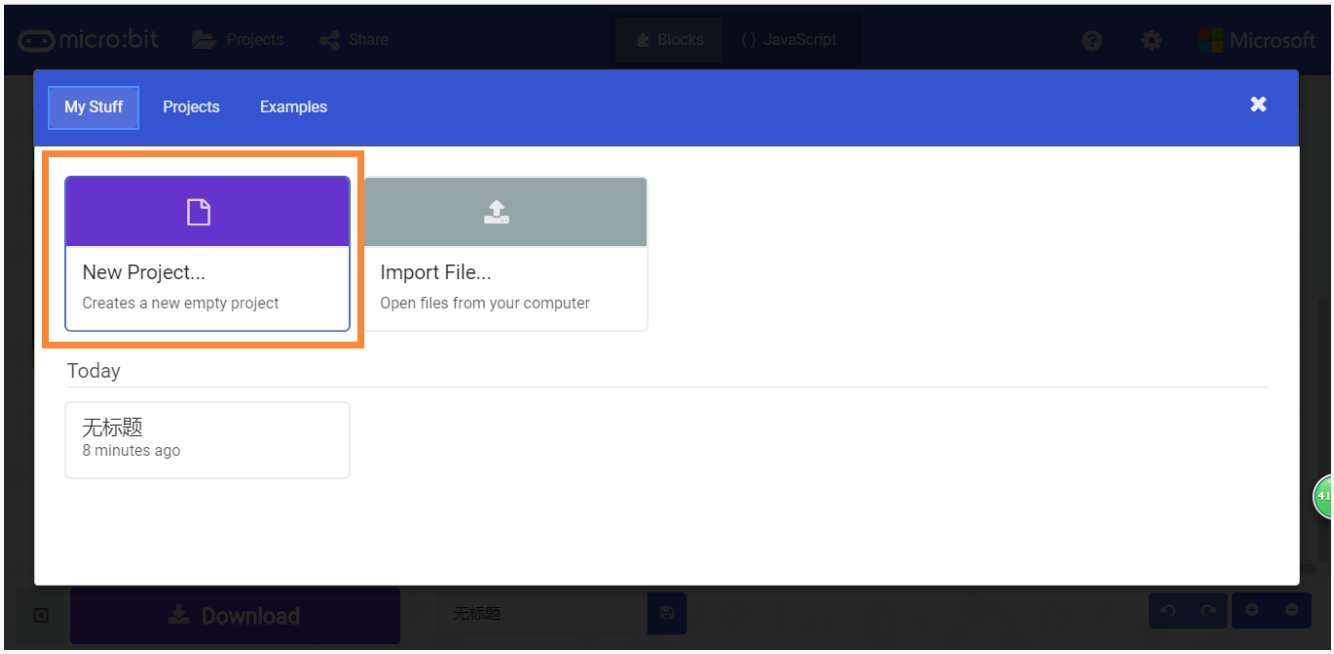
Function area: Where you can find all the function blocks, including input, output, loop, logic, etc..

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

Click "Project" at the top of the simulation window.

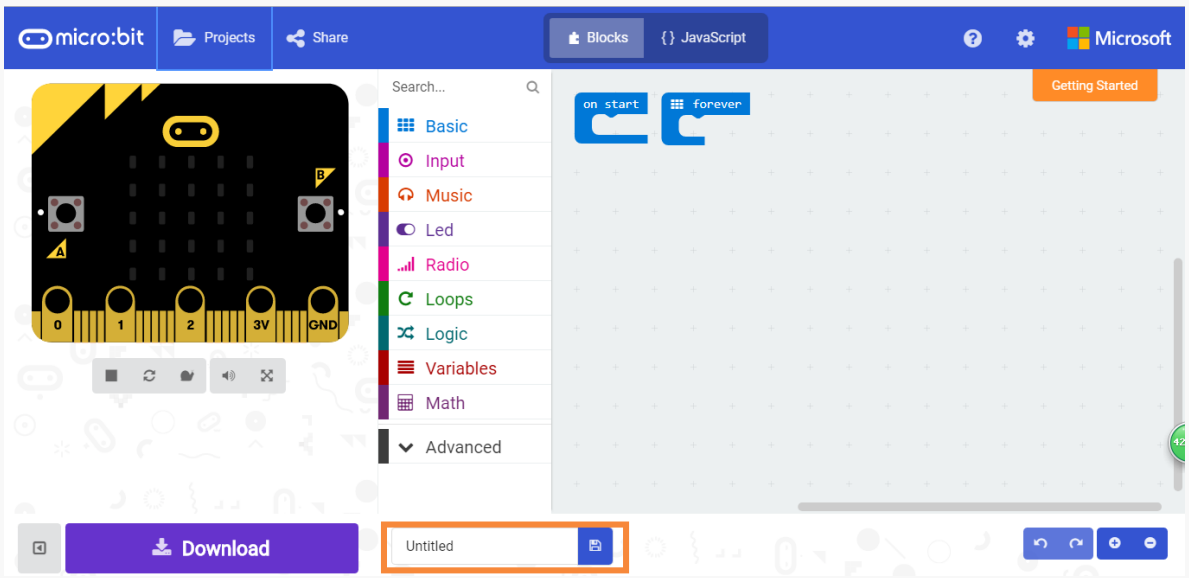


Then, click "New Project".

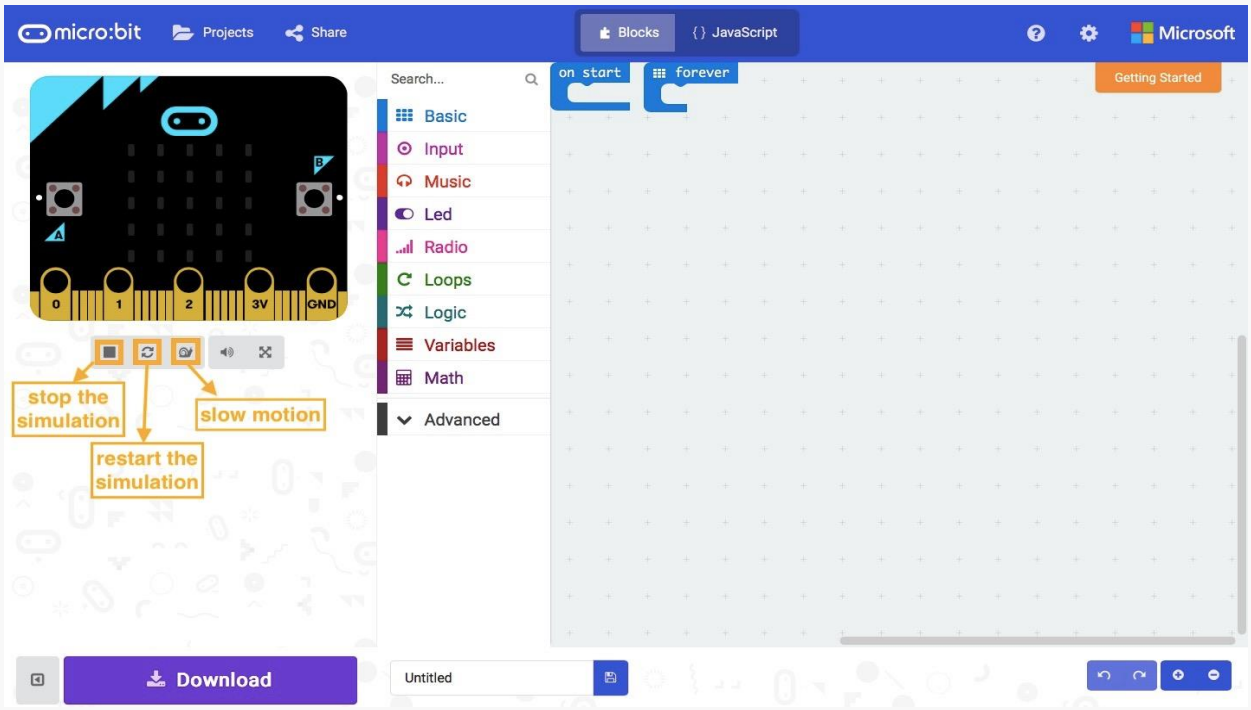


STEP 4: Download the program and upload to micro: bit

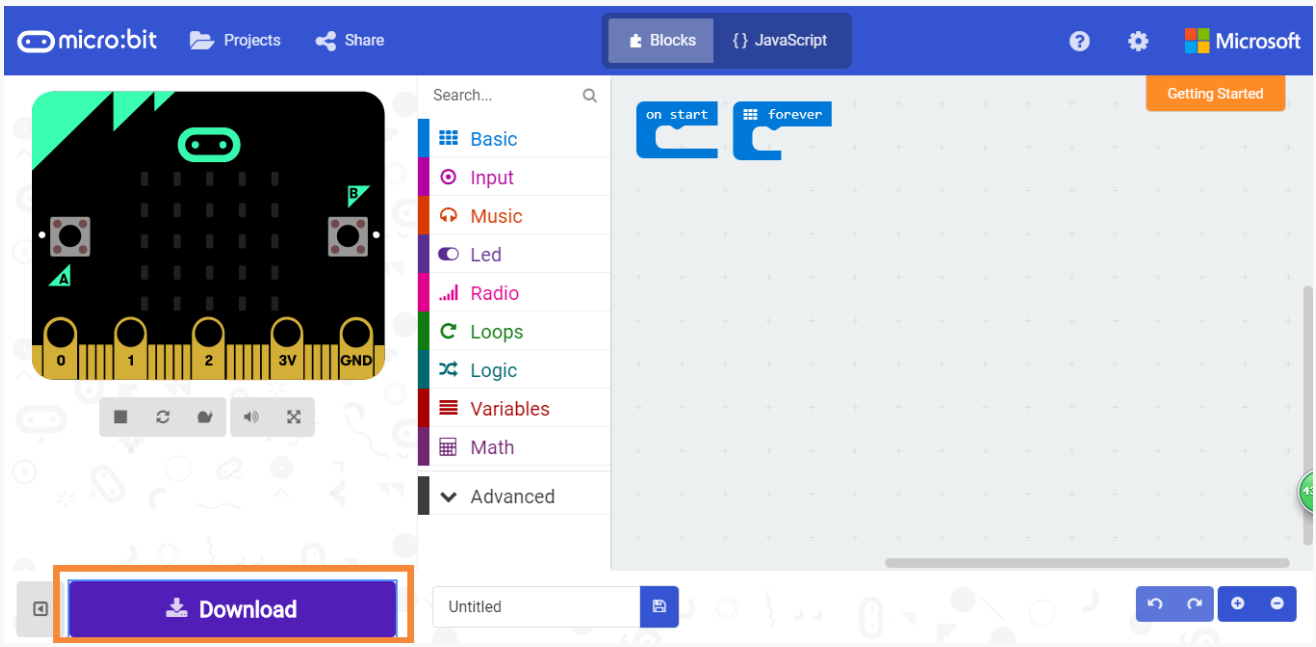
When we have finished our program, we can now download it from the website and uploaded to micro: bit. We can also rename the project and save it in the browser. The project will stay in "my stuff ".

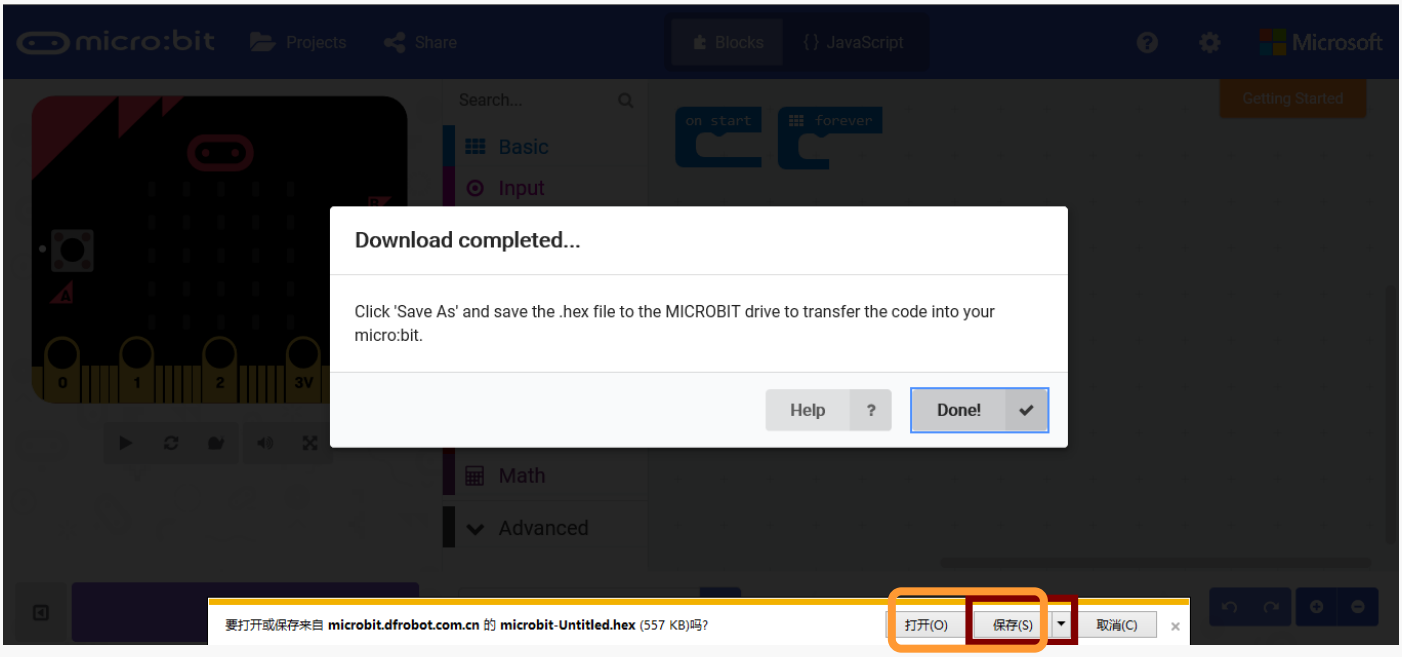


Before downloading, we can check the simulated results in the simulation window. The buttons in the lower part of the simulation window can be used to control the analog micro:bit.

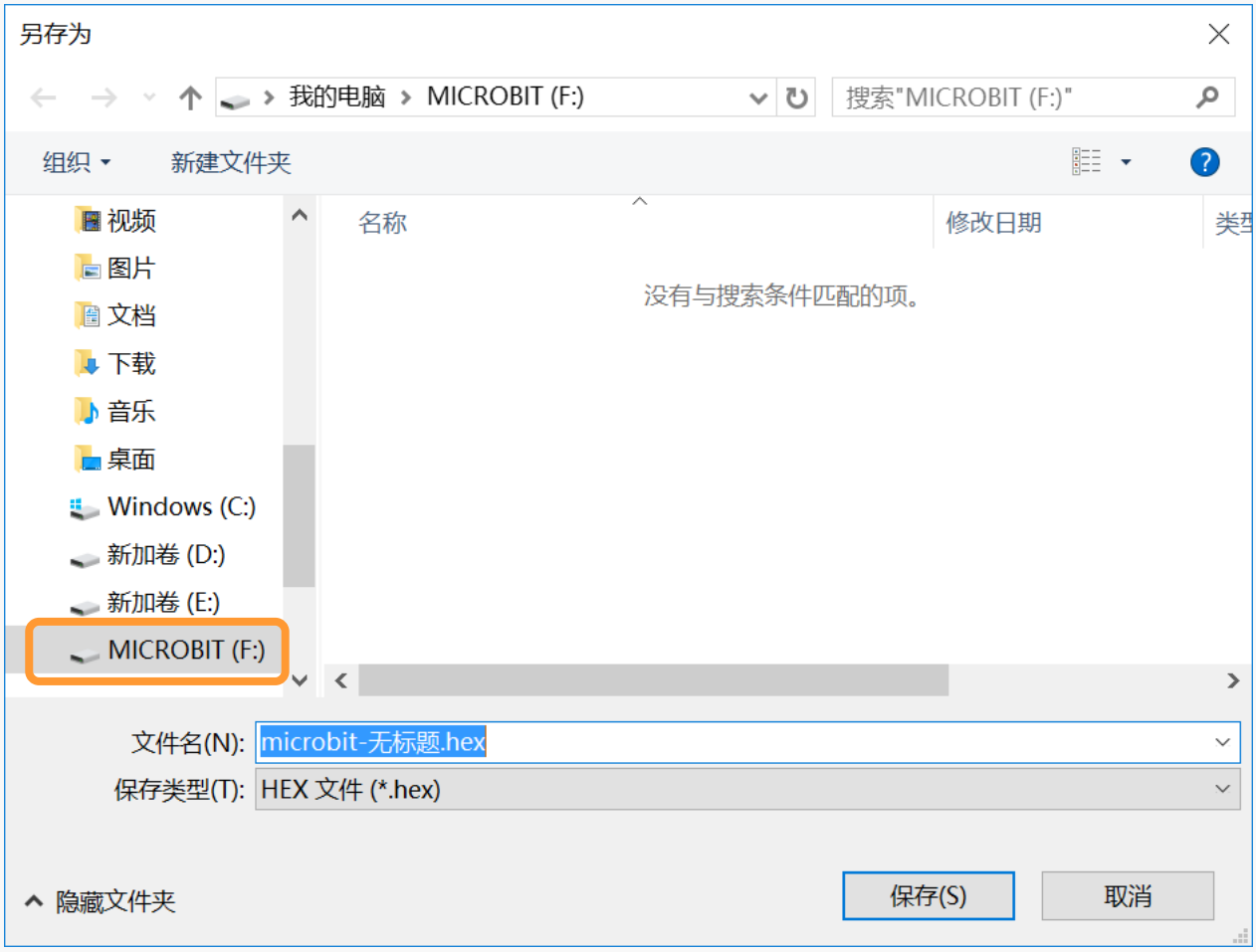


Click "Download" in the lower part of the simulation window and select "Save As" in the pop-up dialog box.





Choose to save the ".hex" file to "MICROBIT Disk" and click "Save".



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.

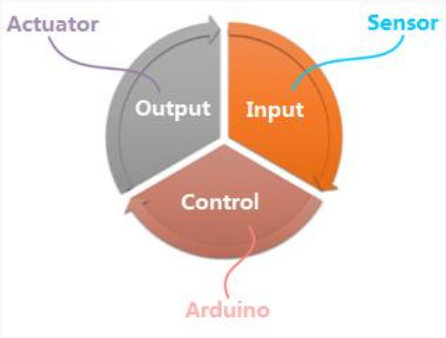
Now you are all good for basic setups. Remember to visit our blog to check out more micro: bit projects. Please also leave a comment if there you have question or new idea to share.

https://www.dfrobot.com/index.php?route=DFblog/blogs

Chapter 2： What Makes a Machine “Come to Life”

Interactive devices

In the next few chapters, we'll build interactive projects such as a button controlled single LED, a simple colorful LED animation, or even make our own music. These devices can be categorized as "interactive devices." In order to help build a better understanding of their mechanisms, we need to first understand how they are composed.



The simplest interaction device consists of 3 parts:

- Input unit for command acceptance or data collection
- Control unit for data or signal processing
- Output unit for sending data or executing actions

You must be confused by these conceptions and eager to know how they actually function. Let's taking our own body as an example, we collect information in the form of light, sound, taste, and power through our eyes, ears, nose, and skin. These information enter our brains and determine what response to take. Ultimately, we take physical actions based on these information to change the material environment. Specifically, your friend, for example, says "hello" to you, and you respond "hello". Here, your ears act as input units, your brain acts as a control unit, and your mouth plays the role of output unit.

Similarly, while building projects by using the micro: bit, we will use a variety of sensors as the input units, the micro:bit as the control unit, and the actuator as the output unit.

Input Unit —— sensor

Sensors (also called transducers) are physical components that detect the environmental characteristics such as light, temperature, humidity, etc. and convert them into signal data. In this tutorial, sensors such as buttons, sound sensors and temperature sensors will be applied.

Control Unit ——micro: bit

The micro: bit will act as a control unit in this tutorial, using the signal pins to establish connections between input units and output units and processing the data in the calculation processing module.

Output Unit —— actuator

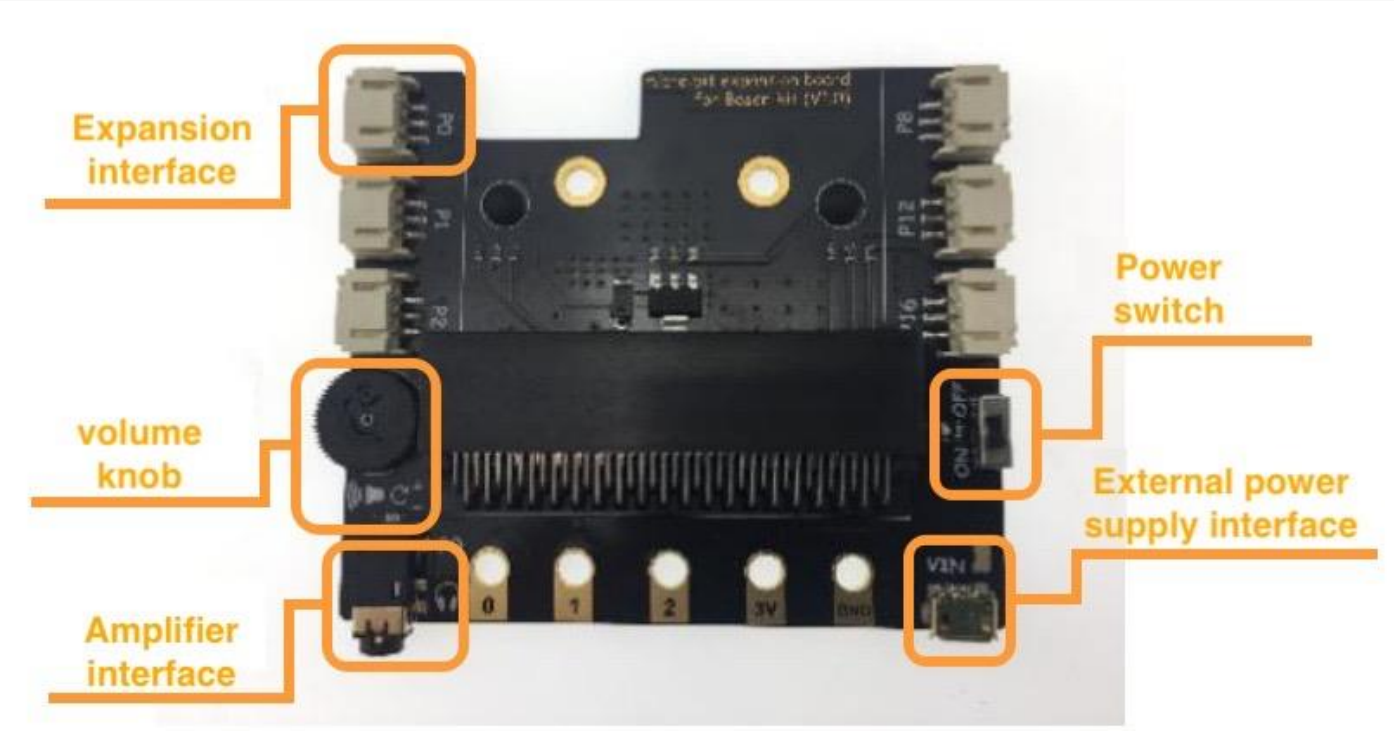
The actuator is the one responsible for moving or controlling a system or a mechanism. It converts electrical energy into motion, sound and light. In this tutorial, the actuators such as LEDs, small fans and servos will be applied.

Relationship between Program and Hardware

The input unit, control unit and output unit mentioned above are physical components. As for human-being, the hardware includes the body, brain and limb. However, what actually controls the behavior of our physical body is our “mind”. The program here is like our mind and what we will explain is how to setup the “mind” of the micro: bit so it will do whatever we ask it to do.

The Boson Expansion Board for micro: bit

The Boson expansion board is to enable micro: bit to connect external modules, including buttons, switches modules, and sensors. It not only simplifies the circuit connection, but also make micro: bit more powerful.



Chapter 3：Get hands on!

You may now have a rough idea about what micro: bit and expansion board are and how MakeCode operates. You must be eager to get a hand on it by yourself. So, let's start the magical journey of Boson for micro: bit!

Project 1: The Mysterious Micro: bit

We will start our first project, a simple emojis panel, to get a bit more familiar with what we have just learned in the previous chapter.

Emojis can be seen on almost every smartphone and computer, they might appear as follow:

love

be cool

I am watching you

In this chapter, we will show you how to design your own emojis by programming the micro:bit.

There are 5 * 5 (25) on-board LEDs on the micro: bit. We can control these LEDs to form different patterns to show the emojis.

Components list

1 × micro:bit

1 × USB cable

Connection

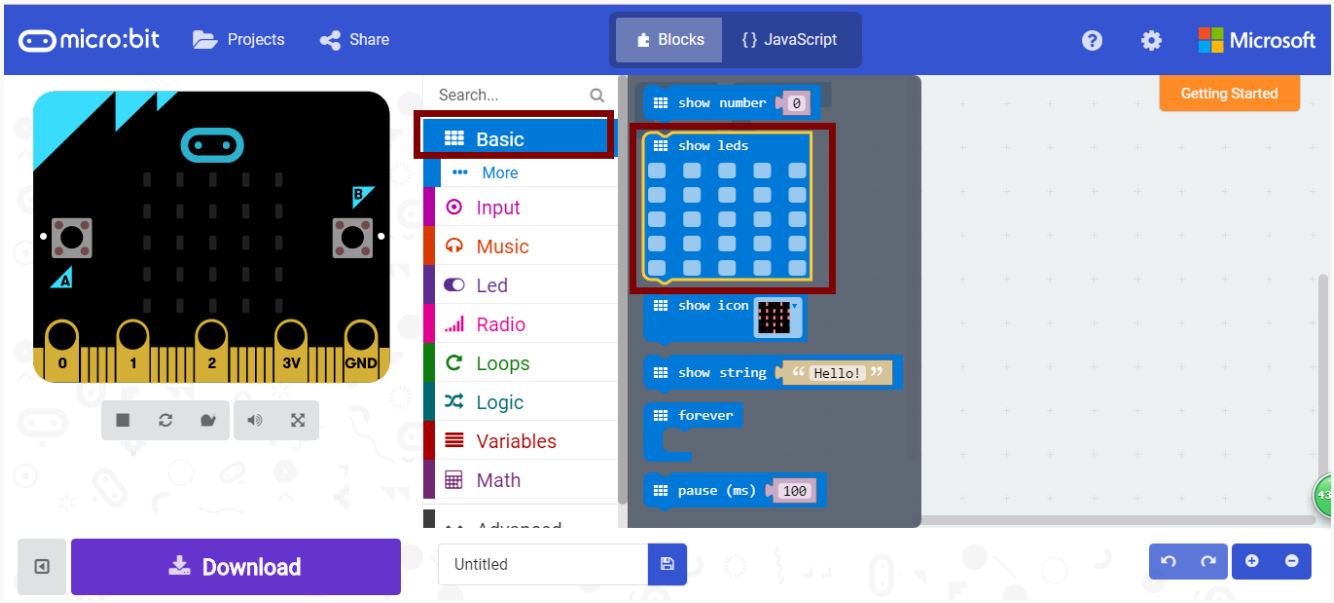
Connect micro: bit to computer via USB cable.

Program

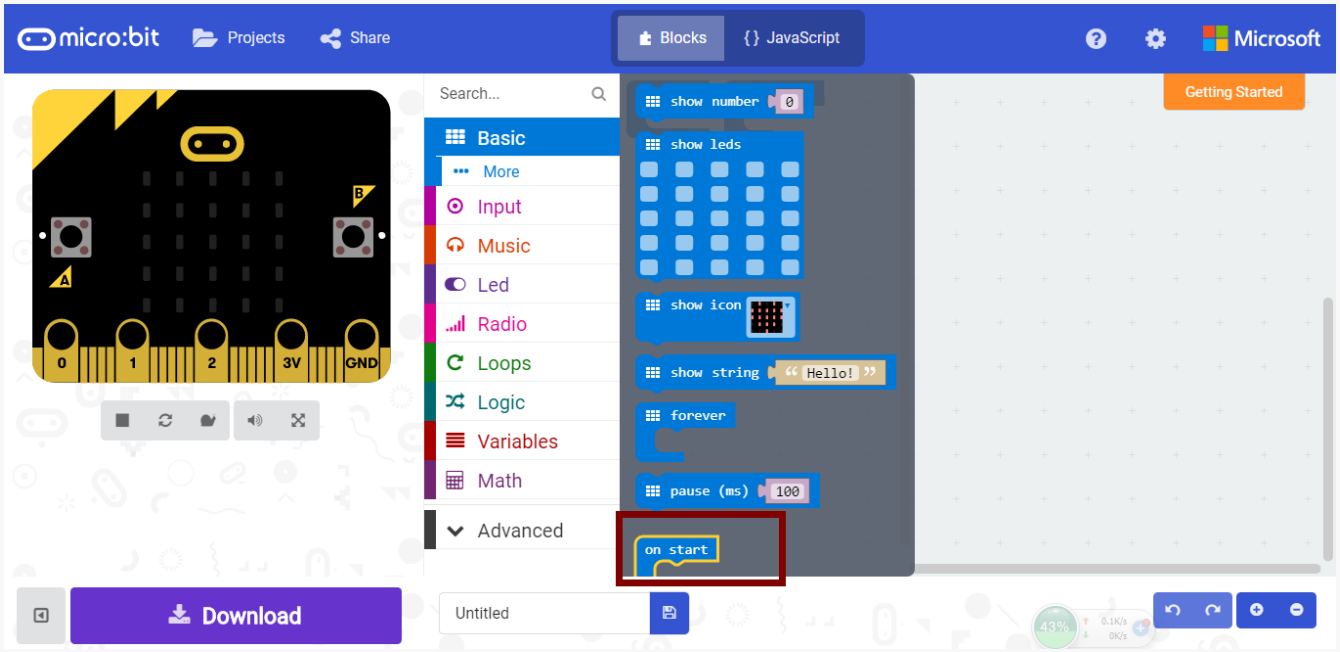
Open MakeCode: https://makecode.microbit.org/.

STEP 1: Start a new project. You may go back to the last chapter if you forget how to do it.

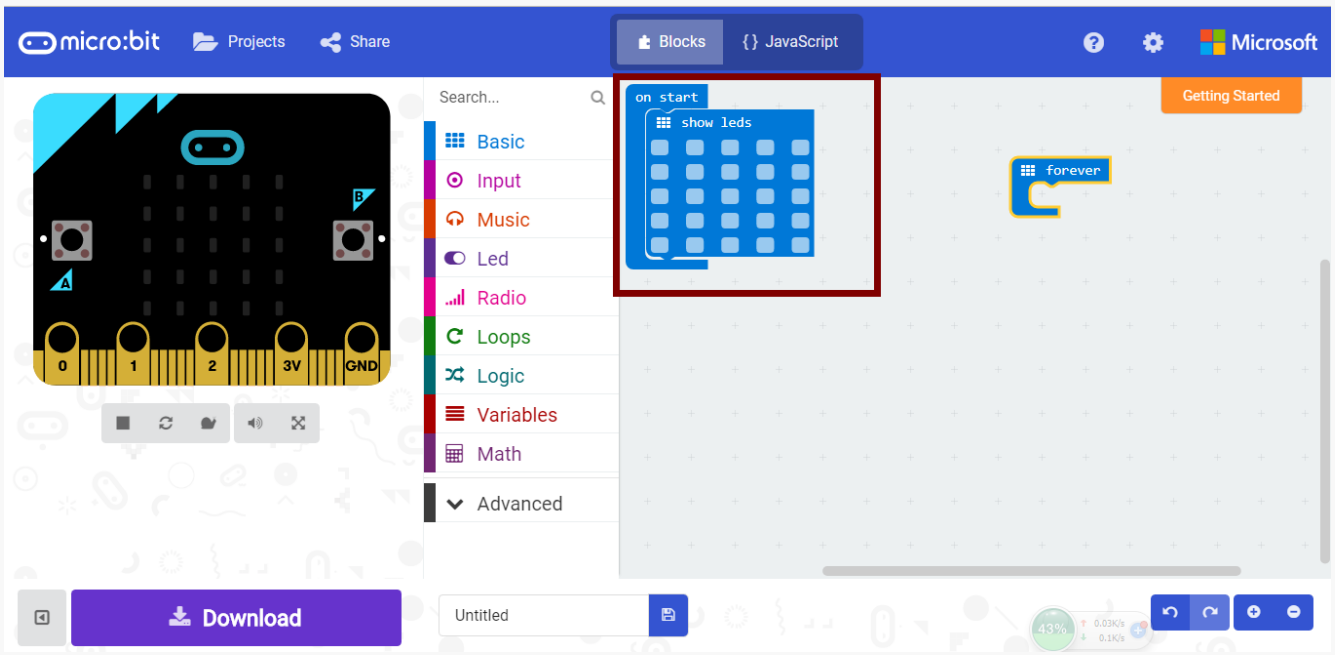
STEP 2: Click "Basic" at the top of the function area and find the "show leds" function.



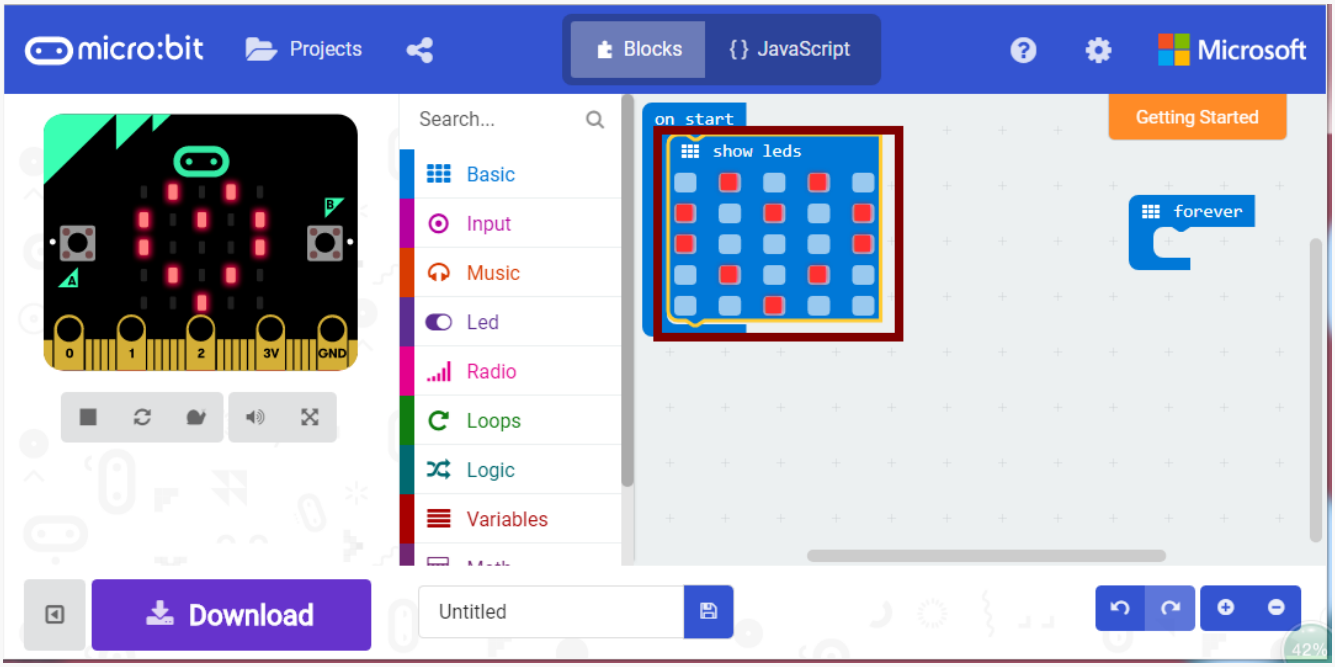
STEP 3: The module "on start" will automatically appears in the programming area after you first open the online website of MakeCode, or you can drag the "on start" function on the bottom of "Basic" to the programming area.



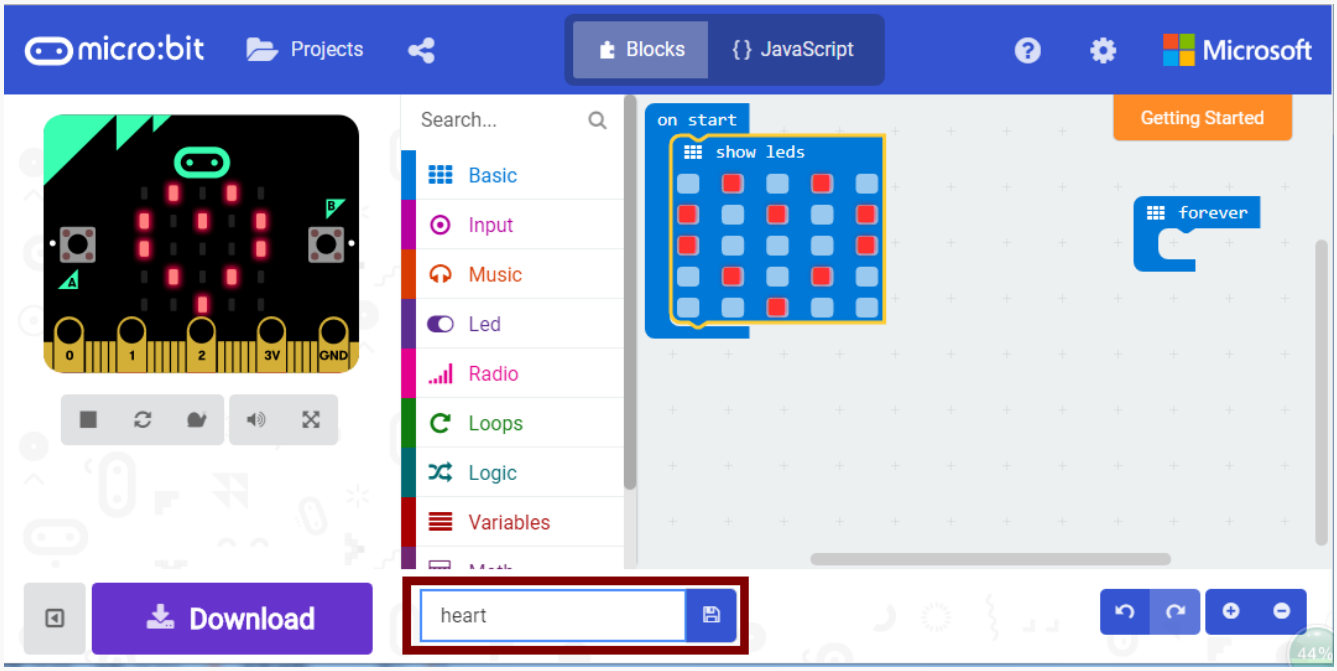
STEP 4: Drag the "show leds" function to the programming area and place it in "on start".



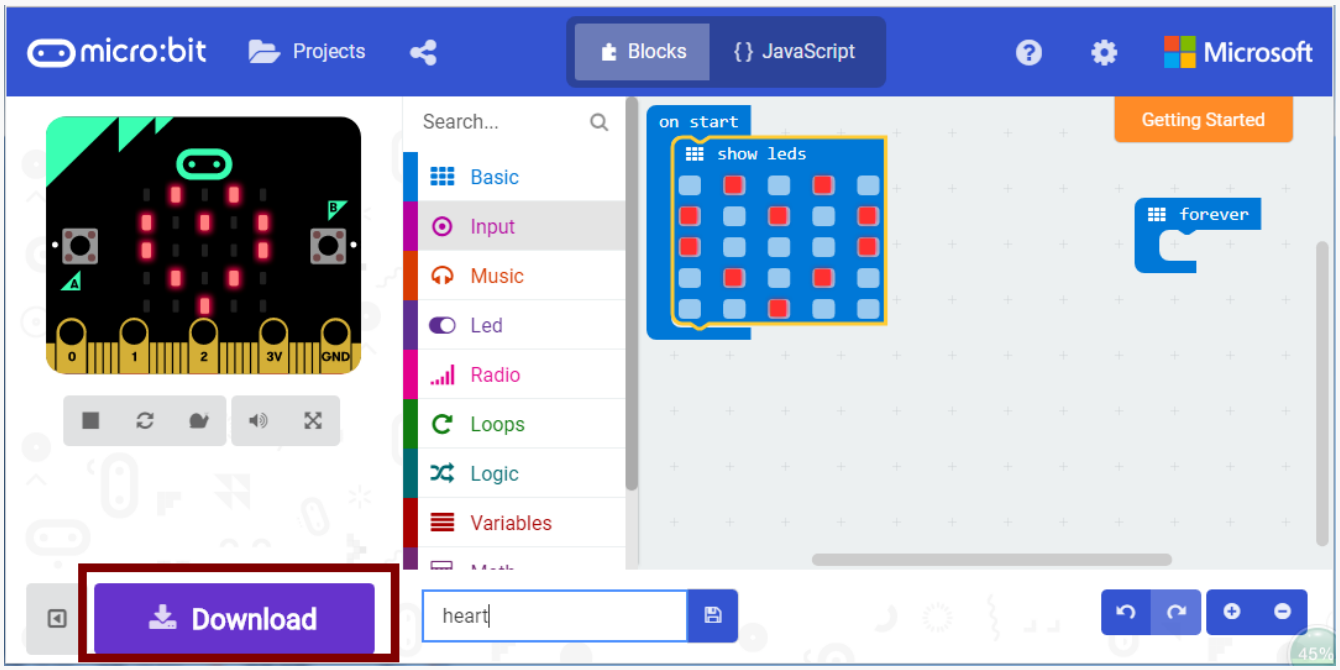
When "show leds" function is inside the “on start” module, you can then click the light blue dot inside the block to edit the pattern. Also, whatever we draw inside the block will be displayed in the simulation window.



STEP 5: Here, we draw a heart in the block. To help us remember, we may rename it as "heart".



STEP 6: Click "Download" to save the file to MICROBIT to the computer, or you may directly save it to micro: bit. If everything goes well, the micro: bit will then display a "heart " emoji. (If you forget how to upload the code, go back and check the STEP 4 in Chapter 1)



Exercises

Try to design an emoji that can switch between two different patterns.

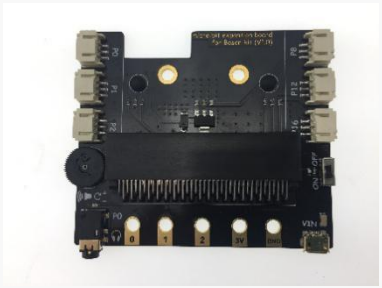
Project 2：Flashing LED Light

In the previous part, we have learnt how to program the LED penal to make our own emojis. In this chapter, we will connect an external LED module and learn how to light it up and make it blink.

Components list

1 × micro:bit

1 × Boson Expansion board



1 × LED module

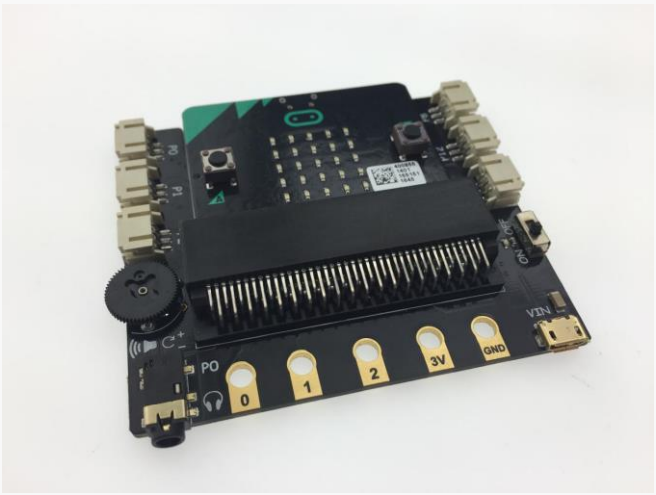
1 × USB cable

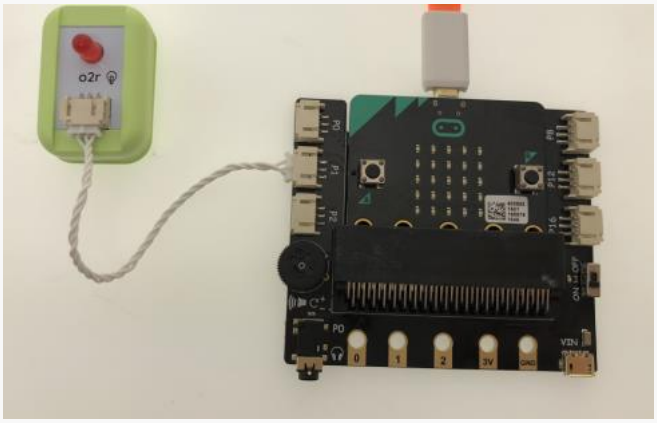
Connection

- Insert micro:bit into the expansion board.

Note： micro: bit will stay inside the expansion board in all following chapters. We will skip this step in all following chapters.

- Connect the LED module to P1





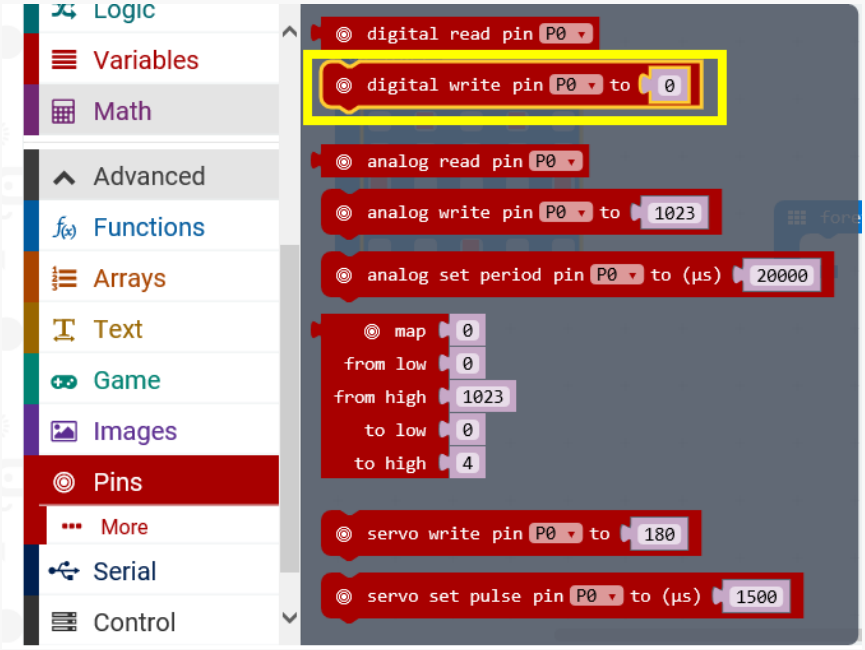
Program

Task 1: Turn on the LED

.

STEP 1: Open MakeCode website: https://makecode.microbit.org/ and start a new project.

STEP 2: This time, Since the LED module is connected to P1 of the micro: bit as an external component, to control the LED, we will need to set the status of the pin that the LED connects to. The function block “digital write” is the one that does the job. “Digital write” can either make the pin output a high voltage (presented as 1), or a low voltage (presented as 0), which correspondingly turn on and turn off the LED. First, we turn on the LED. To do this, go to “Advance"->"Pins", and set digital write pin p1 to HIGH (1).



STEP 3: Put the function “digital write” into "forever" loop (the “forever” loop will show up in the area once create a new program, it can also be found in “Basic”) and the LED connected to P1 lights up. It will be finished as below:

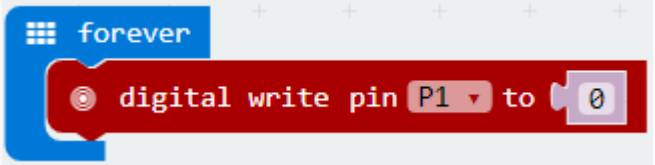
Task 2: Flashing LED

Goal: After we know how to light an LED, we will take a further step to learn how to make it blink.

STEP 1: Start a new project. You may always go back to the STEP 3 in chapter 1 if you forget how to do this.

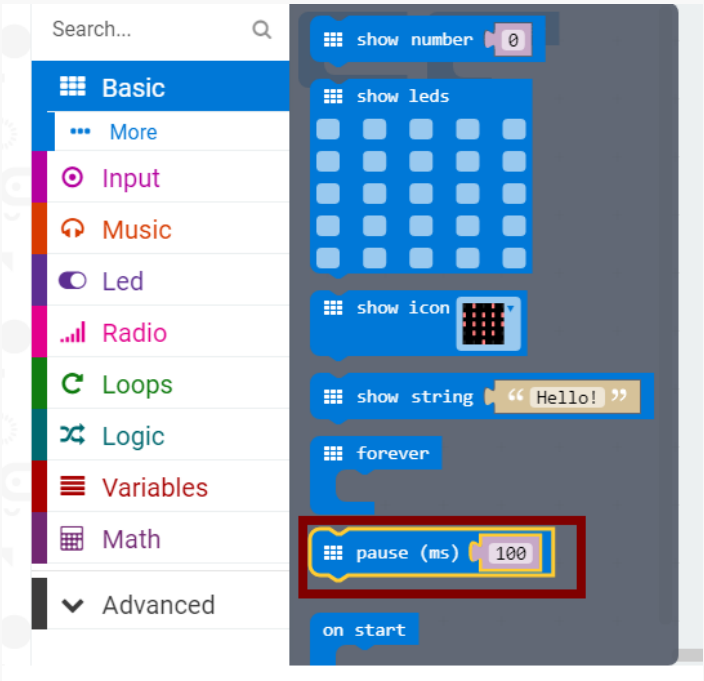
STEP 2: Set the pin status of micro: bit to either turn on and off the LED. LEDs in this experiment belong to digital output. Click the "Advance"->"Pins", set “digital write pin p1” to (0). The pin value can only be 0 or 1. When it is 1, the light stays on. When it is 0, the light stays off.

Set the value of Pin P1 to high (1) and the LED will light up



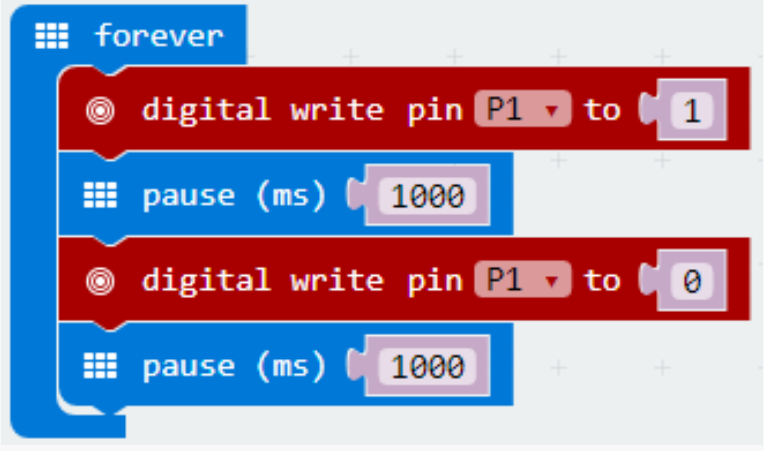
Set the value of Pin P1 to low (0) and the LED will turn off

STEP 3: You will also need to keep the external LED on for 1 second and off for another second in a sequence. To do this, go to "Basic" and select "Pause (ms) (100)". The "Pause" function keeps the same status lasting for a certain period of time.



STEP 4: Put the "Pause" function into "forever" loop and adjusting the numbers.

Combine all the function blocks listed above altogether, we will have the final program as below. In this picture: set "pin P1" as 1 (LED on) and "pause(ms)" as 1000, pin P2 as 2 (off) and "pause(ms)" as 1000, and the program inside the "forever" loop will be running repeatedly in a sequence.



Exercise

You may have heard the SOS emergency signal. Why not invite your classmates to learn and try to program it!

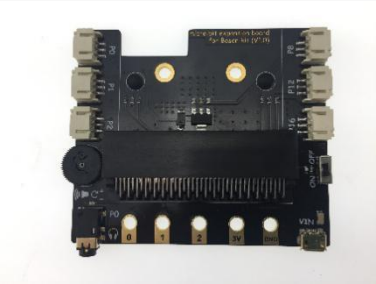
Project 3: Notification Light

As you have learnt how to light an external LED through micro: bit, we will now apply an external button module to control the LED. Also, we will learn how to set the brightness of the light by using a knob.

Components list

1 × micro:Bit

1 × Boson Expansion board



1 × LED module

1 × Button module

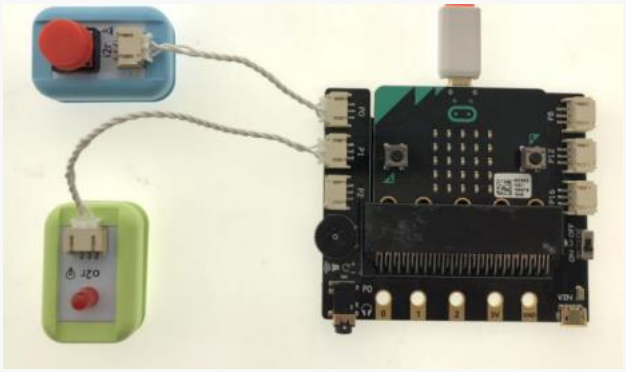
1 × knob module

1 × USB cable

Connection

Connect the button module (knob module) to P0

Connect the LED module to P1

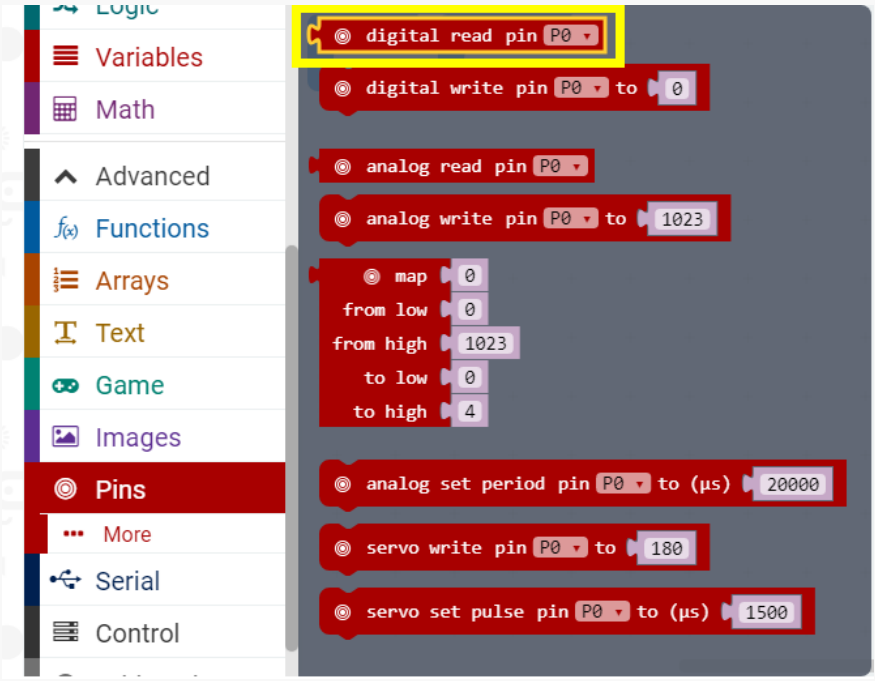


Program

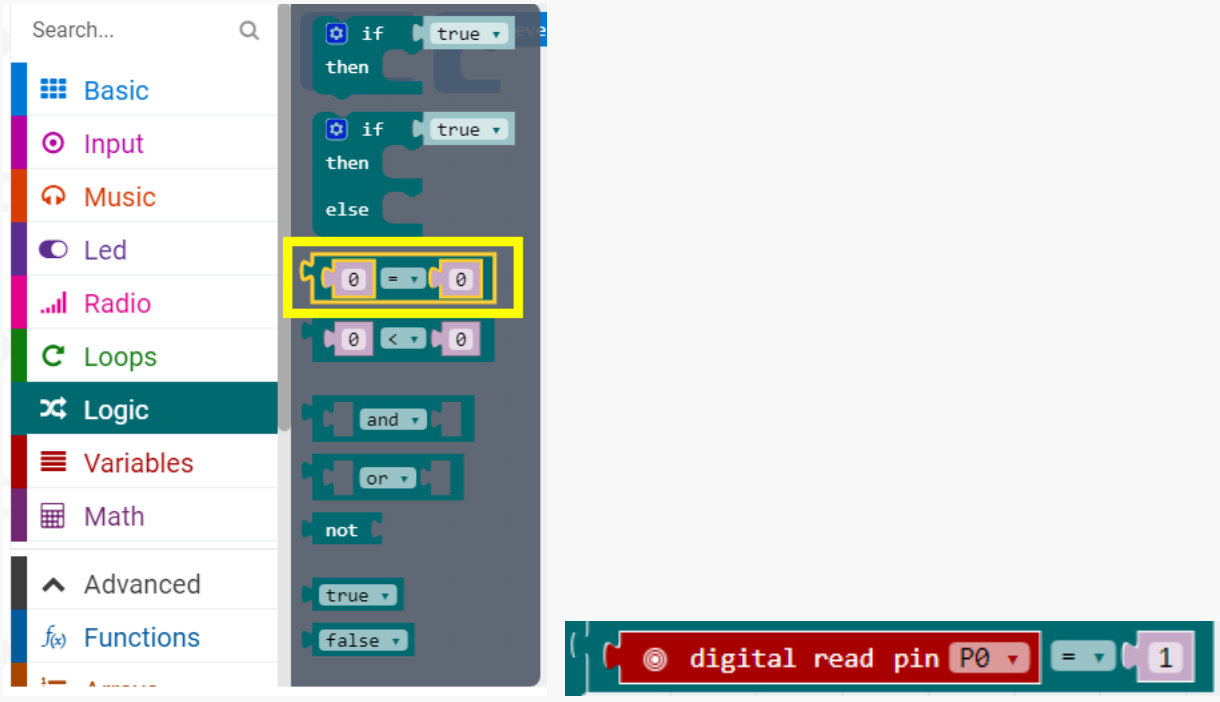
Task 1: button controlled lamp

Goal: when the button is pressed, the LED module turns on. When the button is released, the LED turns off.

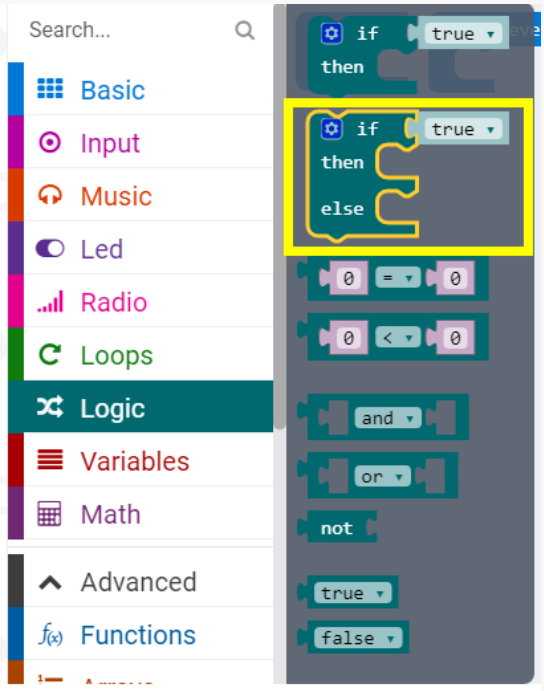
STEP 1: Connect the button module to P0. By reading the value of P0, micro: bit will then get the status of the button. To link the button with the LED, we can write a simple program: When the button is pressed micro: bit receives a “1”, so we will then set the value of the LED (pin 1) as 1 and the LED turns on. When released, set the value of the pin as 0 and the LED turns off. This button module is a digital input module and the “digital read” function can be found under “Advance" ->"Pins".



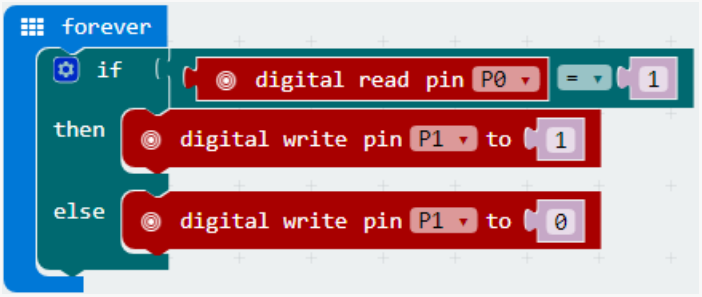
STEP 2: The condition operator "=" under “Logic” is used to determine whether the button is pressed or not. If the status of the button equals 1, then we will know the button is being pressed, vice versa.



STEP 3: If the button is pressed, the external LED lights up; otherwise, it stays off. We will use another condition operator, the “if-else” function. The function is also under “Logic”



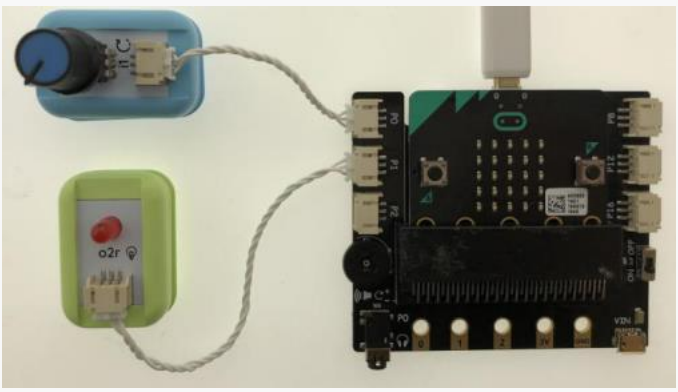
STEP 4: By putting together all the functions mentioned as above, we will have the following program. What the program does is to detect the status of the button, and lights up the LED when the button is pressed.



Task 2： Knob controlled LED

Goal: in the part, we will not only control the LED from on to off, but also use the knob to change its brightness.

STEP 1: Connect the circuit as below. The button module used in task1 is now replaced by the knob module.



STEP 2: The brightness of the LED is linked to the input value of the knob. The knob is an analog input module, which means it generates a signal range from 0 - 1023, by which controls the brightness of the LED correspondingly. The “analog read” function is under “Pins”. Also, remember to set the pin to P0.

STEP 3: The LED is now used as an analog output module. Similarly, the analog output signal is also ranged from 0-1023. "Analog write" function can be found under “Pins”

STEP 4：Now, all what we need is to link the input directly to the output so that the LED will be under control of the knob. Put the “analog read” function inside the “analog write” function, then put them all inside the “forever” loop, and we are done. The final program is shown as bellow.

Exercise:

Make your LED more like a real lamp!

Project 4: Electric Fan

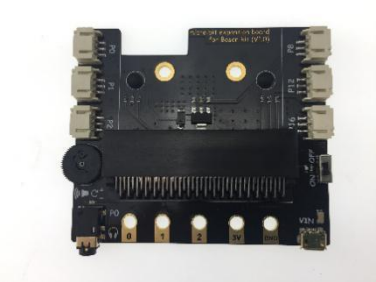
In this part, we will learn how to build an electric fan that runs at different speed.

Before getting started, we will have a quick review of how to switch on or off the fan with the button module. After that, we will learn how to control the speed of a fan with the knob module.

Components list

1 × micro:bit

1 × Boson Expansion board



1 × fan module

1 × button module

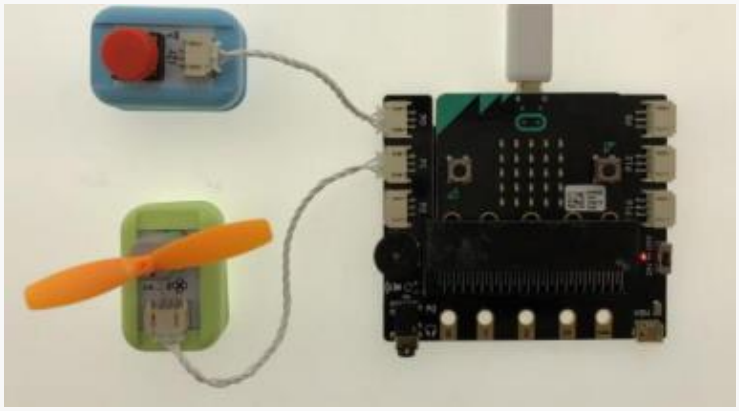
1 × knob module

1 × USB cable

Connection

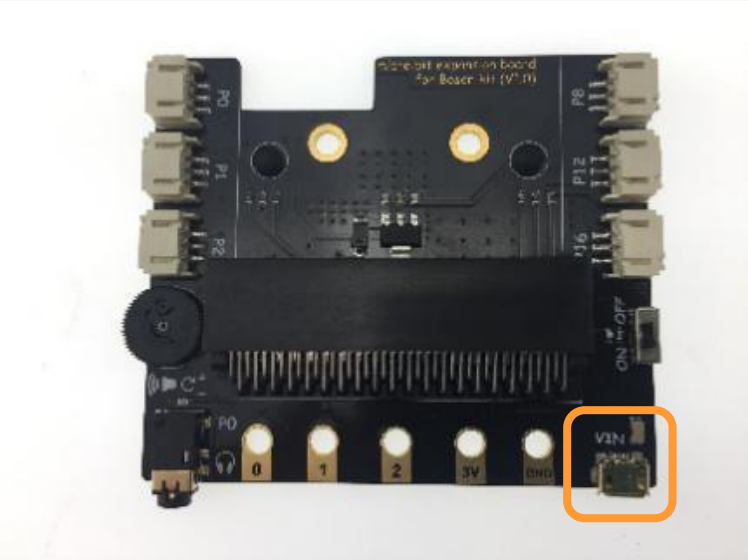
Connect the button module (knob module in the second part) to the P0.

Connect the fan module to the P1.



Electric fan “eats” power!

When connecting Boson modules, we should notice that due to the power consumption of modules such as fans or servos, an external power supply will be necessary. Therefore, we will first use the USB cable to program the micro: bit. When finished, we will switch USB cable to the external power supply on the expansion board.



Program

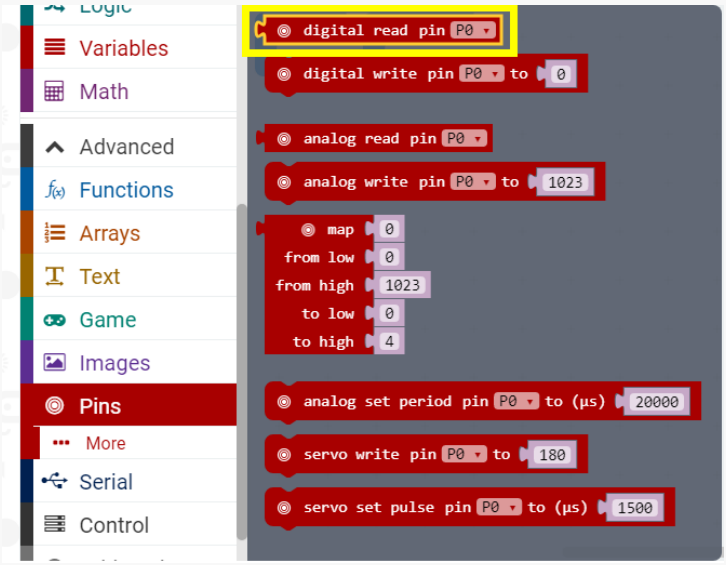
Task 1：Control the fan with the button module

Goal: when the button is pressed, the fan starts running; when released, the fan stops.

STEP 1: The button module connects to P0, which means we will need to use the "digital read" function on P0 to first detect the status of the button.

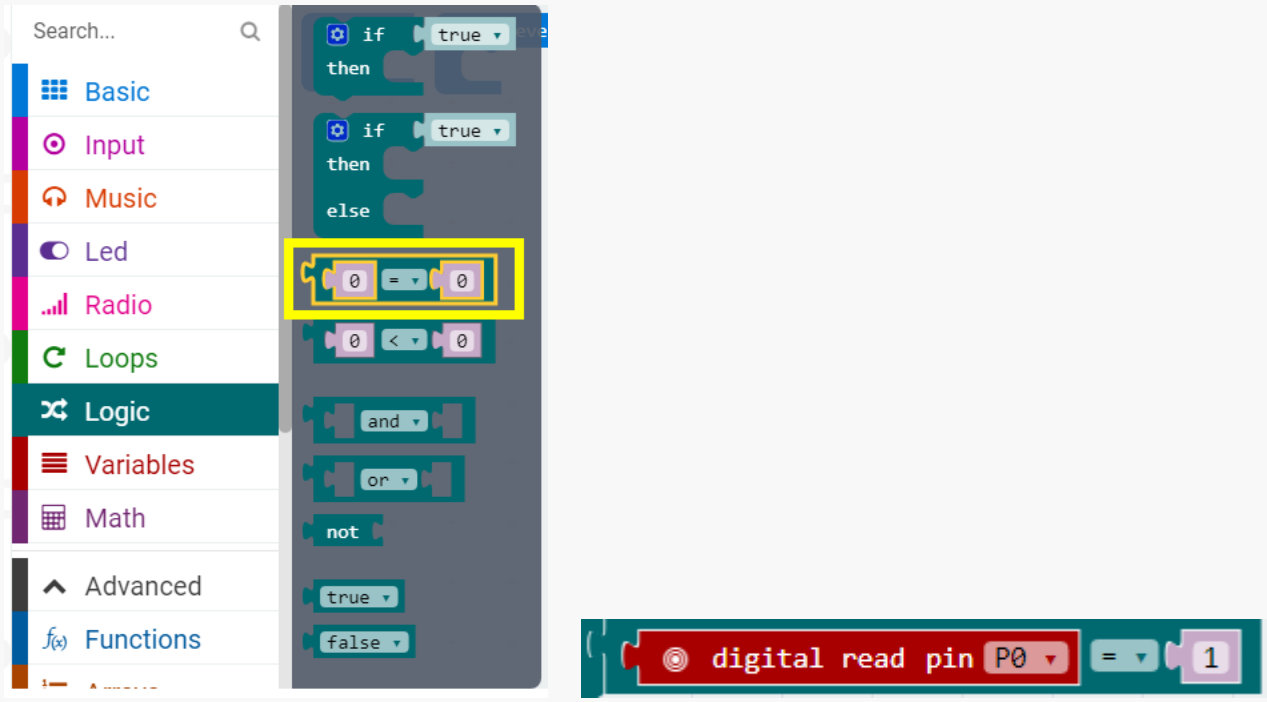
The fan module connects to P1. Likewise, to control the fan, we will use the "digital write" function on P1 to switch it between ON an OFF.

The "digital read" and "digital write" function can be found under "Pins."

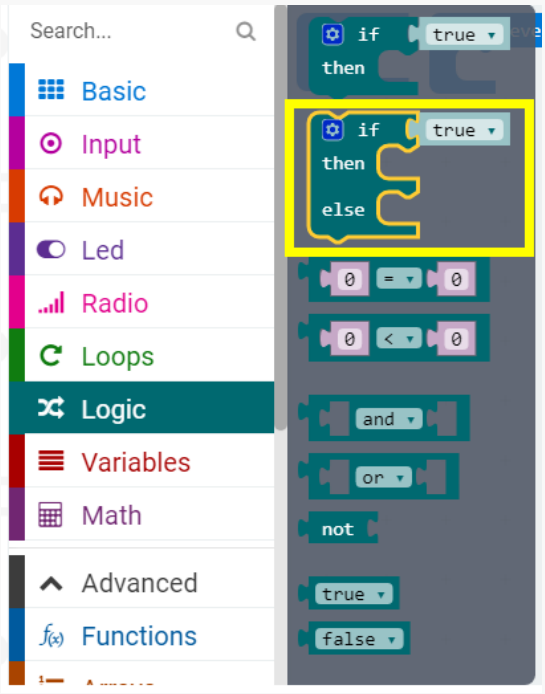


STEP 2: The condition operator "=" under "Logic" is used to detect whether the button is pressed or not. When the button is pressed, the "digital read "function returns "1", which meets the

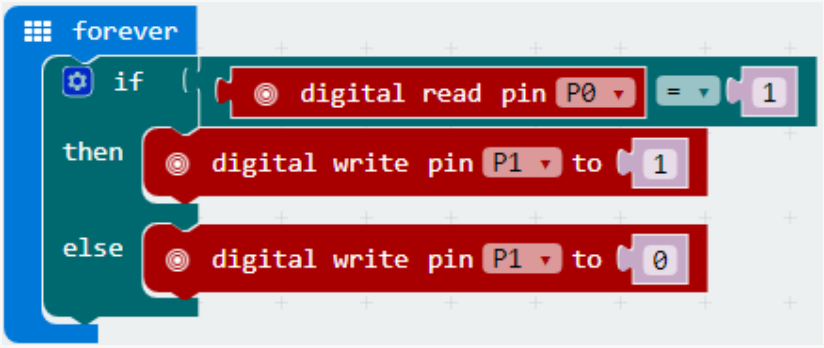
condition of "1" that we preset at beginning, so the program under the statement will be executed. However, when it returns "0", the program under the statement will be skipped.



STEP 3: If the button is pressed, the fan will be turned on; otherwise it stays off. The "if-else" function under "logic" will help us make the choice depending on the status of the button.



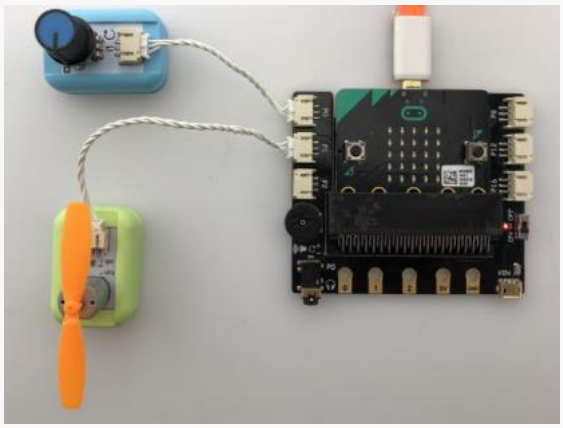
STEP 4: By putting the function blocks mentioned above all together, we should end up with a program with the following function: when the button is pressed, the fan is turned on, when it is released, the fan is turned off.



Task 2: Set the fan running at different speed

Goal: In this task, we will learn how to use a knob module to control the speed of the fan. When a larger analog input value the knob reaches, the faster speed the fan operates, vice versa. The speed of the fan continuously changes when the knob rotates.

STEP 1: Connect the circuit as below. The button module used in task 1 should be replaced by a knob module.



STEP 2: We will be using the fan module here again. To make the speed of the fan controlled by the knob, we need to build a link between the speed and the input value of the knob. The knob is an analog input module, which means it generates a signal ranging from 0 - 1023, by which controls the speed of the fan. To read the signal, connect the knob module to P0, use the “analog read” function under “Pins”.

STEP 3: The fan module is now used as an analog output module, and its value is also ranged from 0-1023. The "analog write" function can be found under “Pins”

STEP4： Put the “analog read” function into “analog write”, and then place all of them into the “forever” loop. The final program is as following:

Chapter 4: A bit further

Above we have learnt how to use the button module and knob module to control an electric fan. But there is still a long way to go, you may eager to explore more about the micro: bit. In this chapter, more sensors will be introduced to make your device smarter and more functional.

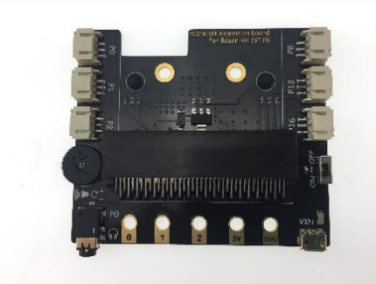
Project 1: Electronic Candle

In this project, we are going to make a device simulating a candle to reduce carbon emissions. Also, to make it more like a real one, the fire should be able to be put off by blowing air onto it. Let's make it happen!

Components list

1 × micro:bit

1 × Boson Expansion board



1 × LED module

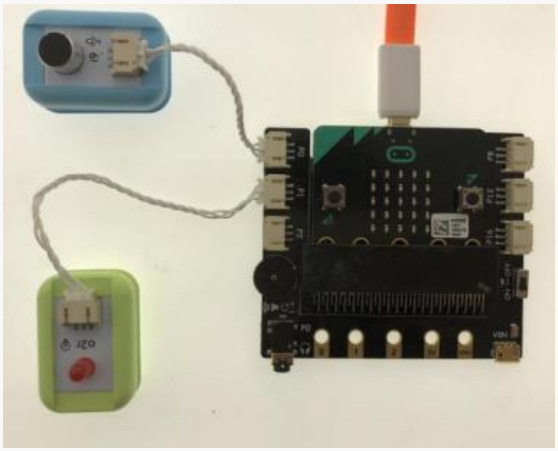
1 × sound sensor module

1 × USB cable

Connection

Connect the sound sensor to P0;

Connect the LED module to P1.



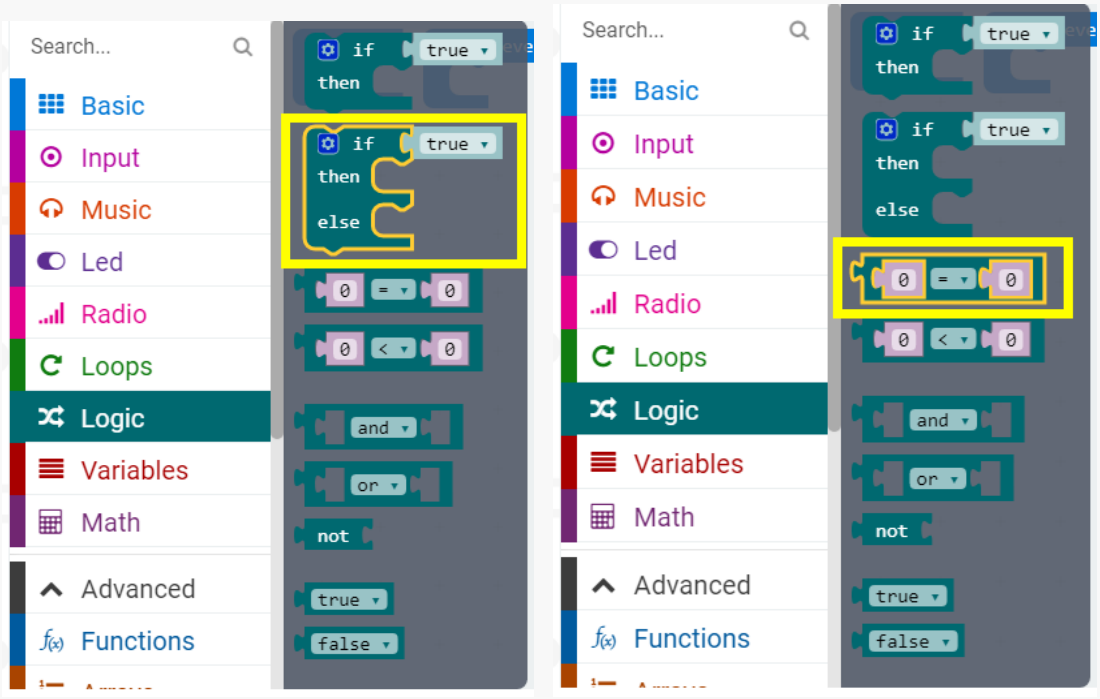
Program

To build an electronic candle, we must first know how the sound sensor works. We can connect the sound sensor to the micro: bit and see how strong our sound is by looking at the number displayed on the LED panel.

Task 1： Measure the Volume

Goal: When the analog value of sound sensor is less than 50, the on-board LED displays a number "1", indicating that the volume is low; when it is greater than 50, the on-board LED displays a number "2", indicating that the volume is high.

STEP 1: We will put the LED module aside for a moment and focus on the logic part. Function “if else” and operator “=” will be applied in this part. Both of them should be placed into “forever” loop. The “if else” function can be used to judge which number should be displayed. Which means that if the logic statement on the right side is met, the code sentence to the right of "then" should be executed, whereas the sentence to the right of "else" should be executed if the logic statement is not met.



STEP 2: The sound sensor is connected to P0, which means we will use the “analog read” function to get the readout on P0. As explained before, the analog signal ranges from 0 to 1023.

STEP 3: Now, we will need to let the program determine the level of the voice. The operator "<" under the "Logic" does the job. As shown in the picture below, When the analog value of sound sensor is less than 50, the on-board LED displays a number "1", indicating that the volume is low; when it is greater than 50, the on-board LED displays a number "2", indicating that the volume is high.

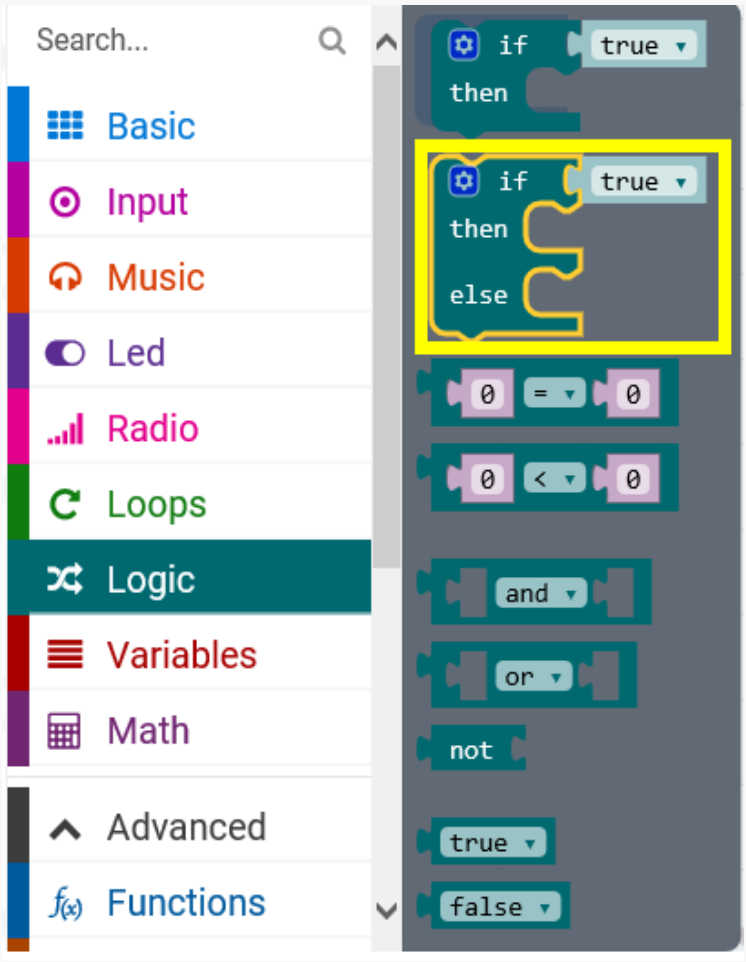
Make some noise around the sensor, or simply blow some air to test if the micro: bit is able to show different number with respect to the volume.

Also, if the sensor is being too responsive, you may need to change the "50" after the "<" operator. It also works the same way around if the sensor is not responsive.

Task 2: electronic candle

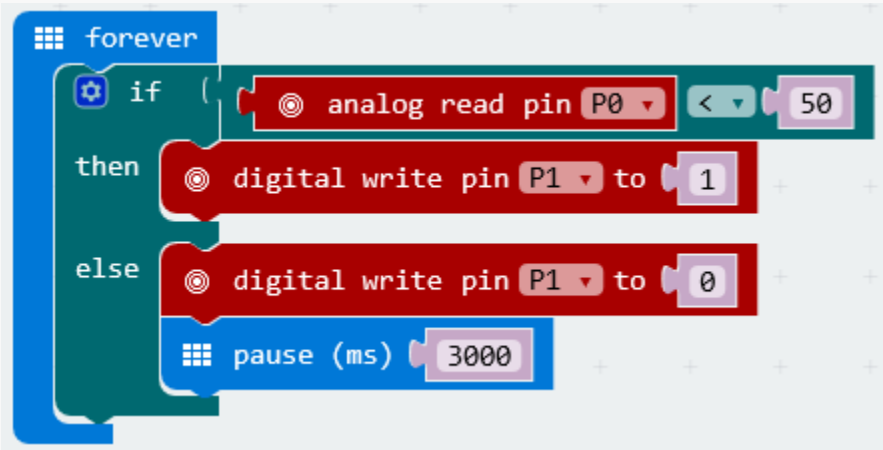
Goal: The micro: bit can measure the volume using the sound sensor, based on that, we want to use it to control the candle. Here is what we are going to program: once we blow air towards the sensor, the candle goes off for a second.

STEP 1: We will use the "if else" function again in our program. However, instead of controlling an LED panel, we just need to control the LED connects to P2. The logic is very straight forward, once the volume exceeds the preset value, the light turns on, vice versa. Since we want the candle to keep on burning, the program should be put inside the "forever" loop.



STEP 2: However, that's still not enough! When we blow air to the sensor, the volume will only exceed the preset value for a very short time. Sometimes it happens so fast that we won't be able to notice the change. To solve this issue, we will need to add a "Pause" function to extend the duration. Say if the duration set to be 3000 (3000ms = 3 seconds), once we blow the candle, the light will go off for 3 seconds, and then go back to the beginning of the "forever "loop and light up again.

Here is how the program looks like.



Project 2: Automatic Door

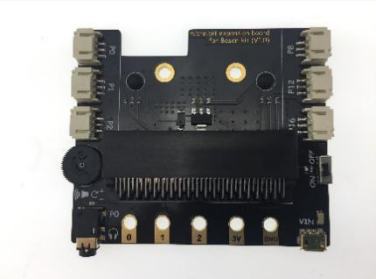
Automatic doors can be found quite often in our daily life. When we are about to enter a convenient store, the door opens automatically and sometimes along with the sound "welcome~".

In this lesson, we are going to learn how to build an automatic door using a motion sensor, which detects if someone comes over, and a servo to simulate the action of opening the door. Now, let' s begin！

Components list

1 × micro:bit

1 × Boson Expansion board



1 × button module

1 × motion sensor module

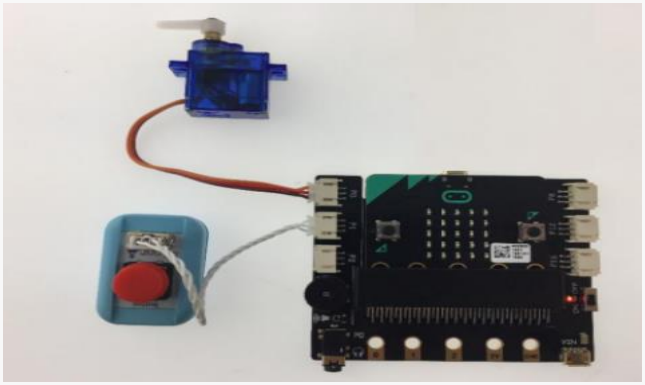
1 × servo module

1 × USB cable

Connection

Connect the servo to P0

Connect the button module (motion sensor in the second part) to P1

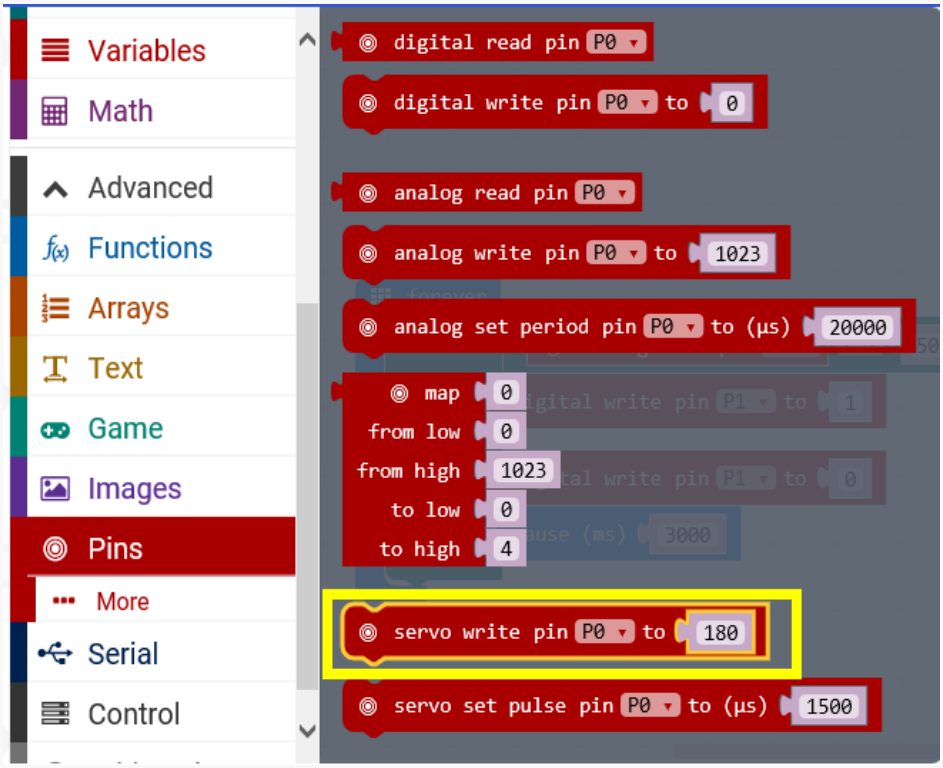


Program

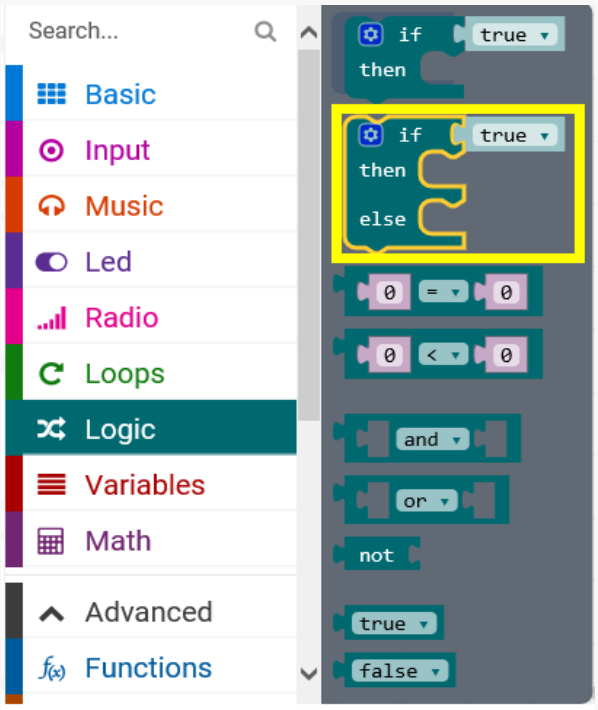
Task 1: Use the button module to control the servo

Goal: When the button is pressed, the servo rotates to 100° , and the LED penal of the micro: bit shows "O"; When released, the servo stays at the 0° , and the LED penal of the micro: bit shows "X".

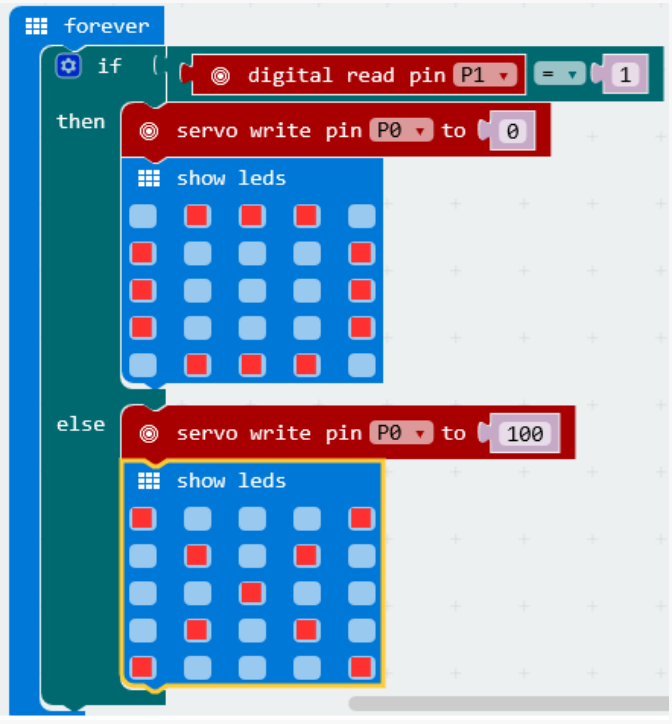
STEP 1：The "servo write" function under "Pins" is used to control the servo. Since the servo is connected to P0, we will leave the "P0" as it is. The number "180" is the angle that we want the servo to reach. By changing this value we can set the servo to rotate between 0° ~180° .



STEP 2：Now, use the "if-else" function to determine the position of the servo based on the status of the button. When the button is pressed, the servo goes to 100° , otherwise it stays at 0° . Also, don't forget to add the "O" and "X" to indicate the status of the button.



STEP 3: By putting everything together, we will end up with a program shown as below.

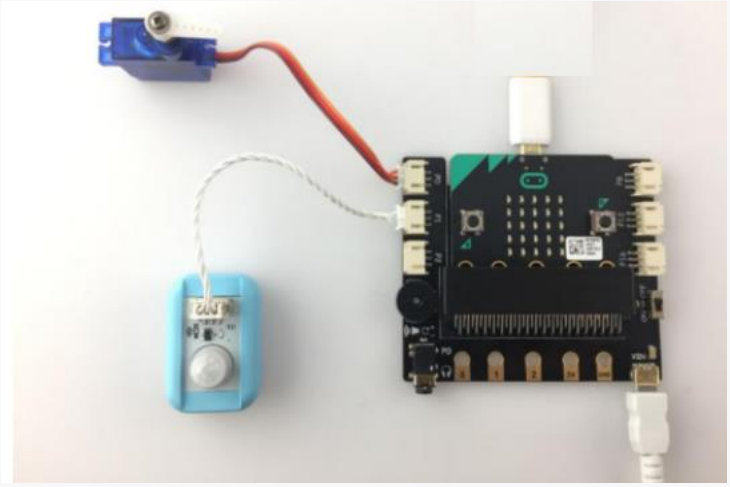


Need to notice that the servo has a relatively large power consumption, so we will need to connect to the power source through the external USB power port.

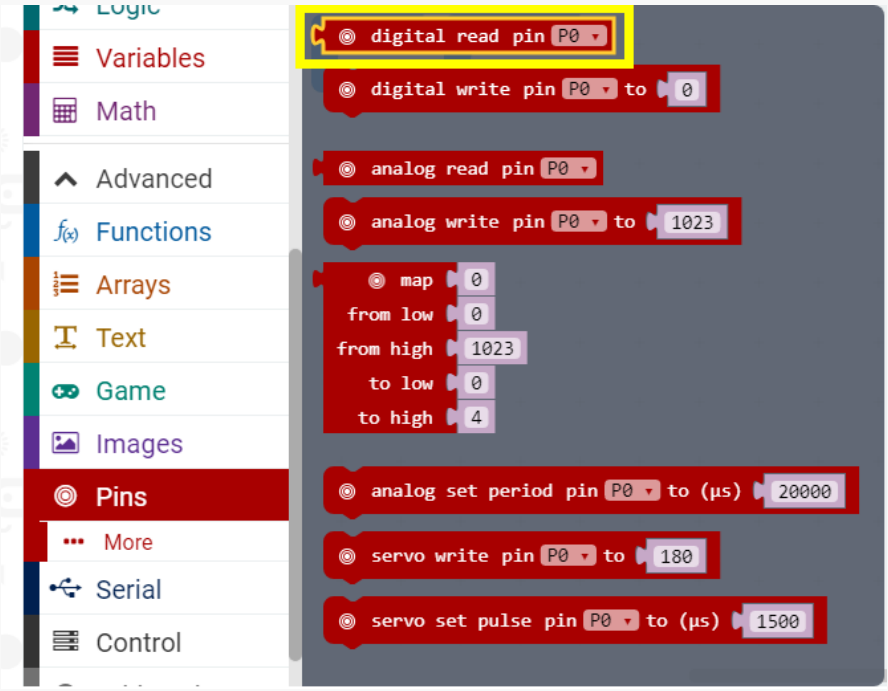
Task 2: Motion controlled door

Goal: we will replace the button with a motion sensor to control the servo. When someone approaches, the door opens. Otherwise it stays closed.

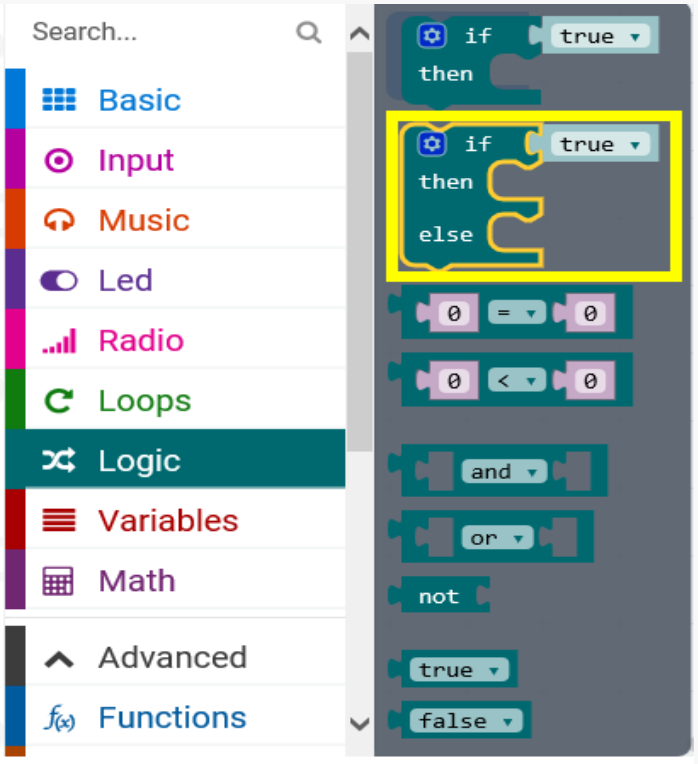
STEP 1: Replace the button module with the motion sensor.



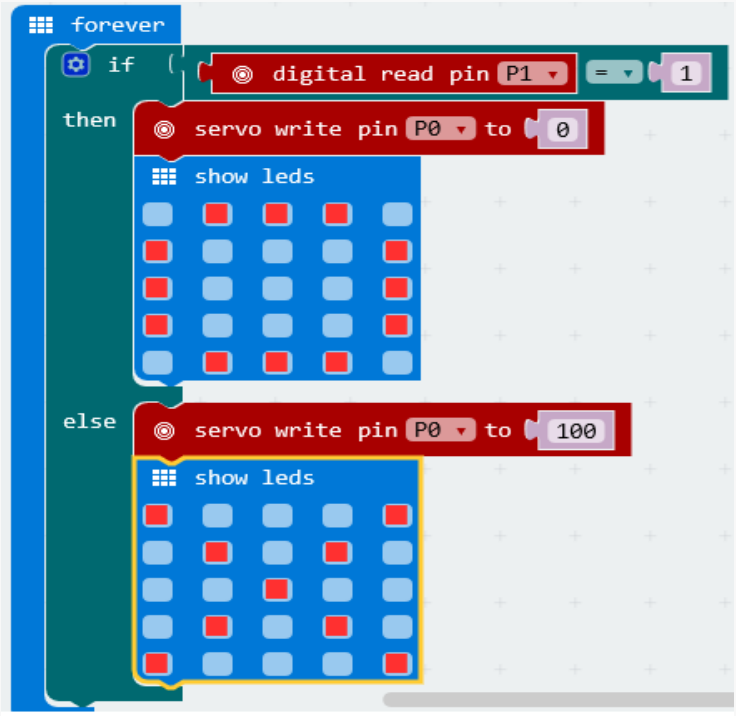
STEP 2: The motion sensor is connected to P1. Thus, we can read the value of the motion sensor. Function used in this part is: "Pins"=>"digital read". Remember to replace the default P0 by P1.



STEP 3：When someone is detected by the motion sensor, the servo rotates to 100° , the micro: bit shows "O"; otherwise the servo rotates to 0° and the micro: bit shows "X". The function block used in this step is "if else".



STEP 4: By putting all above mentioned functions altogether, we will have the final program as below:



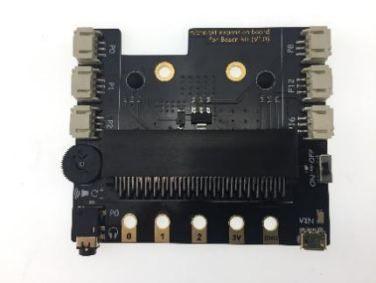
Project 3: Music Box

Various eye-catching music boxes in different shapes have always been popular in gift shops. You must have been deeply touched by their ornate appearances and clever designs. This lesson, We will help you in building a smart music box. Now, let's begin.

Components list

1 × micro:bit

1 × Boson Expansion board



1 × headphone

1 × motion sensor module

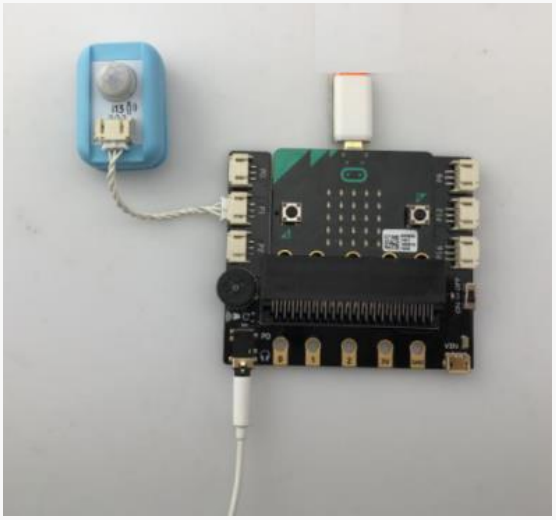
1 × USB cable

Connection

Connect micro: bit to the computer and plug the headphone into the jack.

Connect the motion sensor module to P1.

Note: since P0 is occupied by the audio jack, it will no longer be available to connect other modules.

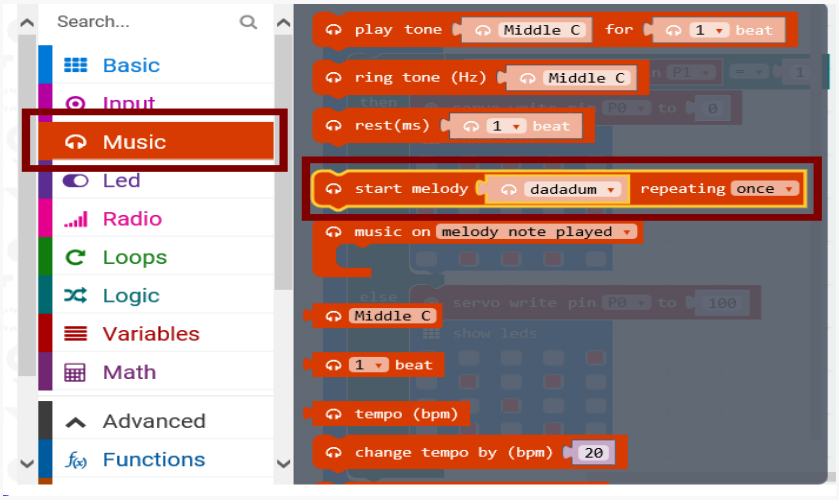


Program

Task 1: Electronic music box

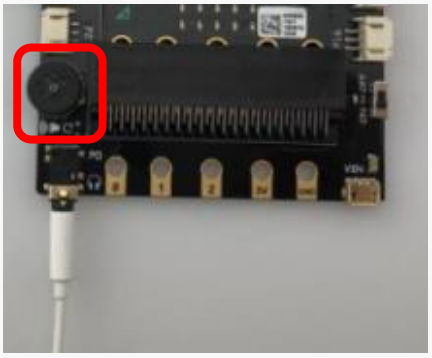
Goal: In this part, we will learn how to let micro: bit play a piece of music.

STEP 1: In this step, we will use the "Music" function. Move the function "start melody 'dadadum' repeating once" from "Music" into the "forever" loop. You may also choose other songs that are available from the list.



STEP 2: Since the "dadadum" lasts for 4 beats, the “rest (ms)” function from “Music” is required for this part. The final program is as following:

volume of micro: bit.

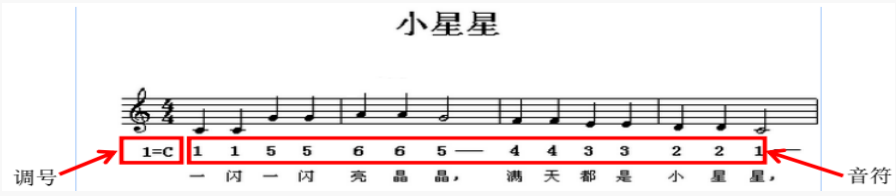


Task 2 : Smart music box

Goal: We will build a smart music box in this task. When someone approaches, the micro: bit will play a piece of music of your choice; when no one is near, it stays quite..

STEP 1: Edit the "Twinkle, Twinkle Little Star" .

Firstly, we will try to edit the first sentence of the “Twinkle, Twinkle Little Star”. Different tones and notes are required for editing the vocal frequency.

Tone 

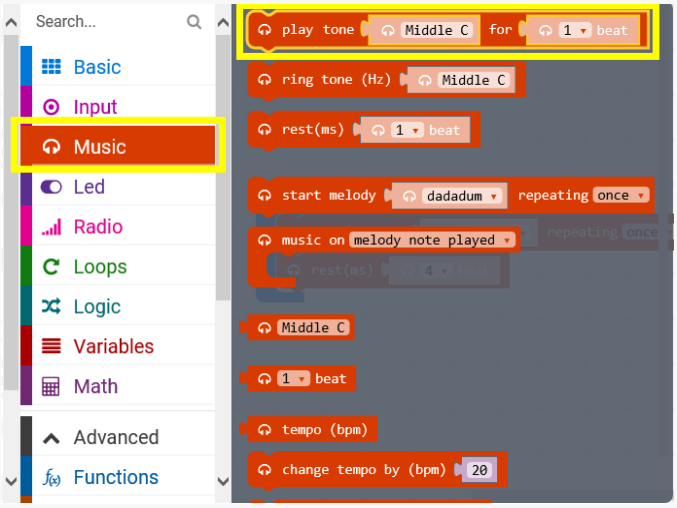
Notes

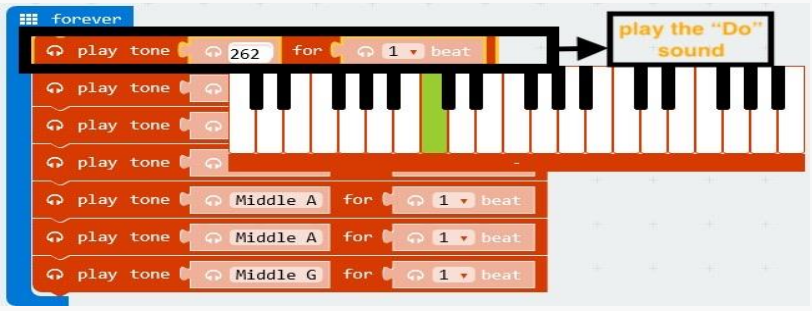
"Twinkle, Twinkle Little Star" .

Take the middle C as example, the sound frequency and its corresponding notes are shown in the following table:

not es	1	2	3	4	5	6	7
Hz	262	294	330	350	393	441	495

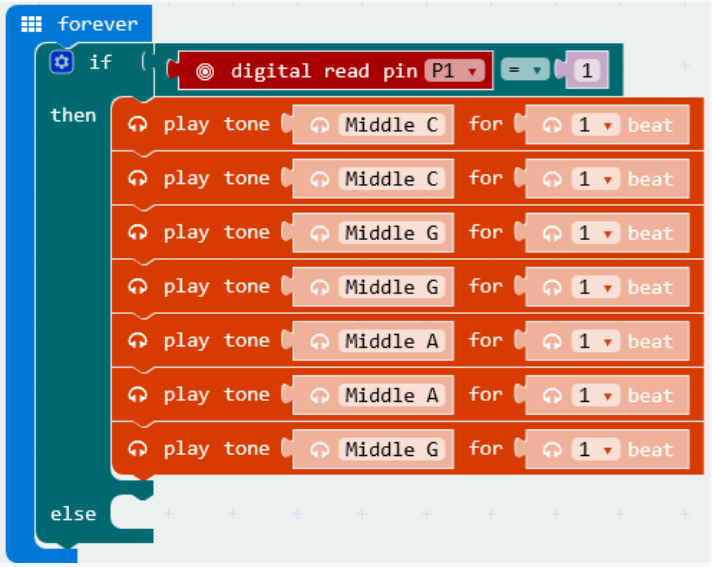
The function “play tone Middle C for 1 beat” from “Music” will be used in this step. Here, you can set the tone and beats.





When above listed functions are combined altogether inside the "forever" loop, our micro:bit is able to play the first sentence of the "Twinkle, Twinkle Little Star". But can it become even smarter? The motion sensor module will be our key again. When combining it with the "if else" function, the smart music box will come to life.

STEP 2: When motion sensor is connected and "if else" function is added to program, by putting all functions listed above altogether, the final program of the smart music box looks like below:



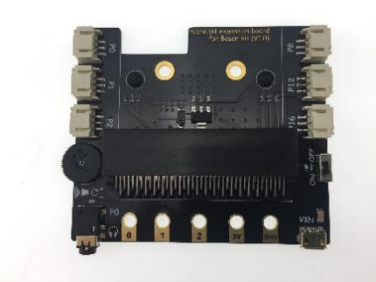
Project 4: Colorful LED Strip

After having learned how to build a smart music box, you may still want to explore more about the micro: bit. Music boxes in our daily life always wear colorful lights. Therefore, the project to make colorful LED strip will be introduced here to help you building a distinctive music box. Come and join us to get into the fantastic light world!

Components list

1 × micro:bit

1 × Boson Expansion board



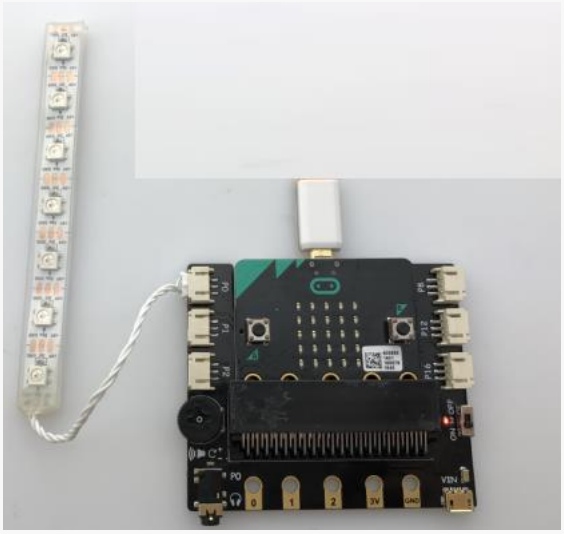
1 × RGB LED strip

1 × Sound sensor module

1 × USB cable

Connection

Connect the RGB LED strip to P0.

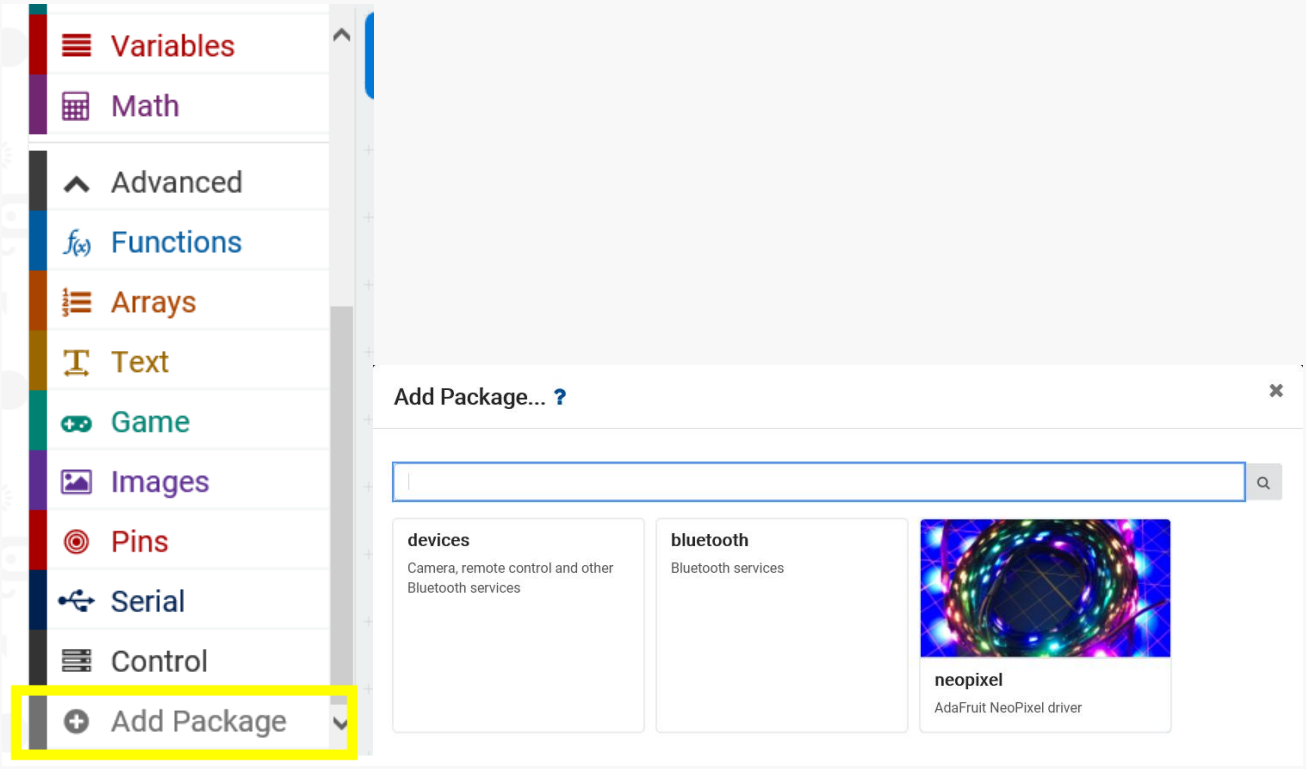


Program

Task 1: Turn on the RGB LED strip

Goal: In this part, we will learn how to use micro: bit to program the RGB LED strip.

STEP 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.

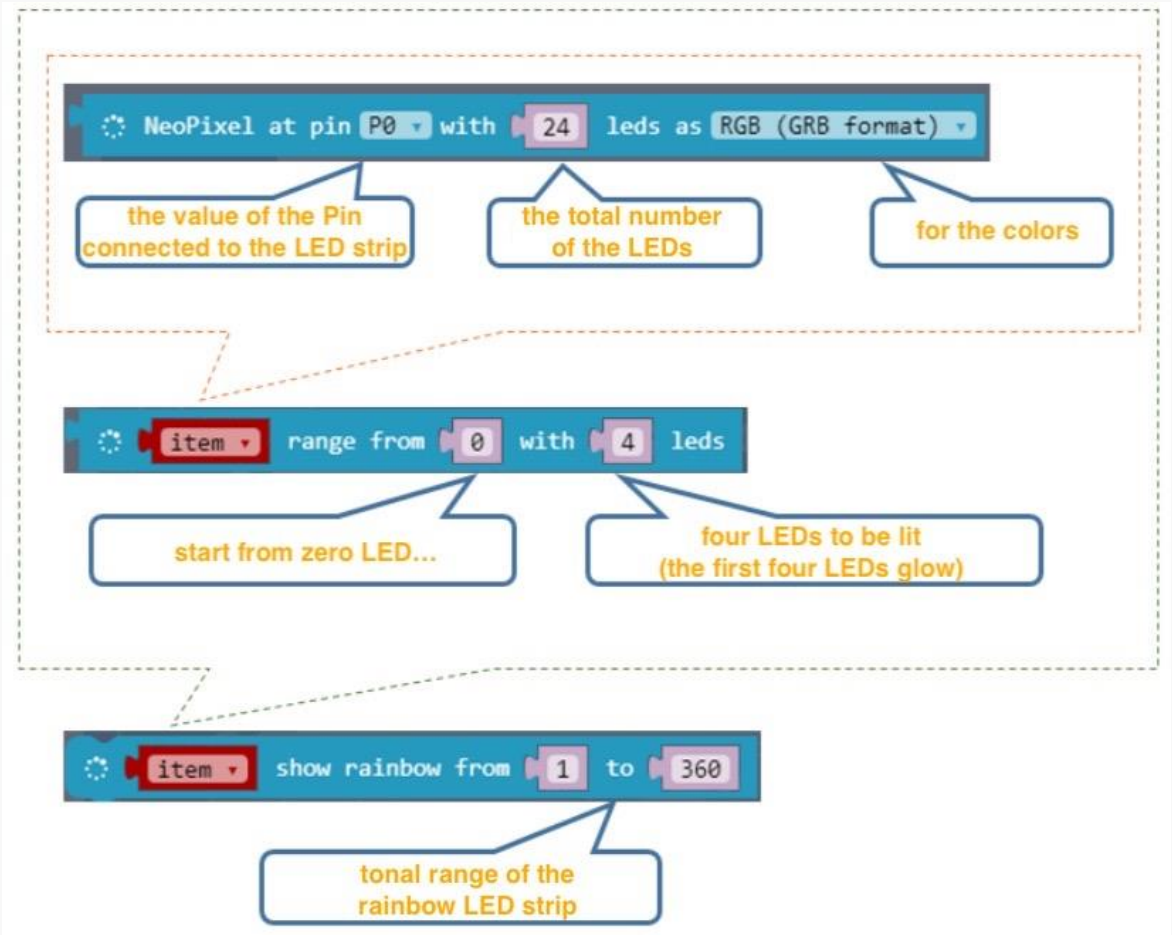


STEP 2: To start using the Neopixel function, we will first setup how many LEDs are included in the light strip and which pin it connects to. Will need to put these two parameter in the function block as following:

After this, we need 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:

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:

The diagram will help you to have an overall understanding the meaning of each parameter: (try to adjust the parameter and see how the light effect changes)

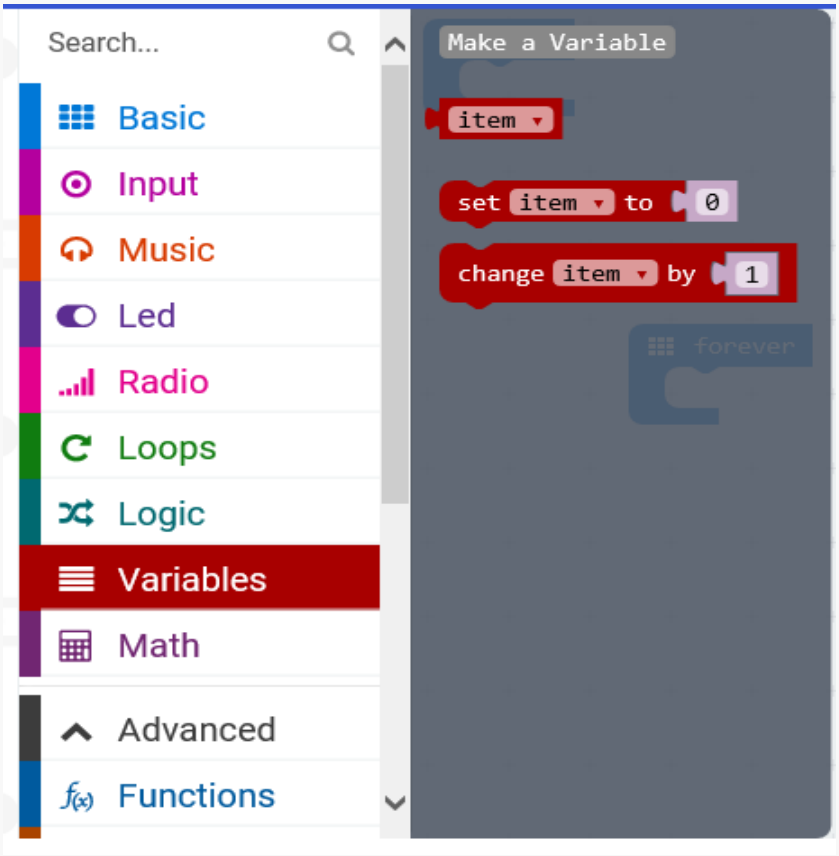


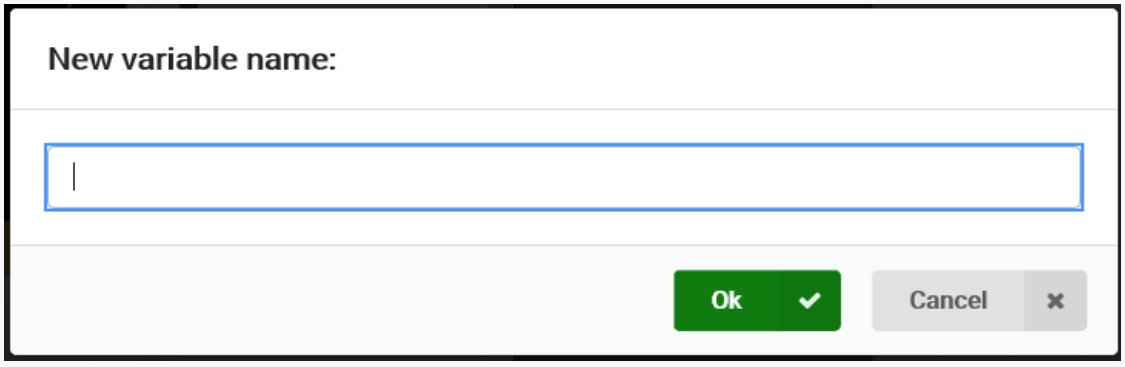
STEP 3: The pin P0 connected the strip comes with 7 LEDs. To add a rainbow light effect to the strip, we will need to set the color range from 1-360 (red gradually changes to blue). Then, combine all the function blocks and put them inside the "forever" loop, and our final program looks as following:

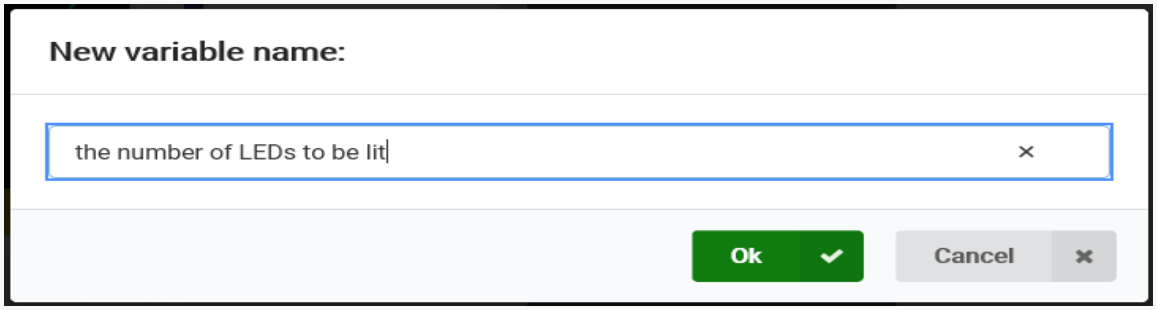
Task 2: Turn on the rainbow lights

Goal: We have learned how to turn on and set light effect to the LED strip. But can we make a simple light animation, say make the LEDs turn on in a sequence?

STEP 1: The key to create an animation with the LED strip is to change the parameter of the light effect. We will create a variable that changes throughout the time and use it to replace the preset constant parameter. To setup such a variable, go to: “Variables” => “Make a Variable”, and set the name of the variable (to be clear, we named the variable as "the number of LEDs to be lit"), Then click OK.

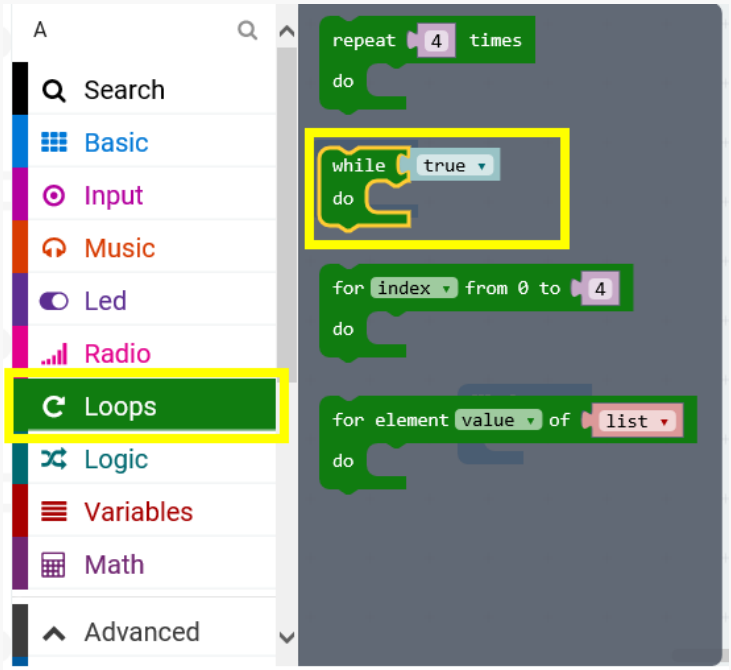


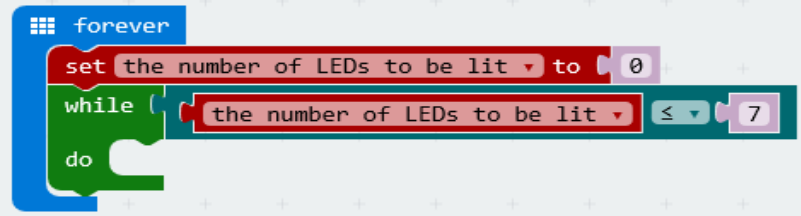




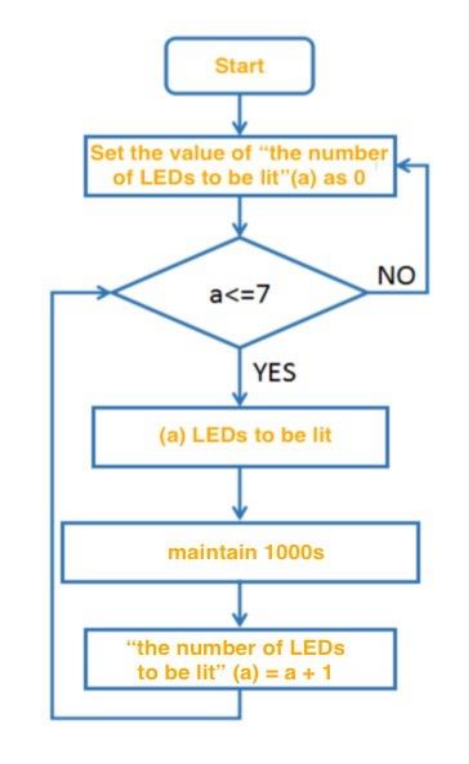
STEP 2: All the LEDs are starting turned off, which means the initial value of the variable "the number of LEDs to be lit" should be set to 0. Then, we will use a loop function to make the variable getting larger by 1 in each loop. Also, to make sure that the total number of the LED never exceeds 7, we will need to setup the initial condition by using "while do" function.

Use the logic operator "<" to set the condition, we will have: when "the number of LEDs to be lit" is less than 7, the "while do" loop will make the LED light up 1 by 1 until all the LEDs are turned on. By putting all above listed function together, the final program goes as:

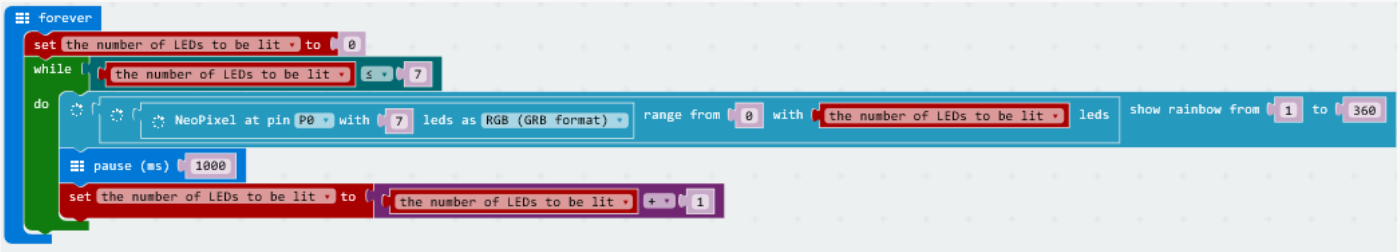




STEP 3: Now, the rest we need to do is to make the parameter getting larger by 1 in every loop. However, we don't want this happen too quickly, a "rest" function may help us slow it down. The flow chart below may be helpful in helping us figure out what the program looks like:



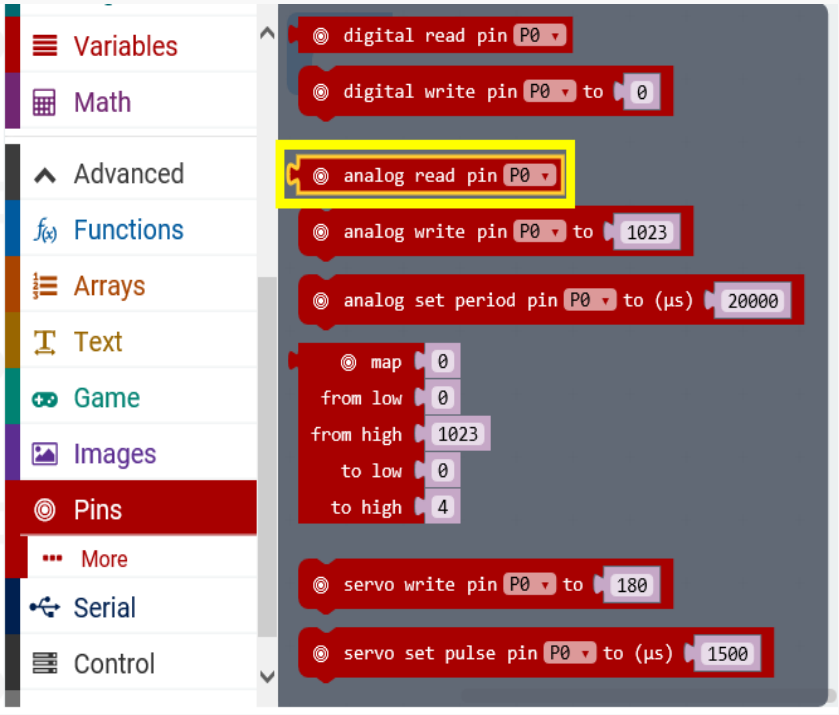
STEP 4: To make the LED light up 1 by 1 in every 1 second, we will need to put the variable "the number of LEDs to be lit "inside the "LED range from" function, and put a "pause 1000ms" in the loop. And here is what we have.



Task 3: RGB LED strip controlled by a sound sensor

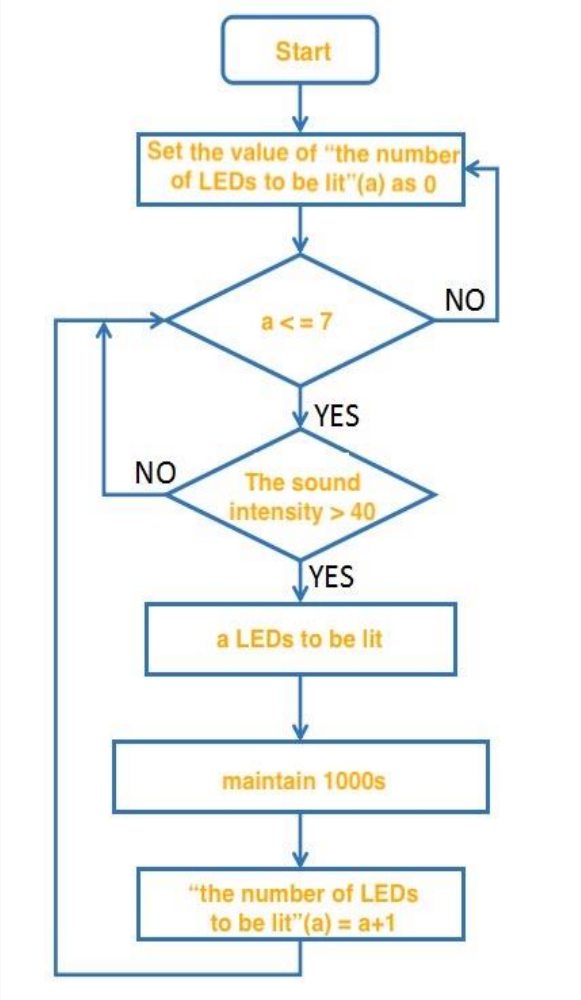
Goal: What we will learn in this part is to use a sound sensor to control the LED strip: when sound is detected, 1 more LED lights up on the strip.

STEP 1: First we need to setup the sound sensor just as what we did in previous chapters. Remember in our case, the sensor is connected to P1.

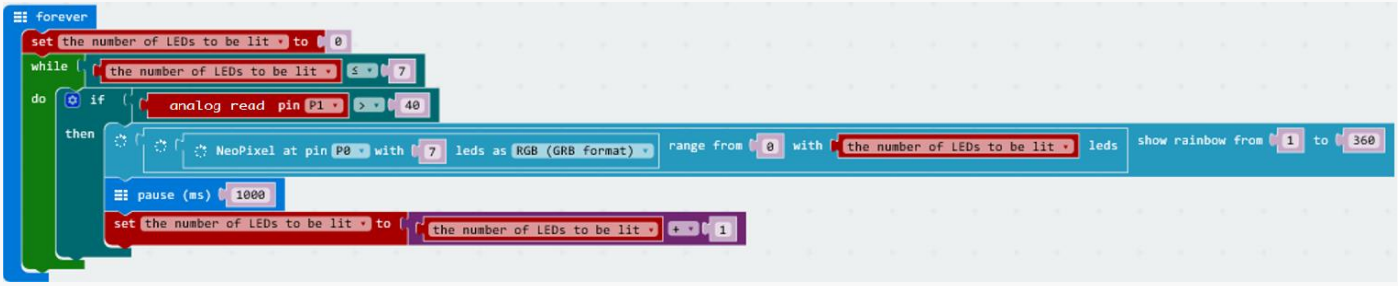


STEP 2: When the value exceeds 40 and the total number of lighted LEDs is less than 7, one more LED lights up. The sound sensor will be continuously detecting the value.

The flow chart below may be helpful in understanding the mechanism:



STEP 3：Put the condition "if the analog read P1 > 40" in the program that we build in the last part, we will have the final program as following:



Chapter 5: Become an expert

We have built so many funny works with different types of modules in previous chapters. You may now have a deeper understanding of the micro: bit. Is that enough for the exploration? The answer is doubtlessly NO! Hang on there and we will take another further step into the micro:bit.

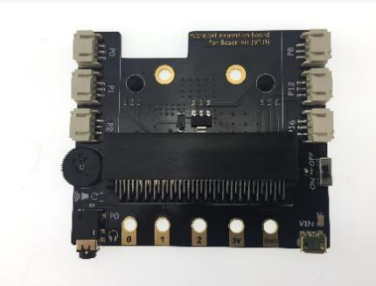
Project 1: Electronic Stabilizer

Have you heard of the accelerometer? The accelerometer measures the acceleration along 3 direction and it is now extremely popular in all kinds of electronic gadgets. For example, if we turn the phone or laptop 90 degrees, its page will also turn 90 degrees automatically, which saves a lot of time in setting up the direction. In this part, we are going to play with the accelerometer with the micro: bit!

Components list

1 × micro:bit

1 × Boson Expansion board

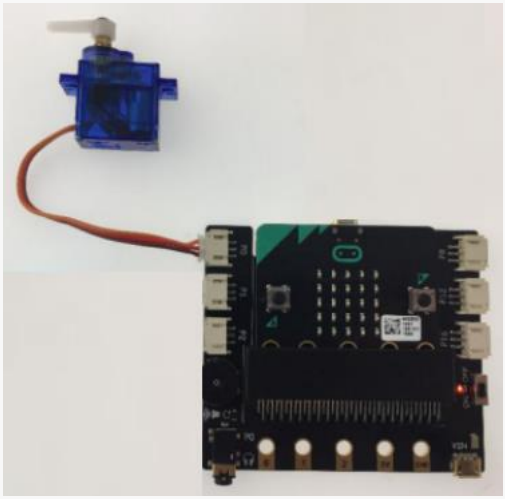


1 × servo module

1 × USB cable

Connection

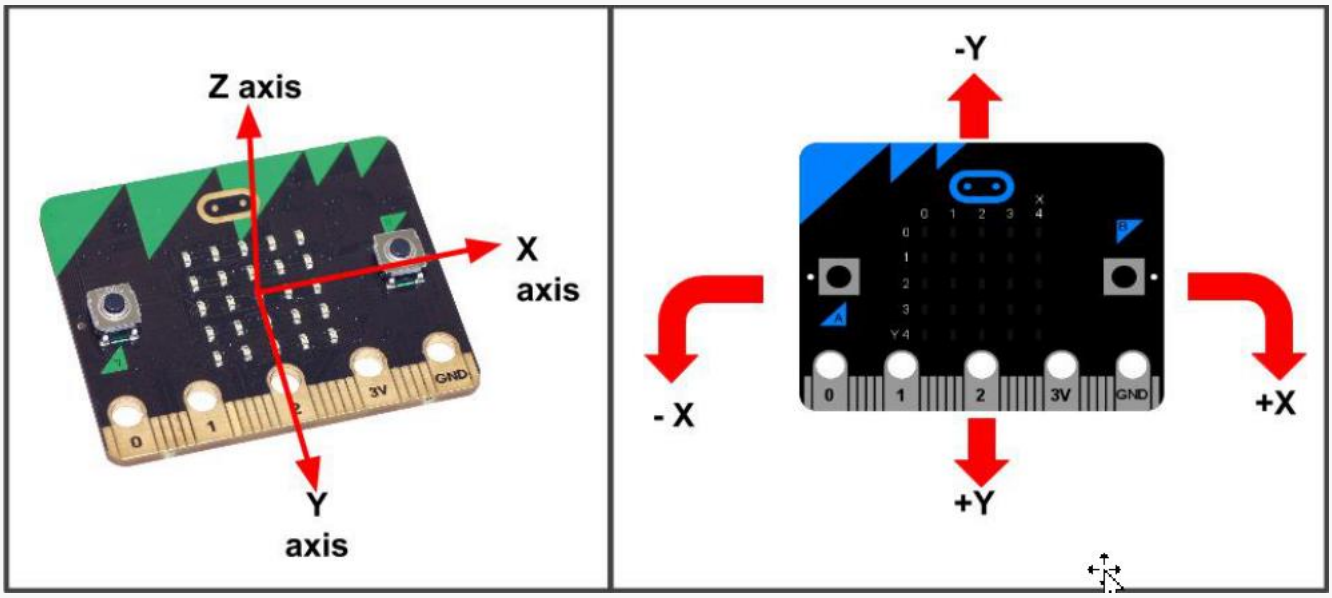
Connect the servo to P0



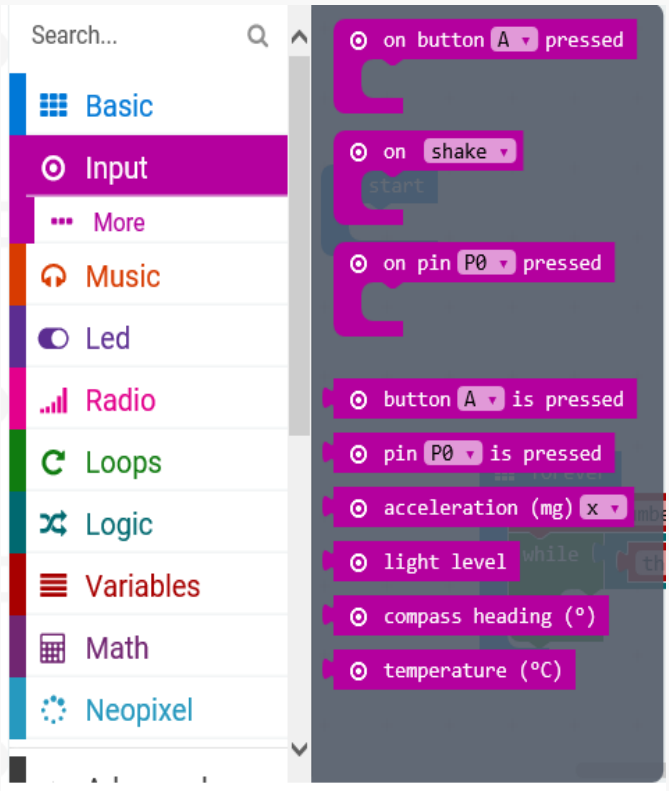
Program

Goal: Fix the servo on the micro: bit, servo will keep pointing upright regardless how we tile board.

Micro: bit detects the gesture automatically through the accelerometer. Gravitation force is also a form of acceleration ,and micro: bit can detect the gravitational force along X, Y, and Z directions, whereas X is along the left-right direction, Y is along the front-back direction, Z is perpendicular to the board and along the up-down direction.



The values of the X, Y, and Z axes can be achieve from the function “acceleration” under "Input".

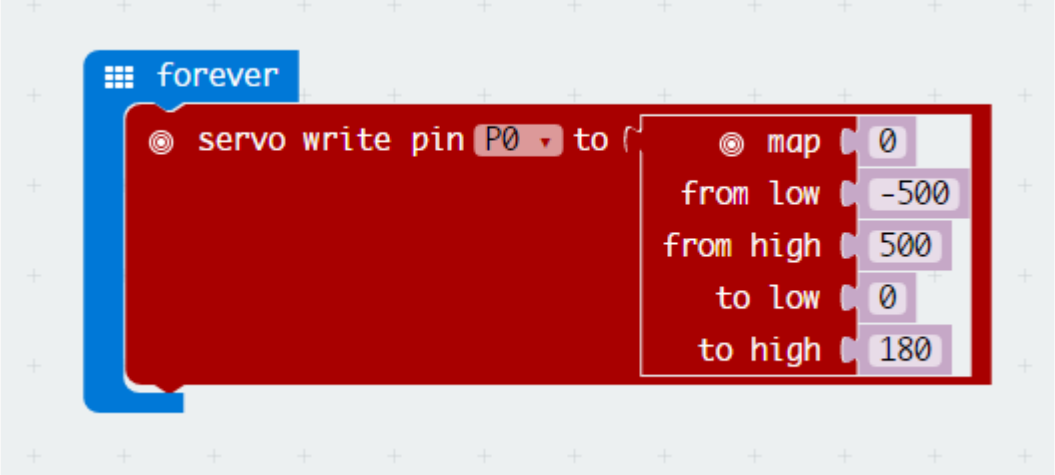


When the mainboard lays flat on the desktop, gravitational force goes along up-and-down direction, so that the values of X and Y axis are close to zero and the Z axis is around 1000 (mm / S2), which roughly equals to the gravitational acceleration.

When we tilt the board, the values of X and Y change. When it leans to the left, the value of X is negative or, conversely, positive.

STEP 1: When the board is tilted to the left, the accelerometer will produce a negative output value along the X direction. We will rotate the micro: bit toward 0 degree to compensate this angle and

vice versa. Since the output value of the accelerometer ranges between -2048 and 2048, and in our case, the maximum value it will reach is -500 and 500. we can make the -500 ~ 500 correspond to 0 ~ 180 degrees of the servo through the function “map”.



STEP 2: Fix the servo and expansion board with tape so that these two remain relatively still. Note that the orientation of the servo needs to be the same as that of the USB port so that the gear shaft is perpendicular to the X axis and the degree of rotating of the servo is complementary to the degree of tilting of micro: bit.

Exercises

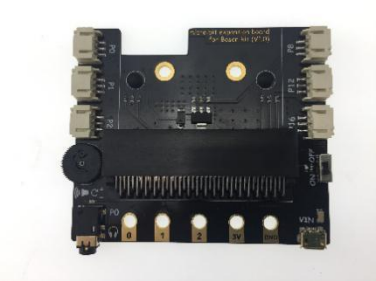
Try building a compass using the electronic compass built into micro: bit.

Project 2: DJ panel

Components list

1 × micro:bit

1 × Boson Expansion board



1 × RGB LED strip

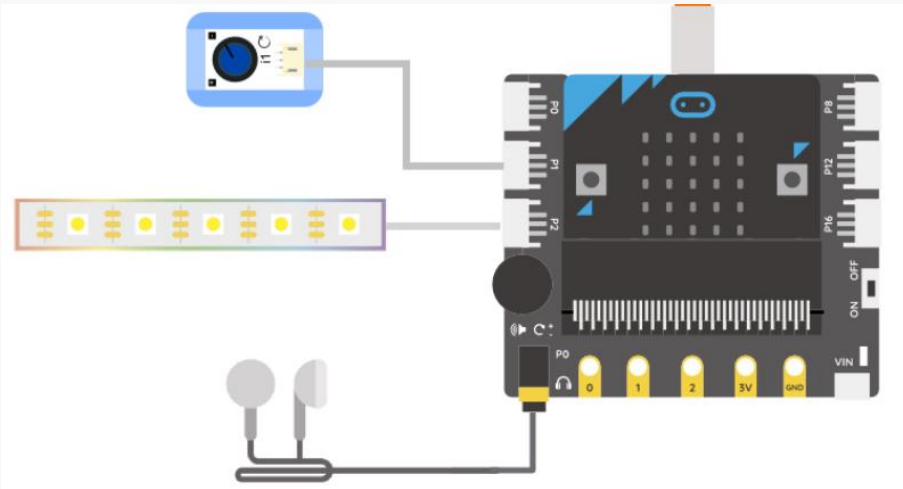
1 × knob module

1 × USB cable

Connection

Connect the knob module to P1;

Connect the RGB LED strip to P2.



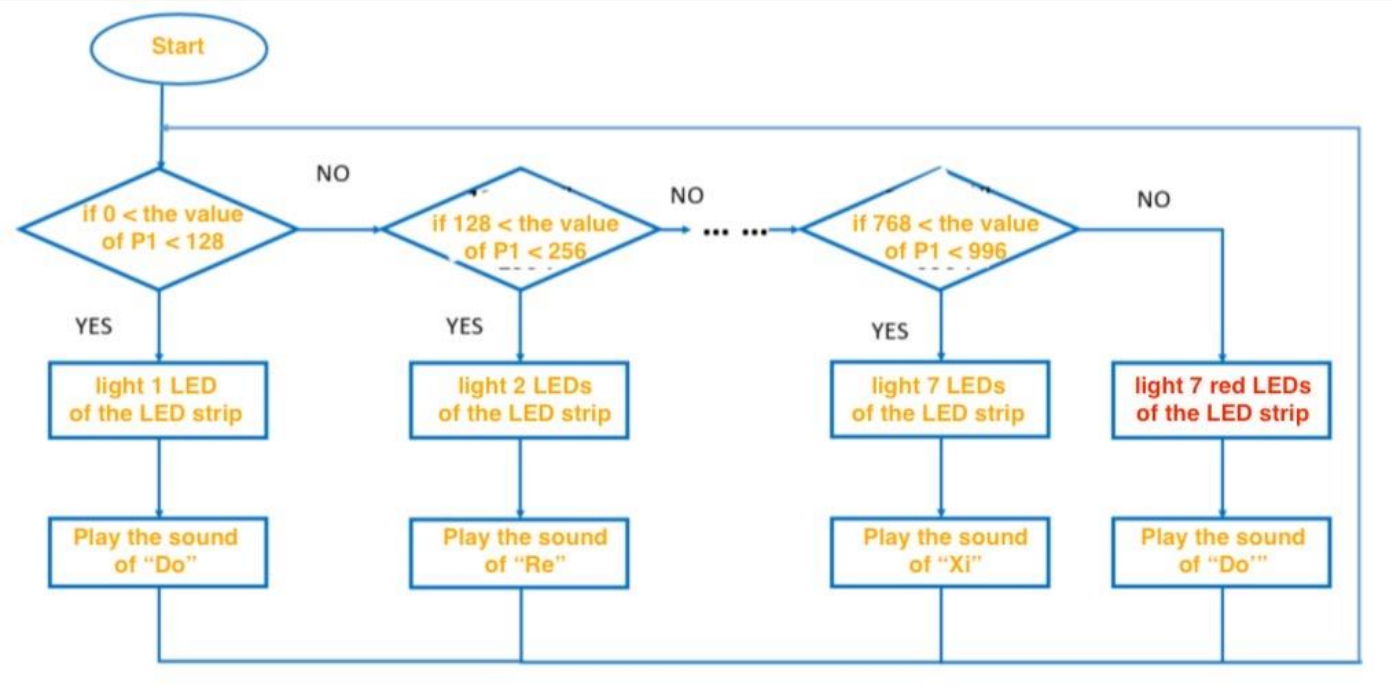
Program

Goal: In this project, the smart music box and the LED strip be combined to realize the music performance with the LED strip when manually adjusting the rotation angle of the knob.

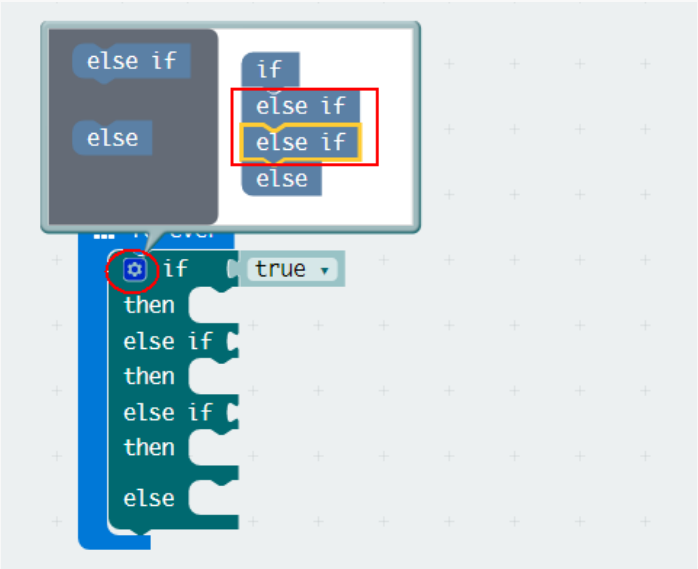
When the knob is turned by hand, the value of P1 changes from 0 to 1023 and is divided into 8 small ranges, corresponding to one octave "Do ~ Do'" and the number of LED(s) to be lighted .

The input value of P1	Sound	The LED strip
0-128	Do	Light one LED
128-256	Re	Light two LEDs
256-384	Mi	Light three LEDs
384-512	Fa	Light four LEDs
512-640	So	Light five LEDs
640-768	La	Light six LEDs
768-996	Ti	Lights seven LEDs
996-1023	Do'	Light seven LEDs in red

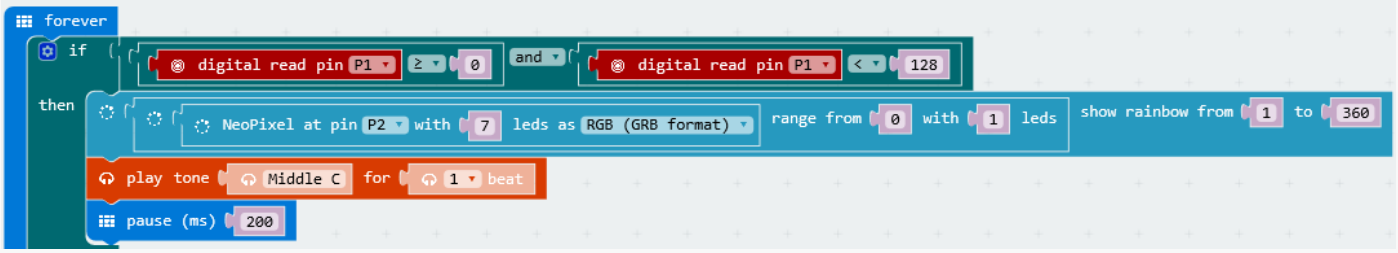
STEP 1: The logic diagram below will help you to build a general idea about how to realize a DJ PANEL.



STEP 2: In this step, we will learn how to edit the "if else" function. Firstly, place the "if else" from "Logic" under "forever" loop. As we may see from below that there lies a blue gear-like icon(circled in red as below) 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" on the right side. Following diagram will help you to finish this step.

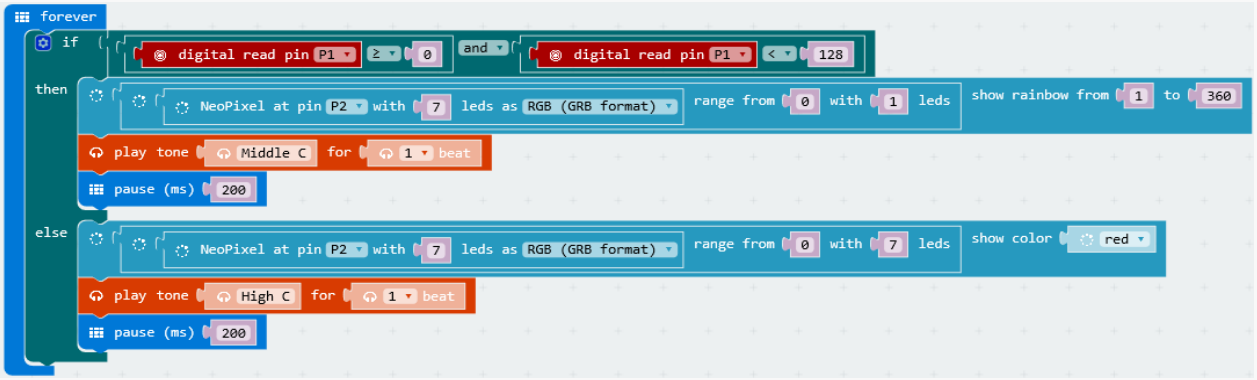


STEP 3: In this step, the "and" function and the operator "<" under "Logic" will be applied to judge the input value range of P1. Functions from "NeoPixel" will be used to control the lighting effect of the GRB LED strip. And functions from "Music" and "pause (ms)" from "Basic" will be taken to control the music. The following is the finished program in this step:



STEP 4: Since there are only 7 LEDs, so the eighth sound "do'" can be expressed by a different lighting effect.

STEP 5: Combine all above listed functions altogether, the final program goes as below:



Exercises

You must have noticed that there is the 5×5 LED matrix on the micro:bit. Why do not make a full use of it! Try to use it to display corresponding music notes when they have being played.

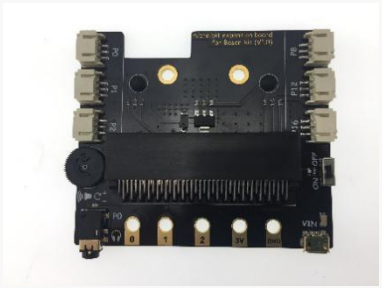
Project 3: Remote Doorbell

For those who have a doorbell in their homes, it may be quite a common issue that sometimes the ring is not loud enough to travel over rooms. To solve this problem, a wireless alarm may be helpful. In the chapter, we will learn how to make the doorbell to control a remote LED that we can carry around.

Component list

2 × micro:bit

1 × Boson Expansion board



1 × LED module

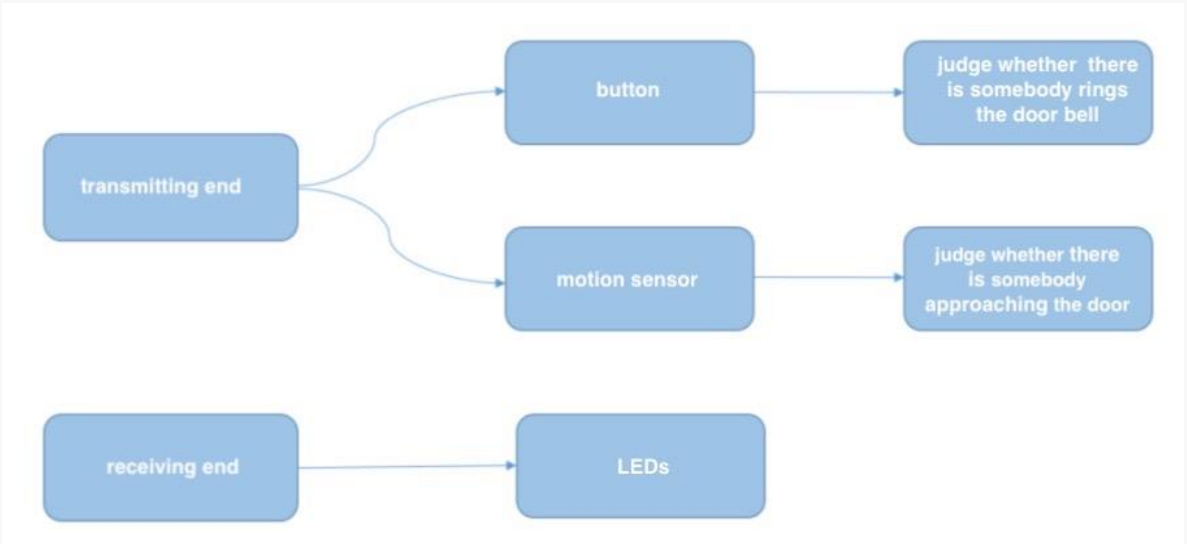
1 × motion sensor module

1 × knob module

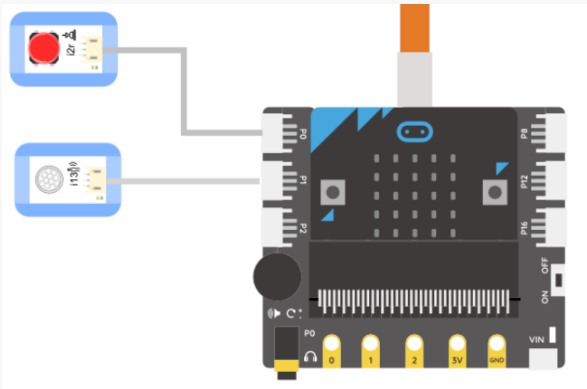
1 × USB cable

Connection

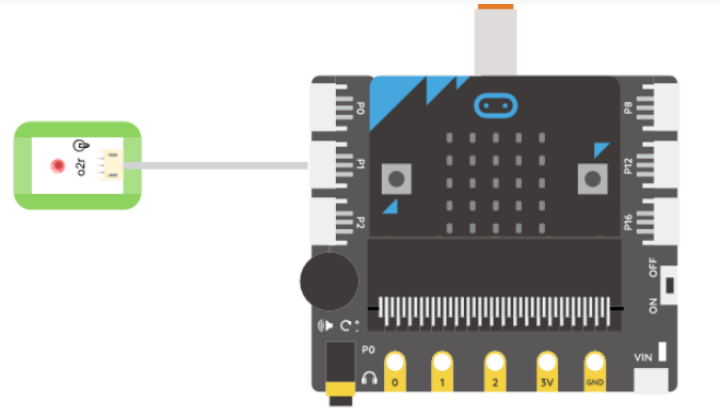
We will need two micro: bit in this project, one is attached to the door bell and sends out message (transmitting end), the other receives the signal and controls the LED indicator (receiving end). The micro: bit communicates over radio.



Connection diagram of micro: bit transmitter:



Connection diagram of micro: bit receiver:

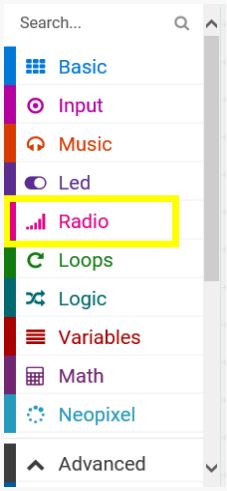


Program

Task1: Program the transmitter

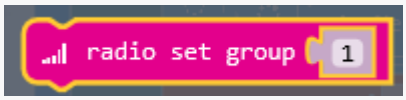
STEP 1: We have been quite familiar with the button module and the motion sensor, however, how can we send out their status through over radio? To achieve this, we will learn how exactly the radio works in the following session.

The "Radio" function can be found on the left side of the MakeCode window. .



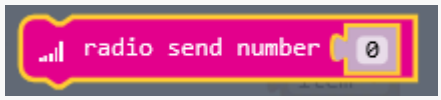
We need to first understand the following 3 the most commonly used functions under “Radio”:

“set group”：



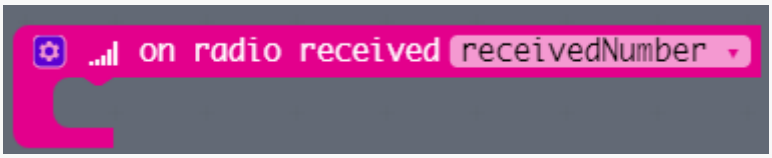
Function description: There are a total of 255 different radio channels. The transmitter and the receiver should be set in the same group to be able to communicate.

“send out number”：



Function description: send out a number over radio.

“on radio received”：

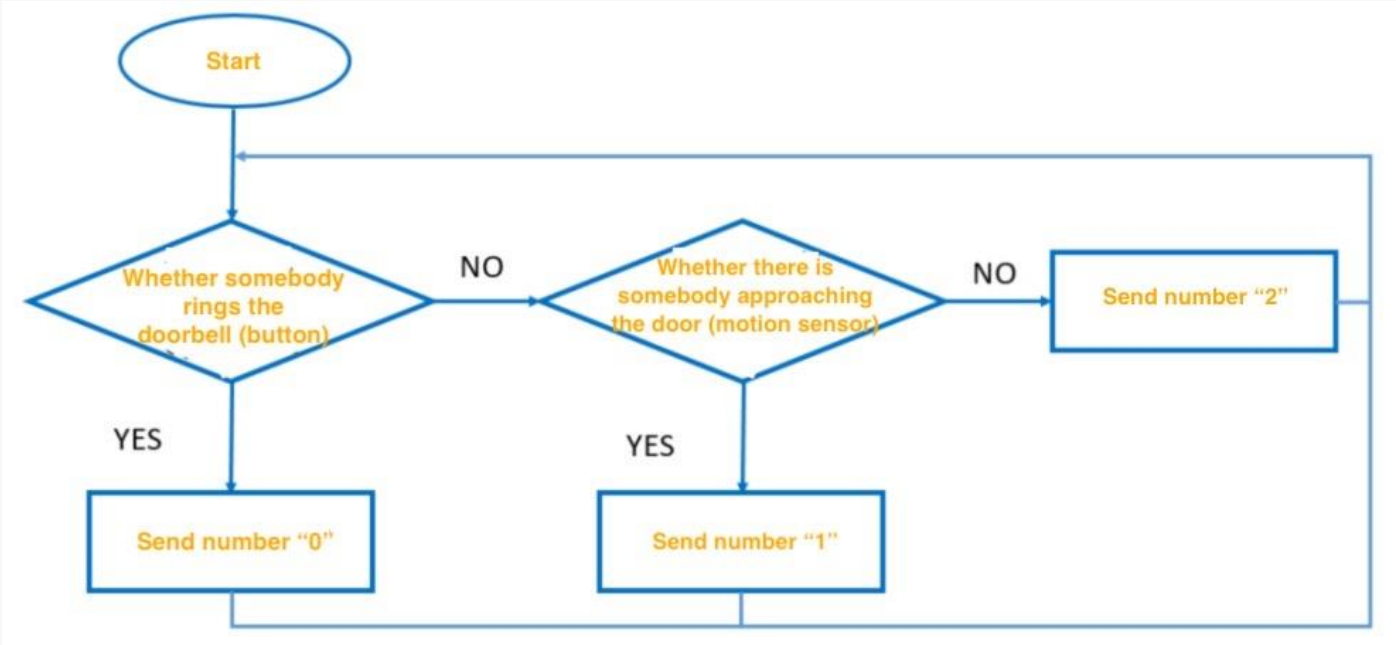


Function description: Upon a number is received, it will execute the program inside the “on radio received” loop.

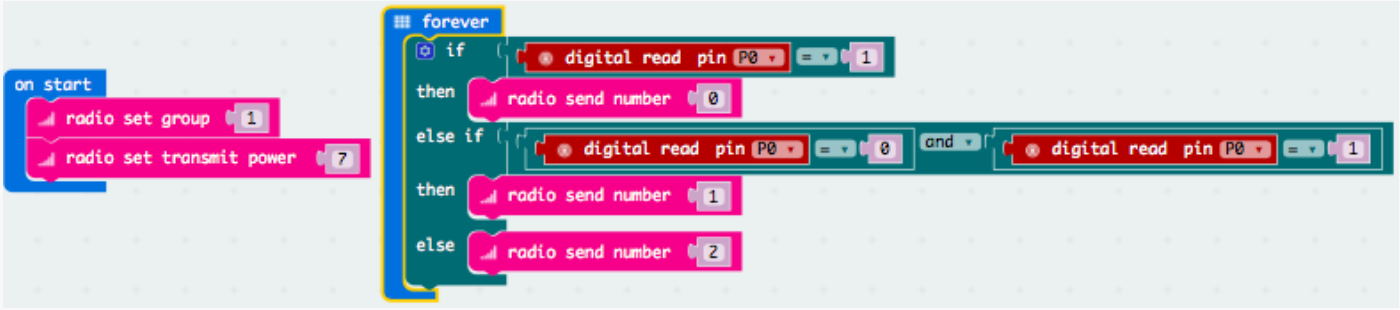
We will also need to program the receiver to make corresponding reaction based on the received number.

Notice: the variable "receivedNumber" inside the “on radio received” function represents the received number.

STEP 2: After knowing how to use the radio function, it is the time to use input modules to trigger the radio signal. To make it a bit more challenging, we will try to program the micro:bit send out a message when someone rings the bell, and send out another message when there is just someone passing by. To achieve this, we will need the button module and a motion sensor. The flow chart below gives you an idea of how to write the program.

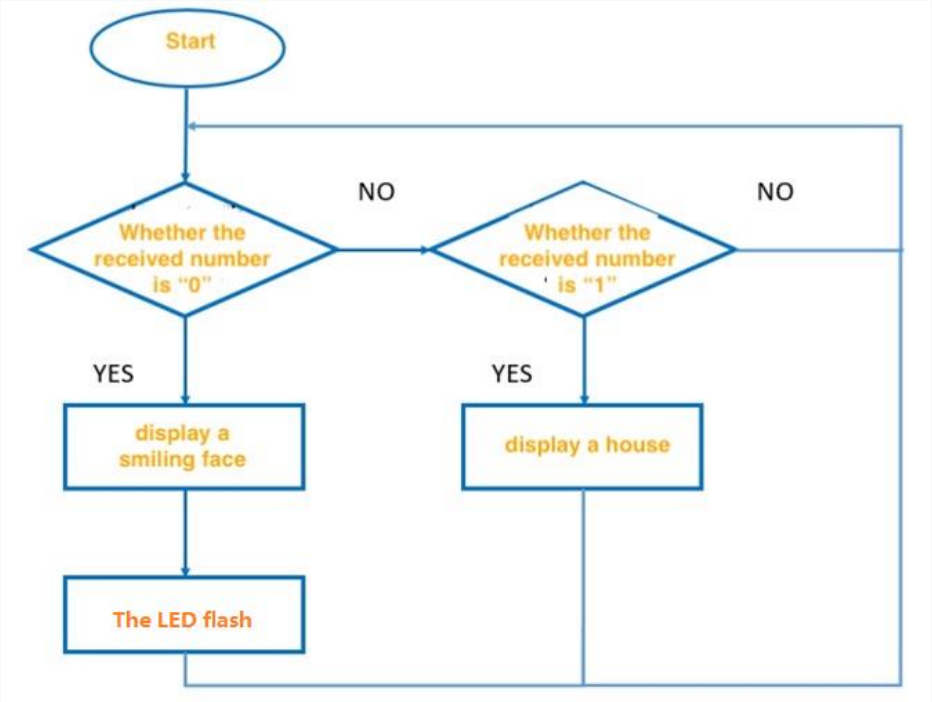


STEP 3: Put the function blocks together. The complete program will be looked like as following.

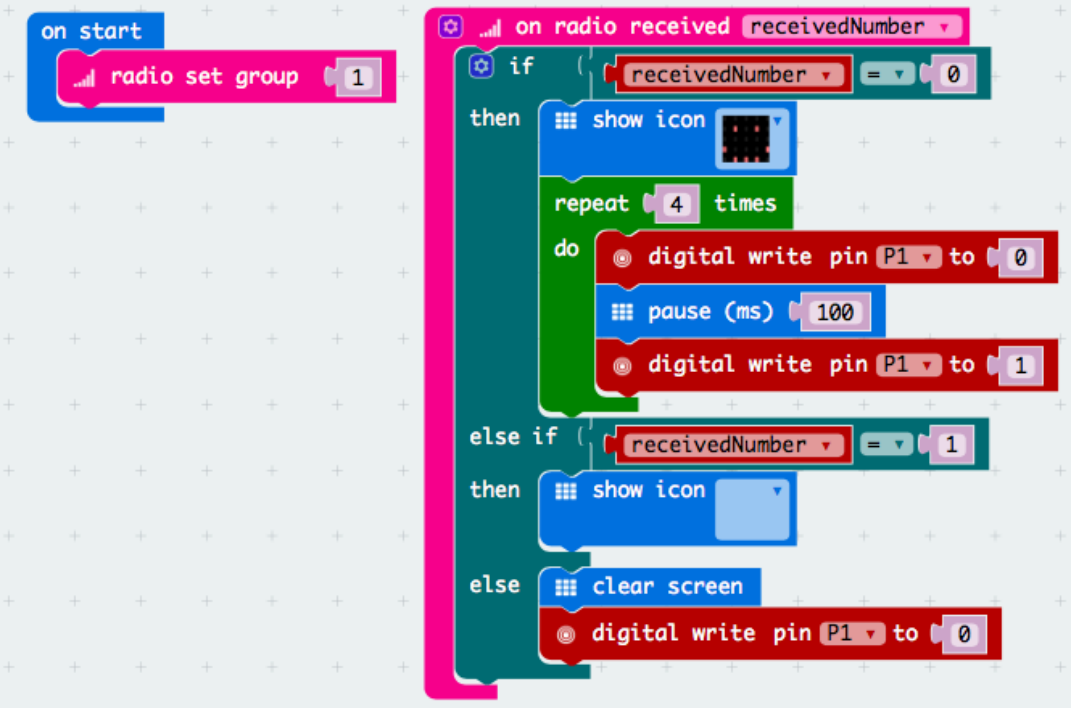


Task 2: Program the receiver

STEP 1: The receiver is able to execute different action based on the received signal. The flow chart below will help you in building the program.



STEP 3: Put the function blocks together. The complete program will be as following.



Exercise

Add a “Please wait” notice to the doorbell when someone rings the bell.

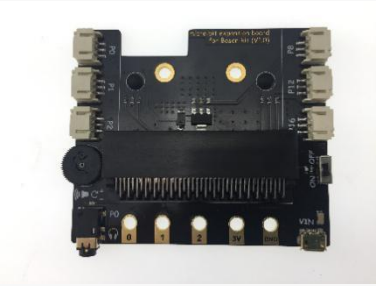
Project 4：Escape the maze

Micro:bit is trapped inside the maze with a time bomb! Poor micro:bit cannot move by itself. However, it sends out signal through the LED panel to lead the way. We need to help him escape the maze before the bomb explodes!

Components list

1 × micro:Bit

1 × Boson Expansion board

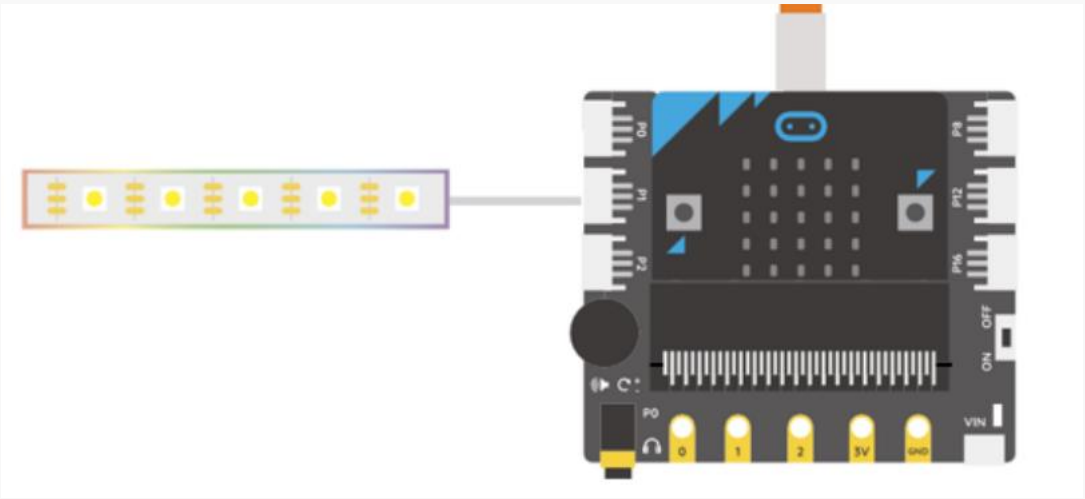


1 × RGB LED strip

1 × USB cable

Connection

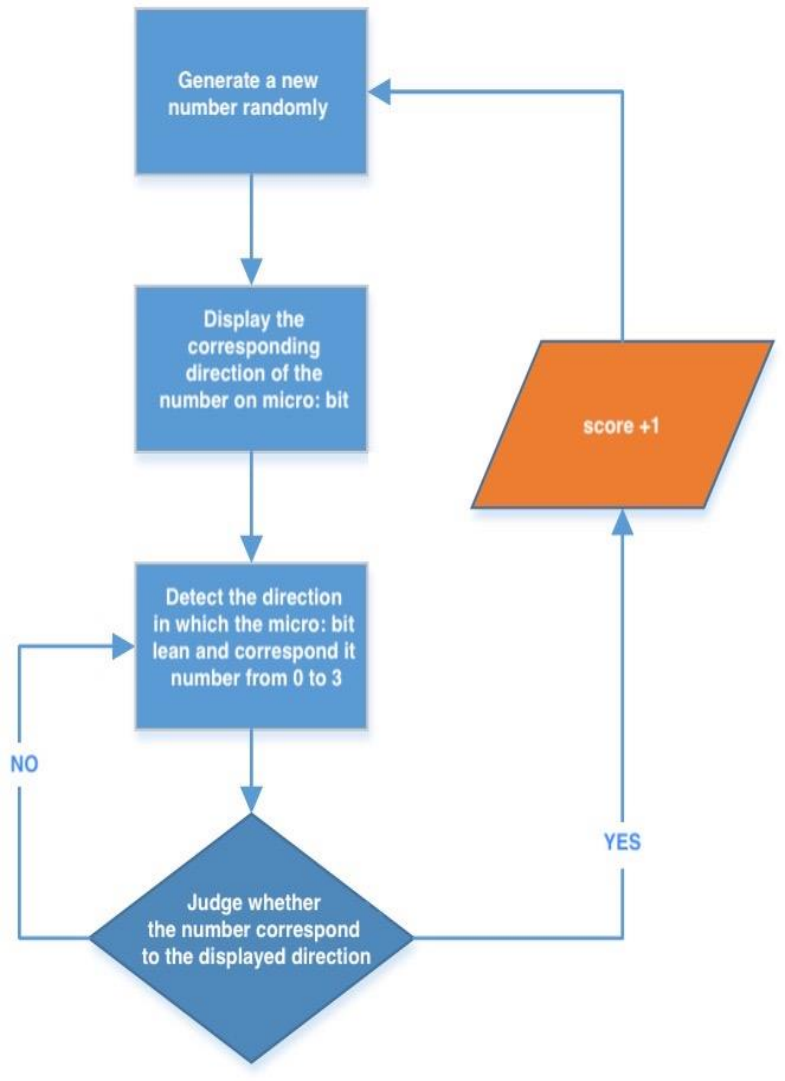
Connect the RGB LED strip to P1.



Program

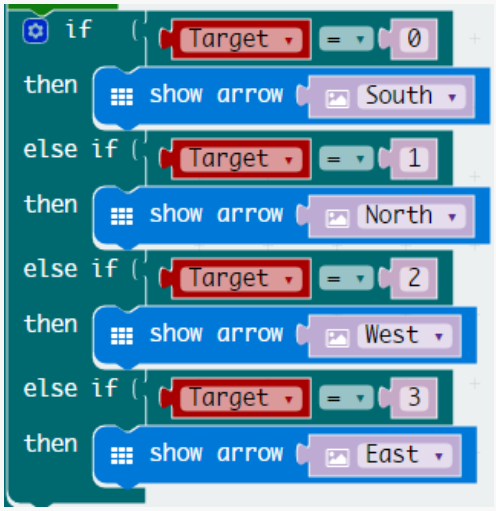
STEP 1: First, we will need to let micro:bit randomly generate some movements to simulate the maze. Also, micro:bit should be able to know which way it tilts. If the two directions match, 1 point will be added to the final score, which means that we are 1 step closer from escaping the maze.

The following flow chart may be helpful in explaining how the program works.

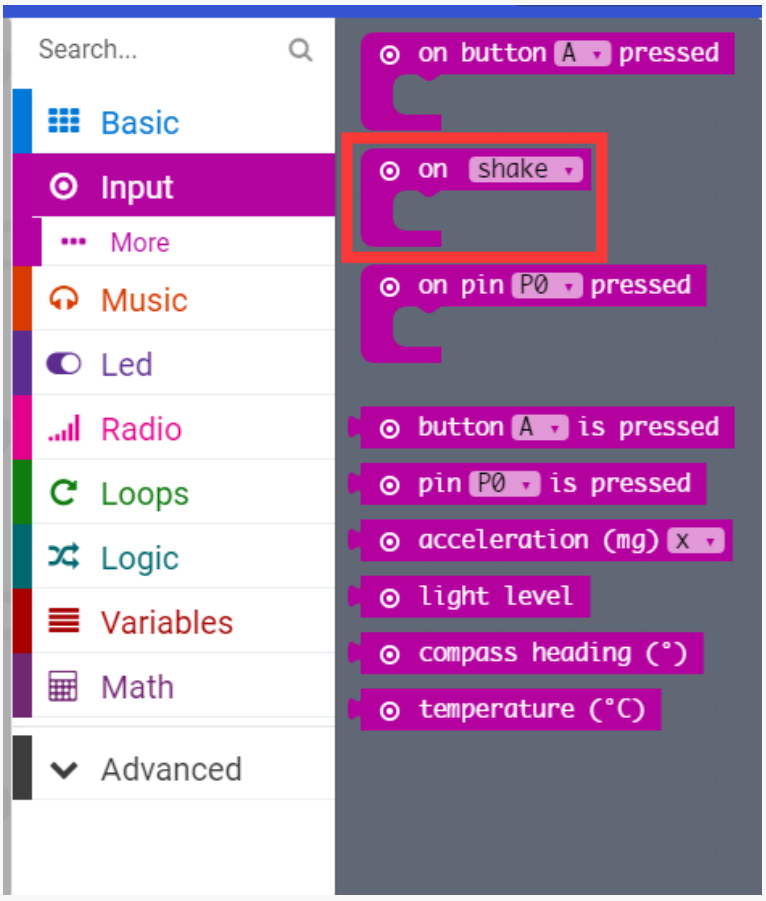


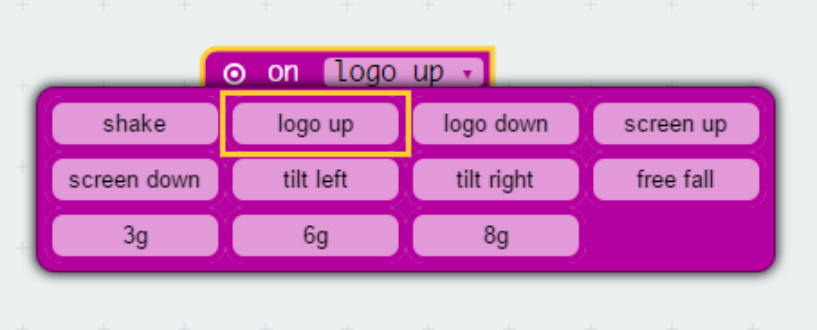
We will use number 0 ~ 3 to represent four target directions respectively. The function "pick random" under "Math" help us to get a random number from 0 ~ 3.

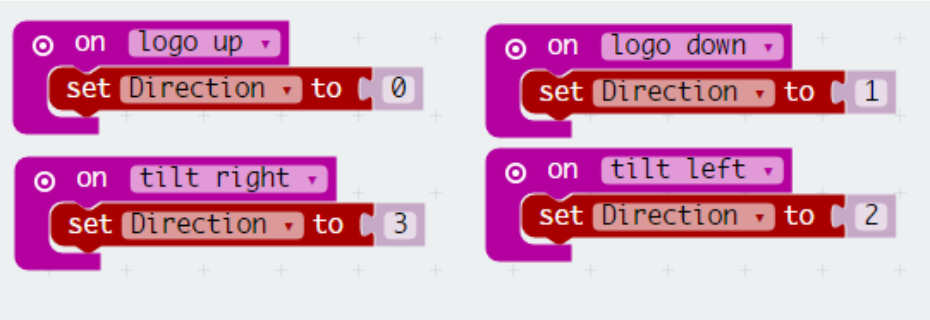
STEP 2: Show the arrow image on the LED panel. The arrow should correspond to the randomly generated target direction we got from above. Again, the "if-else" function will be our choice to do the job.



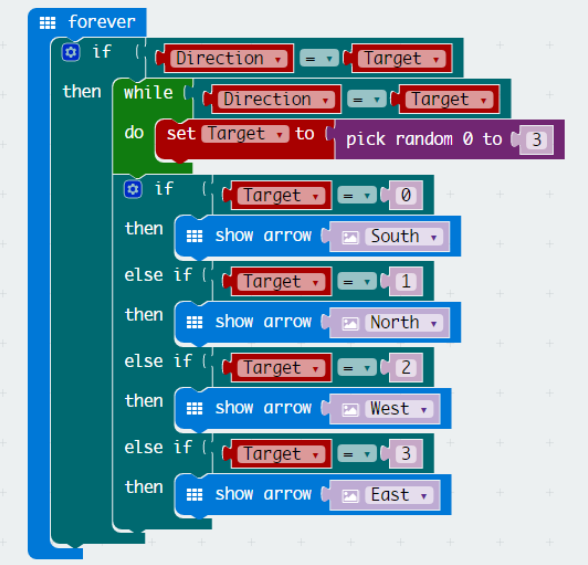
STEP 3: Use the "on xxx gesture" function under "input" to sense which direction the micro: bit tilts. The number inside the function loop represents four different directions.







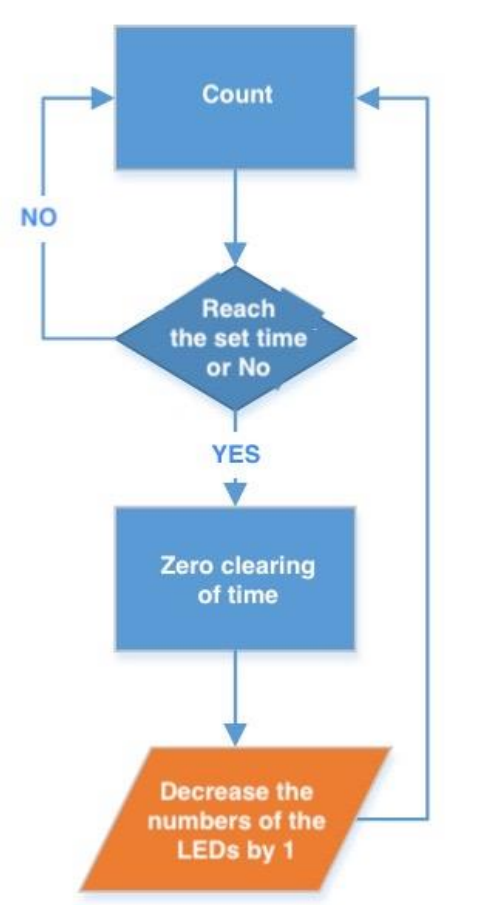
STEP 4: Put everything we mentioned above inside an "if-then" loop to compare if the two directions match. Also, don't forget the "forever" loop to keep the game always on going.



Note: it will be helpful to add a "While" function inside the loop, so as to avoid identical directions appear in a row.

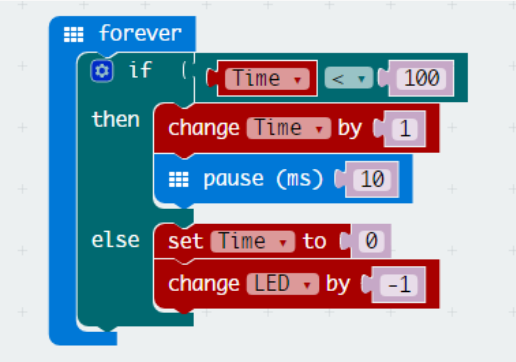
STEP 5: it's the time to add the bomb. The bomb will start ticking as soon as the game begins. Also, we will use the RGB LED strip to simulate the fuse. With the time running, the LEDs on the fuse will go off one after another in a steady pace. If micro:bit cannot make it to escape the maze before the fuse burning out, the game is over.

The flow chart below shows how to set the LED go one by one with the time going.

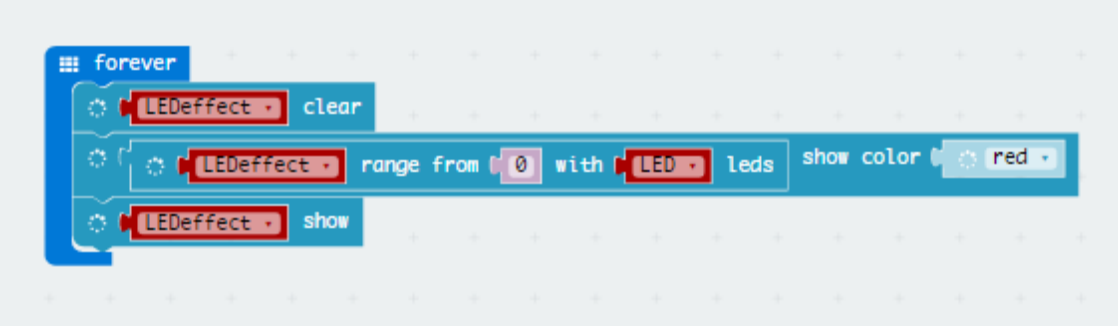


We will be explaining the functions that we need to achieve it.

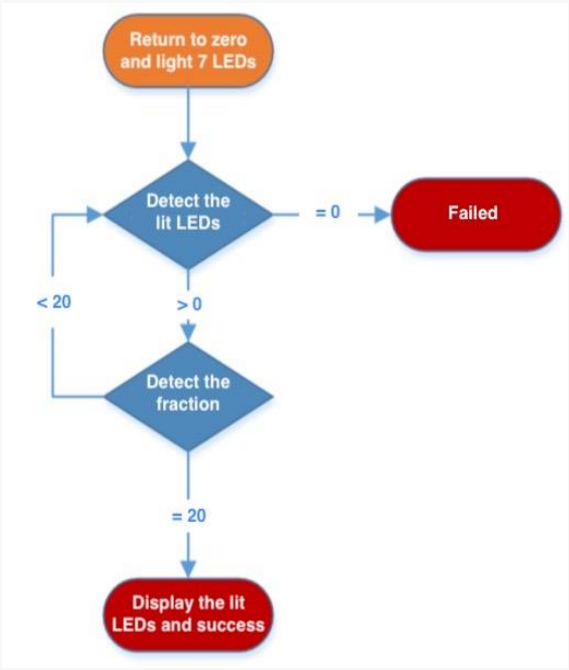
Note: Micro: bit needs extra time to process the data, so the clock will run a bit slower than what we set.



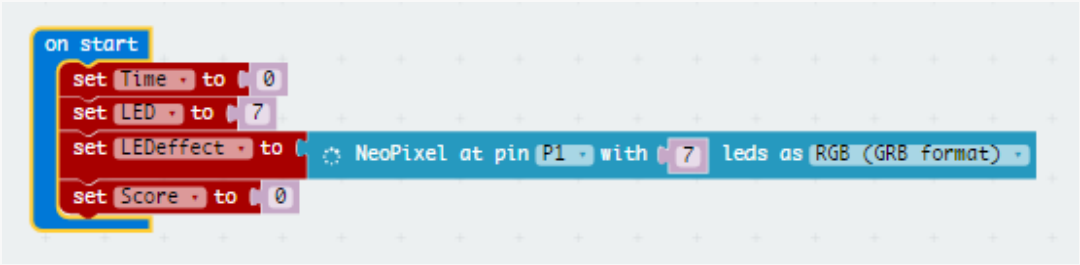
The following program is used to set how many LED lights up.



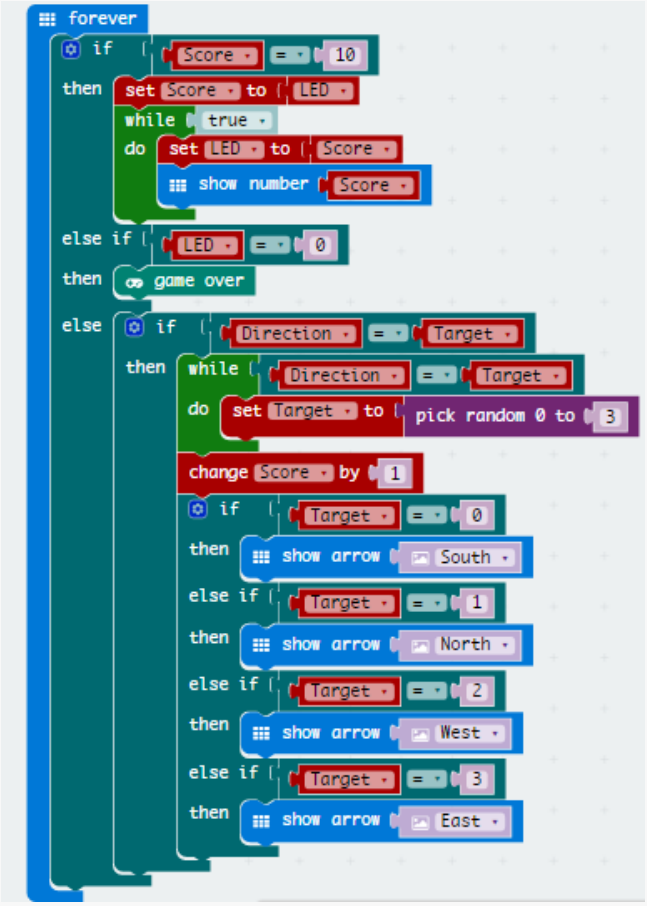
STEP 6: A scoring mechanism will be needed to tell us whether we made it to escape. The following flow chat may give you an idea of how it works..



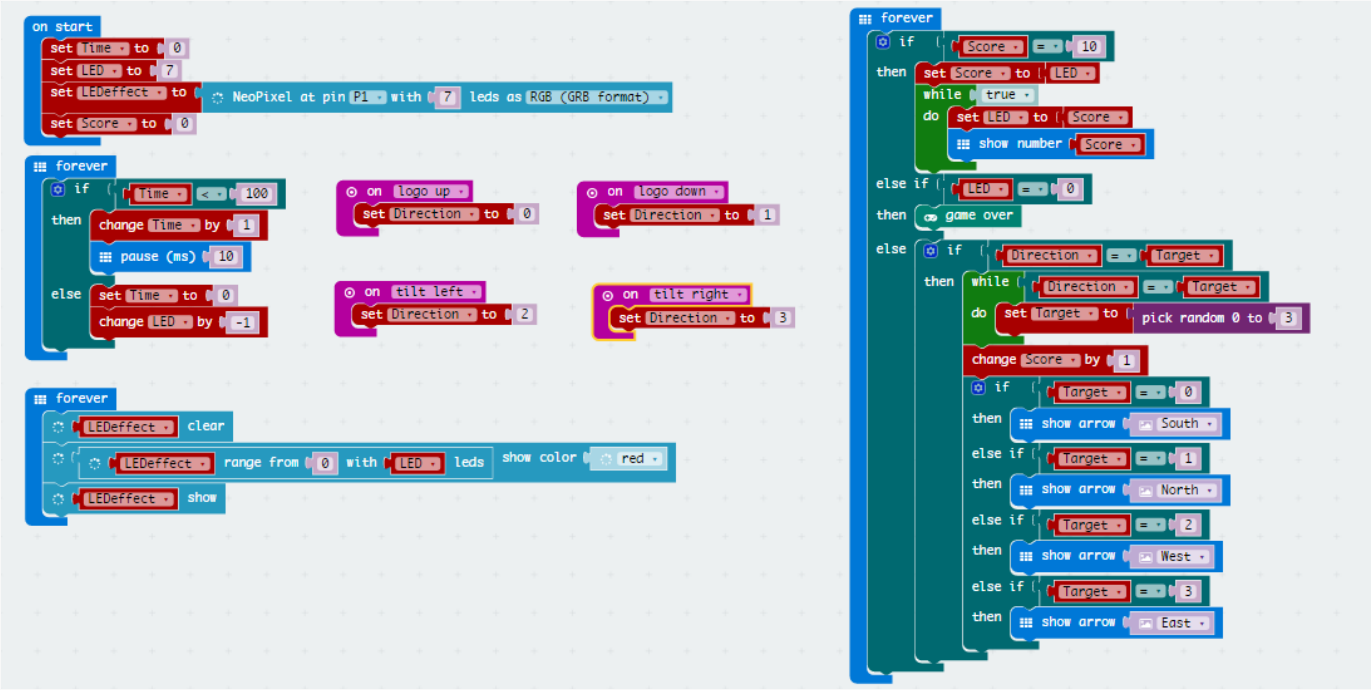
Remember to reset the score at the beginning of the program.



STEP 7: Now, integrate the scoring function to the main program.



STEP 8: Here is how the final program looks like



Exercise

Try to make the clock tick faster to increase the difficulty;

Program the LEDs to make them flash at a pace based on the remaining time;

Unintuitive challenge! Set different directions in pairs and see if you can still finish the game. For example, you will need to tile top to match a left arrow, tile bottom to match a right arrow.

Your journey of micro: bit wouldn’t stop here. If you would like to share your ideas with us, please visit the DFRobot blog and share your idea with us!

Welcome to the DFRobot blog!

www.dfrobot.com/blog-tag-microbit.html