1. Introduction

ELECFREAKS micro:bit basic kit is an entry-level kit. We carefully select 5 most common electric bricks, which can be easily connected to micro:bit via basic:bit. This kit can help students learn how to build their micro:bit projects quickly. And these electric bricks can be driven by Makecode defaulted blocks directly. You don't have to add any extra package. It is helpful for students to understand the usage of electric bricks from its principle.

1.1. Components

Module | Quantity :-: | :-: micro:bit |1 basic:bit|1 LED Module|1 Crash Sensor|1 Potentiometer| 1 Servo|1 ADKey|1 USB Cable|1 Crystal Battery Box|1 Basic:kit Manual Book|1

1.2. FAQ

2. Basic:bit Introduction

2.1. Introduction

Basic:bit is a basic breakout board of micro:bit. It carries a buzzer and three groups of GVS pins(P0/P1/P2) on board. Each group of GVS pins separately leads out the IO port, 3V port and GND port on micro:bit. Small size with simple structure, it is quite enough for you to complete 98% micro:bit projects.

2.2. Features

- Small size with smart design, it is a perfect match for your micro:bit.
- Easily connected to micro:bit and lead out its power and IO ports with screws only.
- Lead out P0,P1,P2,3V,GND port on micro:bit board.
- Allow you to complete 98% micro:bit projects with three IO ports.
- The switch on board controls the ON/OFF status of buzzer.

2.3. Parameter

- Voltage: 3.3V （Powered by micro:bit）
- IO Ports: 3
- Dimension: 51.7mm x 29.1 mm

2.4. Packing List

- 1 x basic:bit
- 7 x screws（5 is enough）

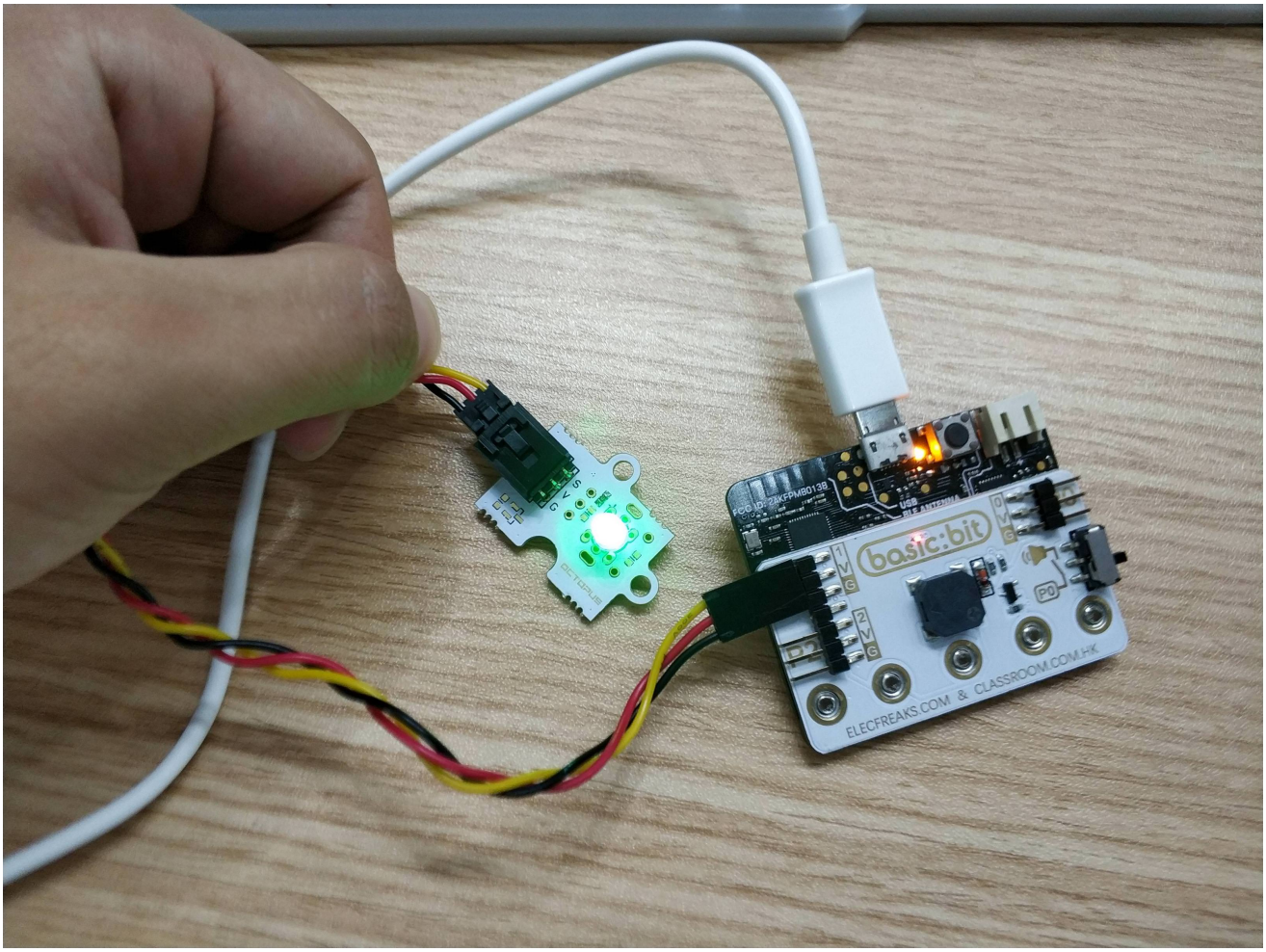
2.5. Case Study

Material Needed:

- 1 x micro:bit

- 1 x basic:bit
- 1 x Octopus LED Module
- 1 x USB Cable
- 1 x GVS wire（Or Jumper Wire)

Assembly

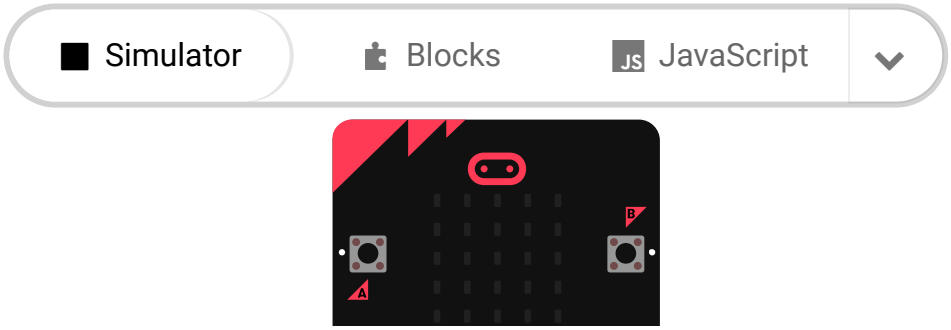


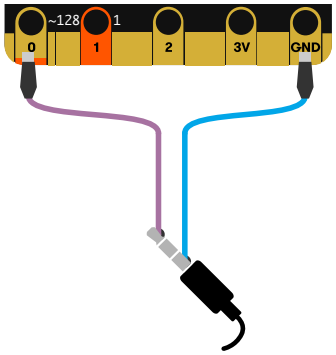
Fix your basic:bit onto micro:bit with screws and nuts. Connect Octopus LED module to P1 port on your basic:bit. Then slide the switch to the speaker end.

Example Code

You can see the whole program via the link below:
https://makecode.microbit.org/_AvW79jCcU2XA

Or you can download the program directly from the page below:





Final Effect

After saving the above code into micro:bit, the buzzer starts to play the music again and again. At the same time, Octopus LED module twinkles with 1 second space.

2.6. Documents

3. case 01 Light Controller

3.1. Our Goal

- Use basic kit to control light.

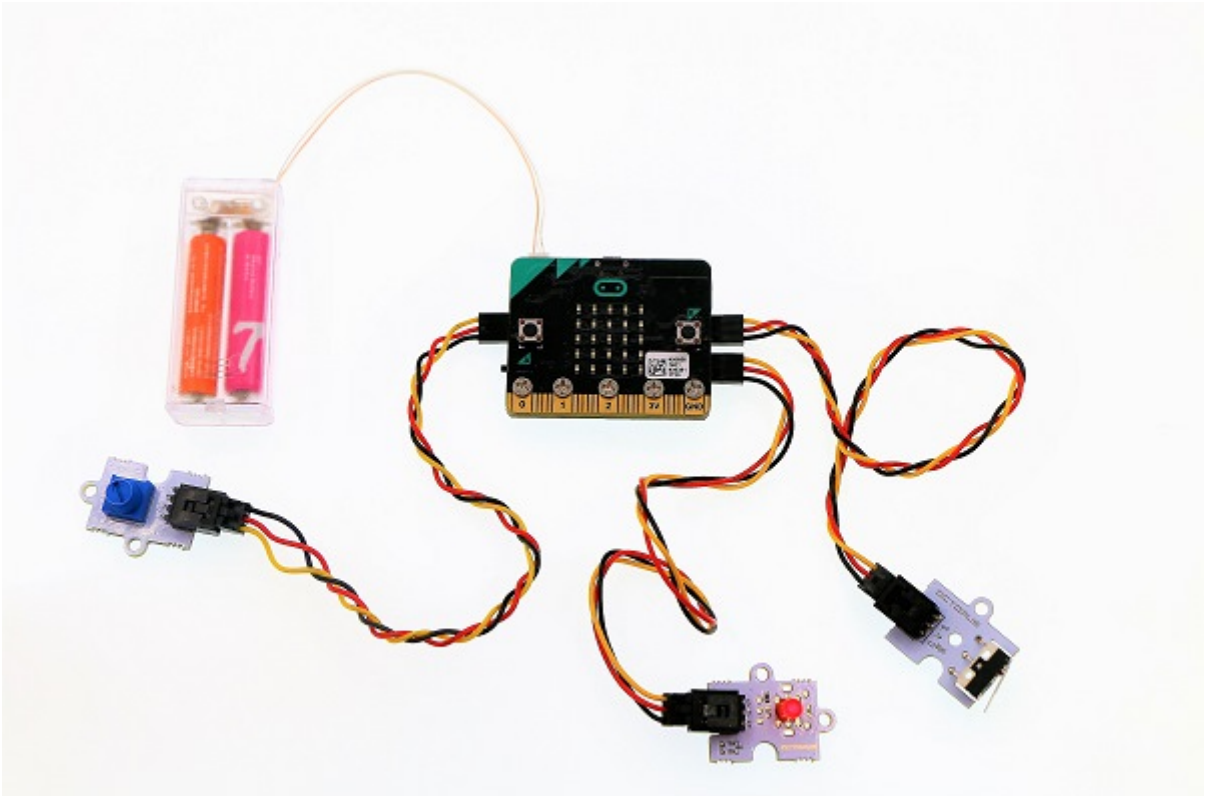
3.2. Material

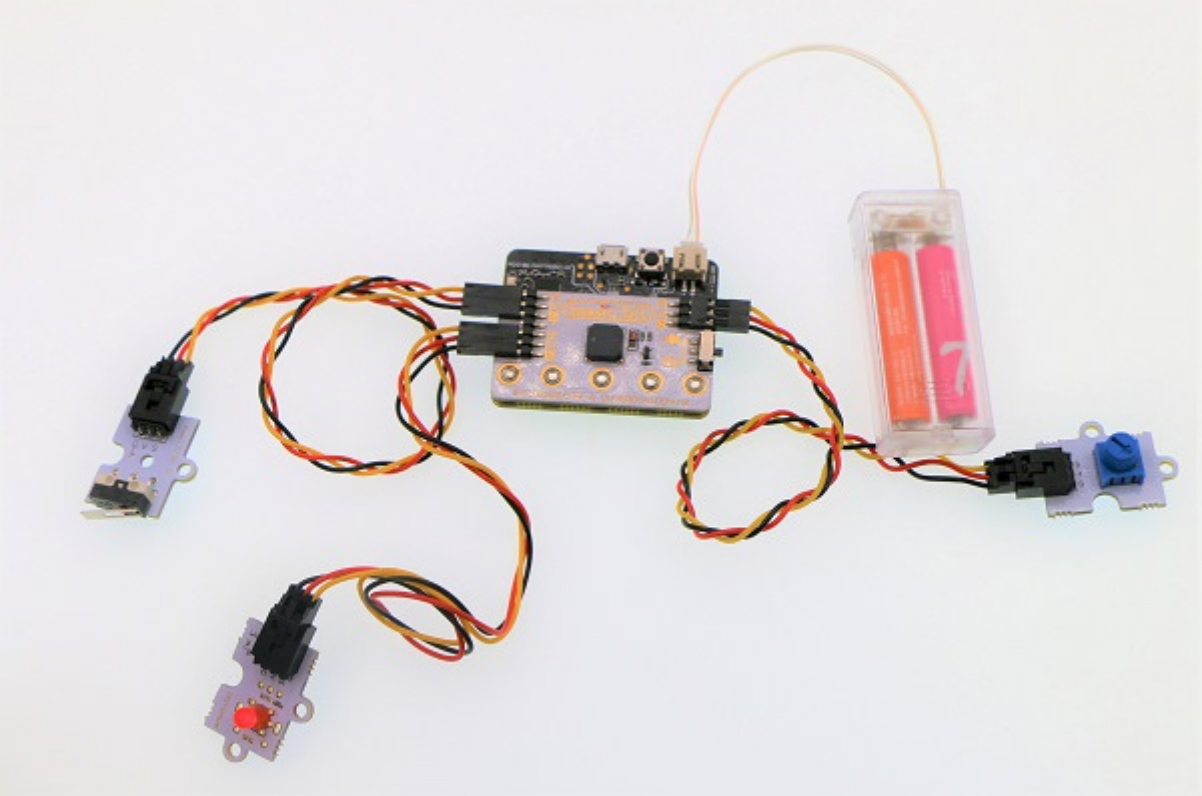
- 1 x basic kit

3.3. Background Knowledge

3.4. Hardware Connection

- Connect the crash sensor to P1 port on basic:bit, connect an LED module to P2 and connect a potentiometer module to P0, just like the picture showed below.





- **Note**：The switch on basic:bit must be shifted to P0 end, or the buzzer onboard will be connected to P0 port.

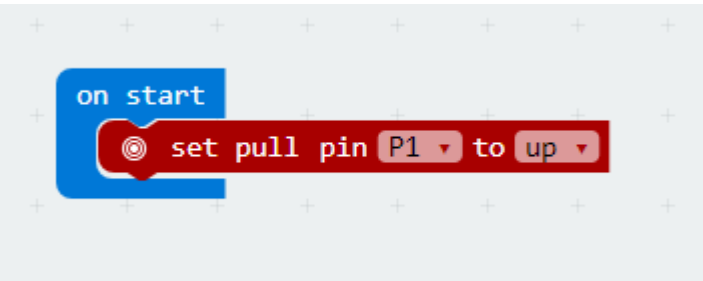
3.5. Software

- Microsoft Makecode

3.6. Programming

Step 1

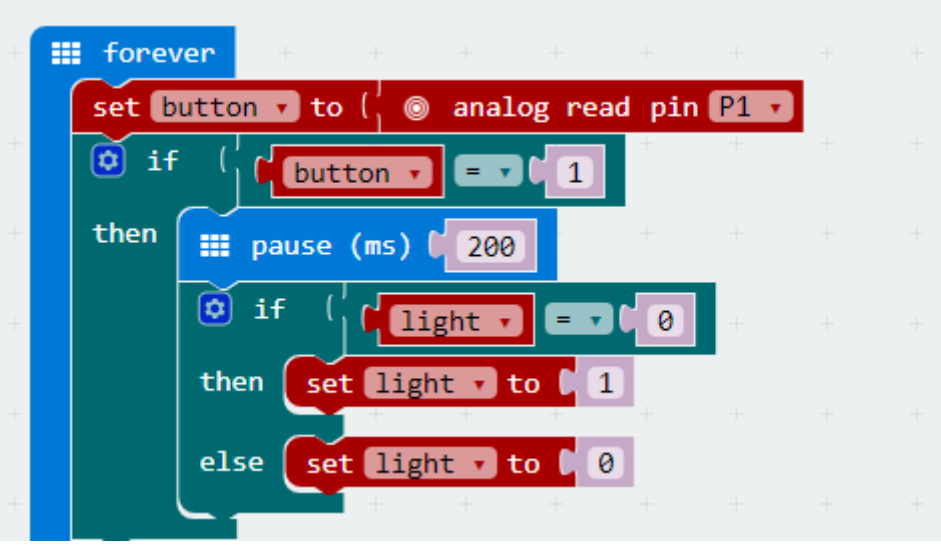
- When start, set a pull-up to P1 port to initialize the crash sensor.



Step 2

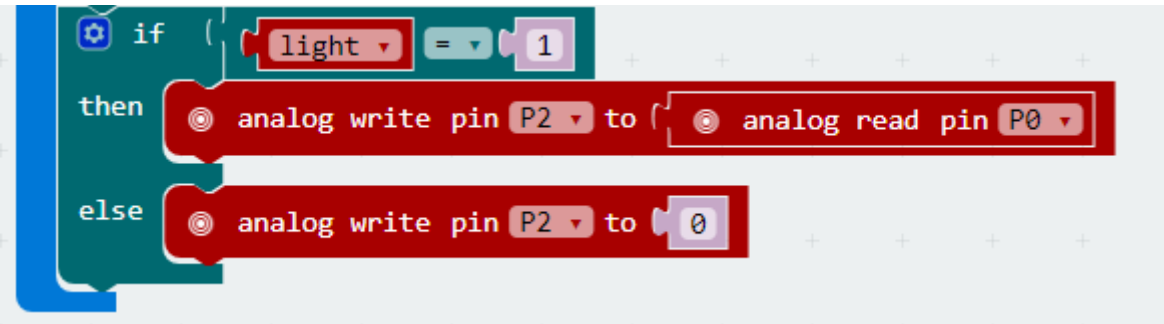
- Read the return value of P1 port in digital way, and assign it to variable `button` to obtain the status of crash sensor.

- Judge if `button` equals to 1. If `button` is equal to 1, it means the crash sensor is pressed down and we have to delay time and debounce.
- Judge the variable `light`. If the value of `light` is 0, it means the light is turned off, and we set `light` to 1. Or else, we set it to 0.



Step 3

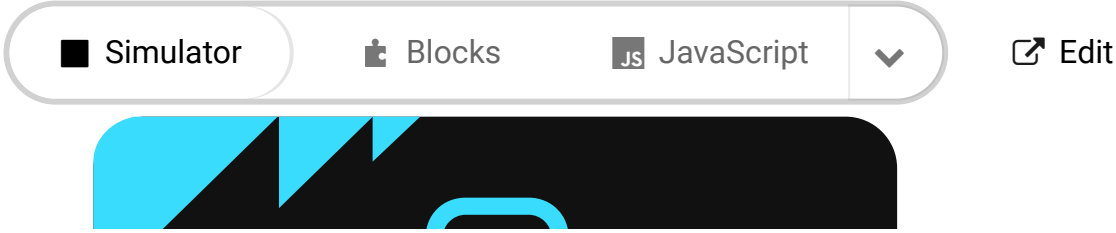
- Judge the variable `light`. If `light` is equal to 1, it means the light is turned on. Then read the value of P0 port(i.e. the return value of the potentiometer) in analog way, while writing it to P2 port with the same method. Now, this value becomes the parameter of LED module.
- If `light` isn't equal to 1, it means the light is turned off. Then analog write P2 to 0 and turn off LED.

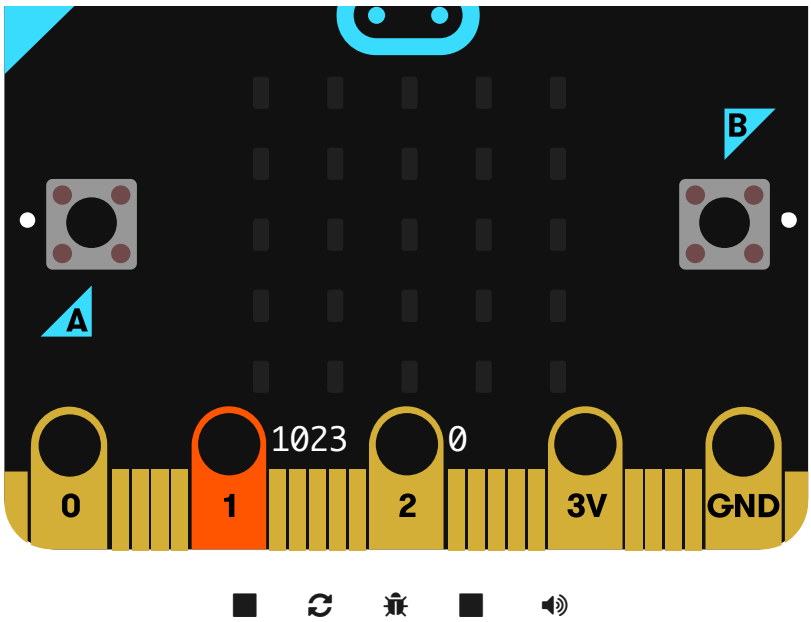


Program

The link of the whole program: https://makecode.microbit.org/_DL87DfRpaL7z

You can also check the program from the page below.





3.7. Result

- Press the crash sensor for once, the LED module is turned on; press again, then it is turned off. When the LED is turned on, we can adjust the brightness of LED by rotating the knob on the potentiometer.

3.8. Think

- Why we need to delay time and debounce ?

3.9. FAQ

3.10. Relative Readings

4. case 02 Morse Code

4.1. Our Goal

- Use basic kit to complete morse code.

4.2. Material

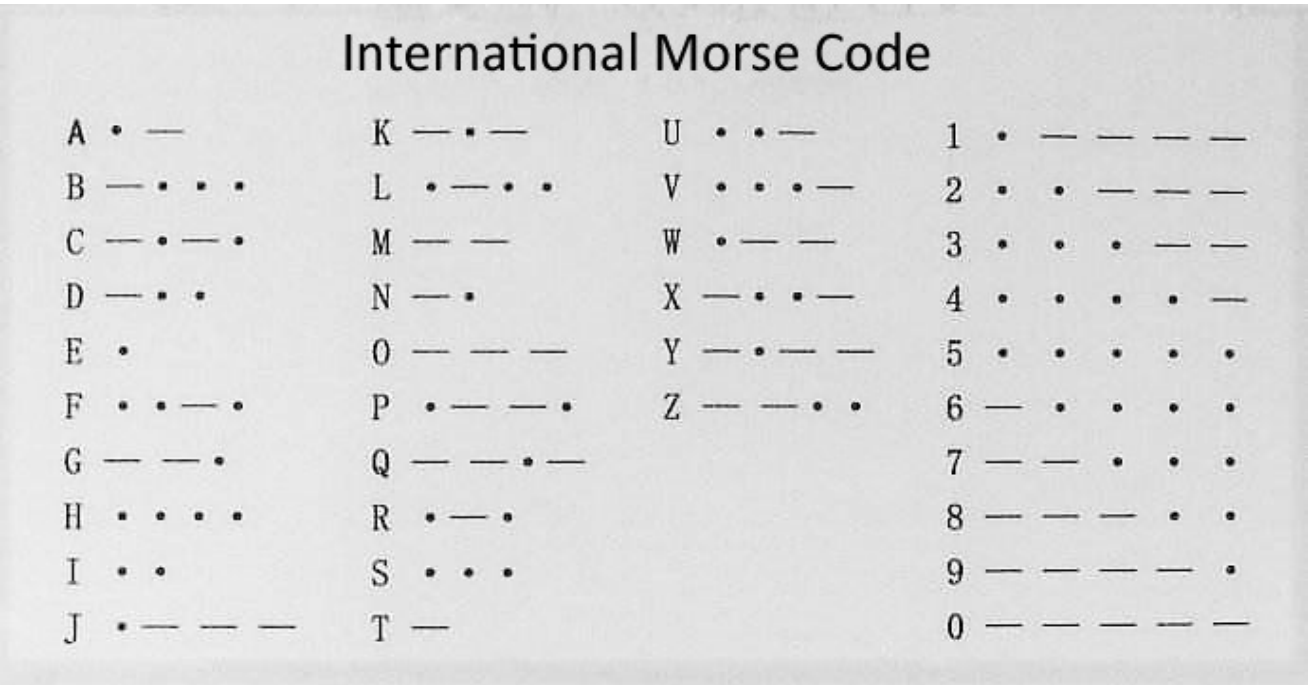
- 1 x basic kit

4.3. Background Knowledge

Morse Code

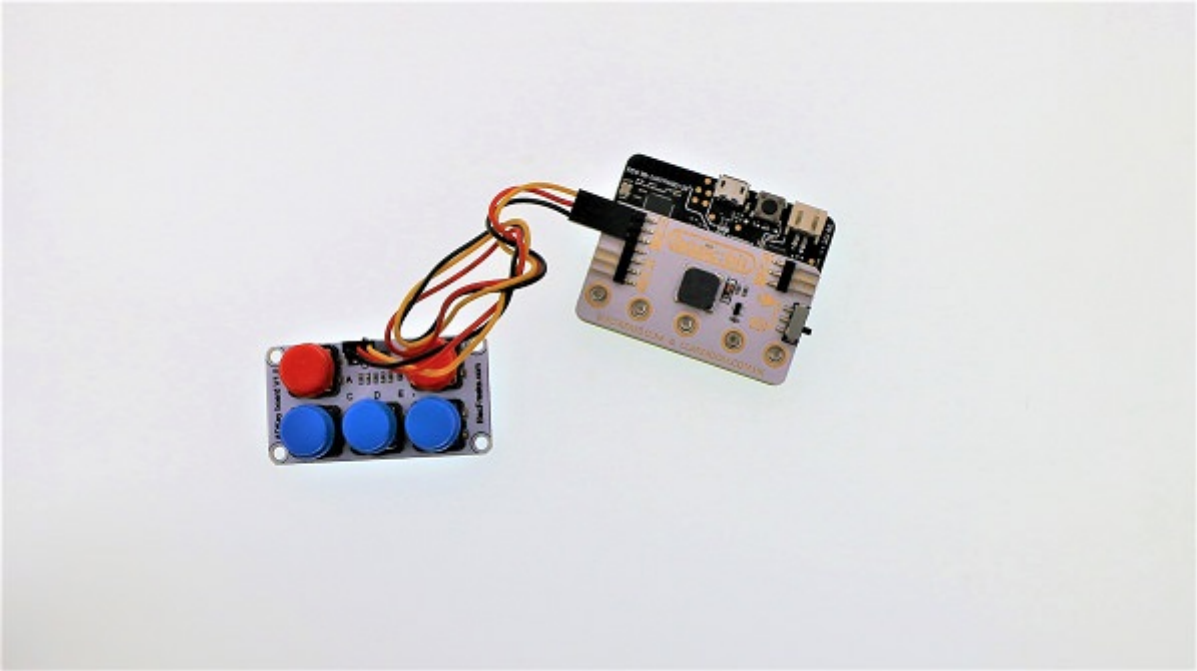
- Morse Code is a method of transmitting text information as a series of on-off tones, lights, or clicks that can be directly understood by a skilled listener or observer without special equipment. It is named for Samuel F. B. Morse, an inventor of the telegraph. The International Morse Code encodes the ISO basic Latin alphabet, some extra Latin letters, the Arabic numerals and a small set of punctuation and procedural signals (prosigns) as standardized sequences of short and long signals called “dots” and “dashes”, or “dits” and “dahs”, as in amateur radio practice.

Table of Morse Code



4.4. Hardware Connection

- Connect the crash sensor to P1 port on basic:bit. See picture below.



4.5. Software

- Microsoft Makecode
- ADKeypad analog read the return value of I/O port. Here's the values:

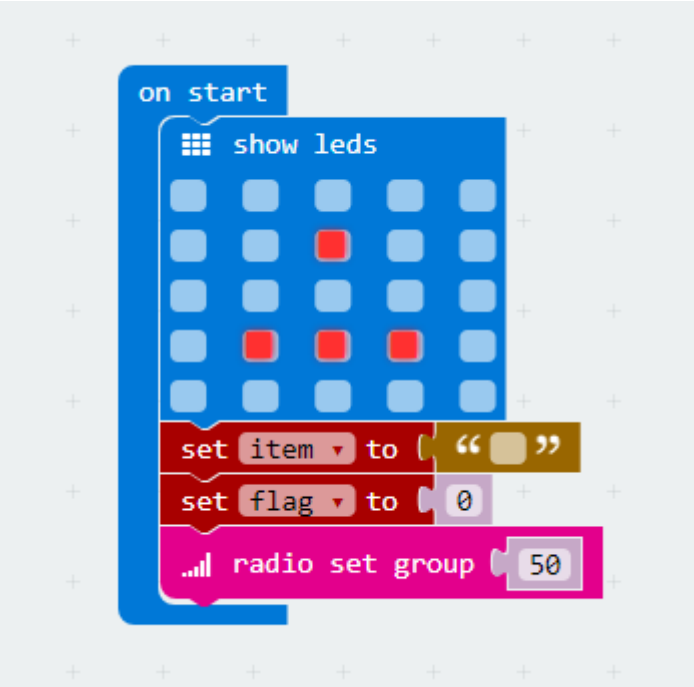
1. Button A <10
2. Button B: 10-80
3. Button C: 80-130

4. Button D: 130-160
5. Button E: 160-600

4.6. Programming

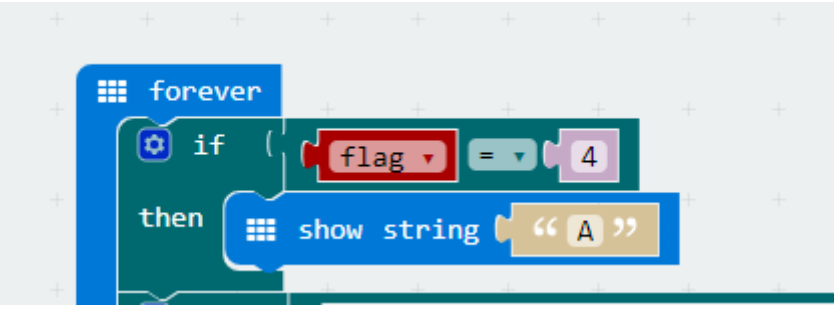
Step 1

- When start, display an image, set variable `item` to the data that ready to be sent, and set variable `flag` to the length of string.
- Set a radio group to 50.



Step 2

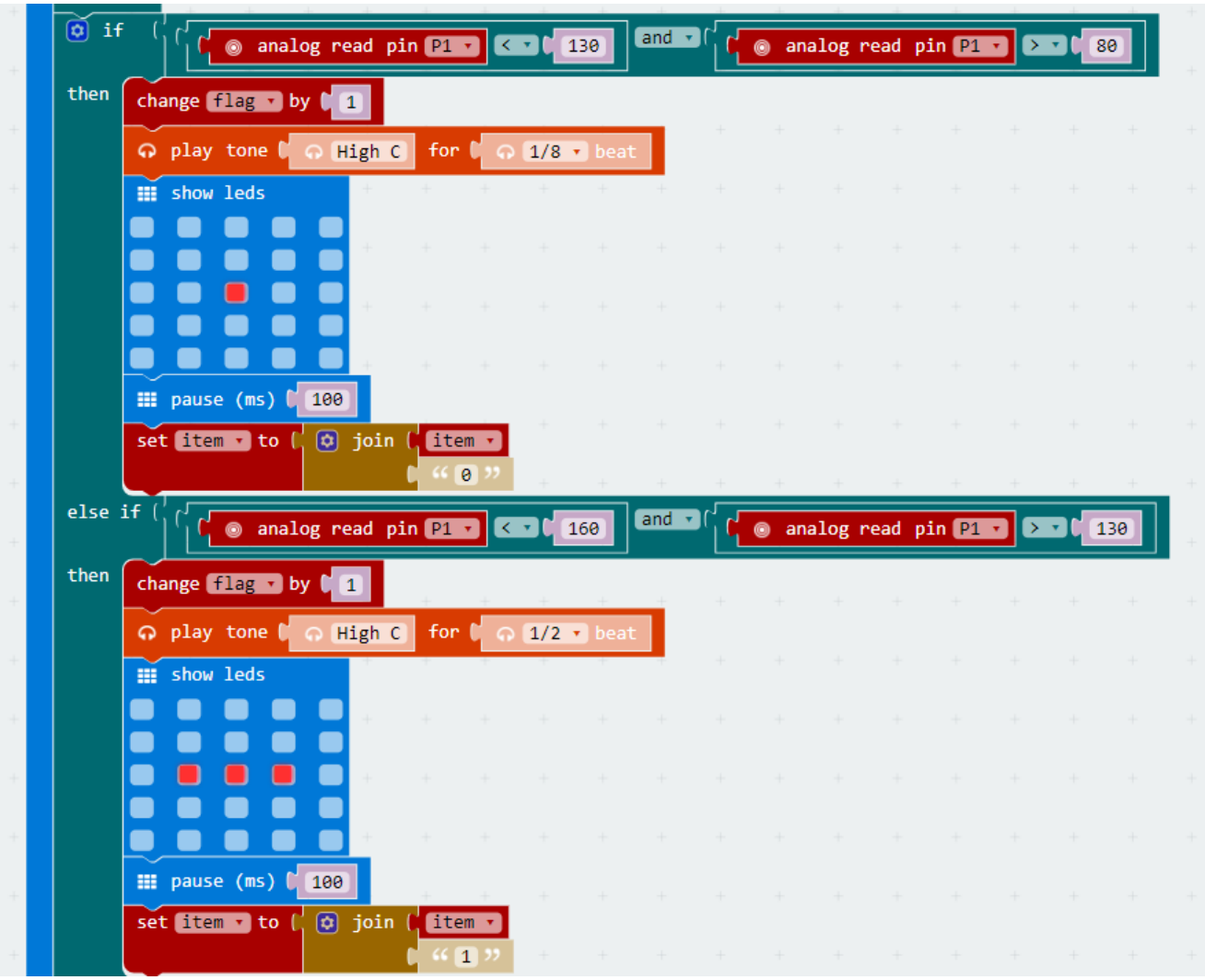
- Judge if the variable `flag` (string length) is 4. (in this case, we use English letters in Morse Code only, so the maximum length is 4.) If it is 4, then show string “A” to the user.



Step 3

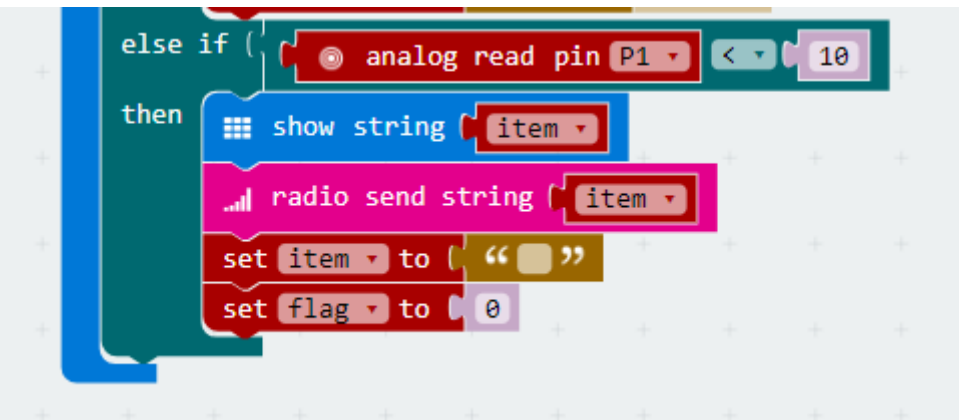
- Analog read the value of P1 port. Judge which button is pressed. If the return value is among 80-130, then it is button C that was pressed.

- When button C is pressed, increase the string length variable(`flag`) by 1, and play a tone for 1/8 beat, then display a pixel point on micro:bit screen, which stands for a dot in morse code. In the final, delay time to de-shock, and write character “0” to the end of the string.
- The event of button D is similar to button C, but it will display a dash (in morse code) on micro:bit screen, and write character “1” to the end of the string.



Step 4

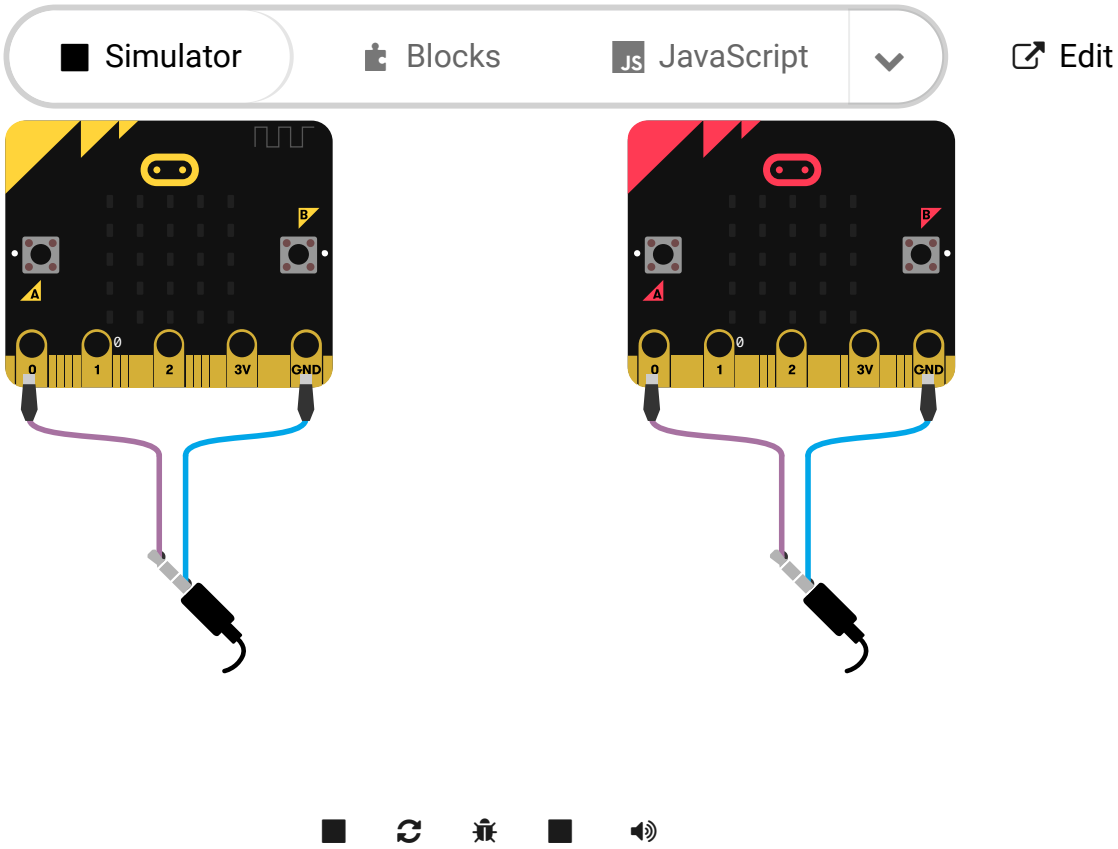
- When button A is pressed, display the string and then send it. At the same time, set variable `item` and `flag` to 0 and initialize the program.



Program

The link of the whole program: https://makecode.microbit.org/_3JrVPeeDVY2r

You can also check the program from the page below.



4.7. Result

- Press down button C, micro:bit screen will display a dot; press down button D, then it will display a dash. When the total press amount for button C or D is 4, then micro:bit will display the character A. At this time, we need to press button A to restart the program.

4.8. Think

- Why we need to delay time and debounce ?

4.9. FAQ

4.10. Relative Readings

5. case 03 Function Selector

5.1. Our Goal

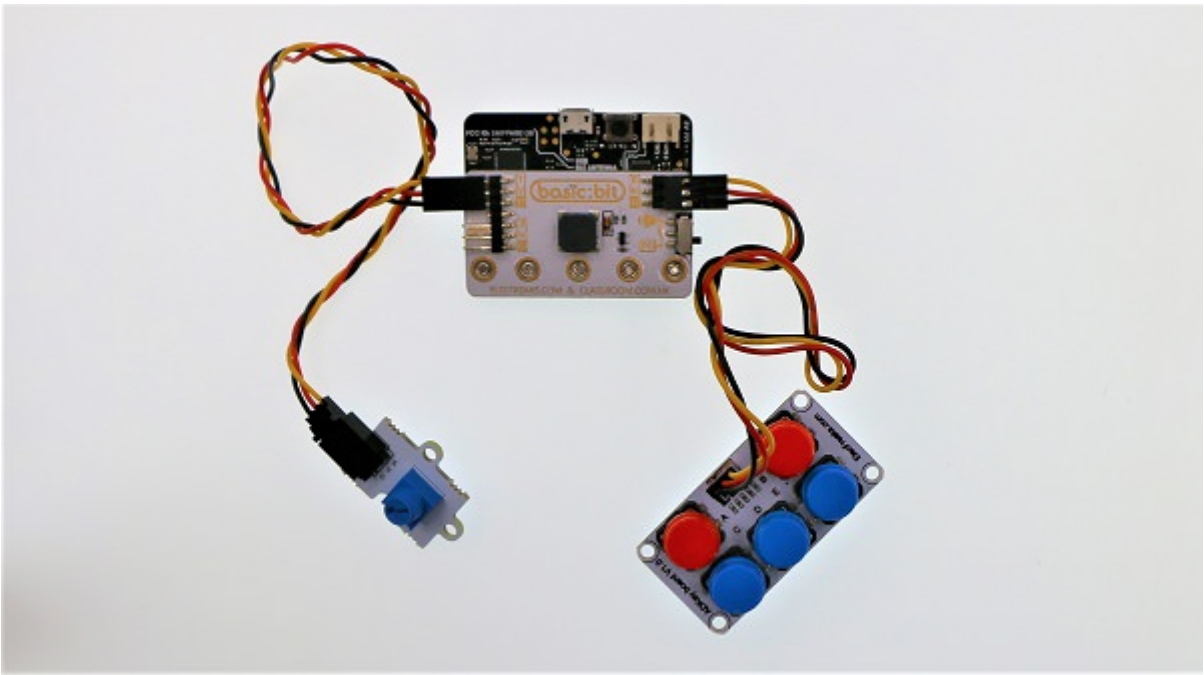
- Use basic kit to create a function selector.

5.2. Material

- 1 x basic kit

5.3. Hardware Connection

- Connect the crash sensor to P0 port on basic:bit and connect the potentiometer to P1. See picture below.



5.4. Software

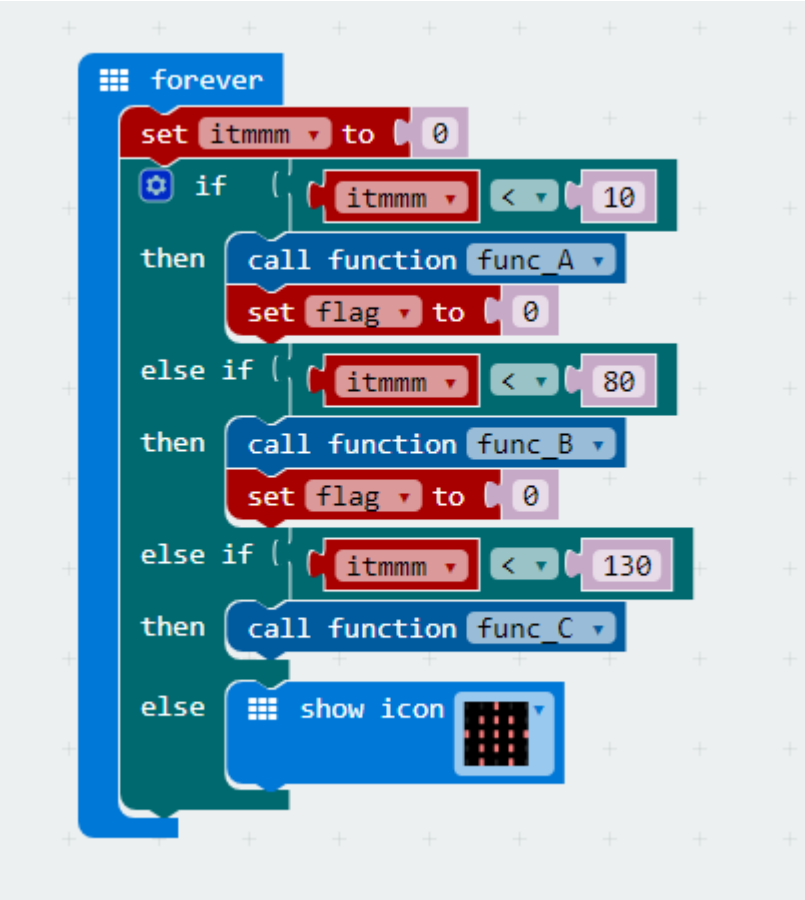
- Microsoft Makecode
- The crash sensor analog read the return value of I/O port. Here's the values:

1. Button A <10
2. Button B: 10-80
3. Button C: 80-130
4. Button D: 130-160
5. Button E: 160-600

5.5. Programming

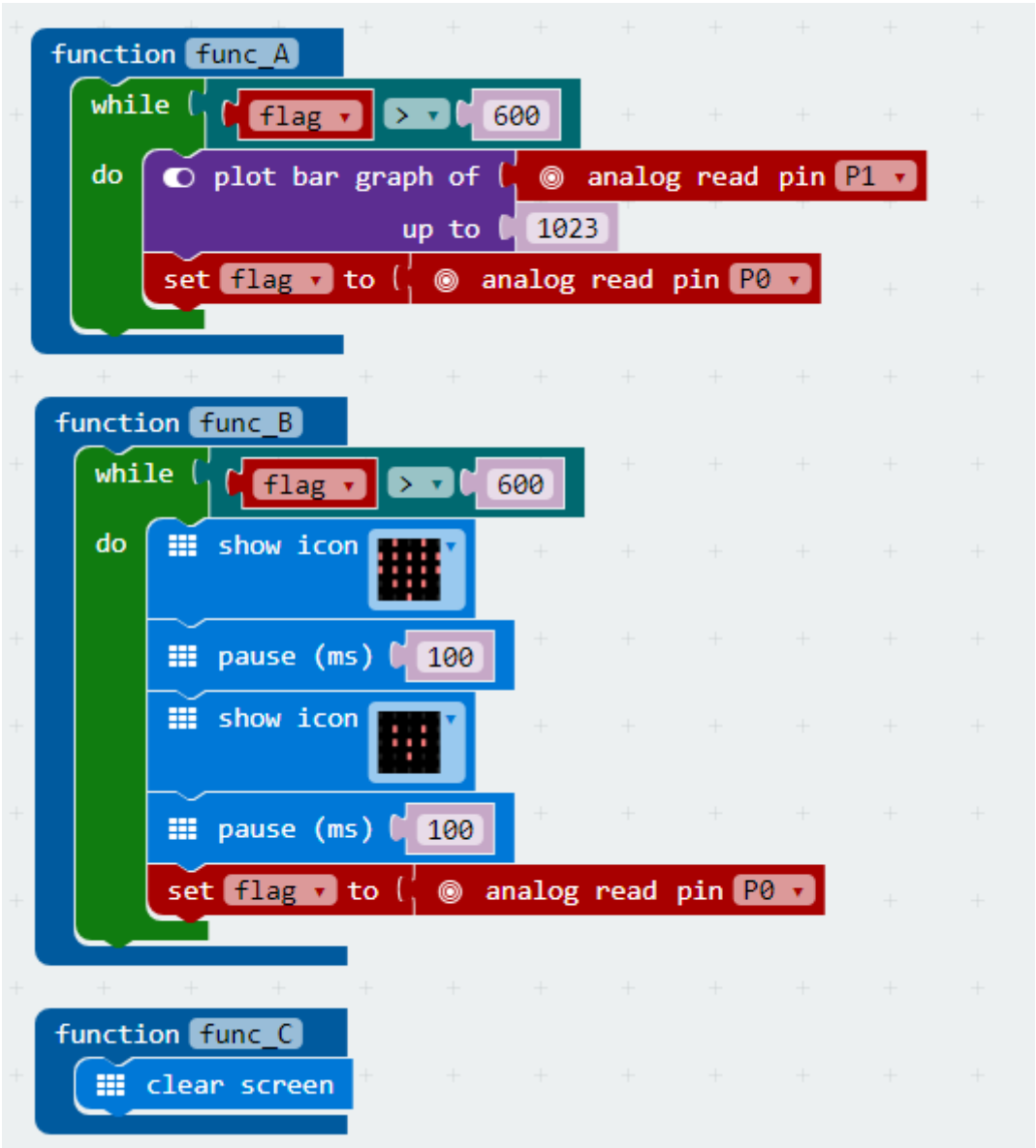
Step 1

- Create a forever loop, read the value of P0 port in analog way, and then assign the return value to variable `itmmm` so as to judge which button is pressed.
- If `itmmm` is under 10, it means button A is pressed. When button A is pressed, call function `func_A` and set variable `flag` (Variable for judging the internal loop in the function) to 0 after call.
- If `itmmm` is under 80, it means button B is pressed. When button B is pressed, call function `func_B` and set variable `flag` to 0 after call.
- If `itmmm` is under 130, it means button C is pressed. When button B is pressed, call function `func_C`.



Step 2

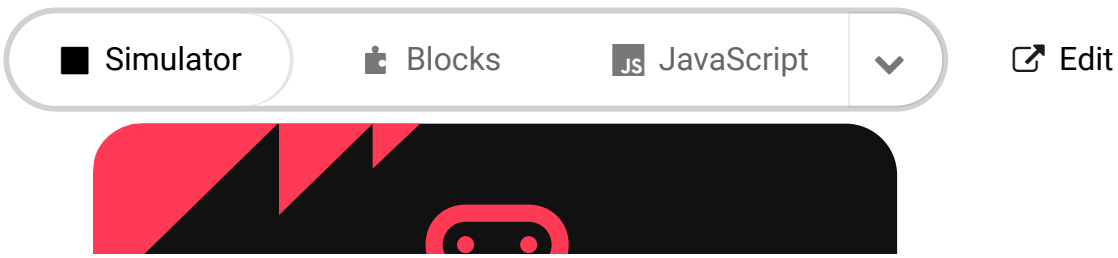
- Function `func_A`: When `flag` is beyond 600(i.e. no buttons pressed), read the value of P1 port and plot it on micro:bit screen. Read the button status of P0 port. When button E is pressed, the loop is terminated and the function call is finished.
- Function `func_B`: When `flag` is beyond 600(i.e. no buttons pressed), display a flashing heart. Read the button status of P0 port after each flash. When button E is pressed, the loop is terminated and the function call is finished.
- Function `func_C`: Clear the screen and finish the call of function.

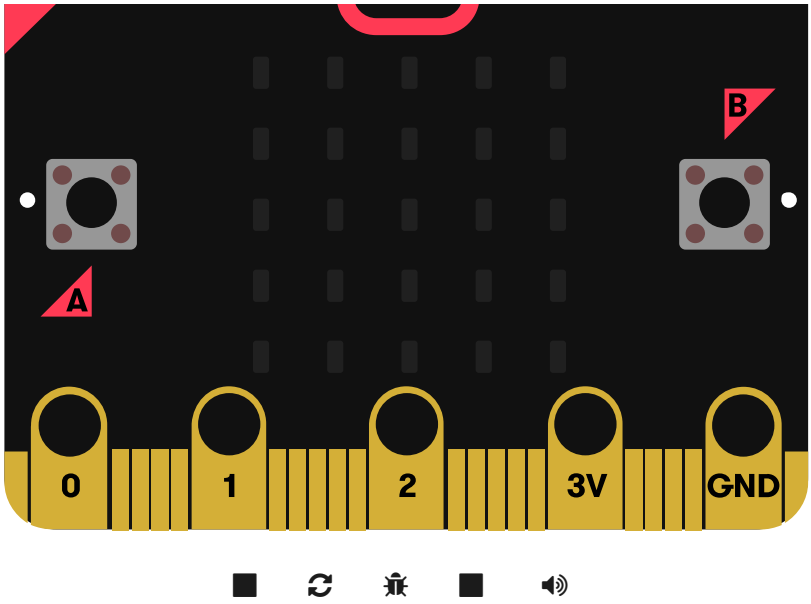


Program

The link of the whole program: https://makecode.microbit.org/_cuufKuP6FARo

You can also check the program from the page below.





5.6. Result

- When startup, micro:bit will display an image of house.
- Press button A to call function `func_A`. We can use potentiometer to control the brightness of the LED screen . And we can press any button to finish the call of function.
- Press button B to call function `func_B`, and micro:bit will display a flashing heart. And we can press any button to finish the call of function.
- Press button C to call function `func_C` and clear the screen.
- For other situations, micro:bit will display a house image.

5.7. Think

5.8. FAQ

Q: why there is nothing happened when a button is pressed down?
A: The button status judgement is not always happen. When other section of code is running, the button program will stop judge.

5.9. Relative Readings

Interrupt: When something unexpected occured during the operation of a computer, it will stop the current program and transfer to a new program. Once the new program is processed, it will return to and continue its original suspended program.

6. case 04 Show Box for Stickers

6.1. Goal

- Make a show box for stickers you like.

6.2. Materials

- 1 x Servo
- 1 x Cardboard
- 1 x Hot melt glue gun
- 1 x Scissors
- 1 x Handmade knife
- 1 x Crash sensor
- 1 x Batteries pack
- 1 x basic kit board

6.3. Background

What is a show box for your stickers?

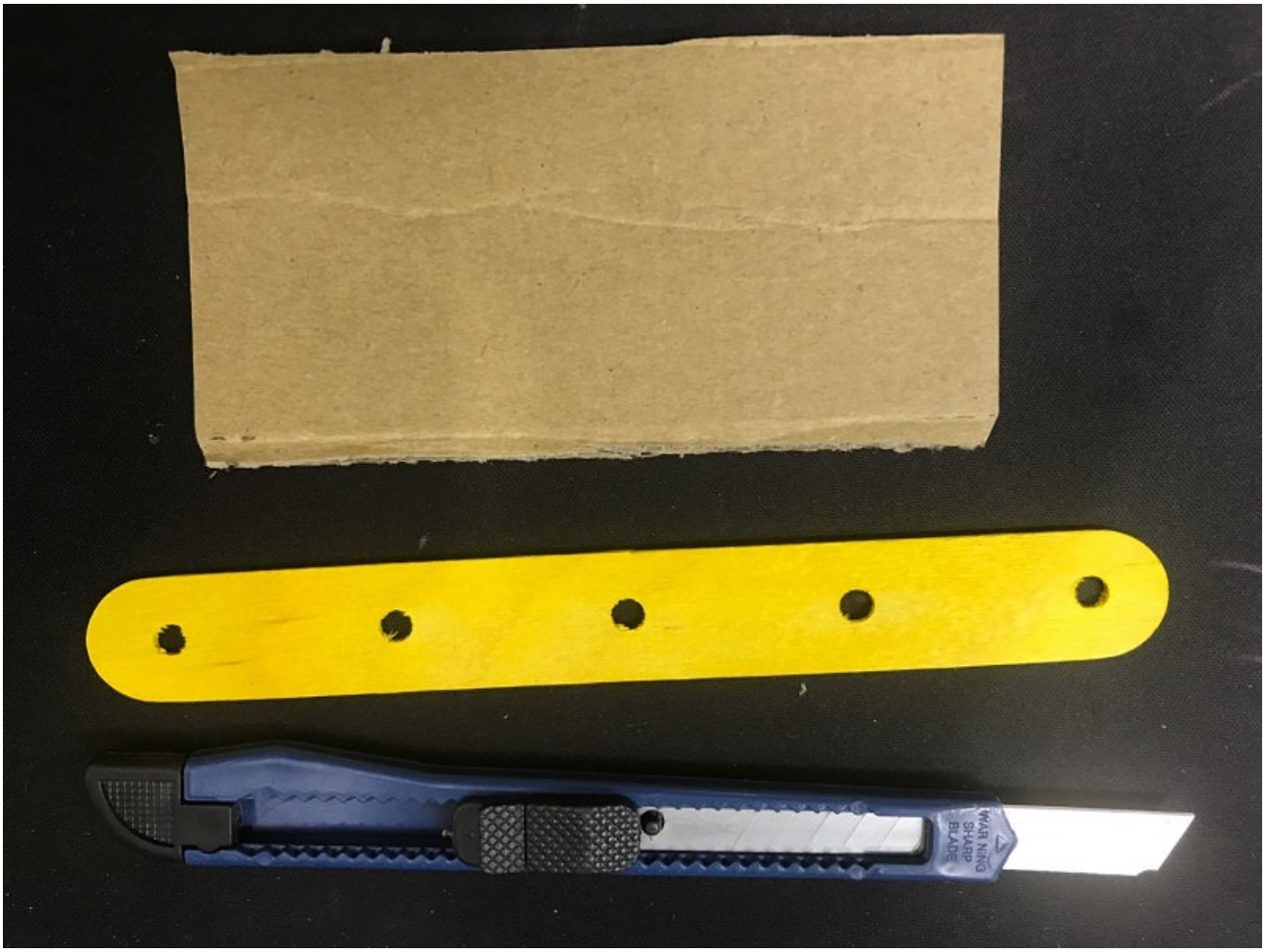
- Do you want to show your lovely stickers to others ? Let's make a show box for stickers.

How does it work?

- When micro:bit is received signal detected by crash sensor by basic:kit board, the servo turns and the door of show box opens.

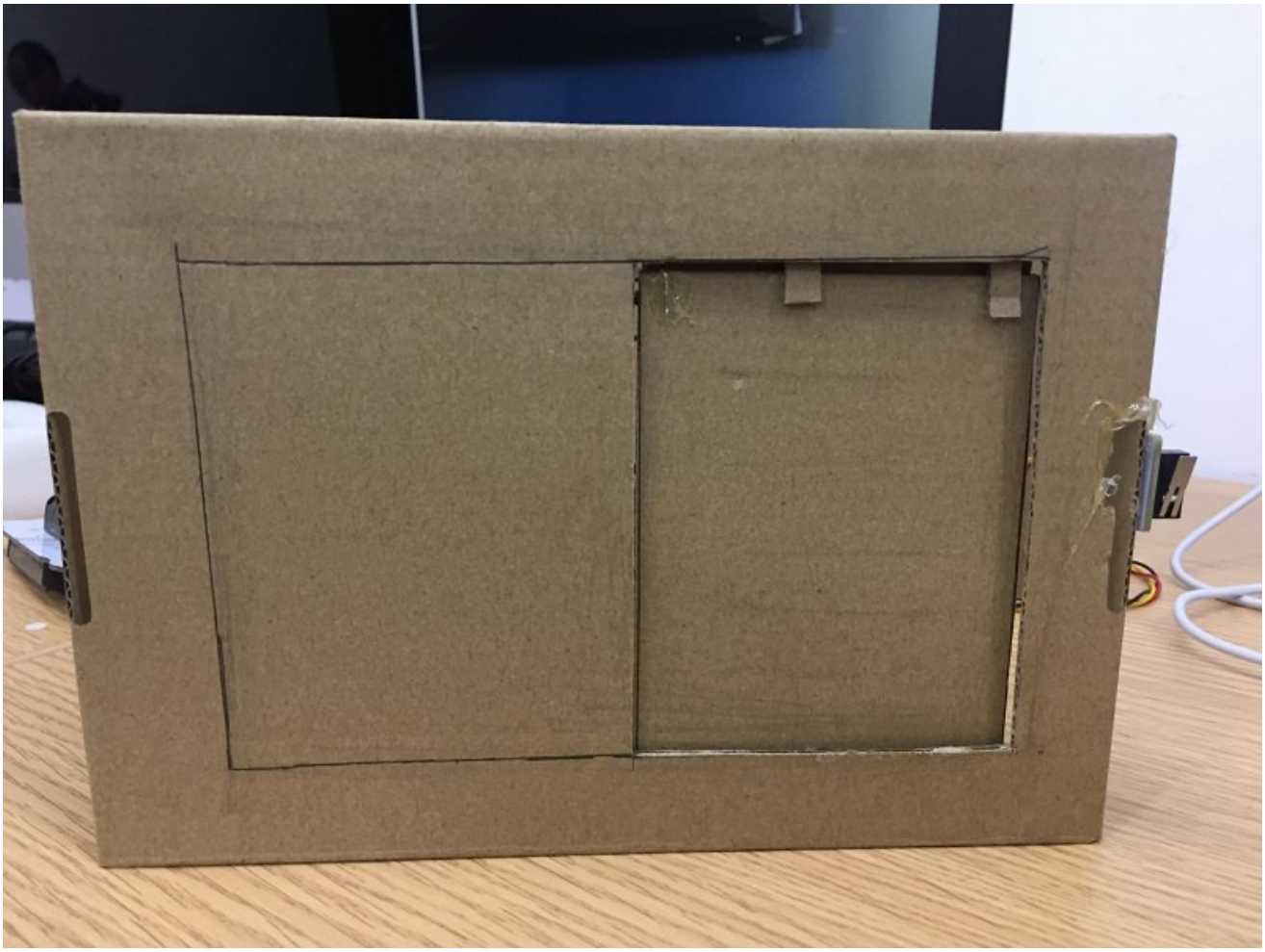
6.4. Practical operation

Prepare a handmade knife, a hot melt glue gun, a scissors and a piece of cardboard.

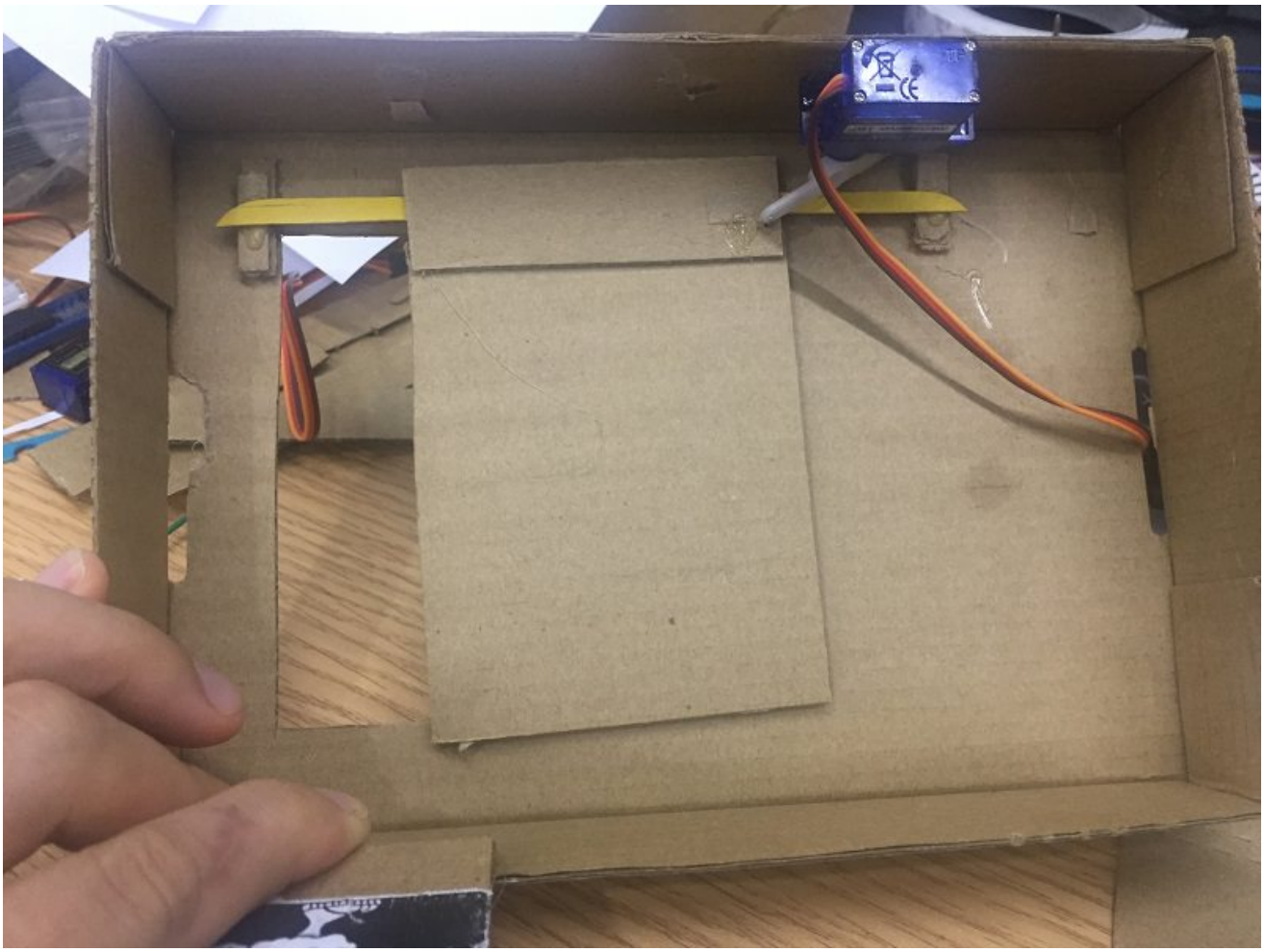

WARNING
SHARP
BLADE

Set up as below：

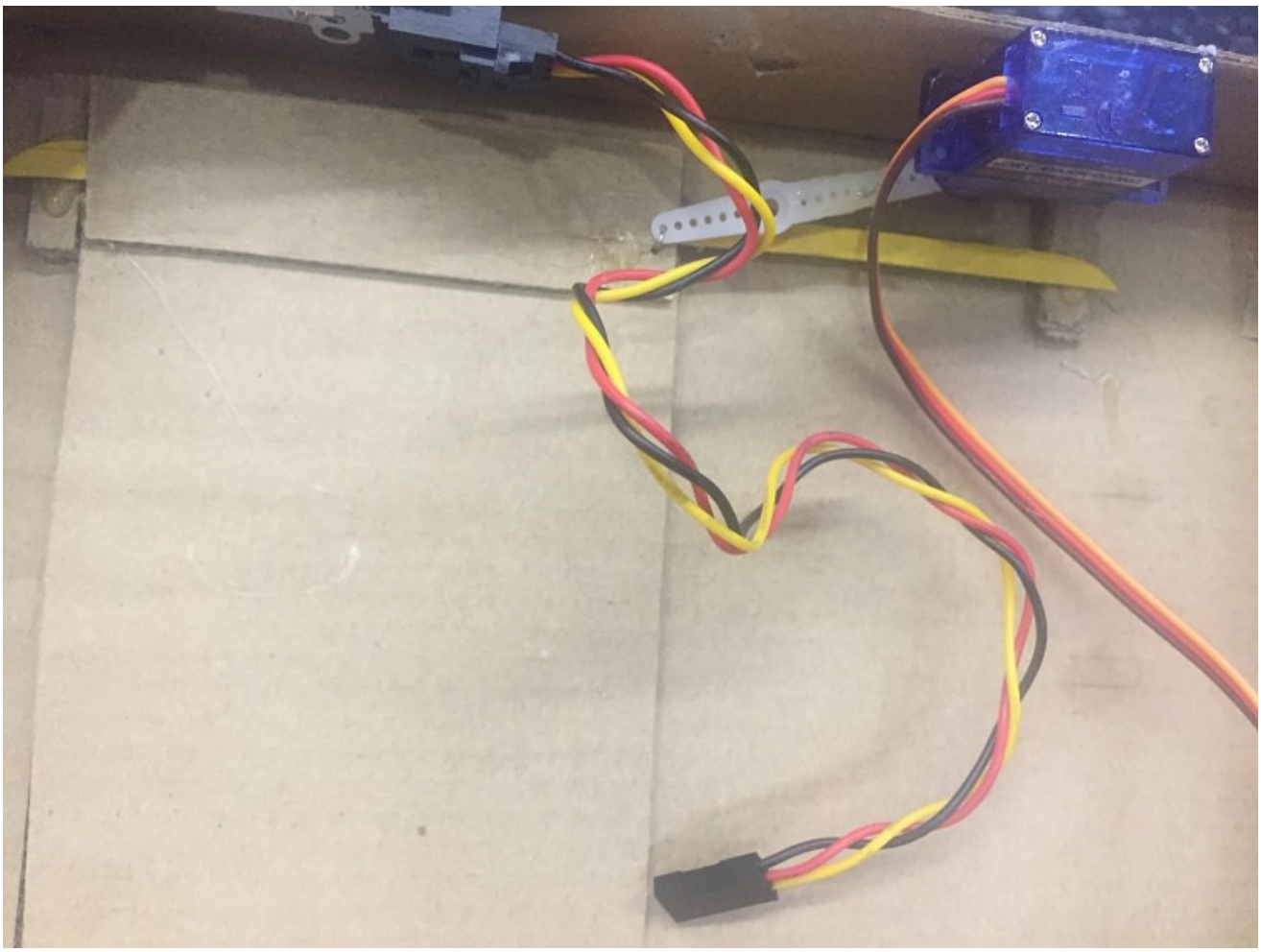
Front side：



Back side：

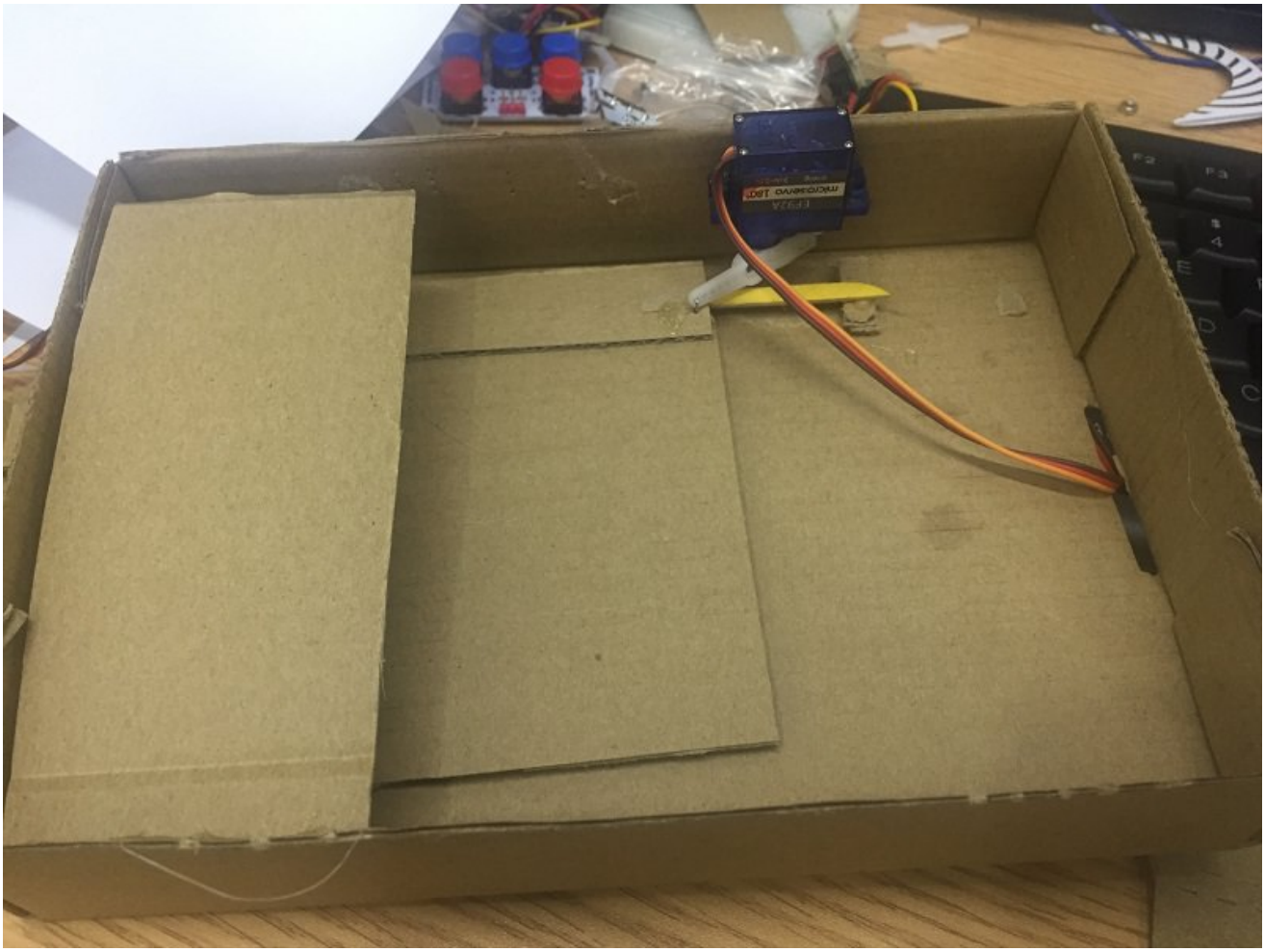


Set and stick components as below.



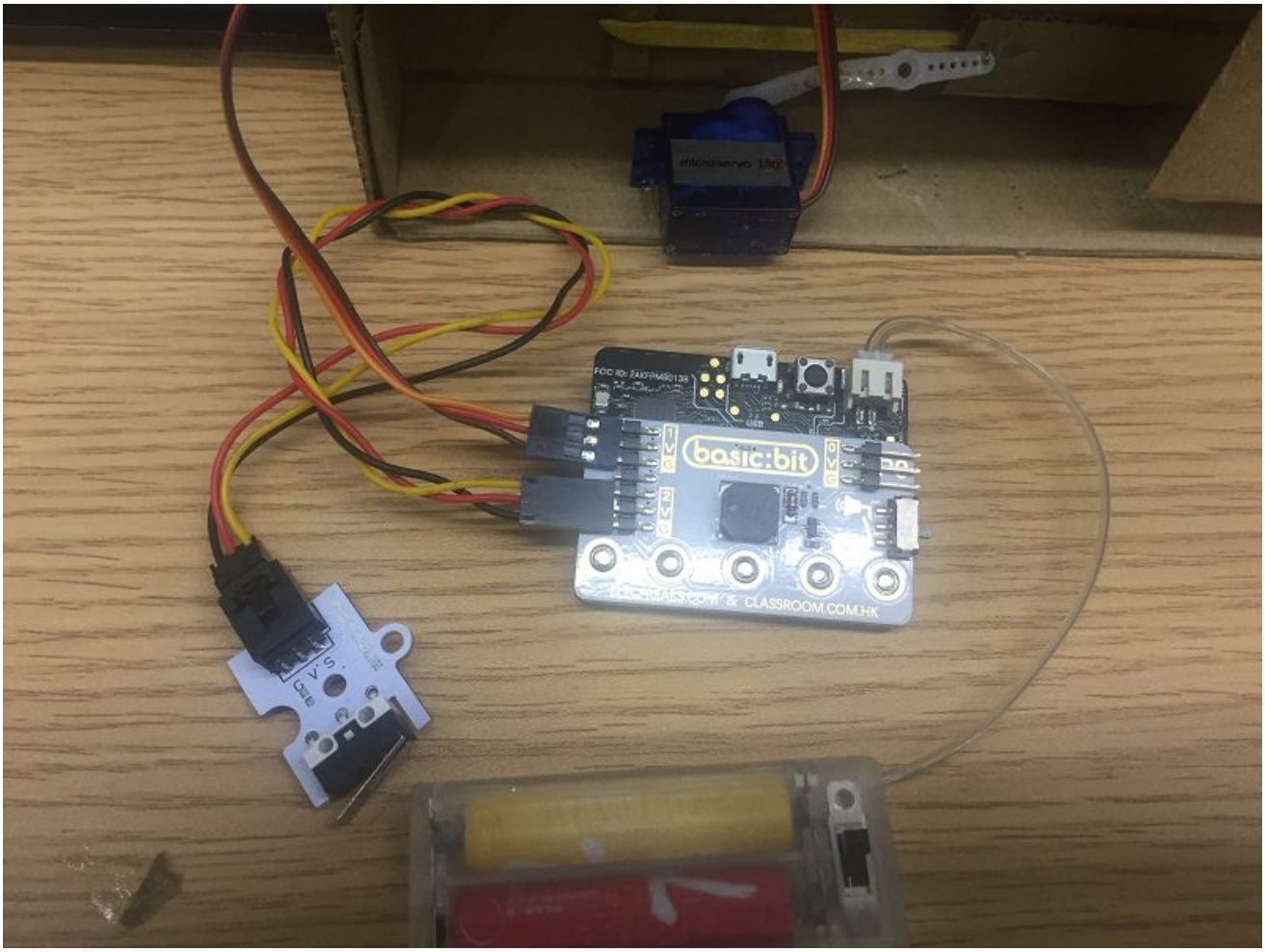
Prepare your lovely stickers and stcik them as below.

Thread wires through the hole of front side.



6.5. Hardware connect

- Connect servo to P1, crash sensor to P2 and battaries pack to micro:bit. (as below)



6.6. Software

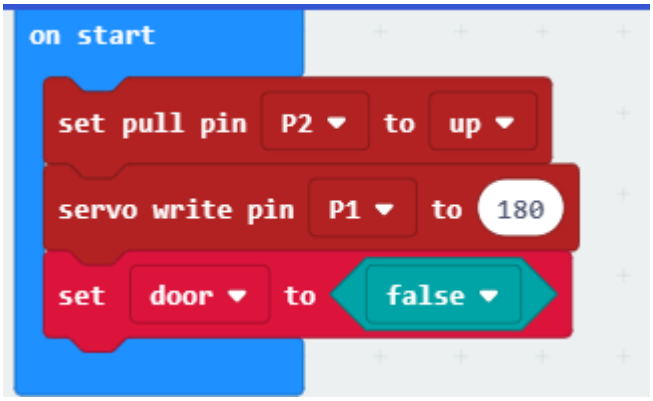
makecode

6.7. Programming

Step 1

Under “on start”, set pull pin P2 to up to keep signal stable. Set servo write pin P1 to 180, then the sticker board is at rest.

Set door as a variable, and its initial value is false for saving status of sticker board.

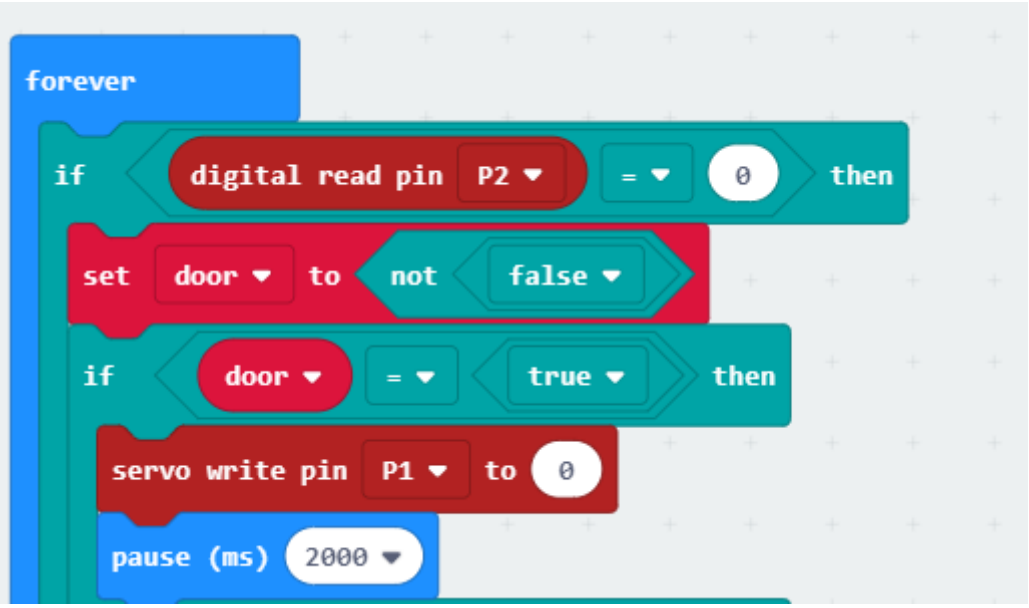


Step 2

Under forever, snap if statement to judge if the P2 port is equal to 0, it means if the sensor switch is pressed.

When the value P2 port is equal to 2, the sensor switch is pressed, set variable door to not door.

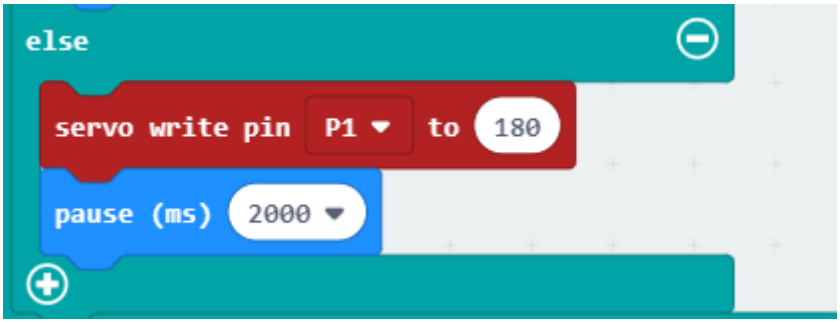
Then, snap another if statement to control the value of varible door, which means to judge the status of the door. If variable door is equal to true, then the sticker board will move(P1=0).



Step 3

When the value of P2 is not equal to 0, the sensor switch is not be pressed.

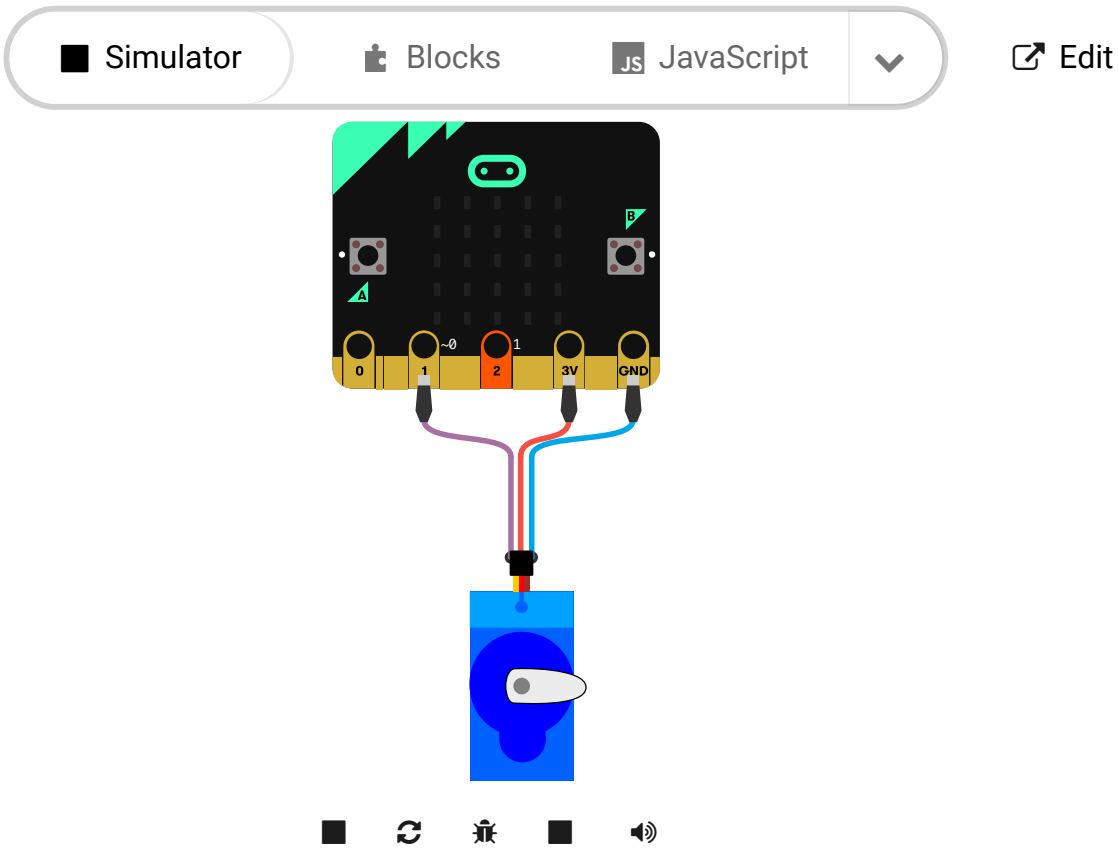
Then, the sticker board is at rest(set P1 to 180).



Program

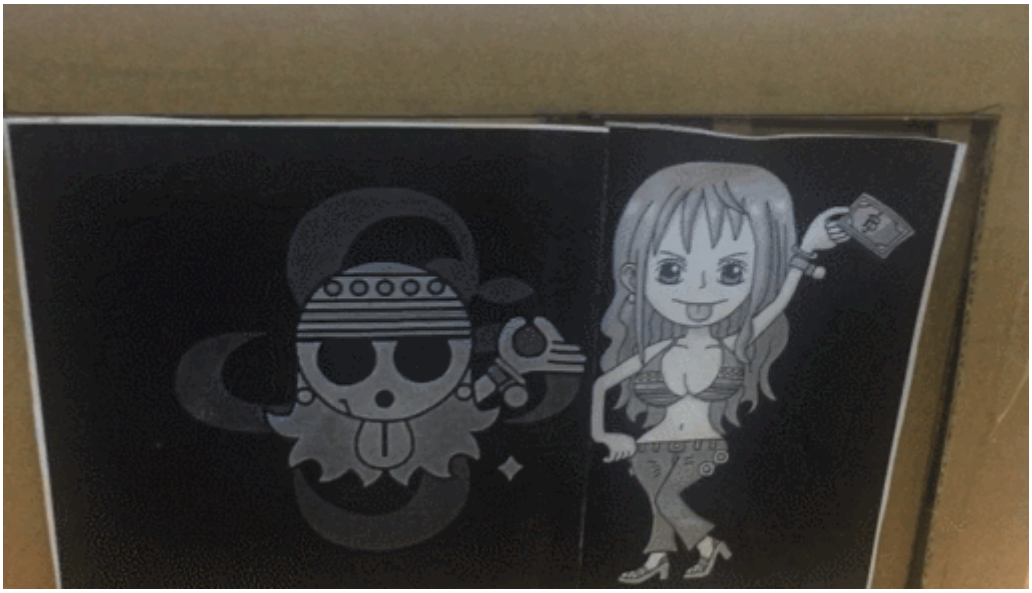
Please refer to detail programming：https://makecode.microbit.org/_0JmTbLKuXA8s

You also could directly download program visit website as below:



6.8. Result

Press the crash module, your stickers begins to moving.



6.9. Think

How to show more stickers to your friends?

6.10. Questions

6.11. More Information

7. case 05 Shock Box

7.1. Goal

- Make a shock box.

7.2. Materials

- 1 x Servo
- 1 x Cardboard
- 1 x Hot melt glue gun
- 1 x Scissors
- 1 x Handmade knife
- 1 x Crash sensor
- 1 x Batteries pack
- 1 x basic kit board (Basic:bit)

7.3. Background

What is a shock box ?

- Want to be special? Let's make a shock box and you will find more when you turn off the light.

How does it work?

- When micro:bit is received signal detected by crash sensor by basic:kit board, the servo turns and the shock box turns and shows different faces.

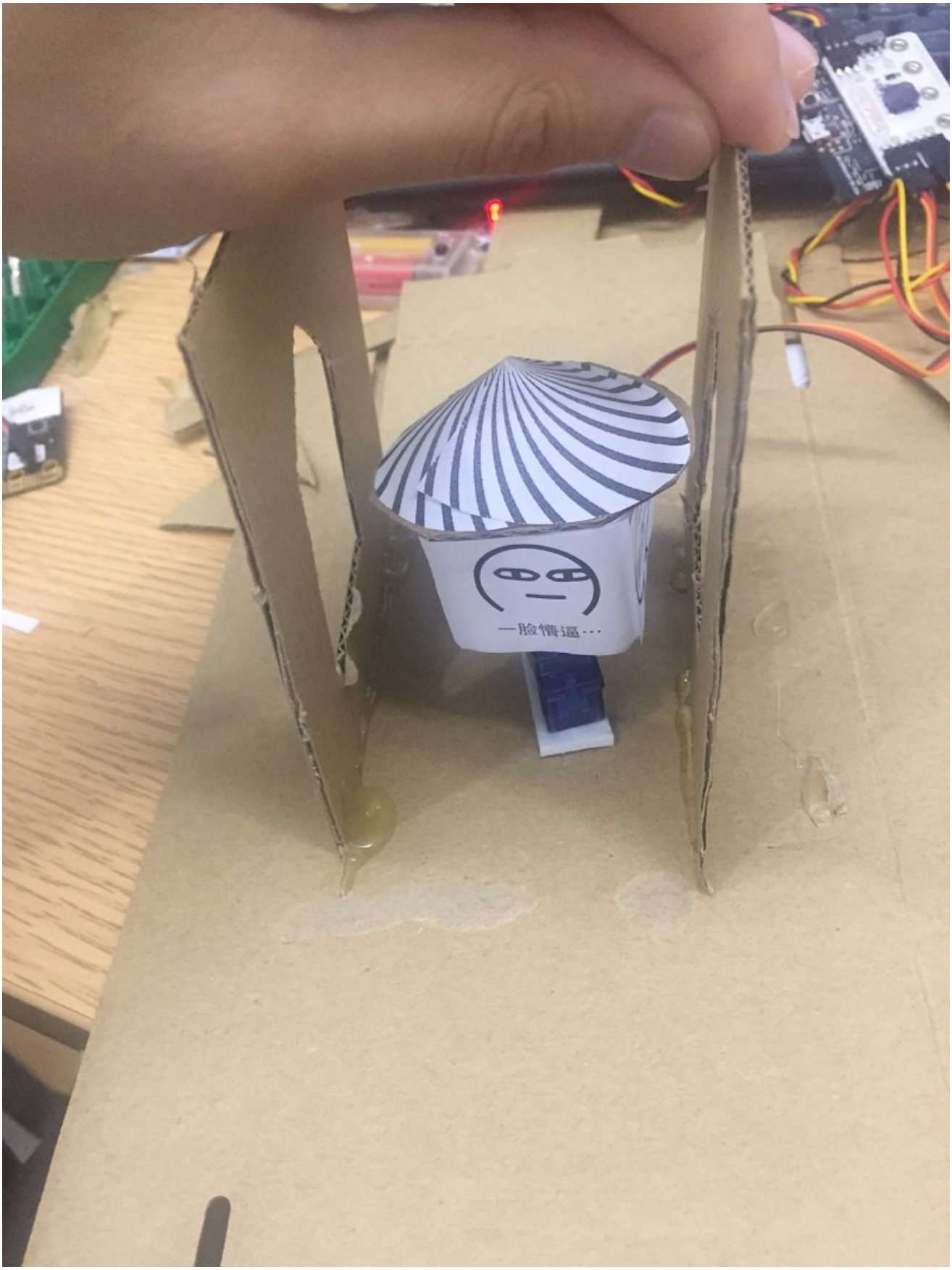
7.4. Practical operation

Prepare a handmade knife, a hot melt glue gun, a scissors and a piece of cardboard.

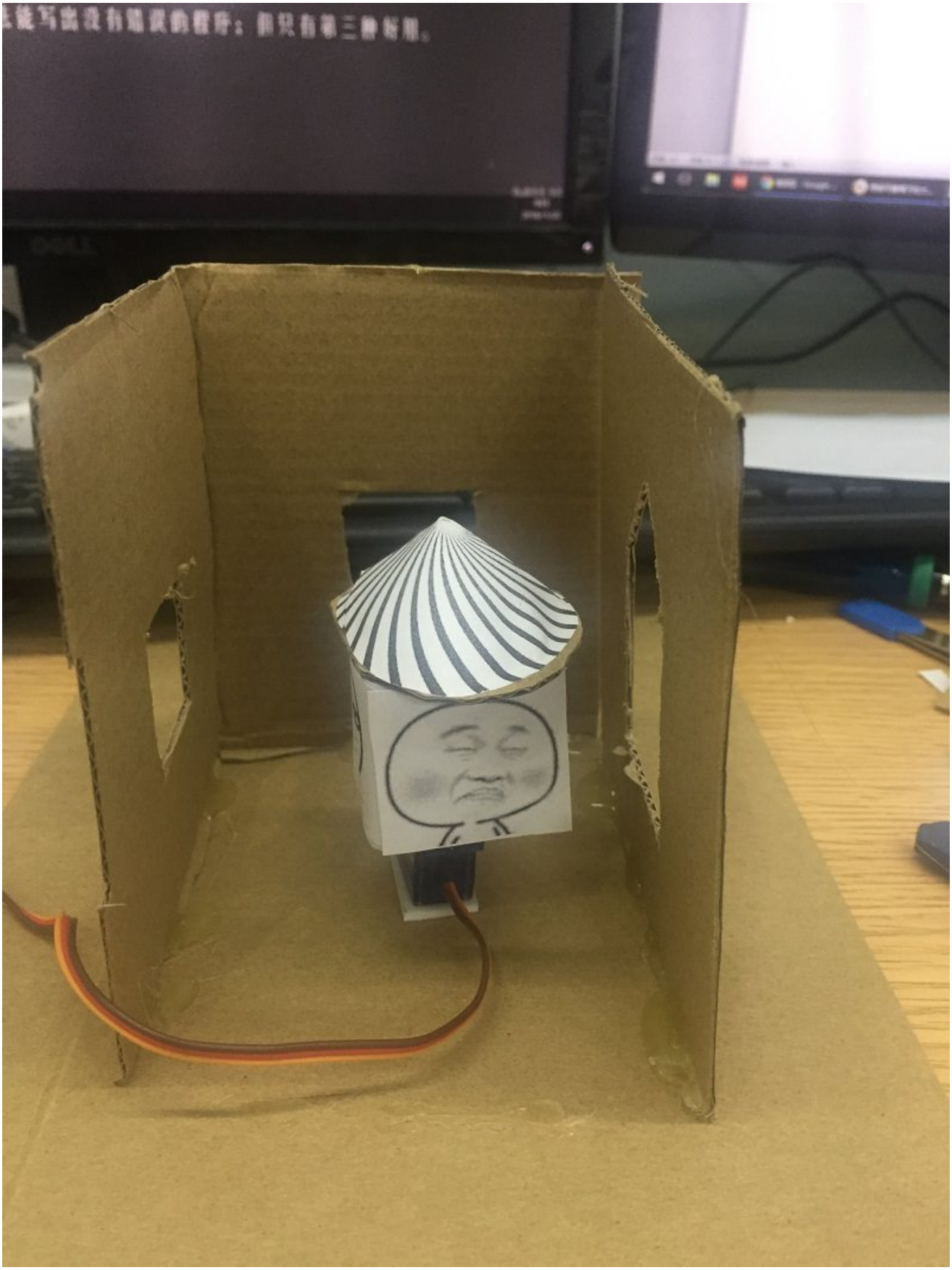

WAR NING
SHARP
BLADE

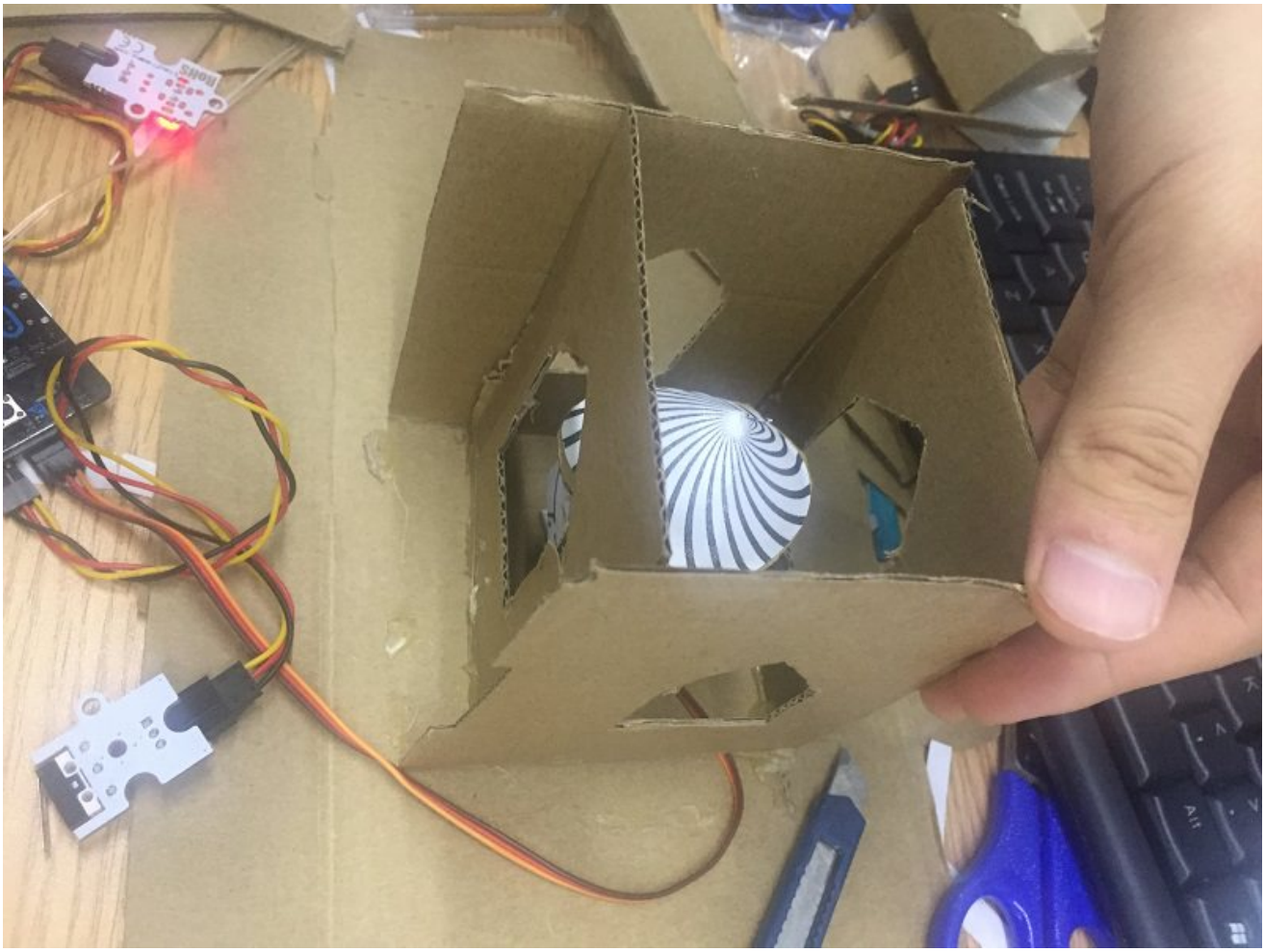
Set up as below：

Make a little square box which has four special faces(You also could draw it by yourself).
Make a little paper cover on the box and set the box on servo.

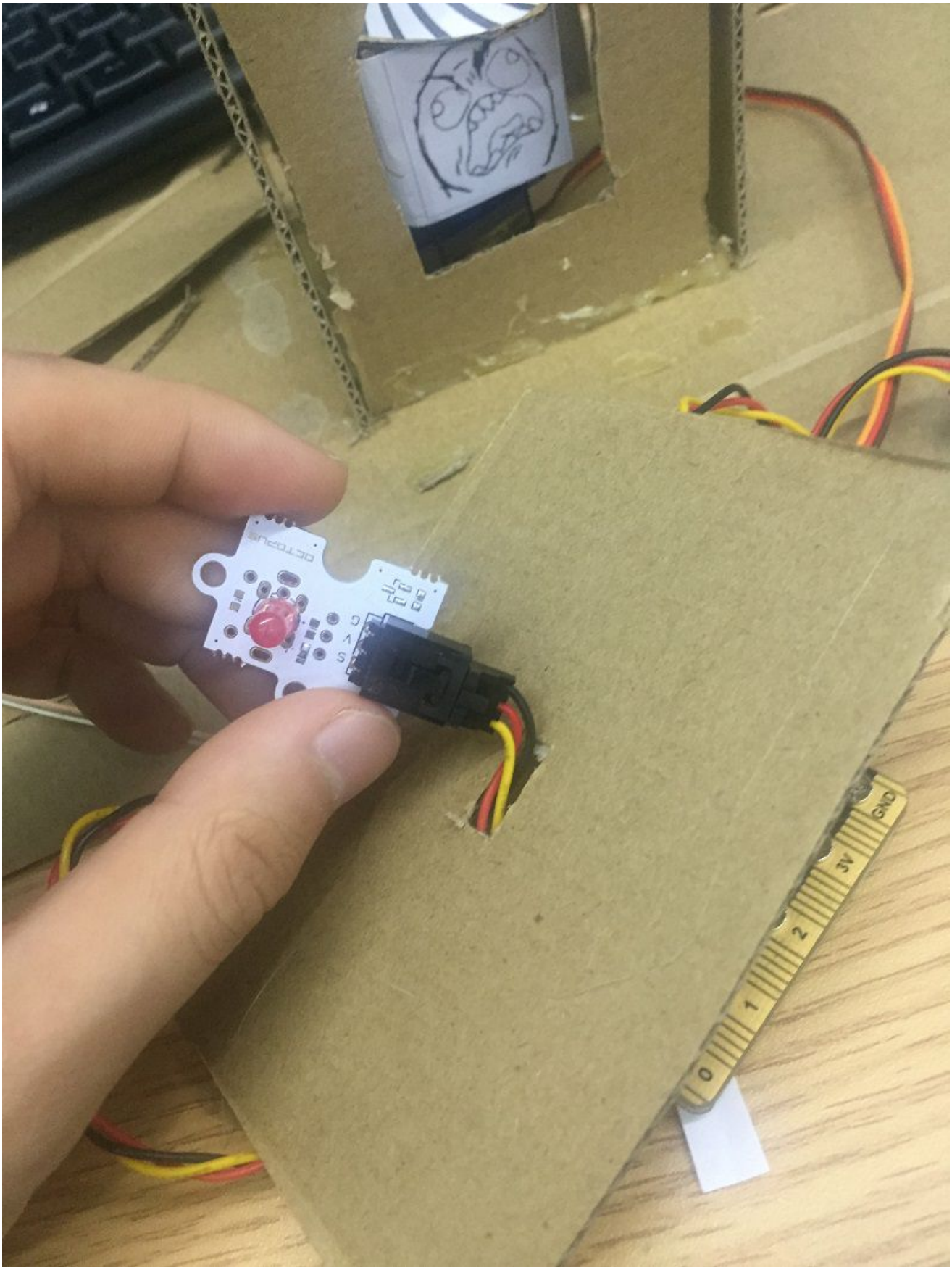


Clip four pieces of the cardboard and cut the centre part of the four pieces, then place the four cardboard around the box. You could place them closely to the box and you could clearly see the faces.

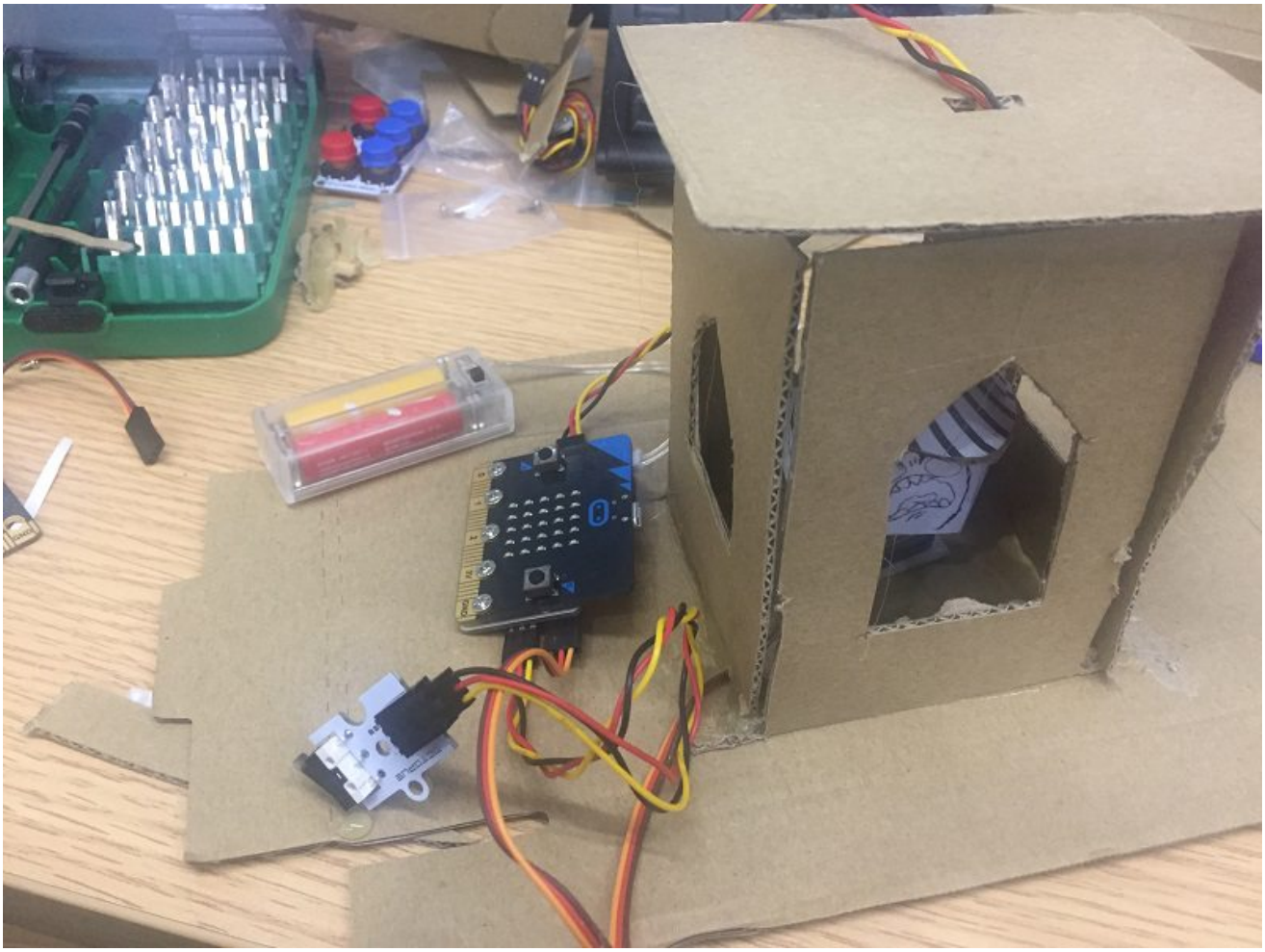




Make a roof by cutting your cardborad and drill a small hole of the roof. Then thread light wire through the hole.

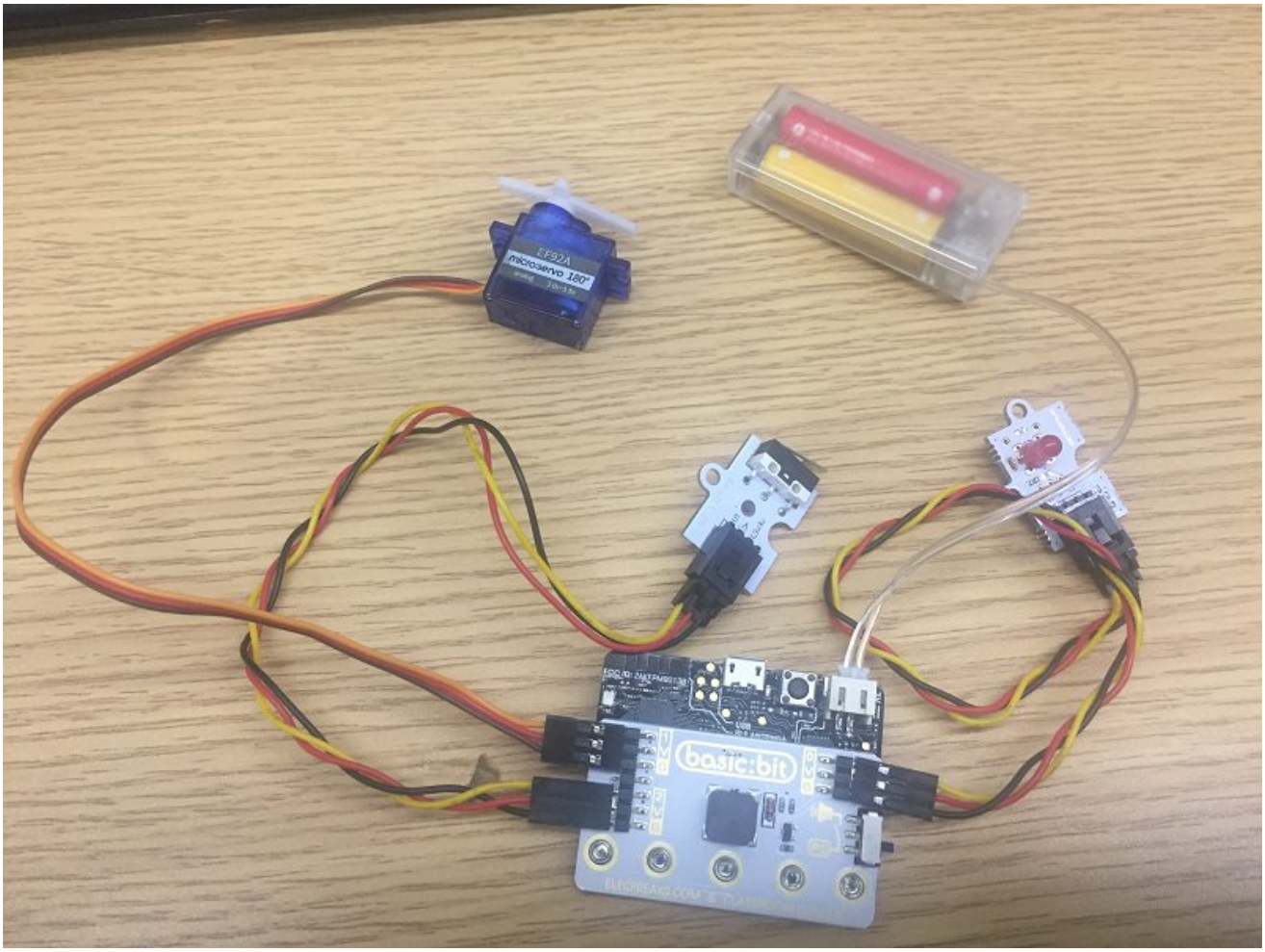

GND
3V
2
1
0

Set and stick components as below.



7.5. Hardware connect

Connect small LED to P0, servo ro P1, crash sensor to P2 and battaries pack to micro:bit. (as below)



###Please note the swith of basic:bit should at P0.

7.6. Software

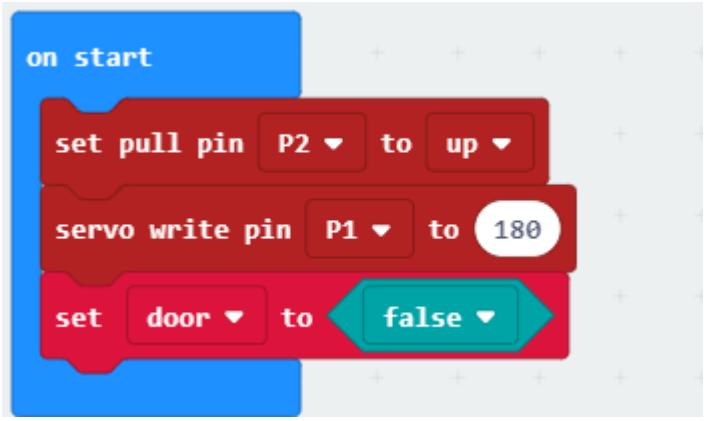
makecode

7.7. Programming

Step 1

Under “on start”, set pull pin P2 to up to keep signal stable. Set servo write pin P1 to 180, then the sticker board is at rest.

Set door as a variable, and its initial value is false for controlling turn of faces box.

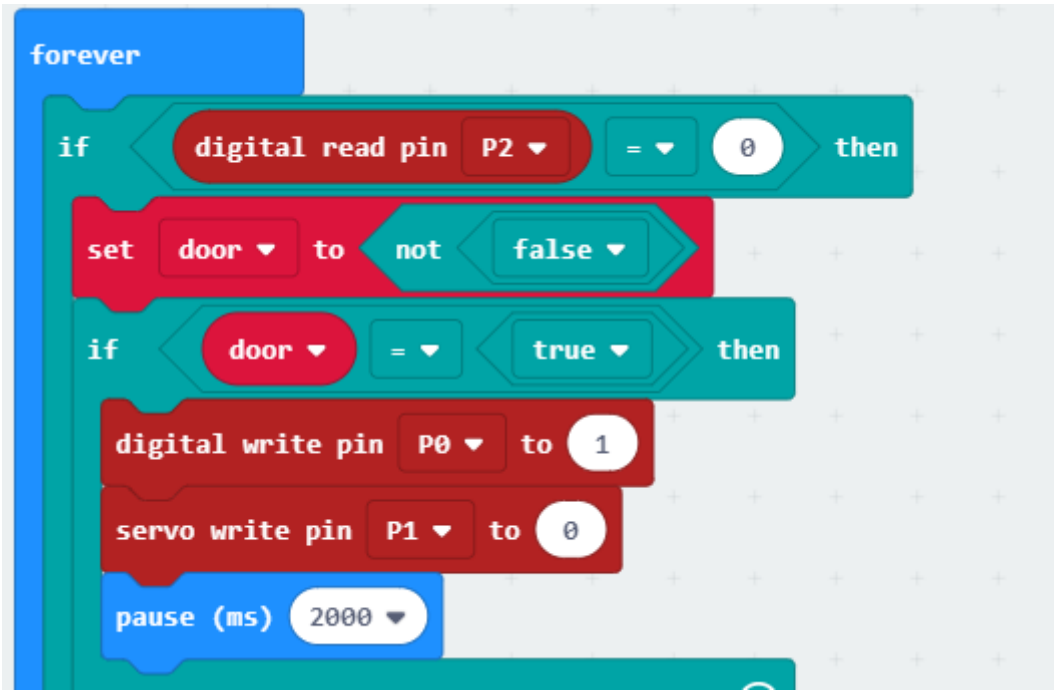


Step 2

Under forever, snap if statement to judge if the P2 port is equal to 0, it means if the sensor switch is pressed.

When the value P2 port is equal to 2, the sensor switch is pressed, set variable door to not door.

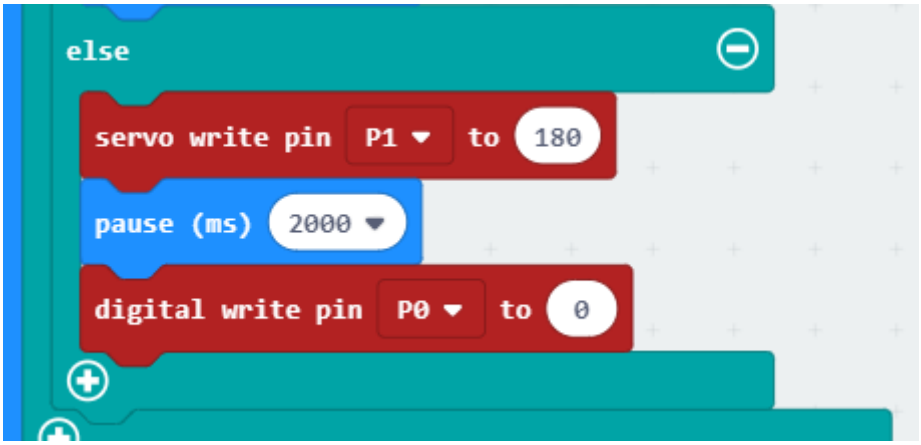
Then, snap another if statement to control the value of varible door, which means to judge the status of the door. If variable door is equal to true, then the show will turn and the light will up(P1=0).



Step 3

When the value of P2 is not equal to 0, the sensor switch is not be pressed.

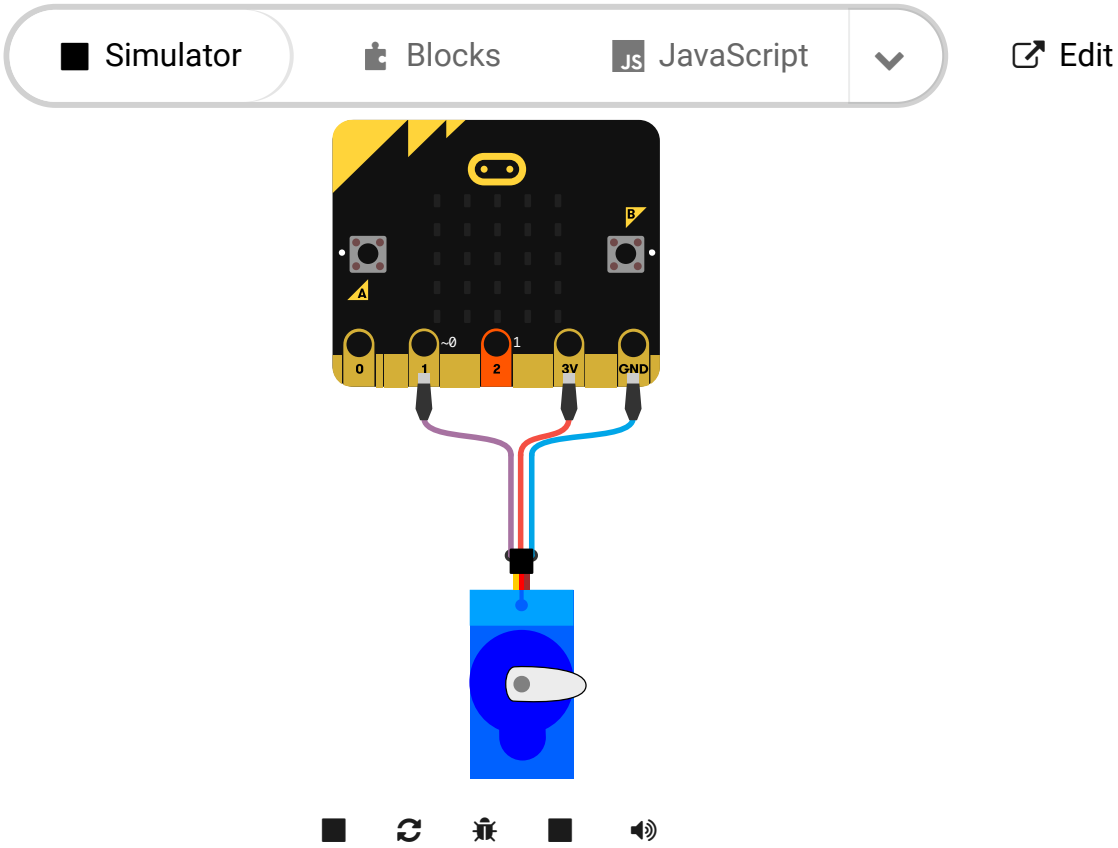
Then, the box at rest(set P1 to 180) and light will off.



Program

Please refer to detail programming：https://makecode.microbit.org/_M9M33fRYo9KY

You also could directly download program visit website as below:



7.8. Result

Press the crash module, the box turns and shows different faces

7.9. Think

How to make the faces box turns face by face ?

7.10. Questions

7.11. More Information