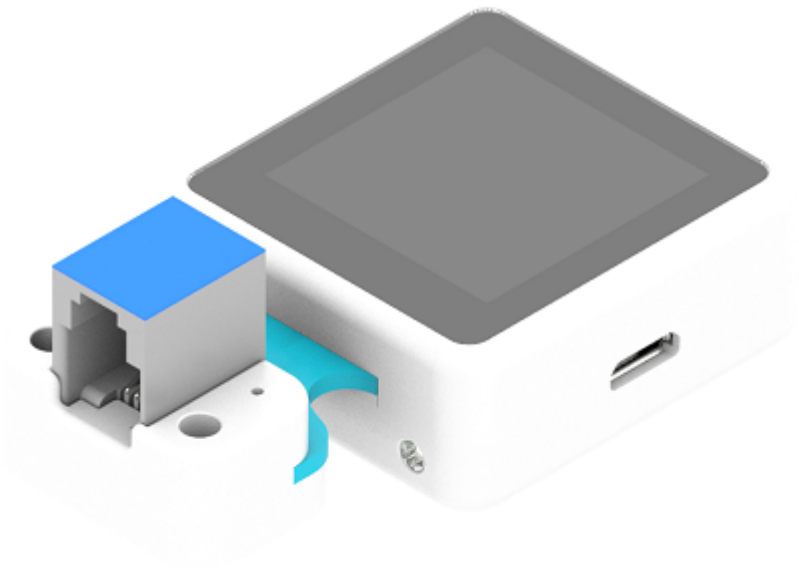
1.1. AI Lens

Introduction

The AI Lens is able to achieve the functions of face recognition, balls tracking, cards recognition and characteristics acquisition.



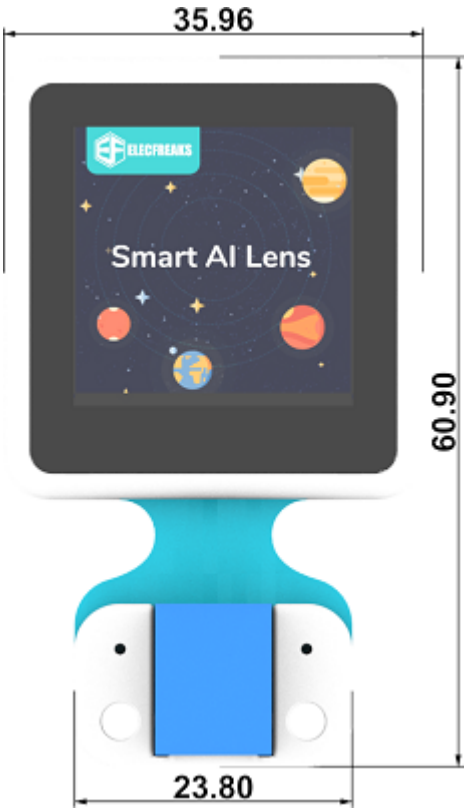
Characteristics

- Designed with RJ11 connection and easy to plug.

Specification

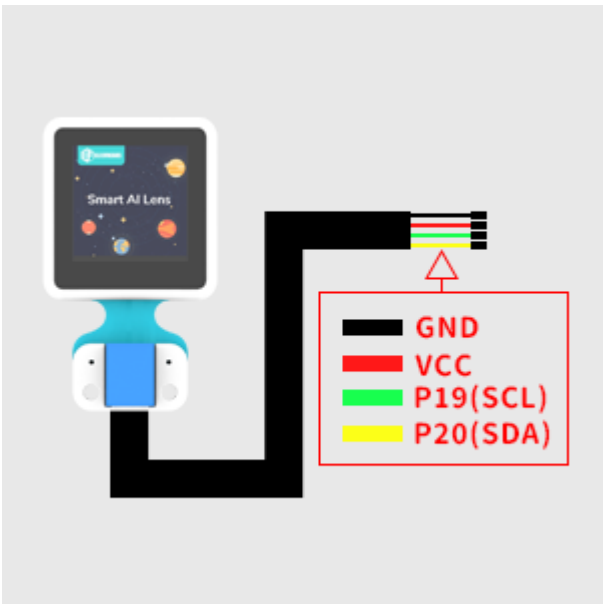
Item	Parameter
SKU	EF05045
Connection	RJ11
Connection Type	IIC
Working Voltage	3.3V
Core IC	K210

Outlook and Dimension

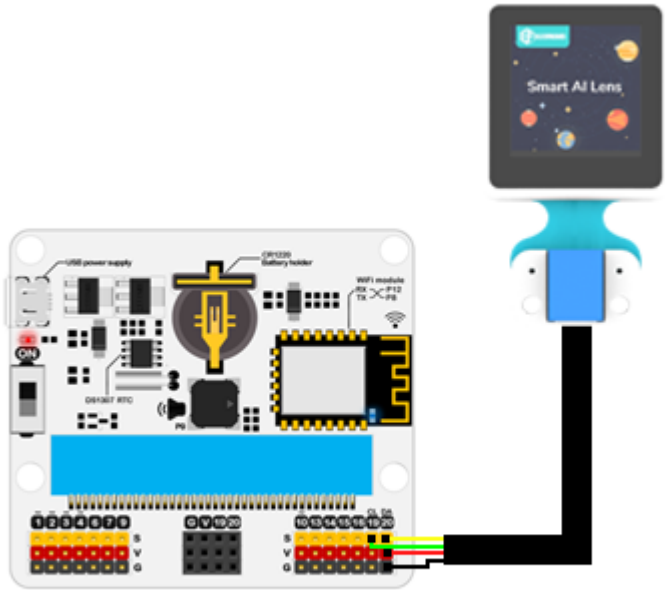


Connection Type: RJ11 to Dupont connector

While the micro:bit breakout board doesn't have the RJ11 connections, we can choose to use a Dupont wire that has the RJ11 connections. We need to connect the RJ11 to the Lens and the other ends connect to the breakout board(Black wire to GND; Red to VCC; Green to P19(SDA); Yellow to P20(SCL)). If the Lens doesn't work from the start, please try powering it with a standalone power sourse.



Take IoT:bit for an example:



Note:

1. If you do not use Nezha expansion board to drive the AI Lens, please search with the package through this link: https://github.com/elecfreaks/pxt-PlanetX-AI
2. Under the one button to acquire mode, you will have to re-learn the objects if you restart the device.
3. After downloading the code, if the AI lens gets stuck in the starting page with an init AI Lens error instruction, please try to power off the device and restart it. (Or make sure you connect the ready-programmed micro:bit board after the AI Lens connects the breakout board well)

AI Lens firmware update

For a better experience of the AI Lens, we are making an irregularly update for the new functions and optimization of the performance, a manually update of the firmware is required after the new version is released.

How to check the current version



The current version number shall be displayed on the lower right corner of the boot screen(starting-on page), if there is no prompt, it should be the oldest version, please update asap.

Latest version firmware

firmware v1.0.12 click to download。

The newest firmware version

1. First, installation of camera serial port driver

Windows system downloading CH341SerSetup.exe

macOS system downloading CH34x_Install_V1.5.pkg

1. Install “kflash” for downloading the firmware.

Windows system downloading kflash_gui_v1.6.5_2_windows.7z and unzipping it.

macOS system downloading kflash_gui_v1.6.5_2_macOS.dmg

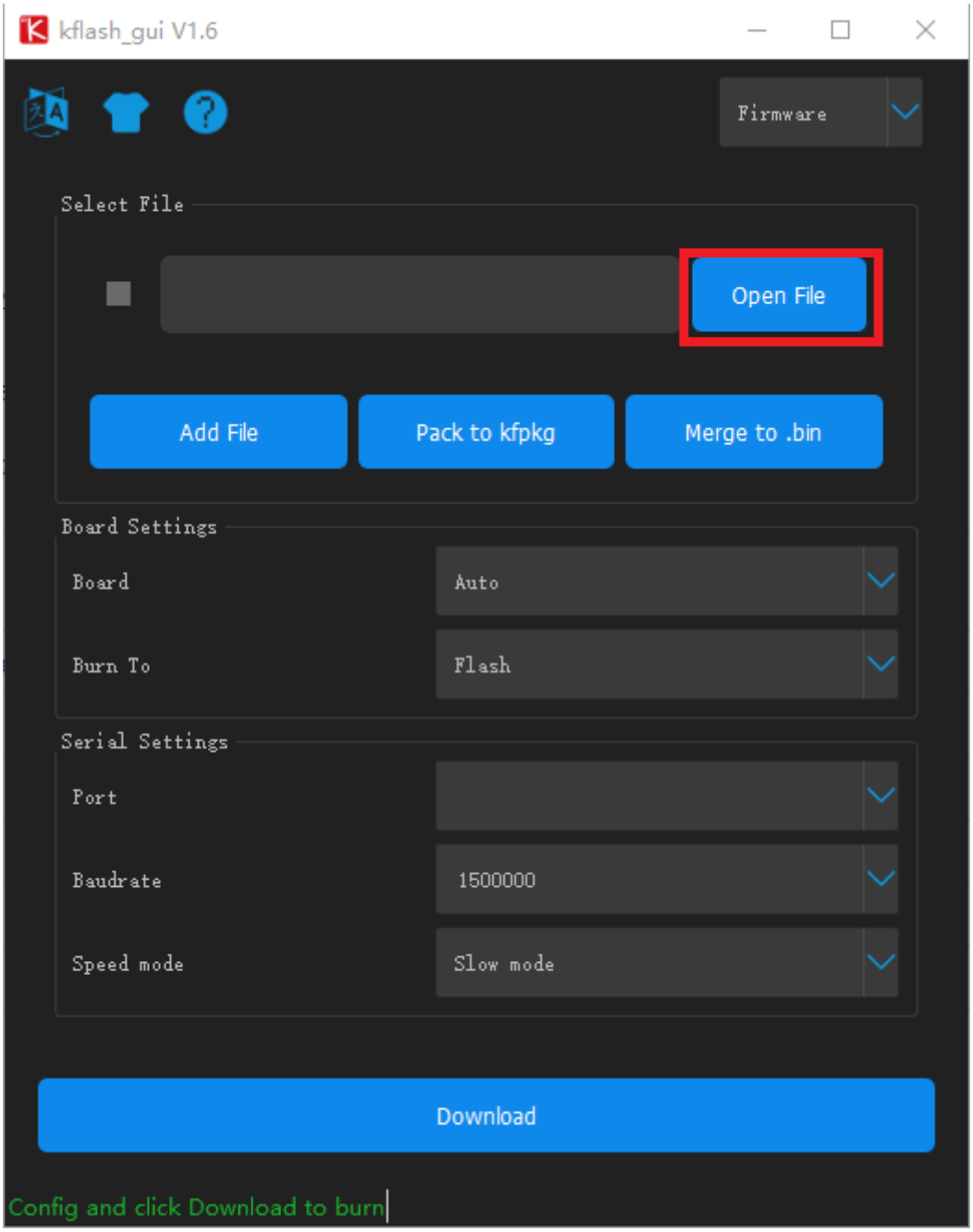
For windows system, open the file folder kflash_gui and find kflash_gui.exe.

api-ms-win-crt-stdio-l1-1-0.dll	2020/7/29 18:41	应用程序扩展	27 KB
api-ms-win-crt-string-l1-1-0.dll	2020/7/29 18:41	应用程序扩展	27 KB
api-ms-win-crt-time-l1-1-0.dll	2020/7/29 18:41	应用程序扩展	23 KB
api-ms-win-crt-utility-l1-1-0.dll	2020/7/29 18:41	应用程序扩展	21 KB
base_library.zip	2020/7/29 18:41	WinRAR ZIP 压缩...	760 KB
d3dcompiler_47.dll	2020/7/29 18:41	应用程序扩展	4,077 KB
kflash_gui.conf	2021/1/5 17:49	CONF 文件	1 KB
kflash_gui.exe	2020/7/29 18:41	应用程序	1,811 KB
kflash_gui.exe.manifest	2020/7/29 18:41	MANIFEST 文件	2 KB
libcrypto-1_1-x64.dll	2020/7/29 18:41	应用程序扩展	2,426 KB
libEGL.dll	2020/7/29 18:41	应用程序扩展	24 KB
libGLESv2.dll	2020/7/29 18:41	应用程序扩展	3,290 KB
libssl-1_1-x64.dll	2020/7/29 18:41	应用程序扩展	518 KB
MSVCP140.dll	2020/7/29 18:41	应用程序扩展	577 KB
MSVCP140_1.dll	2020/7/29 18:41	应用程序扩展	31 KB
opengl32sw.dll	2020/7/29 18:41	应用程序扩展	20,433 KB

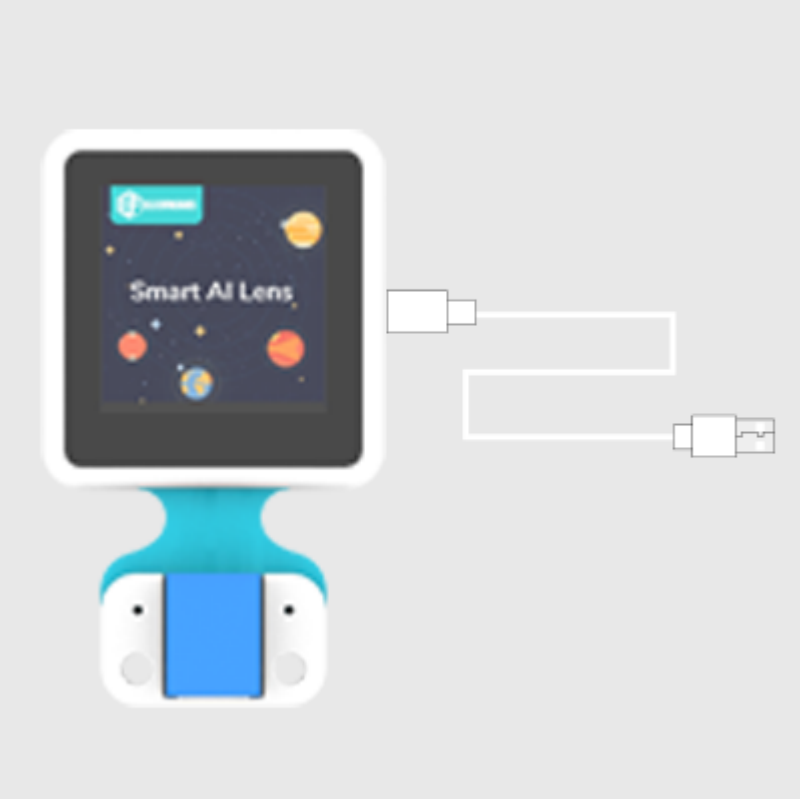
For macOS system, click the icon of kflash_gui after the downloading.



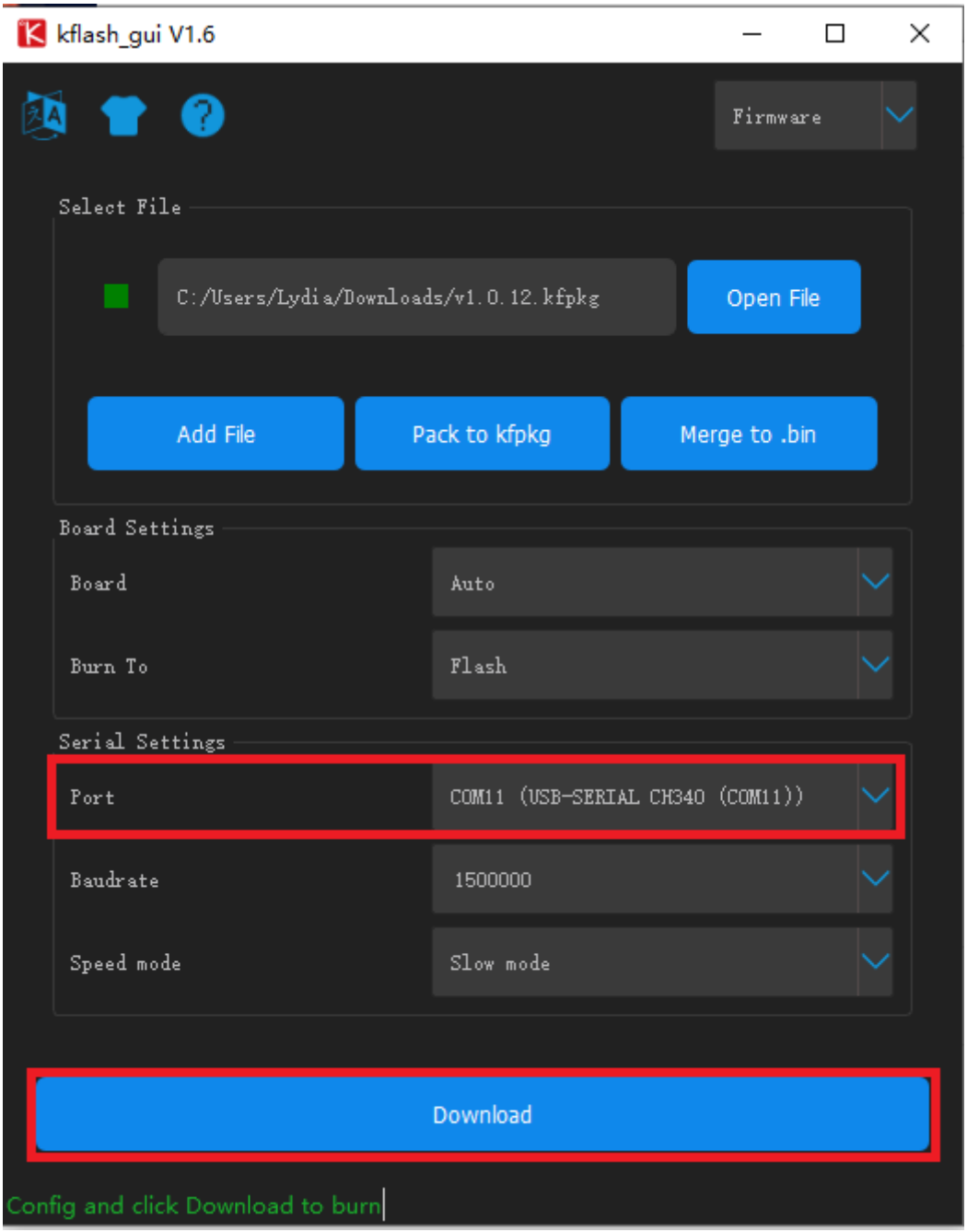
Double click kflash_gui.exe to open the file and choose to open the newest firmware.



Connect the AI Lens with the USB wire.

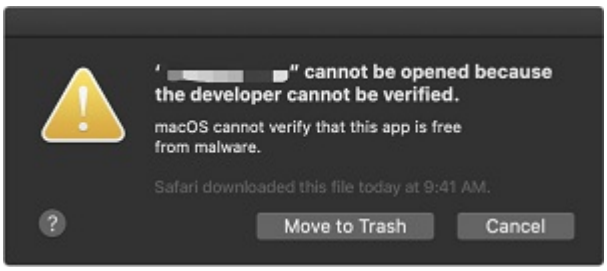


Choose the port to download it.



FAQ

In macOS operation system, if you fail to install the firmware because the developer cannot be verified.



Please visit https://support.apple.com/en-us/HT202491 for solutions.

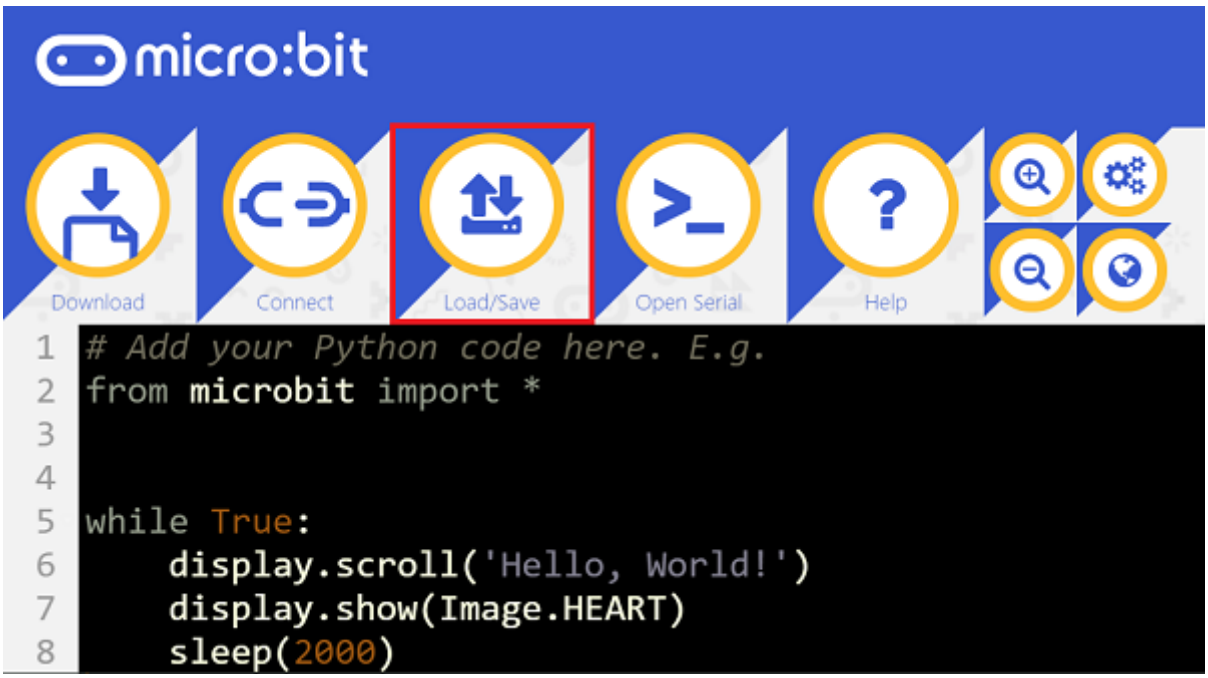
1.2. AI-Lens-Python Samples Code

Add Python File

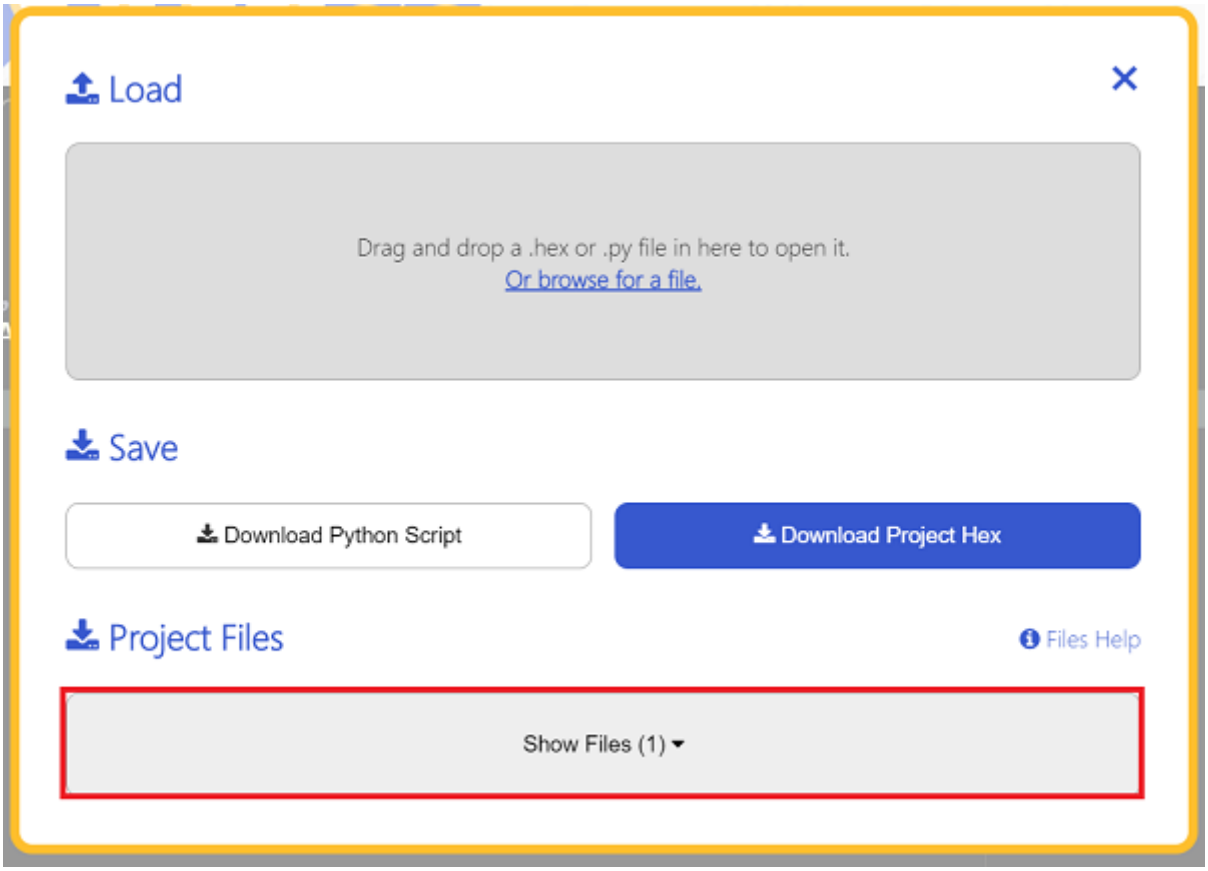
We've created a codebase EF_Produce_MicroPython-master for a convenient python programming to AI Lens, you will only need to use the functions and modify the parameters for application.

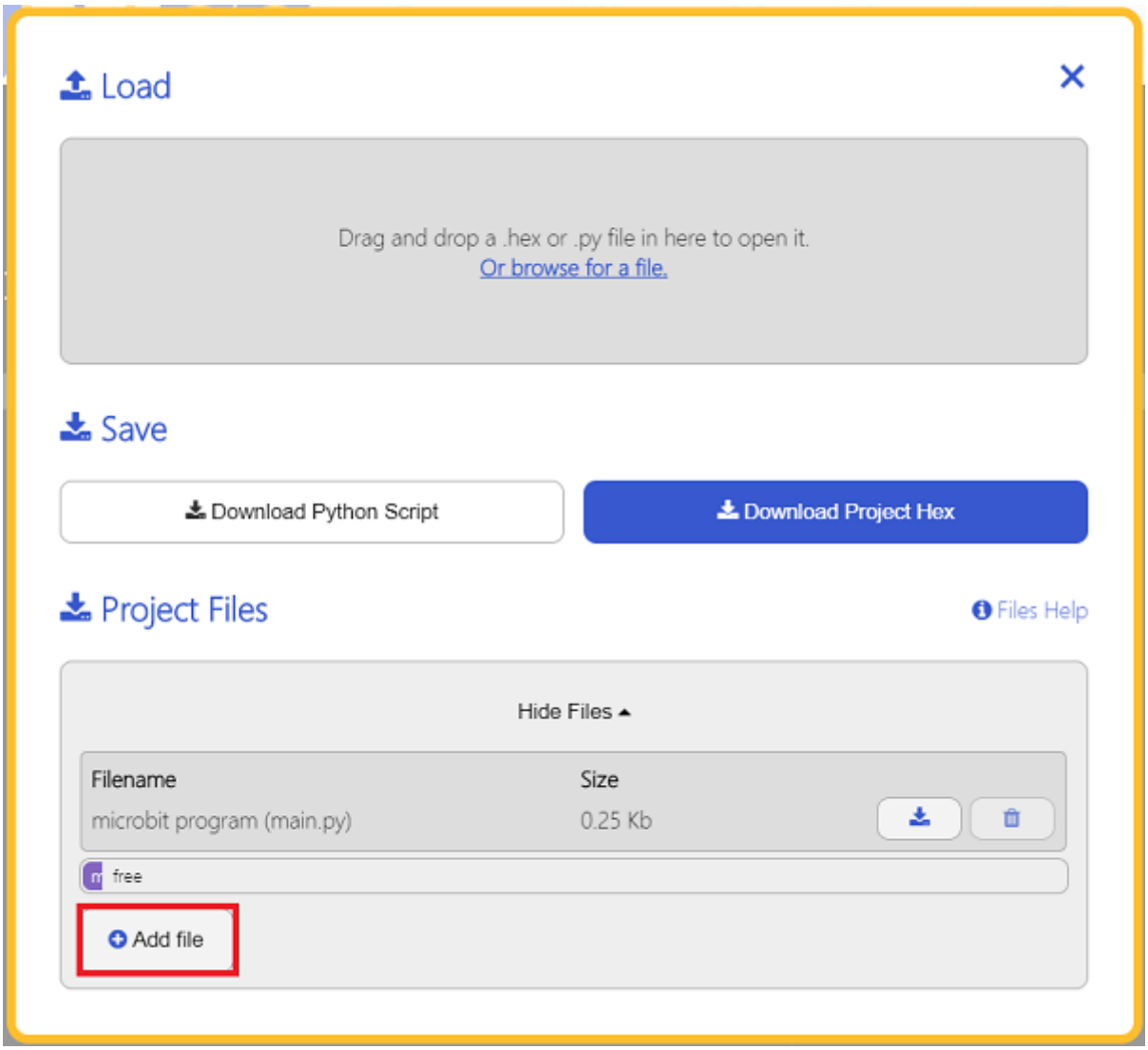
Download the package and unzip it: EF_Produce_MicroPython-master

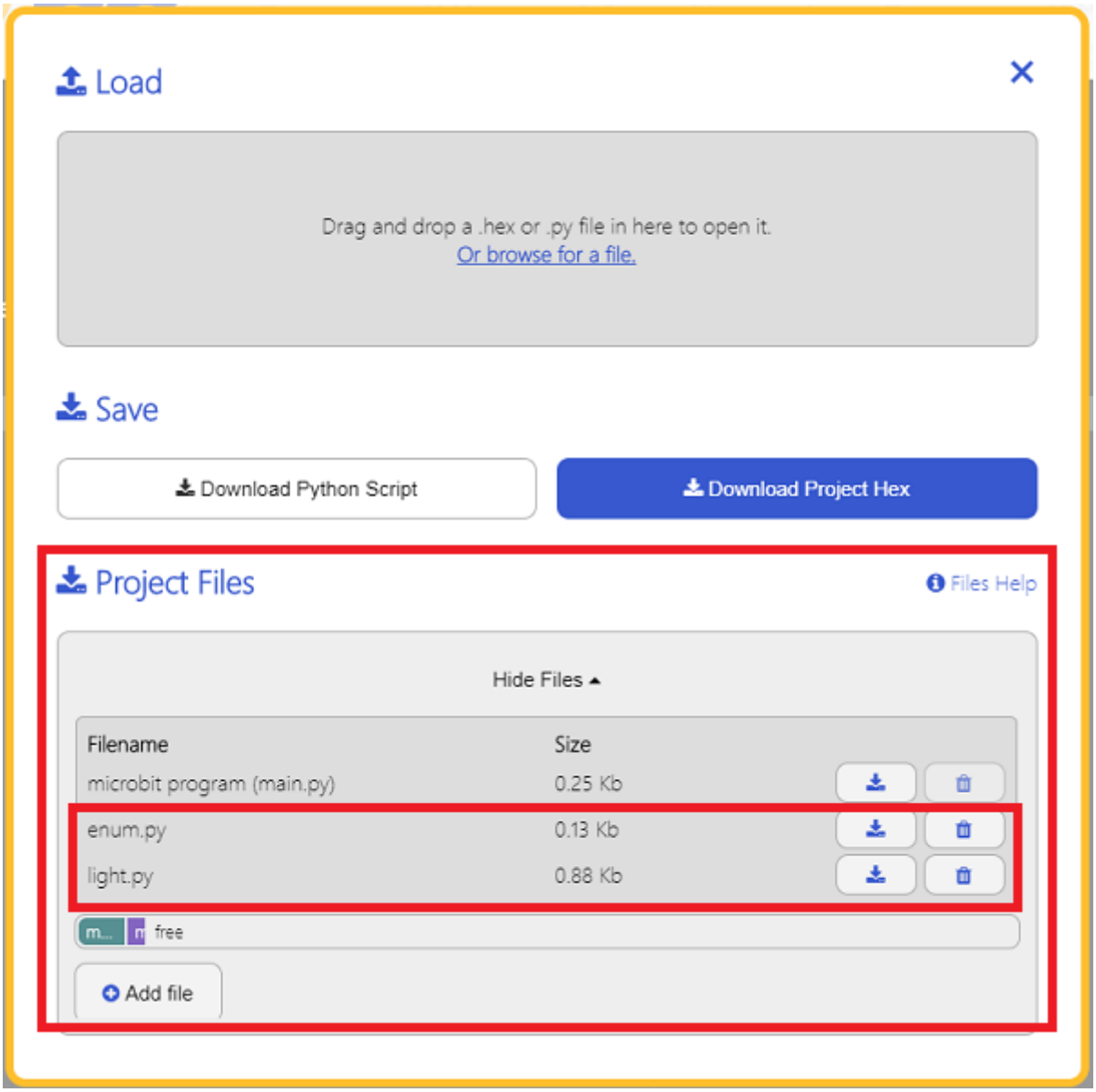
Go to Python editor



For programming, we need to add the file of AILens.py. Click Load/Save first and again with Show Files(1), choose "add file" to get to the download page to find the file folder of EF_Produce_MicroPython-master, then add AILens.py.


Load
Drag and drop a .hex or .py file in here to open it.
Or browse for a file.
Save
Download Python Script
Download Project Hex
Project Files
Files Help
Show Files (1)


Load
Drag and drop a .hex or .py file in here to open it.
Or browse for a file.
Save
Download Python Script
Download Project Hex
Project Files
Files Help
Hide Files
Filename
Size
microbit program (main.py)
0.25 Kb
free
Add file



API

API	
`AILENS()`	
`switch_function(func)`	Choose AI Lens functions, `func` choose function: (`Learn` charac
`get_card_content()`	
`get_card_data()`	Get the info of the cards from th
`get_face()`	
`get_face_data()`	Get the info of the face(s) from the
`get_ball_color()`	
`get_ball_data()`	Get the info of the ball(s) from the
`get_track_data()`	Get the info o

`get_color_type()`	
`get_color_data()`	Get the info of the color(s) from the
`learn_object(learn_id)`	
`get_learn_data()`	Get the in

`class AILENS(object)`

Init AI Lens

`def switch_function(self, func)`

Choose AI Lens functions, `func` choose function, `Learn` characteristics learn, `Card` card recognition； `Face` face recognition； `Tracking` tracking recognition； `Color` color recognition； `Ball` ball recognition）

`def get_image(self)`

Get an image

`def get_ball_color(self):`

Recognize the colors of the balls in the Ai Lens(If ball is blue, the returned result is 'Blue'; if the ball is red, the result is 'Red'; or it returned with a 'No Ball')

`def get_ball_data(self)`

Return the info of the balls in the AI Lens，BallData [x,y,w,h,confidence,total,order]。

x：X-axis

y：Y-axis

w：width

h：height

confidence：Confidence coefficient

total：total numbers of the balls in the AI Lens

order：ID of the current ball

`def get_face(self)`

Judge if any face(s) in the Ai Lens

```
def get_face_data(self)
```

Return the info of human face in the AI lens, FaceData [x,y,w,h,confidence,total,order]。

x：X-axis

y：Y-axis

w：width

h：height

confidence：Confidence coefficient

total：total numbers of the face(s) in the AI Lens

order：ID ID of the current face

```
def get_card_content(self)
```

Return the contents on the cards, the possible value are:

numberCards = [“0”, “1”, “2”, “3”, “4”, “5”, “6”, “7”, “8”, “9”]

letterCards = [“A”, “B”, “C”, “D”, “E”]

otherCards = [“Mouse”, “micro:bit”, “Ruler”, “Cat”, “Peer”, “Ship”, “Apple”, “Car”, “Pan”, “Dog”, “Umbrella”, “Airplane”, “Clock”, “Grape”, “Cup”, “Turn left”, “Turn right”, “Forward”, “Stop”, “Back”]

If no cards get recognized, the returned value is ‘No Card’

```
def get_color_type(self)
```

Return the colors of the cards, the possible colors are:

Green card：”Green”

Blue card：”Blue”

Yellow card：”Yellow”

Black card： “Black”

Red card： “Red”

White card： “White”

If no colors get recognized, the returned value is ‘No Color’

```
def get_color_data(self)
```

Return the info of the color(s), ColorData [x,y,w,h,confidence,total,order]

x：X-axis

y：Y-axis

w： width

h：height

confidence：Confidence coefficient

total：total numbers of the color(s) in the AI Lens

order： ID of the current color

```
def get_track_data(self)
```

Return the info of the segment(s) LineData [angel,width,len]

angel：return the deviation angel of the segment in the AI Lens

width： return the deviation distance of the segment in the AI Lens

len： return the length of the segment in the AI Lens

```
def learn_object(self, learn_id)
```

Name with an ID number for a learnt object

ID of the learnt object(s): learn_id

```
def get_learn_data(self)
```

Return the info of the learnt object(s), return info: LearnData [ID,confidence]

Object(s) ID： ID

Confidence coefficient of the learnt object： confidence

Samples

Sample 1: Characteristics learn

```
from microbit import *
from AILens import *
ai = AILENS()
ai.switch_function(Learn)
while True:
    ai.get_image()
    if button_a.is_pressed():
        ai.learn_object(1)
    buff = ai.get_learn_data()
    if buff[0] == 1:
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Result

- Press button A to learn the object, a smile face displays on the micro:bit while it recognizes the object or it displays a sad face.

Sample 2: Color recognition

```
from microbit import *
from AILens import *
ai = AILENS()
ai.switch_function(Color)
while True:
    ai.get_image()
    if (ai.get_color_type() == "Green"):
        display.show(Image.HAPPY)
        sleep(2000)
    else:
        display.show(Image.SAD)
```

Result

- A smile face displays on the micro:bit for two seconds if the green card is recognized or it displays a sad face.

Sample 3: Balls recognition

```
from microbit import *
from AILens import *
ai = AILENS()
ai.switch_function(Ball )
while True:
    ai.get_image()
    if (ai.get_ball_color() == "Red"):
        display.show(Image.HAPPY)
        sleep(2000)
    else:
        display.show(Image.SAD)
```

Result

- A smile face displays on the micro:bit for two seconds if the red ball is recognized or it displays a sad face.

Sample 4: Recognize the black line

```
from microbit import *
from AILens import *
ai = AILENS()
ai.switch_function(Tracking)
while True:
    ai.get_image()
    buff = ai.get_track_data()

    if buff[1] > 100:
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Result

- The micro:bit displays a sad face if the black line was deteacted deviating to the left by the AI lens; It displays a smile face if the black line was deteacted deviating to the right by the AI lens

Sample 5: Human face(s) recognition

```
from microbit import *
from AILens import *
ai = AILENS()
ai.switch_function(Face )
while True:
    ai.get_image()

    if ai.get_face():
        display.show(Image.HAPPY)
    else:
        display.show(Image.SAD)
```

Result

- A smile face displays on the micro:bit if the human face is detected or it displays a sad face.

Sample 6: Cards recognition

```
from microbit import *
from AILens import *
ai = AILENS()
ai.switch_function(Card)
while True:
    ai.get_image()

    if (ai.get_card_content() == "Stop"):
        display.show(Image.HAPPY)
        sleep(2000)
    else:
        display.show(Image.SAD)
```

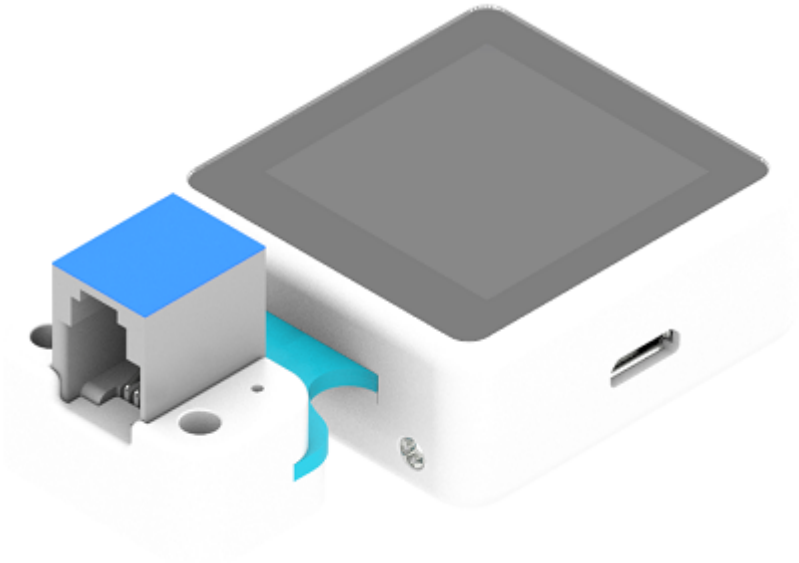
Result

- A smile face displays on the micro:bit if a “stop” card is detected or it displays a sad face.

1.3. Case 01: Password Recognition

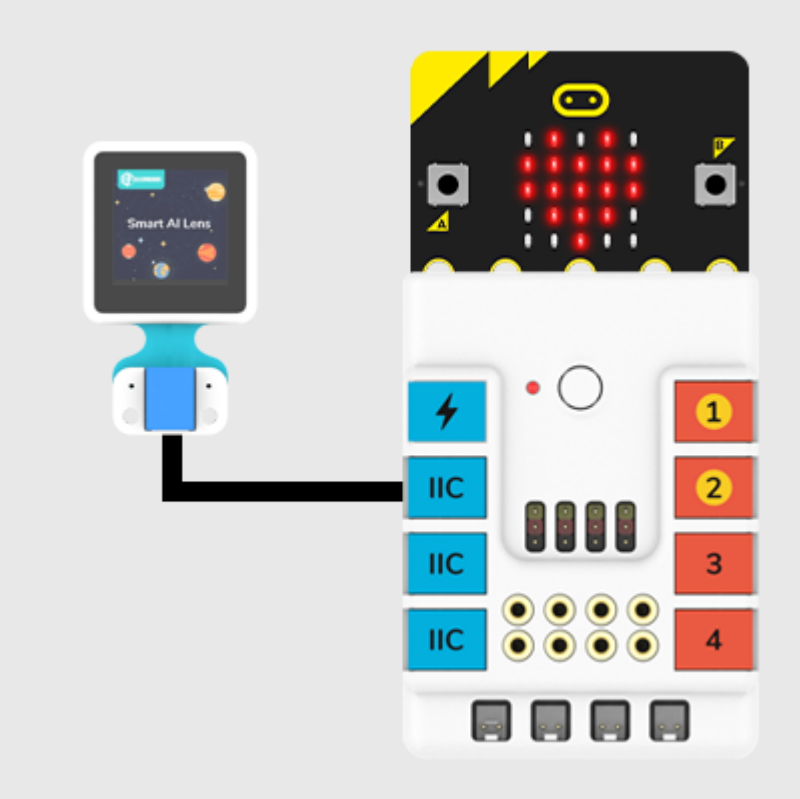
Purpose

To make the AI Lens be able to recognize the password on the card.



Materials Required and Connection Diagram

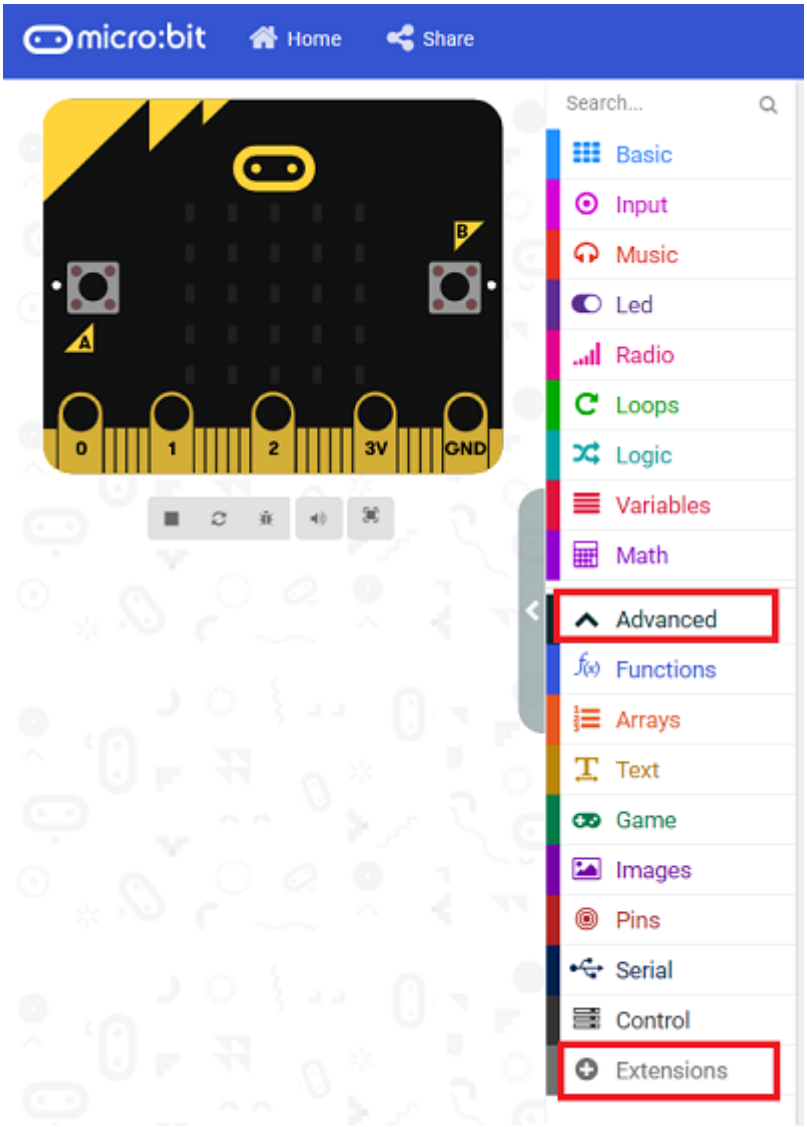
- Connect the AI Lens to the IIC port on the Nezha expansion board as the picture shows.



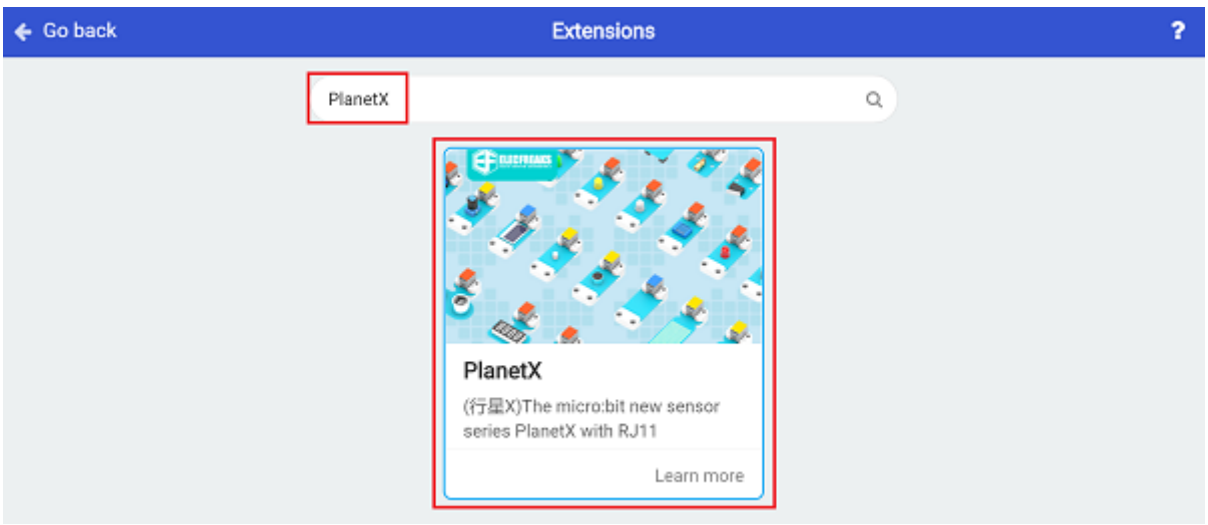
MakeCode Programming

Step 1

Click “Advanced” in the MakeCode drawer to see more choices.



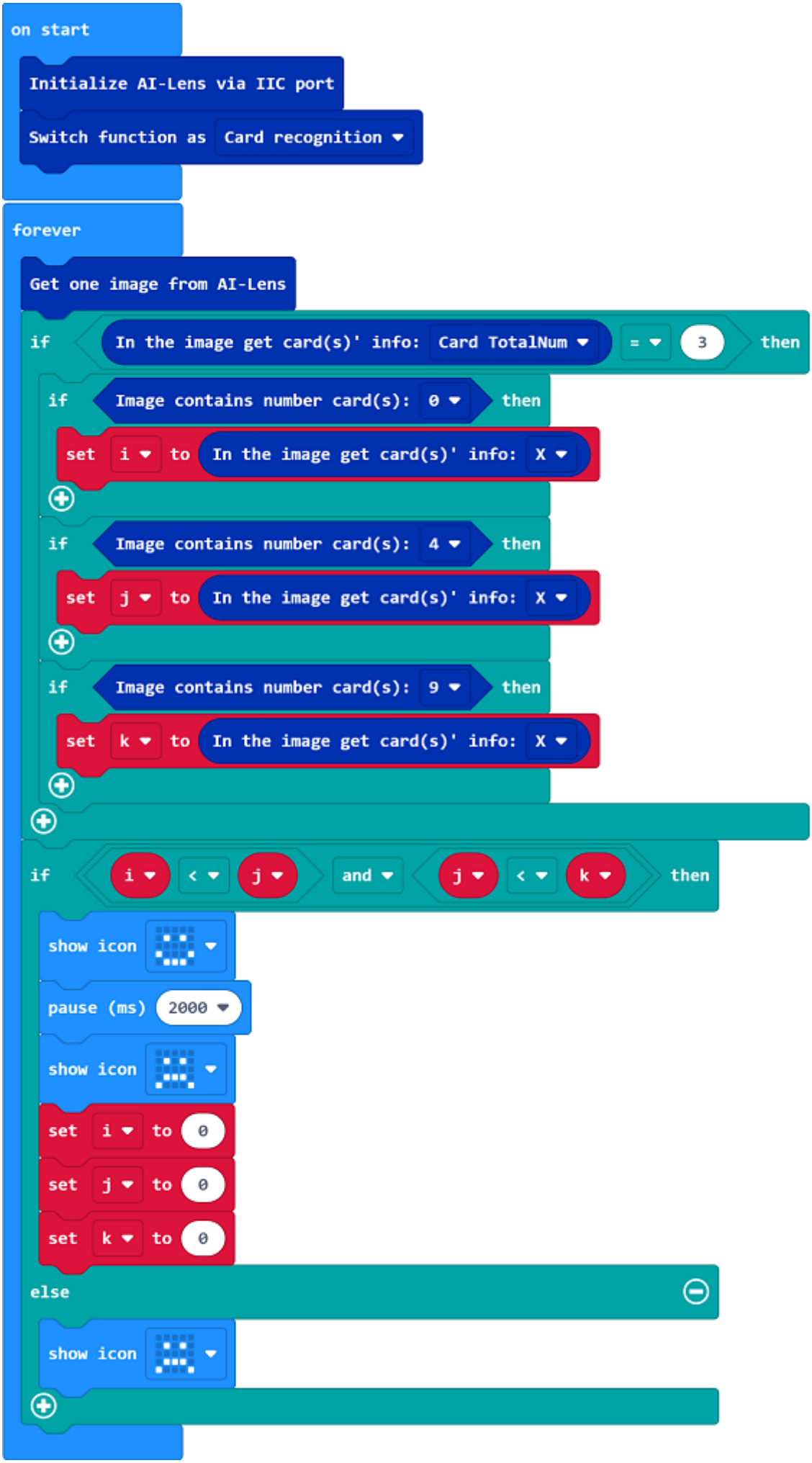
We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “PlanetX” in the dialogue box to download it.



Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.

Step 2

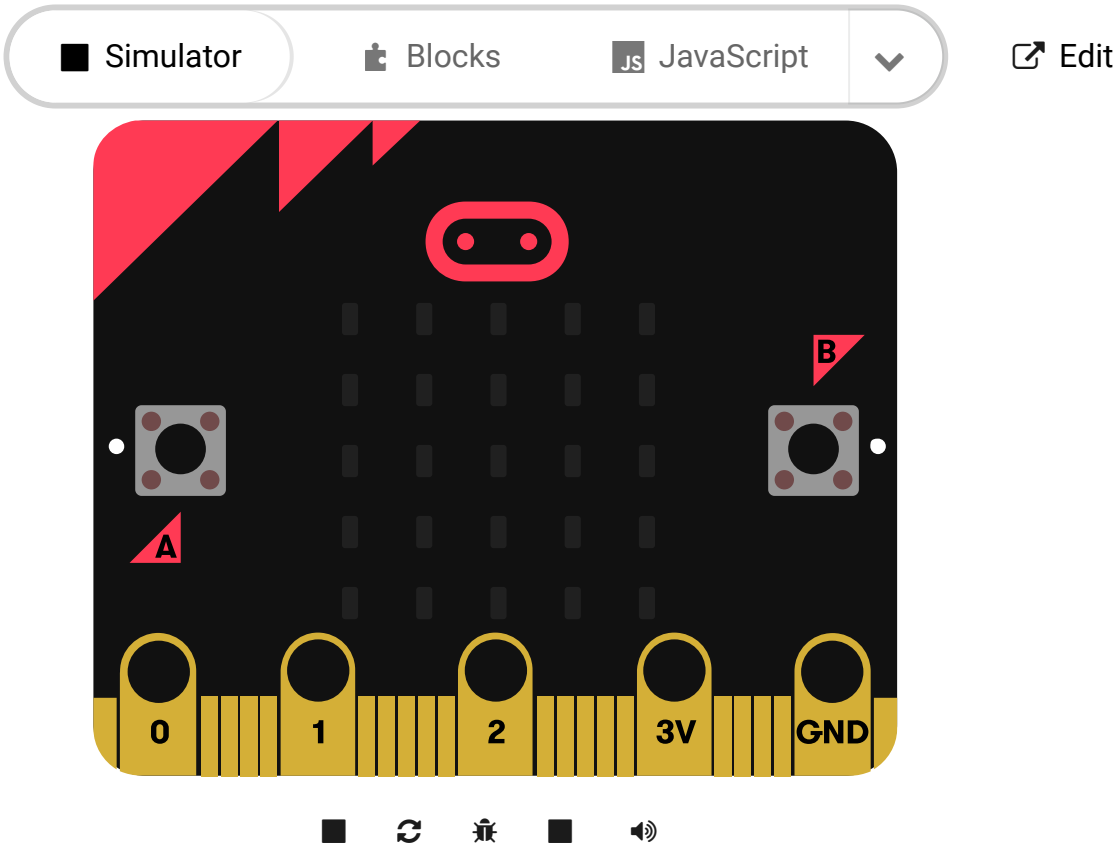
Code as below:


on start
Initialize AI-Lens via IIC port
Switch function as Card recognition
forever
Get one image from AI-Lens
if In the image get card(s)' info: Card TotalNum = 3 then
if Image contains number card(s): 0 then
set i to In the image get card(s)' info: X
if Image contains number card(s): 4 then
set j to In the image get card(s)' info: X
if Image contains number card(s): 9 then
set k to In the image get card(s)' info: X
if i < j and j < k then
show icon
pause (ms) 2000
show icon
set i to 0
set j to 0
set k to 0
else
show icon

Link

Link: https://makecode.microbit.org/_RsmCMdCo1W3A

You may also download it directly below:



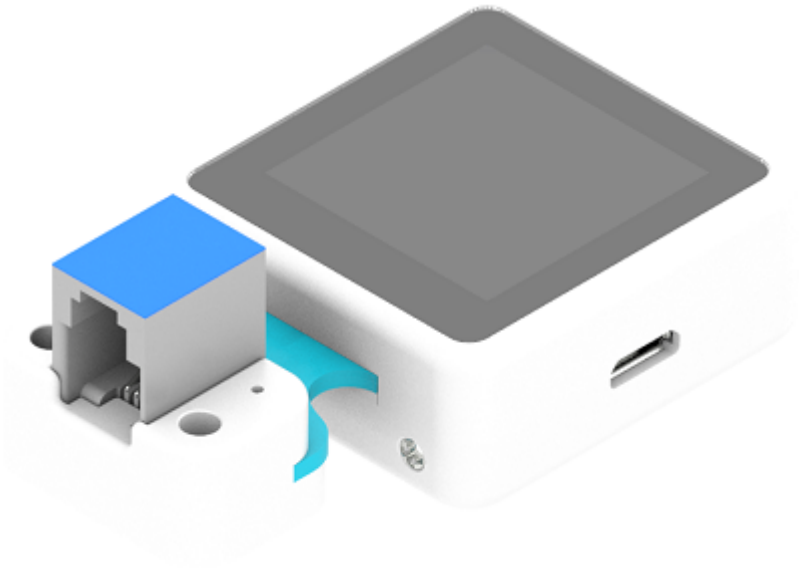
Result

- Place the cards named 0,4,9 accordingly in the front of the lens, a smile face should display on the micro:bit; or it displays a sad face.

1.4. Case 02: Color-controlled Lights

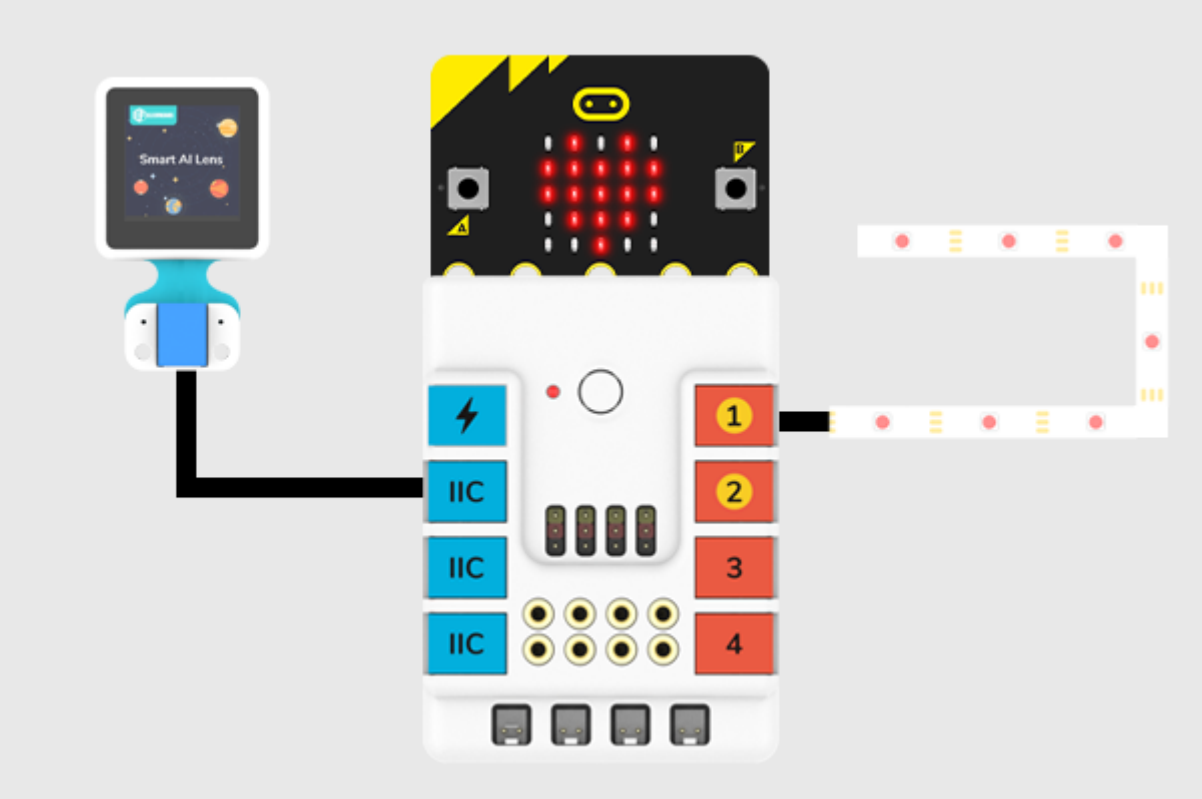
Purpose

To make the LED strip be able to adjust the color of the lights according to the recognized color from the AI Lens.



Materials Required and Connection Diagram

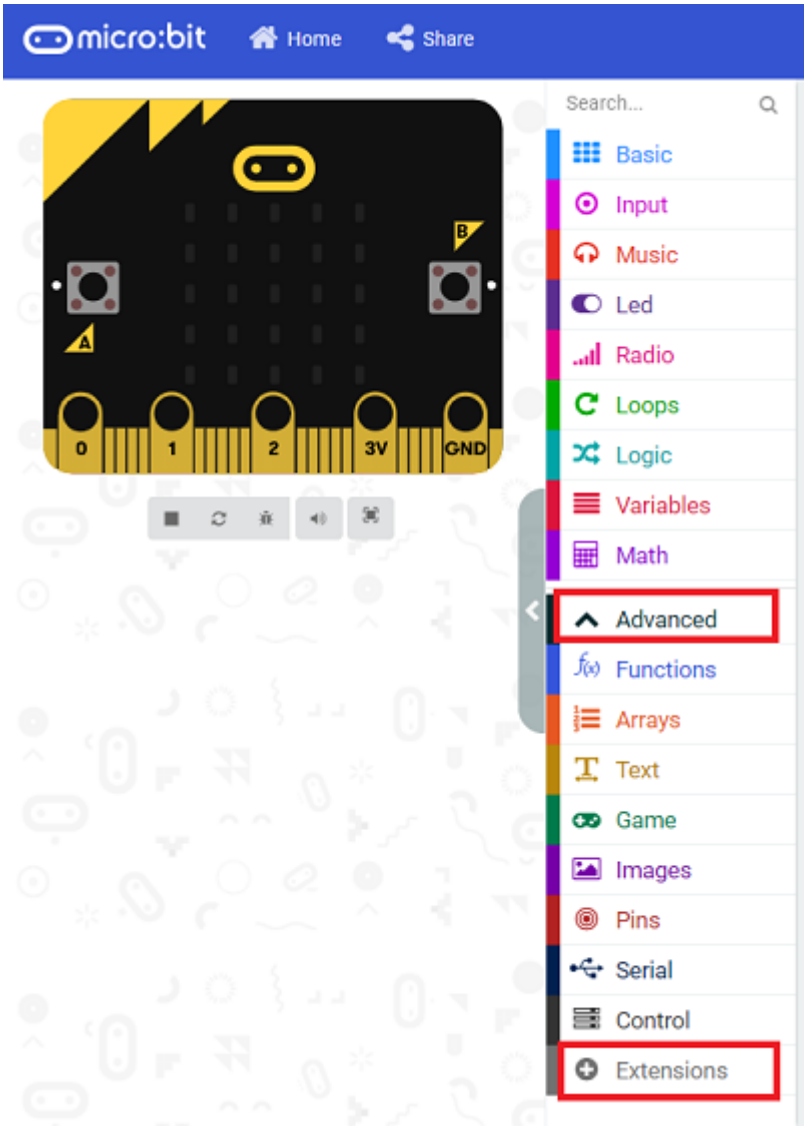
- Connect the AI Lens to the IIC port and the LED strip to the J1 port on the Nezha expansion board as the picture shows.



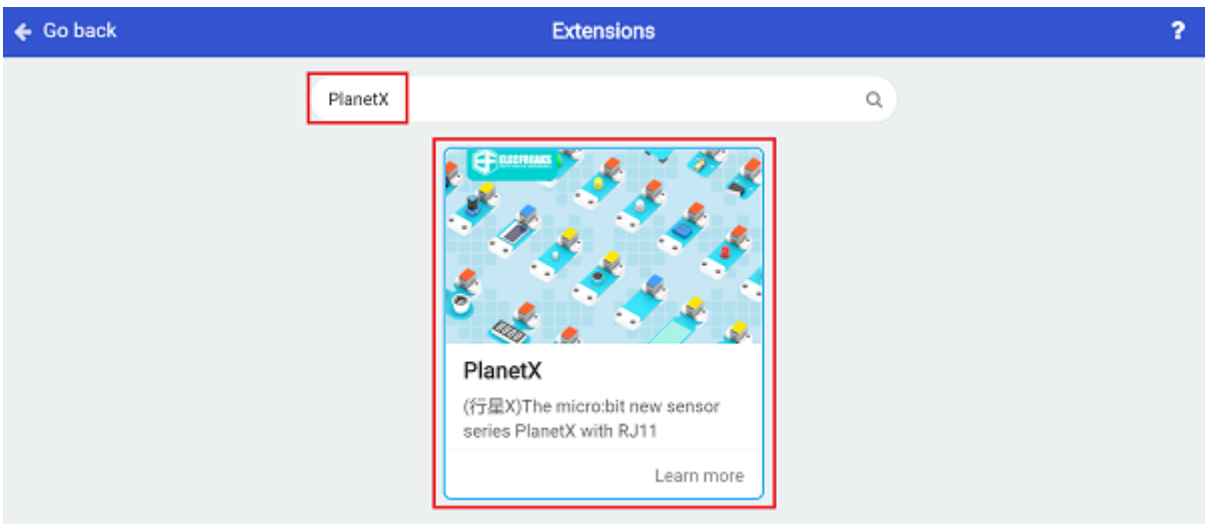
MakeCode Programming

Step 1

Click “Advanced” in the MakeCode drawer to see more choices.



We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “PlanetX” in the dialogue box to download it.



Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.

Step 2

Project 1: Card(s) Recognition

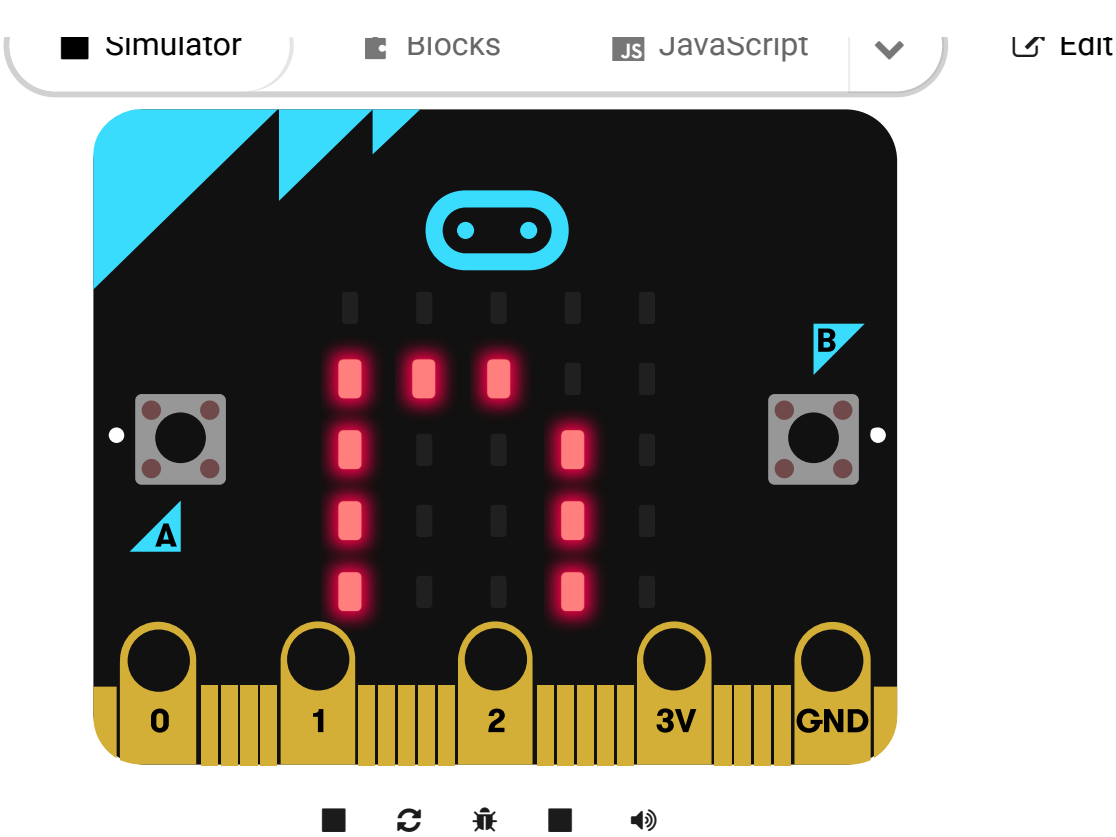
Code as below:



Link

Link: https://makecode.microbit.org/_iqM92mR7oL1d

You may also download it directly below:



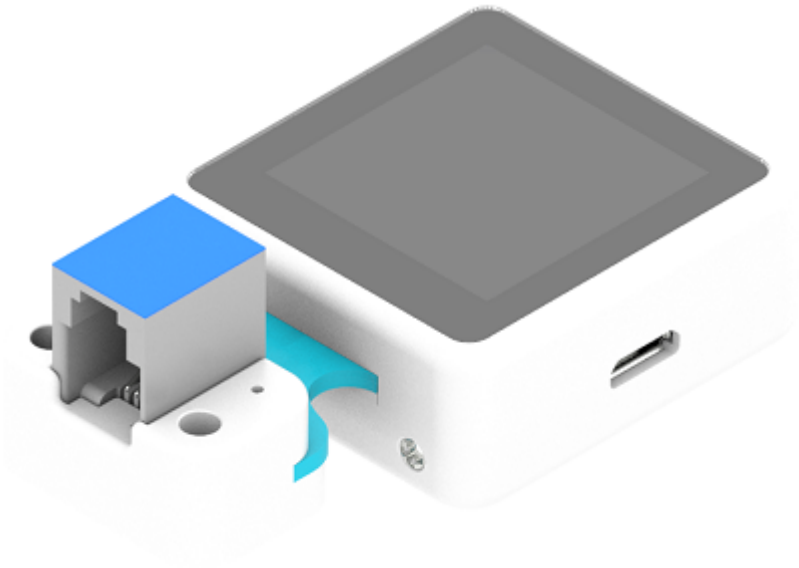
Result

- The LED strip displays the color in accordance with the color that is recognized from the AI Lens.

1.5. Case 03: Color-controlled Piano

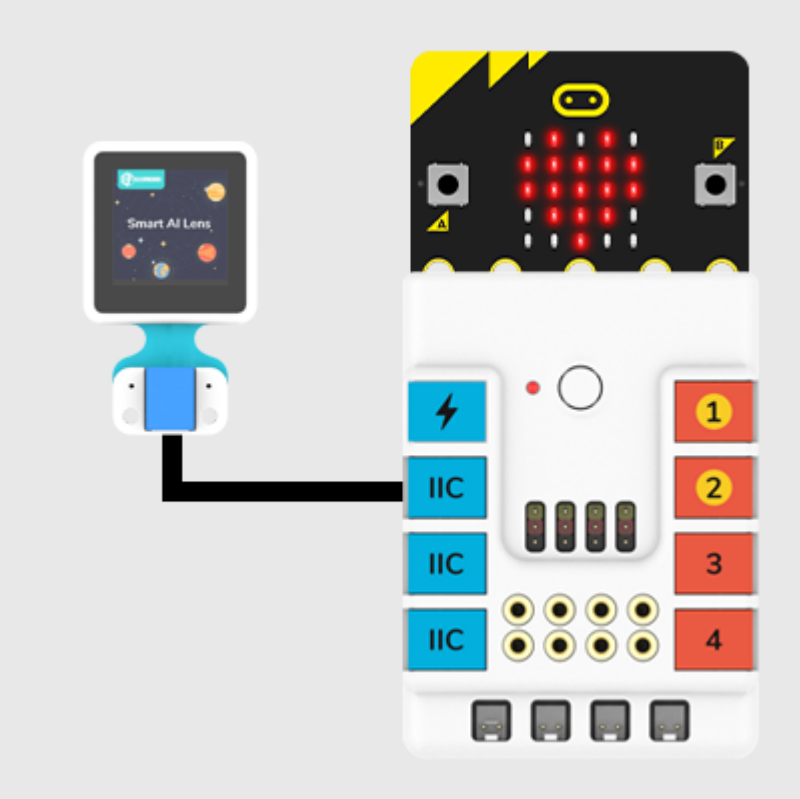
Purpose

To make the Nezha expansion board be able to play the different sounds according to the recognized color from the AI Lens.



Materials Required and Connection Diagram

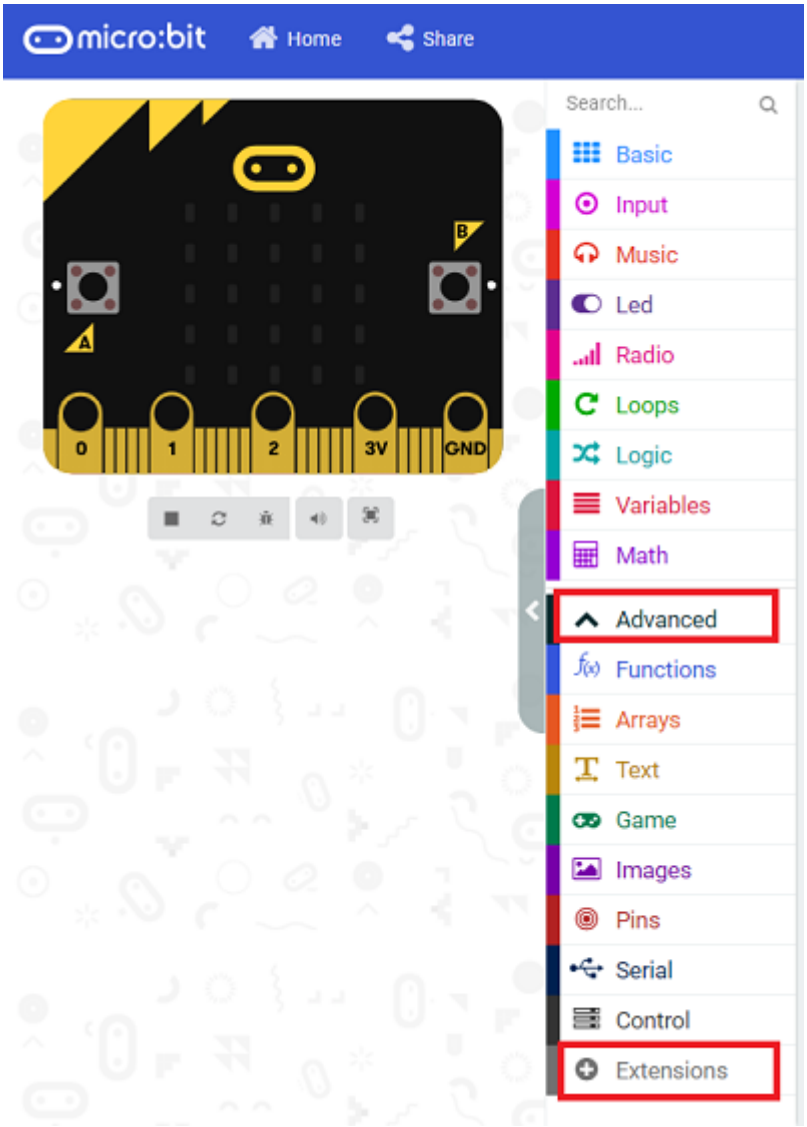
- Connect the AI Lens to the IIC port on the Nezha expansion board as the picture shows.



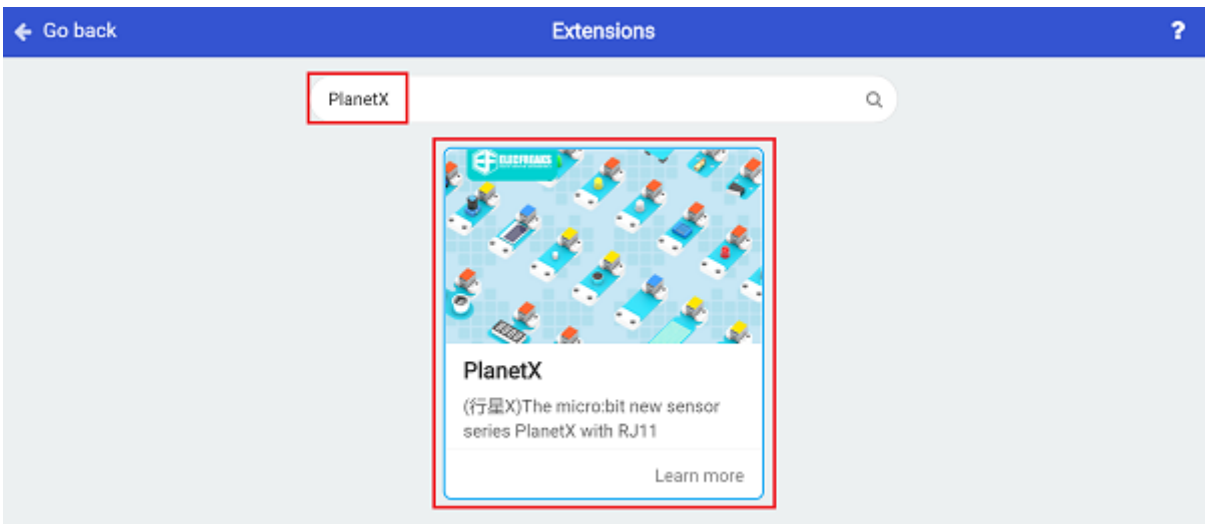
MakeCode Programming

Step 1

Click “Advanced” in the MakeCode drawer to see more choices.



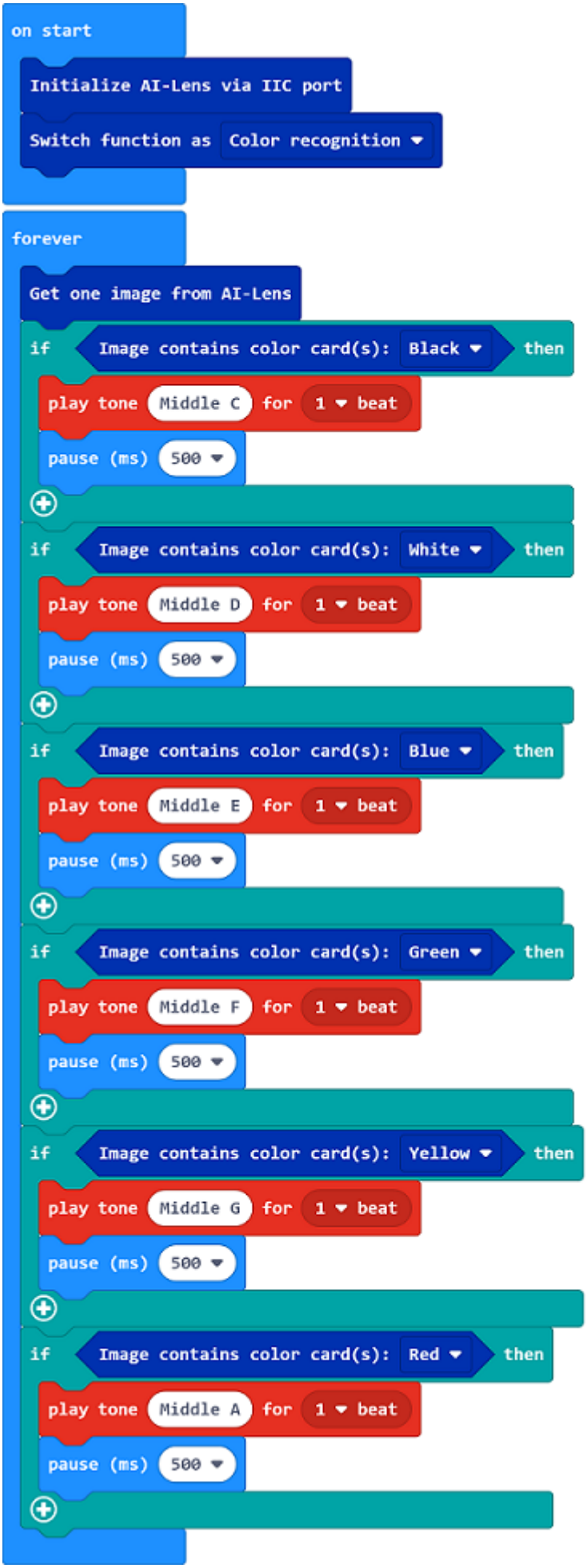
We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “PlanetX” in the dialogue box to download it.



Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.

Step 2

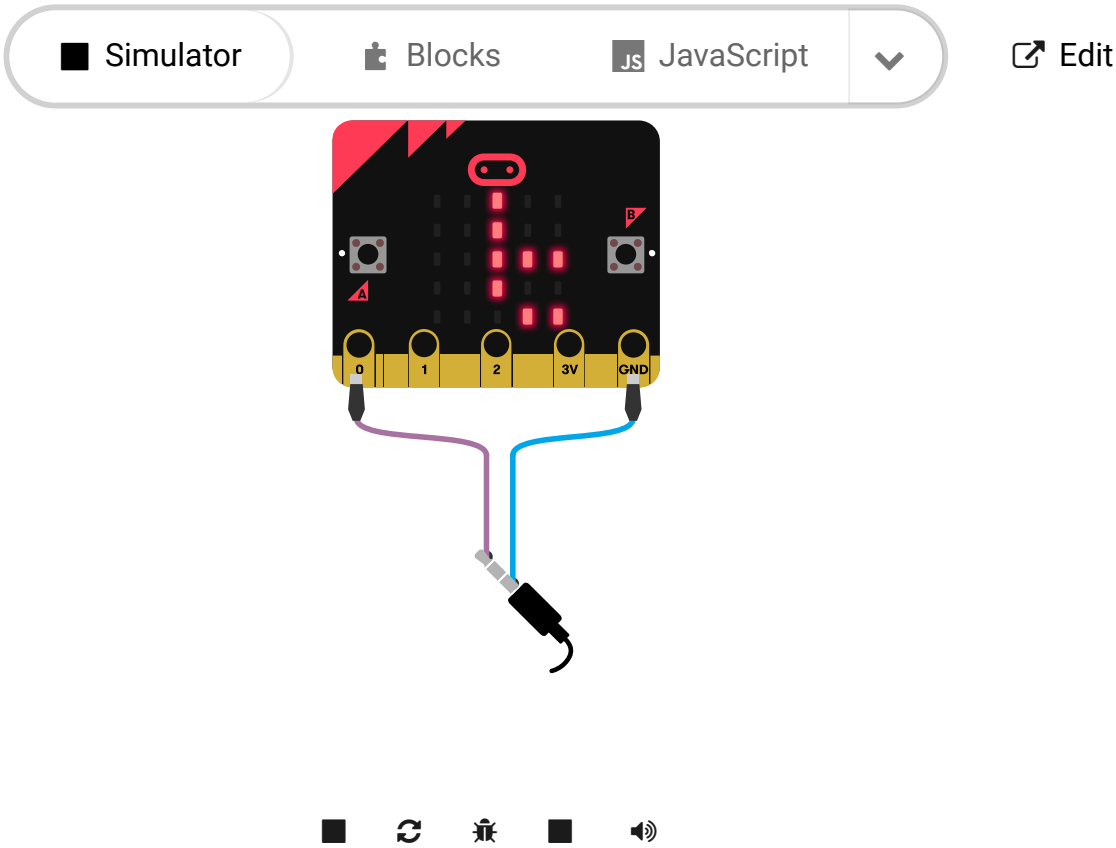
Code as below:



Link

Link: https://makecode.microbit.org/_8Vj2uH6Do59i

You may also download it directly below:



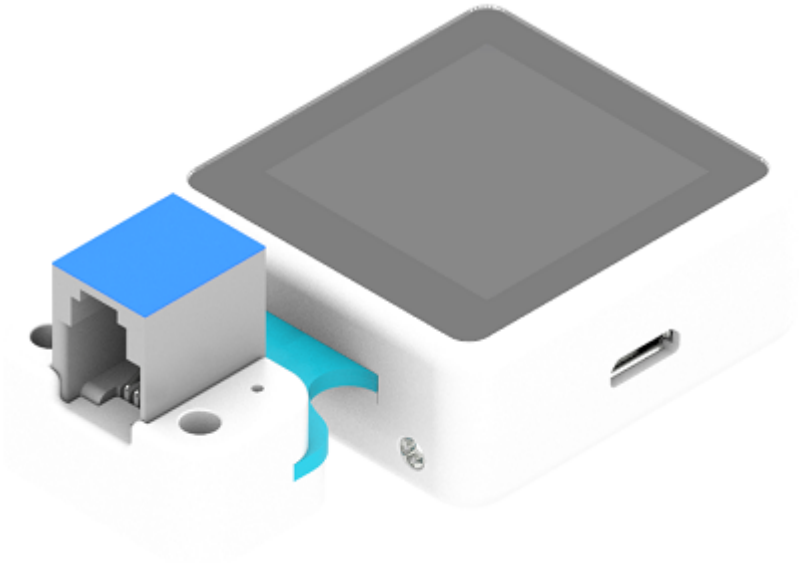
Result

- The buzzer on the Nezha expansion board plays different sounds aoccording to the different color recoginzed by the AI Lens.

1.6. Case 04: Face-tracked Device

Purpose

To make the AI Lens be able to recognize the face and track it.



Materials Required and Connection Diagram

- Build a device with the bricks as below:

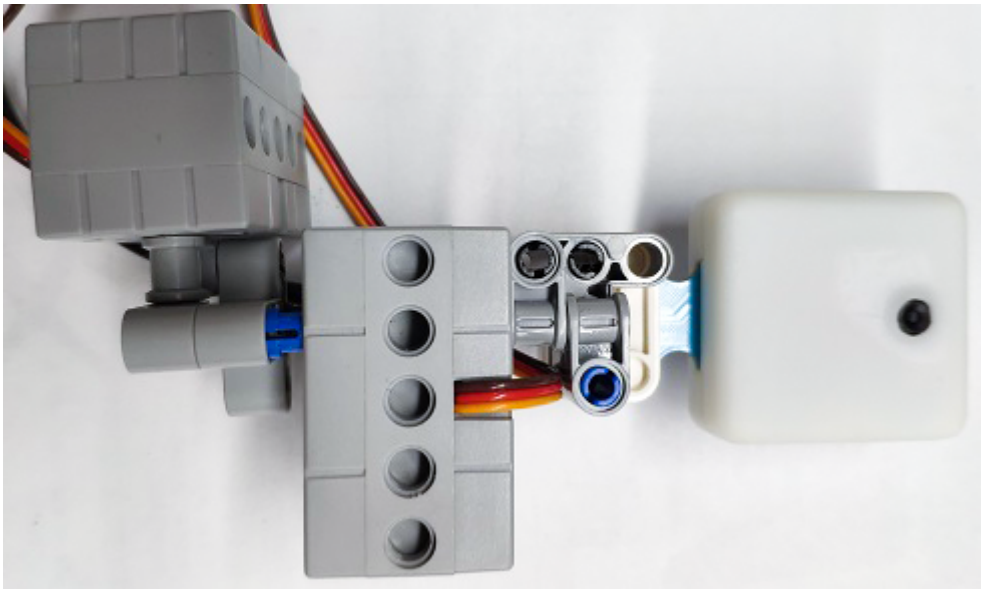




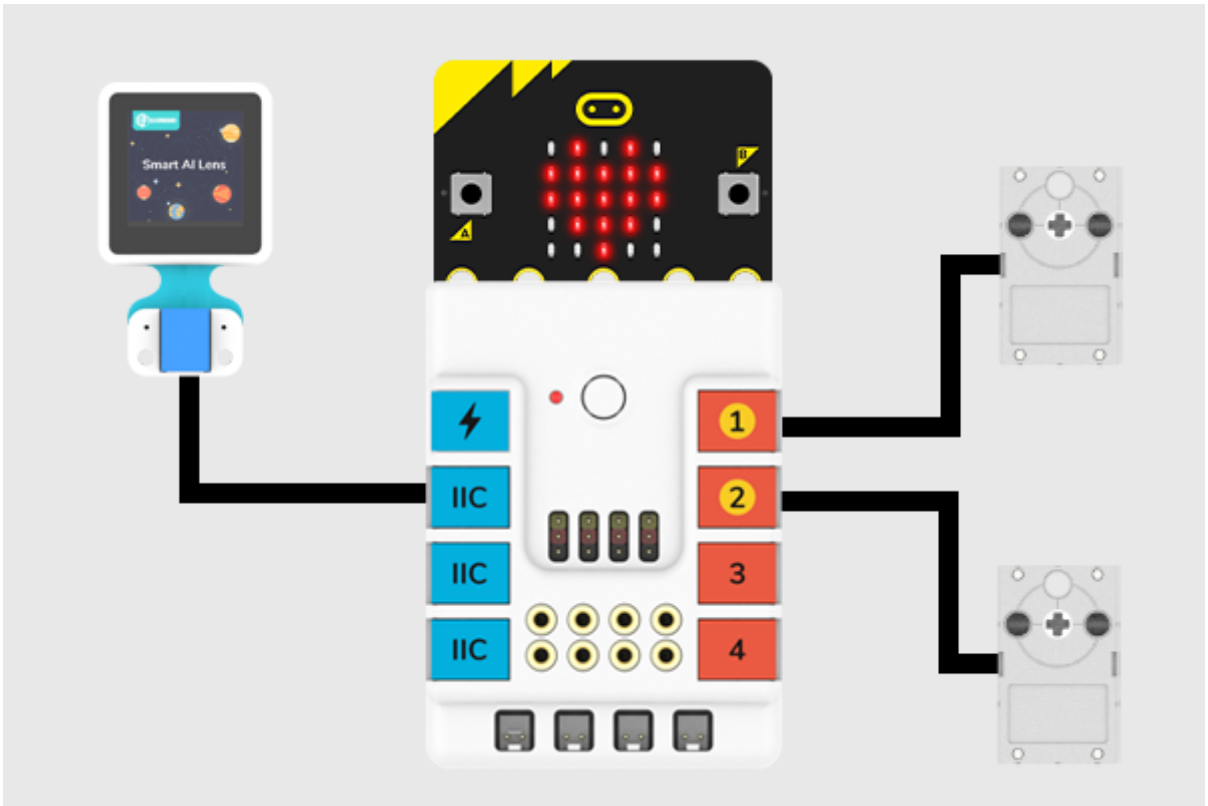
- Connect the device with a servo:



- Connect the AI Lens with the device:



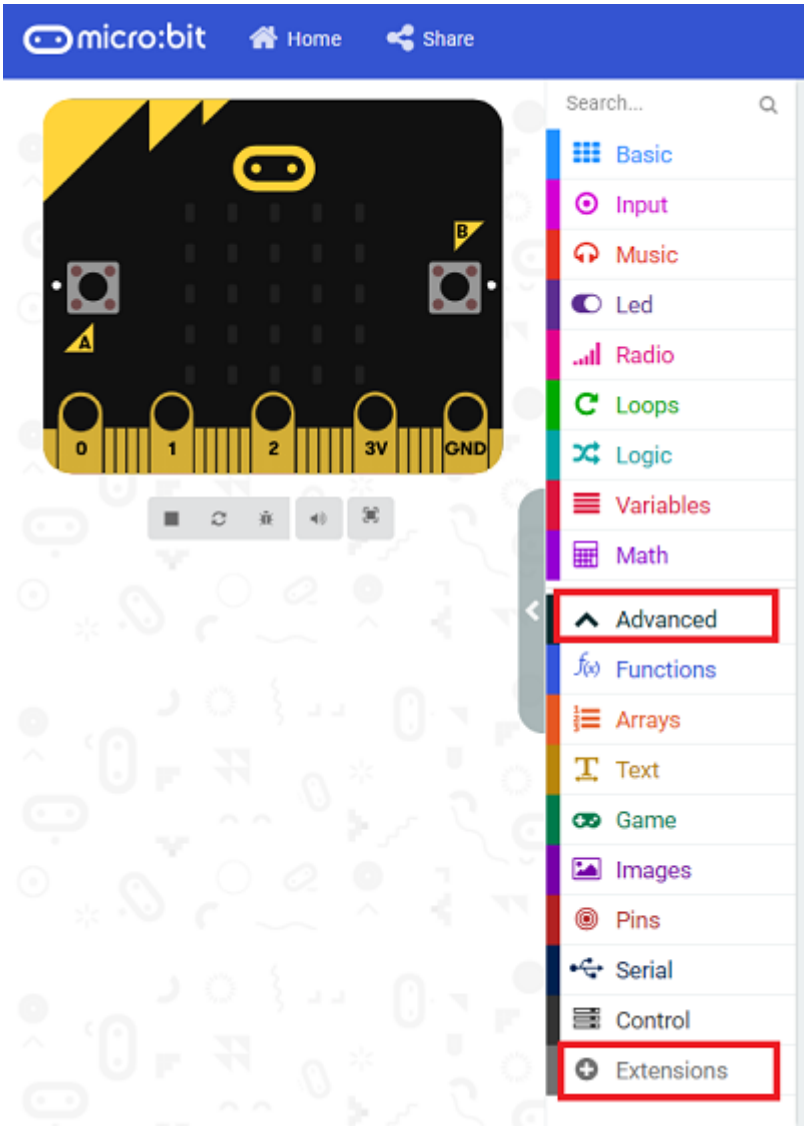
- Connect the AI Lens to the IIC port and the two servos to S1 and S2 separately.(Here S1 controls the movement of “X”, S2 controls the movement of “Y”)



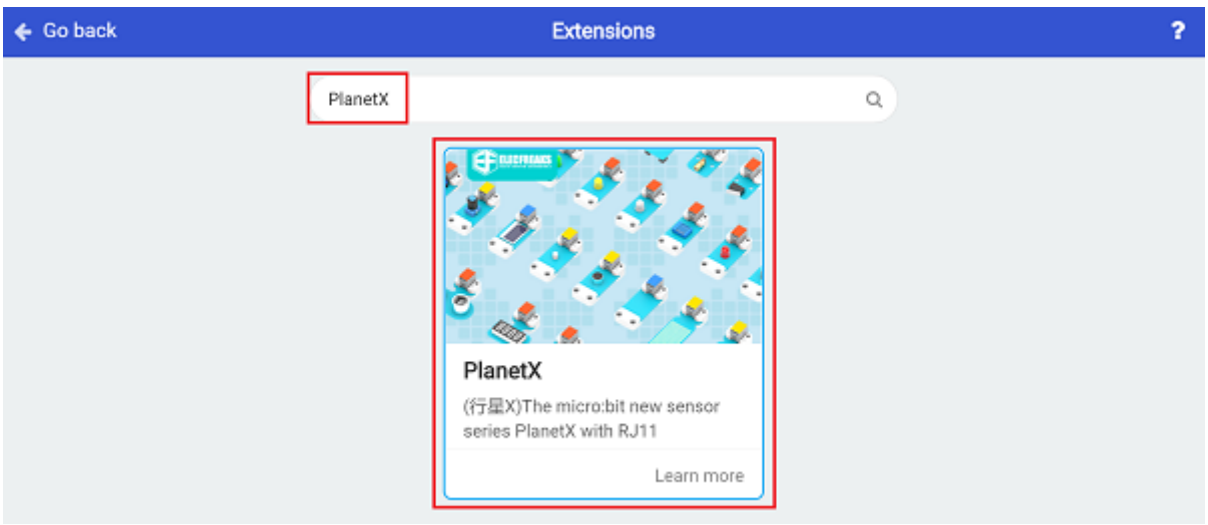
MakeCode Programming

Step 1

Click “Advanced” in the MakeCode drawer to see more choices.



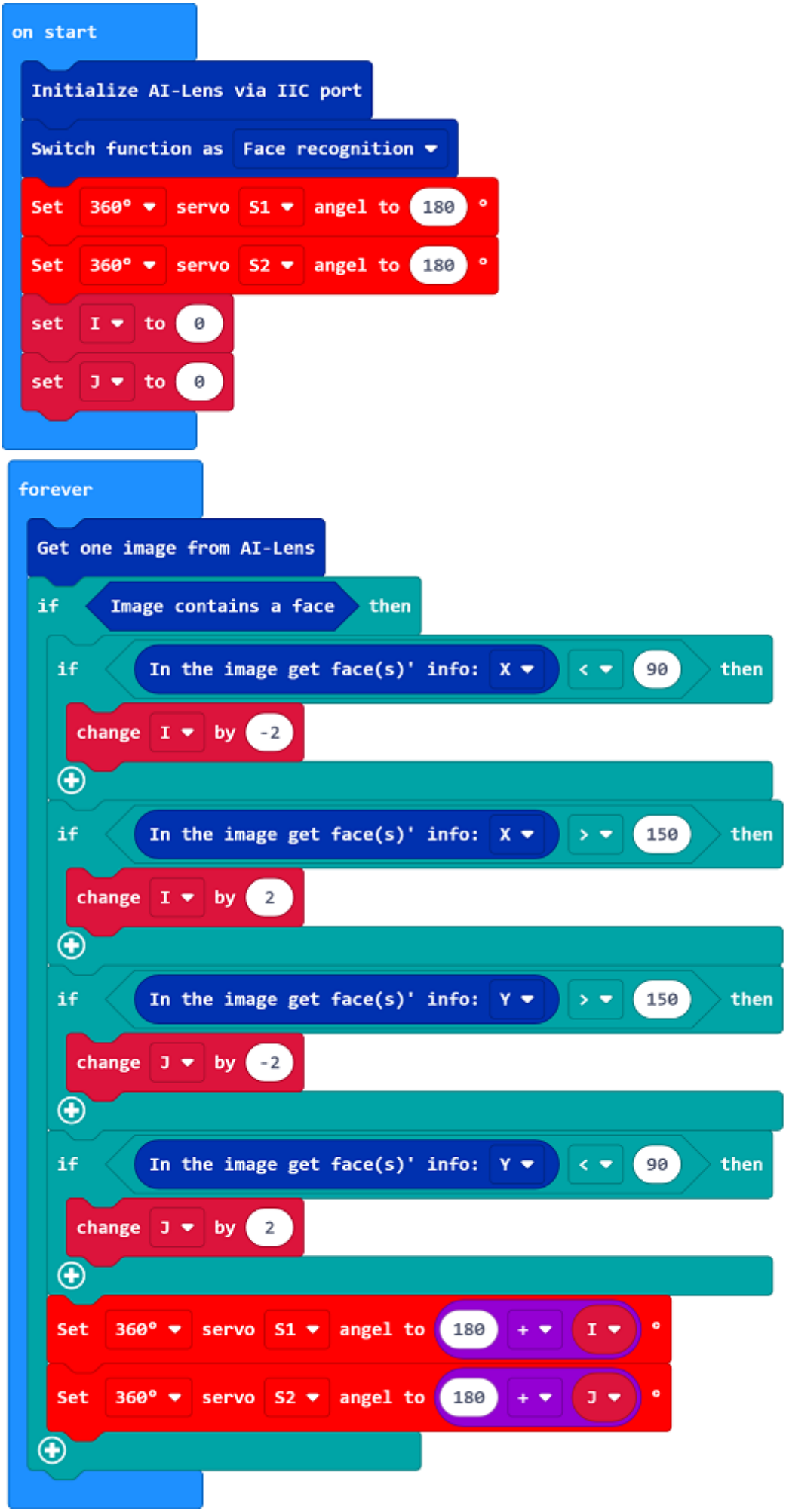
We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “PlanetX” in the dialogue box to download it.



Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.

Step 2

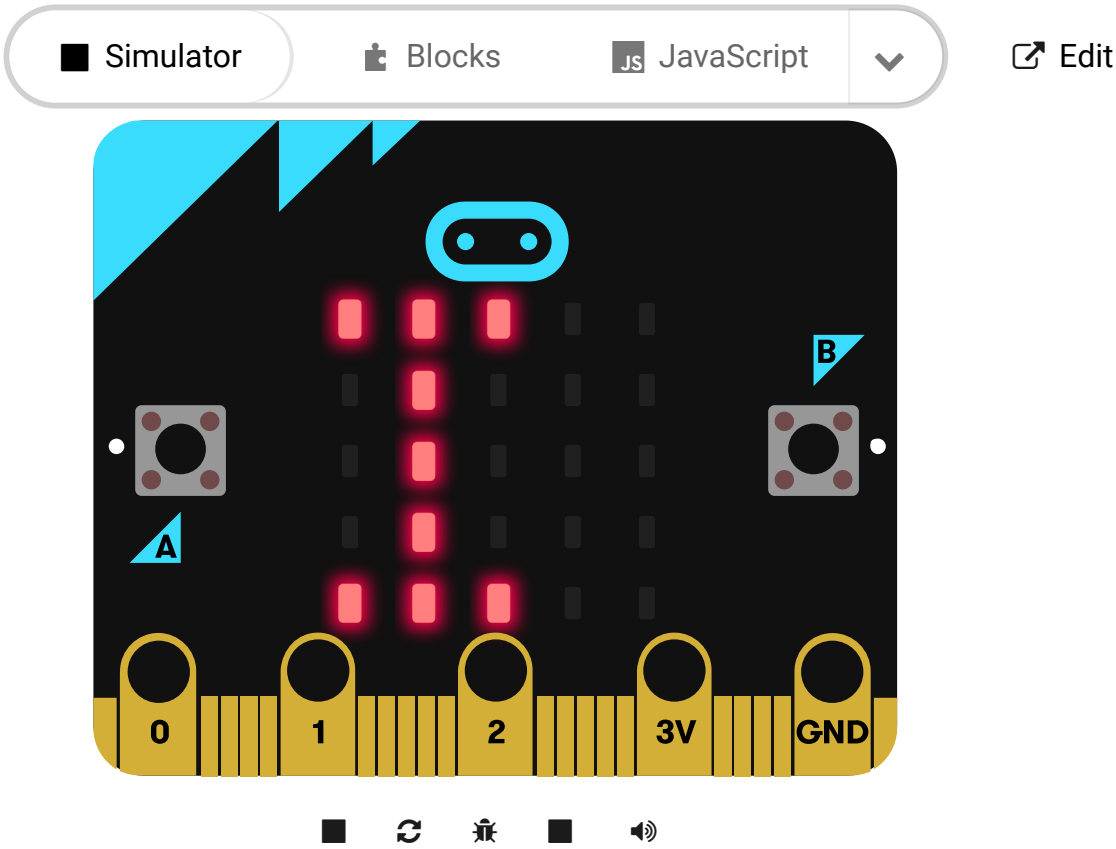
Code as below:


on start
Initialize AI-Lens via IIC port
Switch function as Face recognition
Set 360° servo S1 angel to 180 °
Set 360° servo S2 angel to 180 °
set I to 0
set J to 0
forever
Get one image from AI-Lens
if Image contains a face then
if In the image get face(s)' info: X < 90 then
change I by -2
if In the image get face(s)' info: X > 150 then
change I by 2
if In the image get face(s)' info: Y > 150 then
change J by -2
if In the image get face(s)' info: Y < 90 then
change J by 2
Set 360° servo S1 angel to 180 + I °
Set 360° servo S2 angel to 180 + J °

Link

Link: https://makecode.microbit.org/_WPUVLt8vp5bH

You may also download it directly below:



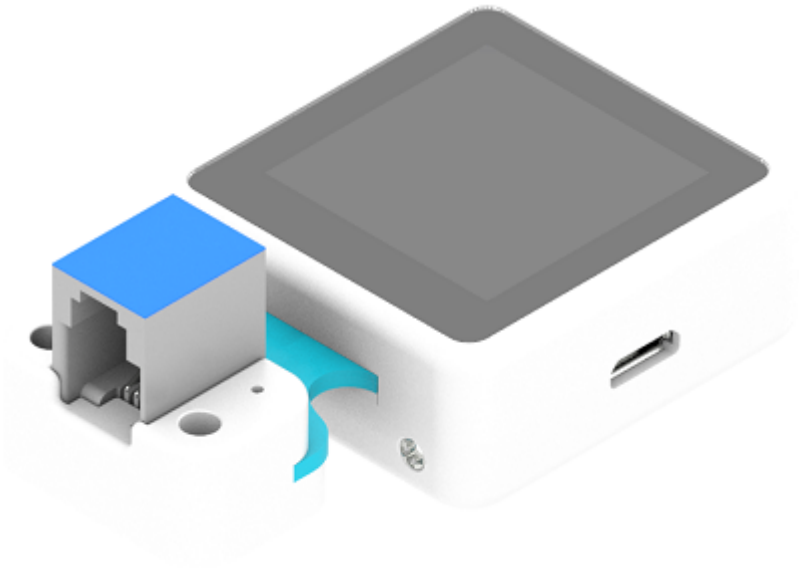
Result

- The AI Lens would track your face(move along with your face).

1.7. Case 05: Find the Bigger Face

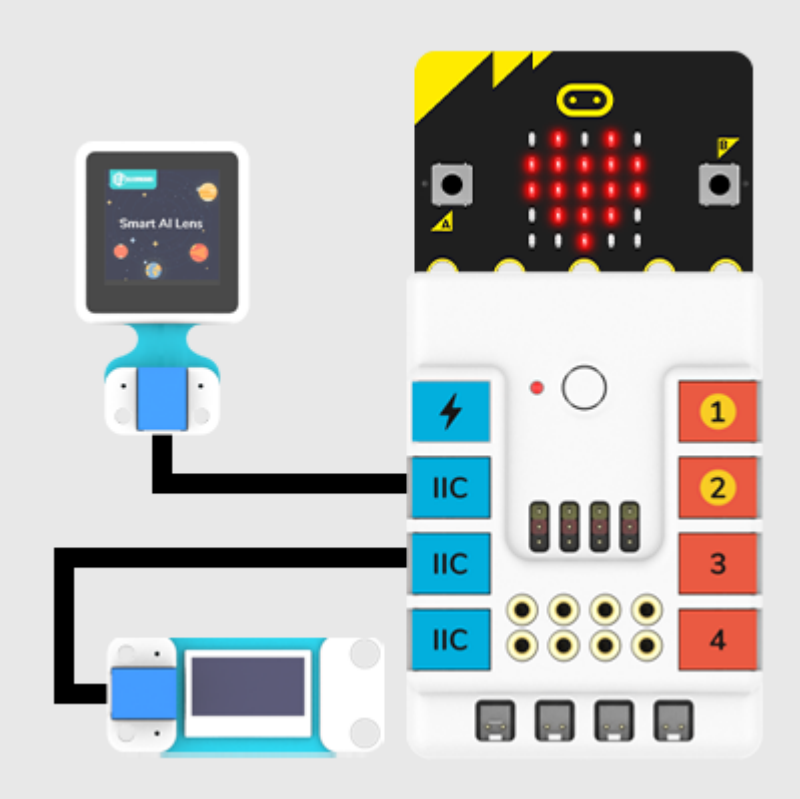
Purpose

To make the AI Lens be able to recognize the face and display the width of the face on the OLED module.



Materials Required and Connection Diagram

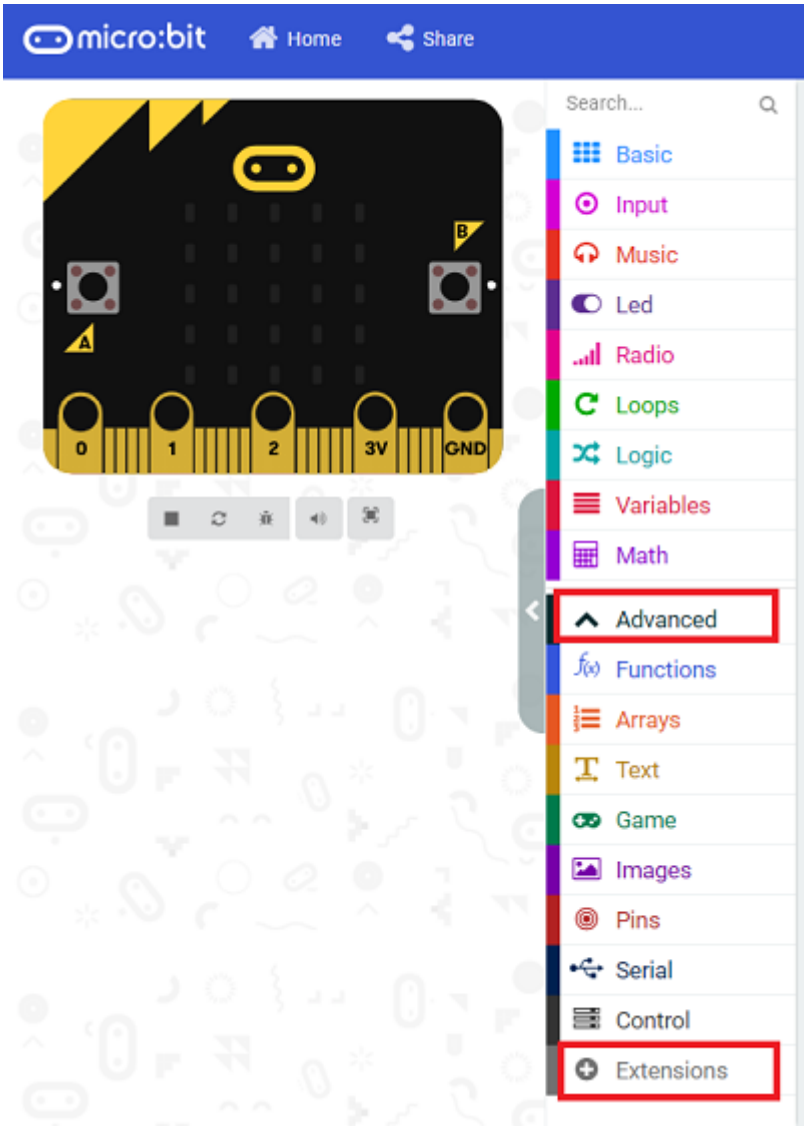
- Connect the AI Lens and the OLED display to the IIC ports on Nezha expansion board.



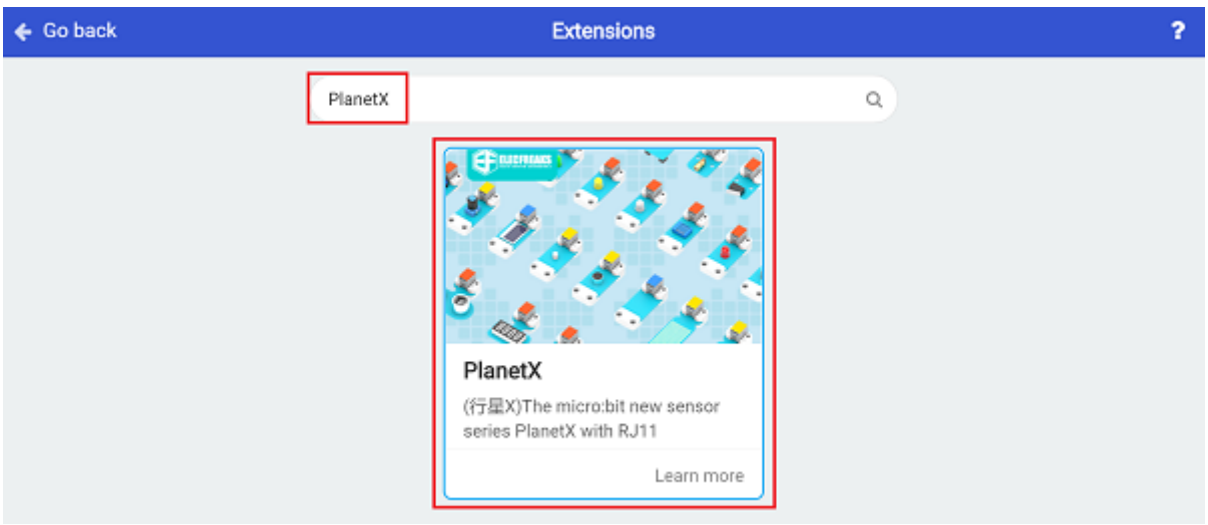
MakeCode Programming

Step 1

Click “Advanced” in the MakeCode drawer to see more choices.



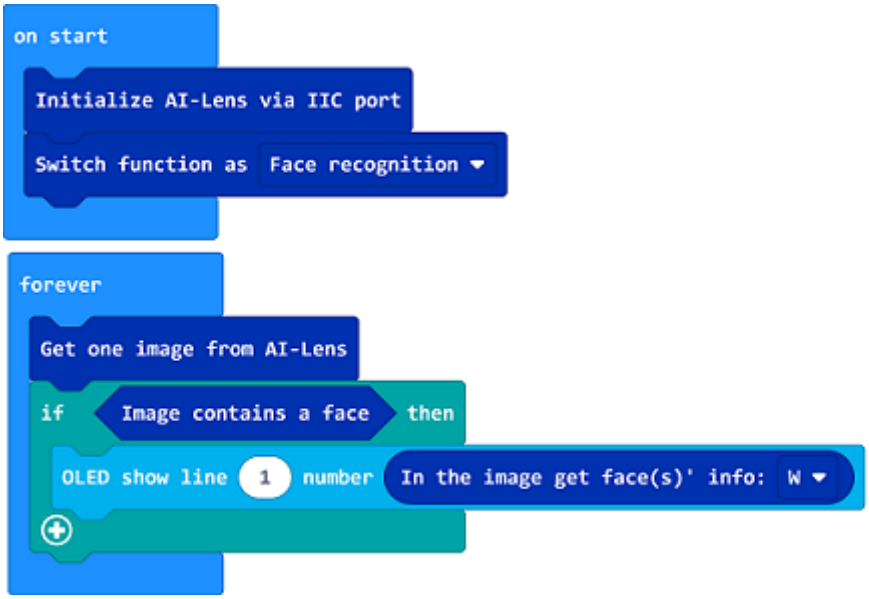
We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “PlanetX” in the dialogue box to download it.



Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.

Step 2

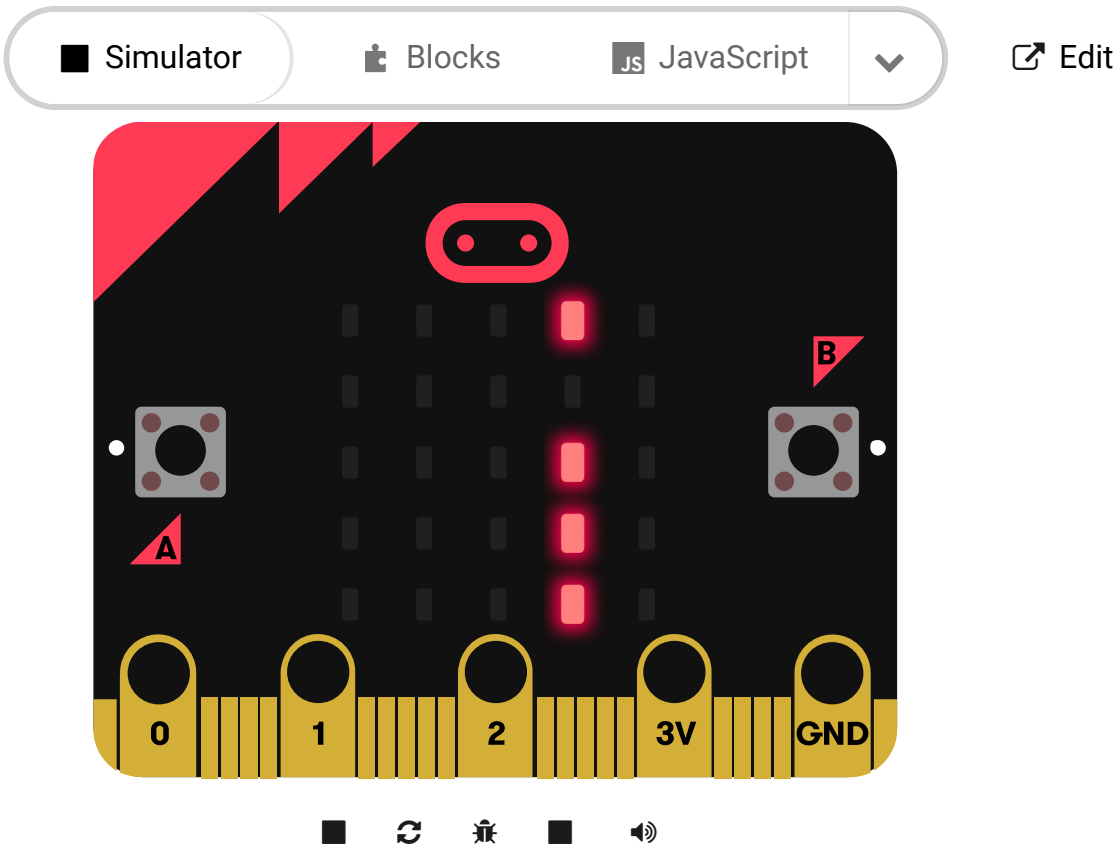
Code as below:



Link

Link: https://makecode.microbit.org/_gPrDDWhrAF7m

You may also download it directly below:



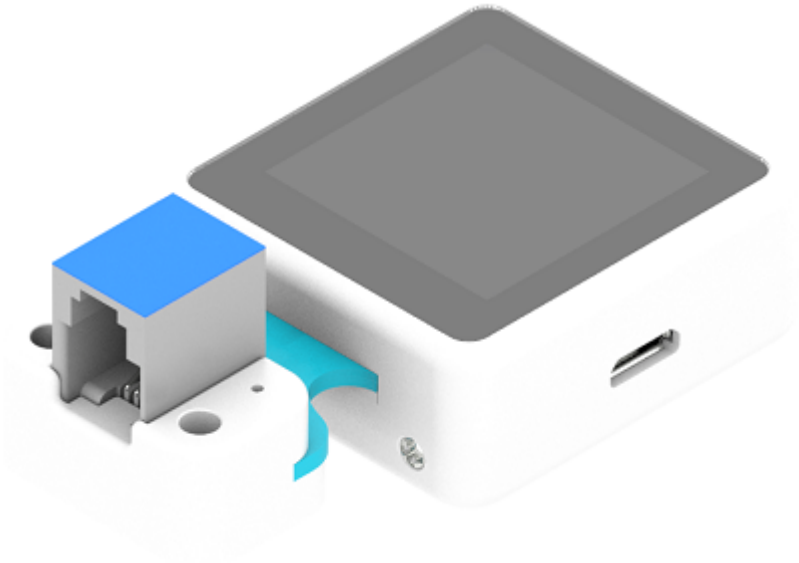
Result

- The width of the regcognized face displays on the OLED module.

1.8. Case 06: Characteristics Acquisition

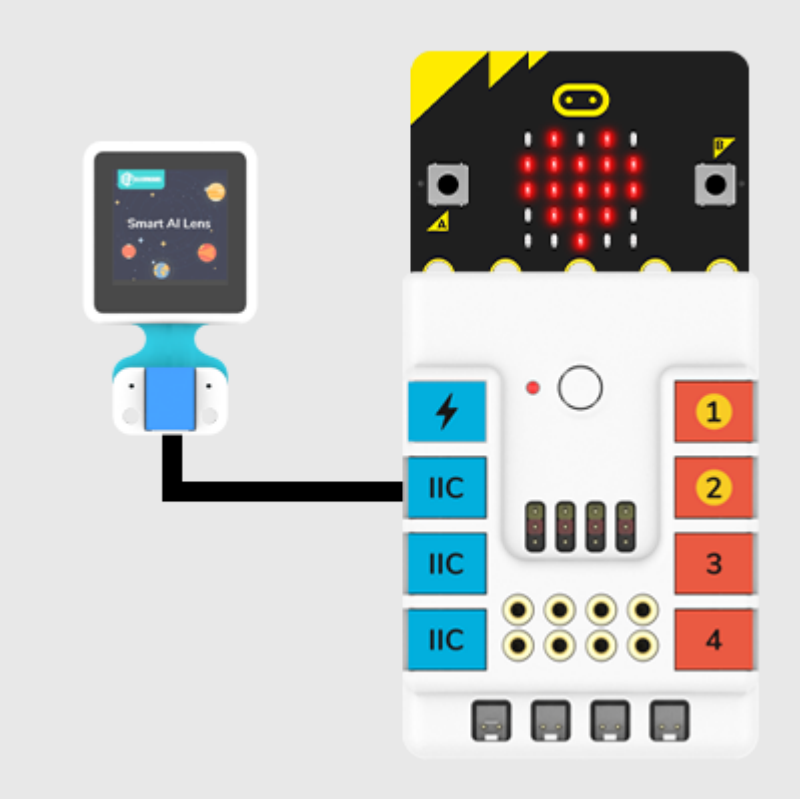
Purpose

To make the AI Lens be able to recognize and learn from the objects.



Materials Required and Connection Diagram

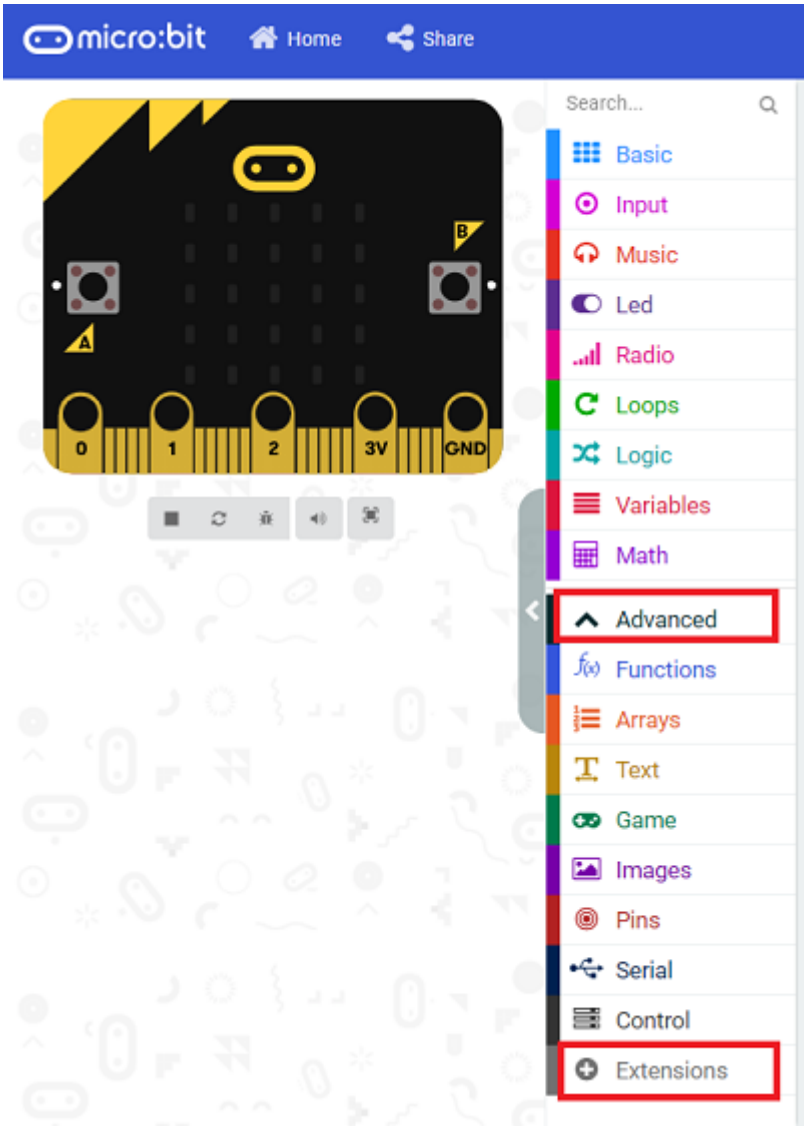
- Connect the AI Lens to the IIC port on the Nezha expansion board as the picture shows.



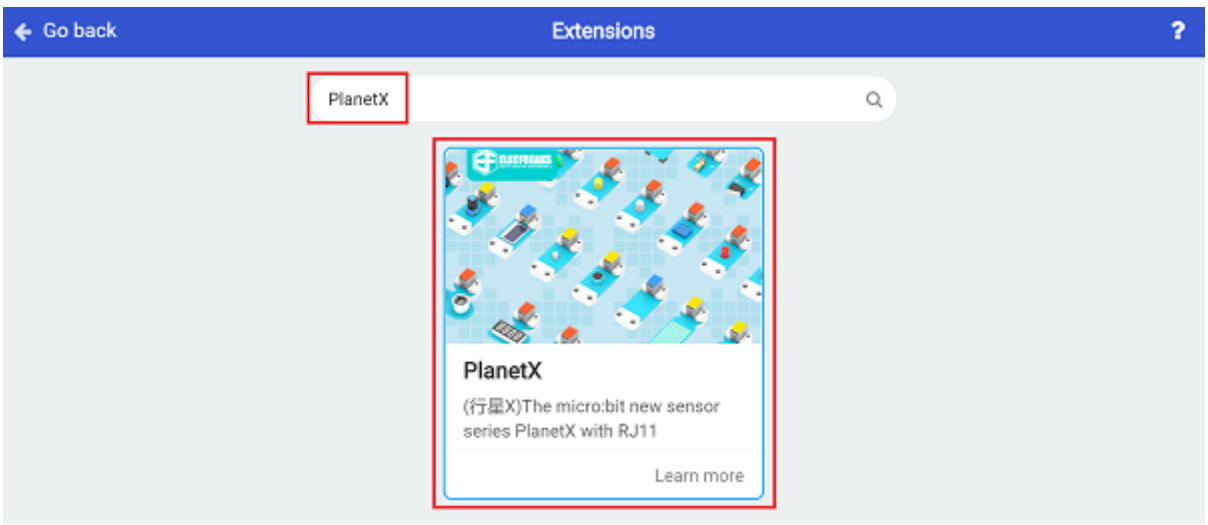
MakeCode Programming

Step 1

Click “Advanced” in the MakeCode drawer to see more choices.



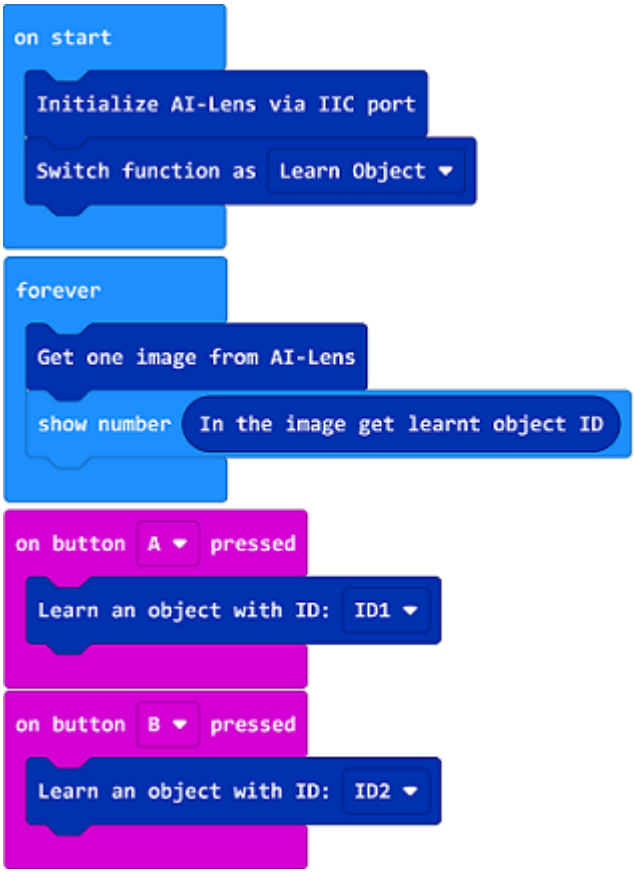
We need to add a package for programming. Click “Extensions” in the bottom of the drawer and search with “PlanetX” in the dialogue box to download it.



Note: If you met a tip indicating that the codebase will be deleted due to incompatibility, you may continue as the tips say or build a new project in the menu.

Step 2

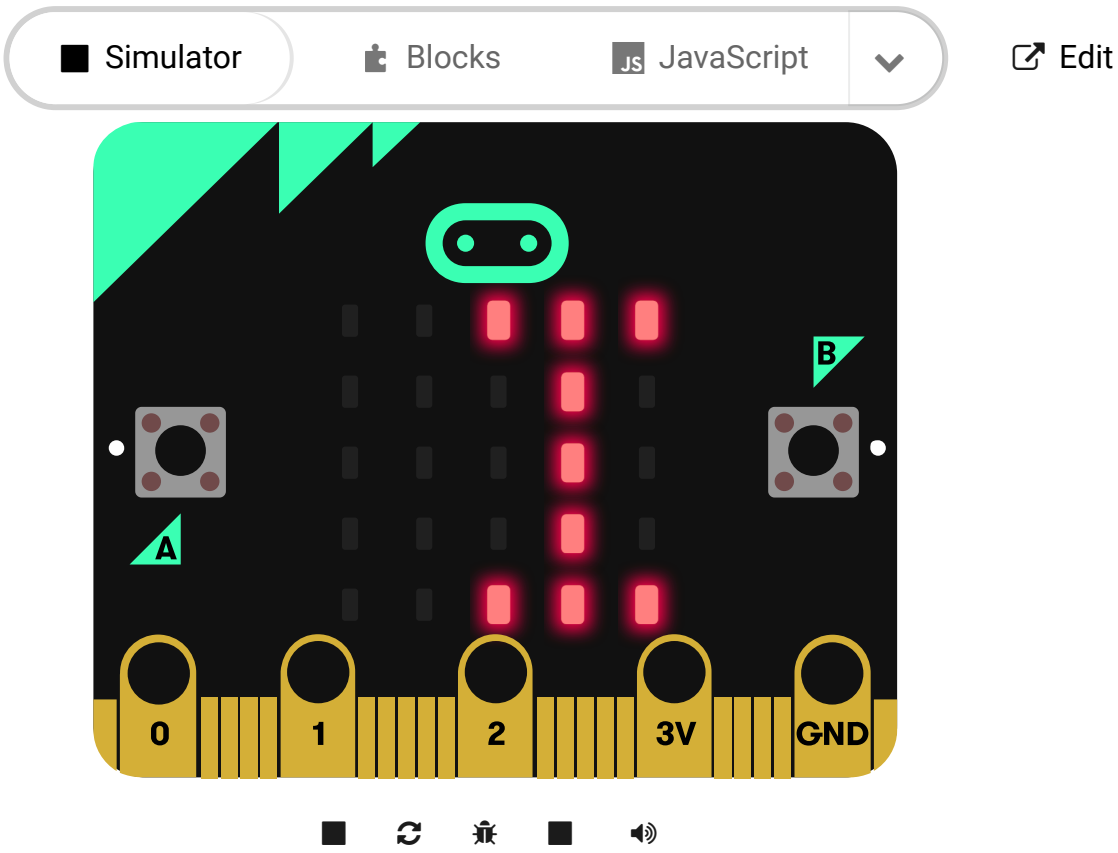
Code as below:



Link

Link: https://makecode.microbit.org/_46c9WCd5oc1z

You may also download it directly below:



Result

- Press button A to recognize the object and give it an ID 1; press button B to recognize the object and give it an ID 2. Then the equivalent ID number should display on the micro:bit in accordance with the object that you place in the front of the AI Lens.