Project 1: Numbered Musical Notation of Color

Is music related to color?

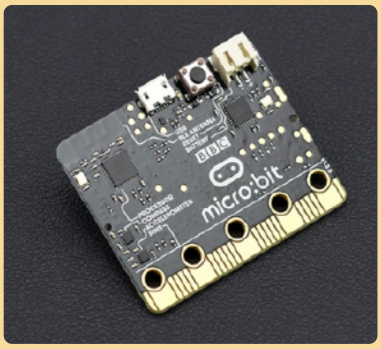
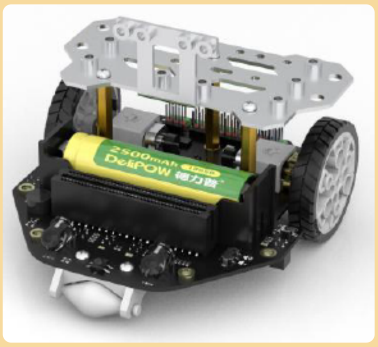
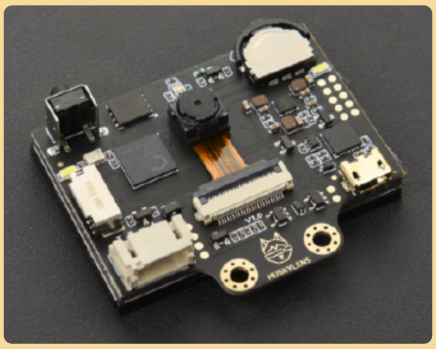
Many musicians compare music with color. The famous American musician Maryon once said: "Sound is audible color, color is visible music". The use of different timbres in music works is very similar to the use of different colors in art works. Both timbre and color can give people feelings like clear, bright, warm or dim.

Although there is no scientific theory to prove the correlation between music and color, we can make fun interactive devices between music and color. The simplest thing is to connect the seven tones in the scale with seven colors so that each color represents a scale. Let’s make a colored score, through which the wonderful music generated!

Function Description:

This project uses color recognition function of HuskyLens to recognize color blocks of different colors. By using Maqueen Plus to play different scales, your music score is not only good-looking but also pleasant to listen, which makes an absolutely wonderful audio-visual effect.

Materials Checklist:

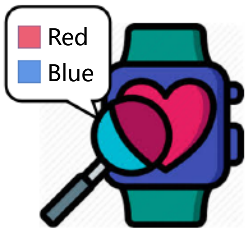
		
Micro:bit ×1	Maqueen Plus ×1	HUSKYLENS ×1

Knowledge Extension:

In today's society, automated production has become the development trend. As the eyes of "robots", machine vision is particularly significant.

As one of the most important technical directions, color recognition has undergone many generations of technological upgrading. This project is based on the color recognition function of HuskyLens sensor.

I.What is color identification?



What is color identification? First we need to know what color is.

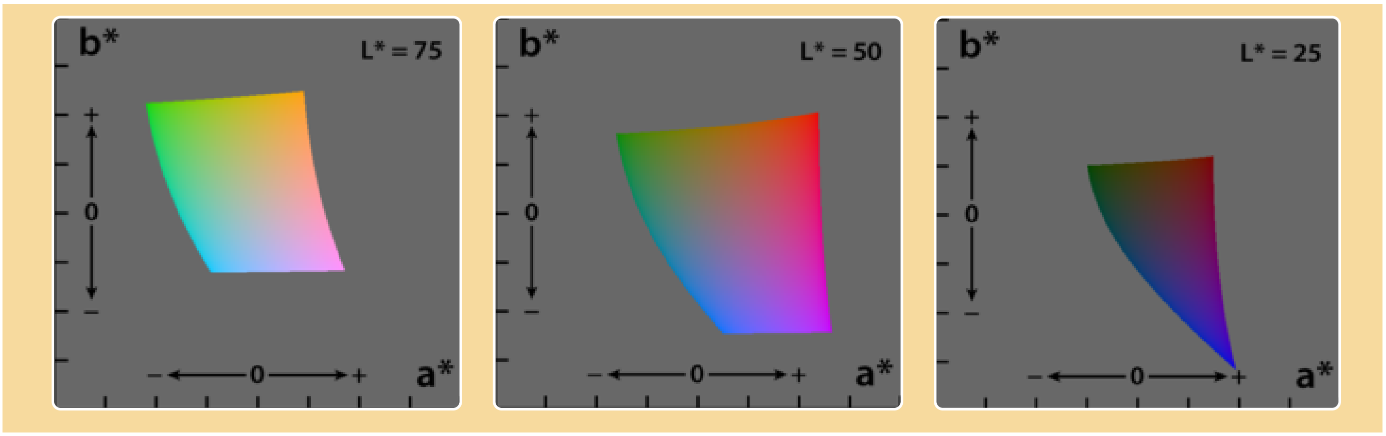
Color is a visual effect on light generated by eyes, brain and our life experience. Light we see with our naked eyes is generated by electromagnetic waves with very narrow wave-length range. Electromagnetic waves with different wavelengths show different colors.

Color recognition is based on color attributes under different lightness to identify and distinguish.

II. The Working Principle of Color Recognition

Color recognition is based on Lab color space, with dimension L representing lightness, a and b representing dimension of opposite color value, and CIE XYZ color space coordinates based on nonlinear compression.

Comparing the Lab parameters of the recognized and learned colors, when the two colors match within a certain error range, they are identified as the same color.

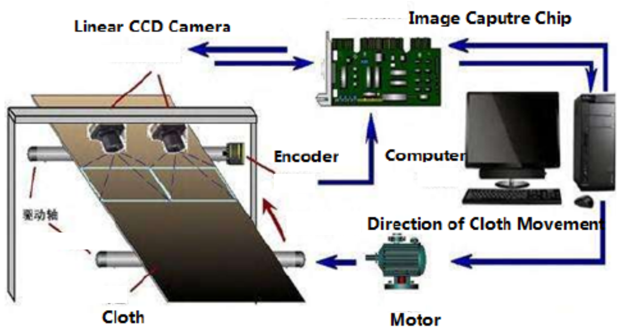


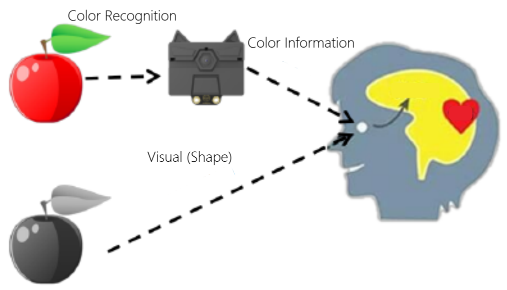
In our usual use of color recognition, the hue and saturation in the color attribute of the same module are fixed, but the lightness will change under the influence of the environment light change. Therefore, when using color recognition function of HuskyLens, we must try our best to ensure that the environment light during learning and recognition is consistent with actual work.

III. Main Application Fields of Color Recognition

Industrial field: Color recognition is currently widely used in the industrial field, such as printing, paint coating and textile, for color monitoring and calibration.

Personal life: usage as an auxiliary recognition for people with color weakness or visual impairment can enhance their understanding of color.



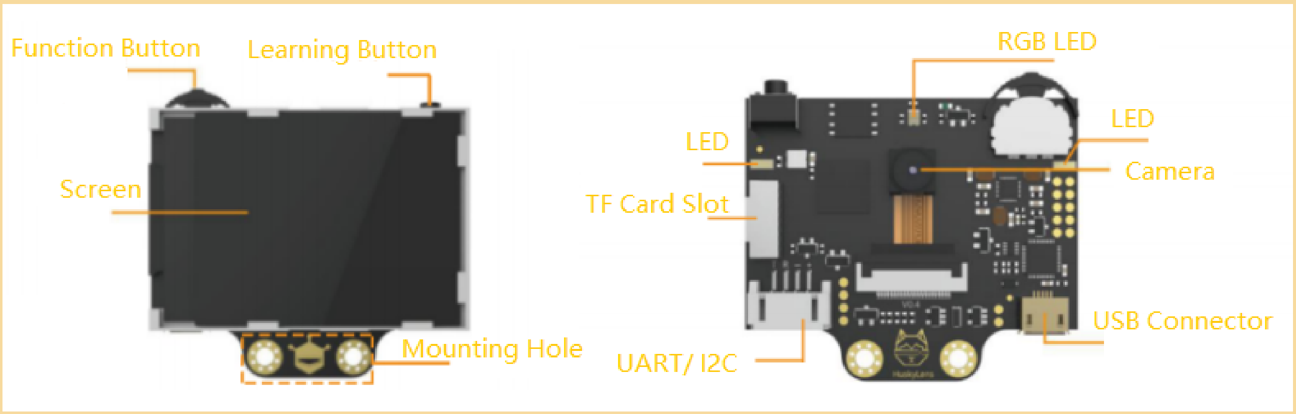


IV. Demonstration of HuskyLens Color Recognition Function

The color recognition function of HuskyLens sensor is to use the built-in algorithm, by learning and recording different colors, the ID of different colors can be identified and fed back to the mainboard.

The default setting in HuskyLens is to learn, recognize and track only one color. But we can change the default setting to make it recognize multiple colors.

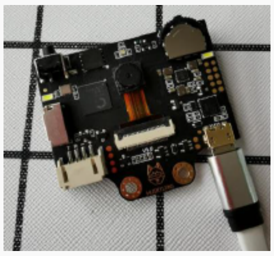
How to operate it step by step? First, take out your HuskyLens. Let's do it together.



For first-time users, please refer to WIKI website for firmware burning and language setting: http://wiki2.dfrobot.com.cn/HuskyLens_V1.0_SKU_SEN0305_SEN0336

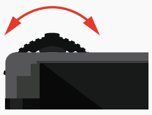
STEP1 Switch On

HuskyLens has its own independent USB connector, which can be switched on by connecting the USB cable.

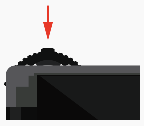


STEP2 Operation and Setting - Learn Multiple

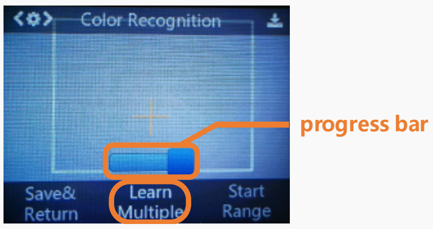
① Dial the function button to the right or left until the word "Color Recognition" is displayed at the top of the screen.



② Long press the function button to enter the parameter setting of the color recognition function.



③ Dial the function button until "Learn Multiple" is displayed, then short press the function button, and dial to the right to turn on the "Learn Multiple" switch, that is, progress bar turns blue and the square icon on the progress bar moves to the right. Then short press the function button to confirm this parameter.



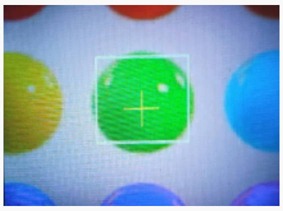
④ Dial the function button to the left until "Save & Return" shows. And the screen prompts "Do you want to save the parameters?" Select "Yes" in default, now short-press the function button to save the parameters and return automatically.

In this way, we finish the set-up of "Learn Multiple" function.

STEP3 Operation and Setting - Learning and Detection

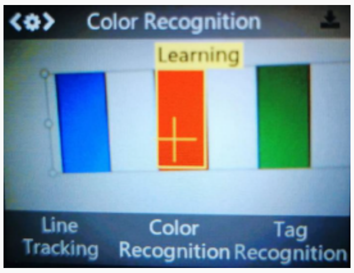
(1) Color Detection

Point the icon "+" in the center of the HuskyLens screen to the target color block, and a white box will appear on the screen, which selects the target color block automatically. Adjust the angle and distance of the HuskyLens to the color block so that the white box frames the entire target color block as far as possible.



(2) Color Learning

After detecting the color, long press the "learning button" to learn the first color, and then release the "learn button" to finish learning. A message will prompt on the screen: "Click again to continue! Click other button to finish".



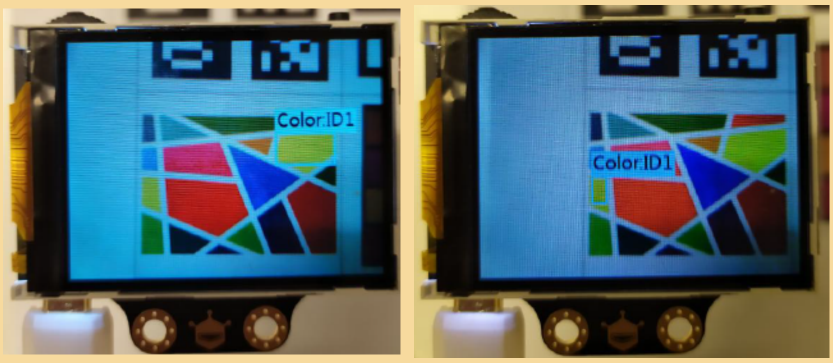
Please short press the "learning button" before the countdown ends if you want to learn next color. If not, short press the "function button" before the countdown ends, or do not press any button to let the countdown ends.

The color ID displayed by HuskyLens is in the same order as the learning color, that is, the ID will be marked as "ID1", "ID2", "ID3" and so on. And the frame colors corresponding to different colors are also different.

(3) Color Identification

When encountering the same or similar color blocks, a colored frame with an ID will be automatically displayed on the screen, and the size of the colored frame is same as the size of the color blocks. The frame will automatically track the color block. Different colors can be identified and tracked at the same time, and the frame colors corresponding to different colors are also different.

When multiple color blocks in the same color appear in the picture, only one color block can be identified at a time.



* Color recognition is greatly affected by environment light. Sometimes HuskyLens may misidentify similar colors. Please try to keep the environment light consistent and use under moderate lighting.

Project practice:

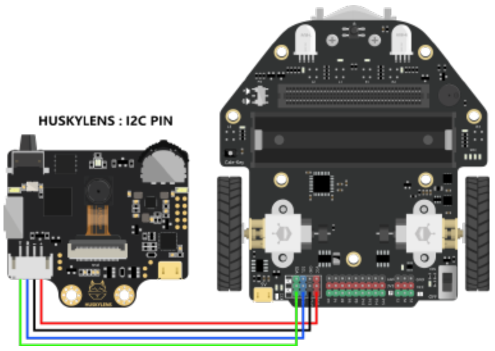
How is HuskyLens color recognition function used? How can we map colors to scales one by one? Let's break down the whole project into several small tasks and complete the color numbered musical notation step by step.

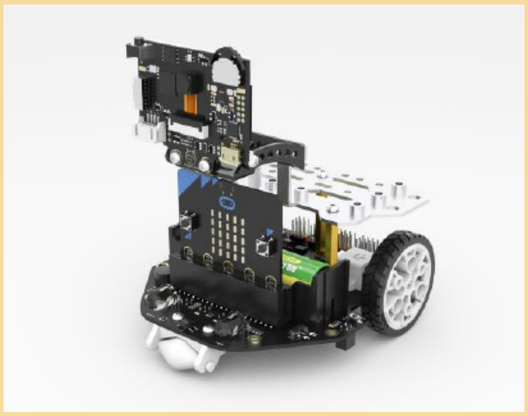
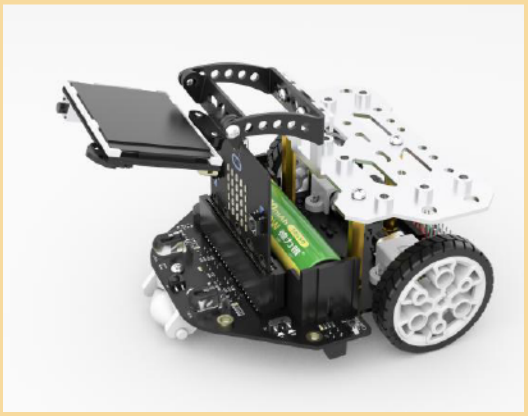
This project will be divided into three steps to complete the task. First, we will learn to use color recognition function of HuskyLens and output the recognized color ID. Then we can play the scales according to the output color ID.

Task 1: Get to Know Color Recognition

1. Hardware Settings

Connection Diagram: HuskyLens sensor uses IIC connector. Please pay attention to the cable sequence. Do not connect it wrongly or reversely.





Assembly Diagram: HuskyLens sensor has its own bracket structure, which can be fixed to Maqueen Plus by screws, and HuskyLens can adjust various angles.

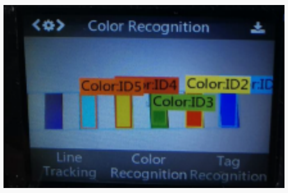
2. Program Design

Here we take HuskyLens sensor to learn 3 colors as an example, and output color ID through serial port to facilitate real-time viewing.

* Serial port: Serial port is a way of real-time communication between computer and hardware. For example, in this task, through the serial port, the data of HuskyLens on the Maqueen Plus can be viewed in real time on the computer.

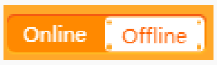
STEP 1 Learning and Recognition

Before designing the program, we need HuskyLens sensor to learn each color. (Note: First switch to "Learn Multiple" function)

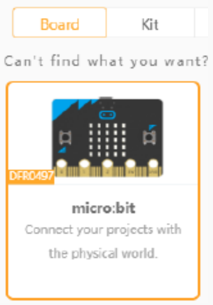


STEP 2 Mind+ Software Settings

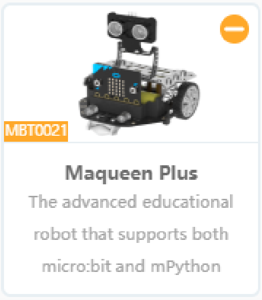
Open Mind+ Software (Version 163 or above):



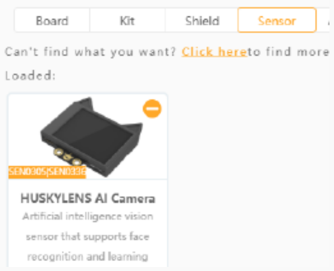
① Switch to "Upload mode";


Board
Kit
Can't find what you want?
DFR0497
micro:bit
Connect your projects with the physical world.

② Click "Extensions" and load "micro:bit" under "Board";


MBT0021
Maqueen Plus
The advanced educational robot that supports both micro:bit and mPython

③ Continue to click and load "Maqueen Plus Domestic Edition" under "Expansion";


Board
Kit
Shield
Sensor
Can't find what you want? Click hereto find more
Loaded:
SEN0305|SEN0336
HUSKYLENS AI Camera
Artificial intelligence vision sensor that supports face recognition and learning

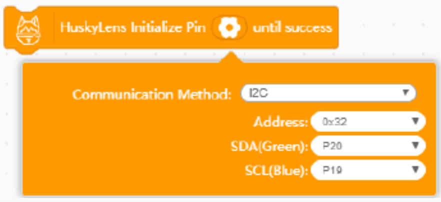
④ Click and load "HuskyLens AI Camera" under "Sensor" to finish.

STEP3 Instruction Learning

Let's take a look at some of the main instructions.

①Initializing: placed between the start of the main program and loop execution, only need to be executed once. I2C or serial port can be selected, I2C address does not need to be changed.

Note: HuskyLens side needs to adjust the "output protocol" in the setting to be consistent with the program, otherwise the data cannot be red.

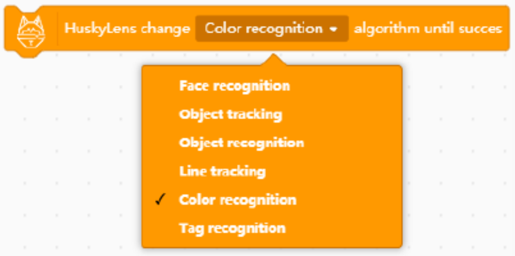

HuskyLens Initialize Pin ⚙ until success

Communication Method: I2C
Address: 0x32
SDA(Green): P20
SCL(Blue): P19

STEP3 Instruction Learning

②Switching algorithms: other algorithms can be switched to at any time, and only one algorithm can exist at the same time.

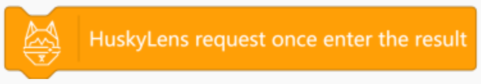
Note: Algorithm switch takes some time.



HuskyLens change [Color recognition ▾] algorithm until succes

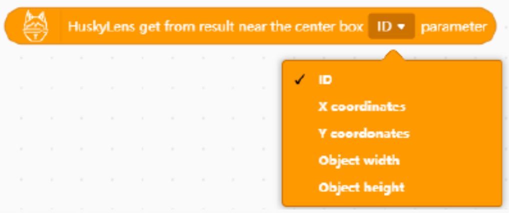
Face recognition
Object tracking
Object recognition
Line tracking
✓ Color recognition
Tag recognition

③The main board requests HuskyLens to store the data in the "result" once (data stored in the memory of the main board, refreshed by each request), then the data can be obtained from the "result", and the latest data will be obtained in the "result" only after this module is called.



HuskyLens request once enter the result

④The information near the center box in the current interface is obtained from the requested "result". The box id that has not been learned is 0, otherwise, -1 is returned.

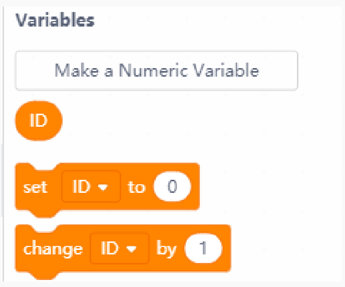


HuskyLens get from result near the center box [ID ▾] parameter

✓ ID
X coordinates
Y coordonates
Object width
Object height

⑤Variable: Variable refers to the amount of change, which is convenient to store the changed number.

Click "Make a Numeric Variable" to create a new variable. You can directly set the value of the variable as well as increase or decrease the value of the variable.



Variables

Make a Numeric Variable

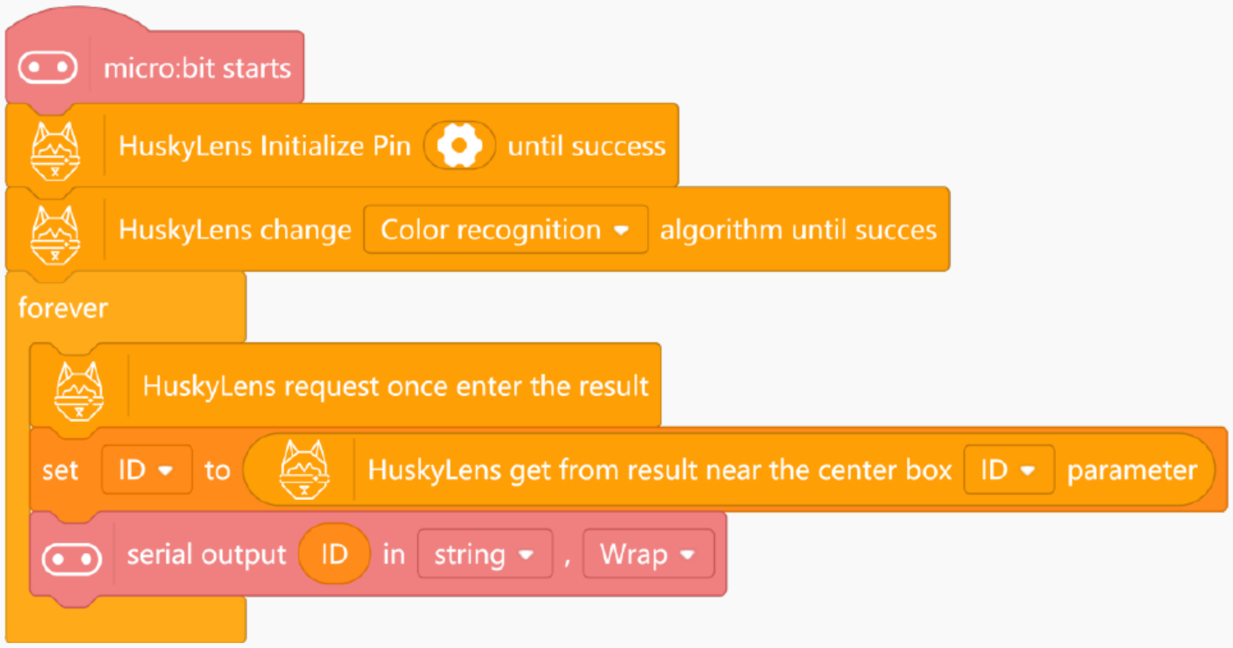
ID

set [ID ▾] to (0)

change [ID ▾] by (1)

3 Program Example

The program

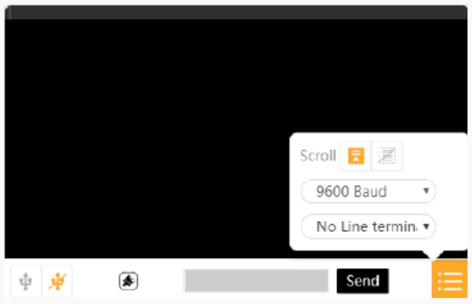


```
micro:bit starts
HuskyLens Initialize Pin [I2C] until success
HuskyLens change [Color recognition ▾] algorithm until succes
forever
    HuskyLens request once enter the result
    set [ID ▾] to (HuskyLens get from result near the center box [ID ▾] parameter)
    serial output (ID) in [string ▾] , [Wrap ▾]
```

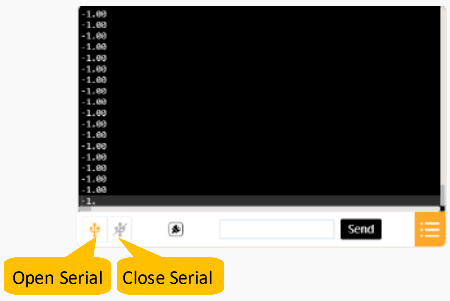
4 Execution Result

When using a serial port, the computer and the main board need to be connected through a USB cable. In Mind+, connect the corresponding COM port.

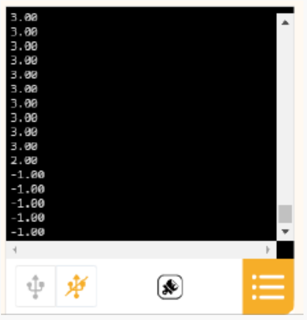
Serial port Operation Method:



1. Serial baud rate defaults to 9600



2. Open the serial port



Run window:

```
3.00
3.00
3.00
3.00
3.00
3.00
3.00
3.00
3.00
3.00
2.00
-1.00
-1.00
-1.00
-1.00
-1.00
```

When the HuskyLens sensor has learned three colors, it recognizes the ID of the corresponding color and displays the corresponding number on the serial port.1 denotes the first-learned color, 2 denotes the second -learned color, 3 denotes the third-learned color, and -1 denotes the color that has not been learned.

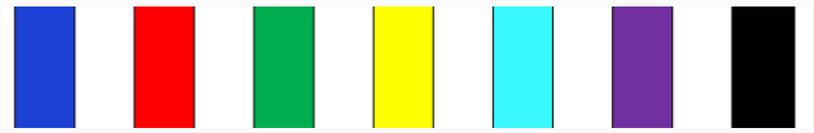
Task 2: Define Scale for Each Color

1. Program Design

STEP1 Operation and Setting-Learning and Recognition

Before designing the program, we need have HuskyLens to learn 7 colors. In order to avoid false recognition, 7 colors with big color difference should be selected.

The colors of ID1-ID7 are respectively set to blue, red, green, yellow, cyan, purple and black. Please learn colors in this order to correspond to the following procedures. After learning the 7 colors, HuskyLens stopped learning.



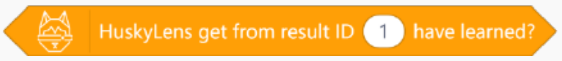
In combination with the lights on Maqueen Plus, identify a color and let the lights display the corresponding color.

See? Is the visual effect more glorious?

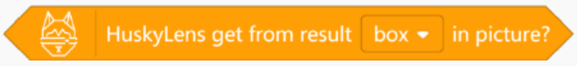
STEP2 Instruction Learning

Let's take a look at some of the main instructions.

①Obtaining whether IDx has learned from the requested "result".

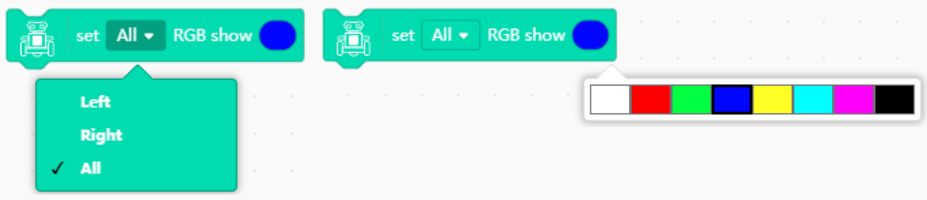


②Obtaining whether there is a box or arrow in the current interface from the requested "result", including learned (id is greater than 0) and not learned, if there is one or more, return 1.

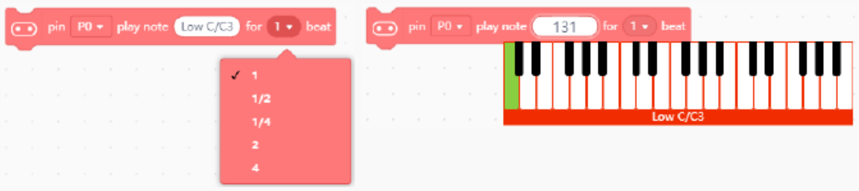


STEP2 Instruction Learning

③Set the car lights to display different colors, and the program provides 8 colors.



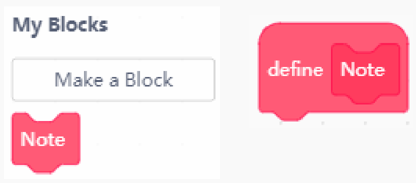
④Instructions of playing notes: are divided into low, medium and high notes, and various beats.



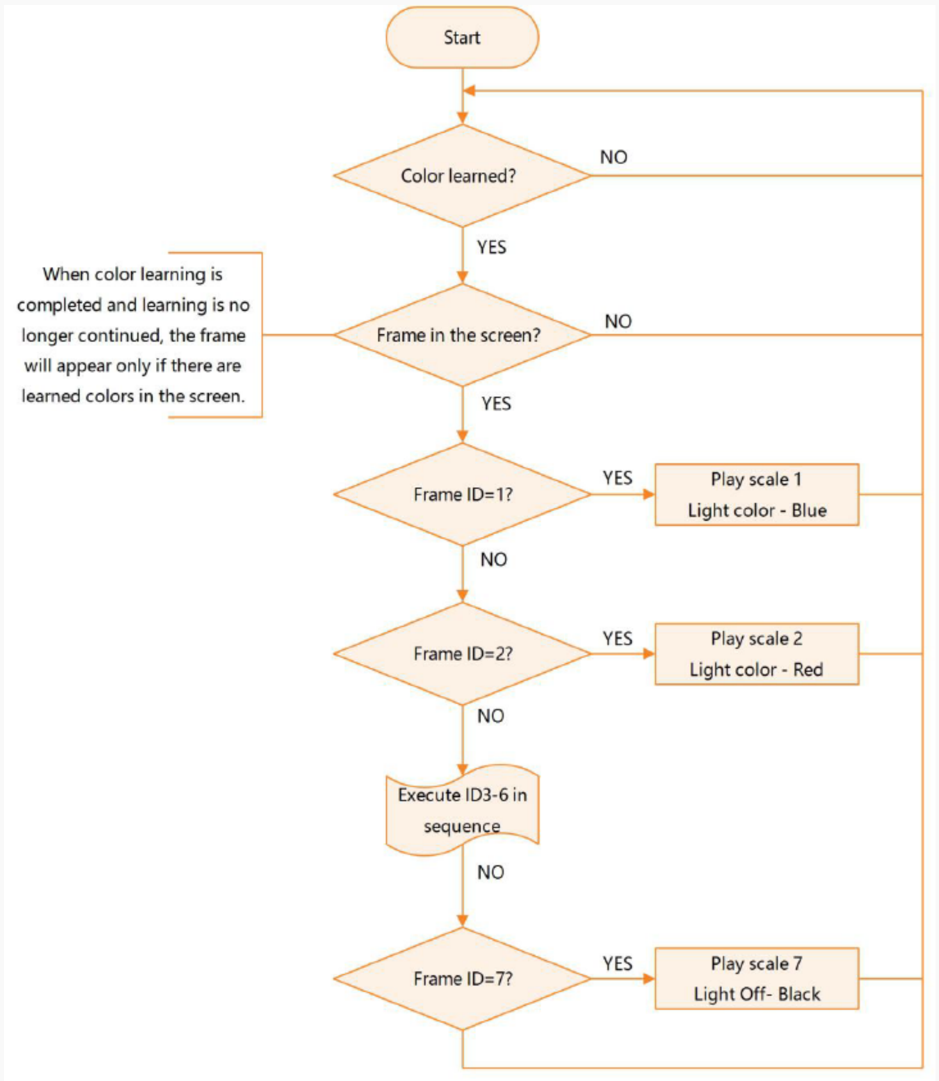
⑤Function: A function can be understood as a program that performs a specified function.

Click "Make a Block" to create a new function.

Sometimes too many instructions in the main program will affect our understanding. Functions can simplify the main program and facilitate understanding of the whole program.

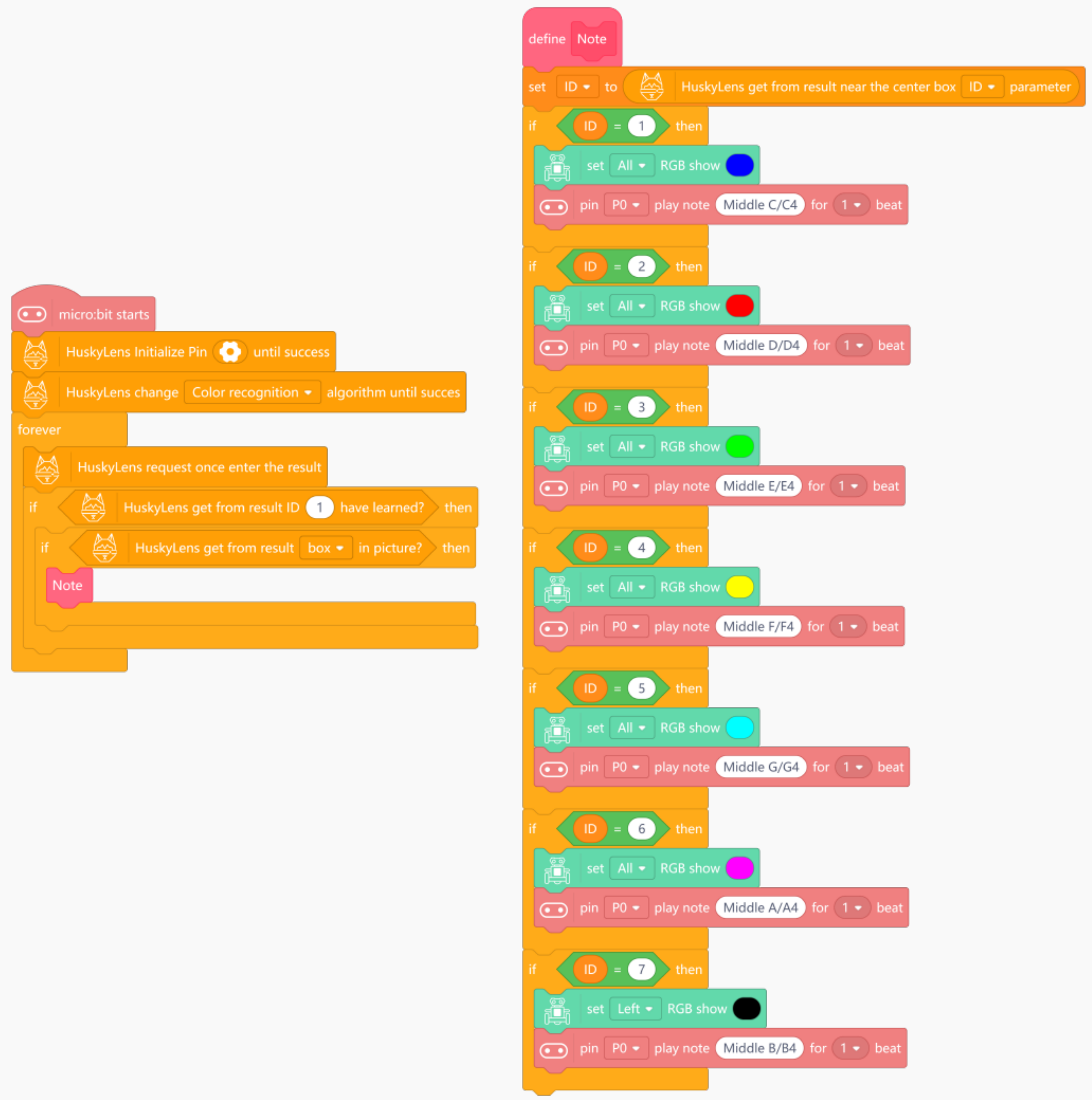


STEP 3 Flow Chart Analysis



2.Program Examples

The program

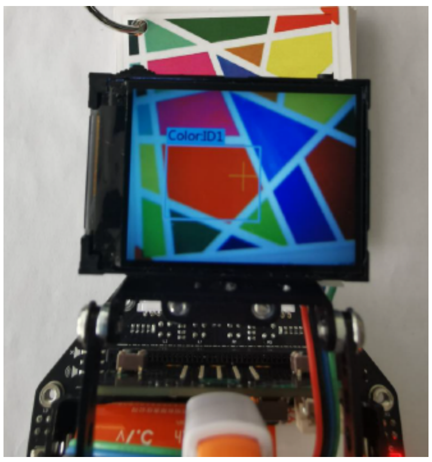


```
micro:bit starts
  HuskyLens Initialize Pin ● until success
  HuskyLens change Color recognition ▾ algorithm until succes
forever
  HuskyLens request once enter the result
  if HuskyLens get from result ID (1) have learned? then
    if HuskyLens get from result box ▾ in picture? then
      Note

define Note
  set ID ▾ to HuskyLens get from result near the center box ID ▾ parameter
  if ID = 1 then
    set All ▾ RGB show ●
    pin P0 ▾ play note Middle C/C4 for 1 ▾ beat
  if ID = 2 then
    set All ▾ RGB show ●
    pin P0 ▾ play note Middle D/D4 for 1 ▾ beat
  if ID = 3 then
    set All ▾ RGB show ●
    pin P0 ▾ play note Middle E/E4 for 1 ▾ beat
  if ID = 4 then
    set All ▾ RGB show ●
    pin P0 ▾ play note Middle F/F4 for 1 ▾ beat
  if ID = 5 then
    set All ▾ RGB show ●
    pin P0 ▾ play note Middle G/G4 for 1 ▾ beat
  if ID = 6 then
    set All ▾ RGB show ●
    pin P0 ▾ play note Middle A/A4 for 1 ▾ beat
  if ID = 7 then
    set Left ▾ RGB show ●
    pin P0 ▾ play note Middle B/B4 for 1 ▾ beat
```

3. Execution Result

We have learned colors in the order of blue, red, green, yellow, cyan, purple and black given above. Then execute the program, when the center of HuskyLens display screen is blue, play note 1 and the light is blue; when it is red, play note 2, and the light turns red, and so forth.



If your programs do not work properly, please check the following points:

(1) Whether they have learned colors in the order of blue, red, green, yellow, cyan and purple;

(2) Whether the mode is switched to "Stop Learning" after learning the above 7 colors; if not, a white box will always be output on the camera screen, which will affect the program execution.

Task 3: Remove Misjudged Interference of Small Color Blocks

1. Program Design

STEP1 Analysis of Problems Encountered

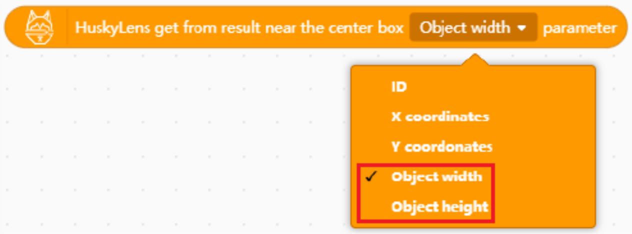
The program in Task 2 can basically realize the project function, but some small color blocks may become interference when there are multiple color blocks unexpectedly appeared in the camera screen. How to eliminate such interference?

One solution is to judge whether the color block is the target color block by the area proportion on the camera screen. We may set a threshold value, and the color block below the threshold value will automatically be judged as interference.

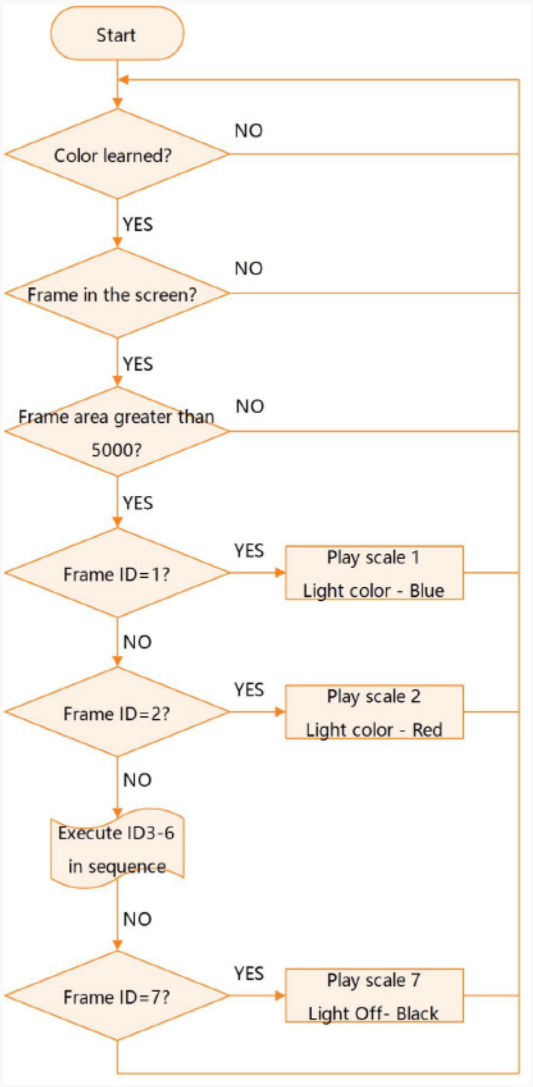
STEP 2 Instruction Learning

Let's take a look at some of the main instructions.。

①Obtain the information in the frame near the center box from the requested "result" in the current interface, the frame area can be obtained by multiplying "Object width" and "Object height".



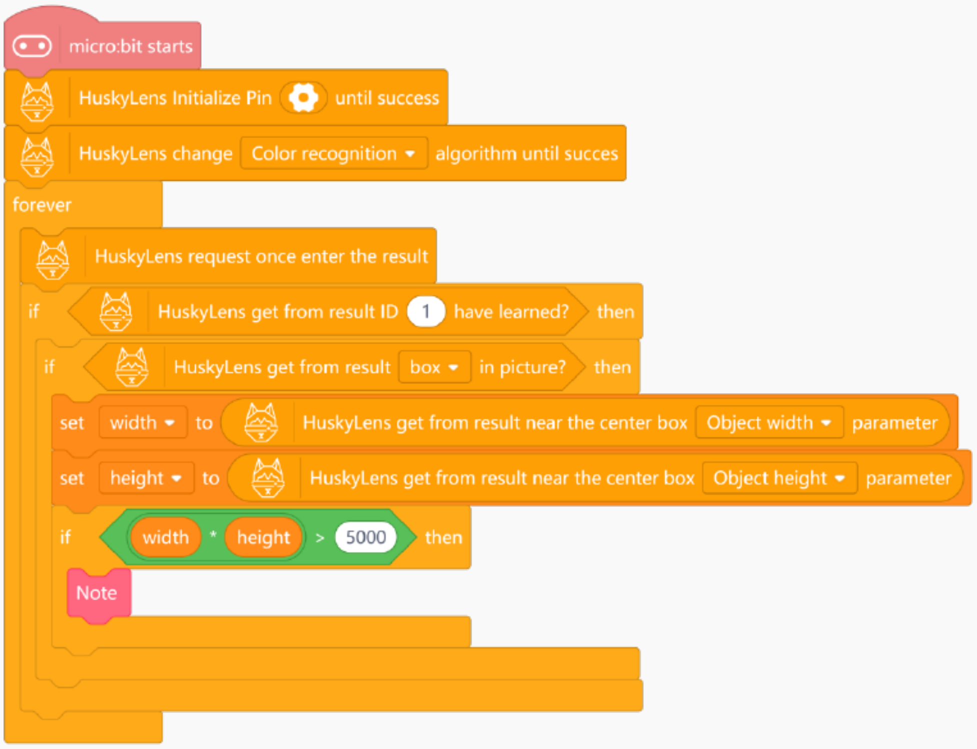
STEP3 Flow Chart Analysis



2. Program Examples

The program

Modify the program in Task 2. The function "note" remains unchanged. The main program is modified as follows.



```
micro:bit starts
HuskyLens Initialize Pin until success
HuskyLens change [Color recognition] algorithm until succes
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID (1) have learned? then
        if HuskyLens get from result [box] in picture? then
            set [width] to HuskyLens get from result near the center box [Object width] parameter
            set [height] to HuskyLens get from result near the center box [Object height] parameter
            if width * height > 5000 then
                Note
```

3. Execution Result

When the center of HuskyLens displays blue, play note 1 and the light is blue; when turns red, play note 2, the light turns red, and so forth.

Project Summary:

Project Review

Understand the working principle of color recognition in this lesson, and learn the color recognition function by using HuskyLens.

Color recognition is a very important function in AI vision recognition and is widely used in industrial fields. What other interesting functions can color recognition achieve? Please think about it.

Knowledge Nodes Recap

1. Understand the working principle of color recognition;
2. Learn color recognition function of HuskyLens and the operation method of recognizing multiple colors.

Project Extension:

After this project is completed, you can search for a numbered musical notation through internet, spell out the numbered musical notation with color blocks to have Maqueen Plus play music after printing it out!

However, we will definitely find a problem: the number of scales is relatively small, and the medium, low and high notes cannot be realized. If we want to add color blocks, with the increase number of colors, there will be many similar colors, which may lead to misrecognition. So is there any solution to widen the sound range?

(Tips: The AB button on micro:bit board can be used to realize the functions of raising and lowering the key.)

Project 2: Easy ETC (Electronic Toll Collection) System

When we drive on the expressway, we would find that it's nothing special at ordinary times, but if it comes to holidays, the toll stations are always jammed with vehicles!

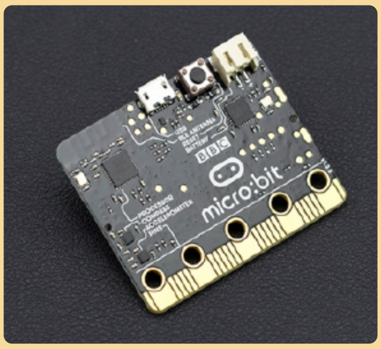
With the promotion of ETC, congestion at toll stations has also reduced. The so-called ETC is a non-stop electronic toll collection system. It adopts a way of collecting user fee electronically, when cars owners with ETC drive through toll stations in a low speed, tolls are charged automatically.

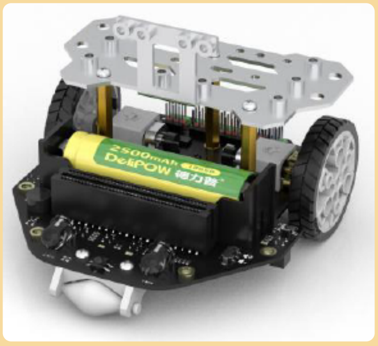
Can Maqueen Plus realize ETC function? In this project, let's make a simple and easy ETC system together, so that when the Maqueen Plus is on the "expressway", it can also achieve non-stop charge!

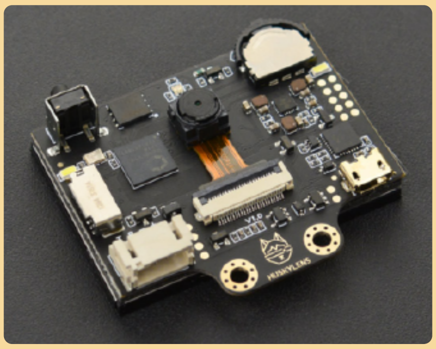
Function Description:

Tag recognition function of HuskyLens is applied in this project. Two tags stand for the entrance and exit of toll stations respectively. By recording the interval time, mileage and toll are calculated, thus realizing an easy ETC function of Maqueen Plus.

Materials Checklist:


Micro:bit ×1

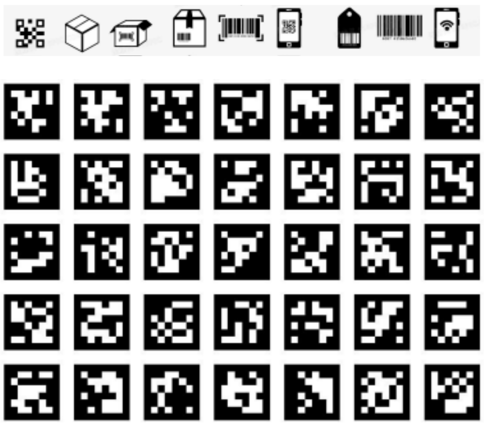

Maqueen Plus ×1


HUSKYLENS ×1

Knowledge Extension:

Tag recognition is a branch of machine vision and is widely used in production and life.

1. What is Tag Recognition?



Tag recognition technology refers to a technical method of effective and standardized encoding and identification for goods. In life, there are many kinds of tags, such as barcode and QR code.

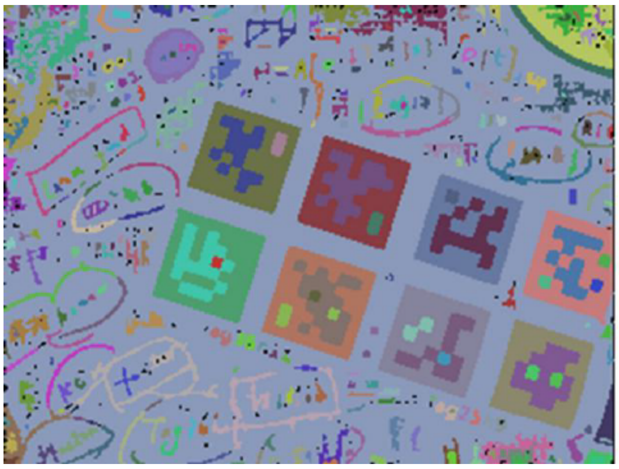
AprilTag visual reference library is adopted in HuskyLens. AprilTag can quickly detect labels and calculate relative positions through specific labels (similar to QR code, but with reduced complexity to meet real-time requirements).

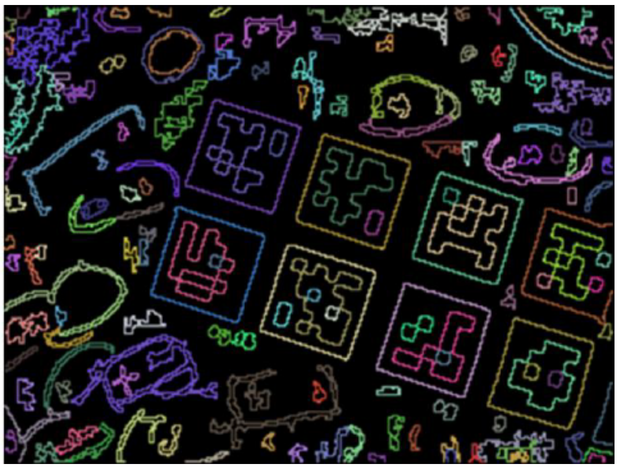
HuskyLens only supports recognition of the built-in AprilTag visual reference library tags, as shown in the following picture.

2. The Working Principle of Tag Recognition

AprilTag's algorithm mainly includes the following steps:

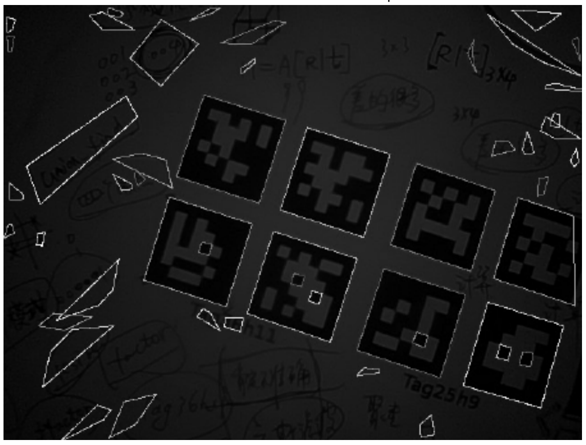
i. Edge detection, looking for the edge contour in the image.

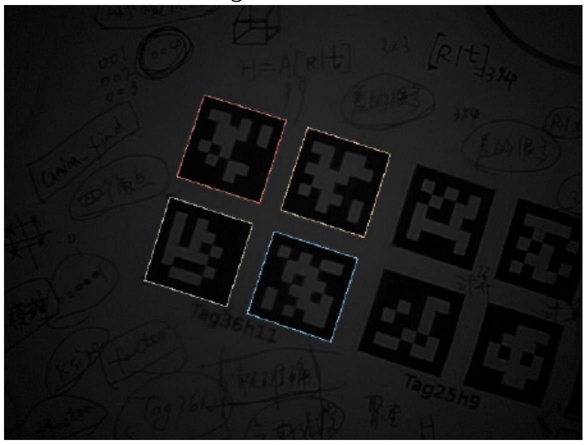




ii. Quadrangle detection, finding out quadrangles in all contour shapes.

iii. Decoding, pairing and checking the quadrangles detected.

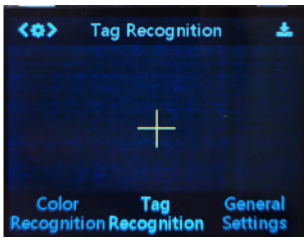




3. Demonstration of HuskyLens Tag Recognition Function

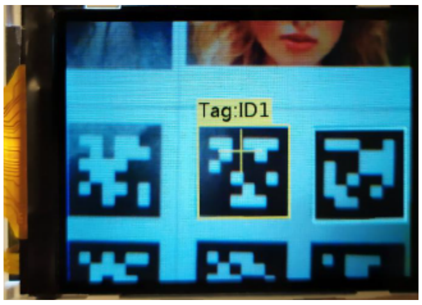
1. Detecting Tag

When HuskyLens detects the QR code tags, all the detected tags of QR code will be automatically selected with a white frame on the screen.



2. Learning Tag

Point the "+" symbol at the tag, and short press the "learning button" to complete the learning of the first Tag.



After releasing the "learning button", the screen will prompt: "Click again to continue! Click other button to end". Please short press the "learning button" within countdown if you want to learn next tag. If not, short press the "function button" before the countdown ends, or do not press any button to wait for the countdown end.

In this project, two tags need to be learned, so before the countdown ends, press the "learning button", then point the "+" symbol at the next tag to be learned, and short press the "learning button" to learn.

The number of tag ID is in the same order as the tag learning, that is, the ID will be marked as "tag: ID1", "tag: ID2", "tag: ID3", and so on. And the frame colors for different tags are also different.

3. Recognizing Tag

When HuskyLens encounters the tags that have been learnt, a colored frame with an ID will be displayed on the screen. The size of the frame will change according to the size of the QR code, and the frames will automatically trace these QR code.

Project practice:

How is the tag recognition of HuskyLens used? How to calculate the interval time? Let's break down the whole project into several small tasks and complete the simple ETC step by step.

This project will be broken down to three steps to complete the task. First, we will learn to use tag recognition function of HuskyLens and display the tag ID on the LED screen of the mainboard. Then we will learn how to use "system runtime" to calculate the time difference between two recognitions. Finally, we'll improve the whole project and achieve simple ETC function.

Task 1: Get to Know Tag Recognition

1.Hardware Connection

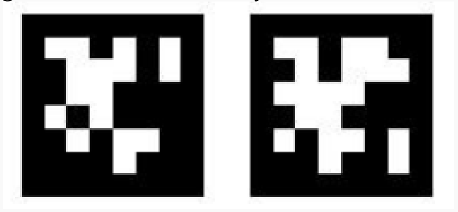
Like in the project of the numbered musical notation of color, HuskyLens is fixed on Maqueen Plus through a bracket and connected to the I2C connector.

The connection steps are the same in the following projects, so as not be repeated.

2.Program Design

STEP 1 Learning and Recognition

Here, you can choose two Tags at will to let HuskyLens learn. (Note: Firstly, switch to "learn multiple" mode)



STEP 2 Mind+ Software Settings

Just like the color recognition project, open Mind+ software (163 version or above) and switch to "Upload mode".

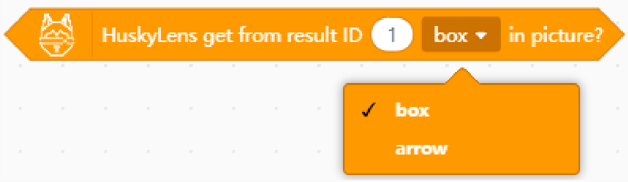
Click "Extensions" and load "micro:bit" under "Board", click to load "Maqueen Plus" under "Expansion", and click to load "HuskyLens AI Camera" under "Sensor".

The software settings in the following projects are all the same and will not be repeated.

STEP 3 Instruction Learning

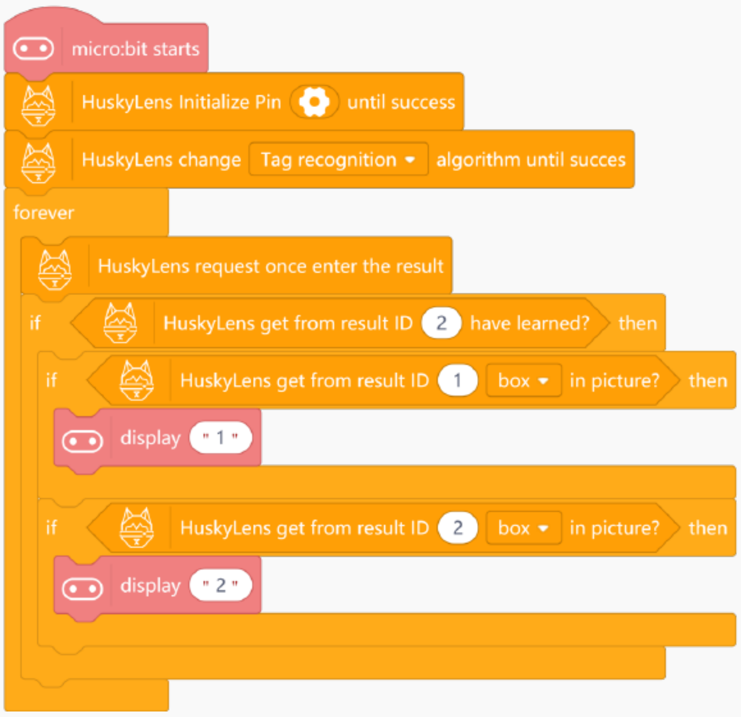
Let's take a look at some of the main instructions.

①Obtaining whether IDx is in the picture from the requested "result". The box corresponds to the algorithm targeted with box on the screen, and the arrow corresponds to the algorithm targeted with arrow on the screen. Currently, the arrow is only selected for line tracking algorithm, and the box is selected in all other algorithms.



3.Program Example

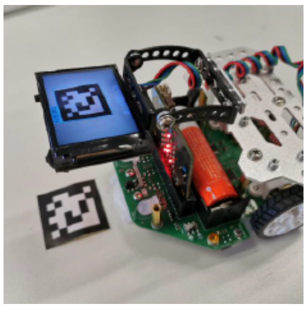
The program



```
micro:bit starts
HuskyLens Initialize Pin ⚙ until success
HuskyLens change Tag recognition ▾ algorithm until succes
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID 2 have learned? then
        if HuskyLens get from result ID 1 box ▾ in picture? then
            display " 1 "
        if HuskyLens get from result ID 2 box ▾ in picture? then
            display " 2 "
```

4.Execution Result

When HuskyLens recognizes the QR code in ID1, the LED screen of the mainboard displays 1; when HuskyLens recognizes the QR code in ID2, the screen displays 2.



Task 2: Record recognition time point of two QR codes and calculate the time difference

1. Program Design

STEP 1 Function Analysis

From life experience, a car is charged from entering into one expressway toll station, and charging stops when it exits from another toll station.

Corresponding to our project, the start time point is recorded when ID1 QR code is identified and the ending time is recorded when the ID2 QR code is identified. After that, the time difference is concluded and the mileage fee is calculated.

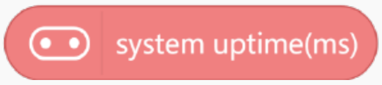
There may be a problem here: what if the ID1 QR code is recognized twice by accident?

Each "in" corresponds to an "out" on expressway. We can adopt such a one-to-one correspondence method as well. When ID1 is recognized for the first time, the start time is recorded, and thereafter the end time is recorded only when ID2 is recognized afterwards.

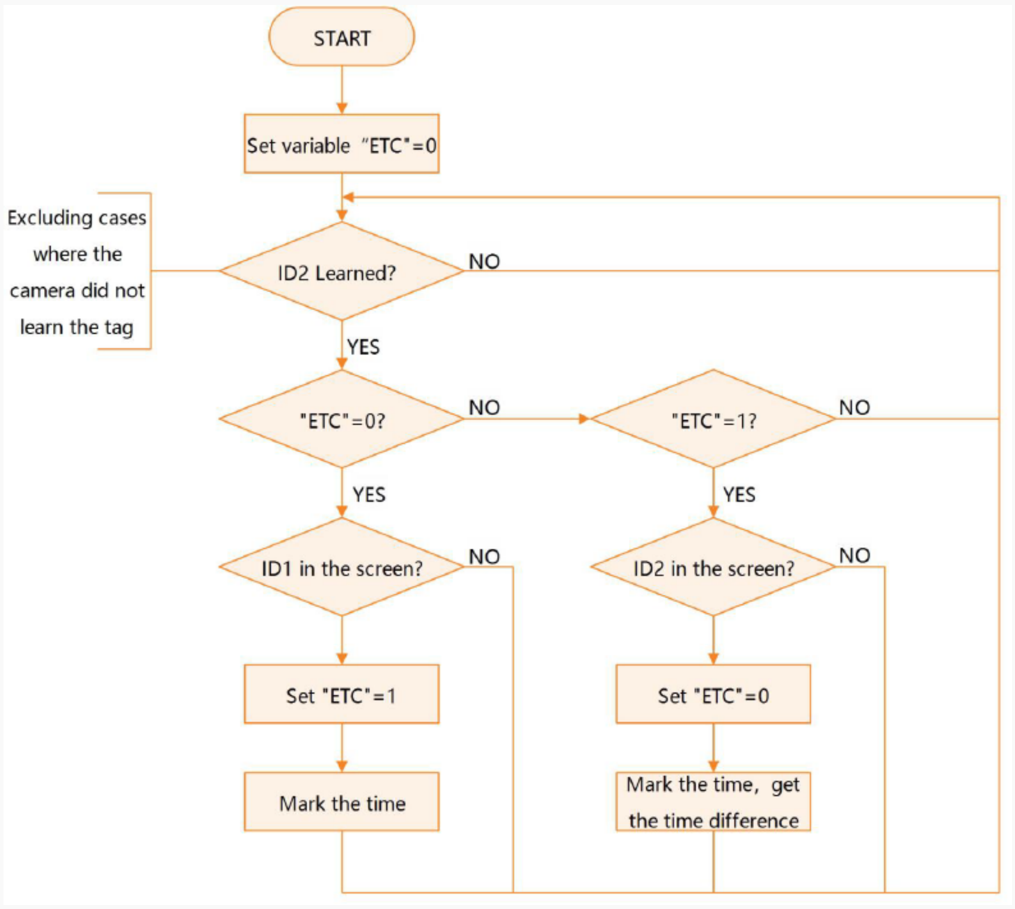
STEP 2 Instruction Learning

Let's take a look at some of the main instructions.

Indicating the real-time after the program starts running. The time unit for reading is millisecond.

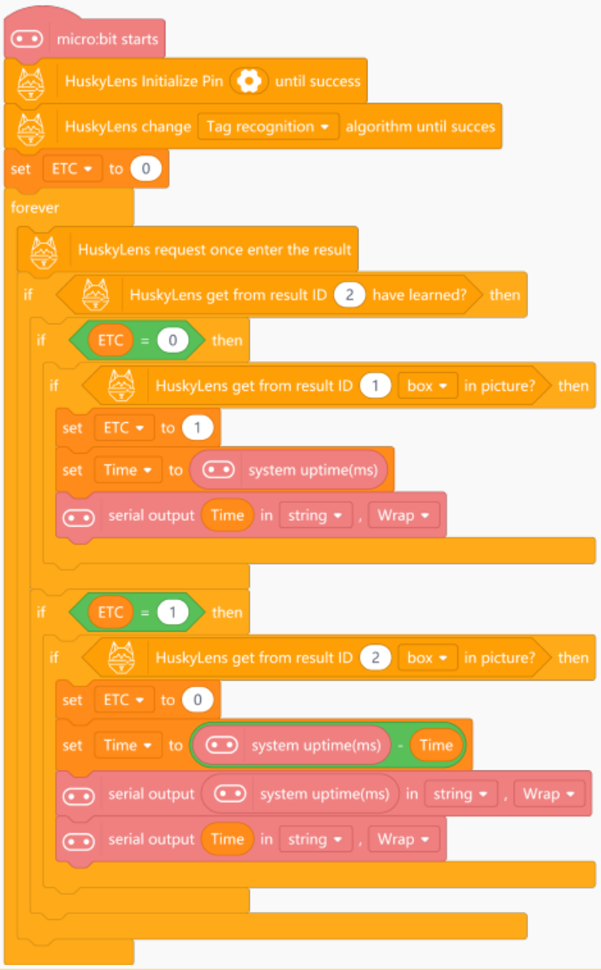


STEP 3 Flow Chart Analysis



2. Program Example

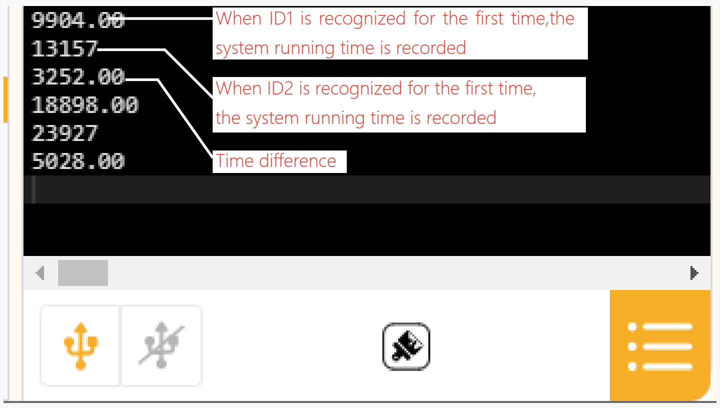
The program



```
micro:bit starts
HuskyLens Initialize Pin until success
HuskyLens change Tag recognition algorithm until succes
set ETC to 0
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID 2 have learned? then
        if ETC = 0 then
            if HuskyLens get from result ID 1 box in picture? then
                set ETC to 1
                set Time to system uptime(ms)
                serial output Time in string , Wrap
        if ETC = 1 then
            if HuskyLens get from result ID 2 box in picture? then
                set ETC to 0
                set Time to system uptime(ms) - Time
                serial output system uptime(ms) in string , Wrap
                serial output Time in string , Wrap
```

3. Execution Result

When ID1 is recognized, the serial port outputs the system running time once, and then only when ID2 is recognized, the serial port outputs the system running time again and time difference as well.



```
9904.00
13157
3252.00
18898.00
23927
5028.00
```

When ID1 is recognized for the first time,the system running time is recorded
When ID2 is recognized for the first time, the system running time is recorded
Time difference

Task 3: Start Maqueen Plus

1. Program Design

STEP 1 Function Analysis

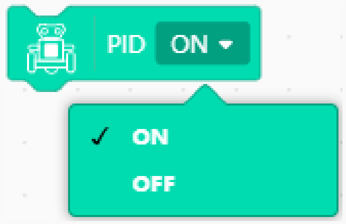
In order to see the effect easily, we can start Maqueen Plus when the ID1 QR code is recognized and stop it when the ID2 QR code is recognized.

STEP 2 Instruction Learning

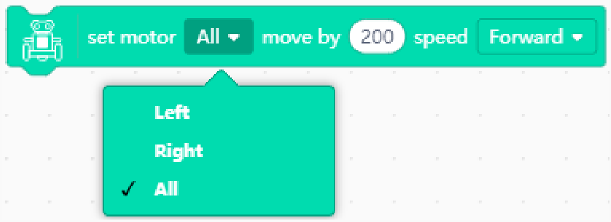
Let's take a look at some of the main instructions.

①Opening PID can control the rotation speed of the wheel in a closed loop, so that the actual rotation speed of the wheel is not interfered by the environment and is close to the set rotation speed.

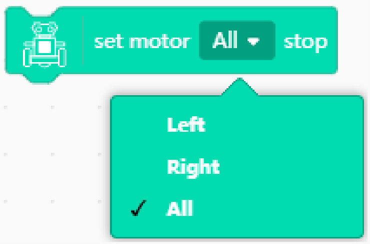
Note: PID algorithm has a certain delay and is not recommended to be used together with greyscale line tracking.



②Set the motor speed and direction.



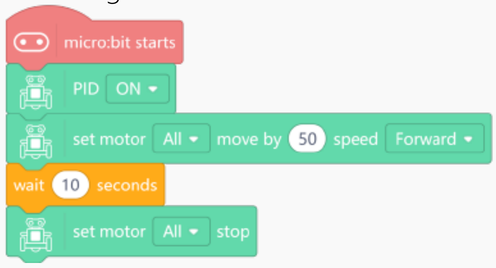
③Set motor stop



STEP3 Speed Test

Using PID to control the wheel speed helps for precision of distance measurement. In order to calculate the actual mileage of Maqueen Plus, the speed must be measured first.

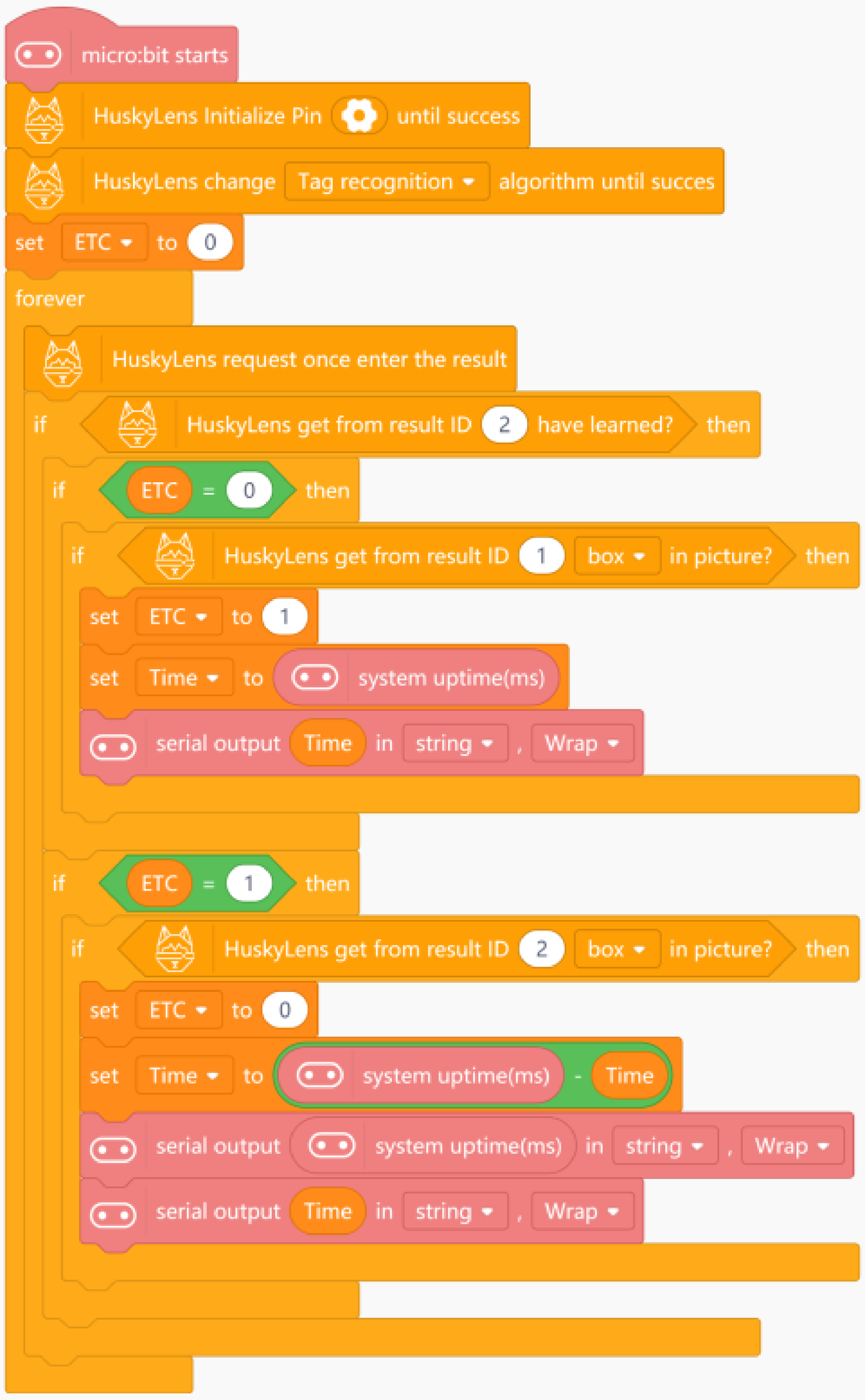
Test Program:



Test Method: When the Maqueen Plus runs the above program, it will move forward for 10 seconds at a speed of 50. Record the starting and ending positions, measure the distance S (unit: cm), and then S/10 is the car mileage speed (unit: cm/sec).

2. Program Example

The program



```
micro:bit starts
HuskyLens Initialize Pin ⚙ until success
HuskyLens change Tag recognition ▾ algorithm until succes
set ETC ▾ to 0
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID 2 have learned? then
        if ETC = 0 then
            if HuskyLens get from result ID 1 box ▾ in picture? then
                set ETC ▾ to 1
                set Time ▾ to system uptime(ms)
                serial output Time in string ▾ , Wrap ▾
        if ETC = 1 then
            if HuskyLens get from result ID 2 box ▾ in picture? then
                set ETC ▾ to 0
                set Time ▾ to system uptime(ms) - Time
                serial output system uptime(ms) in string ▾ , Wrap ▾
                serial output Time in string ▾ , Wrap ▾
```

3. Execution Result

When HuskyLens recognizes ID1 QR code, Maqueen Plus starts; after that, when HuskyLens recognized the ID2 QR code, Maqueen Plus stops.

Project Summary:

Project Review

Understand the working principle of tag recognition in this lesson and learn the tag recognition function by using HuskyLens.

Combined with the PID function of Maqueen Plus, a simple ETC was created. The mileage and expressway toll were not calculated at the end of the project. Believing you can finish this part of data calculation on your own and display the final result on the LED screen.

Knowledge Nodes Recap

1.Understand the working principle of tag recognition;

2.Learn function and operation method of tag recognition of HuskyLens;

3.Learn PID control of Maqueen Plus

Project Extension:

After the completion of this project, you can search for local expressway toll rates on the Internet, and finalize the calculation of the toll in this project.

Please think about a question: expressway toll rates vary from provinces, but now some ETCs have already implemented trans-provincial charging, by which, no need to enter or exit trans-provincial toll stations. How does it work? Can we also simulate trans-provincial non-stop charging in our project?

(Tips: more QR codes can be used to represent different provinces. Each province is identified once. Finally, with the cost charged in each section, the total cost can be calculated.)

Project 3: AI Sorting Master

At Shanghai Yangshan Deep Water Port, we can always see such scenes. Bright red bridge cranes stretch their huge arms; tower cranes line up along the port; rail cranes lock the containers accurately. Huge containers are quickly grabbed one by one and automated guided vehicle already awaits beneath them; those containers are quickly loaded and towed to the designated position. The traffic flow in the port area is incessant, but you can hardly see a worker there.

Automated Guided Vehicle, which abbreviation is AGV, refers to an autonomous unmanned vehicle that has electromagnetic or optical automated guidance and can run along a specified guidance path.

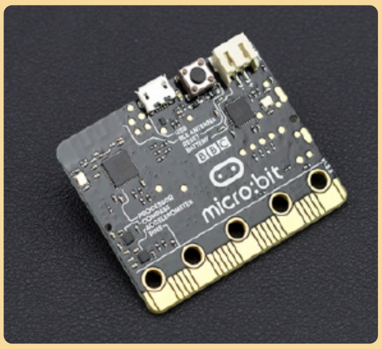
Can Maqueen Plus be acted as an AGV? Even smarter than AGV? For example, recognizing the type of transported goods and transporting to the designated position according to the recognition result. Let's make Maqueen Plus an AGV with "Eyes"!

Function Description:

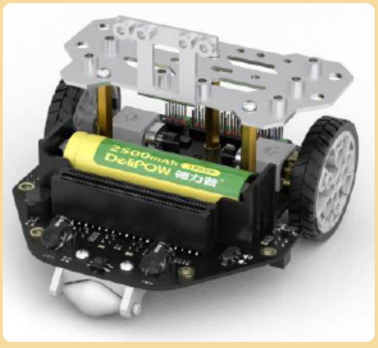
This project uses object recognition function of HuskyLens to identify different items and transport them to different positions according to the recognition results, so that Maqueen Plus can be transformed into AI sorting master!

In order to realize accurate transportation, the built-in line tracking sensor of Maqueen Plus will be used to realize fixed-point transportation through a simple line tracking algorithm.

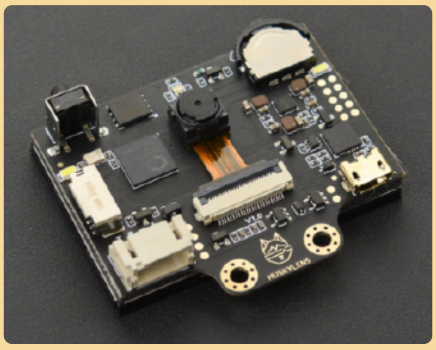
Materials Checklist:



Micro:bit ×1



Maqueen Plus ×1



HUSKYLENS ×1

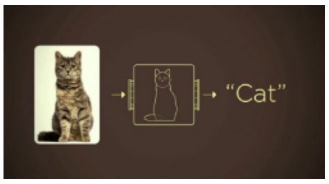
Knowledge Extension:

When we talk about computer vision, the first thing that comes to mind is image classification (object recognition). Yes, image classification is one of the most basic assignments of computer vision.

In this project, the object recognition function was applied to distinguish different kinds of objects.

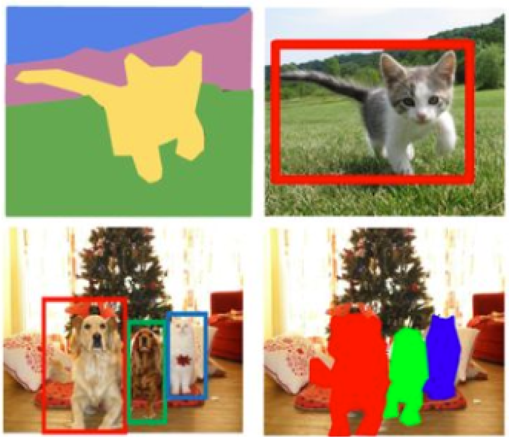
I. What is Object Recognition?

Object recognition mainly refers to the perception and understanding to the entity and environment in the three-dimensional world, and it belongs to the category of advanced computer vision.



II. The Working Principle of Object Recognition

In object recognition, there are many complicated and interesting tasks, such as target detection, object localization, image segmentation, etc. Target detection can be seen as a combination of image classification and image localization. Take an example as a given picture, the target detection system should be able to identify the target in the picture and give its location.



At present, there are two popular target detection algorithms, one is R-CNN and the other is Yolo. R-CNN algorithm has a higher accuracy but slower speed. Yolo algorithm is faster, but its accuracy is lower. The following is a brief introduction to Yolo algorithm.

Yolo Algorithm:

Yolo is an abbreviation of "you only look once", which means prediction can be made just after look once. The inspiration comes from humans ourselves. Because when humans look at a picture, we can know the location of various types of targets in the picture at a glance.

Yolo excels in its simplicity and speed. If target detection is regarded as a process of fishing, other algorithms are just like sniping fishes with fork one by one. When it comes to Yolo, it just scatters a fishing net and catch them all!

Yolo makes prediction based on the whole picture, and it will output all the detected target information at once, including the classification and location. Yolo algorithm is implemented by splitting the input picture into SxS grids. If the center of an object falls in one grid, the corresponding grid is responsible for predicting the size and class of the target. As shown in the picture below, the center of the dog falls into the blue grid, so the blue grid is responsible for the information prediction of this target object.

"The grid on which the center of the object falls is responsible for predicting the object" is divided into two stages, including training stage and testing stage.

In the training stage, if the center of the object falls in this grid, the algorithm will teach this grid to predict the objects in the image. In the testing stage, the grid will naturally continue to do so because it has been taught in the training stage to predict the objects whose centers fall in the grid.

III. Demonstration of HuskyLens Object Recognition Function

The object recognition function in HuskyLens can identify what the object is and track it.

At present, 20 kinds of objects are supported: planes, bicycles, birds, boats, bottles, buses, cars, cats, chairs, cattle, dining tables, dogs, horses, motorcycles, mankind, potted plants, sheep, sofas, trains and televisions.

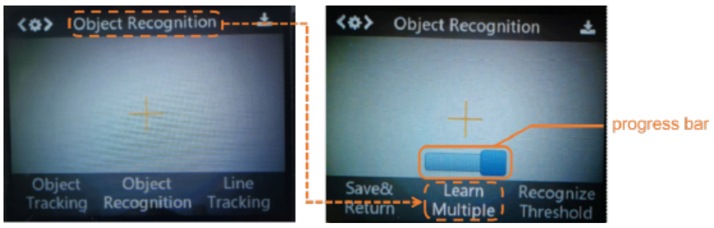
The default setting is to identify one object, but learn multiple can also be set up.

1. Operation and Setting - Learn Multiple

Dial the function button to the right or left until the word "Object Recognition" is displayed at the top of the screen. Long press the function button to enter the parameter setting of the object recognition function.

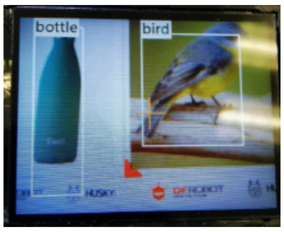
Dial the function button until "Learn Multiple" is displayed, then short press the function button, and dial to the right to turn on the "Learn Multiple" switch, i.e. progress bar turns blue and the square icon on the progress bar moves to the right. Then short press the function button to confirm this parameter.

Dial the function button to the left until "Save & Return" shows. And the screen prompts "Do you want to save the parameters?" Select "Yes" in default, now short-press the function button to save the parameters and return automatically.



2. Detecting Object

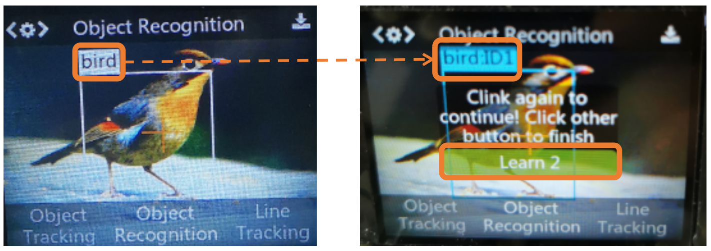
When detecting objects, HuskyLens will automatically recognize it. On the screen, white bounding boxes pick out the object with their names. At present, only 20 built-in objects can be recognized, and the rest objects cannot be recognized so far.



3. Learning Object

Point HuskyLens at the target object. When the object displayed on the screen is detected and its name is displayed, point the "+" symbol at the object, then short press the "learning button". When pressing, the color of the bounding box changes from white to blue, and the name of the object and its ID number will appear on the screen. At the same time, a message prompts: "Click again to continue! Click other button to finish". Please short press the "learning button" before the countdown ends if you want to learn next object. If not, short press the "function button" before the countdown ends, or do not press any button to let the countdown ends.

The object IDs displayed on HuskyLens are in the same order as the marking objects, that is, the IDs will be marked as "ID1", "ID2", "ID3" and so on. And the bounding box colors corresponding to different object IDs are also different.

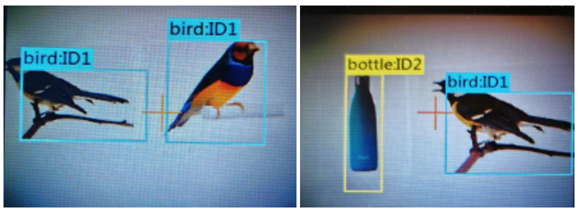


4. Recognizing Object

When HuskyLens encounters the objects that have been learnt, there will be colored bounding boxes on the screen to pick out these objects and display the object names and IDs. The size of the boundary will change according to the size of the object, and the bounding boxes will automatically trace these objects. The same class of objects are in same box color, name and ID. Simultaneous recognition of multiple types of objects are supported, such as recognizing bottles and birds at the same time.

This function can be used as a simple filter to find out the target object from a pile of objects and trace it.

* This function are not able to tell the difference within same class of objects. For example, it can only identify the object is cat, but cannot identify what kind of cat it is. So it is different from face recognition function which different faces can be distinguished.



Project practice:

How is object recognition of HuskyLens used? How to control Maqueen Plus to follow the designated route? Let's break down the whole project into several small tasks and complete AI sorting master step by step.

This project will be broken down into three steps. First, we will learn to use object recognition function of HuskyLens and output the object name through serial port. Then we will learn the grey-scale line tracking algorithm to realize the fixed-point movement. Finally, we'll improve the whole project and simulate sorting and transportation scenario.

Task 1: Get to Know Object Recognition

1.Program Design

Learning and Recognition: Select 3 items here for HuskyLens to learn. (Note: First switch to "Learn Multiple" function)

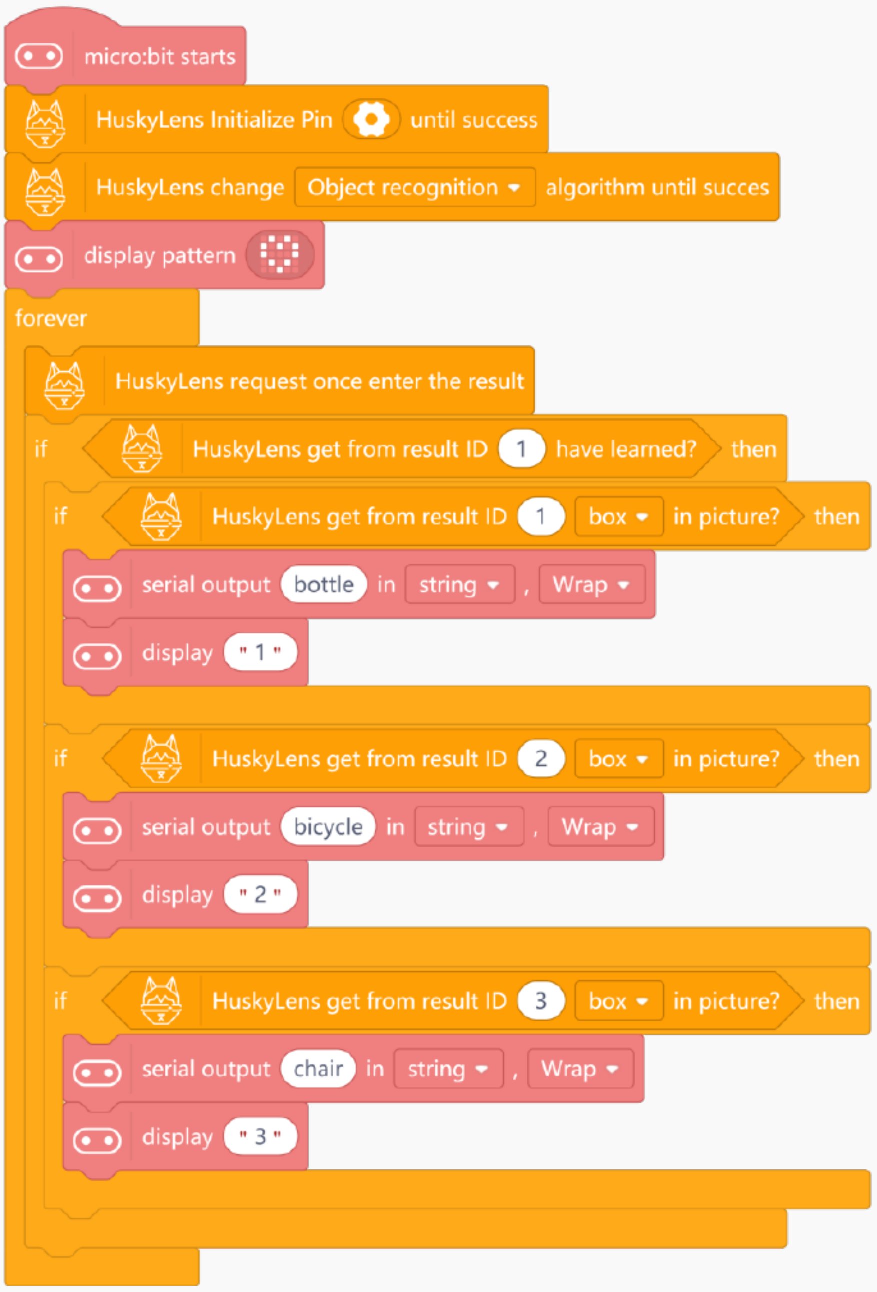
ID1: Bottle

ID2: Bicycle

ID3: Chair

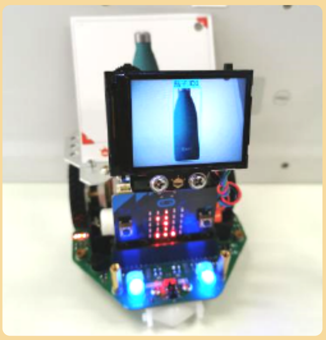
2.Program Example

The program



```
micro:bit starts
  HuskyLens Initialize Pin ⚙ until success
  HuskyLens change Object recognition ▾ algorithm until succes
  display pattern
forever
  HuskyLens request once enter the result
  if HuskyLens get from result ID 1 have learned? then
    if HuskyLens get from result ID 1 box ▾ in picture? then
      serial output bottle in string ▾ , Wrap ▾
      display " 1 "
    if HuskyLens get from result ID 2 box ▾ in picture? then
      serial output bicycle in string ▾ , Wrap ▾
      display " 2 "
    if HuskyLens get from result ID 3 box ▾ in picture? then
      serial output chair in string ▾ , Wrap ▾
      display " 3 "
```

3.Execution Result


ID1: Bottle

ID2: Bicycle

ID3: Chair

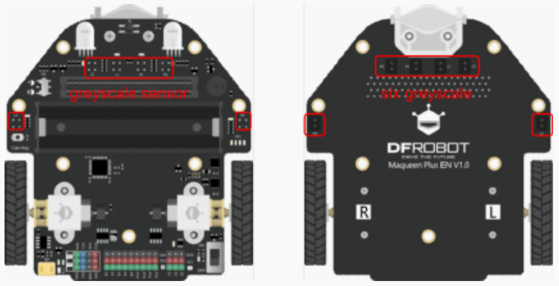
When HuskyLens recognizes the bottle, the screen of the mainboard displays 1 and the serial port outputs bottle. When HuskyLens recognizes the bicycle, the screen on the mainboard displays 2 and the serial port outputs bicycle. When HuskyLens recognizes the chair, the screen of the mainboard displays 3 and the serial port outputs chair.

Task 2: Greyscale Line Tracking - 2IO Algorithm

1. Program Design

STEP 1 Function Analysis

As an AI sorting master, Maqueen Plus uses HuskyLens' object recognition to identify the object type. But the sorting task can only be completed when the object is transported to the designated location, just like the AGV in the port, which completes the automated transportation.



For this kind of point-to-point accurate transportation, you can use the built-in greyscale sensor of Maqueen Plus and implement it through line tracking.

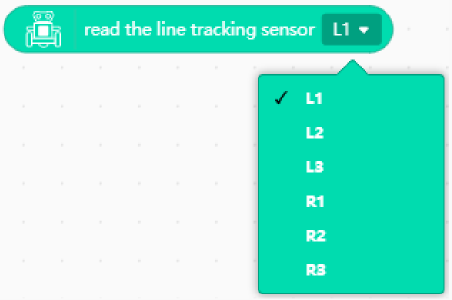
Maqueen Plus has 6 built-in greyscale sensors at the bottom, which can be used to detect black line.

When the greyscale sensor faces white background, the greyscale indicator LED is off, and the detection value of the greyscale sensor is 0; when the greyscale sensor faces the black line, the greyscale indicator LED is on, and the detection value of the greyscale sensor is 1.

STEP 2 Instruction Learning

Let's take a look at some of the main instructions.

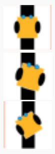
①Read the value of the line tracking sensor, and the feedback value is 0 or 1, 1 is indicated on the black line. Select greyscale in the drop-down box, L1, L2, L3, R1, R2, R3 are consistent with the logo at the bottom of Maqueen Plus.



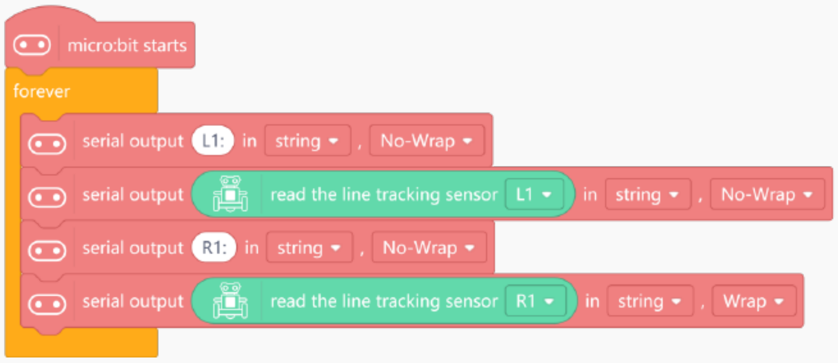
STEP3 Greyscale Test

Output greyscale value through serial port. Test program:

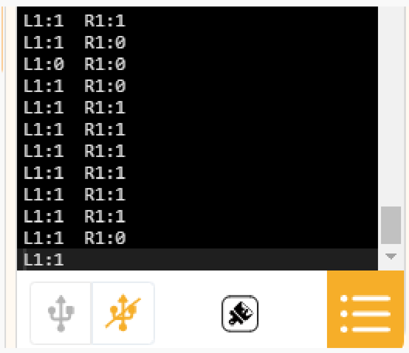
Test results:



When the greyscales L1 and R1 are both on the black line, the serial port outputs 1, 1;
when only L1 is on the black line, serial port outputs 1, 0;
when only R1 is on the black line, serial port outputs 0, 1;
if both greyscales are not on the black line, 0, and 0 will be output.



```
micro:bit starts
forever
  serial output L1: in string , No-Wrap
  serial output read the line tracking sensor L1 in string , No-Wrap
  serial output R1: in string , No-Wrap
  serial output read the line tracking sensor R1 in string , Wrap
```

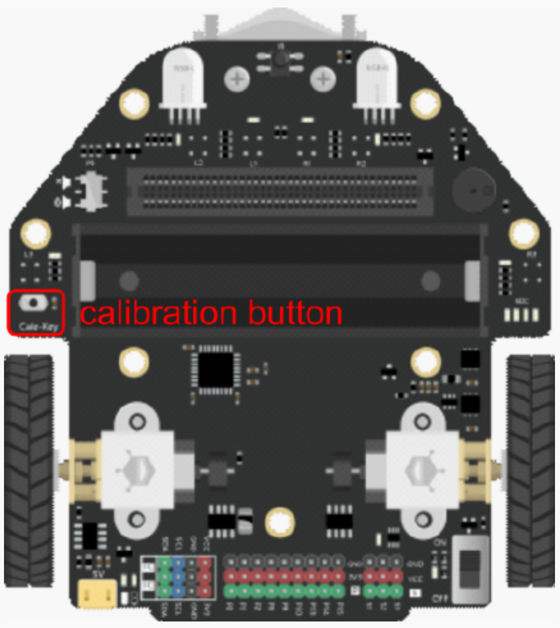


```
L1:1  R1:1
L1:1  R1:0
L1:0  R1:0
L1:1  R1:0
L1:1  R1:1
L1:1  R1:1
L1:1  R1:1
L1:1  R1:1
L1:1  R1:1
L1:1  R1:1
L1:1  R1:0
L1:1
```

If the program does not work properly, troubleshoot the following problems:

(1) The black color printed by the printer may not be correctly recognized. The black adhesive tape and printed map can be used normally.

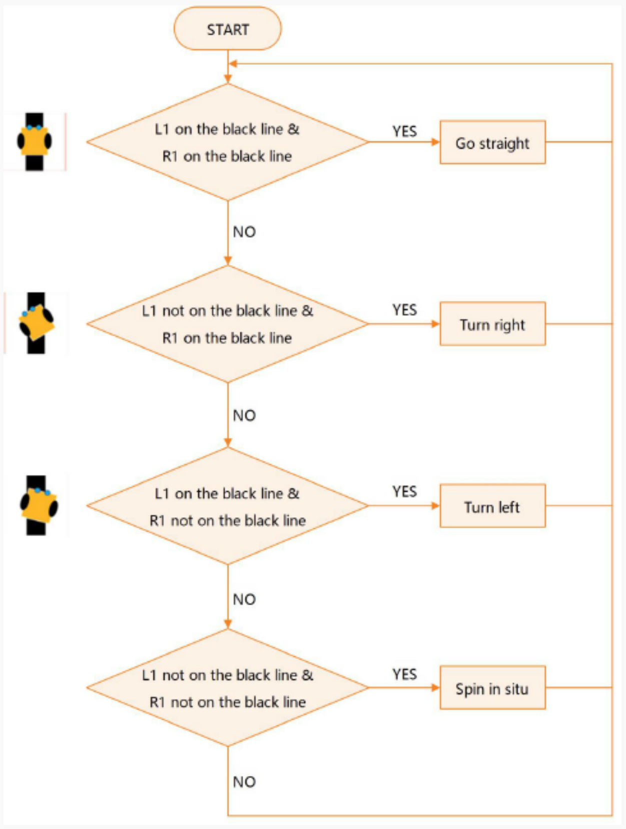
(2) Ambient light may affect the greyscale sensor. When the light changes greatly, the greyscale need to be recalibrated. Greyscale calibration method: Maqueen Plus has a one-button greyscale calibration function. The calibration button is shown in the following picture. When in use, ensure that all line tracking sensors are in the black calibration area. Press the calibration button for 1 second. The two RGB lights in front of Maqueen Plus flash in green, indicating the calibration is completed. Release the button to complete.



*Principle of greyscale sensor: each greyscale sensor consists of an infrared emitter and an infrared receiver. Because it is often used to control robots to walk along the line, it is also called a line tracking sensor. The infrared transmitter continuously emits infrared light to the ground. If the infrared light is reflected (such as meeting white or other light-colored planes), the receiver receives the infrared signal, outputs a value of 0, and the greyscale indicator LED is off. If infrared light is absorbed or cannot be reflected, the receiver will not receive infrared signals, output a value of 1, and the greyscale indicator LED will be on.

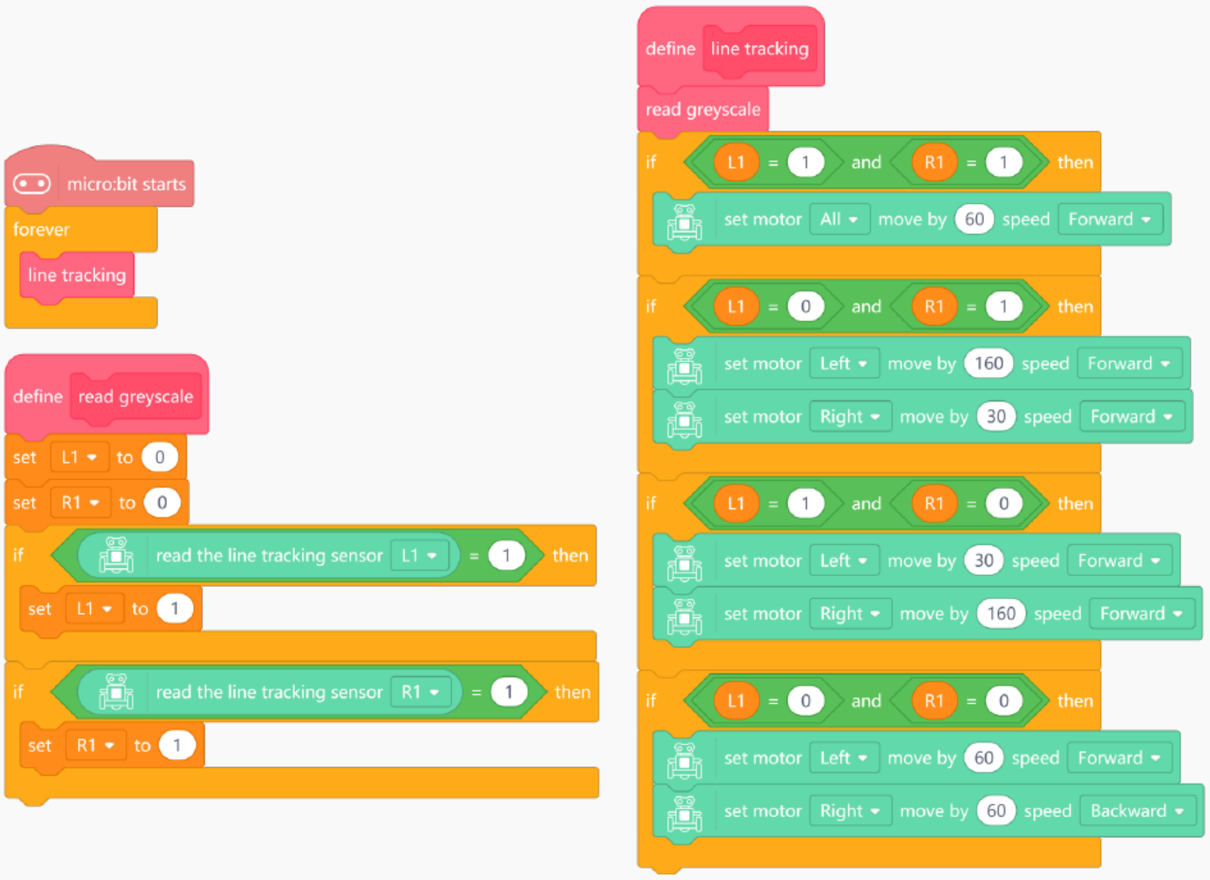
STEP 4 Flow Chart Analysis

Here we use two greyscales L1 and R1 to patrol the line, and the default width of black line is 2cm.When L1 and R1 are both on the black line, Maqueen Plus goes straight ahead.


START
L1 on the black line & R1 on the black line
YES
Go straight
NO
L1 not on the black line & R1 on the black line
YES
Turn right
NO
L1 on the black line & R1 not on the black line
YES
Turn left
NO
L1 not on the black line & R1 not on the black line
YES
Spin in situ
NO

2. Program Examples

The program



```
micro:bit starts
forever
    line tracking

define read greyscale
set L1 to 0
set R1 to 0
if read the line tracking sensor L1 = 1 then
    set L1 to 1
if read the line tracking sensor R1 = 1 then
    set R1 to 1

define line tracking
read greyscale
if L1 = 1 and R1 = 1 then
    set motor All move by 60 speed Forward
if L1 = 0 and R1 = 1 then
    set motor Left move by 160 speed Forward
    set motor Right move by 30 speed Forward
if L1 = 1 and R1 = 0 then
    set motor Left move by 30 speed Forward
    set motor Right move by 160 speed Forward
if L1 = 0 and R1 = 0 then
    set motor Left move by 60 speed Forward
    set motor Right move by 60 speed Backward
```

3. Operation Effect

If you don't have a suitable map at hand, it is recommended to stick a patrol map with 2cm-wide black tape, such as the following picture.

When running the program, the car will automatically patrol the black line.

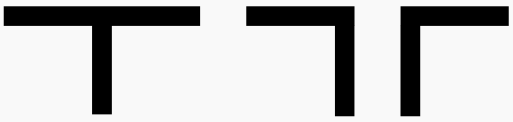
Task 3: Setting up and Optimizing Programs

1. Program Design

STEP 1 Function Analysis

Assuming that three kinds of objects need to be sorted, Maqueen Plus will complete transportation along a certain track after recognizing the object class.

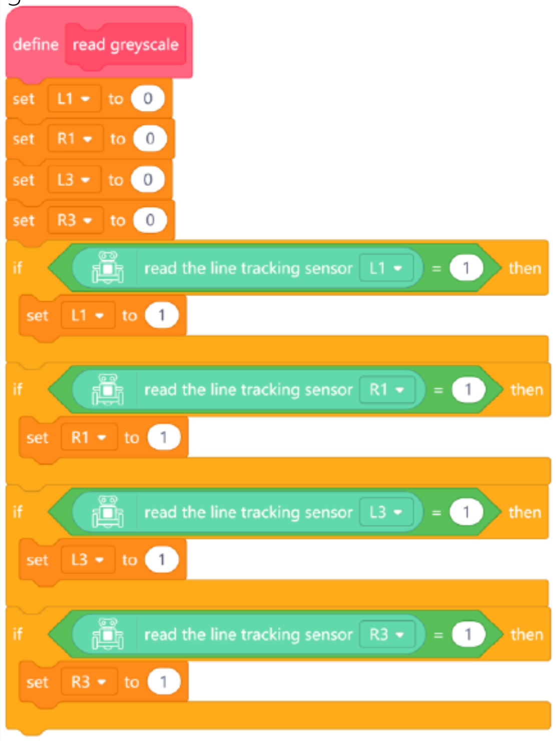
In order to accurately control Maqueen Plus, T-junction, left junction and right junction (as shown in the following picture) are added to the line tracking map, and the junction judgment is correspondingly added to the programs.



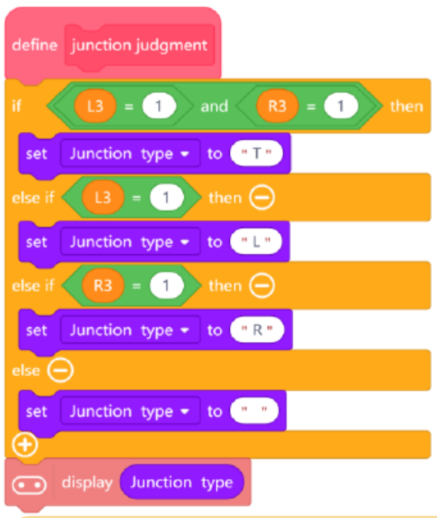
Using L3 and R3 greyscales on Maqueen Plus to assist in judging junctions.

STEP 2 Junction Judgment Program

Modify the function "read greyscale" in task 2 and create functions "junction judgment", "turn left" and "turn right".

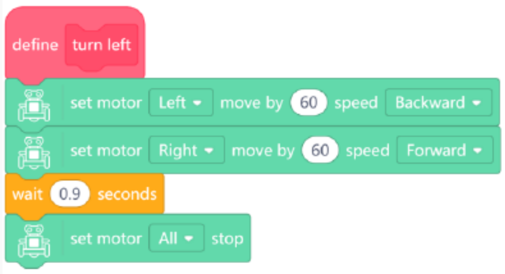


```
define read greyscale
set L1 to 0
set R1 to 0
set L3 to 0
set R3 to 0
if read the line tracking sensor L1 = 1 then
    set L1 to 1
if read the line tracking sensor R1 = 1 then
    set R1 to 1
if read the line tracking sensor L3 = 1 then
    set L3 to 1
if read the line tracking sensor R3 = 1 then
    set R3 to 1
```

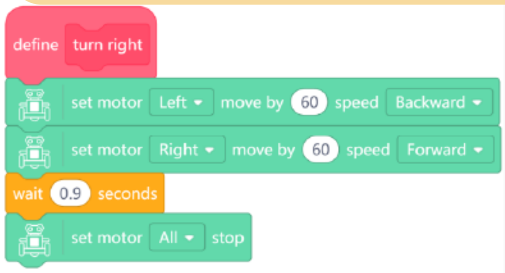


```
define junction judgment
if L3 = 1 and R3 = 1 then
    set Junction type to "T"
else if L3 = 1 then
    set Junction type to "L"
else if R3 = 1 then
    set Junction type to "R"
else
    set Junction type to " "
display Junction type
```

It is suggested that when debugging the program, the above functions "junction judgment", "turn left" and "turn right" should be run separately to test whether each function can execute the corresponding function.

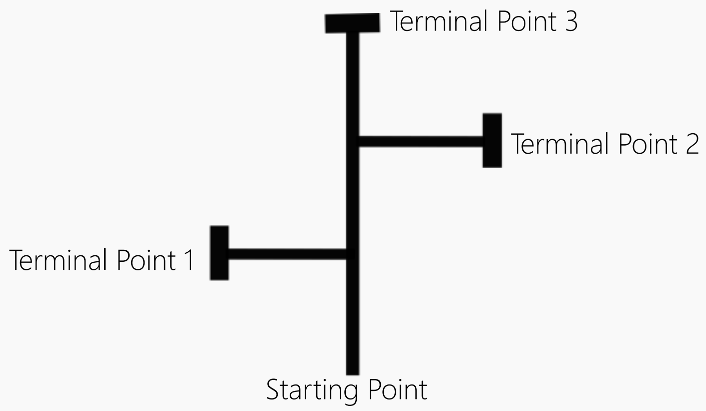


```
define turn left
set motor Left move by 60 speed Backward
set motor Right move by 60 speed Forward
wait 0.9 seconds
set motor All stop
```



```
define turn right
set motor Left move by 60 speed Backward
set motor Right move by 60 speed Forward
wait 0.9 seconds
set motor All stop
```

STEP 3 Map Example



STEP 4 Flow Chart Analysis

Different maps need different route planning. Take the map above as an example, the route planning from the starting point to the end point 1 is: if Maqueen Plus patrols along the line until it meets the left junction, Maqueen Plus turns left, and it continues to move along the line until it meets the T junction, it reaches the terminal point 1.If HuskyLens recognizes the object of ID1, it will go to the terminal point 1 according to the route planning. Terminal point 2, 3 and so forth.

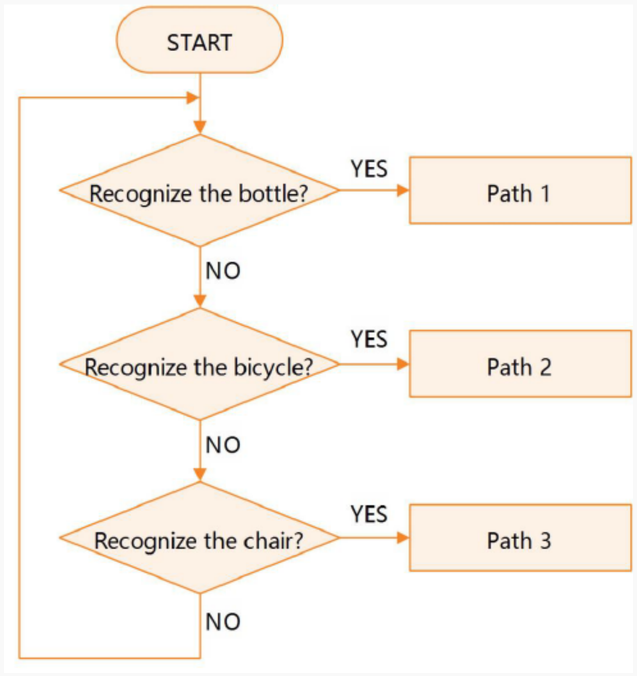
The sequence of object recognition in Task 1 is copied here.

ID1: Bottle

ID2: Bicycle

ID3: Chair

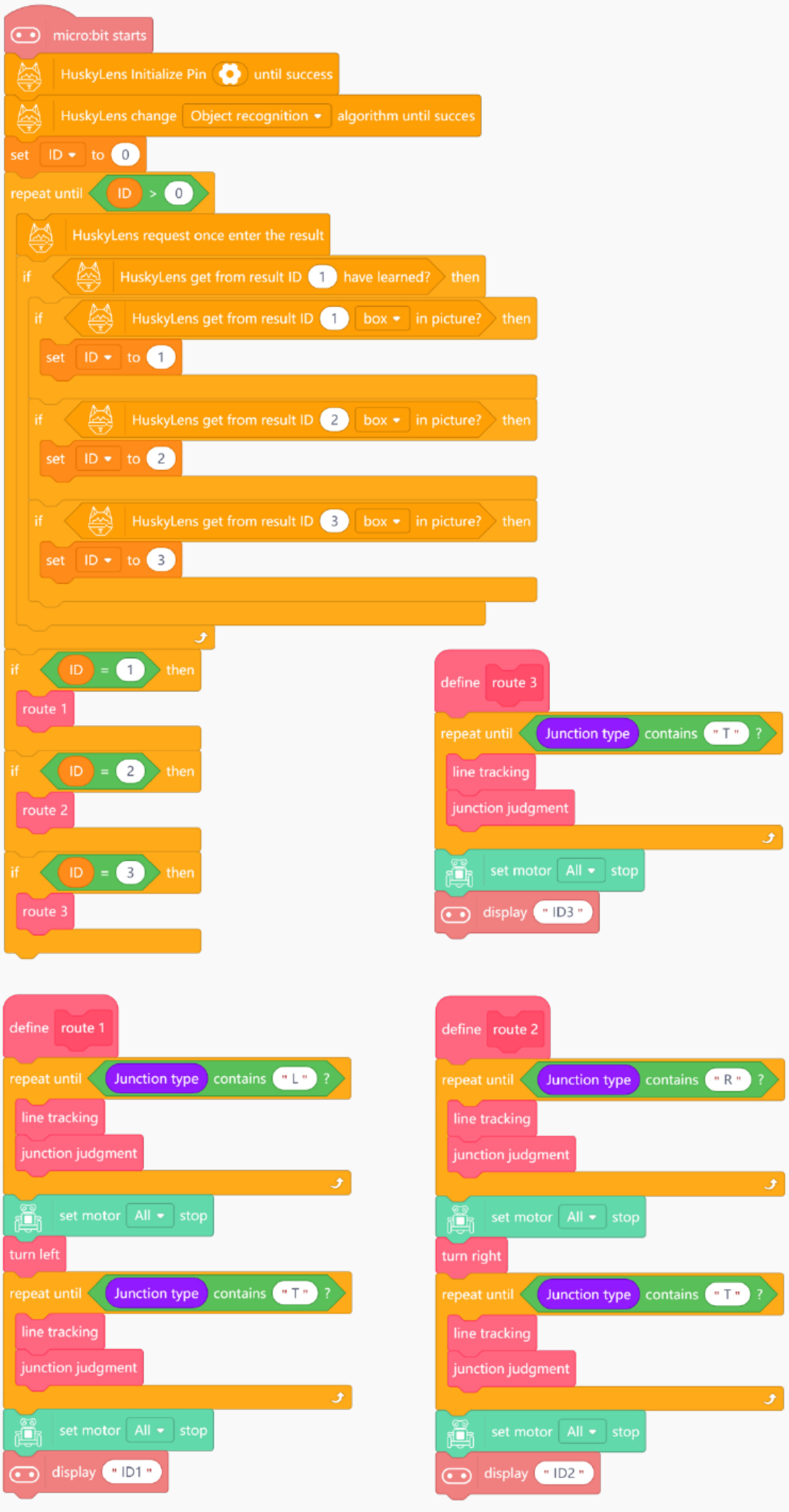
The flow chart shows as follows:



2. Program Example

Modify the main program and add functions "route 1", "route 2" and "route 3".When you are actually programming, refer to your own map to make a route plan. The following program uses the previous example map.

The program



```
micro:bit starts
    HuskyLens Initialize Pin [I2C] until success
    HuskyLens change [Object recognition ▾] algorithm until succes
    set [ID ▾] to (0)
    repeat until <(ID) > (0)>
        HuskyLens request once enter the result
        if <HuskyLens get from result ID (1) have learned?> then
            if <HuskyLens get from result ID (1) [box ▾] in picture?> then
                set [ID ▾] to (1)
            if <HuskyLens get from result ID (2) [box ▾] in picture?> then
                set [ID ▾] to (2)
            if <HuskyLens get from result ID (3) [box ▾] in picture?> then
                set [ID ▾] to (3)
    if <(ID) = (1)> then
        route 1
    if <(ID) = (2)> then
        route 2
    if <(ID) = (3)> then
        route 3

define route 3
    repeat until <(Junction type) contains (" T ") ?>
        line tracking
        junction judgment
    set motor [All ▾] stop
    display (" ID3 ")

define route 1
    repeat until <(Junction type) contains (" L ") ?>
        line tracking
        junction judgment
    set motor [All ▾] stop
    turn left
    repeat until <(Junction type) contains (" T ") ?>
        line tracking
        junction judgment
    set motor [All ▾] stop
    display (" ID1 ")

define route 2
    repeat until <(Junction type) contains (" R ") ?>
        line tracking
        junction judgment
    set motor [All ▾] stop
    turn right
    repeat until <(Junction type) contains (" T ") ?>
        line tracking
        junction judgment
    set motor [All ▾] stop
    display (" ID2 ")
```

Project 4: Undercover Detective

After years of development, face recognition technology has begun to roll out. For example, when entering the railway station, the ID card information is compared through the camera. And the face scan clock in/out at work, the face scan payment by Alipay, the face-to-unlock on the smart phone, etc.

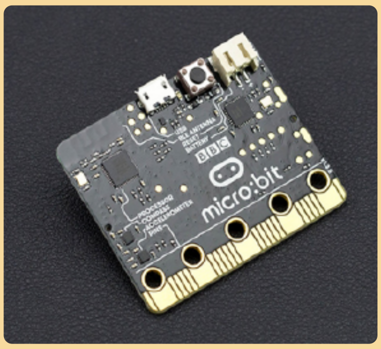
Face recognition technology has also been effectively applied in the field of public security, such as arresting fugitives. In recent years, many fugitives have been successfully arrested many times in singing stars' concerts. Using AI face recognition technology, when a fugitive passes through security check, the camera captures the face information of the fugitive and compares it with the back-end database, then the system will issue a warning message.

HuskyLens has a built-in face recognition function. Can Maqueen Plus combine with HuskyLens to catch bad guys? In real life, bad guys often have to take the initiative to go to places with face recognition before they can be identified and caught. So how can bad guys be caught at those places without face recognition cameras?

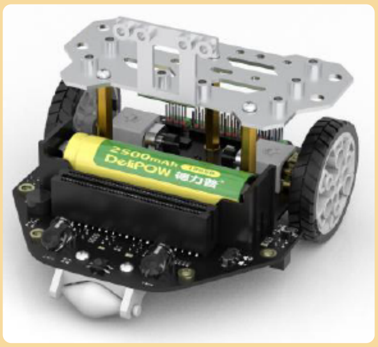
Function Description:

This project uses HuskyLens face recognition function to make Maqueen Plus become an undercover detective in the folk. At ordinary times, it can move freely and has basic obstacle avoidance function. Through face recognition, the face in the screen is compared with the back-end database, and an alarm will be triggered once bad guy is found.

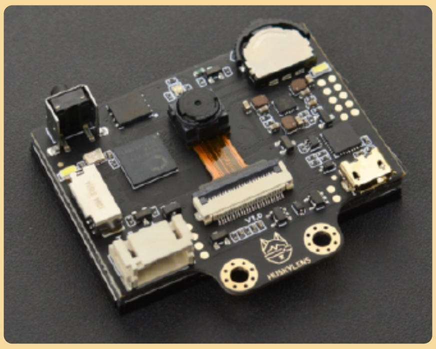
Materials Checklist:



Micro:bit ×1



Maqueen Plus ×1



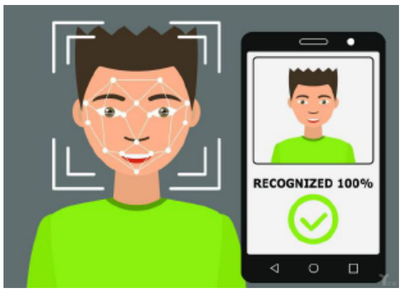
HUSKYLENS ×1

Knowledge Extension:

The human face, like other biological features of the human body (the fingerprint, the iris, etc.), is born with uniqueness and is not easy to be duplicated. Face images belong to the earliest category of images studied and are also the most widely used in the field of computer vision. This project uses HuskyLens face recognition function.

I. What is Face Recognition?

Face recognition is a kind of biometric identification technology based on human facial feature information. By collecting images or videos containing human faces with image camera or video camera, automatically detecting image information and tracking human faces, a series of technical analysis on the detected human faces is performed.



II. The Working Principle of Face Recognition

There are four key steps in the process of face recognition:

Four key steps in the process of face recognition

Face Detection   Face Alignment   Face Encoding   Face Comparison

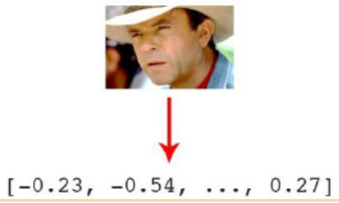
The following is a brief description of these 4 steps.

①Face Detection: Find the location of face in the picture, usually marked with a bounding box.

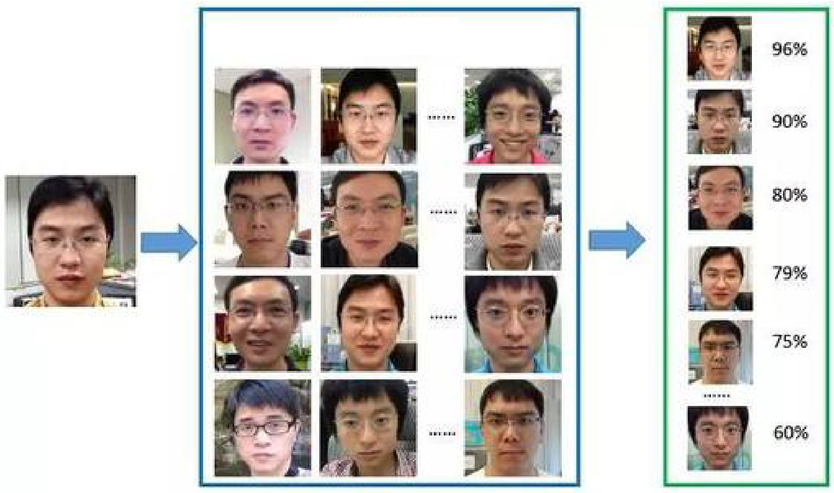
②Face Alignment: Identify faces from different angles by locating feature points on the face.

Left Side Face (Profile)
Left Side Face (Simi-Frontal)
Frontal Face
Right Side Face (Simi-Frontal)

③Face Encoding: It can be simply understood as extracting face information and converting it into information understandable by computers.



④Face Comparison: Compare the face information to the existing database to obtain a similarity score and give comparison result.



Face recognition is also considered to be one of the most difficult research topics in the field of biometric recognition or even artificial intelligence area. The difficulty of face recognition is mainly attributed to the characteristics of human face as a biological feature.

Similarity: There is little difference between each individual. The structure of all human faces is similar, even the structure and appearance of human face organs are very similar. This feature is beneficial to locate human face, but unbeneficial to distinguish human individual by human face.

Changeability: The appearance of a human face is unstable. People can produce many expressions through changes in the face, and the visual images of the face vary greatly in different observation angles. In addition, face recognition is also influenced by various factors including lighting conditions (such as day and night, indoor and outdoor, etc.), covers on the face (such as masks, sunglasses, hair, beard, etc.), age, etc.

III. Face Recognition Application Scenarios

Access Control System: Areas under security protection can identify the identity of those trying to enter through face recognition, such as prisons, detention centers, residential areas, schools, etc.

Video Surveillance System: Monitoring crowds in public places such as banks, airports, stadiums, shopping malls, supermarkets, etc. to achieve the purpose of identification. For example, monitoring system is installed in airport to prevent terrorists from boarding planes.

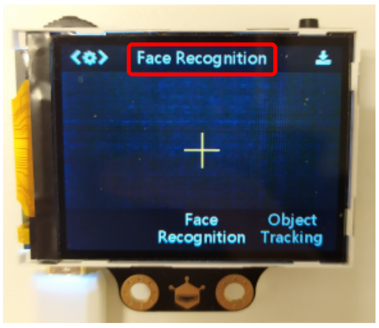
Network Application: Using face recognition to assist credit card online payment to prevent non-credit card owners from using credit cards or to prevent false claim of social insurance, etc.

Face recognition is widely used in all walks of life at present, such as student attendance system, camera, unlocking mobile phone, and all-in-one person authentication system.

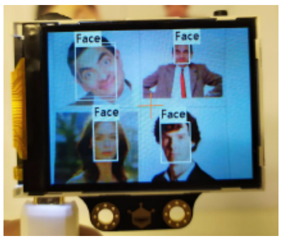
IV. Demonstration of HuskyLens Face Recognition Function

1. Select the "Face Recognition" Function

Dial the function button to the left until the word "Face Recognition" is displayed at the top of the screen.



2. Face Learning



Point the HuskyLens at area with faces. When a face is detected, it will be automatically selected by white bounding box on the screen, with word "Face" above.

Point the “+” symbol at a face, short press the "learning button" to learn the face. If the same face is detected by HuskyLens, a blue box with words "Face: ID1" will be displayed on the screen, which indicates that HuskyLens has learned the face before and can recognize it now.

* Keep pressing the "learning button", a face from different angles can be learnt.

* If there is no “+” symbol in the center of the screen before learning, it means that the HuskyLens has already learned the face in the current function. At this time, short press the "learning button", the screen will display "click again to forget". Before the countdown ends, short press the "learning button" again to delete the learned face information.

Project practice:

How is HuskyLens face recognition used? How to implement obstacle avoidance running of Maqueen Plus? How to detect bad guy? Let's break down the whole project into several small tasks and complete the undercover detective task step by step.

This project will be completed in three steps. Firstly, learn to use HuskyLens face recognition function and display the corresponding face ID on the screen of the mainboard. Then learn how to implement obstacle avoidance running of Maqueen Plus. Finally, improve the whole project by allowing Maqueen Plus to run while avoiding obstacles and detecting bad guy. If bad guy is found, an alarm will be triggered.

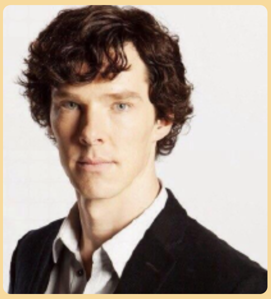
Task 1: Get to Know Face Recognition

1.Program Design

STEP 1 Learning and Recognition

Here you can choose a few face pictures at will and let HuskyLens learn. (Note: First switch to "learn multiple" function)

ID1

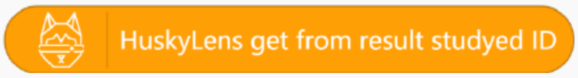

ID2

ID3

STEP 2 Instruction Learning

Let's take a look at some of the main instructions.

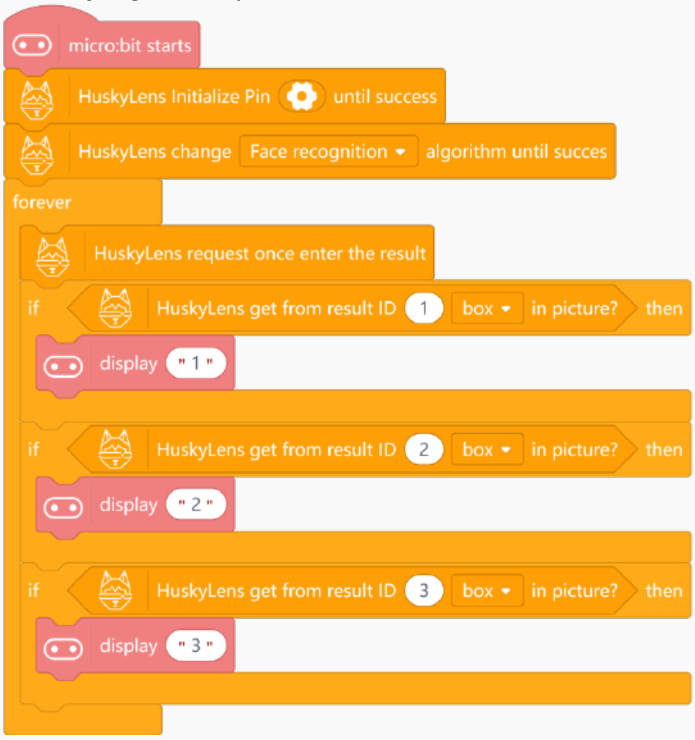
①Get how many targets have been learned under the current algorithm from the requested "result". Please note that whether to learn multiple targets can be set from HuskyLens by long press the function button to enter the parameter setting.



```
HuskyLens get from result studyed ID
```

STEP 3 Function Analysis

Assuming there are only 3 facial information, the program is easy to implement. As shown in the following picture, only need to judge one by one.

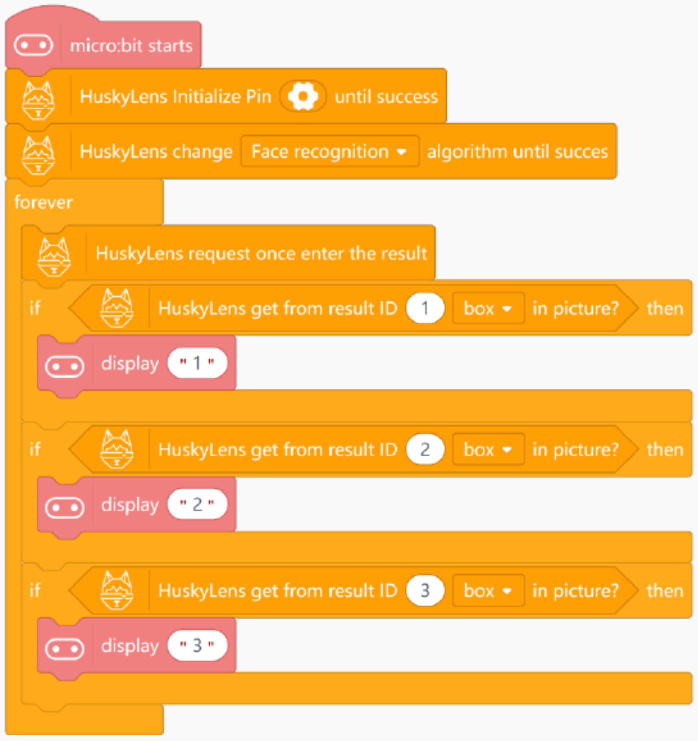


```
micro:bit starts
HuskyLens Initialize Pin ⚙ until success
HuskyLens change Face recognition ▾ algorithm until succes
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID 1 box ▾ in picture? then
        display "1"
    if HuskyLens get from result ID 2 box ▾ in picture? then
        display "2"
    if HuskyLens get from result ID 3 box ▾ in picture? then
        display "3"
```

However, when there are 5, 10 or more faces learned, how can they be implemented by programs? Judging one by one is rather wordy. Have you found any patterns from the above program?

STEP 3 Function Analysis

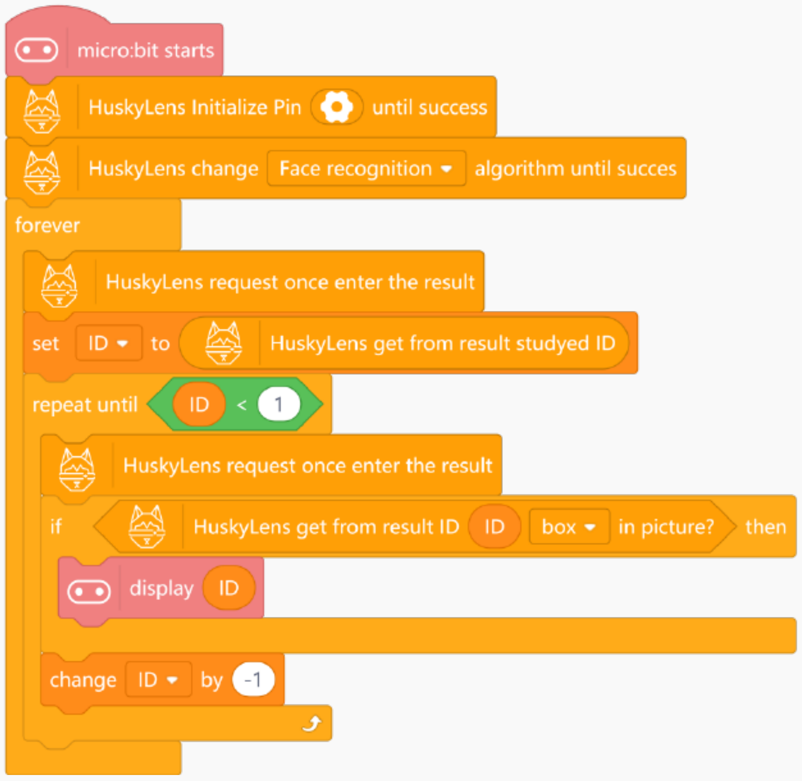
Assuming there are only 3 facial information, the program is easy to implement. As shown in the following picture, only need to judge one by one.



```
micro:bit starts
HuskyLens Initialize Pin [I2C] until success
HuskyLens change [Face recognition ▾] algorithm until succes
forever
    HuskyLens request once enter the result
    if <HuskyLens get from result ID (1) [box ▾] in picture?> then
        display "1"
    if <HuskyLens get from result ID (2) [box ▾] in picture?> then
        display "2"
    if <HuskyLens get from result ID (3) [box ▾] in picture?> then
        display "3"
```

However, when there are 5, 10 or more faces learned, how can they be implemented by programs? Judging one by one is rather wordy. Have you found any patterns from the above program?

In fact, as long as adding one variable, the parts with high repeatability in the above program can be simplified. The program is as follows, so that no matter how many IDs have been learned, they can be judged one by one.

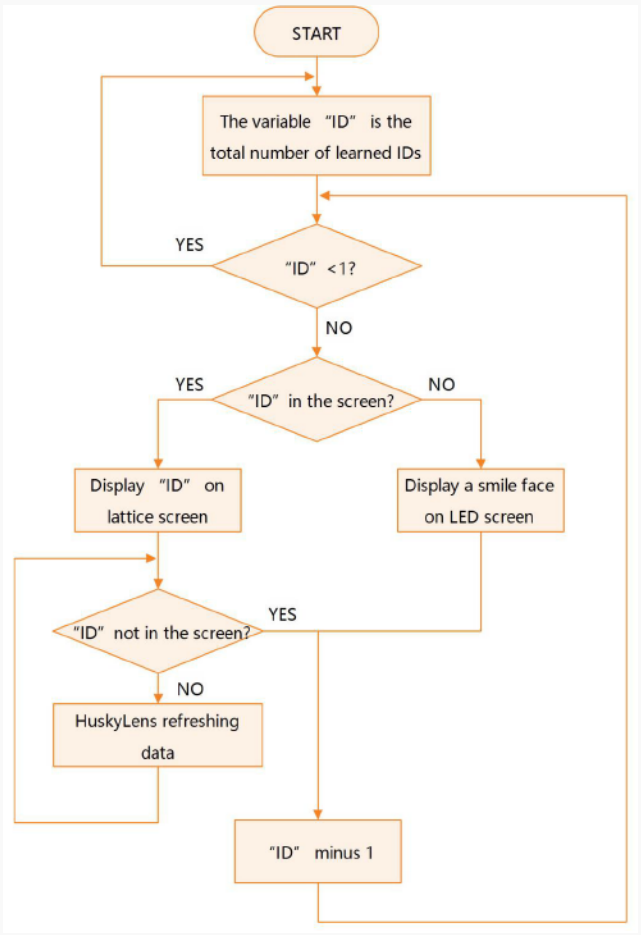


```
micro:bit starts
HuskyLens Initialize Pin [I2C] until success
HuskyLens change [Face recognition ▾] algorithm until succes
forever
    HuskyLens request once enter the result
    set [ID ▾] to (HuskyLens get from result studyed ID)
    repeat until <(ID) < (1)>
        HuskyLens request once enter the result
        if <HuskyLens get from result ID (ID) [box ▾] in picture?> then
            display (ID)
        change [ID ▾] by (-1)
```

STEP 4 Flow Chart Analysis

Optimize the function of the program, when there is no learned faces detected, the LED screen displays smiling face, and when there is a learned face, its ID is displayed.

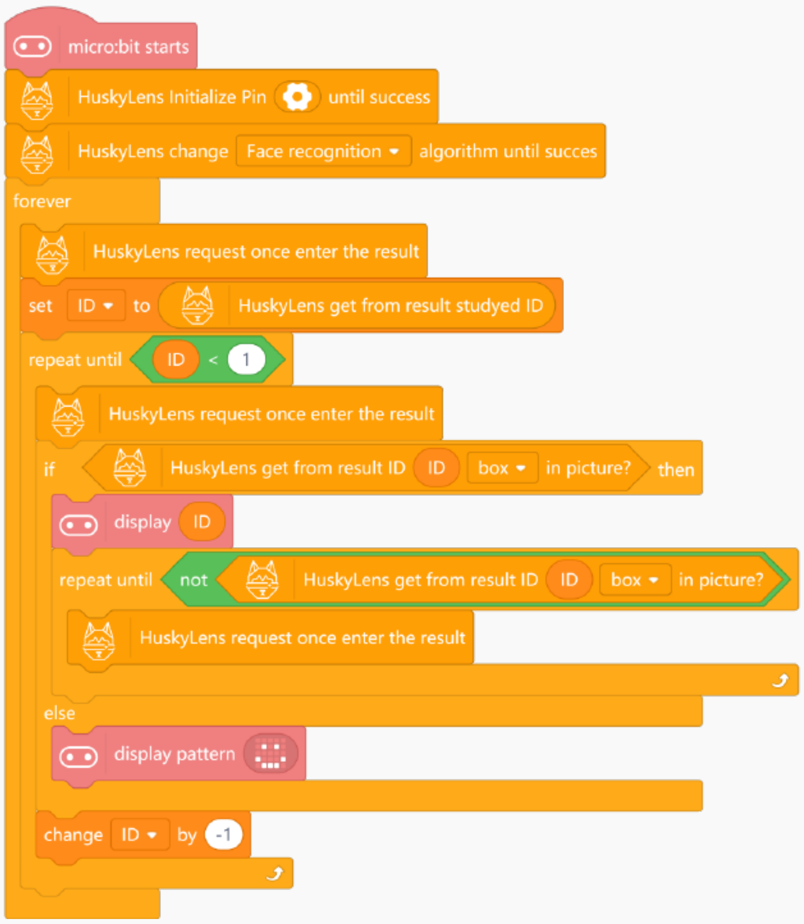
The analysis of the program flow chart is as follows. The logic may be a little bit complicated. You can use the flow chart to sort out the program logic, or you can use the program to understand the method of realizing the function of the flow chart.



START
The variable "ID" is the total number of learned IDs
"ID" <1?
YES
NO
"ID" in the screen?
YES
NO
Display "ID" on lattice screen
Display a smile face on LED screen
"ID" not in the screen?
YES
NO
HuskyLens refreshing data
"ID" minus 1

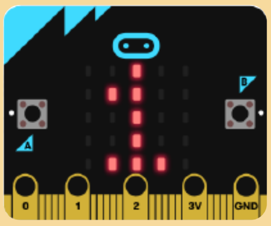
2.Program Example

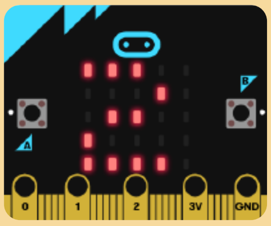
The program

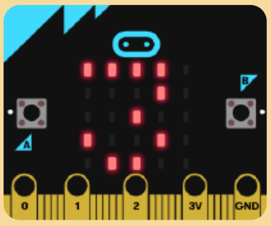


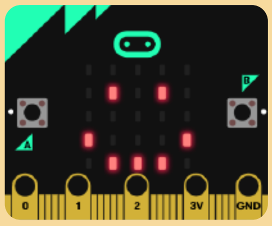
```
micro:bit starts
HuskyLens Initialize Pin ⚙ until success
HuskyLens change Face recognition ▾ algorithm until succes
forever
    HuskyLens request once enter the result
    set ID ▾ to HuskyLens get from result studyed ID
    repeat until ID < 1
        HuskyLens request once enter the result
        if HuskyLens get from result ID ID box ▾ in picture? then
            display ID
            repeat until not HuskyLens get from result ID ID box ▾ in picture?
                HuskyLens request once enter the result
        else
            display pattern
        change ID ▾ by -1
```

3.Execution Result


ID1 recognized


ID2 recognized


ID3 recognized


No ID in the screen

Task 2: Obstacles Avoidance Running

1. Program Design

STEP 1 Function Analysis

Maqueen Plus, as an undercover detective in folk, needs to have a basic function of independent driving. How to deal with obstacles? How to avoid obstacles automatically?

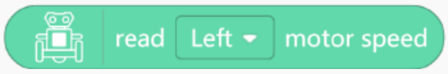
You might think of cameras to be used to avoid obstacles. In fact, the basic obstacle avoidance function doesn't need a camera at all, and the Maqueen Plus can handle it by itself. Let's have a look!

Maqueen Plus motor has its own encoder, with which the motor speed can be read in real time. When the car encounters obstacles during driving, the car is forced to stop and the motor speed is close to 0.Therefore, the motor speed can be read to determine whether it meets obstacles. If it meets obstacles, the car will implement some obstacle avoidance actions, such as retreating and then veering.

STEP 2 Instruction Learning

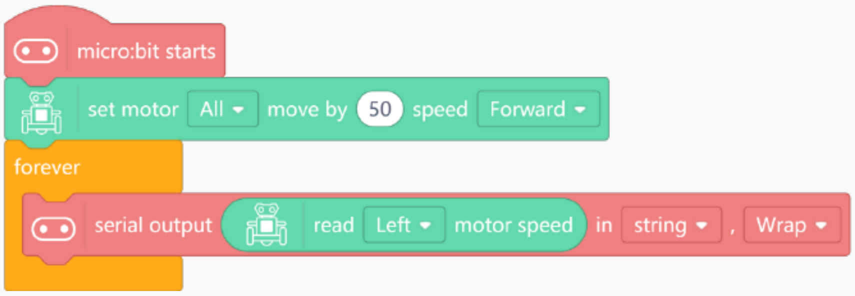
Let's take a look at some of the main instructions.

①Read the current motor speed and select the left or right motor.



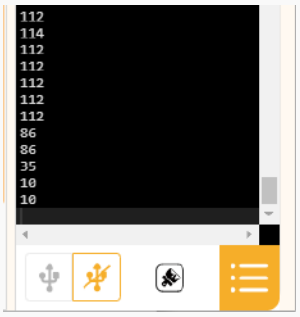
STEP 3 Test Program

Output motor speed through serial port. Take the left motor as an example. Test program:


micro:bit starts
set motor All move by 50 speed Forward
forever
serial output read Left motor speed in string , Wrap

Test results: The serial port area shows the speed of the left motor. Give the left wheel a little resistance by hand, and the motor speed can be obviously reduced.

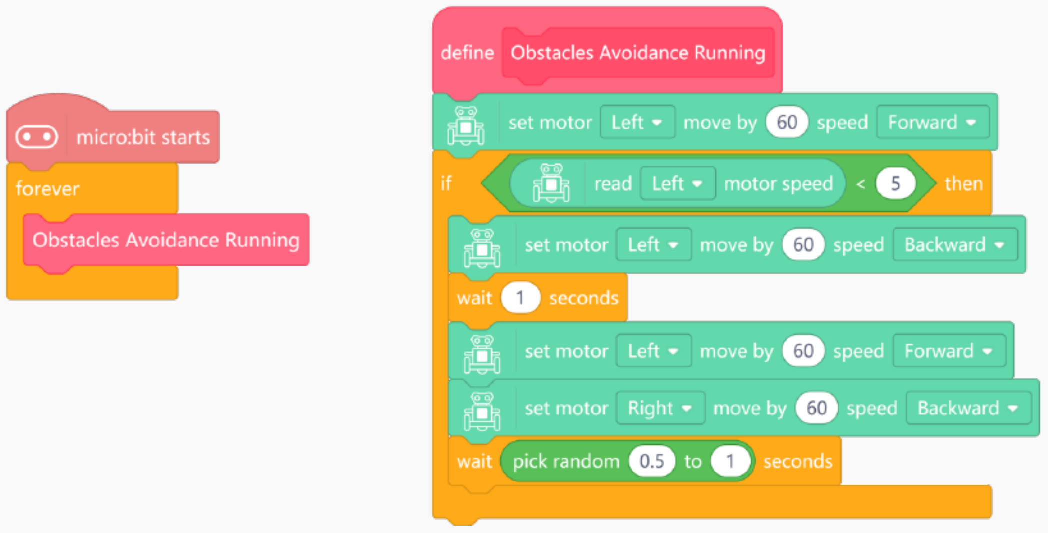
* when you look at the speed, you will find that the speed read by the motor has a certain deviation from the speed set by the program, which is related to the battery power and the friction force on the wheels. Do you still remember the PID control learned in the last project? In fact, PID is to adjust the speed in real time through the algorithm after reading the motor speed through the encoder, so that the motor speed is as close to the set speed as possible.



```
112
114
112
112
112
112
112
86
86
35
10
10
```

2. Program Examples

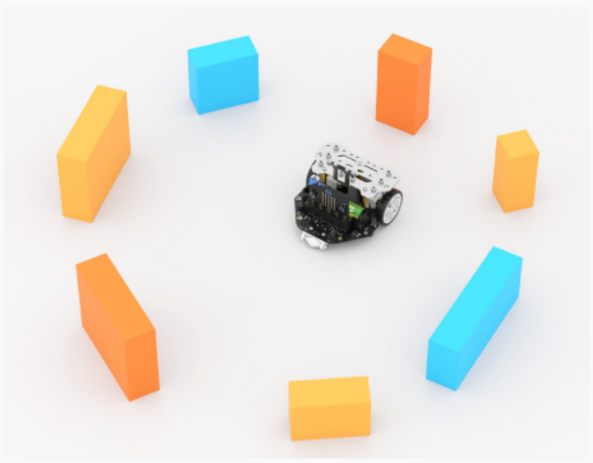
The program



```
micro:bit starts
forever
    Obstacles Avoidance Running

define Obstacles Avoidance Running
    set motor Left move by 60 speed Forward
    if read Left motor speed < 5 then
        set motor Left move by 60 speed Backward
        wait 1 seconds
        set motor Left move by 60 speed Forward
        set motor Right move by 60 speed Backward
        wait pick random 0.5 to 1 seconds
```

3. Execution Result



Maqueen Plus will automatically avoid obstacles and run. When encountering obstacles, Maqueen Plus will retreat and change direction to avoid obstacles.

* When the program is started, since the motor speed does not immediately increase, the obstacle avoidance action will be executed first. This only happened when it was just started.

Task 3: Alert if meeting with a bad person

1、程序设计

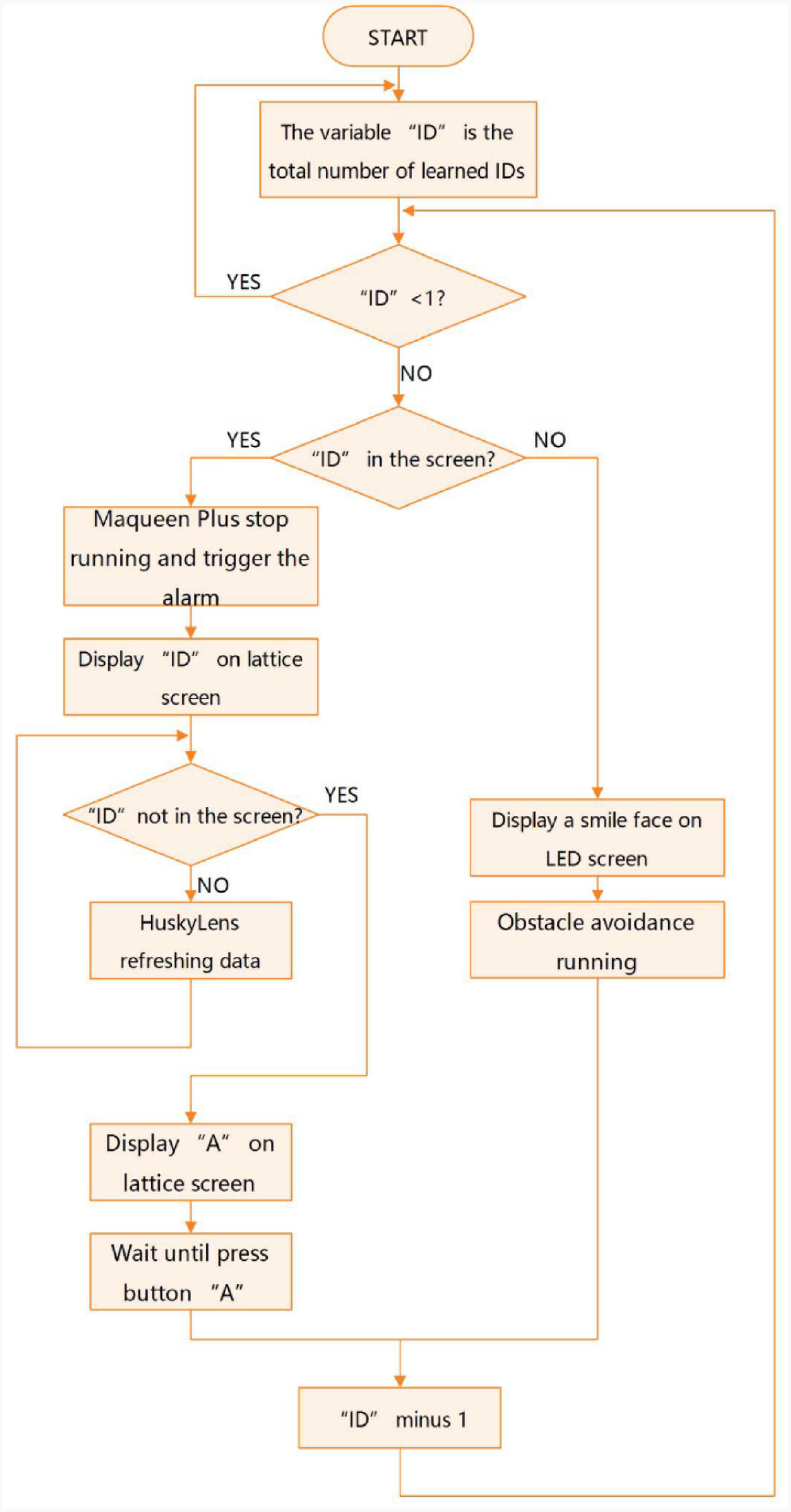
STEP 1 Function Analysis

In task 1 we implemented the function of face recognition, and in task 2 we implemented the function of obstacle avoidance. Combining the two functions and it is what we want for the undercover detective!

When HuskyLens did not recognize the designated face, Maqueen Plus kept avoiding obstacles. When the designated face is recognized, Maqueen Plus stops moving, triggers an alarm in situ, and displays the recognized face ID on the screen of the main board.

STEP 2 Flow Chart Analysis

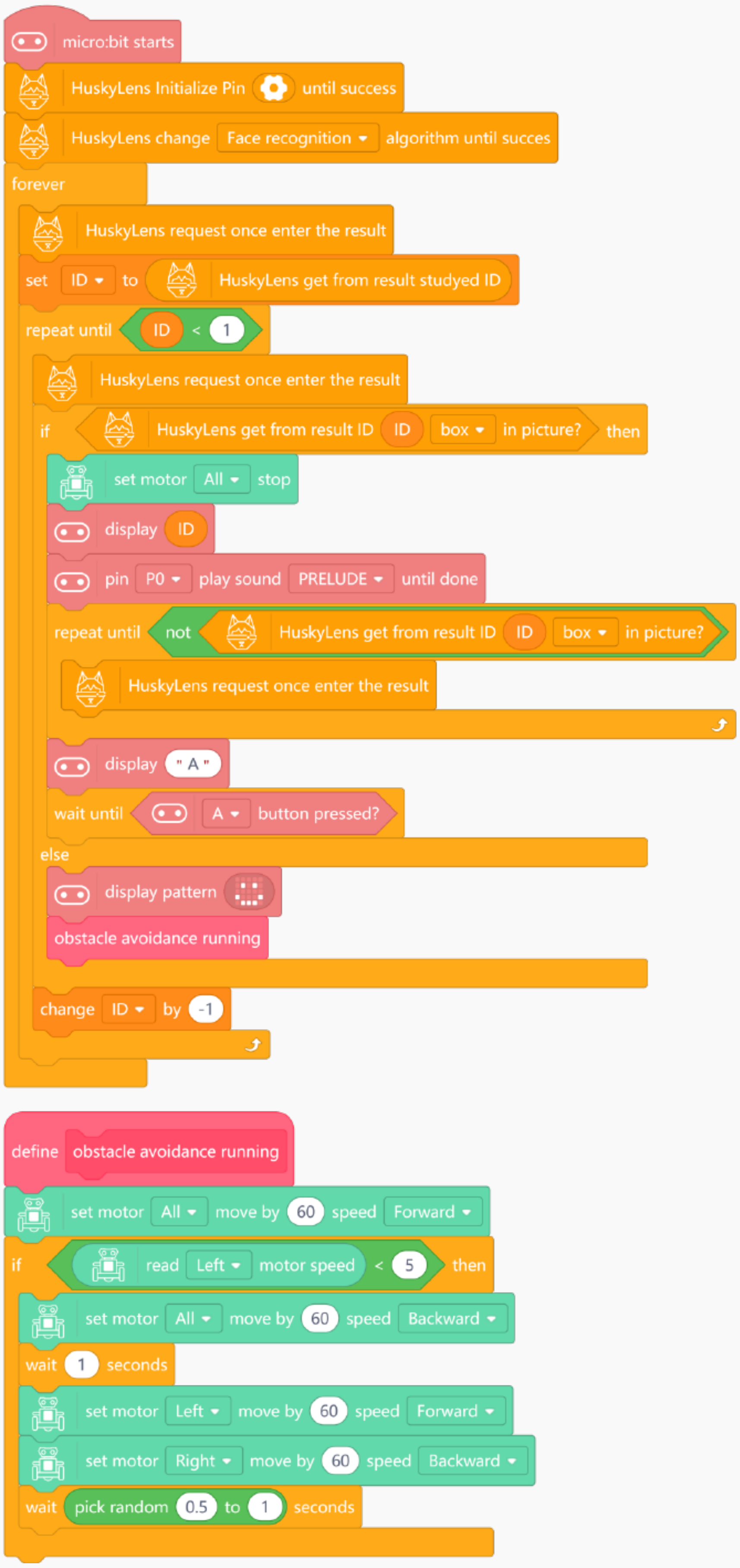
In order to keep the program running continuously, after the face is recognized, if the face is no longer in the picture, press the A key on the main control board to restart the program. The flow chart is as follows:


START
The variable "ID" is the total number of learned IDs
"ID" <1?
YES
NO
"ID" in the screen?
YES
NO
Maqueen Plus stop running and trigger the alarm
Display "ID" on lattice screen
"ID" not in the screen?
YES
NO
HuskyLens refreshing data
Display a smile face on LED screen
Obstacle avoidance running
Display "A" on lattice screen
Wait until press button "A"
"ID" minus 1

2. Program Examples

The program

The function "Avoid Obstacles" remains unchanged, and the main program is modified. The complete program is as follows.



```
micro:bit starts
HuskyLens Initialize Pin ⚙ until success
HuskyLens change Face recognition ▾ algorithm until succes
forever
  HuskyLens request once enter the result
  set ID ▾ to HuskyLens get from result studyed ID
  repeat until ID < 1
    HuskyLens request once enter the result
    if HuskyLens get from result ID ID box ▾ in picture? then
      set motor All ▾ stop
      display ID
      pin P0 ▾ play sound PRELUDE ▾ until done
      repeat until not HuskyLens get from result ID ID box ▾ in picture?
        HuskyLens request once enter the result
      display " A "
      wait until A ▾ button pressed?
    else
      display pattern
      obstacle avoidance running
    change ID ▾ by -1

define obstacle avoidance running
  set motor All ▾ move by 60 speed Forward ▾
  if read Left ▾ motor speed < 5 then
    set motor All ▾ move by 60 speed Backward ▾
    wait 1 seconds
    set motor Left ▾ move by 60 speed Forward ▾
    set motor Right ▾ move by 60 speed Backward ▾
    wait pick random 0.5 to 1 seconds
```

3. Execution Result

First, let HuskyLens learn multiple faces by face recognition function. When running the program, place Maqueen Plus at any position and the car will automatically avoid obstacles.

If HuskyLens sees the learned face during driving, the car stops moving, triggers an alarm, and the screen of the mainboard displays the corresponding face ID. After that, when there is no learned face detected by HuskyLens, the screen of the mainboard displays A, indicating that pressing the A button can restart Maqueen Plus and continuing to search for bad guys!

Project Summary:

Project Review

Understand the working principle of face recognition in this lesson, and learn the operation method of face recognition by using HuskyLens.

HuskyLens combining with Maqueen Plus motor encoder makes Maqueen Plus become an undercover detective, and can constantly recognize the collect facial information while executing obstacles avoidance driving.

Knowledge Nodes Recap

1.Understand the working principle of face recognition;

2.Learn the operation method of the face recognition function of HuskyLens;

3.Learn the function of Maqueen Plus motor encoder.

Project Extension:

In this project, our detective Maqueen Plus can find the bad guy hidden in the crowd, but can we automatically follow the bad guy after capturing them, making bad guys have nowhere to escape under AI face recognition system!

If you think tracing is difficult, we will talk about how to track them in the next project. Let's continue to learn together.

Project 5: Pokémon

Speaking of Pikachu, everyone is familiar with it. It is the first Pokémon of Ash Ketchum, the hero character of the animation "Pokémon". After being domesticated by Ash Ketchum, it has always accompanied him around and grown up with him.

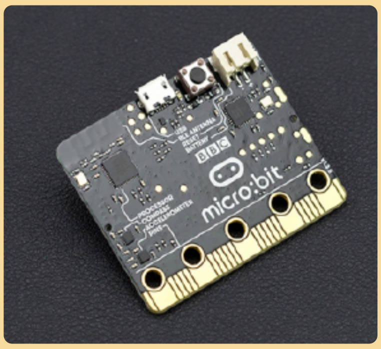
Such companionship is really enviable. Although there is no real Pokémon in our life, everyone wants to have such a Pokémon that can always accompany us.

Let's think about it, can we create a Pokémon by ourselves?

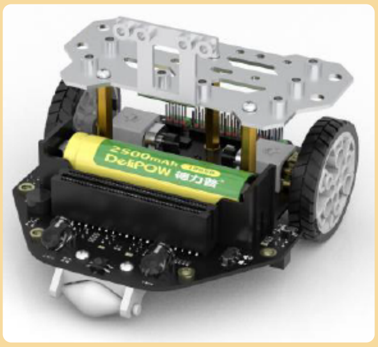
Function Description:

This project uses HuskyLens object tracking function to turn the Maqueen Plus into your exclusive Pokémon, which can move flexibly and always follow behind you.

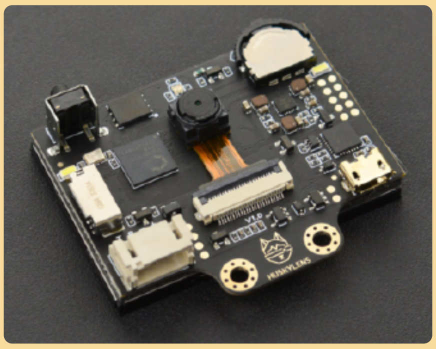
Materials Checklist:



Micro:bit ×1



Maqueen Plus ×1



HUSKYLENS ×1

Knowledge Extension:

When we need to track a moving object, we need to use visual object tracking technology in addition to manual operation. This technology is widely used in our life, such as video surveillance, drone follow shooting, etc. This project is to use HuskyLens object tracking function.

I. What is Object Tracking?

Object tracking is an important assignment in computer vision. It refers to the process of continuously inferring the state of objects in video sequences, simply speaking, to identify and trace specified objects.

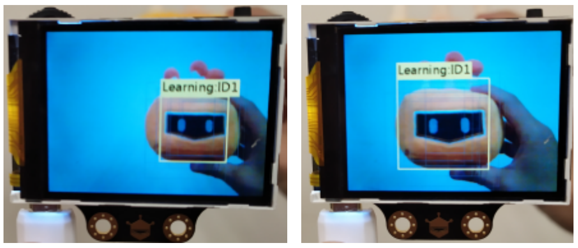
II. The Working Principle of Object Tracking

The image is collected by a single camera and the image information is transmitted to a computer. After analyzing and processing, the relative position of the moving object is calculated. At the same time, the camera is controlled to rotate to track the object in real time.

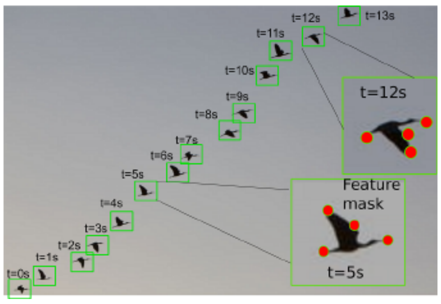
When the object tracking system performs the tracking function, it is mainly divided into four steps: object recognition, object tracking, object movement prediction and camera control.

①Object Recognition: Under the static background, accurate moving object information can be obtained through some image processing algorithms.

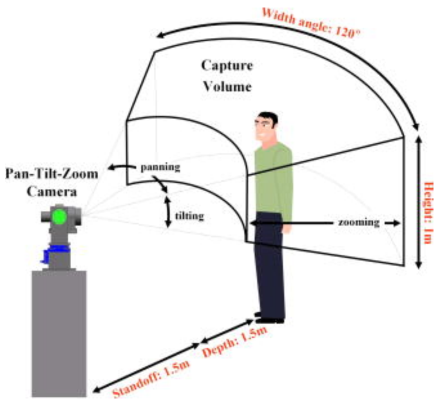
②Object Tracking: According to the position of moving object obtained in the previous step, the algorithm is used to track the following image sequence according to the color probability distribution, and further learning can be carried out in the subsequent tracking to make the tracking more and more accurate.



Object Movement Prediction: In order to improve the efficiency, the algorithm is used to calculate and predict the position of the moving object image in the next frame. The algorithm can be optimized and the efficiency can be improved.



Camera Control: Move the camera while collecting image information to adjust the direction of the camera along with the moving direction of the object, which generally requires to cooperate with PTZ or other motion modules.



III. Object Tracking Application Field

Intelligent video surveillance: based on motion recognition (human recognition, automated object detection, etc.), automated monitoring (monitoring a scene to detect suspicious behavior); traffic monitoring (collecting traffic data in real time to direct traffic flow).

Human-computer Interaction: traditional human-computer interaction is carried out through the computer keyboard and mouse. In order to enable the computer to have the ability to recognize and understand people's poses, actions and gestures, tracking technology is the key.

Virtual Reality: 3D interaction and virtual character motion simulation in virtual environment directly benefit from the research results of video human motion analysis, which can give participants more abundant forms of interaction, and human body tracking analysis is its key technology.

IV. Demonstration of HuskyLens Object Tracking Function

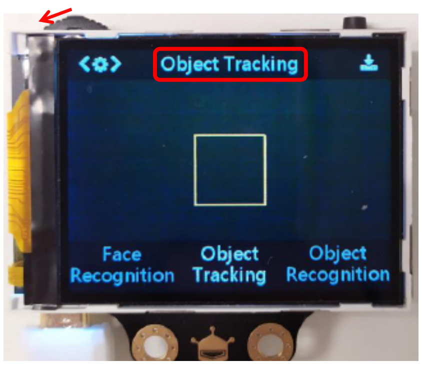
HuskyLens has built-in object tracking function, which can track objects by learning their features and feedback the position information to the mainboard.

Different from other functions such as color recognition or face recognition, object tracking can learn and recognize an object (or person) completely. Color recognition is only for colors, while face is only a part of human body. Object tracking is to learn and track the overall characteristics of this object.

The object tracking function can only track one object and does not support tracking multiple objects for the time being. It is better for the learned object to have obvious contours so that they can be easier to identify.

1. Select the "Object Tracking" Function

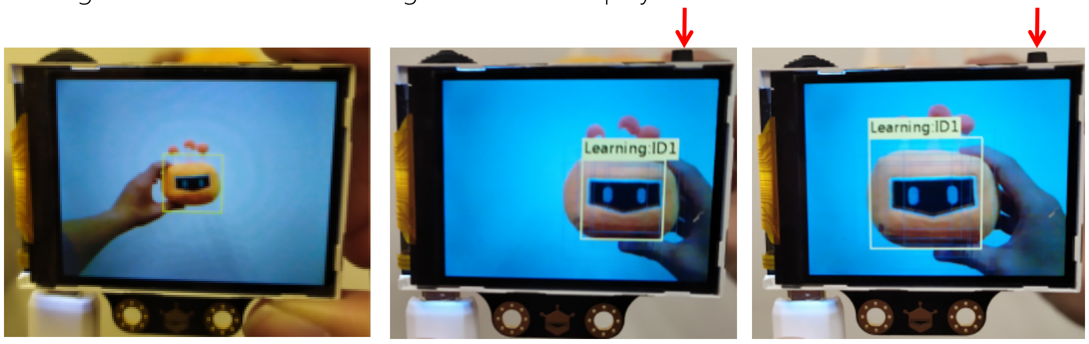
Dial the function button to the left or right until the word "Object Tracking" is displayed at the top of the screen.



2. Object Learning

Point HuskyLens to the target object, adjusting the distance and until the object is contained in the orange bounding box of the center of the screen. It is also acceptable that only part of the object included in the box but within distinct features.

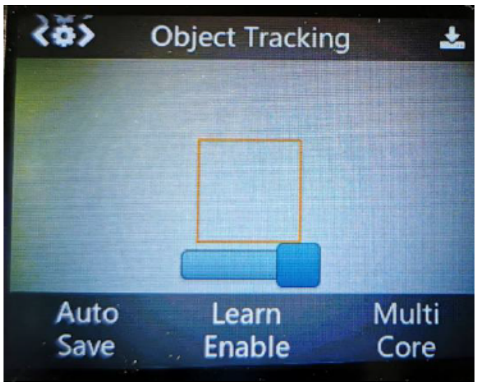
Then long press "learning button" to learn the object from various angles and distances. During the learning process, the orange box with words "Learning: ID1" will be displayed on the screen.



When HuskyLens can track the object at different angles and distances, then release the "learning button" to complete the learning.

* If there is no orange box on the center of the screen, it means that the HuskyLens has already learned an object. Please select "Forget Learned Object" and learn again.

3. Keep Learning



Under the object tracking function, HuskyLens can keep learning, that is, as long as the camera sees the learned object, it will keep learning the current state of the object, which is conducive to capturing dynamic object.

Operation method: Long press the function button to enter the parameter setting of the object tracking function. Dial the function button to the right to select "Learn Enable", then short press the function button, and dial it to the right to turn the "Learn Enable" ON, that is, the square icon on the progress bar is turned to the right. Then short press the function button to confirm this parameter.

4. Save the Model

When restarting HuskyLens, the last learned object is not saved by default, and you can turn on the switch to save models automatically.

Operation method: the same as above, after entering the parameter setting, switch "Auto Save" ON. In this way, you only need to learn the object once. Restarting the camera, the object you learned last time will be saved.

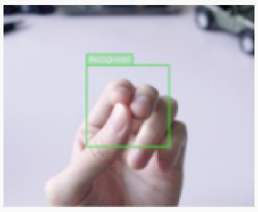
Project practice:

Task 1: Get to Know Object Tracking

Program Design

STEP 1 Learning and Recognition

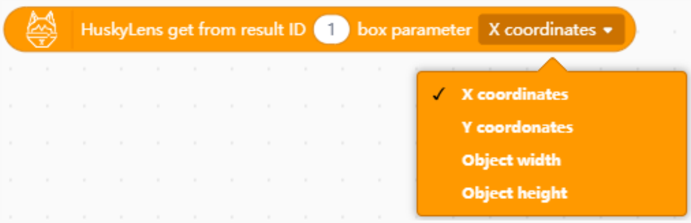
Here you can select an object with obvious contour features, such as gestures.



STEP 2 Instruction Learning

Let's take a look at some of the main instructions.

①The parameter of IDx is obtained from the requested "result", and -1 will be returned if this ID is not or has not been learned in the screen.

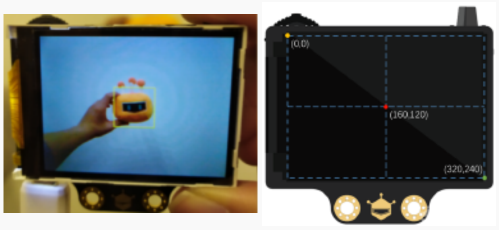


HuskyLens get from result ID 1 box parameter X coordinates ▾

✓ X coordinates
Y coordonates
Object width
Object height

STEP 3 Coordinates Analysis

HuskyLens sensor screen resolution is 320*240, as shown in the following picture. The coordinates of the object center point obtained through the program are also within this range. For example, if the coordinate values obtained are (160, 120), the object being tracked is in the center of the screen.



"X coordinates" and "Y coordinates" refer to the position of the box center point in the screen coordinate.

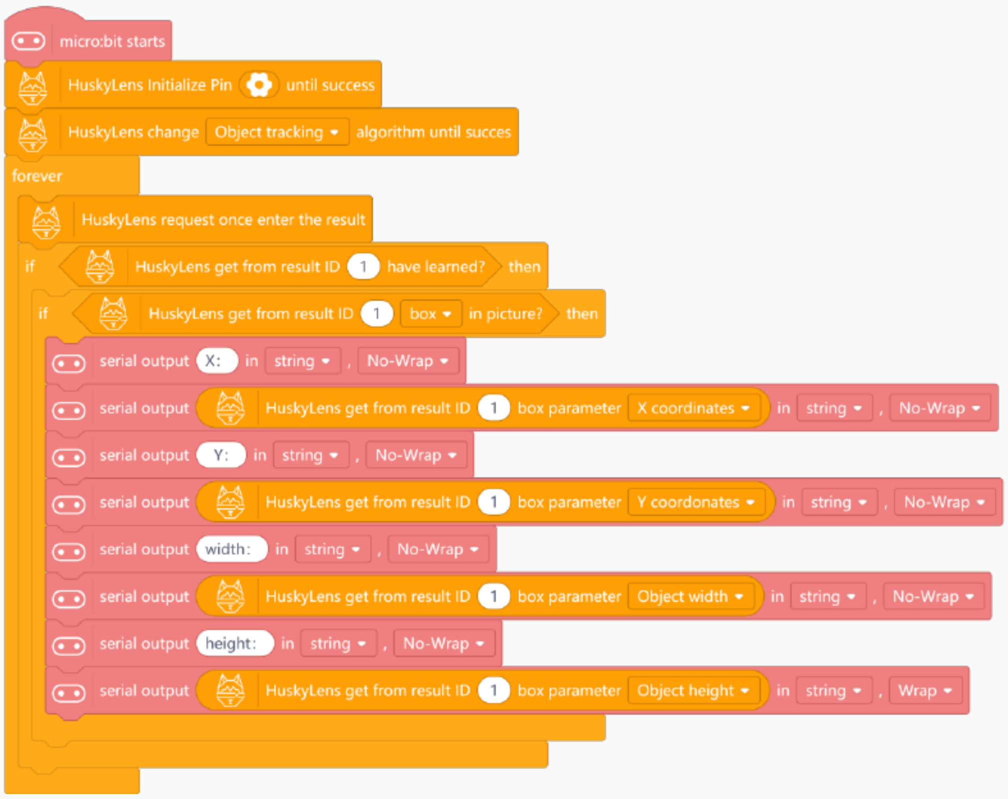
"Object width" and "Object height" refer to the size of the frame. Under the object tracking function, the frame is square, so the width and height are equal.

STEP 3 Function Analysis

Check each parameter of the frame through the serial port.

2. Program Example

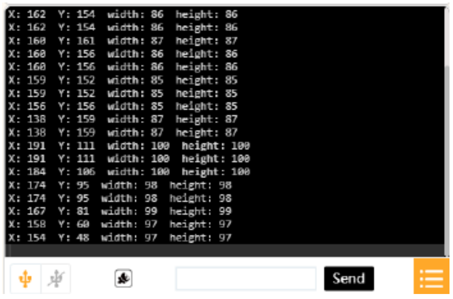
The program



```
micro:bit starts
HuskyLens Initialize Pin ⚙ until success
HuskyLens change Object tracking ▾ algorithm until succes
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID 1 have learned? then
        if HuskyLens get from result ID 1 box ▾ in picture? then
            serial output X: in string ▾ , No-Wrap ▾
            serial output HuskyLens get from result ID 1 box parameter X coordinates ▾ in string ▾ , No-Wrap ▾
            serial output Y: in string ▾ , No-Wrap ▾
            serial output HuskyLens get from result ID 1 box parameter Y coordonates ▾ in string ▾ , No-Wrap ▾
            serial output width: in string ▾ , No-Wrap ▾
            serial output HuskyLens get from result ID 1 box parameter Object width ▾ in string ▾ , No-Wrap ▾
            serial output height: in string ▾ , No-Wrap ▾
            serial output HuskyLens get from result ID 1 box parameter Object height ▾ in string ▾ , Wrap ▾
```

3. Execution Result

In the serial port area of Mind+, turn on the serial port switch to see the parameters. Try to move the object left and right to observe the numerical change of the X center. Move the object up and down to observe the numerical change of Y center. Move the object back and forth to observe the numerical change of width and height.



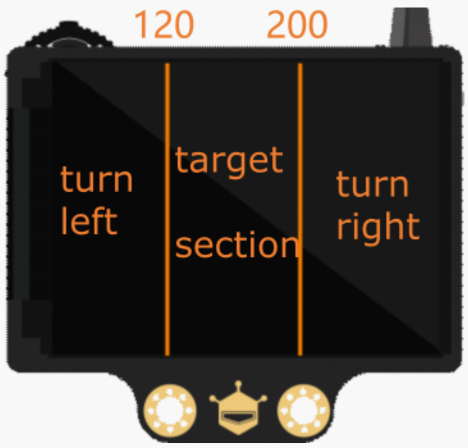
```
X: 162  Y: 154  width: 86  height: 86
X: 162  Y: 154  width: 86  height: 86
X: 160  Y: 161  width: 87  height: 87
X: 160  Y: 156  width: 86  height: 86
X: 160  Y: 156  width: 86  height: 86
X: 159  Y: 152  width: 85  height: 85
X: 159  Y: 152  width: 85  height: 85
X: 156  Y: 156  width: 85  height: 85
X: 138  Y: 159  width: 87  height: 87
X: 138  Y: 159  width: 87  height: 87
X: 191  Y: 111  width: 100  height: 100
X: 191  Y: 111  width: 100  height: 100
X: 184  Y: 106  width: 100  height: 100
X: 174  Y: 95  width: 98  height: 98
X: 174  Y: 95  width: 98  height: 98
X: 167  Y: 81  width: 99  height: 99
X: 158  Y: 60  width: 97  height: 97
X: 154  Y: 48  width: 97  height: 97
```

Task 2: Adjust Left and Right

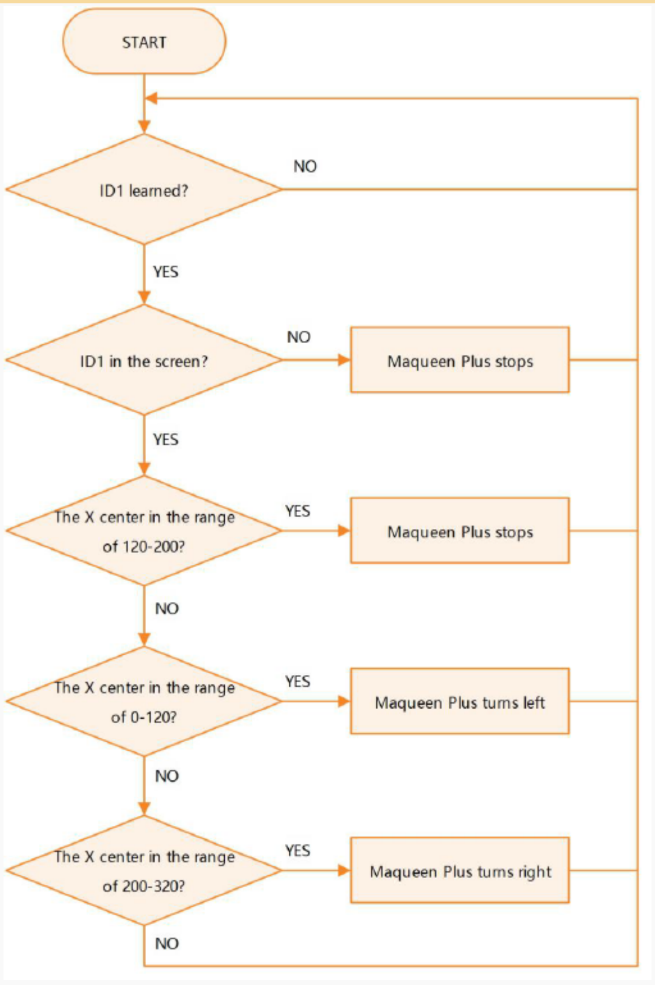
1. Program Design

STEP 1 Function Analysis

As shown in the following picture, the screen is divided into 3 sections according to the X axis of the camera screen coordinate system, and the middle section is our target section. When the camera continuously detects the state of the target object in the picture, its X center is 120-200, which means that the target is in the center of the field of vision and Maqueen Plus does not need to adjust its position; its X center is 0-120, Maqueen Plus need to adjust by turning right; its X center is 200-320, Maqueen Plus need to turn left to adjust.

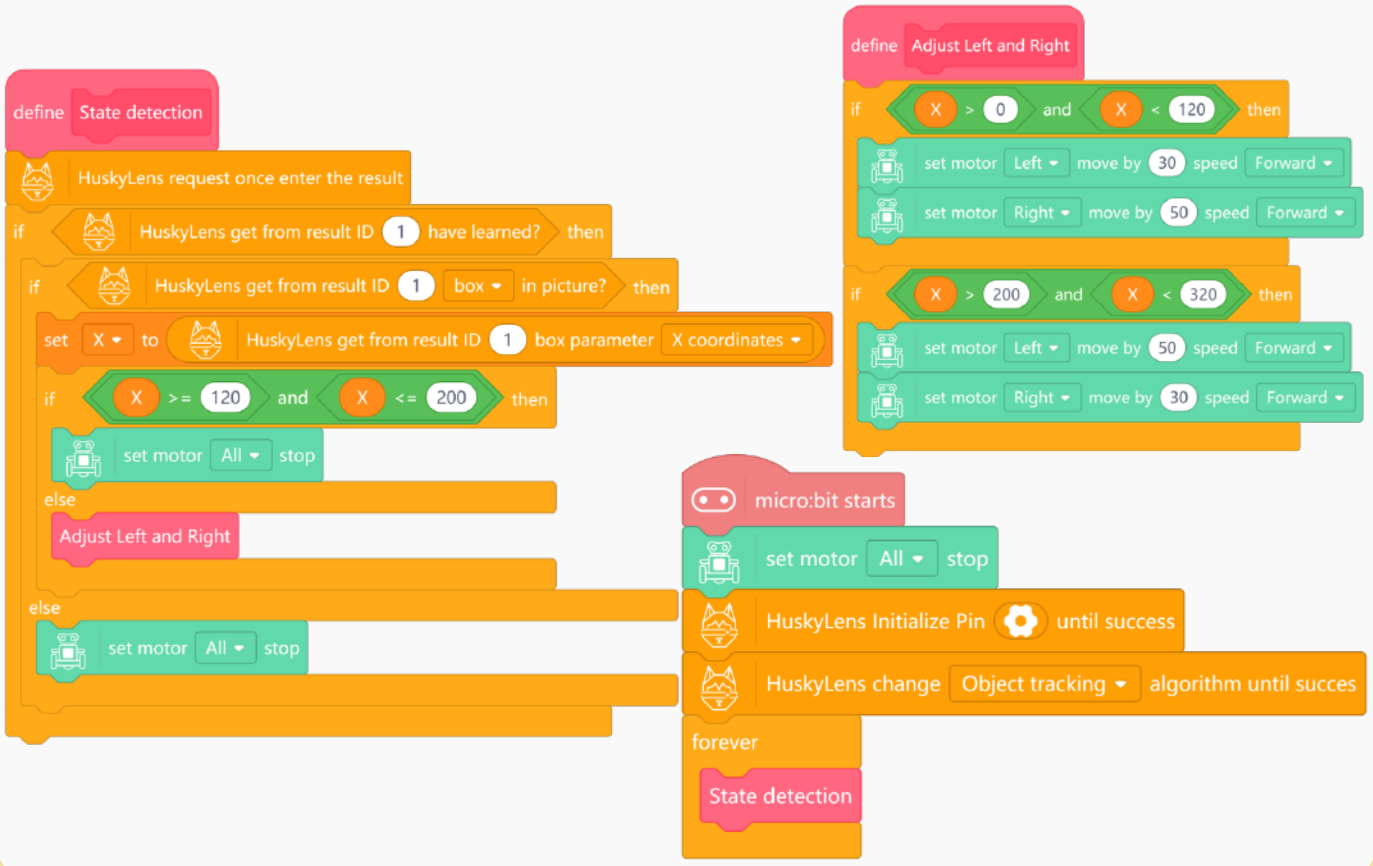


STEP 2 Flow Chart Analysis



2. Program Example

STEP 2 Flow Chart Analysis



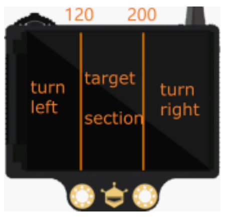
```
define State detection
HuskyLens request once enter the result
if HuskyLens get from result ID 1 have learned? then
    if HuskyLens get from result ID 1 box in picture? then
        set X to HuskyLens get from result ID 1 box parameter X coordinates
        if X >= 120 and X <= 200 then
            set motor All stop
        else
            Adjust Left and Right
    else
        set motor All stop

define Adjust Left and Right
if X > 0 and X < 120 then
    set motor Left move by 30 speed Forward
    set motor Right move by 50 speed Forward
if X > 200 and X < 320 then
    set motor Left move by 50 speed Forward
    set motor Right move by 30 speed Forward

micro:bit starts
set motor All stop
HuskyLens Initialize Pin until success
HuskyLens change Object tracking algorithm until succes
forever
    State detection
```

3. Execution Result

When the box of identified object is in the center of the screen, the car stops; when the box is on the left or right side of the screen, the car automatically adjusts the position left or right until the box is located in the target section of the screen.



Task 3: Adjust Back and Forth

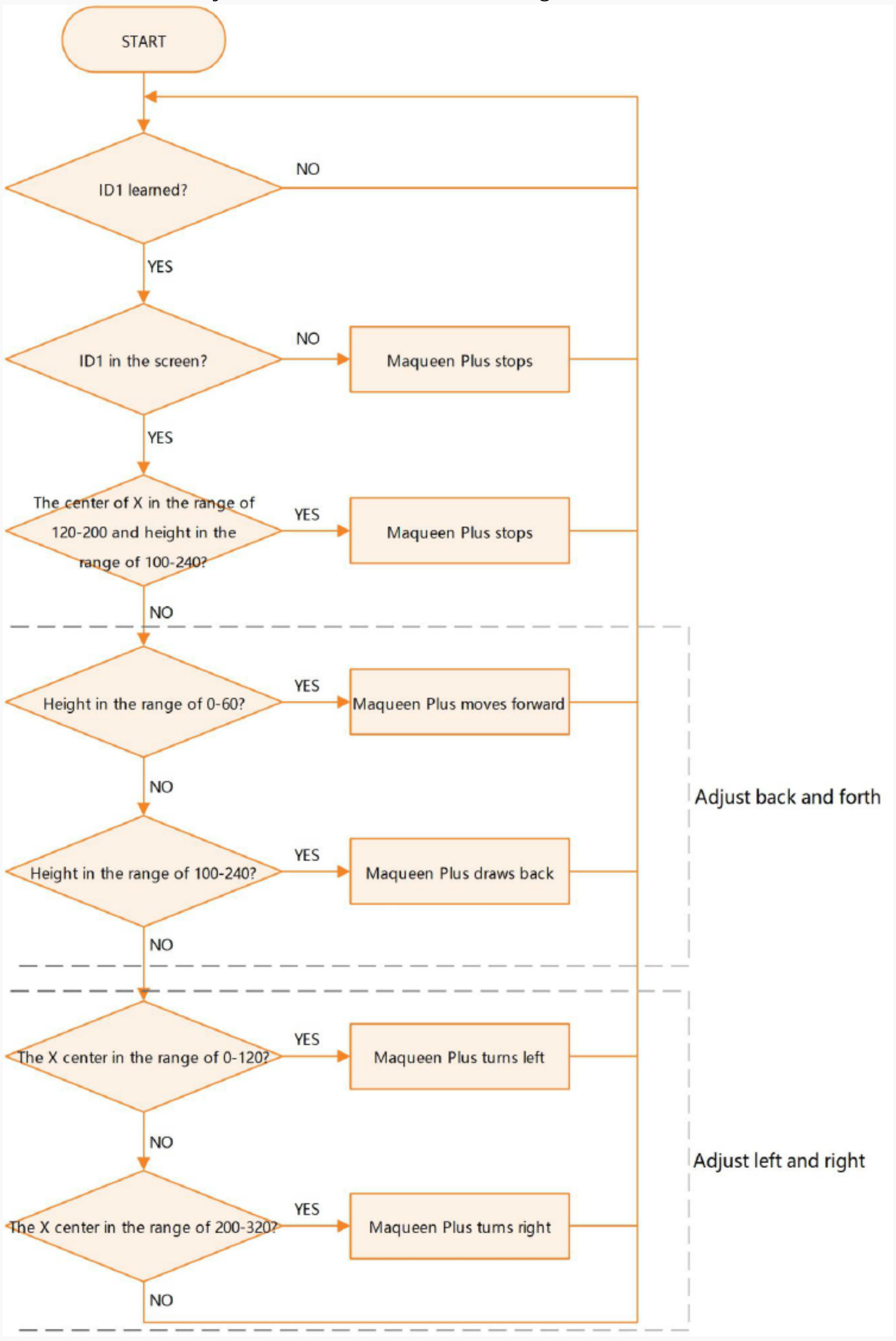
1. Program Design

STEP 1 Function Analysis

As the owner's Pokémon, it should also be able to follow the owner to move together. When it is relatively far away from the owner, it will automatically chase after. And when it is too close, it will retreat to a safe distance.

STEP 2 Flow Chart Analysis

Based on task 2, the distance between Maqueen Plus and the target object is judged by the height of the box, and the back and forth adjustments are made according to the results.

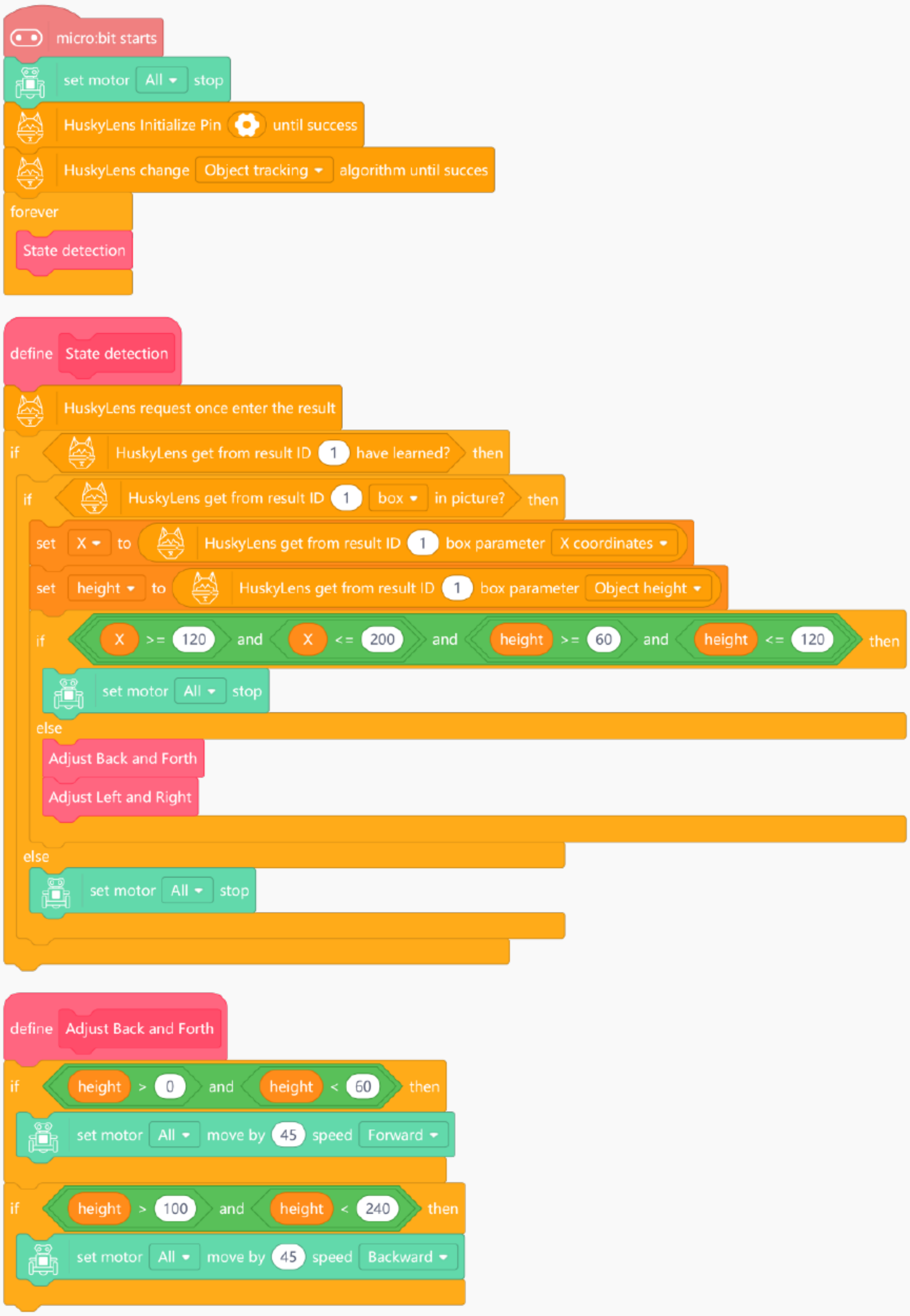


START
ID1 learned?
NO
YES
ID1 in the screen?
NO
Maqueen Plus stops
YES
The center of X in the range of 120-200 and height in the range of 100-240?
YES
Maqueen Plus stops
NO
Height in the range of 0-60?
YES
Maqueen Plus moves forward
NO
Height in the range of 100-240?
YES
Maqueen Plus draws back
NO
Adjust back and forth
The X center in the range of 0-120?
YES
Maqueen Plus turns left
NO
The X center in the range of 200-320?
YES
Maqueen Plus turns right
NO
Adjust left and right

2. Program Example

The program

Based on programs in task 2, the function "Adjust left and right" remains unchanged, and the program is modified. The complete program is as follows.



```
micro:bit starts
set motor All stop
HuskyLens Initialize Pin until success
HuskyLens change Object tracking algorithm until succes
forever
    State detection

define State detection
HuskyLens request once enter the result
if HuskyLens get from result ID 1 have learned? then
    if HuskyLens get from result ID 1 box in picture? then
        set X to HuskyLens get from result ID 1 box parameter X coordinates
        set height to HuskyLens get from result ID 1 box parameter Object height
        if X >= 120 and X <= 200 and height >= 60 and height <= 120 then
            set motor All stop
        else
            Adjust Back and Forth
            Adjust Left and Right
    else
        set motor All stop

define Adjust Back and Forth
if height > 0 and height < 60 then
    set motor All move by 45 speed Forward
if height > 100 and height < 240 then
    set motor All move by 45 speed Backward
```

3. Execution Result

After Maqueen Plus finish learning an object, it will automatically follow the object to move forward, backward, left and right, keeping the object box in the center of the screen and at a suitable distance.

Project Summary:

Project Review

Understand the working principle of object tracking and the operation method of face recognition? by using HuskyLens.

HuskyLens sensor can output the X center, Y center, height and width of the target box. These parameters can be used to judge the relative position and distance of the target object.

When the Maqueen Plus is used as a tracking car, these parameters can assist to locate the target object, which is still valid under other functions of the HuskyLens. For example, you can turn this project into a human face follower, and let Maqueen Plus follow the recognized the human face.

Knowledge Nodes Recap

1.Understand the working principle of object tracking;

2.Learn the operating method of HuskyLens object tracking function;

3.Learn to use HuskyLens to make Maqueen Plus follow the target.

Project Extension:

Just like the animation in which the hero character tames his Pokémon to make it stronger, can you make your Maqueen Plus have more functions? For example, add gesture control. When your hand draws a circle to the left, the car will circle to the left. When your hand pushes forward, the car will move forward. Can you implement it with programming?

Project 6: Following the "Right Track"

Nowadays robots are getting more and more powerful. They can do almost anything high on top of the mountain and deep into the sea. Look at them! Some of them serve well as waiters in restaurants. Some work hard as "couriers" in factory workshops. Some work as safety inspectors in power grid hubs with high sense of responsibility....

If you look closely, you will find that there are lines on the ground of where these robots work. And they are following the lines. Line tracking? How do they do it?

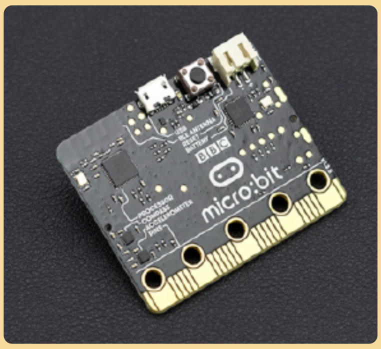
HuskyLens also has the function of line tracking. Imagine, if some guide lines are pasted at home, can Maqueen Plus become our life assistant like the food delivery robot in the robot restaurant?

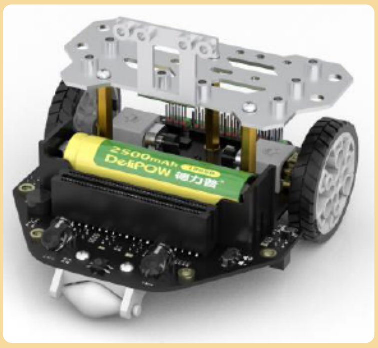
Let's explore by DIY a Maqueen Plus that follows the "right track"!

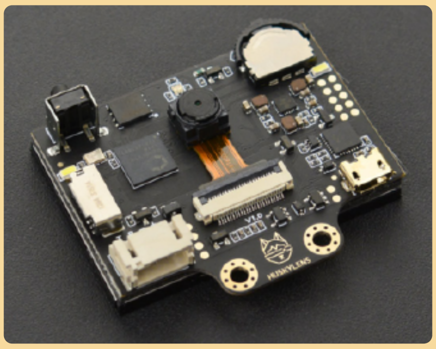
Function Description:

This project uses HuskyLens line tracking function to make Maqueen Plus follow the line track on the ground.

Materials Checklist:


Micro:bit ×1


Maqueen Plus ×1


HUSKYLENS ×1

Knowledge Extension:

If we want Maqueen Plus to move according to the lines on the ground, we need some sensors to identify these lines. According to the different sensors, there are several kinds of line tracking methods. This project uses the line tracking function of the visual sensor ——HuskyLens to implement the line tracking effect.

I. What is Line Tracking?

Line tracking refers to the process of object moving along a designated route. A fully functional line tracking robot uses a mobile robot as a carrier, a visible light camera, an infrared thermal imager and other detection instruments as a load system, a multi-field information fusion of machine vision, electromagnetic field, GPS and GIS as a navigation system for autonomous movement and tracking of the robot, and an embedded computer as a software and hardware development platform for the control system.

II. The Comparison of Two Commonly Used Line Tracking Methods

Type Comparison Items	Infrared Line Tracking Sensor	Visual Sensor
Cost	Low	High
Range of Vision	The sensor needs to be close to the ground and has a small range of view.	The range of vision is wide, and the movement state can be adjusted in advance according to the line changes.
Environment Adaptation	When the usage environment is changed, the sensitivity of the sensor needs to be adjusted, and the adjustment process is complicated.	When the use environment is changed, only the lines need to be relearned, and the operation is simple.
Map Adaptation	Generally only suitable for simple maps with clear background lines, black and white lines or solid lines	Suitable for maps with clear background lines, multi-color lines, solid lines, dotted lines and other complex conditions.

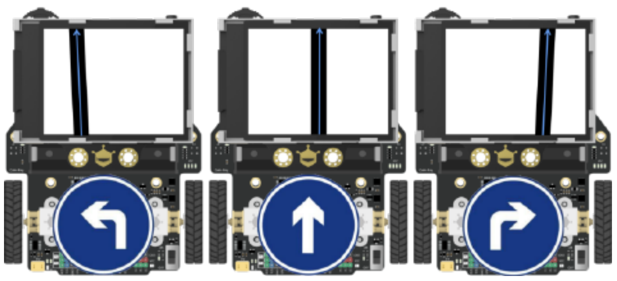
III. The Principle of Line Tracking Function

HuskyLens line tracking function is based on Pixy, an open source project of Carnegie Mellon University.

Pixy's algorithm can recognize the color of pictures. Its basic idea is to use the color space to remove the background that all users are not interested in and extract the foreground (such as lines).

Lines/Background, extract lines and color bars, remove background color

How can Maqueen Plus follow the black line on the tracking map (which has black lines on white ground)? In fact, we only need to know the relative position of Maqueen Plus to the black line. It can be divided into the following three situations:

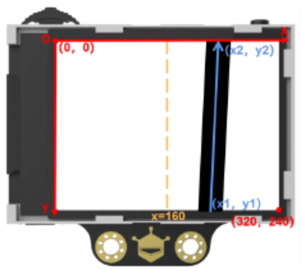


1. When Maqueen Plus is on the right side of the black line, control Maqueen Plus to turn left;

2. When Maqueen Plus is relatively center aligned with the black line, control Maqueen Plus to go straight;

3. When Maqueen Plus is on the left side of the black line, control Maqueen Plus to turn right.v

How should this be implemented? We stripped out the information displayed on the screen by HuskyLens during the line tracking and abstracted it into the geometric mathematical model shown in the picture below.



The resolution of HuskyLens screen is 320×240. The O point in the upper left corner of the screen is the origin of the screen's coordinates (0, 0), the horizontal right direction is the positive direction of the X axis, and the vertical down direction is the positive direction of the Y axis, so the coordinates in the lower right corner of the screen are (320, 240).The dotted orange line in the above picture is the central axis of the screen, and the abscissa value of this line is 160.The black line in the screen above is the map line "seen" by HuskyLens camera. The blue arrow is the line direction calculated by HuskyLens. The starting point coordinates of the blue arrow are (x1, y1) and the ending point coordinates are (x2, y2).

To sum up, we only need to judge the position of the starting point of the blue arrow relative to the central axis to implement line tracking.

IV. Demonstration of HuskyLens Line Tracking Function

1. Object Learning

This function can track lines of specified colors and make path prediction. The default setting is to track lines of only one color. This project will be explained by taking line tracking of one color as an example.

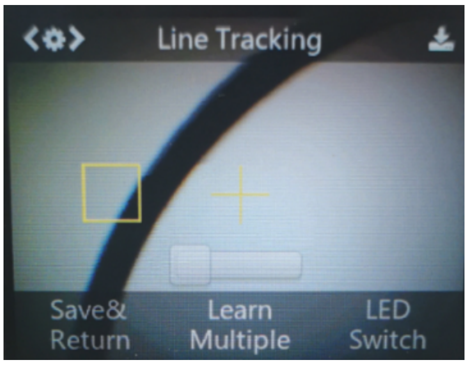
Operation and Setting:

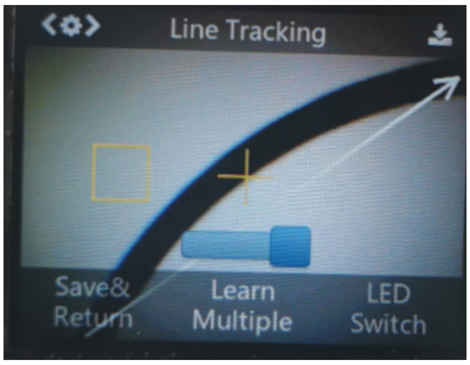
① Dial the function button to the left or right until the word "Line Tracking" is displayed at the top of the screen.

② Long press the function button to enter the parameter setting of the line tracking function.

③ Dial the function button right or left until "Learn Multiple" is selected, then short press the function button, and dial it to the left to turn off the "Learn Multiple" switch, that is, the square icon on the progress bar is turned to the left. Then short press the function button to complete this parameter.

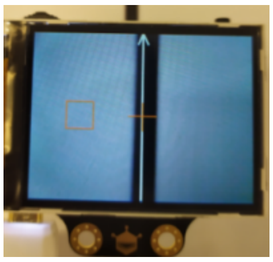
④ You can also turn on the LED by setting "LED Switch". This is very useful in the dark environment. Referring to the above method, turn on the "LED Switch".



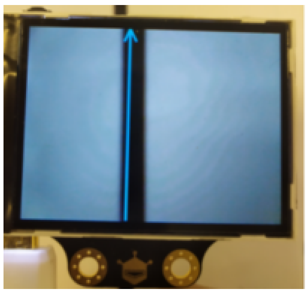
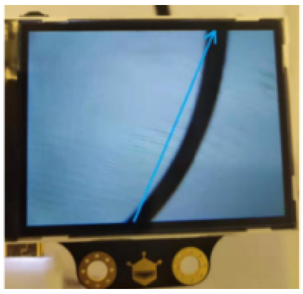


⑤ Dial the function button to the left until "Save & Return" is selected, and short press the function button to save the parameters and it will return automatically.

2. Learning and Tracking



Line Learning: Point the "+" symbol at the line, then point the orange box at the background area. It is recommended that within the view field of HuskyLens, just remain lines to learn and no any cross lines. Try to keep HuskyLens parallel to the target line, then HuskyLens will automatically detect the line and a white arrow will appear. Then short press the "learning button" and the white arrow turns into a blue arrow.

Line Learning: Point the "+" symbol at the line, then point the orange box at the background area. It is recommended that within the view field of HuskyLens, just remain lines to learn and no any cross lines. Try to keep HuskyLens parallel to the target line, then HuskyLens will automatically detect the line and a white arrow will appear. Then short press the "learning button" and the white arrow turns into a blue arrow.

* 1. When learning the line, we need to adjust the position of HuskyLens to be parallel to the line.

* 2. HuskyLens can learn line in any color that have an obvious color contrast to background, but these lines must be monochrome lines with obvious color contrast from background to keep line tracking stable.

* 3. HuskyLens can learn and track multiple lines according to the different colors of the lines, but the lines must be monochromatic lines with obvious color contrast from the background. In most cases, only use line in one color during line tracking. Therefore, in order to ensure stability, we recommend to track the line in single color.

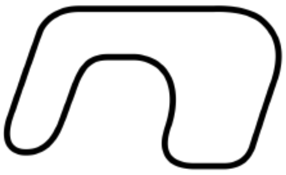
* 4. The color of the lines has a lot to do with the ambient light. When patrolling the line, please try to keep the ambient light as stable as possible.

Project practice:

We will implement the project according to the line tracking logic and continuously optimize the line tracking effect in several steps so that Maqueen Plus can pass the map quickly and stably. First, we will learn to use HuskyLens line tracking function, read the abscissa data of lines, write a simple program to adjust the motion state to meet the line tracking requirements, and then improve our project program according to the debugging result.

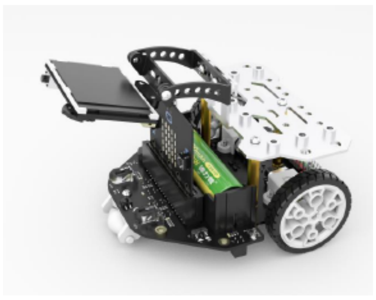
Task 1: Line Tracking Algorithm 1

The following picture is a reference line tracking map.



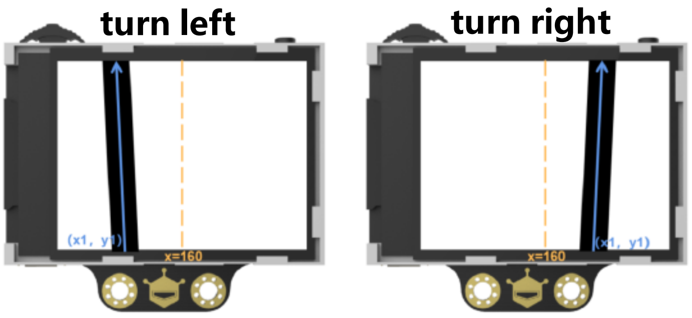
1.Structure Construction

Using screws to fix HuskyLens and Maqueen Plus, it should be noted that we need to adjust the camera obliquely downward for line tracking, so that Maqueen Plus can see the black line closer and the tracking effect would be better.

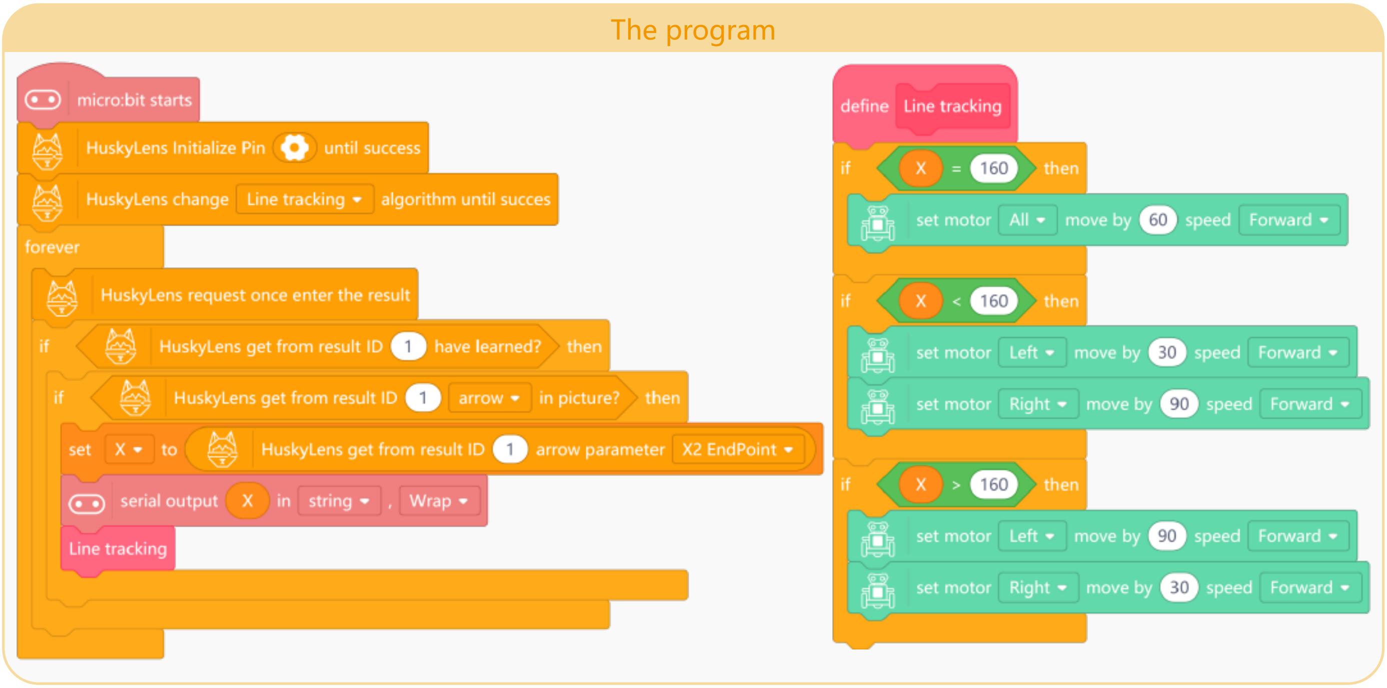


2.Program Design

When HuskyLens detects that the black line is on the left side of the screen, i.e. the abscissa value x1<160 of the starting point of the blue arrow, control Maqueen Plus to turn left; When the black line is on the right side of the screen, x1>160, control Maqueen Plus to turn right; When the black line is in the middle of the screen, x1=160, control Maqueen Plus to go straight.



3.Program Example



The program

```
micro:bit starts
HuskyLens Initialize Pin ✿ until success
HuskyLens change Line tracking ▾ algorithm until succes
forever
    HuskyLens request once enter the result
    if HuskyLens get from result ID 1 have learned? then
        if HuskyLens get from result ID 1 arrow ▾ in picture? then
            set X ▾ to HuskyLens get from result ID 1 arrow parameter X2 EndPoint ▾
            serial output X in string ▾ , Wrap ▾
            Line tracking

define Line tracking
if X = 160 then
    set motor All ▾ move by 60 speed Forward ▾
if X < 160 then
    set motor Left ▾ move by 30 speed Forward ▾
    set motor Right ▾ move by 90 speed Forward ▾
if X > 160 then
    set motor Left ▾ move by 90 speed Forward ▾
    set motor Right ▾ move by 30 speed Forward ▾
```

3. Execution Result

HuskyLens learns bottle, bicycle and chair under object recognition function. When the program is running, place Maqueen Plus at the starting point. When HuskyLens recognizes the bottle, Maqueen Plus will track the line to the terminal point 1.When HuskyLens recognizes the bicycle, Maqueen Plus will track the line to the terminal point 2.When HuskyLens recognizes the chair, Maqueen Plus will track the line to the terminal point 3.

Project Summary:

Project Review

Understand the working principle of object recognition in this lesson and the operation method of object recognition by using HuskyLens AI sensor.

HuskyLens combines with Maqueen Plus built-in line tracking sensor, transforming Maqueen Plus into an AI sorting master which is capable of automated sorting and transportation.

Imagine if Maqueen Plus could load or unload cargo with a mechanical arm, the whole application would be more intelligent.

Knowledge Nodes Recap

1.Understand the working principle of object recognition;

2.Learn the operating method of object recognition function of HuskyLens;

3.Learn the line-tracking control of Maqueen Plus.

Project Extension:

In this project, we implemented the function of sorting and transportation, but how can we get back to the starting point after Maqueen Plus is transported to the designated location? Can Maqueen Plus continue to track the line and come back? Try to implement it with a program.

4.Execution Result

Under the HuskyLens line tracking function, learn the line and background, and line tracking function will work as long as the camera downward.

Maqueen Plus can complete the basic line tracking tasks, but at the same time it also exposes the following problems:

(1) Maqueen Plus obviously sways left and right while moving forward, the speed changes are not consistent, and the straight movement isn't stable;

(2) The speed cannot be set too fast or it is easy to offtrack at the corner;

(3) Different bends require different turning speeds. When there are several bends in the tracking map, it is easy to offtrack.

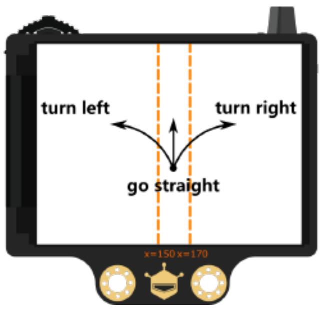
* If Maqueen Plus depart from the line during the turn, it needs to adjust the speed of the left and right motors continuously.

How to optimize it? Let's have a look!

Task 2: Line Tracking Algorithm 2

1.Program Design

In Task 1, Maqueen Plus sways left and right in its forward movement and cannot steadily move straight. Why is this happened? Because the horizontal axis domain of Maqueen Plus straight moving is only a line, and there is inertia in the movement process. So keep straight moving at x=160 is difficult to achieve.

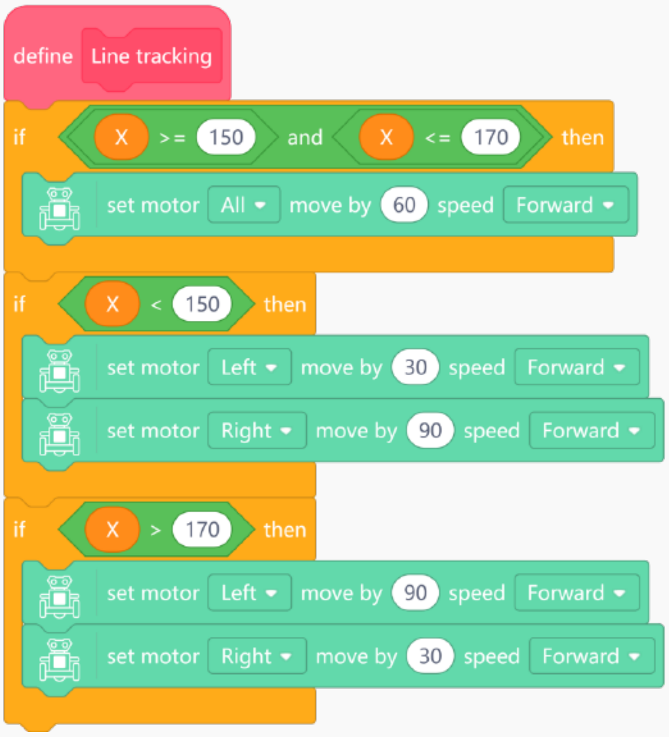


The optimization idea is to expand the straight movement domain. As shown in the following picture, we set the domain [150,170] as the Maqueen Plus straight movement domain. When the starting point coordinate x1 is within this domain, control the Maqueen Plus to go straight. When x1<150, control Maqueen Plus to turn left; When x1>170, control Maqueen Plus to turn right.

2.Program Example

The program

On the basis of program of task 1, the main program is unchanged, and the function "line tracking" is modified as follows.



```
define Line tracking
if X >= 150 and X <= 170 then
    set motor All move by 60 speed Forward
if X < 150 then
    set motor Left move by 30 speed Forward
    set motor Right move by 90 speed Forward
if X > 170 then
    set motor Left move by 90 speed Forward
    set motor Right move by 30 speed Forward
```

3.Execution Result

Through these adjustments, we find that the line tracking effect is better, and when it encounters an arc, it can almost stay on line. However, it is still easy to offtrack in tracking maps with large changes of bend.

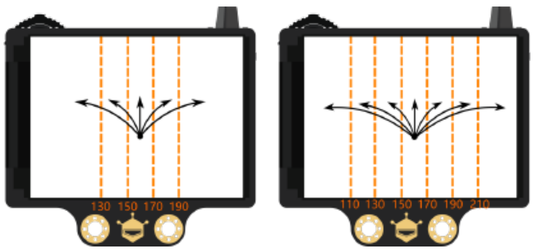
Task 3: Line Tracking Algorithm 3

Try to prevent the car from offtrack when entering the bend.

1.Program Design

Let's continue to adjust. Since we can divide the detected black line into three domains, can we continue to divide the domains into five as we did?

The closer the black line is to extreme sides, the faster the turning speed is. The closer the black line is to the middle, the more it tends to move straight.

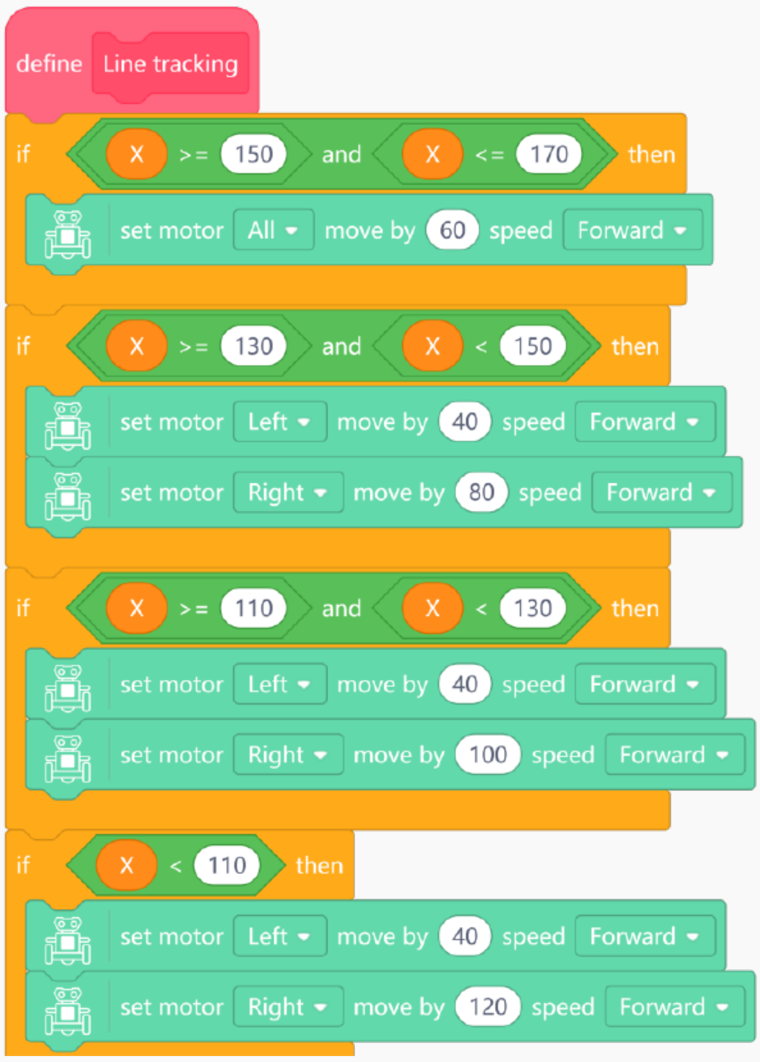


Or further, can it be directly divided into 7 speed regulation domains?

2.Program Example

The program

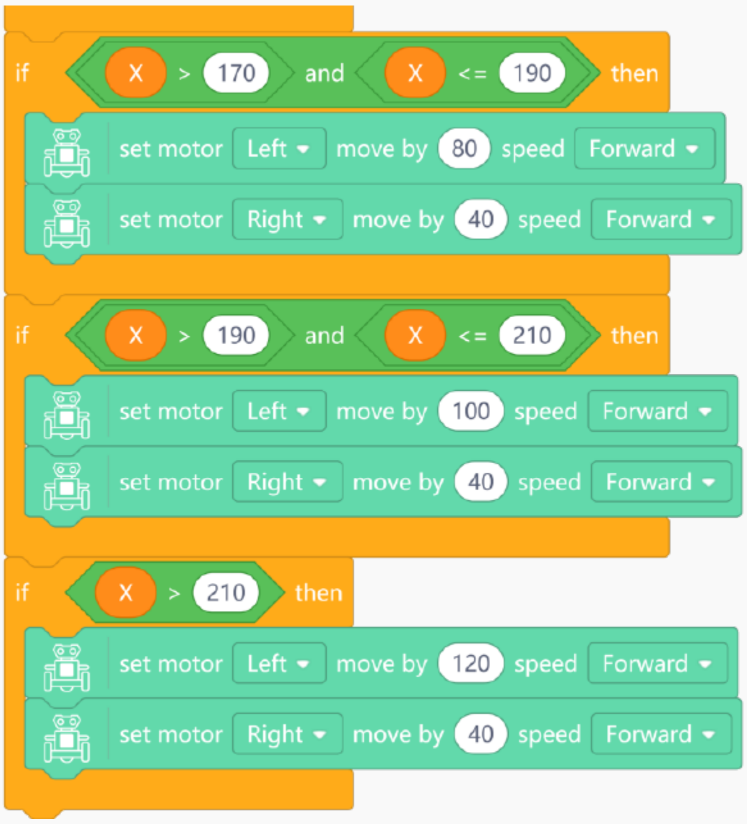
On the basis of the task 2 program, the main program remains unchanged and the function "line tracking" is modified as follows.



```
define Line tracking
if X >= 150 and X <= 170 then
    set motor All move by 60 speed Forward
if X >= 130 and X < 150 then
    set motor Left move by 40 speed Forward
    set motor Right move by 80 speed Forward
if X >= 110 and X < 130 then
    set motor Left move by 40 speed Forward
    set motor Right move by 100 speed Forward
if X < 110 then
    set motor Left move by 40 speed Forward
    set motor Right move by 120 speed Forward
```

The next page

The program



```
if X > 170 and X <= 190 then
    set motor Left move by 80 speed Forward
    set motor Right move by 40 speed Forward
if X > 190 and X <= 210 then
    set motor Left move by 100 speed Forward
    set motor Right move by 40 speed Forward
if X > 210 then
    set motor Left move by 120 speed Forward
    set motor Right move by 40 speed Forward
```

3.Execution Result

The line tracking speed of Maqueen Plus now can be faster, and the speed change is much smoother no matter turning or moving straight.

Project Summary:

Project Review

In this project, we implement the line tracking algorithm from simple to complex, making the effect of line tracking faster and more stable.

Knowledge Nodes Recap

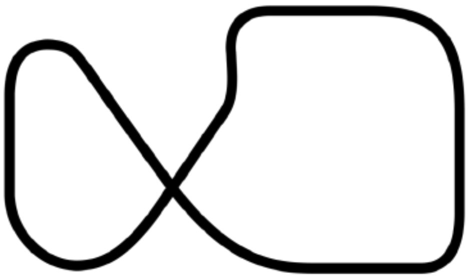
1.Learn the main ideas of line tracking

2.Learn the operation method of HuskyLens line tracking function;

3.Optimization Method of Line Tracking Algorithm

Project Extension:

After completing the line tracking without crossing, it may be necessary to handle the crossing line tracking as shown in the following picture. Can the tag recognition be used here to identify and let Maqueen Plus choose the correct path at the intersection?



Knowledge Expansion:

The motion status or line tracking domains of Maqueen Plus can be divided into 2 gears, 3 gears, 5 gears and 7 gears during line tracking. Is it still possible to further subdivide? 9 gears?11 gears?......

until the infinite subdivision. The more subdivided the speed regulation domain is, the better the line tracking effect is. But programs will get longer and longer. Any solution?

The PID speed regulation algorithm can help us solve this problem. PID stands for portion, integral and differential control. It is a closed-loop control system. Closed-loop control is a control method that corrects according to the feedback from the controlled object. It can correct according to a certain standard according to the Error between the measured actual value and the planned value.

If you are interested in it, you can search for relevant information on the internet and continue to optimize our line tracking effect.