⚠️
This tutorial is a beta version so you may find some issues here. But please do not worry, we are continuously updating it. Besides, any ideas or feedbacks are welcome. Please feel free to leave your comments on the **update #12**. Thanks!

1. Introduction

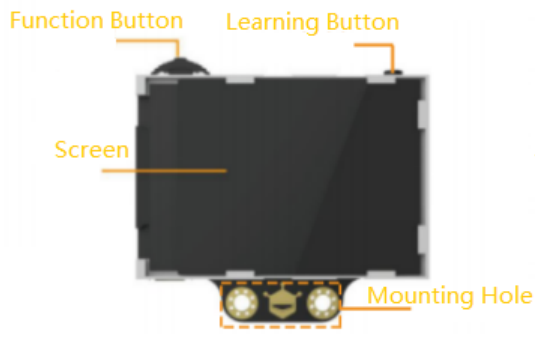
HuskyLens is an easy-to-use AI machine vision sensor with 6 built-in functions: face recognition, object tracking, object recognition, line following, color detection and tag detection.

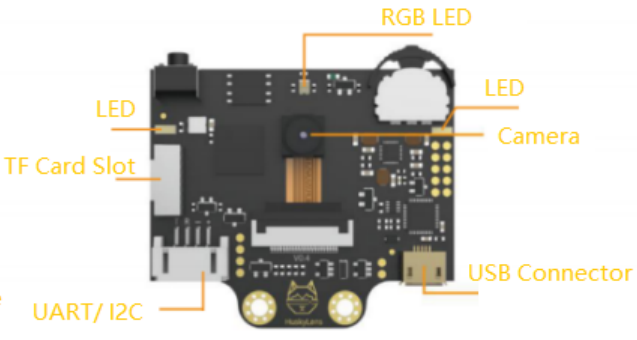
Through the UART / I2C port, HuskyLens can connect to Arduino, Raspberry Pi, or micro:bit to help you make very creative projects without playing with complex algorithms.

2. Specification

- Processor: Kendryte K210
- Image Sensor:
  - SEN0305 HuskyLens: OV2640 (2.0Megapixel Camera)
  - SEN0336 HuskyLens PRO: OV5640 (5.0MegaPixel Camera)
- Supply Voltage: 3.3~5.0V
- Current Consumption(TYP): 320mA@3.3V, 230mA@5.0V (face recognition mode; 80% backlight brightness; fill light off)
- Connection Interface: UART; I2C
- Display: 2.0-inch IPS screen with 320*240 resolution
- Built-in Algorithms: Face Recognition, Object Tracking, Object Recognition, Line Tracking, Color Recognition, Tag Recognition
- Dimension: 52mm*44.5mm / 2.05*1.75"

3. Board Overview





3.1 Connectors

- USB Connector: power supply for Huskylens; connect to the computer to upgrade the firmware
- 4pin Connector in UART Mode

Num	Label	Pin Function	Description
1	T	TX	TX pin of HuskyLens
2	R	RX	RX pin of HuskyLens
3	-	GND	Negative pole of power supply(0V)
4	+	VCC	Positive pole of power supply(3.3~5.0V)

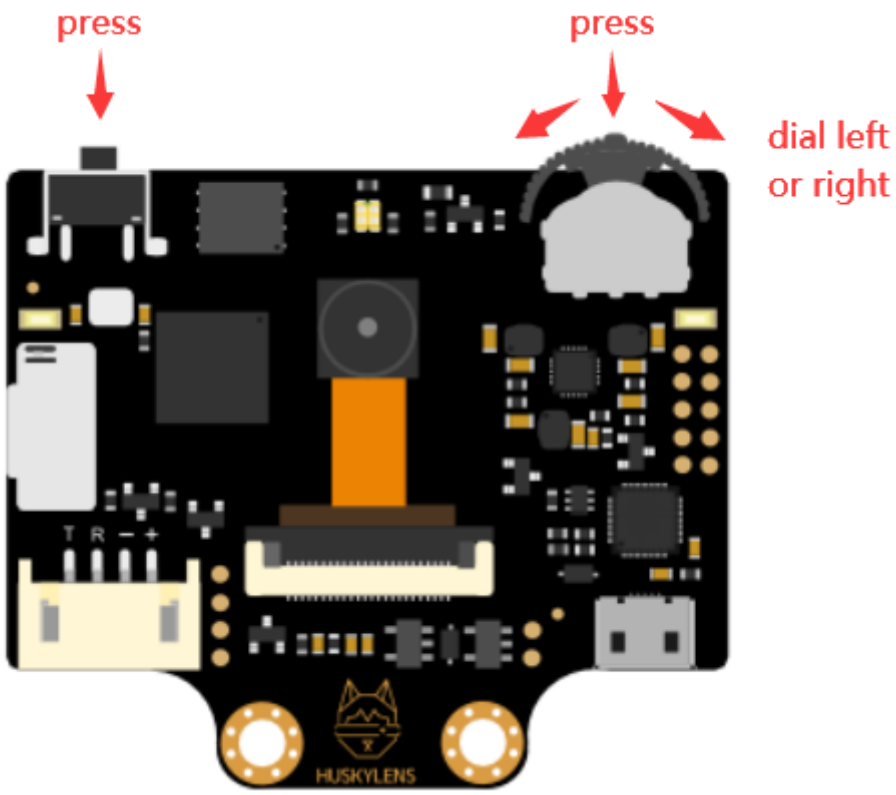
- 4pin Connector in I2C Mode

Num	Label	Pin Function	Description
1	T	SDA	Serial clock line
2	R	SCL	Serial data line
3	-	GND	Negative (0V)
4	+	VCC	Positive (3.3~5.0V)

3.2 Buttons

There are two buttons on the HuskyLens, the function button and the learning button. The basic operations of these two buttons are shown as follows:

- Dial the "function button" to left or right to switch different functions.
- Short press the "Learning button" to learn the specified object; long press the "Learning button" to continuously learn the specified object from different angles and distances; if HuskyLens has learned the object before, short press the "Learn button" to make it forget.
- Long press the "function button" to enter into the second-level menu(parameter setting) in the current function. Dial left, right or short press the "function button" to set related parameters.



4. Upgrade Firmware

Before using this product, it is strongly recommended to upgrade HuskyLens' firmware to the latest version, especially for Kickstarter backers. We recommand to upload the firmware on windows using the K-Flash software since it features a GUI, and easy-to-use.

4.1 In Windows

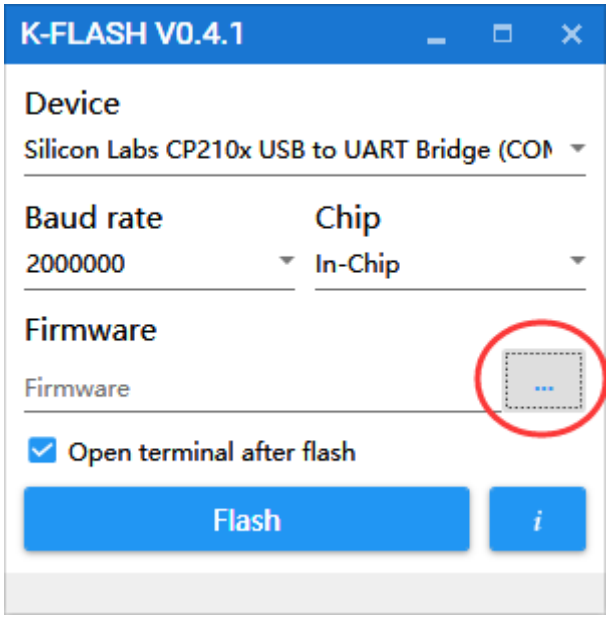
It is recommended to run the K-Flash software on Windows 10.

The steps are shown below:

1. Download the K-Flash software. **Click here** to download it. If K-Flash cannot run, please install .NET Framework 4.7.1 first. **Click here** to download it.
2. Download the USB to UART driver, and install it. **Click here** to download it.

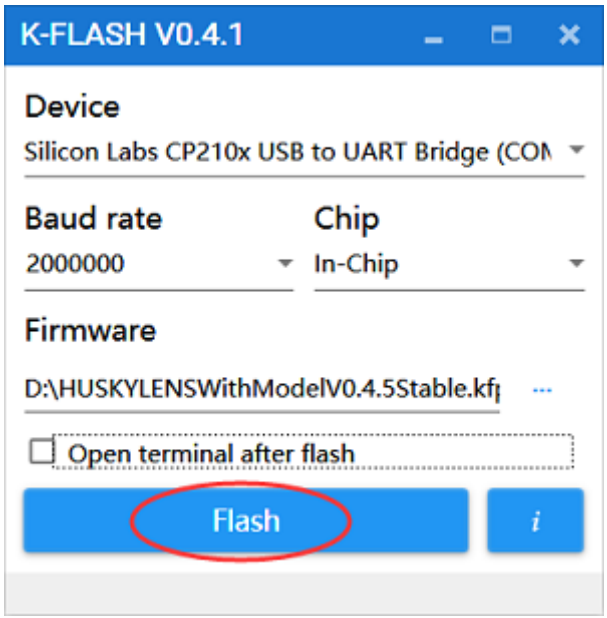
   HuskyLesn adopts the CP2102N chip to implement the USB to serial port function.
3. Download the latest firmware. **Click here** to check the firmwares of all versions.

   In this tutorial, we adopts this firmware: **HUSKYLENSWithModelV0.4.6Stable.kfpkg**.
4. Open the K-Flash software, then click the button(...) to load the new firmware.

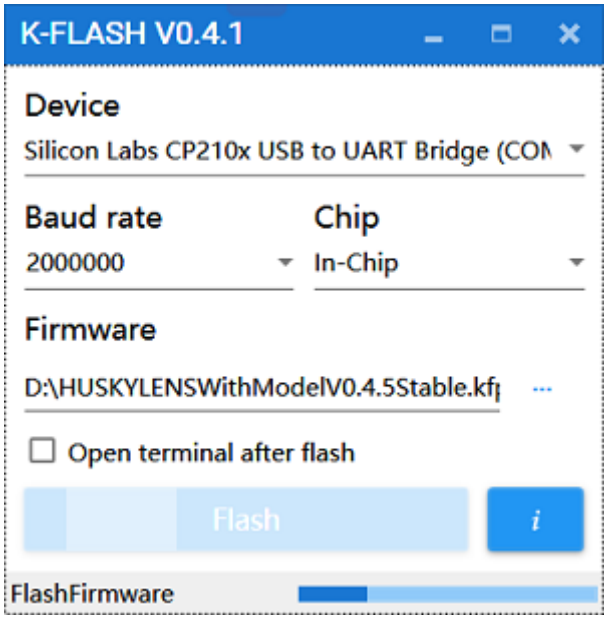


5. Set the K-Flash software according to the parameters shown below:

- Device: select the COM port on your computer
- Baud rage: 2000000
- Chip: In-Chip
- Uncheck "Open terminal after flash"



6. Click the Flash button. Wait about 5 minutes to complete the uploading. There is a major upgrade so it may take a little bit time.



7. A box pops out and shows "successful". Upgrade has been completed now. Enjoy it.

4.2 In Linux or Mac

In this section, we take ubuntu 18.04.4 as an example to show you how to upgrade Huskylens firmware on Linux or Mac. These steps are shown as follows:

1. Download the USB to UART driver, and install it. **Click here** to download it.

   HuskyLesn adopts the CP2102N chip to implement the USB to serial port function.

   In Ubuntu 18.04.4, the USB serial port of HuskyLens can be directly identified when plugged in, so the driver is not required to be installed.

2. Download the latest firmware and kflash.py script. **Click here** to check them.

   In this tutorial, we adopts this firmware: **HUSKYLENSWithModelV0.4.6Stable.kfpkg**.

   You can clone the entire repository of "HuskyLens / HUSKYLENSUploader" to your computer by git command.

3. Install `pip3` first if you do not have it in your OS.

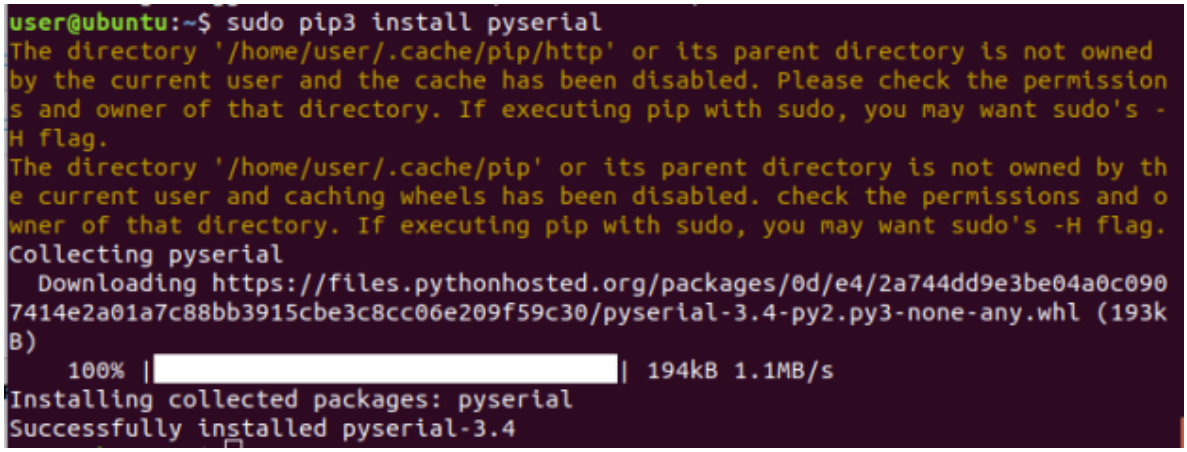
```
sudo apt install python3-pip
```

```
user@ubuntu:~$ sudo apt install python3-pip
[sudo] password for user:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  build-essential dh-python dpkg-dev fakeroot g++ g++-7 gcc gcc-7
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan4 libatomic1 libc-dev-bin libc6-dev libcilkrts5 libexpat1-dev
  libfakeroot libgcc-7-dev libitm1 liblsan0 libmpx2 libpython3-dev
  libpython3.6-dev libquadmath0 libstdc++-7-dev libtsan0 libubsan0
  linux-libc-dev make manpages-dev python-pip-whl python3-dev
  python3-distutils python3-lib2to3 python3-setuptools python3-wheel
  python3.6-dev
Suggested packages:
  debian-keyring g++-multilib g++-7-multilib gcc-7-doc libstdc++6-7-dbg
  gcc-multilib autoconf automake libtool flex bison gcc-doc gcc-7-multilib
  gcc-7-locales libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg
```

````
Install `pip3` on MAC
```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
brew install python3
```
````

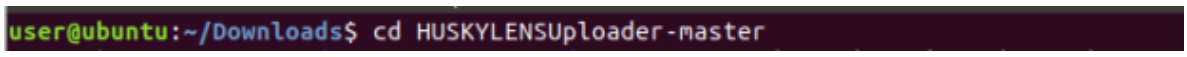
4. Run the following script to install `pyserial`.

```
sudo pip3 install pyserial
```



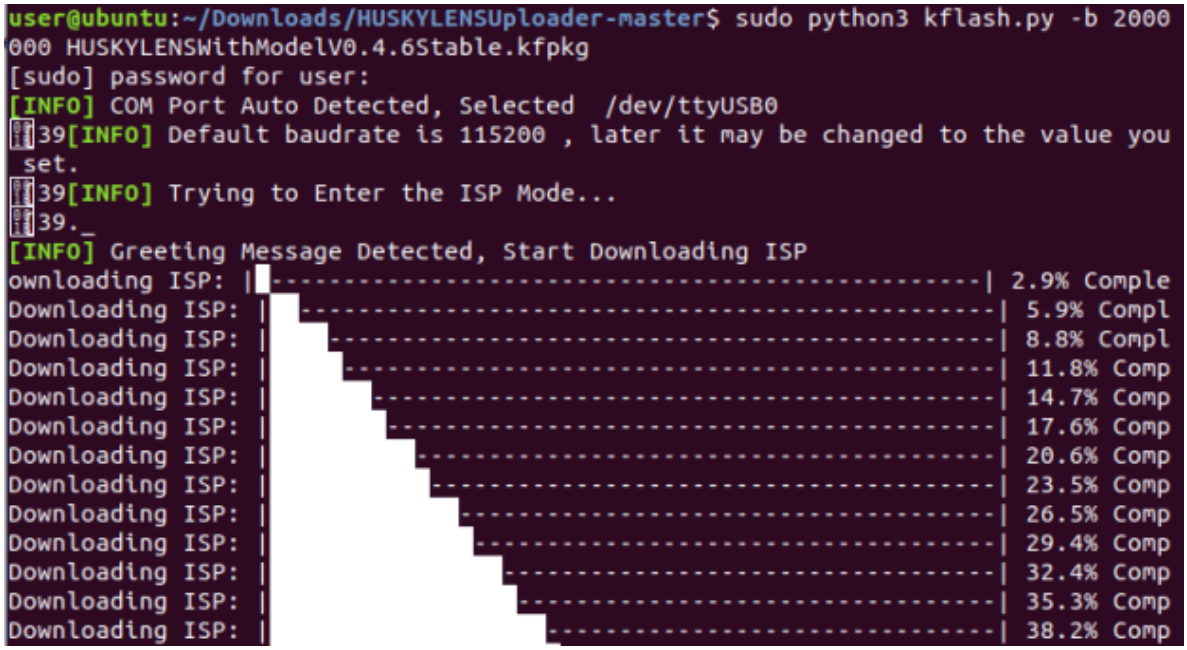
5. Go to the `HUSKYLENSUploader` folder.

```
cd HUSKYLENSUploader
```



6. Run the following script to upload the firmware.

```
sudo python3 kflash.py -b 2000000 HUSKYLENSWithModelV0.4.6Stable.kfpkg
```



7. Wait about 5 minutes to complete the uploading.

```
[INFO] Writing clearFlash.bin into 0x00d90000
ownloading: |                                                  | 100.0% Complete
[INFO] Writing clearFlash.bin into 0x00da0000
ownloading: |                                                  | 100.0% Complete
[INFO] Writing clearFlash.bin into 0x00db0000
ownloading: |                                                  | 100.0% Complete
[INFO] Writing detect.kmodel into 0x00600000
Downloading: |                                                 | 100.0% Complet
e
[INFO] Writing key_point.kmodel into 0x0065f000
Downloading: |                                                 | 100.0% Complet
e
[INFO] Writing feature.kmodel into 0x00680000
Downloading: |                                                 | 100.0% Complet
e
[INFO] Writing mobilenetv1_1.0.kmodel into 0x001bb000
Downloading: |                                                 | 100.0% Complet
e
[INFO] Writing object_detect.bin into 0x009d4000
Downloading: |                                                 | 100.0% Complet
e
[INFO] Rebooting...
```

8. Upgrade has been completed now. Enjoy it.

5. General Settings

5.1 Basic Operation

There are 10 different parameter settings in the general settings.

The basic operation is shown as follows:

1. Select the general settings: Dial the function button right until the words “General Settings” is displayed at the top of the screen, then select it.
2. Enter the general setting mode: Short press the function button to enter it.

3. Dial the function button left or right to select different parameter, then short press the function button to set the parameter. Dial the function button left or right to adjust the parameter. Then short press the function button again to confirm the parameter.
4. Save the settings: After adjusting the parameters, dial the function button left to select "Save & Return", then short press the function button. A message “Do you save data?” will appear. The default seletion is "Yes". At this time, short press the funtion button to save and exit.

5.2 Parameters Introduction

- **Protocol Type**

  Huskylens supports three UART baud rates (9600, 115200, 1000000), and I2C protocol. In addition, it supports auto-detection of the protocols, that is to say, Huskylens will automatically switch between UART and I2C. We recommend to use the auto detection protocol, which is simple and convenient. The default value is auto-detection.

- **Screen Brightness**

  The screen supports the brightness from 1 ~ 100. The default value is 80.

- **Menu Auto-hide**

  When you don't operate the Huskylens for a period of time, the menu on the screen will automatically hide. This duration time can be adjusted from 1-100 seconds. The default value is 10 seconds.

- **LED Light**

  There are two LED lights on the front of the Huskylens. You can set it ON or OFF. The default value is OFF.

- **LED Brightness**

  The brightness of these two LED lights ranges from 1 to 100. The default value is 50.

- **RGB Light**

  There is also an RGB light on the front of the Huskylens. You can set it ON or OFF. The default value is ON.

- **RGB Brightness**

  The brightness range of this RGB light is 1--100. The default is 20.

- **Factory Reset**

  Huskylens can be reseted to factory settings via this function.

- **Version**

  The current version of the built-in firmware.

- **Language**

  Huskylens supports Chinese and English.

6. Definitions You Need to Know

6.1 Color Instructions

In each function, the color definitions of the frame and the symbol "+" in the center of the screen are all the same, which helps you know the current status of HuskyLens.

Color	Status
From orange to yellow, then from yellow to orange	Have not learned the object yet but ready to learn
Yellow	Learning the new object
Blue	Have learned the object and recognized it

The RGB LED indicator is currently only used to indicate the status of the face recognition function. Its colors are defined as follows.

Color	Status
Blue	Have not learned the face yet, but detected the face
Yellow	Learning the new face
Green	Have learned the face and recognized it

6.2 Coordinate System

When HuskyLens is detecting a learned object, the target will be automatically selected by a color frame on the screen. The coordinates of the color frame position x and y are assigned according to the following coordinate system. After getting the coordinates from the UART / I2C port, you can know the position of the object.

Format: (x,y)

7. Functions Introduction

7.1 Face Recognition

This function can detect any face contour, recognize and track the learned face.

7.1.1. Learn one face

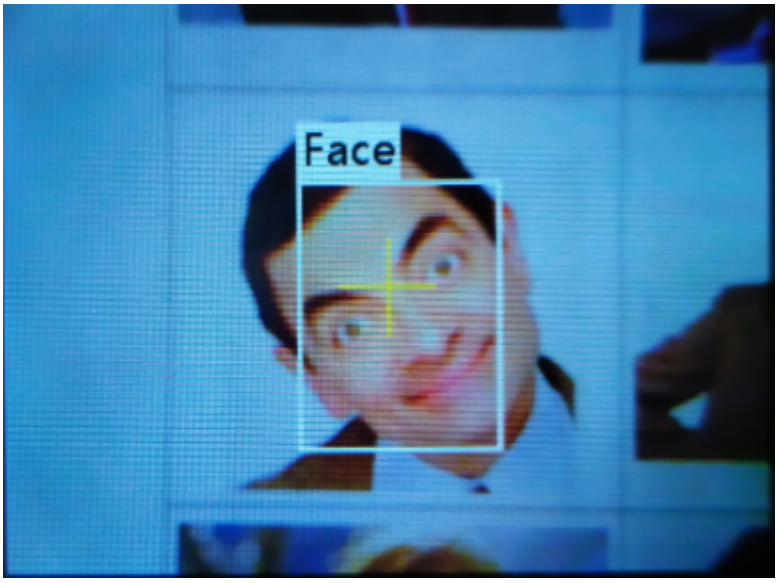
The default setting is to learn and recognize a single face.

Operation and Setting

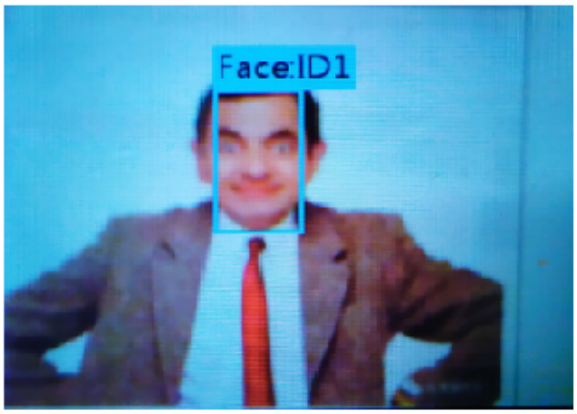
Dial the function button to the left until the word "Face recognition" is displayed at the top of the screen.

Learning and Detection

1. Face Detection:
Point the HuskyLens at any faces. When a face is detected, it will be automatically selected by a white frame with words "Face" on the screen.

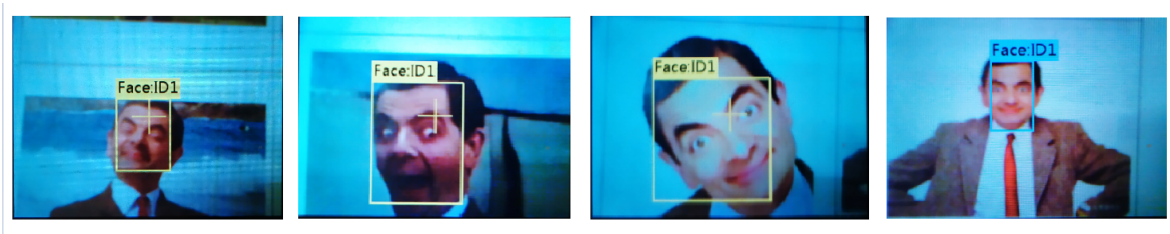


2. Face Learning: Point the “+” symbol at a face, short press the "learning button" to learn the face. If the same face is detected by HuskyLens, a blue frame with words "Face: ID1" will be displayed on the screen， which indicates that HuskyLens has learned the face before and can recognize it now.



However, HuskyLens only learned one plane (one-dimensional) of the face after the above operation, while human face is three-dimensional. If the angle of the face has been changed, HuskyLens may not recognize it. So you need to let HuskyLens learn a face from its different angles. The operation shows as follows：(Before HuskyLens learning news thing, please let it forget the former things first.)

Keep pressing the “learning button”, point HuskyLens' "+" at different angles of the face. During this process, a yellow frame with words "Face: ID1" will be displayed on the face, which indicates HuskyLens is learning the face. After HuskyLens learned the face from all angles, release the "learning button". Then when Huskylens detected the learned face, a blue frame with words "Face: ID1" will be displayed, indicating that the process of face learning is completed successfully. Now HuskyLens can recognize the face from different angles.

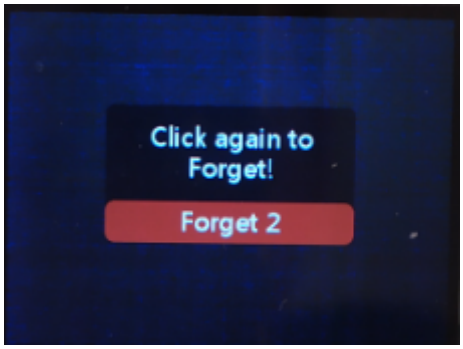


Tips: If there is no “+” symbol in the center of the screen before learning, it means that the HuskyLens has already learned the face in the current function, now HuskyLens is detecting it. If you want to let HuskyLens learn a new face, you have to make it forget the learned face first.

3. Face Recognition:
The learned face information will be automatically saved. When HuskyLens detects the learned face from multiple faces, this face will be selected by a blue frame and identified as face: ID1.

4. Forget the Learned Face:
If you want to recognize another face, or re-enter face information interface, you need to delete the current face information. When HuskyLens is in the face recognition mode, 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, then the yellow "+" symbol is displayed. Now you can let HuskyLens learn a new face.

The operation of forgetting is totally the same in other functions. Therefore, this operation will not be repeated in subsequent chapters.



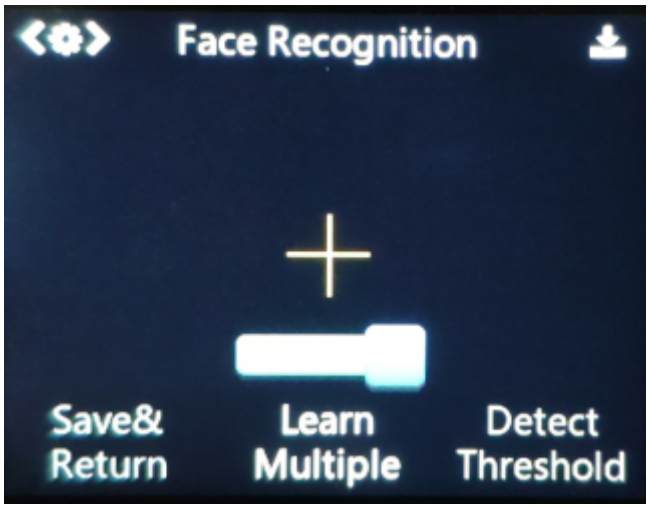
7.1.2. Learn multiple faces

The default setting is to learn a single face. In order to learn multiple faces, we need to enable "Learn Multiple" of face recognition.

Operation and Setting

1. Dial the function button to the left until the word "Face recognition" is displayed at the top of the screen.
2. Long press the function button to enter the parameter setting of the face 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.

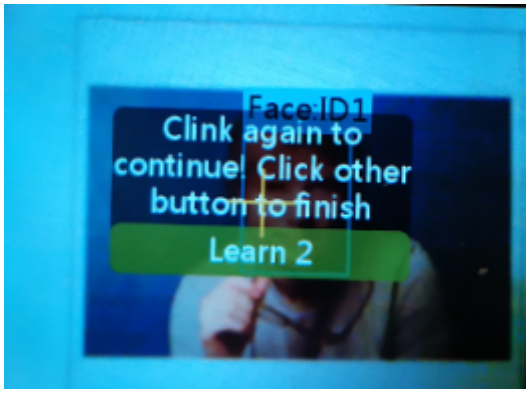


4. Dial the function button to the left until "Save & Return" shows. And the screen prompts "Do you want to save the parameters?" Select "Yes" in default, now short-press the function button to save the parameters and return automatically.

Learning and Detection

1. Multiple Faces Learning:

Point the “+” symbol at the face, long press the "learning button" to learn the face of the first person. Then release the "learning button", a blue frame with words "Face: ID1" will be displayed if HuskyLens detects the same face， meanwhile, a message "Click again to continue! Click other button to finish" will be displayed. Please short press the "learning button" before the countdown ends if you want to learn the face of other person. If not, short press the "function button" before the countdown ends, or do not press any button to let the countdown ends.

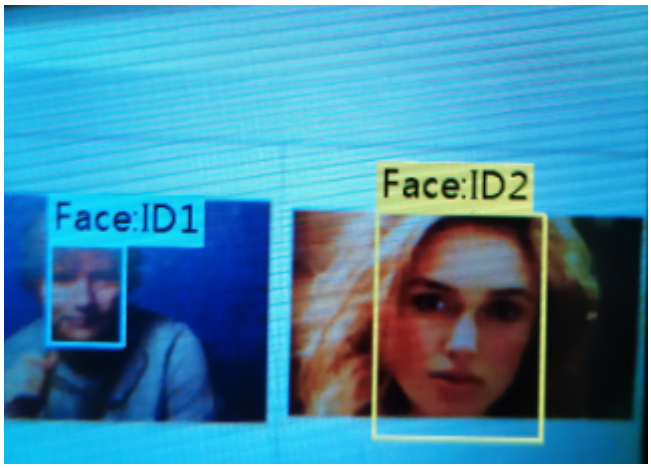


In this chapter, we will learn the next face continuously. So we need to short press the "learning button" before the countdown ends. Then we can let HuskyLens learn the face of the second person. The same as the steps to recognize the first face, point the “+” symbol at the second face, long press the "learning button" to learn the face of the second person. Then release the "learning button", a blue frame with words "Face: ID2" will be displayed if HuskyLens detects the same face.

Tips: If there is no “+” symbol in the center of the screen before learning, it means that the HuskyLens has already learned, now HuskyLens is detecting face. If you want to let HuskLens learn new face, you need to let HuskyLens forget the learned face first.

Please turn to the 7.1.1. Learn one face to check the way to forget the learned face.

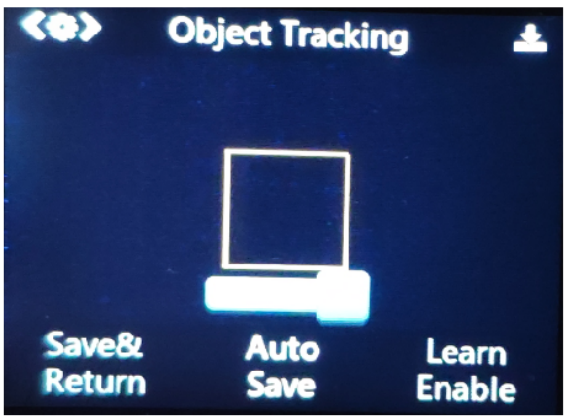
2. Multiple Faces Recognition:
The learned face information will be saved automatically. When HuskyLens detects the learned face from multiple faces, the learned face will be selected with a frame and identified by the message face: IDx. For example, when HuskyLens detects the learned face of the first person, it will be selected with a blue frame and identify face: ID1; when HuskyLens detects the learned face of the second person, it will be selected with a yellow frame and identify face: ID2; and so on.



7.2 Object Tracking

Operation and Setting

1. Dial the function button to the left or right until the word "Object Tracking" is displayed at the top of the screen.
2. Long press the function button to enter the parameter setting of the object tracking function.
3. 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. The method to turn on the switch of saving models automatically is the same as before. According to the steps above to switch "Auto Save" ON.

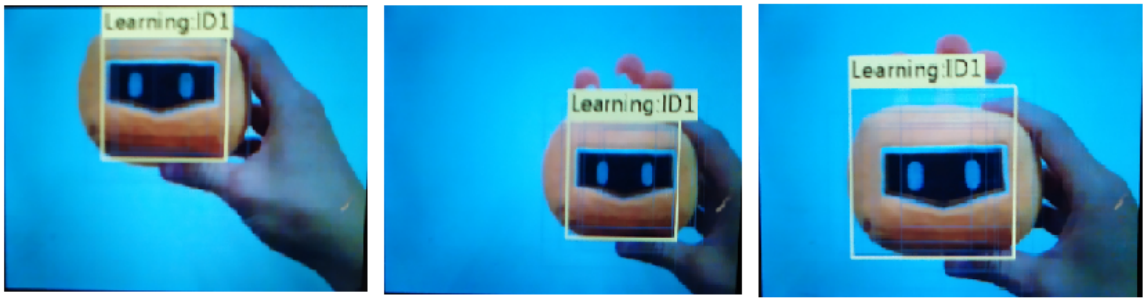


5. You can also adjust the size of the frame by setting "Frame Ratio" and "Frame Size" to match the shape of the object.
6. Dial the function button to the left to select "Save & Return", and short press the function button to save the parameters and return automatically.

Learning and Detection

1. Obejct Learning:
Point Huskylens to the target object, djusting the distance and until the object is included in the yellow frame of the center of the screen. Then long press "learning button" to learn the object from various angles and distances. During the learning process, the yellow frame with words "Learning: ID1" will be displayed on the screen. Then release the "learning button" to complete the learning.

2. Obejct Tracking:
Move the HuskyLens or the target, the frame will track the target automatically.
When tracking the object, the yellow words “Learning: ID1” will be displayed, indicating that HuskyLens is tracking the object while learning. This setting improves the object tracking ability. You can also press and hold the "function button" to enter the secondary menu parameter settings, select "Learn on" and turn off this parameter.



Tips:

- Only one object can be tracked at a time. It can be any object with a clear outline, even various gestures.
- If there is no yellow frame on the center of the screen, it means that the HuskyLens has already learned a object. Please refer to the method of delete faces in face recognition to delete the learned object.

7.3 Obejct Recognition

HuskyLens can recognize 20 built-in objects. They are aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, dining-able, dog, horse, motorbike, person, potted lant, sheep, sofa, train, TV.

7.3.1. Recognize a single object

The default setting is to recognize a single object.

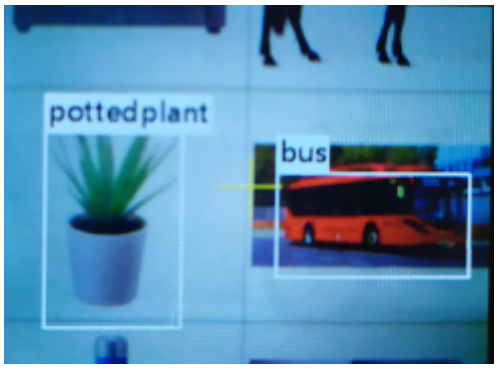
Operation and Setting

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

Learning and Detection

1. Obejct Detection:
When detecting objects, HuskyLens will automatically recognize it, and the object will be displayed by the white frame with it's name on the screen.

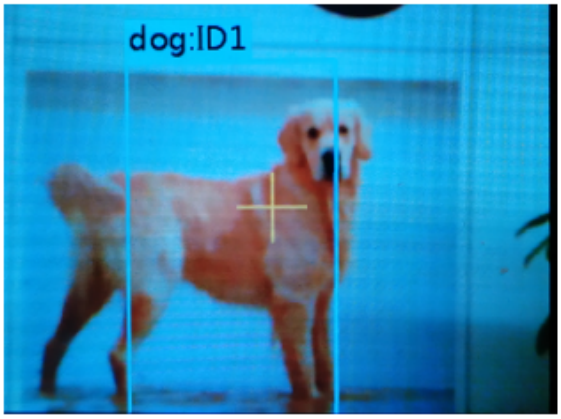
At present, only 20 built-in objects can be recognized, and the remaining objects cannot be recognized temporarily.



2. Obejct Mark:
Point the “+” symbol at the object, then short press the “learning button”. When pressing, the color of the frame changes from white to blue, and the name of the object and it's ID number will appear on the screen.

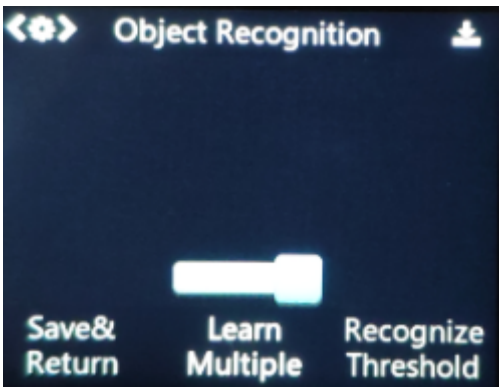
Tips: If there is no yellow "+" symbol on the center of the screen, it means that the HuskyLens has already learned a object. Please refer to the method of delete faces in face recognition to delete the learned object.

3. Obejct Recognition:
When encountering the learned objects, they will be selected by the blue frame, and the name and ID number will be displayed. When encountering new ones, the selection frame is white. This can be used as a simple filter to find out what you need from a bunch of objects.



7.3.2. Recognize multiple objects

The default setting is to recognize a single object, so we need to enable "Learn Multiple" parameter of the object recognition function first.
Please refer to the multiple faces learning and recognition chapter for setting, this chapter will not repeat to introduce it.



The ID number is related to the order of marking objects. For example, if a dog is marked for the first time and a cat is marked for the second time, when the dog is recognized, the words "dog: ID1" will be displayed on the screen; and when the cat is recognized, the words "cat: ID2" will be displayed on the screen.

7.4 Line Tracking

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 Multipe" is selected, then short press the function button, and dial it to the left to turn off the "Learn Multipe" 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 LEDs by setting "LED Switch". This is very useful in the dark environment.

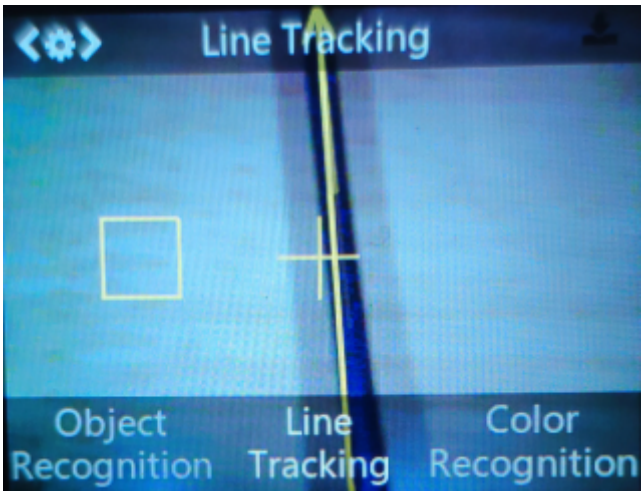
5. Dial the function button to the left until "Save & Return" is selected, and short press the function button to save the parameters and it will return automatically.

Learning and Detection

1. Line Learning:

Point the "+" symbol at the line, then point the yellow frame at the background area. It is recommended that within the view field of HuskyLens, just remain lines to learn and no any cross lines.

Then, short press the "learning button" to complete the learning process. A blue route direction arrow will appear on the screen.



Tips: If there is no yellow frame and "+" symbol on the center of the screen, it means that the HuskyLens has already learned the line. Please refer to the method of delete faces in face recognition to delete the learned line.

2. Line Prediction:

When HuskyLens detects the line which has been learned, a blue arrow will appear automatically on the screen. The direction of the arrow indicates the predicted direction of the line.

Tips:

- When learning the line, we need to adjust the position of HuskyLens to be parallel to the line.
- Huskylens can learn multiple lines according to the color of lines, but these lines must be monochrome lines with obvious color that different from the background.
- In most cases, the color of tracking line is only one. Threfore, in order to ensure stability, we recommend to track the single line.
- 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.

7.5 Color Recognition

7.5.1. Learn a single color

The default setting is to learn a single color.

Operation and Setting

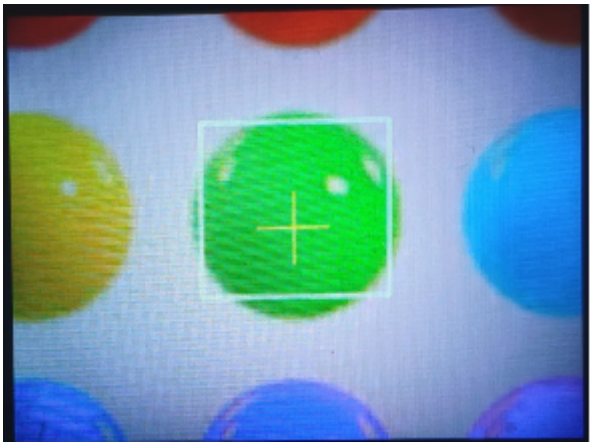
Dial the function button to the right or left until the word "Color Recognition" is displayed at the top of the screen.

Learning and Detection

1. Color Detection:

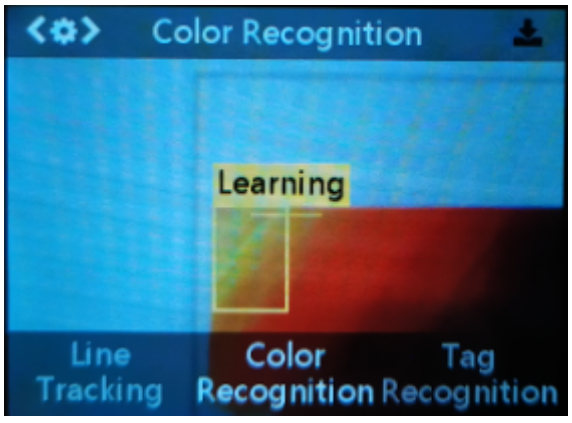
Point the icon "+" in the center of the HuskyLens screen to the target color block, and a white box will appear on the screen, which selects the target color block automatically.

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:

Point the “+” symbol at the color block, and long press the “learning button”. A yellow frame will displayed on the screen, indicating that HuskyLens is learning the color. At this time, adjust the distance and angle between HuskyLens and the color block, to make the size of yellow frame same as the color block. Then, release the "learning button" to complete the learning.

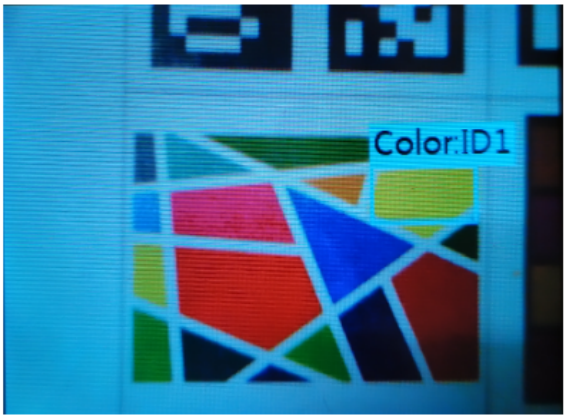


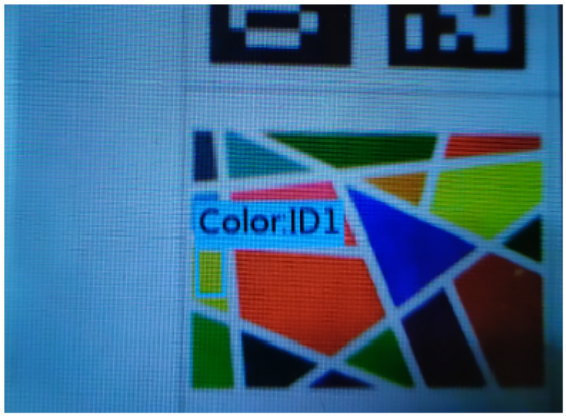
Tips: If there is no “+” symbol in the center of the screen before learning, it means that the HuskyLens has already learned, now HuskyLens is detecting color. If you want to let HuskLens learn new color, you need to let HuskyLens forget the learned color first.

3. Color Recognition:

When encountering the same or similar color blocks, a blue frame with an ID will be automatically displayed on the screen, and the size of the blue frame is same as the size of the color blocks.

When there are multiple same or similar color blocks appear at the same time, the other color blocks cannot be recognized, that is, only one color block can be recognized at each time.

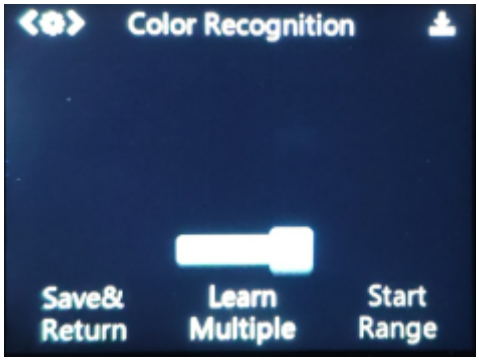




Tips: Color recognition is greatly affected by ambient light. Sometimes HuskyLens may misidentify similar colors. Please try to keep the ambient light unchanged.

7.5.2. Learn multiple colors

The default setting is to recognize a single color, so we need to enable "Learn Multiple" parameter of the color recognition function. Please refer to the multiple faces learning and recognition chapter for setting, this chapter will not repeat it.



The ID number is related to the order of learned color. For example, if a yellow block is marked for the first time and a green block is marked for the second time, when the yellow block is recognized, the words "Color: ID1" will be displayed on the screen, and when the green block is recognized, the words "Color: ID2" will be displayed on the screen.

7.6 Tag Recognition

7.6.1. Learn a single tag

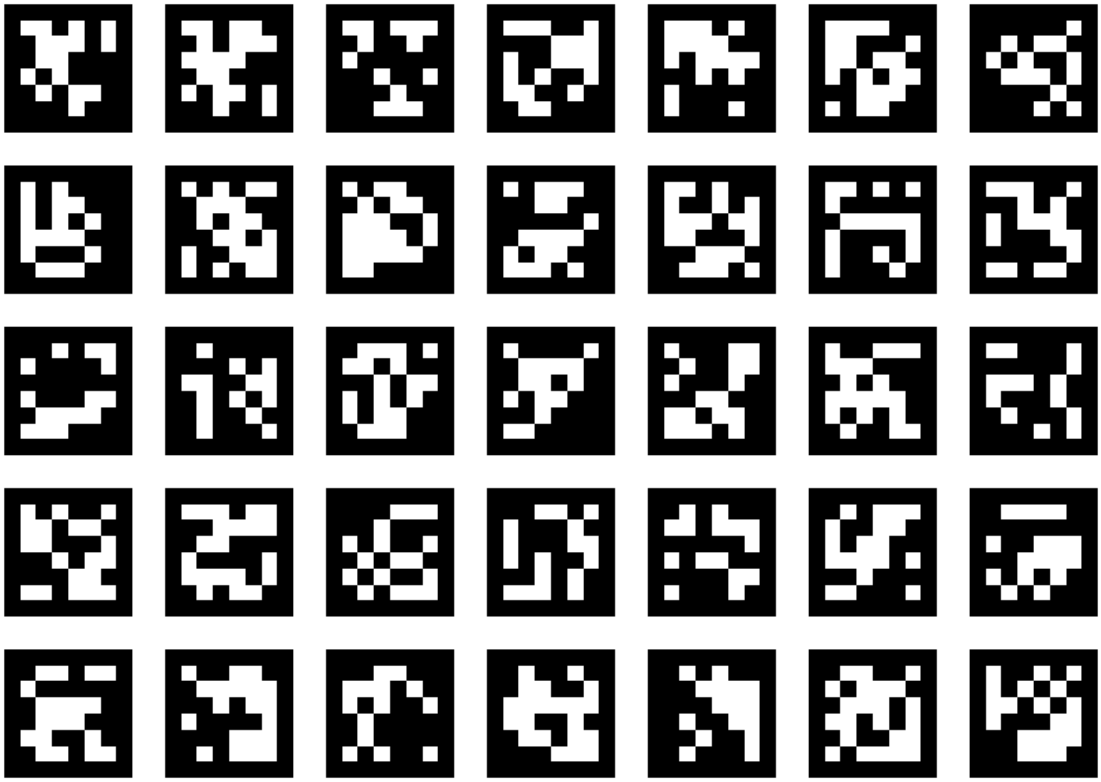
The default setting is to learn a single tag.

Operation and Setting

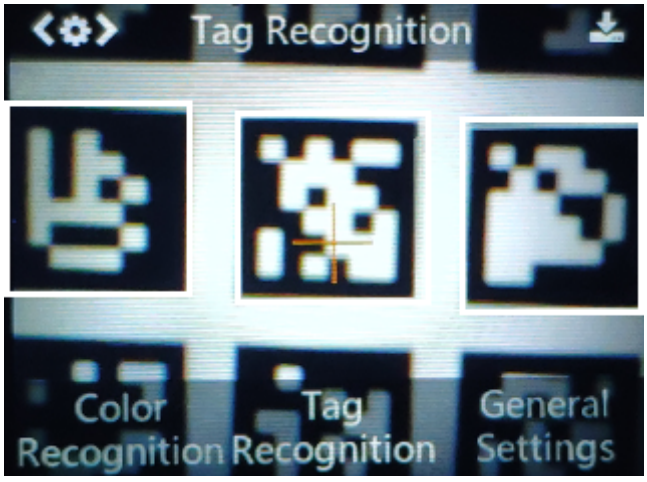
Dial the function button to the right or left until the words "Tag Recognition" is displayed at the top of the screen.

Learning and Detection

You can use the following tag / QR code pictures to test this function.

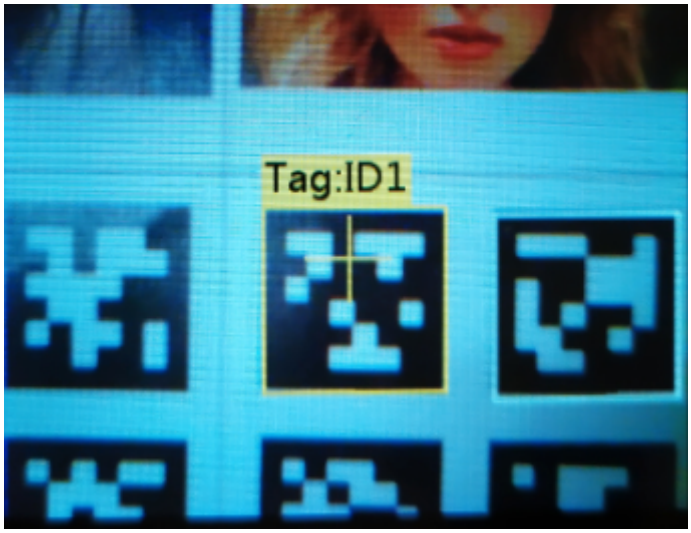


1. Tag Detection:
When Huskylens detects the tag, the tag will be automatically selected by the white frame on the screen.



2. Tag Learning:

Point the “+” symbol at the tag, and press the “learning button”. A yellow frame with words "Tag:ID1" will be displayed on the screen, indicating that HuskyLens is learning the tag now. Then, release the "learning button" to complete the learning process.



Tips: If there is no “+” symbol in the center of the screen before learning, it means that the HuskyLens has already learned, now HuskyLens is detecting tag. If you want to let HuskLens learn new tag, you need to let HuskyLens forget the learned tag first.

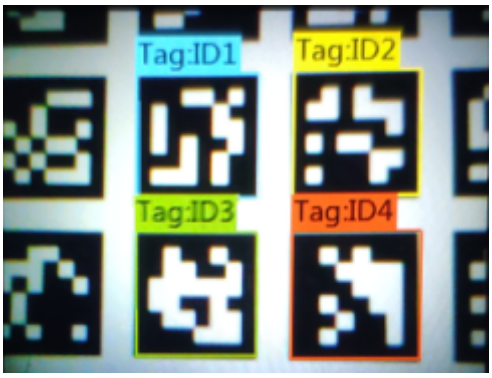
3. Tag Recognition

When encountering the learned tag, a blue frame with an ID will be automatically displayed on the screen.

7.6.2. Learn multiple tags

The default setting is to recognize a single tag, so we need to enable “Learn Multiple” parameter of the tag recognition function. Please refer to the multiple faces learning and recognition chapter for setting, this chapter will not repeat it.

Similarly, the ID is related to the order of learned tags.

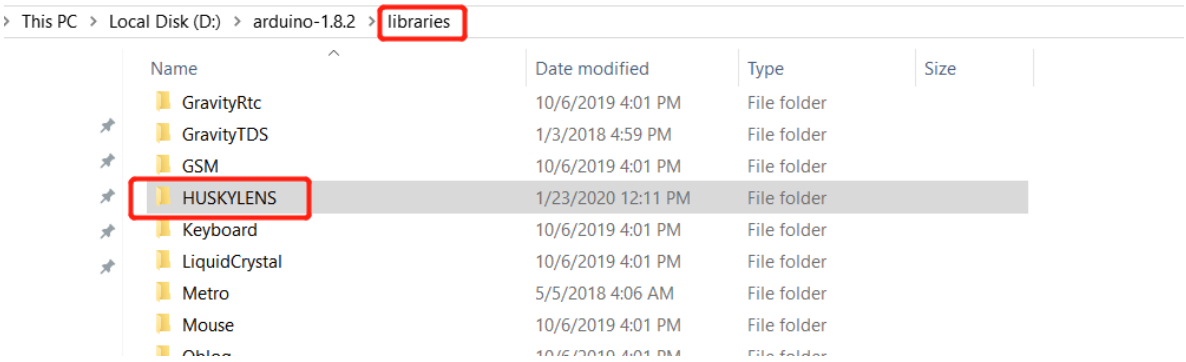


8. Arduino Tutorial

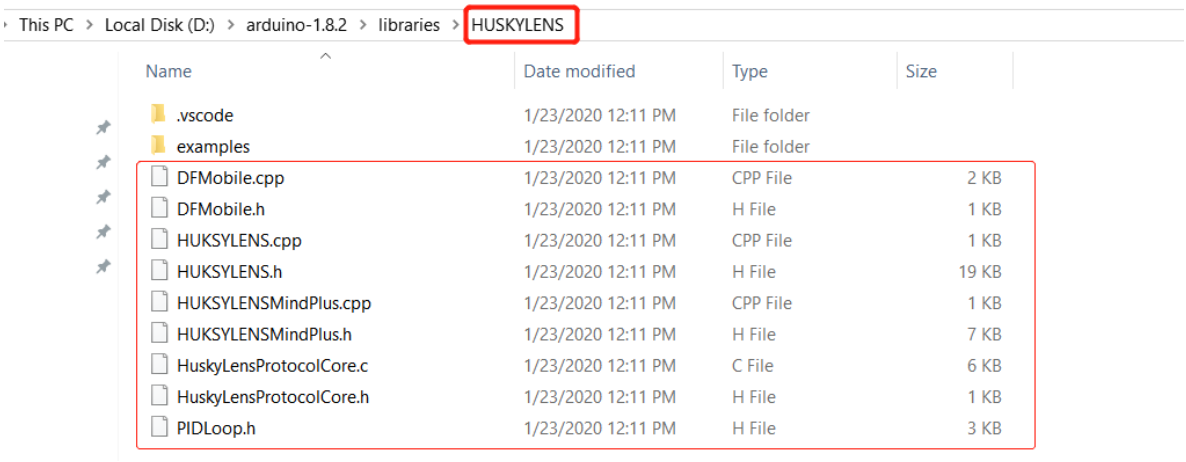
Please download and install the **HUSKYLENS Library** first.

8.1 Install the Library

1. Unzip the file, then copy the folder to the "libraries" folder of the Arduino IDE. Then check whether the folder name is "HUSKYLENS". If not, please change it as "HUSKYLENS". Please note that the library file name must be HUSKYLENS.



2. All .h files and .cpp files must in the root directory of the "HUSKYELSN" folder.



8.2 API Introduction

Please click the link below to view the API introduction documents.

https://github.com/HuskyLens/HUSKYLENSArduino/blob/master/HUSKYLENS%20Arduino%20API.md

8.3 Project 1: Read Position Data

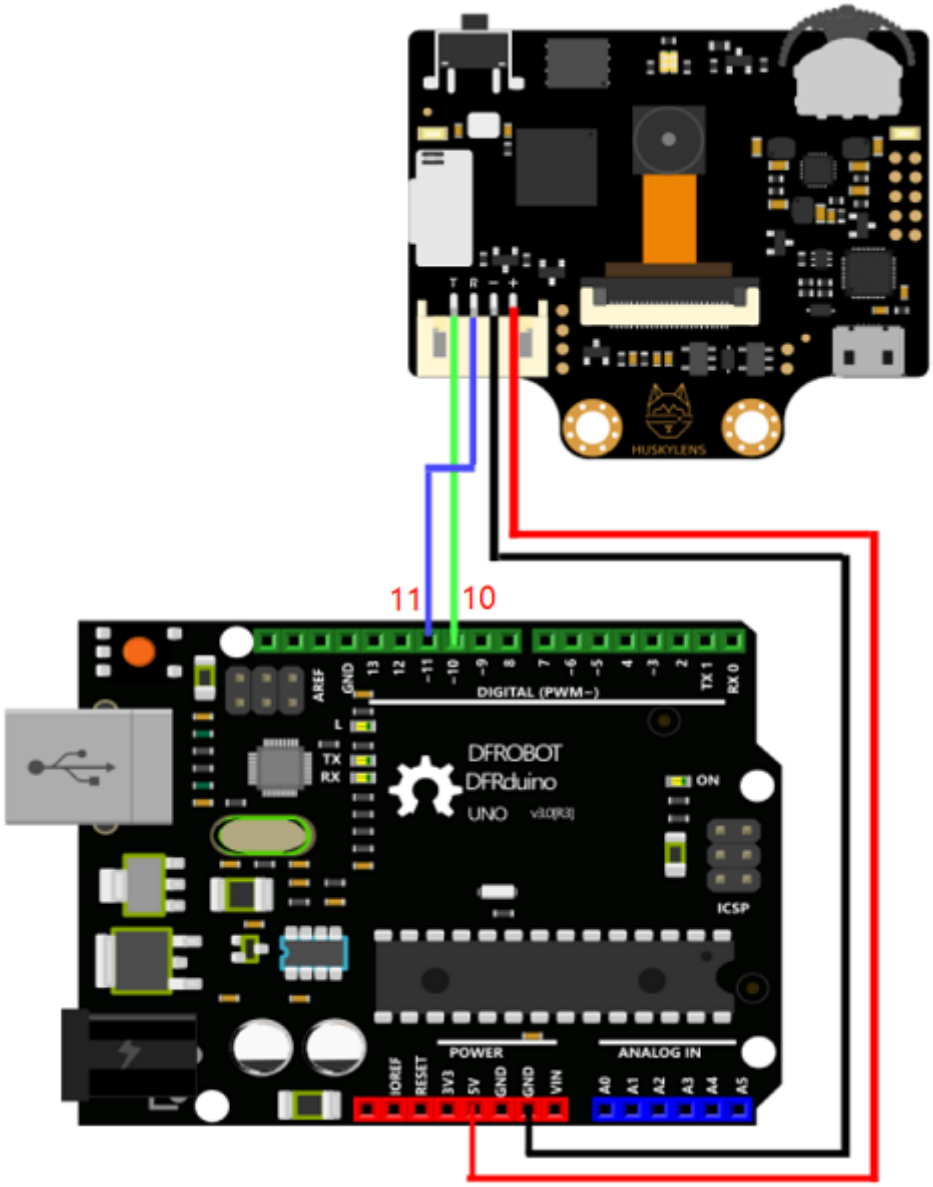
In this project, HuskyLens will be connected to Arduino mainboard. And Arduino Uno will read position data of the object from HuskyLens. Then the serial port monitor will print the data. So that, you can read the position of the object in real time.

Requirements

- **Hardware**
  - DFRduino UNO R3 (or similar) x 1
  - HUSKYLENS x 1
  - M-M/F-M/F-F Jumper wires
- **Software**
  - Arduino IDE(version 1.8.x is recommended)
  - Download and install the **HUSKYLENS Library**. (About how to install the library?)

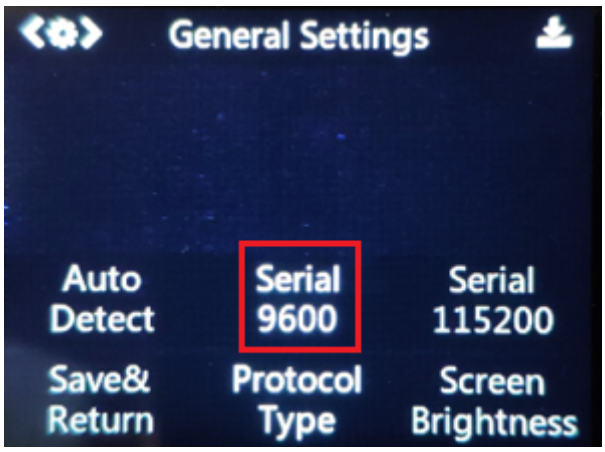
UART Mode(SoftwareSerial)

Connection Diagram



HuskyLens Protocol Setting

You need to set the protocol type of HuskyLens. The protocol should be 'Serial 9600'. Of course, your can adopt the Auto Detect protocol, which is easy-to-use and convenient.



Sample Code

```
#include "HUSKYLENS.h"
#include "SoftwareSerial.h"
```

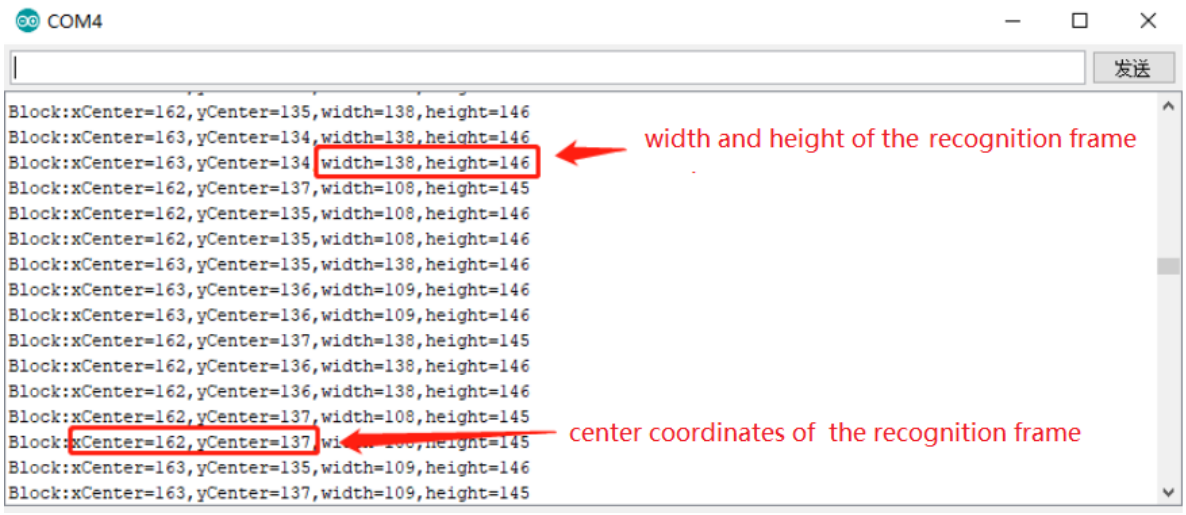
```
HUSKYLENS huskylens;
SoftwareSerial mySerial(10, 11); // RX, TX
//HUSKYLENS green line >> Pin 10; blue line >> Pin 11
void printResult(HUSKYLENSResult result);

void setup() {
    Serial.begin(115200);
    mySerial.begin(9600);
    while (!huskylens.begin(mySerial))
    {
        Serial.println(F("Begin failed!"));
        Serial.println(F("1.Please recheck the \"Protocol Type\" in HUSKYLENS
(General Settings>>Protocol Type>>Serial 9600)"));
        Serial.println(F("2.Please recheck the connection."));
        delay(100);
    }
}

void loop() {
    if (!huskylens.request()) Serial.println(F("Fail to request data from
HUSKYLENS, recheck the connection!"));
    else if(!huskylens.isLearned()) Serial.println(F("Nothing learned, press
learn button on HUSKYLENS to learn one!"));
    else if(!huskylens.available()) Serial.println(F("No block or arrow appears
on the screen!"));
    else
    {
        Serial.println(F("###########"));
        while (huskylens.available())
        {
            HUSKYLENSResult result = huskylens.read();
            printResult(result);
        }
    }
}

void printResult(HUSKYLENSResult result){
    if (result.command == COMMAND_RETURN_BLOCK){

Serial.println(String()+F("Block:xCenter=")+result.xCenter+F(",yCenter=")+result
.yCenter+F(",width=")+result.width+F(",height=")+result.height+F(",ID=")+result.
ID);
    }
    else if (result.command == COMMAND_RETURN_ARROW){

Serial.println(String()+F("Arrow:xOrigin=")+result.xOrigin+F(",yOrigin=")+result
.yOrigin+F(",xTarget=")+result.xTarget+F(",yTarget=")+result.yTarget+F(",ID=")+r
esult.ID);
    }
    else{
        Serial.println("Object unknown!");
    }
}
```

Operations and Expected Results

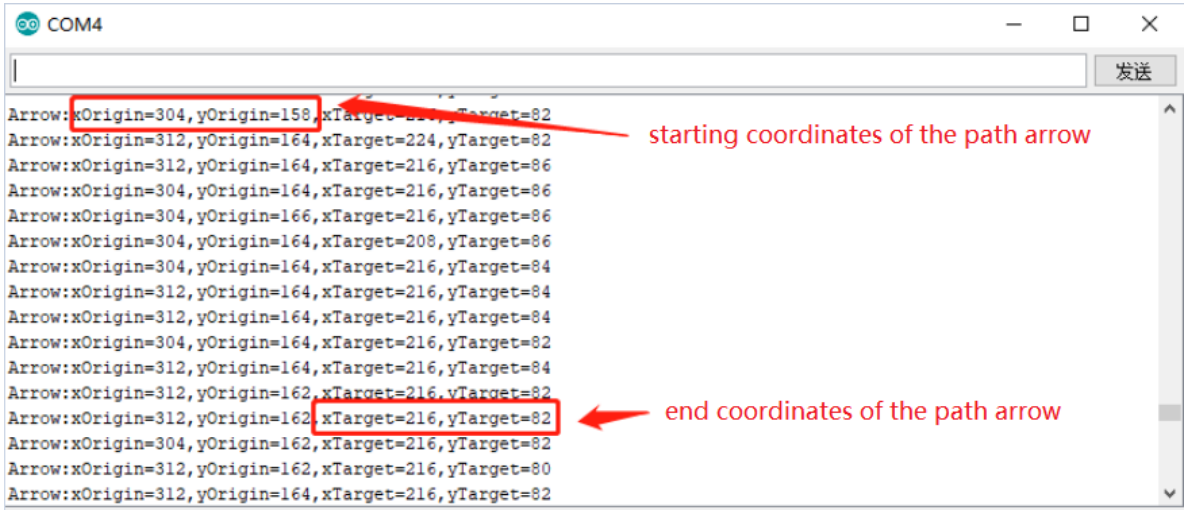
1. Upload the above codes to your Arduino board.

2. Let your HuskyLens learn a new thing first. You can refer to the previous chapters of this tutorial.
3. Open the serial monitor of Arduino IDE, then you will get the position data of the object.

If HuskyLens is in the face recognition, object tracking, object recognition, color recognition, tag recognition mode, you will get the results like follows:

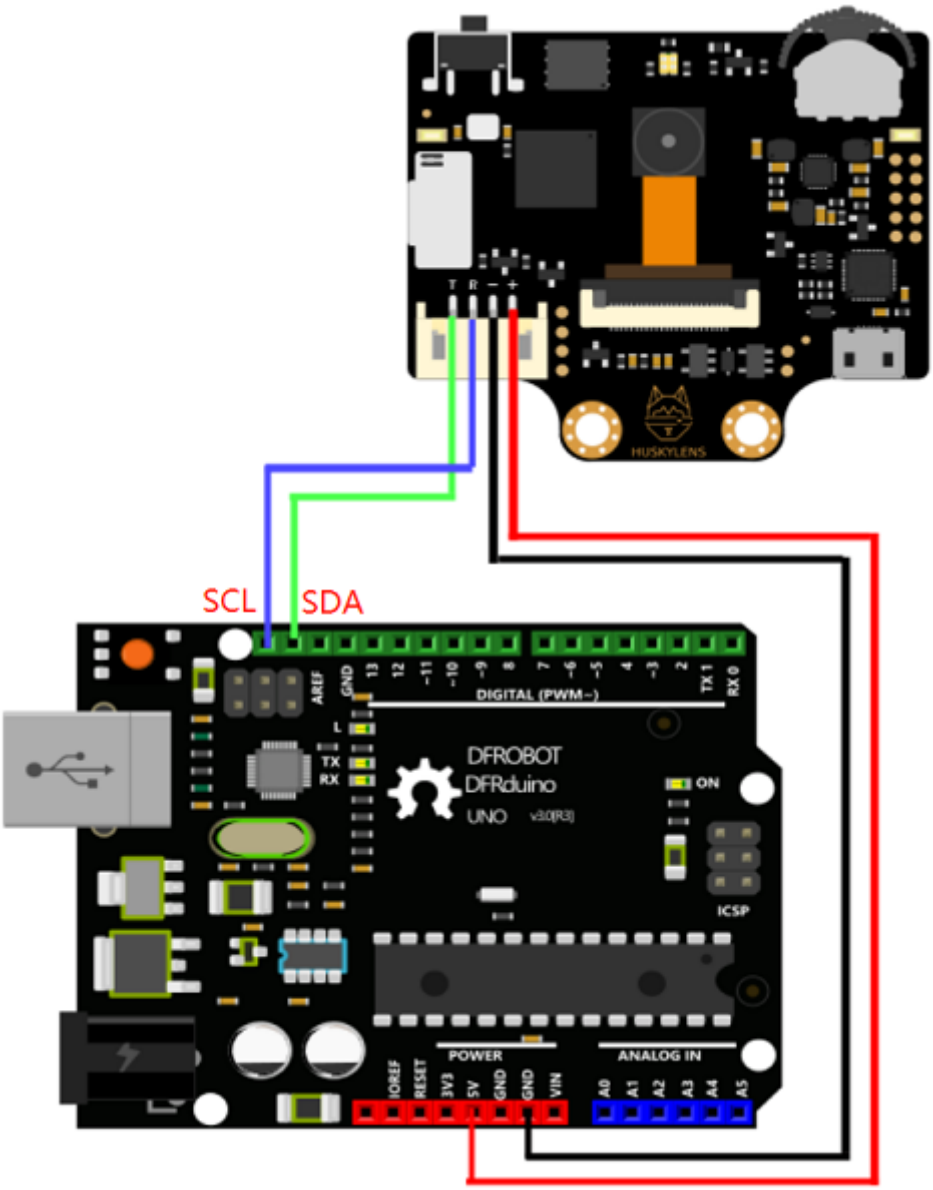


If HuskyLens is in the line tracking mode, you will get the results like follows:



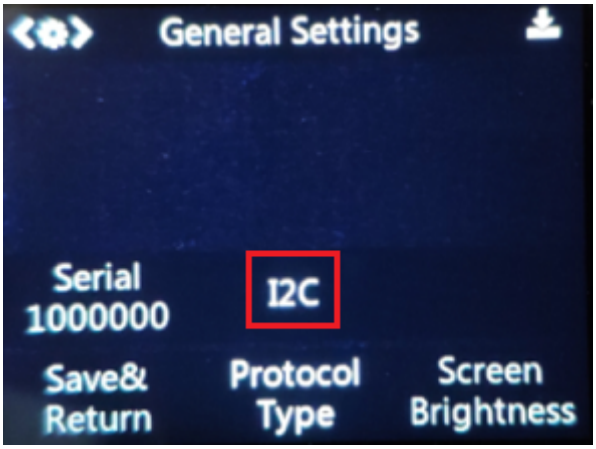
I2C Mode

Connection Diagram



HuskyLens Protocol Setting

You need to set the protocol type of HuskyLens. The protocol should be 'I2C'. Of course, your can adopt the auto detect protocol, which is easy-to-use and convenient.



Sample Code

```
#include "HUSKYLENS.h"
#include "SoftwareSerial.h"

HUSKYLENS huskylens;
```

```
//HUSKYLENS green line >> SDA; blue line >> SCL
void printResult(HUSKYLENSResult result);

void setup() {
    Serial.begin(115200);
    Wire.begin();
    while (!huskylens.begin(Wire))
    {
        Serial.println(F("Begin failed!"));
        Serial.println(F("1.Please recheck the \"Protocol Type\" in HUSKYLENS
(General Settings>>Protocol Type>>I2C)"));
        Serial.println(F("2.Please recheck the connection."));
        delay(100);
    }
}

void loop() {
    if (!huskylens.request()) Serial.println(F("Fail to request data from
HUSKYLENS, recheck the connection!"));
    else if(!huskylens.isLearned()) Serial.println(F("Nothing learned, press
learn button on HUSKYLENS to learn one!"));
    else if(!huskylens.available()) Serial.println(F("No block or arrow appears
on the screen!"));
    else
    {
        Serial.println(F("###########"));
        while (huskylens.available())
        {
            HUSKYLENSResult result = huskylens.read();
            printResult(result);
        }
    }
}

void printResult(HUSKYLENSResult result){
    if (result.command == COMMAND_RETURN_BLOCK){

Serial.println(String()+F("Block:xCenter=")+result.xCenter+F(",yCenter=")+result
.yCenter+F(",width=")+result.width+F(",height=")+result.height+F(",ID=")+result.
ID);
    }
    else if (result.command == COMMAND_RETURN_ARROW){

Serial.println(String()+F("Arrow:xOrigin=")+result.xOrigin+F(",yOrigin=")+result
.yOrigin+F(",xTarget=")+result.xTarget+F(",yTarget=")+result.yTarget+F(",ID=")+r
esult.ID);
    }
    else{
        Serial.println("Object unknown!");
    }
}
```

Operations and Expected Results

1. Upload the above codes to your Arduino board.
2. Let your HuskyLens learn a new thing first. You can refer to the previous chapters of this tutorial.

3. Open the serial monitor of Arduino IDE, then you will get the position data of the object, same as the results in UART mode. Please refer to the previous chapter, which will not be repeated here.

9. Raspberry Pi Tutorial

In this chapter, we will use the Raspberry Pi to read the data from the HuskyLens. The communication protocol is I2C.

9.1 Initialize Raspberry Pi

On your Raspberry Pi, you must enable I2C in settings before using it. Therefore, open a terminal on your Raspberry Pi and run the following commands.

1. Run `sudo raspi-config`
2. Use the down arrow to select 5 Interfacing Options
3. Arrow down to P5 I2C.
4. Select yes when it asks you to enable I2C
5. Also select yes if it asks about automatically loading the kernel module.
6. Use the right arrow to select the Finish button.
7. Select yes when it asks to reboot.
8. After reboot , run `sudo apt-get install -y i2c-tools`
9. Run `sudo apt-get install python-smbus`
10. Run `sudo pip3 install pyserial`

9.2 I2C Wiring Guide

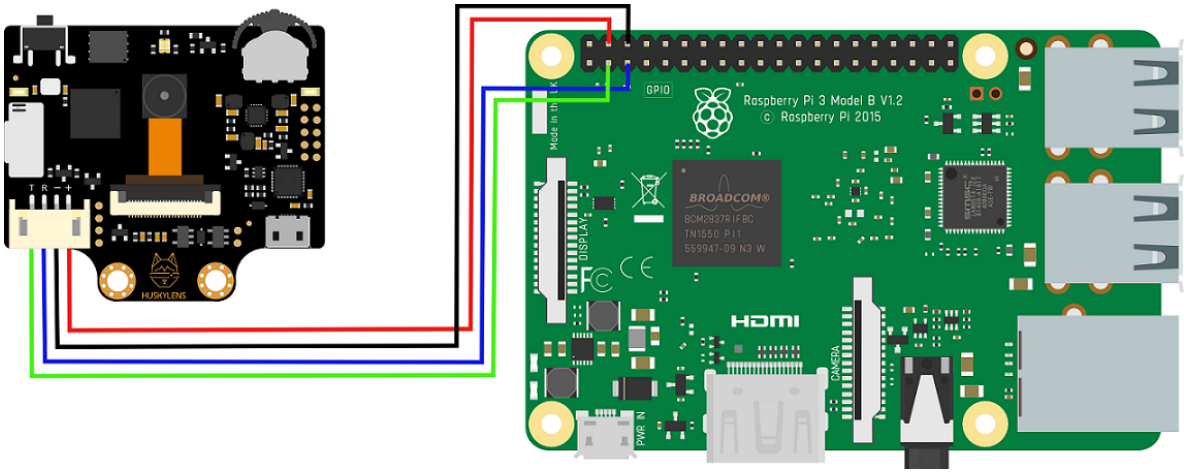
The primary protocol for communication between the HuskyLens and the Raspberry Pi is I2C. This requires you to use the 4-Pin connector to wire ground, power, SDA, and SCL. To read more about how I2C works, please check out the following link: https://en.wikipedia.org/wiki/I%C2%B2C

Pin Outline(Raspberry Pi)

Pin Outline(HUSKYLENS)

Label	Pin Function	Description
T	SDA	serial clock line
R	SCL	serial data line
-	GND	negative pole of power supply(0V)
+	VCC	positive pole of power supply(3.3~5.0V)

Connection Diagram



Tips: HuskyLens consumes heavy current, up to 3.3V 320mA, 5V 230mA or more. We recommand connecting HuskyLens to the 5V power supply pins on your Raspberry Pi, which can supply enough power to HuskyLens.

9.3 Coding Guide

1. Download the HuskyLens Python Library.
2. Place the huskylensPythonLibrary.py in your projects folder.
3. In your python file (e.g. test.py), import the using library

```
from huskylensPythonLibrary import HuskyLensLibrary
```

4. Init the HuskyLens

```
my_Var= HuskyLensLibrary("I2C","",address=0x32)
```

5. Now begin calling functions !

```
# Check if HuskyLens can recieve commands
print(my_Var.command_request_knock())
# Get all the current blocks on screen
blocks=my_Var.command_request_blocks()
# Print the data
print(blocks)
```

9.4 API Introduction

```
command_request()
     => Return all data

command_request_blocks()
     => Return all blocks on the screen

command_request_arrows()
     => Return all arrows on the screen(only in line tracking mode)

command_request_learned()
     => Return all learned objects on screen

command_request_blocks_learned()
```

```
     => Return all learned blocks on screen

command_request_arrows_learned()
     => Return all learned arrows on screen(only in line tracking mode)

command_request_by_id(idVal)
     *idVal is an integer
     => Return the object with id of idVal

command_request_blocks_by_id(idVal) *idVal is an integer
     *idVal is an integer
     => Return the block with id of idVal

command_request_arrows_by_id(idVal) *idVal is an integer(only in line tracking
mode)
     *idVal is an integer
     => Return the arrow with id of idVal

command_request_algorthim(ALG_NAME)
    * ALG_NAME is a string whose value can be the following
        "ALGORITHM_OBJECT_TRACKING"
        "ALGORITHM_FACE_RECOGNITION"
        "ALGORITHM_OBJECT_RECOGNITION"
        "ALGORITHM_LINE_TRACKING"
        "ALGORITHM_COLOR_RECOGNITION"
        "ALGORITHM_TAG_RECOGNITION"
        "ALGORITHM_OBJECT_CLASSIFICATION"

command_request_knock()
    => Returns "Knock Recieved" on success
```

9.5 Project 1: Read Position Data

In this project, we use the Raspberry Pi to read the position data of the object from the HuskyLens. The communication protocol is I2C.

Requirements

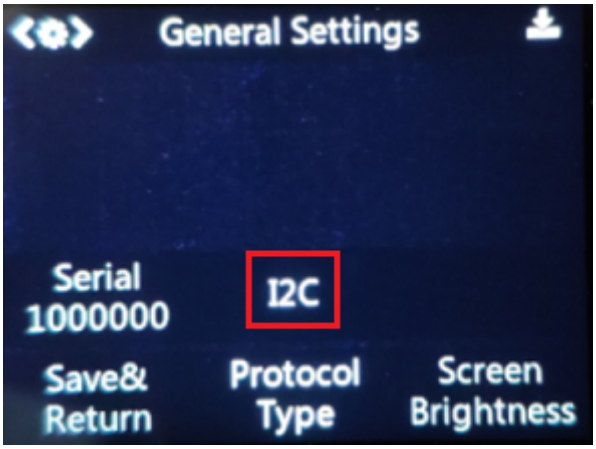
- **Hardware**
  - Raspberry Pi 3 Model B+ (or similar) x 1
  - HUSKYLENS x 1
  - M-M/F-M/F-F Jumper wires
- **Software**
  - HUSKYLENS Python Library

Connection Diagram

Refer the chapter "9.2 I2C Wiring Guide".

HuskyLens Protocol Setting

You need to set the protocol type of HuskyLens. The protocol should be I2C.



Sample Code

The following code is in the test.py. Click here to view and download it.

```
# Import the library
from huskylensPythonLibrary import HuskyLensLibrary
# Initlialize the HuskyLens
test = HuskyLensLibrary("I2C","",address=0x32)
print("First request a knock: {}".format(test.command_request_knock()))

# Change to face recognition algorhtim
test.command_request_algorthim("ALGORITHM_FACE_RECOGNITION")

# Display a simple menu where you can call every function in a loop!
ex=1
print("""
        Menu options:
        1) command_request()
        2) command_request_blocks()
        3) command_request_arrows()
        4) command_request_learned()
        5) command_request_blocks_learned()
        6) command_request_arrows_learned()
        7) command_request_by_id() ***format 7 ID_VAL***
        8) command_request_blocks_by_id() ***format 8 ID_VAL***
        9) command_request_arrows_by_id() ***format 9 ID_VAL***
        10) Exit
        """)
while(ex==1):
    v=input("Enter cmd number:")
    numEnter=v
    if(numEnter=="10"):
        ex=0
    v=int(v[0])
    if(v==1):
        print(test.command_request())
    elif(v==2):
        print(test.command_request_blocks())
    elif(v==3):
        print(test.command_request_arrows())
    elif(v==4):
        print(test.command_request_learned())
    elif(v==5):
        print(test.command_request_blocks_learned())
```

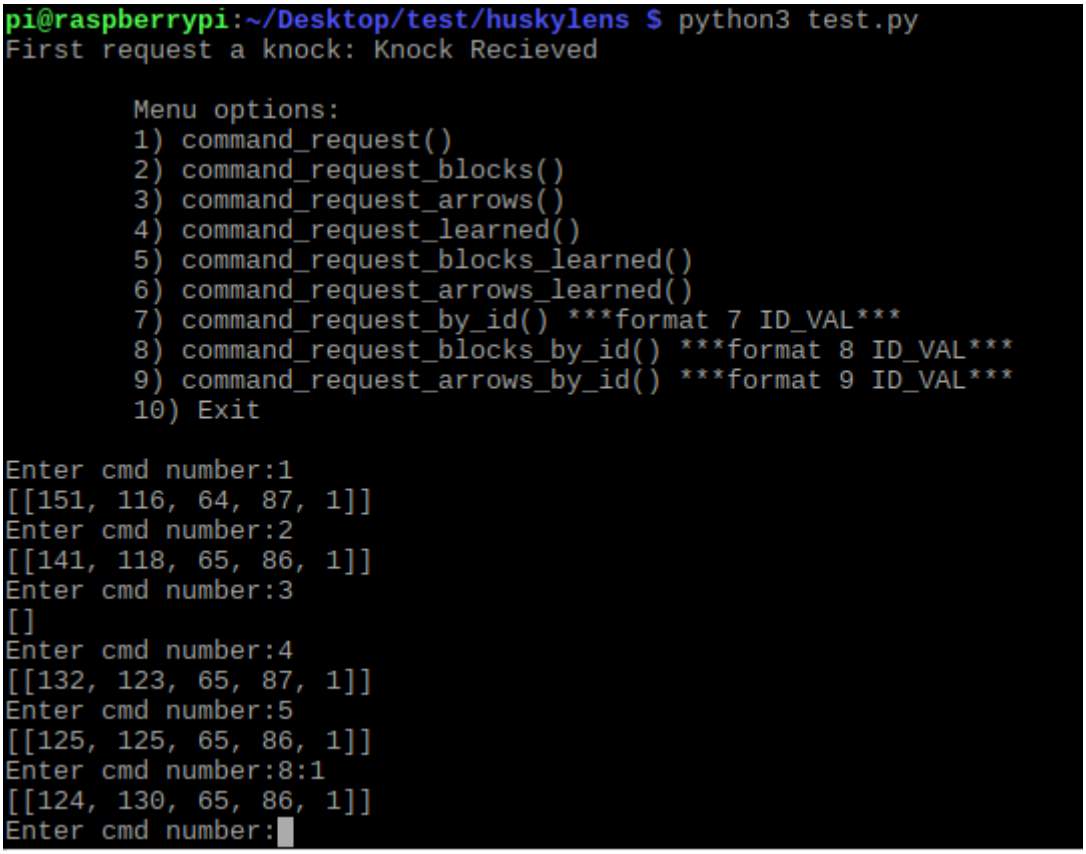
```
    elif(v==6):
        print(test.command_request_arrows_learned())
    elif(v==7):
        print(test.command_request_by_id(int(numEnter[2:])))
    elif(v==8):
        print(test.command_request_blocks_by_id(int(numEnter[2:])))
    elif(v==9):
        print(test.command_request_arrows_by_id(int(numEnter[2:])))
```

Operations and Expected Results

1. Run the following code in the terminal on your Raspberry Pi.

```
python3 test.py
```

2. Let your HuskyLens learn a new thing first, e.g. your face. You can refer to the previous chapters of this tutorial.
3. Point the HuskyLens at your face. Then input the command number in your terminal. You will get the results as follows:



The results provide the x, y coordinates, width, height of the frame on the screen, and the ID of the object. The format is shown as below:

```
[X Center of Block, Y Center of Block, Width of Block, Height of Block, Index of the learned items]
```

10. micro:bit Tutorial

In this chapter, we will use micro:bit board to read data from the HuskyLens. The communication protocol is I2C. We adopt Mind+ and MakeCode, then combine with several project cases to demonstrate the usage.

Now we adopt Mind + for demonstration first.

10.1 Mind+ Introduction

Mind+ is a Scratch 3.0-based programming tool, which allows you to build a program by dragging and snapping coding blocks. With tons of tutorials, sample projects and a large community, it is one of the best tools for you to learn programing from absolutely zero!

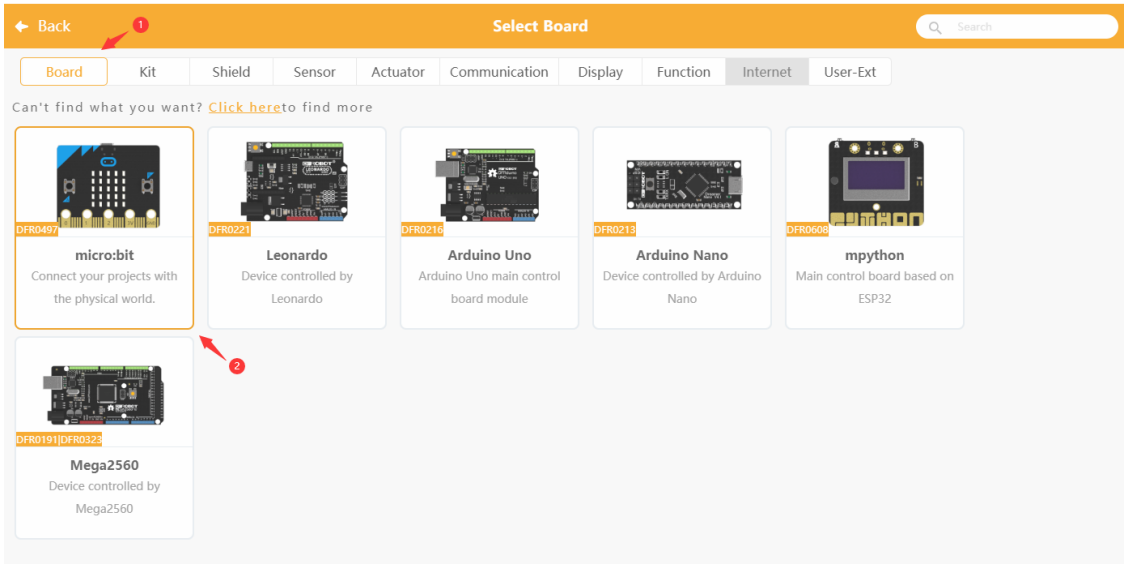
Mind+ supports a wide range of hardware including Arduino, micro:bit or even a series of ESP32-based educational microcontrollers.

Please view Mind+ official site http://mindplus.cc to download the latest Mind+.

10.2 Load HuskyLens Extension

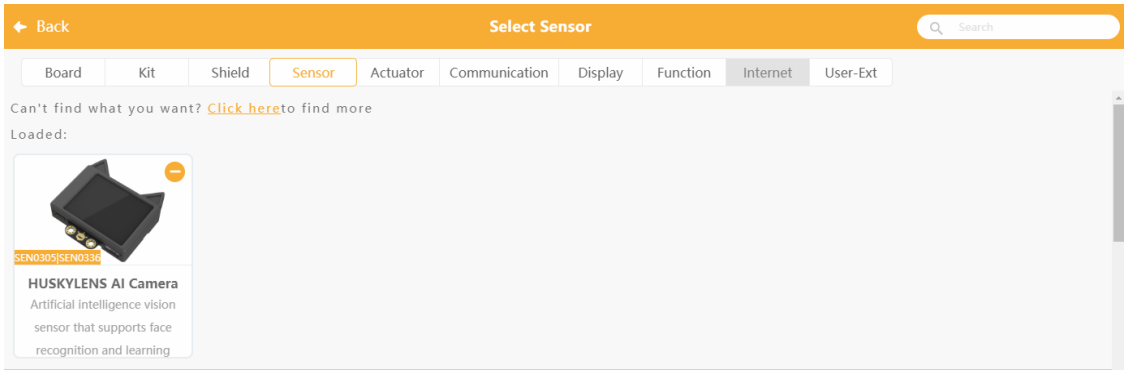
1. In the upper right corner of the Mind+ window, dial the switch to the offline mode.

2. Click the `Extensions` button in the lower left corner to view the extensions window.
3. Select a main control board. Here we select the micro:bit.

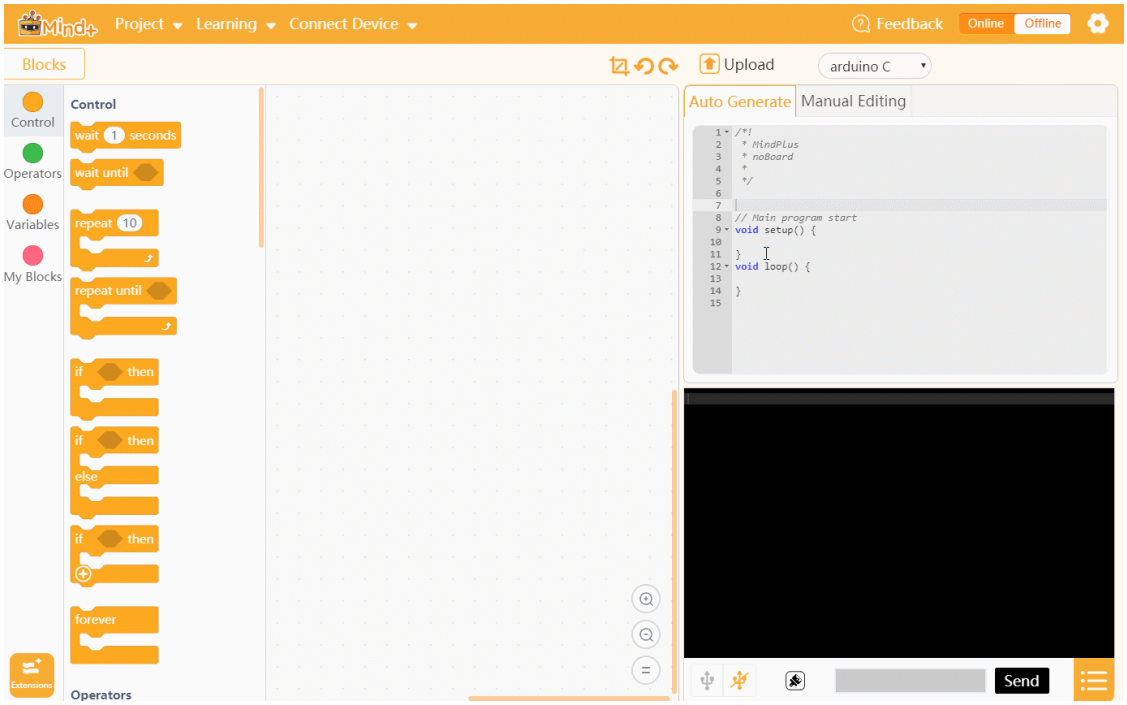


4. After selecting the main control board, the corresponding extension labels become selectable. Then click `Sensor` label, find HuskyLens. Click it to load.

   You can enter the sensor's name in the search bar in upper right corner when there are many sensors listed in the menu.



5. After selecting the main control board and extensions, click `Back` button in the upper left corner to return to the programming window.



10.3 Mind+ Project 1: Face Recognition

This chapter demonstrates how to connect HuskyLens to the micro: bit board, then the micro: bit board reads the face recognition results from HuskyLens. If HuskyLens recognizes you (the learned face), the dot-matrix screen of the micro: bit displays a smiling face, otherwise it displays a crying face.

The communication protocol between HuskyLens and micro: bit is I2C.

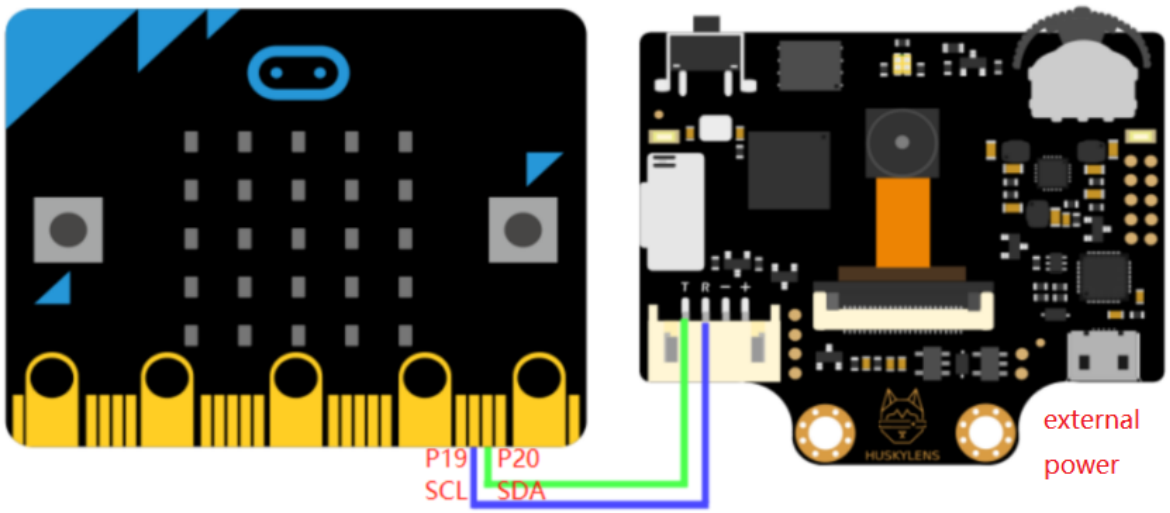
Requirements

- Hardware
  - micro:bit board x 1
  - micro:bit expansion board x 1
  - HUSKYLENS x 1
  - M-M/F-M/F-F Jumper wires
- Software
  - Mind+
  - HUSKYLENS Extension: Mind+ built-in

Connection Diagram

The following picture is only for reference when wiring. The R and T pins of HuskyLens (their functions are SCL and SDA here) are connected to the SCL (P19) and SDA (P20) pins of the micro: bit respectively. The communication protocol between HuskyLens and micro: bit is I2C.

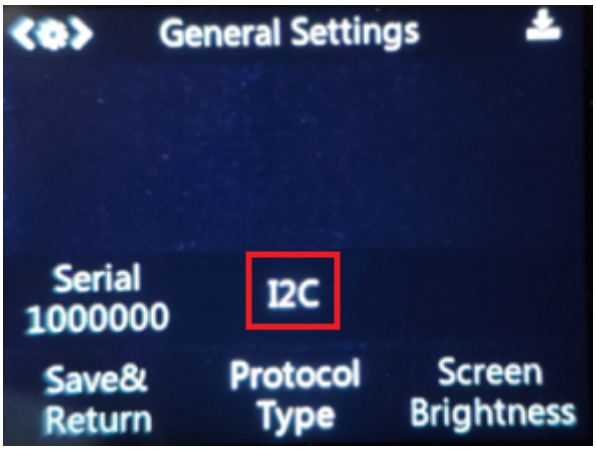
A micro: bit expansion board is recommended for simplify wiring.



Tips: HuskyLens consumes heavy current, up to 3.3V 320mA or more. The mciro: bit board is not enough to supply power. Therefore, external power supply is required. You can connect the external power supply to the external power connector of the micro: bit expansion board or the USB connector of HuskyLens.

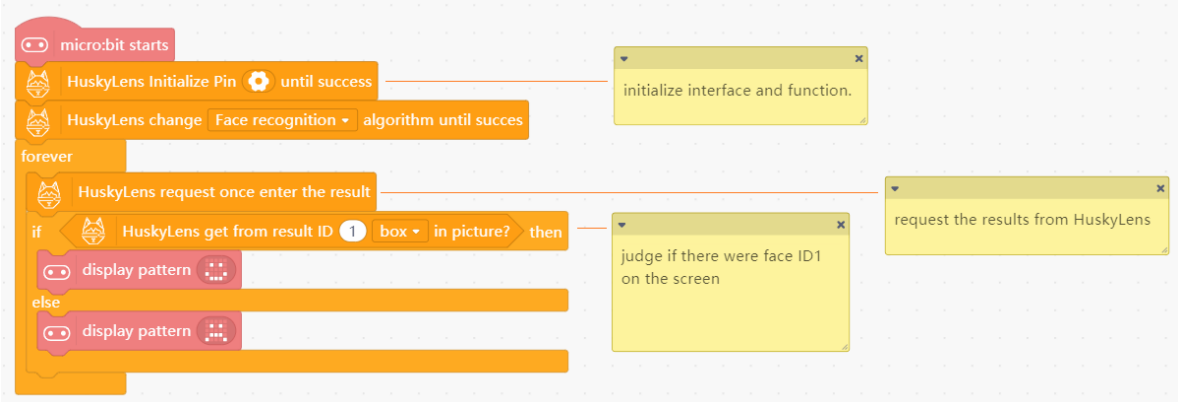
HuskyLens Protocol Setting

You need to set the protocol type of HuskyLens. The protocol should be I2C. Of course, your can adopt the auto detect protocol, which is easy-to-use and convenient.



Sample Code

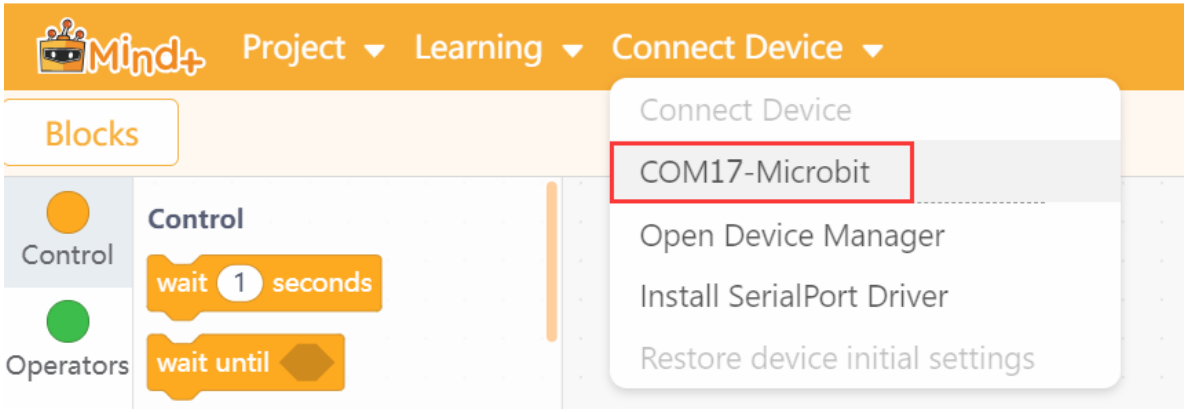
Drag and snap the coding blocks to program, and make a simple face recognition projects. The sample code is shown below.



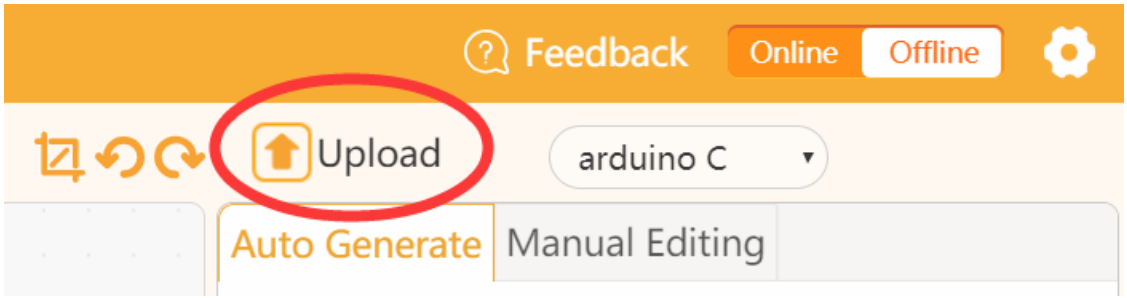
Operations and Expected Results

1. Click the `Connect Device` button, then select the COM port corresponding to the microbit.

   If no COM port is found or you use mcirobit for the first time, please click the `Install Serialport Driver` button to install the driver with one click.



2. Click the `Upload` button to upload the sample code to your micro:bit board.



3. Let your HuskyLens learn your face first. You can refer to the chapter 7.1 of this tutorial.
4. When HuskyLens recognizes your face, the dot-matrix screen on the micro: bit board will show a smiling face. If it were not your face, or no face appeared, it would display a crying face.

10.4 Coding Block Introduction

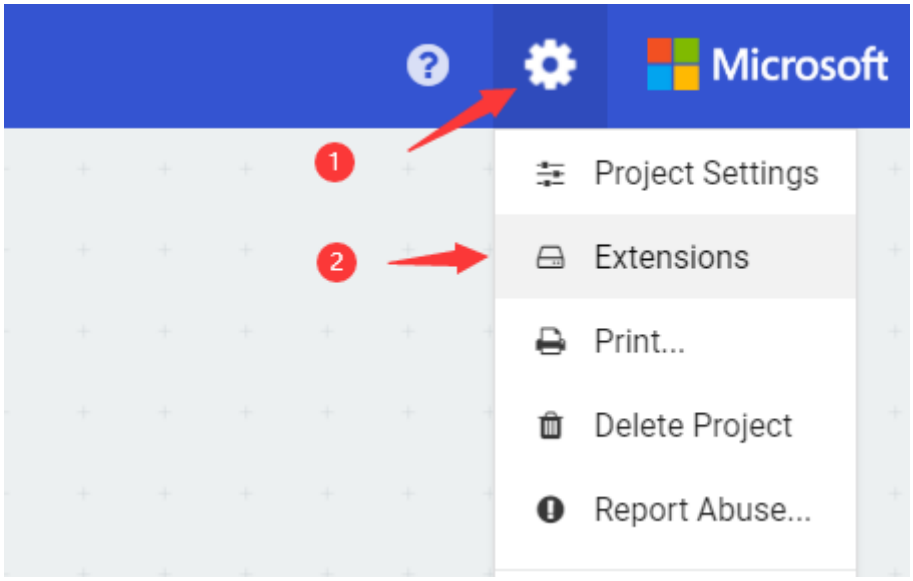
We adopt makecode for demonstration below.

10.5 MakeCode Introduction

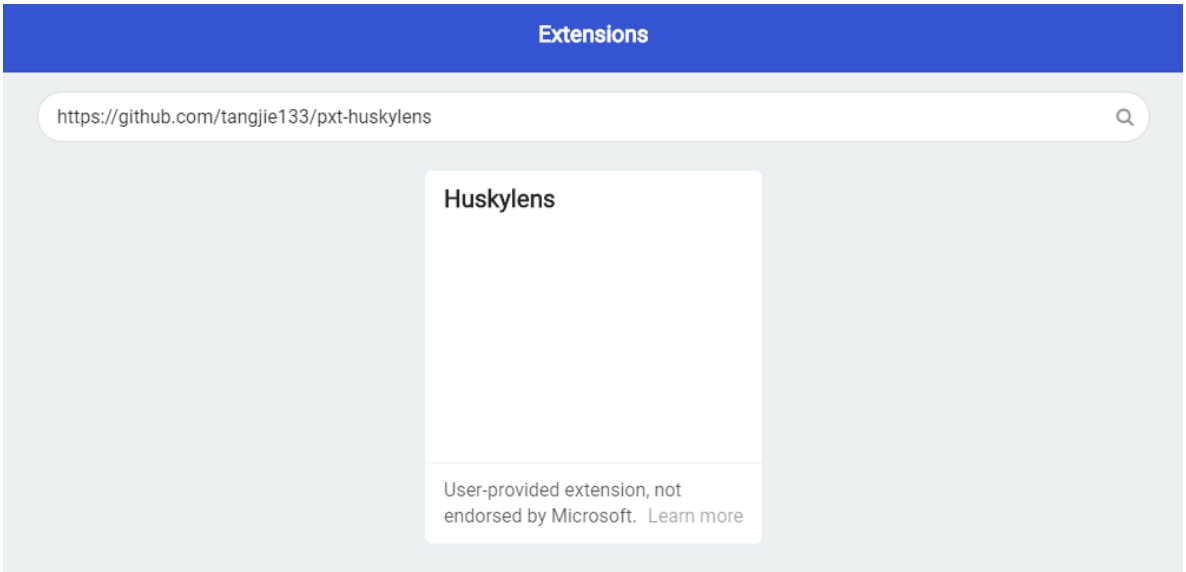
Microsoft MakeCode is a free, open source platform for creating engaging computer science learning experiences that support a progression path into real-world programming. Click here to view the MakeCode for micro:bit.

10.6 Load HuskyLens Extension

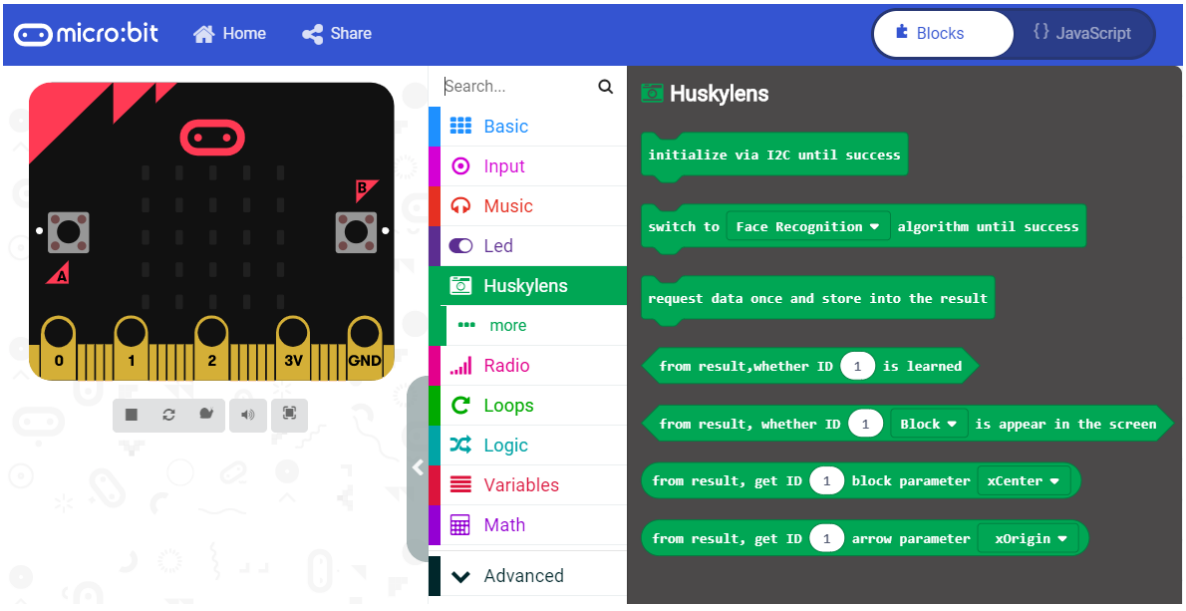
1. Create a new project in MakeCode web version, and then click the "More..." button (gear icon) at the top right and select "Extensions" from its drop-down menu to open the extension page.



2. Enter `https://github.com/tangjie133/pxt-huskylens` in the search bar, then click the search button (the magnifying glass button on the right of the search bar) , you will see the HuskyLens extensions. Then click it to load the HuskyLens extension into the MakeCode.



3. In the programming page, you can see the Huskylens module.



10.7 Project 1: Face Recognition

This chapter demonstrates how to connect HuskyLens to the micro: bit board, then the micro: bit board reads the face recognition results from HuskyLens. If HuskyLens recognizes you (the learned face), the dot-matrix screen of the micro: bit displays a smiling face, otherwise it displays a crying face.

The communication protocol between HuskyLens and micro: bit is I2C.

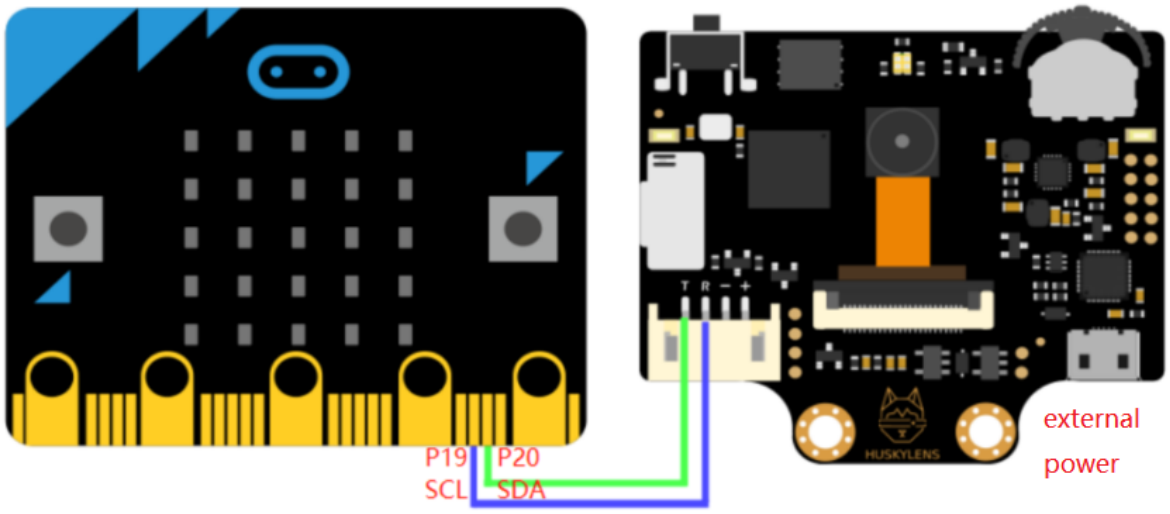
Requirements

- **Hardware**
  - micro:bit board x 1
  - micro:bit expansion board x 1
  - HUSKYLENS x 1
  - M-M/F-M/F-F Jumper wires
- **Software**
  - Microsoft MakeCode for micro:bit
  - HUSKYLENS MakeCode Extension

Connection Diagram

The following picture is only for reference when wiring. The R and T pins of HuskyLens (their functions are SCL and SDA here) are connected to the SCL (P19) and SDA (P20) pins of the micro: bit respectively. The communication protocol between HuskyLens and micro: bit is I2C.

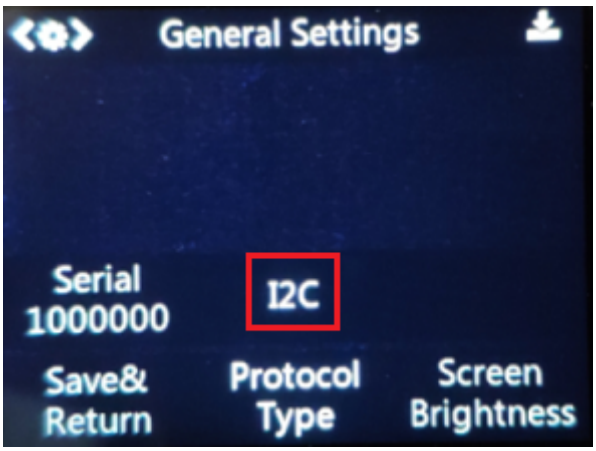
A micro: bit expansion board is recommended for simplify wiring.



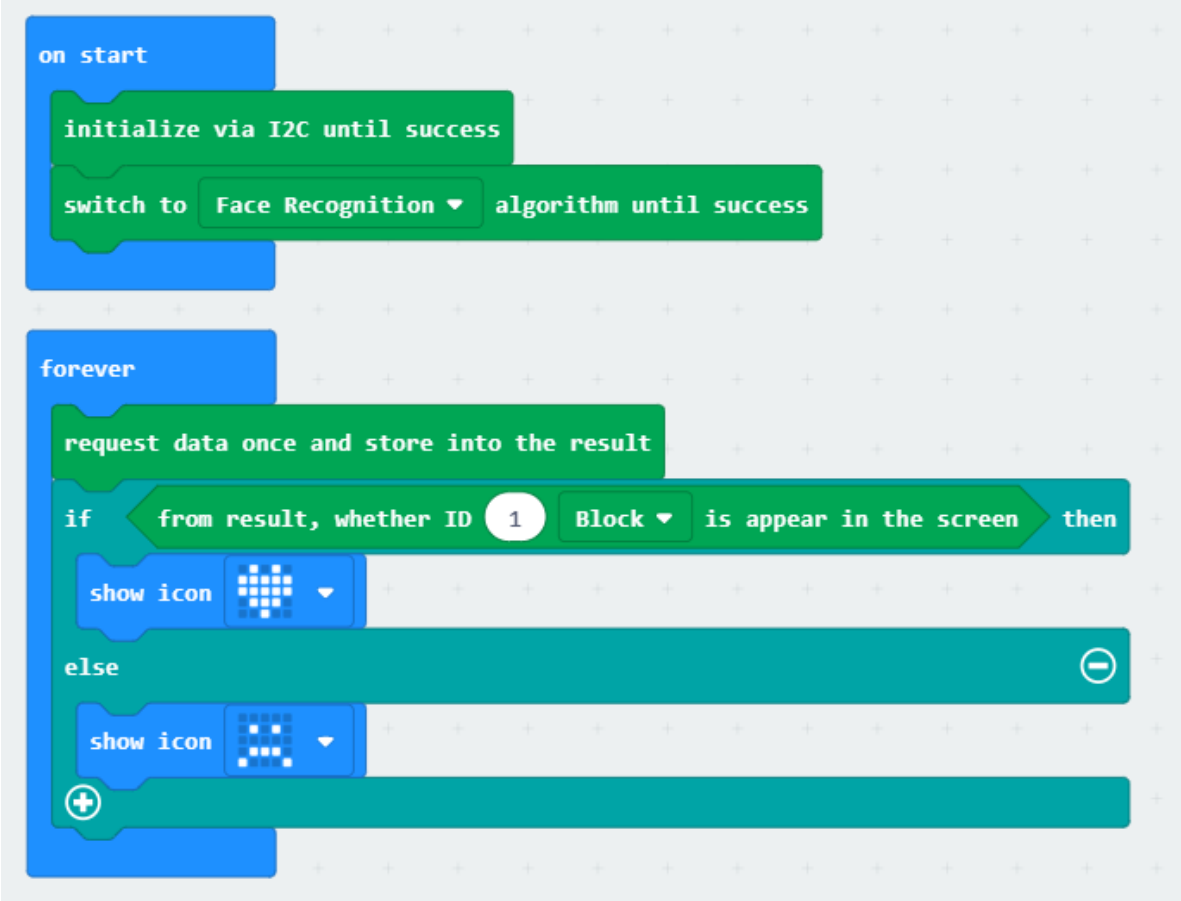
Tips: HuskyLens consumes heavy current, up to 3.3V 320mA or more. The mciro: bit board is not enough to supply power. Therefore, external power supply is required. You can connect the external power supply to the external power connector of the micro: bit expansion board, or the HuskyLens USB connector.

HuskyLens Protocol Setting

You need to set the protocol type of HuskyLens. The protocol should be I2C. Of course, your can adopt the auto detect protocol, which is easy-to-use and convenient.



Sample Code



Operations and Expected Results

1. Upload the above codes to the micro: bit board.
2. Refer to the previous chapter which explaining the face recognition function(chapter 7.1), let your HuskyLens learn a face, such as your face.
3. When HuskyLens recognizes your face, the dot-matrix screen on the micro: bit board will show a smiling face. If it were not your face, or no face appeared, it would display a crying face.

10.8 Coding Block Introduction

11. More Documents

- Arduino Library(github)
- Raspberry Pi Python Library(github)

- micro:bit Makecode Library(github)
- Protocol Document
- Tag Pictures
- Color Block Pictures
- 3D model file(.stp)
- Download .NET Framework 4.7.1 for K-Flash
- Download WIKI Page(PDF)

 Get **HUSKYLENS** from DFRobot Store or **DFRobot Distributor**.

Turn to the Top