DFROBOT
DRIVE THE FUTURE

Maqueen Plus Advanced Tutorial

Mind+

- Get Known to Hardware
- Vehicle Sharing
- Auto-Tracking Vehicle
- Fixed-Point Transportation
- Self Driving Truck
- Out of the Maze
- Relay Transport

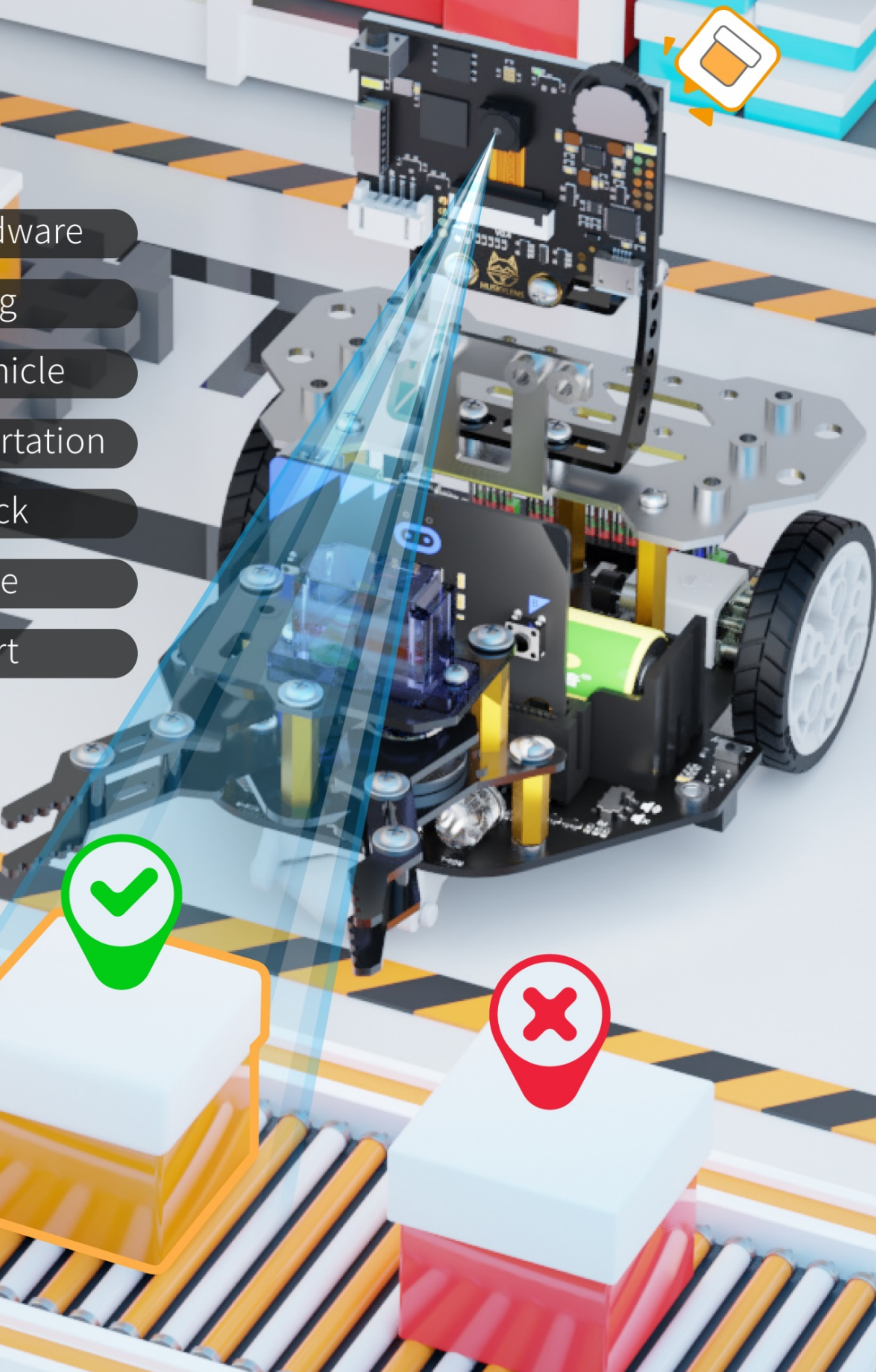


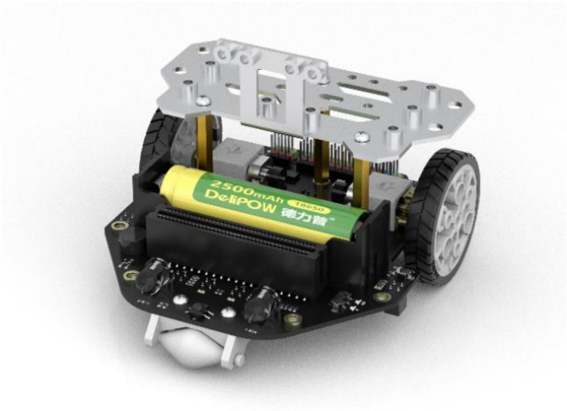
Table of Contents

Project1: Get Known to Hardware

Maqueen Plus

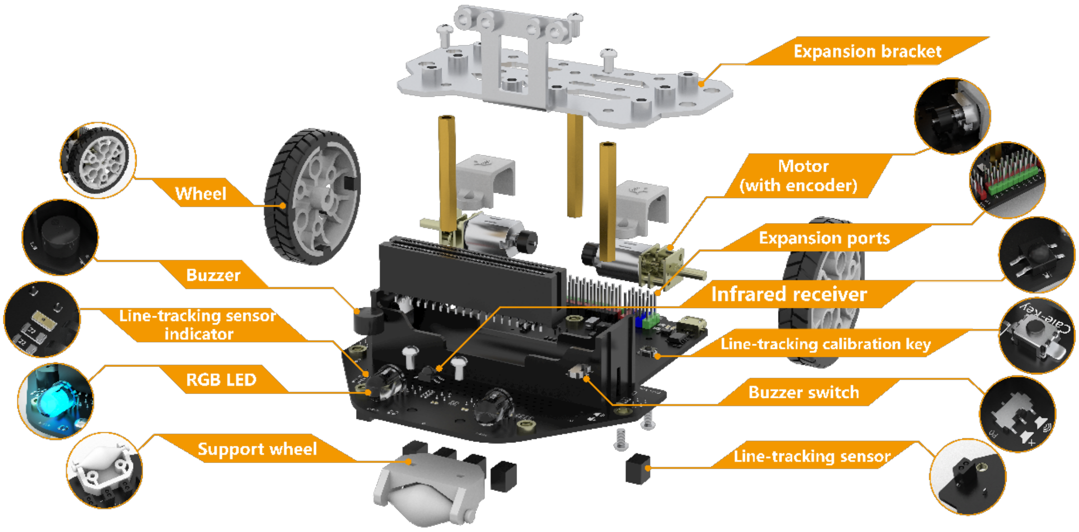
Introduction

Maqueen Plus is a smart programmable educational robot. It supports Mind+ and Makecode programming platforms, on which we can program Maqueen Plus to realize awesome functions by simply dragging and snapping the graphical blocks. Follow Maqueen Plus to enter the world of robotic, and while learn something about coding in playing!

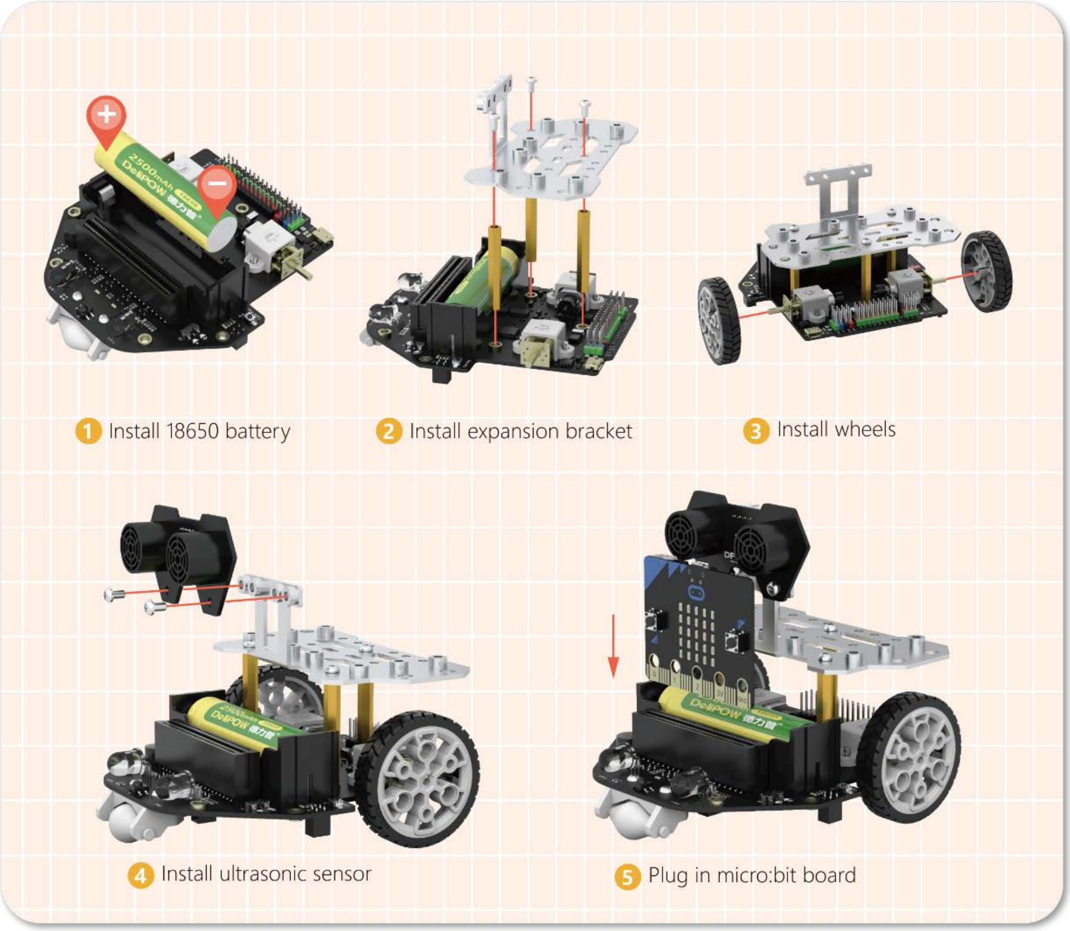


Function

The Maqueen Plus is equipped with a variety of sensors, including line tracking sensor, infrared receiving sensor, encoder, metal motor, buzzer, and RGB lights, as well as multiple expansion ports for supporting more electronic modules.



Assembly

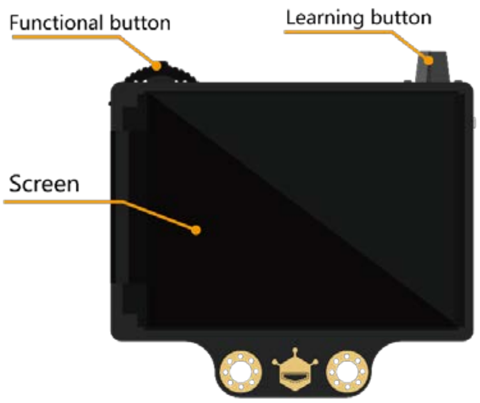


1 Install 18650 battery
2 Install expansion bracket
3 Install wheels
4 Install ultrasonic sensor
5 Plug in micro:bit board

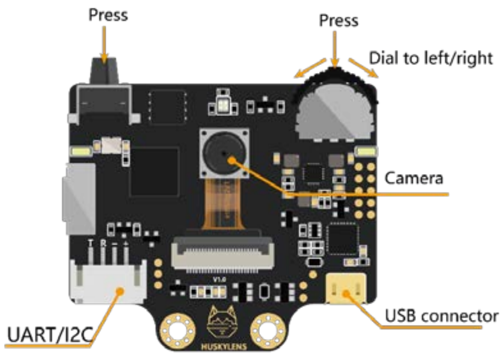
HUKSYLENS

Appearance

HUSKYLENS is a “WYSIWYG” AI vision sensor. It is called as husky image recognition in the maker circle. The sensor can directly display the result recognized by the camera.



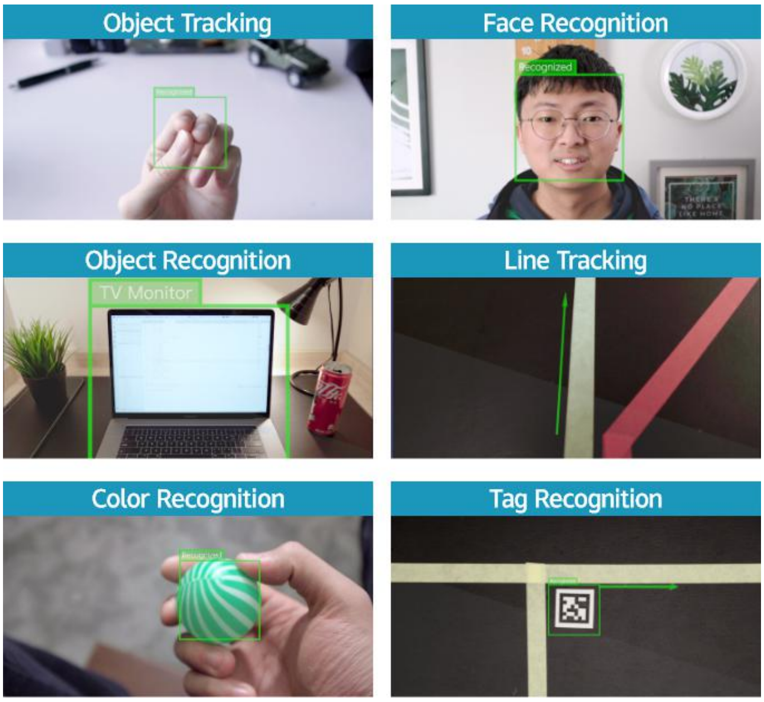
Front

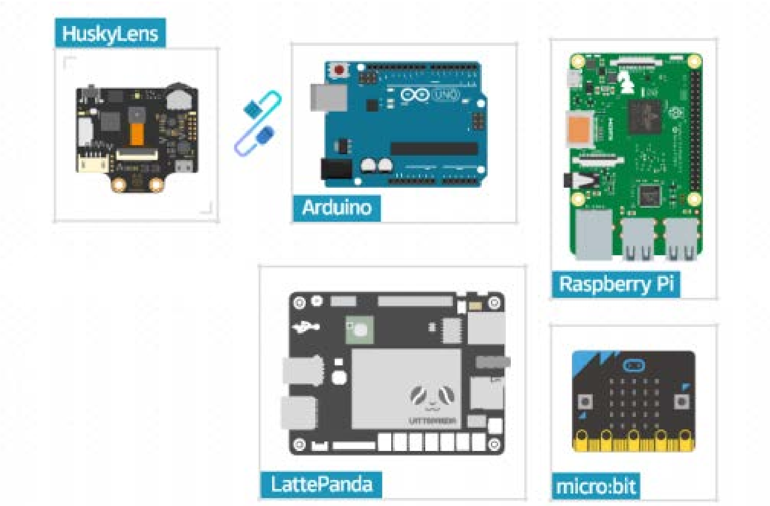


Back

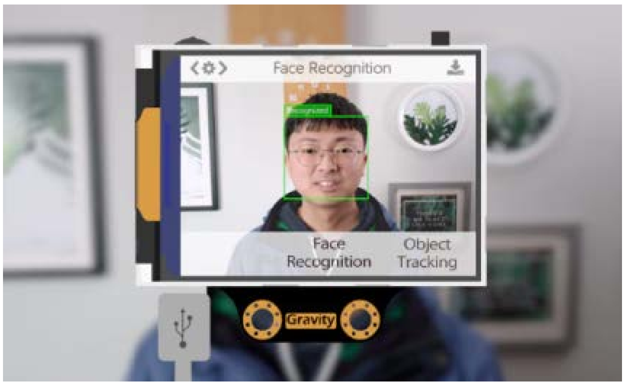
Function

HUSKYLENS is an easy-to-use AI machine vision sensor. Integrated with new AI chip and built-in machine learning technology. it is able to realize the functions of face recognition, object recognition, object tracking, color recognition, line tracking, QR code recognition, etc. The on-board UART/I2C port can be seamlessly connected with mainstream controllers, such as Arduino, micro:bit, LattePanda, Raspberry Pi, which are widely applied in AI education, STEAM education and maker field.



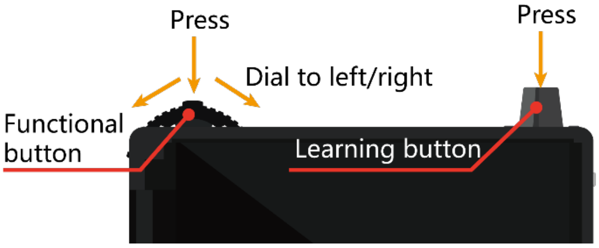


The HUSKYLENS comes with a 2.0" IPS display, so you don' t need the help of a computer when setting the parameters. The progress and result will be displayed on the screen directly. What you see is what you get, convenient to use.



Basic Operations

With only two buttons, HUSKYLENS can achieve all the operations: function buttons to switch six algorithms, learning button to achieve AI training and learn new things. Here are the basic operations of these two buttons:



- Function button: dial to the left/right or press;
- Learning button: short press or long press;
- Dial the function button to the left/right to switch between the 6 functions;

- Short press the “learning button” to learn a certain object. The ID of the learned object will be displayed on the screen.
- Long press the “learning button” to learn a certain object from different angles and distances.
- If the object has been studied before, then press the "learning button" to tell HUSKLENS to forget what it has learned under the current function.

Check the Firmware

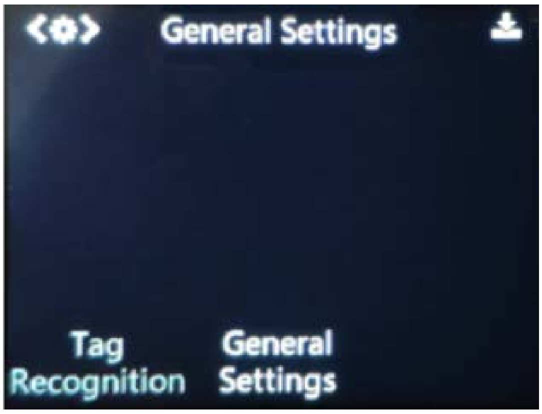
The firmware is the equivalent of an operating system of HuskyLens, and it will be updated as programmers update the algorithm. So how do we check the firmware version of our camera? Let's start the first use of HUSKYLENS according to step1 to 4 below!

Step1: Connect the Power

HUSKYLENS comes with an independent USB power supply port. Connect the USB cable and then it will work.

Step2: Select “General Settings”

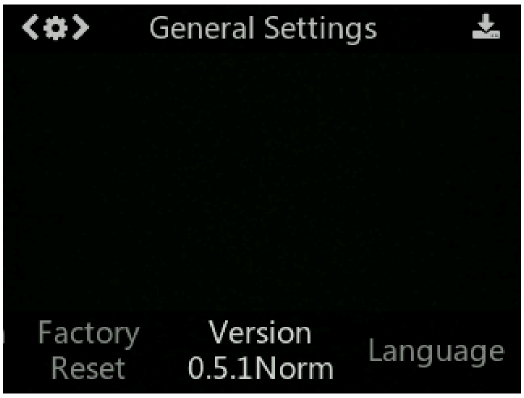
Dial the “function button” to the left or right till “General Settings” shows on the display.



Step3: Expand the Submenu

Here are submenus under the six functions and general settings, and the detailed parameter settings can be carried out. Long press the “function button” to enter the submenu of the current function; dial the function button to the left or right and press the “function button” to set the relevant parameters.

The picture below is the submenu of "General Settings".



The way to check the firmware: After entering the submenu of the “General Settings”, dial the “function button” to the right till the “Version” shows at the bottom of the display, and then you can check the firmware version.

Switch language: HUSKYLENS supports Chinese and English. Dial the “function button” to the right till “Language” appears at the bottom of the screen. Press the function button to switch and the select language.

Note: Under the submenu, only "Function button" is supported. Dial left and right or short-press the "Function button" to set relevant parameters.

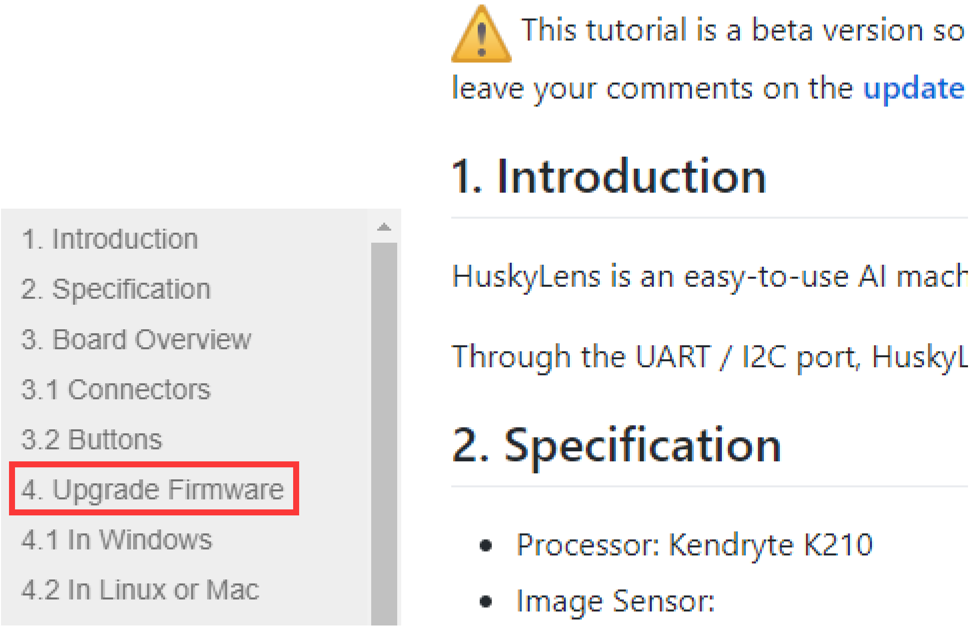
The other parameters in the submenu will be learned in the following project. You only need to know how to open the submenu when using it for the first time.

Step4: Update the Firmware

Click the website to check the latest firmware version:

https://wiki.dfrobot.com/HUSKYLENS_V1.0_SKU_SEN0305_SEN0336#target_5

After opening the web page, click "4. Update Firmware" in the red box as below to complete the firmware update according to the operation steps.



1. Introduction
2. Specification
3. Board Overview
3.1 Connectors
3.2 Buttons
4. Upgrade Firmware
4.1 In Windows
4.2 In Linux or Mac

This tutorial is a beta version so leave your comments on the **update**

1. Introduction

HuskyLens is an easy-to-use AI mach

Through the UART / I2C port, HuskyL

2. Specification

- Processor: Kendryte K210
- Image Sensor:

Basic Application

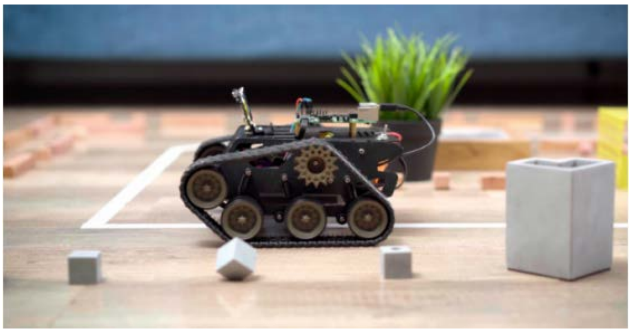
With HUSKYLENS, your project will have many new ways of interaction for you to explore, such as interactive gesture control, self-driving cars, robot eyes, smart tracker and other projects. Come and try!

Interactive Gesture Control

Learn a specific gesture by using the object tracking function of HUSKYLENS, then the sensor can recognize the learned gesture and provide its location data. With this data, creating an amazing interactive project has never been easier.

Self-Driving Cars

HUSKYLENS provides a new way of line tracking. With only selecting the mode of line tracking, you can let HUSKYLENS learn lines, and then it can start line tracking.



Robot Eyes

HuskyLens can be the eyes of robots. which allows your robot to recognize you, understand your hand gesture commands, or help you put stuff in order and so on. With Huskylens, nothing is impossible!

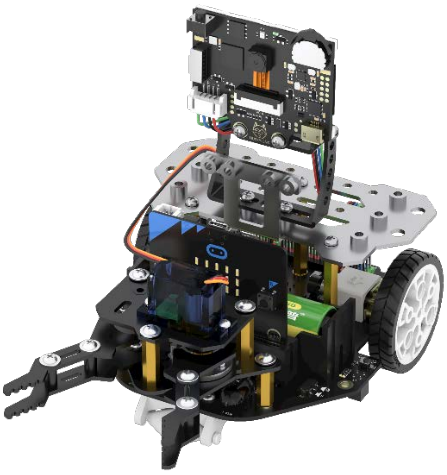
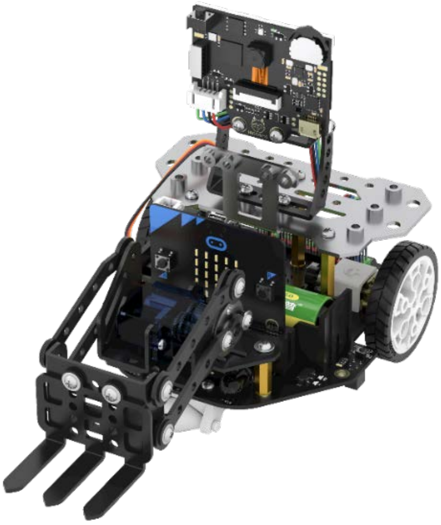
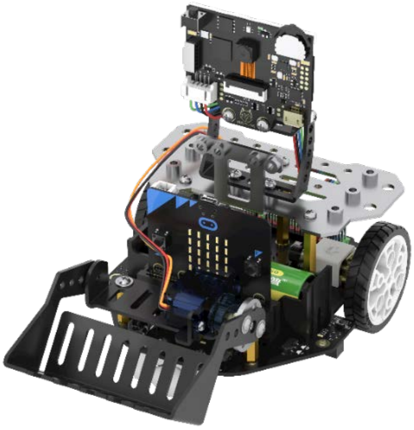
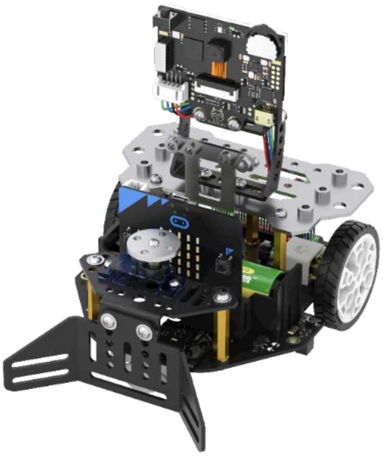
Smart Tracker

Use the object recognition function of HUSKYLENS to track an object. For example, let it track the location of your pet.

Maqueen Mechanic Kit

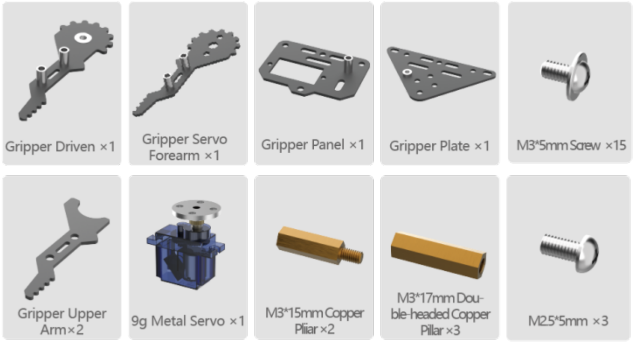
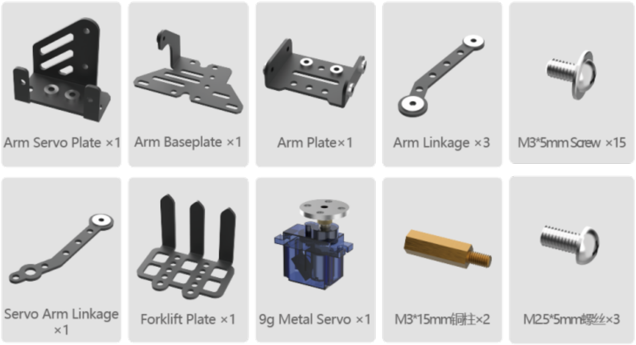
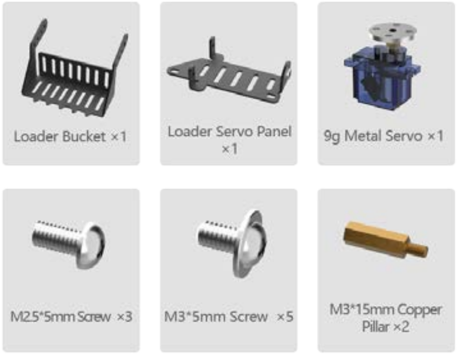
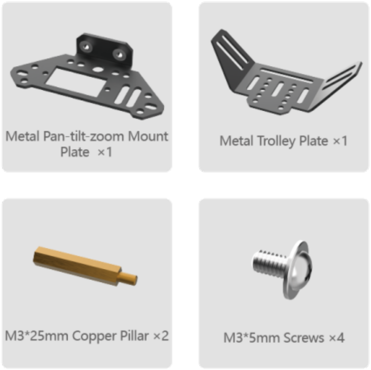
Product Introduction

The Maqueen Mechanic Kit is a set of machine accessories designed for Maqueen Plus. There are four different shapes we can build with this mechanic kit: mechanical Beetle, Forklift, Loader, and Push.

	
Maqueen Mechanic-Beetle	**Maqueen Mechanic-Forklift**
	
Maqueen Mechanic-Loader	**Maqueen Mechanic-Push**

Accessory

Before using the Maqueen mechanic kit to build projects, we have to check if we have all the accessories.

 Gripper Driven ×1 Gripper Servo Forearm ×1 Gripper Panel ×1 Gripper Plate ×1 M3*5mm Screw ×15 Gripper Upper Arm×2 9g Metal Servo ×1 M3*15mm Copper Pliar ×2 M3*17mm Double-headed Copper Pillar ×3 M2.5*5mm ×3	Maqueen Mechanic-Beetle
 Arm Servo Plate ×1 Arm Baseplate ×1 Arm Plate×1 Arm Linkage ×3 M3*5mm Screw ×15 Servo Arm Linkage ×1 Forklift Plate ×1 9g Metal Servo ×1 M3*15mm铜柱×2 M2.5*5mm螺丝×3	**Maqueen** **Mechanic-Forklift**
 Loader Bucket ×1 Loader Servo Panel ×1 9g Metal Servo ×1 M2.5*5mm Screw ×3 M3*5mm Screw ×5 M3*15mm Copper Pillar ×2	**Maqueen** **Mechanic-Loader**
 Metal Pan-tilt-zoom Mount Plate ×1 Metal Trolley Plate ×1 M3*25mm Copper Pillar ×2 M3*5mm Screws ×4	**Maqueen** **Mechanic-Push**

From learning about the Maqueen Plus, HUSKYLENS, and the Maqueen Mechanic Kit, do you have any idea about your project? If you don't, continue following this tutorial!

Project 2: Vehicle Sharing

With the development of society, all kinds of sharing transportation systems are shown in our daily life, such as, sharing bikes and sharing vehicles.

The promotion of the concept of "sharing", on the one hand, satisfies the consumers driving demand; on the other hand, it avoids the waste caused by idle vehicles and ineffective utilization of resources.

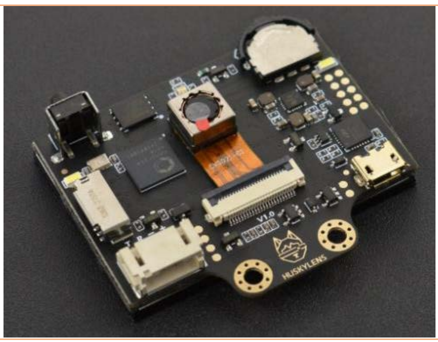
Here we are going to make a sharing vehicle with Maqueen Plus and HUSKYLENS.

Function

This project aims to realize face unlock on Maqueen Plus with the face recognition function of HUSKYLENS. The mileage is calculated by the formula after counting the revolution the wheel turns via encoders. When you don' t want to use the vehicle anymore, just press button A to return it. Meanwhile, the OLED screen will display mileage, duration, and cost.

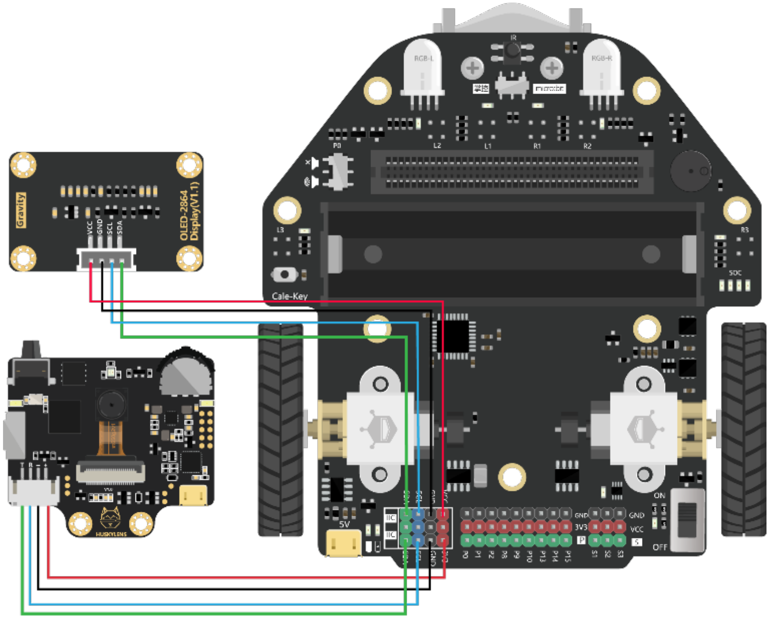
Bill of Materials

micro:bit ×1	**Maqueen Plus ×1**

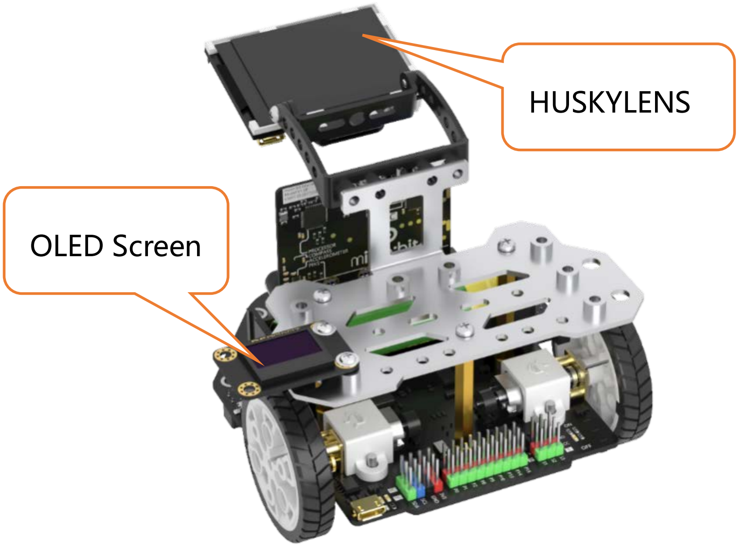
	
HUSKYLENS ×1	**OLED ×1**
HUSKYLENS Installing Bracket	

Hardware Connection

Connection Diagram: Both the HUSKYLENS sensor and the OLED screen use I2C interfaces, so you need to pay attention to the wiring order and avoid wrong connections.



Assembly Diagram: The HUSKYLENS sensor comes with a bracket structure which can be screwed on the Maqueen Plus. And the sensor can be freely adjusted among certain angles. The OLED display can also be screwed to the Maqueen Plus car.



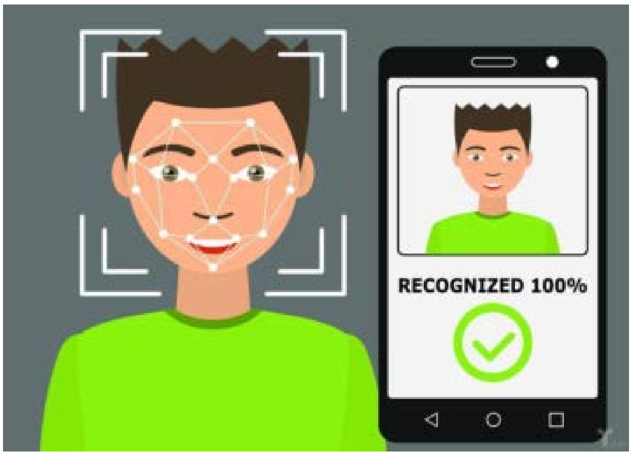
Note: To install the HUSKYLENS, the ultrasonic sensor needs to be removed.
Point the HUSKYLENS straight ahead under all functions except line tracking.

Knowledge Field

Like other biological characteristics of the human body (fingerprint, iris, etc.), human face is inherent, unique and unrepeatable. It is the first kind of image that has been studied, and also the widely used one in the field of computer vision. This project is making use of the face recognition of HUSKYLENS.

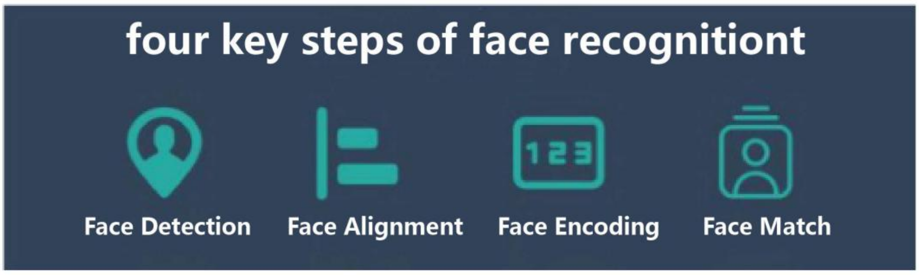
1. What is Face Recognition?

Face recognition is a biological recognition technology for identity recognition based on information about facial features. It uses a camera to collect images and videos containing human faces and automatically detects and tracks the faces, then performs a series of analytical techniques of the detected faces.



2. Operating Principles of Face Recognition

There are four key steps of face recognition:



Here is a brief explanation of these four steps.

Face Detection: detect the face positions and select them by frames.

Face Alignment: recognize faces from different angles by locating the features.

Face Encoding: extract face information and make it understandable for a computer.

Face Match: match the face information with the existing database to get a similarity score and give out the matching result.

3. Application Scenarios of Face Recognition

Access Control System: through face recognition, the identity of the person entering into the protected areas such as prisons, detention houses, neighborhoods, schools and houses, can be recognized.

Camera Monitoring System: It can be used for monitoring the crowd in public places such as banks, airports, stadiums, shopping malls, and supermarkets.

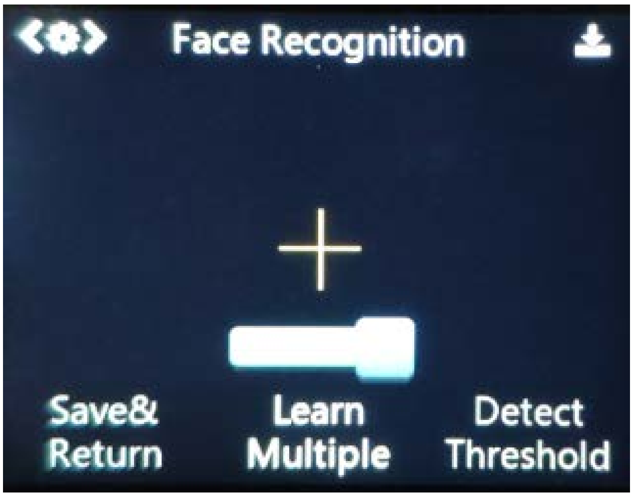
Internet Application: Make use of face recognition as assistance for online payment to prevent others from embezzling credit cards and social security.

At present, face recognition is widely used in all walks of life, such as attendance system, unlocking phones, the all-in-one machine of human identity verification, face payment and so on.

4. Demonstration of HUSKYLENS Face Recognition

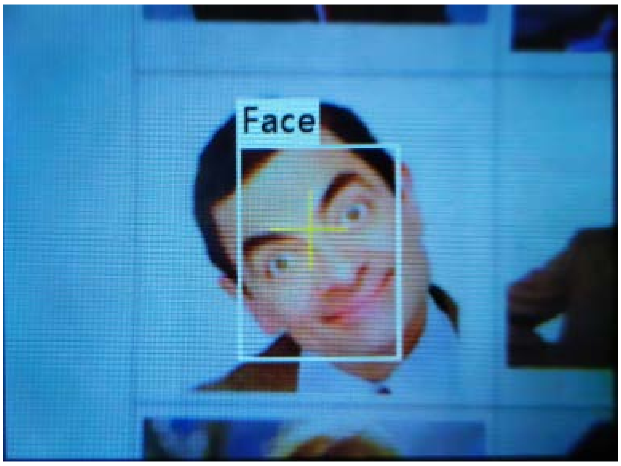
1. Select "Face Recognition" Function

Dial the "function button" to the left till "face recognition" is displayed at the top of the screen.

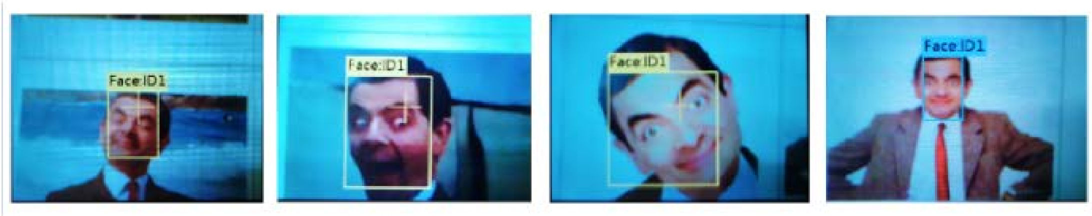


2. Learn Faces

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

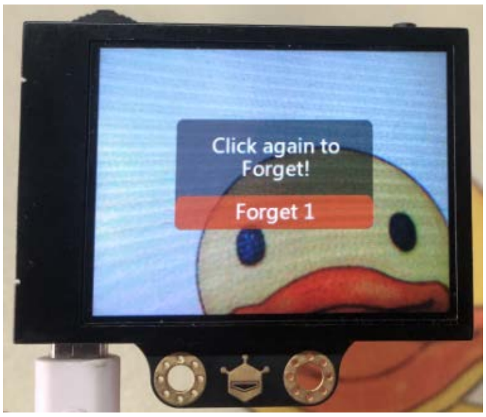


Point the "+" symbol at the face that needs to be learned and short press the "learning button". If the same face is detected by HUSKYLENS, a blue frame with words "Face: ID1" will be displayed on the screen, which indicates that HUSKYLENS has learned the face before and can recognize it now.

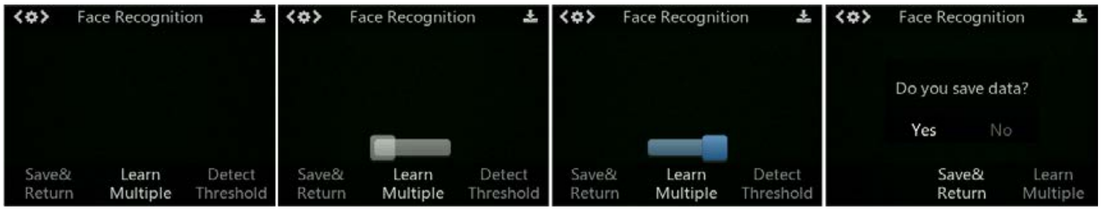


Note: Long-press the "learning button" then let HUSKYLENS learn the face from different angles.

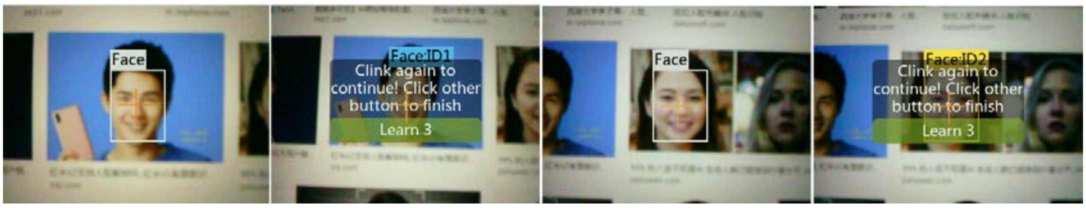
*If there is no "+" symbol on the screen, it means that HUSKYLENS has already learned it in the current function. When HUSKYLENS is in the face recognition mode, short press the "learning button", the screen will display "click again to forget". Before the countdown ends, short press the "learning button" again to delete the learned face information.



If you want to learn multiple faces, long-press the "function button" to enter the submenu of face recognition and move the "multiple learning" bar to the maximum value (far right).



Different from single face learning, in Multiple face learning mode, the learned face information will be named in order, such as face: ID1, face: ID2, face: ID3... And different face IDs have different frame colors.


Face
Face:ID1
Clink again to continue! Click other button to finish
Learn 3
Face
Face:ID2
Clink again to continue! Click other button to finish
Learn 3

1. Short press the "learning button" before the countdown ends to learn the second face.
2. For learning the third face, short press "learning button" before the countdown ends.

Note: The face learning is done when the countdown ends.

The information of the learned face will be automatically saved. When the camera detects the learned face from multiple faces, it will select the learned face with a frame and display the corresponding ID number.

Project Practice

How do we use face recognition of HUSKYLENS in sharing vehicles? How does it count the mileage and cost? Let’ s finish this project with three steps.

First, learn how to use the face recognition function of HUSKYLENS, and scan the face to unlock the vehicle; then, learn how to count the mileage and duration; finally, complete the whole project and display relevant information.

Task1: Unlock Maqueen Plus with Face

Program Design

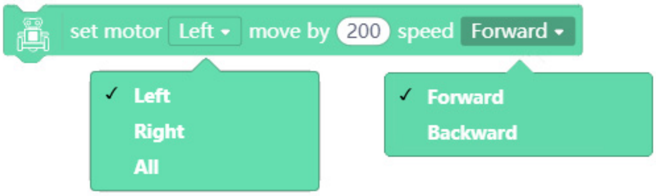
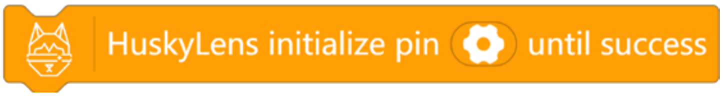
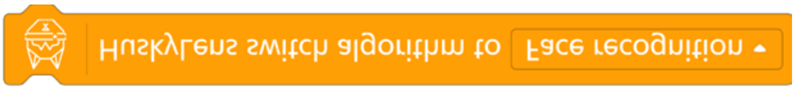
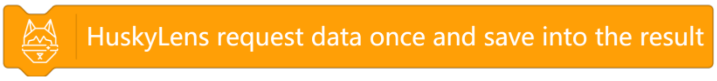
Step1: Face Learning and Recognition

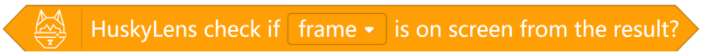
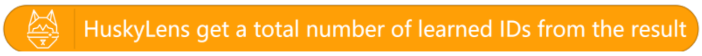
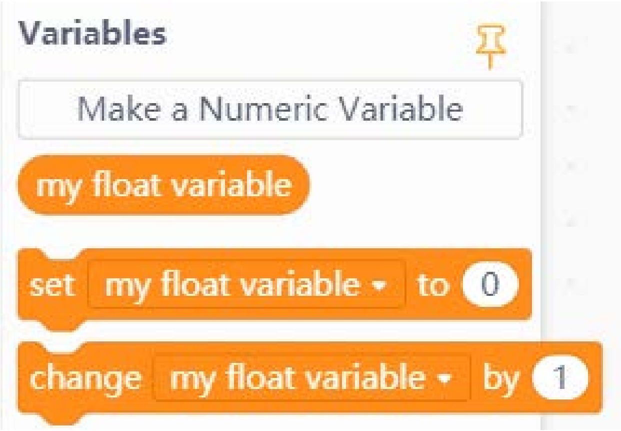
Here we can let HUSKYLENS learn a few face images at will. (Note: Turn on the "multiple learning" function first)

ID1:	ID2:	ID3:

Step2: Instruction Learning

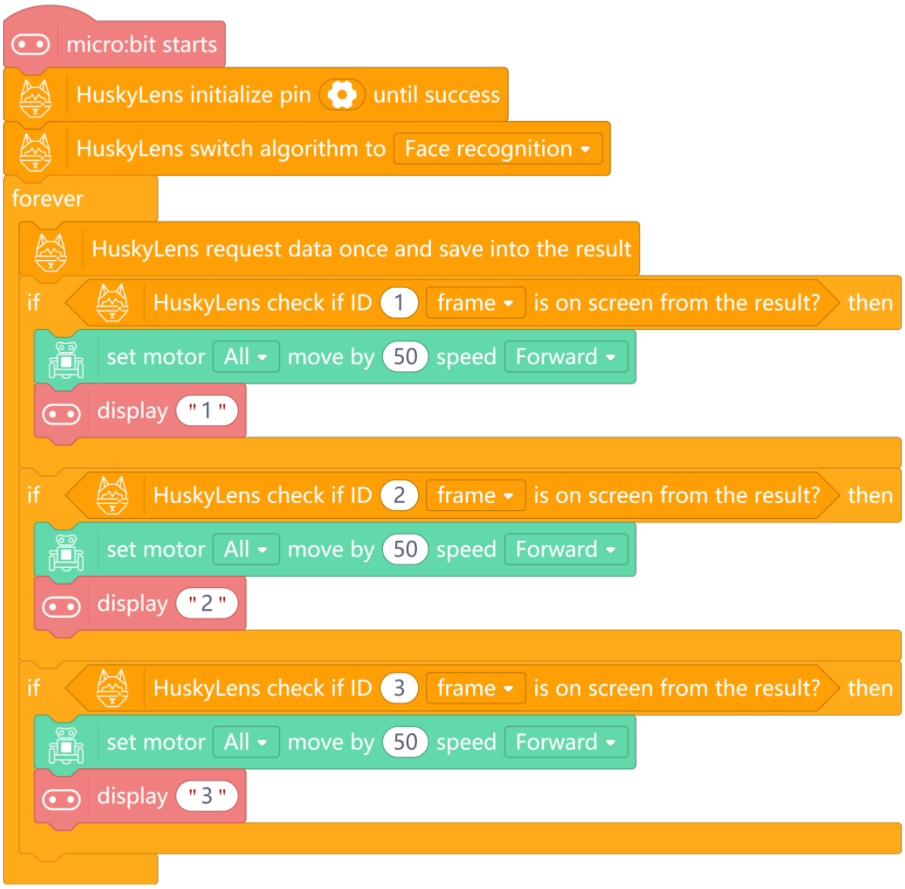
Graphical Block	Function Description

 set motor Left move by 200 speed Forward ✓ Left Right All ✓ Forward Backward	**Motor Controlling:** Drive Maqueen Plus forward, backward, turn left, turn right, stop. **Position of Motor:** Left, Right, Both **Rotation Direction:** Forward, backward **Rotation Speed**: 0~255
	**Initialize I2C:** Use this module to initialize the I2C communication protocol. Put it inside the "On start" program and execute it only once..
	**Switch Algorithm:** You can switch to other algorithms at any time, and there can only be one algorithm at the same time, which is usually executed when the device is turned on. There are 6 modes: **face recognition, object tracking, object recognition, line tracking, color recognition, QR code recognition.**
	**Request Data from HuskyLens Once:** The main control board requests HUSKYLENS once for storing data in the "result" (store in the main control board's memory variables, one request refreshes the data stored in the memory once), and then data can be

	obtained from the "result", only after this module is called, the latest data can be obtained from the "results".
 HuskyLens check if frame ▾ is on screen from the result?	**Determine Whether the Learned ID is on the Screen:** Get "result" from the requested to see whether the IDx is on the screen. The frame refers to the algorithm whose target is a frame, and the arrow refers to the algorithm whose target is an arrow. Select the arrow for the line tracking algorithm, and select the frame for other algorithms.
 HuskyLens get a total number of learned IDs from the result	**Obtain the Total Number of Learned IDs:** Get the number of targets that have been learned under the current algorithm from the "results" obtained by the request. Note that after long-pressing the function button of the HUSKYLENS you can set whether to learn multiple targets.
 Variables Make a Numeric Variable my float variable set my float variable ▾ to 0 change my float variable ▾ by 1	**Variable:** Variables refer to the changing items, which is convenient for storing changing numbers. Click "Set Variable" to create a new variable. You can directly set the value of a variable, and also increase or decrease the value of the variable.

Step3: Function Analysis

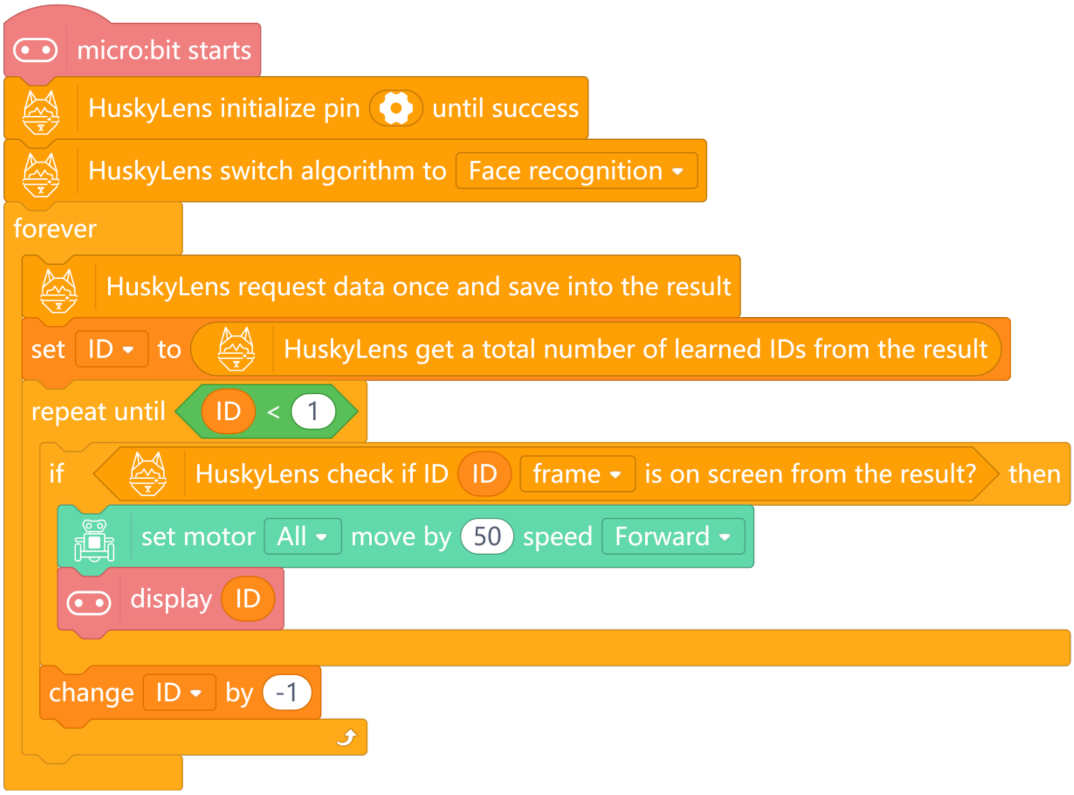
Assuming there are only three faces, the program is easy to implement. As shown in the figure below, we just need to determine them one by one.



```
micro:bit starts
    HuskyLens initialize pin until success
    HuskyLens switch algorithm to Face recognition
forever
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 frame is on screen from the result? then
        set motor All move by 50 speed Forward
        display " 1 "
    if HuskyLens check if ID 2 frame is on screen from the result? then
        set motor All move by 50 speed Forward
        display " 2 "
    if HuskyLens check if ID 3 frame is on screen from the result? then
        set motor All move by 50 speed Forward
        display " 3 "
```

But how do we achieve it with the program when we have to learn five, ten, or more faces? Should we determine them one by one? That's a little verbose, and makes the program lengthy. Do you find any **patterns** in the above program?

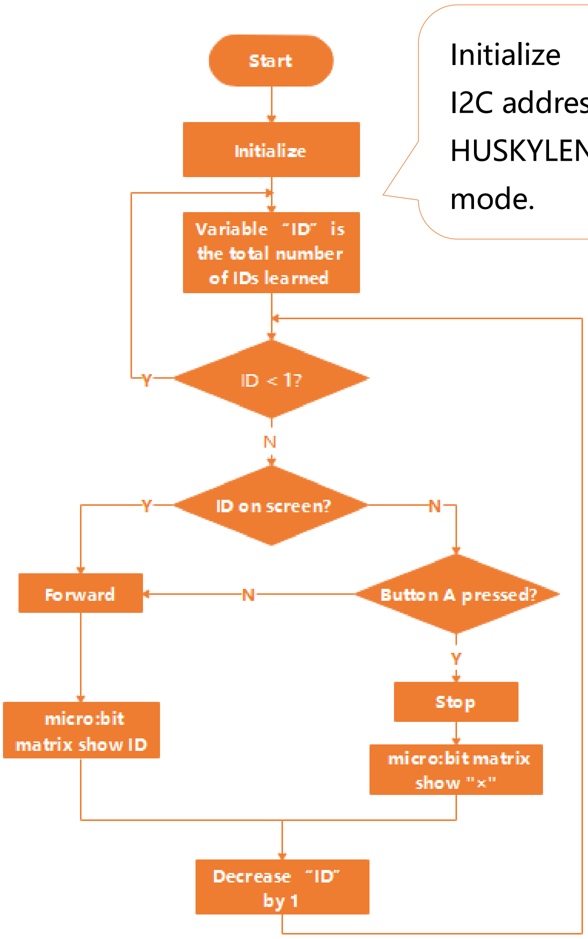
In fact, as long as we add a **variable,** we can greatly simplify the repetitive part of the above program. The program is as follows:



```
micro:bit starts
    HuskyLens initialize pin ⚙ until success
    HuskyLens switch algorithm to Face recognition ▾
forever
    HuskyLens request data once and save into the result
    set ID ▾ to HuskyLens get a total number of learned IDs from the result
    repeat until ID < 1
        if HuskyLens check if ID ID frame ▾ is on screen from the result? then
            set motor All ▾ move by 50 speed Forward ▾
            display ID
        change ID ▾ by -1
```

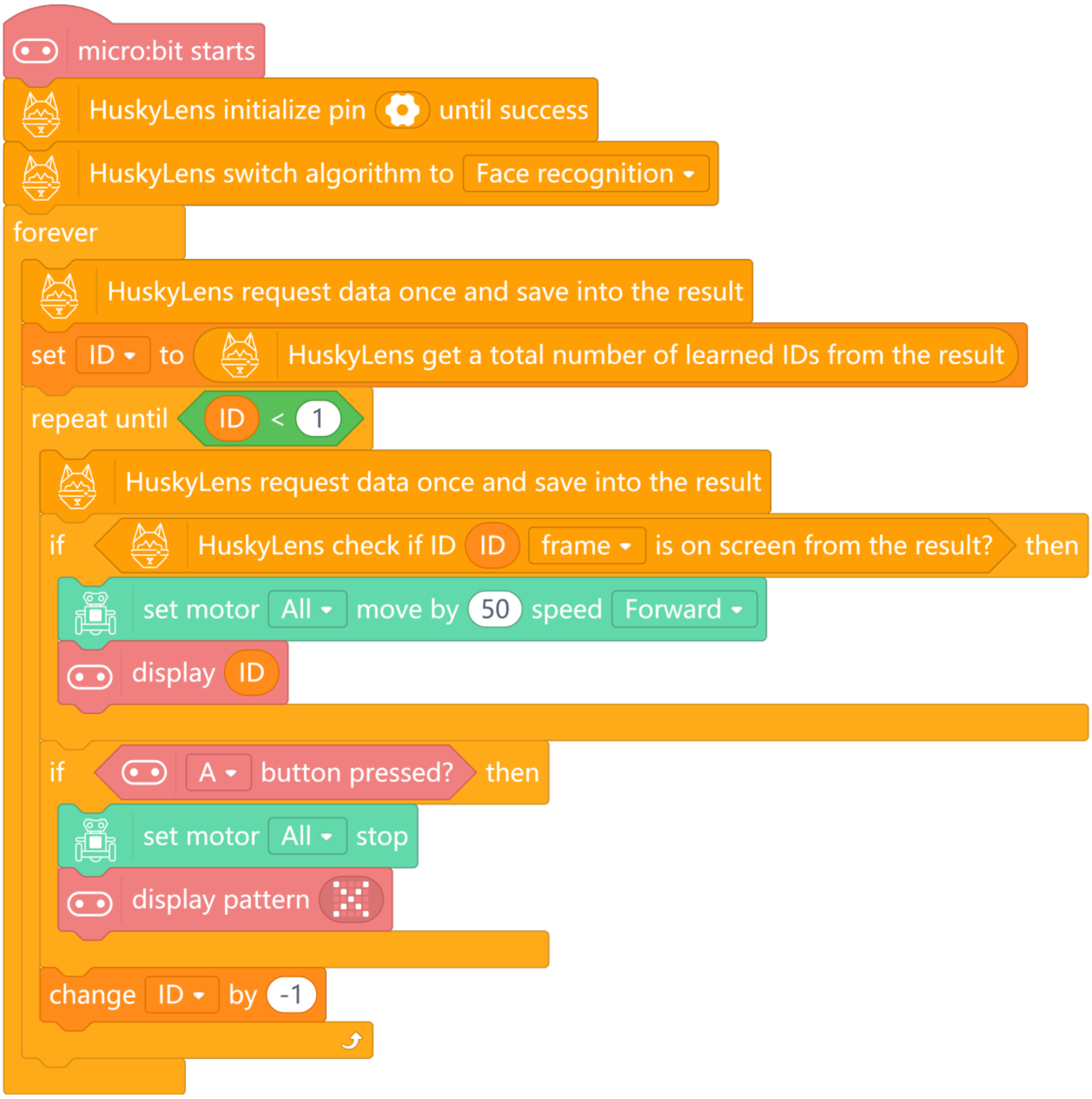
According to the above program, we may find that the Maqueen Plus can not stop after it is started? If another person wants to unlock it, they can only cut off the power and restart. So, we still have to improve this program. When the Maqueen Plus is not used, the stop of Maqueen Plus will be triggered by its button pressed. And We only need to scan the face one more time for the next use.

Task1: The flowchart of the overall program of Unlocking McQueen Plus with Face is as shown on the right. You can use the flowchart to clarify the program logic, and you can also use the program to understand the flowchart about how to realize the function.



Initialize the I2C address, HUSKYLENS mode.

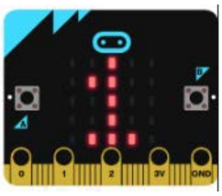
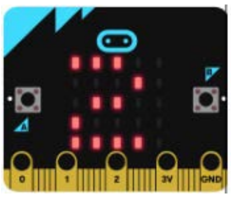
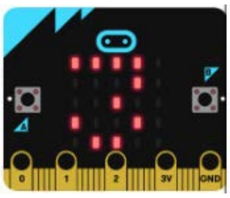
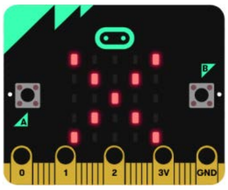
Sample Program



```
micro:bit starts
    HuskyLens initialize pin I2C until success
    HuskyLens switch algorithm to Face recognition
forever
    HuskyLens request data once and save into the result
    set ID to HuskyLens get a total number of learned IDs from the result
    repeat until ID < 1
        HuskyLens request data once and save into the result
        if HuskyLens check if ID ID frame is on screen from the result? then
            set motor All move by 50 speed Forward
            display ID
        if A button pressed? then
            set motor All stop
            display pattern
        change ID by -1
```

Note: all the program links are at the end of the article.

Operating Effect

	Recognizing D1	Recognizing D2	Recognizing D3	Press Button A
Status of micro:bit				
Status of Maqueen Plus	Go Forward			Stop

Task2: Pricing

Program Design

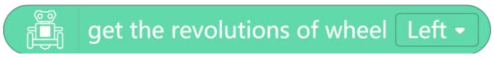
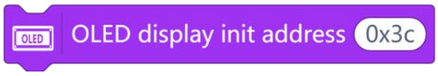
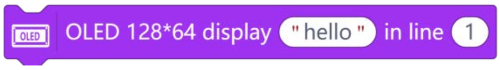
Step1: Charging Formula

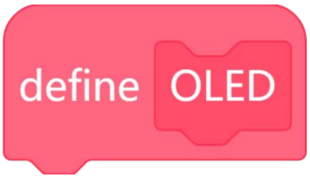
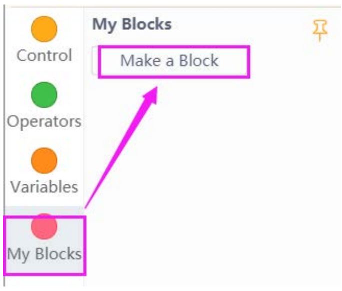
How do we charge users for sharing vehicles? To figure out the final cost, we need to calculate the following figures. The calculation formula is shown in the following table:

Figure	Formula	Note
Perimeter of the wheel	Perimeter * 3.14	The tire perimeter of Maqueen Plus is 4.27cm
Mileage	Number of turns * Perimeter	
Duration	（Ending time – Beginning time）/1000	In Micro:bit, the duration is measured in ms. If you want to measure the duration in s, you need to divide it by 1000.
Cost	Unit Price * Mileage + Unit Price * Duration	

Assume that the charging standard of the sharing vehicles is: 1yuan/km+0.1yuan /min.

Step2: Instruction Learning

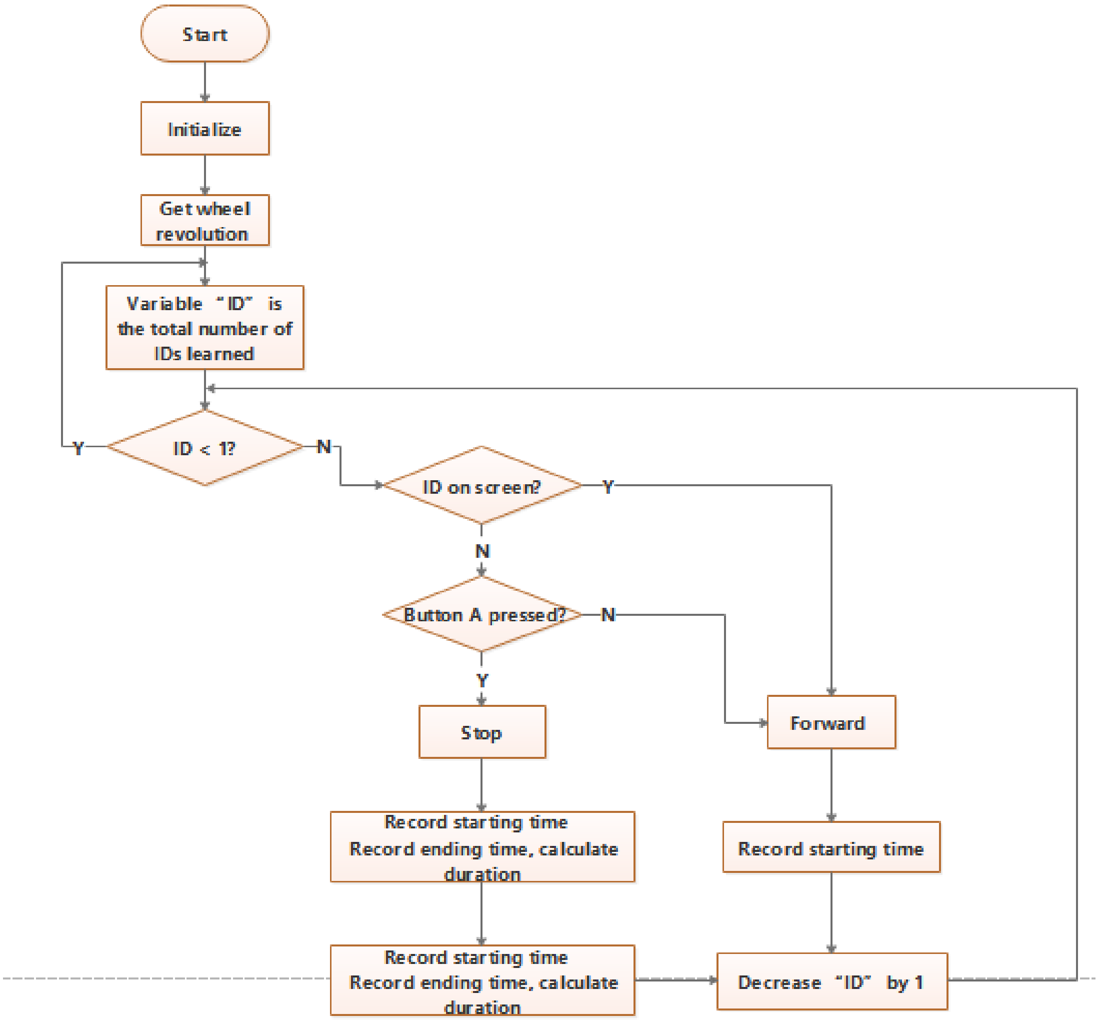
Graphical Block	Function Description
 get the revolutions of wheel Left ▾	**Obtain Wheel Revolutions:** Use the encoder to obtain the tire revolutions, which can be used to calculate the mileage of Maqueen Plus. **Position of motor**：Left, Right
 OLED display init address 0x3c	**Initialize OLED:** Initialize the OLED screen and place it at "On start" for execution only once
 OLED 128*64 display "hello" in line 1	**Text Displaying:** Display text on the OLED screen, and you can set the number of lines from 0 to 7.

	**Custom Function:** It refers that if you want to use particularly complex calculations in formulas or calculations, you can encapsulate the complex functions through custom functions. It is convenient for users to understand or modify. Click Function→Make a Function  My Blocks Make a Block Control Operators Variables My Blocks
	Call the Custom Function: call your own packaged function, drag and drop the block to the corresponding position.
	Numerical Round Operator: The counting method of rounding is adopted. When the first digit after the decimal point is less than 5, it will be rounded down; if it is greater than or equal to 5, the previous digit will be incremented by "1". The final result is an integer.

Step3: Function Analysis

In real life, the charging starts at the point of unlocking, and ends when the vehicles returned.

Corresponding to our project, when the face is recognized, it starts to record the mileage and duration; when the A button is pressed, the end time and final mileage are recorded. Finally, it will calculate the duration and cost through the formula. The program flowchart is shown as follows.

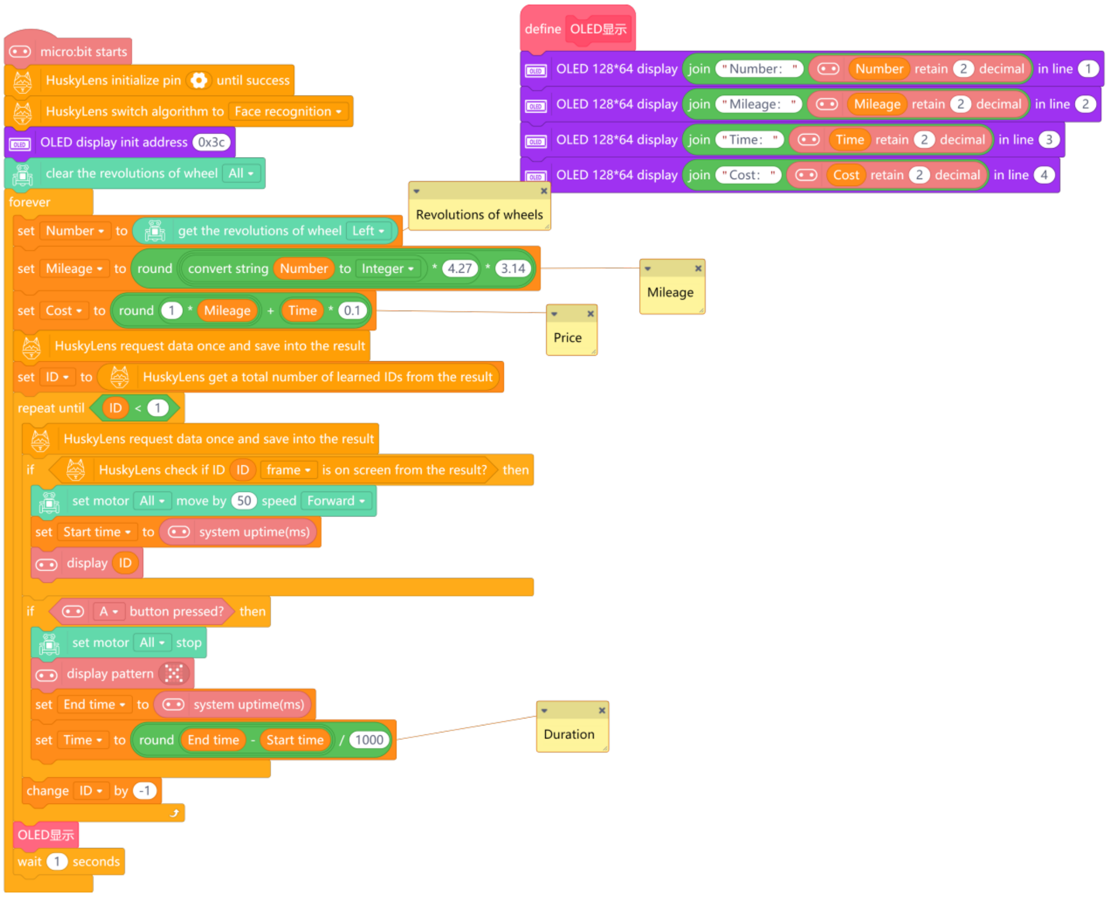


Note:

The function of “Get the number of turns” requires Version EN-V1.4 and above. We can check the version of Maqueen Plus through the instruction “ get version ”. If the version is too low, you need to burn new firmware to upgrade. Refer to Attachment 1 for the method of burning firmware.

For a better effect, here we use “cm” as the unit of mileage and “second” as the unit of time. Therefore, the pricing standard of the sharing vehicles for the Mcqueen Plus is: 1 yuan/cm+0.1 yuan/sec.

Sample Program



```
micro:bit starts
  HuskyLens initialize pin until success
  HuskyLens switch algorithm to Face recognition
  OLED display init address 0x3c
  clear the revolutions of wheel All

forever
  set Number to get the revolutions of wheel Left
  set Mileage to round convert string Number to Integer * 4.27 * 3.14
  set Cost to round 1 * Mileage + Time * 0.1
  HuskyLens request data once and save into the result
  set ID to HuskyLens get a total number of learned IDs from the result
  repeat until ID < 1
    HuskyLens request data once and save into the result
    if HuskyLens check if ID ID frame is on screen from the result? then
      set motor All move by 50 speed Forward
      set Start time to system uptime(ms)
      display ID
    if A button pressed? then
      set motor All stop
      display pattern
      set End time to system uptime(ms)
      set Time to round End time - Start time / 1000
    change ID by -1
  OLED显示
  wait 1 seconds

define OLED显示
  OLED 128*64 display join "Number: " Number retain 2 decimal in line 1
  OLED 128*64 display join "Mileage: " Mileage retain 2 decimal in line 2
  OLED 128*64 display join "Time: " Time retain 2 decimal in line 3
  OLED 128*64 display join "Cost: " Cost retain 2 decimal in line 4
```

Revolutions of wheels

Mileage

Price

Duration

Note: The overall program link is at the end of the article.

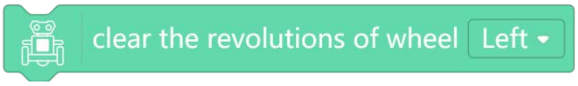
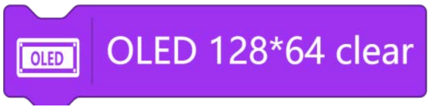
Operating Effect

When detecting human faces, Maqueen Plus will go forward and information such as the number (revolution) and mileage will be displayed on the OLED screen. When button A is pressed, the OLED screen will show the mileage (136.63cm), duration (23s), and cost (136.63*1+23*0.1≈139.30).

Task3: Set A Flag Bit to Realize Charging Reset

Program Design

Step1: Instruction Learning

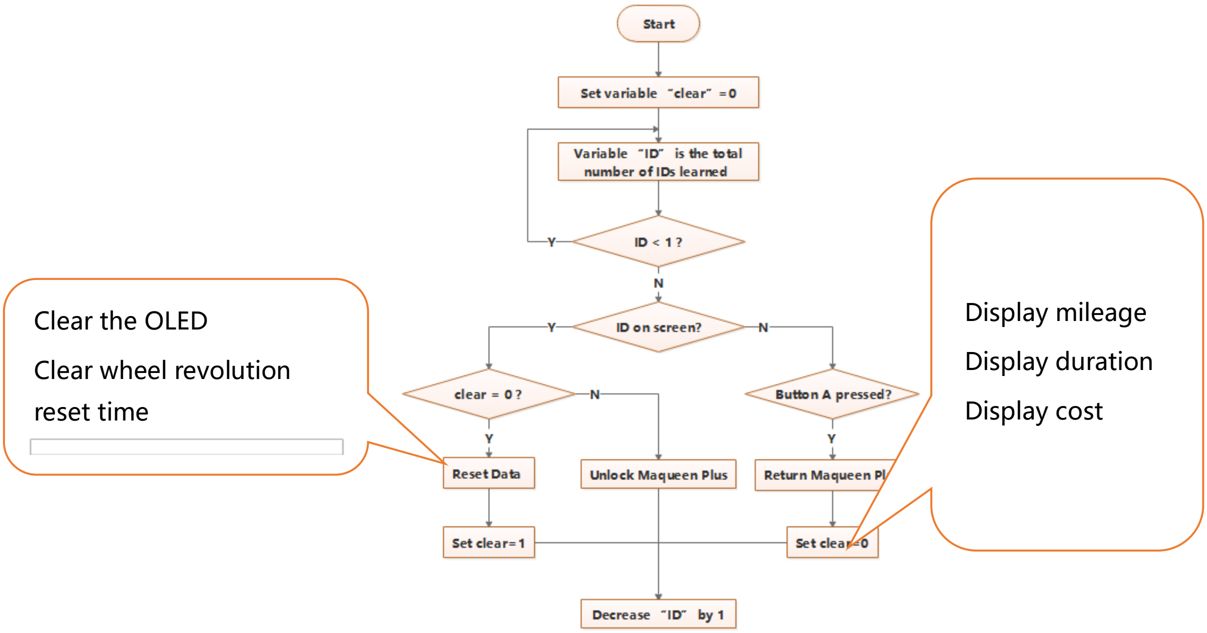
Graphical Block	Function Description
 clear the revolutions of wheel Left ▾	**Clear wheel Revolutions :** Clear the number of turns of wheel **Position of Motor**: Left, Right, Both
 OLED 128*64 clear	**OLED Clearance:** Clear the content on the OLED screen.

Step2: Function Analysis

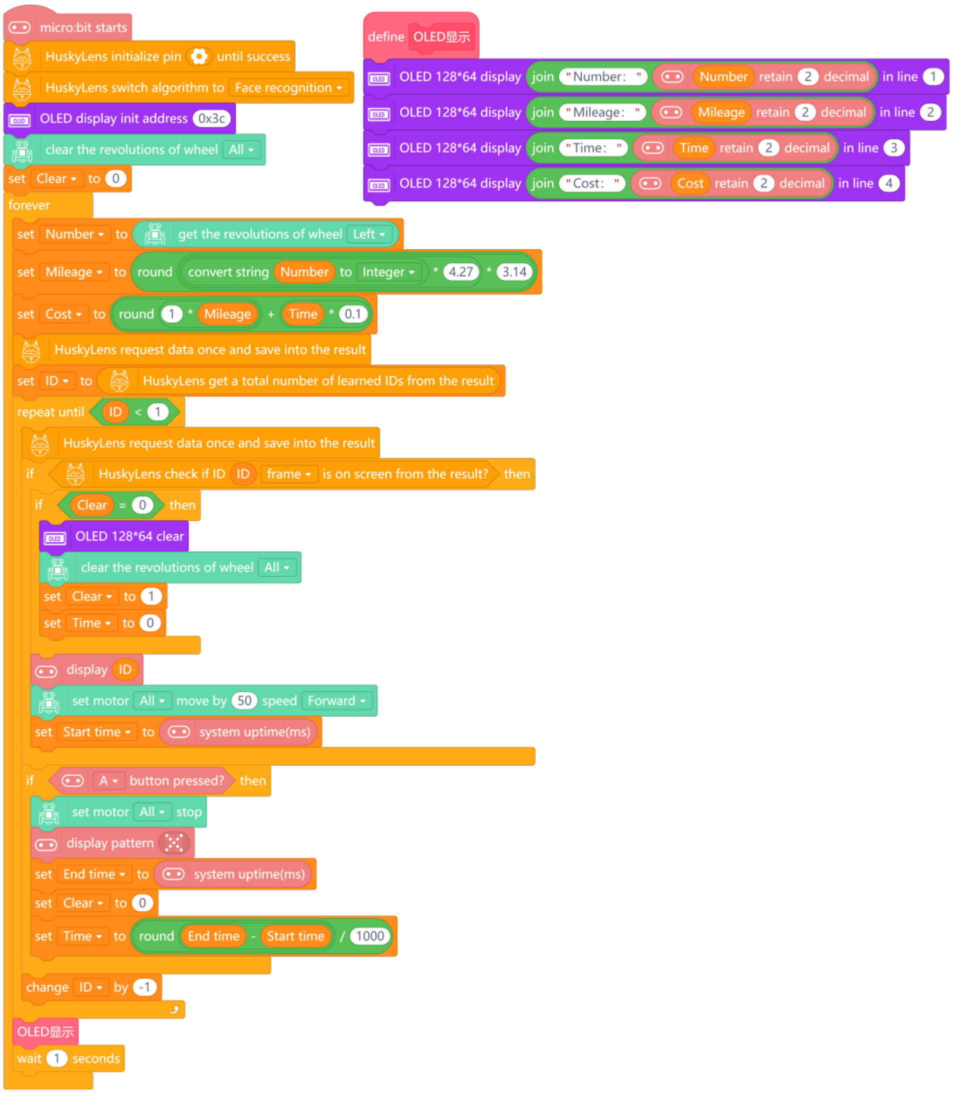
From life experience, after each user has paid the fee, the sharing car will re-record the mileage, duration and cost when it is restarted.

Corresponding to our project, after the ID1 user has paid the fee, when the ID2 user uses the car, the mileage, duration, and cost will be recalculated.

On the basis of the task2, we need to add that when the Maqueen Plus stops and detects the face again, it will reset the tire revolutions and duration. The program flowchart is as follows.



Sample Program



```
micro:bit starts
    HuskyLens initialize pin until success
    HuskyLens switch algorithm to Face recognition
    OLED display init address 0x3c
    clear the revolutions of wheel All
    set Clear to 0

forever
    set Number to get the revolutions of wheel Left
    set Mileage to round convert string Number to Integer * 4.27 * 3.14
    set Cost to round 1 * Mileage + Time * 0.1
    HuskyLens request data once and save into the result
    set ID to HuskyLens get a total number of learned IDs from the result
    repeat until ID < 1
        HuskyLens request data once and save into the result
        if HuskyLens check if ID ID frame is on screen from the result? then
            if Clear = 0 then
                OLED 128*64 clear
                clear the revolutions of wheel All
                set Clear to 1
                set Time to 0
            display ID
            set motor All move by 50 speed Forward
            set Start time to system uptime(ms)
        if A button pressed? then
            set motor All stop
            display pattern
            set End time to system uptime(ms)
            set Clear to 0
            set Time to round End time - Start time / 1000
        change ID by -1
    OLED显示
    wait 1 seconds

define OLED显示
    OLED 128*64 display join "Number: " Number retain 2 decimal in line 1
    OLED 128*64 display join "Mileage: " Mileage retain 2 decimal in line 2
    OLED 128*64 display join "Time: " Time retain 2 decimal in line 3
    OLED 128*64 display join "Cost: " Cost retain 2 decimal in line 4
```

Note: the overall program link is at the end of the article

Operating Effect

After the user ID1 finishes using, if the user ID2 scans the face to unlock the Maqueen Plus car, the car starts. The data of the user ID1 will be reset and data recording restarts.

Project3: Auto-Tracking Vehicle

When we are at the airport, we always have to keep on hand on the suitcase, and this leaves us in a mess. I wonder how good it is if there is an "auto-driving" suitcase that just can follow us.

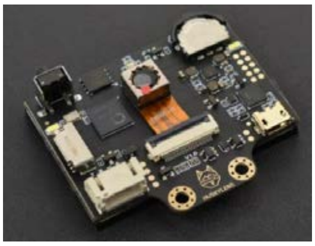
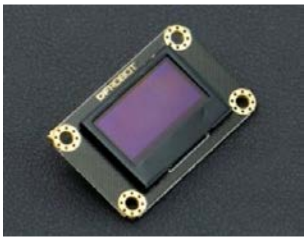
The auto-tracking suitcase mainly uses AI visual cameras to recognize human bones + human faces + characteristic identification, and then combines tracking algorithms to perform stable and accurate recognition of users.

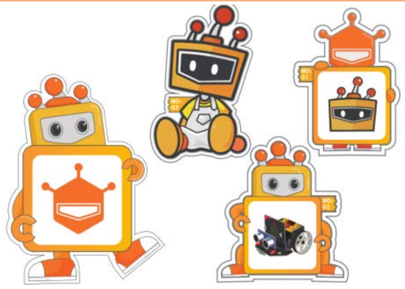
Let' s think about it, can we use the principle of the auto-tracking suitcase to create an auto-tracking Maqueen Plus?

Function

The project mainly uses the object tracking function of HUSKYLENS to let Maqueen Plus follow the vehicle in front "flexibly".

Bill of Materials

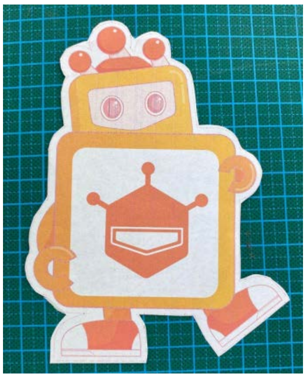
micro:bit ×1	**Maqueen Plus ×2 (2 and above)**
	
HUSKYLENS ×1	**OLED ×1**

	
Wireless IR Remote Controller	**Target Sticker**

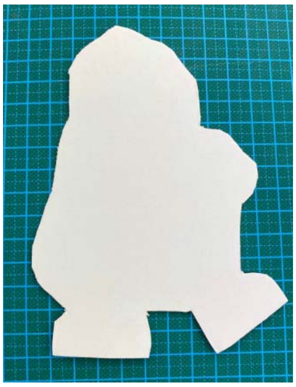
Hardware Connection

1.The Leading Vehicle: It is the first vehicle, which does not need to be connected to peripheral hardware, but the target sticker should be attached onto it. The steps are as follows.

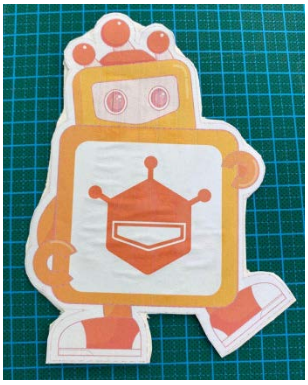
(1) Print out the target sticker, then choose the one you like, cut it out along its outline and paste it on the paperboard. Finally, stick it on the Maqueen Plus vehicle.



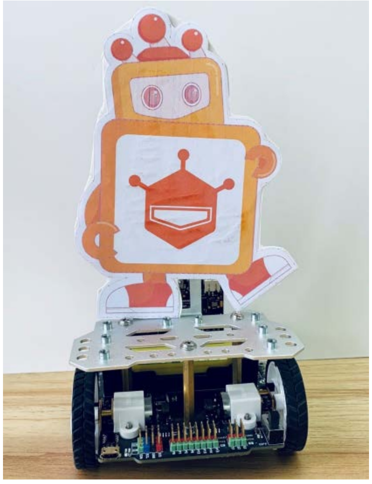
1. Trim the target pattern



2. Trim the paperboard

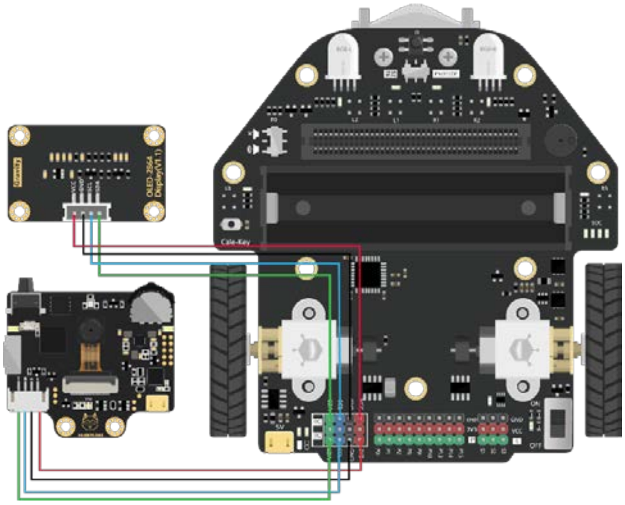


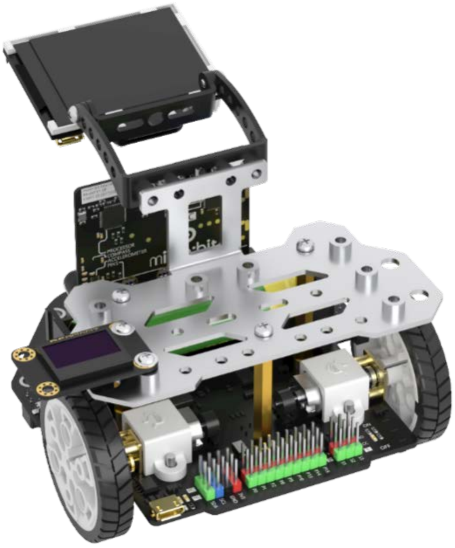
3. Paste it on the paperboard



4. Stick the finished target pattern on the Maqueen Plus

2.Following Vehicles: The hardware connection diagram and installing diagram are as follows.





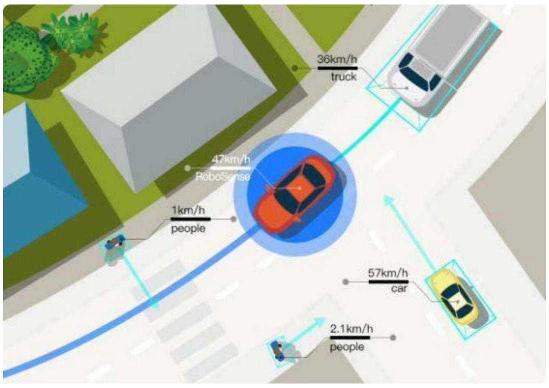
Note: If you are using more than 2 Maqueen Plus cars, stick the target pattern on the back of each car in turn.

Knowledge Field

When we are going to track a moving object, visual object tracking is needed besides manual operation. This technology has already been widely used in our life, such as video monitoring, UAV shotting with follow, etc. In this project, we make use of the object tracking of HUSKYLENS.

1. What is object tracking?

As one of the vital functions of AI visual recognition, object tracking belongs to one type of behavior recognition. Object tracking is a key point in computer vision, referring to the process of making continuous inferences about the target' s state in a video sequence, which can be simply regarded as recognizing and tracking the objects moving within the visual range of the camera.

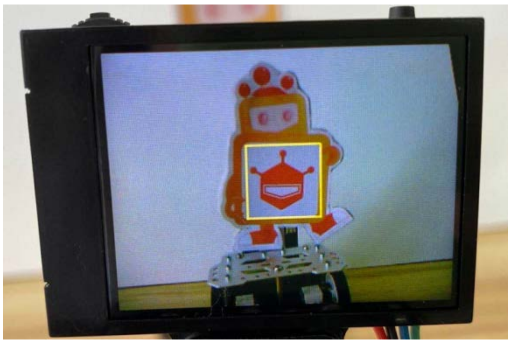


2. Operating principles:

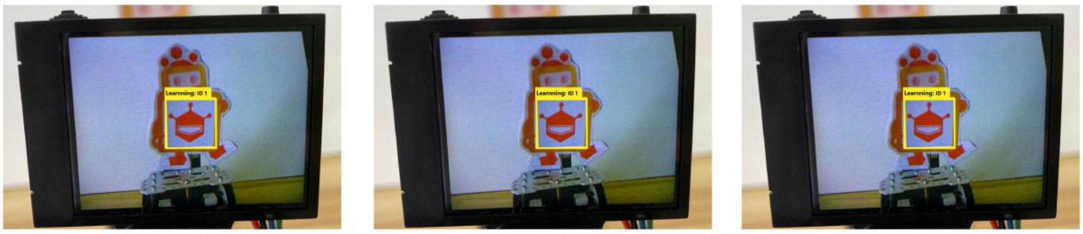
The image information is collected by the camera and sent to the computer. After analysis and process, the computer can work out the relative position of the moving object. Meanwhile, it will make the camera rotate to carry out real-time tracking. The object tracking system is mainly divided into four steps: object recognition, object tracking, movement prediction, camera controlling.

Object recognition **Object tracking** **Movement prediction** **Camera controlling**

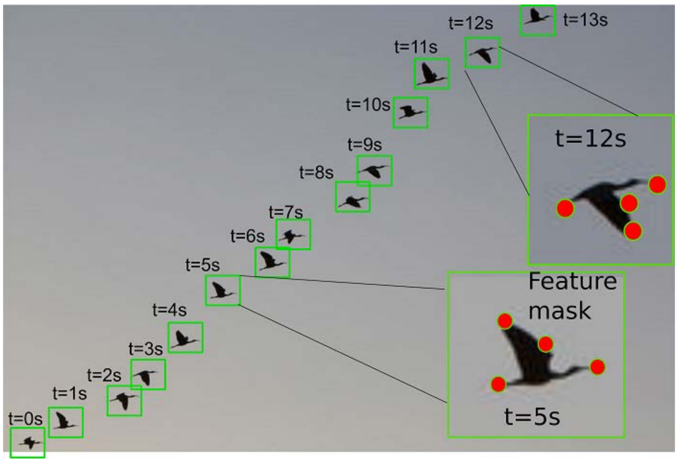
Object recognition: It is to obtain accurate appearance information of the object through some image processing algorithms under a static background, and the shape of the object can be recognized and marked, as shown in the figure.



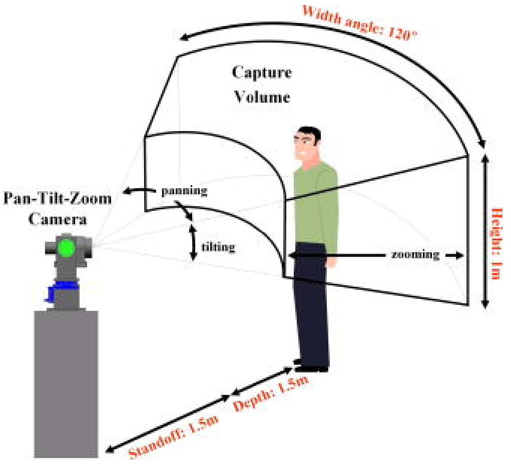
Object tracking: It refers to tracking the subsequent image sequence through algorithms according to the appearance characteristics of the object obtained from the previous step, and carry out more in-depth learning in the subsequent tracking to make the tracking more and more accurate.



Movement prediction: it means predicting the image of a moving object in the next frame using algorithms so as to optimize the algorithm and improve efficiency.



Camera controlling: It is to move the camera according to the moving direction of the object while collecting the image information. It usually requires coordination with a cloud platform or other movement mechanism.



3. Application Fields

Smart Video Monitoring: Based on motion recognition (human recognition basing on footwork, automatic object detection), automatic monitoring (to monitor the suspicious acts), traffic monitoring (collecting the real-time traffic data to direct the traffic).

Human-computer Interaction: The traditional human-computer interaction is carried out by the keyboard and mouse of the computer. Tracking technology is the key point when a computer needs to be able to recognize and understand the posture, movement, and gesture.

VR: 3D interaction and virtual character action simulation in the virtual environment directly benefit from the research results of video human motion analysis. They provide richer forms of interaction for the participants. And human tracking and analysis are the key technologies.

4. Demonstration of HUSKYLENS Sensor-object tracking Function

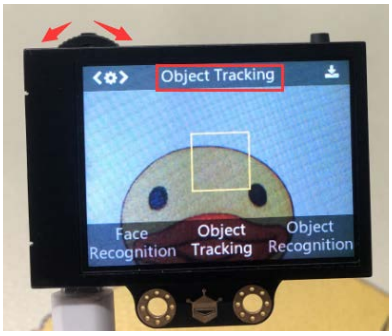
HUSKYLENS has a built-in object tracking function, allowing to learn the features of the object, track the position of object on the screen and feedback the position information to the main control board.

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

The object tracking function can only track one object, and does not support tracking multiple objects now. Therefore, it is better for the learned object to have a clear outline, so that it can be recognized more easily.

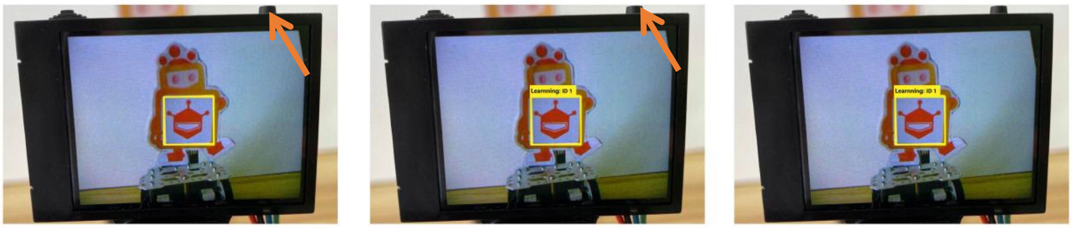
1. Select "Object Tracking" Function

Dial the "function button" to the left till the "object tracking" shows at the top of the screen.



2. Learning

Point HUSKYLENS to the target object, adjusting the distance till the object is included in the yellow frame of the center of the screen. If the object is difficult to be completely contained in the yellow frame, containing distinctive features is also okay. Then long press "learning button" to learn the object from various angles and distances. During the learning process, the words "Learning: ID1" with a yellow frame will be displayed on the screen.



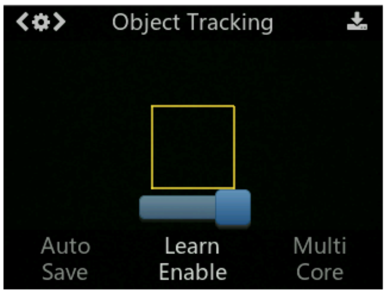
When HUSKYLENS can track the object at different angles and distances, you can release the "learning button" to end the learning.

Note: If there is no yellow frame in the center of the screen, it means that HUSKYLENS has learned an object before. Please let it forget the learned one and learn again.

3. *Keep Learning*

The object tracking feature allows HUSKYLENS to keep learning the current state of the object as soon as the camera sees it, which helps it capture moving objects.

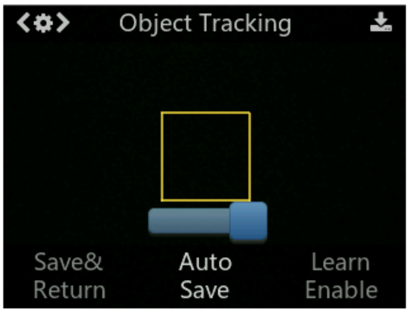
Operation method: Long press the "function button" to enter the submenu of the object tracking function, select "learning enable", then short press the "function button", and then dial the "function button" to turn on the "learning enable" switch, that is to say: The color of the progress bar turns blue, and the square on the progress bar is located at the right of the progress bar. When exiting, select "Yes" to save the parameters.



4. *Save the Model*

When HUSKYLENS is rebooted, it is default not to save the last object you studied, but you can realize it by turning on --” Auto Save” .

Operation method: It is the same as above. After entering the submenu, turn on the "Auto Save" function. It will save the object learned last time.



Program Practice:

How does HUSKYLENS realize the object recognition in the project of the auto-tracking vehicle? How does it make the vehicle follow the front Maqueen Plus car? There are 4 steps for this project.

First learn to design the program of the first vehicle, use the infrared remote controller to control the first vehicle to move forward, turn left, and turn right; then learn to use the object recognition function of HUSKYLENS to output various parameters of the target object on the OLED screen (X coordinate and height); then learn how to make the vehicle follow the specified target. When the target moves, it will adjust; finally complete the whole project so that Maqueen Plus can follow the vehicle ahead as required.

Note: the programs after task 2 are all for the second following vehicle.

Task1: Design Program for the First Vehicle

Program Design

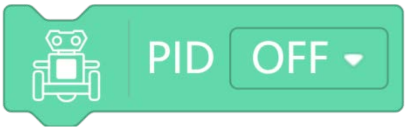
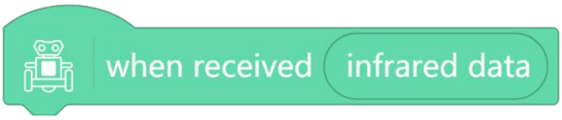
Step1: Learn the Controller

An Infrared remote controller is used in this project, so we need to understand the code value corresponding to each key on it.

Button	Code Value	Button	Code Value
Red	0	**VOL+**	1
FUNC/STOP	2	**Left Arrow**	4
Pause	5	**Right Arrow**	6
Down Arrow	8	**VOL-**	9
Up Arrow	10	**0**	12

EQ	13	**ST/REST**	14
1	16	**2**	17
3	18	**4**	20
5	21	**6**	22
7	24	**8**	25
9	26		

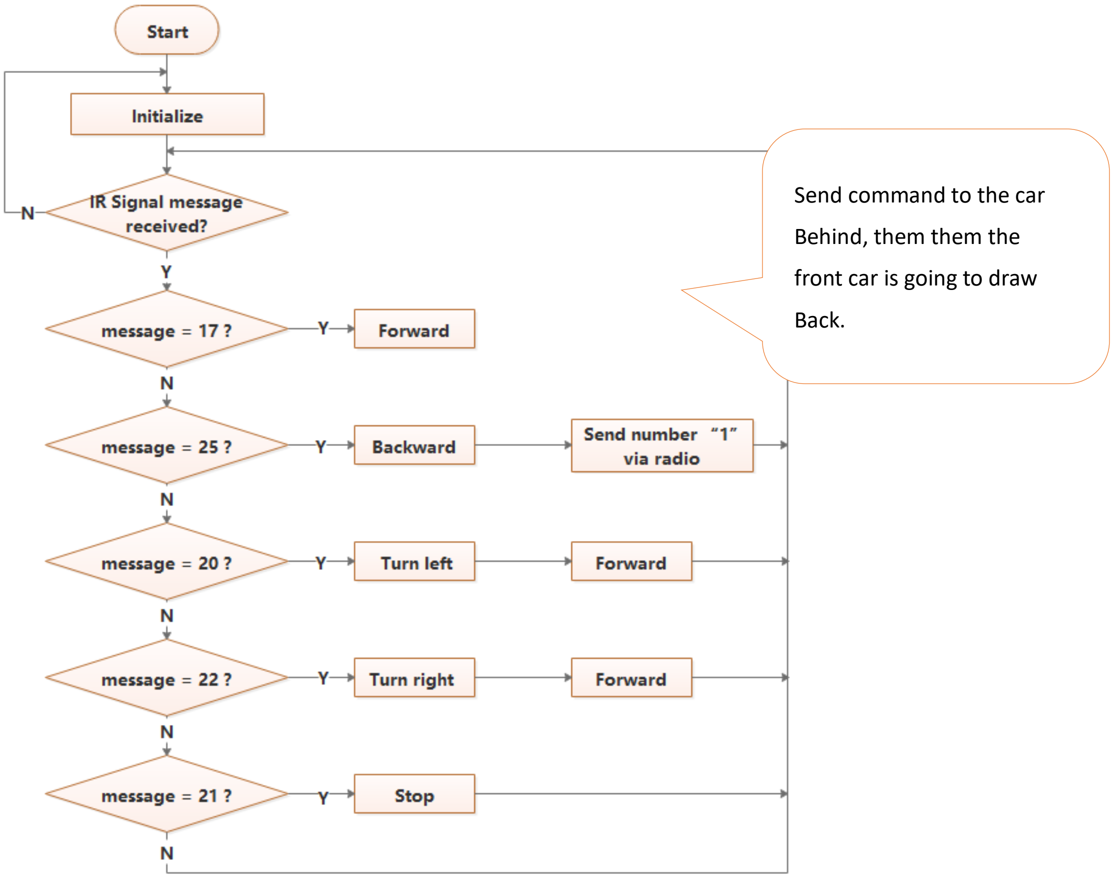
Step2: Instruction Learning

Graphical Block	Function Description
	**PID Switch** **Function Introduction:** Set whether the method of driving the motor adopts PID mode. When the PID is turned on, the car will adjust the speed and torque of the motor in real time. In the PID state, it can maintain a precise rotation speed and a large torque at a low rotation speed. But PID adjustment has a delay period of about 50 milliseconds. So, it is not suitable for occasions with high real-time control. **Option:** ON, OFF
	**When infrared data is received (Event triggering mode)** **Function Introduction :** When infrared data is received, it will be stored in the variable "message" and all the programs in this block will be executed. **Data Type**: Decimal integer (Read the last two digits of the hexadecimal key value of the remote

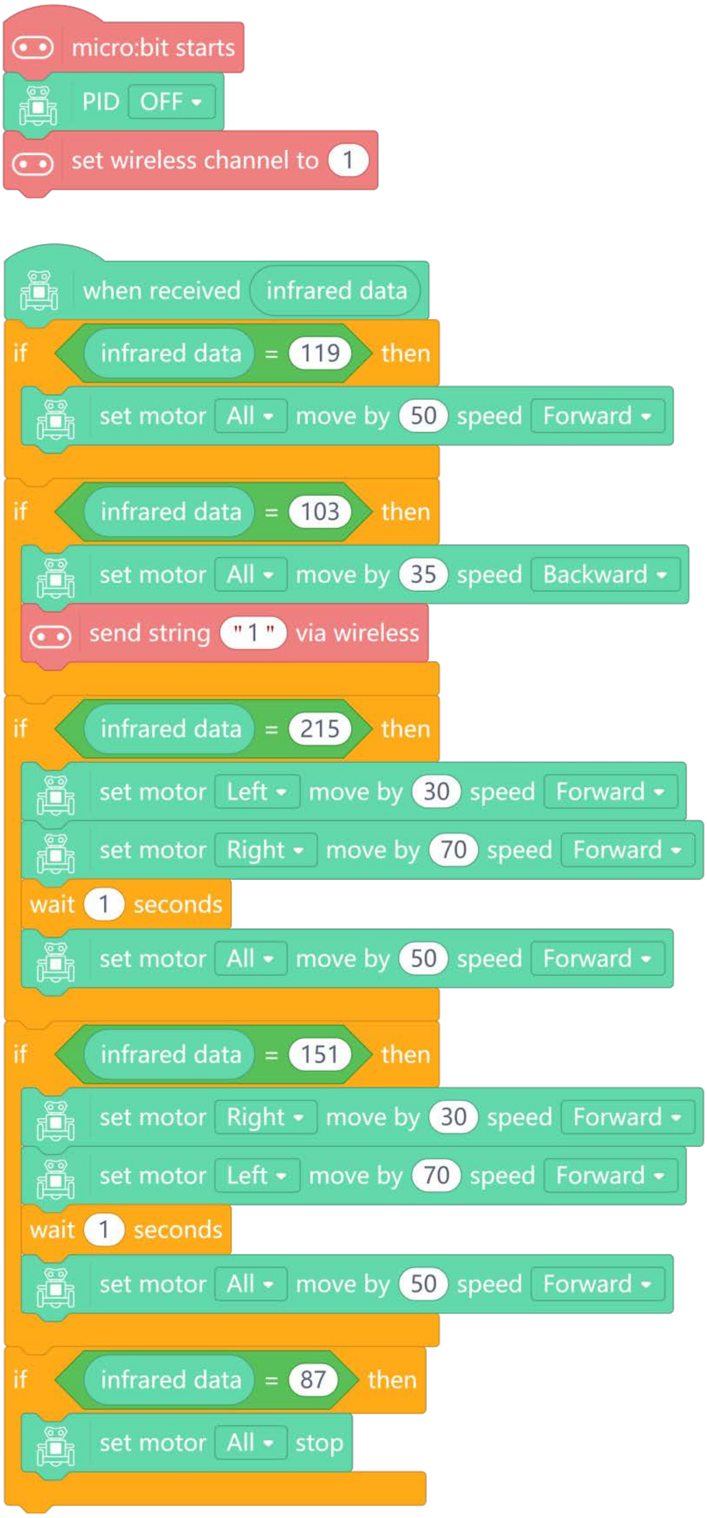
	controller and convert it to a decimal number) **Protocol Type:** NEC

Step3: Function Analysis

The infrared remote controller is used to control the first Maqueen vehicle to move forward, backward, turn left, turn right, and stop. The procedure flow chart is as follows.



Sample Program

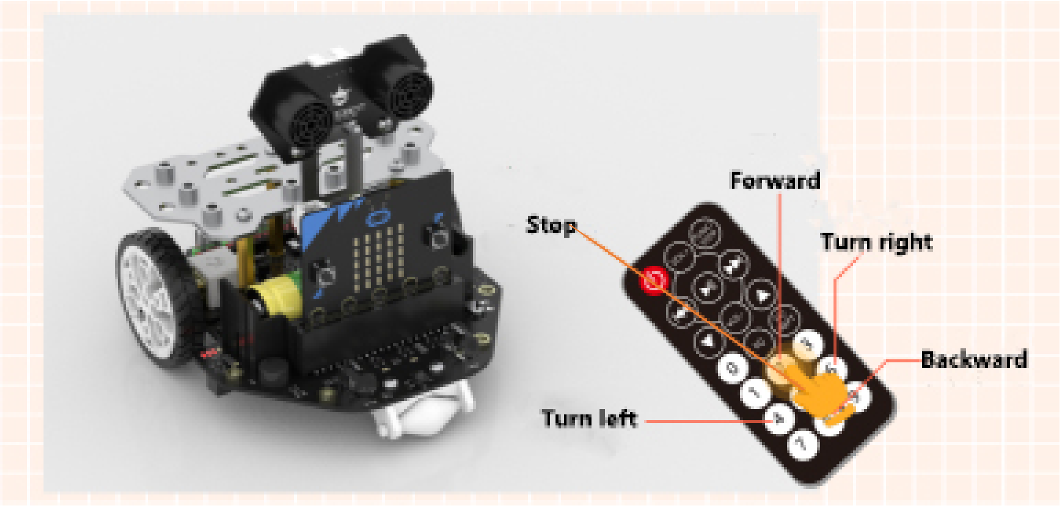


```
micro:bit starts
    PID OFF
    set wireless channel to 1

when received infrared data
    if infrared data = 119 then
        set motor All move by 50 speed Forward
    if infrared data = 103 then
        set motor All move by 35 speed Backward
        send string "1" via wireless
    if infrared data = 215 then
        set motor Left move by 30 speed Forward
        set motor Right move by 70 speed Forward
        wait 1 seconds
        set motor All move by 50 speed Forward
    if infrared data = 151 then
        set motor Right move by 30 speed Forward
        set motor Left move by 70 speed Forward
        wait 1 seconds
        set motor All move by 50 speed Forward
    if infrared data = 87 then
        set motor All stop
```

Note: the overall program link is at the end of the article

Operating Effect

Turn on Maqueen Plus and press the buttons 2, 8, 4, 6 and 5 on the remote controller to make Maqueen Plus go forward, backward, left, right and stop accordingly.

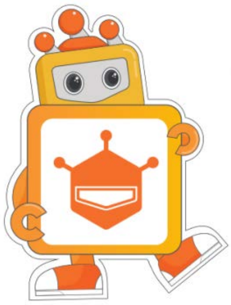


Task2: Learn Object Tracking

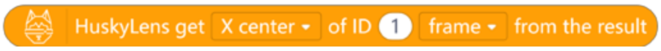
Program Design

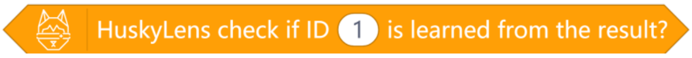
Step1: Learning and Recognition

Select a target pattern you like and let it be learned.



Step2: Instruction Learning

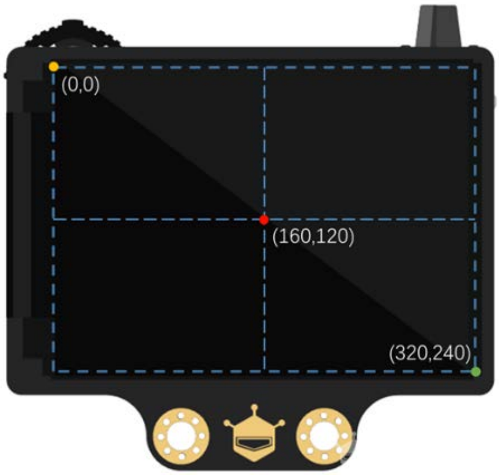
Graphical Block	Function Description
 HuskyLens get X center of ID 1 frame from the result	**Read the parameters of a specified ID frame** Get the parameter of IDx from the requested "result" to see whether the IDx on the screen is learned, if not, it will

	return -1 **Parameter options:** X center, Y center, frame width, frame height (unit: pixel)
 HuskyLens check if ID 1 is learned from the result?	**Determine whether an identified target has been learned** After identifying multiple targets, HUSKYLENS will determine whether the target has been learned so that it can avoid confusion among multiple targets during data invocation. **Value learned target returns:** true **Value unlearned target returns:** false

Step3: Function Analysis

Coordinate Analysis

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



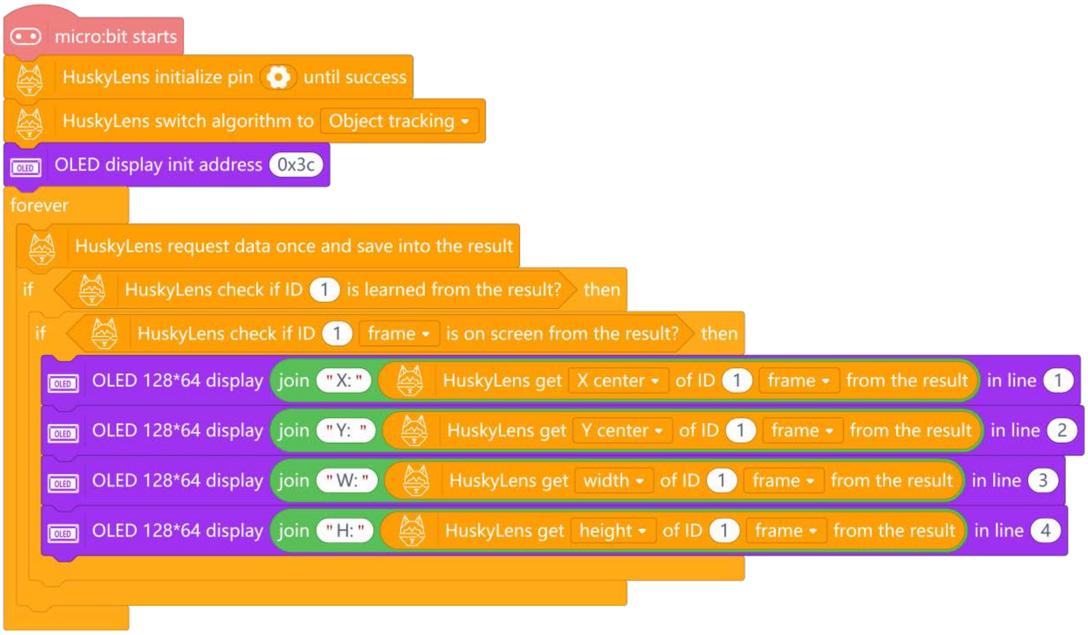
The "X center" and "Y center" of the frame parameter refer to the position of the center point of the recognition frame in the screen coordinate system.

The "width" and "height" of the frame parameters refer to the size of the recognition frame. Under the object recognition function, the recognition frame is a square, so the width and height are equal.

Function Analysis

This task is mainly to display the X center, Y center, width, and height of the frame through OLED.

Sample Program



```
micro:bit starts
    HuskyLens initialize pin ⚙ until success
    HuskyLens switch algorithm to Object tracking ▾
    OLED display init address 0x3c
forever
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
        if HuskyLens check if ID 1 frame ▾ is on screen from the result? then
            OLED 128*64 display join "X: " HuskyLens get X center ▾ of ID 1 frame ▾ from the result in line 1
            OLED 128*64 display join "Y: " HuskyLens get Y center ▾ of ID 1 frame ▾ from the result in line 2
            OLED 128*64 display join "W: " HuskyLens get width ▾ of ID 1 frame ▾ from the result in line 3
            OLED 128*64 display join "H: " HuskyLens get height ▾ of ID 1 frame ▾ from the result in line 4
```

Note: the overall program link is at the end of the article

Operating Effect

We can see the parameters from the OLED. Try moving the object left and right and observe the numerical change of the X; up and down and the change of Y; forward and backward and the change of width and height values.

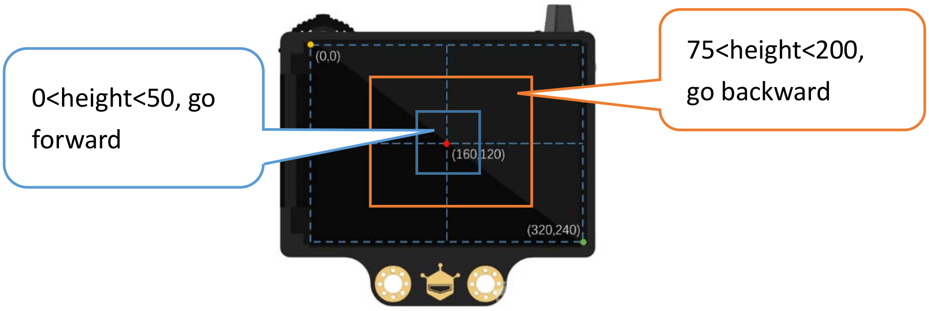
Task3: Front and Back Adjustment

Program Design

Step1: Function Analysis

Maqueen Plus must move forward and backward following the "steps" of the vehicle in front.

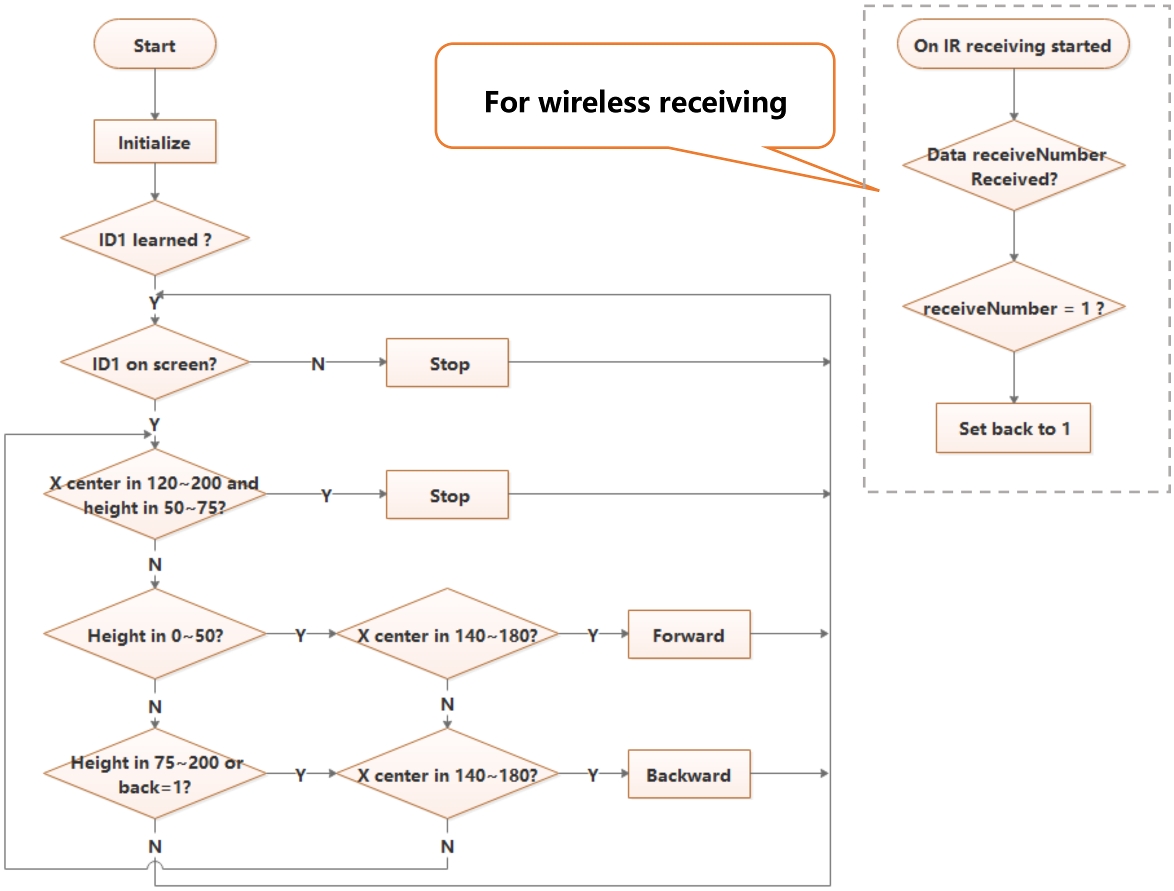
Based on the data obtained from the task 2, we can judge the distance of the target object through the height of the box, and then make the adjustment according to the result.



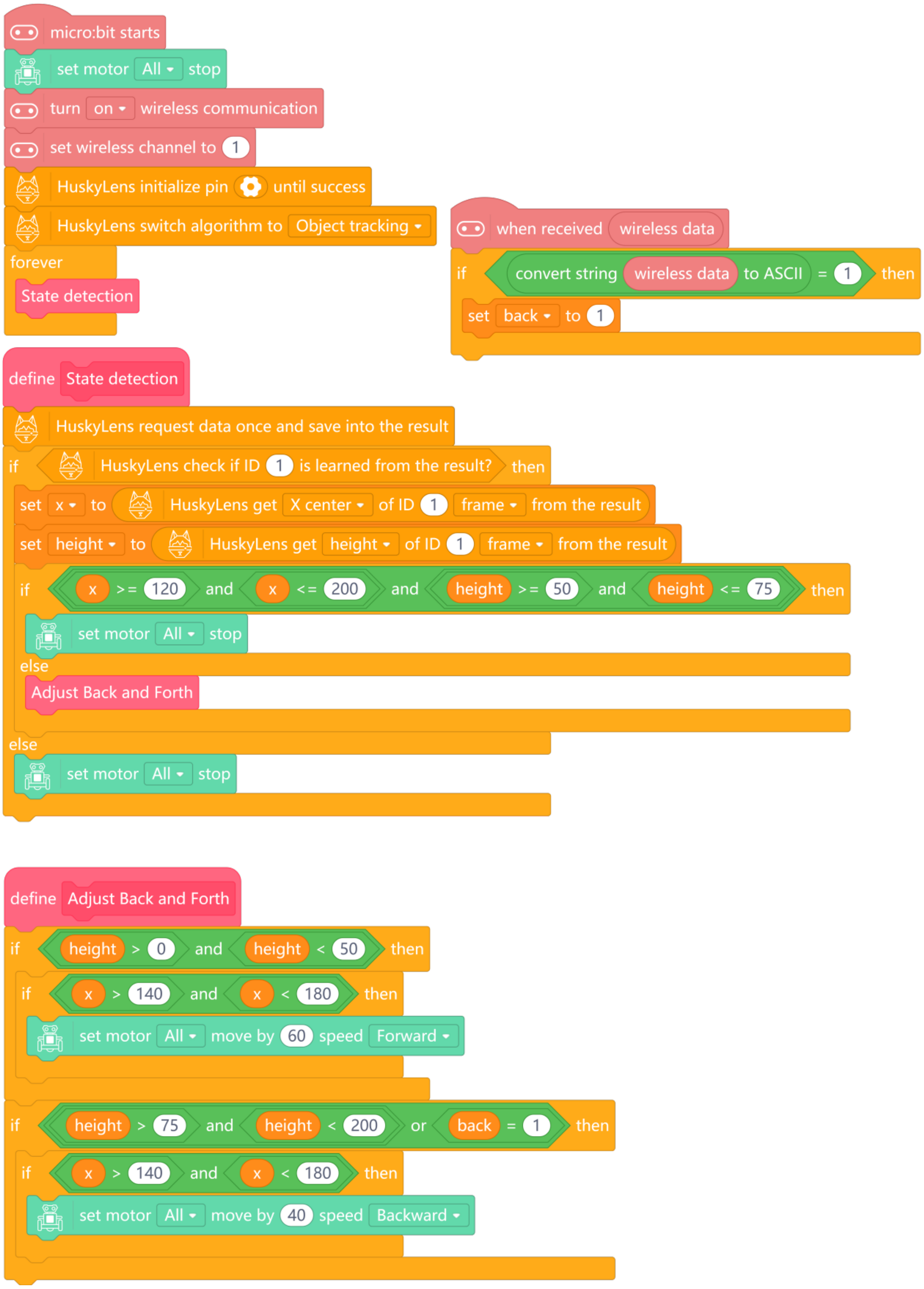
0<height<50, go forward

75<height<200, go backward

Step2: Flowchart Analysis



Sample Program



```
micro:bit starts
    set motor All ▾ stop
    turn on ▾ wireless communication
    set wireless channel to 1
    HuskyLens initialize pin until success
    HuskyLens switch algorithm to Object tracking ▾
    forever
        State detection

when received wireless data
    if convert string wireless data to ASCII = 1 then
        set back ▾ to 1

define State detection
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
        set x ▾ to HuskyLens get X center ▾ of ID 1 frame ▾ from the result
        set height ▾ to HuskyLens get height ▾ of ID 1 frame ▾ from the result
        if x >= 120 and x <= 200 and height >= 50 and height <= 75 then
            set motor All ▾ stop
        else
            Adjust Back and Forth
    else
        set motor All ▾ stop

define Adjust Back and Forth
    if height > 0 and height < 50 then
        if x > 140 and x < 180 then
            set motor All ▾ move by 60 speed Forward ▾
    if height > 75 and height < 200 or back = 1 then
        if x > 140 and x < 180 then
            set motor All ▾ move by 40 speed Backward ▾
```

Note: the overall program link is at the end of the article

Operating Effect

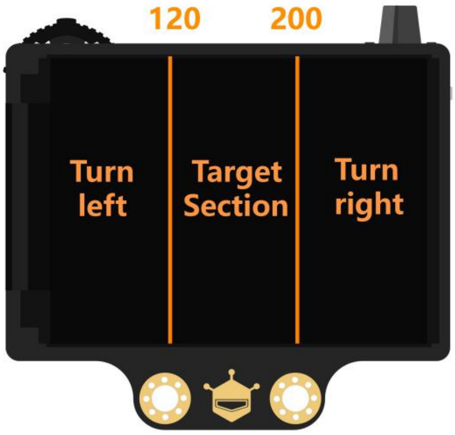
After learning the target pattern behind the vehicle in front, Maqueen Plus will automatically follow the pattern to move forward and backward, and always stay within a suitable distance range. If the vehicle in front moves, Maqueen Plus will follow it.

Task4: Left and right Adjustment

Program Design

Step1: Function Analysis

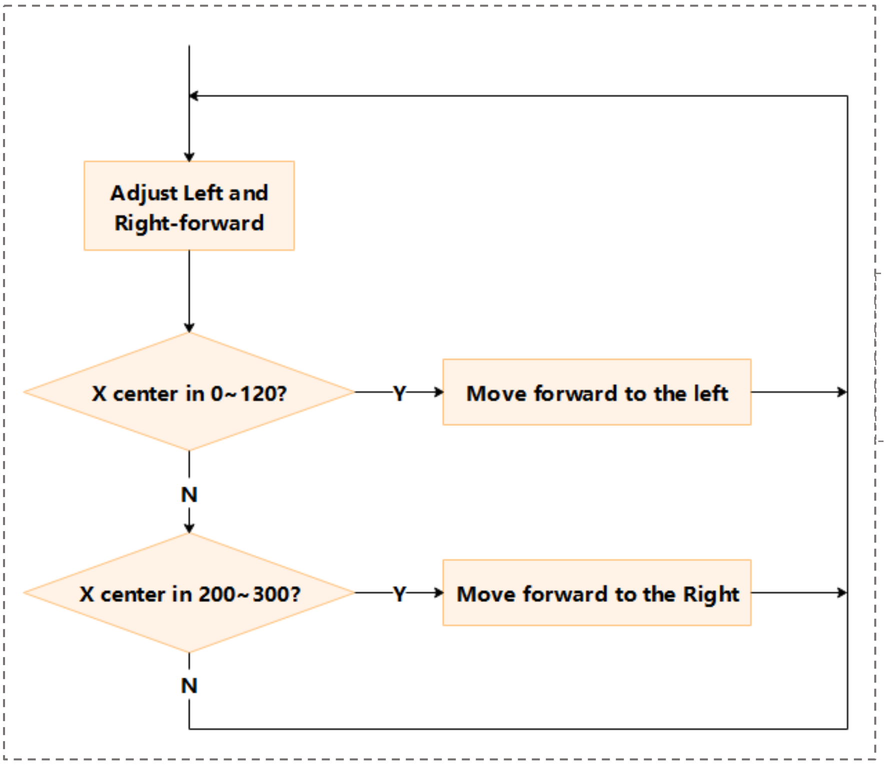
As shown in the picture, the screen is divided into 3 sections according to the X axis of the camera screen coordinate system, and the middle section is the target section. The camera continuously detects the state of the target object. When Maqueen Plus is moving forward, and its X center is between 120 and 200, which means that the target is in the center of the field of view, and there is no need to adjust its position; when its X center is between 0 and 120, Maqueen Plus moves to the left; when its X center is between 200 and 320, Maqueen Plus moves to the right. When Maqueen Plus moves back, and its X center is between 0 and 140, it goes back to the right; when its X center is between 180 and 320, it goes backward to the left.



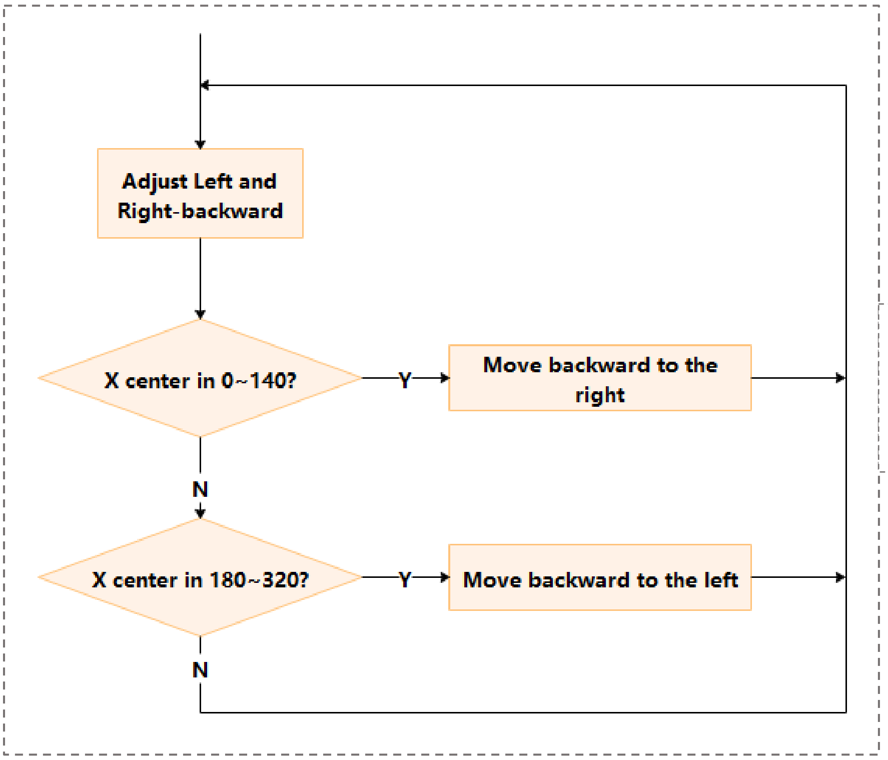
Step2: Flowchart Analysis

Based on task 2, the direction of the vehicle in front is judged by the position of the X center point. Maqueen Plus makes left-right adjustments based on the results.

Note: Left and right adjustment is divided into left and right adjustment when forward (left and right forward adjustment), and left and right adjustment when backward (left and right backward adjustment).

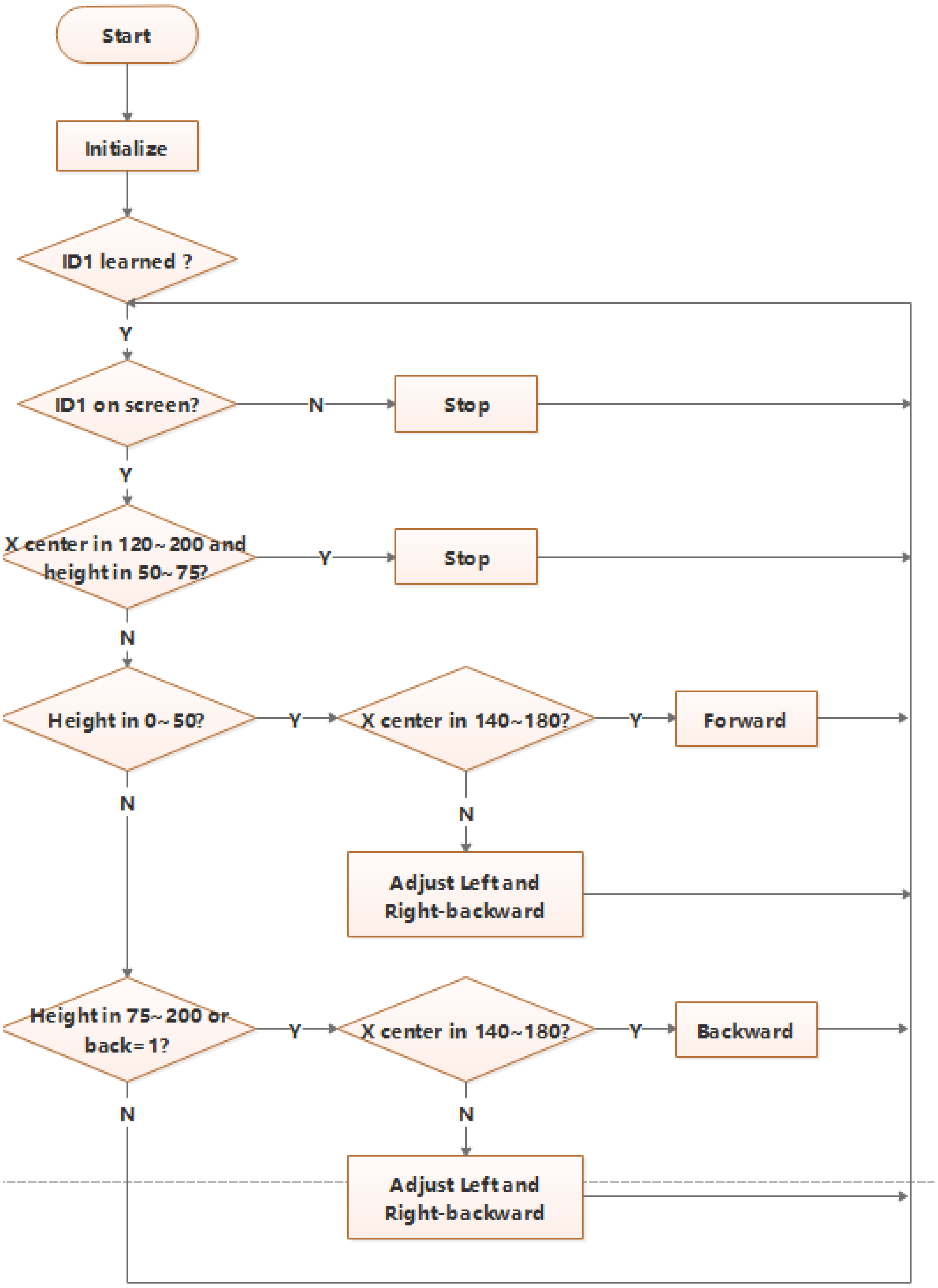


Left and right forward adjustment


Adjust Left and Right-backward
X center in 0~140?
Y
Move backward to the right
N
X center in 180~320?
Y
Move backward to the left
N

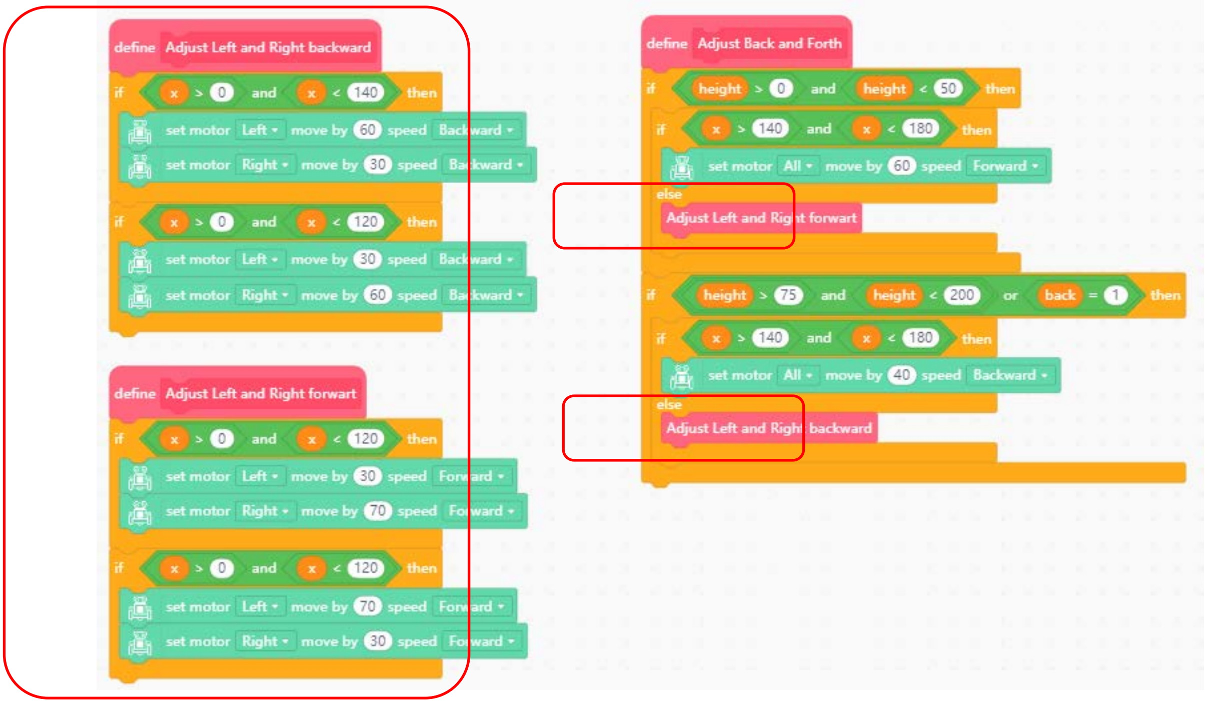
Left and right backward adjustment

The overall flowchart is as follows:



Sample Program

Based on task 3, the functions of "adjust left and right forward" and "adjust left and right backward" are added. The final program is as follows:



```
define Adjust Left and Right backward
if x > 0 and x < 140 then
    set motor Left move by 60 speed Backward
    set motor Right move by 30 speed Backward
if x > 0 and x < 120 then
    set motor Left move by 30 speed Backward
    set motor Right move by 60 speed Backward

define Adjust Left and Right forwart
if x > 0 and x < 120 then
    set motor Left move by 30 speed Forward
    set motor Right move by 70 speed Forward
if x > 0 and x < 120 then
    set motor Left move by 70 speed Forward
    set motor Right move by 30 speed Forward

define Adjust Back and Forth
if height > 0 and height < 50 then
    if x > 140 and x < 180 then
        set motor All move by 60 speed Forward
    else
        Adjust Left and Right forwart
if height > 75 and height < 200 or back = 1 then
    if x > 140 and x < 180 then
        set motor All move by 40 speed Backward
    else
        Adjust Left and Right backward
```

Note: the overall program link is at the end of the article

Operating Effect

After learning the target behind the vehicle in front, Maqueen Plus will automatically follow the target to move forward, backward, left and right, always keeping the object frame in the center of the screen, and the two vehicles will always keep a proper distance.

Project Development

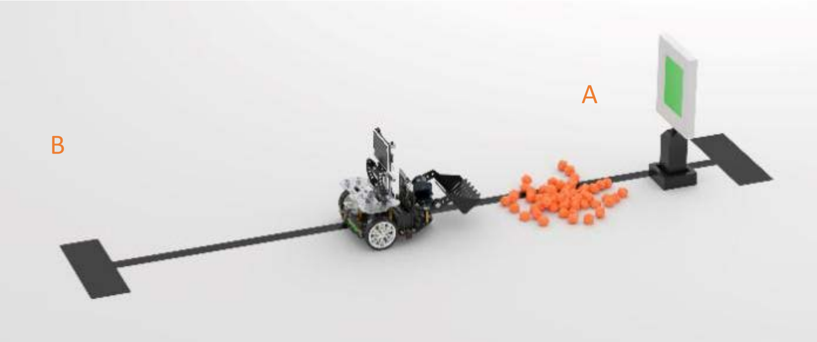
In this project, we use two vehicles for experiments. Can you use more vehicles to follow one by one? Let them always follow the locomotive just like a train.

Note: During the experiment, please adjust the wheel speed according to the actual situation.

Project4: Fixed-Point Transportation

With the development of industrial automation, many large factories now start to use intelligent robots to move workpieces, assemble, and transfer goods on production line.

Think about it, can Maqueen Plus also become an intelligent robot and move the goods from point A to point B?

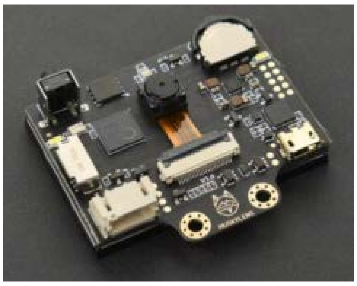
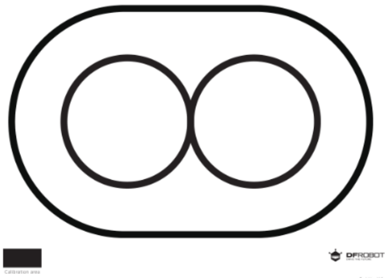
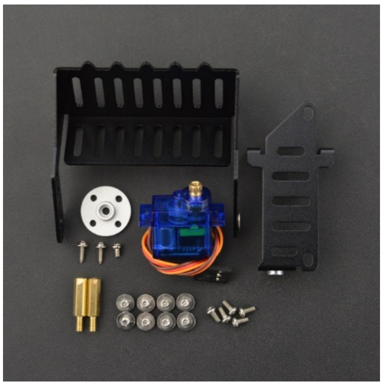


Function

This project mainly uses the color recognition of HUSKYLENS and the line tracking function of Maqueen Plus. First turn on the line tracking function and let Maqueen Plus drive along the black line. After HUSKYLENS recognizes the color, turn off the line tracking function, and then locate the goods according to the recognition result. When Maqueen Plus reaches the designated location, it will use the loader to scoop up the goods, turn around and turn on the line tracking function to transport the goods to the designated unloading point. This simply realizes the function of fixed-point transportation.

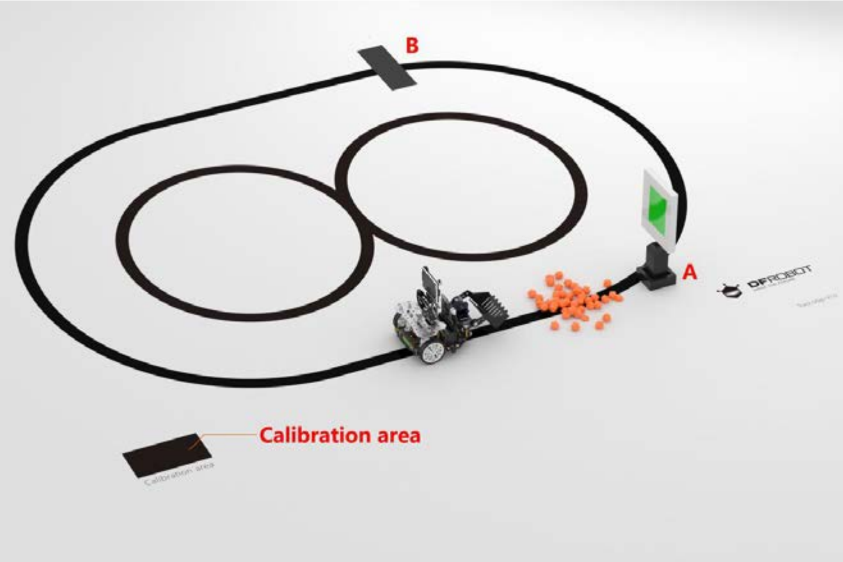
Bill of Materials

micro:bit ×1	**Maqueen Plus ×1**

	
HUSKYLENS ×1	**Tracking Map ×1**
Color Paperboard	**Black Electrical Tape**
	
Loader Accessories	

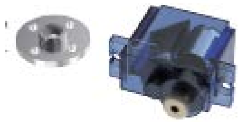
Hardware Connection

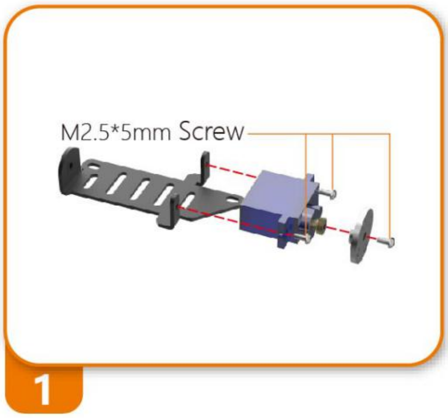
1. Map Designation: Choose a position on the oval map as the unloading point (point B), then mark it with black tape (the marked area should be larger than the black calibration area), and then choose another position as the loading point (point A), and place the color identification card (there should be no other colors behind the color identification card). Place goods in front of point A.

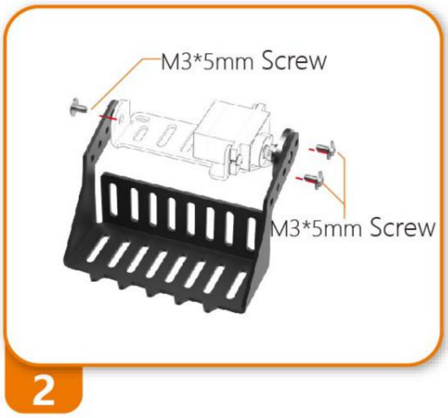


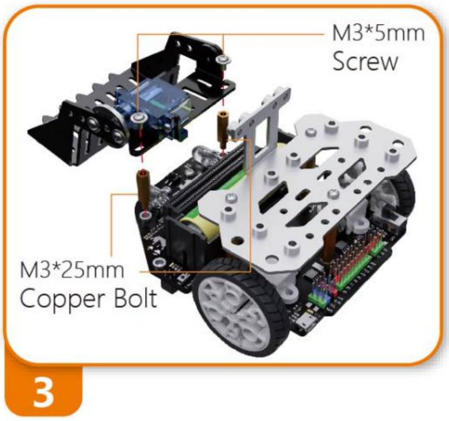
2. Assembly Diagram: Install the Loader and HUSKYLENS to Maqueen Plus.

Bill of Materials:

		
Loader Bucket X1	**Loader Servo Panel X1**	**9g Metal Servo X1**
M3*15 Copper Pillar X2	**M3*5mm Nut X8**	**M2.5*5mm Nut X5**

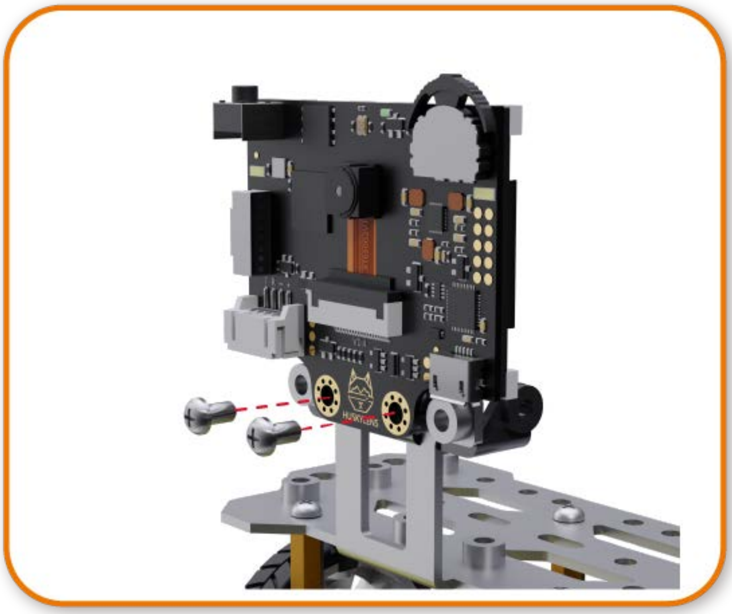




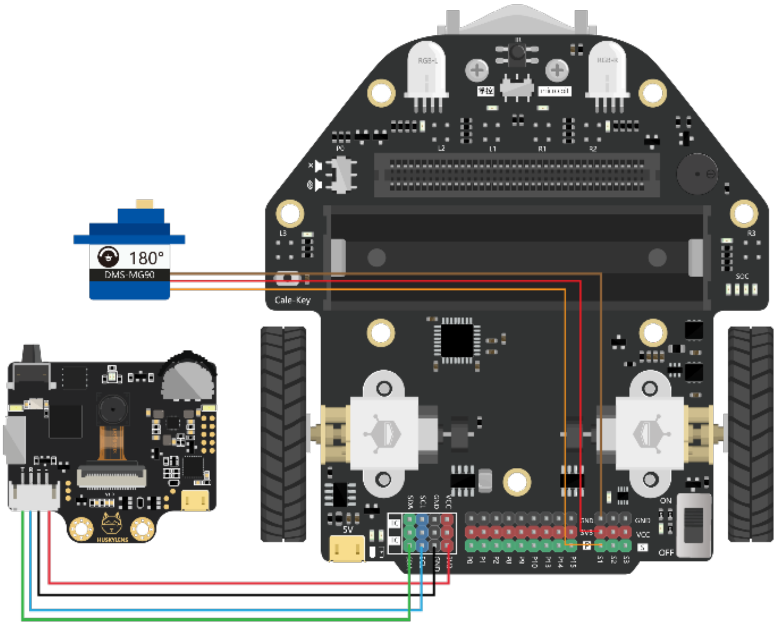


Installation Steps of HUSKYLENS:

After finishing installing the loader, install HUSKYLENS to the expansion bracket on the back of Maqueen Plus.



3. Connection Diagram: Connect HUSKYLENS to the I2C interface and the servo to the S1 interface. Pay attention to the order of wiring, do not connect wrongly or reversely.

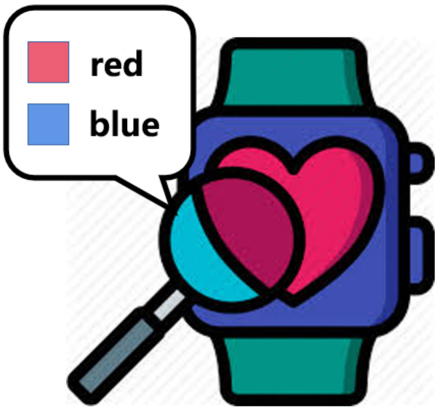


Knowledge Field

Nowadays, automated production has become a development trend. Machine vision, as the eyes of "robots", is particularly paramount.

As one of the important technical directions, color recognition has experienced multiple generations of technology upgrades. This project is based on the color recognition function of the HUSKYLENS sensor.

1. What Is Color Recognition?



What is color recognition? First, we have to learn **what color is.**

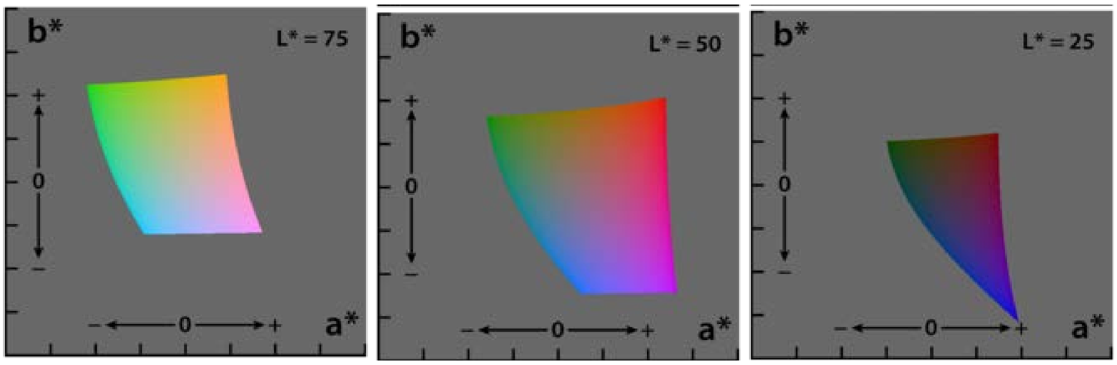
Color is a visual effect on light produced by our eyes, brain and our life experience. The light we see with the naked eye is produced by electromagnetic waves which are with a very narrow wavelength range. Electromagnetic waves of different wavelengths show different colors.

Color recognition is to recognize and distinguish color attributes based on different brightness.

2. Principles of Color Recognition

Color recognition is based on **Lab color space**, with dimension L representing brightness, a and b representing opposite dimensions of color, which is based on the CIE XYZ color space coordinates of nonlinear compression.

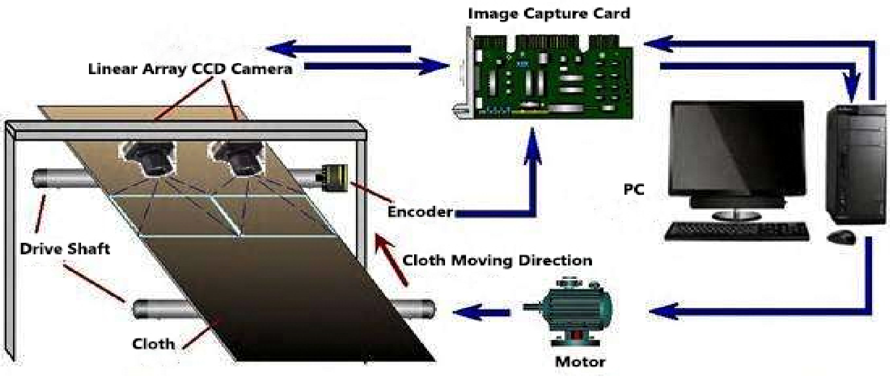
Compare the Lab parameters of the recognized and learned colors. When the two colors match within a certain error range, they are judged to be the same color.



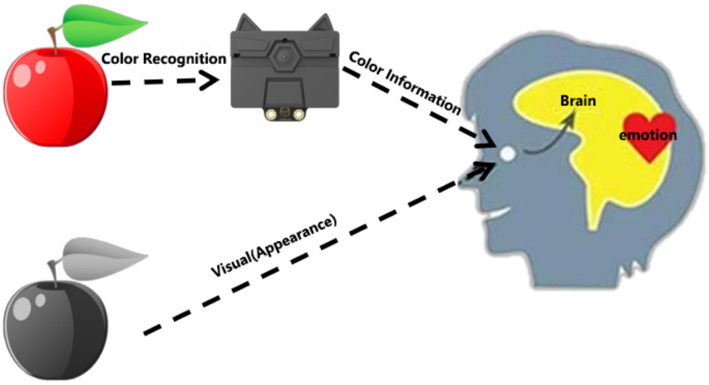
In our usual use of color recognition, the hue and saturation in the color attributes of the same module are fixed, but the brightness will change due to the influence of the ambient brightness, so you must try to ensure that when using the HUSKYLENS color recognition function the environment brightness during learning and recognition is consistent with that during actual work.

3. Application Scenario of Color Recognition

Industry: Color recognition is now widely used in industries such as printing, coatings and textiles for color monitoring and calibration.



Personal Life: Help people with color weakness or visual impairment recognize and enhance their understanding of color.



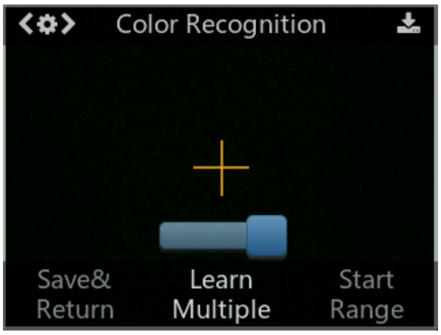
4. Demonstration of Color Recognition of HUSKYLENS

The color recognition function in the HUSKYLENS sensor uses the sensor's built-in algorithm to identify the ID of different colors and feed them back to the main control board by learning and recording different colors.

The HUSKYLENS sensor is set by default to learn, recognize and track only one color, but we can set it to recognize multiple colors.

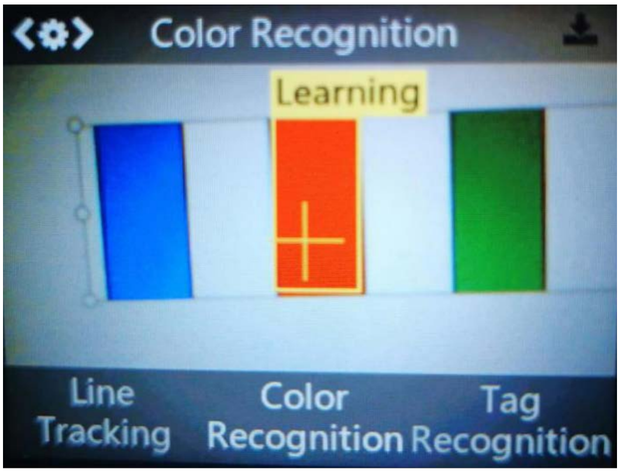
1. Choose "Color Recognition" function

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

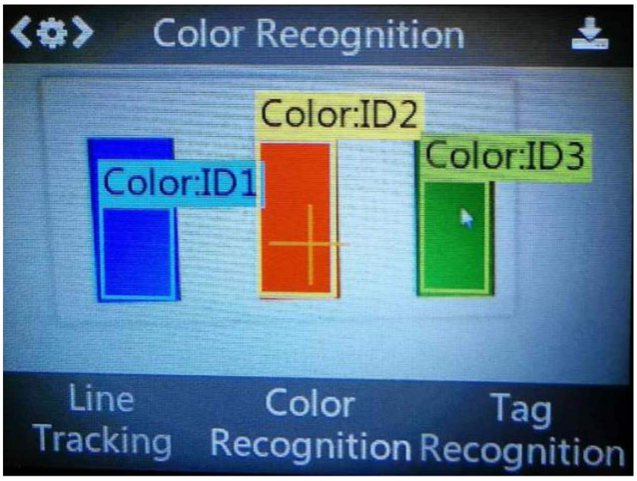


2. Color Learning

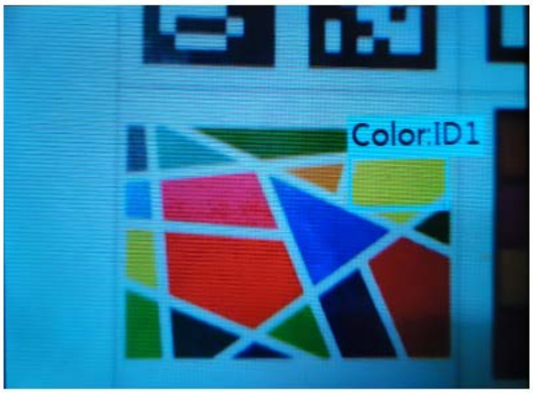
Point the "+" symbol at the color block, and long press the "learning button". A yellow frame will be displayed on the screen, indicating that HuskyLens is learning the color.

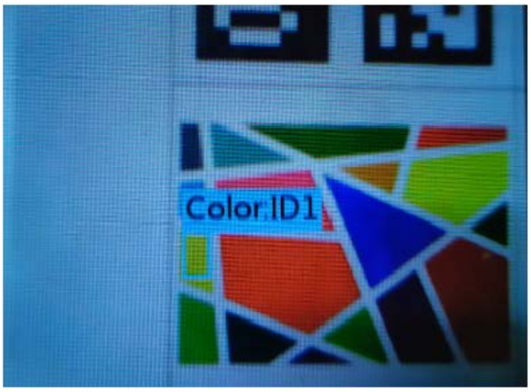


After finishing the study, when encountering the same or similar color blocks, a frame with an ID will be automatically displayed on the screen, and the size of the frame changes with the size of the color blocks.



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.

5. What Is Line Tracking

Line tracking refers to the process of objects driving along a designated route. The 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 the load system, and a multi-field information fusion of machine vision-electromagnetic field-GPS-GIS as the robot's autonomous movement navigation system, an embedded computer as the software and hardware development platform of the control system.

6. Principles of Line Tracking

Line tracking means that the robot judges the position deviation between the robot and the line by detecting the intensity of the light reflected (the light is reflected on the white or light-color light surface; the light is absorbed on the black surface). For example: the line tracking sensor on the right detects white, indicating that the robot has shifted to the right; the line tracking sensor on the left detects white, indicating that the robot has shifted to the left; if it does not detect white, it means that the robot is not yaw and the robot will go straight.

Project Practice

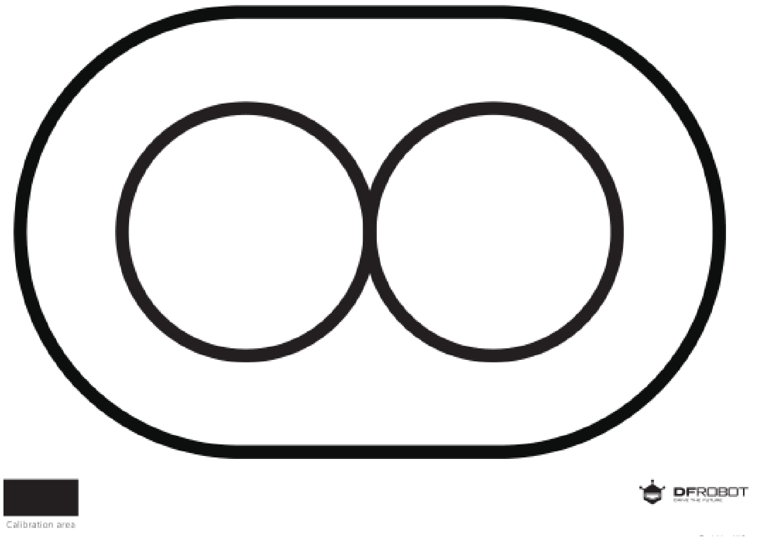
How is the color recognition of HUSKYLENS used in the fixed-point transportation project? How to make Maqueen Plus locate the goods based on the recognition results of HUSKYLENS? After loading the goods, how to accurately transport the goods to the unloading point (point B)? Now, we are going to finish this project step by step.

First, design the Maqueen Plus line tracking function so that Maqueen Plus will drive along the black line. Then, learn to use the color recognition function of HUSKYLENS, adjust the position of Maqueen Plus and the loader to lift goods according to the orientation of the color card; finally, combine the line tracking and the color recognition function of HUSKYLENS to realize the fixed-point transportation from point A to point B.

Note: All the missions should be completed with a fully-charged battery.

Task1: Basic Tracking

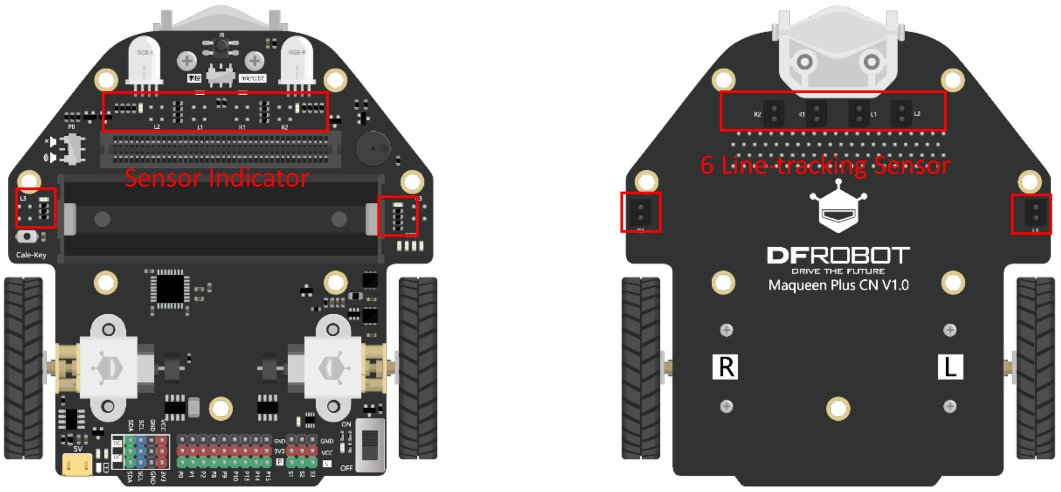
The following is a map for reference.



Program Design

Step1: Function Analysis

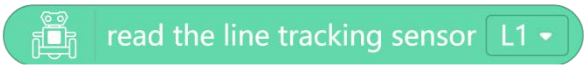
For this kind of point to point accurate transportation, we can directly use the line tracking sensors on Maqueen Plus.



For Maqueen Plus, there are 6 line tracking sensors at the bottom, which can be used to detect black lines.

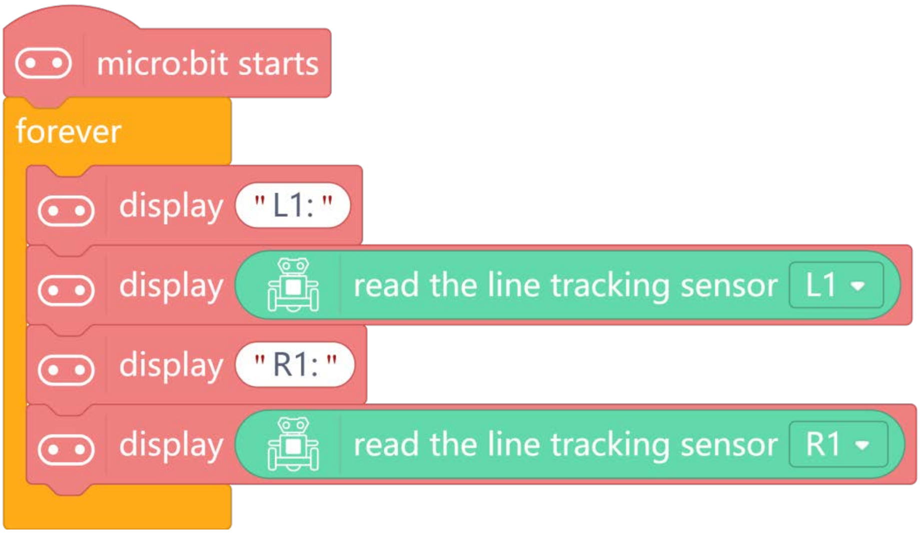
When the sensor faces a white background, the sensor’ s indicator is off, and the line tracking sensor detection value is 0; when the sensor faces a black line, the sensor’ s indicator is on, and the line tracking sensor detection value is 1.

Step2: Instruction Learning

Instruction Learning	Function Description
 read the line tracking sensor L1 ▾	**Read the line tracking sensor** **Function Description:** Read the value of the line tracking sensor, the feedback value is 0 or 1, and 1 means that it is on the black line. Select the position in the drop-down box, L1, L2, L3, R1, R2, R3 are consistent with the marks on the bottom of the car.

Step3: Line Tracking Sensor Testing

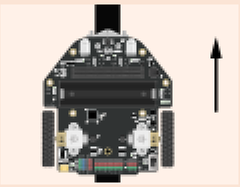
Test the program through the feedback value output by LED matrix screen.

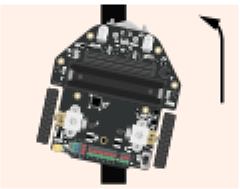


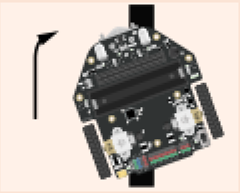
```
micro:bit starts
forever
    display "L1:"
    display read the line tracking sensor L1 ▾
    display "R1:"
    display read the line tracking sensor R1 ▾
```

Test Results :

Status Diagram	Status Description	Output Value	Maqueen Plus Movement

	Both L1 and R1 are on the black line	L1 = 1
		R1 = 1
			Go straight

	L1 sensor is on the black line
	R1 sensor is not on the black line
		L1 = 1
		R1 = 0
			Turn left

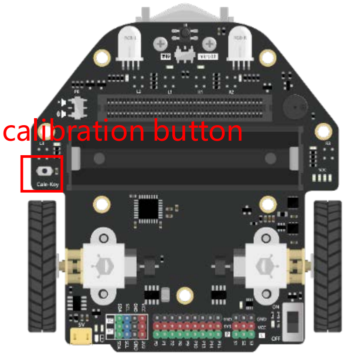
	L1 sensor is not on the black line
	R1 sensor is on the black line
		L1 = 0
		R1 = 1
			Turn right

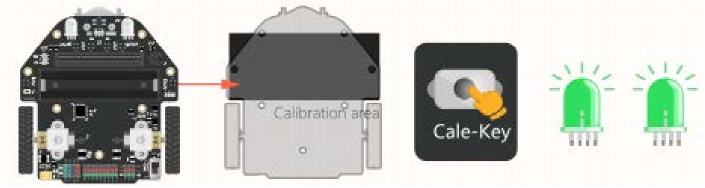
If the program does not work properly, troubleshoot the following problems:

① The black printed by the printer may not be correctly recognized. The black tape and printed map from DFRobot can be used normally.

② Ambient light may affect the line-tracking sensor. When the light changes greatly, recalibrate the sensors.

Method of calibration: Maqueen Plus comes with a one-button calibration function. The calibration button is as shown in the figure below. When using it, make sure that all line tracking sensors are in the black calibration area. Press the calibration button for 1 second, the two RGB light in front of the car flashes green, indicating that the calibration is completed. Release the button to complete the calibration.

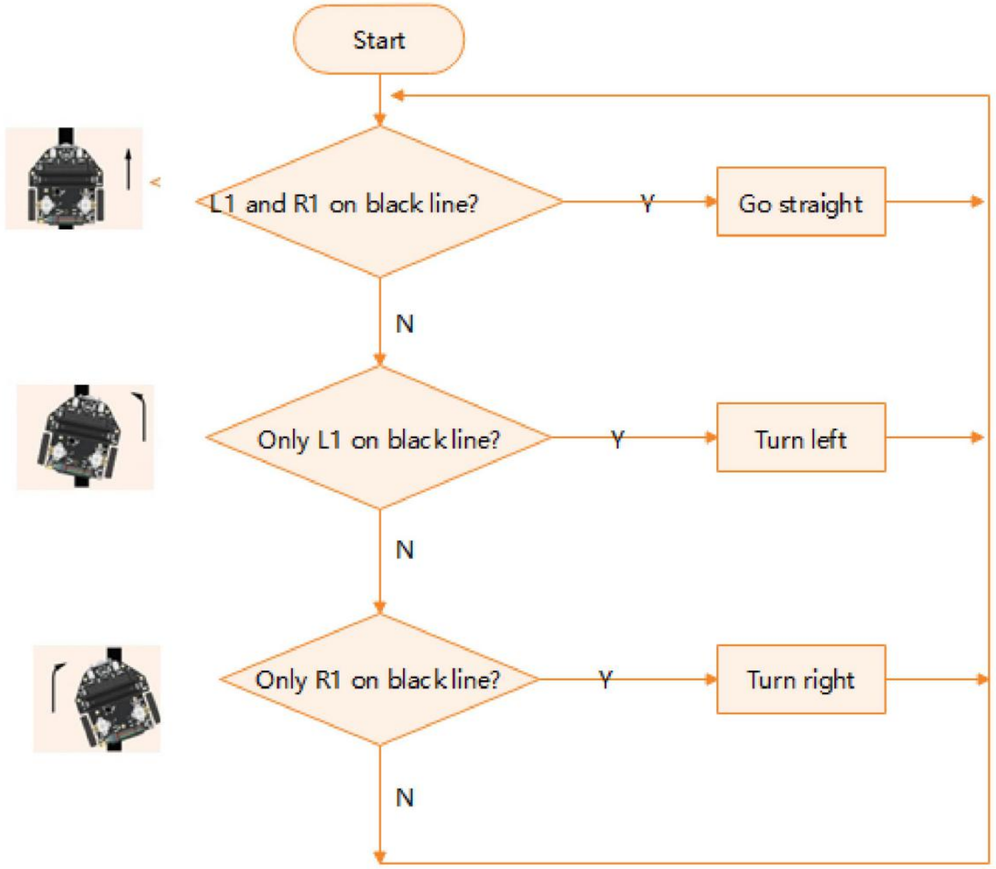




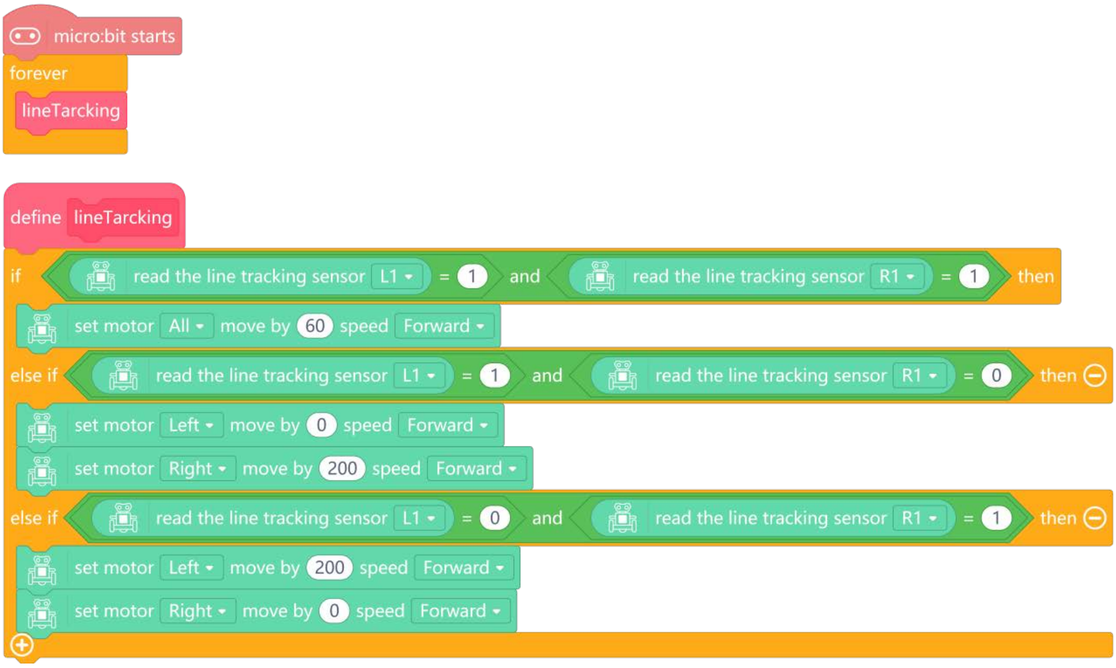
Principle of line tracking sensor: Each line tracking sensor consists of an **infrared transmitter** and an **infrared receiver**. Because it is often used to control the robot to walk along the line, it is also called the line tracking sensor. The infrared transmitter continuously emits infrared light to the ground. If the infrared light is reflected (for example, when it encounters a white or other light-color surface), the receiver will receive the infrared signal, the output value will be 0, and the sensor indicator will go out; if the infrared light is absorbed or it cannot be reflected, the receiver cannot receive the infrared signal, the output value will be 1, and the indicator light will go on.

Step4: Flowchart Analysis

First, we use L1 and R1 to track the line. Select the ellipse line tracking map. If you want to design a map by yourself, note that the width of the black line needs to be 2cm and ensure that both L1 and R1 can be on the black line.



Sample Program



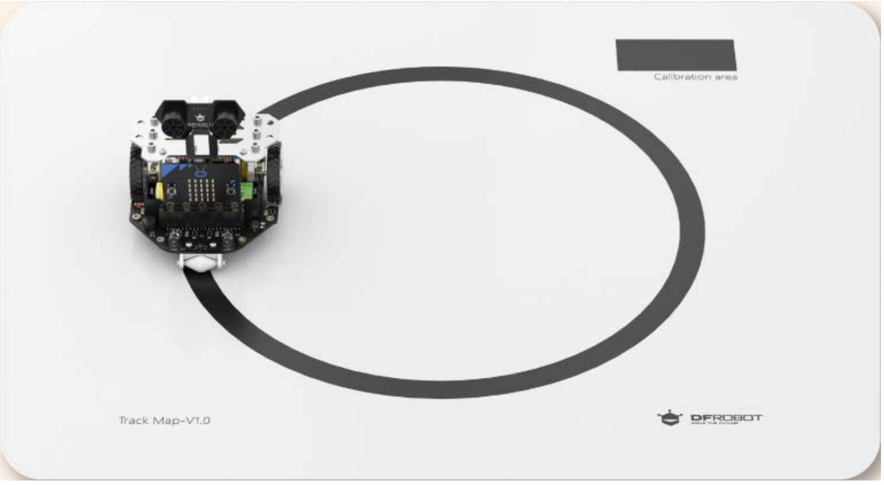
```
micro:bit starts
forever
    lineTarcking

define lineTarcking
if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 1 then
    set motor All move by 60 speed Forward
else if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 0 then
    set motor Left move by 0 speed Forward
    set motor Right move by 200 speed Forward
else if read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 1 then
    set motor Left move by 200 speed Forward
    set motor Right move by 0 speed Forward
```

Note: the overall program link is at the end of the article

Note: Adjust the speed according to the situation of your own car to achieve the best effect.

Operating Effect

When the program is run, Maqueen Plus will automatically drive along the black line.

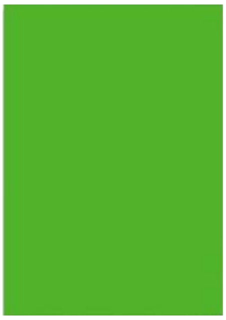


Task2: Color Recognition

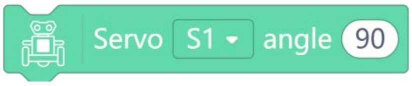
Program Design

Step1: Learning and Recognition

Choose a color paper card you like and let HUSKTLENS learn it. Stick the card on a white background plate (to avoid other color interference when recognizing).



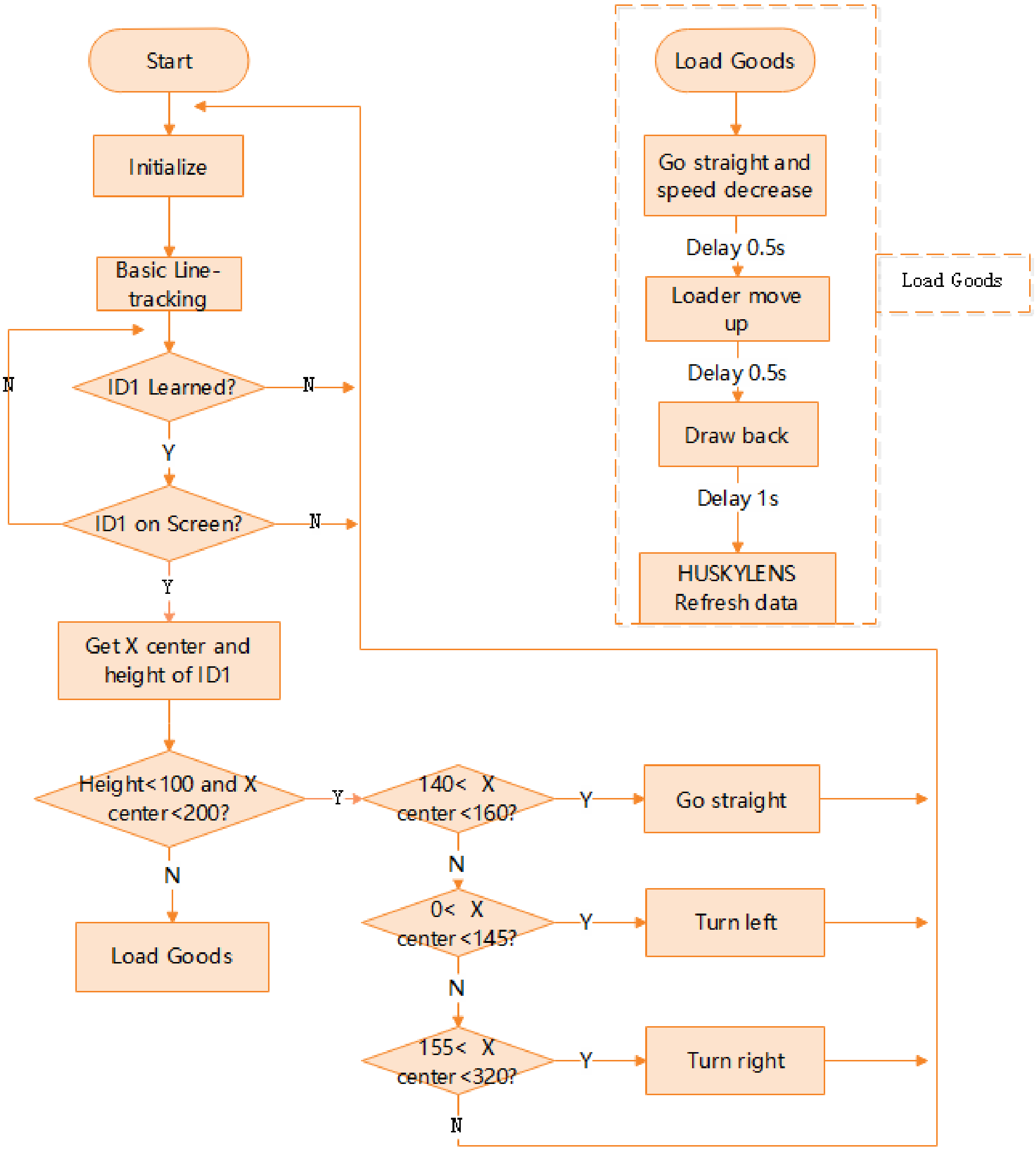
Step2: Instruction Learning

Graphical Block	Function Description
 Servo S1 angle 90	**Servo Driver** **Function Description:** Control the rotation angle of the servo connected to the interfaces S1, S2 and S3. **Interface:** S1, S2, S3 **Angle:** 0~180° (It is recommended that the angle should

	not exceed 170°)

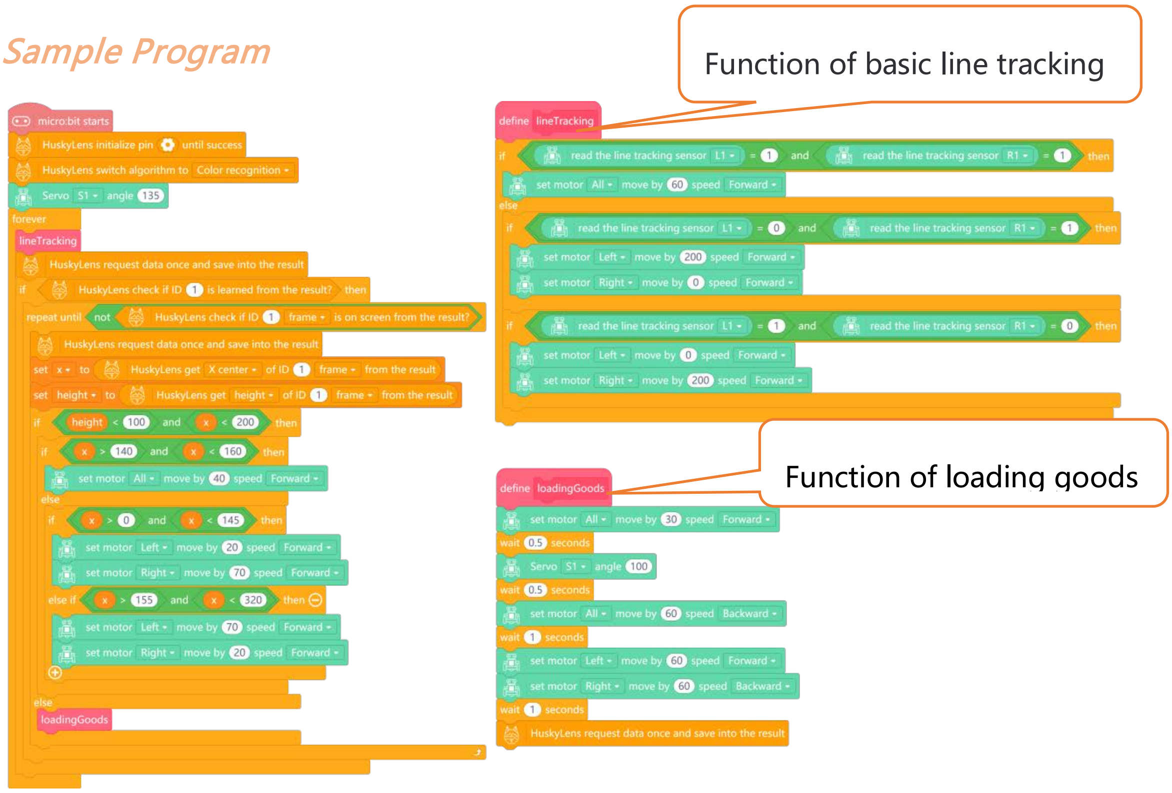
Step3: Function Analysis

After task 1 is completed, Maqueen Plus is able to drive along the black lines. Next, we will use the color recognition function to locate the position of the goods. When Maqueen Plus recognizes the color ID1, it will turn off the line tracking function, and adjust the location of Maqueen Plus by judging the position and distance to the color card. When Maqueen Plus arrives within the specified range, the loader will start to work. The program flow chart is as follows:



Note: Because the angle of installation may be different for each person, we need to find 0 degrees in advance, and then initialize the servo angle according to the situation.

Sample Program



Function of basic line tracking

Function of loading goods

```
micro:bit starts
HuskyLens initialize pin until success
HuskyLens switch algorithm to Color recognition
Servo S1 angle 135
forever
  lineTracking
  HuskyLens request data once and save into the result
  if HuskyLens check if ID 1 is learned from the result? then
    repeat until not HuskyLens check if ID 1 frame is on screen from the result?
      HuskyLens request data once and save into the result
      set x to HuskyLens get X center of ID 1 frame from the result
      set height to HuskyLens get height of ID 1 frame from the result
      if height < 100 and x < 200 then
        if x > 140 and x < 160 then
          set motor All move by 40 speed Forward
        else
          if x > 0 and x < 145 then
            set motor Left move by 20 speed Forward
            set motor Right move by 70 speed Forward
          else if x > 155 and x < 320 then
            set motor Left move by 70 speed Forward
            set motor Right move by 20 speed Forward
      else
        loadingGoods

define lineTracking
if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 1 then
  set motor All move by 60 speed Forward
else
  if read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 1 then
    set motor Left move by 200 speed Forward
    set motor Right move by 0 speed Forward
  if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 0 then
    set motor Left move by 0 speed Forward
    set motor Right move by 200 speed Forward

define loadingGoods
set motor All move by 30 speed Forward
wait 0.5 seconds
Servo S1 angle 100
wait 0.5 seconds
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 1 seconds
HuskyLens request data once and save into the result
```

Note: the overall program link is at the end of the article

Operating Effect

The Maqueen Plus initially drives along the black line. After HUSKYLENS recognizes the color of the card, it will use the recognition result to adjust the distance. When the identified X center and frame height do not meet the conditions (when Maqueen Plus is very close to the color paper), the loader will work.

Task3: Set up the scene and improve the program

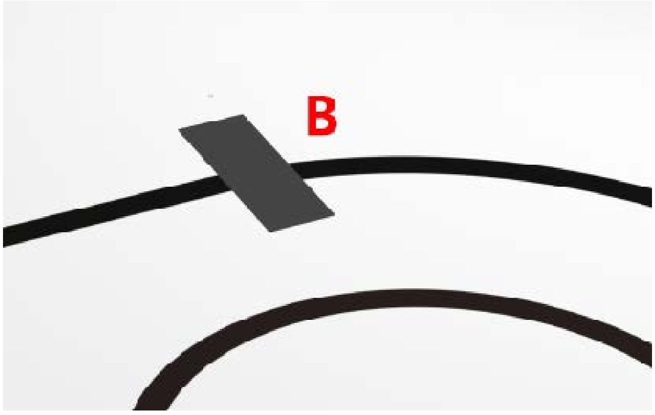
Program Design

Step1: Function Analysis

After completing task two, the Maqueen Plus is able to locate the goods through the results of color recognition. Then we need to transport the picked goods to point B along the black track.

After reaching point B, the car turns around and drives along the black track to point A to load goods. Just like this, it will go back and forth between A and B.

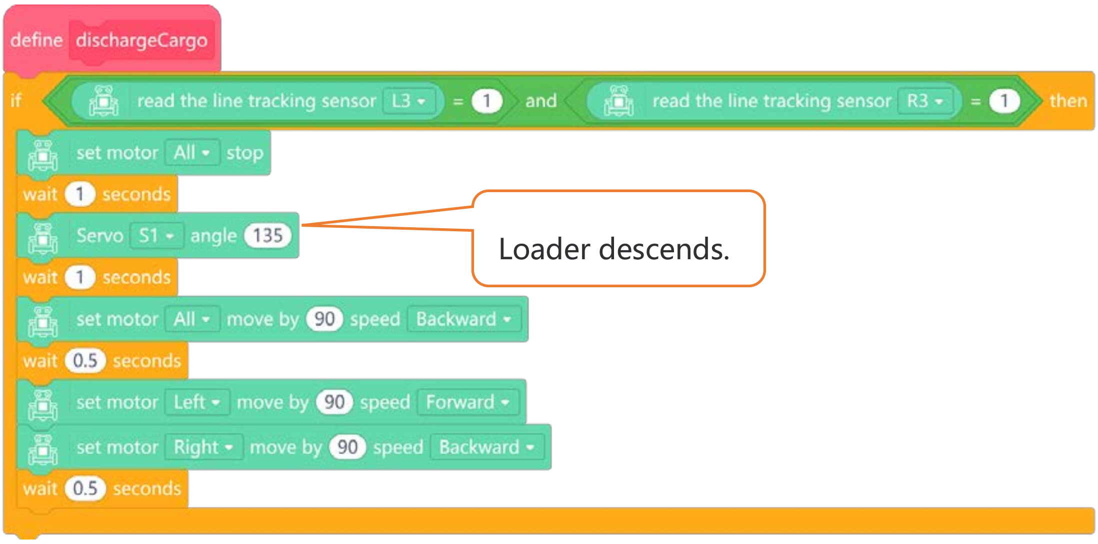
For the car to accurately find point B, we need to use black tape to stick a piece of unloading point that is the same size as the calibration area at point B (mainly covering the L3 and R3 sensor probes).



Use the L3 and R3 line tracking sensors on the Maqueen Plus to recognize the unloading point (point B). After reaching point B, unload the cargo.

Step2: Point B Judgment Program

Create a new function to discharge cargo. When the L3 and R3 line tracking sensors detect black, the Maqueen Plus stops. After putting down the cargo, it moves back and turns around.



```
define dischargeCargo
if read the line tracking sensor L3 = 1 and read the line tracking sensor R3 = 1 then
    set motor All stop
    wait 1 seconds
    Servo S1 angle 135
    wait 1 seconds
    set motor All move by 90 speed Backward
    wait 0.5 seconds
    set motor Left move by 90 speed Forward
    set motor Right move by 90 speed Backward
    wait 0.5 seconds
```

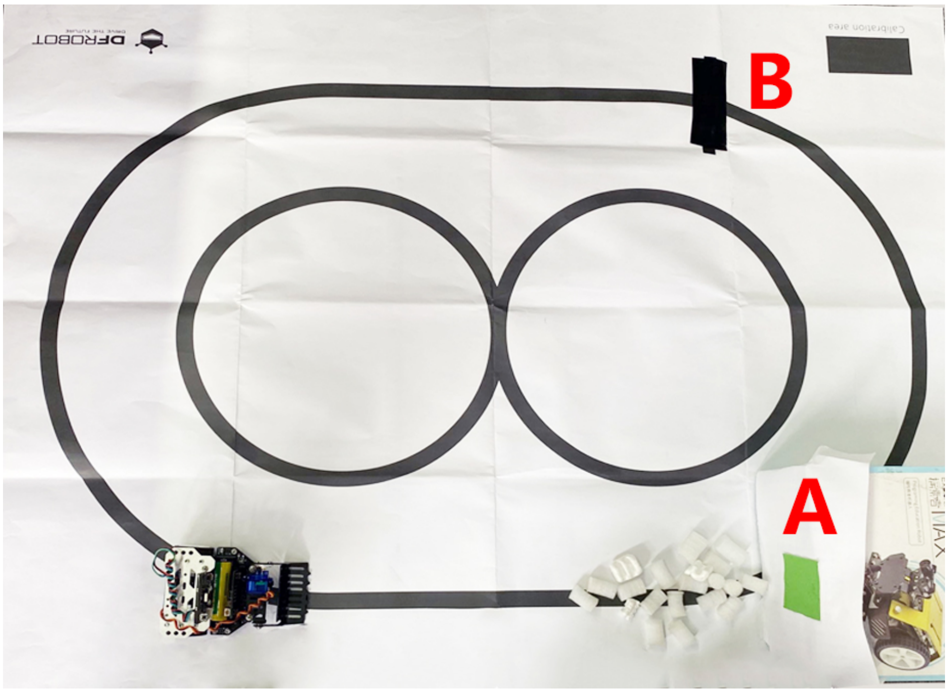
Loader descends.

Note: Pay attention to the pause time when debugging the program.

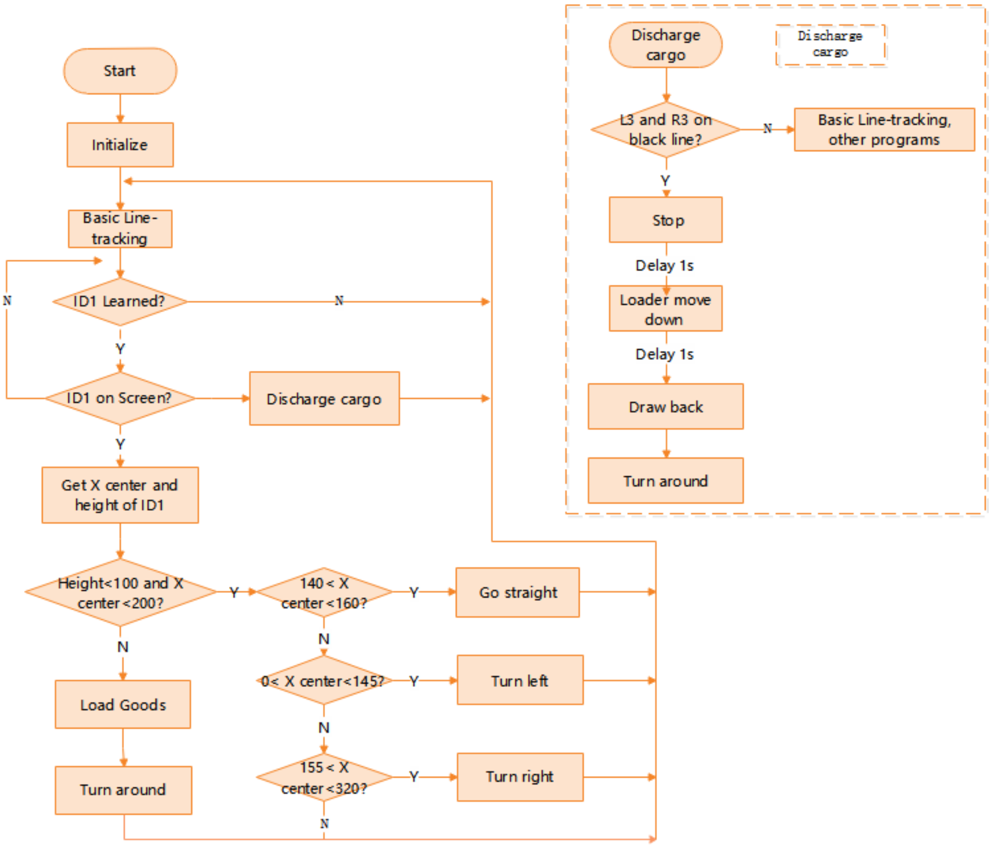
Step3 Flowchart Analysis

We can build the scene according to the following figure. The path from point A to point B is: if the vehicle starts from point B (L3, R3 cannot detect black), it will drive along the black track, and when HUSKYLENS recognizes the color, it will turn off the line tracking function. The distance between Maqueen Plus and the color card is controlled by the result of HUSKYLENS recognition.

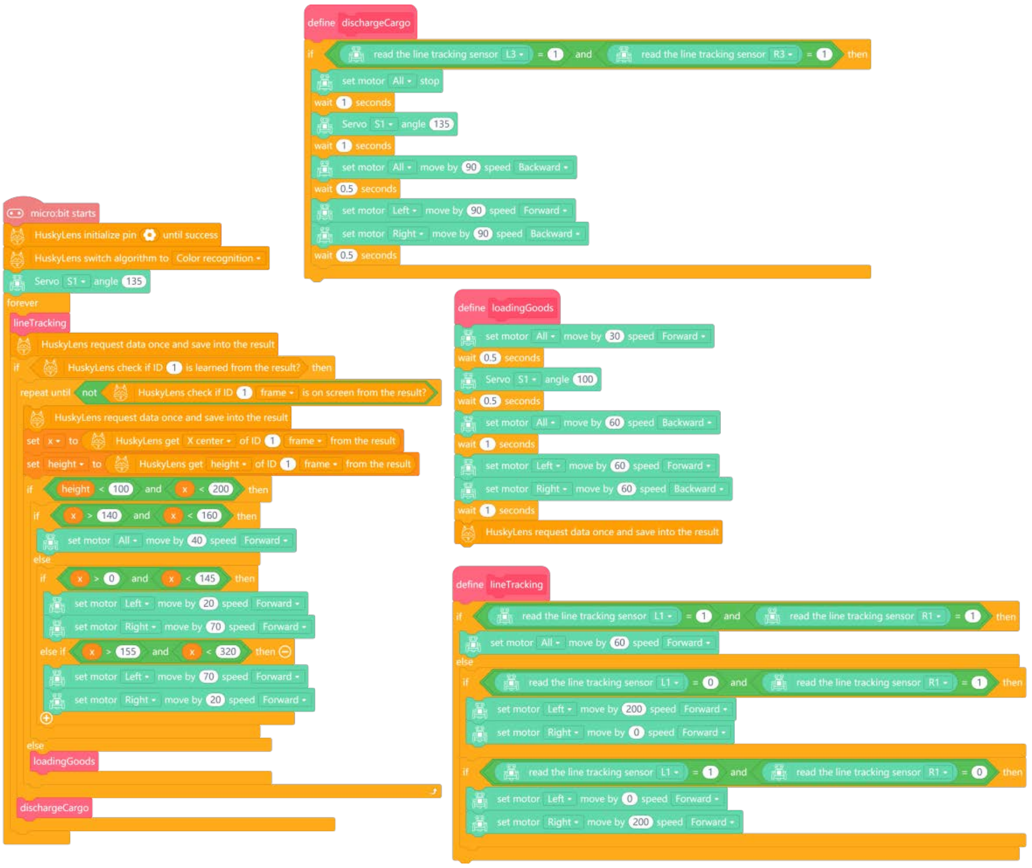
When Maqueen Plus reaches the designated position, it picks up the goods and turns around and drives along the black track. After reaching point B, it will unload the cargo, then turn around and repeat the previous steps.



The program flowchart is as follows:



Sample Program

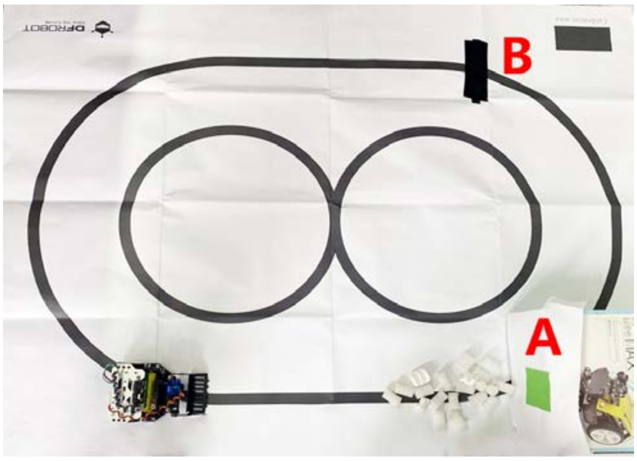


```
define dischargeCargo
if read the line tracking sensor L3 = 1 and read the line tracking sensor R3 = 1 then
    set motor All stop
    wait 1 seconds
    Servo S1 angle 135
    wait 1 seconds
    set motor All move by 90 speed Backward
    wait 0.5 seconds
    set motor Left move by 90 speed Forward
    set motor Right move by 90 speed Backward
    wait 0.5 seconds

micro:bit starts
HuskyLens initialize pin until success
HuskyLens switch algorithm to Color recognition
Servo S1 angle 135
forever
    lineTracking
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 is learned from the result? then
        repeat until not HuskyLens check if ID 1 frame is on screen from the result?
            HuskyLens request data once and save into the result
            set x to HuskyLens get X center of ID 1 frame from the result
            set height to HuskyLens get height of ID 1 frame from the result
            if height < 100 and x < 200 then
                if x > 140 and x < 160 then
                    set motor All move by 40 speed Forward
                else
                    if x > 0 and x < 145 then
                        set motor Left move by 20 speed Forward
                        set motor Right move by 70 speed Forward
                    else if x > 155 and x < 320 then
                        set motor Left move by 70 speed Forward
                        set motor Right move by 20 speed Forward
            else
                loadingGoods
        dischargeCargo

define loadingGoods
set motor All move by 30 speed Forward
wait 0.5 seconds
Servo S1 angle 100
wait 0.5 seconds
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 1 seconds
HuskyLens request data once and save into the result

define lineTracking
if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 1 then
    set motor All move by 60 speed Forward
else
    if read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 1 then
        set motor Left move by 200 speed Forward
        set motor Right move by 0 speed Forward
    if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 0 then
        set motor Left move by 0 speed Forward
        set motor Right move by 200 speed Forward
```

Note: the overall program link is at the end of the article

Operating Effect

Maqueen Plus starts from point B (L3 and R3 cannot detect black line when starting), drives along the black line, installs the foam block at point A, turns around and drives back along the black line, unloads the foam block after reaching point B, and then turns around and continues moving along the black line to point A to carries the foam block. It will keep going back and forth between points A and B in this way.



Project Development

In the project, a green card (ID1) is used to indicate that there are goods at point A. Every time HUSKYLENS recognizes the color of ID1, it will go to point A to transport the goods. If all the goods are transported, how should we let Maqueen Plus stop?

For example: when HUSKYLENS recognizes the color of ID1 (green) and Maqueen Plus goes to point A to transport the goods. When the goods are shipped, replace the card with a red one, and when HUSKYLENS recognizes the color of ID2 (red), tell Maqueen Plus that the goods are all shipped, and it will stop working.

Can you follow the instructions to complete this program?

Project 5: Self Driving Truck

Nowadays, the ongoing coverage of the logistics transportation industries is becoming a trend of era development with the boosting of economy and technology. While, it comes with a problem faced by all cargo distributing centers like port and storehouse---how to optimize truck's loading efficiency?
The invention of AGV opened a new chapter in logistics transportation. Unmanned and intelligent ports and warehouses are established in cities. Compared with traditional docks and warehouses, it no longer depends on operators to control the equipment, since there are AGV car driving along the appointed route automatically to transport the goods to the designated place. This technology not only reduces the labor cost, but also improves positioning accuracy and reliability.
Automated Guided Vehicle (AGV) refers to an unmanned automatic vehicle with an electromagnetic or optical device that can drive along the marked path.

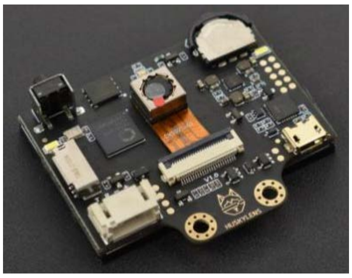
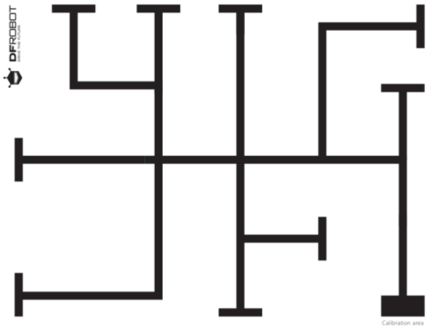
Can we turn Maqueen Plus into a small AGV truck? For instance, let it identify tags on the cargo, then transport cargo to the designated place according to the recognized result.

Function

This project will be mainly taking advantage of the HuskyLens Tag Recognition function. Each item has a unique tag. Maqueen Plus will select the corresponding path to drive along according to the ID number recognized by Huskylens sensor. For example, transport the cargo with tag ID1 to the endpoint of path 1, and so forth. Then all cargo can be placed into the designated position. The line-tracking sensors on Maqueen Plus will be used here.

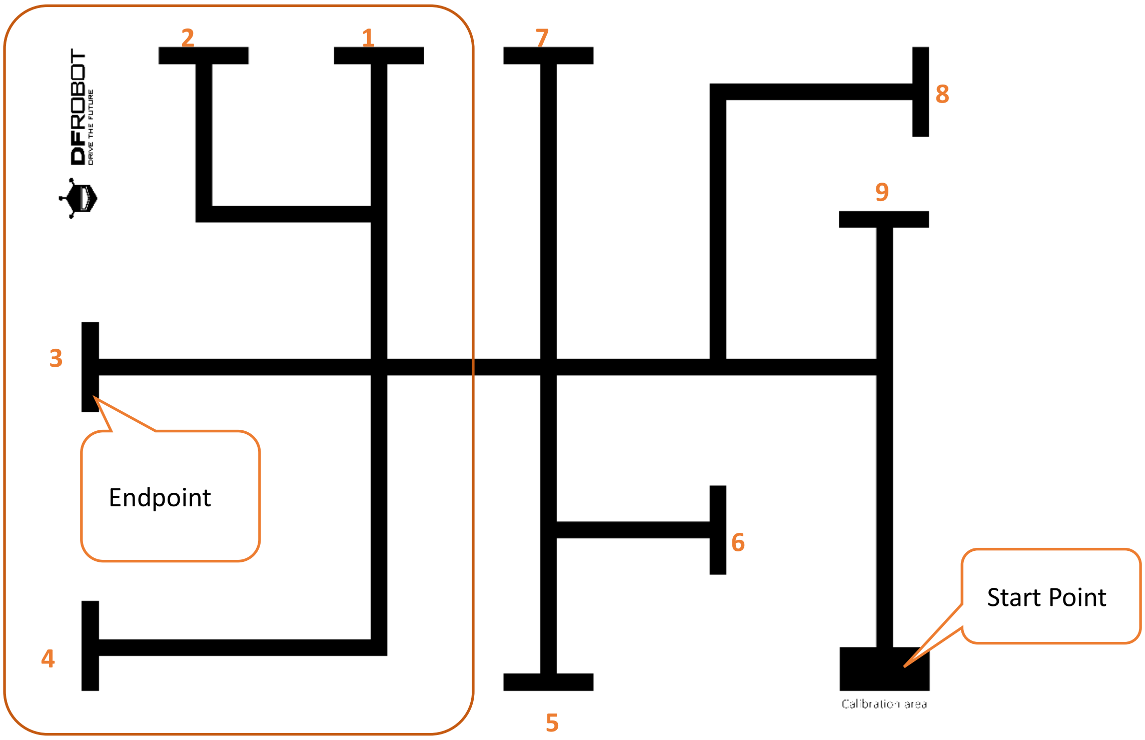
Bill of Materials

micro:bit ×1	Maqueen Plus ×1

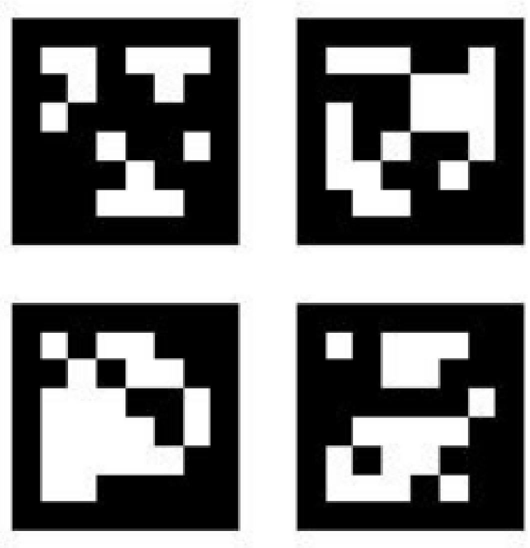
	
HUSKYLENS ×1	Object Tags×4
	
Line-tracking Map	

Hardware Connection

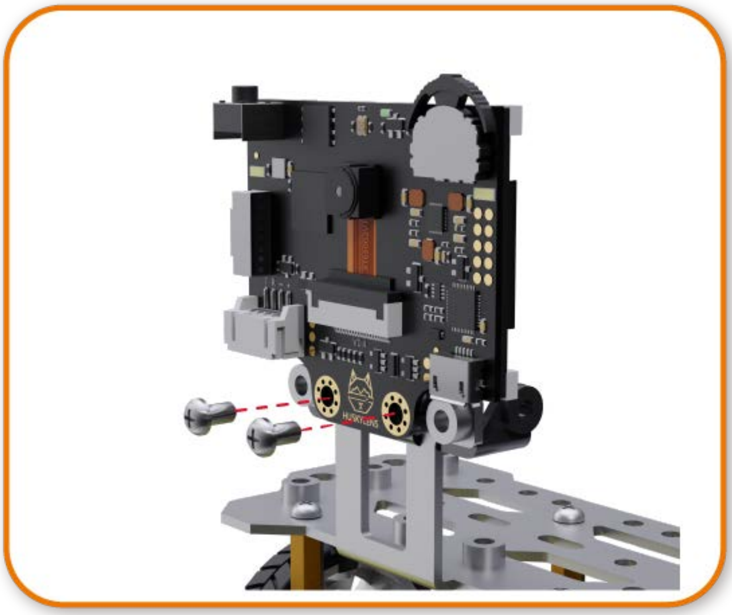
1. **Map:** prepare a map like the one below. The calibration area will be the starting point. There are 9 endpoints marked in the image below, which also means 9 paths for Maqueen Plus, but only paths 1 to 4 will be used in this project.



2. **Print Tags:** select four tags from the object tags above, and print them out.



3. **HUSKYLENS Assembly:** install Huskylens sensor onto the top plate of Maqueen Plus, as shown below:



Knowledge Field

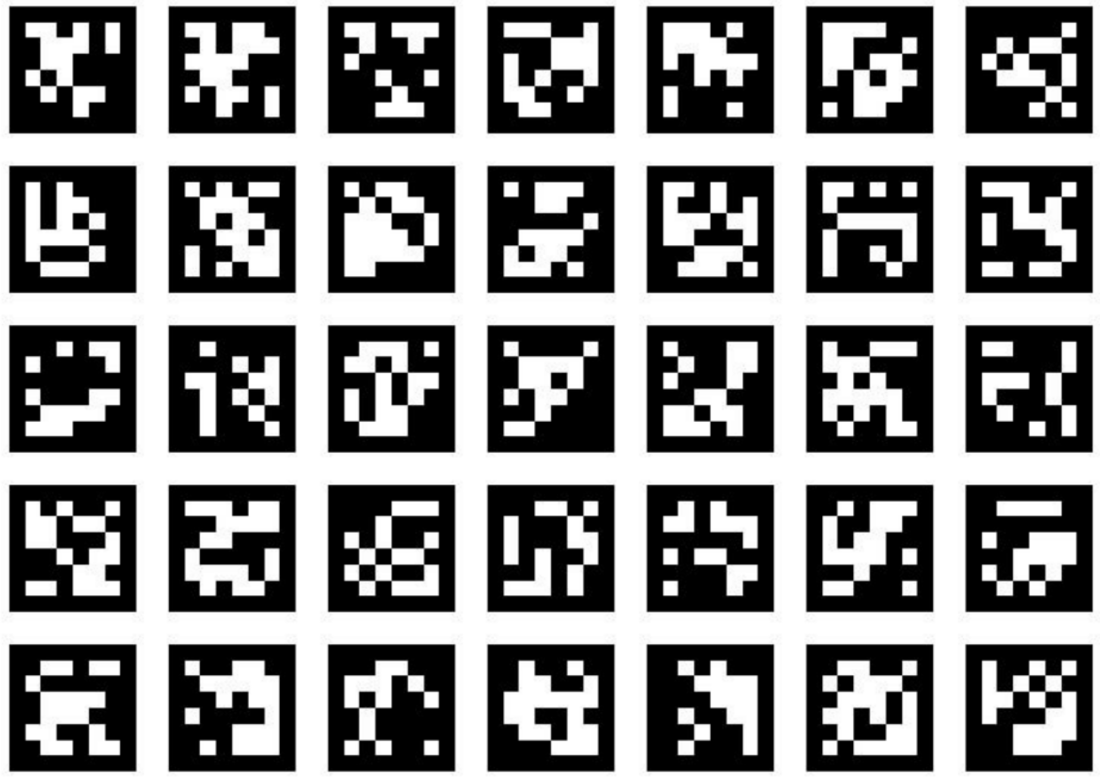
Tag recognition, a branch of machine vision, is widely used in all aspects of our production and daily life, such as product anti-counterfeiting, information acquisition.

1. What is tag recognition?

Tag recognition technology refers to a technical method for encoding and identifying goods in an effective and standardized way. There are many kinds of tags in our daily life, such as barcode, and QR codes.

AprilTag visual fiducial library is adopted in HuskyLens. AprilTag is a kind of vision positioning based on Quick Response Code road sign developed in recent years, it accurate three-dimensional position, direction and the tag ID of the two-dimensional code label relative to camera can be calculated.

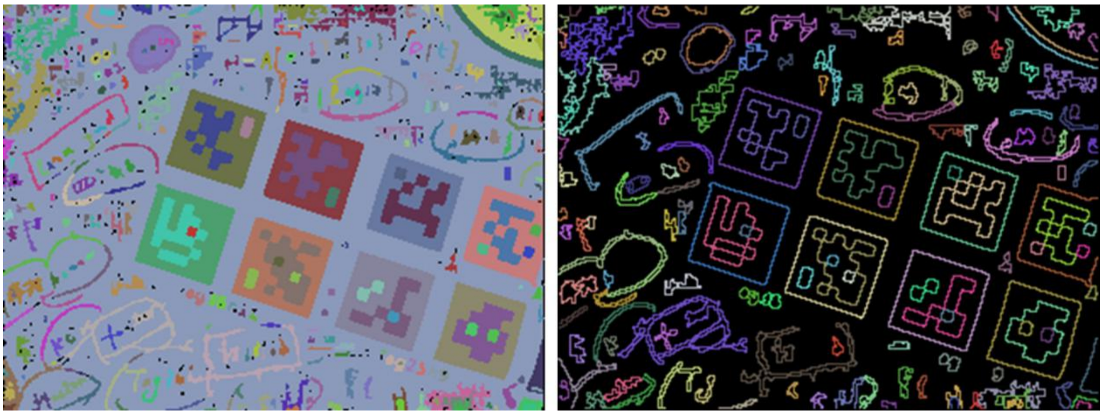
Huskylens only supports for the built-in AprilTag visual fiducial library, as shown below:



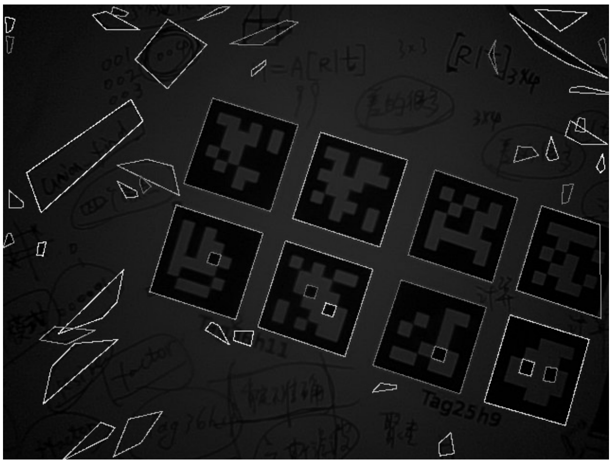
2. Tag Recognition Principle

AprilTag algorithm mainly includes the following steps:

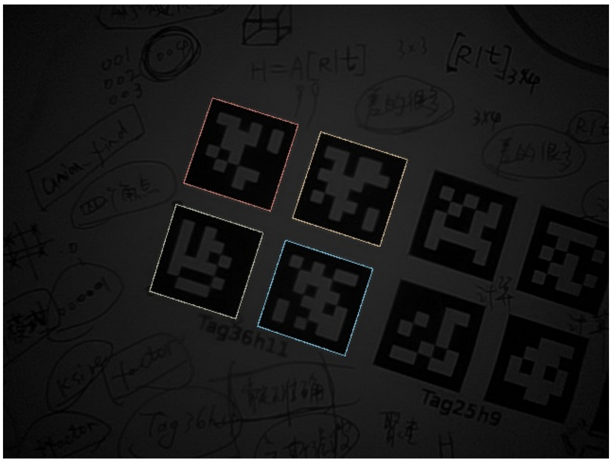
① Edge detection, find out the edge contour in the image.



② **Quadrangle detection:** find out the quadrangles in the contour.



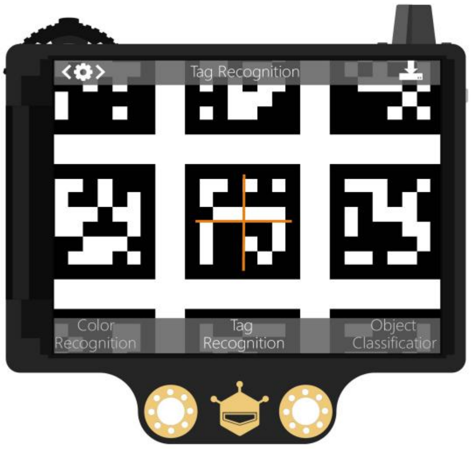
③ **Decoding:** match and check the quadrangles.



3. Demonstration of HuskyLens Tag Recognition Function

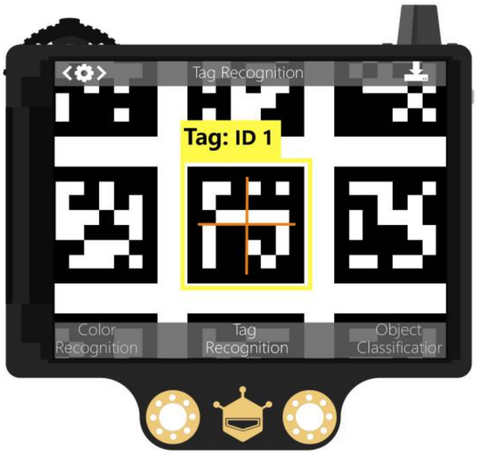
① *Tag Detection*

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 *Tag Learning*

Point the "+" symbol at the tag, long or short press the "Learning button" to complete the first tag learning.



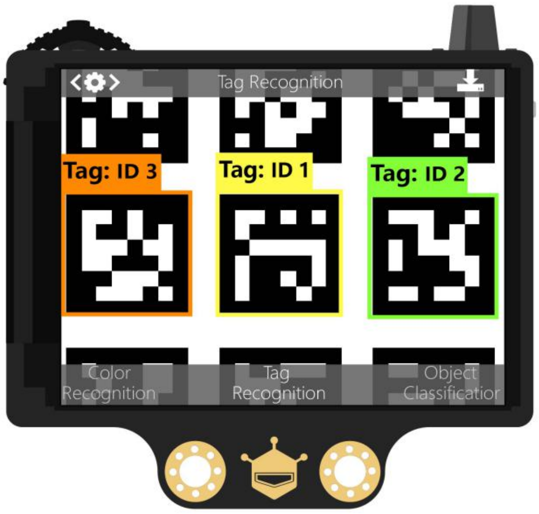
After releasing the"Learning button", the screen will display: "Press the button again to continue! Press other buttons to end." Please short press the "Learning button" within the countdown if you want to learn the next tag, if not, short press the "Function button" before the countdown ends, or do not press any button to wait for the ending of the countdown.

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 consistent with the sequence of the tag imported. This is to say, the ID will be marked as "tag: ID1", "tag: ID2", "tag: ID3", and so forth. And the frame colors for different tags are different.

③ *Tag Recognition*

When HuskyLens encounters the tags that have been learned, 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 codes.



Project Practice

How will the tag recognition of HuskyLens be used in this project? How does Maqueen Plus select a path to go? Let's break down the whole project into several small tasks and complete it step by step!

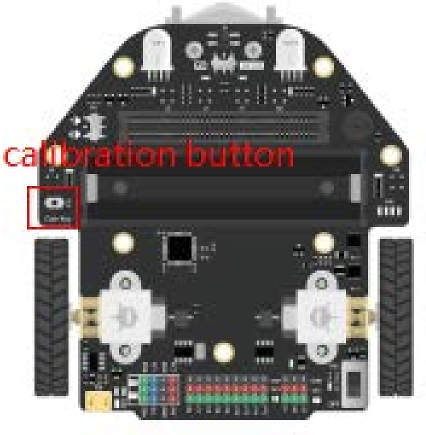
Task 1: Complete Basic Line-tracking Algorithms

Program Design

Step 1 Function Analysis

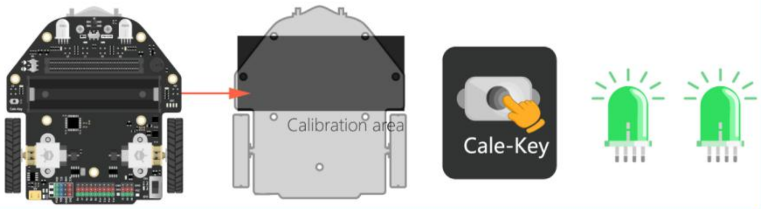
As an auto-driving truck, Maqueen Plus will identify the tag on the goods by using HuskyLens Tag recognition, select the corresponding path based on the recognized result, and then transfer the goods to the designated position.

The point-fixed transportation can be directly realized by the line-tracking sensors on Maqueen Plus itself. The 6 line-tracking sensors should be calibrated before use.



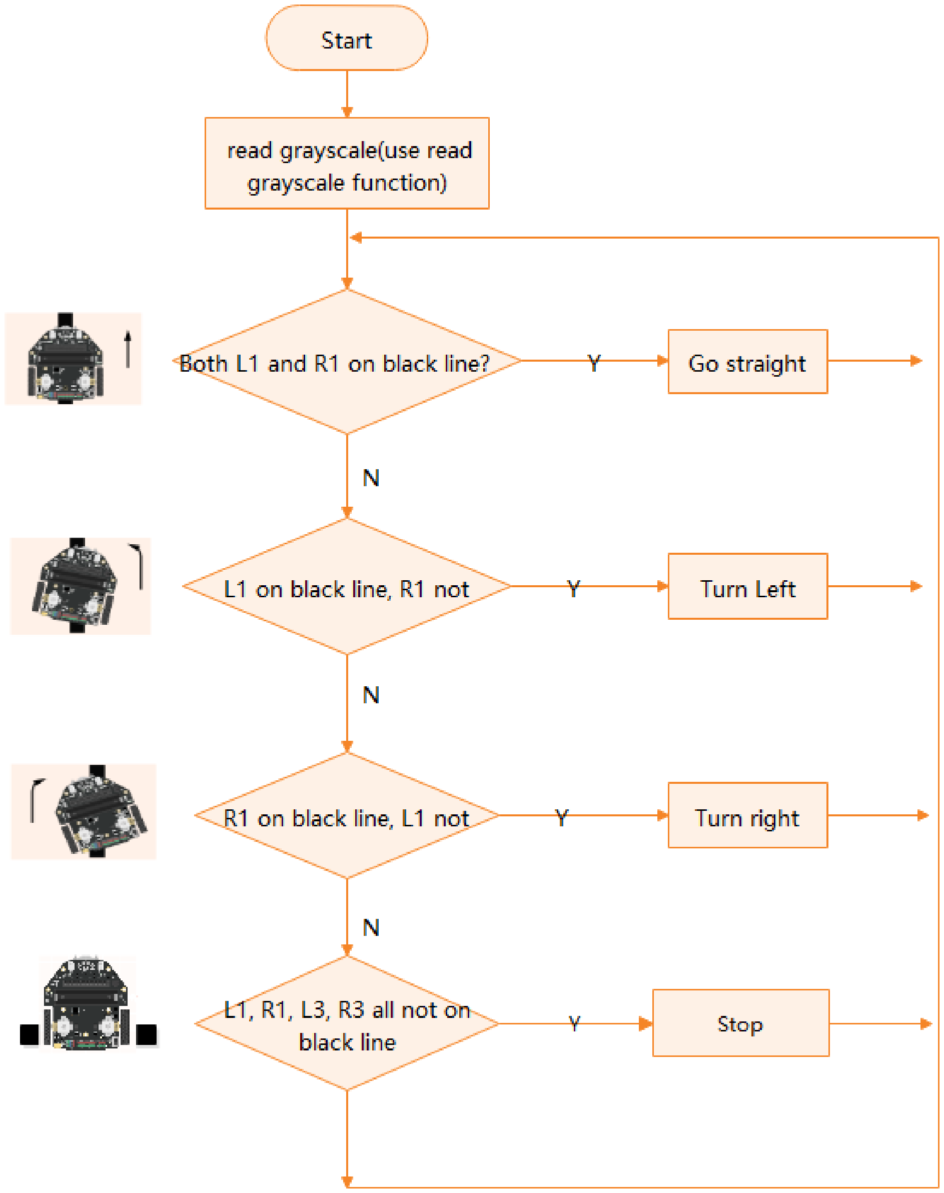
Put the Maqueen Plus on the black calibrating area of the map, and make sure all the line-tracking sensors are within that area. Press down the calibration button for 1 second, when the front two RGB LEDs flash green light, release the button, then calibration is done.

Note: the black line printed by the printer may not be detected accurately. It is recommended to use the map we provided or make a map with black tape.



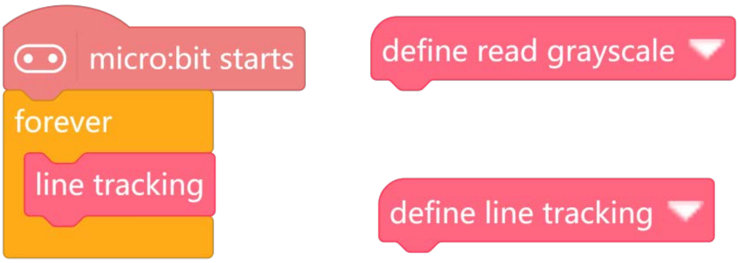
Step 2 Flowchart Analysis

First of all, complete the function of sensor reading and line-tracking:


Start
read grayscale(use read grayscale function)
Both L1 and R1 on black line?
Y
Go straight
N
L1 on black line, R1 not
Y
Turn Left
N
R1 on black line, L1 not
Y
Turn right
N
L1, R1, L3, R3 all not on black line
Y
Stop

Sample Program

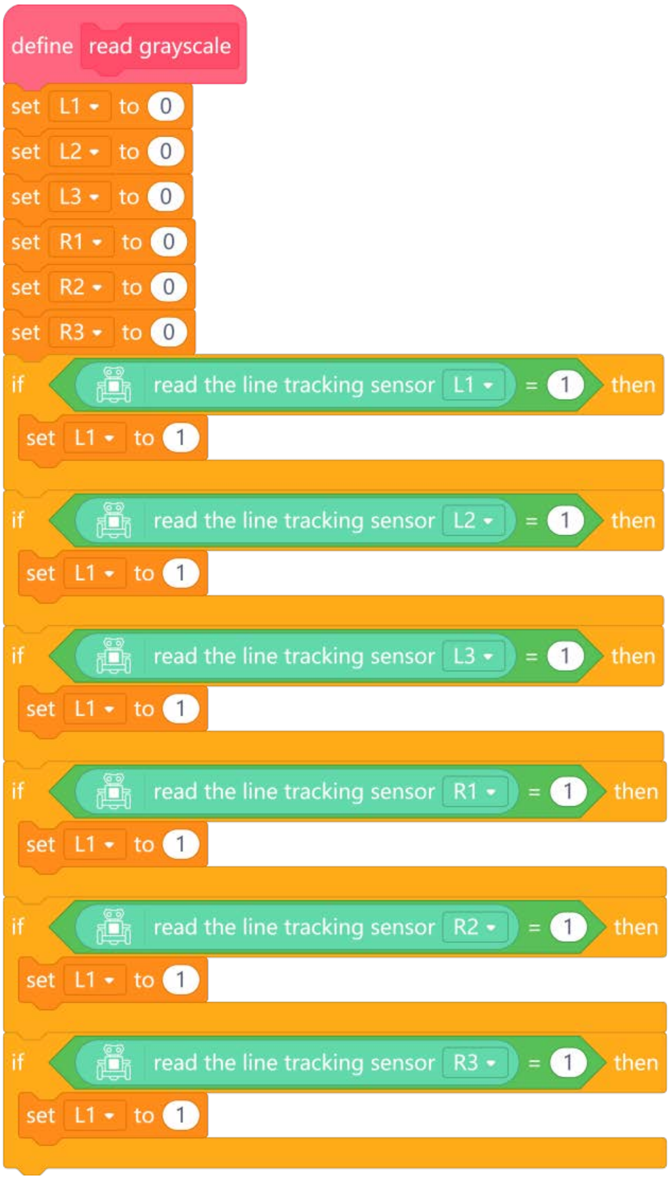
There are two functions encapsulated in the basic line-tracking algorithm: read grayscale and line tracking, as shown below:



```
micro:bit starts
forever
    line tracking

define read grayscale

define line tracking
```

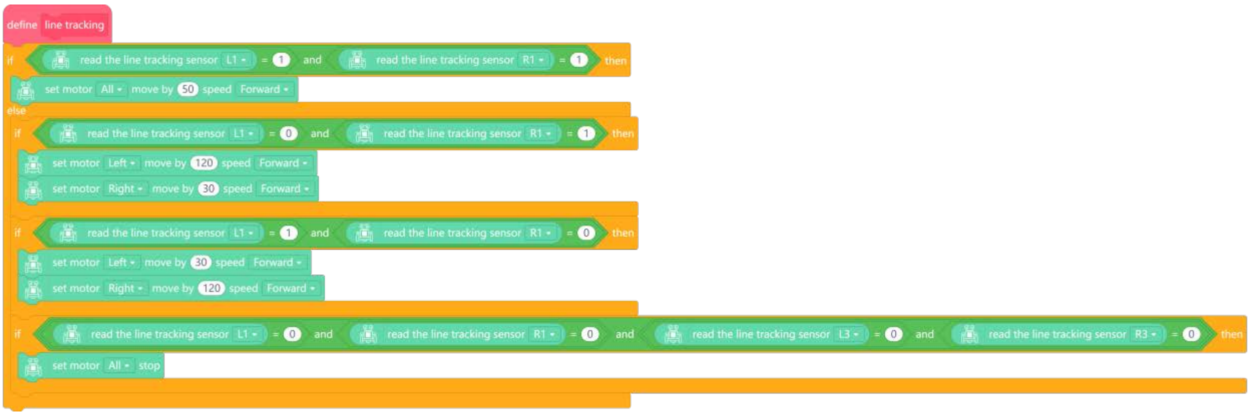
Note: the function block is hidden due to the editing limit. The detailed functions are as follows.

Read grayscale function:



```
define read grayscale
set L1 to 0
set L2 to 0
set L3 to 0
set R1 to 0
set R2 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 L2 = 1 then
    set L1 to 1
if read the line tracking sensor L3 = 1 then
    set L1 to 1
if read the line tracking sensor R1 = 1 then
    set L1 to 1
if read the line tracking sensor R2 = 1 then
    set L1 to 1
if read the line tracking sensor R3 = 1 then
    set L1 to 1
```

Line tracking function:

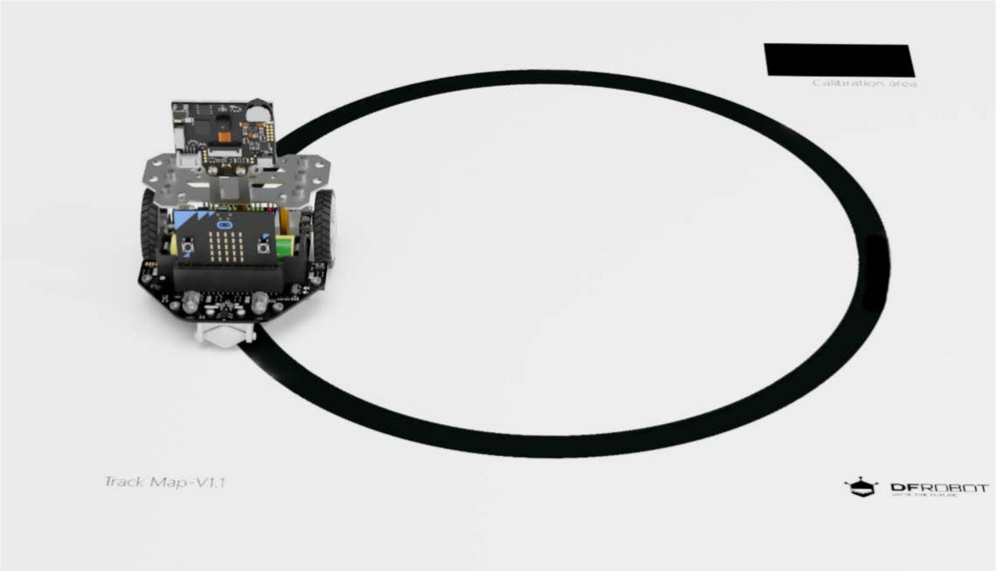


```
define line tracking
if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 1 then
    set motor All move by 50 speed Forward
else
    if read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 1 then
        set motor Left move by 120 speed Forward
        set motor Right move by 30 speed Forward
    if read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 0 then
        set motor Left move by 30 speed Forward
        set motor Right move by 120 speed Forward
    if read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 0 and read the line tracking sensor L3 = 0 and read the line tracking sensor R3 = 0 then
        set motor All stop
```

Note: the whole program link is attached at the end of the article.

Effect Display

Turn on Maqueen Plus's power switch, put it on the map, then it will drive along the black line.

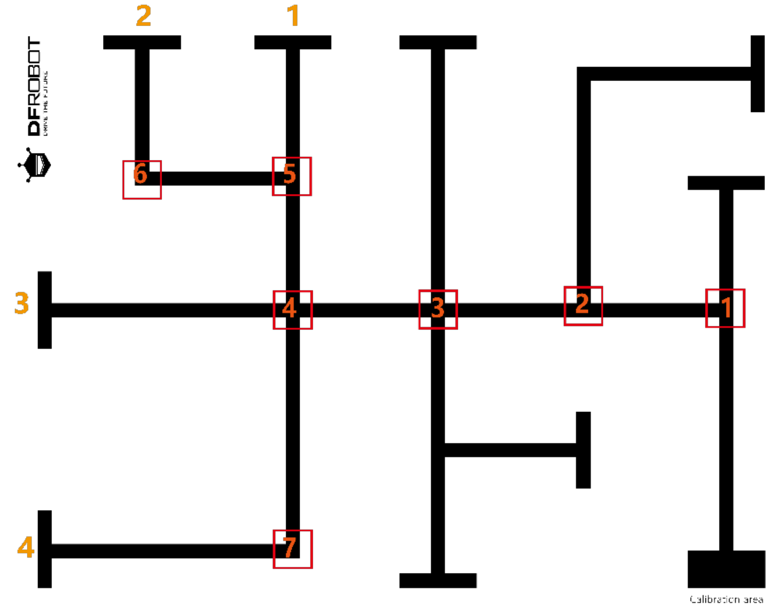


Task 2: Determine Direction at the Road Junction

Program Design

Step 1 Function Analysis

Four transportation paths(1~4) are selected for Maqueen plus, and there are junctions on each path. So at first, it is necessary to make sure which direction Maqueen Plus needs to go at the road junction in order to arrive at the designated position.



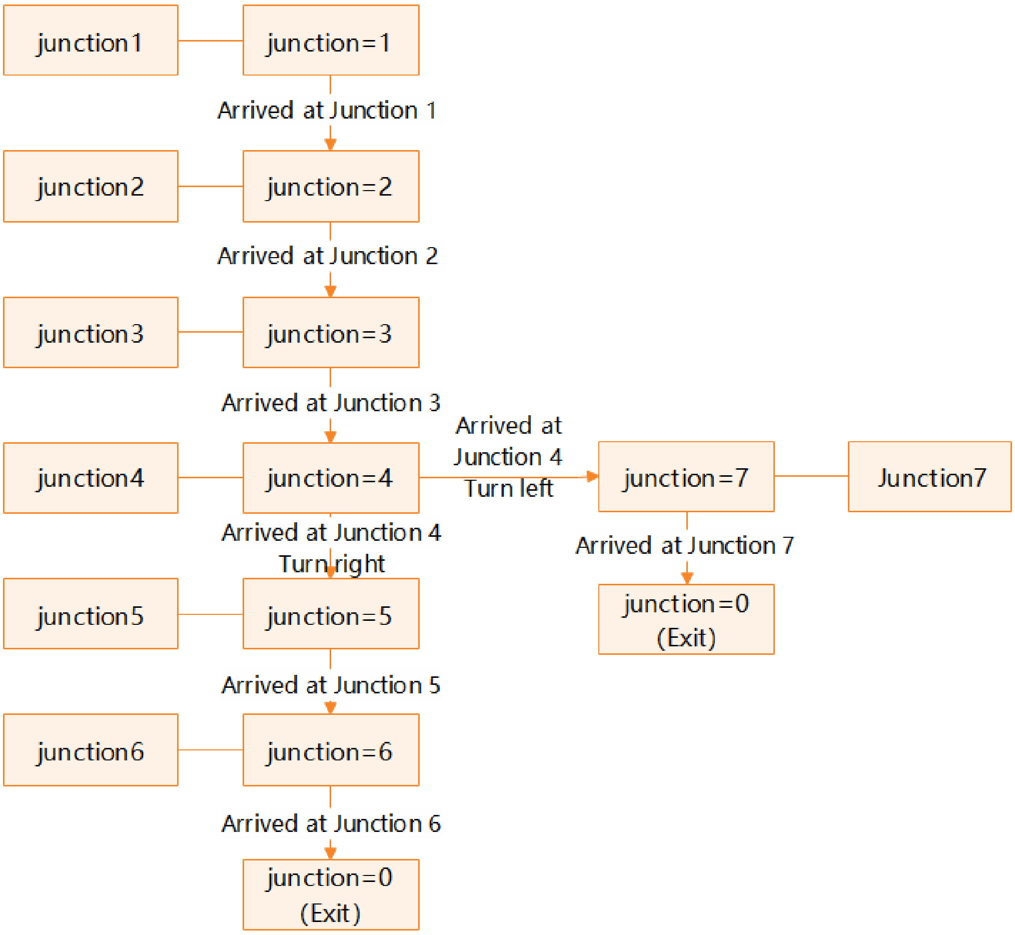
Step 2 Determine Direction

Junction Recognized	Direction
1	Turn Left
2	Go Straight
3	Go Straight
4	For path 1 or 2, turn right For path 4, turn left
5	Turn left (For path 1, ignore the program of junction 5)
6	Turn right
7	Turn right

Note: The junction programs above are only for paths 1 to 4. If other paths are used, the programs need to be revised accordingly.

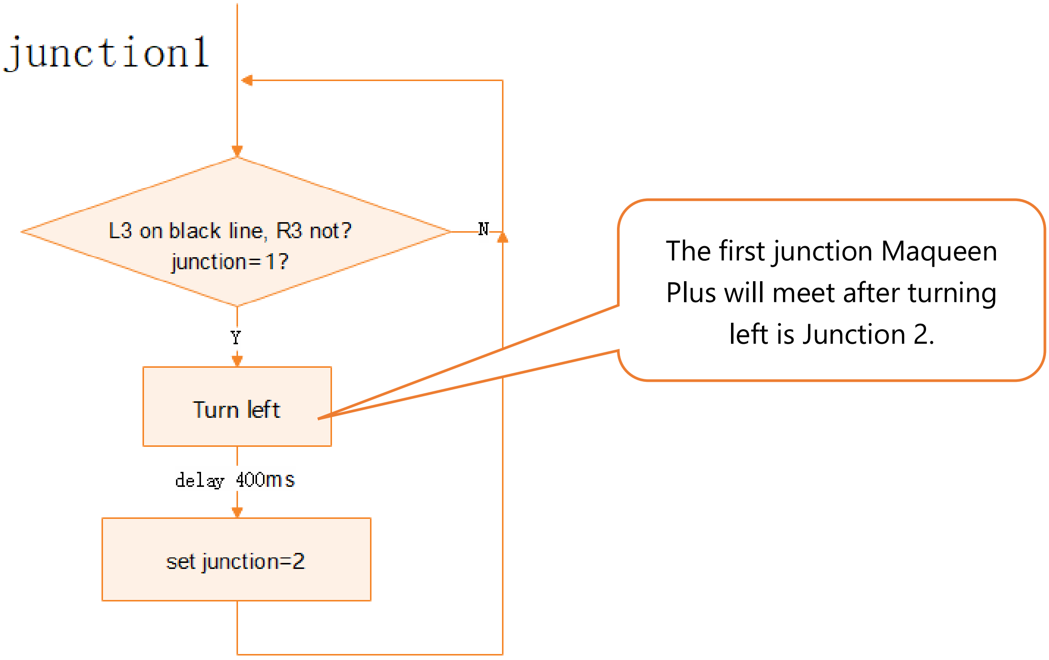
Step 3 Junction Judgement Flowchart

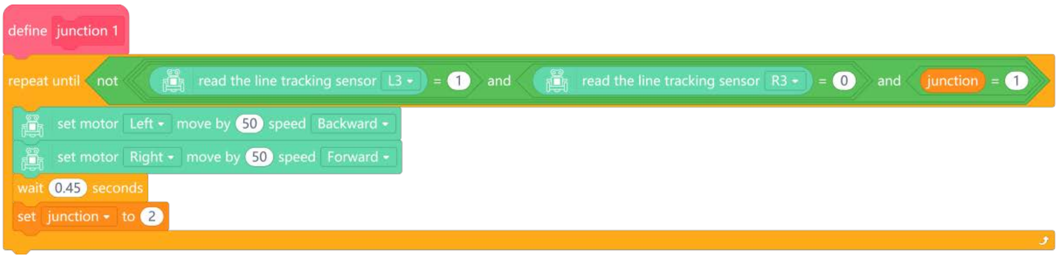
How do we know whether Maqueen Plus will pass a junction in the program? If two similar situations are detected, what should Maqueen Plus do? Seems complicated, so we must carefully design the program, otherwise, the expected results couldn’t be achieved. Here, a variable “Junction” needs to be created to record whether the Maqueen Plus has arrived at a junction. As shown below：



Sample Program

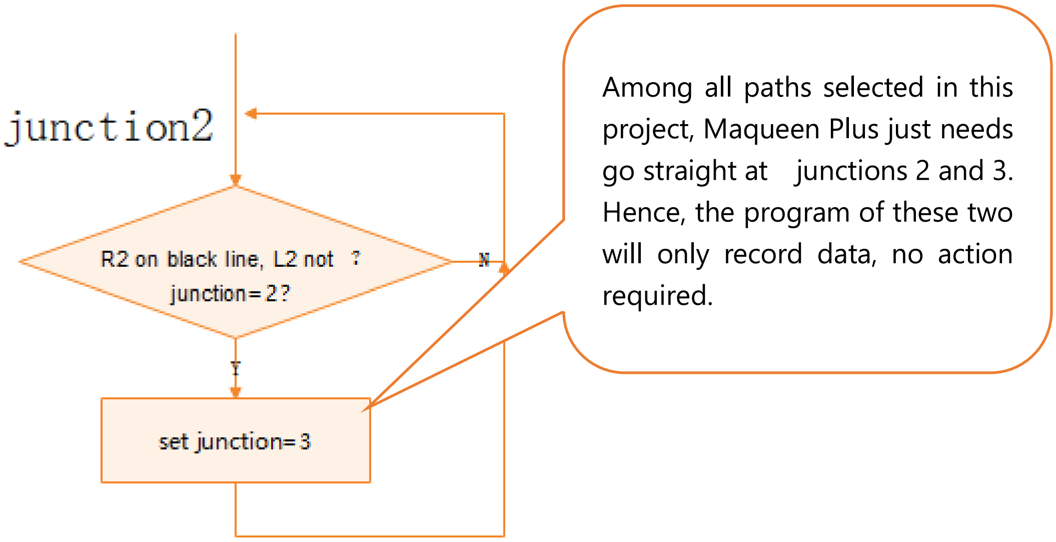
Junction 1


junction1
L3 on black line, R3 not?
junction=1?
N
Y
Turn left
delay 400ms
set junction=2
The first junction Maqueen Plus will meet after turning left is Junction 2.

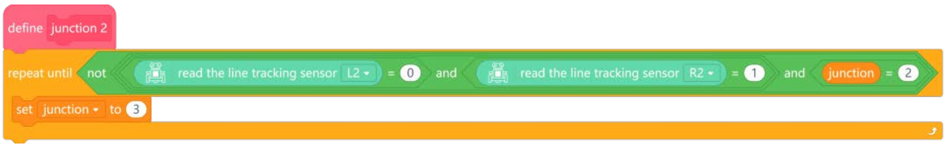


```
define junction 1
repeat until not read the line tracking sensor L3 = 1 and read the line tracking sensor R3 = 0 and junction = 1
    set motor Left move by 50 speed Backward
    set motor Right move by 50 speed Forward
    wait 0.45 seconds
    set junction to 2
```

Junction 2

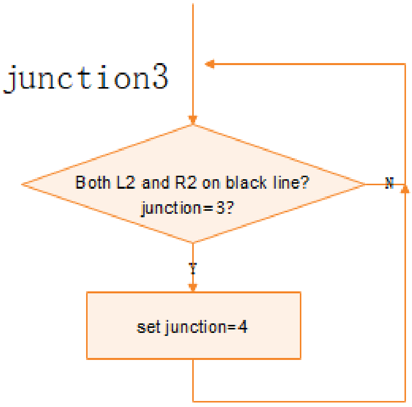


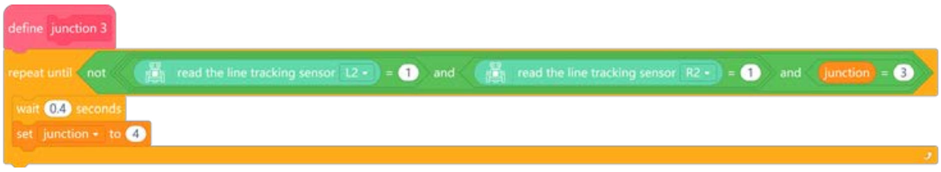
Among all paths selected in this project, Maqueen Plus just needs go straight at junctions 2 and 3. Hence, the program of these two will only record data, no action required.



```
define junction 2
repeat until not read the line tracking sensor L2 = 0 and read the line tracking sensor R2 = 1 and junction = 2
    set junction to 3
```

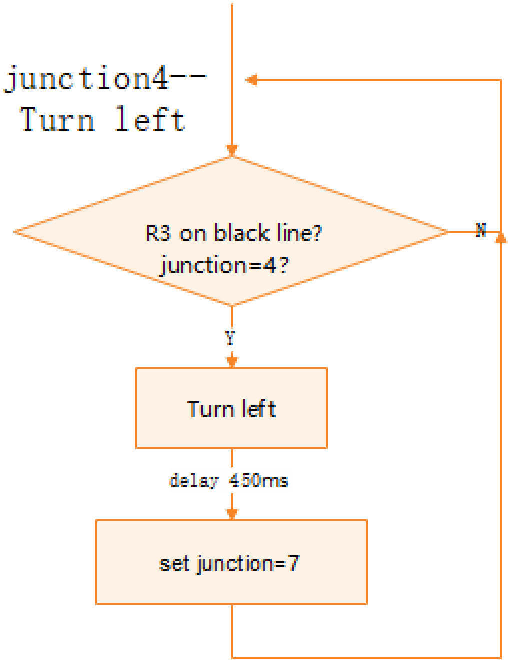
Junction 3

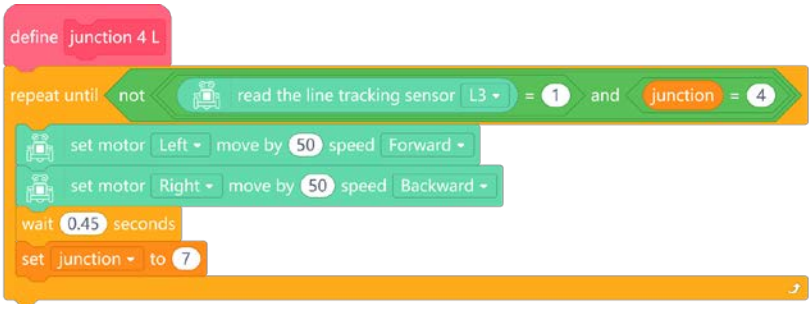




```
define junction 3
repeat until not read the line tracking sensor L2 = 1 and read the line tracking sensor R2 = 1 and junction = 3
    wait 0.4 seconds
    set junction to 4
```

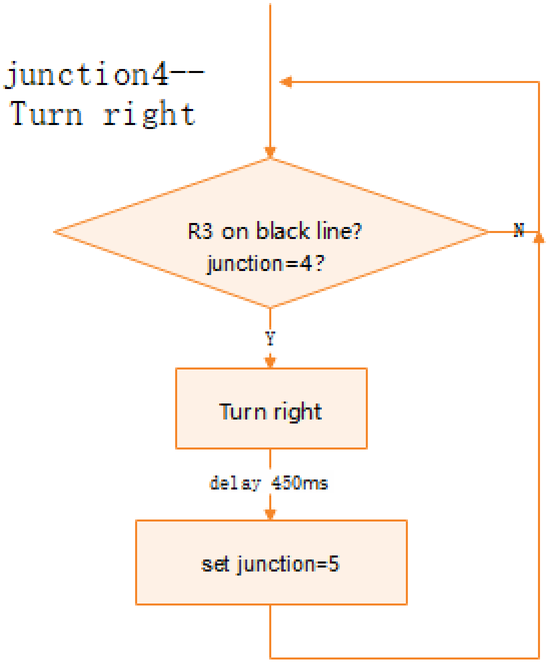
Junction 4--Turn Left

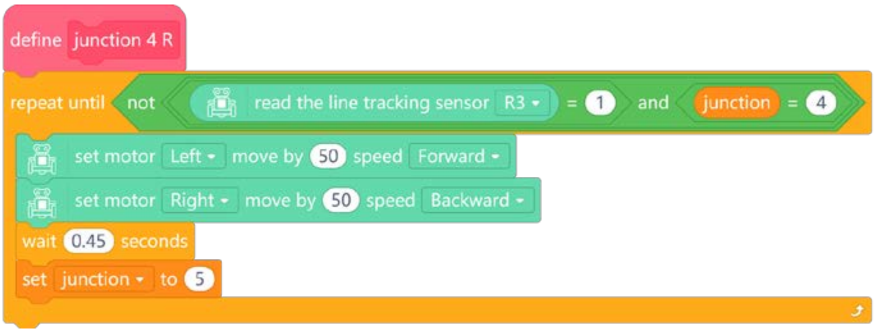




```
define junction 4 L
repeat until not read the line tracking sensor L3 = 1 and junction = 4
    set motor Left move by 50 speed Forward
    set motor Right move by 50 speed Backward
    wait 0.45 seconds
    set junction to 7
```

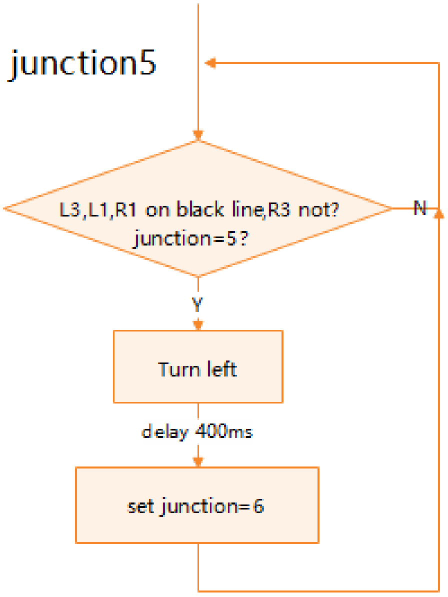
Junction 4--Turn Right

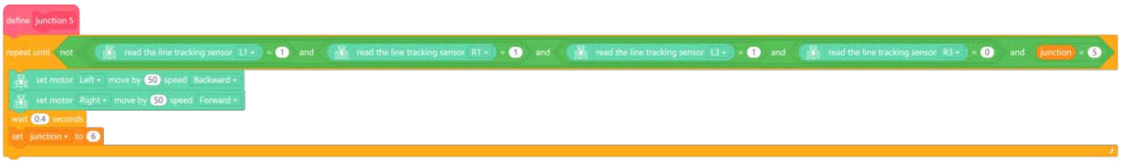




```
define junction 4 R
repeat until not read the line tracking sensor R3 = 1 and junction = 4
    set motor Left move by 50 speed Forward
    set motor Right move by 50 speed Backward
    wait 0.45 seconds
    set junction to 5
```

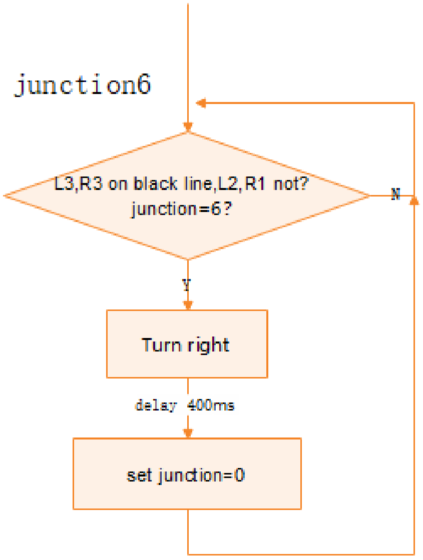
Junction 5

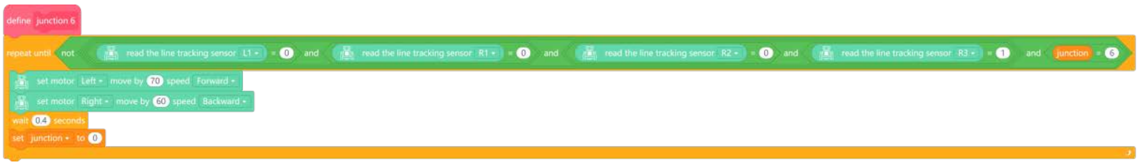




```
define junction 5
repeat until not read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 1 and read the line tracking sensor L3 = 1 and read the line tracking sensor R3 = 0 and junction = 5
    set motor Left move by 50 speed Backward
    set motor Right move by 50 speed Forward
    wait 0.4 seconds
    set junction to 6
```

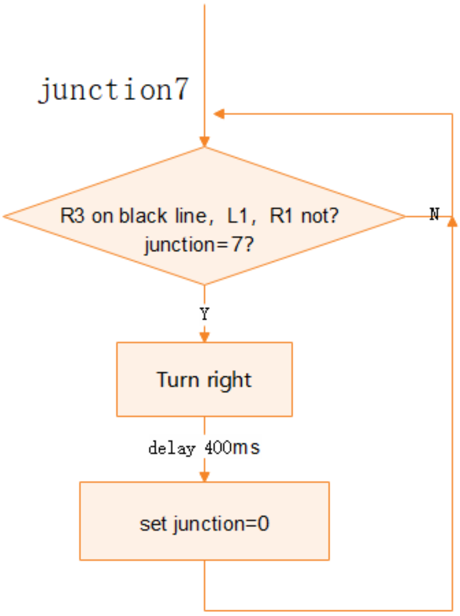
Junction 6

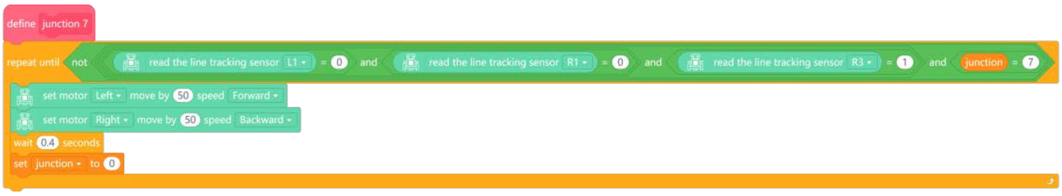




```
define junction 6
repeat until not read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 0 and read the line tracking sensor R2 = 0 and read the line tracking sensor R3 = 1 and junction = 6
    set motor Left move by 70 speed Forward
    set motor Right move by 60 speed Backward
    wait 0.4 seconds
    set junction to 0
```

Junction 7





```
define junction 7
repeat until not read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 0 and read the line tracking sensor R3 = 1 and junction = 7
    set motor Left move by 50 speed Forward
    set motor Right move by 50 speed Backward
    wait 0.4 seconds
    set junction to 0
```

Effect Display

When Maqueen Plus arrives at a junction, the corresponding program will be executed. More detailed effects will be discussed in Task 3.

Task 3: Complete the Junction Program

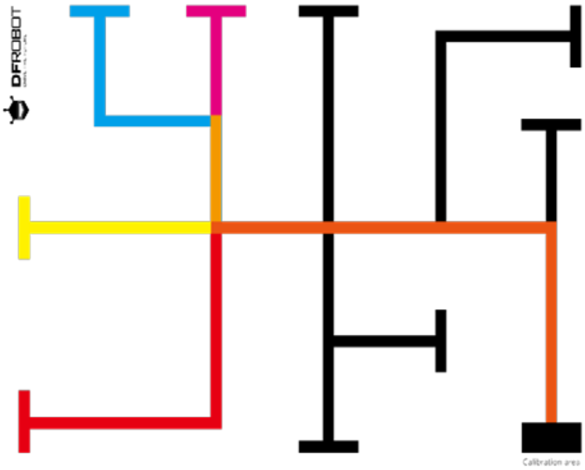
Program Design

Step 1 Route Planning

Four paths on the line-tracking map will be used in this tutorial, as shown below:

- 🟧 1/2/3/4 Shared part
- 🟨 1/2 shared part
- 🟪 Only for path 1
- 🟦 Only for path 2
- 🟨 Only for path 3
- 🟥 Only for path 4

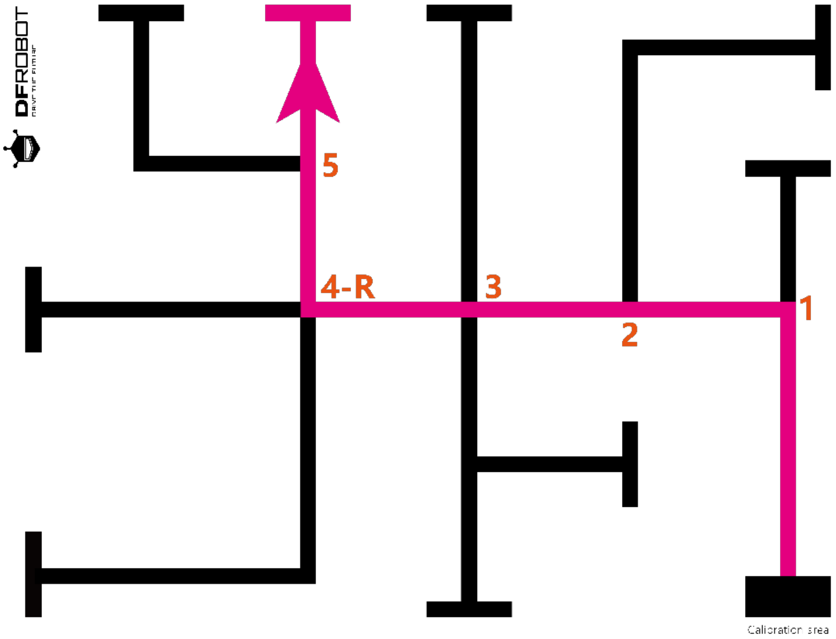
Note: the line color on the map is black, colors here are for distinguishing paths only.



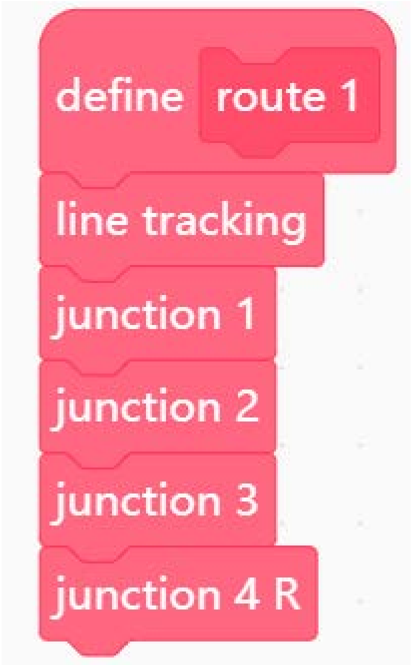
Step 2 Flowchart Analysis

For transferring goods on routes 1 to 4, which junctions will Maqueen Plus pass?

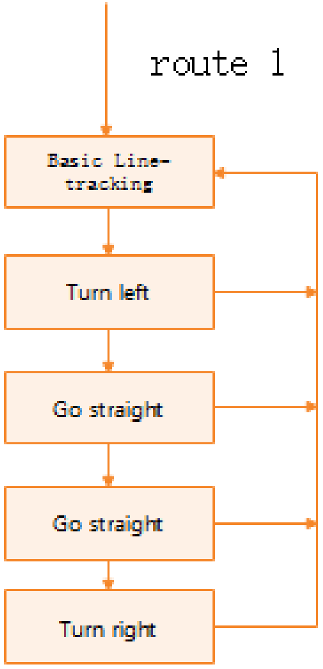
Path 1



From the image above we can see that path 1 includes junctions 1, 2, 3 4, and 5, so the functions required should be:

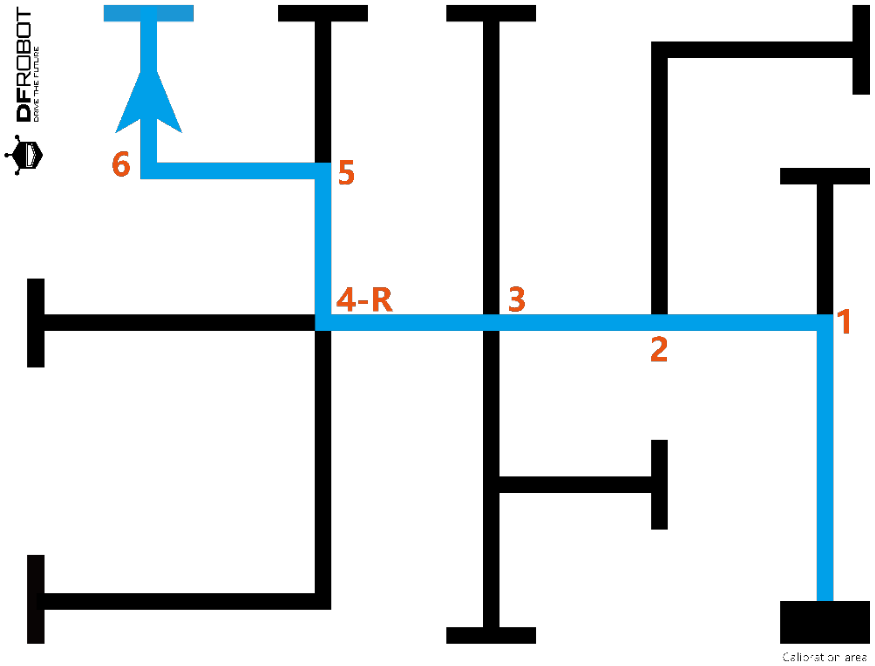


```
define route 1
line tracking
junction 1
junction 2
junction 3
junction 4 R
```

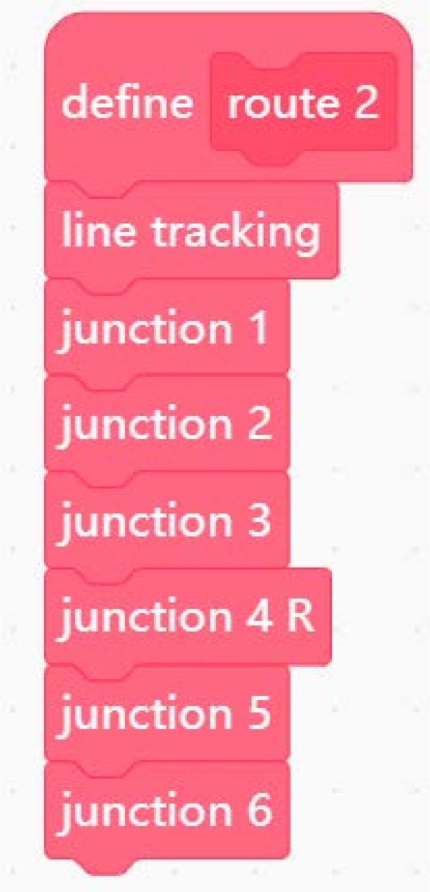


Note: when Maqueen Plus passed Junction 4, it can arrive at the endpoint of Path 1 using basic line-tracking function without calling the function of junction 5.

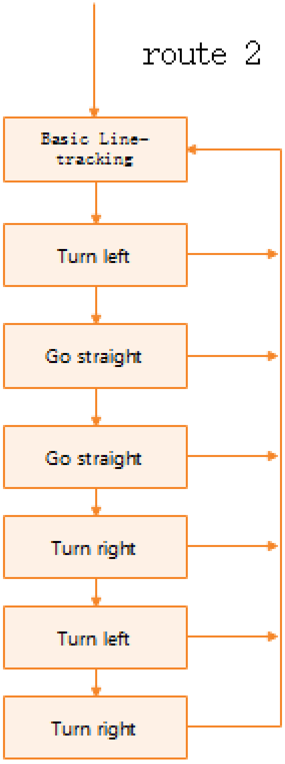
Path 2



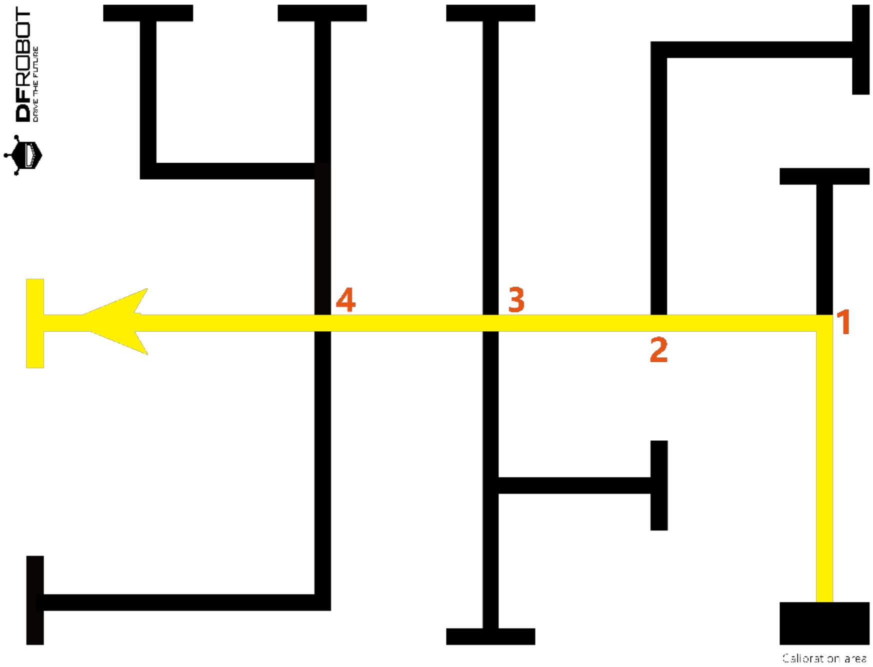
Maqueen Plus has to pass junctions 1, 2, 3, 4, and 5 on Path 2. As shown below:



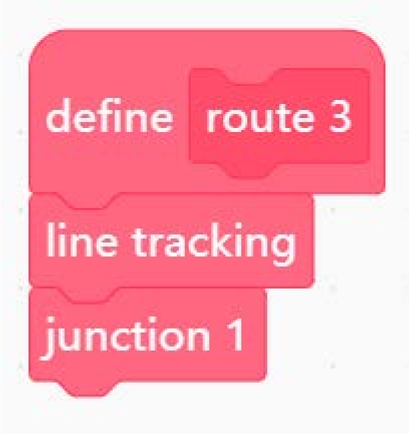
```
define route 2
line tracking
junction 1
junction 2
junction 3
junction 4 R
junction 5
junction 6
```



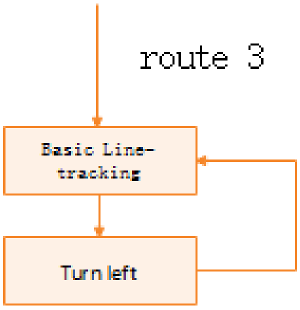
Path 3



Path 3 contains junctions 1, 2, 3 and 4.

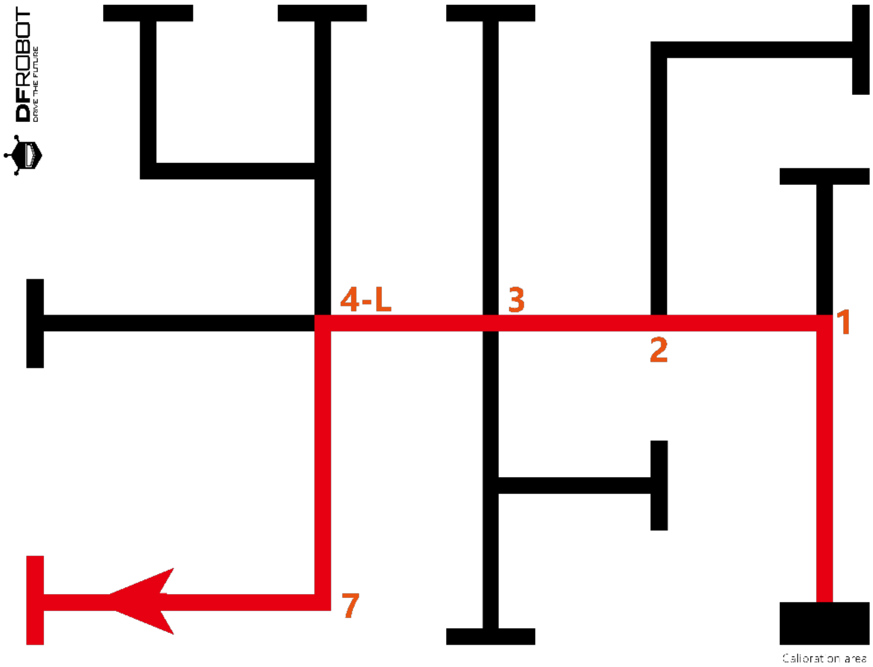


```
define route 3
line tracking
junction 1
```

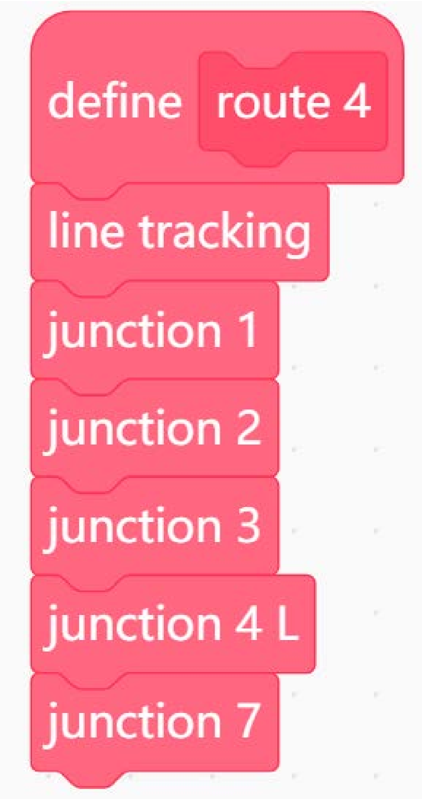


Note: on Path 3, after Maqueen Plus passed junction 1, it can directly arrive at the endpoint of Path 3 using the basic line-tracking. So only the function junction-1 needs to be called here.

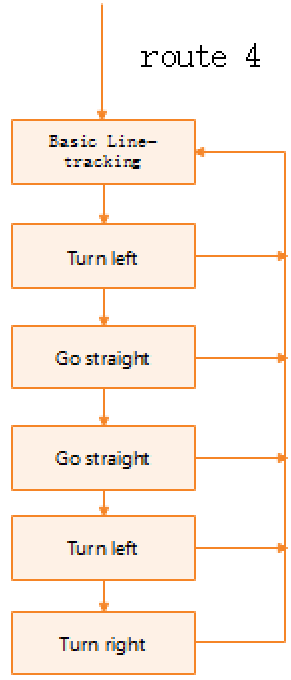
Path 4



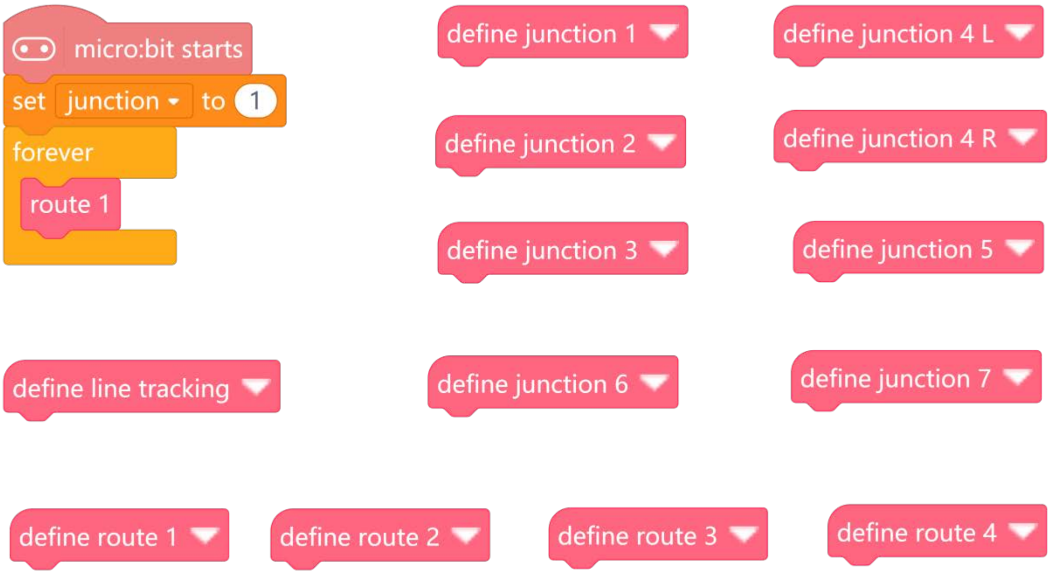
For Path 4, Maqueen Plus will pass the junctions 1, 2, 3, 4, and 7. Seen as below:



```
define route 4
line tracking
junction 1
junction 2
junction 3
junction 4 L
junction 7
```



Sample Program



```
micro:bit starts
set junction to 1
forever
    route 1

define junction 1
define junction 2
define junction 3
define junction 4 L
define junction 4 R
define junction 5
define line tracking
define junction 6
define junction 7
define route 1
define route 2
define route 3
define route 4
```

Note: the complete program is attached at the end of this article.

Effect Display

Independently call the function route 1, route 2, route 3, route 4 in forever. Maqueen Plus will drive along the route 1/2/3/4.

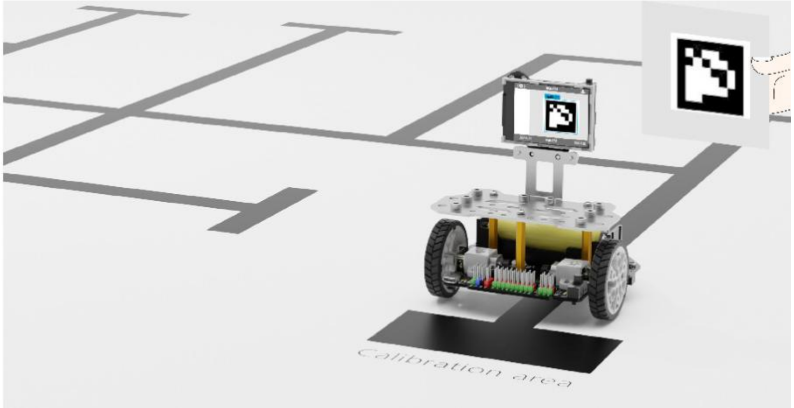
Task 4: Select Route According to the Recognized Tag

Program Design

Step 1 Function Analysis

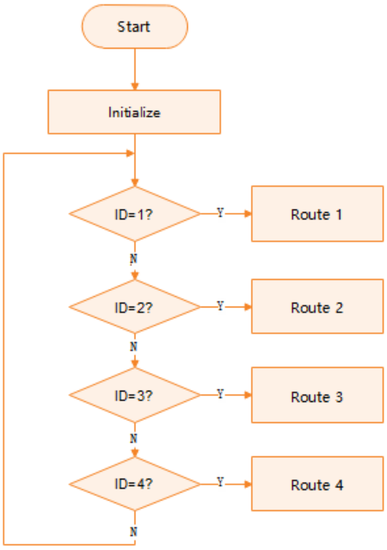
The basic fixed-point transportation can be realized after completing task 3. If we want Maqueen Plus to transfer 4 kinds of goods to 4 positions, how to design the program?

To make sure each kind of goods can be accurately placed at the designated position, we need to attach 4 different tags on the four goods. Maqueen Plus will select the corresponding route according to the tag ID the HuskyLens recognized. For instance, if the tag ID1 is recognized, Maqueen Plus will drive along path 1, and so forth.



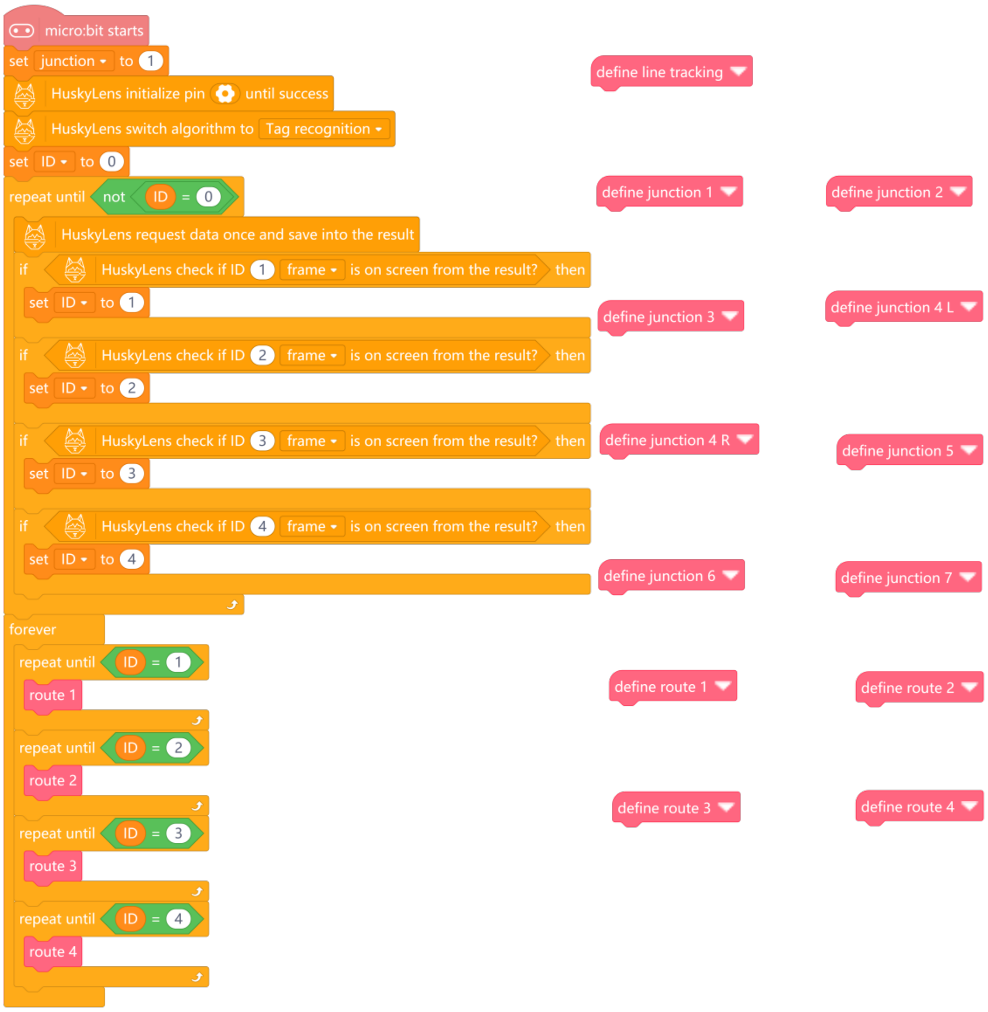
Step 2 Flowchart Analysis

When the HuskyLens sensor recognized the tag ID1, then Maqueen Plus will drive along path 1 to arrive at its endpoint, and so on.



Sample Program

Based on the program of Task3, add the tag recognition function, and then add the function “route 1”, “route 2”, “route 3” and “route 4”. Of course, you can select different routes for Maqueen Plus when programming. The codes below are only for route 1 to 4.



```
micro:bit starts
  set junction to 1
  HuskyLens initialize pin until success
  HuskyLens switch algorithm to Tag recognition
  set ID to 0
  repeat until not ID = 0
    HuskyLens request data once and save into the result
    if HuskyLens check if ID 1 frame is on screen from the result? then
      set ID to 1
    if HuskyLens check if ID 2 frame is on screen from the result? then
      set ID to 2
    if HuskyLens check if ID 3 frame is on screen from the result? then
      set ID to 3
    if HuskyLens check if ID 4 frame is on screen from the result? then
      set ID to 4

forever
  repeat until ID = 1
    route 1
  repeat until ID = 2
    route 2
  repeat until ID = 3
    route 3
  repeat until ID = 4
    route 4

define line tracking
define junction 1
define junction 2
define junction 3
define junction 4 L
define junction 4 R
define junction 5
define junction 6
define junction 7
define route 1
define route 2
define route 3
define route 4
```

Note: the complete program is attached at the end of this article.

Effect Display

Let the HuskyLens sensor learn these tags under the tag recognition mode. When the program starts, put Maqueen Plus in the starting position(calibration area). If the HuskyLens sensor recognized the Tag ID 1, Maqueen Plus will drive along route 1, and so forth.

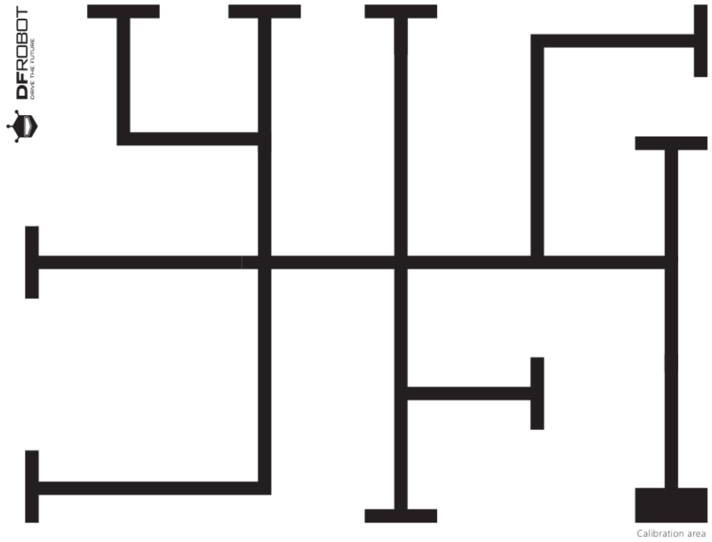
Project Development

This project presents the basic working principle of object recognition and the usage of HuskyLens tag recognition. Combining with the line-tracking function, Maqueen Plus can automatically select a route and transfer goods. But it would be more awesome if Maqueen Plus can complete the whole goods loading and unload by itself, right? Put on the mechanic accessories on Maqueen Plus to make a more intelligent self-driving truck!

Project 6: Out of the Maze

Line-following and maze algorithms are always a classic combination. How can we find the best way out of a complicated maze? We have become known about the basic line-tracking function of Maqueen Plus in the previous chapters, now can we make a project to let Maqueen Plus auto-get through a maze by the left-hand rule? Let’s have a try! The line-tracking map can be used as a maze.

Left-hand Rule: turn left first when arriving at a junction.

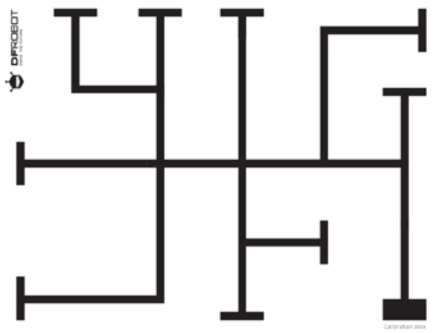


Function

In this project, Maqueen Plus will judge the road junction type using its line-tracking sensors. At the cross-junction, T-junction, and left L-junction, Maqueen Plus first turns left. The traveled routes will be memorized, and the best one will be calculated after all routes are recorded. At last, when driving from the starting point again, Maqueen Plus will drive along the best route to reach the endpoint.

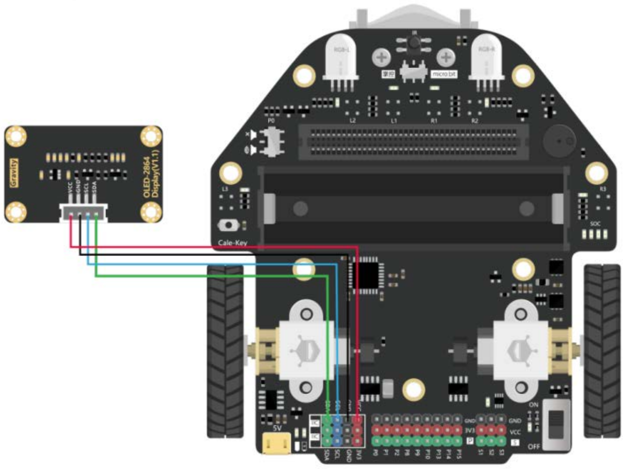
Bill of Material

micro:bit ×1	**Maqueen Plus ×1**

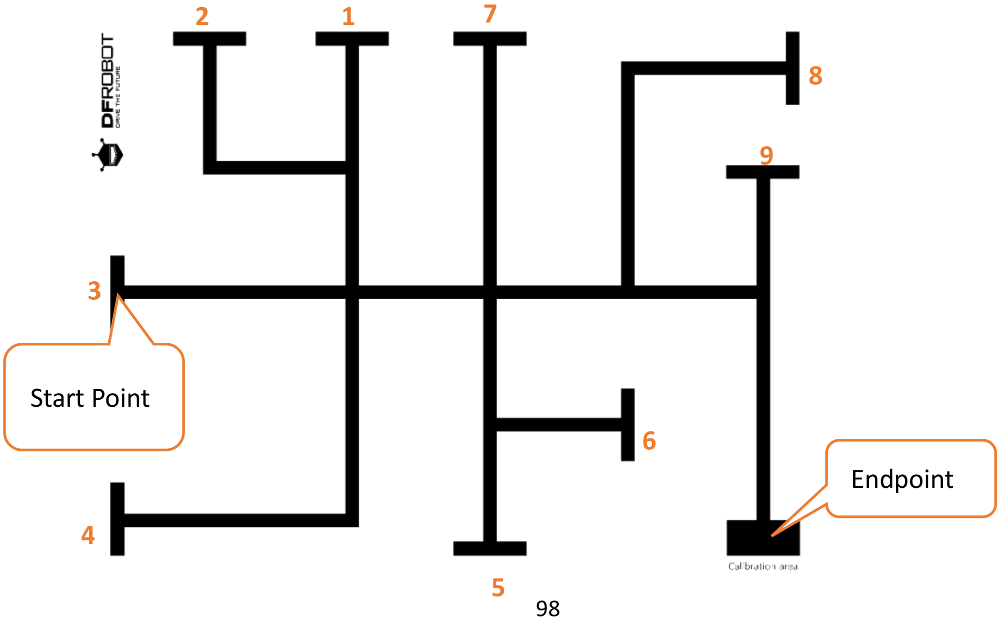
	
Line-tracking Map ×1	**OLED ×1**

Hardware Connection

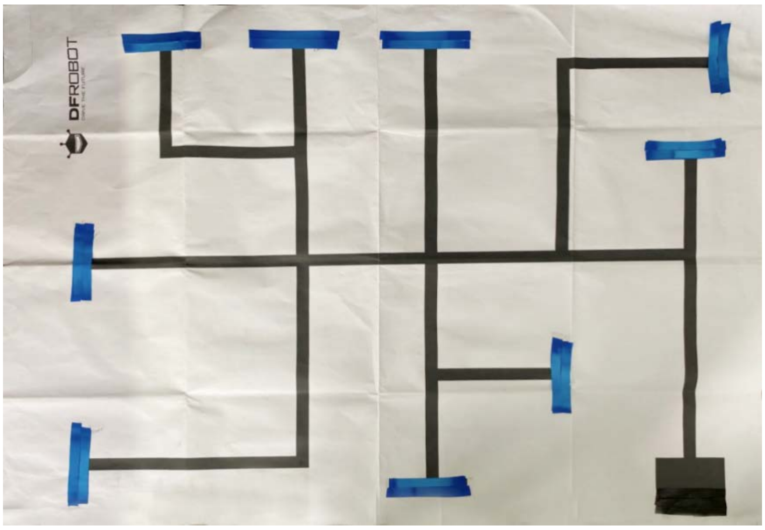
1. **Connection:** connect the OLED screen on the I2C interface.



2. **Map:** prepare a map like the one below. The calibration area will be the endpoint. The starting points are marked as follows. There are 9 starting points in total.



3. **Reform Map:** to complete this project, we still have to reform the map above. Hide the short lines of the start points with non-black tape (we use blue tape here). Extend the black area of the endpoint with black tape for improving error tolerance rate, as shown below:



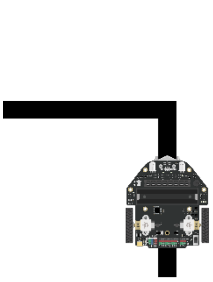
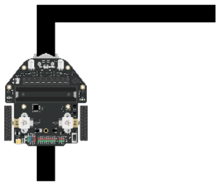
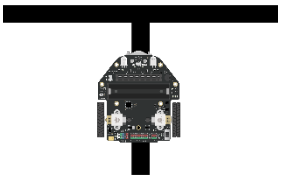
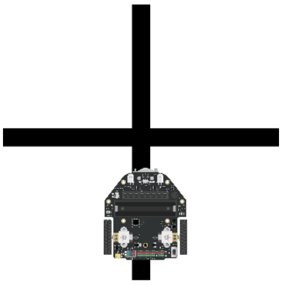
Note: it is recommended to use white tape to hide the lines, for other colors, you'd better apply two layers of tape on them. The intention of hiding the black short lines is to make sure that the detected value is 0 when Maqueen Plus is at the start point.

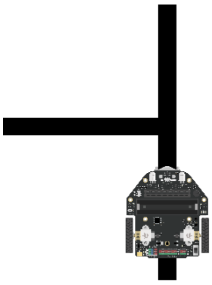
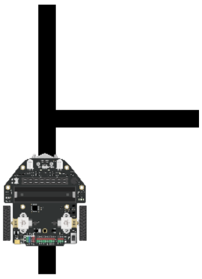
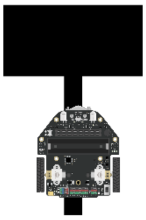
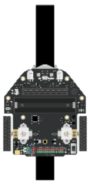
Knowledge Field

There could be many kinds of junctions in a maze, such as, straight, T junction, cross-junction, dead end. The most common way for traversing mazes is to use the "left-hand rule" or "right-hand rule" to handle all the routes in the maze, and then get the most optimized path, which also means the least time a line-tracking car will take from the beginning to end.

1. What is a Maze Map?

A maze map usually contains a complicated pattern, which is composed of black lines. And there are 8 types of junctions on a maze map: cross, T-junction, left L-junction, right L-junction, straight or go left, straight or go right, start/endpoint, dead end.

8 Types of Junctions in a Maze			
			
Left L-Junction	Right L-Junction	T-junction	Cross-junction

			
Straight or go left	Straight or go right	Endpoint	Dead end

2. Strategy for Out of the Maze

① Left-hand Rule

According to the left-hand rule, the priority level direction for the line-tracking car is: turn left->go straight->turn right. That is to say, in the maze, when arriving at a cross-junction or T-junction(can turn left, go straight or turn right), the line-tracking car always turns left; at the “straight or go right” junction, go straight. It only turns right at the right L-junction.

② Right-hand Rule

Opposite to the rule above, the right-hand rule follows the priority of right->straight->left, which means that the line-tracking car first turns right at a cross-junction or T-junction; at the “straight or go left” junction, go straight; It only turns left at the left L-junction.

The line-tracking car can get out of a maze in both ways. You can choose the rule you prefer.

Project Practice

The project will be mainly making use of the 6 line-tracking sensors on Maqueen Plus to recognize the road junctions, and then it will determine which way to go based on the recognized result. While, how do we make Maqueen Plus distinguish junction type, and select the corresponding program to execute after that? Let’s find the approaches to these problems step by step in the following Tasks!

Task 1: Basic Line-tracking of 4 Sensors

Program Design

Step 1 Function Analysis

Line-tracking is an essential part of a maze traversing project, and 4 line-tracking sensors will be used in the project for improving the error tolerance rate. Then, let’s get to know how to use 4 line-tracking sensors.

The line-tracking sensors need to be calibrated before use.

Step 2 Flowchart Analysis

The function of 4 line-tracking sensors:

Sensor Status				Motor Status
L2	L1	R1	R2	
0	1	1	0	Go Straight

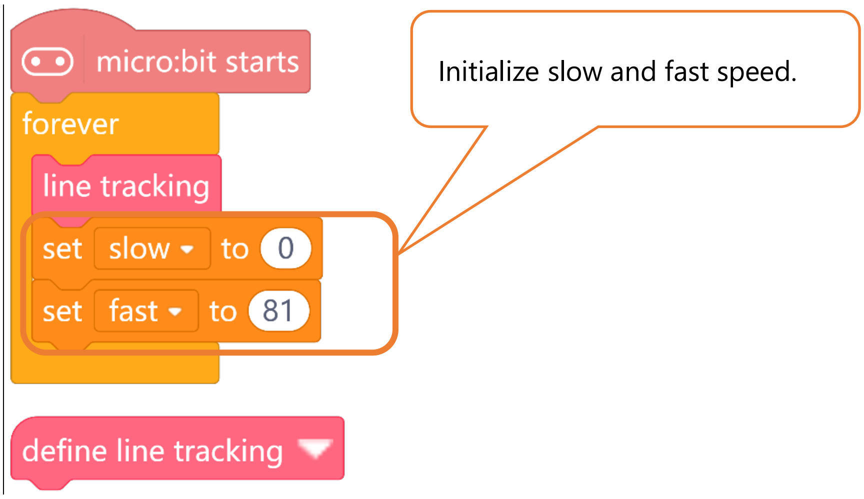
0	0	1	0	Small right turn
0	0	1	1	Medium right turn
0	0	0	1	Large right turn
0	1	0	0	Small left turn
1	1	0	0	Medium left turn
1	0	0	0	Large left turn

Note:

1) The output 0 means no black line is detected, 1 for black line detected.
2) Small, medium, large refer to the speed difference of the two wheels of Maqueen Plus. When the value is small, Maqueen Plus will make a small turn, and so forth.

Sample Program

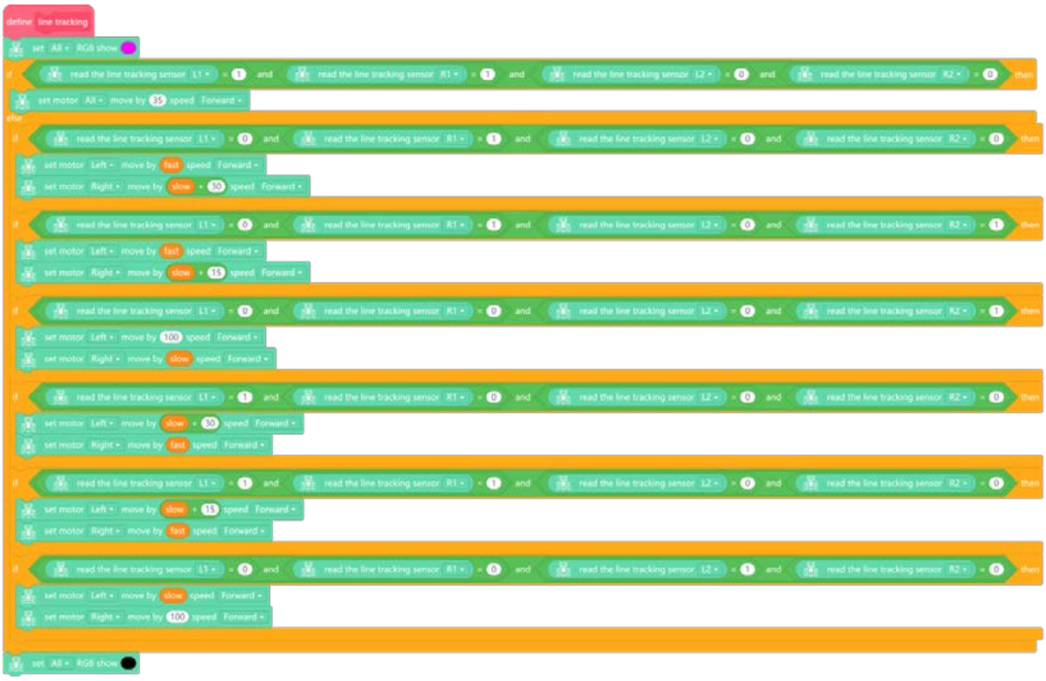
Whole sample program:



```
micro:bit starts
forever
  line tracking
set slow ▾ to 0
set fast ▾ to 81

define line tracking
```

Initialize slow and fast speed.

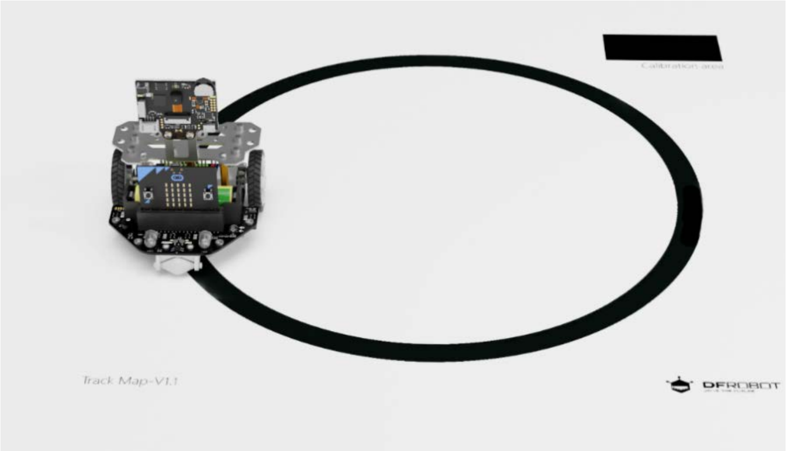
Line-tracking Function:



```
define line tracking
set All ▾ RGB show ●
if read the line tracking sensor L1 ▾ = 1 and read the line tracking sensor R1 ▾ = 1 and read the line tracking sensor L2 ▾ = 0 and read the line tracking sensor R2 ▾ = 0 then
  set motor All ▾ move by 35 speed Forward ▾
else
  if read the line tracking sensor L1 ▾ = 0 and read the line tracking sensor R1 ▾ = 1 and read the line tracking sensor L2 ▾ = 0 and read the line tracking sensor R2 ▾ = 0 then
    set motor Left ▾ move by fast speed Forward ▾
    set motor Right ▾ move by slow + 30 speed Forward ▾
  if read the line tracking sensor L1 ▾ = 0 and read the line tracking sensor R1 ▾ = 1 and read the line tracking sensor L2 ▾ = 0 and read the line tracking sensor R2 ▾ = 1 then
    set motor Left ▾ move by fast speed Forward ▾
    set motor Right ▾ move by slow + 15 speed Forward ▾
  if read the line tracking sensor L1 ▾ = 0 and read the line tracking sensor R1 ▾ = 0 and read the line tracking sensor L2 ▾ = 0 and read the line tracking sensor R2 ▾ = 1 then
    set motor Left ▾ move by 100 speed Forward ▾
    set motor Right ▾ move by slow speed Forward ▾
  if read the line tracking sensor L1 ▾ = 1 and read the line tracking sensor R1 ▾ = 0 and read the line tracking sensor L2 ▾ = 0 and read the line tracking sensor R2 ▾ = 0 then
    set motor Left ▾ move by slow + 30 speed Forward ▾
    set motor Right ▾ move by fast speed Forward ▾
  if read the line tracking sensor L1 ▾ = 1 and read the line tracking sensor R1 ▾ = 1 and read the line tracking sensor L2 ▾ = 0 and read the line tracking sensor R2 ▾ = 0 then
    set motor Left ▾ move by slow + 15 speed Forward ▾
    set motor Right ▾ move by fast speed Forward ▾
  if read the line tracking sensor L1 ▾ = 0 and read the line tracking sensor R1 ▾ = 0 and read the line tracking sensor L2 ▾ = 1 and read the line tracking sensor R2 ▾ = 0 then
    set motor Left ▾ move by slow speed Forward ▾
    set motor Right ▾ move by 100 speed Forward ▾
set All ▾ RGB show ●
```

Effect Display

Turn on Maqueen Plus's power switch, put it on the map. Maqueen Plus will drive along the black line, and the frequency of deviating from the black line at the corner gets smaller compared with only 2 line-tracking sensors used.



Task 2: Junction Analysis

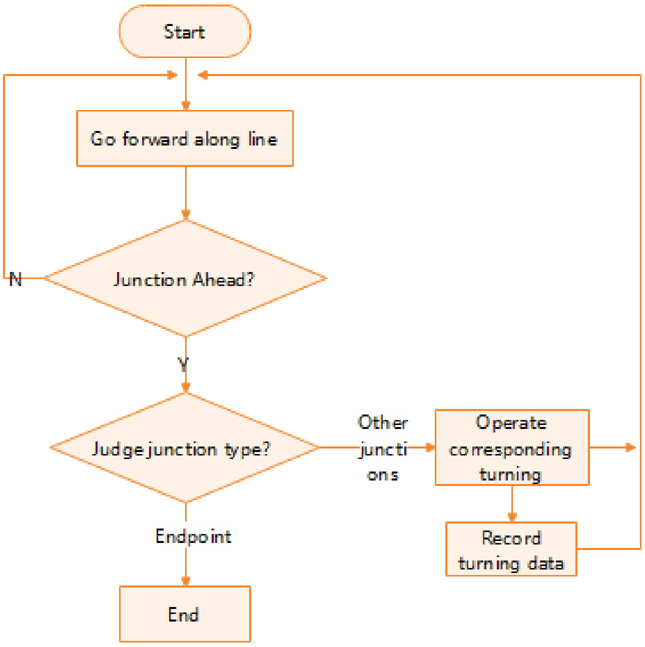
Program Design

Step 1 Process of Getting Through the Maze

There are 3 steps for Maqueen Plus to complete during the process of getting through the maze.

1. Move forward along the path and detect if there is a junction ahead.
2. Judge the type of junction when a junction is detected.
3. Select the corresponding turning mode after the last step is finished.

The steps above will be executed repeatedly until Maqueen Plus arrived at the endpoint.

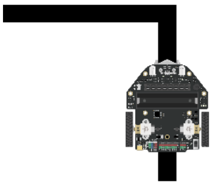
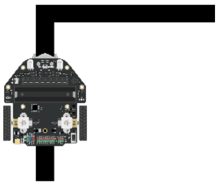
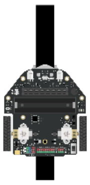


Step 2 Junction Recognition and Judgement

Maqueen Plus can drive along the path now after completing Task 1, but it cannot judge the junction or do correct action at each junction. Well, let's focus on these two points.

As we mentioned before that there are 8 types of junctions in the maze, and now we will analyze them in detail.

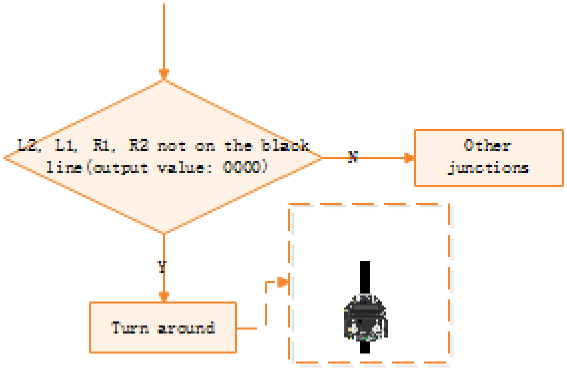
Normally, a junction is where two or more roads meet, which means there are over one direction a car can go. So theoretically the left L-junction, right L-junction and dead-end cannot be called a junction. At these positions, Maqueen Plus will only directly turn left/right/around without "judging" which direction to go.

Junctions with one direction		
		
Left L-junction	Right L-Junction	Dead End

1. Dead End

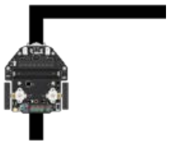
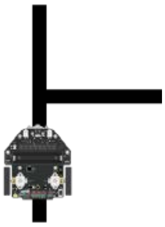
Dead end is the simplest junction to be recognized and handled. When the sensors L2, L1, R1, R2 all output "0000", Maqueen Plus arrived at a dead end.

Maqueen Plus has to turn around at the dead end (call function U-turn). As shown below:



2. Right L-junction and "Straight or go right" junction

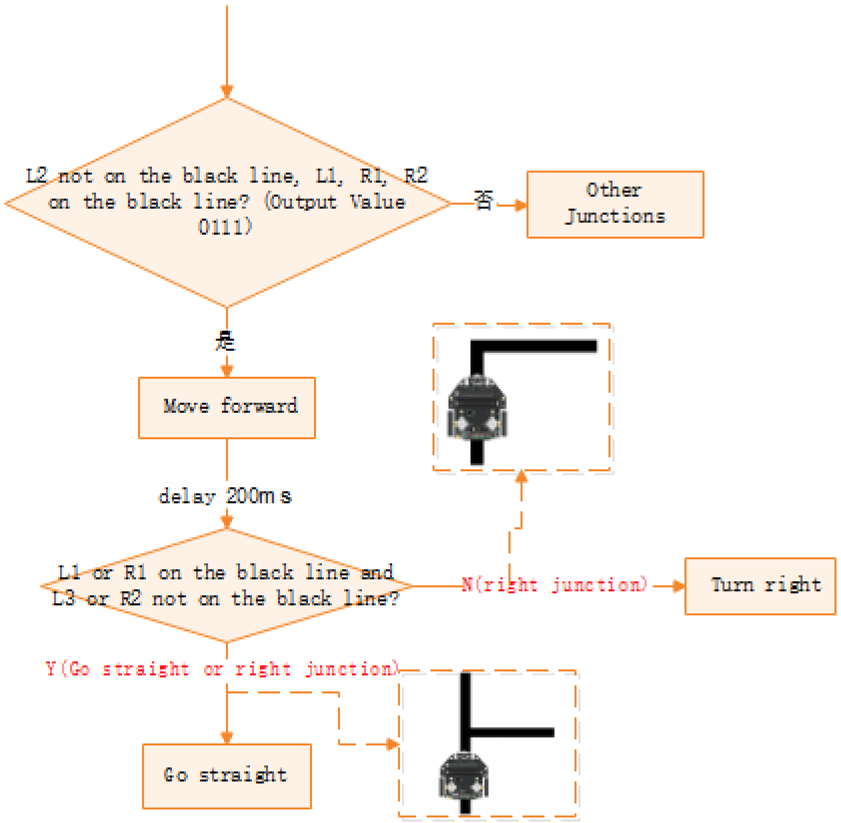
When encountering these two junctions, the values of line-tracking sensors of L2, L1, R1 and R2 will be "0111", but how does Maqueen Plus distinguish them?

	
Right L-Junction	Straight or go right

What we have to do here is to make Maqueen Plus move forward a little bit when the values of the sensors above are “0111”. After the four sensors crossed the junction, detect again, now if the sensor readings of L1 and R1 are both 1, then the junction must be “straight or go right”. If both are 0, it is a right L-junction.

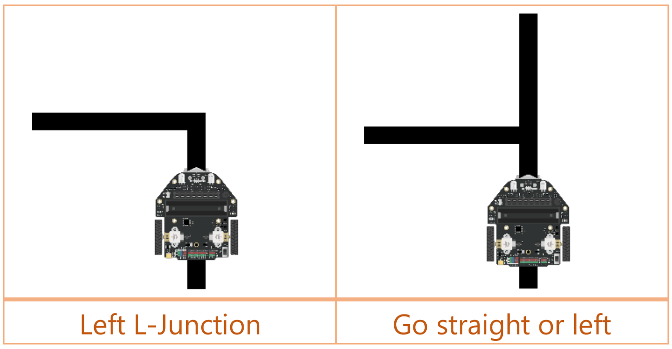
According to the “left-hand rule”, Maqueen Plus has to react as follows at these two junctions:

Turn right (call State when turning right function) at theGo straight or right” junction; Go straight (call State when going straight function) at the “Go straight or right” junction. As shown below:


L2 not on the black line, L1, R1, R2 on the black line? (Output Value 0111)
否
Other Junctions
是
Move forward
delay 200ms
L1 or R1 on the black line and L3 or R2 not on the black line?
N(right junction)
Turn right
Y(Go straight or right junction)
Go straight

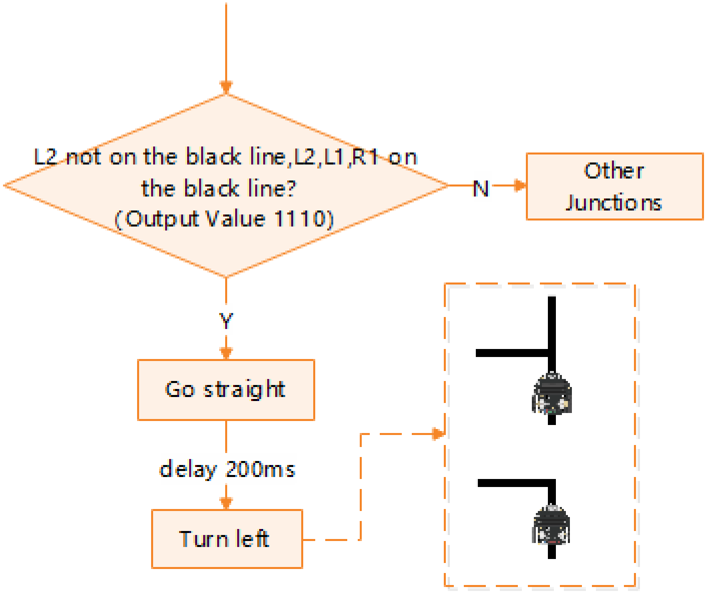
3. Left L-junction and “Go straight or left” junction

The sensors readings of L1, L2, R1, R2 will be “1110” when Maqueen Plus arrived at these two junctions.



Left L-Junction Go straight or left

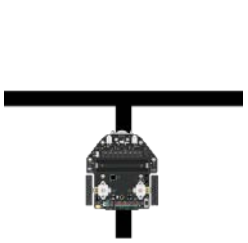
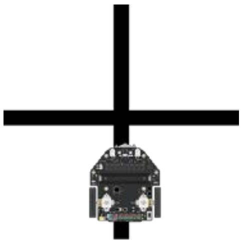
Maqueen Plus has to turn left(call State when turning left function) at both two junctions based on the “left-hand rule”. The flowchart is shown as follows:



4. T-junction and Cross-junction

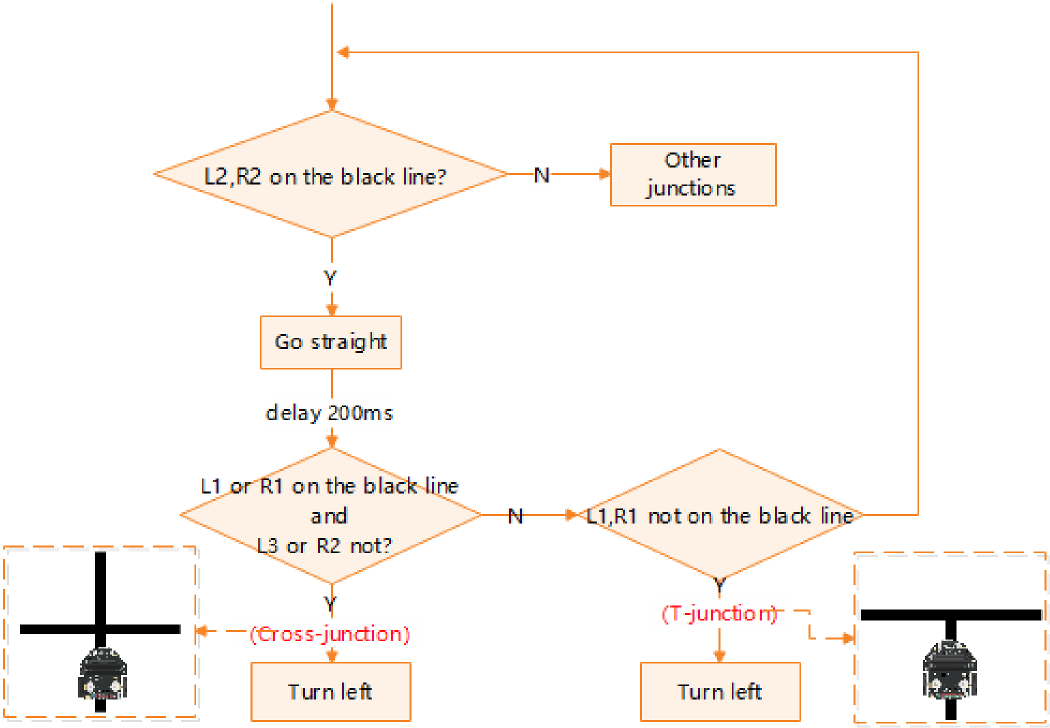
The sensor readings of L2, L1, R1 and R2 will be "1111" when Maqueen Plus arrived at a T-junction or Cross-junction. How does it judge the junction type?

Note: when L2 and R2 both output 1, then it must be arriving at "T-junction" or "Cross-junction".

	
T-junction	Cross-junction

Similarly, here Maqueen Plus also needs to move forward slightly to cross the junction for determining the junction type. If L1 and R1 output 1, it must be a "cross-junction; if 0, "T-junction".

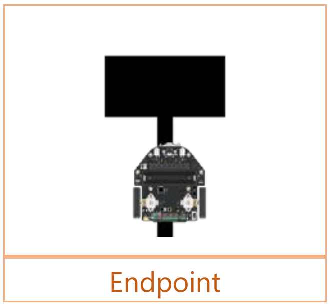
Maqueen Plus needs to turn left at both two junctions as per "left-hand rule".



5. Endpoint

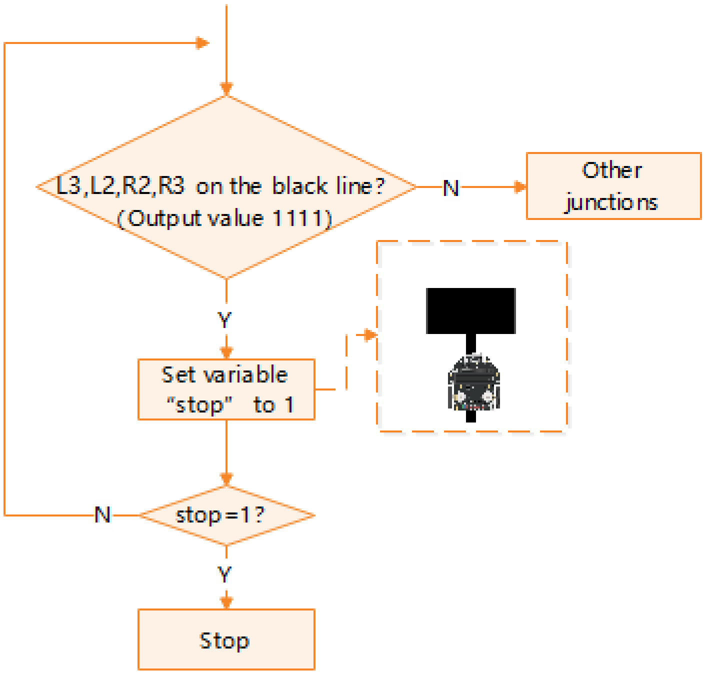
When arriving at the endpoint, the values of line-tracking sensors of L3, L2, L1, R3, R2, R1 will be "111111", and Maqueen Plus stop moving at this time.

Note: when L3, L2, R2, R3 all output 1, it arrives at the endpoint.

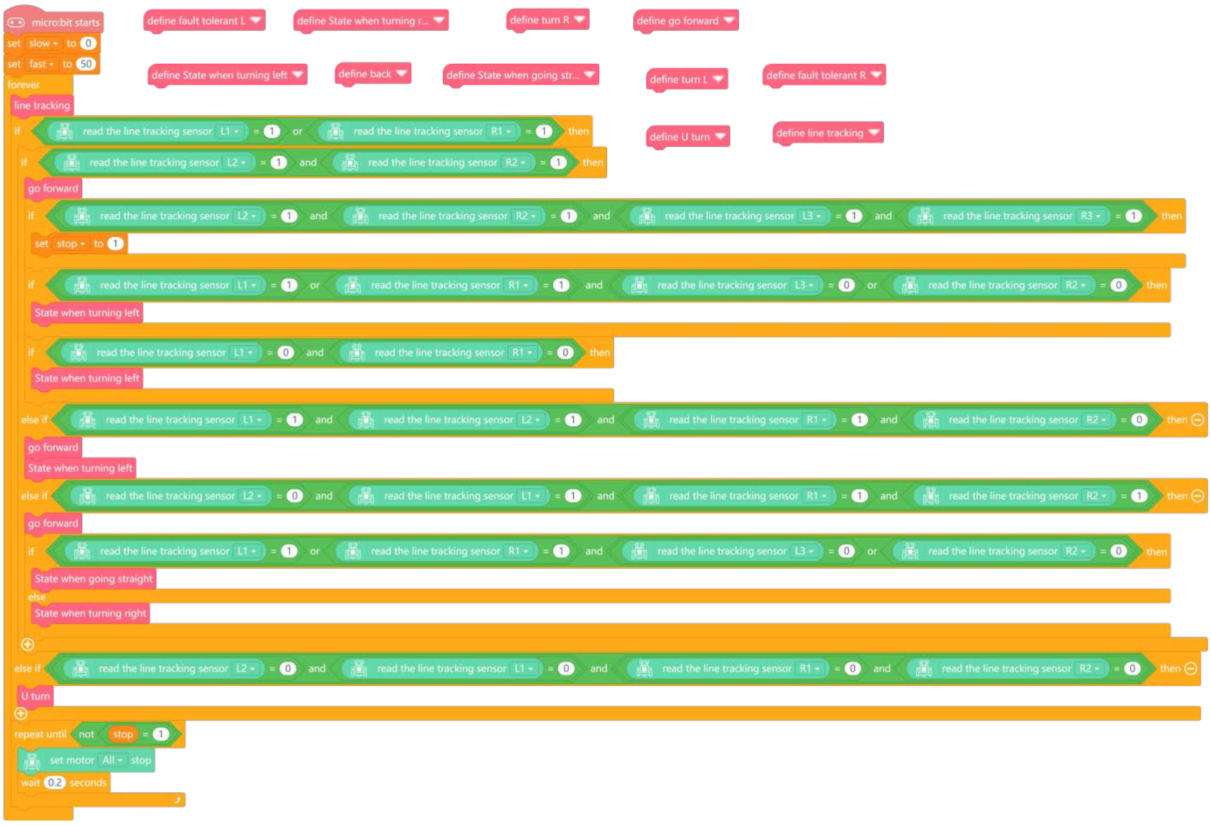


Endpoint

The related flowchart is shown below:



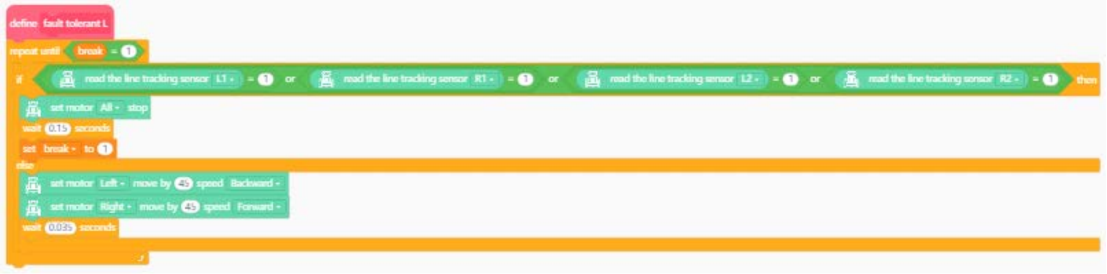
1. Sample Program



```
micro:bit starts
set slow to 0
set fast to 50
forever
  line tracking
  if read the line tracking sensor L1 = 1 or read the line tracking sensor R1 = 1 then
    if read the line tracking sensor L2 = 1 and read the line tracking sensor R2 = 1 then
      go forward
      if read the line tracking sensor L2 = 1 and read the line tracking sensor R2 = 1 and read the line tracking sensor L3 = 1 and read the line tracking sensor R3 = 1 then
        set stop to 1
      if read the line tracking sensor L1 = 1 or read the line tracking sensor R1 = 1 and read the line tracking sensor L3 = 0 or read the line tracking sensor R2 = 0 then
        State when turning left
      if read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 0 then
        State when turning left
    else if read the line tracking sensor L1 = 1 and read the line tracking sensor L2 = 1 and read the line tracking sensor R1 = 1 and read the line tracking sensor R2 = 0 then ⊖
      go forward
      State when turning left
    else if read the line tracking sensor L2 = 0 and read the line tracking sensor L1 = 1 and read the line tracking sensor R1 = 1 and read the line tracking sensor R2 = 1 then ⊖
      go forward
      if read the line tracking sensor L1 = 1 or read the line tracking sensor R1 = 1 and read the line tracking sensor L3 = 0 or read the line tracking sensor R2 = 0 then
        State when going straight
      else
        State when turning right
    ⊕
  else if read the line tracking sensor L2 = 0 and read the line tracking sensor L1 = 0 and read the line tracking sensor R1 = 0 and read the line tracking sensor R2 = 0 then ⊖
    U turn
  ⊕
  repeat until not stop = 1
  set motor All stop
  wait 0.2 seconds

define fault tolerant L
define State when turning r...
define turn R
define go forward
define State when turning left
define back
define State when going str...
define turn L
define fault tolerant R
define U turn
define line tracking
```

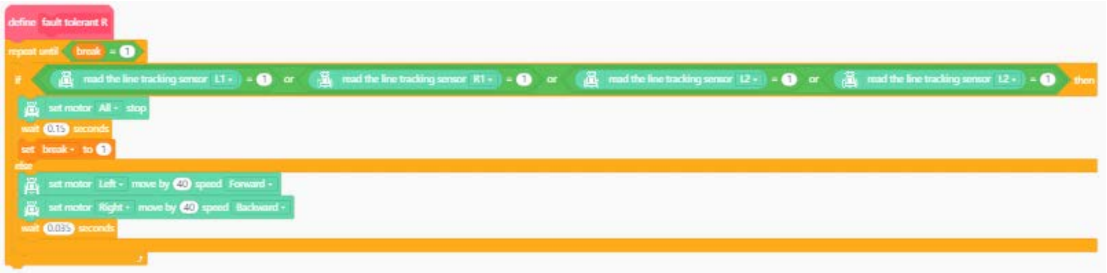
Details of some important functions:

fault_tolerant_L: fault-tolerant function when turning left



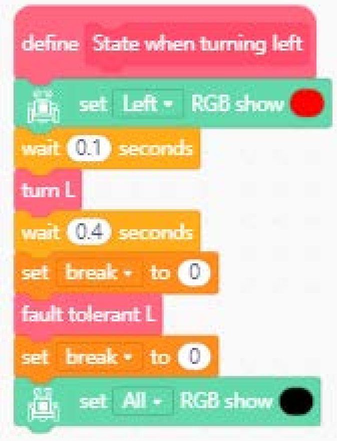
```
define fault tolerant L
repeat until break = 1
    if read the line tracking sensor L1 = 1 or read the line tracking sensor R1 = 1 or read the line tracking sensor L2 = 1 or read the line tracking sensor R2 = 1 then
        set motor All stop
        wait 0.15 seconds
        set break to 1
    else
        set motor Left move by 45 speed Backward
        set motor Right move by 45 speed Forward
        wait 0.035 seconds
```

fault_tolerant_R: fault-tolerant function when turning right



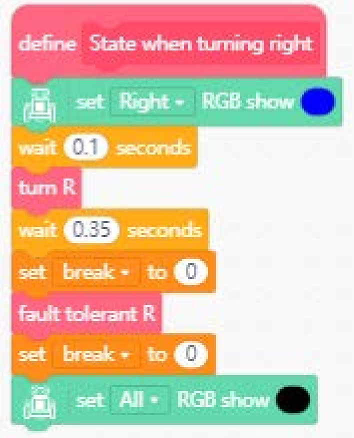
```
define fault tolerant R
repeat until break = 1
    if read the line tracking sensor L1 = 1 or read the line tracking sensor R1 = 1 or read the line tracking sensor L2 = 1 or read the line tracking sensor L2 = 1 then
        set motor All stop
        wait 0.15 seconds
        set break to 1
    else
        set motor Left move by 40 speed Forward
        set motor Right move by 40 speed Backward
        wait 0.035 seconds
```

State when turning left: the left RGB LED keeps on in red when turning left at a junction.



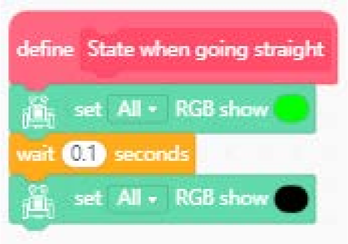
```
define State when turning left
set Left RGB show 🔴
wait 0.1 seconds
turn L
wait 0.4 seconds
set break to 0
fault tolerant L
set break to 0
set All RGB show ⚫
```

State when turning right: the right RGB LED keeps on in blue when turning right at a junction.



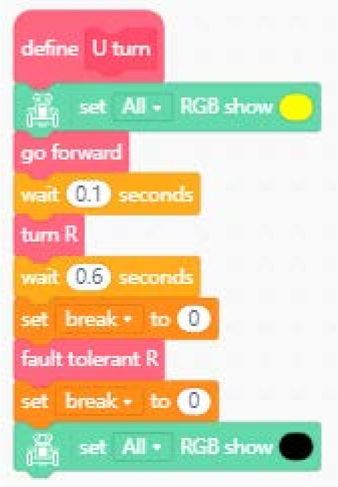
```
define State when turning right
set Right RGB show 🔵
wait 0.1 seconds
turn R
wait 0.35 seconds
set break to 0
fault tolerant R
set break to 0
set All RGB show ⚫
```

State when going straight: all the RGB LEDs turn on in green when going straight at a junction.



```
define State when going straight
    set All ▾ RGB show ●
    wait 0.1 seconds
    set All ▾ RGB show ●
```

U-turn: all the RGB LEDs keep on in yellow when turning around at a dead end.

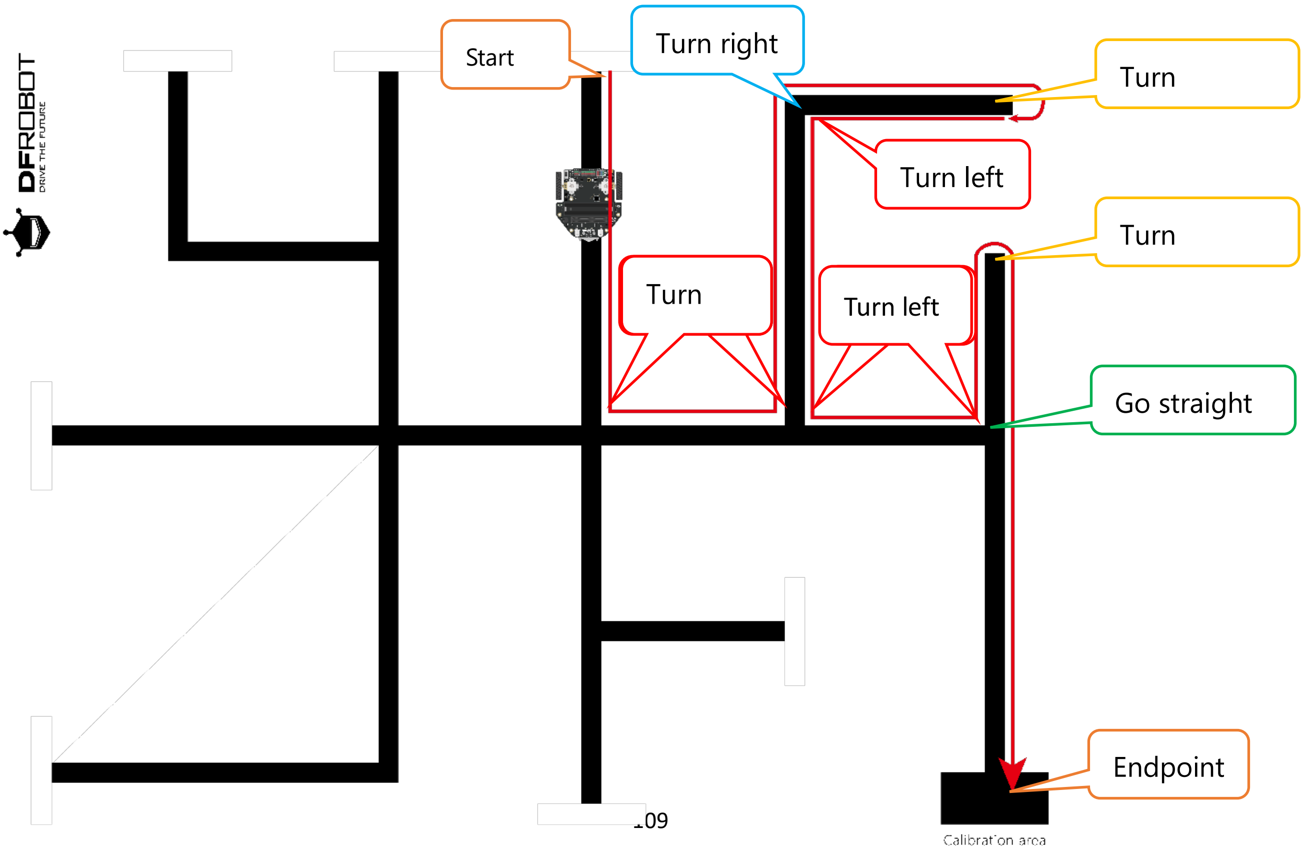


```
define U turn
    set All ▾ RGB show ●
    go forward
    wait 0.1 seconds
    turn R
    wait 0.6 seconds
    set break ▾ to 0
    fault tolerant R
    set break ▾ to 0
    set All ▾ RGB show ●
```

Note: the complete program is attached at the end of the tutorial.

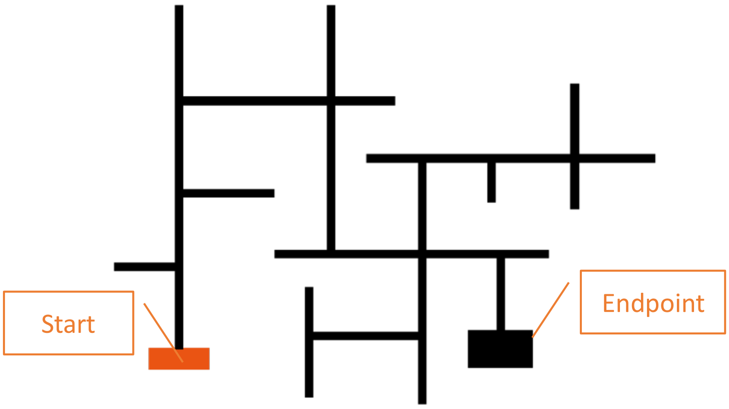
Effect Display

Put Maqueen Plus at a starting point, as shown below. The driving routes: start → turn left → turn left → turn right → turn around → turn left → turn left → turn left → turn around → go straight → endpoint.



Project Development

Now, Maqueen Plus can find the optimal route of the maze. If we put it on a more complicated map like the one below, will the learned algorithm work here? If not, what change we have to make? Try calculating the best route of the maze below based on the tips.



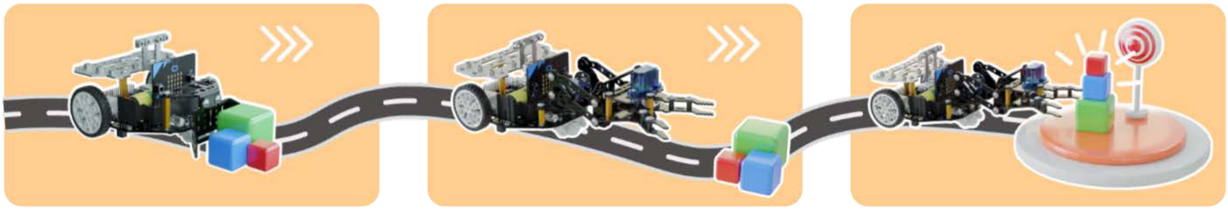
Tip:

1. Use black electrical tape to make the maze map.

Project 7: Relay Transport

Relay Transport is to deliver cargo in a segmented way using one or more means of transportation, by which to greatly improve the efficiency. The organization of relay transport includes several parts: calculate the transport capacity of different facilities, arrange cargo handover date and place, unloading and loading, etc.

Based on that, can we use two or more Maqueen Plus cars to make a relay transport project? For example, the first Maqueen Plus delivers cargo to a transfer station, then the second Maqueen Plus transport the cargo to the endpoint.



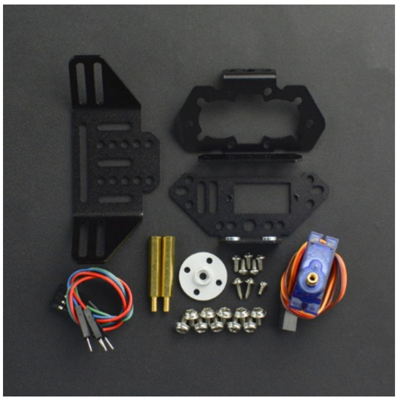
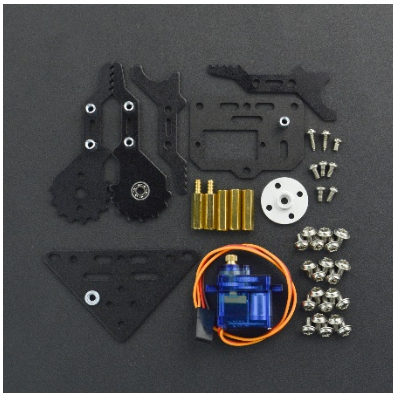
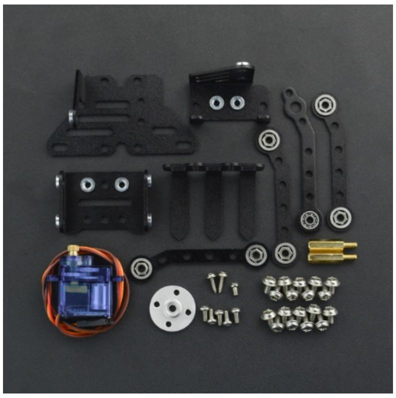
Function

In this project we will be using GamePad to control Maqueen bulldozer and mechanic Beetle, and let them transfer cargo to the endpoint by relay, and stack up neatly.

Transport Rule: control the bulldozer to push the cargo to the intermediate transit point, and then use the mechanic beetle to deliver cargo to the endpoint and put them in order. The project is completed successfully only when the cargo is stacked up neatly.

Bill of Materials

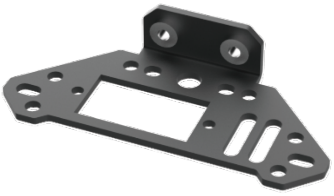
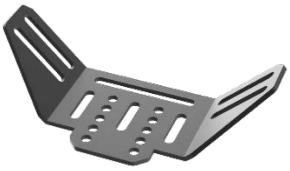
micro:bit ×1	**Maqueen Plus ×2**

	
GamePad Remote Controller ×1	**Push Kit**
	
Beetle Kit	Fork Kit
Upgraded Mechanic Beetle Kit	

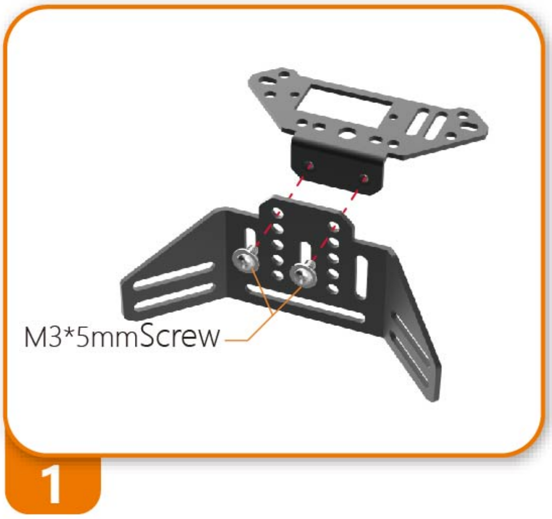
Hardware Connection

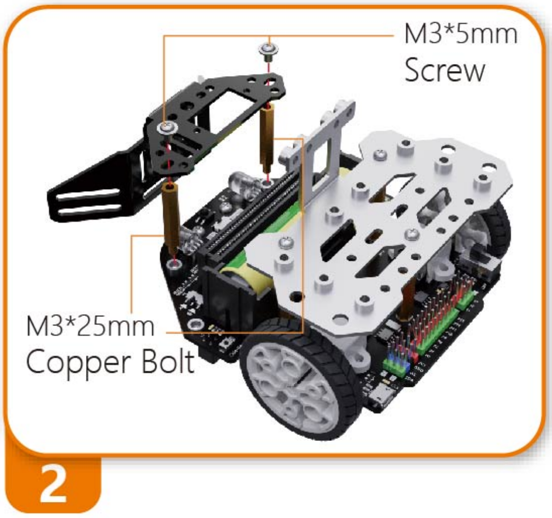
1. Assembly: install the push kit onto Maqueen Plus according to the instruction below.

Accessory List

	
Pan-tilt-zoom Servo Mount Plate X1	**Trolley Plate X1**
M3*25mm Copper Pillar X2	**M3*5mm Screw X4**

Installation Steps:

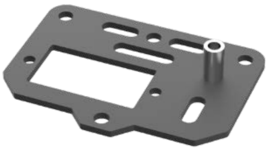
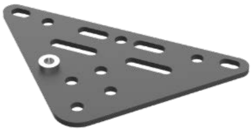


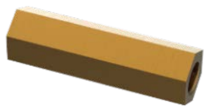
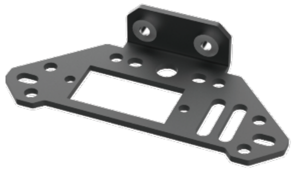
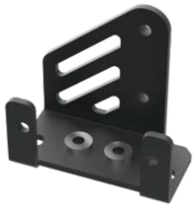
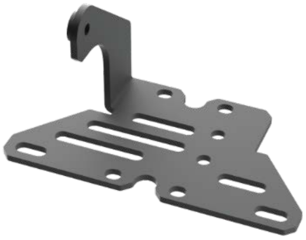
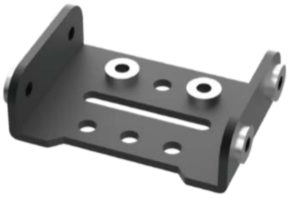
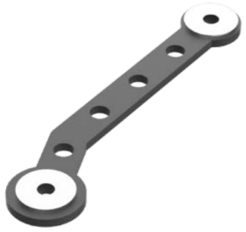
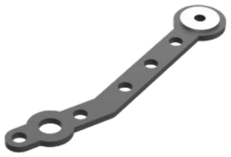


Install the lift-type beetle onto Maqueen Plus.

Note: the lift-type beetle is composed of Maqueen beetle and forklift, and you just have to find the related accessories in the mechanic kit.

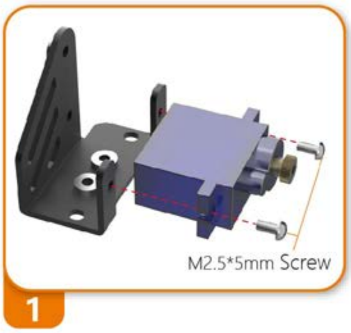
Bill of Material:

			
Gripper Driven Forearm X1	**Gripper Servo Forearm X1**	**Gripper Panel X1**	**Gripper Triangle Panel X1**
M3*5mm Screw X30	**Gripper Upper Arm X2**	**Servo X2**	**M3*15mm Copper Pillar X4**

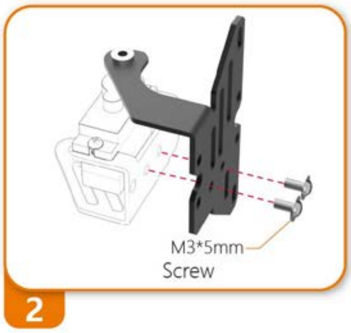
			
M3*17 mm Copper Pillar X3	**M2.5*5mm Screw X6**	**Pan-tilt-zoom Servo Mount Plate X1**	**Arm Servo Base X1**
			
Arm Base Plate X1	**Arm Plate X1**	**Arm Linkage X3**	**Servo Arm Linkage X1**

Assembly Steps:

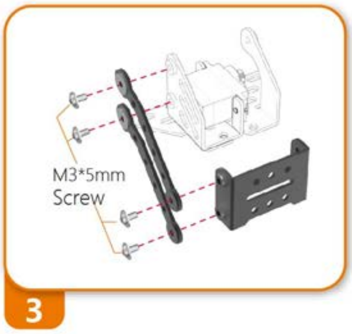
Install the lift-type part, as shown below:



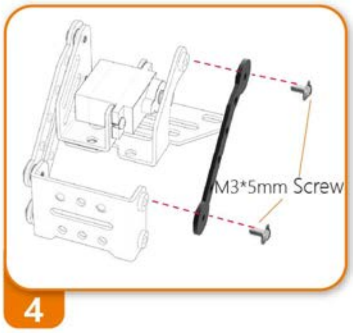
1



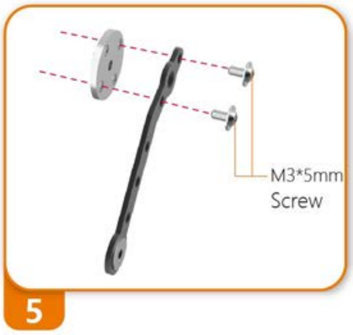
2



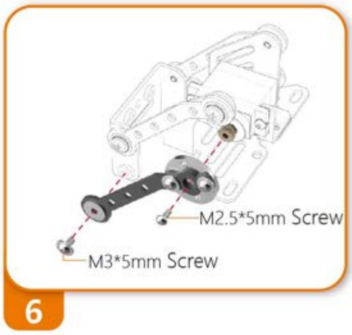
3



4

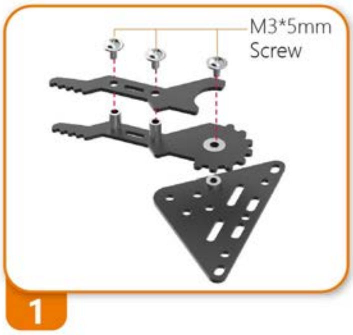


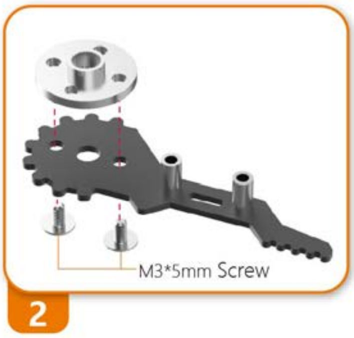
5

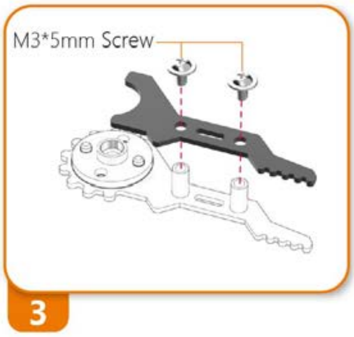


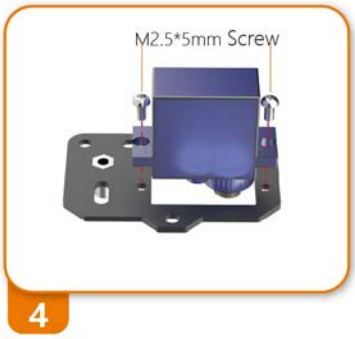
6

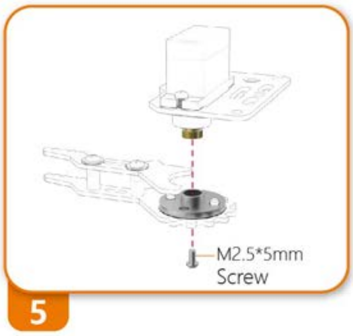
Then install the gripper part:

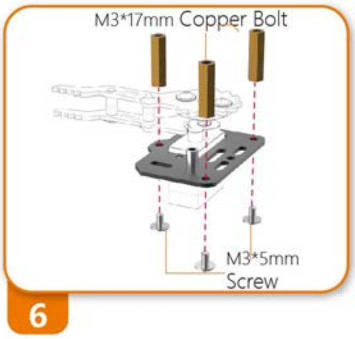


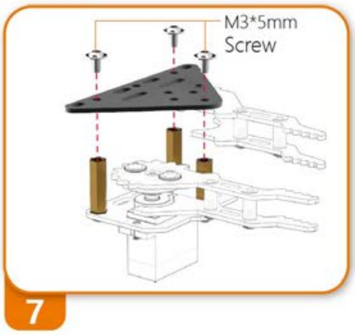




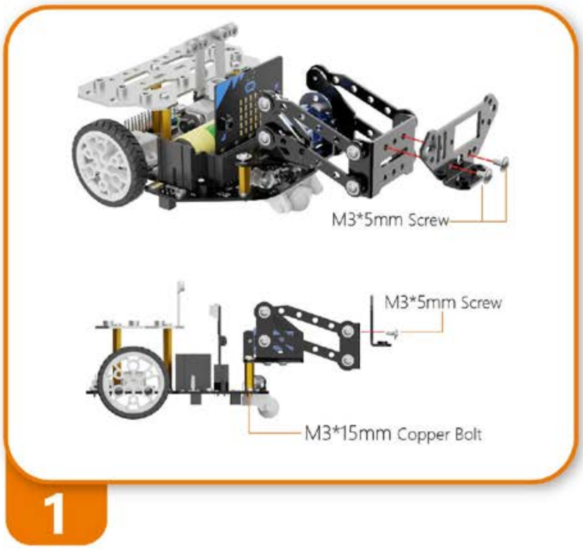


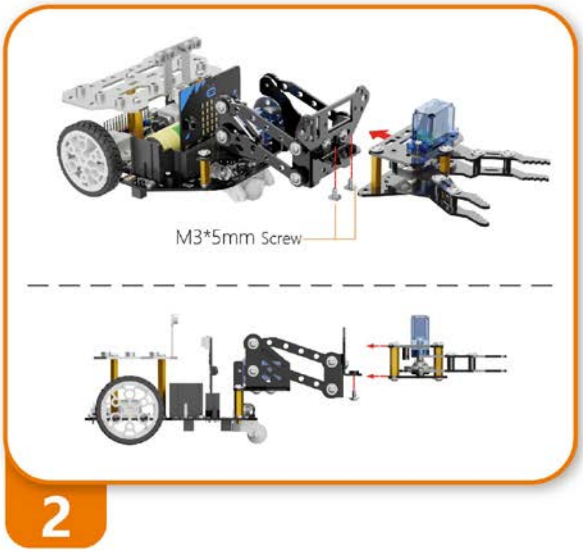




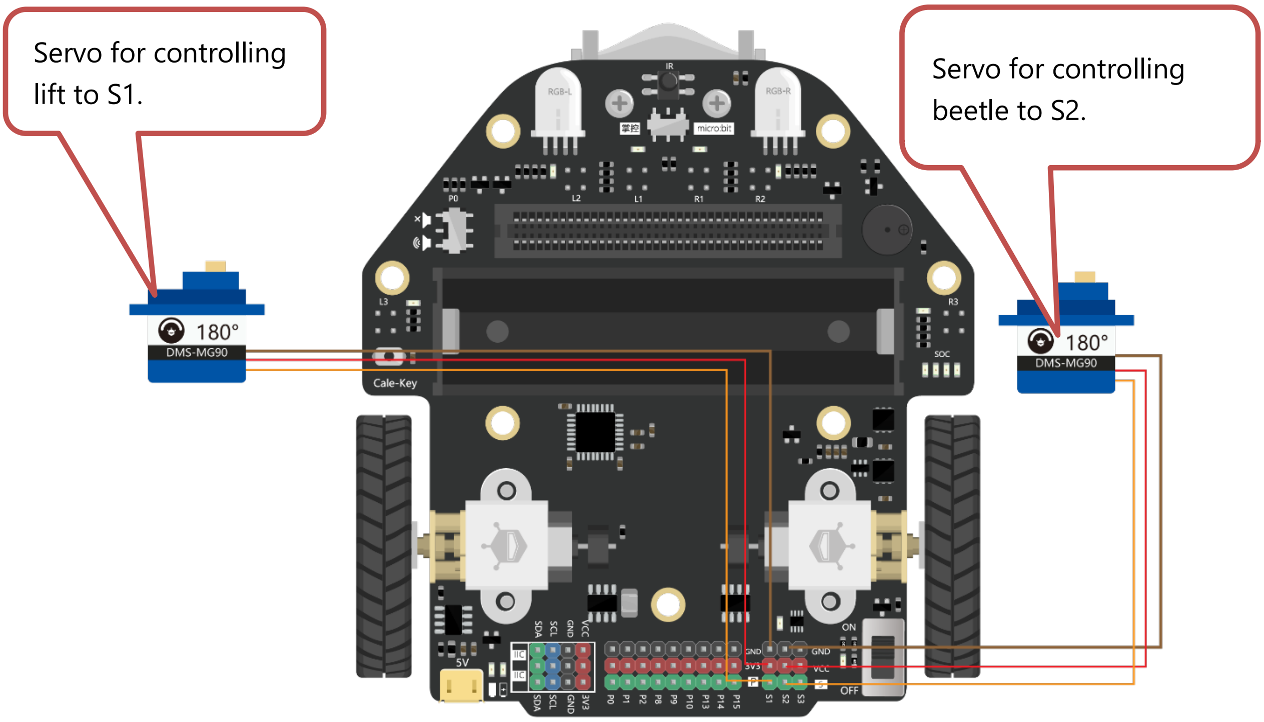


Install the two parts onto Maqueen Plus:





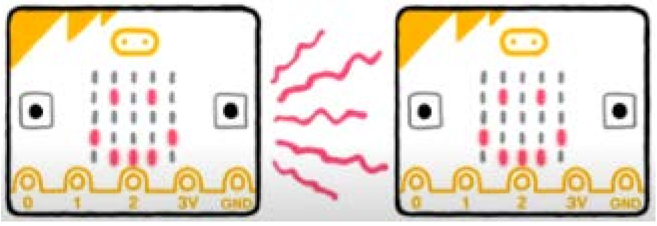
2. Connection: connect the two servos of the lift-type beetle to S1 and S2: the servo for controlling lift goes to S1, beetle to S2. Please do not reverse them.


Servo for controlling lift to S1.

Servo for controlling beetle to S2.

Knowledge Field

This project will be mainly making use of wireless communication to control the Maqueen bulldozer and lift-type mechanic beetle. Micro:bit itself supports wireless communication that can be directly used to transmit data between mainboards.



In wireless communication, a mainboard can be used as a transmitter or receiver. In this relay transport, the Gamepad will be the transmitter, and Maqueen bulldozer and beetle will be the receiver.

1. What is wireless communication?

Radio communication is a kind of communication that uses the electromagnetic signal to transmit information across space instead of wire, optical fiber and other media. At present, this technology is mainly used in radio and television, cell phones, remote control, wireless networking and satellite communication.

2. How does wireless communication work?

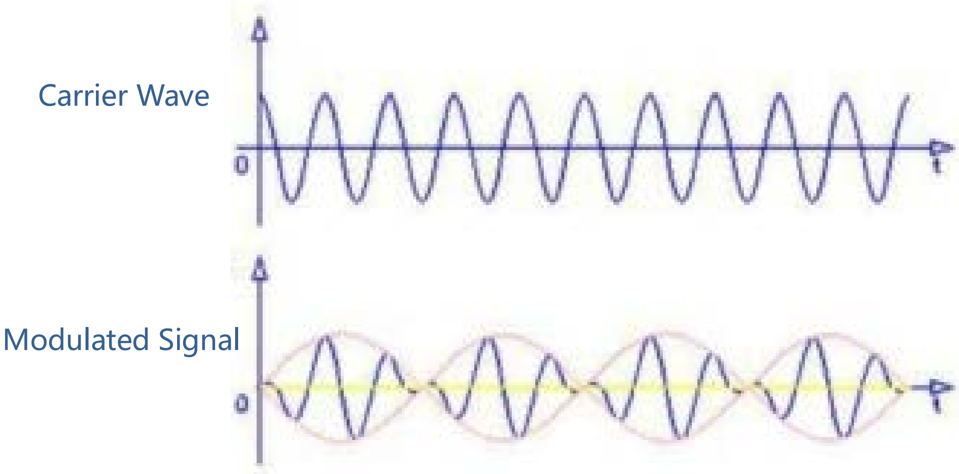
The commonly-seen wireless communication used in mobile phones, WiFi or even satellite propagates data by radio waves.

Information Encoding: radio waves, (a kind of electromagnetic wave similar to visible light) have amplitude, frequency and phase elements that can be modulated by the transmitter, by which to encode and transfer information. Basically, this is how radio wave transmits.

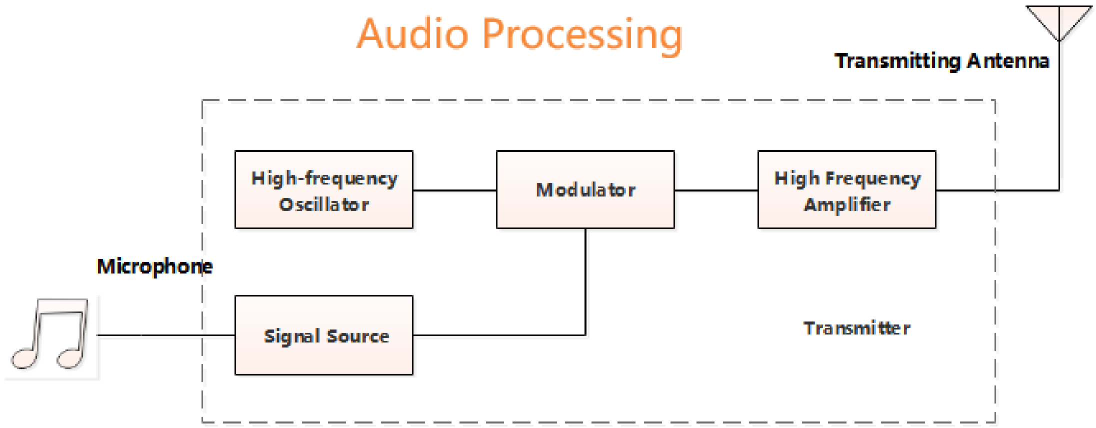
Transfer Low-frequency Signal: it is known that human sound can only propagate through a limited distance. If we want our voice to travel further, how can we make that happen?

A hypothesis like this was made by scientists: If we want to reach a destination, it is for sure that walking is not as fast as taking a car. So, if the transmission of low frequency (audio) can also be like taking a car, get on a car, arrive at the destination and get off, then the long-distance transmission of low-frequency (audio) can be completed. But, where do we find a car for low-frequency?

Travel by High-frequency: high-frequency electromagnetic waves can carry more information per unit time than low-frequency waves, and perform better in keeping the completeness of the information. Hence, we can “store” low-frequency signals onto these high-frequency waves and send them out.



Signal transmitting: different transmitters employ high-frequency of various frequencies, and they do not interfere with each other. This is the process of signal transmitting.

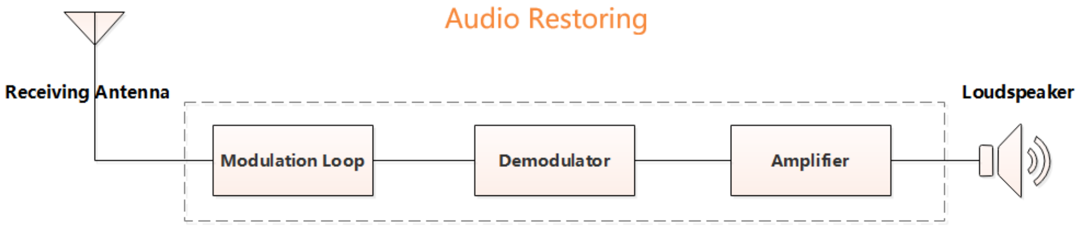


How can we let low-frequency "get off the car"?

Signal Receiving: after the radio wave is transferred to the receiving place, the receiver will receive the signal and recover it.

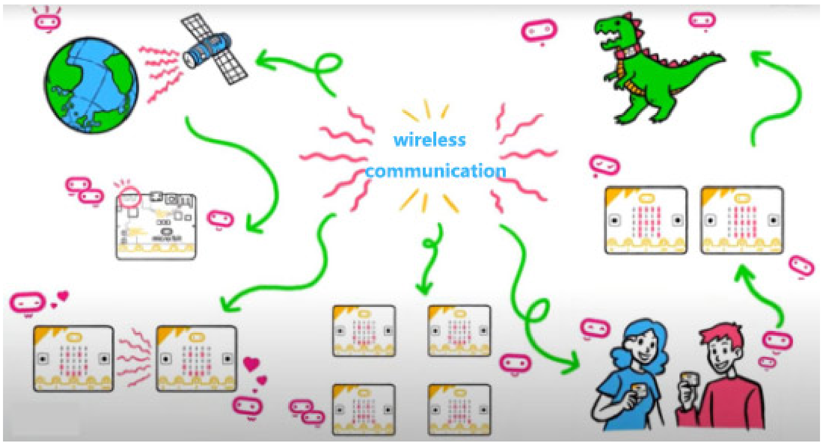
Modulate Carrier Frequency: the signal received by the receiving antenna in the same period of time not only includes the signal we need, but also many signals of other frequencies. The reason why these radio signals adopt different carrier wave frequencies(properties of high-frequency electromagnetic wave) is to let the receiver try to "select" the signals they want to receive according to the carrier wave(high-frequency electromagnetic wave with audio).

Signal Recover: after receiving the high-frequency electromagnetic wave, we can get our audio by the restoration process.



3. micro:bit Radio Communication

The principle of micro:bit radio communication is based on radio waves. There is an antenna on the back of micro:bit that allows data transmission between two micro:bit boards.



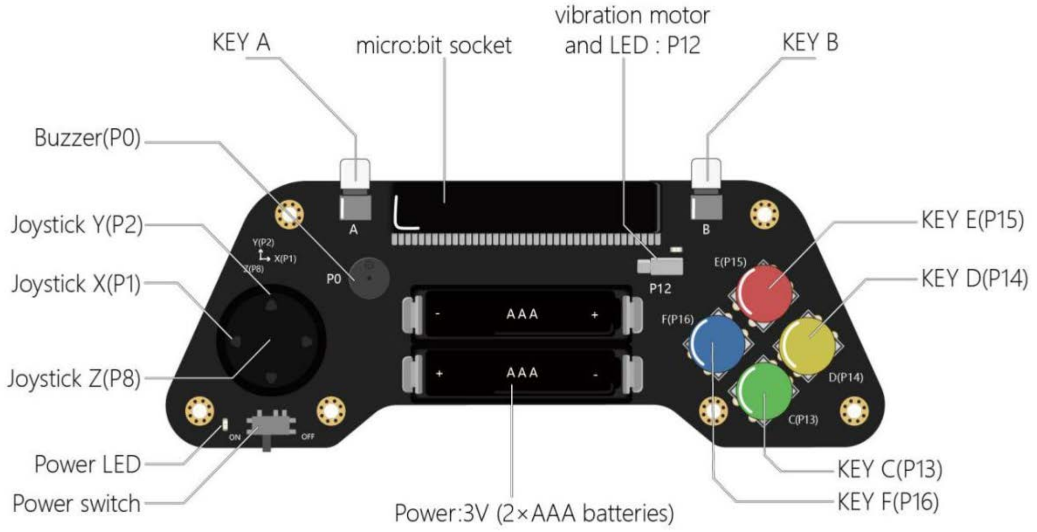
Project Development

The Gamepad will be used to control Maqueen bulldozer and beetle. And they will transfer the goods to the destination in a relaying way. There are 3 steps in this project.
First of all, complete the program of the transmitter-Gamepad, use keys A and B to switch radio setting group and send commands to receiving end; then completes the program of the receiving end of the bulldozer, and executes the forward, backward, left turn and right turn commands according to the received command; finally completes the program of the receiving end of the lift-type beetle, and executes the instructions of forward, backward, left turn, right turn, gripper up, down, open and close according to the received commands.

Task 1: Program for the GamePad

Program Design

Step 1 Get to know Gamepad
It is a micro:bit-based gamepad with a joystick, and can be used as a remote control handle when plugging a micro:bit in.

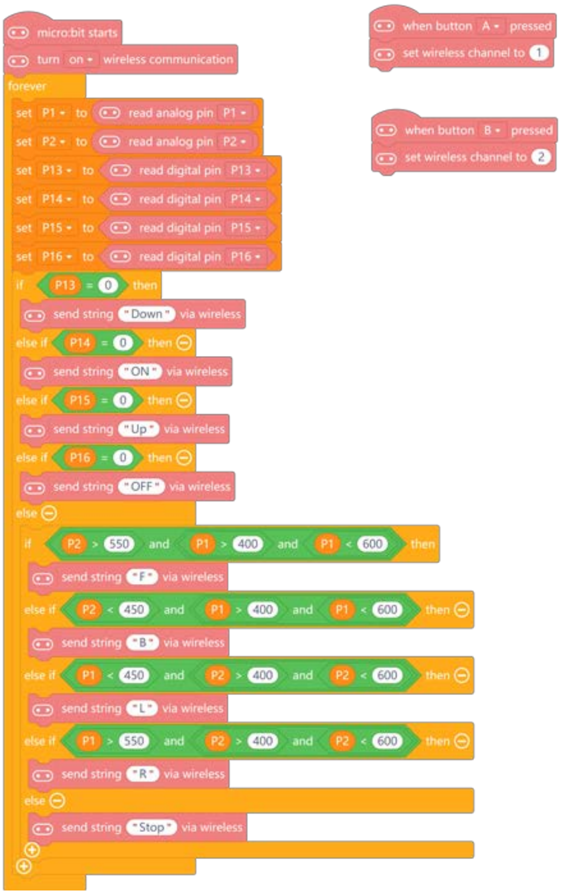


Step 2 Function

In the project, the joystick will be mainly used to control the movement of Maqueen bulldozer and lift-type mechanic beetle. The buttons on the Gamepad will be used for moving the gripper up and down, and closing and opening the gripper.
The detailed commands are shown below:

Joystick Status	Pin	Command
Press down button A	A	Enter radio group 1
Press down button B	B	Enter radio group 2
Pull up joystick 	P2	“F”
Pull down joystick 	P2	“B”
Pull joystick to left 	P1	“L”
Pull joystick to right 	P1	“R”
Joystick stays in the middle 	P8	“Stop”
Press down button E	P15	“Up”
Press down button C	P13	“Down”
Press down button F	P16	“OFF”
Press down button D	P14	“ON”

Sample Program



```
micro:bit starts
    turn on wireless communication

forever
    set P1 to read analog pin P1
    set P2 to read analog pin P2
    set P13 to read digital pin P13
    set P14 to read digital pin P14
    set P15 to read digital pin P15
    set P16 to read digital pin P16
    if P13 = 0 then
        send string "Down" via wireless
    else if P14 = 0 then
        send string "ON" via wireless
    else if P15 = 0 then
        send string "Up" via wireless
    else if P16 = 0 then
        send string "OFF" via wireless
    else
        if P2 > 550 and P1 > 400 and P1 < 600 then
            send string "F" via wireless
        else if P2 < 450 and P1 > 400 and P1 < 600 then
            send string "B" via wireless
        else if P1 < 450 and P2 > 400 and P2 < 600 then
            send string "L" via wireless
        else if P1 > 550 and P2 > 400 and P2 < 600 then
            send string "R" via wireless
        else
            send string "Stop" via wireless

when button A pressed
    set wireless channel to 1

when button B pressed
    set wireless channel to 2
```

Note: the complete program is attached at the end of the article.

Effect Display

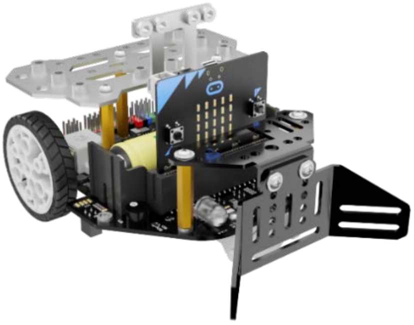
Press down button A to enter radio group 1 control mode, button B to enter radio group 2 control mode. Move the joystick up to send the forward command “F”; move it down to send backward command “B”; move the joystick left or right to send left/right turn command “L” or “R”. Press down button “E” to send command “Up”; button C to send “Down”; button D to send “ON”; button F to send “OFF”.

Task 2: Maqueen Bulldozer

Program Design

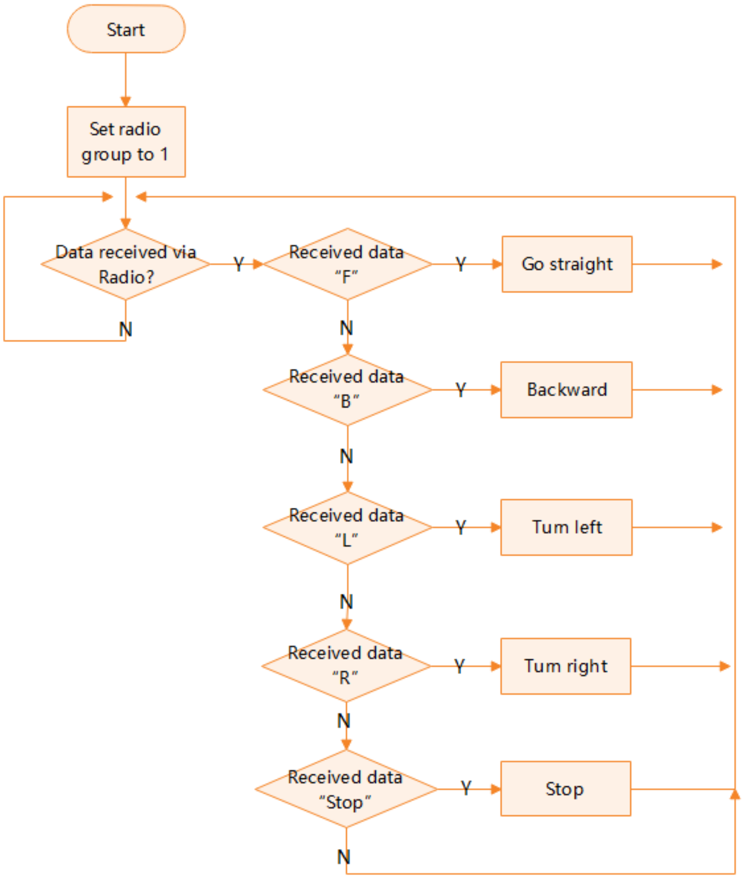
Step 1 Maqueen Bulldozer

The bulldozer blade on the front of the Maqueen Plus can be used to push objects to the designated place, and it is suitable for applications like football matches, venue cleaning and so on.

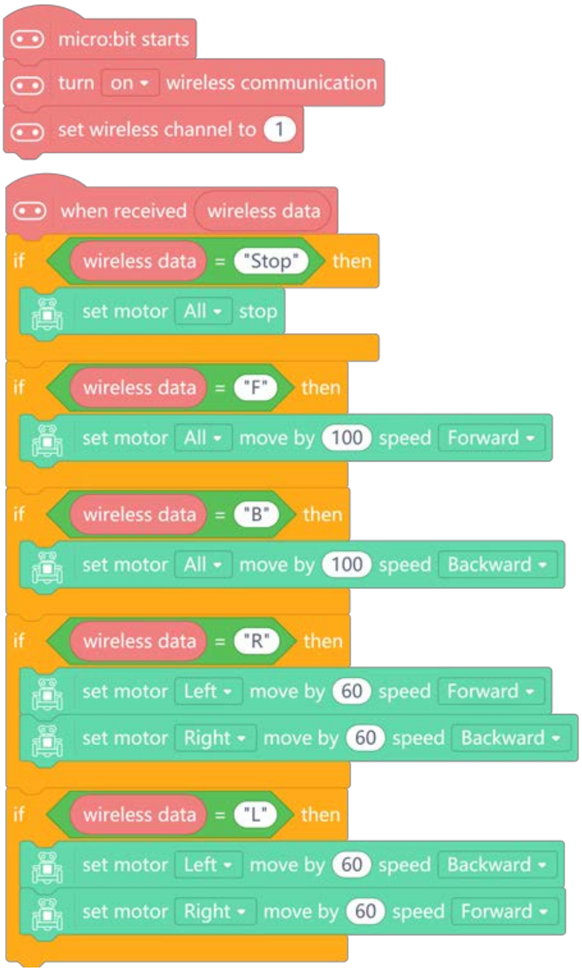


Step 2 Function Analysis

In this project, the commands sent from the gamepad will be used to control the movement of the bulldozer. The flowchart is shown below:



Sample Program



```
micro:bit starts
    turn on wireless communication
    set wireless channel to 1

when received wireless data
    if wireless data = "Stop" then
        set motor All stop
    if wireless data = "F" then
        set motor All move by 100 speed Forward
    if wireless data = "B" then
        set motor All move by 100 speed Backward
    if wireless data = "R" then
        set motor Left move by 60 speed Forward
        set motor Right move by 60 speed Backward
    if wireless data = "L" then
        set motor Left move by 60 speed Backward
        set motor Right move by 60 speed Forward
```

Note: the complete program will be attached at the end of the article.

Effect Display

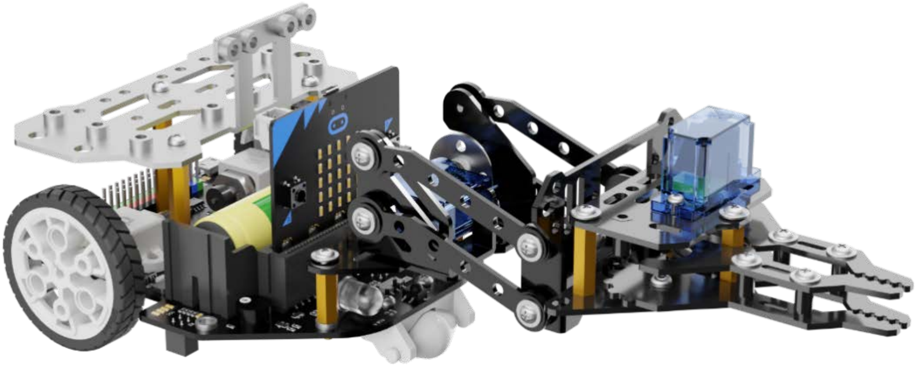
Press down button A to enter the radio group 1, also the bulldozer controlling mode. Move the joystick to control the Maqueen Bulldozer to move forward/backward and turn left/right.

Task 3: Lift-type Mechanic Beetle

Program Design

Step 1 Introducing lift-type Mechanic Beetle

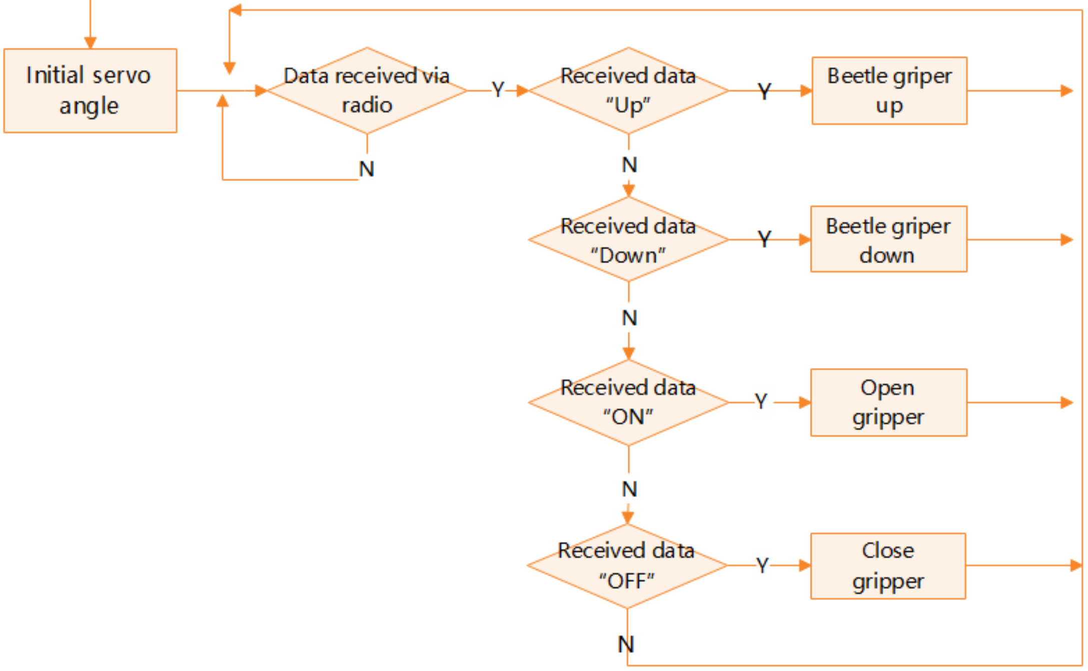
The lift-type mechanic beetle is composed of Maqueen forklift and beetle. And it will be used to pick up items and transfer them. The maximum height the beetle's gripper can reach is about 10cm.



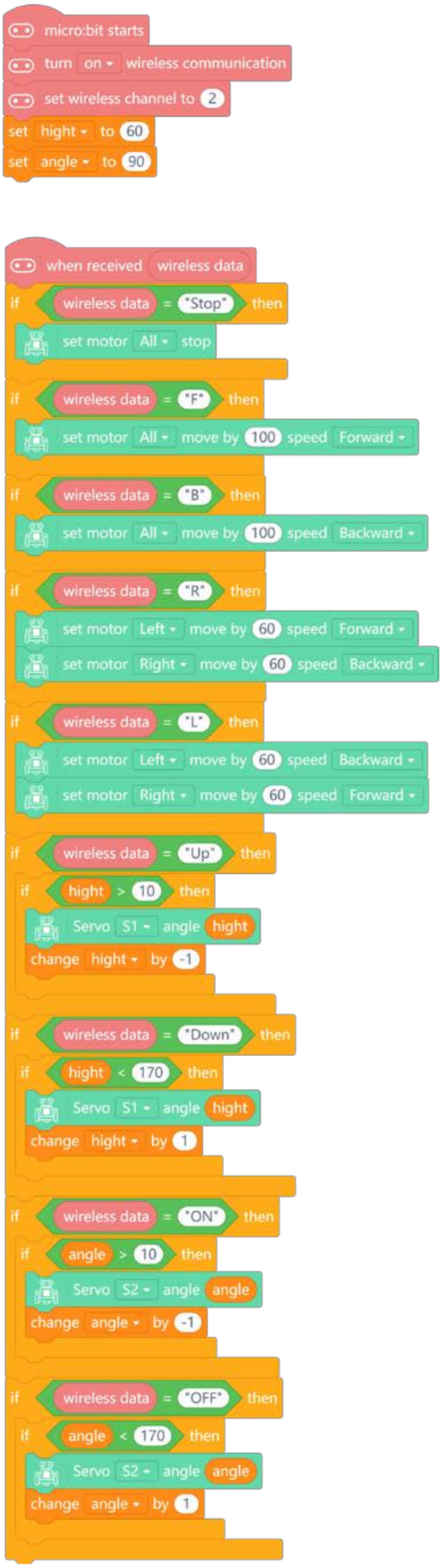
Step 2 Function Analysis

We will be using the joystick to control the Mechanic beetle’s movement, and the buttons to control the gripper to move up/down or close/open. Buttons E and C for up and down, D and F for open and close respectively.

Note：the movement program of the mechanic beetle is the same as the bulldozer above, and so it will be omitted here.



Sample Program

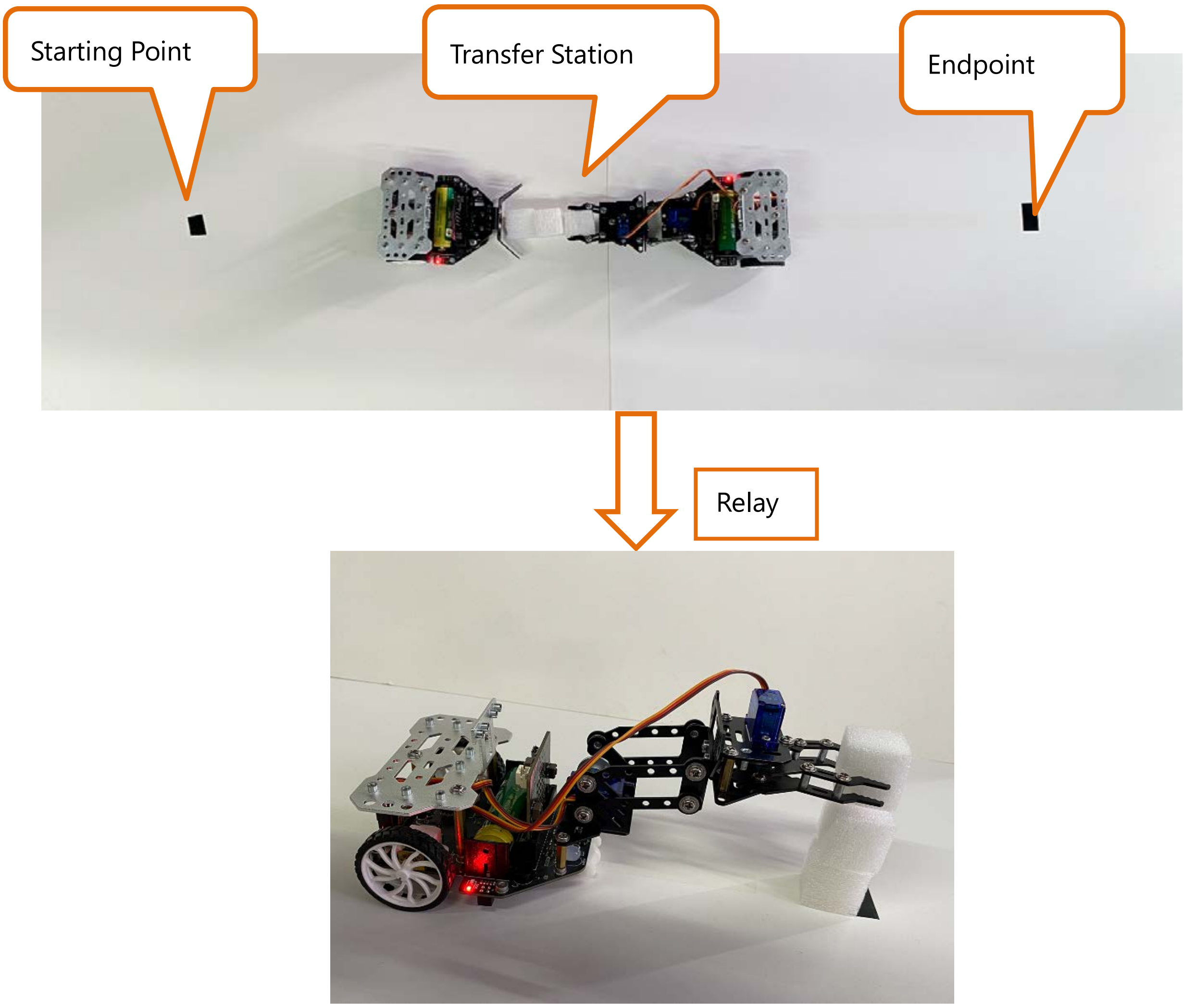


```
micro:bit starts
    turn on ▾ wireless communication
    set wireless channel to 2
    set hight ▾ to 60
    set angle ▾ to 90

when received wireless data
    if wireless data = "Stop" then
        set motor All ▾ stop
    if wireless data = "F" then
        set motor All ▾ move by 100 speed Forward ▾
    if wireless data = "B" then
        set motor All ▾ move by 100 speed Backward ▾
    if wireless data = "R" then
        set motor Left ▾ move by 60 speed Forward ▾
        set motor Right ▾ move by 60 speed Backward ▾
    if wireless data = "L" then
        set motor Left ▾ move by 60 speed Backward ▾
        set motor Right ▾ move by 60 speed Forward ▾
    if wireless data = "Up" then
        if hight > 10 then
            Servo S1 ▾ angle hight
            change hight ▾ by -1
    if wireless data = "Down" then
        if hight < 170 then
            Servo S1 ▾ angle hight
            change hight ▾ by 1
    if wireless data = "ON" then
        if angle > 10 then
            Servo S2 ▾ angle angle
            change angle ▾ by -1
    if wireless data = "OFF" then
        if angle < 170 then
            Servo S2 ▾ angle angle
            change angle ▾ by 1
```

Note: the complete program is attached at the end of the article.

Effect Display

Simulate the scenario of relay transport, put the Maqueen bulldozer to the starting point, and the lift-type beetle to the intermediate transfer station. Press button A to enter bulldozer controlling mode. Move the joystick to control the bulldozer to push the object to the transfer station; Press down button B to enter beetle controlling mode: button D for opening the gripper, F for closing, E for moving up, C for moving down. During the transferring, we can use these buttons to make Maqueen Plus pick up items, and move the joystick to let it transfer the items to the endpoint, and operate the gripper to stack up items neatly.



Stack up

Project Development

Now, Maqueen plus has been able to successfully transport goods to the destination through the mechanic accessories. Next, we'll upgrade the difficulty: get all the Maqueen Plus mechanic car assembled, put them at three transfer stations, then operate the gamepad to control these cars to deliver cargo to the endpoint in order, the team who finished the task with the least time wins.

Tip：

1. Write the corresponding function program for each Maqueen Plus car and Gamepad.
2. Four players control the Mechanic cars with gamepad handles. After the first person's Maqueen Plus pushed the cargo to the first relay point, the second player starts to operate his mechanic car to push the cargo from the first relay point to the second relay point, and so on until the cargo is transported to the destination. The one who finished the task with the least time wins.
3. In the process of the competition, players should only operate their Maqueen Plus cars via GamePad.
4. Once the order is determined, it can not be changed during the competition.