BBC Micro:bit Smart Car -- MakeCode

Product Introduction

Are you curious about the working principle of motor control, ultrasonic ranging and RGB LED lighting? BBC Micro: bit smart car will lead you to explore their mysteries.

The smart car's control board is a micro: bit development board launched by the BBC, which is designed for youth programming education. Its car body is small, easy to carry and simple in structure. Building block holes are reserved for expansion. It is also with a variety of functions, such as ranging, light metering, black and white line detection, infrared remote control and Bluetooth control.

The kit supports graphical programming (MakeCode) and Python. The former helps beginners to understand program logic and develop programming thinking, while the latter, also called MicroPython running on the BBC Micro: bit, is an Python implementation optimized for microcontrollers and embedded systems.

Product Features:

Building block programming, easy to learn (online programming)

Small volume (only half of the adult palm), easy to carry, round shape, comfortable feeling

Convenient assembly, only a few parts need to be assembled

12GFN20 motor, with stable operation and strong power

Strong extensibility, 8 pins, two IIC interfaces and one servo dedicated interface

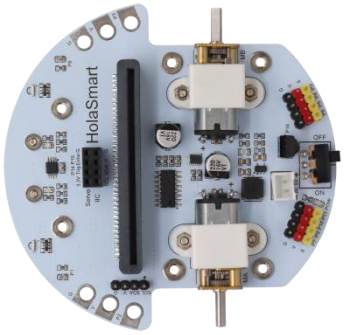
Reserved holes connecting building blocks

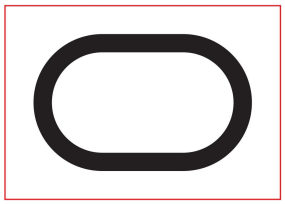
Product Parameters:

1. Connector port input: DC 4-5v
2. Sensor operating voltage: 3V
3. Motor speed: 200RPM
4. Operating temperature: 0-50℃
5. Assembled dimensions: 93x90x39mm
6. Environmental attributes: ROHS

Kit List

When receiving this product, please check on the following list to ensure that all accessories are intact. If any components are missing, please contact our sales staff immediately.

#	PIC	NAME	QTY
1		Car drive board	1
2		Holasmart ultrasonic sensor	1
3		Remote control	1
4		3-slot AAA battery holder	1
5		35mm wheel	2

#	PIC	NAME	QTY
6		3M double-sided adhesive tape 50 * 20 * 1mm	1
7		Map	1

What is Micro:bit?

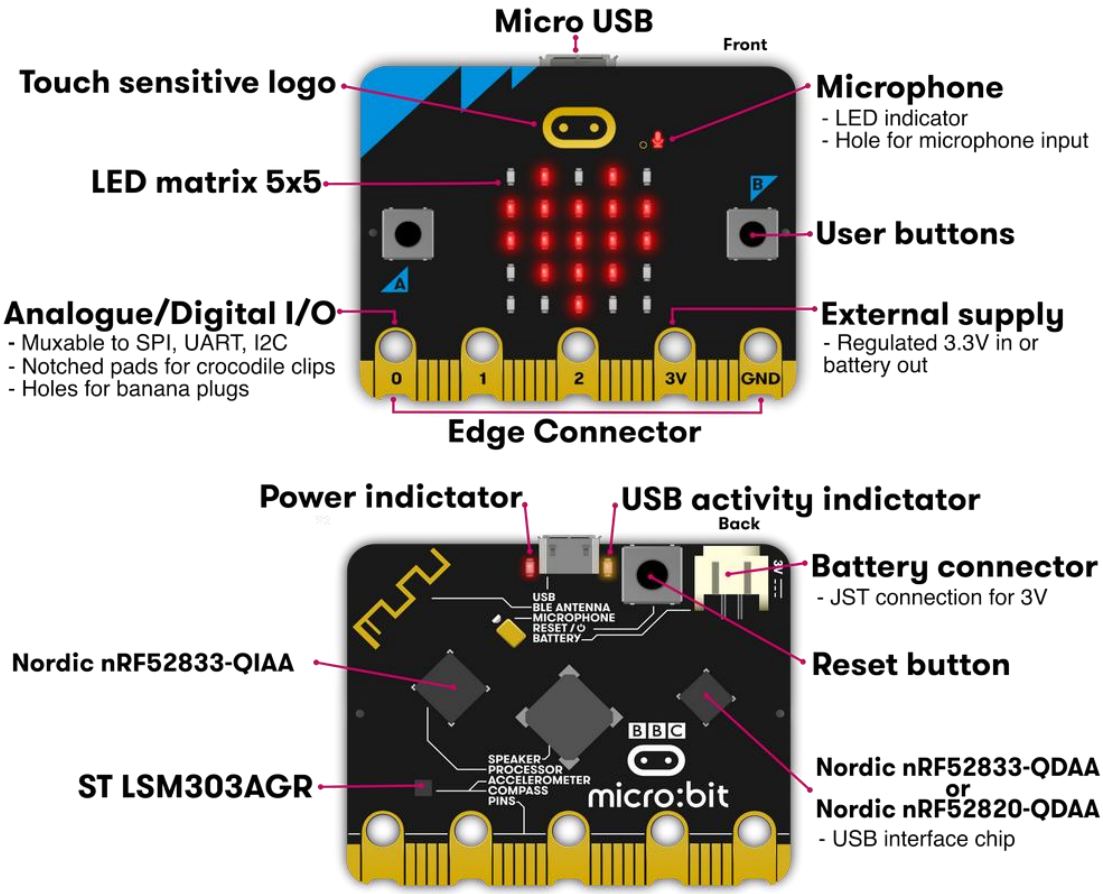
Micro:bit is an open source hardware platform based on the ARM architecture launched by British Broadcasting Corporation (BBC) together with ARM, Barclays, element14, Microsoft and other institutions. The core device is a 32-bit Arm Cortex-M4 with FPU micro-processing.

Though it is just the size of a credit card, the Micro:bit main board is equipped with loads of components,including a 5*5 LED dot matrix, 2 programmable buttons, an accelerometer, a compass, a thermometer, a touch-sensitive logo and a MEMS microphone, a Bluetooth module of low energy, and a buzzer and others. Thus, it also boasts multiple functions.

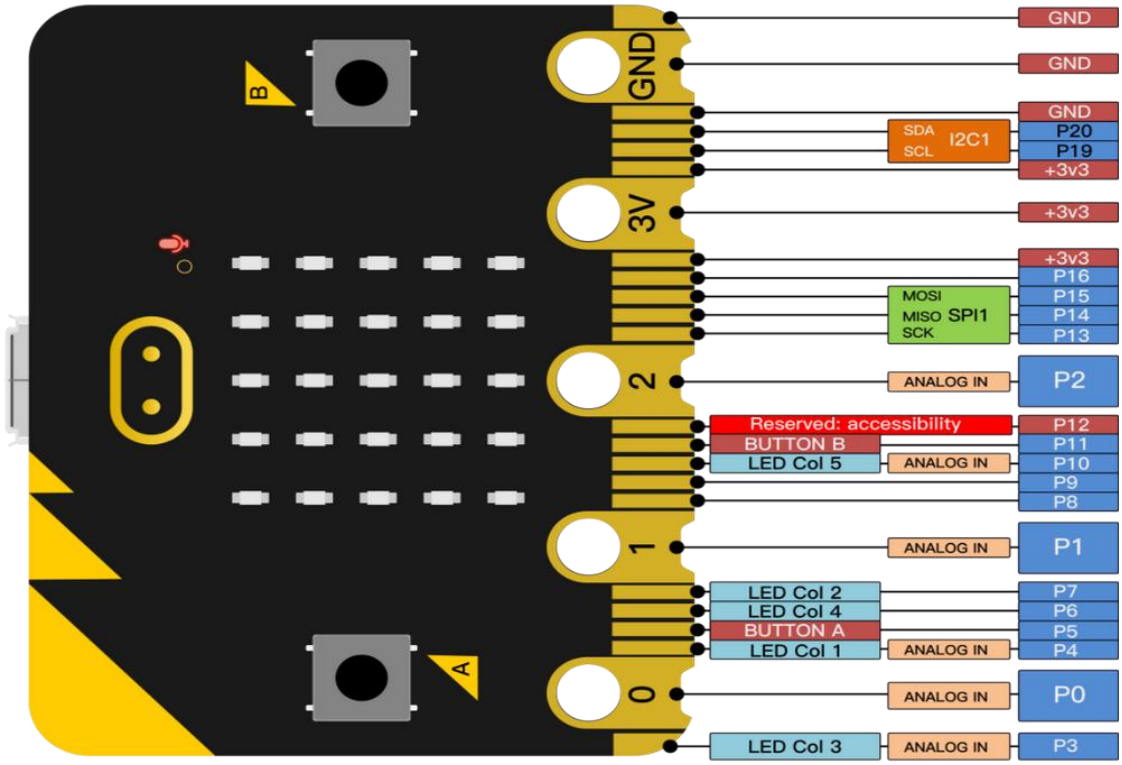
The buzzer built in the other side of the board plays all kinds of sounds without any external equipment. The golden fingers and gears provide a better fixing of crocodile clips. Moreover, this board has a sleeping mode to lower power consumption of batteries when users long-press the Reset & Power button on the back. It is capable of reading the data of sensors, controlling servos and RGB lights, and it is attaching with a shield so as to connect with various sensors. It also supports a variety of programming platforms and is compatible with almost all PCs and mobile devices. Also, driver is required. It is of high integration of electronic modules, and has a serial monitoring function for easy debugging.

The board has found wild applications. It can be applied in programming video games, making interactions between light and sound, controlling a robot, conducting scientific experiments, developing wearable devices. Basically, it always be utilized to make some cool inventions like robots and musical instruments.

1. Micro:bit V2 Mainboard Layout



2. Micro:bit V2 Pin-out



Micro:bit pin functions:

Function	Pin
GPIO	P0, P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P19, P20
ADC/DAC	P0, P1, P2, P3, P4, P10
IIC	P19(SCL), P20(SDA)
SPI	P13(SCK), P14(MISO), P15(MOSI)
PWM(commonly used)	P0, P1, P2, P3, P4, P10
PWM(uncommonly used)	P5, P6, P7, P8, P9, P11, P12, P13, P14, P15, P16, P19, P20
occupied	P3(LED Col3), P4(LED Col1), P5(Button A), P6(LED Col4), P7(LED Col2), P10(LED Col5), P11(Button B)

Visit the official website for more details:

Microbit hardware

https://microbit.org/guide/hardware/pins/

3. Notes for the Application of Micro:bit

- It is recommended to cover it with a silicone protector to prevent short circuit for it has a lot of sophisticated electronic components.

- Its IO port is very weak in driving since it can merely handle current less than 300mA. Therefore, do not connect it with devices operating in large current, such as servo MG995 and DC motor or it will get burnt. Furthermore, you must figure out the current requirements of the devices before you use them and it is generally recommended to use the board together with a Micro:bit shield.

- It is recommended to power the main board via the USB interface or via the battery of 3V. The IO port of this board is 3V, so it does not support sensors of 5V. If you need to connect sensors of 5 V, a Micro: Bit expansion board is required.

- When using pins(P3, P4, P6, P7 and P10) shared with the LED dot matrix, blocking them from the matrix or the LEDs may display randomly and the data about sensors connected maybe wrong.

- Pin 19 and 20 can not be used as IO ports though the Makecode shows they can. They can only be used as I2C communication.

- The battery port of 3V cannot be connected with battery more than 3.3V or the main board will be damaged.

- Forbid to operate it on metal products to avoid short circuit.

To put it simple, Micro:bit V2 main board is like a microcomputer which has made programming at our fingertips and enhanced digital innovation. And as for programming environment, BBC provides a website: https://microbit.org/code/, which has a graphical MakeCode program easy for use.

Drive Board

1.Introduction

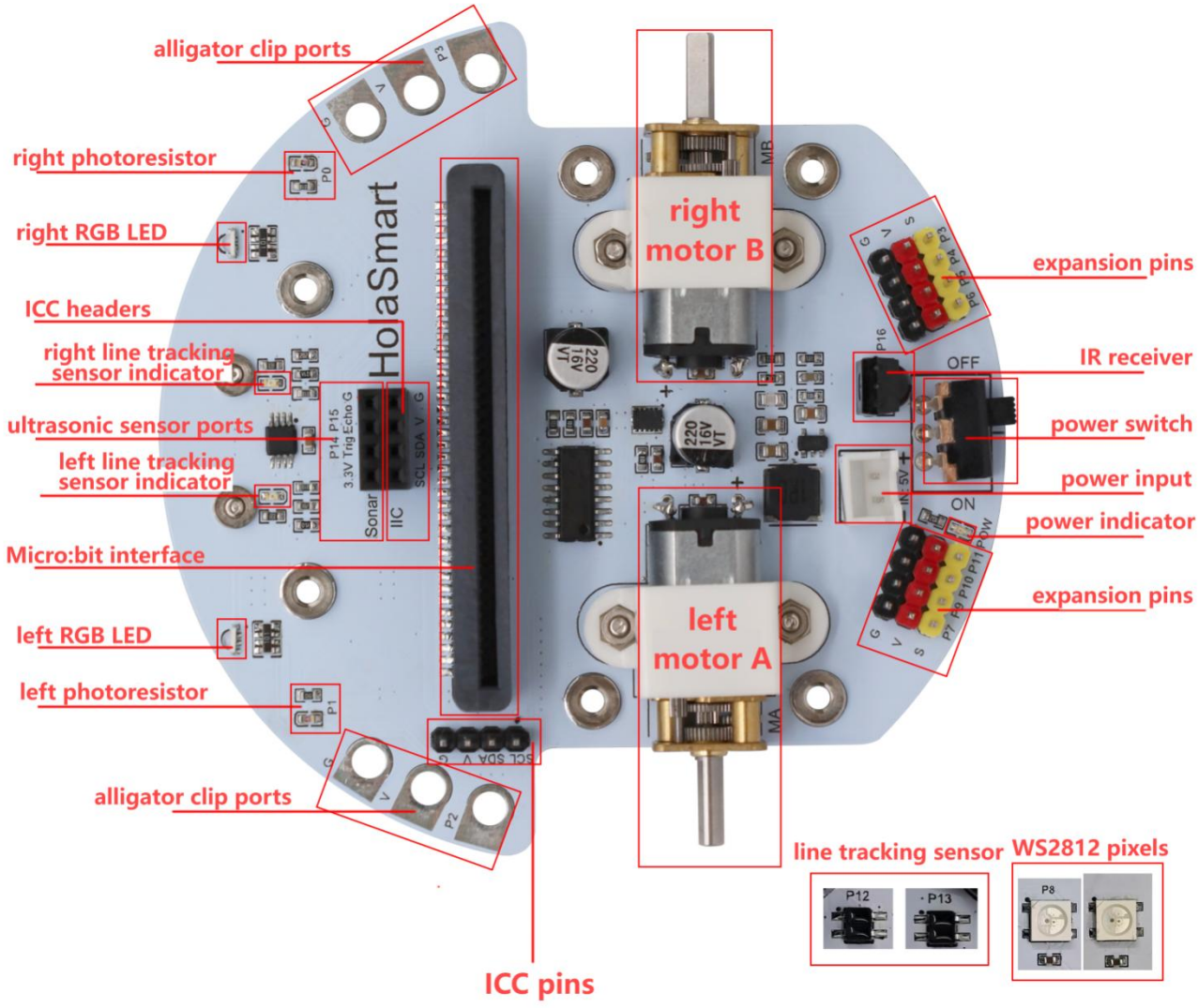
The car drive board integrates the infrared receiver, photoresistors, line tracking sensors, RGB LED, ws2812 pixels, ultrasonic sensor, IIC interface, motors and so on. So there is no worry about wiring. There are also 8 IO ports on the board for expansions.

Due to the shortage of micro: bit IO ports, the board sends commands to STC8G1K08 through IIC, and then ST8G1K08 controls HR8833 motor to drive IC via PWM, so as to control the motor rotation and LED RGB.

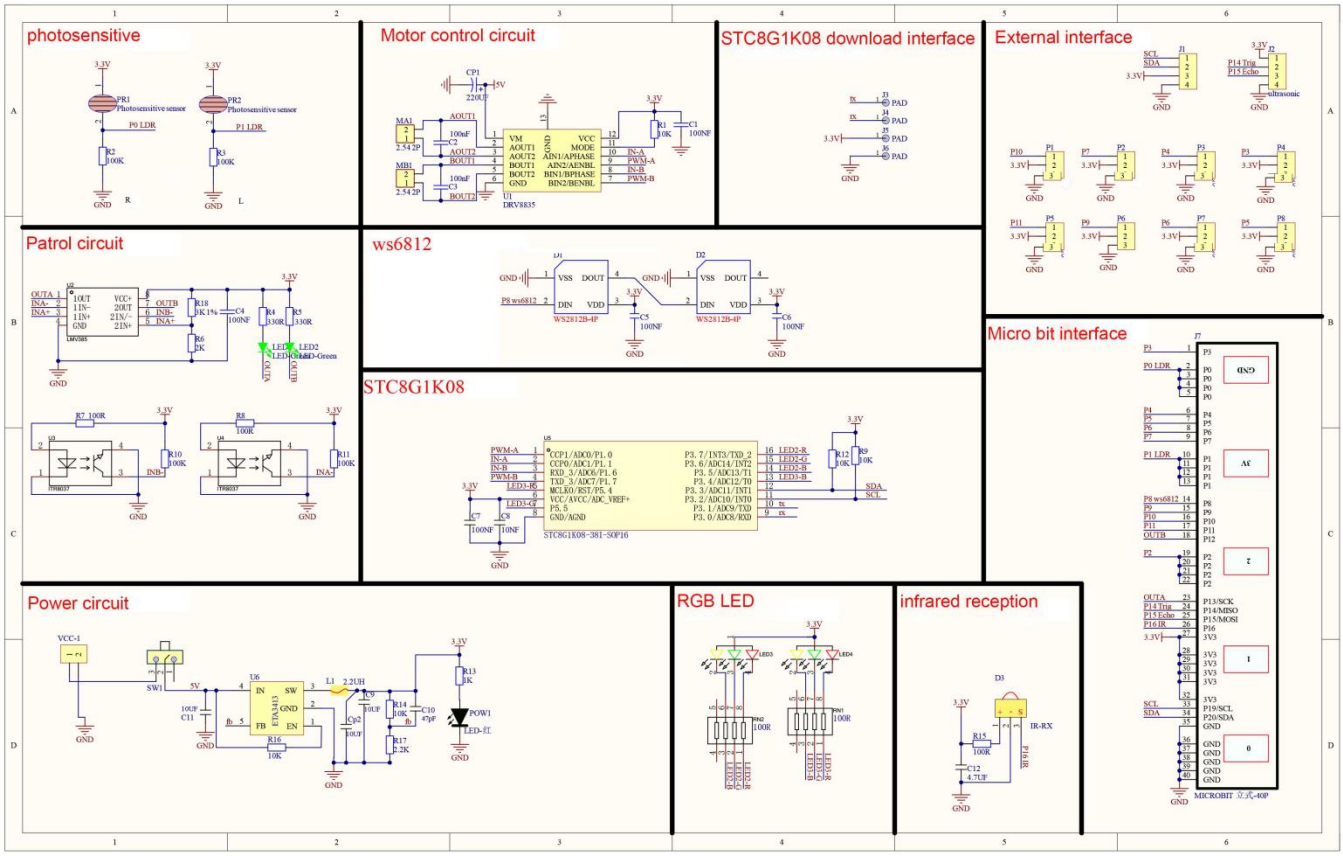
2.Parameters

- Connector port input: DC 5v
- Motor speed: 200RPM
- Operating temperature: 0-50℃
- Dimensions: 90*88*15mm
- Environmental attributes: ROHS

3.Function Diagram



4.Schematic Diagram

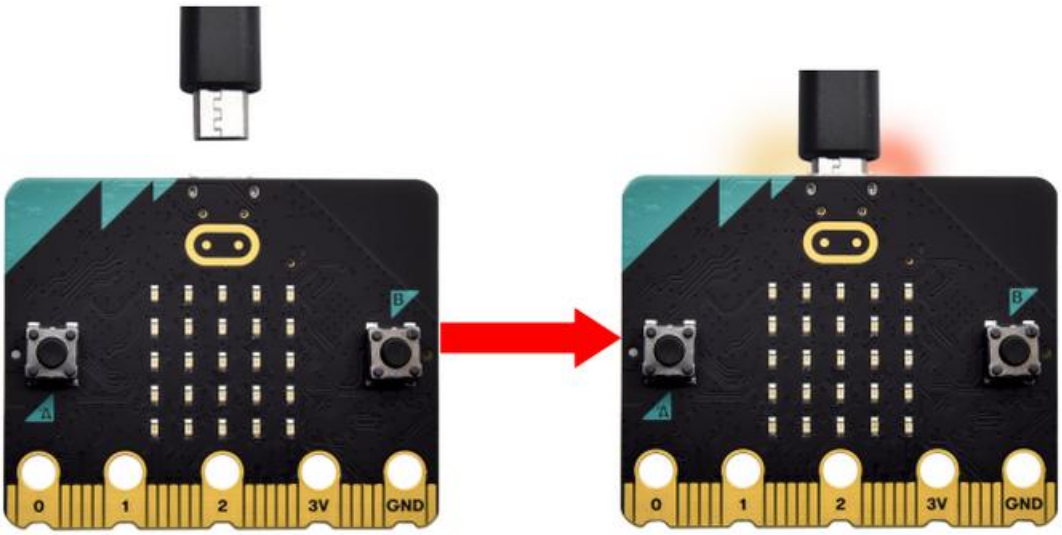


Micro:bit Driver installation instructions

Micro:bit is free of driver installation. However, in case your computer fail to recognize the main board, you can install the diver too.

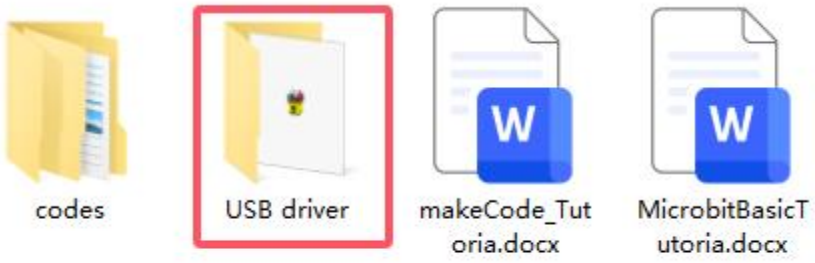
Driver installation: Please download tutorials in network disk.

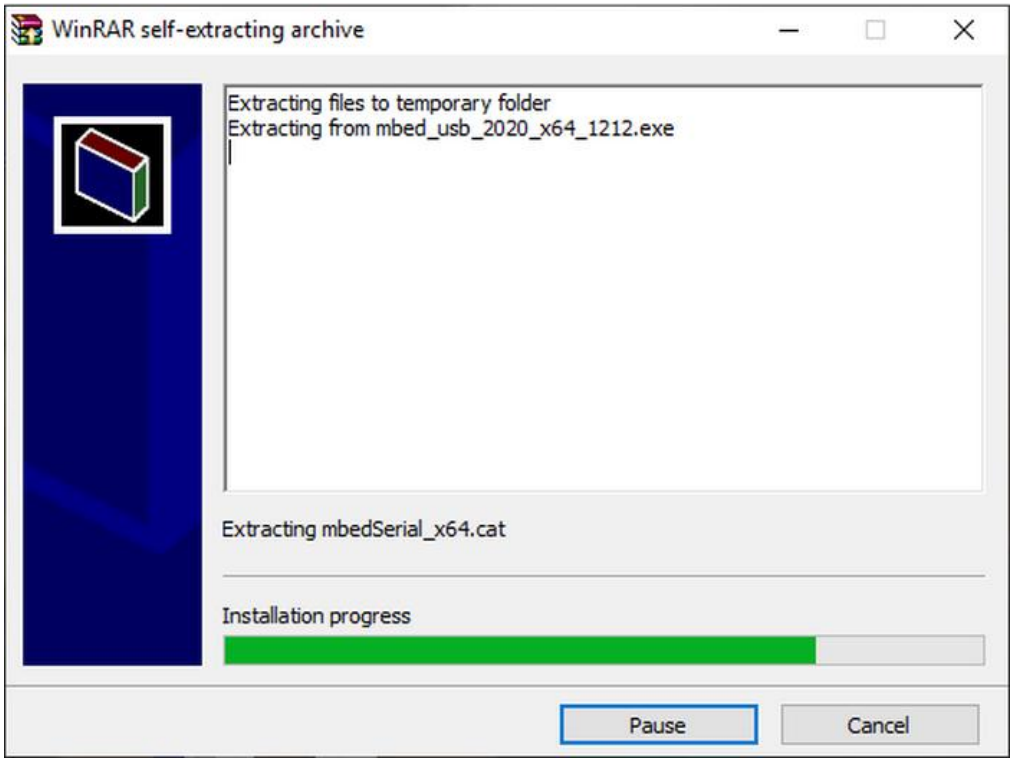
Connect micro:bit main board to computer via USB cable.



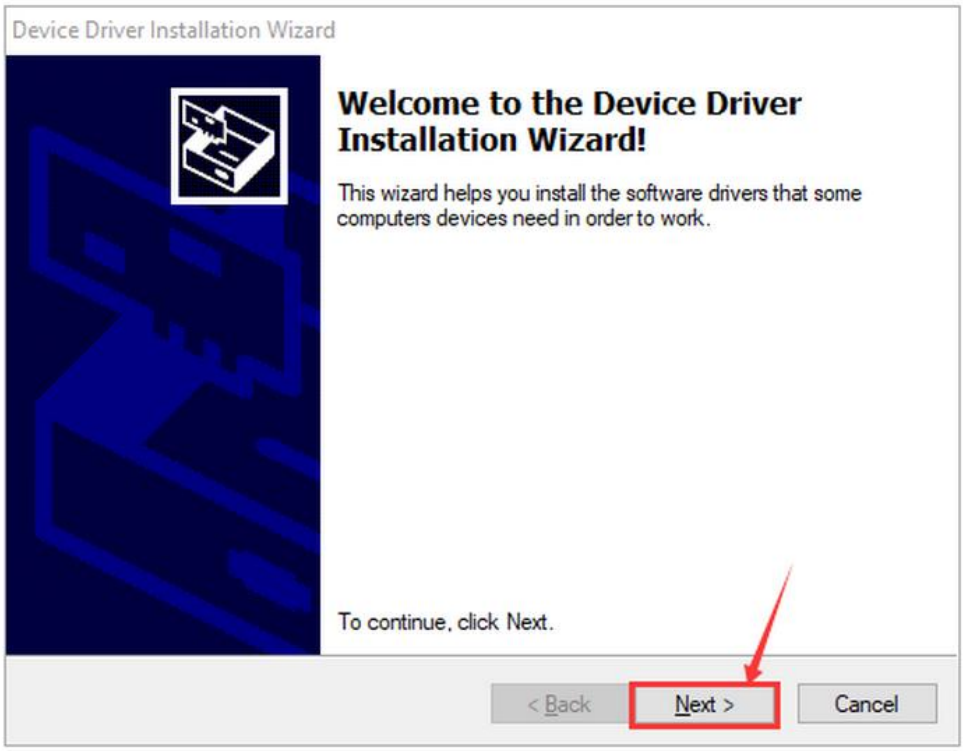
Click the driver file and "Install".

Driver file location:

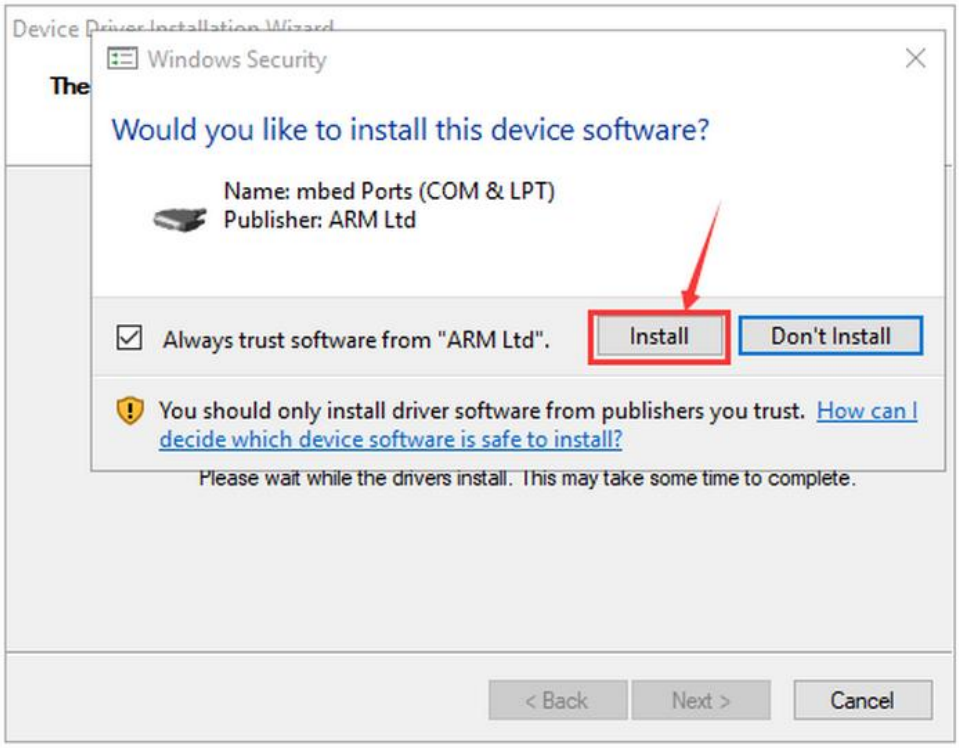


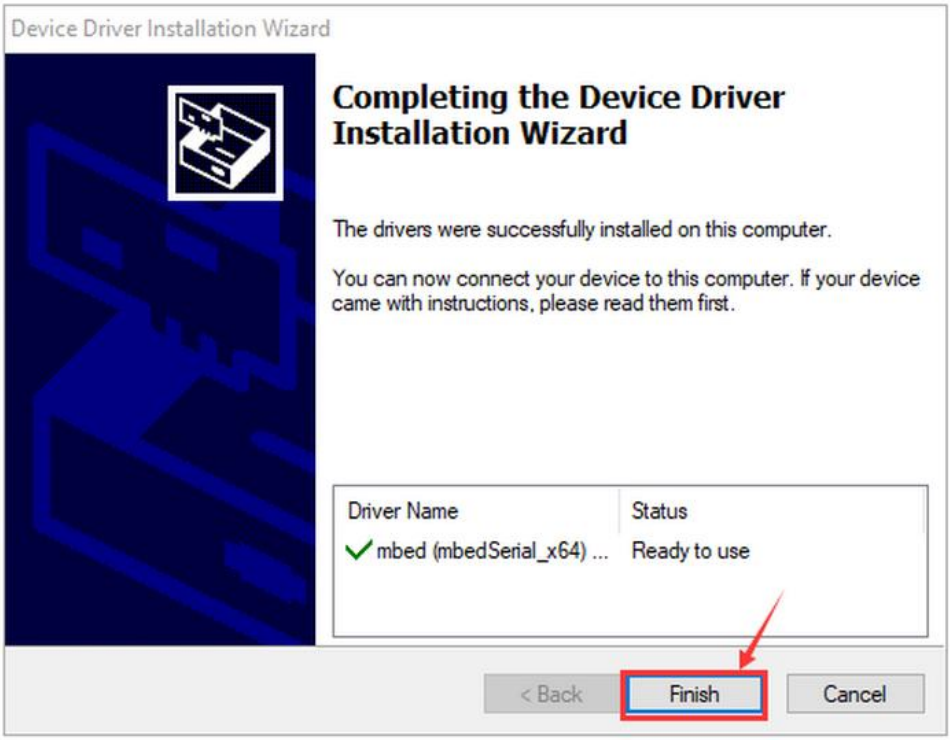


Click Install.

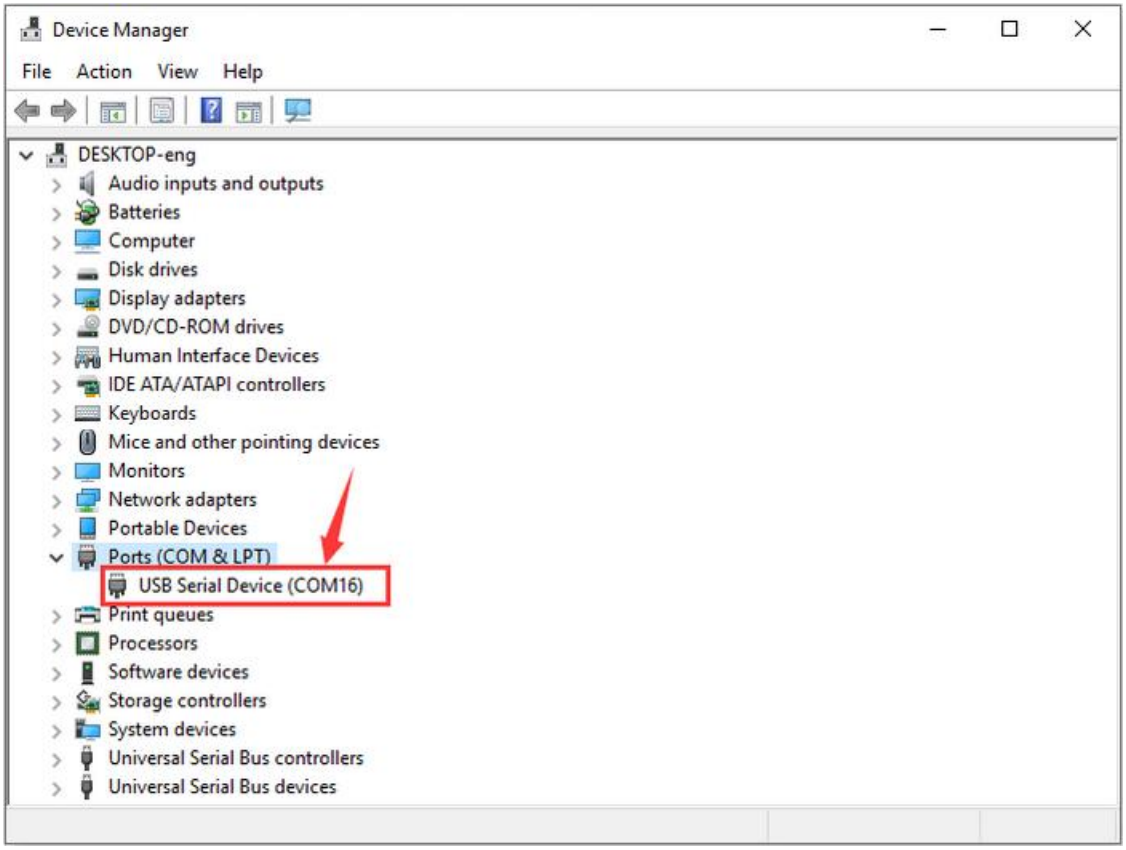


Click “Install” and “Finish”.





Click “Computer” —> “Properties” —> “Device manager”:



Programming

The following instructions are applied for Windows system but can also serve as a reference if you are using a different system.

1. Procedures

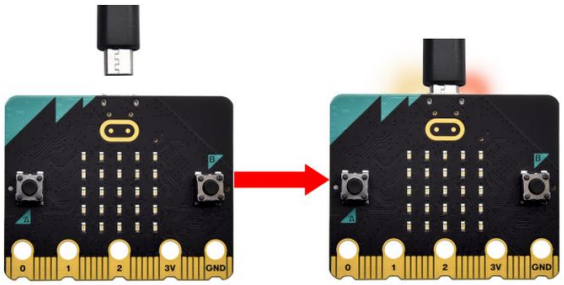
This chapter describes how to write program and load the program to the Micro: Bit mainboard. Visit official website for more details: https://microbit.org/guide/quick/

Step 1: Connect to Micro:bit

Connect the board to computer via USB cable.

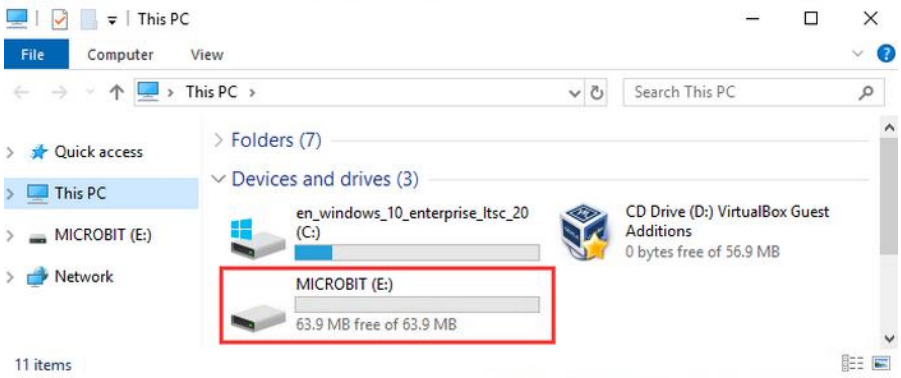
For how to program via mobile deceives: https://microbit.org/get-started/user-guide/mobile/

Multiple operation systems are compatible with this board, including Macs, PCs, Chromebooks and Linux (Raspberry Pi).



If the red LED on the back of the board is on, that means the board is powered. When your computer communicates with the main board via the USB cable, the yellow LED on it will flashes. For example, it will flash when you burn a “hex” file.

Then Micro: bit main board will appear on your computer as a driver named “MICROBIT(E:)”. Please note that it is not an ordinary USB disk as shown below.



Step 2: Write programs

Online version of Makecode: https://makecode.microbit.org/

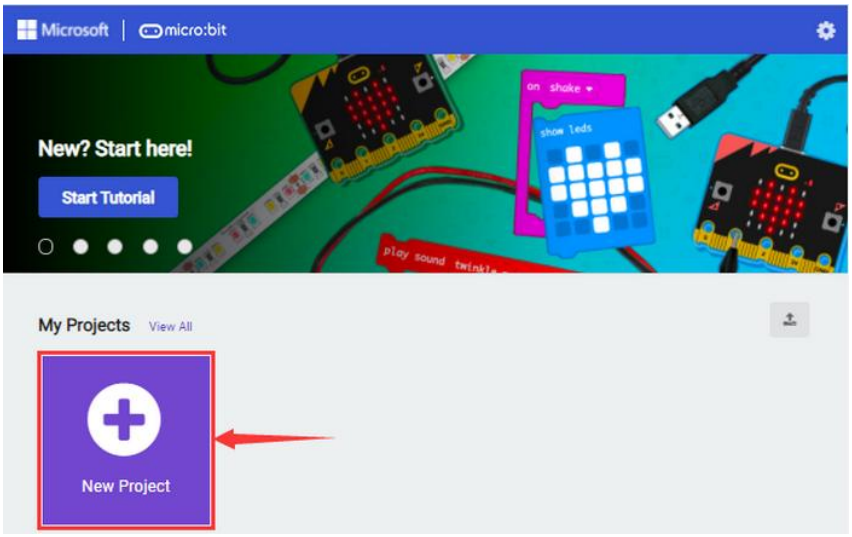
Click **New Project**;

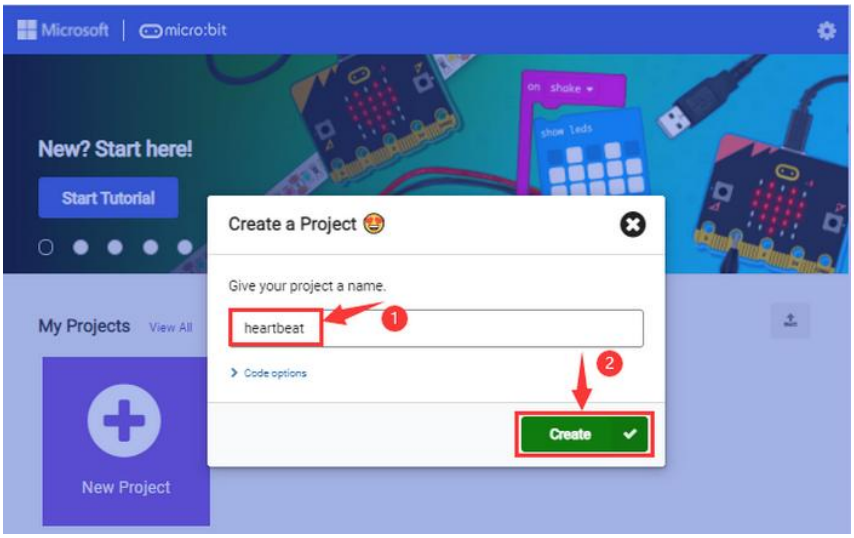
The dialog box **Create a Project** appears, fill it with **heartbeat** and click **Create** √ to edit.

If your computer has the Windows 10 operating system, you can also use MakeCode software for programming, which is exactly the same as programming on a browser.

Download: https://www.microsoft.com/

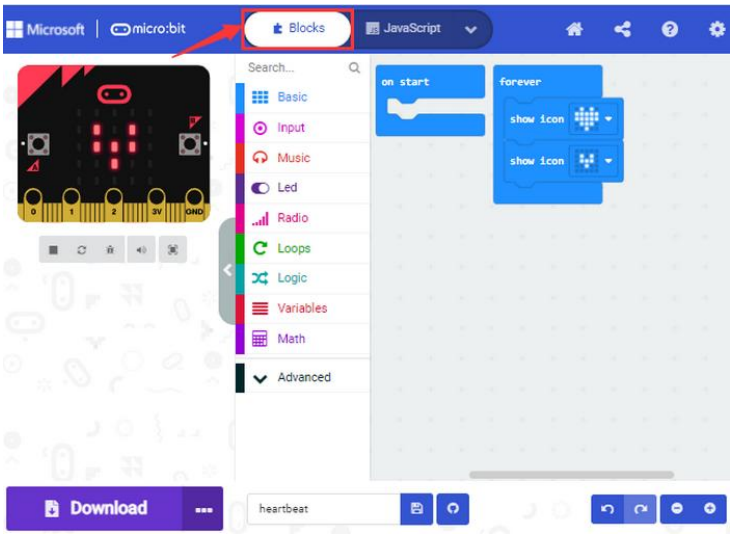
Here we demonstrate on Google Chrome.



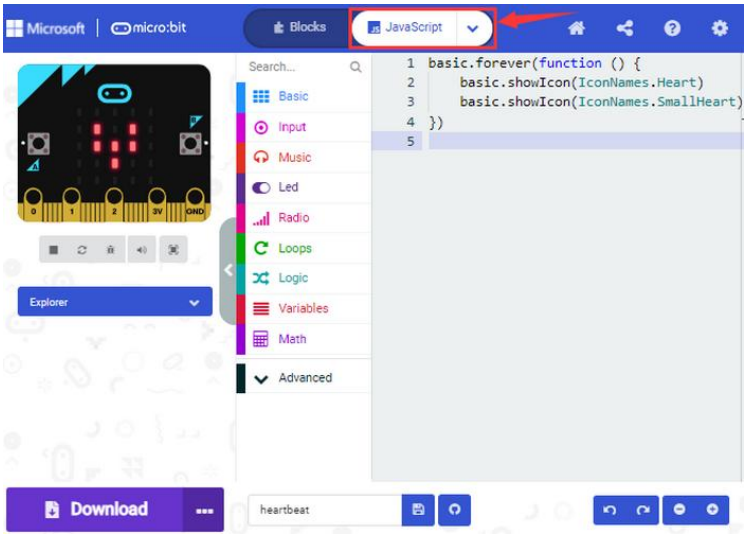


Write a micro:bit code.

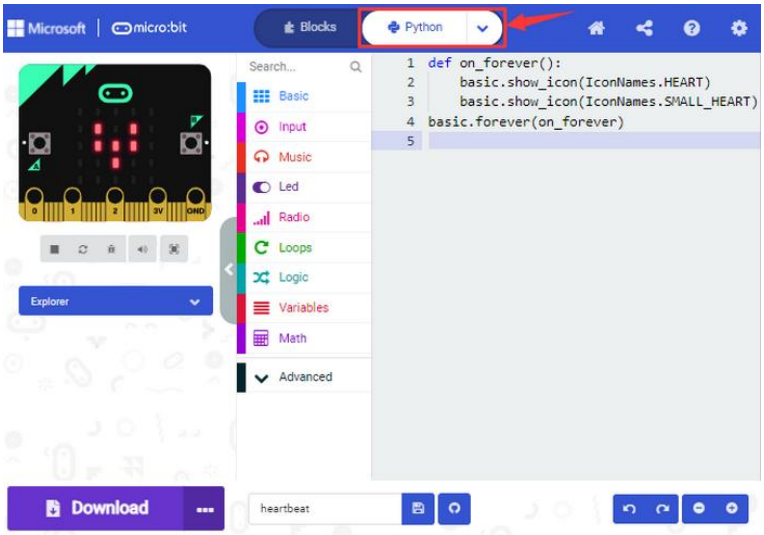
You can drag some Blocks to the editing area and then run your program in Simulator as shown below: we demonstrate how to edit **heartbeat** program.



Click " JS JavaScript" to check JavaScript language.



Click the arrow to switch to "Python" language.



Step 3: Download code

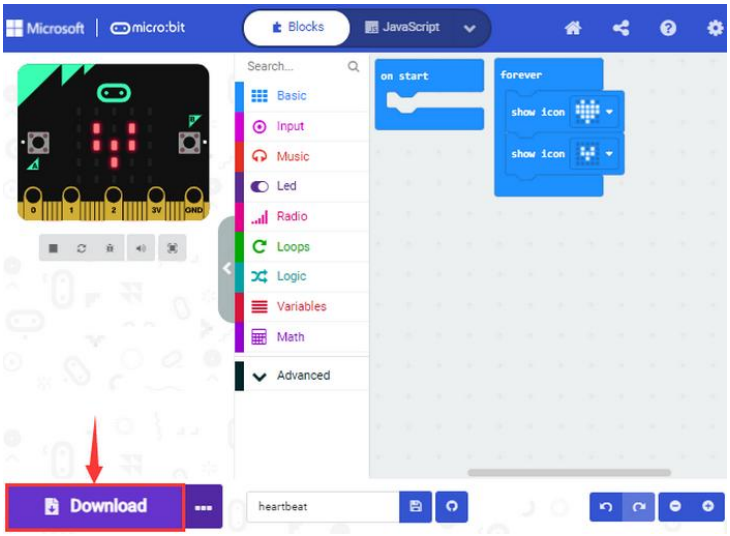
Generally, for systems with Windows 10 or higher using the MakeCode webpage to program, simply clicking the "Download" button will directly download the code program to the micro:bit board without any additional steps.

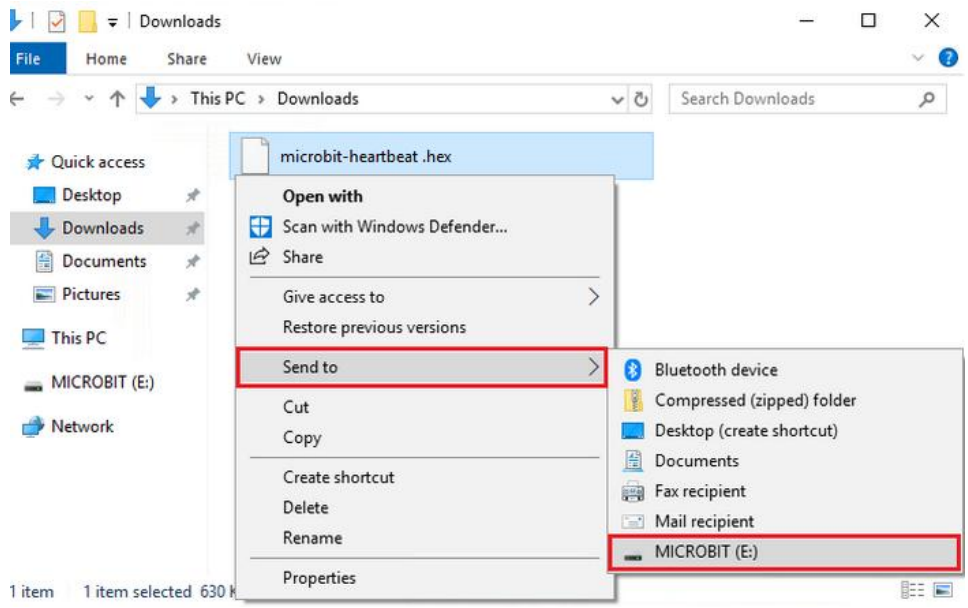
If you cannot directly download the code to the micro:bit board, please follow these steps:

1. Click the "Download" button in the editor. This will download a "hex" file, which is a format that the micro:bit board can read.

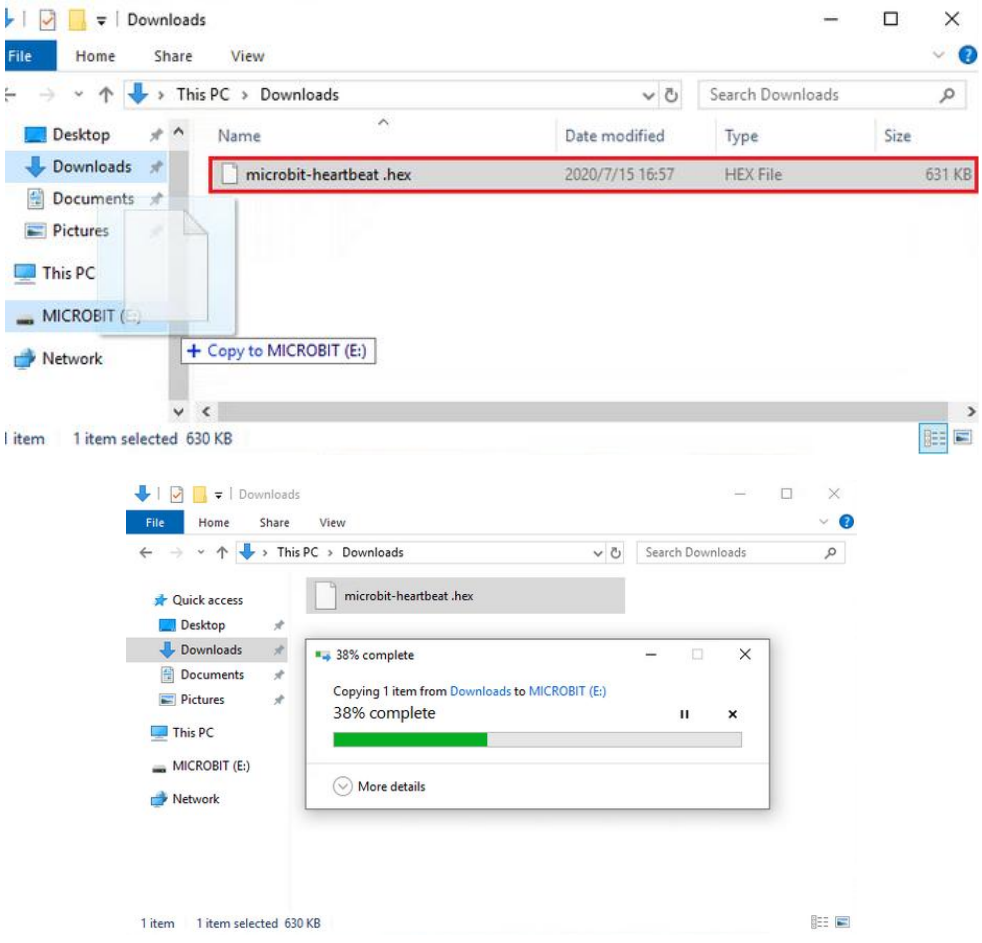
2. After the hexadecimal file is downloaded, copy it to your micro:bit board just like you would copy a file to a USB drive.

3. On Windows, you can also right-click on the "hex" file and select "Send to → MICROBIT" to copy the file to the micro:bit board.





Or, you may directly drag the “hex” file in **MICROBIT**.



During the process of copying the hex file to the Micro: bit, the yellow LED on the back of the board flashes. When the duplication is completed, the LED will stop flashing and remain on.

Step 4: Run program

After the program is uploaded to the Micro: bit, you can power it via USB cable or an external power. Then the 5 x 5 LED dot matrix displays a heartbeat pattern.

Power via USB

Power via external 3V

Caution:

When you programs each time, the driver of Micro: bit will automatically eject and return so the hex files will disappear. The board only has access to hex files rather than save them.

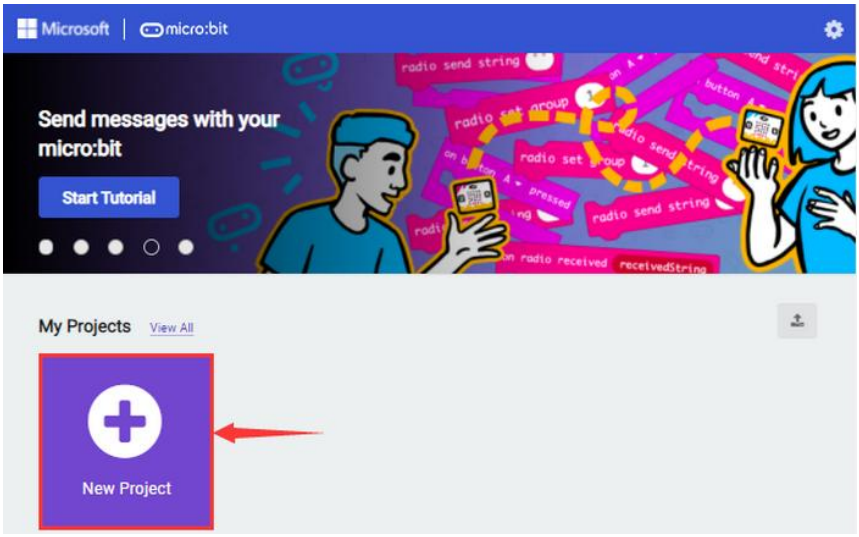
Step 5: Other programming languages

This chapter has described how to use the Micro:bit main board.

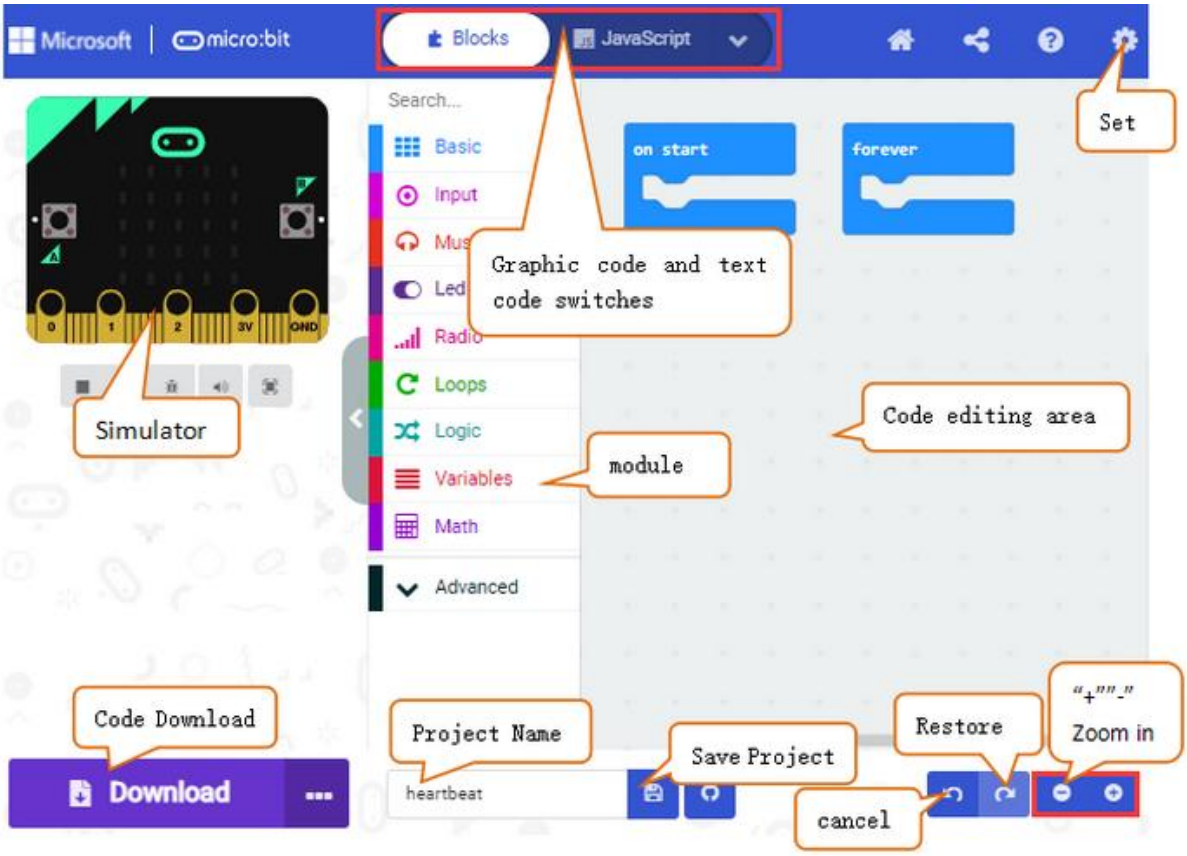
Except for the Makecode graphical programming, if you want to write Micro:bit programs in other languages, visit https://microbit.org/code/ to learn more, or visit Make it: code it | micro:bit (microbit.org) to find something you want to have a go.

2. Makecode

Google Chrome online version: https://makecode.microbit.org/



Click “New Project” and enter “heartbeat” to edit the code. Here is the main interface of Makecode.



There are blocks “on start” and “forever” in the code editing area.

When the power is plugged or reset, “on start” means that the code in the block only executes once, while “forever” implies that the code runs cyclically.

3. Quick Download

As mentioned before, if your computer is Windows 10 and you have downloaded the APP MakeCode. you can quickly download codes to the Micro: Bit main board by selecting ‘Download’.

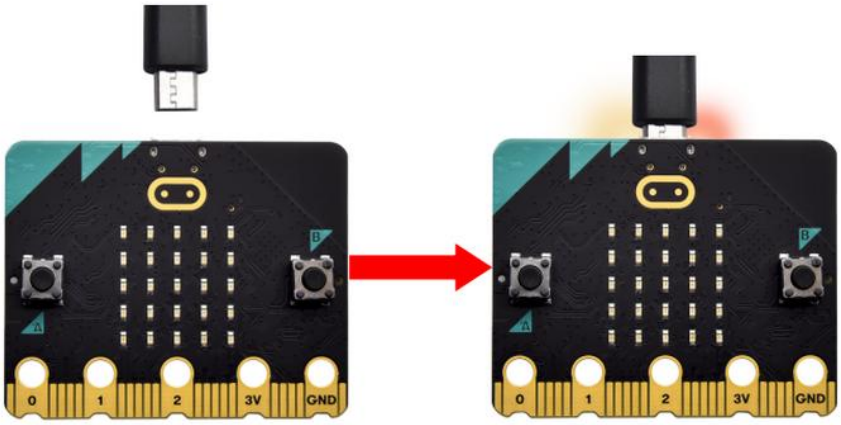
While it is a little more trickier if you are using a browser to enter Makecode. However, if you adopt Google Chrome that is suitable for Android, ChromeOS, Linux, macOS and Windows 10, the process can be easier, too.

We use the webUSB of Chrome to access the hardware device connected by USB.

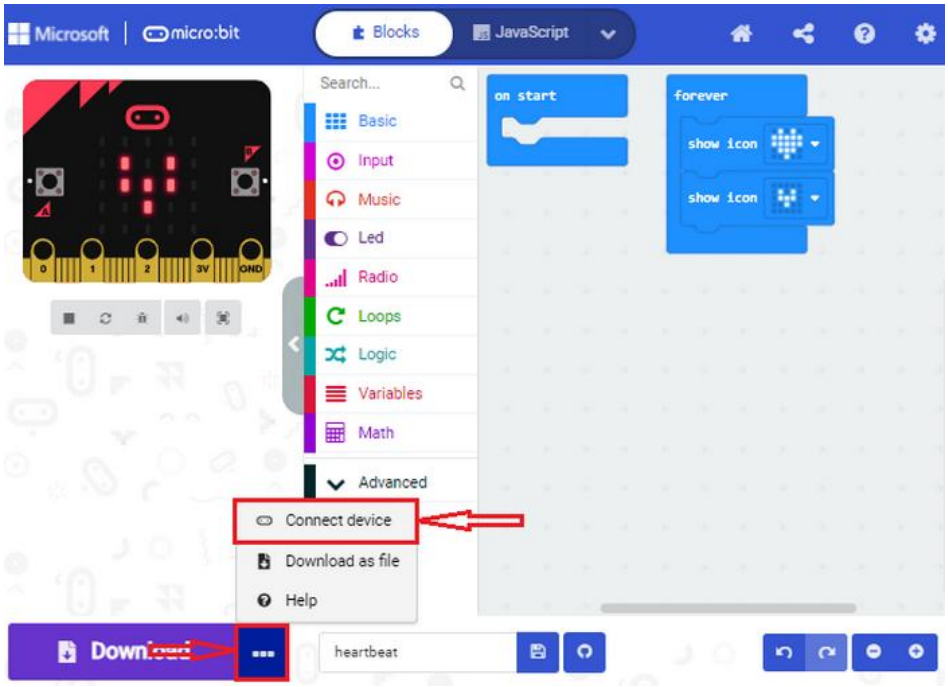
You could refer to the following steps to connect and pair devices.

Devices Pairing

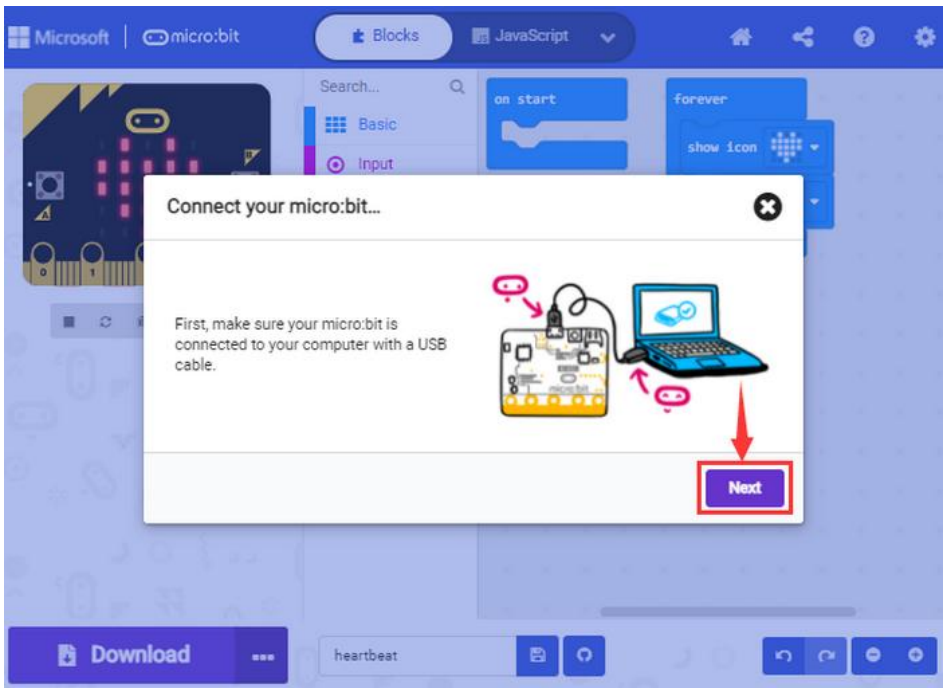
Connect the board to computer via USB cable.



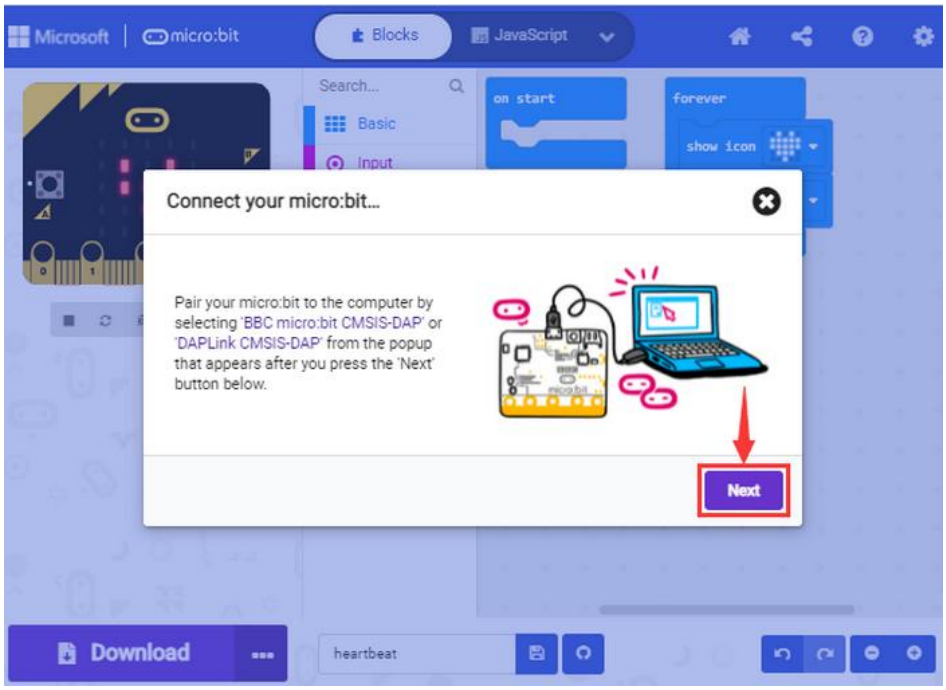
Click “...” and “Connect device”.


Microsoft
micro:bit
Blocks
JavaScript
Search...
Basic
Input
Music
Led
Radio
Loops
Logic
Variables
Math
Advanced
Connect device
Download as file
Help
on start
forever
show icon
show icon
heartbeat

“Next”.


Microsoft
micro:bit
Blocks
JavaScript
Search...
Basic
Input
on start
forever
show icon
Connect your micro:bit...
First, make sure your micro:bit is connected to your computer with a USB cable.
Next
Download
heartbeat

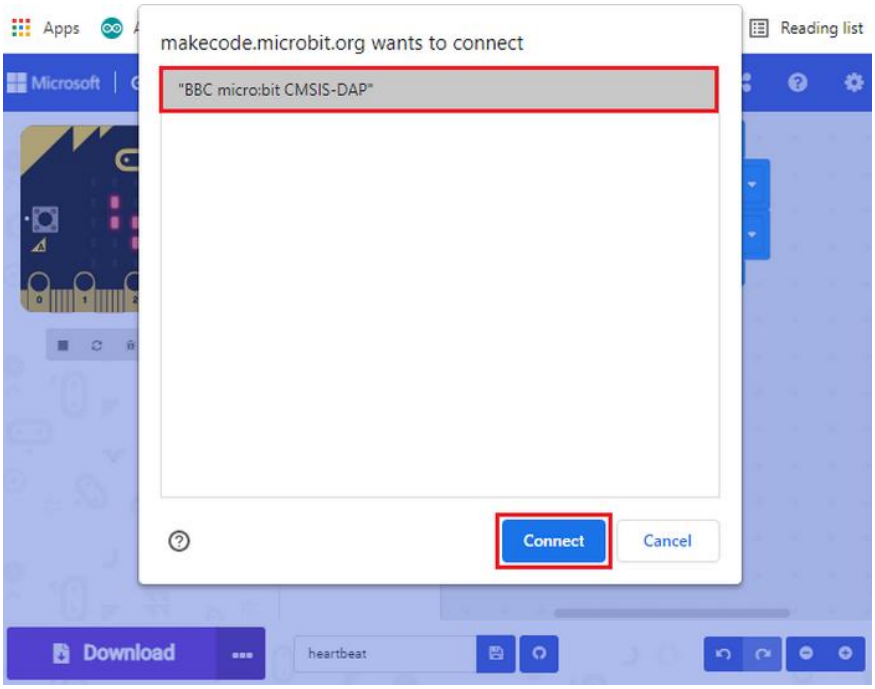
“Next”.



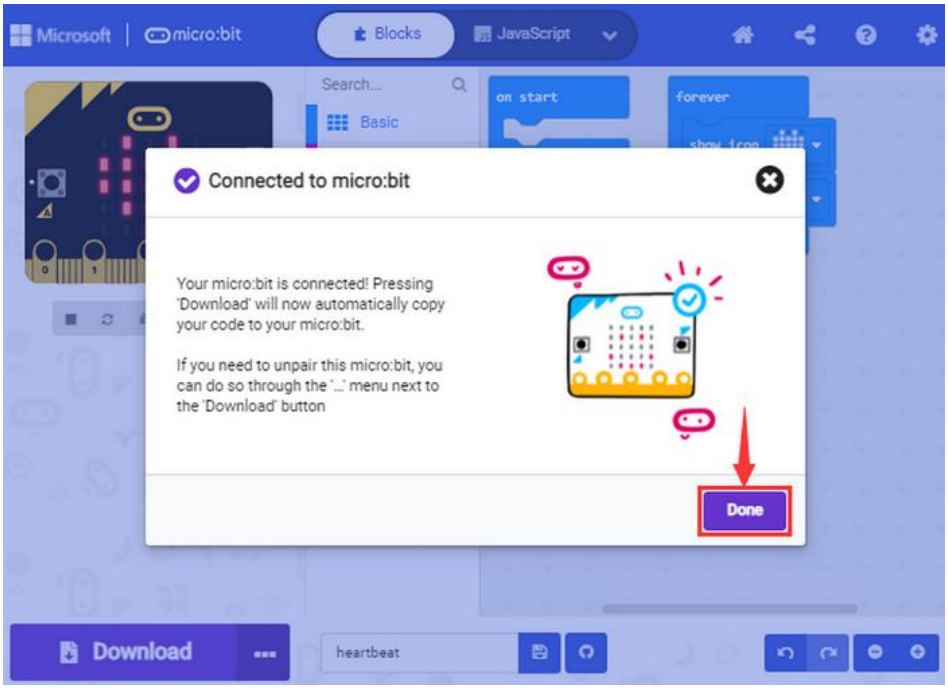
Then select the corresponding device and click “Connect”. If no device shows up for selection, please refer to: https://makecode.microbit.org/device/usb/webusb/troubleshoot

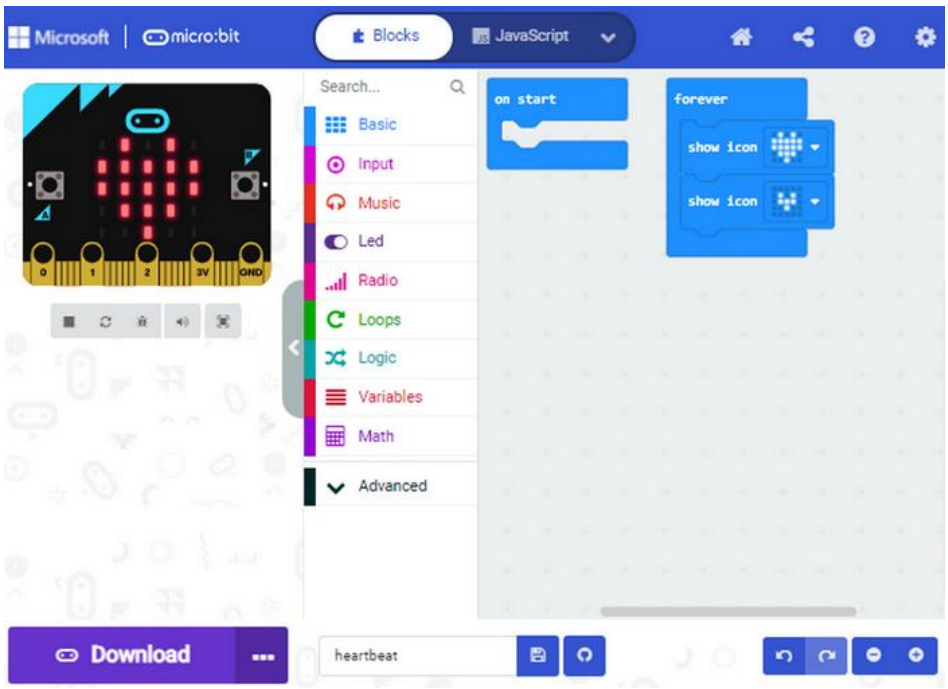
If the links are too troublesome, we also provide **Troubleshooting** in tutorial.

For how to update micro:bit firmware: https://microbit.org/guide/firmware/



Click “Done”.



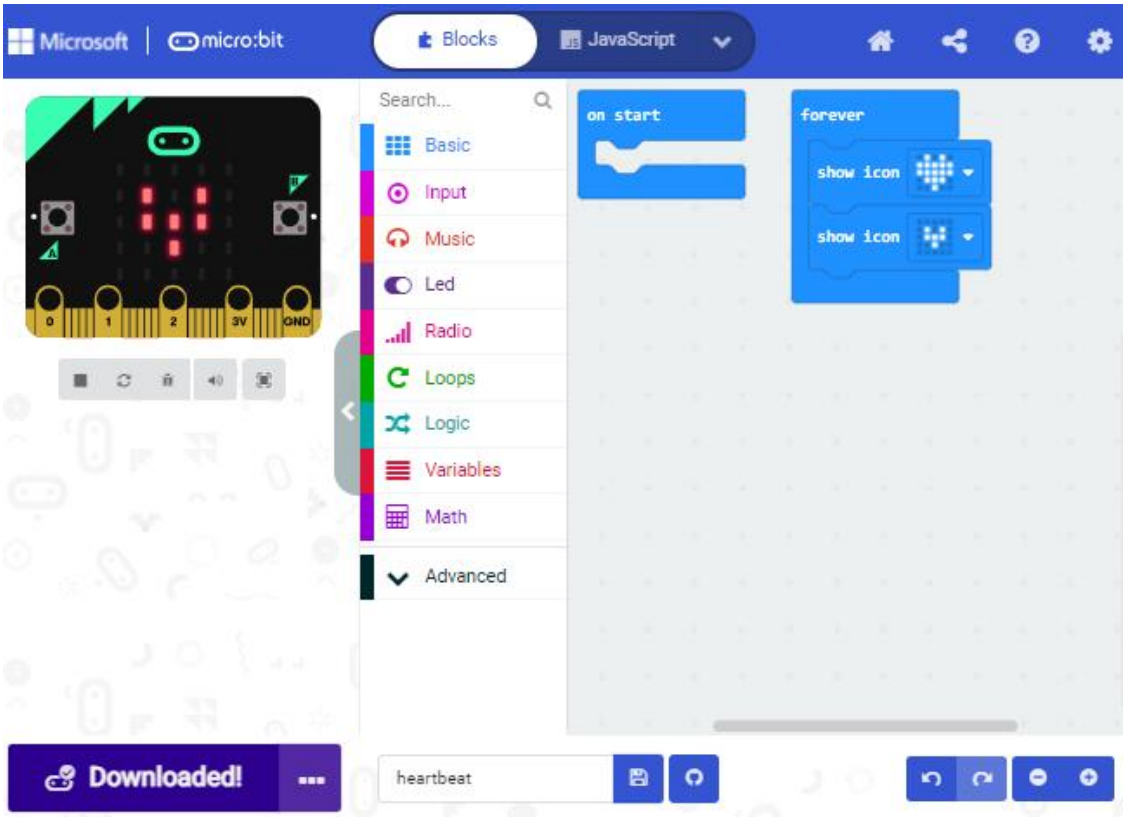


Download program

After connection, click 

and it will change into 

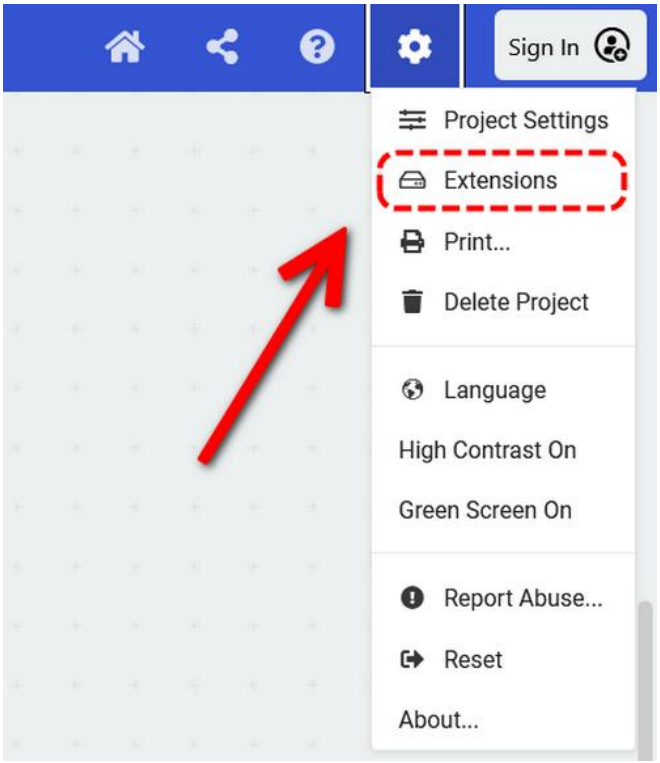
.



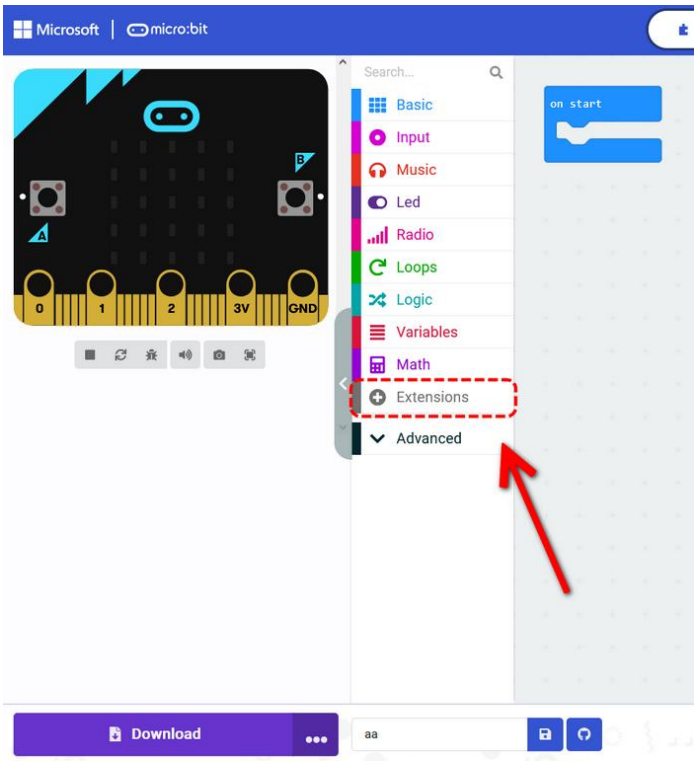
4. Makecode Extension Library

Please follow the steps to add extension files:

Open makecode to enter a certain project, click the gear-shaped icon(settings) in the upper right corner to choose “Extensions”.



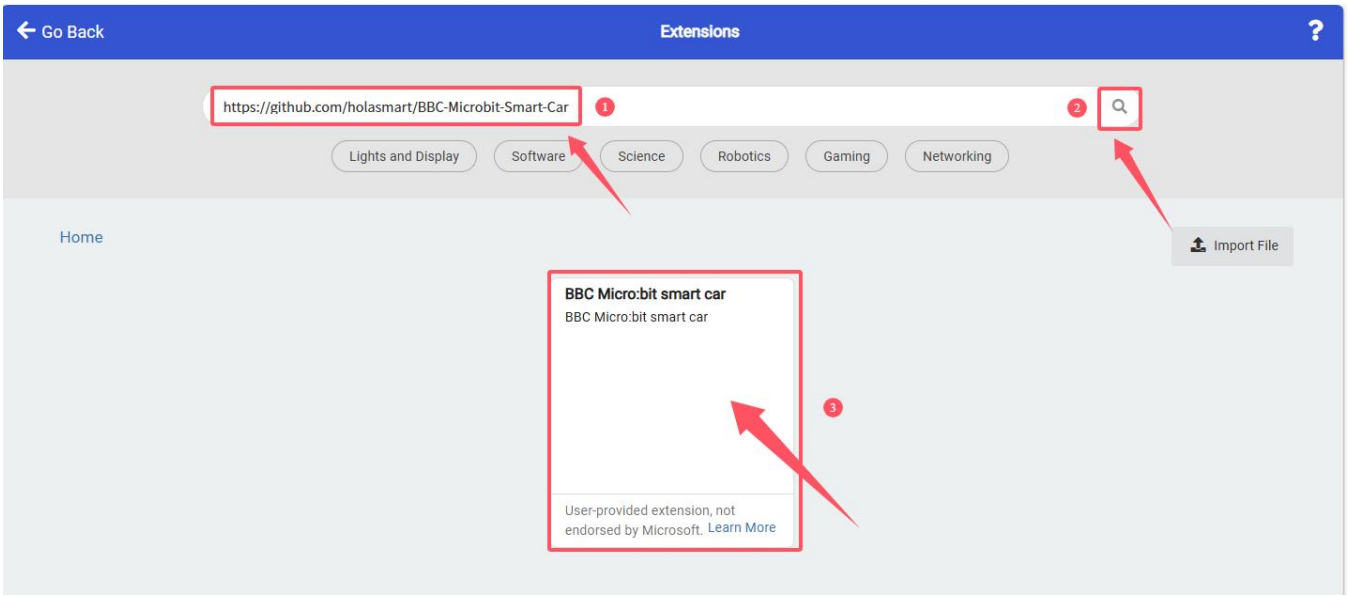
Or click Advanced to add Extensions.



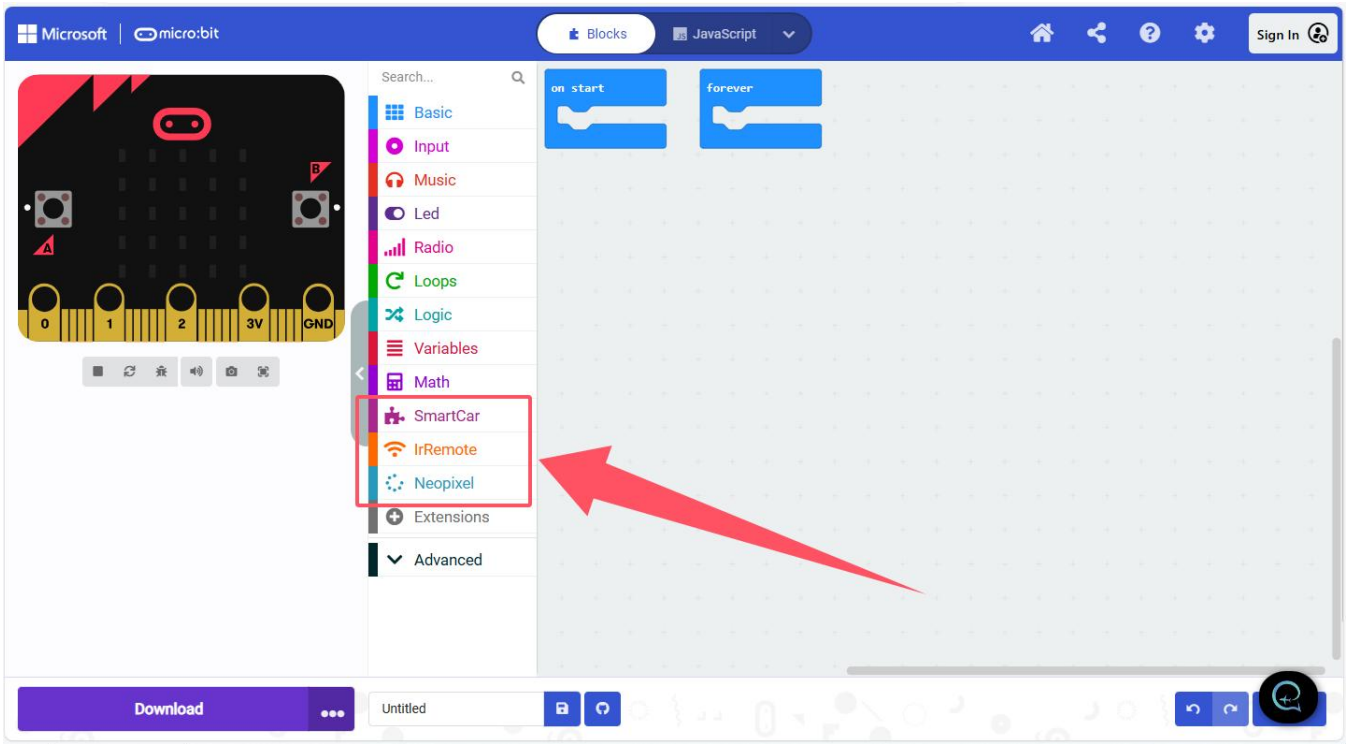
Search the library you want.

Copy and paste the link in the search box: https://github.com/holasmart/BBC-Microbit-Smart-Car

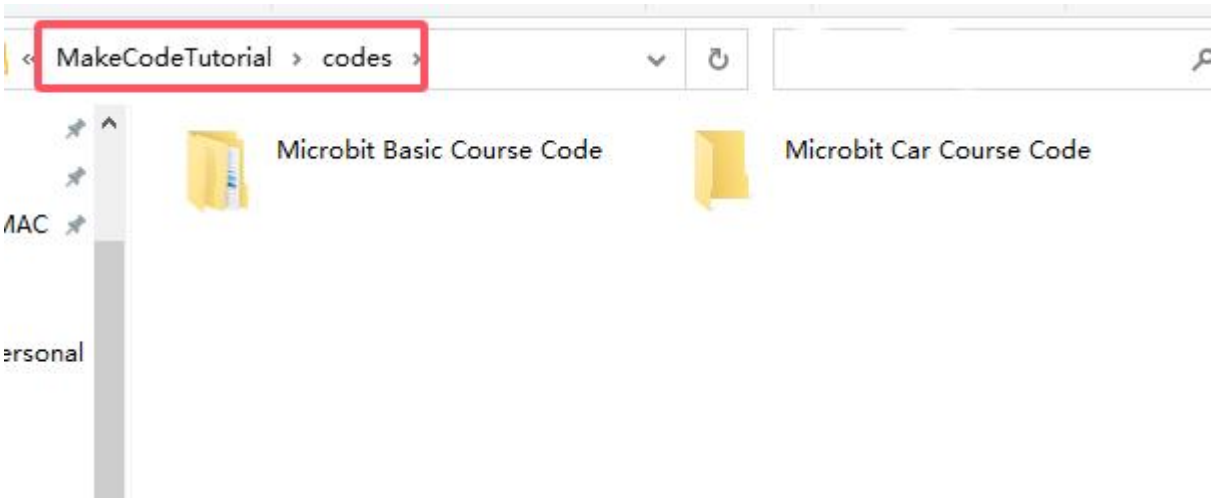
Load the extension and you can see the blocks. If you directly load the code file we provided, you do not have to import the extension again.



Successfully imported:



5.Resources and Test Code



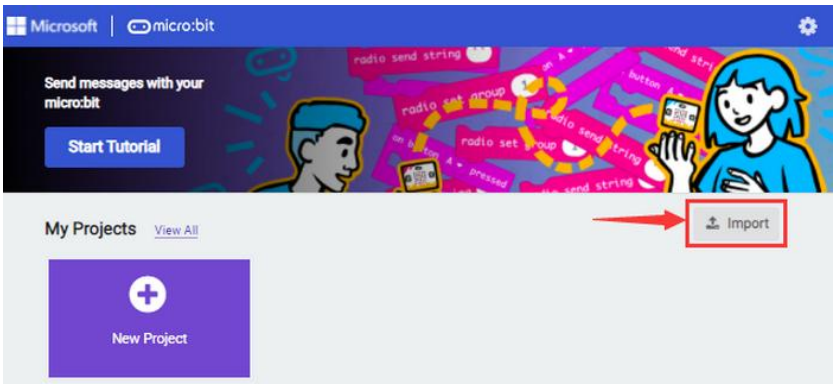
1. Import code

We provide hexadecimal code files (project files) for each project. The file contains all the contents of the project and can be imported directly, or you can manually build the code blocks.

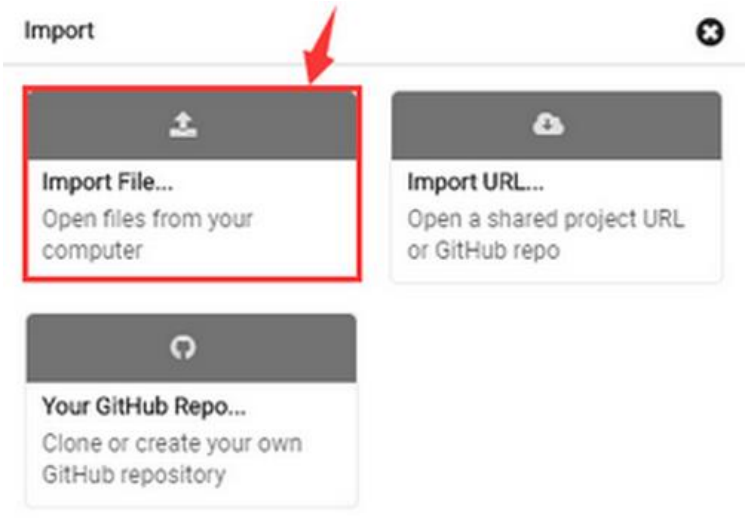
For simple projects, dragging a block of code to complete the program is recommended.

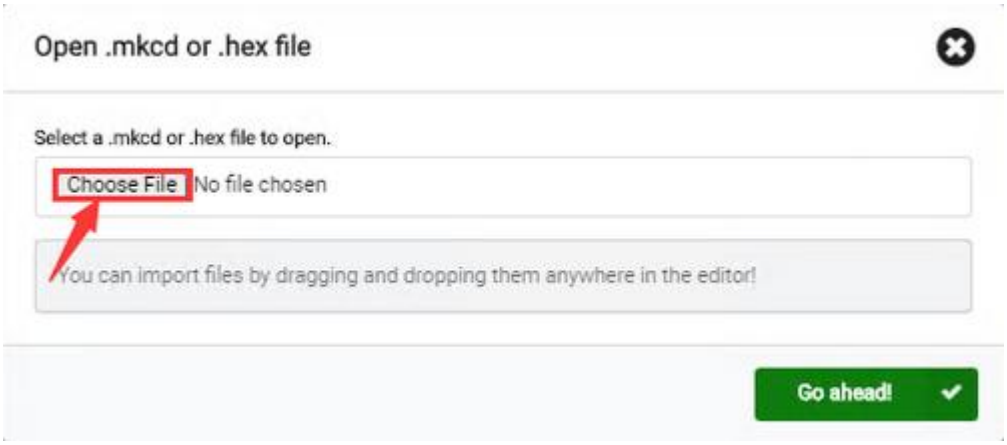
For complex ones, it is recommended to conduct the program by loading the hex code files.

Let's take the "Heartbeat" project as an example to show how to load the code. Open the Web version of Makecode or the Windows 10 App Makecode, and click “Import”.

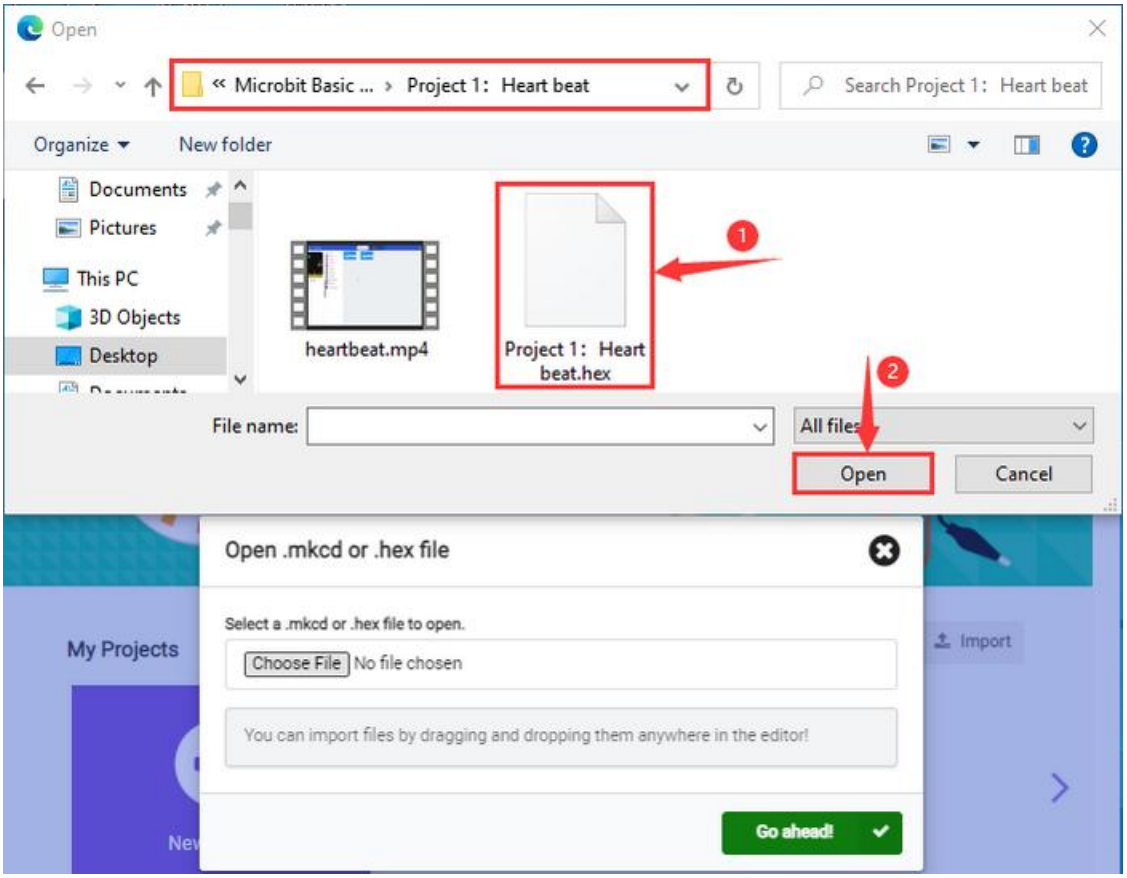


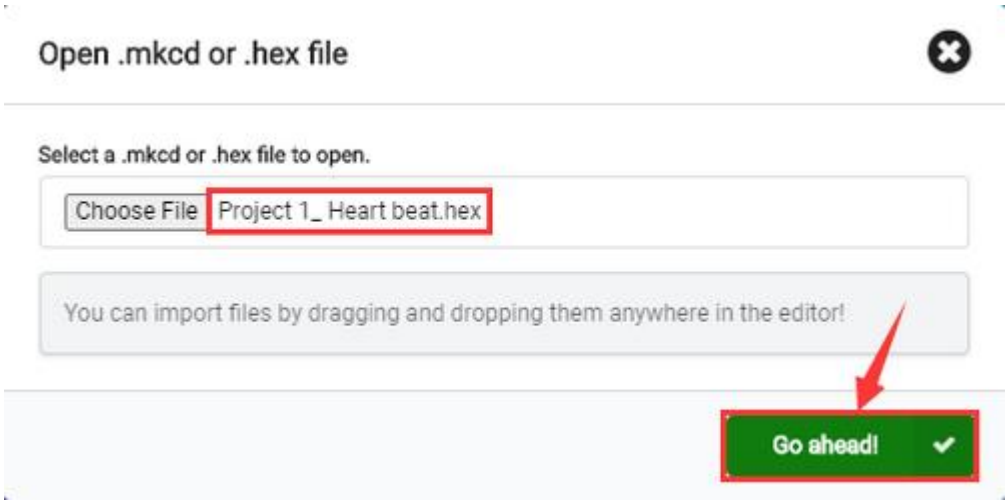
Click “Import File...”.



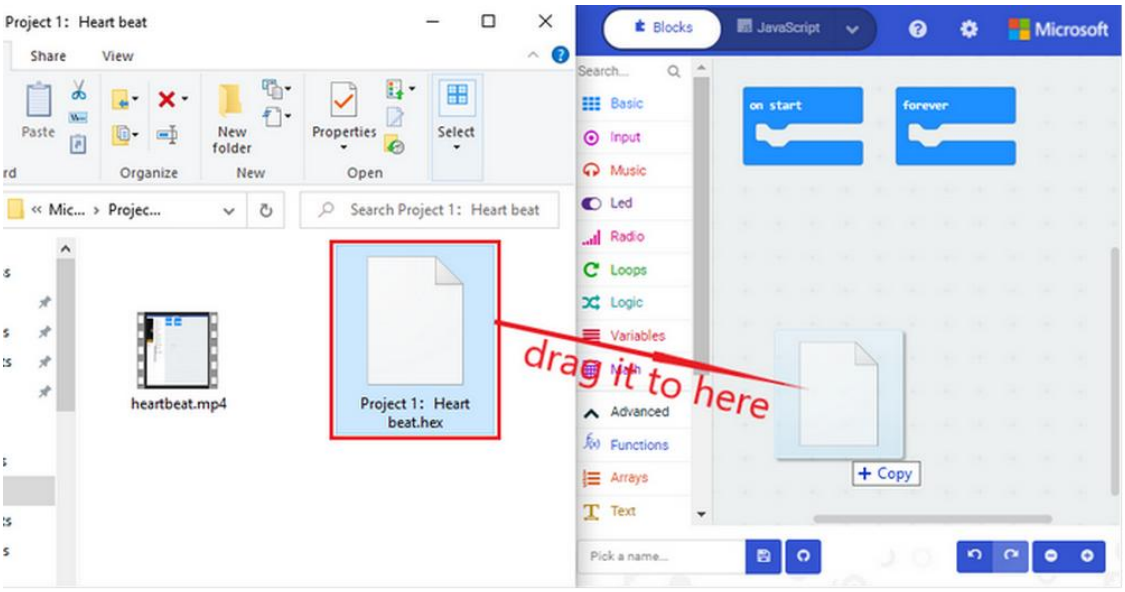


Choose “Heart beat.hex”.

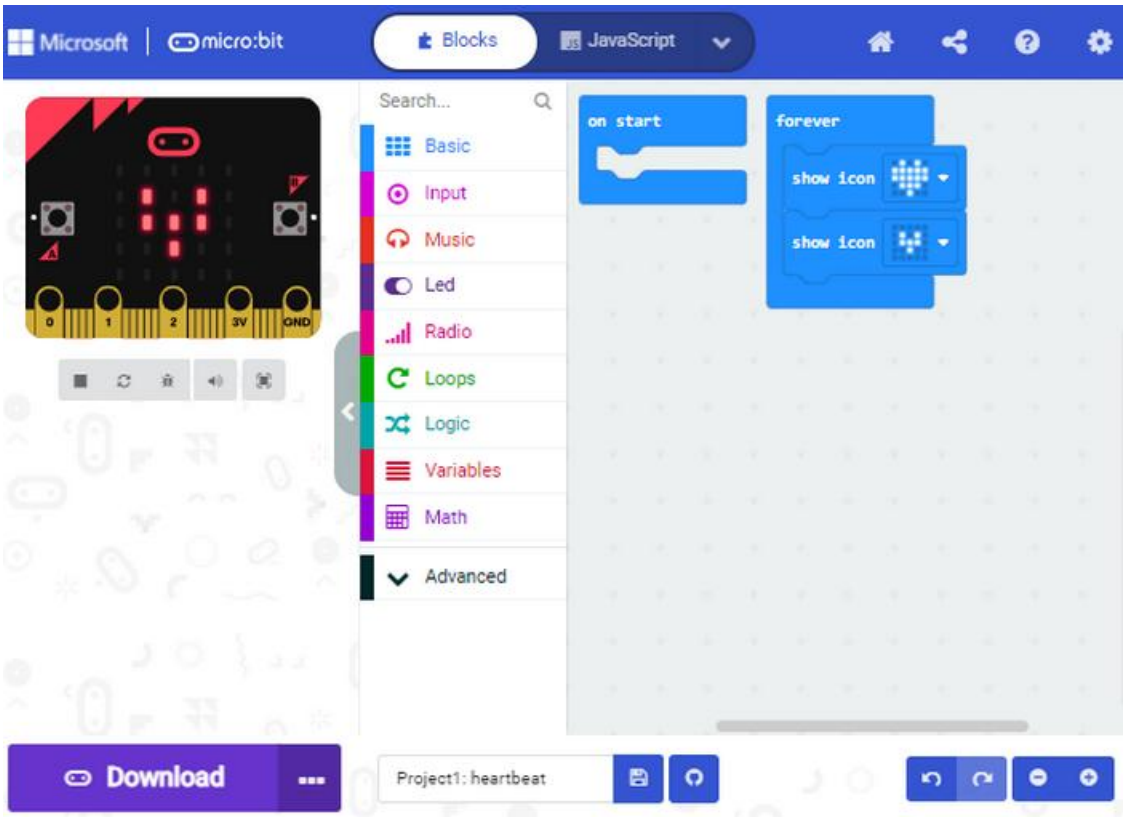




In addition to the above method, you can also drag the the test code into the code editing area, as shown below:



Wait for loading.

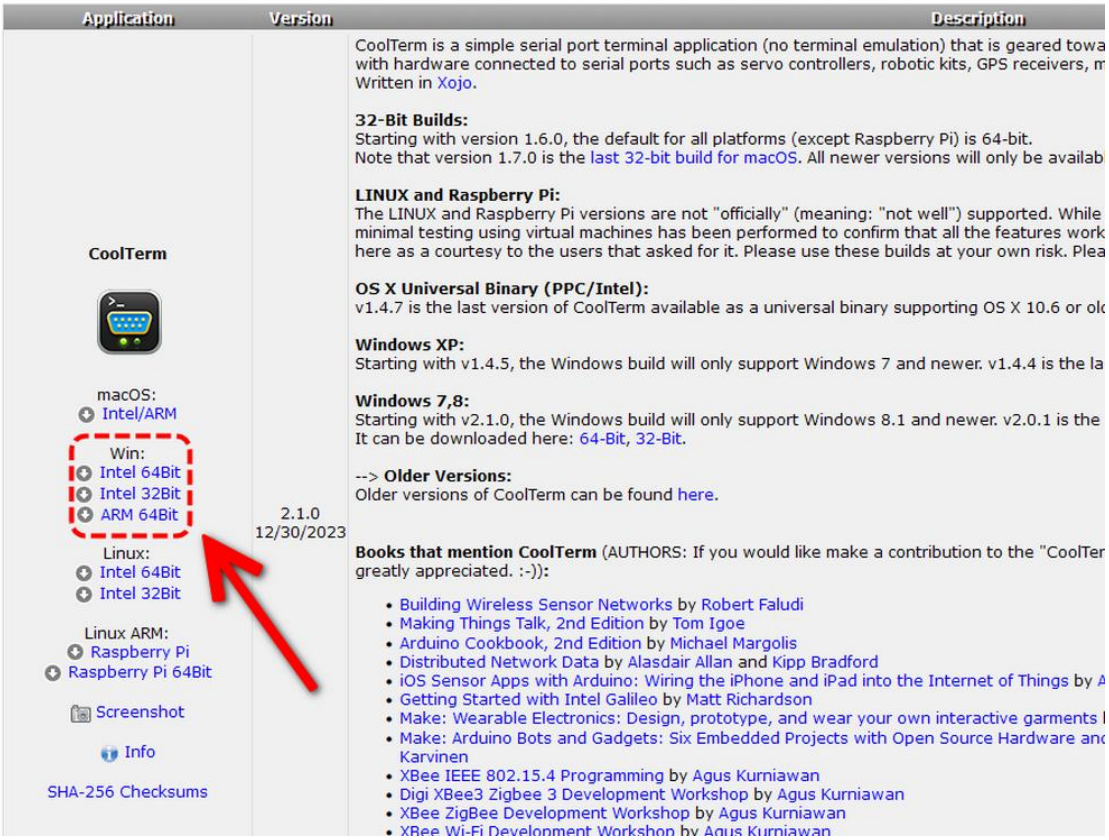


If your computer is Win7/8, the pairing cannot be done via Google Chrome. Therefore, digital signal or analog signal of sensors and modules cannot be shown on the serial simulator. So what can we do? CoolTerm software is a nice choice to read the serial data.

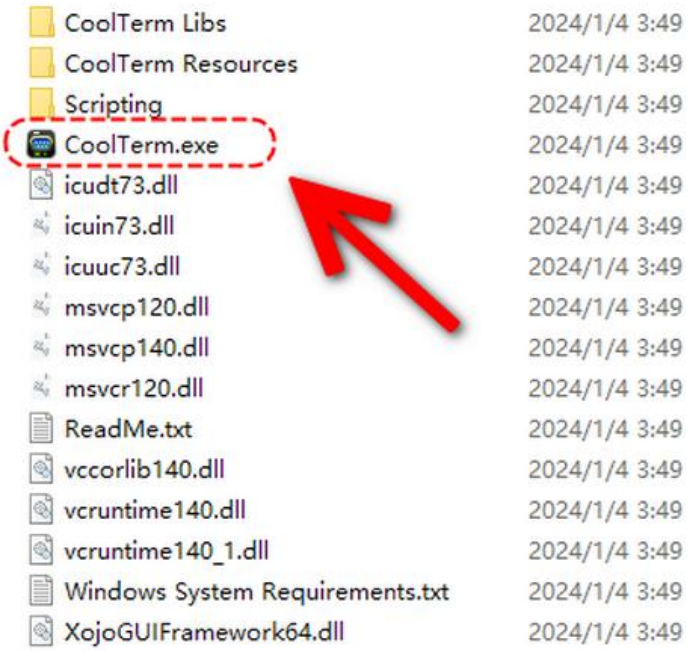
2. Install CoolTerm

CoolTerm download: https://freeware.the-meiers.org/

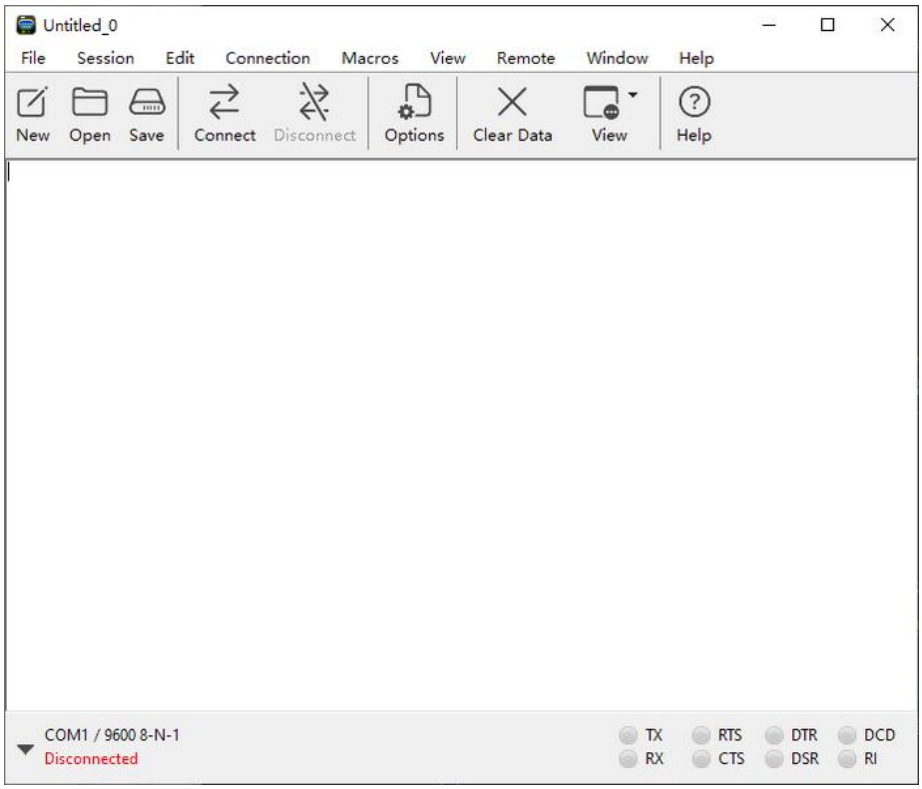
1.We take PC Windows as an example to download and unzip CoolTerm Win, and Mac/Linux can take it as a reference.

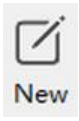


2.Make sure the driver is connect to computer, and click to execute the software.



3.The functions of each button on the Toolbar are listed below:

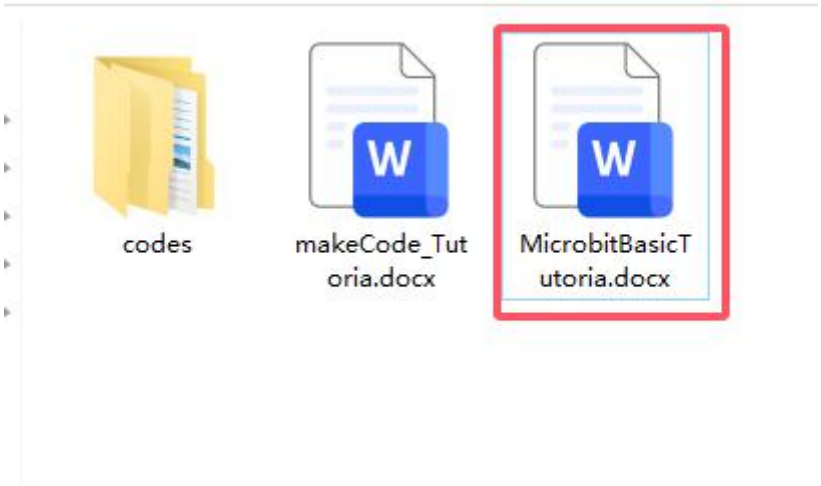


Icon	Function
	Opens up a new Terminal
	Opens a saved Connection
	Saves the current Connection to disk
	Opens the Serial Connection
	Closes the Serial Connection
	Clears the Received Data
	Opens the Connection Options Dialog
	Displays the Terminal Data in Hexadecimal Format
	Displays the Help Window

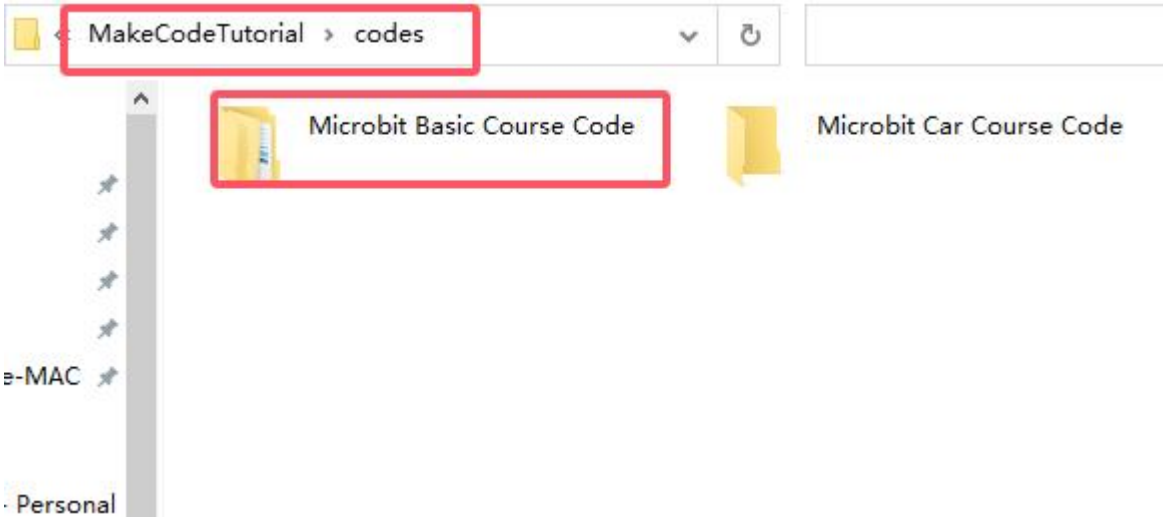
Micro:bit MakeCode Basic Tutorial

Attention: If you are a beginner, it is recommended to first learn the basic tutorial of Mciro: bit MakeCode. The basic tutorial includes some usage tutorials for sensors/modules on Micro: bit boards, which can help you better understand Micor: bit development boards.

Basic tutorial file location:

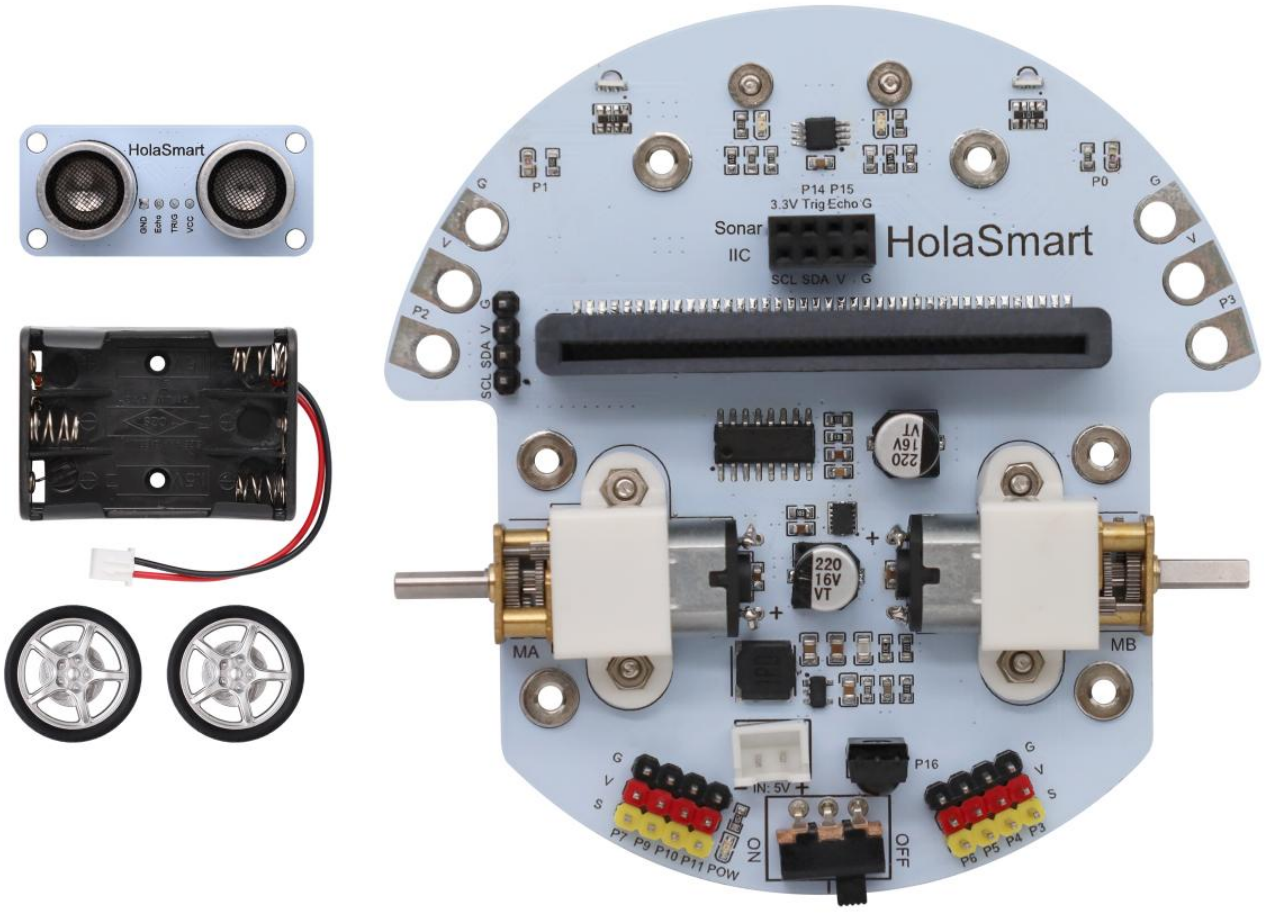


Location of Basic Tutorial Code Files:

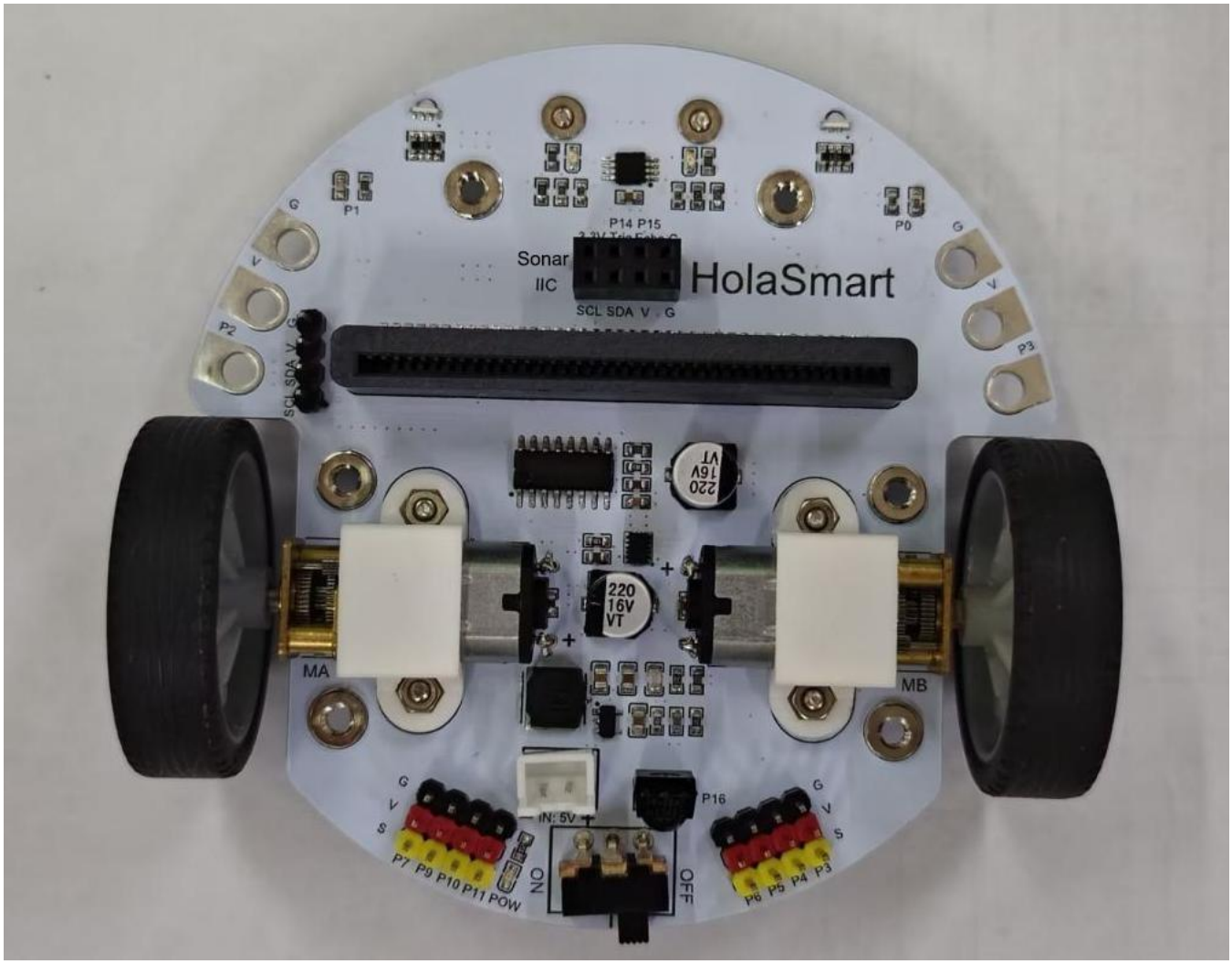


Assembly

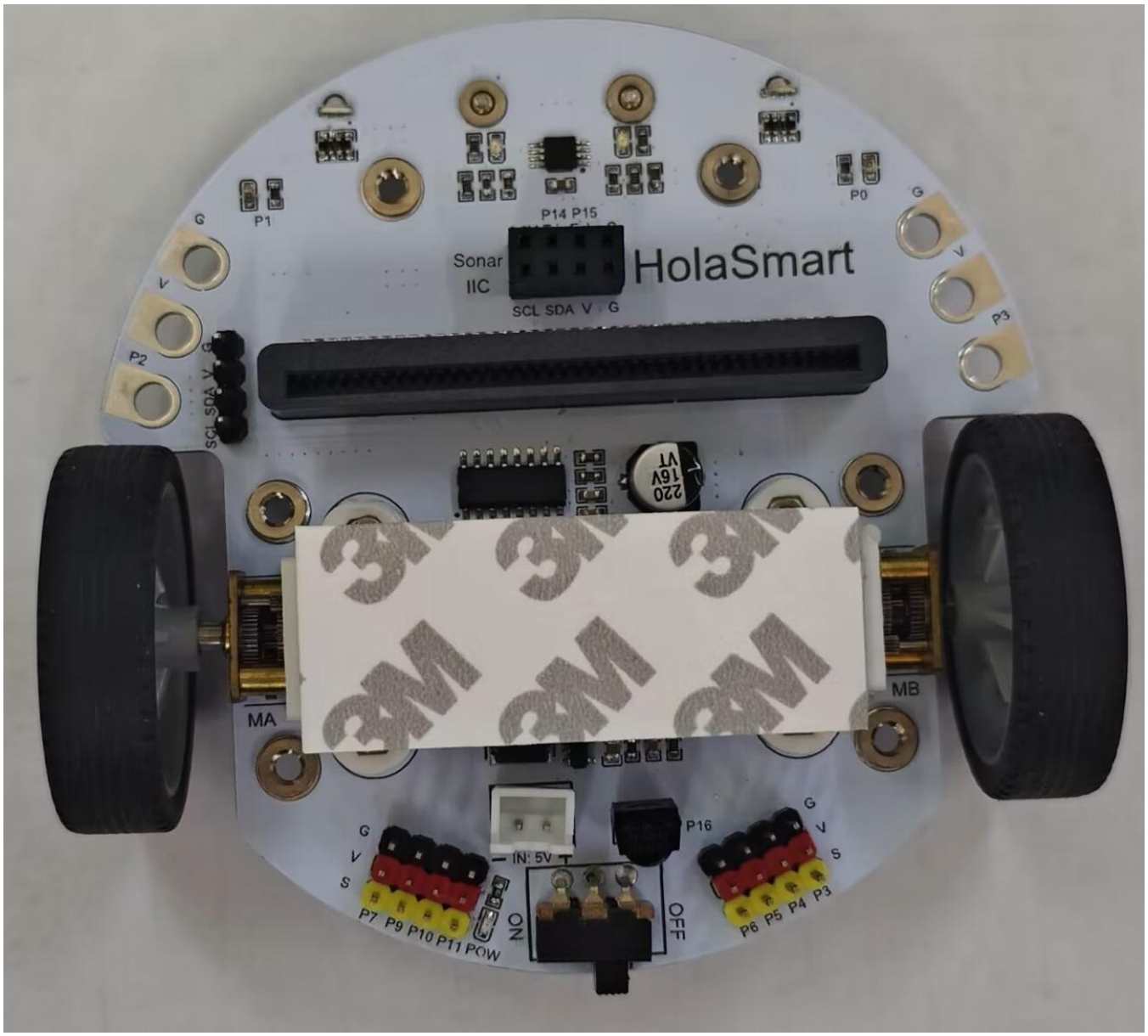
1. Remove the ultrasonic sensor, battery holder, wheels, double-sided tape, and car drive board from the package.



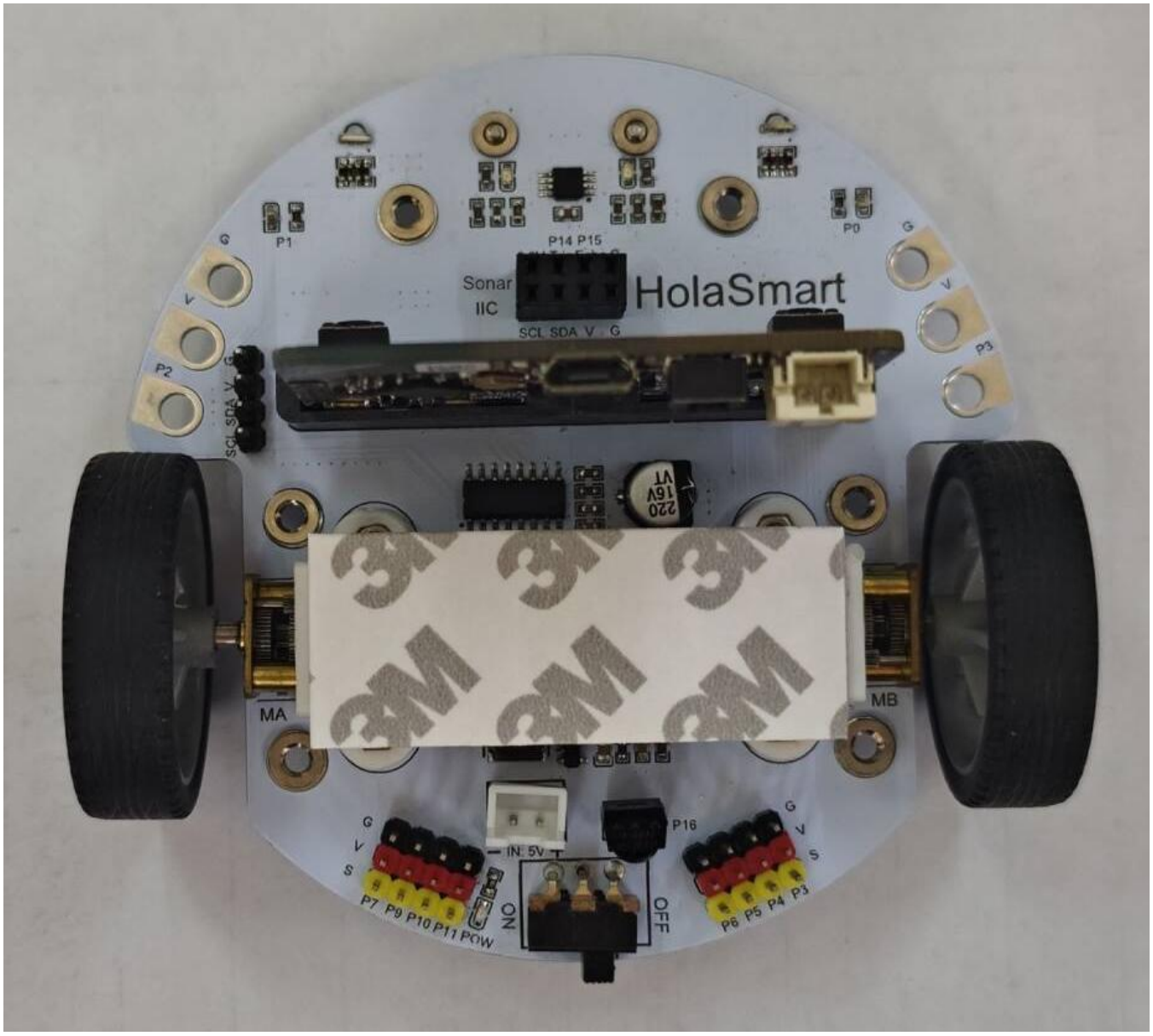
2. Install the two wheels onto the motor shaft.



3. Install tape, tear off the cover and attach it to the motor support.

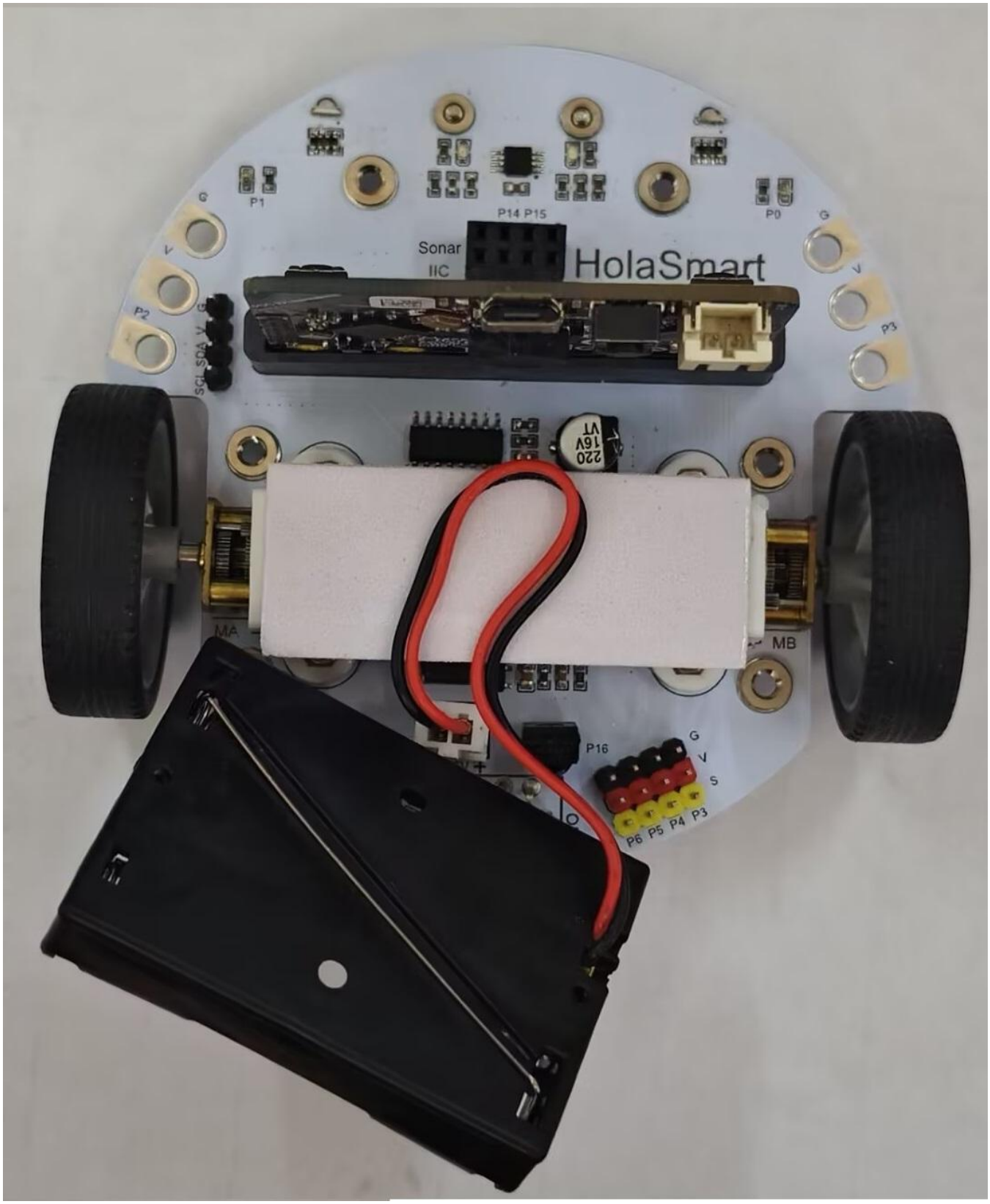


4. Install Micro bit board.

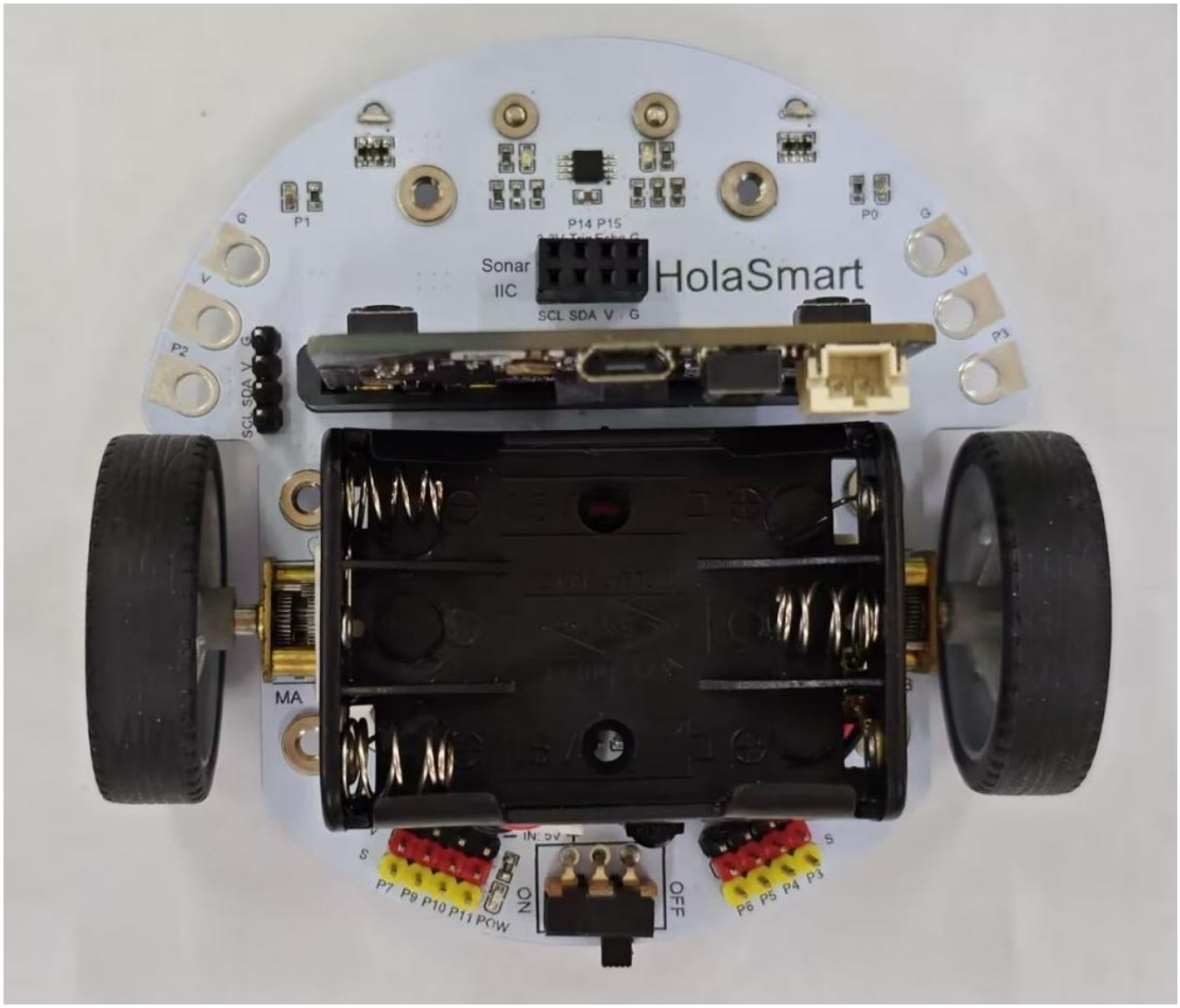


5. Mount battery holder.

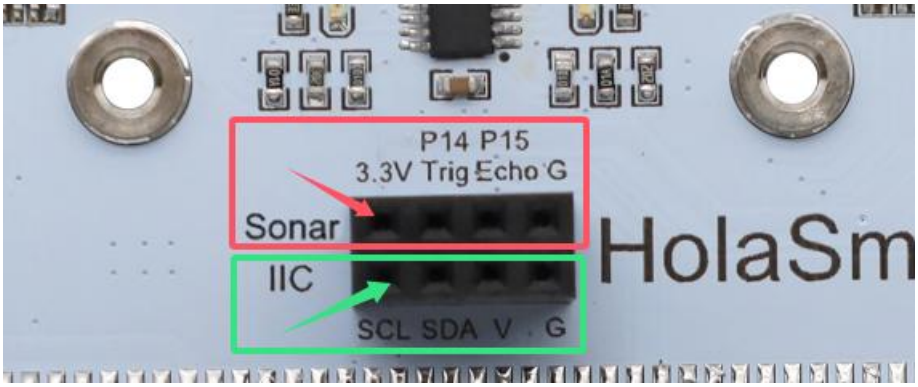
Peel off the tape cover, and then plug the interface of the battery holder into the drive board, and paste the wire on the tape as follows:

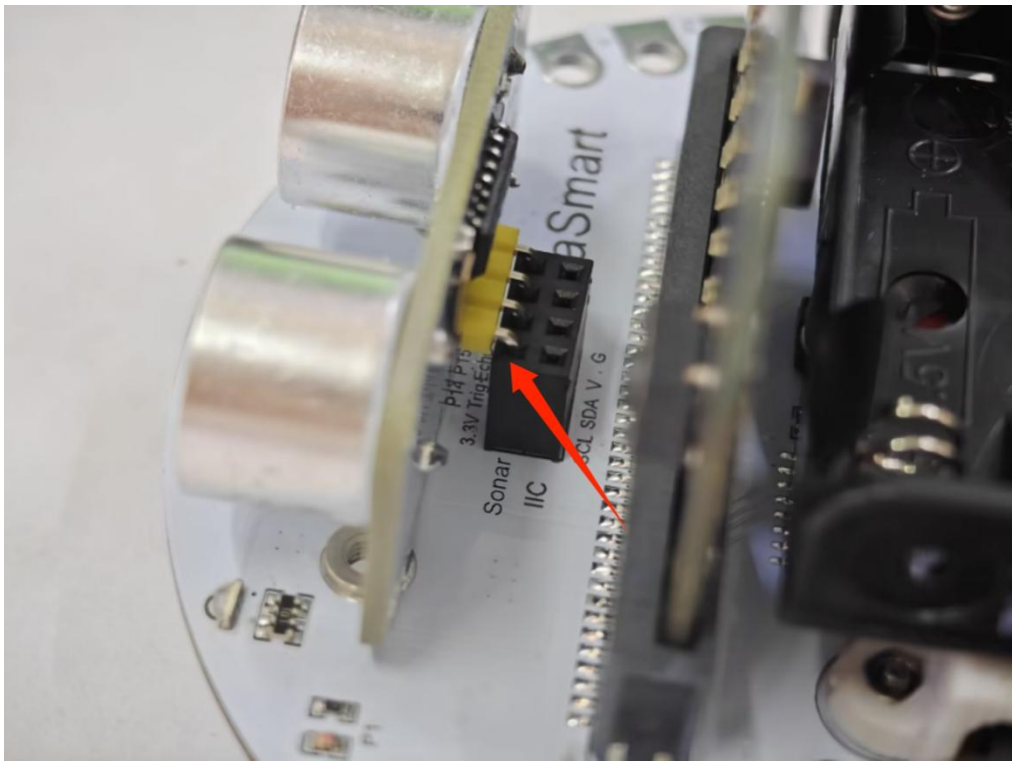

Sonar
IIC
HolaSmart
P1
P14 P15
P0
P2
P3
SCL SDA V G
220
16V
MA
MB
P16
G
V
S
P6 P5 P4 P3

Put the battery box at the middle position and press it tightly. (Note: please install the Micro: bit board before battery holder, because there should be a distance between the Micro: bit board and the battery holder as shown in the picture.)



6. Install the ultrasonic sensor. Please pay attention the position of it. Ports in the red box are for the ultrasonic sensor while that in the green are for ICC.





7. Now the small car is fully assembled. If you fell not convenience when mounting batteries, you may remove the Microbit for a while.

Smart Car Projects

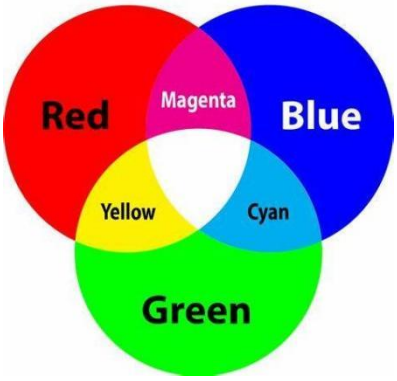
Project 1 RGB LED

1.1 Introduction

RGB color mode is a color standard in the industry. A variety of colors can be got through the change of red (R), green (G), blue (B) three color channels and their superposition. This standard is one of the most widely used color systems, which includes almost all the colors that human vision can perceive.

Most of the display adopts RGB color standard. They show colors through the electron gun emitting red, green and blue. Generally, computers can display 32-bit with more than 10 million colors. All the colors are made up of the three colors mixed in different proportions. And a group of reds, greens, and blues is a minimal display unit. Any color can be recorded and expressed by a set of RGB values, so they are also known as the three primary colors of light: R(red), G(green), B(blue).

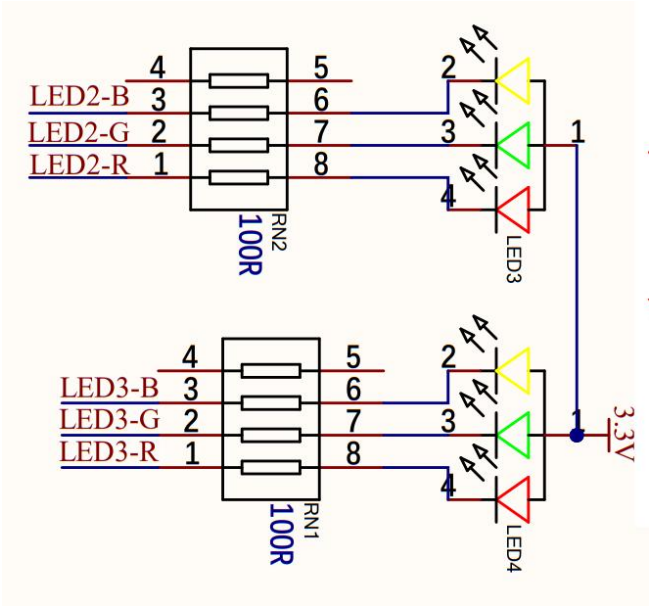
RGB, simply, shows colors by mixing red, green, blue. When they are superficially combined, the color is mixed, and the brightness is equal to the sum of the three. Thus, the more mixed lights is, the higher the brightness will be. That's what we called addictive mixing. The brightest area in the center is white. Usually we called seven-color LED in daily life.

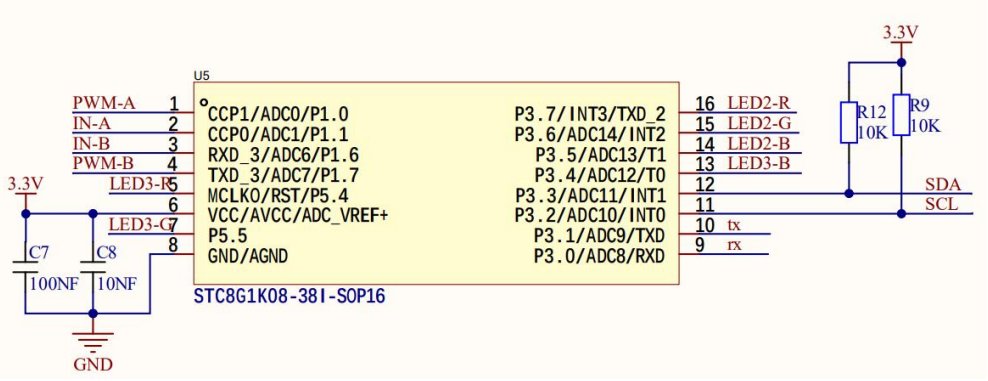


Color	RGB (R,G,B)	Color	RGB (R,G,B)
LED Off	255,255,255	red	0,255,255
green	255,0,255	blue	255,255,0
cyan	255,0,0	magenta	0,255,0
yellow	0,0,255	white	0,0,0
.....	.....	.....	.....

In this project, we will finish two experiments. One is to light up the two LED in red, green, blue, cyan, magenta, yellow and white circularly; The other is to make them gradient-display different colors of light.

1.2 Working Principle





Working Principle:

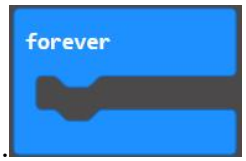
With microbit as the master, it sends instructions to the slave STC8G1K08 through IIC to output PWM to control RGB LED. This greatly saves the IO ports because only IIC is needed to control two motors and RGB LED.

1.3 Code Blocks

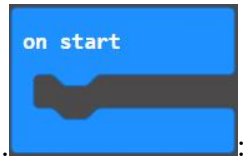
Basic Blocks:

In 

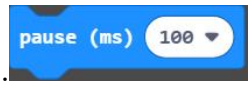
:

1. 

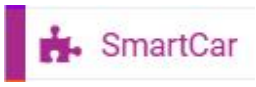
: codes in it will be repeatedly executed.

2. 

: this is an initialization block, which is used to initialize serial port and sensors and define variables. Codes in this block will only run once.

3. 

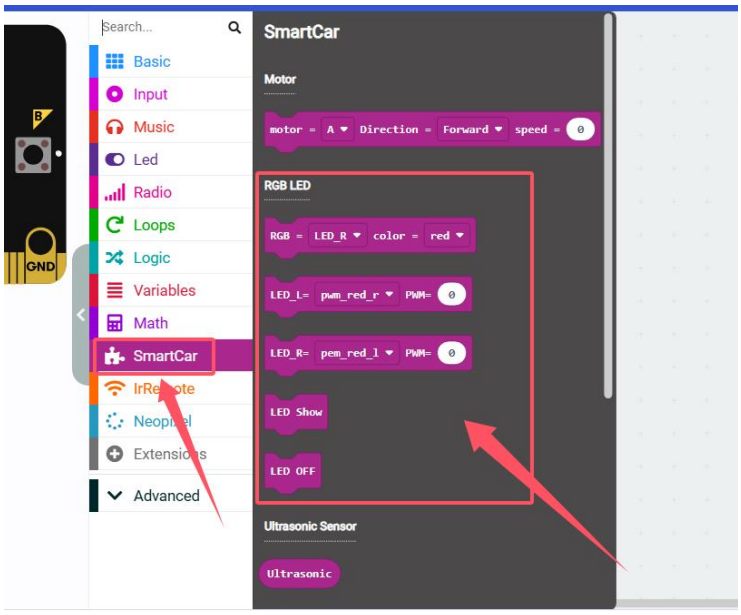
: it is a delay block, and you only need to set a delay time in it. Its unit is ms, 1s=1000ms.

In 

:

If there is no `SmarCar` in your MakeCode, please import the library first.

The detailed tutorial can be found in the directory：Programming → 4. Makecode Extension Library



4. 

: this block controls the colors of RGB LED on the car. You only need to set to`LED_R`(left RGB LED) or `LED_L`(right RGB LED) and the color.

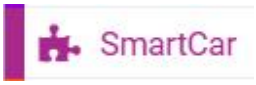
5. 

: a light show is integrated in this block, which can be directly add to code when using.

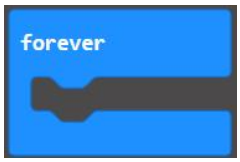
6. 

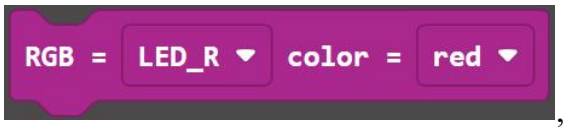
: this block turns off RGB LED.

1.4 Test Code

1.In 

, drag 

and put it in 

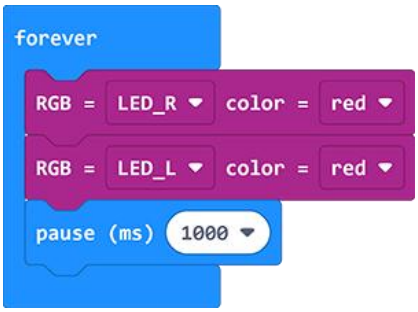
, and add a 

, set RGB LED to `LED_L` and color to `red`.

2.In 

, find and drag 

and set the delay time to `1 second`(1000ms).



Both left and right RGB LED show red for 1s

3.Similarly, build blocks to light up them in green for 1s. Duplicate the above blocks and modify the color to `green`.

Both left and right RGB LED show red for 1s and then green for 1s

4.Build blocks to light up them in blue for 1s. Duplicate the above blocks and modify the color to `blue`

5.Build blocks to light up them in cyan for 1s. Duplicate the above blocks and modify the color to `cyan`

6.Build blocks to light up them in purple for 1s. Duplicate the above blocks and modify the color to `purple`

7.Build blocks to light up them in white for 1s. Duplicate the above blocks and modify the color to `white`

8.Build blocks to light up them in yellow for 1s. Duplicate the above blocks and modify the color to `yellow`

9.Build blocks to turn them off. Duplicate the above blocks and modify to `Turn off LED`

Complete Code:


forever
RGB = LED_R color = red
RGB = LED_L color = red
pause (ms) 1000
RGB = LED_R color = green
RGB = LED_L color = green
pause (ms) 1000
RGB = LED_R color = blue
RGB = LED_L color = blue
pause (ms) 1000
RGB = LED_R color = cyan
RGB = LED_L color = cyan
pause (ms) 1000
RGB = LED_R color = purple
RGB = LED_L color = purple
pause (ms) 1000
RGB = LED_R color = white
RGB = LED_L color = white
pause (ms) 1000
RGB = LED_R color = yellow
RGB = LED_L color = yellow
pause (ms) 1000
RGB = LED_R color = Turn off LED
RGB = LED_L color = Turn off LED
pause (ms) 1000

1.5 Test Result

After uploading the code, the two RGB LED on the front of the car will be on in red, green, blue, cyan, purple, white, yellow in turn, with each color lighting for one second, and then they will go off for one second. These actions repeat.

Project 2 Breathing RGB LED

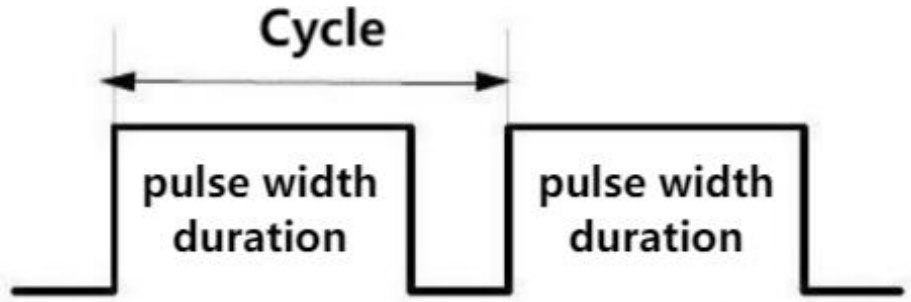
2.1 Introduction

Breathing LED uses the on-board programmable PWM to output analog waves whose duty cycle controls the brightness of LED. With PWM, the brightness is able to change as time, so the LED will gradually light up and dim, generating colorful or tranquil light effect.

2.2 Working Principle

PWM (Pulse width modulation) simulates the change of analog signal through digital signal.

Pulse width is the high level in a complete square wave cycle. So, pulse width modulation is to adjust the high level(of course, in other words, low level is also adjusted).



- **PWM frequency**: the number of times the signal going from high level to low level and back to high level in 1 second (one cycle), that is, how many cycles there are in a second.

  Unit: Hz

  Expression: 50Hz 100Hz

- **PWM cycle**

  $T = \frac{1}{f}\ Cycle = \frac{1}{frequency}$

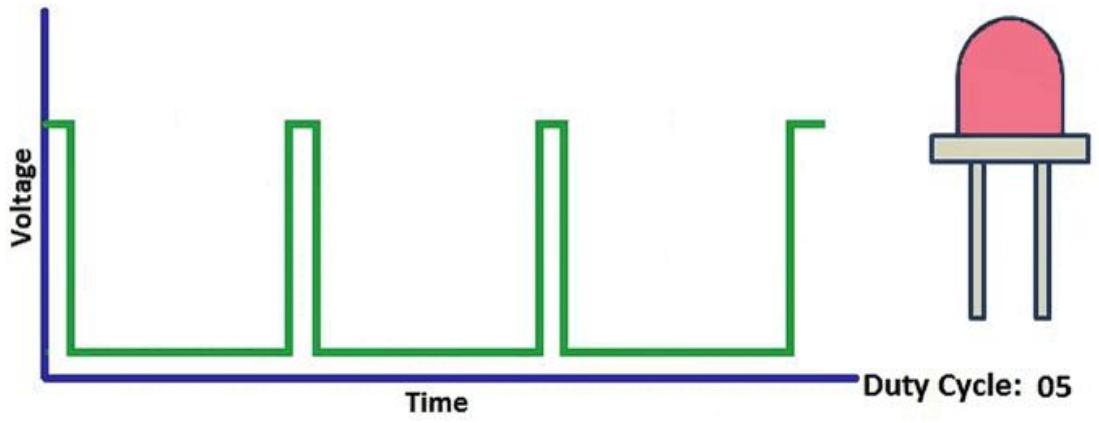
If the frequency is 50Hz, the cycle will be 20ms, i.e., there are 50 PWM cycles in one second.

- **PWM duty cycle**: the ratio of high level time to the whole cycle time.

  Unit: %(1% ~ 100%)

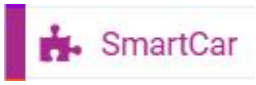
  Cycle: The time of a pulse signal. The number of cycles in 1s equals the frequency.

  Pulse width time: high level time.



The relationship between duty cycle and LED brightness

2.3 Code Blocks

In 

:

1. 

: this block controls the PWM of the left RGB LED (its brightness depends on PWM). You only need to set the color (only red, green and blue are optional) and input PWM value (range 0-255).

2. 

: this block controls the PWM of the right RGB LED (its brightness depends on PWM). You only need to set the color (only red, green and blue are optional) and input PWM value (range 0-255).

In 

:

3. 

: it sets the repeating times. Blocks in it will run for a certain number of times.

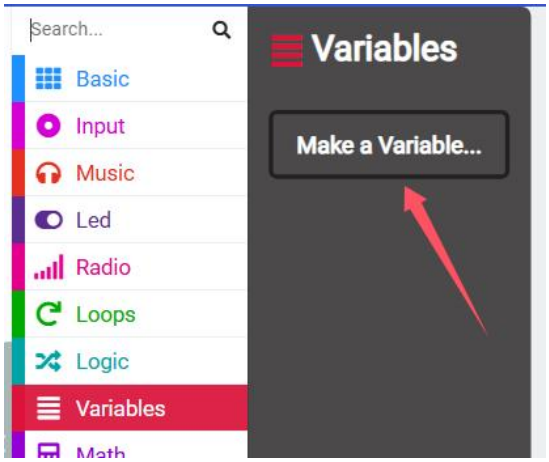
In this tutorial, we only list the most simple usage. For more detailed and advanced information, please visit: Serial (microbit.org)

In 

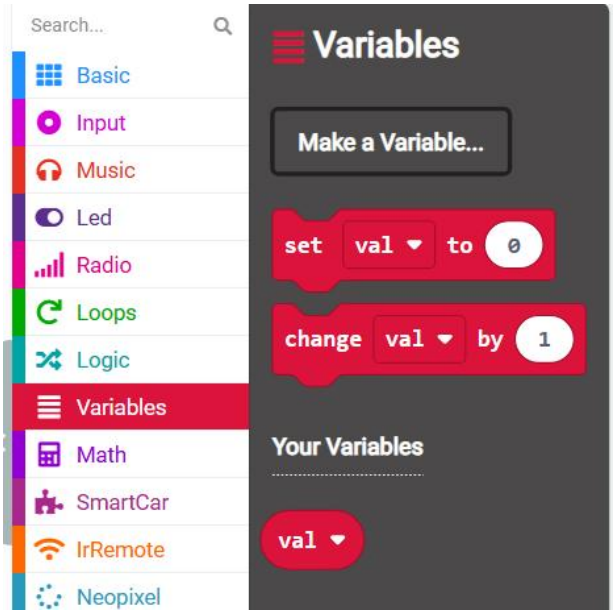
:

Click 

to create a variable and name it to “val” (you can name whatever you like):



The variable is defined:



4. 

: this block assigns a value to a variable.

5. 

: this block automatically increases/decreases the variable. For instance, input "1" and the variable adds 1; input "-1" and the variable minus 1.

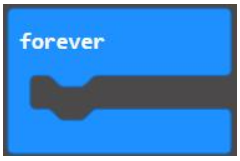
6. 

: this is the variable. When using, choose the variable you want to do operations.

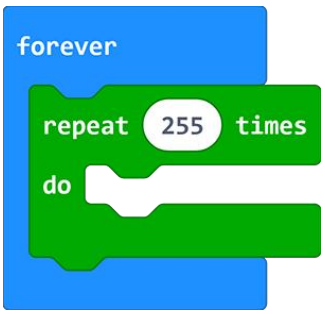
2.4 Test Code

1.In 

, drag 

and set times to "255" and put it into 

.

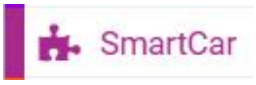


2.In 

, define a variable “val” and add 

into 

. Herein, we need to increase the variable so we set to 1.

3.In 

, find 

and 

, maintain the default color, and put them under 

.

4.In 

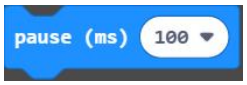
, put 

into 

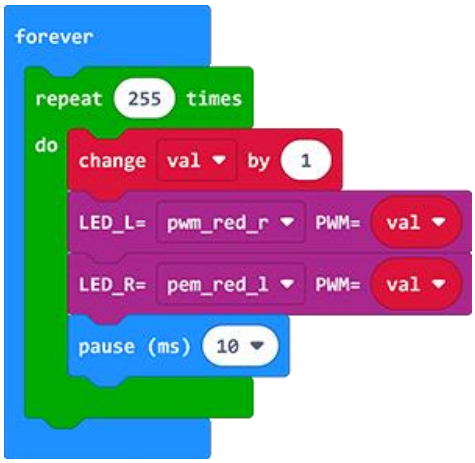
and 

to set the variable as the PWM value.

5.In 

, add a delay 

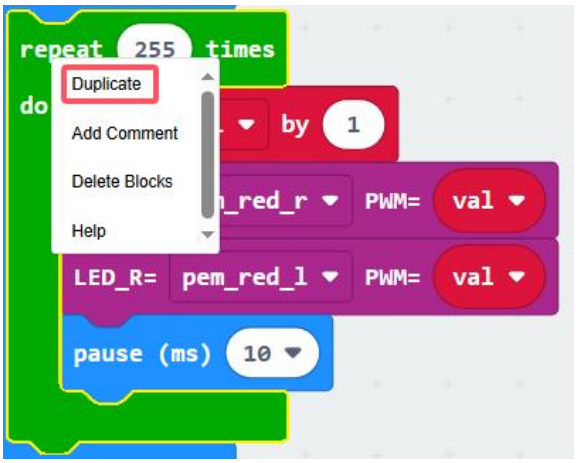
and set to 10ms.



Both RGB LED change from dark to bright gradually

6.Right-click 

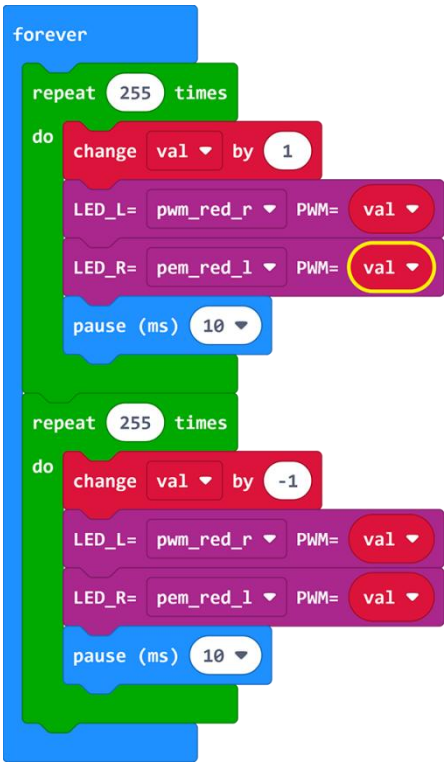
and choose `Duplicate`.



7.In the copy blocks, modify “1” to “-1” in 

, so that LED will gradually light up (from bright to dark).

Complete Code:



2.5 Test Result

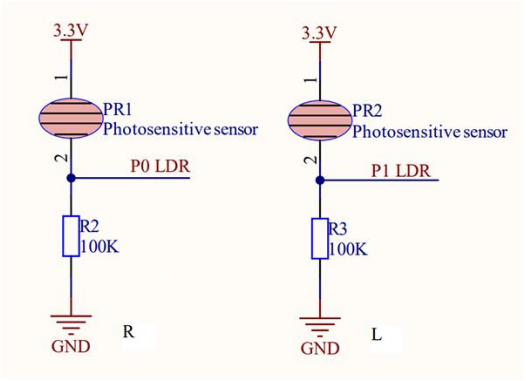
After uploading the code, both RGB LED will gradually light up in red and then dim. Then they gradually light up again, in a loop.

Project 3 Photoresistor

3.1 Introduction

Have you ever noticed that the street lights outside turn on when it gets dark and turn off when it gets light! In this project, we will mainly adopt a photoresistor to simulate this light.

3.2 Working Principle



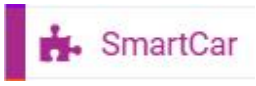
Working Principle:

Photoresistor works on the basis of semiconductor photoelectric effect, so it is very sensitive to ambient light whose resistance value changes with light intensity. We design a circuit with photoresistor in this experiment.

There is an analog port on the photoresistor. When the light gets brighter, the resistance decreases and the port voltage increases, so the analog value of the MCU also rises. On the contrary, when the light weakens, the resistance increases and the voltage decreases, so the analog value of the MCU becomes smaller. In this way, the photoresistor can be used to read the corresponding analog values and sense the ambient light.

Generally, photoresistors are applied to light measurement and control, photoelectric conversion (converting changes in light into changes in electricity) and light regulation in light control circuits and switches.

3.3 Code Blocks

In 

:

1. 

: this block reads the analog value of the right photoresistor.

2. 

: this block reads the analog value of the left photoresistor.

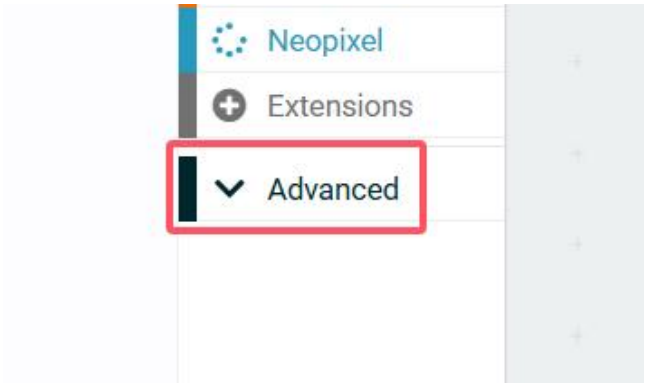
In 

:

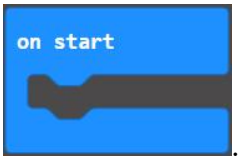
Click 

and you will see 

:



1. 

: this block enables the USB serial port. When using, put it in 

.

2. 

: this block enables the serial monitor to print contents. When using, set a name to distinguish the values. The value in the last box will be printed.

More details, please visit: Serial (microbit.org)

3.4 Test Code

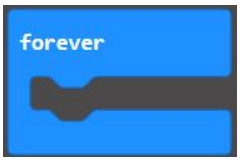
1.In 

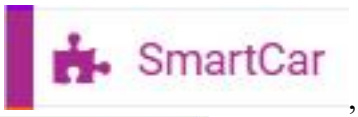
, put 

into 

2.In 

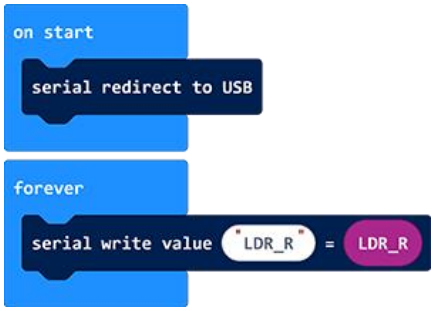
, put 

into 

and set “LDR_R” as the print content; in 

, drag 

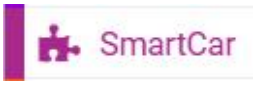
and put it in the last box of 



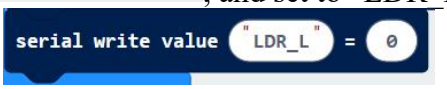
3.In 

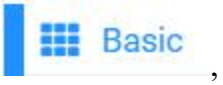
, add another 

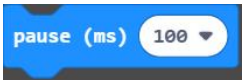
under 

, and set to “LDR_L”; In 

, drag 

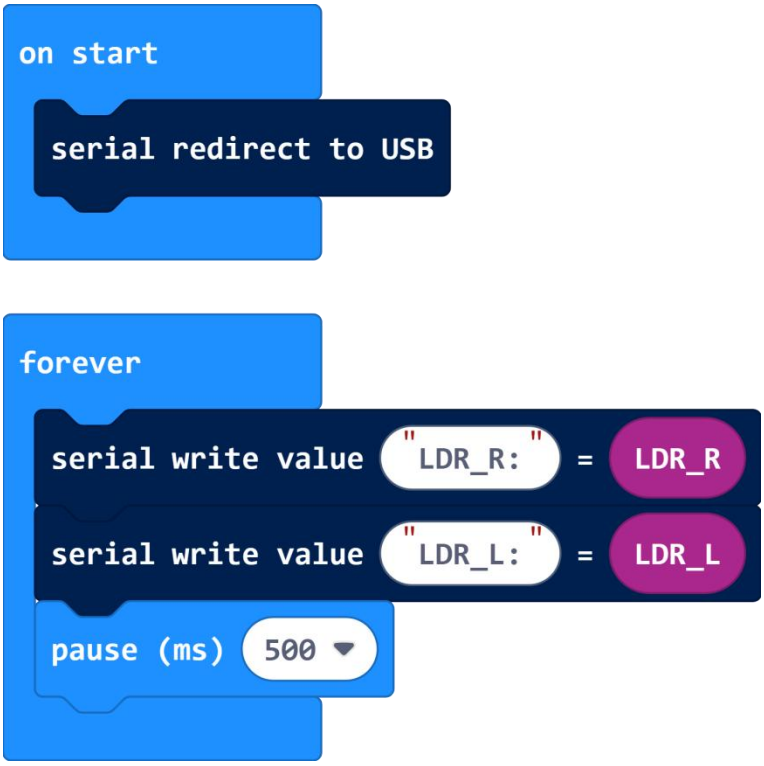
and put it in the last box of 

4.In 

, add a delay 

and set to 500ms

Complete Code:

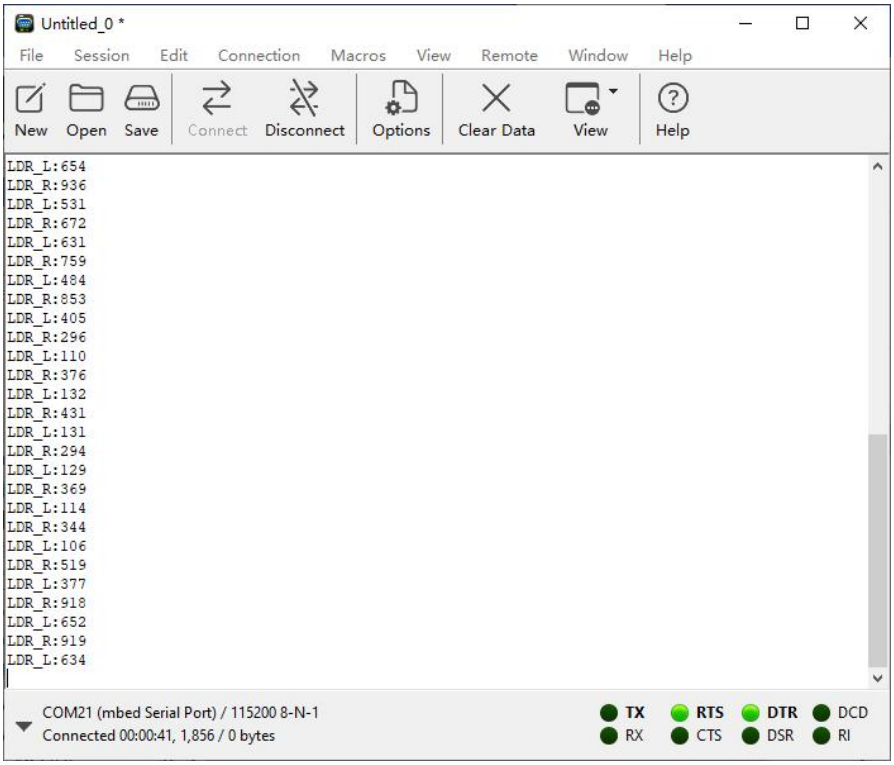


3.5 Test Result

For Windows 10, click “Show data Device” to see the analog values of the photoresistors.

If your operation system is not Windows 10, or WebUSB is disabled, a serial tool is required when printing.

The detailed tutorial can be found in the directory：Programming → 5.Resources and Test Code → 2. Install CoolTerm



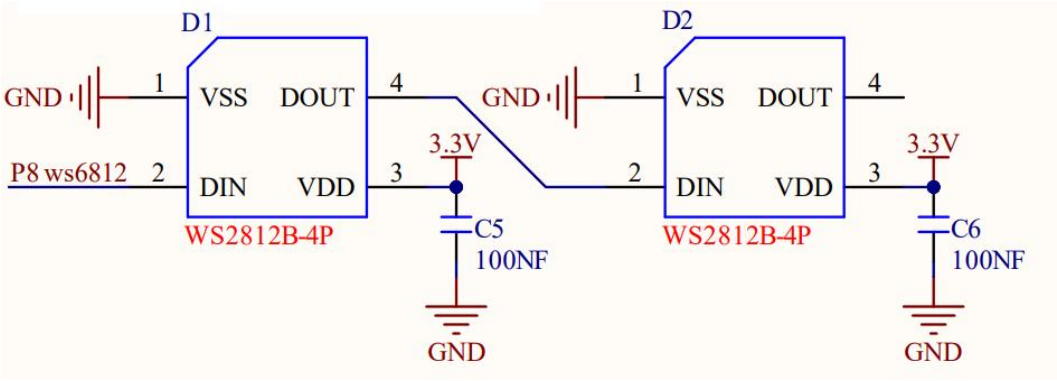
Project 4 WS2812 Pixel

4.1 Introduction

The WS2812 pixel integrates the control circuit and the RGB chip in the 5050 package to form a complete pixel control unit. Each RGB LED is with 256 brightness levels so they form 16,777,216 colors with a refresh rate of no less than 400Hz.

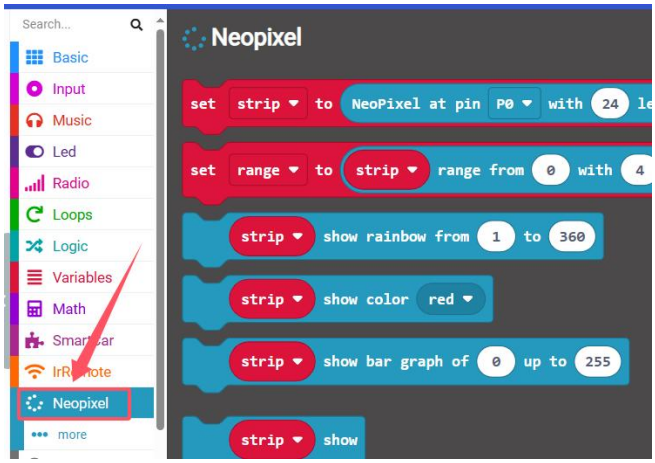
Besides, it supports cascade control to transmit signals through single wire series ports, and there is no additional circuits requirement for transmission within 5 meters. At a refresh rate of 30fps, it can control no less than 512 pixels at low-speed mode and more than 1024 pixels at high-speed mode.

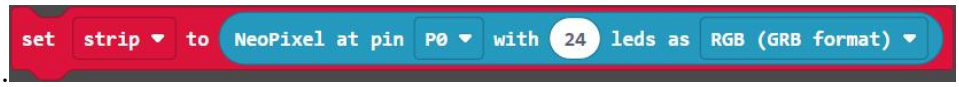
4.2 Working Principle



WS2812 adopts single-bus communication protocol, and each pixel supports 24-bit color control (RGB888). Signals are input through DIN, and after passing one pixel, the first 24 bits of data will be latched, and the remaining data will be output from DOUT after shaping, which forms cascade control.

4.3 Code Blocks

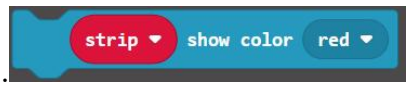


1. 

: this block initializes WS2812 RGB. The WS2812 control pin and the number of LED should be set firstly. The last option is usually left as default.

2. 

: this block adopts a variable to control a related pixel(LED). For instance, there are only two LED on the car, so we define `left` as the left LED and `right` as the right one. The left is 0-1 and the right is 1-1.

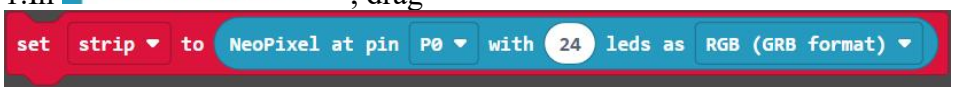
3. 

: this block shows the color of the WS2812 LED. the LED name (`left` and `right`) and the color need to be set.

4.4 Test Code

1.In 

, drag



and put it into

, and set the pin to `P8` and LED to `2`

2.In 

, create two variables name `left` and `right`

3.In 

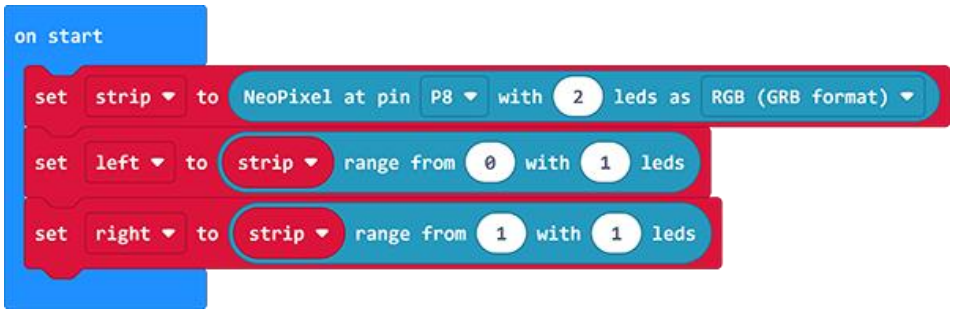
, find 

and modify `range` to `left` and set the range from `0` to `1`

4.In 

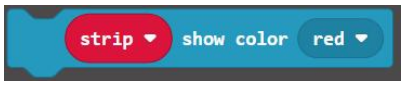
, drag another 

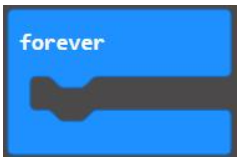
and set to `right`, and set the range from `1` to `1`

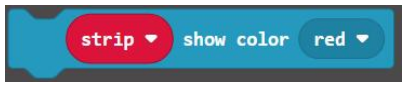


Condigure the pins and LED numbers of ws2812

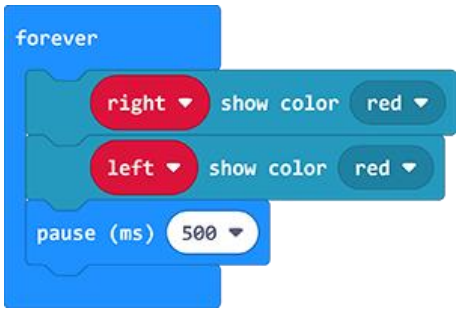
5.In 

, put 

into 

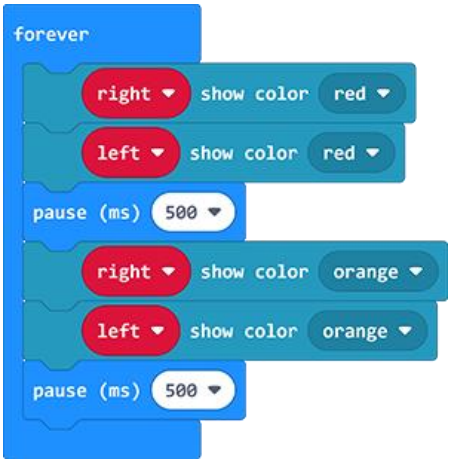
, change `strip` to `right` and set color to `red`. Drag one more 

and set to `left` and `red`. At last, add a delay of 500ms.



WS2812 show red

6.Similarly, build blocks to light up LED in `orange`



WS2812 show red for 500ms and change into orange

7.Build blocks to light up LED in `yellow`

8.Build blocks to light up LED in `green`

9.Build blocks to light up LED in `blue`

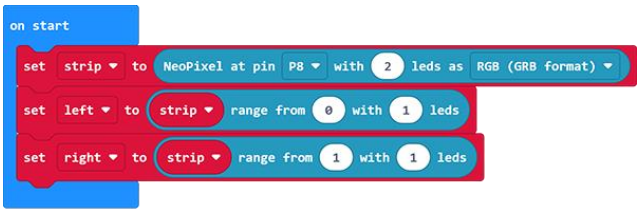
10.Build blocks to light up LED in indigo

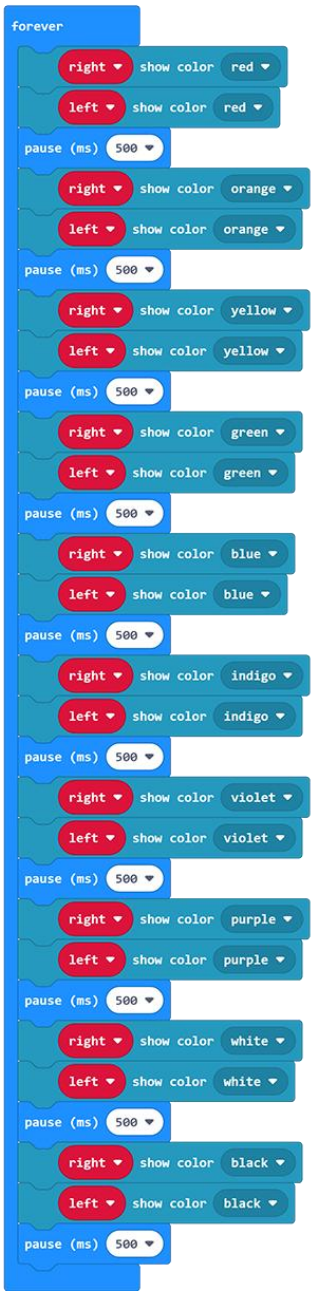
11.Build blocks to light up LED in violet

12.Build blocks to light up LED in purple

13.Build blocks to light up LED in white

14.Build blocks to turn off ws2812, so we set color to black





4.5 Test Result

After uploading the code, the ws2812 LED at the bottom of the car will light up and take turns in red, orange, yellow, green, blue, indigo, violet, purple, white, and off. Each maintains for 500ms.

Project 5 Line Tracking Sensor

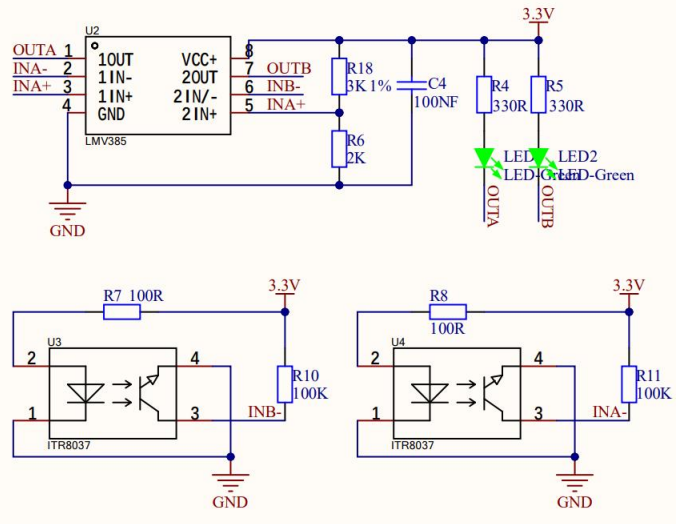
5.1 Introduction

Line tracking sensor detects and tracks a particular line by infrared/photoelectric technology to determine the presence/absence of a path by transmitting and receiving light. It is widely used in intelligent cars, autonomous vehicles and industrial robots.

Generally, the sensor is composed of a transmitter and a receiver. When the light encounters a black or dark surface, most of it will be absorbed and little is reflected (reflectivity is low). So the sensor will detect a change in signal, so as to determine whether the car deviates from the line. For accurate tracking, multiple sensors can be used simultaneously to form a detection array, providing more accurate motion control for mobile devices.

It features fast response speed, accurate positioning, simple structure and low cost, so it is widely applied to intelligent technology. With development, this sensor also improves towards higher intelligence and integration.

5.2 Working Principle



5.3 Code Blocks

In SmartCar:

Line Tracking: this block reads the values of the line tracking sensor. It will output different values when black or white line is detected.

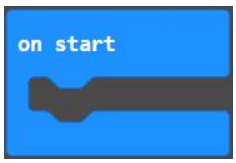
As shown below:

Left line tracking sensor (P12)	Right line tracking sensor (P13)	Output value
black	black	0
white	black	1
black	white	2
white	white	3

5.4 Test Code

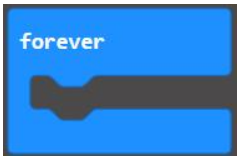
1.In 

, drag 

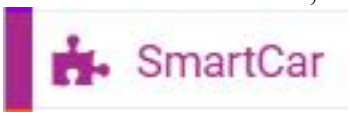
and put it in 

2.In 

, drag 

and put it into 

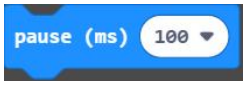
, set the serial write value to “line tracking”, and find 

in 

and put it in the last box of 

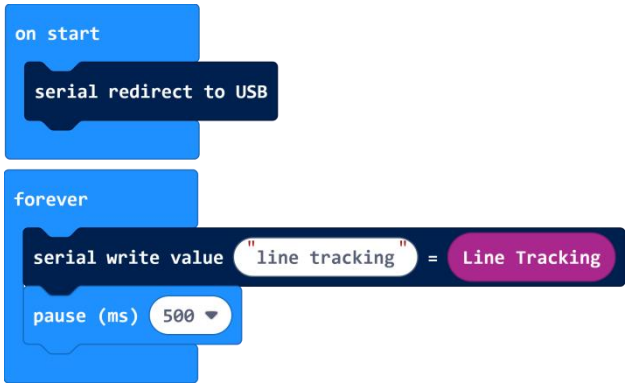
.

3.In 

, add a delay 

and set to 500ms.

Complete Code:

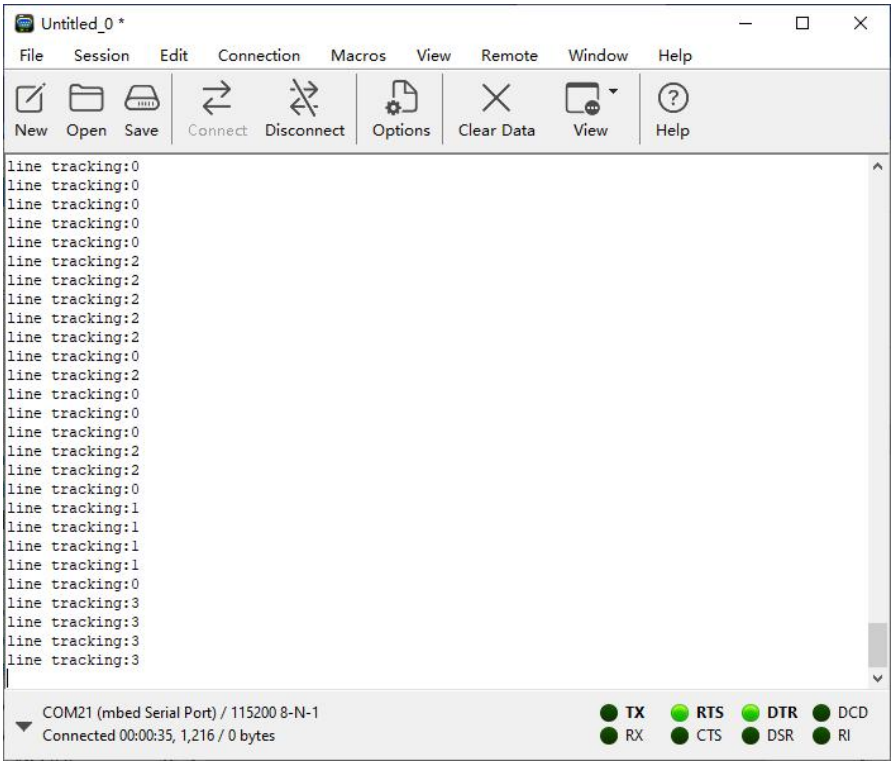


5.5 Test Result

After uploading the code, the serial monitor will print the values of the line tracking sensor. The results will be refreshed every 500ms.

For Windows 10, click “Show data Device” to see the values.

If your operation system is not Windows 10, or WebUSB is disabled, a serial tool is required when printing.



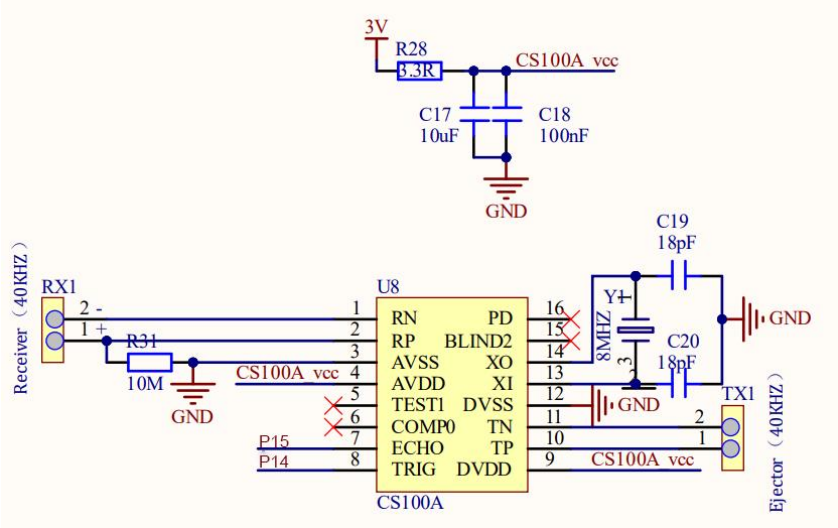
Project 6 Ultrasonic Sensor

6.1 Introduction

Ultrasonic sensor measures the distance of an object through sound waves.

Its transmitter emits sound waves, and its receiver receives the returned sound waves, so that the distance can be calculated according to the time difference between them. Beyond distance measurement, it can also be used to detect the shape of objects, build automatic doors, measure flow rates and pressures, detect the presence of objects, and so on.

6.2 Working Principle



Working Principle:

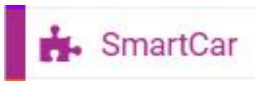
Like bats, the ultrasonic sensor sends an ultrasonic signal with a high frequency that human cannot hear. If these signals encounter obstacles, they will immediately reflect back and be received by the sensor. After that, the distance between the sensor and the obstacle is calculated according to the time difference between signals transmitting and receiving.

Maximum detection distance: 3M

Minimum detection distance: 4cm

Detection angle: no greater than 15 degrees

6.3 Code Blocks

In 

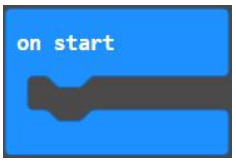
:

: this block reads the distance value detected by the ultrasonic sensor.

6.4 Test Code

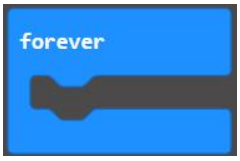
1.In 

, put 

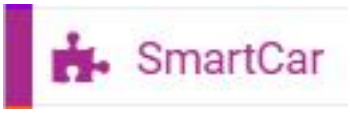
into 

2.In 

, put 

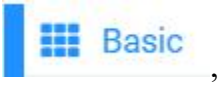
into 

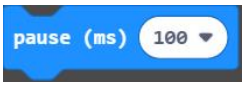
, and set serial write value to “LDR_R”, and drag 

in 

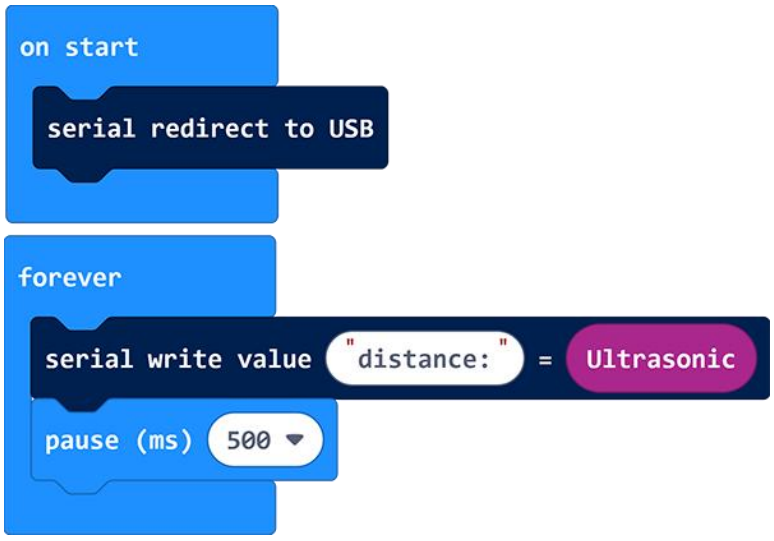
and put it in the last box of 

.

3.In 

, add a delay of 

and set to 500ms

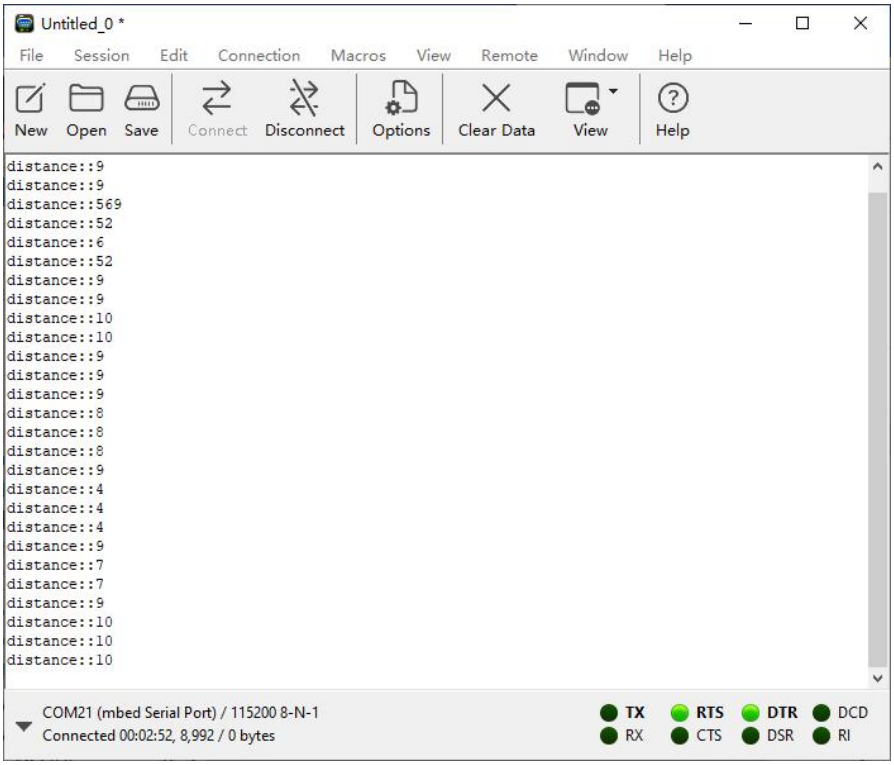


6.5 Test Result

After upload code, the serial monitor prints the distance value detected by the ultrasonic sensor. The results will be refreshed every 500ms.

For Windows 10, click “Show data Device” to see the values.

If your operation system is not Windows 10, or WebUSB is disabled, a serial tool is required when printing.



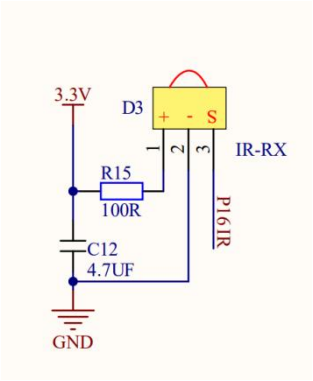
Project 7 IR Receiver

7.1 Introduction

Infrared remote control is so ubiquitous in daily life that it is hard to imagine what the world would be like without it. It is used to control various home appliances such as televisions, stereos, video recorders and satellite signal receivers.

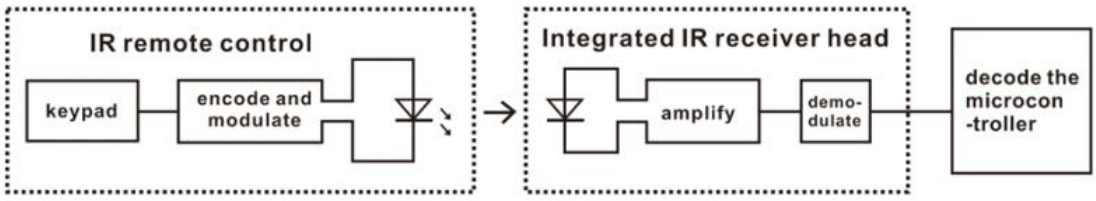
Infrared remote control is composed of infrared emission and infrared receiving system. To put it simple, they are a remote control, an infrared receiver and a single chip computer that can decode.

7.2 Working Principle



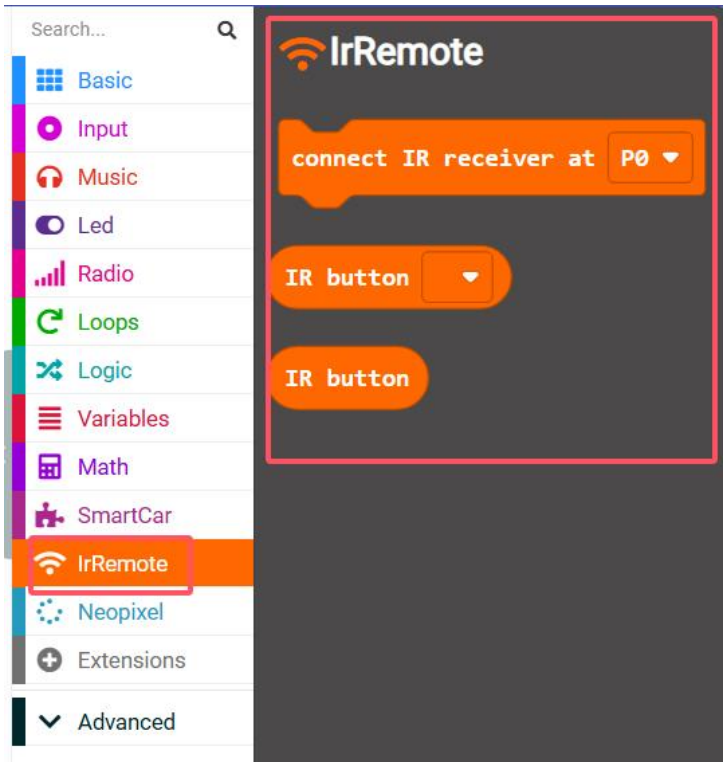
The 38K infrared carrier signal emitted by the infrared remote control is encoded by the coding chip in the remote control. It is composed of a guide code, user code, data code, data reverse code, using the time interval of the pulse to distinguish between signal 0 or signal 1(the ratio of high and low levels is about 1:1 is considered to be signal 0), and the coding is composed of these 0 and 1 signals. The user code of the same remote control is unchanged, and the data is different to distinguish the keys pressed by the remote control. When the remote control button is pressed, the remote control sends an infrared carrier signal. When the infrared receiver receives the signal, the program decodes the carrier signal and determines which key is pressed through the different data codes. The MCU decodes the 0 and 1 signals received, and determines what key the remote control presses.

Infrared receiving is the infrared receiving module on the expansion board, mainly composed of infrared receiving head, it is a set of receiving, amplifying, demodulation of one of the devices, its internal IC has been completed to understand the adjustment, can complete from the infrared to the output and TTL level signal compatible with all the work, the output is the digital signal. It is suitable for infrared remote control and infrared data transmission.



The control pin of the IR receiver on the car is pin P16 of the micro bit board.

7.3 Code Blocks



1. 

: this block initializes the IR receiver, and you only need to set the control pin. In this project, the pin of the car is P16.

2. 

: this block indicates the buttons on the remote control. Click  and you will have a clear look of all buttons on the remote control.

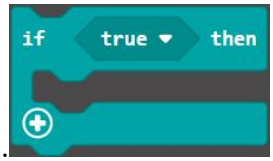
3. 

: this block reads the IR receiver values. Combined with 

, the button value can be known.

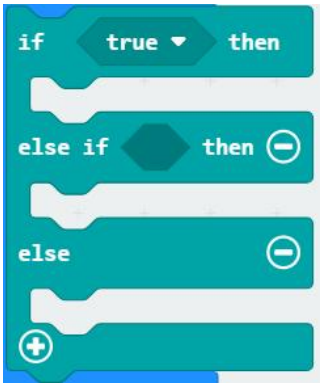
In 

:

1. 

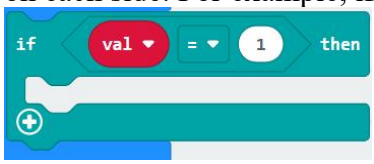
: this block determines whether the condition is true (“1” for True, “0” for False). You only need to add the condition command into 

. If it is satisfied, the following code blocks will be executed. Click  to add more “else” conditions. If the first condition is false, it will continue to determine the next one. If it is true, the conditions behind it will be skipped.



2. 

: this block determines whether the two values equal each other. Add a value on each side. For example, if we want to determine whether variable “ir” = 1, we write:



; Click 

to change the operator.

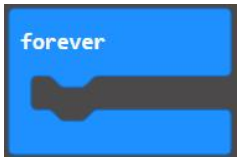
Operator	Meaning
	equal. Determine whether the two values equal each other.
	not equal. Determine whether the two values are not equal to each other.
	less than. Determine whether the left value is less than the right one.

Operator	Meaning
	less than or equal. Determine whether the left value is less than or equal to the right one.
	greater than. Determine whether the left value is greater than the right one.
	greater than or equal. Determine whether the left value is greater than or equal to the right one.

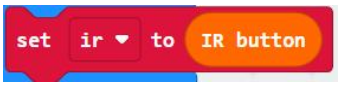
7.4 Test Code

1.In  , drag and put  into 

2.In  , place  under the  , and set the pin to P16

3.In  , define a variable named `ir`, and drag  and put it into  .

4.In  , put  into the box of 

5.In  , drag a  and put it under 

6.In 

, drag 

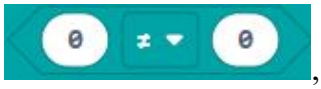
and add it into the condition box of 

, and modify the 

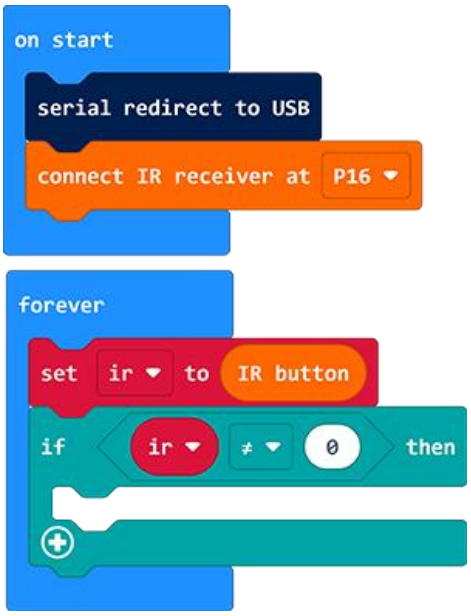
into 

7.In 

, drag 

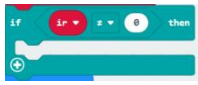
to the left box of the 

, and set the right value to “0”



8.In 

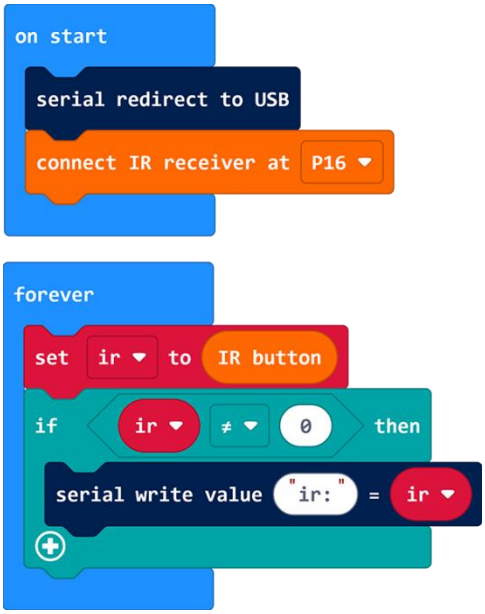
, drag 

into 
, and set the serial write value to “ir:”; find 

in 

and put it into the right box of 

Complete Code:

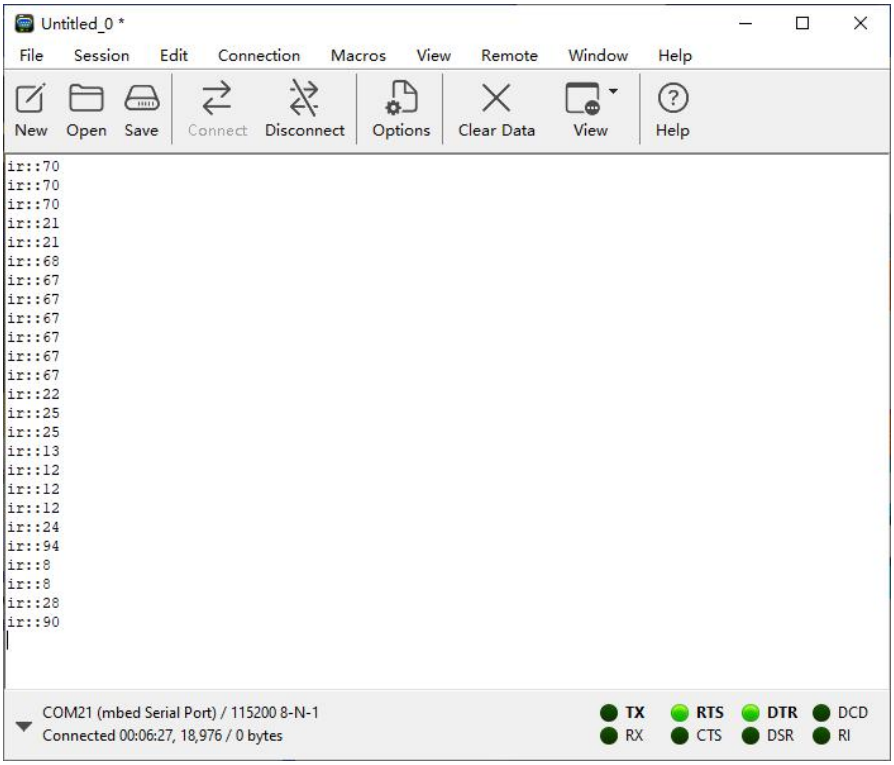


7.5 Test Result

After uploading the code, the serial monitor will print the button value received by the IR receiver. We add conditions in the code, so the monitor only outputs values when buttons on the remote control are pressed.

For Windows 10, click “Show data Device” to see the values.

If your operation system is not Windows 10, or WebUSB is disabled, a serial tool is required when printing.



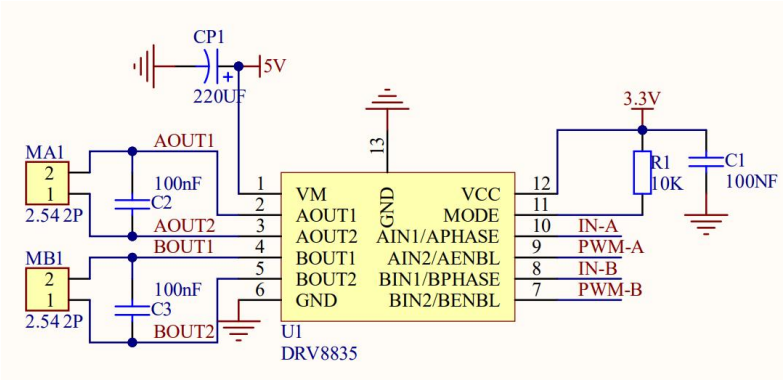
Project 8 Motor Drive

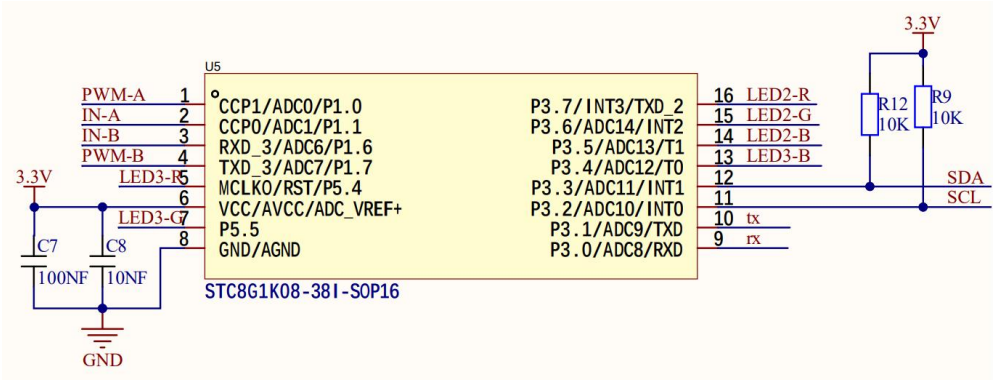
8.1 Introduction

The expansion board of the car is equipped with two DC gear motors, which boast additional gear reducers compared to ordinary DC motors. These reducers provide lower speed and greater torque, and their different reduction ratios correspond to different speeds and torques. This has greatly increased the use of DC motors in the automation industry.

Gear Motor refers to the integration of a reducer and a motor, and it is with simple design and small volume. In application, it is widely used in steel and machinery industry.

8.2 Working Principle

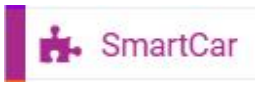




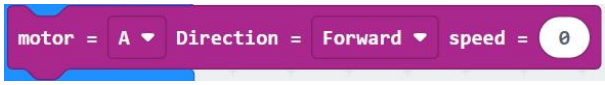
Motors on the car expansion board are controlled by STC8G1K08 chip and DRV8835 motor control chip.

To save IO ports, instructions are sent to the STC8G1K08 chip through the micro:bit IIC to enable the DRV8835 motor driver chip to control the rotation direction and speed of the two DC gear motors.

8.3 Code Blocks

In 

:



: this block controls the rotation of motors. You can choose 

(left motor) or B(right motor), and set the direction to 

`Forward` or `Backward`. In 

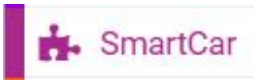
, input the speed value in it within the range of 0-255 (0 for stop, 255 for the fastest speed).

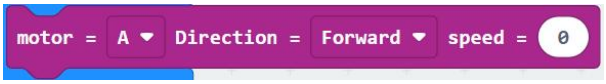
Motor control logic table:

A - left motor	B - right motor	speed value	car state

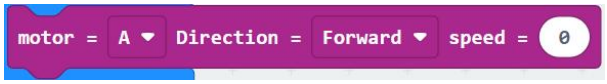
A - left motor	B - right motor	speed value	car state
Forward	Forward	200	Forward
Backward	Backward	200	Backward
Forward	Backward	200	turn left
Backward	Forward	200	turn right
Forward or Backward	Forward or Backward	0	Stop

8.4 Test Code

1.In 

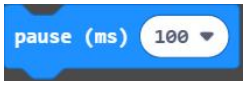
, drag 

and set the motor A direction to `Forward` and speed to `200`

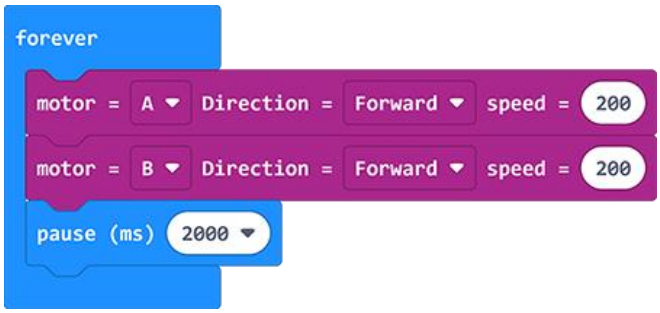
2.Add another 

and set the motor B direction to `Forward` and speed to `200`

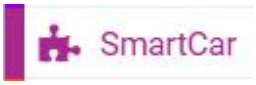
3.In 

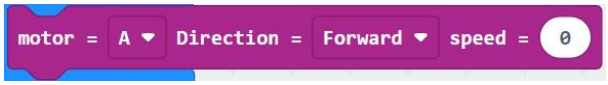
, add a delay 

and set to 2s (= 2000ms).

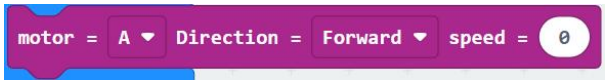


The car goes forward for 2s at a speed of 200

4.In 

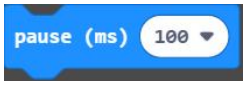
, add 

and set motor A direction to `Backward` and speed to `200`

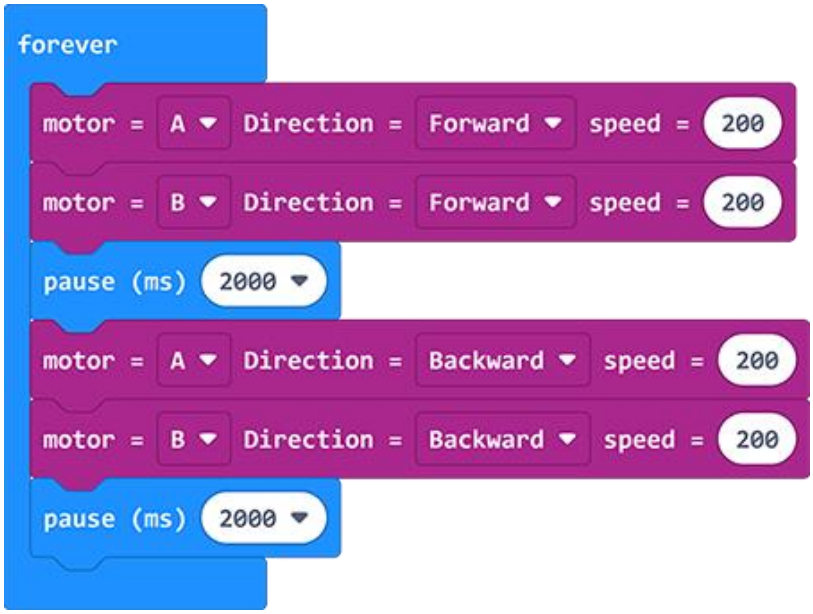
5.Add another 

and set motor B direction to `Backward` and speed to `200`

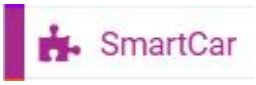
6.In 

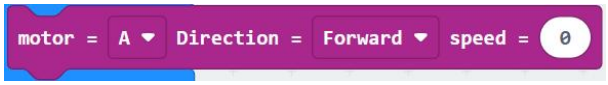
, add a delay 

and set to 2s (= 2000ms).

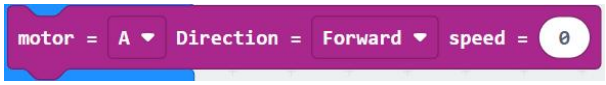


The car goes forward for 2s at a speed of 200 and then goes back for 2s

7.In 

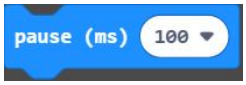
, add 

and set motor A direction to `Backward` and speed to `200`

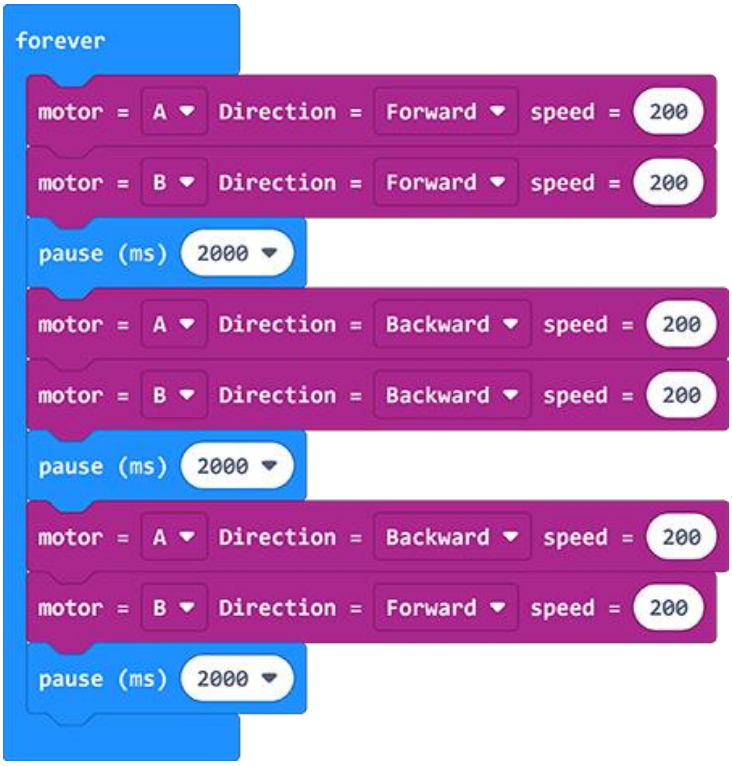
8.Add another 

and set motor B direction to `Forward` and speed to `200`.

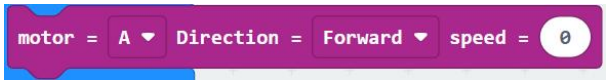
9.In 

, add a delay 

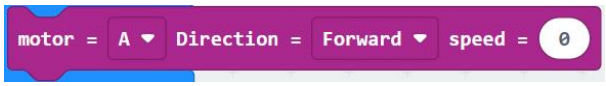
and set to 2s (= 2000ms).



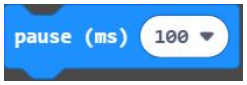
The car goes forward for 2s at a speed of 200, goes back for 2s and then turns left for 2s

10.In SmartCar, add 

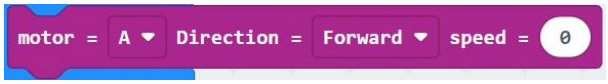
and set motor A direction to `Forward` and speed to `200`.

11.Add another 

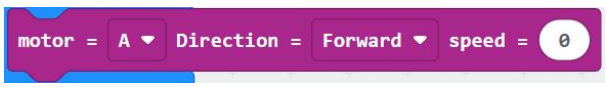
and set motor B direction to `Backward` and speed to `200`.

12.In Basic, add a delay 

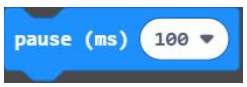
and set to 2s (= 2000ms).

13.In SmartCar, add 

and set motor A direction to `Forward` and speed to `0`

14.Add another 

and set motor B direction to `Backward` and speed to `0`

15.In Basic, add a delay 

and set to 2s (= 2000ms).

Complete Code:

8.5 Test Result

After uploading the code, the car goes forward, goes back, turns left, turns right and then stop, with each action lasts for 2s.

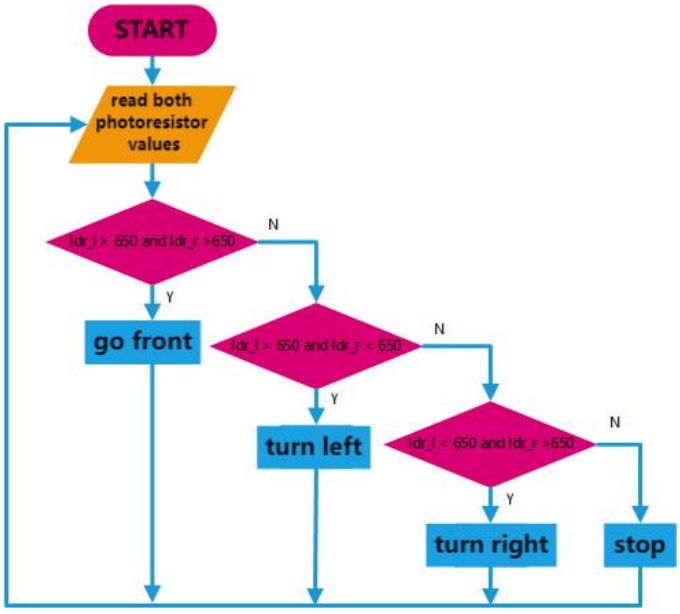
Project 9 Light Following Car

9.1 Introduction

How to make a light following car? We compare the analog values of the two photoresistors on both side of the car, and then program to control the car to go with the brighter side. If the two values are

almost equal, the car will go forward. But note that please do this experiment in a relatively dark environment, and turn on your flashlight.

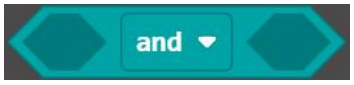
9.2 Code Flow



9.3 Code Blocks

In 

:



: this block outputs true only when conditions in both sides are true. If one or both of the conditions is(are) false, it will output false.

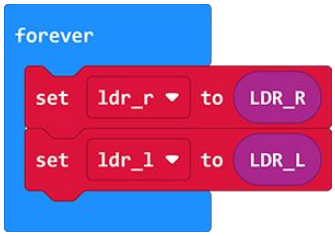
9.4 Test Code

1.In 

, define two variables named `ldr_l` and `ldr_r`, and assign them to the two photoresistors 

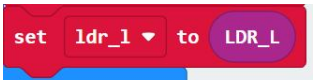
and 

,

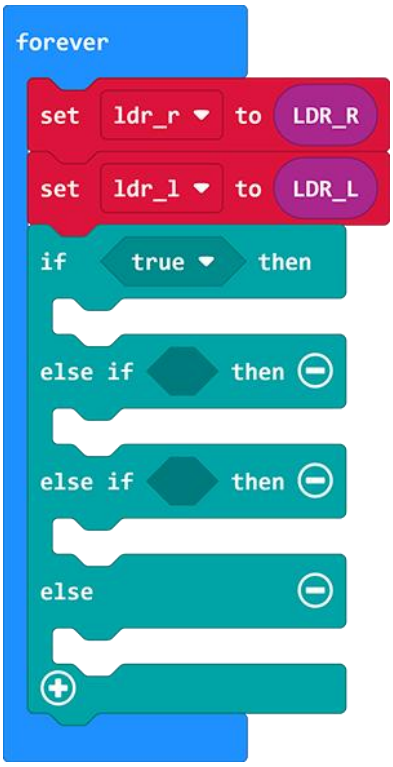


2.In 

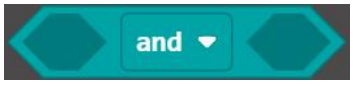
, drag a 

under 

and click  to add three more “else”:



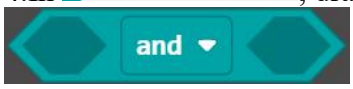
3.In 

, put 

into the first condition box;

4.In 

, drag two 

into the both sides of 

:

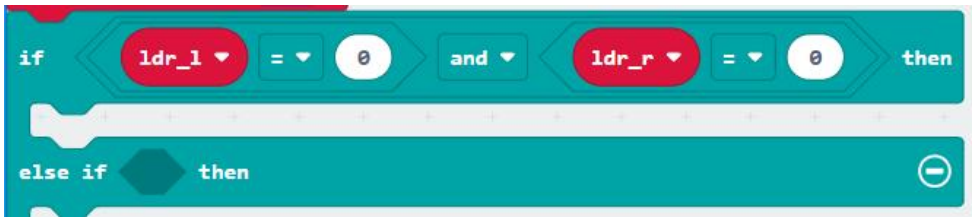
5.Put 

into the left box of the first 

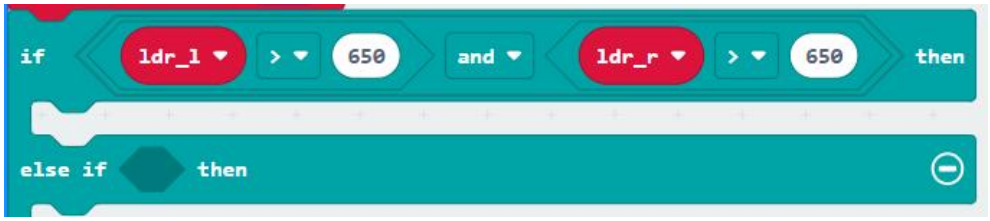
, and put 

into the left box of the second 

.

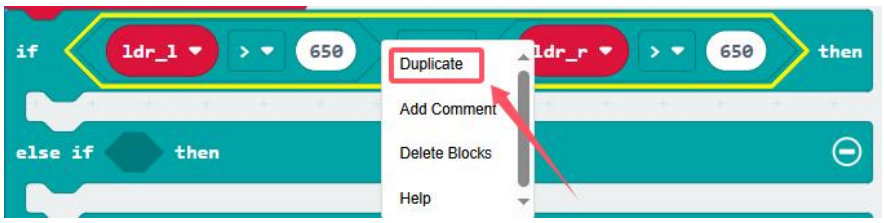


6.Modify the first  into ldr_l > 650; and modify the second  into ldr_r > 650. So it becomes: `ldr_l > 650 and ldr_r > 650`



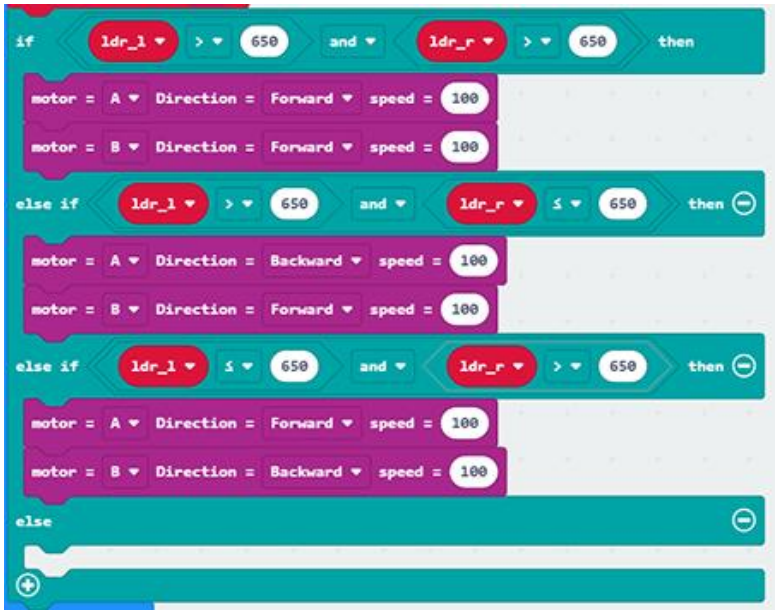
7.Add code blocks under "if…then…" block, and these codes only execute when the condition is true. When both photoresistor values are greater than 650, the car goes forward at a speed of 100

8.`Duplicate` the condition.



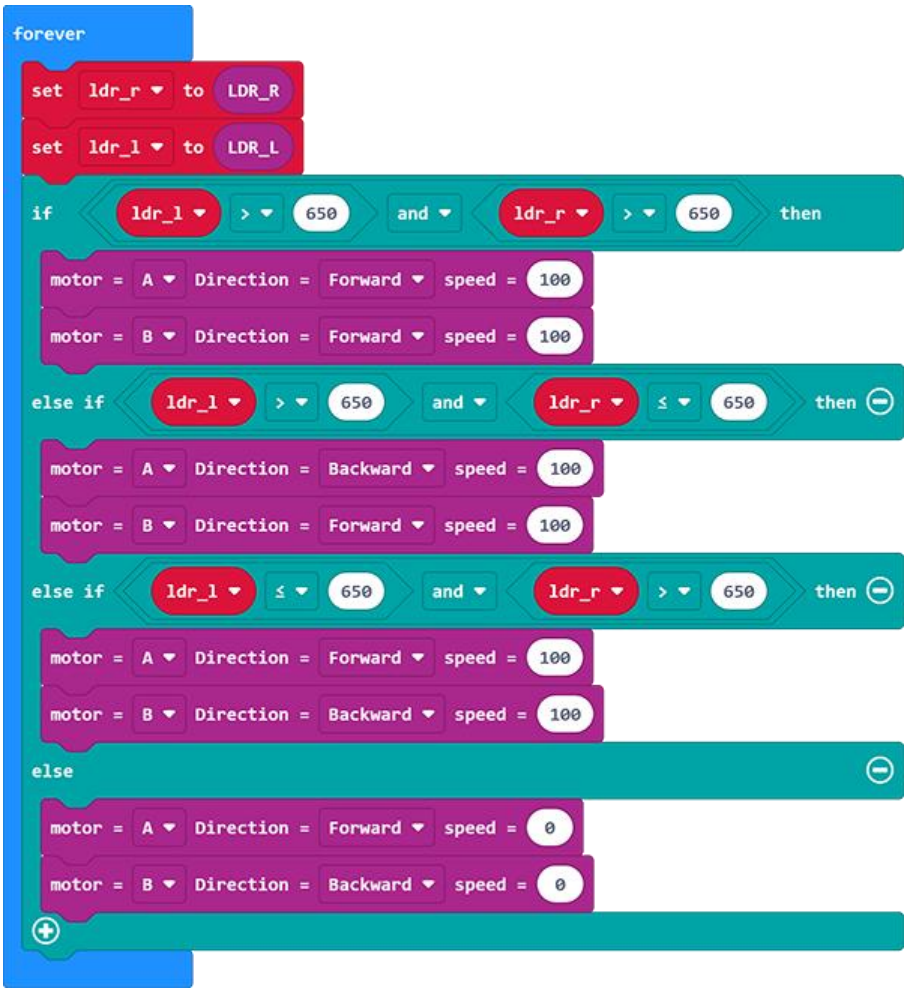
9.Modify the second condition into: `ldr_l > 650 and ldr_r ≤ 650` . When the left value is greater than the right one, the car turns left.

10.Duplicate the condition again and modify the second condition into: `ldr_l ≤ 650 and ldr_r > 650`. When the right value is greater than the left one, the car turns right.



11.Build blocks to stop the car in the last “else”.

Complete Code:



9.5 Test Result

After uploading the code, put the car in a relatively dark environment. Turn on your flashlight and the car will follow the light to move.

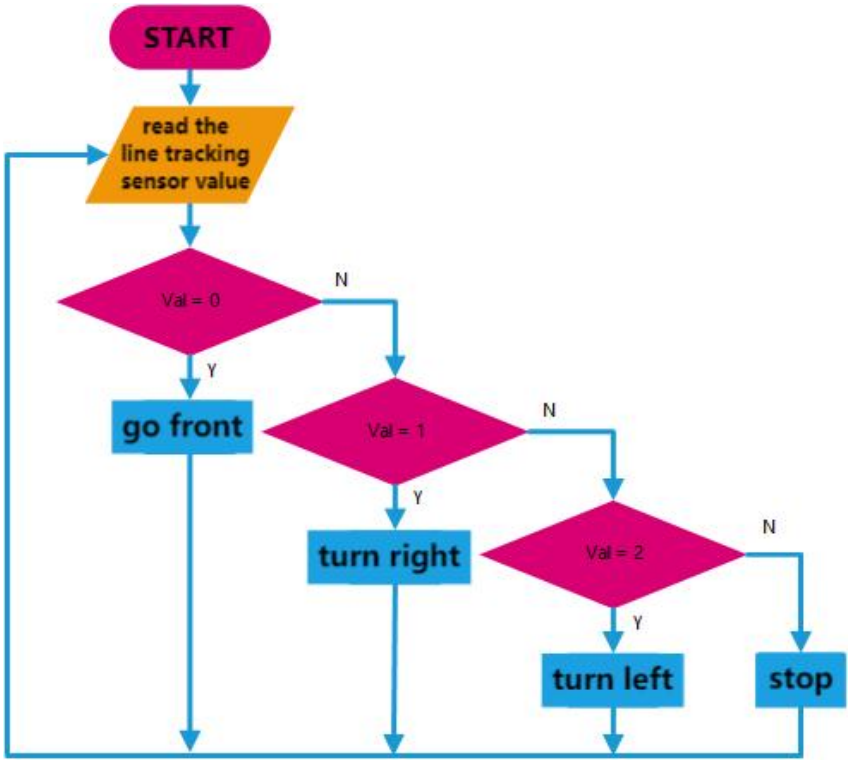
If you want the car to turn left, focus the light on the left side of its head. Similarly, focus the light on the right and it turns right. Focus the light on the front and it goes forward.

Project 10 Line Tracking Car

10.1 Introduction

In this project, the car adopts line tracking sensor to determine whether it is black or white under it. If it is white, the car turns to the black side. If it is always black, the car keeps going. In the kit, we provide you with a map with black line.

10.2 Code Flow

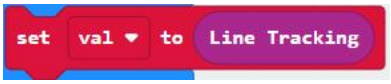


10.3 Test Code

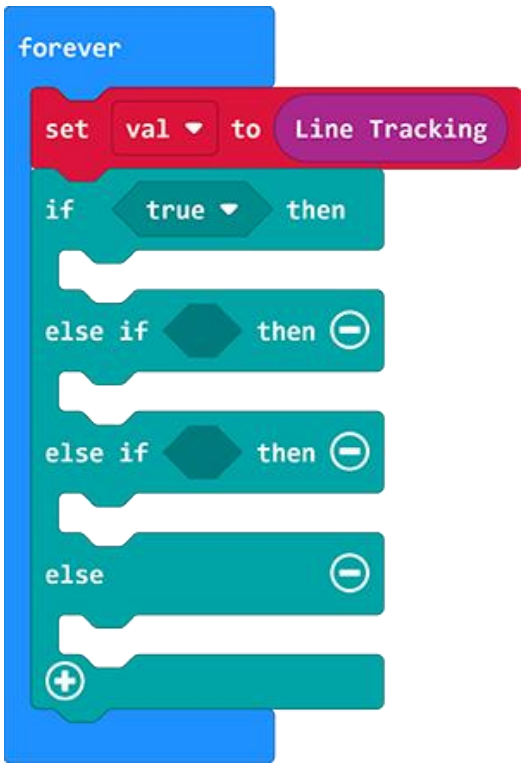
1.Define a variable named `val` and assign the line tracking sensor value to the variable.

2.In 

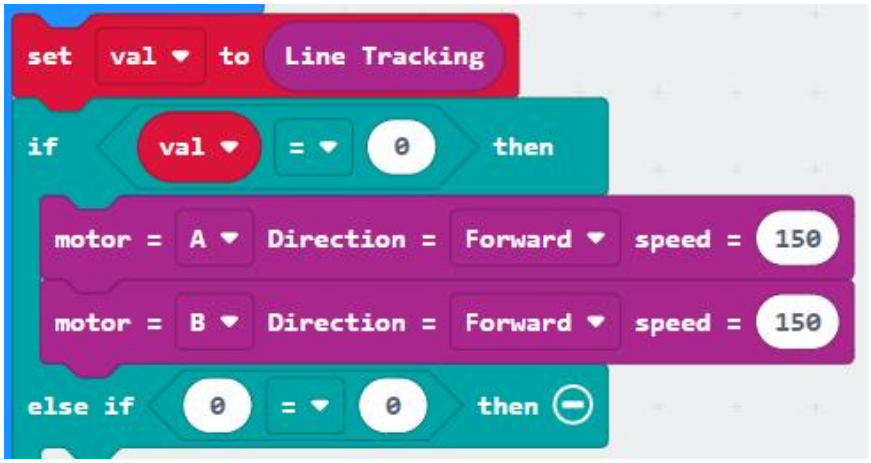
, drag 

under 

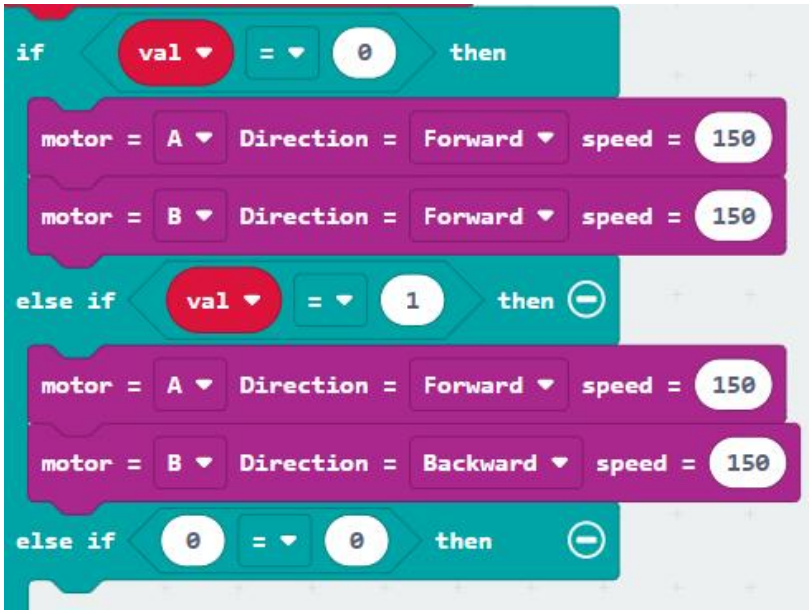
, and click 
to add three more “else”.



3.Add  into these condition box. Set the first condition to: `val = 0`. Add code blocks under “if…then…” block, and these codes only execute when the condition is true. If both sensors detects black, val = 0, and the car will go forward.



4.Set the second condition to `val = 1`. If the left sensor detects white while the right one detects black, val = 1, and the car will turn right.



5.Set the third condition to `val = 2`. If the left sensor detects black while the right one detects white, val = 2, and the car will turn left.

6.Add blocks in the last “else” to stop the car. If all conditions above are not satisfied (both sensors detect white), the car will stop.

Complete Code:

10.4 Test Result

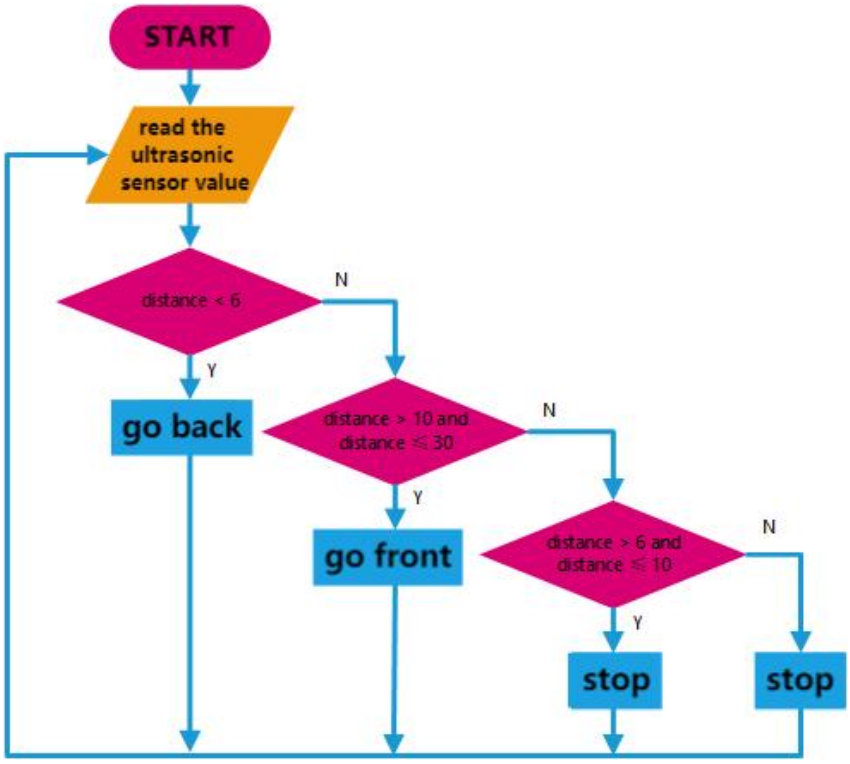
After uploading the code, unfold the map in the kit, and put the car on the map. You will see that the car follows the black line.

Project 11 Ultrasonic Object Following Car

11.1 Introduction

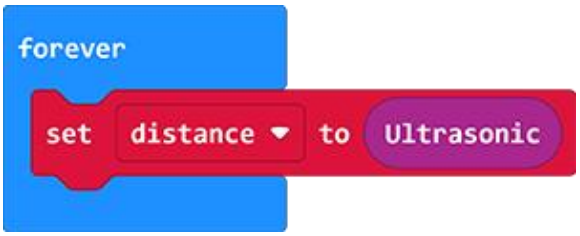
We make an object following car by an ultrasonic sensor in this project. If the distance between the car and the obstacle in front is within a certain range, the car moves forward and follows; if the they are too close to a certain range, the car moves back.

11.2 Code Flow



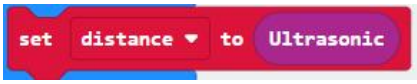
11.3 Test Code

1.Define a variable named `distance`, and assign the distance value detected by the ultrasonic sensor to the variable

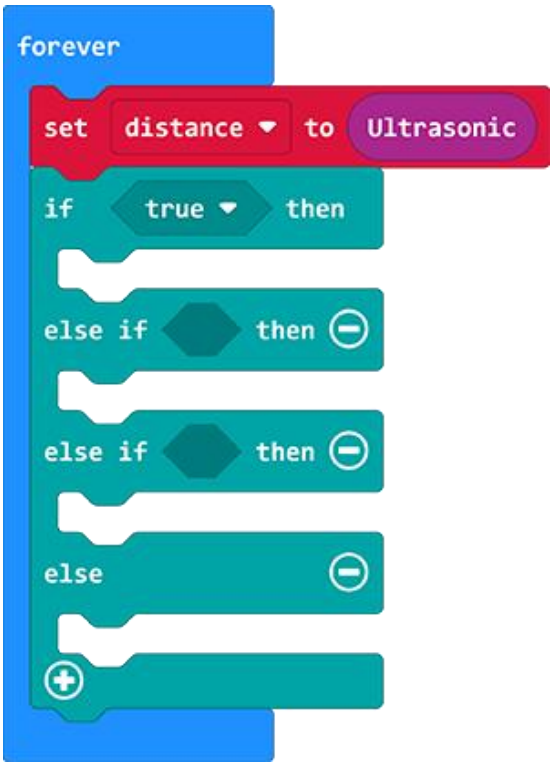


2.In 

, put 

under 

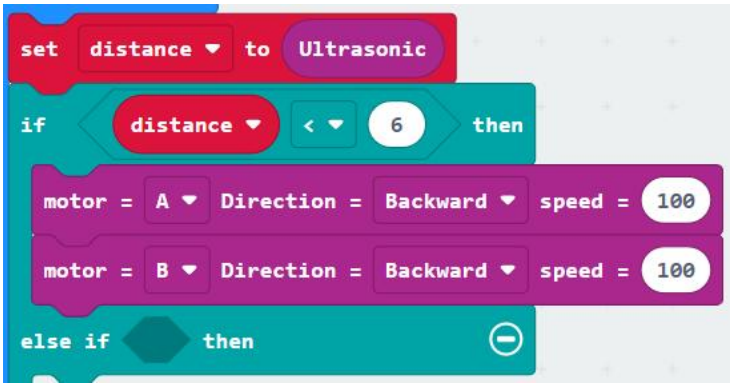
, and click  to add three more “else”.



3.In 

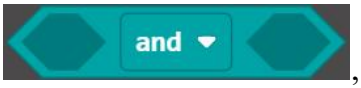
, drag 

, and modify to `distance < 6`. Put it into the first condition box of “if…then” block, and add codes in it to make the car go back.

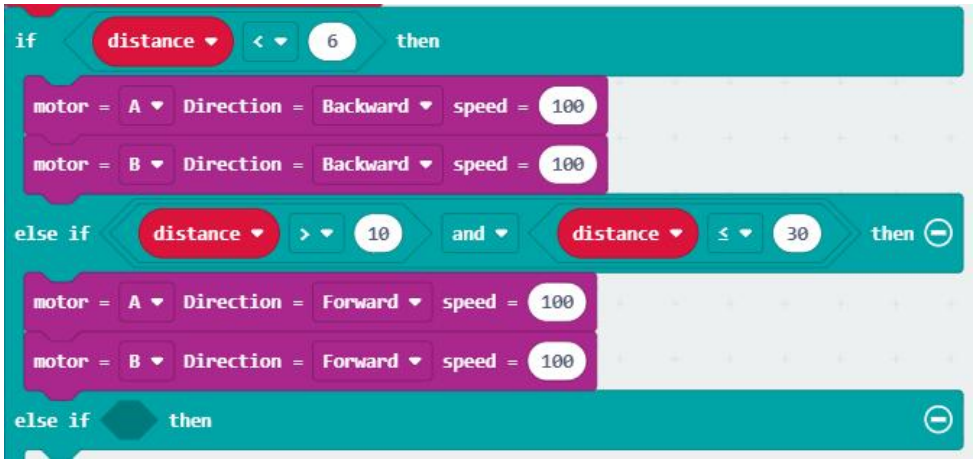


4.In 

, drag 

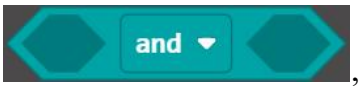
and 

, and modify the condition into: `distance > 10 and distance ≤ 30`. Put it into the second condition box, and add codes to make the car go forward.



5.In 

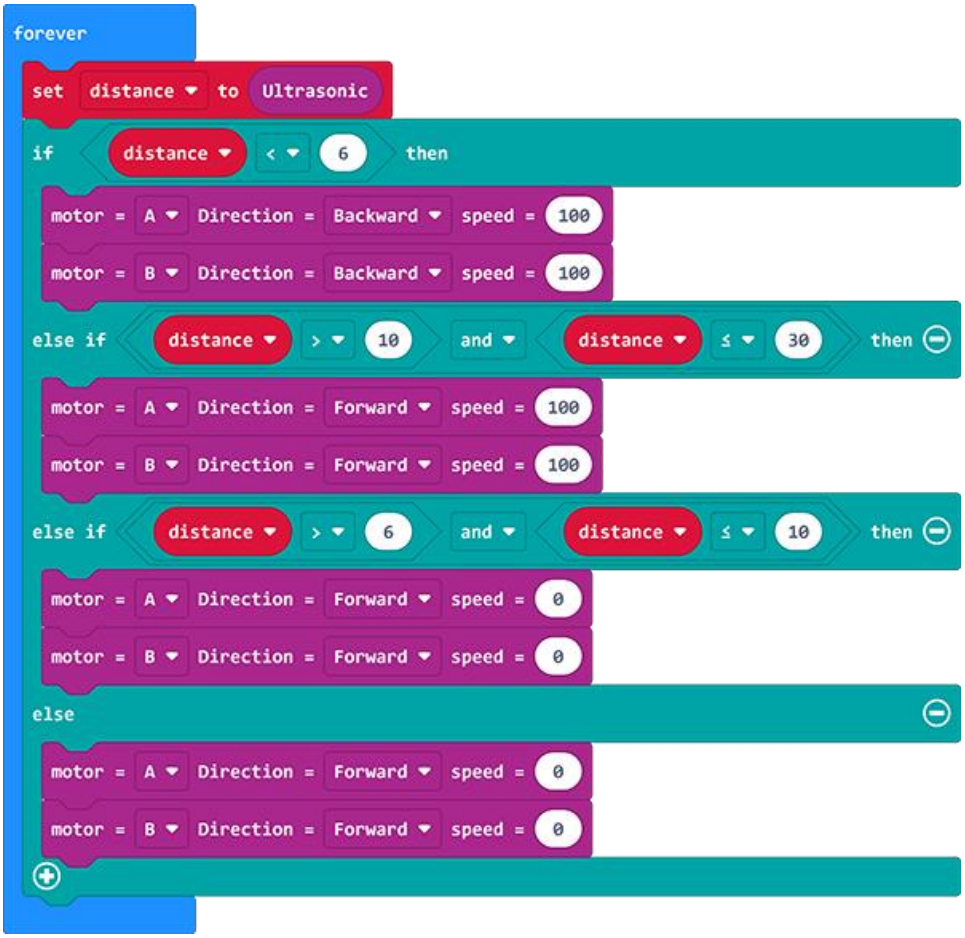
, drag 

and 

, and modify the condition into: `distance > 6 and distance ≤ 10`. Put it into the third condition box, and add codes to make the car stop.

6.Add codes to make the car stop in the last “else”.

Complete Code:



11.4 Test Result

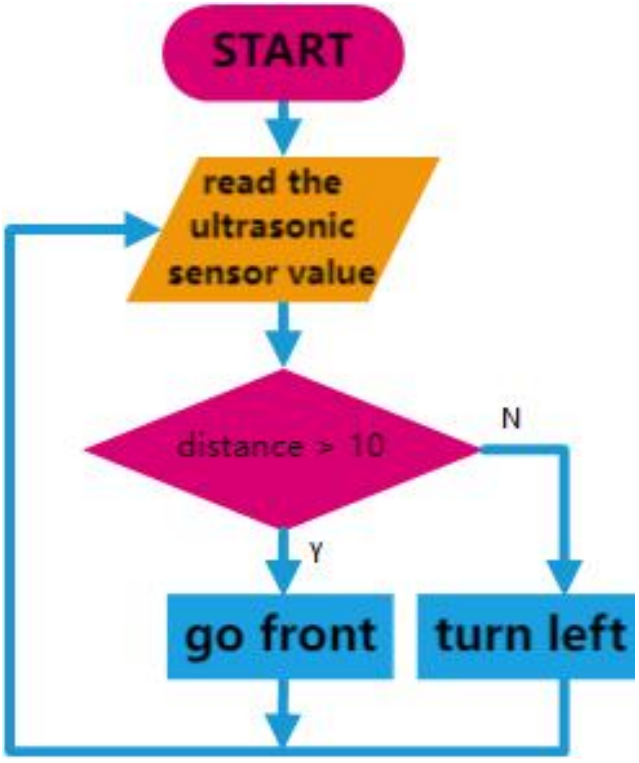
After uploading the code, put your hand in front of the car. When the distance between the car and your hand is less than 6CM, the car goes back. When the distance is within 10CM and 30CM, the car goes forward. If 6CM < distance ≤ 10CM, the car stops. If none of the conditions are satisfied, the car still stops.

Project 12 Ultrasonic Obstacle Avoidance Car

12.1 Introduction

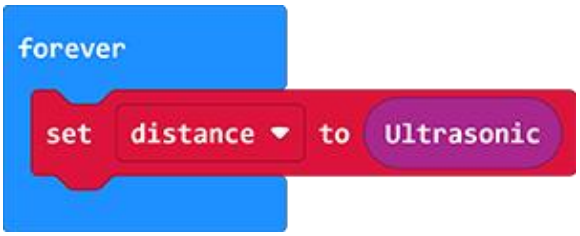
In this project, an ultrasonic sensor is adopted to detect the distance value between the car and obstacles, so that the car can turn left to avoid them if the value is less than the set value.

12.2 Code Flow



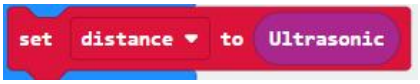
12.3 Test Code

1.Define a variable `distance`, and assign the distance value detected by the ultrasonic sensor to the variable.



2.In 

, drag 

under 

, and click  to add one more “else”.

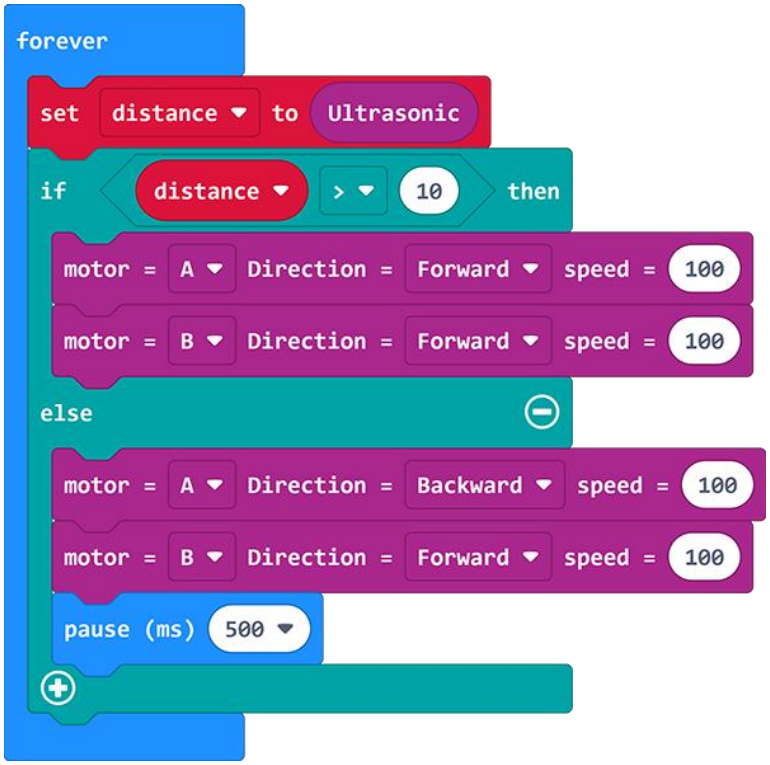
3.In 

, drag 

and modify it into `distance > 10`. Put it into the condition box and add codes to make the car go forward.

4.Add codes to make the car turn left under `else`.

Complete Code:



12.4 Test Result

After uploading code, the ultrasonic sensor detects distance between the car and obstacles. When the value is greater than 10CM, the car goes forward. When it is less than 10CM, the car turns left for 500ms.

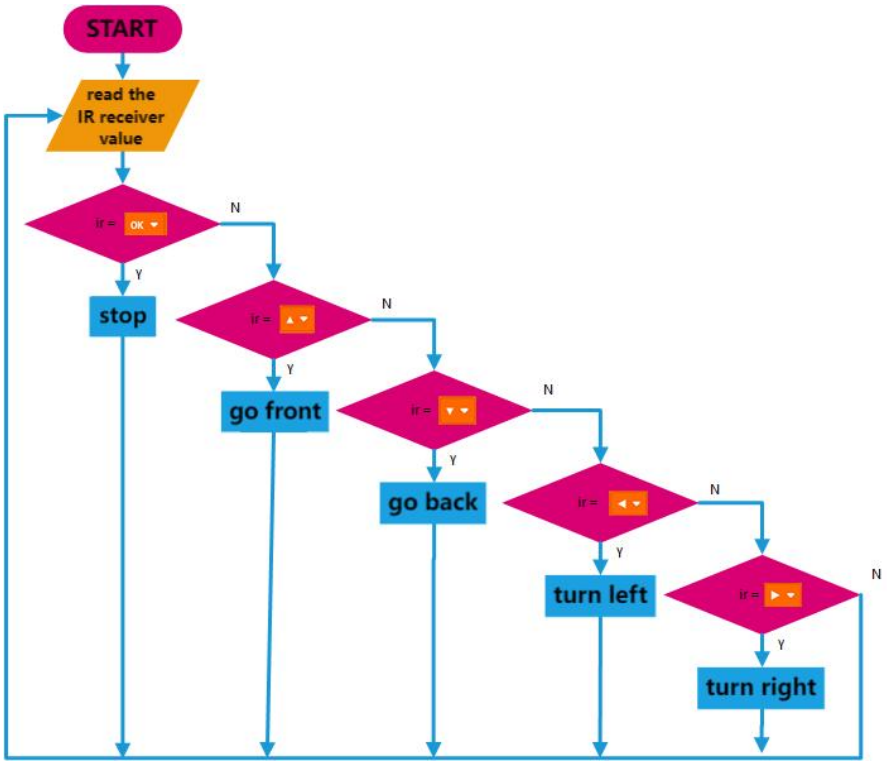
Project 13 IR Remote Control Car

13.1 Introduction

IR remote control car is controlled by infrared signals. A remote control are included in this kit to emit infrared signals, and n infrared receiver is designed on the car to receive signals to direct the movement of the car. During controlling, you only need to press the button on the remote control to make the car go forward, backward, turn or stop.

Besides, it features simple structure and easy control method, so it is suitable for users of all ages to experience interesting and interactive pleasure.

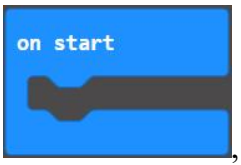
13.2 Code Flow



13.3 Test Code

1.In 

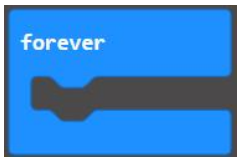
, drag 

into 

, and set the pin to P16.

2.In 

, define a variable named `ir`, and put 

into 

.

3.In 

, drag 

and put it into the value box of 

.

4.In 

, drag 

under 

, and click 
to add five more “else”. For the last `else` without a condition box, click 
to remove it.

5.In 

, drag 

into the first condition box of 

.

6.In 

, put 

into the left box of 

.

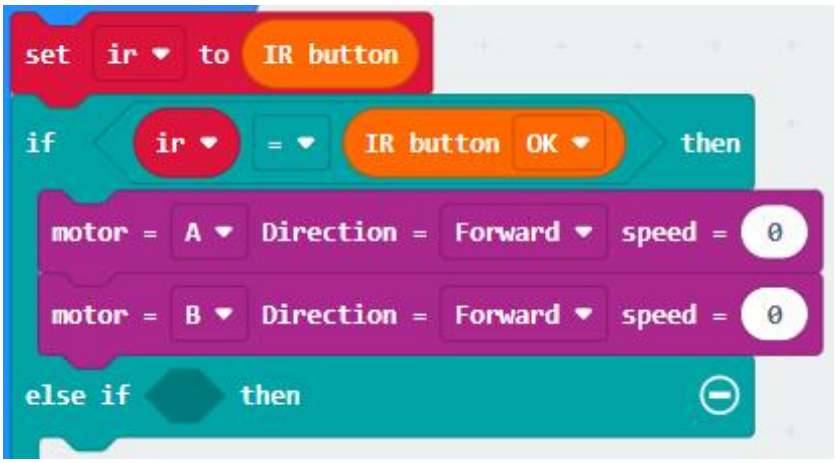
7.In 

, drag 

into the right box of the 

, and modify the button to 

, and add codes to make the car stop under it.



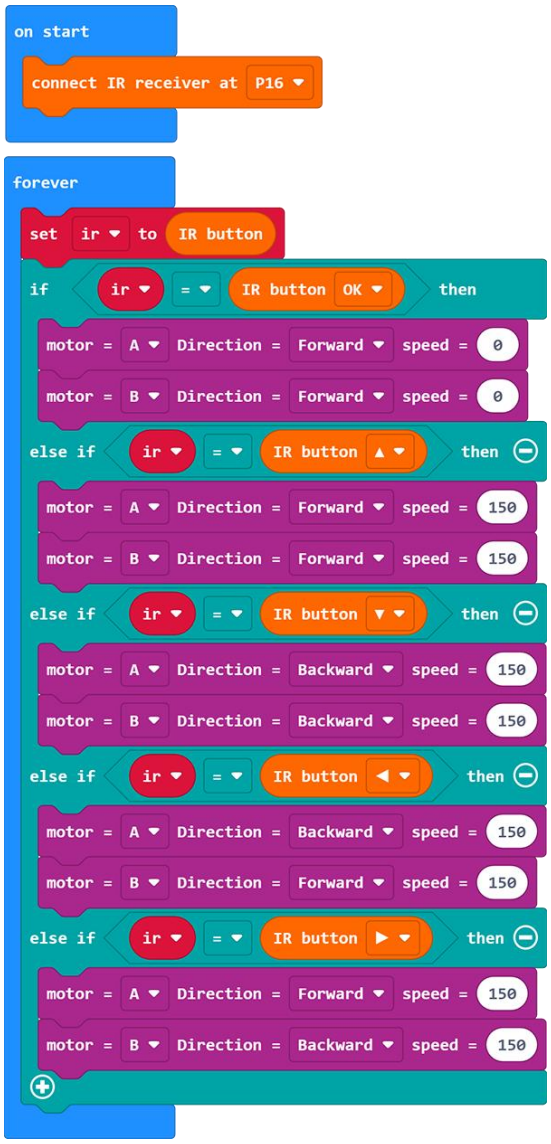
8.Duplicate  and paste it in the next condition box, modify  to  , and add codes to make the car go forward.

9.Duplicate  and paste it in the next condition box, modify  to  , and add codes to make the car go back.

10.Duplicate  and paste it in the next condition box, modify  to  , and add codes to make the car turn left.

11.Duplicate  and paste it in the next condition box, modify  to  , and add codes to make the car turn right.

Complete Code:



13.4 Test Result

After uploading the code, you can control the car with the remote control in the kit.

Press ▲ and the car goes forward; Press ▼ and the car goes back; Press ◀ and the car turns left, Press ▶ and the car turns right; Press OK and the car stops.

Project 14 Multi-function Bluetooth Car

4.1 Introduction

4.2 Working Principle

4.3 Code Blocks

4.4 Test Code

4.5 Test Result

Troubleshooting

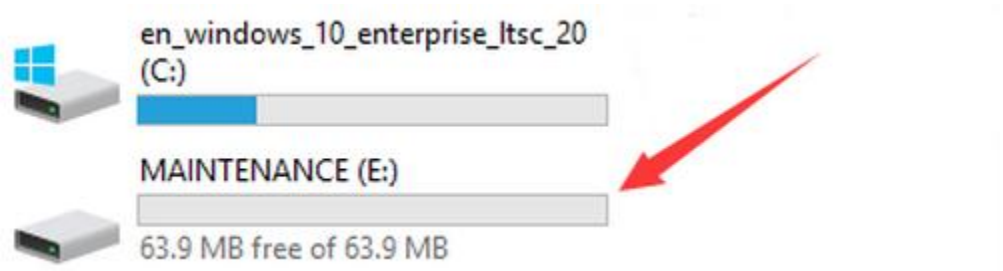
1. Code fails to download to Micro:bit

Problem

Recently, many users encounter the issue that Micro:bit board doesn't respond when download code.

If the way you operate is correct, maybe you accidentally press the reset button and enter the Maintenance mode or the firmware is lost due to mis-operation.

Plug in Micro:bit board, the "MAINTENANCE" drive appears, which means the program can't be downloaded.



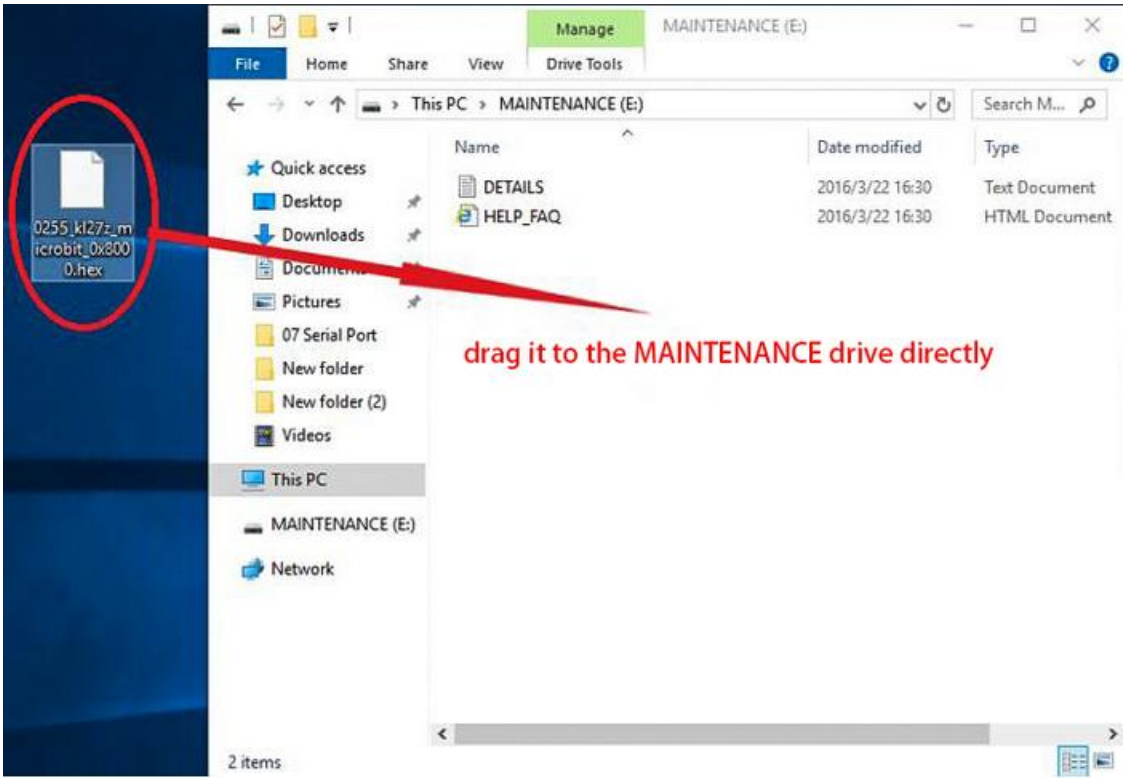
Solution

7. Download the **hex file** from this page to your computer.

   Down load the latest micro:bit firmware-0255: https://www.microbit.org/get-started/user-guide/firmware/

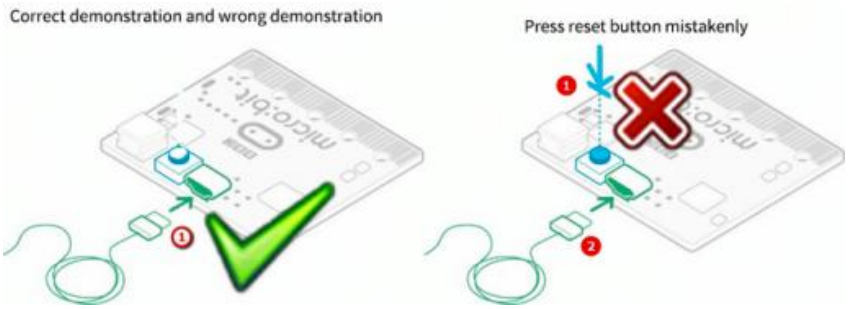
   If you do not want to download from this website, we also provide it in our tutorial.

8. After the latest firmware is downloaded, then drag it into the "MAINTENANCE" to make Micro:bit back to normal mode.



Avoid to Enter “MAINTENANCE”

Make sure the Reset button is **not** pressed when plugging the board by USB cable.



Don't unplug the cable suddenly during downloading micro:bit program, otherwise, the firmware will be lost and micro:bit will enter “MAINTENANCE” mode.

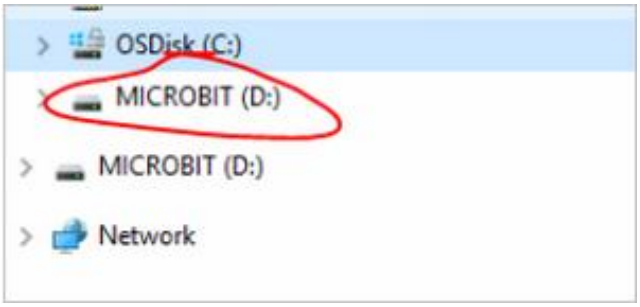
In the experiment, wrong wiring also cause a short circuit or firmware lost.

2.Troubleshooting-Download with WebUSB

Having trouble pairing the Micro: bit with WebUSB (/ device/usb/webusb)?

Step 1: Check cable

Make sure that your micro:bit is connected to your computer with a micro USB cable. You should see a **MICROBIT** drive appear in Windows Explorer when it's connected.



If you can see the MICROBIT, please go to step 2.

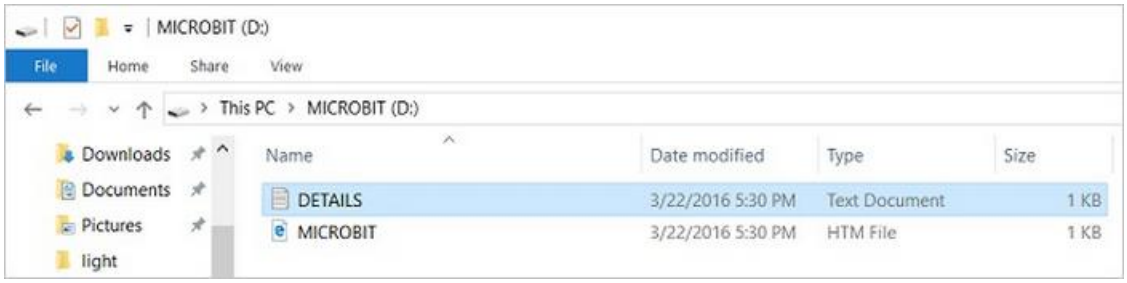
If you can't:

- Make sure that the USB cable is working. Does the cable works on another computer? If not, find a different cable to use. Some cables may only provide a power connection and don't actually transfer data.
- Try another USB port on your computer.
- Is the cable good but you still can't see the **MICROBIT** drive? Hmm, you might have a problem with your micro:bit.
- Try the additional steps described in the falut finding at microbit.org.
- If this doesn't help, you can create a support ticket to notify the Micro:bit Foundation of the problem. If you do so, **skip the rest of these steps**.

Step 2: Check firmware version

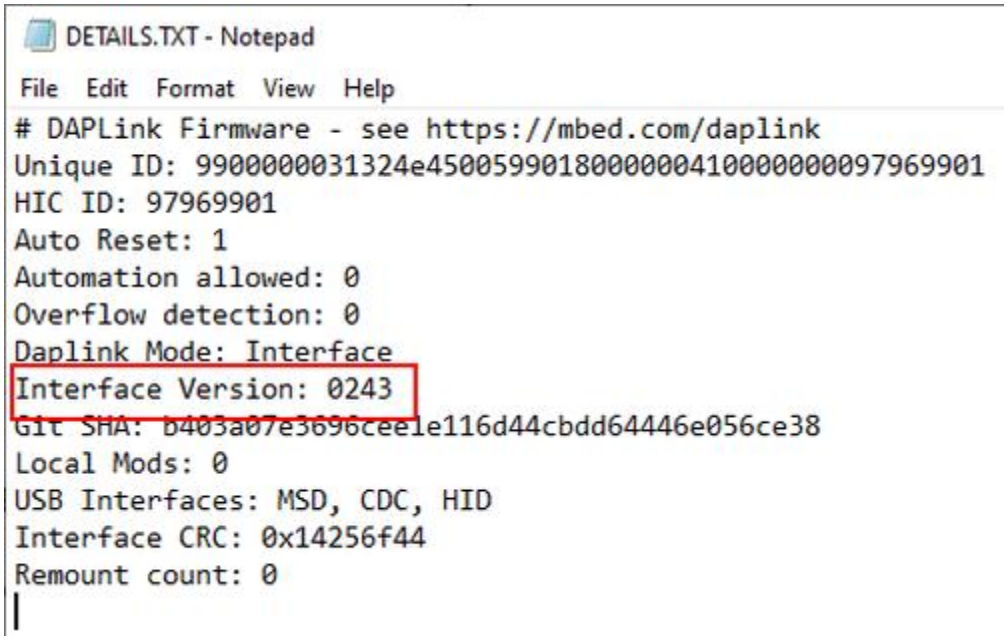
It's possible that the firmware version on the micro:bit needs an update. Let's check:

9. Go to the **MICROBIT** drive.
10. Open the **DETAILS.TXT** file.



11. Look for the version number. It should say **Version: ...**

Or **Interface Version: ...**



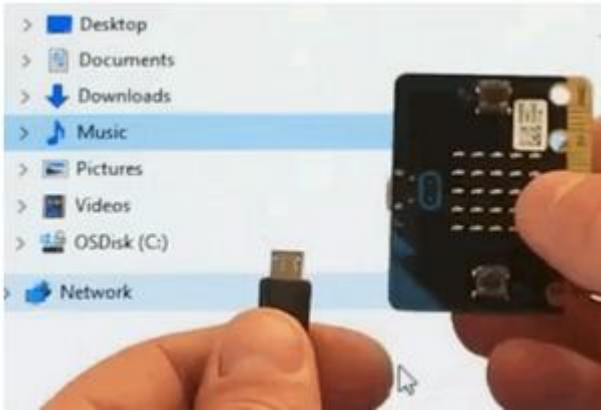
If the version is **0234**, **0241**, **0243**, you need to update the firmware on your micro:bit. Go to **Step 3** and follow the upgrade instructions.

If the version is **0249**, **0250** or higher, you have the right firmware, just go to **step 4**.

Step 3: Update firmware

12. Put your micro:bit into **MAINTENANCE Mode**.

    To do this, please unplug the USB cable from the micro:bit and then re-connect the USB cable after pressing and holding the reset button. Once you insert the cable, you can release the reset button. You should now see **MAINTENANCE** instead of the **MICROBIT** drive. Also, a yellow LED indicator will stay on.



13. Download firmware .hex file: https://microbit.org/guide/firmware/

14. Drag the file into the **MAINTENANCE** drive.

15. The yellow LED will flash while the HEX file is copying. After that, the LED will go off and the micro:bit resets. The **MAINTENANCE** drive now changes to **MICROBIT**.

16. The upgrade is complete! You can open the **DETAILS.TXT** file to check the firmware version that matches the one of the **HEX** file you copied.

If you want to know more about connecting the board, MAINTENANCE Mode, and upgrading the firmware, please refer to Firmware guide.

Step 4: Check version of Browser

WebUSB may require you to update your browser.

Check that your browser version matches one of these: **Android**, **Chrome OS**, **Linux**, **macOS** and **Windows 10 Chrome 65+**.

Step 5: Pair device

Once you've updated the firmware, open the **Chrome Browser**, go to the editor and click on **Pair Device** in settings.

See WebUSB (/ device / usb / webusb) for pairing instructions.