
Mini Tank Robot

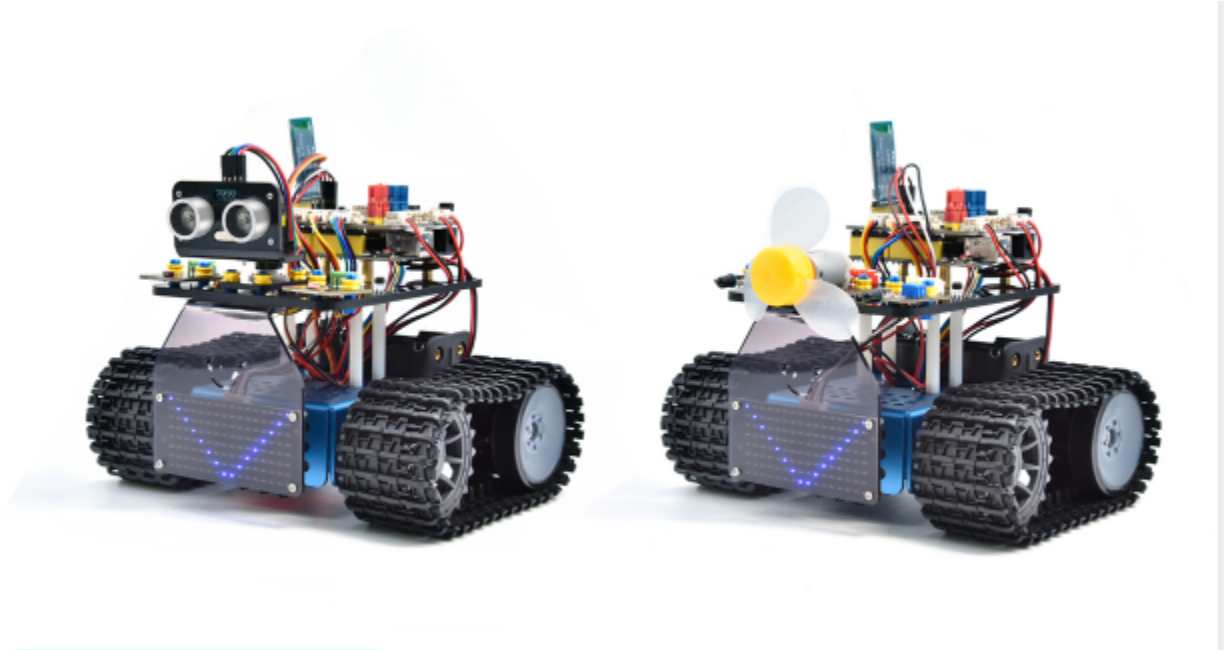
keyestudio

Jan 17, 2024

KS0555 MINI TANK ROBOT V3 TUTORIAL

1	1. Download Resource	3
2	2. Introduction	5
3	3. Features	7
4	4. Parameters	9
5	5. Kit List	11
6	Arduino Tutorial	17
6.1	1. Get started with Arduino	17
6.1.1	1. Keystudio V4.0 Development Board	17
6.1.2	2. Install Arduino IDE and Driver	20
6.2	2. Assemble Ultrasonic Tank Robot	20
6.3	3. Projects	53
6.3.1	Project 1: LED Blinks	53
6.3.2	Project 2: Adjust LED Brightness	58
6.3.3	Project 3: Photoresistor	63
6.3.4	Project 4: Line Tracking Sensor	69
6.3.5	Project 5: Servo Control	75
6.3.6	Project 6: Ultrasonic Sensor	80
6.3.7	Project 7: IR Reception	88
6.3.8	Project 8: Motor Driving and Speed Control	93
6.3.9	Project 9: 8*16 Facial Expression LED Dot Matrix	100
6.3.10	Project 10: Light Following Tank	112
6.3.11	Project 11: Ultrasonic Following Tank	117
6.3.12	Project 12: Ultrasonic Obstacle Avoidance Tank	123
6.3.13	Project 13: Move-in-Confined-Space Tank	128
6.3.14	Project 14: Line-tracking Tank	133
6.3.15	Project 15: IR Remote Control Tank	138
6.3.16	Project 16: Bluetooth Remote Control	146
6.3.17	Project 17: Bluetooth Control Tank	159
6.3.18	Project 18: BT Speed Control Robot	170
6.3.19	Project 19: Ultrasonic Tank Robot Multiple Functions	176
6.4	4. Experiment Extension	191
6.4.1	Assemble Fire Extinguishing Robot	191
6.4.2	Project 20: Flame Sensor	196
6.4.3	Project 21: Fan	202
6.4.4	Project 22: Fire Extinguishing Tank	206
6.4.5	Project 23: Fire Extinguishing Robot Multiple Functions	211

7	Kidsblock Tutorial	227
7.1	1. Getting started with kidsblock	227
7.1.1	1. Instruction:	227
7.1.2	2. Download and install KidsBlock software:	227
7.1.3	3. Choose TankRobot device:	227
7.2	2. Assemble Ultrasonic Tank Robot	234
7.3	3. Projects	269
7.3.1	Project 1: LED Blinks	269
7.3.2	Project 2: Adjust LED Brightness	274
7.3.3	Project 3: Photoresistor	281
7.3.4	Project 4: Line Tracking Sensor	289
7.3.5	Project 5: Servo Control	300
7.3.6	Project 6: Ultrasonic Sensor	305
7.3.7	Project 7: IR Reception	316
7.3.8	Project 8: Motor Driving and Speed Control	326
7.3.9	Project 9: 8*16 Facial Expression LED Dot Matrix	332
7.3.10	Project 10: Light-following Tank	345
7.3.11	Project 11: Ultrasonic Sound-following Tank	351
7.3.12	Project 12: Ultrasonic Obstacle Avoidance Tank	358
7.3.13	Project 13: Move-in-Confined-Space Tank	366
7.3.14	Project 14:Line-tracking Tank	374
7.3.15	Project 15: IR Remote Control Tank	381
7.3.16	Project 16: Bluetooth Remote Control	388
7.3.17	Project 17: Bluetooth Control Tank	405
7.3.18	Project 18: Ultrasonic Tank Robot Multiple Functions	415
7.4	4. Experiment Extension	419
7.4.1	Assemble Fire Extinguishing Robot	419
7.4.2	Project 19: Flame Sensor	424
7.4.3	Project 20: Fan	433
7.4.4	Project 21: Fire Extinguishing Tank	441
7.4.5	Project 22: Fire Extinguishing Robot Multiple Functions	447



1. DOWNLOAD RESOURCE

Download libraries and codes:

- `libraries_and_codes`
- `Preparation_and_Troubleshooting`

2. INTRODUCTION

This STEM educational V3.0 tank robot is newly upgraded, adding an line-tracking and a fire- extinguishing function. It vigorously enhances the relationship between kids and parents, and sparks children's imagination through programming and coding.

In the course of assembly process, you can see its multiple functions like light following, line tracking, IR and BT remote control, speed adjustment and so on. Additionally, there are some small parts that can help you assemble the robot car.

There are basic sensors and modules, such as a flame sensor, a BT sensor, an obstacle avoidance sensor, an line tracking sensor and an ultrasonic sensor are included.

The two tutorials for C language code of Arduino IDE and KidsBlock graphical programming are also suitable for the enthusiasts at different ages.

It is really the best choice for you.

3. FEATURES

1. Multiple functions Confinement, line tracking, fire extinguishing, light following, IR and BT remote control, speed control and so on.
2. Easy to build: assemble the robot with some parts.
3. High tenacity: Aluminum alloy brackets, metal motors, high quality wheels
4. High extension: connect many sensors and modules through motor driver shield and LEGO parts
5. Multiple controls: IR remote control, App control(iOS and Android system)
6. Basic programming C language code of Arduino IDE and KidsBlock graphical programming.

4. PARAMETERS

- Working voltage: 5v
- Input voltage: 6-9V
- Maximum output current: 1.5A
- Maximum power dissipation: 32W
- Motor speed: 5V 200 rpm / min
- Motor drive mode: dual H bridge drive(HR8833)
- Ultrasonic induction angle: $<15^{\circ}$
- Ultrasonic detection distance: 2cm-300cm
- Infrared remote control distance: 10 meters (measured)
- BT remote control distance: 30 meters (measured)

5. KIT LIST

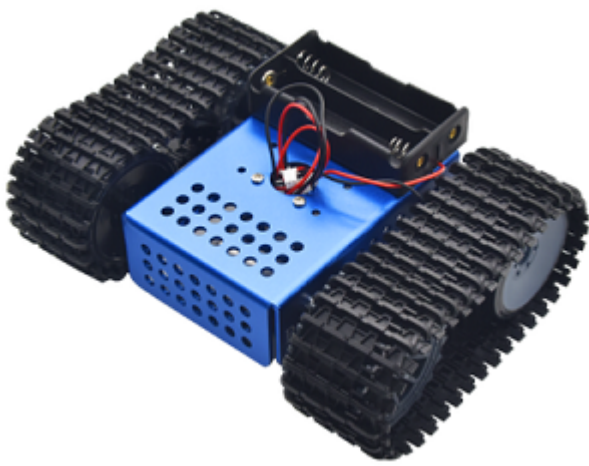



1		Tank Robot Chassis
2		Keyestudio V4.0 development board (compatible with Arduino Uno)
3		Keyestudio 8833 Motor Driver Expansion Board
4		DX-BT24 V5.1 BLE BT Module

Table 1 – continued from previous page

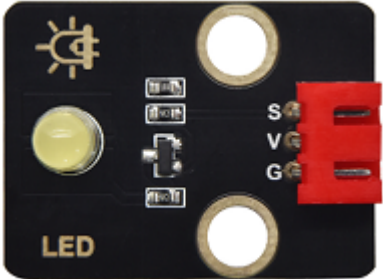
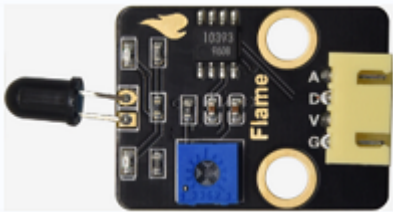
5		HC-SR04 Ultrasonic Sensor
6		Keyestudio 8*16 LED Panel
7		Yellow LED Module
8		Flame Sensor
9		130 Motor Module
10		Photoresistor

Table 1 – continued from previous page









11		Acrylic Board for 8*16 LED Panel
12		Top Acrylic Board
13		Acrylic Board
14		Keyestudio JMFP-4 17-Key Remote Control (Batt
15		Keyestudio 9G 180 ° Servo
16		USB Cable
17		Winding Pipe
18		3.0*40MM Screwdriver

Table 1 – continued from previous page




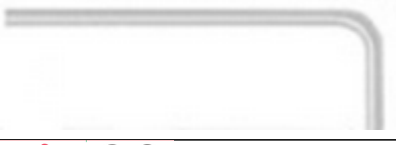





19		3*100MM Ties
20		L Type M2.5 Wrench
21		L Type M3 Wrench
22		L Type M1.5 Wrench
23		Cardboard
24		4P M-F PH2.0mm to 2.54 Dupont Wire (Green-B
25		4P HX-2.54 Dupont Wire (Black-Red-White-Brow
26		5P JST-PH2.0MM Dupont Wire
27		3P-3P XH2.54 to 2.54 Dupont Wire Yellow-Red-B
28		3P-3P XH2.54 to PH2.0 Dupont Wire Yellow-Red-

Table 1 – continued from previous page


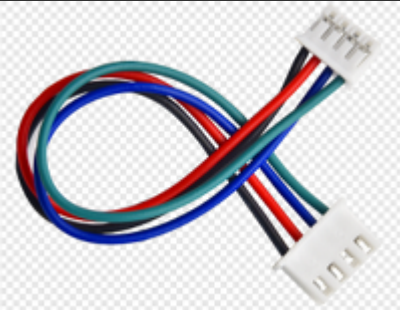



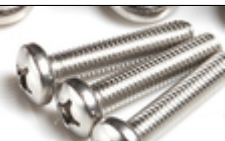
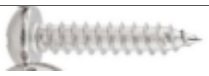








29		4P-3P XH2.54 to PH2.0 Dupont Wire Yellow-Red
30		4P XH2.54 to PH2.0 Dupont Wire Green-Blue-Red
31		M1.4*8MM Round-head Screws
32		M1.4 Nuts
33		M2 Nuts
34		M2*8MM Round-head Screws
35		M1.2*5MM Round-head Screws
36		M3*6MM Round-head Screws
37		M3*10MM Round-head Screws
38		M3 Nuts
39		M3*10MM Dual-pass Copper Pillar

Table 1 – continued from previous page

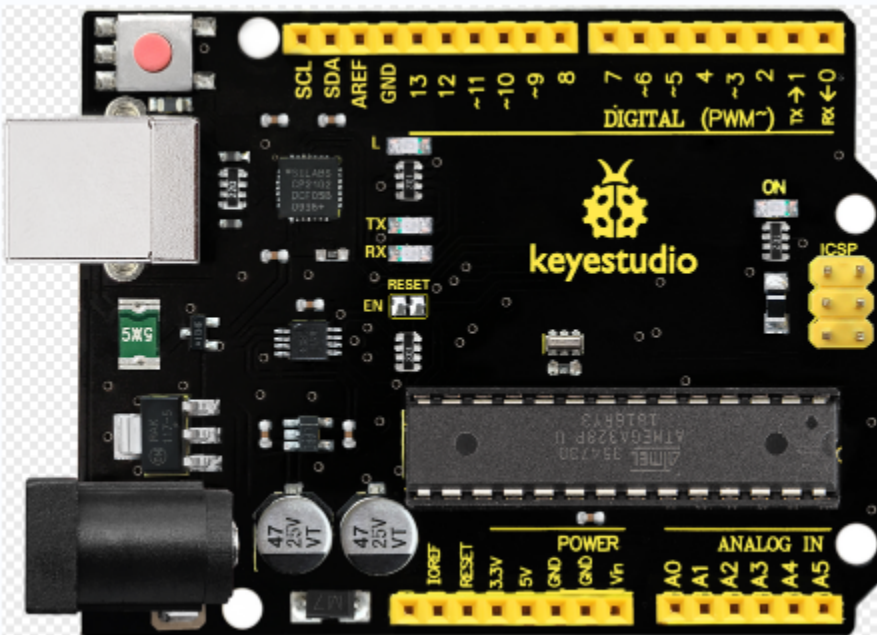
40		M3*40MM Dual-pass Copper Pillar
41		43093 Blue Technic Axle Pin with Friction Ridges
42		4265c Technic Bush
43		Blue Jumper Cap
44		Red Jumper Cap

ARDUINO TUTORIAL

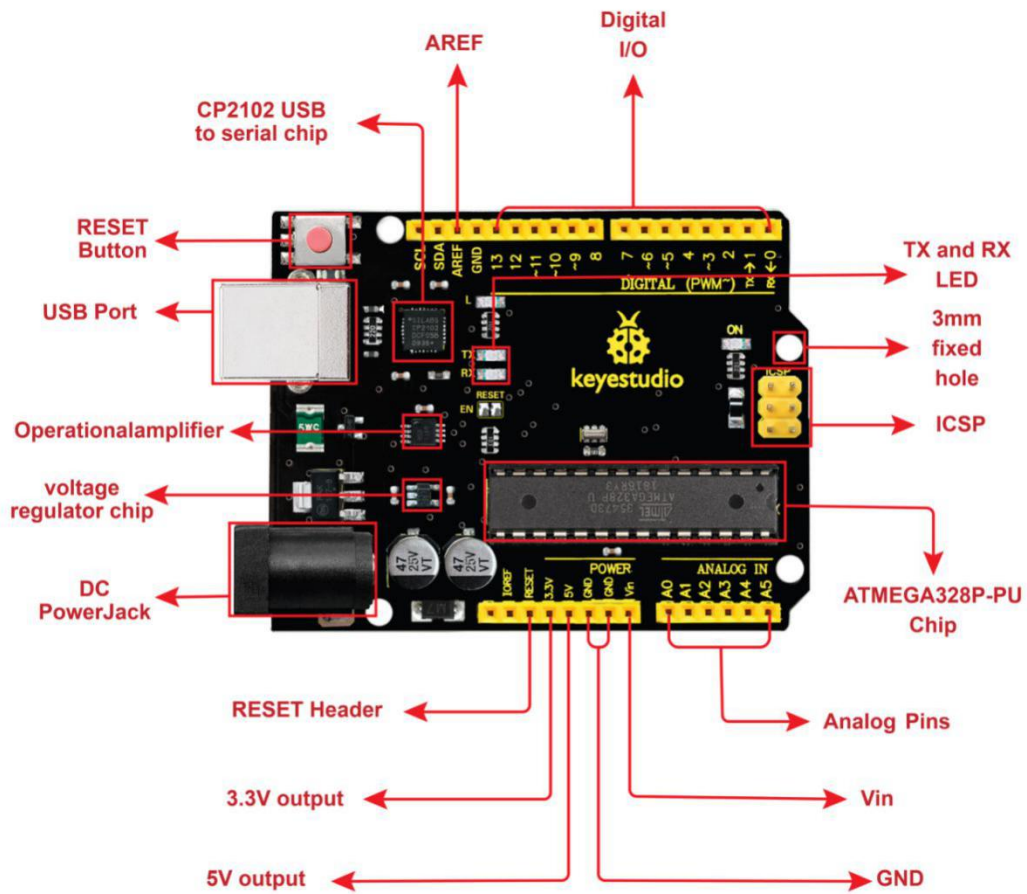
6.1 1. Get started with Arduino

6.1.1 1. Keystudio V4.0 Development Board

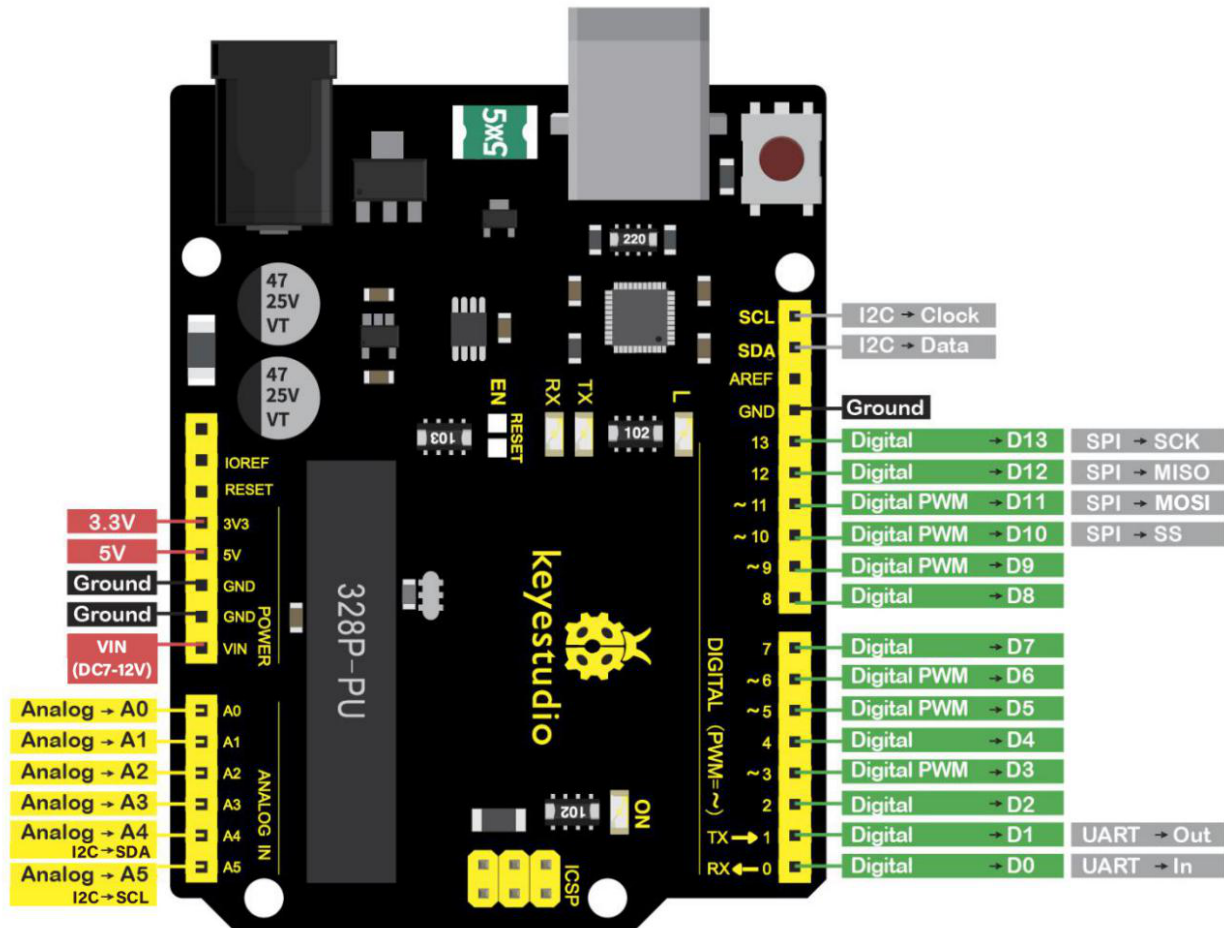
You need to know that keystudio V4.0 development board is the core of this smart car.



Keystudio V4.0 development board is based on ATmega328P MCU, and with a CP2102 Chip as a UART-to-USB converter.



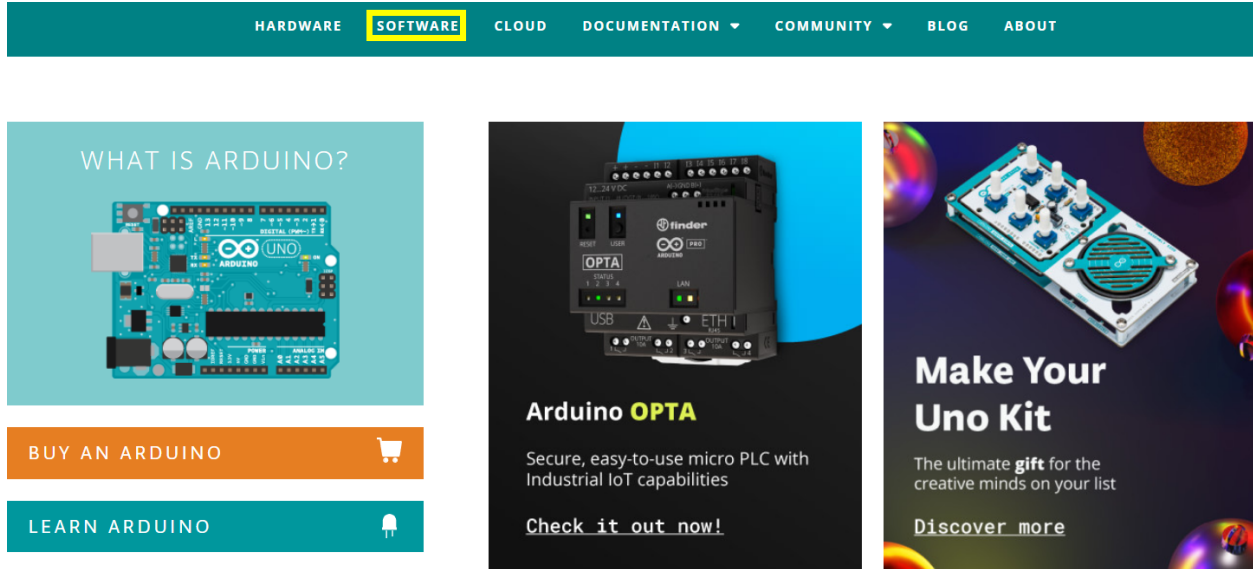
It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, 2 ICSP headers and a reset button.



We can power it with a USB cable, the external DC power jack (DC 7-12V) or female headers Vin/ GND(DC 7-12V)

Micro controller	ATmega328P-PU
Operating Voltage	5V
Input Voltage (recommended)	DC7-12V
Digital I/O Pins	14 (D0-D13) (of which 6 provide PWM output)
PWM Digital I/O Pins	6 (D3, D5, D6, D9, D10, D11)
Analog Input Pins	6 (A0-A5)
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P-PU) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P-PU)
EEPROM	1 KB (ATmega328P-PU)
Clock Speed	16 MHz
LED_BUILTIN	D13

6.1.2 2. Install Arduino IDE and Driver



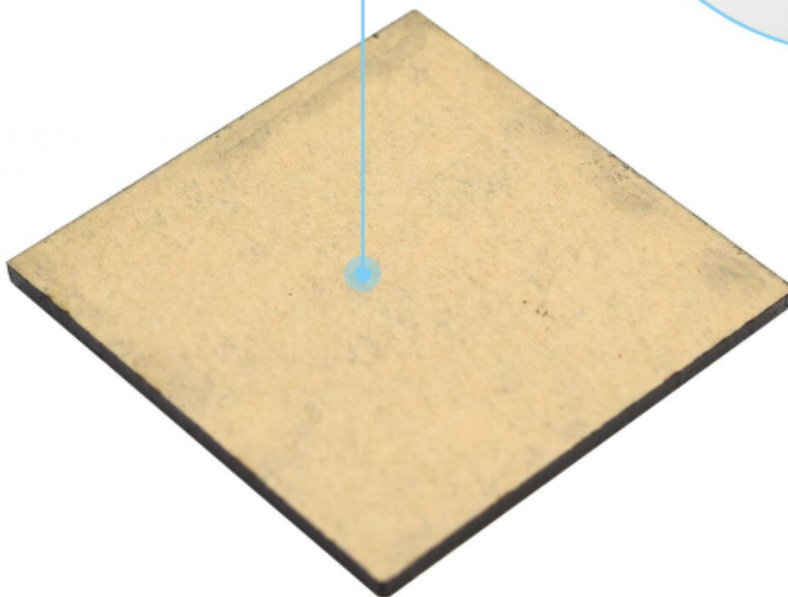
Note: Click the link to start learning how to download software, install drivers, upload code, and install library files.

<https://getting-started-with-arduino.readthedocs.io>

6.2 2. Assemble Ultrasonic Tank Robot

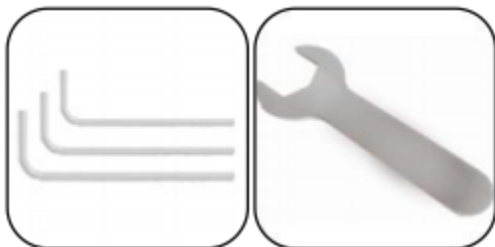
Caution: Set the initial angle of the servo Peel thin films off boards before installing this robot

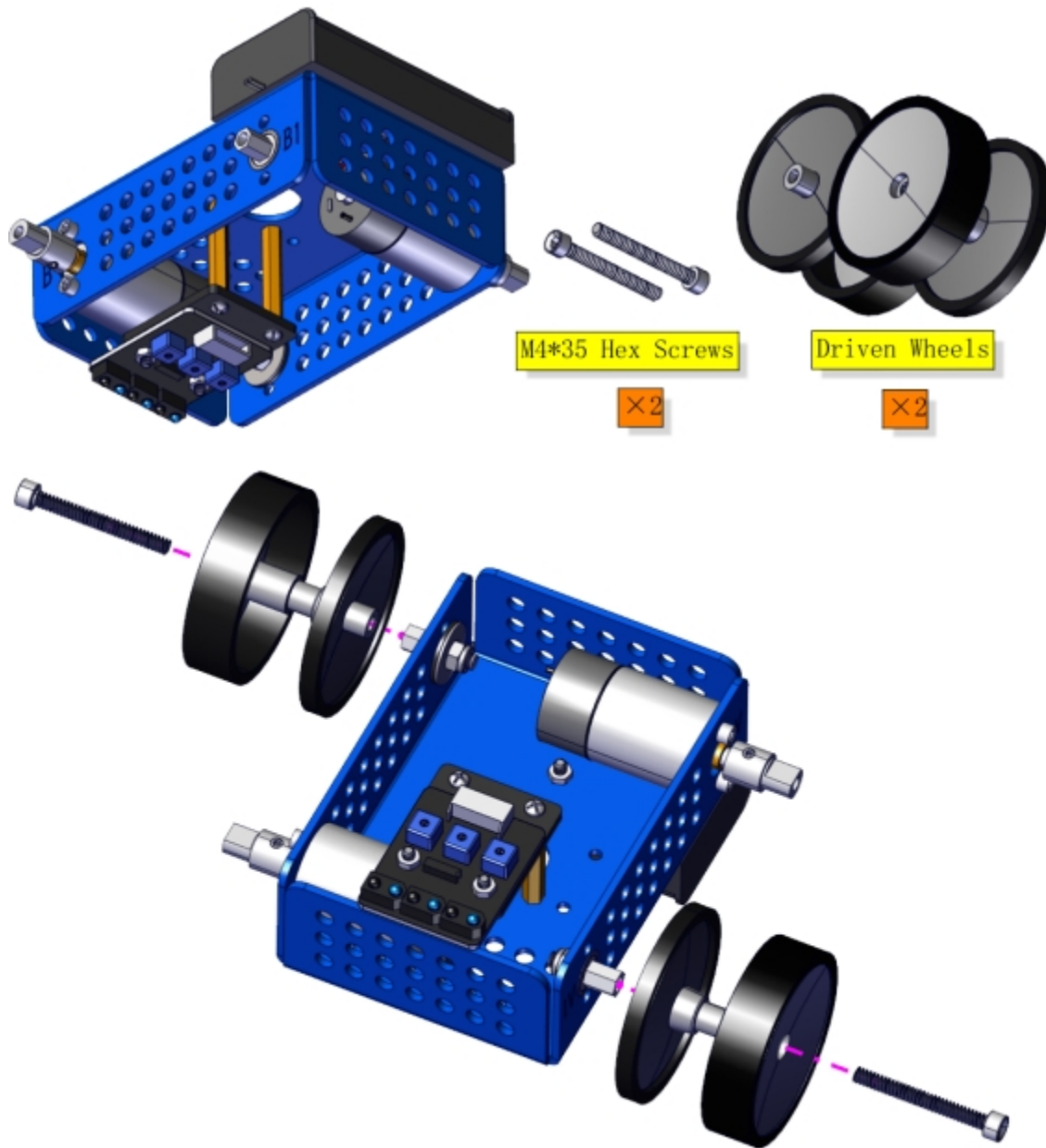
**Before assembly, please
tear off the protective
film on the acrylic.**



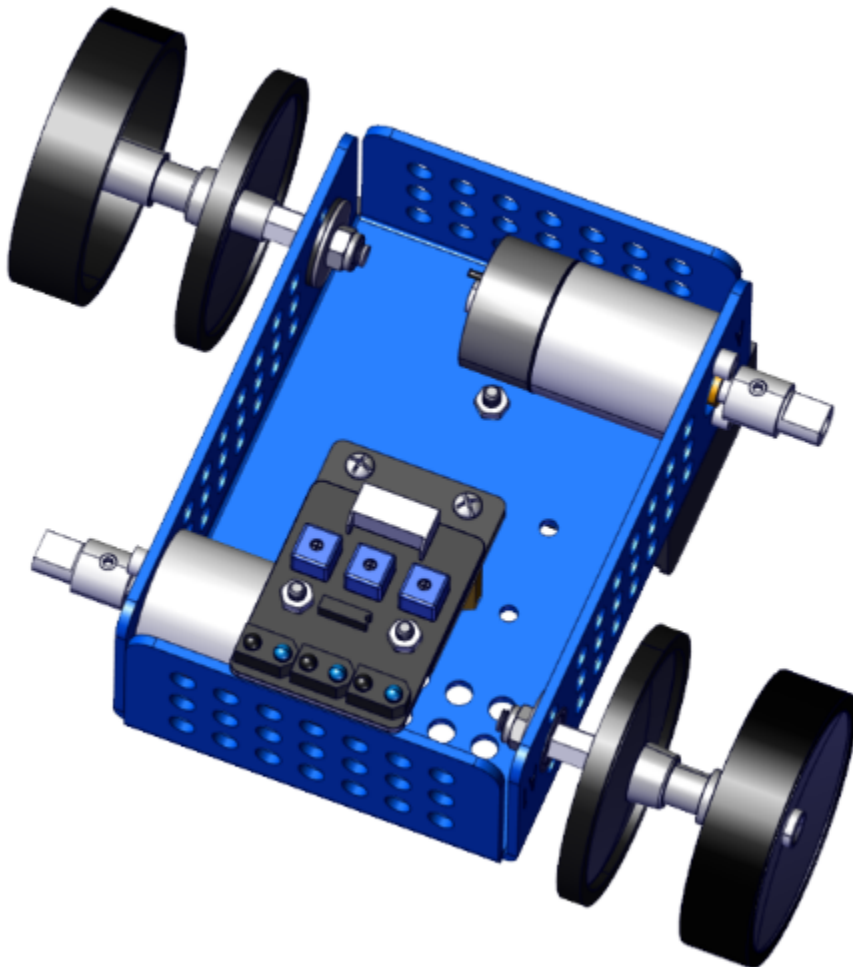
Step1

Tools needed:

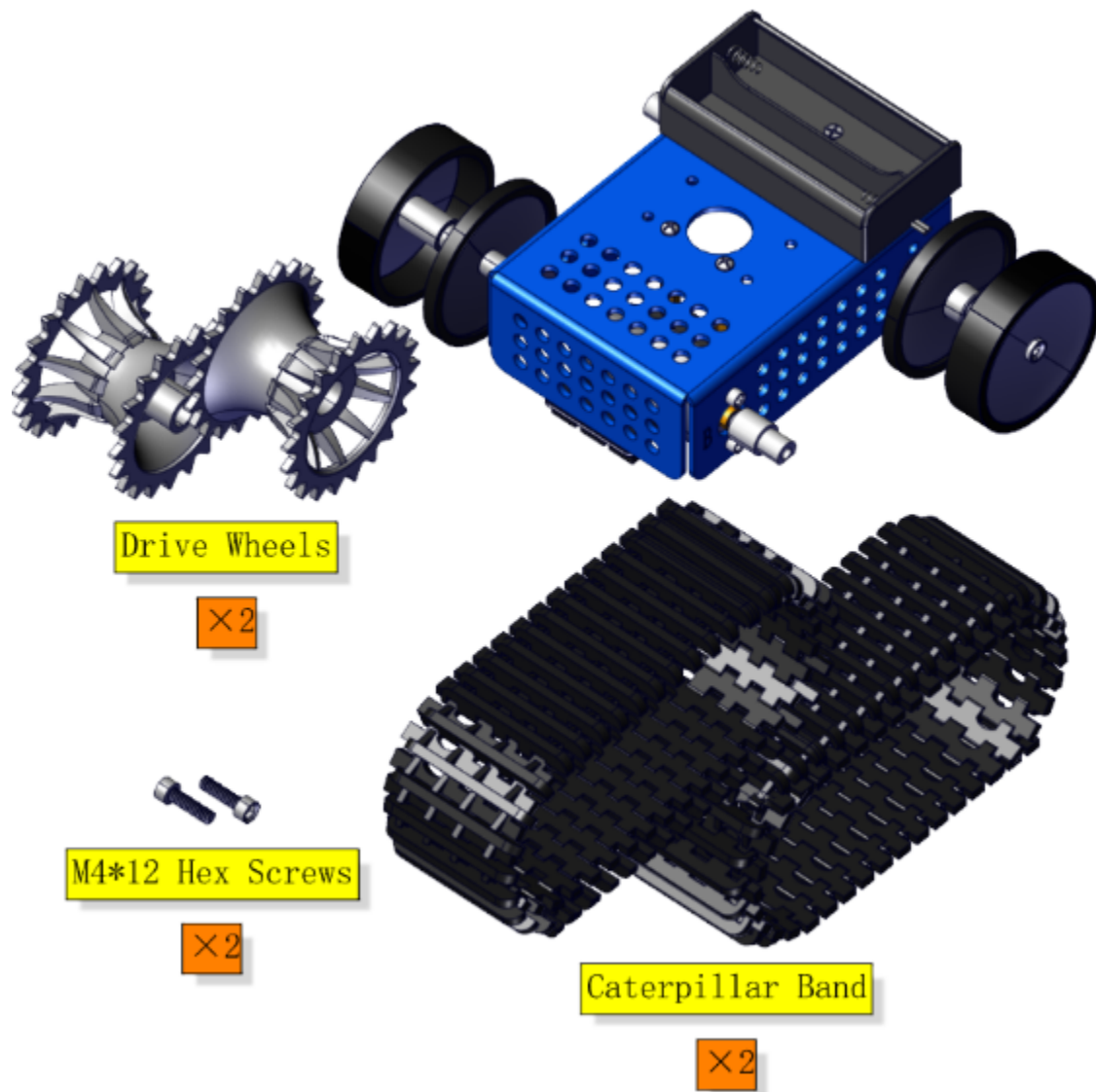


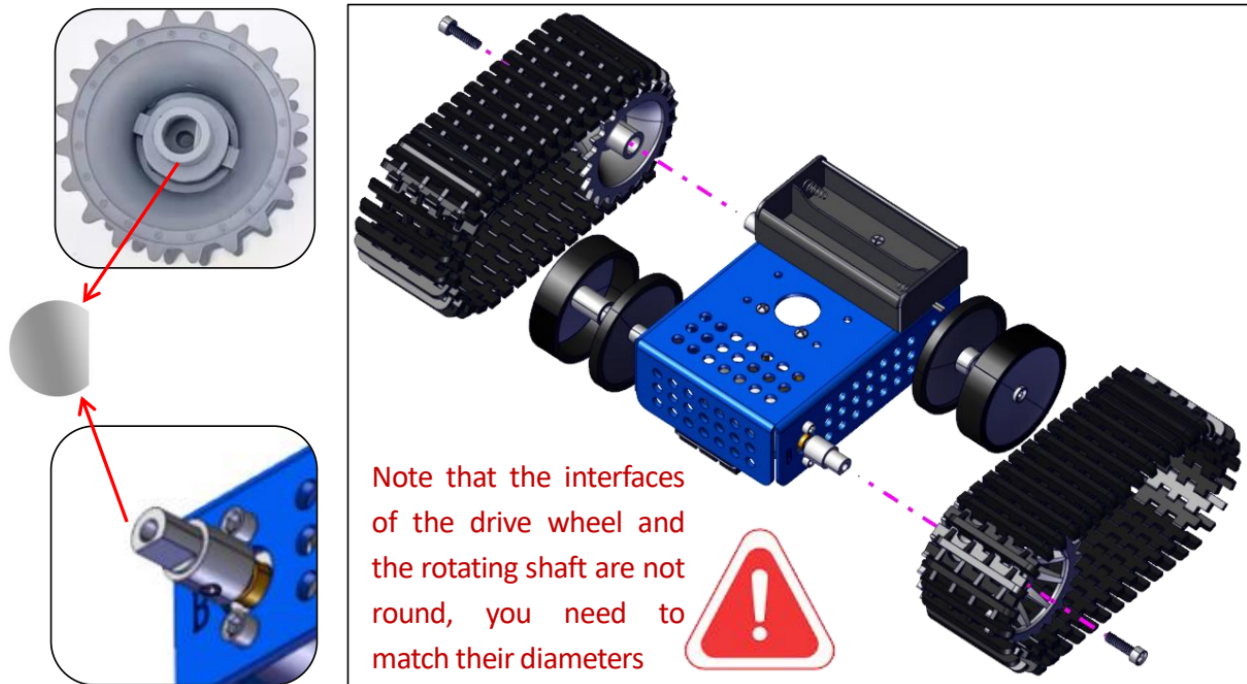


Pay attention to the installation direction of the wheels. The thick side is on the outside.

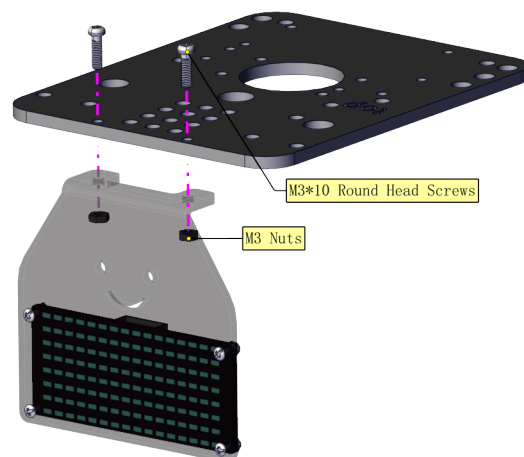
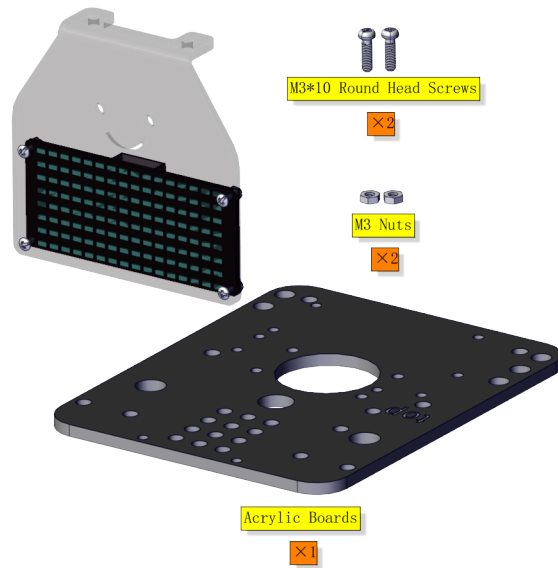


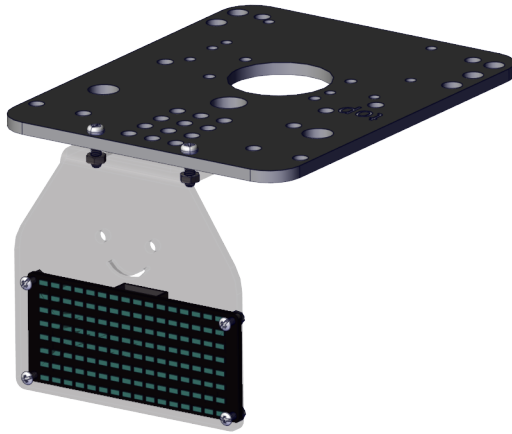
Step2



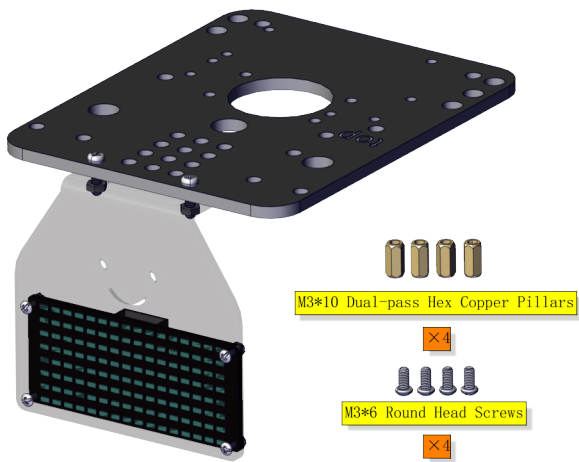


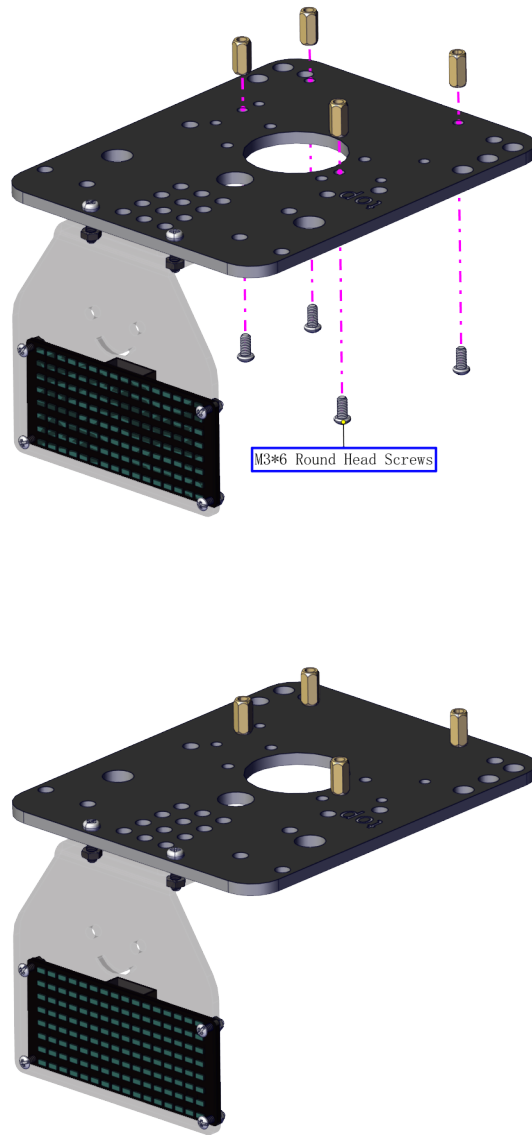
Step3



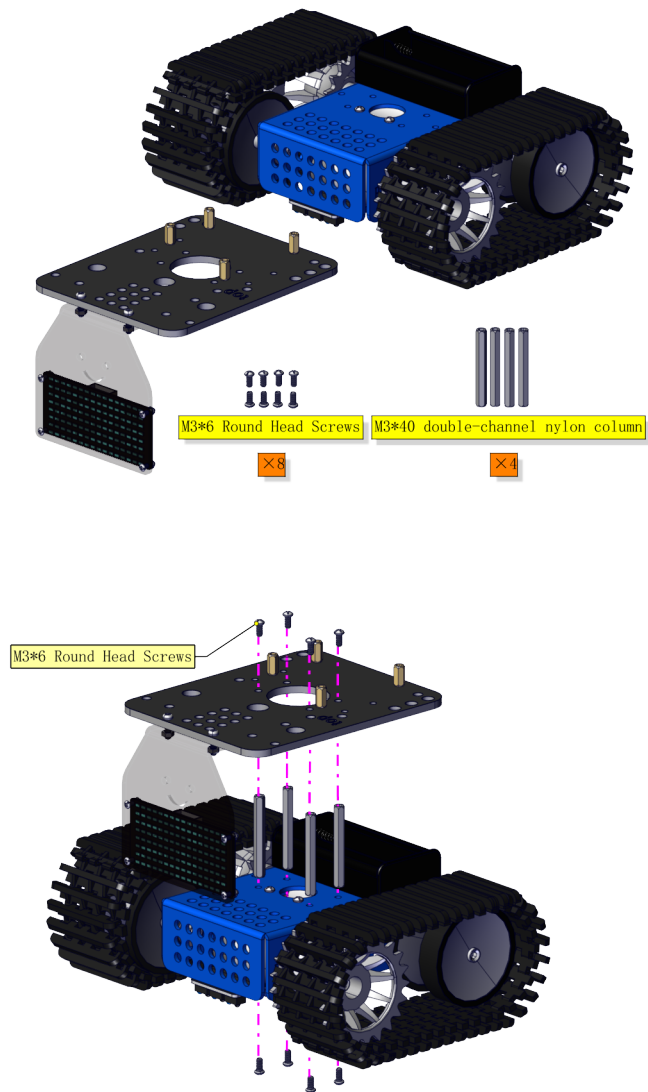


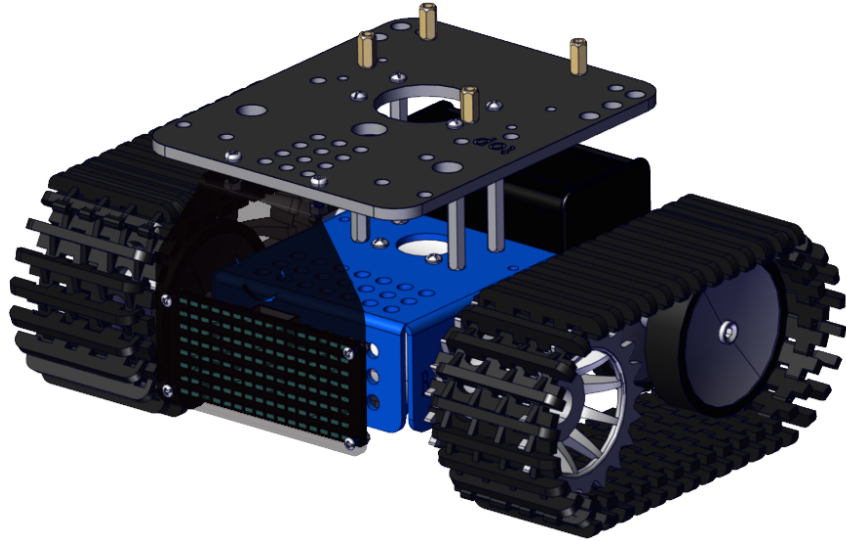
Step 4



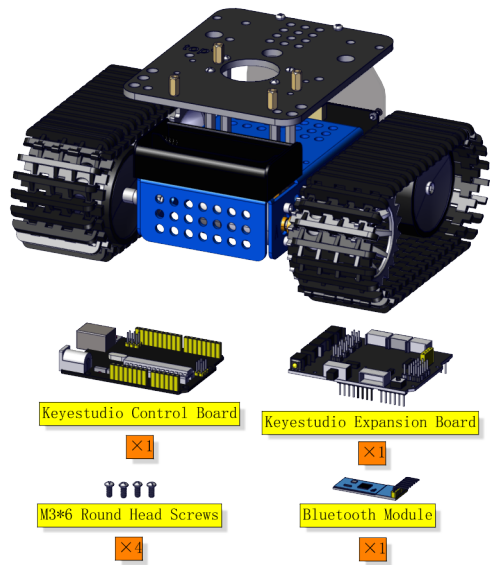


Step 5

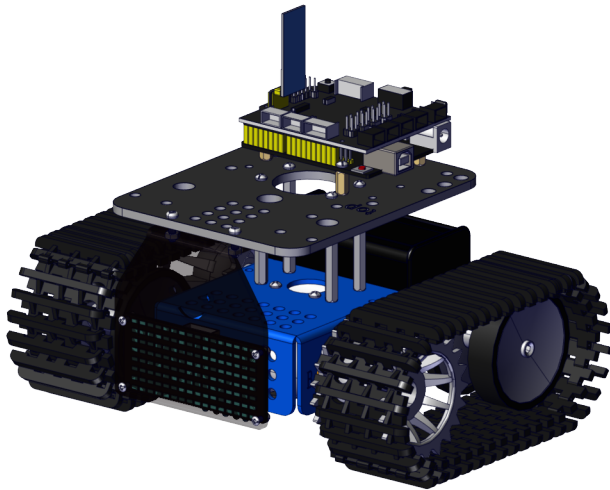
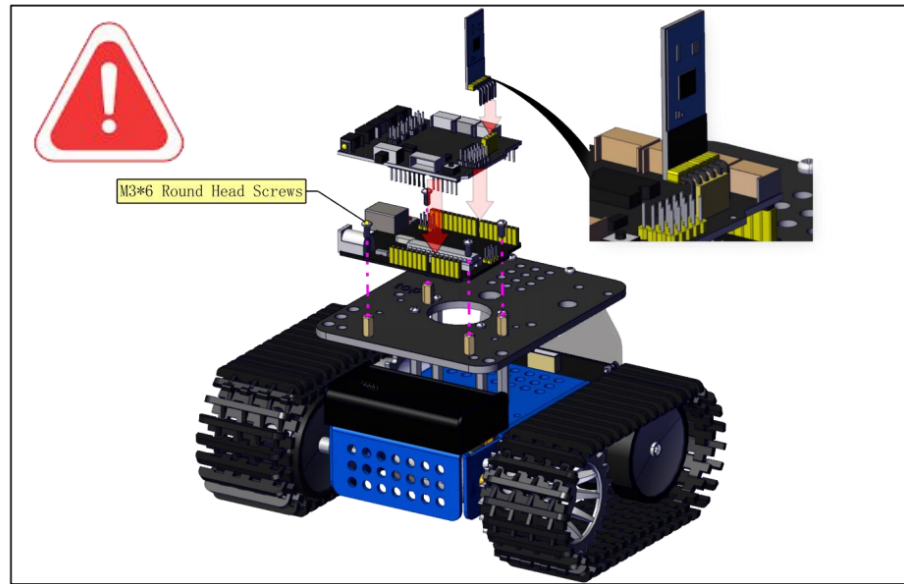




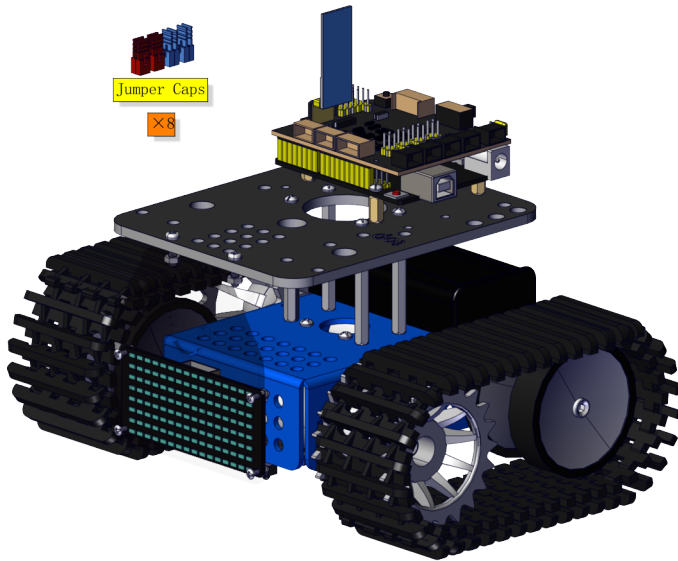
Step 6



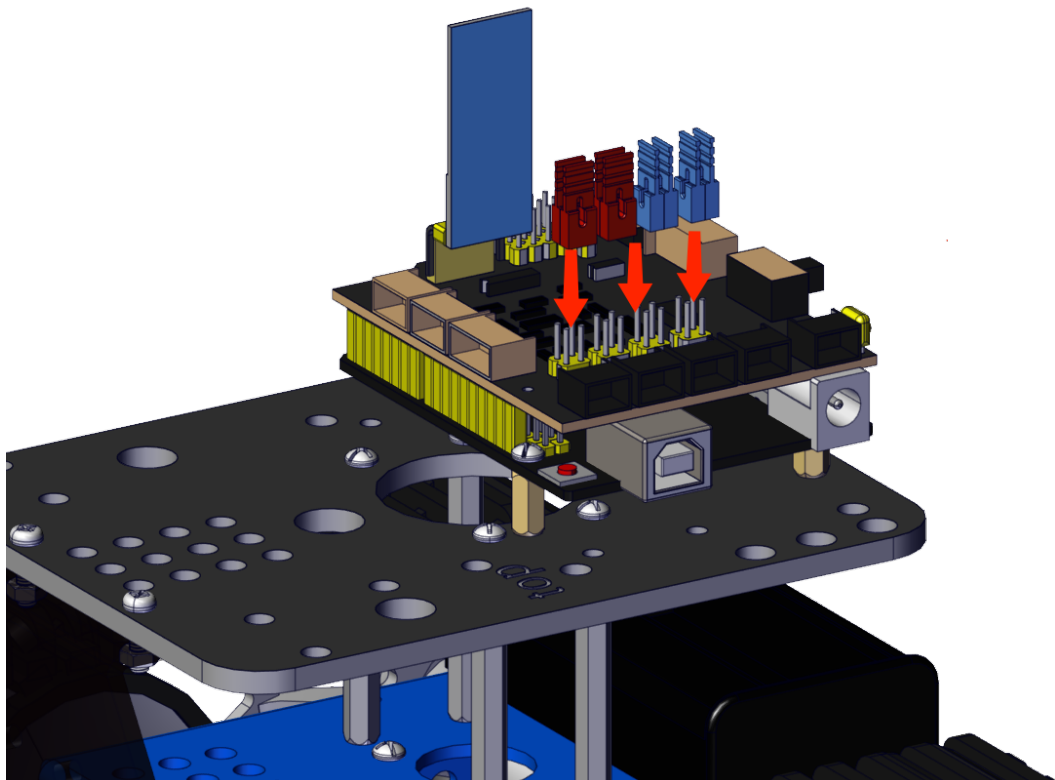
Note: The Bluetooth module will occupy the serial port. Please do not connect to Bluetooth when uploading code.

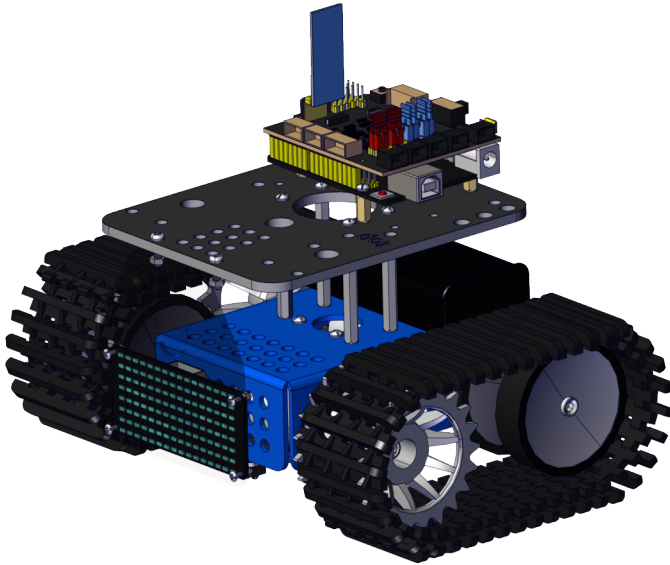


Step 7

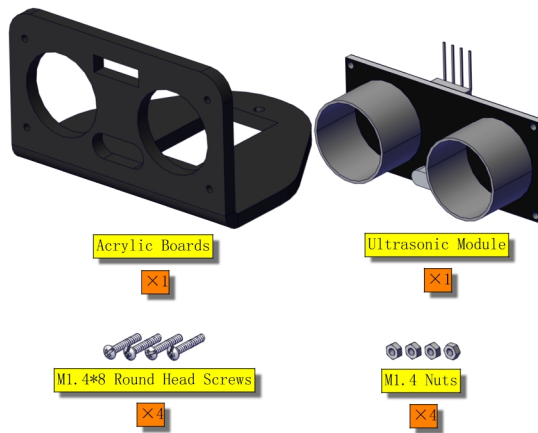


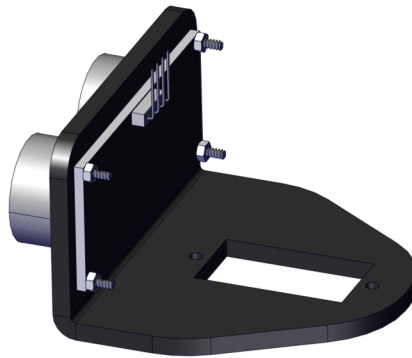
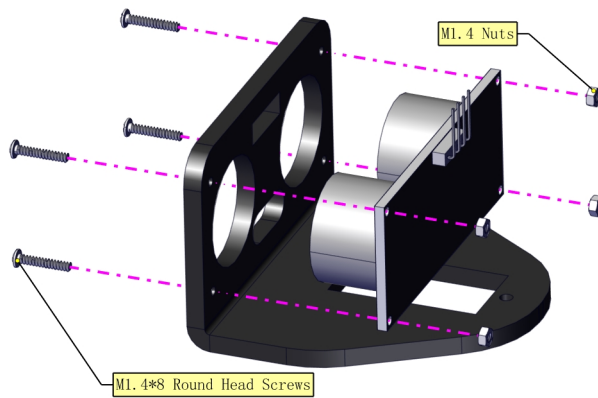
Note the direction of jumper caps



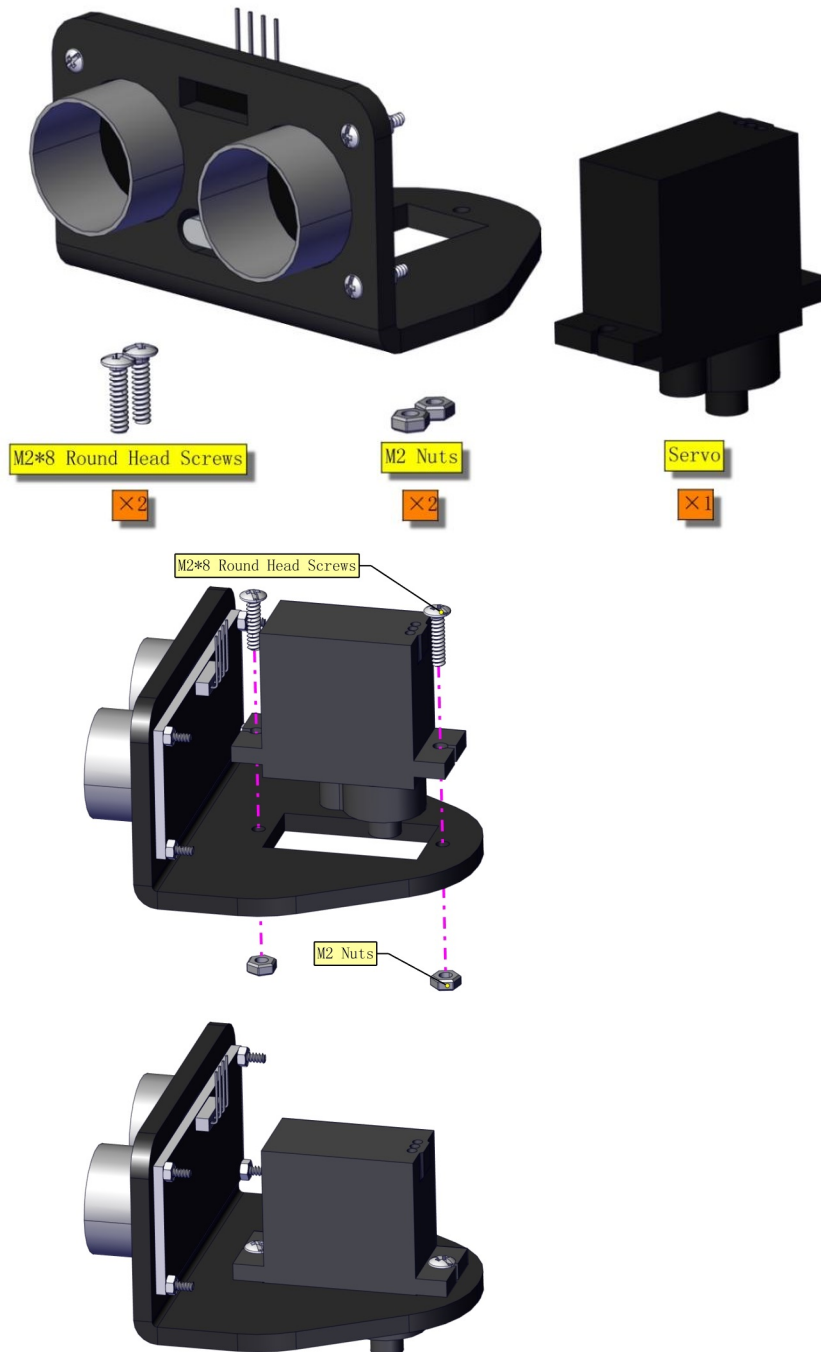


Step 8

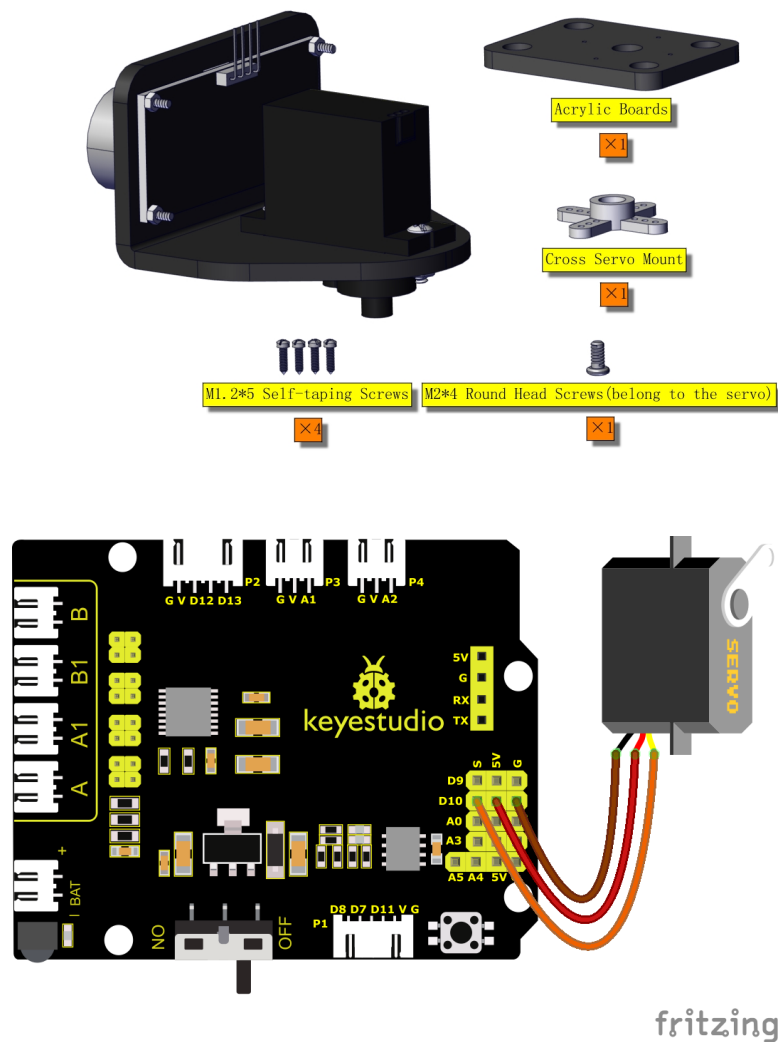




Step 9

**Step 10**

Need to adjust the angle of the servo



Set the angle of the servo to 90°

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

Add the servo library first, then copy the code below into the Arduino IDE and upload it to the motherboard, or just open the code provided by us and upload it to the motherboard.

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 0
  set servo
  http://www.keyestudio.com
*/
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

```

(continues on next page)

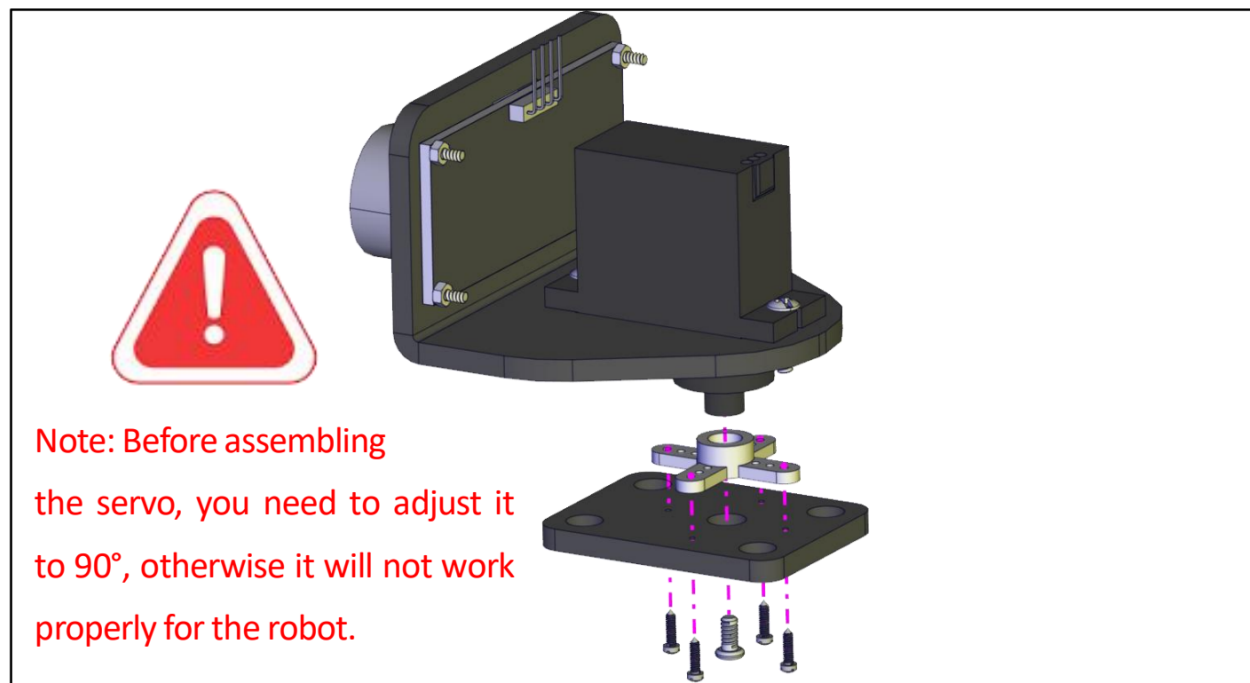
(continued from previous page)

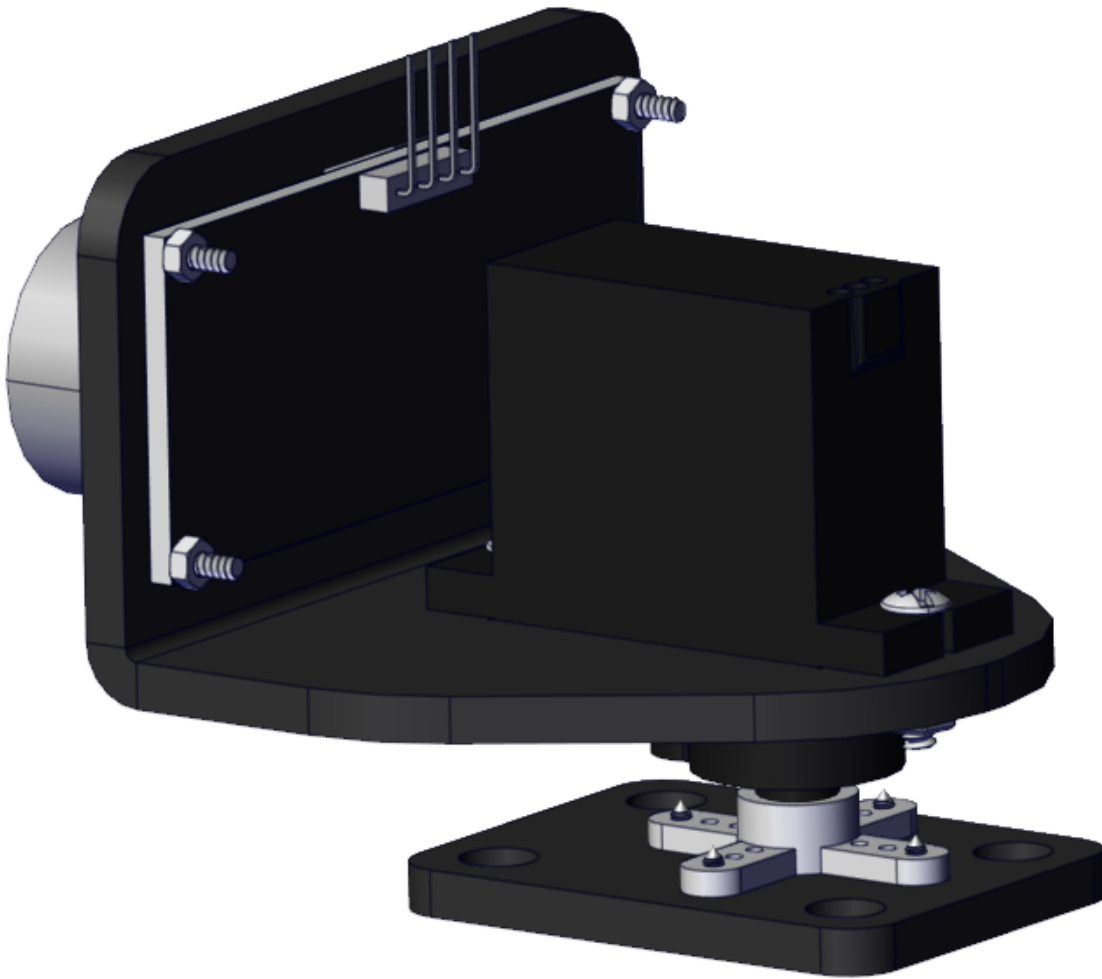
```
int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(10); // attaches the servo on pin 10 to the servo object
  myservo.write(0);
  delay(500);
  myservo.write(90);
  delay(500);
  myservo.write(180);
  delay(500);
  myservo.write(90);
  delay(500);
}

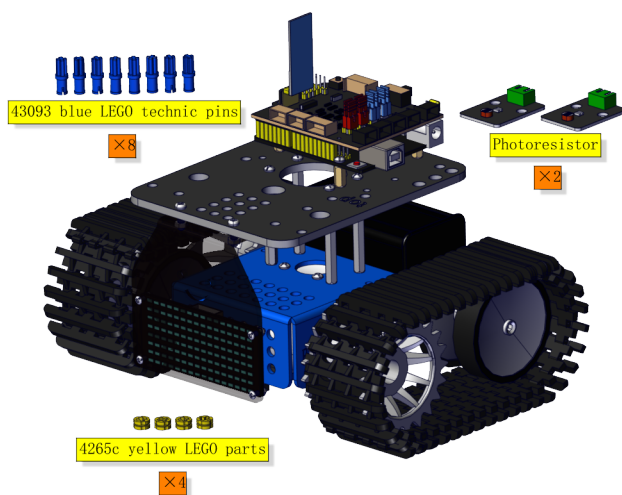
void loop() {
}
```

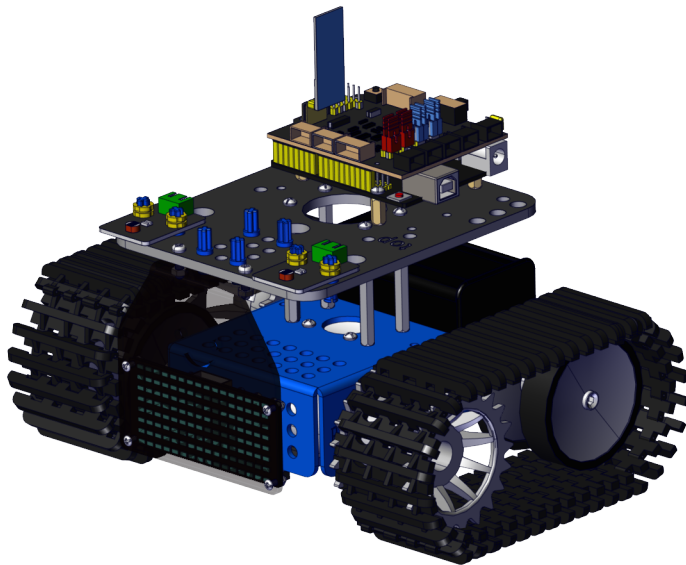
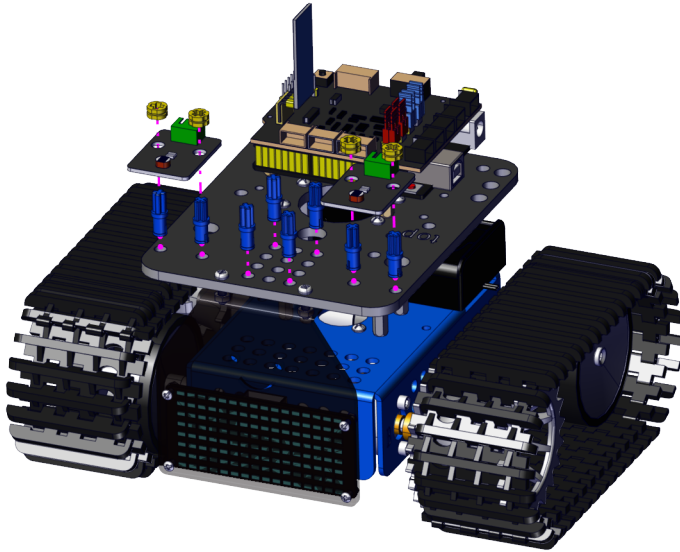
Keep the ultrasonic sensor parallel to the board



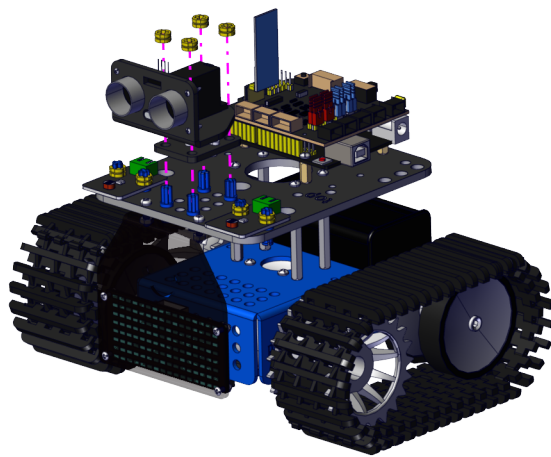
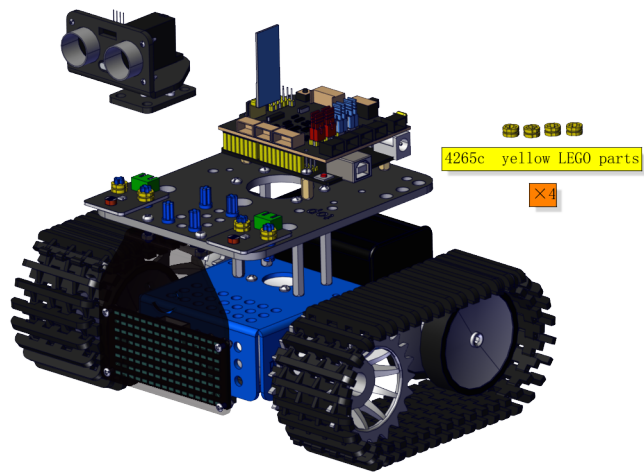


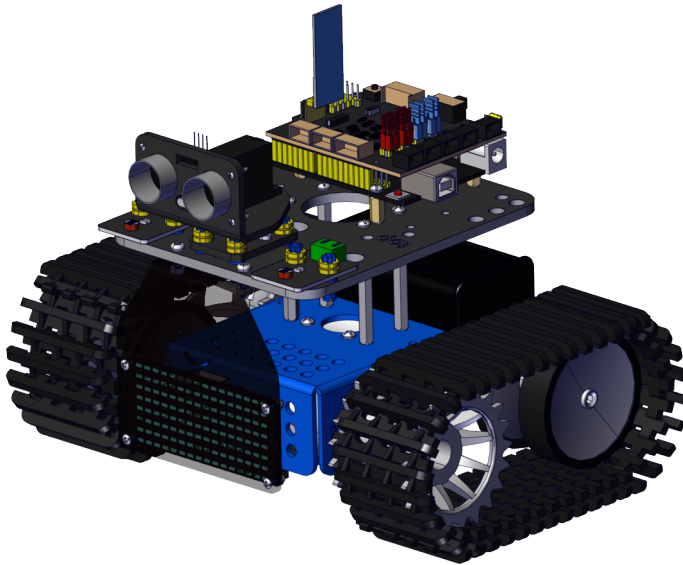
Step 11





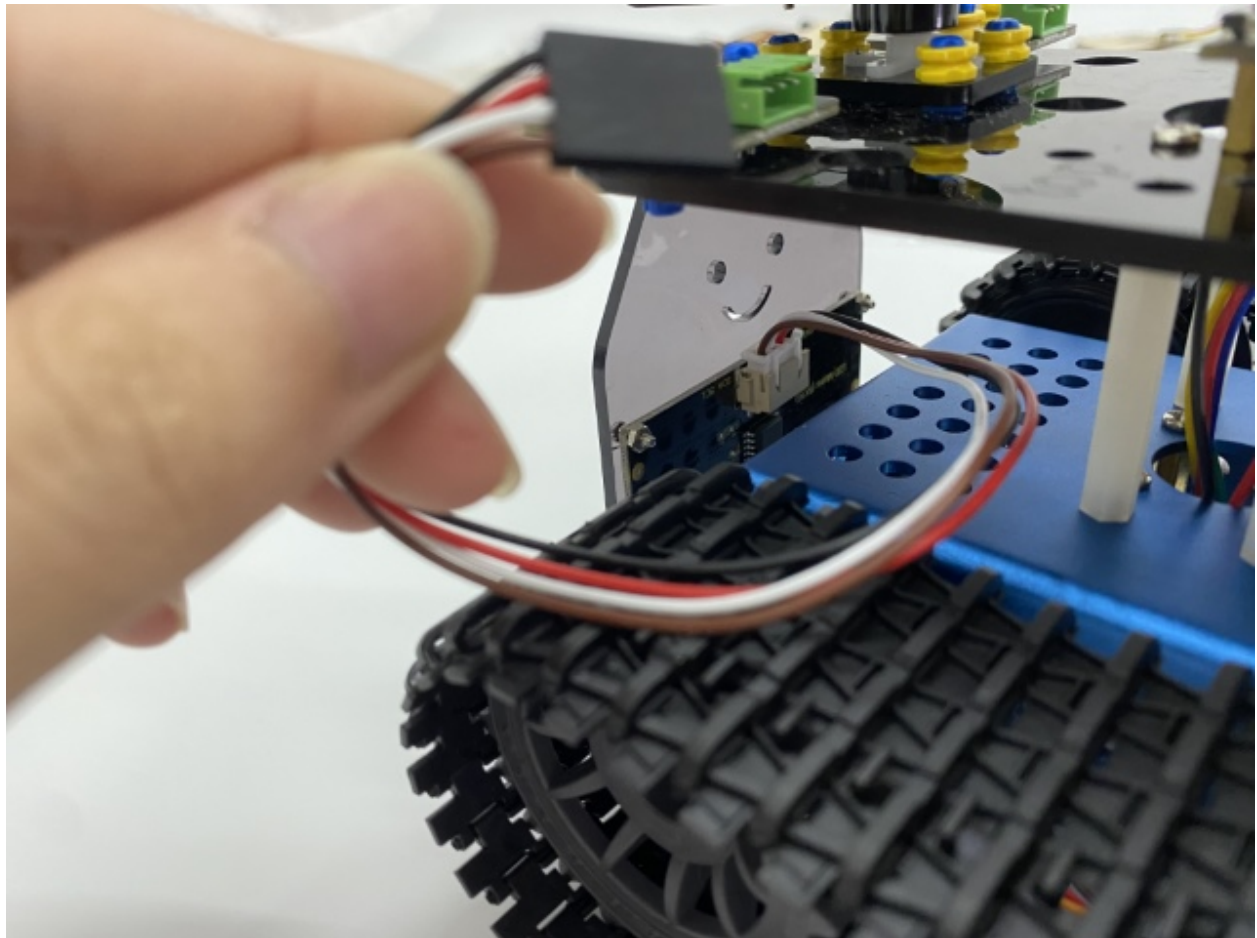
Step 12

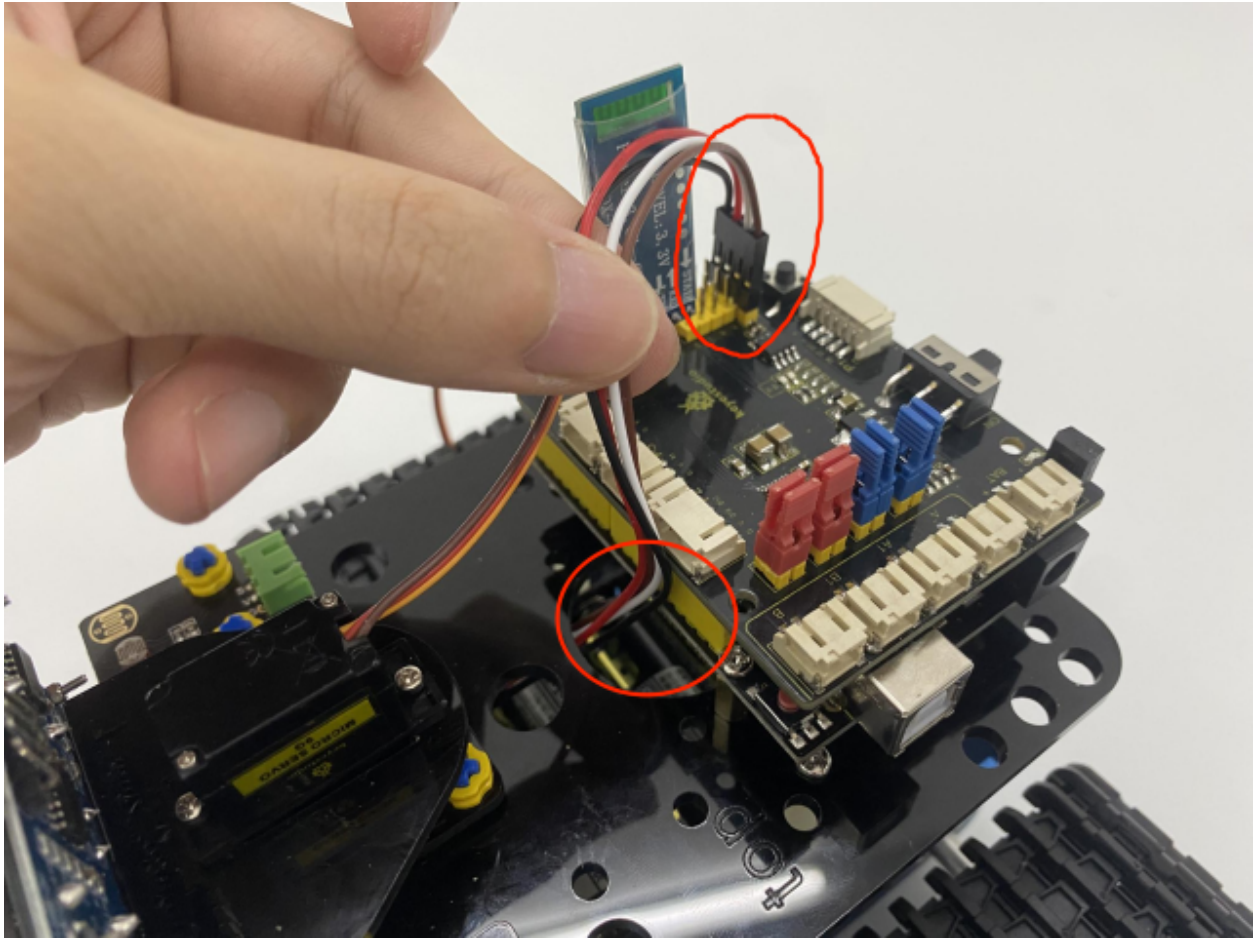




Wire up

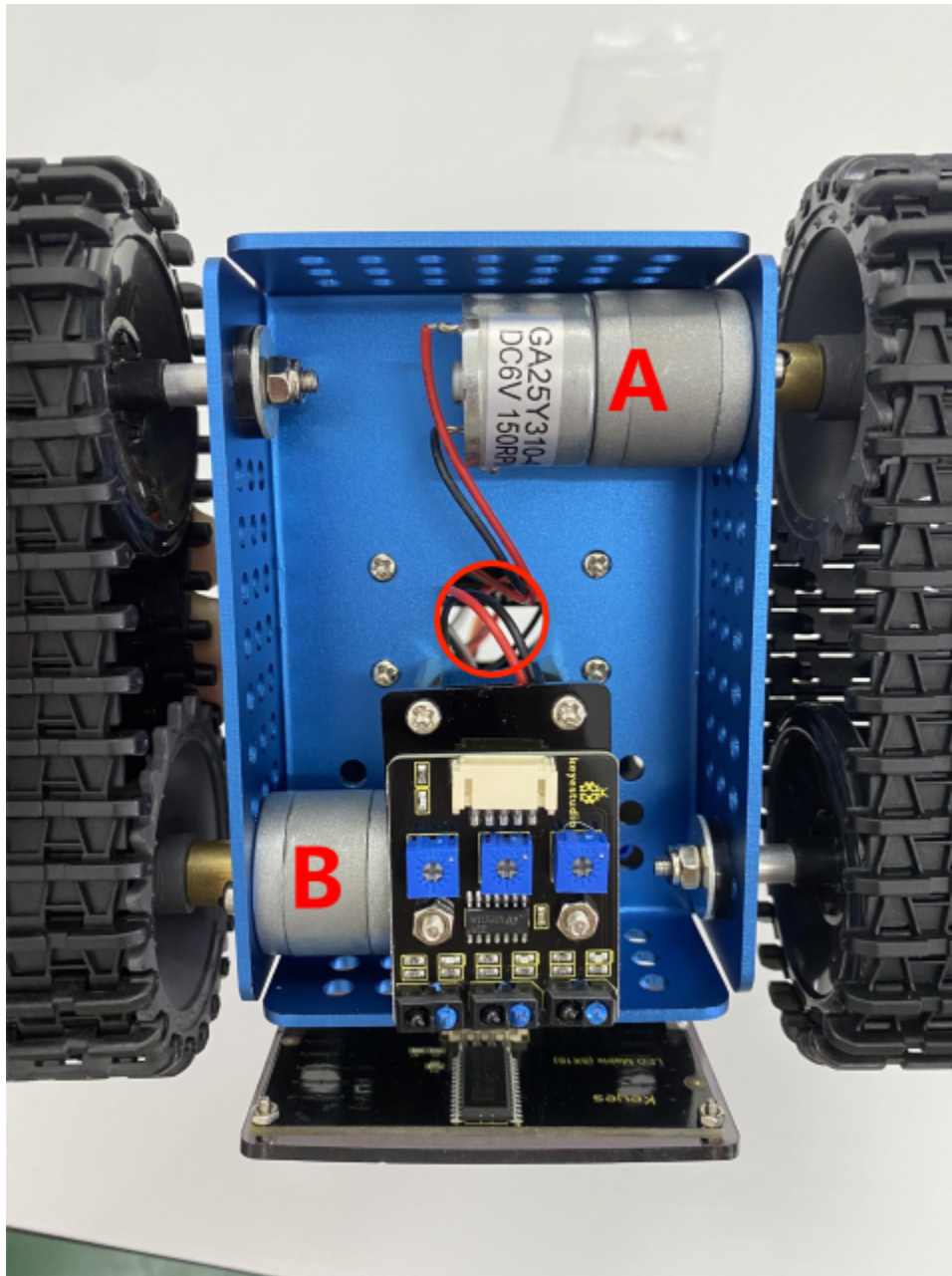
For 8*16LED panel, Make wires connect to A4 and A5

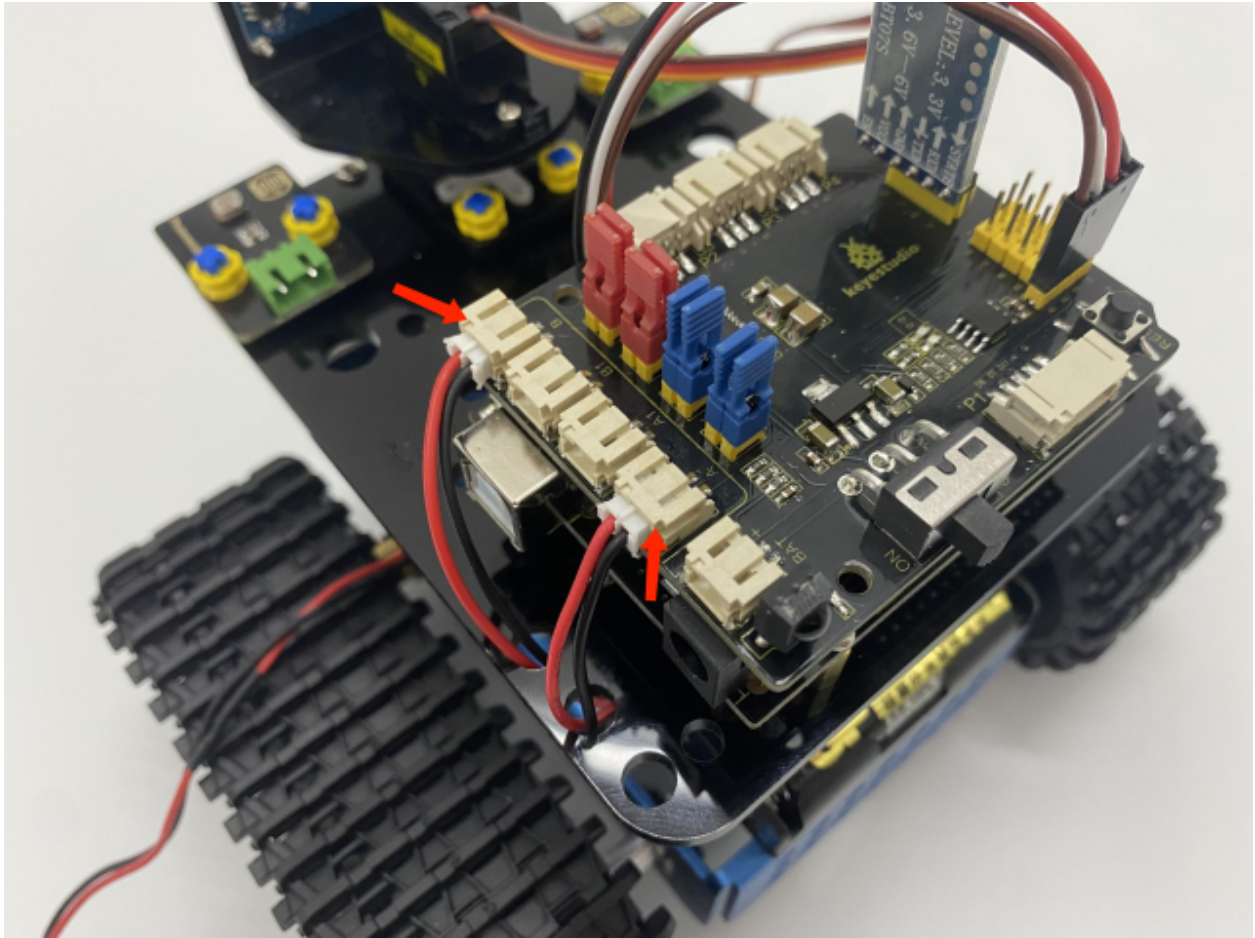




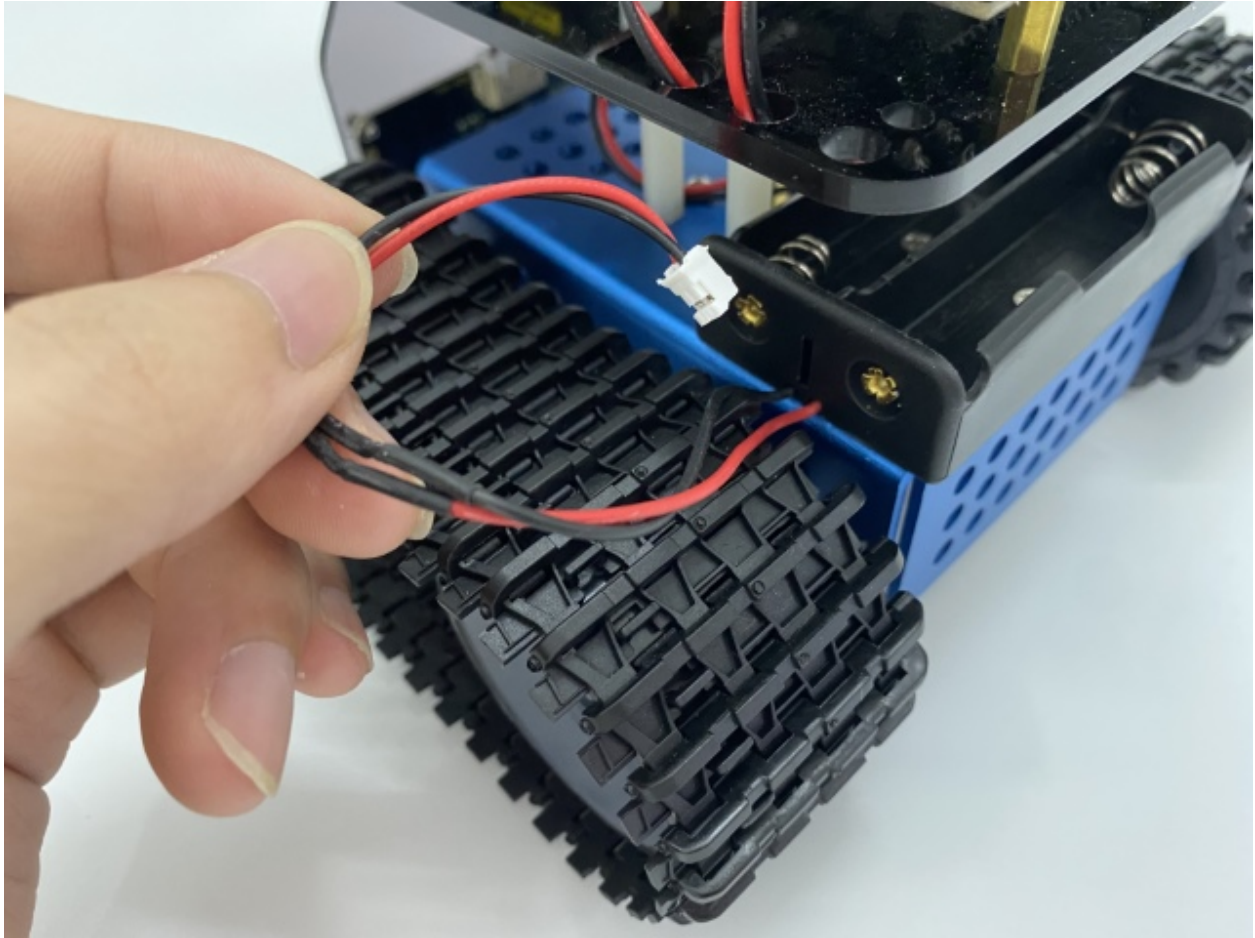
8*16 LED Panel	Keyestudio 8833 Motor Driver Board
GND	G
VCC	V
SDA	A4
SCL	A5

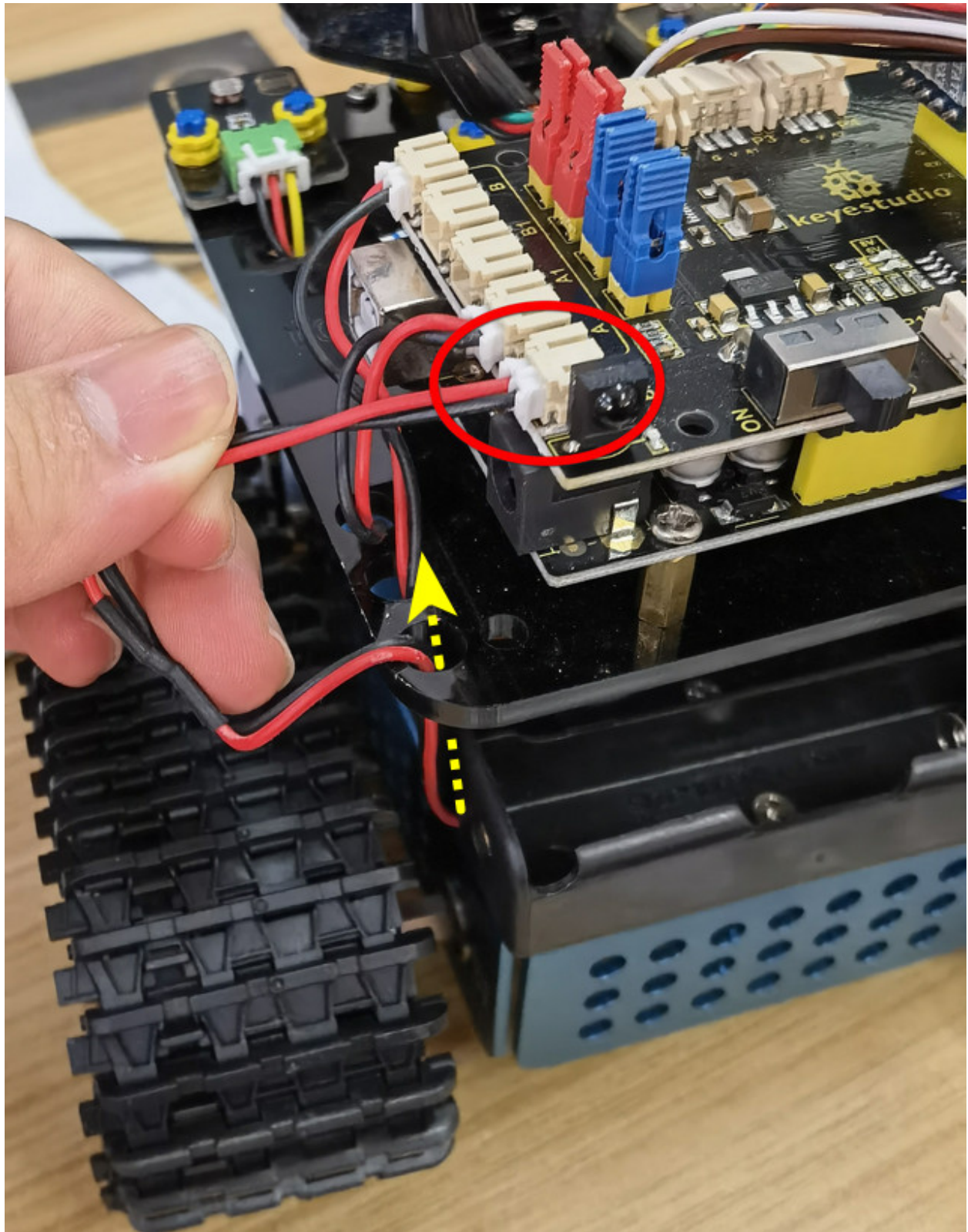
Connect the motor A to B port and make the motor B to A port.





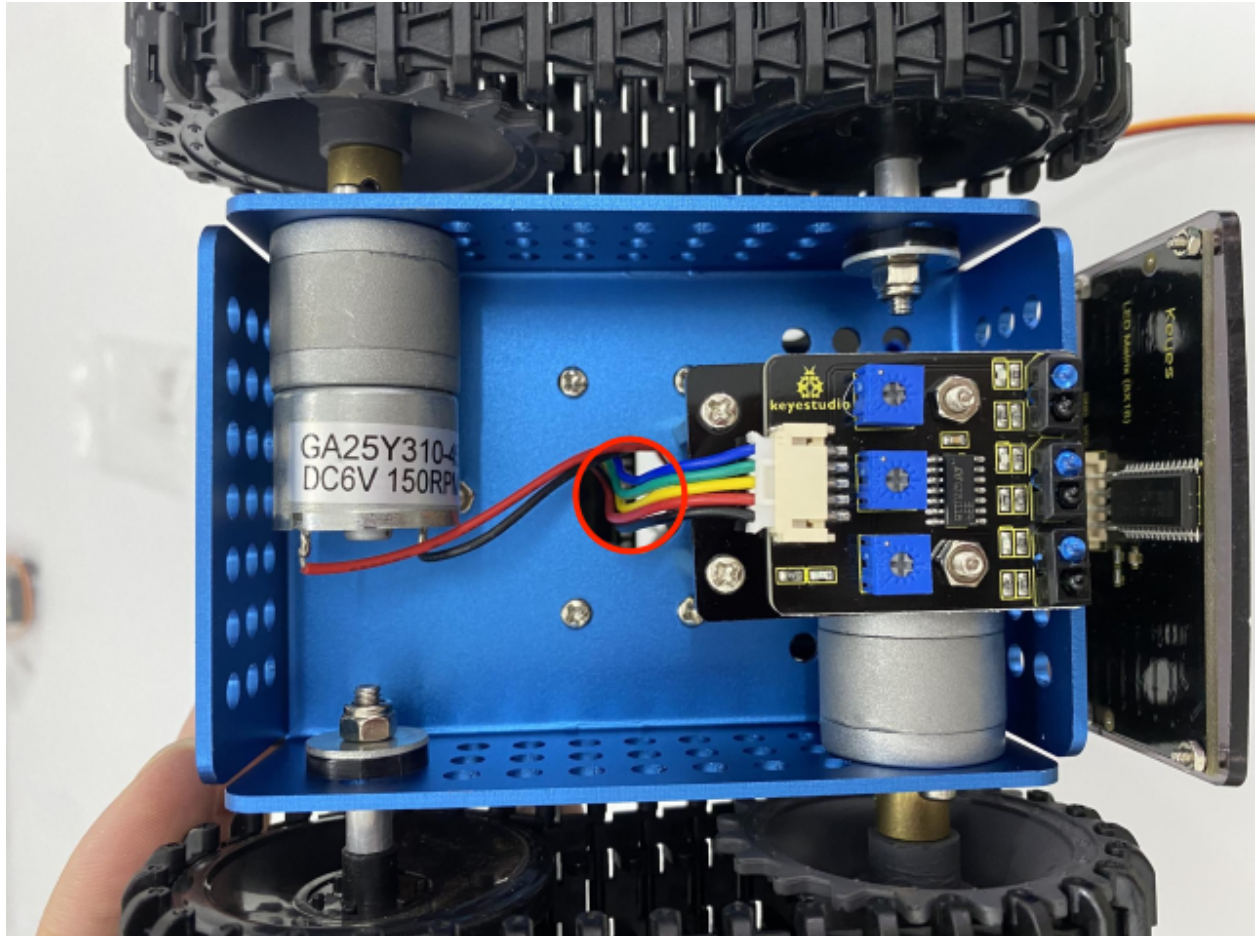
Connect the power wire

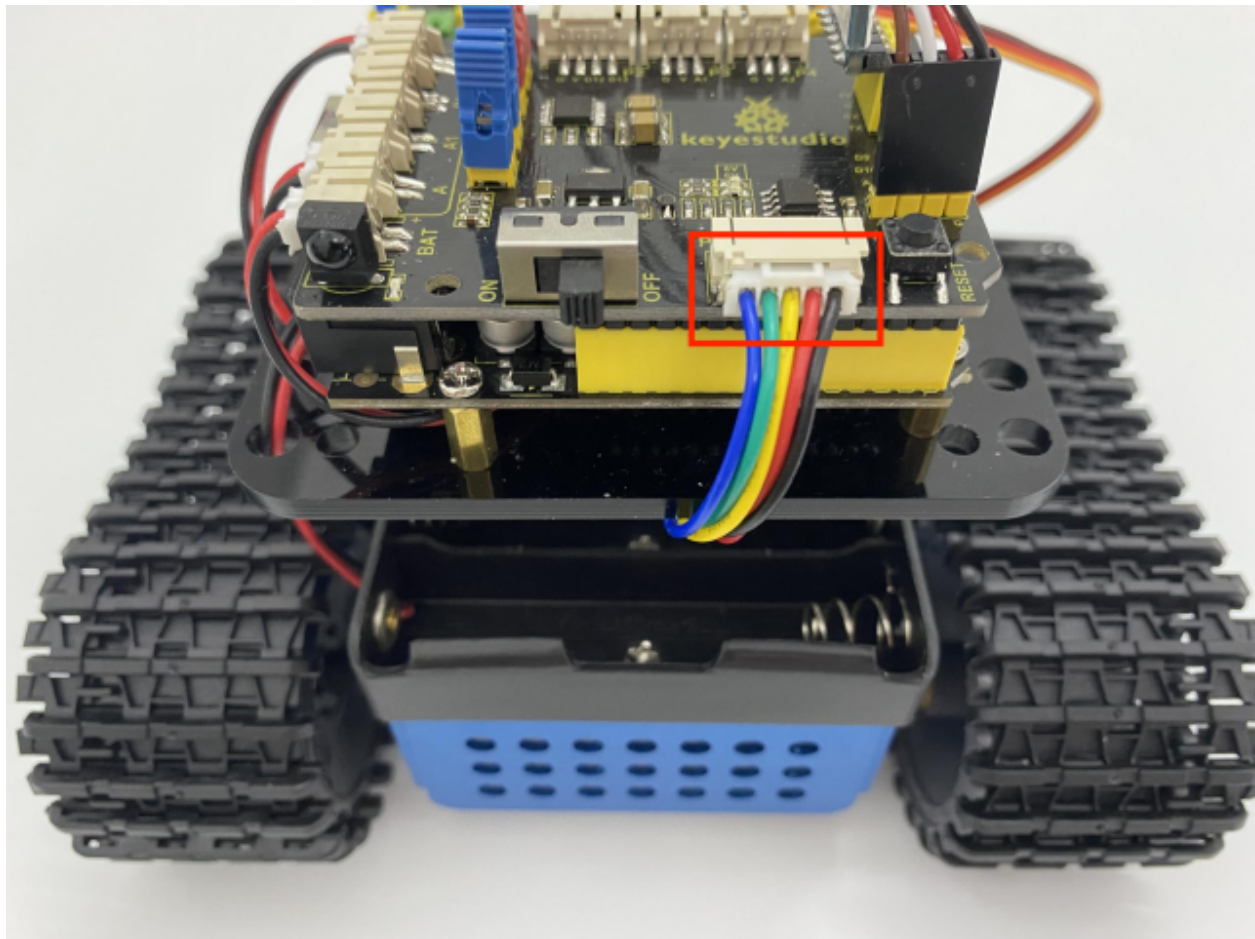




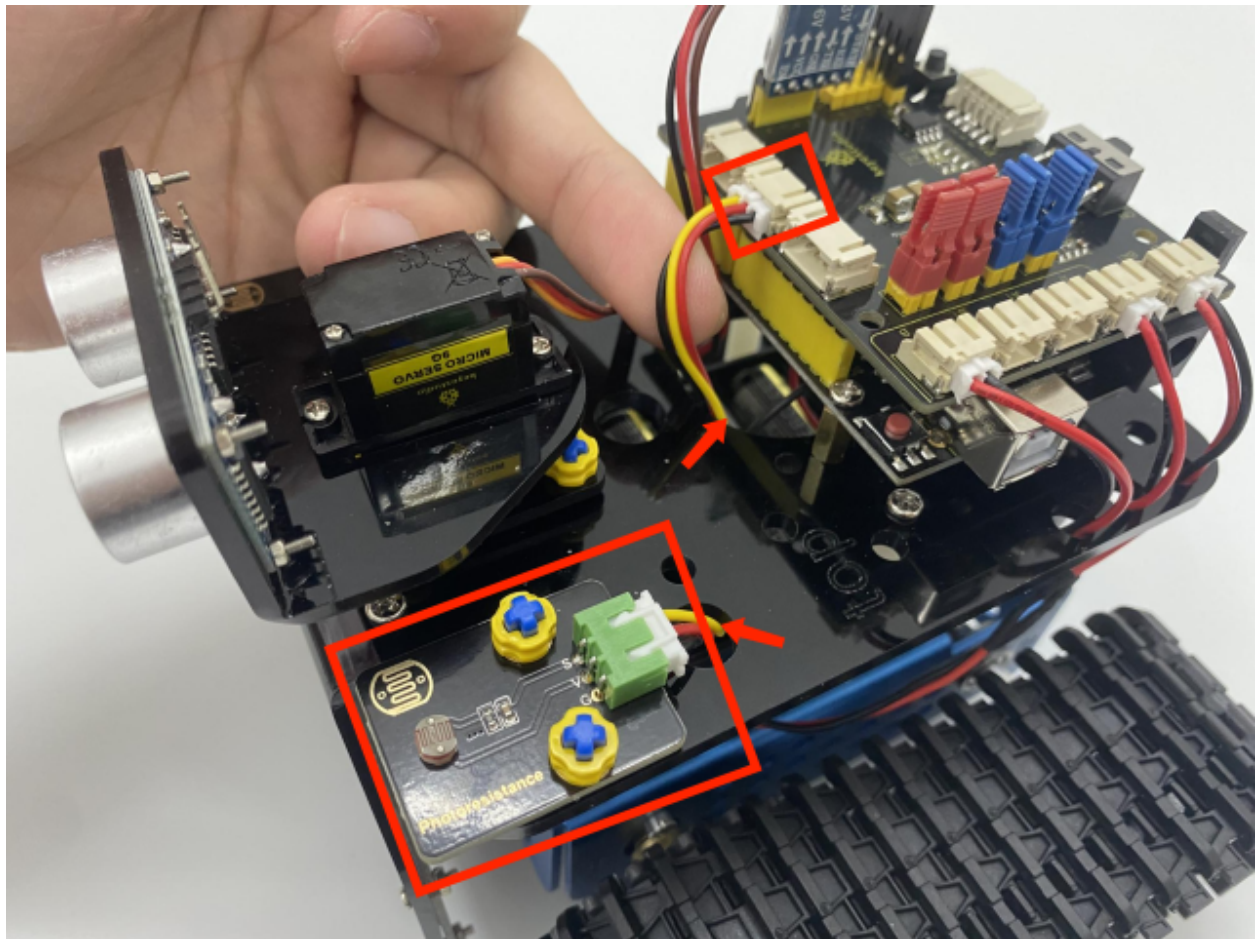
Line Tracking Sensor(see the picture)

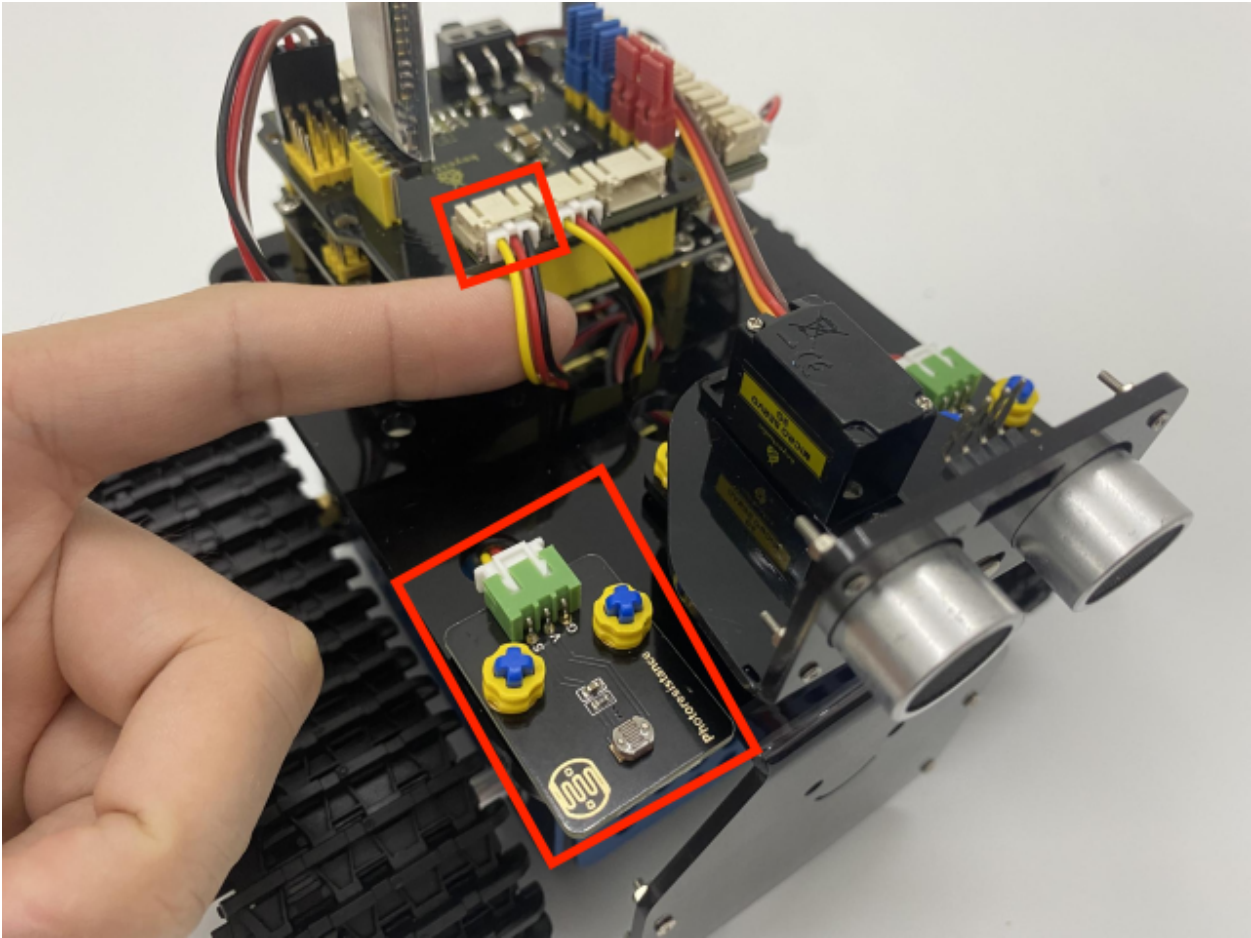






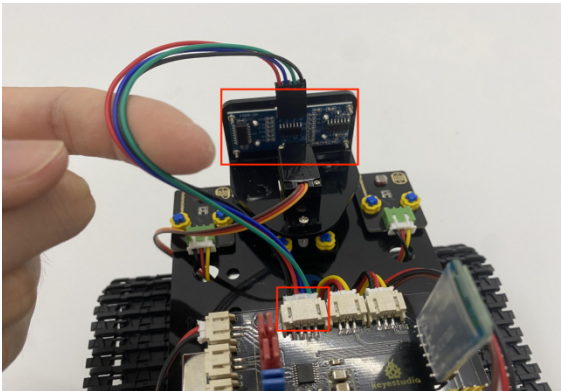
Wire up the photoresistors





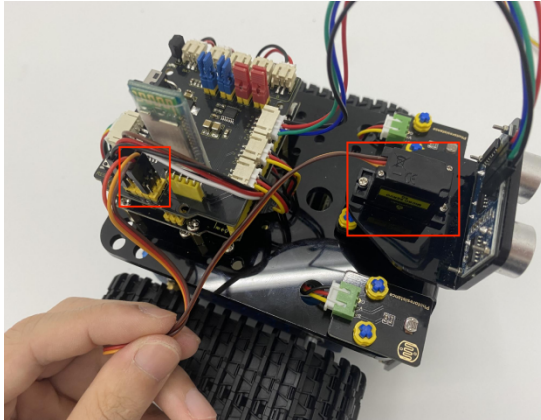
Left	Keyestudio 8833 Board	Right	Keyestudio 8833 Board
G	G	G	G
V	V	V	V
S	A1	S	A2

Wire up ultrasonic sensor



Ultrasonic Sensor	Keyestudio 8833 Board
Vcc	V
Trig	D12
Echo	D13
Gnd	G

Wire up the servo(D10)



Servo	Keyestudio 8833 Board
Brown	G
Red	V(5V)
Orange	D10

We adopt a model 18650 lithium battery with a pointed positive pole, whose power and capacity are not required.



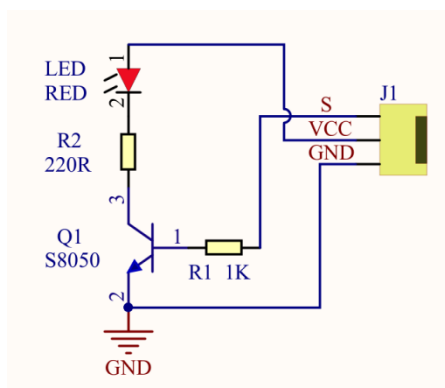
6.3 3. Projects

In this section, we will start from a single sensor to complex experiments to introduce you how the tank robot works

Note: (G), marked on each sensor and module, is the negative pole and connected to “G”, “-” or “GND” on the sensor shield or control board ; (V) is the positive pole and linked with V , VCC, + or 5V on the sensor shield or control board.

6.3.1 Project 1: LED Blinks

(1)Description:

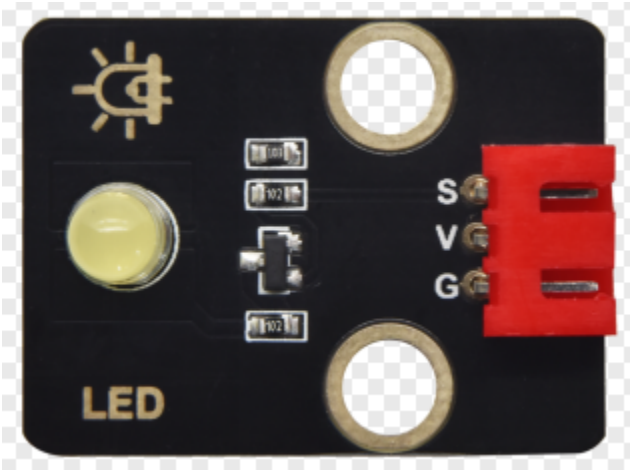


For starters and enthusiasts, LED Blink is a fundamental program. LED, the abbreviation of light emitting diodes, consists of Ga, As, P, N chemical compounds and so on. The LED can flash in diverse colors by altering the delay time

in the test code. When in control, power on GND and VCC, the LED will be on if S end is in high level; nevertheless, it will go off.

(2)Parameters:

Control interface: digital port

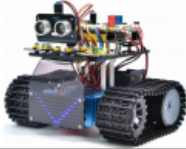



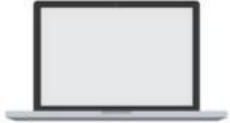


Working voltage: DC 3.3-5V

Pin spacing: 2.54mm

LED display color: yellow

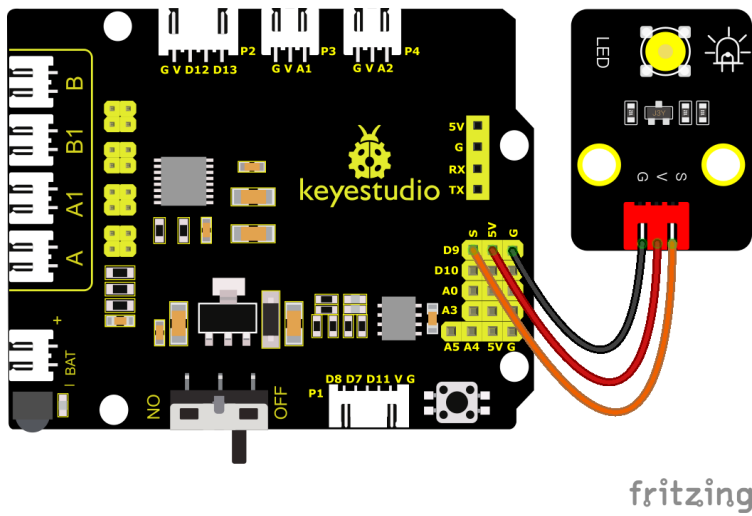
(3)Components Required:

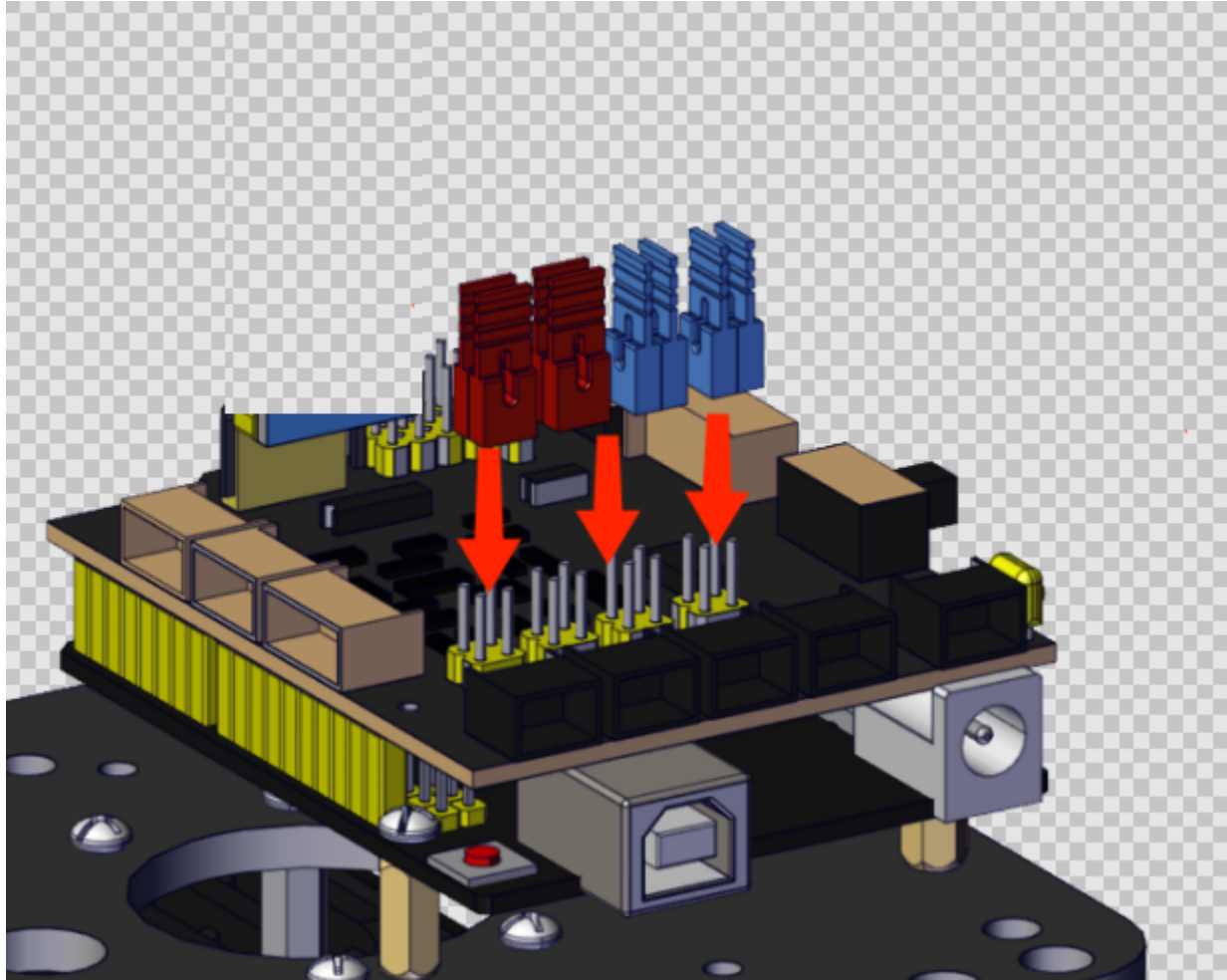
Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

(4) Motor driver expansion board:

The Keystudio 8833 motor driver expansion board is compatible with the Arduino UNO development board. Just stack it onto the development board when using it.

(5) Connection Diagram:





NOTE: LED is connected to D9 port, and remember to install jumper caps onto the shield

(6)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*  
  
Keyestudio Mini Tank Robot V3 (Popular Edition)  
  
lesson 1.1  
  
Blink  
  
http://www.keyestudio.com  
  
*/  
  
int LED = 9; //Define the pin of LED to connect with digital port 3
```

(continues on next page)

(continued from previous page)

```

void setup(){
    pinMode(LED, OUTPUT); //Initialize the LED pin to output mode
}

void loop() //infinite loop
{
    digitalWrite(LED, HIGH); //Output high level and turn on the LED
    delay(1000); //Wait for 1s
    digitalWrite(LED, LOW); //Output low level and turn on LED
    delay(1000); //Wait for 1s
}

```

(7)Test Results:

Upload the program, LED blinks at the interval of 1s.

(8)Code Explanation:

pinMode(LEDOUTPUT) - This function can denote that the pin is INPUT or OUTPUT

digitalWrite(LEDHIGH) - When pin is OUTPUT, we can set it to HIGH(output 5V) or LOW(output 0V)

(9)Extension Practice:

We have succeeded in blinking LED. Next, let's observe what will happen to the LED if we modify pins and delay time.

Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)

lesson 1.2

Blink

http://www.keyestudio.com

*/

int LED = 9; //Define the pin of the LED as 9

void setup(){
    pinMode(LED, OUTPUT); //Set the pin of the LED to OUTPUT
}

```

(continues on next page)

(continued from previous page)

```
void loop() //Infinite loop
{
    digitalWrite(LED, HIGH); //output high levels, light up LED
    delay(100); //Wait for 0.1s
    digitalWrite(LED, LOW); //LED output low levels, turn off LED
    delay(100); //Wait for 0.1s
}
```

The test result shows that the LED flashes faster. Therefore, we can draw a conclusion that pins and time delaying affect flash frequency.

6.3.2 Project 2: Adjust LED Brightness

(1)Description:

In previous lesson, we control LED on and off and make it blink.

In this project, we will control LED's brightness through PWM simulating breathing effect. Similarly, you can change the step length and delay time in the code so as to demonstrate different breathing effects.

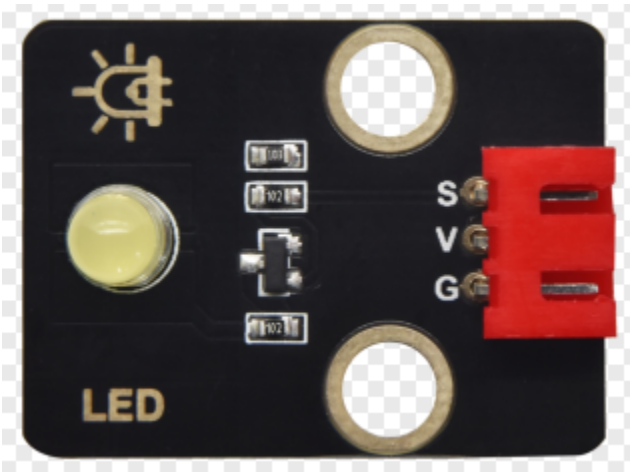
PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output.

In general, the input voltages of ports are 0V and 5V. What if the 3V is required? Or a switch among 1V, 3V and 3.5V? We cannot change resistors constantly. For this reason, we resort to PWM.

For Arduino digital port voltage outputs, there are only LOW and HIGH levels, which correspond to the voltage outputs of 0V and 5V respectively. You can define LOW as "0" and HIGH as "1", and let the Arduino output five hundred '0' or '1' within 1 second. If output five hundred '1', that is 5V; if all of which is '0', that is 0V; if output 250 01 pattern, that is 2.5V.

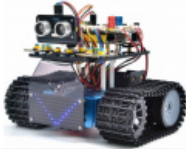




This process can be likened to showing a movie. The movie we watch are not completely continuous. Actually, it generates 25 pictures per second, which cannot be told by human eyes. Therefore, we mistake it as a continuous process. PWM works in the same way. To output different voltages, we need to control the ratio of 0 and 1. The more '0' or '1' output per unit time, the more accurate the control.

(2)Parameters:



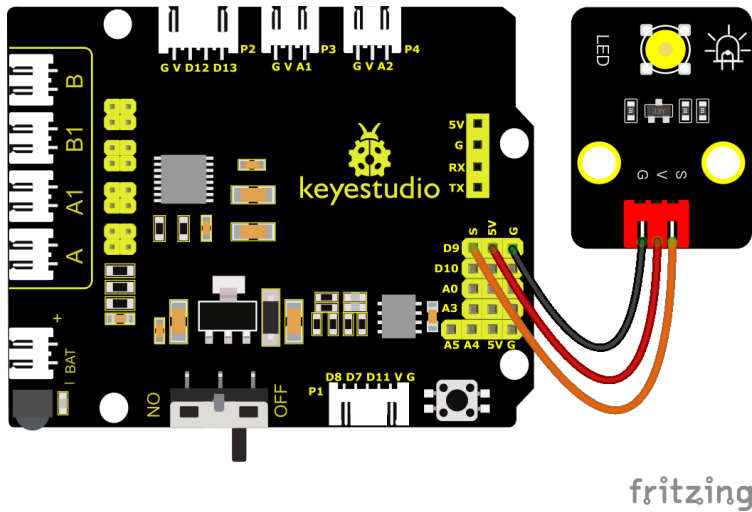
Control interface: Digital port 3
Working voltage: DC 3.3-5V
Pin spacing: 2.54mm
LED display color: yellow

(3)Components Required:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

(4)Connection Diagram:

PWM pins of Arduino are connected to 356910 and 11. Keep the pin 9 unchanged



(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)

lesson 2.1

pwm

http://www.keyestudio.com
*/

int LED = 9; //Define the pin of the LED as 9

void setup () {
    pinMode(LED, OUTPUT); //Set the pin of the LED to OUTPUT
}

void loop () {
    for (int value = 0; value < 255; value = value + 1) {
        analogWrite(LED, value); // LED ON
        delay(5); //delay in 5ms
    }
    for (int value = 255; value > 0; value = value - 1) {
        analogWrite(LED, value); //LED gets dim
        delay(5); //delay in 5ms
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
}

```

(6)Test Results:

Upload test code successfully, LED gradually changes from bright to dark, like human's breath, rather than turning on and off immediately.

(7)Code Explanation:

To repeat some certain statements, we could use FOR statement. FOR statement format is shown below:

```

① for (cycle initialization; ② condition is true ④
③ loop body statement; cycle adjustment statement) {
}

```

FOR cyclic sequence:

Round 11 → 2 → 3 → 4

Round 22 → 3 → 4

...

Until number 2 is not established, "for" loop is over.

After knowing this order, go back to code:

```

for (int value = 0; value < 255; value=value+1){
...}
for (int value = 255; value >0; value=value-1){
...}

```

The two "for" statements make value increase from 0 to 255, then reduce from 255 to 0, then increase to 255, ... infinitely loop

There is a new function in the following — analogWrite()

We know that digital port only has two state of 0 and 1. So how to send an analog value to a digital value? Here, this function is needed. Let's observe the Arduino board and find 6 pins marked "~" which can output PWM signals.

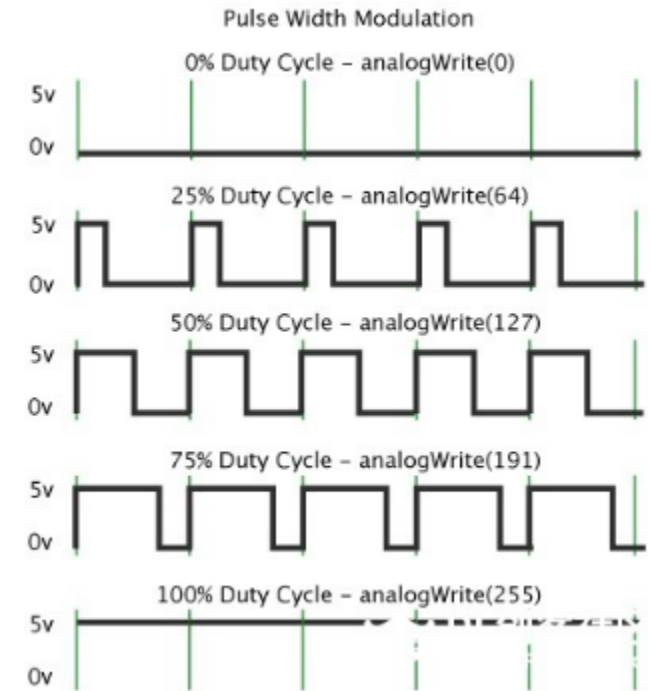
Function format as follows:

analogWrite(pin,value)

analogWrite() is used to write an analog value from 0~255 for PWM port, so the value is in the range of 0~255. Attention that you only write the digital pins with PWM function, such as pin 3, 5, 6, 9, 10, 11.

PWM is a technology to obtain analog quantity through digital method. Digital control forms a square wave, and the square wave signal only has two states of turning on and off (that is, high or low levels). By controlling the ratio of the duration of turning on and off, a voltage varying from 0 to 5V can be simulated. The time turning on (academically referred to as high level) is called pulse width, so PWM is also called pulse width modulation.

Through the following five square waves, let's acknowledge more about PWM.



In the above figure, the green line represents a period, and value of analogWrite() corresponds to a percentage which is called Duty Cycle as well.

Duty cycle implies that high-level duration is divided by low-level duration in a cycle. From top to bottom, the duty cycle of first square wave is 0% and its corresponding value is 0. The LED brightness is lowest, that is, light off. The more time the high level lasts, the brighter the LED. Therefore, the last duty cycle is 100%, which correspond to 255, and LED is the brightest. And 25% means darker.

PWM mostly is used for adjusting the LED's brightness or the rotation speed of motors.

It plays a vital role in controlling smart robot cars. I believe that you cannot wait to learn next project.

(8)Extension Practice:

Let's modify the value of delay and remain the pin unchanged, then observe how LED changes.

Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*
Keyestudio Mini Tank Robot V3 (Popular Edition)

lesson 2.2

pwm-slow

http://www.keyestudio.com
*/
```

(continues on next page)

(continued from previous page)

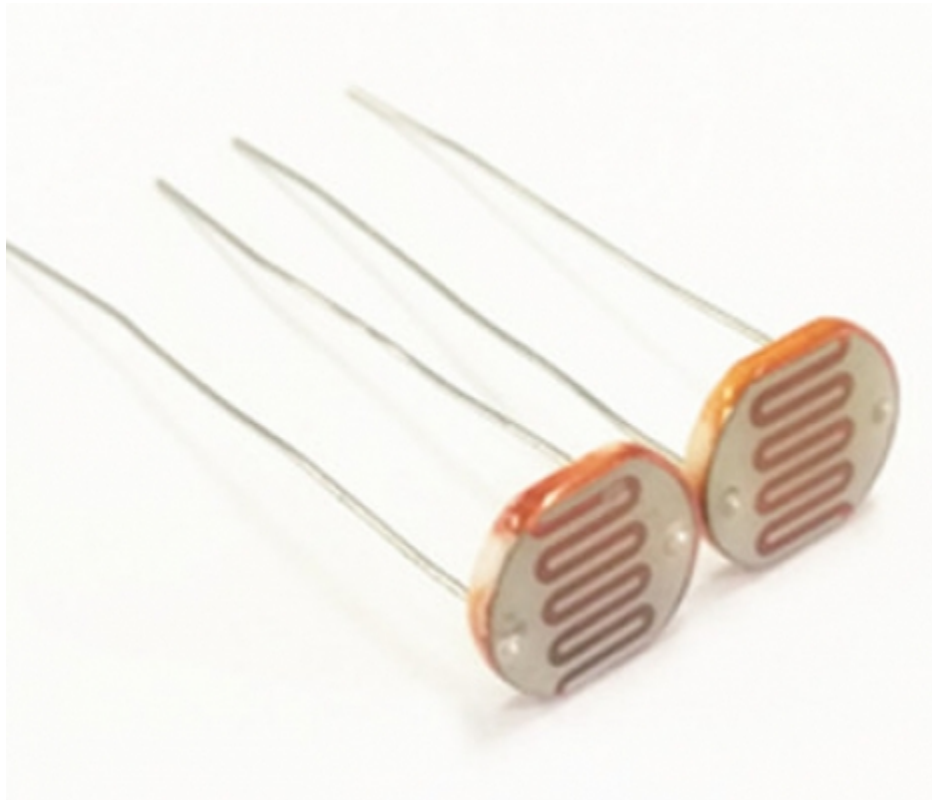
```
int LED = 9; //Define the pin of the LED as 9

void setup() {
    pinMode(LED, OUTPUT); //Set the pin of the LED to OUTPUT
}

void loop () {
    for (int value = 0; value < 255; value = value + 1) {
        analogWrite(LED, value); // LED ON
        delay(30); // delay in 30ms
    }
    for (int value = 255; value > 0; value = value - 1) {
        analogWrite(LED, value); //LED gets dim
        delay (30); // delay in 30ms
    }
}
```

Upload the code to development board, LED flashes more slowly.

6.3.3 Project 3: Photoresistor

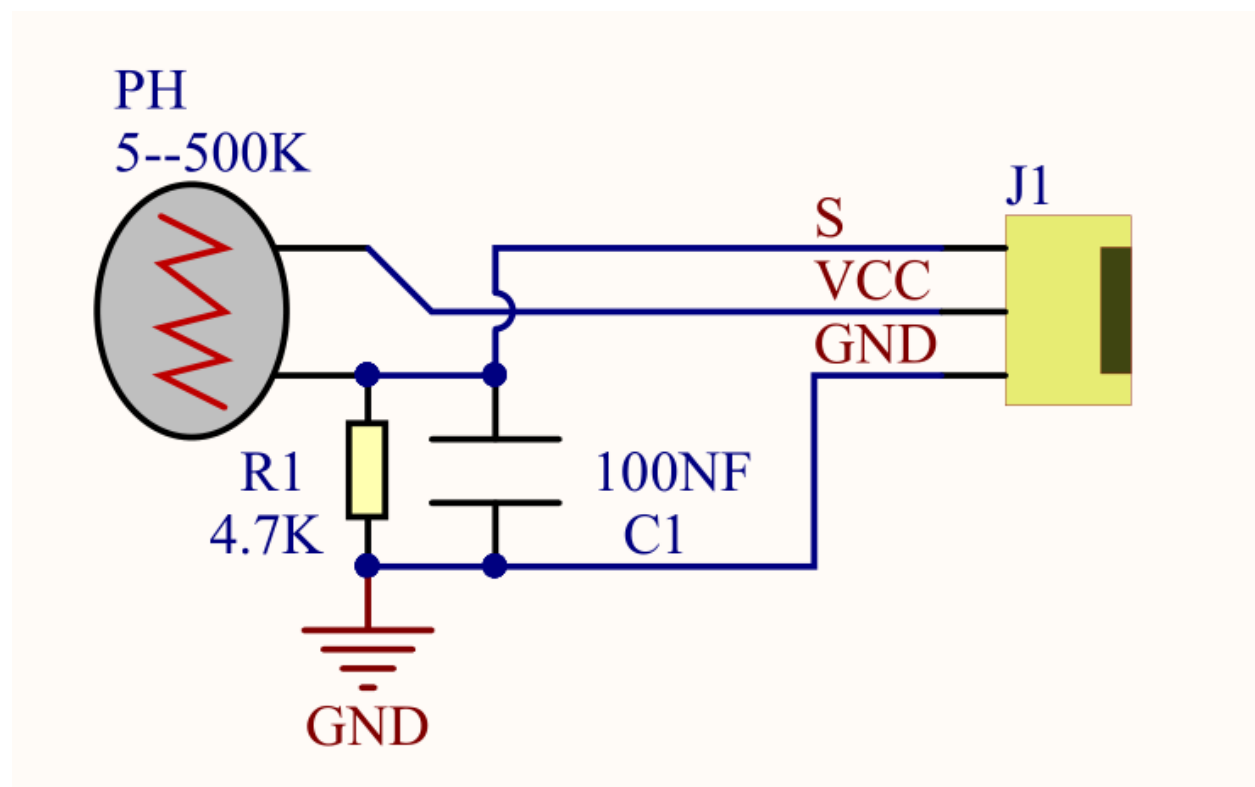
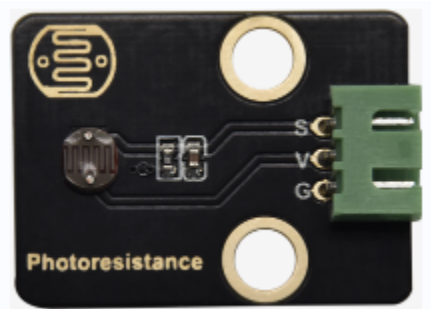


(1)Description:

The photosensitive resistor is a special resistor made of a semiconductor material such as a sulfide or selenium, and a moisture-proof resin is also coated with a photoconductive effect. The photosensitive resistance is most sensitive to the ambient light, different illumination strength, and the resistance of the photosensitive resistance is different. We use the photosensitive resistance to design the photosensitive resistor module.

The module signal is connected to the microcontroller analog port. When the light intensity is stronger, the larger the analog port voltage, that is, the simulation value of the microcontroller is also large; in turn, when the light intensity is weaker, the smaller the analog port voltage, that is, the simulation value of the microcontroller is also small.

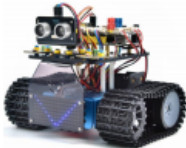




In this way, we can read the corresponding analog value using the photosensitive resistor module, and the intensity of the light in the inductive environment.



(2)Parameters:

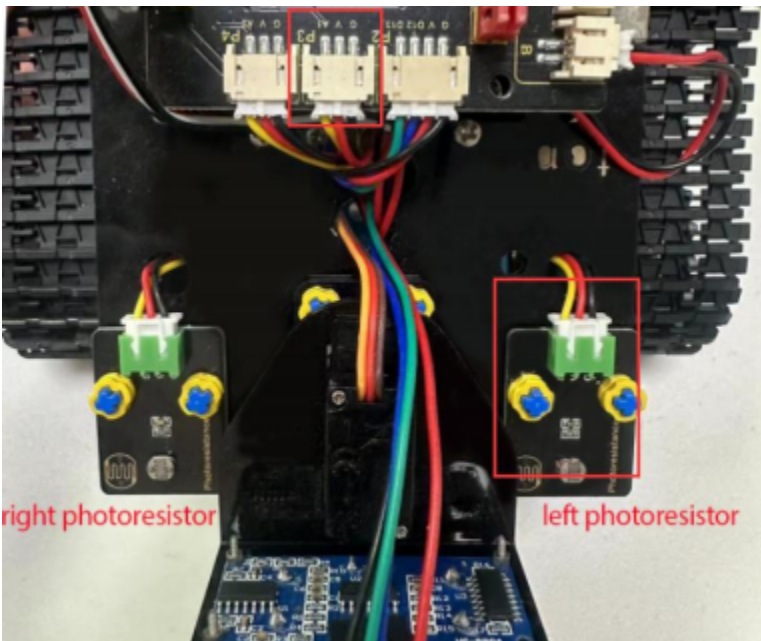
Photosensitive resistance resistance value: 5K Ou-0.5m
Interface type: simulation port A0, A1
Working voltage: 3.3V-5V
Pin spacing: 2.54mm

(3)Components Needed:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

(4)Connection Diagram:

What we are going to test next isthe photoresistor module on the leftside ofthe robot

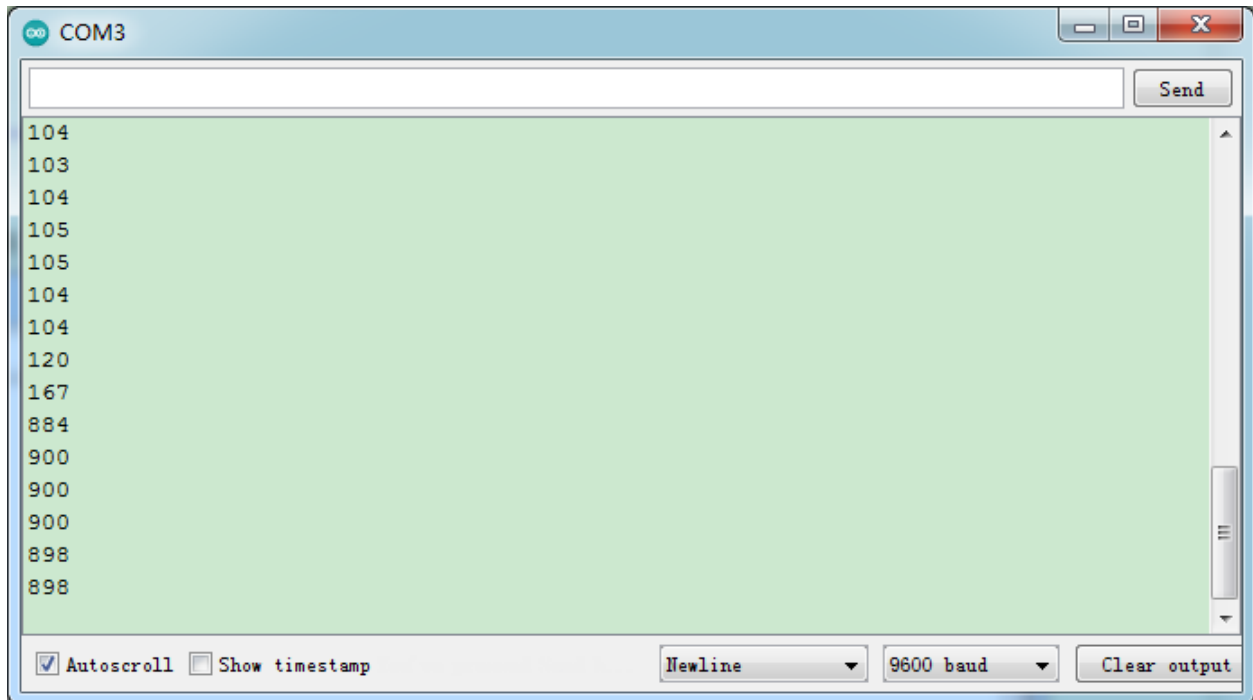


(continued from previous page)

```

    delay(500); //Delay in 500ms
}

```

(6)Test Results:

When covering it, the value gets smaller; if not, the value gets larger

(7)Code Explanation:

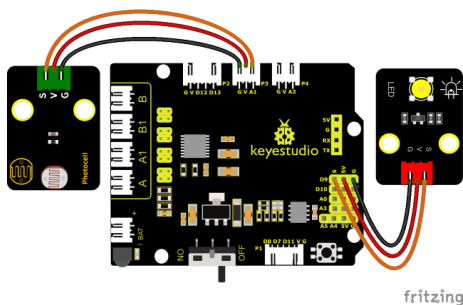
analogRead(sensorPin): read the analog value of photoresistors

Serial.begin(9600): initialize serial port and set baud rate to 9600****

Serial.println: serial prints

(8)Extension Practice:

We know the value of the photoresistor. How about controlling the LED's brightness by it?



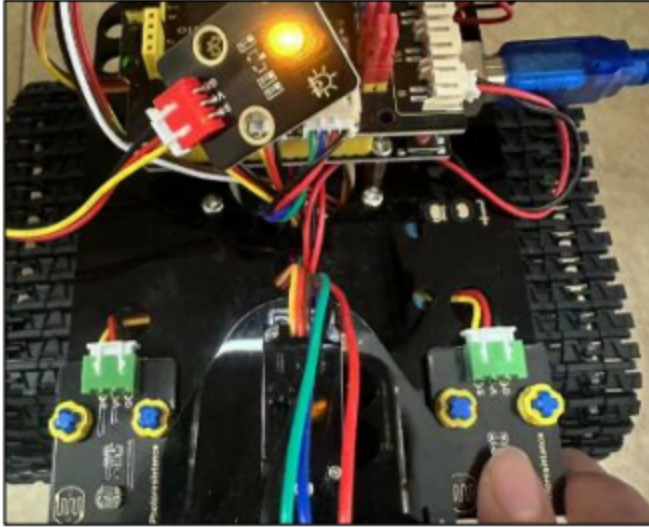
The LED's brightness is controlled by PWM. Therefore, we connect the LED to PMW pin(pin 9) of the shield.

Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*  
  
Keyestudio Mini Tank Robot V3 (Popular Edition)  
  
lesson 3.2  
  
photocell-analog output  
  
http://www.keyestudio.com  
  
*/  
  
int analogInPin = A1; // A1 is the input pin of photoresistor  
int analogOutPin = 9; // Digital port 9 is the output of PMW  
int sensorValue = 0; // save the variable of the resistance value of photoresistors  
int outputValue = 0; // Value output to PMW  
  
void setup() {  
    Serial.begin(9600); //Open the serial port monitor and set the baud rate to 9600  
}  
  
void loop() {  
    sensorValue = analogRead(analogInPin); //Read the analog value from the photoresistor  
    //↪ sensor  
    // Map the analog values 0~1023 to the PWM output values 255~0  
    outputValue = map(sensorValue, 0, 1023, 255, 0);  
    // Change analog output  
    analogWrite(analogOutPin, outputValue);  
    Serial.println(sensorValue); //The serial port prints the value of the photoresistor  
    delay(2);  
}
```

Upload code to the development board, then cover the photoresistor and observe the LED's brightness.



6.3.4 Project 4: Line Tracking Sensor

(1)Description:



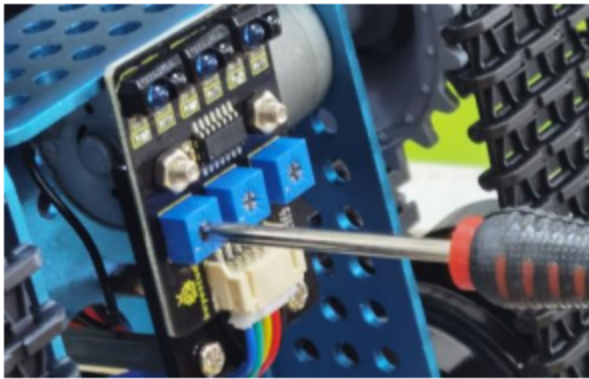
The tracking sensor is actually an infrared sensor. The component used here is the TCRT5000 infrared tube.

Its working principle is to use different reflectivity of infrared light to colors, then convert the strength of the reflected signal into a current signal.

During the process of detection, black is active at HIGH level while white is active at LOW level. The detection height is 0-3 cm.

Keyestudio 3-channel line tracking module has integrated 3 sets of TCRT5000 infrared tube on a single board, which is more convenient for wiring and control.

If the Line Tracking Sensor does not work as expected, you will need to use a screwdriver to adjust its potentiometer to make it more sensitive. When your finger is close to the sensor, its on-board LED light turns on, and when your finger moves away, its on-board LED light turns off. At this time, its sensitivity is relatively good.

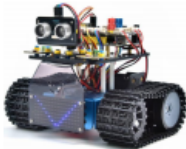



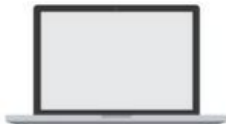


(2)Parameters:

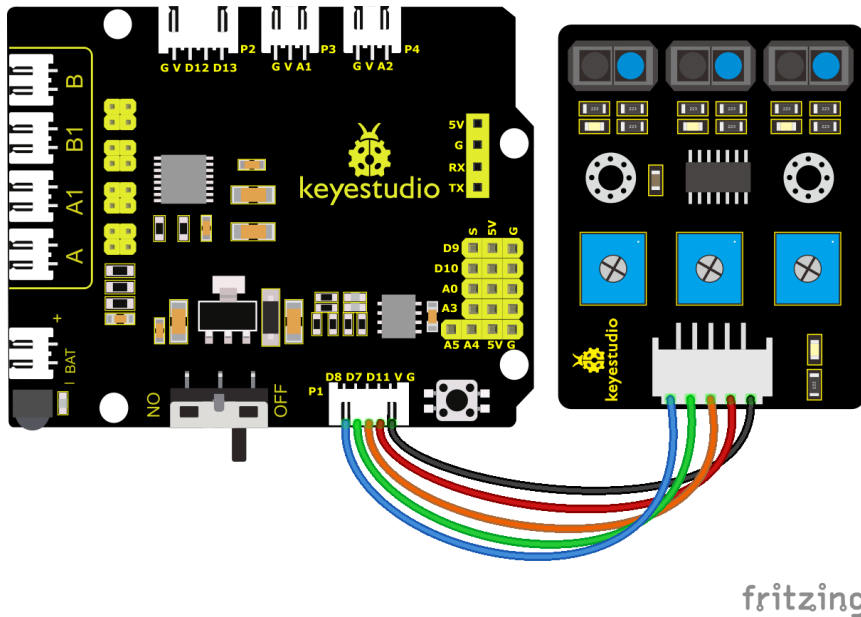
- Operating Voltage: 3.3-5V (DC)
- Interface: 5PIN
- Output Signal: Digital signal
- Detection Height: 0-3 cm

Special note: before testing,rotate the potentiometer on the sensor to adjust the detection sensitivity. When adjust the LED at the threshold between ON and OFF, the sensitivity is the best.

(3)Components Required:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

Note: the line tracking sensor is installed under the bottom of the robot.

(4) Connection Diagram:**(5) Test Code:**

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)

lesson 4.1

Line Track sensor

http://www.keyestudio.com

*/

//The wiring of line tracking sensors

#define L_pin 11 //left

#define M_pin 7 //middle

#define R_pin 8 //right

void setup(){
  Serial.begin(9600); //Set the baud rate to 9600
  pinMode(L_pin, INPUT); //Set all pins of the line tracking sensors to input mode
  pinMode(M_pin, INPUT);

```

(continues on next page)

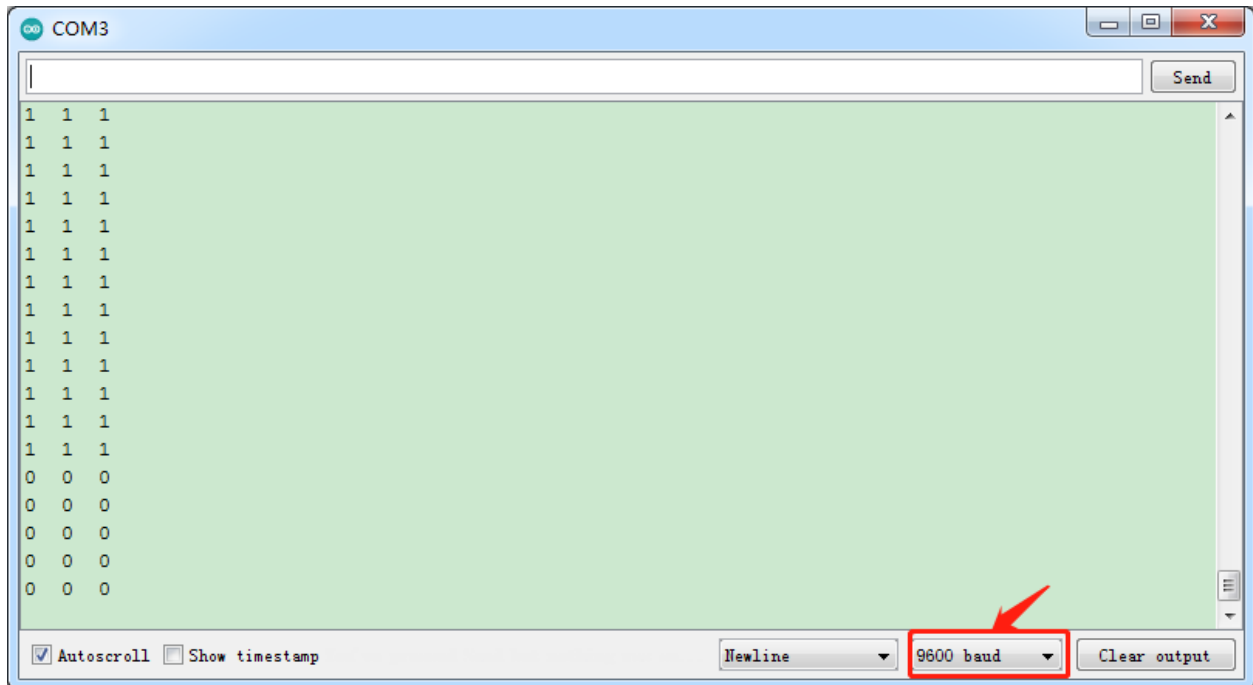
(continued from previous page)

```
pinMode(R_pin, INPUT);  
}  
  
void loop () {  
  int L_val = digitalRead(L_pin); //Read the value of the left sensor  
  int M_val = digitalRead(M_pin); //Read the value of the middle sensor  
  int R_val = digitalRead(R_pin); //Read the value of the right sensor  
  
  Serial.print(L_val);  
  Serial.print(" ");  
  Serial.print(M_val);  
  Serial.print(" ");  
  Serial.print(R_val);  
  Serial.println(" ");  
  delay(100); //delay in 100ms  
}
```

(6)Test Results:

Upload the code on development board, open serial monitor to check line tracking sensors. And the displayed value is 1 (high level) when no signals are received. The value shifts into 0 when the sensor is covered with paper.





(7)Code Explanation:

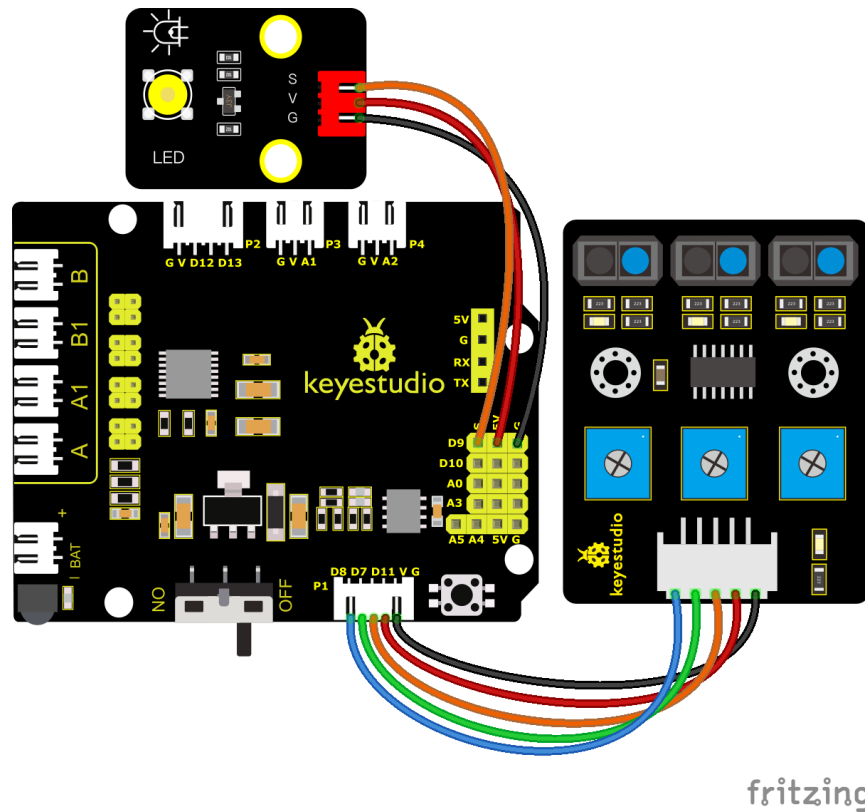
Serial.begin(9600)- Initialize serial port, set baud rate to 9600

pinMode- Define the pin as input or output mode

digitalRead- Read the state of pin, which are generally HIGH and LOW level

(8)Extension Practice:

After knowing its working principle, you can connect an LED to D9. so as to control LED by line tracking sensor.



fritzing

Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

Keyestudio Mini Tank Robot V3 (Popular Edition)

lesson 4.2

Line Track sensor

<http://www.keyestudio.com>

*/

//LED pin

#define LED 9

//The wiring of line tracking sensors

#define L_pin 11 //left

#define M_pin 7 //middle

#define R_pin 8 //right

void setup(){

Serial.begin(9600); //Set the baud rate to 9600

pinMode(LED, OUTPUT); //Set LED to output mode

pinMode(L_pin, INPUT); //Set all pins of the line tracking sensors to input mode

(continues on next page)

(continued from previous page)

```

pinMode(M_pin, INPUT);
pinMode(R_pin, INPUT);
}

void loop (){
    int L_val = digitalRead(L_pin); //Read the value of the left sensor
    int M_val = digitalRead(M_pin); //Read the value of the middle sensor
    int R_val = digitalRead(R_pin); //Read the value of the right sensor
    Serial.print(L_val);
    Serial.print(" ");
    Serial.print(M_val);
    Serial.print(" ");
    Serial.print(R_val);
    Serial.println(" ");
    delay(100); //Delay in 100ms

    if (L_val == 0 || M_val == 0 || R_val == 0) {
        digitalWrite(LED, HIGH);
    }
    else {
        digitalWrite(LED, LOW);
    }
}

```

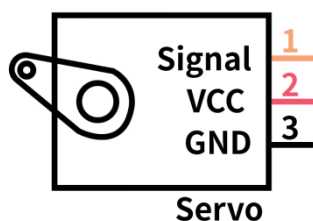
6.3.5 Project 5: Servo Control

(1)Description:

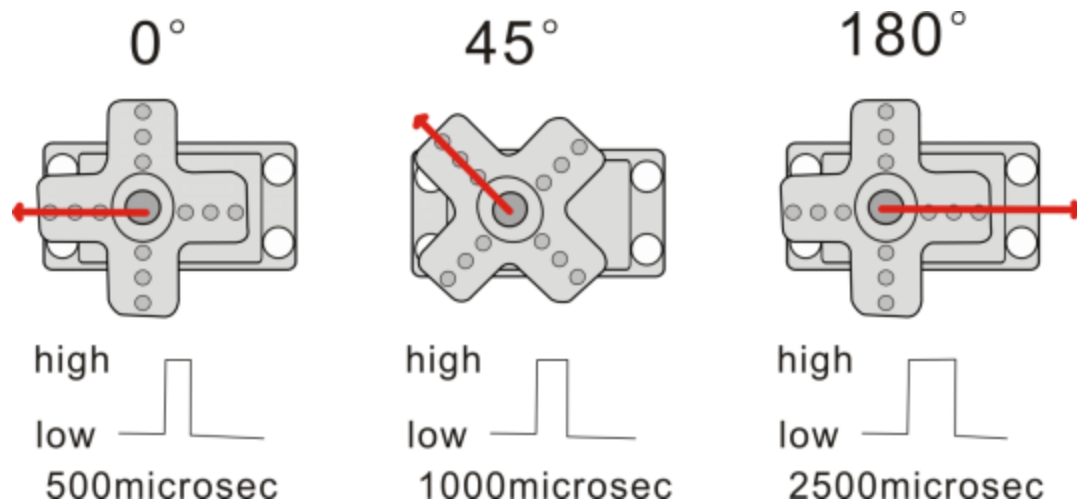
Servo motor is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor. Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of 1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and obtain the voltage difference output.

When the motor speed is constant, the potentiometer is driven to rotate through the cascade reduction gear, which leads that the voltage difference is 0, and the motor stops rotating. Generally, the angle range of servo rotation is 0° – 180°

The rotation angle of servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms (50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle from 0° to 180° . But note that for different brand motors, the same signal may have different rotation angles.



In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.



The angle of the servo:

High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree

(2)Parameters:

- Working voltage: DC 4.8V ~ 6V
- Operating angle range: about 180 ° (at 500 → 2500 sec)
- Pulse width range: 500 → 2500 sec
- No-load speed: 0.12 ± 0.01 sec / 60 (DC 4.8V) 0.1 ± 0.01 sec / 60 (DC 6V)
- No-load current: 200 ± 20mA (DC 4.8V) 220 ± 20mA (DC 6V)
- Stopping torque: 1.3 ± 0.01kg · cm (DC 4.8V) 1.5 ± 0.1kg · cm (DC 6V)
- Stop current: 850mA (DC 4.8V) 1000mA (DC 6V)
- Standby current: 3 ± 1mA (DC 4.8V) 4 ± 1mA (DC 6V)

(continues on next page)

(continued from previous page)

```

void setup() {
    pinMode(servoPin, OUTPUT); //Set the pin of servo as output
    procedure(0); //Set the angle of servo to 0°
}

void loop() {
    for (pos = 0; pos <= 180; pos += 1) { // From 1° to 180°
        // in steps of 1 degree
        procedure(pos); // Rotate to the angle of 'pos'
        delay(15); //Control the speed of rotation
    }
    for (pos = 180; pos >= 0; pos -= 1) { // From 180° to 1°
        procedure(pos); // Rotate to the angle of 'pos'
        delay(15);
    }
}

//The function controls the servo
void procedure(int myangle) {
    pulsewidth = myangle * 11 + 500; //Calculate the value of pulse width
    digitalWrite(servoPin, HIGH);
    delayMicroseconds(pulsewidth); //The time in high level represents the pulse width
    digitalWrite(servoPin, LOW);
    delay((20 - pulsewidth / 1000)); //As the cycle is 20ms, the time left is in low
    ↪ level
}

```

Upload code, we will see the servo move from 0° to 180°. In the further chapters, we will introduce how to drive a servo. Additionally, we can control a servo with a servo library of Arduino.

Note: This servo library file uses timer 1, and the PWM output of IO ports 9 and 10 also uses timer 1, so we cannot use this servo library when using the PWM output of D9 and D10 later.

You can refer to the use of the servo library: <https://www.arduino.cc/en/Reference/Servo>.

(6)Test Code2:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 5.2
Servo
<http://www.keyestudio.com>
*/

#include <Servo.h>

Servo myservo; // create servos
int pos = 0; // Save the variables of angle

```

(continues on next page)

(continued from previous page)

```

void setup() {
    myservo.attach(10); //Connect the servo with digital port 10
}

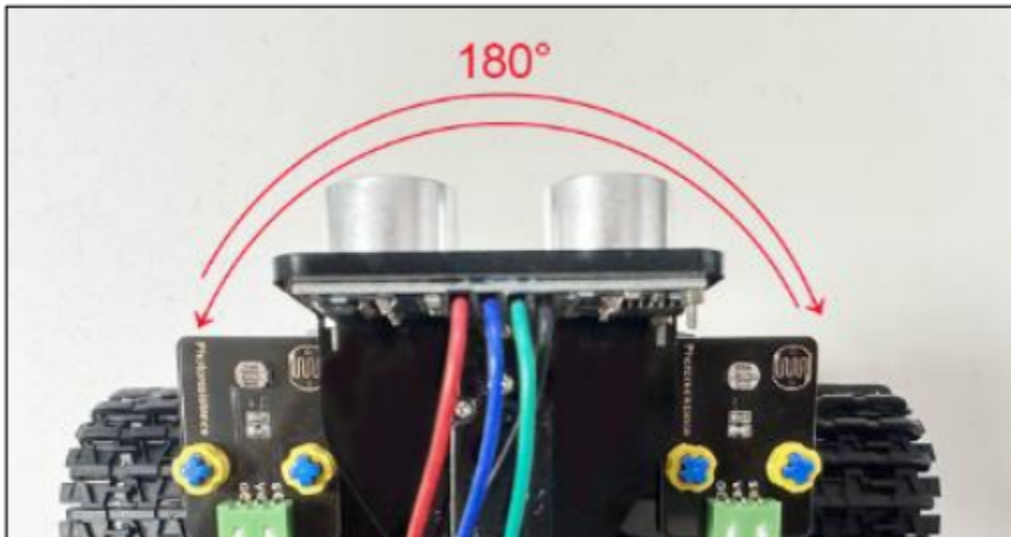
void loop() {
    for (pos = 0; pos <= 180; pos += 1) { //From 0° to 180°
        //step length is 1
        myservo.write(pos); // Rotate to the angle of 'pos'
        delay(15); // Wait for 15ms to control speed
    }

    for (pos = 180; pos >= 0; pos -= 1) { //From 180° to 0°
        myservo.write(pos); // Rotate to the angle of 'pos'
        delay(15); // Wait for 15ms to control speed
    }
}

```

(7)Test Results:

Upload code, plug in power and servo moves in the range of 0° and 180°.



(8)Code Explanation:

Arduino comes with **#include <Servo.h>** (servo function and statement

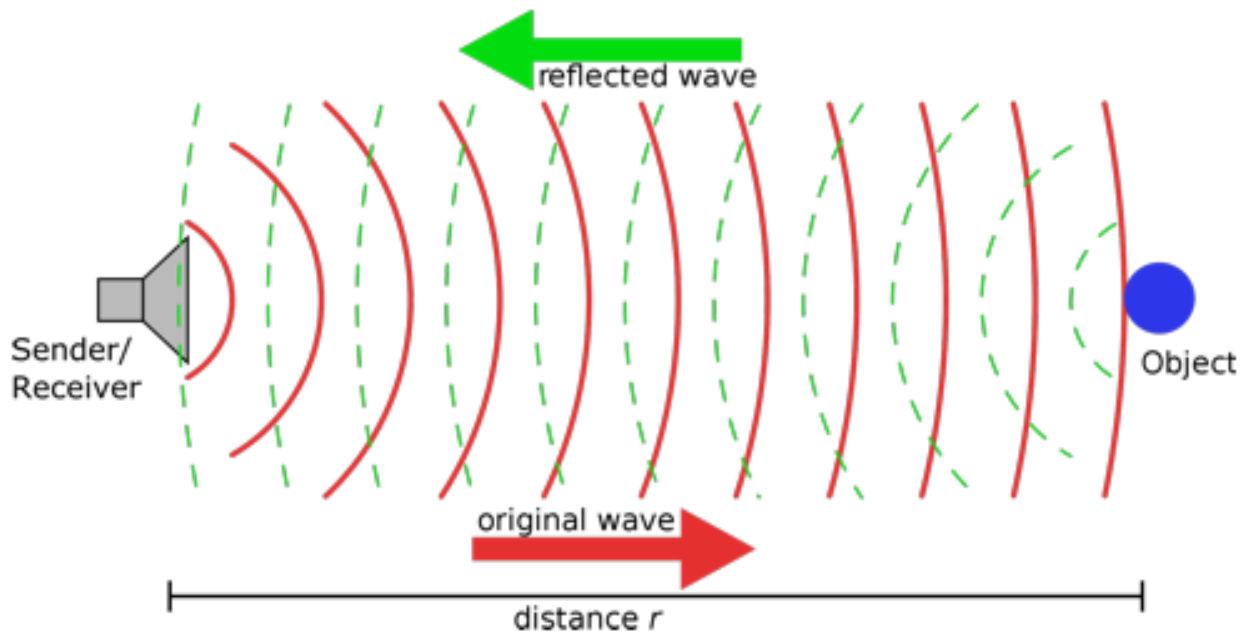
The following are some common statements of the servo function:

1. **attachinterface**——Set servo interface, port 9 and 10 are available
2. **writeangle**——The statement to set rotation angle of servo, the angle range is from 0° to 180°
3. **read**——The statement to read angle of servo, read the command value of “write()”
4. **attached**——Judge if the parameter of servo is sent to its interface

Note: The above written format is “servo variable name, specific statement”, for instance: `myservo.attach(10)`

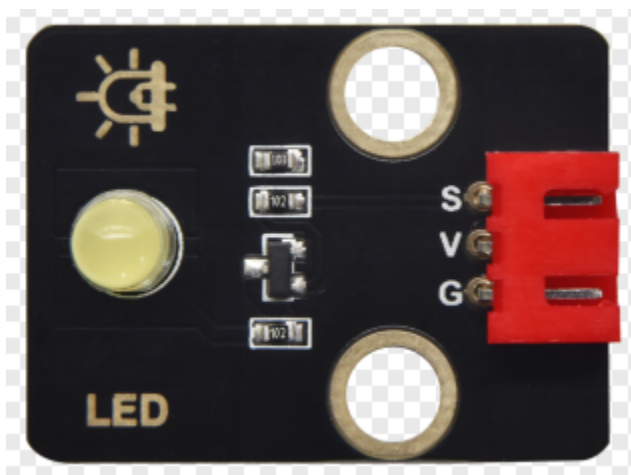
6.3.6 Project 6: Ultrasonic Sensor

(1) Description:



The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like what bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.

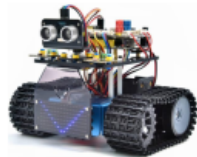


The HC-SR04 or the ultrasonic sensor is being used in a wide range of electronics projects for creating obstacle detection and distance measuring application as well as various other applications. Here we have brought the simple method to measure the distance with Arduino and ultrasonic sensor and how to use ultrasonic sensor with Arduino.



(2)Parameters:

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS

(3)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1
		

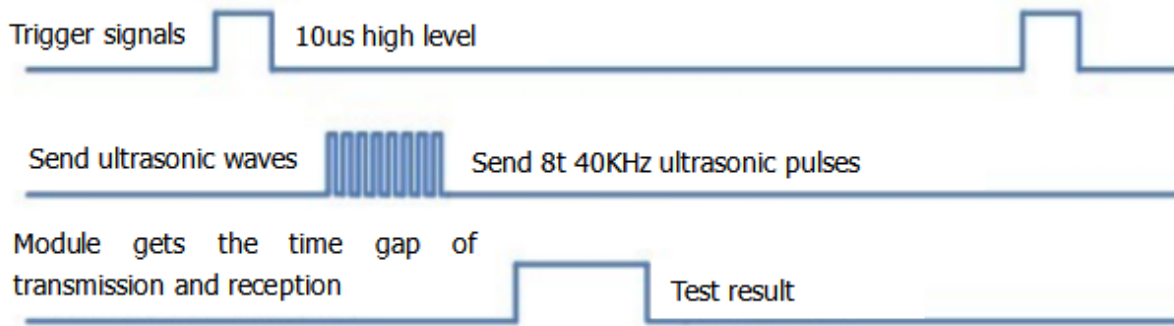
(4) The principle of ultrasonic sensor:

As the above picture shown, it is like two eyes. One is transmitting end, the other is receiving end.

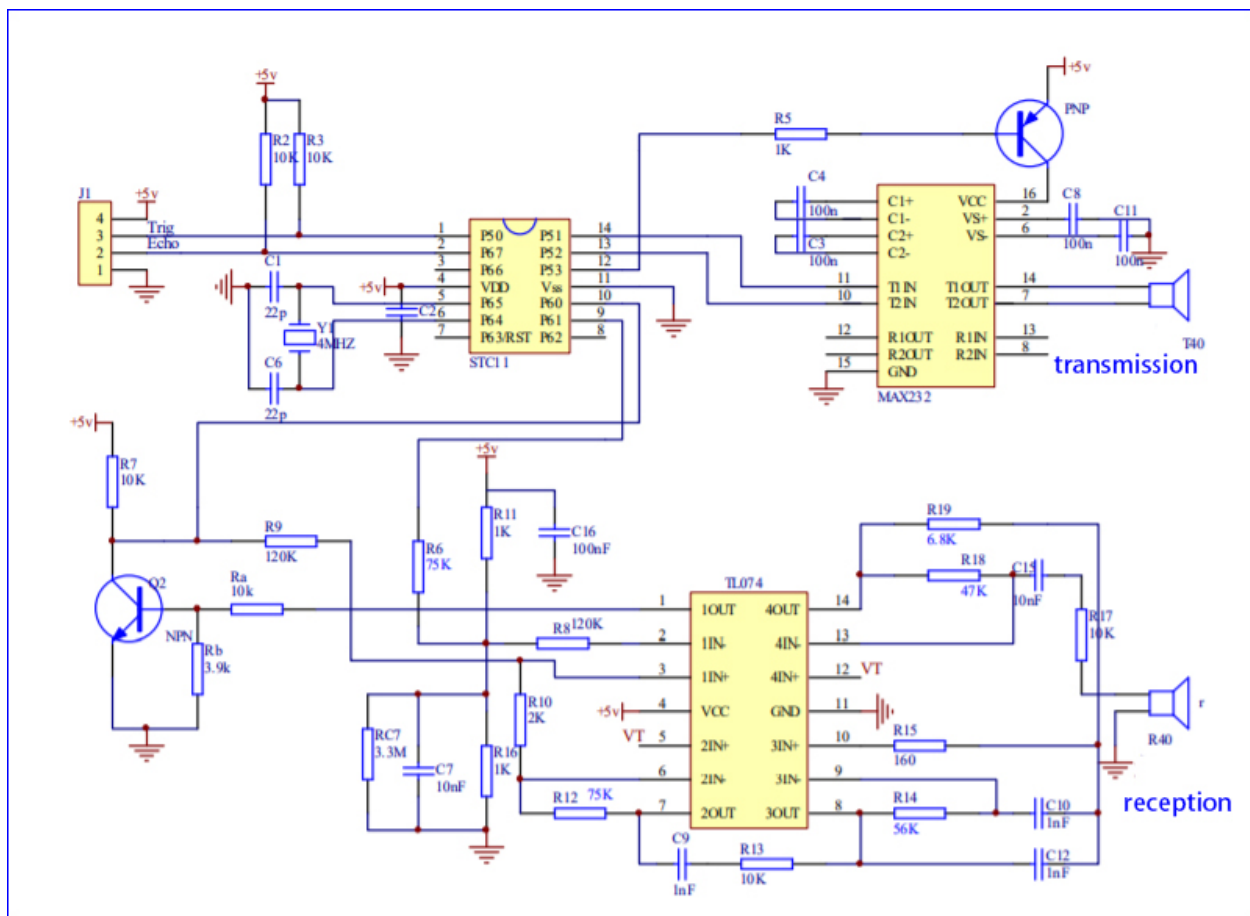
The ultrasonic module will emit the ultrasonic waves after triggering a signal. When the ultrasonic waves encounter the object and are reflected back, the module outputs an echo signal, so it can determine the distance of the object from the time difference between the trigger signal and echo signal.

The t is the time that emitting signal meets obstacle and returns. And the propagation speed of sound in the air is about 343m/s, and distance = speed * time. However, the ultrasonic wave emits and comes back, which is 2 times of distance. Therefore, it needs to be divided by 2, the distance measured by ultrasonic wave = (speed * time)/2

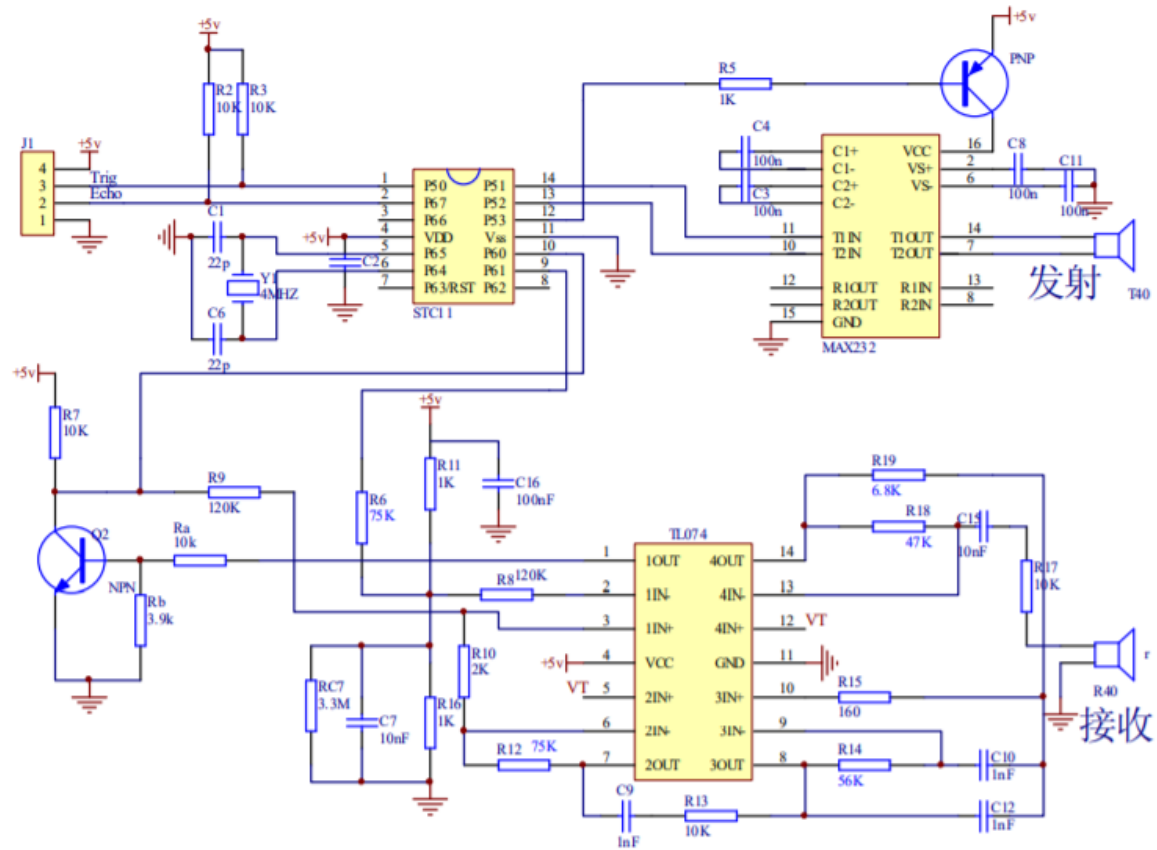
1. Use method and timing chart of ultrasonic module:
2. Setting the delay time of Trig pin of SR04 to 10s at least, which can trigger it to detect distance.
3. After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step will be completed automatically by the module.
4. If the signal returns, the Echo pin will output a high level, and the duration of the high level is the time from the transmission of the ultrasonic wave to the return.



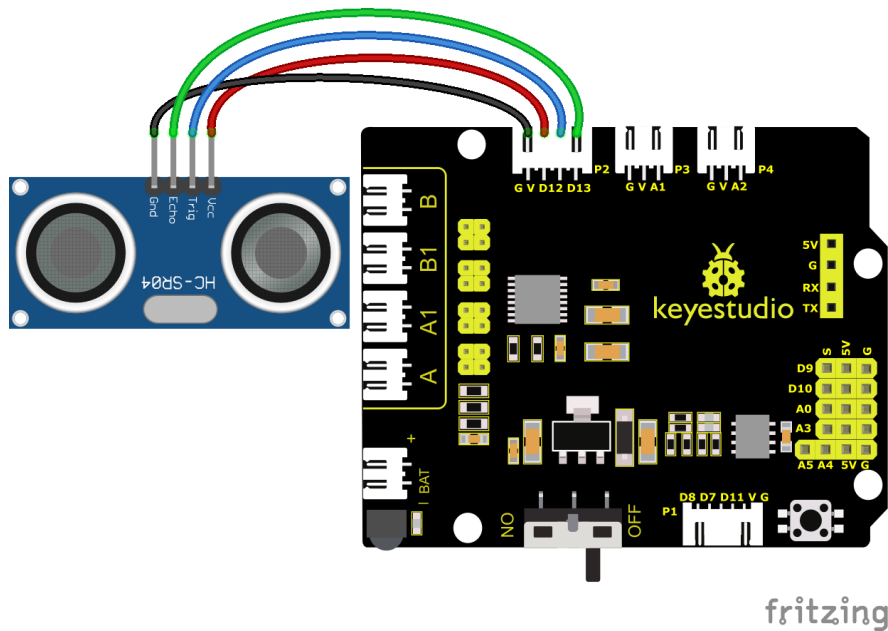
Circuit diagram of ultrasonic sensor:



(5) Connection Diagram:



(6)Connection Diagram:



Note: The pin VCC, Trig, Echo and Gnd of the ultrasonic sensor are respectively connected to 5v(V), 12(S), 13(S) and Gnd(G) of the shield

(7)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 6.1
Ultrasonic sensor
http://www.keyestudio.com
*/

int trigPin = 12; // Pin Trig attaches to 12
int echoPin = 13; //Pin Echo attaches to 13
long duration, cm, inches;

void setup() {
  //Serial Port begin
  Serial.begin(9600); //Set the baud rate to 9600
  //Define input and output
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  //Pre-given a short low pulse to ensure a clean high pulse

```

(continues on next page)

(continued from previous page)

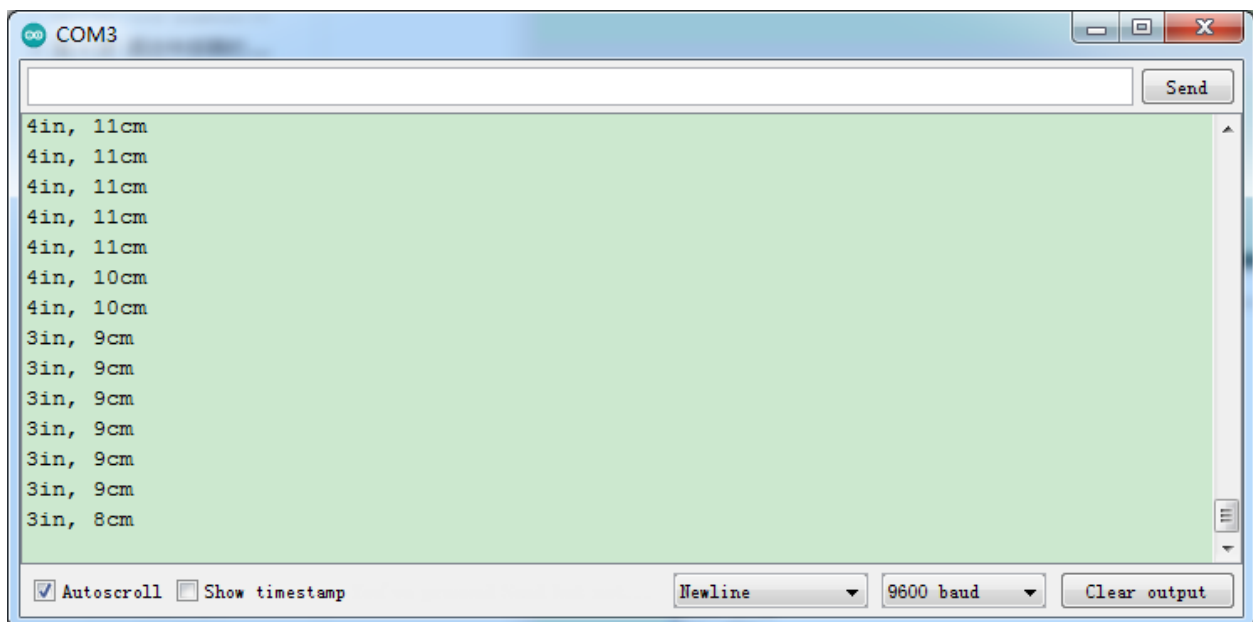
```

digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); //At least give 10us high level trigger
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
// The time in high level equals the time gap between the transmission and the
return of the ultrasonic sound
duration = pulseIn(echoPin, HIGH);
// Translate into distance
cm = (duration / 2) / 29.1; // convert to centimeters
inches = (duration / 2) / 74; // Convert to inch
//serial port prints out
Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();
delay(50);
}

```

(8)Test Results:

Upload test code on the development board, open serial monitor and set baud rate to 9600. The detected distance will be displayed, and the unit is cm and inch. Hinder the ultrasonic sensor by hand, the displayed distance value gets smaller.



(9)Code Explanation:

int trigPin = 12; this pin is defined to transmit ultrasonic waves, generally output.

int echoPin = 13; this is defined as the pin of reception, generally input

cm = (duration/2) / 29.1; inches = (duration/2) / 74; by 0.0135

We can calculate the distance by using the following formula:

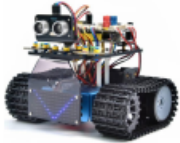




$\text{distance} = (\text{traveltime}/2) \times \text{speed of sound}$

The speed of sound is: $343\text{m/s} = 0.0343 \text{ cm/uS} = 1/29.1 \text{ cm/uS}$

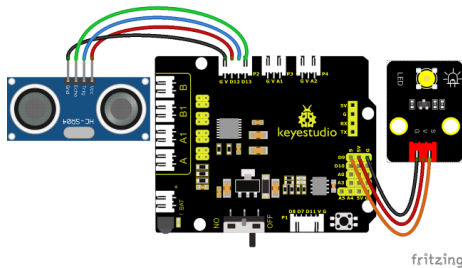
Or in inches: $13503.9\text{in/s} = 0.0135\text{in/uS} = 1/74\text{in/uS}$

We need to divide the traveltime by 2 because we have to take into account that the wave was sent, hit the object, and then returned back to the sensor.

(10)Extension Practice:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1	3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1
				

We have just measured the distance displayed by the ultrasonic. How about controlling the LED with the measured distance? Let's try it and connect an LED light module to the D9 pin.



Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 6.2
Ultrasonic LED
http://www.keyestudio.com
*/

```

(continues on next page)

(continued from previous page)

```

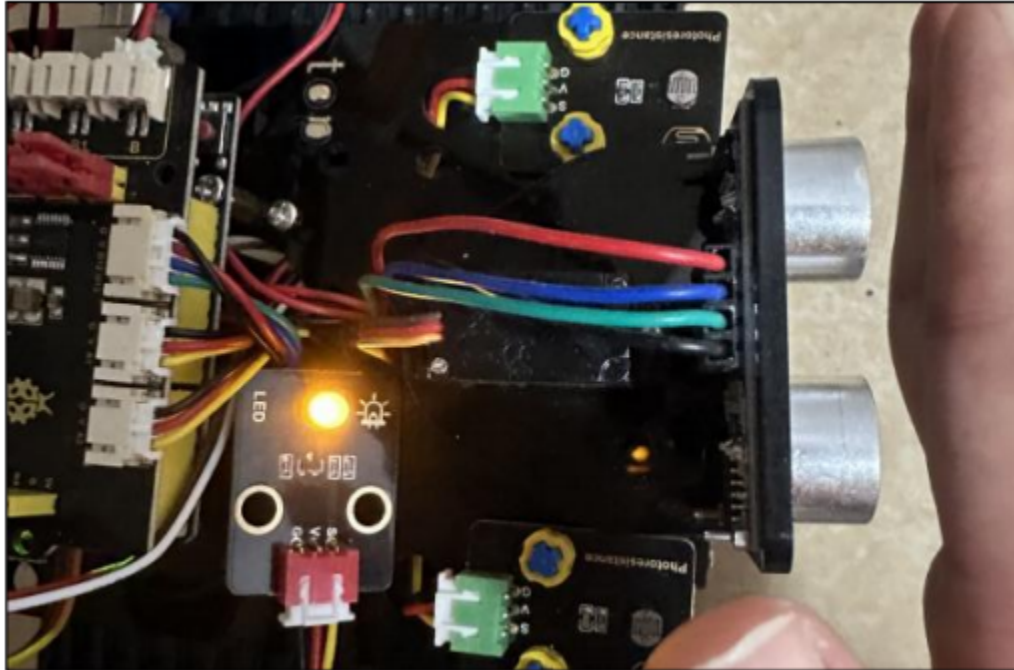
int trigPin = 12; // Trig is connected to 12
int echoPin = 13; // Echo is connected to 13
int LED = 9;
long duration, cm, inches;

void setup() {
  //start serial port
  Serial.begin (9600); //set baud rate to 9600
  //define input and output
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(LED, OUTPUT);
}

void loop() {
  //Pre-given a short low pulse to ensure a clean high pulse
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH); //Give at least 10us high level trigger
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // The duration of the high level is the time from the launch to the return of the
  ↳ ultrasonic wave
  duration = pulseIn(echoPin, HIGH);
  // convert to distance
  cm = (duration / 2) / 29.1; // convert to centimeters
  inches = (duration / 2) / 74; // convert to inches
  //serial port prints out
  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  if (cm >= 2 && cm <= 10) {
    digitalWrite(LED, HIGH); //turn on LED
  } else {
    digitalWrite(LED, LOW); //turn off LED
  }
  delay(50);
}

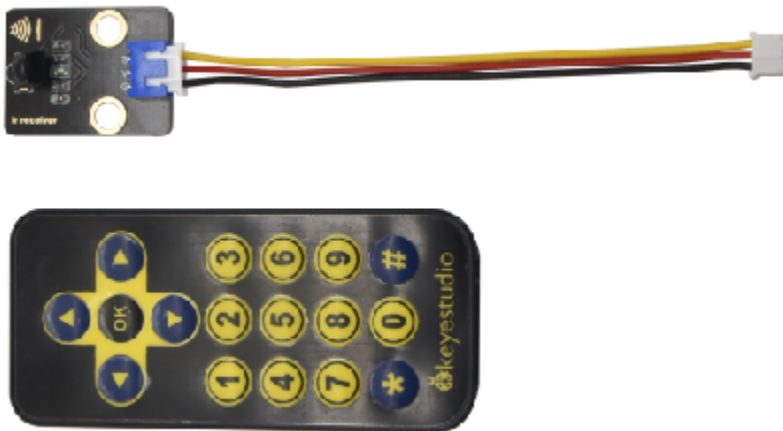
```

Upload test code to development board and block ultrasonic sensor by hand, then check if LED is on



6.3.7 Project 7: IR Reception

(1)Description:



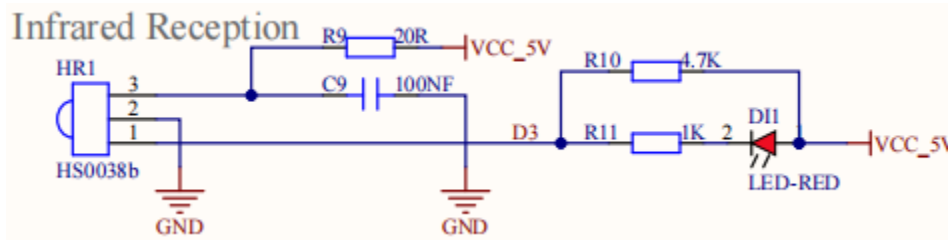
There is no doubt that infrared remote control is ubiquitous in daily life. It is used to control various household appliances, such as TVs, stereos, video recorders and satellite signal receivers. Infrared remote control is composed of infrared transmitting and infrared receiving systems, that is, an infrared remote control and infrared receiving module and a single-chip microcomputer capable of decoding.

The 38K infrared carrier signal emitted by remote controller is encoded by the encoding chip in the remote controller. It is composed of a section of pilot code, user code, user inverse code, data code, and data inverse code. The time interval of the pulse is used to distinguish whether it is a 0 or 1 signal and the encoding is made up of these 0, 1 signals.

The user code of the same remote control is unchanged while the data code can distinguish the key.

When the remote control button is pressed, the remote control sends out an infrared carrier signal. When the IR receiver

receives the signal, the program will decode the carrier signal and determines which key is pressed. The MCU decodes the received 01 signal, thereby judging what key is pressed by the remote control.



Infrared receiver we use is an infrared receiver module. Mainly composed of an infrared receiver head, which is a device that integrates reception, amplification, and demodulation. Its internal IC has completed demodulation, and can achieve from infrared reception to output and be compatible with TTL signals.

Additionally, it is suitable for infrared remote control and infrared data transmission. The infrared receiving module made by the receiver has only three pins, signal line, VCC and GND. It is very convenient to communicate with Arduino and other microcontrollers.

(2)Parameters:

Operating Voltage: 3.3-5VDC

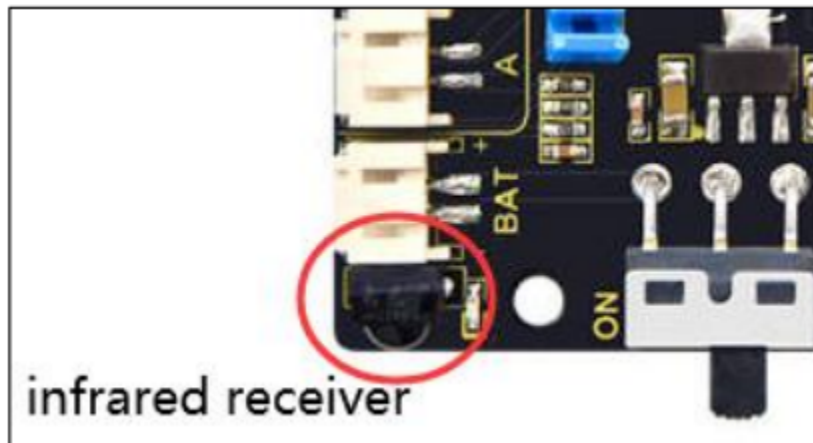
Interface: 3PIN

Output Signal: Digital signal

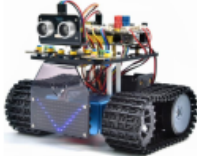

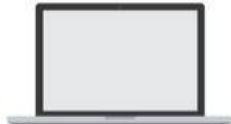

Receiving Angle: 90 degrees

Frequency: 38khz

Receiving Distance: 10m



(3) Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	Remote Control*1
			

(4) Test Code:

You need to import the IR library first.

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 7.1
IRremote
http://www.keyestudio.com
*/

#include <IRremote.h> // IRremote library statement

int RECV_PIN = 3; //define the pins of IR receiver as 3
IRrecv irrecv(RECV_PIN);
decode_results results; //decode results exist in the "result" of "decode results"

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn(); //Enable receiver
}

void loop() {
  if (irrecv.decode(&results)) //decode successfully, receive a set of infrared signals
  {
    Serial.println(results.value, HEX); //Wrap word in 16 HEX to output and receive
    ↪code
    irrecv.resume(); //Receive the next value
  }
}

```

(continues on next page)

(continued from previous page)

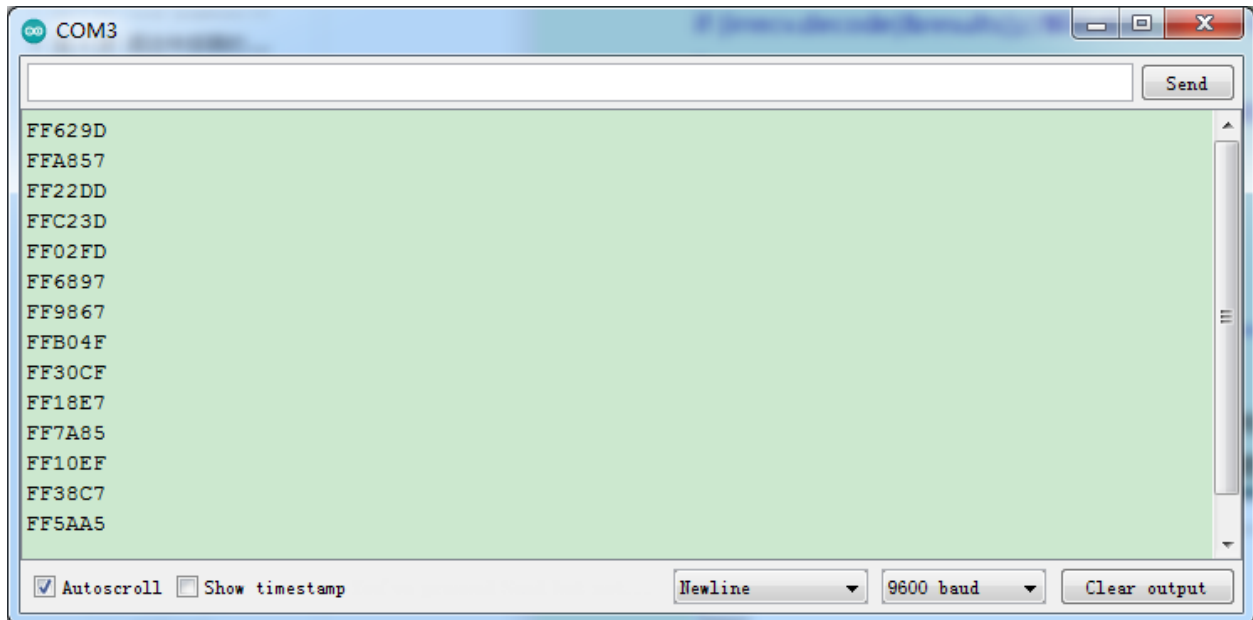
```

    delay(100);
}

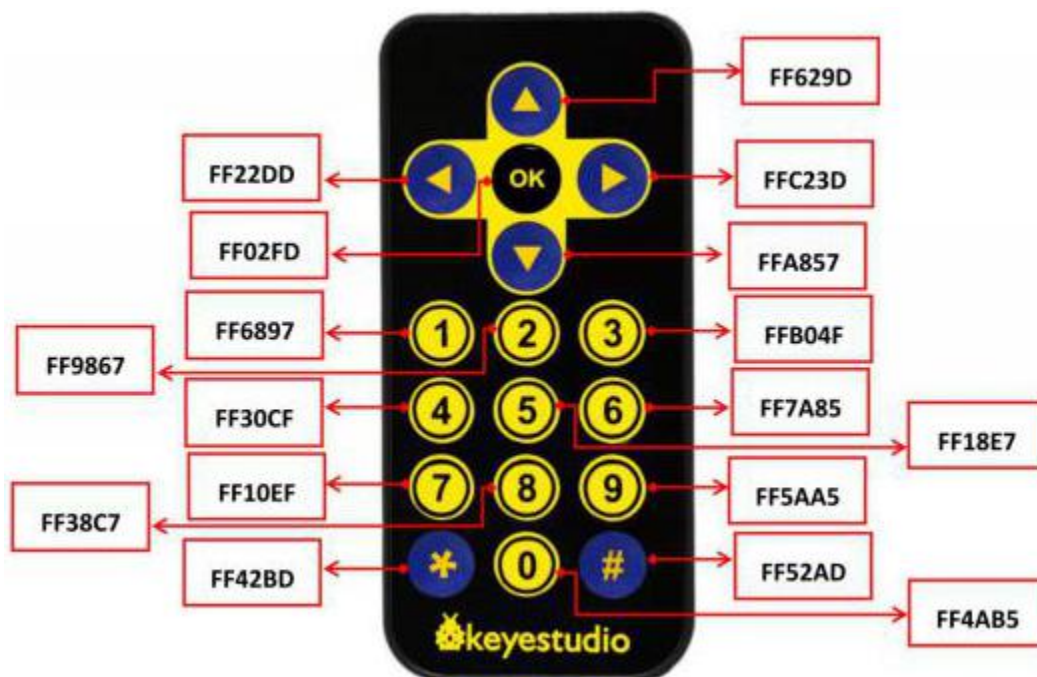
```

(5)Test Results:

Upload test code, open serial monitor and set baud rate to 9600, point remote control to IR receiver. Then the corresponding value will be shown. If holding down keys for a while, the error codes will appear.



Below we have listed out each button value of keystudio remote control. So you can keep it for reference.



The IR receiver is connected to D3, so we don't need to wire up

(6)Code Explanation:

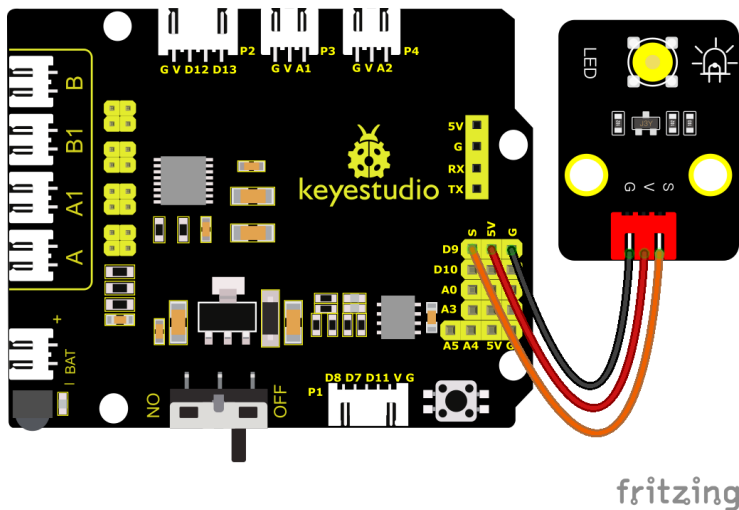
irrecv.enableIRIn(); after enabling IR decoding, the IR signals will be received, then function“decode()”will check continuously if decode successfully.

irrecv.decode(&results); after decoding successfully, this function will come back to “true”, and keep result in “results”. After decoding a IR signals, run the resume()function and receive the next signal.

(7)Extension Practice:

We decoded the key value of IR remote control. How about controlling LED by the measured value? We could design an experiment.

Attach an LED to D9, then press the keys of remote control to make LED light on and off.



Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 7.2
IRremote
http://www.keyestudio.com
*/

#include <IRremote.h> //IRremote statement

int RECV_PIN = 3; //define the pin of LED as pin 3
int LED = 9;
bool flag = 0;
IRrecv irrecv(RECV_PIN);
decode_results results; //decode results exist in the“result” of “decode results
```

(continues on next page)

(continued from previous page)

```

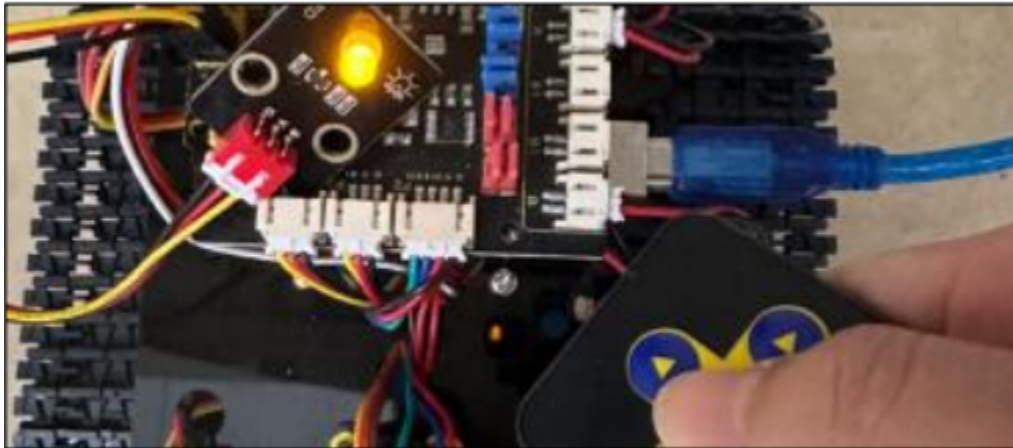
void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT); //set LED to output
  irrecv.enableIRIn(); //Enable receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    if (results.value == 0xFF02FD & flag == 0) //The above key code, we use the OK
    ↪ button on the remote control, if you press the OK button
    {
      digitalWrite(LED, HIGH); //LED on
      flag = 1;
    }

    else if (results.value == 0xFF02FD & flag == 1) //press again
    {
      digitalWrite(LED, LOW); //LED off
      flag = 0;
    }
    irrecv.resume(); // Receive the next value
  }
}

```

Upload code to development board, press“OK”key on remote control to make LED on and off.

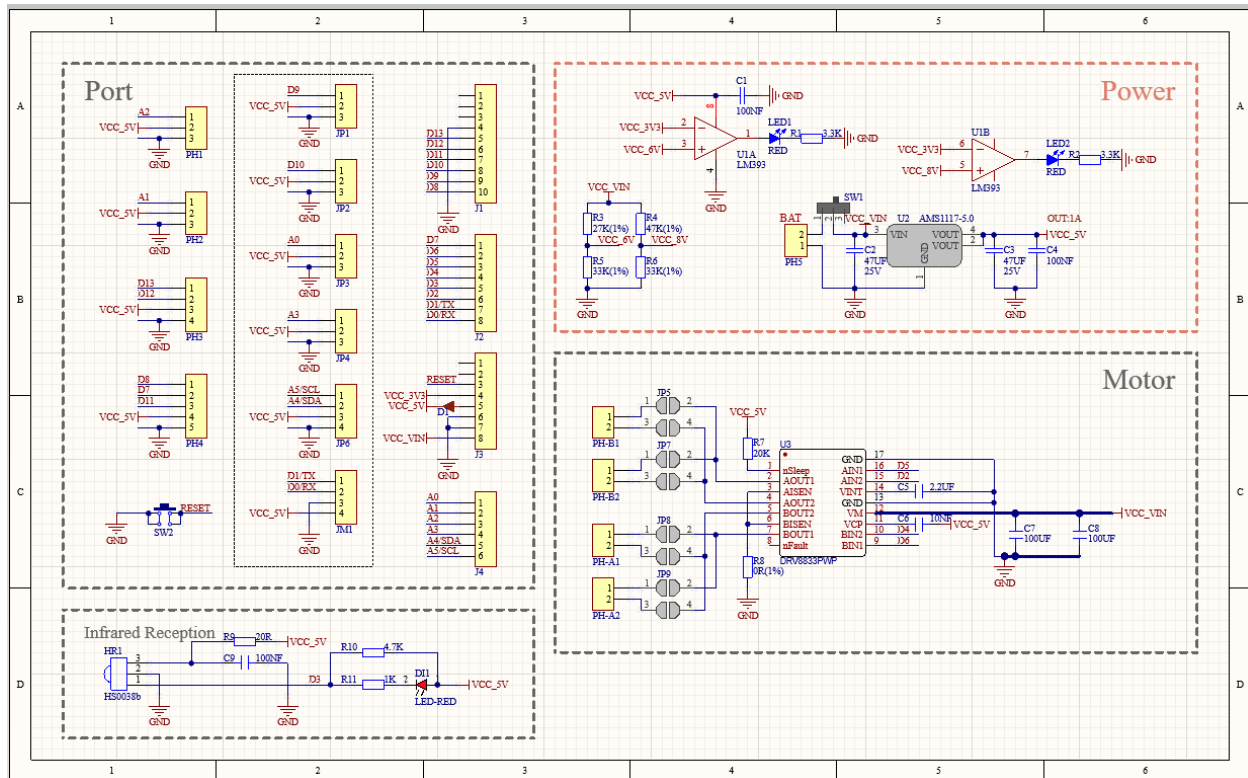


6.3.8 Project 8: Motor Driving and Speed Control

(1)Description:

There are many ways to drive motors. Our smart car uses the most common solution called L298P. L298P, produced by STMicroelectronics, is an excellent driving chip specially designed for driving high-power motors .

It can directly drive DC motors, two-phase and four-phase motors with the driving current reaching 2A. And the motor's output terminal adopts 8 high-speed Schottky diodes as protection.



(2)Parameters

- Logic part input voltage: DC 5V
- Driving part input voltage: DC 7-12V
- Logic part working current: 36mA
- Driving part working current: 2A
- Maximum dissipation power: 25W (T=75°C)
- Control signal input level:
 - High level: 2.3V Vin 5V
 - Low level: 0V Vin 1.5V
- Working temperature: -25°C130°C

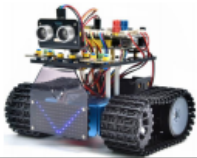



(3)Drive the robot to move

The direction pin of A motor is D2, the speed control pin is D5; the direction pin of B motor is in D4 and the speed control pin is D6,

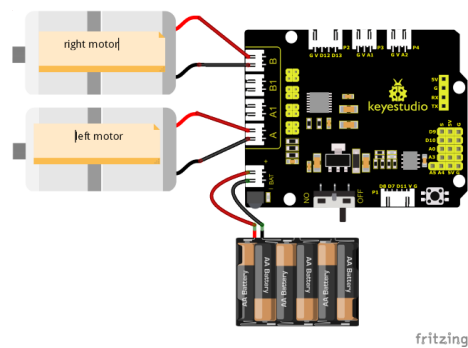
According to the table below, we can know how to control the movement of the robot by controlling the rotation of two motors through the digital ports and PWM ports . Among them, the range of PWM value is 0-255. The larger the value is, the faster the motor rotates.

Function	D4	D6PWM	Motor leftB	D2	D5PWM	MotorRightA
Move Forward	HIGH	255-200	Rotate Left	HIGH	255-200	Rotate Left
Go Back	LOW	200	Rotate Right	LOW	200	Rotate Right
Rotate Left	LOW	200	Rotate Right	HIGH	255-200	Rotate Left
Rotate Right	HIGH	255-200	Rotate Left	LOW	200	Rotate Right
Stop	LOW	0	Stop	LOW	0	Stop

(4)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(5)Connection Diagram:



Note:

The 4-pin connector is marked with A, A1, B1 and B. The right rear motor is connected to B of the 8833 board and left front one is connected to A port.

(6)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 8.1
motor driver
http://www.keyestudio.com
*/

```

(continues on next page)

(continued from previous page)

```

#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor

void setup()
{
    pinMode(ML_Ctrl, OUTPUT); //Define the direction control pin of the left motor as
    ↪ OUTPUT
    pinMode(ML_PWM, OUTPUT); //Define the PWM control pin of the left motor as OUTPUT
    pinMode(MR_Ctrl, OUTPUT); //Define the direction control pin of the right motor as
    ↪ OUTPUT
    pinMode(MR_PWM, OUTPUT); //Define the PWM control pin of the right motor as OUTPUT
}

void loop()
{
    //front
    digitalWrite(ML_Ctrl, HIGH); //Set direction control speed of the left motor to HIGH
    analogWrite(ML_PWM, 55); //PWM control speed of the left motor is 200
    digitalWrite(MR_Ctrl, HIGH); //Set direction control speed of the right motor to HIGH
    analogWrite(MR_PWM, 55); //PWM control speed of the right motor is 200
    delay(2000); //delay in 2s

    //back
    digitalWrite(ML_Ctrl, LOW); //Set direction control speed of the left motor to LOW
    analogWrite(ML_PWM, 200); //PWM control speed of the left motor is 200
    digitalWrite(MR_Ctrl, LOW); //Set direction control speed of the right motor to LOW
    analogWrite(MR_PWM, 200); //PWM control speed of the right motor is 200
    delay(2000); //delay in 2s

    //turn left
    digitalWrite(ML_Ctrl, LOW); //Set direction control speed of the left motor to LOW
    analogWrite(ML_PWM, 200); //PWM control speed of the left motor is 200
    digitalWrite(MR_Ctrl, HIGH); //Set direction control speed of the right motor to HIGH
    analogWrite(MR_PWM, 55); //PWM control speed of the right motor is 200
    delay(2000); //delay in 2s

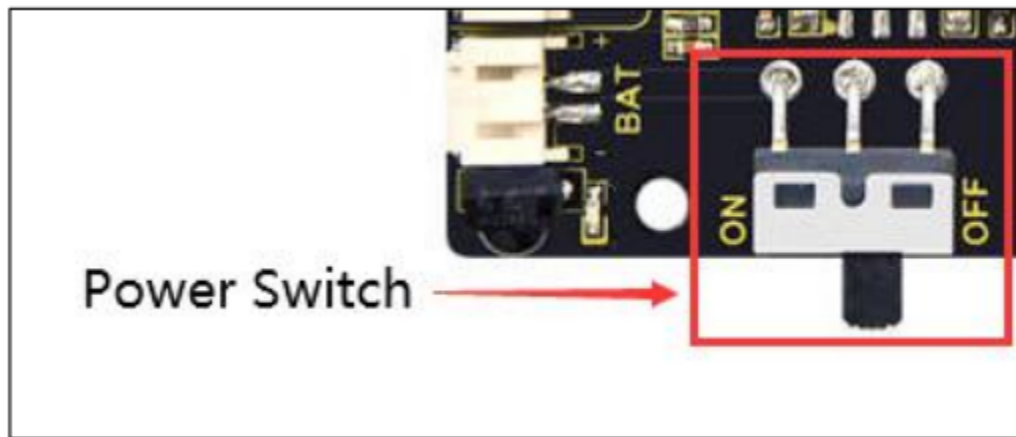
    //turn right
    digitalWrite(ML_Ctrl, HIGH); //Set direction control speed of the left motor to HIGH
    analogWrite(ML_PWM, 55); //PWM control speed of the left motor is 200
    digitalWrite(MR_Ctrl, LOW); //Set direction control speed of the right motor to LOW
    analogWrite(MR_PWM, 200); //PWM control speed of the right motor is 200
    delay(2000); //delay in 2s

    //stop
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0); //PWM control speed of the left motor is 0
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0); //PWM control speed of the right motor is 0
    delay(2000); //delay in 2s
}

```

(7)Test Results:

After wiring according to the diagram, uploading the test code and powering it up.



the smart car moves forward for 2s, steps back for 2s, turns left for 2s, turns right for 2s and stops for 2s and repeats this sequence.

(8)Code Explanation:

digitalWrite(ML_Ctrl,LOW);

The change between high and low levels can makes motors to rotate clockwise or anticlockwise. General digital pins can be used to control these movements.

analogWrite(ML_PWM,200);

The speed adjustment of the motor is realized by PWM, and the pin that controls the speed of the motor must be the PWM pin of Arduino.

(9)Expansion Project:

We adjust the speed of motors by controlling PWM and the wiring remains the same.

Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 8.2
motor driver pwm
http://www.keyestudio.com
*/

#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
```

(continues on next page)

(continued from previous page)

```

void setup() {
    pinMode(ML_Ctrl, OUTPUT); //Define the direction control pin of the left motor as OUTPUT
    pinMode(ML_PWM, OUTPUT); //Define the PWM control pin of the left motor as OUTPUT
    pinMode(MR_Ctrl, OUTPUT); //Define the direction control pin of the right motor as OUTPUT
    pinMode(MR_PWM, OUTPUT); //Define the PWM control pin of the right motor as OUTPUT
}

void loop() {
    //front
    digitalWrite(ML_Ctrl, HIGH); //Set direction control speed of the left motor to HIGH
    analogWrite(ML_PWM, 155); //PWM control speed of the left motor is 100
    digitalWrite(MR_Ctrl, HIGH); //Set direction control speed of the right motor to HIGH
    analogWrite(MR_PWM, 155); //PWM control speed of the right motor is 100
    delay(2000); //delay in 2s

    //back
    digitalWrite(ML_Ctrl, LOW); //Set direction control speed of the left motor to LOW
    analogWrite(ML_PWM, 100); //PWM control speed of the left motor is 100
    digitalWrite(MR_Ctrl, LOW); //Set direction control speed of the right motor to LOW
    analogWrite(MR_PWM, 100); //PWM control speed of the right motor is 100
    delay(2000); //delay in 2s

    //left
    digitalWrite(ML_Ctrl, LOW); //Set direction control speed of the left motor to LOW
    analogWrite(ML_PWM, 100); //PWM control speed of the left motor is 100
    digitalWrite(MR_Ctrl, HIGH); //Set direction control speed of the right motor to HIGH
    analogWrite(MR_PWM, 155); //PWM control speed of the right motor is 100
    delay(2000); //delay in 2s

    //right
    digitalWrite(ML_Ctrl, HIGH); //Set direction control speed of the left motor to HIGH
    analogWrite(ML_PWM, 155); //PWM control speed of the left motor is 100
    digitalWrite(MR_Ctrl, LOW); //Set direction control speed of the right motor to LOW
    analogWrite(MR_PWM, 100); //PWM control speed of the right motor is 100
    delay(2000); //delay in 2s

    //stop
    digitalWrite(ML_Ctrl, LOW); //Set direction control speed of the left motor to LOW
    analogWrite(ML_PWM, 0); //PWM control speed of the left motor is 0
    digitalWrite(MR_Ctrl, LOW); //Set direction control speed of the right motor to LOW
    analogWrite(MR_PWM, 0); //PWM control speed of the right motor is 0
    delay(2000); //delay in 2s
}

```

Upload the code, the speed of the motor is slower.

Low current will cause the motor to rotate slowly.

6.3.9 Project 9: 8*16 Facial Expression LED Dot Matrix

(1)Description:

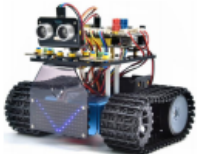

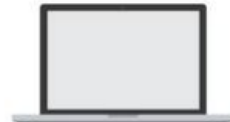

Won't it be fun if a expression board is added to the robot? And the Keystudio 8*16 LED dot matrix can do the trick. With the help of it, you could design facial expressions, images, patterns and other displays by yourselves.

The 8*16 LED board comes with 128 LEDs. The data of the microprocessor (Arduino) communicates with the AiP1640 through a two-wire bus interface. Therefore, it can control the on and off of 128 LEDs on the module, so as to make the dot matrix on the module to display the pattern you need. A HX-2.54 4Pin cable is provided for your convenience of wiring.

(2)Parameters:

- Working voltage: DC 3.3-5V
- Power loss: 400mW
- Oscillation frequency: 450KHz
- Drive current: 200mA
- Working temperature: -40~80°C
- Communication mode: two-wire bus

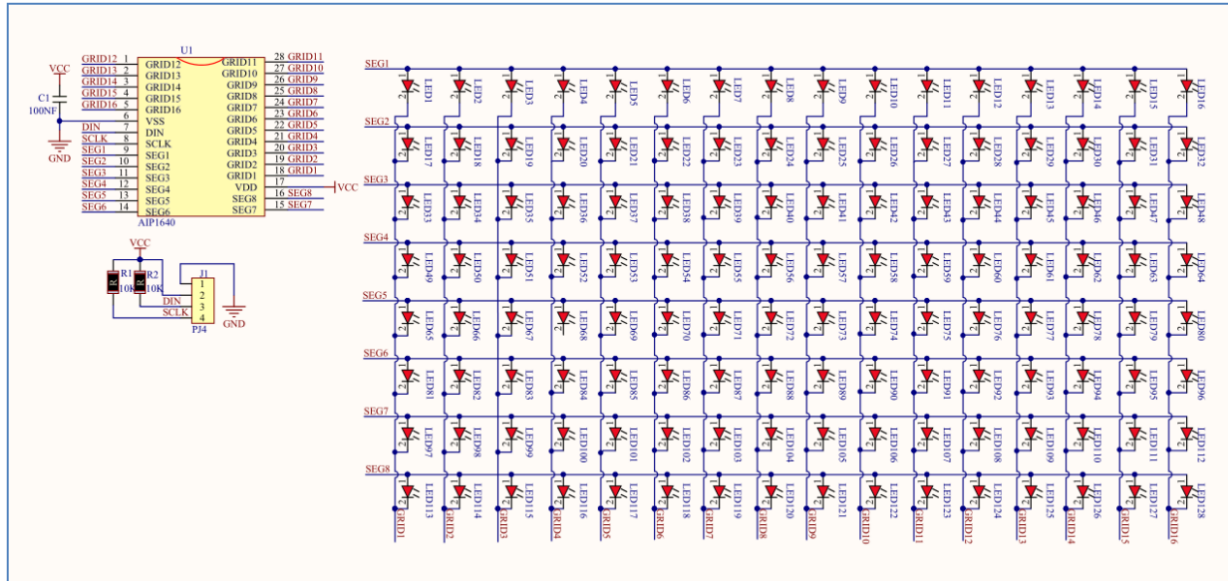
(3)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(4)Knowledge:

Principle of the 8*16 LED dot matrix

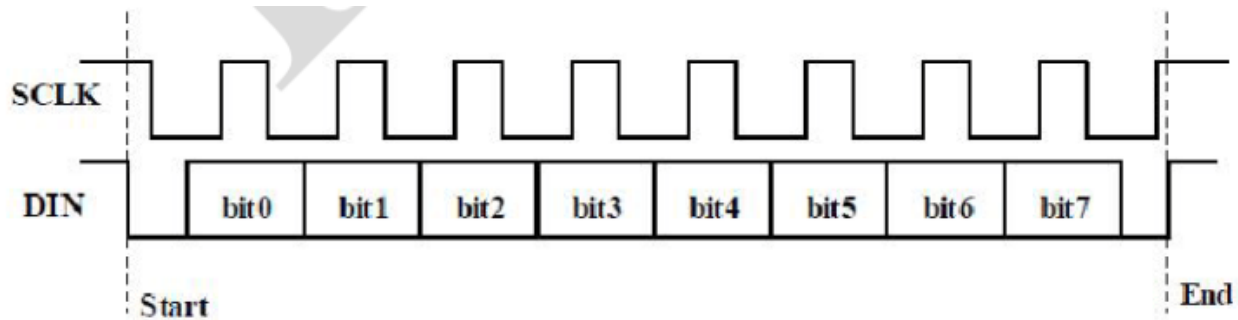
How to control each LED of the 8*16 dot matrix? It is known that each byte has 8 bits and each bit is 0 or 1. when it is 0, LED is off while when it is 1 LED is on. One byte can control one column of the LED, and naturally 16 bytes can control 16 columns of LEDs, that's the 8*16 dot matrix.



Pins description and communication protocol

The data of the microprocessor (Arduino) communicates with the AiP1640 through a two-wire bus cable.

The communication protocol diagram is as follows (SCLK) is SCL, (DIN) is SDA



The starting condition for data input: SCL is high level and SDA changes from high to low.

For data command setting, there are methods as shown in the figure below

In our sample program, select the way to **add 1 to the address automatically**, the binary value is 0100 0000 and the corresponding hexadecimal value is 0x40

B7	B6	B5	B4	B3	B2	B1	B0	Description
0	1	Irrelevant choice, fill in 0			0	Irrelevant choice, fill in 0		add 1 to the address automatically
0	1				1			Fixed address
0	1		0					Universal mode
0	1		1					Test mode (Inner only)

For address command setting, the address can be selected as shown below.

The first 00H is selected in our sample program, and the binary number 1100 0000 corresponds to the hexadecimal 0xc0

B7	B6	B5	B4	B3	B2	B1	B0	Display address
1	1	Irrelevant choice, fill in 0		0	0	0	0	00H
1	1			0	0	0	1	01H
1	1			0	0	1	0	02H
1	1			0	0	1	1	03H
1	1			0	1	0	0	04H
1	1			0	1	0	1	05H
1	1			0	1	1	0	06H
1	1			0	1	1	1	07H
1	1			1	0	0	0	08H
1	1			1	0	0	1	09H
1	1			1	0	1	0	0AH
1	1			1	0	1	1	0BH
1	1			1	1	0	0	0CH
1	1			1	1	0	1	0DH
1	1			1	1	1	0	0EH
1	1			1	1	1	1	0FH

The requirement for data input is that when SCL is at high level when inputting data, the signal on SDA must remain unchanged. Only when the clock signal on SCL is at low level, can the signal on SDA be changed. The input of data is the low bit first, and the high bit later.

The condition for the end of data transmission is that when SCL is at low level, SDA at low level and SCL at high level, the level of SDA becomes high.

Display control, set different pulse width, pulse width can be selected as shown in the figure below

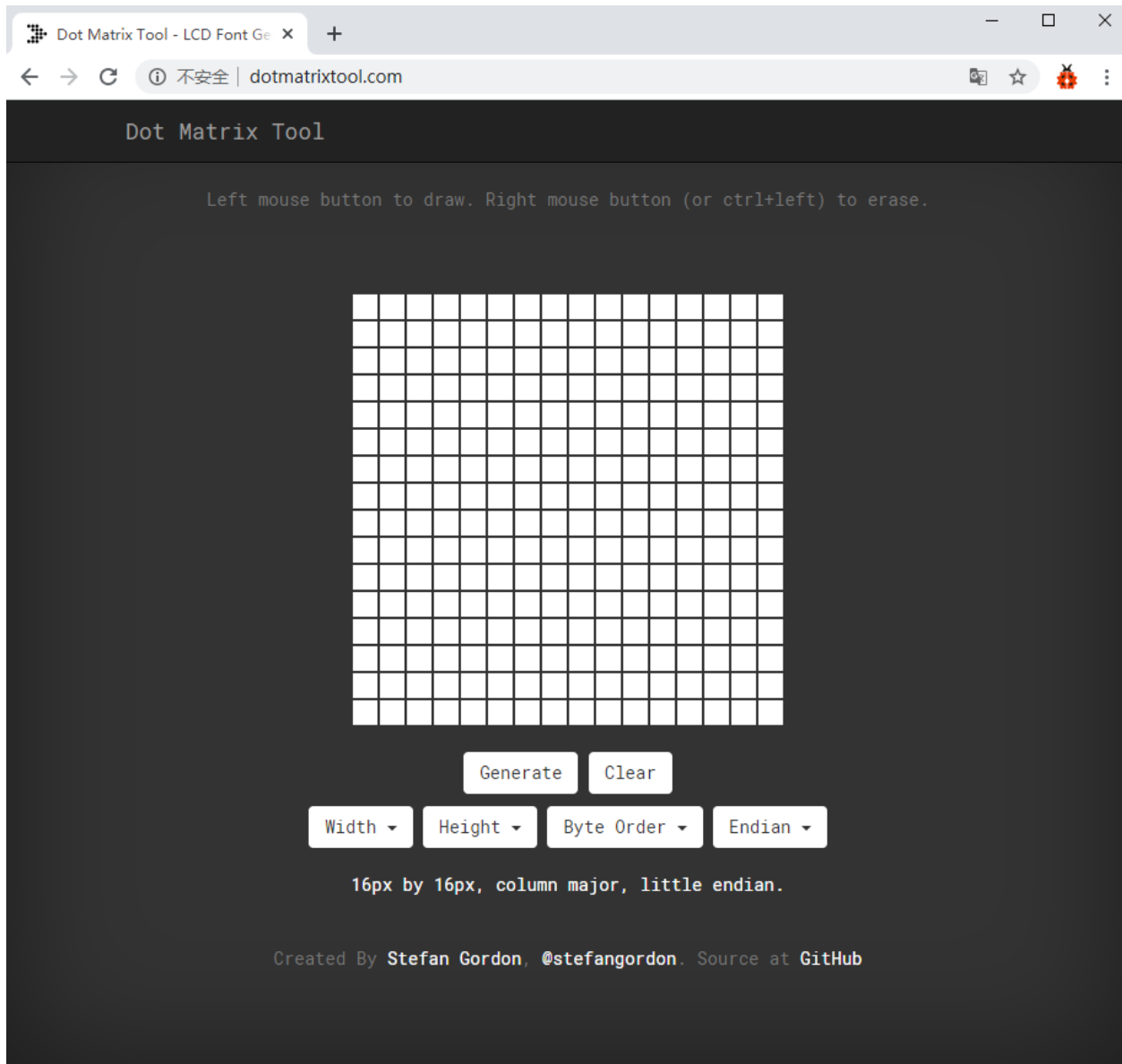
In the example, the pulse width is 4/16, and the hexadecimal corresponding to 1000 1010 is 0x8A

B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
1	0	Irrelevant choice, fill in 0		1	0	0	0	Clear quantity setting (Brightness setting)	Set pulse width to 1/16
1	0			1	0	0	1		Set pulse width to 2/16
1	0			1	0	1	0		Set pulse width to 4/16
1	0			1	0	1	1		Set pulse width to 10/16
1	0			1	1	0	0		Set pulse width to 11/16
1	0			1	1	0	1		Set pulse width to 12/16
1	0			1	1	1	0		Set pulse width to 13/16
1	0			1	1	1	1		Set pulse width to 14/16
1	0			0	X	X	X	Display switch setting	On
1	0			1	X	X	X		off

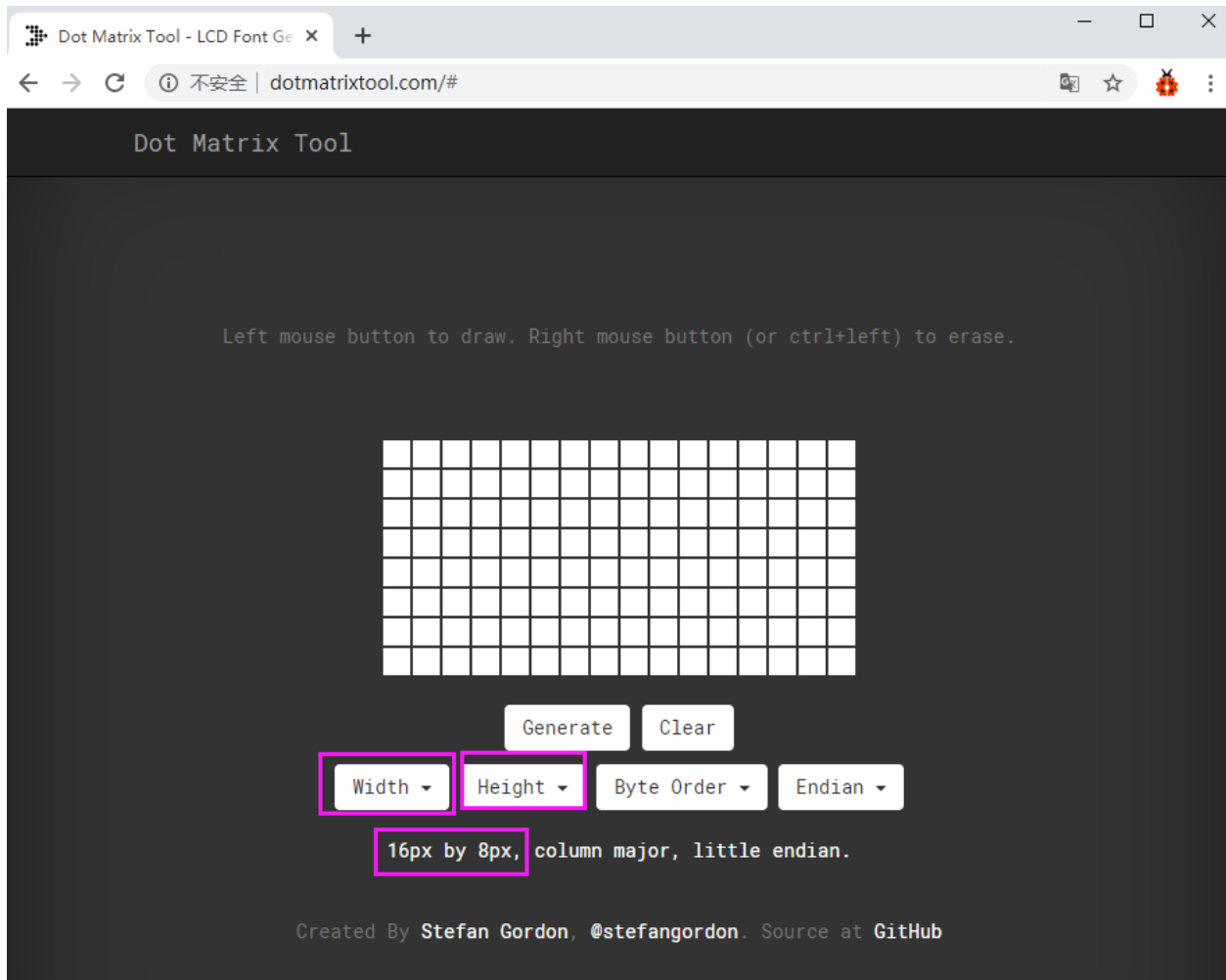
4. Instructions for the use of modulus tool

The dot matrix tool uses the online version, and the link is: <http://dotmatrixtool.com/#>

Enter the link and the page appears as shown below

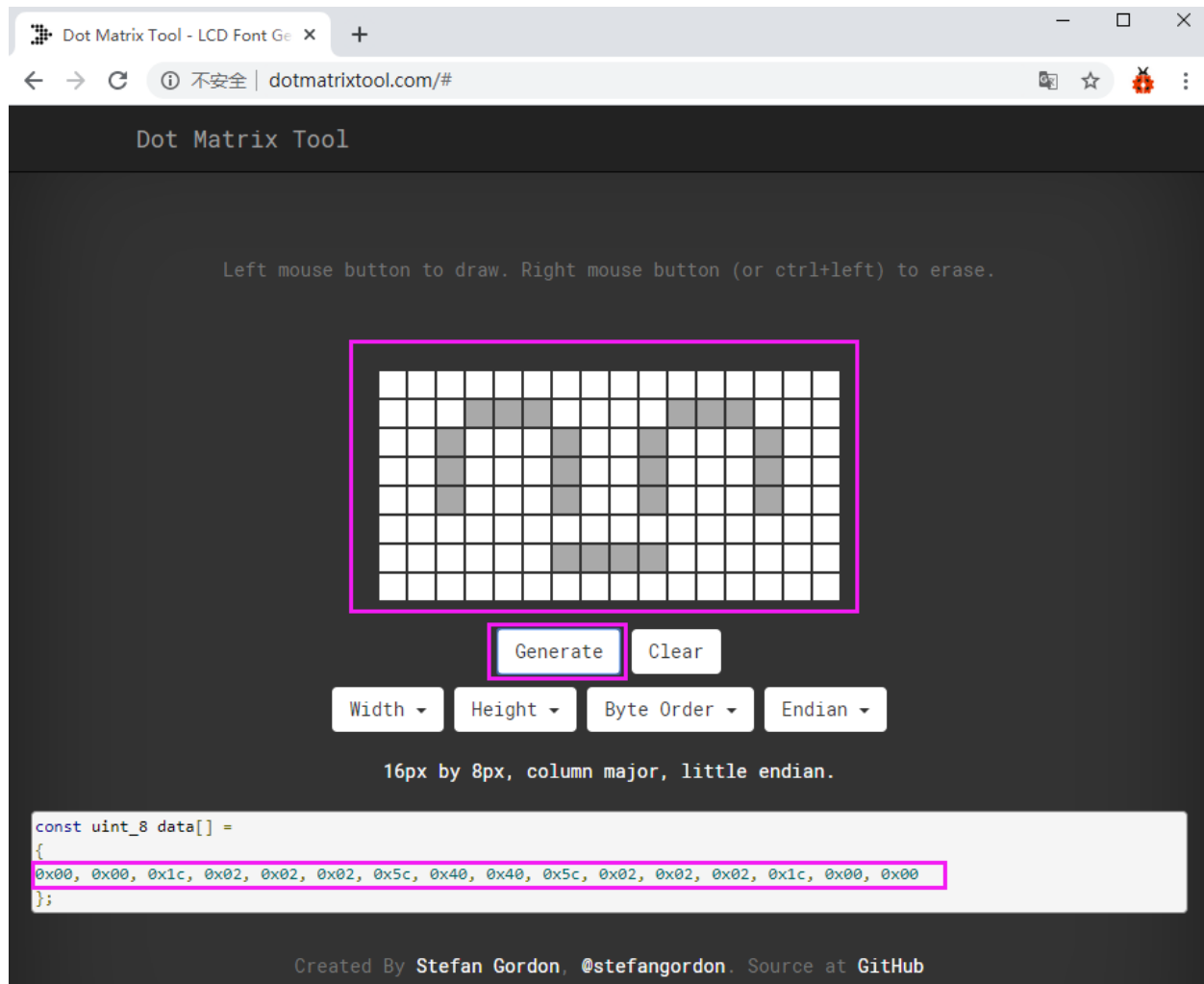


The dot matrix is 8*16, so adjust the height to 8 and width to 16, as shown in the figure below

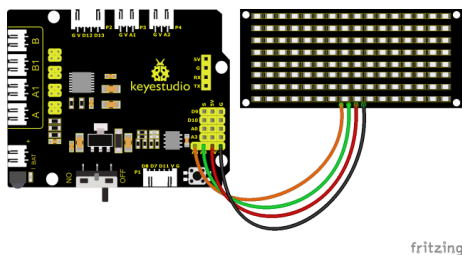


Generate hexadecimal data from the pattern

As shown in the figure below, press the left mouse button to select, right click to cancel; draw the pattern you want, click Generate, and the hexadecimal data we need will be generated.



(5)Connection Diagram:



The GND, VCC, SDA, and SCL of the 8x16 LED light board are respectively connected to the keystudio sensor expansion board-(GND), + (VCC), A4, A5 for two-wire serial communication.

(Note: though it is connected with the IIC pin of Arduino, this module is not for IIC communication. And the IO port here is to simulate I2C communication and can be connected with any two pins)

(6)Test Code:

The code to show the smile face

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 9.1
  Matrix face
  http://www.keyestudio.com
*/
//get the data of smile image from a modulus tool
unsigned char smile[] = {0x00, 0x00, 0x1c, 0x02, 0x02, 0x02, 0x5c, 0x40, 0x40, 0x5c,
↪0x02, 0x02, 0x02, 0x1c, 0x00, 0x00};

#define SCL_Pin  A5 //set a pin of clock to A5
#define SDA_Pin  A4 //set a data pin to A4

void setup() {
  //set the pin to OUTPUT
  pinMode(SCL_Pin, OUTPUT);
  pinMode(SDA_Pin, OUTPUT);
  //clear screen
  //matrix_display(clear);
}
void loop() {
  matrix_display(smile); //display the smile image
}
//this function is used for the display of dot matrix
void matrix_display(unsigned char matrix_value[])
{
  IIC_start(); //use the function to start transmitting data
  IIC_send(0xc0); //select an address

  for (int i = 0; i < 16; i++) //image data have 16 characters
  {
    IIC_send(matrix_value[i]); //data to transmit pictures
  }

  IIC_end(); //end the data transmission of pictures

  IIC_start();
  IIC_send(0x8A); //show control and select pulse width 4/16
  IIC_end();
}

//the condition that data starts transmitting
void IIC_start()
{
  digitalWrite(SDA_Pin, HIGH);
  digitalWrite(SCL_Pin, HIGH);
}

```

(continues on next page)

(continued from previous page)

```

delayMicroseconds(3);
digitalWrite(SDA_Pin, LOW);
delayMicroseconds(3);
digitalWrite(SCL_Pin, LOW);
}

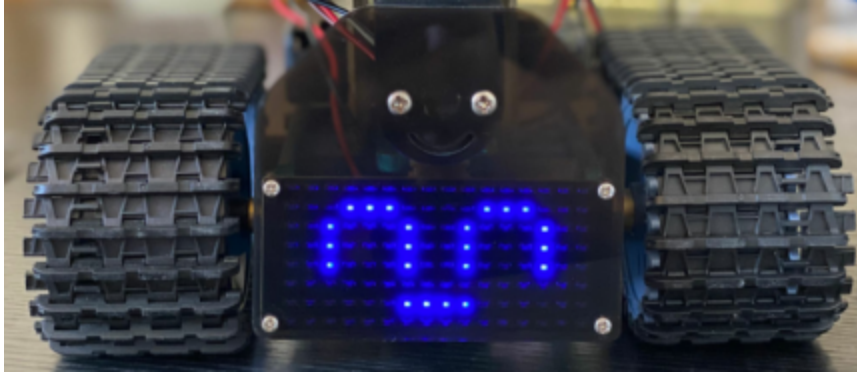
//the sign that transmission of data ends
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, HIGH);
    delayMicroseconds(3);
}

//transmit data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <=< 1) //each character has 8 digits, which is
    ↪ detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, HIGH); //pull up the clock pin SCL_Pin to end the transmission
    ↪ of data
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of
    ↪ SDA
    }
}

```

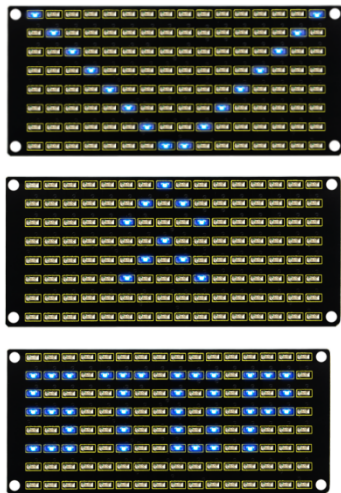
(7)Test Results:

After uploading the test code successfully, connecting according to the wiring diagram, dialing the DIP switch to the right end and powering it on, a smile-shaped pattern shows on the dot matrix.



(8)Expansion Project:

We use the modulus tool we just learned, <http://dotmatrixtool.com/#>, to make the dot matrix display the pattern start , going forward, and stop and then clear the pattern. The time interval is 2000 ms.



Code obtained from the module tool

Code for the pattern start:

```
0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01
```

Code for the pattern going forward:

```
0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00
```

Code for the pattern stepping back:

```
0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00
```

Code for the pattern turning left

```
0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00
```

Code for the pattern turning right

```
0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00
```

Code for the pattern stop

```
0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00
```

Code to clear screen

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00

Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 9.2
  Matrix face
  http://www.keyestudio.com
*/

//Array, used to save data of images, can be calculated by yourself or gotten from
↳modulus tool
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40,
↳0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00,
↳0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28,
↳0x10, 0x44, 0x28, 0x10, 0x00};
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22,
↳0x3E, 0x00, 0x3E, 0x0A, 0x0E, 0x00};
unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

#define SCL_Pin  A5 //set a pin of clock to A5
#define SDA_Pin  A4 //set a data pin to A4

void setup() {
  //set the pin to OUTPUT
  pinMode(SCL_Pin, OUTPUT);
  pinMode(SDA_Pin, OUTPUT);
  //clear screen
  matrix_display(clear);
}

void loop() {
  matrix_display(start01); //show "Start" image
  delay(2000);
  matrix_display(front); //show "front" image
  delay(2000);
  matrix_display(STOP01); //show "STOP01" image
  delay(2000);
  matrix_display(clear); //show "clear" image
  delay(2000);
}
//this function is used for the display of dot matrix

```

(continues on next page)

(continued from previous page)

```

void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //use the function to start transmitting data
    IIC_send(0xc0); //select an address

    for (int i = 0; i < 16; i++) //image data have 16 characters
    {
        IIC_send(matrix_value[i]); //data to transmit pictures
    }

    IIC_end(); //end the data transmission of pictures

    IIC_start();
    IIC_send(0x8A); //show control and select pulse width 4/16
    IIC_end();
}

//the condition that data starts transmitting
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

//the sign that transmission of data ends
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, HIGH);
    delayMicroseconds(3);
}

//transmit data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <=> 1) //each character has 8 digits, which is
    ↪ detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
    }
}

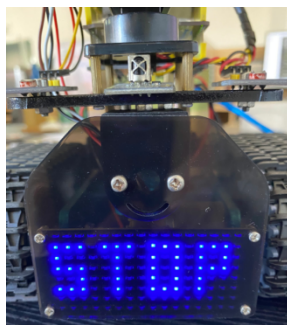
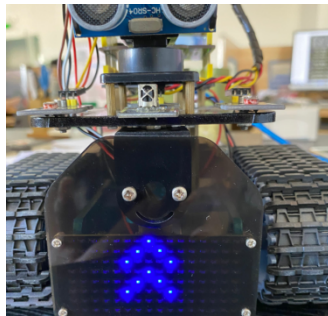
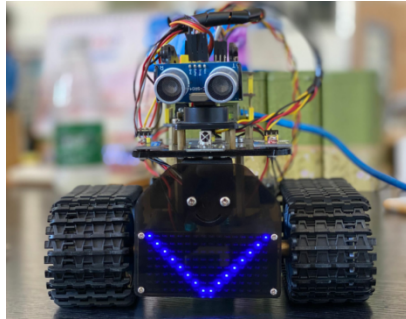
```

(continues on next page)

(continued from previous page)

```
digitalWrite(SCL_Pin, HIGH); //pull up the clock pin SCL_Pin to end the transmission.  
↪ of data  
  delayMicroseconds(3);  
  digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of.  
↪ SDA  
}  
}
```

After uploading test code, the facial expression board shows these patterns orderly and repeats this sequence.



6.3.10 Project 10: Light Following Tank

(1)Description:

In previous projects, we introduced in detail the use of various sensors, modules, and expansion boards on the smart car. Now let's move to the projects of the smart car . The light-following smart cars, as the name suggests, is a smart car that can follow the light.

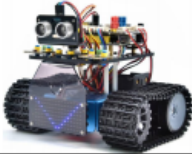



We can combine the knowledge from projects photoresistor and motor drive to make a light-seeking smart car. In the project, we use two photoresistor modules to detect the light intensity on the left and right sides of the smart car, read

the corresponding analog values, and then control the rotation of the two motors based on these two data so as to control the movements of the smart car.

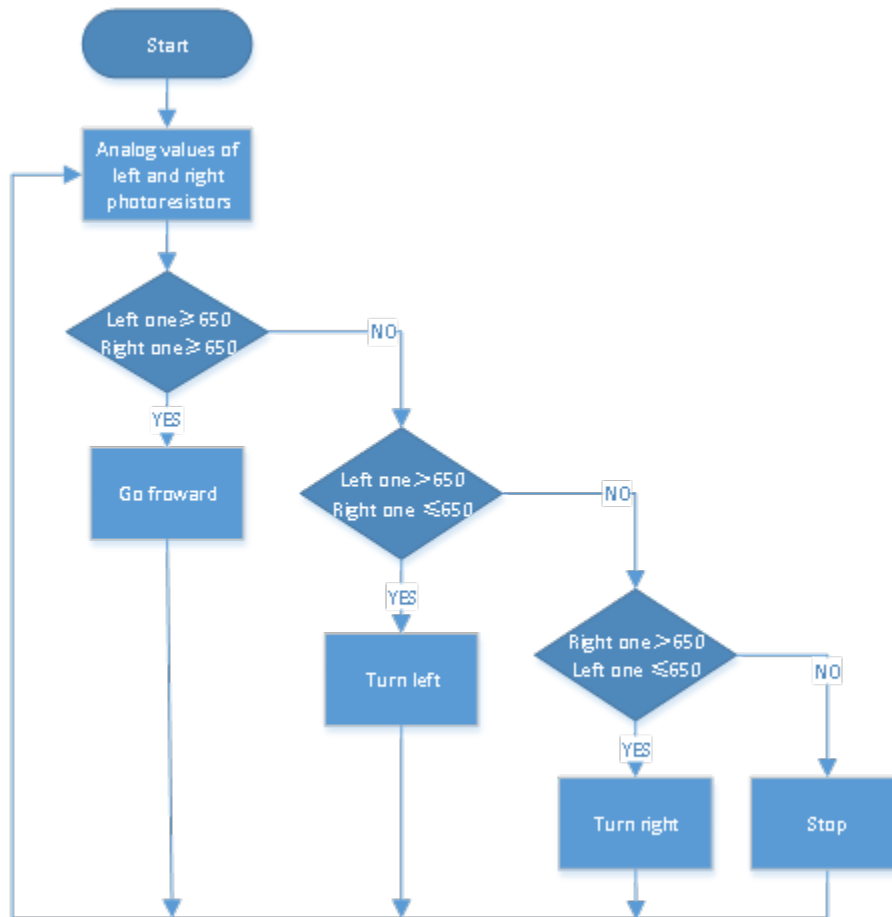
The specific logic of the light-following smart car is shown as below.

Detection	the bigger the brightness, the bigger the value
photoresistor module on the left	left_light
photoresistor module on the right	right_light
Condition	Movement
left_light650 and right_light650	Move forwardset PWM to 200
left_light650 and right_light650	Rotate leftset PWM to 200
left_light650 and right_light650	Rotate rightset PWM to 200
left_light650 and right_light650	stop

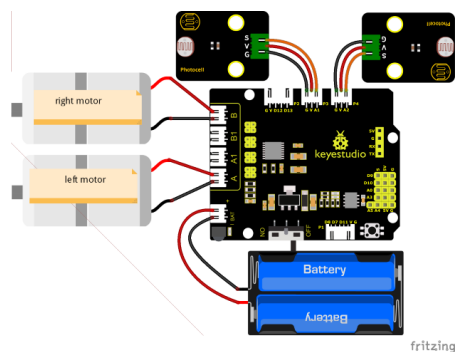
(2)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:



(4)Connection Diagram:



Note: The pin “G”, “V” and S of the left photoresistor module are connected to G (GND), V (VCC), A1 respectively; The pin “G”, “V” and S of the right photoresistor module are connected to the G (GND), V (VCC), and A2 respectively. The 4pin cable is marked with A, A1, B1 and B. The right rear motor is connected to B port of the 8833 motor driver expansion board and the left front motor is connected to A port of the 8833 motor driver expansion board

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 10
  light follow tank
  http://www.keyestudio.com
*/
#define light_L_Pin A1 //Define the pin of the photosensitive sensor on the left
#define light_R_Pin A2 //Define the pin of the photosensitive sensor on the right
#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
int left_light;
int right_light;
void setup() {
  Serial.begin(9600);
  pinMode(light_L_Pin, INPUT);
  pinMode(light_R_Pin, INPUT);
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
}
void loop() {
  left_light = analogRead(light_L_Pin);
  right_light = analogRead(light_R_Pin);
  Serial.print("left_light_value = ");
  Serial.println(left_light);
  Serial.print("right_light_value = ");
  Serial.println(right_light);
  if (left_light > 650 && right_light > 650) //go front
  {
    Car_front();
  }
  else if (left_light > 650 && right_light <= 650) //turn left
  {
    Car_left();
  }
  else if (left_light <= 650 && right_light > 650) //turn right
  {
    Car_right();
  }
  else //otherwise, stop
  {
    Car_Stop();
  }
}

```

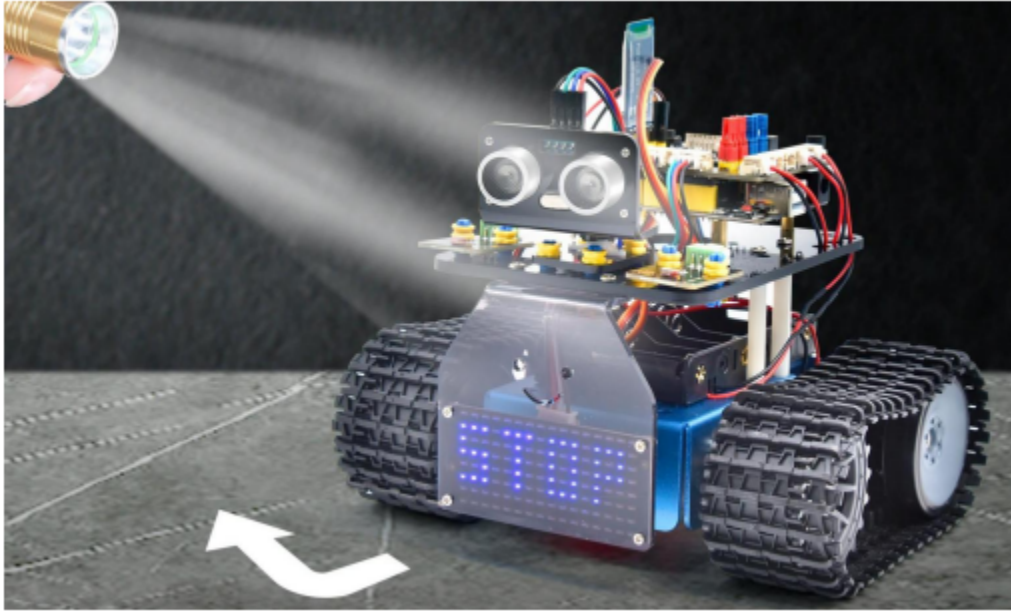
(continues on next page)

(continued from previous page)

```
void Car_front()
{
  digitalWrite(MR_Ctrl, HIGH);
  analogWrite(MR_PWM, 55);
  digitalWrite(ML_Ctrl, HIGH);
  analogWrite(ML_PWM, 55);
}
void Car_left()
{
  digitalWrite(MR_Ctrl, HIGH);
  analogWrite(MR_PWM, 55);
  digitalWrite(ML_Ctrl, LOW);
  analogWrite(ML_PWM, 200);
}
void Car_right()
{
  digitalWrite(MR_Ctrl, LOW);
  analogWrite(MR_PWM, 200);
  digitalWrite(ML_Ctrl, HIGH);
  analogWrite(ML_PWM, 55);
}
void Car_Stop()
{
  digitalWrite(MR_Ctrl, LOW);
  analogWrite(MR_PWM, 0);
  digitalWrite(ML_Ctrl, LOW);
  analogWrite(ML_PWM, 0);
}
```

(6)Test Result

After uploading the test code successfully, connecting according to the wiring diagram, dialing the DIP switch to the right end and powering it on, the smart car follows the light to move.



6.3.11 Project 11: Ultrasonic Following Tank

(1)Description:

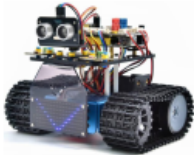

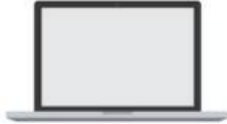

In the previous lesson, we learned about the light-following smart car. And in this lesson, we can combine the knowledge to make an ultrasonic sound-following car.

In the project, we use ultrasonic sensors to detect the distance between the car and the obstacle in front, and then control the rotation of the two motors based on this data so as to control the movements of the smart car.

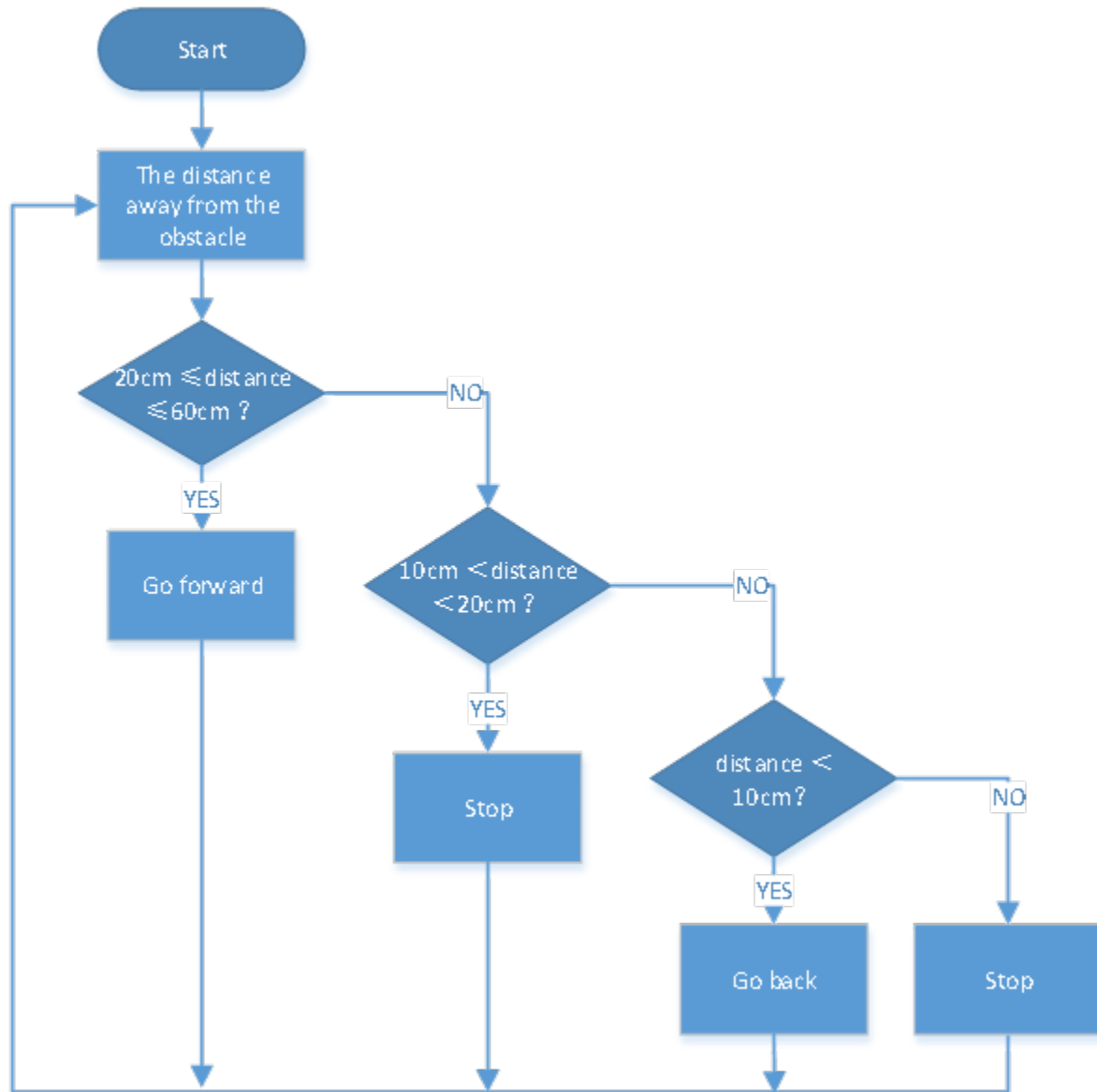
The specific logic of the ultrasonic sound- following smart car is shown in the table blow:

Detection	Setting
The distance(cm) between the car and the obstacle front	Set the angle of the servo to 90°
Condition	Movement
distance20 and distance50	Go front
10distance20distance50	Stop
distance10	go back

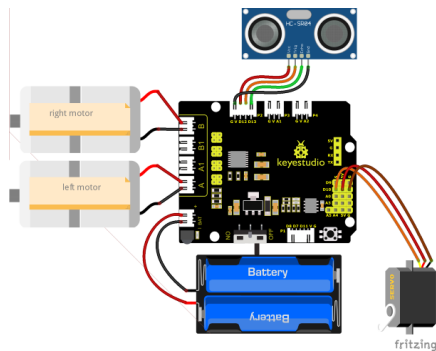
(2) Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:



(4) Connection Diagram:



(5) Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 11
  Ultrasonic follow tank
  http://www.keyestudio.com
*/
#define servoPin 10 //The pin of servo

#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
#define Trig 12
#define Echo 13
float distance;

void setup() {
  pinMode(servoPin, OUTPUT);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
  procedure(90); //Set the angle of the servo to 90°
  delay(500); //delay in 500ms
}

void loop() {
  distance = checkdistance(); //Assign the distance measured by ultrasonic sound to
  distance
  if (distance >= 20 && distance <= 50) //go front
  {
    Car_front();
  }
}

```

(continues on next page)

(continued from previous page)

```

}
else if (distance > 10 && distance < 20) //stop
{
    Car_Stop();
}
else if (distance <= 10) //go back
{
    Car_back();
}
else //In other conditions, it stops
{
    Car_Stop();
}
}
void Car_front()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
}
void Car_back()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
}
void Car_left()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
}
void Car_right()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
}
void Car_Stop()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
}

//The function to control servos
void procedure(byte myangle) {

```

(continues on next page)

(continued from previous page)

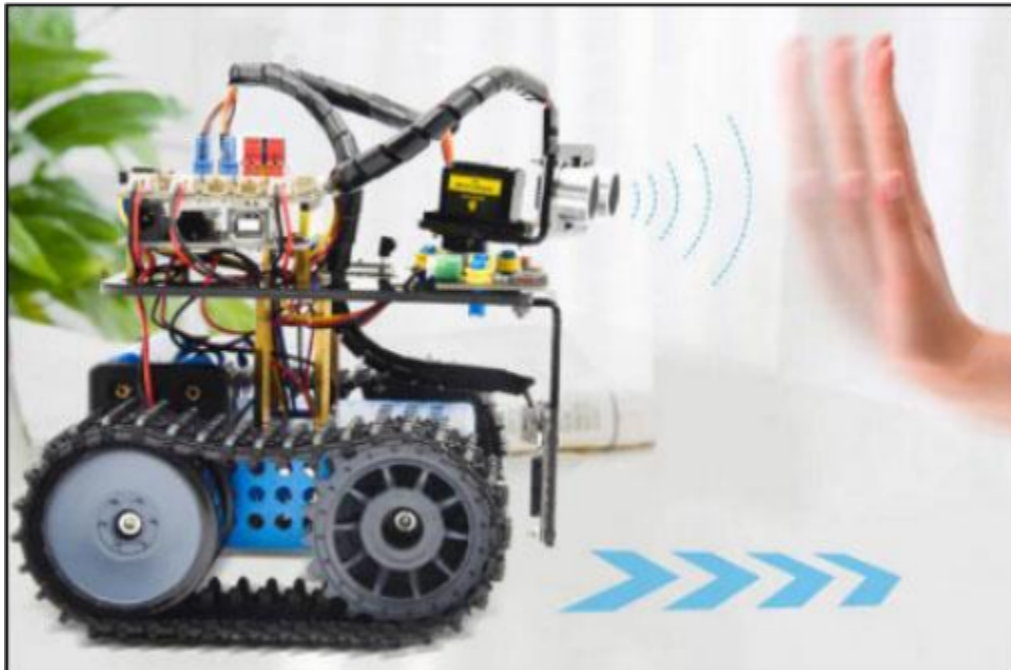
```

int pulsewidth;
for (int i = 0; i < 5; i++) {
    pulsewidth = myangle * 11 + 500; //Calculate the value of pulse width
    digitalWrite(servoPin, HIGH);
    delayMicroseconds(pulsewidth); //The time in high level represents the pulse width
    digitalWrite(servoPin, LOW);
    delay((20 - pulsewidth / 1000)); //As the cycle is 20ms, the time left is in low
    level
}
}
//The function to control ultrasonic sound
float checkdistance() {
    static float distance;
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);
    distance = pulseIn(Echo, HIGH) / 58.20; //The 58.20 here comes from 2*29.1=58.2
    delay(10);
    return distance;
}

```

(6)Test Results:

Upload the test code successfully, wire up, dial the DIP switch to the right end, power up and set the servo to 90°the smart car follows the obstacle to move.



6.3.12 Project 12: Ultrasonic Obstacle Avoidance Tank

(1)Description:

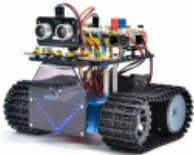



In the previous project, we made an ultrasonic sound-following smart car. In fact, using the same components and the same wiring method, we only need to change the test code to turn it into an ultrasonic obstacle avoidance smart car. This smart car can move with the movement of the human hands.

We use ultrasonic sensors to detect the distance between the smart car and the obstacle in front, and then control the rotation of the two motors based on this data so as to control the movements of the smart car

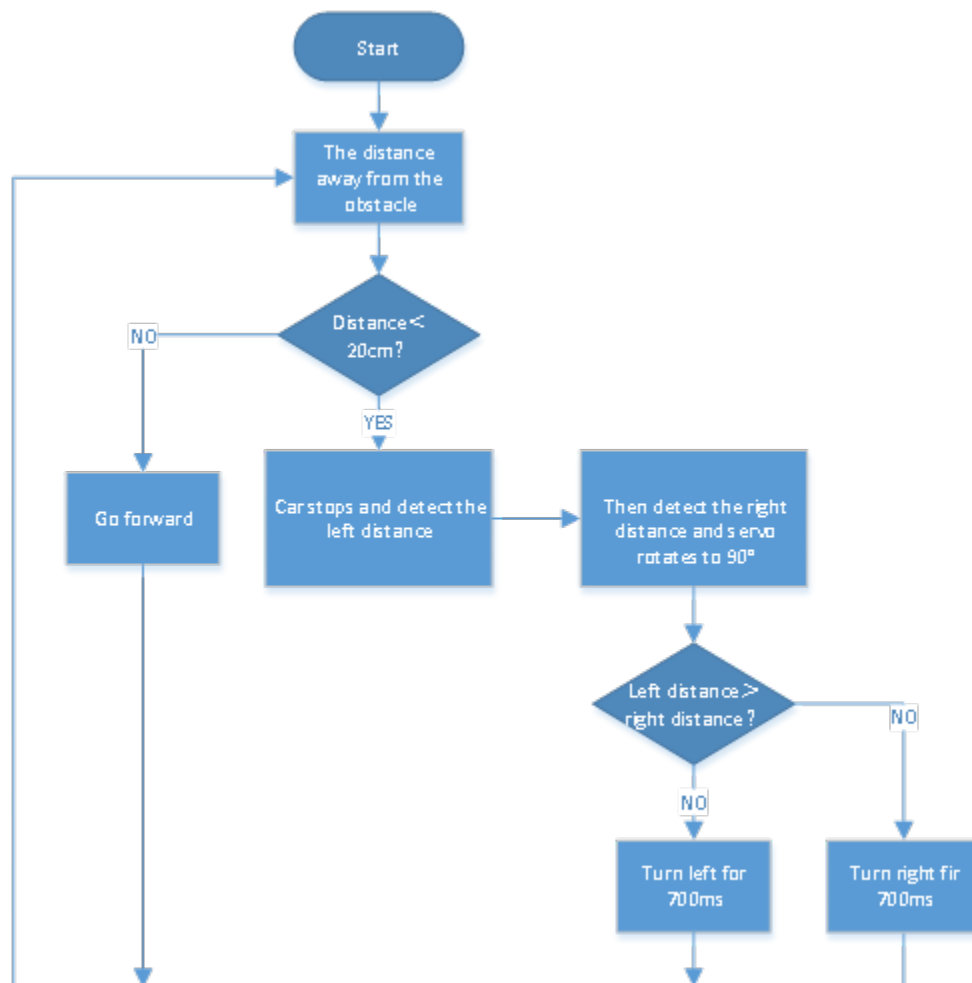
Detection	
Distance measured by the ultrasonic sensor between the car and the obstacle in front set the angle of the servo to 90°	a(cm)
Distance measured by the ultrasonic sensor between the car and the obstacle on the right set the angle of the servo to 20°	a2(cm)
Distance measured by the ultrasonic sensor between the car and the obstacle on the left set the angle of the servo to 160°	a1(cm)
Setting: set the starting angle of the servo to 90°	

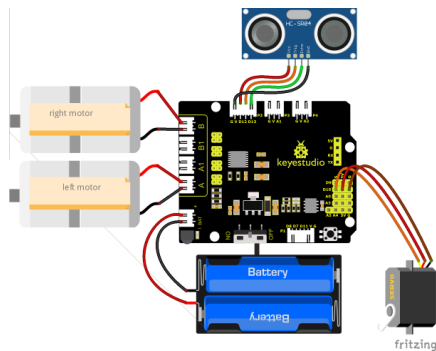
Con- di- tion 1	Con- di- tion 2	Con- di- tion 3	Movement
a20			Stop for 500ms set the angle of the servo to 180°, read a1, delay in 100ms set the angle of the servo to 0°, read a2, delay in 0.1s.
	a150 or a250		Compare a1 with a2
		a1a2	Set the angle of the servo to 90°, rotate left for 700ms (set PWM to 255) move forward set PWM to 200.
		a1a2	Set the angle of the servo to 90°, rotate right for 700ms (set PWM to 255) move forward set PWM to 200.
Con- di- tion 1	Con- di- tion 2		Movement
a20	a150 and a250	dom	set the angle of the servo to 90°, rotate left for 500ms (set PWM to 255) move forward (set PWM to 200) set the angle of the servo to 90°, rotate right for 500ms (set PWM to 255) move forward (set PWM to 200)
Con- di- tion			Movement
a20			move forward (set PWM to 100)

(2) Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3) Flow chart:



(4)Connection Diagram:

(Note: the brown, red and orange wires of the servo are respectively connected to G (GND), 5V and D10 of the expansion board and for the ultrasonic sensor, the VCC pin is connected to the 5v (V), the Trig pin to digital 12 (S), the Echo pin to digital 13 (S), and the Gnd pin to Gnd (G); the same as last project.

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 12
  Ultrasonic avoid tank
  http://www.keyestudio.com
*/
#define servoPin 10 //The pin of servo
int a, a1, a2;
#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
#define Trig 12
#define Echo 13
float distance;

void setup() {
  Serial.begin(9600);
  pinMode(servoPin, OUTPUT);
  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
  procedure(90); //Set the angle of the servo to 90°
  delay(500); //delay in 500ms
}
void loop() {

```

(continues on next page)

(continued from previous page)

```

    a = checkdistance(); //Assign the distance to the front detected by ultrasonic sensor_
    ↪to the variable a

    if (a < 20) { //When the distance to the front is less than 20cm
        Car_Stop(); //The robot stops
        delay(500); //delay in 500ms
        procedure(180); //Ultrasonic pan-tilt turns left
        delay(500); //delay in 500ms
        a1 = checkdistance(); //Assign the distance to the left detected by ultrasonic_
    ↪sensor to the variable a1
        delay(100); //read value
        procedure(0); //Ultrasonic pan-tilt turns right
        delay(500); //delay in 500ms
        a2 = checkdistance(); //Assign the distance to the right detected by ultrasonic_
    ↪sensor to the variable a2
        delay(100); //read value

        procedure(90); //Back to 90°
        delay(500);
        if (a1 > a2) { //When the distance to the left is bigger than to the right
            Car_left(); //The robot turns left
            delay(700); //turn left 700ms
        } else {
            Car_right(); //It turns left for 700ms
            delay(700);
        }
    }
    else { //When the distance to the front is >=20cm the robot moves forward
        Car_front(); //go front
    }
}

void Car_front()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
}

void Car_back()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
}

void Car_left()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, LOW);

```

(continues on next page)

(continued from previous page)

```

    analogWrite(ML_PWM, 200);
}
void Car_right()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
}
void Car_Stop()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
}

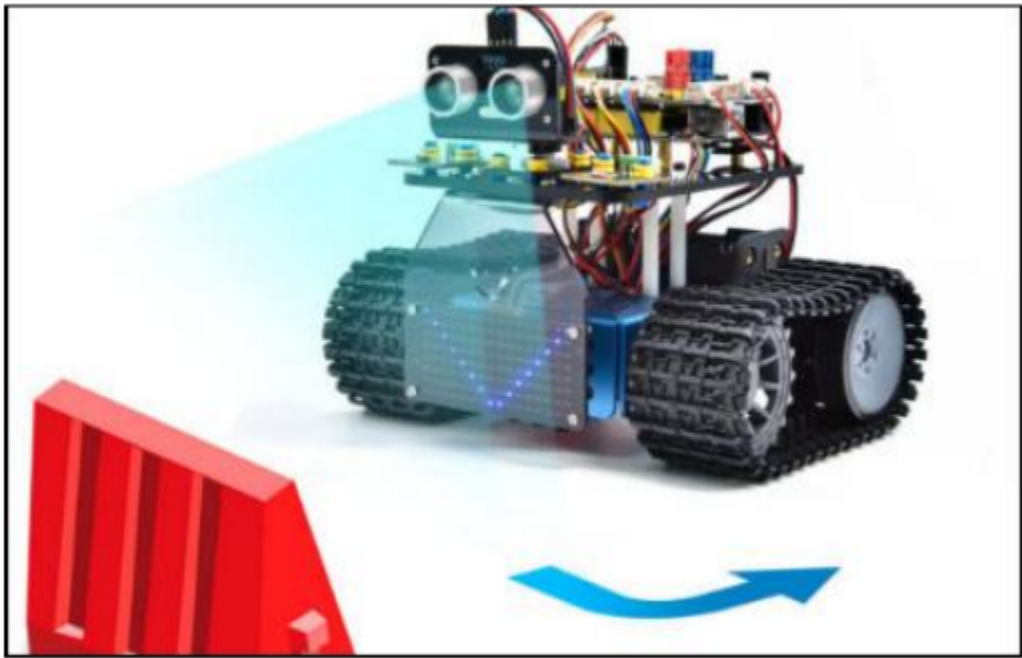
//The function controls servos
void procedure(byte myangle) {
    int pulsewidth;
    for (int i = 0; i < 5; i++) {
        pulsewidth = myangle * 11 + 500; //Calculate the value of pulse width
        digitalWrite(servoPin, HIGH);
        delayMicroseconds(pulsewidth); //The time in high level represents the pulse width
        digitalWrite(servoPin, LOW);
        delay((20 - pulsewidth / 1000)); //As the cycle is 20ms, the time left is in low
        ↪ level
    }
}

//The function controls ultrasonic sound
float checkdistance() {
    float distance;
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);
    distance = pulseIn(Echo, HIGH) / 58.20; //The 58.20 here comes from 2*29.1=58.2
    delay(10);
    return distance;
}

```

(6)Test Result:

After upload the test code successfully, wire up, turn the DIP switch to the ON end, and power up, the smart car moves forward and automatically avoids obstacles.



6.3.13 Project 13: Move-in-Confind-Space Tank

(1)Description:

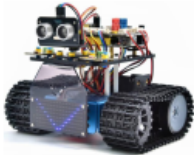



The ultrasonic sound-following and obstacle avoidance functions of the smart car have been introduced in previous projects. Here we intend to combine the knowledge in the previous courses to confine the smart car to move in a certain space.

In the experiment, we use the line-tracking sensor to detect whether there is a black line around the smart car, and then control the rotation of the two motors according to the detection results, so as to lock the smart car in a circle drawn in black line.

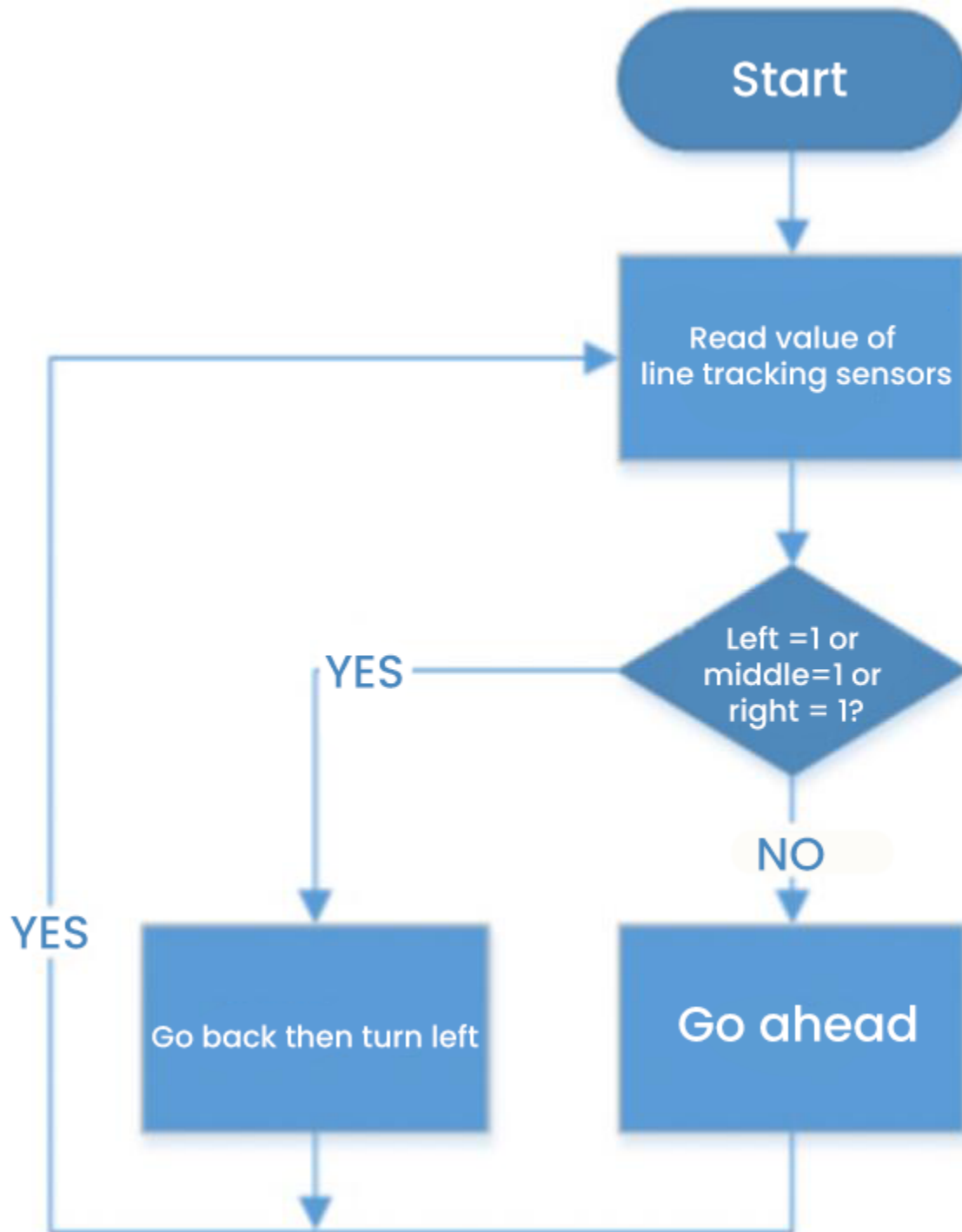
The specific logic of the line-tracking smart car is shown in the table blow:

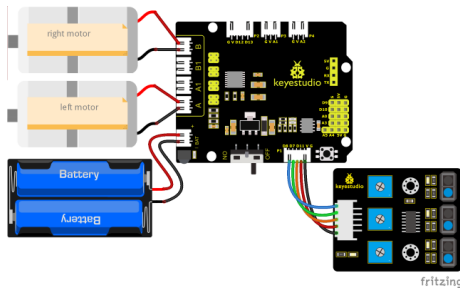
Detection	
Line-tracking sensor in the middle	Black line detected: in high levelWhite line detected: in low level
Line-tracking sensor on the left	Black line detected: in high levelWhite line detected: in low level
Line-tracking sensor on the right	Black line detected: in high levelWhite line detected: in low level
	Condition
	All the three line-tracking sensors detect no black lines
	Any of the three line-tracking sensors detects black lines

(2)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:



(4)Connection Diagram:**(5)Test Code:**

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 13
  draw a circle for tank
  http://www.keyestudio.com
*/

//The wiring of line tracking sensor
#define L_pin  11 //left
#define M_pin  7  //middle
#define R_pin  8  //right

#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6  //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5  //Define the PWM control pin of the right motor
int L_val, M_val, R_val;

void setup()
{
  Serial.begin(9600); //Set the baud rate to 9600
  pinMode(L_pin, INPUT); //Set all pins of the line tracking sensor as input mode
  pinMode(M_pin, INPUT);
  pinMode(R_pin, INPUT);
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
}

void loop () {
  L_val = digitalRead(L_pin); //Read the value of the left sensor
  M_val = digitalRead(M_pin); //Read the value of the middle sensor
  R_val = digitalRead(R_pin); //Read the value of the right sensor

```

(continues on next page)

(continued from previous page)

```
if ( L_val == 0 && M_val == 0 && R_val == 0 ) { //when black lines are not detected,
↪go front
    Car_front();
}
else { //black lines are detected, go back then turn left
    Car_back();
    delay(700);
    Car_left();
    delay(800);
}
}

void Car_front()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 100);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 100);
}

void Car_back()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 150);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 150);
}

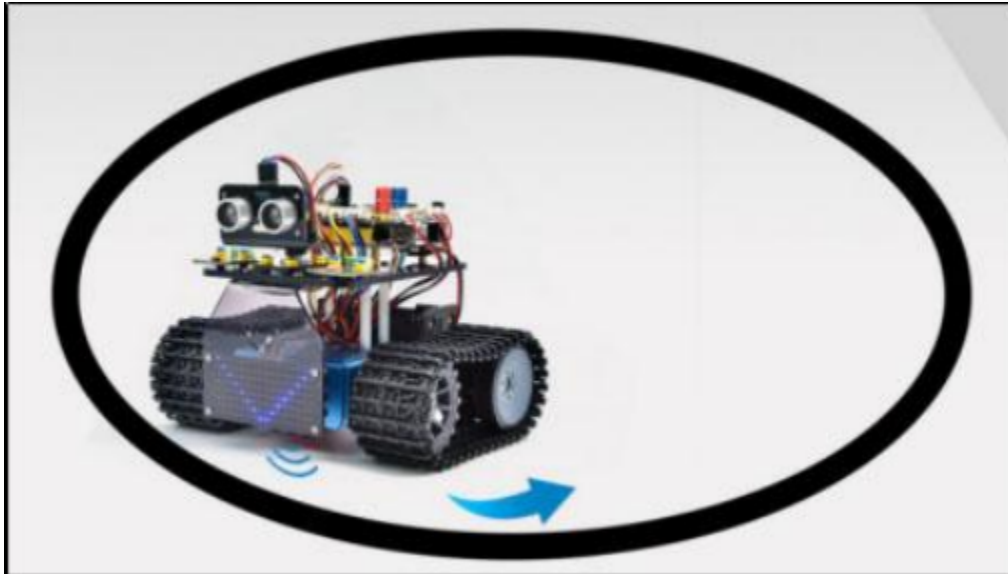
void Car_left()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 100);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 150);
}

void Car_right()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 150);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 100);
}

void Car_Stop()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
}
```

(6)Test Results:

After uploading the test code successfully and powering it up, the smart car moves in a confined space, the circle drawn in black line.

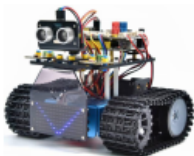

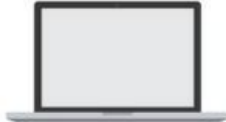

**6.3.14 Project 14: Line-tracking Tank****(1)Description:**

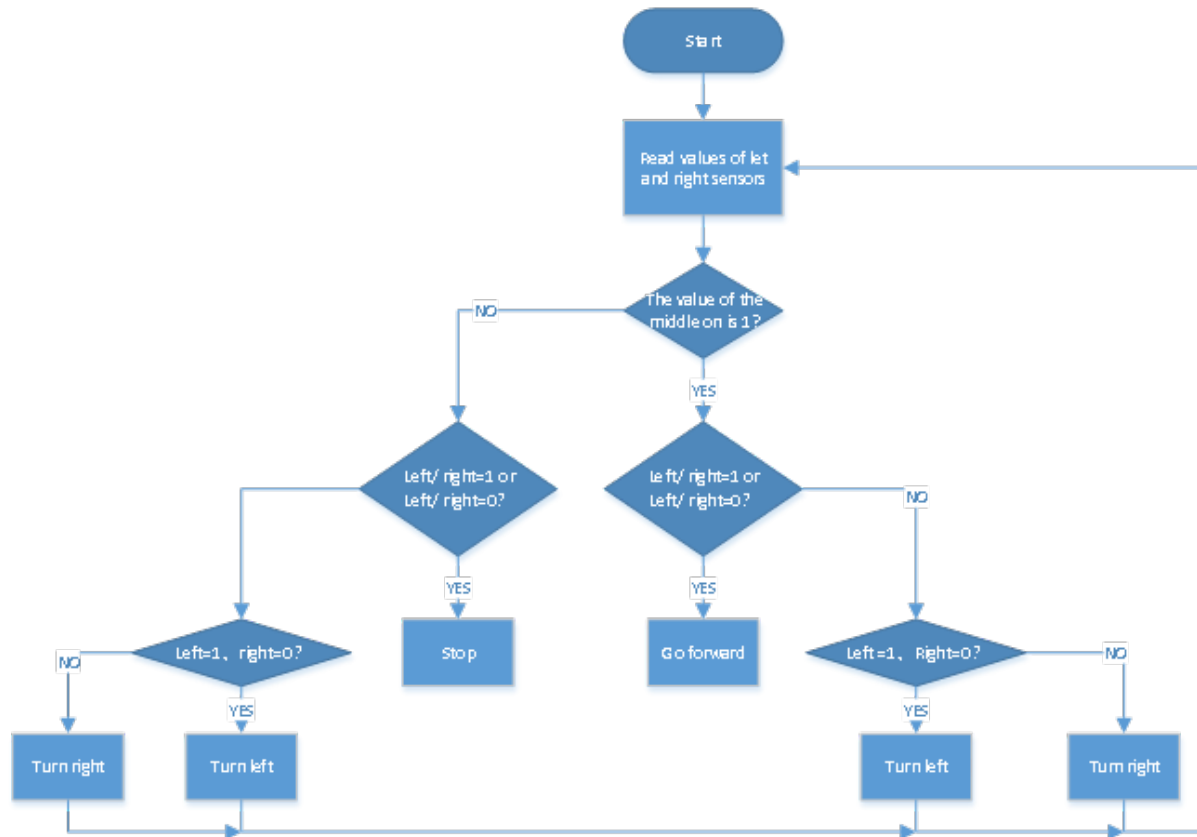
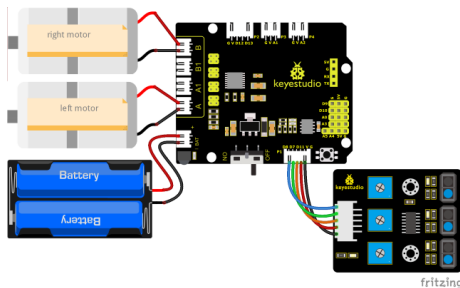
The previous project has introduced how to confine the smart car to move in a certain space. In this project, we could use the knowledge learned before to make it a line-tracking smart car. In the experiment, we use the line-tracking sensor to detect whether there is a black line around the smart car, and then control the rotation of the two motors according to the detection results, so as to make the smart car to move along the black line.

The specific logic of the line-tracking smart car is shown in the table blow:

Detection		
Line-tracking sensor in the middle	Black line detected: in high level White line detected: in low level	
Line-tracking sensor on the left	Black line detected: in high level White line detected: in low level	
Line-tracking sensor on the right	Black line detected: in high level White line detected: in low level	
Condition 1	Condition 2	Movement
Line-tracking sensor in the middle detects the black line	Line-tracking sensor on the left detects the black line and the one on the right detects white line	Rotate leftset PWM to 200
	Line-tracking sensor on the left detects white line and the one on the right detects the black line	Rotate rightset PWM to 200
	Both the left and right line-tracking sensors detect white lineor- Both the left and right detect the black line	Move forward
Condition 1	Condition 2	Movement
Line-tracking sensor in the middle detects white line	Line-tracking sensor on the left detects the black line and the one on the right detects white line	Rotate leftset PWM to 200
	Line-tracking sensor on the left detects white lineand the one on the right detects the black line	Rotate rightset PWM to 200
	Both the left and right line-tracking sensors detect white lineor- Both the left and right line-tracking sensors detect the black line	Stop

(2)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:**(4)Wiring Diagram:****(5)Test Code:**

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 14
  Line track tank

```

(continues on next page)

```

http://www.keyestudio.com
*/

//The wiring of line tracking sensor
#define L_pin 11 //left
#define M_pin 7 //middle
#define R_pin 8 //right
#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
int L_val, M_val, R_val;
void setup()
{
  Serial.begin(9600); //Set the baud rate to 9600
  pinMode(L_pin, INPUT); //Set all pins of the line tracking sensor as input mode
  pinMode(M_pin, INPUT);
  pinMode(R_pin, INPUT);
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
}

void loop () {
  L_val = digitalRead(L_pin); //Read the value of the left sensor
  M_val = digitalRead(M_pin); //Read the value of the middle sensor
  R_val = digitalRead(R_pin); //Read the value of the right sensor
  if (M_val == 1) { //the middle one detects black lines
    if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↳on the right, turn left
      Car_left();
    }
    else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right, not
↳on the left, turn right
      Car_right();
    }
    else { //otherwise, go front
      Car_front();
    }
  }
  else { //The middle one doesn't detect black lines
    if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↳on the right, turn left
      Car_left();
    }
    else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right, not
↳on the left, turn right
      Car_right();
    }
    else { //otherwise, stop
      Car_Stop();
    }
  }
}

```

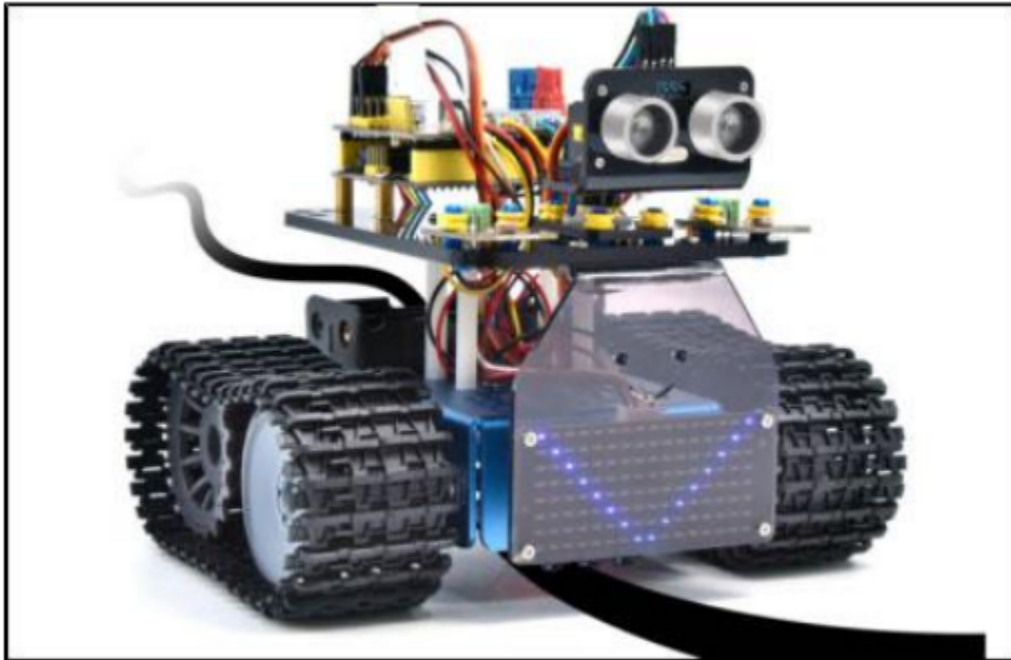
(continues on next page)

(continued from previous page)

```
    }  
  }  
}  
  
//go front  
void Car_front()  
{  
  digitalWrite(MR_Ctrl, HIGH);  
  analogWrite(MR_PWM, 100);  
  digitalWrite(ML_Ctrl, HIGH);  
  analogWrite(ML_PWM, 100);  
}  
  
//go back  
void Car_back()  
{  
  digitalWrite(MR_Ctrl, LOW);  
  analogWrite(MR_PWM, 150);  
  digitalWrite(ML_Ctrl, LOW);  
  analogWrite(ML_PWM, 150);  
}  
  
//turn left  
void Car_left()  
{  
  digitalWrite(MR_Ctrl, HIGH);  
  analogWrite(MR_PWM, 100);  
  digitalWrite(ML_Ctrl, LOW);  
  analogWrite(ML_PWM, 150);  
}  
  
//turn right  
void Car_right()  
{  
  digitalWrite(MR_Ctrl, LOW);  
  analogWrite(MR_PWM, 150);  
  digitalWrite(ML_Ctrl, HIGH);  
  analogWrite(ML_PWM, 100);  
}  
  
//stop  
void Car_Stop()  
{  
  digitalWrite(MR_Ctrl, LOW);  
  analogWrite(MR_PWM, 0);  
  digitalWrite(ML_Ctrl, LOW);  
  analogWrite(ML_PWM, 0);  
}
```

(6)Test Result:

After uploading the test code successfully and powering it up, the smart car moves along the black line.




6.3.15 Project 15: IR Remote Control Tank






(1)Description:

Infrared remote control is one of the most common remote control found applications in electric motors, electric fans, and many other household appliances. In this project, we use the knowledge we learned before to make an infrared remote control smart car.

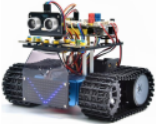

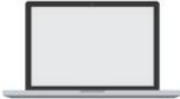


In the 9th lesson, we have tested the corresponding key value of each key of the infrared remote control. In the project, we can set the code (key value) to make the corresponding button to control the movements of the smart car, and display the movement patterns on the 8X16 LED dot matrix.

The specific logic of the line-tracking smart car is shown in the table:

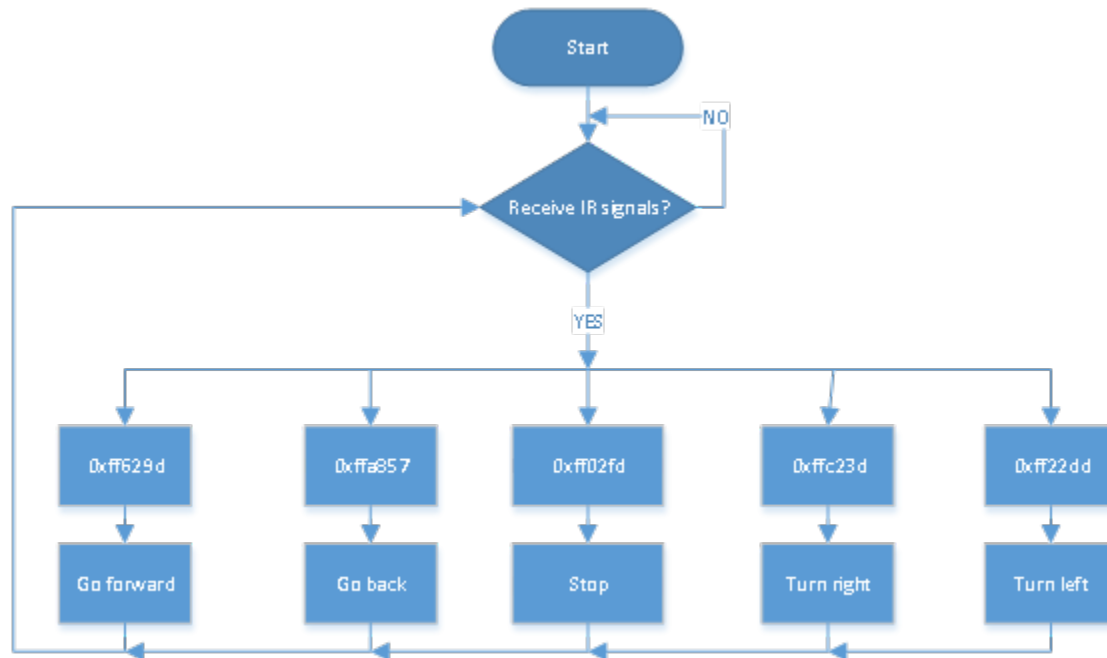
Initial setting: 8X16 LED dot matrix shows the pattern“”.

Ultrasonic key	Key value	Instructions from keys
	FF629D	Move forward set PWM to 200 display the pattern of going forward
	FFA857	Go back set PWM to 200 display the pattern of going back
	FF22DD	Turn left display the pattern "STOP"
	FFC23D	Turn right display the pattern of turning left
	FF02FD	Stop display the pattern "STOP"

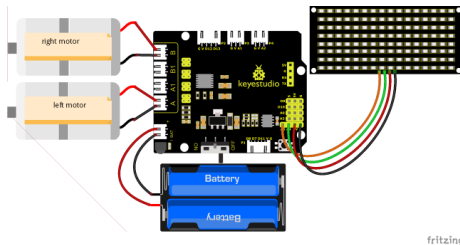
(2) Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2	Remote Control*1
				

(3)Flow chart:



(4)Connection Diagram:



Note:

GND, VCC, SDA and SCL of the 8x16 LED panel are connected to (GGND), (VVCC). SDA and SCL of the expansion board.

Since the 8833 board integrates the IR receiver, you don't need to wire it up. The pins of the IR receiver are (GGND), (VVCC) and D3.

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 15
IRremote Control Tank
  
```

(continues on next page)

(continued from previous page)

```

    http://www.keyestudio.com
*/
#include <IRremote.h>
IRrecv irrecv(3); //
decode_results results;
long ir_rec; //Used to store the received infrared values

//Array, used to save data of images, can be calculated by yourself or gotten from
↳ modulus tool
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40,
↳ 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24,
↳ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00,
↳ 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28,
↳ 0x10, 0x44, 0x28, 0x10, 0x00};
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44,
↳ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22,
↳ 0x3E, 0x00, 0x3E, 0x0A, 0x0E, 0x00};
unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
↳ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
#define SCL_Pin  A5 //Set the clock pin as A5
#define SDA_Pin  A4 //Set the data pin as A4

#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor

void setup() {
    Serial.begin(9600);
    irrecv.enableIRIn(); //Initialize infrared receiver library

    pinMode(ML_Ctrl, OUTPUT);
    pinMode(ML_PWM, OUTPUT);
    pinMode(MR_Ctrl, OUTPUT);
    pinMode(MR_PWM, OUTPUT);

    pinMode(SCL_Pin, OUTPUT);
    pinMode(SDA_Pin, OUTPUT);
    matrix_display(clear); //clear screens
    matrix_display(start01); //show the image to start
}

void loop() {
    if (irrecv.decode(&results)) { //Receive infrared remote control value
        ir_rec = results.value;
        String type = "UNKNOWN";
        String typelist[14] = {"UNKNOWN", "NEC", "SONY", "RC5", "RC6", "DISH", "SHARP",
↳ "PANASONIC", "JVC", "SANYO", "MITSUBISHI", "SAMSUNG", "LG", "WHYNTER"};

```

(continues on next page)

(continued from previous page)

```

    if (results.decode_type >= 1 && results.decode_type <= 13) {
        type = typelist[results.decode_type];
    }
    Serial.print("IR TYPE:" + type + " ");
    Serial.println(ir_rec, HEX);
    irrecv.resume();
}

switch (ir_rec) {
    case 0xFF629D: Car_front();    break;    //the command to go front
    case 0xFFA857: Car_back();     break;    //the command to go back
    case 0xFF22DD: Car_T_left();   break;    //the command to turn left
    case 0xFFC23D: Car_T_right();  break;    //the command to turn right
    case 0xFF02FD: Car_Stop();     break;    //the command to stop
    case 0xFF30CF: Car_left();     break;    //the command to rotate to left
    case 0xFF7A85: Car_right();    break;    //the command to rotate to right
    default: break;
}
}

/*****The function to run the motor*****/
void Car_back() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
    matrix_display(back); //Go back
}

void Car_front() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
    matrix_display(front); //show the image to go front
}

void Car_left() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
    matrix_display(left); //show the image to turn lefr
}

void Car_right() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
    matrix_display(right); //show the image to turn right
}

```

(continues on next page)

(continued from previous page)

```

void Car_Stop() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
    matrix_display(STOP01); //show the image to stop
}

void Car_T_left() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 100);
    matrix_display(left); //show the image to turn left
}

void Car_T_right() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 100);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 0);
    matrix_display(right); //show the image to turn right
}

//This function is used for dot matrix screen display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //Function to call data transfer start condition
    IIC_send(0xc0); //Choose an address
    for (int i = 0; i < 16; i++) //Pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //transfer pattern data
    }
    IIC_end(); //End pattern data transfer
    IIC_start();
    IIC_send(0x8A); //display control, select pulse width as 4/16
    IIC_end();
}

//Conditions for the start of data transfer
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

//the sign of ending data transmission

```

(continues on next page)

(continued from previous page)

```

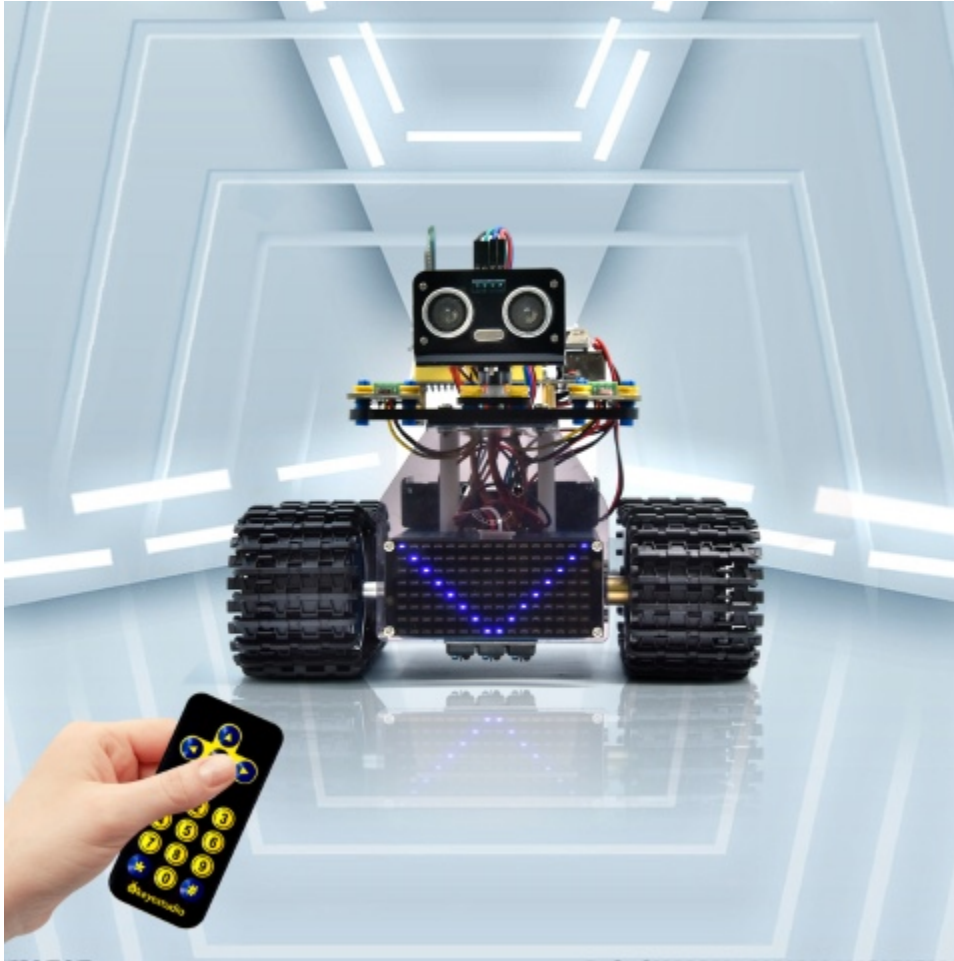
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, HIGH);
    delayMicroseconds(3);
}

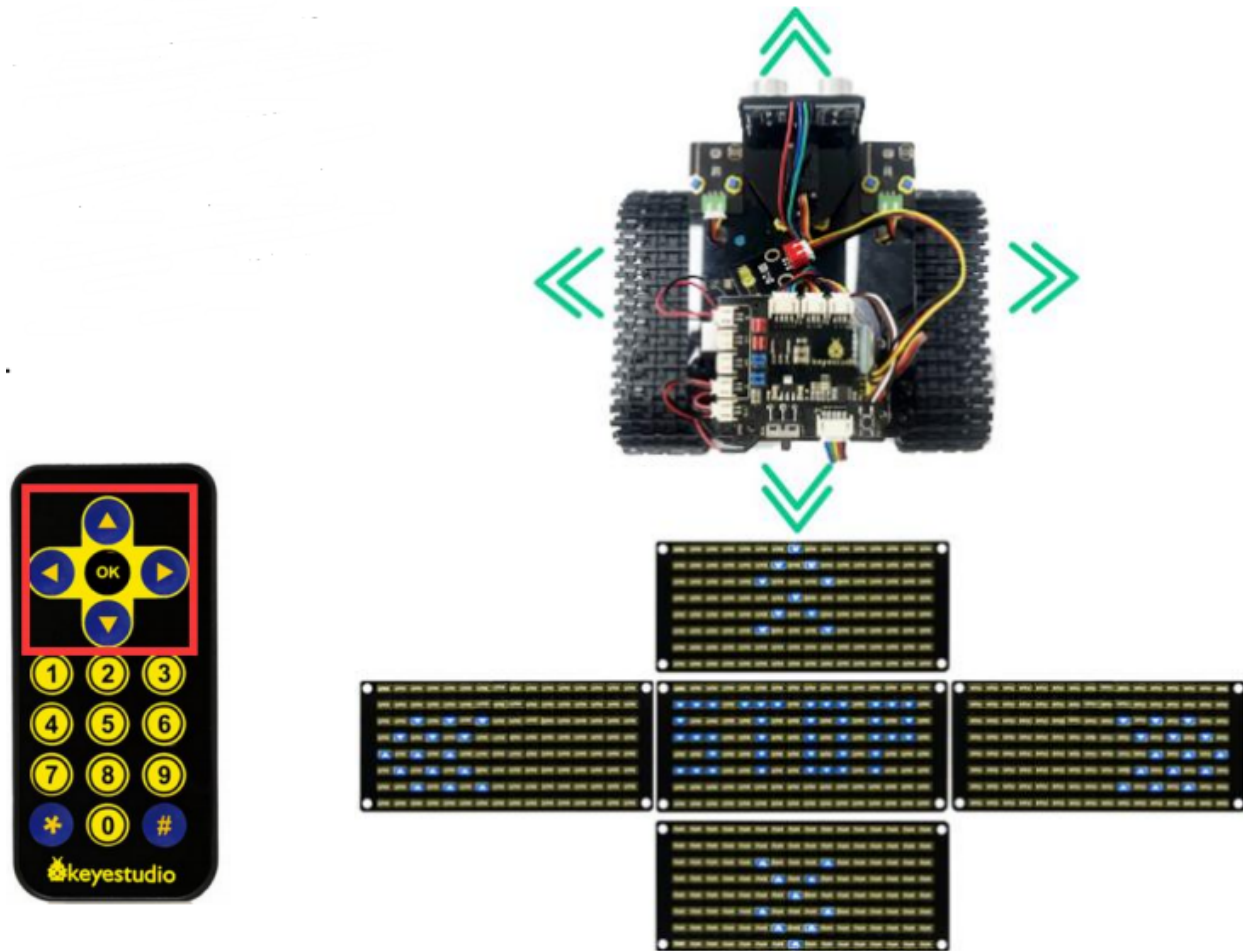
//transfer data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <=> 1) //each character has 8 digits, which is
    ↪ detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, HIGH); //Pull the clock pin SCL_Pin high to stop data
    ↪ transmission
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of
    ↪ SDA
    }
}

```

(6)Test Result:

After uploading the code, turn on the power switch of the motor drive shield. Place the robot on the floor, refer to the table above and press different buttons, the robot will move in the corresponding preset direction.





6.3.16 Project 16: Bluetooth Remote Control



(1)Description:

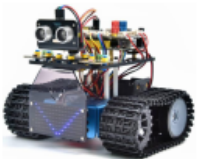

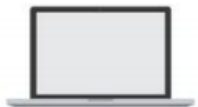


In the last several decades, Bluetooth has become the most popular wireless communication module for it is easy to use and has found wide applications in most devices powered by batteries.

In order to adjust with the time and reality and need the needs of customers, Bluetooth has been upgraded several times. In recent years, it embraces lots of transformations in terms of data transfer rate, power consumption of wearable devices and IoT devices, and security systems and others. Here, we plan to learn about DX-BT24 with Arduino board.

(2)Parameter:

- Bluetooth Protocol: Bluetooth Specification V5.1 BLE
- Serial port sending and receiving without byte limit
- Communication distance: 40m (open environment)
- Operating frequency: 2.4GHz ISM band
- Modulation method: GFSK (Gaussian Frequency Shift Keying)
- Security Features: Authentication and Encryption
- Support Services: Central and Peripheral UUIDs FFE0, FFE1, FFE2
- Power consumption: automatic sleep mode, standby current 400uA~800uA, 8.5mA during transmission.
- Power supply: 5V
- Operating temperature: -10 to +65 degrees Celsius

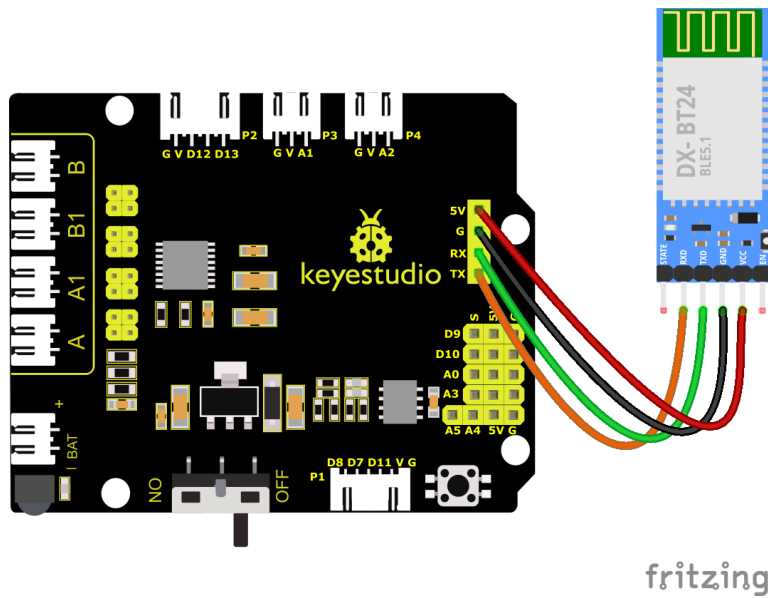
(3)Application:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

(4)Connection Diagram:

- 1.STATE is the status test pin connected to the internal light-emitting diode and usually remains unconnected.
- 2.RXD is the serial port interface for receiving terminal.
- 3.TXD is the serial port interface for sending terminal.
- 4.GND is for ground.
- 5.VCC is the positive pole.
- 6.EN/BRK: the disconnection of it represents the disconnection of the Bluetooth and it usually remains unconnected.

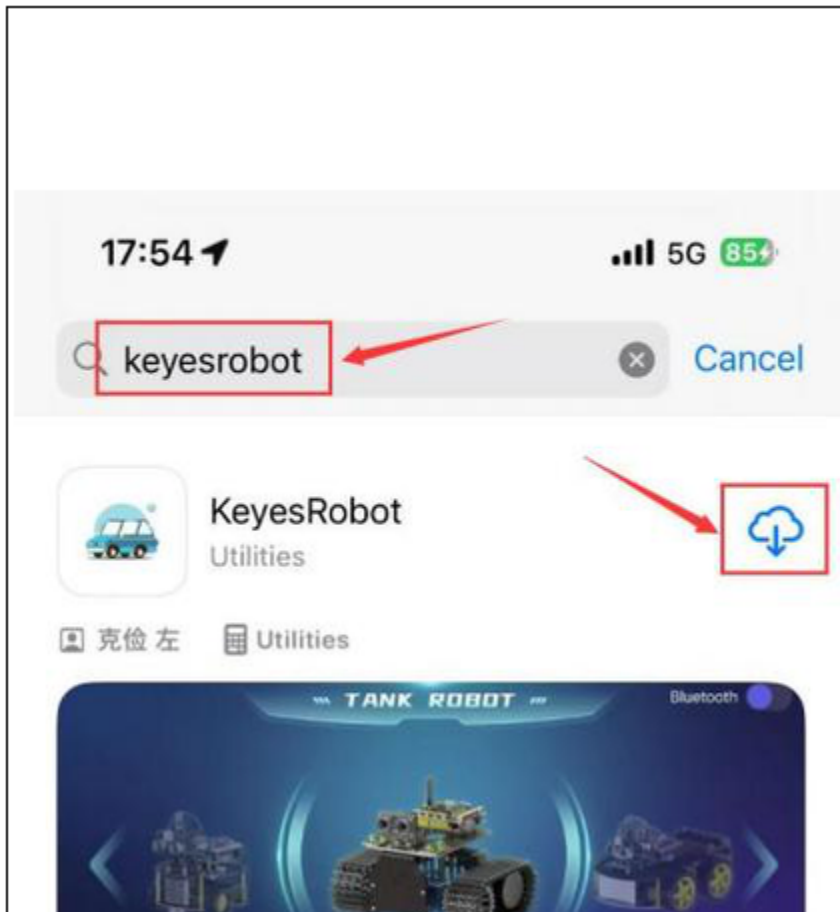
(Note: here the Bluetooth is directly linked with the V2 shield and please pay attention to the direction)



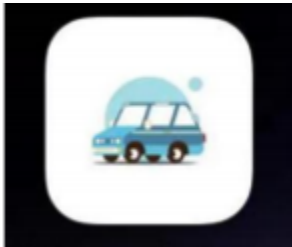
(5)Download and install APP:

For IOS system

1. Open App Store.
2. Search KeyesRobot in the Apple Store and click download.

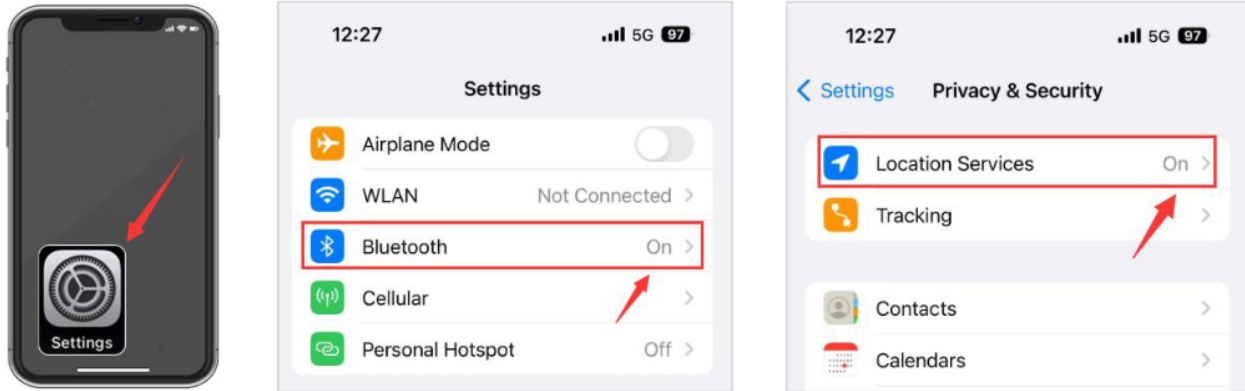


3. After the app is installed, you will see the following icon on your phone desktop.

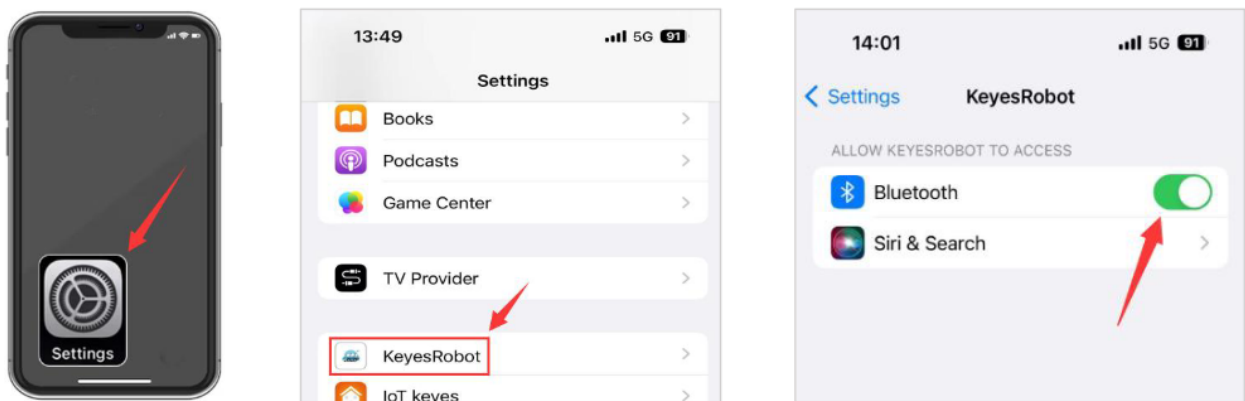


How to connect iOS Phone to Bluetooth module:

1. Turn on the Bluetooth and location services on phone through settings.



2. Allow KeyesRobot APP to access Bluetooth through settings.



3. Click to open KeyesRobot App.

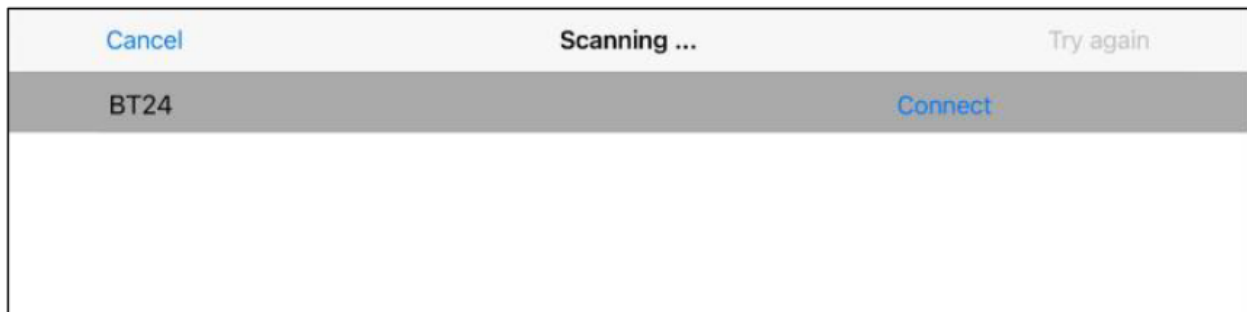


4. KeyesRobot App is a universal APP, which is applied to multiple keystudio robots. If the interface does not display “TANK ROBOT”, you can click the left and right buttons to find “TANK ROBOT”.

5. Click the Bluetooth button  in the upper right corner to scan the bluetooth



6. You will see a Bluetooth named **BT24**, click the Connect button.



7. If the onboard LED on the Bluetooth module stops flashing and stays on, it means your phone is successfully connected to the Bluetooth module.



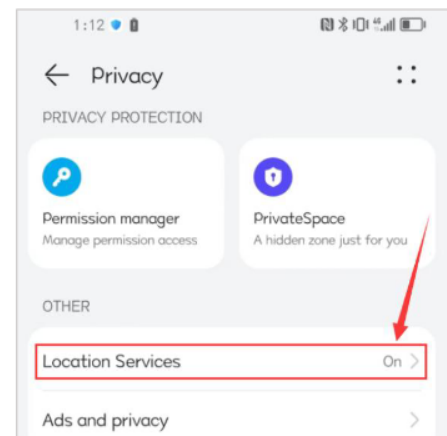
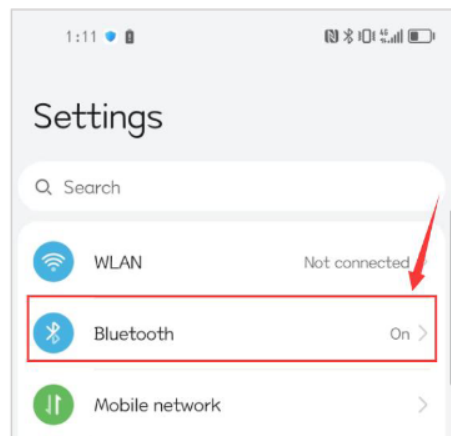
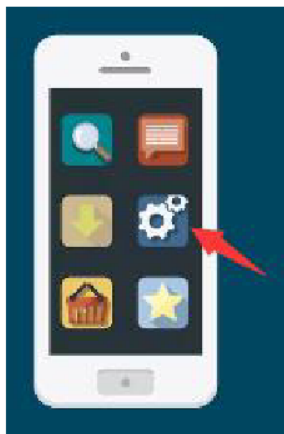
For Android System

1. Search **KeyesRobot** in Google Play, or open the following link to download and install the app.

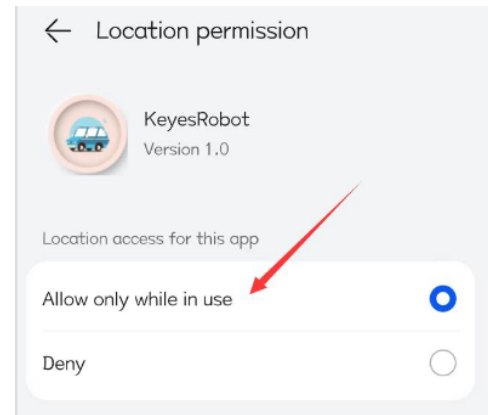
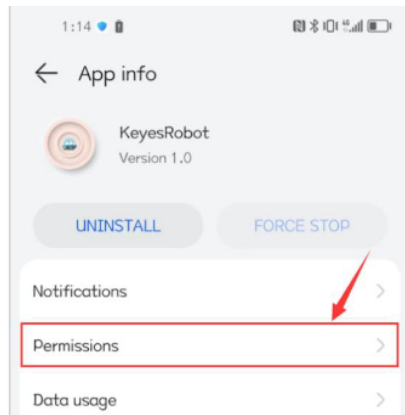
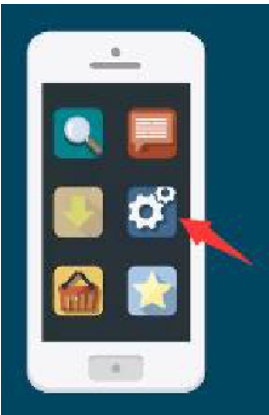
<https://play.google.com/store/apps/details?id=com.keyestudio.keyestudio>



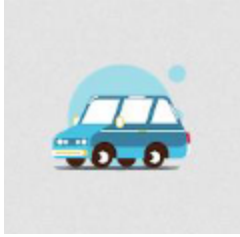
2. Turn on the Bluetooth and the location services of the mobile phone



3. Find the KeyesRobot Bluetooth app from settings, click on the permission options of the app, and enable Location and nearby device permissions.(Note: Some mobile phones do not have nearby device permissions function.)



4. Click to open KeyesRobot App.



5. KeyesRobot App is a universal APP, which is applied to multiple keyestudio robots. If the interface does not display “TANK ROBOT”, you can click the left and right buttons to find “TANK ROBOT”.

6. Click the Bluetooth button  in the upper right corner to scan the bluetooth



7. You will see a Bluetooth named **BT24**, click the Connect button.



8. When your phone is successfully connected to the Bluetooth module, the onboard LED on the Bluetooth module

will stop flashing and stay on.



(5) Test the Bluetooth APP:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*  
Keyestudio Mini Tank Robot V3 (Popular Edition)  
lesson 16.1  
Bluetooth  
http://www.keyestudio.com  
*/  
  
char ble_val; //Character variable(used to store the value received by Bluetooth)  
  
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {
```

(continues on next page)

(continued from previous page)

```

    if (Serial.available() > 0) //Determine whether there is data in the serial port
    ↪buffer
    {
        ble_val = Serial.read(); //Read the data in the serial port buffer
        Serial.println(ble_val); //Print out
    }
}

```

Upload the code to the development board, then plug in the Bluetooth module, and then connected the mobile phone to the Bluetooth module.

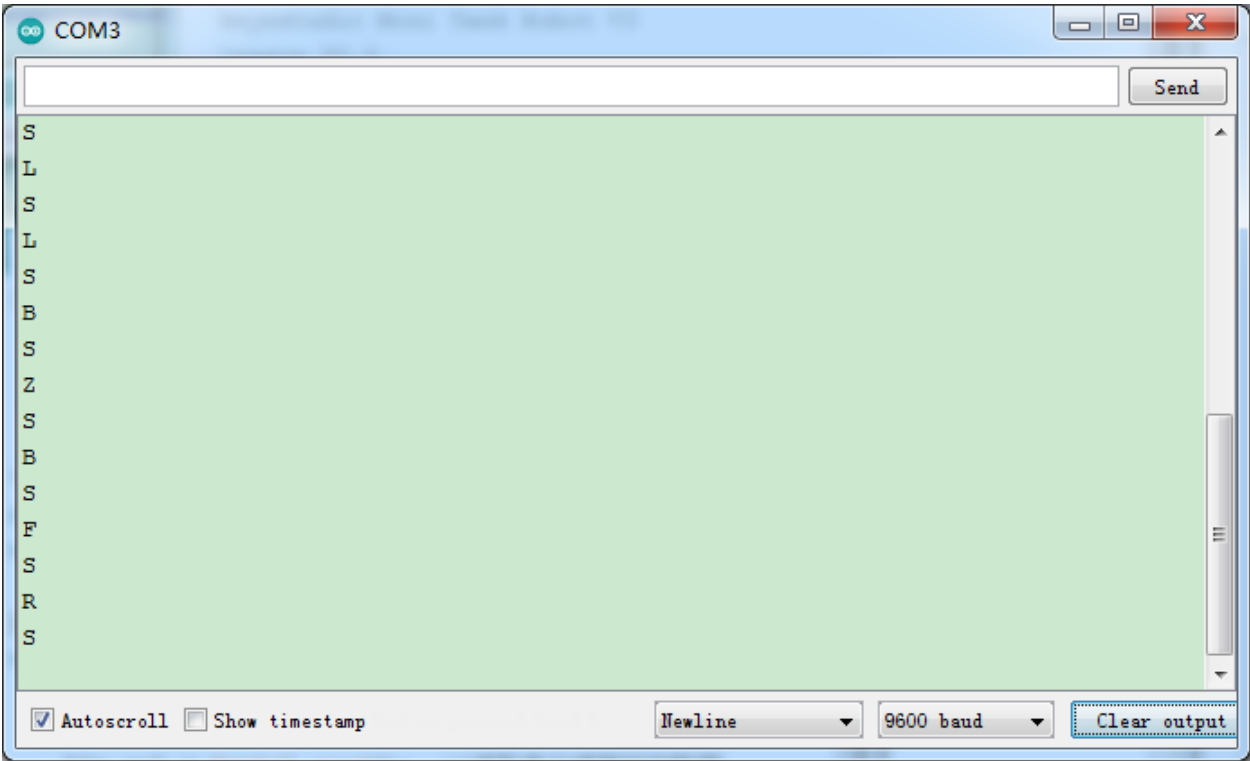
After the mobile phone is successfully connected to the Bluetooth module, click to open the Bluetooth APP and click the Select button on the homepage.



The main interface of the Bluetooth app is shown in the figure below.



After the code above is successfully uploaded, open the serial monitor of the arduino IDE and set the baud rate to 9600. Click the icon on the APP interface and the serial monitor will display command sent by button.










(6)Code Explanation:

Serial.available() represents the number of characters currently remaining in the serial port buffer.

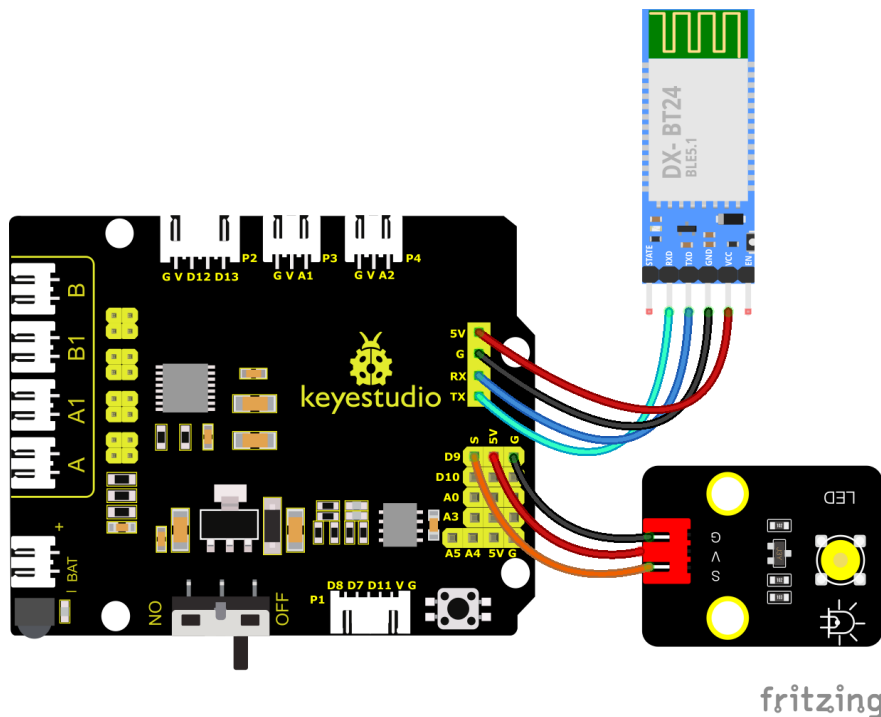
This function is generally used to determine whether there is data in this area. When `Serial.available()`>0, it means that the serial port has received data and can be read.

Serial.read() refers to taking out and reading a Byte of data from the serial port buffer. For example, if a device sends data to the Arduino through the serial port, we can use `Serial.read()` to read the sent data.

(7)Expansion Project:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1
			
Yellow LED Module*1	3P-3P XH2.54 to 2.54 Dupont Wire	18650 Battery*2	
			

Here we use the command sent by the mobile phone to turn on or off an LED light. Looking at the wiring diagram, an LED is connected to the D9 pin,



Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*
Keystudio Mini Tank Robot V3 (Popular Edition)
lesson 16.2
Bluetooth
http://www.keystudio.com
*/

int LED = 9;
char ble_val; //Integer variable used to store the value received by Bluetooth

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}

void loop() {

  if (Serial.available() > 0) //Determine whether there is data in the serial port
  {
    ble_val = Serial.read(); //Read data from serial port buffer
    Serial.print("DATA RECEIVED:");
  }
}
```

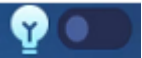
(continues on next page)

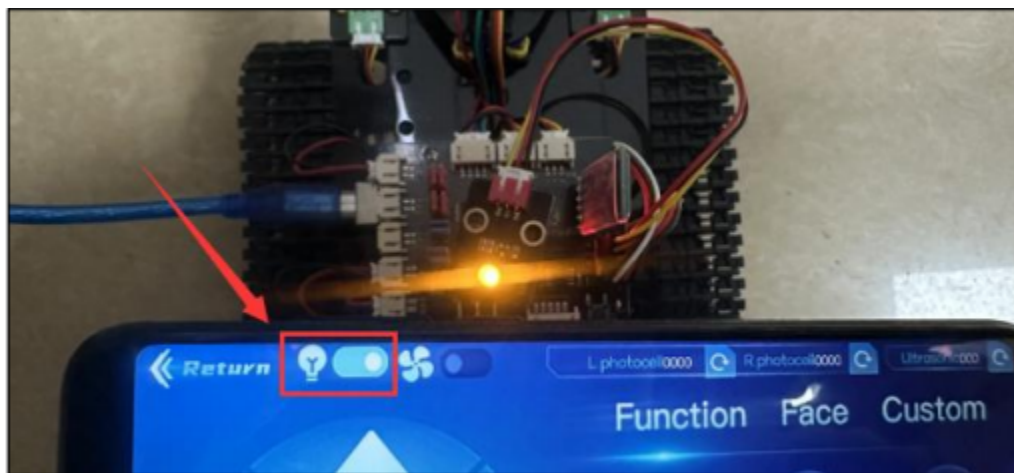
(continued from previous page)

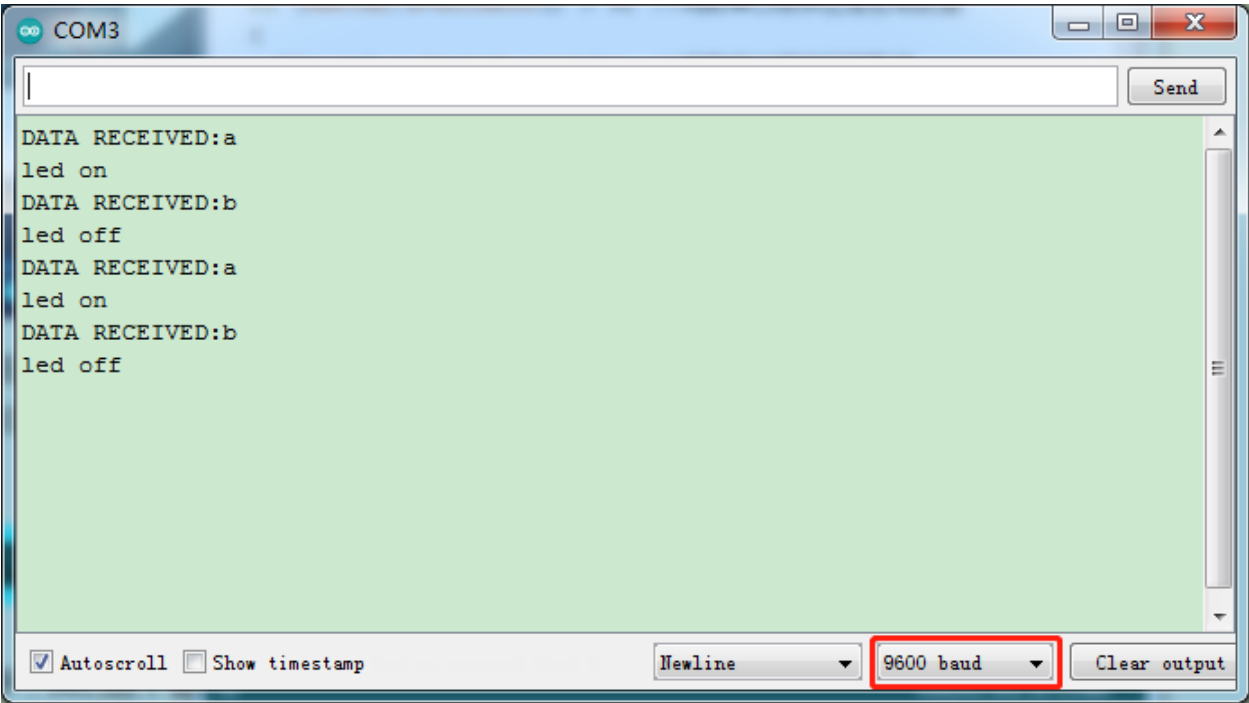
```
Serial.println(ble_val);
if (ble_val == 'a') {
    digitalWrite(LED, HIGH);
    Serial.println("led on");
}
if (ble_val == 'b') {
    digitalWrite(LED, LOW);
    Serial.println("led off");
}
}
```



After the code above is successfully uploaded, open the serial monitor of the arduino IDE and set the baud rate to 9600.

Click  to control the LED. When clicking it, a character a will be sent, then LED will be on. If this button is pressed again, the LED will be off.





You need to remove the BT module if you finish projects.

6.3.17 Project 17: Bluetooth Control Tank

(1)Description:

We have learned the basic knowledge of Bluetooth in the previous project . In this lesson, we will use Bluetooth to control the smart car. Since it involves Bluetooth, a sending end and a receiving end are needed. In the project, we use the mobile phone as the sender (master), and the smart car connected with the HM-10 Bluetooth module (slave) as the receiver.

We have learned earlier that sending a bit can control LEDs. And the principle of controlling this robot car is the same. We first understand the function of each button on the APP, and then use the button on the APP to control the tank.

(2)Key Function on the APP

The following table illustrates the functions of corresponding keys:

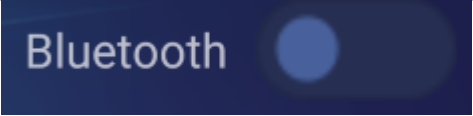

KEYS	FUNCTIONS
	Pair and connect HM-10 Bluetooth module; click again to disconnect
	select the robot to operate

Table 1 – continued from previous page







KEYS	FUNCTIONS
	to control the movements of the robot by buttons
	To control the movements of the robot by joystick
	To control the movements of the robot by gravity
	Send “F”when pressed and “S”when releasedThe car mov
	Send “L”when pressed and “S”when released The car turn
	Send “R”when pressed and “S”when releasedThe car turn

Table 1 – continued from previous page

KEYS	FUNCTIONS
	Send “B”when pressed and “S”when releasedThe car turn
	Send “u”+digit+“#”when draggedDrag to change the speed
	Send “v”+digit+“#”when draggedDrag to change the speed
	Select to enter Function page
	Send “G”when pressed and “S”when pressed againEnter o
	Send “h”when pressed and “S”when pressed againEnter f
	Send “e”when pressed and “S”when pressed againEnter li
	Send “f”when pressed and “S”when pressed againEnter m
	Send “i”when pressed and “S”when pressed againEnter li

Table 1 – continued from previous page

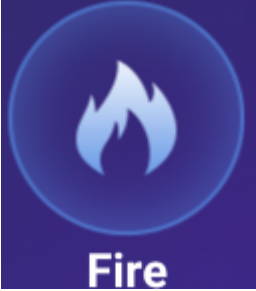

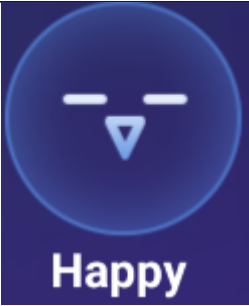
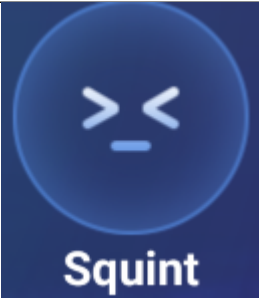
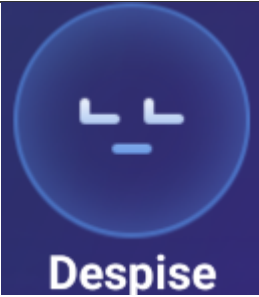

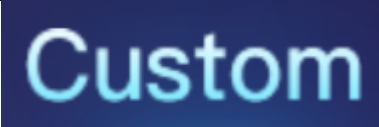



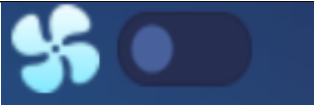
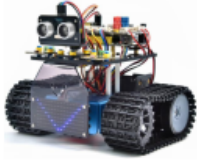




KEYS	FUNCTIONS
	Send “j”when pressed and “S”when pressed againEnter fi
	Select to enter facial expression display mode
	Send “k”when pressed and “z”when pressed againShow s
	Send “l”when pressed and “z”when pressed againShow d
	Send “m”when pressed and “z”when pressed againShow l

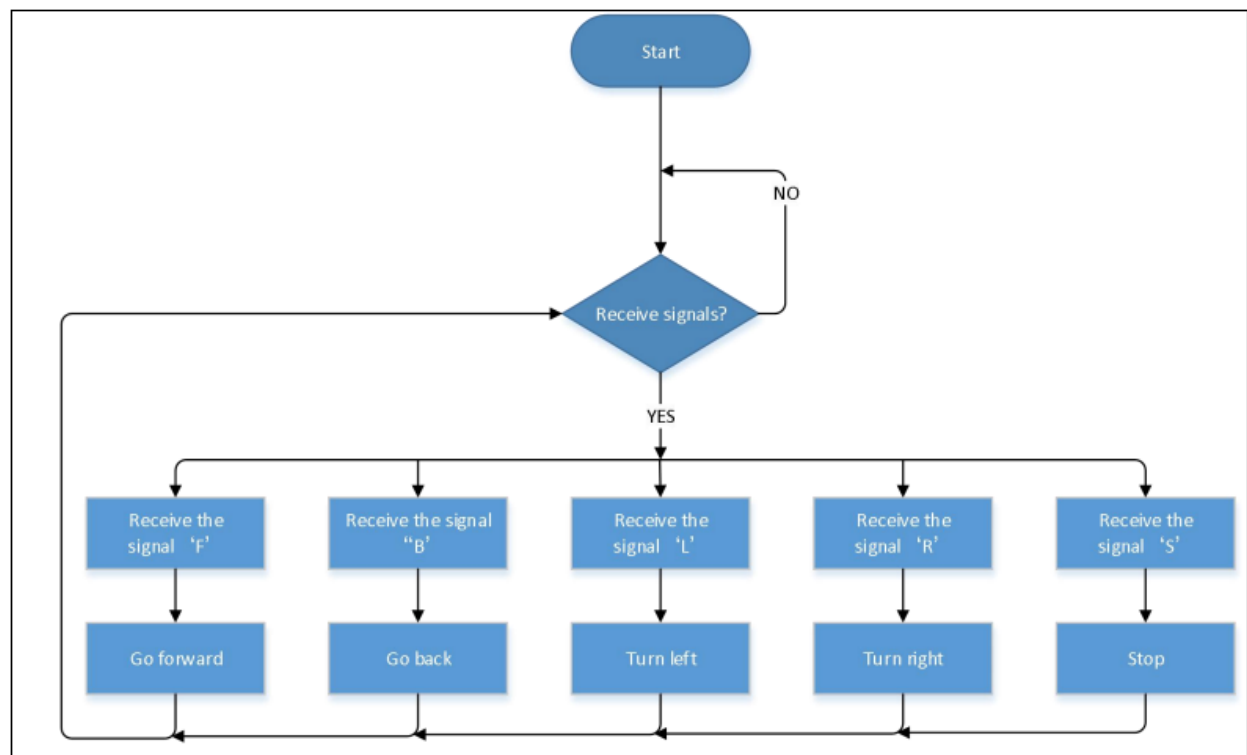
Table 1 – continued from previous page

KEYS	FUNCTIONS
	Send “n”when pressed and “z”when pressed againShow s
	Send “o”when pressed and “z”when pressed againShow d
	Send “p”when pressed and “z”when pressed againShow h
	Choose to enter the custom function interface; there are si
	Click to send “w”Click to display the analog value detecte
	Click to send“y”Click to display the analog value detected
	Click to send“x” Click to show the distance detected by ul
	Click to send“c” Click again to send“d”Press to turn on th

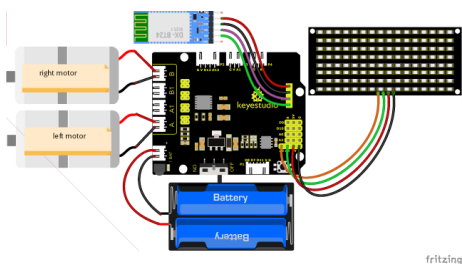
(3) Components Required:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

(4) Flow Chart:



(5) Wiring Diagram:



The GND, VCC, SDA, and SCL of the 8x16 LED dot matrix are respectively connected to-(GND), + (VCC), SDA, SCL of the expansion board;

The STATE and BRK pins of the Bluetooth module do not need to be connected.

(6)Test Code:

(Note: When uploading the code, the Bluetooth module must be unplugged, and the Bluetooth can be reconnected after the uploading process. Otherwise the code may not be burned.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 17.
  bluetooth Control tank
  http://www.keyestudio.com
*/

//Array, used to save data of images, can be calculated by yourself or gotten from
↳modulus tool
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40,
↳0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00,
↳0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28,
↳0x10, 0x44, 0x28, 0x10, 0x00};
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22,
↳0x3E, 0x00, 0x3E, 0x0A, 0x0E, 0x00};
unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
#define SCL_Pin  A5 //Set the clock pin as A5
#define SDA_Pin  A4 //Set the data pin as A4

#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
char ble_val; //Used to store the value obtained by Bluetooth

void setup() {
  Serial.begin(9600);

  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  pinMode(SCL_Pin, OUTPUT);
  pinMode(SDA_Pin, OUTPUT);
  matrix_display(clear); //clear screens

```

(continues on next page)

(continued from previous page)

```

    matrix_display(start01); //show the image to start
}

void loop() {
    if (Serial.available())
    {
        ble_val = Serial.read();
        Serial.println(ble_val);
    }
    switch (ble_val)
    {
        case 'F': //the command to go front
            Car_front();
            break;
        case 'B': //the command to go back
            Car_back();
            break;
        case 'L': //the command to turn left
            Car_left();
            break;
        case 'R': //the command to turn right
            Car_right();
            break;
        case 'S': //the command to stop
            Car_Stop();
            break;
    }
}

/*****The function to run the motor*****/
void Car_back() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
    matrix_display(back); //Go back
}

void Car_front() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
    matrix_display(front); //show the image to go front
}

void Car_left() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 55);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 200);
    matrix_display(left); //show the image to turn lefr

```

(continues on next page)

(continued from previous page)

```

}

void Car_right() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 200);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 55);
    matrix_display(right); //show the image to turn right
}

void Car_Stop() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
    matrix_display(STOP01); //show the image to stop
}

//This function is used for dot matrix screen display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //Function to call data transfer start condition
    IIC_send(0xc0); //Choose an address
    for (int i = 0; i < 16; i++) //Pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //transfer pattern data
    }
    IIC_end(); //End pattern data transfer
    IIC_start();
    IIC_send(0x8A); //display control, select pulse width as 4/16
    IIC_end();
}

//Conditions for the start of data transfer
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

//the sign of ending data transmission
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
}

```

(continues on next page)

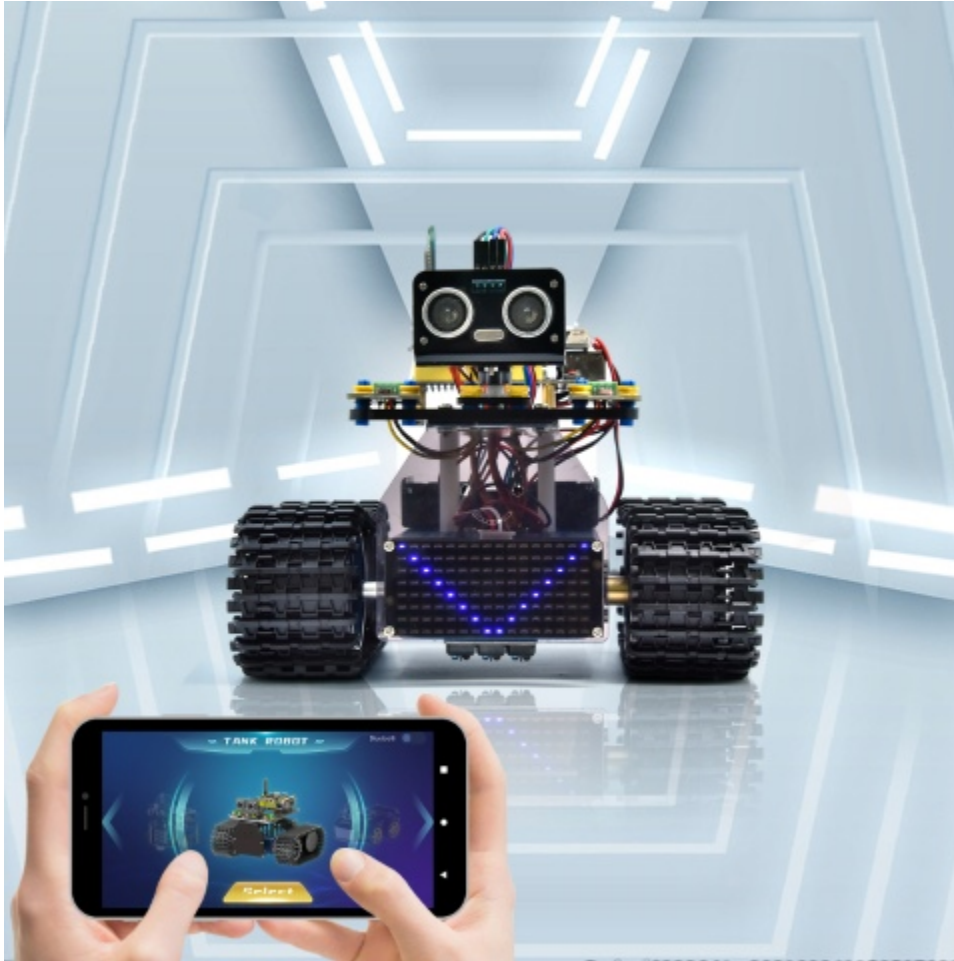
(continued from previous page)

```
digitalWrite(SDA_Pin, HIGH);
delayMicroseconds(3);
}

//transfer data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <=<= 1) //each character has 8 digits, which is
    ↪ detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, HIGH); //Pull the clock pin SCL_Pin high to stop data
    ↪ transmission
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of
    ↪ SDA
    }
}
```

(7)Test Result:

After uploading the code, connect the robot to the Bluetooth module and pair the Bluetooth APP. Turn on the power-switch of the motor drive shield. Place the robot on the floor, you can use these buttons of the Bluetooth app to control the robot.

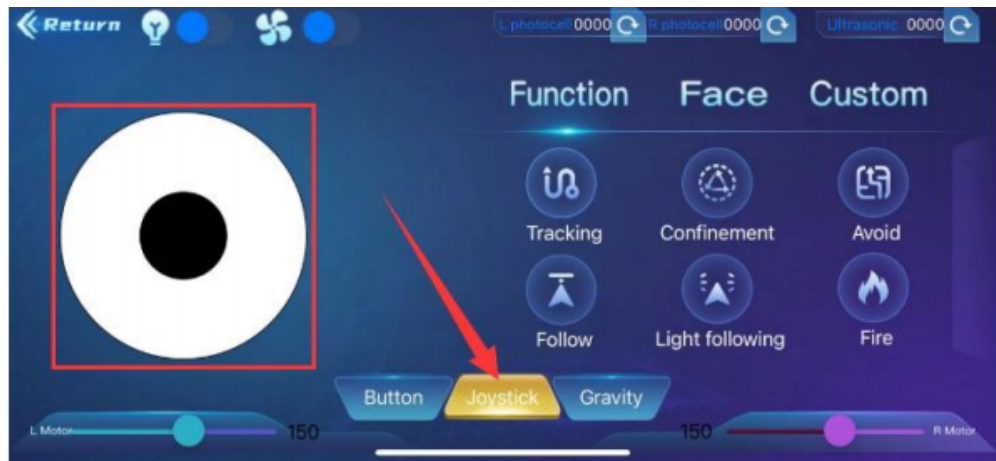


1. The up, down, left and right arrows control the robot to move forward, backward, left and right respectively.



2. Click the joystick button and pull the direction of the black point in the white circle to control the movement direction of the robot.

2



3. Click the Gravity button and tilt the phone in the forward, backward, left, and right directions, and the robot will move in the direction in which the phone is tilted.

3



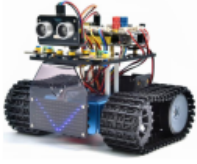




6.3.18 Project 18: BT Speed Control Robot

(1)Description:

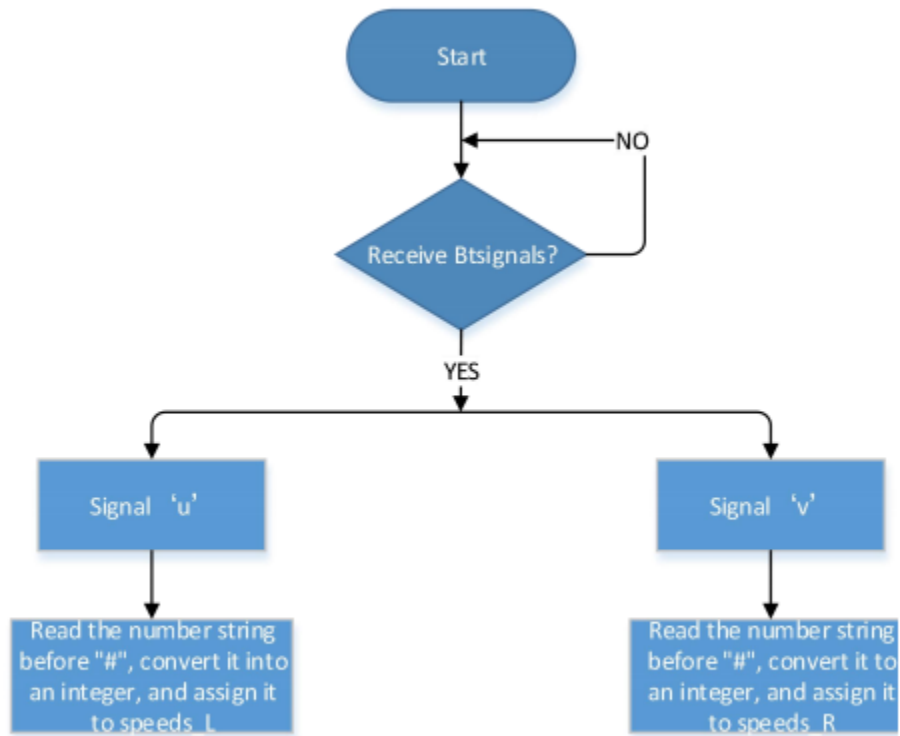
In the previous project, we learned how to control the smart tank with Bluetooth. The PWM value of the motor we used in front of us is 200 (the speed is 200).

In this lesson, we will use Bluetooth to adjust the speed of the smart car. It is not limited to Fixed speed of 200. We define two variables to store the speed values of the left and right motors respectively. Through the previous study, we know that the range of this value can only take 0 to 255.

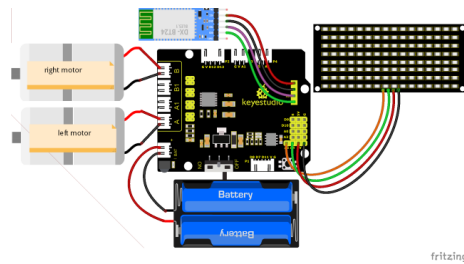
(2) Components Required:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

(3) Flow chart:



(4)Connection Diagram:



The GND, VCC, SDA, and SCL of the 8x16 LED dot matrix are respectively connected to-(GND), + (VCC), SDA, SCL of the expansion board;

(5)Test Code:

(Note: When uploading the code, the Bluetooth module must be unplugged, and the Bluetooth can be reconnected after the uploading process. Otherwise the code may not be burned.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 18
  bluetooth control speed tank
  http://www.keyestudio.com
*/

//Array, used to save data of images, can be calculated by yourself or gotten from
↳modulus tool
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40,
↳0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00,
↳0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28,
↳0x10, 0x44, 0x28, 0x10, 0x00};
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22,
↳0x3E, 0x00, 0x3E, 0x0A, 0x0E, 0x00};
unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char speed_a[] = {0x00, 0x00, 0x00, 0x20, 0x10, 0x08, 0x04, 0x02, 0xff, 0x02,
↳0x04, 0x08, 0x10, 0x20, 0x00, 0x00};
unsigned char speed_d[] = {0x00, 0x00, 0x00, 0x04, 0x08, 0x10, 0x20, 0x40, 0xff, 0x40,
↳0x20, 0x10, 0x08, 0x04, 0x00, 0x00};
#define SCL_Pin  A5 //set the pin of clock to A5
#define SDA_Pin  A4 //A4 set data pin to A4

#define ML_Ctrl 4 //define the direction control pin of the left motor
#define ML_PWM 6 //define the PWM control pins of the left motor
#define MR_Ctrl 2 //define the direction control pin of the right motor
#define MR_PWM 5 //define the PWM control pin of the right motor

```

(continues on next page)

(continued from previous page)

```

char ble_val;           //define the PWM control pin of the right motor
byte speeds_L = 200; //The initial speed of the left motor is 200
byte speeds_R = 200; //The initial speed of the right motor is 200
String speeds_l, speeds_r; //Receive a string of PWM to convert to an integer PWM value

void setup() {
  Serial.begin(9600);

  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  pinMode(SCL_Pin, OUTPUT);
  pinMode(SDA_Pin, OUTPUT);
  matrix_display(clear); //clear screens
  matrix_display(start01); //show the image to start
}

void loop() {
  if (Serial.available() > 0) {
    ble_val = Serial.read();
    Serial.println(ble_val);
    switch (ble_val) {
      case 'F': //the command to go front
        Car_front();
        break;
      case 'B': //the command to go back
        Car_back();
        break;
      case 'L': //the command to turn left
        Car_left();
        break;
      case 'R': //the command to turn right
        Car_right();
        break;
      case 'S': //the command to stop
        Car_Stop();
        break;
      case 'u': //Receive a string starting with u and ending with #, and convert it to
↳an integer value
        speeds_l = Serial.readStringUntil('#');
        speeds_L = String(speeds_l).toInt();
        break;
      case 'v': //Receive a string starting with v and ending with #, and convert it to
↳an integer value
        speeds_r = Serial.readStringUntil('#');
        speeds_R = String(speeds_r).toInt();
        break;
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

/*****The function to run the motor*****/

void Car_back() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, speeds_R);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, speeds_L);
    matrix_display(back); //Go back
}

void Car_front() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 255 - speeds_R);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 255 - speeds_L);
    matrix_display(front); //show the image to go front
}

void Car_left() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 255 - speeds_R);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, speeds_L);
    matrix_display(left); //show the image to turn left
}

void Car_right() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, speeds_R);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 255 - speeds_L);
    matrix_display(right); //show the image to turn right
}

void Car_Stop() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
    matrix_display(STOP01); //show the image to stop
}

//This function is used for dot matrix screen display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //Function to call data transfer start condition
    IIC_send(0xc0); //Choose an address
    for (int i = 0; i < 16; i++) //Pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //transfer pattern data
    }
}

```

(continues on next page)

(continued from previous page)

```

IIC_end(); //End pattern data transfer
IIC_start();
IIC_send(0x8A); //display control, select pulse width as 4/16
IIC_end();
}

//Conditions for the start of data transfer
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

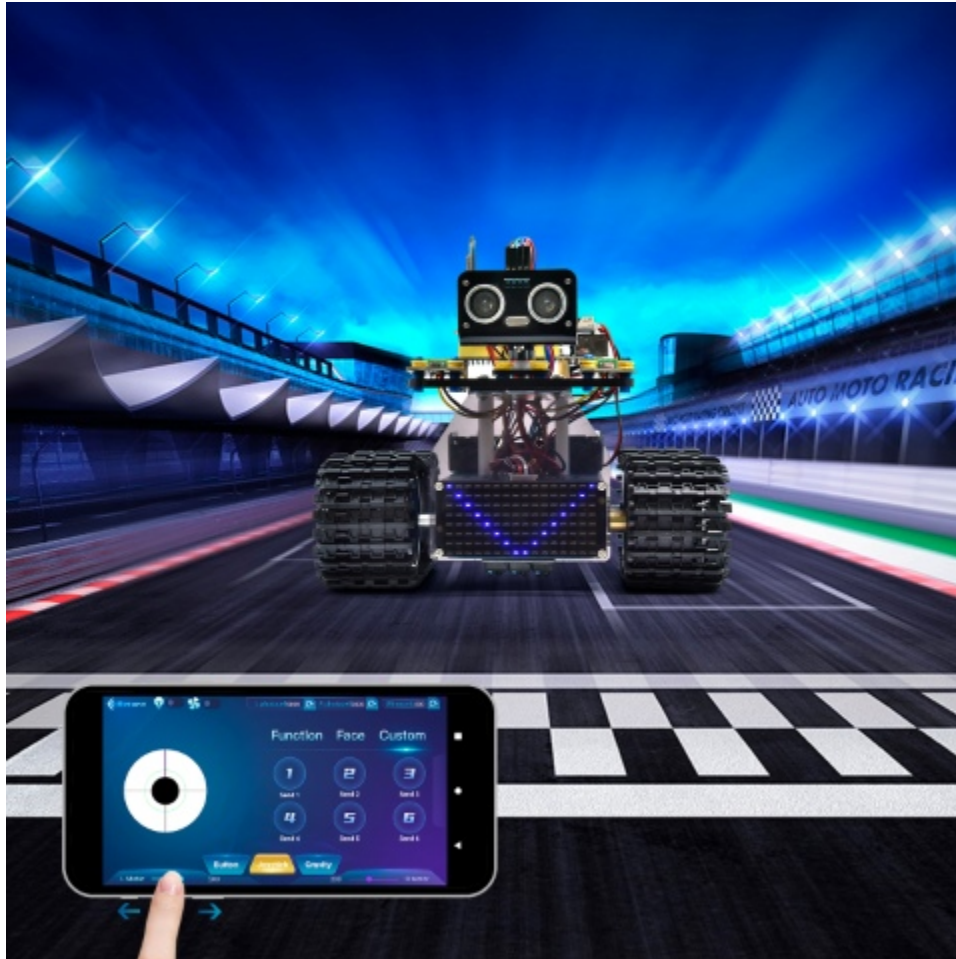
//the sign of ending data transmission
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, HIGH);
    delayMicroseconds(3);
}

//transfer data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <=< 1) //each character has 8 digits, which is
    ↳detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, HIGH); //Pull the clock pin SCL_Pin high to stop data
    ↳transmission
        delayMicroseconds(3);
        digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of
    ↳SDA
    }
}

```

(6)Test Results:

After uploading the test code successfully, dialing the DIP switch to the right end, powering it on, and pairing the APP with Bluetooth, the smart car can be controlled to move by the APP. And the speed of the car can be regulated by pulling the speed dials of the left and right motors.



(You can refer to function table in project 17)

6.3.19 Project 19: Ultrasonic Tank Robot Multiple Functions

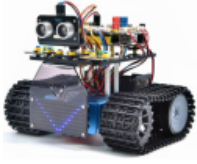




(1)Description:

The smart car has performed a single function in every previous project.

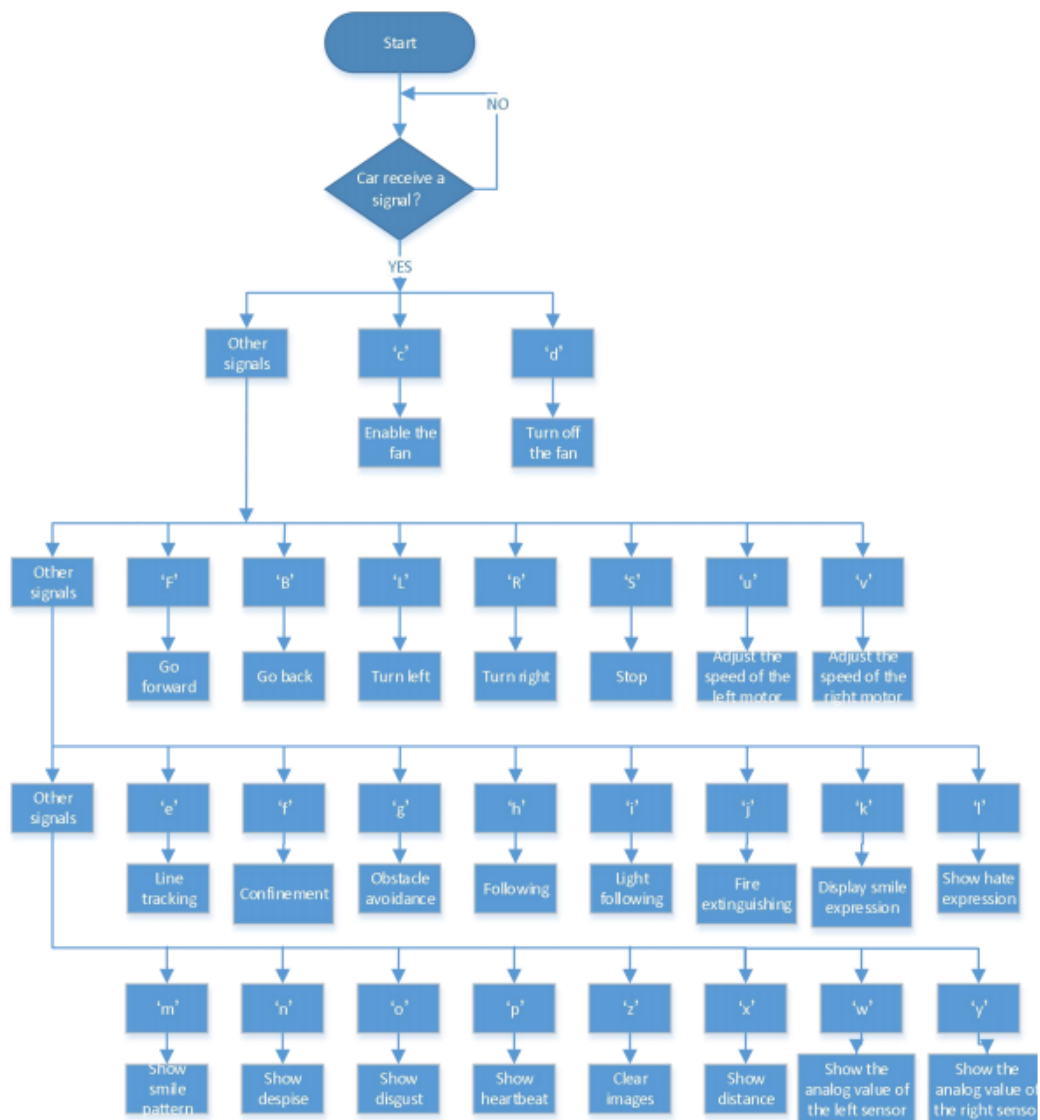
Can it display multiply functions at a time ? Positive.

In this last big project, we intend to use a complete code to control the smart car to show off all functions mentioned in previous projects. We use the keys on the Bluetooth APP to automatically switch various functions, quite simple and convenient.

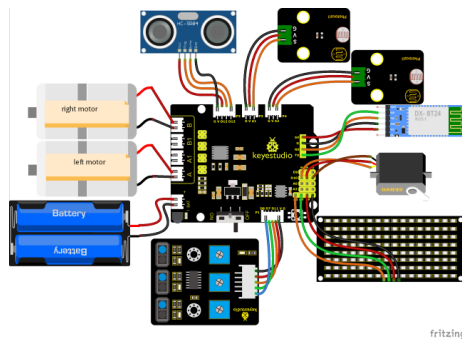
(2) Components Required:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

(3) Flow Diagram:



(4)Connection Diagram:



1. GND, VCC, SDA and SCL of the 8x16 board are connected to GGND), +VCC), A4 and A5 of the expansion board.
2. VCC, Trig, Echo and Gnd of the ultrasonic sensor are connected to 5V(V), 12(S), 13(S) and Gnd(G)
3. The brown wire, red wire and orange wire of the servo are connected to Gnd(G), 5v(V) and D10.
4. RXD, TXD, GND and VCC of the BT module are connected to TX, RX, GGND) and 5VVCC. STATE and BRK don't need to be interfaced.
5. The pin "G", "V" and S of the left photoresistor module are connected to G (GND), V (VCC), A1 respectively; The right photoresistor module is connected to the G (GND), V (VCC), and A2 respectively.
6. The distal port of the line tracking sensor is 11, 7 and 8.

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keystudio Mini Tank Robot V3 (Popular Edition)
  lesson 19
  Ultrasonic Tank Robot Multiple Functions
  http://www.keystudio.com
*/
#include <IRremote.h>
IRrecv irrecv(3); //
decode_results results;
long ir_rec; //used to save the IR value

/*****/
#define USE_FAN_FUNCTION 0

//Array, used to save data of images, can be calculated by yourself or gotten from
↳modulus tool
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40,
↳0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22,
↳0x3E, 0x00, 0x3E, 0x0A, 0x0E, 0x00};
unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24,
↳0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

```

(continues on next page)

(continued from previous page)

```

unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00,
↳ 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28,
↳ 0x10, 0x44, 0x28, 0x10, 0x00};
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44,
↳ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

unsigned char Smile[] = {0x00, 0x00, 0x1c, 0x02, 0x02, 0x02, 0x5c, 0x40, 0x40, 0x5c,
↳ 0x02, 0x02, 0x02, 0x1c, 0x00, 0x00};
unsigned char Disgust[] = {0x00, 0x00, 0x02, 0x02, 0x02, 0x12, 0x08, 0x04, 0x08, 0x12,
↳ 0x22, 0x02, 0x02, 0x00, 0x00, 0x00};
unsigned char Happy[] = {0x02, 0x02, 0x02, 0x02, 0x08, 0x18, 0x28, 0x48, 0x28, 0x18,
↳ 0x08, 0x02, 0x02, 0x02, 0x02, 0x00};
unsigned char Squint[] = {0x00, 0x00, 0x00, 0x41, 0x22, 0x14, 0x48, 0x40, 0x40, 0x48,
↳ 0x14, 0x22, 0x41, 0x00, 0x00, 0x00};
unsigned char Despise[] = {0x00, 0x00, 0x06, 0x04, 0x04, 0x04, 0x24, 0x20, 0x20, 0x26,
↳ 0x04, 0x04, 0x04, 0x04, 0x00, 0x00};
unsigned char Heart[] = {0x00, 0x00, 0x0C, 0x1E, 0x3F, 0x7F, 0xFE, 0xFC, 0xFE, 0x7F,
↳ 0x3F, 0x1E, 0x0C, 0x00, 0x00, 0x00};

unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
↳ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

#define SCL_Pin  A5  //set the pin of clock to A5
#define SDA_Pin  A4  //set the data pin to A4

#define ML_Ctrl 4  //define the direction control pin of the left motor as 4
#define ML_PWM 6  //define the PWM control pin of the left motor
#define MR_Ctrl 2  //define the direction control pin of the right sensor
#define MR_PWM 5  //define the PWM control pin of the right motor
char ble_val;      //used to save the Bluetooth value
byte speeds_L = 200; //the initial speed of the left motor is 200
byte speeds_R = 200; //the initial speed of the right motor is 200
String speeds_l, speeds_r; //receive PWM characters and convert them into PWM value

//wire up the line tracking sensor
#define L_pin  11  //left
#define M_pin  7   //middle
#define R_pin  8   //right
int L_val, M_val, R_val;

#if USE_FAN_FUNCTION  /****use fan*****/
int flame_L = A1; //define the analog port of the left flame sensor to A1
int flame_R = A2; //define the analog port of the right flame sensor to A2
int flame_valL, flame_valR;

//the pin of 130 motor
int INA = 12;
int INB = 13;

#else /****use the ultrasonic sensor*****/
#define servoPin  10  //servo Pin

```

(continues on next page)

(continued from previous page)

```

#define light_L_Pin A1    //define the pin of the left photoresistor
#define light_R_Pin A2    //define the pin of the right photoresistor
int left_light;
int right_light;

#define Trig 12
#define Echo 13
float distance; //Store the distance values detected by ultrasonic for following

//Store the distance values detected by ultrasonic for obstacle avoidance
int a;
int a1;
int a2;

#endif

bool flag; //flag invariable, used to enter and exit a mode

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn(); //Initialize the library of the IR remote

  pinMode(SCL_Pin, OUTPUT);
  pinMode(SDA_Pin, OUTPUT);

  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  pinMode(L_pin, INPUT); //define the pins of sensors to INPUT
  pinMode(M_pin, INPUT);
  pinMode(R_pin, INPUT);

  matrix_display(clear); //clear screen
  matrix_display(start01); //show start

#ifdef USE_FAN_FUNCTION //****use the fan*****/
  pinMode(INA, OUTPUT); //set INA to OUTPUT
  pinMode(INB, OUTPUT); //set INB to OUTPUT

  //define inputs of the flame sensor
  pinMode(flame_L, INPUT);
  pinMode(flame_R, INPUT);
#else //****use the ultrasonic sensor*****/
  pinMode(servoPin, OUTPUT);
  pinMode(light_L_Pin, INPUT);
  pinMode(light_R_Pin, INPUT);

  pinMode(Trig, OUTPUT);
  pinMode(Echo, INPUT);
  procedure(90); //set the angle of the servo to 90°

```

(continues on next page)

(continued from previous page)

```

#endif
}

void loop() {
  if (Serial.available()) //if there is data in the serial buffer
  {
    ble_val = Serial.read();
    Serial.println(ble_val);
    switch (ble_val) {
      case 'F': Car_front(); break; //the command to go front

      case 'B': Car_back(); break; //the command to go back

      case 'L': Car_left(); break; //the command to turn left

      case 'R': Car_right(); break; //the command to turn right

      case 'S': Car_Stop(); break; //stop

      case 'e': Tracking(); break; //enter the line tracking mode

      case 'f': Confinement(); break; //enter the confinement mode

      #if USE_FAN_FUNCTION/****use fan*****/
      case 'j': Fire(); break; //enable extinguishing fire mode

      case 'c': fan_begin(); break; //enable the fan

      case 'd': fan_stop(); break; //turn off the fan

      #else/****use the ultrasonic sensor*****/
      case 'g': Avoid(); break; //enter obstacle avoidance mode

      case 'h': Follow(); break; //enter light following mode

      case 'i': Light_following(); break; //enter light following mode
    #endif
    case 'u':
      speeds_l = Serial.readStringUntil('#');
      speeds_L = String(speeds_l).toInt();
      break; //start by receiving u, end by receiving characters # and convert into
      ↪ the integer

    case 'v':
      speeds_r = Serial.readStringUntil('#');
      speeds_R = String(speeds_r).toInt();
      break; //start by receiving u, end by receiving characters # and convert into
      ↪ the integer

    case 'k': matrix_display(Smile); break; //show "smile" face
    case 'l': matrix_display(Disgust); break; //show "disgust" face
    case 'm': matrix_display(Happy); break; //show "happy" face
  }
}

```

(continues on next page)

(continued from previous page)

```

        case 'n': matrix_display(Squint);    break;    //show "Sad" face
        case 'o': matrix_display(Despise);  break;    //show "despise" face
        case 'p': matrix_display(Heart);    break;    //show the heartbeat image
        case 'z': matrix_display(clear);    break;    //clear images

        default: break;
    }
}

#if (USE_FAN_FUNCTION != 1)/*the function to not use the fan*****/
//The following three signals are mainly used for cyclic printing
if (ble_val == 'x') {
    distance = checkdistance(); Serial.println(distance);
    delay(50);
} else if (ble_val == 'w') {
    left_light = analogRead(light_L_Pin);
    Serial.println(left_light);
    delay(50);
} else if (ble_val == 'y') {
    right_light = analogRead(light_R_Pin);
    Serial.println(right_light);
    delay(50);
}
#endif

if (irrecv.decode(&results)) { //Receive infrared remote control value
    ir_rec = results.value;
    Serial.println(ir_rec, HEX);
    switch (ir_rec) {
        case 0xFF629D: Car_front();    break;    //go front
        case 0xFFA857: Car_back();     break;    //go back
        case 0xFF22DD: Car_left();     break;    //rotate to left
        case 0xFFC23D: Car_right();    break;    //rotate to right
        case 0xFF02FD: Car_Stop();     break;    //stop
        default: break;
    }
    irrecv.resume();
}
}

#if (USE_FAN_FUNCTION != 1)/*use the ultrasonic sensor*****/
//Control the ultrasonic sensor
float checkdistance() {
    float distance;
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);
    distance = pulseIn(Echo, HIGH) / 58.20;    //
    delay(10);
}

```

(continues on next page)

(continued from previous page)

```

    return distance;
}

//the function to control the servo
void procedure(int myangle) {
    int pulsewidth;
    pulsewidth = map(myangle, 0, 180, 500, 2000); //Calculate the pulse width value,
    ↪which should be the mapping value from 500 to 2500. Considering the influence of the
    ↪infrared library, 500~2000 is used here.
    for (int i = 0; i < 5; i++) {
        digitalWrite(servoPin, HIGH);
        delayMicroseconds(pulsewidth); //The duration of the high level is the pulse width
        digitalWrite(servoPin, LOW);
        delay((20 - pulsewidth / 1000)); //The period is 20ms, so the low level lasts the
    ↪rest of the time
    }
}

/*****obstacle avoidance*****/
void Avoid()
{
    flag = 0;
    while (flag == 0)
    {
        a = checkdistance(); //the front distance is set to a
        if (a < 20) { //When the distance in front is less than 20cm
            Car_Stop(); //stop
            delay(500); //delay in 500ms
            procedure(180); //servo turns left
            delay(500); //delay in 500ms
            a1 = checkdistance(); //the left distance is set to a1
            delay(100); //read value

            procedure(0); //servo turns right
            delay(500); //delay in 500ms
            a2 = checkdistance(); //the right distance is set to a2
            delay(100); //read value

            procedure(90); //back to 90°
            delay(500);
            if (a1 > a2) { //When the distance on the left is greater than the distance on the
    ↪right
                Car_left(); //the robot turns left
                delay(700); //turn left 700ms
            } else {
                Car_right(); //turn right
                delay(700);
            }
        }
        else { //if the front distance 20cmrobot goes front
            Car_front(); //go front

```

(continues on next page)

(continued from previous page)

```

    }
    //receive the Bluetooth value to exit the loop
    if (Serial.available())
    {
        ble_val = Serial.read();
        if (ble_val == 'S') //receive S
        {
            flag = 1; //Set flag to 1 to exit the loop
            Car_Stop();
        }
    }
}

/*****following*****/
void Follow() {
    flag = 0;
    while (flag == 0) {
        distance = checkdistance(); //set the distance value to distance
        if (distance >= 20 && distance <= 60) //go front
        {
            Car_front();
        }
        else if (distance > 10 && distance < 20) // stop
        {
            Car_Stop();
        }
        else if (distance <= 10) //go back
        {
            Car_back();
        }
        else //stop
        {
            Car_Stop();
        }
        if (Serial.available())
        {
            ble_val = Serial.read();
            if (ble_val == 'S')
            {
                flag = 1; //exit the loop
                Car_Stop();
            }
        }
    }
}

/*****light following*****/
void Light_following() {
    flag = 0;
    while (flag == 0) {
        left_light = analogRead(light_L_Pin);
    }
}

```

(continues on next page)

(continued from previous page)

```

    right_light = analogRead(light_R_Pin);
    if (left_light > 650 && right_light > 650) //go forward
    {
        Car_front();
    }
    else if (left_light > 650 && right_light <= 650) //turn left
    {
        Car_left();
    }
    else if (left_light <= 650 && right_light > 650) //turn right
    {
        Car_right();
    }
    else //or else, stop
    {
        Car_Stop();
    }
    if (Serial.available())
    {
        ble_val = Serial.read();
        if (ble_val == 'S') {
            flag = 1;
            Car_Stop();
        }
    }
}

#else/****use the fan*****/
/*****enable the fan*****/
void fan_begin() {
    digitalWrite(INA, LOW);
    digitalWrite(INB, HIGH);
}

/*****stop fanning*****/
void fan_stop() {
    digitalWrite(INA, LOW);
    digitalWrite(INB, LOW);
}

/*****extinguish fire*****/
void Fire() {
    flag = 0;
    while (flag == 0) {
        //Read the analog value of the flame sensor
        flame_valL = analogRead(flame_L);
        flame_valR = analogRead(flame_R);
        if (flame_valL <= 700 || flame_valR <= 700) {
            Car_Stop();
            fan_begin();
        } else {

```

(continues on next page)

(continued from previous page)

```

fan_stop();
L_val = digitalRead(L_pin); //Read the value of the left sensor
M_val = digitalRead(M_pin); //Read the value of the left sensor
R_val = digitalRead(R_pin); //Read the value of the right sensor

if (M_val == 1) { //the middle one detects black lines
    if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but
↳not on the right, turn left
        Car_left();
    }
    else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↳not on the left, turn right
        Car_right();
    }
    else { //go front
        Car_front();
    }
}
else { //the middle one detects black lines
    if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but
↳not on the right, turn left
        Car_left();
    }
    else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↳not on the left, turn right
        Car_right();
    }
    else { //otherwise stop
        Car_Stop();
    }
}
}
if (Serial.available())
{
    ble_val = Serial.read();
    if (ble_val == 'S') {
        flag = 1;
        Car_Stop();
    }
}
}

#endif

/*****line tracking*****/
void Tracking() {
    flag = 0;
    while (flag == 0) {
        L_val = digitalRead(L_pin); //Read the value of the left sensor
        M_val = digitalRead(M_pin); //Read the value of the intermediate sensor
        R_val = digitalRead(R_pin); //Read the value of the sensor on the right

```

(continues on next page)

(continued from previous page)

```

    if (M_val == 1) { //the middle one detects black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↪on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //go front
            Car_front();
        }
    }
    else { //the middle sensor doesn't detect black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↪on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //otherwise stop
            Car_Stop();
        }
    }
    if (Serial.available())
    {
        ble_val = Serial.read();
        if (ble_val == 'S') {
            flag = 1;
            Car_Stop();
        }
    }
}

/*****Confinement*****/
void Confinement() {
    flag = 0;
    while (flag == 0) {
        L_val = digitalRead(L_pin); //Read the value of the left sensor
        M_val = digitalRead(M_pin); //Read the value of the intermediate sensor
        R_val = digitalRead(R_pin); //Read the value of the sensor on the right
        if ( L_val == 0 && M_val == 0 && R_val == 0 ) { //Go front when no black lines are
↪detected      Car_front();
        }
        else { //
            Car_back();
            delay(700);
            Car_left();
            delay(800);

```

(continues on next page)

(continued from previous page)

```

    }
    if (Serial.available())
    {
        ble_val = Serial.read();
        if (ble_val == 'S') {
            flag = 1;
            Car_Stop();
        }
    }
}

//*****dot matrix*****/
//this function is used for the display of dot matrix
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //use the function to start transmitting data
    IIC_send(0xc0); //select an address
    for (int i = 0; i < 16; i++) //image data have 16 characters
    {
        IIC_send(matrix_value[i]); //data to transmit pictures
    }
    IIC_end(); //end the data transmission of pictures
    IIC_start();
    IIC_send(0x8A); //show control and select pulse width 4/16
    IIC_end();
}

//the condition that data starts transmitting
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

//transmit data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <= 1) //each character has 8 digits, which is
    ↪ detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
        delayMicroseconds(3);
    }
}

```

(continues on next page)

(continued from previous page)

```

    digitalWrite(SCL_Pin, HIGH); //pull up the clock pin SCL_Pin to end the transmission
    ↪ of data
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of
    ↪ SDA
    }
}

//the sign that transmission of data ends
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, HIGH);
    delayMicroseconds(3);
}

/******motor runs******/
void Car_back() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, speeds_R);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, speeds_L);
    matrix_display(back); //show the image of going back
}

void Car_front() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 255 - speeds_R);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 255 - speeds_L);
    matrix_display(front); //show the image of going front
}

void Car_left() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 255 - speeds_R);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, speeds_L);
    matrix_display(left); //show the image of turning left
}

void Car_right() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, speeds_R);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 255 - speeds_L);
    matrix_display(right); //show the image of turning right
}

```

(continues on next page)

(continued from previous page)

```
void Car_Stop() {  
  digitalWrite(MR_Ctrl, LOW);  
  analogWrite(MR_PWM, 0);  
  digitalWrite(ML_Ctrl, LOW);  
  analogWrite(ML_PWM, 0);  
  matrix_display(STOP01); //show the stop image  
}
```

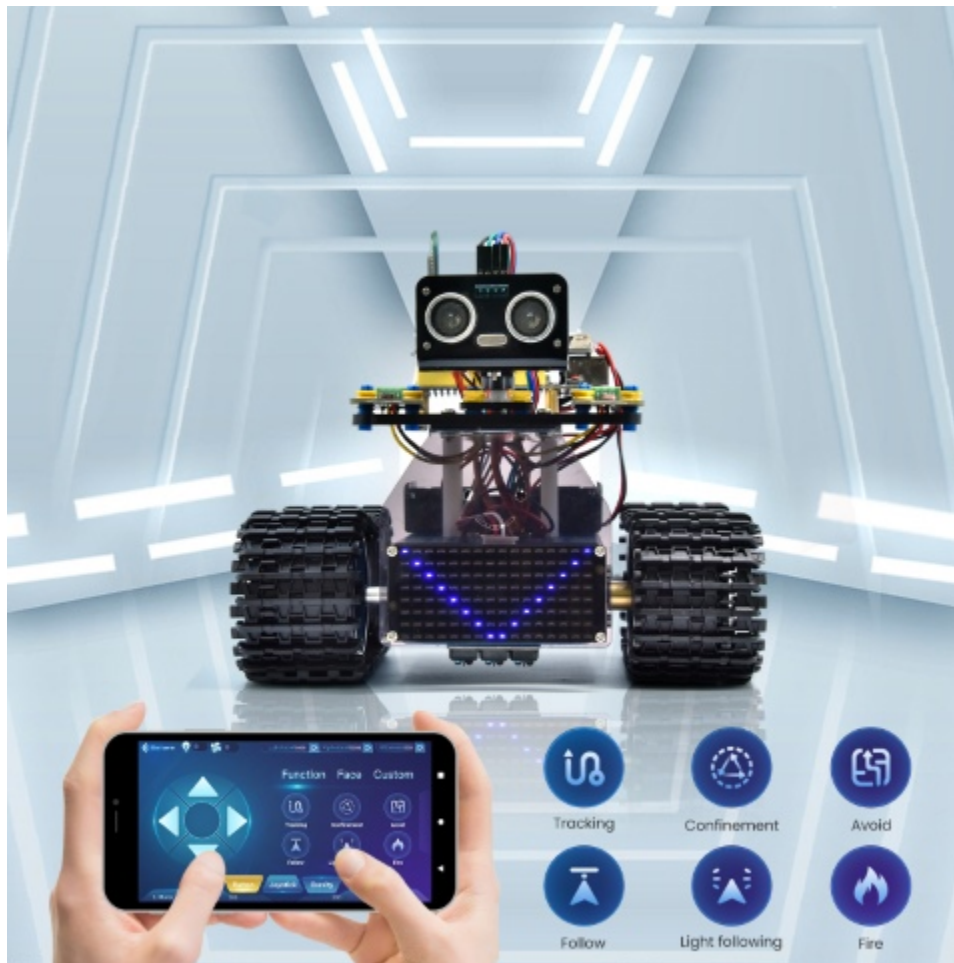
(6)Test Result:

Before uploading the program code, the Bluetooth module needs to be removed, otherwise the code upload will fail.

After uploading the code successfully, open the positioning, and then connect the Bluetooth module.

After uploading the code successfully, plug in the Bluetooth module, after power-on, after the mobile APP is connected to the Bluetooth successfully, we can use the mobile APP to control the tank robot

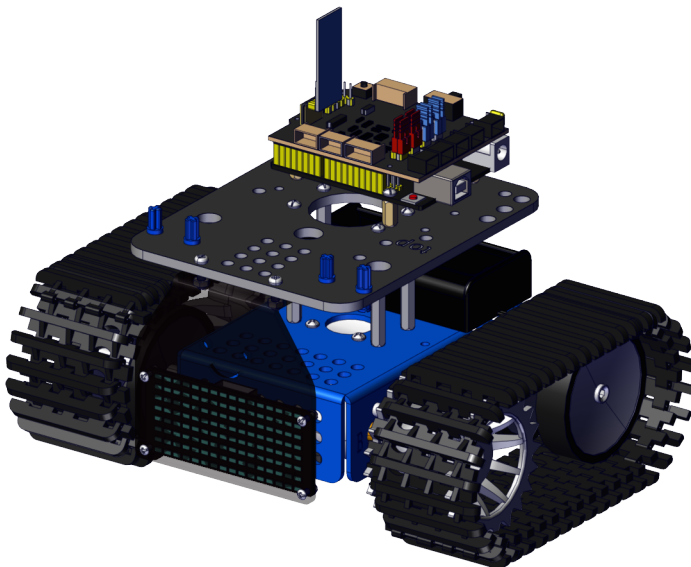
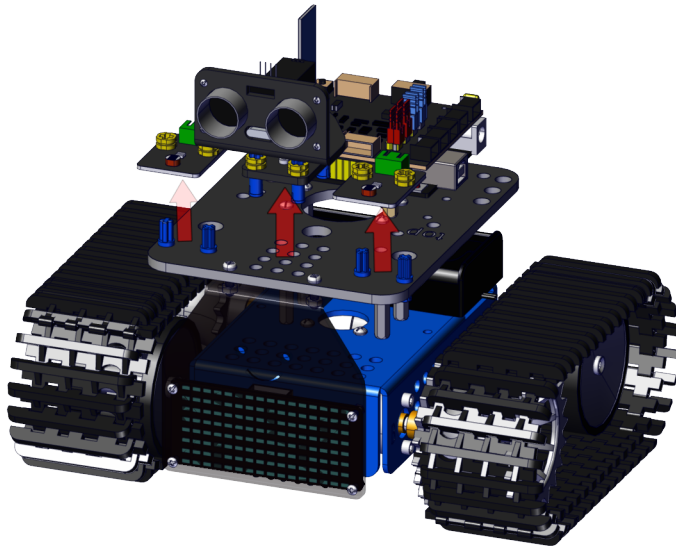
You can also control the robot with the remote control



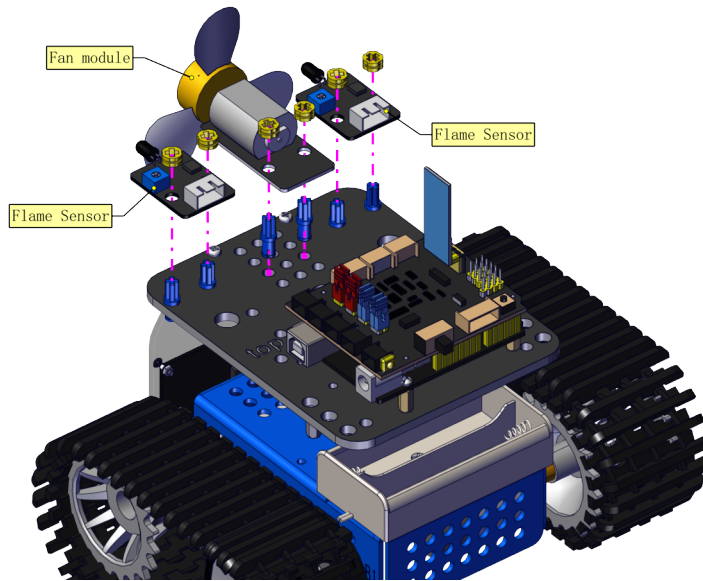
6.4 4. Experiment Extension

6.4.1 Assemble Fire Extinguishing Robot

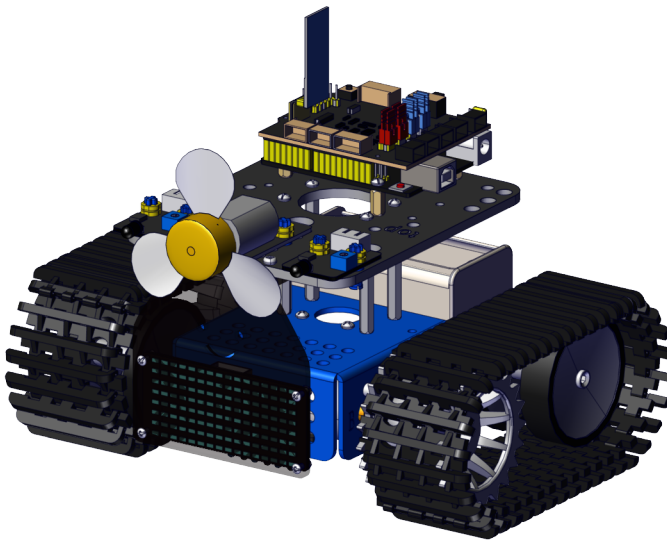
Remove the ultrasonic sensor and two photoresistors



Put on a fan module and two flame sensors

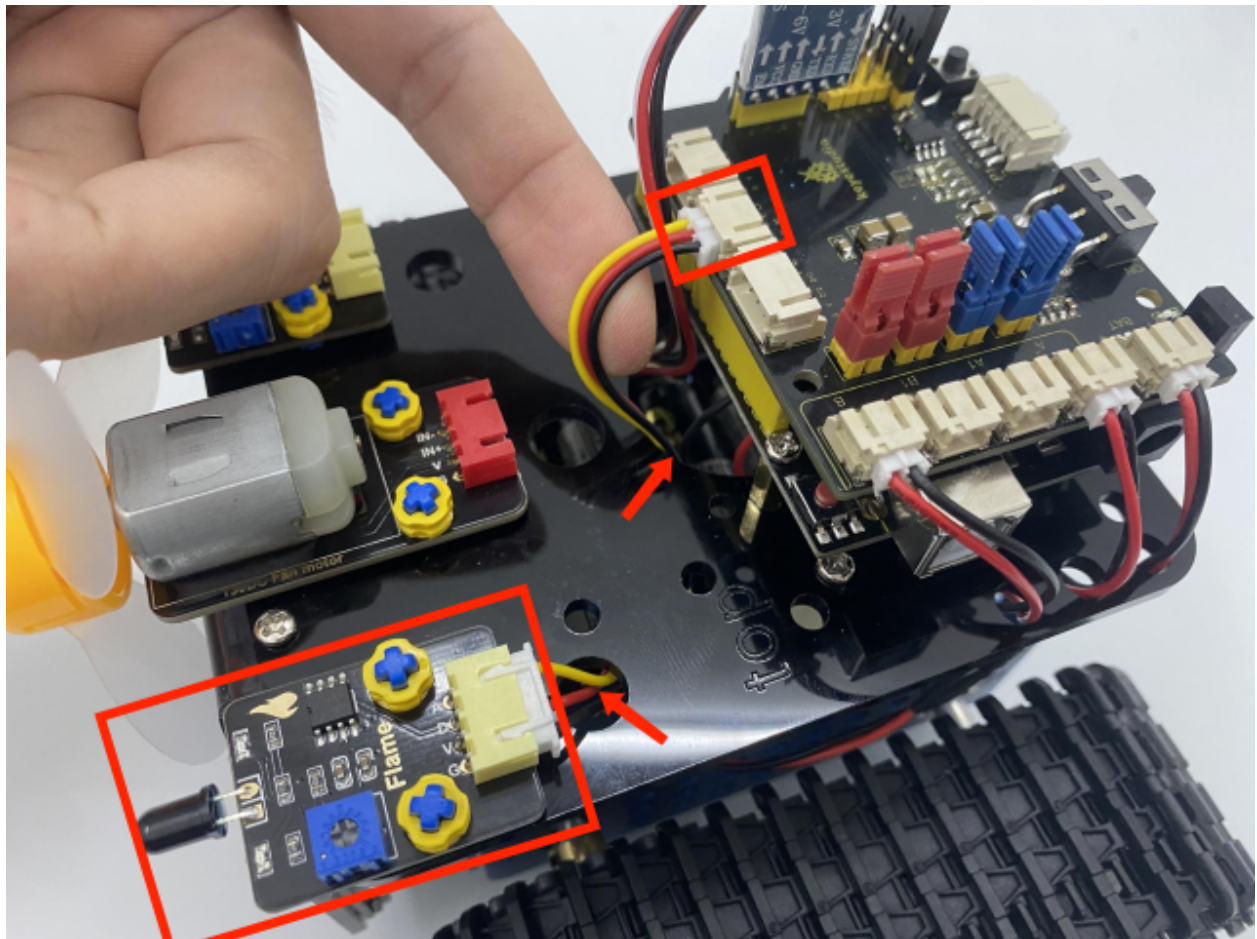


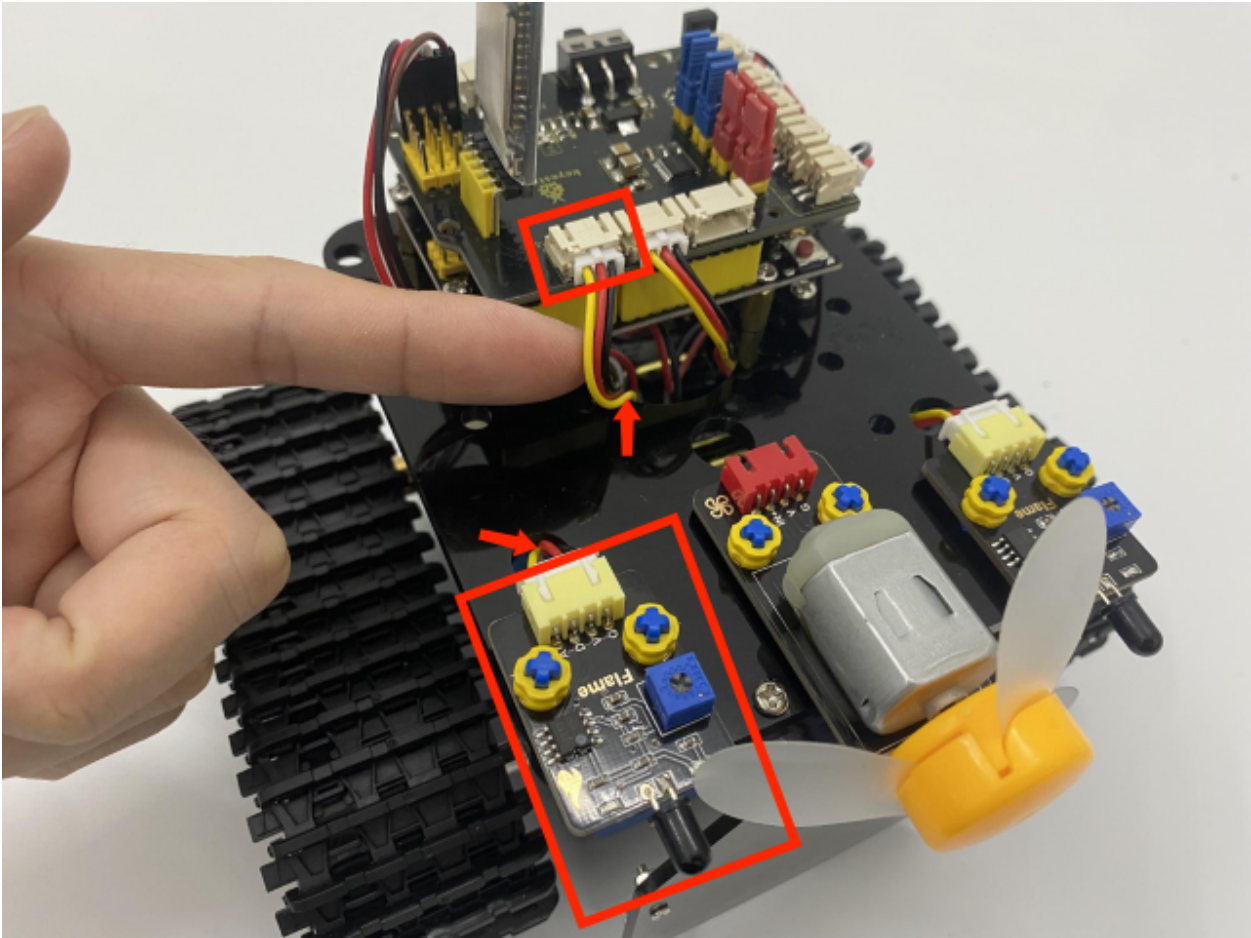
You can make the fan module install further if the fan module and flame sensors interfere



Wire up

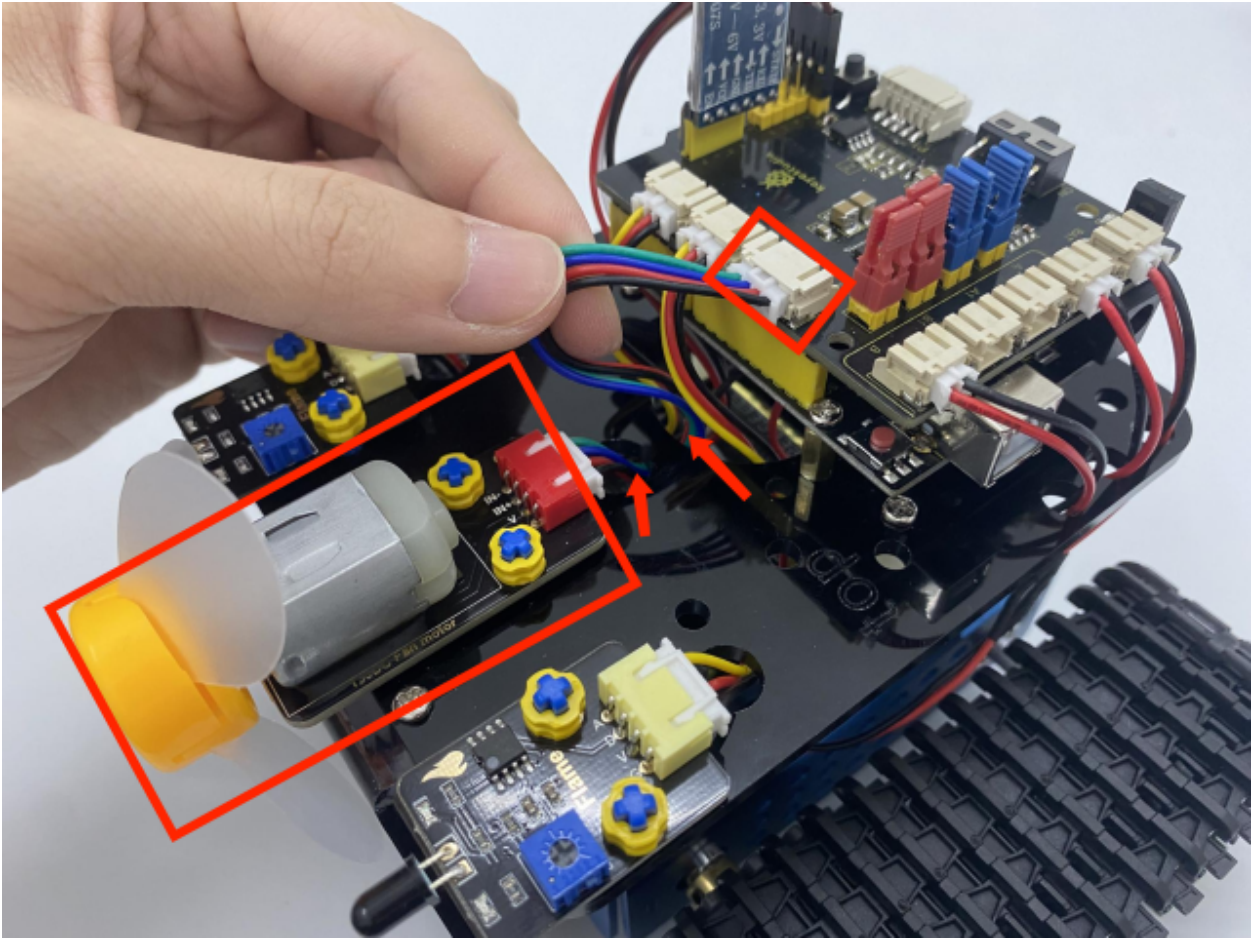
Wire up the two flame sensors





Left	Keyestudio 8833 Board	Right	Keyestudio 8833 Board
G	G	G	G
V	V	V	V
A	A1	A	A2

Wire up the fan module



DC130 Motor	Keyestudio 8833 Board
G	G
V	V
IN+	D12
IN-	D13

We adopt a model 18650 lithium battery with a pointed positive pole, whose power and capacity are not required.



6.4.2 Project 20: Flame Sensor



(1)Description

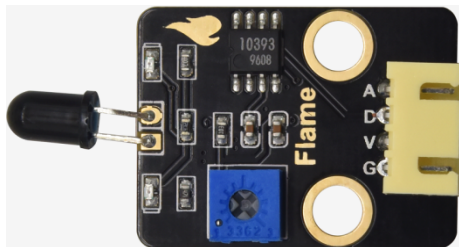
The flame sensor uses IR receiving tube to detect flames, converts the brightness of the flame into signals with high and low levels, input them into the central processor. The corresponding program processing. In both flames close to and without flames, the voltage value of the analog port is varied.

If there is no flame, the analog port is about 0.3V; when there is a flame, the analog port is 1.0V. The closer the flame is, the more the voltage value is. It can be used to detect the fire source or make a smart robot.

Note the probe of flame sensors only bears the temperature between -25 °C and 85°C

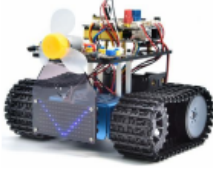

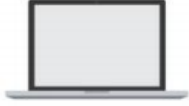


In the process of use, pay attention to keep the flame sensor in certain distance to avoid getting damaged.

(2)Parameters

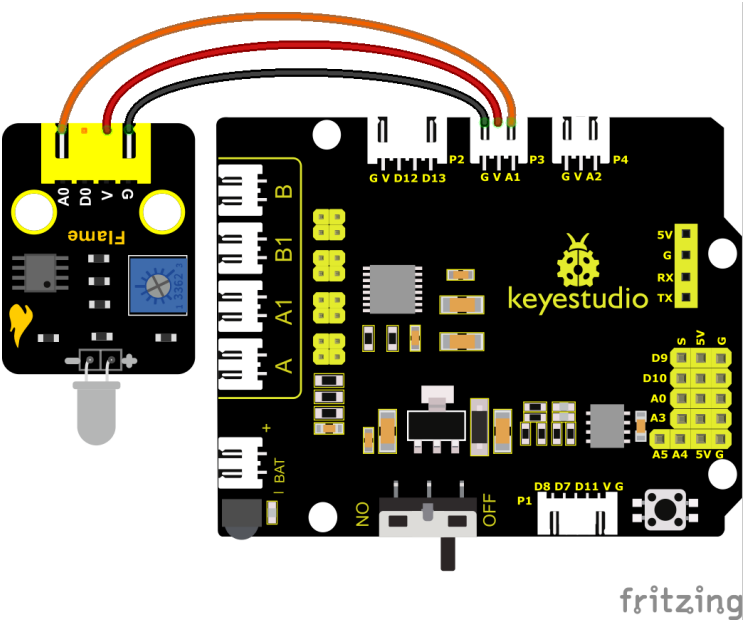


- Working voltage: 3.3V-5V (DC)
- Current: 100mA
- Maximum power: 0.5W
- Work temperature: -10 ° C to +50 degrees Celsius
- Sensor size: 31.6mmx23.7mm
- Interface: 4pin turn 3PIN interface
- Output signal: analog signals A0, A1

(3)Components Needed:

Robot tank without BT module	USB Cable*1	Computer*1	18650 Battery*2	Lighter*1
				

(4)Connection Diagram:



Flame sensors are connected to A1 and A2. When we remove ultrasonic sensors and photoresistors, then add flame sensors and fan modules. The fire extinguishing robot is created.

Note: 1This experiment requires the use of a fire source. Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. 2The flame sensor is not fireproof, please do not burn it directly with flame.

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```
/*  
  
Keyestudio Mini Tank Robot V3 (Popular Edition)  
  
lesson 20.1  
  
flame sensor  
  
http://www.keyestudio.com  
  
*/  
  
int flame = A1; //Define the flame pin as analog pin A1  
int val = 0; //Define digital variables  
  
void setup() {  
    pinMode(flame, INPUT); //Define the buzzer as an input source  
    Serial.begin(9600); //Set the baud rate to 9600  
}  
  
void loop() {  
    val = analogRead(flame); //Read the analog value of the flame sensor  
    Serial.println(val); //Output analog value and print it  
    delay(100); //Delay in 100ms  
}
```

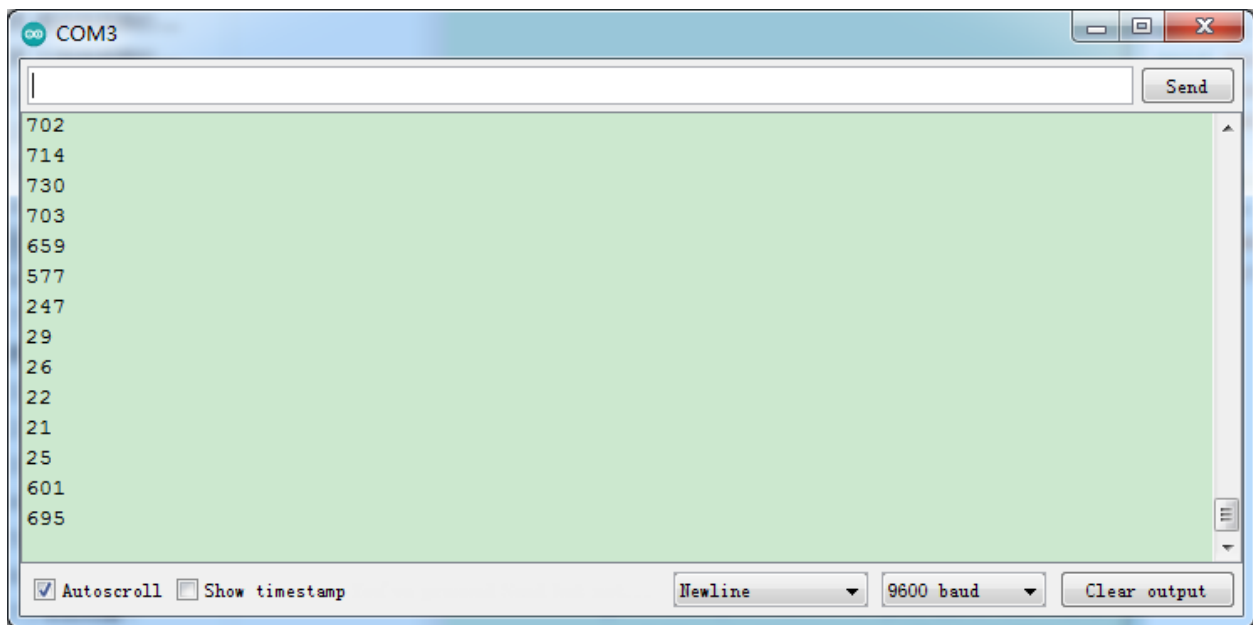
(6)Test Result

Wire up components, burn the code, open the serial monitor and set the baud rate to 9600.

You can view the simulation value of flame sensor.

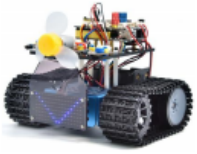

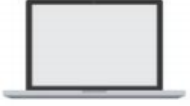




The closer the flame, the smaller the simulation value.

Adjust the potentiometer on the module to maintain D1 at the critical point. When the sensor does not detect flame, the D1 will be off, but if the sensor detects flame, the D1 will be on.



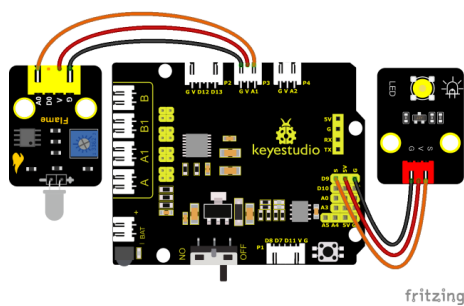
Note: Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. The flame sensor is not fireproof, please do not burn it directly with flame.

(7)Extension Practice:

Robot without BT module	USB Cable*1	Computer*1	18650 Battery*2	Lighter*1
				
Yellow LED Module*1	3P-3P XH2.54 to 2.54 Wire			
				

Note: 1This experiment requires the use of a fire source. Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. 2The flame sensor is not fireproof, please do not burn it directly with flame.

Next, connect an LED to pin 9 and we can control it by a flame sensor, as shown below;



Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)

lesson 20.2

flame sensor

http://www.keyestudio.com

*/

int flame = A1; //Define the flame pin as analog pin A1

```

(continues on next page)

(continued from previous page)

```

int LED = 9; //Define the LED as digital port 9
int val = 0; //Define digital variables

void setup() {
  pinMode(flame, INPUT); //Define the buzzer as an input source
  pinMode(LED, OUTPUT); //Set LED to output mode
  Serial.begin(9600); //Set the baud rate to 9600
}

void loop() {
  val = analogRead(flame); //Read the analog value of the flame sensor
  Serial.println(val); //Output analog value and print it
  if (val < 300) { //When analog value is less than 300, LED is on
    digitalWrite(LED, HIGH); //LED is on
  } else {
    digitalWrite(LED, LOW); //LED is off
  }
  delay(50); //Delay in 50ms
}

```

(8) Test Results

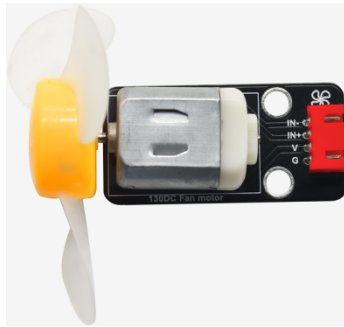
You can use the flame of a lighter near the left flame sensor. When the flame sensor detects a flame, the LED module will light up as an alarm.



Note: Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. The flame sensor is not fireproof, please do not burn it directly with flame.

6.4.3 Project 21: Fan

(1)Description



This fan module uses a HR1124S motor-controlling chip, a single-channel H-bridge driver chip containing a low-conductivity resistance PMOS and NMOS power tubes. The low-conducting resistance can ease the power consumption, contributing to the safe work of the chip for longer time.

In addition, its low standby current and low static working current makes itself apply to toys. We can control the rotation direction and speed of the fan by outputting IN + and IN- signals and PWM signals.

(2)Specification:

Working voltage: 5V

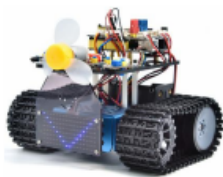

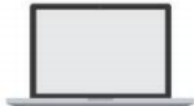

Current: 200MA

Maximum power: 2W

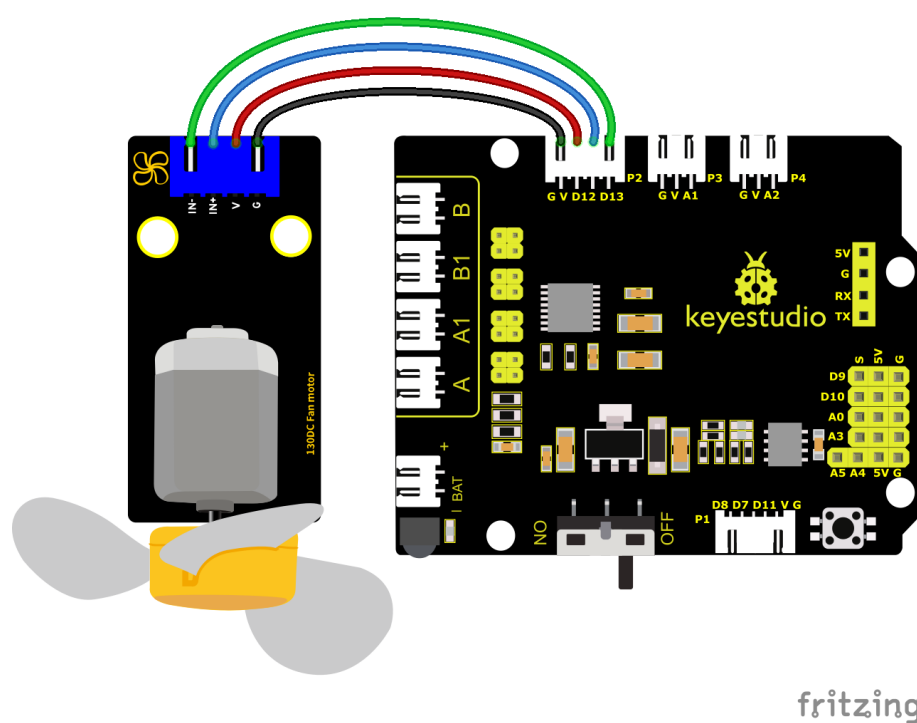
Operating temperature: -10 degrees Celsius to +50 degrees Celsius

Size: 47.6MM *23.8MM

(3)Components Required:

Robot tank without BT module	USB Cable*1	Computer*1	18650 Battery*2
			

The fan module needs driving by large current; therefore, we install a battery holder.

(4)Connection Diagram:

The pin GND, VCC, IN+ and IN- of the fan module are connected to pin G, V, D12 and D13 of the shield.

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 21.1
130 motor
http://www.keyestudio.com
*/

int INA = 12;
int INB = 13;

void setup() {
  pinMode(INA, OUTPUT); //Set digital port INA as output
  pinMode(INB, OUTPUT); //Set digital port INA as output
}

void loop() {
  //Set the fan to rotate anticlockwise for 3s
  digitalWrite(INA, LOW);
  digitalWrite(INB, HIGH);

```

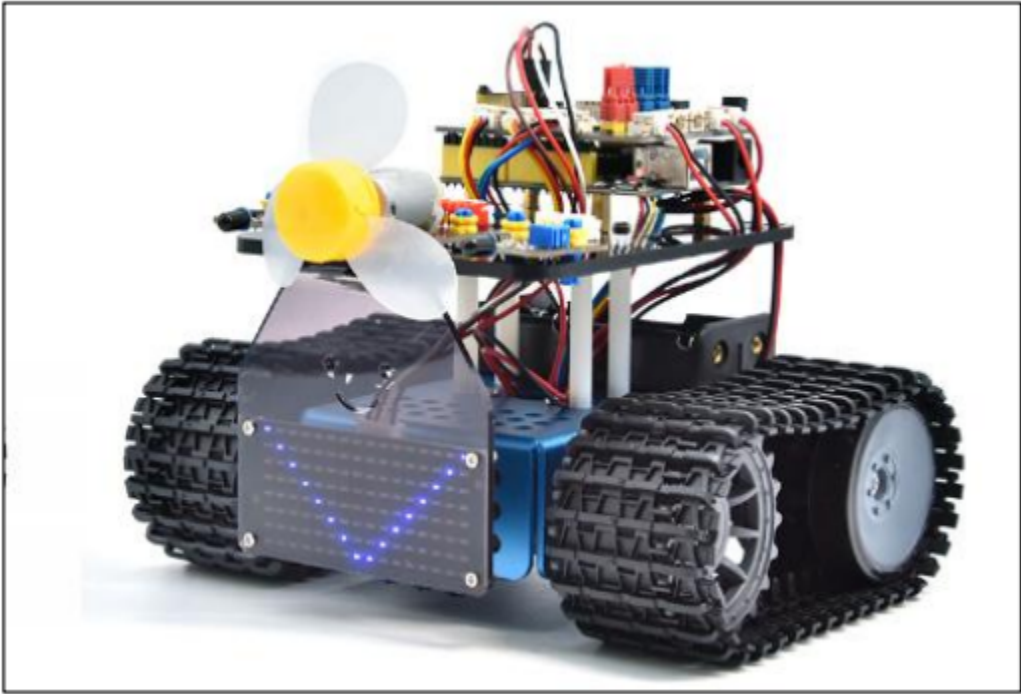
(continues on next page)

(continued from previous page)

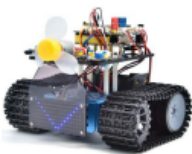

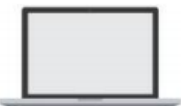


```
delay(3000);  
//Set the fan to stop for 1s  
digitalWrite(INA, LOW);  
digitalWrite(INB, LOW);  
delay(1000);  
//Set the fan to rotate clockwise for 3s  
digitalWrite(INA, HIGH);  
digitalWrite(INB, LOW);  
delay(3000);  
}
```

(7)Test Results

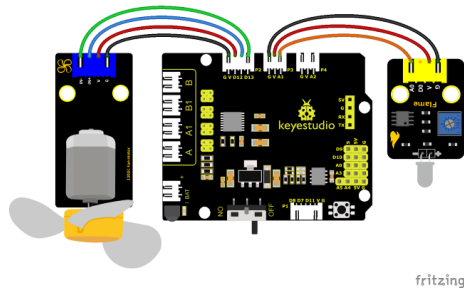
Upload code, wire up components and plug in power. The small fan will turn anticlockwise for 3000ms, stop for 1000ms, and clockwise for 300ms.



(8)Extension Practice:

Robot without BT module	USB Cable*1	Computer*1	18650 Battery*2	Lighter*1
				

We have understood the working principle of the flame sensor. Next, hook up a flame sensor in the circuit , as shown below. Then control the fan to blew out fire with the flame sensor.



(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
Keyestudio Mini Tank Robot V3 (Popular Edition)
lesson 21.2
130 motor
http://www.keyestudio.com
*/

int INA = 12;
int INB = 13;
int flame = A1; //Define the flame pin as analog pin A1
int val = 0; //Define digital variables

void setup() {
  pinMode(INA, OUTPUT); //Set digital port INA as output
  pinMode(INB, OUTPUT); //Set digital port INB as output
  pinMode(flame, INPUT); //Define the buzzer as an input source
}

void loop() {
  val = analogRead(flame); //Read the analog value of the flame sensor
  if (val <= 700) { //When analog value 700, LED is on
    //Turn on the fan when flame is detected
    digitalWrite(INA, LOW);
    digitalWrite(INB, HIGH);
  } else {
    //Otherwise it stops operating
    digitalWrite(INA, LOW);
    digitalWrite(INB, LOW);
  }
}

```

6.4.4 Project 22: Fire Extinguishing Tank

(1)Description:





The line-tracking function of the smart tank has been explained in the previous project. And in this project we use the flame sensor to make a fire extinguishing robot.

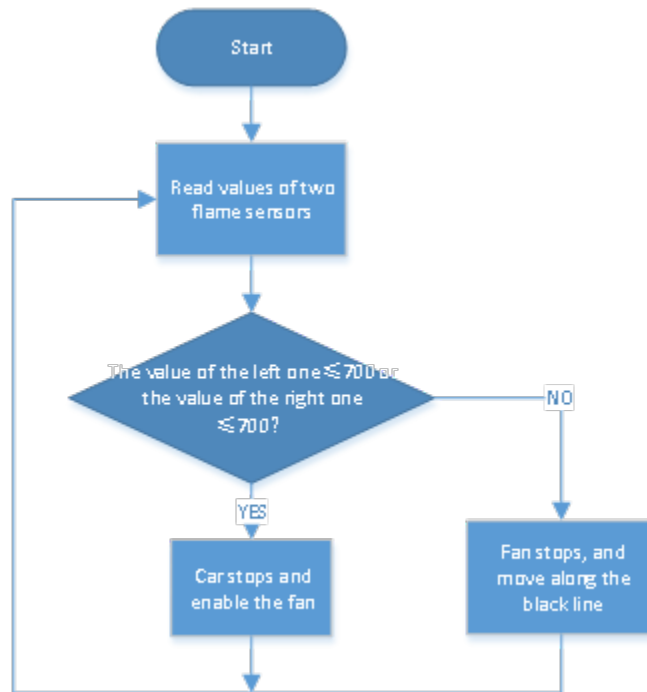
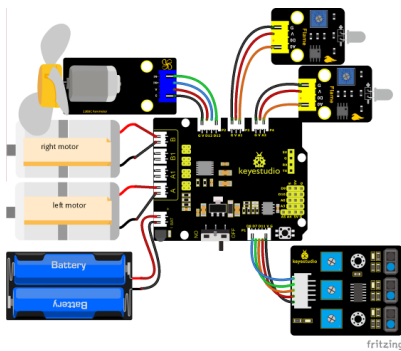
When the car encounters flames, the motor of the fan will rotate to blow out the fire. Of course, we need to replace the ultrasonic sensor and two photoresistors with a fan module and flame sensors first.

The specific logic of the line-tracking smart car is shown in the table below:

Flame sensors		Status
Left	Right	
≤ 700	≤ 700	Car stops, fan starts rotating to blow up flame
≥ 700	≥ 700	Car stops, fan starts rotating to blow up flame
≥ 700	≥ 700	Car stops, fan starts rotating to blow up flame
> 700	> 700	Fan stops, car moves

(2)Components Required:

Robot tank without BT module	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:**(4)Connection Diagram:****(5)Test Code:**

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keystudio Mini Tank Robot V3 (Popular Edition)
  lesson 22
  Fire extinguishing tank
  http://www.keystudio.com
*/
  
```

(continues on next page)

(continued from previous page)

```

int flame_L = A1; //Define the flame interface on the left as the analog pin A1
int flame_R = A2; //Define the flame interface on the right as the analog pin A2
//The wiring of line tracking sensor
#define L_pin 11 //left
#define M_pin 7 //middle
#define R_pin 8 //right
//The pin of the servo 130
int INA = 12;
int INB = 13;
#define ML_Ctrl 4 //Define the direction control pin of the left motor
#define ML_PWM 6 //Define the PWM control pin of the left motor
#define MR_Ctrl 2 //Define the direction control pin of the right motor
#define MR_PWM 5 //Define the PWM control pin of the right motor
int L_val, M_val, R_val, flame_valL, flame_valR;

void setup()
{
  Serial.begin(9600);
  //Set all pins of the line tracking sensor as input mode
  pinMode(L_pin, INPUT);
  pinMode(M_pin, INPUT);
  pinMode(R_pin, INPUT);
  //Define the flame as INPUT
  pinMode(flame_L, INPUT);
  pinMode(flame_R, INPUT);
  //Define the motor as OUTPUT
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
  pinMode(INA, OUTPUT); //Set digital port INA as OUTPUT
  pinMode(INB, OUTPUT); //Set digital port INB as OUTPUT
}

void loop () {
  //Read the analog value of the flame sensors
  flame_valL = analogRead(flame_L);
  flame_valR = analogRead(flame_R);
  Serial.print(flame_valL);
  Serial.print(" ");
  Serial.print(flame_valR);
  Serial.println(" ");
  // delay(500);
  if (flame_valL <= 700 || flame_valR <= 700) {
    Car_Stop();
    fan_begin();
  } else {
    fan_stop();
    L_val = digitalRead(L_pin); //Read the value of the left sensor
    M_val = digitalRead(M_pin); //Read the value of the middle sensor
    R_val = digitalRead(R_pin); //Read the value of the right sensor
  }
}

```

(continues on next page)

(continued from previous page)

```

    if (M_val == 1) { //the middle one detects black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↪on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //otherwise, go front
            Car_front();
        }
    }
    else { //The middle one doesn't detect black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↪on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //otherwise, stop
            Car_Stop();
        }
    }
}
}

void fan_stop() {
    //stop rotating
    digitalWrite(INA, LOW);
    digitalWrite(INB, LOW);
}

void fan_begin() {
    //fan rotates
    digitalWrite(INA, LOW);
    digitalWrite(INB, HIGH);
}

void Car_front()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 150);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 150);
}

void Car_back()
{
    digitalWrite(MR_Ctrl, LOW);

```

(continues on next page)

(continued from previous page)

```
    analogWrite(MR_PWM, 100);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 100);
}
void Car_left()
{
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 150);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 100);
}
void Car_right()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 100);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 150);
}
void Car_Stop()
{
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 100);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 100);

    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
}
```

(6)Test Result:

After uploading the test code successfully and powering it up, the smart car puts out the fire when it detects flame and continues moving along the black line.



Note: Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. The flame sensor is not fireproof, please do

not burn it directly with flame.

6.4.5 Project 23: Fire Extinguishing Robot Multiple Functions

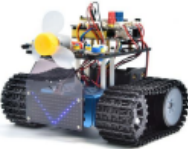

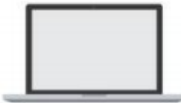


(1)Description:

The smart car has performed a single function in every previous project.

Can it display multiply functions at a time ? Positive.

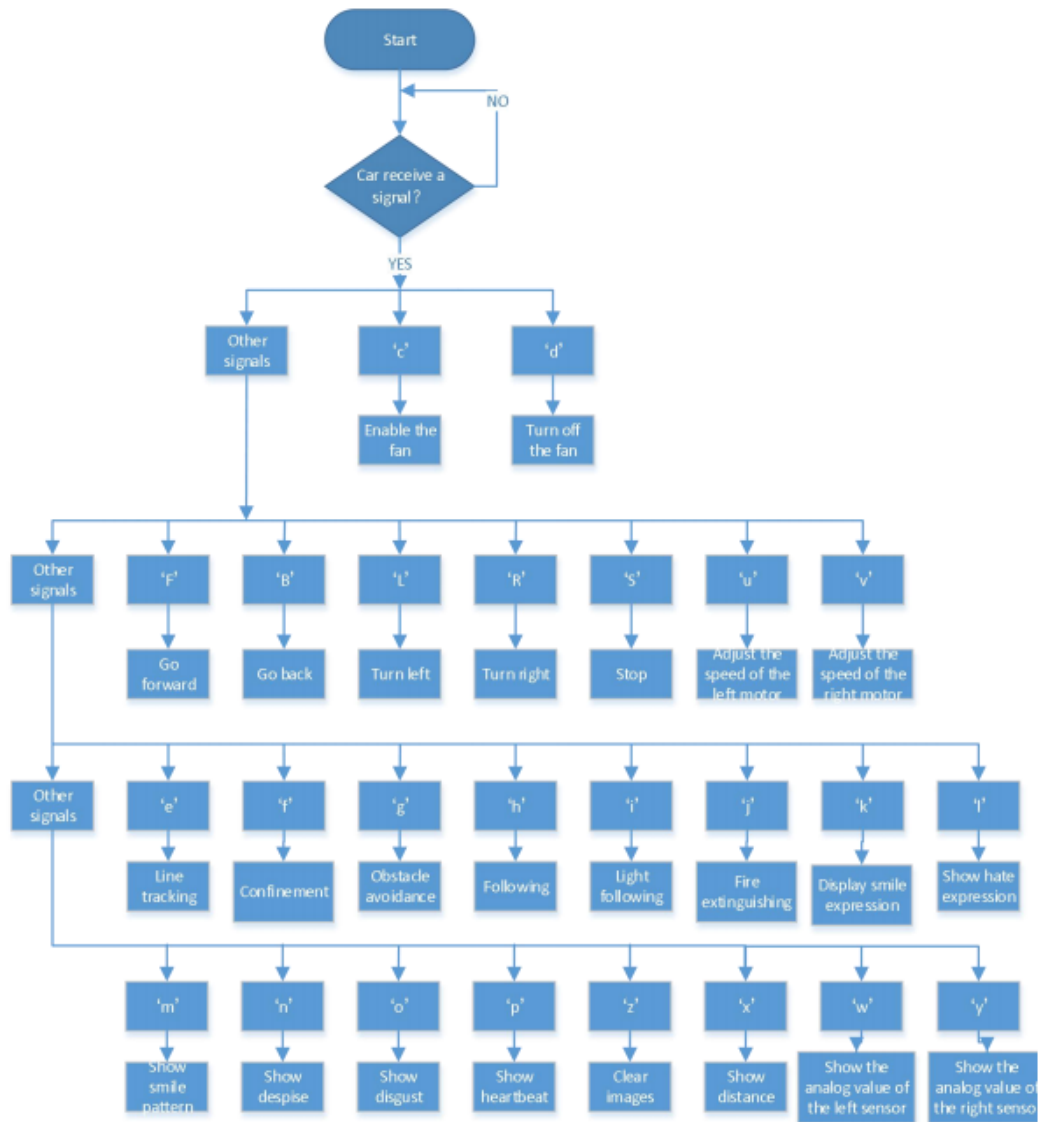
In this last big project, we intend to use a complete code to control the smart car to show off all functions mentioned in previous projects. We use the keys on the Bluetooth APP to automatically switch various functions, quite simple and convenient.

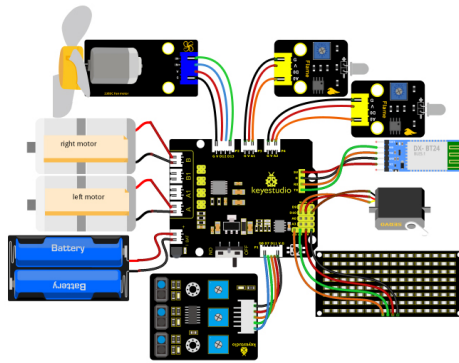
(2)Components Required:

Robot without BT module	USB Cable*1	Computer*1	18650 Battery*2	Bluetooth module*1
				

Please referto Project 16 to install and configure Bluetooth APP

(3)Flow Diagram:



(4)Connection Diagram:

1. GND, VCC, SDA and SCL of the 8x16 board are connected to GND), +VCC), A4 and A5 of the expansion board.
2. VCC, IN+, IN- and Gnd of the ultrasonic sensor are connected to 5V(V), 12(S), 13(S) and Gnd(G)
3. The brown wire, red wire and orange wire of the servo are connected to Gnd(G), 5v(V) and D10.
4. RXD, TXD, GND and VCC of the BT module are connected to TX, RX, GGND) and 5VVCC. STATE and BRK don't need to be interfaced.
5. The pin "G", "V" and S of the left Flame sensors are connected to G (GND), V (VCC), A1 respectively; The right Flame sensors is connected to the G (GND), V (VCC), and A2 respectively.
6. The distal port of the line tracking sensor is 11, 7 and 8

(5)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

```

/*
  Keyestudio Mini Tank Robot V3 (Popular Edition)
  lesson 23
  Fire Extinguishing Robot Multiple Functions
  http://www.keyestudio.com
*/
#include <IRremote.h>
IRrecv irrecv(3); //
decode_results results;
long ir_rec; //used to save the IR value

/*****/
#define USE_FAN_FUNCTION 1

//Array, used to save data of images, can be calculated by yourself or gotten from
↳modulus tool
unsigned char start01[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, 0x80, 0x40,
↳0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
unsigned char STOP01[] = {0x2E, 0x2A, 0x3A, 0x00, 0x02, 0x3E, 0x02, 0x00, 0x3E, 0x22,
↳0x3E, 0x00, 0x3E, 0x0A, 0x0E, 0x00};

```

(continues on next page)

(continued from previous page)

```

unsigned char front[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x12, 0x09, 0x12, 0x24,
↪ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char back[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x48, 0x90, 0x48, 0x24, 0x00,
↪ 0x00, 0x00, 0x00, 0x00, 0x00};
unsigned char left[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x28, 0x10, 0x44, 0x28,
↪ 0x10, 0x44, 0x28, 0x10, 0x00};
unsigned char right[] = {0x00, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44, 0x10, 0x28, 0x44,
↪ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

unsigned char Smile[] = {0x00, 0x00, 0x1c, 0x02, 0x02, 0x02, 0x5c, 0x40, 0x40, 0x5c,
↪ 0x02, 0x02, 0x02, 0x1c, 0x00, 0x00};
unsigned char Disgust[] = {0x00, 0x00, 0x02, 0x02, 0x02, 0x12, 0x08, 0x04, 0x08, 0x12,
↪ 0x22, 0x02, 0x02, 0x00, 0x00, 0x00};
unsigned char Happy[] = {0x02, 0x02, 0x02, 0x02, 0x08, 0x18, 0x28, 0x48, 0x28, 0x18,
↪ 0x08, 0x02, 0x02, 0x02, 0x02, 0x00};
unsigned char Squint[] = {0x00, 0x00, 0x00, 0x41, 0x22, 0x14, 0x48, 0x40, 0x40, 0x48,
↪ 0x14, 0x22, 0x41, 0x00, 0x00, 0x00};
unsigned char Despise[] = {0x00, 0x00, 0x06, 0x04, 0x04, 0x04, 0x24, 0x20, 0x20, 0x26,
↪ 0x04, 0x04, 0x04, 0x04, 0x00, 0x00};
unsigned char Heart[] = {0x00, 0x00, 0x0C, 0x1E, 0x3F, 0x7F, 0xFE, 0xFC, 0xFE, 0x7F,
↪ 0x3F, 0x1E, 0x0C, 0x00, 0x00, 0x00};

unsigned char clear[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
↪ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

#define SCL_Pin A5 //set the pin of clock to A5
#define SDA_Pin A4 //set the data pin to A4

#define ML_Ctrl 4 //define the direction control pin of the left motor as 4
#define ML_PWM 6 //define the PWM control pin of the left motor
#define MR_Ctrl 2 //define the direction control pin of the right sensor
#define MR_PWM 5 //define the PWM control pin of the right motor
char ble_val; //used to save the Bluetooth value
byte speeds_L = 200; //the initial speed of the left motor is 200
byte speeds_R = 200; //the initial speed of the right motor is 200
String speeds_l, speeds_r; //receive PWM characters and convert them into PWM value

//wire up the line tracking sensor
#define L_pin 11 //left
#define M_pin 7 //middle
#define R_pin 8 //right
int L_val, M_val, R_val;

#if USE_FAN_FUNCTION /****use fan*****/
int flame_L = A1; //define the analog port of the left flame sensor to A1
int flame_R = A2; //define the analog port of the right flame sensor to A2
int flame_valL, flame_valR;

//the pin of 130 motor
int INA = 12;
int INB = 13;

```

(continues on next page)

(continued from previous page)

```

#else /****use the ultrasonic sensor*****/
#define servoPin 10 //servo Pin
#define light_L_Pin A1 //define the pin of the left photoresistor
#define light_R_Pin A2 //define the pin of the right photoresistor
int left_light;
int right_light;

#define Trig 12
#define Echo 13
float distance;//Store the distance values detected by ultrasonic for following

//Store the distance values detected by ultrasonic for obstacle avoidance
int a;
int a1;
int a2;

#endif

bool flag; //flag invariable, used to enter and exit a mode

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn(); //Initialize the library of the IR remote

  pinMode(SCL_Pin, OUTPUT);
  pinMode(SDA_Pin, OUTPUT);

  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  pinMode(L_pin, INPUT); //define the pins of sensors to INPUT
  pinMode(M_pin, INPUT);
  pinMode(R_pin, INPUT);

  matrix_display(clear); //clear screen
  matrix_display(start01); //show start

#ifdef USE_FAN_FUNCTION /****use the fan*****/
  pinMode(INA, OUTPUT); //set INA to OUTPUT
  pinMode(INB, OUTPUT); //set INB to OUTPUT

  //define inputs of the flame sensor
  pinMode(flame_L, INPUT);
  pinMode(flame_R, INPUT);
#else /****use the ultrasonic sensor*****/
  pinMode(servoPin, OUTPUT);
  pinMode(light_L_Pin, INPUT);
  pinMode(light_R_Pin, INPUT);

  pinMode(Trig, OUTPUT);

```

(continues on next page)

(continued from previous page)

```

pinMode(Echo, INPUT);
procedure(90); //set the angle of the servo to 90°
#endif
}

void loop() {
  if (Serial.available()) //if there is data in the serial buffer
  {
    ble_val = Serial.read();
    Serial.println(ble_val);
    switch (ble_val) {
      case 'F': Car_front(); break; //the command to go front

      case 'B': Car_back(); break; //the command to go back

      case 'L': Car_left(); break; //the command to turn left

      case 'R': Car_right(); break; //the command to turn right

      case 'S': Car_Stop(); break; //stop

      case 'e': Tracking(); break; //enter the line tracking mode

      case 'f': Confinement(); break; //enter the confinement mode

      #if USE_FAN_FUNCTION/****use fan*****/
      case 'j': Fire(); break; //enable extinguishing fire mode

      case 'c': fan_begin(); break; //enable the fan

      case 'd': fan_stop(); break; //turn off the fan

      #else/****use the ultrasonic sensor*****/
      case 'g': Avoid(); break; //enter obstacle avoidance mode

      case 'h': Follow(); break; //enter light following mode

      case 'i': Light_following(); break; //enter light following mode
    #endif
    case 'u':
      speeds_l = Serial.readStringUntil('#');
      speeds_L = String(speeds_l).toInt();
      break; //start by receiving u, end by receiving characters # and convert into
      ↪ the integer

    case 'v':
      speeds_r = Serial.readStringUntil('#');
      speeds_R = String(speeds_r).toInt();
      break; //start by receiving u, end by receiving characters # and convert into
      ↪ the integer

    case 'k': matrix_display(Smile); break; //show "smile" face
  }
}

```

(continues on next page)

(continued from previous page)

```

    case 'l': matrix_display(Disgust); break; //show "disgust" face
    case 'm': matrix_display(Happy); break; //show "happy" face
    case 'n': matrix_display(Squint); break; //show "Sad" face
    case 'o': matrix_display(Despise); break; //show "despise" face
    case 'p': matrix_display(Heart); break; //show the heartbeat image
    case 'z': matrix_display(clear); break; //clear images

    default: break;
}
}

#if (USE_FAN_FUNCTION != 1)/*the function to not use the fan*****/
//The following three signals are mainly used for cyclic printing
if (ble_val == 'x') {
    distance = checkdistance(); Serial.println(distance);
    delay(50);
} else if (ble_val == 'w') {
    left_light = analogRead(light_L_Pin);
    Serial.println(left_light);
    delay(50);
} else if (ble_val == 'y') {
    right_light = analogRead(light_R_Pin);
    Serial.println(right_light);
    delay(50);
}
#endif

if (irrecv.decode(&results)) { //Receive infrared remote control value
    ir_rec = results.value;
    Serial.println(ir_rec, HEX);
    switch (ir_rec) {
        case 0xFF629D: Car_front(); break; //go front
        case 0xFFA857: Car_back(); break; //go back
        case 0xFF22DD: Car_left(); break; //rotate to left
        case 0xFFC23D: Car_right(); break; //rotate to right
        case 0xFF02FD: Car_Stop(); break; //stop
        default: break;
    }
    irrecv.resume();
}
}

#if (USE_FAN_FUNCTION != 1)/*use the ultrasonic sensor*****/
//Control the ultrasonic sensor
float checkdistance() {
    float distance;
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);

```

(continues on next page)

(continued from previous page)

```

distance = pulseIn(Echo, HIGH) / 58.20; //
delay(10);
return distance;
}

//the function to control the servo
void procedure(int myangle) {
    int pulsewidth;
    pulsewidth = map(myangle, 0, 180, 500, 2000); //Calculate the pulse width value,
    ↪which should be the mapping value from 500 to 2500. Considering the influence of the
    ↪infrared library, 500~2000 is used here.
    for (int i = 0; i < 5; i++) {
        digitalWrite(servoPin, HIGH);
        delayMicroseconds(pulsewidth); //The duration of the high level is the pulse width
        digitalWrite(servoPin, LOW);
        delay((20 - pulsewidth / 1000)); //The period is 20ms, so the low level lasts the
    ↪rest of the time
    }
}

/*****obstacle avoidance*****/
void Avoid()
{
    flag = 0;
    while (flag == 0)
    {
        a = checkdistance(); //the front distance is set to a
        if (a < 20) { //When the distance in front is less than 20cm
            Car_Stop(); //stop
            delay(500); //delay in 500ms
            procedure(180); //servo turns left
            delay(500); //delay in 500ms
            a1 = checkdistance(); //the left distance is set to a1
            delay(100); //read value

            procedure(0); //servo turns right
            delay(500); //delay in 500ms
            a2 = checkdistance(); //the right distance is set to a2
            delay(100); //read value

            procedure(90); //back to 90°
            delay(500);
            if (a1 > a2) { //When the distance on the left is greater than the distance on the
    ↪right
                Car_left(); //the robot turns left
                delay(700); //turn left 700ms
            } else {
                Car_right(); //turn right
                delay(700);
            }
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

else { //if the front distance 20cmrobot goes front
    Car_front(); //go front
}
//receive the Bluetooth value to exit the loop
if (Serial.available())
{
    ble_val = Serial.read();
    if (ble_val == 'S') //receive S
    {
        flag = 1; //Set flag to 1 to exit the loop
        Car_Stop();
    }
}
}
}

/*****following*****/
void Follow() {
    flag = 0;
    while (flag == 0) {
        distance = checkdistance(); //set the distance value to distance
        if (distance >= 20 && distance <= 60) //go front
        {
            Car_front();
        }
        else if (distance > 10 && distance < 20) // stop
        {
            Car_Stop();
        }
        else if (distance <= 10) //go back
        {
            Car_back();
        }
        else //stop
        {
            Car_Stop();
        }
        if (Serial.available())
        {
            ble_val = Serial.read();
            if (ble_val == 'S')
            {
                flag = 1; //exit the loop
                Car_Stop();
            }
        }
    }
}

/*****light following*****/
void Light_following() {
    flag = 0;

```

(continues on next page)

(continued from previous page)

```

while (flag == 0) {
  left_light = analogRead(light_L_Pin);
  right_light = analogRead(light_R_Pin);
  if (left_light > 650 && right_light > 650) //go forward
  {
    Car_front();
  }
  else if (left_light > 650 && right_light <= 650) //turn left
  {
    Car_left();
  }
  else if (left_light <= 650 && right_light > 650) //turn right
  {
    Car_right();
  }
  else //or else, stop
  {
    Car_Stop();
  }
  if (Serial.available())
  {
    ble_val = Serial.read();
    if (ble_val == 'S') {
      flag = 1;
      Car_Stop();
    }
  }
}
}

*/else/****use the fan*****/
*/*****enable the fan*****/
void fan_begin() {
  digitalWrite(INA, LOW);
  digitalWrite(INB, HIGH);
}

*/*****stop fanning*****/
void fan_stop() {
  digitalWrite(INA, LOW);
  digitalWrite(INB, LOW);
}

*/*****extinguish fire*****/
void Fire() {
  flag = 0;
  while (flag == 0) {
    //Read the analog value of the flame sensor
    flame_valL = analogRead(flame_L);
    flame_valR = analogRead(flame_R);
    if (flame_valL <= 700 || flame_valR <= 700) {
      Car_Stop();
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

    fan_begin();
} else {
    fan_stop();
    L_val = digitalRead(L_pin); //Read the value of the left sensor
    M_val = digitalRead(M_pin); //Read the value of the left sensor
    R_val = digitalRead(R_pin); //Read the value of the right sensor

    if (M_val == 1) { //the middle one detects black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but
↪not on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //go front
            Car_front();
        }
    }
    else { //the middle one detects black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but
↪not on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //otherwise stop
            Car_Stop();
        }
    }
}
if (Serial.available())
{
    ble_val = Serial.read();
    if (ble_val == 'S') {
        flag = 1;
        Car_Stop();
    }
}
}

#endif

/*****line tracking*****/
void Tracking() {
    flag = 0;
    while (flag == 0) {
        L_val = digitalRead(L_pin); //Read the value of the left sensor

```

(continues on next page)

(continued from previous page)

```

    M_val = digitalRead(M_pin); //Read the value of the intermediate sensor
    R_val = digitalRead(R_pin); //Read the value of the sensor on the right
    if (M_val == 1) { //the middle one detects black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↪on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //go front
            Car_front();
        }
    }
    else { //the middle sensor doesn't detect black lines
        if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not
↪on the right, turn left
            Car_left();
        }
        else if (L_val == 0 && R_val == 1) { //If a black line is detected on the right,
↪not on the left, turn right
            Car_right();
        }
        else { //otherwise stop
            Car_Stop();
        }
    }
    if (Serial.available())
    {
        ble_val = Serial.read();
        if (ble_val == 'S') {
            flag = 1;
            Car_Stop();
        }
    }
}

/*****Confinement*****/
void Confinement() {
    flag = 0;
    while (flag == 0) {
        L_val = digitalRead(L_pin); //Read the value of the left sensor
        M_val = digitalRead(M_pin); //Read the value of the intermediate sensor
        R_val = digitalRead(R_pin); //Read the value of the sensor on the right
        if ( L_val == 0 && M_val == 0 && R_val == 0 ) { //Go front when no black lines are
↪detected      Car_front();
        }
        else { //
            Car_back();
            delay(700);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    Car_left();
    delay(800);
}
if (Serial.available())
{
    ble_val = Serial.read();
    if (ble_val == 'S') {
        flag = 1;
        Car_Stop();
    }
}
}
}

/*****dot matrix*****/
//this function is used for the display of dot matrix
void matrix_display(unsigned char matrix_value[])
{
    IIC_start(); //use the function to start transmitting data
    IIC_send(0xc0); //select an address
    for (int i = 0; i < 16; i++) //image data have 16 characters
    {
        IIC_send(matrix_value[i]); //data to transmit pictures
    }
    IIC_end(); //end the data transmission of pictures
    IIC_start();
    IIC_send(0x8A); //show control and select pulse width 4/16
    IIC_end();
}

//the condition that data starts transmitting
void IIC_start()
{
    digitalWrite(SDA_Pin, HIGH);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW);
}

//transmit data
void IIC_send(unsigned char send_data)
{
    for (byte mask = 0x01; mask != 0; mask <= 1) //each character has 8 digits, which is
    ↪ detected one by one
    {
        if (send_data & mask) { //set high or low levels in light of each bit(0 or 1)
            digitalWrite(SDA_Pin, HIGH);
        } else {
            digitalWrite(SDA_Pin, LOW);
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    }
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH); //pull up the clock pin SCL_Pin to end the transmission
    ↳ of data
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, LOW); //pull down the clock pin SCL_Pin to change signals of
    ↳ SDA
    }
}

//the sign that transmission of data ends
void IIC_end()
{
    digitalWrite(SCL_Pin, LOW);
    digitalWrite(SDA_Pin, LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin, HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin, HIGH);
    delayMicroseconds(3);
}

/******motor runs******/
void Car_back() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, speeds_R);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, speeds_L);
    matrix_display(back); //show the image of going back
}

void Car_front() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 255 - speeds_R);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 255 - speeds_L);
    matrix_display(front); //show the image of going front
}

void Car_left() {
    digitalWrite(MR_Ctrl, HIGH);
    analogWrite(MR_PWM, 255 - speeds_R);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, speeds_L);
    matrix_display(left); //show the image of turning left
}

void Car_right() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, speeds_R);
    digitalWrite(ML_Ctrl, HIGH);
    analogWrite(ML_PWM, 255 - speeds_L);

```

(continues on next page)

(continued from previous page)

```
matrix_display(right); //show the image of turning right
}

void Car_Stop() {
    digitalWrite(MR_Ctrl, LOW);
    analogWrite(MR_PWM, 0);
    digitalWrite(ML_Ctrl, LOW);
    analogWrite(ML_PWM, 0);
    matrix_display(STOP01); //show the stop image
}
```

(6)Test Result

Before uploading the program code, the Bluetooth module needs to be removed, otherwise the code upload will fail.

After uploading the code successfully, open the positioning, and then connect the Bluetooth module.

After uploading the code successfully, plug in the Bluetooth module, after power-on, after the mobile APP is connected to the Bluetooth successfully, we can use the mobile APP to control the tank robot

You can also control the robot with the remote control.



KIDSBLOCK TUTORIAL

7.1 1. Getting started with kidsblock

7.1.1 1. Instruction:

The Kidsblock, based on the Scratch graphical programming software, integrates multiple mainstream mainboards, sensors as well as modules. It can be programmed by dragging kidsBlock graphical blocks and using the C/C++ programming language, making programming easy and interesting for children to learn.




7.1.2 2. Download and install KidsBlock software:

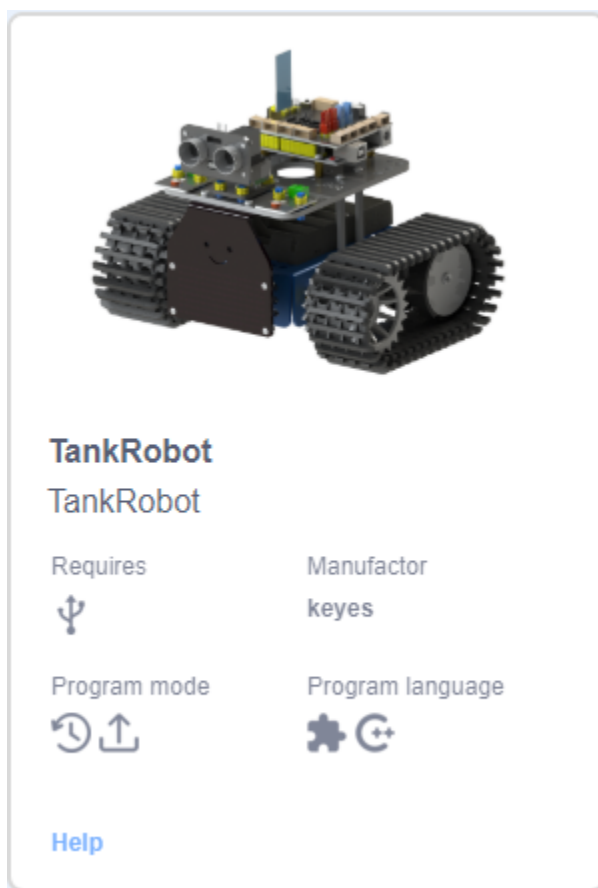
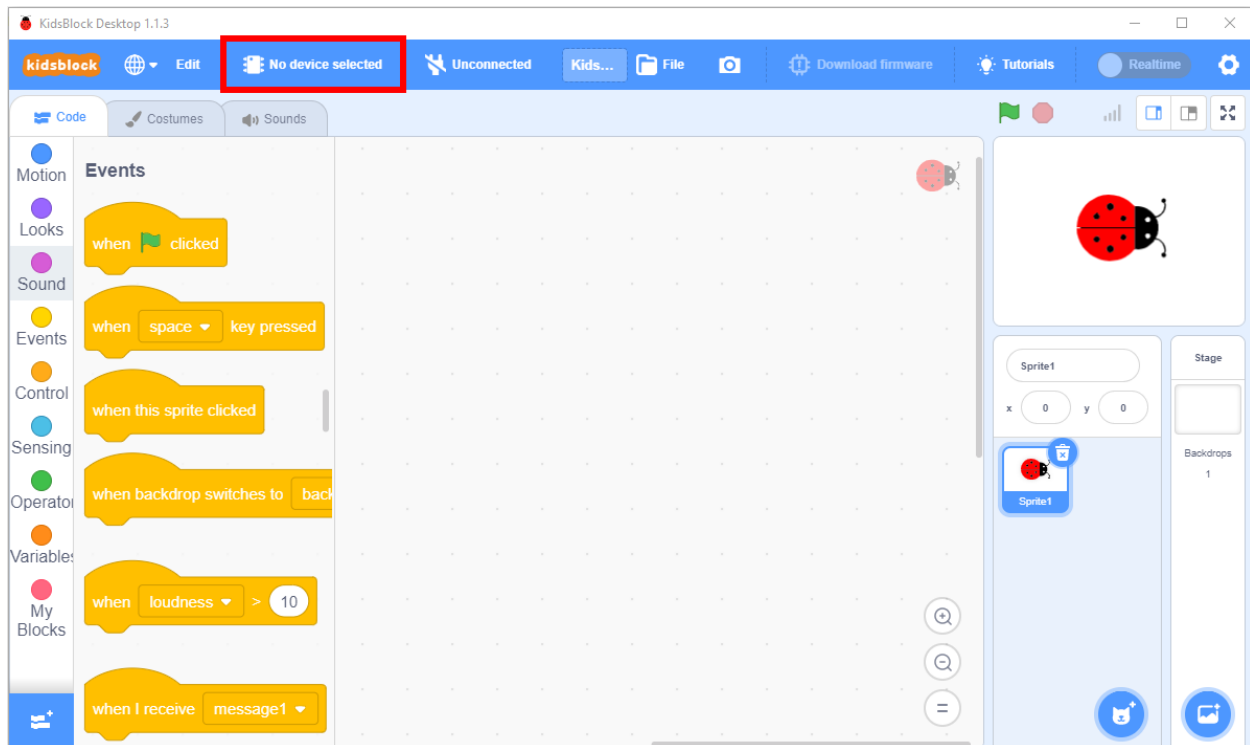
Please refer to the link to install and use the KidsBlock software

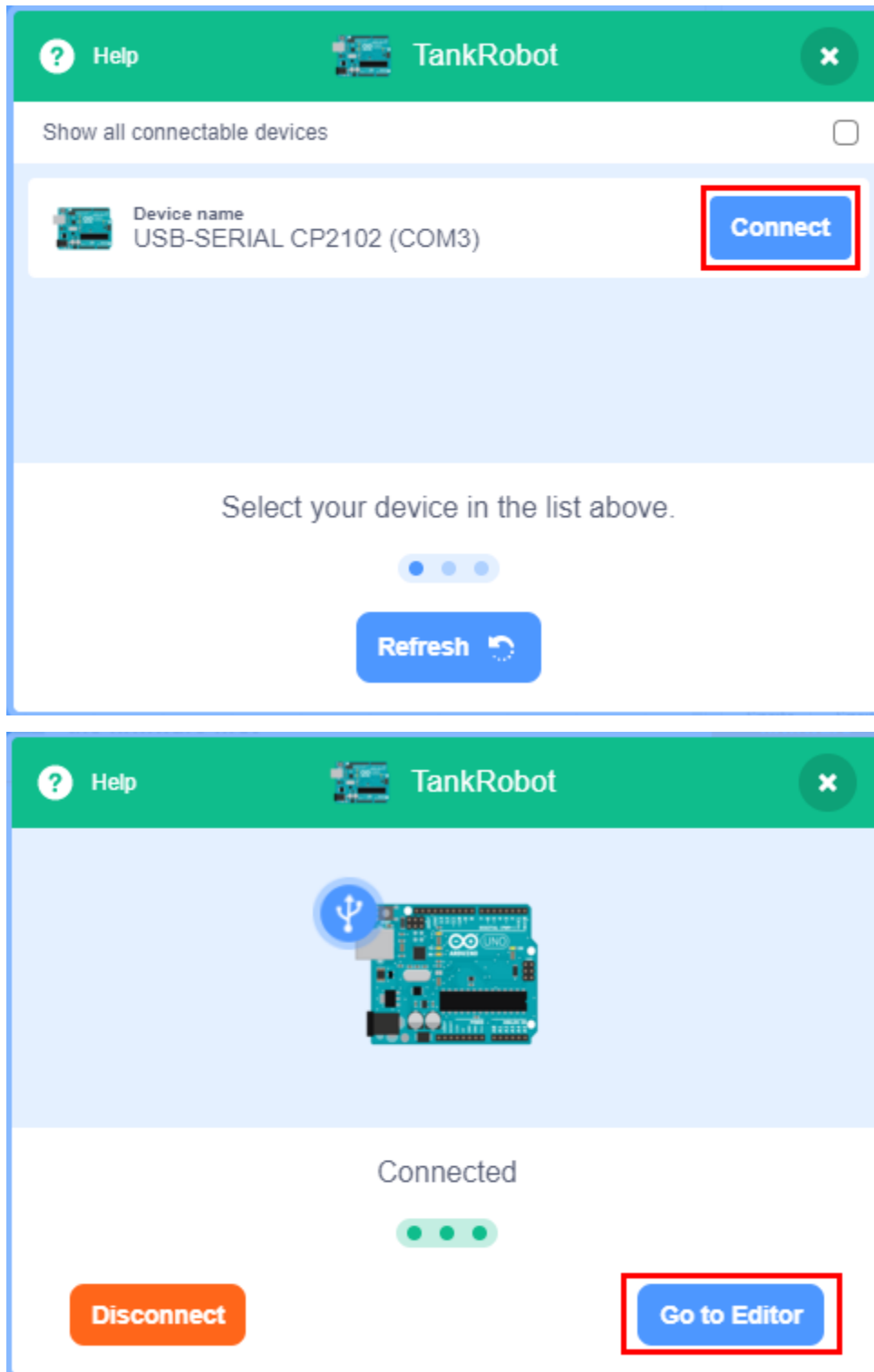
<https://kidsblocksite.readthedocs.io/en/latest/>

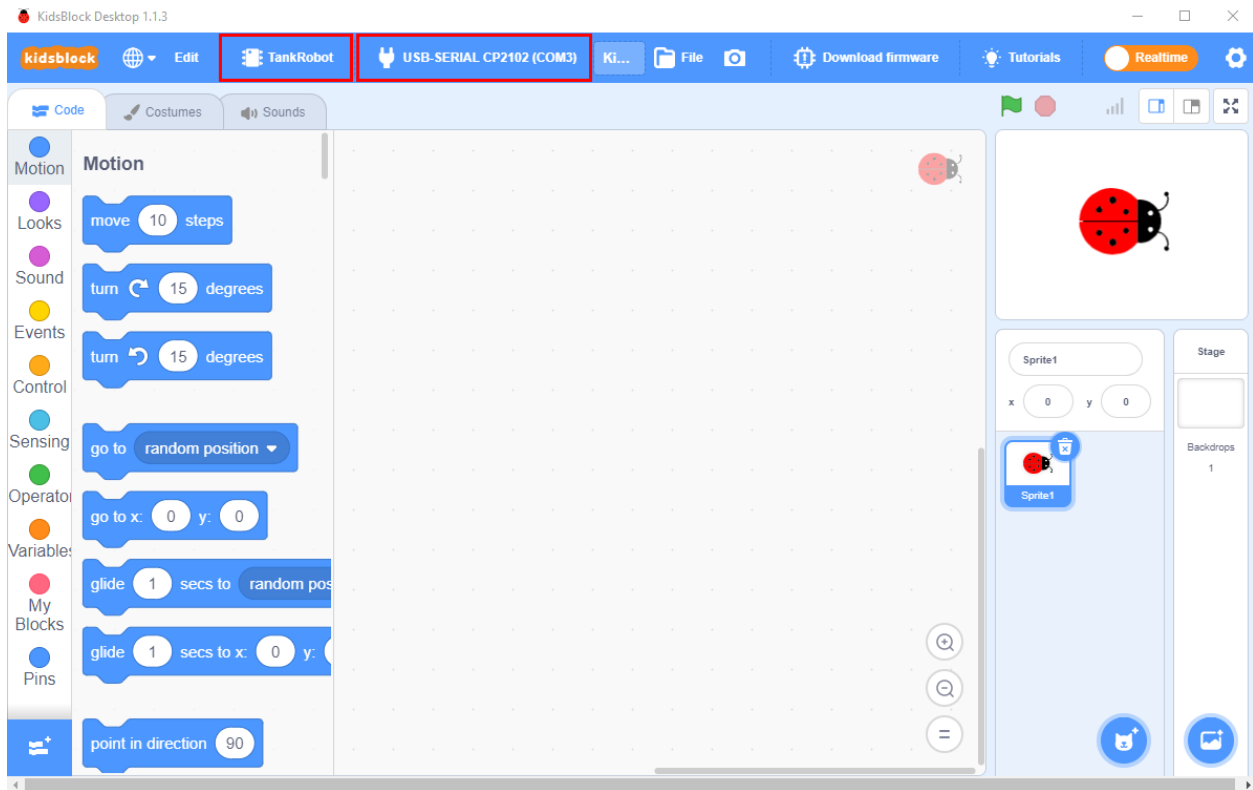
7.1.3 3. Choose TankRobot device:




Open KidsBlock, click  select “TankRobot” and click “Connect”. Then the “TankRobot” is connected. Click “Go to Editor” to return the editor.

 will turn into  turn into , which means the “TankRobot” is connected to (COM) port.

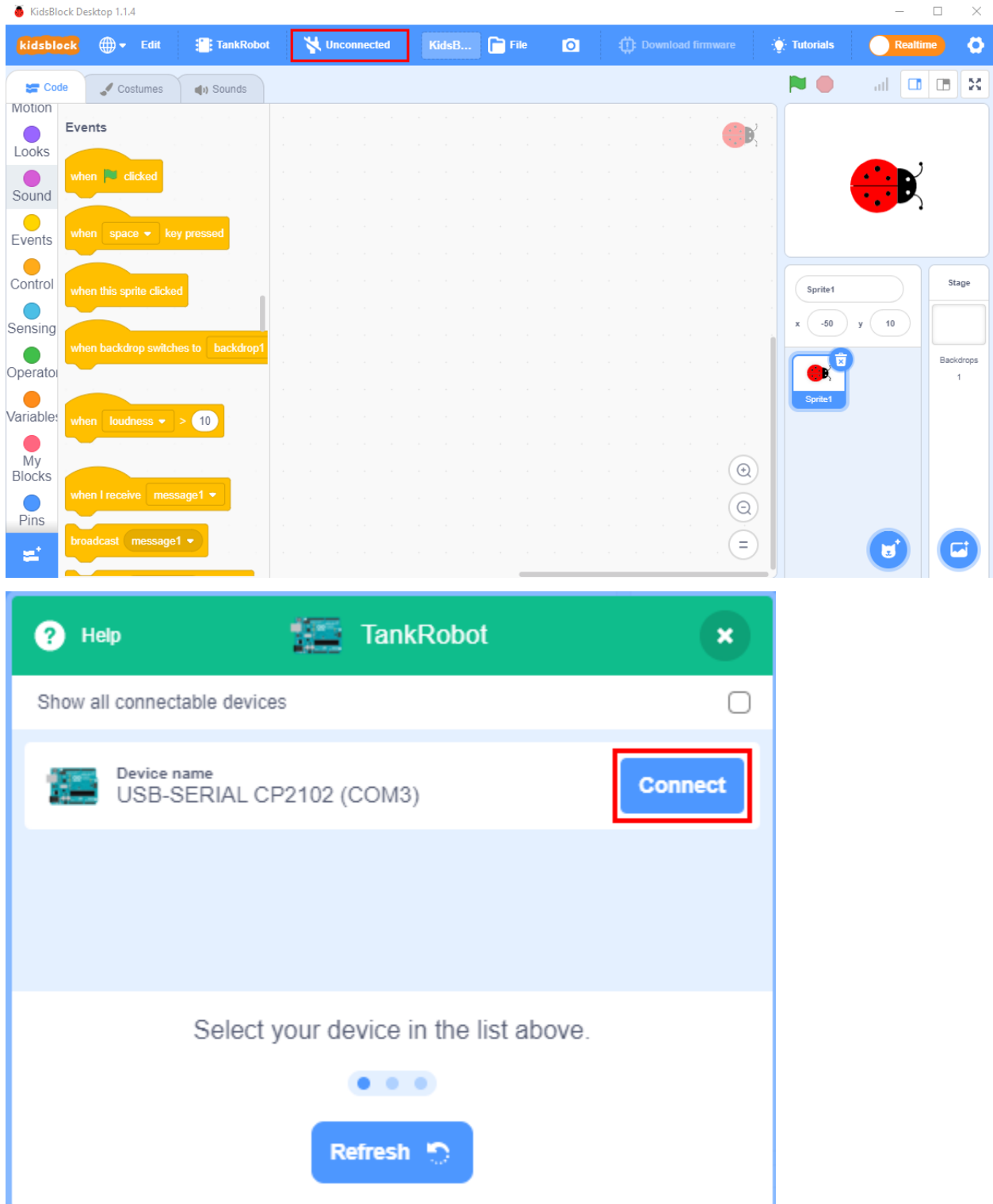


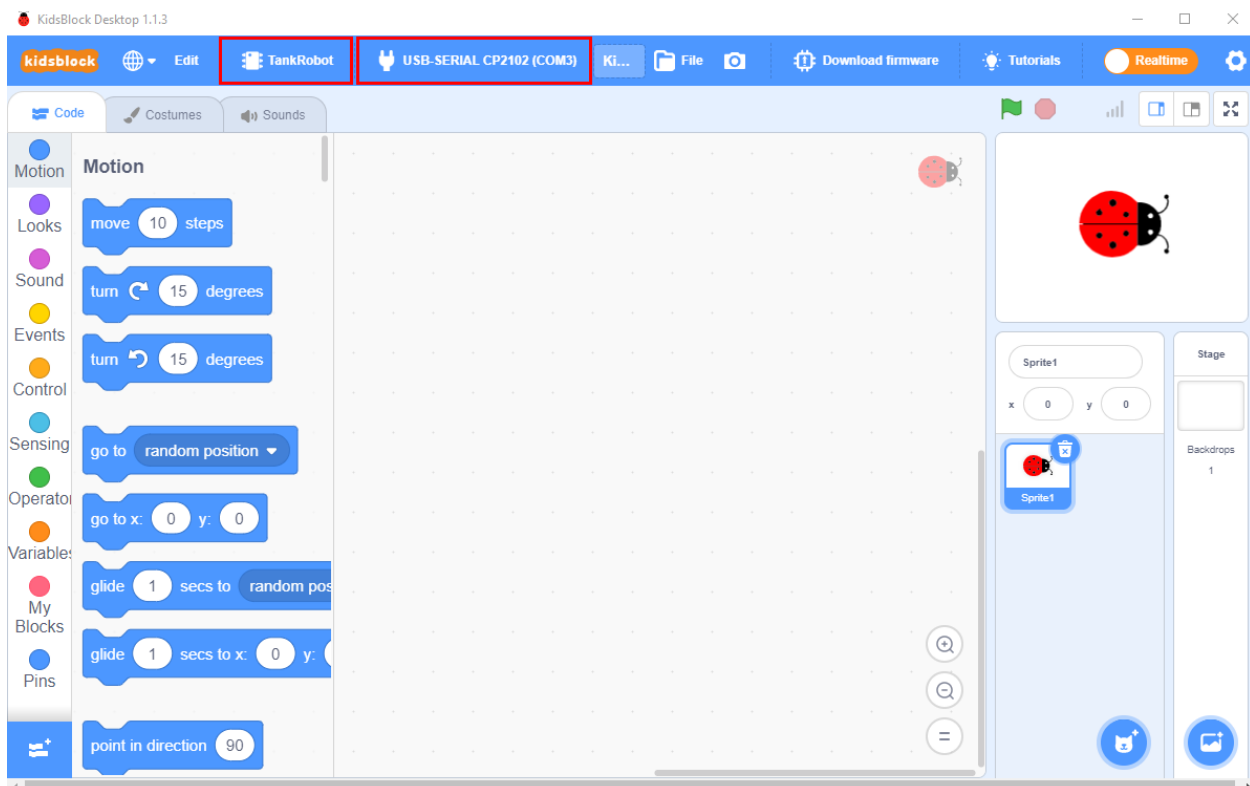
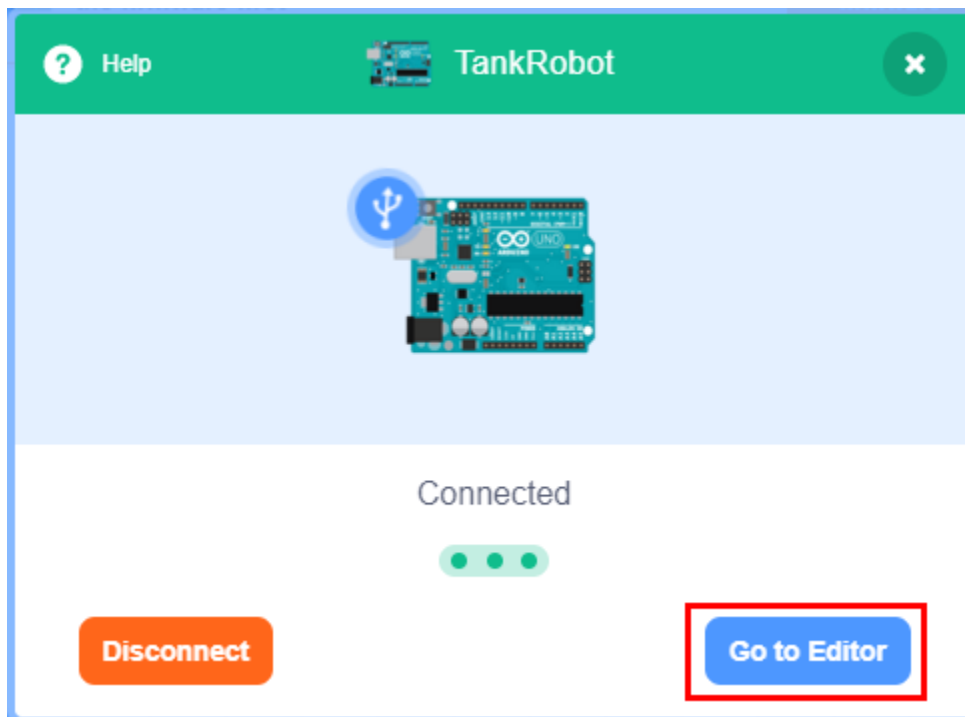







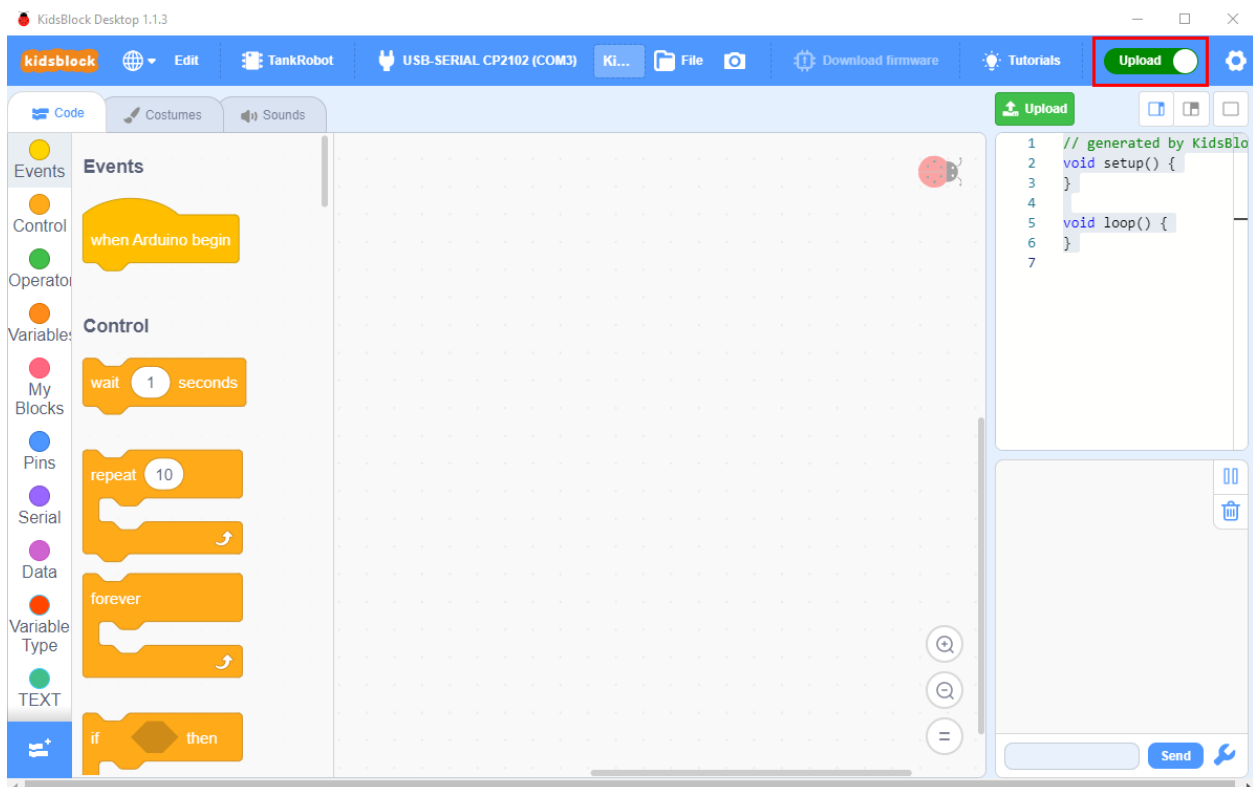
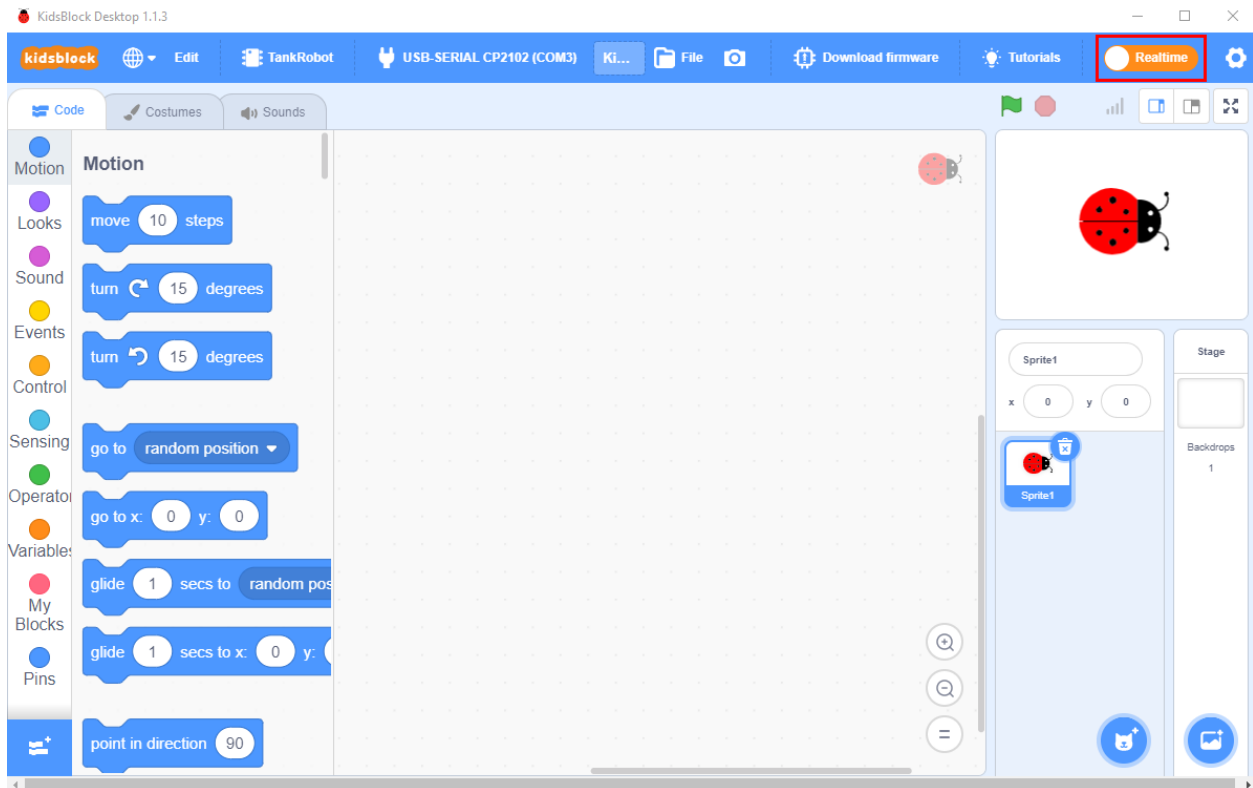
If “**TankRobot**” is connected,  **Disconnected** doesn't turn into  **USB-SERIAL CP2102 (COM3)**, you need to click  **Disconnected** to connect port (COM).

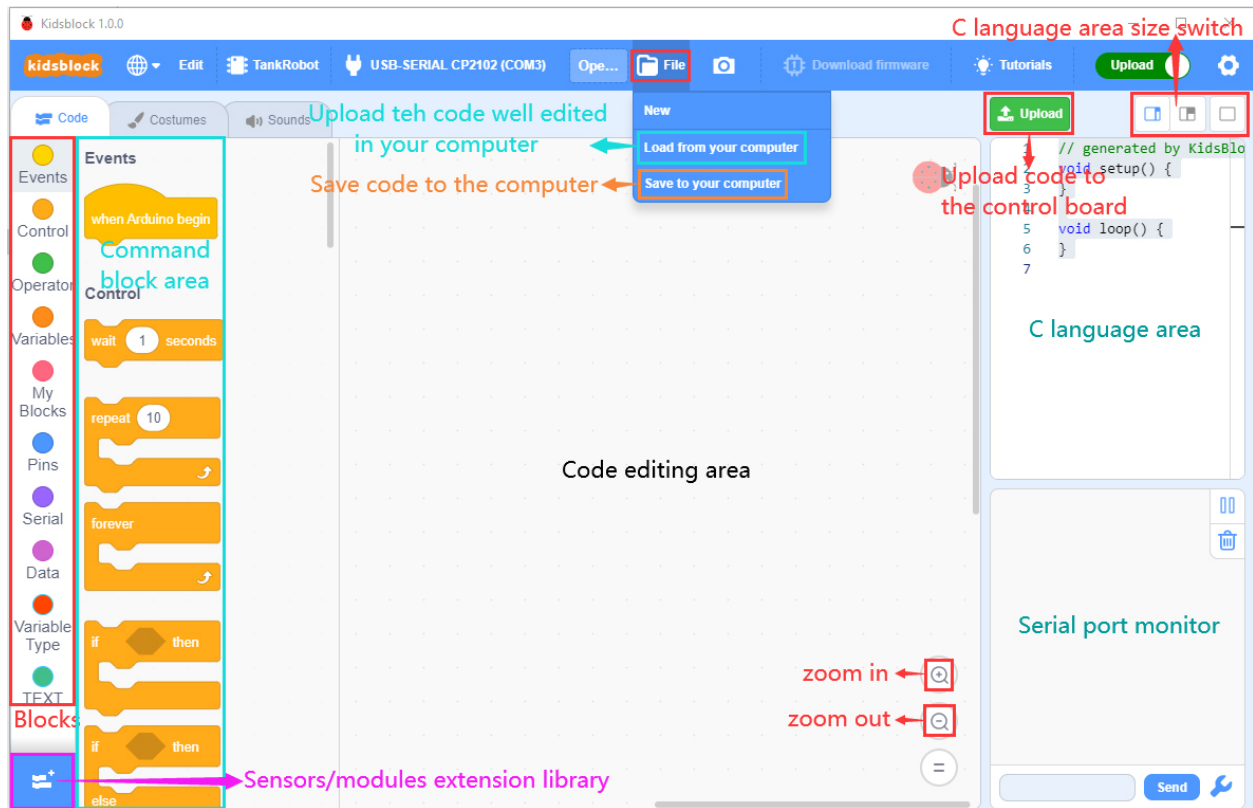
Click  **Disconnected**, then click “**Connect**”. **Connected** page will be shown, which means connecting.





“TankRobot” and COM port are connected, then click  to switch mode.  will switch into .

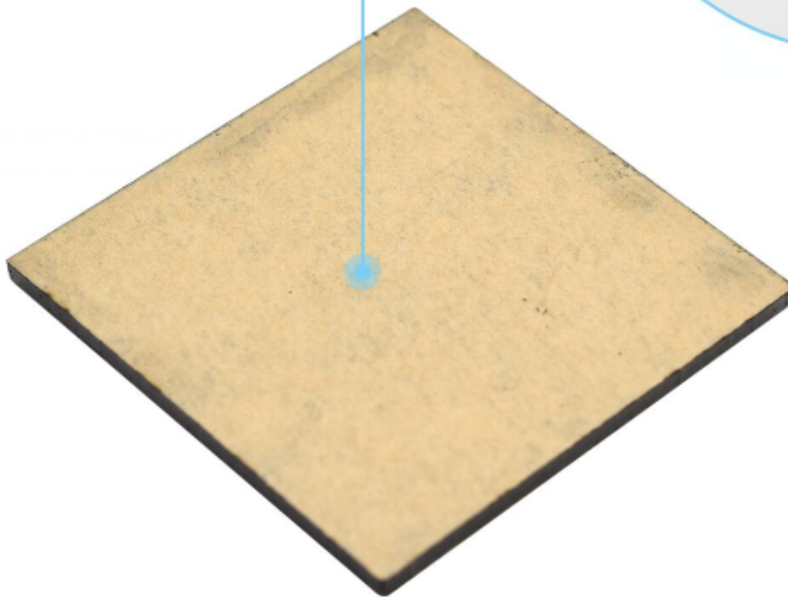




7.2 2. Assemble Ultrasonic Tank Robot

Caution: Set the initial angle of the servo Peel thin films off boards before installing this robot

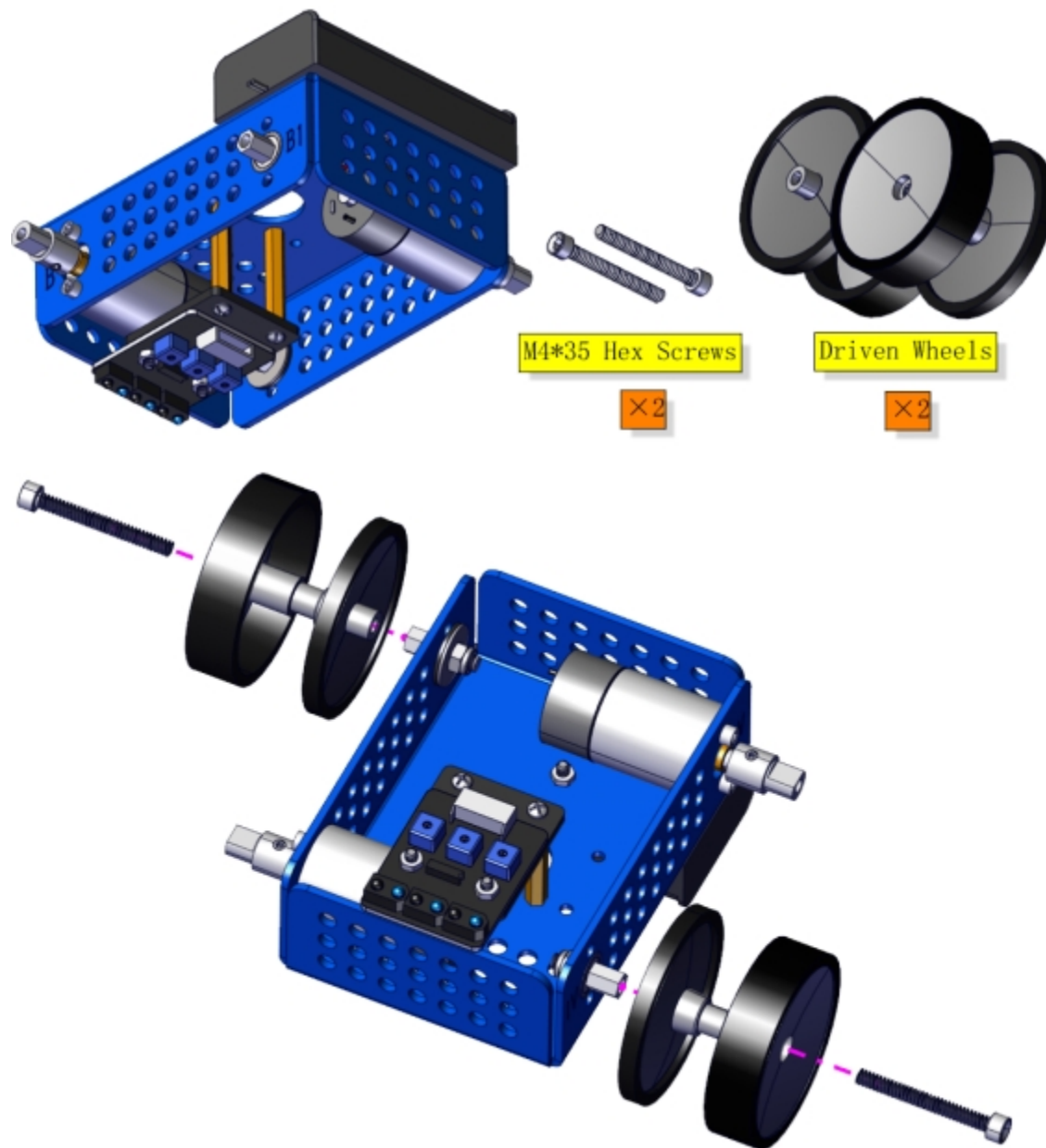
**Before assembly, please
tear off the protective
film on the acrylic.**



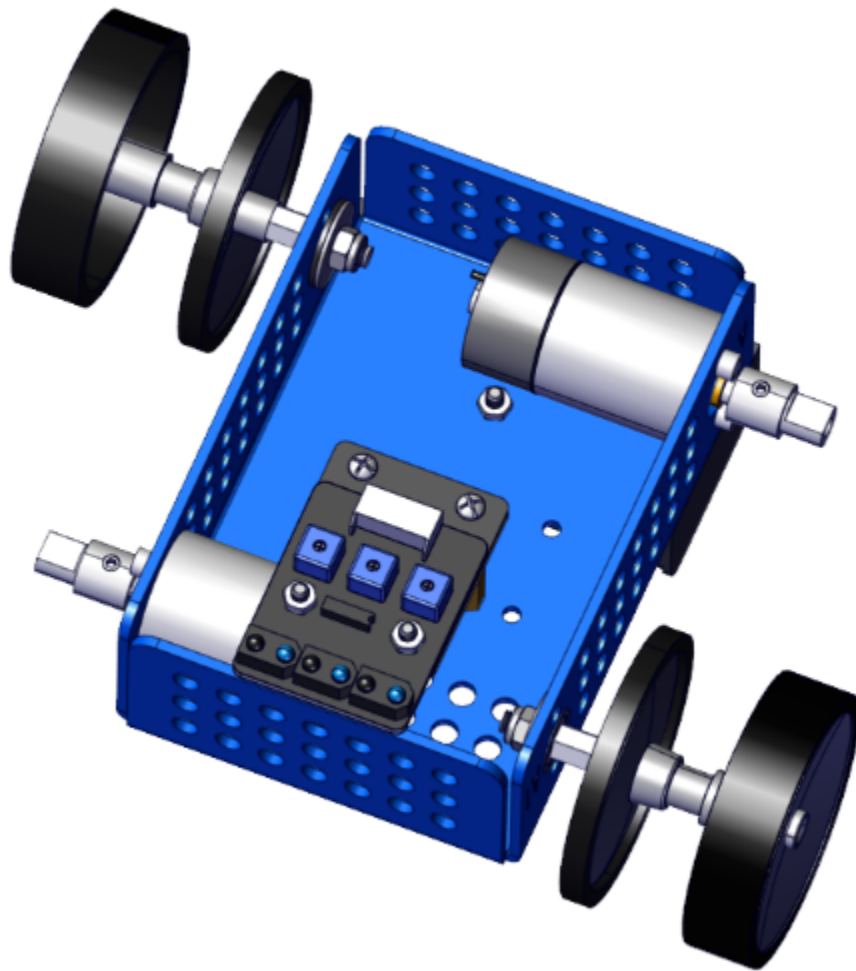
Step1

Tools needed:

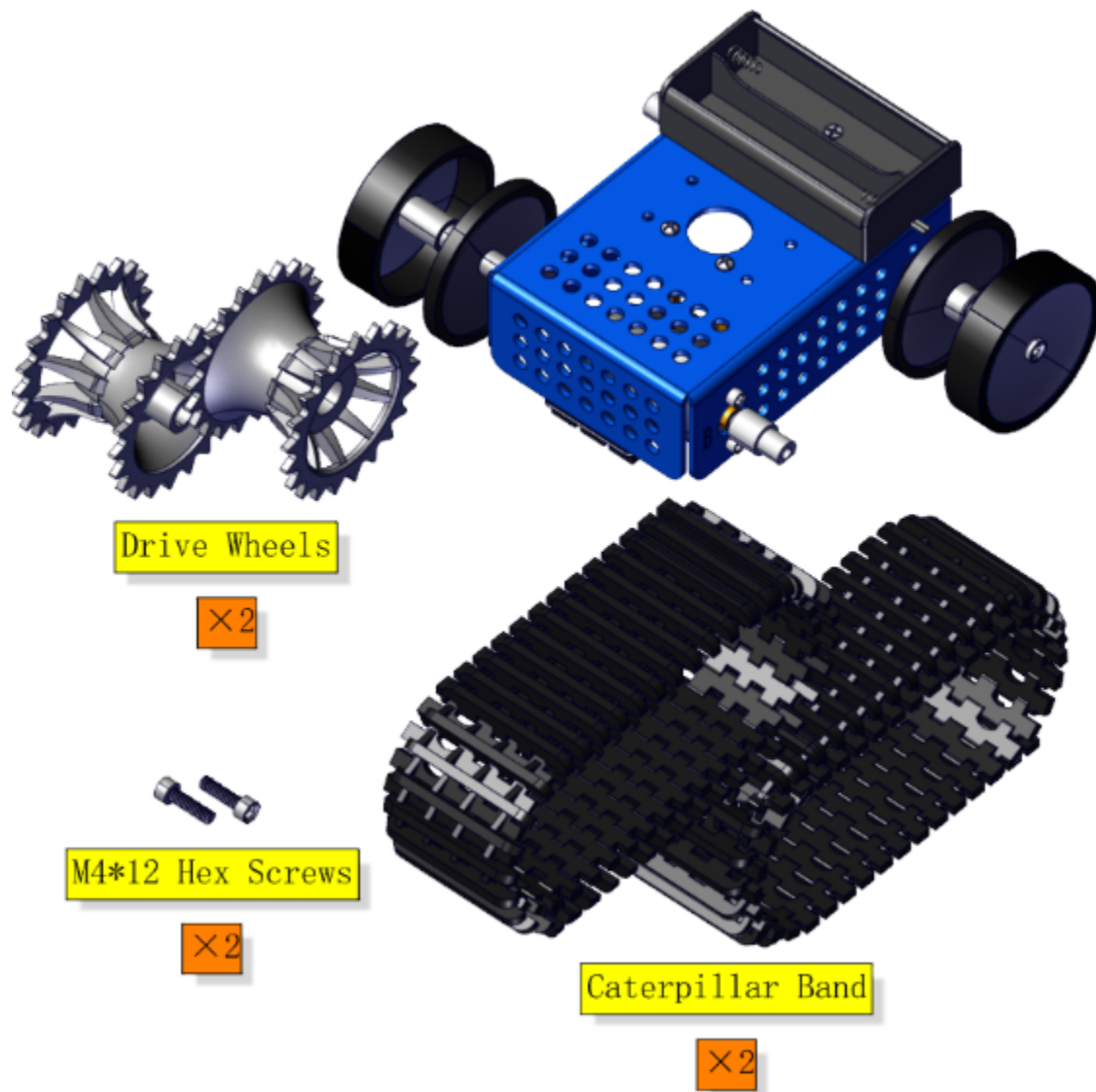


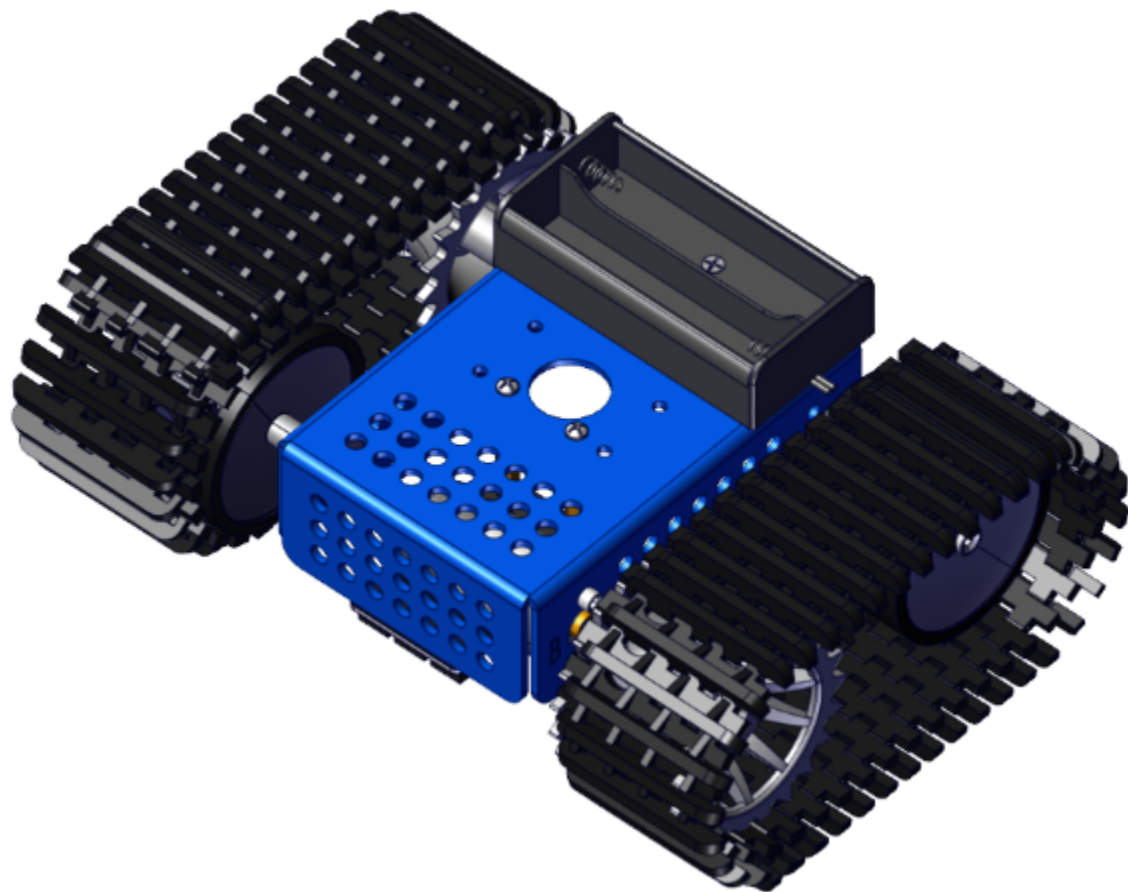
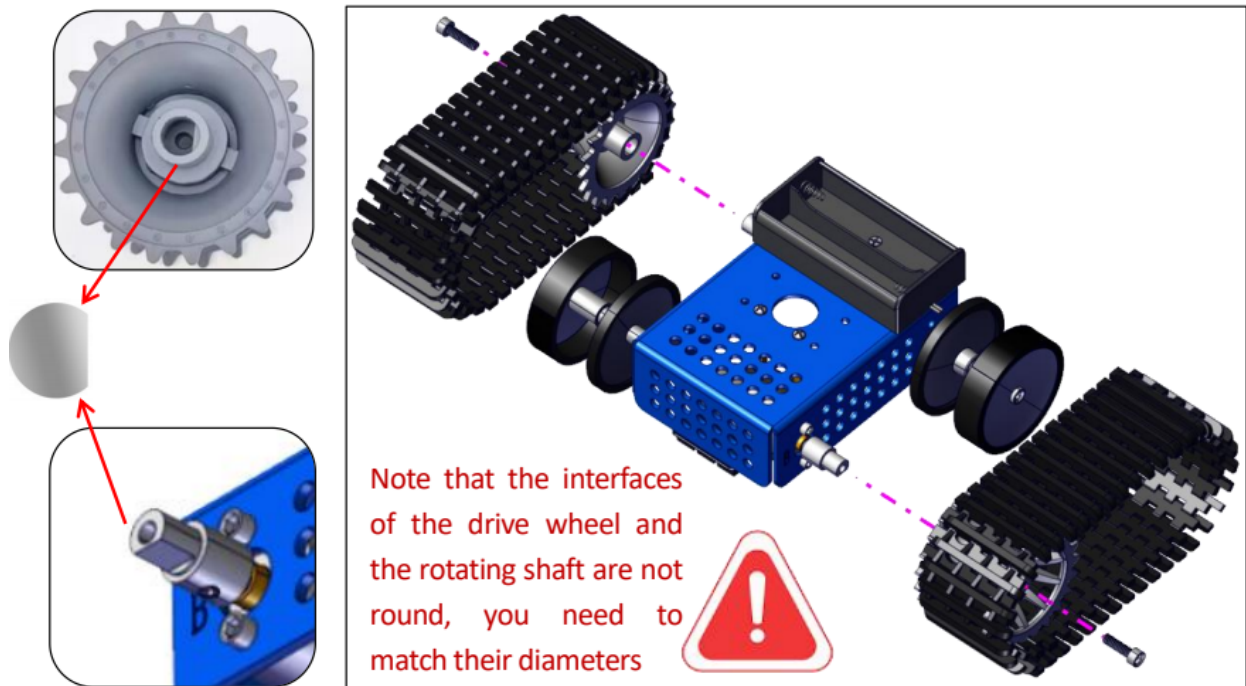


Pay attention to the installation direction of the wheels. The thick side is on the outside.

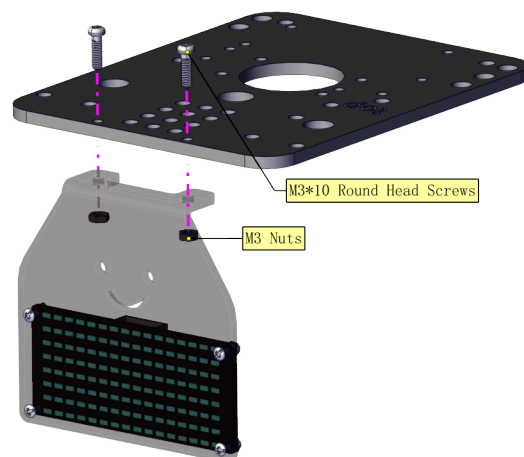
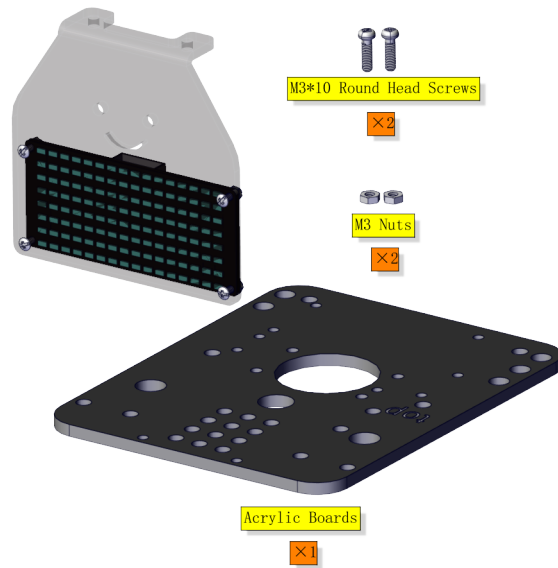


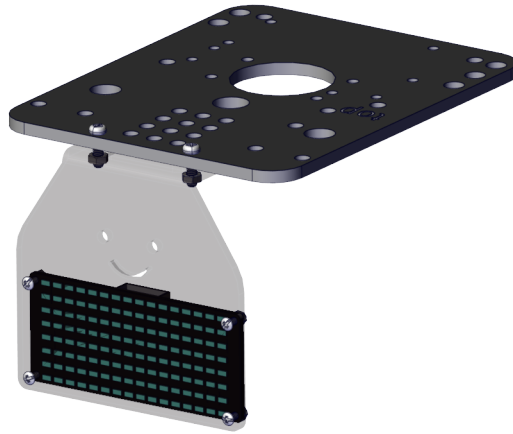
Step2



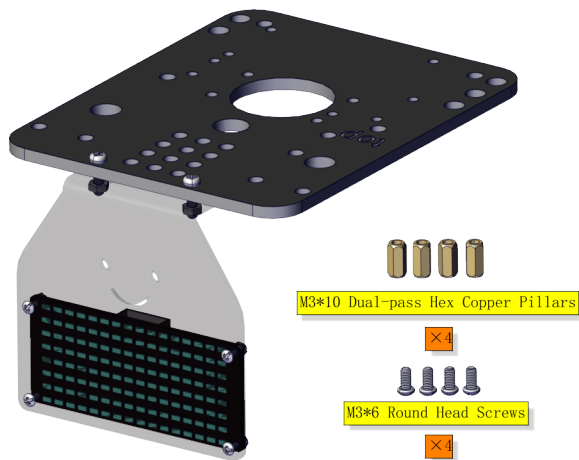


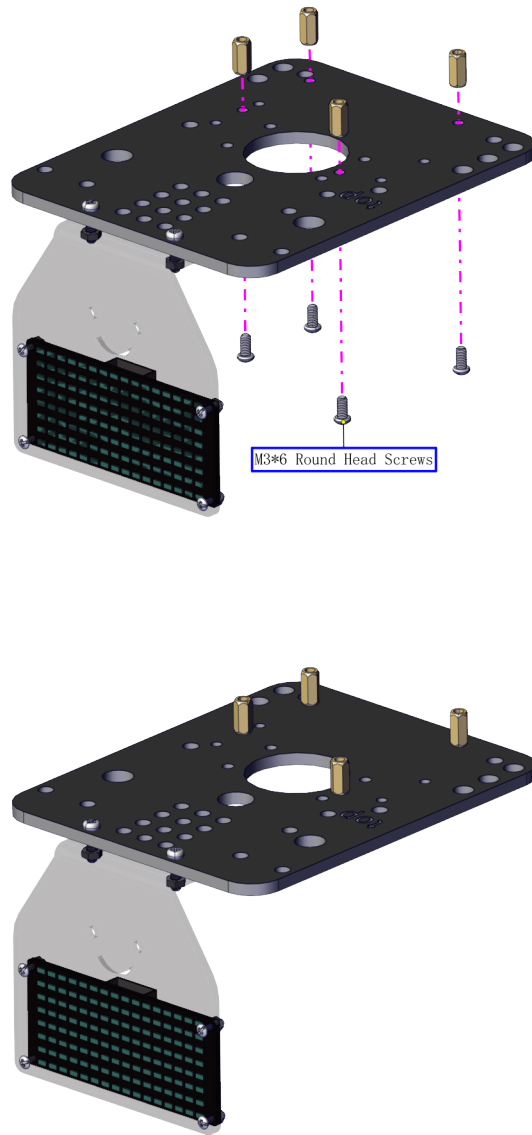
Step3



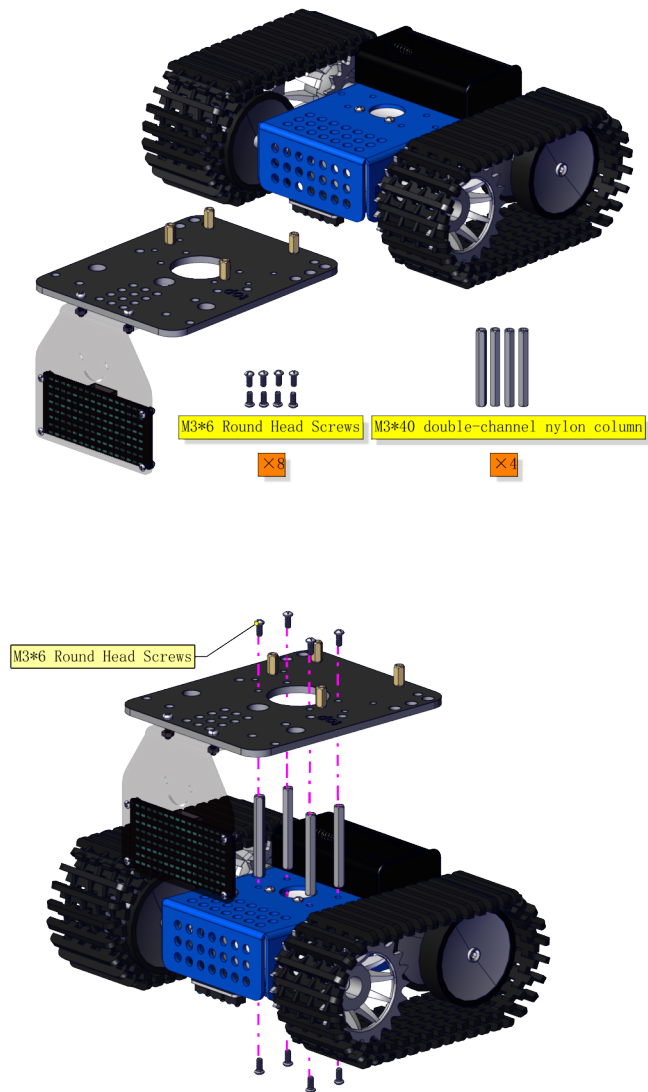


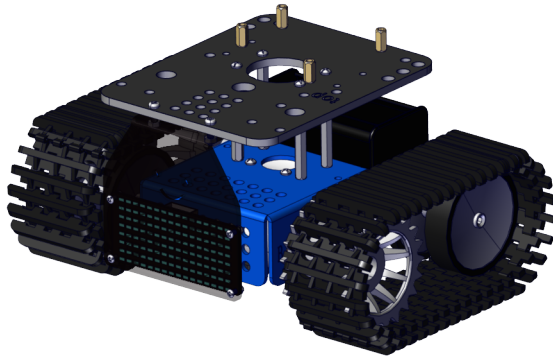
Step 4



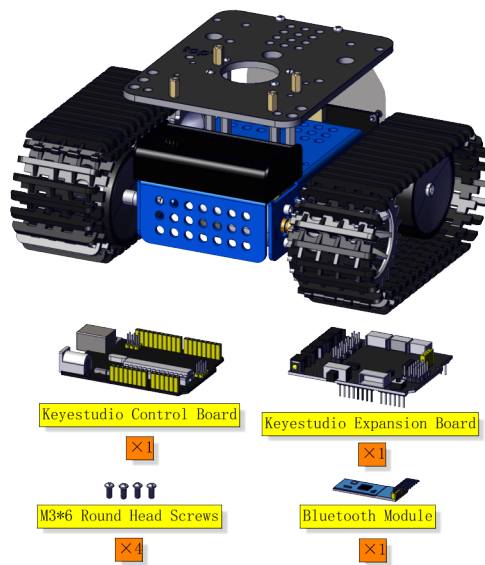


Step 5

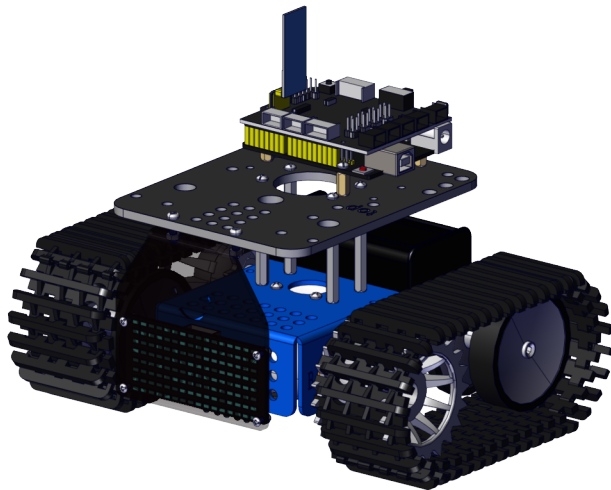
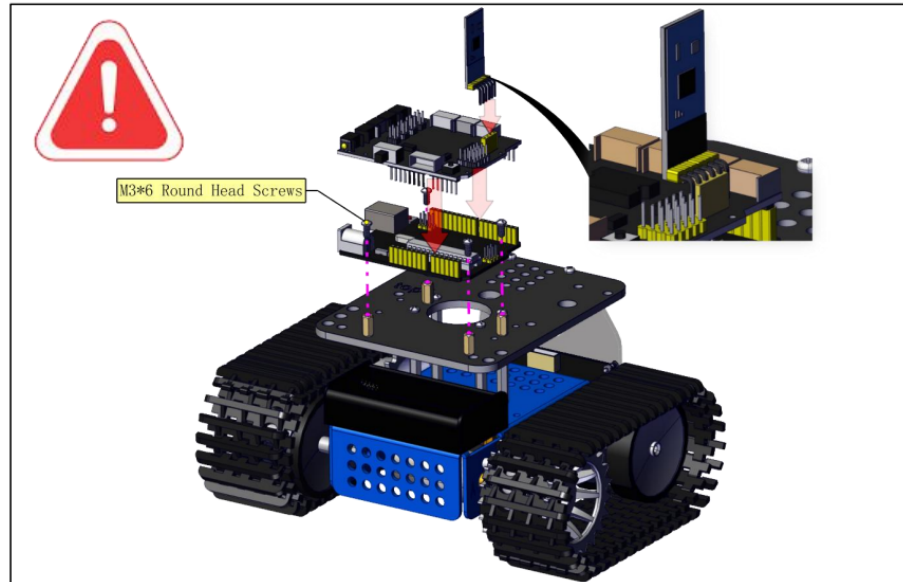




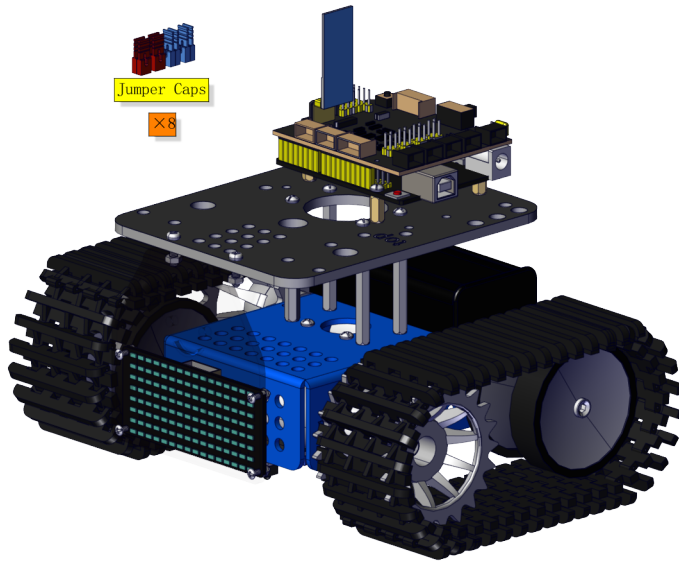
Step 6



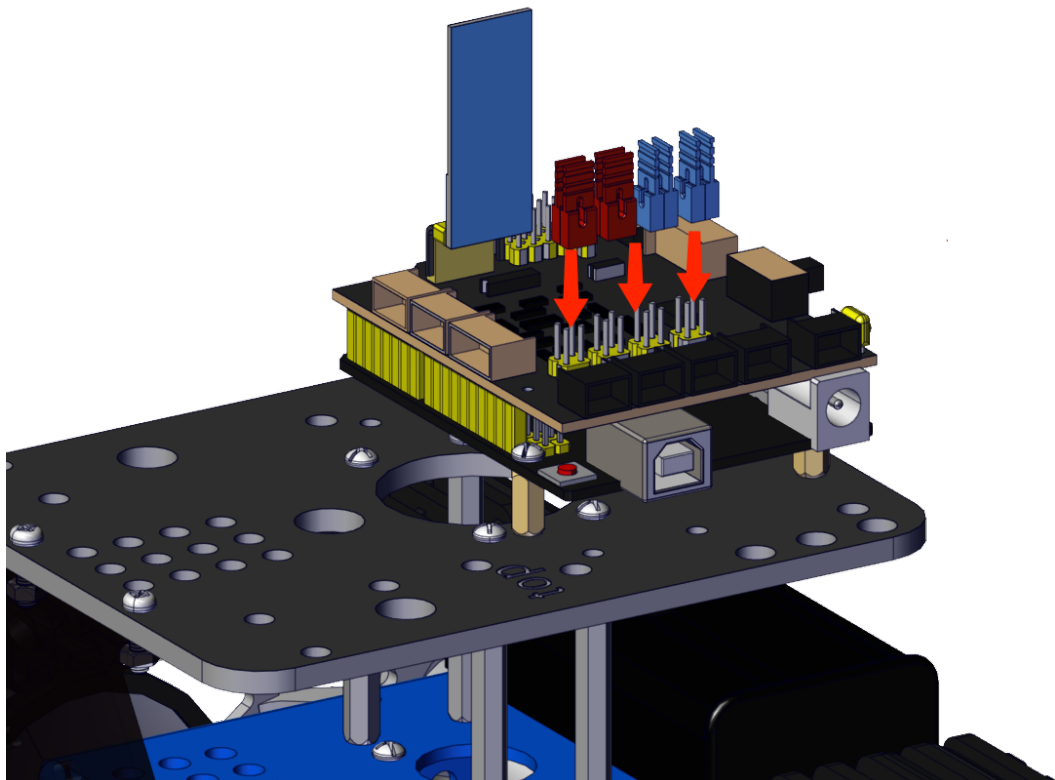
Note: The Bluetooth module will occupy the serial port. Please do not connect to Bluetooth when uploading code.

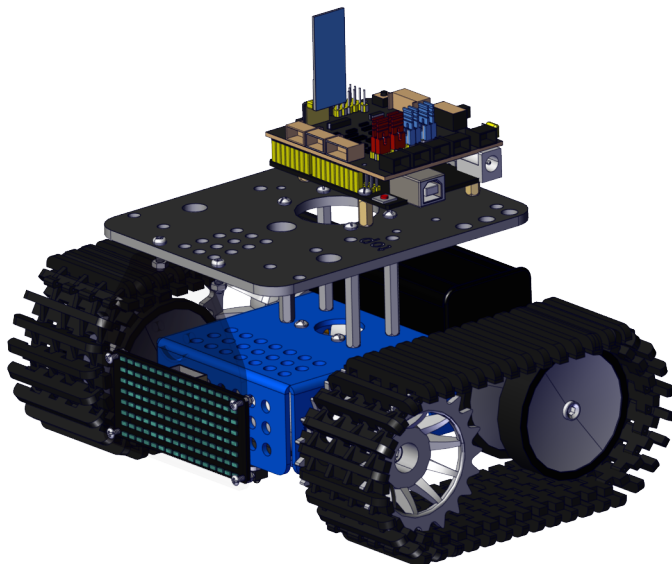


Step 7

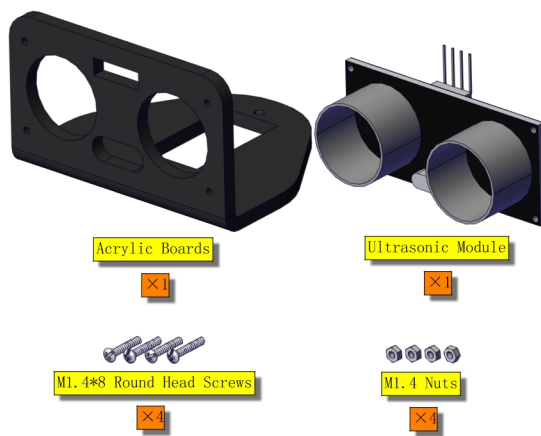


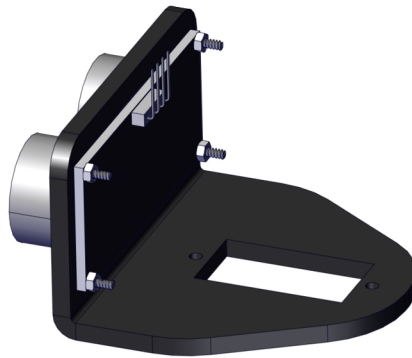
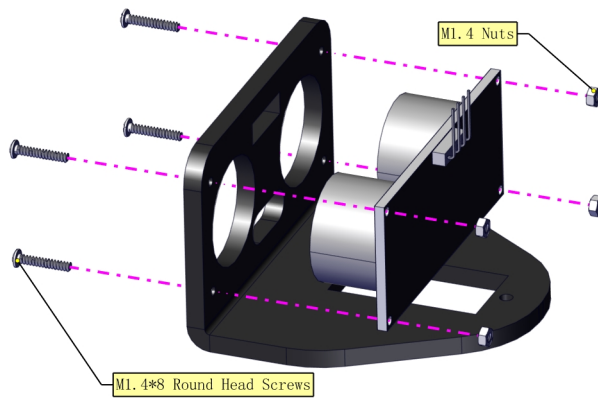
Note the direction of jumper caps



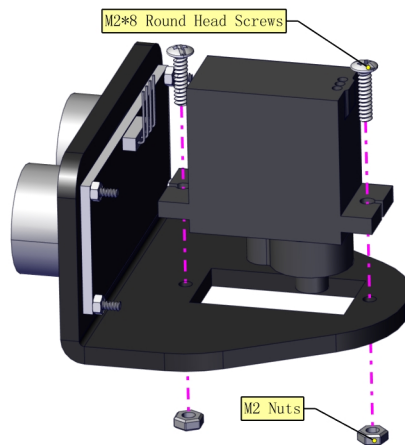
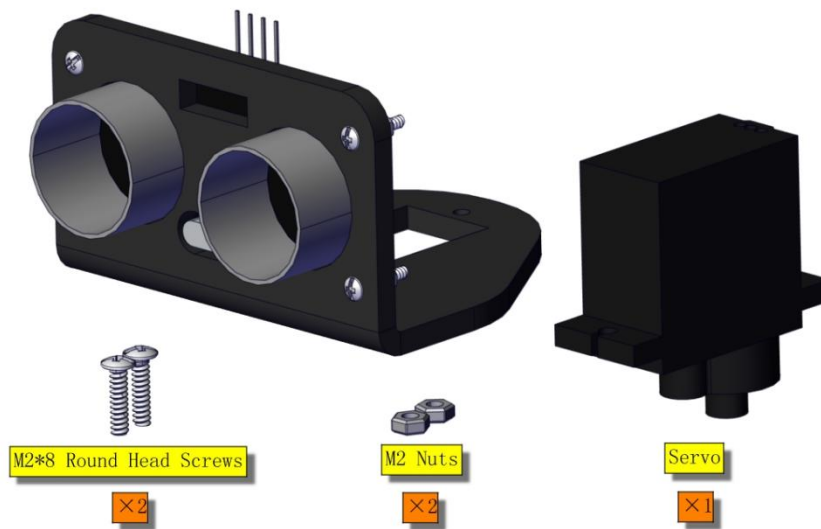


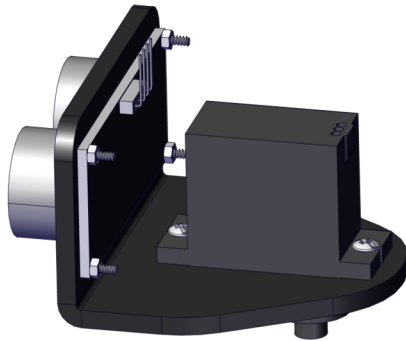
Step 8





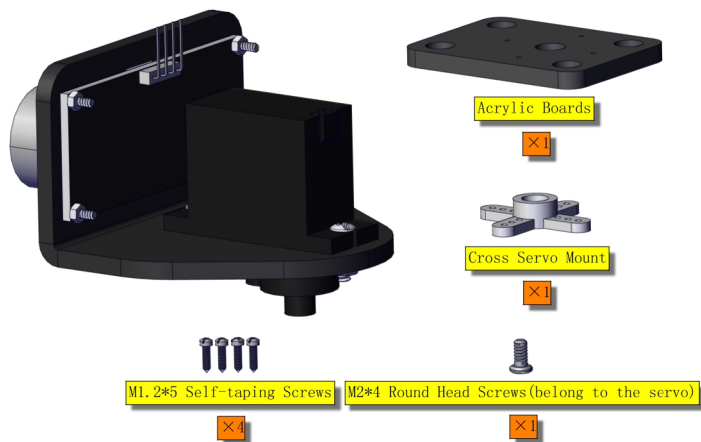
Step 9

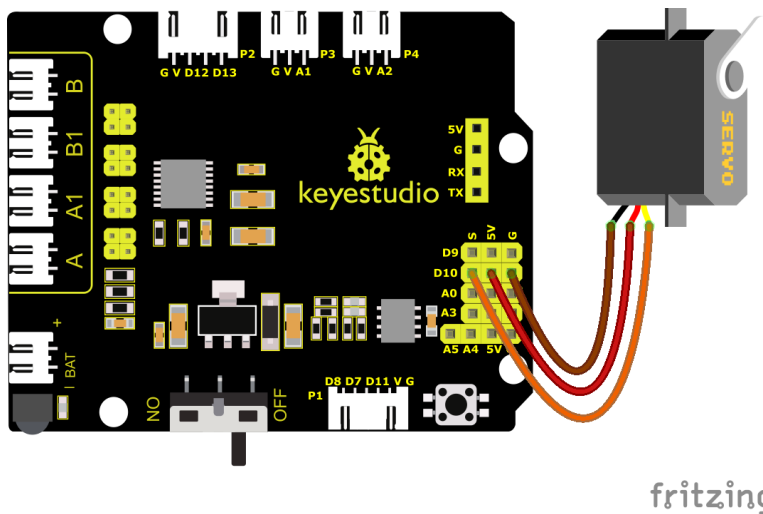




Step 10

Need to adjust the angle of the servo



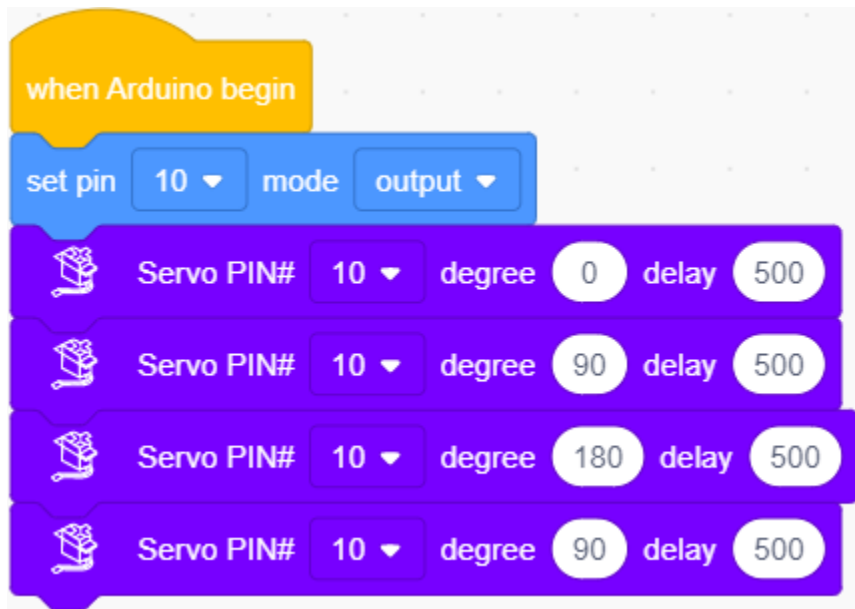


fritzing

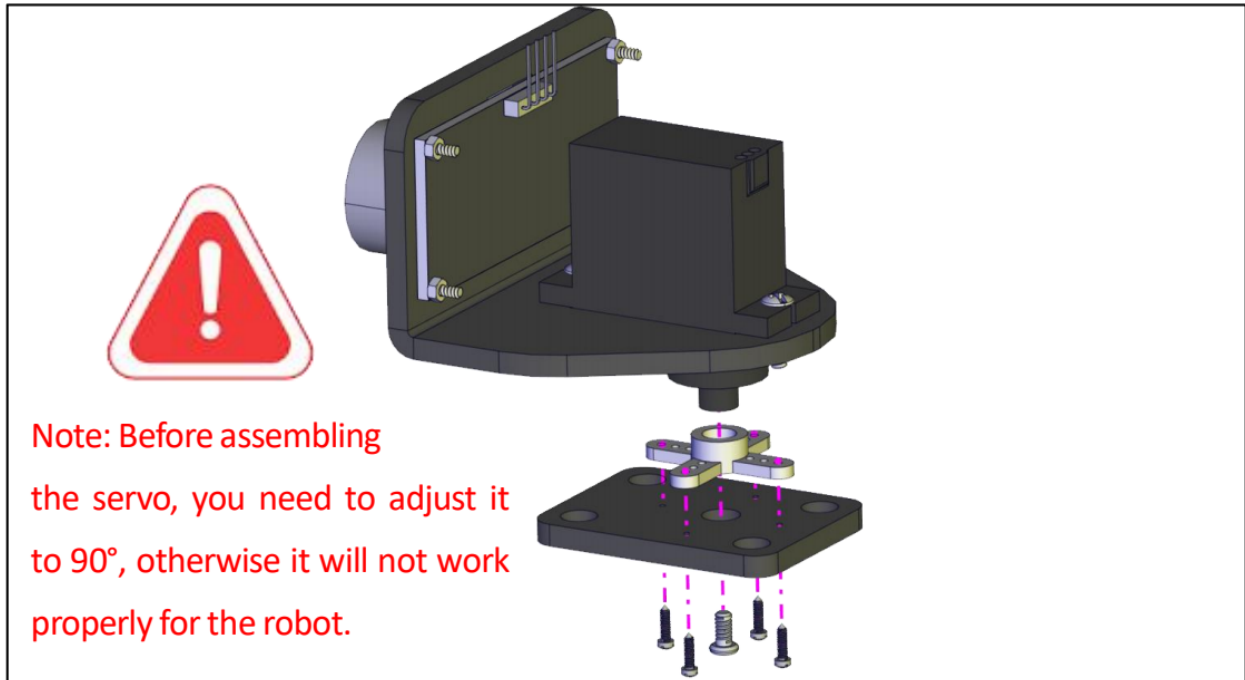
Set the angle of the servo to 90°

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

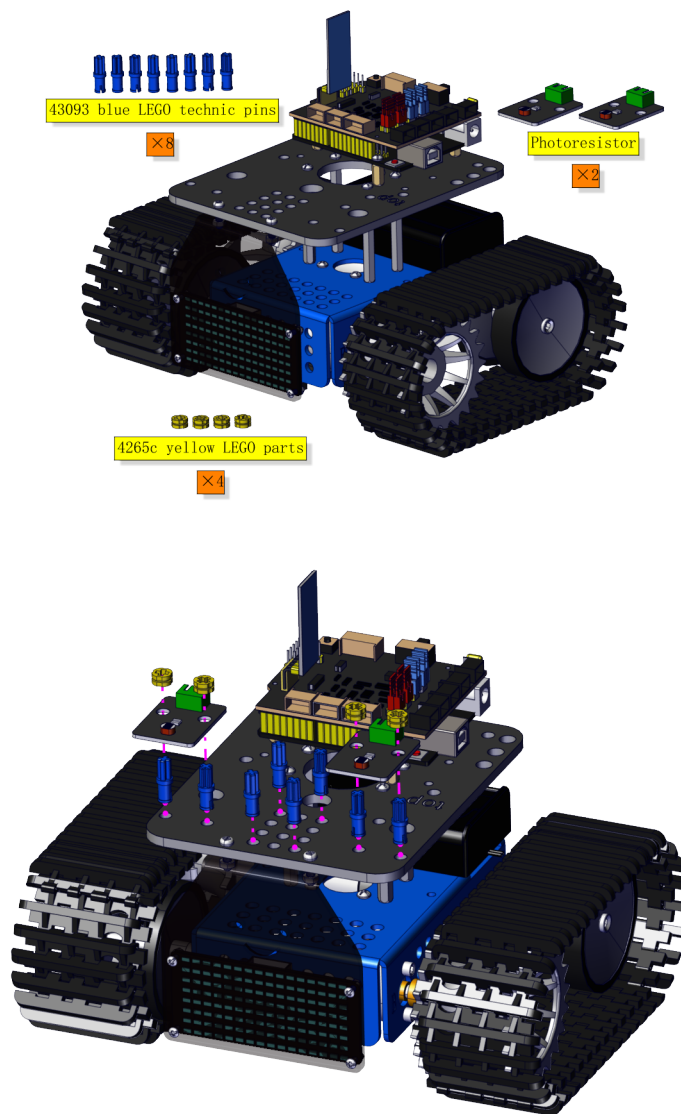
Writing the following code in kidsBlock software and upload it to the motherboard, or just open the code provided by us and upload it to the motherboard.

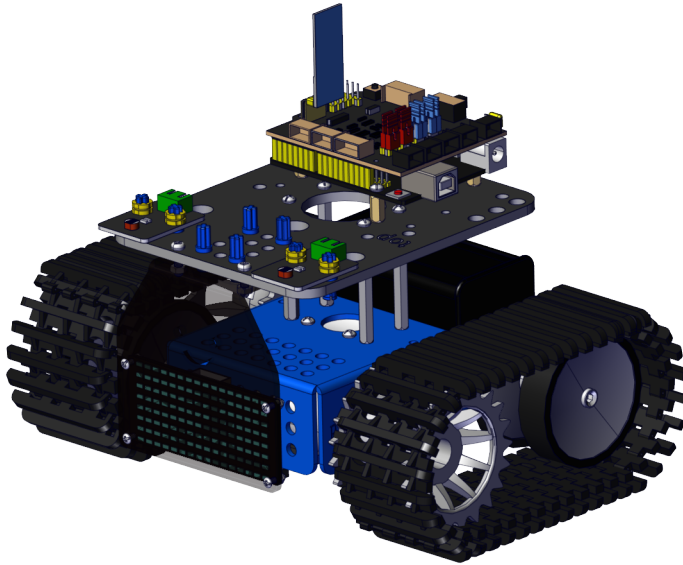


Keep the ultrasonic sensor parallel to the board

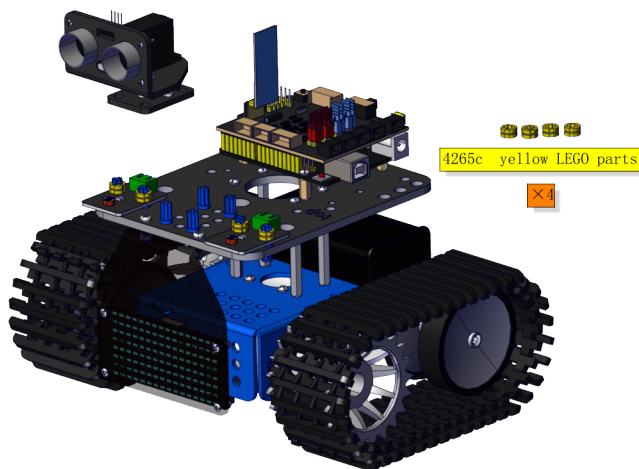


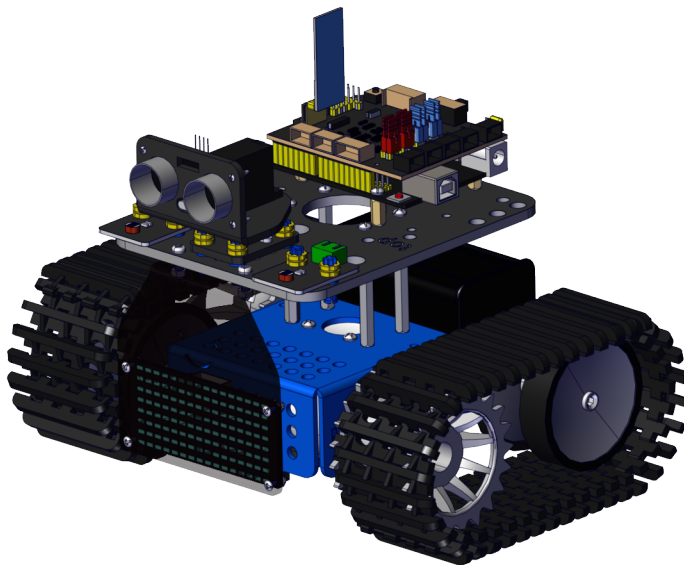
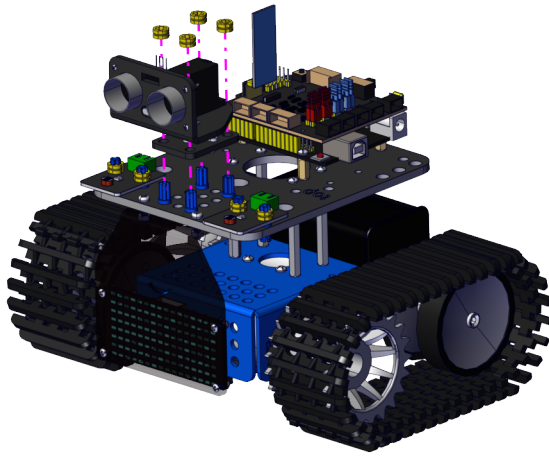
Step 11





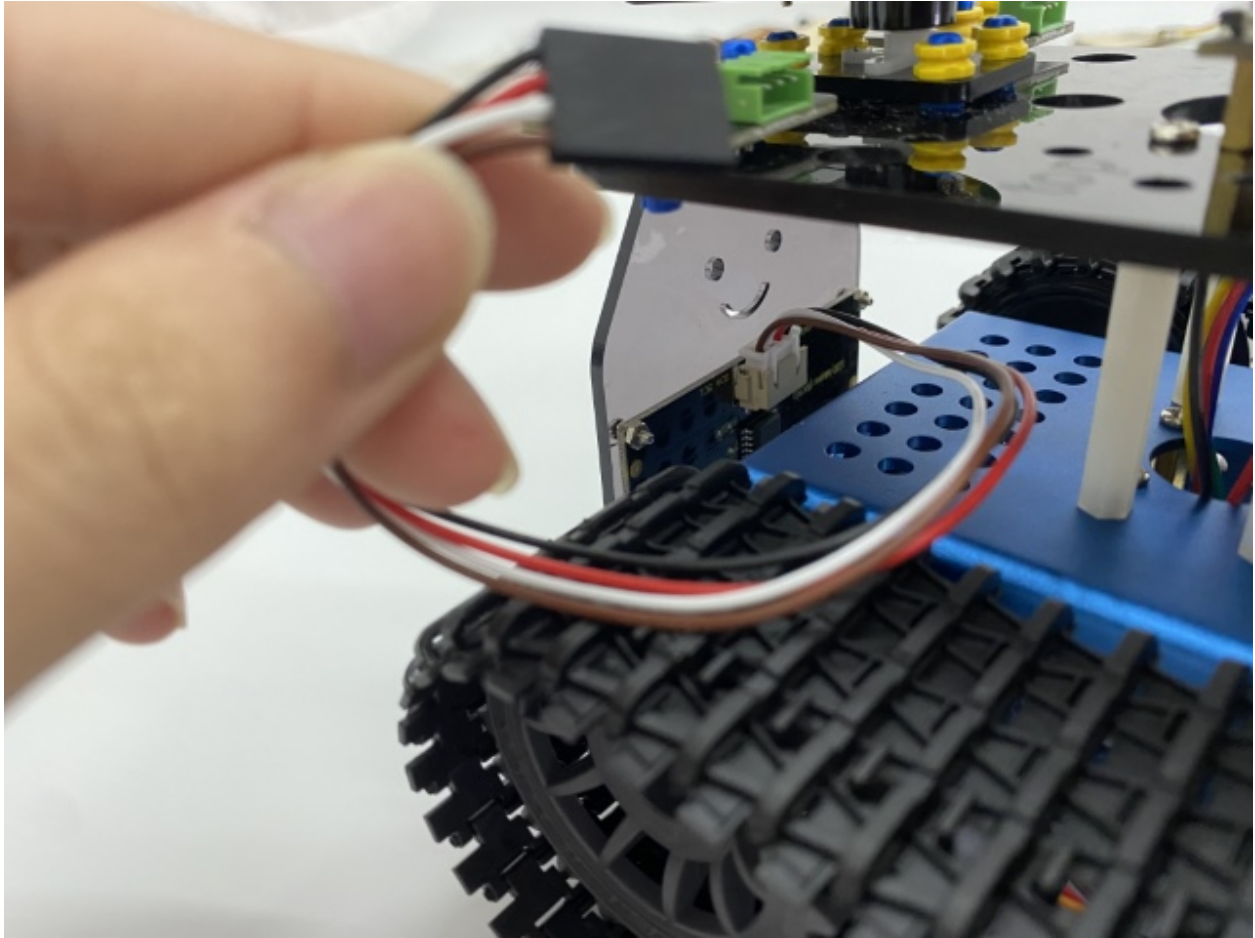
Step 12

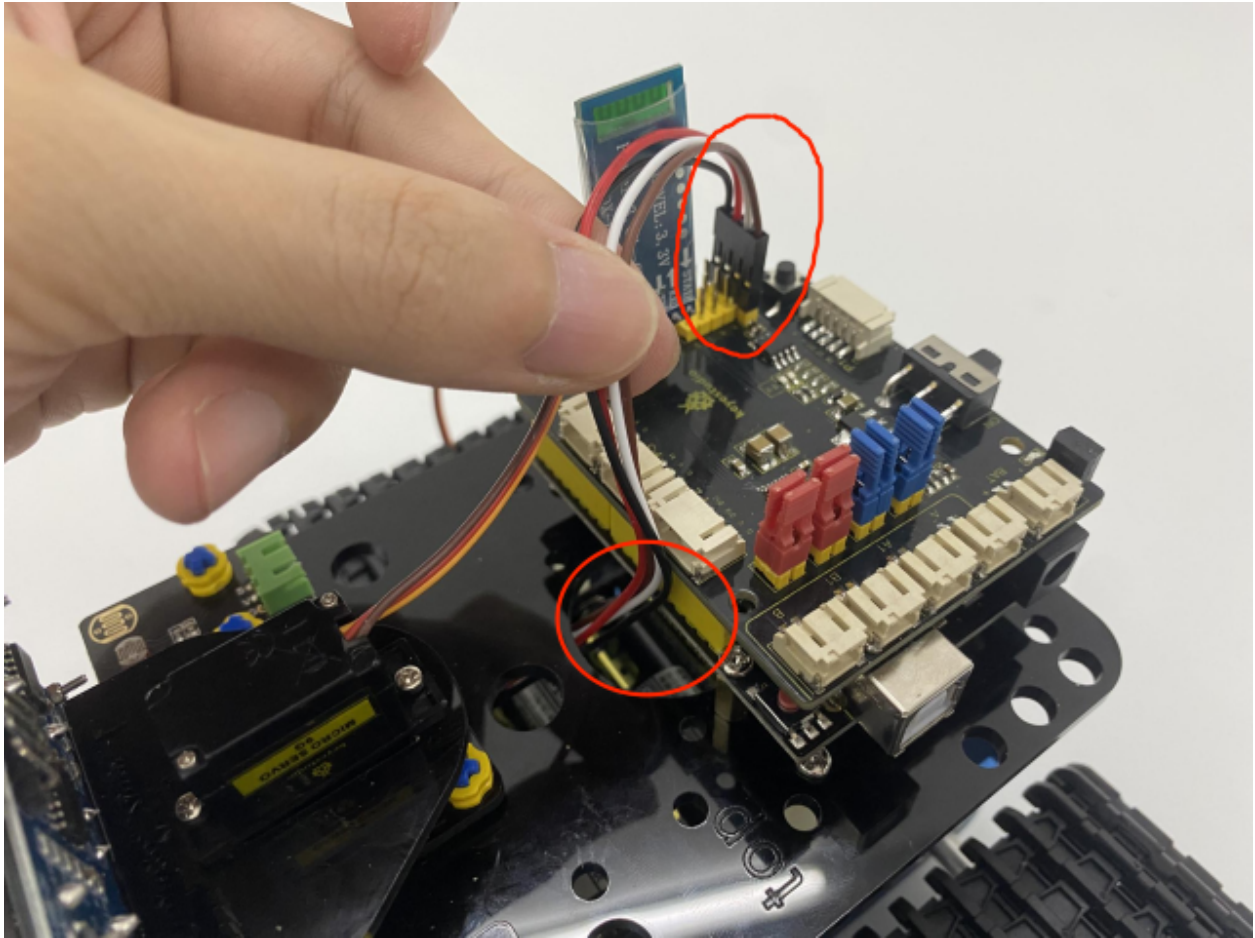




Wire up

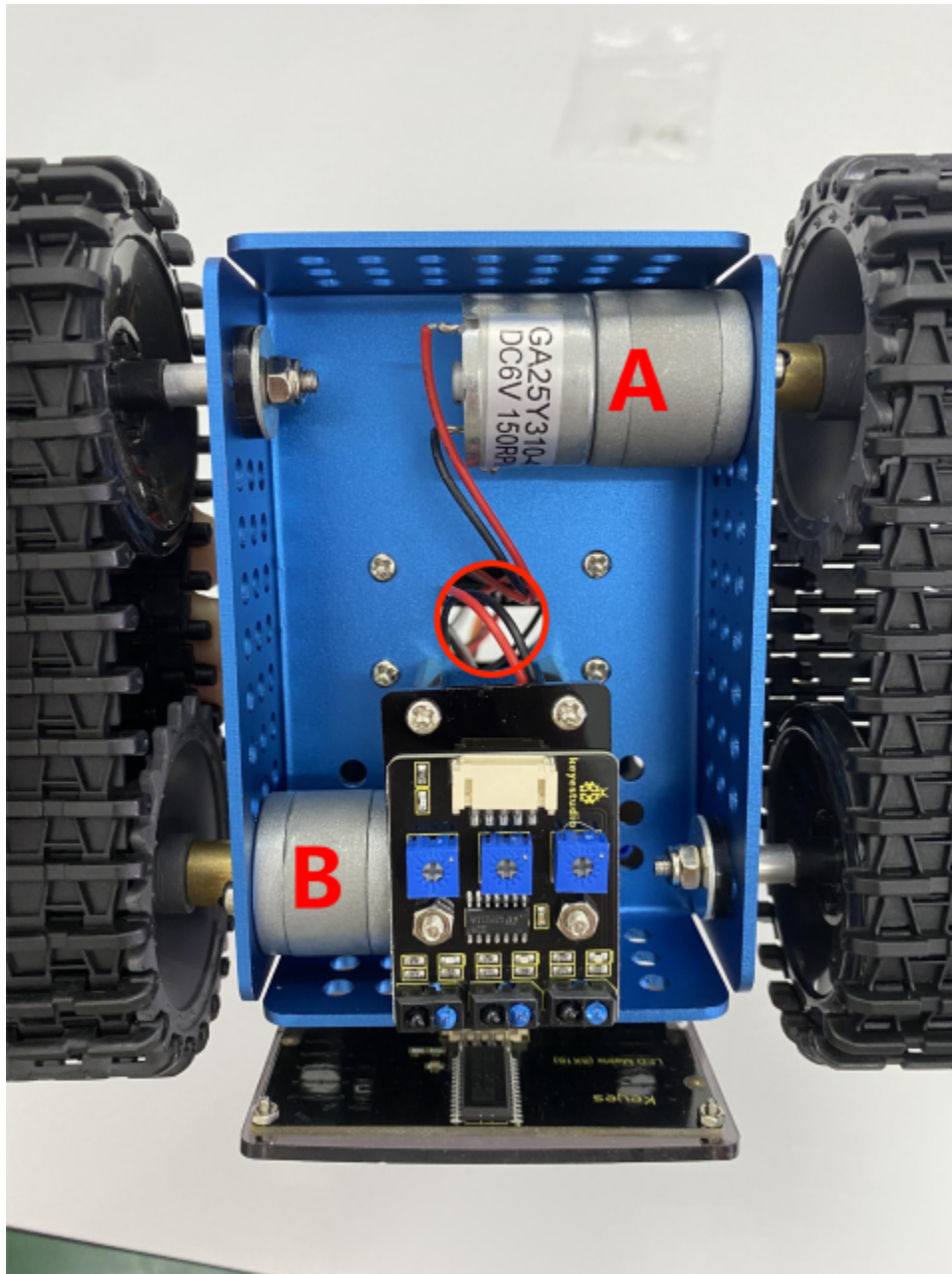
For 8*16LED panel, Make wires connect to A4 and A5

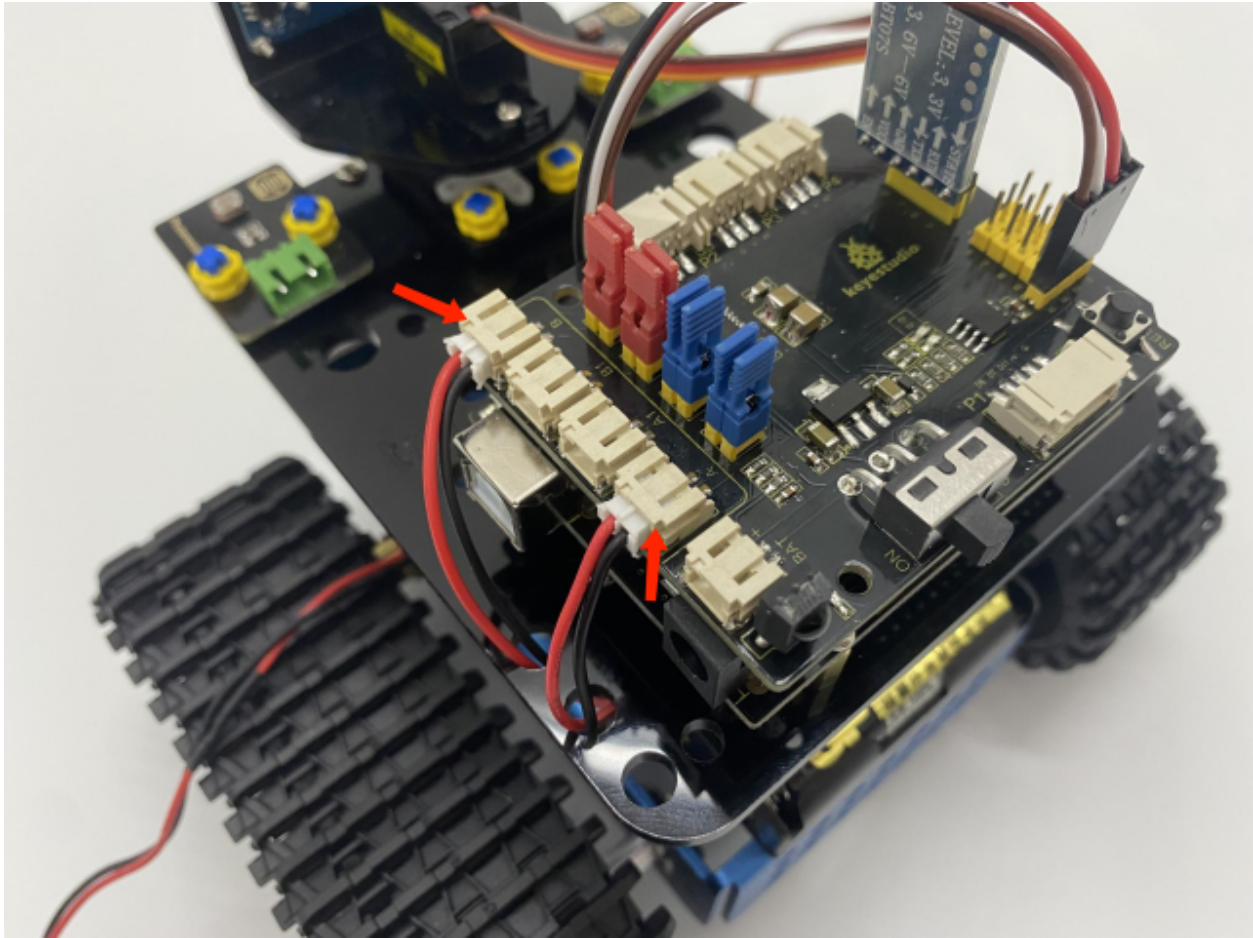




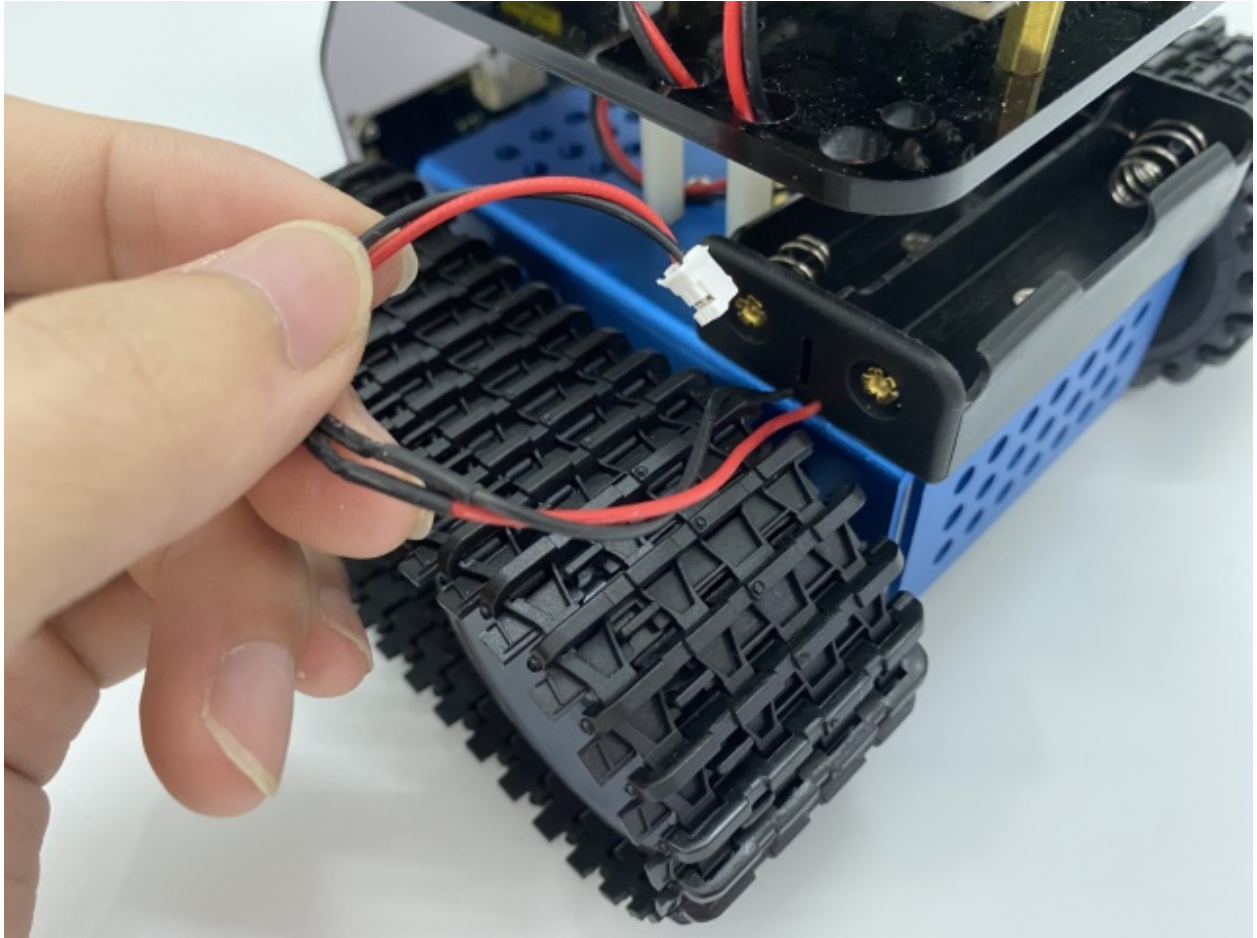
8*16 LED Panel	Keyestudio 8833 Motor Driver Board
GND	G
VCC	V
SDA	A4
SCL	A5

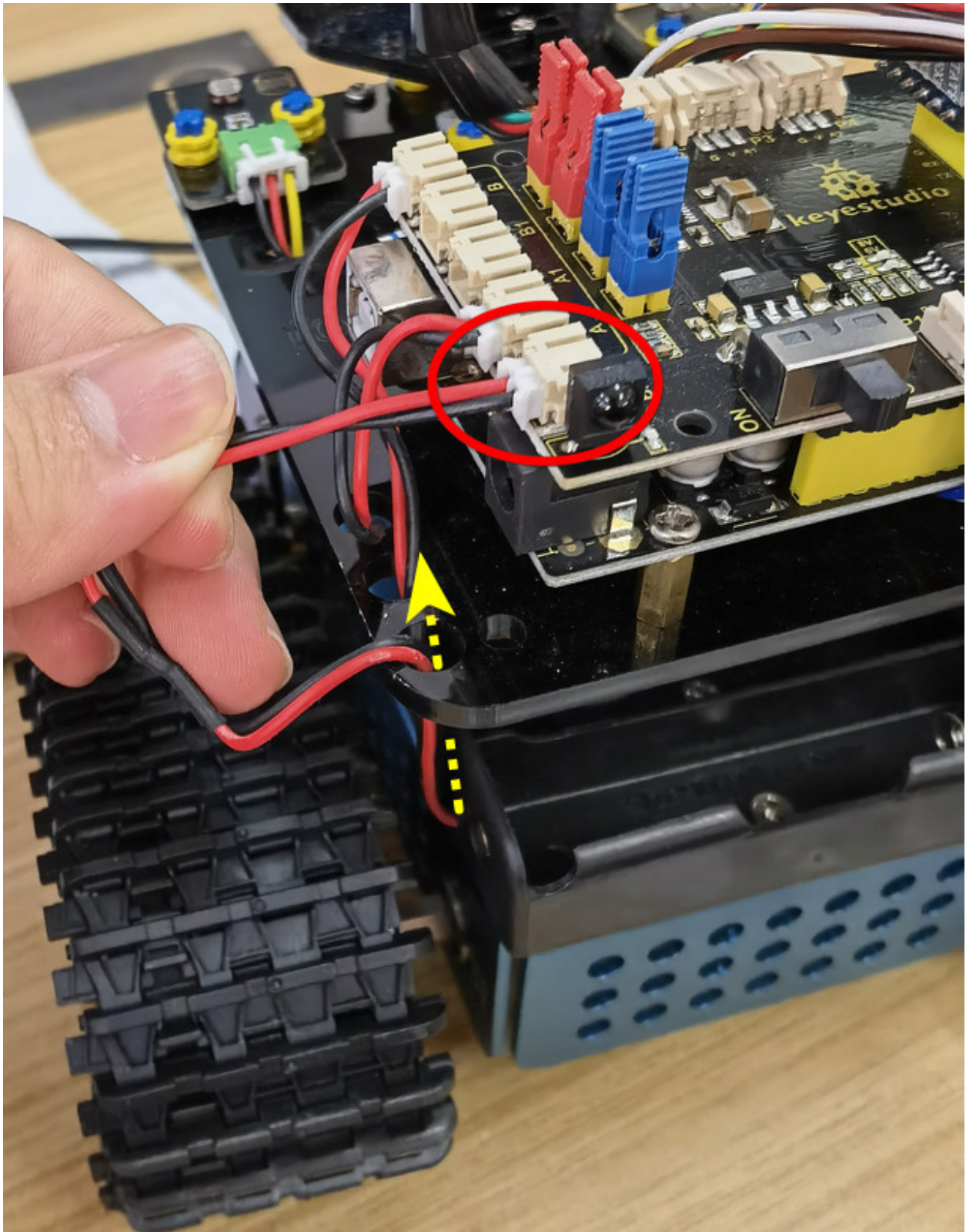
Connect the motor A to B port and make the motor B to A port.





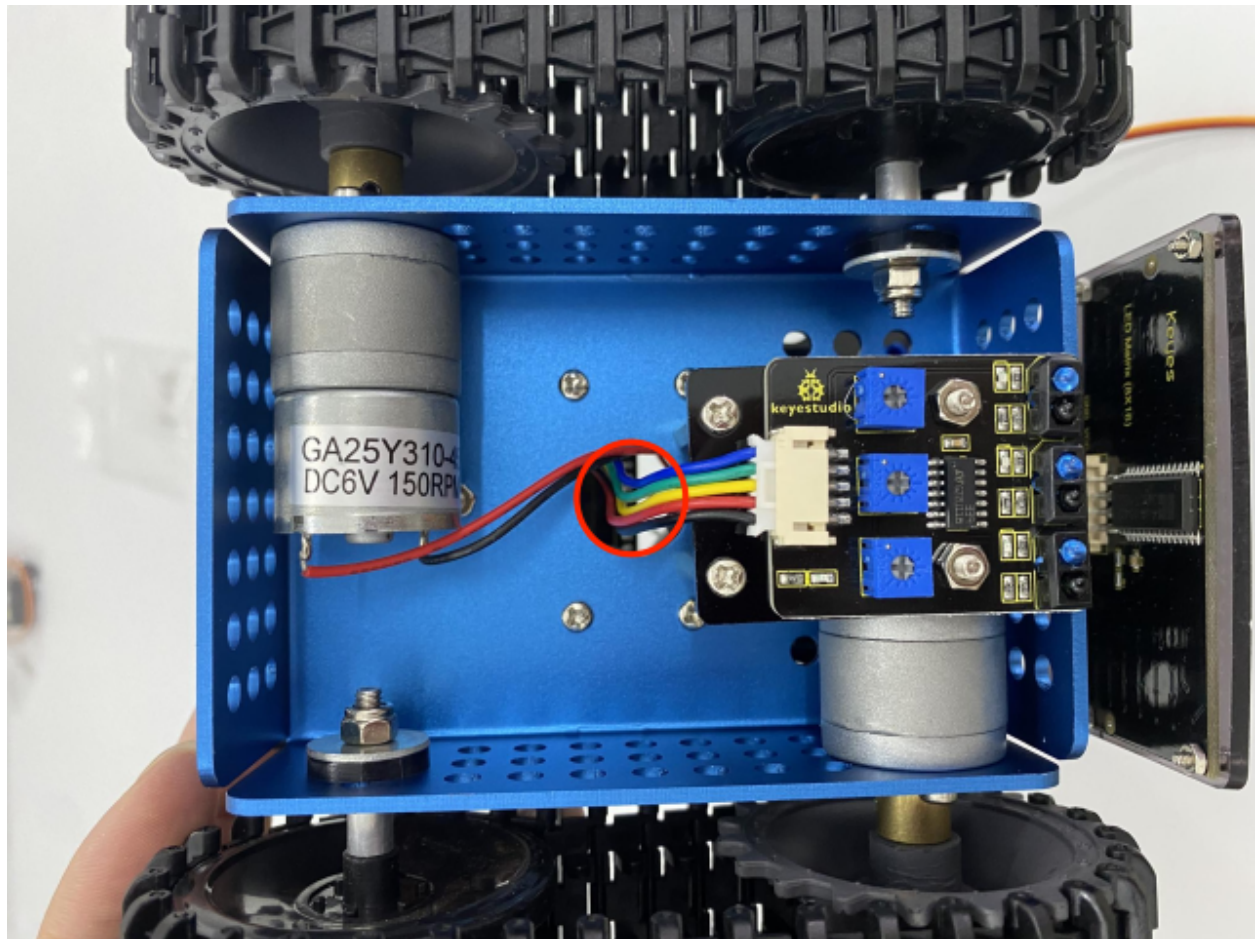
Connect the power wire

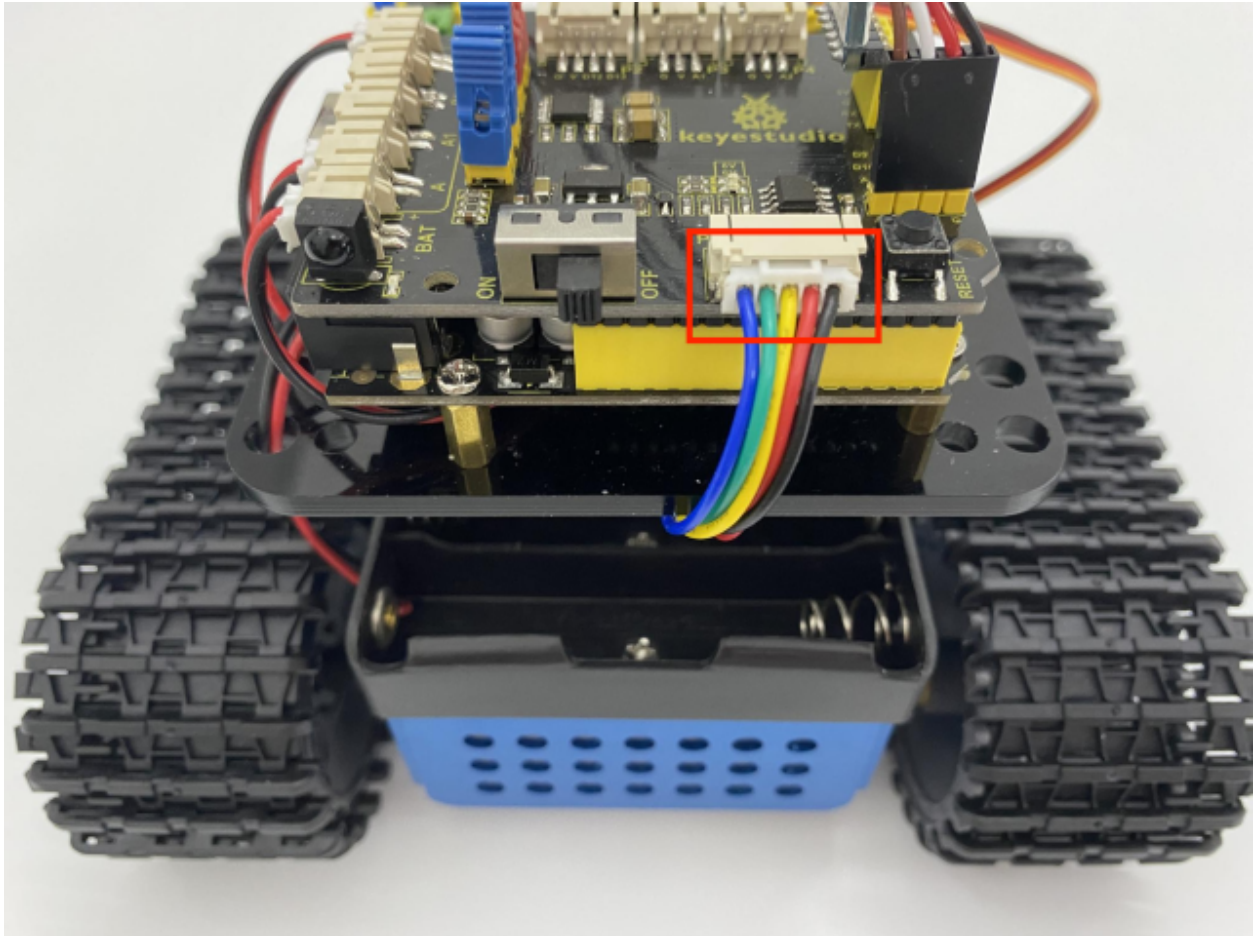




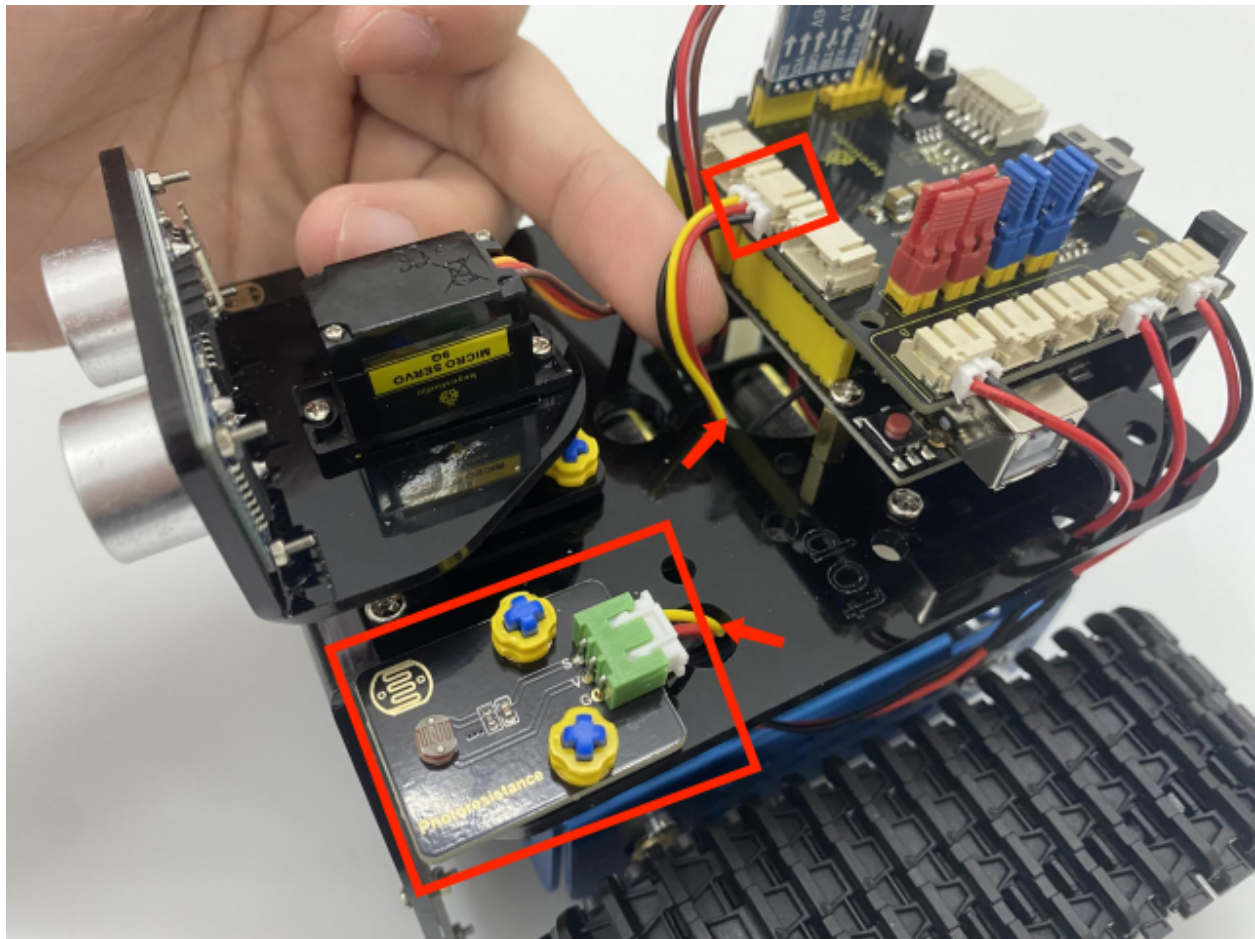
Line Tracking Sensor(see the picture)

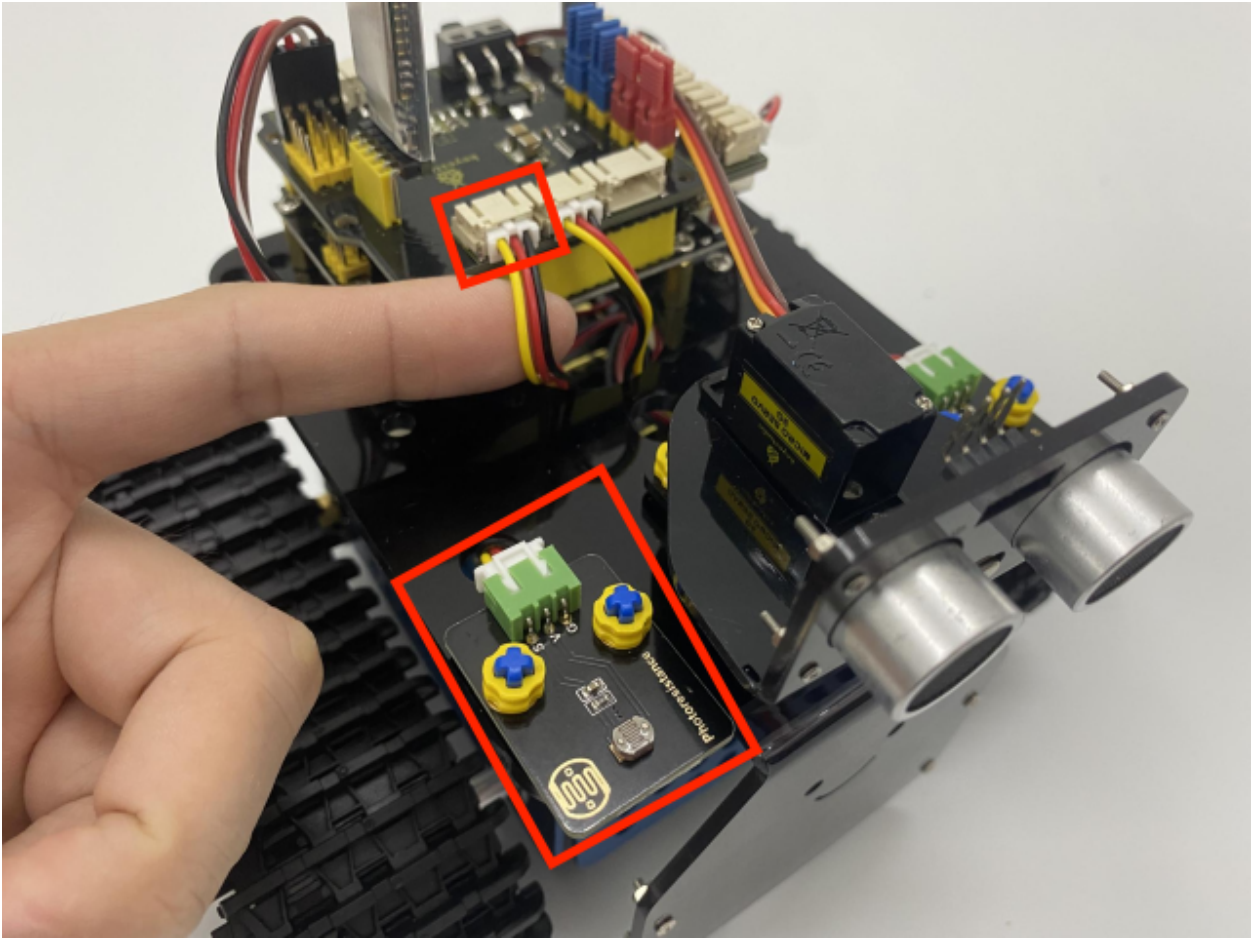






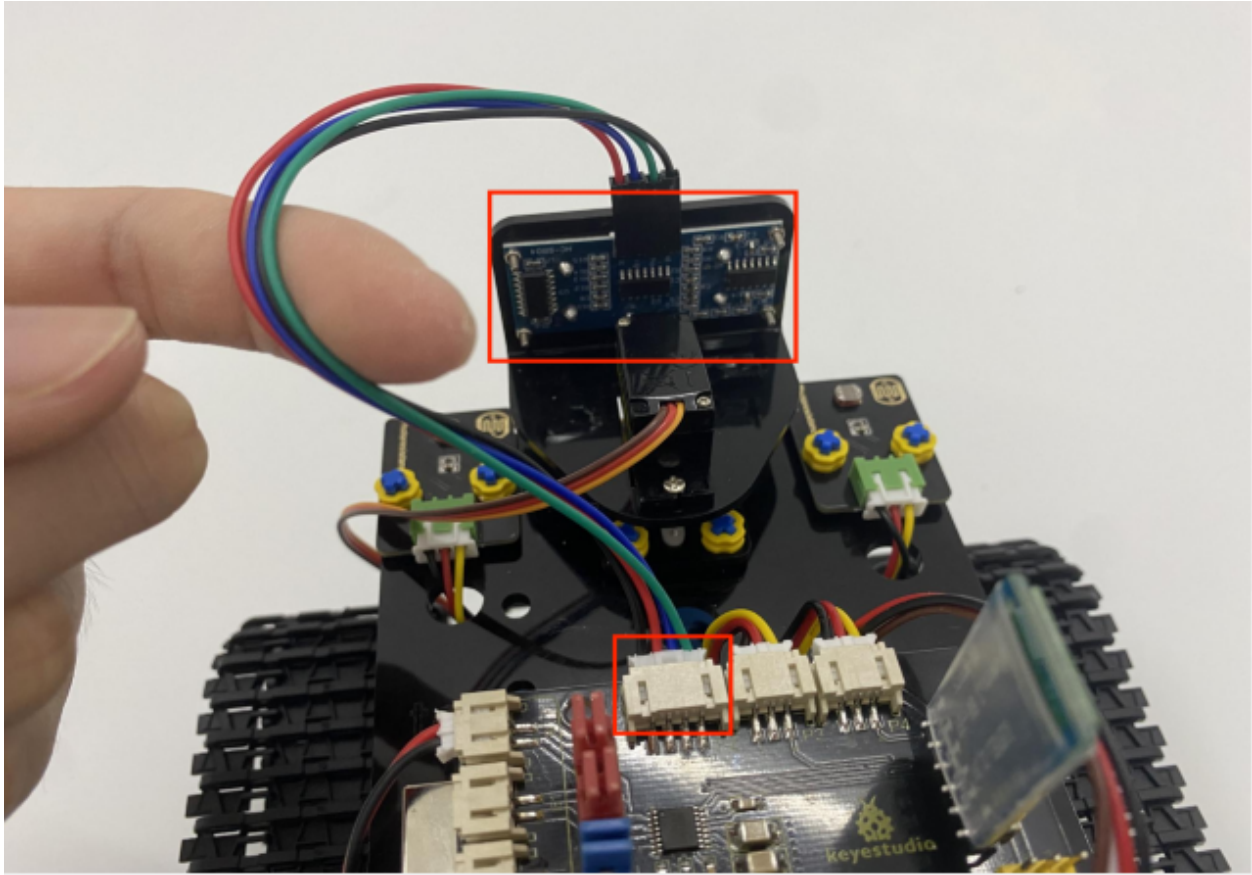
Wire up the photoresistors





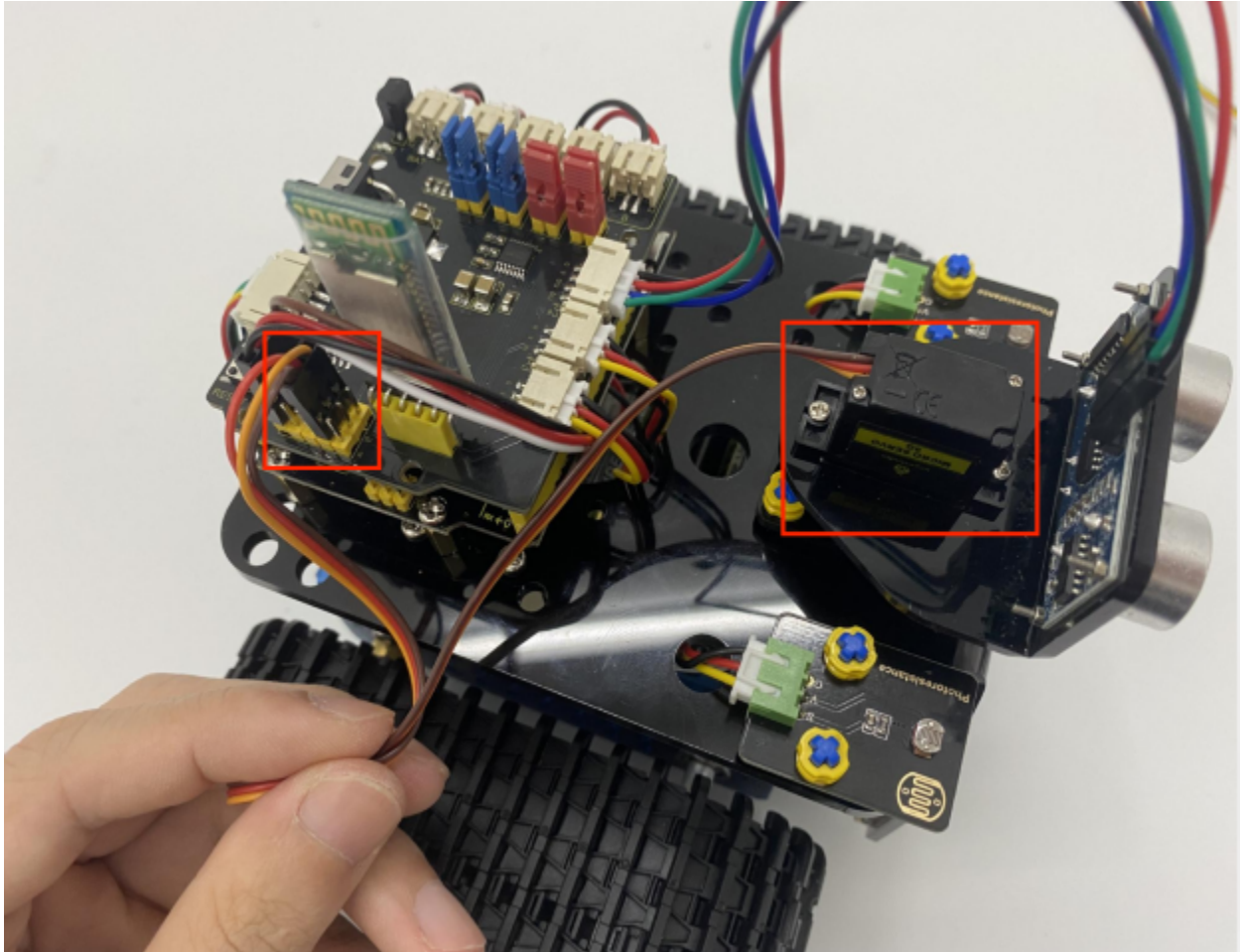
Left	Keyestudio 8833 Board	Right	Keyestudio 8833 Board
G	G	G	G
V	V	V	V
S	A1	S	A2

Wire up ultrasonic sensor



Ultrasonic Sensor	Keyestudio 8833 Board
Vcc	V
Trig	D12
Echo	D13
Gnd	G

Wire up the servo(D10)



Servo	Keyestudio 8833 Board
Brown	G
Red	V(5V)
Orange	D10

We adopt a model 18650 lithium battery with a pointed positive pole, whose power and capacity are not required.



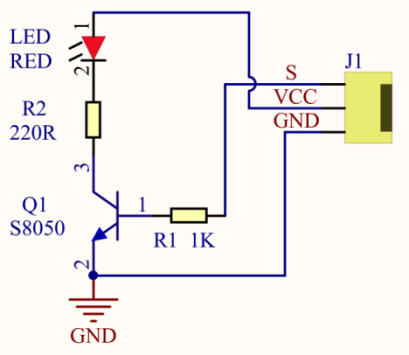
7.3 3. Projects

7.3.1 Project 1: LED Blinks

(1)Description

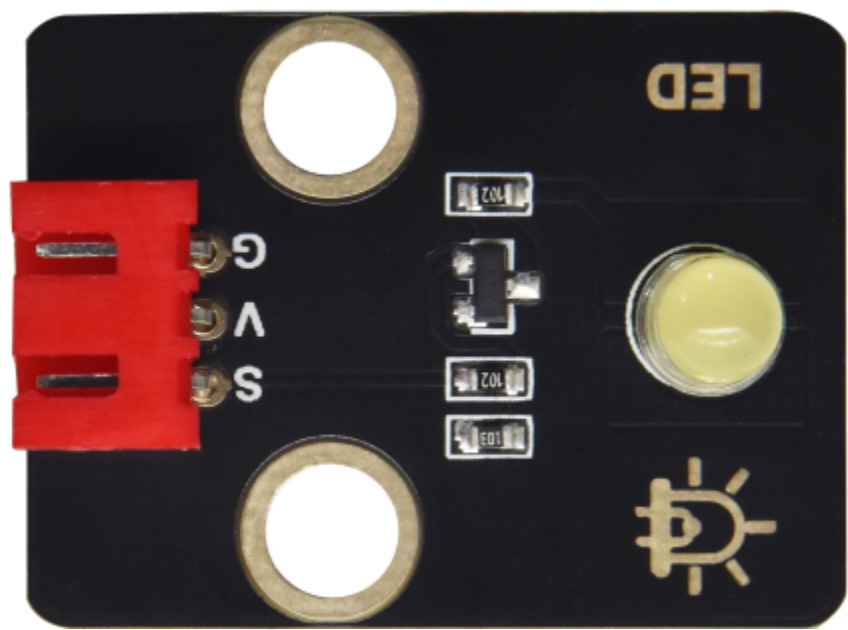


For starters and enthusiasts, LED Blink is a fundamental program. LED, the abbreviation of light emitting diodes, consists of Ga, As, P, N chemical compounds and so on. The LED can flash in diverse colors by altering the delay time in the test code. When in control, power on GND and VCC, the LED will be on if S end is in high level; nevertheless, it will go off.

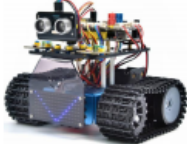






(2)Parameters:

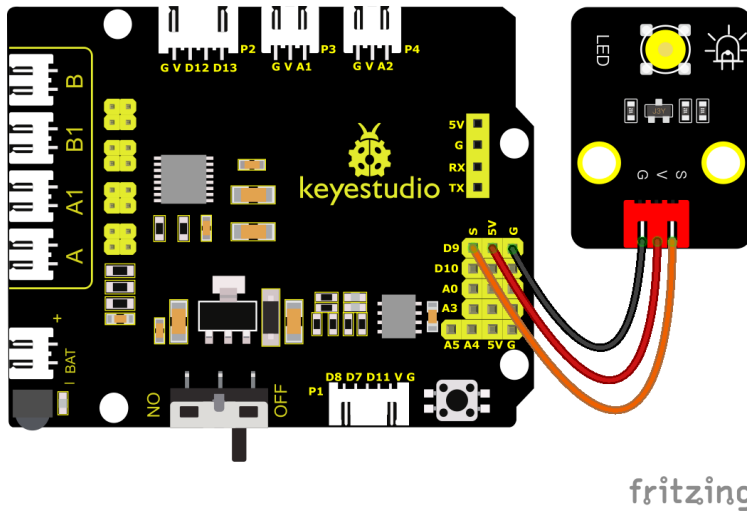
- Control interface: digital port
- Working voltage: DC 3.3-5V
- Pin spacing: 2.54mm
- LED display color: yellow



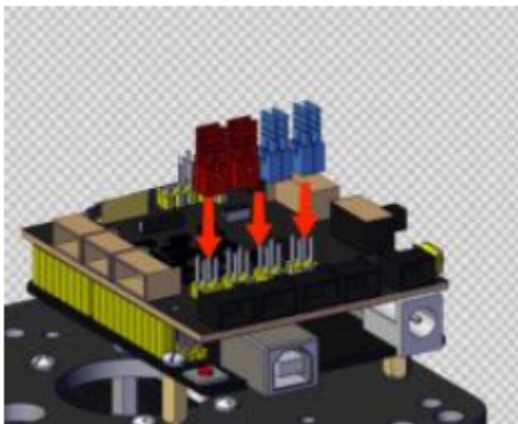
(3)Components Needed:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

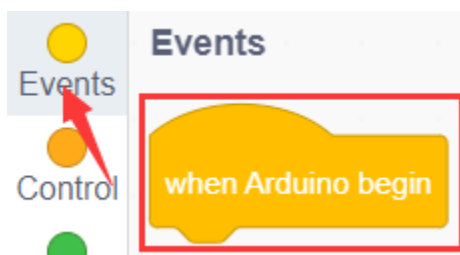
The Keystudio 8833 motor driver expansion board is compatible with the Arduino UNO development board. Just stack it onto the development board when using it.

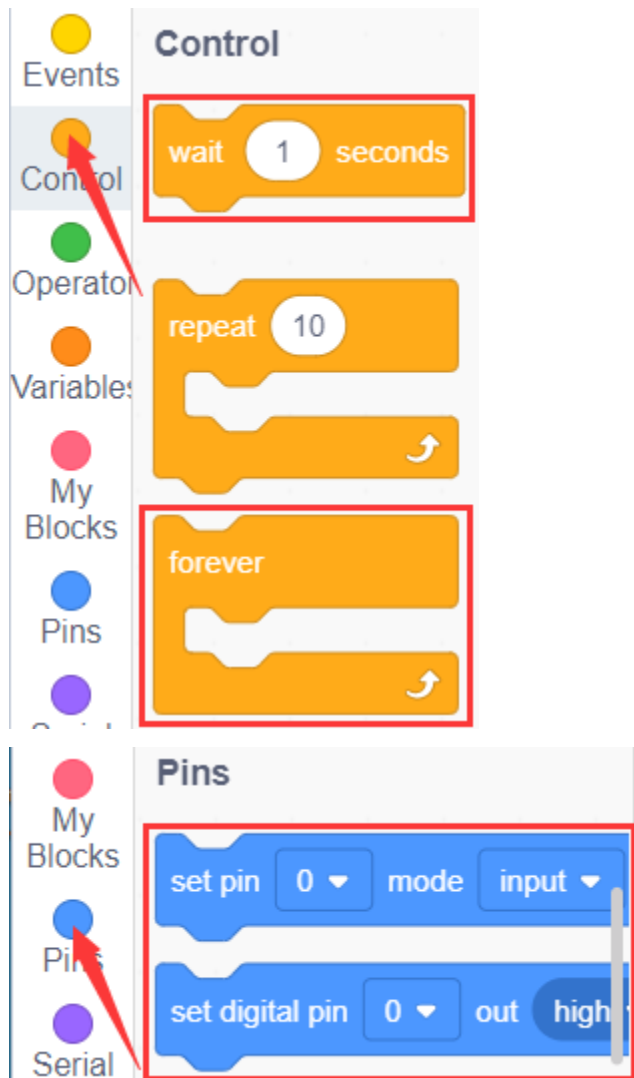
(4)8833 Motor Driver Expansion Board:**(5)Connection Diagram:**

LED is connected to D9 port, and remember to install jumper caps onto the shield.

**(6)Test Code:**

You can also drag blocks to edit your code, as shown below





Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

**(7)Test Results:**

Upload the program, LED blinks at the interval of 1s.

(8)Extension Practice:

We have known how to control the LED, then let's change the frequency of the LED.

We can the frequency of the LED without changing the pin of the LED. Let's modify the code.

You can also drag blocks to edit your code, as shown below

Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



The test result shows that the LED flashes faster. Therefore, we can draw a conclusion that pins and time delaying affect flash frequency.

7.3.2 Project 2: Adjust LED Brightness

(1)Description:

In previous lesson, we control LED on and off and make it blink.

In this project, we will control LED's brightness through PWM simulating breathing effect. Similarly, you can change the step length and delay time in the code so as to demonstrate different breathing effects.

PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output. In general, the input voltages of ports are 0V and 5V. What if the 3V is required? Or a switch among 1V, 3V and 3.5V? We cannot change resistors constantly. For this reason, we resort to PWM.

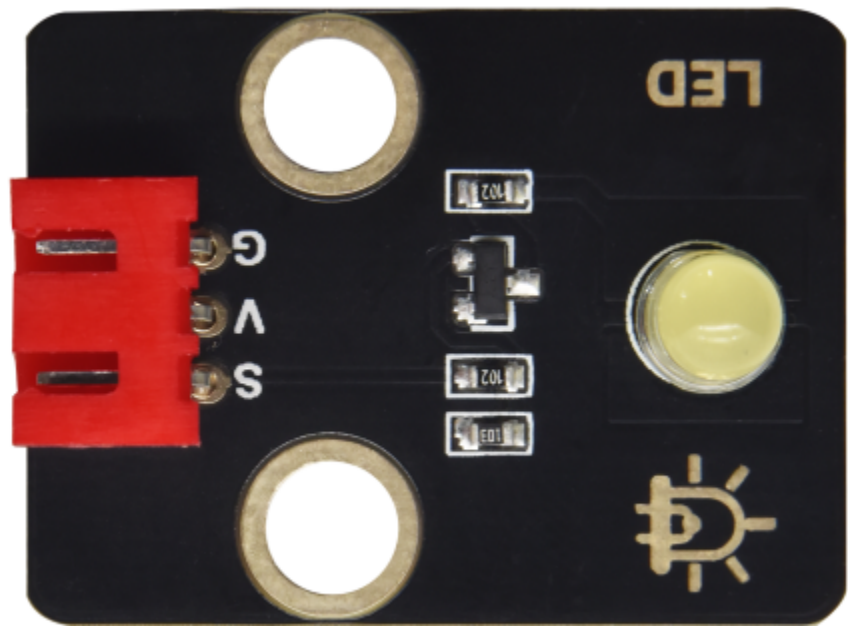
For Arduino digital port voltage outputs, there are only LOW and HIGH levels, which correspond to the voltage outputs of 0V and 5V respectively. You can define LOW as "0" and HIGH as "1", and let the Arduino output five hundred '0' or '1' within 1 second. If output five hundred '1', that is 5V; if all of which is '0', that is 0V; if output 250 01 pattern, that is 2.5V.

This process can be likened to showing a movie. The movie we watch are not completely continuous. Actually, it generates 25 pictures per second, which cannot be told by human eyes. Therefore, we mistake it as a continuous

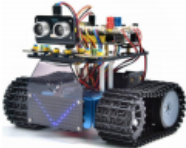




process. PWM works in the same way. To output different voltages, we need to control the ratio of 0 and 1. The more ‘0’ or ‘1’ output per unit time, the more accurate the control.

(2)Parameters:

- Control interface: Digital port 3
- Working voltage: DC 3.3-5V
- Pin spacing: 2.54mm
- LED display color: yellow



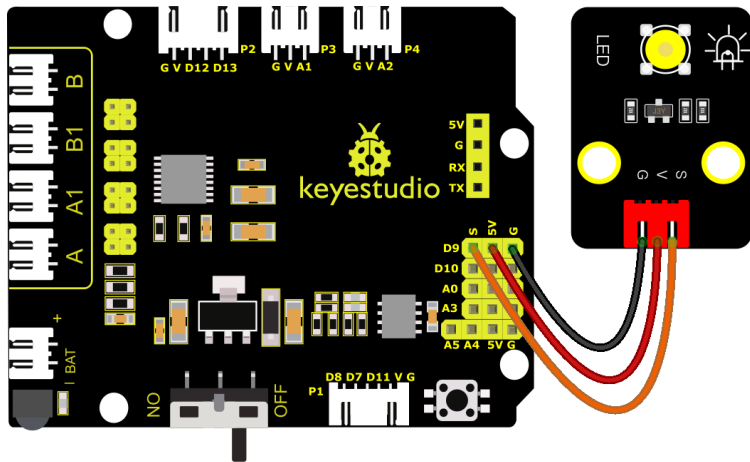
(3)Components Needed:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

(4)Wiring Diagram

PWM pins of the Arduino are pin 3, 5, 6, 9, 10 and 11

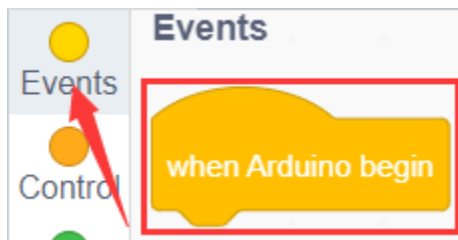
Keep the D9 unchanged

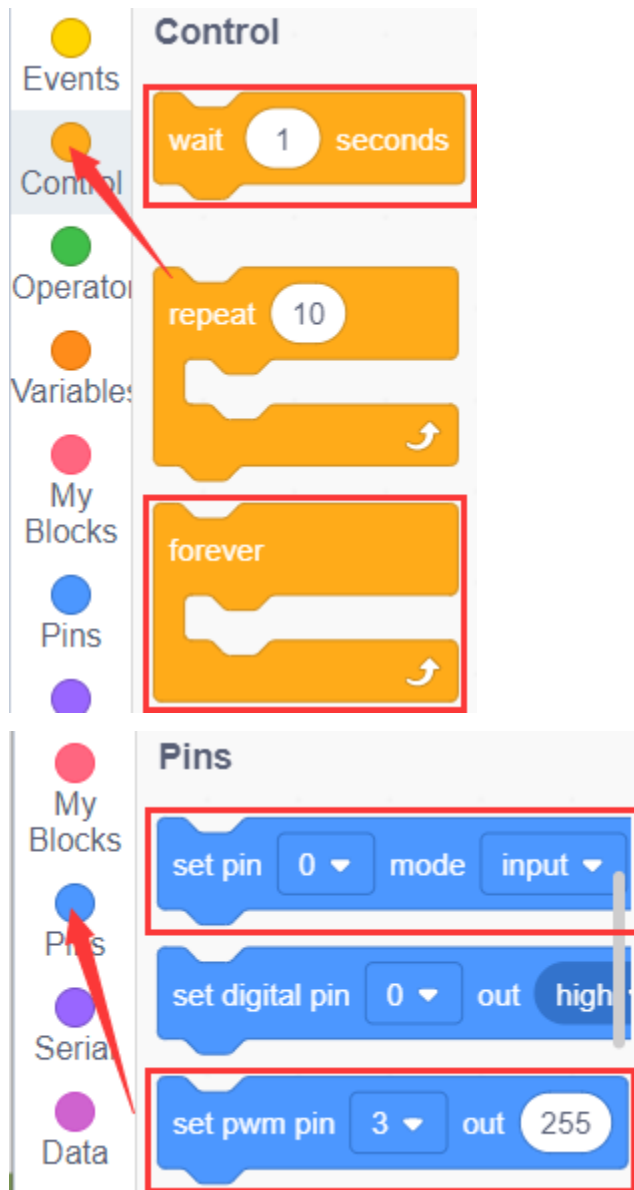


fritzing

(5)Test Code:

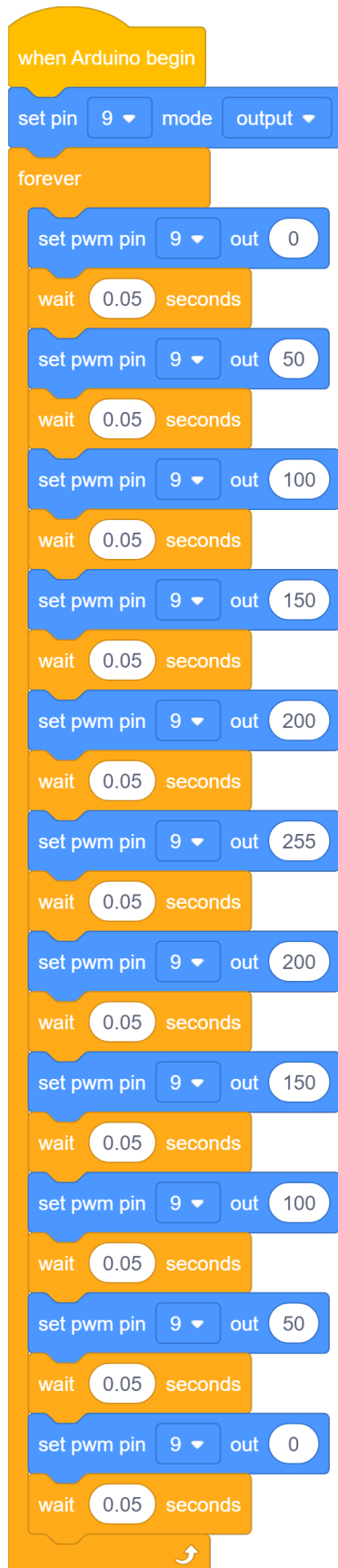
You can also drag blocks to edit your code, as shown below





Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Results

Upload test code successfully, LED gradually changes from bright to dark, like human's breath, rather than turning on and off immediately.

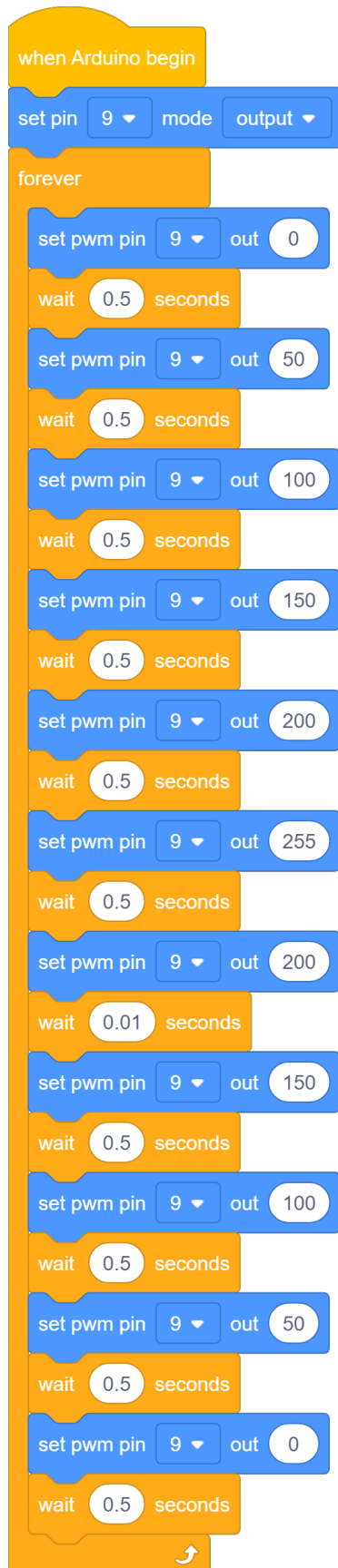
(7)Extension Practice:

You can not change the position of the lamp pin, just by changing the program code (the value after wait)

You can also drag blocks to edit your code, as shown below

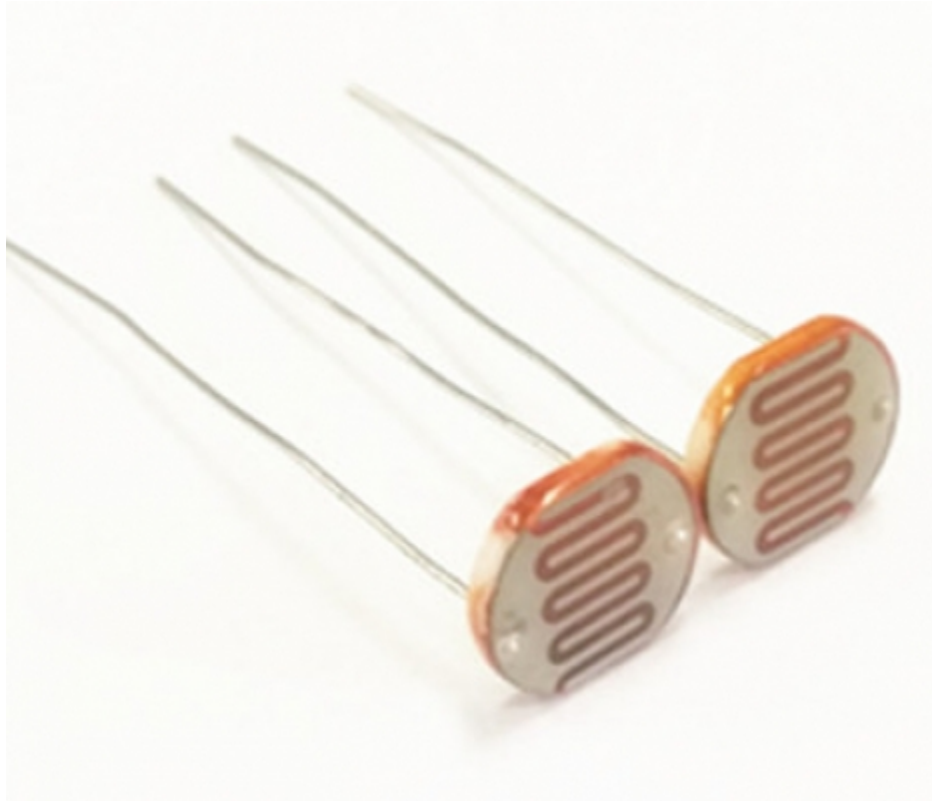
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



Upload the code, observe the LED

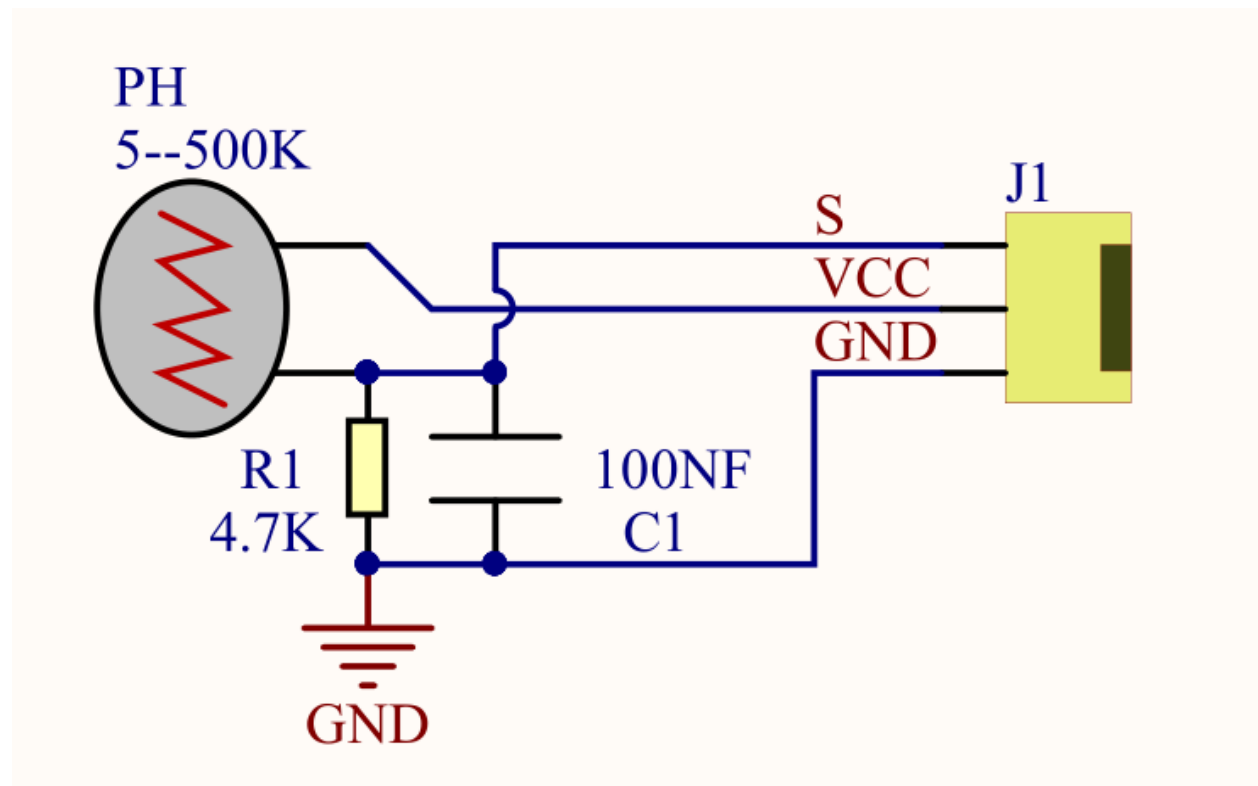
7.3.3 Project 3: Photoresistor



(1)Description:

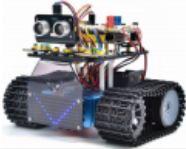



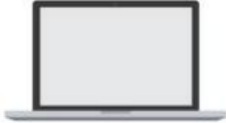
The photosensitive resistor is a special resistor made of a semiconductor material such as a sulfide or selenium, and a moisture-proof resin is also coated with a photoconductive effect. The photosensitive resistance is most sensitive to the ambient light, different illumination strength, and the resistance of the photosensitive resistance is different. We use the photosensitive resistance to design the photosensitive resistor module. The module signal is connected to the microcontroller analog port.

When the light intensity is stronger, the larger the analog port voltage, that is, the simulation value of the microcontroller is also large; in turn, when the light intensity is weaker, the smaller the analog port voltage, that is, the simulation value of the microcontroller is also small. . In this way, we can read the corresponding analog value using the photosensitive resistor module, and the intensity of the light in the inductive environment.

**(2)Parameters:**

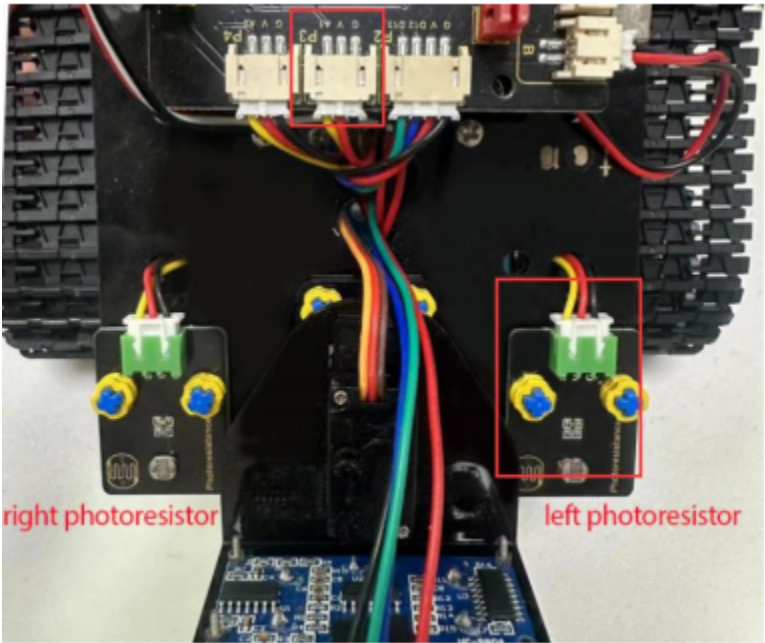
- Photosensitive resistance resistance value: 5K Ou-0.5m
- Interface type: simulation port A0, A1
- Working voltage: 3.3V-5V
- Pin spacing: 2.54mm

(3)Components Needed:

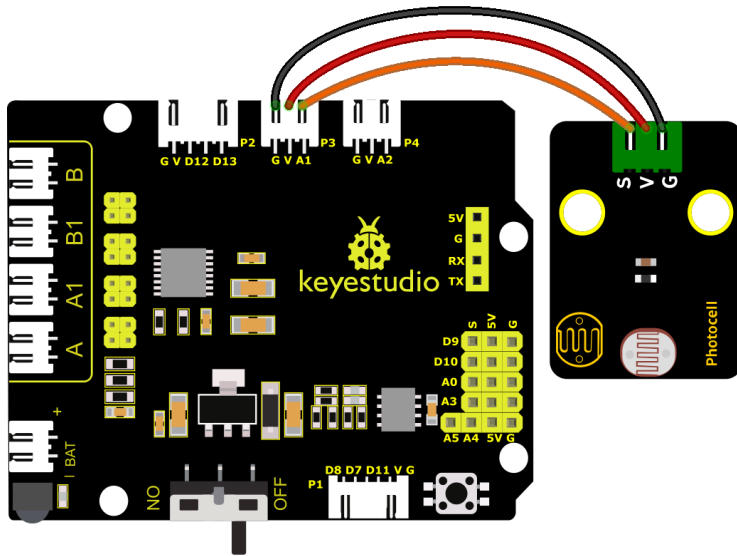
Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

(4)Connection Diagram:

What we are going to test next is the photoresistor module on the left side of the robot



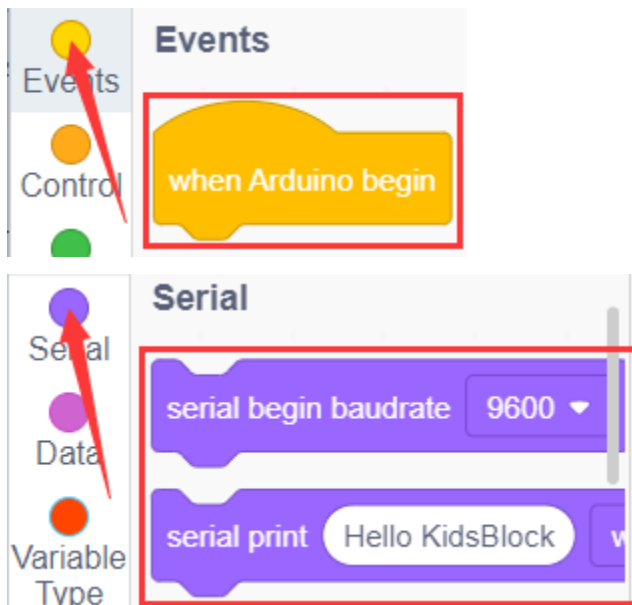
The left photoresistor is connected to A1/P3 of the motor drive shield.

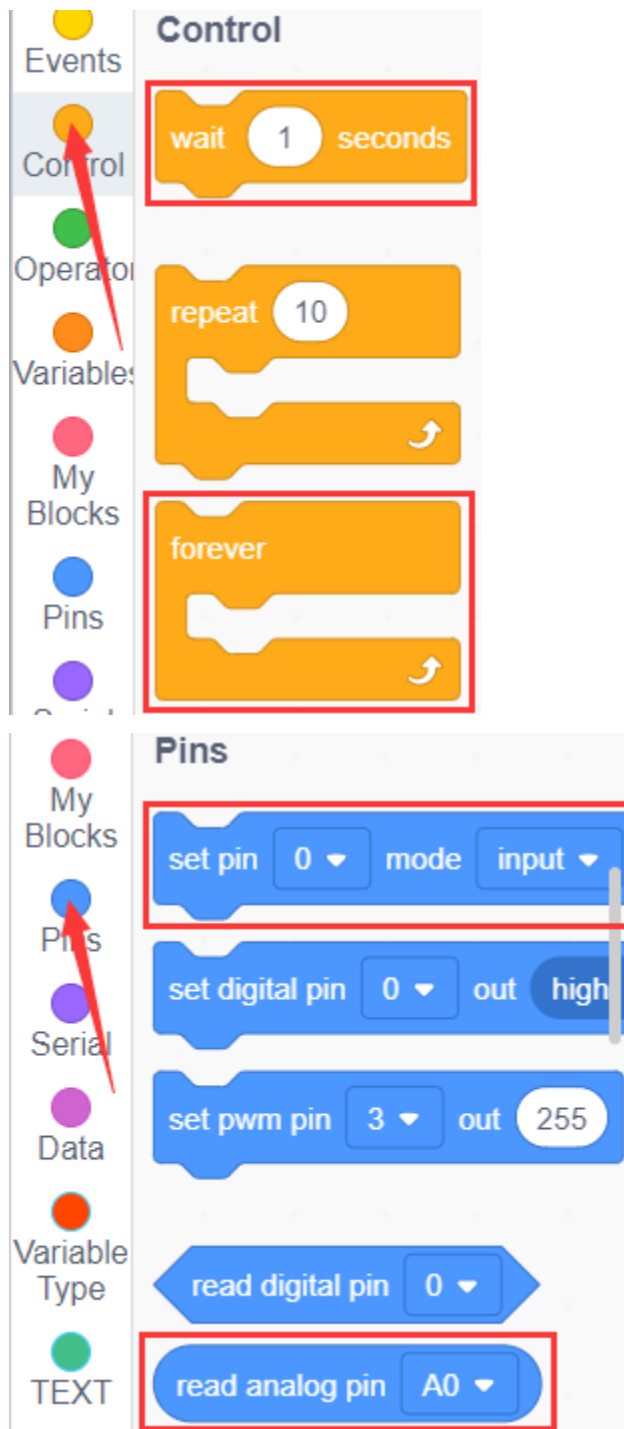


fritzing

(5)Test Code:

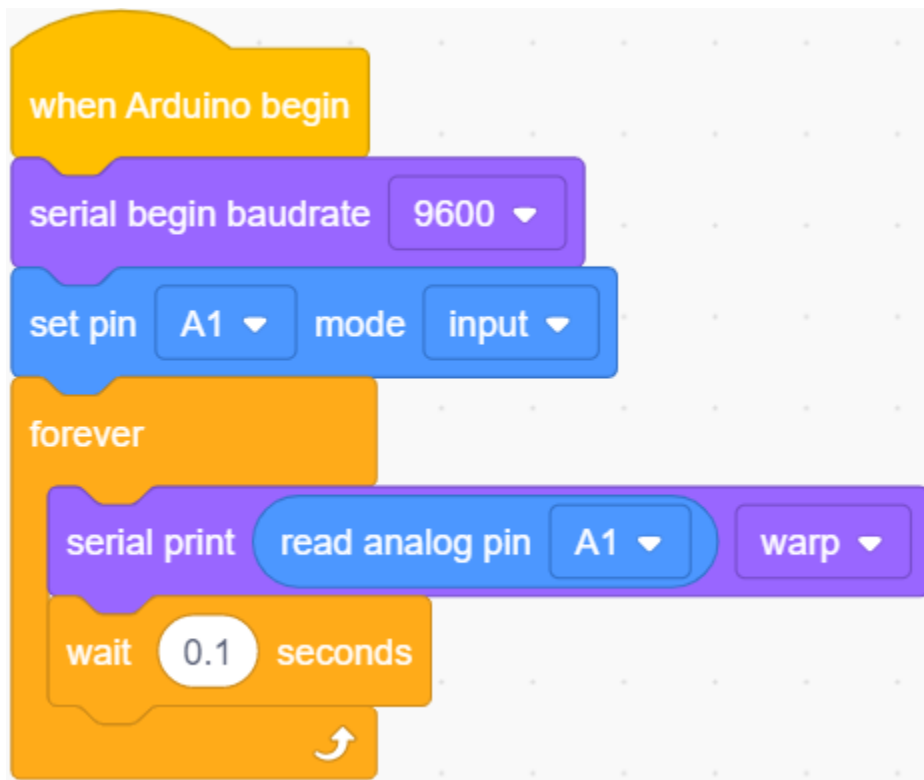
You can also drag blocks to edit your code, as shown below






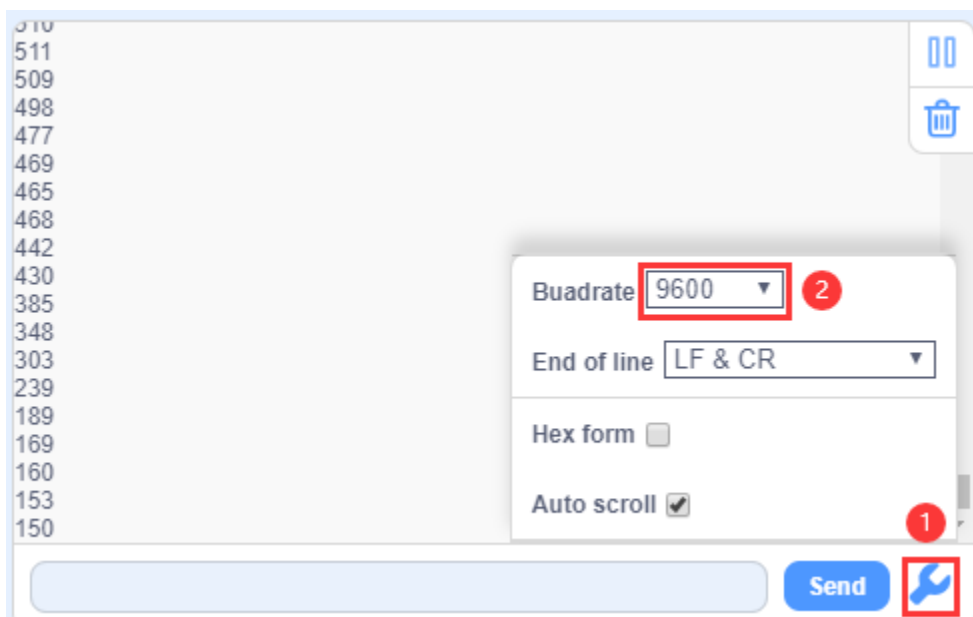
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



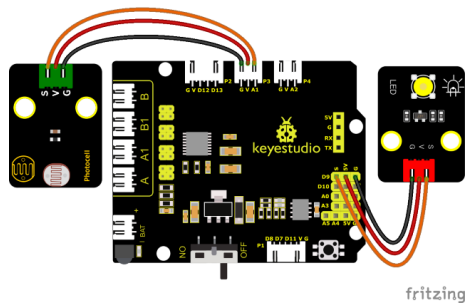
(6)Test Results:

Upload the code to the development board. Click  to set baud rate 9600. When covering it with your hand, the value gets smaller; if not, the value gets larger.



(7)Extension Practice:

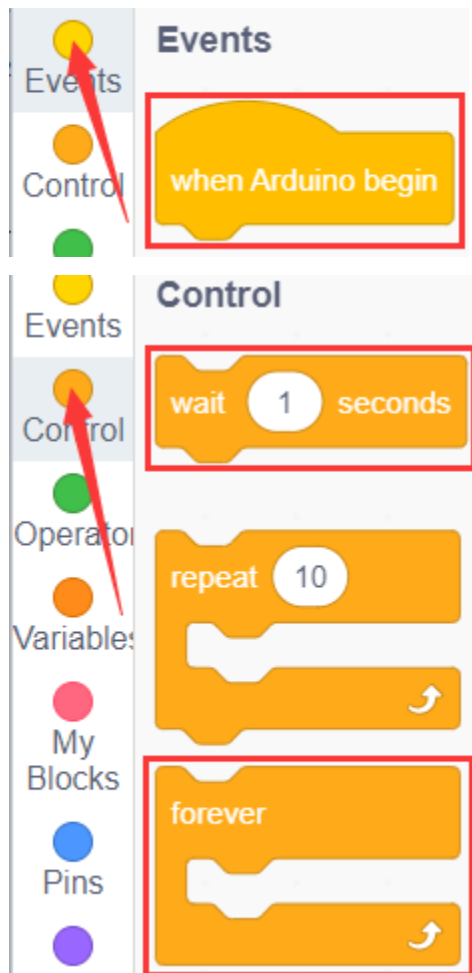
The above code just reads the value of the photoresistor. We can make the photosensitive and LED combine to change the LED.



PWM can change the light brightness, that is, LED should be connected to the PWM of the development board

Connect the LED to D9 and keep other pins unchanged, then we edit code.

You can also drag blocks to edit your code, as shown below



The screenshot shows the Kidsblock IDE interface. On the left sidebar, the 'Pins' category is selected, indicated by a red arrow. The 'Pins' section contains the following blocks:

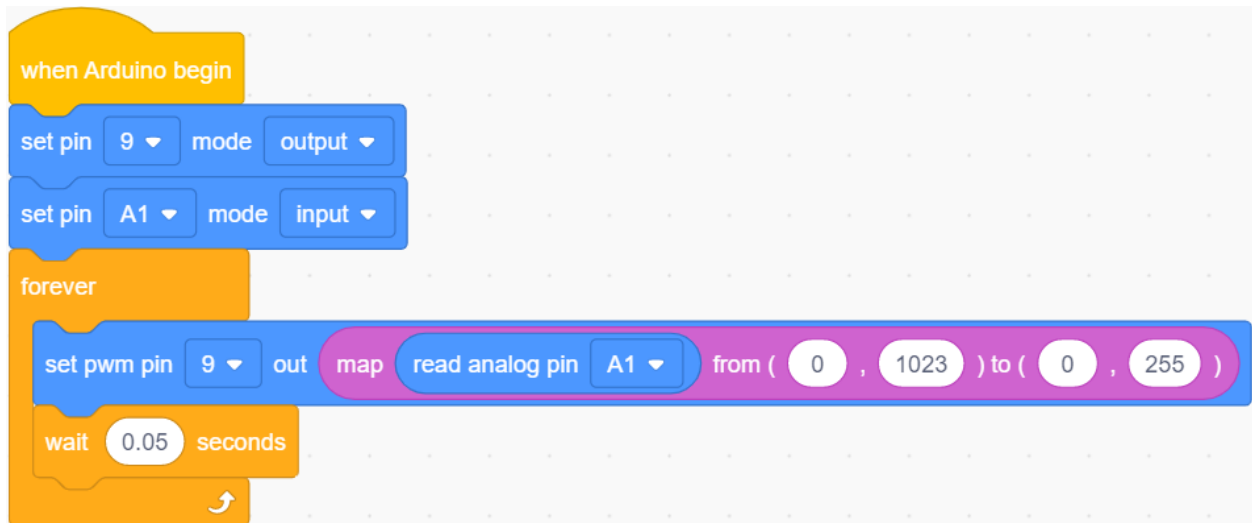
- set pin 0 mode input
- set digital pin 0 out high
- set pwm pin 3 out 255
- read digital pin 0
- read analog pin A0

The 'Data' section contains the following blocks:

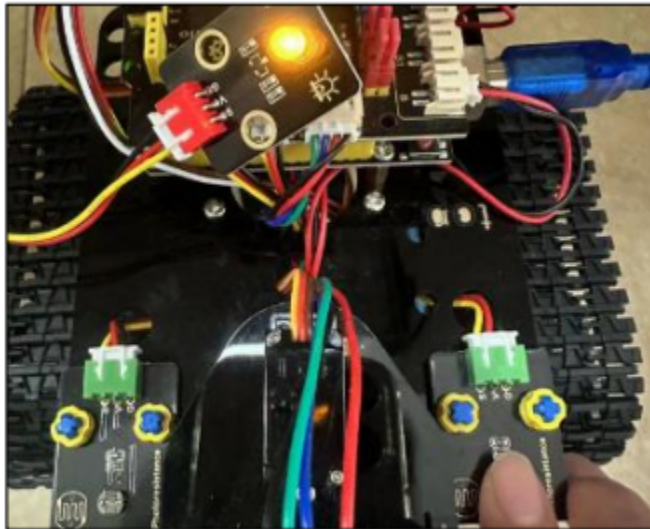
- map 50 from (1 , 100) to (1 , 1000)
- constrain 50 between (1 , 100)

Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



Upload the code to the development board, we press the photoresistor to see if the brightness of the LED light has changed.



7.3.4 Project 4: Line Tracking Sensor

(1)Description:



The tracking sensor is actually an infrared sensor. The component used here is the TCRT5000 infrared tube.

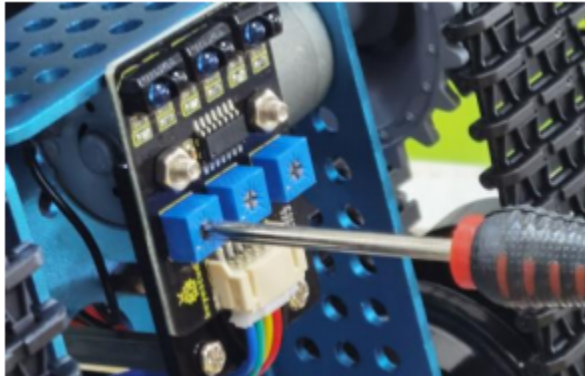
Mini Tank Robot

Its working principle is to use different reflectivity of infrared light to colors, then convert the strength of the reflected signal into a current signal.

During the process of detection, black is active at HIGH level while white is active at LOW level. The detection height is 0-3 cm.

Keyestudio 3-channel line tracking module has integrated 3 sets of TCRT5000 infrared tube on a single board, which is more convenient for wiring and control.

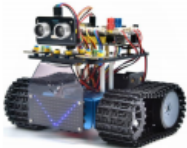




If the Line Tracking Sensor does not work as expected, you will need to use a screwdriver to adjust its potentiometer to make it more sensitive. When your finger is close to the sensor, its on-board LED light turns on, and when your finger moves away, its on-board LED light turns off. At this time, its sensitivity is relatively good.

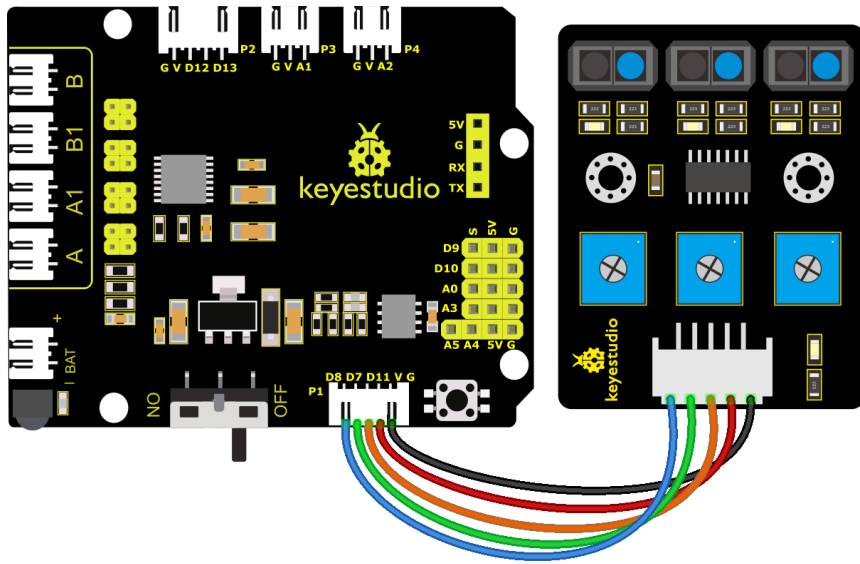


(2)Parameters:

- Operating Voltage: 3.3-5V (DC)
- Interface: 5PIN
- Output Signal: Digital signal
- Detection Height: 0-3 cm

(3)Components Required:

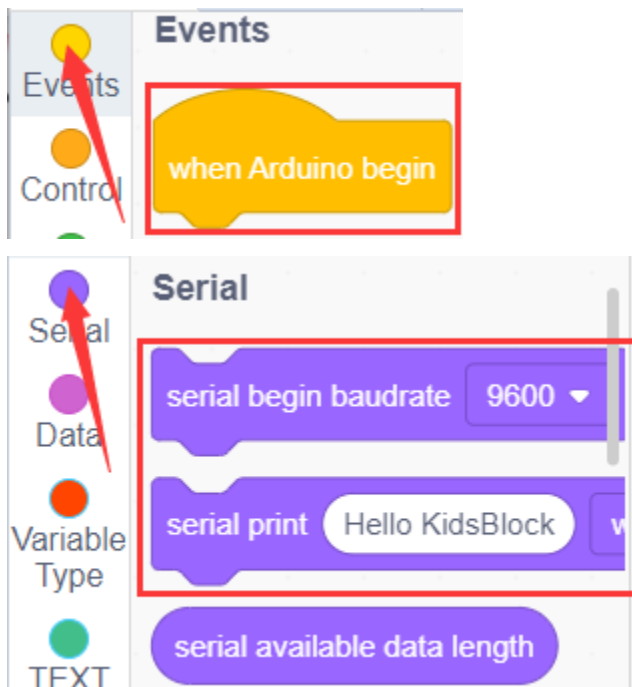
Robot without BT Module*1	USB Cable*1	Yellow LED Module*1
		
3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1	
		

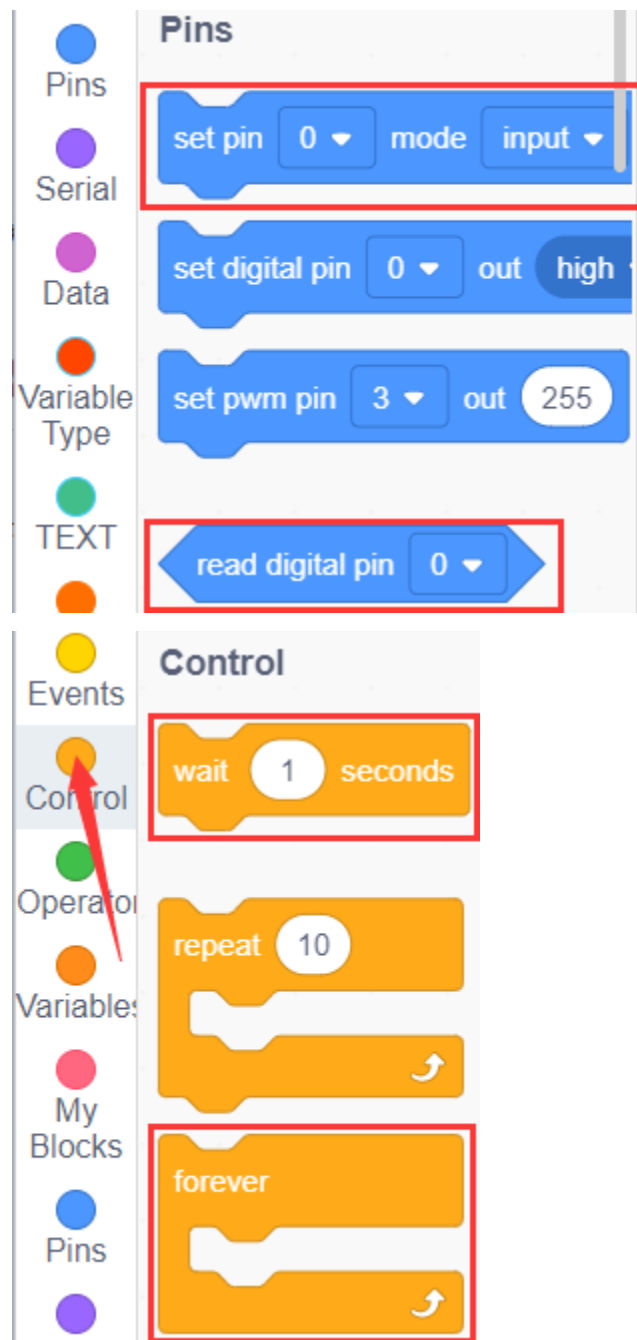
(4)Connection Diagram:

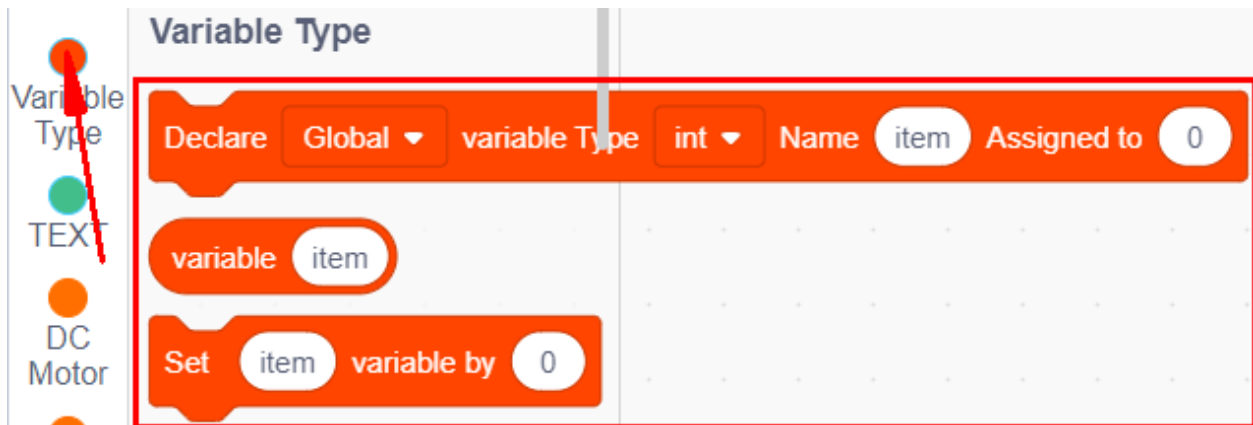
fritzing

(5)Test Code:

You can also drag blocks to edit your code, as shown below

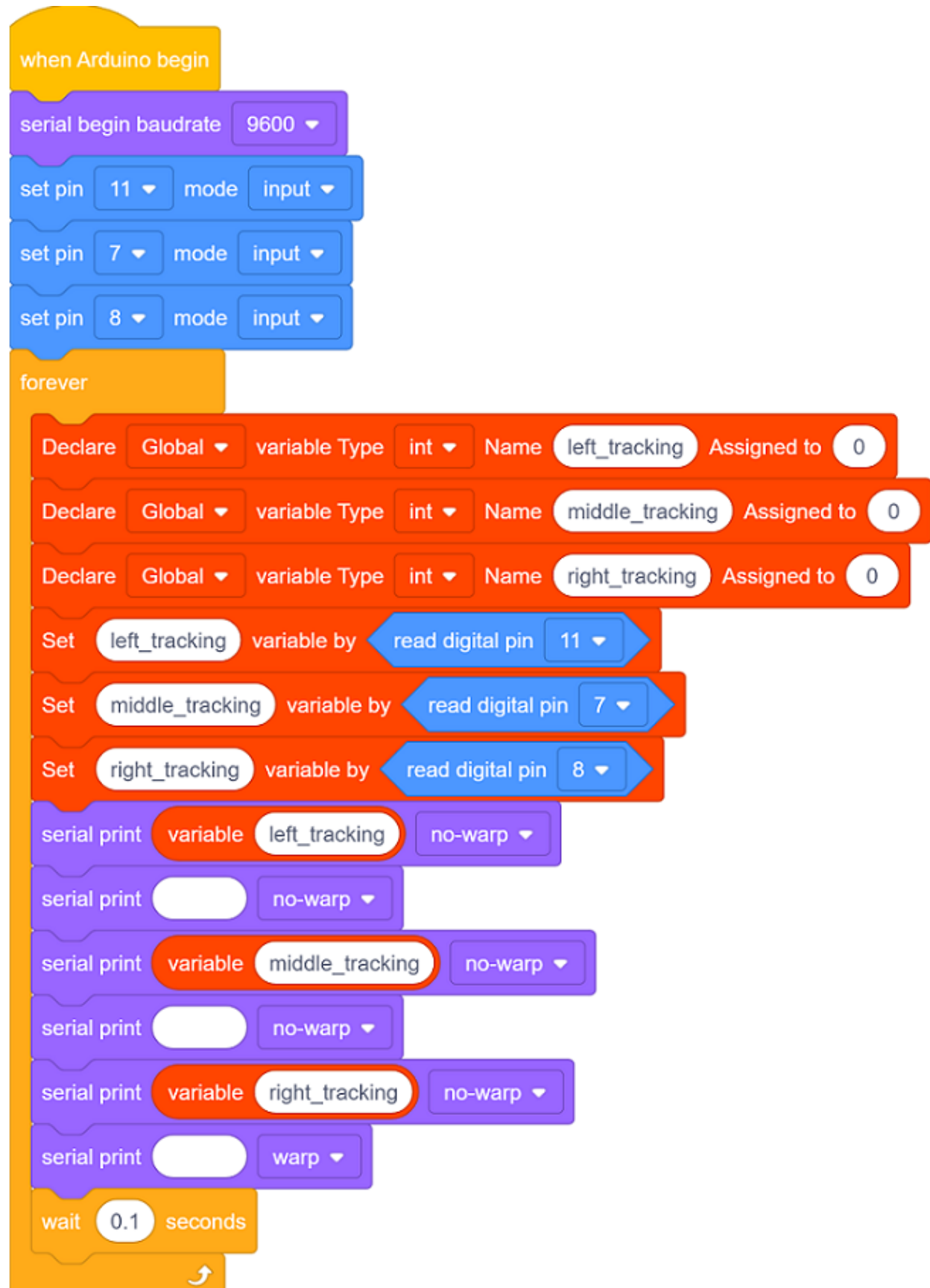






Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

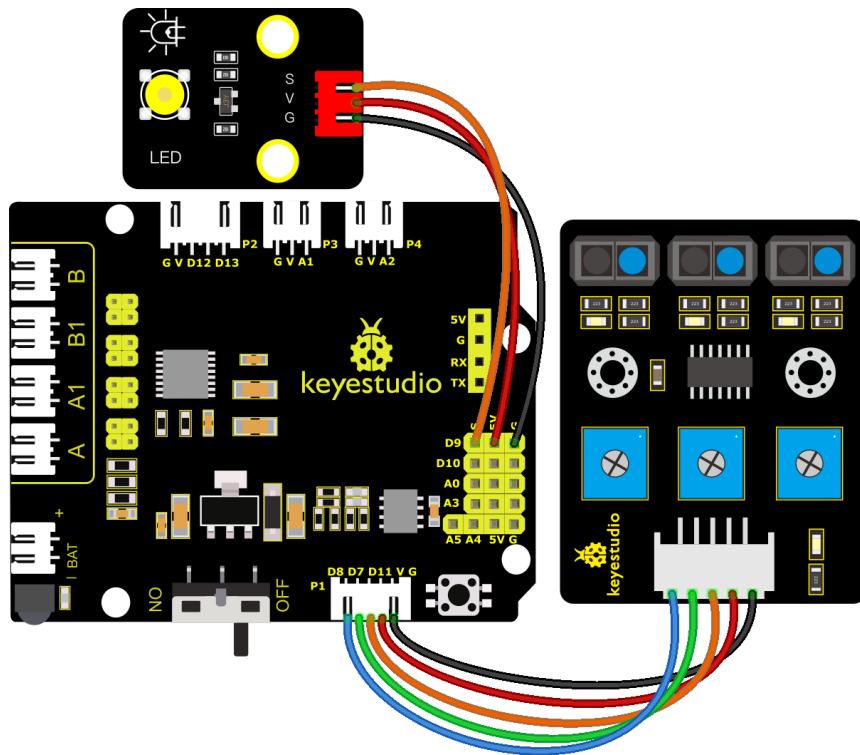


(6)Test Results:

Upload the code to the development board, open serial monitor to 9600 and check line tracking sensors. And the displayed value is 1(high level) when no signals are received. The value shifts into 0 when the sensor is covered with paper.

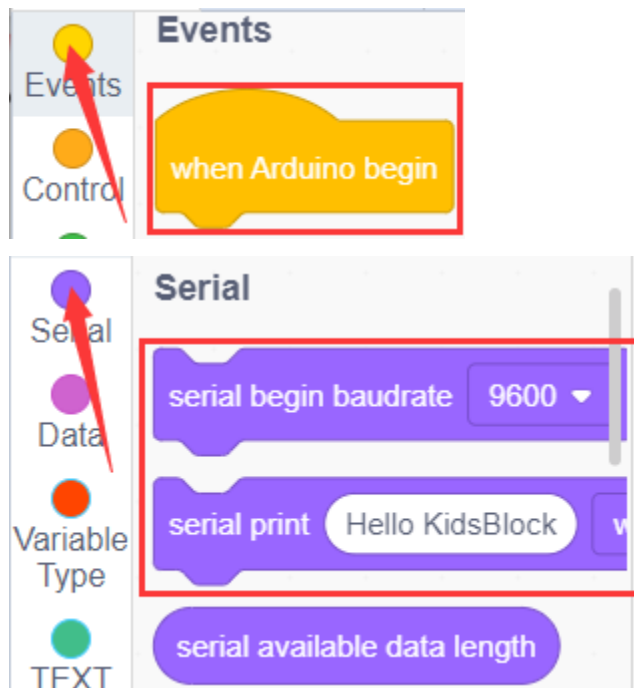
**(7)Extension Practice:**

We can control an LED with this sensor. The LED is connected to D9. If we cover it , the LED will light up.



fritzing

You can also drag blocks to edit your code, as shown below



The image shows the Scratch IDE interface with the 'Pins' and 'Variable Type' categories selected in the left sidebar. The 'Pins' category contains three blue blocks: 'set pin 0 mode input', 'set digital pin 0 out high', and 'set pwm pin 3 out 255'. The 'Variable Type' category contains an orange 'forever' loop block, an orange 'if-then-else' conditional block, and an orange 'Set item variable by 0' block. Red boxes highlight these blocks, and red arrows point to the 'Control' and 'Variable Type' categories in the sidebar.

Pins

- set pin 0 mode input
- set digital pin 0 out high
- set pwm pin 3 out 255

Control

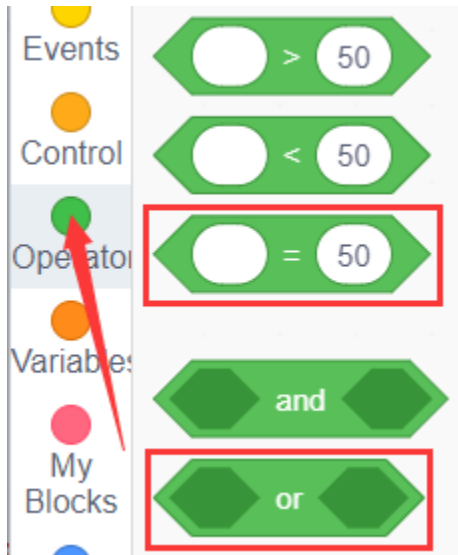
- forever

Variable Type

- Declare Global variable Type int Name item Assigned to 0
- variable item
- Set item variable by 0

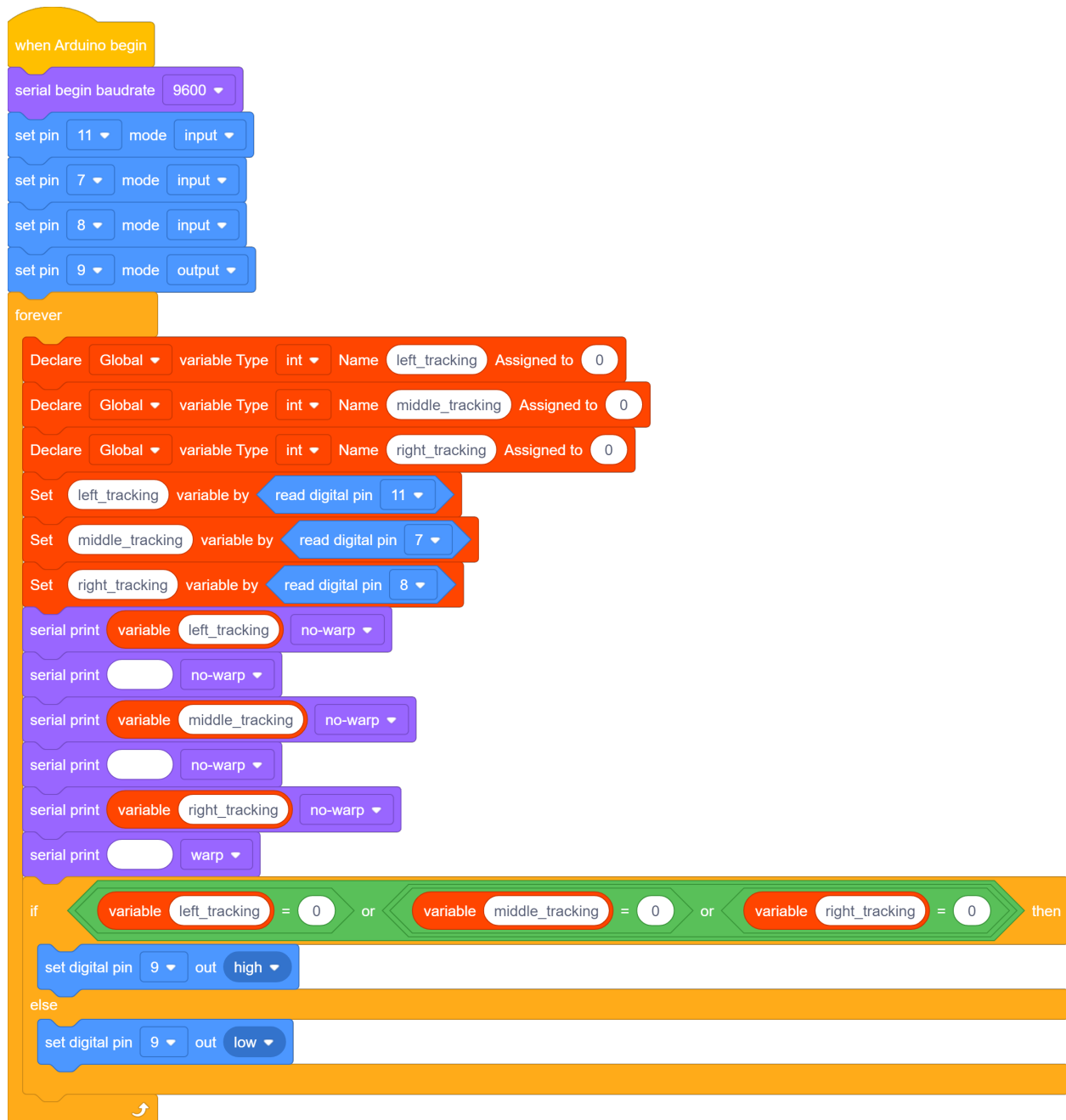
Control

- if then else



Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



When an object (such as paper or finger) approaches the line-following sensor, the sensor detects the return signal emitted by itself, and the LED module lights up. When the sensor does not detect any return signal, and the LED module turns off.



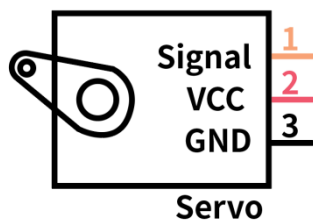
7.3.5 Project 5: Servo Control

(1)Description:

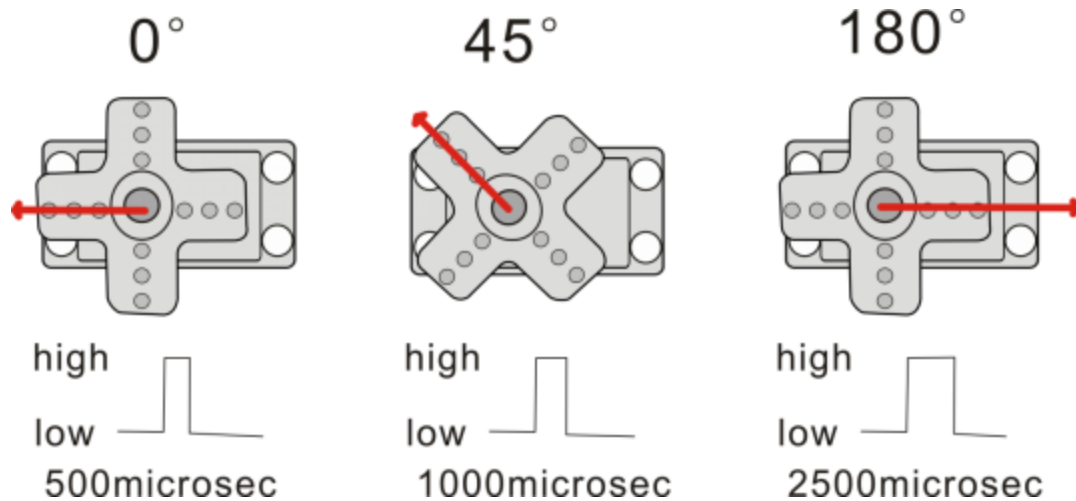
Servo motor is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor. Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of 1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and obtain the voltage difference output.

When the motor speed is constant, the potentiometer is driven to rotate through the cascade reduction gear, which leads that the voltage difference is 0, and the motor stops rotating. Generally, the angle range of servo rotation is 0° – 180°

The rotation angle of servo motor is controlled by regulating the duty cycle of PWM (Pulse-Width Modulation) signal. The standard cycle of PWM signal is 20ms (50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds the rotation angle from 0° to 180° . But note that for different brand motors, the same signal may have different rotation angles.



In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.



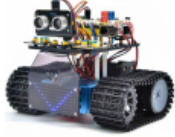

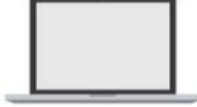

The angle of the servo:

High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree

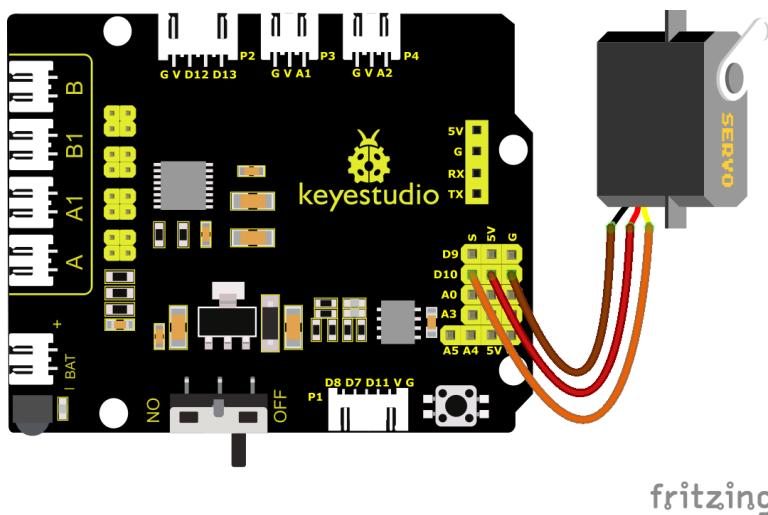
(2)Parameters:

- Working voltage: DC 4.8V ~ 6V
- Operating angle range: about 180 ° (at 500 → 2500 sec)
- Pulse width range: 500 → 2500 sec
- No-load speed: 0.12 ± 0.01 sec / 60 (DC 4.8V) 0.1 ± 0.01 sec / 60 (DC 6V)
- No-load current: 200 ± 20mA (DC 4.8V) 220 ± 20mA (DC 6V)
- Stopping torque: 1.3 ± 0.01kg · cm (DC 4.8V) 1.5 ± 0.1kg · cm (DC 6V)
- Stop current: 850mA (DC 4.8V) 1000mA (DC 6V)
- Standby current: 3 ± 1mA (DC 4.8V) 4 ± 1mA (DC 6V)

(3) Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

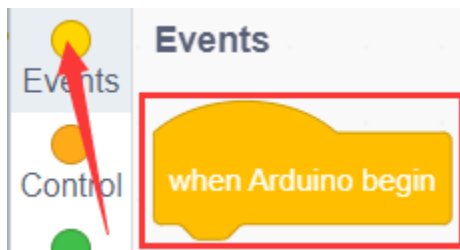
(4) Connection Diagram:

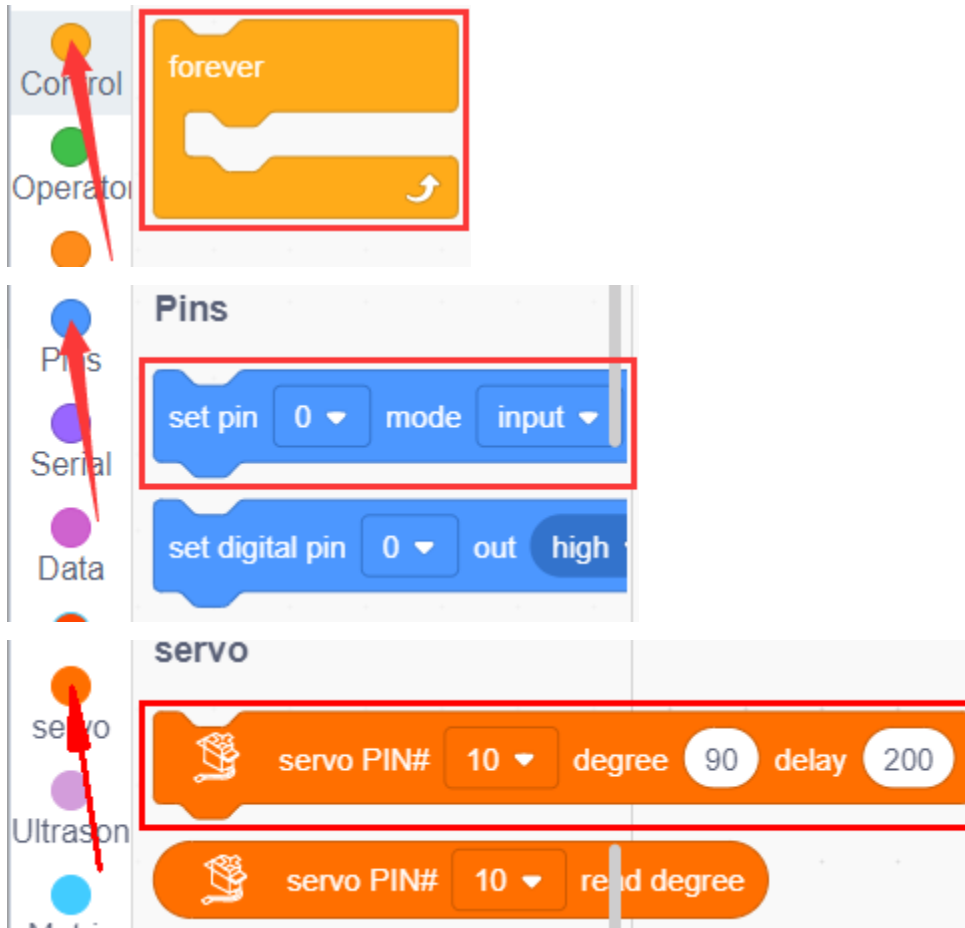


Note: The brown, red and orange wire of the servo are respectively attached to Gnd(G), 5v(V) and D10 of the shield. Remember to connect an external power because of the high current of the servo. If not, the development board will be burnt out.

(5) Test Code:

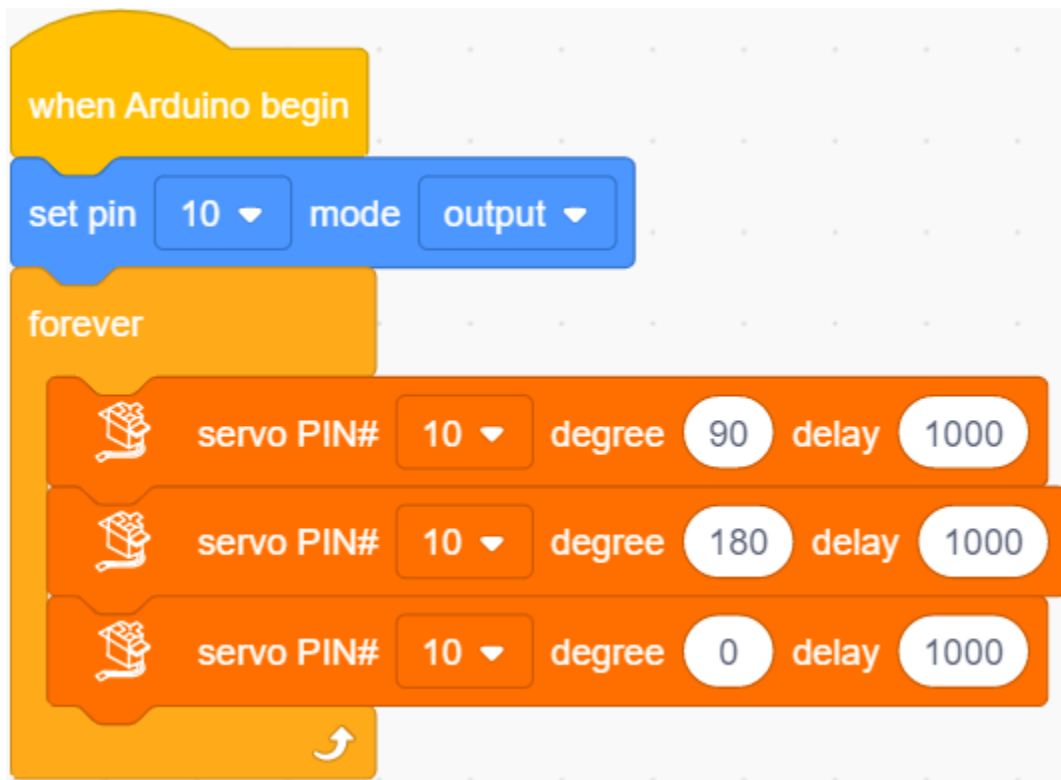
You can also drag blocks to edit your code, as shown below





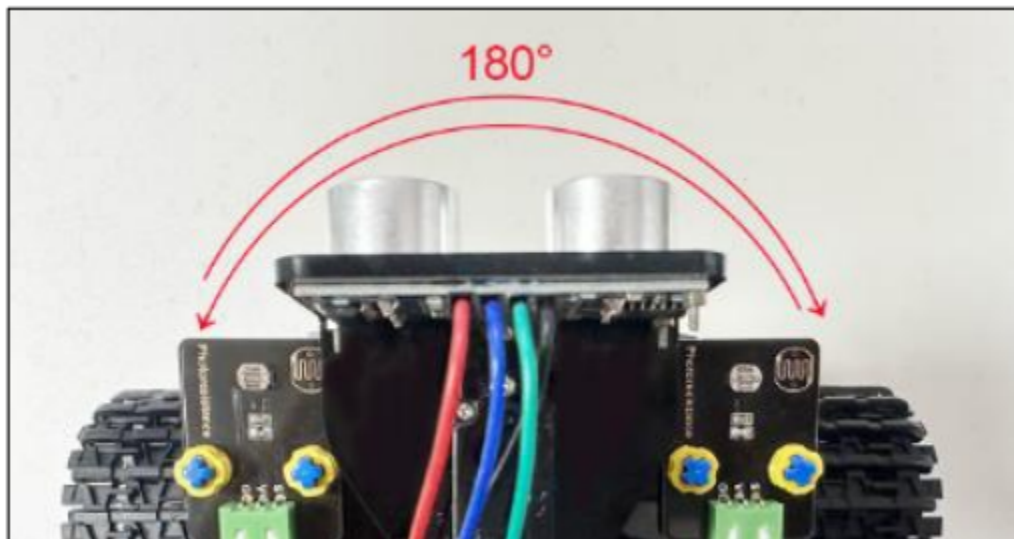
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



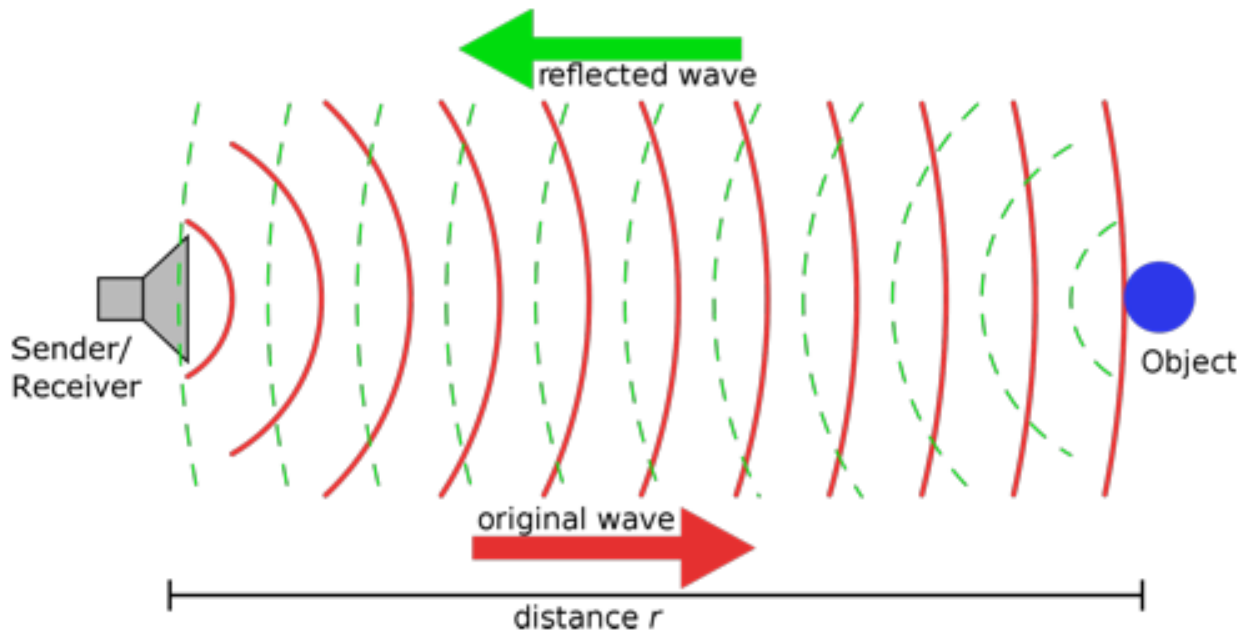
(6)Test Results:

Upload code, plug in power and servo moves in the range of 0° and 180°.



7.3.6 Project 6: Ultrasonic Sensor

(1)Description:



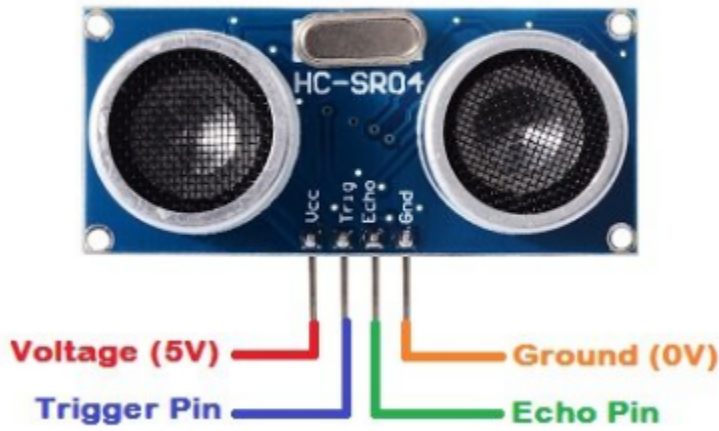
The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object like what bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.

The HC-SR04 or the ultrasonic sensor is being used in a wide range of electronics projects for creating obstacle detection and distance measuring application as well as various other applications. Here we have brought the simple method to measure the distance with Arduino and ultrasonic sensor and how to use ultrasonic sensor with Arduino.

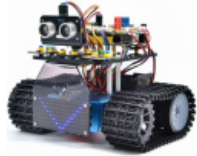


(2)Parameters:

- Power Supply :+5V DC
- Quiescent Current : <2mA
- Working Current: 15mA
- Effectual Angle: <15°
- Ranging Distance : 2cm – 400 cm
- Resolution : 0.3 cm
- Measuring Angle: 30 degree
- Trigger Input Pulse width: 10uS

Ultrasonic Sensor Pinout



(3) Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1
		

(4) The principle of ultrasonic sensor:

As the above picture shown, it is like two eyes. One is transmitting end, the other is receiving end.

The ultrasonic module will emit the ultrasonic waves after triggering a signal. When the ultrasonic waves encounter the object and are reflected back, the module outputs an echo signal, so it can determine the distance of the object from the time difference between the trigger signal and echo signal.

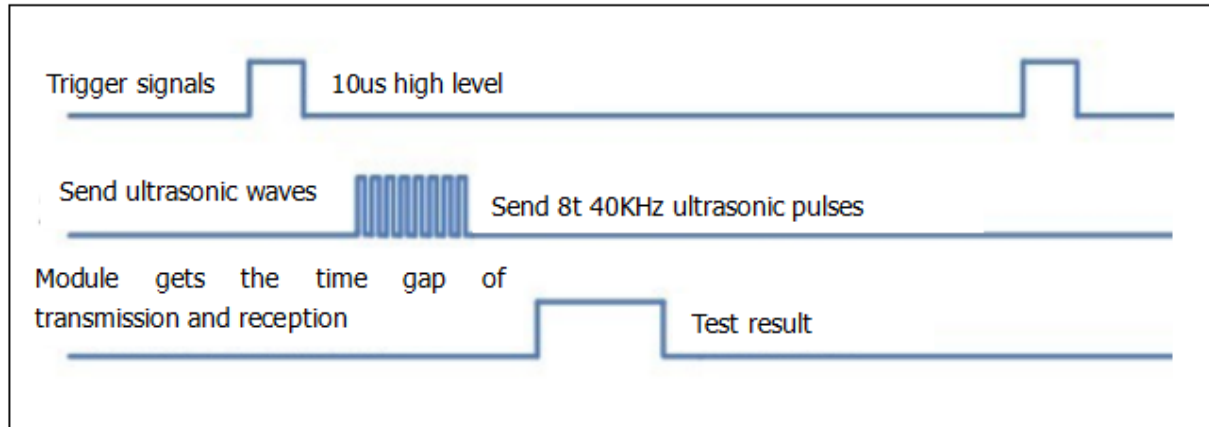
The t is the time that emitting signal meets obstacle and returns. And the propagation speed of sound in the air is about 343m/s, and $\text{distance} = \text{speed} * \text{time}$. However, the ultrasonic wave emits and comes back, which is 2 times of distance. Therefore, it needs to be divided by 2, the distance measured by **ultrasonic wave** = $(\text{speed} * \text{time})/2$

1. Use method and timing chart of ultrasonic module:

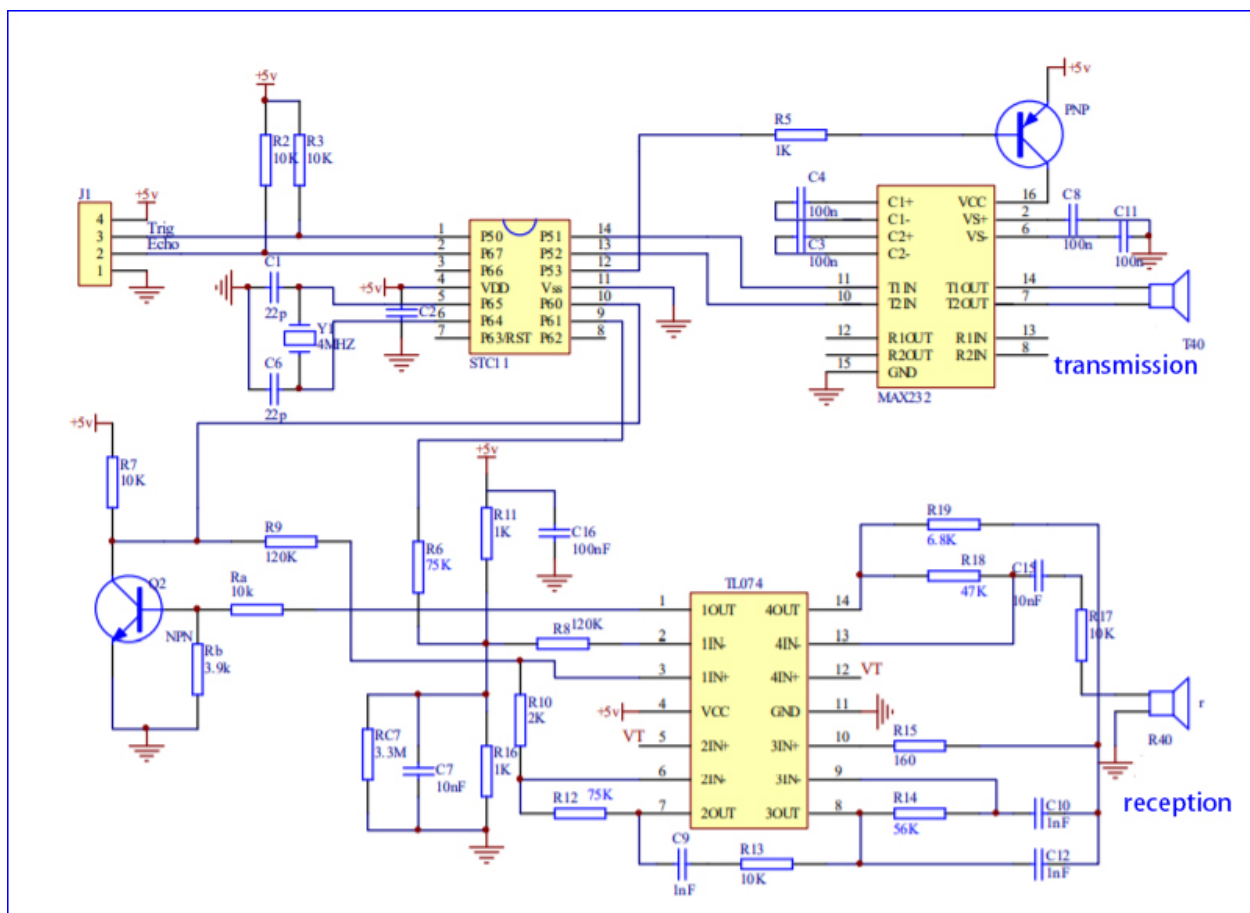
2. Setting the delay time of Trig pin of SR04 to 10s at least, which can trigger it to detect distance.

3. After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step will be completed automatically by the module.

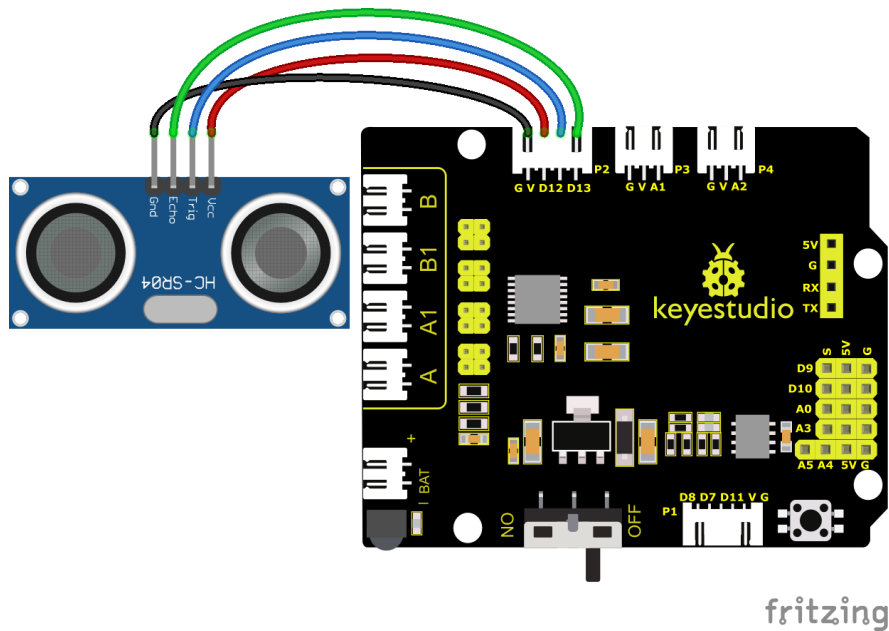
4. If the signal returns, the Echo pin will output a high level, and the duration of the high level is the time from the transmission of the ultrasonic wave to the return.



(5) Circuit diagram of ultrasonic sensor:



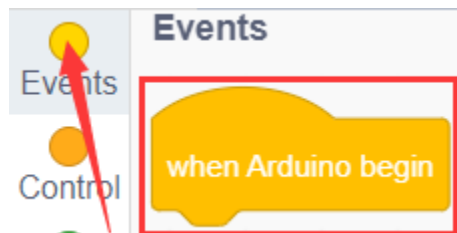
(6)Connection Diagram:

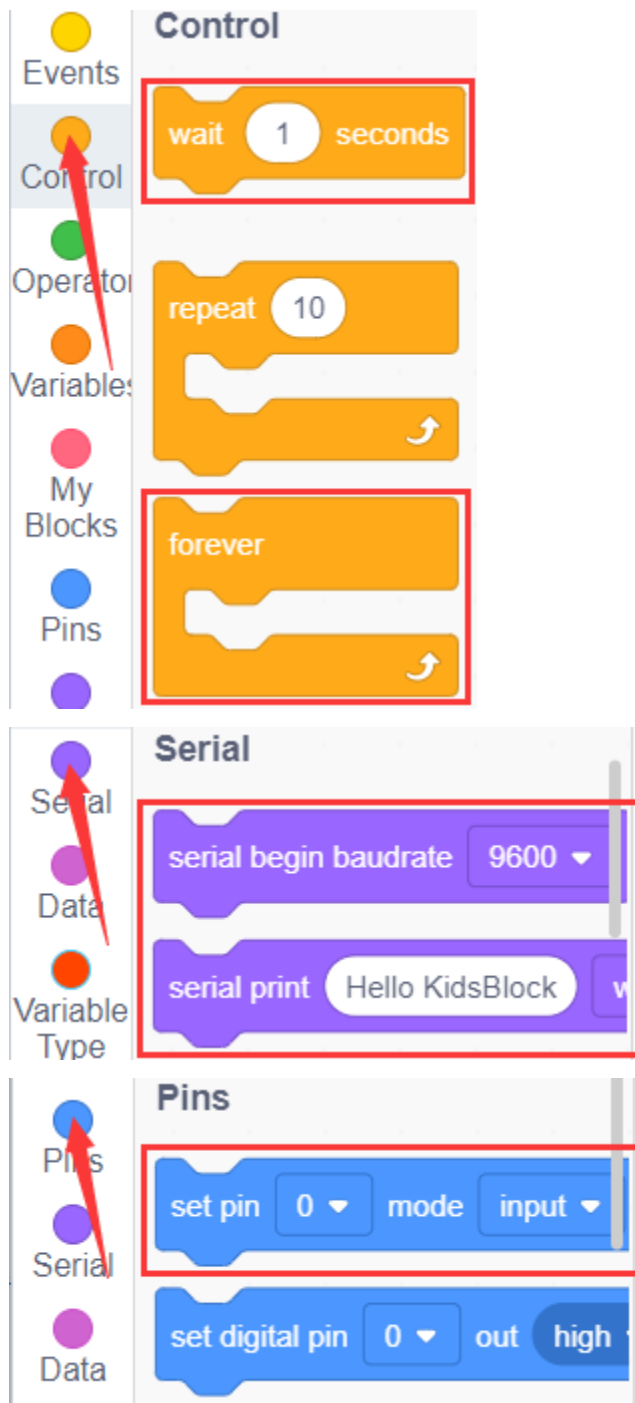


Wiring Note: The VCC pin of the ultrasonic sensor module is connected to the 5v(V) of the Keyestudio 8833 motor drive expansion board, the Trig pin is connected to digital D12, the Echo pin is connected to digital D13, and the Gnd pin is connected to Gnd(G);

(7)Test Code:

You can also drag blocks to edit your code, as shown below





The screenshot shows the Kidsblock IDE interface with three categories visible: Variable Type, Operators, and Ultrasonic. Red arrows point to the 'Variable Type' category, the 'Operators' category, and the 'Ultrasonic' category. Red boxes highlight specific blocks in each category.

Variable Type

- Declare Global variable Type int Name item Assigned to 0
- variable item
- Set item variable by 0

Operators

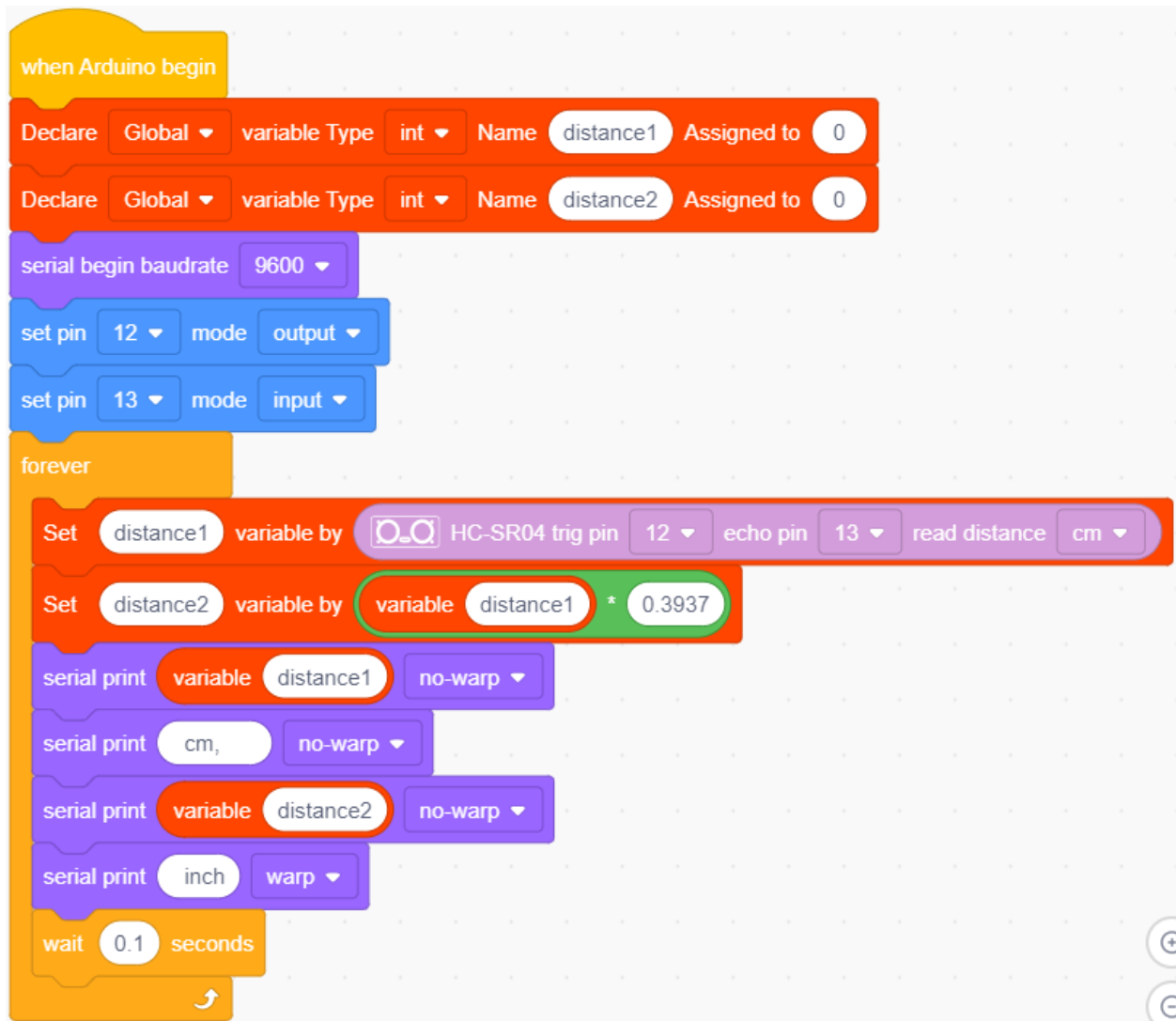
- 0 + 0
- 0 - 0
- 0 * 0
- 0 / 0

Ultrasonic

- HC-SR04 trig pin 12 echo pin 13 read distance cm

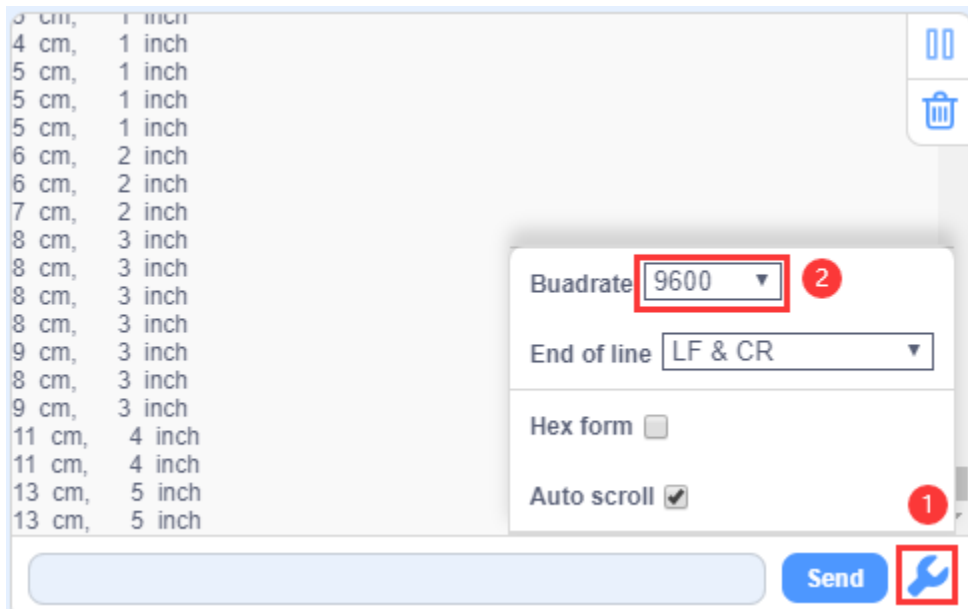
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)








(8)Test Results:

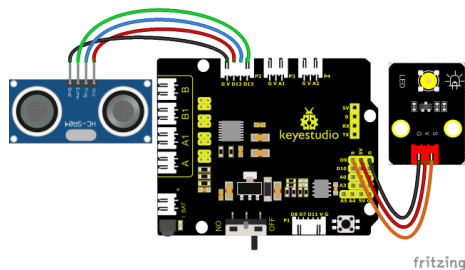
Upload test code on the development board, open serial monitor and set baud rate to 9600. The detected distance will be displayed, and the unit is cm and inch. Hinder the ultrasonic sensor by hand, the displayed distance value gets smaller.



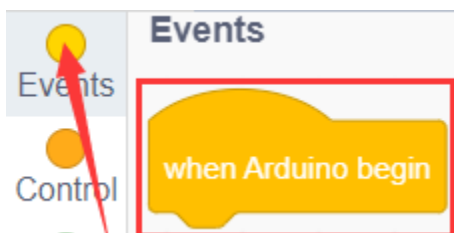
(9)Extension Practice:

Robot without BT Module*1	USB Cable*1	Yellow LED Module*1	3P-3P XH2.54 to 2.54 Dupont Wire*1	Computer*1
				

We have just measured the distance displayed by the ultrasonic. How about controlling the LED with the measured distance? Let's try it and connect an LED light module to the D9 pin.



You can also drag blocks to edit your code, as shown below



The image shows a block palette for a Mini Tank Robot project, organized into three main sections: Control, Serial, and Pins. Red arrows point to the 'Control' and 'Serial' categories in the left sidebar.

Control Section:

- forever** loop block.
- if-then-else** conditional block.

Serial Section:

- serial begin baudrate** block, set to 9600.
- serial print** block, with the text "Hello KidsBlock".

Pins Section:

- set pin** block, set to pin 0, mode input.
- set digital pin** block, set to pin 0, out high.
- set pwm pin** block, set to pin 3, out 255.

The screenshot displays the Kidsblock IDE interface. On the left, a vertical toolbar contains various block categories: Variable Type (red), TEXT (green), DC Motor (orange), Events (yellow), Control (orange), Operator (green), Variables (pink), My Blocks (blue), Pins (purple), Serial (purple), Data (purple), Ultrasonic (purple), and Matrix (blue). A red arrow points from the 'Variable Type' category to the 'Variable Type' block in the workspace. Another red arrow points from the 'Ultrasonic' category to the 'Ultrasonic' block in the workspace.

The workspace shows two main sections:

- Variable Type:** This section contains three blocks:
 - Declare:** A block with a dropdown menu set to 'Global', a dropdown menu set to 'int', a text field with 'item', and a dropdown menu set to '0'.
 - variable:** A block with a dropdown menu set to 'item'.
 - Set:** A block with a dropdown menu set to 'item', a text field with 'variable by', and a dropdown menu set to '0'.
- Operator:** This section contains several blocks:
 - 0 * 0:** A block with two input fields containing '0' and a multiplication operator.
 - 0 / 0:** A block with two input fields containing '0' and a division operator.
 - pick random:** A block with a dropdown menu set to '1' and a dropdown menu set to '10'.
 - > 50:** A block with an input field containing '50' and a greater-than operator.
 - < 50:** A block with an input field containing '50' and a less-than operator.
 - = 50:** A block with an input field containing '50' and an equals operator.
 - and:** A block with two input fields and an 'and' operator.
 - or:** A block with two input fields and an 'or' operator.
- Ultrasonic:** This section contains one block:
 - HC-SR04 trig pin:** A block with a dropdown menu set to '12', a dropdown menu set to '13', a text field with 'read distance', and a dropdown menu set to 'cm'.

Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the

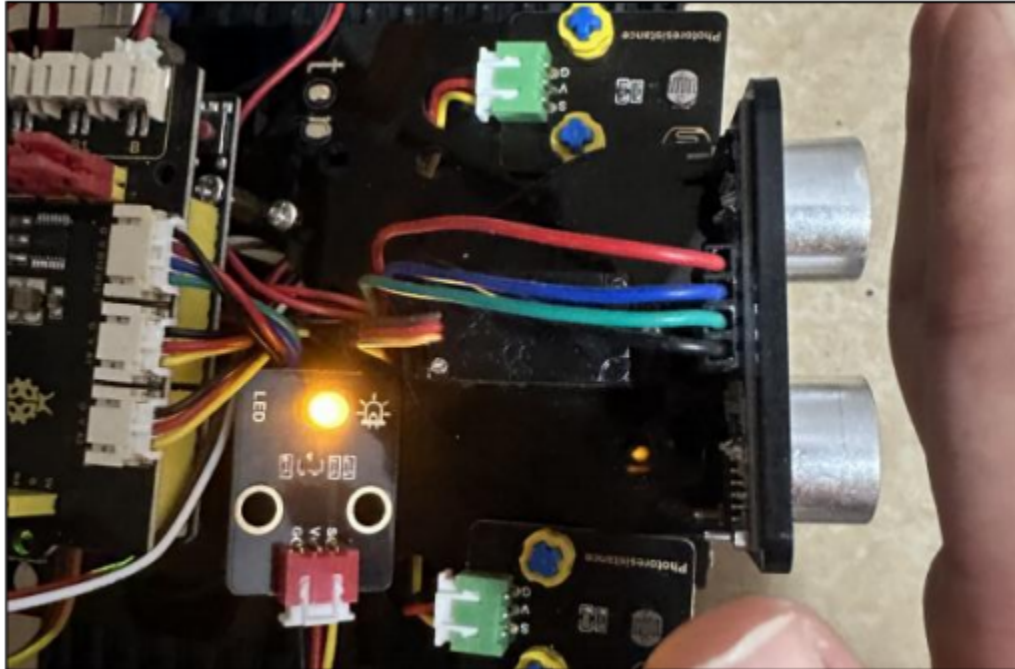
uploading of the code to fail.)

```

when Arduino begin
  Declare Global variable Type int Name distance1 Assigned to 0
  Declare Global variable Type int Name distance2 Assigned to 0
  serial begin baudrate 9600
  set pin 9 mode output
  set pin 12 mode output
  set pin 13 mode input

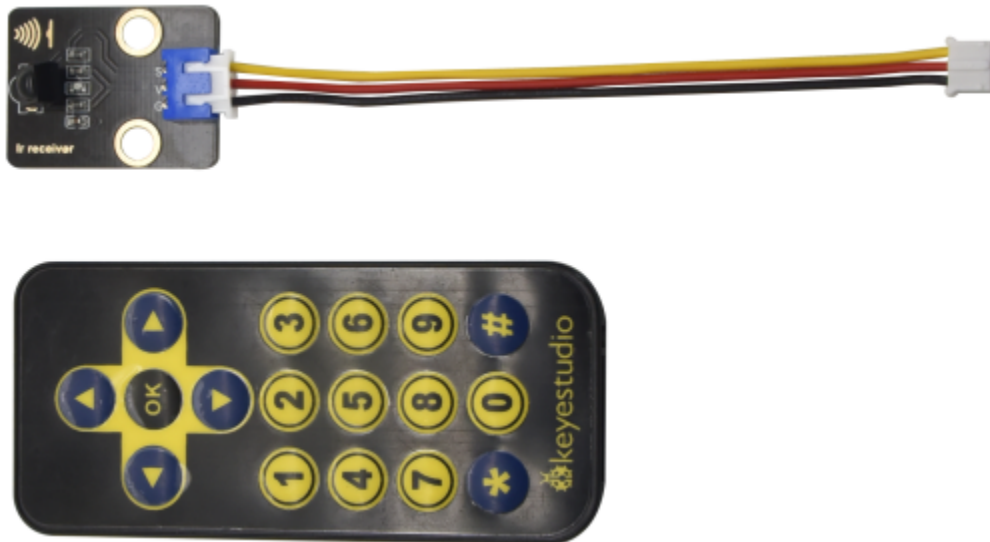
  forever
    Set distance1 variable by HC-SR04 trig pin 12 echo pin 13 read distance cm
    Set distance2 variable by variable distance1 * 0.3937
    serial print variable distance1 no-warp
    serial print cm, no-warp
    serial print variable distance2 no-warp
    serial print inch warp
    if (variable distance1 > 2 and variable distance1 < 10) then
      set digital pin 9 out high
    else
      set digital pin 9 out low
  
```

Upload test code to development board and move your hand and close to the ultrasonic sensor, then check if the LED is on.



7.3.7 Project 7: IR Reception

(1)Description:

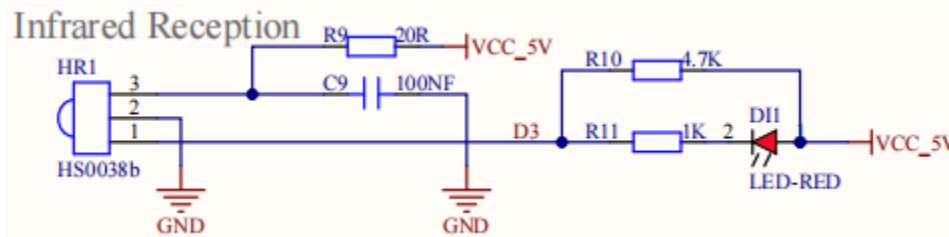


There is no doubt that infrared remote control is ubiquitous in daily life. It is used to control various household appliances, such as TVs, stereos, video recorders and satellite signal receivers. Infrared remote control is composed of infrared transmitting and infrared receiving systems, that is, an infrared remote control and infrared receiving module and a single-chip microcomputer capable of decoding.

The 38K infrared carrier signal emitted by remote controller is encoded by the encoding chip in the remote controller. It is composed of a section of pilot code, user code, user inverse code, data code, and data inverse code. The time interval of the pulse is used to distinguish whether it is a 0 or 1 signal and the encoding is made up of these 0, 1 signals.

The user code of the same remote control is unchanged while the data code can distinguish the key.

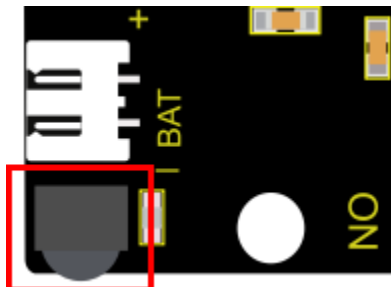
When the remote control button is pressed, the remote control sends out an infrared carrier signal. When the IR receiver receives the signal, the program will decode the carrier signal and determines which key is pressed. The MCU decodes the received 01 signal, thereby judging what key is pressed by the remote control.



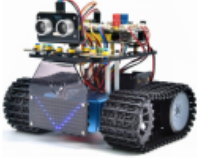



Infrared receiver we use is an infrared receiver module. Mainly composed of an infrared receiver head, which is a device that integrates reception, amplification, and demodulation. Its internal IC has completed demodulation, and can achieve from infrared reception to output and be compatible with TTL signals. Additionally, it is suitable for infrared remote control and infrared data transmission. The infrared receiving module made by the receiver has only three pins, signal line, VCC and GND. It is very convenient to communicate with Arduino and other microcontrollers.

(2)Parameters:

- Operating Voltage: 3.3-5VDC
- Interface: 3PIN
- Output Signal: Digital signal
- Receiving Angle: 90 degrees
- Frequency: 38khz
- Receiving Distance: 10m



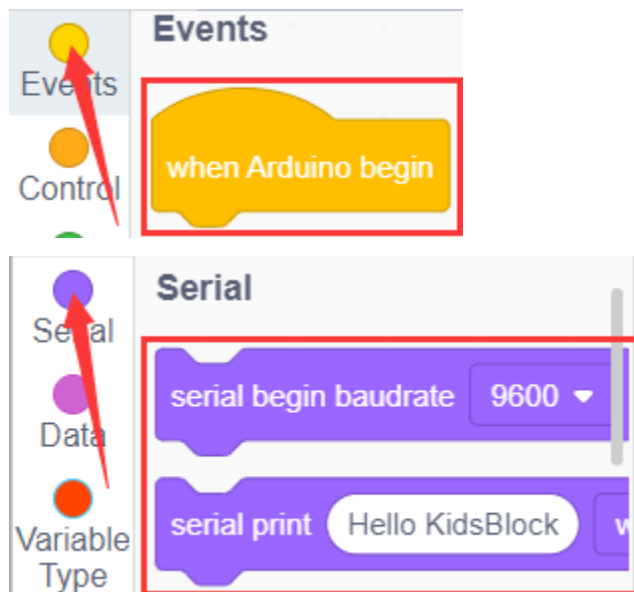
(3) Components Required:

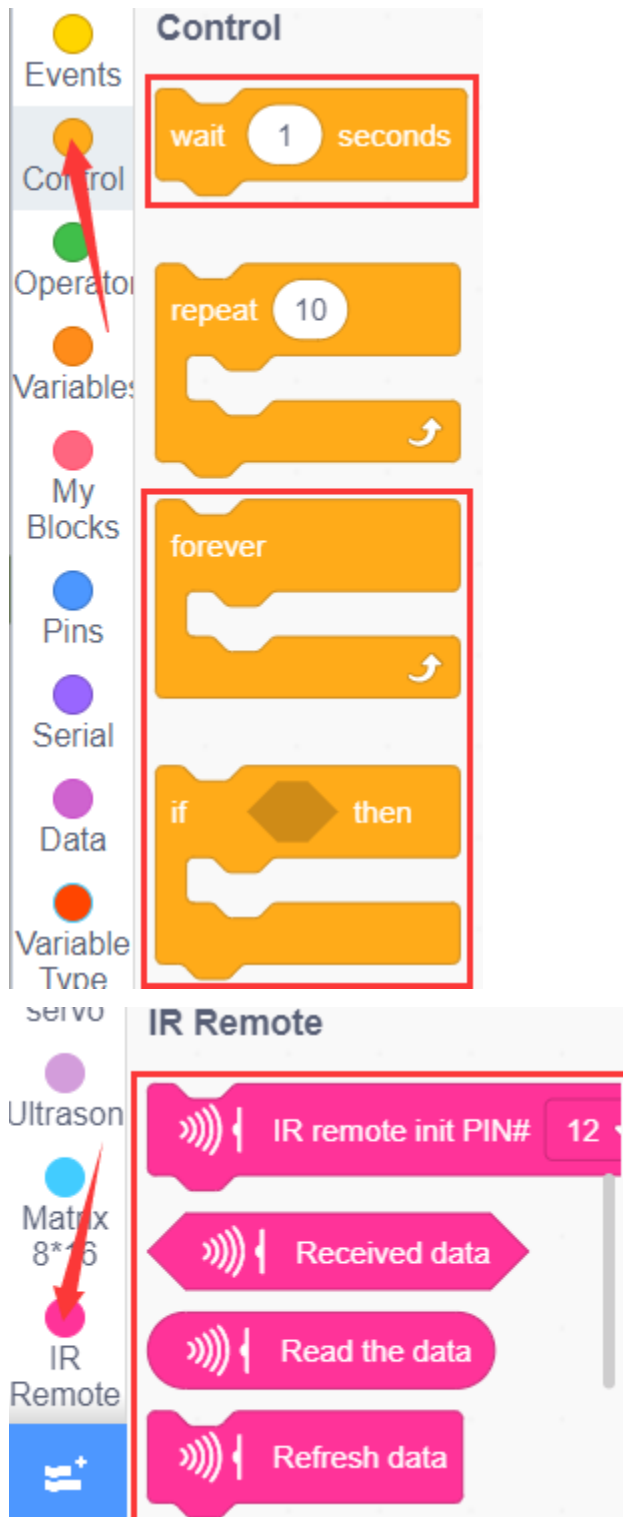
Robot without BT module*1	USB Cable*1	Computer*1	Remote Control*1
			

Note: Since the IR receiver is integrated in the Keystudio 8833 motor drive expansion board, no additional wiring is required. The pins of the IR receiver on the Keystudio 8833 motor drive expansion board are G (GND), V (VCC) and D3.

(4) Test Code:

You can also drag blocks to edit your code, as shown below





Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

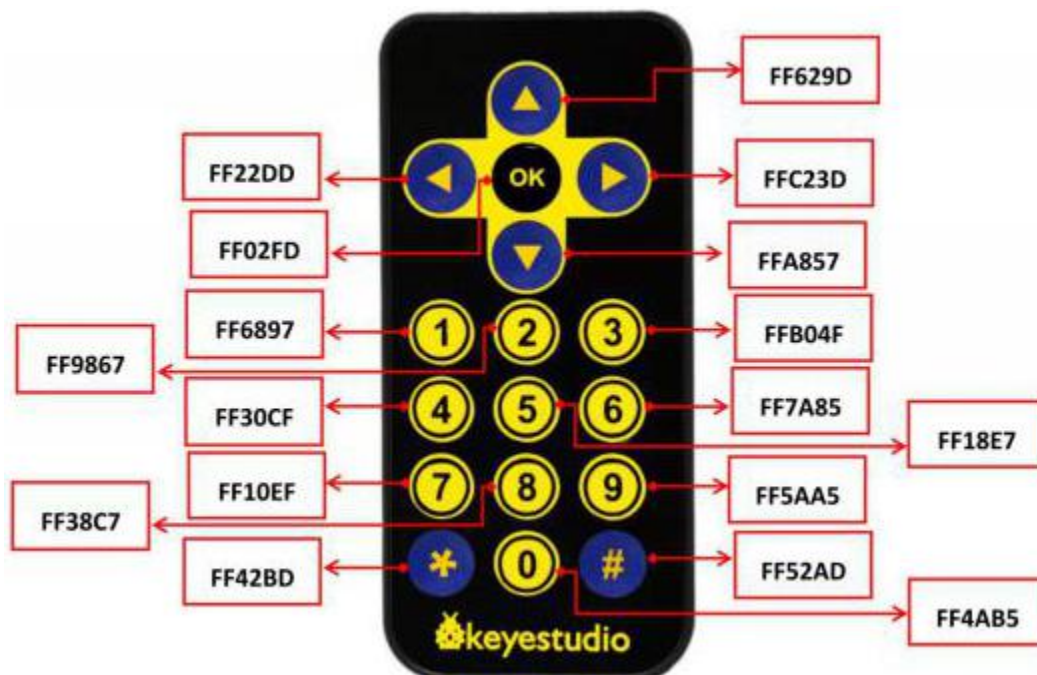


(5)Test Results:

Upload the code to the development board, click to set the baud rate to 9600, take out the remote control, aim at the infrared receiving sensor to send the signal, you can see the key value of the corresponding key, if the key time is too long, garbled FFFFFFFF will easily appear.

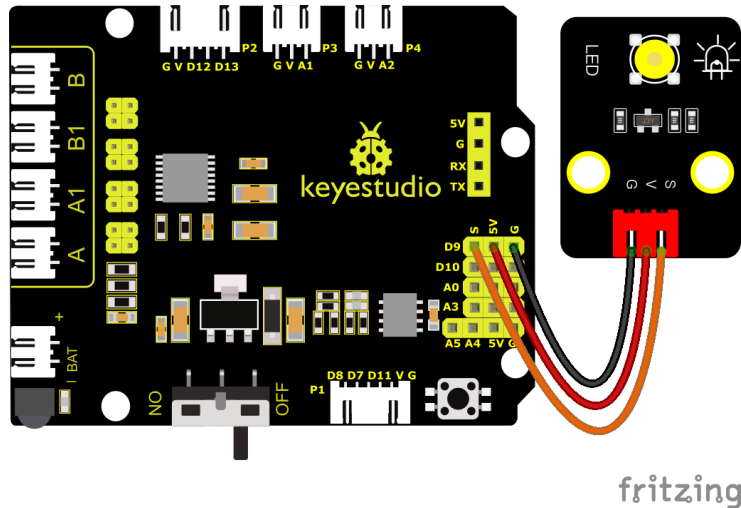


Below we have listed out each key value of keystudio remote control. So you can keep it for reference.

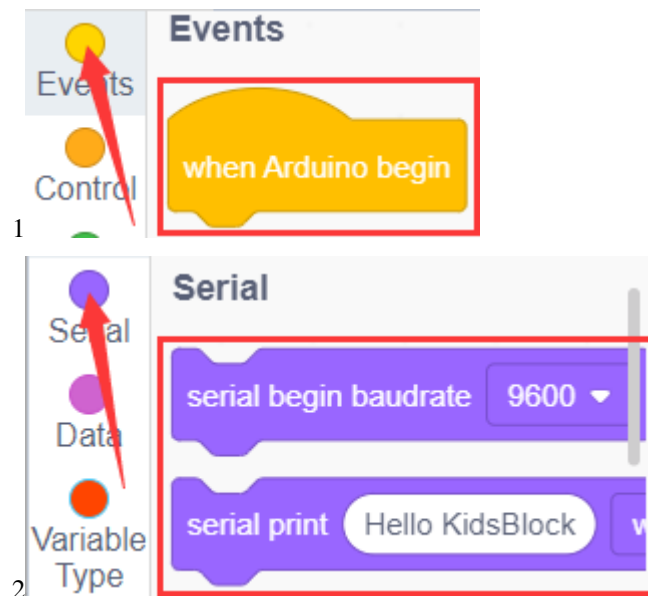


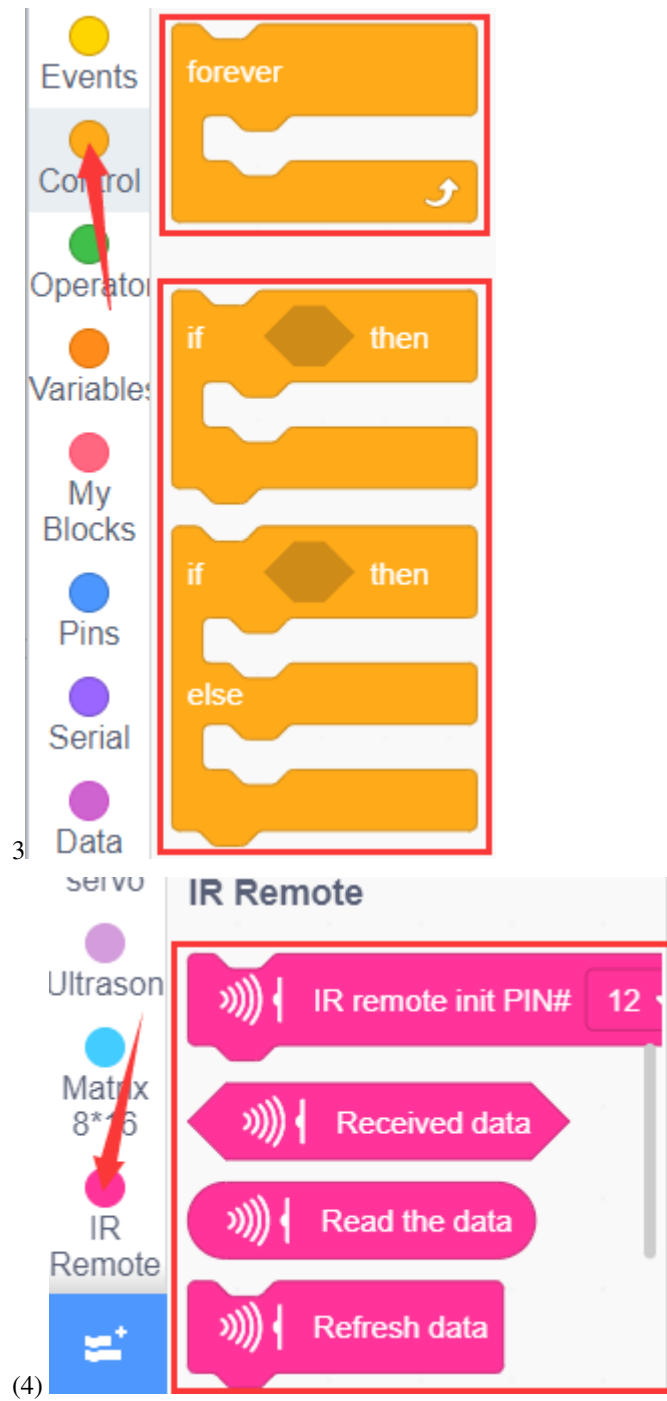
(6)Extension Practice:

We just decoded the key values of the IR remote. If you control an LED light on and off. We need to connect an LED light module to the D9 pin, and the pin position of the infrared receiver remains unchanged. When the OK button on the remote control is pressed, the LED connected to D9 will light up, and when the OK button is pressed again, the LED will be off.



You can also drag blocks to edit your code, as shown below





Blocks Pins

Serial

5 Data

Variable Type

6 DC Motor

Control

Operator

7 Variables

Pins

set pin 0 mode input

set digital pin 0 out high

Variable Type

Declare Global variable Type int Name item Assigned to 0

variable item

Set item variable by 0

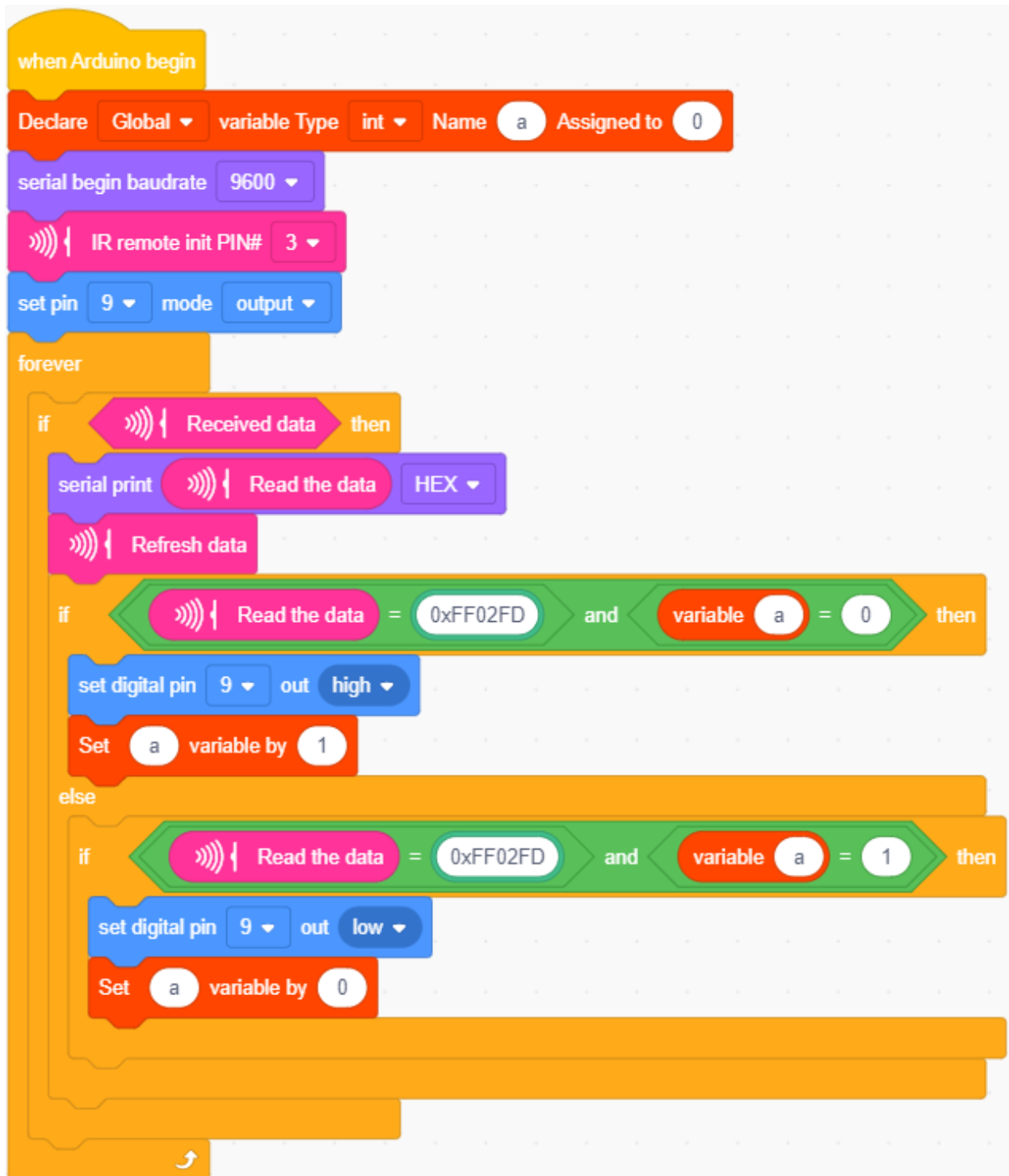
Control

= 50

and

Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



Upload code to development board, press the“OK”key on remote control to make LED on and off.



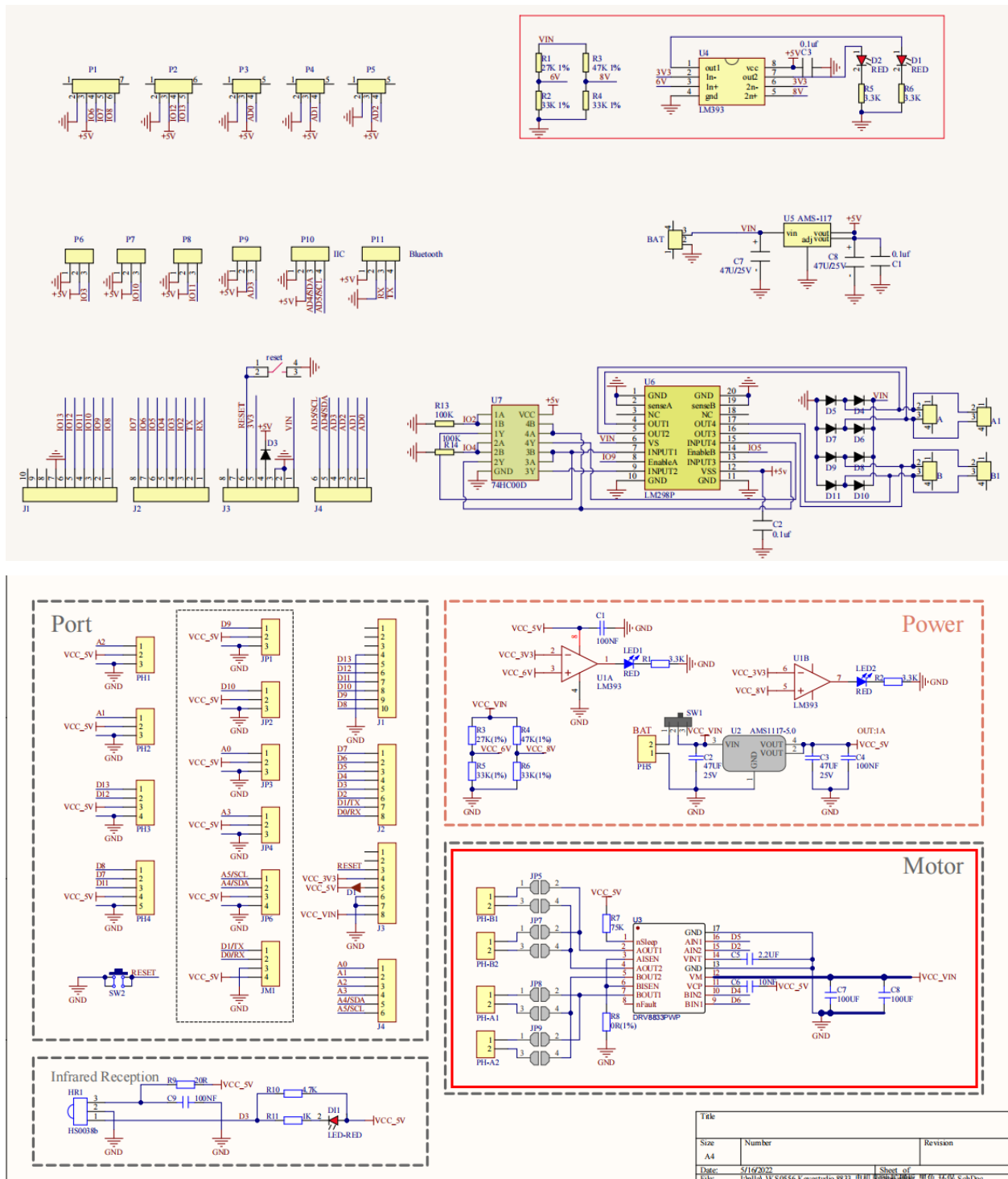
7.3.8 Project 8: Motor Driving and Speed Control

(1)Description:

There are many ways to drive motors. Our smart car uses the most common solution called L298P. L298P, produced by STMicroelectronics, is an excellent driving chip specially designed for driving high-power motors . It can directly drive DC motors, two-phase and four-phase motors with the driving current reaching 2A. And the motor's output terminal adopts 8 high-speed Schottky diodes as protection. We have designed an expansion board based on the L298P circuit of which the laminated design can be directly plugged into the UNO R3 board for use reducing the technical difficulties for users in using and driving the motor.

Stack the expansion board on the board, power the BAT , turn the DIP switch to the ON end, and power the expansion board and the UNO R3 board at the same time via external power supply. In order to facilitate wiring, the expansion board is equipped with anti-reverse interface (PH2.0 -2P -3P -4P -5P) and thus it can be directly plug with motors, power supply, and sensors /modules. The Bluetooth interface of the drive expansion board is fully compatible with the Keyestudio HM-10 Bluetooth module. Therefore, we only need to insert the HM-10 Bluetooth module into the corresponding interface when connecting. At the same time, the drive extension board also uses 2.54 pin headers to extend out some available digital ports and analog ports, so that you can continue to add other sensors and carry out expansion experiments.

The expansion board can be connected to 4 DC motors. In the default jumper cap connection mode, the A and A1, B and B1 interface motors are connected in parallel, and their motion pattern is the same. 8 jumper caps can be used to control the rotation direction of the 4 motor interfaces. For example, when the two jumper caps in front of the motor A interface are changed from a horizontal connection to a vertical connection, the rotation direction of the motor A now is opposite to the original rotation direction.



(2)Parameters:

- Logic part input voltage: DC 5V
- Driving part input voltage: DC 7-12V
- Logic part working current: 36mA
- Driving part working current: 2A
- Maximum dissipation power: 25W (T=75°C)
- Control signal input level:
High level: 2.3V $V_{in} \leq 5V$
Low level: 0V $V_{in} \leq 1.5V$
- Working temperature: -25°C~130°C

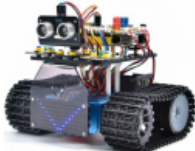



(3)Drive the robot to move:

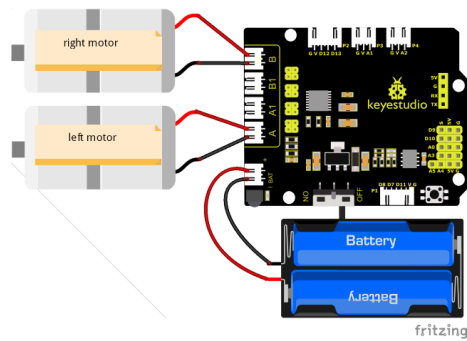
The direction pin of A motor is D2, the speed control pin is D5; the direction pin of B motor is in D4 and the speed control pin is D6,

According to the table below, we can know how to control the movement of the robot by controlling the rotation of two motors through the digital ports and PWM ports . Among them, the range of PWM value is 0-255. The larger the value is, the faster the motor rotates.

Function	D4	D6PWM	Motor leftB	D2	D5PWM	MotorRightA
Move Forward	HIGH	0	Rotate Left	HIGH	0	Rotate Left
Go Back	LOW	255	Rotate Right	LOW	255	Rotate Right
Rotate Left	LOW	255	Rotate Right	HIGH	100	Rotate Left
Rotate Right	HIGH	100	Rotate Left	LOW	255	Rotate Right
Stop	LOW	0	Stop	LOW	0	Stop

(5)Components Required:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

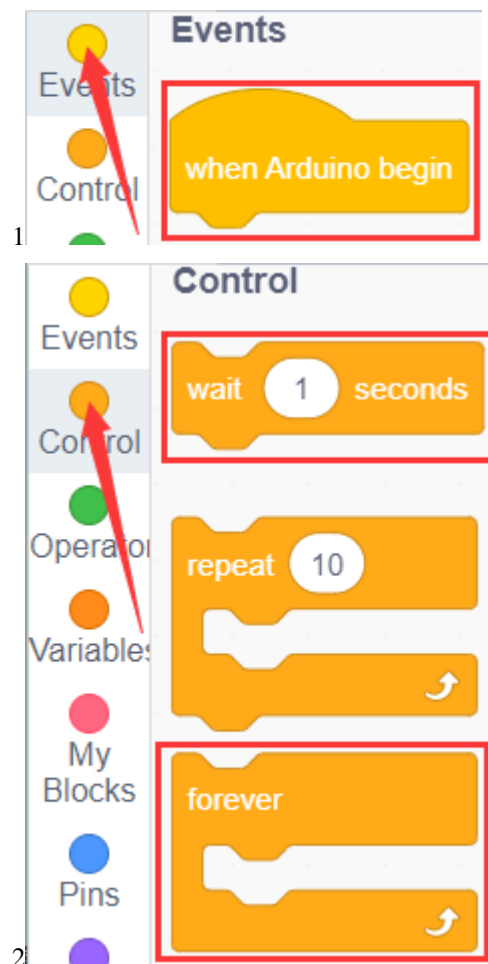
(6) Connection Diagram:

Note:

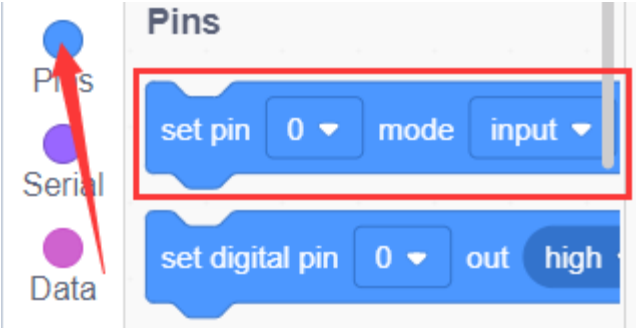
The 4-pin connector is marked with A, A1, B1 and B. The right rear motor is connected to B of the 8833 board and left front one is connected to A port.

(7) Test Code:

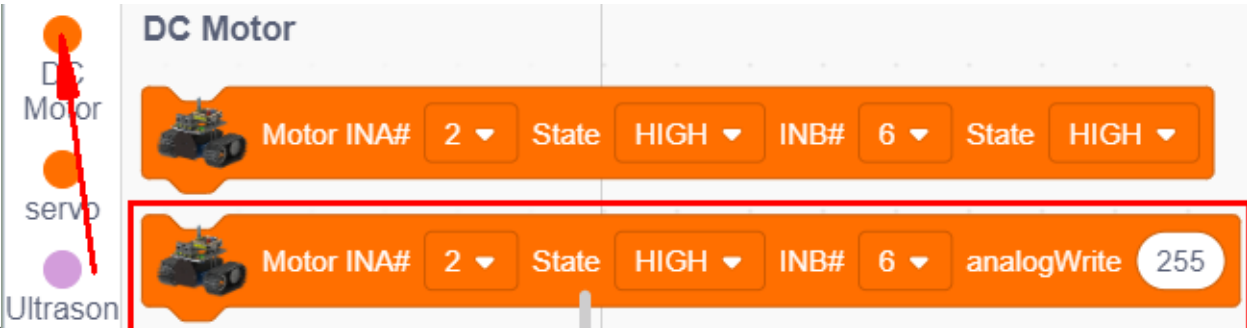
You can also drag blocks to edit your code, as shown below



3

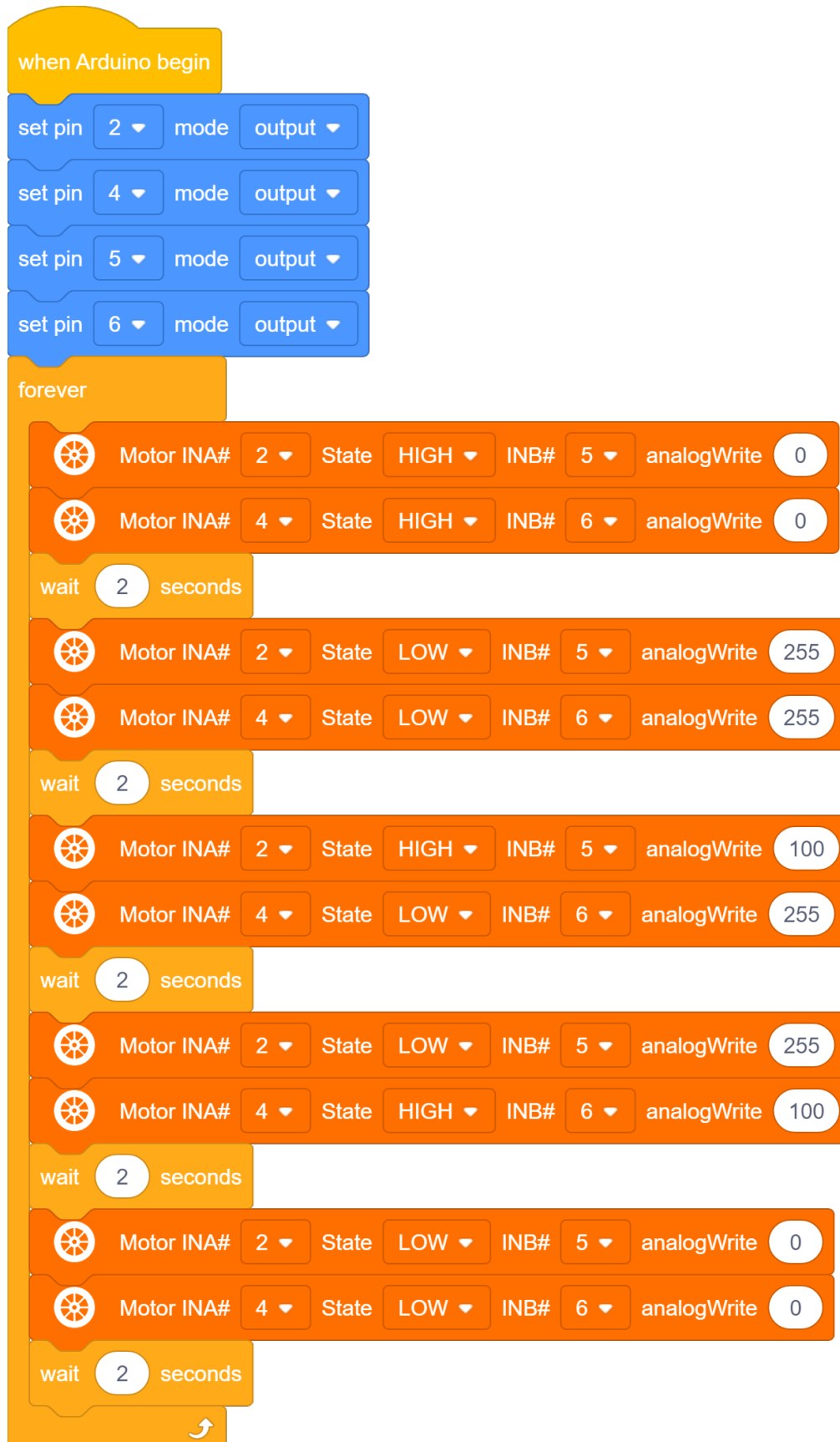


4



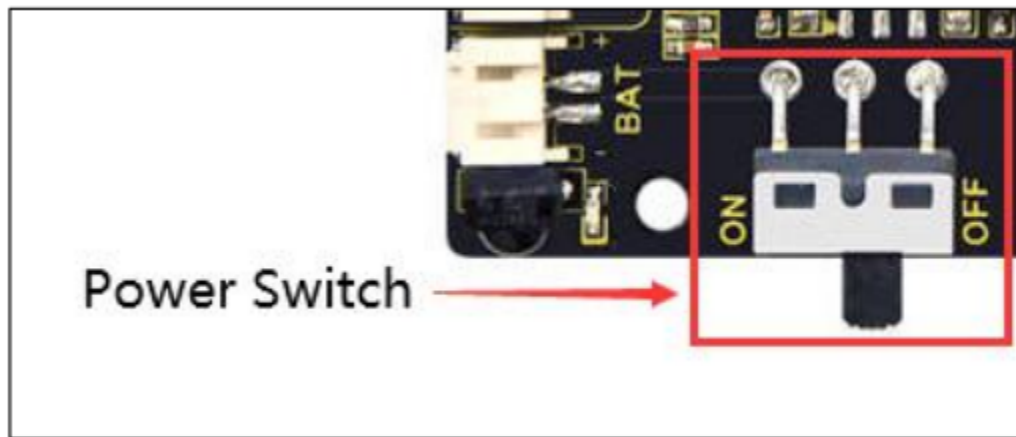
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(8)Test Results:

After wiring according to the diagram, uploading the test code and powering it up.



the smart car moves forward for 2s, goes back for 2s, turns left for 2s, turns right for 2s and stops for 2s

7.3.9 Project 9: 8*16 Facial Expression LED Dot Matrix

(1)Description:

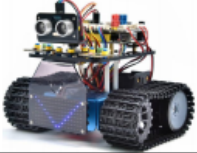

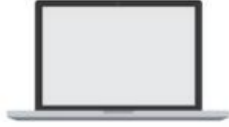

Won't it be fun if a expression board is added to the robot? And the Keystudio 8*16 LED dot matrix can do the trick. With the help of it, you could design facial expressions, images, patterns and other displays by yourselves.

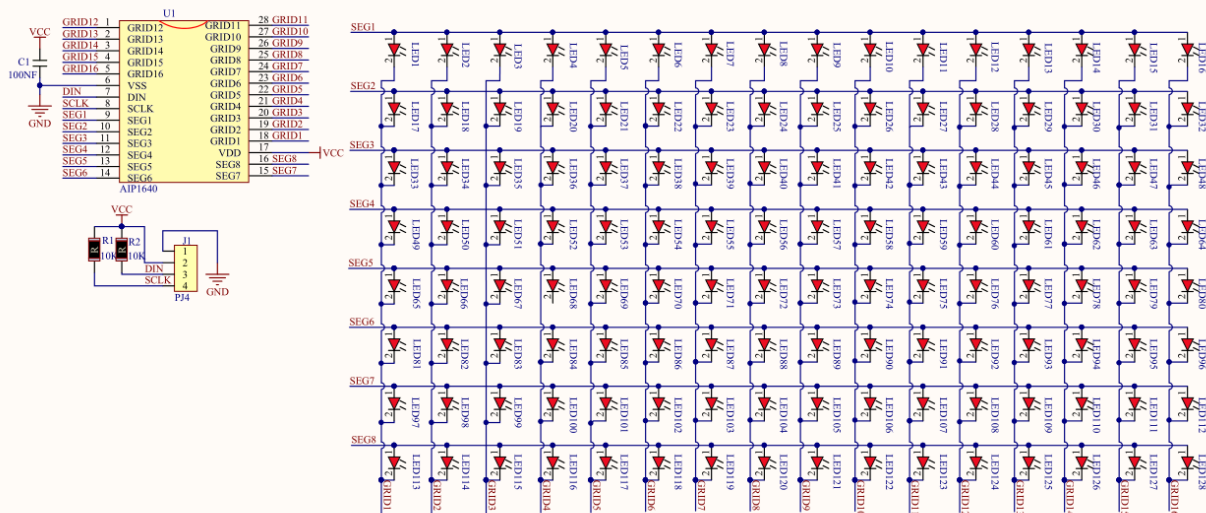
The 8*16 LED board comes with 128 LEDs. The data of the microprocessor (Arduino) communicates with the AiP1640 through a two-wire bus interface. Therefore, it can control the on and off of 128 LEDs on the module, so as to make the dot matrix on the module to display the pattern you need. A HX-2.54 4Pin cable is provided for your convenience of wiring.

(2)Parameters:

- Working voltage: DC 3.3-5V
- Power loss: 400mW
- Oscillation frequency: 450KHz
- Drive current: 200mA
- Working temperature: -40~80°C
- Communication mode: two-wire bus

(3)Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

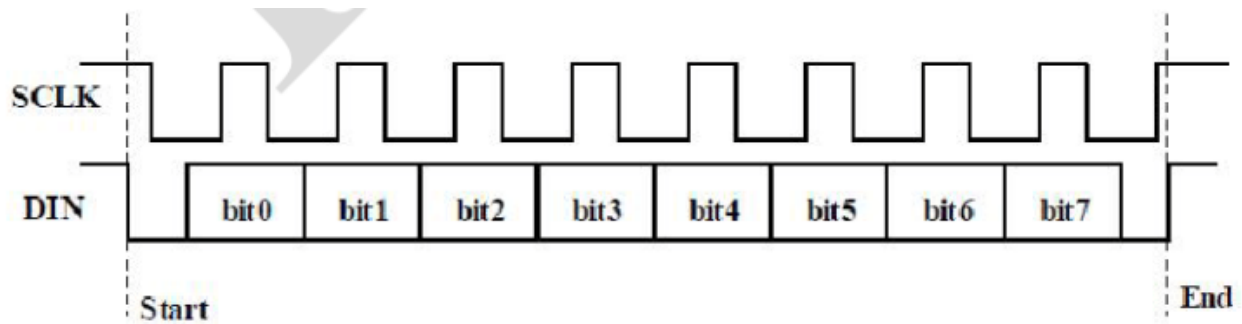
(4)About the 8*16 Dot Matrix:**Circuit of the 8*16 LED dot matrix****Principle of the 8*16 LED dot matrix**

How to control each LED of the 8*16 dot matrix? It is known that each byte has 8 bits and each bit is 0 or 1. when it is 0, LED is off while when it is 1 LED is on. One byte can control one column of the LED, and naturally 16 bytes can control 16 columns of LEDs, that's the 8*16 dot matrix.

Pins description and communication protocol

The data of the microprocessor (Arduino) communicates with the AiP1640 through a two-wire bus cable.

The communication protocol diagram is as follows (SCLK) is SCL, (DIN) is SDA



The starting condition for data input: SCL is high level and SDA changes from high to low.

For data command setting, there are methods as shown in the figure below

In our sample program, select the way to **add 1 to the address automatically**, the binary value is 0100 0000 and the corresponding hexadecimal value is 0x40

B7	B6	B5	B4	B3	B2	B1	B0	Description
0	1	Irrelevant choice, fill in 0			0	Irrelevant choice, fill in 0		add 1 to the address
0	1				1			automatically
0	1		0					Fixed address
0	1		1					Universal mode
								Test mode

For address command setting, the address can be selected as shown below.

The first 00H is selected in our sample program, and the binary number 1100 0000 corresponds to the hexadecimal 0xc0

B7	B6	B5	B4	B3	B2	B1	B0	Display address
1	1	Irrelevant choice, fill in 0		0	0	0	0	00H
1	1			0	0	0	1	01H
1	1			0	0	1	0	02H
1	1			0	0	1	1	03H
1	1			0	1	0	0	04H
1	1			0	1	0	1	05H
1	1			0	1	1	0	06H
1	1			0	1	1	1	07H
1	1			1	0	0	0	08H
1	1			1	0	0	1	09H
1	1			1	0	1	0	0AH
1	1			1	0	1	1	0BH
1	1			1	1	0	0	0CH
1	1			1	1	0	1	0DH
1	1			1	1	1	0	0EH
1	1			1	1	1	1	0FH

The requirement for data input is that when SCL is at high level when inputting data, the signal on SDA must remain unchanged. Only when the clock signal on SCL is at low level, can the signal on SDA be changed. The input of data is the low bit first, and the high bit later.

The condition for the end of data transmission is that when SCL is at low level, SDA at low level and SCL at high level, the level of SDA becomes high.

Display control, set different pulse width, pulse width can be selected as shown in the figure below.

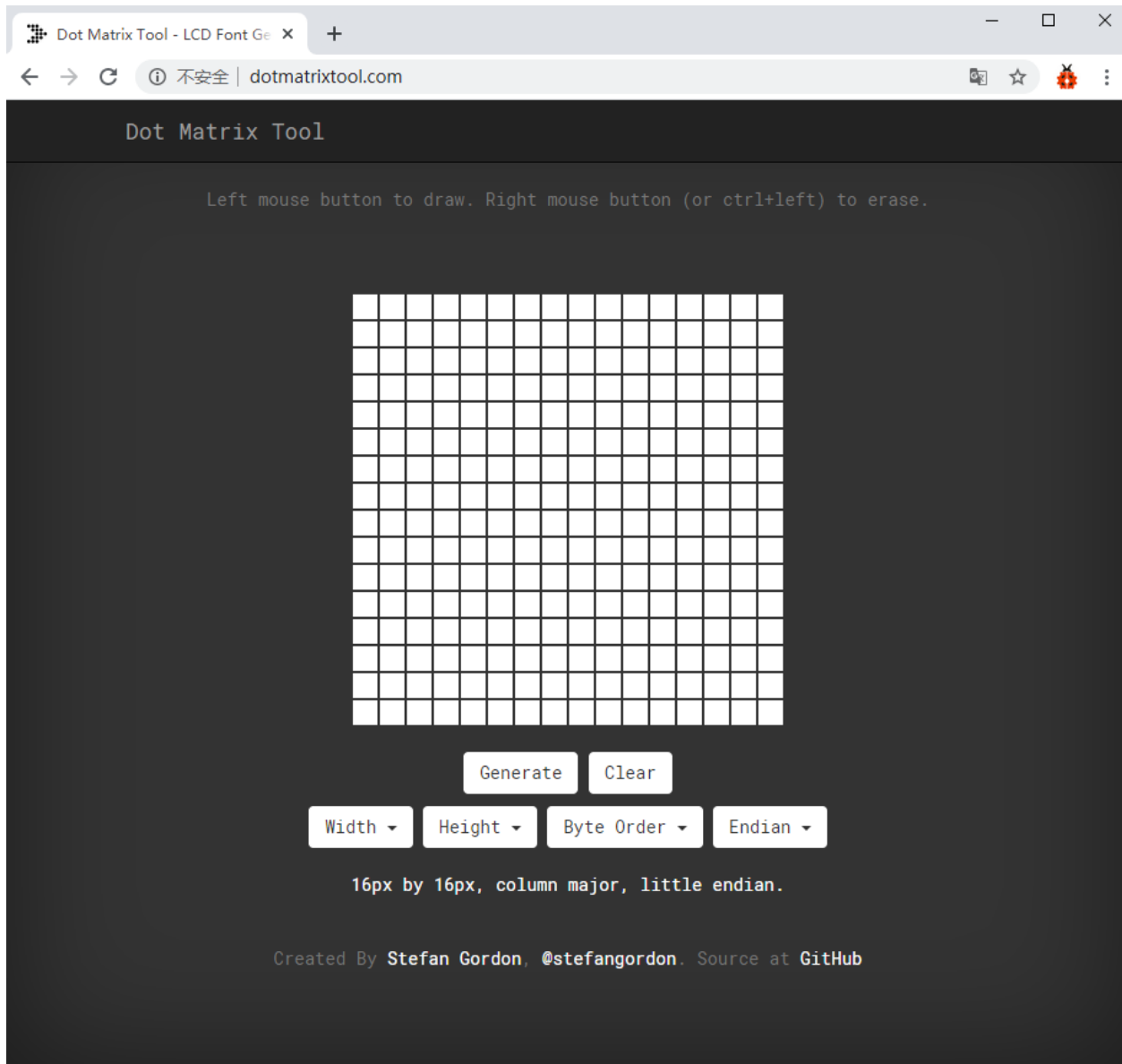
In the example, the pulse width is 4/16, and the hexadecimal corresponding to 1000 1010 is 0x8A

B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
1	0	Irrelevant choice, fill in 0		1	0	0	0	Clear quantity setting (Brightness setting)	Set pulse width to 1/16
1	0			1	0	0	1		Set pulse width to 2/16
1	0			1	0	1	0		Set pulse width to 4/16
1	0			1	0	1	1		Set pulse width to 10/16
1	0			1	1	0	0		Set pulse width to 11/16
1	0			1	1	0	1		Set pulse width to 12/16
1	0			1	1	1	0		Set pulse width to 13/16
1	0			1	1	1	1		Set pulse width to 14/16
1	0			0	X	X	X	Display switch setting	On off
1	0			1	X	X	X		

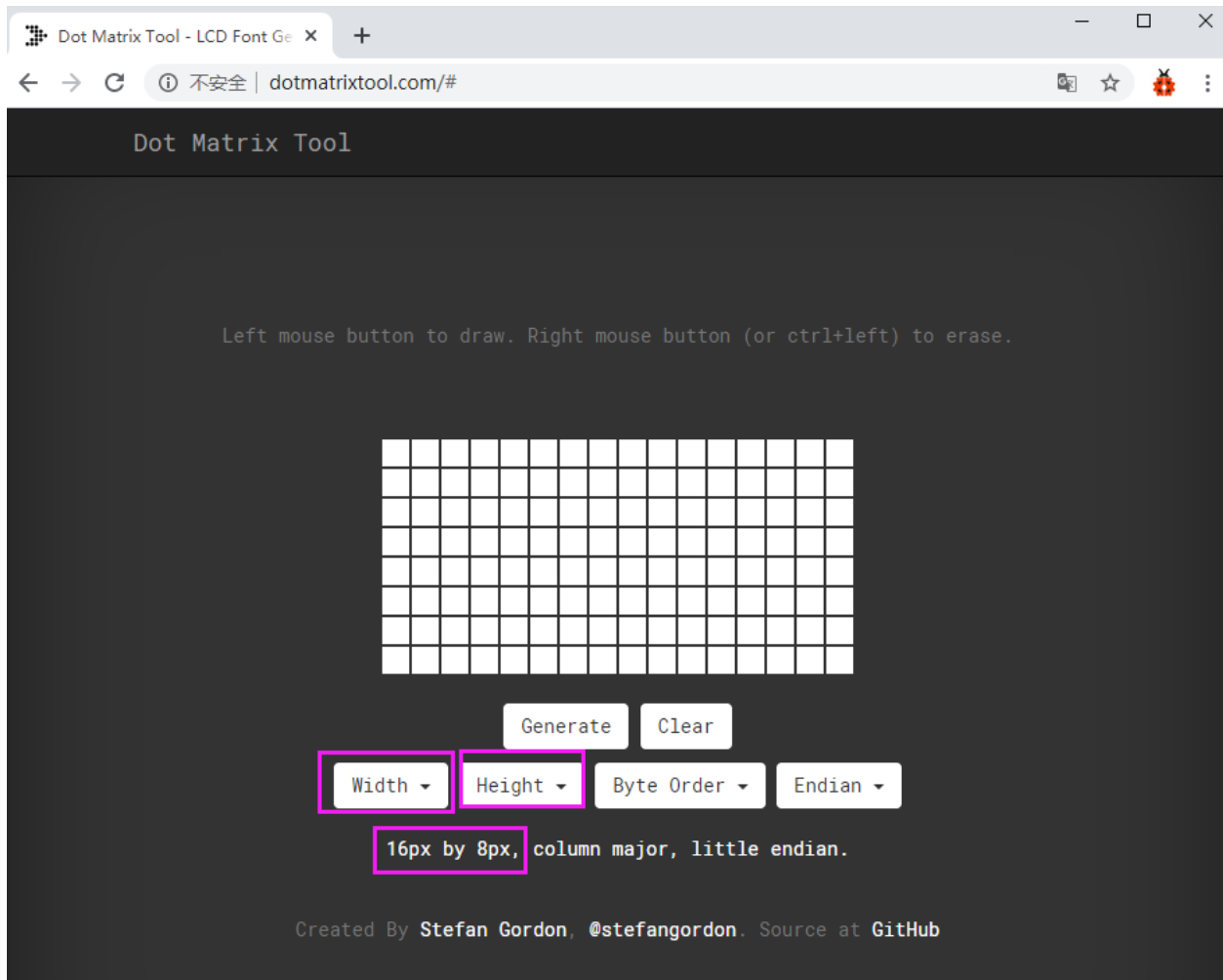
4. Instructions for the use of modulus tool

The dot matrix tool uses the online version, and the link is: <http://dotmatrixtool.com/#>

Enter the link and the page appears as shown below

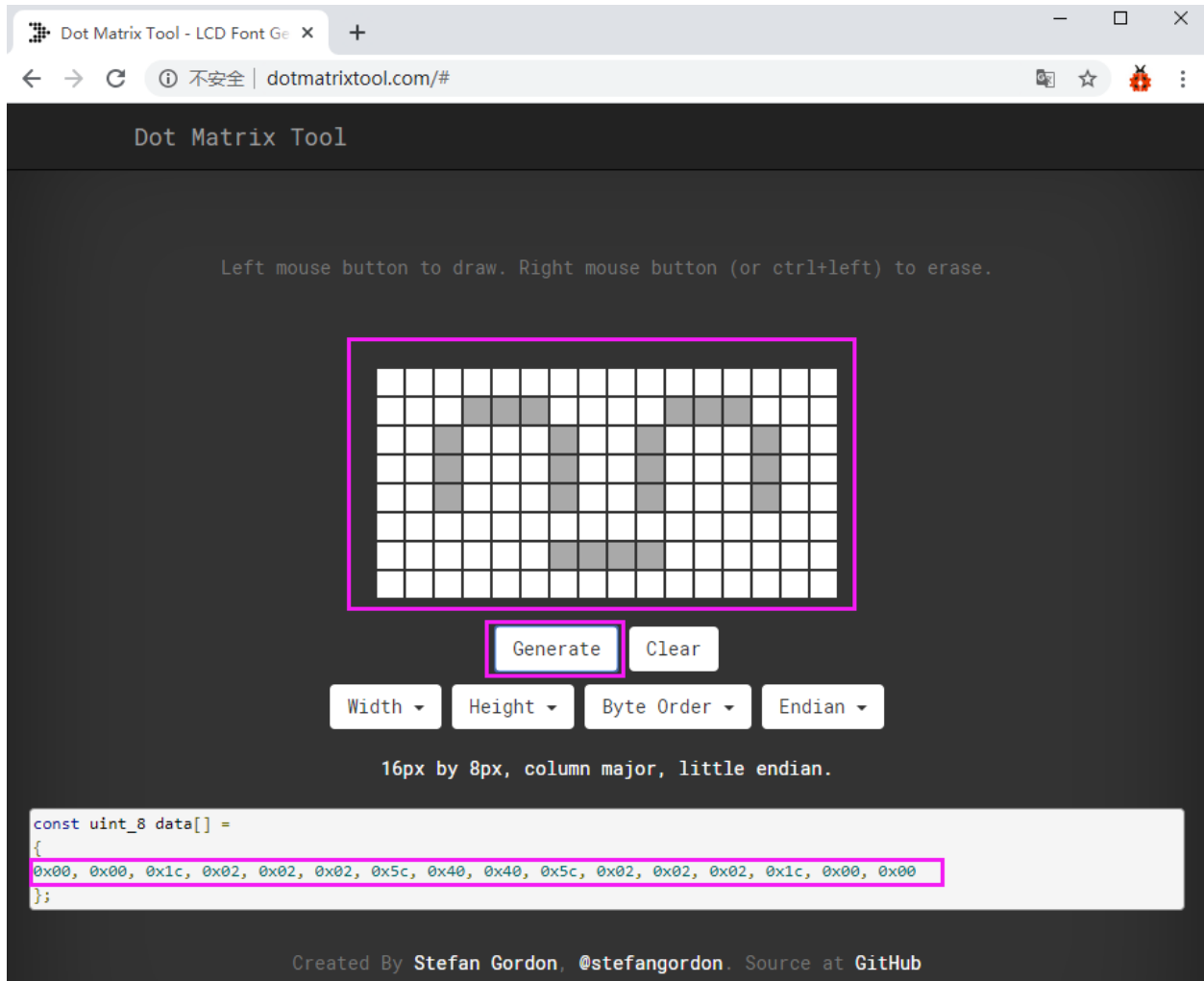


The dot matrix is 8*16, so adjust the height to 8 and width to 16, as shown in the figure below

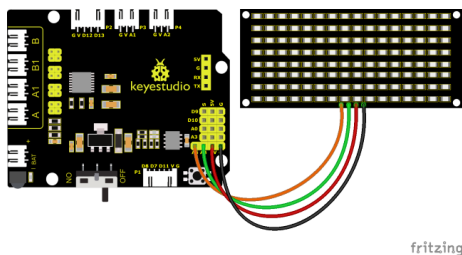


Generate hexadecimal data from the pattern

As shown in the figure below, press the left mouse button to select, right click to cancel; draw the pattern you want, click Generate, and the hexadecimal data we need will be generated.



(5)Connection Diagram:



The GND, VCC, SDA, and SCL of the 8x16 LED light board are respectively connected to the G(GND), V (VCC), A4 and A5 of the expansion board for two-wire serial communication.

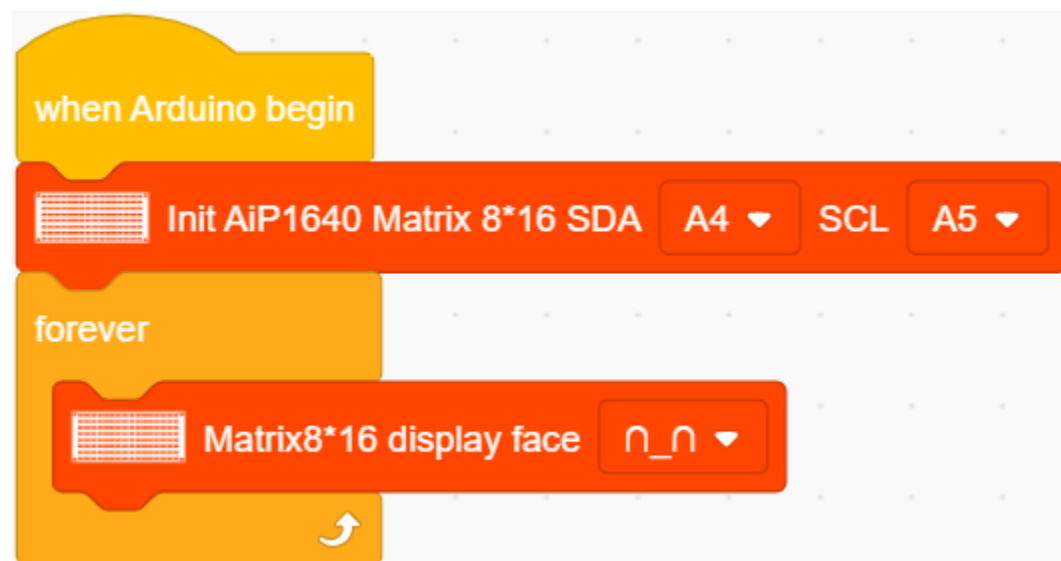
(Note: though it is connected with the IIC pin of Arduino, this module is not for IIC communication. And the IO port here is to simulate I2C communication and can be connected with any two pins)

(6)Test Code:

You can also drag blocks to edit your code, as shown below

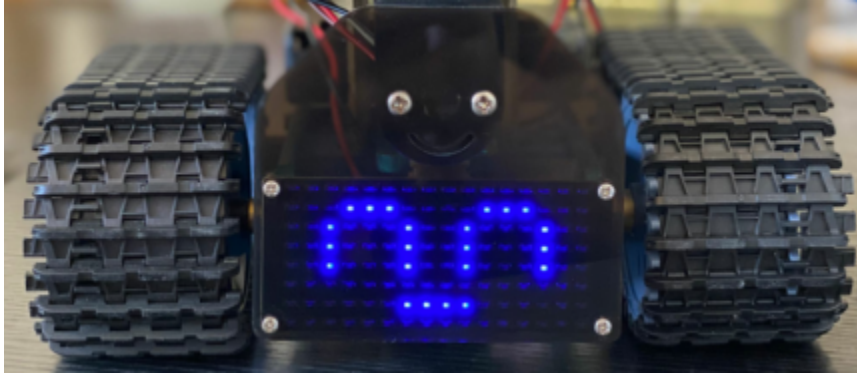
**Complete Test Code**

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

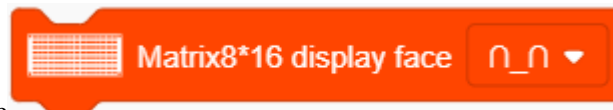


(7)Test Results:

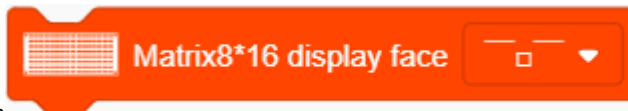
After uploading the test code successfully, wire up, turn the DIP switch to the ON end and power up, a smile-shaped pattern shows on the dot matrix.

**(8)Extension Practice:**

We use the modulus tool we just learned, <http://dotmatrixtool.com/#>, to make the dot matrix display the pattern start , go forward, and stop and then clear the pattern. The time interval is 2000 ms.



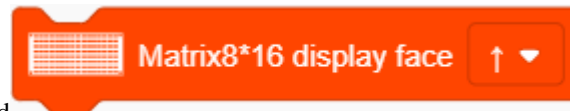
Block to show smile face



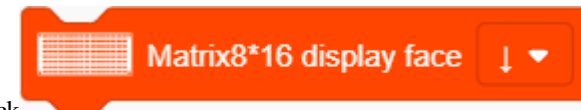
Code to show expression



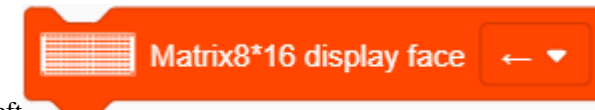
Block to show heart



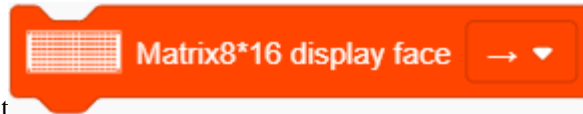
Code for moving forward



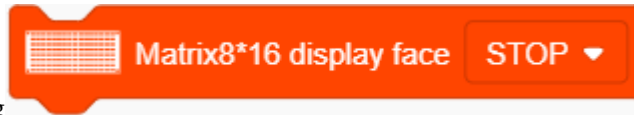
Block for going back



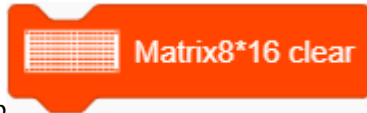
Block for turning left



Block for turning right

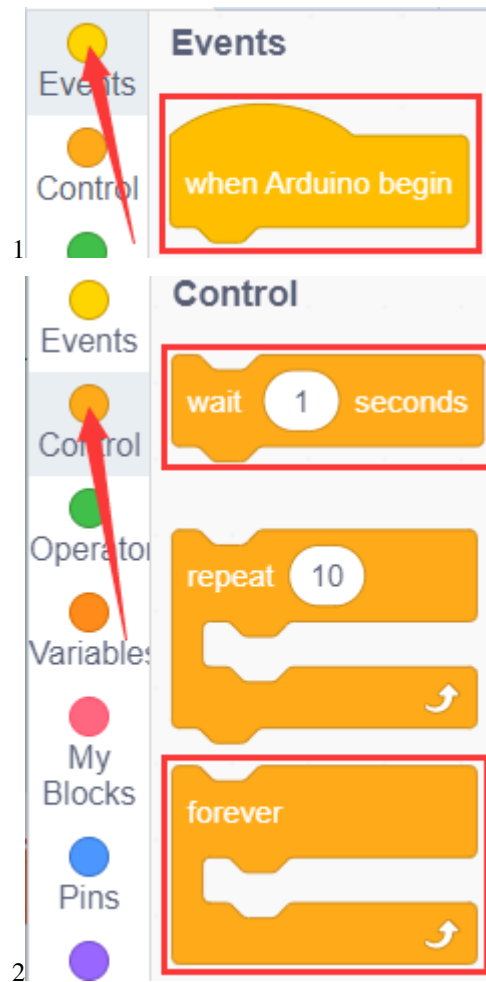


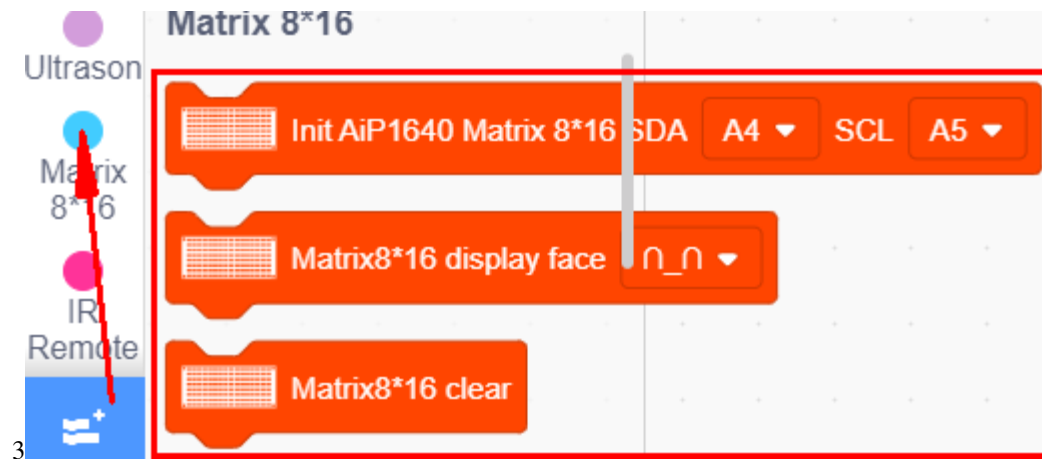
Block for stopping



Block for clearing up

You can also drag blocks to edit your code, as shown below



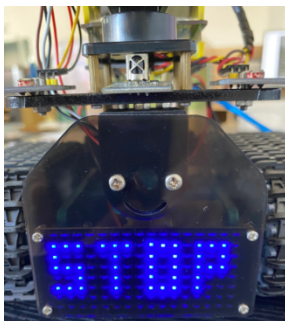


Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



Upload the code to the development board, the 8*16 board will show following patterns.



7.3.10 Project 10: Light-following Tank

(1)Description:

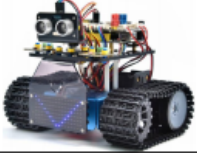



In previous projects, we introduced in detail the use of various sensors, modules, and expansion boards on the smart car. Now let's move to the projects of the smart car. The light-following smart cars, as the name suggests, is a smart car that can follow the light.

We can combine the knowledge from projects photoresistor and motor drive to make a light-seeking smart car. In the project, we use two photoresistor modules to detect the light intensity on the left and right sides of the smart car, read the corresponding analog values, and then control the rotation of the two motors based on these two data so as to control the movements of the smart car.

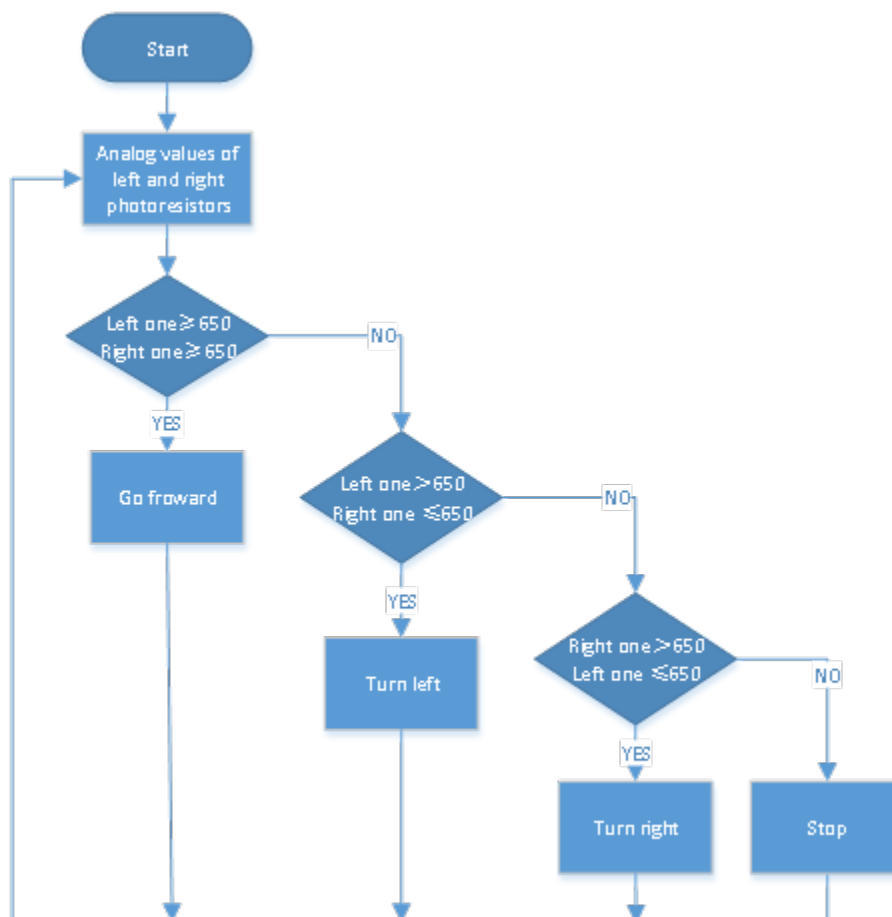
The specific logic of the light-following smart car is shown as below.

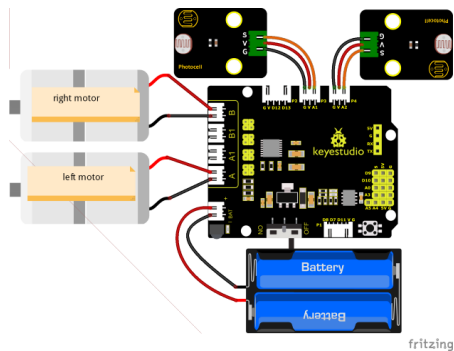
Detection (the bigger the brightness, the bigger the value)	photoresistor module on the left	left_light
	photoresistor module on the right	right_light
Condition	left_light > 650 and right_light > 650	
Movement	Move forward (set PWM to 200)	
Condition	left_light > 650 and right_light ≤ 650	
Movement	Rotate left (set PWM to 200)	
Condition	left_light ≤ 650 and right_light > 650	
Movement	Rotate right (set PWM to 200)	
Condition	left_light ≤ 650 and right_light ≤ 650	
Movement	stop	

(2) Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3) Flow chart:

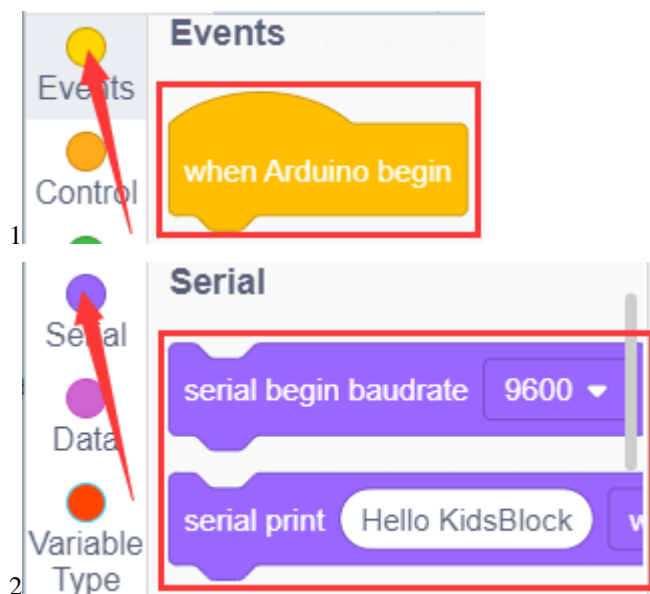


(4)Connection Diagram:

Note: The pin “G”, “V” and S of the left photoresistor module are connected to G (GND), V (VCC), A1 respectively; The pin “G”, “V” and S of the right photoresistor module are connected to the G (GND), V (VCC), and A2 respectively. The 4pin cable is marked with A, A1, B1 and B. The right rear motor is connected to B port of the 8833 motor driver expansion board and the left front motor is connected to A port of the 8833 motor driver expansion board.

(5)Test Code:

You can also drag blocks to edit your code, as shown below



3

My Blocks
Pins
Serial
Data
Variable Type
TEXT

Pins

- set pin 0 mode input
- set digital pin 0 out high
- set pwm pin 3 out 255
- read digital pin 0
- read analog pin A0

4

Events
Control
Operator
Variables
My Blocks
Pins
Serial
Data

Control

- forever
- if then
- if then else

Variable Type

Declare Global variable Type int Name item Assigned to 0

variable item

Set item variable by 0

Events

Control

Operator

Variables

My Blocks

Pins

DC Motor

servo

Ultrason

5

6

7

DC Motor

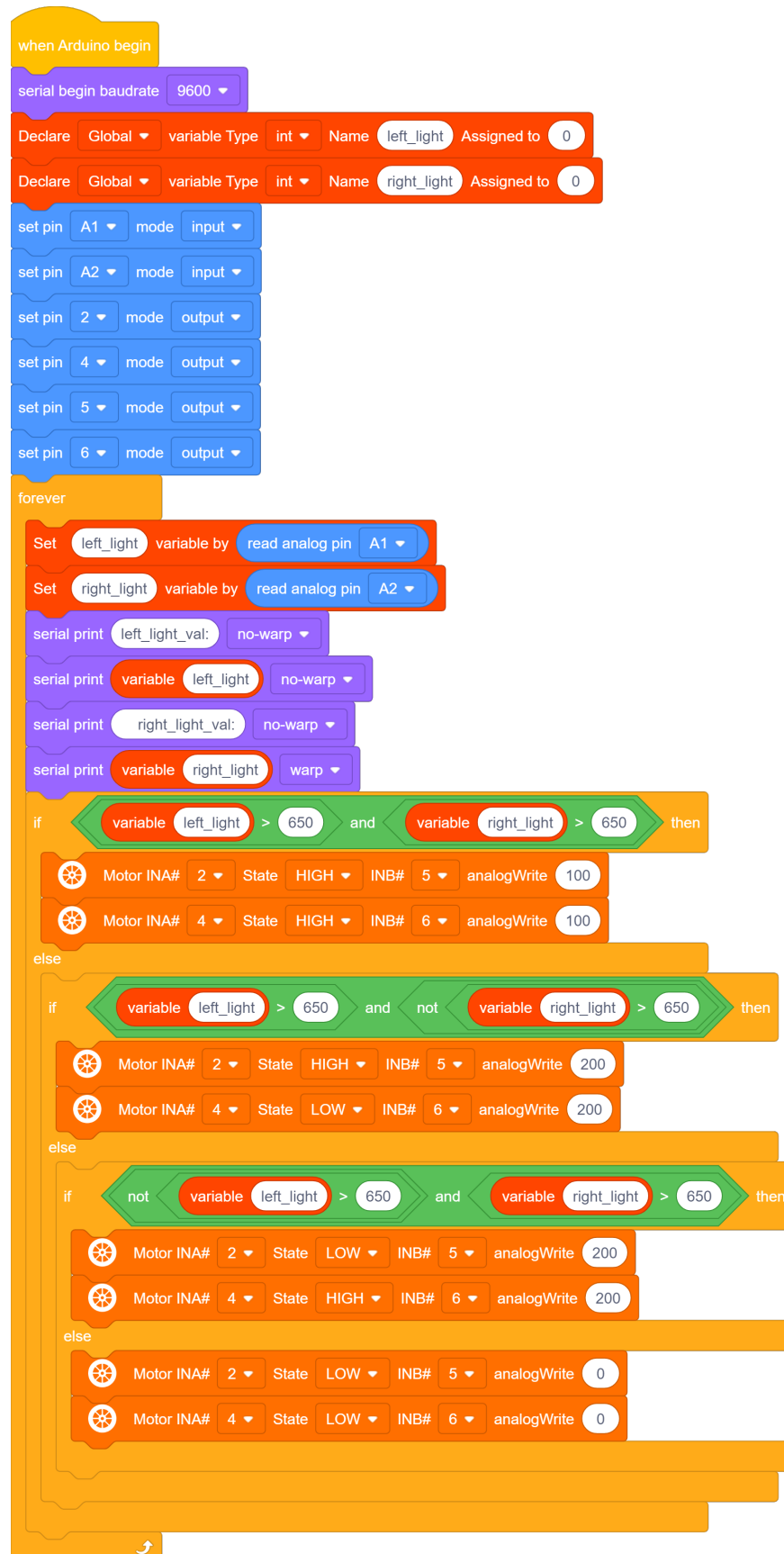
Motor INA# 2 State HIGH INB# 6 State HIGH

Motor INA# 2 State HIGH INB# 6 analogWrite 255

Complete Test Code

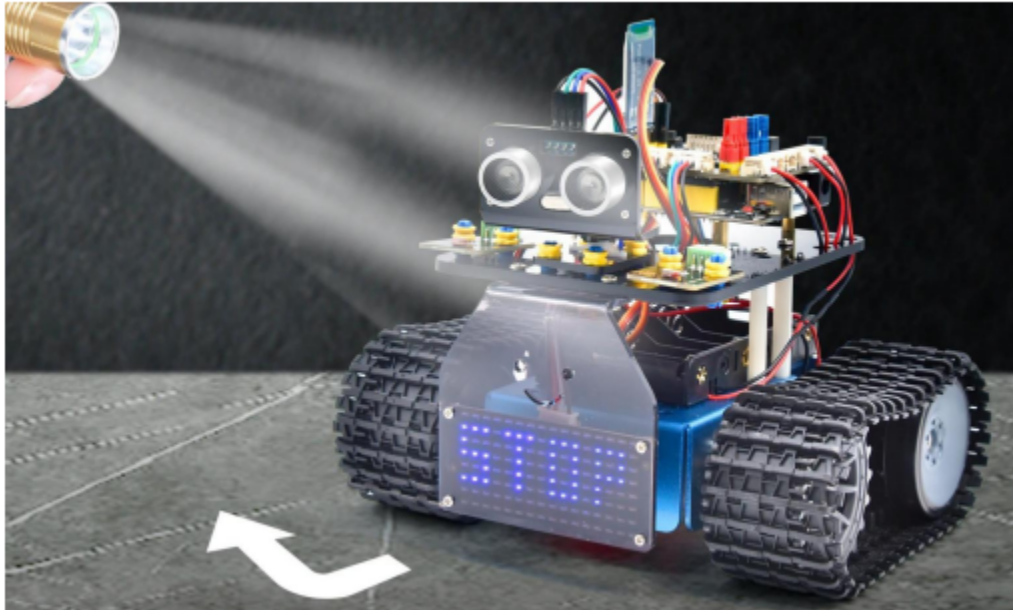
(Note: The threshold 650 in the code can be adjusted appropriately according to the specific light intensity.)

Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Results:

After uploading the test code successfully, wire up, turn DIP switch to the ON end and power on, the smart car follows the light to move.

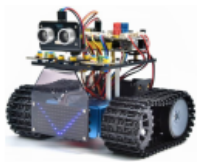



**7.3.11 Project 11: Ultrasonic Sound-following Tank****(1)Description:**

In the previous lesson, we learned about the light-following smart car. And in this lesson, we can combine the knowledge to make an ultrasonic sound-following car. In the project, we use ultrasonic sensors to detect the distance between the car and the obstacle in front, and then control the rotation of the two motors based on this data so as to control the movements of the smart car.

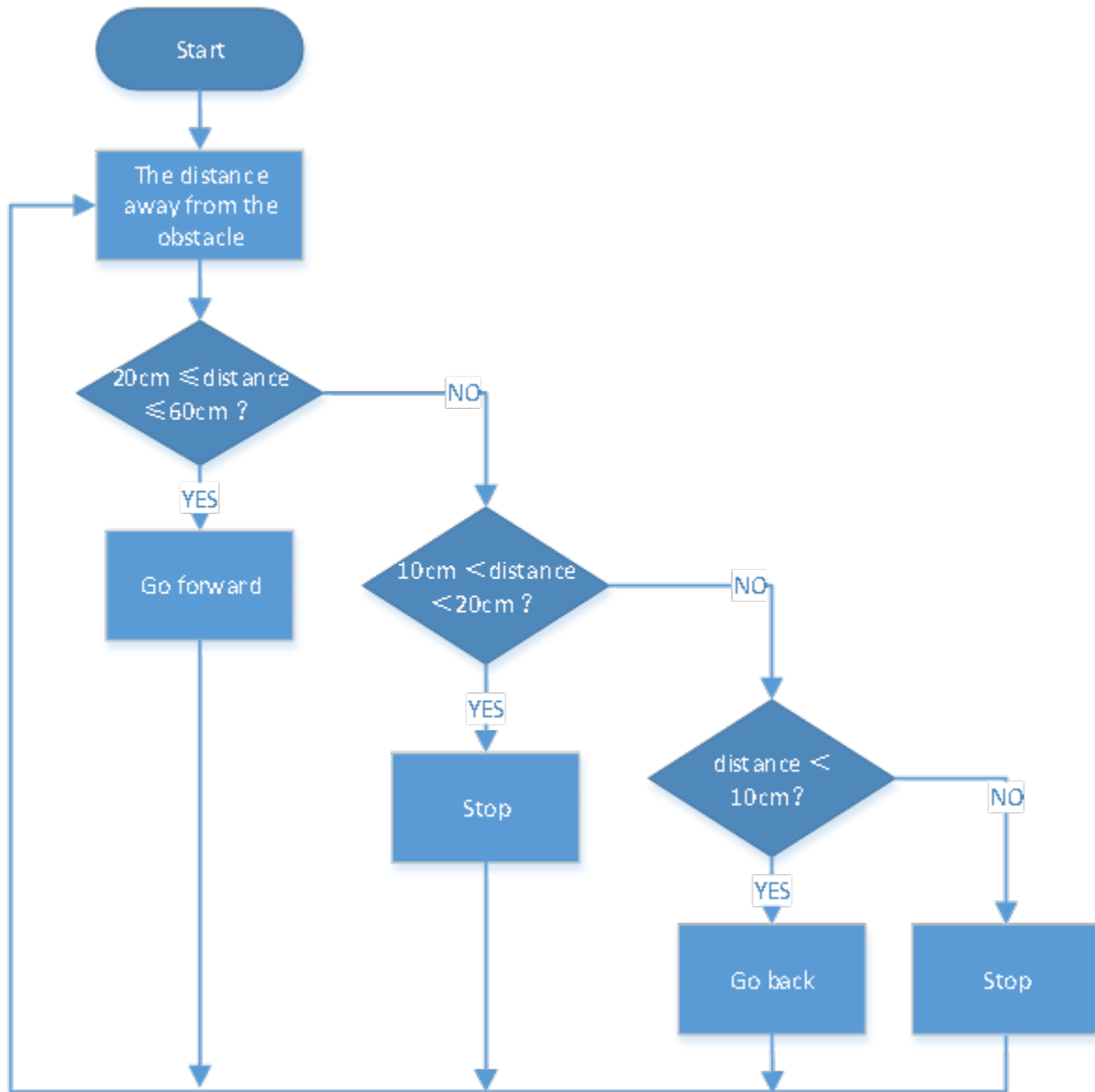
The specific logic of the ultrasonic sound- following smart car is shown in the table blow:

Detection	The distance between the car and the obstacle front	Distance (unit: cm)
Setting	Set the angle of the servo to 90°	
Condition	distance \geq 20 and distance \leq 60	
Movement	(set PWM to 200)	
Condition	distance > 10 and distance < 20	
	distance > 60	
Movement	Stop	
Condition	distance \leq 10	
Movement	Move back (set PWM to 200)	

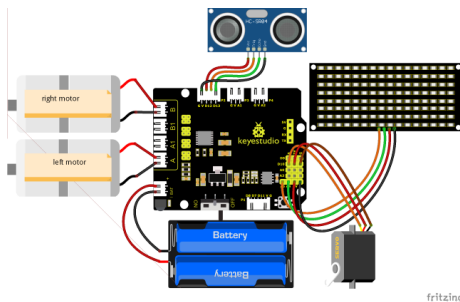
(2)Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:



(4)Connection Diagram:



Note: The wiring of the ultrasonic sensor, the servo and the motor is the same as the previous project experiment. The GND, VCC, SDA, and SCL of the 8x16 LED panel are respectively connected to G (GND), V (VCC), A4, and A5 on the expansion board

(5)Test Code:

You can also drag blocks to edit your code, as shown below

The screenshot displays the KidsBlock IDE interface with the following code blocks:

- Events Section:**
 - when Arduino begin** (Yellow block)
 - forever** loop (Orange block) containing:
 - if** block (Orange block) with a condition (represented by a diamond) and a **then** clause.
- Pins Section:**
 - set pin** block (Blue block) with pin number **0** and mode **input**.
 - set digital pin** block (Blue block) with pin number **0** and output **high**.
- Serial Section:**
 - serial begin baudrate** block (Purple block) with baudrate **9600**.
 - serial print** block (Purple block) with the text **Hello KidsBlock**.

Red arrows on the left sidebar indicate the categories used: **Events** (yellow), **Pins** (blue), and **Serial** (purple).

Variable Type

5

Variable Type

TEXT

DC Motor

5

Variable Type

Global variable Type int Name item Assigned to 0

variable item

Set item variable by 0

servo

6

servo

Ultrason

Matrix

6

servo

Ultrason

Matrix

6

DC Motor

DC Motor

servo

Ultrason

(7)

Ultrason

Matrix

(8)

servo PIN# 10 degree 90 delay 200

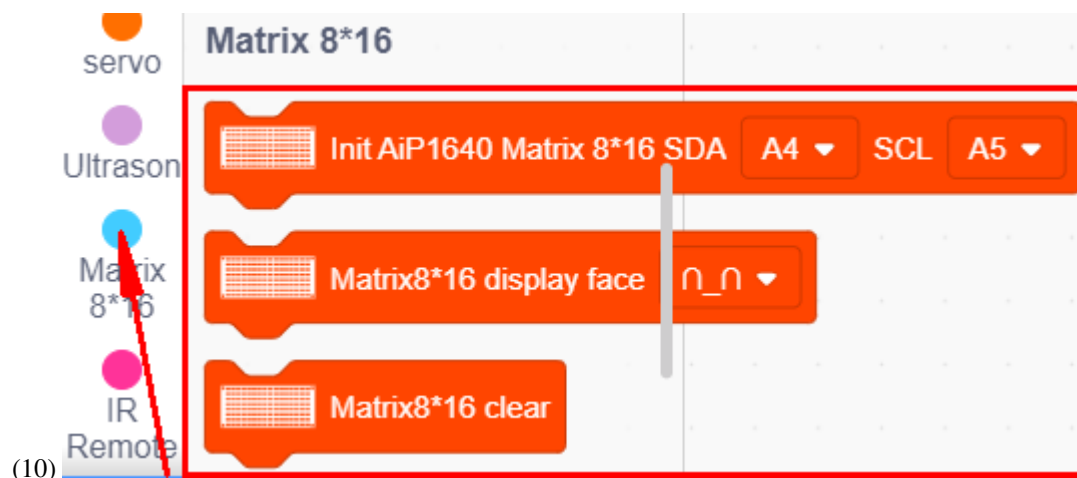
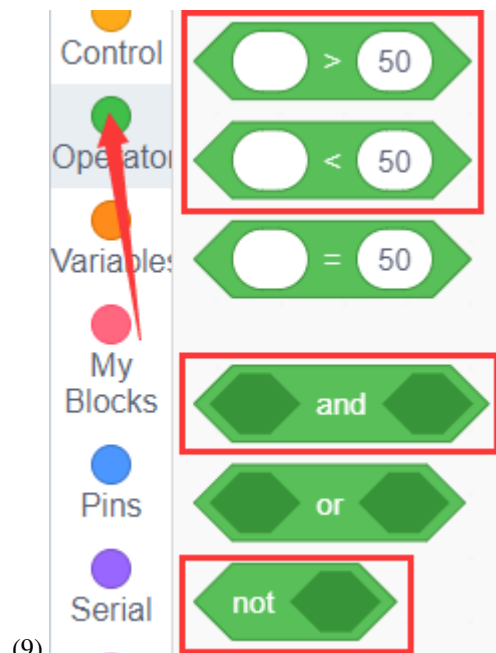
servo PIN# 10 read degree

Motor INA# 2 State HIGH INB# 6 State HIGH

Motor INA# 2 State HIGH INB# 6 analogWrite 255

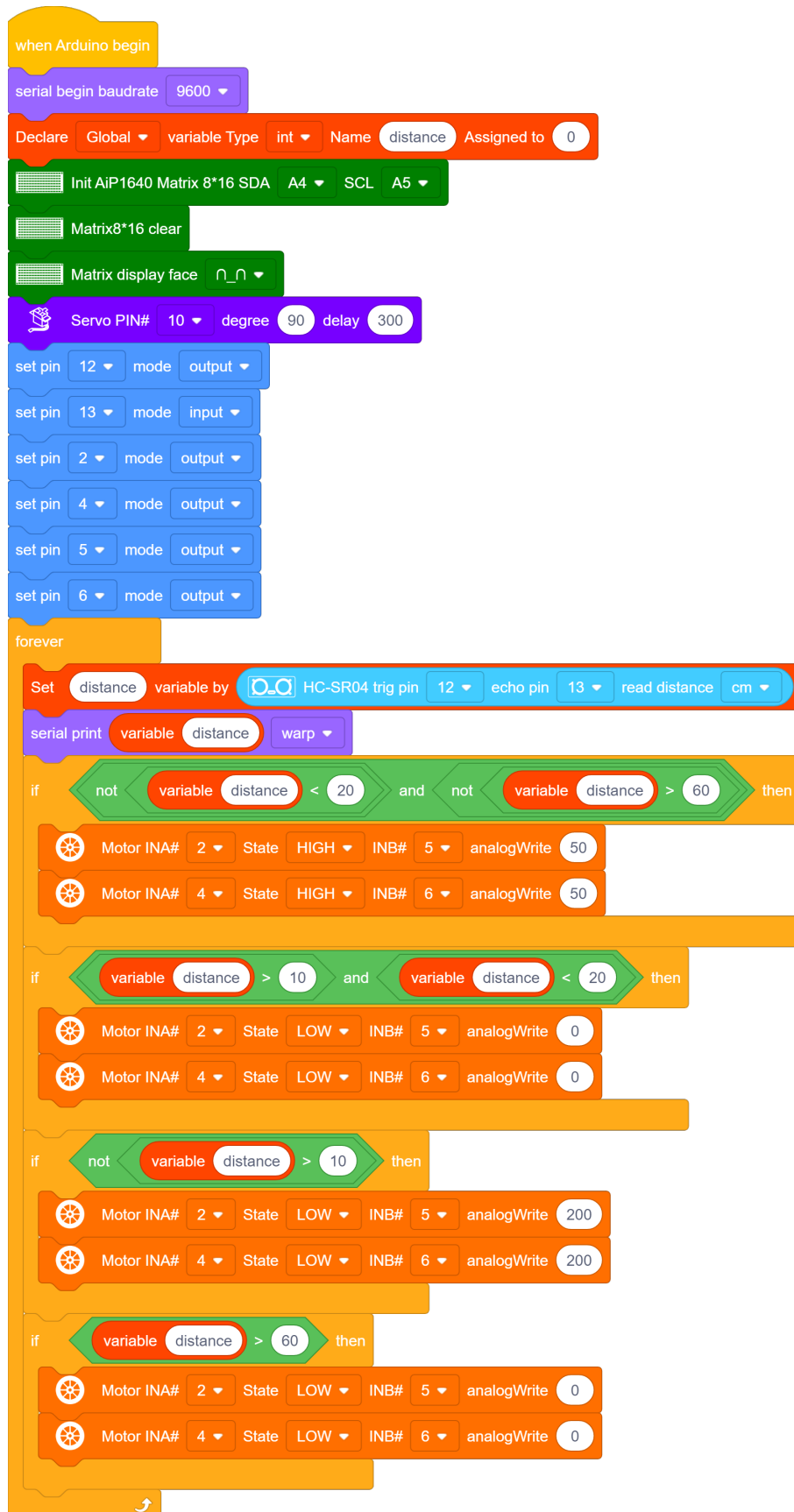
Ultrasonic

HC-SR04 trig pin 12 echo pin 13 read distance cm




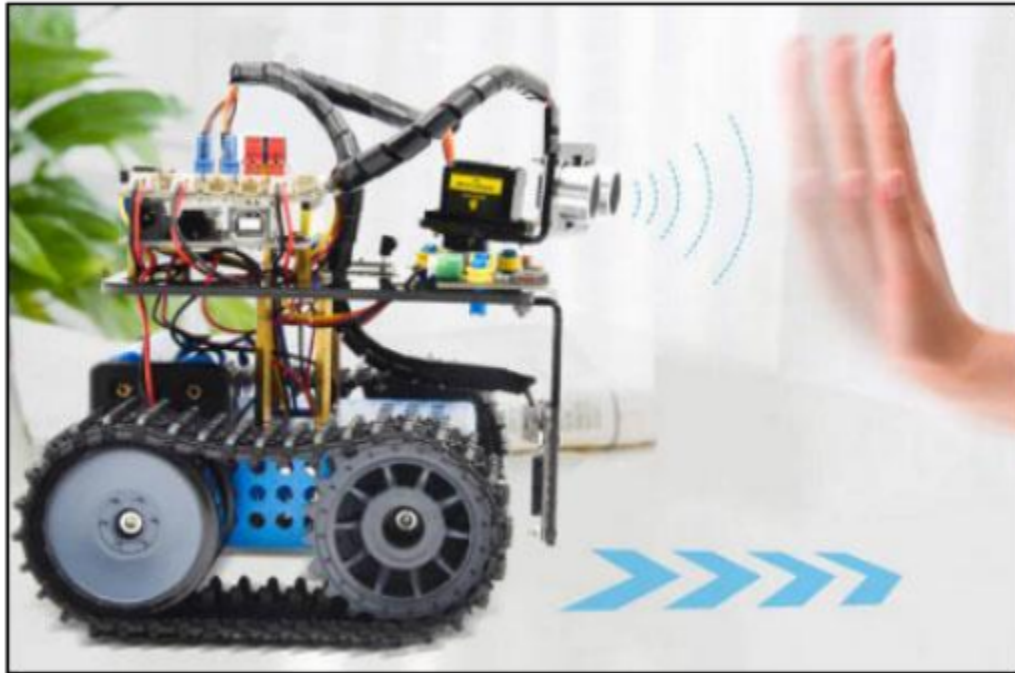
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Result:

Upload the code, power up and turn the DIP switch to ON. The servo will rotate 90°, the 8X16 LED panel will show  and the car follows the obstacle to move.



7.3.12 Project 12: Ultrasonic Obstacle Avoidance Tank

(1)Description:

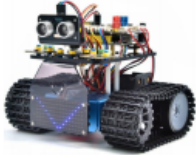



In the previous project, we made an ultrasonic sound-following smart car. In fact, using the same components and the same wiring method, we only need to change the test code to turn it into an ultrasonic obstacle avoidance smart car. This smart car can move with the movement of the human hands. We use ultrasonic sensors to detect the distance between the smart car and the obstacle in front, and then control the rotation of the two motors based on this data so as to control the movements of the smart car.

Detection	
Distance measured by the ultrasonic sensor between the car and the obstacle in front set the angle of the servo to 90°	a (cm)
Distance measured by the ultrasonic sensor between the car and the obstacle on the right set the angle of the servo to 0°	a2 (cm)
Distance measured by the ultrasonic sensor between the car and the obstacle on the left set the angle of the servo to 180°	a1 (cm)

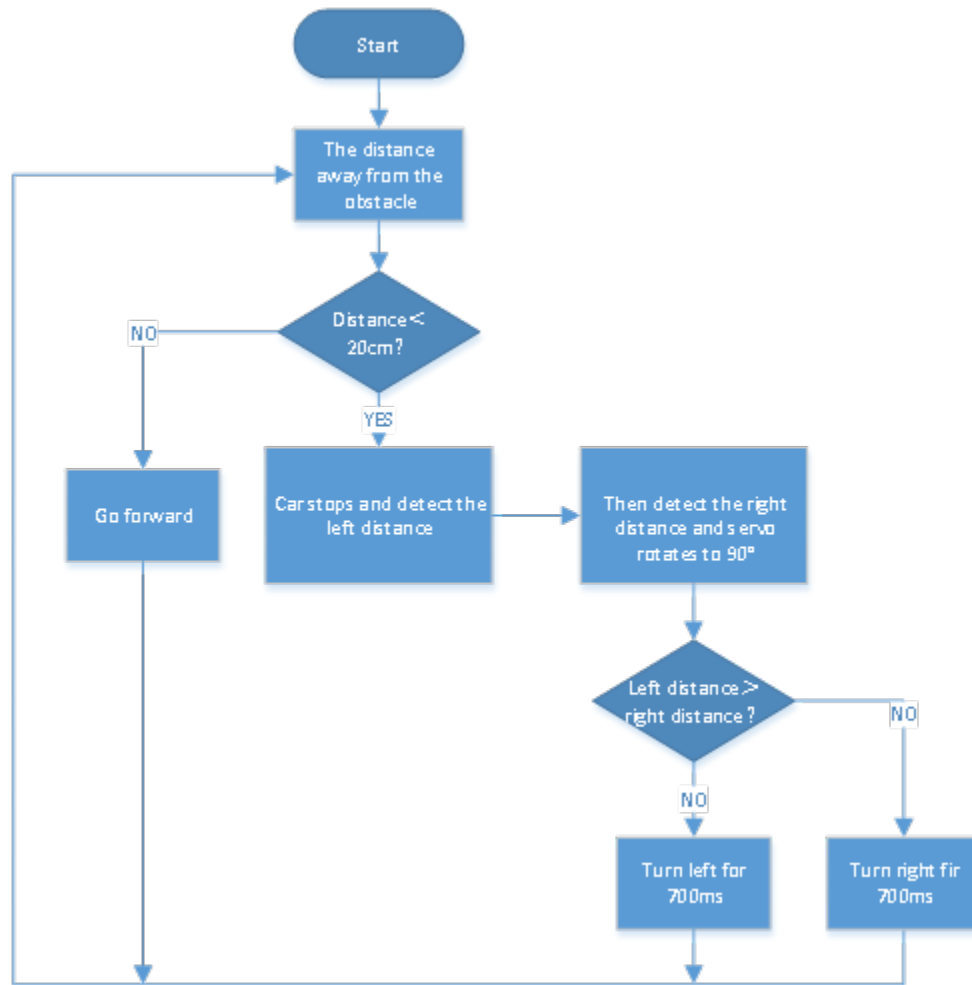
Setting: set the starting angle of the servo to 90°

Con- di- tion 1	Con- di- tion 2	Condi- tion 3	Movement
a20			Stop for 500msset the angle of the servo to 180°read a1delay in 100msset the angle of the servo to 0°read a2delay in 0.1s.
	a150ora250	Compare a1 with a2	
		a1a2	Set the angle of the servo to 90°rotate left for 700msset PWM to 255and move forwardset PWM to 200.
		a1a2	Set the angle of the servo to 90°rotate right for 700msset PWM to 255and move forwardset PWM to 200.
	a150and a250	Random	set the angle of the servo to 90°rotate left for 500msset PWM to 255and move forwardset PWM to 200.set the angle of the servo to 90°rotate right for 500msset PWM to 255and move forwardset PWM to 200.
a20			move forwardset PWM to 100

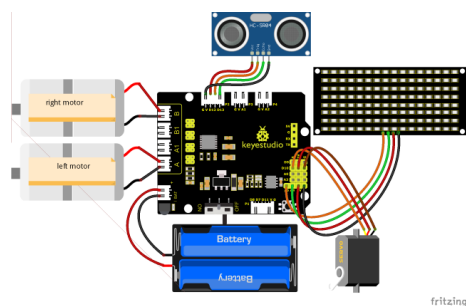
(2)Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:



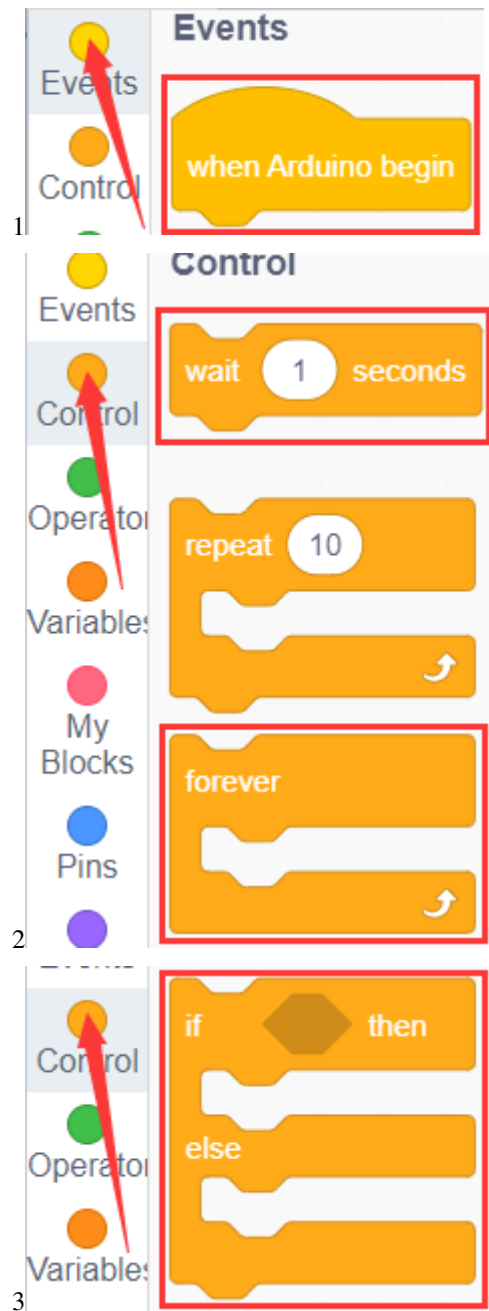
(4)Connection Diagram:



(Note: the brown, red and orange wires of the servo are respectively connected to G (GND), 5V and D10 of the expansion board and for the ultrasonic sensor, the VCC pin is connected to the 5v (V), the Trig pin to digital 12 (S), the Echo pin to digital 13 (S), and the Gnd pin to Gnd (G); the same as last project.

(6)Test Code:

You can also drag blocks to edit your code, as shown below



4

Pins

set pin 0 mode input

set digital pin 0 out high

5

Serial

serial begin baudrate 9600

serial print Hello KidsBlock

6

Variable Type

Declare Global variable Type int Name item Assigned to 0

variable item

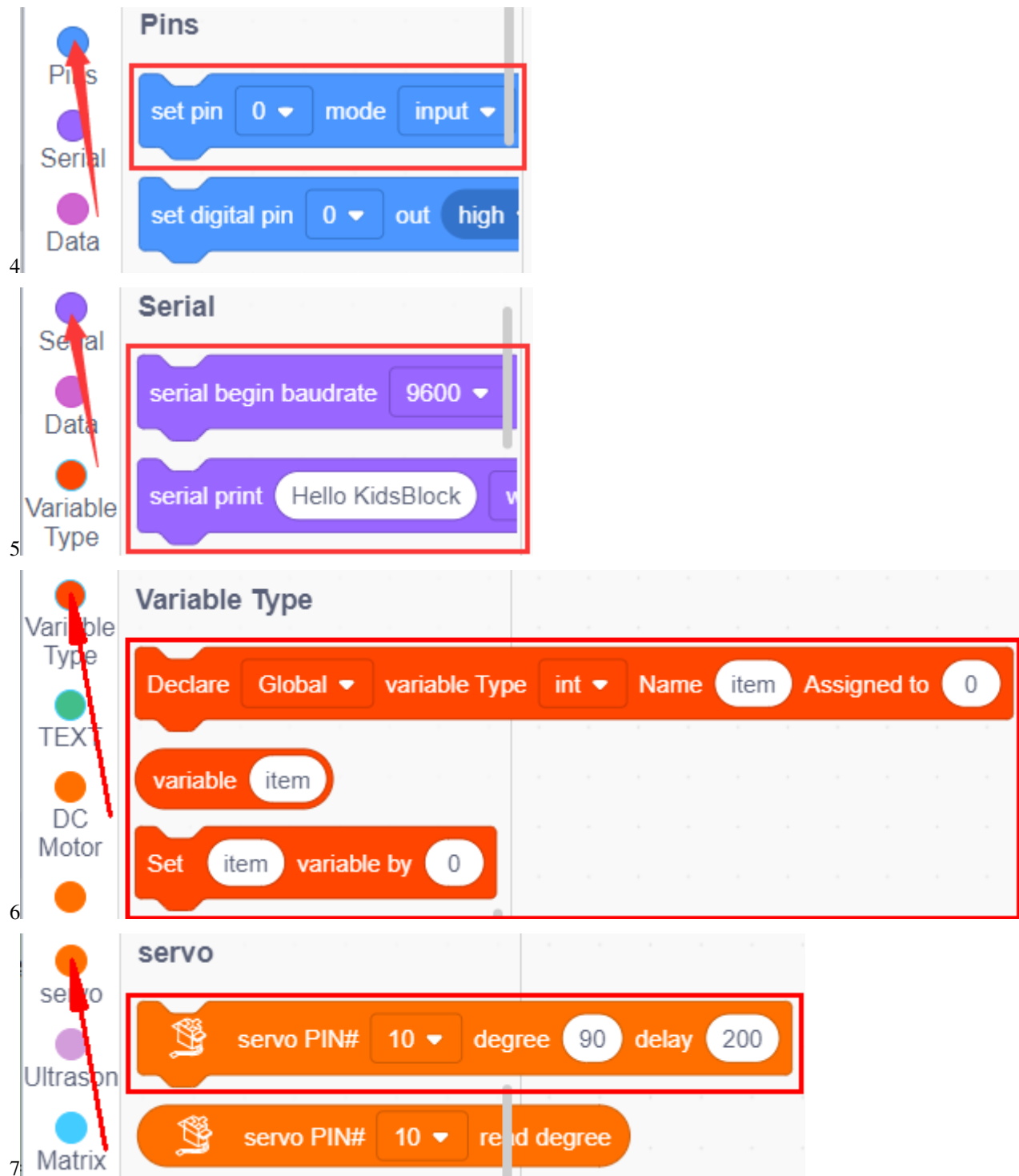
Set item variable by 0

7

servo

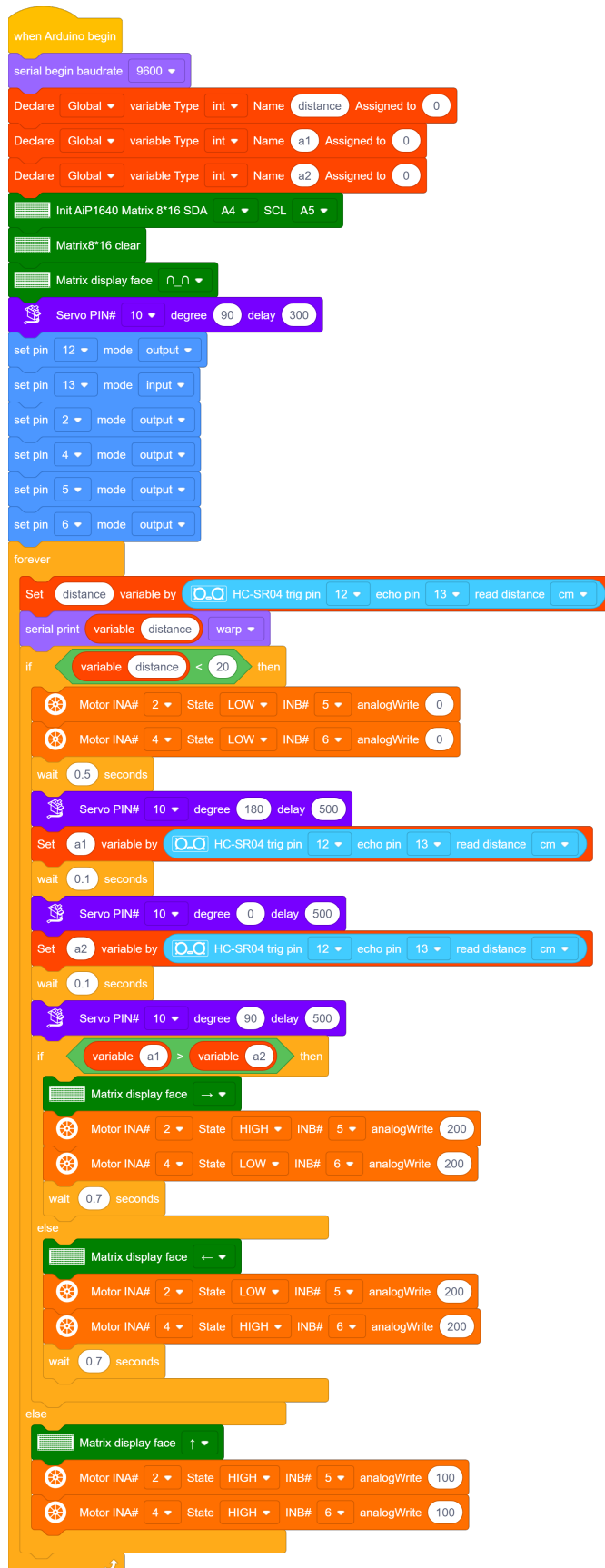
servo PIN# 10 degree 90 delay 200

servo PIN# 10 read degree



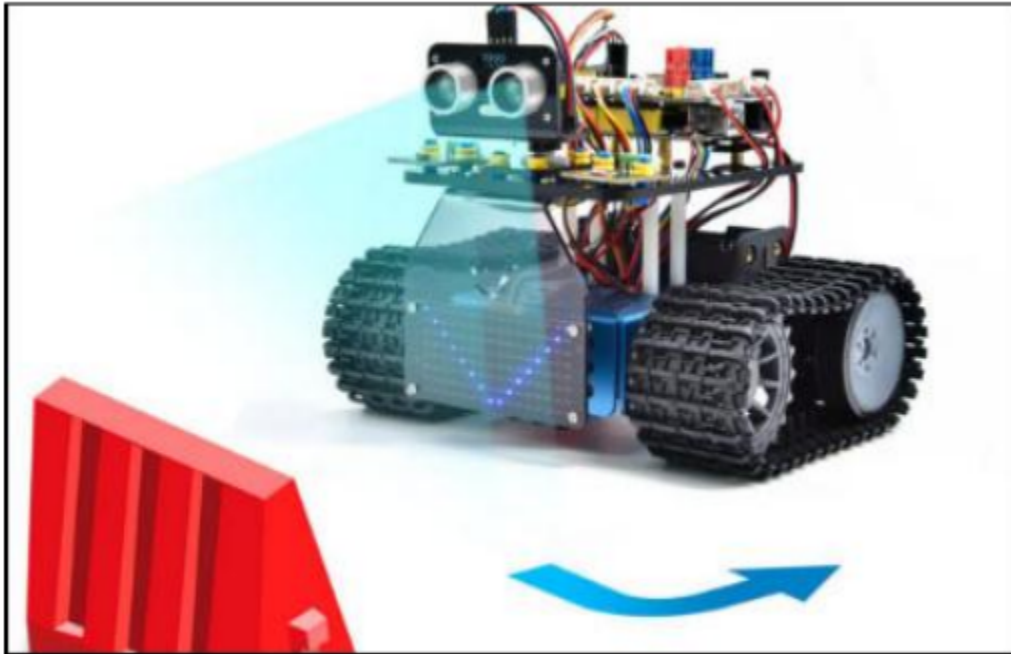
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Results:

After upload the test code successfully, wire up, turn the DIP switch to the ON end, and power up, the smart car moves forward and automatically avoids obstacles.



7.3.13 Project 13: Move-in-Confinned-Space Tank

(1)Description:

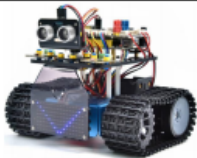

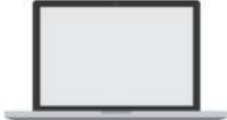

The ultrasonic sound-following and obstacle avoidance functions of the smart car have been introduced in previous projects. Here we intend to combine the knowledge in the previous courses to confine the smart car to move in a certain space. In the experiment, we use the line-tracking sensor to detect whether there is a black line around the smart car, and then control the rotation of the two motors according to the detection results, so as to lock the smart car in a circle drawn in black line.

The specific logic of the line-tracking smart car is shown in the table blow:

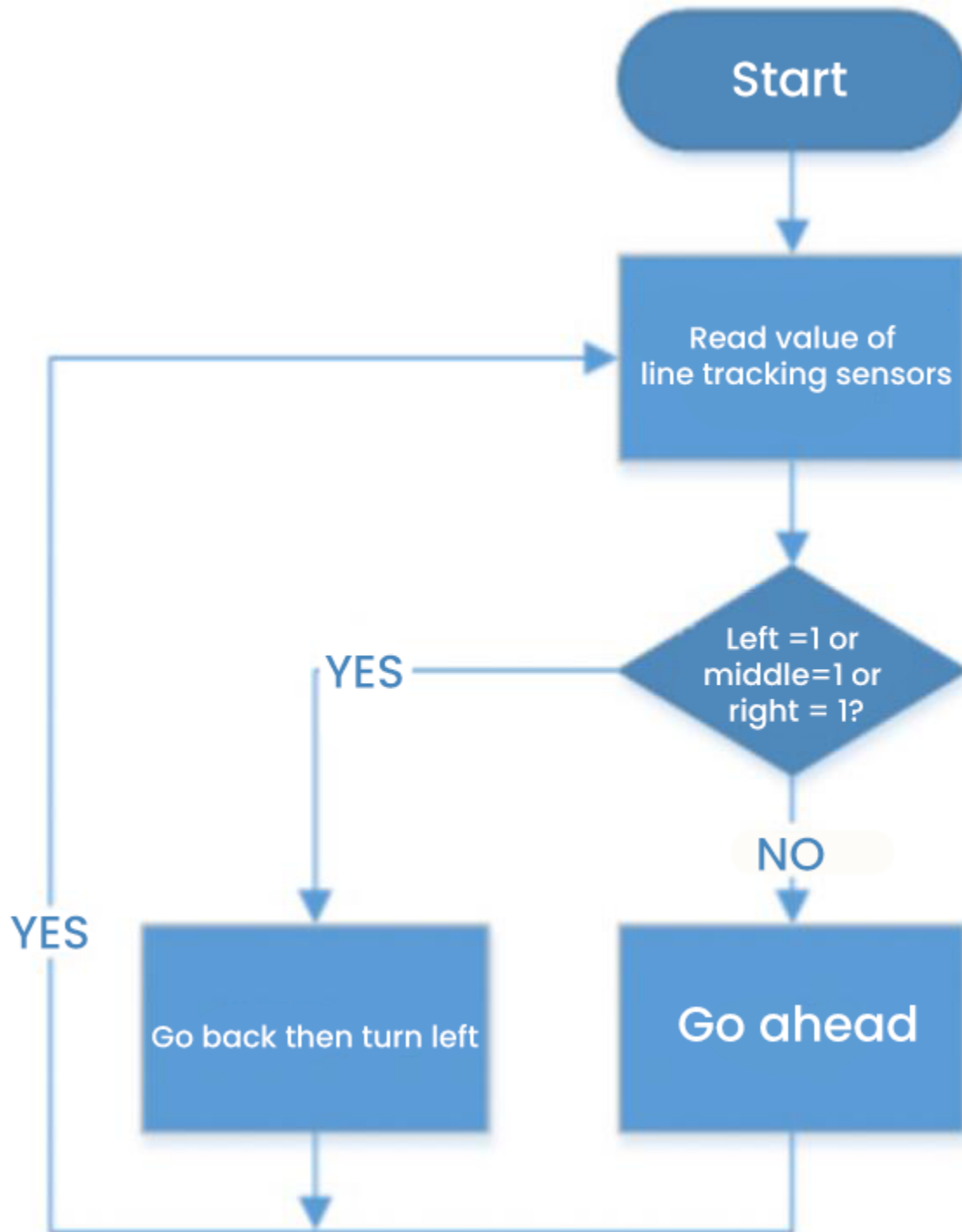
Detection	Line-tracking sensor in the middle	Black line detected: in high level
		White line detected: in low level
	Line-tracking sensor on the left	Black line detected: in high level
		White line detected: in low level
	Line-tracking sensor on the right	Black line detected: in high level
		White line detected: in low level

Condition	Movement
If one of three line tracking sensors detects black lines	Go backset PWM to 150Then turn leftset PWM to 150
None of them detects black lines	Go forwardset PWM to 100

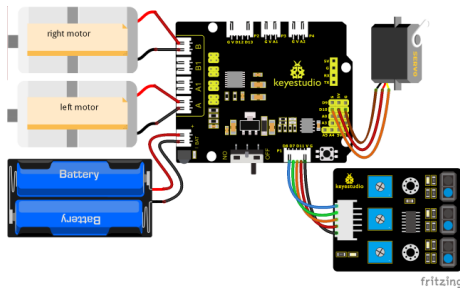
(2)Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart

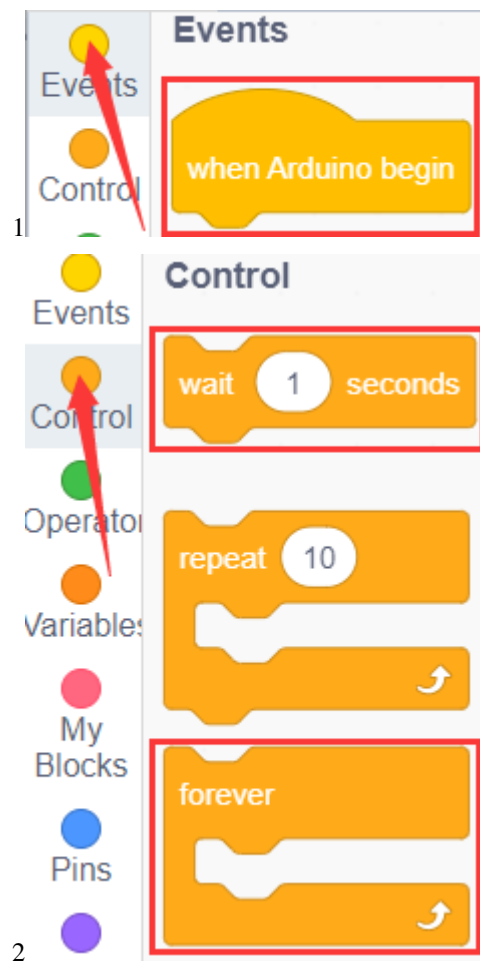


(4) Connection Diagram:



(5) Test Code:

You can also drag blocks to edit your code, as shown below



3

Control

Operator

Variables

if then else

4

My Blocks

Pins

Serial

Data

Variable Type

TEXT

5

Serial

Data

Variable Type

Pins

set pin 0 mode input

set digital pin 0 out high

set pwm pin 3 out 255

read digital pin 0

read analog pin A0

Serial

serial begin baudrate 9600

serial print Hello KidsBlock

Variable Type

- Declare Global variable Type int Name item Assigned to 0
- variable item
- Set item variable by 0

servo

- servo PIN# 10 degree 90 delay 200
- servo PIN# 10 read degree

DC Motor

- Motor INA# 2 State HIGH INB# 6 State HIGH
- Motor INA# 2 State HIGH INB# 6 analogWrite 255

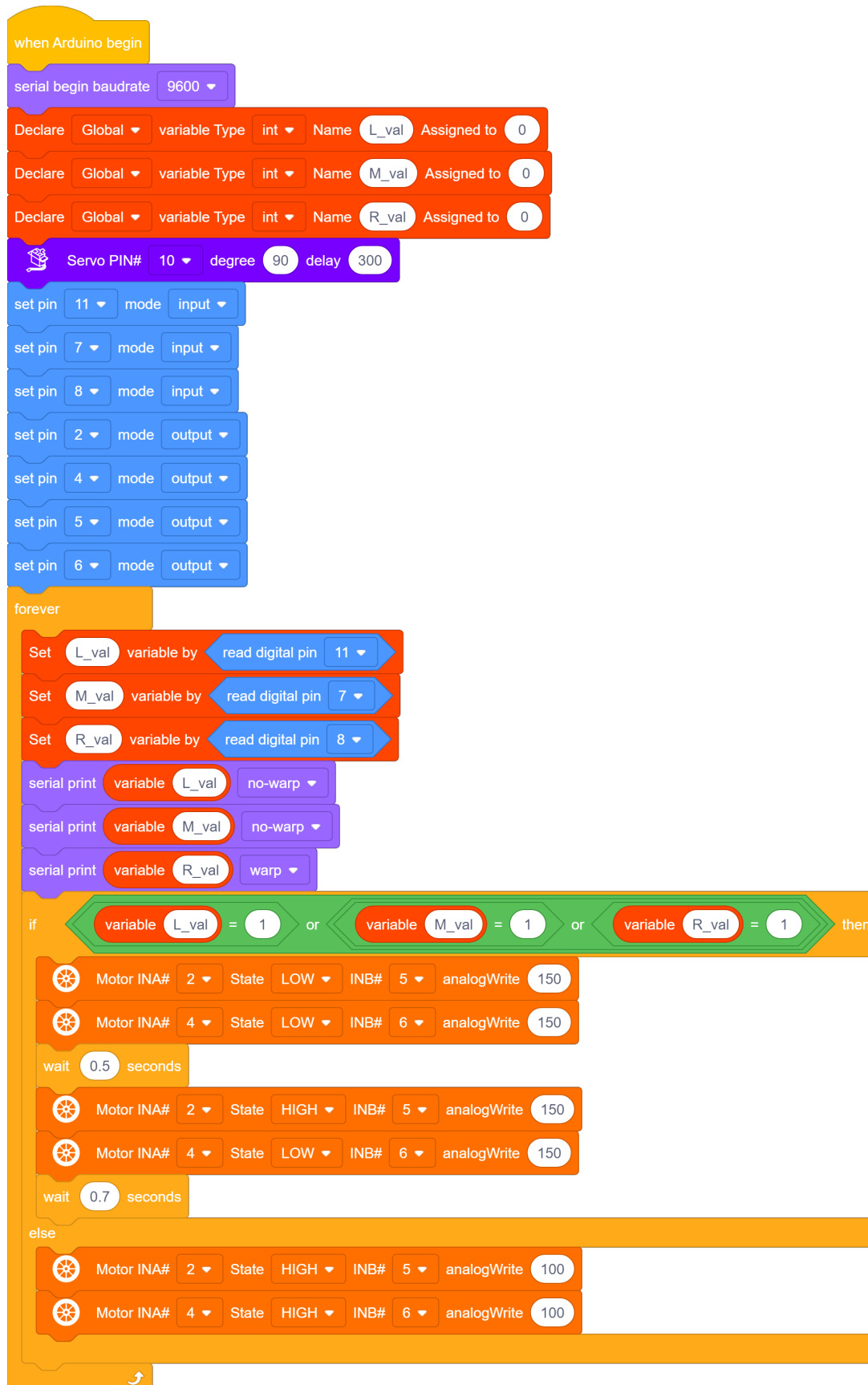
Operator

- < 50
- = 50
- and
- or

Complete Test Code

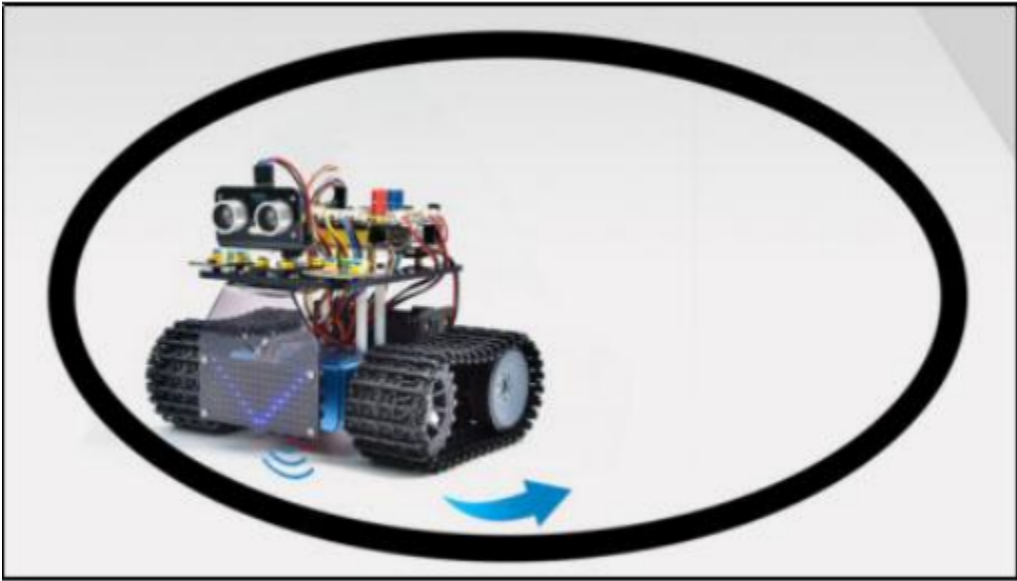
(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the

uploading of the code to fail.)



(6)Test Results:

After upload the test code successfully and power up, the smart car moves in a circle drawn in black line.



7.3.14 Project 14:Line-tracking Tank

(1)Description:

The previous project has introduced how to confine the smart car to move in a certain space. In this project, we could use the knowledge learned before to make it a line-tracking smart car. In the experiment, we use the line-tracking sensor to detect whether there is a black line around the smart car, and then control the rotation of the two motors according to the detection results, so as to make the smart car to move along the black line.

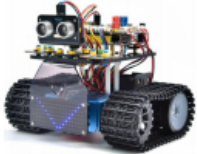

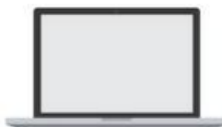

The specific logic of the line-tracking smart car is shown in the table blow:

Detection	Line-tracking sensor in the middle	Black line detected: in high level
		White line detected: in low level
	Line-tracking sensor on the left	Black line detected: in high level
		White line detected: in low level
	Line-tracking sensor on the right	Black line detected: in high level
		White line detected: in low level

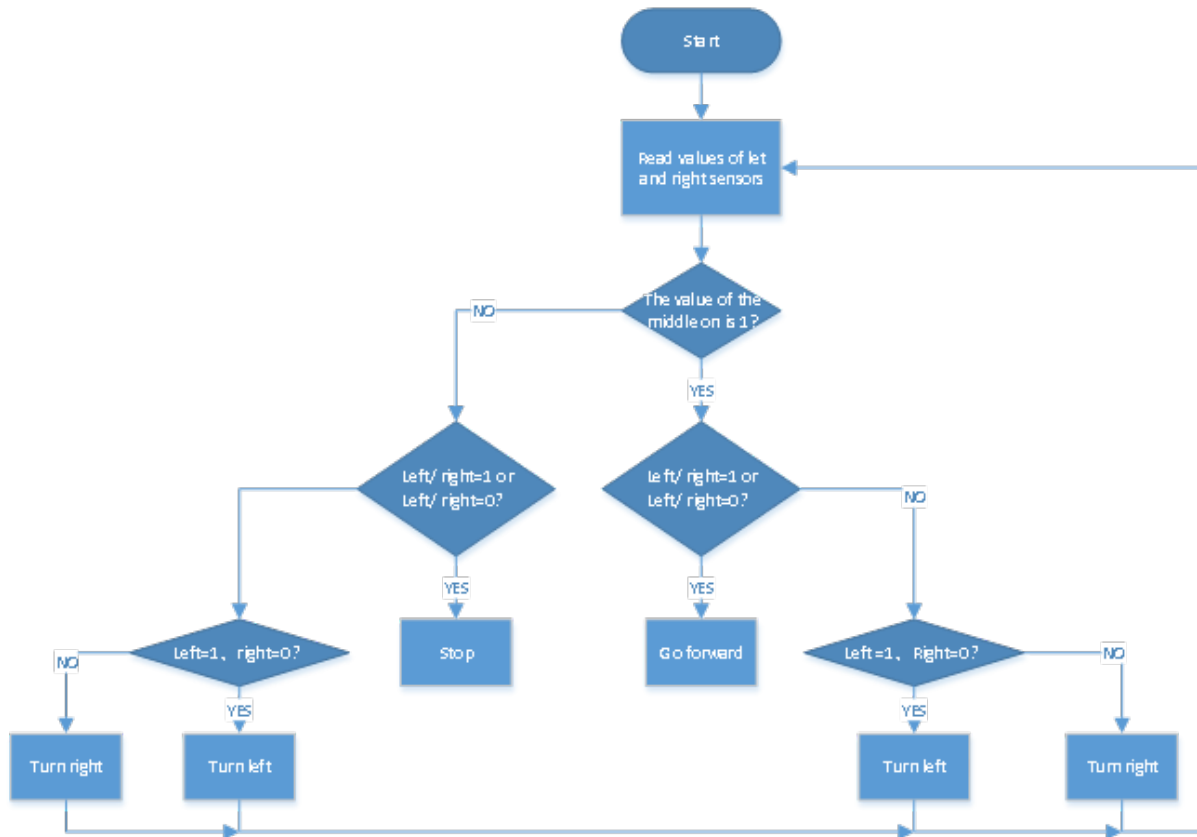
Condition 1	Condition 2	Movement
Line-tracking sensor in the middle detects the black line	Line-tracking sensor on the left detects the black line the one on the right detects white lines	Rotate left
	Line-tracking sensor on the left detects white lines the one on the right detects the black line	Rotate right
	Both the left and right line-tracking sensors detect white lines Both the left and right line-tracking sensors detect the black line	Move forward
Line-tracking sensor in the middle detects white lines	Line-tracking sensor on the left detects the black line the one on the right detects white lines	Rotate left
	Line-tracking sensor on the left detects white lines the one on the right detects the black line	Rotate right
	Both the left and right line-tracking sensors detect white lines Both the left and right line-tracking sensors detect the black line	Stop

The following flow chart shows the values the line tracking sensor detect.

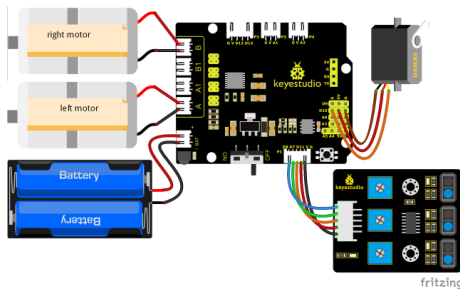
(2) Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	18650 Battery*2
			

(3)Flow chart:

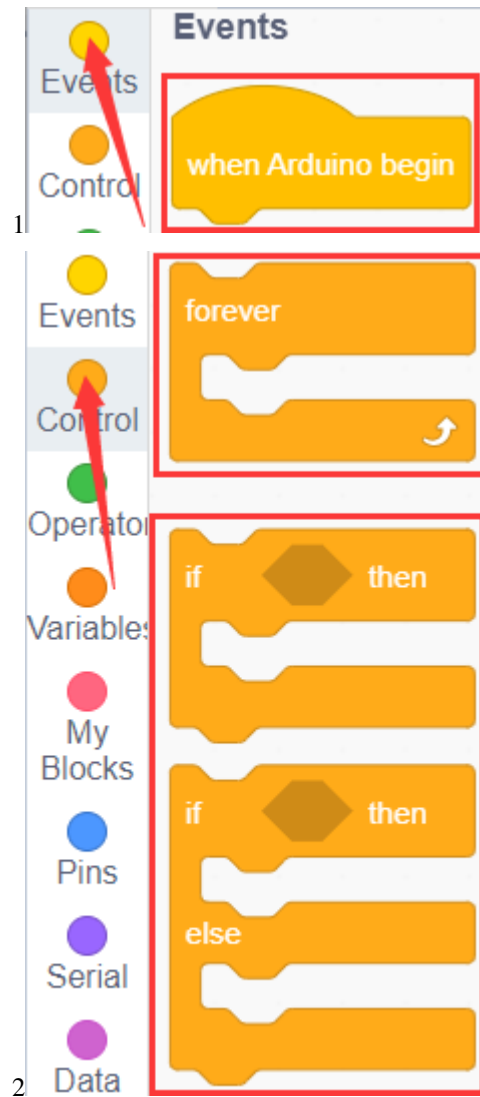


(4)Connection Diagram:



(5)Test Code:

You can also drag blocks to edit your code, as shown below



3

My Blocks

Pins

Serial

Data

Variable Type

TEXT

4

Serial

Data

Variable Type

5

Variable Type

TEXT

DC Motor

Pins

set pin 0 mode input

set digital pin 0 out high

set pwm pin 3 out 255

read digital pin 0

read analog pin A0

Serial

serial begin baudrate 9600

serial print Hello KidsBlock

Variable Type

Declare Global variable Type int Name item Assigned to 0

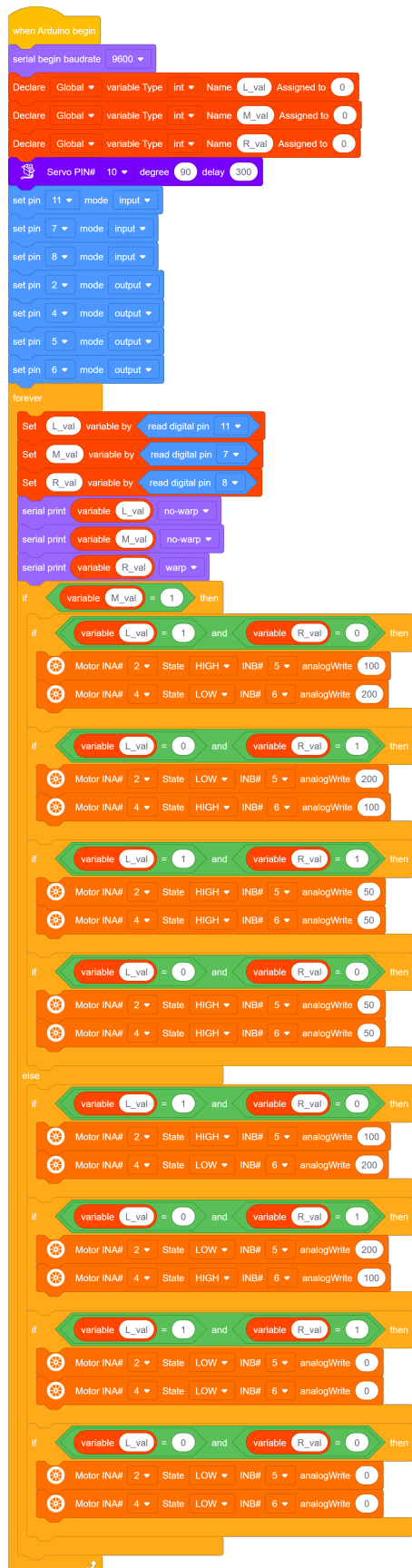
variable item

Set item variable by 0

The screenshot shows the Scratch IDE interface. On the left, the 'servo' and 'DC Motor' categories are highlighted with red arrows. The main workspace contains two code blocks. The first block is a 'servo' block with the following settings: servo PIN# 10, degree 90, delay 200, and a 'read degree' block. The second block is a 'DC Motor' block with the following settings: Motor INA# 2, State HIGH, INB# 6, State HIGH, and an 'analogWrite' block with value 255. The 'Operator' category is selected on the left, showing comparison blocks (>, <, =) and logical blocks (and, or, not). A red box highlights the '=' comparison block.

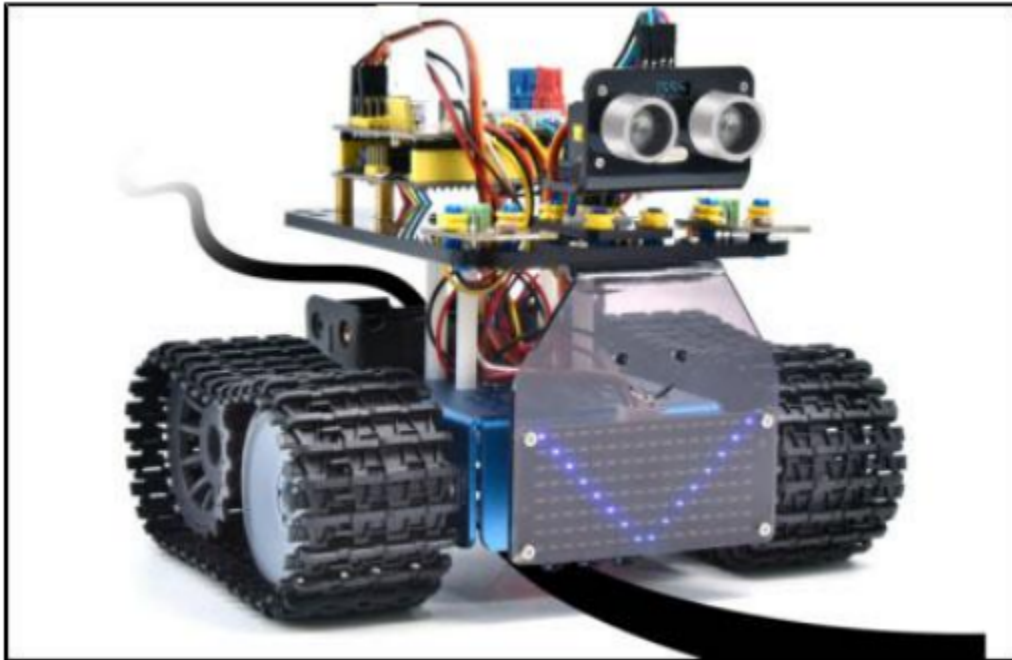
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Results:


After upload the test code successfully and power on, the smart car moves along the black line.






**7.3.15 Project 15: IR Remote Control Tank****(1)Description:**

Infrared remote control is one of the most common remote control found applications in electric motors, electric fans, and many other household appliances. In this project, we use the knowledge we learned before to make an infrared remote control smart car.

In the 9th lesson, we have tested the corresponding key value of each key of the infrared remote control. In the project, we can set the code (key value) to make the corresponding button to control the movements of the smart car, and display the movement patterns on the 8X16 LED dot matrix.

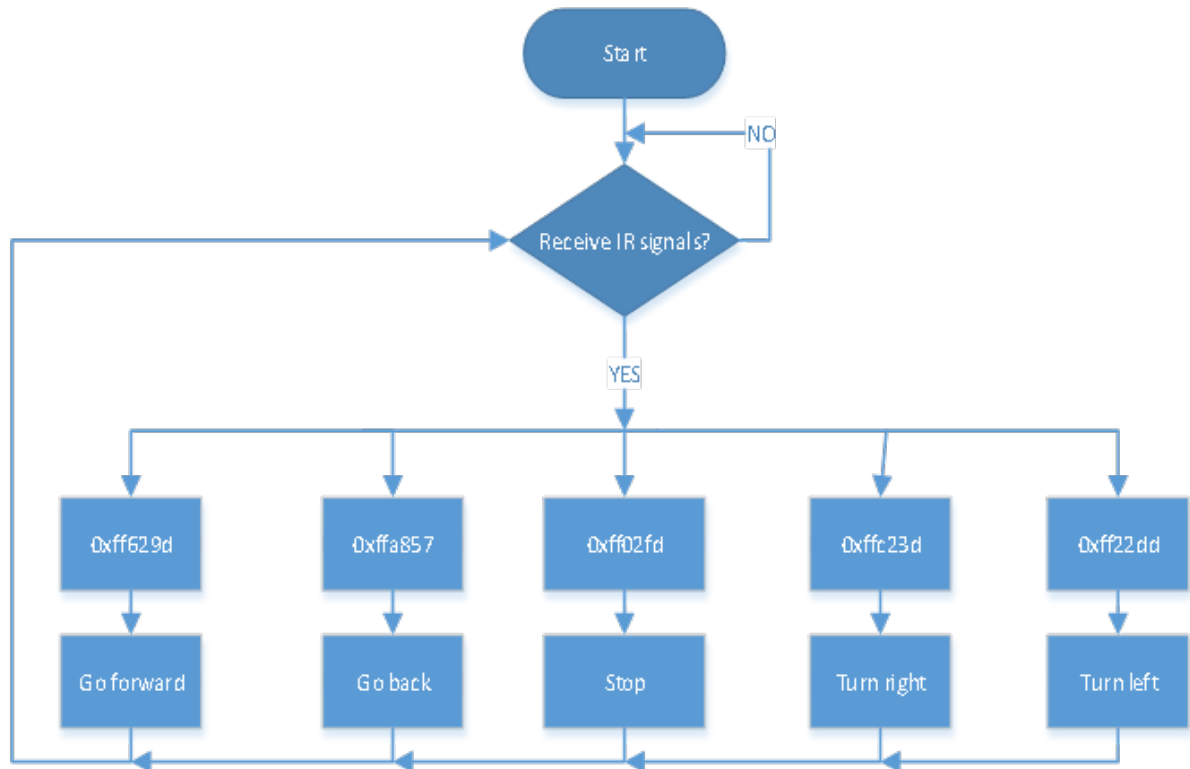
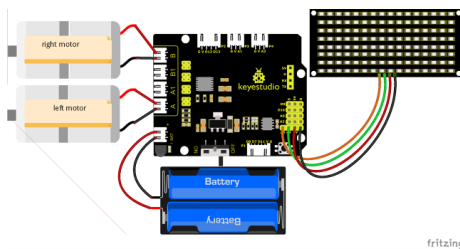
The specific logic of the line-tracking smart car is shown in the table:

Initial setting: 8X16 LED dot matrix shows the pattern“”

Ultrasonic key	Key value	Instructions from keys
	FF629D	Move forward set PWM to 200 display the pattern of going forward
	FFA857	Go back set PWM to 200 display the pattern of going back
	FF22DD	Turn left display the pattern "STOP"
	FFC23D	Turn right display the pattern of turning left
	FF02FD	Stop display the pattern "STOP"

(2) Components Needed:

Robot without BT module*1	USB Cable*1	Computer*1	Remote Control*1	18650 Battery*2
				

(3)Flow chart:**(4)Connection Diagram:**

Note:

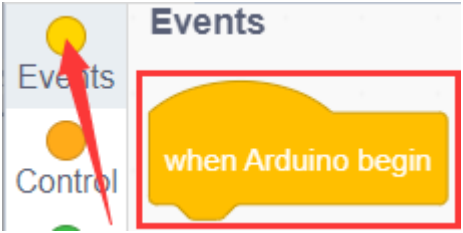
GND, VCC, SDA and SCL of the 8x16 LED panel are connected to GGND), VVCC). A4 and A5 of the expansion board.

Since the 8833 board integrates the IR receiver, you don't need to wire it up. The pins of the IR receiver are GGND), VVCC) and D3.

(5)Test Code:

You can edit blocks to build up your code

1

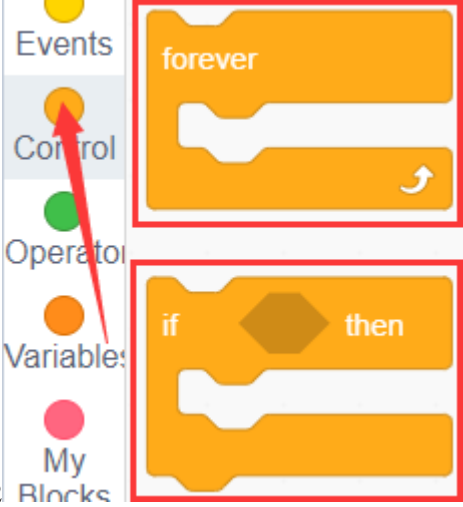


Events

when Arduino begin

Control

2



Events

forever

Control

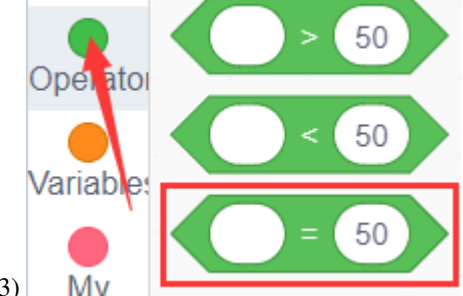
if then

Operator

Variables

My Blocks

(3)



Operator

> 50

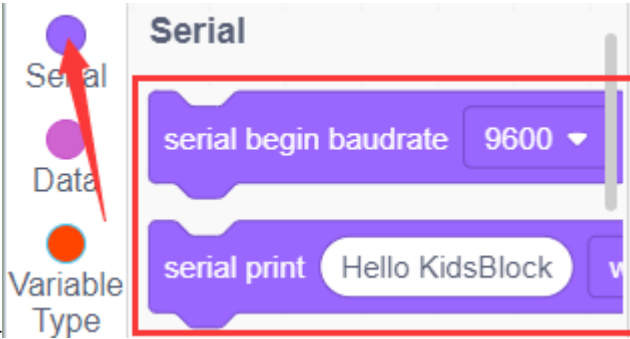
< 50

= 50

Variables

Mv

4



Serial

serial begin baudrate 9600

serial print Hello KidsBlock

Data

Variable Type

5

6

7

8

DC Motor

Motor INA# 2 State HIGH INB# 6 State HIGH

Motor INA# 2 State HIGH INB# 6 analogWrite 255

TEXT

char a

string hello

123

Matrix 8*16

Init AiP1640 Matrix 8*16 SDA A4 SCL A5

Matrix8*16 display face 0_0

Matrix8*16 clear

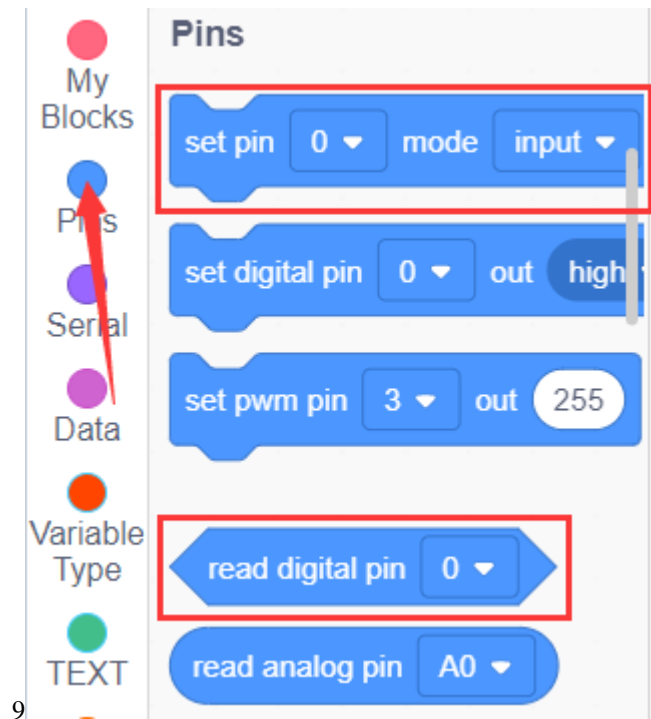
IR Remote

IR remote init PIN# 12

Received data

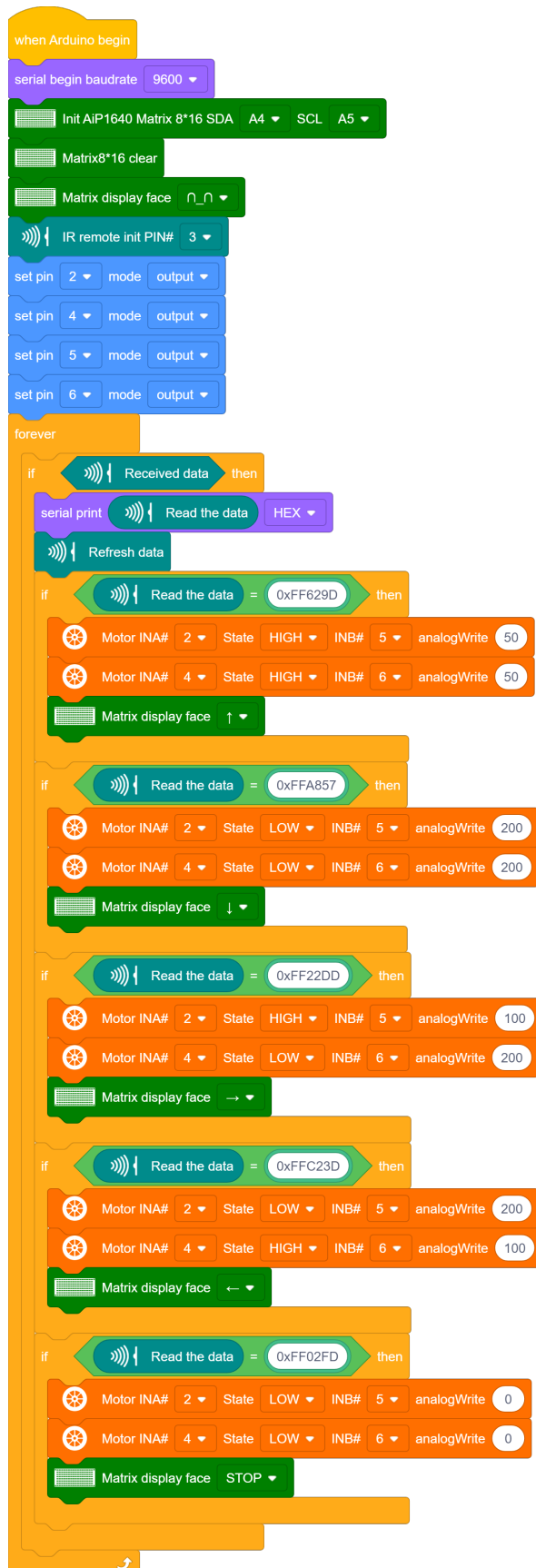
Read the data

Refresh data



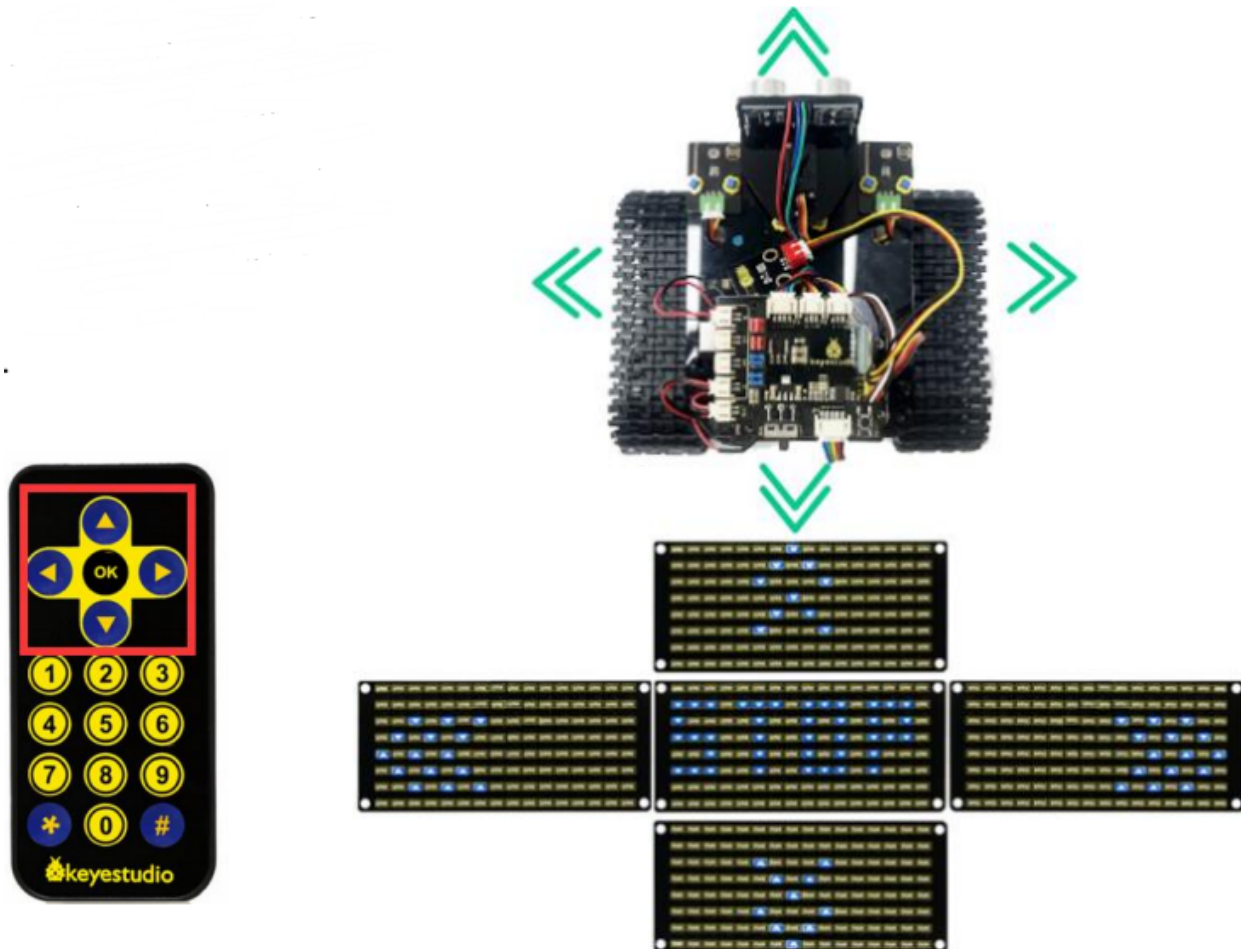
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Results:

After upload the test code successfully and power up, the smart car can be controlled to move by IR remote control and the 8*16 shows the corresponding patterns of its movements.



7.3.16 Project 16: Bluetooth Remote Control

(1)Description:

In the last several decades, Bluetooth has become the most popular wireless communication module for it is easy to use and has found wide applications in most devices powered by batteries.

In order to adjust with the time and reality and need the needs of customers, Bluetooth has been upgraded several times. In recent years, it embraces lots of transformations in terms of data transfer rate, power consumption of wearable devices and IoT devices, and security systems and others. Here, we plan to learn about DX-BT24 with Arduino board.

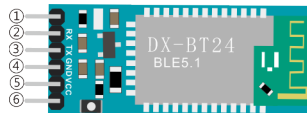
(2)Parameter:

- Bluetooth protocol: Bluetooth Specification V5.1 BLE
- Working distance: In an open environment, achieve 40m ultra-long distance communication Operating frequency: 2.4GHz ISM band
- Communication interface: UART Bluetooth certification: in line with FCC CE ROHS REACH certification standards
- Serial port parameters: 9600, 8 data bits, 1 stop bit, invalid bit, no flow control
- Power: 5V DC
- Operating temperature: -10 to +65 degrees Celsius

(3)Application:

The DX-BT24 module also supports the BT5.1 BLE protocol, which can be directly connected to iOS devices with BLE Bluetooth function, and supports resident running of background programs. Mainly used in the field of short-distance wireless data transmission. Avoid cumbersome cable connections and can directly replace serial cables. Successful application areas of BT24 modules:

- * Bluetooth wireless data transmission;
- * Mobile phone, computer peripheral equipment;
- * Handheld POS equipment;
- * Wireless data transmission of medical equipment;
- * Smart home control;
- * Bluetooth printer;
- * Bluetooth remote control toys;
- * Shared bicycles;

(4)Pins description:

STATEstate pin

RXreception pin

TXsending pin

GNDgrounded

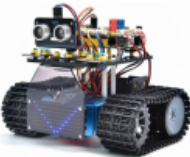




VCCpower pin

ENenable pin

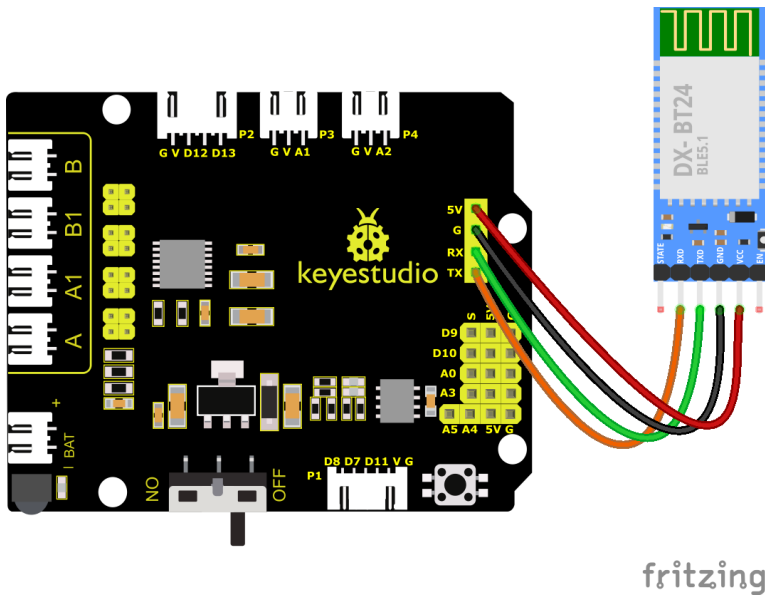
Connect Bluetooth to the development board

Uno	BT24
TX	RX
RX	TX
VCC	5V
GND	GND

(5)Components Required:

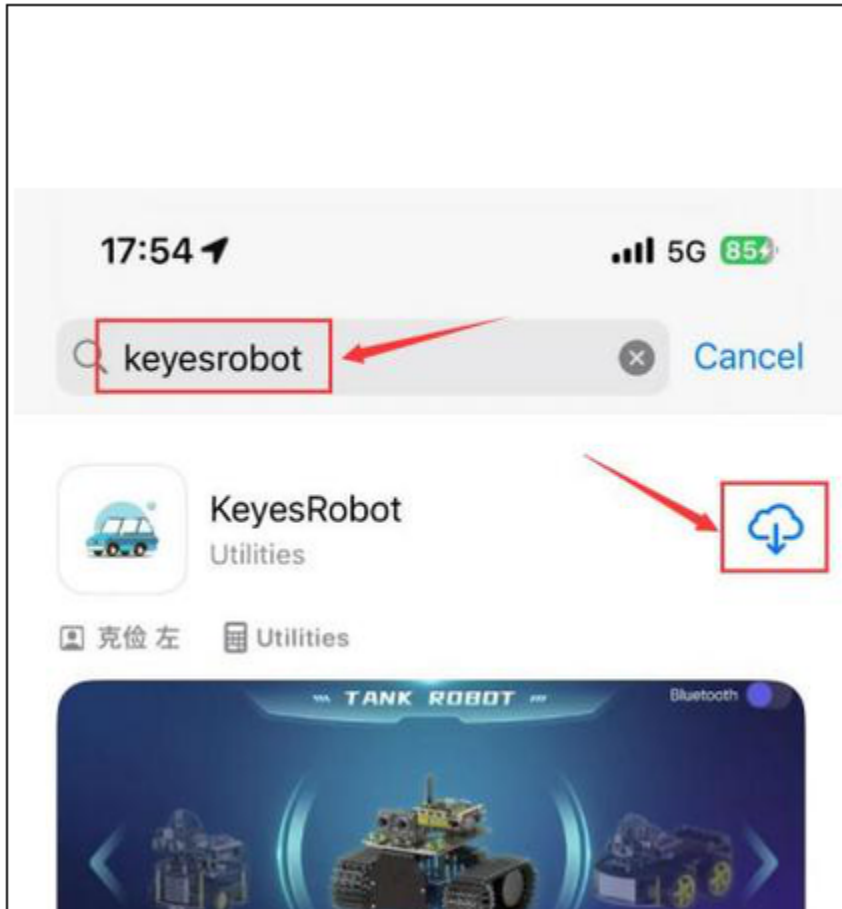
Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

(6)Connection Diagram:

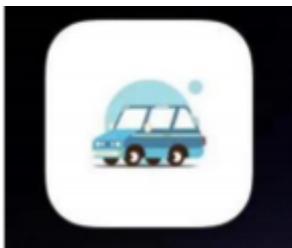


(7)Download APP:**For IOS system**

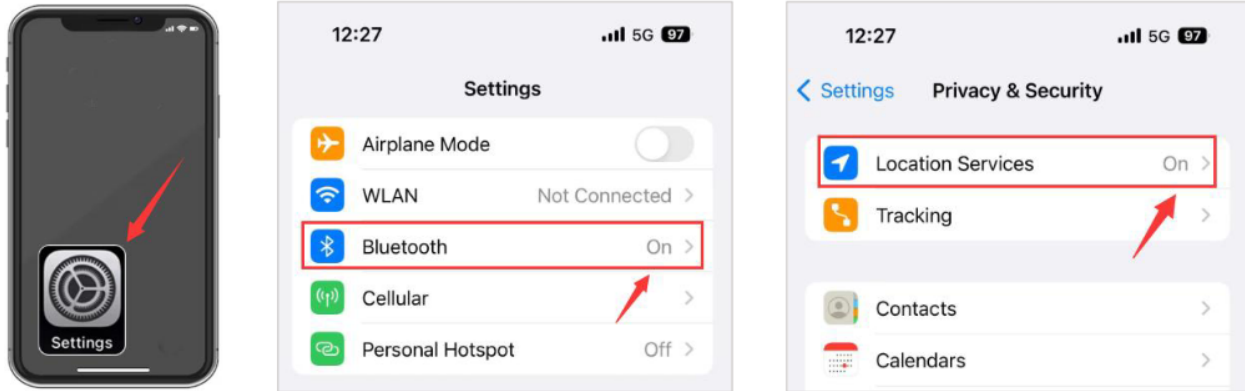
1. Open App Store.
2. Search KeyesRobot in the Apple Store and click download.



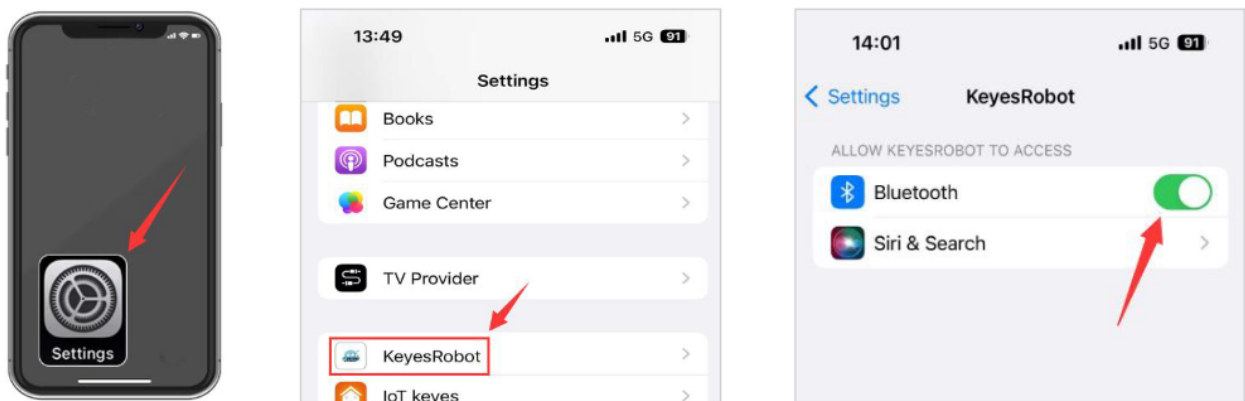
3. After the app is installed, you will see the following icon on your phone desktop.

**How to connect iOS Phone to Bluetooth module:**

1. Turn on the Bluetooth and location services on phone through settings.



2. Allow KeyesRobot APP to access Bluetooth through settings.



3. Click to open KeyesRobot App.



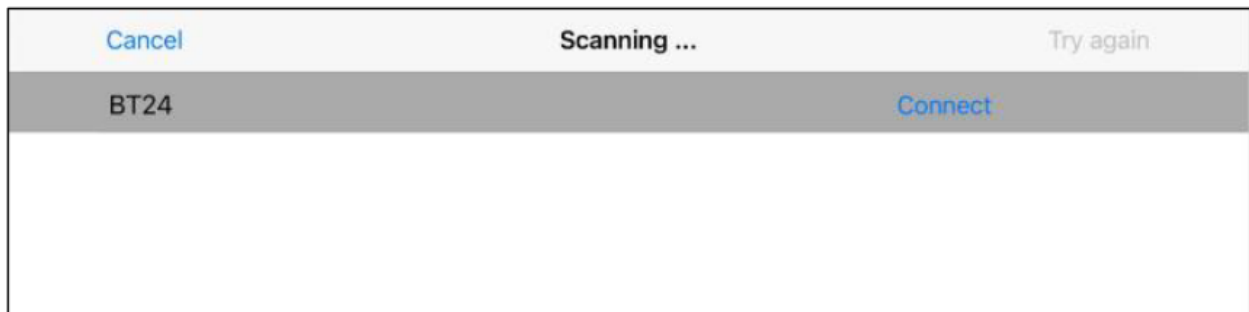
4. KeyesRobot App is a universal APP, which is applied to multiple keystudio robots. If the interface does not display “TANK ROBOT”, you can click the left and right buttons to find “TANK ROBOT”.



5. Click the Bluetooth button in the upper right corner to scan the bluetooth



6. You will see a Bluetooth named **BT24**, click the Connect button.



7. If the onboard LED on the Bluetooth module stops flashing and stays on, it means your phone is successfully connected to the Bluetooth module.



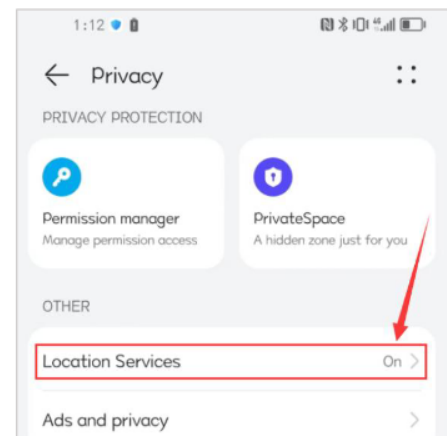
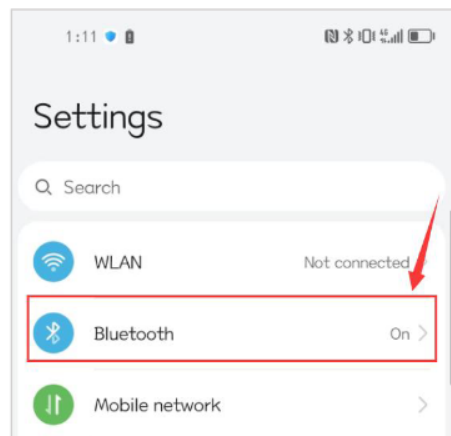
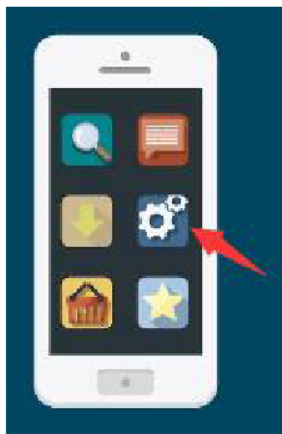
For Android System

1. Search **KeyesRobot** in Google Play, or open the following link to download and install the app.

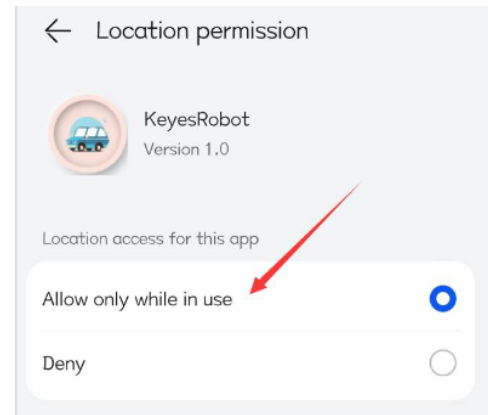
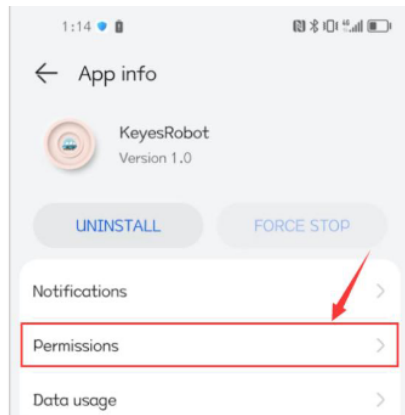
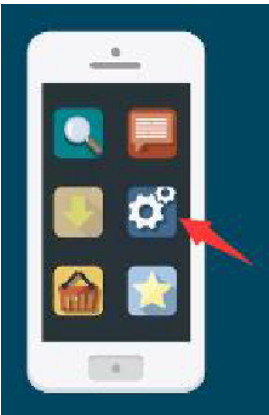
<https://play.google.com/store/apps/details?id=com.keyestudio.keyestudio>



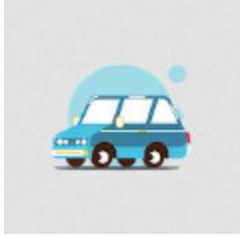
2. Turn on the Bluetooth and the location services of the mobile phone



3. Find the KeyesRobot Bluetooth app from settings, click on the permission options of the app, and enable Location and nearby device permissions.(Note: Some mobile phones do not have nearby device permissions function.)



4. Click to open KeyesRobot App.



5. KeyesRobot App is a universal APP, which is applied to multiple keyestudio robots. If the interface does not display “TANK ROBOT”, you can click the left and right buttons to find “TANK ROBOT”.

6. Click the Bluetooth button  in the upper right corner to scan the bluetooth



7. You will see a Bluetooth named **BT24**, click the Connect button.



8. When your phone is successfully connected to the Bluetooth module, the onboard LED on the Bluetooth module

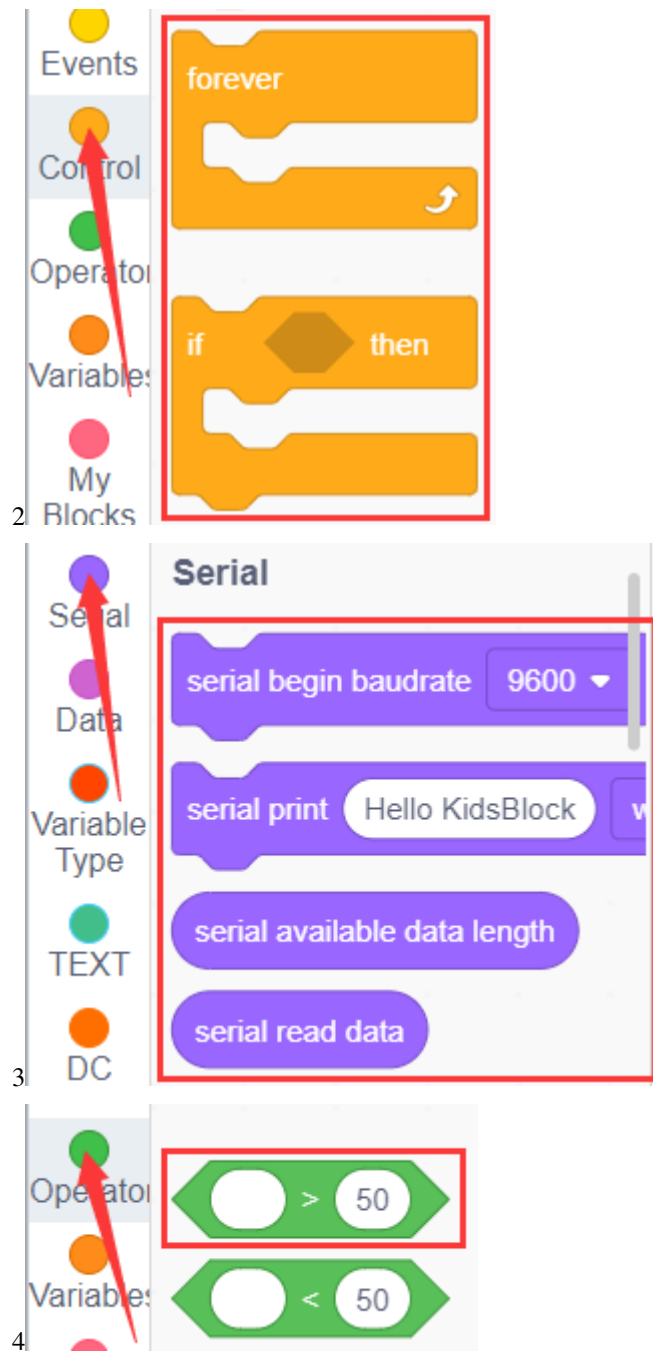
will stop flashing and stay on.



(8)BT Test Code:

You can also drag blocks to edit your code, as shown below

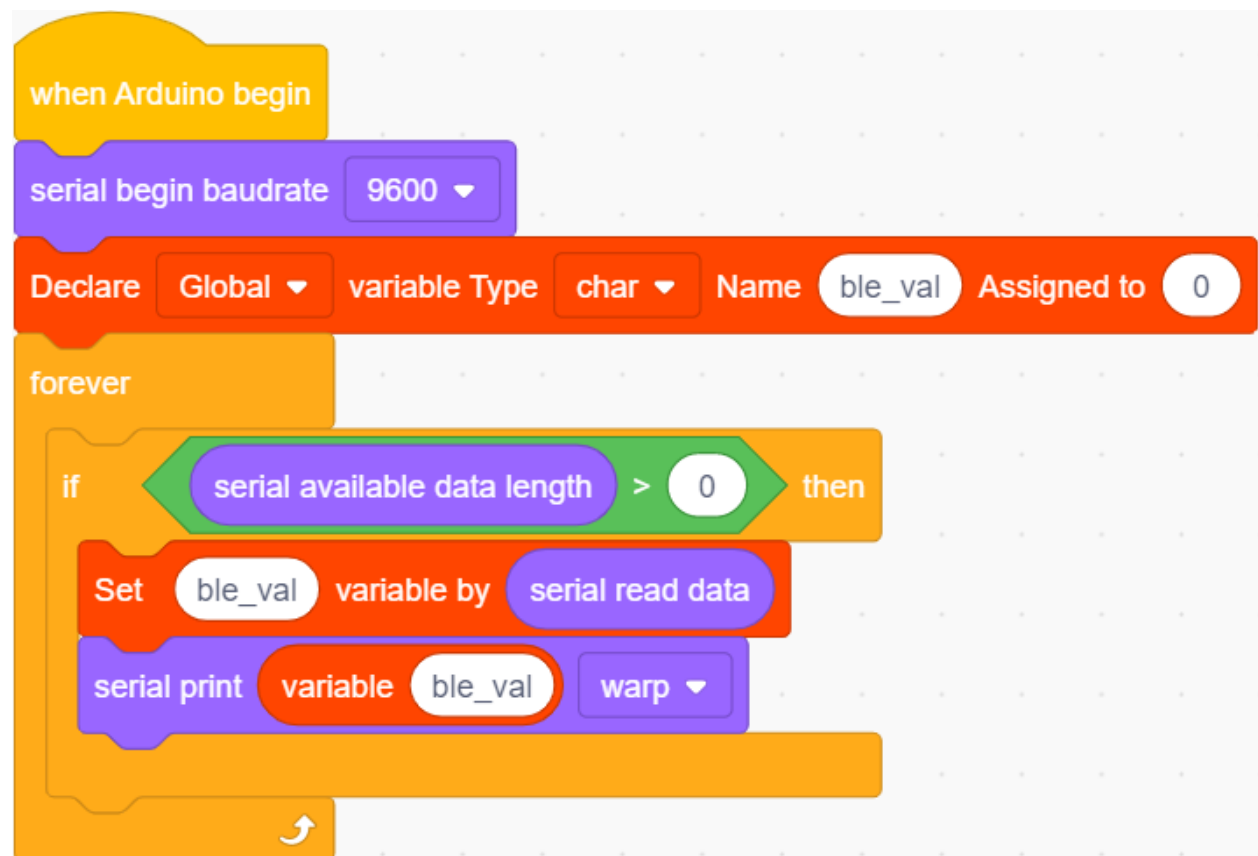






Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)




Upload the code to the development board, then plug in the Bluetooth module, and then connected the mobile phone to the Bluetooth module.

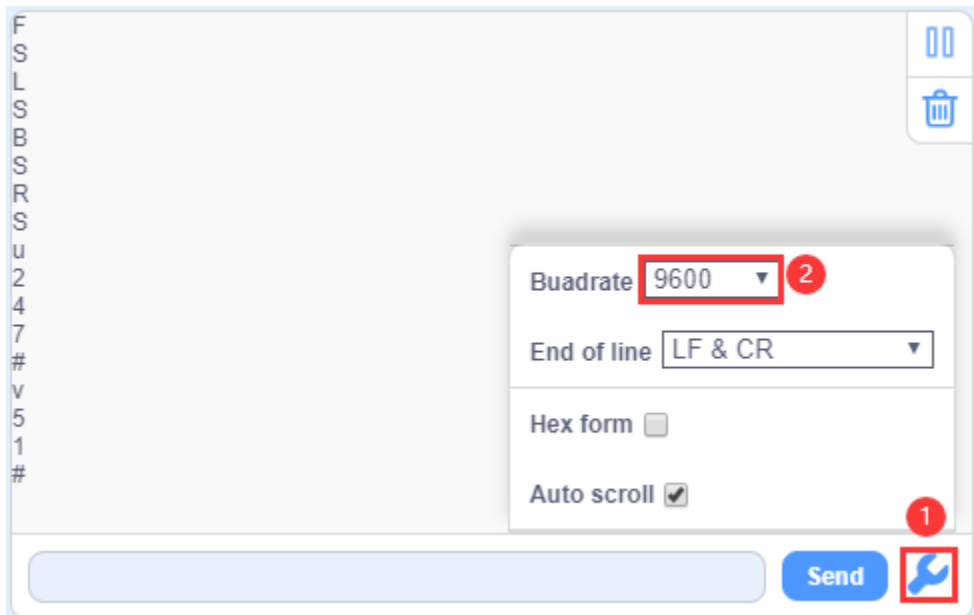
After the mobile phone is successfully connected to the Bluetooth module, click to open the Bluetooth APP and click the Select button on the homepage.



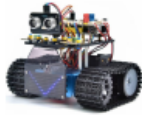






The main interface of the Bluetooth app is shown in the figure below.



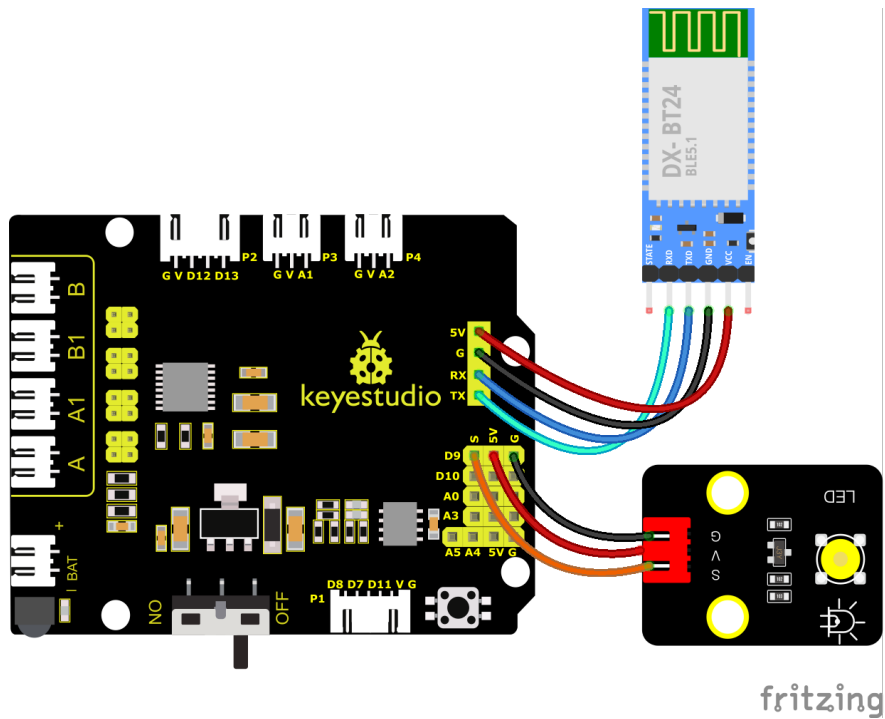
Click  and set the baud rate to 9600. Click the icon on the APP interface and the serial monitor will display command sent by button.



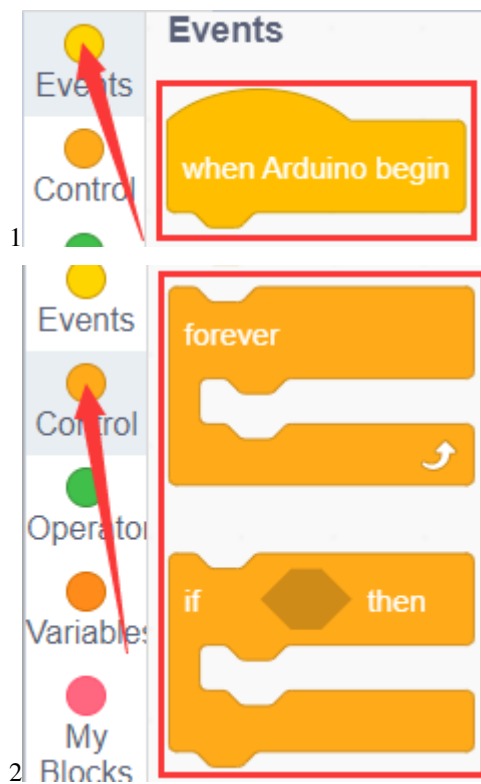
(9)Extension Practice:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1
			
Yellow LED Module*1	3P-3P XH2.54 to 2.54 Dupont Wire	18650 Battery*2	
			

In the above project, Bluetooth receives the signal sent by the mobile phone and displays it on the serial port of the development board. Here we use the command sent by the mobile phone to turn on or off an LED. Looking at the wiring diagram, an LED is connected to the D9 pin,



You can also drag blocks to edit your code, as shown below



3

Serial

serial begin baudrate 9600

serial print Hello KidsBlock

serial available data length

serial read data

4

Operator

> 50

< 50

= 50

My

5

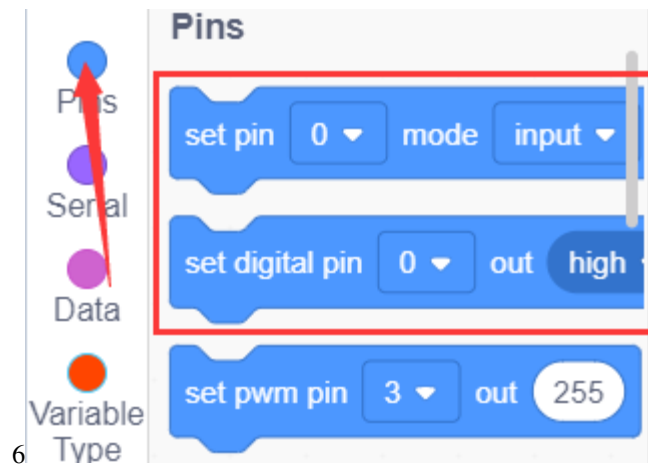
Variable Type

Declare Global variable Type int Name item Assigned to 0

variable item

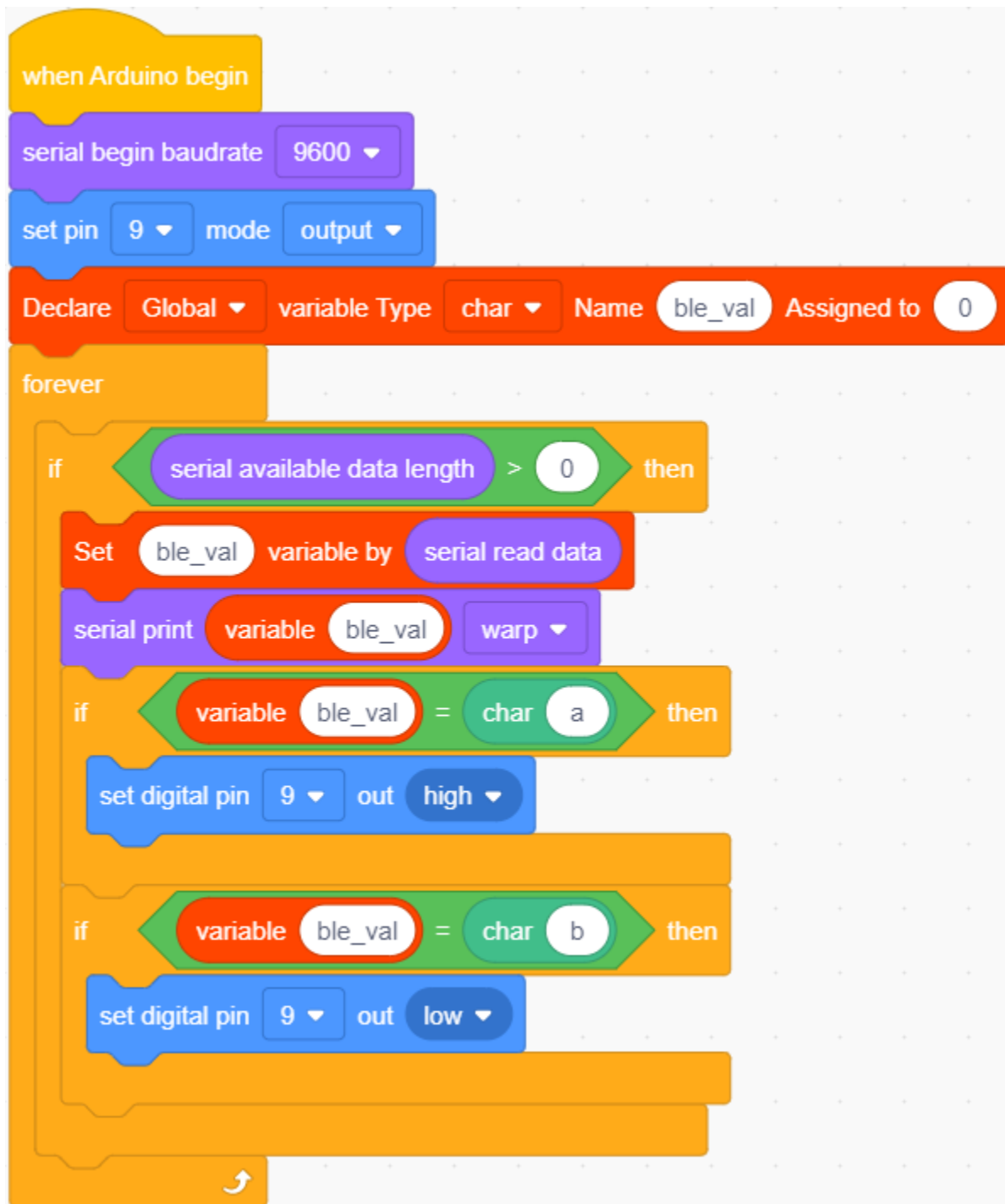
Set item variable by 0


The image shows a screenshot of the Kidsblock IDE interface. On the left sidebar, there are several category icons: Serial (purple), Data (pink), Variable Type (orange), TEXT (green), and DC (orange). Red arrows point to these categories. The main workspace is divided into three sections. The top section, titled 'Serial', contains four purple blocks: 'serial begin baudrate 9600', 'serial print Hello KidsBlock', 'serial available data length', and 'serial read data'. The middle section, titled 'Operator', contains three green comparison blocks: '> 50', '< 50', and '= 50'. The bottom section, titled 'Variable Type', contains three orange blocks: 'Declare Global variable Type int Name item Assigned to 0', 'variable item', and 'Set item variable by 0'. Red boxes highlight the Serial blocks, the Operator comparison blocks, and the Variable Type blocks. The numbers 3, 4, and 5 are placed to the left of the Serial, Operator, and Variable Type sections respectively.

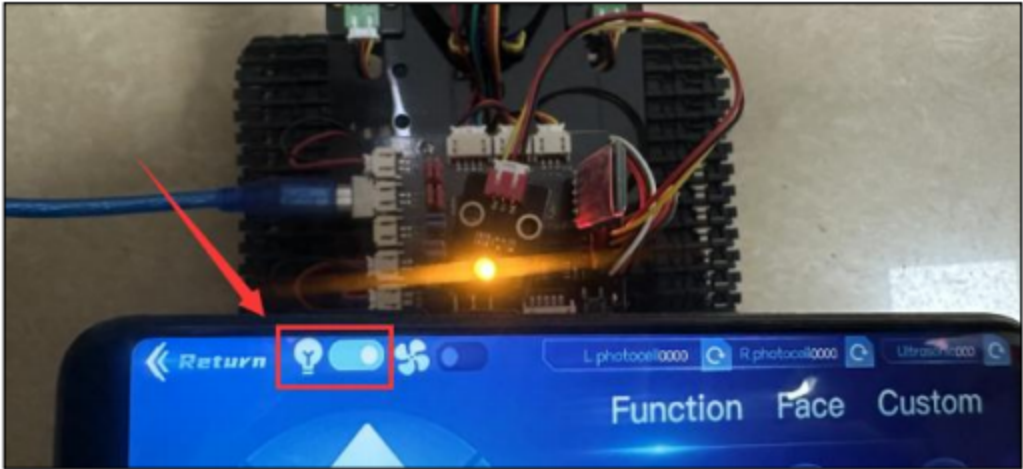


Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



After the code above is successfully uploaded. Click  to control the LED.



After you finish the BT project, remove it.

7.3.17 Project 17: Bluetooth Control Tank

(1)Description:

We have learned the basic knowledge of Bluetooth in the previous project . In this lesson, we will use Bluetooth to control the smart car. Since it involves Bluetooth, a sending end and a receiving end are needed. In the project, we use the mobile phone as the sender (master), and the smart car connected with the HM-10 Bluetooth module (slave) as the receiver.

We have learned earlier that sending a bit can control LEDs. And the principle of controlling this robot car is the same.

In order to better control the intelligent tank robot, we specially made an APP. In this lesson, we will read all the key value on this APP through code, and then introduce the exclusive APP of our tank robot.

(2)Key Function on the APP:

The following table illustrates the functions of corresponding keys:

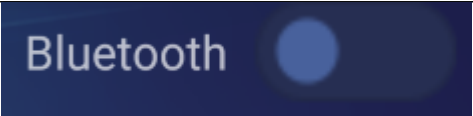


Keys		Fun
		
		Pair
		select to c

Table 1 – continued








Keys			Fun
			To
			To
		Send “F”when pressed and “S”when released	The
		Send “L”when pressed and “S”when released	The
		Send “R”when pressed and “S”when released	The
		Send “B”when pressed and “S”when released	The
		Send “u”+digit+“#”when dragged	Dra








Table 1 – continued

Keys			
		Send “v”+digit+“#”when dragged	Dr
		Select to enter Function page	
		Send “G”when pressed and “S”when pressed again	Ent
		Send “h”when pressed and “S”when pressed again	Ent
		Send “e”when pressed and “S”when pressed again	Ent
		Send “f”when pressed and “S”when pressed again	Ent
		Send “i”when pressed and “S”when pressed again	Ent
		Send “j”when pressed and “S”when pressed again	Ent

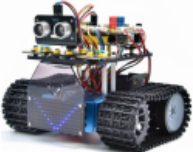

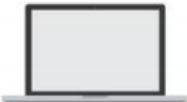


Table 1 – continued

Keys			Fun
		Select to enter facial expression display mode	
		Send “k”when pressed and “z”when pressed again	Sho
		Send “l”when pressed and “z”when pressed again	Sho
		Send “m”when pressed and “z”when pressed again	Sho
		Send “n”when pressed and “z”when pressed again	Sho

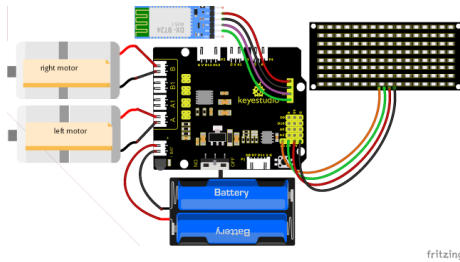
Table 1 – continued

Keys			
		Send “o”when pressed and “z”when pressed again	Fun
		Send “p”when pressed and “z”when pressed again	Sho
			Cho
		Click to send “w”	Clic
		Click to send“y”	Clic
		Click to send“x”	Clic
		Click to send“c” Click again to send“d”	Pre

(3)Components Needed:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

(4)Connection Diagram:

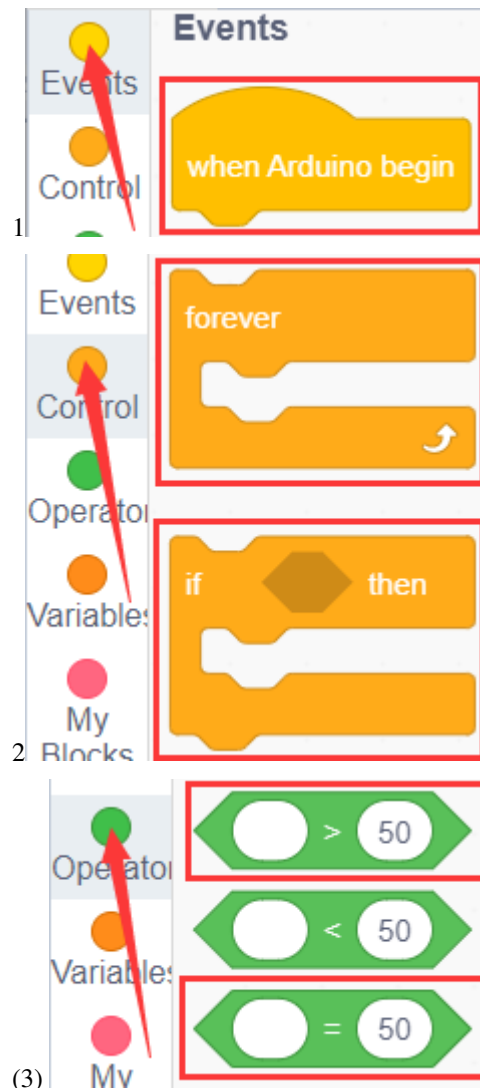


Note:

GND, VCC, SDA and SCL of the 8x16 LED panel are connected to GGND), V5V), A4 and A5 of the expansion board. STATE and BRK don't need to be interfaced. The BT module is inserted into the expansion board.

(5)Test Code:

You can drag blocks to edit your code



4

Serial

serial begin baudrate 9600

serial print Hello KidsBlock

5

DC Motor

Motor INA# 2 State HIGH INB# 6 State HIGH

Motor INA# 2 State HIGH INB# 6 analogWrite 255

6

TEXT

char a

string hello

7

Matrix 8*16

Init AiP1640 Matrix 8*16 SDA A4 SCL A5

Matrix8*16 display face 0_0

Matrix8*16 clear

Serial

DC Motor

TEXT

Matrix 8*16

8

The screenshot displays the Kidsblock IDE interface. On the left sidebar, the 'Variable Type' block is highlighted with a red circle and a red arrow. The main workspace is divided into two sections: 'Variable Type' and 'Pins'. The 'Variable Type' section contains three orange blocks: a 'Declare' block with 'Global' selected, 'variable Type' set to 'int', 'Name' set to 'item', and 'Assigned to' set to '0'; a 'variable' block with 'item' selected; and a 'Set' block with 'item' selected and 'variable by' set to '0'. The 'Pins' section contains four blue blocks: a 'set pin' block with '0' selected and 'mode' set to 'input'; a 'set digital pin' block with '0' selected and 'out' set to 'high'; a 'set pwm pin' block with '3' selected and 'out' set to '255'; and a 'read digital pin' block with '0' selected. A red box highlights the 'set pin' and 'read digital pin' blocks. A red arrow points to the 'Pins' block in the sidebar.

Variable Type

8

Variable Type

TEXT

DC Motor

9

My Blocks

Pins

Serial

Data

Variable Type

TEXT

Variable Type

TEXT

set pin 0 mode input

set digital pin 0 out high

set pwm pin 3 out 255

read digital pin 0

read analog pin A0

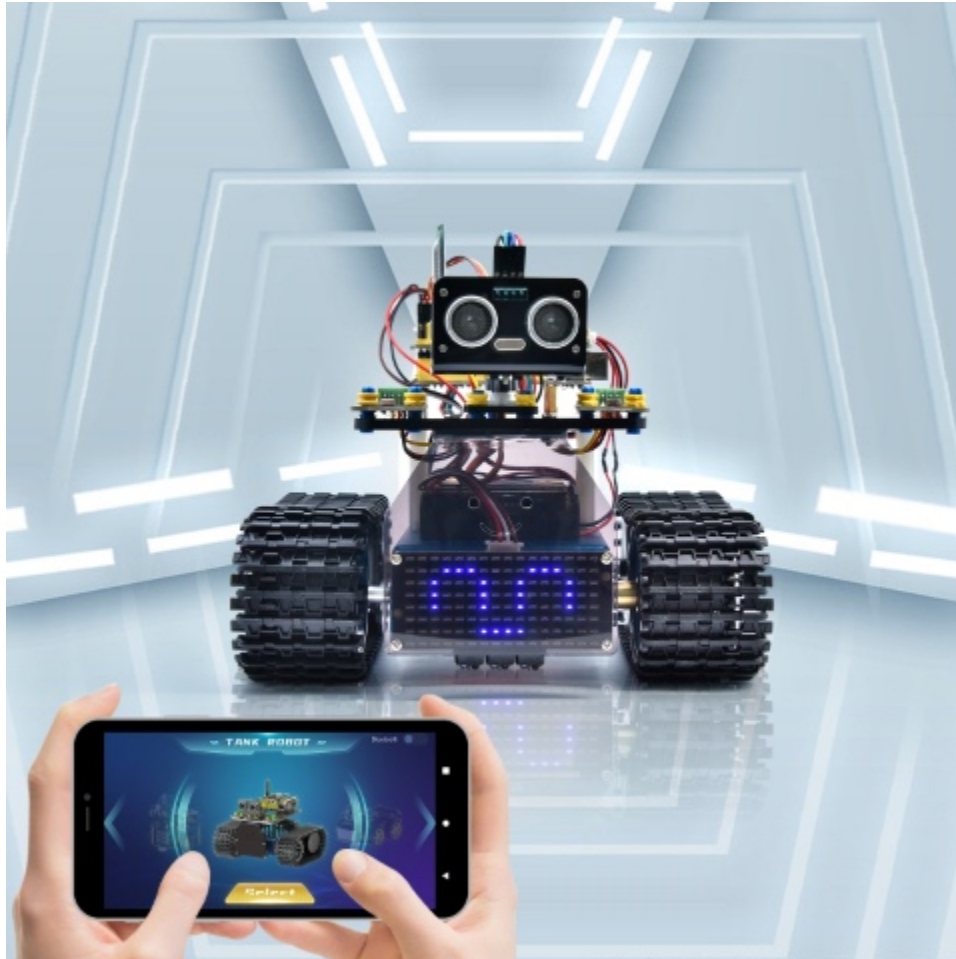
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



(6)Test Result:

After uploading the code, connect the robot to the Bluetooth module and pair the Bluetooth APP. Turn on the power-switch of the motor drive shield. Place the robot on the floor, you can use these buttons of the Bluetooth app to control the robot.



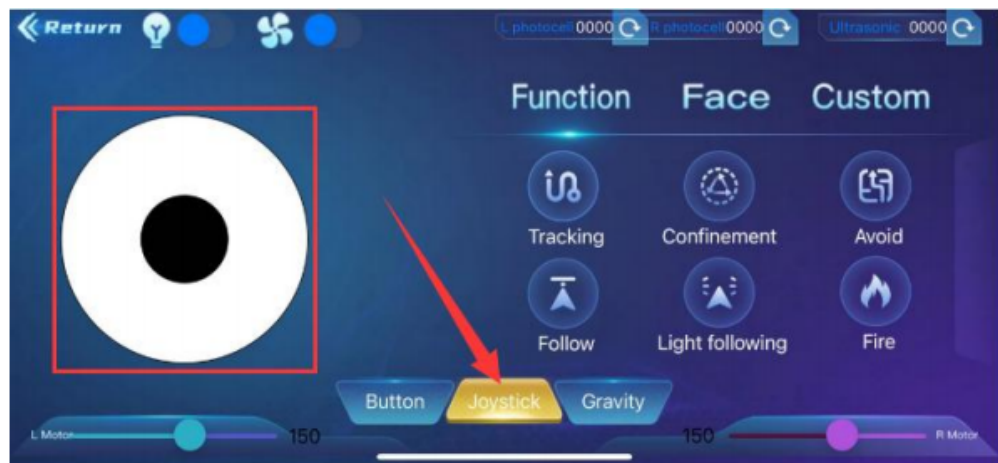
1. The up, down, left and right arrows control the robot to move forward, backward, left and right respectively.



2. Click the joystick button and pull the direction of the black point in the white circle to control the movement

direction of the robot.

2



3. Click the Gravity button and tilt the phone in the forward, backward, left, and right directions, and the robot will move in the direction in which the phone is tilted.

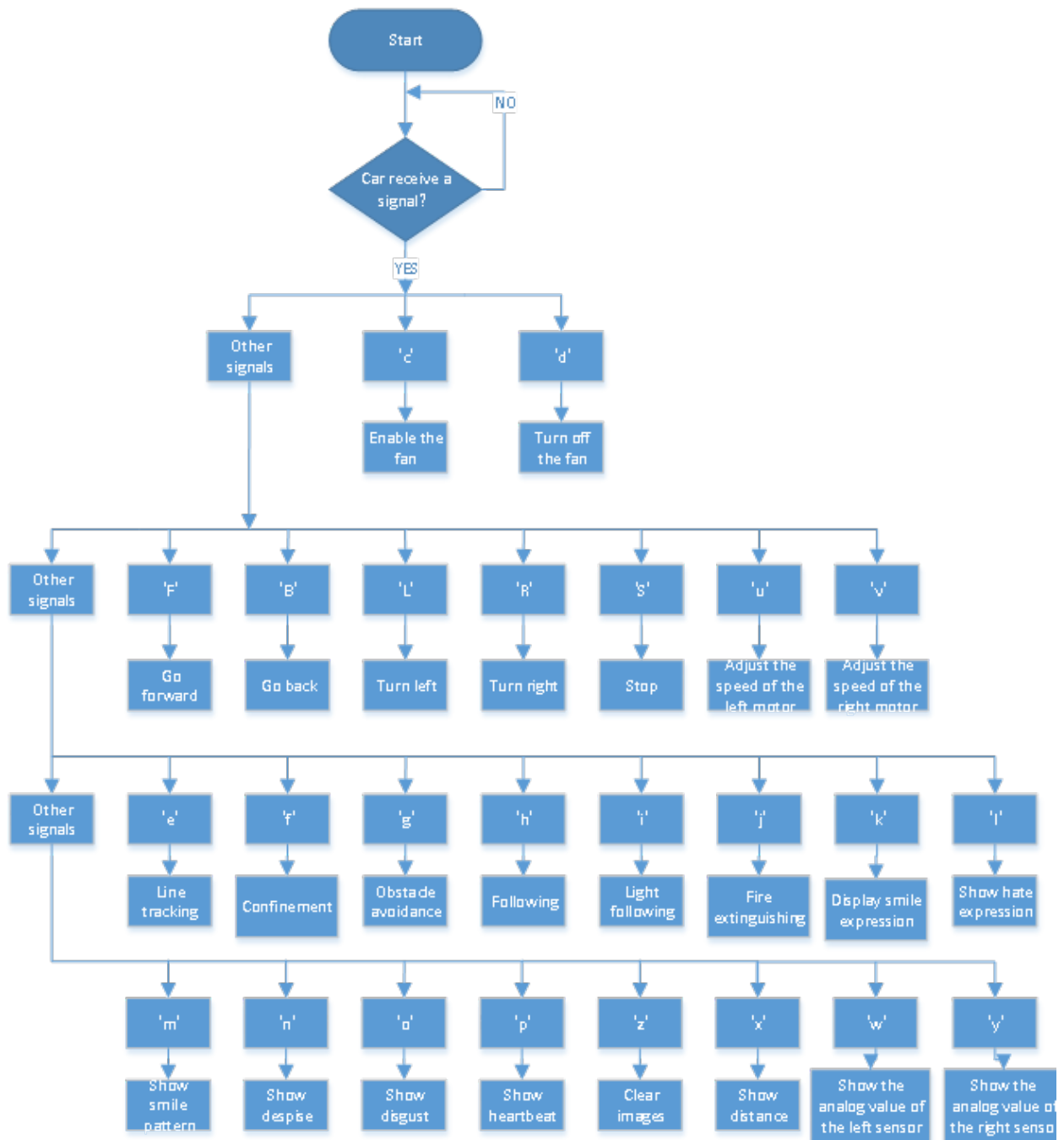
3



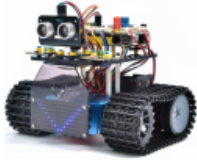




7.3.18 Project 18: Ultrasonic Tank Robot Multiple Functions

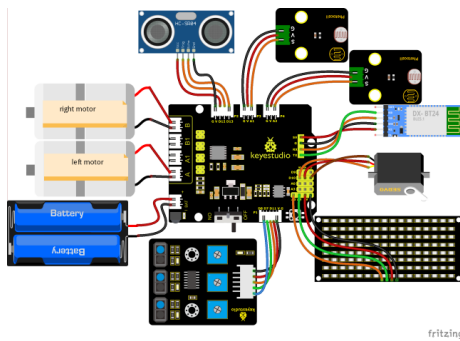
(1)Description:

The smart car has performed a single function in every previous project. Can it display multiply functions at a time ? Positive. In this last big project, we intend to use a complete code to control the smart car to show off all functions mentioned in previous projects. We use the keys on the Bluetooth APP to automatically switch various functions, quite simple and convenient.



(2) Components Needed:

Robot tank*1	USB Cable*1	Computer*1	Bluetooth module*1	18650 Battery*2
				

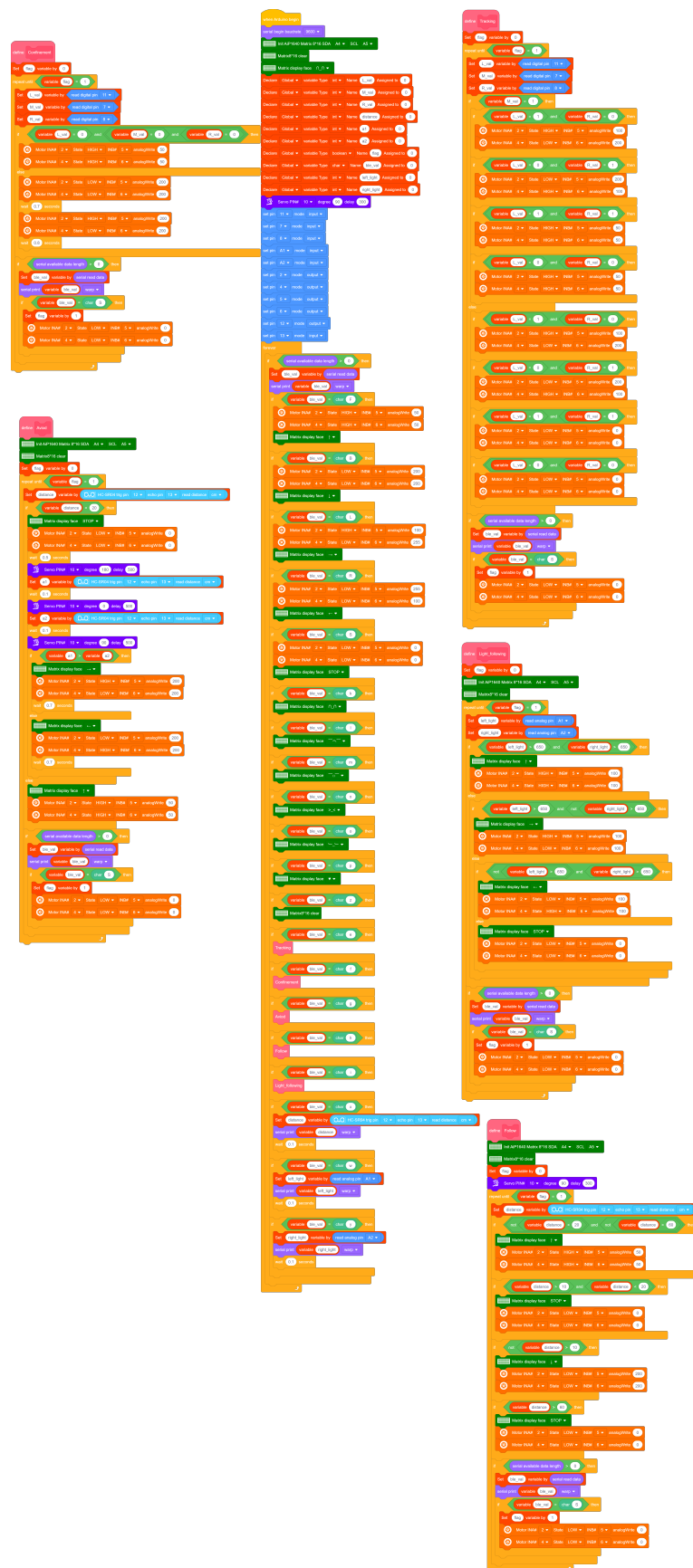
(3) Connection Diagram:

1. GND, VCC, SDA and SCL of the 8x16 board are connected to GGND), +VCC), A4 and A5 of the expansion board.
2. VCC, Trig, Echo and Gnd of the ultrasonic sensor are connected to 5V(V), 12(S), 13(S) and Gnd(G)
3. The brown wire, red wire and orange wire of the servo are connected to Gnd(G), 5v(V) and D10.
4. RXD, TXD, GND and VCC of the BT module are connected to TX, RX, GGND) and 5VVCC. STATE and BRK don't need to be interfaced.
5. The pin "G", "V" and S of the left photoresistor module are connected to G (GND), V (VCC), A1 respectively; The right photoresistor module is connected to the G (GND), V (VCC), and A2 respectively.
6. The distal port of the line tracking sensor is 11, 7 and 8

(4) Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

Note: you can't speed up the car via App.

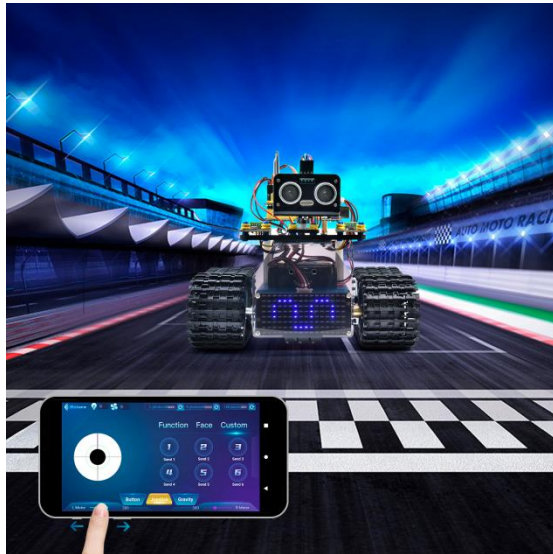


(5)Test Result:

Before uploading the program code, the Bluetooth module needs to be removed, otherwise the code upload will fail.

After uploading the code successfully, open the positioning, and then connect the Bluetooth module.

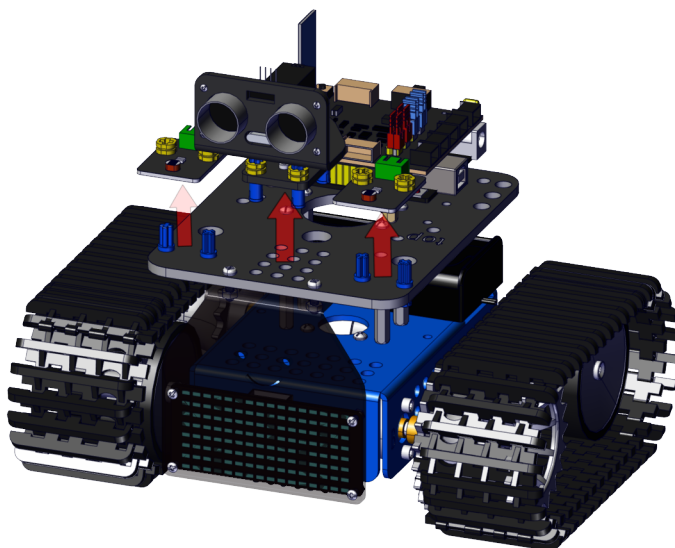
After uploading the code successfully, plug in the Bluetooth module, after power-on, after the mobile APP is connected to the Bluetooth successfully, we can use the mobile APP to control the tank robot.

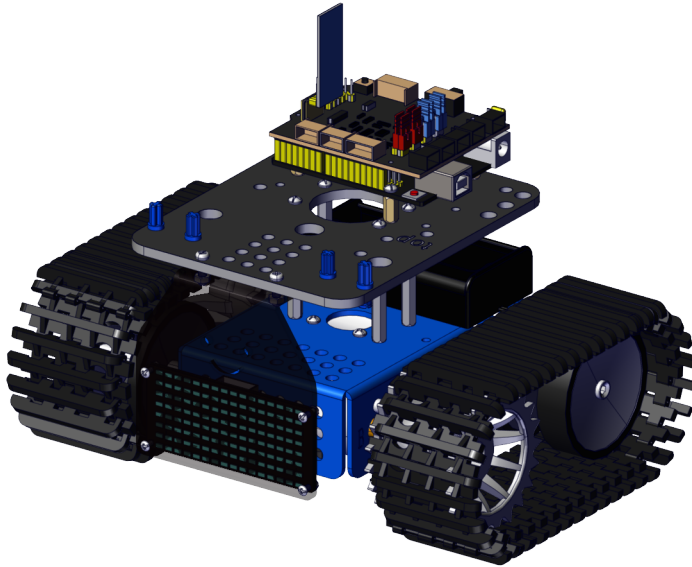


7.4 4. Experiment Extension

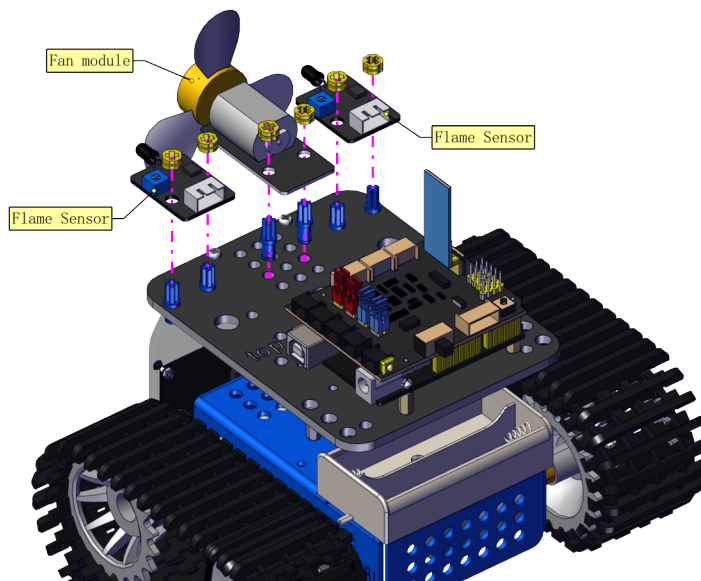
7.4.1 Assemble Fire Extinguishing Robot

Remove the ultrasonic sensor and two photoresistors

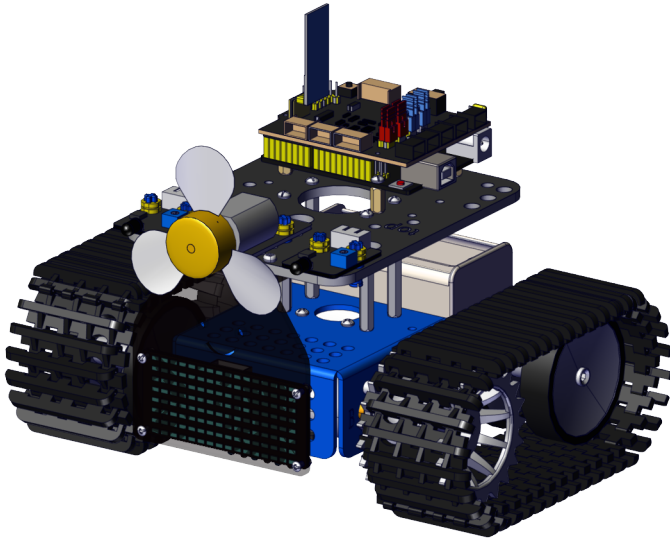




Put on a fan module and two flame sensors

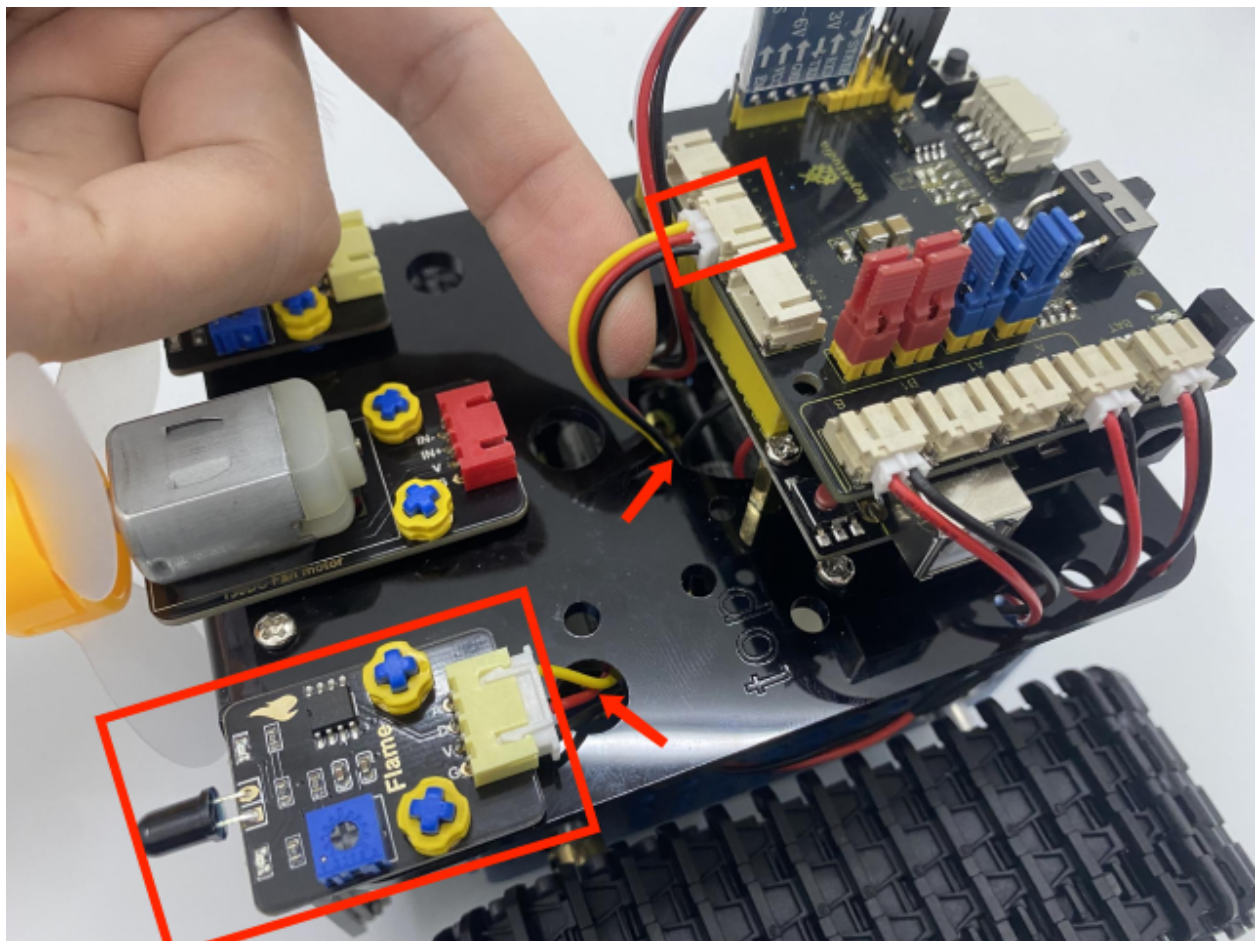


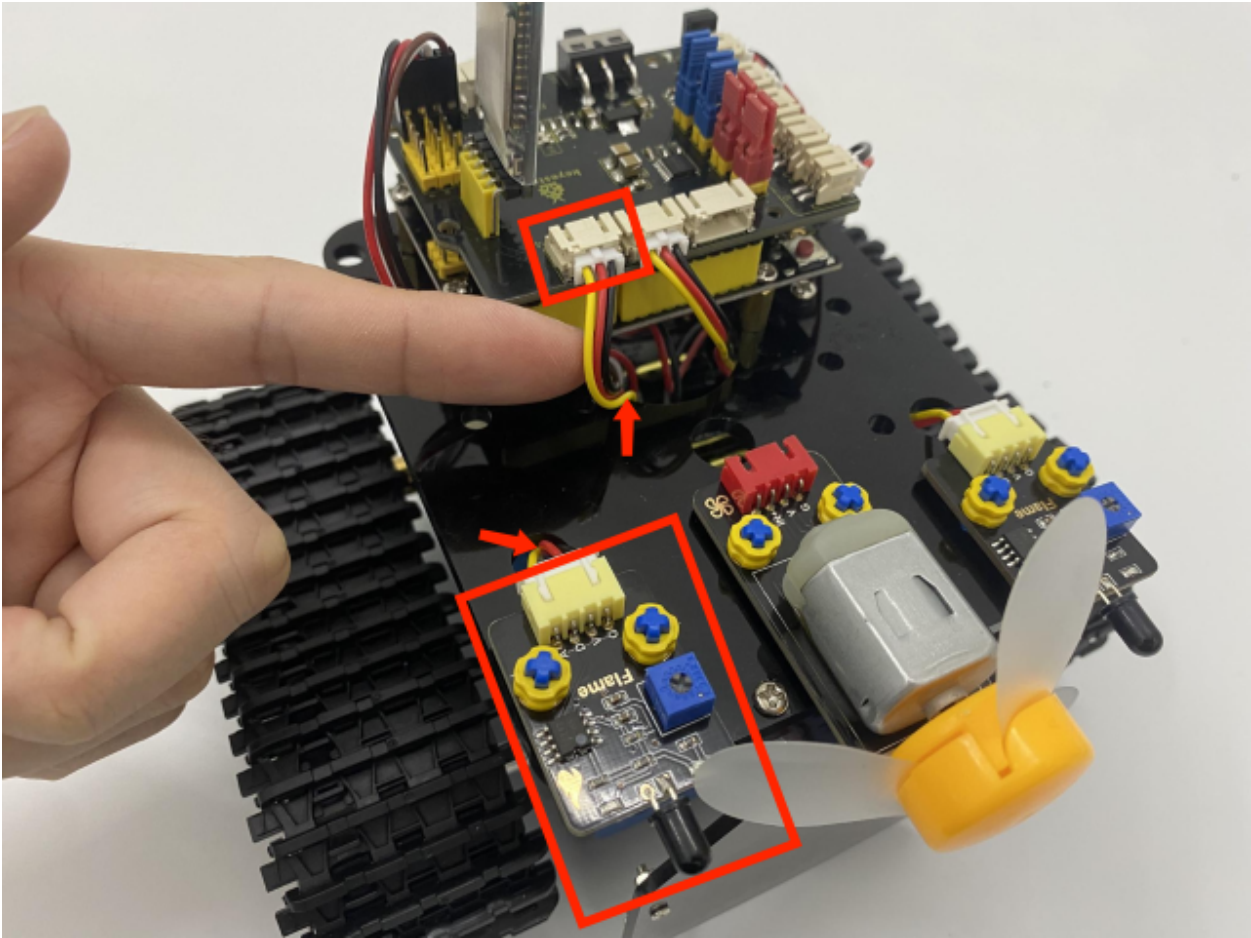
You can make the fan module install further if the fan module and flame sensors interfere



Wire up

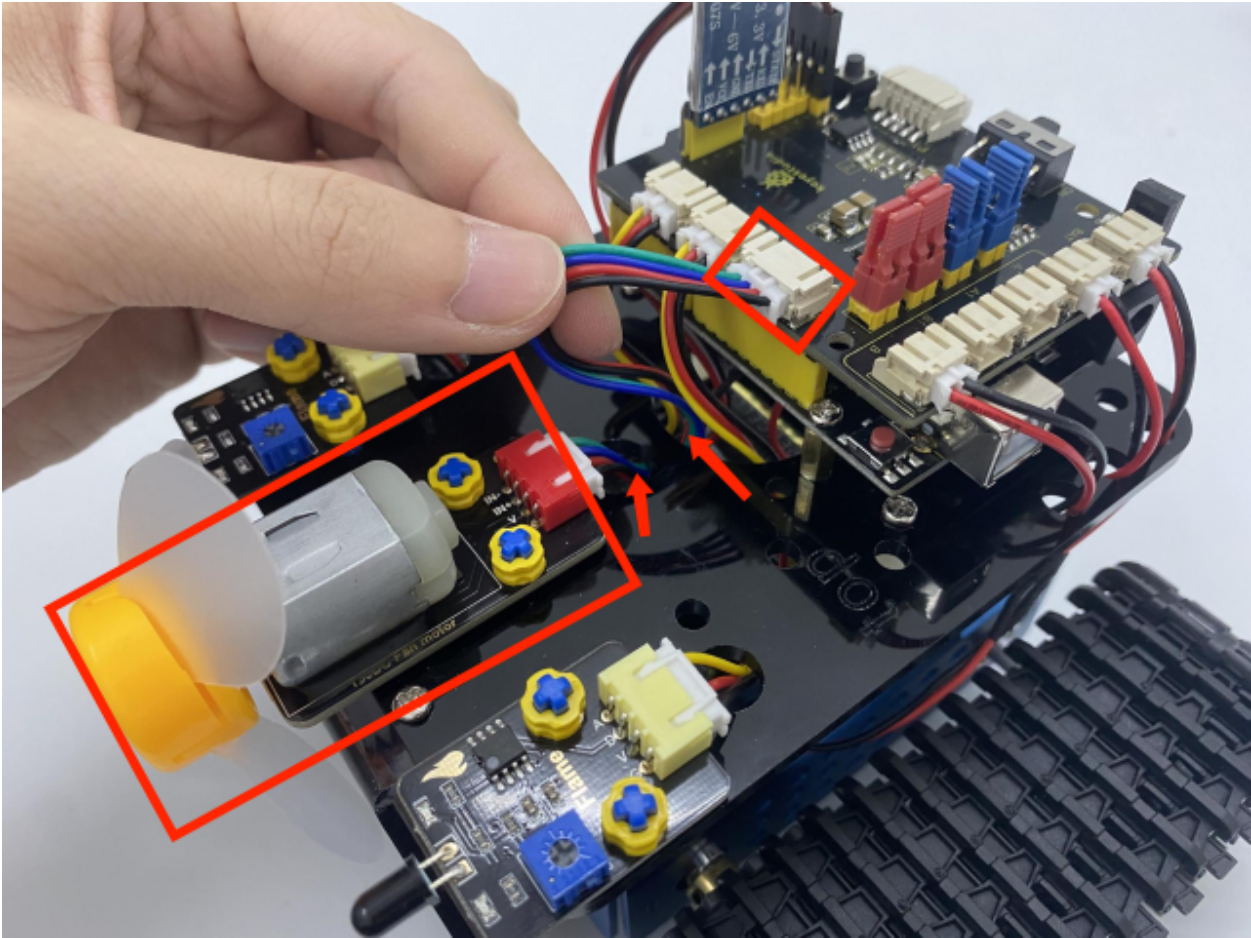
Wire up the two flame sensors





Left	Keyestudio 8833 Board	Right	Keyestudio 8833 Board
G	G	G	G
V	V	V	V
A	A1	A	A2

Wire up the fan module



DC130 Motor	Keyestudio 8833 Board
G	G
V	V
IN+	D12
IN-	D13

We adopt a model 18650 lithium battery with a pointed positive pole, whose power and capacity are not required.



7.4.2 Project 19: Flame Sensor



(1)Description:

The flame sensor uses IR receiving tube to detect flames, converts the brightness of the flame into signals with high and low levels, input them into the central processor. The corresponding program processing. In both flames close to and without flames, the voltage value of the analog port is varied.

If there is no flame, the analog port is about 0.3V; when there is a flame, the analog port is 1.0V. The closer the flame is, the more the voltage value is. It can be used to detect the fire source or make a smart robot.

Note the probe of flame sensors only bears the temperature between -25 °C and 85°C.

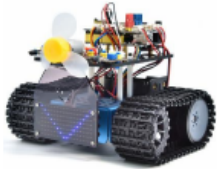

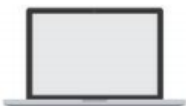


In the process of use, pay attention to keep the flame sensor in certain distance to avoid getting damaged.

(2)Parameters:

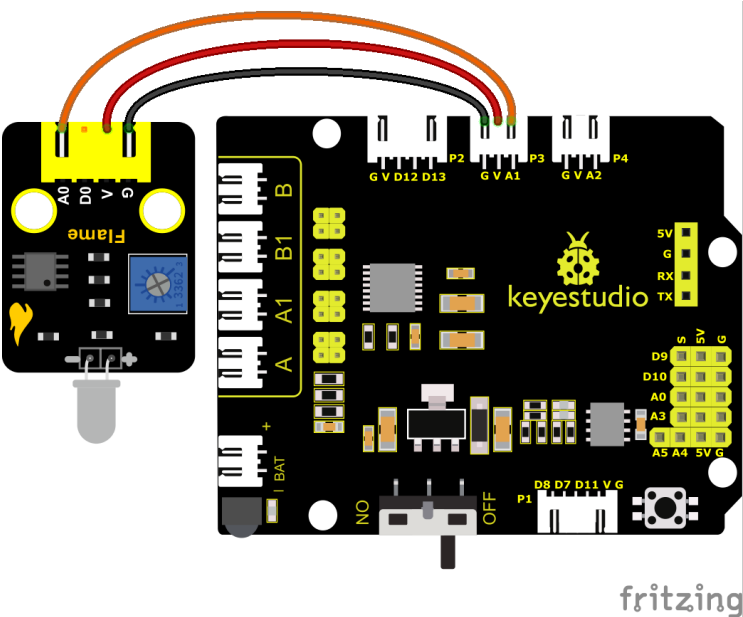


- Working voltage: 3.3V-5V (DC)
- Current: 100mA
- Maximum power: 0.5W
- Work temperature: -10 ° C to +50 degrees Celsius
- Sensor size: 31.6mmx23.7mm
- Interface: 4pin turn 3PIN interface
- Output signal: analog signals A0, A1

(3)Components Required:

Robot tank without BT module	USB Cable*1	Computer*1	18650 Battery*2	Lighter*1
				

(4)Connection Diagram:

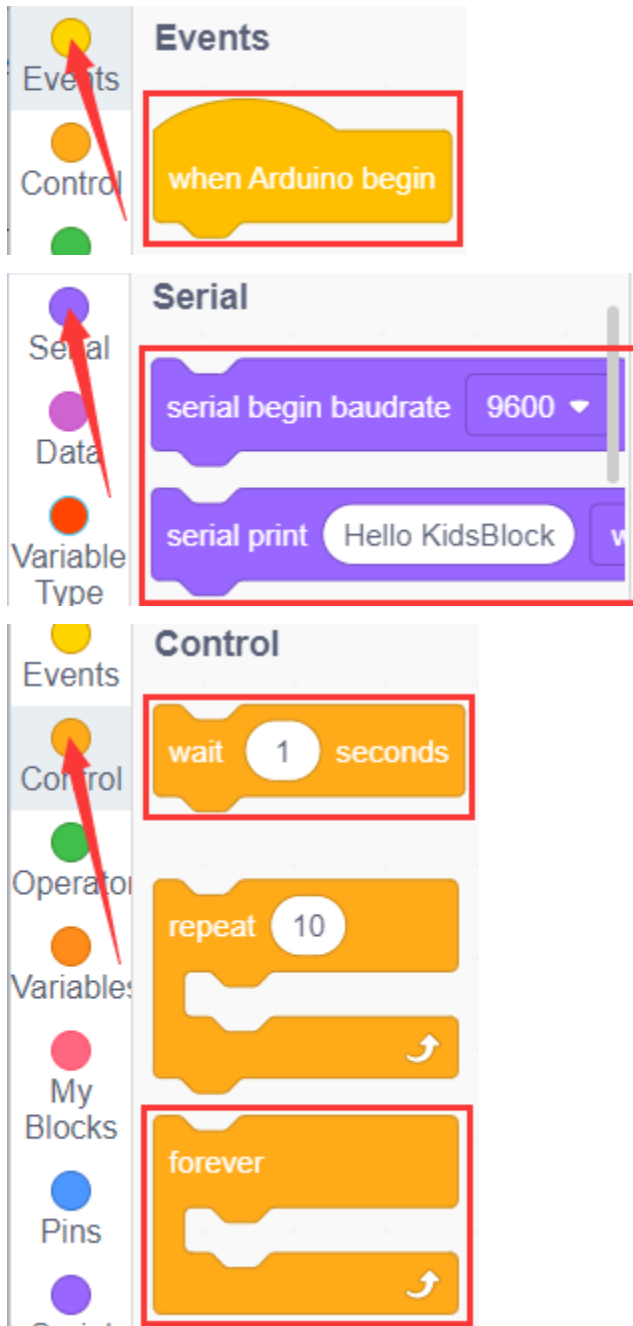


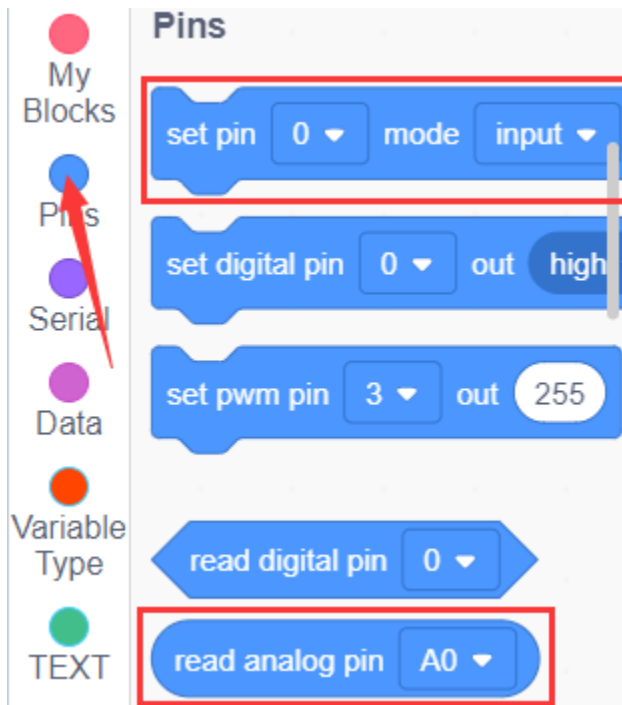
Pin A of two photoresistors are connected to A1 and A2. We connect the flame sensor to A1 and A2. We replace two photoresistors and the ultrasonic sensor with two flame sensors and a fan, an extinguishing car is created.

Note: 1This experiment requires the use of a fire source. Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. 2The flame sensor is not fireproof, please do not burn it directly with flame.

(5)Test Code:

You can also drag blocks to edit your code, as shown below





Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



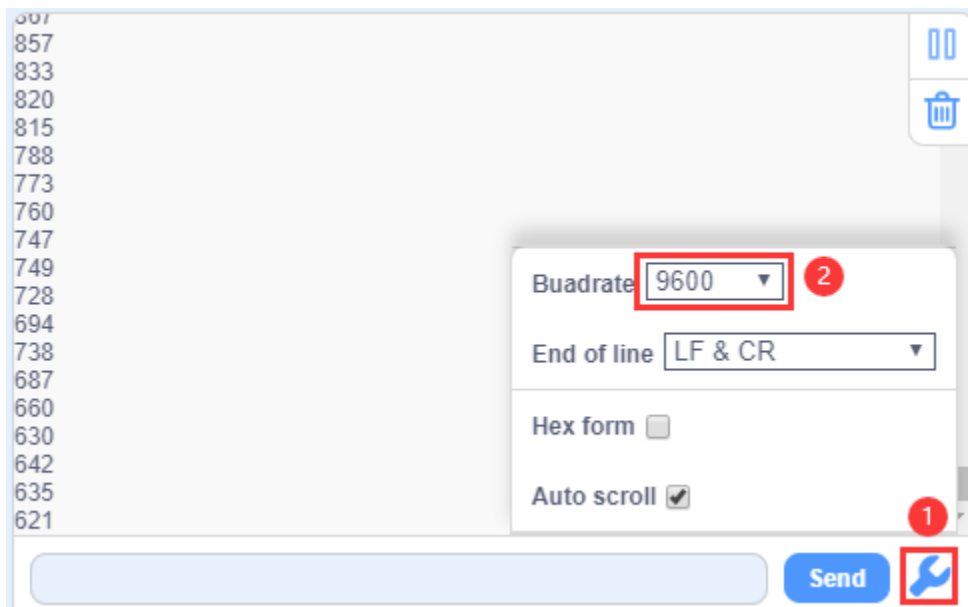
(6)Test Results:

Wire up components, burn the code, open the serial monitor and set the baud rate to 9600.

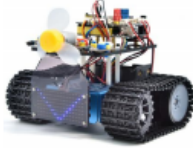






You can view the simulation value of flame sensor.

The closer the flame, the smaller the simulation value.

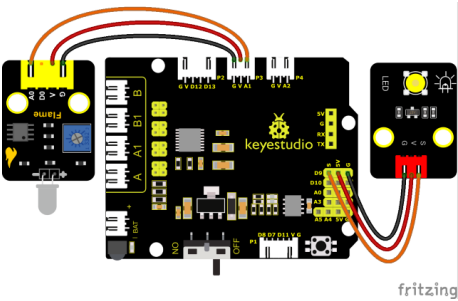
Adjust the potentiometer on the module to maintain the LED at the critical point. When the sensor does not detect flame, the LED will be off, but if the sensor detects flame, the LED will be on.



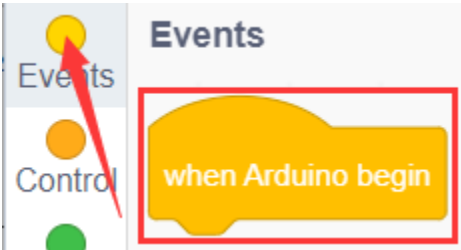
(7)Extension Practice:

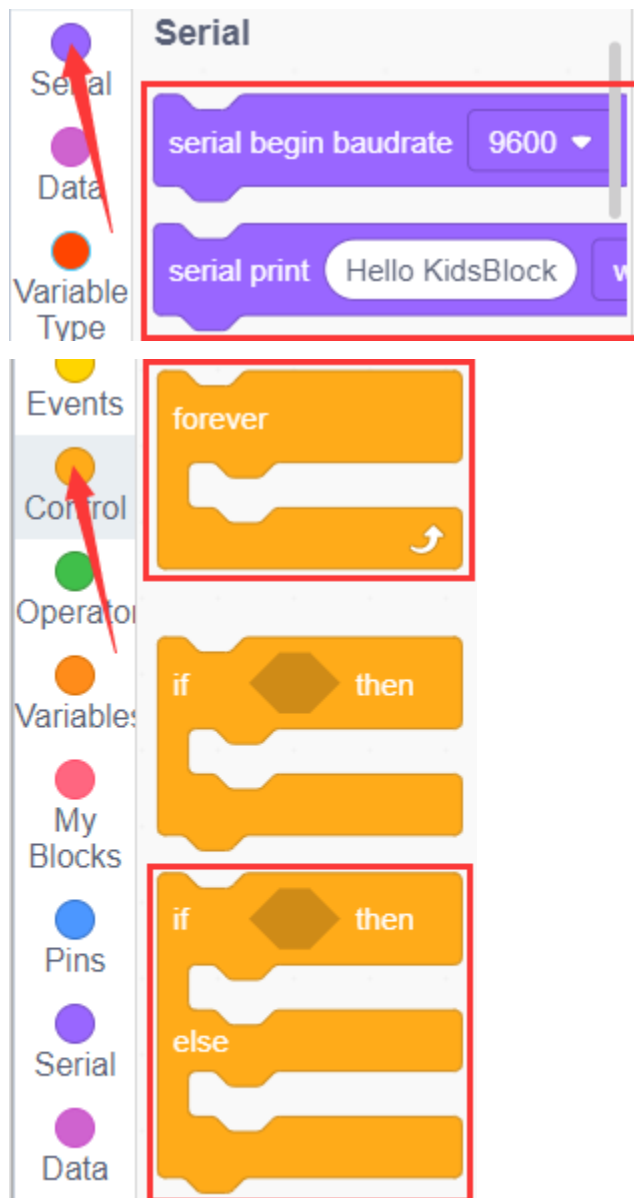
Robot without BT module	USB Cable*1	Computer*1	18650 Battery*2	Lighter*1
				
Yellow LED Module*1	3P-3P XH2.54 to 2.54 Wire			
				

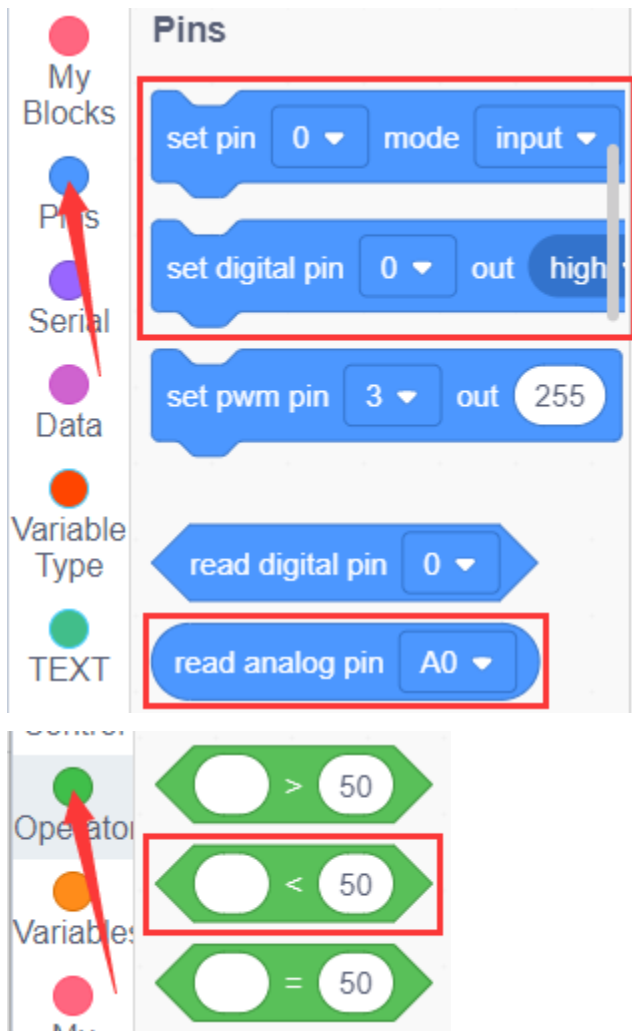
Note: 1This experiment requires the use of a fire source. Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. 2The flame sensoris not fireproof, please do not burn it directly with flame. We can control an external LED with the flame sensor. The LED still is connected to D9. When fire is connected, LED will be on.



You can drag blocs to edit your code, as shown below

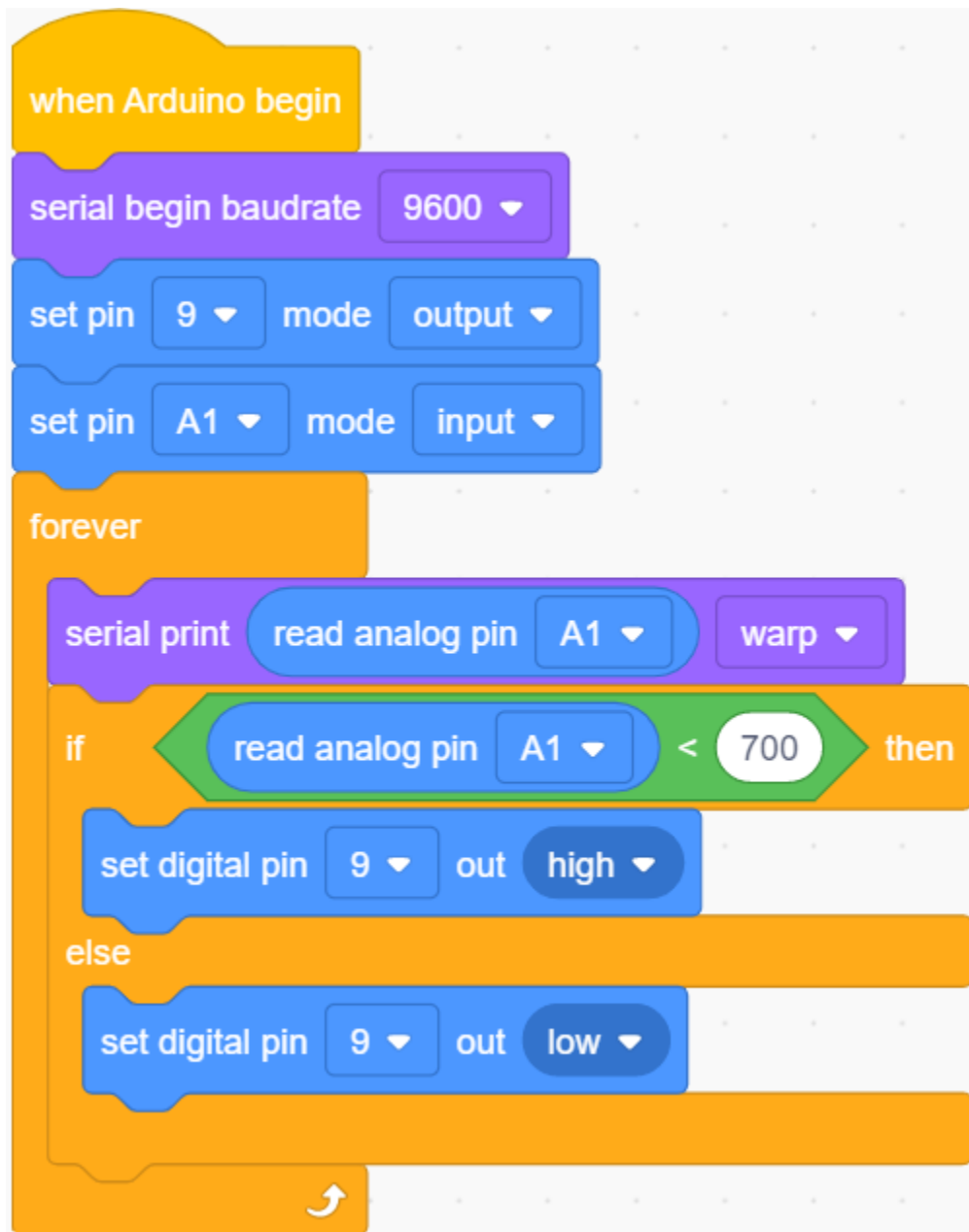






Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



You can use the flame of a lighter near the left flame sensor. When the flame sensor detects a flame, the LED module will light up as an alarm.



7.4.3 Project 20: Fan

(1)Description



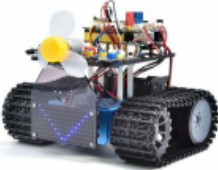



This fan module uses a HR1124S motor-controlling chip, a single-channel H-bridge driver chip containing a low-conductivity resistance PMOS and NMOS power tubes. The low-conducting resistance can ease the power consumption, contributing to the safe work of the chip for longer time.

In addition, its low standby current and low static working current makes itself apply to toys. We can control the rotation direction and speed of the fan by outputting IN + and IN- signals and PWM signals.

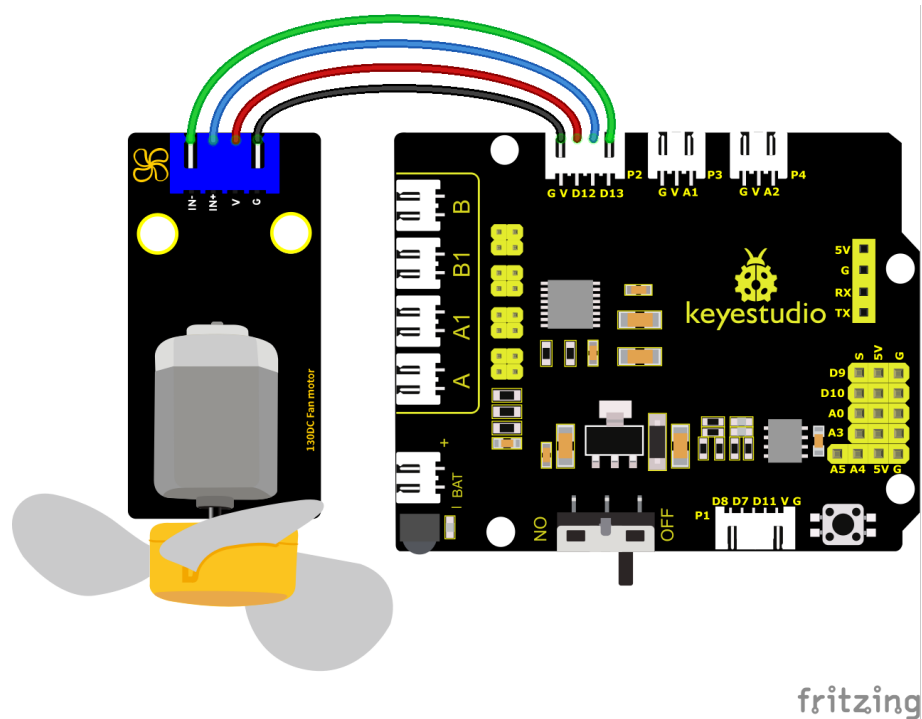
(2)Parameters:

- Working voltage: 5V
- Current: 200mA
- Maximum power: 2W
- Work temperature: -10 ° C to +50 degrees Celsius
- Size: 47.6mm * 23.8mm

(3) Components Needed:

Robot tank without BT module	USB Cable*1	Computer*1	18650 Battery*2
			

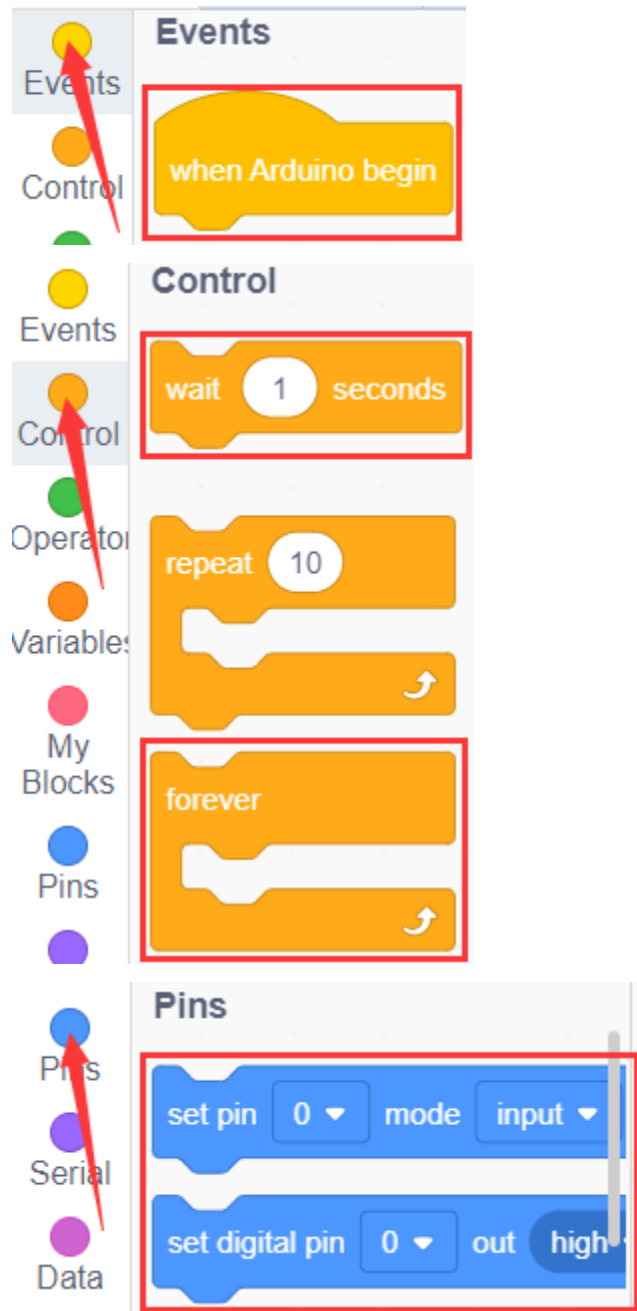
(4) Connection Diagram:



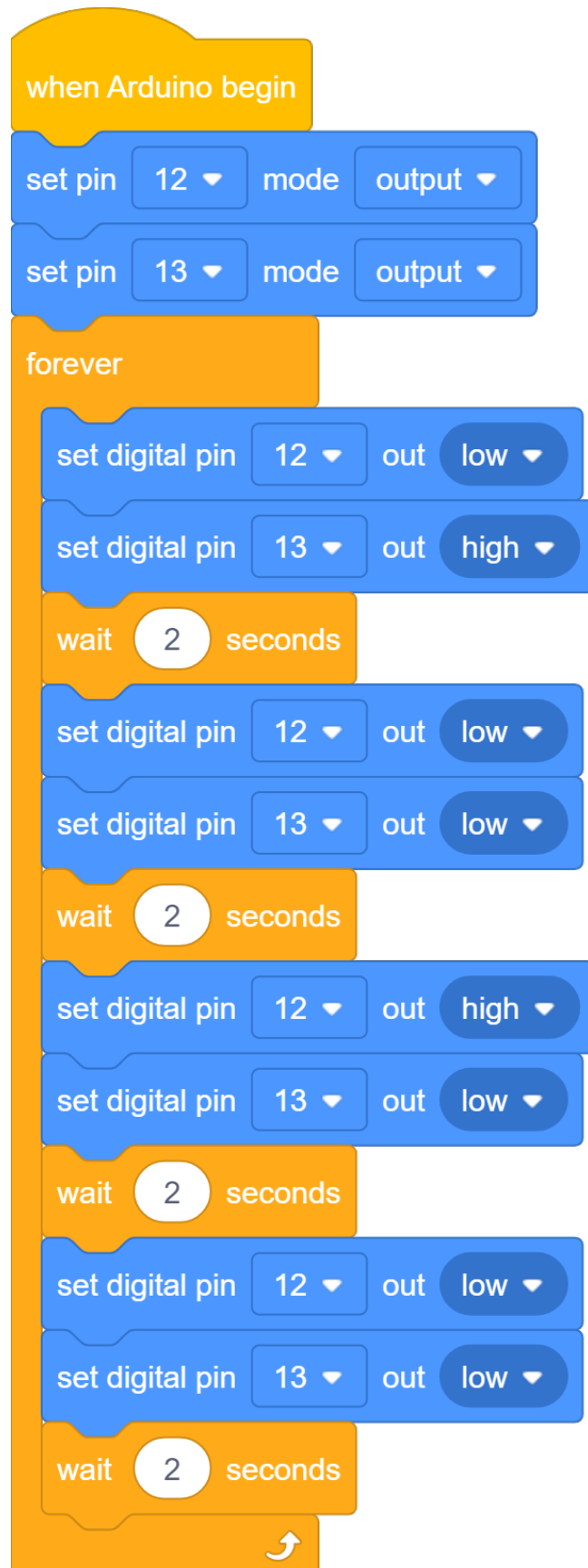
The pin GND, VCC, IN+ and IN- of the fan module are connected to pin G, V, 12 and 13 of the shield.

(5)Test Code:

You can also drag blocks to edit your code, as shown below

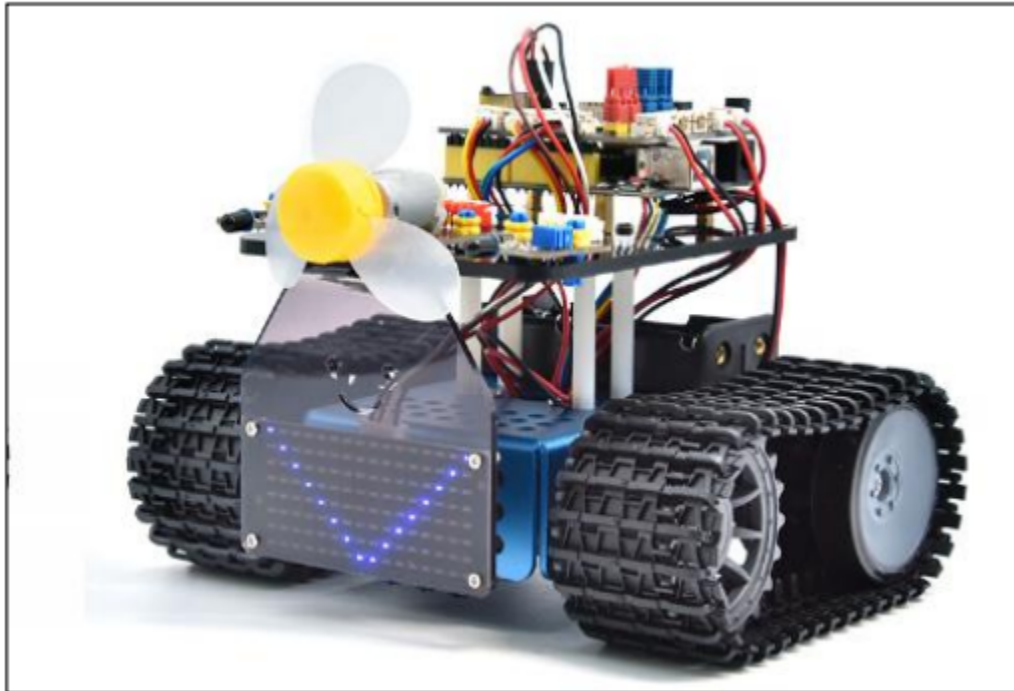
**Complete Test Code**

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

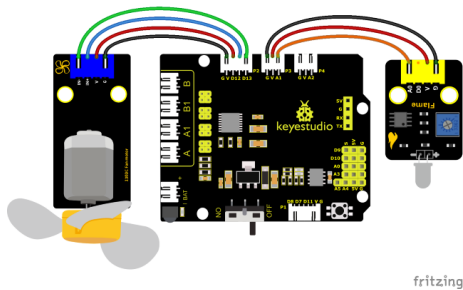


(6)Test Results:

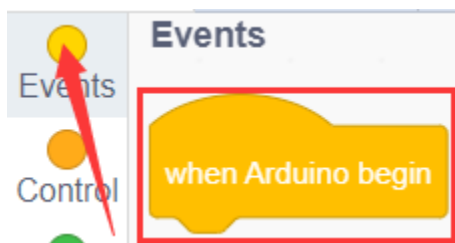
Upload code, wire up components, power on and turn the DIP switch to ON. The small fan will turn clockwise for 2s, stop for 2s and anticlockwise for 2s

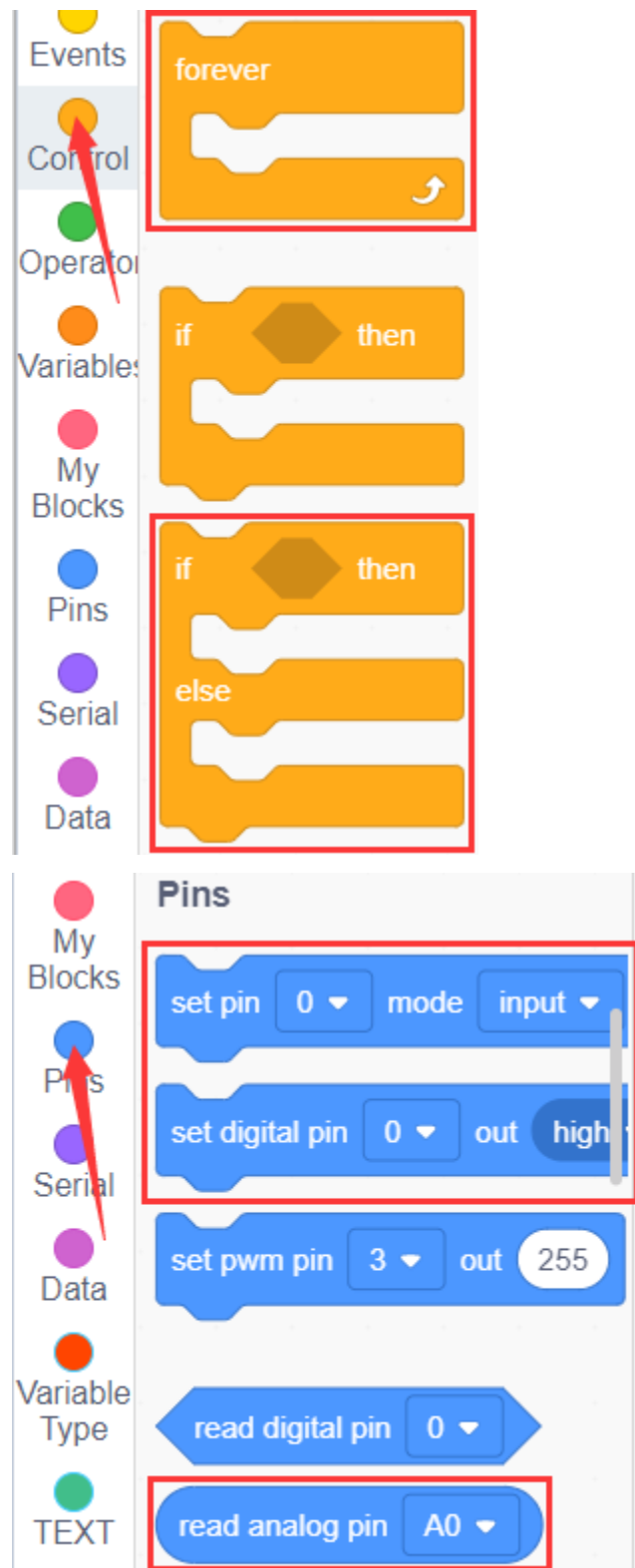
**(7)Extension Practice:**

We have understood the working principle of the flame sensor. Next, hook up a flame sensor in the circuit , as shown below. Then control the fan to blow out fire with the flame sensor.



You can drag blocs to edit your code, as shown below

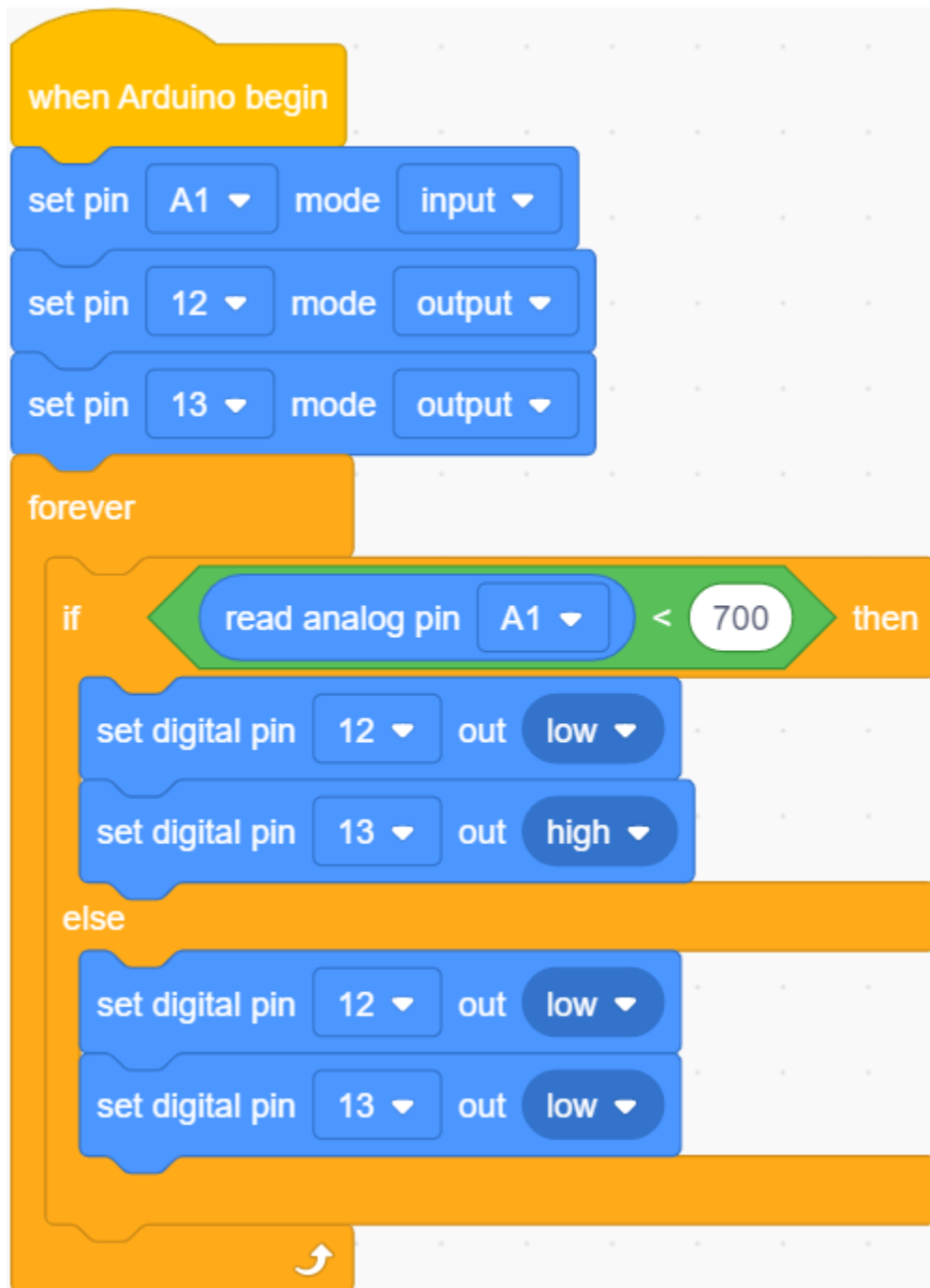




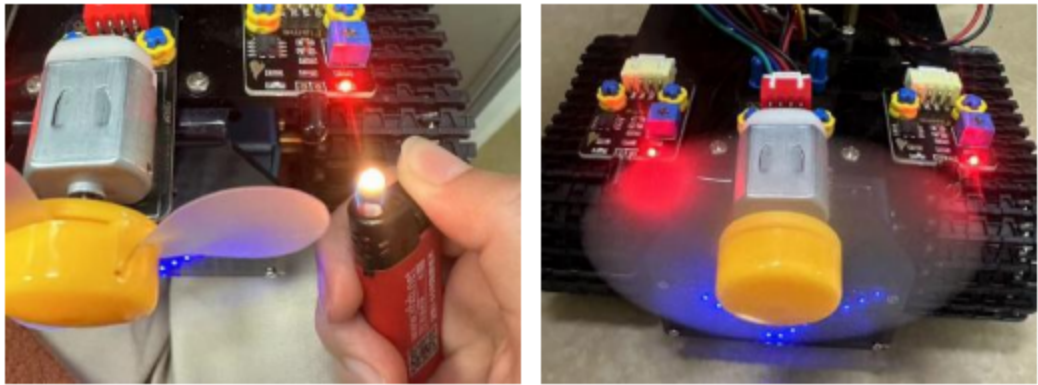


Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)



After uploading the code, turn on the power switch of the motor drive shield, you can turn on the fan when flame is detected from the left flame sensor of the robot.



7.4.4 Project 21: Fire Extinguishing Tank






(1)Description:

The line-tracking function of the smart tank has been explained in the previous project. And in this project we use the flame sensor to make a fire extinguishing robot. When the car encounters flames, the motor of the fan will rotate to blow out the fire. Of course, we need to replace the ultrasonic sensor and two photoresistors with a fan module and flame sensors first.

The specific logic of the line-tracking smart car is shown in the table blow:

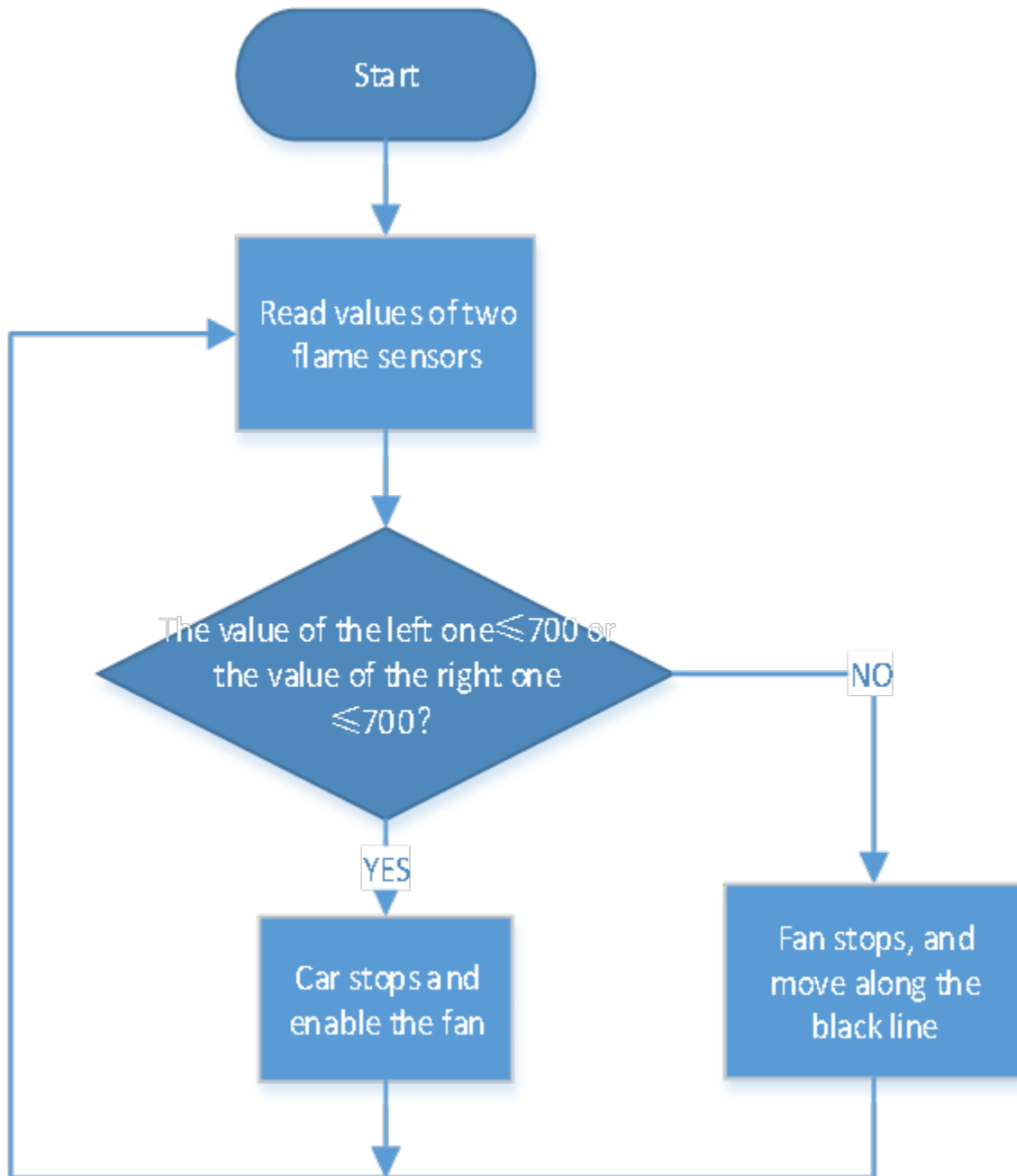
Left Flame Sensors	Right Flame Sensors	Status
700	700	Car stopsfan starts rotating to blow up flame
700	700	Car stopsfan starts rotating to blow up flame
700	700	Car stopsfan starts rotating to blow up flame
700	700	Fan stopscar moves

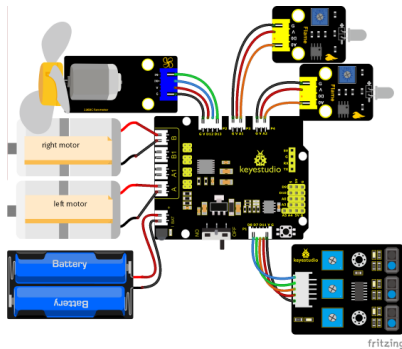
(2)Components Needed:

Robot without BT module	USB Cable*1	Computer*1	18650 Battery*2	Lighter*1
				

Note: 1This experiment requires the use of a fire source. Please make it away from flammable items to prevent fire. Children should experiment under adult supervision. If you cannot confirm that you are safe, please abandon the experiment. 2The flame sensor is not fireproof, please do not burn it directly with flame. We can control an external LED with the flame sensor. The LED is still connected to D9. When fire is connected, LED will be on.

(3)Flow chart:



(4) Connection Diagram:

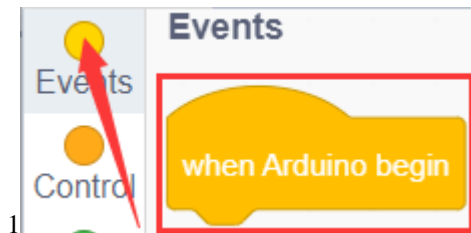
Note:

GND, VCC, SDA and SCL of the 8x16 LED panel are connected to GGND, VVCC), A4 and A5.

G, V and A of two flame sensors are interfaced with GGND), VVCC), A1 and A2 of the expansion board.

(5) Test Code:

You can also drag blocks to edit your code, as shown below



2

Events

Control

Operator

Variables

My Blocks

Pins

Serial

Data

3

My Blocks

Pins

Serial

Data

Variable Type

TEXT

DC

forever

if then

if then else

set pin 0 mode input

set digital pin 0 out high

set pwm pin 3 out 255

read digital pin 0

read analog pin A0

4

5

6

Variable Type

Declare Global variable Type int Name item Assigned to 0

variable item

Set item variable by 0

DC Motor

Motor INA# 2 State HIGH INB# 6 State HIGH

Motor INA# 2 State HIGH INB# 6 analogWrite 255

Events

Control

Operator

Variables

My Blocks

Pins

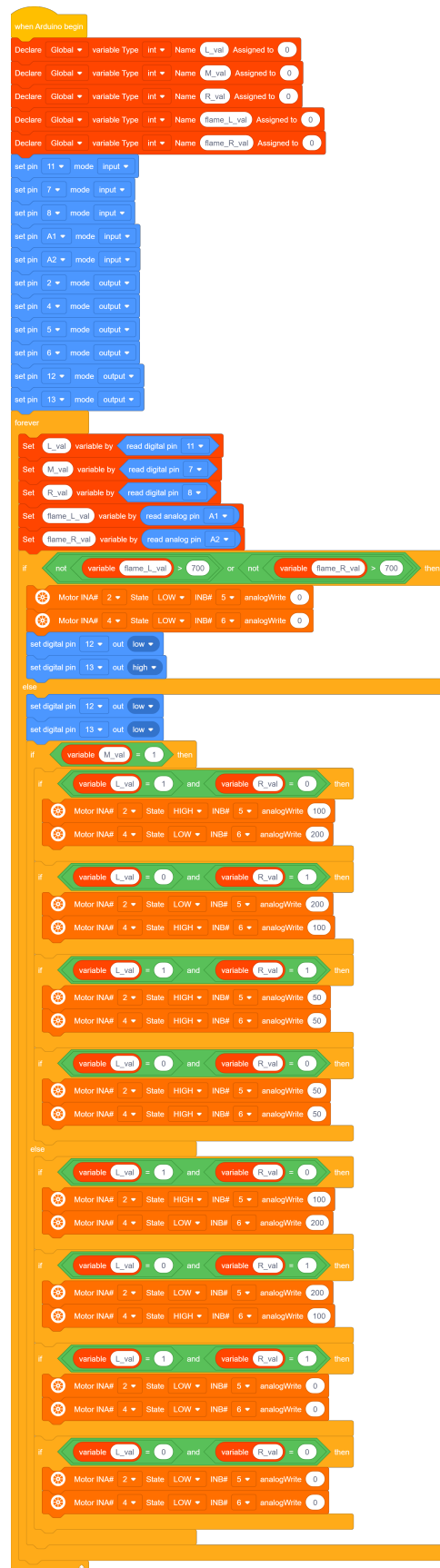
and

or

not

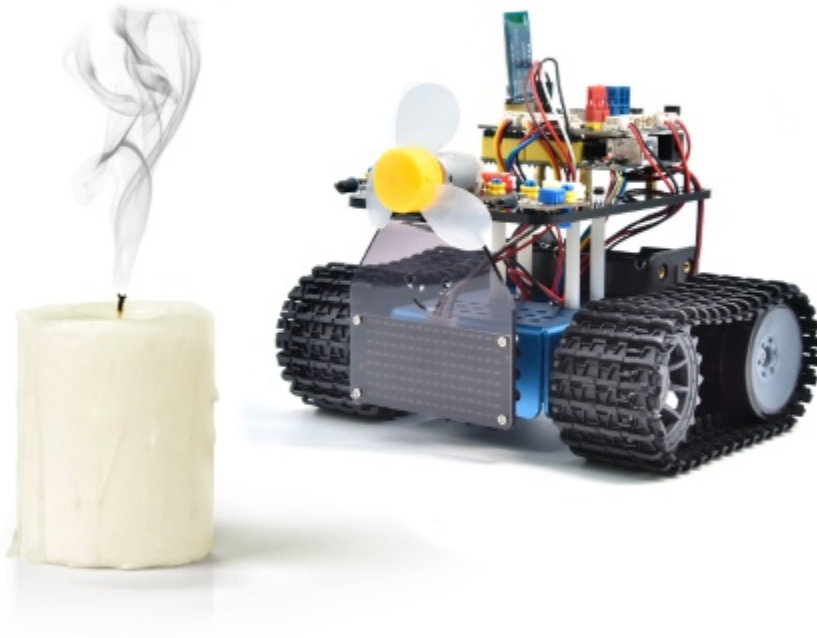
Complete Test Code

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.)

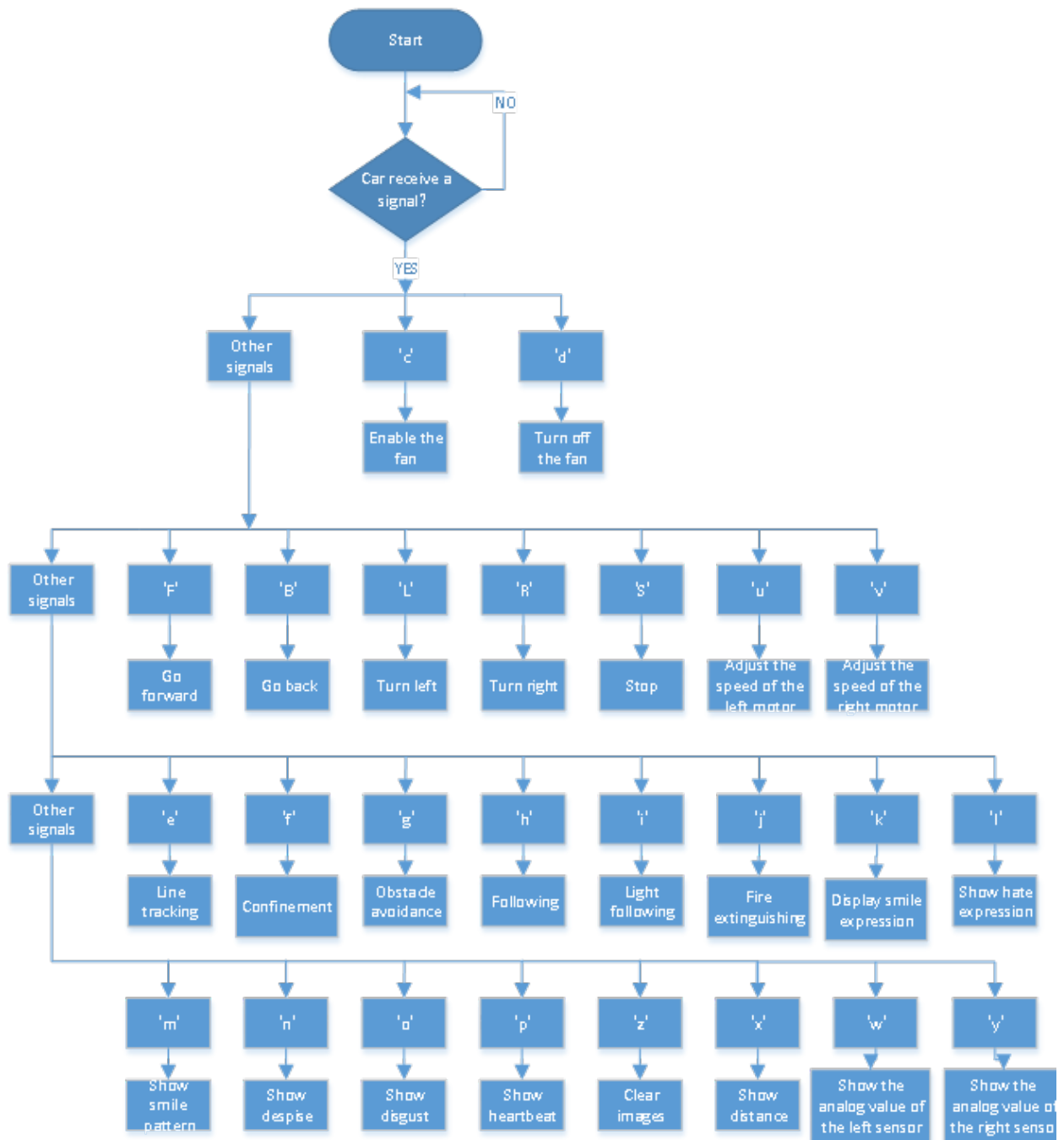


(6)Test Results:

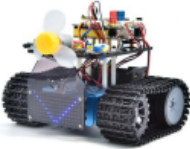




After upload the test code successfully, power up and turn the DIP switch to ON end. The smart car will put out the fire when it detects flame.

**7.4.5 Project 22: Fire Extinguishing Robot Multiple Functions****(1)Description:**

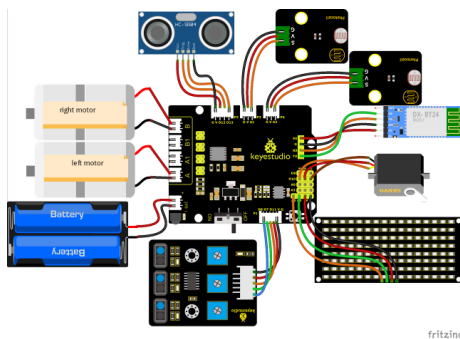
The smart car has performed a single function in every previous project. Can it display multiply functions at a time ? Positive. In this last big project, we intend to use a complete code to control the smart car to show off all functions mentioned in previous projects. We use the keys on the Bluetooth APP to automatically switch various functions, quite simple and convenient.



(2)Components Needed:

Robot without BT module	USB Cable*1	Computer*1	18650 Battery*2	Bluetooth module*1
				

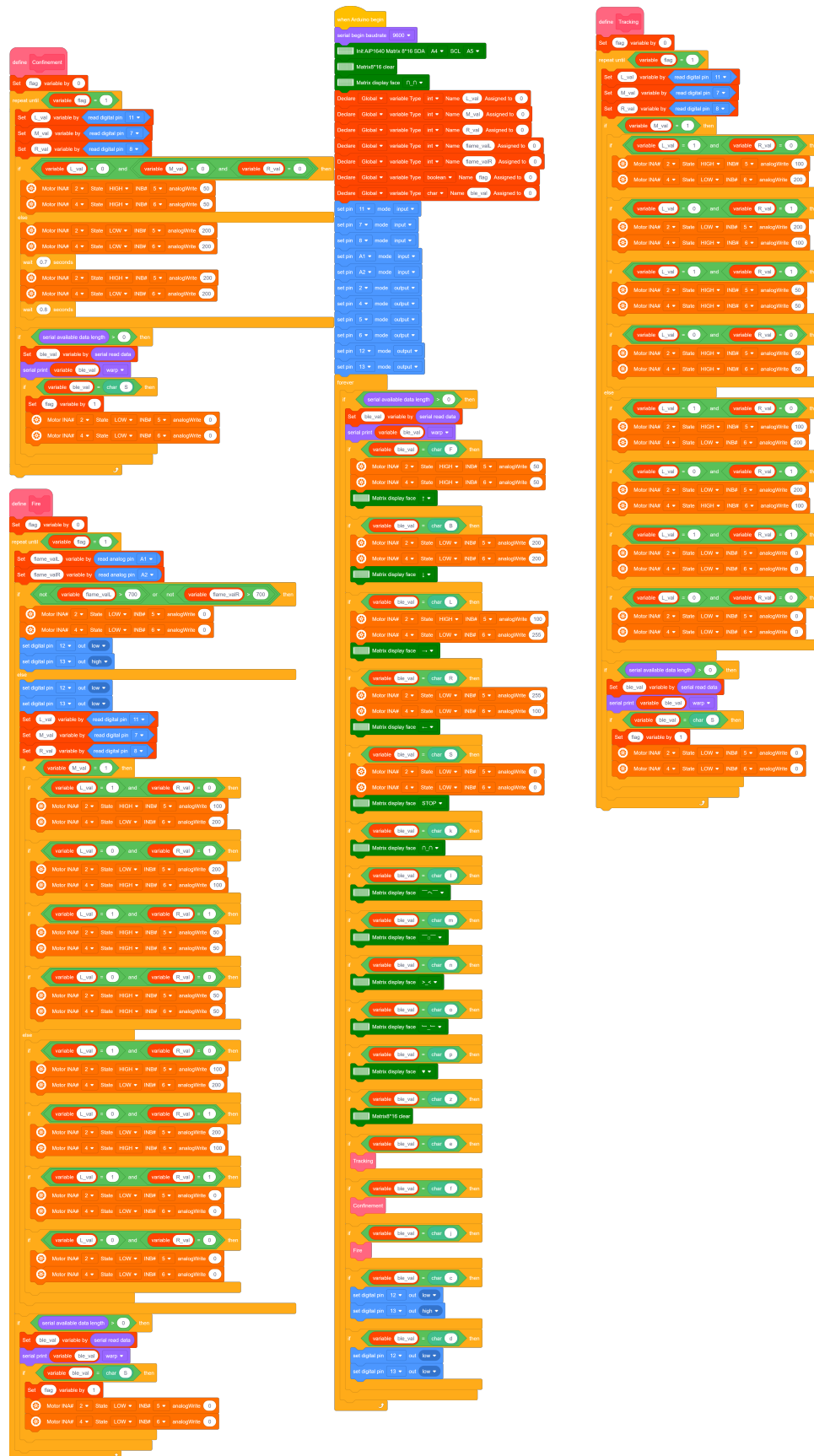
Please refer to Project 16 to install and configure Bluetooth APP

(3)Connection Diagram:

1. GND, VCC, SDA and SCL of the 8x16 board are connected to GND), +VCC), A4 and A5 of the expansion board.
2. VCC, IN+, IN- and Gnd of the ultrasonic sensor are connected to 5V(V), 12(S), 13(S) and Gnd(G)
3. The brown wire, red wire and orange wire of the servo are connected to Gnd(G), 5v(V) and D10.
4. RXD, TXD, GND and VCC of the BT module are connected to TX, RX, GGND) and 5VVCC. STATE and BRK don't need to be interfaced.
5. The pin "G", "V" and S of the left Flame sensors are connected to G (GND), V (VCC), A1 respectively; The right Flame sensors is connected to the G (GND), V (VCC), and A2 respectively.
6. The distal port of the line tracking sensor is 11, 7 and 8.

(4)Test Code:

(Note: Do not connect the Bluetooth module before uploading the code, because the uploading of the code also uses serial communication, and there may be conflicts with the serial communication of the Bluetooth, which can cause the uploading of the code to fail.) Note: you can't speed up the car via App.



(5)Test Result:

Before uploading the program code, the Bluetooth module needs to be removed, otherwise the code upload will fail.

After uploading the code successfully, open the positioning, and then connect the Bluetooth module.

After uploading the code successfully, plug in the Bluetooth module, after power-on, after the mobile APP is connected to the Bluetooth successfully, we can use the mobile APP to control the tank robot

You need to comment out the unused code, and uncomment the used code. Because some modules use the same IO port, you must comment out a part. The above two codes are the corresponding two ways to play.

