

---

# **Beetlebot 3 in 1 Robot**

**keyestudio WiKi**

**Jan 23, 2024**





## CONTENTS:

|          |  |            |
|----------|--|------------|
| <b>1</b> | <b>1. Libraries, All Codes and Other Details:</b>    | <b>3</b>   |
| <b>2</b> | <b>2. Description</b>                                | <b>5</b>   |
| <b>3</b> | <b>3. Features</b>                                   | <b>7</b>   |
| <b>4</b> | <b>4. Specification</b>                              | <b>9</b>   |
| <b>5</b> | <b>5. Kit List</b>                                   | <b>11</b>  |
| <b>6</b> | <b>6. Keystudio Nano Board</b>                       | <b>19</b>  |
| 6.1      | 1. Description: . . . . .                            | 19         |
| 6.2      | 2. Specification: . . . . .                          | 19         |
| 6.3      | 3. Pins: . . . . .                                   | 20         |
| <b>7</b> | <b>7. PCB Board</b>                                  | <b>25</b>  |
| <b>8</b> | <b>Arduino tutorial</b>                              | <b>27</b>  |
| 8.1      | 1. Get started with Arduino . . . . .                | 27         |
| 8.2      | 2. Assemble Beetlebot Robot . . . . .                | 28         |
| 8.3      | 3. Projects . . . . .                                | 54         |
| 8.3.1    | Project 1: LED Blinking . . . . .                    | 54         |
| 8.3.2    | Project 2: 6812 RGB . . . . .                        | 56         |
| 8.3.3    | Project 3: Play Music . . . . .                      | 61         |
| 8.3.4    | Project 4: 8*8 Dot Matrix . . . . .                  | 65         |
| 8.3.5    | Project 5: Servo Rotation . . . . .                  | 75         |
| 8.3.6    | Project 6: Motor Driving and Speed Control . . . . . | 78         |
| 8.3.7    | Project 7: Ultrasonic Sensor . . . . .               | 81         |
| 8.3.8    | Project 8: Line Tracking Sensor . . . . .            | 97         |
| 8.3.9    | Project 9: Light Following . . . . .                 | 103        |
| 8.3.10   | Project 10: IR Remote Control . . . . .              | 110        |
| 8.3.11   | Project 11: WIFI Control . . . . .                   | 123        |
| <b>9</b> | <b>Kidsblock tutorial</b>                            | <b>179</b> |
| 9.1      | 1. Getting started with Kidsblock software . . . . . | 179        |
| 9.1.1    | 1. Instruction: . . . . .                            | 179        |
| 9.1.2    | 2. Download and install KidsBlock software . . . . . | 179        |
| 9.1.3    | 3. Interface Setting . . . . .                       | 179        |
| 9.2      | 2. Assemble Beetlebot Robot . . . . .                | 186        |
| 9.3      | 3. Projects . . . . .                                | 212        |
| 9.3.1    | Project 1: LED Blinking . . . . .                    | 212        |

|           |  |            |
|-----------|--|------------|
| 9.3.2     | Project 2: 6812 RGB . . . . .                        | 214        |
| 9.3.3     | Project 3: Play Music . . . . .                      | 217        |
| 9.3.4     | Project 4: 8*8 Dot Matrix . . . . .                  | 219        |
| 9.3.5     | Project 5: Servo Rotation . . . . .                  | 230        |
| 9.3.6     | Project 6: Motor Driving and Speed Control . . . . . | 234        |
| 9.3.7     | Project 7: Ultrasonic Sensor . . . . .               | 239        |
| 9.3.8     | Project 8: Line Tracking Sensor . . . . .            | 254        |
| 9.3.9     | Project 9: Light Following . . . . .                 | 261        |
| 9.3.10    | Project 10: IR Remote Control . . . . .              | 269        |
| 9.3.11    | Project 11: WIFI Control . . . . .                   | 281        |
| <b>10</b> | <b>Soccer tutorial . . . . .</b>                     | <b>323</b> |
| 10.1      | 1. Description . . . . .                             | 324        |
| 10.2      | 2. How to install the soccer robot: . . . . .        | 324        |
| 10.3      | 3. Install a soccer goal . . . . .                   | 343        |
| 10.4      | 4. Codes: . . . . .                                  | 347        |
| 10.4.1    | (1)Arduino Code . . . . .                            | 347        |
| 10.4.2    | (2)Kidsblock Code . . . . .                          | 351        |
| 10.5      | 5. Test Result . . . . .                             | 352        |
| <b>11</b> | <b>Catapul tutorial . . . . .</b>                    | <b>353</b> |
| 11.1      | <b>1. Description . . . . .</b>                      | <b>354</b> |
| 11.2      | <b>2. How to build up a catapult . . . . .</b>       | <b>354</b> |
| 11.3      | 3. Code: . . . . .                                   | 382        |
| 11.3.1    | (1) Arduino Code . . . . .                           | 382        |
| 11.3.2    | (2) Kidsblock Code . . . . .                         | 386        |
| 11.4      | 4. Test Result . . . . .                             | 387        |
| <b>12</b> | <b>Handling tutorial . . . . .</b>                   | <b>389</b> |
| 12.1      | 1. Description . . . . .                             | 390        |
| 12.2      | 2. How to build up a handling robot: . . . . .       | 390        |
| 12.3      | 3. Code: . . . . .                                   | 408        |
| 12.3.1    | (1)Arduino Code . . . . .                            | 408        |
| 12.3.2    | (3)Kidsblock Code** . . . . .                        | 412        |
| 12.4      | 4. Test Result . . . . .                             | 413        |





## **1. LIBRARIES, ALL CODES AND OTHER DETAILS:**

- Arduino\_Libraries
- All\_Codes
- Other\_Details



## 2. DESCRIPTION

The Beetlebot smart robot, compatible with LEGO building blocks, is a STEM educational product which can automatically dodge obstacles, follow black lines and light to move. Besides, it has three cool forms such as the soccer robot, the siege robot, the handling robot. As for beginners, they can create whatever they want by LEGO building blocks.

Various improvements have been made on the Beetlebot car in comparison with other smart cars. It integrates a motor driver and a large number of sensors and is easy to assemble.

Going forward, not only can it impart basic programming knowledge and AI application to children and the youth, but also it can cultivate their creativity, hands-on ability, problem-solving capability, interpersonal communication as well as teamwork ability. With this kit, you have a chance to experience soccer games using your own robots.





### 3. FEATURES

- Compatible with LEGO building blocks: generate diverse forms with LEGO blocks and sensors.
- Three forms: a soccer robot, a siege engine, a handling robot.
- Various functions: Pictures display, atmosphere light control, line tracking, obstacle avoidance, light following , IR control and WIFI control.
- Easy to build: embedded design on car body; wire up the car body with a few steps.
- High compatibility: reserve ports for the Raspberry Pico board and the ESP32 control board.
- Charging function: integrate a circuit for 18650 batteries, low-cost and effective.
- WiFi Control: adopt WiFi control, can finish tailor-made software development.
- App: compatible with Android and iOS systems, with aesthetic page and flexible control system.



## 4. SPECIFICATION

Working voltage: 5V

Input voltage: 2.5V~4.2V (lithium battery)

Maximum output current: 3A

Maximum power consumption: 15W (T=80°C)

Motor speed: 5V 200 rpm / min

Motor drive form: dual H-bridge

Ultrasonic sensing angle: <15 degrees

Ultrasonic detection distance: 2cm-400cm

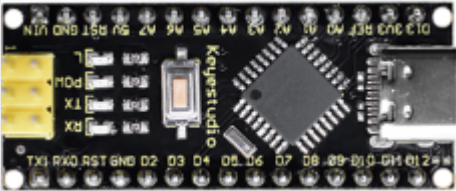
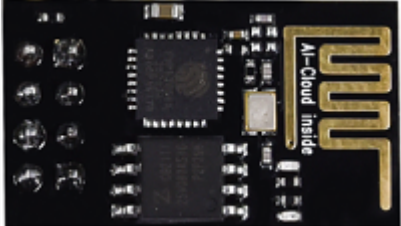
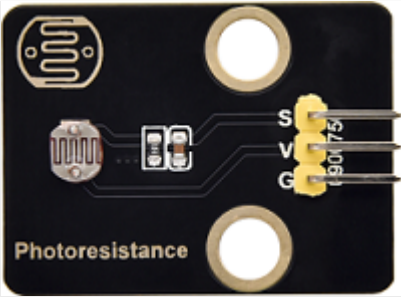
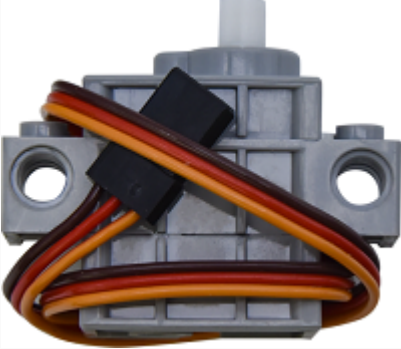
IR control distance: about 7 meters (measured)

Size: 176mm\*137mm\*130mm

Environmental protection attributes: ROHS

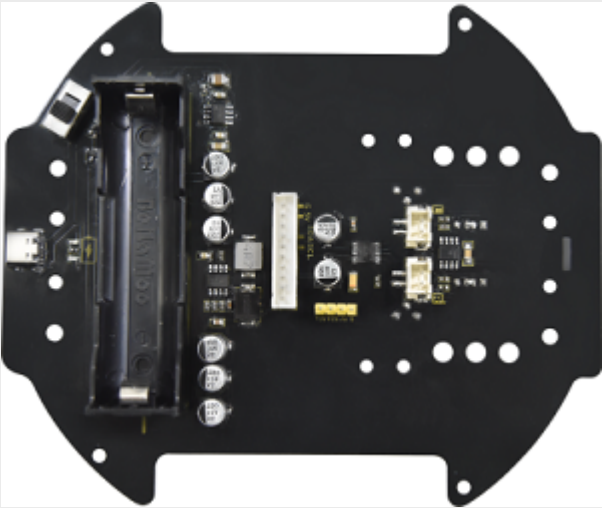
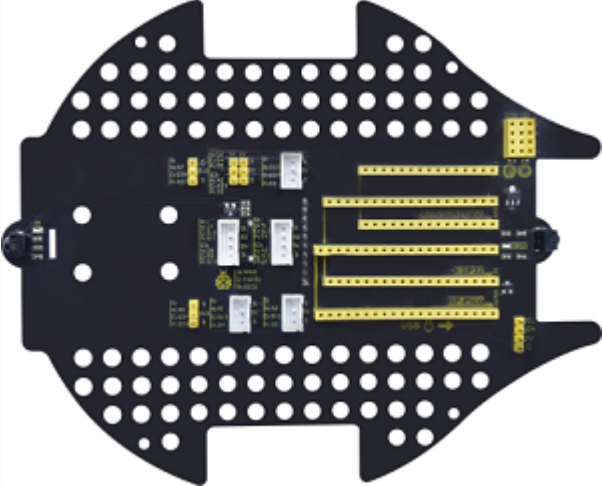



5. KIT LIST

| # | Picture   | Name                                    |
|---|---|---|
| 1 |    | Keyestudio Nano CH340 Development Board |
| 2 |   | ESP8266 Wifi Module                     |
| 3 |  | Keyestudio Photoresistor                |
| 4 |  | 270° Servo                              |

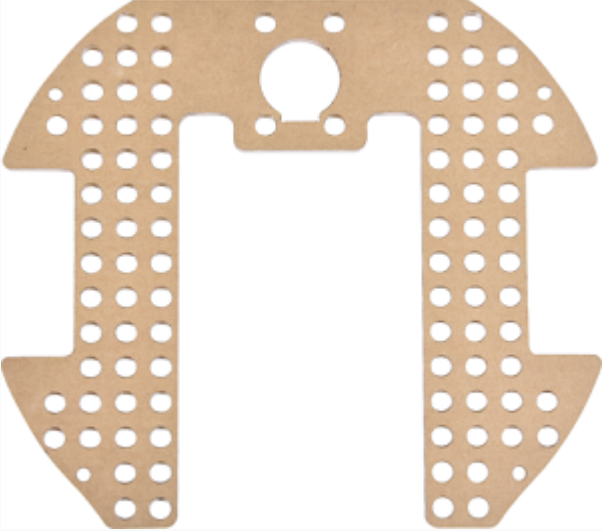

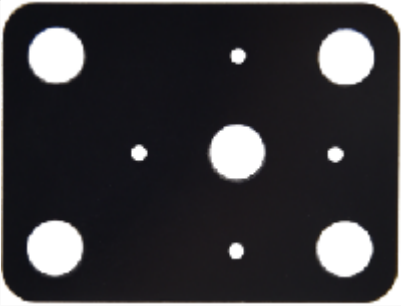
continuu

Table 1 – continued from previous page

| # | Picture   | Name                         |
|---|---|------------------------------|
| 5 |    | Keyestudio Development Board |
| 6 |   | Keyestudio Driver Board      |
| 7 |  | LEGO Bulk Lot                |

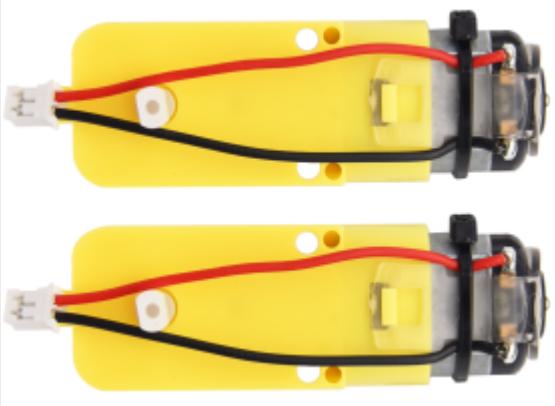
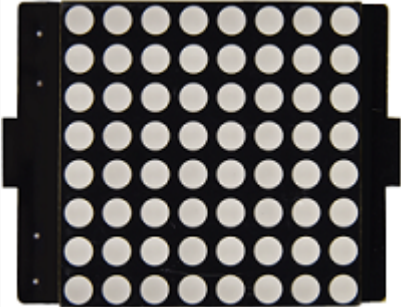
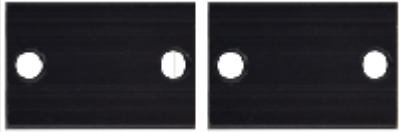

continues

Table 1 – continued from previous page

| #  | Picture   | Name                                       |
|----|---|--|
| 8  |    | Acrylic Board                              |
| 9  |   | MD0487 Acrylic Board for Ultrasonic Sensor |
| 10 |  | Acrylic Board for Servo                    |

continuation




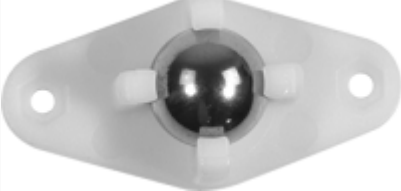




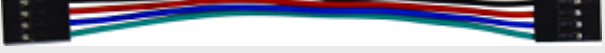

Table 1 – continued from previous page

| #  | Picture   | Name                   |
|----|---|------------------------|
| 11 |    | 4.5V 200R Motor        |
| 12 |    | 8*8 Dot Matrix Display |
| 13 |  | Aluminum               |
| 14 |  | 9G 180°Servo           |

continu

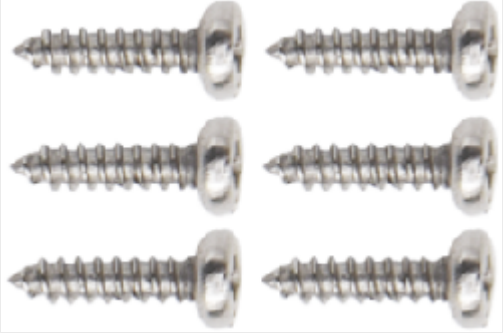


Table 1 – continued from previous page

| #  | Picture   | Name                         |
|----|---|------------------------------|
| 15 |    | Car Wheel                    |
| 16 |    | HC-SR04 Ultrasonic Sensor    |
| 17 |    | Screwdriver                  |
| 18 |   | W420 Universal Wheel         |
| 19 |  | JMFP-4 17-Key Remote Control |
| 20 |  | Black USB Cable              |
| 21 |  | Screwdriver                  |
| 22 |  | 3P F-F Dupont Wire           |
| 23 |  | 4P F-F Dupont Wire           |
| 24 |  | HX2.54mm-4P Dupont Wire      |


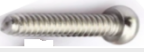







continuu

Table 1 – continued from previous page

| #  | Picture   | Name                             |
|----|---|----------------------------------|
| 25 |    | Winding Pipe                     |
| 26 |    | 10P XH2.54 Dupont Wire           |
| 27 |   | Acrylic Gasket                   |
| 28 |  | M3*40MM Dual Pass Copper Pillars |
| 29 |  | M1.2*5MM Round Head Screws       |

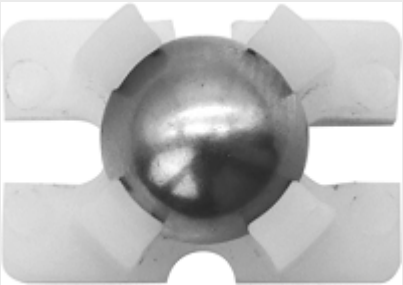


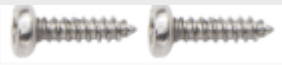
contin

Table 1 – continued from previous page

| #  | Picture   | Name                        |
|----|---|-----------------------------|
| 30 |    | M1.4 Nuts                   |
| 31 |    | M1.4*10MM Round Head Screws |
| 32 |    | M2 Nuts                     |
| 33 |    | M2*8MM Round Head Screws    |
| 34 |    | M3*10MM Round Head Screws   |
| 35 |   | M3*6MM Round Head Screws    |
| 36 |  | M3 Nuts                     |
| 37 |  | M3*30MM Round Head Screws   |
| 38 |  | Soccer Ball                 |

contin

Table 1 – continued from previous page

| #  | Picture   | Name  |
|----|---|---|
| 39 |  | W1515 Universal Wheel   |
| 40 |  | 18650 Batteries KS0543F includes batteriesKS0543 doesn't conclude |
| 41 |  | USB to ESP-01S WIFI Module Expansion Board                        |
| 42 |  | M2.3*16MM Round Head Self-tapping Screw                           |

## 6. KEYESTUDIO NANO BOARD

### 6.1 1. Description:

The processor core of Keystudio Nano CH340 is ATMEGA328P-AU. It is as same as the official Arduino Nano in addition to driver file and USB to serial chip (CH340G).

It also has 14 digital input / output interfaces (6 of which can be used as PWM output), 8 analog input interfaces, 1 16MHz crystal oscillator, 1 mini USB port, 1 ICSP interface, and a reset button.

The ICSP interface is used to program the Atmega328P-Au. We can supply power with a USB cable, the port VIN GND (DC 7-12V) and GND

### 6.2 2. Specification:

Microcontroller ATMEGA328P-AU

Operating Voltage: 5V

Input Voltage (recommended) DC 7-12V

Digital I/O Pins 14 (D0-D13)

PWM Digital I/O Pins 6 (D3 D5 D6 D9 D10 D11)

Analog Input Pins: 8 (A0-A7)

DC Current per I/O Pin: 40 mA

Flash Memory 32 KB of which 2 KB used by bootloader

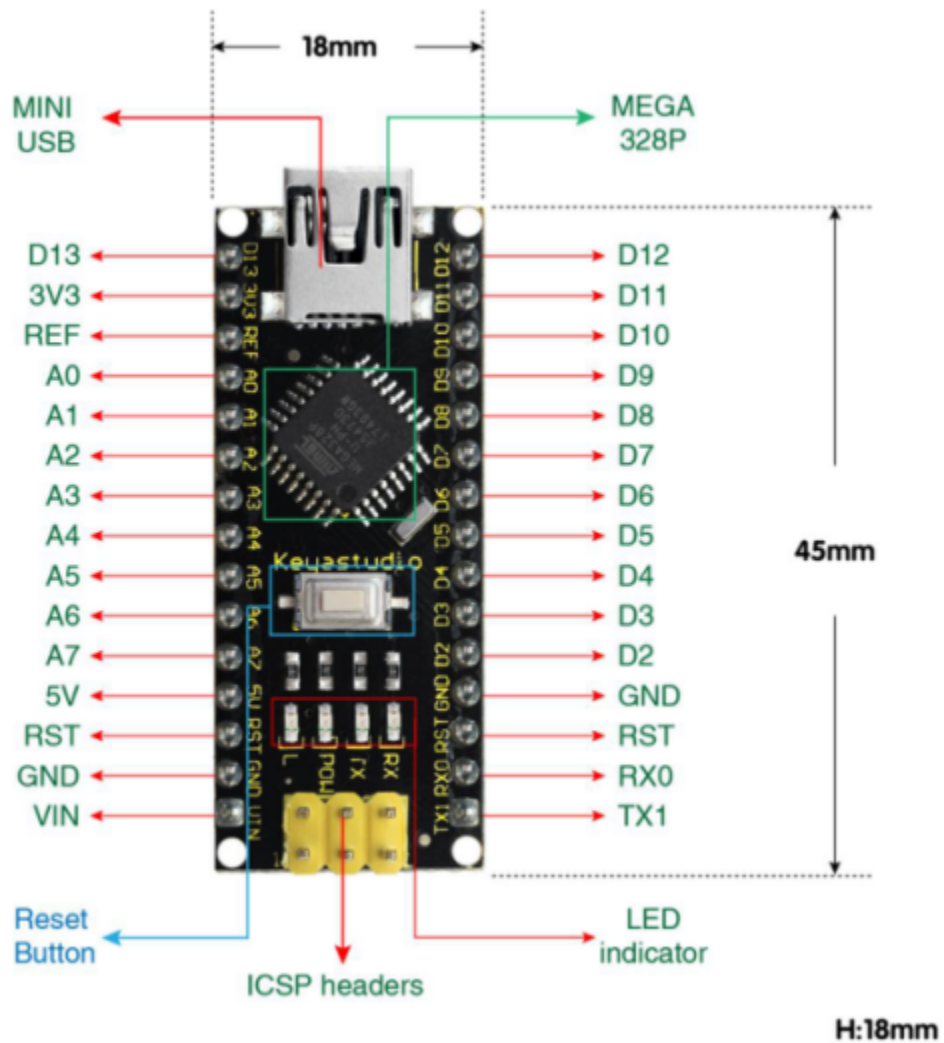
SRAM: 2 KB

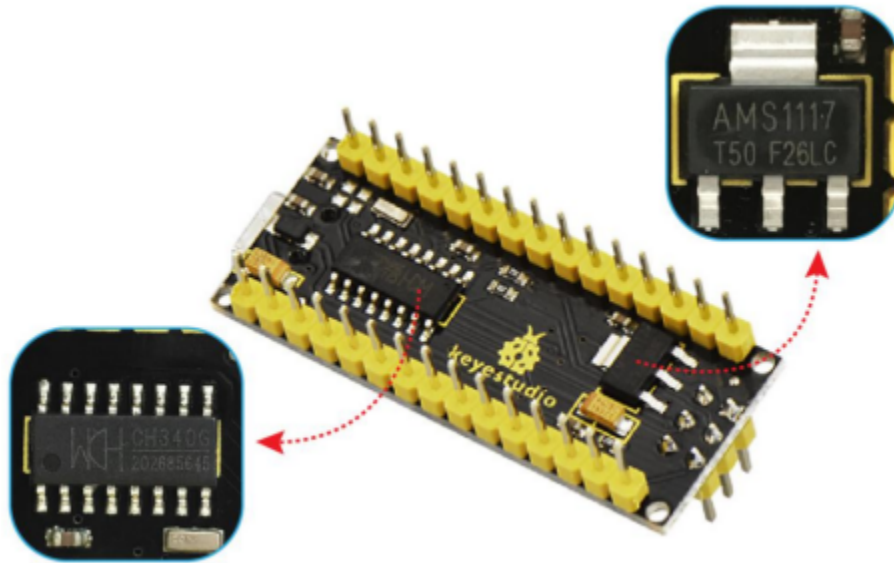
EEPROM: 1 KB

Clock Speed: 16 MHz

LED\_BUILTIN: D13

### 6.3 3. Pins:





|    |                   |  |
|----|-------------------|--|
| 1  | ICSP Header       | ICSP(In-Circuit Serial Programming) Header ICSP is the AVR, an micro-program header consisting of MOSI, MISO, SCK, RESET, VCC, and GND. It is often called the SPI (serial peripheral interface) and can be considered an “extension” of output. In fact, slave the output devices under the SPI bus host. When connecting to PC, program the firmware to ATMEGA328P-AU. |
| 2  | LED indicatorRX   | Onboard you can find the label: RX(receive )When control board communicates via serial port, receive the message, RX led flashes.  |
| 3  | LED indicatorTX   | Onboard you can find the label: TX (transmit)When control board communicates via serial port, send the message, TX led flashes.  |
| 4  | LED indicator-POW | Power up the control board, LED on, otherwise LED off.   |
| 5  | LED indicatorL    | There is a built-in LED driven by digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.  |
| 6  | RX0D13            | It has 14 digital input/output pins D0-D13 (of which 6 can be used as PWM outputs). These pins can be configured as digital input pin to read the logic value (0 or 1). Or used as digital output pin to drive different modules like LED, relay, etc.   |
| 7  | RST               | Reset pin: connect external button. The function is the same as RESET button.  |
| 8  | MEGA328P          | Each board has its own microcontroller. You can regard it as the brain of your board. Microcontrollers are usually from ATMEL. Before you load a new program on the Arduino IDE, you must know what IC is on your board. This information can be checked at the top surface of IC. The board's microcontroller is ATMEGA328P-AU. More info.                              |
| 9  | MINI USB          | The board can be powered via Mini-B USB connection. Also upload the program to the board via USB port.   |
| 10 | 3V3 pin           | Provides 3.3V voltage output   |
| 11 | REF               | Reference external voltage (0-5 volts) for the analog input pins. Used with <a href="#">analogReference()</a> .  |
| 12 | A0-A7             | The Nano has 8 Analog Pins, labeled A0 through A7.   |
| 13 | 5V                | Provides 5V voltage output   |
| 14 | GND               | Ground pin   |
| 15 | VIN               | Input an external voltage DC7-12V to power the board.  |
| 16 | Reset Button      | Used to reset the control board  |
| 17 | CH340             | USB-to-serial port chip, converting the USB signal into Serial port signal.  |
| 18 | AMS1117           | Convert the external voltage input DC7-12V into DC5V, then transfer it to the processor and other elements.  |

## Specialized Functions of Some Pins:

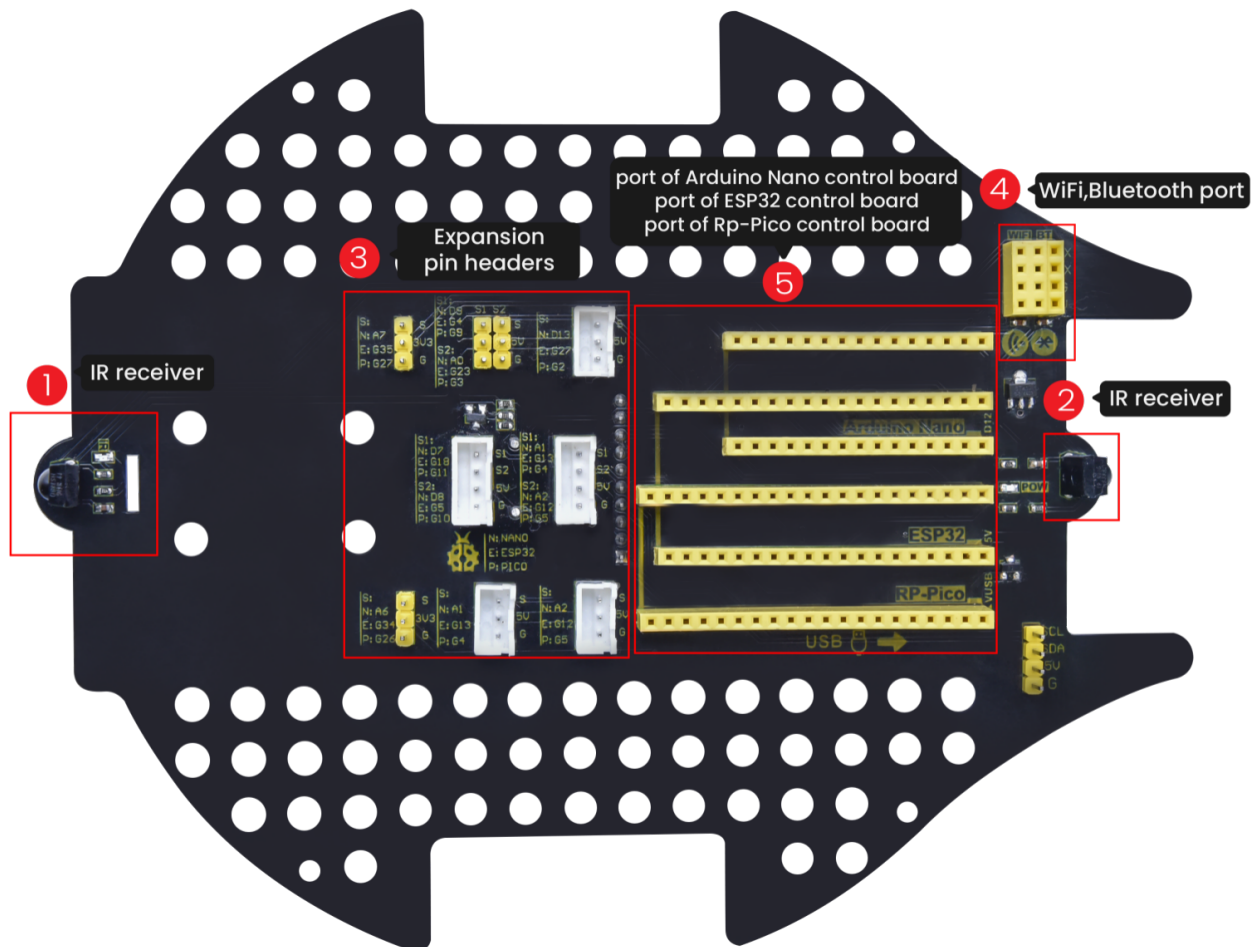
- **Serial communication:** RX0 and TX1.
- **PWM (Pulse-Width Modulation):** D3, D5, D6, D9, D10, D11
- **External Interrupts:** D2 (interrupt 0) and D3 (interrupt 1)

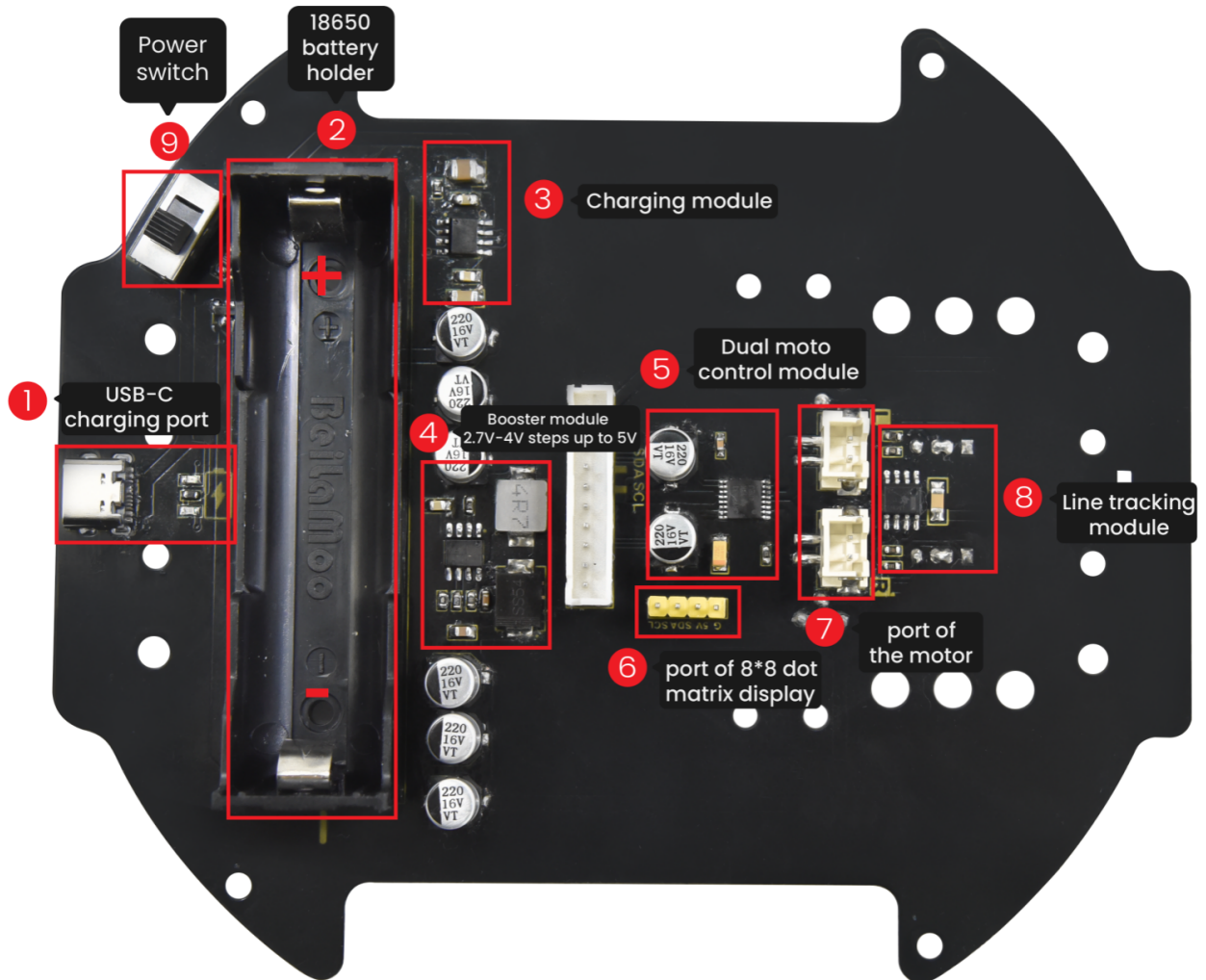


- **SPI communication:** D10 (SS), D11 (MOSI), D12 (MISO), D13 (SCK).
- **IIC communication:** A4 (SDA); A5(SCL)



## 7. PCB BOARD

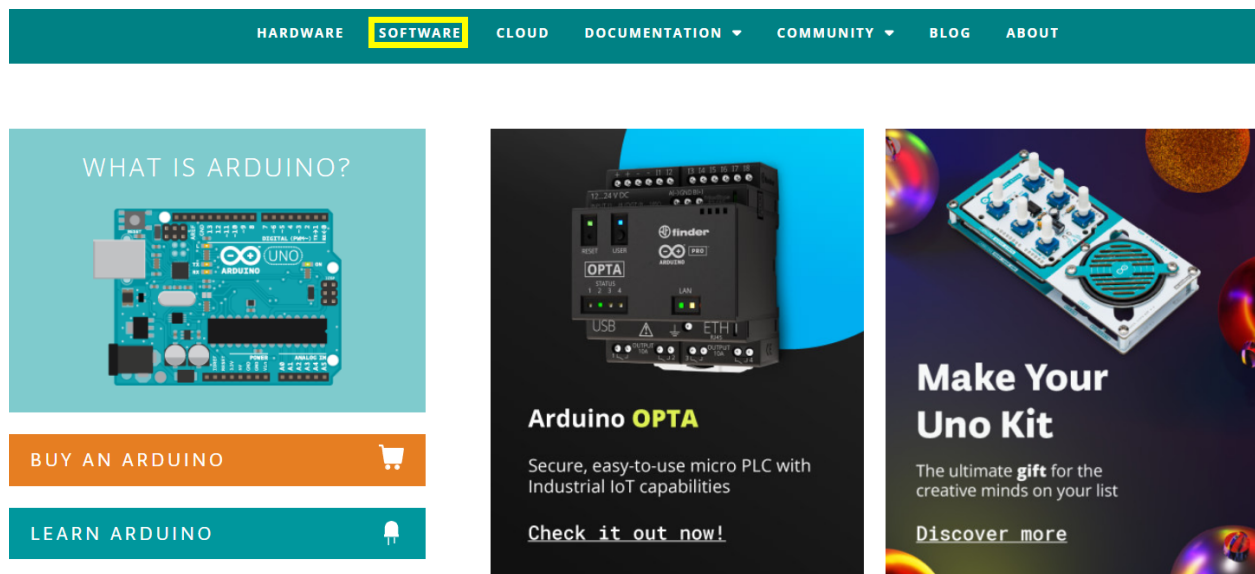




Turn the DIP switch to the OFF end before installing or removing batteries.

## ARDUINO TUTORIAL

### 8.1 1. Get started with Arduino



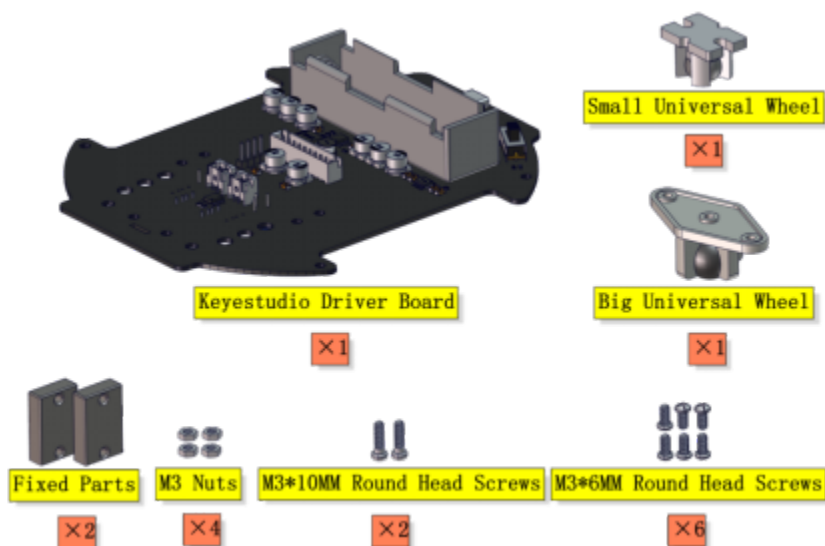
Click the link to start learning how to download software, install drivers, upload code, and the ways of install library files.

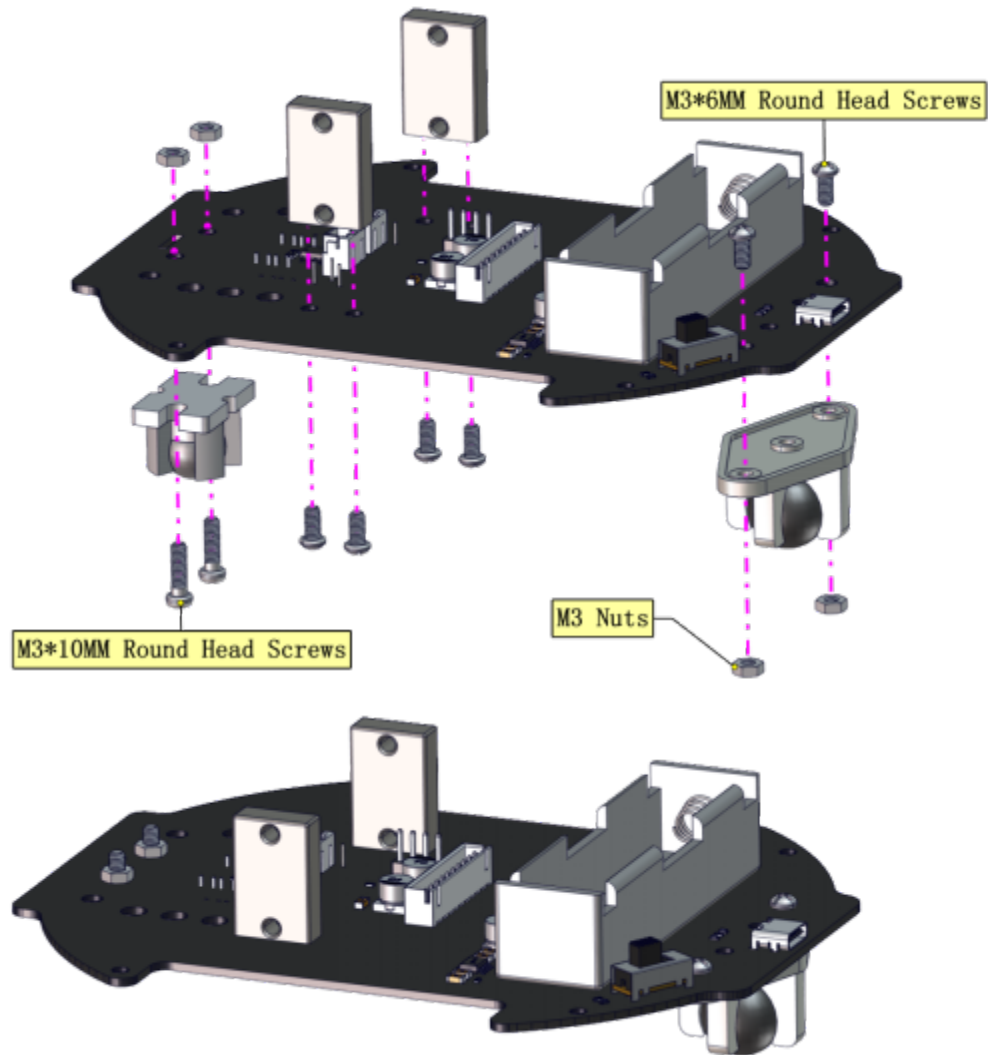
<https://getting-started-with-arduino.readthedocs.io>

## 8.2 2. Assemble Beetlebot Robot

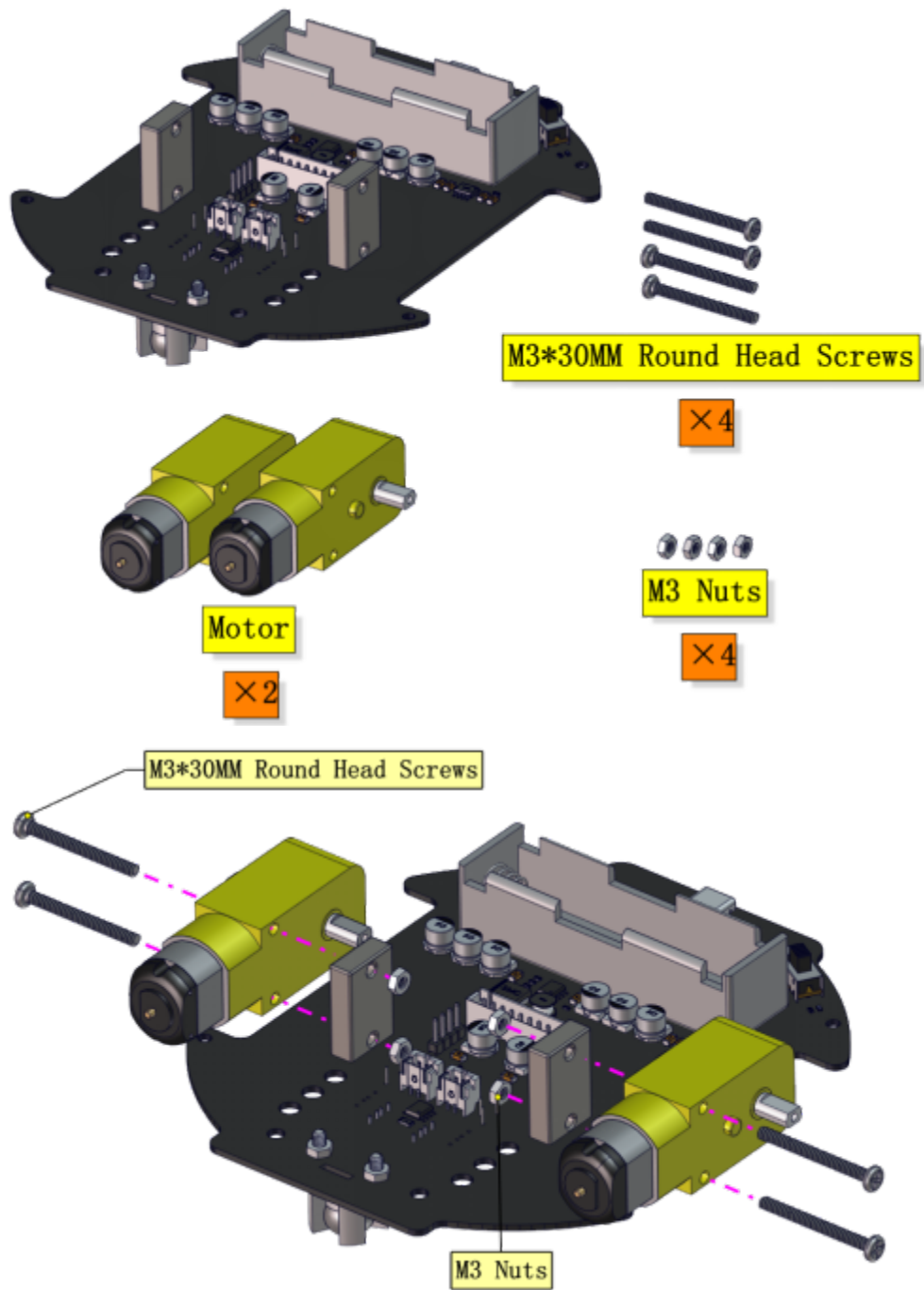


### Step 1

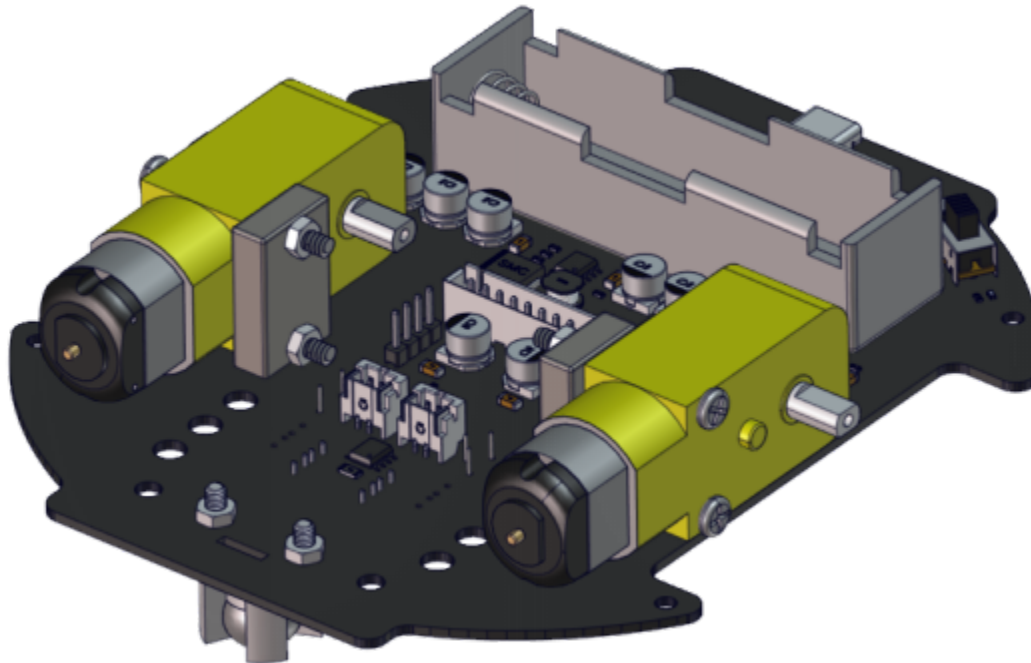




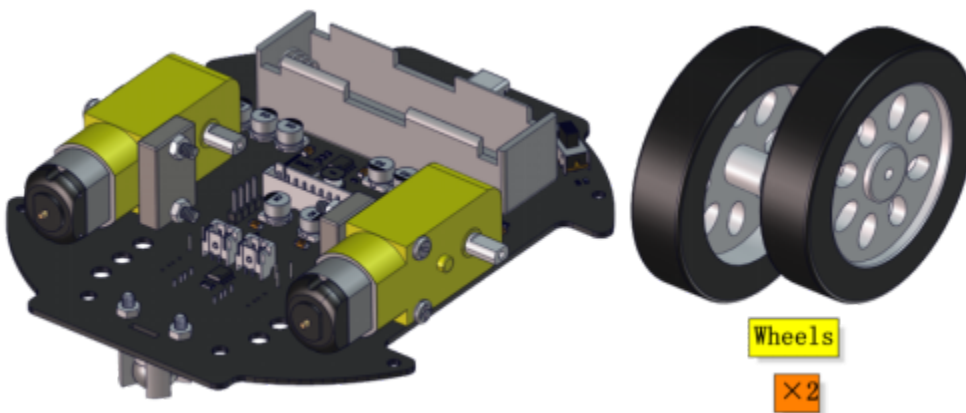
Step 2

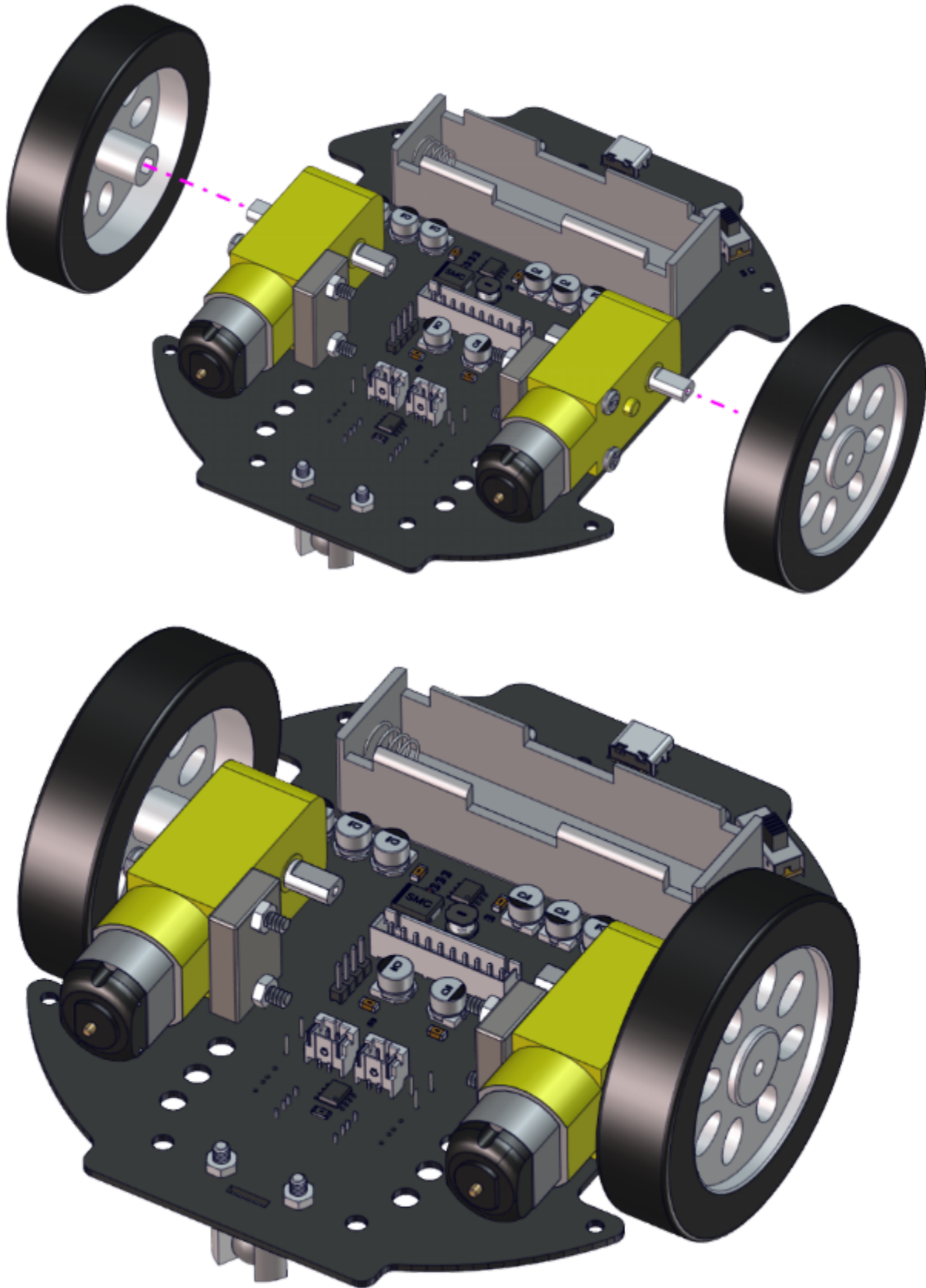




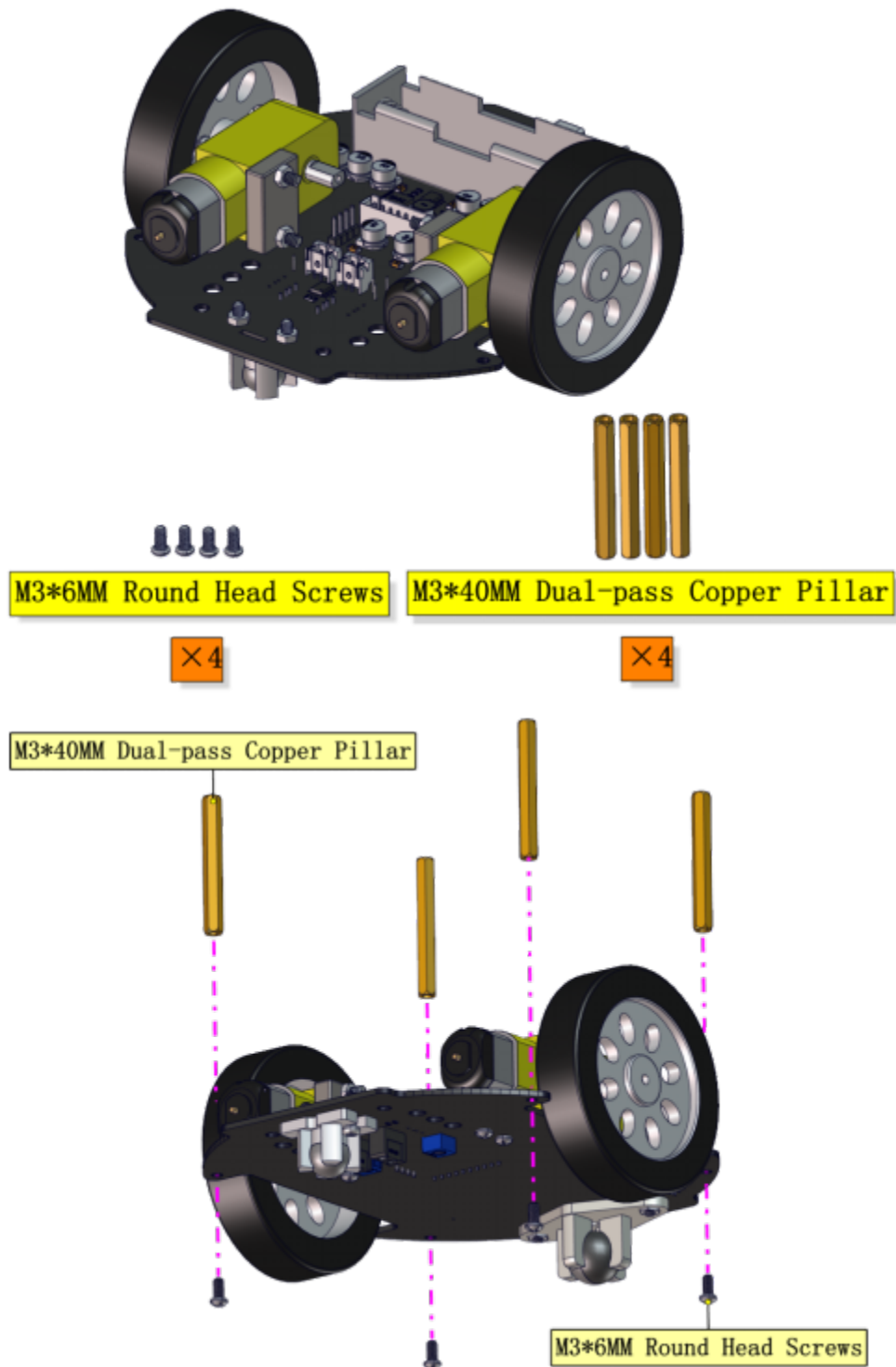


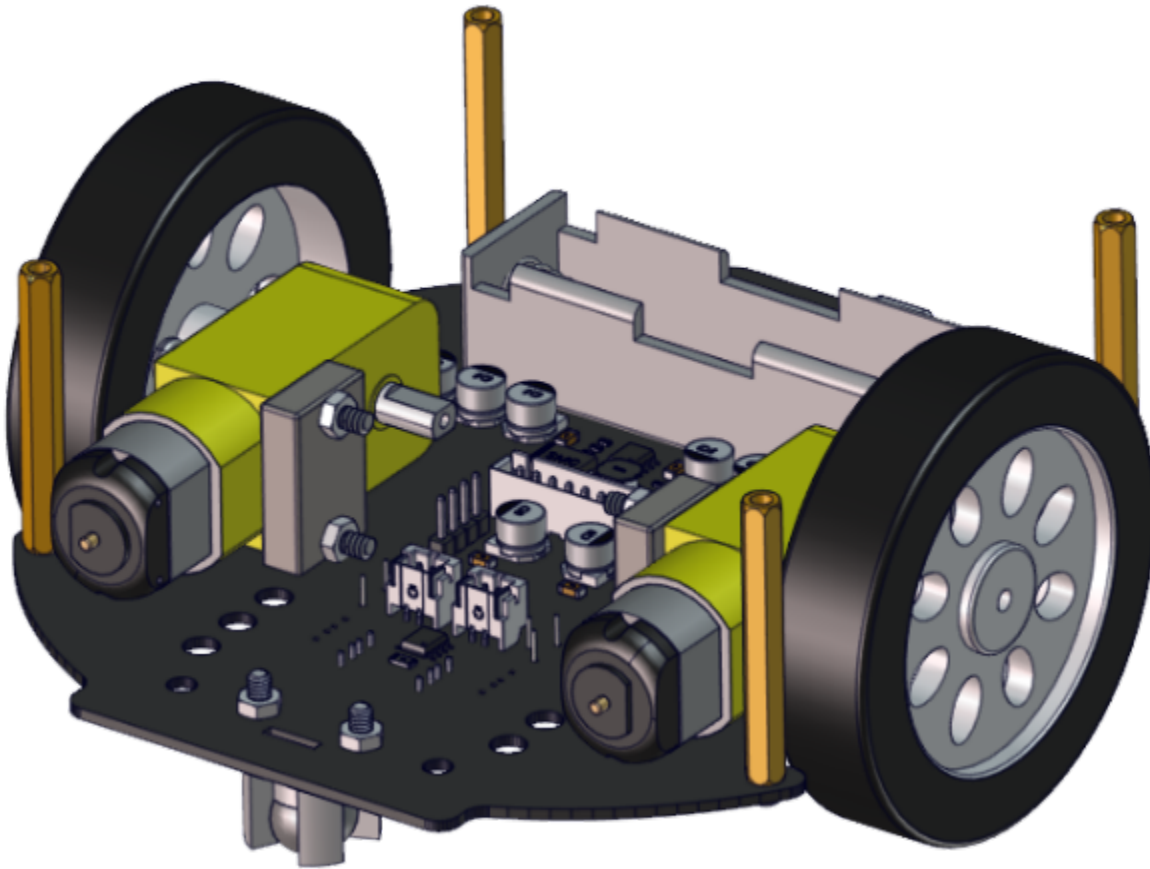
Step 3



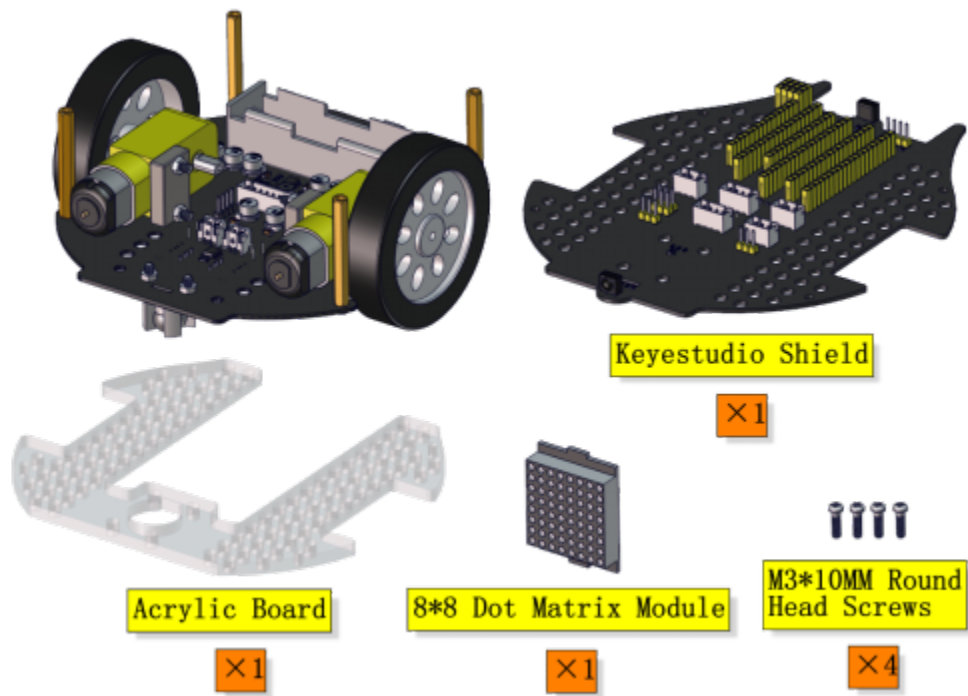


## Step 4

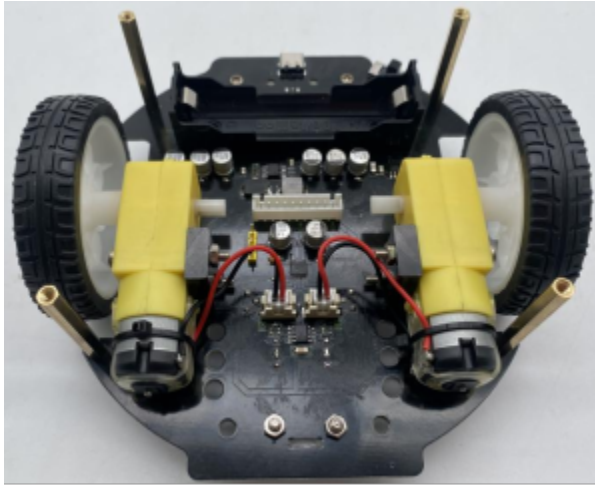




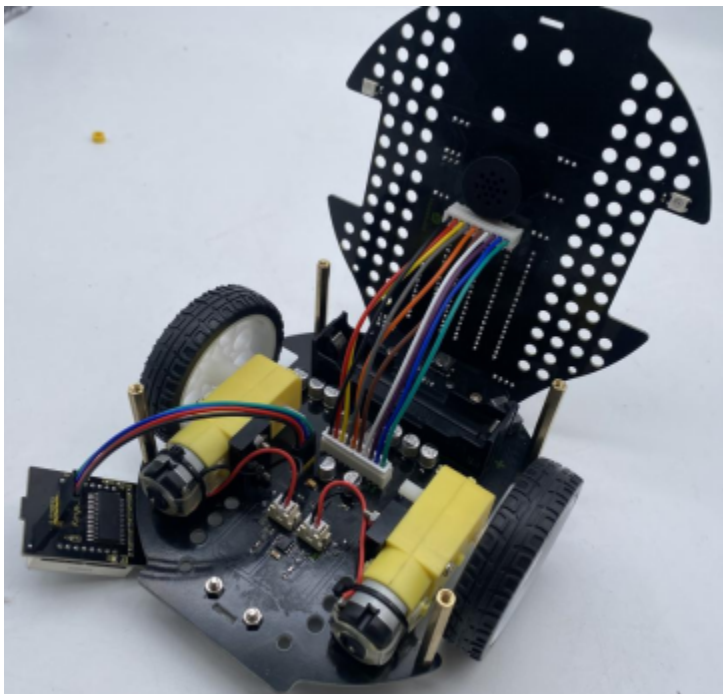
Step 5

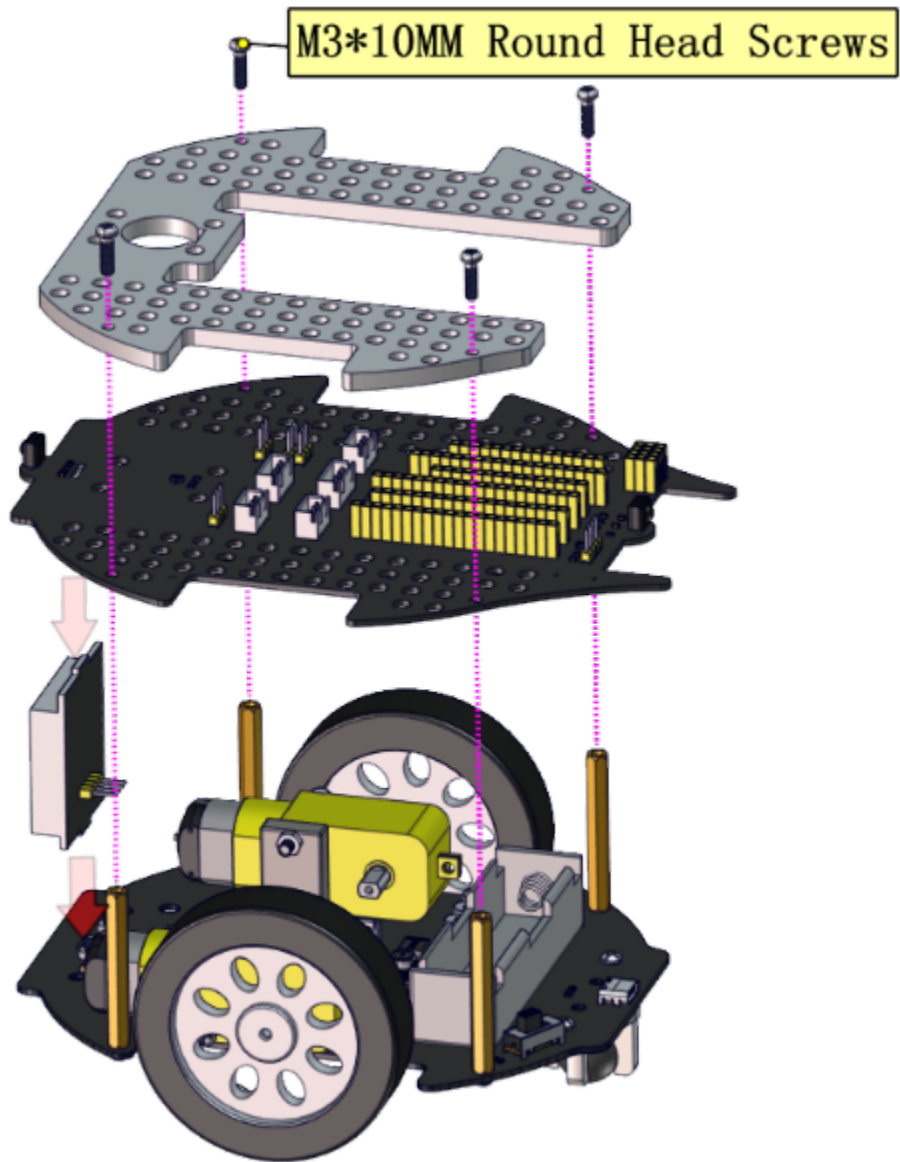


| Left Motor | Right Motor |
|------------|-------------|
| L          | R           |

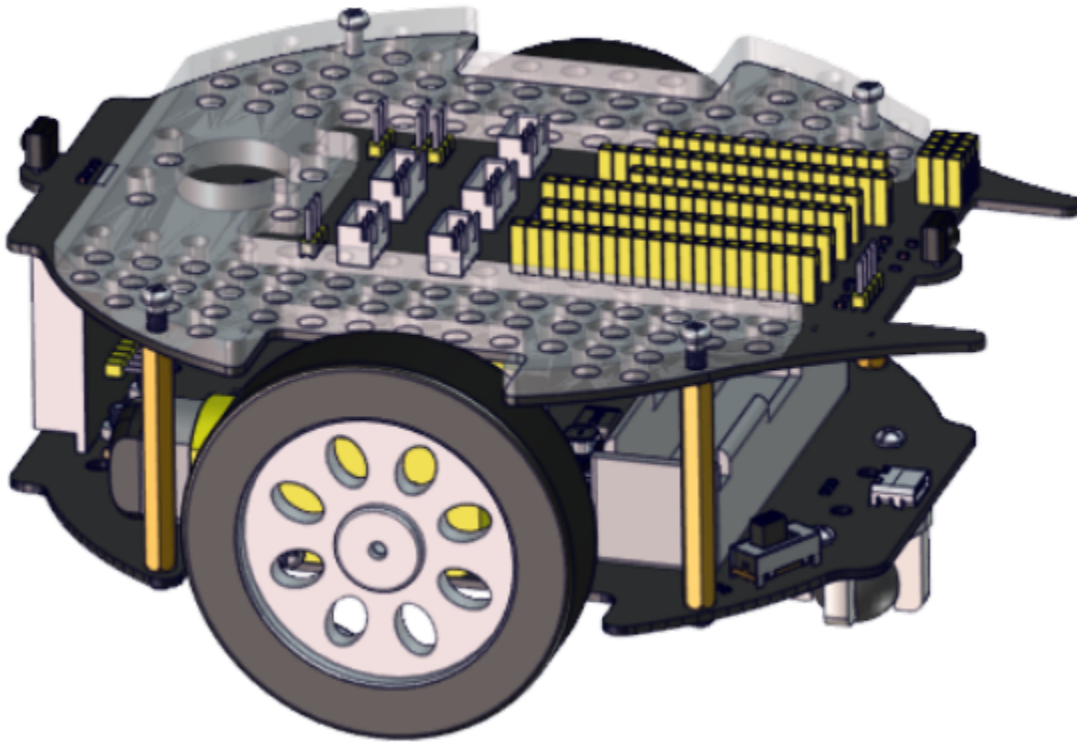


| 8*8 Dot Matrix Display | PCB |
|------------------------|-----|
| G                      | G   |
| 5V                     | 5V  |
| SDA                    | SDA |
| SCL                    | SCL |

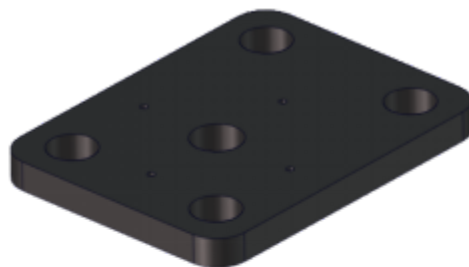








### Step 6



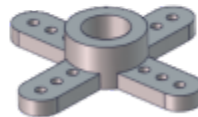
Acrylic Board

×1



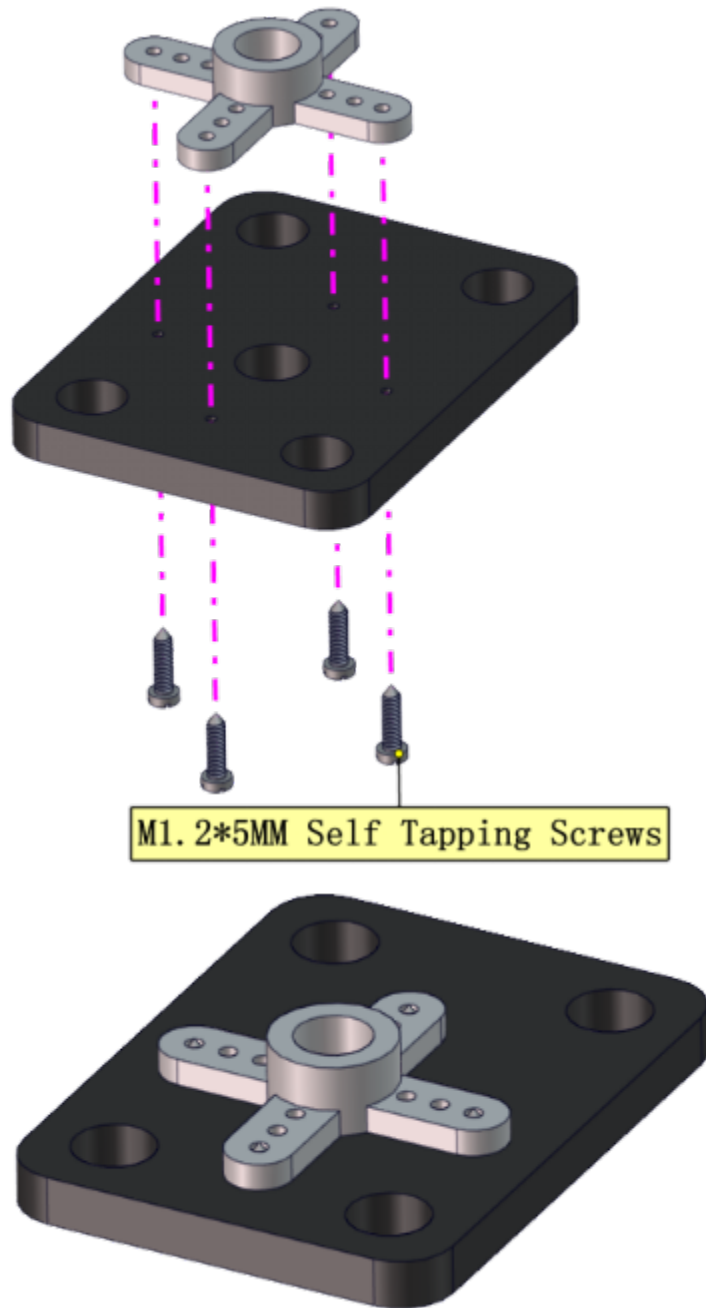
M1.2\*5MM Self Tapping Screws

×4



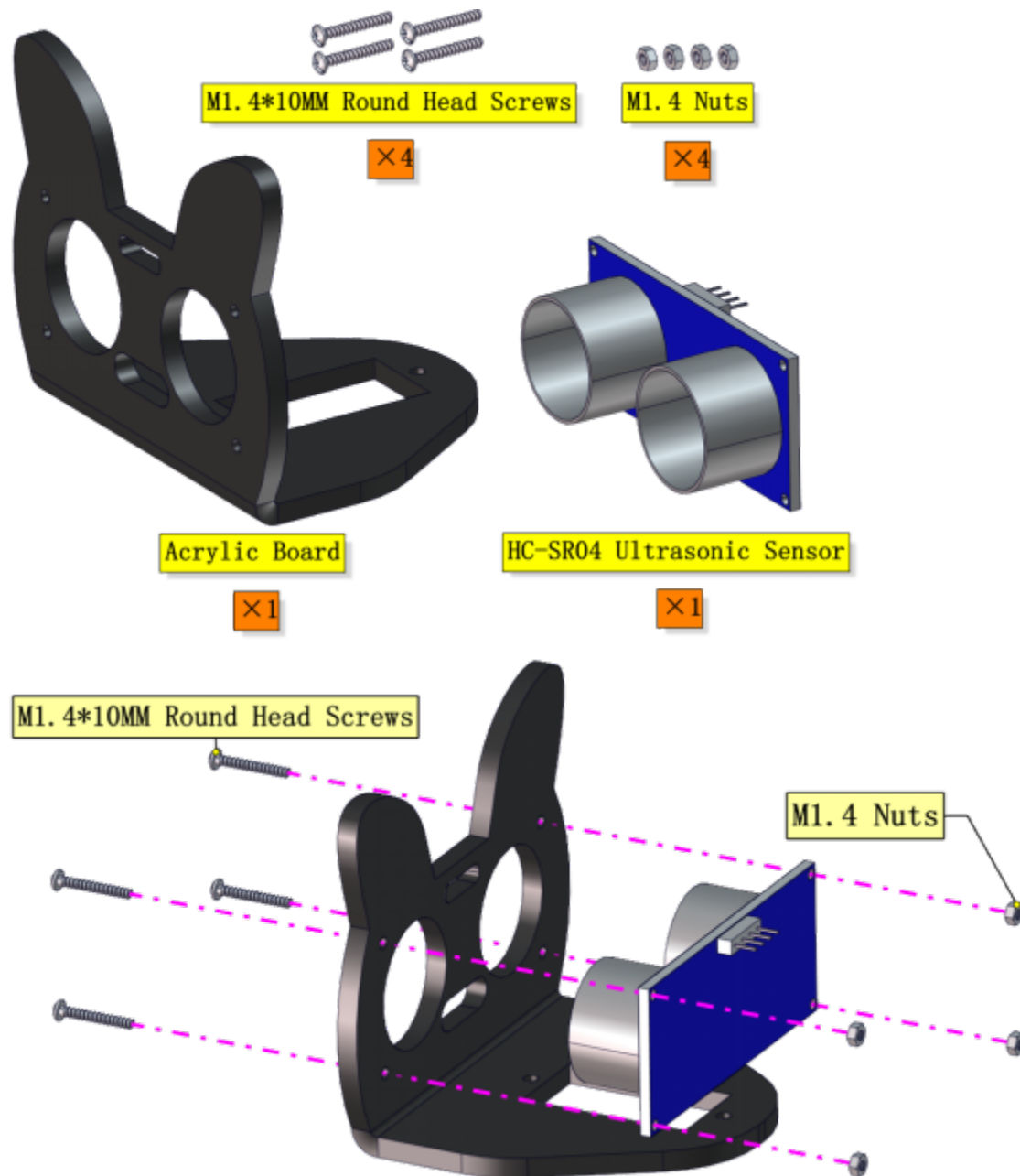
Control horn(belong to servo)

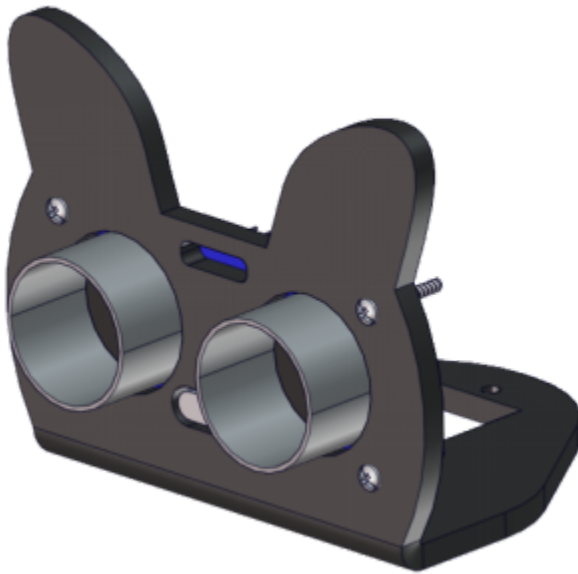
×1



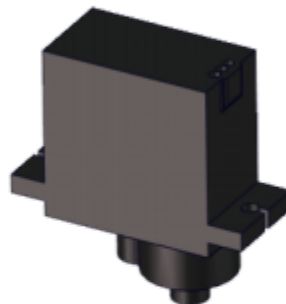
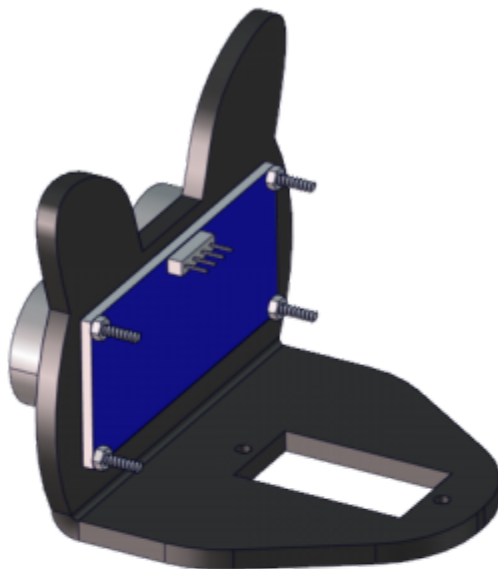
Step 7







Step 8



9G 180° Servo

×1



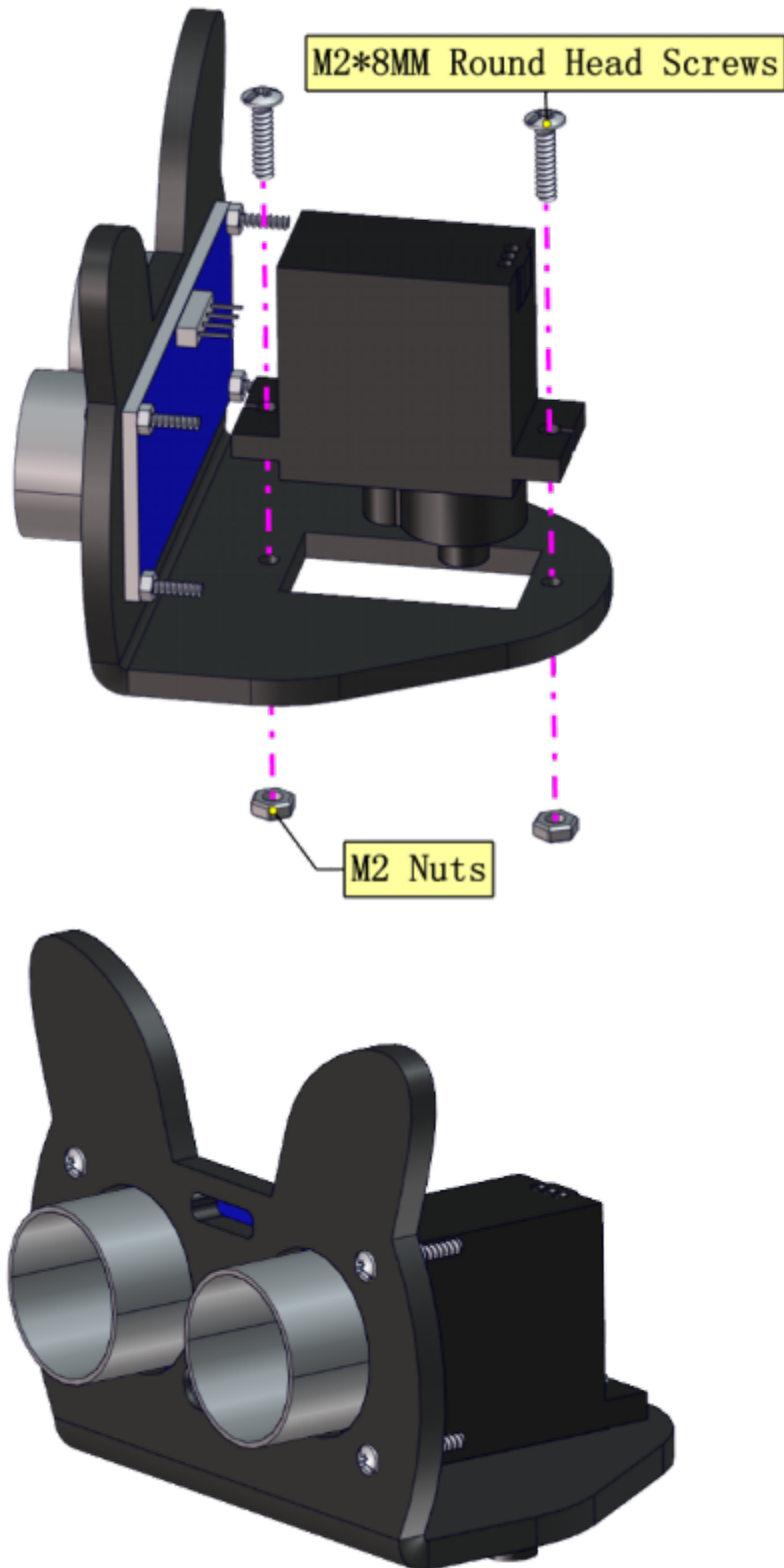
M2 Nuts

×2



M2\*8MM Round Head Screws

×2



Step 9



Adjust the angle of the servo to 90 degree.

| Servo       | PCB  |
|-------------|------|
| Brown line  | G    |
| Red line    | 5V   |
| Orange line | S1D9 |

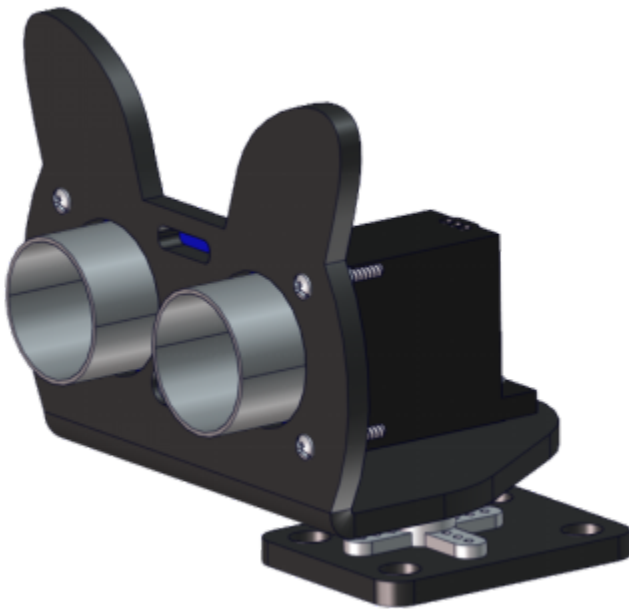
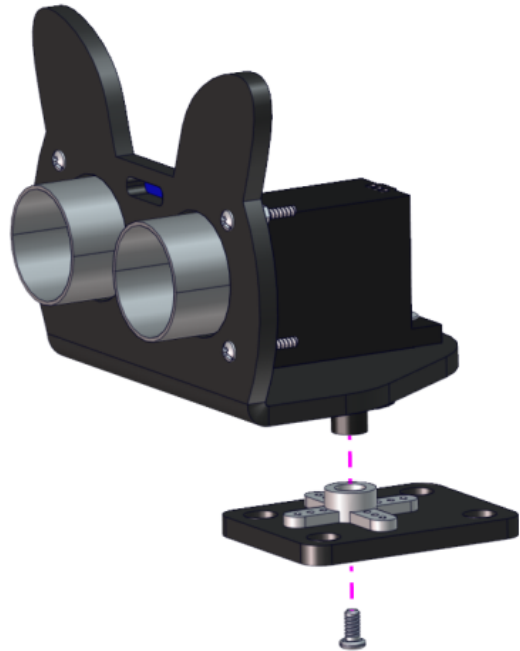
Copy the below code to copy into the Arduino IDE and upload it to the motherboard, or just open the code provided by us and upload it to the motherboard.

```
int servoPin = 9; //Define the pins of the steering gear
void setup() {
  pinMode(servoPin, OUTPUT); //steering pin is set to output
  servopulse(servoPin, 90); //Turn it to 90 degrees
  delay(300); //delay 0.3S
}
void loop(){
}
void servopulse(int pin, int myangle) { //Impulse function
  int pulsewidth = map(myangle, 0, 180, 500, 2500); //Map Angle to pulse width
  for (int i = 0; i < 5; i++) { //Output a few more pulses
    digitalWrite(pin, HIGH); //Set the steering gear interface level to high
    delayMicroseconds(pulsewidth); //Number of microseconds of delayed pulse width value
    digitalWrite(pin, LOW); //Lower the level of steering gear interface
    delay(20 - pulsewidth / 1000);
  }
  digitalWrite(pin, LOW); //Lower the level of steering gear interface
  delay(20 - pulsewidth / 1000);
}
```

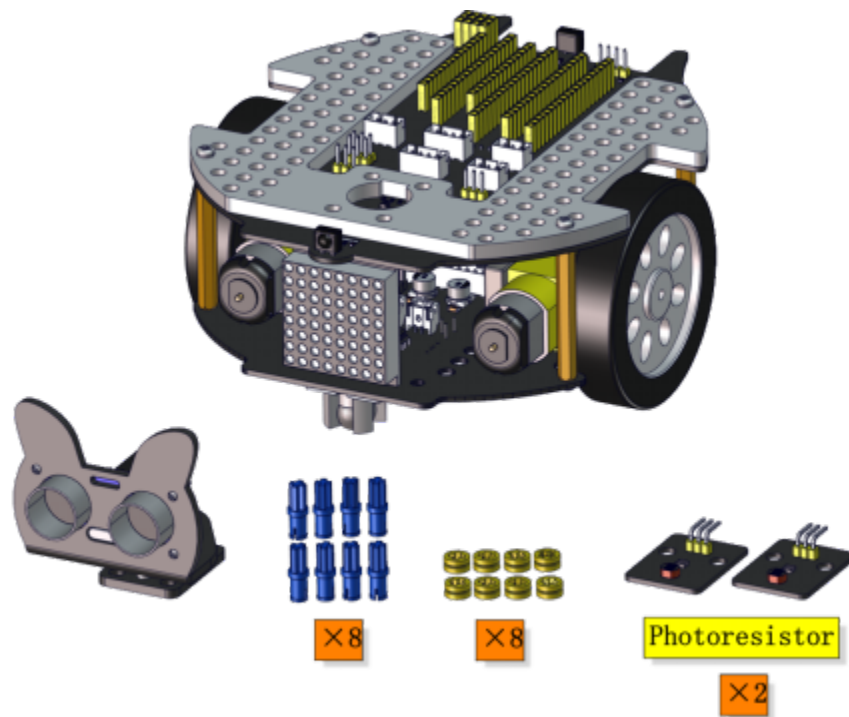
Keep the ultrasonic sensor parallel to the board

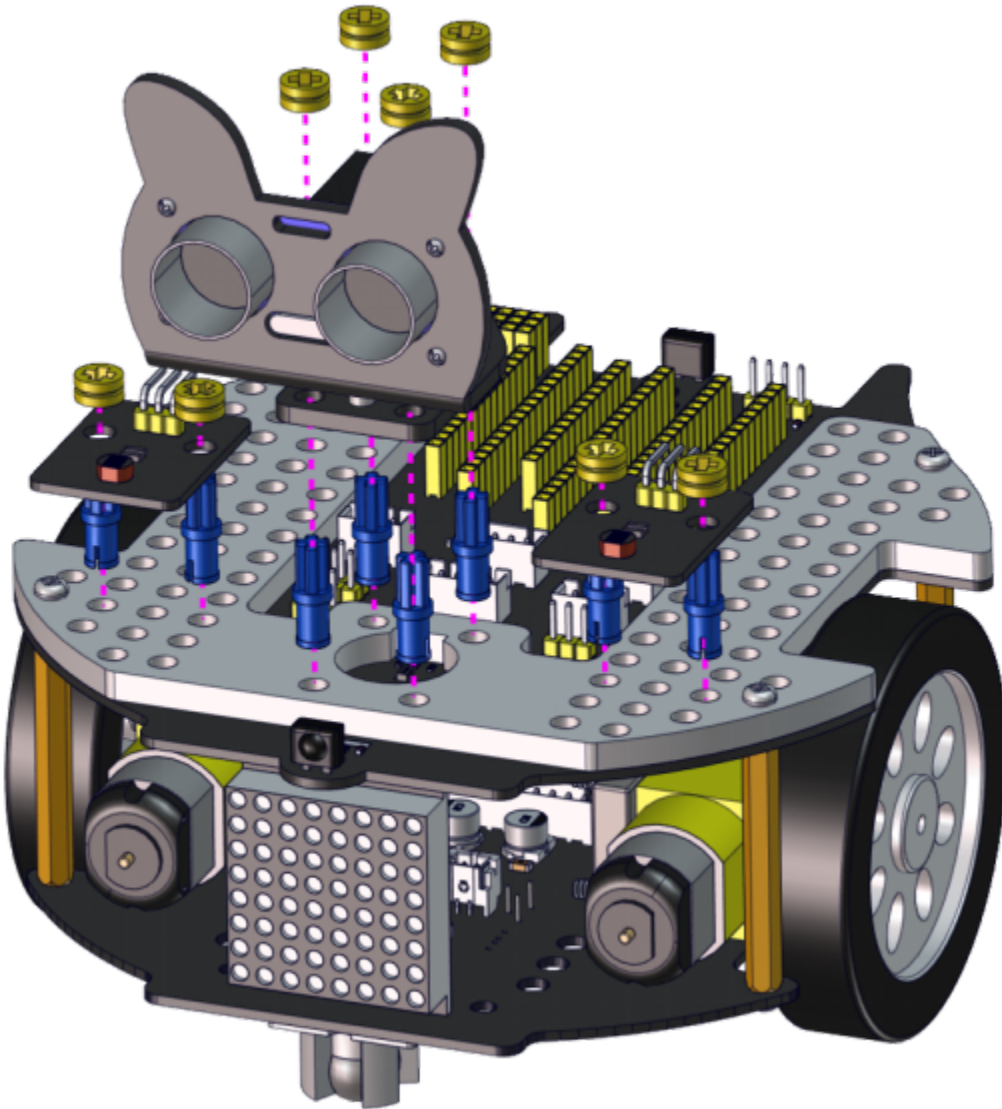


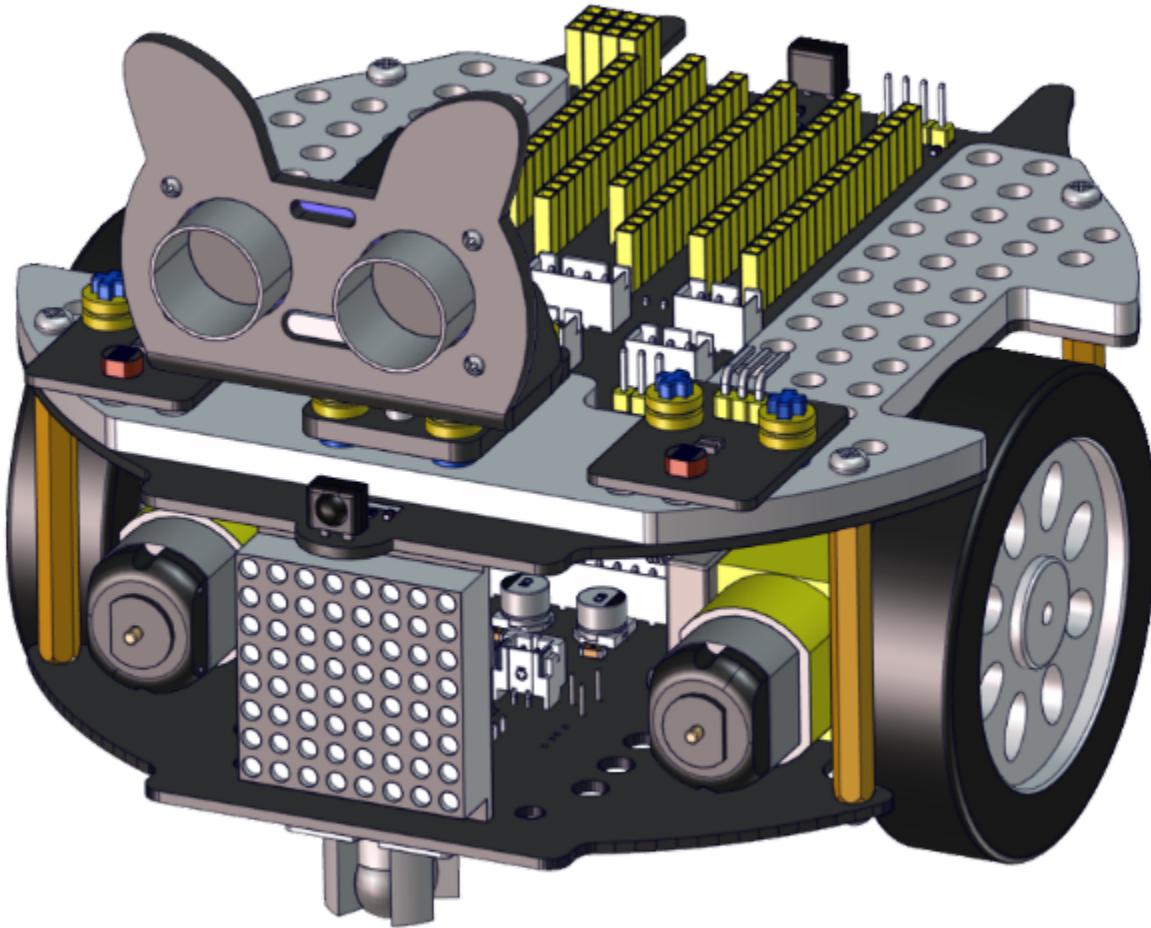
Note: Before assembling the servo, you need to adjust it to 90°, otherwise it will not work properly for the robot.



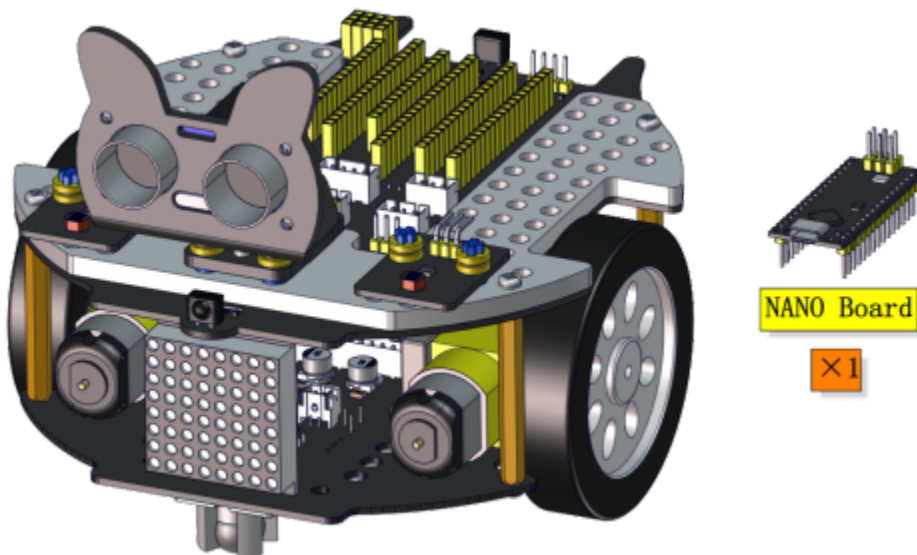
Step 10



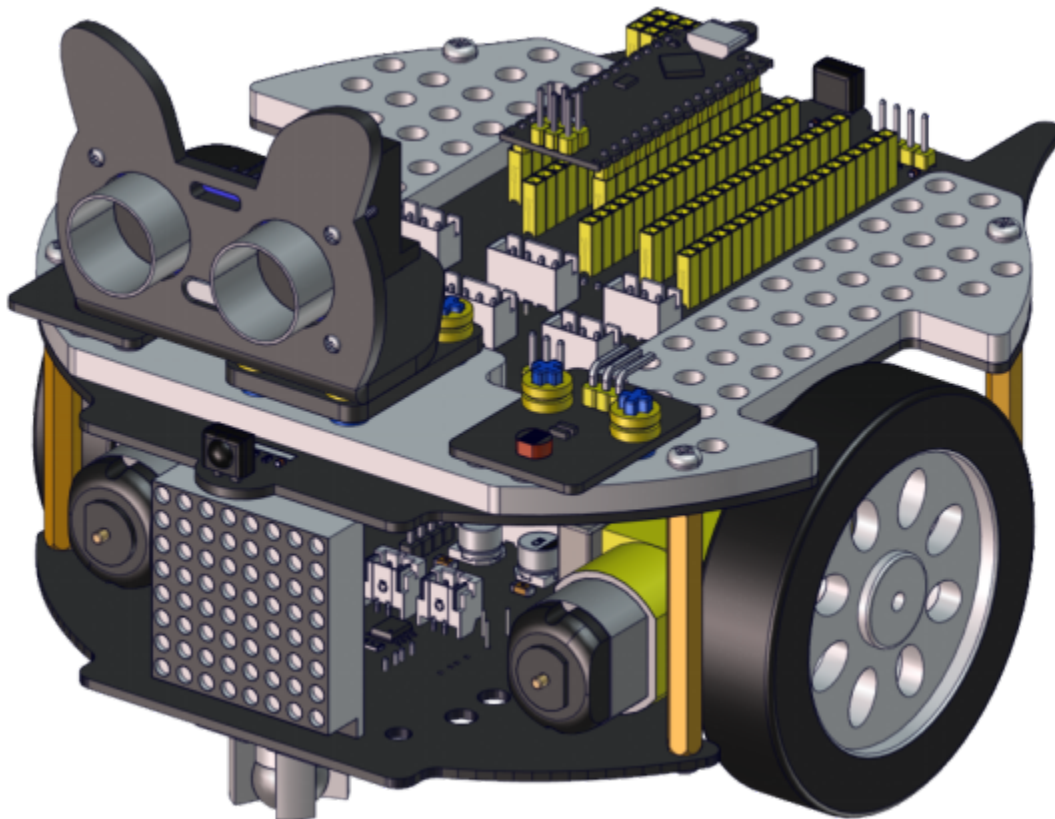
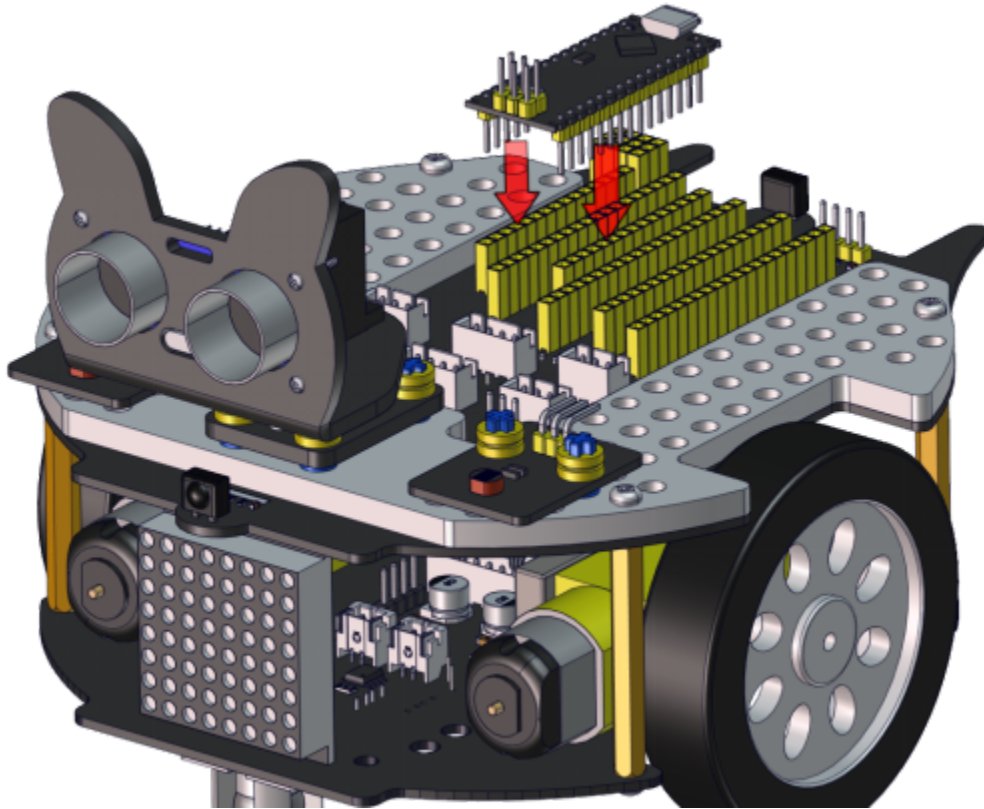




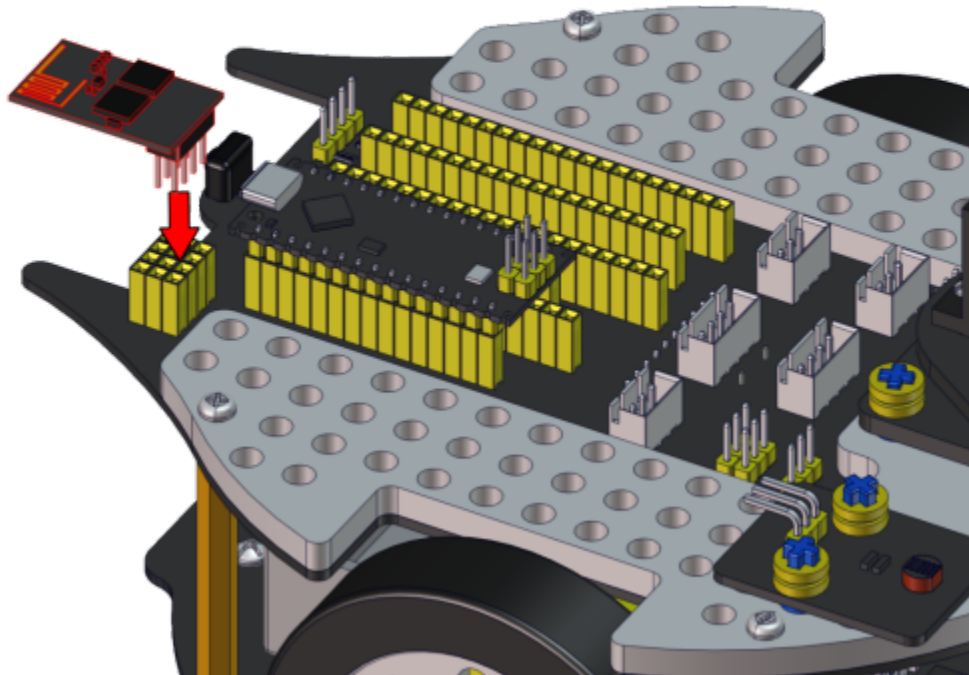
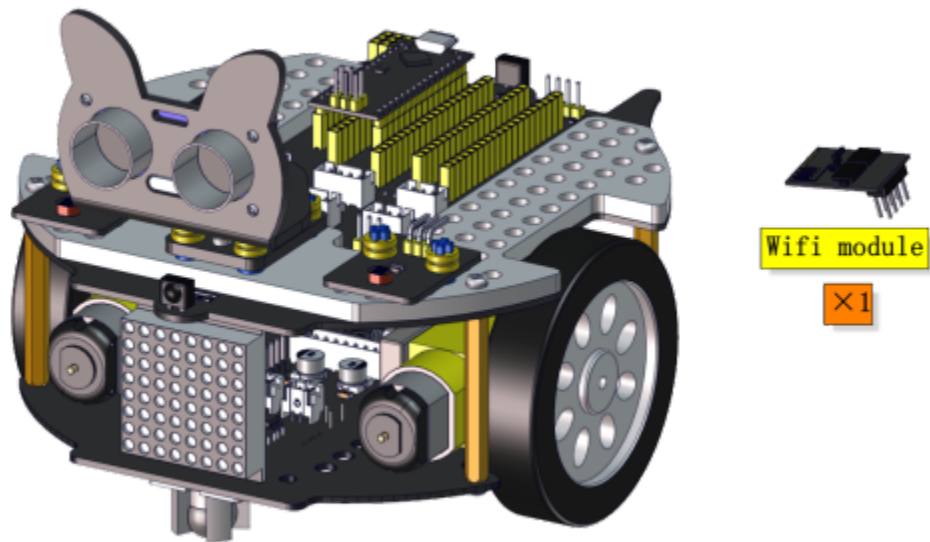
Step 11

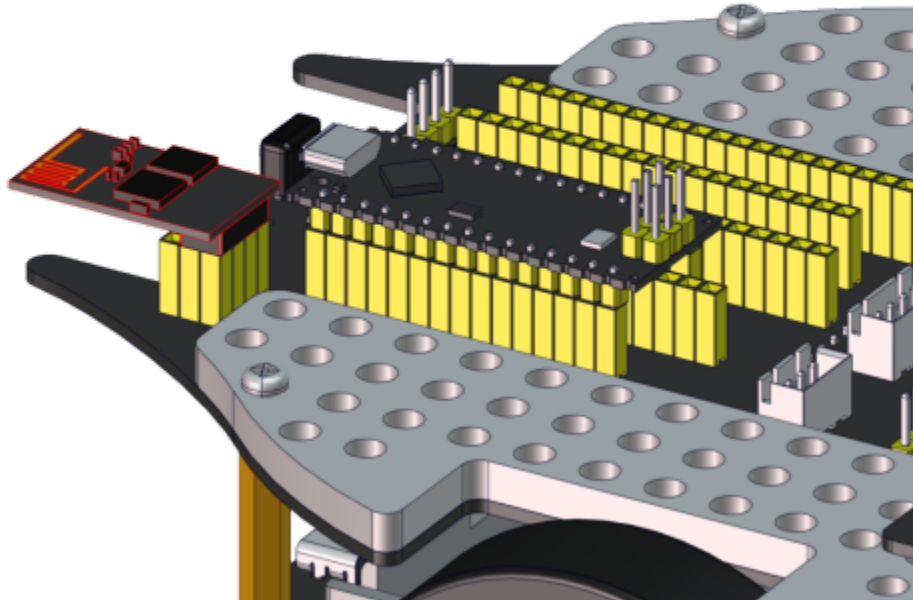


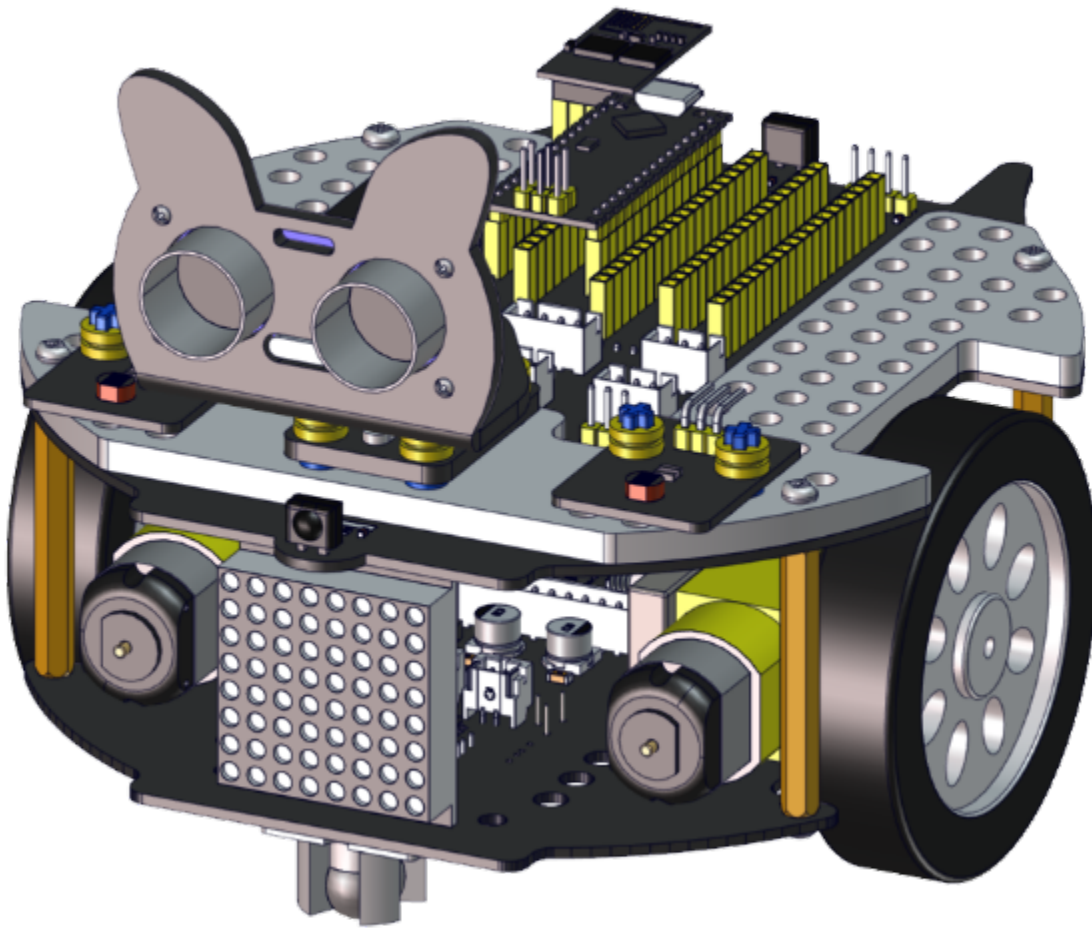




Step 12

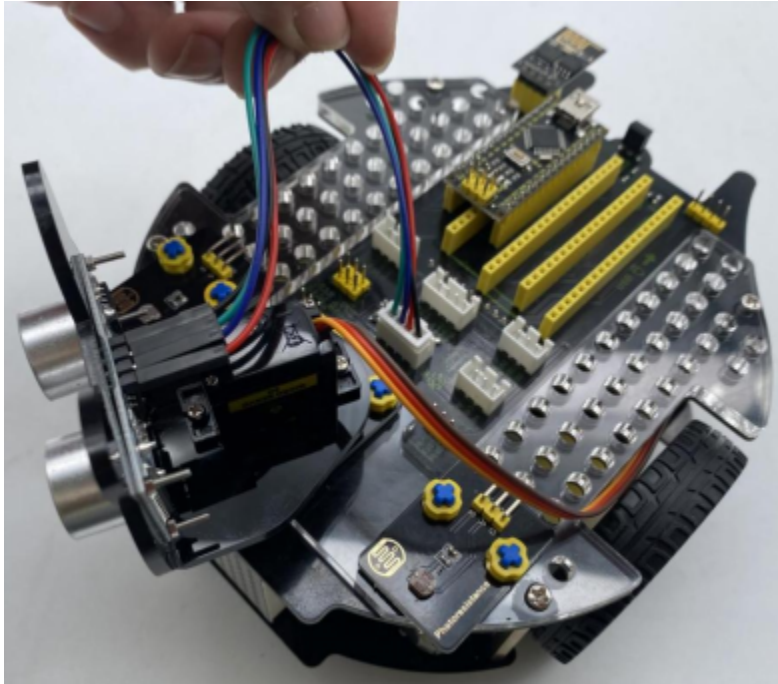






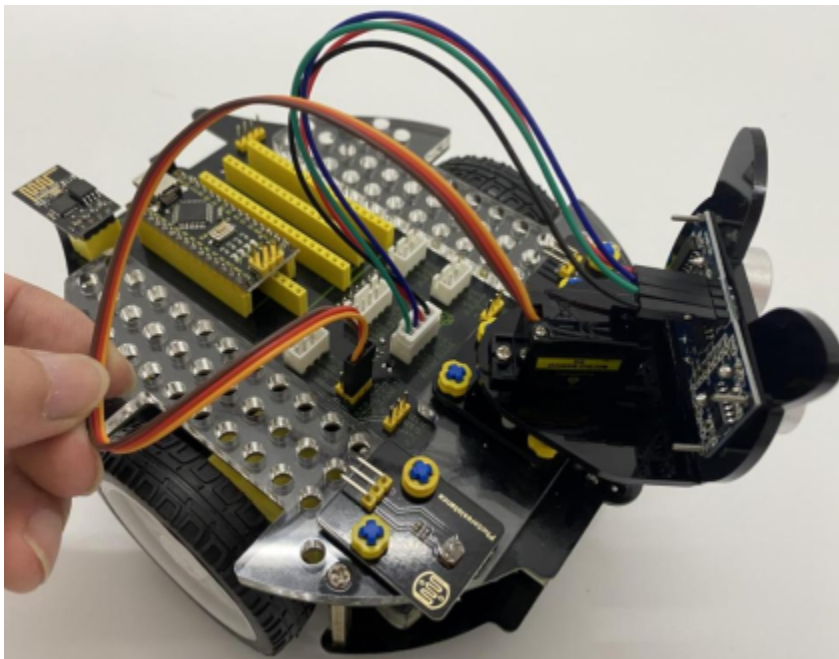
**Wire up the ultrasonic sensor**

| Ultrasonic Sensor | PCB  |
|-------------------|------|
| Vcc               | 5V   |
| Trig              | S2D8 |
| Echo              | S1D7 |
| Gnd               | G    |



Wire up the servo

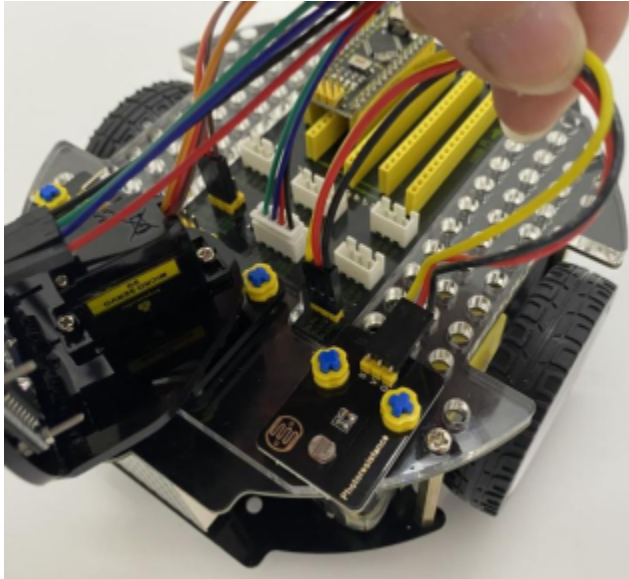
| Servo       | PCB  |
|-------------|------|
| Brown line  | G    |
| Red line    | 5V   |
| Orange line | S1D9 |



Wire up the left photoresistor

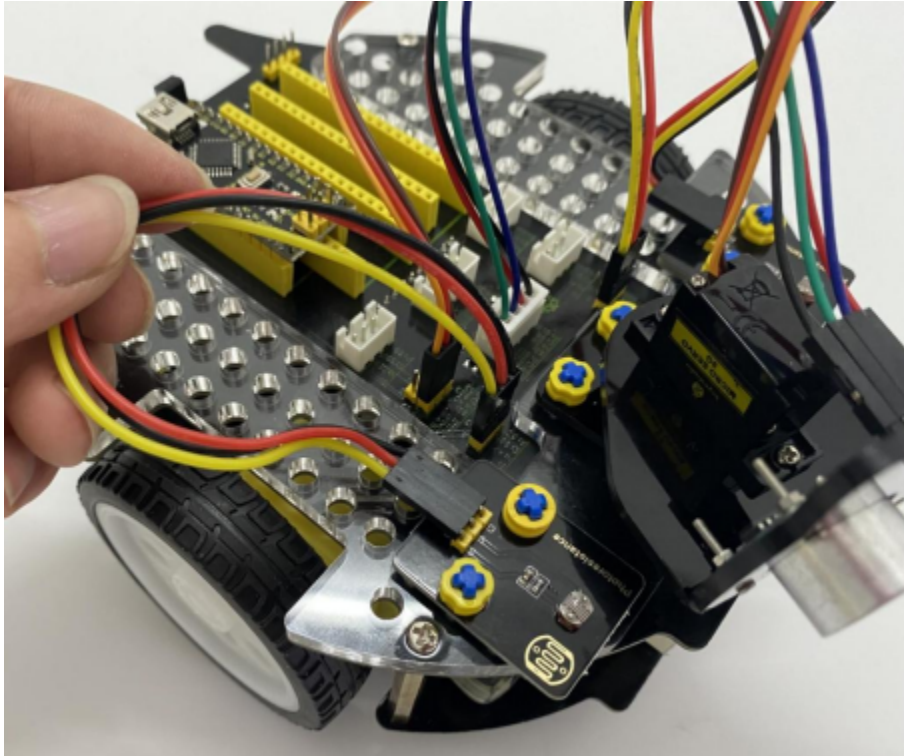


| Left photoresistor | PCB |
|--------------------|-----|
| G                  | G   |
| V                  | V   |
| S                  | SA6 |

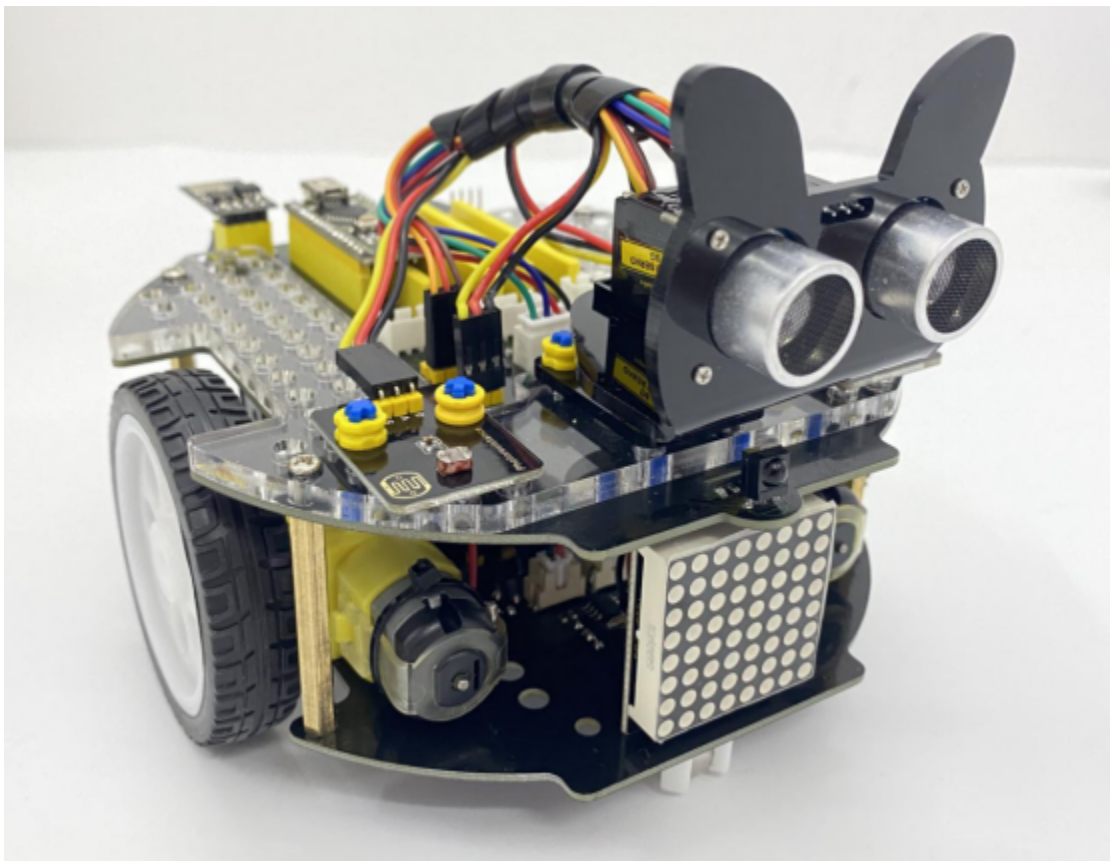


**Wire up the right photoresistor**

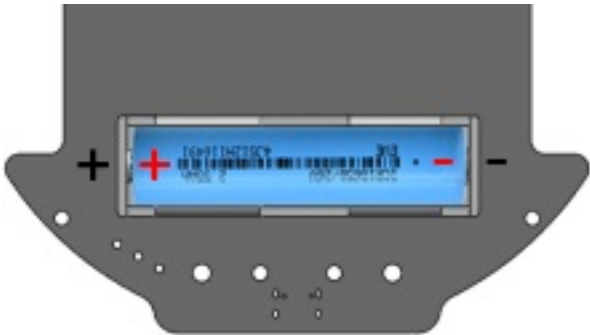
| Right photoresistor | PCB |
|---------------------|-----|
| G                   | G   |
| V                   | V   |
| S                   | SA7 |



Beetle Robot Car



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



### 8.3 3. Projects

#### 8.3.1 Project 1: LED Blinking

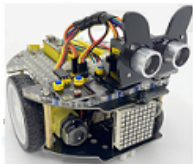



##### (1)Description

There is an onboard LED (L) on our Arduino Nano board connected to D13. In this experiment, we will make this LED blink.

LED blinking is the most basic experimental project for Arduino enthusiasts.

Let’s get started.

##### (2)Components Required

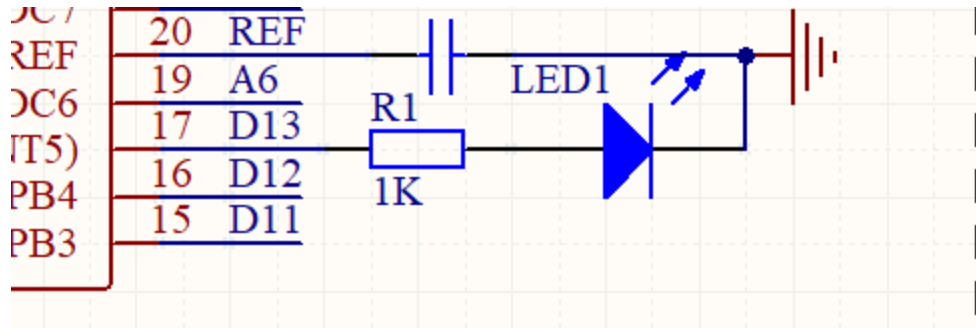
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

##### (3)Knowledge

On-board LED

LED, the abbreviation of light emitting diodes, consists of Ga, As, P, N chemical compounds and so on. It is easy to control through the IO port(D13) of the Arduino Nano board.





#### (4)Test Code

```

/*
  Project 01 LED Blinking
  Turns an LED on for one second, then off for one second, repeatedly.
*/
int ledPin=13; //Define LED pin to D13
// the setup function runs once when you press reset or power the board

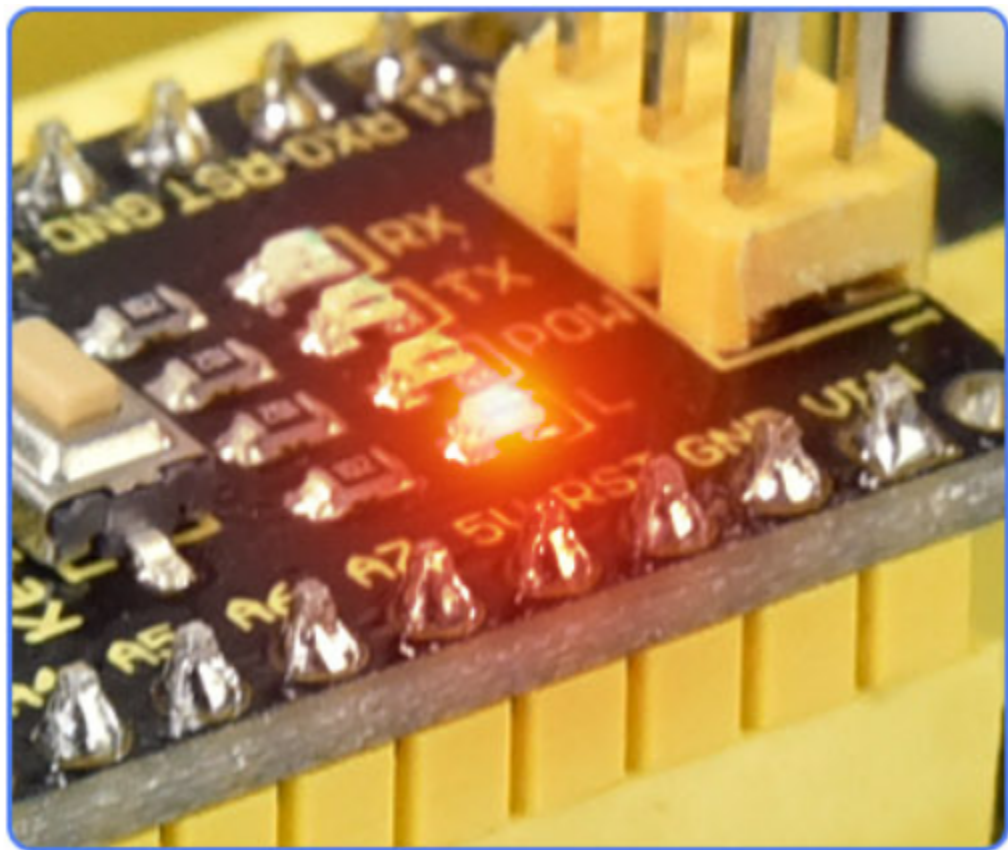
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(ledPin, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(ledPin, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                 // wait for a second
  digitalWrite(ledPin, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                 // wait for a second
}

```

#### (5)Test Result

Upload the test code to the Arduino Nano board and power up with a USB cable. Then the on-board LED will flash.



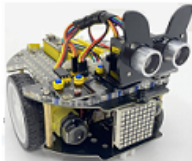



8.3.2 Project 2: 6812 RGB

(1)Description

There are 4 RGB LEDs can be widely used in the decoration of buildings, bridges, roads, gardens, courtyards and so on by colors adjustment.

In this experiment, we will demonstrate different lighting effects with them.

(2)Components Required

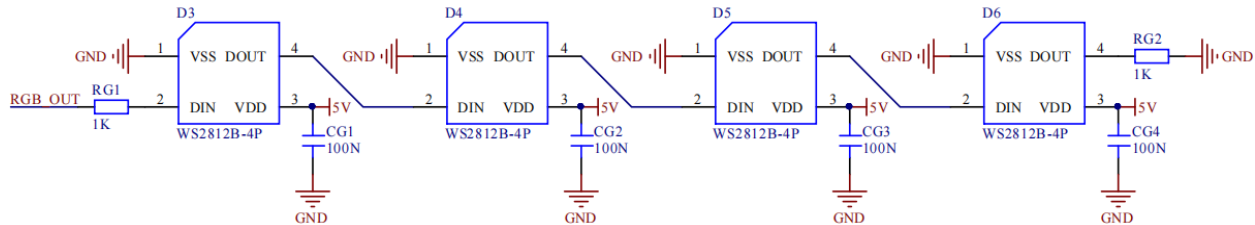
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

### (3)Knowledge

#### Working Principle

From the schematic diagram, we can see that these four pixel lighting beads are all connected in series. In fact, no matter how many they are, we can use a pin to control a light and let it display any color. The pixel point contains a data latch signal shaping amplifier drive circuit, a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.

The data protocol adopts a single-wire zero-code communication method. After the pixel is powered up and reset, the S terminal receives the data transmitted from the controller. The first 24bit data sent is extracted by the first pixel and sent to the data latch of the pixel.



### (4)Test Code

The SK6812RGB on the PCB board is controlled by the IO port (A3).

```

/*
  Project 02 SK6812 RGB
  4 RGBs for various lighting effects.
*/
#include <Adafruit_NeoPixel.h>

#define PIN A3

// Parameter 1 = number of pixels in strip
// Parameter 2 = Arduino pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel strip = Adafruit_NeoPixel(60, PIN, NEO_GRB + NEO_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across
// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input
// and minimize distance between Arduino and first pixel. Avoid connecting
// on a live circuit...if you must, connect GND first.

void setup() {
  strip.begin();
  strip.show(); // Initialize all pixels to 'off'
}

void loop() {
  // Some example procedures showing how to display to the pixels:

```

(continues on next page)

(continued from previous page)

```

colorWipe(strip.Color(255, 0, 0), 50); // Red
colorWipe(strip.Color(0, 255, 0), 50); // Green
colorWipe(strip.Color(0, 0, 255), 50); // Blue
// Send a theater pixel chase in...
theaterChase(strip.Color(127, 127, 127), 50); // White
theaterChase(strip.Color(127, 0, 0), 50); // Red
theaterChase(strip.Color(0, 0, 127), 50); // Blue

rainbow(20);
rainbowCycle(20);
theaterChaseRainbow(50);
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for(uint16_t i=0; i<strip.numPixels(); i++) {
    strip.setPixelColor(i, c);
    strip.show();
    delay(wait);
  }
}

void rainbow(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256; j++) {
    for(i=0; i<strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel((i+j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

// Slightly different, this makes the rainbow equally distributed throughout
void rainbowCycle(uint8_t wait) {
  uint16_t i, j;

  for(j=0; j<256*5; j++) { // 5 cycles of all colors on wheel
    for(i=0; i< strip.numPixels(); i++) {
      strip.setPixelColor(i, Wheel(((i * 256 / strip.numPixels()) + j) & 255));
    }
    strip.show();
    delay(wait);
  }
}

//Theatre-style crawling lights.
void theaterChase(uint32_t c, uint8_t wait) {
  for (int j=0; j<10; j++) { //do 10 cycles of chasing
    for (int q=0; q < 3; q++) {
      for (int i=0; i < strip.numPixels(); i=i+3) {

```

(continues on next page)

(continued from previous page)

```

    strip.setPixelColor(i+q, c);    //turn every third pixel on
}
strip.show();

delay(wait);

for (int i=0; i < strip.numPixels(); i=i+3) {
    strip.setPixelColor(i+q, 0);    //turn every third pixel off
}
}
}

//Theatre-style crawling lights with rainbow effect
void theaterChaseRainbow(uint8_t wait) {
    for (int j=0; j < 256; j++) {    // cycle all 256 colors in the wheel
        for (int q=0; q < 3; q++) {
            for (int i=0; i < strip.numPixels(); i=i+3) {
                strip.setPixelColor(i+q, Wheel( (i+j) % 255));    //turn every third pixel on
            }
            strip.show();

            delay(wait);

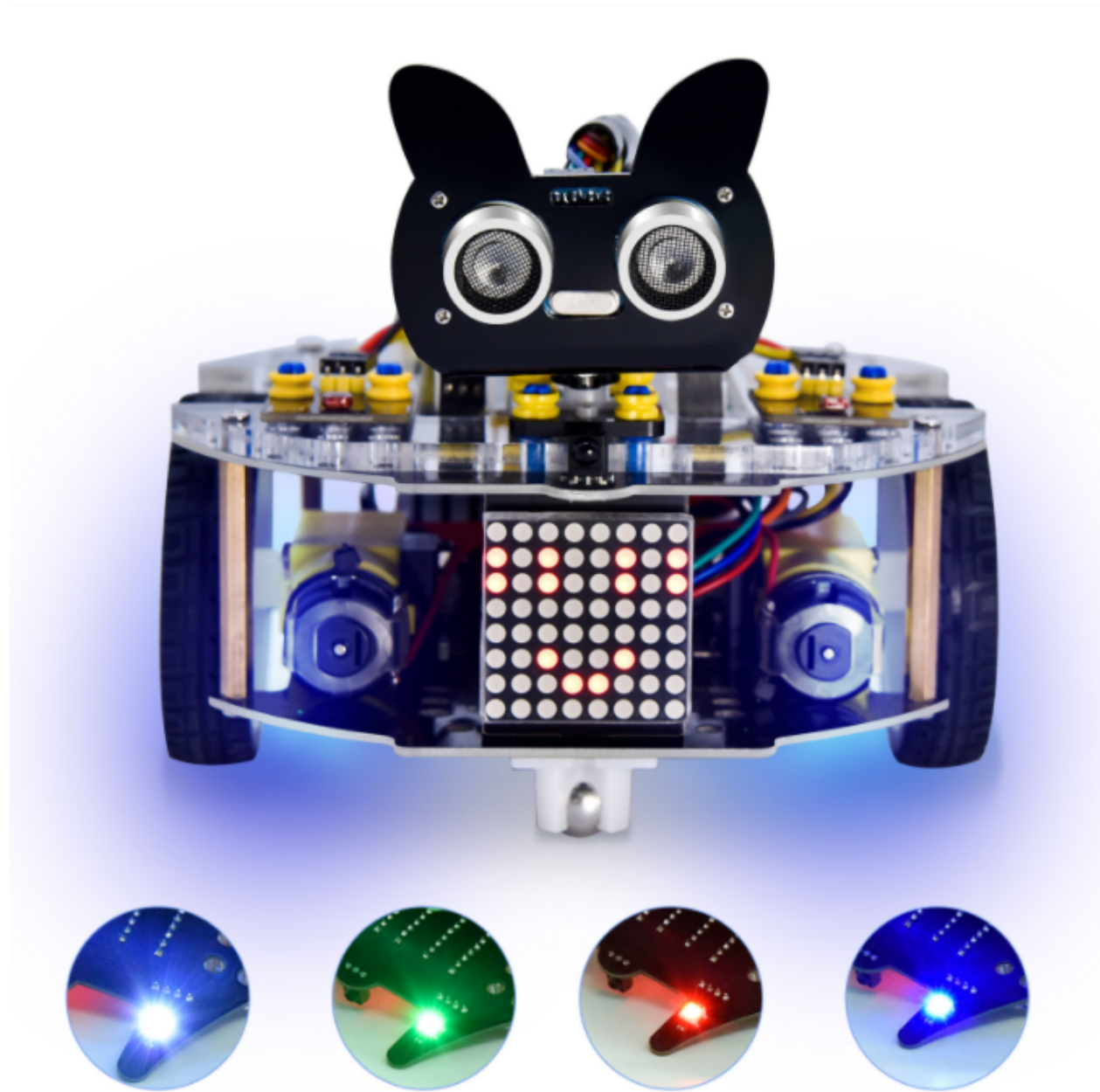
            for (int i=0; i < strip.numPixels(); i=i+3) {
                strip.setPixelColor(i+q, 0);    //turn every third pixel off
            }
        }
    }
}

// Input a value 0 to 255 to get a color value.
// The colours are a transition r - g - b - back to r.
uint32_t Wheel(byte WheelPos) {
    if(WheelPos < 85) {
        return strip.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
    } else if(WheelPos < 170) {
        WheelPos -= 85;
        return strip.Color(255 - WheelPos * 3, 0, WheelPos * 3);
    } else {
        WheelPos -= 170;
        return strip.Color(0, WheelPos * 3, 255 - WheelPos * 3);
    }
}

```

### (5)Test Result

Upload the test code to the Arduino Nano board and power up by a USB cable. Then the four RGB lights on the PCB demonstrate multi-color light effect.



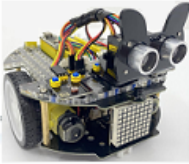



### 8.3.3 Project 3: Play Music

#### (1)Description

There is a power amplifier component on the expansion board, which is often used to play music and serve as an external amplifying device for music playback devices.

In this experiment, we use the speaker amplifier component to play music.

#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

#### (3)Knowledge

Power amplifier modules(equivalent to a passive buzzer) don't have internal oscillation circuits.

The power amplifier module can chime sounds with different frequency when power it up.

#### (4)Test Code

The speaker component on the PCB board is controlled by the D3 of the Arduino Nano board.

```

/*
Project 03 Buzzer
Buzzer plays music
*/
#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69

```

(continues on next page)



(continued from previous page)

```
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
```

(continues on next page)



(continued from previous page)

```

#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
#define REST 0
int tempo=114; // change this to make the song slower or faster
int buzzer = 3; // initializes digital I/O PIN to control the buzzer
// notes of the melody followed by the duration
// a 4 means a quarter note, 8 an eighth, 16 sixteenth, so on
// !!negative numbers are used to represent dotted notes
// so -4 means a dotted quarter note, that is, a quarter plus an eighth
int melody[] = {
    NOTE_E4,4, NOTE_E4,4, NOTE_F4,4, NOTE_G4,4, //1
    NOTE_G4,4, NOTE_F4,4, NOTE_E4,4, NOTE_D4,4,
    NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,
    NOTE_E4,-4, NOTE_D4,8, NOTE_D4,2,
    NOTE_E4,4, NOTE_E4,4, NOTE_F4,4, NOTE_G4,4, //4
    NOTE_G4,4, NOTE_F4,4, NOTE_E4,4, NOTE_D4,4,
    NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,
    NOTE_D4,-4, NOTE_C4,8, NOTE_C4,2,
    NOTE_D4,4, NOTE_D4,4, NOTE_E4,4, NOTE_C4,4, //8
    NOTE_D4,4, NOTE_E4,8, NOTE_F4,8, NOTE_E4,4, NOTE_C4,4,
    NOTE_D4,4, NOTE_E4,8, NOTE_F4,8, NOTE_E4,4, NOTE_D4,4,
    NOTE_C4,4, NOTE_D4,4, NOTE_G3,2,
    NOTE_E4,4, NOTE_E4,4, NOTE_F4,4, NOTE_G4,4, //12
    NOTE_G4,4, NOTE_F4,4, NOTE_E4,4, NOTE_D4,4,
    NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,
    NOTE_D4,-4, NOTE_C4,8, NOTE_C4,2
};
// sizeof gives the number of bytes, each int value is composed of two bytes (16 bits)
// there are two values per note (pitch and duration), so for each note there are four
↳ bytes
int notes=sizeof(melody)/sizeof(melody[0])/2;
// this calculates the duration of a whole note in ms (60s/tempo)*4 beats

```

(continues on next page)

(continued from previous page)

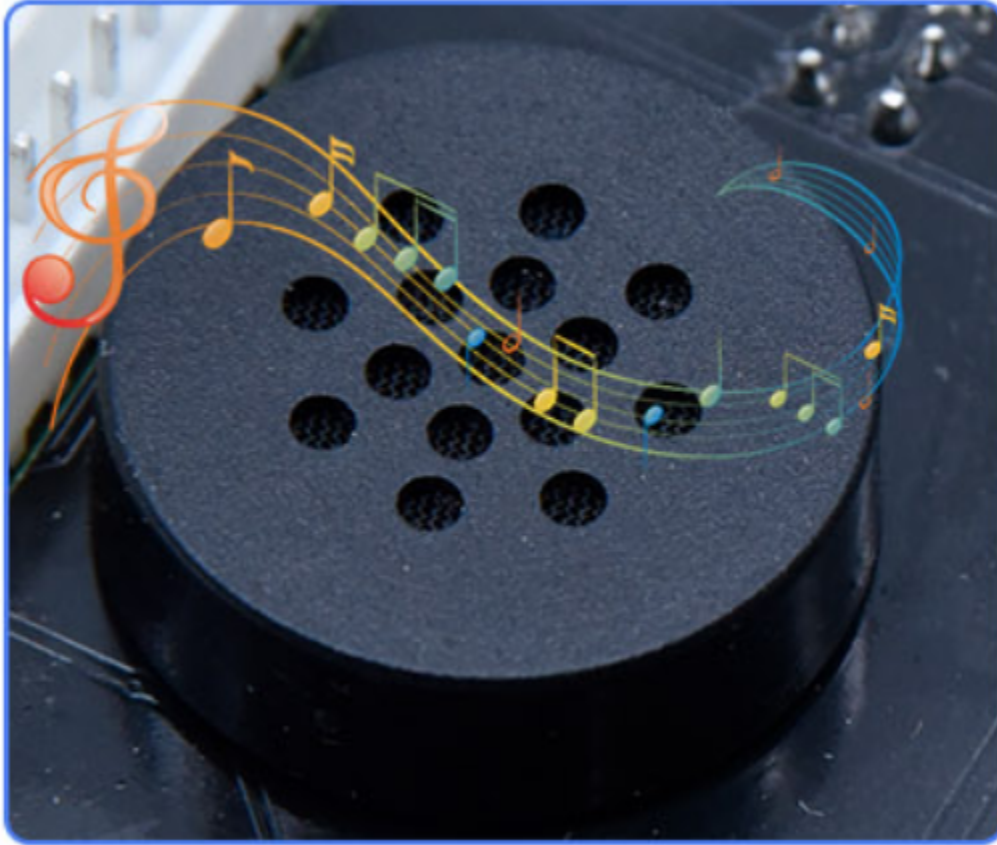
```

int wholenote = (60000 * 4) / tempo;
int divider = 0, noteDuration = 0;
void setup() {
  // iterate over the notes of the melody
  // remember, the array is twice the number of notes (notes + durations)
  for (int thisNote = 0; thisNote < notes * 2; thisNote = thisNote + 2) {
    // calculates the duration of each note
    divider = melody[thisNote + 1];
    if (divider > 0) {
      noteDuration = (wholenote) / divider; // regular note, just proceed
    } else if (divider < 0) {
      // dotted notes are represented with negative durations!!
      noteDuration = (wholenote) / abs(divider);
      noteDuration *= 1.5; // increases the duration in half for dotted notes
    }
    // we only play the note for 90% of the duration, leaving 10% as a pause
    tone(buzzer, melody[thisNote], noteDuration*0.9);
    // Wait for the specified duration before playing the next note
    delay(noteDuration);
    noTone(buzzer); // stop the waveform generation before the next note
  }
}
void loop() {
  // if you want to repeat the song forever,
  // just paste the setup code here instead.
}

```

### (5) Test Result

Upload the test code to the Arduino Nano board and power up with a USB cable. Then the power amplifier component will play music.



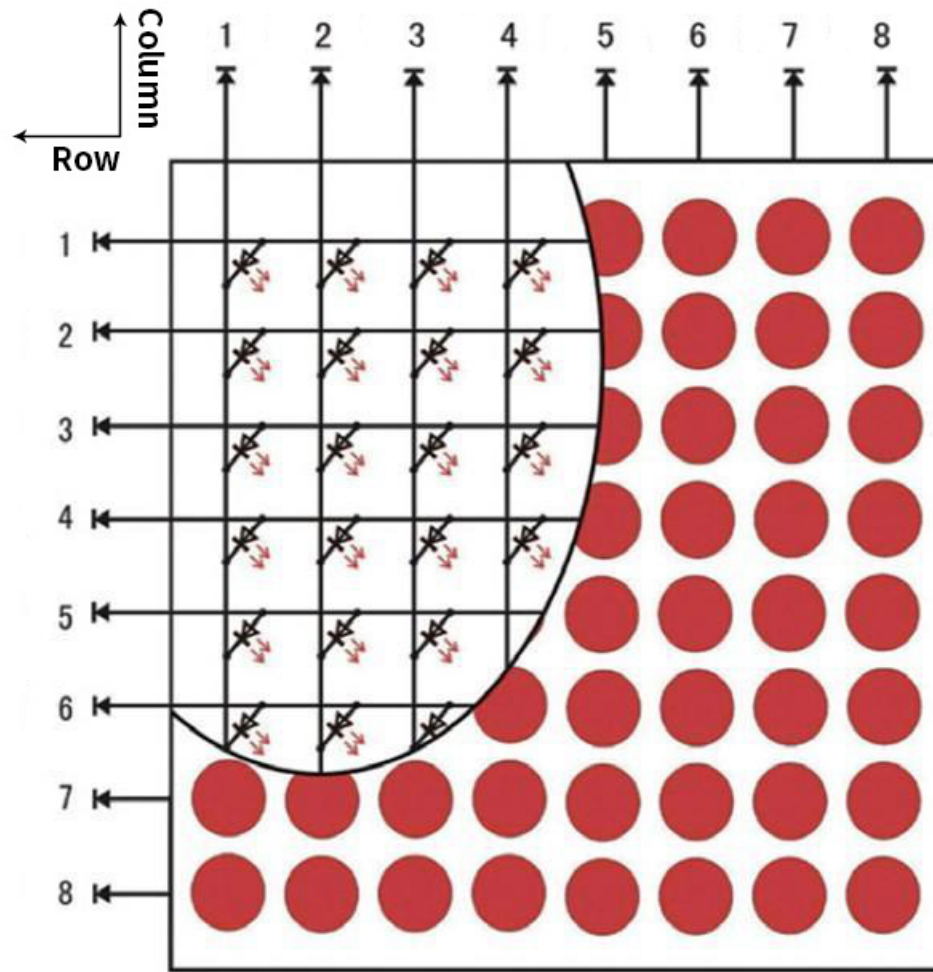
### 8.3.4 Project 4: 8\*8 Dot Matrix

#### (1)Description

Composed of LED emitting tube diodes, the 8\*8 LED dot matrix are applied widely to public information display like advertisement screen and bulletin board, by controlling LED to show words, pictures and videos, etc.

There are different types of matrices, including 4×4, 8×8 and 16×16 and etc. It contains 64 LEDs.

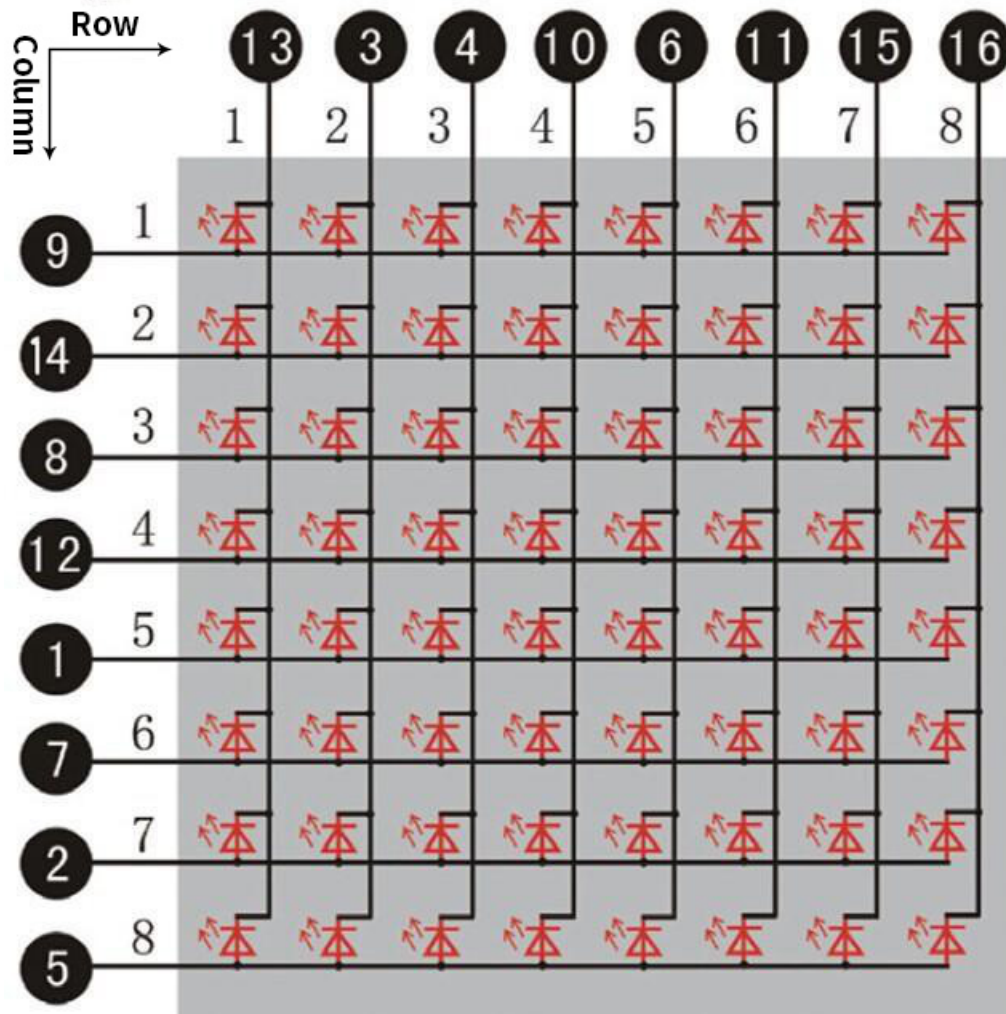
The inner structure of 8×8 dot matrix is shown below.



Every LED is installed on the cross point of row line and column line. When the voltage on a row line increases, and the voltage on the column line reduces, the LED on the cross point will light up. 8×8 dot matrix has 16 pins. Put the silk-screened side down and the numbers are 1, 8, 9 and 16 in anticlockwise order as marked below.

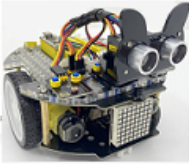





The definition inner pins are shown below:



For instance, to light up the LED on row 1 and column 1, you should increase the voltage of pin 9 and reduce the voltage of pin 13.

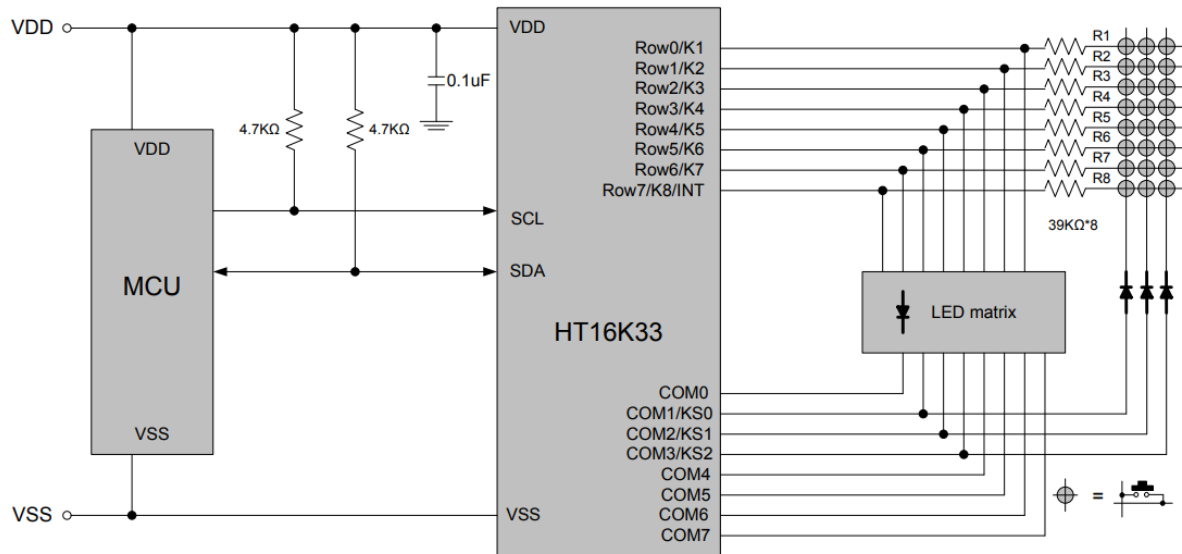
## (2) Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

### (3)HT16K33 8X8 Dot Matrix

The HT16K33 is a memory mapping and multi-purpose LED controller driver. The max. Display segment numbers in the device is 128 patterns (16 segments and 8 commons) with a 13\*3 (MAX.) matrix key scan circuit. The software configuration features of the HT16K33 makes it suitable for multiple LED applications including LED modules and display subsystems. The HT16K33 is compatible with most microcontrollers and communicates via a two-line bidirectional I2C-bus.

The picture below is the working schematic of HT16K33 chip.



We design the drive module of 8\*8 dot matrix based on the above principle. We could control the dot matrix by I2C communication and two pins of microcontroller, according to the above diagram.

### (4)Specification:

Input voltage: 5V

Rated input frequency: 400KHZ

Input power: 2.5W

Input current: 500mA

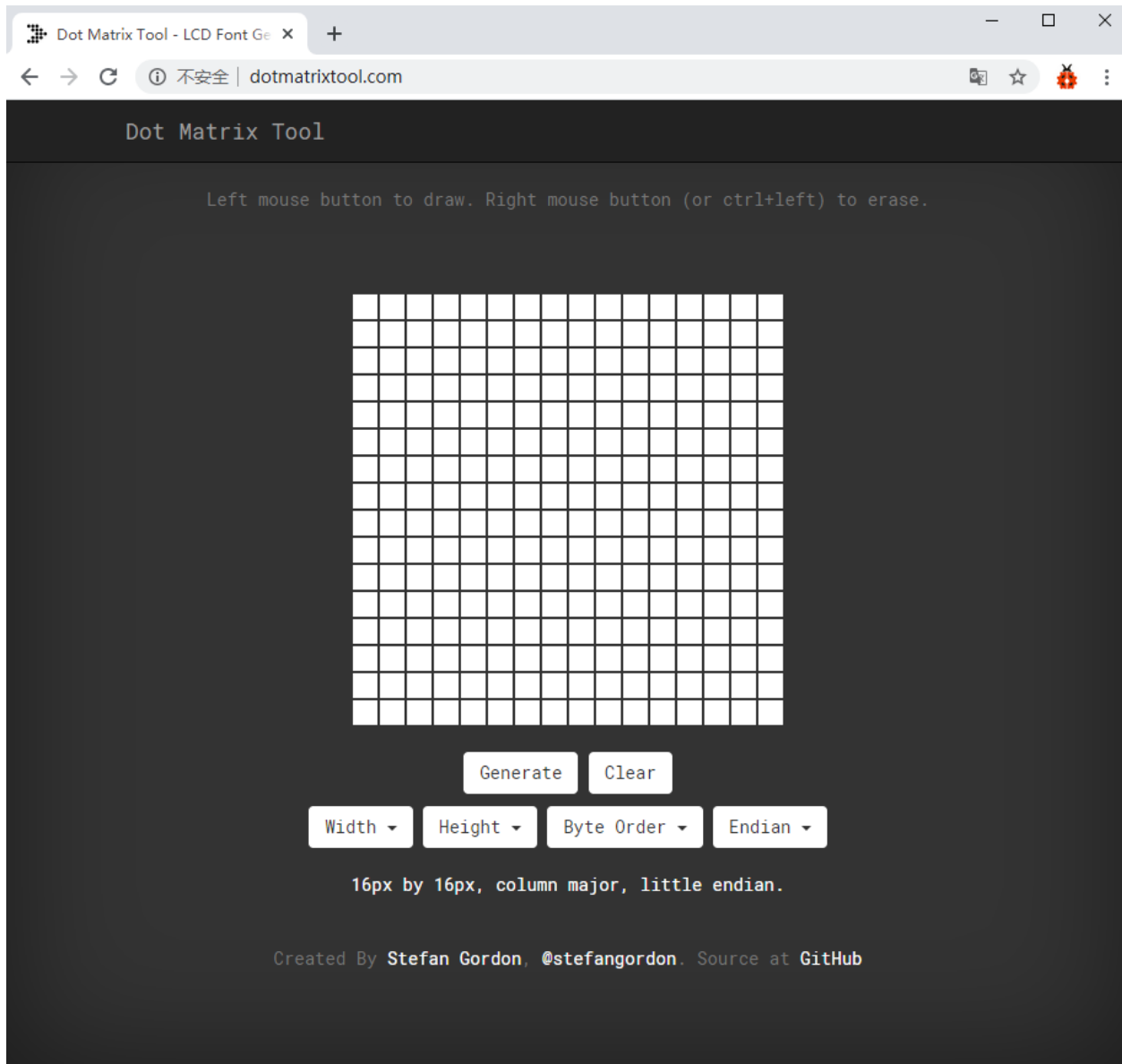
### (5)Introduction for Modulus Tool

The online version of dot matrix modulus tool:

<http://dotmatrixtool.com/#>

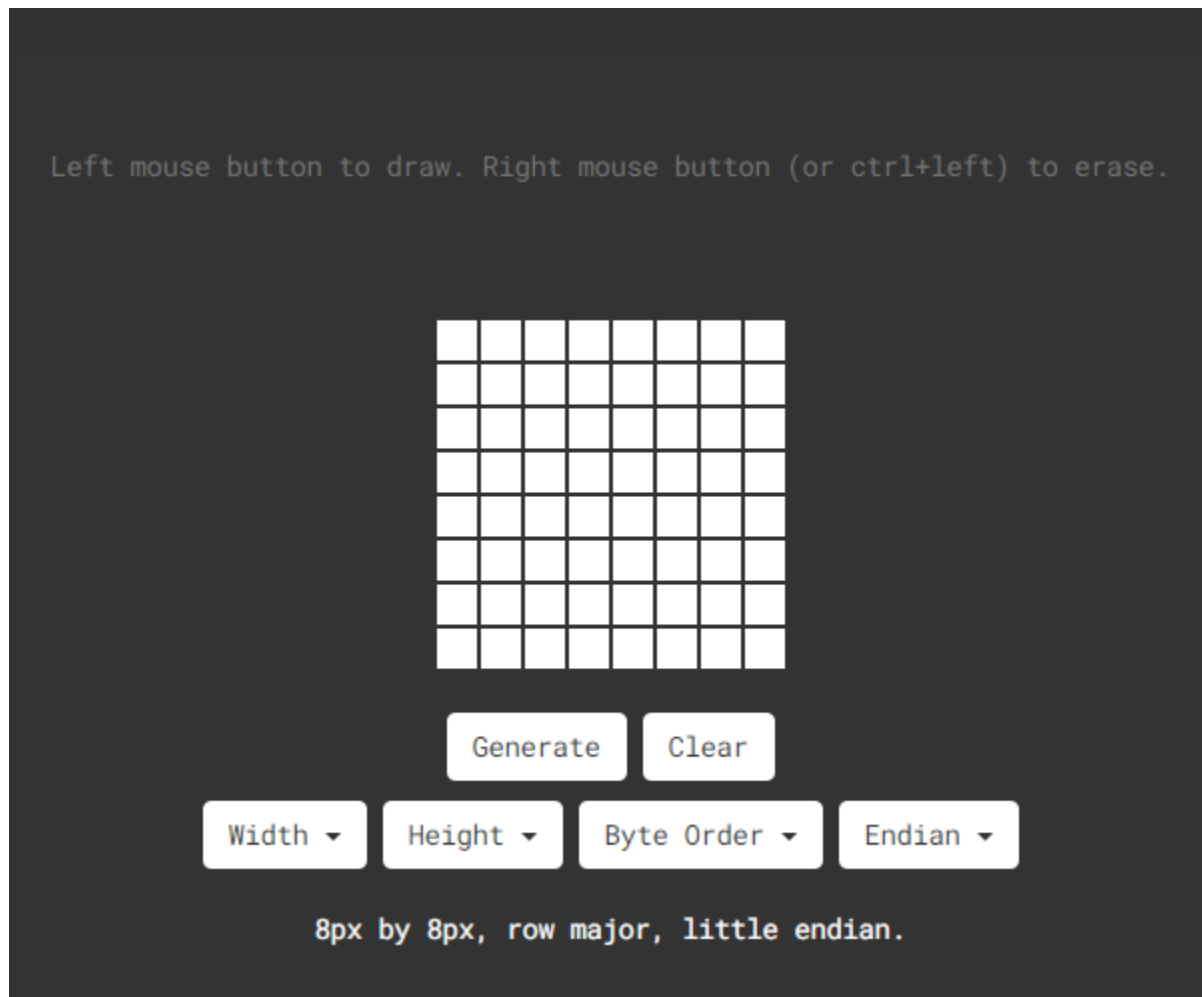
Open the link to enter the following page.



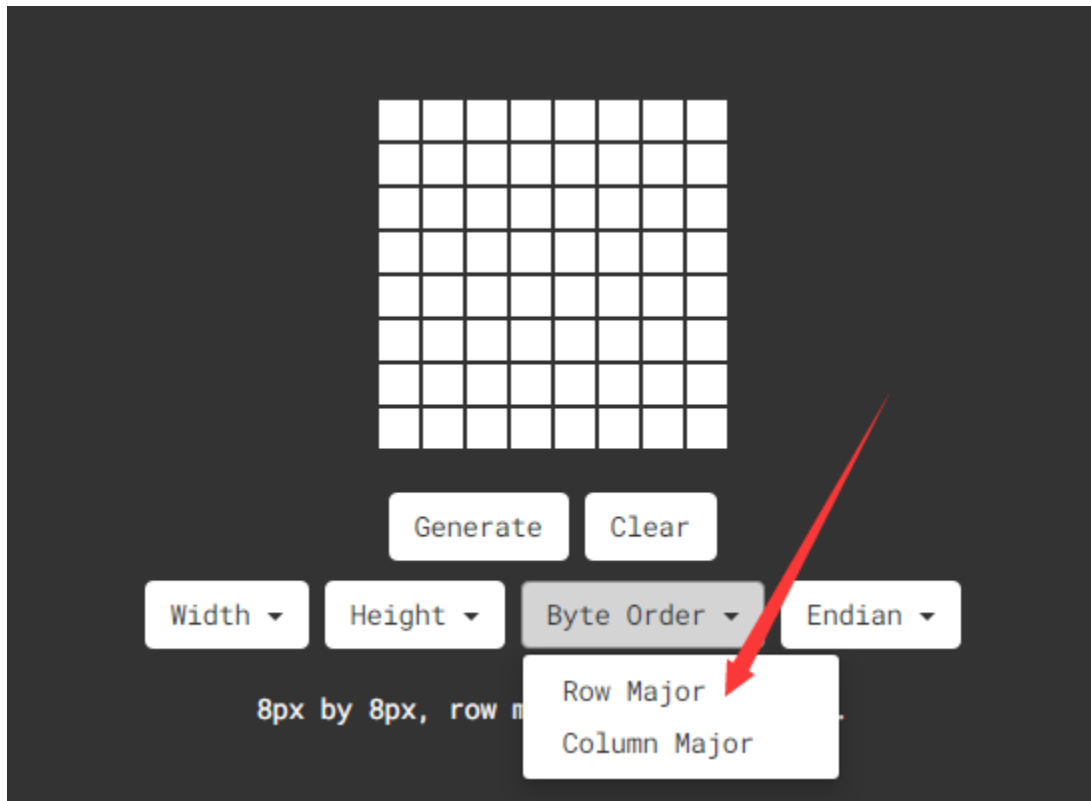


The dot matrix is 8\*8 in this project. So set the height to 8, width to 8; as shown below.



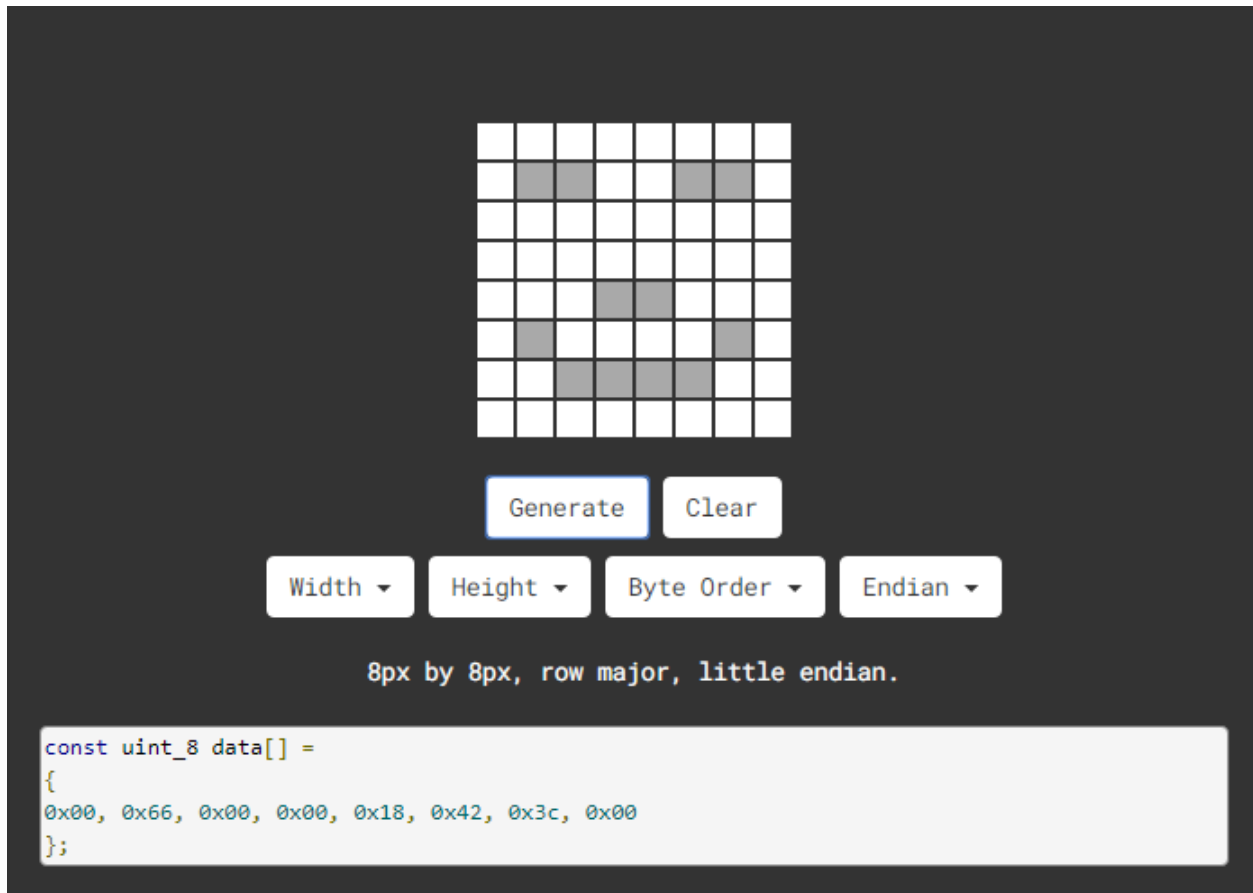


Click Byte order to select Row major



Generate hexadecimal data from the pattern

As shown below, the left button of the mouse is for selection while the right is for canceling. Thus you could use them to draw the pattern you want, then click Generate, to yield the hexadecimal data needed.



The generated hexadecimal code (0x00, 0x66, 0x00, 0x00, 0x18, 0x42, 0x3c, 0x00) is what will be displayed, so you need to save it for next procedure.

## (6)Wiring up

| 8*8 Dot matrix display | PCB Board |
|------------------------|-----------|
| G                      | G         |
| 5V                     | 5V        |
| SDA                    | SDA       |
| SCL                    | SCL       |

## (7)Test Code

The 8\*8 dot matrix is controlled by A4SDA and A5SCL of the Arduino Nano board.

```
/*
  Project 04 8*8 Dot Matrix
  8*8 dot matrix screen to display patterns
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);    //set pins to communication pins
```

(continues on next page)

(continued from previous page)

```

// define an array
uint8_t LedArray1[8]={0x00, 0x66, 0x00, 0x00, 0x18, 0x42, 0x3c, 0x00};
uint8_t LEDArray[8]; //define an array(by modulus tool) without initial value

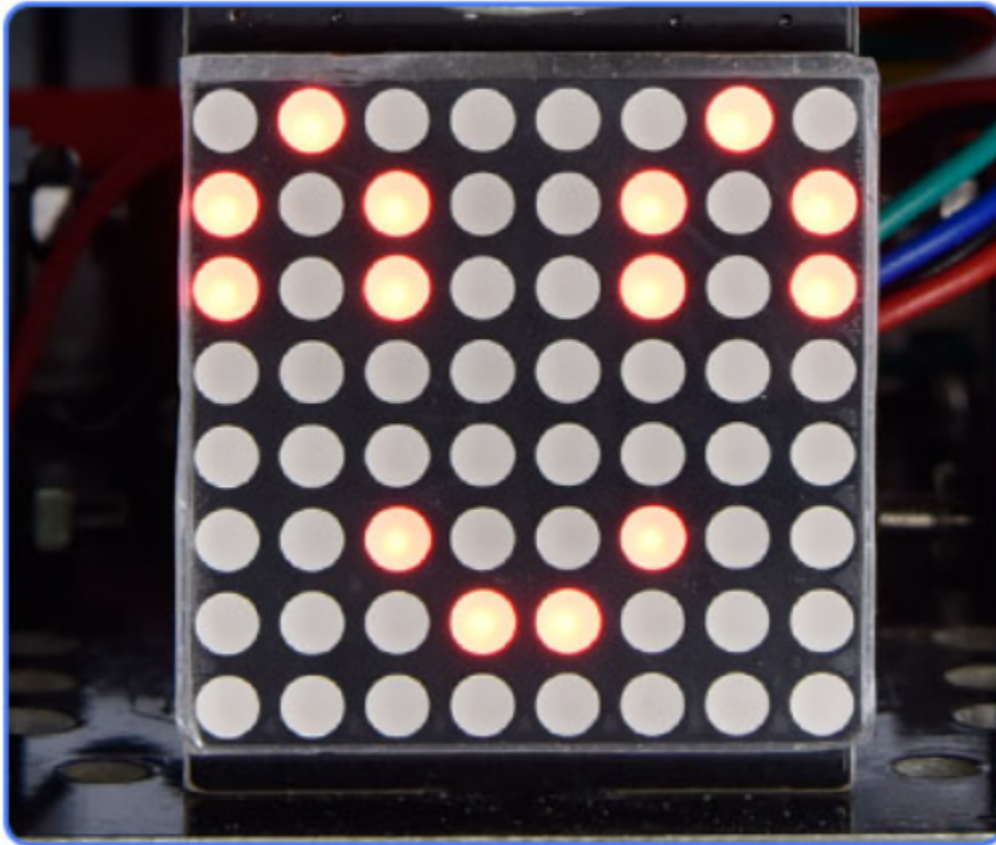
void setup(){
  myMatrix.begin(0x70); //communication address
  myMatrix.clear();    //clear matrix
}

void loop(){
  for(int i=0; i<8; i++) // there is eight data, loop for eight times
  {
    LEDArray[i]=LedArray1[i]; //Call the emoticon array data in the subroutine LEDArray
    for(int j=7; j>=0; j--) //Every data(byte) has 8 bit, therefore, loop for eight times
    {
      if((LEDArray[i]&0x01)>0) //judge if the last bit of data is greater than 0
      {
        myMatrix.drawPixel(j, i,1); //light up the corresponding point
      }
      else //otherwise
      {
        myMatrix.drawPixel(j, i,0); //turn off the corresponding point
      }
      LEDArray[i] = LEDArray[i]>>1; //LEDArray[i] moves right for one bit to judge the
      ↪previous one bit
    }
  }
  myMatrix.writeDisplay(); // dot matrix shows
}

```

## (8)Test Result

Upload the test code to the Arduino Nano board and power up by a USB cable, the 8\*8 dot matrix display will show a“smile”pattern.



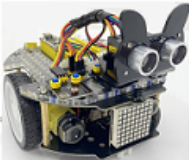



### 8.3.5 Project 5: Servo Rotation

#### (1)Description

There are two servos on the car. We take the servo connected to pin D9 as an example.

The servo is a motor that can rotate very accurately. It has been widely applied to toy cars, remote control helicopters, airplanes, robots and other fields. In this project, we will use the Nano motherboard to control the servo to spin.

#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

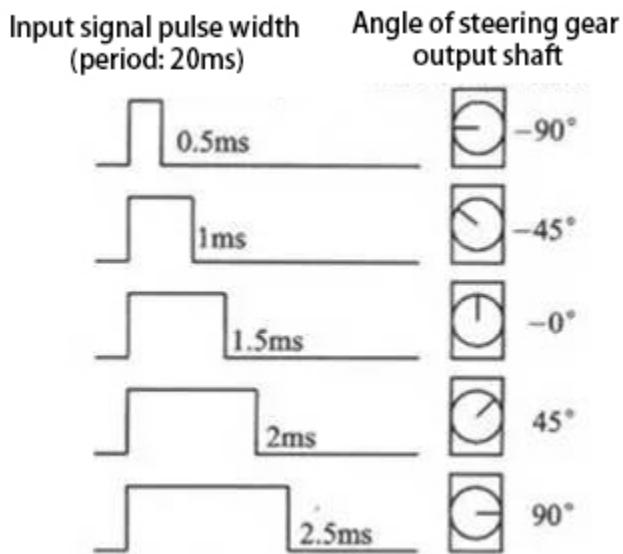
### (3)Knowledge



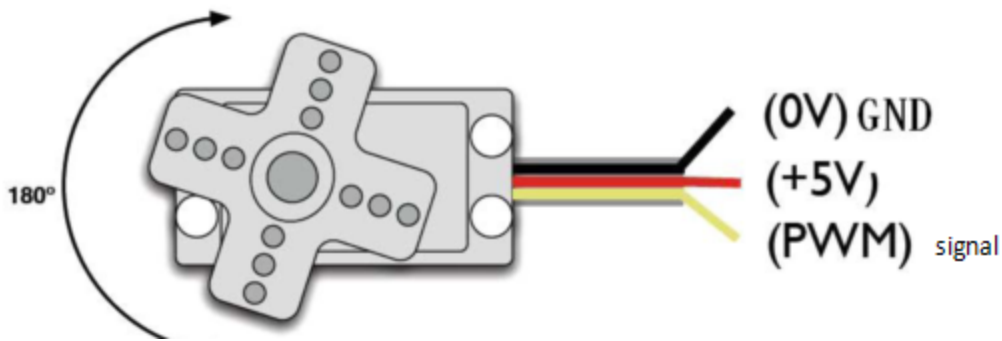
Servo motor is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor. Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of 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  $0^{\circ}$  –  $180^{\circ}$

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  $0^{\circ}$  to  $180^{\circ}$ . 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.



#### (4)Wire up

| Servo  | PCB Board |
|--------|-----------|
| Brown  | G         |
| Red    | 5V        |
| Orange | S1D9      |

#### (5)Test Code

The servo of the ultrasonic sensor is controlled by the GPIO9 of the Pico board.

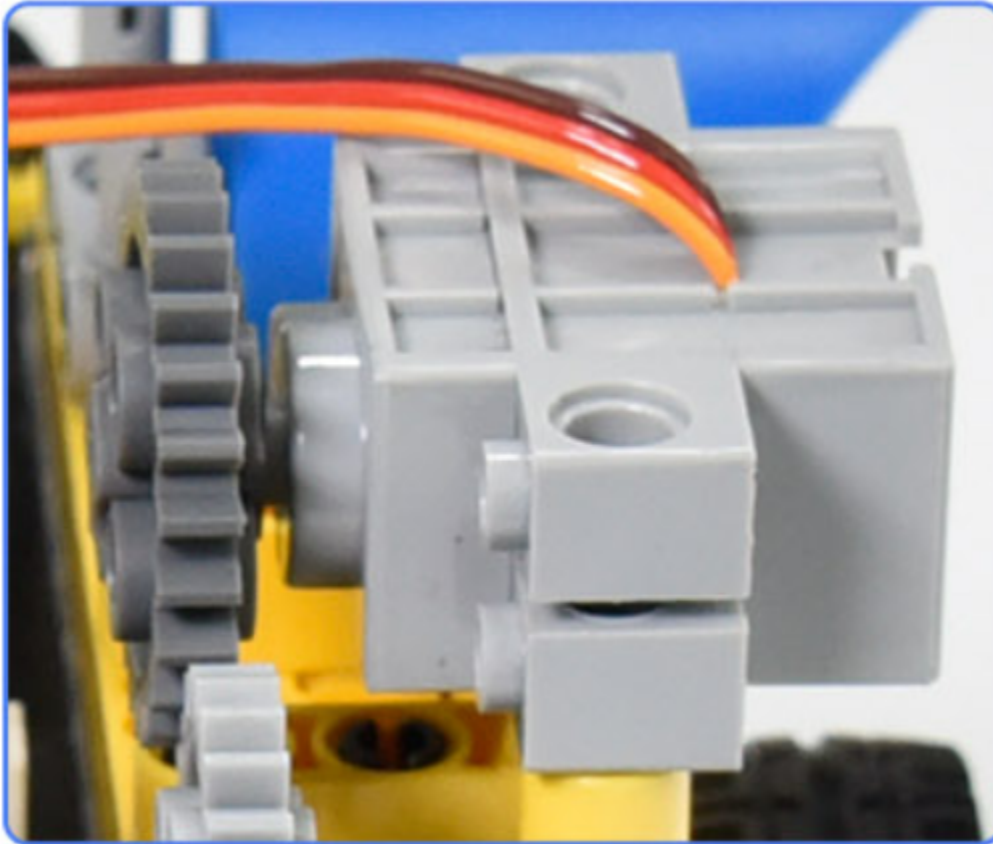
```

/*
Project 05 Servo Rotation
the plastic arm of the servo will rotate at an angle of 0°, 45°, 90°, 135°, and 180°,
↪repeatedly.
*/
#include <Servo.h>
Servo myservo;// define the name of the servo
void setup()
{
myservo.attach(9);// select the pin of the servo(9)
}
void loop()
{
myservo.write(0);// set the rotation angle of the motor
delay(500);
myservo.write(45);// set the rotation angle of the motor
delay(500);
myservo.write(90);// set the rotation angle of the motor
delay(500);
myservo.write(135);// set the rotation angle of the motor
delay(500);
myservo.write(180);// set the rotation angle of the motor
delay(500);
}

```

### (6)Test Result

Upload the test code to the Arduino Nano board, and power up with a USB cable. Then the arm of the servo will rotate to 0°, 45°, 90°, 135° and 180°



## 8.3.6 Project 6: Motor Driving and Speed Control

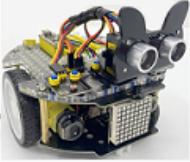



### (1)Description

There are many ways to drive motors. This car uses the most commonly used DRV8833 motor driver chip, which provides a dual-channel bridge electric driver for toys, printers and other motor integration applications.

In this experiment, we use the DRV8833 motor driver chip on the expansion board to drive the two DC motors, and demonstrate the effect of forward, backward, left-turning, and right-turning.



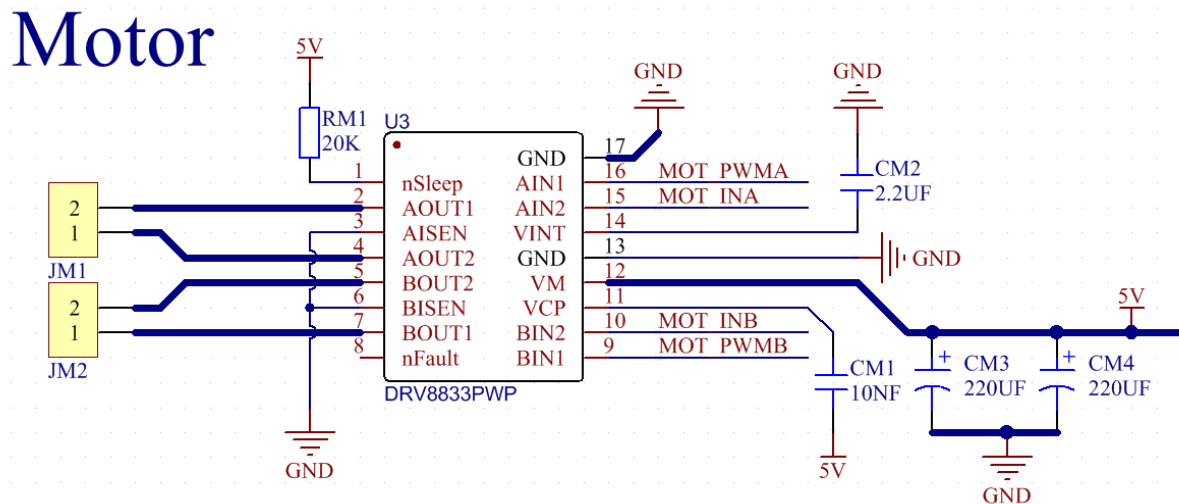
**(2)Components Required**

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

**(3)Knowledge**

DRV8833 motor driver chip: Dual H-bridge motor driver with current control function, can drive two DC motors, one bipolar stepper motor, solenoid valve or other inductive loads. Each H-bridge includes circuitry to regulate or limit winding current.

An internal shutdown function with a fault output pin is used for over-current and short circuit protection, under-voltage lockout and over-temperature. A low-power sleep mode is also added. Let's take a look at the schematic diagram of the DRV8833 motor driver chip driving two DC motors:

**(4)Specification**

Input voltage of logic part: DC 5V

Input voltage of driving part : DC 5V

Working current of logic part: <30mA

Operating current of driving part: <2A

Maximum power dissipation: 10W (T=80°C)

Motor speed: 5V 200 rpm / min

Motor drive form: dual H-bridge drive

Control signal input level: high level  $2.3V < V_{in} < 5V$ , low level  $-0.3V < V_{in} < 1.5V$

Working temperature:  $-25 \sim 130^{\circ}\text{C}$

### (5) Drive the car to move

From the above diagram, the direction pin of the left motor is D4; the speed pin is D6; D2 is the direction pin of the right motor; and D5 is speed pin.

PWM drives the robot car. The PWM value is in the range of 0-255. The more the PWM value is set, the faster the rotation of the motor.

| Function   | D4   | D6PWM | Left motor    | D2   | D5PWM | Right motor   |
|------------|------|-------|---------------|------|-------|---------------|
| forward    | LOW  | 200   | clockwise     | LOW  | 200   | clockwise     |
| Go back    | HIGH | 50    | anticlockwise | HIGH | 50    | anticlockwise |
| Turn left  | HIGH | 200   | anticlockwise | LOW  | 200   | clockwise     |
| Turn right | LOW  | 200   | clockwise     | HIGH | 200   | anticlockwise |
| Stop       | LOW  | 0     | stop          | LOW  | 0     | stop          |

### (6) Test Code

```
/*
  Project 06 Motor drive and speed regulation
  Motor moves forward, backward, left and right
*/
const int left_ctrl = 4; //define the direction control pin(D4) of the left motor
const int left_pwm = 6; // define the speed control pin(D6) of the left motor
const int right_ctrl = 2; //define the direction control pin(D2) of the right motor
const int right_pwm = 5; //define the speed control pin(D5) of the right motor

void setup()
{
  pinMode(left_ctrl, OUTPUT); //Set the direction control pin of the left motor to OUTPUT
  pinMode(left_pwm, OUTPUT); //Set the PWM control speed of the left motor to OUTPUT
  pinMode(right_ctrl, OUTPUT); //Set the direction control pin of the right motor to OUTPUT
  pinMode(right_pwm, OUTPUT); //Set the PWM control speed of the right motor to OUTPUT
}

void loop()
{
  //front
  digitalWrite(left_ctrl, LOW); //Set direction control pins of the left motor to LOW
  analogWrite(left_pwm, 200); //Set the PWM control speed of the left motor to 200
  digitalWrite(right_ctrl, LOW); //set control pins of the right motor to LOW
  analogWrite(right_pwm, 200); //Set the PWM control speed of the right motor to 200
  delay(2000); //delay in 2s

  //back
  digitalWrite(left_ctrl, HIGH); //set control pins of the left motor to HIGH
  analogWrite(left_pwm, 50); //Set the PWM control speed of the left motor to 50
```

(continues on next page)

(continued from previous page)

```

digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
analogWrite(right_pwm,50); //Set the PWM control speed of the right motor to 50
delay(2000); //delay in 2s

//left
digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
delay(2000); //delay in 2s

//right
digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
delay(2000); //delay in 2s

//stop
digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
analogWrite(left_pwm,0); //Set the PWM control speed of the left motor to 0
digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
analogWrite(right_pwm,0); //set Set the PWM control speed of the right motor to 0
delay(2000); //delay in 2s
}

```

## (7)Test Result

Upload the test code to the Arduino Nano board, install batteries, turn the power switch to ON end and power up. The car moves forward for 2s, back for 2s, turn left for 2s, right for 2s and stops for 2s; cyclically

## 8.3.7 Project 7: Ultrasonic Sensor

There is an ultrasonic sensor on the car. It is a very affordable distance-measuring sensor.

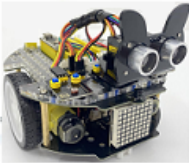



The ultrasonic sensor sends a high-frequency ultrasonic signal that human hearing can't hear. When encountering obstacles, these signals will be reflected back immediately. After receiving the returned information, the distance between the sensor and the obstacle will be calculated by judging the time difference between the transmitted signal and the received signal. It is mainly used for object avoidance and ranging in various robotics projects.

### Project 7.1: Ultrasonic Ranging

#### (1)Description

In this experiment, we use an ultrasonic sensor to measure distance and print the data on a serial monitor.

### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

### (3)Knowledge

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.

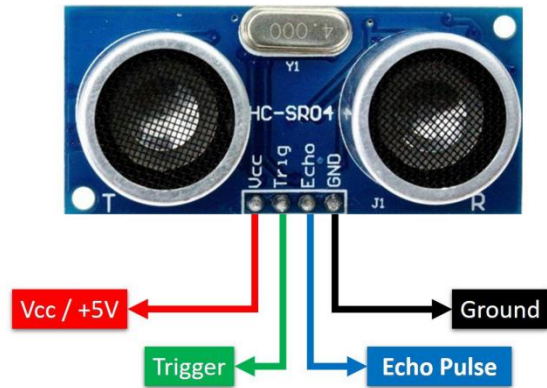


#### Use method and timing chart of ultrasonic module:

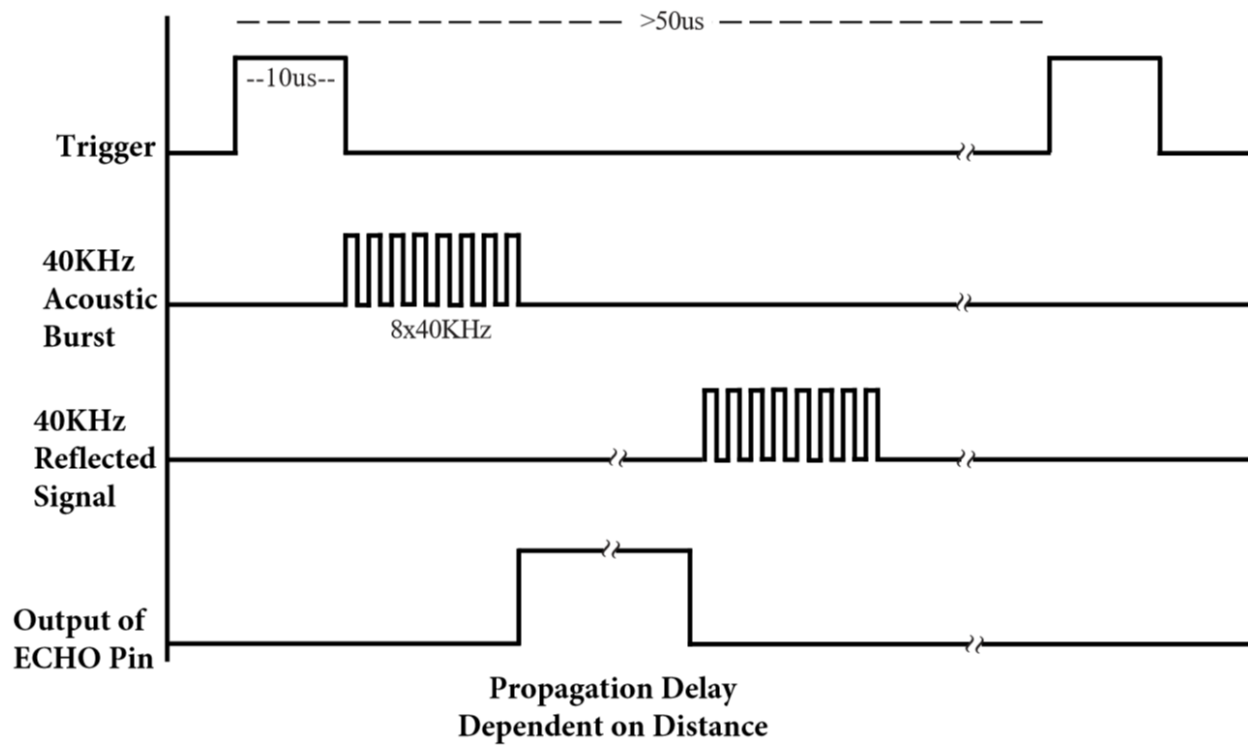
Setting the delay time of Trig pin of SR04 to 10s at least, which can trigger it to detect distance.

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.

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.



HC-SR04 ULTRASONIC MODULE



Time=Echo pulse width, unit: us

Distance<sub>cm</sub>=time/ 58

Distance<sub>(inch)</sub>=time/ 148

The HC-SR04 ultrasonic sensor has four pins: Vcc, Trig, Echo and GND.

The Vcc pin provides power generating ultrasonic pulses and is connected to Vcc/+5V. The GND pin is grounded/GND.

The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse. The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse stroke to the Arduino control board.

#### (4)Wiring Up

| Ultrasonic Sensor | PCB Board |
|-------------------|-----------|
| Vcc               | 5V        |
| Trig              | S2D8      |
| Echo              | S1D7      |
| Gnd               | G         |

#### (5)Test Code

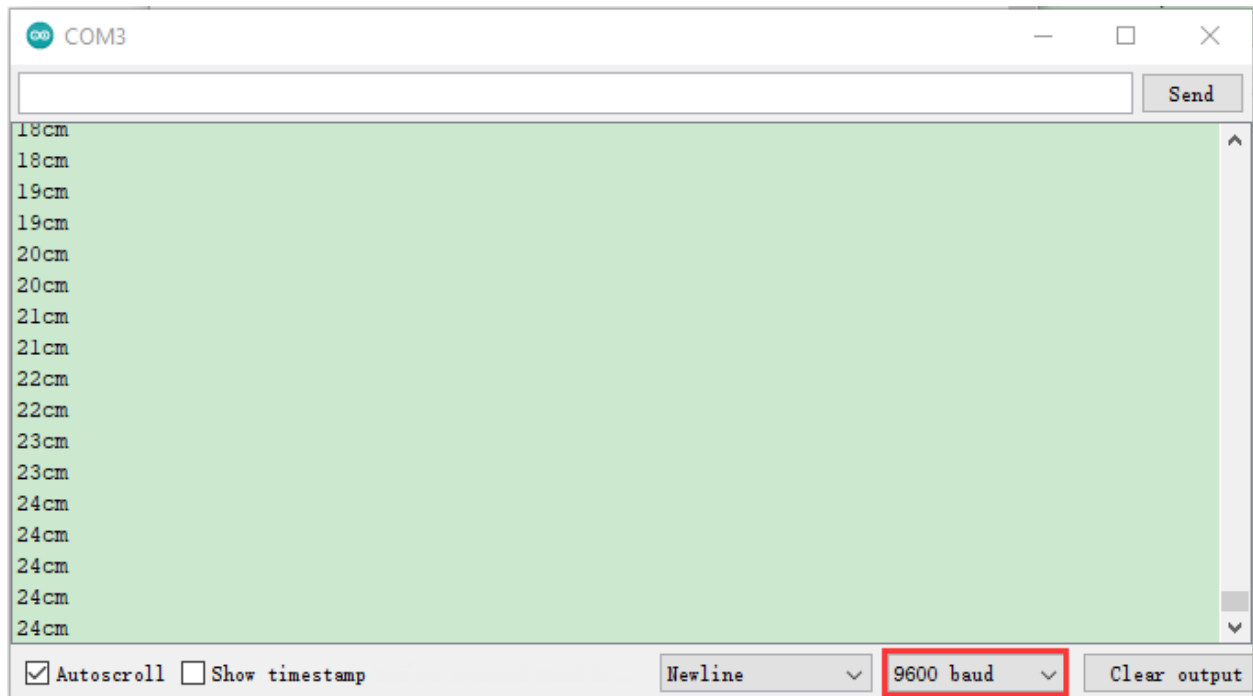
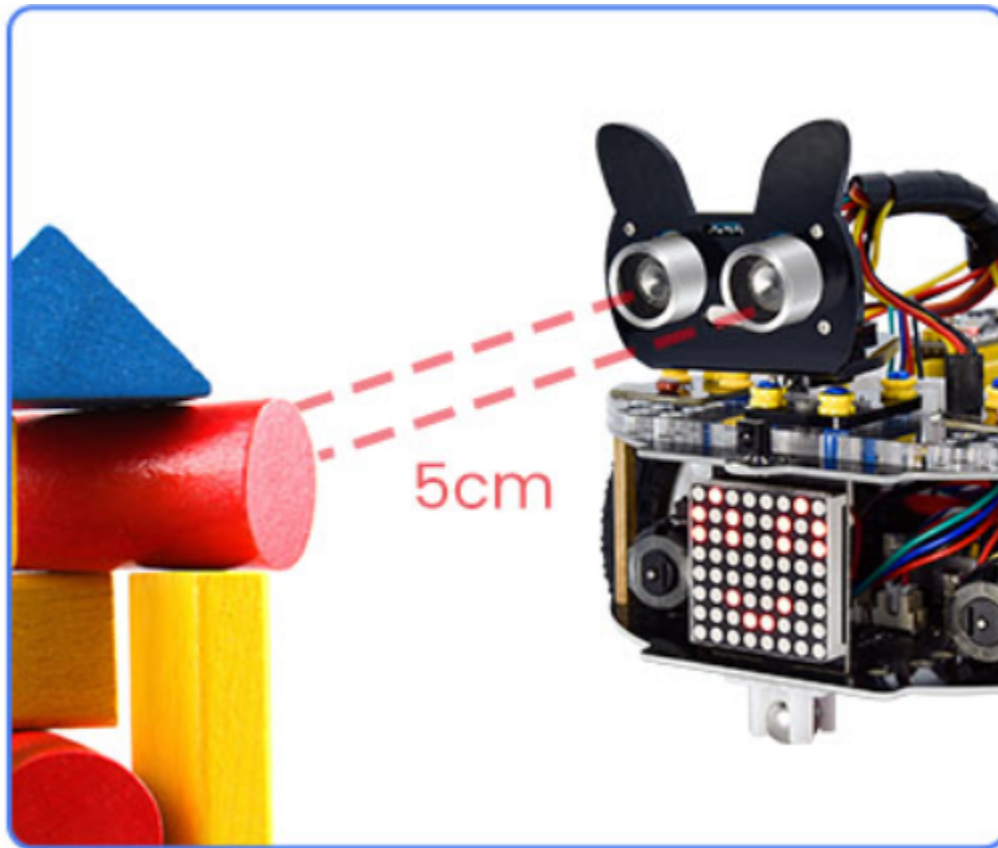
The pin Trig and Echo of the ultrasonic sensor are controlled by the D8 and D7 of the Arduino Nano.

```
/*
Project 07.1 Ultrasonic Ranging
Ultrasonic detection of distance from objects
*/
const int trig = 8; //Define trig pin to D8
const int echo = 7; //Define echo pin to D7
int duration = 0;
int distance = 0; //Define a variable to receive distance
void setup()
{
  pinMode(trig , OUTPUT); // Define the trig pin as the output mode
  pinMode(echo , INPUT); // Define the echo pin as the input mode
  Serial.begin(9600); // Set baud rate to 9600
}
void loop()
{
  digitalWrite(trig , HIGH); //the sensor is triggered by a high pulse of 1000_
  ↳microseconds or more
  delayMicroseconds(1000);
  digitalWrite(trig , LOW); // Give a short low level in advance to ensure a clean high_
  ↳pulse
  duration = pulseIn(echo , HIGH);
  distance = (duration/2) / 28.5 ; //Convert to distance
  Serial.print(distance); // Print the distance in centimeters
  Serial.println("cm");
}
```

#### (6)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor and set baud rate to 9600.

When you move an object in front of the ultrasonic sensor, it will detect the distance and the serial monitor will show the distance value.

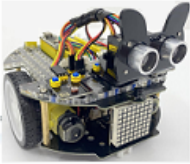





## Project 7.2: Light Following

### (1)Description

In the above experiments, we have learned about the 8\*8 dot matrix, motor drivers and speed regulation, ultrasonic sensors, servos and other hardware. In this experiment, we will combine them to create a follow car with the ultrasonic sensor. The can can follow an object to move through measuring distance.

### (2)Components Required

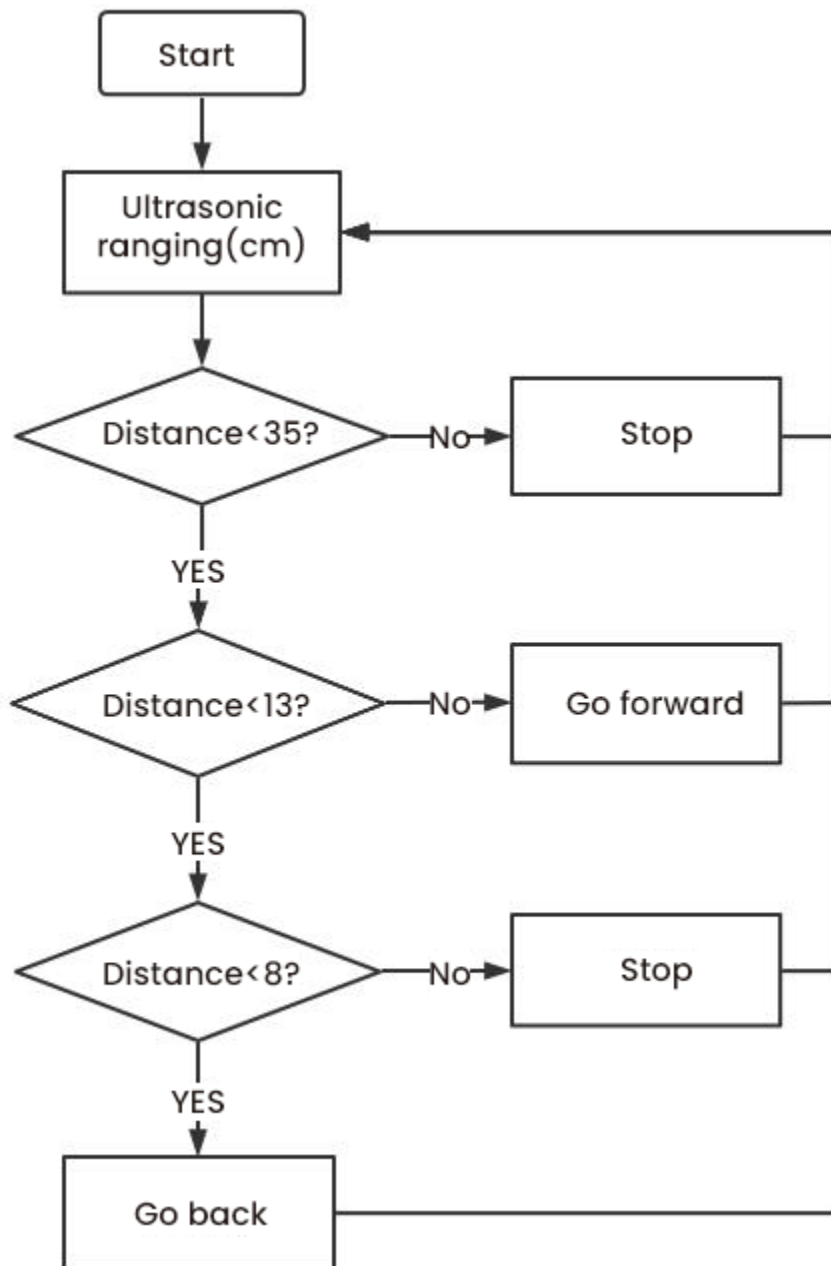
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

### (3)Working Principle

| Detection   | Detect the front distance Distance unitcm |
|-------------|---|
| Condition 1 | Distance8                                 |
| State       | Go backset PWM to 100                     |
| Condition 2 | 8distance<13                              |
| State       | stop                                      |
| Condition 3 | 13distance<35                             |
| State       | Go forwardset PWM to 100                  |
| Condition 4 | distance35                                |
| State       | stop                                      |



## (4)Flow Chart



## (5) Test Code

```

/*
Project 07.2: follow me
Car follows the object
*/
const int left_ctrl = 4; //define direction control pins of the left motor as D4
const int left_pwm = 6; //define speed control pins of the left motor as D6
const int right_ctrl = 2; //define the direction control pin of the right motor D2
const int right_pwm = 5; //define the speed control pin of the right motor D5
#include "SR04.h" //define the ultrasonic module function library
#define TRIG_PIN 8 // define signals input of the ultrasonic as D8
#define ECHO_PIN 7 //define the signal output of the ultrasonic sensor as D7
SR04 sr04 = SR04(ECHO_PIN, TRIG_PIN);
long distance;
const int servopin = 9; //define the pin of the servo as D9
int myangle;
int pulsewidth;

void setup() {
  pinMode(left_ctrl, OUTPUT); //Set the direction control pin of the left motor to OUTPUT
  pinMode(left_pwm, OUTPUT); //Set the PWM control speed of the left motor to OUTPUT
  pinMode(right_ctrl, OUTPUT); //Set the direction control pin of the right motor to OUTPUT
  pinMode(right_pwm, OUTPUT); //Set the PWM control speed of the right motor to OUTPUT
  pinMode(TRIG_PIN, OUTPUT); //Set TRIG_PIN to OUTPUT
  pinMode(ECHO_PIN, INPUT); //Set ECHO_PIN to INPUT
  servopulse(servopin, 90); //set the initial angle to 90
  delay(300);
}

void loop() {
  distance = sr04.Distance(); //the distance detected by the ultrasonic sensor
  if(distance < 8) //if the distance is less than 8
  {
    back(); //go back
  }
  else if((distance >= 8) && (distance < 13)) //if 8distance < 13
  {
    Stop(); //stop
  }
  else if((distance >= 13) && (distance < 35)) //if 13distance < 35
  {
    front(); //follow
  }
  else //if above conditions are not met
  {
    Stop(); //stop
  }
}

void servopulse(int servopin, int myangle) //angles the servo rotate
{
  for(int i=0; i<20; i++)

```

(continues on next page)

(continued from previous page)

```

{
  pulsedwidth = (myangle*11)+500;
  digitalWrite(servopin,HIGH);
  delayMicroseconds(pulsedwidth);
  digitalWrite(servopin,LOW);
  delay(20-pulsedwidth/1000);
}
}

void front()//define the state of going forward
{
  digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
  analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
  digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
  analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void back()//define the state of going back
{
  digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
  analogWrite(left_pwm,50); //Set the PWM control speed of the left motor to 50
  digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
  analogWrite(right_pwm,50); //Set the PWM control speed of the right motor to 50
}

void Stop()//define the state of stop
{
  digitalWrite(left_ctrl,LOW);//Set direction control pins of the left motor to LOW
  analogWrite(left_pwm,0);//set the PWM control speed of the left motor to 0
  digitalWrite(right_ctrl,LOW);//set control pins of the right motor to LOW
  analogWrite(right_pwm,0);//set the PWM control speed of the right motor to 0
}

```

## (6)Test Result

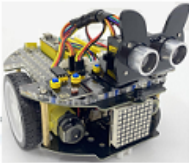



Upload the code to the Arduino Nano board, install batteries and turn the switch to the ON end and power up. Then the car will follow the obstacle to move.

## Project 7.3: Dodge obstacles

### (1)Description

In this project, we will take advantage of the ultrasonic sensor to detect the distance away from the obstacle so as to avoid them.

(2)Components Required

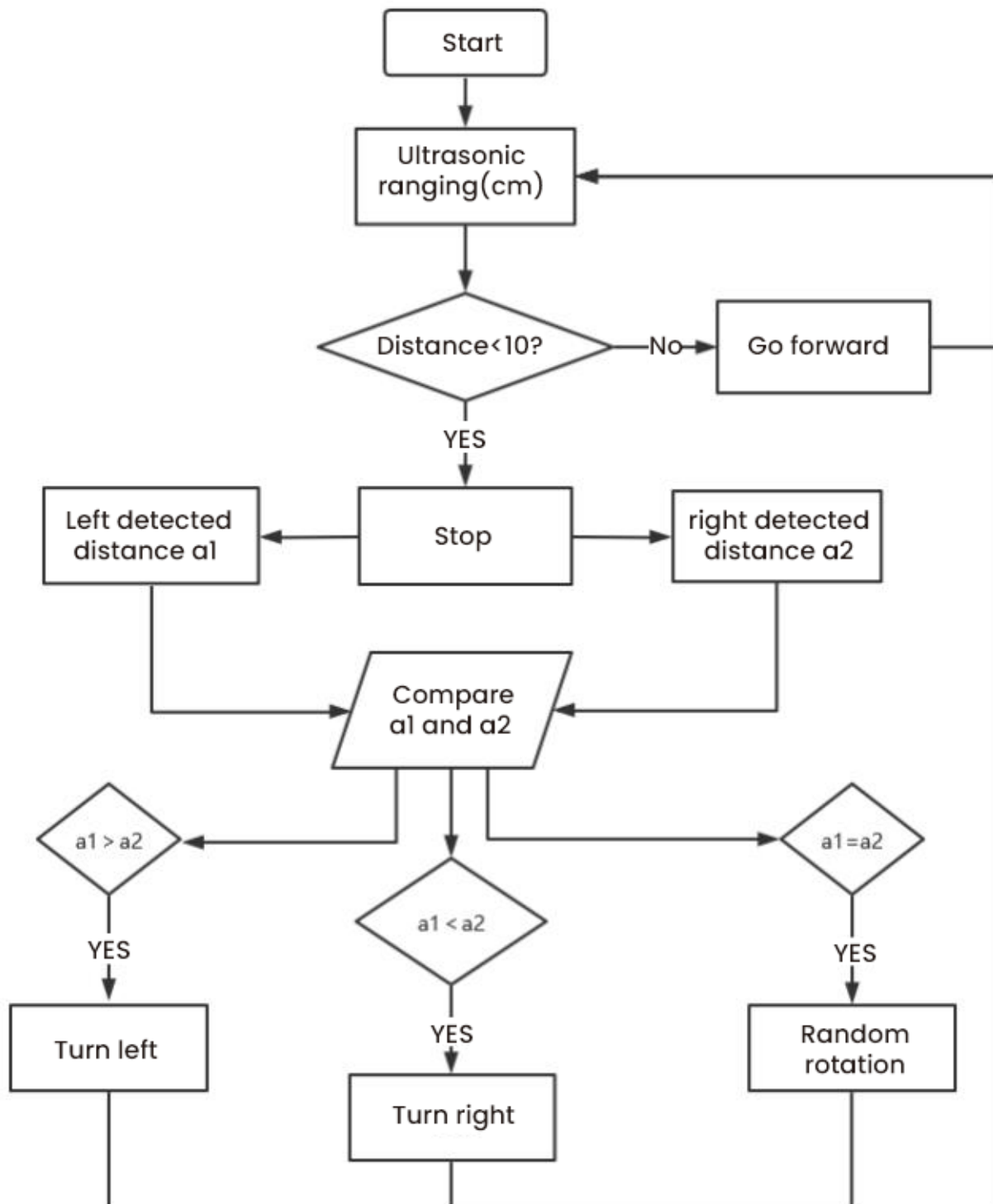
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

(3)Working Principle

|      |   |                         |  |
|------|---|-------------------------|--|
|      | 8*8 Dot matrix display                                |                         |  |
|      | Set servo to 90°                                      |                         |  |
| loop | Detect the distance away from the obstacle (unit: cm) |                         |  |
|      | Condition 1   | State                   |  |
|      | 0<distance < 10                                       | Stop                    |  |
|      |   | Show the "stop" pattern |  |
|      |   | Set the servo to 180°   | Distance away form the obstacle: a1 (unit: cm) |
|      |   | Set the servo to 0°     | Distance away form the obstacle: a2 (unit: cm) |
|      |   | Condition 2             | State  |

|  |                           |  |                                  |
|--|---------------------------|--|----------------------------------|
|  |                           | $a1 < a2$  | Car turns right (set PWM to 200) |
|  |                           |  | show "turning right" pattern     |
|  |                           |  | Set servo to 90°                 |
|  |                           | $a1 \geq a2$   | Turn left (set PWM to 200)       |
|  |                           |  | display "left turning" pattern   |
|  |                           |  | Set servo to 90°                 |
|  | $\text{distance} \geq 10$ | The 8*8 dot matrix display shows "going forward" pattern |                                  |
|  |                           | Go forward (set PWM to 200)                              |                                  |

## (4)Flow Chart



**(5)Test Code**

```

/*
Project 07.3: avoid obstacles
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5); // define pins of the dot matrix display as A4 and A5
//Array, used to store pattern data, which can be calculated by yourself or obtained
↳ from the touch tool
uint8_t matrix_front[8]={0x18,0x24,0x42,0x99,0x24,0x42,0x81,0x00};
uint8_t matrix_back[8]={0x00,0x81,0x42,0x24,0x99,0x42,0x24,0x18};
uint8_t matrix_left[8]={0x12,0x24,0x48,0x90,0x90,0x48,0x24,0x12};
uint8_t matrix_right[8]={0x48,0x24,0x12,0x09,0x09,0x12,0x24,0x48};
uint8_t matrix_stop[8]={0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18};
uint8_t LEDArray[8];
const int left_ctrl = 4; //define the direction control pin of the left motor to D4
const int left_pwm = 6; //define the speed control pin of the left motor D6
const int right_ctrl = 2; //define the direction control pin of the right motor D2
const int right_pwm = 5; //define the speed control pin of the right motor D5
#include "SR04.h" //define the ultrasonic module function library
#define TRIG_PIN 8 // define signals input of the ultrasonic as D8
#define ECHO_PIN 7 //define the signal pin of the ultrasonic sensor as D7
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance,a1,a2; //define three distance variables
const int servopin = 9; //define the pin of the servo as D9
int myangle;
int pulsewidth;
int val;

void setup() {
  pinMode(left_ctrl,OUTPUT); //Set the direction control pin of the left motor to OUTPUT
  pinMode(left_pwm,OUTPUT); //Set the PWM control speed of the left motor to OUTPUT
  pinMode(right_ctrl,OUTPUT); //Set the direction control pin of the right motor to OUTPUT
  pinMode(right_pwm,OUTPUT); //Set the PWM control speed of the right motor to OUTPUT
  pinMode(TRIG_PIN,OUTPUT); //Set TRIG_PIN to OUTPUT
  pinMode(ECHO_PIN,INPUT); //Set ECHO_PIN to INPUT
  servopulse(servopin,90); //set the initial angle of the servo to 90
  delay(300);
  myMatrix.begin(112);
  myMatrix.clear();
}

void loop()
{
  avoid(); //run the code of obstacle avoidance
}

void avoid()
{
  distance=sr04.Distance(); //obtain the value detected by the ultrasonic sensor
  if((distance < 10)&&(distance != 0)) // if 0<distance <10
  {
    car_Stop(); //stop
  }
}

```

(continues on next page)

(continued from previous page)

```

myMatrix.clear();
myMatrix.writeDisplay();// show stop pattern
matrix_display(matrix_stop); //show the pattern to stop
delay(100);
servopulse(servopin,180);//servo rotates to 180°
delay(200);
a1=sr04.Distance();//measure distance
delay(100);
servopulse(servopin,0);//rotate to 0°
delay(200);
a2=sr04.Distance();//measure distance
delay(100);
if(a1 > a2)//compare distance, the left one is longer than the right
{
    car_left();//turn left
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_left); //show the pattern to turn left
    servopulse(servopin,90);//rotate to 90°
    //delay(50);
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_front); //show the pattern to go front
}
else//if the right distance is longer than the left distance
{
    car_right();//turn right
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_right); //show the patter to turn right
    servopulse(servopin,90);//the servo rotate to 90°
    //delay(50);
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_front); //show the pattern to go front
}
}
else//if above conditions are not met
{
    car_front();//go front
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_front); //show the pattern to go front
}
}

void servopulse(int servopin,int myangle)//
{
    for(int i=0; i<20; i++)
    {
        pulsewidth = (myangle*11)+500;
        digitalWrite(servopin,HIGH);
    }
}

```

(continues on next page)



(continued from previous page)

```

    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delay(20-pulsewidth/1000);
}
}

void car_front()//define the state of going front
{
    digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void car_back()//define the state of going back
{
    digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
    analogWrite(left_pwm,50); //Set the PWM control speed of the left motor to 50
    digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
    analogWrite(right_pwm,50); //Set the PWM control speed of the right motor to 50
}

void car_left()//define the state of turning left
{
    digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,LOW); //Set control pins of the right motor to LOW
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void car_right()//define the state of turning left
{
    digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void car_Stop()//define the state of stopping
{
    digitalWrite(left_ctrl,LOW);//Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,0);//Set the PWM control speed of the left motor to 0
    digitalWrite(right_ctrl,LOW);//set control pins of the right motor to LOW
    analogWrite(right_pwm,0);//Set the PWM control speed of the right motor to 0
}

//show functions of patterns
void matrix_display(unsigned char matrix_value[])
{
    for(int i=0; i<8; i++)
    {
        LEDArray[i]=matrix_value[i];
        for(int j=7; j>=0; j--)
        {
            if((LEDArray[i]&0x01)>0)
                myMatrix.drawPixel(j, i,1);
        }
    }
}

```

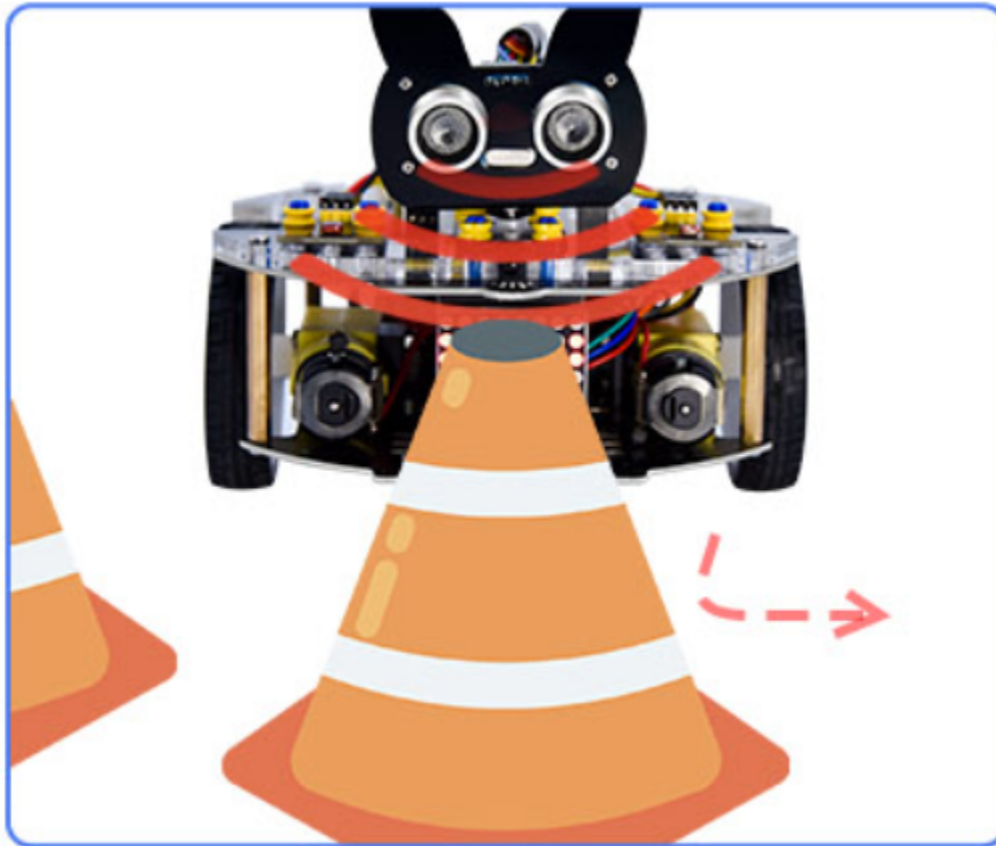
(continues on next page)

(continued from previous page)

```
    LEDArray[i] = LEDArray[i]>>1;
  }
}
myMatrix.writeDisplay();
}
```

### (6)Test Result

Upload the test code to the Arduino Nano board, put batteries in the battery holder, turn the power switch to the ON end and power up. Then the car can automatically dodge obstacles.



### 8.3.8 Project 8: Line Tracking Sensor

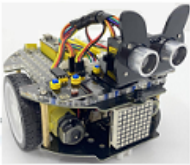



There are two IR line tracking sensors on the car. They are actually two pairs of ST188L3 infrared tubes and used to detect black and white lines. In this project, we will make a line tracking car.

#### Project 8.1: Reading Values

##### (1)Description

In this experiment, we use ST188L3 infrared tubes to detect black and white lines, then print the data on the serial monitor.

##### (2)Components Required

|  |   |   |   |
|--|---|---|---|
| Robot without Wifi module*1  | USB Cable*1   | Computer*1  | 18650 Battery*1   |
|  |  |  |  |

##### (3)Knowledge

##### Infrared line tracking:

The IR line tracking sensor boasts a pair of ST188L3 infrared tubes. ST188L3 tubes has an infrared emitting diode and a receiver tube. When the emitting diode emits an infrared signal then received by the receiving tube after being reflected by the white object. Once the receiving tube receives the signal, the output terminal will output a low level (0); when the infrared emitting diode emits an infrared signal, and the infrared signal is absorbed by the black object, a high level (1) will be output, thus realizing the function of detecting signals through infrared rays.

**Warning:** Reflective optical sensors (including IR line tracking sensors) shouldn't be applied under sunlight as there is a lot of invisible light such as infrared and ultraviolet.

Values detected by the line tracking sensor are shown in the table.

The value will be 1 if detecting black or no objects and the value 0 will appear if detecting white objects.

he detected black object or no object represents 1, and the detected white object represents 0.

| Left | Right | ValueBinary |
|------|-------|-------------|
| 0    | 0     | 00          |
| 0    | 1     | 01          |
| 1    | 0     | 10          |
| 1    | 1     | 11          |

#### (4)Test Code

The line tracking sensors of the PCB board are controlled by D11 and D10 of the Arduino Nano board.

```
/*
Project 08.1: Tracking sensor read value
*/
int tracking_left = 11; //define the pin of the left sensor as D11
int tracking_right = 10; //define the pin of the right sensor as D10
int L_val,R_val; //define two variables of two sensors

void setup() {
  Serial.begin(9600); //set baud rate to 9600
  pinMode(tracking_left, INPUT); //set pins of the left sensor to OUTPUT
  pinMode(tracking_right, INPUT); //set pins of the right sensor to INPUT
}

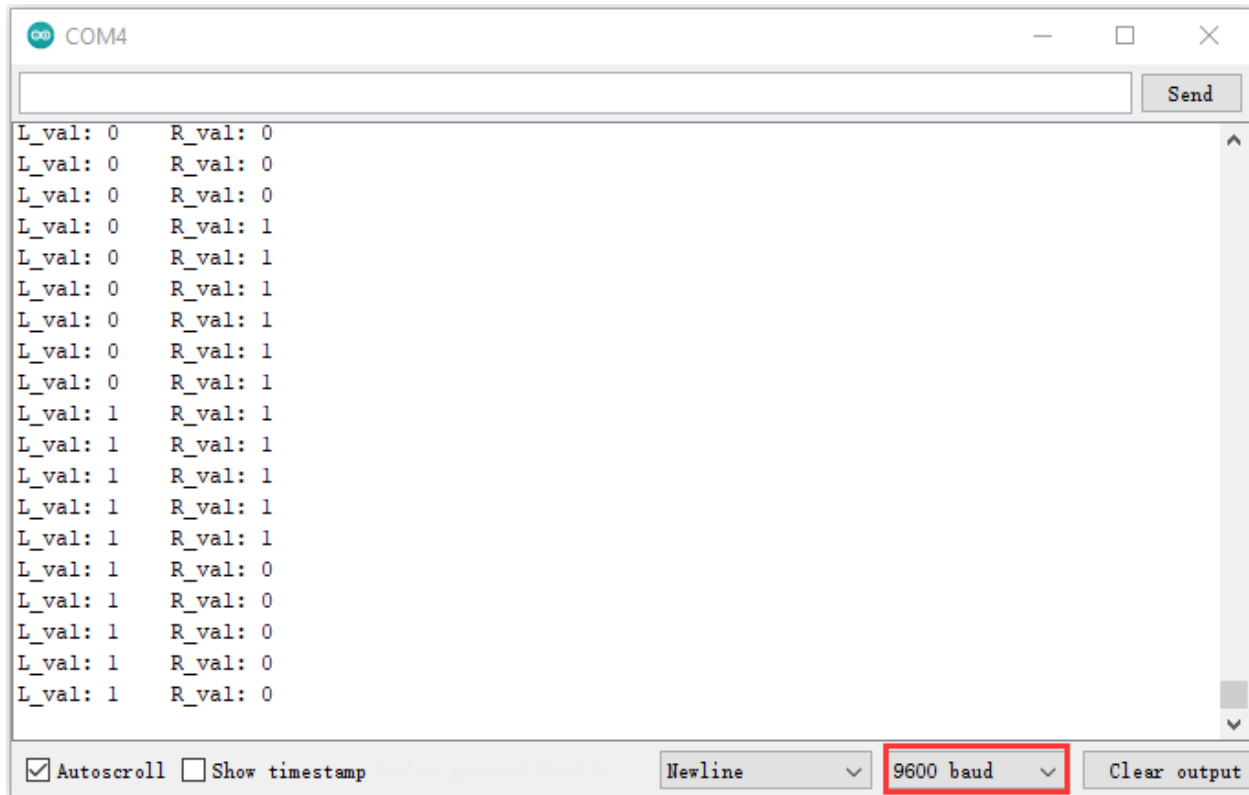
void loop() {
  L_val = digitalRead(tracking_left); //read the value of the left sensor
  R_val = digitalRead(tracking_right); //read the value of the right sensor
  Serial.print("L_val: "); //serial print L_val
  Serial.print(L_val); //serial prints L_val
  Serial.print(" "); //serial prints space key
  Serial.print("R_val: "); //serial prints R_val
  Serial.println(R_val); //serial prints the R_val
  delay(300); //delay in 0.3s
}
```

#### (5)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor and set baud rate to 9600.

Put a black thing under the line tracking sensor of the car and move it, you will see different indicators light up, and at the same time you will see the value on the serial monitor.

The sensitivity can be adjusted by rotating the potentiometer. When the indicator light is adjusted to the critical point of on and off state, the sensitivity is the highest.







## Project 8.2: Line Tracking

### (1)Description

We've introduced the knowledge of motor drivers, speed regulation, and infrared line tracking. In this experiment, the car will perform different actions according to the values transmitted by the infrared tracking.

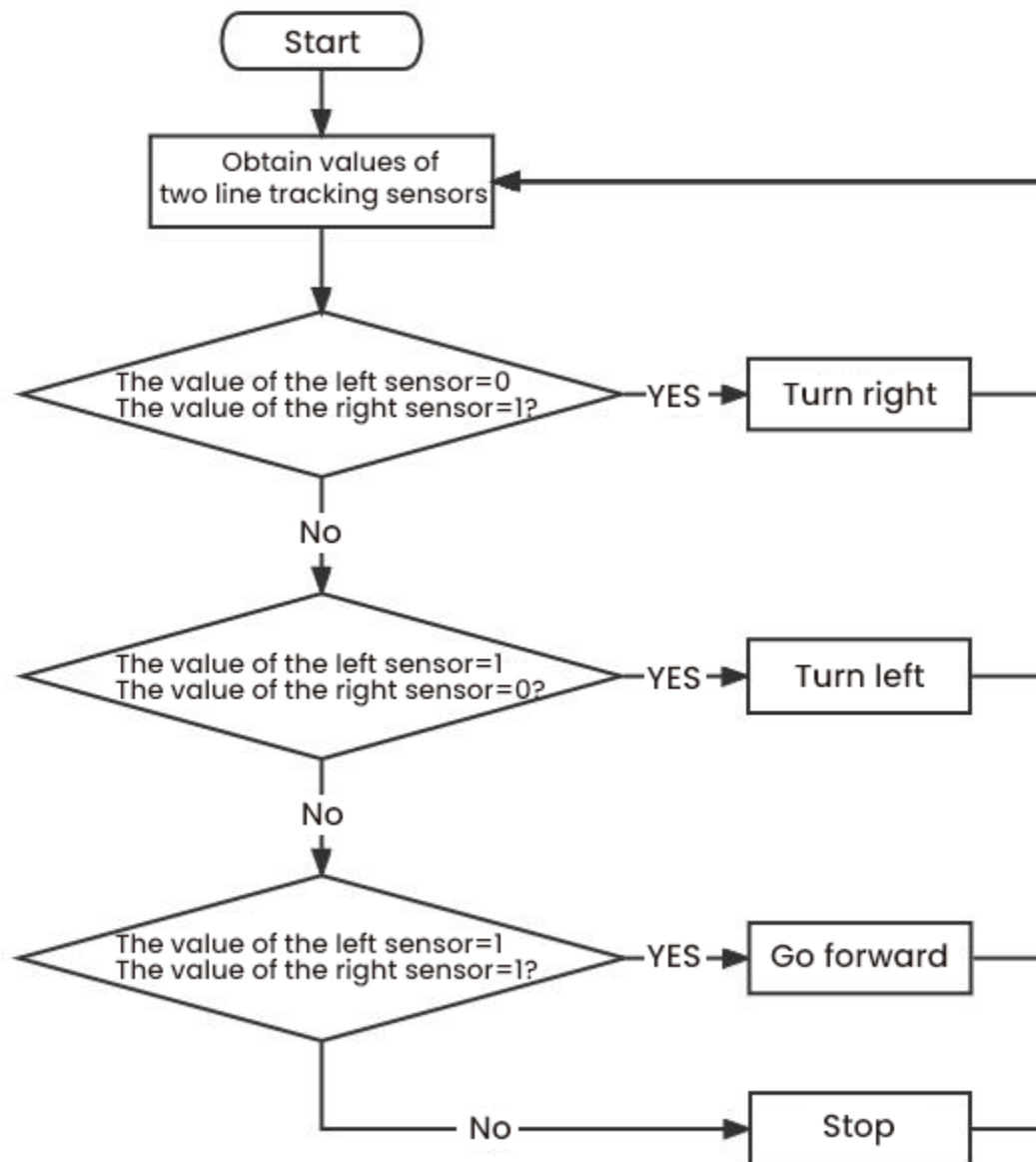
### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

## (3)Working Principle

| Left | Right | ValueBinary | State        |
|------|-------|-------------|--------------|
| 0    | 0     | 00          | Stop         |
| 0    | 1     | 01          | Turn right   |
| 1    | 0     | 10          | Turn left    |
| 1    | 1     | 11          | Move forward |

## (4)Flow Chart



**(5)Test Code**

```

/*
Project 08.2: Follow line to walk
*/
const int left_ctrl = 4; //define direction control pins of the left motor as D4
const int left_pwm = 6; //define speed control pins of the left motor as D5
const int right_ctrl = 2; //define the direction control pin of the right motor D2
const int right_pwm = 5; //define the speed control pin of the right motor D5
int tracking_left = 11; //define the pin of the left sensor as D11
int tracking_right = 10; //define the pin of the right sensor as D10
int L_val, R_val; //define two variables of two line tracking sensors
const int servopin = 9; //define the pin of the servo as D9
int myangle;
int pulsewidth;

void setup() {
  pinMode(left_ctrl, OUTPUT); //Set the direction control pin of the left motor to OUTPUT
  pinMode(left_pwm, OUTPUT); //Set the PWM control speed of the left motor to OUTPUT
  pinMode(right_ctrl, OUTPUT); //Set the direction control pin of the right motor to OUTPUT
  pinMode(right_pwm, OUTPUT); //Set the PWM control speed of the right motor to OUTPUT
  pinMode(tracking_left, INPUT); //set the pin of the left sensor to INPUT
  pinMode(tracking_right, INPUT); //set the pin of the right sensor to INPUT
  servopulse(servopin, 90); //set the initial angle of the servo to 90
  delay(300);
}

void loop()
{
  tracking(); //run the main program
}

void tracking()
{
  L_val = digitalRead(tracking_left); //Read the value of the left sensor
  R_val = digitalRead(tracking_right); //Read the value of the right sensor
  if((L_val == 1) && (R_val == 1)) //if sensors detect black lines
  {
    front(); //go forward
  }
  else if((L_val == 1) && (R_val == 0)) //if only the left sensor detects black lines
  {
    left(); //turn left
  }
  else if((L_val == 0) && (R_val == 1)) //if only the right one detects the black line
  {
    right(); //turn right
  }
  else //if none of sensors detects black lines
  {
    Stop(); //stop
  }
}

```

(continues on next page)

(continued from previous page)

```

void servopulse(int servopin,int myangle)//angles the servo run
{
  for(int i=0; i<20; i++)
  {
    pulsewidth = (myangle*11)+500;
    digitalWrite(servopin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delay(20-pulsewidth/1000);
  }
}

void front()//define the state of going front
{
  digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
  analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
  digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
  analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void left()//define the state of turning left
{
  digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
  analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
  digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
  analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void right()//define the state of turning left
{
  digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
  analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
  digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
  analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void Stop()//define the state of stopping
{
  digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
  analogWrite(left_pwm,0); //Set the PWM control speed of the left motor to 0
  digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
  analogWrite(right_pwm,0); //Set the PWM control speed of the right motor to 0
}

```



## (6)Test Result

Upload the test code to the Arduino Nano board, turn the power switch to the ON end, power up and put the car on a map we provide. Then it will perform different functions via values sent by line tracking sensors.

## 8.3.9 Project 9: Light Following

There are two photoresistors on the car. They can vary with the light intensity and send information to the Nano board to control the car.

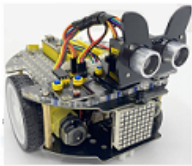



Photoresistors can determine and conduct the car to move by detecting light.

### Project 9.1 Read Values

#### (1)Description

In this experiment, we will learn the working principle of the photoresistor.

#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

#### (3)Knowledge

##### Photoresistor:

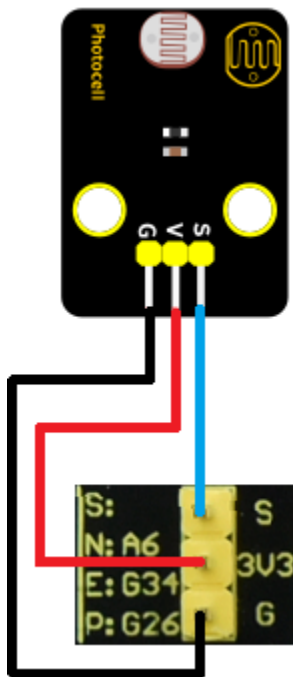
It mainly uses a photosensitive resistance element whose resistance varies from the light intensity. The signal terminal of the sensor is connected to the analog port of the microcontroller. When the light is stronger, the analog value at the analog port will increase; on the contrary, when the light intensity is weaker, the analog value of the microcontroller will reduce. In this way, the corresponding analog value can reflect the ambient light intensity.

#### (4)Wire up

Through the wiring-up diagram, signal pins of two photoresistors are connected to A6 and A7 of the Nano board.

For the following experiment, we use the photoresistor connected to A6 to finish experiments. First, let's read analog values.

| Left photoresistor | PCB board |
|--------------------|-----------|
| G                  | G         |
| V                  | V         |
| S                  | SA6       |



#### (5)Test Code

The left photoresistor is controlled by the A6 of the Arduino Nano board.

```

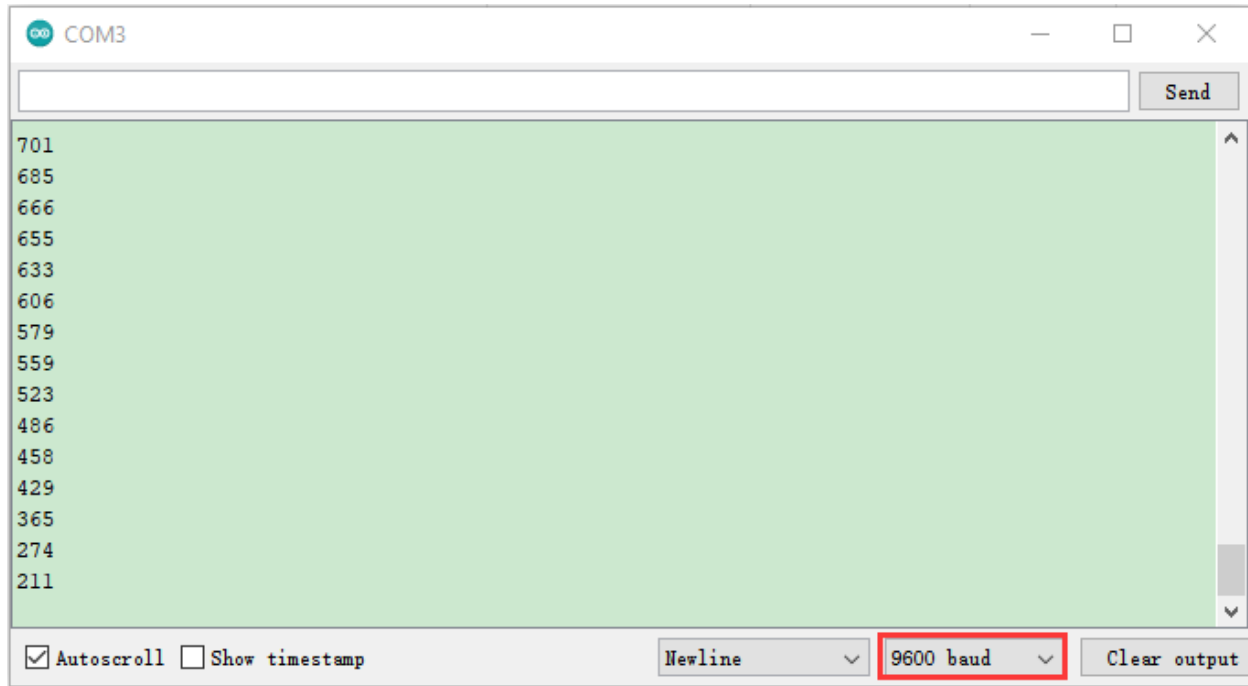
/*
Project 09.1:Read Photosensor Value
*/
int sensorPin = A6;    // select the input pin for the photocell
int sensorValue = 0;   // variable to store the value coming from the sensor
void setup() {
  Serial.begin(9600);
}
void loop() {
  sensorValue = analogRead(sensorPin); // read the value from the sensor:
  Serial.println(sensorValue); //Serial port prints the value of photoresistor
  delay(500);
}

```

## (6)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor and set baud to 9600.

When the light intensifies, the analog value will get increased; on the contrary, the analog value will get reduced.







## Project 9.2: Light Following Car

### (1)Description

We have learned the working principle of photoresistor, motor and speed regulation. In this experiment, we will use a photoresistor to detect the intensity of light as as to achieve the light following effect.

### (2)Components Required

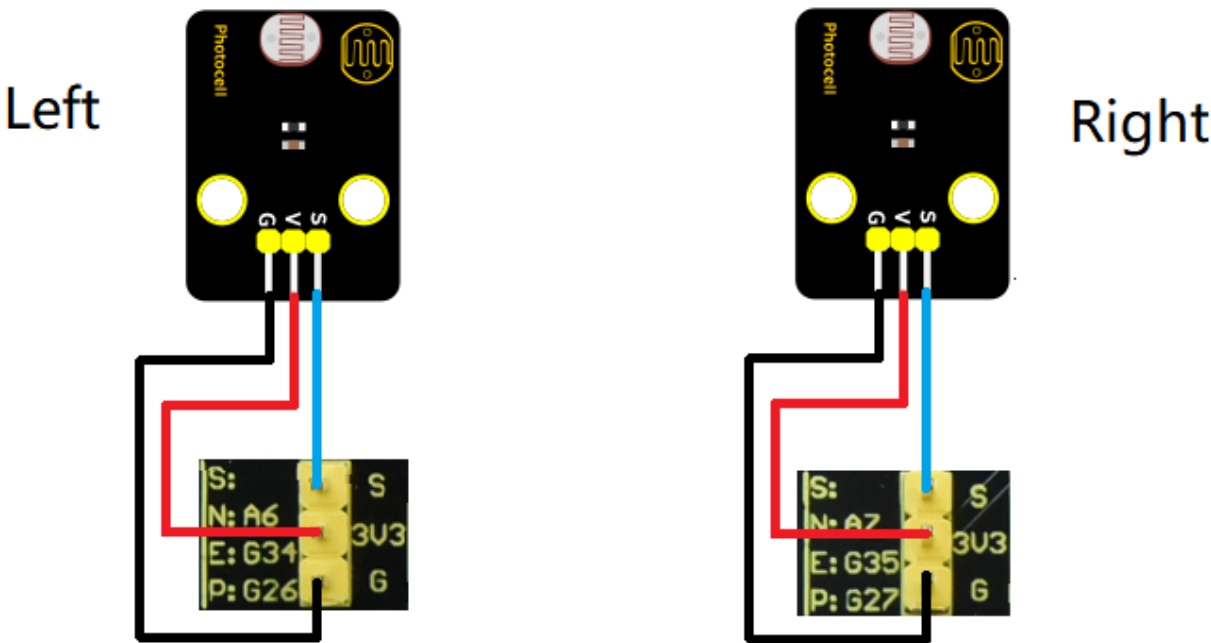
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

(3)Working Principle

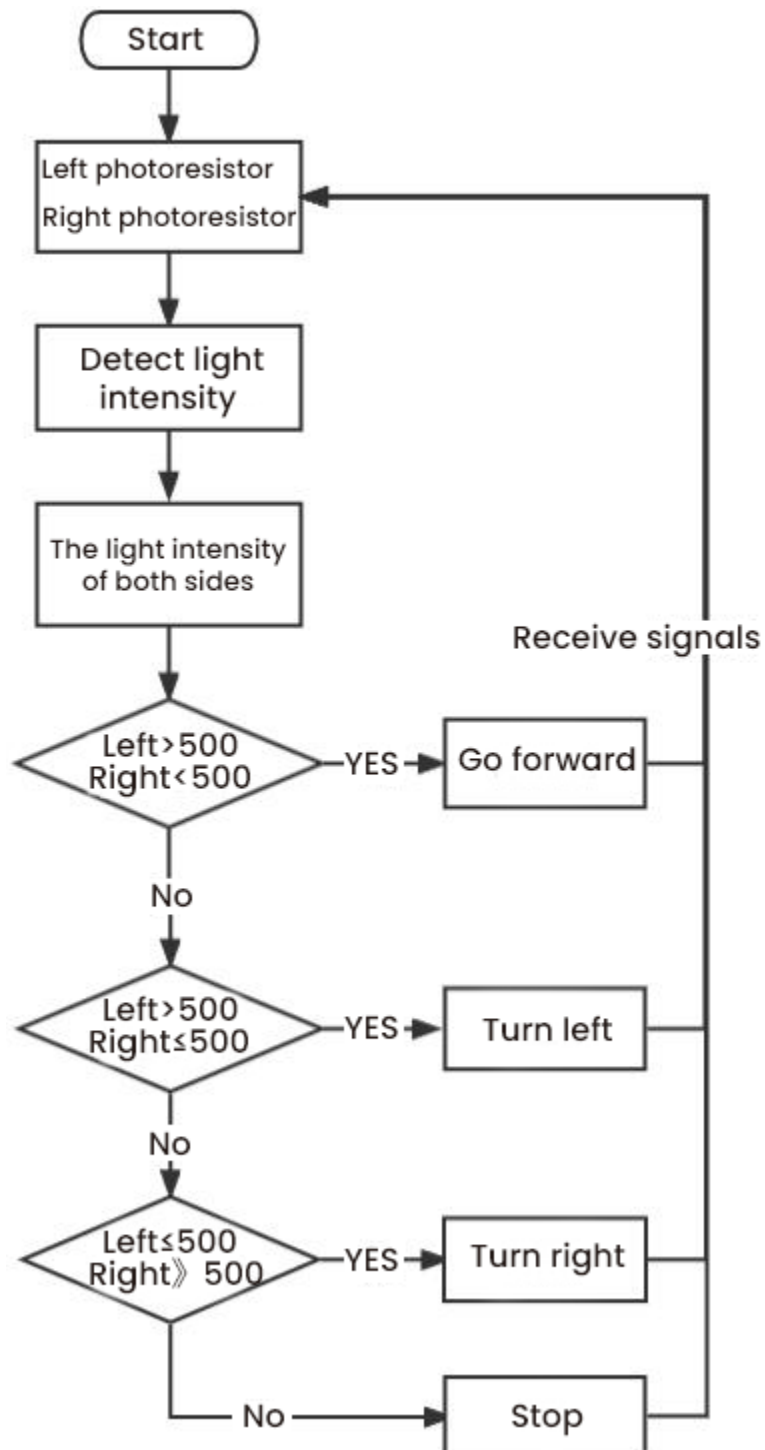
| Analog value of the left sensor | Analog value of the right sensor | Function      |
|---------------------------------|----------------------------------|---------------|
| >500                            | >500                             | Move forward  |
| >500                            | 500                              | Move to left  |
| 500                             | >500                             | Move to right |
| <500                            | <500                             | Stop          |

(4)Wiring up

| Left Photoresistor | PCB Board | Right photoresistor | PCB Board |
|--------------------|-----------|---------------------|-----------|
| G                  | G         | G                   | G         |
| V                  | V         | V                   | V         |
| S                  | SA6       | S                   | SA7       |



## (5)Flow Chart



## (6)Test Code

Left and right photoresistors are controlled by A6 and A7 of the Arduino Nano board.

```

/*
Project 09.2:Light Following Car
*/
const int light_L_Pin = A6;    //define the pins of the left photoresistor to A6
const int light_R_Pin = A7;    //define the pins of the right photoresistor to A7
const int left_ctrl = 4;       //define direction control pins of the left motor as D4
const int left_pwm = 6;        //define speed control pins of the left motor as D6
const int right_ctrl = 2;      //define the direction control pin of the right motor D2
const int right_pwm = 5;       //define the speed control pin of the right motor D5
int left_light;
int right_light;
const int servopin = 9;       //define the pin of the servo as D9
int myangle;
int pulsewidth;

void setup(){
  Serial.begin(9600);
  pinMode(light_L_Pin, INPUT); //Set pins of the left photoresistor to INPUT
  pinMode(light_R_Pin, INPUT); //Set pins of the right photoresistor to INPUT
  pinMode(left_ctrl,OUTPUT);   //Set the direction control pin of the left motor to OUTPUT
  pinMode(left_pwm,OUTPUT);    //Set the PWM control speed of the left motor to OUTPUT
  pinMode(right_ctrl,OUTPUT);  //Set the direction control pin of the right motor to OUTPUT
  pinMode(right_pwm,OUTPUT);   //Set the PWM control speed of the right motor to OUTPUT
  servopulse(servopin,90);     //set the initial angle of the servo to 90
  delay(300);
}

void loop(){
  left_light = analogRead(light_L_Pin); //read the value of the left photoresistor
  right_light = analogRead(light_R_Pin); //read the value of the right photoresistor
  Serial.print("left_light_value = ");
  Serial.println(left_light);
  Serial.print("right_light_value = ");
  Serial.println(right_light);
  if (left_light > 500 && right_light > 500) //range photoresistors can detect
  {
    Car_front(); //go forward
  }
  else if (left_light >500 && right_light <= 500) //range photoresistors can detect
  {
    Car_left(); //turn left
  }
  else if (left_light <= 500 && right_light > 500) //range photoresistors can detect
  {
    Car_right(); //turn right
  }
  else //if above conditions are not met
  {
    Car_Stop(); //stop
  }
}

```

(continues on next page)

(continued from previous page)

```

    }
}

void servopulse(int servopin,int myangle)//angles the servo run
{
    for(int i=0; i<20; i++)
    {
        pulsewidth = (myangle*11)+500;
        digitalWrite(servopin,HIGH);
        delayMicroseconds(pulsewidth);
        digitalWrite(servopin,LOW);
        delay(20-pulsewidth/1000);
    }
}

void Car_front()
{
    digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void Car_left()
{
    digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200;
}

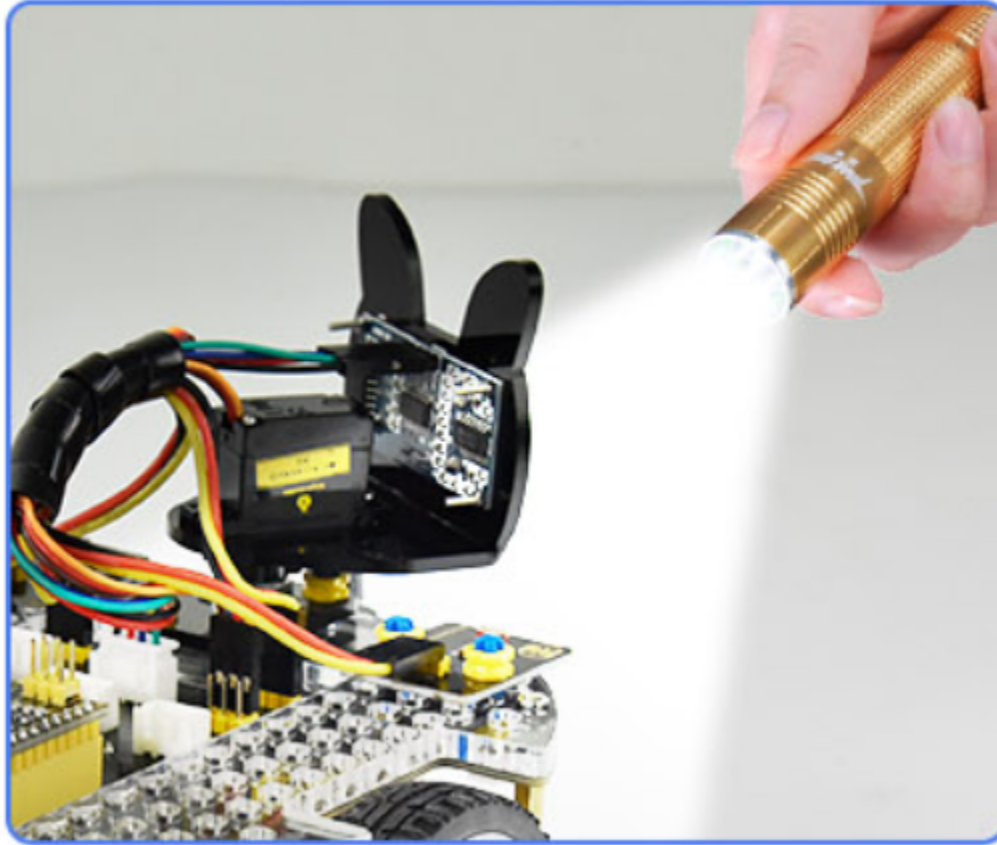
void Car_right()
{
    digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}

void Car_Stop()
{
    digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,0); //Set the PWM control speed of the left motor to 0
    digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
    analogWrite(right_pwm,0); //Set the PWM control speed of the right motor to 0
}

```

### (7)Test Result

Upload the test code to the Arduino Nano board, put batteries in the battery holder, turn the power switch to the ON end and power up. Then the car will follow the light to move.



### 8.3.10 Project 10: IR Remote Control

Infrared remote controls are everywhere in daily life. It is used to control various home appliances, such as TV, speakers, video recorders and satellite signal receivers.

The remote control is composed of an IR emitter, an IR receiver and a decoding MCU. In this project, we will make a IR remote control car.

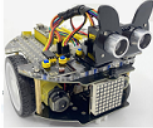

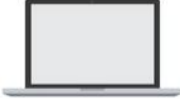


#### Project 10.1: IR Remote and Reception

##### (1)Description

In this experiment, we will combine the IR receiver and the IR remote control to read key values and show them on the serial monitor.



**(2)Components Required**

|   |   |   |  |   |
|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | Remote Control*1   | 18650 Battery*1   |
|  |  |  |  |  |

**(3)Knowledge****IR Remote Control**

It is a device with buttons. When the key is pressed, IR signals will be sent.

Infrared remote control technology is widely used, such as TVs, air conditioners and so on. And it can control air conditioners and TVs

The infrared remote control adopts NEC coding, and the signal period is 110ms.

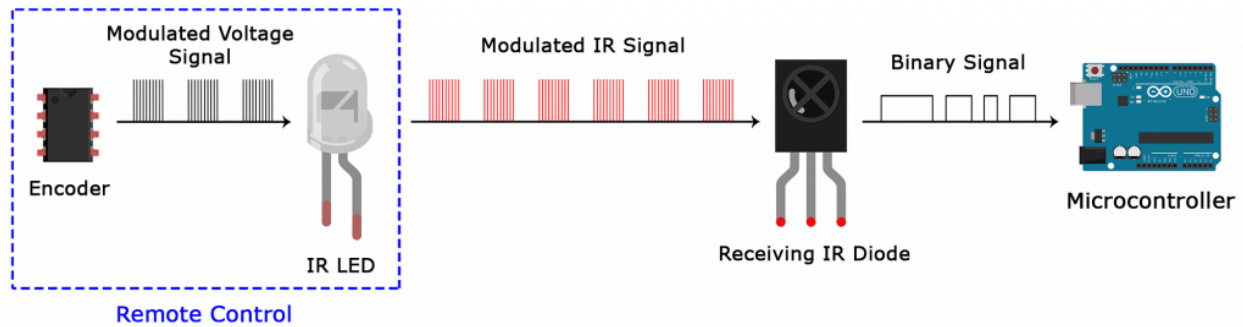
The remote control is shown below:



**Infrared (IR) receiver:**

It can receive infrared light and be used to detect the infrared signal emitted by the infrared remote control.

It can demodulate the received infrared light signal and convert it back to binary, and then transmit the information to the microcontroller.



### NEC Infrared communication protocol

#### NEC Protocol

To my knowledge the protocol I describe here was developed by NEC (Now Renesas). I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format.

I do admit that I don't know exactly who developed it. What I do know is that it was used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC.

This description was taken from my VCR's service manual. Those were the days, when service manuals were filled with useful information!

#### Features

8 bit address and 8 bit command length.

Extended mode available, doubling the address size.

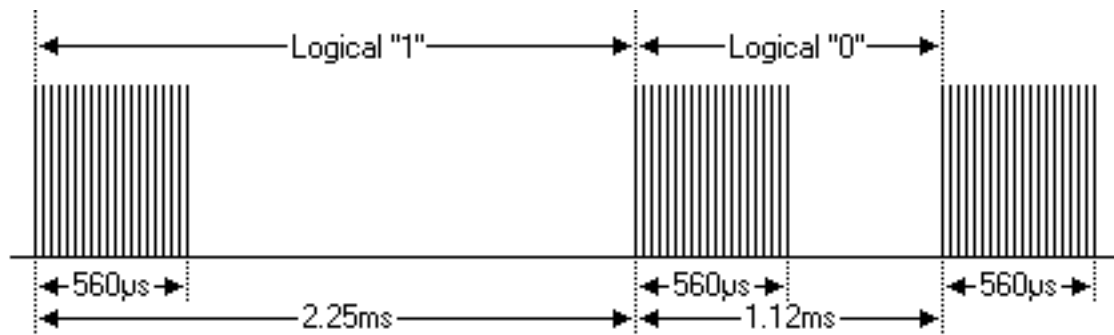
Address and command are transmitted twice for reliability.

Pulse distance modulation.

Carrier frequency of 38kHz.

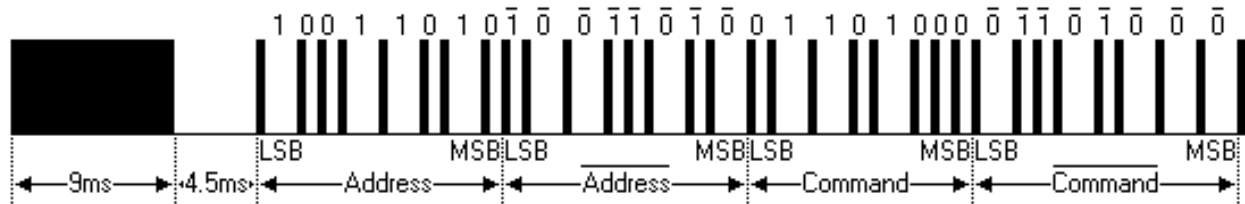
Bit time of 1.125ms or 2.25ms.

#### Modulation



The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

#### Protocol

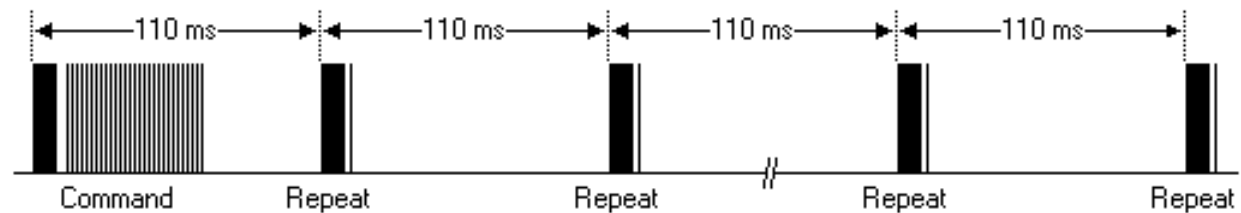


The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address \$59 and Command \$16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!

Keep in mind that one extra 560μs burst has to follow at the end of the message in order to be able to determine the value of the last bit.



A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560μs burst.

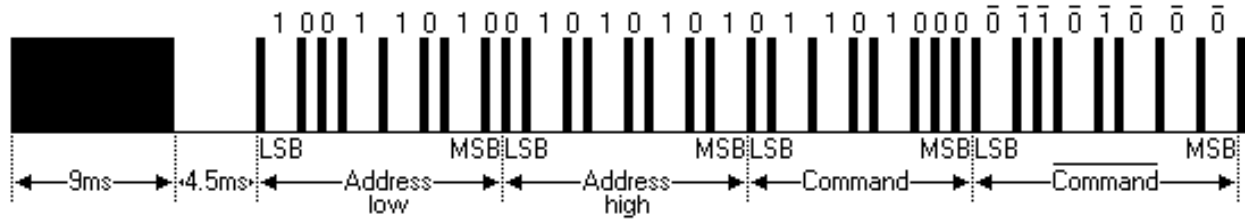


### Extended NEC protocol

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol.

By extending the address range this way the total message time is no longer constant. It now depends on the total number of 1's and 0's in the message. If you want to keep the total message time constant you'll have to make sure the number 1's in the address field is 8 (it automatically means that the number of 0's is also 8). This will reduce the maximum number of different addresses to just about 13000.

The command redundancy is still preserved. Therefore each address can still handle 256 different commands.



Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

#### (4)Test Code

The IR receiver on the PCB board is controlled by IO port(D12) of the Arduino Nano board.

```
/*
Project 10.1:Infrared remote and receiver
*/
#include <IRremote.h>
int RECV_PIN = 12;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // start receiving signals
}
void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    irrecv.resume(); // receive the next value
  }
  delay(100);
}
```

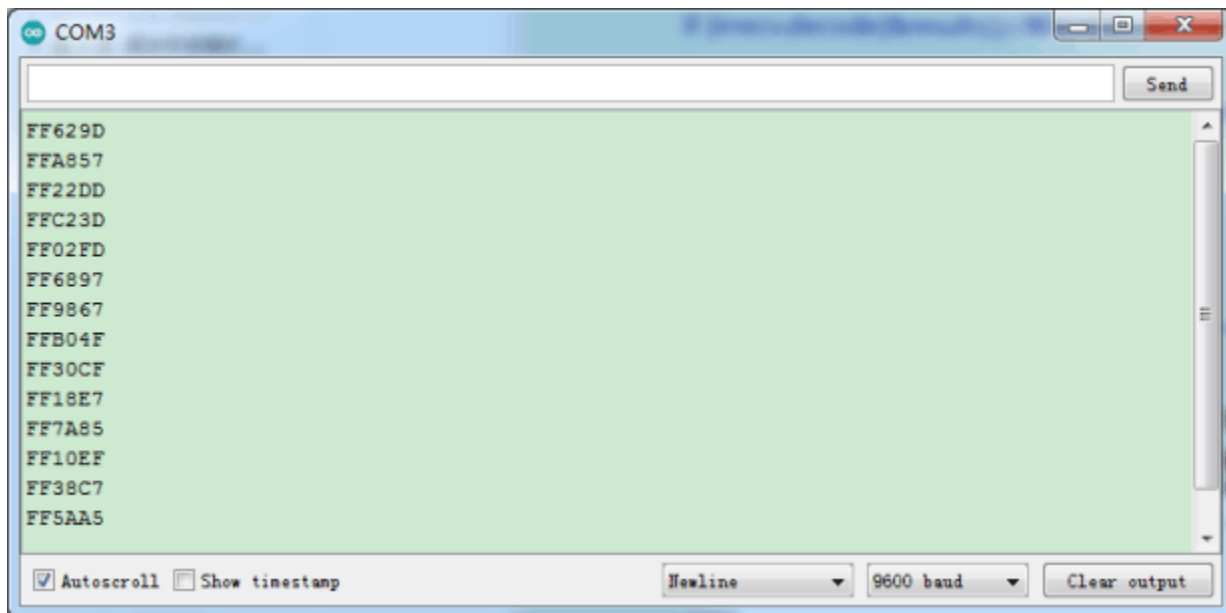
#### (5)Test Result:

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor and set to

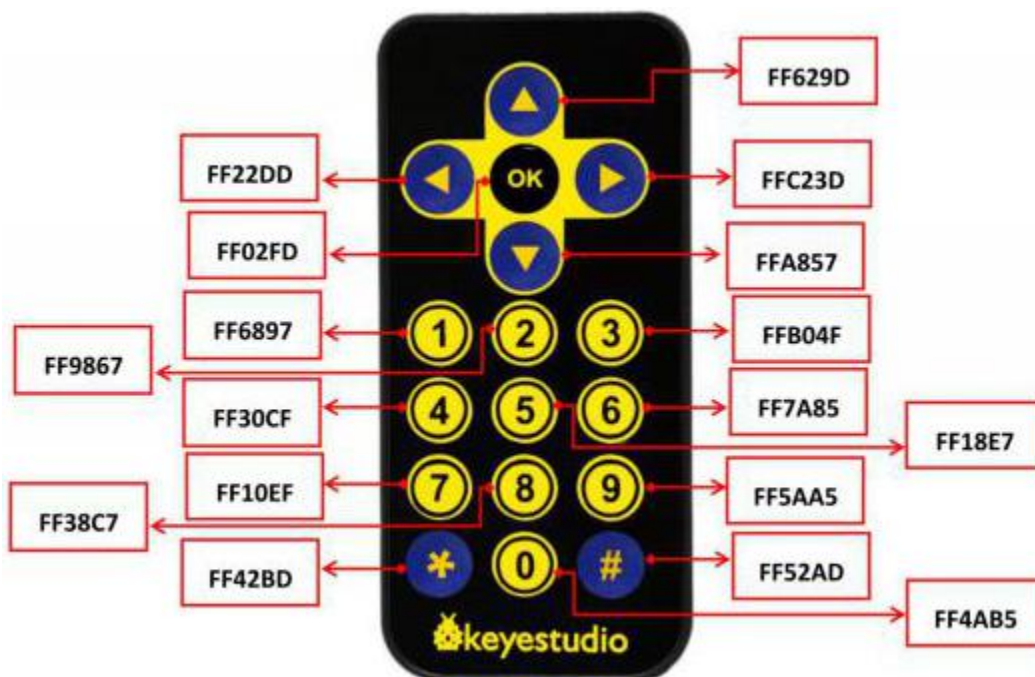


9600.

Press a key on the IR remote control, you will view a code on the serial monitor. If FFFFFFFF shows up, just ignore it.



Code of each key.

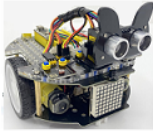

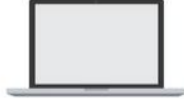




## Project 10.2: IR Remote Control Car

### (1)Description

In the above experiment, we have learned about the knowledge of the 8\*8 dot matrix display, the motor driver and speed regulation, the infrared receiver and the infrared remote control. In this experiment, we will use the infrared remote control and the infrared receiver to control the car.

### (2)Components Required

|   |   |   |  |   |
|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | Remote Control*1   | 18650 Battery*1   |
|  |  |  |  |  |

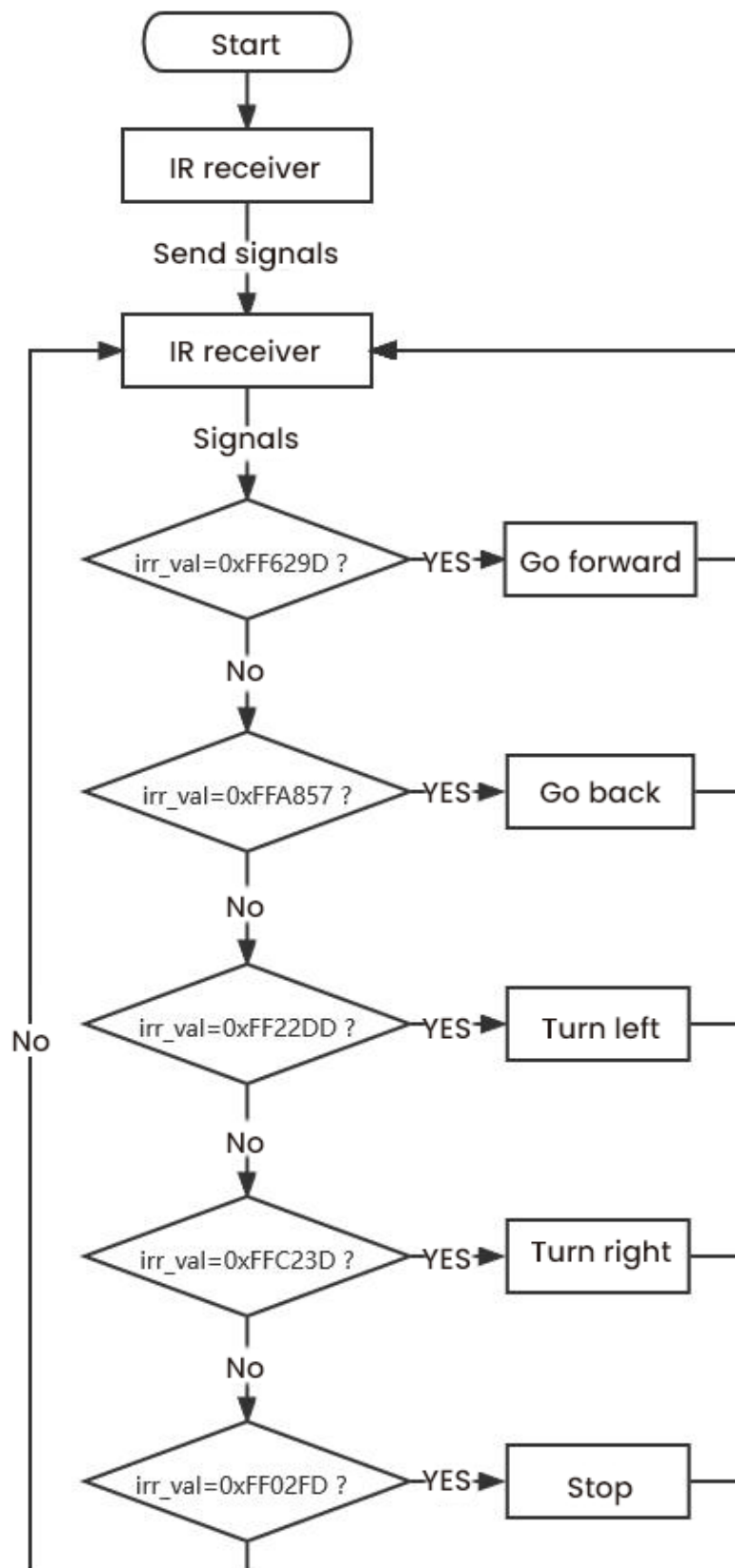
### (3)Working Principle

| Keys  | Keys Code | Functions                                 |
|---|-----------|---|
|  | FF629D    | Go forward<br>Display “forward” pattern   |
|  | FFA857    | Go back<br>Display “back” pattern         |
|  | FF22DD    | Turn left<br>Show“left” pattern           |
|  | FFC23D    | Turn right<br>Show“right turning” pattern |
|  | FF02FD    | stop<br>show“stop” pattern                |





## (4)Flow Chart



## (5)Test Code

```

/*
Project 10.2:Infrared remote control car
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);//define pins of the dot matrix display as A4, and A5
//Array, used to store pattern data, which can be calculated by yourself or obtained
↳from the touch tool
uint8_t matrix_front[8]={0x18,0x24,0x42,0x99,0x24,0x42,0x81,0x00};
uint8_t matrix_back[8]={0x00,0x81,0x42,0x24,0x99,0x42,0x24,0x18};
uint8_t matrix_left[8]={0x12,0x24,0x48,0x90,0x90,0x48,0x24,0x12};
uint8_t matrix_right[8]={0x48,0x24,0x12,0x09,0x09,0x12,0x24,0x48};
uint8_t matrix_stop[8]={0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18};
uint8_t LEDArray[8];
const int left_ctrl = 4;//define the control pin of the left motor as D4
const int left_pwm = 6;//define the control pin of the left motor as D6
const int right_ctrl = 2;//define the control pin of the right motor as D2
const int right_pwm = 5;//define the control pin of the right motor as D5
#include <IRremote.h>//IR remote function library
int RECV_PIN = 12;//define the pin of the IR reception as D12
IRrecv irrecv(RECV_PIN);
long irr_val;
decode_results results;
const int servopin = 9;//define the pin of the servo as D9
int myangle;
int pulsewidth;

void setup()
{
  Serial.begin(9600);//open serial port and set baud rate 9600
  pinMode(left_ctrl,OUTPUT);//set the direction control pin of the left motor to OUTPUT
  pinMode(left_pwm,OUTPUT);//set the pwm control pin of the left motor to OUTPUT
  pinMode(right_ctrl,OUTPUT);//set the control pin of the right motor to OUTPUT
  pinMode(right_pwm,OUTPUT);//set the pwm control pin of the right motor to OUTPUT
  pinMode(RECV_PIN,INPUT);//set the pin of the IR receiver to INPUT
  // In case the interrupt driver crashes on setup, give a clue
  // to the user what's going on.
  Serial.println("Enabling IRin");
  irrecv.enableIRIn(); // start receiving signals
  Serial.println("Enabled IRin");
  myMatrix.begin(112);
  myMatrix.clear();
  myMatrix.writeDisplay();
  servopulse(servopin,90);//set the initial angle of the servo to 90
  delay(300);
}

void loop()
{
  if (irrecv.decode(&results))
  {
    irr_val = results.value;

```

(continues on next page)

(continued from previous page)

```

Serial.println(irr_val, HEX);//the serial prints the IR remote signals
switch(irr_val)
{
    case 0xFF629D :
        car_front();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_front);
        break;
    case 0xFFA857 :
        car_back();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_back);
        break;
    case 0xFF22DD :
        car_left();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_left);
        break;
    case 0xFFC23D :
        car_right();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_right);
        break;
    case 0xFF02FD :
        car_Stop();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_stop);
        break;
}
    irrecv.resume(); // receive the next value
}
}

void servopulse(int servopin,int myangle)//the servo runs angles
{
    for(int i=0; i<20; i++)
    {
        pulsewidth = (myangle*11)+500;
        digitalWrite(servopin,HIGH);
        delayMicroseconds(pulsewidth);
        digitalWrite(servopin,LOW);
        delay(20-pulsewidth/1000);
    }
}

void car_front()//define the state of going front
{

```

(continues on next page)

(continued from previous page)

```

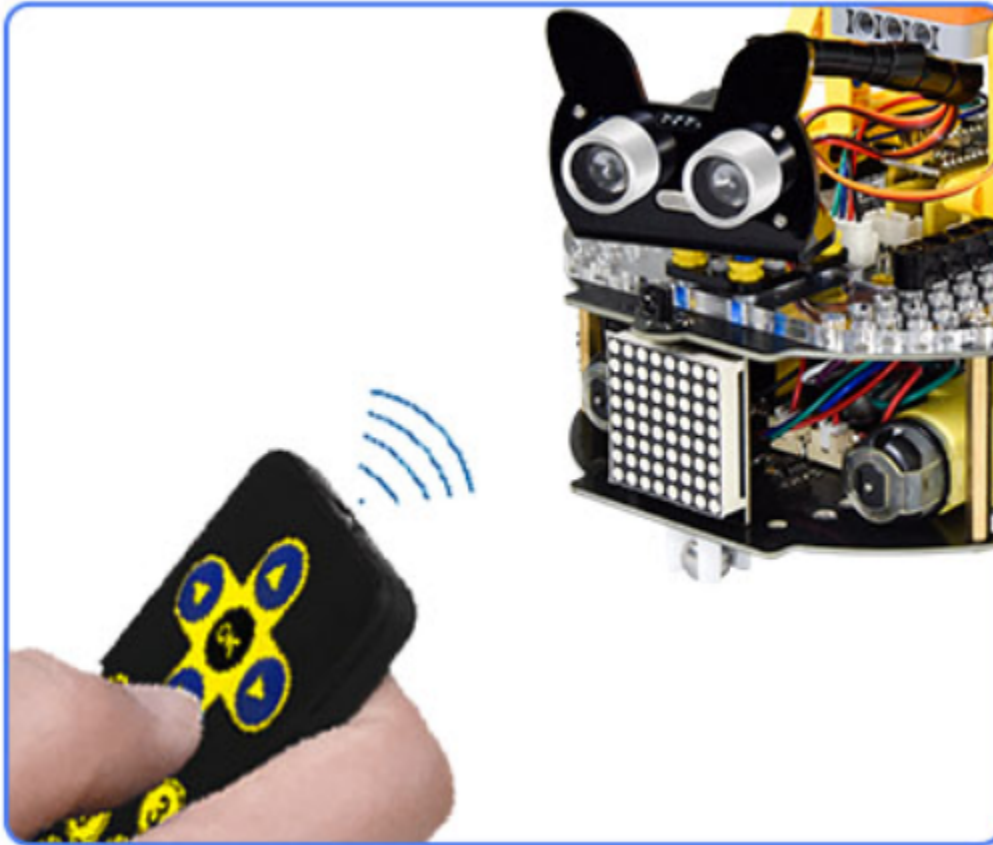
digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}
void car_back()//define the state of going back
{
    digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
    analogWrite(left_pwm,50); //Set the PWM control speed of the left motor to 50
    digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
    analogWrite(right_pwm,50); //Set the PWM control speed of the right motor to 50
}
void car_left()//define the state of turning left
{
    digitalWrite(left_ctrl,HIGH); //set control pins of the left motor to HIGH
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,LOW); //set control pins of the right motor to LOW
    analogWrite(right_pwm,200); //Set the PWM control speed of the right motor to 200
}
void car_right()//define the state of turning right
{
    digitalWrite(left_ctrl,LOW); //Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,200); //Set the PWM control speed of the left motor to 200
    digitalWrite(right_ctrl,HIGH); //Set direction control pins of the right motor to HIGH
    analogWrite(right_pwm,200); //set the PWM control speed of the right motor to 200
}
void car_Stop()//define the state of stop
{
    digitalWrite(left_ctrl,LOW);//Set direction control pins of the left motor to LOW
    analogWrite(left_pwm,0);//Set the PWM control speed of the left motor to 0
    digitalWrite(right_ctrl,LOW);//set control pins of the right motor to LOW
    analogWrite(right_pwm,0);//Set the PWM control speed of the right motor to 0
}

//
void matrix_display(unsigned char matrix_value[])
{
    for(int i=0; i<8; i++)
    {
        LEDArray[i]=matrix_value[i];
        for(int j=7; j>=0; j--)
        {
            if((LEDArray[i]&0x01)>0)
                myMatrix.drawPixel(j, i,1);
            LEDArray[i] = LEDArray[i]>>1;
        }
    }
    myMatrix.writeDisplay();
}

```

### (6)Test Result

Upload the test code to the Arduino Nano motherboard, install batteries, turn the power switch to the ON end, power up and press a key of the IR remote control. Then the car will make the corresponding movement.



### 8.3.11 Project 11: WIFI Control

In this lesson, we control the car through app. The Beetlebot APP sends commanders to the WIFI ESP-01 module then transfers to it to the microcontroller. By doing this, the car can perform different functions.







#### Project 11.1: WIFI Test

##### (1)Description

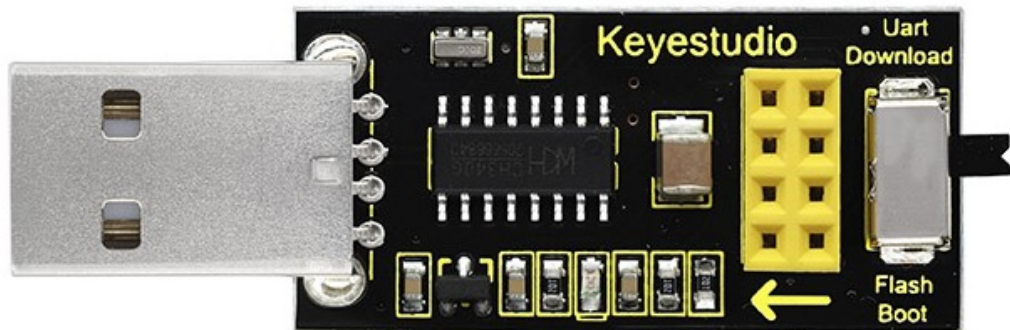
The ESP8266 serial WiFi ESP-01 module is an ultra-low-power UART-WiFi transparent transmission module and designed for mobile devices and IoT applications.

It can achieve networking functions by connecting devices to Wifi internet.

(2)Components Required:

|   |   |   |   |  |   |
|---|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | WiFi module*1   | USB Serial ESP-01S WIFI Expansion Module *1  | 18650 Battery*1   |
|  |  |  |  |  |  |

(3)Knowledge



USB to ESP-01S WiFi module serial shield:

It is suitable for the ESP-01S WiFi module. Turn the DIP switch on the USB to ESP-01S WiFi module serial Expansion Board to Flash Boot, and plug into computer’s USB port. You can use serial debugging tool to test the AT command.

Turn the DIP switch on the USB to ESP-01S WiFi module serial expansion board to the UartDownload, ESP-01 module is at download mode. You can download the firmware to ESP-01 module using AT firmware.

ESP8266 serial WiFi ESP-01:



ESP8266 serial WiFi ESP-01 is an ultra-low-power UART-WiFi transparent transmission module. It can be widely used in smart grids, intelligent transportation, smart furniture, handheld devices, industrial control and other fields.

Features:

Support wireless 802.11 b/g/n standards  
 Support STA/AP/STA+AP three modes of operation  
 Built-in TCP/IP protocol stack to support multi-channel TCP Client connections  
 Supports many Socket AT commands  
 Supports UART / GPIO data communication interface  
 Supports Smart Link smart networking function  
 Supports remote firmware upgrades(OTA)  
 Built-in 32-bit MCU, can also be used as an application processor  
 Ultra-low-power and highly integrated Wi-Fi chip for battery-powered applications  
 Working temperature range: -40 ° C to + 125 ° C  
 3.3V single power supply

#### (4)Functions

##### A. Main functions

The main functions that can be achieved by ESP8266 include: serial port transparent transmission , PWM regulation, GPIO control.

※Serial port transparent transmission: The transmission is reliable with a maximum transmission rate of 460800bps.

※PWM regulation: Adjusting lights and tricolor LED, motor speed control, etc.

※GPIO control: Control switch, relay, etc.

##### B.Working modes

The ESP8266 module supports three operating modes, STA/AP/STA+AP.

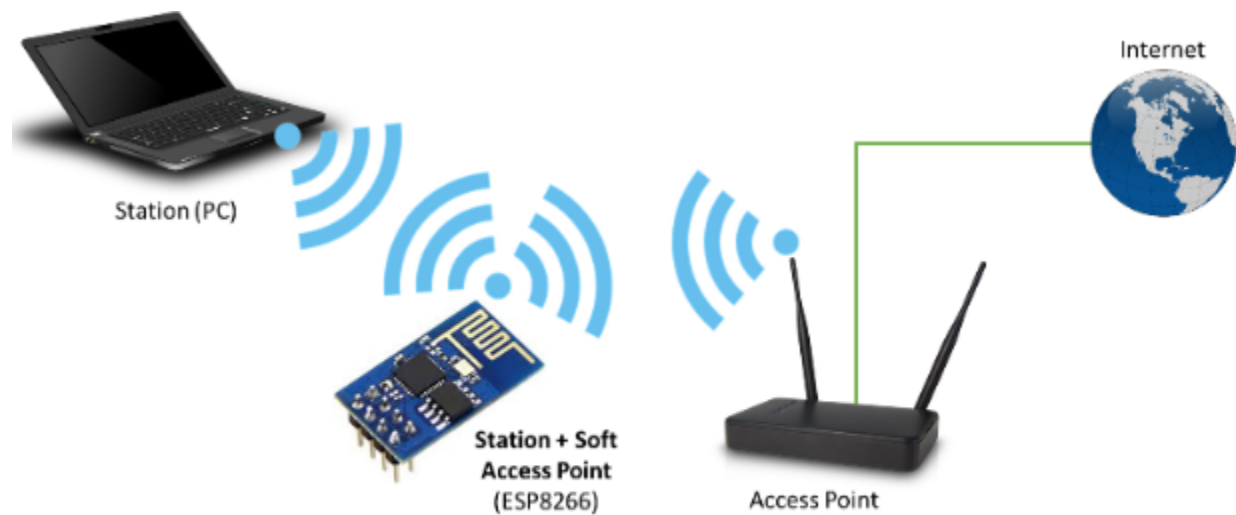
STA mode: The ESP8266 module can access to the Internet through a router, so the mobile phone or computer can remotely control the device through the Internet.



ESP8266 operating in the **Station** mode

AP mode: ESP8266 module, as a hotspot, allows the direct communication with the module and cellphones/computers, achieving wireless control of the local area network (LAN).

STA+AP mode: two modes coexist, that is, the Internet can achieve free switch.



---

### ESP8266 operating in the **Station + Soft Access Point Mode** mode

---

#### Applications

Serial CH340 to Wi-Fi

Industrial transparent transmission DTU

Wi-Fi remote monitoring/control

Toy industry

Color LED control

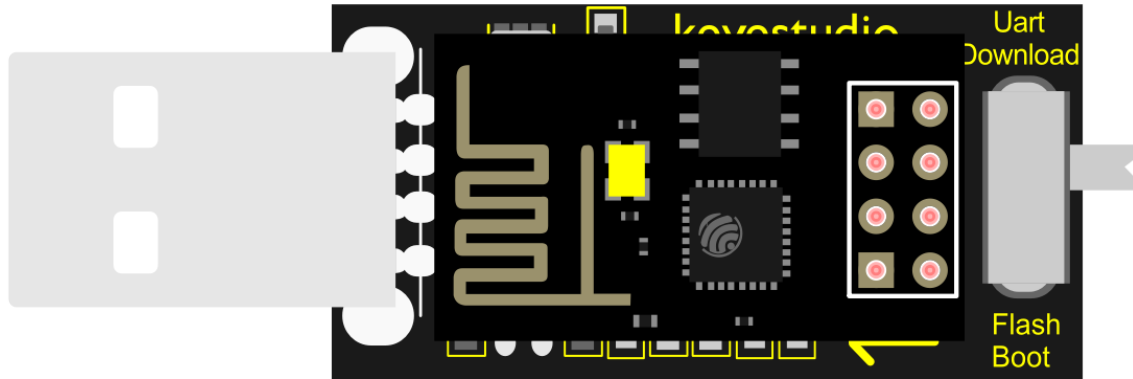
Integrated management of fire protection and security intelligence

Smart card terminals, wireless POS machines, Wi-Fi cameras, handheld devices, etc



**(5) Insert the Wifi serial port expansion board into the USB port of your PC:**

Insert the ESP8266 serial WIFI ESP-01 module into the USB to ESP-01S WIFI expansion board.

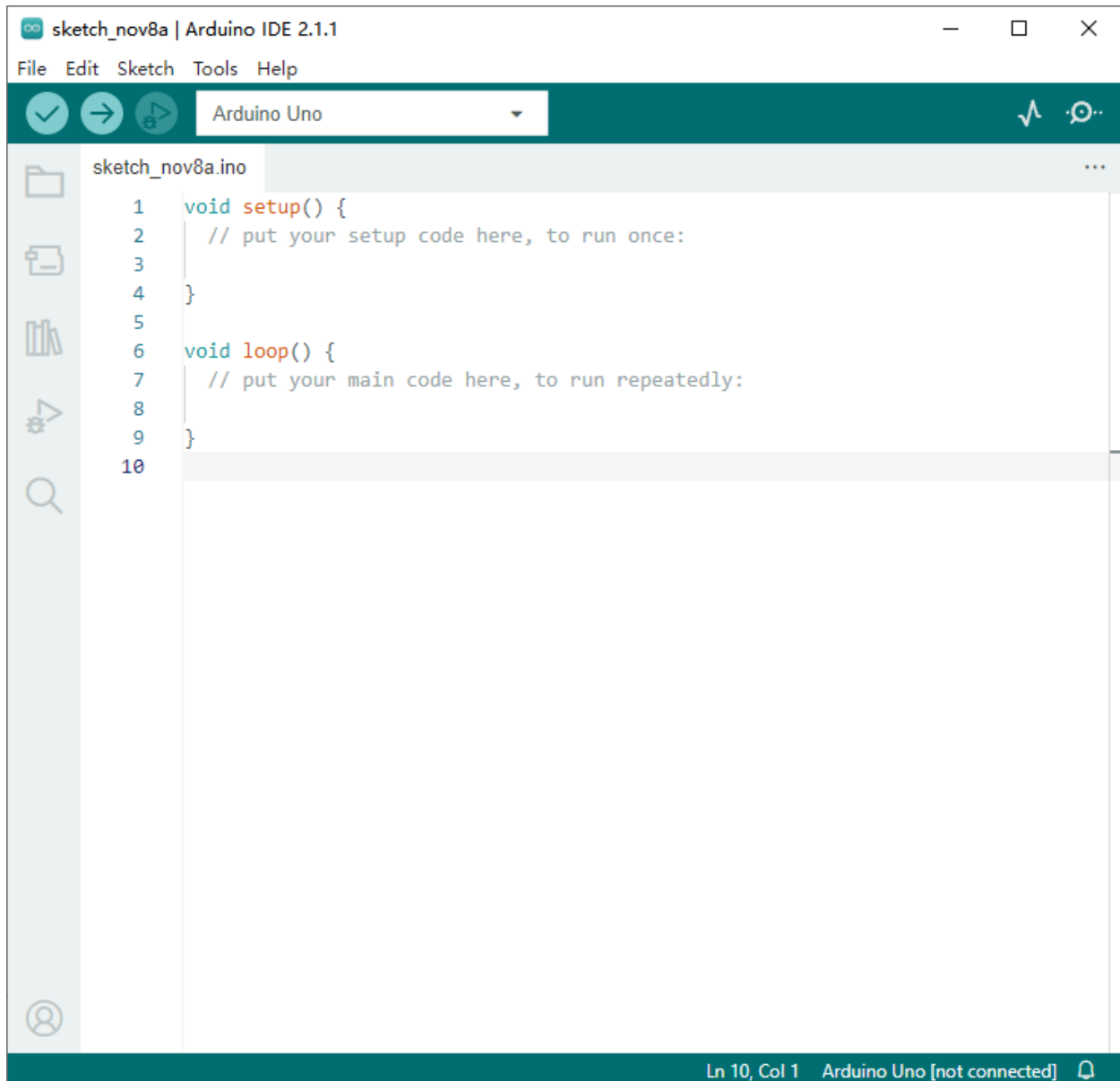


Turn the DIP switch of the USB to ESP-01S WIFI expansion board to UartDownload end and plug it to the USB port of your computer

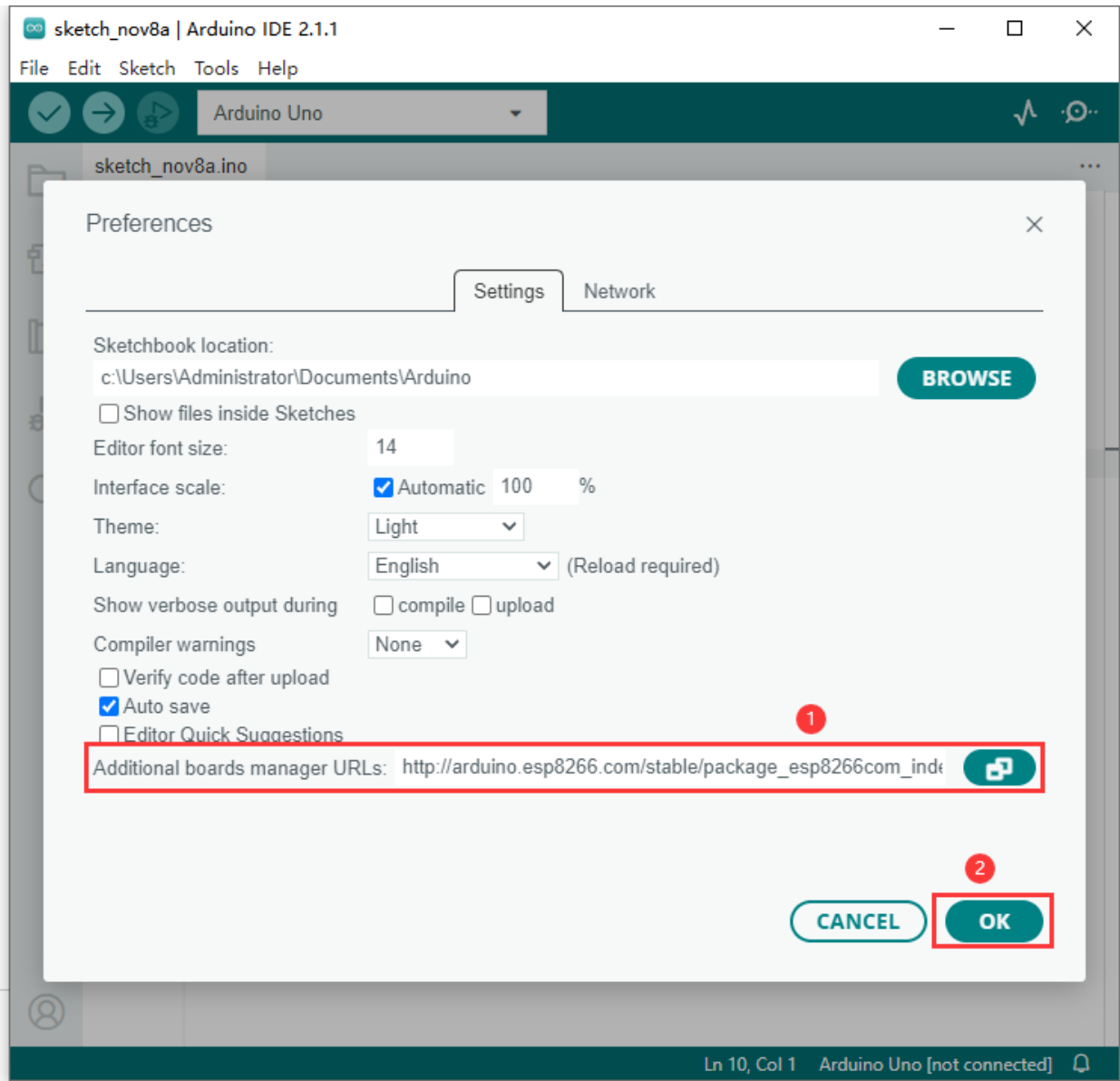


**(6) Set up the Esp8266 development environment:**

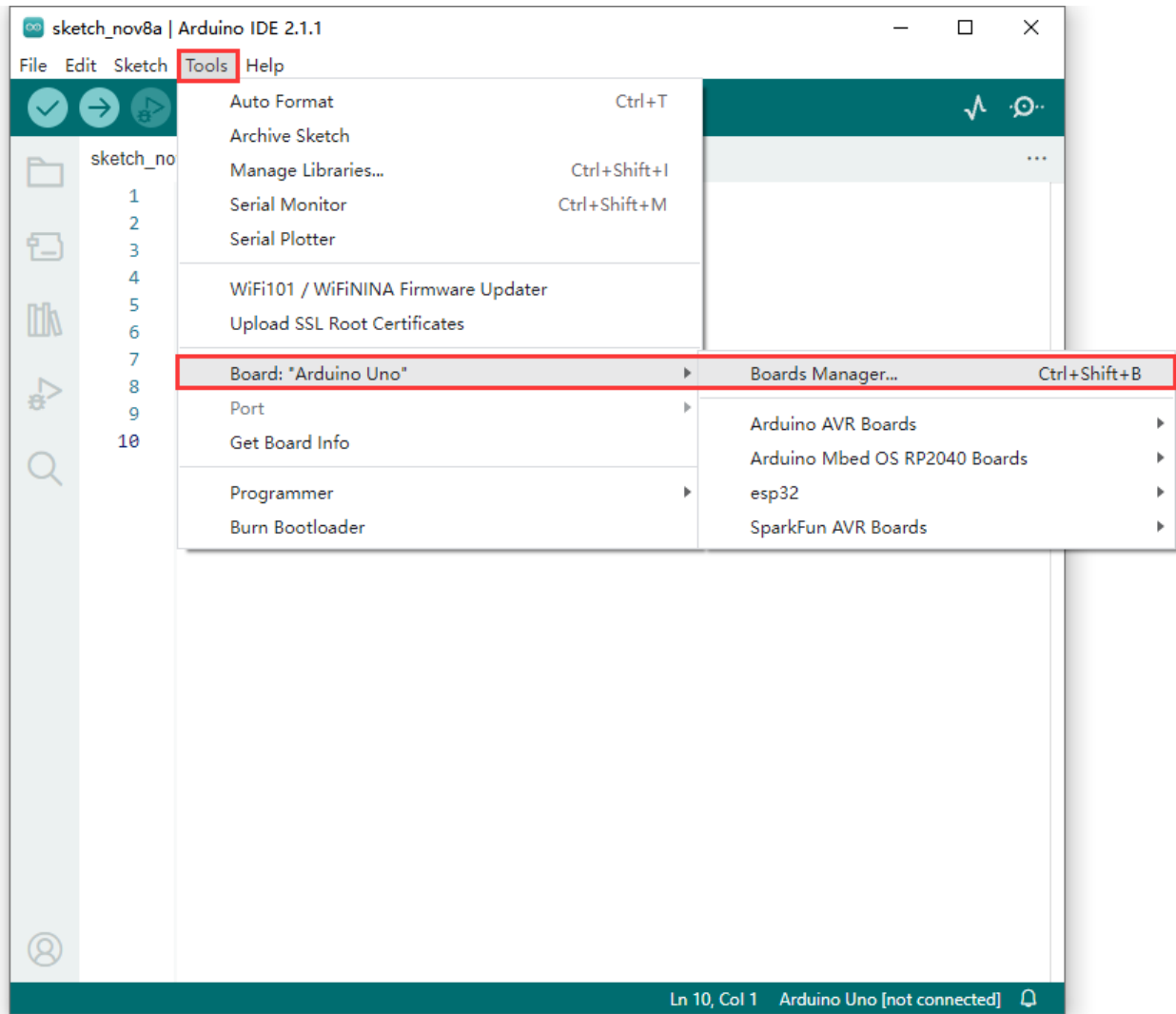
Insert the ESP8266 serial WiFi ESP-01 module into the USB to ESP-01S WiFi expansion board correctly, and then plug it into the USB port of the computer. Click to enter the arduino 2.1.1 folder (you can also use the latest version) to enter the IDE interface.

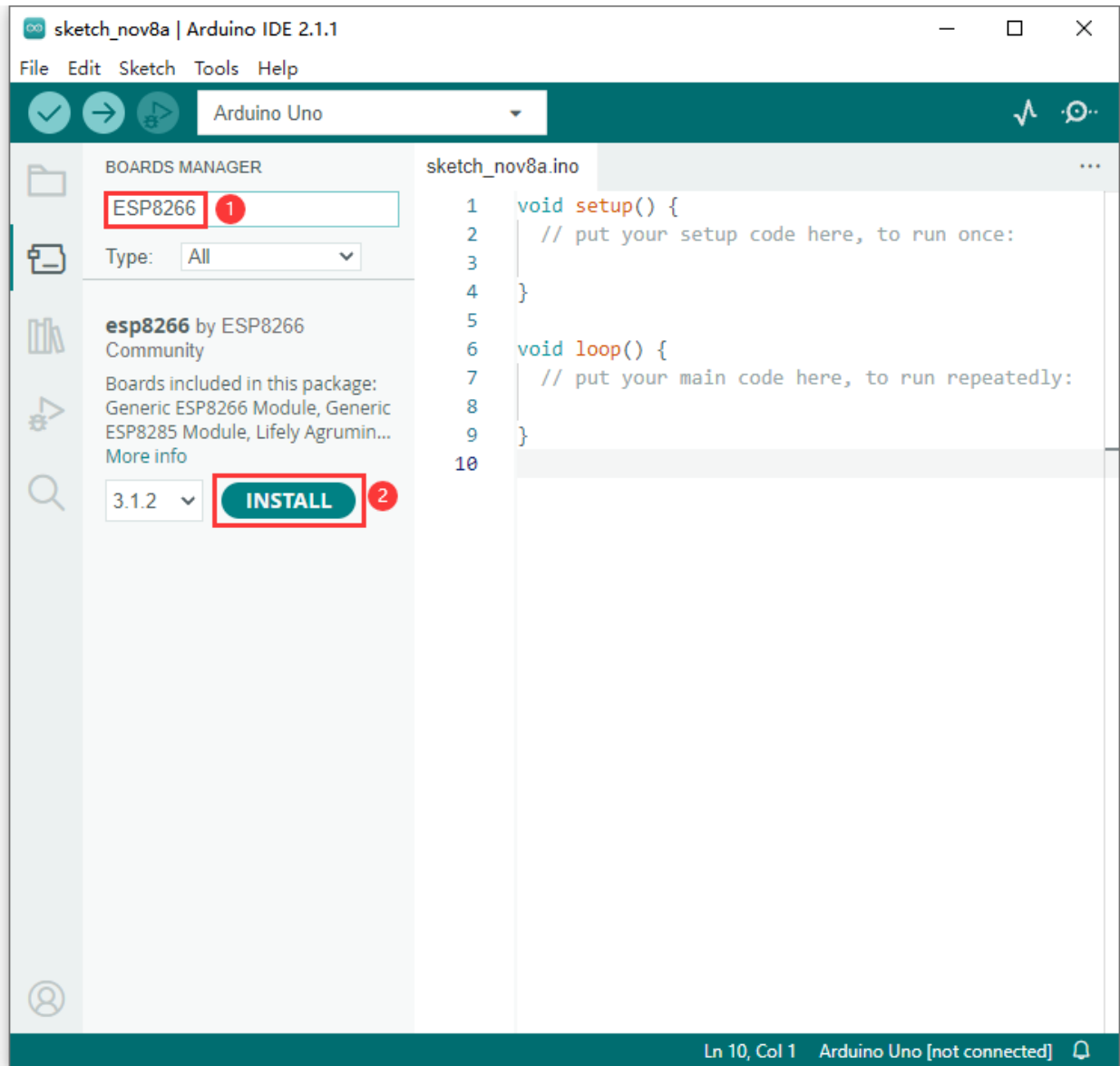
**Download and install from the Arduino IDE**

Click File → Preferences, copy and paste this address ([http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json)) in the "Additional Boards Manager URLs:", then click "OK" to save this address.

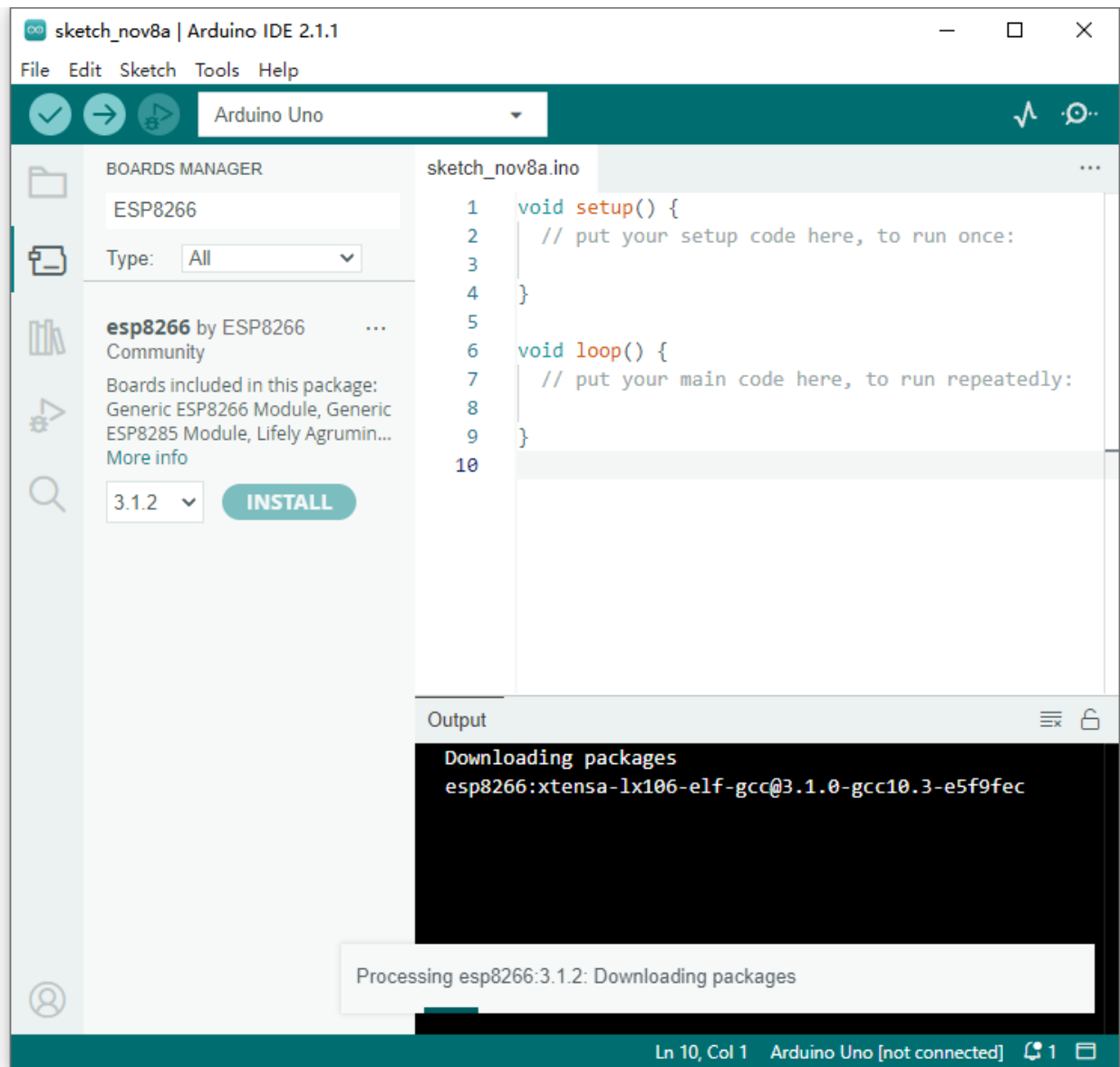


Click “Tools” → “Board:”, then click on “Board Manager...” to enter the “Board Manager” page, type “ESP8266”. Then select the latest version to install.



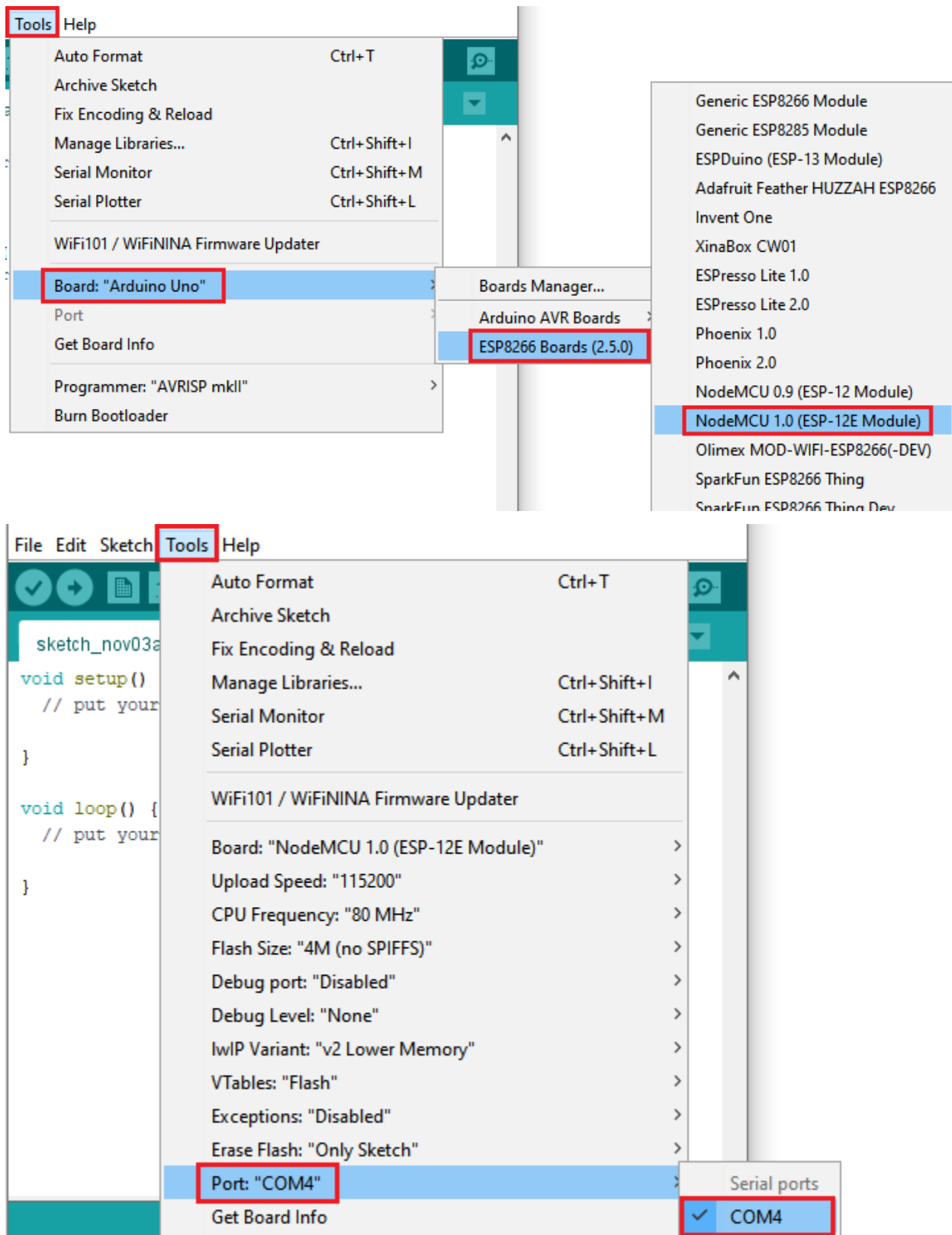


Click “Install” to start to install the relevant plug-ins. (If downloading unsuccessfully, just click “Install” again.)



However, due to network reasons, most users may not be able to search esp8266 by esp8266 Community, so, we recommend you to install ESP8266 by tools.

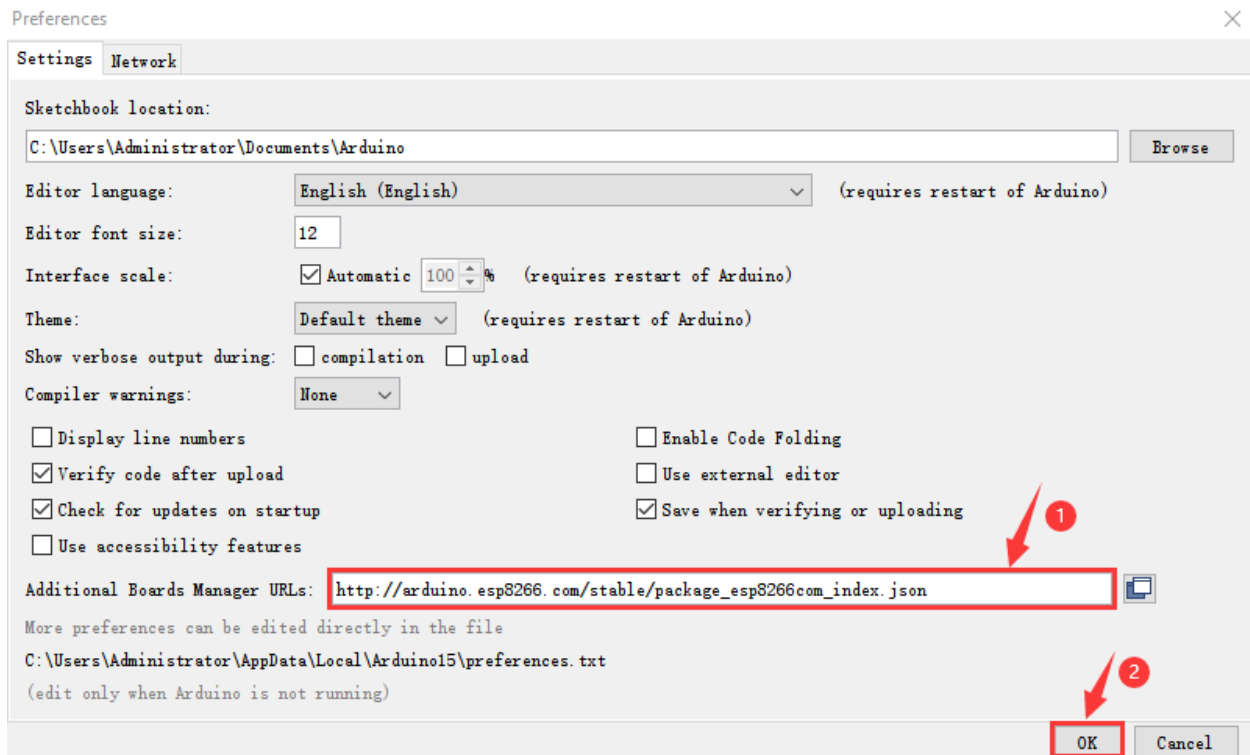
Close the page, and then click “Tools” → “Board:”, you can view different models of ESP8266 development boards. Select the corresponding ESP8266 development board model and COM port to program ESP8266.



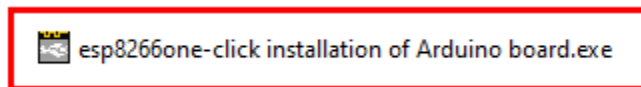
### Install ESP8266 by tools (Recommended)

Click File > Preferences, copy the link [http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) to "Addi-

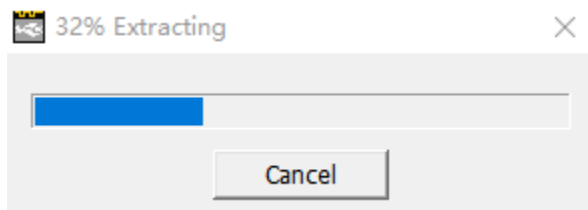
tional Boards Manager URLs:"box.



Use“ESP8266 one-click installation of Arduino board version 2.5.0.exe”to install ESP8266. This method is recommended.

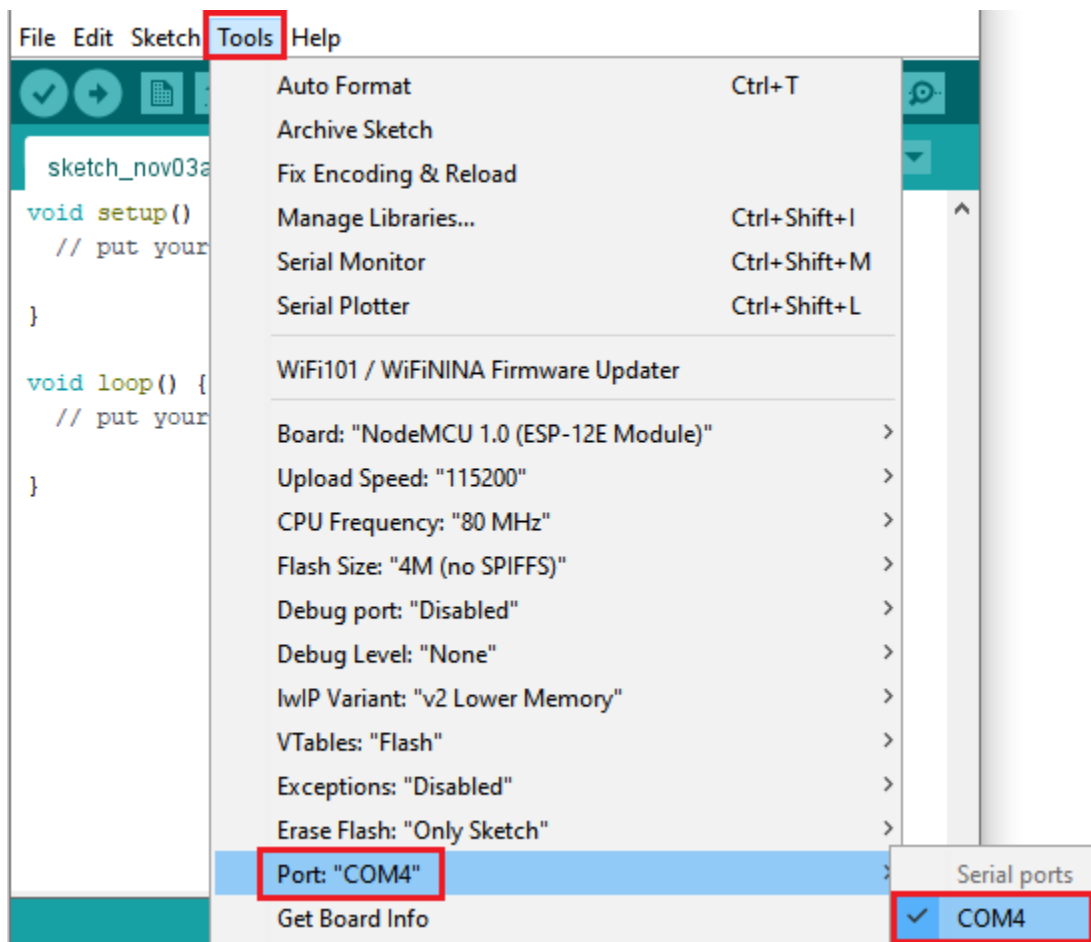
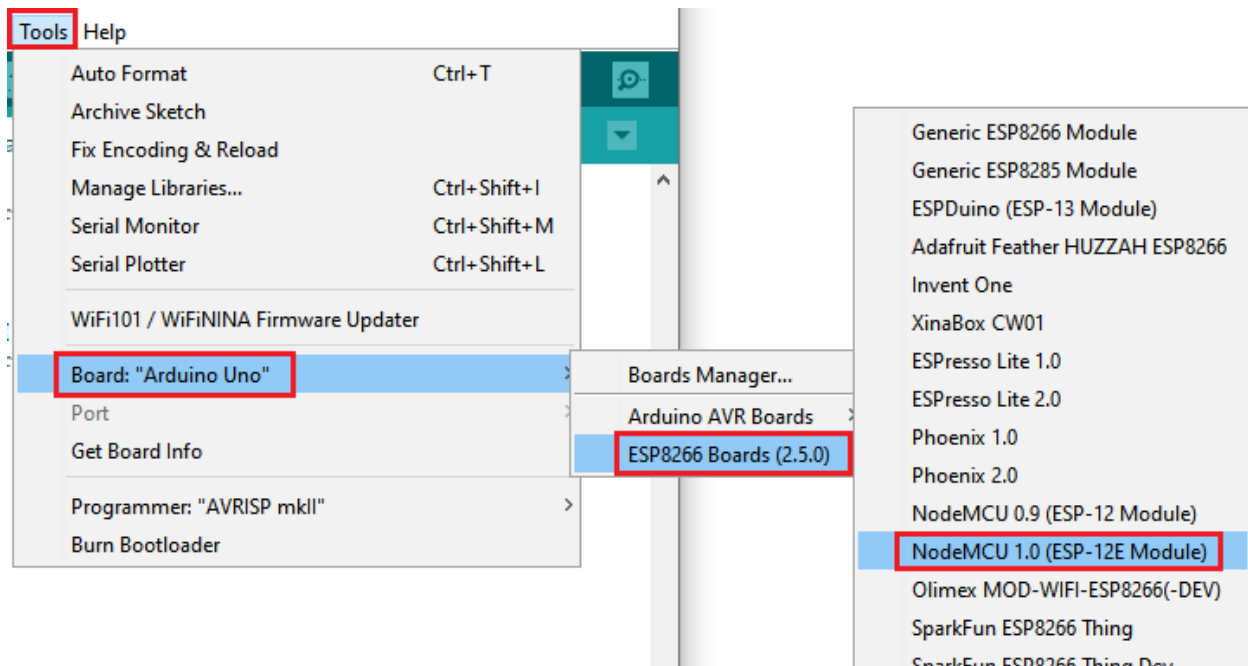


Double-click“ESP8266 one-click installation of Arduino board version 2.5.0.exe”,then the installation is finished.



After the above tool is installed, restart the Arduino IDE software and click on the Arduino menu bar“Tools”→“Board”, you can view different models of ESP8266 development boards in it. Select the corresponding ESP8266 development board model and COM port to program ESP8266.





## (6)Test Code

Note: After opening the IDE, set the board type and COM port first. If you don't have WiFi at home, you can turn your phone hotspot on to enable shared WiFi.

The UTXD pin of the WIFI ESP-01 module is controlled by the IO port RX (0) of the Arduino Nano motherboard, and the URXD pin is controlled by the IO port TX(1) of the Arduino Nano board.

Note: you need to change the account and password of Wifi in the code into yours.

```
#ifndef STASSID
#define STASSID "ChinaNet-2.4G-0DF0" //the name of user's Wifi
#define STAPSK "ChinaNet@233" //the password of the user's wifi
#endif
```

```
/*
Project 10.1 WIFI test
*/
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiClient.h>

#ifndef STASSID
// #define STASSID "your-ssid"
// #define STAPSK "your-password"
#define STASSID "ChinaNet-2.4G-0DF0" //the name of user's wifi
#define STAPSK "ChinaNet@233" //the password of user's wifi
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

// TCP server at port 80 will response the HTTP requirement
WiFiServer server(80);

void setup(void) {
  Serial.begin(115200);

  // connect WiFi
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.println("");

  // wait connection
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
```

(continues on next page)

(continued from previous page)

```

// set the mDNS responder::
// - in this example. the first parameter is domain name
//   The fully qualified domain name is "esp8266.local"
// - the second parameter is IP address
//   send the IP address via WiFi
if (!MDNS.begin("esp8266")) {
    Serial.println("Error setting up MDNS responder!");
    while (1) {
        delay(1000);
    }
}
Serial.println("mDNS responder started");

// activate TCP (HTTP) server
server.begin();
Serial.println("TCP server started");

// add the server to MDNS-SD
MDNS.addService("http", "tcp", 80);
}

void loop(void) {

    MDNS.update();

    // check the client side is connected or not
    WiFiClient client = server.available();
    if (!client) {
        return;
    }
    Serial.println("");
    Serial.println("New client");

    // wait the effective data from the client side
    while (client.connected() && !client.available()) {
        delay(1);
    }

    // read the first row of HTTP requirement
    String req = client.readStringUntil('\r');

    // the first row of the HTTP requirement is shown below: "GET /path HTTP/1.1"
    // Retrieve the "/path" part by finding the spaces
    int addr_start = req.indexOf(' ');
    int addr_end = req.indexOf(' ', addr_start + 1);
    if (addr_start == -1 || addr_end == -1) {
        Serial.print("Invalid request: ");
        Serial.println(req);
        return;
    }
    req = req.substring(addr_start + 1, addr_end);
    Serial.print("Request: ");

```

(continues on next page)

(continued from previous page)

```

Serial.println(req);
client.flush();

String s;
if (req == "/") {
  IPAddress ip = WiFi.localIP();
  String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +
  ↪String(ip[3]);
  s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
  ↪Hello from ESP8266 at ";
  s += ipStr;
  s += "</html>\r\n\r\n";
  Serial.println("Sending 200");
} else {
  s = "HTTP/1.1 404 Not Found\r\n\r\n";
  Serial.println("Sending 404");
}
client.print(s);

Serial.println("Done with client");
}

```

## (7)Test Result

After the account and password of Wifi is changed, turn the DIP switch of the USB to ESP-01S WIFI module to the Uart Download end and plug ESP-01S WIFI module into the USB port of your PC.

Set board type and COM port.

And upload the ESP8266 code to the ESP8266 serial WIFI ESP-01 module.

If the test code is not uploaded successfully, check the board type and the COM port first, then unplug the ESP-01S WIFI module and restart it.

```

Project_10.1_WIFI_Test | Arduino 1.8.16
File Edit Sketch Tools Help

Project_10.1_WIFI_Test

//*****
/*
Project 10.1 WIFI test
*/
#include <ESP8266WiFi.h>
#include <ESP8266DNS.h>
#include <WiFiClient.h>

#ifndef STASSID
//#define STASSID "your-ssid"
//#define STAPSK  "your-password"
#define STASSID "ChinaNet-2.4G-0DF0"    //the name of user's wifi
#define STAPSK  "ChinaNet@233"         //the password of user's wifi
#endif

const char* ssid = STASSID;
const char* password = STAPSK;

// TCP server at port 80 will response the HTTP requirement
WiFiServer server(80);

void setup(void) {
  Serial.begin(115200);
}

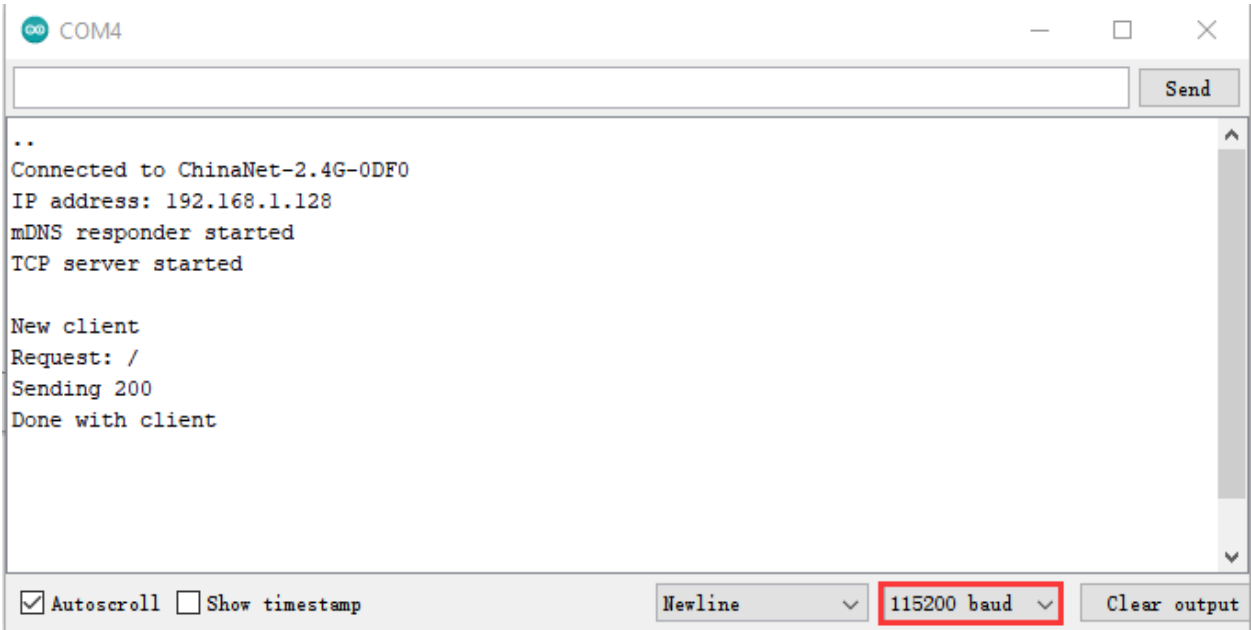
```

Done uploading.

variables use 28032 bytes (34%) of dynamic memory, leaving 53888 bytes for local  
 ng 303792 bytes from C:\Users\ADMINI~1\AppData\Local\Temp\arduino\_build\_112044/Pr  
 ..... [ 26% ]  
 ..... [ 53% ]  
 ..... [ 80% ]  
 ..... [ 100% ]

Module), 80 MHz, Flash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

After the ESP8266 code is successfully uploaded, first unplug the USB to ESP-01S WIFI module serial test expansion board from the computer, then turn its DIP. switch to Flash Boot and interface it with the USB port of your PC again. Open the serial monitor and set baud rate to 115200, as shown below:



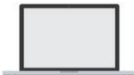





Project 11.2 : Control 8\*8 Dot Matrix Display Via WIFI

(1)Description

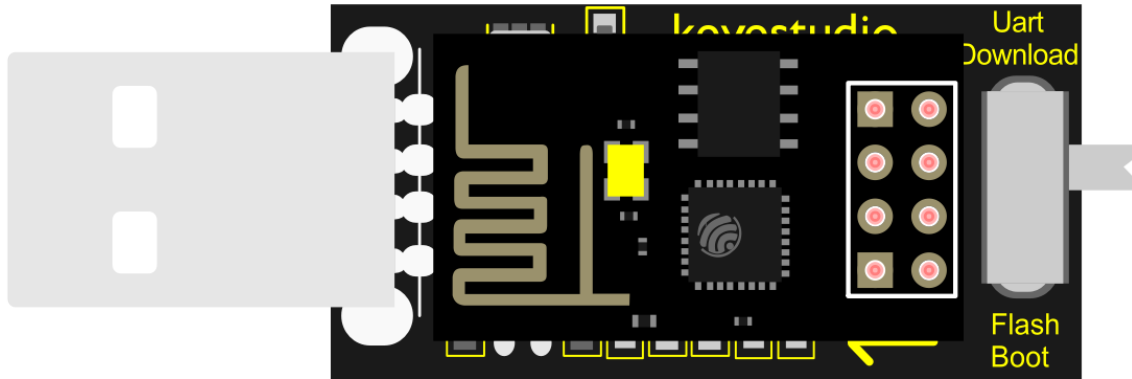
In this experiment, we will use the ESP8266 serial WIFI ESP-01 module to control the 8\*8 dot matrix display on the car through APP and WIFI.

(2)Components Required:

|   |   |   |   |  |   |
|---|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | WiFi module*1   | USB Serial ESP-01S WIFI Expansion Module *1  | 18650 Battery*1   |
|  |  |  |  |  |  |

**(3) Insert the Wifi serial port expansion board into the USB port of your PC:**

Insert the ESP8266 serial WIFI ESP-01 module into the USB to ESP-01S WIFI expansion board.

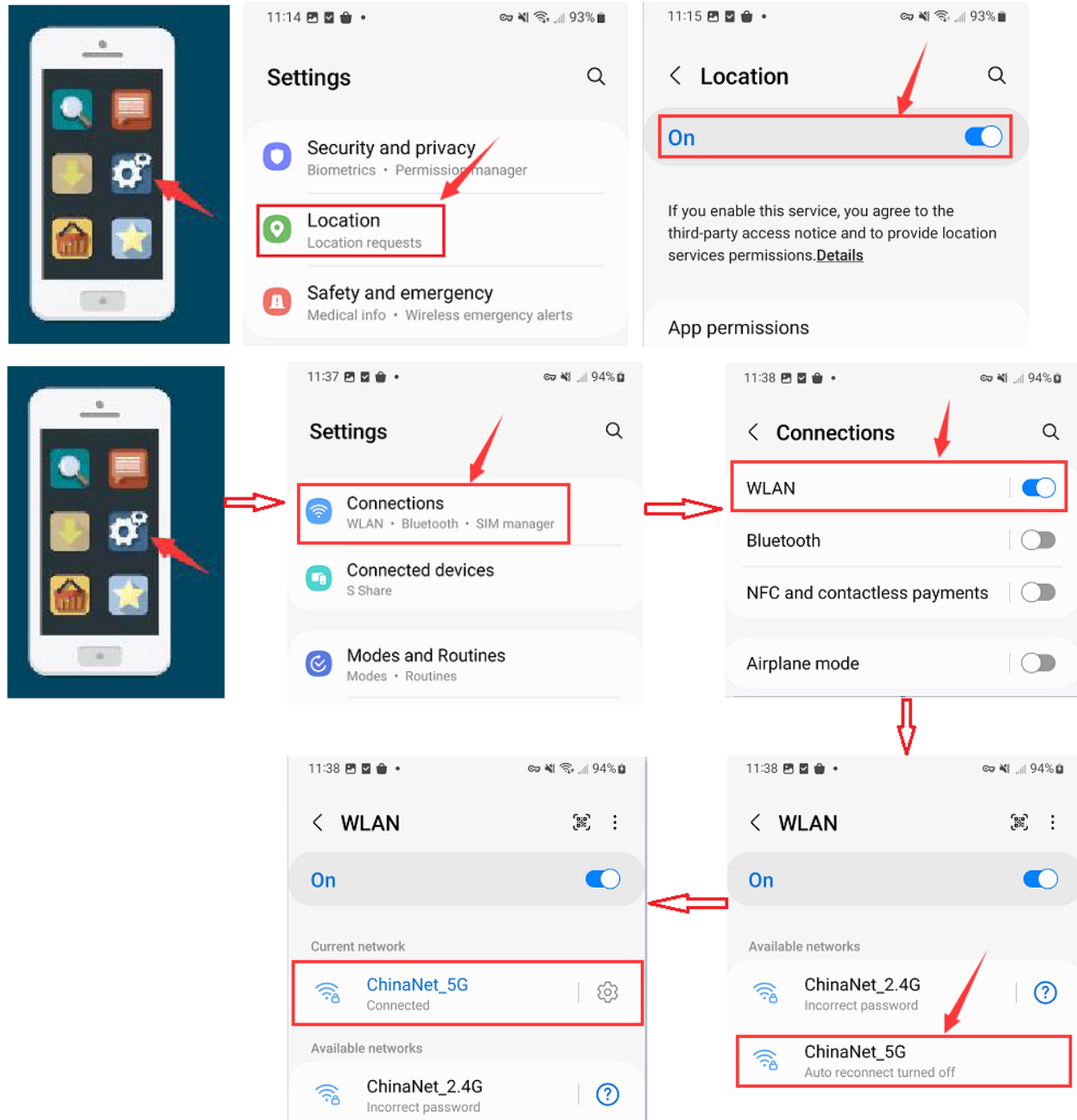


Turn the DIP switch of the USB to ESP-01S WIFI expansion board to UartDownload end and plug it to the USB port.



**(4)APP:****For Android system**

(1). Turn on the location services of the mobile phone and connect the wifi of yourself.



(2). Search **Beetlebot** in Google Play, or open the following link to download and install the app.

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



11:03

95%



Google Play



beetlebot

Apps &amp; games

Movies

Books

**Beetlebot**  
keyestudio

50+

Downloads

3+

Rated for 3+ ⓘ

Install

See in Play Store app

11:03

95%

Google Play



# Beetlebot

keyestudio

3+

Rated for 3+ ⓘ

Install

Sponsored · People also installed



## Complete account setup

Review your account to continue installing apps on Google Play



(3). Click **Open** button and enter interface of the app.

11:04

95%

Google Play



Beetlebot

keyestudio

Uninstall

Open

Sponsored · Suggested for you

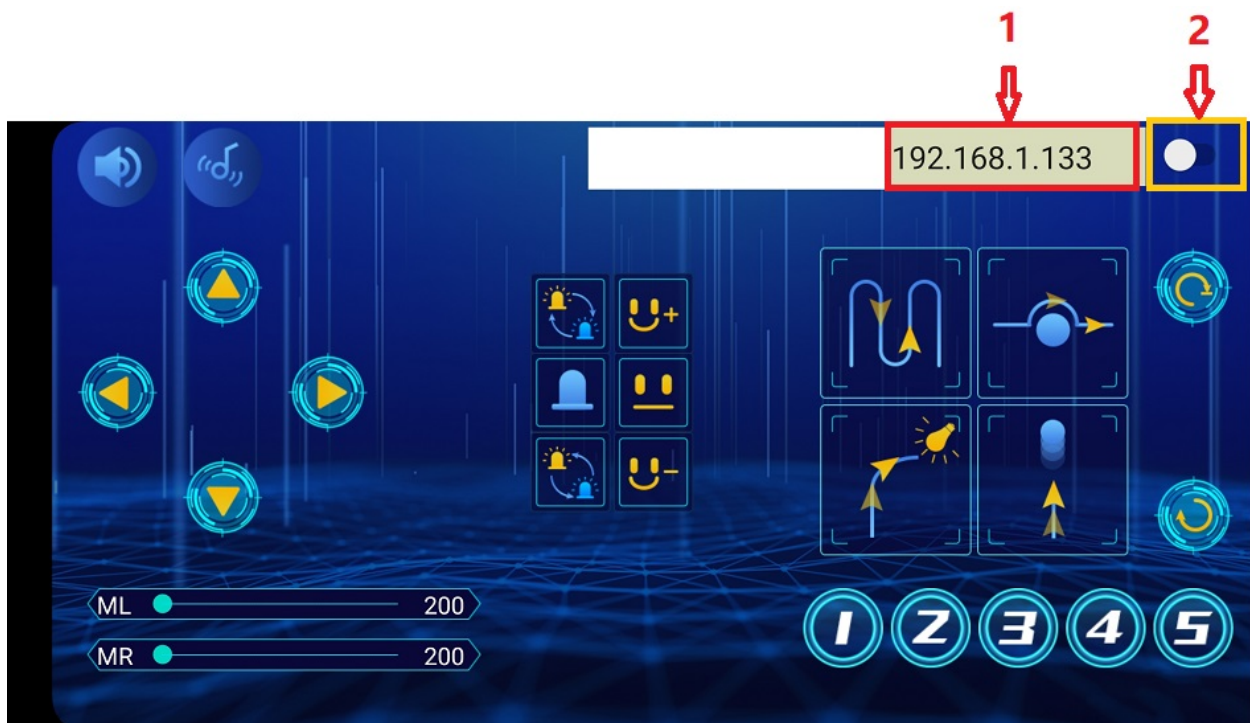




(4). Input the detected Wifi IP address(for example, the IP address in the serial monitor is 192.168.1.134), and Slide



the button to the right to connect Wifi. At same time, the IP address will be shown at the left box, which means that Wifi is connected well.

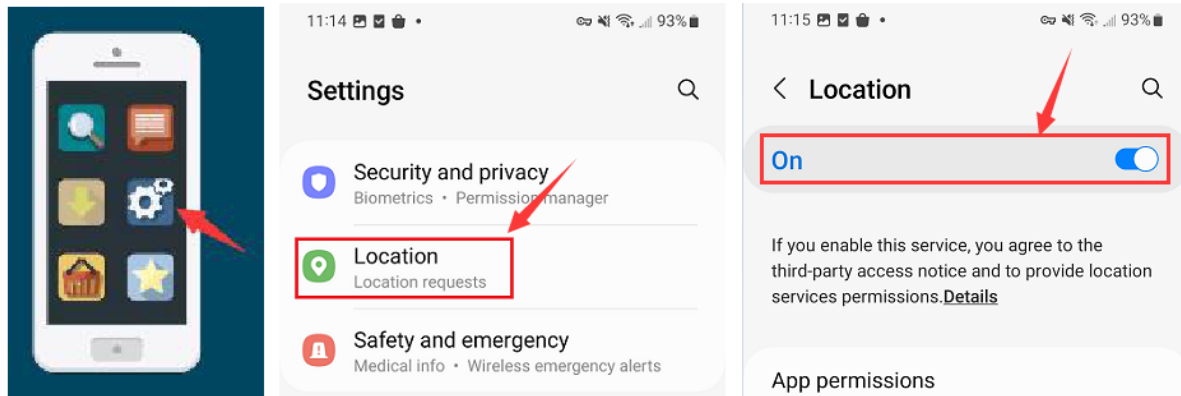


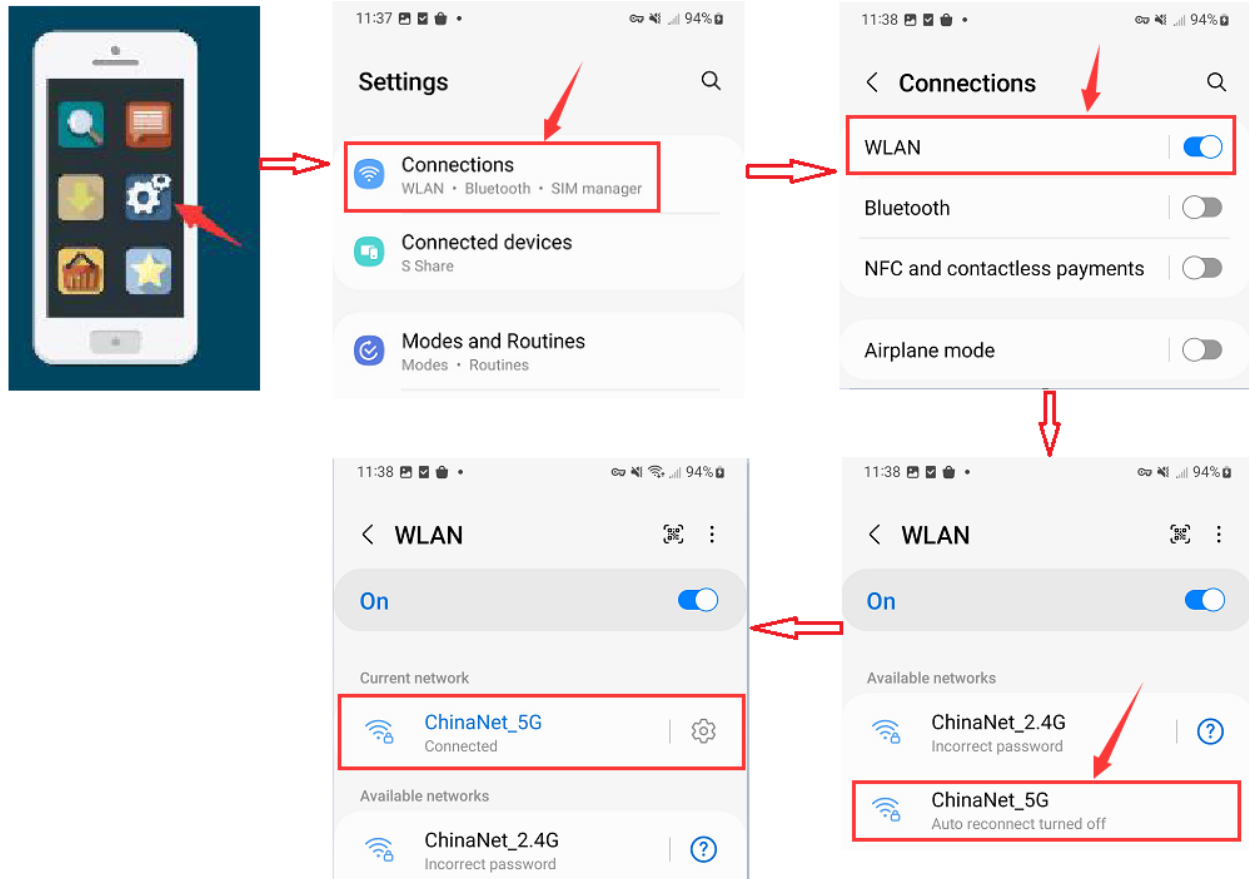


Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

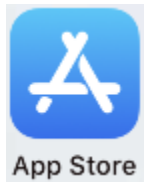
### For IOS system

(1). Turn on the location services of the mobile phone and connect the wifi of yourself.

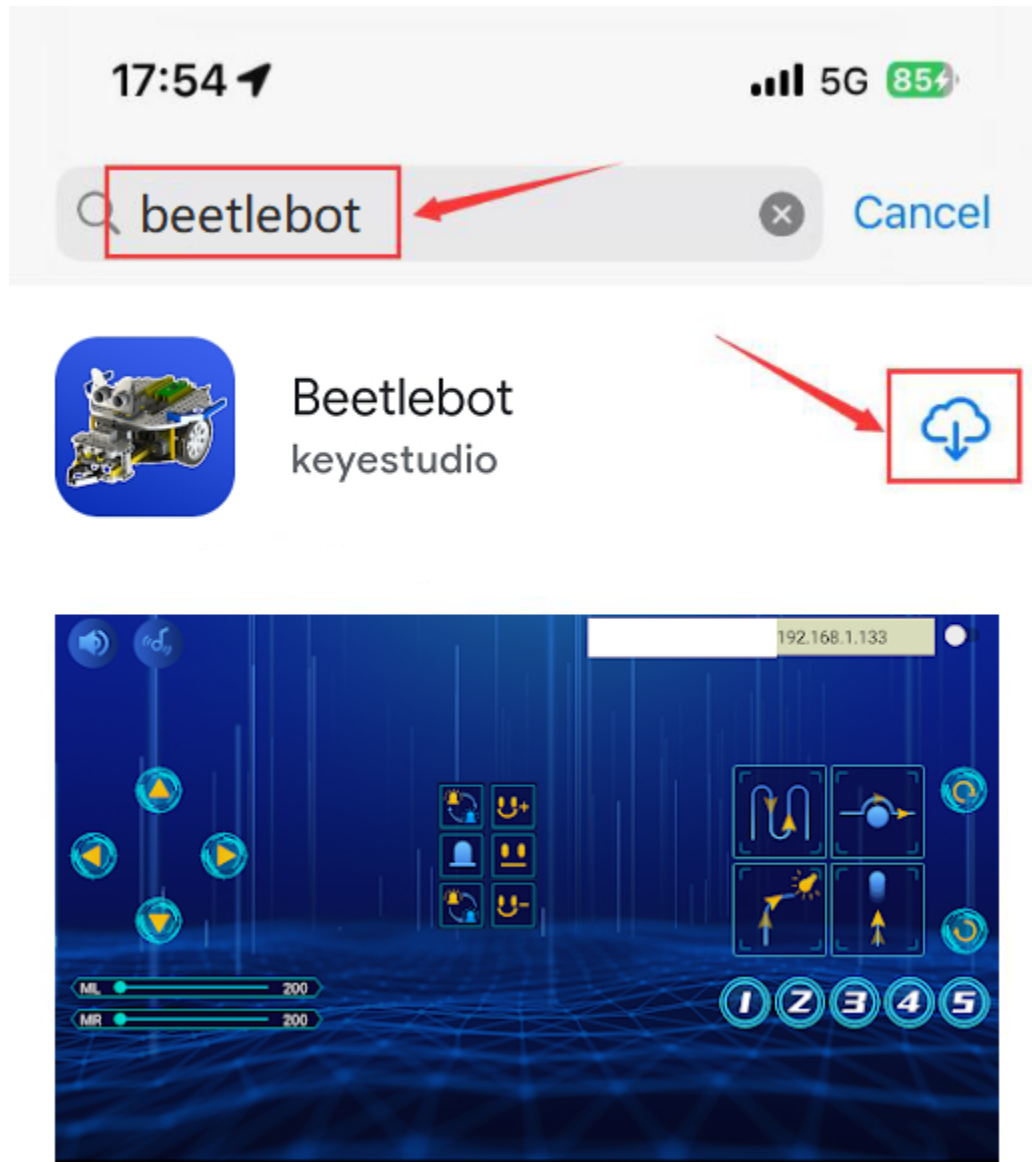




(2). Open App Store

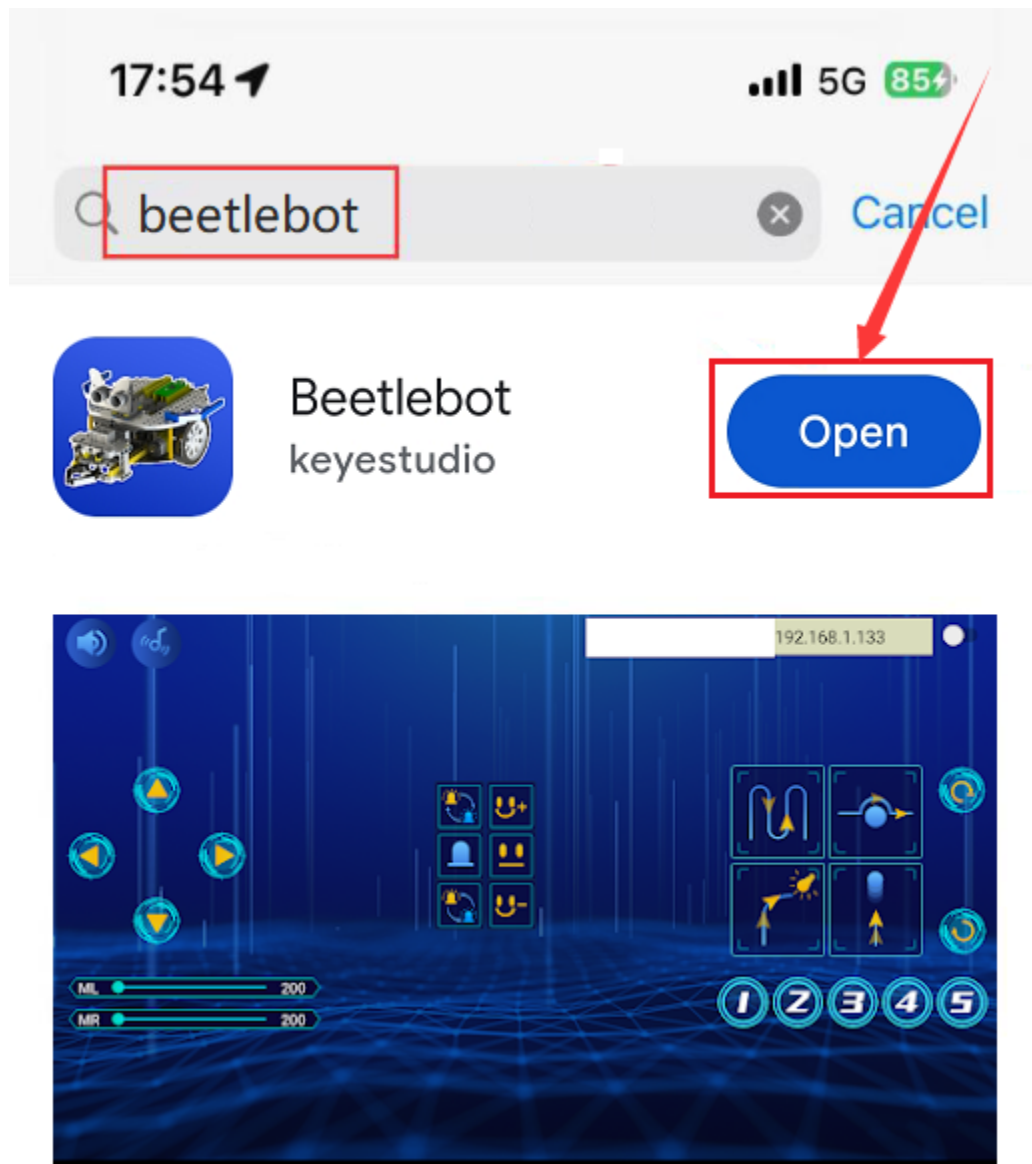


(3). Search **Beetlebot** in App Store click “” to download Beetlebot APP.



(4). Click **Open** button and enter interface of the app.





(5). Input the detected Wifi IP address(for example, the IP address in the serial monitor is 192.168.1.134), and Slide



the button to the right to connect Wifi. At same time, the IP address will be shown at the left box, which means that Wifi is connected well.



Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

### (5)ESP8266 Code

The pin UTXD and URXD of the ESP8266 serial WIFI ESP-01 module are controlled by RX(0) and TX(1) of the Arduino Nano board.

Note: you need to change the account and password of Wifi in the code into yours.

```
#ifndef STASSID
#define STASSID "ChinaNet-2.4G-0DF0" //the name of user's Wifi
#define STAPSK "ChinaNet@233" //the password of the user's wifi
#endif
```

```
/*
Project 11.2_1 ESP8266_Code
*/
// generated by KidsBlock
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <ESP8266mDNS.h>
#include <WiFiClient.h>
//#include <WiFi.h>

#ifndef STASSID
#define STASSID "ChinaNet-2.4G-0DF0"
#define STAPSK "ChinaNet@233"
#endif
const char* ssid = STASSID;
const char* password = STAPSK;

//IPAddress local_IP(192,168,4,22);
//IPAddress gateway(192,168,4,22);
//IPAddress subnet(255,255,255,0);
//
//const char *ssid = "ESP8266_AP_TEST";
//const char *password = "12345678";

WiFiServer server(80);
String unoData = "";
int ip_flag = 0;
int ultra_state = 1;
String ip_str;

void setup() {
  Serial.begin(9600);
  // WiFi.mode(WIFI_AP); //set the APmode
  //
  // WiFi.softAPConfig(local_IP, gateway, subnet); //set the AP address
  // while(!WiFi.softAP(ssid, password)){}; //enable AP
  // Serial.println("AP start successfully");
  //
  // Serial.print("IP address: ");
  // Serial.println(WiFi.softAPIP()); // print the IP address
  //
```

(continues on next page)

(continued from previous page)

```

// WiFi.softAPsetHostname("myHostName"); //print the host name
// Serial.print("HostName: ");
// Serial.println(WiFi.softAPgetHostname()); //print the host name
//
// Serial.print("mac Address: ");
// Serial.println(WiFi.softAPmacAddress()); //print the mac address

WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.print("IP ADDRESS: ");
Serial.println(WiFi.localIP());
if (!MDNS.begin("esp8266")) {
    //Serial.println("Error setting up MDNS responder!");
    while (1) {
        delay(1000);
    }
}
// Serial.println("mDNS responder started");
server.begin();
//Serial.println("TCP server started");
MDNS.addService("http", "tcp", 80);
ip_flag = 1;
}

void loop() {
    //Serial.println(WiFi.softAPgetStationNum()); //
    if(ip_flag == 1)
    {
        for(int i=3; i>0; i--)
        {
            Serial.print("IP: ");
            Serial.print(WiFi.localIP());
            Serial.println('#');
            delay(500);
        }
        ip_flag = 0;
    }
    MDNS.update();
    WiFiClient client = server.available();
    if (!client) {
        return;
    }
    //Serial.println("");
    while (client.connected() && !client.available()) {
        delay(1);
    }
    String req = client.readStringUntil('\r');

```

(continues on next page)

(continued from previous page)

```

int addr_start = req.indexOf(' ');
int addr_end = req.indexOf(' ', addr_start + 1);
if (addr_start == -1 || addr_end == -1) {
    //Serial.print("Invalid request: ");
    //Serial.println(req);
    return;
}
req = req.substring(addr_start + 1, addr_end);
int len_val = String(req).length();
String M_req = String(req).substring(0,6);
//Serial.println(M_req);
if(M_req == "/btn/u")
{
    String s_M_req = String(req).substring(5,len_val);
    Serial.print(s_M_req);
    Serial.print("#");
}
if(M_req == "/btn/v")
{
    String s_M_req = String(req).substring(5,len_val);
    Serial.print(s_M_req);
    Serial.print("#");
}
client.flush();
String s;
if (req == "/") {
    IPAddress ip = WiFi.localIP();
    String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +
    ↪String(ip[3]);
    s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
    ↪Hello from ESP8266 at ";
    s += ipStr;
    s += "</html>\r\n\r\n";
    //Serial.println("Sending 200");
    Serial.println(WiFi.localIP());
    Serial.write('*');
    client.println(WiFi.localIP());
    ip_flag = 0;
}
else if(req == "/btn/F")
{
    Serial.write('F');
    client.println(F("F"));
}
else if(req == "/btn/B")
{
    Serial.write('B');
    client.println(F("B"));
}
else if(req == "/btn/L")
{
    Serial.write('L');

```

(continues on next page)

(continued from previous page)

```
    client.println(F("L"));
}
else if(req == "/btn/R")
{
    Serial.write('R');
    client.println(F("R"));
}
else if(req == "/btn/S")
{
    Serial.write('S');
    client.println(F("S"));
}
else if(req == "/btn/a")
{
    Serial.write('a');
    client.println(F("a"));
}
else if(req == "/btn/b")
{
    Serial.write('b');
    client.println(F("b"));
}
else if(req == "/btn/c")
{
    Serial.write('c');
    client.println(F("c"));
}
else if(req == "/btn/d")
{
    Serial.write('d');
    client.println(F("d"));
}
else if(req == "/btn/e")
{
    Serial.write('e');
    client.println(F("e"));
}
else if(req == "/btn/f")
{
    Serial.write('f');
    client.println(F("f"));
}
else if(req == "/btn/g")
{
    Serial.write('g');
    client.println(F("g"));
}
else if(req == "/btn/z")
{
    Serial.write('z');
    client.println(F("z"));
}
```

(continues on next page)

(continued from previous page)

```
else if(req == "/btn/i")
{
    Serial.write('i');
    client.println(F("i"));
}
else if(req == "/btn/j")
{
    Serial.write('j');
    client.println(F("j"));
}
else if(req == "/btn/k")
{
    Serial.write('k');
    client.println(F("k"));
}
else if(req == "/btn/y")
{
    Serial.write('y');
    client.println(F("y"));
}
else if(req == "/btn/l")
{
    Serial.write('l');
    client.println(F("l"));
}
else if(req == "/btn/m")
{
    Serial.write('m');
    client.println(F("m"));
}
else if(req == "/btn/n")
{
    Serial.write('n');
    client.println("n");
}
else if(req == "/btn/o")
{
    Serial.write('o');
    client.println(F("o"));
}
else if(req == "/btn/p")
{
    Serial.write('p');
    client.println(F("p"));
}
else if(req == "/btn/q")
{
    Serial.write('q');
    client.println("q");
}
else if(req == "/btn/x")
{

```

(continues on next page)

(continued from previous page)

```

        Serial.write('x');
        client.println(F("x"));
    }
    else if(req == "/btn/1")
    {
        Serial.write('1');
        client.println(F("1"));
    }
    else if(req == "/btn/2")
    {
        Serial.write('2');
        client.println("2");
    }
    else if(req == "/btn/3")
    {
        Serial.write('3');
        client.println(F("3"));
    }
    else if(req == "/btn/4")
    {
        Serial.write('4');
        client.println("4");
    }
    else if(req == "/btn/5")
    {
        Serial.write('5');
        client.println(F("5"));
    }
    else if(req == "/btn/0")
    {
        Serial.write('0');
        client.println("0");
    }
    else {
        //s = "HTTP/1.1 404 Not Found\r\n\r\n";
        //Serial.println("Sending 404");
    }

    client.print(F("IP : "));
    client.println(WiFi.localIP());
}

```

After the account and password of Wifi is changed, turn the DIP switch of the USB to ESP-01S WIFI module to the Uart Download end and plug ESP-01S WIFI module into the USB port of your PC.

Note: Set board type and COM port according to the Project 11.1.

And upload the ESP8266 code to the WIFI ESP-01 module.

If the test code is not uploaded successfully, check the board type and the COM port first, then unplug the ESP-01S WIFI module and restart it.

After the ESP8266 code is successfully uploaded, first unplug the USB to ESP-01S WIFI module serial test expansion board from the computer, then disconnect the ESP8266 serial WIFI ESP-01 module from the USB to ESP-01S WIFI



expansion board.

## (6)Test Code

Note: Open the IDE and set the board type and the COM port. If there is no Wifi in your home, just open your cellphone to start the shared Wifi via hotspot.

```

/*
Project 11.2_2 WiFi control dot matrix
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);//Define the dot matrix pins in A4,A5
//Array, used to store the data of the pattern, can be calculated yourself
//or retrieved from the touch tool
uint8_t matrix_smile[8]={0x00,0x66,0x00,0x00,0x18,0x42,0x3c,0x00};
uint8_t matrix_heart[8]={0x0e,0x11,0x21,0x42,0x21,0x11,0x0e,0x00};
uint8_t matrix_ten[8]={0x08,0x08,0x08,0x08,0xff,0x08,0x08,0x08};
uint8_t LEDArray[8];
char wifiData;

void setup() {
  Serial.begin(9600);
  myMatrix.begin(112);
  myMatrix.clear();
  myMatrix.writeDisplay();
}

void loop() {
  if(Serial.available() > 0)
  {
    wifiData = Serial.read();
    Serial.print(wifiData);
    if(wifiData == '#')
    {
      Serial.println("");
    }
    delay(100);

    if(wifiData == 'i')
    {
      myMatrix.writeDisplay();
      matrix_display(matrix_smile);
    }
    else if(wifiData == 'k')
    {
      myMatrix.writeDisplay();
      matrix_display(matrix_heart);
    }
    else if(wifiData == 'j')
    {
      myMatrix.writeDisplay();
      matrix_display(matrix_ten);
    }
  }
}

```

(continues on next page)

(continued from previous page)

```

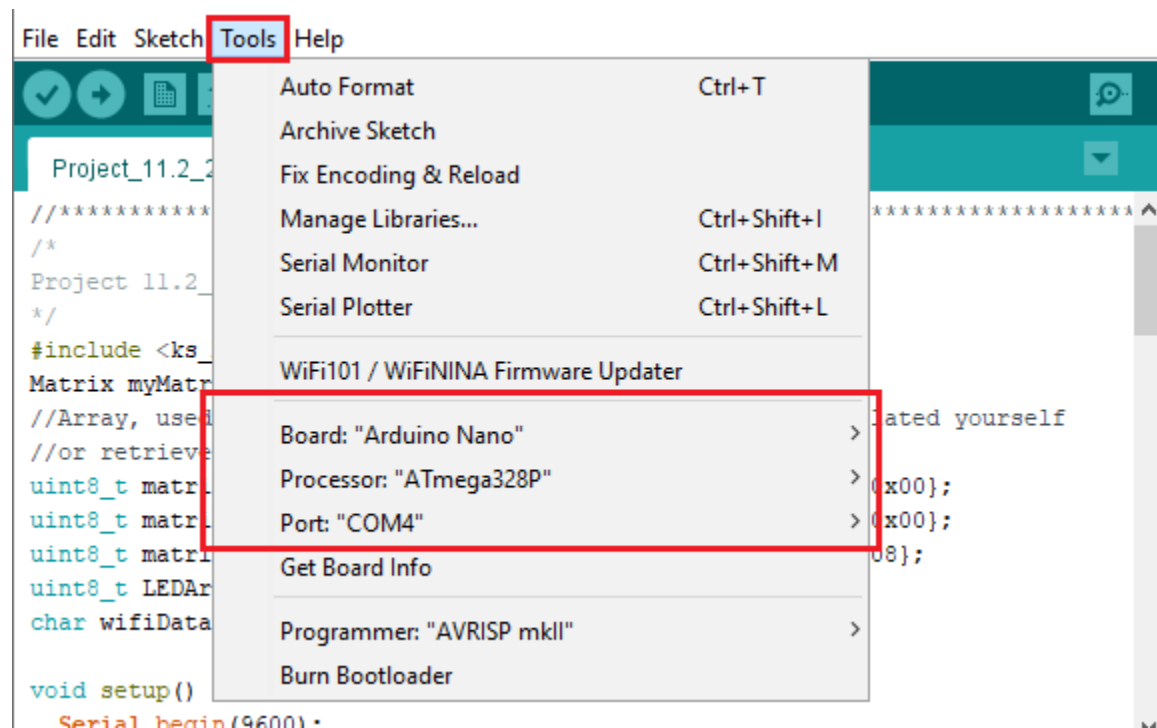
    }
    else if(wifiData == 'y')
    {
        myMatrix.clear();
    }
}
}

//Dot matrix display pattern function
void matrix_display(unsigned char matrix_value[])
{
    for(int i=0; i<8; i++)
    {
        LEDArray[i]=matrix_value[i];
        for(int j=7; j>=0; j--)
        {
            if((LEDArray[i]&0x01)>0)
                myMatrix.drawPixel(j, i,1);
            LEDArray[i] = LEDArray[i]>>1;
        }
    }
    myMatrix.writeDisplay();
}
}

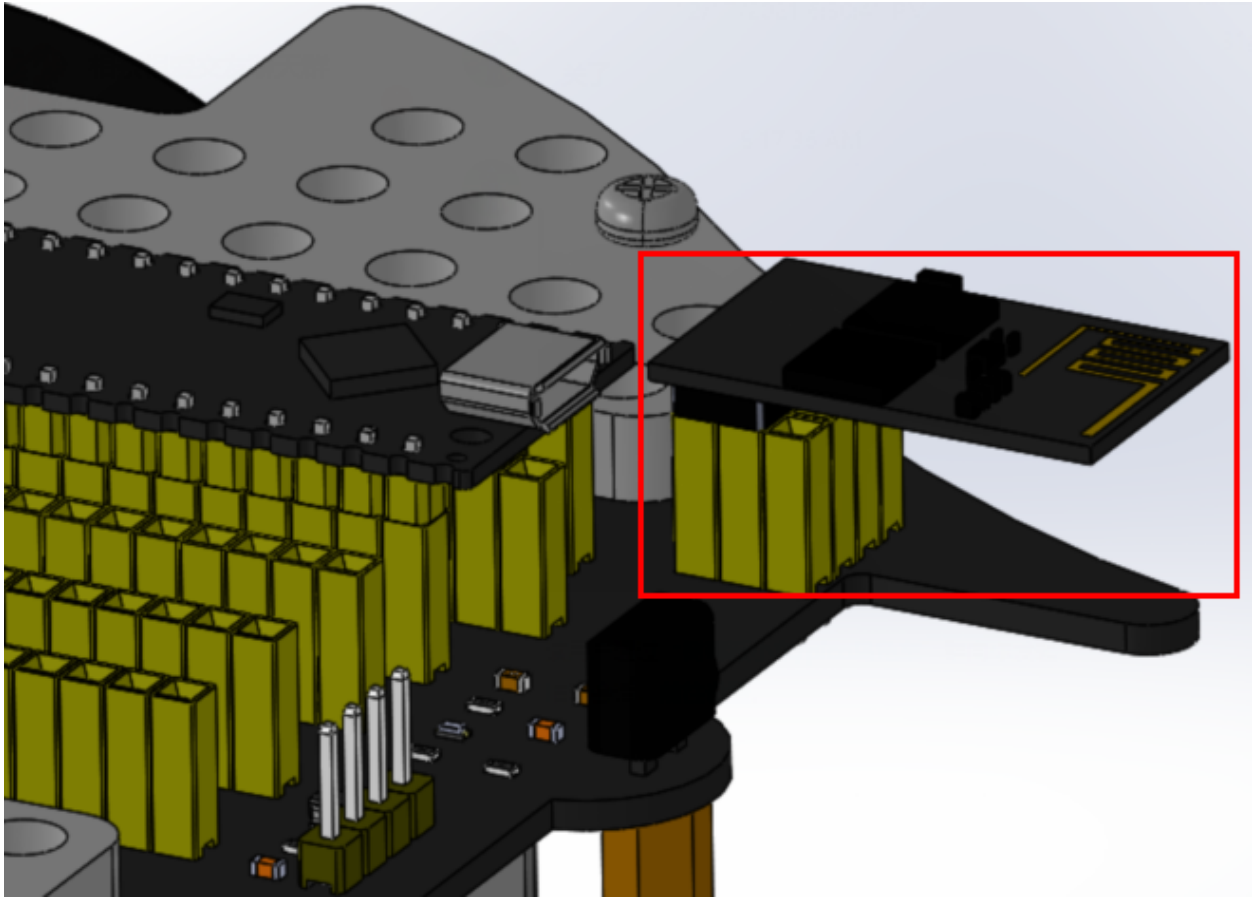
```


## (7)Test Result

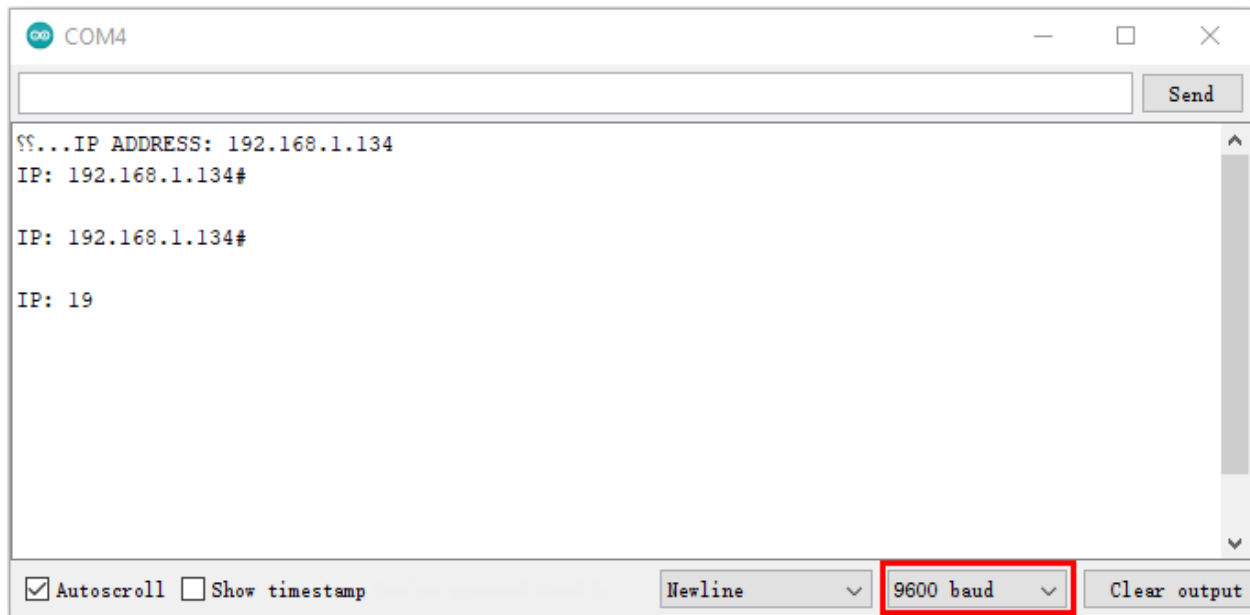
Click“Tools” → “Board”, select Arduino Nano and the correct COM portUpload the test code to the Arduino Nano board.



Insert the ESP8266 serial WIFI ESP-01 module into the Wifi port of the PCB board(Note: don't disconnect the USB cable).



Click  to open the serial monitor and set baud rate to 9600. Then the serial monitor will show the IP address of your Wifi. (IP addresses of Wifi sometimes change. If the original IP address can't be used, detect the IP address of Wifi again).



Open the app and input the detected Wifi IP address(for example, the IP address in the above figure is 192.168.1.134),



and Slide the button to the right to connect Wifi. At same time, the IP address will be shown at the left box, which means that Wifi is connected well.



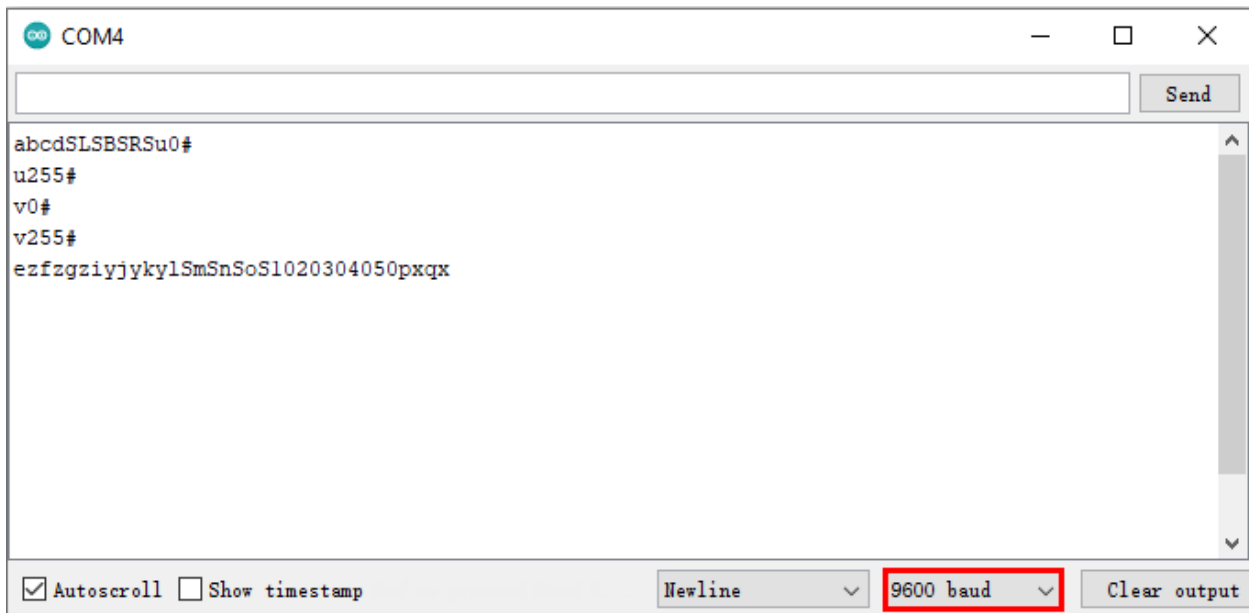


Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

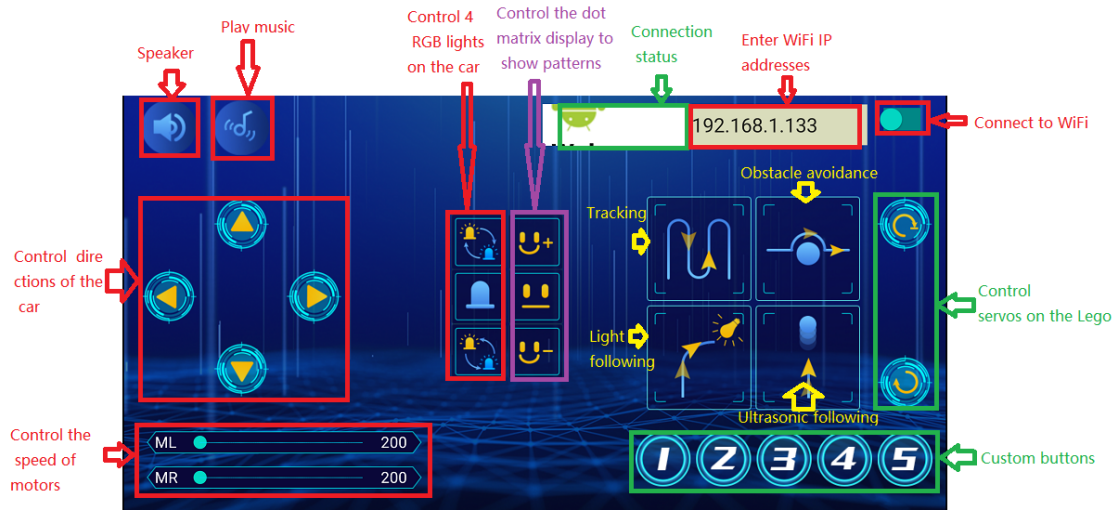
Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

After the APP has connected to the WIFI, start the following operations:

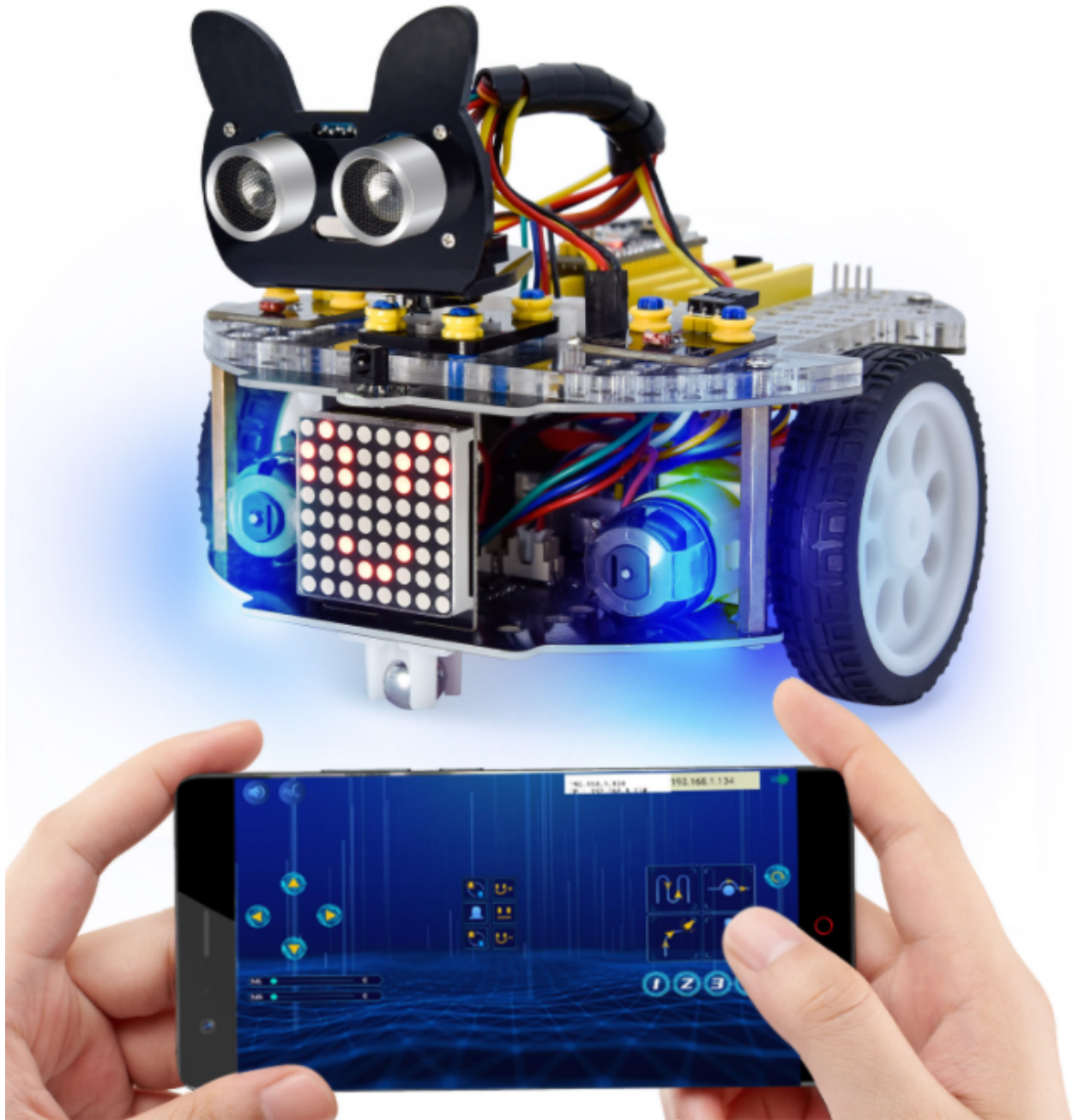
Click buttons on the app, the serial monitor will print some control characters, as shown below.



**Interface of App**







Click  a “smile” pattern will be displayed click  “” will be shown click  “” will be shown.



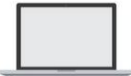
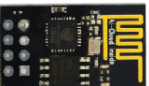




## Project 11.3: Multi-purpose Car

### (1)Description

In this project we will demonstrate multiple functions of the Beetlebot car through app.

### (2)Components Required:

|   |   |   |   |  |   |
|---|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | WiFi module*1   | USB Serial ESP-01S WIFI Expansion Module *1  | 18650 Battery*1   |
|  |  |  |  |  |  |

### (3)Test Code

The code of ESP8266wifi module is not changed, then change the wifi password of the code into yours.

```

//*****
/*
Project 11.3_2_Wifi_Multi_Function
*/
#include<music.h>

#include <Adafruit_NeoPixel.h>
Adafruit_NeoPixel rgb_display_A3 = Adafruit_NeoPixel(4,A3,NEO_GRB + NEO_KHZ800);
#include <Servo.h>
Servo lgservo;
Servo u_servo;
#include <ks_Matrix.h>
uint8_t LEDArray[8];
uint8_t matrix_smile[8]={0x42, 0xa5, 0xa5, 0x00, 0x00, 0x24, 0x18, 0x00};
uint8_t matrix_front[8]={0x18, 0x3c, 0x5a, 0x99, 0x18, 0x18, 0x18, 0x18};
uint8_t matrix_back[8]={0x18, 0x18, 0x18, 0x18, 0x99, 0x5a, 0x3c, 0x18};
uint8_t matrix_left[8]={0x08, 0x04, 0x02, 0xff, 0xff, 0x02, 0x04, 0x08};
uint8_t matrix_right[8]={0x10, 0x20, 0x40, 0xff, 0xff, 0x40, 0x20, 0x10};
uint8_t matrix_stop[8]={0xff, 0x81, 0xbd, 0xa5, 0xa5, 0xbd, 0x81, 0xff};
uint8_t matrix_tsundere[8]={0x00, 0xf7, 0x00, 0x08, 0x14, 0x20, 0x00, 0x00};
uint8_t matrix_squinting[8]={0x00, 0x41, 0x22, 0x14, 0x22, 0x41, 0x1c, 0x00};
uint8_t matrix_despise1[8]={0x00, 0x11, 0x77, 0x00, 0x1c, 0x00, 0x00, 0x00};
uint8_t matrix_speechless[8]={0x00, 0x77, 0x00, 0x1c, 0x14, 0x1c, 0x00, 0x00};
uint8_t matrix_heart[8]={0x00, 0x66, 0x99, 0x81, 0x81, 0x42, 0x24, 0x18};
uint8_t matrix_clear[8]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

#define ML 4
#define ML_PWM 6
#define MR 2
#define MR_PWM 5
#define buz 3

```

(continues on next page)



(continued from previous page)

```

#define Echo 7
#define Trig 8
#define servo1 9
#define trackL 11
#define trackR 10
#define ir 12
#define neo A3
#define servo2 A0

char val;
char wifiData;
int ip_flag = 1;
int neo_flag=0;

music Music(buz);

Matrix myMatrix(A4,A5);
int matrix_flag;
boolean face1_flag = 0;
boolean face2_flag = 0;
int face_count=0;

boolean neo_state = 0;
String left_str,right_str;
int left_val=255;
int right_val=255;

void setup() {
  Serial.begin(9600);
  pinMode(ML, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
  pinMode(buz, OUTPUT);
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
  pinMode(trackL, INPUT);
  pinMode(trackR, INPUT);
  pinMode(ir, INPUT);

  rgb_display_A3.begin();
  lgservo.attach(A0);
  lgservo.write(180);
  u_servo.attach(9);
  u_servo.write(90);
  delay(300);

  myMatrix.begin(112);
  myMatrix.clear();
  matrix_display(matrix_smile);
  delay(100);
}

```

(continues on next page)

(continued from previous page)

```

void loop() {
  if(Serial.available() > 0)
  {
    val = Serial.read();
    Serial.print(val);
    if(val == 'u')
    {
      Serial.println("left speed : ");
      left_str = Serial.readStringUntil('#');
      left_val = String(left_str).toInt();
      Serial.println(left_val);
    }
    if(val == 'v')
    {
      Serial.println("right speed : ");
      right_str = Serial.readStringUntil('#');
      right_val = String(right_str).toInt();
      Serial.println(right_val);
    }
  }
  switch(val)
  {
    case 'F': car_forward(); break;
    case 'B': car_back(); break;
    case 'L': car_left(); break;
    case 'R': car_right(); break;
    case 'S': car_stop(); break;
    case 'a': tone(buz, 294); delay(200); break;
    case 'b': noTone(buz); break;
    case 'c': Music.birthday(); break;
    case 'd': noTone(buz); break;
    case 'e': func_neo1(); break;
    case 'f': neo_stop(); break;
    case 'g': func_neo2(); break;
    case 'z': neo_state = 0; break;
    case 'i': face1(); break;
    case 'j': face_stop(); break;
    case 'k': face2(); break;
    case 'y': face1_flag=0; break;
    case 'l': tracking(); break;
    case 'm': avoid(); break;
    case 'n': followLightCar(); break;
    case 'o': followCar(); break;
  }
}

void followLightCar()
{
  int lightL = analogRead(A6);
  int lightR = analogRead(A7);

```

(continues on next page)

(continued from previous page)

```

Serial.print(lightL);
Serial.print(" ");
Serial.println(lightR);
if((lightL > 500) && (lightR > 500))
{
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,150);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,150);
}
else if((lightL > 500) && (lightR <= 500))
{
    car_left();
}
else if((lightL <= 500) && (lightR > 500))
{
    car_right();
}
else
{
    car_stop();
}
}

void followCar()
{
    int distance = checkdistance();
    Serial.print("distance = ");
    Serial.println(distance);
    if((distance > 10) && (distance < 35))
    {
        digitalWrite(ML,LOW);
        analogWrite(ML_PWM,150);
        digitalWrite(MR,LOW);
        analogWrite(MR_PWM,150);
    }
    else if((distance > 6) && (distance <= 10))
    {
        car_stop();
    }
    else if(distance <= 6)
    {
        digitalWrite(ML,HIGH);
        analogWrite(ML_PWM,100);
        digitalWrite(MR,HIGH);
        analogWrite(MR_PWM,100);
    }
    else
    {
        car_stop();
    }
}

```

(continues on next page)

(continued from previous page)

```
}

void avoid()
{
  int distance = checkdistance();
  Serial.print("distance = ");
  Serial.println(distance);
  if(distance <= 8)
  {
    car_stop();
    delay(300);
    u_servo.write(180);
    delay(500);
    int distanceL = checkdistance();
    delay(50);
    u_servo.write(0);
    delay(600);
    int distanceR = checkdistance();
    delay(50);
    if(distanceL > distanceR)
    {
      car_left();
      u_servo.write(90);
      delay(400);
    }
    else
    {
      car_right();
      u_servo.write(90);
      delay(400);
    }
  }
  else
  {
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,150);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,150);
  }
}

float checkdistance() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);
  float distance = pulseIn(Echo, HIGH) / 58.00;
  delay(10);
  return distance;
}
```

(continues on next page)

(continued from previous page)

```

void tracking()
{
  boolean trackL_val = digitalRead(trackL);
  boolean trackR_val = digitalRead(trackR);
  Serial.print(trackL_val);
  Serial.print(" ");
  Serial.println(trackR_val);
  if((trackL_val == 1) && (trackR_val == 1))
  {
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,120);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,120);
  }
  else if((trackL_val == 1) && (trackR_val == 0))
  {
    digitalWrite(ML,HIGH);
    analogWrite(ML_PWM,150);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,150);
  }
  else if((trackL_val == 0) && (trackR_val == 1))
  {
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,150);
    digitalWrite(MR,HIGH);
    analogWrite(MR_PWM,150);
  }
  else
  {
    car_stop();
  }
}

void face1()
{
  if(face1_flag==0){
    matrix_flag = 1;
  }
  if(matrix_flag == 1)
  {
    face_count++;
    if(face_count == 6)
    {
      face_count = 6;
    }
    matrix_flag = 0;
    face1_flag = 1;
  }
  switch(face_count)
  {
    case 1: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_smile); break;

```

(continues on next page)

(continued from previous page)

```

    case 2: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_tsundere);↵
    ↪break;
    case 3: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_squinting);↵
    ↪break;
    case 4: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_despise1);↵
    ↪break;
    case 5: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_speechless);↵
    ↪break;
    case 6: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_heart); break;
  }
}

void face_stop()
{
  myMatrix.clear();myMatrix.writeDisplay();
}

void face2()
{
  if(face1_flag==0){
    matrix_flag = 1;
  }
  if(matrix_flag == 1)
  {
    face_count--;
    if(face_count == 1)
    {
      face_count = 1;
    }
    matrix_flag = 0;
    face1_flag = 1;
  }
  switch(face_count)
  {
    case 1: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_smile); break;
    case 2: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_tsundere);↵
    ↪break;
    case 3: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_squinting);↵
    ↪break;
    case 4: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_despise1);↵
    ↪break;
    case 5: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_speechless);↵
    ↪break;
    case 6: myMatrix.clear();myMatrix.writeDisplay();matrix_display(matrix_heart); break;
  }
}

int matrix_display(uint8_t led_array[8]){
  for(int i=0; i<8; i++)
  {
    LEDArray[i]=led_array[i];
    for(int j=7; j>=0; j--)

```

(continues on next page)

(continued from previous page)

```

    {
        if((LEDArrary[i]&0x01)>0)
            myMatrix.drawPixel(j, i,1);
        LEDArray[i] = LEDArray[i]>>1;
    }
}
myMatrix.writeDisplay(); // dot matrix shows
}

void func_neo1()
{
    if(neo_state == 0)
    {
        neo_flag++;
        neo_state = 1;
    }
    if(neo_flag >= 6)
    {
        neo_flag = 6;
    }
    switch(neo_flag)
    {
        case 1: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((100 & 0xffffffff) << 16) | ((0 & 0xffffffff) << 8) | 0));rgb_display_A3.show();
        }
        break;
        case 2: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((0 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 0));rgb_display_A3.show();
        }
        break;
        case 3: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((0 & 0xffffffff) << 16) | ((0 & 0xffffffff) << 8) | 100));rgb_display_A3.show();
        }
        break;
        case 4: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((100 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 0));rgb_display_A3.show();
        }
        break;
        case 5: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((0 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 100));rgb_display_A3.show();
        }
        break;
        case 6: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((100 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 100));rgb_display_A3.show();
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    break;
}

}

void func_neo2()
{
    if(neo_state == 0)
    {
        neo_flag--;
        neo_state = 1;
    }
    if(neo_flag <= 1)
    {
        neo_flag = 1;
    }
    switch(neo_flag)
    {
        case 1: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((100 & 0xffffffff) << 16) | ((0 & 0xffffffff) << 8) | 0));rgb_display_A3.show();
        }
        break;
        case 2: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((0 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 0)); rgb_display_A3.show();
        }
        break;
        case 3: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((0 & 0xffffffff) << 16) | ((0 & 0xffffffff) << 8) | 100)); rgb_display_A3.show();
        }
        break;
        case 4: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((100 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 0));rgb_display_A3.show();
        }
        break;
        case 5: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((0 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 100)); rgb_display_A3.show();
        }
        break;
        case 6: for (int i = 1; i <= 4; i = i + (1)) {
            rgb_display_A3.setPixelColor(i-1, (((100 & 0xffffffff) << 16) | ((100 & 0xffffffff) << 8) | 100));rgb_display_A3.show();
        }
        break;
    }
}

```

(continues on next page)



(continued from previous page)

```

void neo_stop()
{
    neo_state = 0;
    for (int i = 1; i <= 4; i = i + (1)) {
        rgb_display_A3.setPixelColor((i)-1, (((0 & 0xffffffff) << 16) | ((0 & 0xffffffff) << 8) |
↪ 0));rgb_display_A3.show();
    }
}

void car_forward()
{
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,left_val);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,right_val);
}

void car_back()
{
    digitalWrite(ML,HIGH);
    analogWrite(ML_PWM,(255-left_val));
    digitalWrite(MR,HIGH);
    analogWrite(MR_PWM,(255-right_val));
}

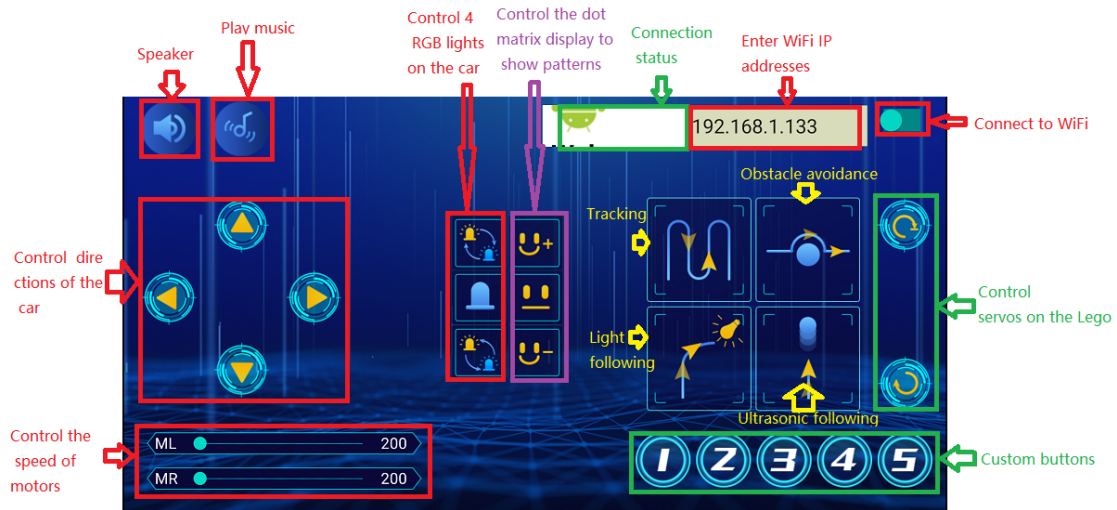
void car_left()
{
    digitalWrite(ML,HIGH);
    analogWrite(ML_PWM,127);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,127);
}

void car_right()
{
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,127);
    digitalWrite(MR,HIGH);
    analogWrite(MR_PWM,127);
}

void car_stop()
{
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,0);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,0);
}
//*****

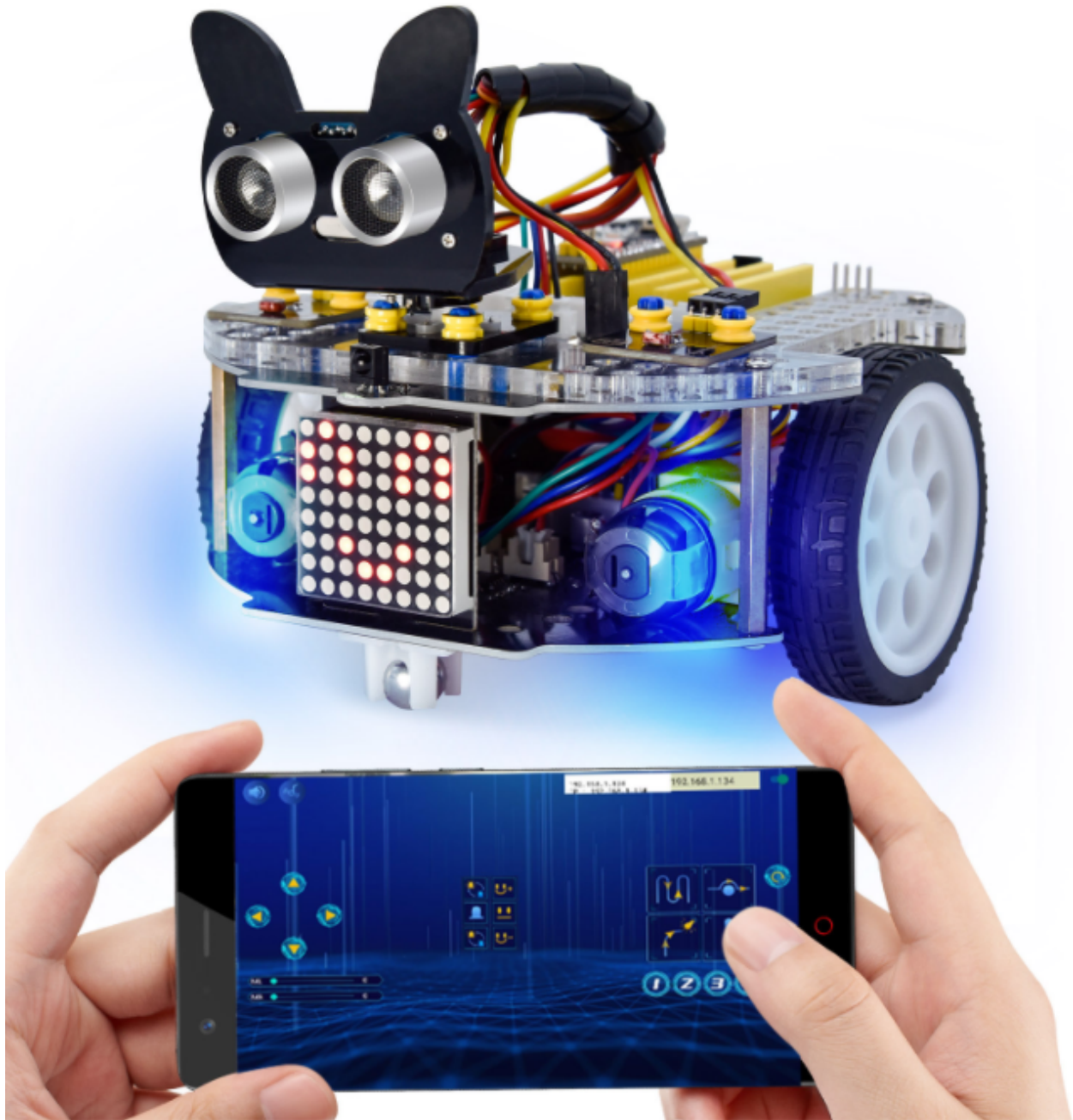
```

### (4)APP operation, as shown below:



Note: See the previous lesson of Project 11.2 for how to connect your app to WiFi

Upload the code to the Arduino Nano board, after uploading the code successfully, after power-on, after the mobile APP is connected to the wifi successfully, we can use the mobile APP to control the beetlebot.





## KIDSBLOCK TUTORIAL

### 9.1 1. Getting started with Kidsblock software

#### 9.1.1 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 graphical blocks and using the C/C++ programming language, making programming easy and interesting for children to learn.

#### 9.1.2 2. Download and install KidsBlock software

[Windows system](#)

[MACOS system](#)

[Install Development Board Driver](#)

[How to install development board driver](#)

[Start your first program](#) [Quick Start](#)

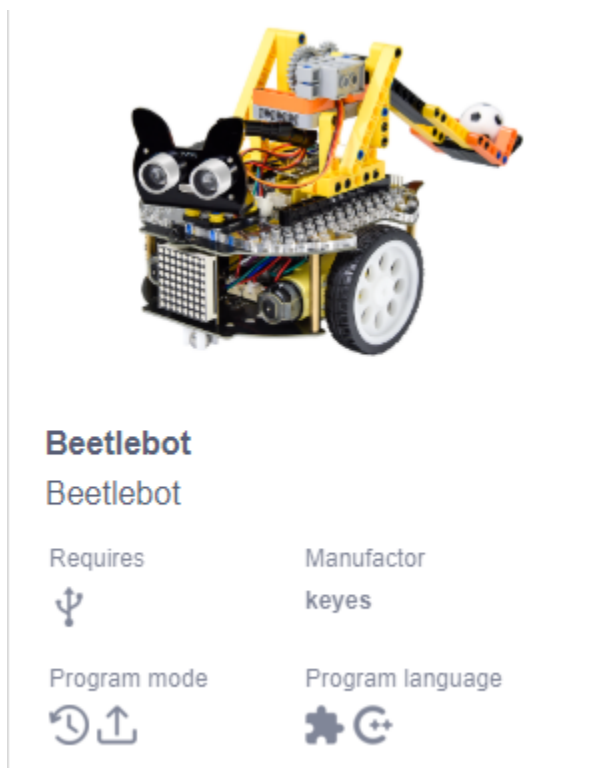
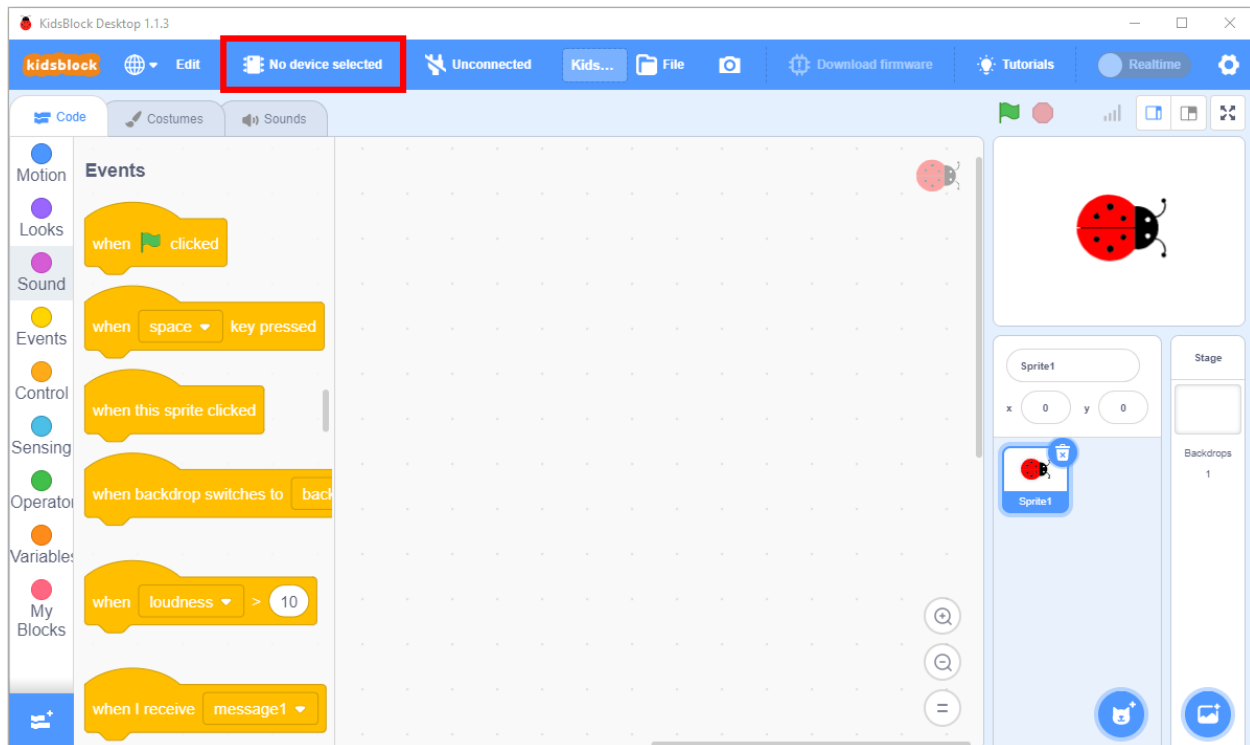
#### 9.1.3 3. Interface Setting

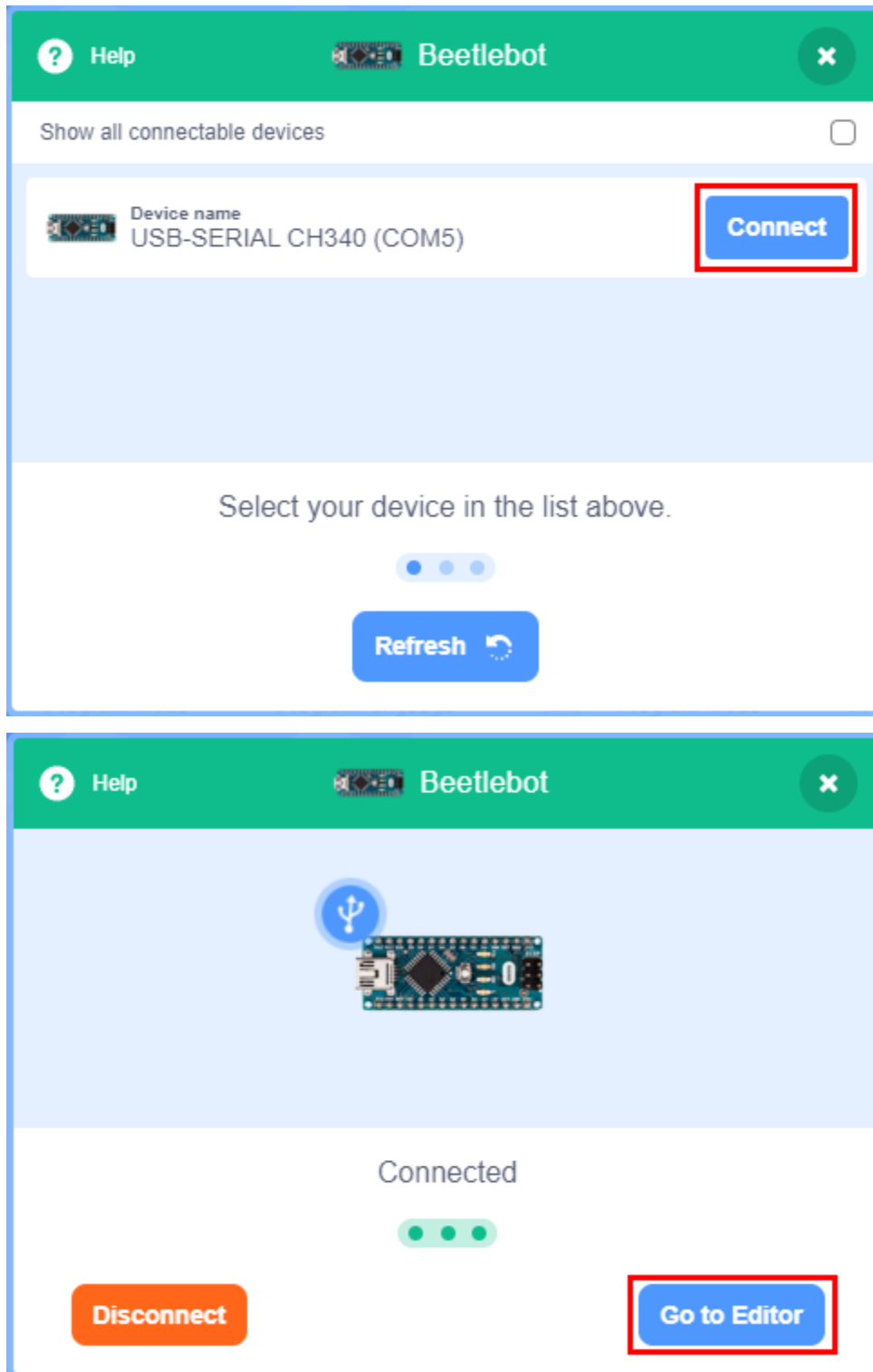
After the KidsBlock is installed open KidsBlock to click  **No device selected** < “Beetlebot” < “Connect”

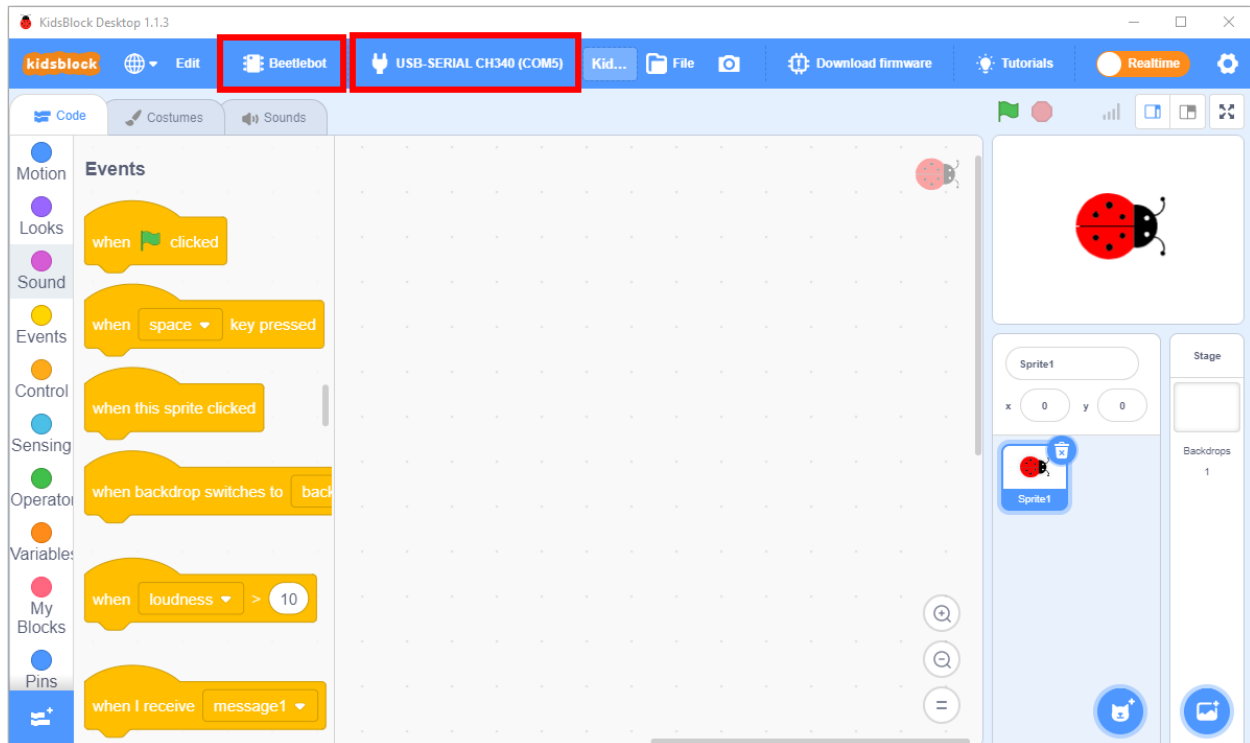
Then the **Beetlebot** is connected. Click “Go to Editor” to return the code editor.



 **No device selected** will change into  **Beetlebot** and  **Unconnected** into  **USB-SERIAL CH340 (COM5)**.

This means the **Beetlebot** is connected to the COM port.

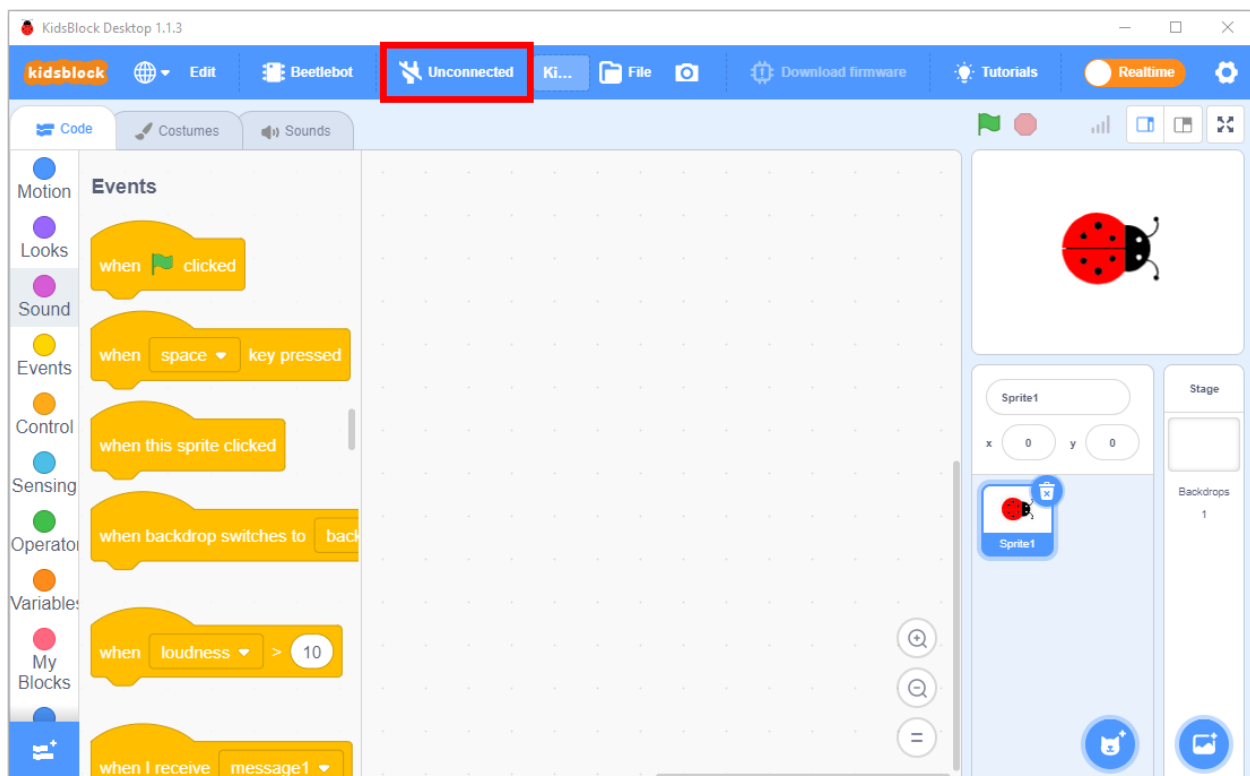




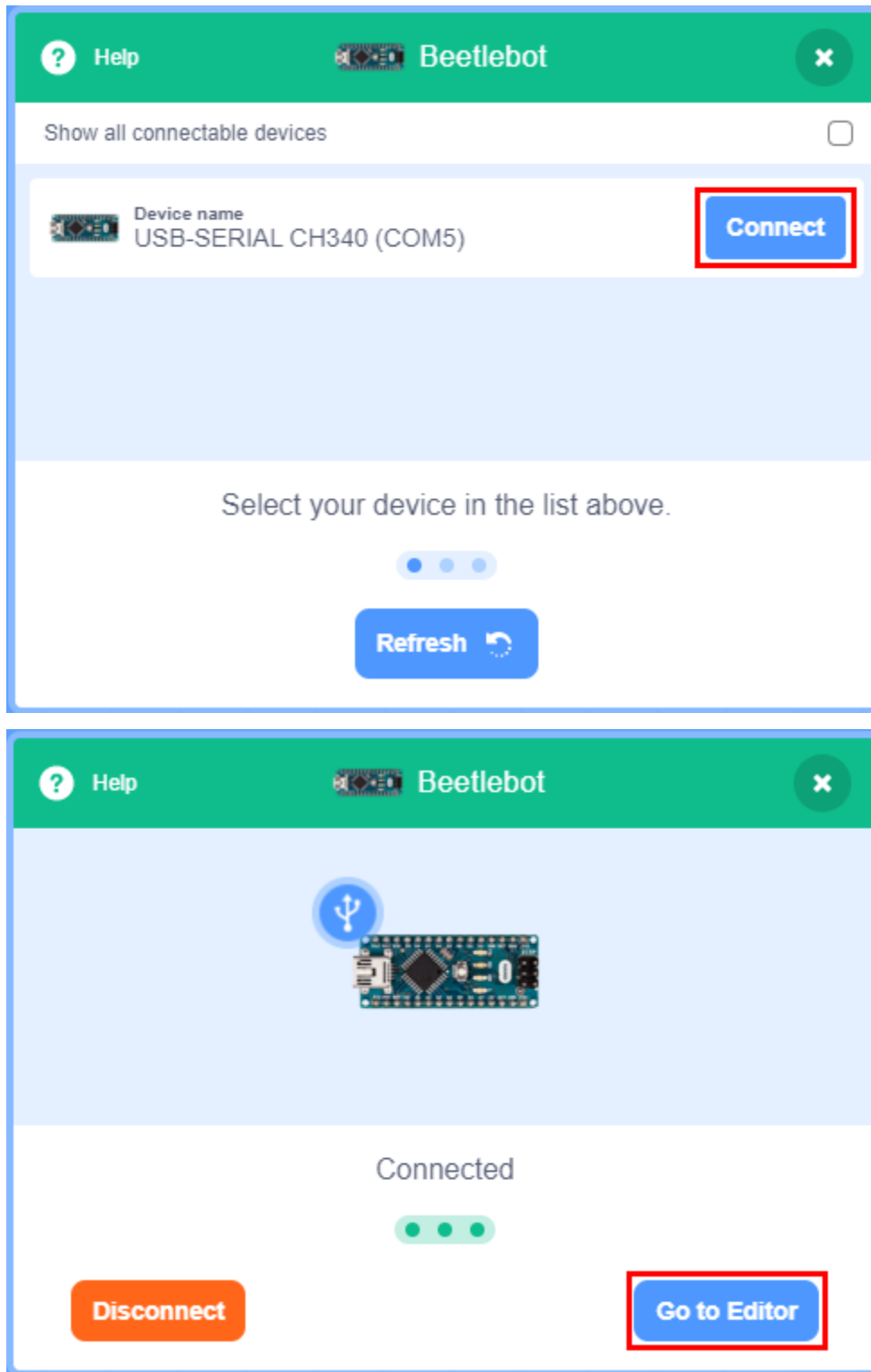


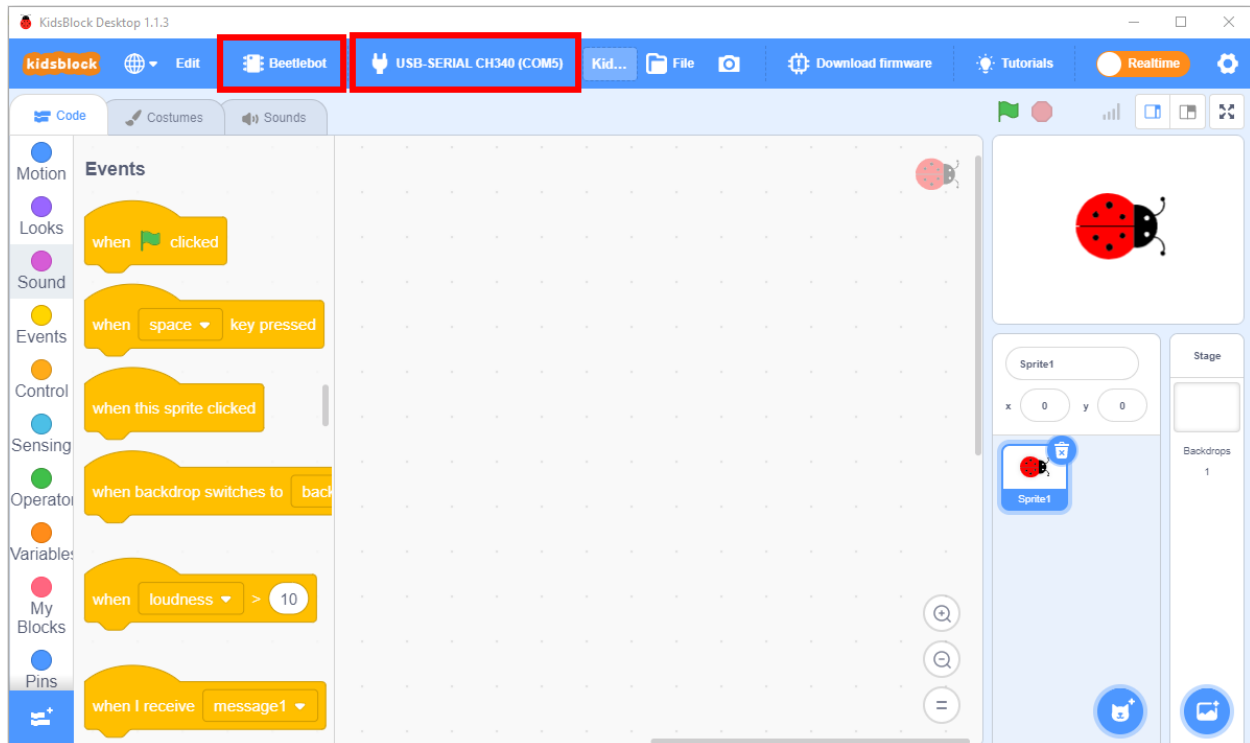
If the **Beetlebot** is connected, but  doesn't change into .




Just click  to connect the COM port. Click  then click “**Connect**” after a while, if the “**Connected**” page pops up the com port will be connected.

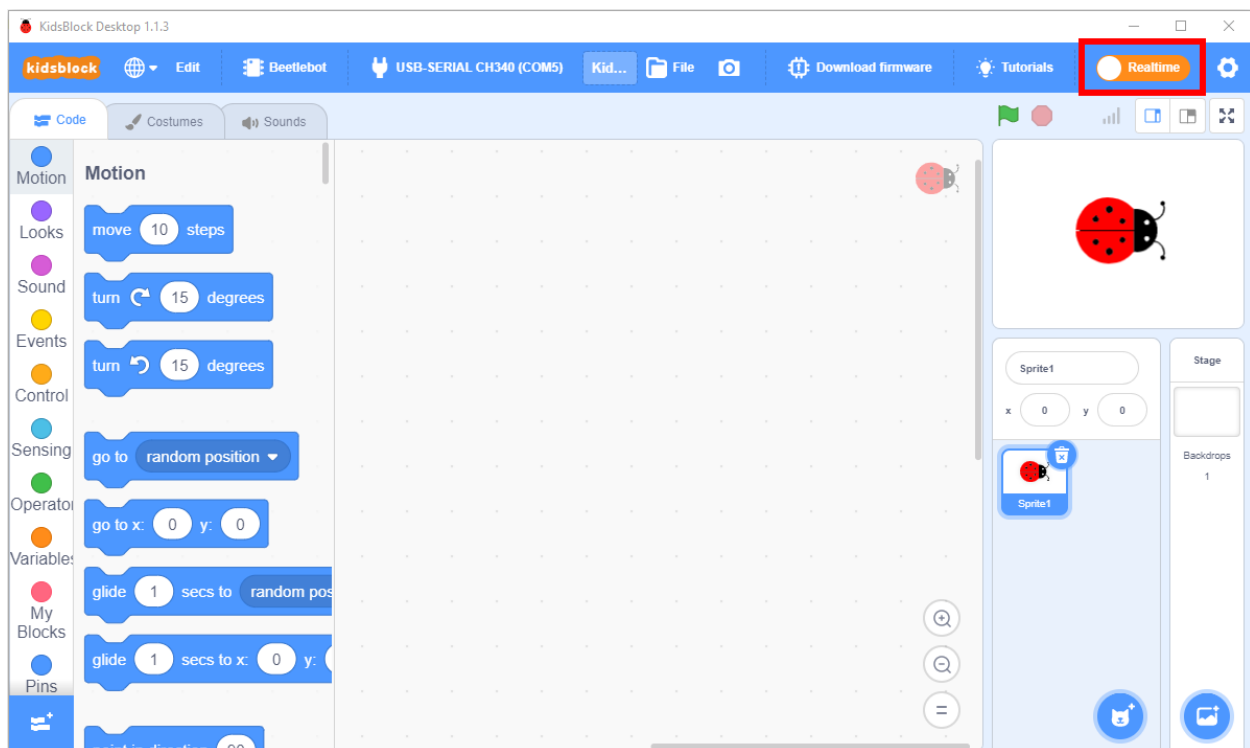


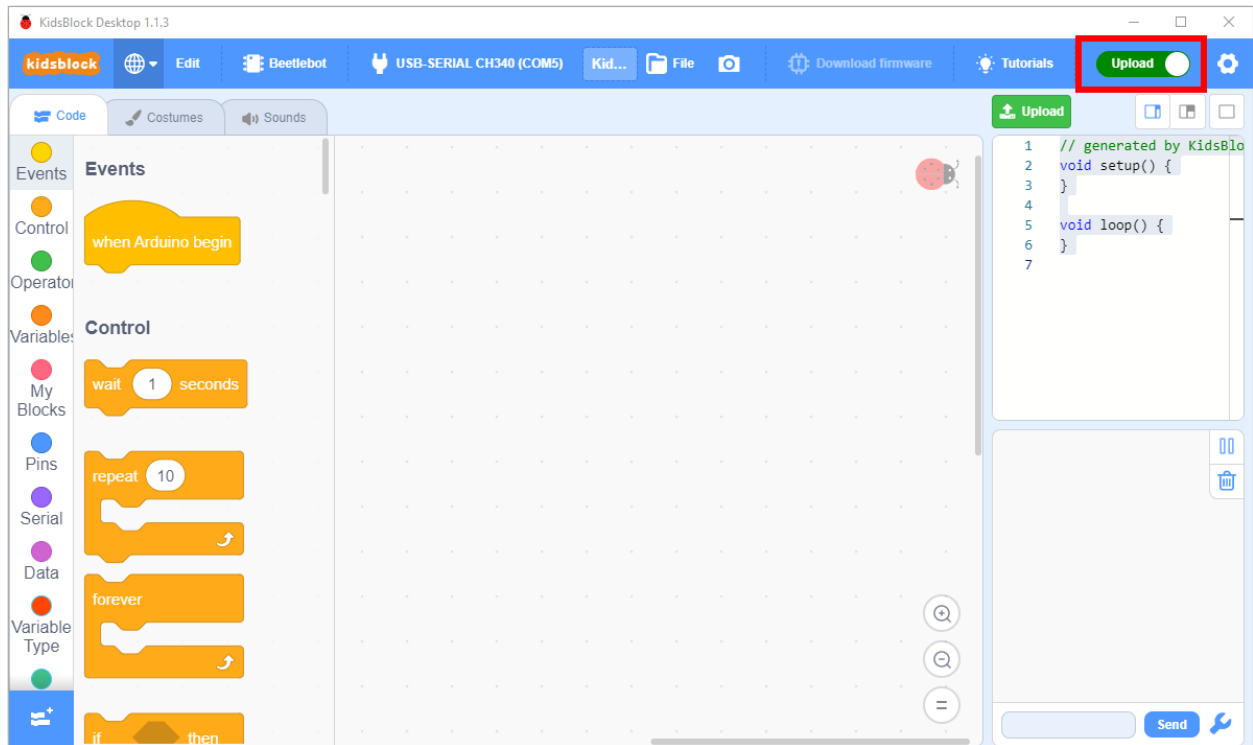






Then click  to switch the mode, the mode  will switch to 

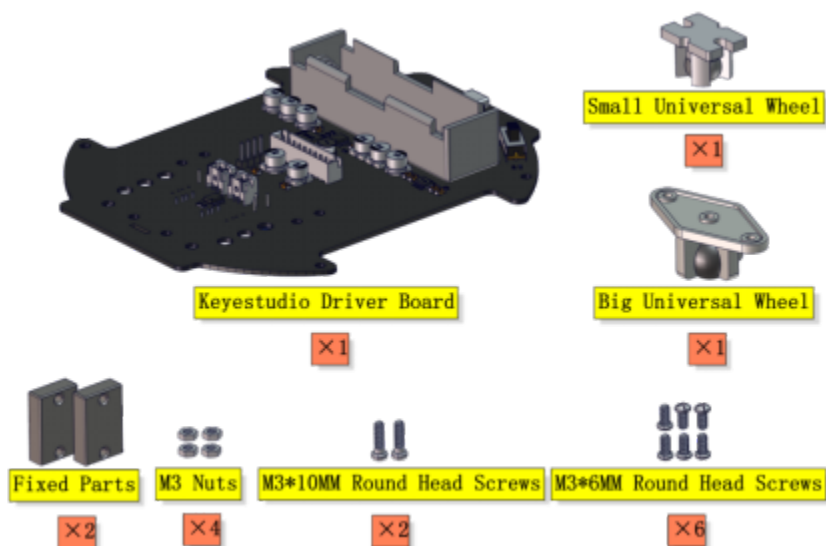


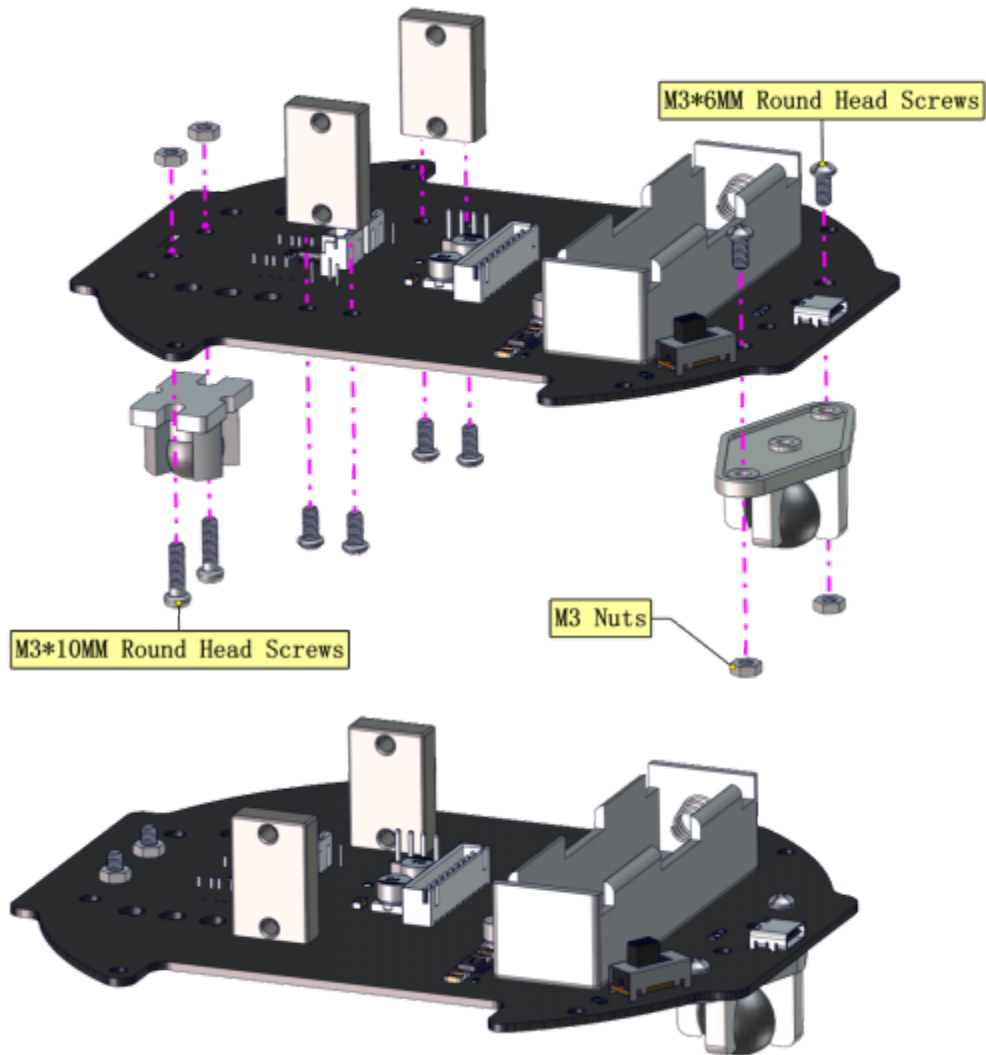


## 9.2 2. Assemble Beetlebot Robot

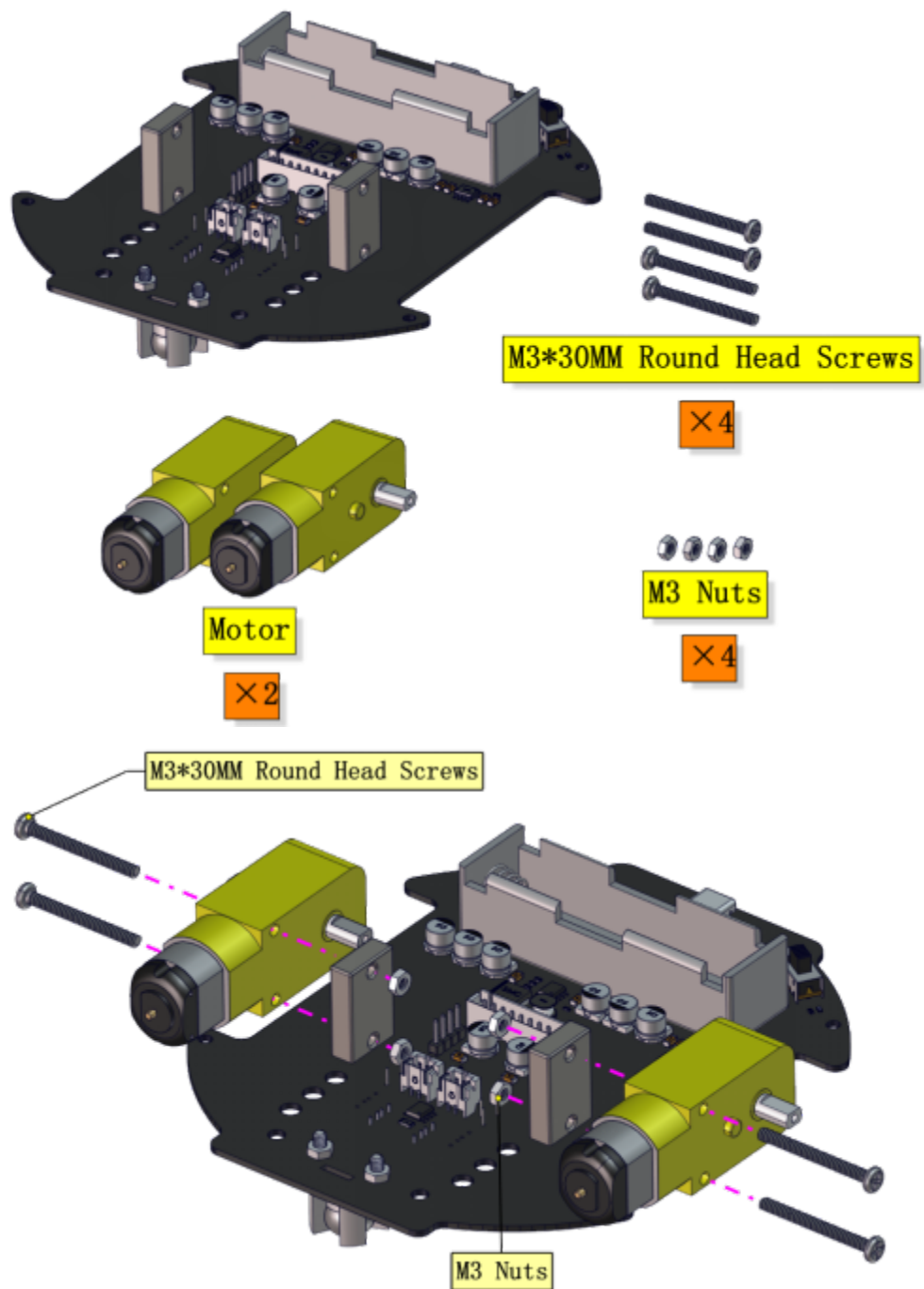


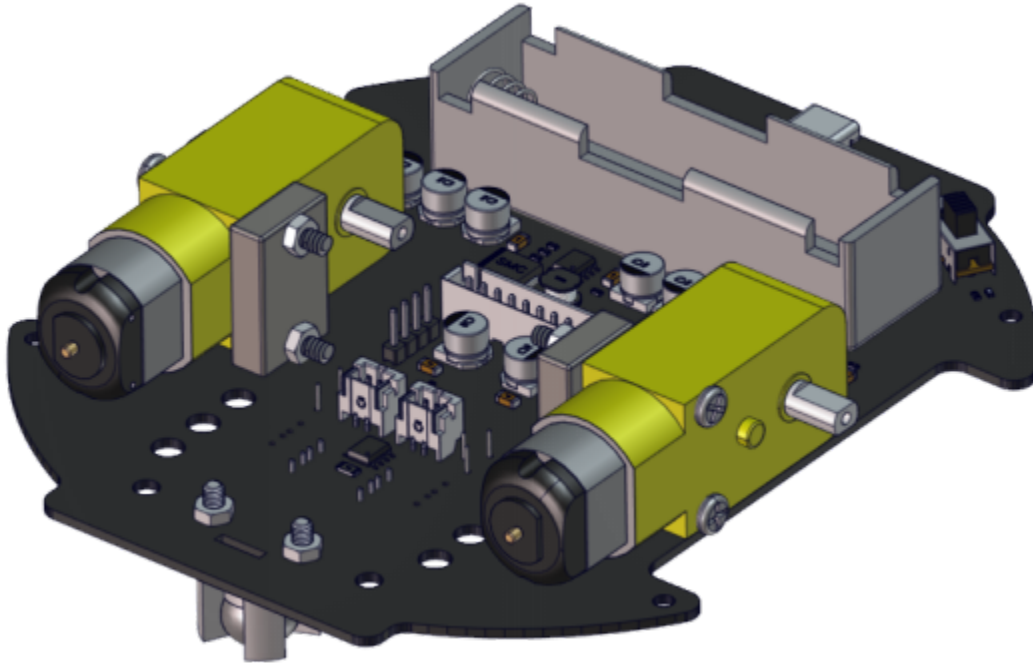
### Step 1



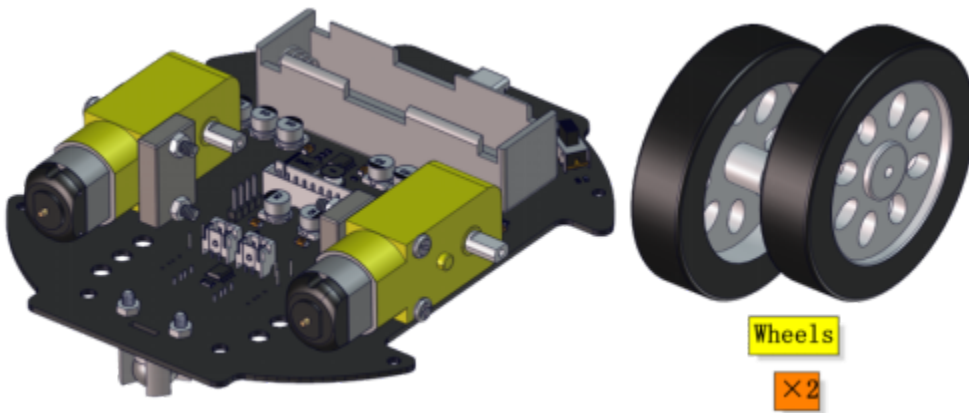


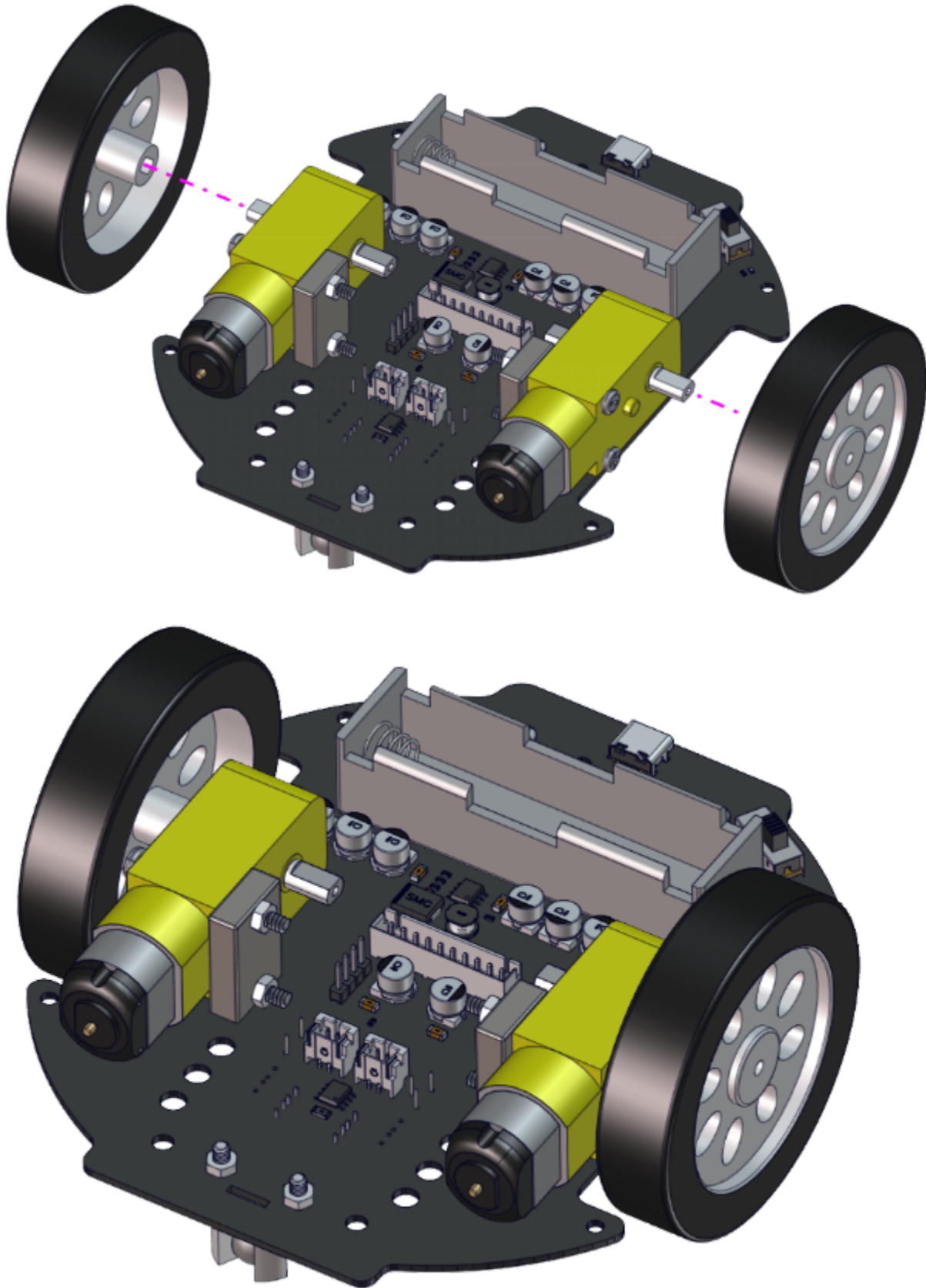
Step 2





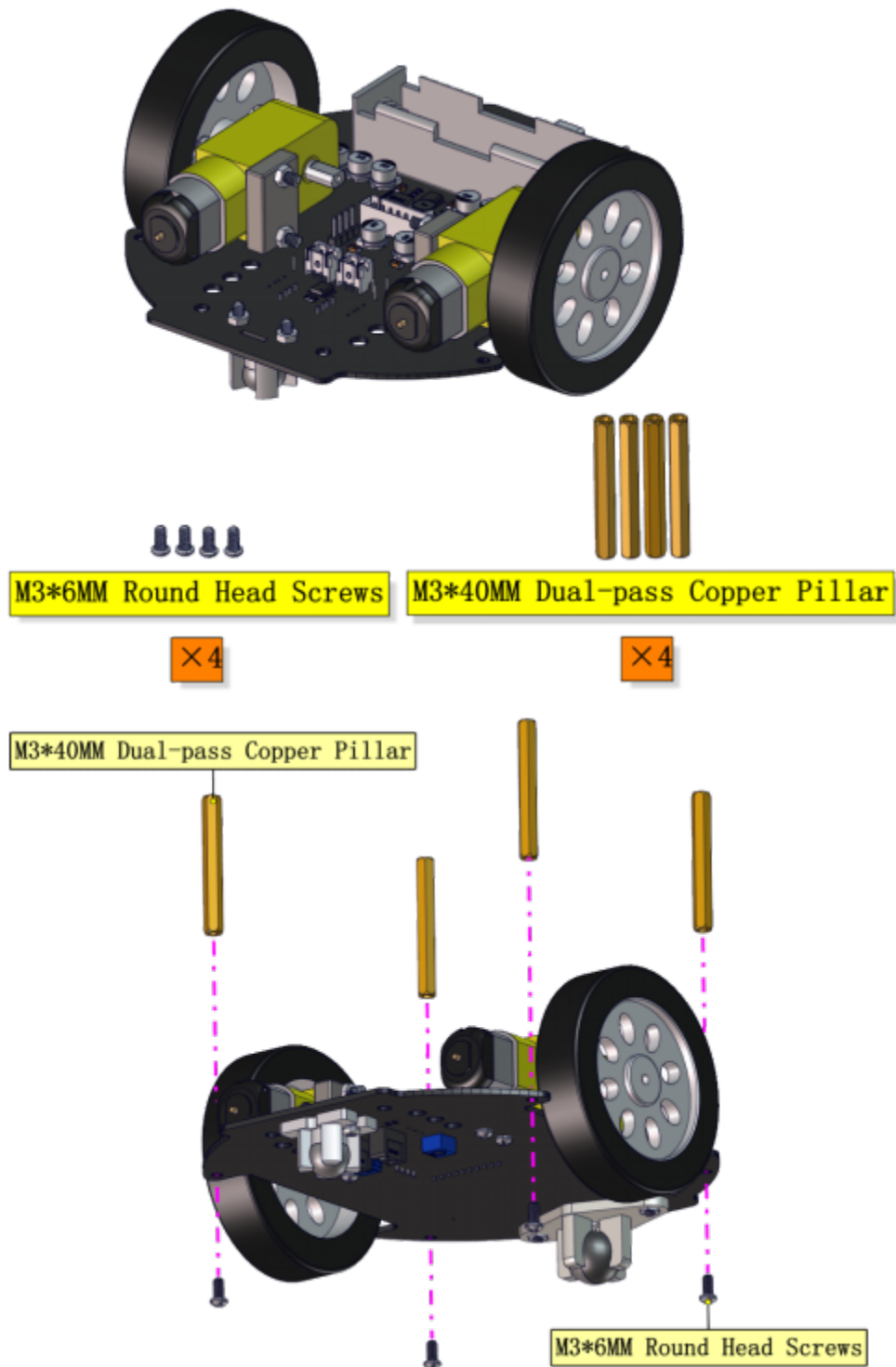
Step 3

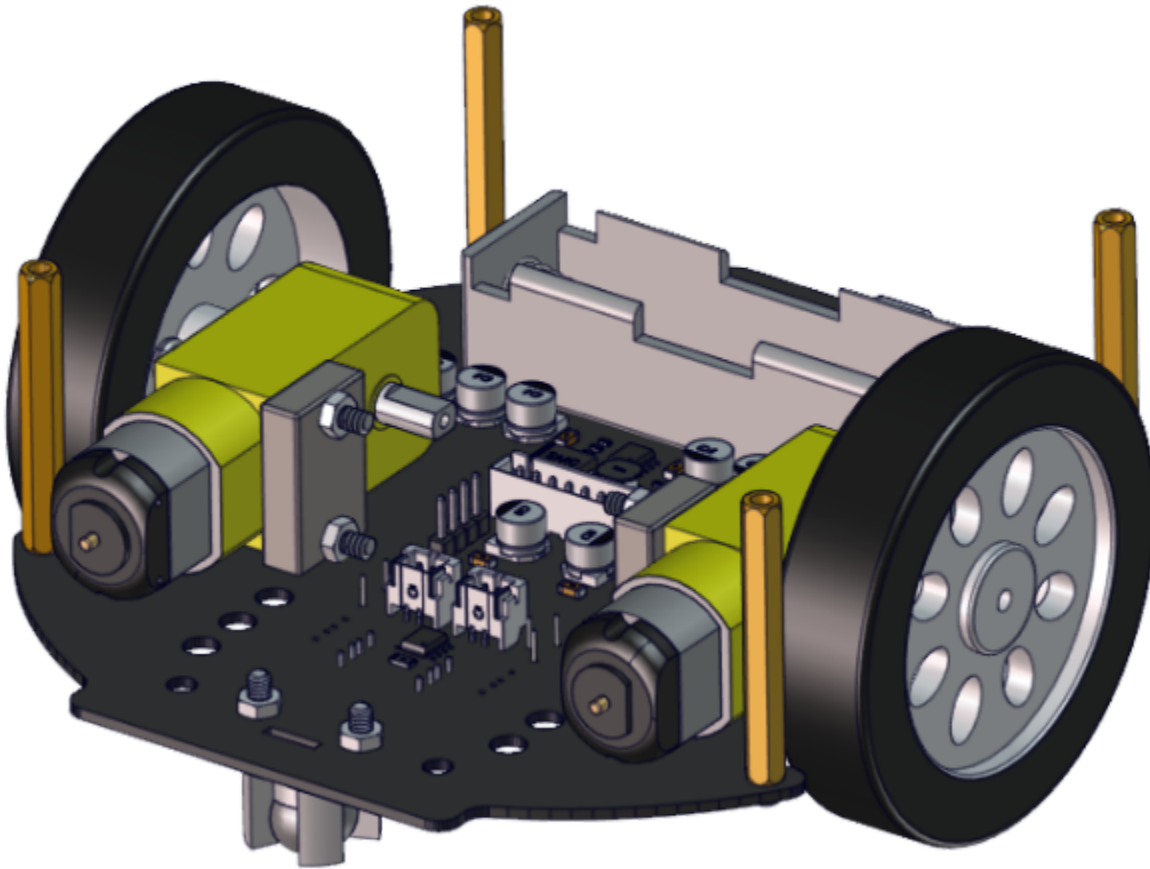




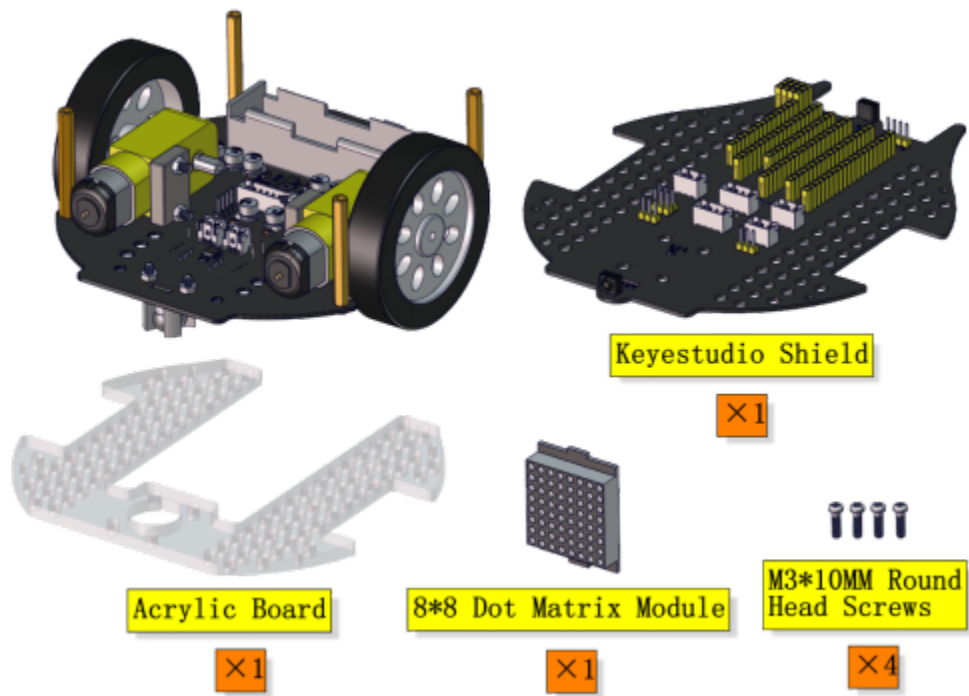


## Step 4

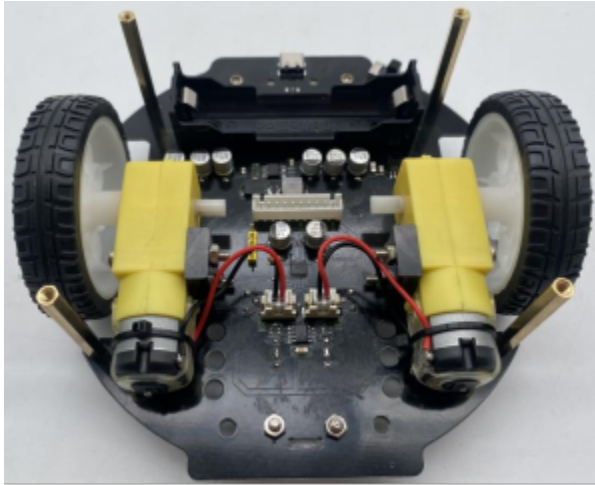




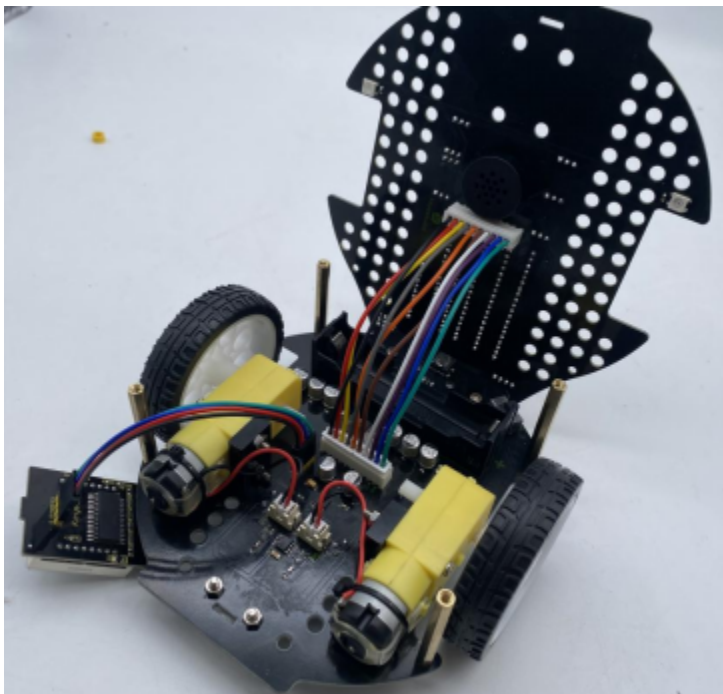
Step 5

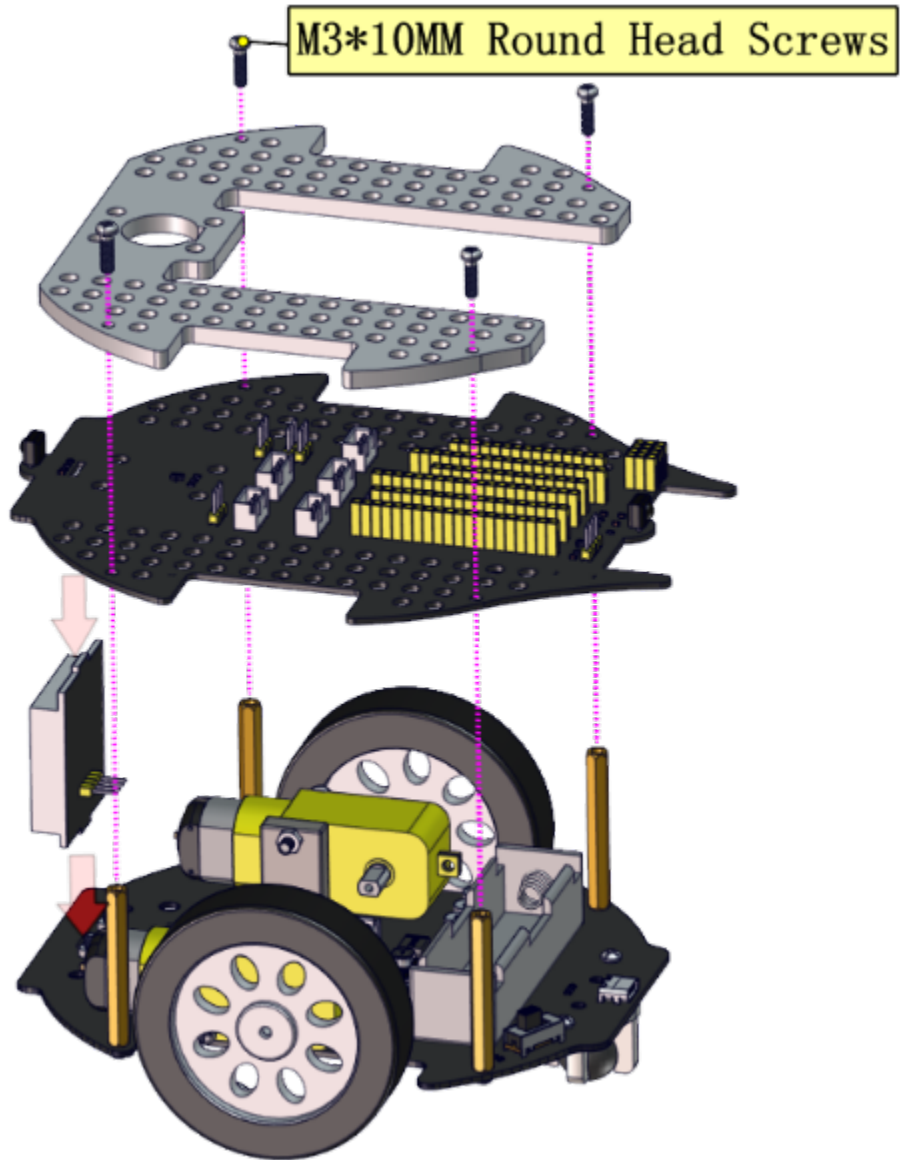


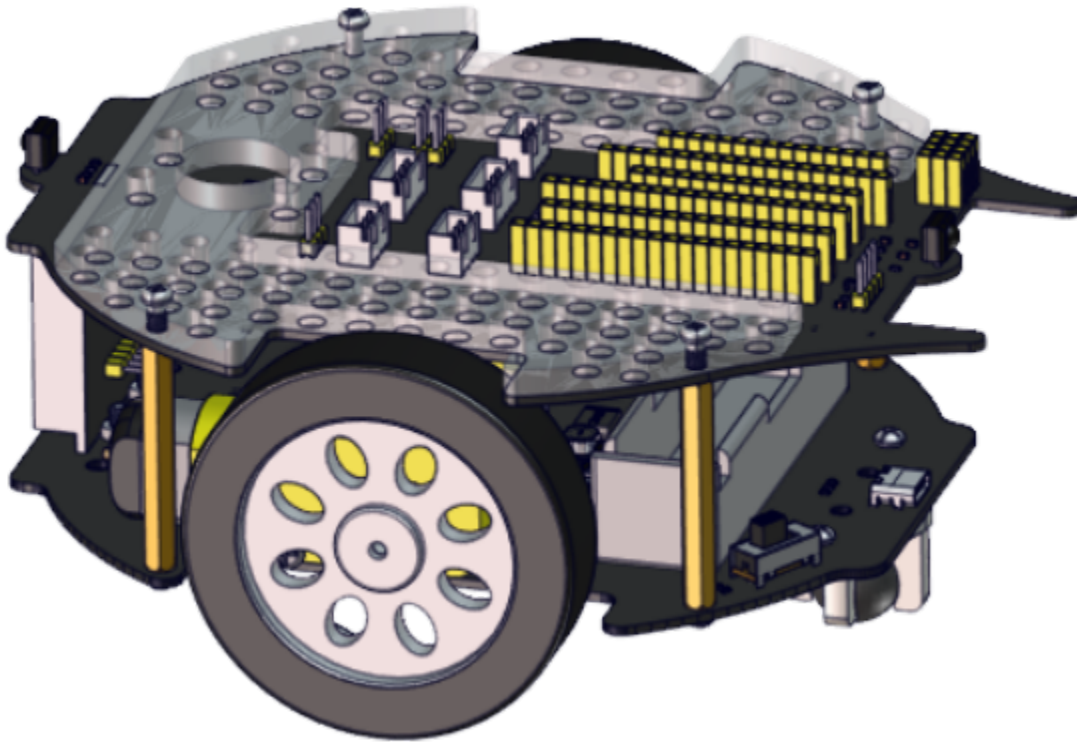
| Left Motor | Right Motor |
|------------|-------------|
| L          | R           |



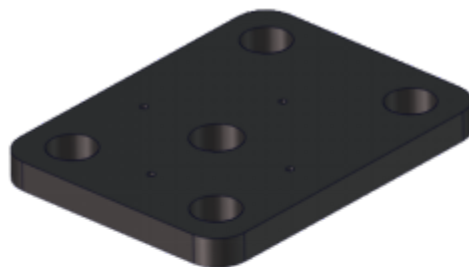
| 8*8 Dot Matrix Display | PCB |
|------------------------|-----|
| G                      | G   |
| 5V                     | 5V  |
| SDA                    | SDA |
| SCL                    | SCL |







### Step 6



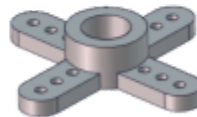
Acrylic Board

×1



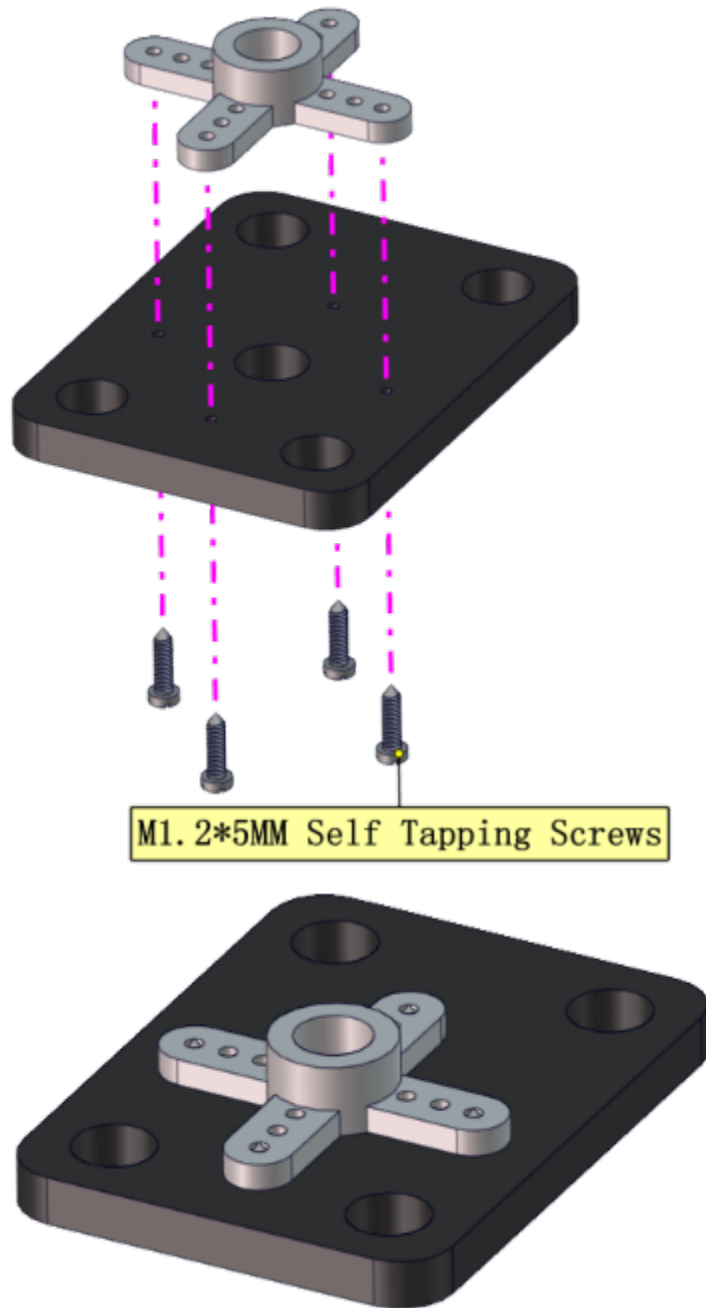
M1.2\*5MM Self Tapping Screws

×4

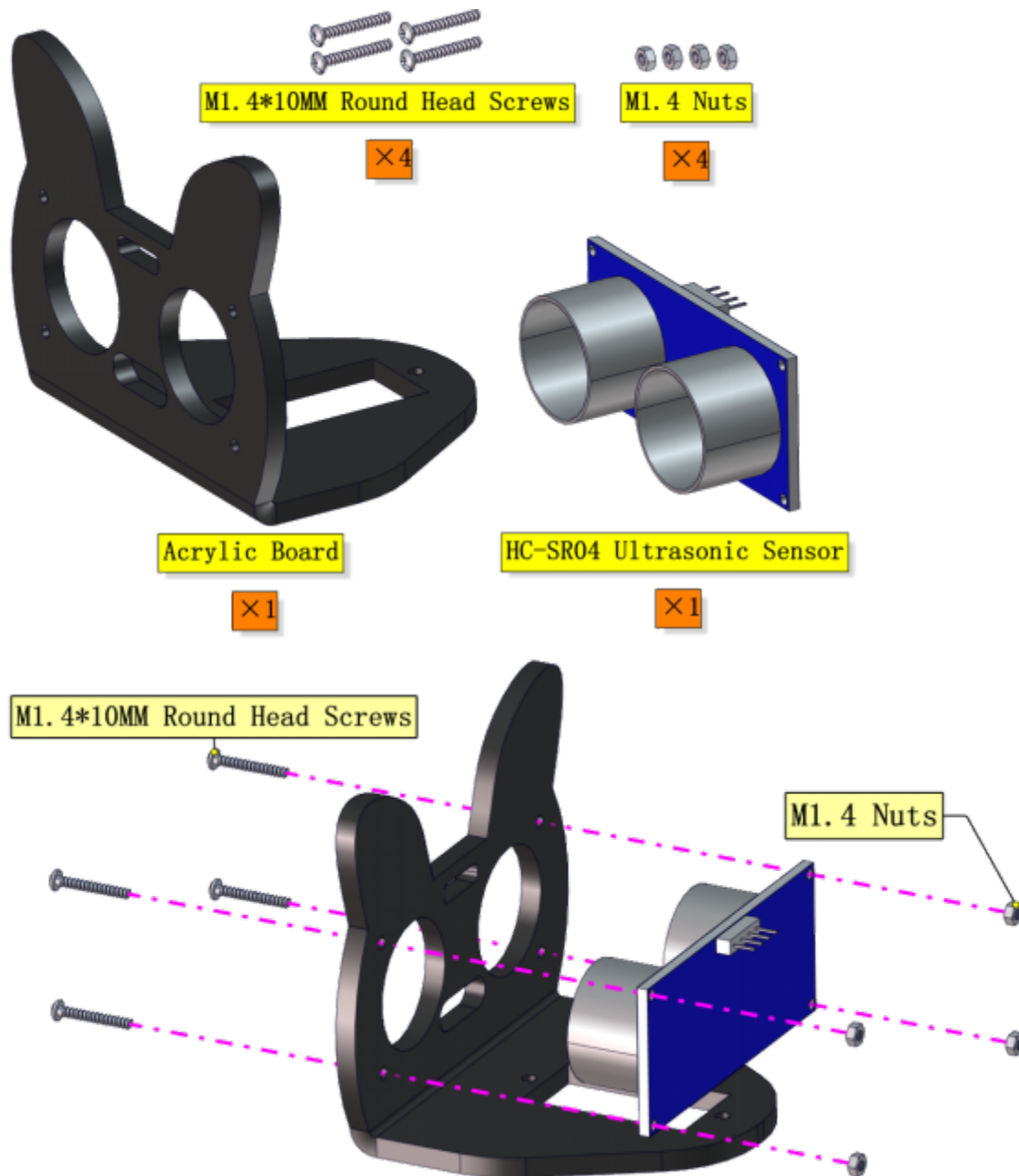


Control horn(belong to servo)

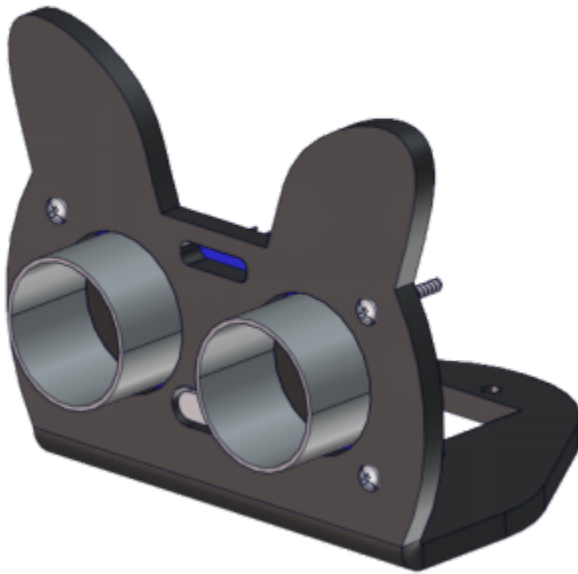
×1



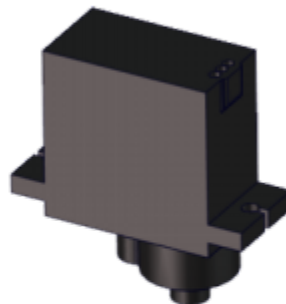
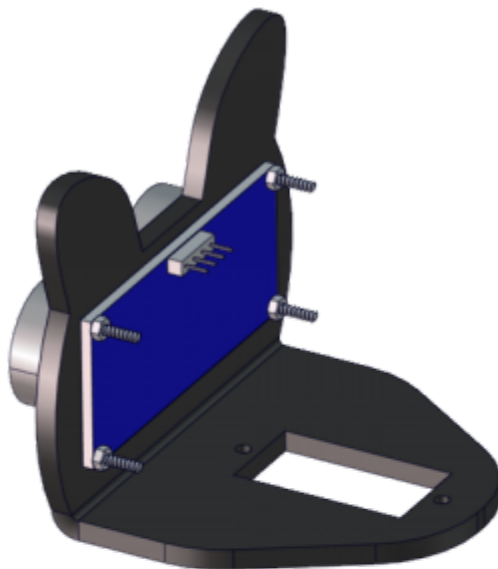
Step 7







Step 8



9G 180° Servo

×1



M2 Nuts

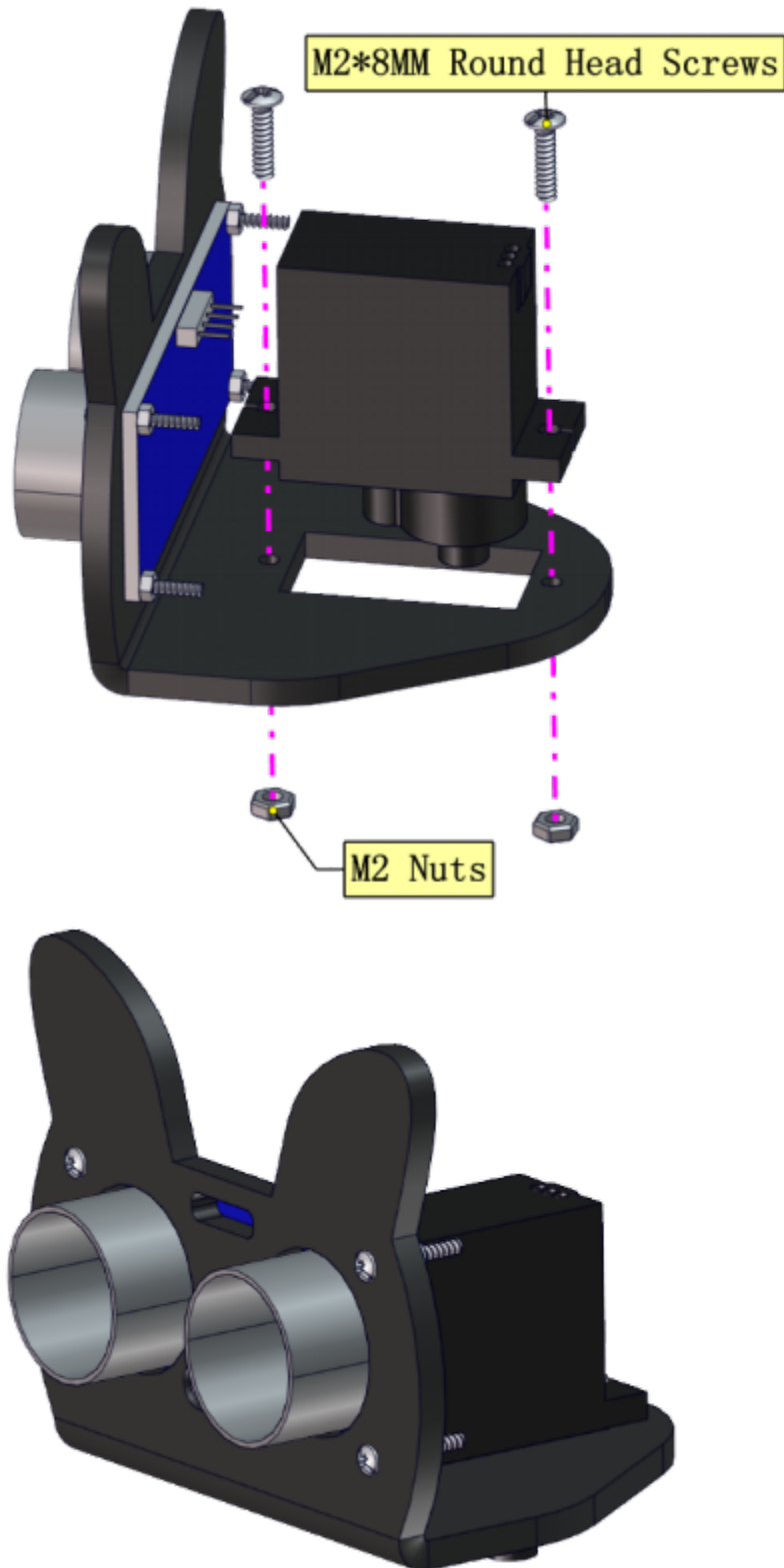
×2



M2\*8MM Round Head Screws

×2





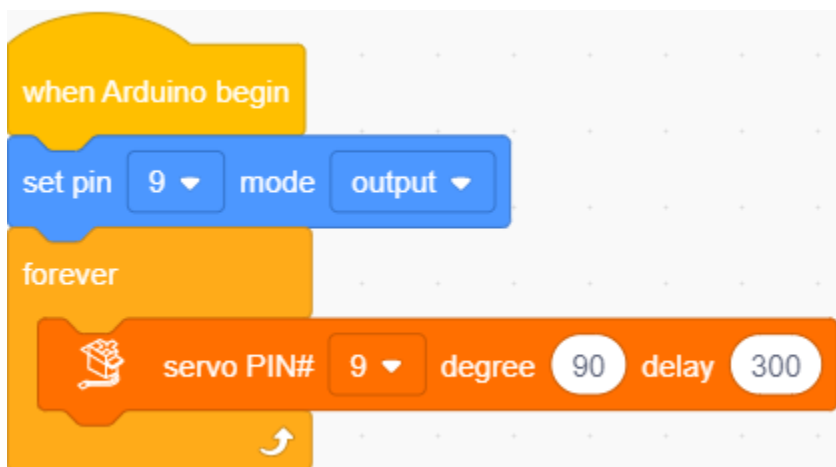
Step 9



Adjust the angle of the servo to 90 degree.

| Servo       | PCB  |
|-------------|------|
| Brown line  | G    |
| Red line    | 5V   |
| Orange line | S1D9 |

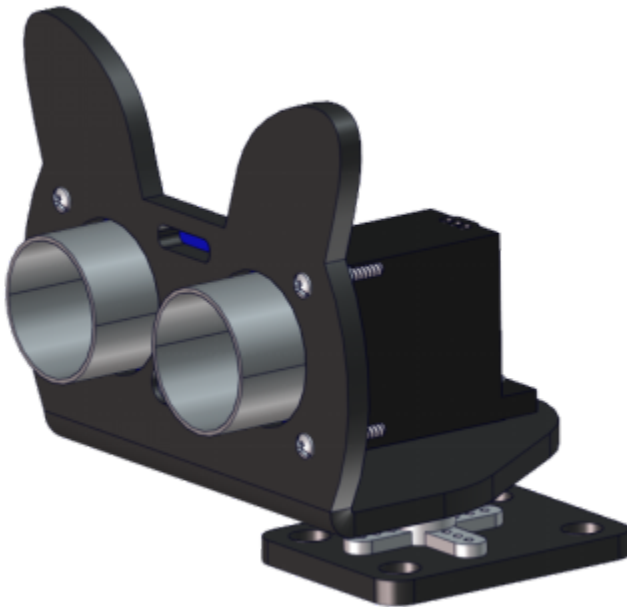
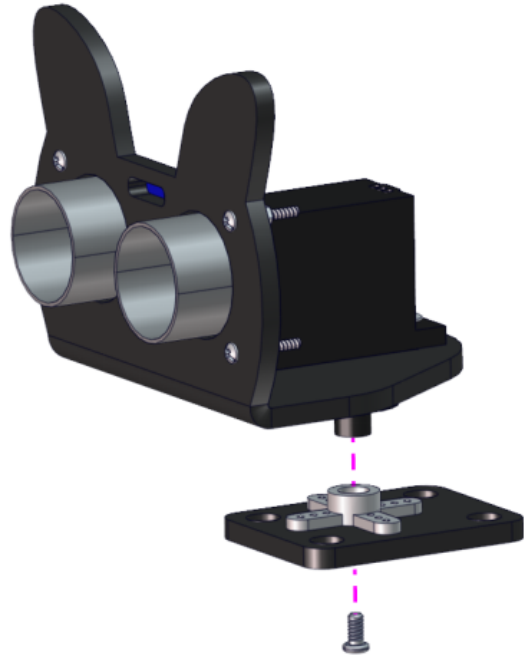
Writing the following code in kidsBlock software and upload it to the motherboard, or just open the code provided by us and upload it to the motherboard.



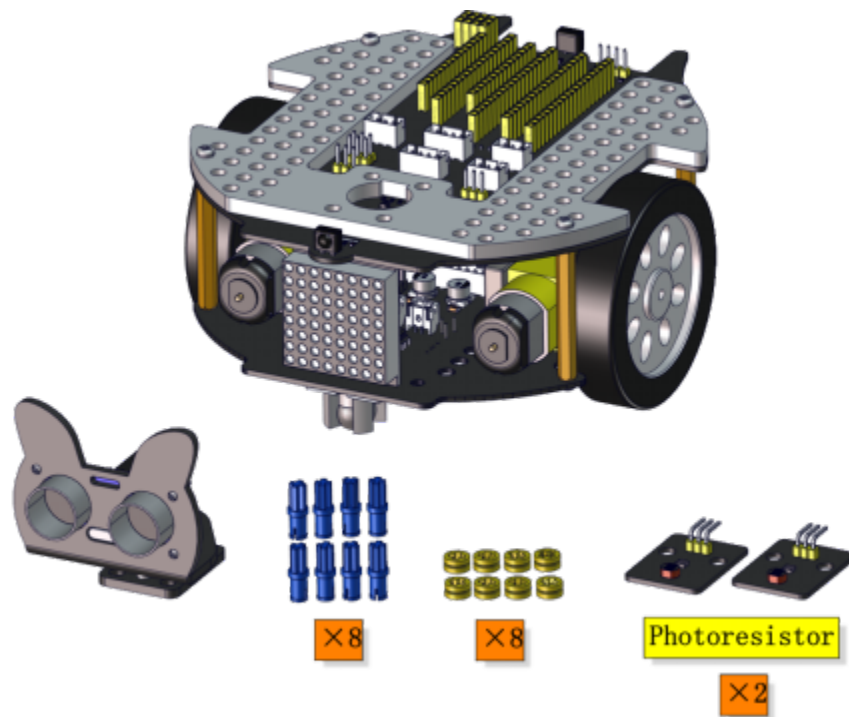
Keep the ultrasonic sensor parallel to the board

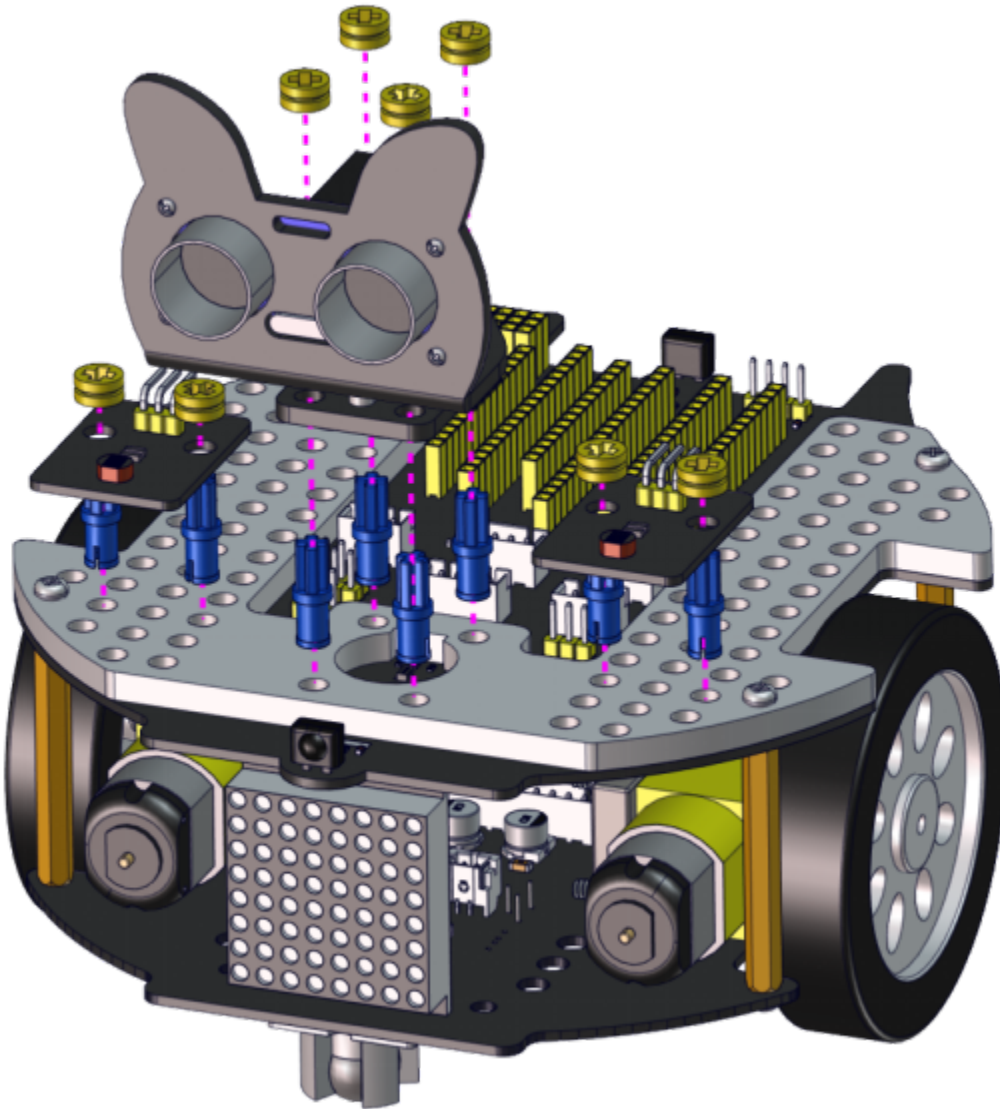


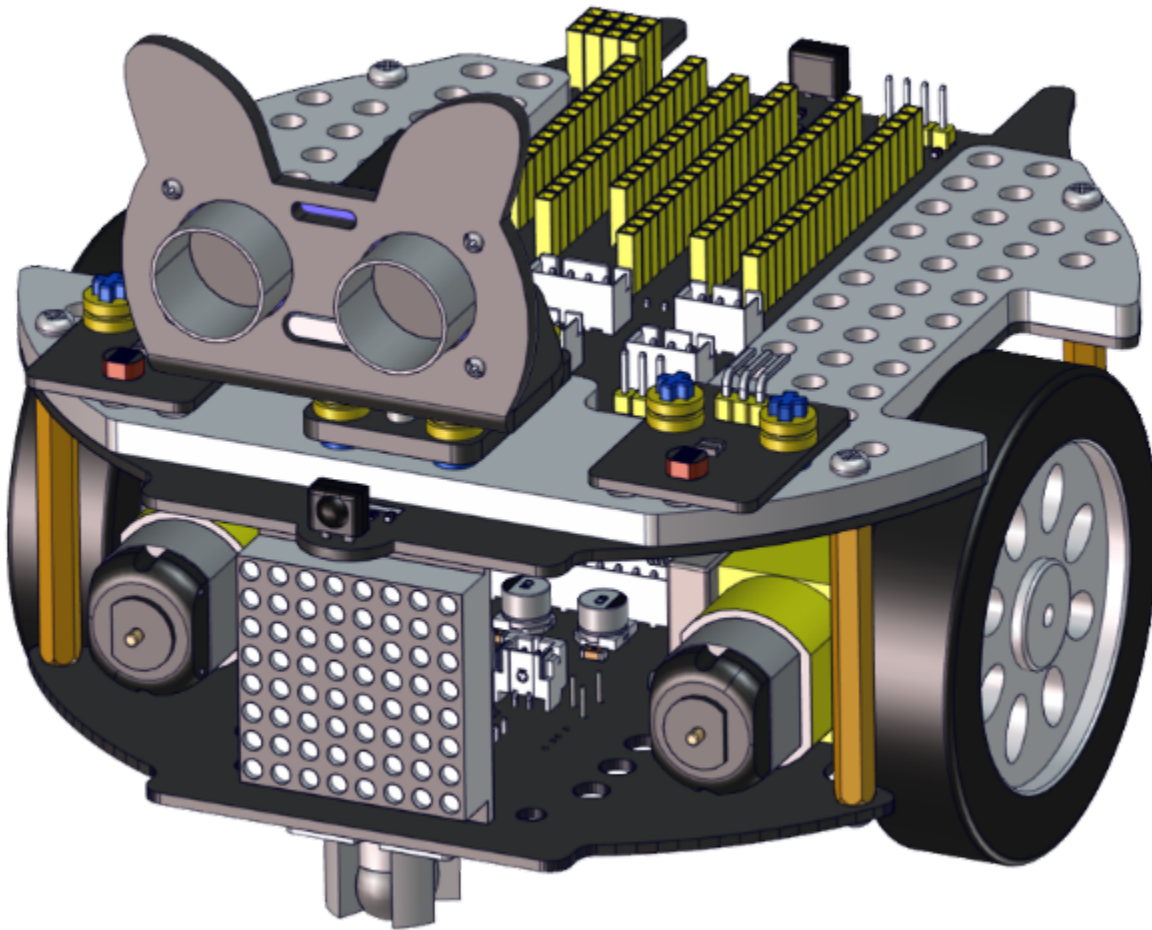
Note: Before assembling the servo, you need to adjust it to 90°, otherwise it will not work properly for the robot.



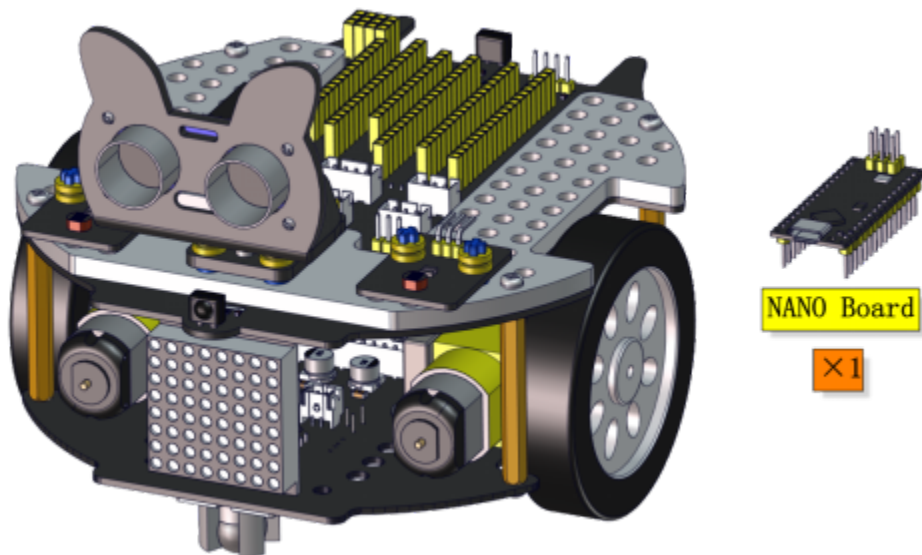
Step 10



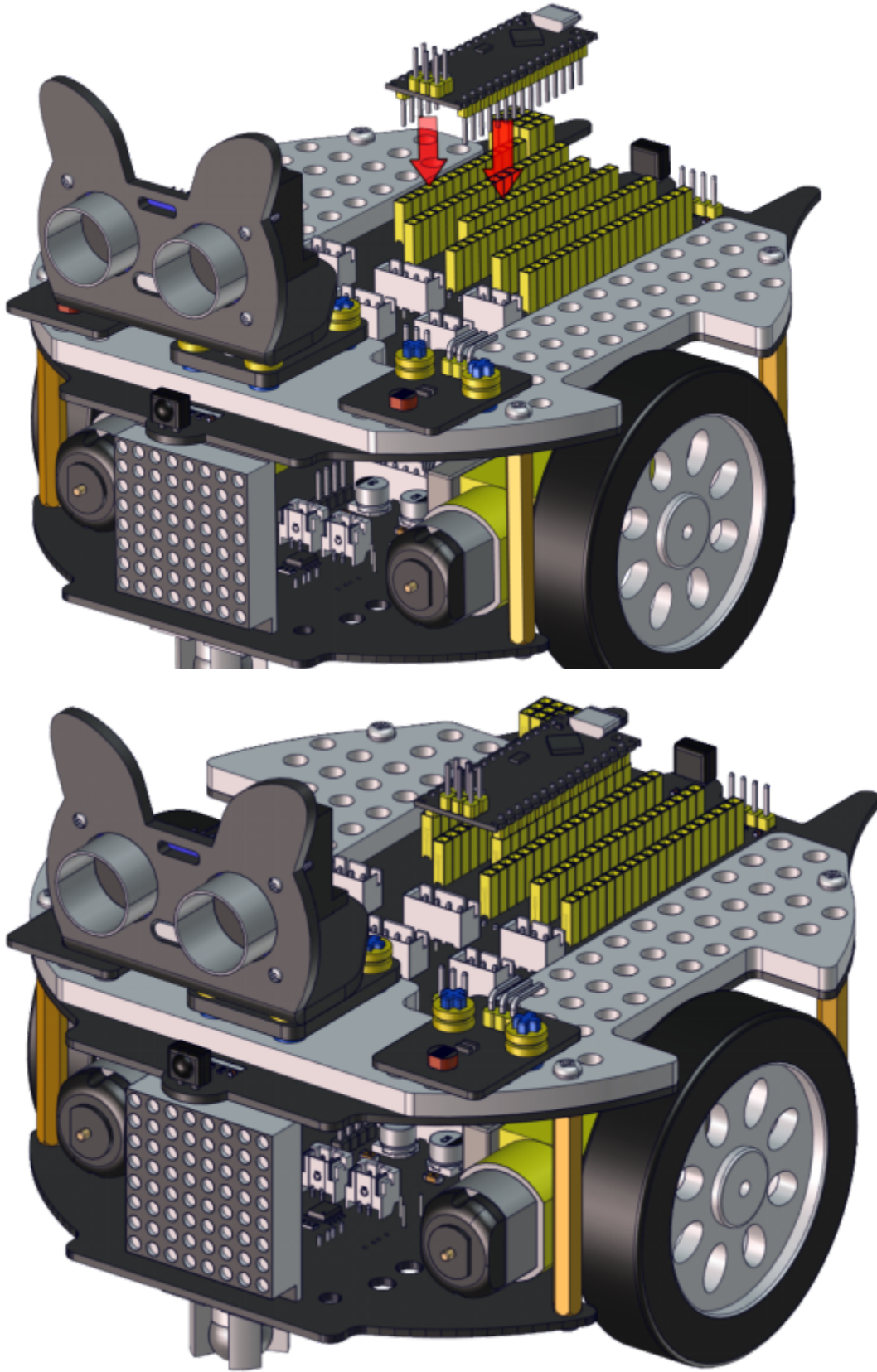




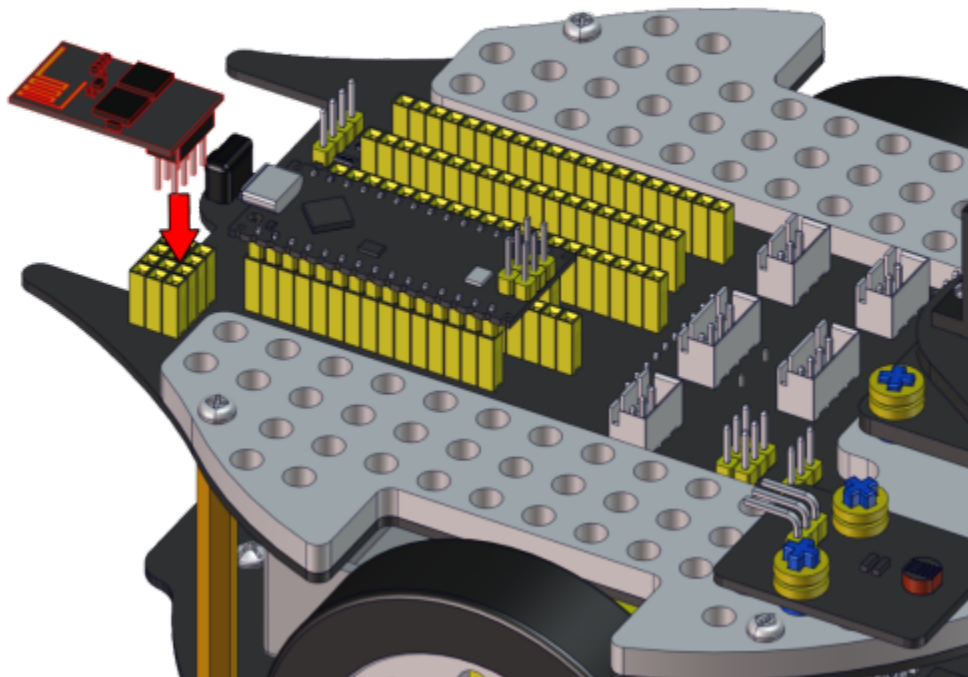
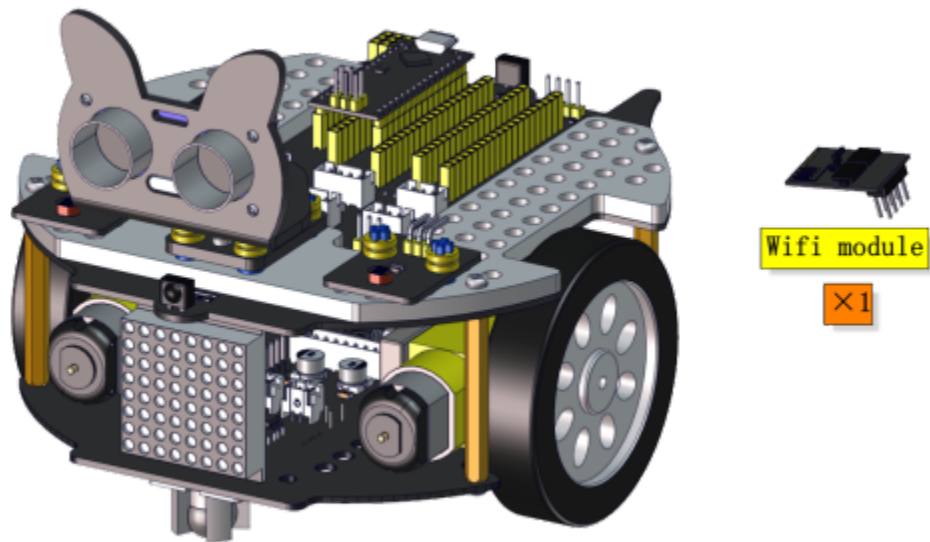
Step 11



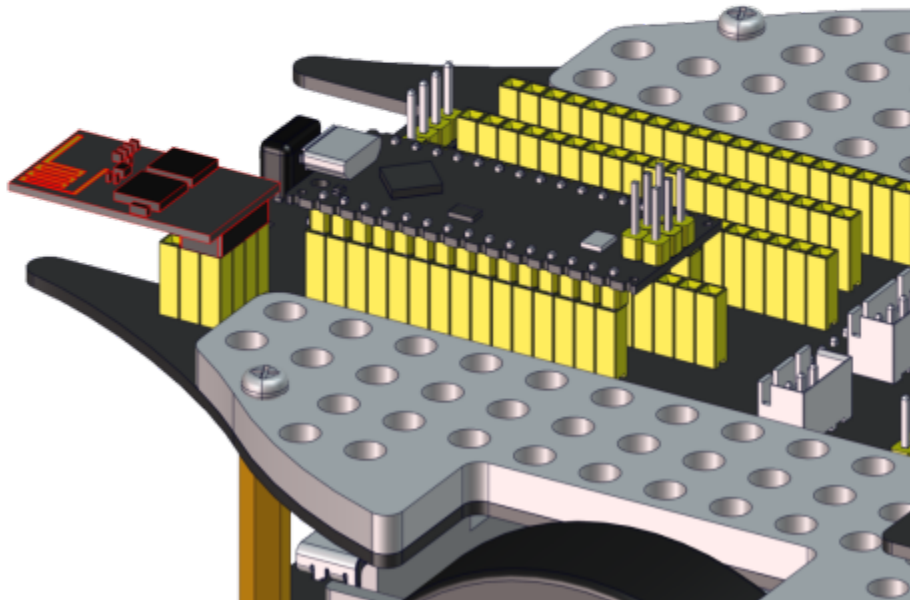


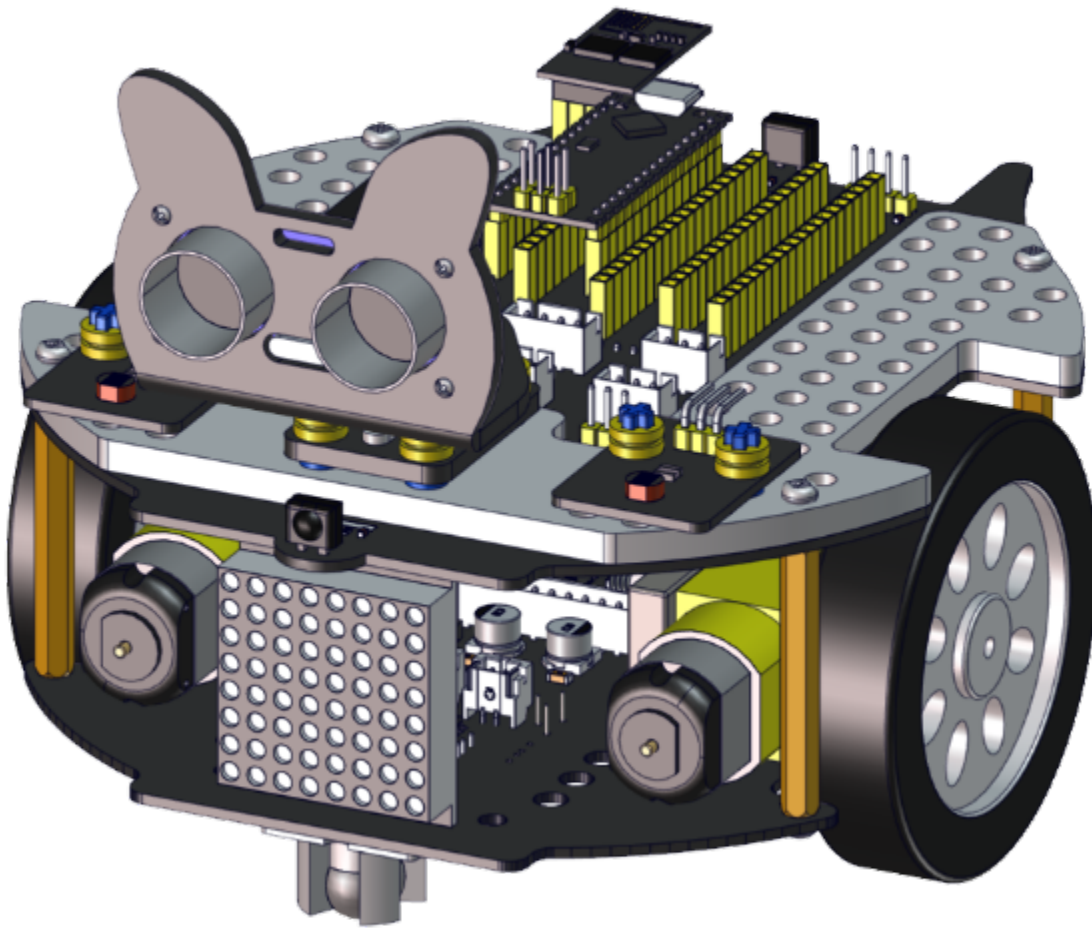


Step 12



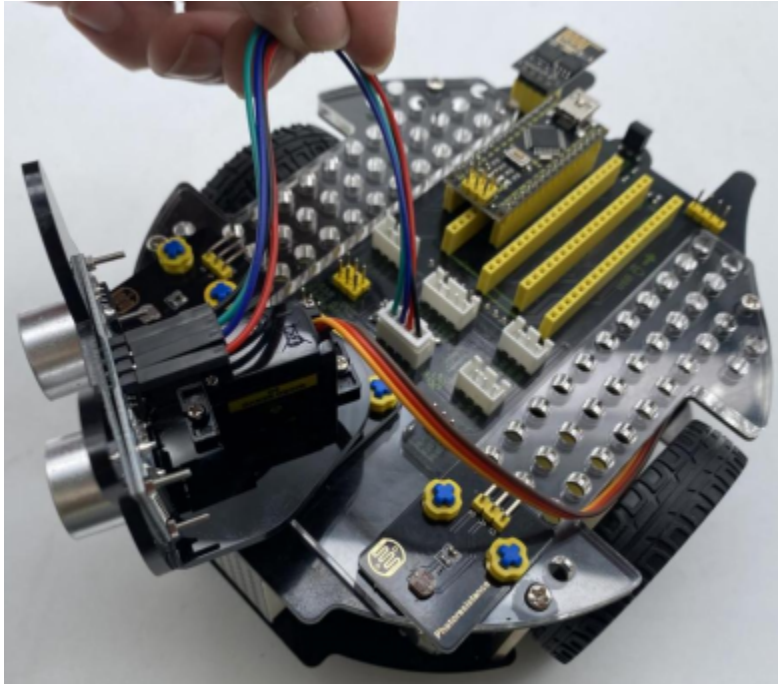






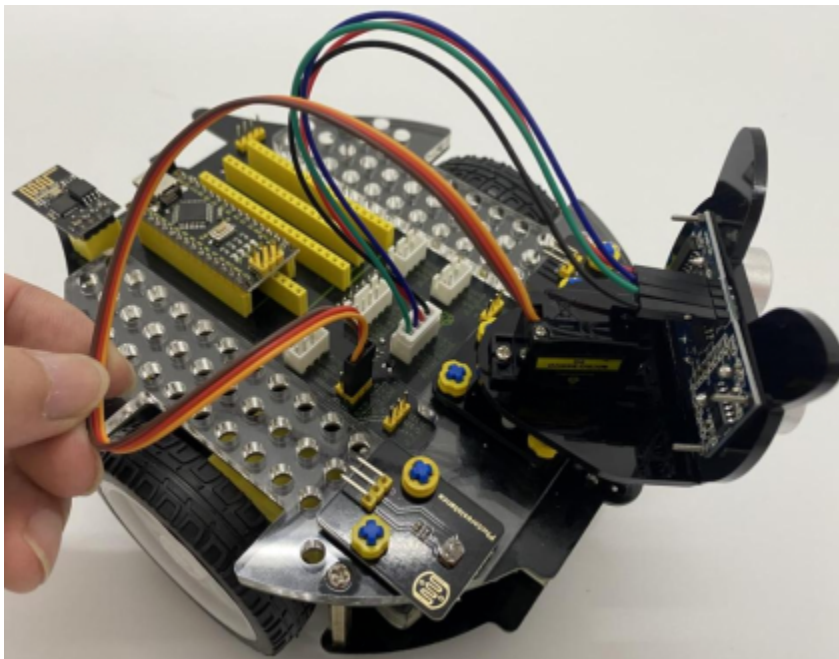
**Wire up the ultrasonic sensor**

| Ultrasonic Sensor | PCB  |
|-------------------|------|
| Vcc               | 5V   |
| Trig              | S2D8 |
| Echo              | S1D7 |
| Gnd               | G    |



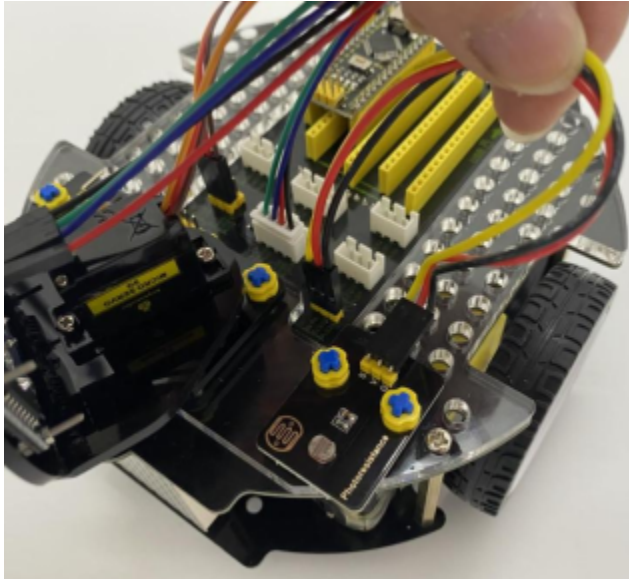
Wire up the servo

| Servo       | PCB  |
|-------------|------|
| Brown line  | G    |
| Red line    | 5V   |
| Orange line | S1D9 |



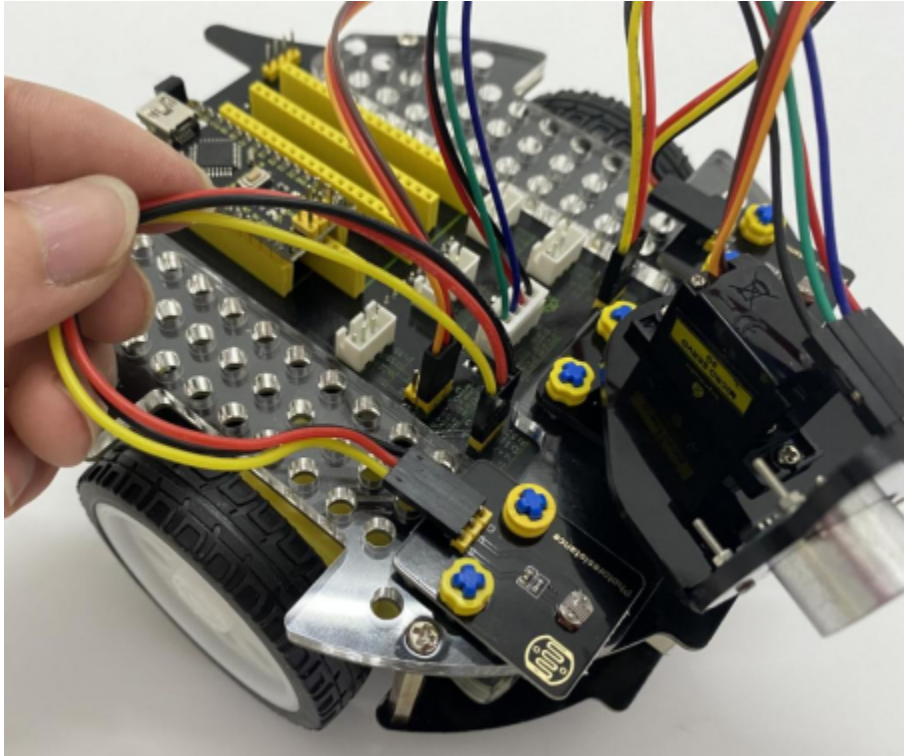
Wire up the left photoresistor

| Left photoresistor | PCB |
|--------------------|-----|
| G                  | G   |
| V                  | V   |
| S                  | SA6 |

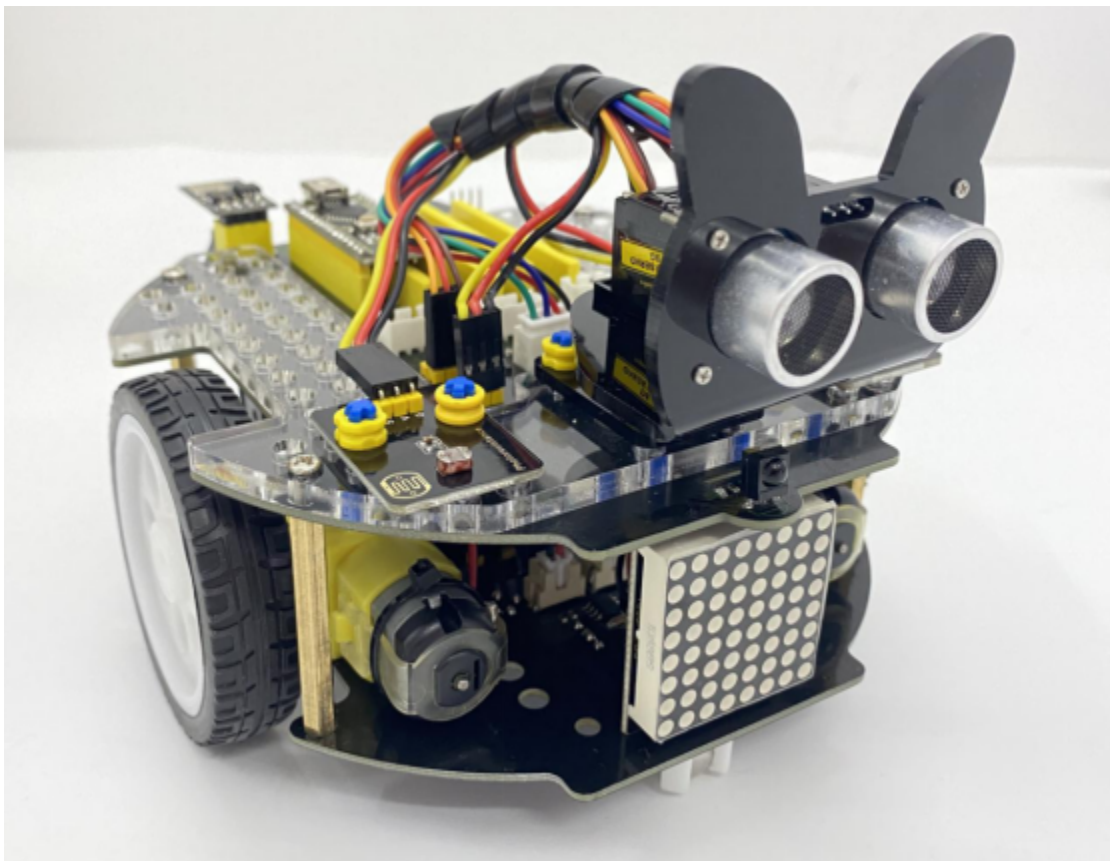


**Wire up the right photoresistor**

| Right photoresistor | PCB |
|---------------------|-----|
| G                   | G   |
| V                   | V   |
| S                   | SA7 |

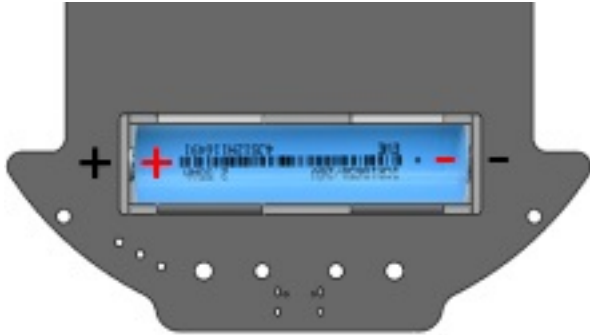


Beetle Robot Car





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



### 9.3 3. Projects

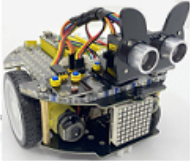



#### 9.3.1 Project 1: LED Blinking

##### (1)Description

There is an onboard LED (L) on our Arduino Nano board connected to D13. In this experiment, we WILL make this LED blink.

LED blinking is the most basic experimental project for Arduino enthusiasts. Let's get started.

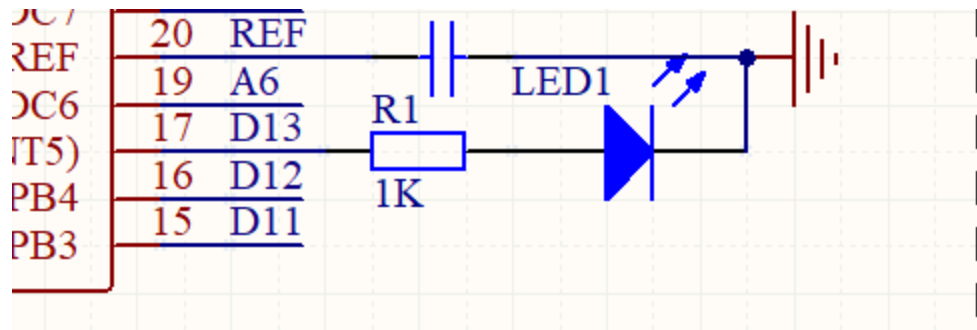
##### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

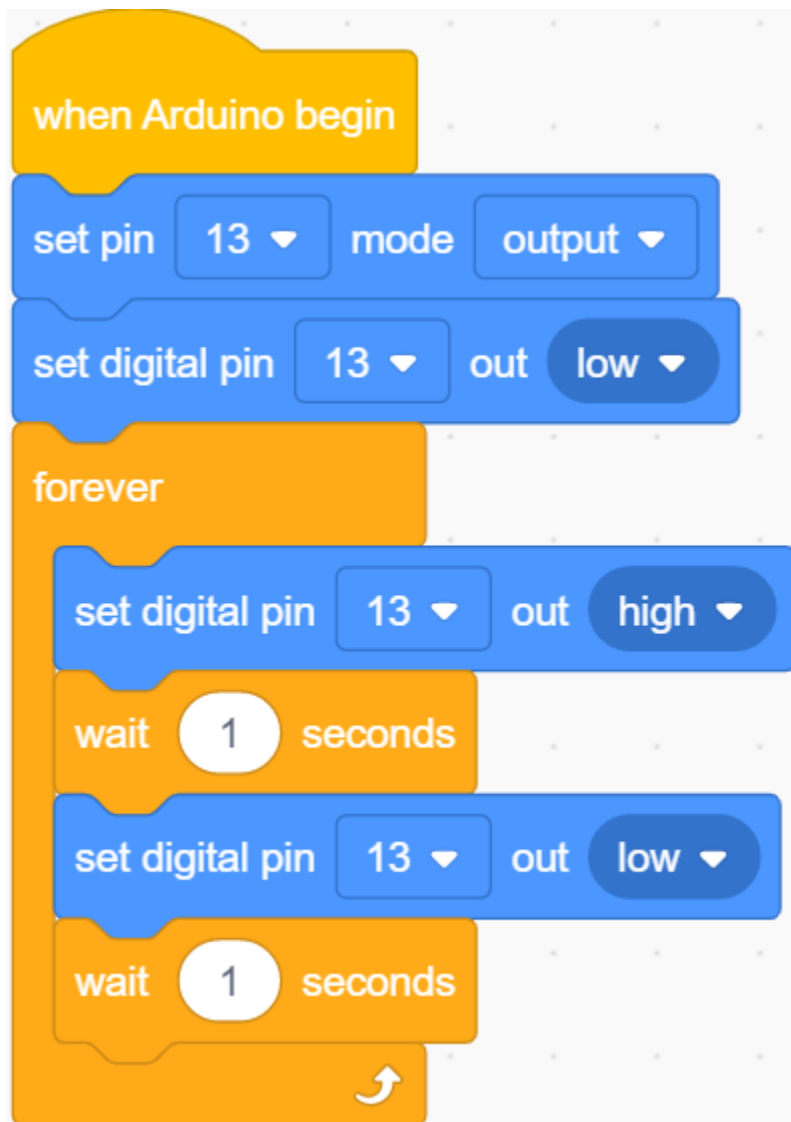
##### (3)Knowledge

###### On-board LED

LED, the abbreviation of light emitting diodes, consists of Ga, As, P, N chemical compounds and so on. It is easy to control through the IO port(D13) of the Arduino Nano board.

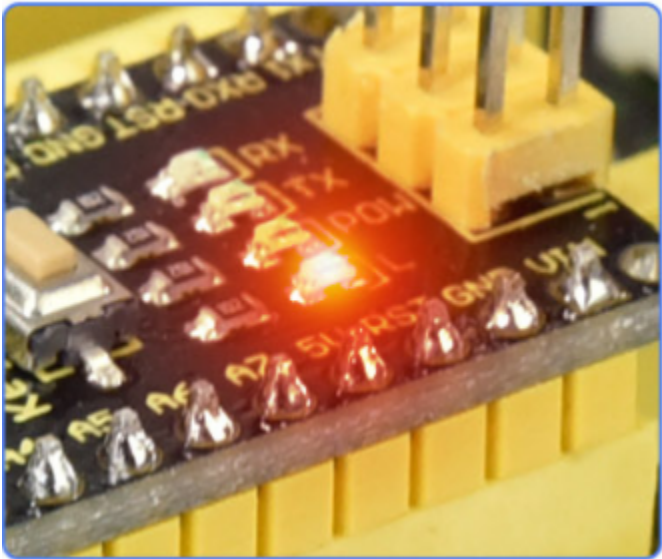


#### (4)Test Code



(5)Test Result

Upload the test code to the Arduino Nano board and power up with a USB cable. Then the on-board LED will flash.



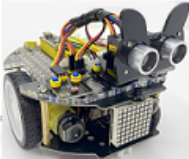



9.3.2 Project 2: 6812 RGB

(1)Description

There are 4 RGB LEDs can be widely used in the decoration of buildings, bridges, roads, gardens, courtyards and so on by colors adjustment.

In this experiment, we will demonstrate different lighting effects with them.

(2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

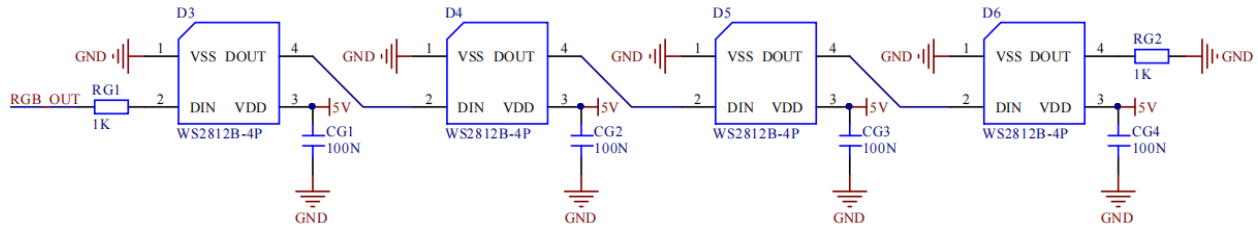


### (3)Component Knowledge

#### SK6812RGB:

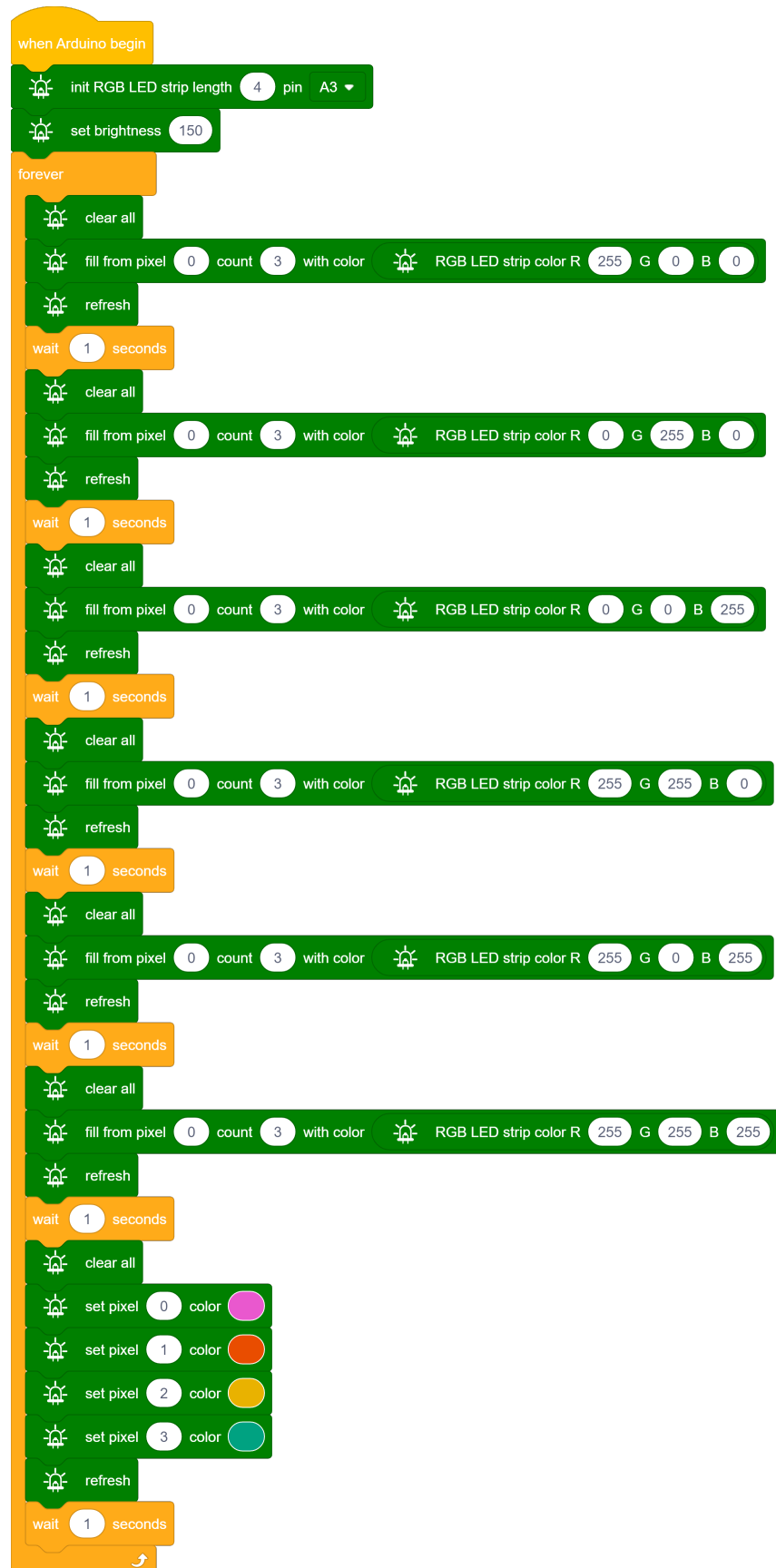
From the schematic diagram, we can see that these four pixel lighting beads are all connected in series. In fact, no matter how many they are, we can use a pin to control a light and let it display any color. The pixel point contains a data latch signal shaping amplifier drive circuit, a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.

The data protocol adopts a single-wire zero-code communication method. After the pixel is powered up and reset, the S terminal receives the data transmitted from the controller. The first 24bit data sent is extracted by the first pixel and sent to the data latch of the pixel.



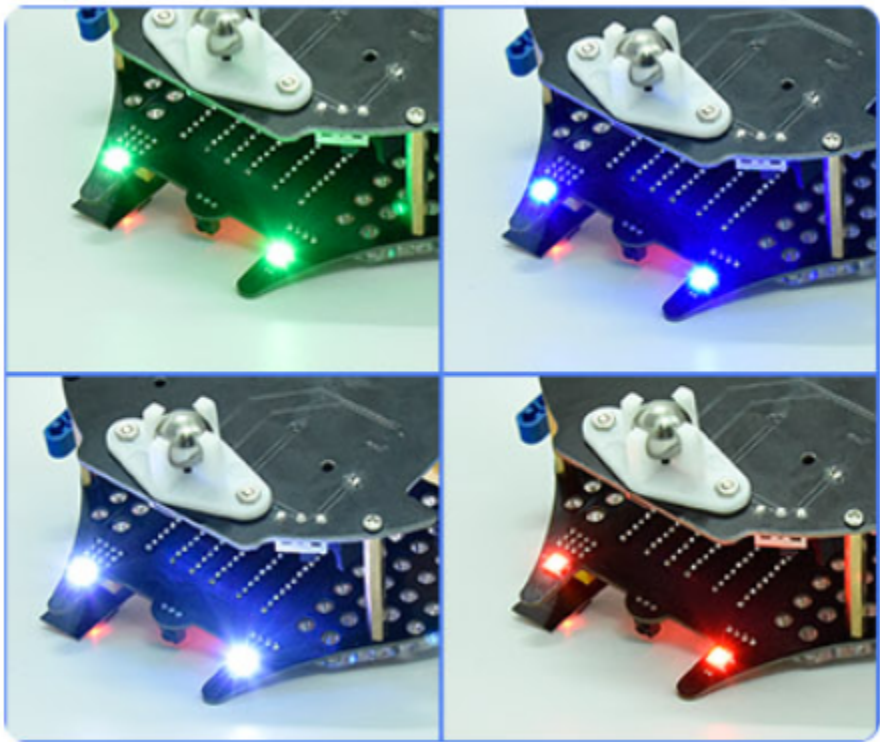
### (4)Test Code

The SK6812RGB on the PCB board is controlled by the IO port (A3).



(5)Test Result

Upload the test code to the Arduino Nano board and power up by a USB cable. Then the four RGB lights on the PCB demonstrate multi-color light effect.



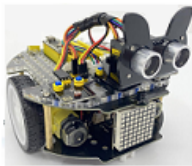



9.3.3 Project 3: Play Music

(1)Description

There is a power amplifier component on the expansion board, which is often used to play music and serve as an external amplifying device for music playback devices.

In this experiment, we use the speaker amplifier component to play music.

(2)Components Required

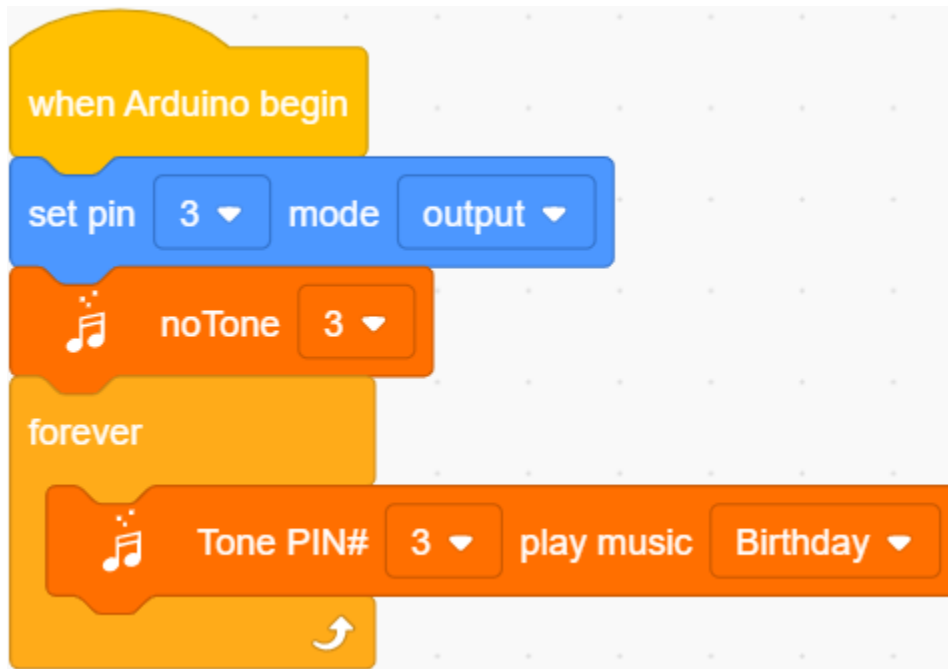
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

### (3)Knowledge

Power amplifier modules(equivalent to a passive buzzer) don't have internal oscillation circuits. The power amplifier module can chime sounds with different frequency when power it up.

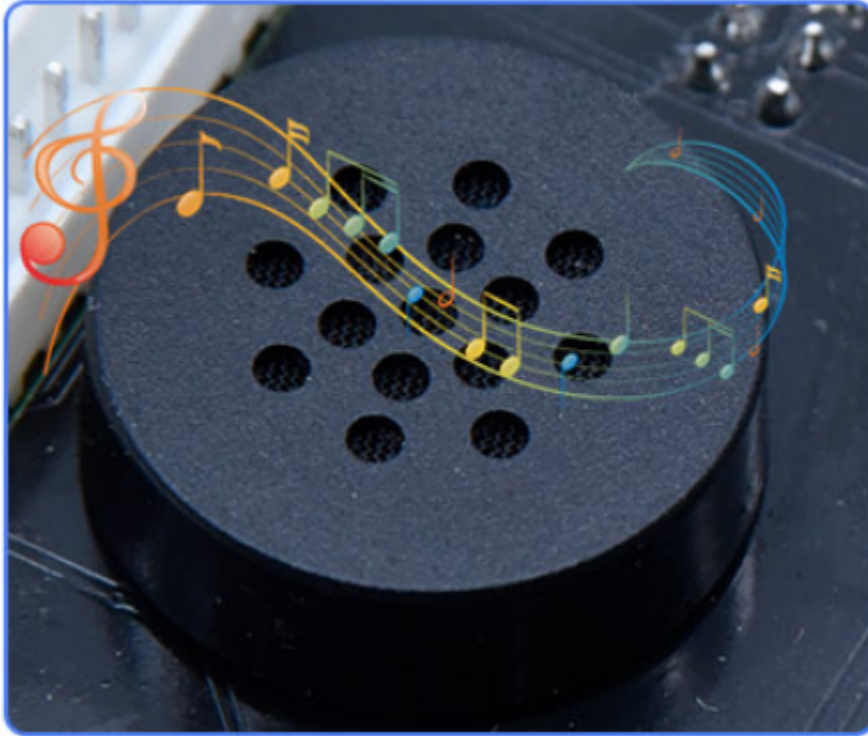
### (4)Test Code

The speaker component on the PCB board is controlled by the D3 of the Arduino Nano board.



### (5)Test Result

Upload the test code to the Arduino Nano board and power up with a USB cable. Then the power amplifier component will play music.







### 9.3.4 Project 4: 8\*8 Dot Matrix

#### (1)Description

Composed of LED emitting tube diodes, the 8\*8 LED dot matrix are applied widely to public information display like advertisement screen and bulletin board, by controlling LED to show words, pictures and videos, etc.

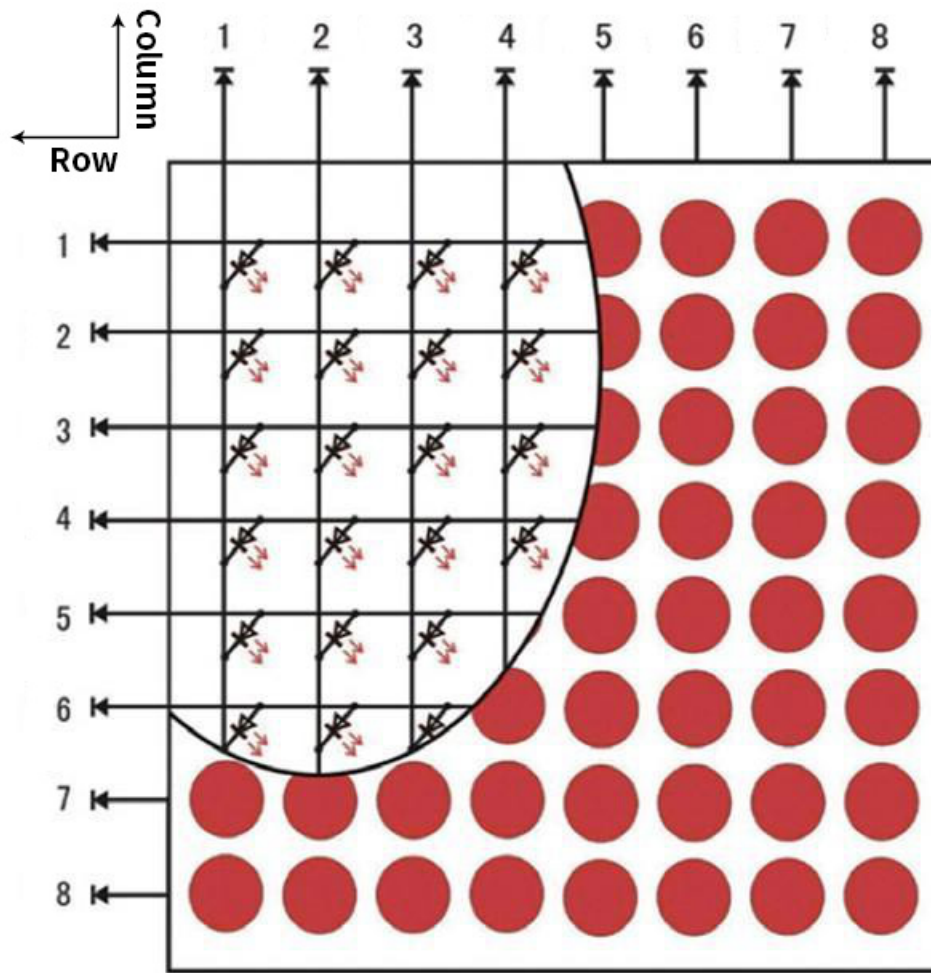
#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

**(3)Knowledge**

There are different types of matrices, including 4×4, 8×8 and 16×16 and etc. It contains 64 LEDs.

The inner structure of 8×8 dot matrix is shown below.

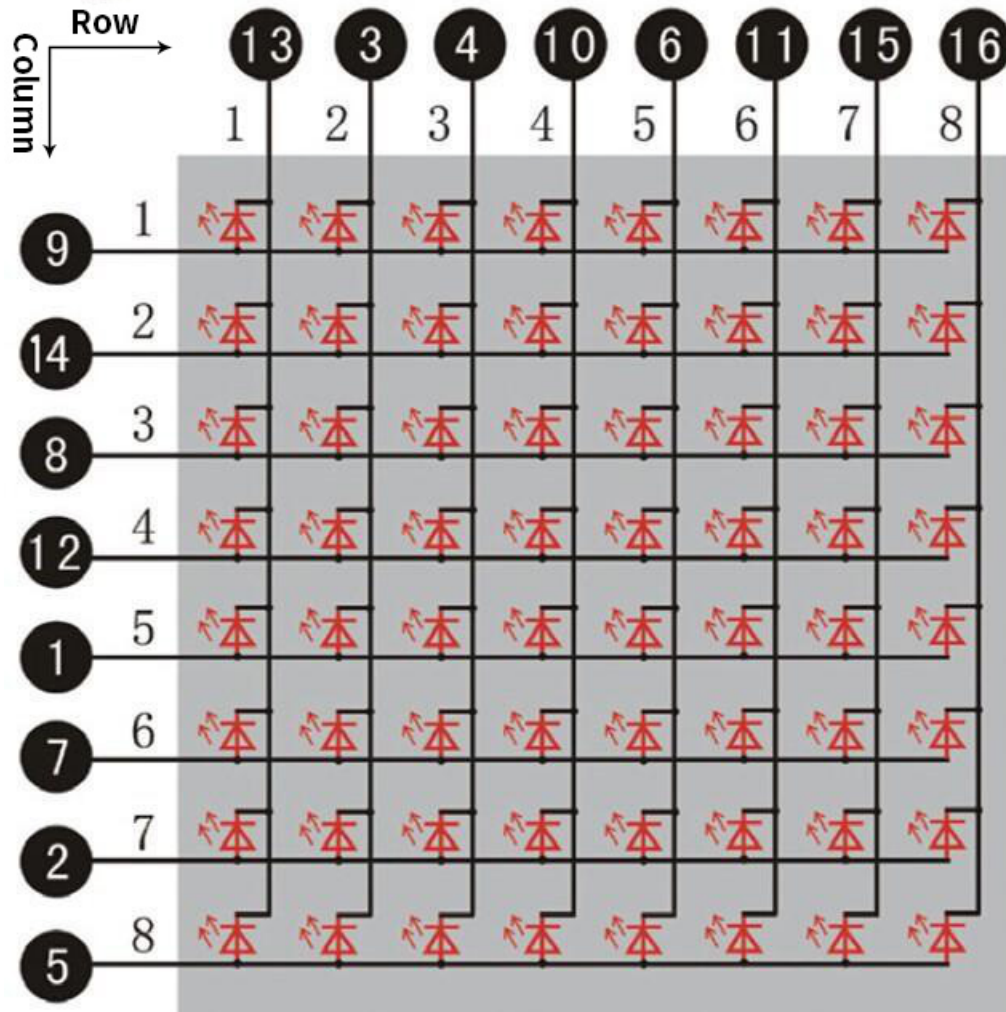


Every LED is installed on the cross point of row line and column line. When the voltage on a row line increases, and the voltage on the column line reduces, the LED on the cross point will light up. 8×8 dot matrix has 16 pins. Put the silk-screened side down and the numbers are 1, 8, 9 and 16 in anticlockwise order as marked below.



The definition inner pins are shown below:





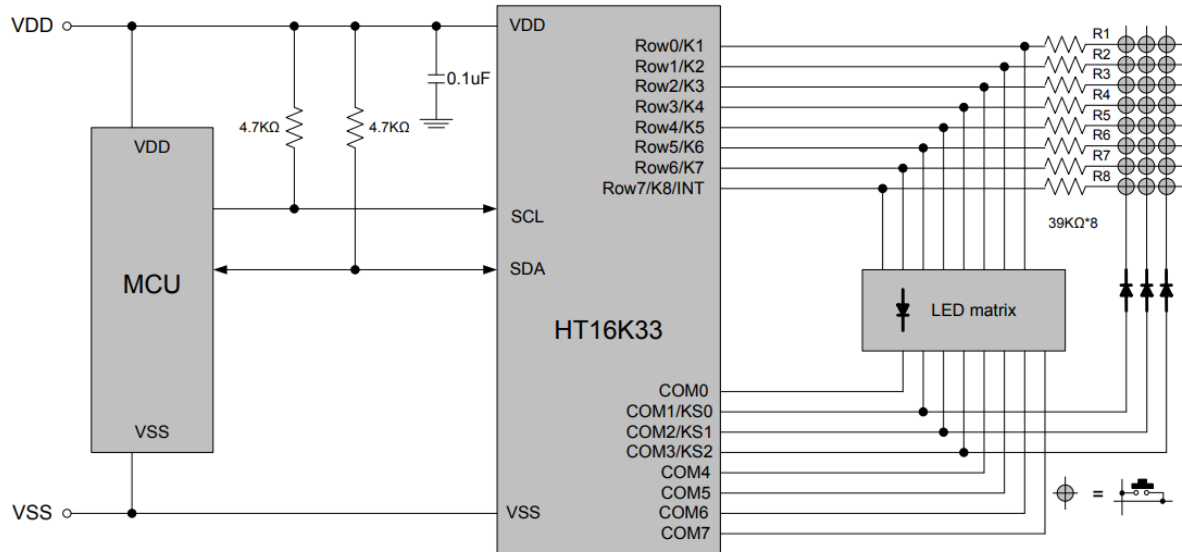
For instance, to light up the LED on row 1 and column 1, you should increase the voltage of pin 9 and reduce the voltage of pin 13.

### HT16K33 8X8 Dot Matrix

The HT16K33 is a memory mapping and multi-purpose LED controller driver. The max. Display segment numbers in the device is 128 patterns (16 segments and 8 commons) with a 13\*3 (MAX.) matrix key scan circuit. The software configuration features of the HT16K33 makes it suitable for multiple LED applications including LED modules and display subsystems. The HT16K33 is compatible with most microcontrollers and communicates via a two-line bidirectional I2C-bus.

The picture below is the working schematic of HT16K33 chip:





We design the drive module of 8\*8 dot matrix based on the above principle. We could control the dot matrix by I2C communication and two pins of microcontroller, according to the above diagram.

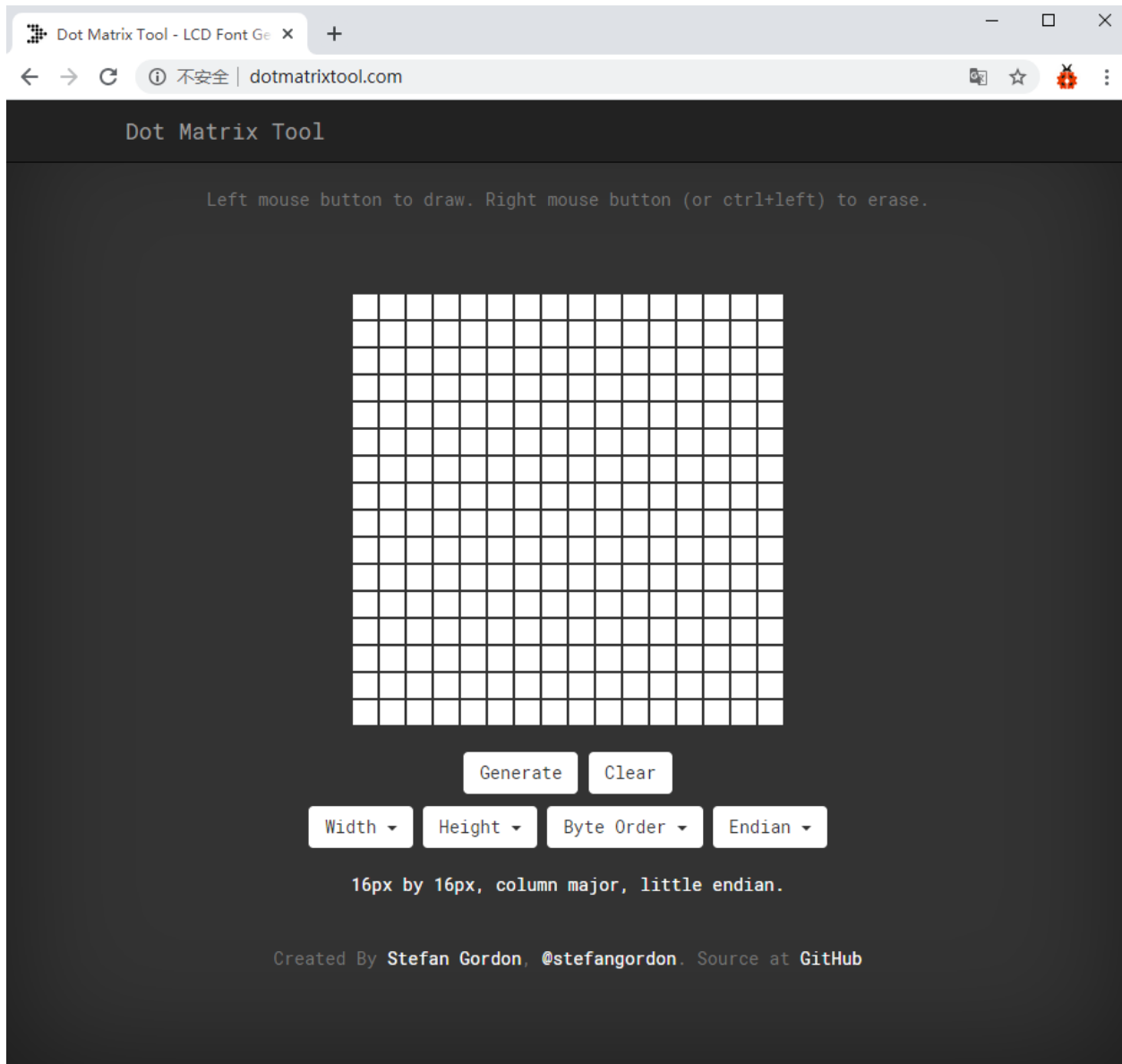
#### Specification:

- Input voltage: 5V
- Rated input frequency: 400KHZ
- Input power: 2.5W
- Input current: 500mA

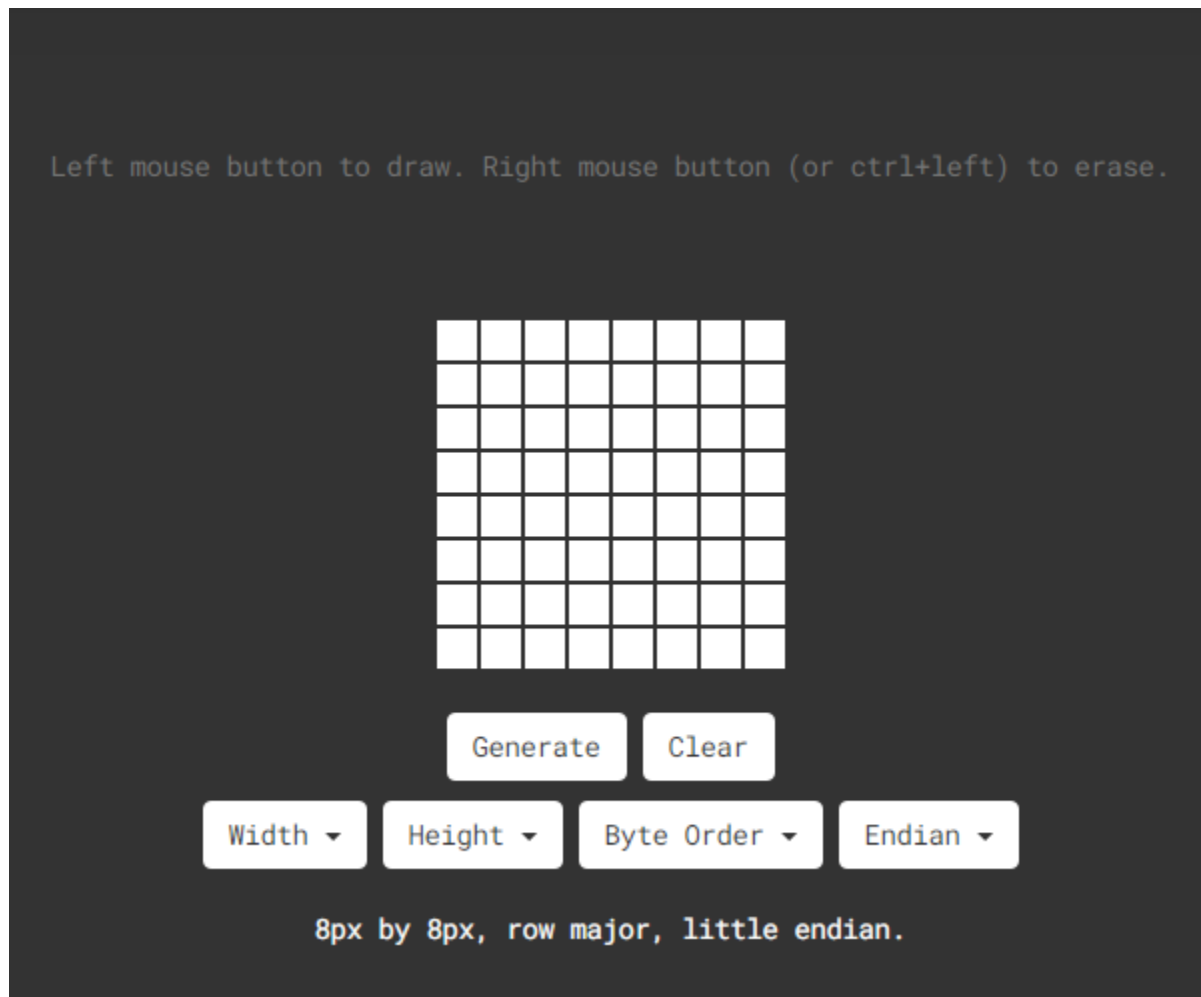
Introduction for Modulus Tool

The online version of dot matrix modulus tool: <http://dotmatrixtool.com/#>

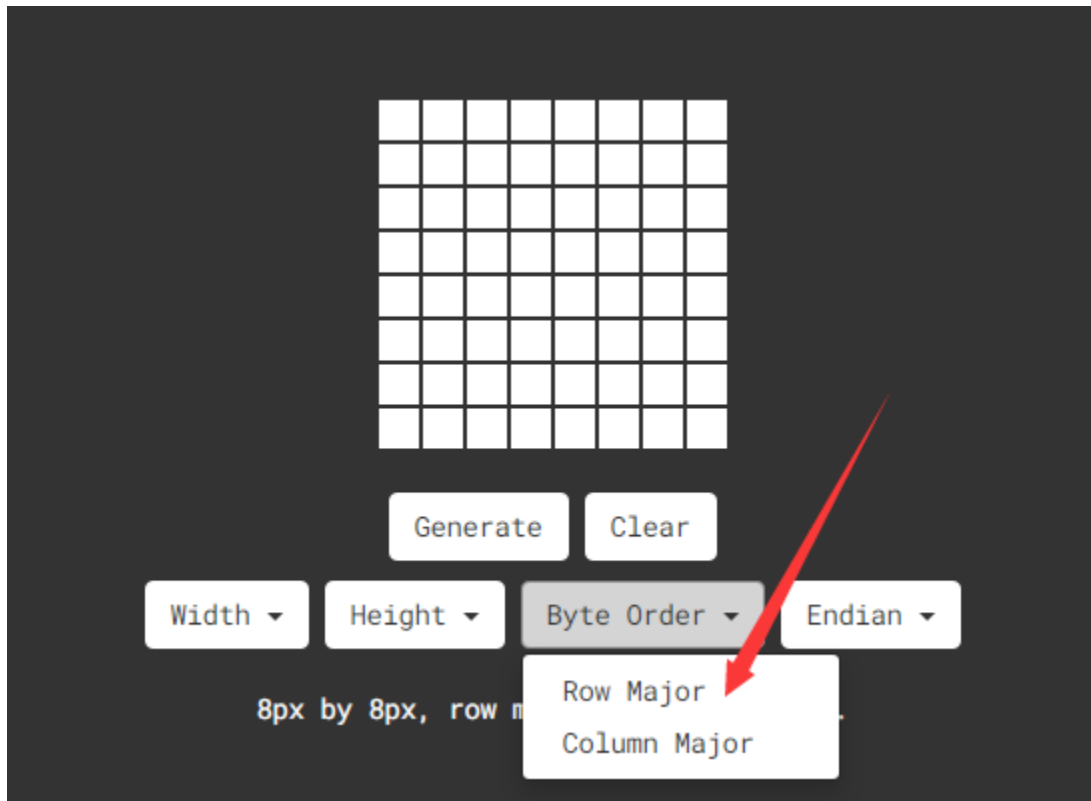
Open the link to enter the following page.



The dot matrix is 8\*8 in this project. So set the height to 8, width to 8; as shown below.

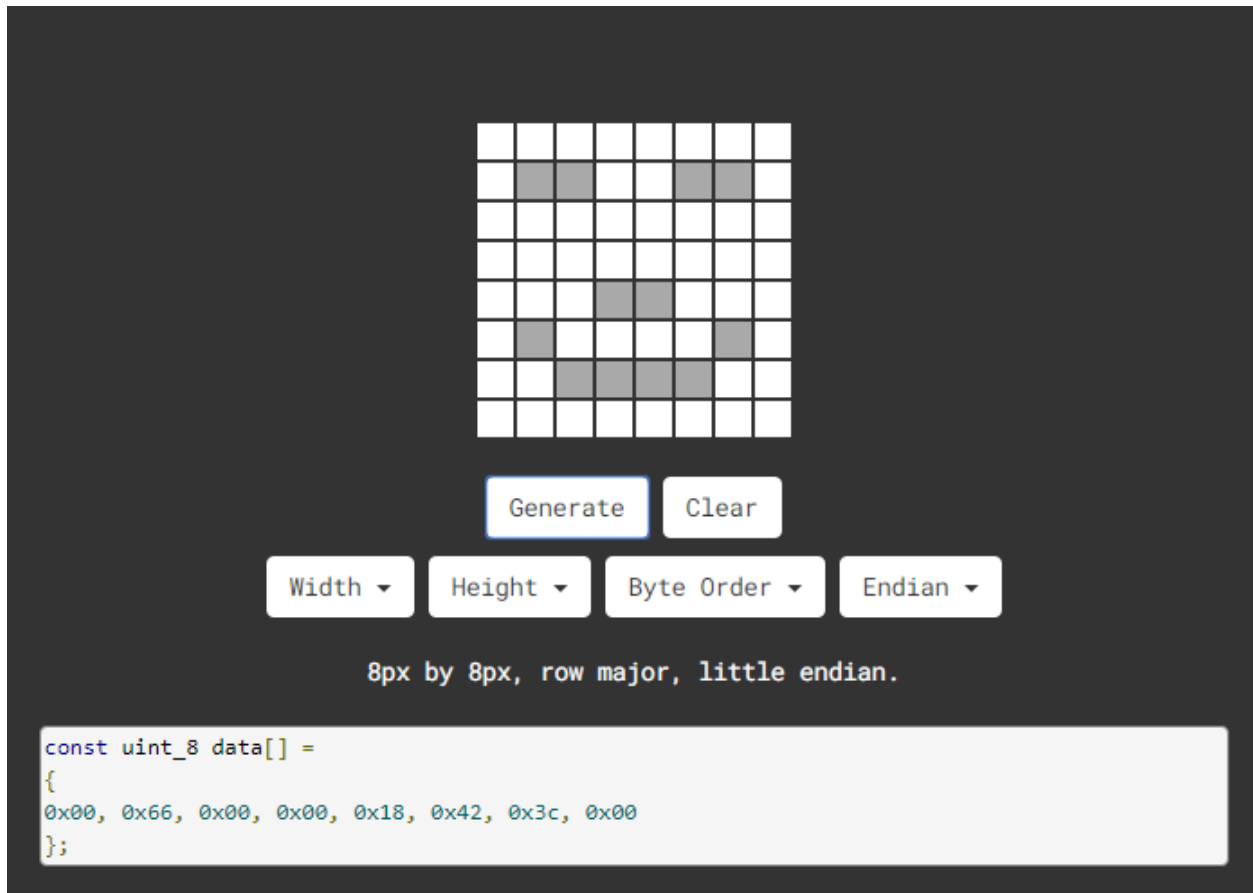


Click **Byte order** to select **Row major**.



Generate hexadecimal data from the pattern.

As shown below, the left button of the mouse is for selection while the right is for canceling. Thus you could use them to draw the pattern you want, then click **Generate** to yield the hexadecimal data needed.



The generated hexadecimal code (0x00, 0x66, 0x00, 0x00, 0x18, 0x42, 0x3c, 0x00) is what will be displayed, so you need to save it for next procedure.

#### (4)Wiring up

| 8*8 Dot matrix display | PCB Board |
|------------------------|-----------|
| G                      | G         |
| 5V                     | 5V        |
| SDA                    | SDA       |
| SCL                    | SCL       |

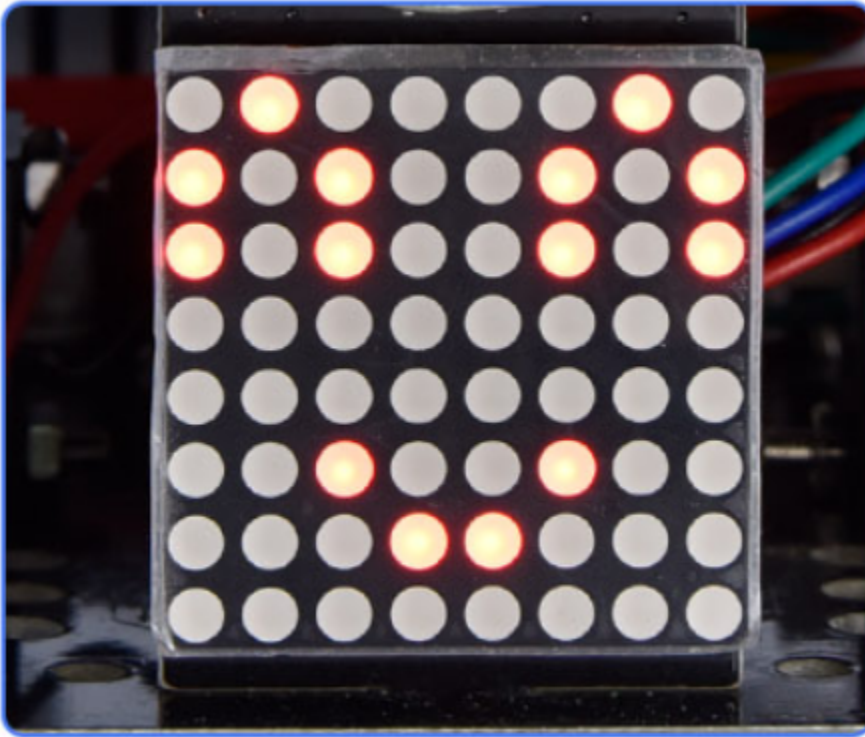
### (5)Test Code

The 8\*8 dot matrix is controlled by A4SDAand A5SCLof the Arduino Nano board.



### (6)Test Result

Upload the test code to the Arduino Nano board and power up by a USB cable, the 8\*8 dot matrix display will show patterns.





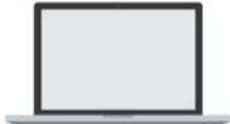

### 9.3.5 Project 5: Servo Rotation

#### (1)Description

There are two servos on the car. We take the servo connected to pin D9 as an example.

The servo is a motor that can rotate very accurately. It has been widely applied to toy cars, remote control helicopters, airplanes, robots and other fields. In this project, we will use the Nano motherboard to control the servo to spin.

#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |



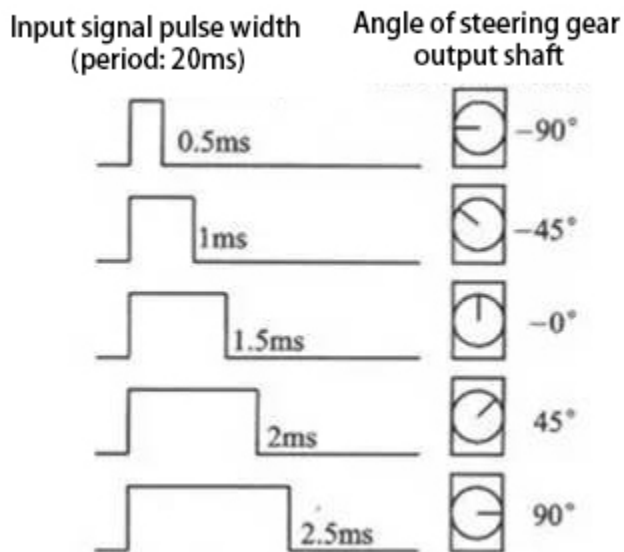
### (3)Knowledge



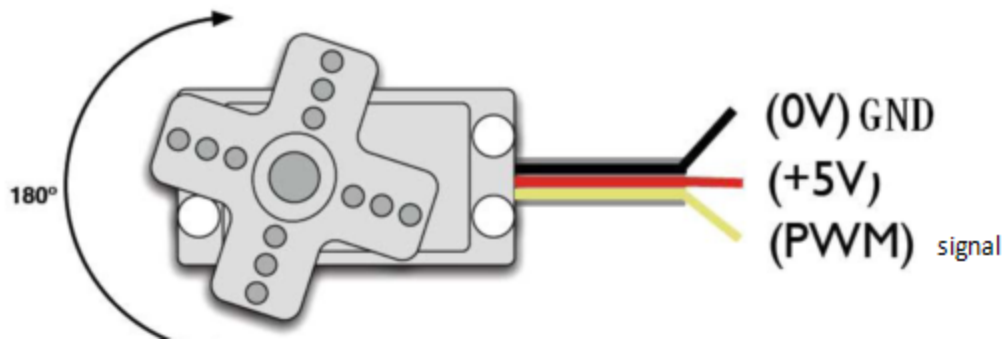
Servo motor is a position control rotary actuator. It mainly consists of a housing, a circuit board, a core-less motor, a gear and a position sensor. Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of 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  $0^{\circ}$ – $180^{\circ}$ .

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  $0^{\circ}$  to  $180^{\circ}$ . 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.

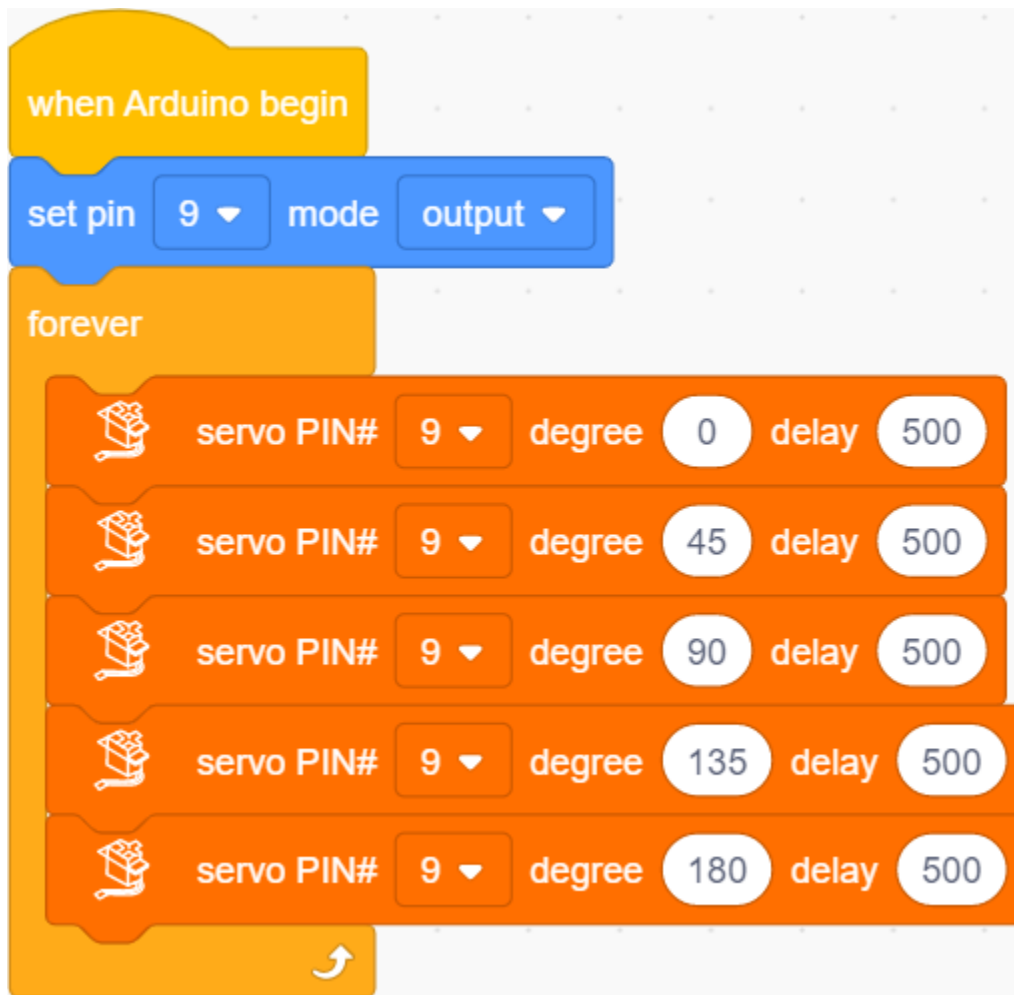


#### (4)Wire up

| Servo  | PCB Board |
|--------|-----------|
| Brown  | G         |
| Red    | 5V        |
| Orange | S1D9      |

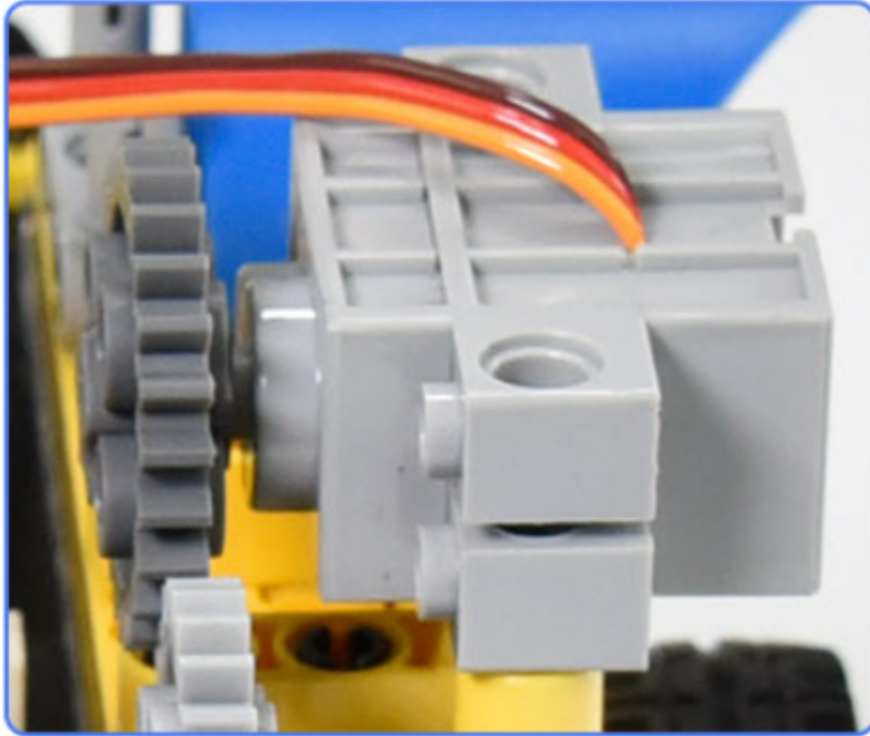
#### (5)Test Code

The servo for controlling the ultrasonic sensor is controlled by the D9 of the Arduino Nano board.



#### (6)Test Result

Upload the test code to the Arduino Nano board, and power up with a USB cable. Then the arm of the servo will rotate to 0°, 45°, 90°, 135° and 180°.







### 9.3.6 Project 6: Motor Driving and Speed Control

#### (1)Description

There are many ways to drive motors. This car uses the most commonly used DRV8833 motor driver chip, which provides a dual-channel bridge electric driver for toys, printers and other motor integration applications.

In this experiment, we use the DRV8833 motor driver chip on the expansion board to drive the two DC motors, and demonstrate the effect of forward, backward, left-turning, and right-turning.

#### (2)Components Required

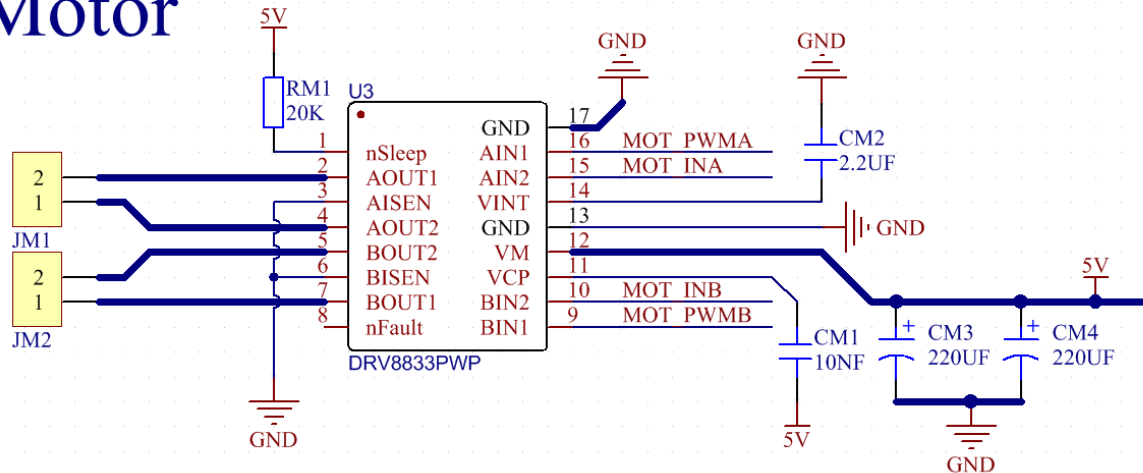
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

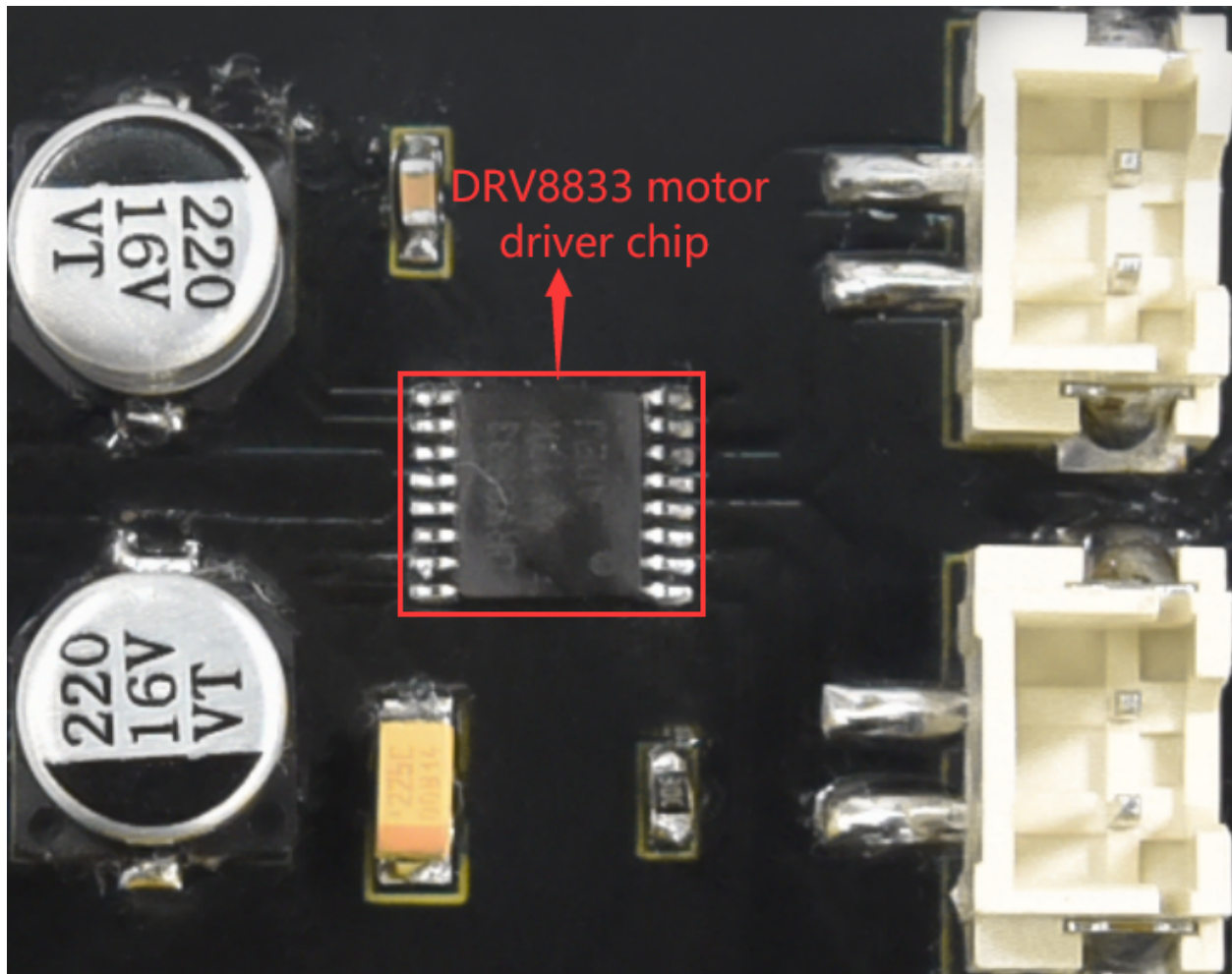
### (3)Knowledge

DRV8833 motor driver chip: Dual H-bridge motor driver with current control function, can drive two DC motors, one bipolar stepper motor, solenoid valve or other inductive loads. Each H-bridge includes circuitry to regulate or limit winding current.

An internal shutdown function with a fault output pin is used for over-current and short circuit protection, under-voltage lockout and over-temperature. A low-power sleep mode is also added. Let's take a look at the schematic diagram of the DRV8833 motor driver chip driving two DC motors:

## Motor





If you want to get insight to it, you can check the specification of this chip. Just browse it in the “Attachments” folder.



#### (4) Specification

Input voltage of logic part: DC 5V

Input voltage of driving part : DC 5V

Working current of logic part: <30mA

Operating current of driving part: <2A

Maximum power dissipation: 10W (T=80°C)

Motor speed: 5V 200 rpm / min

Motor drive form: dual H-bridge drive

Control signal input level: high level  $2.3V < V_{in} < 5V$ , low level  $-0.3V < V_{in} < 1.5V$

Working temperature: -25~130°C

**(5) Drive the car to move**

From the above diagram, the direction pin of the left motor is D4; the speed pin is D6; D2 is the direction pin of the right motor; and D5 is speed pin.

PWM drives the robot car. The PWM value is in the range of 0-255. The more the PWM value is set, the faster the rotation of the motor.

| Function   | D4   | D6PWM | Left motor    | D2   | D5PWM | Right motor   |
|------------|------|-------|---------------|------|-------|---------------|
| forward    | LOW  | 200   | clockwise     | LOW  | 200   | clockwise     |
| Go back    | HIGH | 50    | anticlockwise | HIGH | 50    | anticlockwise |
| Turn left  | HIGH | 200   | anticlockwise | LOW  | 200   | clockwise     |
| Turn right | LOW  | 200   | clockwise     | HIGH | 200   | anticlockwise |
| Stop       | LOW  | 0     | stop          | LOW  | 0     | stop          |

## (6)Test Code





### (7)Test Result

Upload the test code to the Arduino Nano board, install batteries, turn the power switch to ON end and power up. The car moves forward for 2s, back for 2s, turn left for 2s, right for 2s and stops for 2s.

## 9.3.7 Project 7: Ultrasonic Sensor

There is an ultrasonic sensor on the car. It is a very affordable distance-measuring sensor.

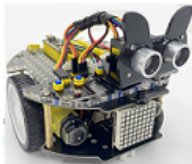



The ultrasonic sensor sends a high-frequency ultrasonic signal that human hearing can't hear. When encountering obstacles, these signals will be reflected back immediately. After receiving the returned information, the distance between the sensor and the obstacle will be calculated by judging the time difference between the transmitted signal and the received signal. It is mainly used for object avoidance and ranging in various robotics projects.

### Project 7.1: Ultrasonic Ranging

#### (1)Description

In this experiment, we use an ultrasonic sensor to measure distance and print the data on a serial monitor.

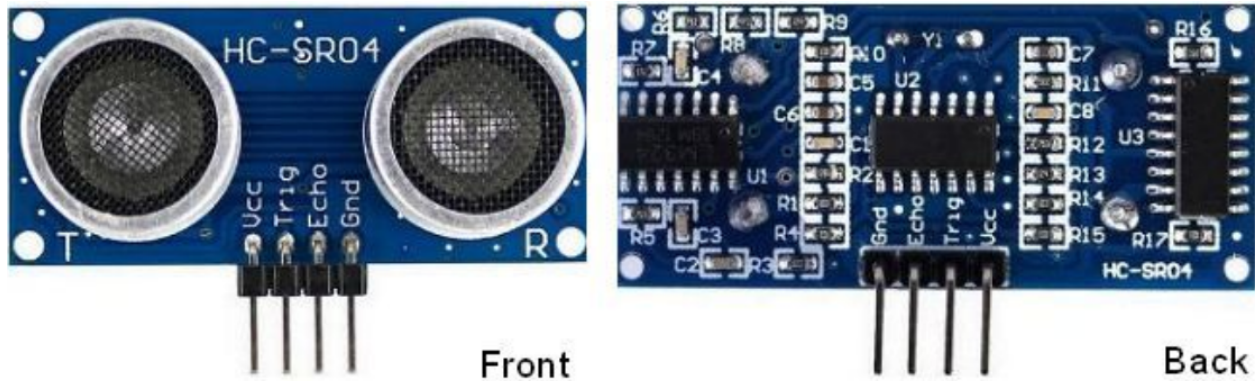
#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

#### (3)Knowledge

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.

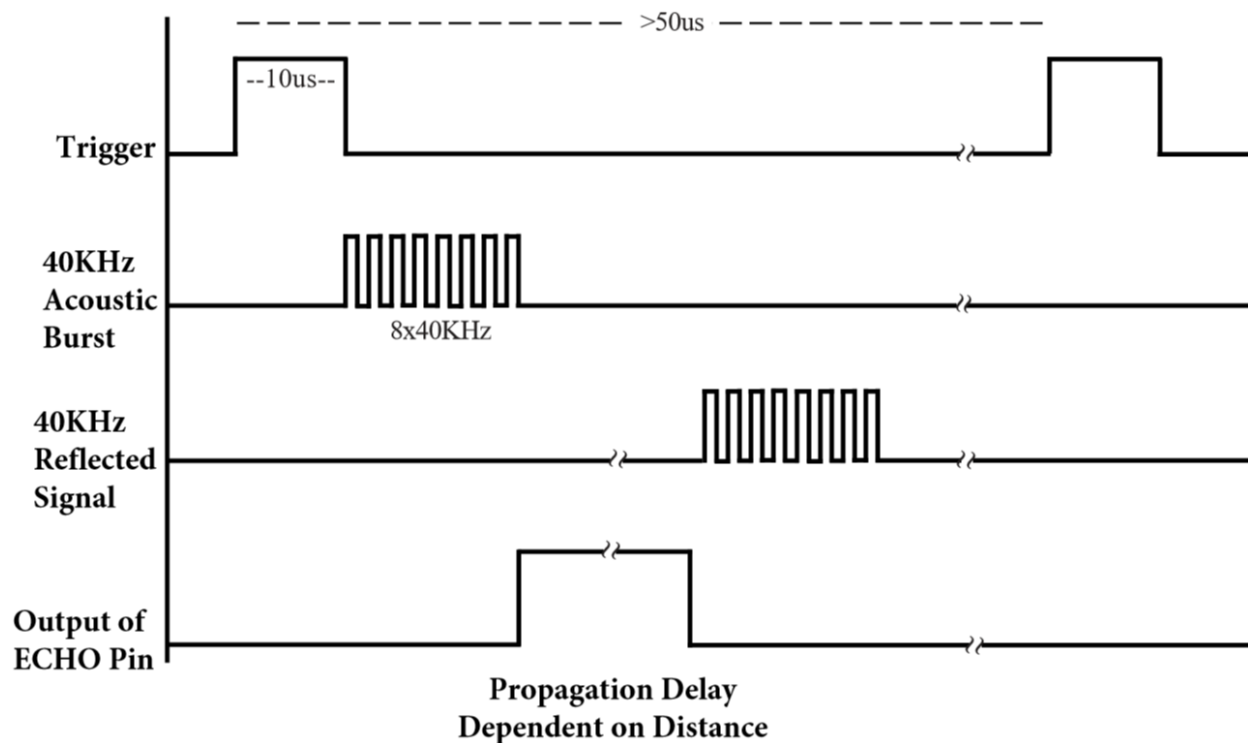


## HC-SR04

Use method and timing chart of ultrasonic module:

1. Setting the delay time of Trig pin of SR04 to 10s at least, which can trigger it to detect distance.
2. 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.
3. 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.

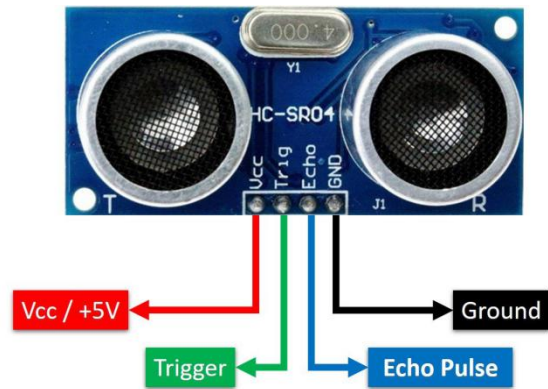
### HC-SR04 ULTRASONIC MODULE



Time=Echo pulse width, unit: us

Distancecm=time/ 58

Distance(inch)=time/ 148



The HC-SR04 ultrasonic sensor has four pins: Vcc, Trig, Echo and GND.

The Vcc pin provides power generating ultrasonic pulses and is connected to Vcc/+5V. The GND pin is grounded/GND.

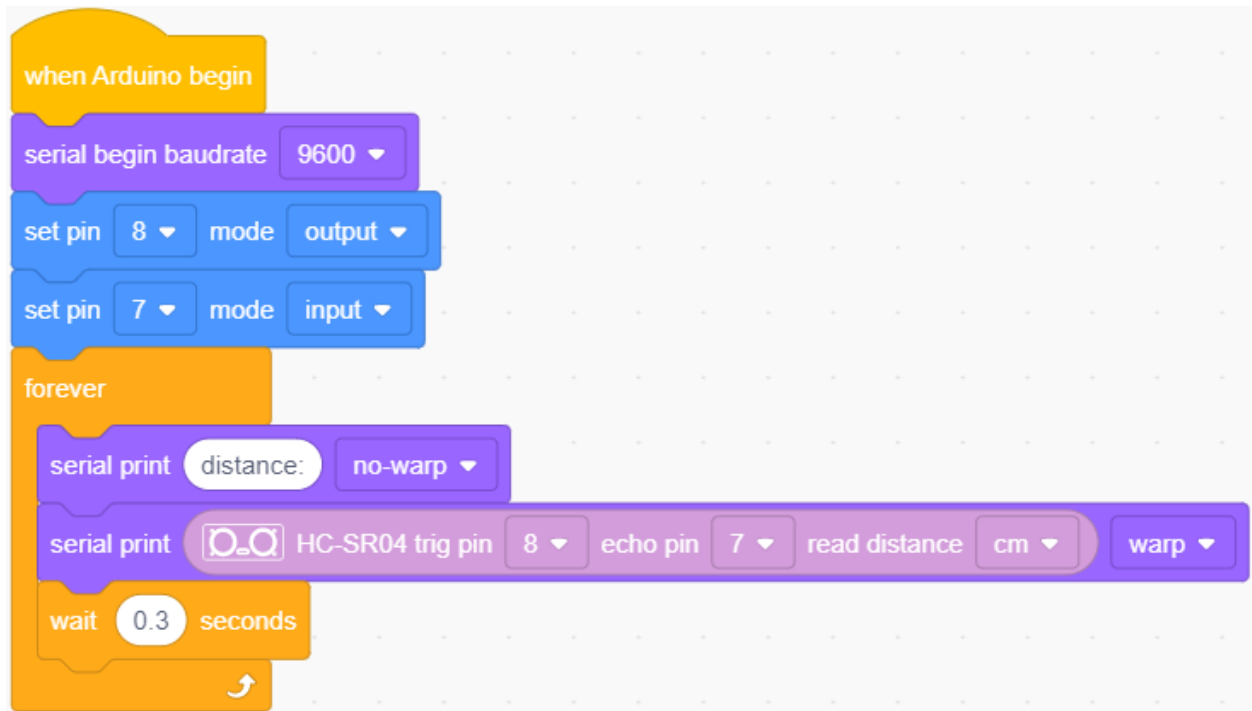
The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse. The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse stroke to the Arduino control board.

#### (4)Wiring Up


| Ultrasonic Sensor | PCB Board |
|-------------------|-----------|
| Vcc               | 5V        |
| Trig              | S2D8      |
| Echo              | S1D7      |
| Gnd               | G         |

#### (5)Test Code

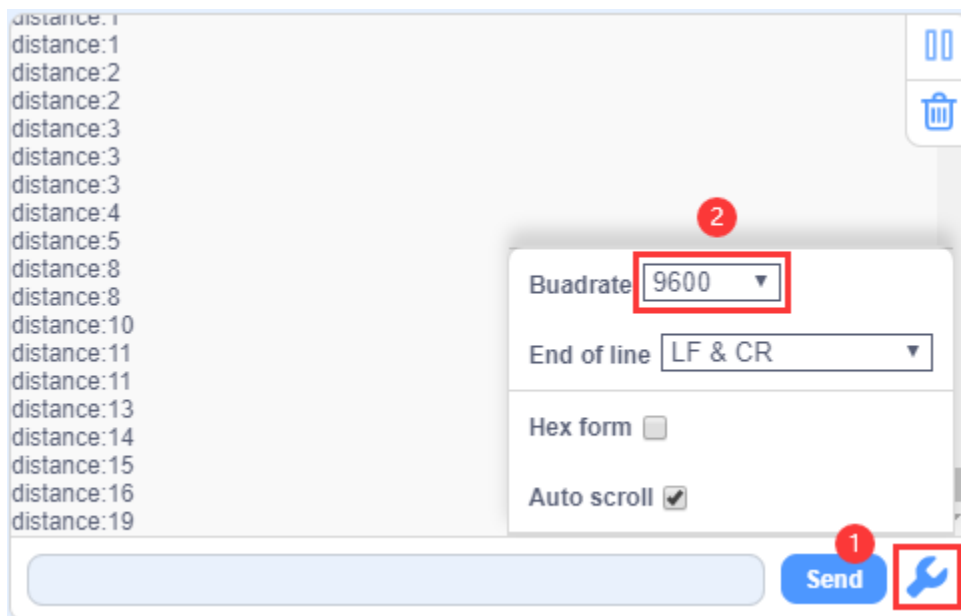
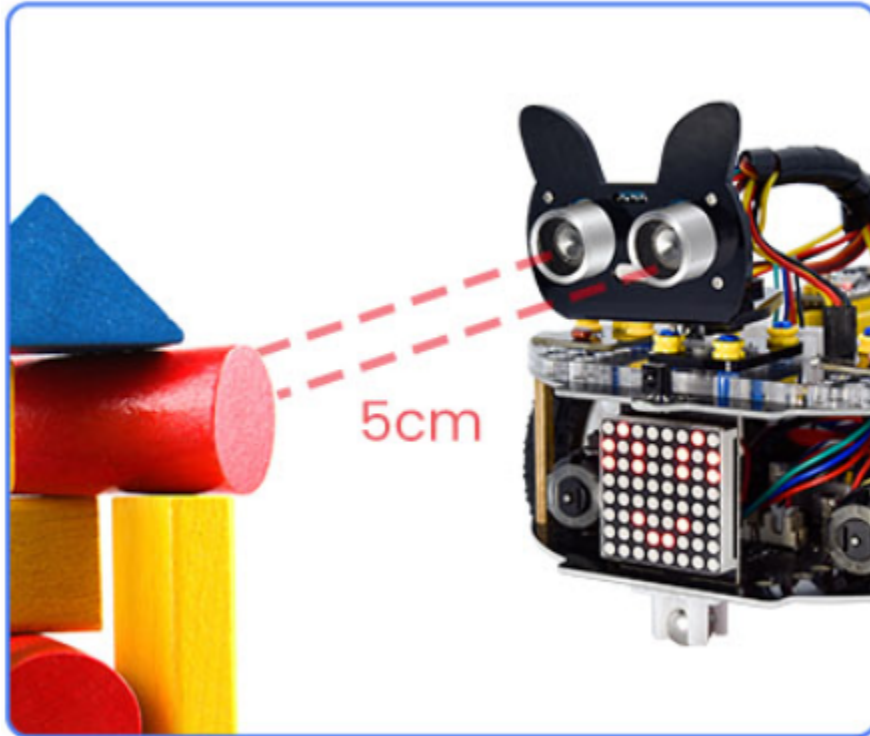
The pin Trig and Echo of the ultrasonic sensor are controlled by the D8 and D7 of the Arduino Nano.



### (6)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor to click  and set baud rate to 9600.

When you move an object in front of the ultrasonic sensor, it will detect the distance and the serial monitor will show the distance value.

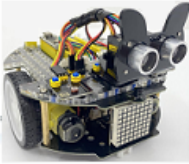





### Project 7.2: Ultrasonic Following

#### (1)Description

In the above experiments, we have learned about the 8\*8 dot matrix, motor drivers and speed regulation, ultrasonic sensors, servos and other hardware. In this experiment, we will combine them to create a follow car with the ultrasonic sensor. The can can follow an object to move through measuring distance.

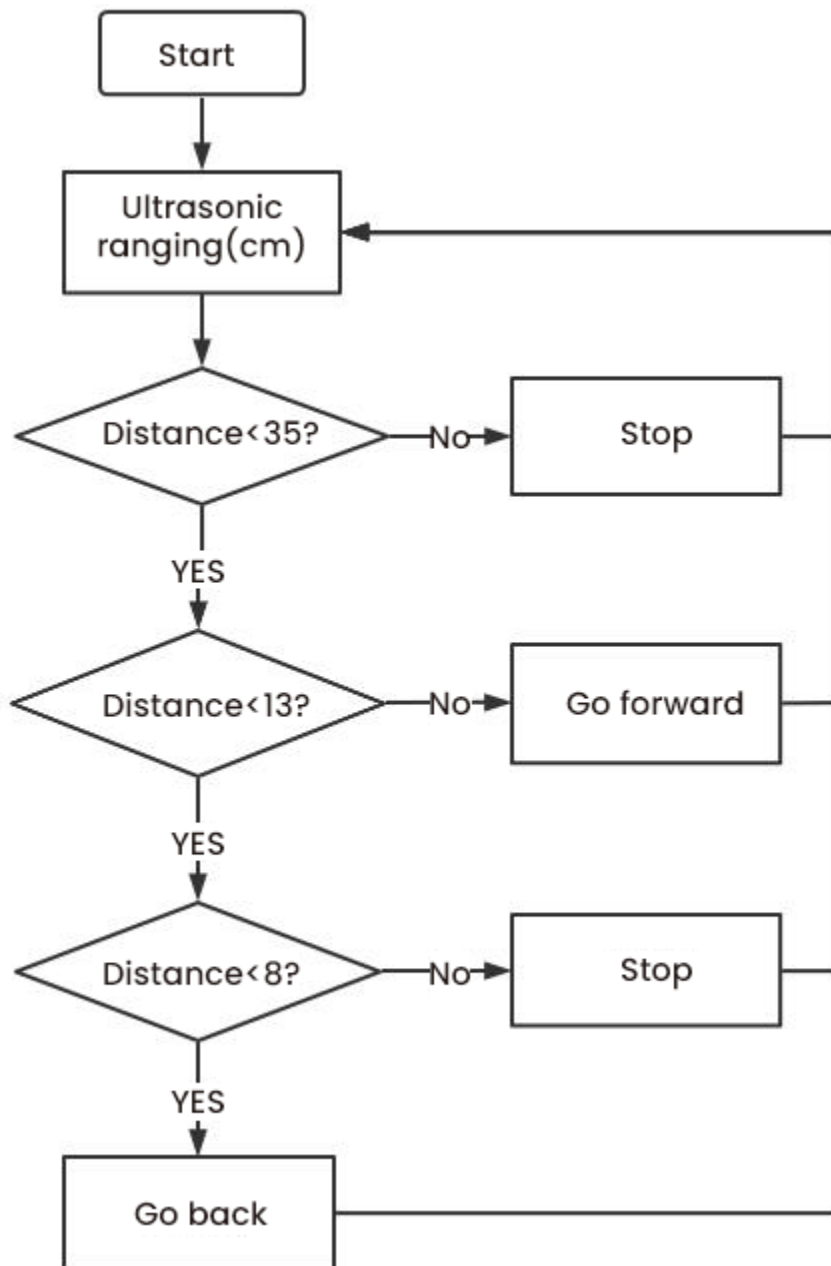
#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

#### (3)Working Principle

| Detection   | Detect the front distance Distanceunitcm |
|-------------|--|
| Condition 1 | Distance8                                |
| State       | Go backset PWM to 100                    |
| Condition 2 | 8distance<13                             |
| State       | stop                                     |
| Condition 3 | 13distance<35                            |
| State       | Go forwardset PWM to 100                 |
| Condition 4 | distance35                               |
| State       | stop                                     |

## (4)Flow Chart







**(5)Test Code**

```

when Arduino begin
  Declare Global variable Type long Name distance Assigned to 0

  set pin 4 mode output
  set pin 6 mode output
  set pin 2 mode output
  set pin 5 mode output
  set pin 8 mode output
  set pin 7 mode input
  set pin 9 mode output

  servo PIN# 9 degree 90 delay 300

  forever
    Set distance variable by trig pin 8 echo pin 7 read distance cm

    if variable distance < 8 then
      Motor INA# 4 State HIGH INB# 6 analogWrite 50
      Motor INA# 2 State HIGH INB# 5 analogWrite 50

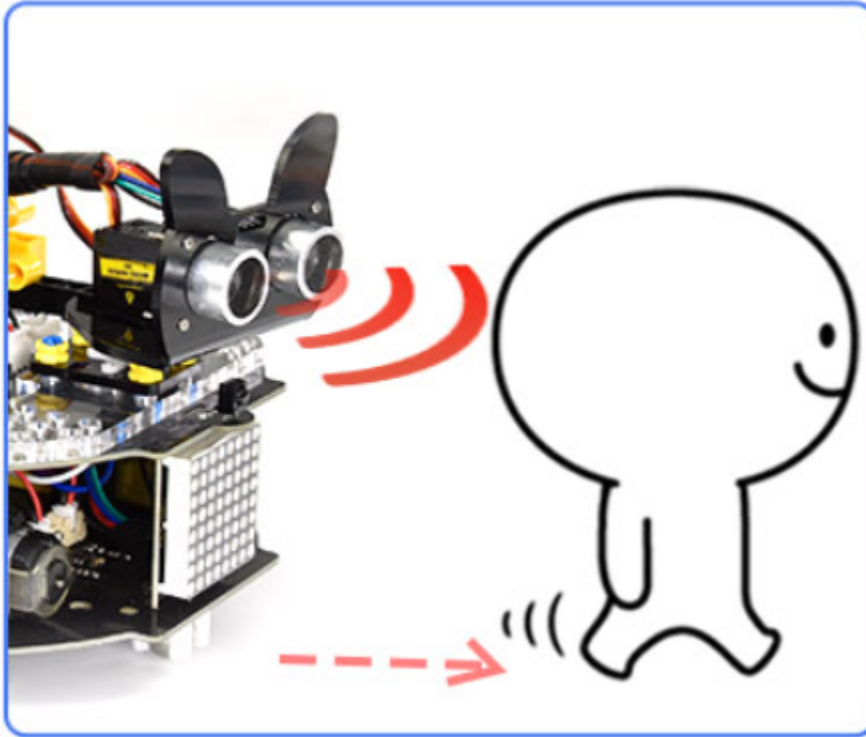
    if not variable distance < 8 and variable distance < 13 then
      Motor INA# 4 State LOW INB# 6 analogWrite 0
      Motor INA# 2 State LOW INB# 5 analogWrite 0

    if not variable distance < 13 and variable distance < 35 then
      Motor INA# 4 State LOW INB# 6 analogWrite 200
      Motor INA# 2 State LOW INB# 5 analogWrite 200

    if not variable distance < 35 then
      Motor INA# 4 State LOW INB# 6 analogWrite 0
      Motor INA# 2 State LOW INB# 5 analogWrite 0
  
```





**(6)Test Result**

Upload the code to the Arduino Nano board, install batteries and turn the switch to the ON end and power up. Then the car will follow the obstacle to move.

**Project 7.3: Dodge obstacles****(1)Description**

In this project, we will take advantage of the ultrasonic sensor to detect the distance away from the obstacle so as to avoid them.

**(2)Components Required**

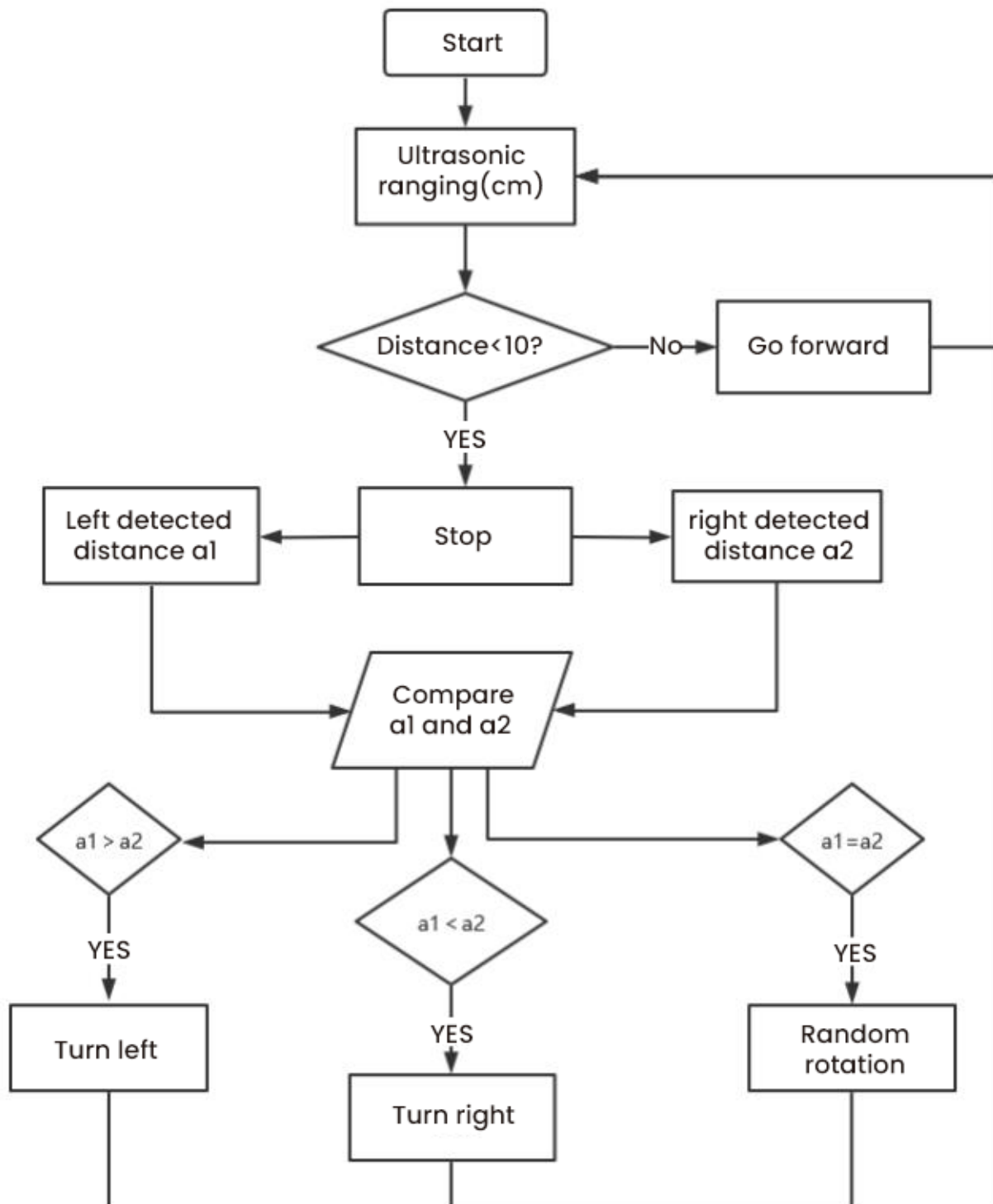
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

## (3)Working Principle

|      |   |                         |   |
|------|---|-------------------------|---|
|      | 8*8 Dot matrix display                                |                         |   |
|      | Set servo to 90°                                      |                         |   |
| loop | Detect the distance away from the obstacle (unit: cm) |                         |   |
|      | Condition 1   | State                   |   |
|      | 0<distance <<br>10                                    | Stop                    |   |
|      |   | Show the "stop" pattern |   |
|      |   | Set the servo to 180°   | Distance away form the obstacle:<br>a1 (unit: cm) |
|      |   | Set the servo to 0°     | Distance away form the obstacle:<br>a2 (unit: cm) |
|      |   | Condition 2             | State   |

|  |                           |  |                                  |
|--|---------------------------|--|----------------------------------|
|  |                           | $a1 < a2$  | Car turns right (set PWM to 200) |
|  |                           |  | show "turning right" pattern     |
|  |                           |  | Set servo to 90°                 |
|  |                           | $a1 \geq a2$   | Turn left (set PWM to 200)       |
|  |                           |  | display "left turning" pattern   |
|  |                           |  | Set servo to 90°                 |
|  | $\text{distance} \geq 10$ | The 8*8 dot matrix display shows "going forward" pattern |                                  |
|  |                           | Go forward (set PWM to 200)                              |                                  |

## (4)Flow Chart



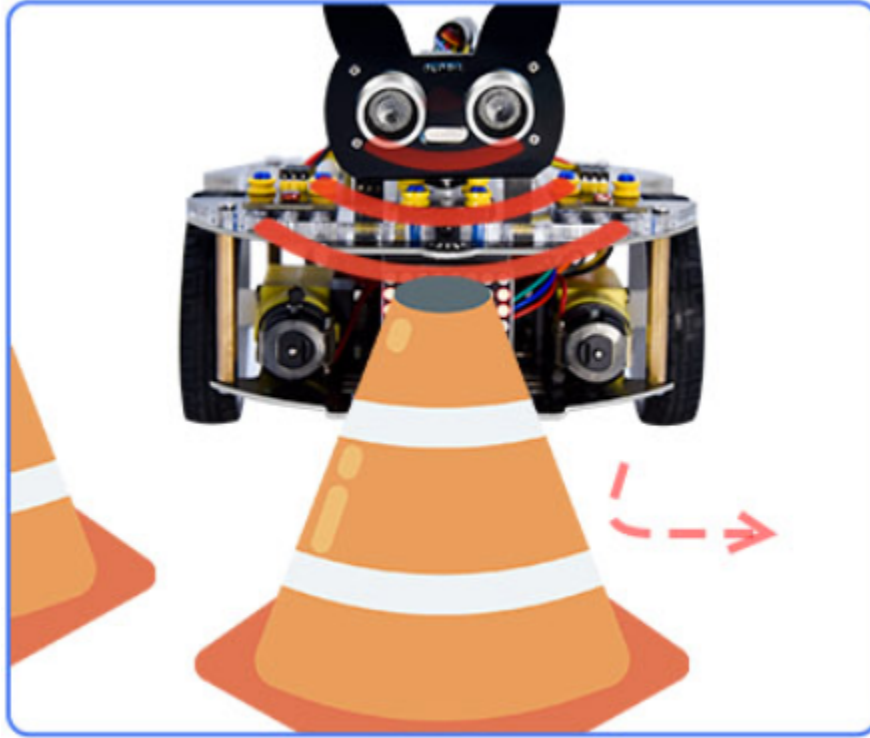


## (5) Test Code



### (6)Test Result

Upload the test code to the Arduino Nano board, put batteries in the battery holder, turn the power switch to the ON end and power up. Then the car can automatically dodge obstacles.



### 9.3.8 Project 8: Line Tracking Sensor

There are two IR line tracking sensors on the car. They are actually two pairs of ST188L3 infrared tubes and used to detect black and white lines. In this project, we will make a line tracking car.

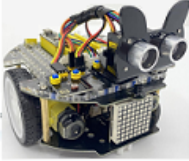

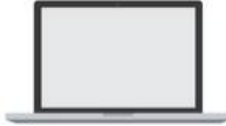

#### Project 8.1: Reading Values

##### (1)Description

In this experiment, we use ST188L3 infrared tubes to detect black and white lines, then print the data on the serial monitor.



**(2)Components Required**

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

**(3)Knowledge****Infrared line tracking:**

The IR line tracking sensor boasts a pair of ST188L3 infrared tubes. ST188L3 tubes has an infrared emitting diode and a receiver tube. When the emitting diode emits an infrared signal then received by the receiving tube after being reflected by the white object. Once the receiving tube receives the signal, the output terminal will output a low level (0); when the infrared emitting diode emits an infrared signal, and the infrared signal is absorbed by the black object, a high level (1) will be output, thus realizing the function of detecting signals through infrared rays.

Warning: Reflective optical sensors (including IR line tracking sensors) shouldn't be applied under sunlight as there is a lot of invisible light such as infrared and ultraviolet.

Values detected by the line tracking sensor are shown in the table.

The value will be 1 if detecting black or no objects and the value 0 will appear if detecting white objects.

The detected black object or no object represents 1, and the detected white object represents 0.


| Left | Right | ValueBinary |
|------|-------|-------------|
| 0    | 0     | 00          |
| 0    | 1     | 01          |
| 1    | 0     | 10          |
| 1    | 1     | 11          |

**(4)Test Code**

The line tracking sensors of the PCB board are controlled by D11 and D10 of the Arduino Nano board.

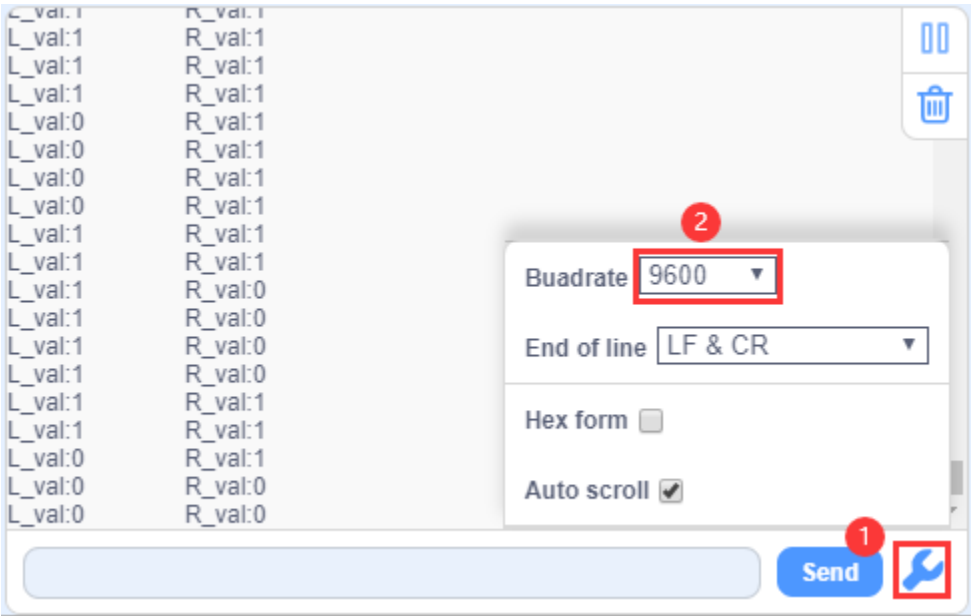


### (5)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor to click  and set baud rate to 9600.

Put a black thing under the line tracking sensor of the car and move it, you will see different indicators light up, and at the same time you will see the value on the serial monitor.

The sensitivity can be adjusted by rotating the potentiometer. When the indicator light is adjusted to the critical point of on and off state, the sensitivity is the highest.







Project 8.2: Line Tracking

(1)Description

We’ve introduced the knowledge of motor drivers, speed regulation, and infrared line tracking. In this experiment, the car will perform different actions according to the values transmitted by the infrared tracking.

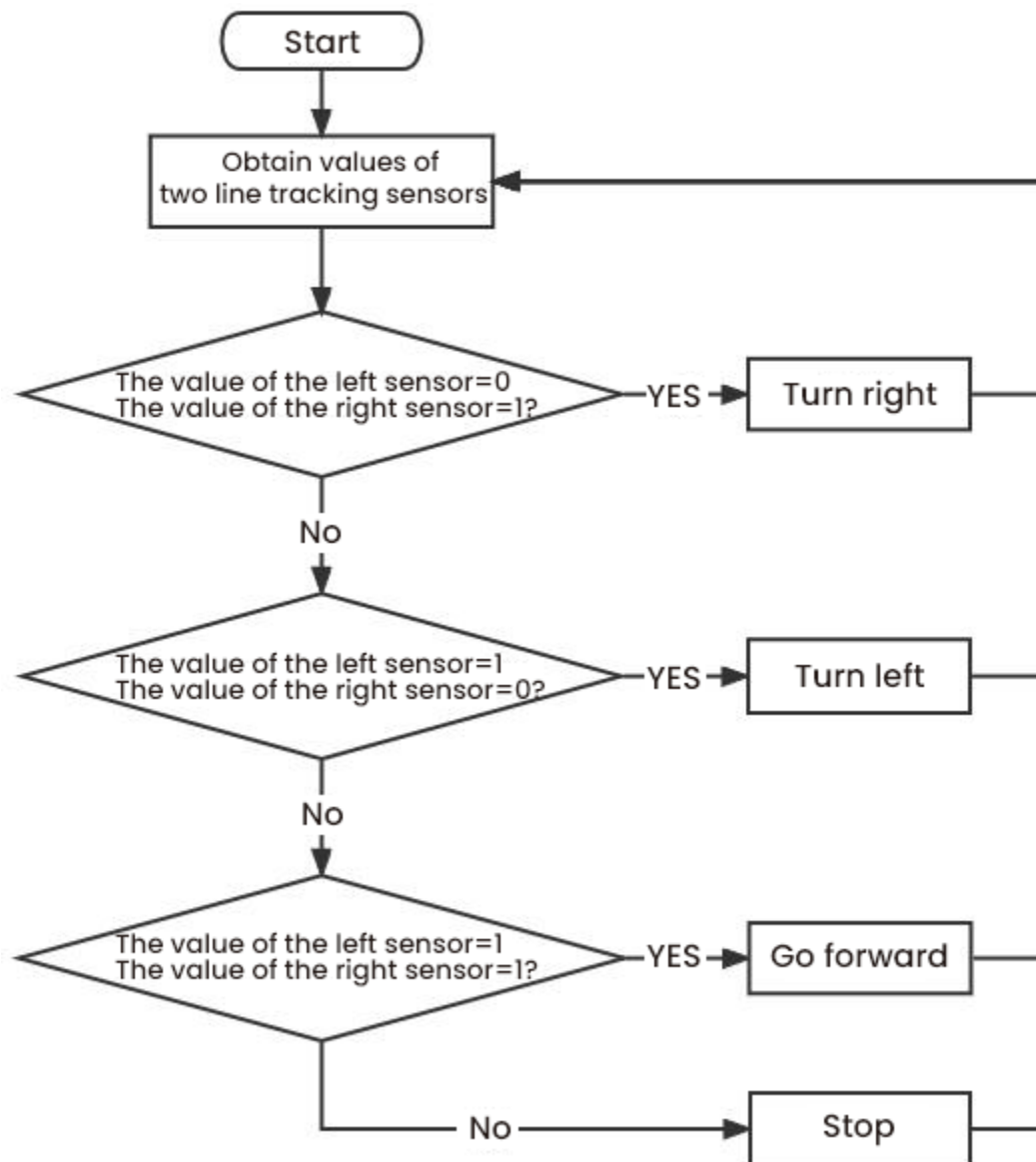
(2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

## (3)Working Principle

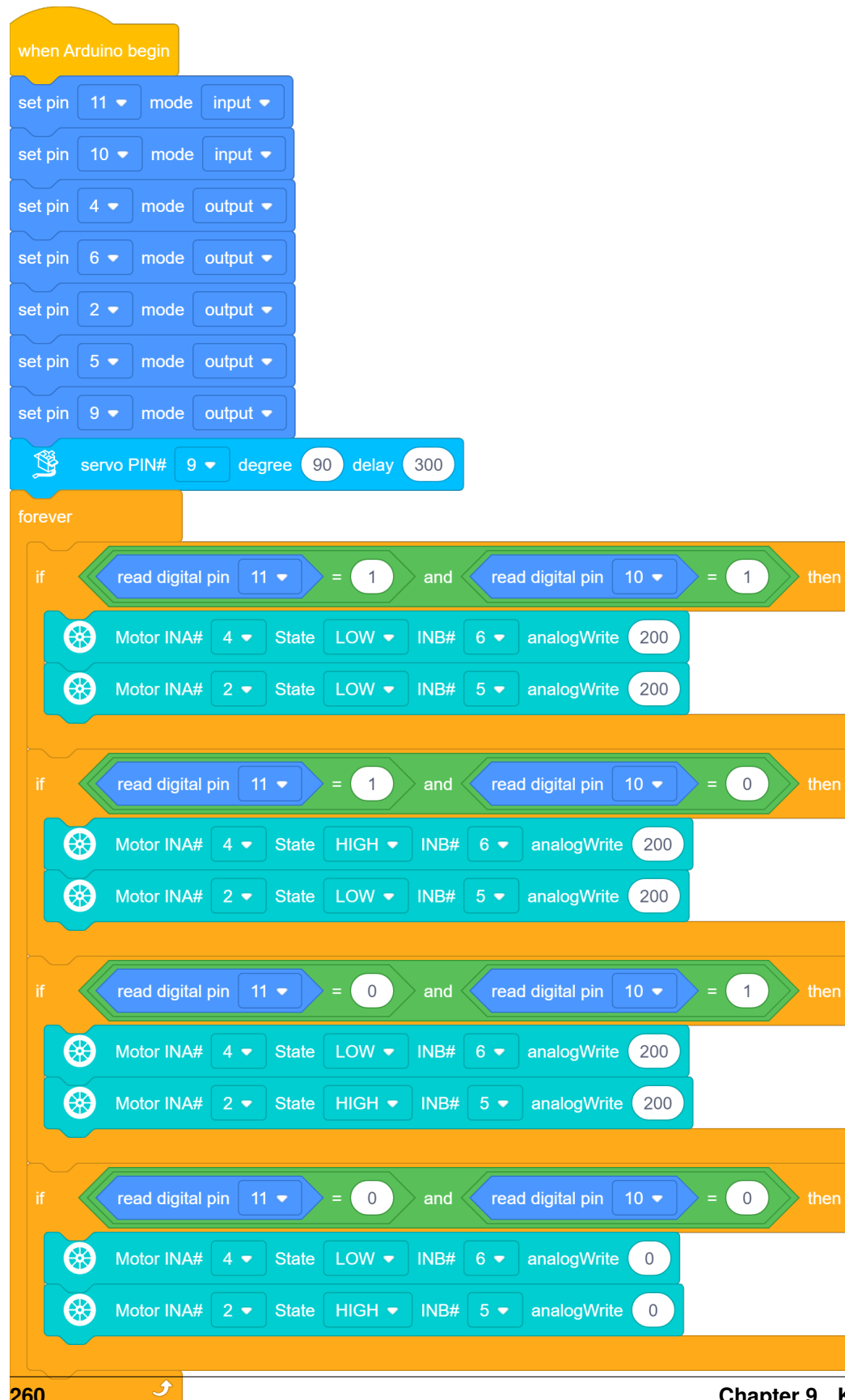
| Left | Right | ValueBinary | State        |
|------|-------|-------------|--------------|
| 0    | 0     | 00          | Stop         |
| 0    | 1     | 01          | Turn right   |
| 1    | 0     | 10          | Turn left    |
| 1    | 1     | 11          | Move forward |

## (4)Flow Chart





## (5) Test Code



The code is written in Scratch and controls the Beetlebot 3 in 1 Robot. It starts with a 'when Arduino begin' block, followed by seven 'set pin' blocks configuring pins 11, 10, 4, 6, 2, 5, and 9 as input or output. A 'servo PIN#' block sets pin 9 to 90 degrees with a 300ms delay. A 'forever' loop contains four 'if' statements based on the states of pins 11 and 10. Each 'if' block has two 'Motor INA#' blocks that set the state of motors 4 and 2 based on the input pins and write a value to pins 6 and 5.

```
when Arduino begin
  set pin 11 mode input
  set pin 10 mode input
  set pin 4 mode output
  set pin 6 mode output
  set pin 2 mode output
  set pin 5 mode output
  set pin 9 mode output
  servo PIN# 9 degree 90 delay 300
  forever loop
    if (read digital pin 11 = 1 and read digital pin 10 = 1) then
      Motor INA# 4 State LOW INB# 6 analogWrite 200
      Motor INA# 2 State LOW INB# 5 analogWrite 200
    if (read digital pin 11 = 1 and read digital pin 10 = 0) then
      Motor INA# 4 State HIGH INB# 6 analogWrite 200
      Motor INA# 2 State LOW INB# 5 analogWrite 200
    if (read digital pin 11 = 0 and read digital pin 10 = 1) then
      Motor INA# 4 State LOW INB# 6 analogWrite 200
      Motor INA# 2 State HIGH INB# 5 analogWrite 200
    if (read digital pin 11 = 0 and read digital pin 10 = 0) then
      Motor INA# 4 State LOW INB# 6 analogWrite 0
      Motor INA# 2 State HIGH INB# 5 analogWrite 0
```

## (6)Test Result

Upload the test code to the Arduino Nano board, turn the power switch to the ON end, power up and put the car on a map we provide. Then it will perform different functions via values sent by line tracking sensors.

## 9.3.9 Project 9: Light Following

There are two photoresistors on the car. They can vary with the light intensity and send information to the Nano board to control the car.

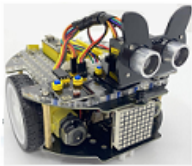



Photoresistors can determine and conduct the car to move by detecting light.

### Project 9.1: Read Values

#### (1)Description

In this experiment, we will learn the working principle of the photoresistor.

#### (2)Components Required

|   |   |  |   |
|---|---|--|---|
| Robot without Wifi<br>module*1  | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

#### (3)Knowledge

##### Photoresistor:

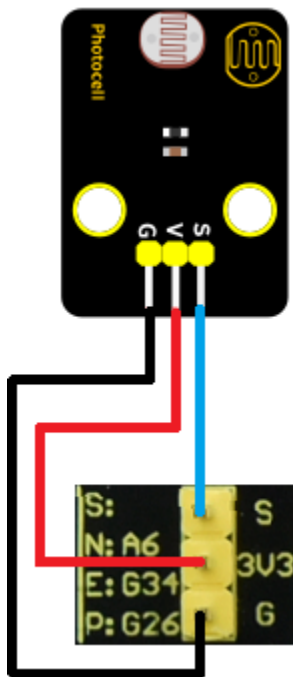
It mainly uses a photosensitive resistance element whose resistance varies from the light intensity. The signal terminal of the sensor is connected to the analog port of the microcontroller. When the light is stronger, the analog value at the analog port will increase; on the contrary, when the light intensity is weaker, the analog value of the microcontroller will reduce. In this way, the corresponding analog value can reflect the ambient light intensity.

#### (4)Wire up

Through the wiring-up diagram, signal pins of two photoresistors are connected to A6 and A7 of the Nano board.

For the following experiment, we use the photoresistor connected to A6 to finish experiments. First, let's read analog values.

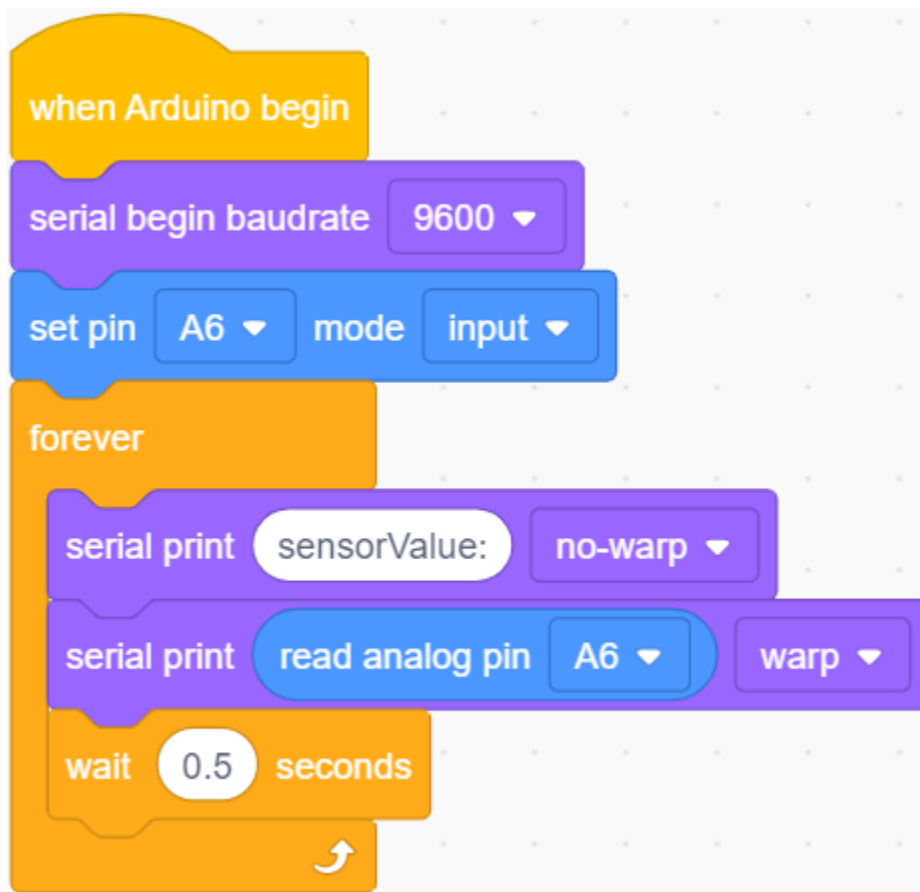
| Left photoresistor | PCB board |
|--------------------|-----------|
| G                  | G         |
| V                  | V         |
| S                  | SA6       |




#### (5)Test Code

The left photoresistor is controlled by the A6 of the Arduino Nano board.

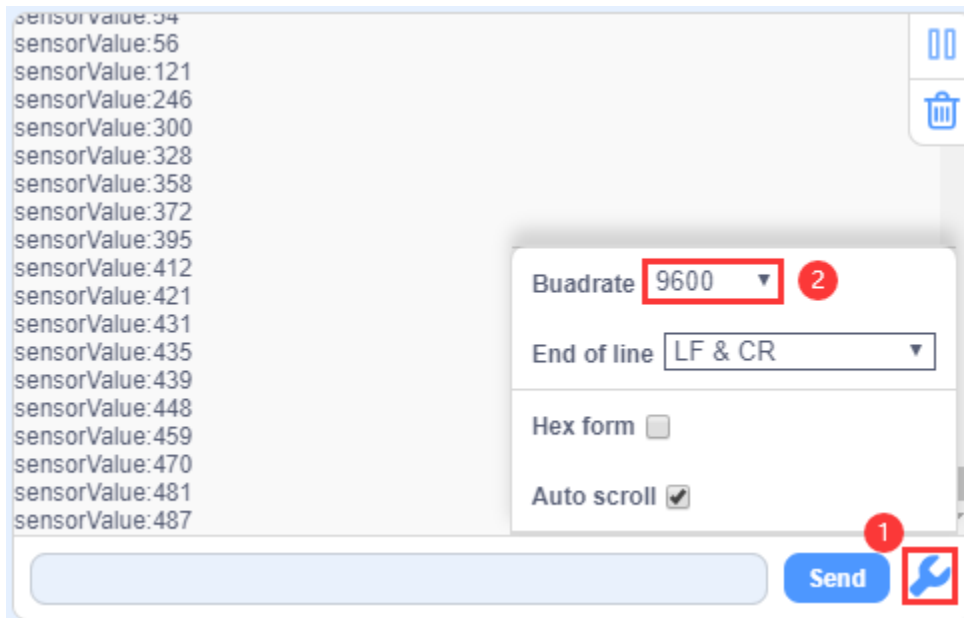




### (6)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor to click  and set baud to 9600.

When the light intensifies, the analog value will get increased; on the contrary, the analog value will get reduced.

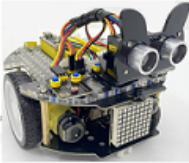





### Project 9.2: Light Following Car

#### (1)Description

We have learned the working principle of photoresistor, motor and speed regulation. In this experiment, we will use a photoresistor to detect the intensity of light as as to achieve the light following effect.

#### (2)Components Required

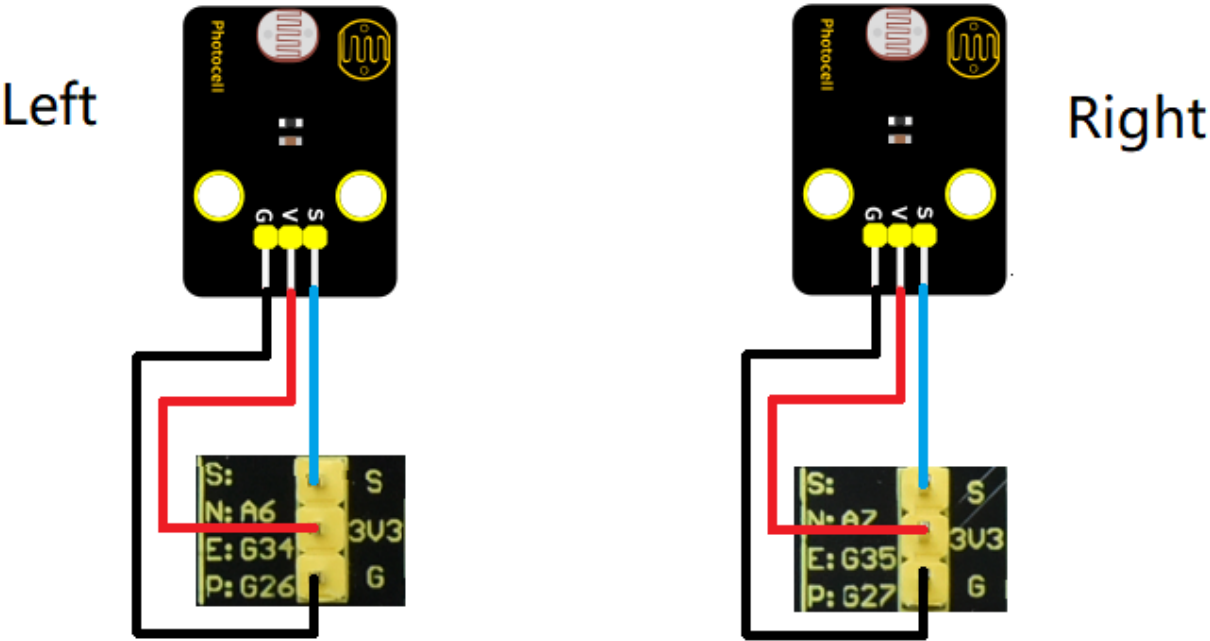
|   |   |  |   |
|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1   | 18650 Battery*1   |
|  |  |  |  |

(3)Working Principle

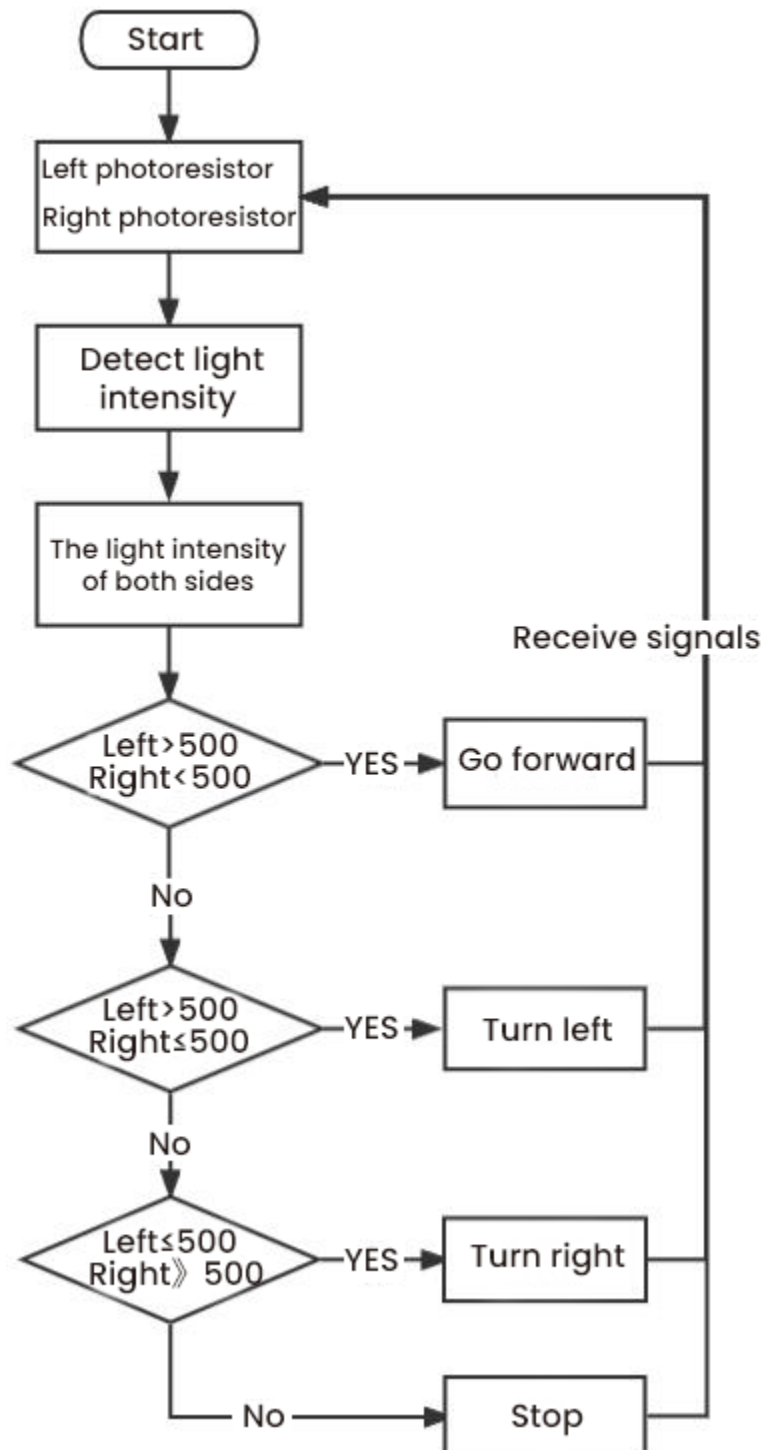
| Analog value of the left sensor | Analog value of the right sensor | Function      |
|---------------------------------|----------------------------------|---------------|
| >500                            | >500                             | Move forward  |
| >500                            | 500                              | Move to left  |
| 500                             | >500                             | Move to right |
| <500                            | <500                             | Stop          |

(4)Wiring up

| Left Photoresistor | PCB Board | Right photoresistor | PCB Board |
|--------------------|-----------|---------------------|-----------|
| G                  | G         | G                   | G         |
| V                  | V         | V                   | V         |
| S                  | SA6       | S                   | SA7       |

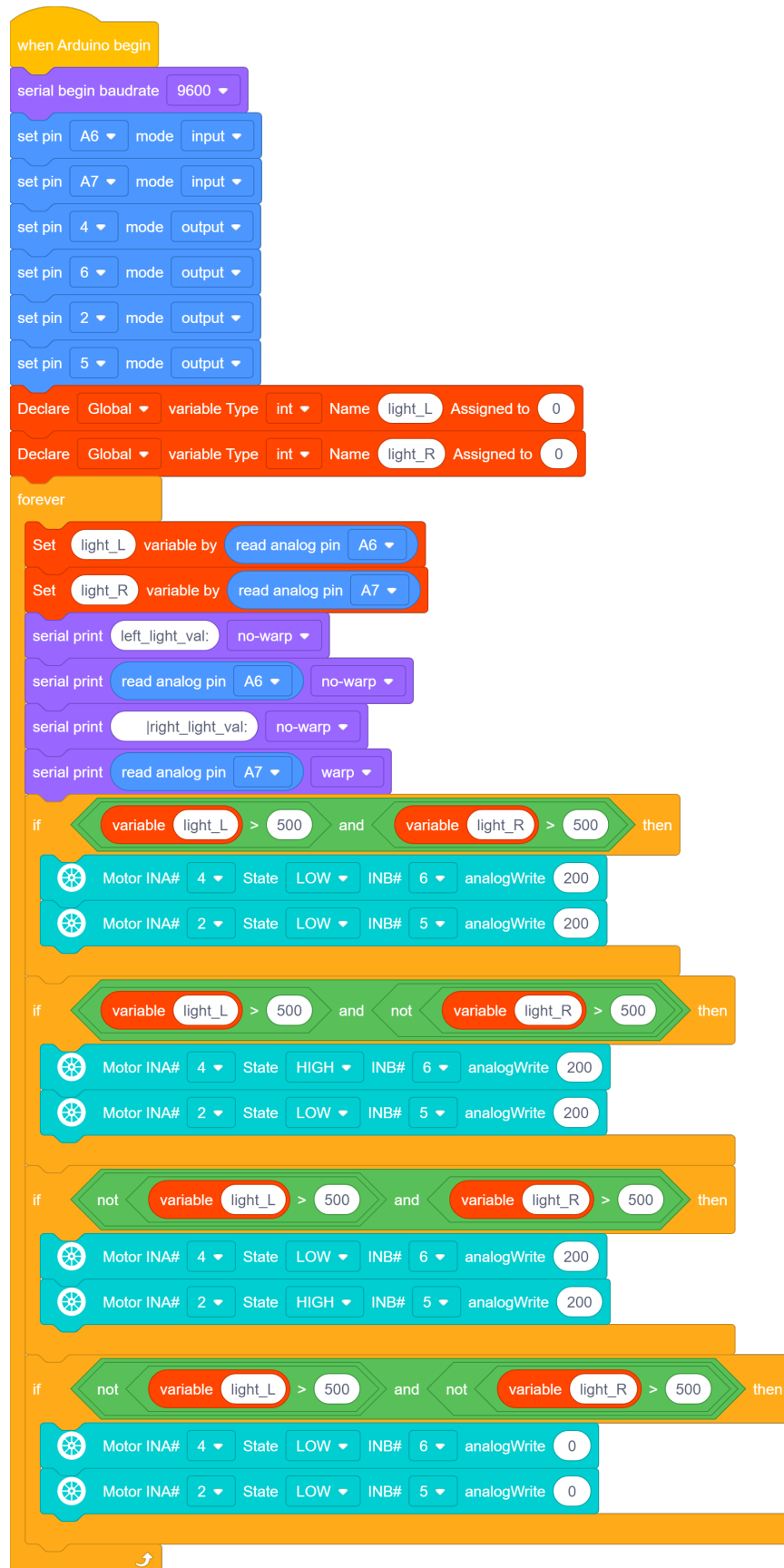


## (5)Flow Chart



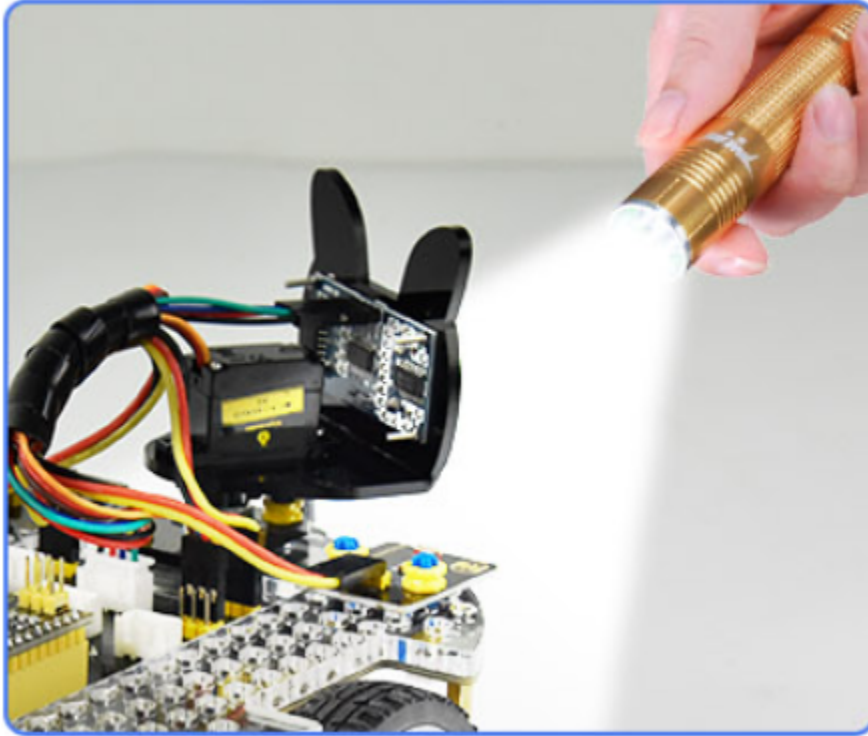
**(6)Test Code**

The left and right photoresistors are controlled by A6 and A7 of the Arduino Nano board.



### (7)Test Result

Upload the test code to the Arduino Nano board, put batteries in the battery holder, turn the power switch to the ON end and power up. Then the car will follow the light to move.



### 9.3.10 Project 10: IR Remote Control

Infrared remote controls are everywhere in daily life. It is used to control various home appliances, such as TV, speakers, video recorders and satellite signal receivers.

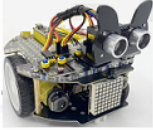

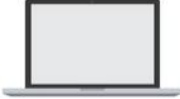


The remote control is composed of an IR emitter, an IR receiver and a decoding MCU. In this project, we will make a IR remote control car.

#### Project 10.1: IR Remote and Reception

##### (1)Description

In this experiment, we will combine the IR receiver and the IR remote control to read key values and show them on the serial monitor.

### (2)Components Required

|   |   |   |  |   |
|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | Remote Control*1   | 18650 Battery*1   |
|  |  |  |  |  |

### (3)Knowledge

#### IR Remote Control

It is a device with buttons. When the key is pressed, IR signals will be sent.

Infrared remote control technology is widely used, such as TVs, air conditioners and so on. And it can control air conditioners and TVs.

The infrared remote control adopts NEC coding, and the signal period is 110ms.

The remote control is shown below:

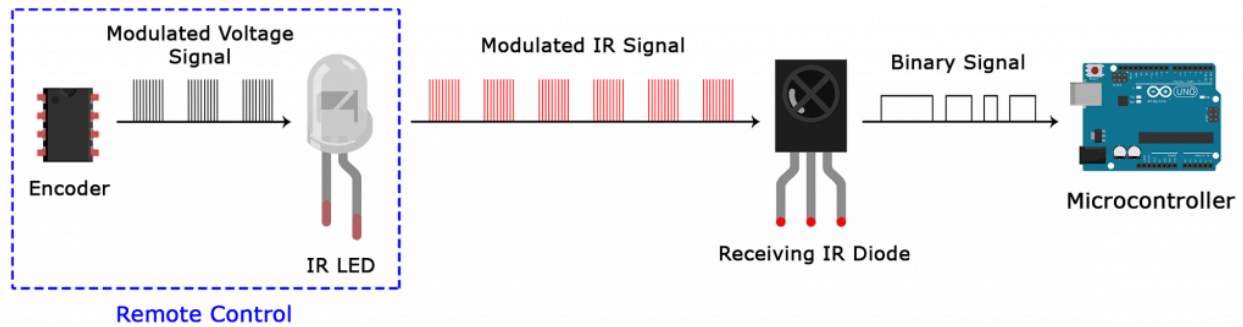




Infrared (IR) receiver:

It can receive infrared light and be used to detect the infrared signal emitted by the infrared remote control.

It can demodulate the received infrared light signal and convert it back to binary, and then transmit the information to the microcontroller.



### NEC Infrared communication protocol

#### NEC Protocol

To my knowledge the protocol I describe here was developed by NEC (Now Renesas). I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format.

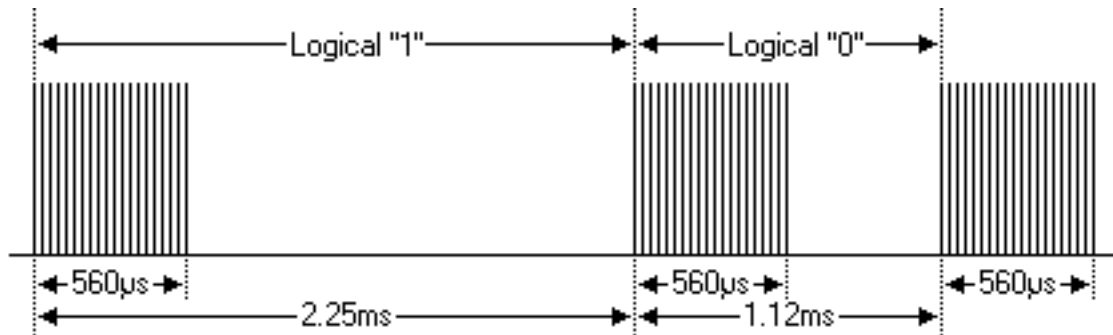
I do admit that I don't know exactly who developed it. What I do know is that it was used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC.

This description was taken from my VCR's service manual. Those were the days, when service manuals were filled with useful information!

#### Features

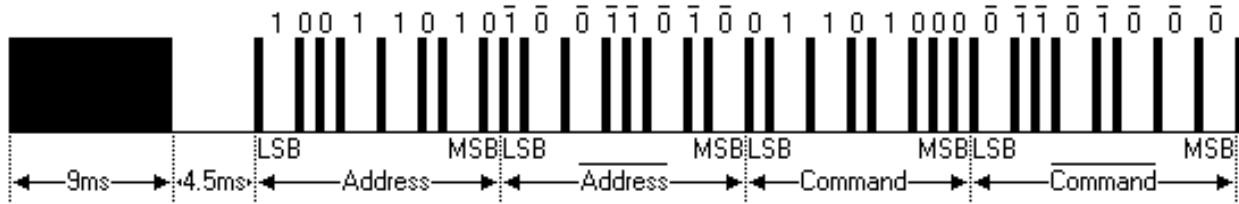
- 8 bit address and 8 bit command length.
- Extended mode available, doubling the address size.
- Address and command are transmitted twice for reliability.
- Pulse distance modulation.
- Carrier frequency of 38kHz.
- Bit time of 1.125ms or 2.25ms.

#### Modulation



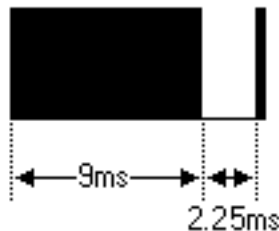
The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

#### Protocol

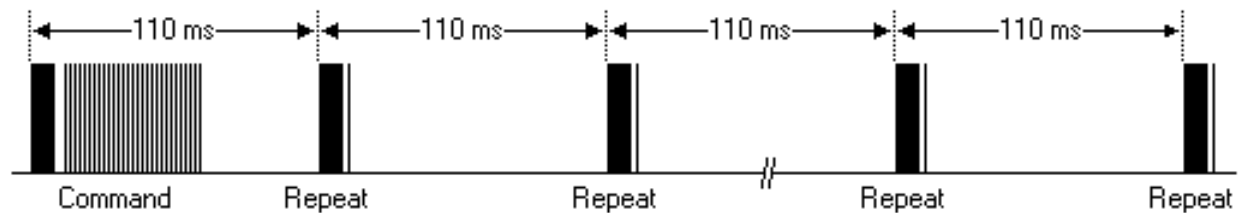


The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address 59 and Command 16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!

Keep in mind that one extra 560μs burst has to follow at the end of the message in order to be able to determine the value of the last bit.



A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560μs burst.

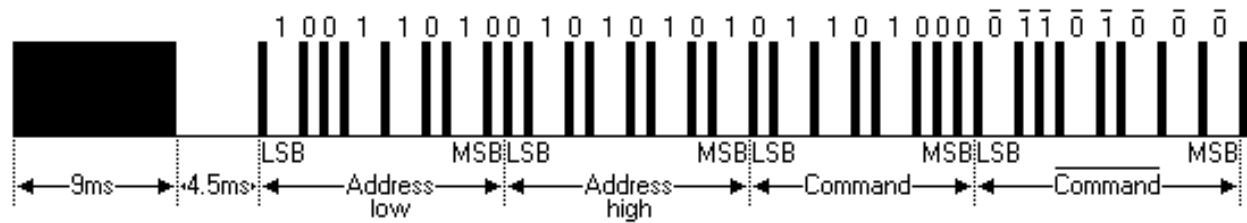


### Extended NEC protocol

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol.

By extending the address range this way the total message time is no longer constant. It now depends on the total number of 1's and 0's in the message. If you want to keep the total message time constant you'll have to make sure the number of 1's in the address field is 8 (it automatically means that the number of 0's is also 8). This will reduce the maximum number of different addresses to just about 13000.

The command redundancy is still preserved. Therefore each address can still handle 256 different commands.




Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

#### (4)Test Code

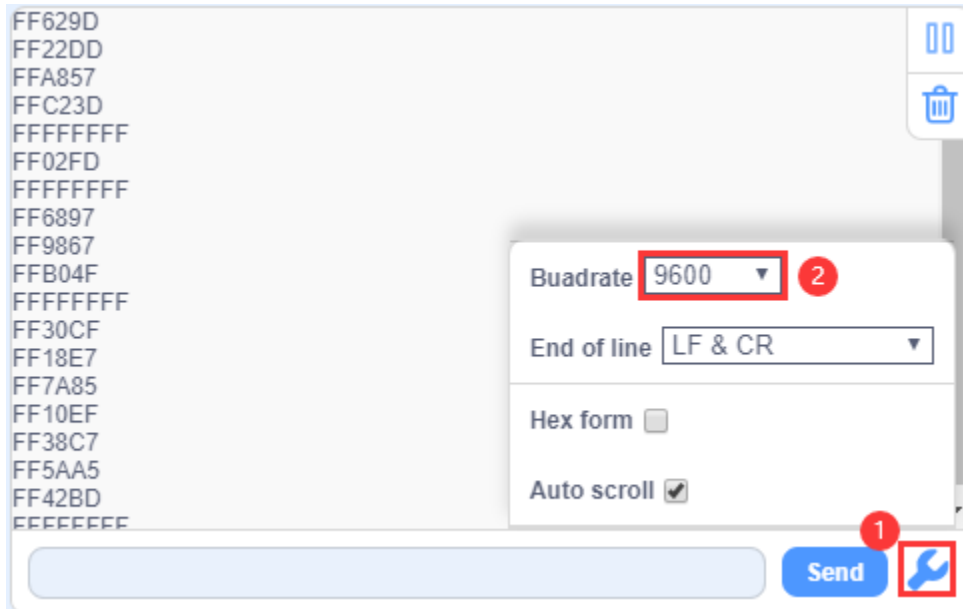
The IR receiver on the PCB board is controlled by IO port(D12) of the Arduino Nano board.



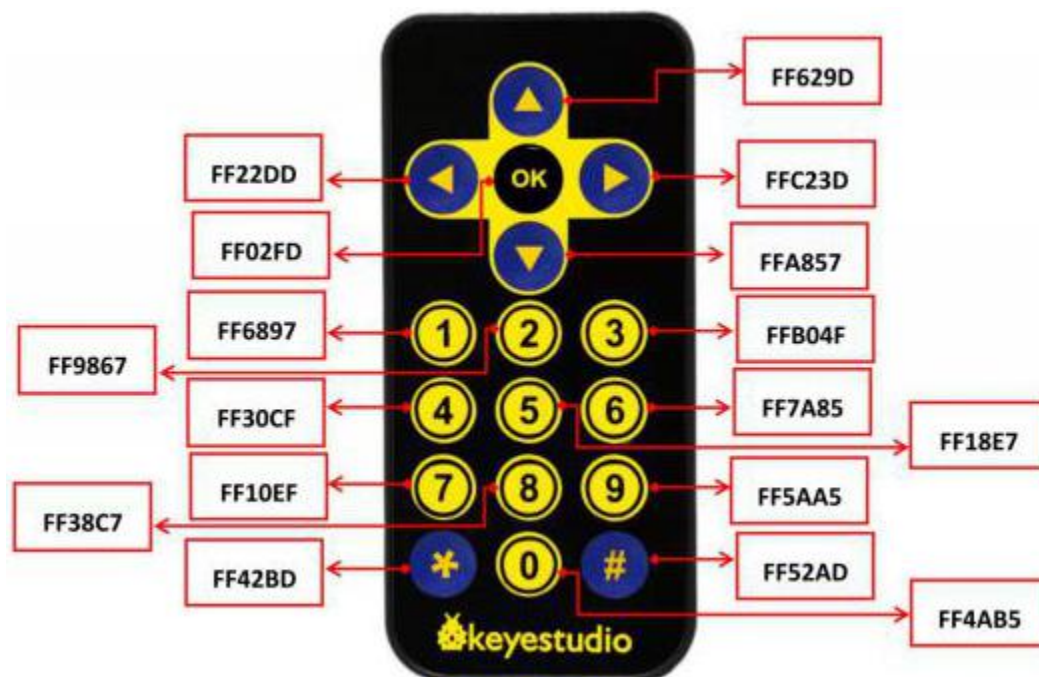
### (5)Test Result

Upload the test code to the Arduino Nano board, power up with a USB cable, open the serial monitor to click  and set to 9600.

Press a key on the IR remote control, you will view a code on the serial monitor. If FFFFFFFF shows up, just ignore it.



Code of each key.

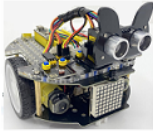

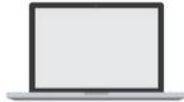




## Project 10.2: IR Remote Control Car






### (1)Description

In the above experiment, we have learned about the knowledge of the 8\*8 dot matrix display, the motor driver and speed regulation, the infrared receiver and the infrared remote control. In this experiment, we will use the infrared remote control and the infrared receiver to control the car.

### (2)Components Required

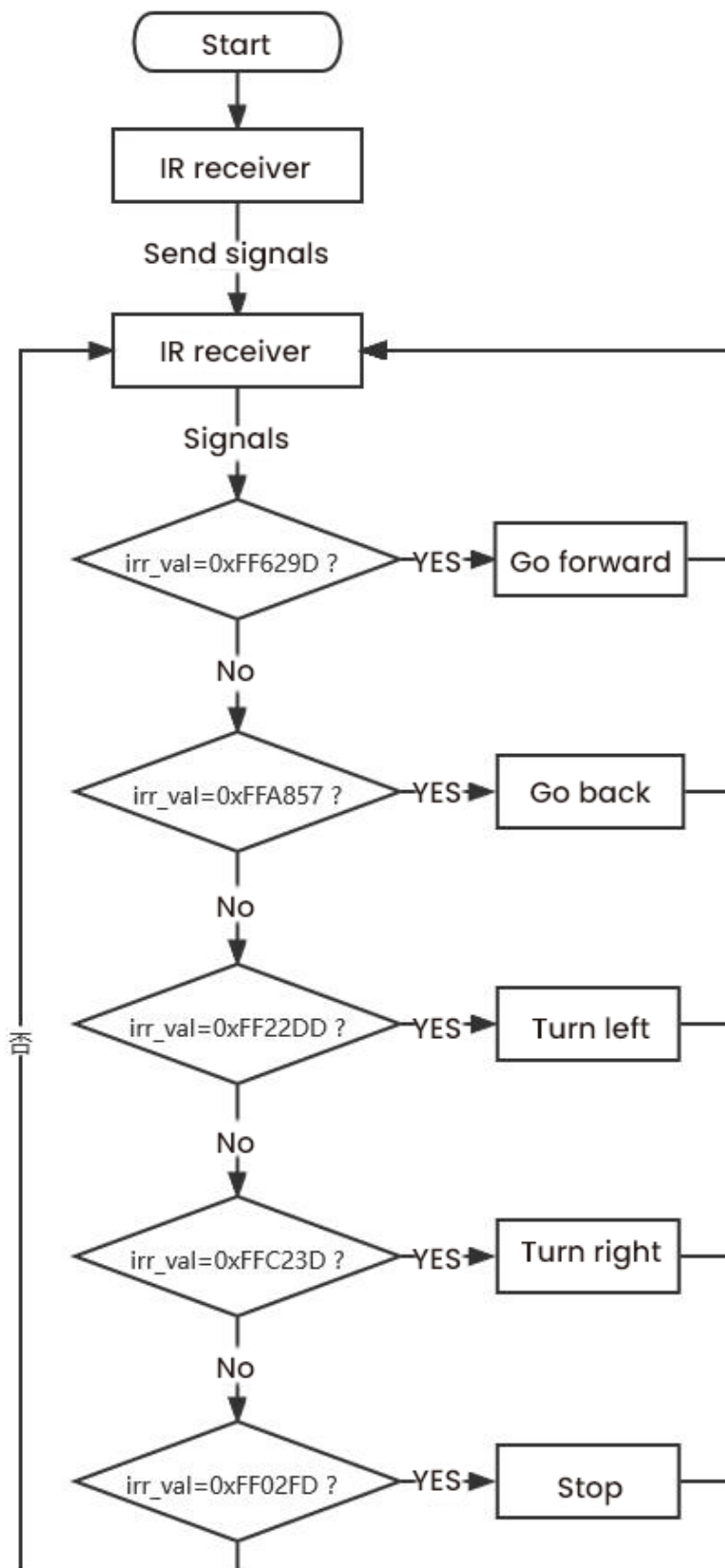
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | Remote Control*1   | 18650 Battery*1   |
|---|---|---|--|---|
|  |  |  |  |  |

### (3)Working Principle

| Keys  | Keys Code | Functions                             |
|---|-----------|---------------------------------------|
|  | FF629D    | Go forwardDisplay “forward” pattern   |
|  | FFA857    | Go backDisplay “back” pattern         |
|  | FF22DD    | Turn leftShow“left” pattern           |
|  | FFC23D    | Turn rightShow“right turning” pattern |
|  | FF02FD    | stopshow“stop” pattern                |



## (4)Flow Chart





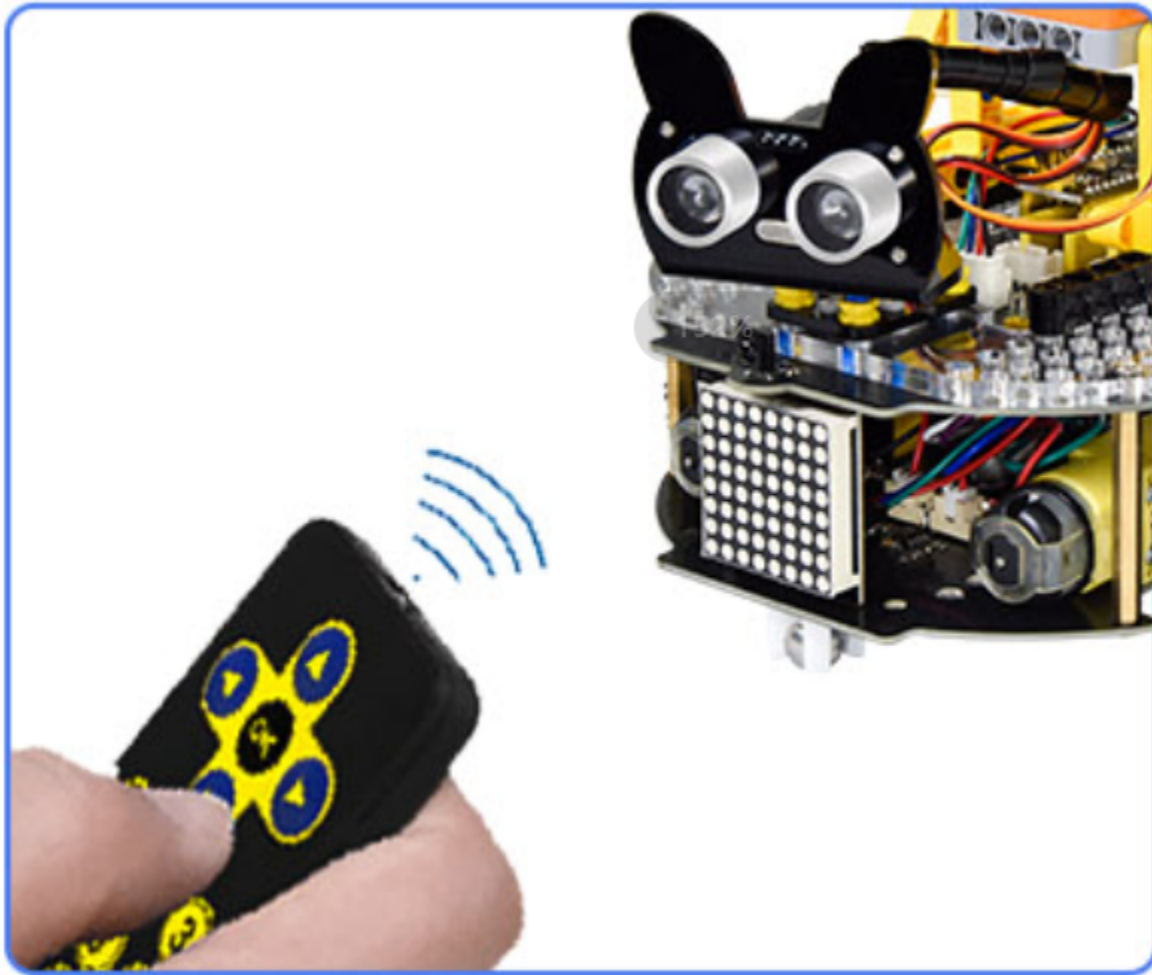


## (5)Test Code



### (6)Test Result

Upload the test code to the Arduino Nano motherboard, install batteries, turn the power switch to the ON end, power up and press a key of the IR remote control. Then the car will make the corresponding movement.



### 9.3.11 Project 11: WIFI Control

In this lesson, we control the car through app. The Beetlebot APP sends commanders to the WIFI ESP-01 module then transfers to it to the microcontroller. By doing this, the car can perform different functions.







## Project 11.1: WIFI Test

### (1)Description

The ESP8266 serial WiFi ESP-01 module is an ultra-low-power UART-WiFi transparent transmission module and designed for mobile devices and IoT applications.

It can achieve networking functions by connecting devices to Wifi internet.

### (2)Components Required

| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | WiFi module*1   | USB Serial ESP-01S WIFI Expansion Module *1  | 18650 Battery*1   |
|---|---|---|---|--|---|
|  |  |  |  |  |  |

### (3)Knowledge



#### USB to ESP-01S WiFi module serial shield:

It is suitable for the ESP-01S WiFi module. Turn the DIP switch on the USB to ESP-01S WiFi module serial Expansion Board to Flash Boot, and plug into computer's USB port. You can use serial debugging tool to test the AT command.

Turn the DIP switch on the USB to ESP-01S WiFi module serial expansion board to the UartDownload, ESP-01 module is at download mode. You can download the firmware to ESP-01 module using AT firmware.



#### ESP8266 serial WIFI ESP-01

ESP8266 serial WiFi ESP-01 is an ultra-low-power UART-WiFi transparent transmission module. It can be widely used in smart grids, intelligent transportation, smart furniture, handheld devices, industrial control and other fields.

**Features**

- Support wireless 802.11 b/g/n standards
- Support STA/AP/STA+AP three modes of operation
- Built-in TCP/IP protocol stack to support multi-channel TCP Client connections
- Supports many Socket AT commands
- Supports UART / GPIO data communication interface
- Supports Smart Link smart networking function
- Supports remote firmware upgrades(OTA)
- Built-in 32-bit MCU, can also be used as an application processor
- Ultra-low-power and highly integrated Wi-Fi chip for battery-powered applications
- Working temperature range: -40°C to + 125°C
- 3.3V single power supply

**(4)Functions****A. Main functions**

The main functions that can be achieved by ESP8266 include: serial port transparent transmission , PWM regulation, GPIO control.

※Serial port transparent transmission: The transmission is reliable with a maximum transmission rate of 460800bps.

※PWM regulation: Adjusting lights and tricolor LED, motor speed control, etc.

※GPIO control: Control switch, relay, etc.

**Working modes**

The ESP8266 module supports three operating modes, STA/AP/STA+AP.

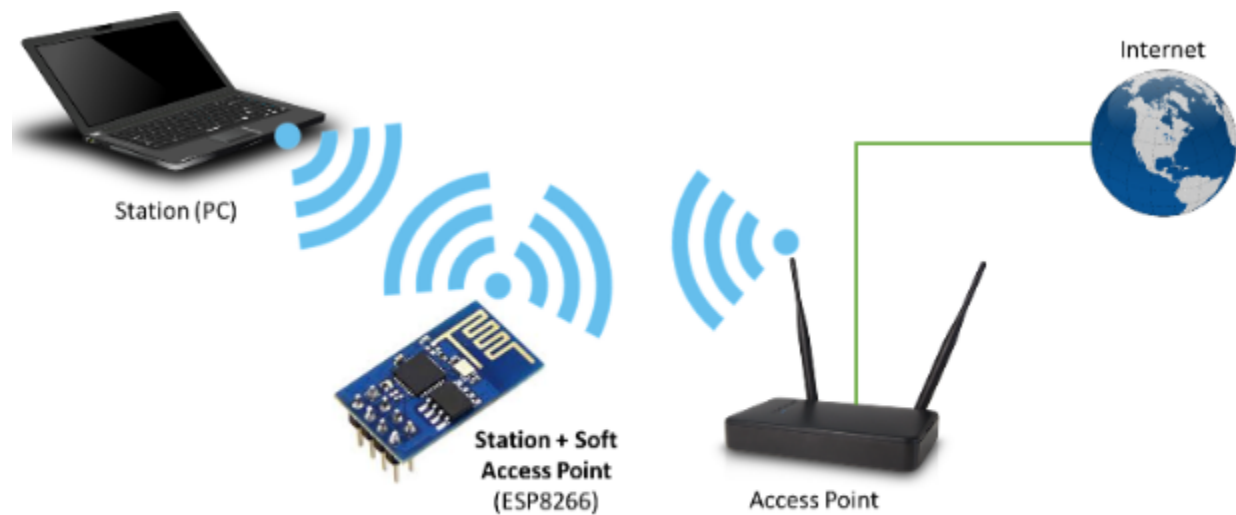
STA mode: The ESP8266 module can access to the Internet through a router, so the mobile phone or computer can remotely control the device through the Internet.



ESP8266 operating in the **Station** mode

AP mode: ESP8266 module, as a hotspot, allows the direct communication with the module and cellphones/computers, achieving wireless control of the local area network (LAN).

STA+AP mode: two modes coexist, that is, the Internet can achieve free switch.



ESP8266 operating in the **Station + Soft Access Point Mode** mode

## (5) Applications

Serial CH340 to Wi-Fi

Industrial transparent transmission DTU

Wi-Fi remote monitoring/control

Toy industry

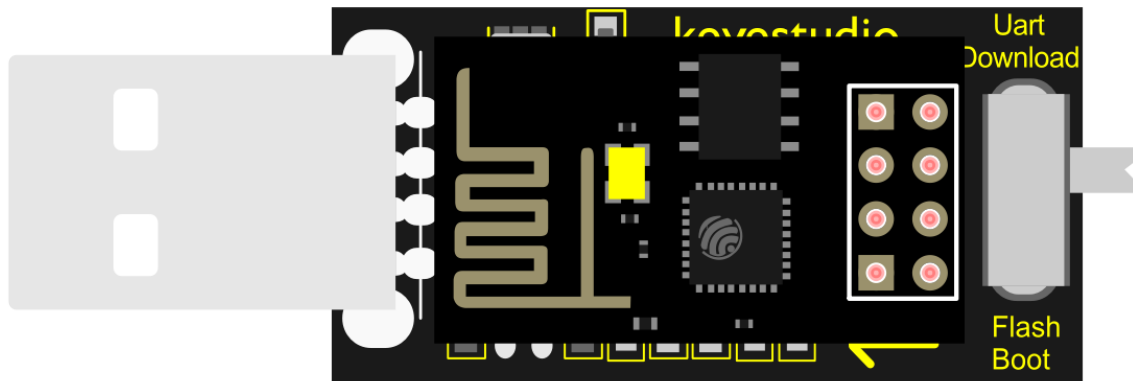
Color LED control

Integrated management of fire protection and security intelligence.

Smart card terminals, wireless POS machines, Wi-Fi cameras, handheld devices, etc.

## (6) Insert the Wifi serial port expansion board into the USB port of your PC

Insert the ESP8266 serial WIFI ESP-01 module into the USB to ESP-01S WIFI expansion board.

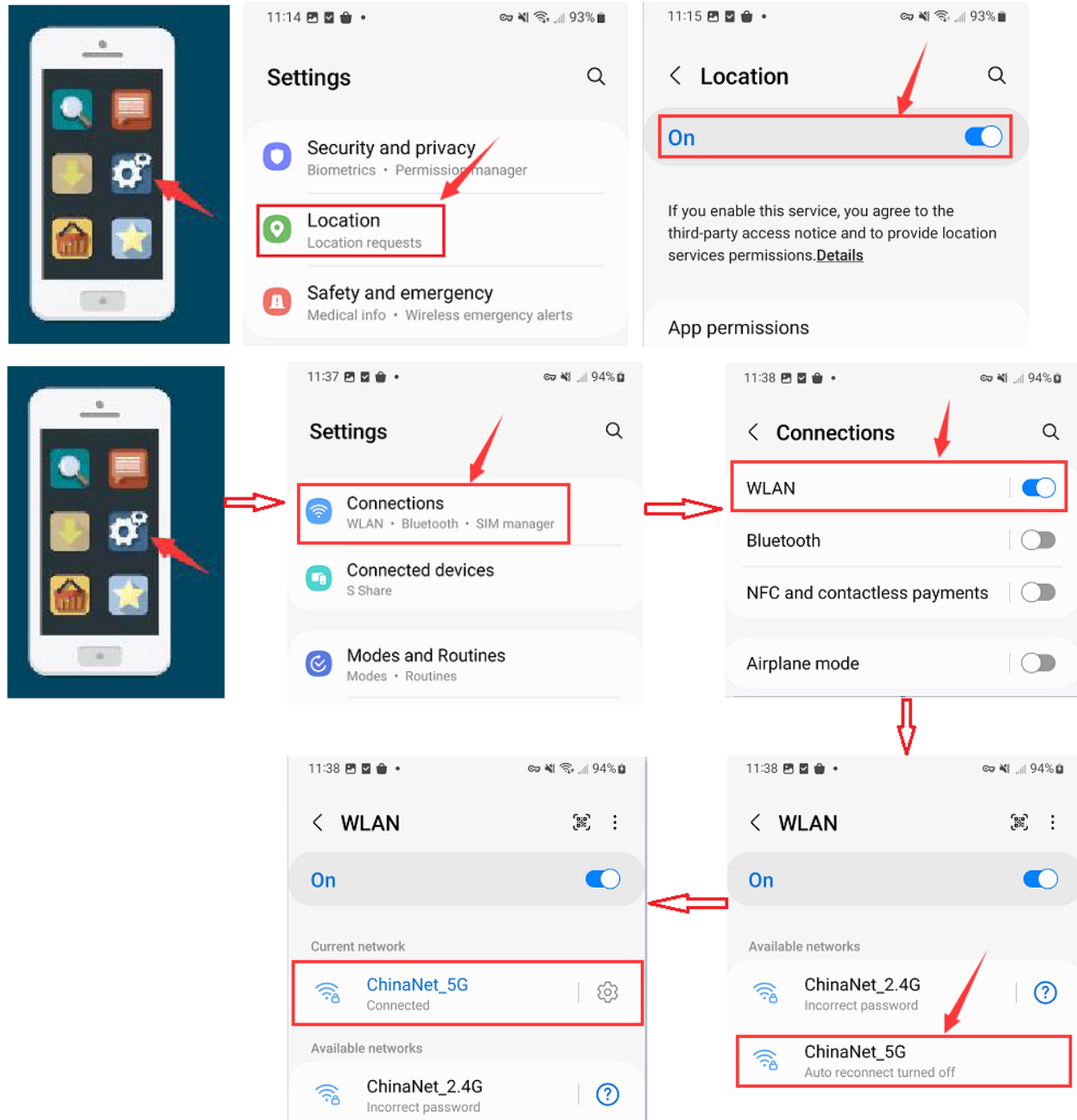


Turn the DIP switch of the USB to ESP-01S WIFI expansion board to UartDownload end and plug it to the USB port of your computer.



**(7)APP:****For Android system**

(1). Turn on the location services of the mobile phone and connect the wifi of yourself.



(2). Search **Beetlebot** in Google Play, or open the following link to download and install the app.

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



11:03

95%



Google Play



beetlebot

Apps &amp; games

Movies

Books

**Beetlebot**  
keyestudio

50+

Downloads

3+

Rated for 3+ ⓘ

Install

See in Play Store app

11:03

95%

Google Play



# Beetlebot

keyestudio

3+

Rated for 3+ ⓘ

Install

Sponsored · People also installed



## Complete account setup

Review your account to continue installing apps on Google Play



(3). Click **Open** button and enter interface of the app.

11:04

95%

Google Play



Beetlebot

keyestudio

Uninstall

Open

Sponsored · Suggested for you

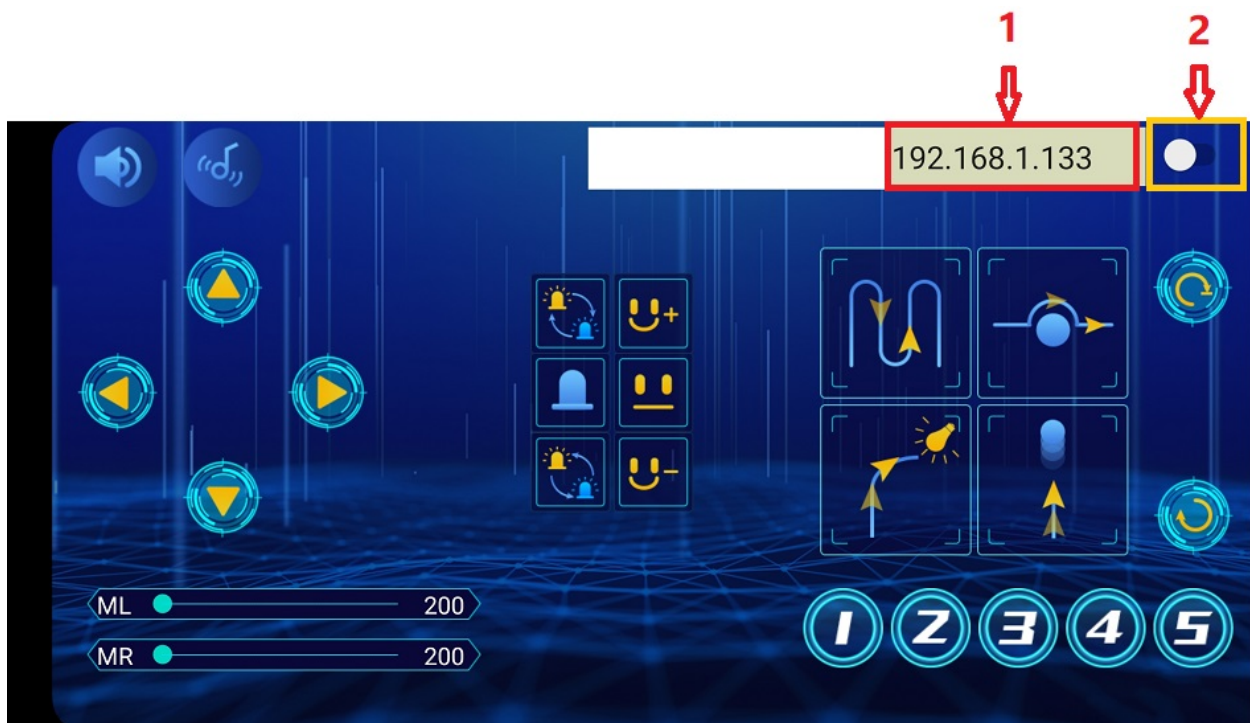




(4). Input the detected Wifi IP address(for example, the IP address in the serial monitor is 192.168.1.134), and Slide



the button to the right to connect Wifi. At same time, the IP address will be shown at the left box, which means that Wifi is connected well.

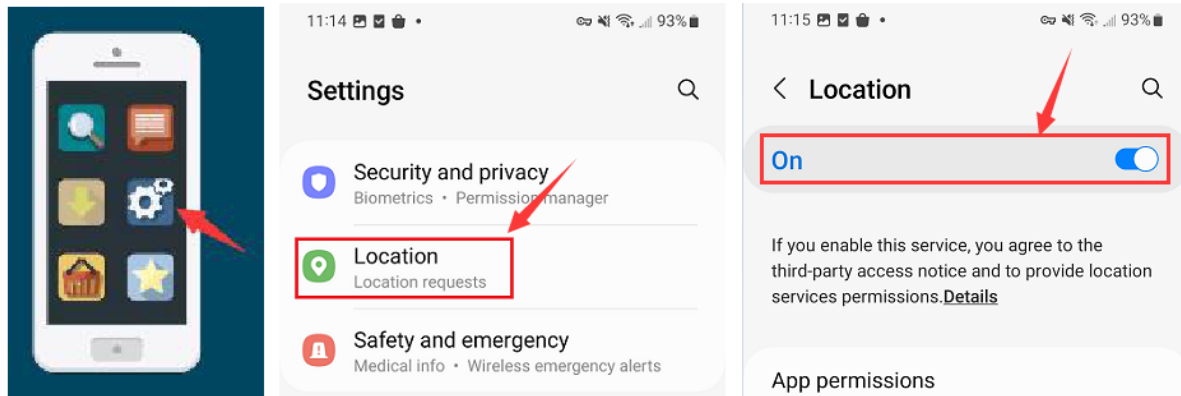


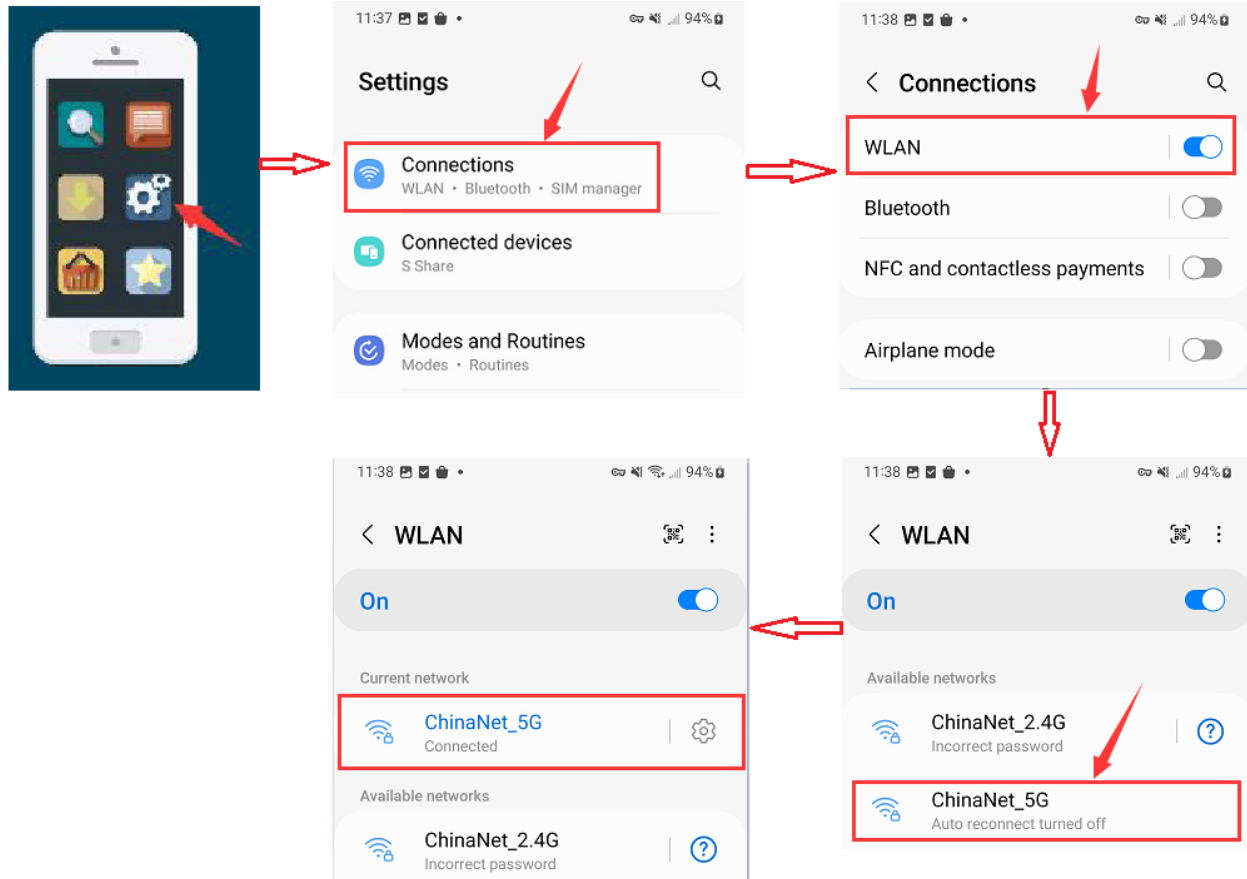


Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

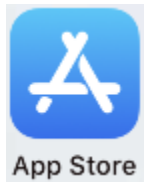
### For IOS system

(1). Turn on the location services of the mobile phone and connect the wifi of yourself.

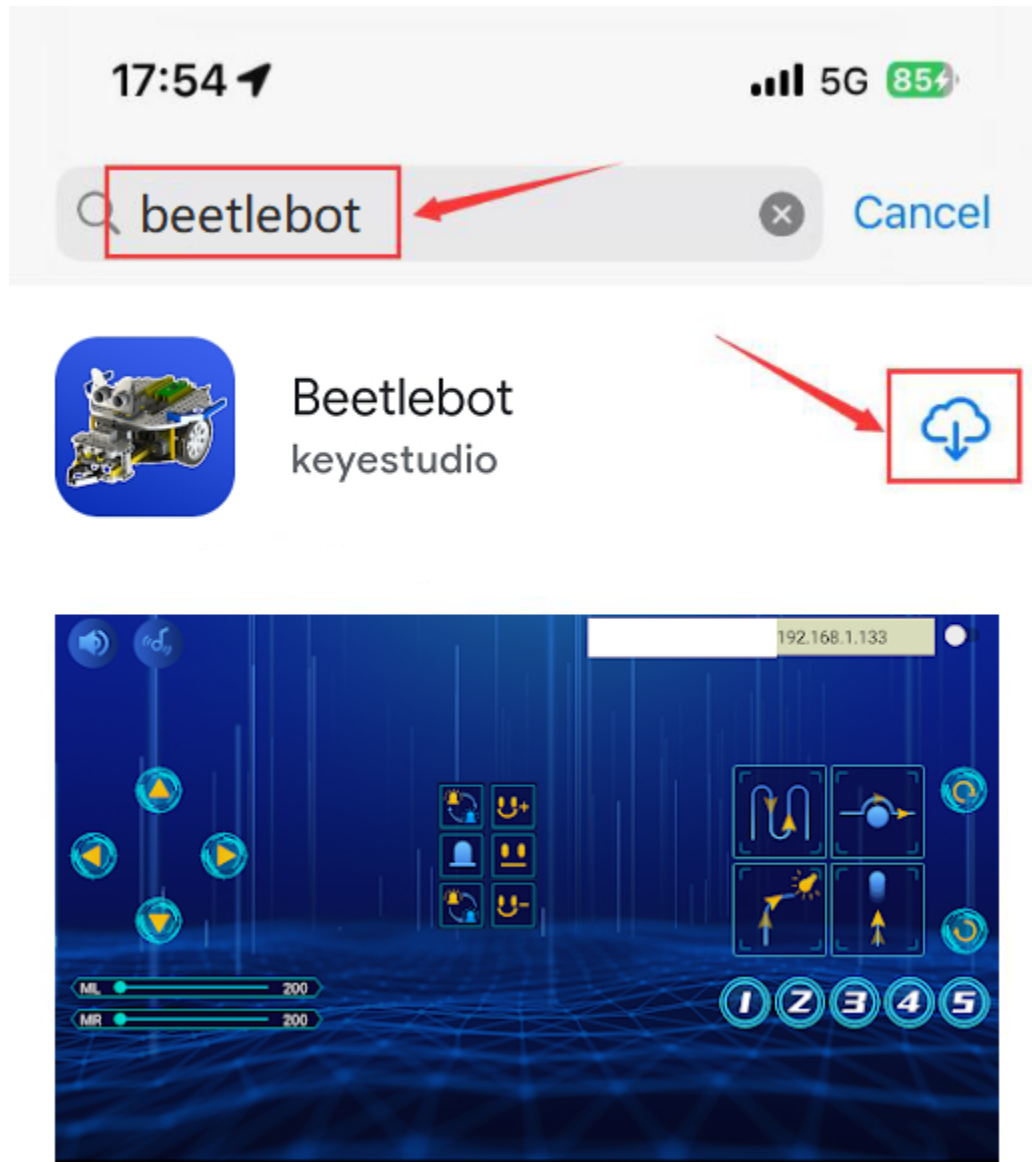




(2). Open App Store

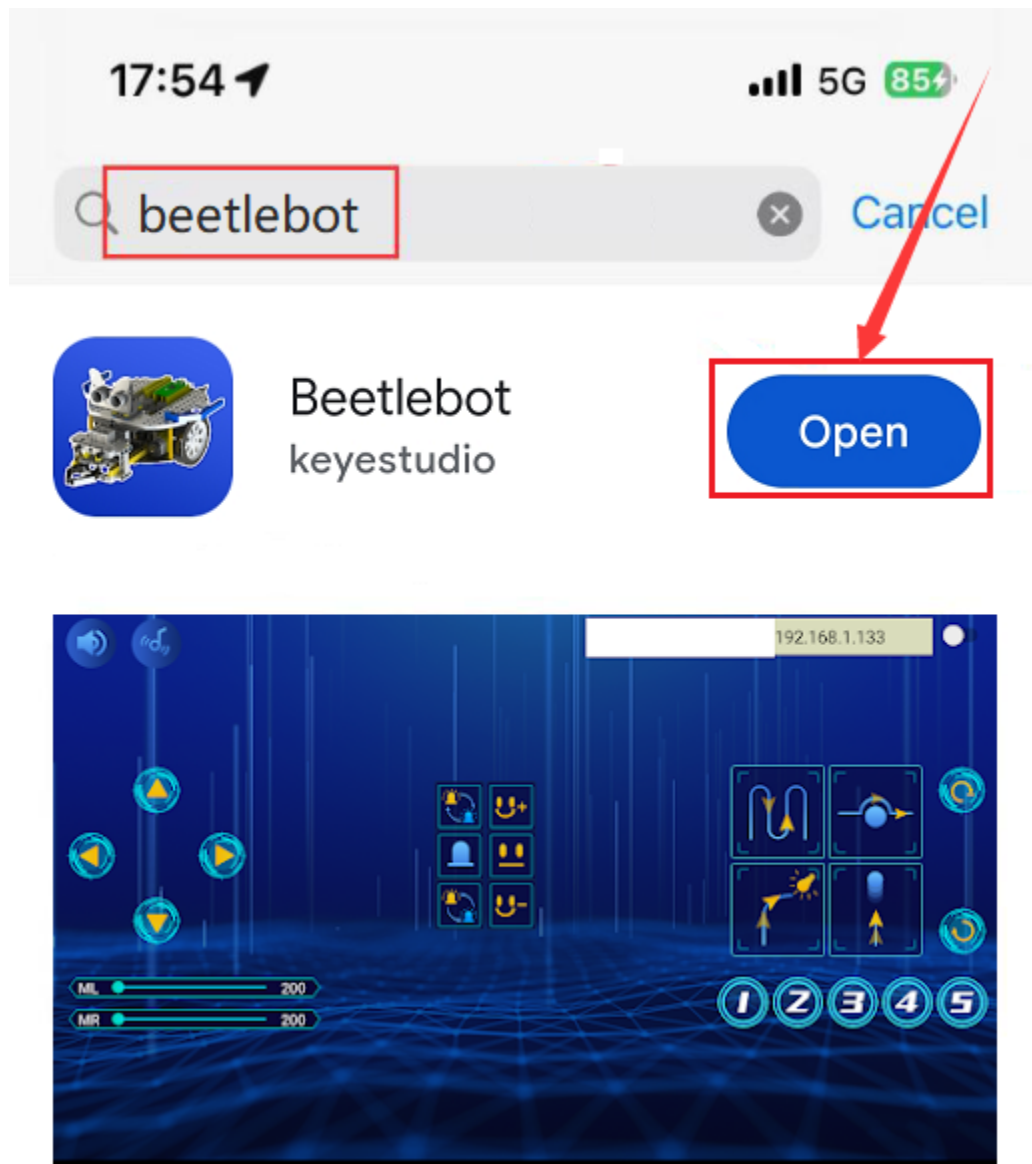


(3). Search **Beetlebot** in App Store click “” to download Beetlebot APP.



(4). Click **Open** button and enter interface of the app.





(5). Input the detected Wifi IP address(for example, the IP address in the serial monitor is 192.168.1.134), and Slide








the button to the right to connect Wifi. At same time, the IP address will be shown at the left box, which means that Wifi is connected well.

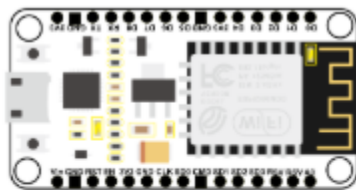
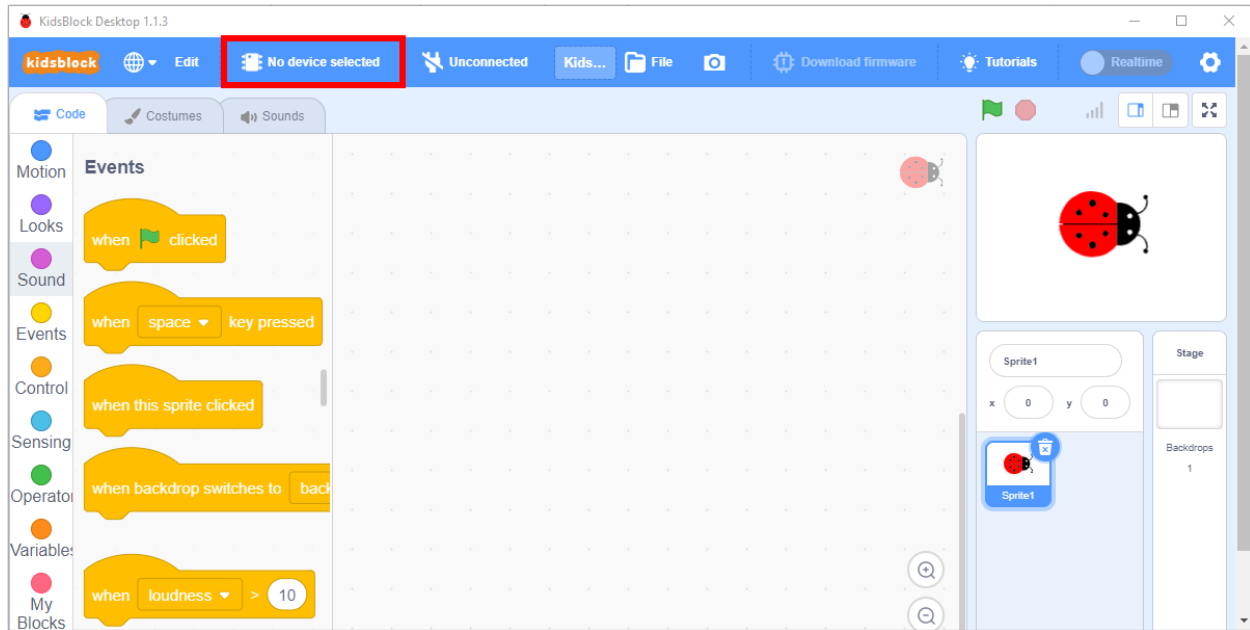


Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

### (8) Add the ESP8266 control board

Click  **No device selected** to enter the main page, select the control board needed. In this project, we select the Plus board and click **Connect**.

Then it is connected, Click **Go to Editor** to return the code editor. Icon  **No device selected** will change into  **Plus** and  **Unconnected** will change into  **USB-SERIAL CP2102 (COM3)**. This means the Plus board is connected to the COM port.



### ESP8266

Low-cost Wi-Fi SOC control board.

Requires



Manufactor

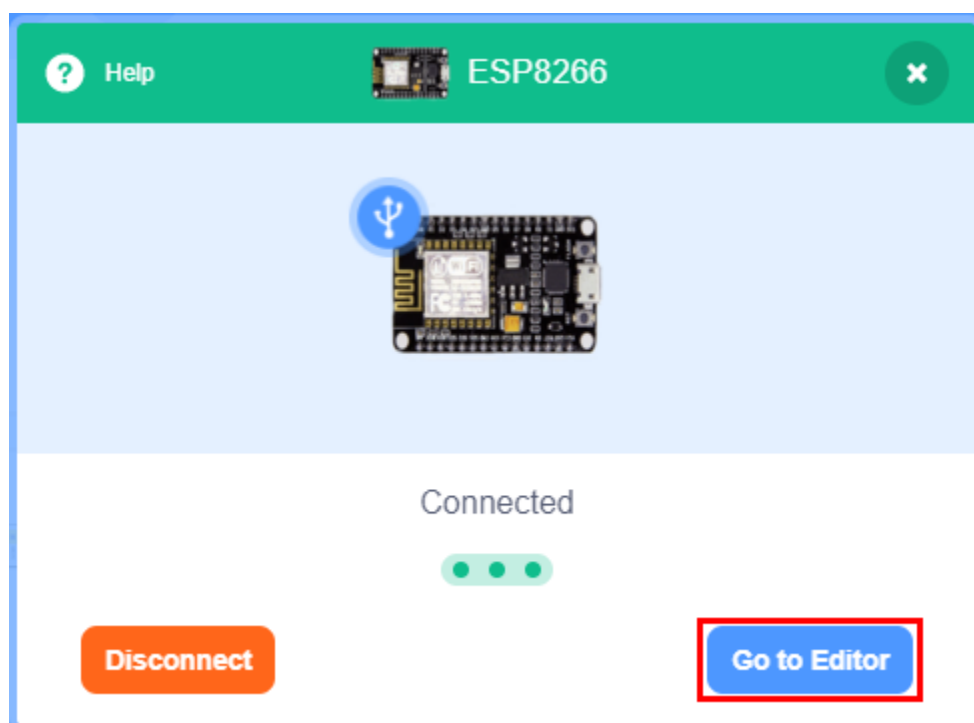
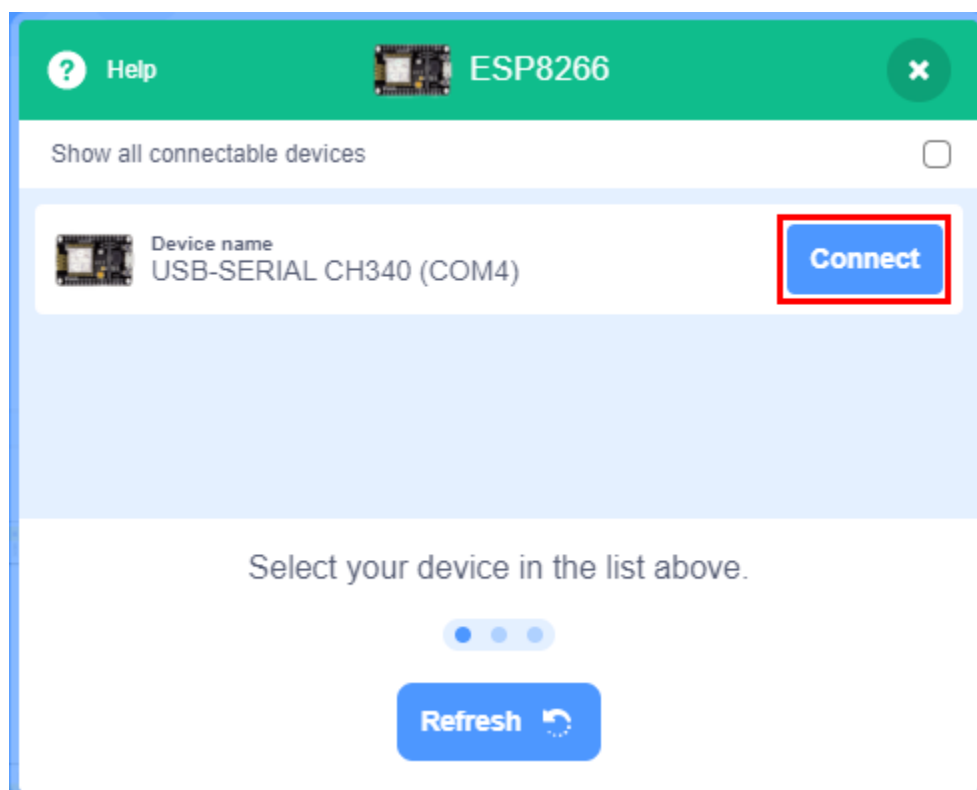
keyes

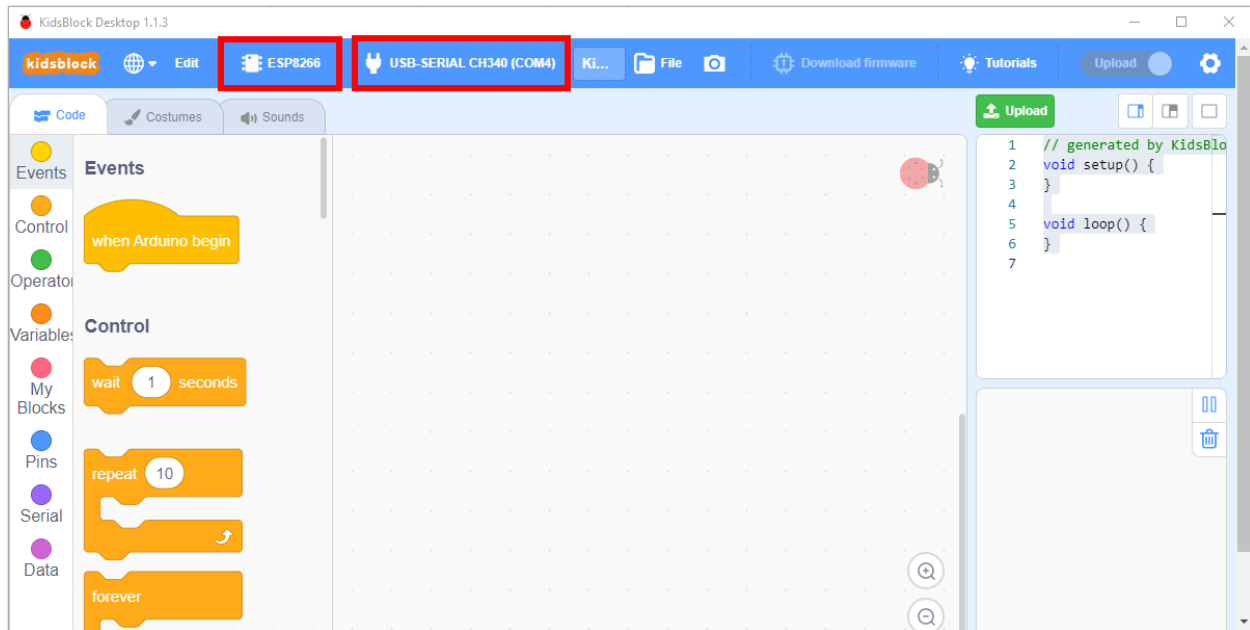
Program mode




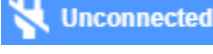


Program language

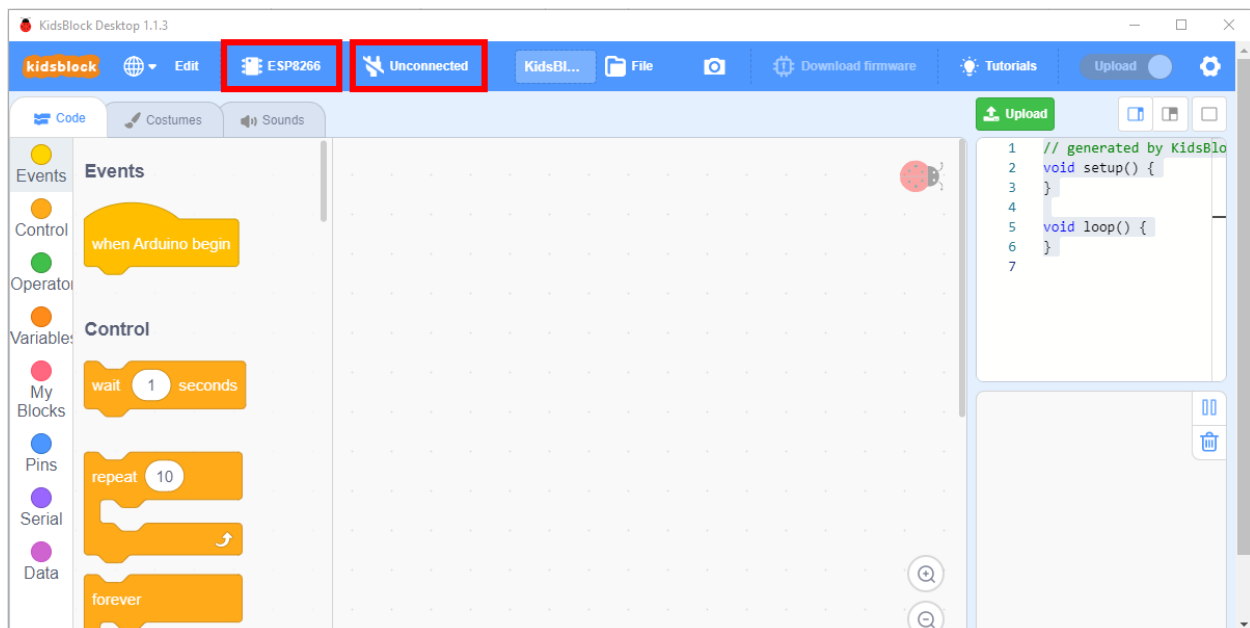


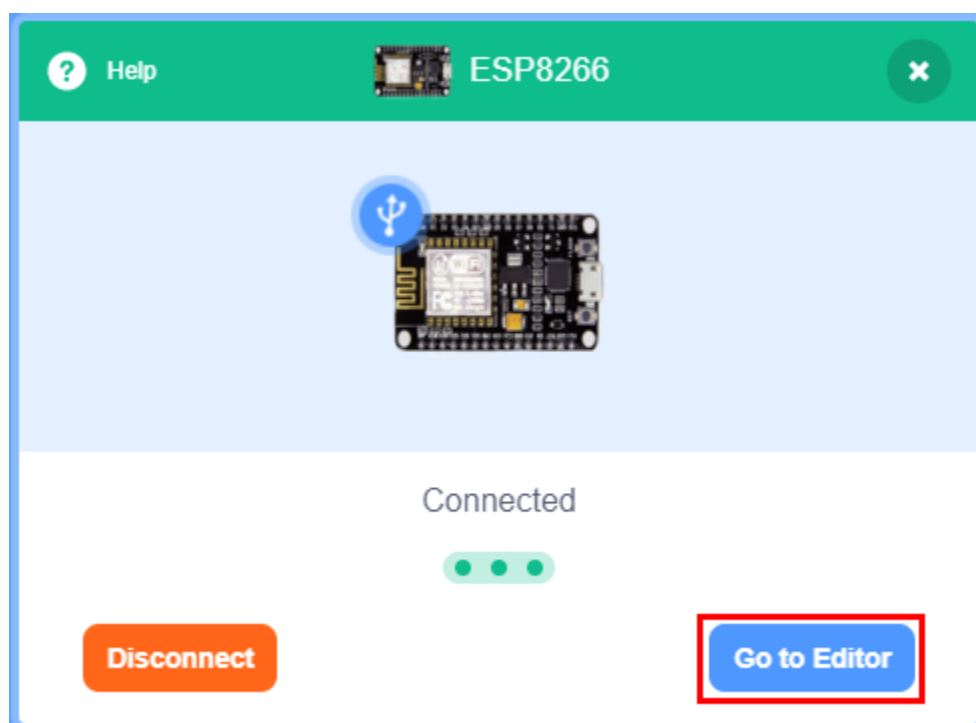
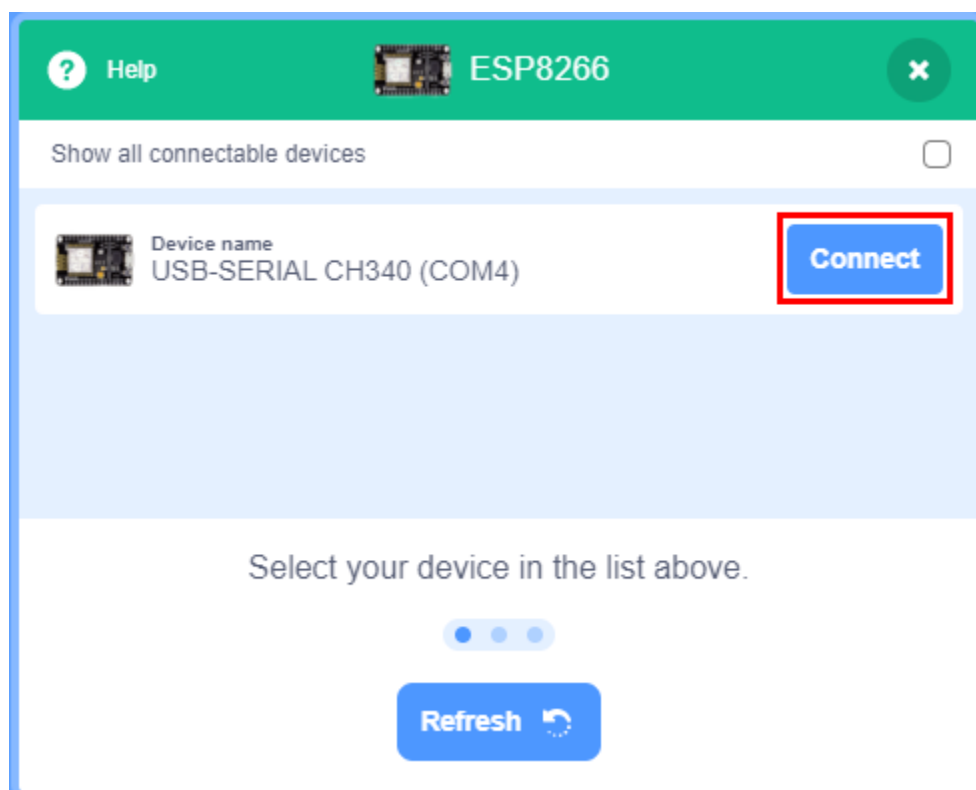


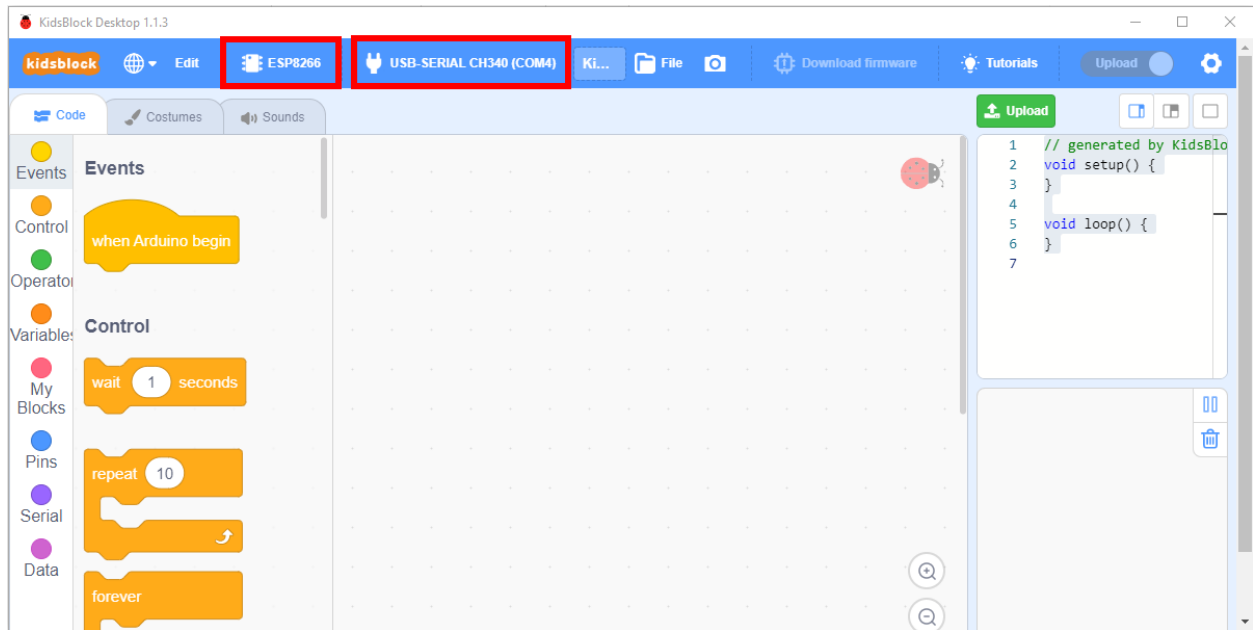


If the ESP8266 board is connected, but icon  doesn't change into . You need to click  to connect the COM port. Click  and **Connect**.


Then you will find a page pop up, showing **Connected**.

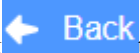


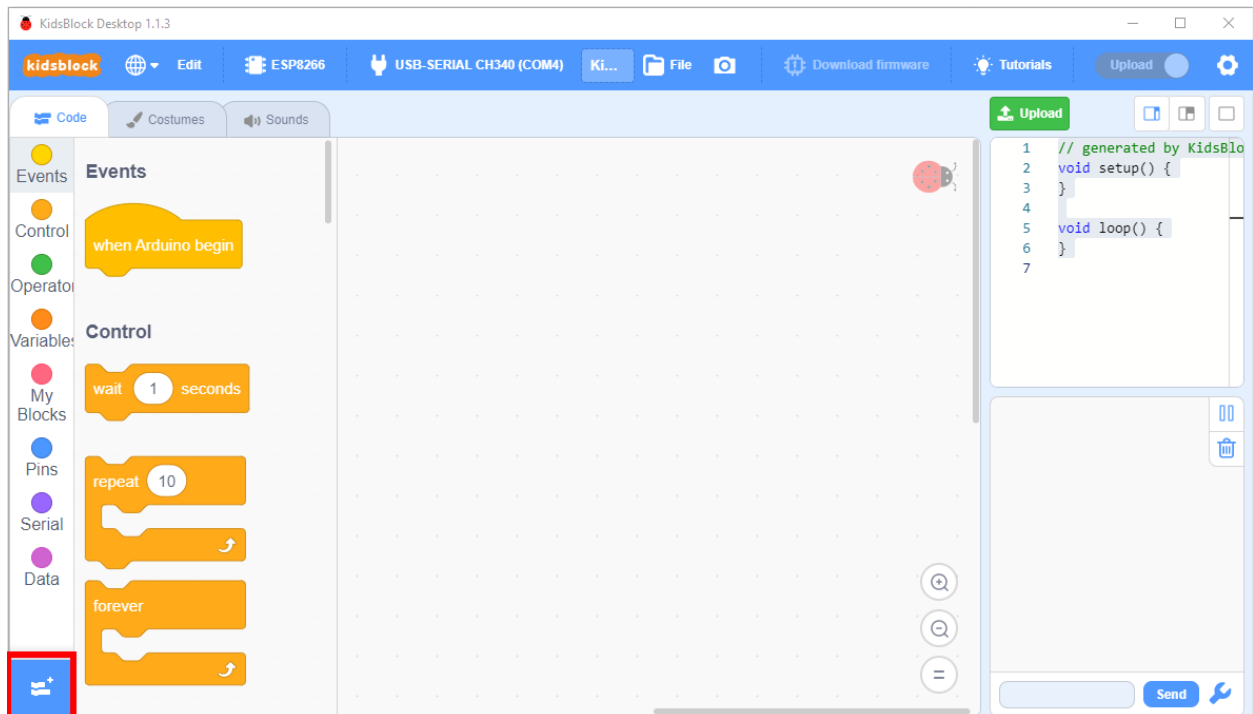


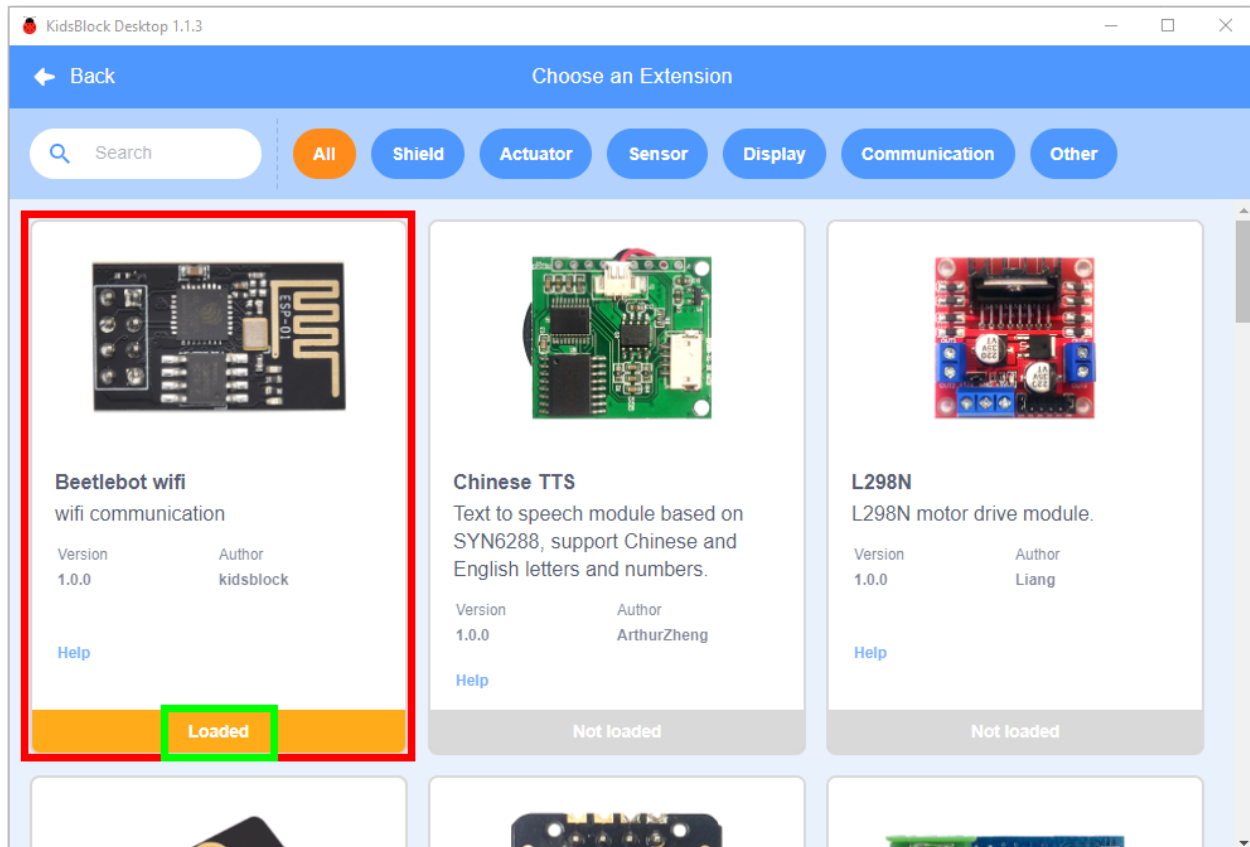
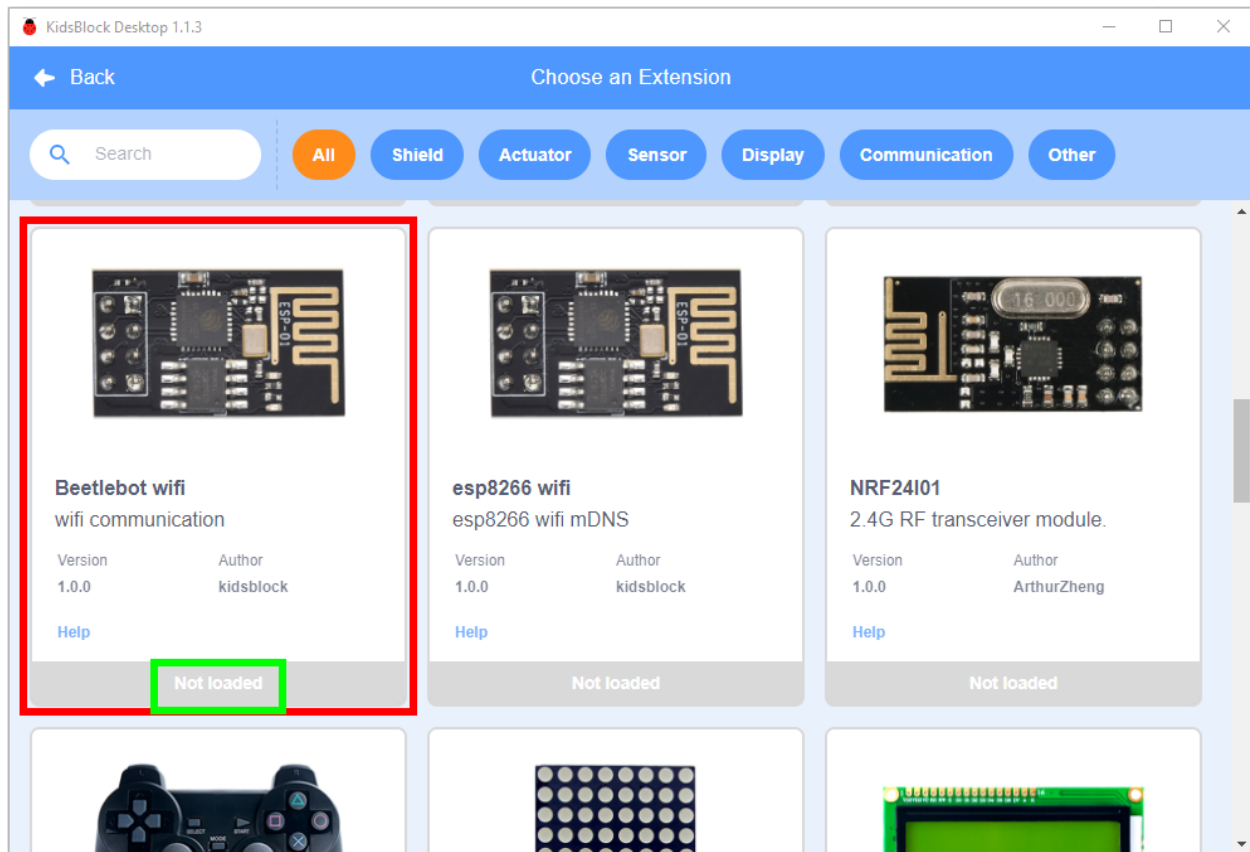


### (9) Add the Beetlebot wifi module

Click  to enter sensor/module expansion interface and click "Beetlebot wifi", "Not loaded" will switch into "loaded". Then the Beetlebot wifi module will be added.

Click  to return the code editor, then you will view the Beetlebot wifi module.







The image shows two screenshots of the KidsBlock Desktop 1.1.3 interface.

**Top Screenshot: Choose an Extension**

The interface has a blue header with a "Back" button (highlighted with a red box) and a search bar. Below the header are tabs for "All", "Shield", "Actuator", "Sensor", "Display", "Communication", and "Other". The "All" tab is selected.

Three extension cards are visible:

- Beetlebot wifi**: wifi communication. Version 1.0.0, Author kidsblock. Status: Loaded (orange bar).
- Chinese TTS**: Text to speech module based on SYN6288, support Chinese and English letters and numbers. Version 1.0.0, Author ArthurZheng. Status: Not loaded (grey bar).
- L298N**: L298N motor drive module. Version 1.0.0, Author Liang. Status: Not loaded (grey bar).

**Bottom Screenshot: Code Editor**

The interface shows the code editor with a toolbar at the top (kidsblock, Edit, ESP8266, USB-SERIAL CH340 (COM4), Ki..., File, Camera, Download firmware, Tutorials, Upload, Settings). The left sidebar shows categories: Events, Control, Data, Operator, Variables, My Blocks, Pins, Serial, and Data.

The code editor contains the following blocks:

- serial available data length
- serial read data
- map 50 from ( 1 , 100 ) to ( 1 , 1000 )
- constrain 50 between ( 1 , 100 )
- convert 123 to integer
- convert 97 to ASCII character
- convert a to ASCII number
- esp8266\_wifi
- ESP8266 connect the wifi ssid ChinaNet-2.4G-0DF0 password ChinaNet@233

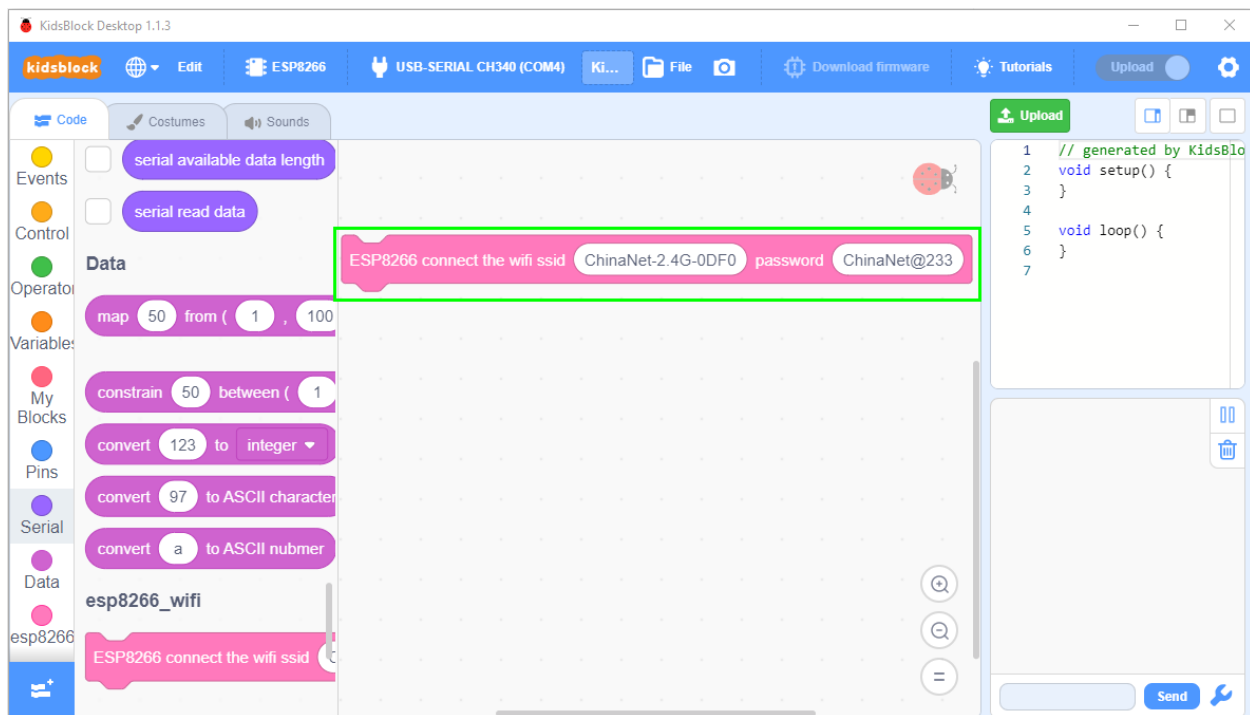
The code editor also shows a code window on the right with the following code:

```
1 // generated by KidsBlo
2 void setup() {
3 }
4
5 void loop() {
6 }
7
```

The "Send" button is visible at the bottom right.

## (10)ESP8266 Code

If there is no wifi in your home, just enable your shared wifi on the phone.



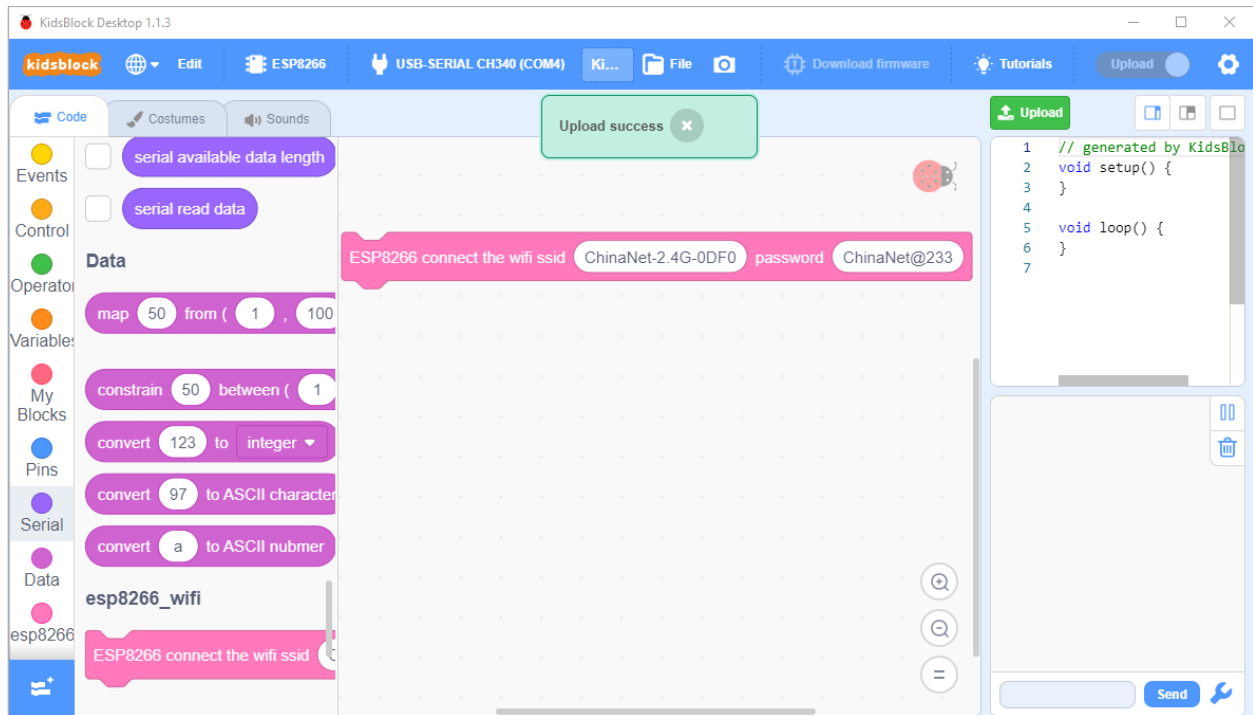
Note You need to change the wifi name and password into yours.



After the Wifi name and Wifi password are changed, plug the USB to ESP-01S WIFI expansion board to the USB port of the computer, turn its DIP switch to the “**Uart Download**” end and click “**Upload**” .

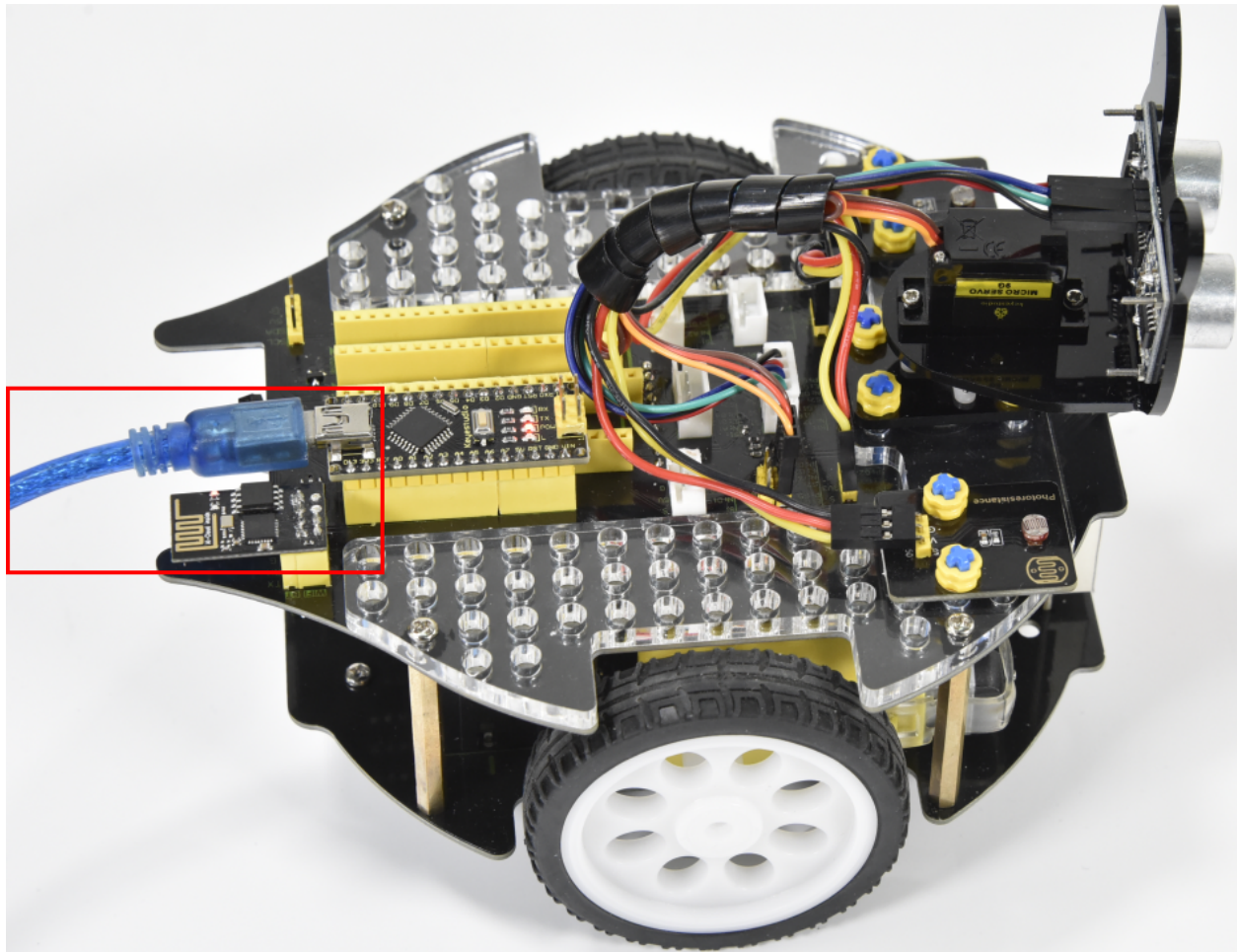
Upload the ESP8266 code to the ESP8266 to WIFI ESP-01 module.

Note: if the code is uploaded successfully, reboot the ESP-01S WIFI expansion board.








After the code is uploaded. Unplug the USB to ESP-01S WIFI expansion board and ESP8266 serial WIFI ESP-01 module.

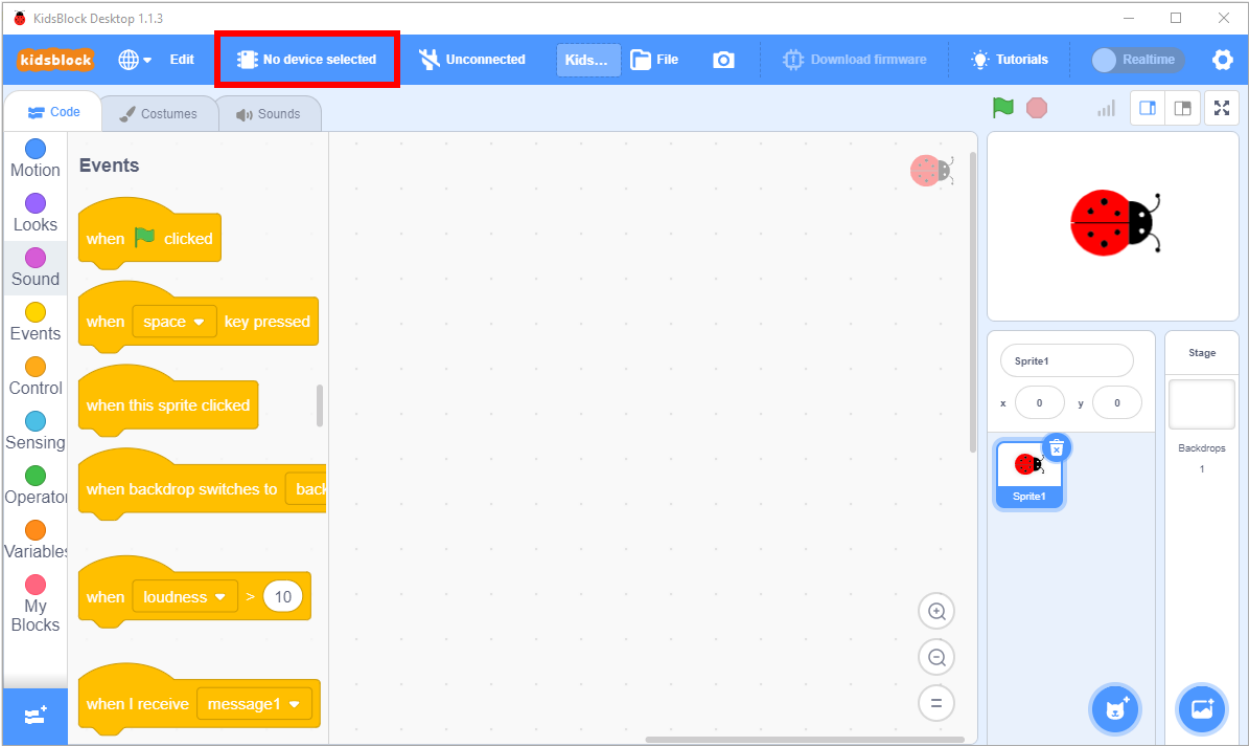
### (11)Interface the Arduino Nano board

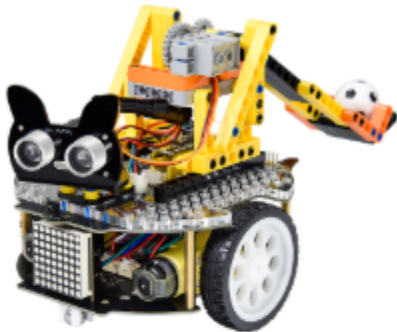


### (12)Set the interface of Beetlebot:

Open KidsBlock to click  **No device selected** to enter the main page. Select **Beetlebot** and click **Connect** then the **Beetlebot** is connected.

Click “**Go to Editor**” to return code editor,  **No device selected** will change into  **Beetlebot** and  **Unconnected** into  **USB-SERIAL CH340 (COM5)**; this indicates the device is connected to the port.







### Beetlebot

Beetlebot

Requires




Program mode

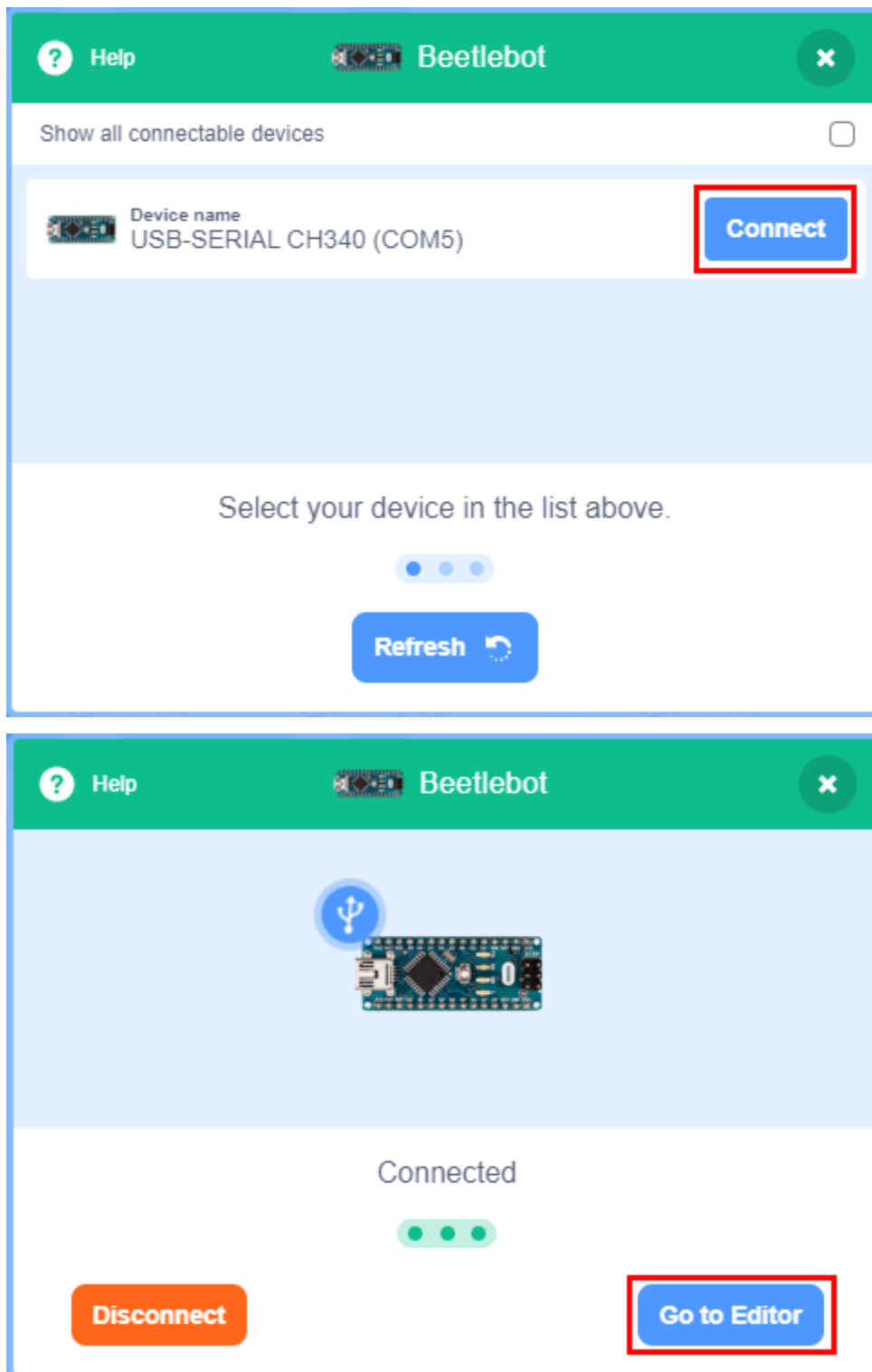


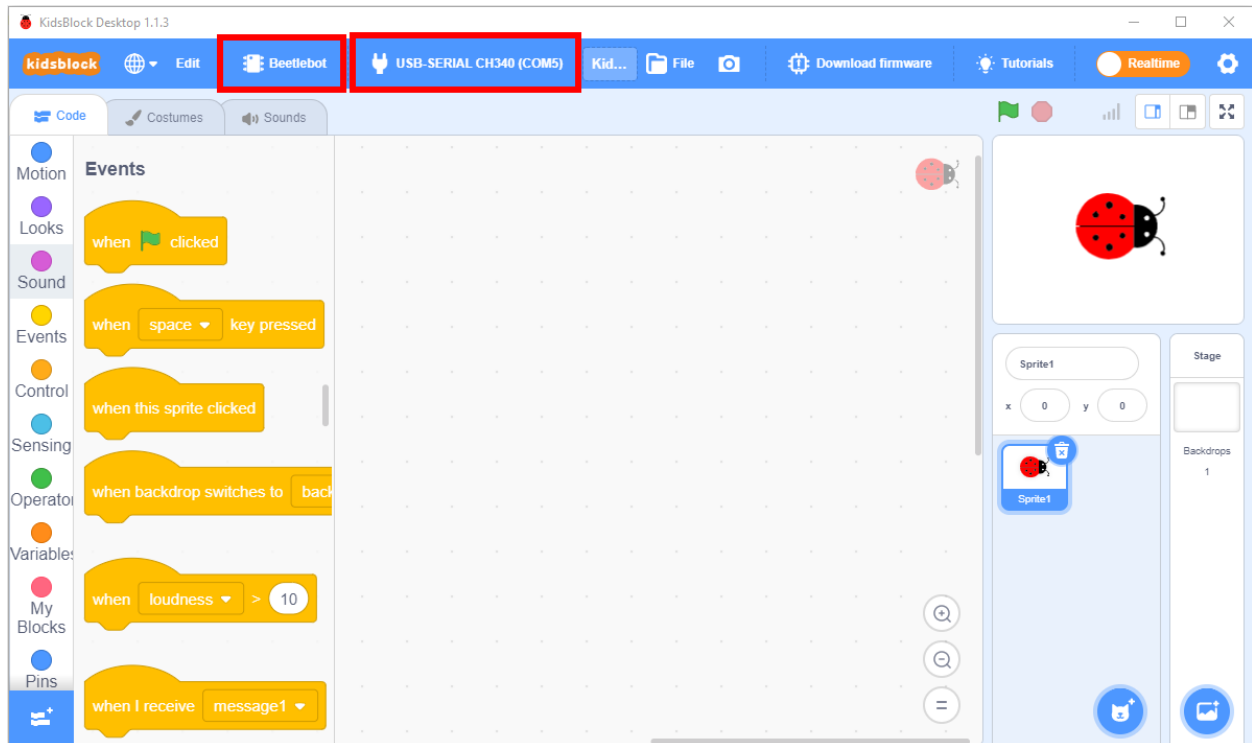
Manufactor

keyes

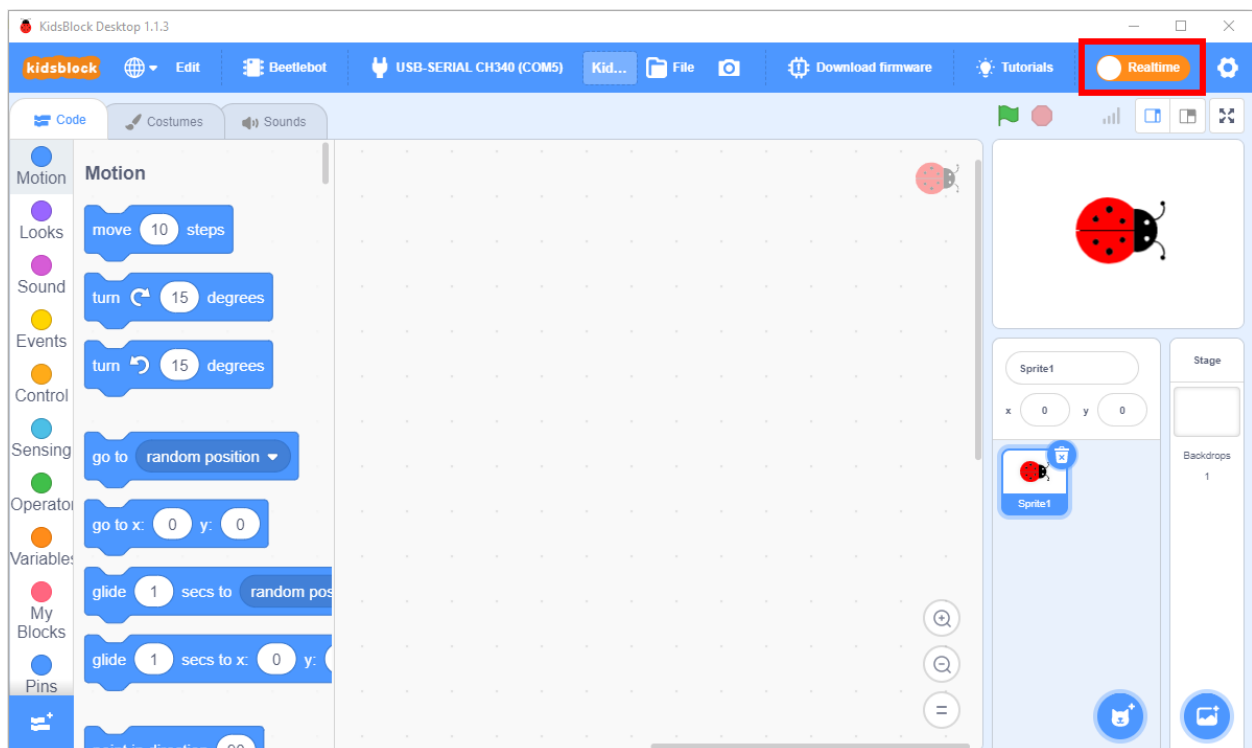
Program language

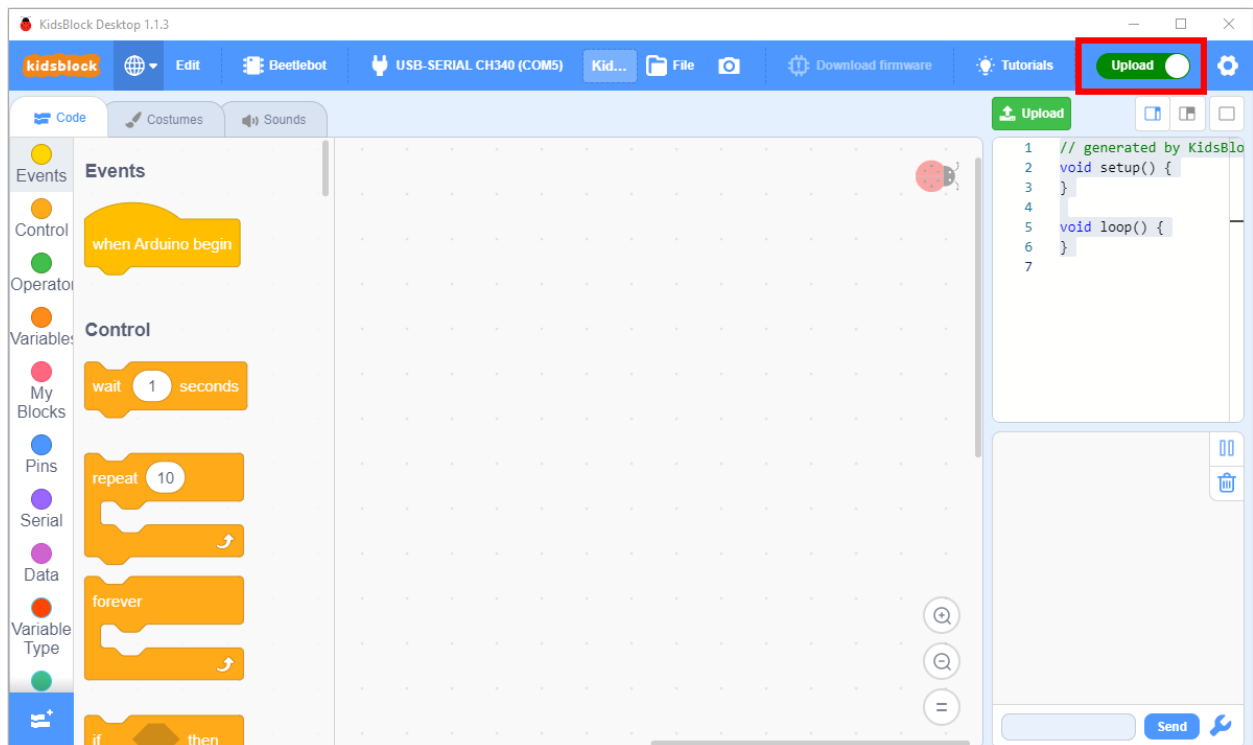






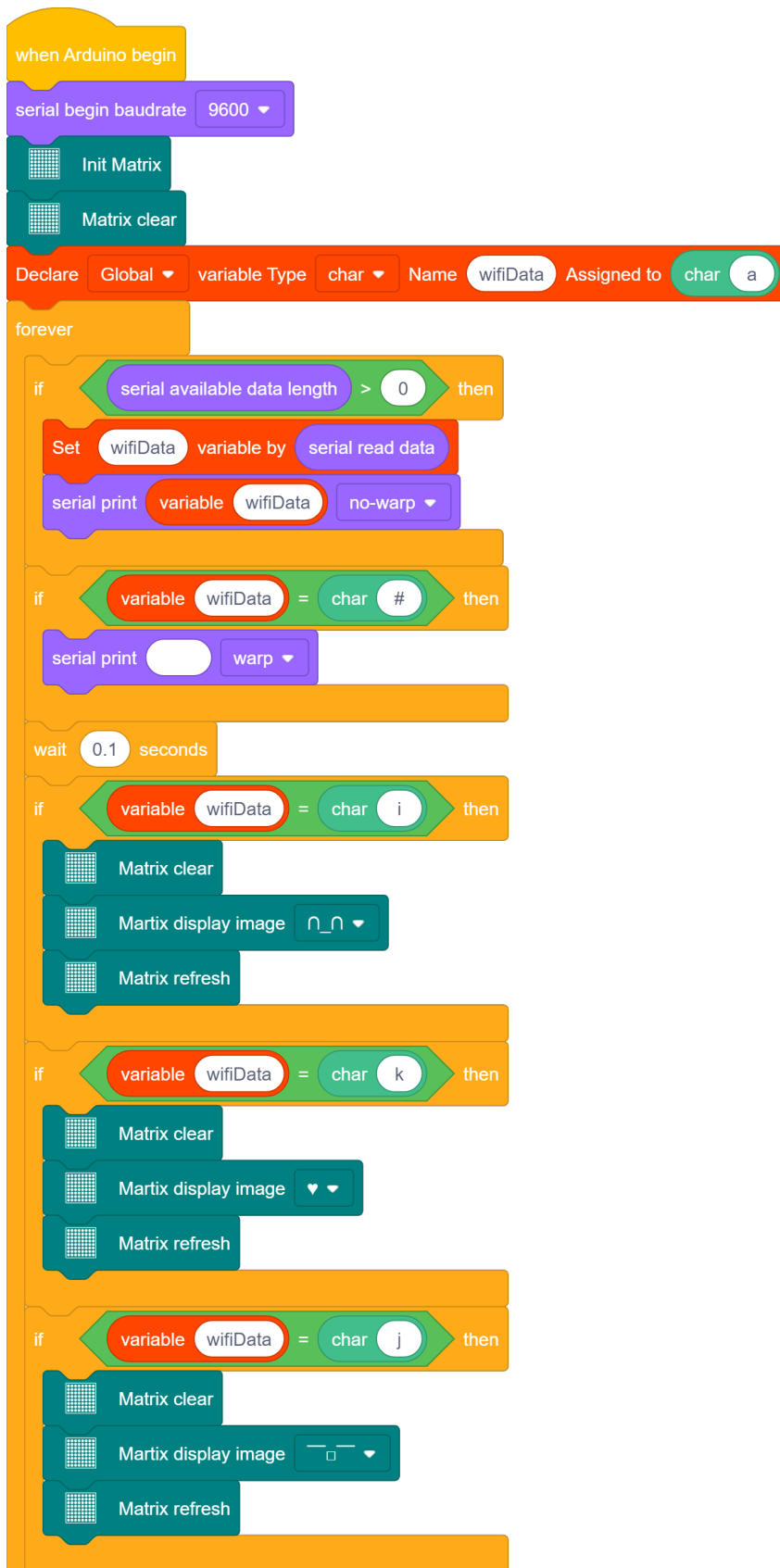
Then click  to switch mode. The  will change into .







## (13)Arduino Nano Test Code

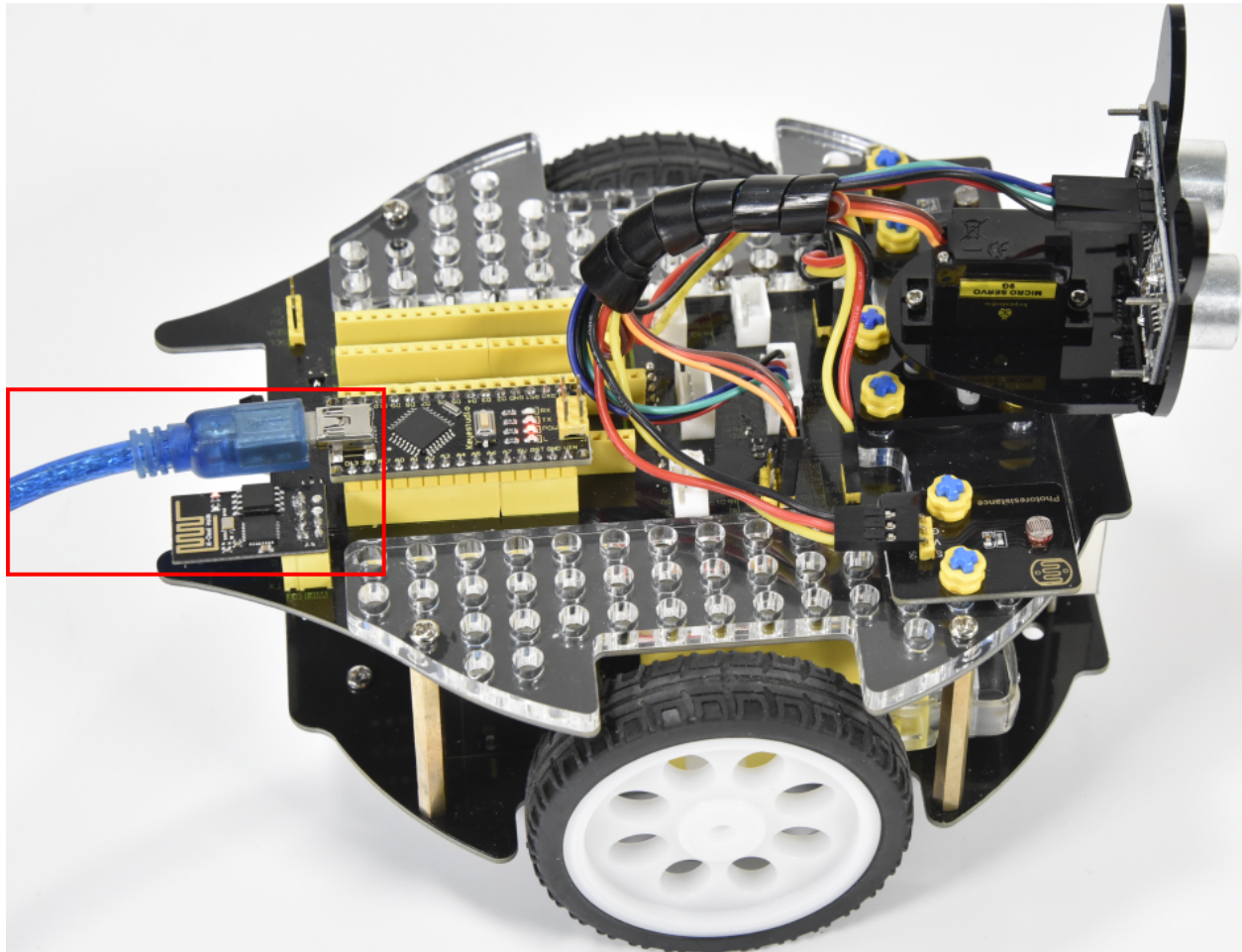


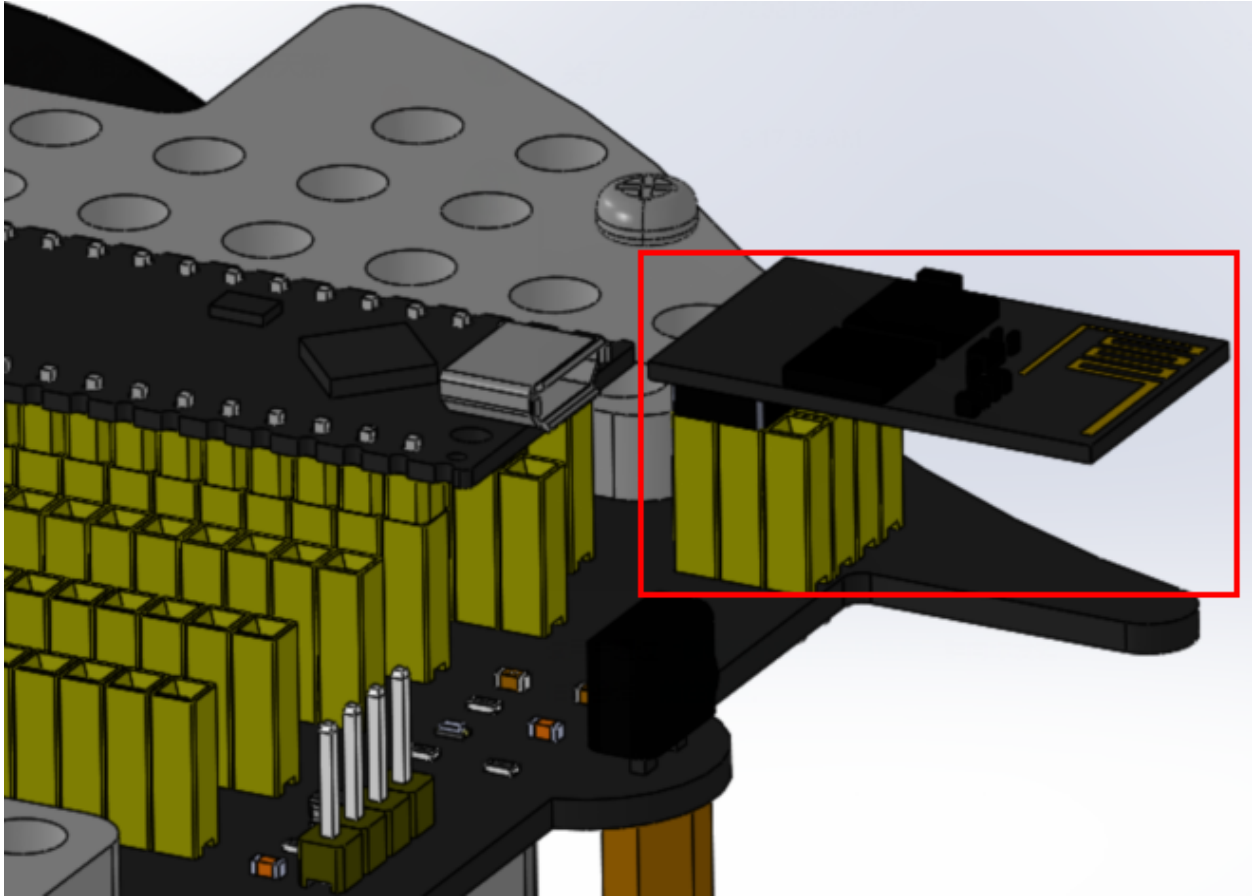
### (14)Test Result


Click **Upload** to upload the test code to the Arduino Nano board.

Then insert the ESP8266 serial WIFI ESP-01 module into the WiFi port of the PCB board.

(Notekeep the USB cable connected.)






Click  and set baud rate to 9600. Then the serial monitor will show your IP address of Wifi.  
The IP address of Wifi sometimes changes. If the original doesn't change, check the IP address of Wifi again.



Input the detected Wifi IP address(for example, the IP address in the serial monitor is 192.168.1.134), and Slide the

button  to the right to connect Wifi. At same time, the IP address will be shown at the left box, which means that Wifi is connected well.

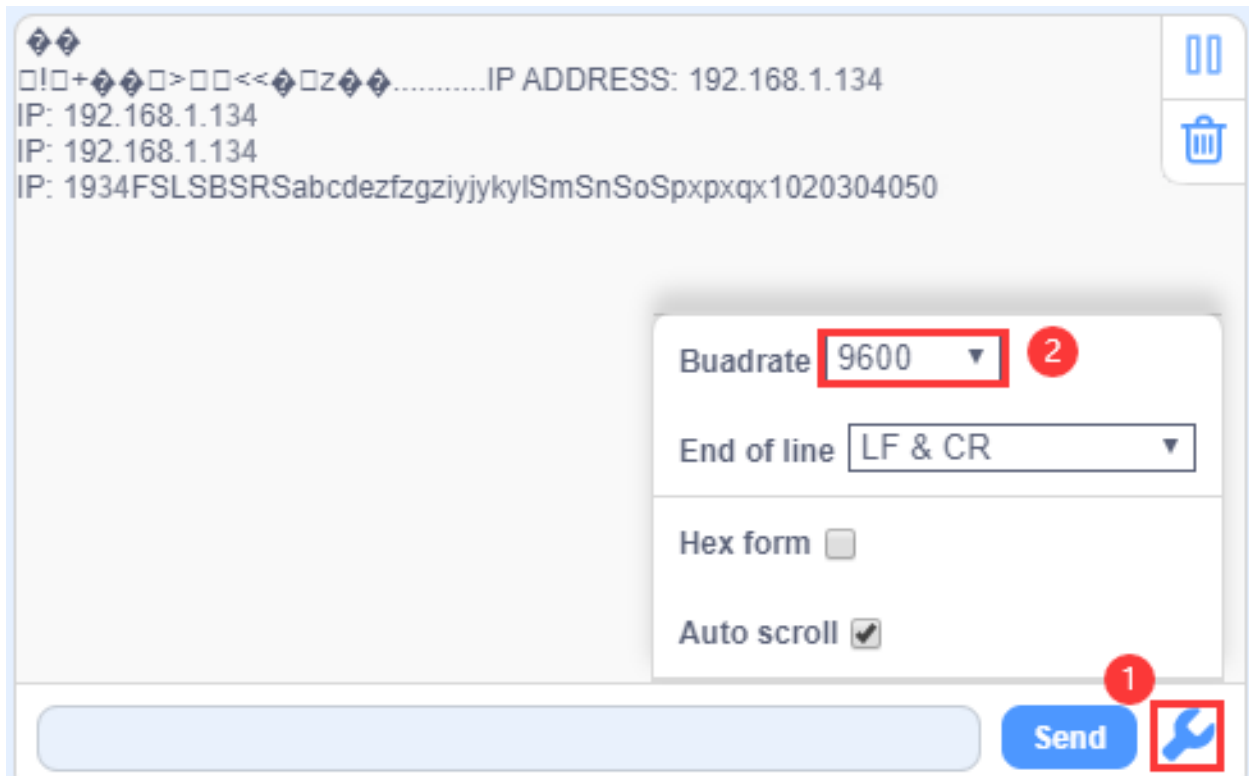




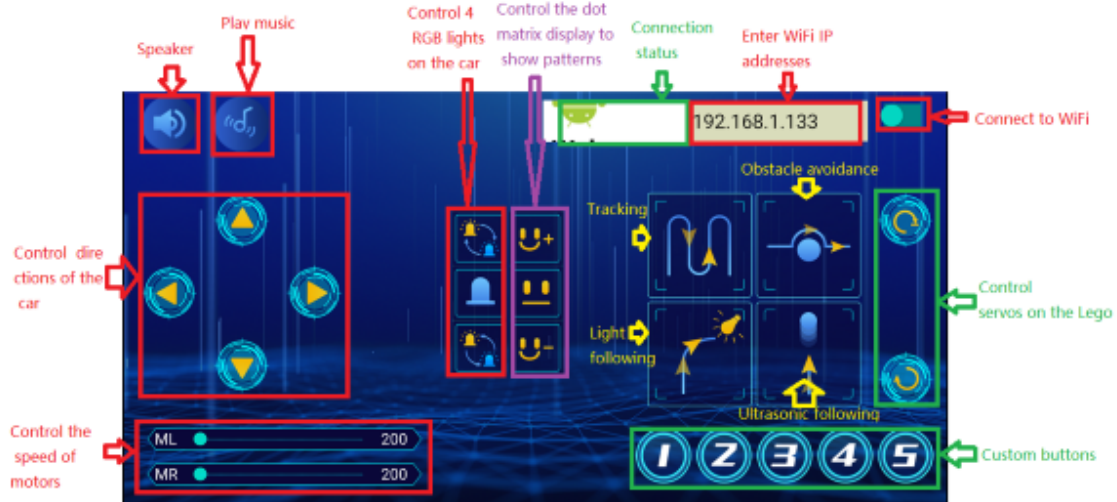
Note: Click buttons on the APP, the blue indicator on the ESP8266 serial WIFI ESP-01 module will flash, indicating that the APP has been connected to WIFI.

After the APP has connected to the WIFI, start the following operations:

Click buttons on the app, the serial monitor will print some control characters, as shown below.



## Interface of App







Click  a “smile” pattern will be displayed click  , “” will be shown click  , “” will be shown.









### Project 11.2: Multi-purpose Car

#### (1)Description

In this project we will demonstrate multiple functions of the Beetlebot car through app.

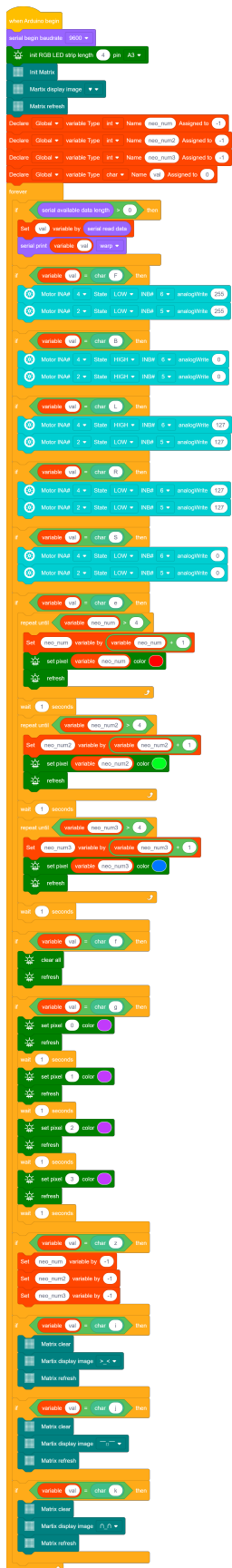
#### (2)Components Required

|   |   |   |   |  |   |
|---|---|---|---|--|---|
| Robot without Wifi module*1   | USB Cable*1   | Computer*1  | WiFi module*1   | USB Serial ESP-01S WIFI Expansion Module *1  | 18650 Battery*1   |
|  |  |  |  |  |  |

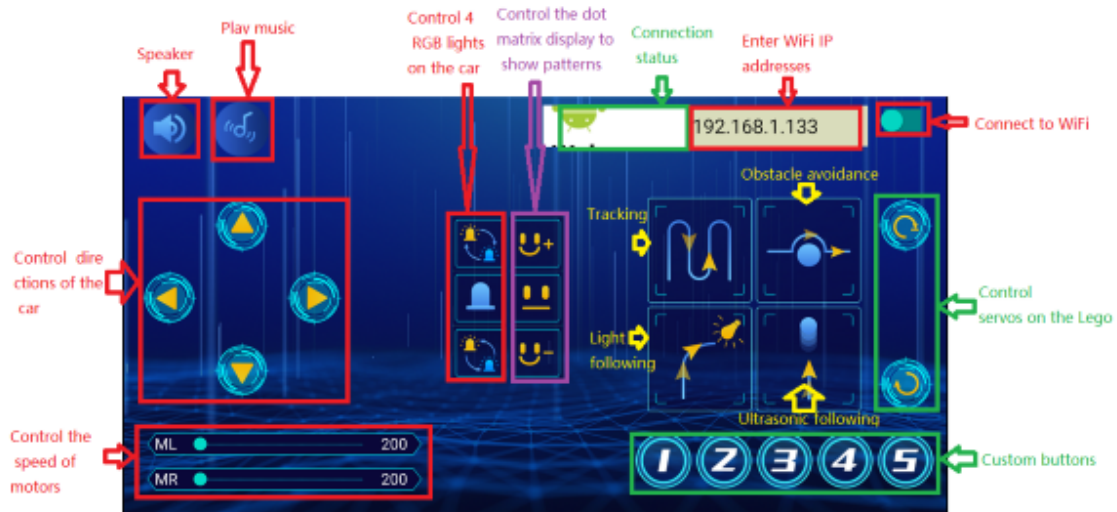
#### (3)Test Code

The code of ESP8266wifi module is not changed, then change the wifi password of the code into yours.

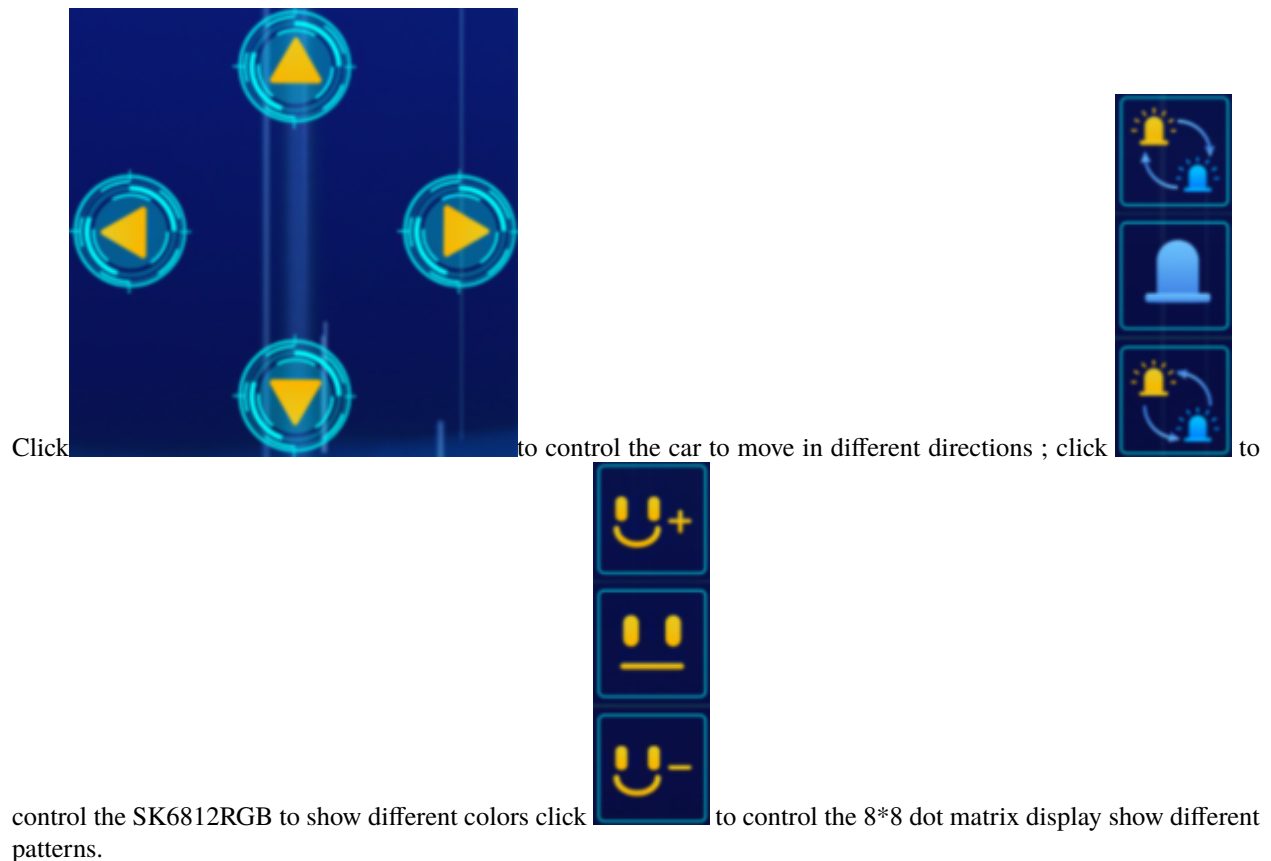




## (4)APP operation, as shown below:



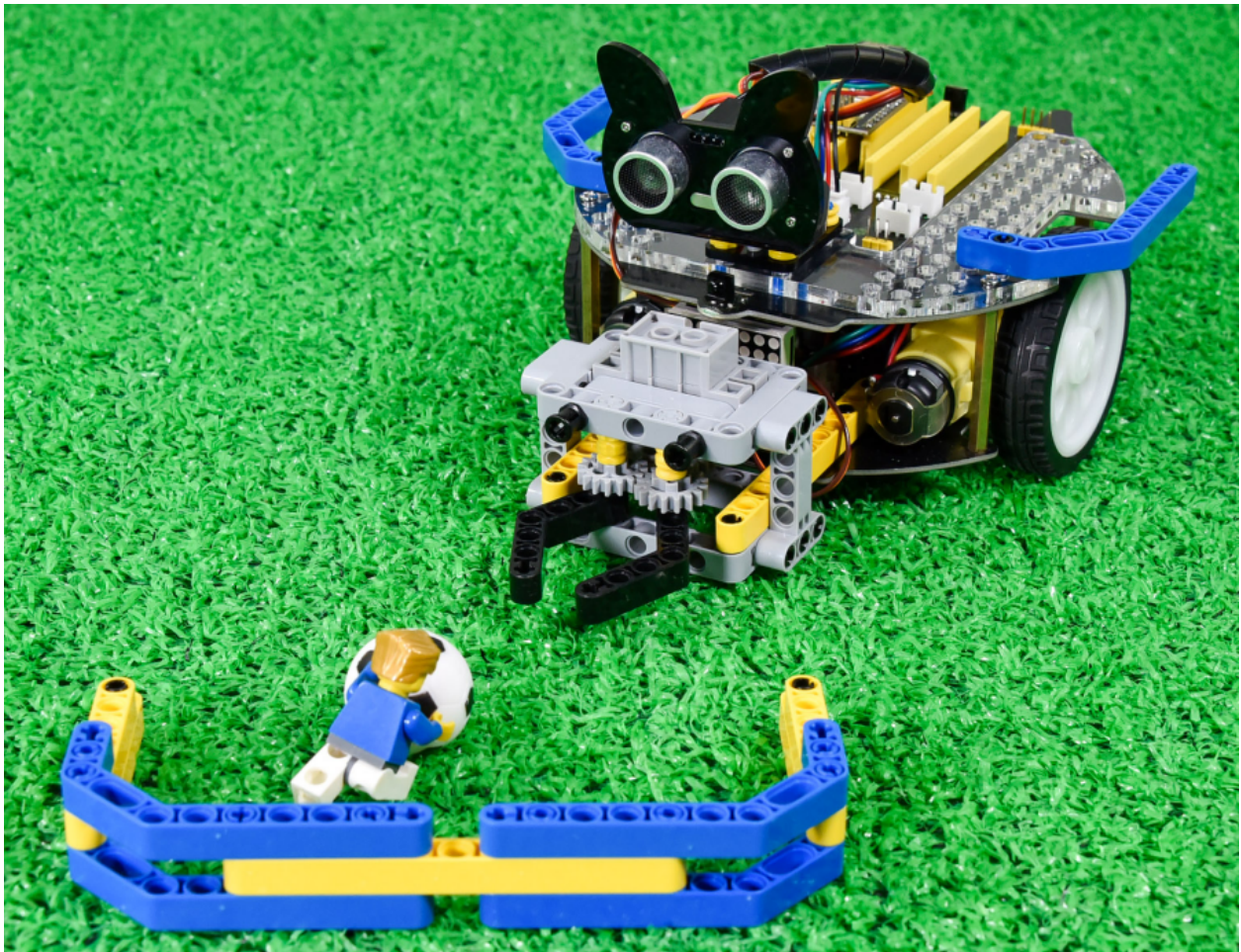
Note: See the previous lesson of Project 11.1 for how to connect your app to WiFi







## SOCCER TUTORIAL





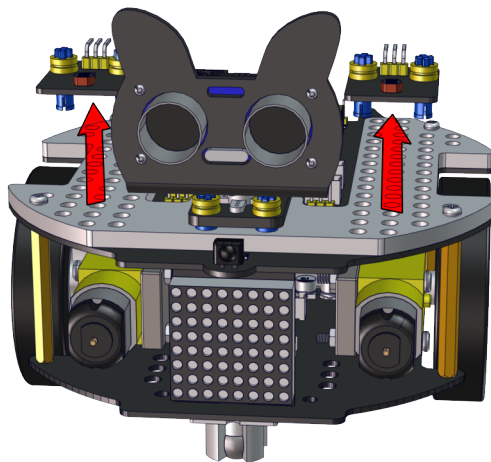
## 10.1 1. Description

Can you imagine that a robot can play soccer? This idea has become realistic. As we know, the RoboCup championship is generally held each year. In this part, we will create a soccer robot to play soccer.

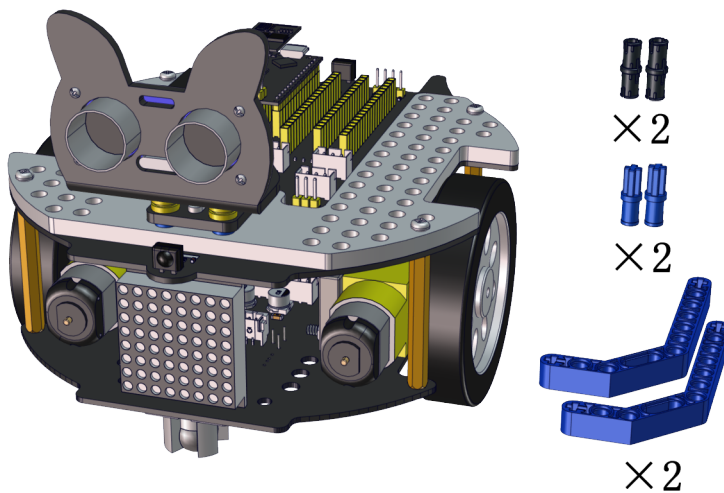
## 10.2 2. How to install the soccer robot:

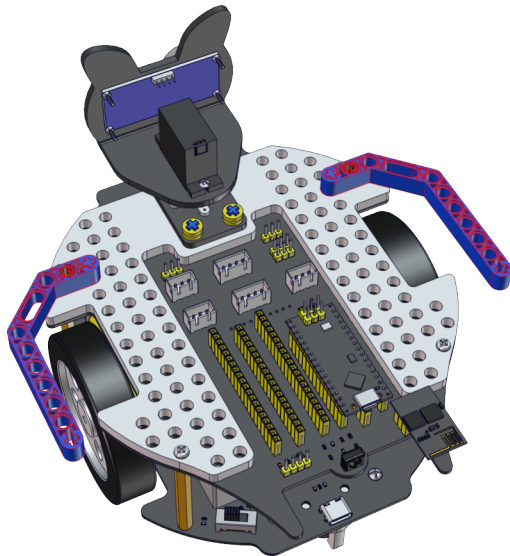
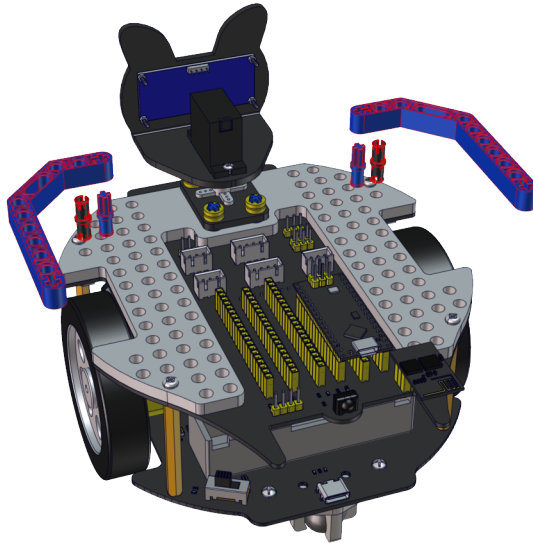
### Step 1:

Remove two photoresistors first

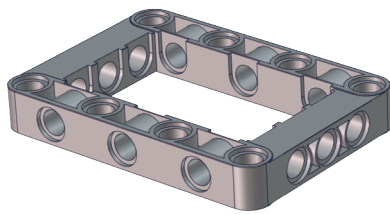


### Step 2:





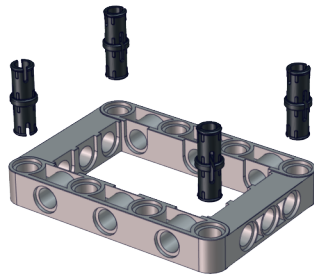
**Step 3:**



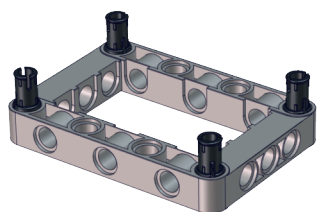
×1



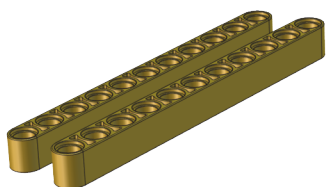
×4







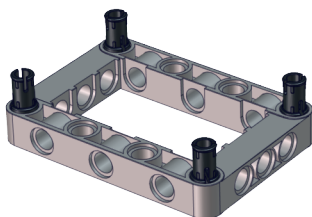
Step 4:

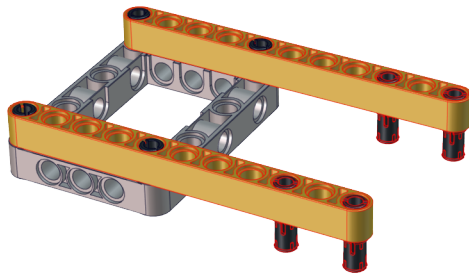
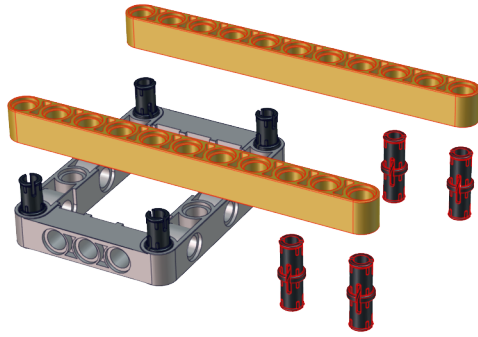


×2

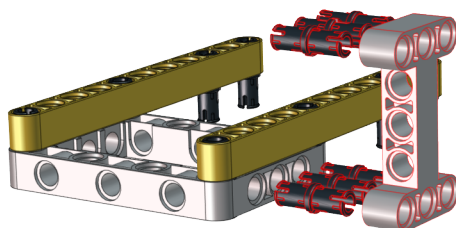
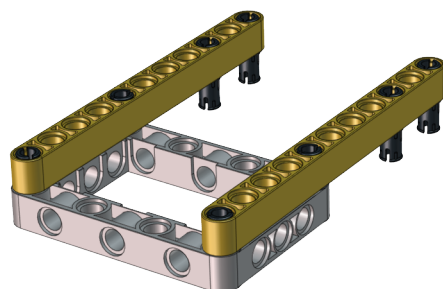
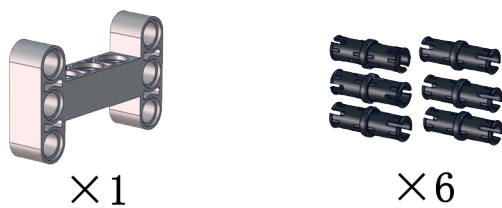


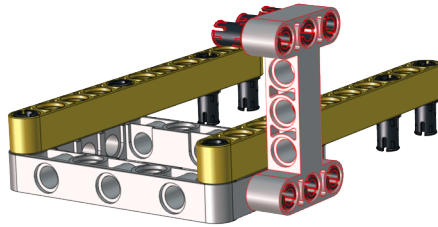
×4



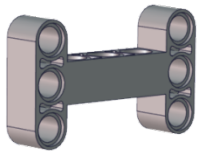


**Step 5:**

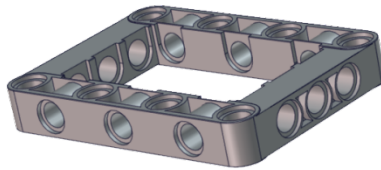




Step 6:



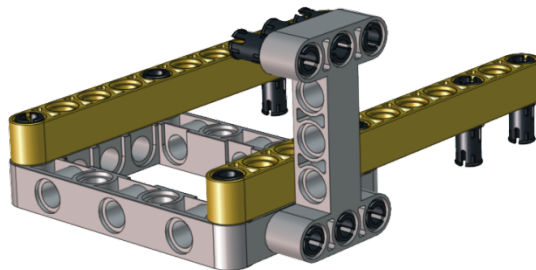
×1

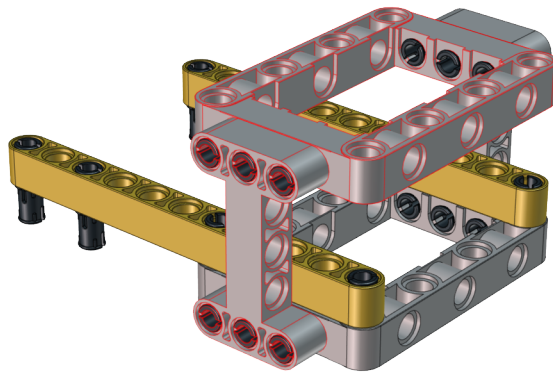
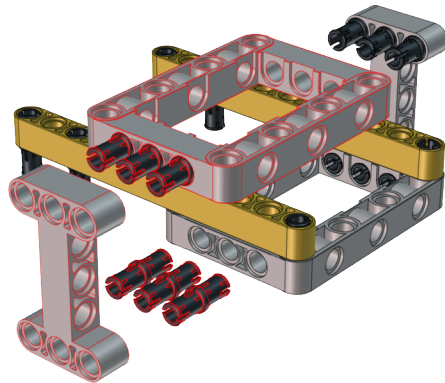


×1

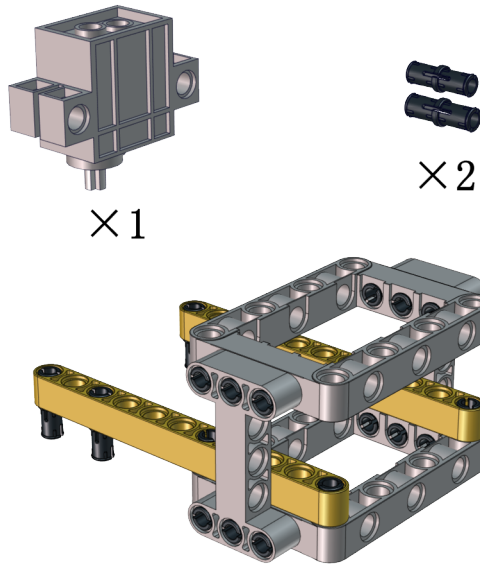


×6

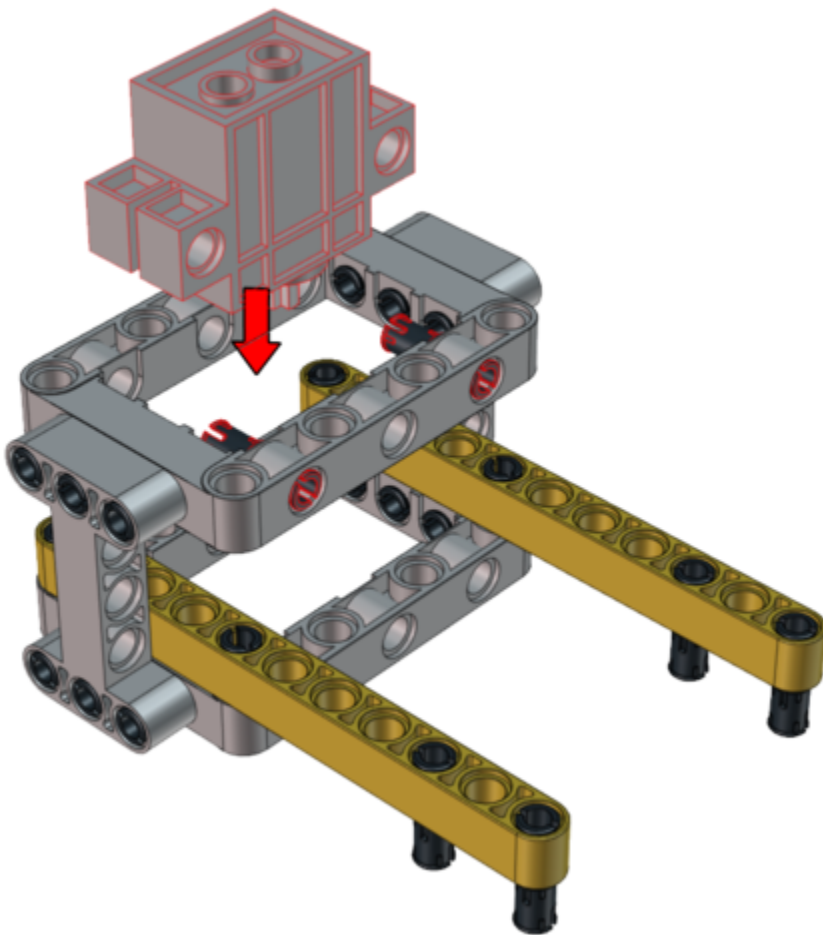


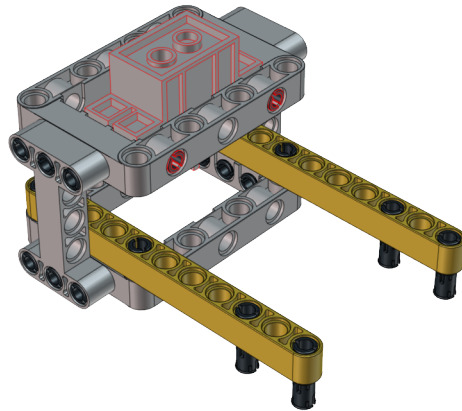


**Step 7:**

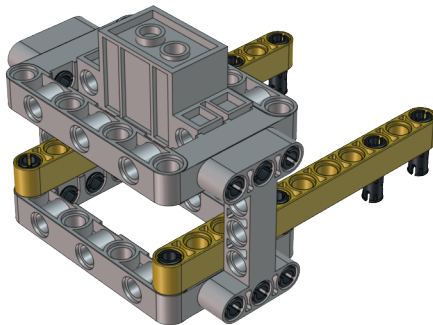


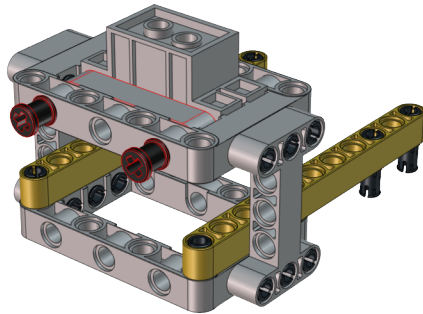
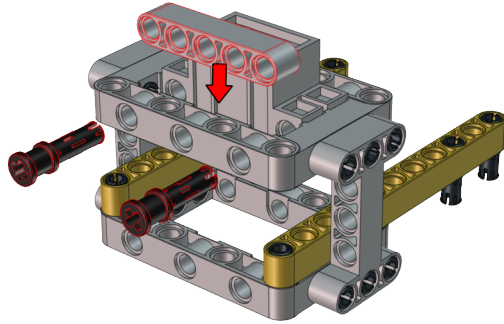
Note the installation direction of the part marked by the red circle





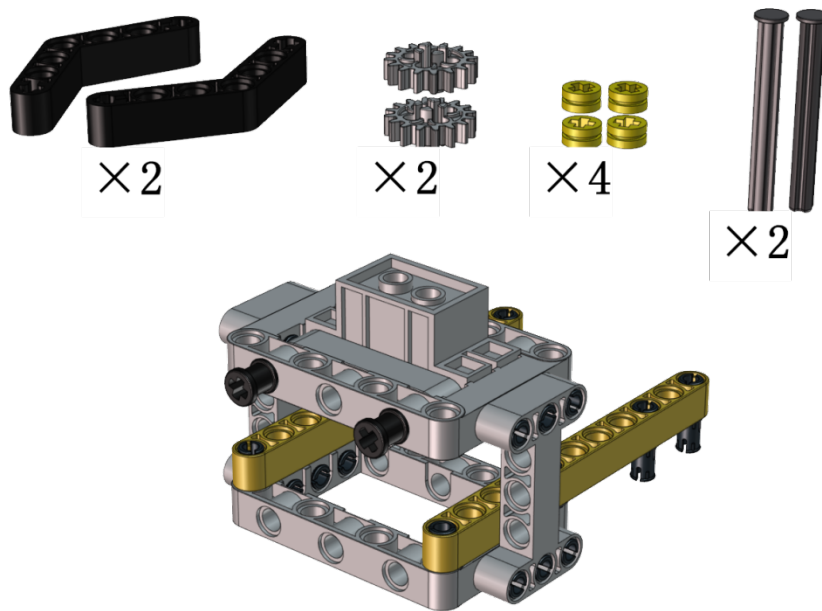
Step 8:



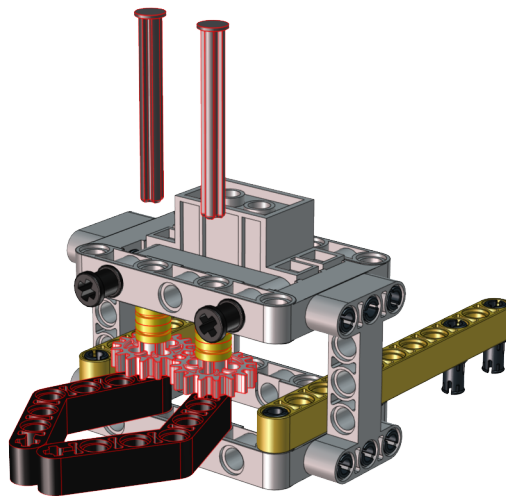


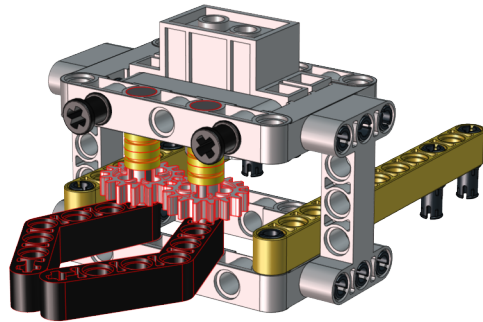
**Step 9:**





Adjust the angle of the claw. Then make it close and face front.



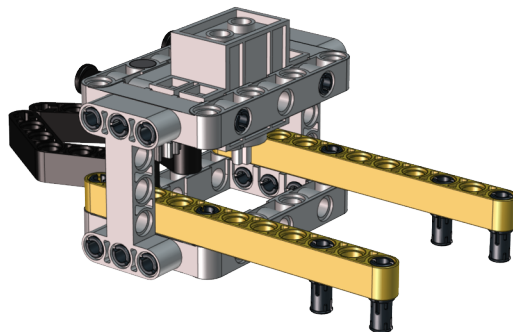


**Step 10:**

Set the angle of the servo to 180 degree



× 1



## Wire Up

| Servo  | PCB Board |
|--------|-----------|
| Brown  | G         |
| Red    | 5V        |
| Orange | S2 (A0)   |

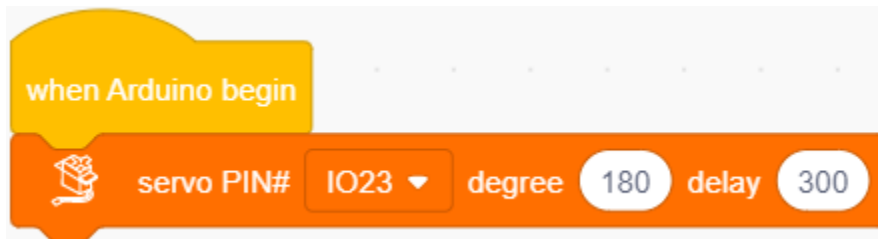
Upload the code of the servo to the main board of the Beetlebot car, as shown below

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo

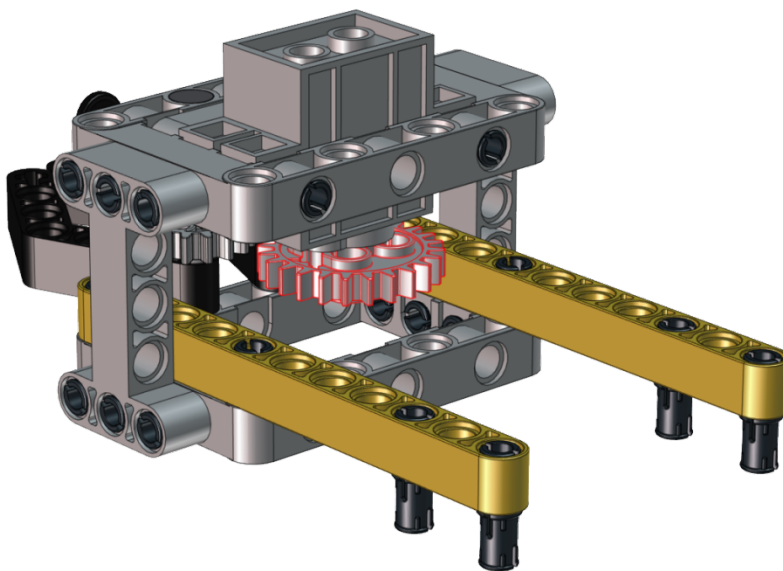
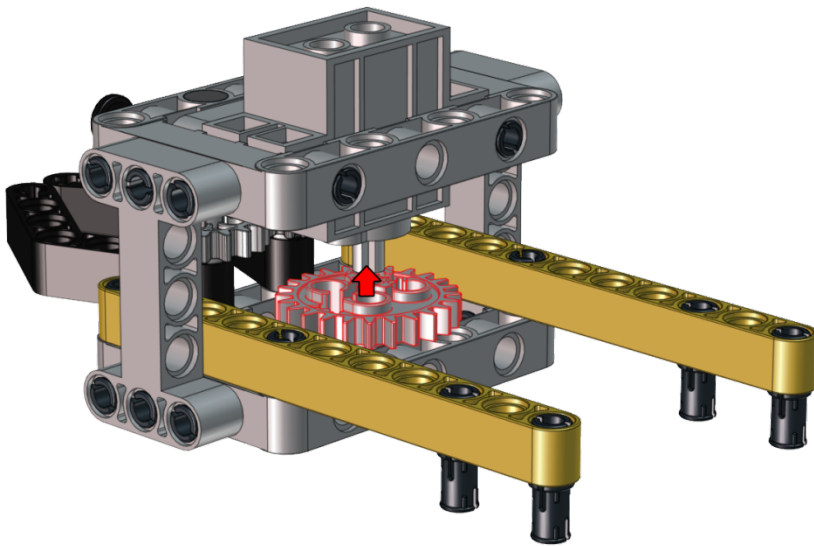
void setup() {
  myservo.attach(A0); // attaches the servo on pin A0 to the servo object
}

void loop() {
  myservo.write(180); // tell servo to go to position
}
```

You can also initialize the angle of the servo through the following code

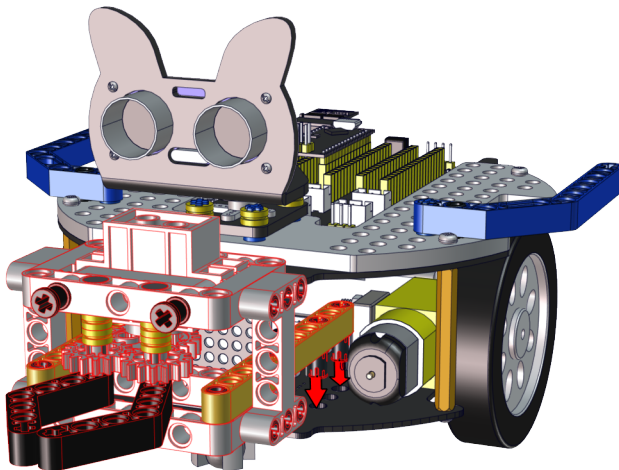
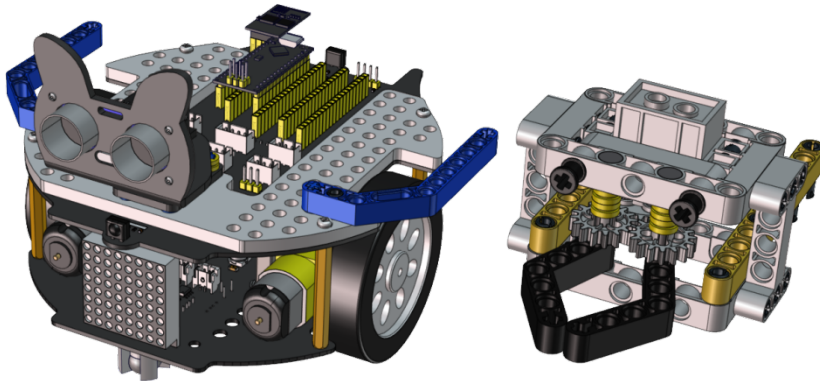


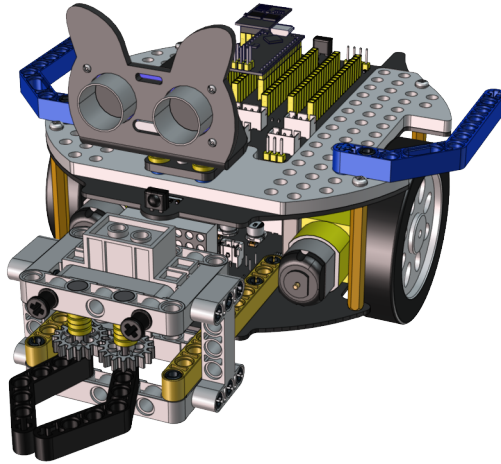
Keep the claw close and face front before installing the gear



**Step 11:**

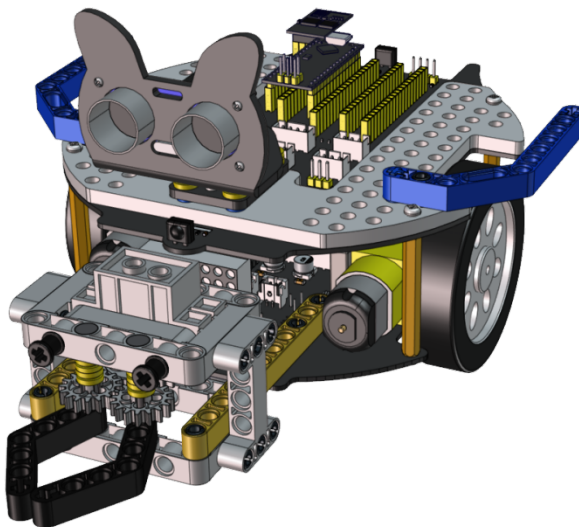
Required Parts





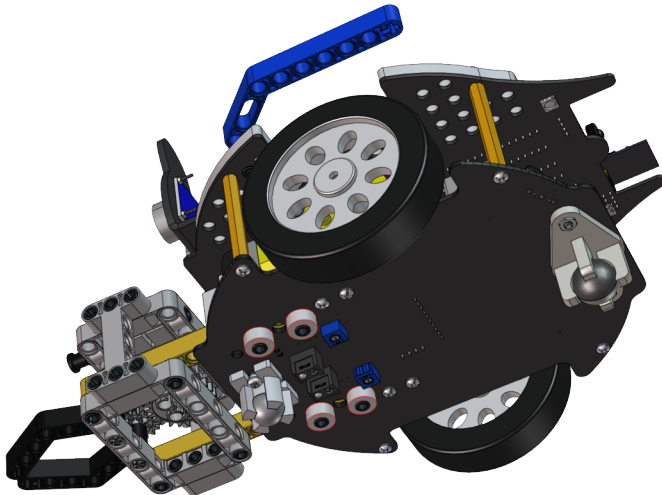
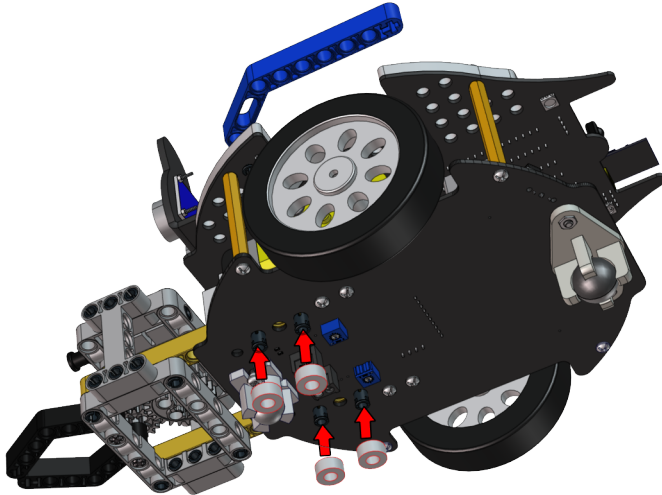
**Step 12:**

Required Parts

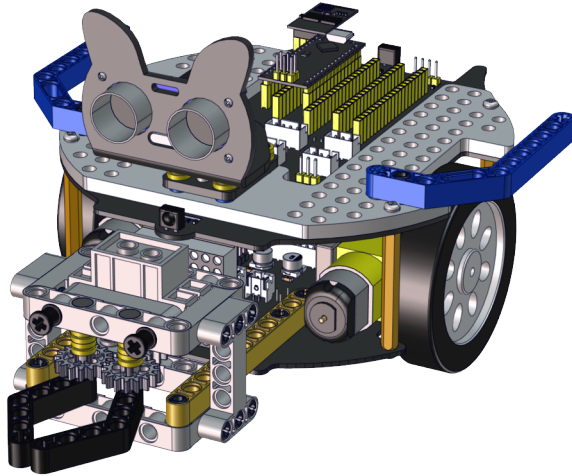


Acrylic Board

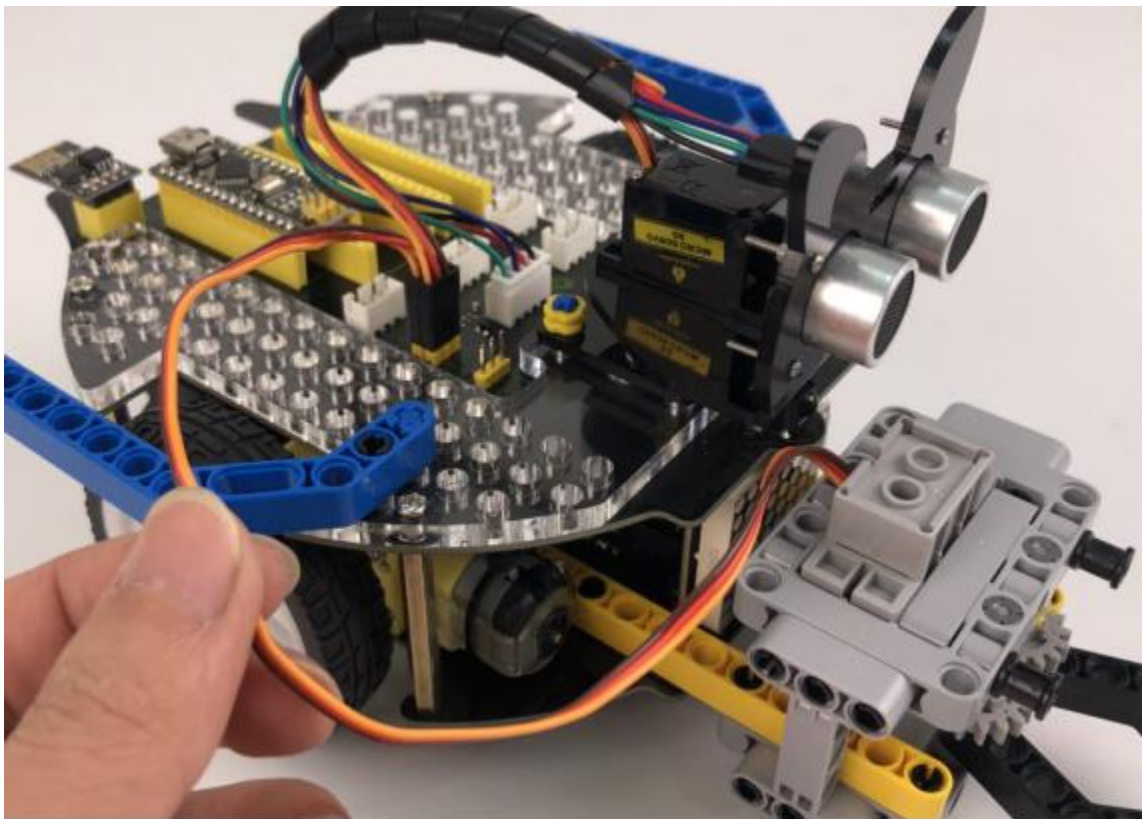
×4







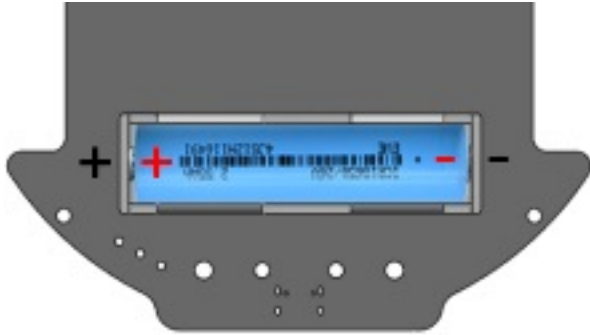
**Step 13:**



---

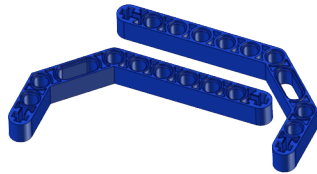
We adopt a model 18650 lithium battery with a pointed positive pole, whose power and capacity are not required.



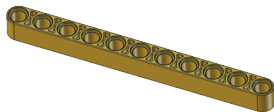


### 10.3 3. Install a soccer goal

Step 1:



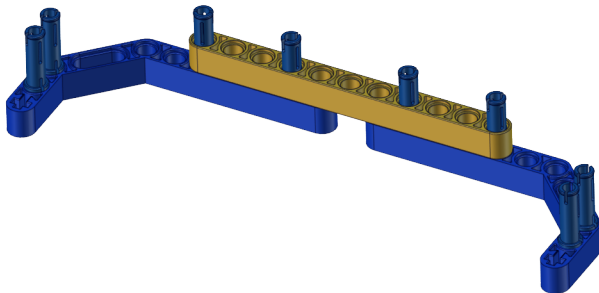
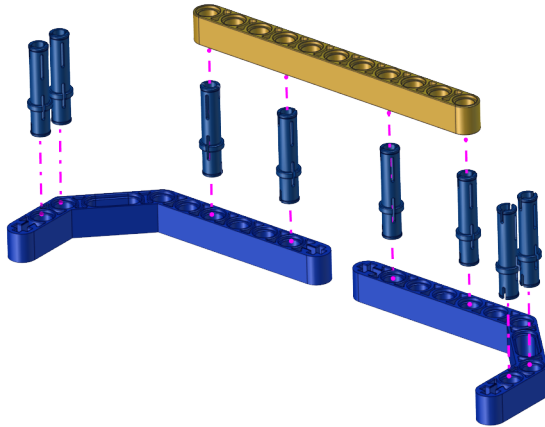
×2



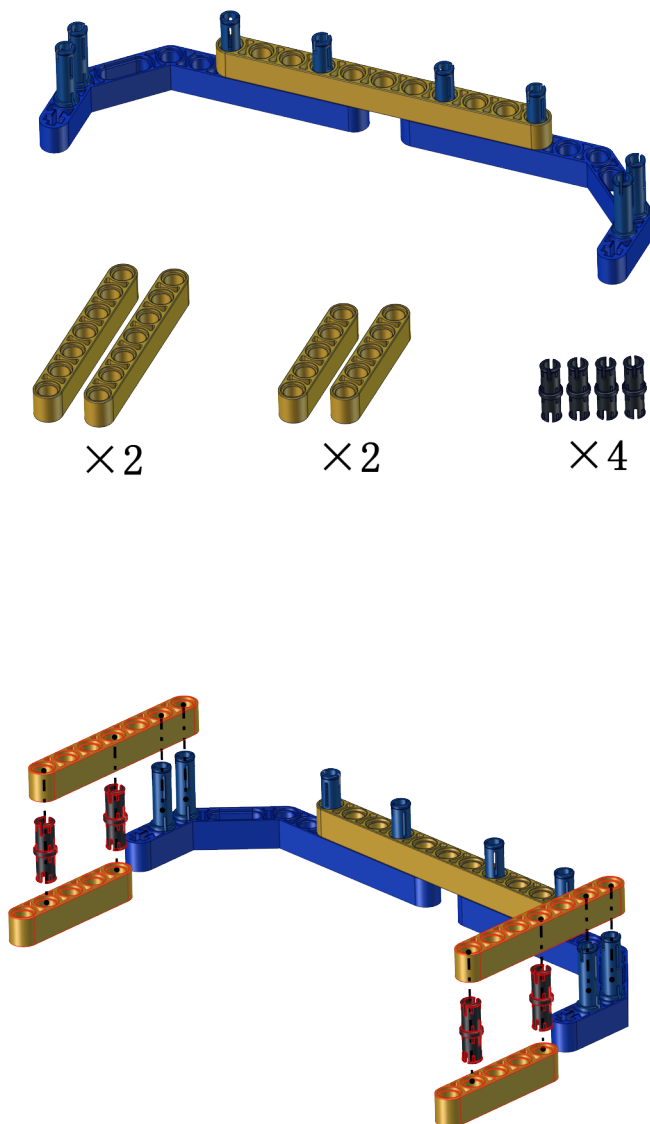
×1

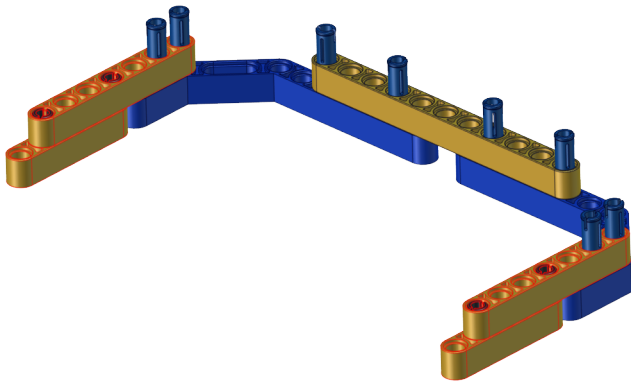


×8

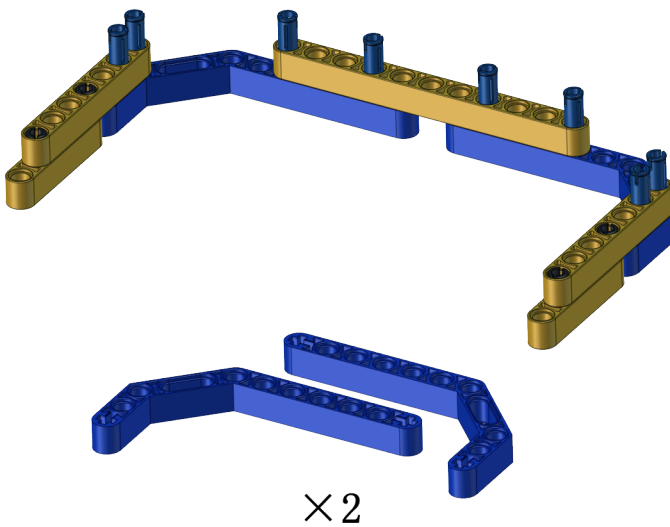


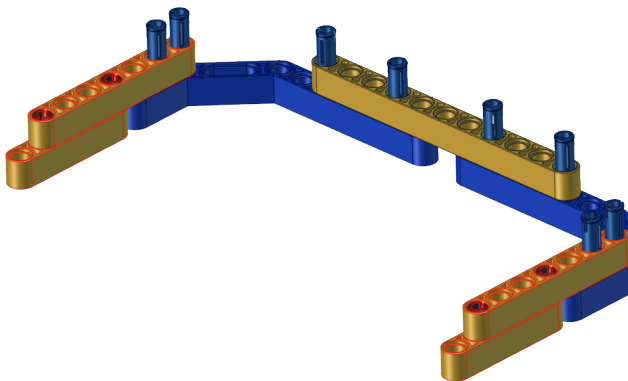
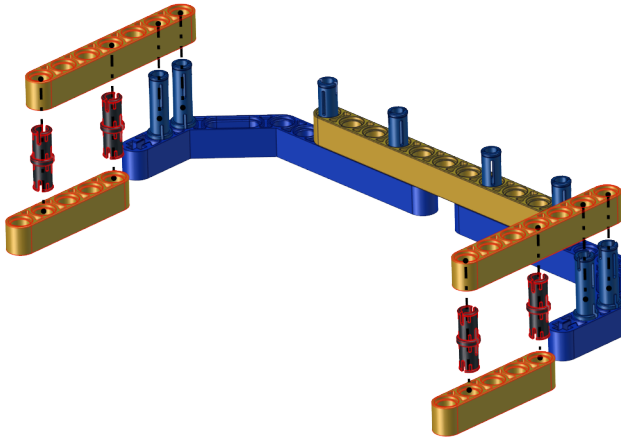
**Step 2:**





Step 3:





## 10.4 4. Codes:

### 10.4.1 (1)Arduino Code

```
#include <Servo.h>
Servo lgservo;
#define ML 4
#define ML_PWM 6
#define MR 2
#define MR_PWM 5
#define servo2 A0

char val;
```

(continues on next page)

(continued from previous page)

```
char wifiData;

void setup() {
  Serial.begin(9600);
  pinMode(ML, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  lgservo.attach(A0);
  lgservo.write(180);
  delay(1000);
  lgservo.write(160);
}

void loop() {
  if(Serial.available() > 0)
  {
    val = Serial.read();
    Serial.print(val);
  }
  switch(val)
  {
    case 'F': car_forward(); break;
    case 'B': car_back(); break;
    case 'L': car_left(); break;
    case 'R': car_right(); break;
    case 'S': car_stop(); break;
    case 'p': lgservo.write(180); break;
    case 'x': lgservo.write(160); break;
  }
}

void car_forward()
{
  digitalWrite(ML, LOW);
  analogWrite(ML_PWM, 127);
  digitalWrite(MR, LOW);
  analogWrite(MR_PWM, 127);
}

void car_back()
{
  digitalWrite(ML, HIGH);
  analogWrite(ML_PWM, 127);
  digitalWrite(MR, HIGH);
  analogWrite(MR_PWM, 127);
}

void car_left()
{

```

(continues on next page)

(continued from previous page)

```
digitalWrite(ML,HIGH);  
analogWrite(ML_PWM,150);  
digitalWrite(MR,LOW);  
analogWrite(MR_PWM,105);  
}  
  
void car_right()  
{  
  digitalWrite(ML,LOW);  
  analogWrite(ML_PWM,105);  
  digitalWrite(MR,HIGH);  
  analogWrite(MR_PWM,150);  
}  
  
void car_stop()  
{  
  digitalWrite(ML,LOW);  
  analogWrite(ML_PWM,0);  
  digitalWrite(MR,LOW);  
  analogWrite(MR_PWM,0);  
}
```





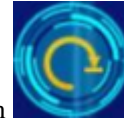
## 10.4.2 (2) Kidsblock Code



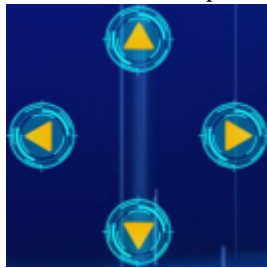
## 10.5 5. Test Result

Note: Please refer to the Project 11.2 of the Arduino tutorial for downloading and operating the APP.

Build up the soccer goal with building blocks and place it at fixed location, connect the robot car through Wifi.



Put a small soccer in the middle of the claw of the robot car, press and hold down the button to enable the



claw to hold the soccer, then press buttons to adjust the car's movement direction so as to put



the soccer close to the soccer goal. At last, release the button to allow the soccer to drop on the floor and roll to the soccer goal. If not, repeat the above step to shoot the goal.

If your friend owns this kind of soccer robot, you guys can hold a soccer match. It sounds amazing, right?

## CATAPUL TUTORIAL



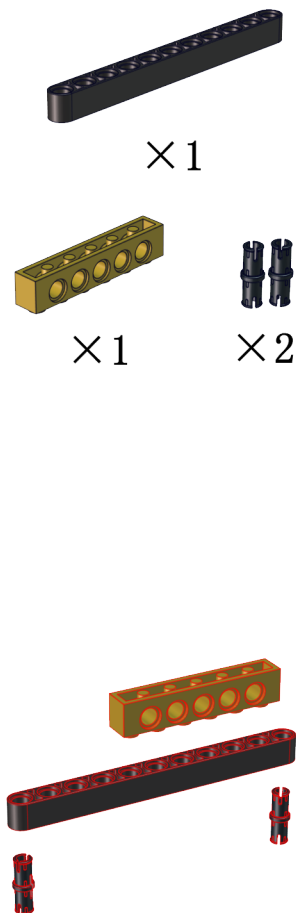
## 11.1 1. Description

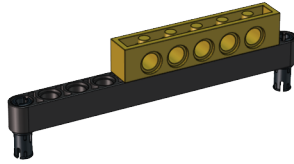
A catapult is a ballistic device used to launch a projectile a great distance without the aid of gunpowder or other propellants – particularly various types of ancient and medieval siege engines. ... We will make a catapult with LEGO building blocks. Equipped with servos and gears, the car has LEGO tower used to carry projectiles.

As the servo rotates to a proper angle then push the long arm backward a projectile will be launched.

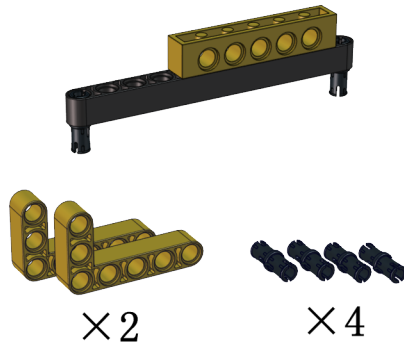
## 11.2 2. How to build up a catapult

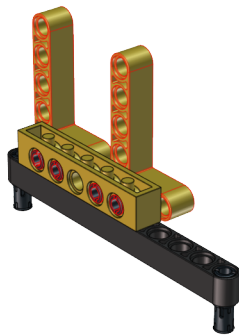
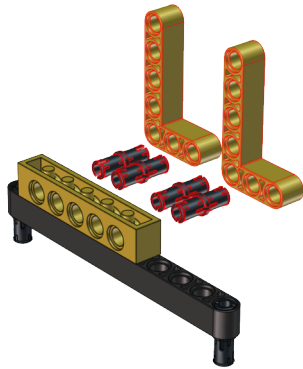
Step 1:



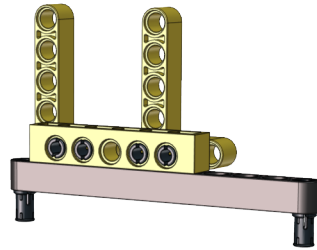


Step 2:





**Step 3:**



×1



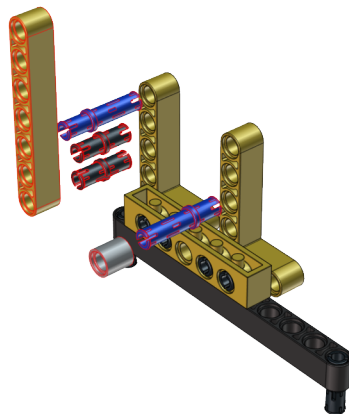
×2

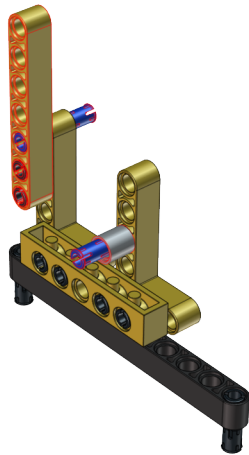


×2

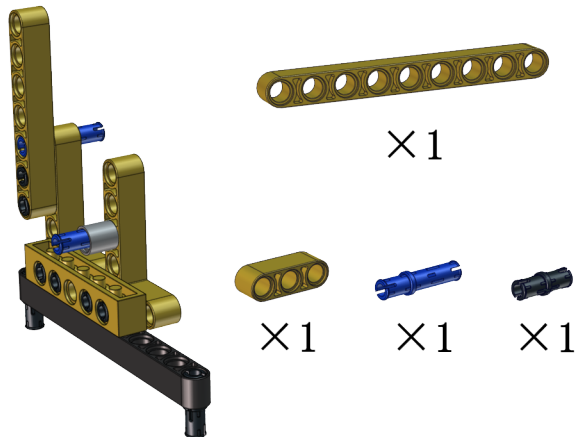


×1

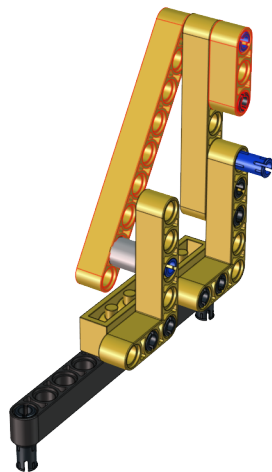
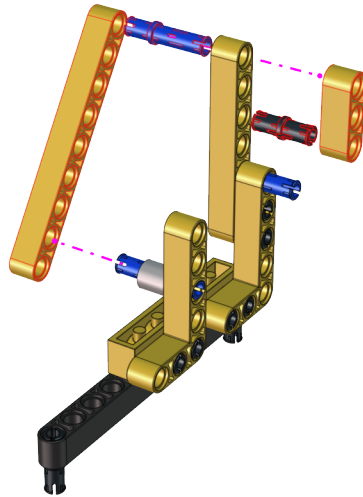




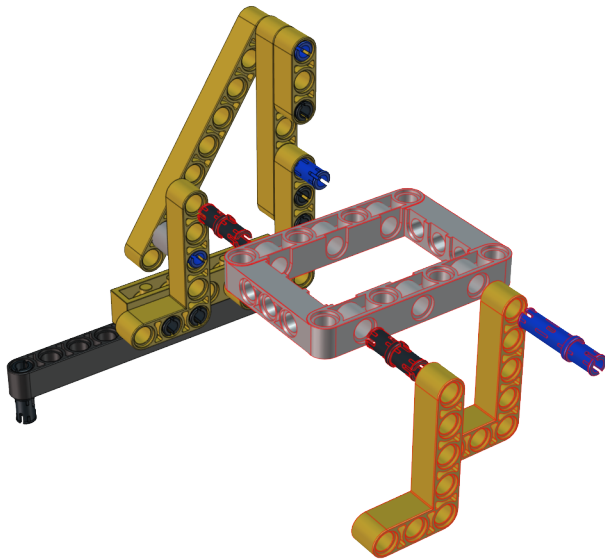
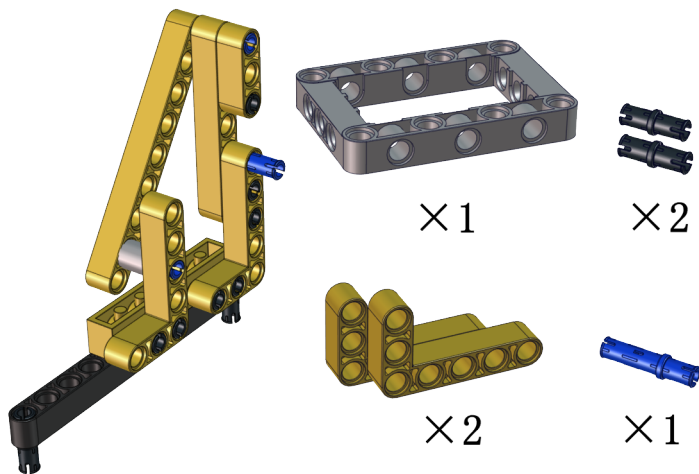
**Step 4:**

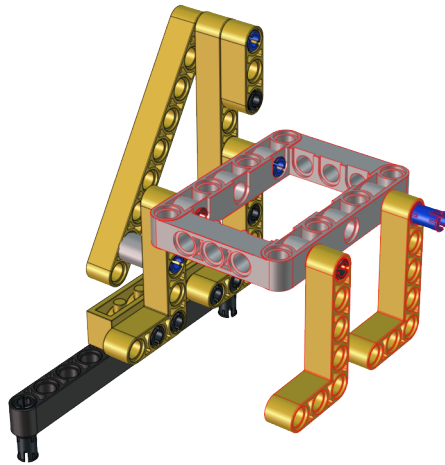




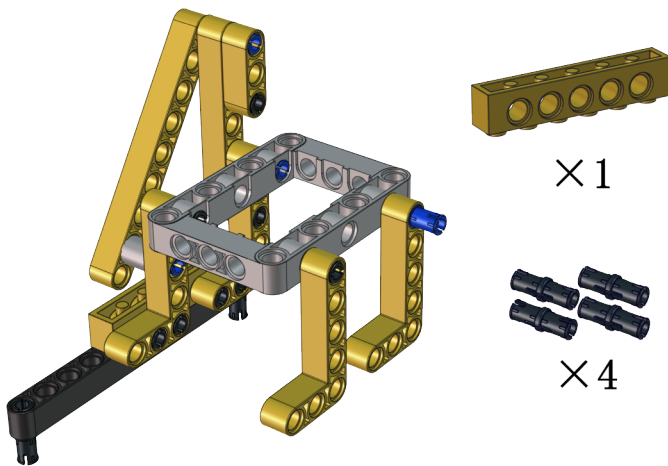


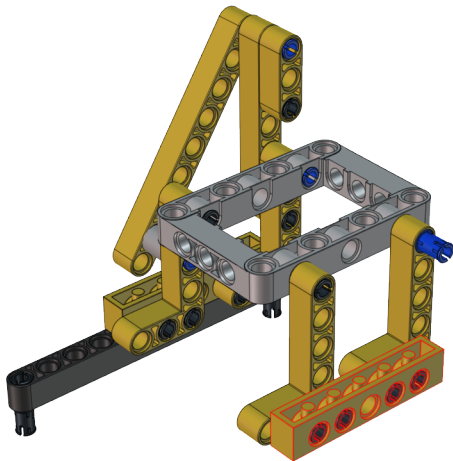
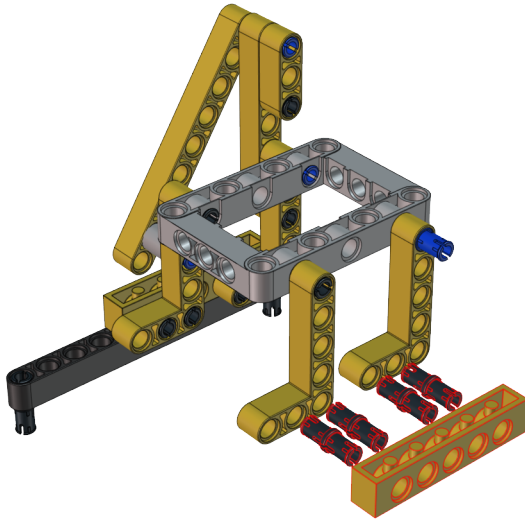
**Step 5:**



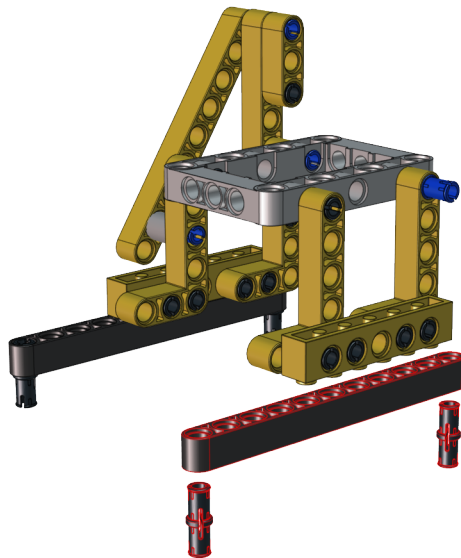
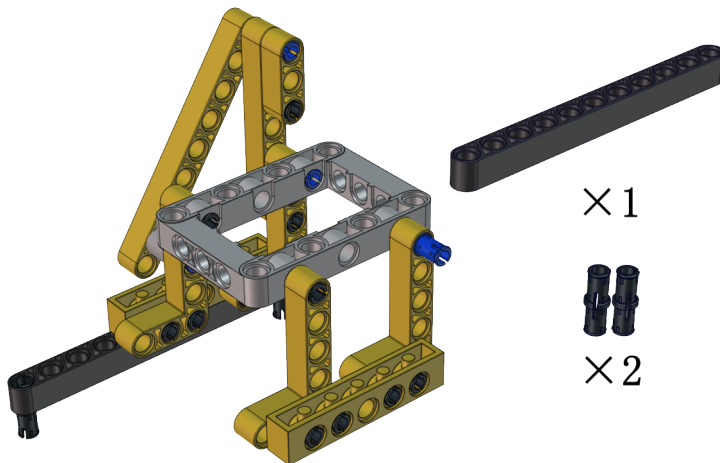


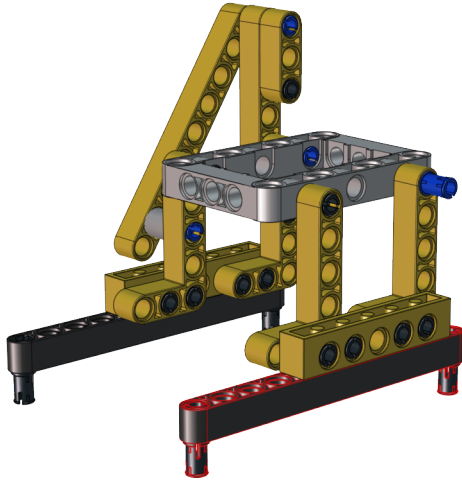
Step 6:



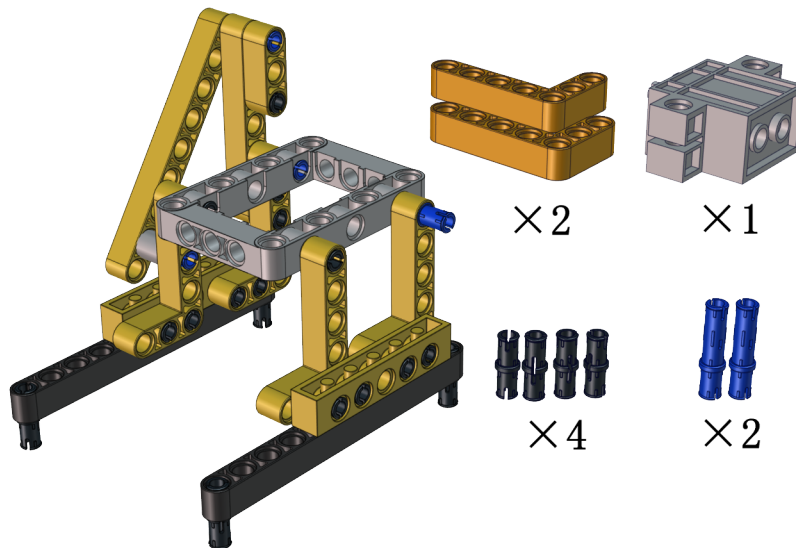


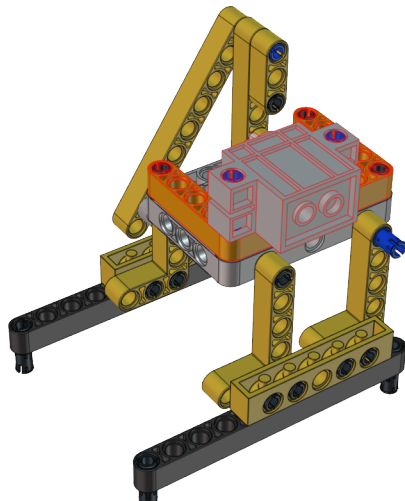
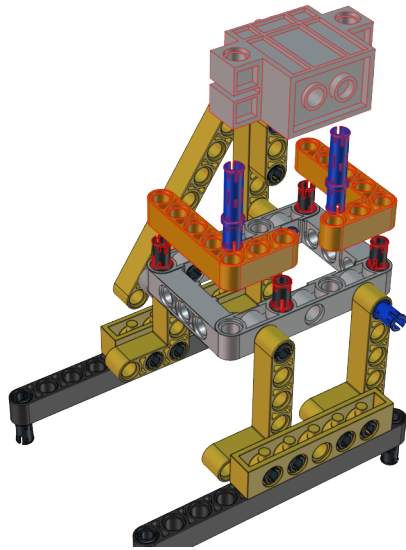
**Step 7:**



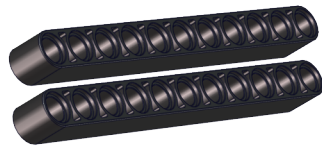


Step 8:

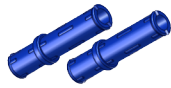




**Step 9:**



×2



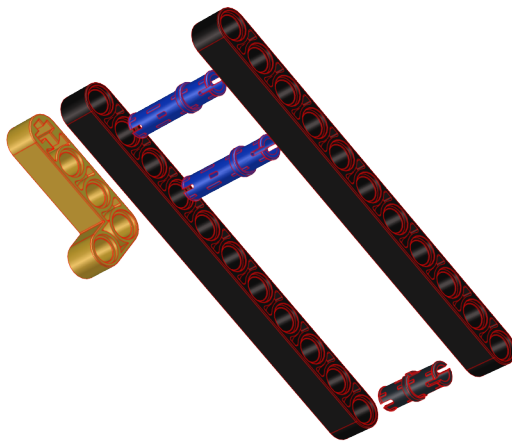
×2



×1



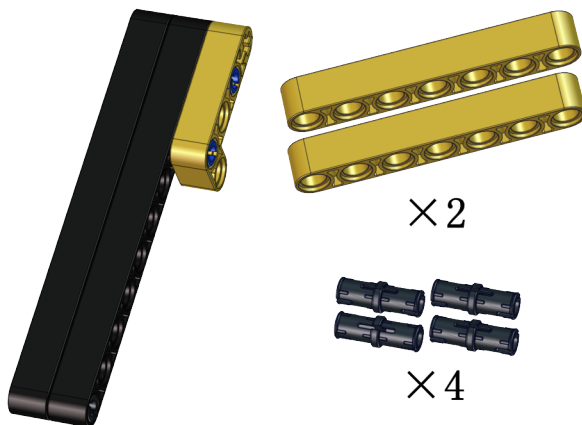
×1

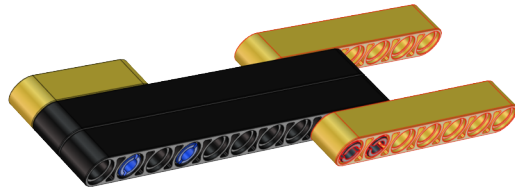
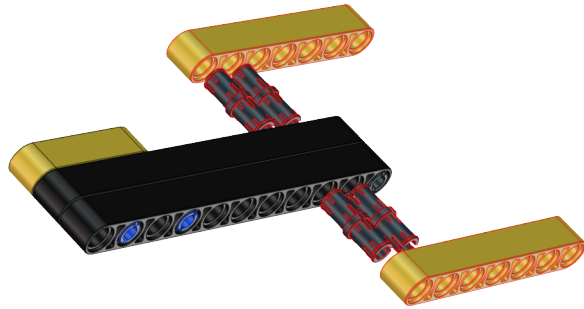




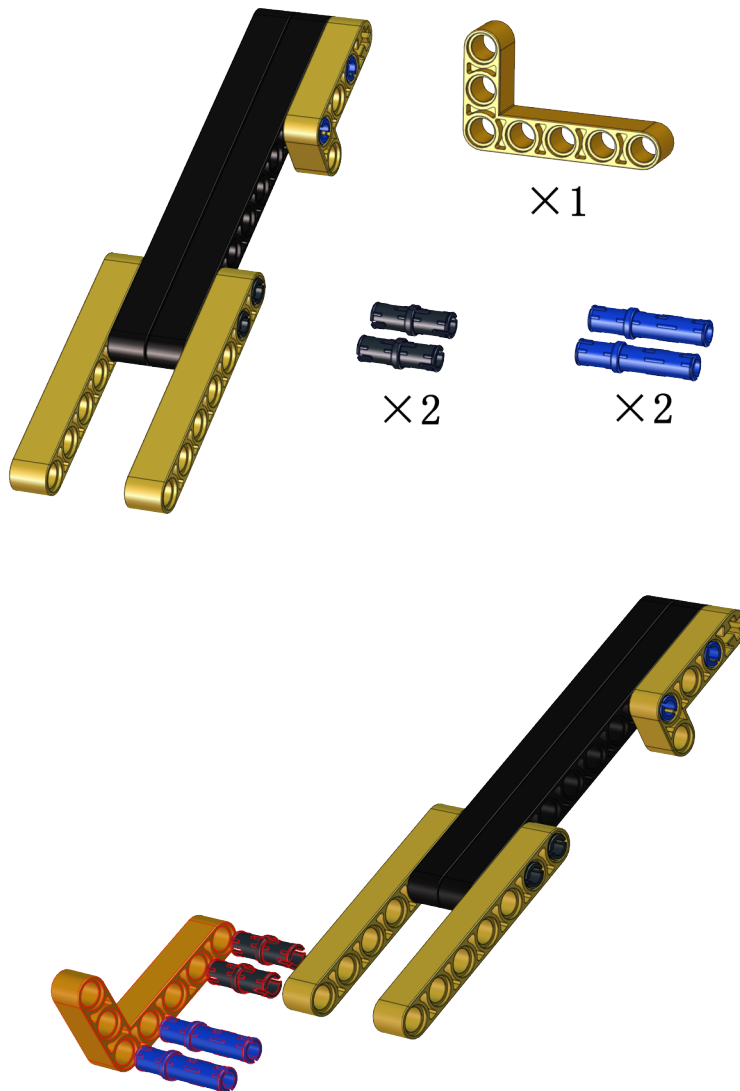


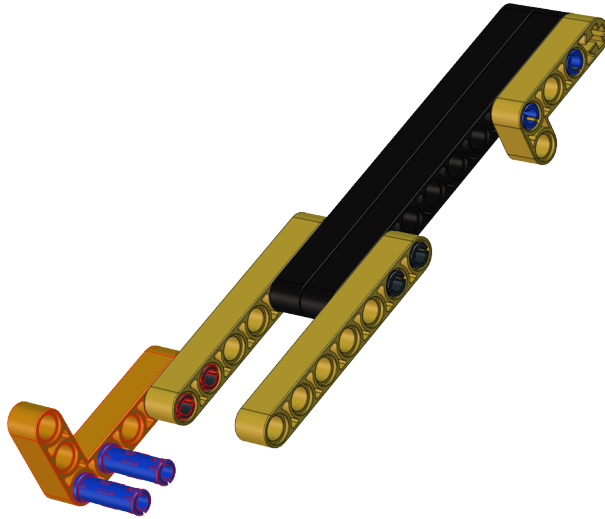
Step 10:



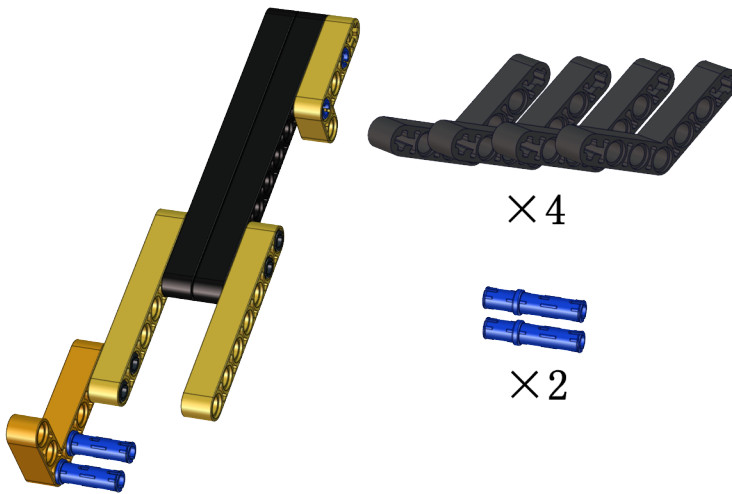


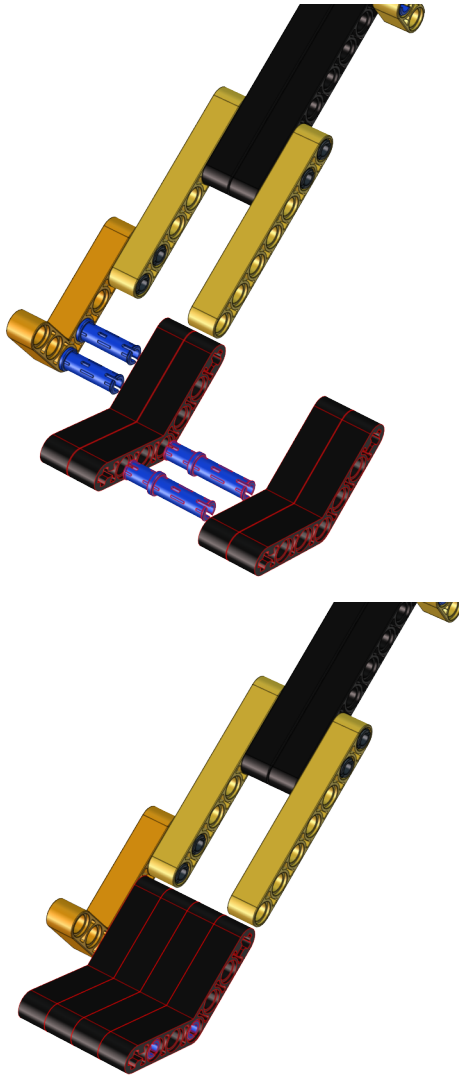
**Step 11:**



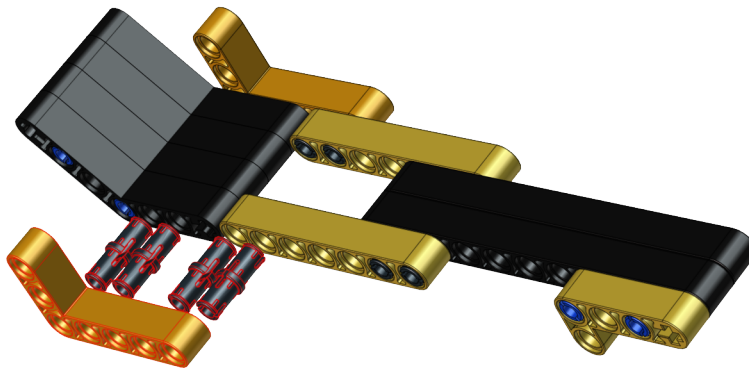
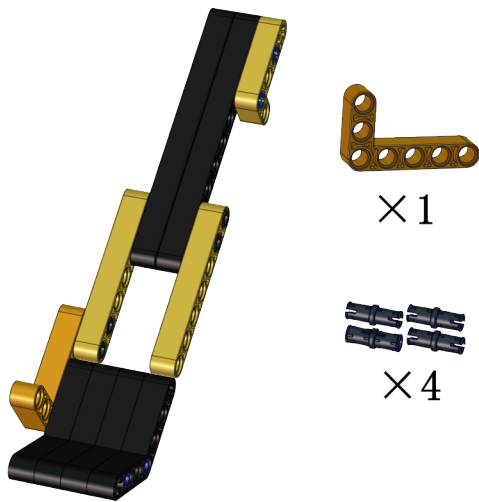


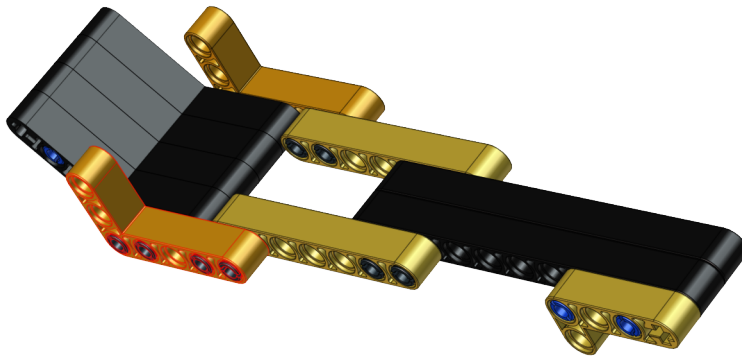
Step 12:



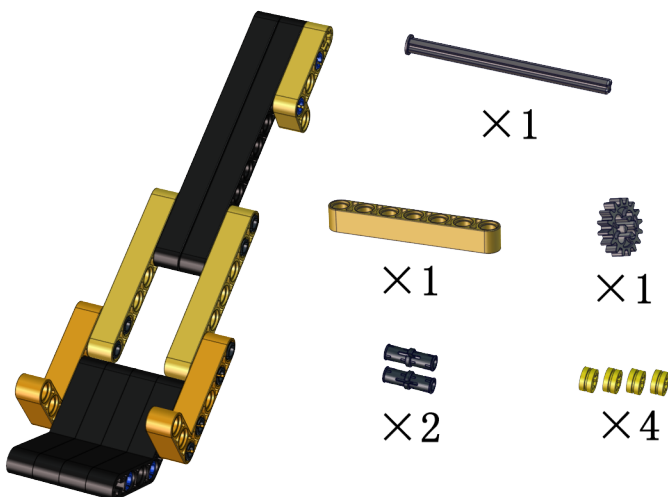


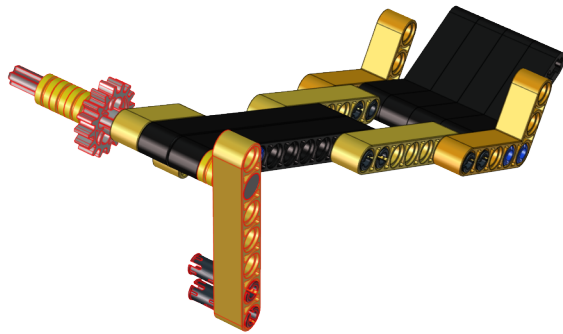
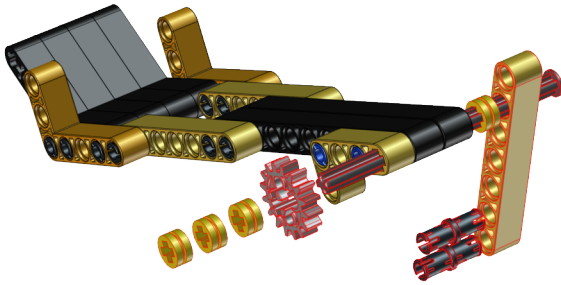
**Step 13:**





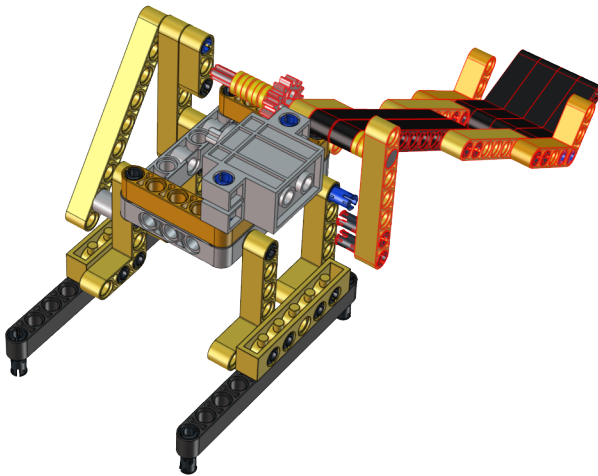
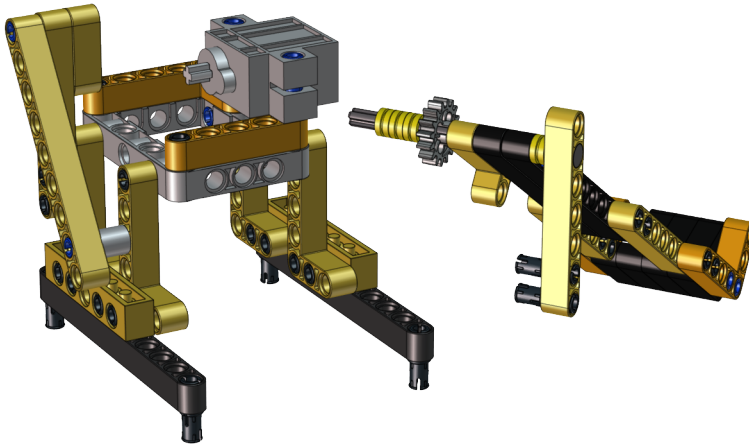
**Step 14:**

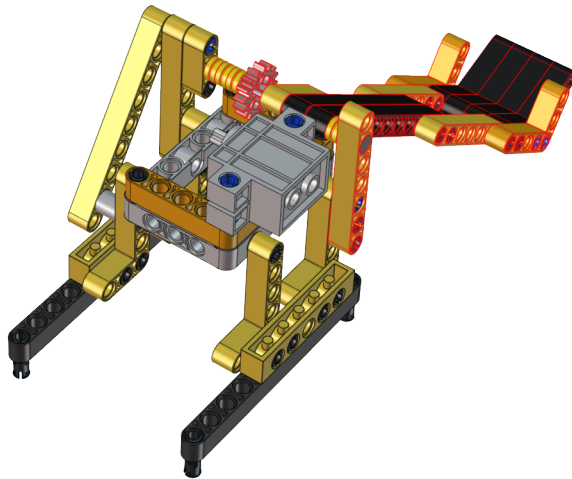




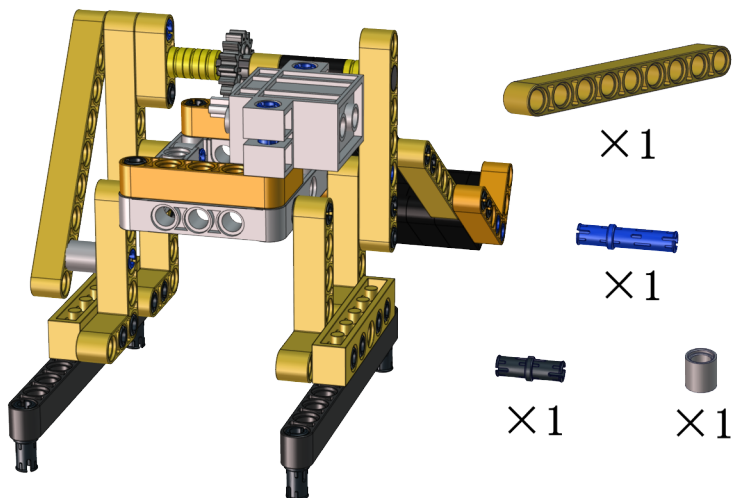
**Step 15:**

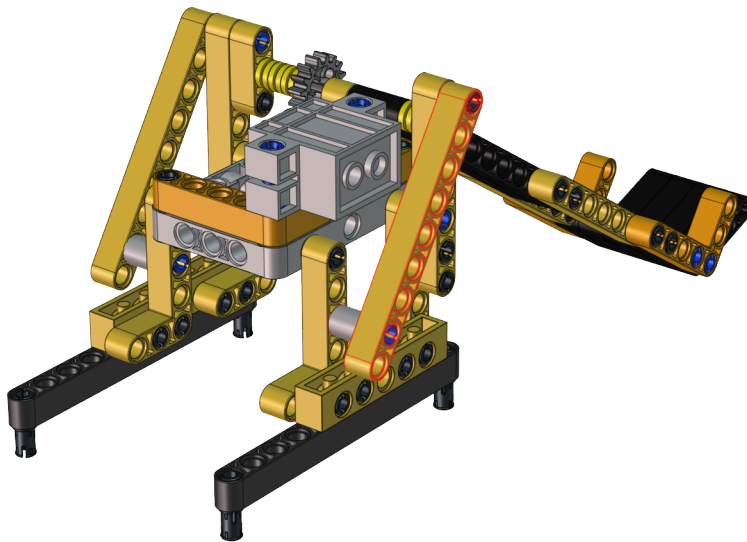
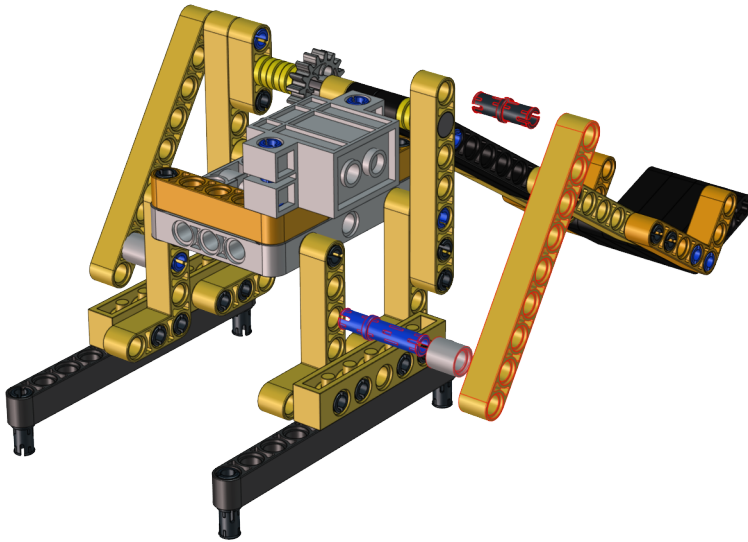




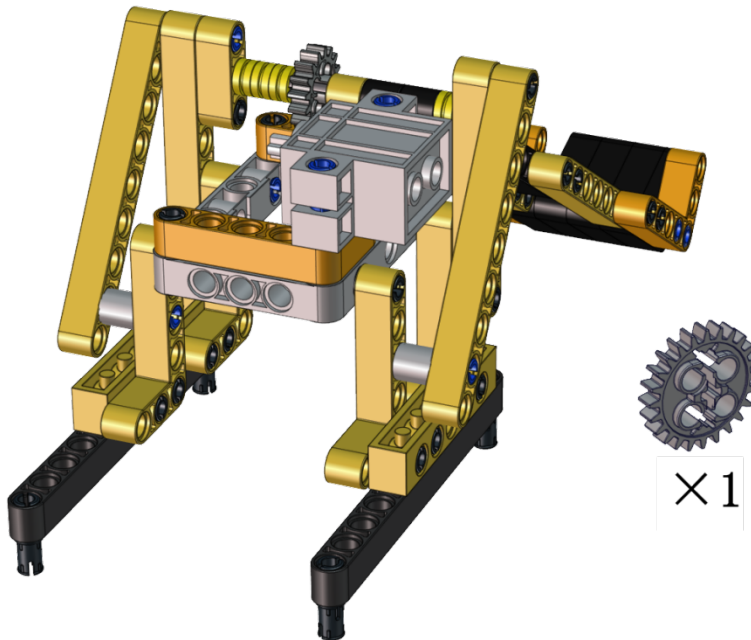


Step 16:



**Step 17:**

Set the angle of the servo to 0 degree



## Wire servo up

|        |           |
|--------|-----------|
| Servo  | PCB Board |
| Brown  | G         |
| Red    | 5V        |
| Orange | S2 (A0)   |

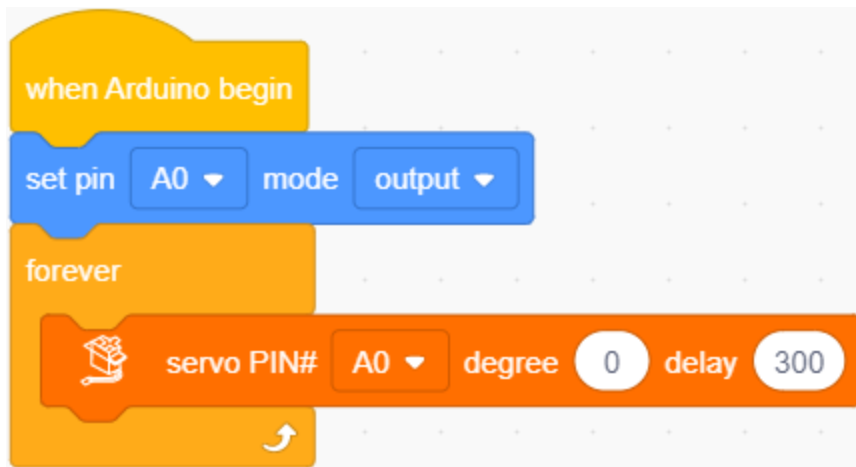
Upload the code of the servo to the main board of the Beetlebot car, as shown below

```
#include <Servo.h>
Servo myservo; // create servo object to control a servo

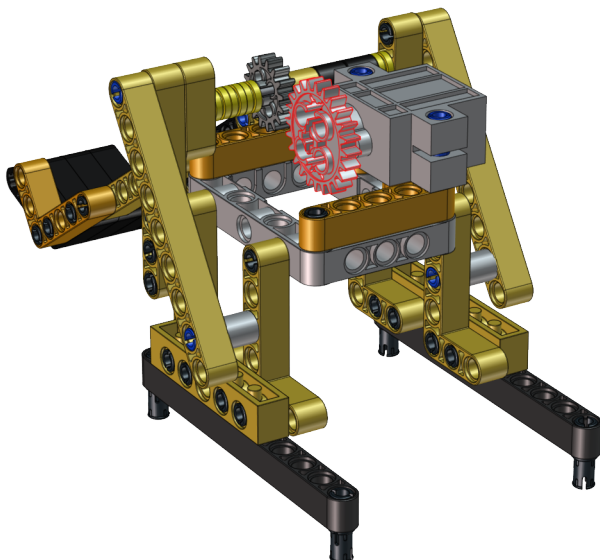
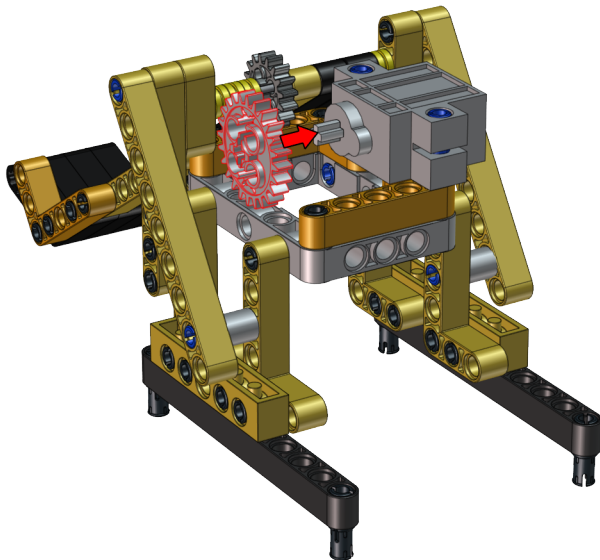
void setup() {
  myservo.attach(A0); // attaches the servo on pin A0 to the servo object
}

void loop() {
  myservo.write(0); // tell servo to go to position
}
```

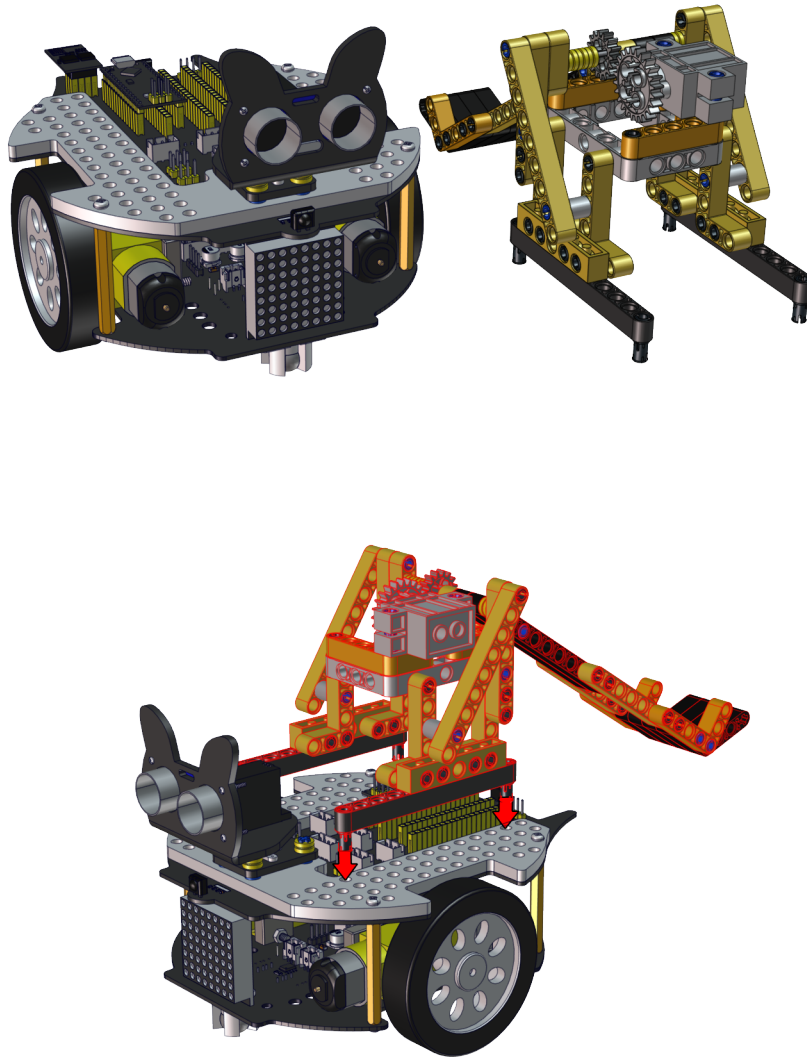
You can also initialize the angle of the servo through the following code

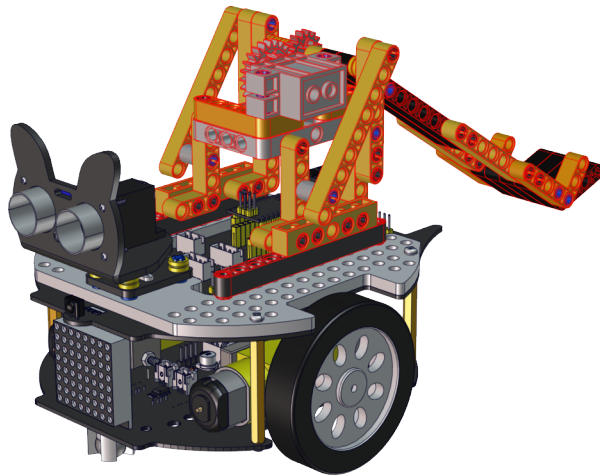


Check the Scratch-KidsBlock code as followsthen upload the code to the main board of the Beetlebot car

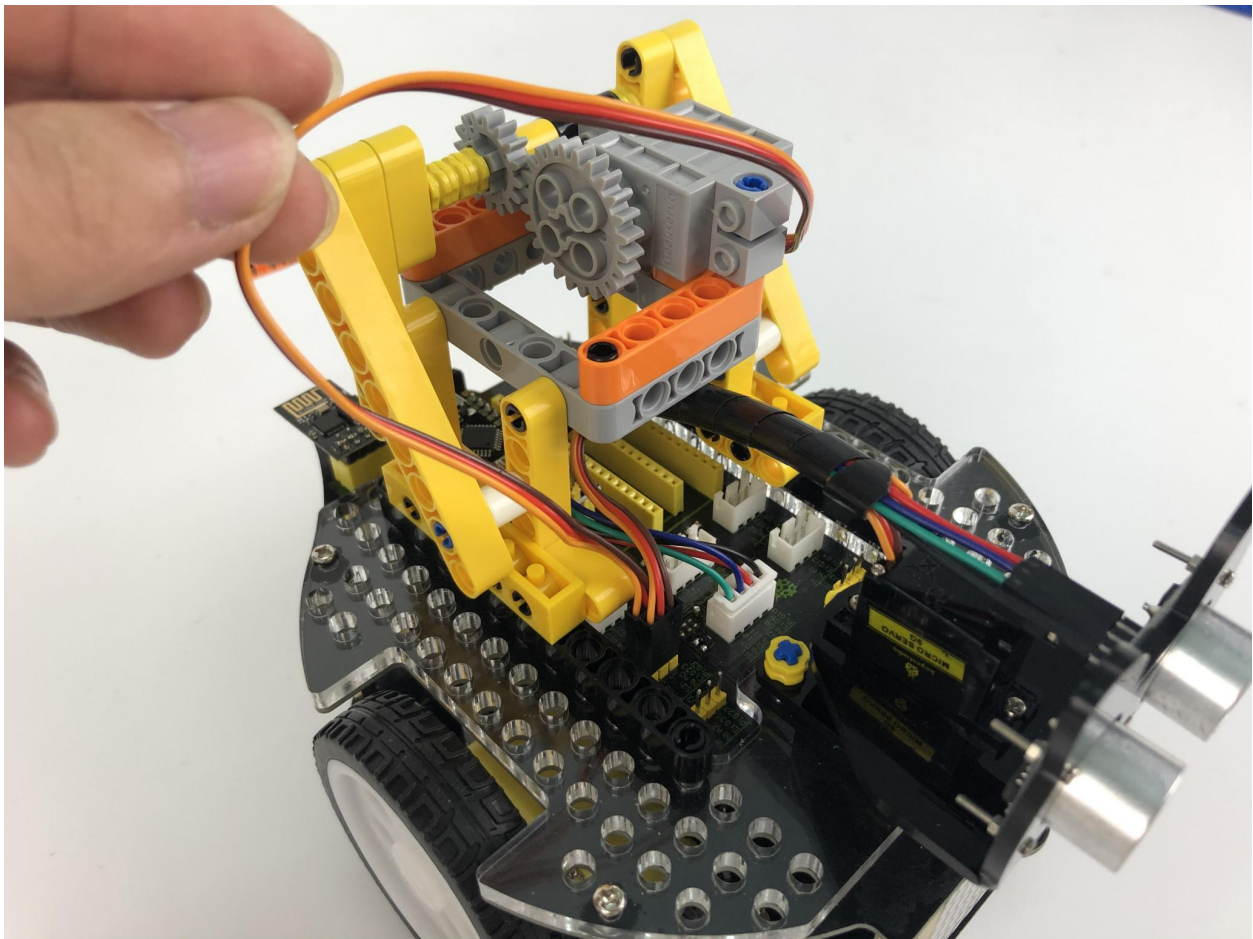


Step 18:



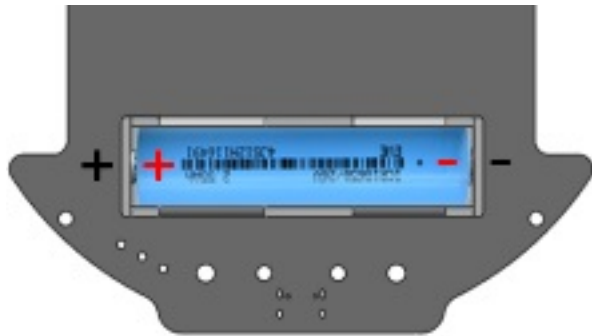


**Step 19:** Wire up  
Interface the servo



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





## 11.3 3. Code:

### 11.3.1 (1) Arduino Code

```
#include <Servo.h>
Servo lgservo;
#define ML 4
#define ML_PWM 6
#define MR 2
#define MR_PWM 5
#define servo2 A0

char val;
char wifiData;
boolean servo_flag = 1;

void setup() {
  Serial.begin(9600);
  pinMode(ML, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  lgservo.attach(A0);
  lgservo.write(0);
}

void loop() {
  if(Serial.available() > 0)
  {
    val = Serial.read();
    Serial.print(val);
  }
  switch(val)
  {
    case 'F': car_forward(); break;
    case 'B': car_back(); break;
    case 'L': car_left(); break;
```

(continues on next page)



(continued from previous page)

```

    case 'R': car_right(); break;
    case 'S': car_stop(); break;
    case 'p': lgservo.write(55); servo_flag = 1; break;
    case 'x': servo_down(); break;
  }
}

void servo_down()
{
  while( servo_flag == 1)
  {
    for(int i=55; i>0; i--)
    {
      lgservo.write(i);
      delay(2);
    }
    servo_flag = 0;
  }
}

void car_forward()
{
  digitalWrite(ML,LOW);
  analogWrite(ML_PWM,255);
  digitalWrite(MR,LOW);
  analogWrite(MR_PWM,255);
}

void car_back()
{
  digitalWrite(ML,HIGH);
  analogWrite(ML_PWM,0);
  digitalWrite(MR,HIGH);
  analogWrite(MR_PWM,0);
}

void car_left()
{
  digitalWrite(ML,HIGH);
  analogWrite(ML_PWM,150);
  digitalWrite(MR,LOW);
  analogWrite(MR_PWM,105);
}

void car_right()
{
  digitalWrite(ML,LOW);
  analogWrite(ML_PWM,105);
  digitalWrite(MR,HIGH);
  analogWrite(MR_PWM,150);
}

```

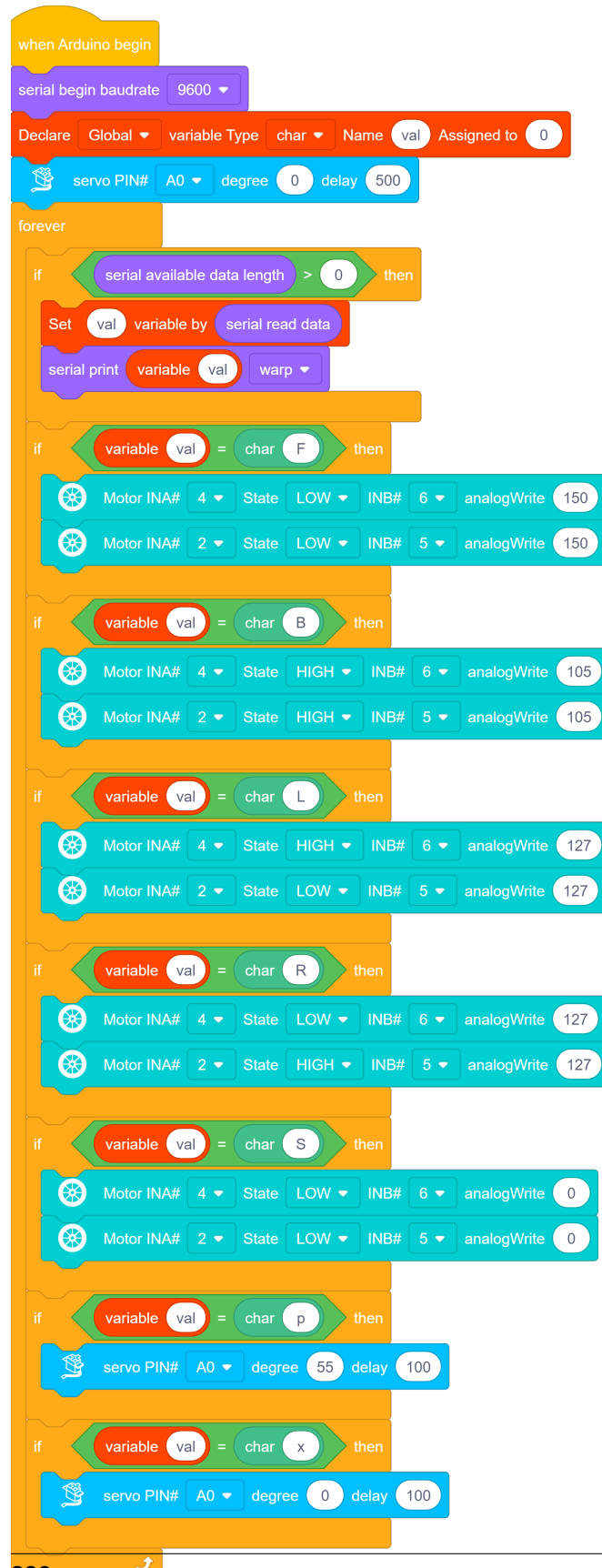
(continues on next page)

(continued from previous page)

```
}  
  
void car_stop()  
{  
    digitalWrite(ML,LOW);  
    analogWrite(ML_PWM,0);  
    digitalWrite(MR,LOW);  
    analogWrite(MR_PWM,0);  
}
```



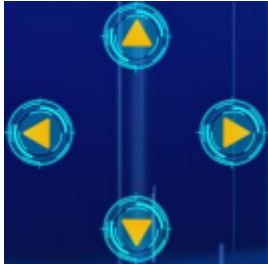
## 11.3.2 (2) Kidsblock Code





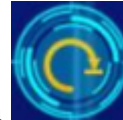
## 11.4 4. Test Result


Note: Please refer to the Project 11.2 of the Arduino tutorial for downloading and operating the APP.

Build up a few target objects with building blocks(object A, B, C, D, E) and keep them in a certain distance away the catapult and connect Wifi.



Click  to make the car to face the object A, hold down the button  to drive the catapult to launch a building block.



Then release the button  to make the long arm return to the original state. Next, let's check if the object A is hit by the launched block

You can repeat above steps to hit the object B, C and D.



## HANDLING TUTORIAL



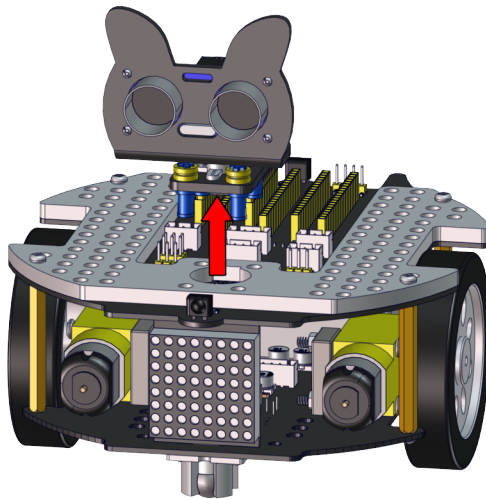
## 12.1 1. Description

Among many industrial robots, handling robots are undoubtedly effective, applied in industrial manufacturing, warehousing and logistics, tobacco, medicine, food, chemical and other industries, or in post offices, libraries, ports and parking lots. In this experiment, we will use LEGO blocks to build a handling robot to carry things.

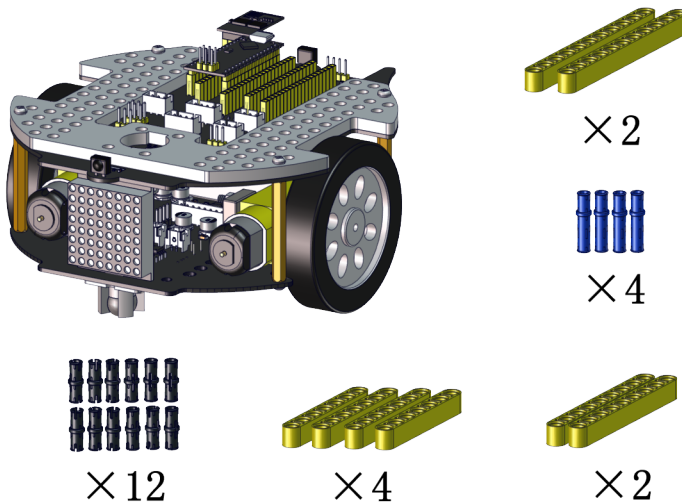
## 12.2 2. How to build up a handling robot:

### Step 1:

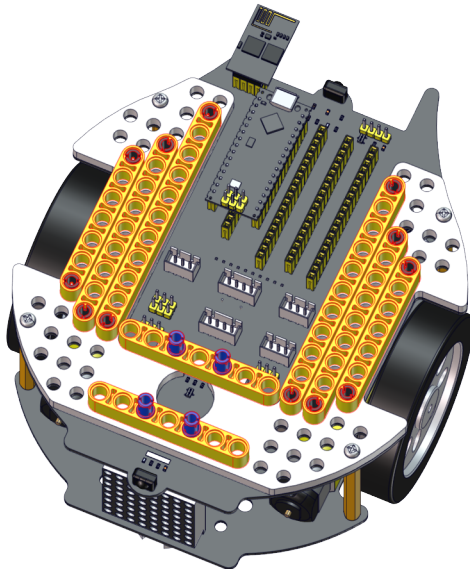
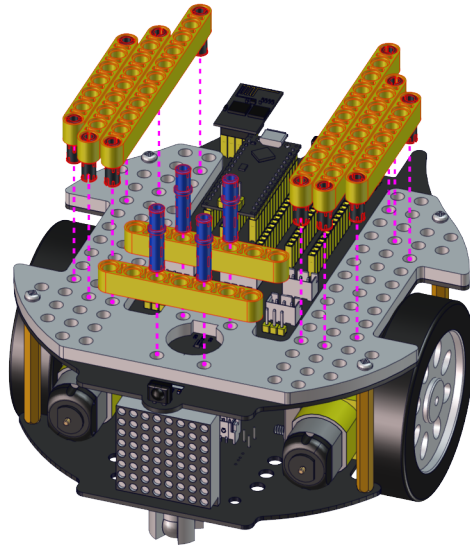
Dismantle the ultrasonic sensor



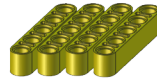
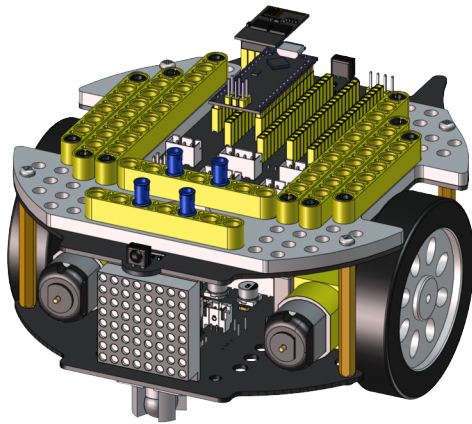
### Step 2:







Step 3:



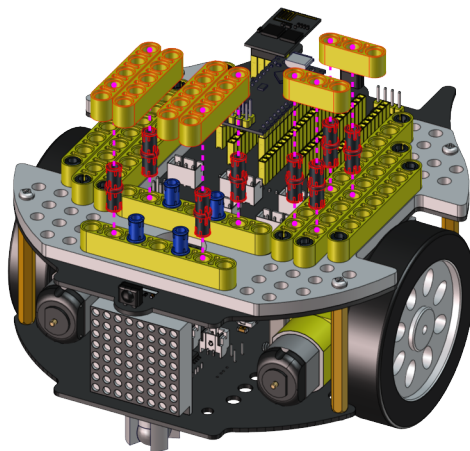
× 4

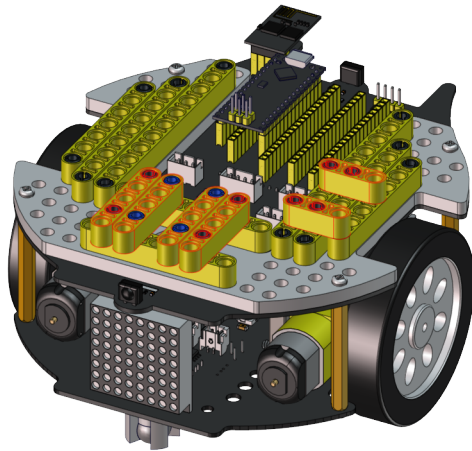


× 2

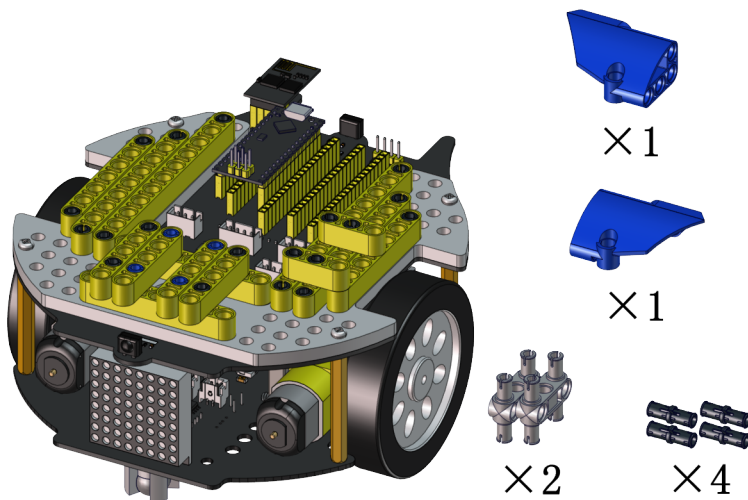


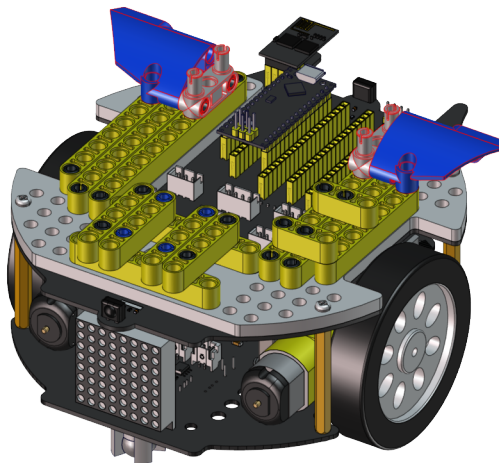
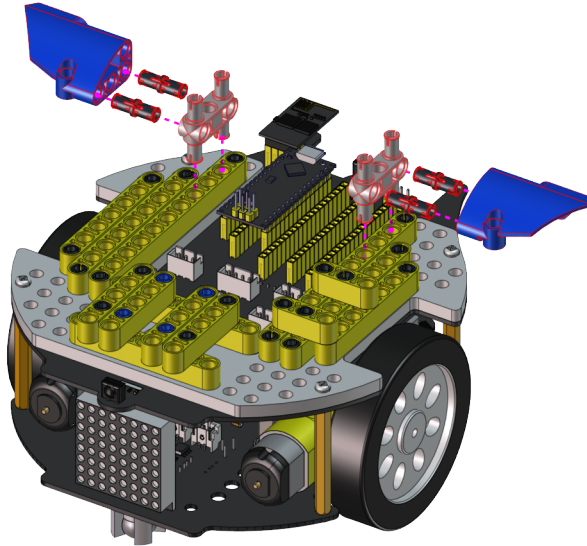
× 8



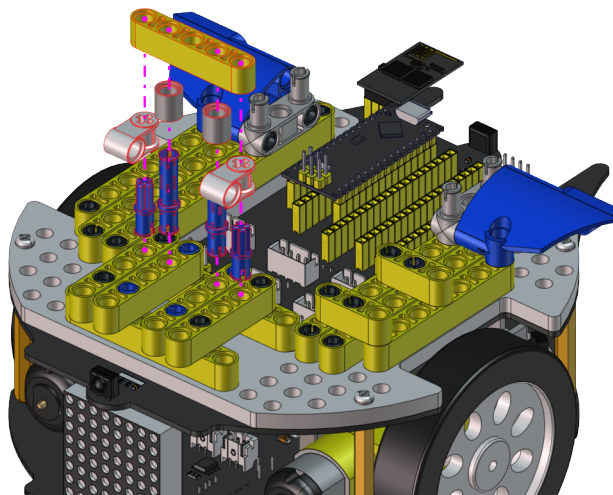
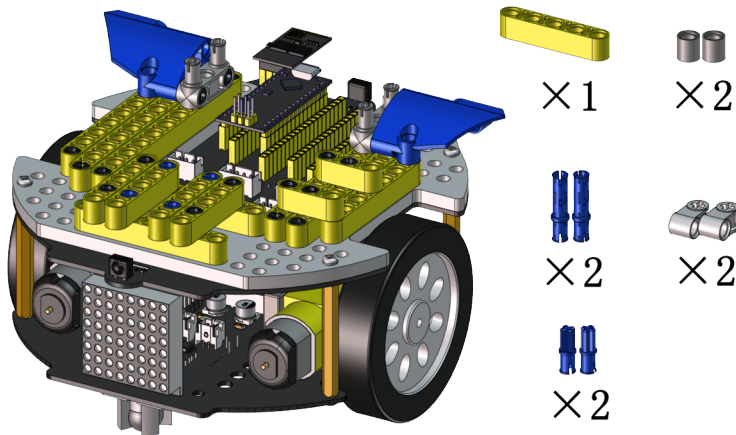


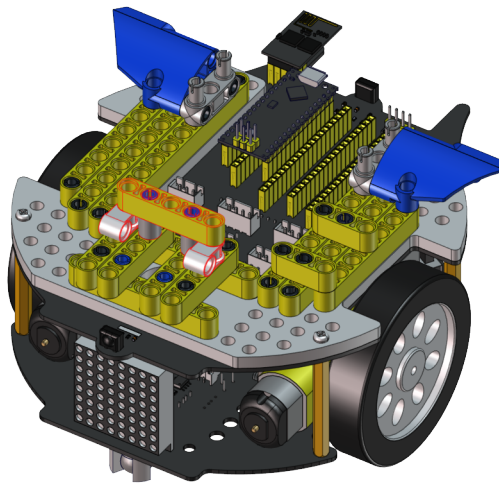
Step 4:



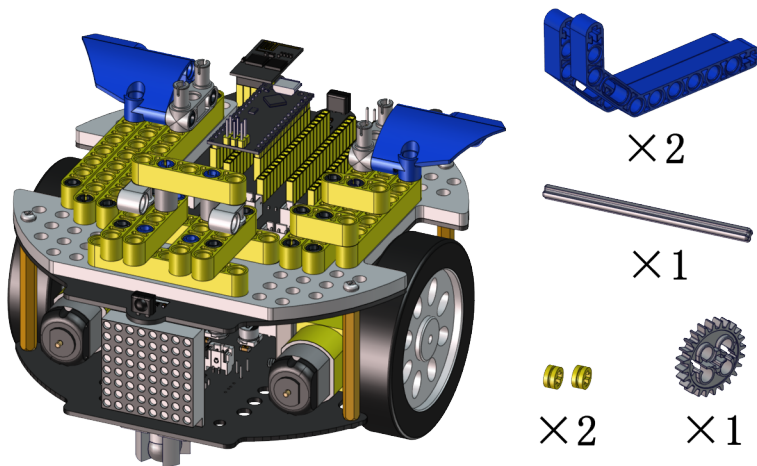


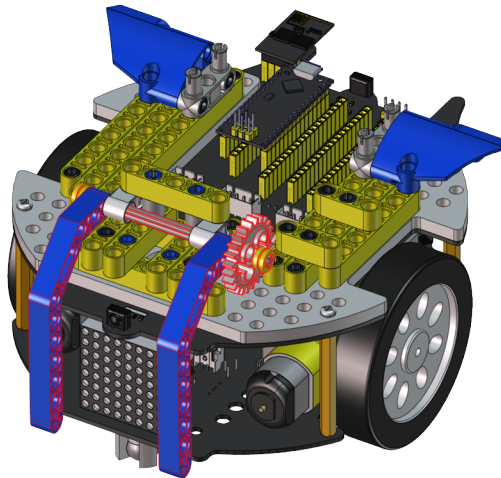
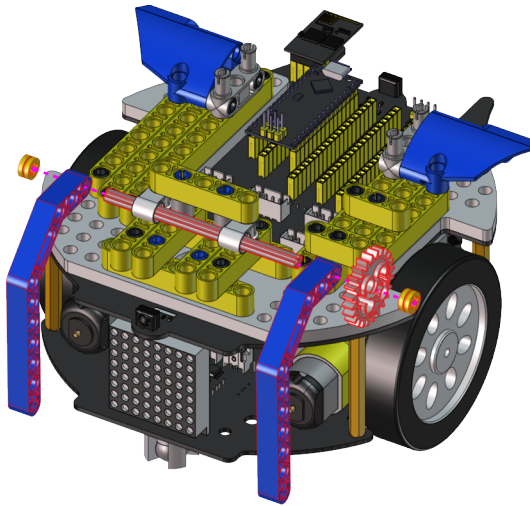
**Step 5:**





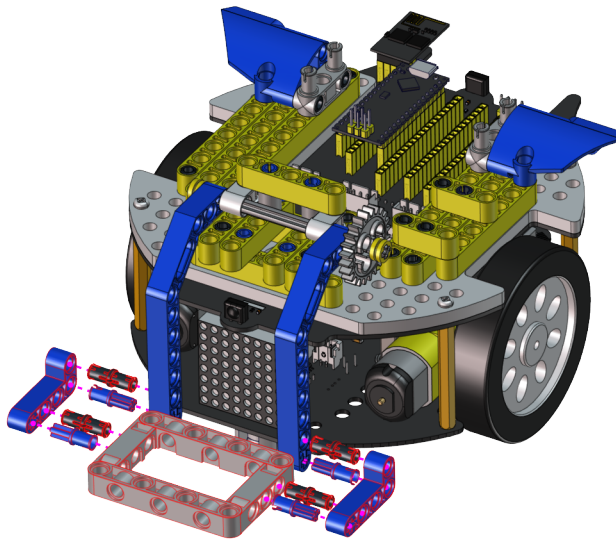
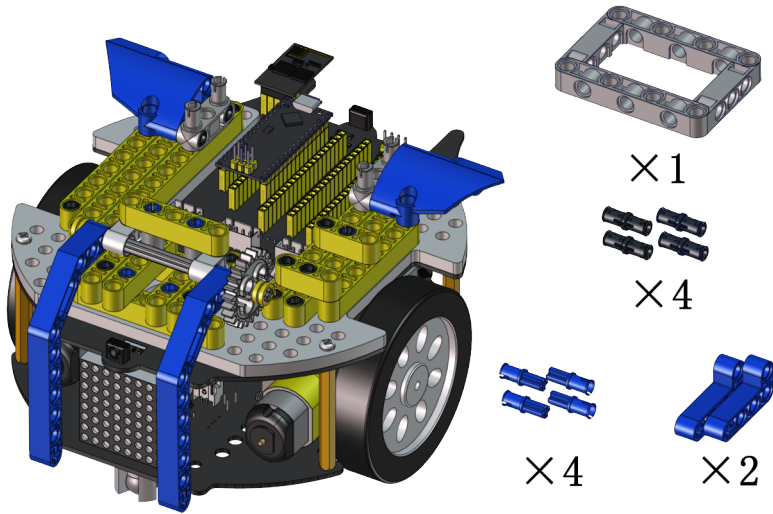
**Step 6:**



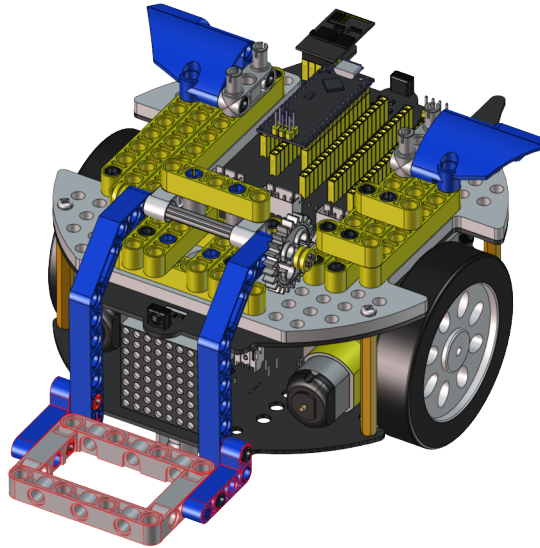


**Step 7:**

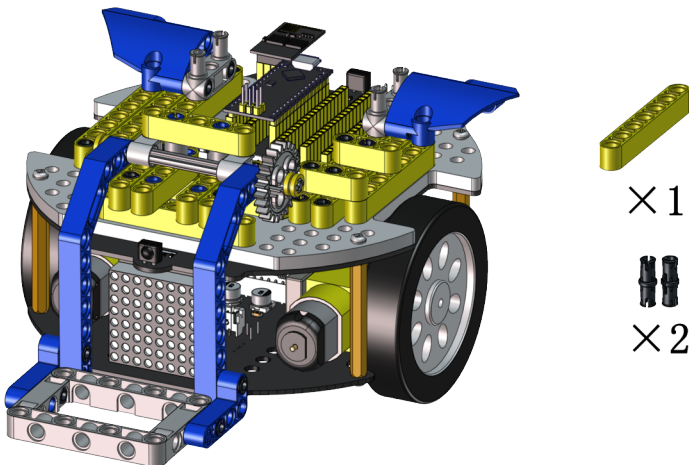


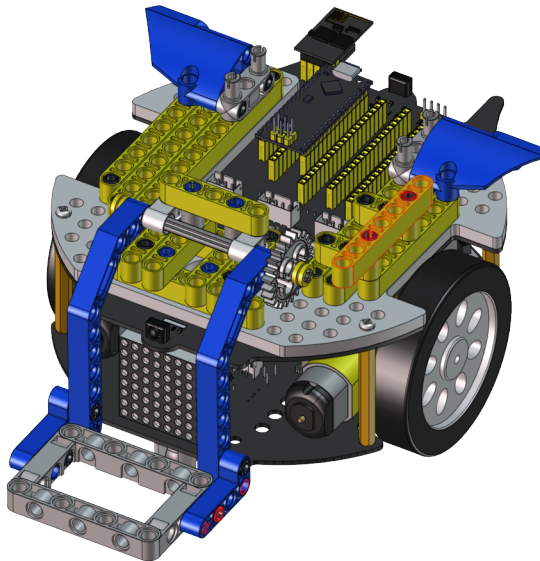
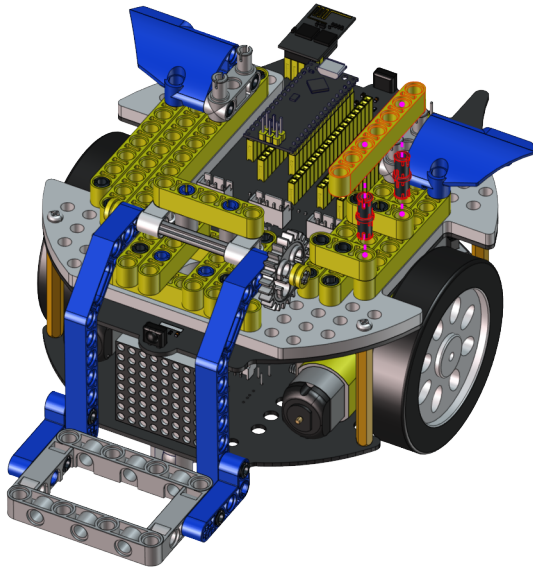




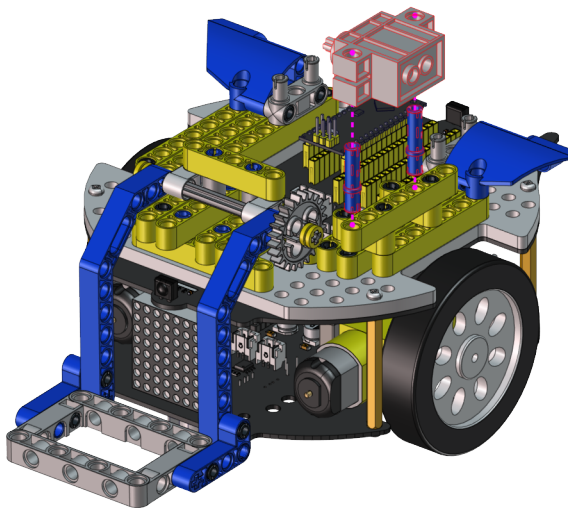
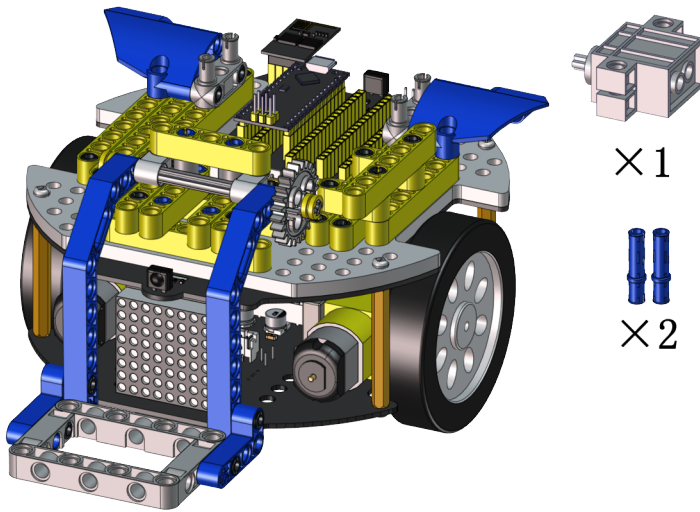


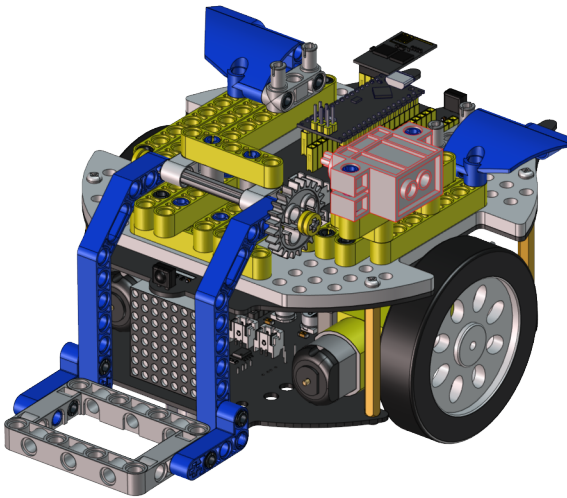
**Step 8:**





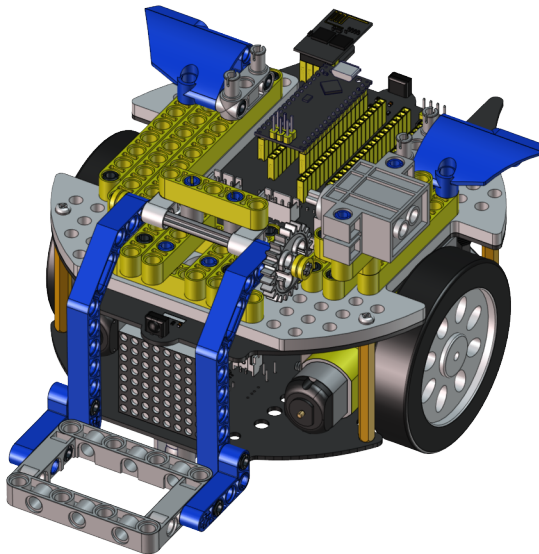
**Step 9:**





**Step 10:**

Set the angle of the servo to 180 degree



×1

# Wire servo up

| Servo  | PCB Board |
|--------|-----------|
| Brown  | G         |
| Red    | 5V        |
| Orange | S2 (A0)   |

Upload the code of the servo to the main board of the Beetlebot car, as shown below

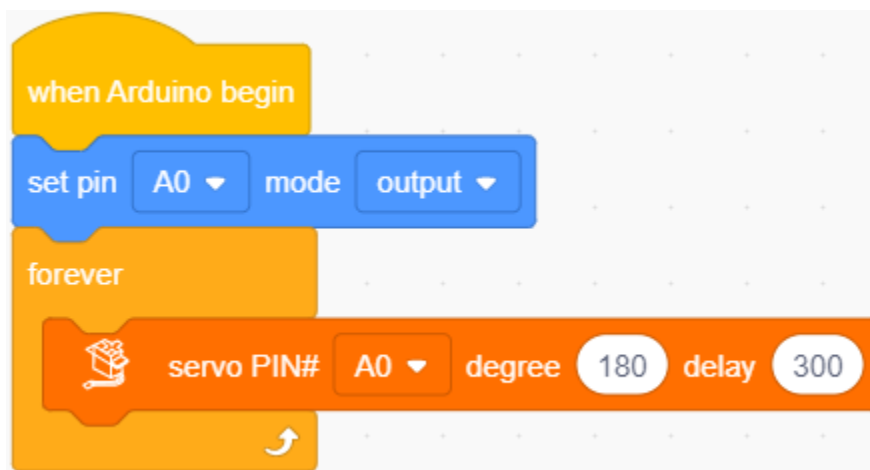
```
#include <Servo.h>

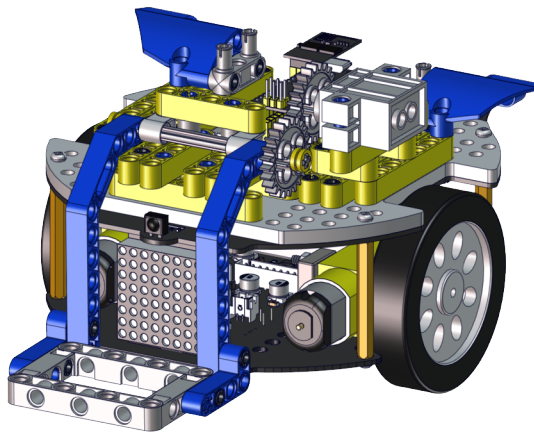
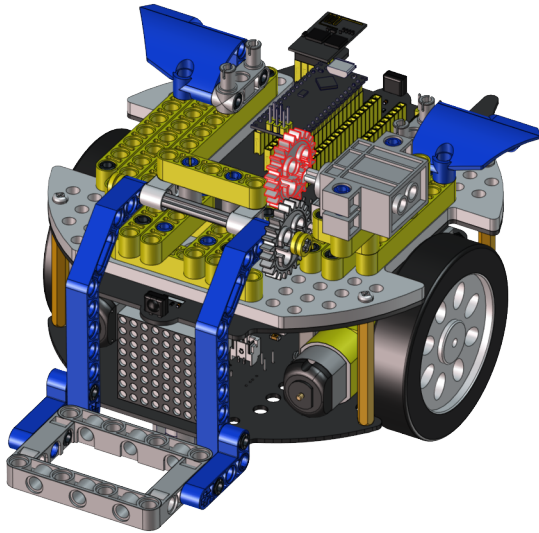
Servo myservo; // create servo object to control a servo

void setup() {
  myservo.attach(A0); // attaches the servo on pin A0 to the servo object
}

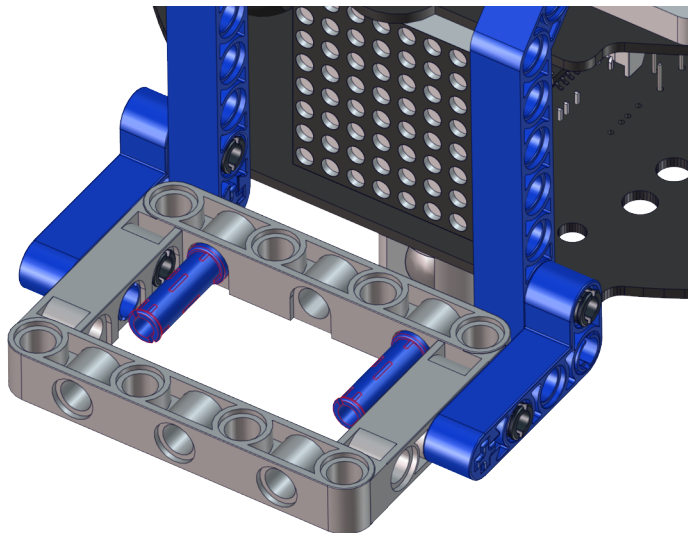
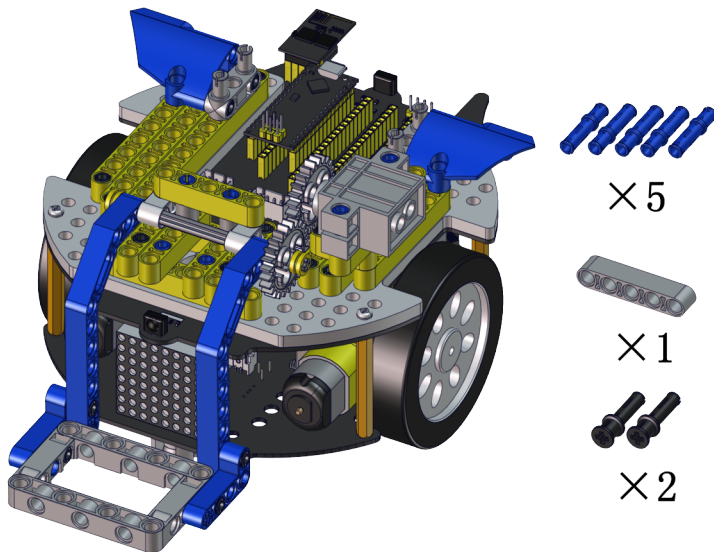
void loop() {
  myservo.write(180); // tell servo to go to position
}
```

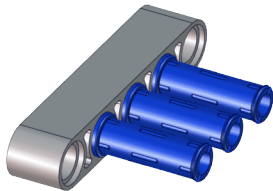
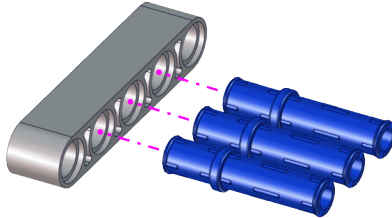
You can also initialize the angle of the servo through the following code



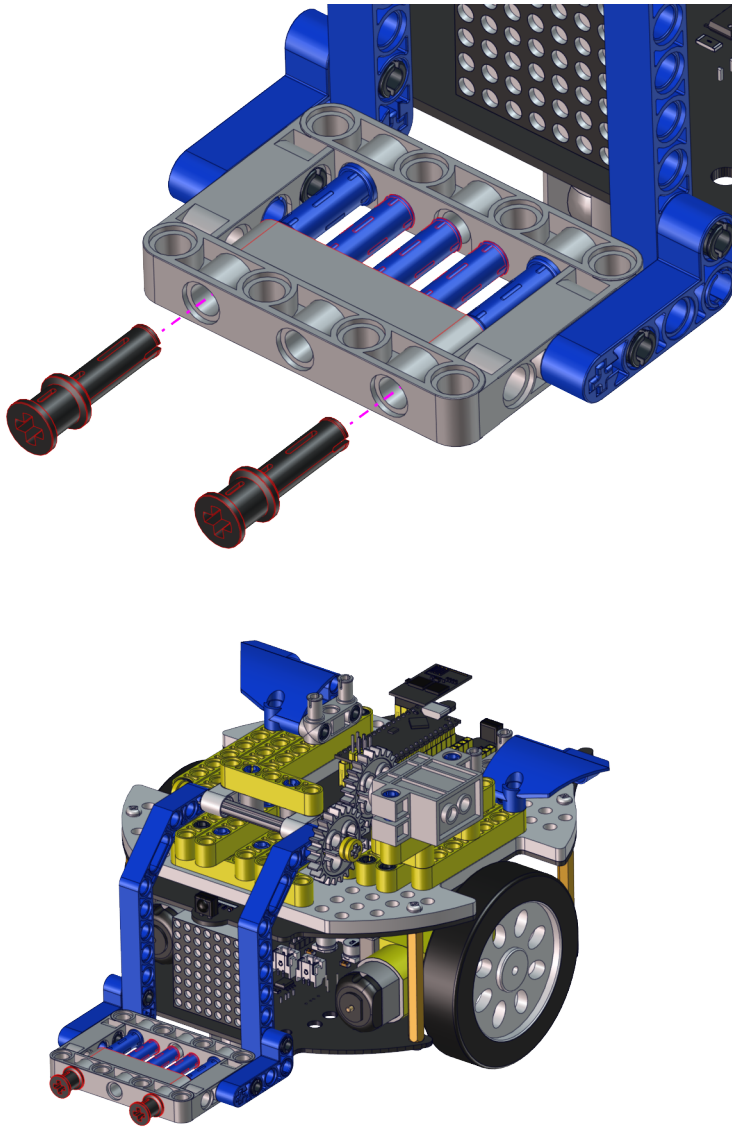


**Step 11:**

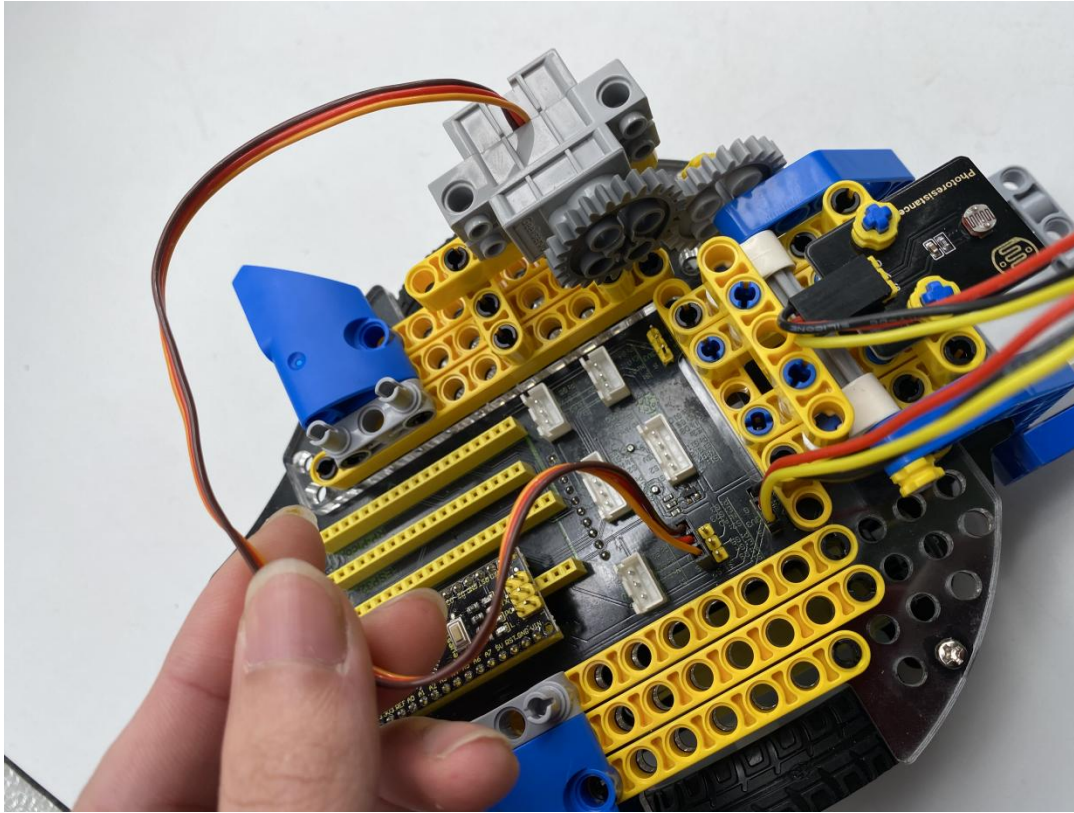




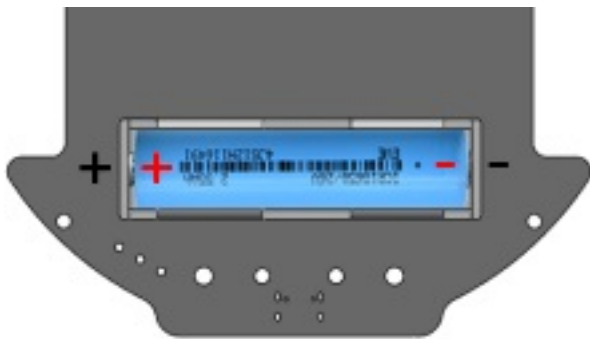




**Step 12:** Wire up servo



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



## 12.3 3. Code:

### 12.3.1 (1)Arduino Code

```
#include <Servo.h>
Servo lgservo;
#define ML 4
#define ML_PWM 6
#define MR 2
```

(continues on next page)

(continued from previous page)

```
#define MR_PWM 5
#define servo2 A0

char val;
char wifiData;

void setup() {
  Serial.begin(9600);
  pinMode(ML, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR, OUTPUT);
  pinMode(MR_PWM, OUTPUT);

  lgservo.attach(A0);
  lgservo.write(180);
}

void loop() {
  if(Serial.available() > 0)
  {
    val = Serial.read();
    Serial.print(val);
  }
  switch(val)
  {
    case 'F': car_forward(); break;
    case 'B': car_back(); break;
    case 'L': car_left(); break;
    case 'R': car_right(); break;
    case 'S': car_stop(); break;
    case 'p': lgservo.write(130); break;
    case 'x': lgservo.write(180); break;
  }
}

void car_forward()
{
  digitalWrite(ML, LOW);
  analogWrite(ML_PWM, 127);
  digitalWrite(MR, LOW);
  analogWrite(MR_PWM, 127);
}

void car_back()
{
  digitalWrite(ML, HIGH);
  analogWrite(ML_PWM, 127);
  digitalWrite(MR, HIGH);
  analogWrite(MR_PWM, 127);
}
```

(continues on next page)

(continued from previous page)

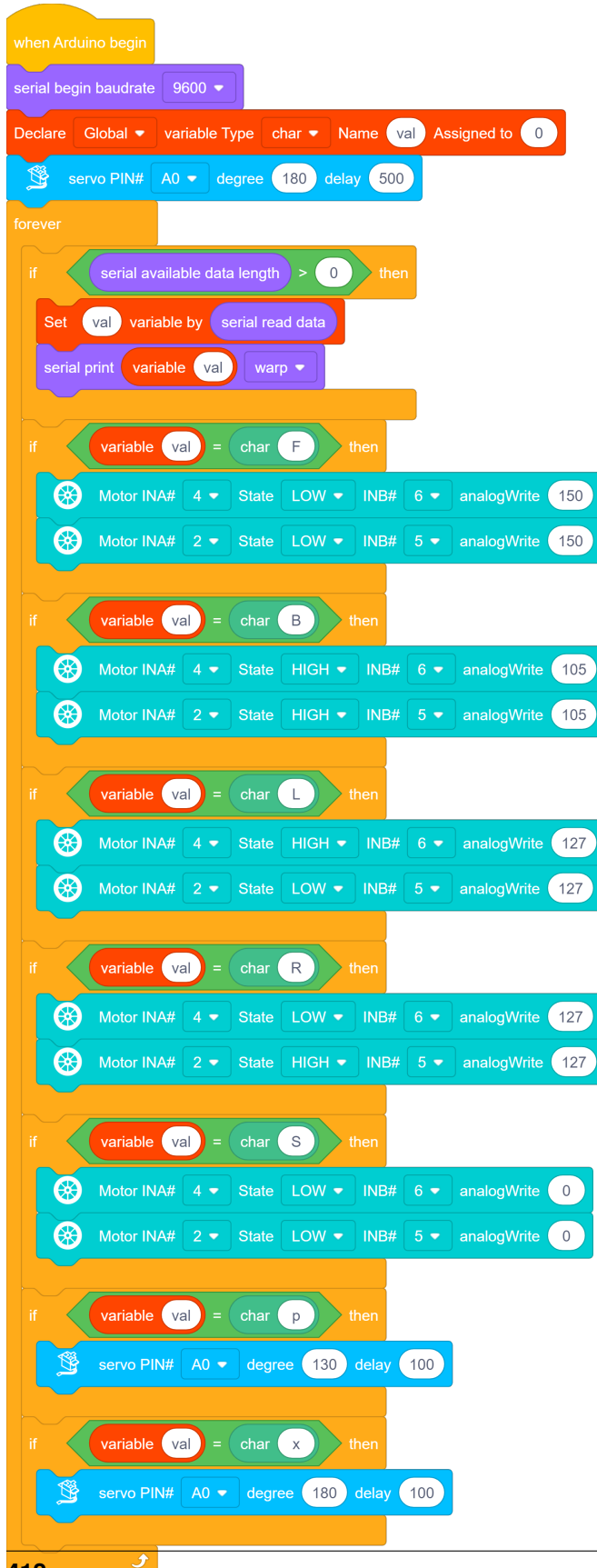
```
void car_left()
{
    digitalWrite(ML,HIGH);
    analogWrite(ML_PWM,150);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,105);
}

void car_right()
{
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,105);
    digitalWrite(MR,HIGH);
    analogWrite(MR_PWM,150);
}

void car_stop()
{
    digitalWrite(ML,LOW);
    analogWrite(ML_PWM,0);
    digitalWrite(MR,LOW);
    analogWrite(MR_PWM,0);
}
```

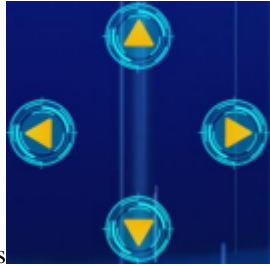


## 12.3.2 (3) Kidsblock Code\*\*

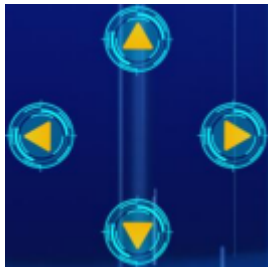


## 12.4 4. Test Result

Note: Please refer to the Project 11.2 of the Arduino tutorial for downloading and operating the APP.



Connect Wifi, click buttons to make the car to move toward building blocks and put some building blocks on the robot.



Then press to drive the robot to move.



Hold down the button to drive the robot to drop building blocks, then building blocks can be conveyed