Thanks for your support on our products, we will continue to provide you better quality and service!

Content

*About keyestudio

Keyestudio is a best-selling brand owned by KEYES Corporation. Our product lines range from controller boards, shields and sensor modules to smart car and complete starter kits for Arduino, Raspberry Pi and BBC micro:bit, which can help customers at any level learn electronics and programming knowledge. Likewise, all of our products comply with international quality standards and are greatly appreciated in a variety of different markets throughout the world.

You can obtain the details and the latest information through visiting the following web sites:http://www.keyestudio.com

*References and After-sales Service

1. Download Profile：https://fs.keyestudio.com/KS0507

2. Feel free to contact us please, if there is missing part or you encounter some troubles. Welcome to send email to us：**service@keyestudio.com.**

We will update projects and products continuously from your sincere advice.

*Warning

1. This product contains tiny parts(screws, copper pillars), keep it out of reach of children under 7 years old please.

2. This product contains conductive parts (control board and electronic module). Please operate according to the requirements of tutorial. Improper operation may cause parts to overheat damage. Do not touch

and immediately disconnect the circuit power.

*Copyright

4WD BT Multi-purpose Car V2.0 Kit

Arduino tutorial

1. Introduction

Nowadays, technological education such as VR, kids programming, and artificial intelligence, has become mainstream in educational industry. Thereby, people attach importance to STEAM education. Arduino is pretty notable in Maker education.

So what is Arduino? Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. Based on this, Keyestudio team has designed a 4wd robot. It has a processor which is programmable using the Arduino IDE, to mapped its pins to sensors and actuators by a shield that plug in the processor, it reads sensors and controls the actuators and decides how to operate.

15 learning projects, from simple to complex, will guide you how to make a smart 4wd robot on you own and introduce the detailed knowledge about sensors and modules.

Simultaneously, it is the best choice if you intend to obtain a DIY robot for learning programming, entertainment and competition requirement.

Note: The experiment you did should be in line with wiring diagram, including about components and wiring method. For example, we supply power with external power in the hook-up diagram, so you also have to use external power rather than USB cable .

2. Features

1. Multi-purpose function: Obstacle avoidance, follow, IR remote control, Bluetooth control, ultrasonic follow and displayed face emoticons.

2. Easy to build: No soldering circuit required, complete assembly easily.

3. High Tenacity: Aluminum alloy bracket, metal motors, high quality wheels and tracks

4. High extension: expand other sensors and modules through motor driver shield and sensor shield

5. Multiple controls: IR remote control, App control(iOS and Android system)

6. Basic programming: C language code of Arduino IDE.

3. Specification

Working voltage: 5v

Input voltage: 7-12V

Maximum output current: 2A

Maximum power dissipation: 25W (T=75℃)

Motor speed: 5V 200 rpm/min

Motor drive mode: dual H bridge drive

Ultrasonic induction angle: <15 degrees

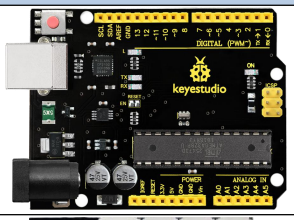
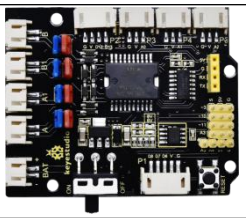
Ultrasonic detection distance: 2cm-400cm

Infrared remote control distance: 10 meters (measured)

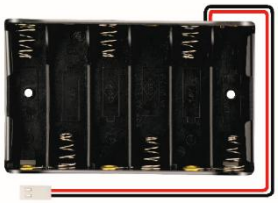
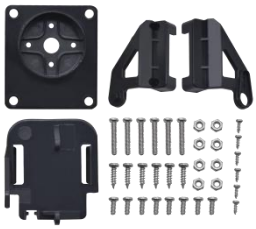
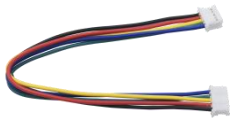
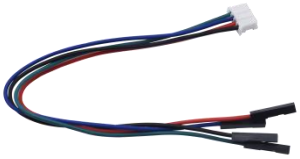
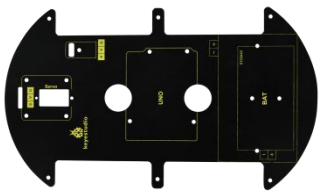
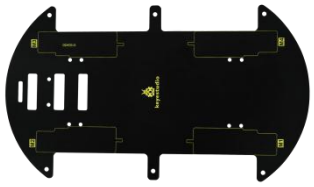
Bluetooth remote control distance: 50 meters (measured)

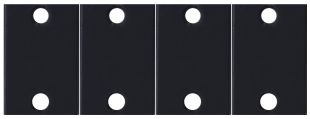
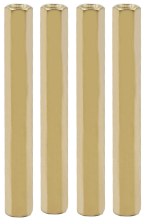
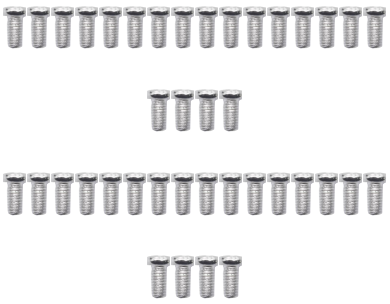
Bluetooth control: support Android and iOS system

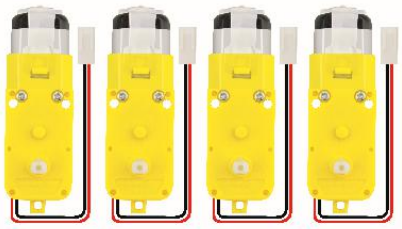
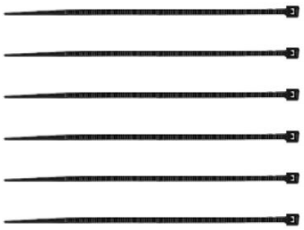
4. Product List

#	Name	QTY	Picture
1	Keyestudio V4.0 Board	1	
2	Keyestudio Motor Driver Shield	1	

3	Keyestudio HM-10 Bluetooth-4.0	1	
4	Red LED Module	1	
5	HC-SR04 Ultrasonic Sensor	1	
6	Keyestudio Line Tracking Sensor	1	
7	Keyestudio IR Receiver Sensor	1	
8	Keyestudio 8*16 LED Dot Matrix	1	
	4pinDupont Line	1	
9	Keyestudio 9G Servo	1	
10	Keyestudio Remote Control	1	
11	USB Cable	1	
12	18650 Battery Holder	1	

13	6-Slot AA Battery Holder	1	
14	Servo Platform	1	
15	Double Head JST-PH2.0MM-5P 24AWG Line 15CM	1	
16	8cm Double Head JST-PH2.0MM-3P 24AWG Line	1	
17	JST-PH2.0mm-4P to 2.54 DuPont Female Line	1	
18	Acrylic Board	1	
19	Keyestudio 4WD Smart Car V2.0 Top Board	1	
20	Keyestudio 4WD Smart Car V2.0 Bottom PCB	1	

21	Fixed Parts	4	
22	Wheel	4	
23	M3*10MM Dual-pass Copper Bush	10	
24	M3*40MM Dual-pass Copper Bush	4	
25	M3*30MM Round Head Screws	8	
26	M3*6MM Round Head Screws	40	
27	M3 Nickel Plated Nuts	16	
28	M2X8MM Round Head Screws	6	

29	M3*8MM Round Head Screws	4	
30	M2 Nickel Plated Nuts	6	
31	M3*10MM Flat Screws	3	
32	Motor (with welding wire)	4	
33	3*40MM Screwdriver	1	
34	Black Nylon Ties 3*100MM	6	
35	Winding Pipe	1	
36	3Pin F-F Dupont Wire (20CM)	3	
37	Decorative Board		

5. Assembly Guide

Note: Peel the plastic film off the board first when installing the smart car.

Step 1:Mount the Bottom PCB

- Prepare the parts as follows:

Gear motor *4

Fixed part *4

M3 nickel plated nut *10

M3*6mm round-head screw *14

4WD bottom PCB *1

Tracking sensor *1

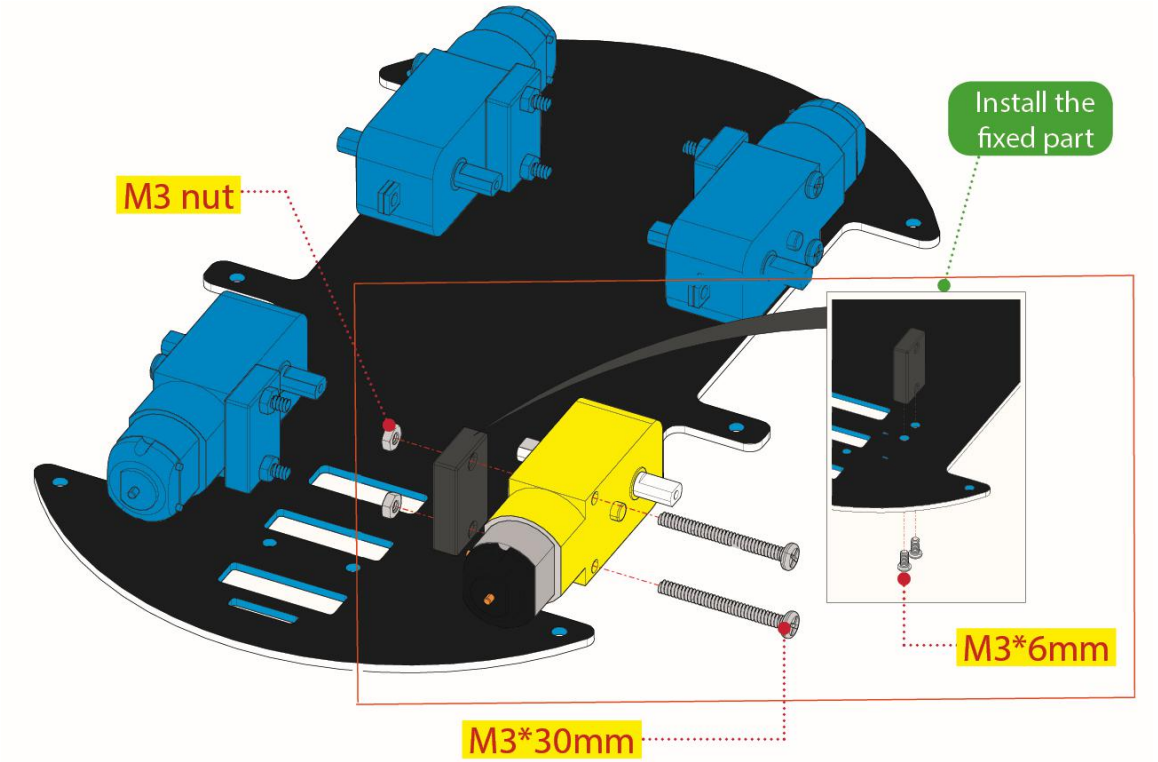
Wheel *4

Anti-reverse and dual 5p wire *1

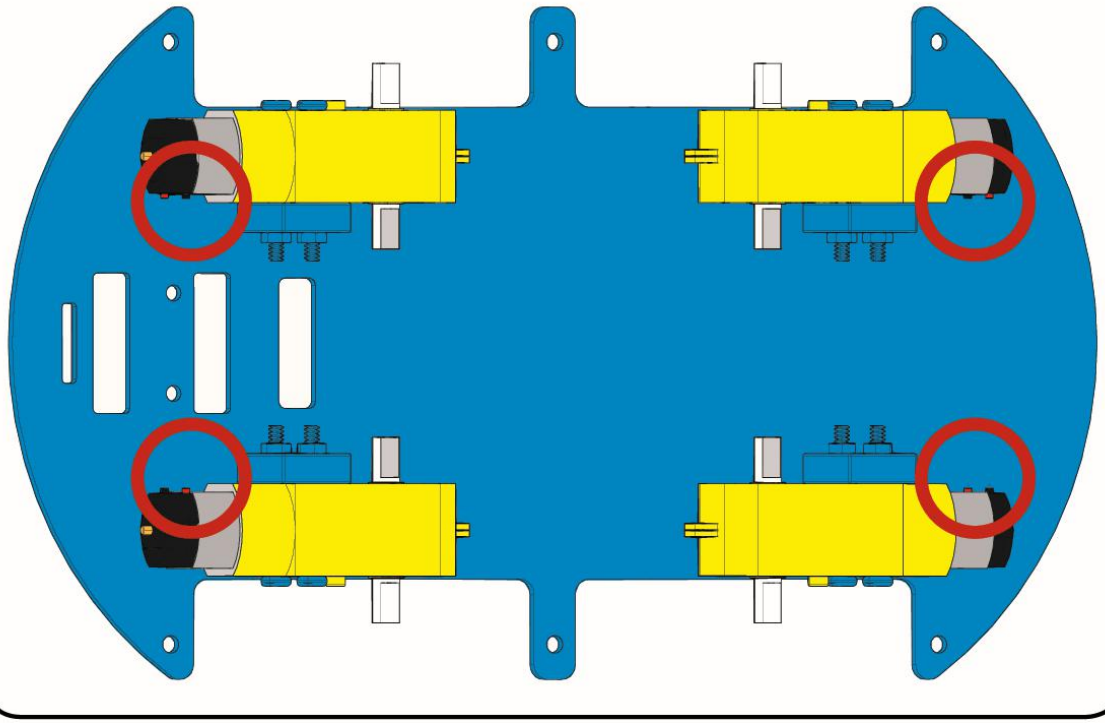
M3*40mm copper pillar*6

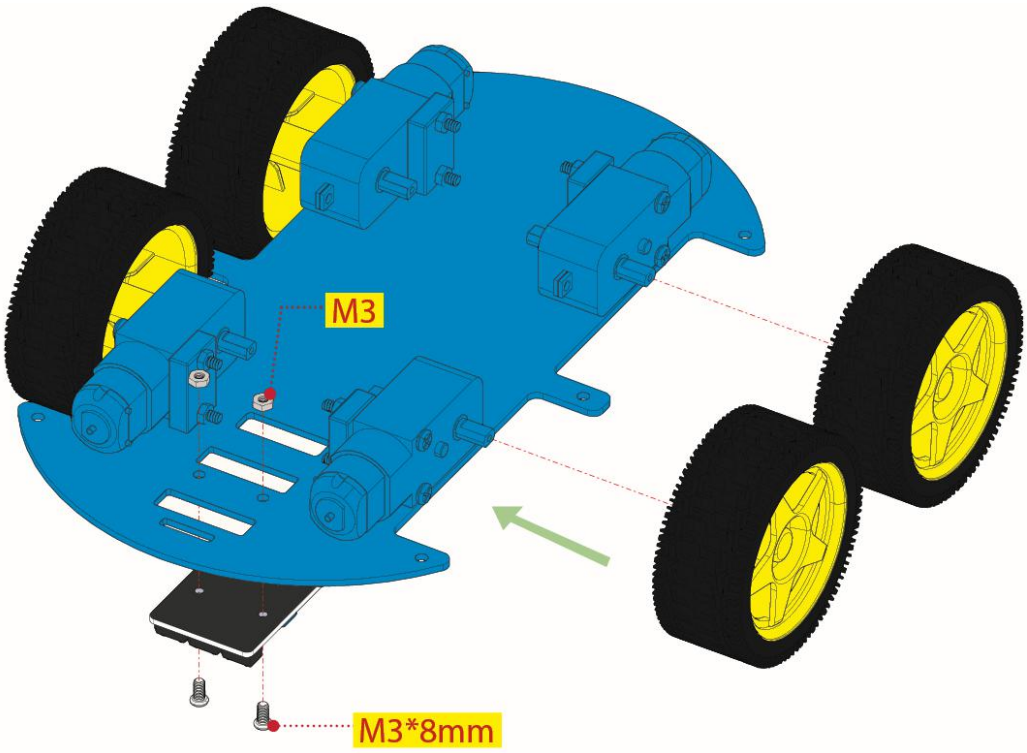
M3*30m round -head screw *8

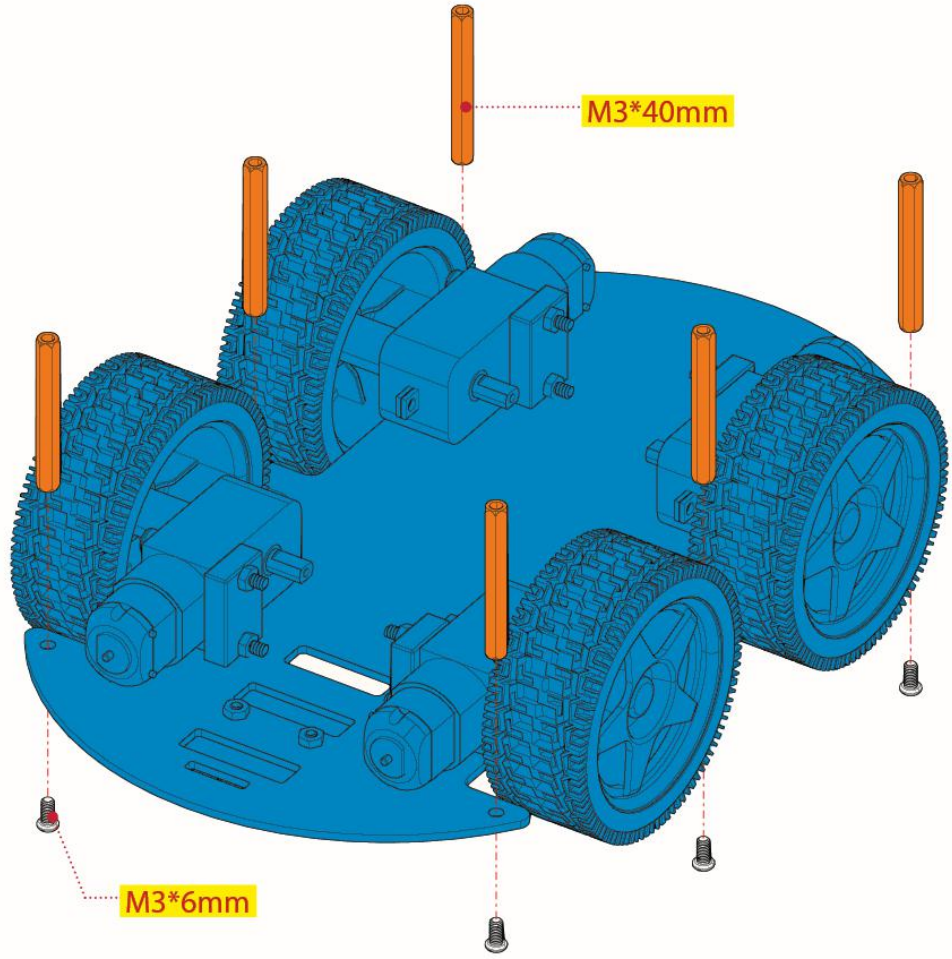
M3*8mm round-head screw *2

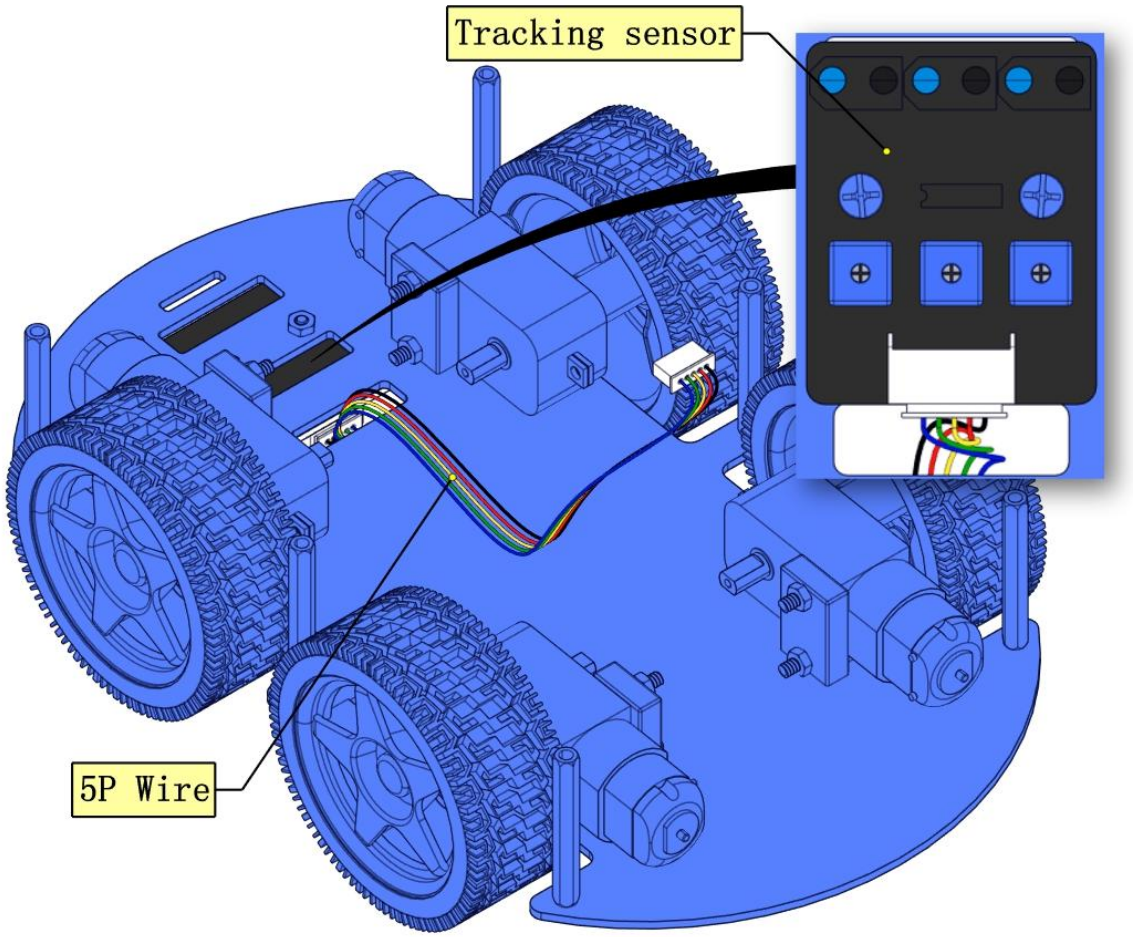


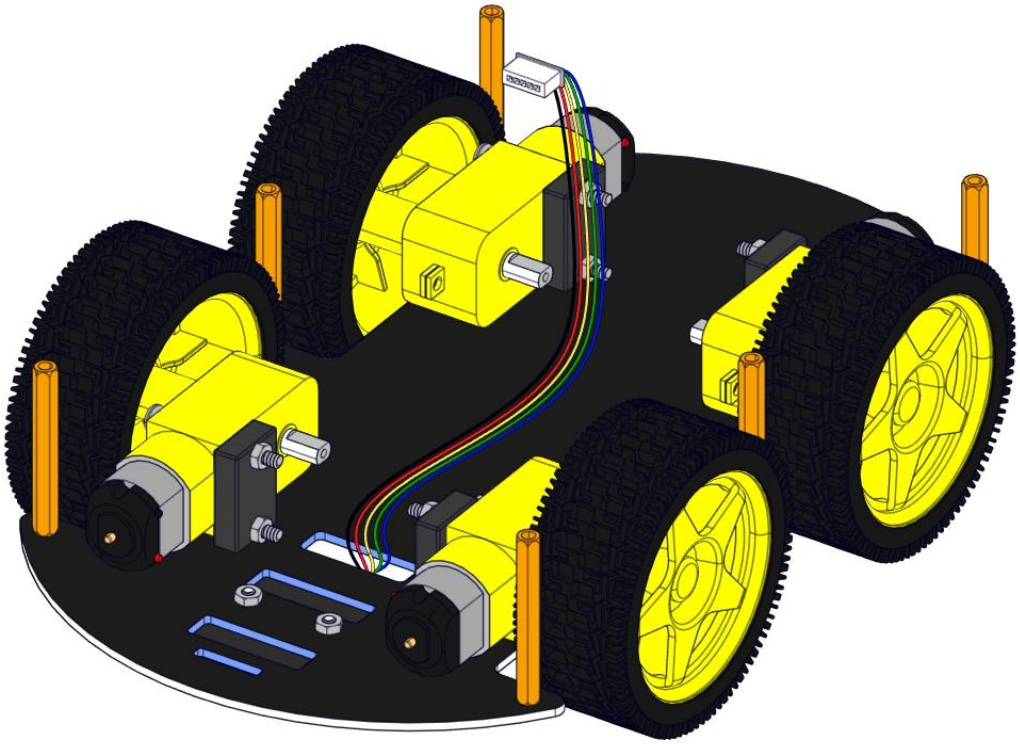
Note: The ways to install motors are same, pay attention to the direction please. Their position holes are downward and the red and black lines are inward




M3
M3*8mm


M3*40mm
M3*6mm


Tracking sensor
5P Wire



Step 2: Install Dot Matrix

- Prepare the parts as follows:

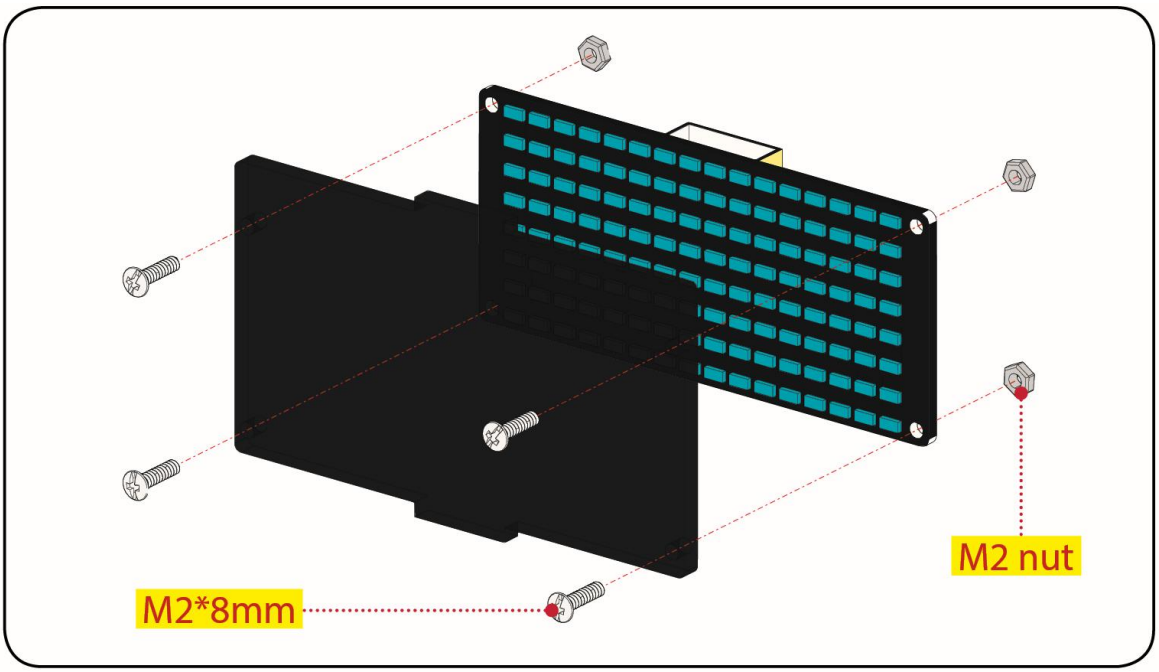
8X16 LED panel *1

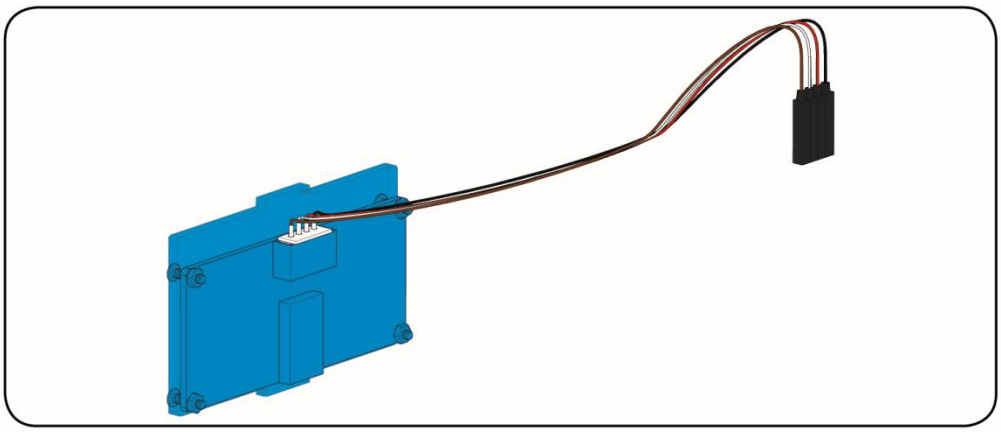
4WD baffle

4P wire *1

M2x8mm round-head screw *4

M2 nut *4





Step 3: Servo plastic platform

- Prepare the parts as follows:

Servo *1

M2*4 screw *1

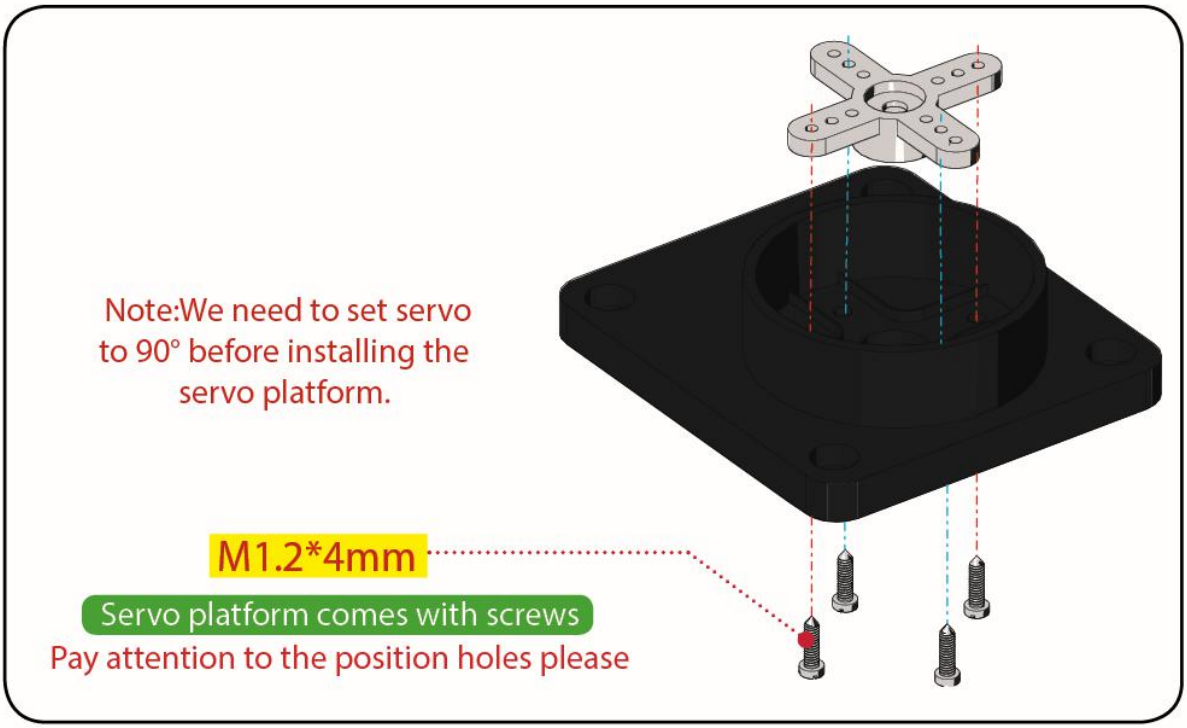
Black cable tie*2

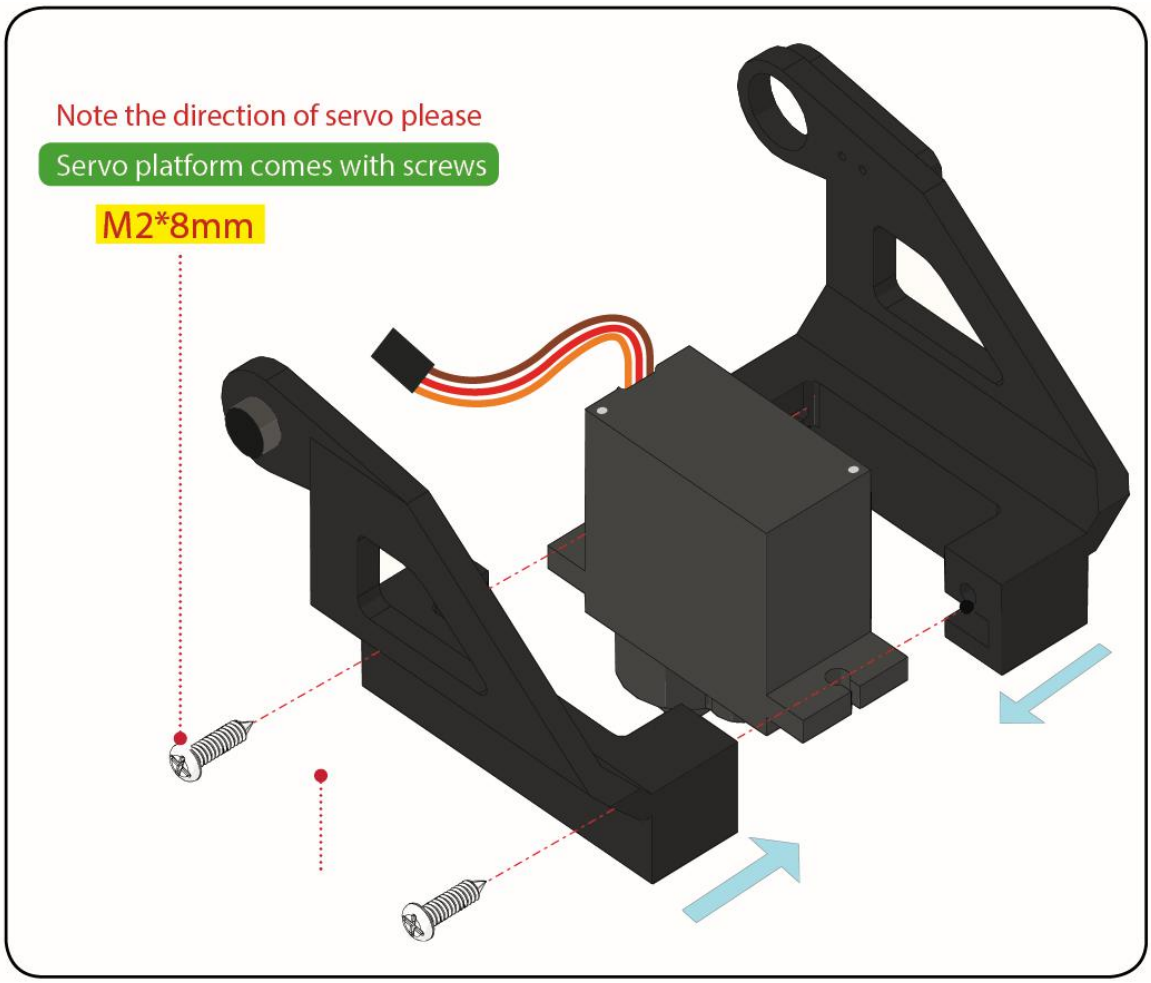
Ultrasonic sensor*1

Black plastic platform *1

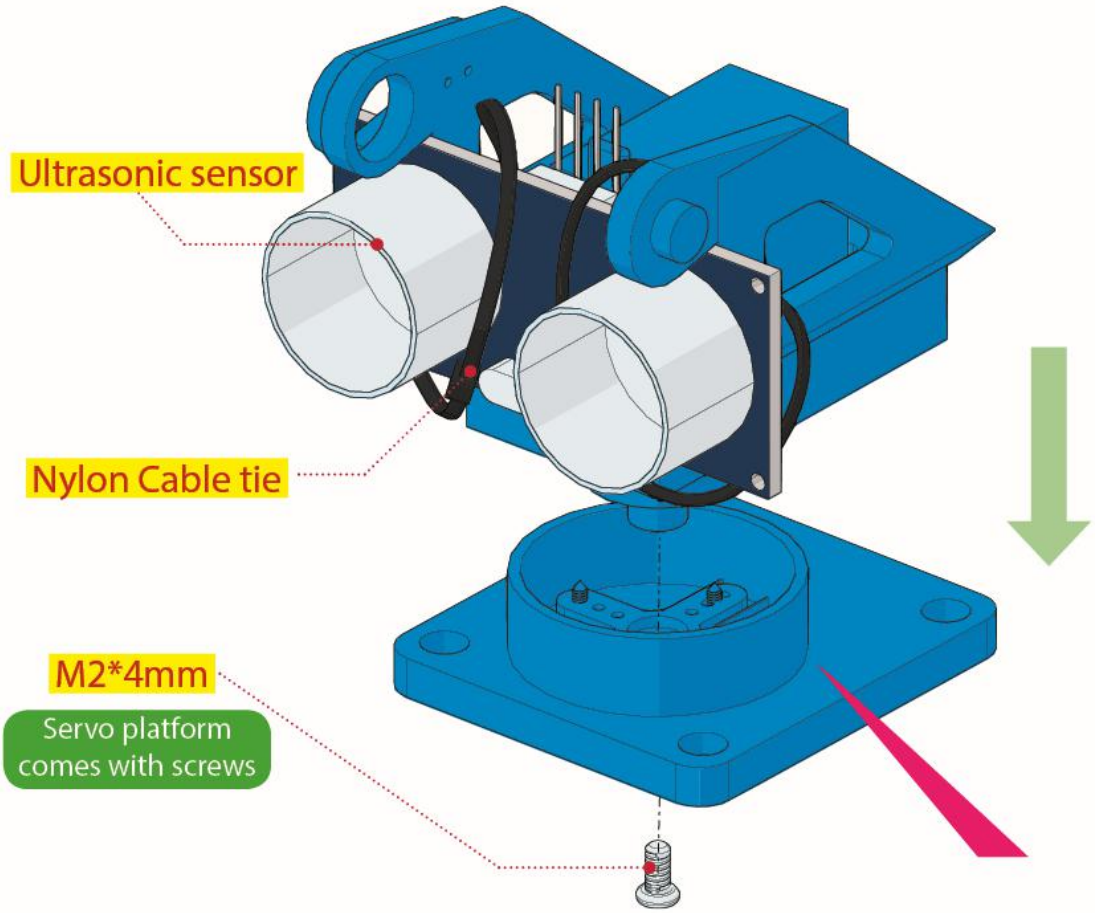
M1.2*4 Tapping screw *4

M2*8 tapping screw *2

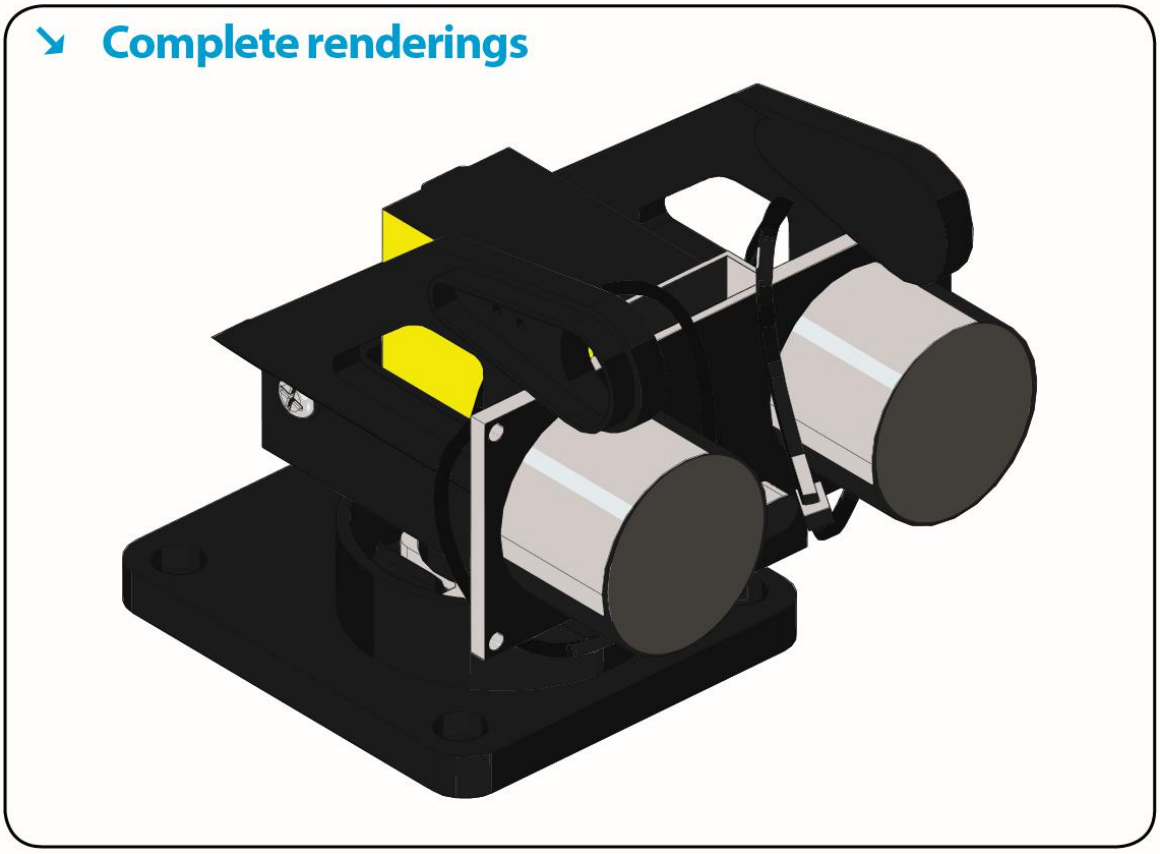



Note the direction of servo please
Servo platform comes with screws
M2*8mm

Complete renderings



Note: The direction of base should be complied with the diagram,
For convenient debugging, confirm that the ultrasonic module is in front of tank robot



Step 4: Install the Top PCB

- Prepare the parts as follows:

Top PCB *1

M3 nut *3

Motor drive board *1

Control board *1

Ir receiver module *1

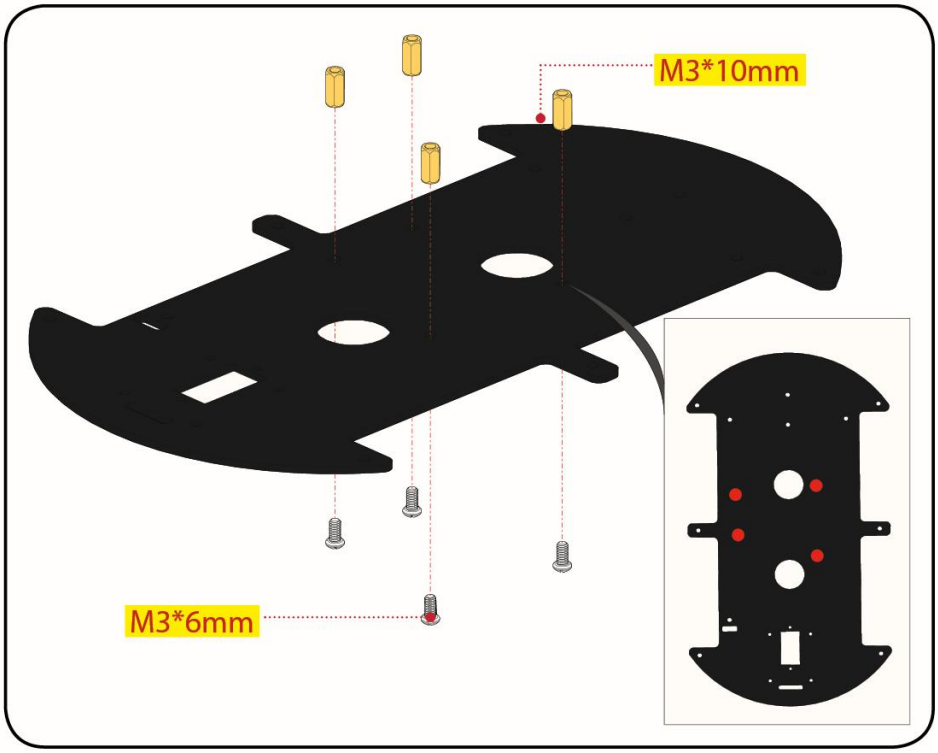
M3*10mm copper pillar *8

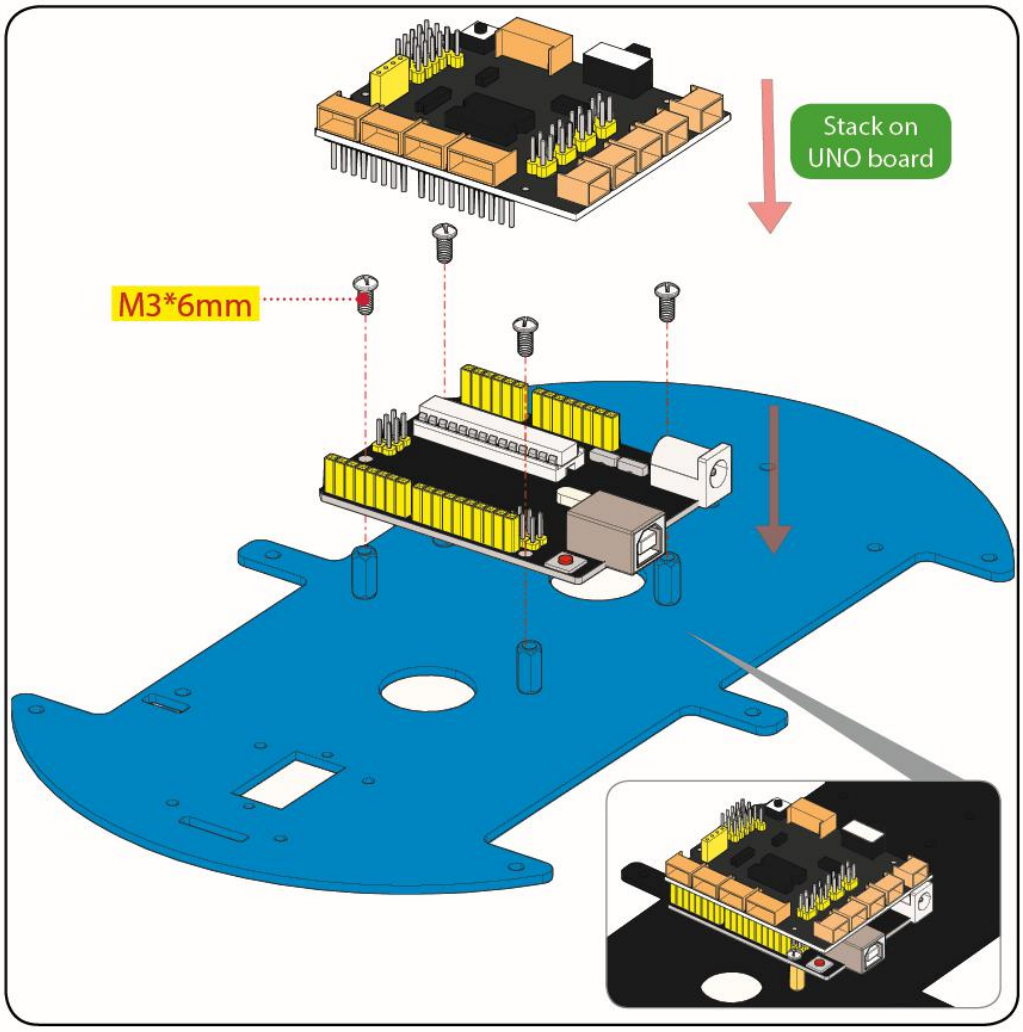
M3*8mm round-head screw *1

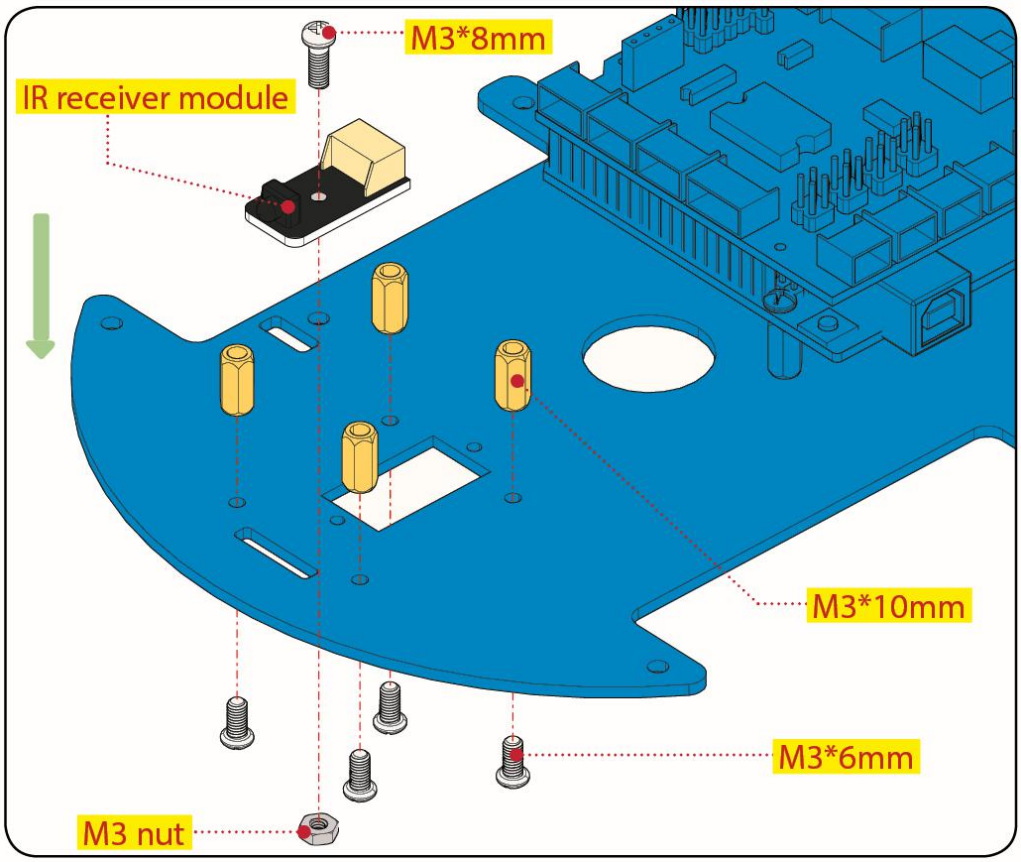
M3*6mm round-head screw *16

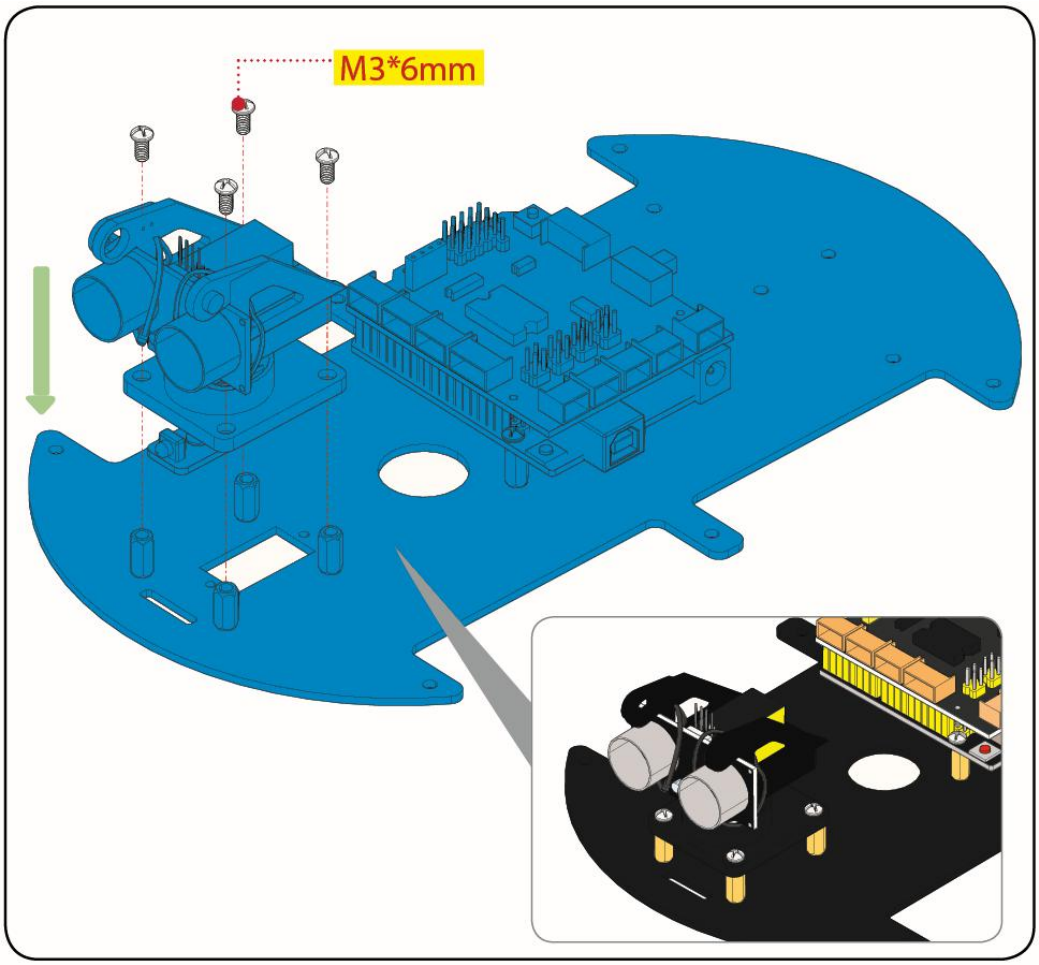
M3*10mm flat screw *2

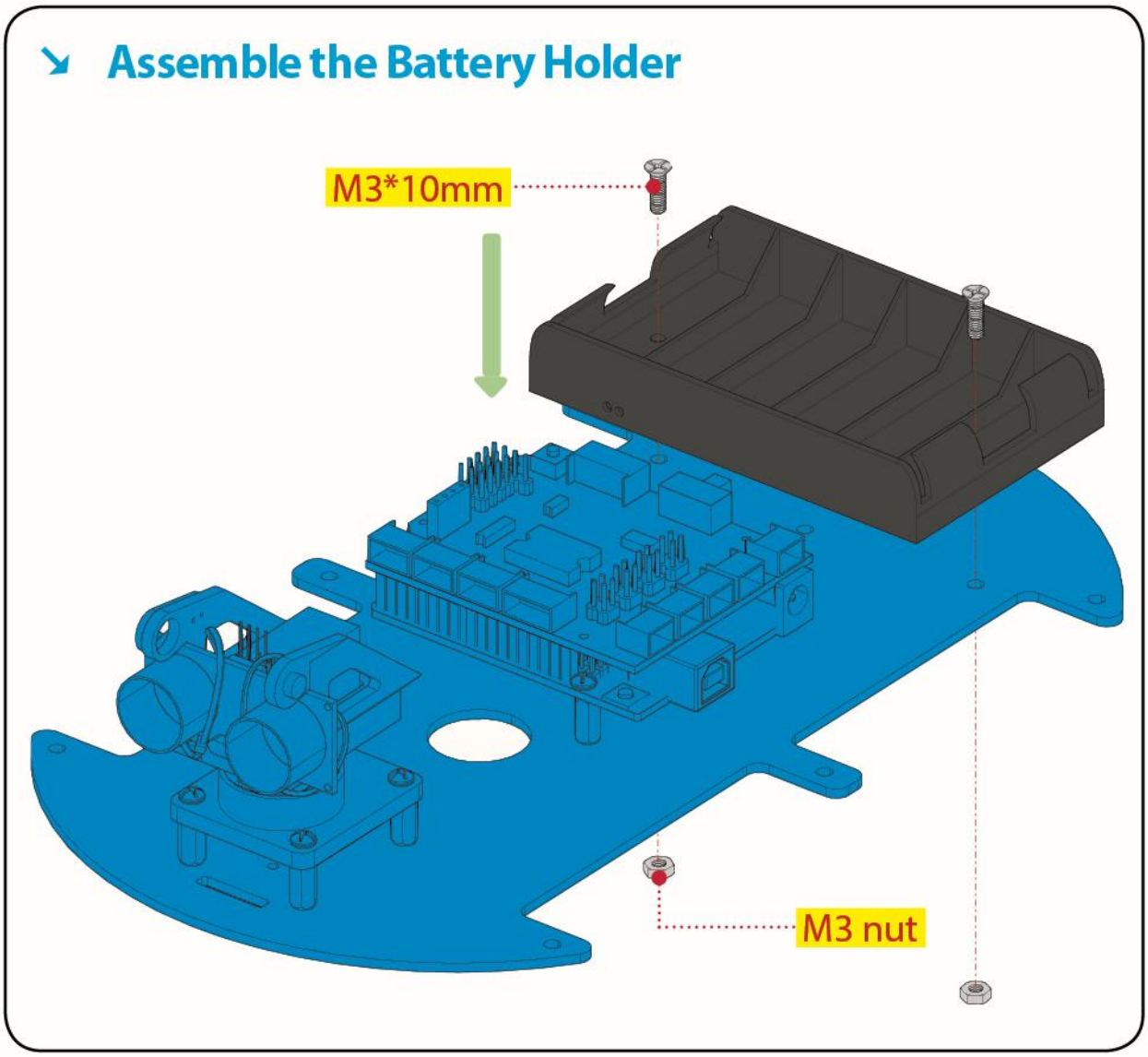
6-Slot AA battery holder *1




Stack on
UNO board
M3*6mm


M3*8mm
IR receiver module
M3*10mm
M3*6mm
M3 nut


M3*6mm



Step 5: Install the Top PCB

- Prepare the parts as follows:

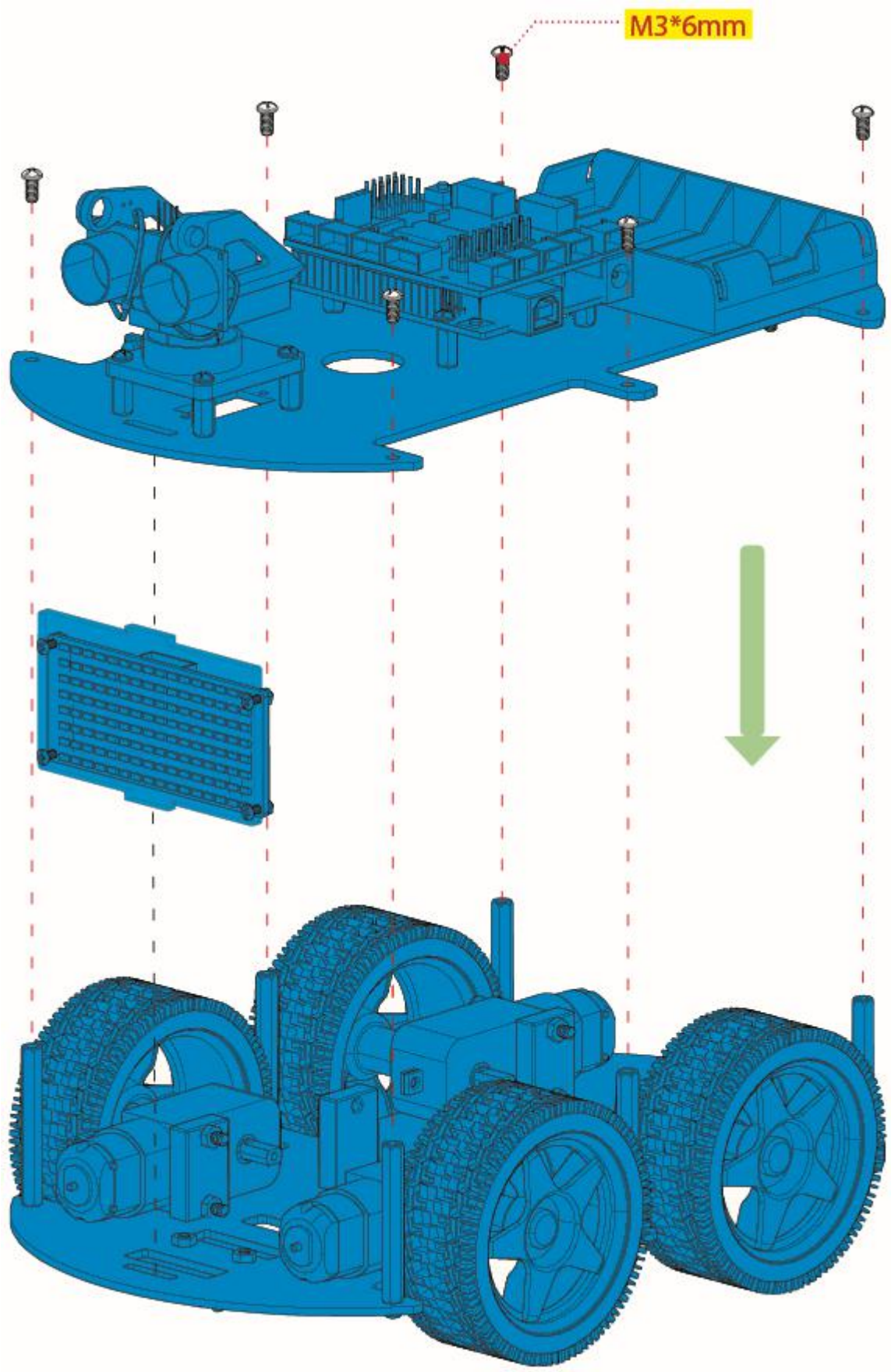
  Bluetooth module *1

  M3*6MM round-head screw *6

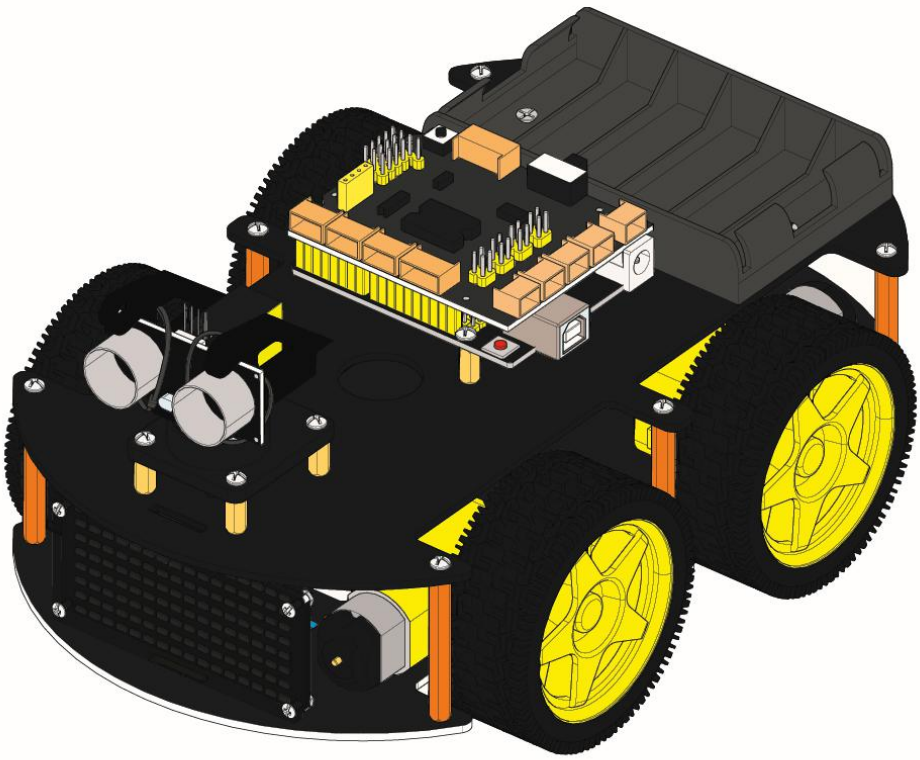
  Jumper caps *8

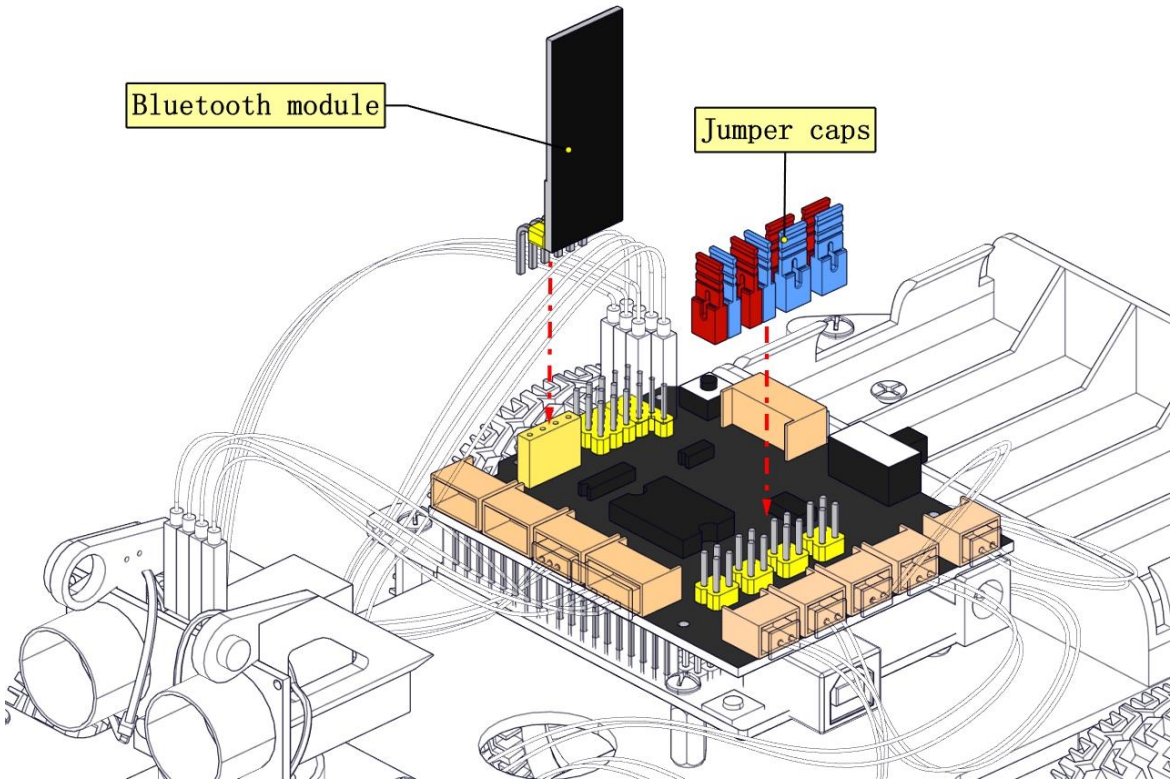
Note: you need to operate the following steps first before stacking

1. Insert 4P wire of dot matrix and lines (M2, M3) of motor into the front hole
2. Insert 5P wire of line tracking sensor and the lines (M1, M4)of motor into the back hole

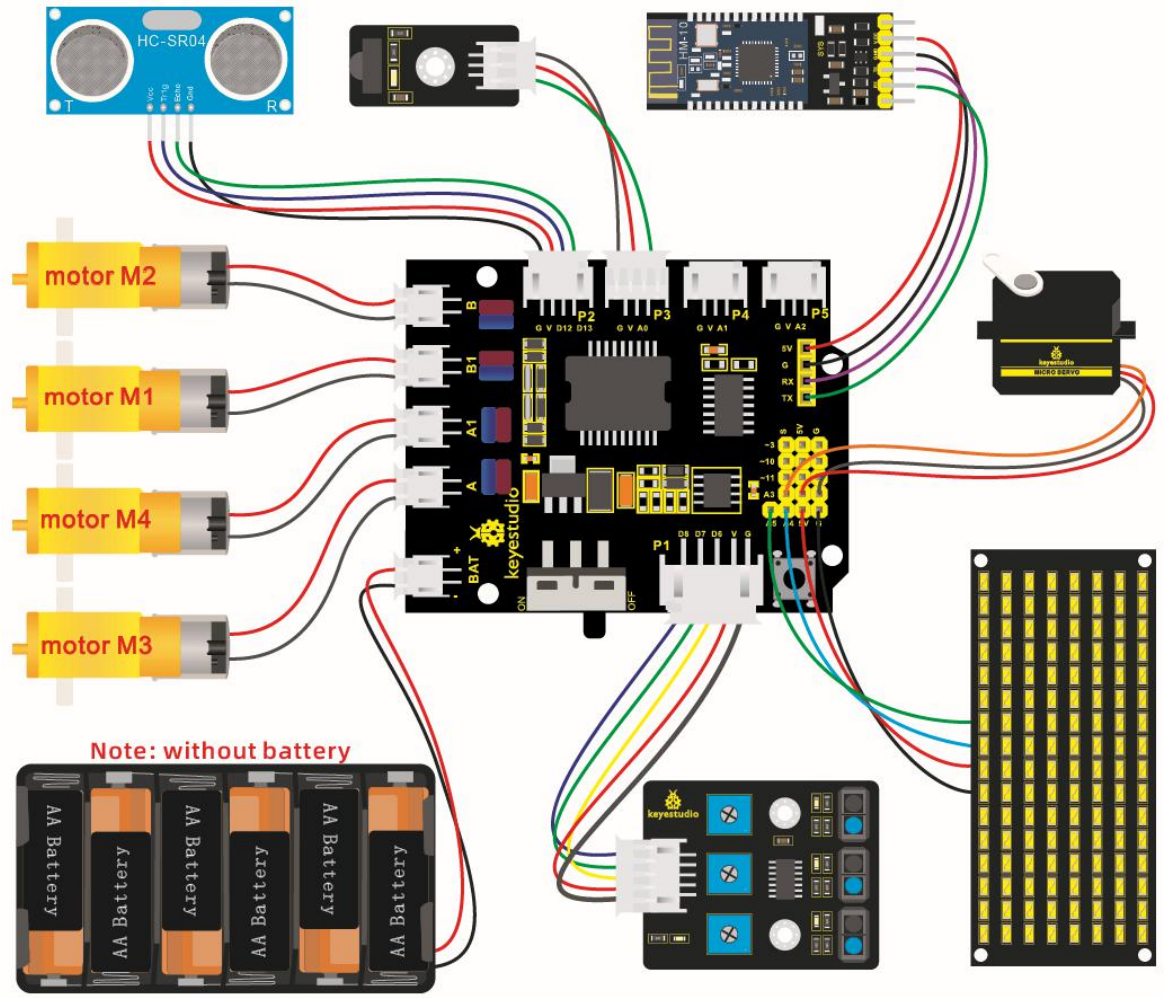

M3*6mm

Complete renderings

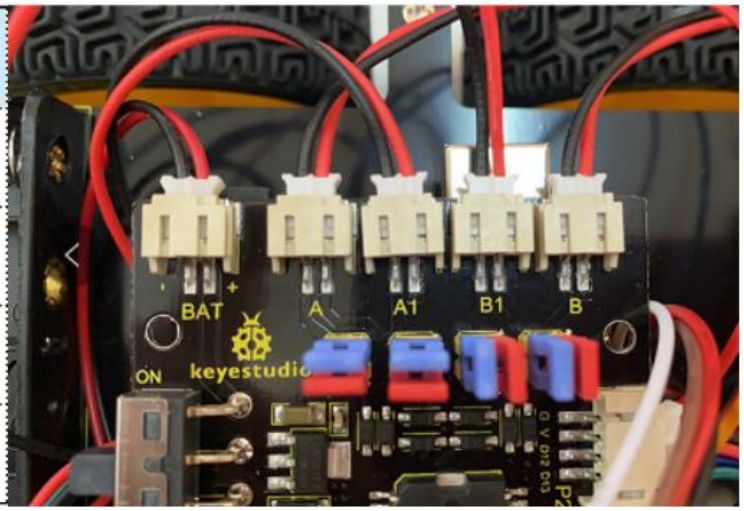




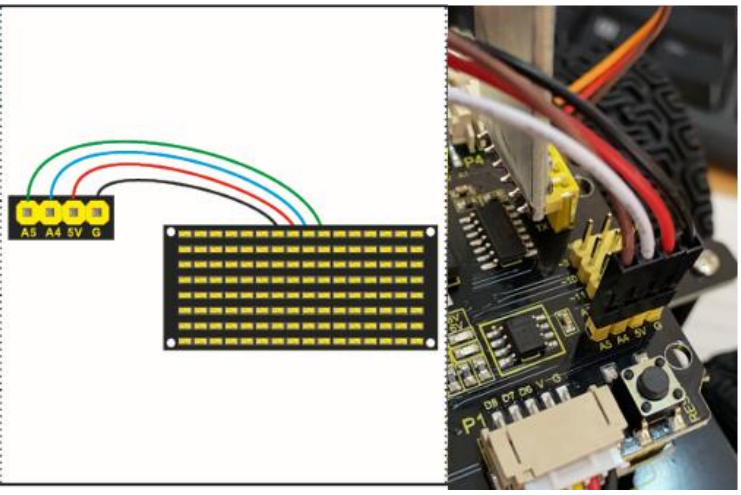
Step 6: Hook-up Guide



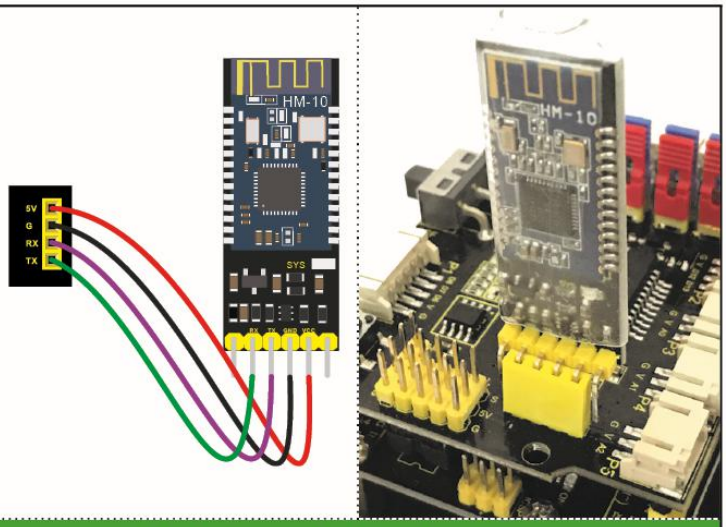
Motor	L298P Shield
M1	B1
M2	B
M3	A
M4	A1



LED Panel	L298P Shield
GND	G
VCC	5V
SDA	A4
SCL	A5

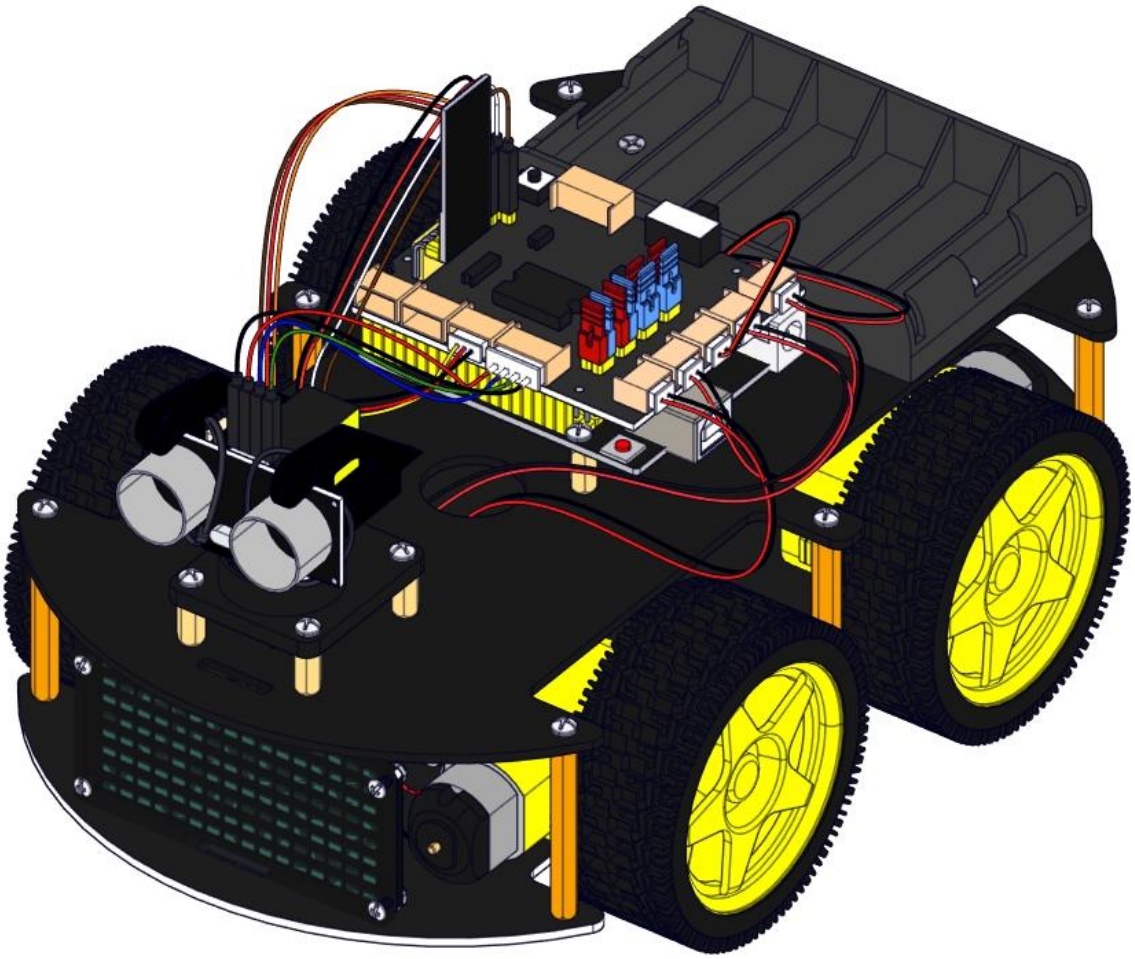


Bluetooth	L298P Shield
RXD	TX
TXD	RX
GND	G
VCC	5V



Note: Remove the Bluetooth module before uploading the program

Servo	L298P Shield
Brown wire	G
Red wire	5V
Orange wire	A3



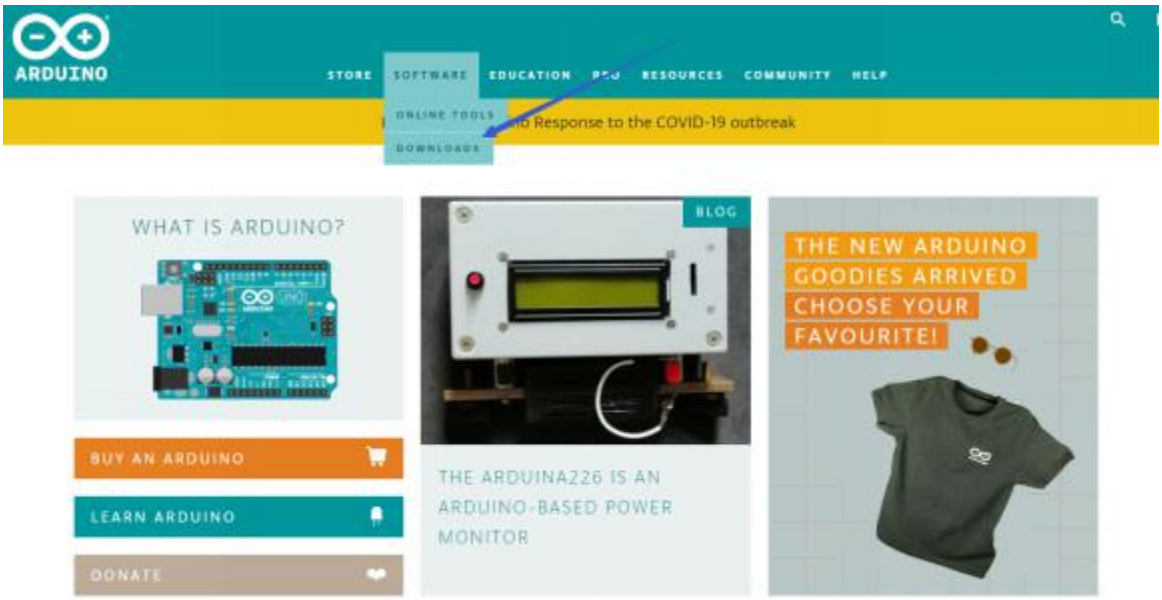
Note: The experiment you did should be in line with wiring diagram, including about components and wiring method. For example, we supply power with external power in the hook-up diagram, so you also have to use external power rather than USB cable .

6. Install Arduino IDE and Driver

(1) Installing Arduino IDE

When we get control board, we need to download Arduino IDE and driver firstly.

You could download Arduino IDE from the official website: https://www.arduino.cc/, click the **SOFTWARE** on the browse bar, click "DOWNLOADS" to enter download page, as shown below:



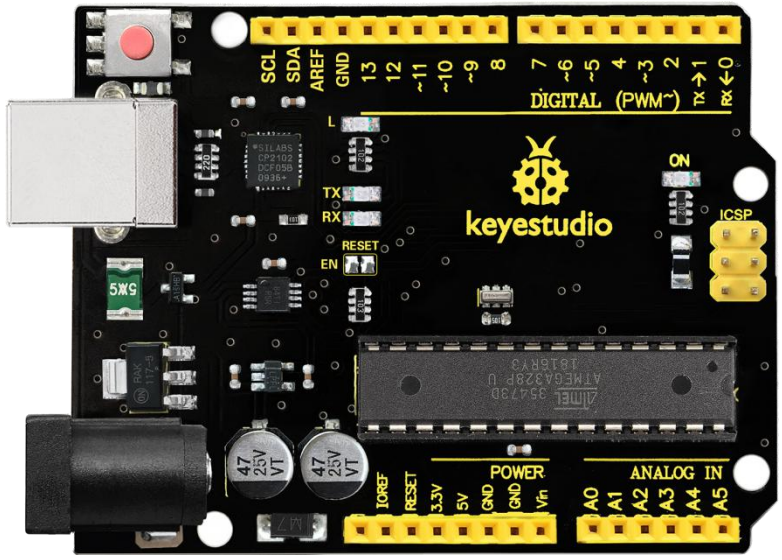
There are various versions Of IDE for Arduino, just download a version that compatible with your system, here we will show you how to download and install the windows version Arduino IDE.

There are two versions of IDE for WINDOWS system, you can choose between the installer (.exe) and the Zip packages. For installer, it can be directly downloaded, without the need of installing manually. For Zip package, you need to install the drivers manually.

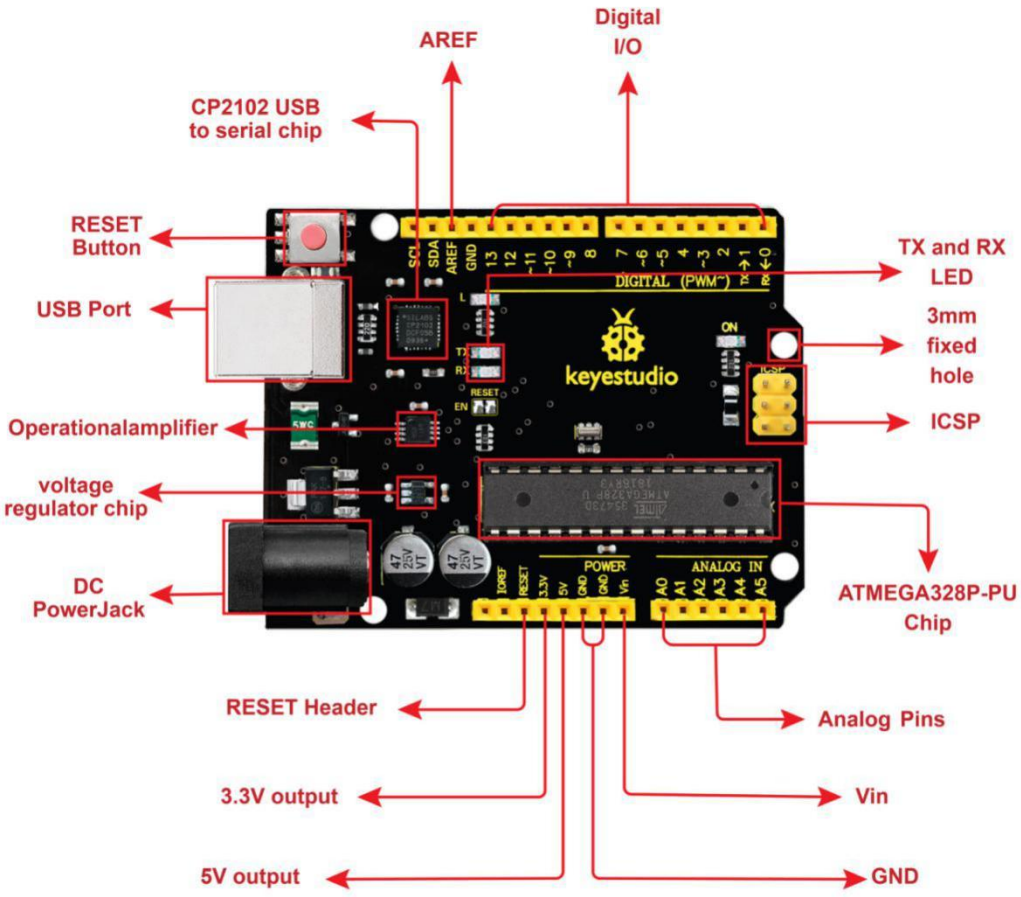
You just need to click JUST DOWNLOAD.

(2) Keyestudio V4.0 Development Board

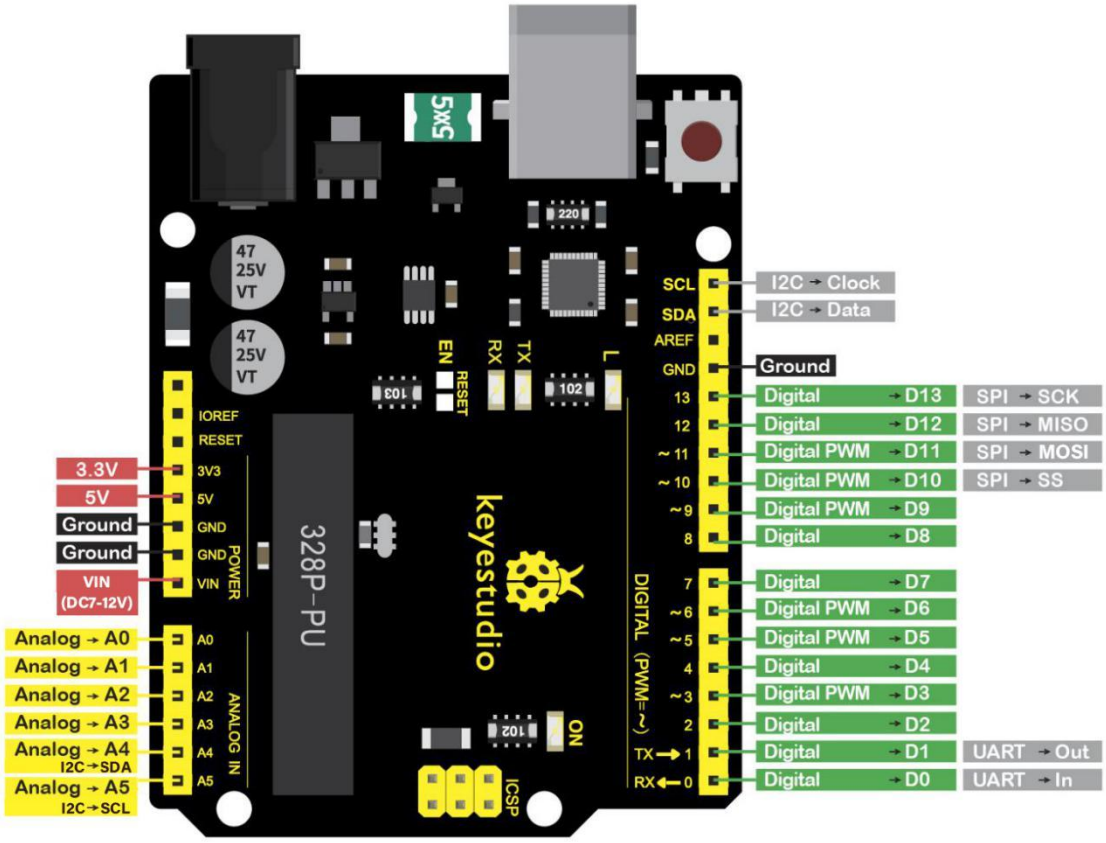
We need to know keyestudio V4.0 development board, as a core of this smart car.



keyestudio V4.0 development board is based on ATmega328P MCU, and with a cp2102 Chip as a UART-to-USB converter.



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



It contains everything needed to support the micro controller, simply connect it to a computer with a USB cable or power it via an external DC p ower jack (DC 7-12V) or via female headers Vin/ GND(DC 7-12V) to get started.

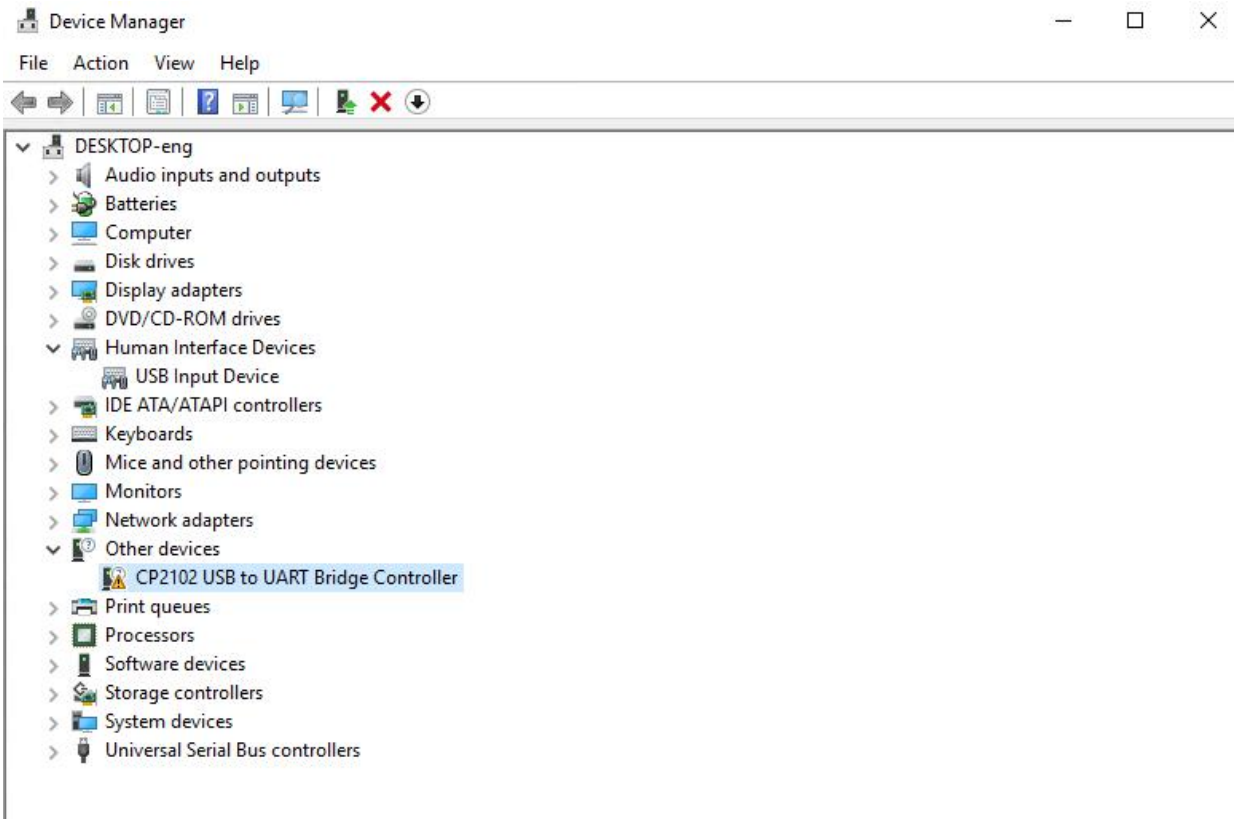
Micro controller	ATmega328P-PU
Operating Voltage	5V
Input Voltage (recommended)	DC7-12V
Digital I/O Pins	14 (D0-D13) (of which 6 provide PWM output)
PWM Digital I/O Pins	6 (D3, D5, D6, D9, D10, D11)
Analog Input Pins	6 (A0-A5)
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32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

(3) Installing Driver of V4.0 Board

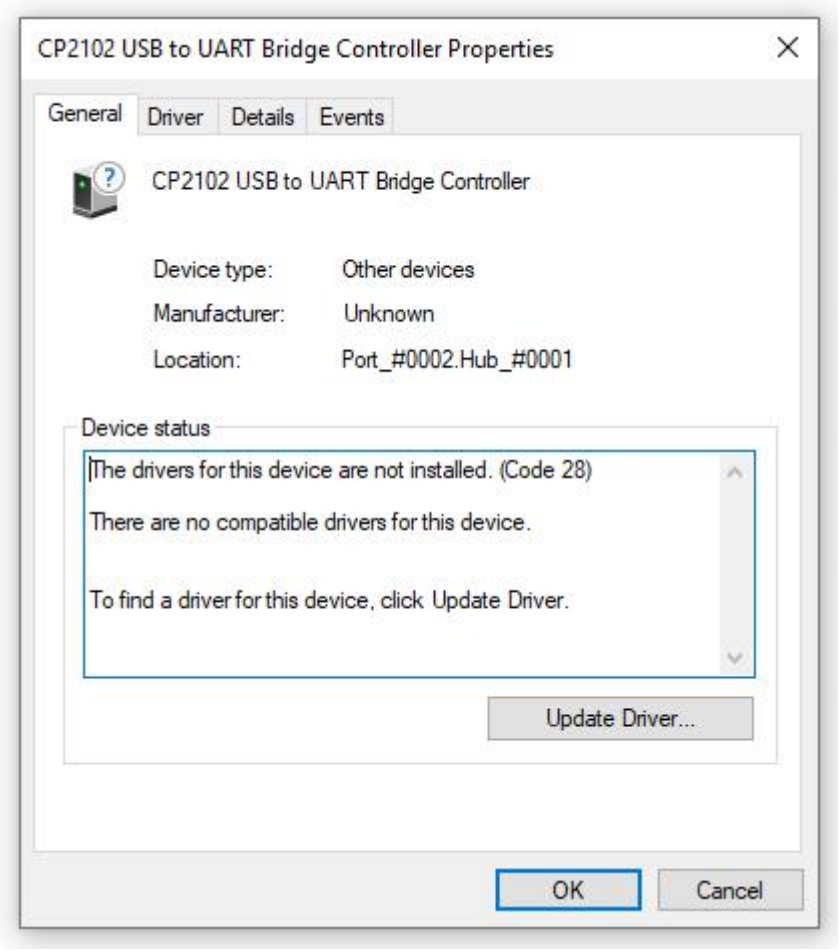
Let’ s install the driver of keyestudio V4.0 board. The USB-TTL chip on V4.0 board adopts CP2102 serial chip. The driver program of this chip is

included in Arduino 1.8 version and above, which is convenient. Plug on USB port of board, the computer can recognize the hardware and automatically install the driver of CP2102.

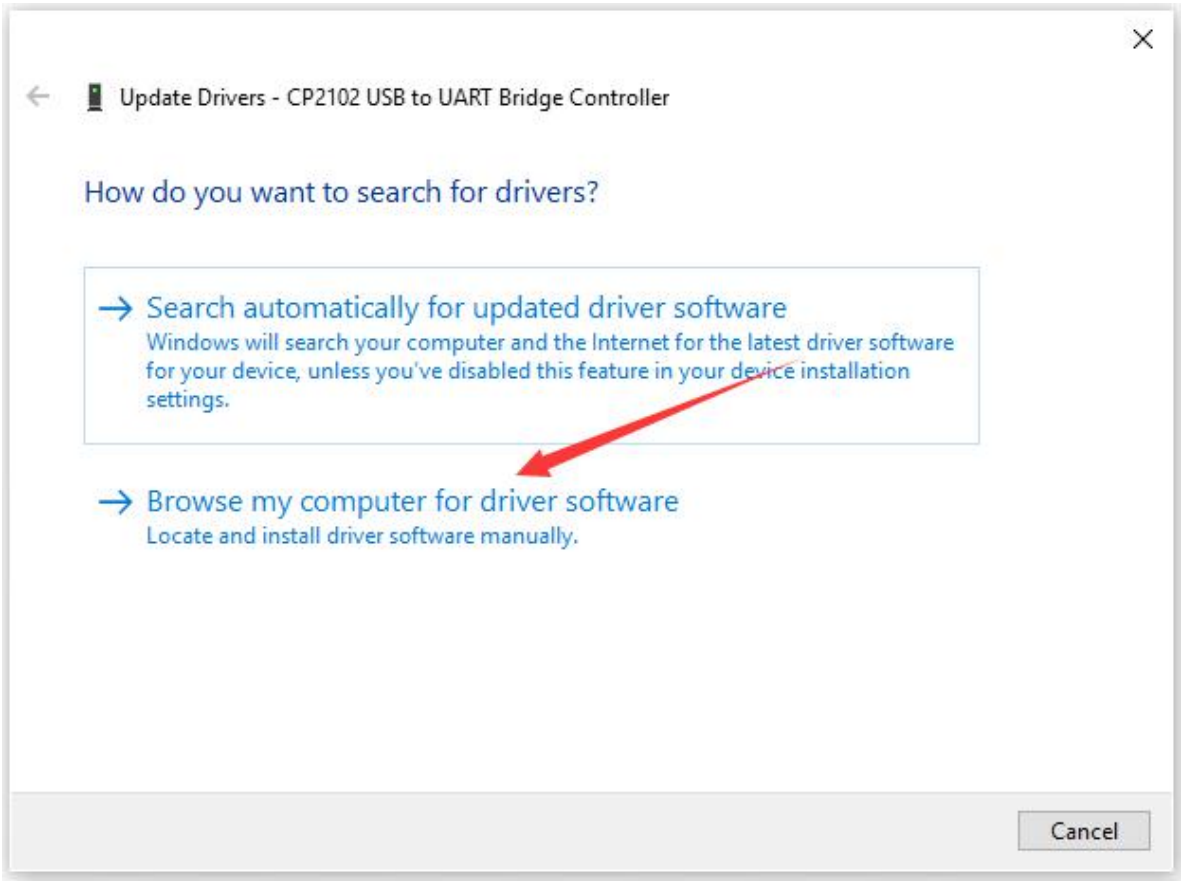
If install unsuccessfully, or you intend to install manually, open the device manager of computer. Right click Computer----- Properties----- Device Manager



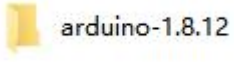
There is a yellow exclamation mark on the page, which implies installing unsuccessfully. Then we double click the hardware and update the driver.



Click "OK" to enter the following page, click "browse my computer for updated driver software", find out the installed or downloaded ARDUINO software. As shown below:

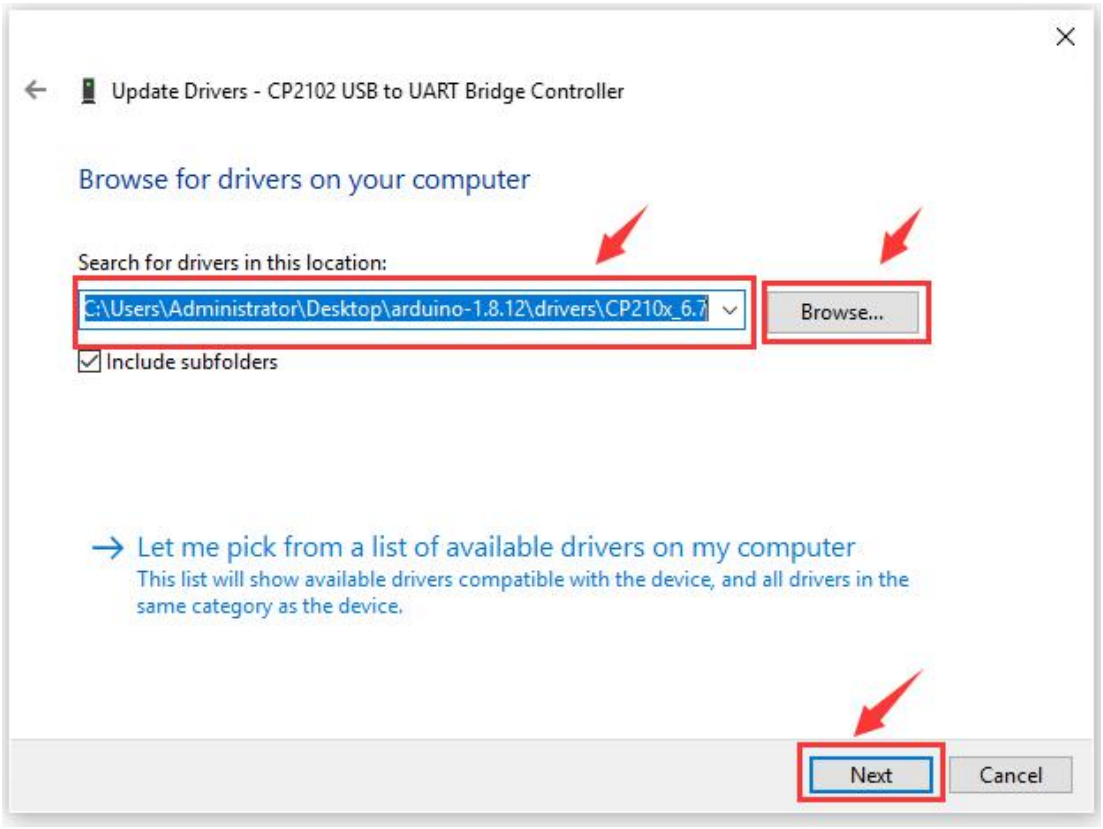


There is a DRIVERS folder in Arduino software installed package

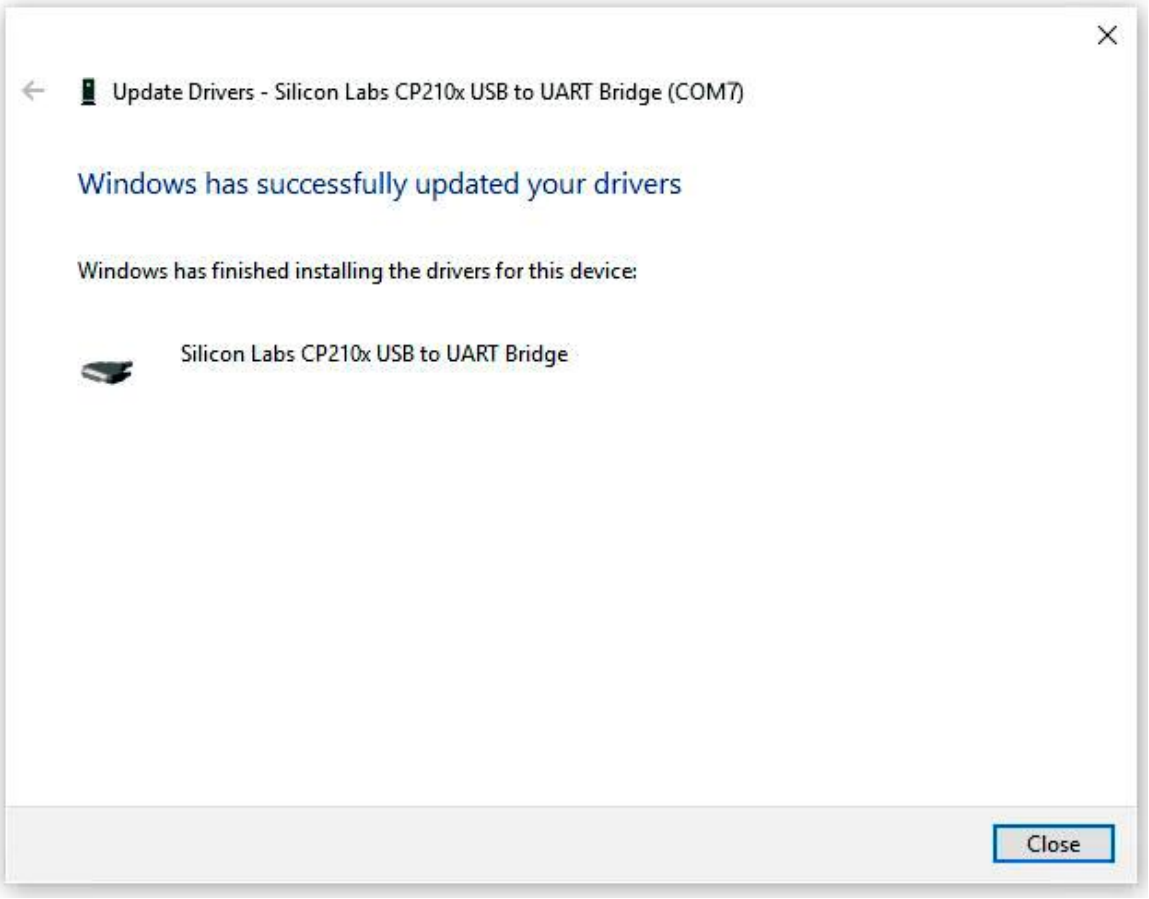
(

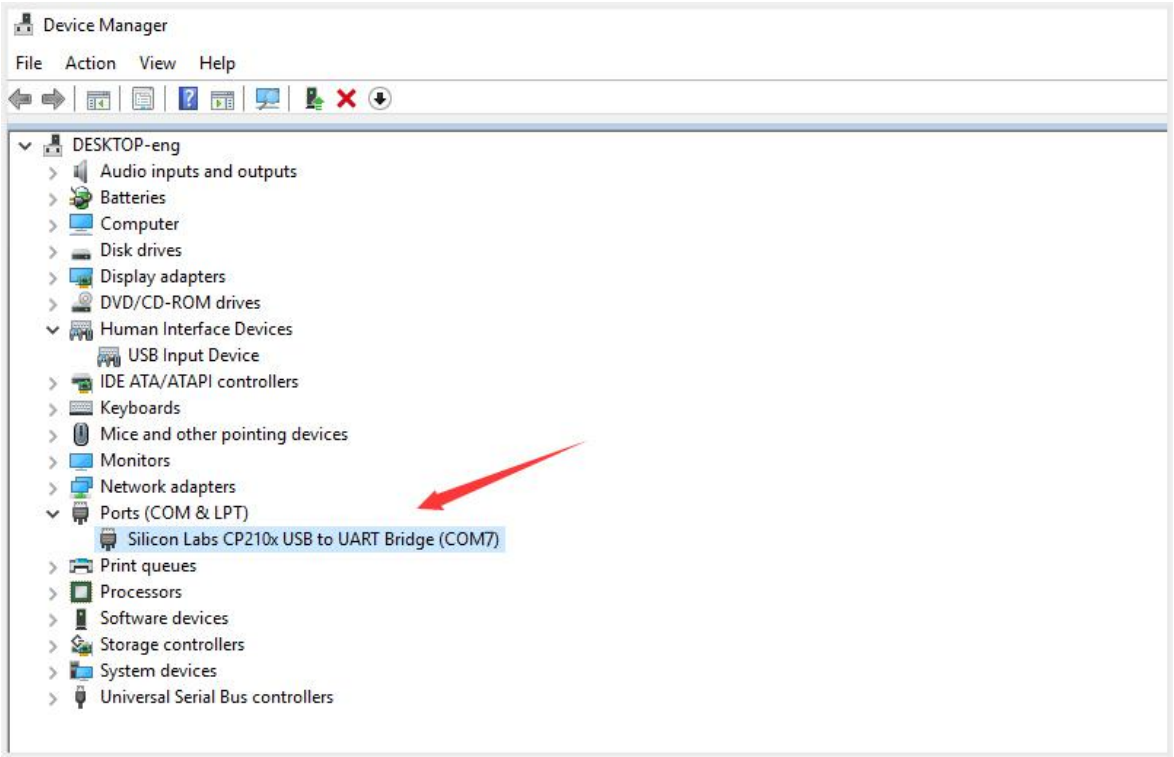
) , open driver folder and you can see the driver of CP210X series chips.

We click “Browse”, then find out the driver folder, or you could enter “driver” to search in rectangular box, then click “next”, the driver will be installed successfully. (I place Arduino software folder on the desktop, you could follow my way)



Open device manager, we will find the yellow exclamation mark disappear. The driver of CP2102 is installed successfully.

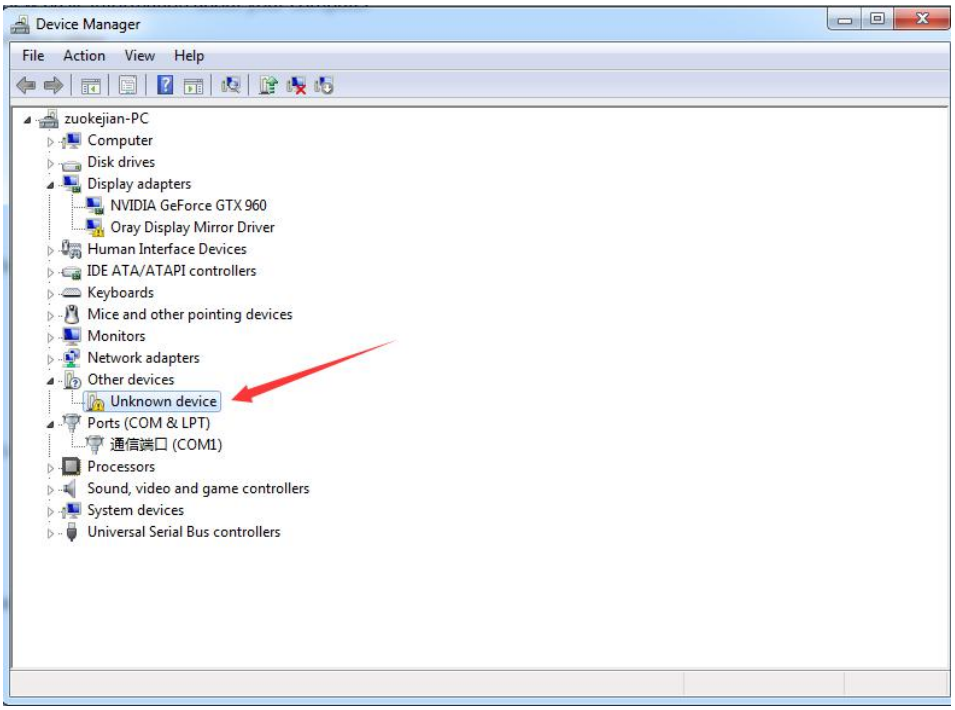

Update Drivers - Silicon Labs CP210x USB to UART Bridge (COM7)
Windows has successfully updated your drivers
Windows has finished installing the drivers for this device:
Silicon Labs CP210x USB to UART Bridge
Close


Device Manager
File Action View Help
DESKTOP-eng
Audio inputs and outputs
Batteries
Computer
Disk drives
Display adapters
DVD/CD-ROM drives
Human Interface Devices
USB Input Device
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
Silicon Labs CP210x USB to UART Bridge (COM7)
Print queues
Processors
Software devices
Storage controllers
System devices
Universal Serial Bus controllers

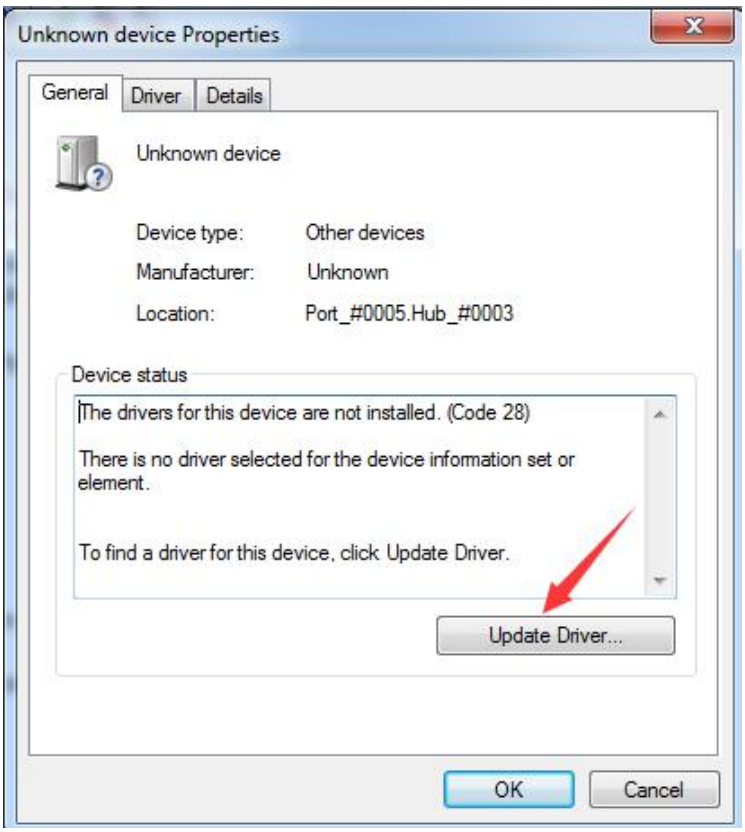
(4) Install other visions of driver

If your development board is Arduino board, install the driver as follows:

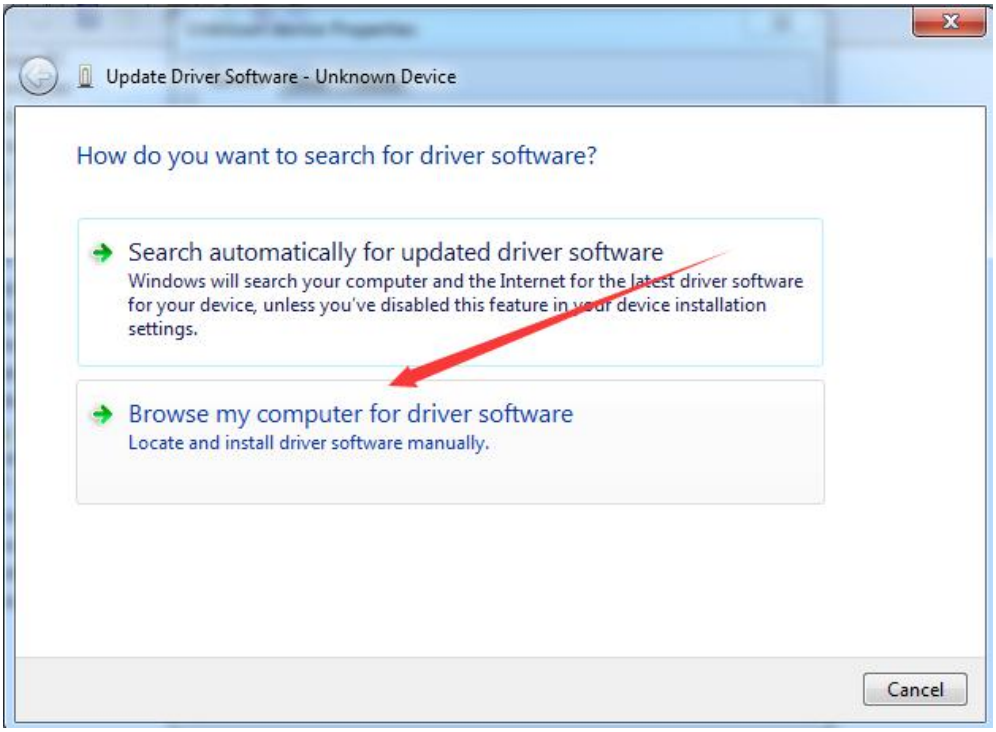
Step 1: Plug in the development board, click Computer----- Properties----- Device Manager, you could see the unknown device is shown.



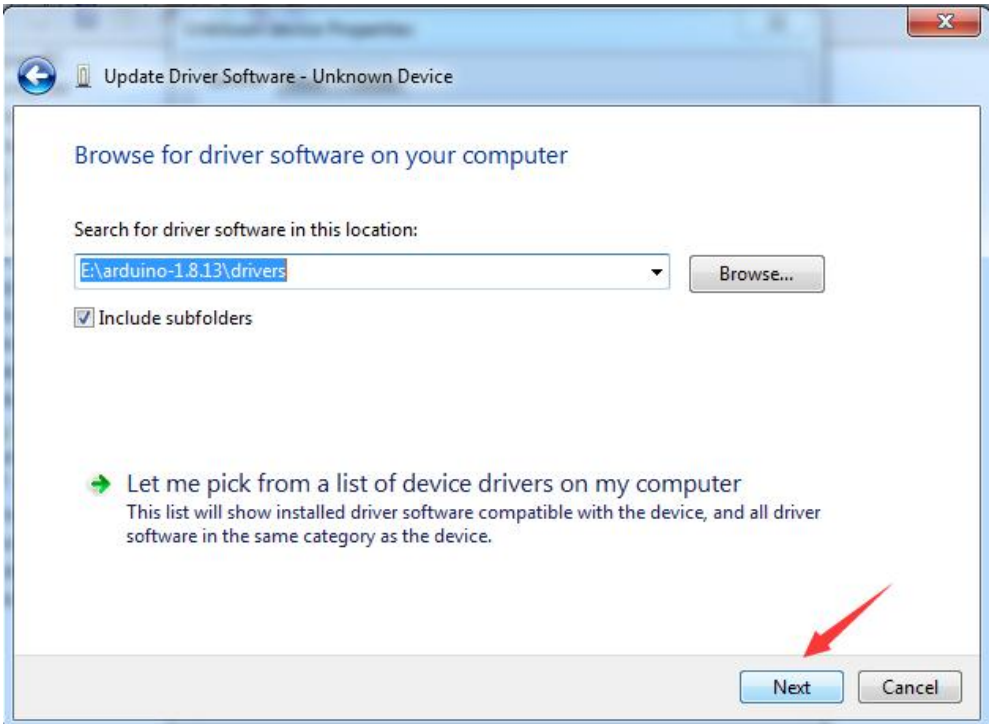
Step 2: Update the driver



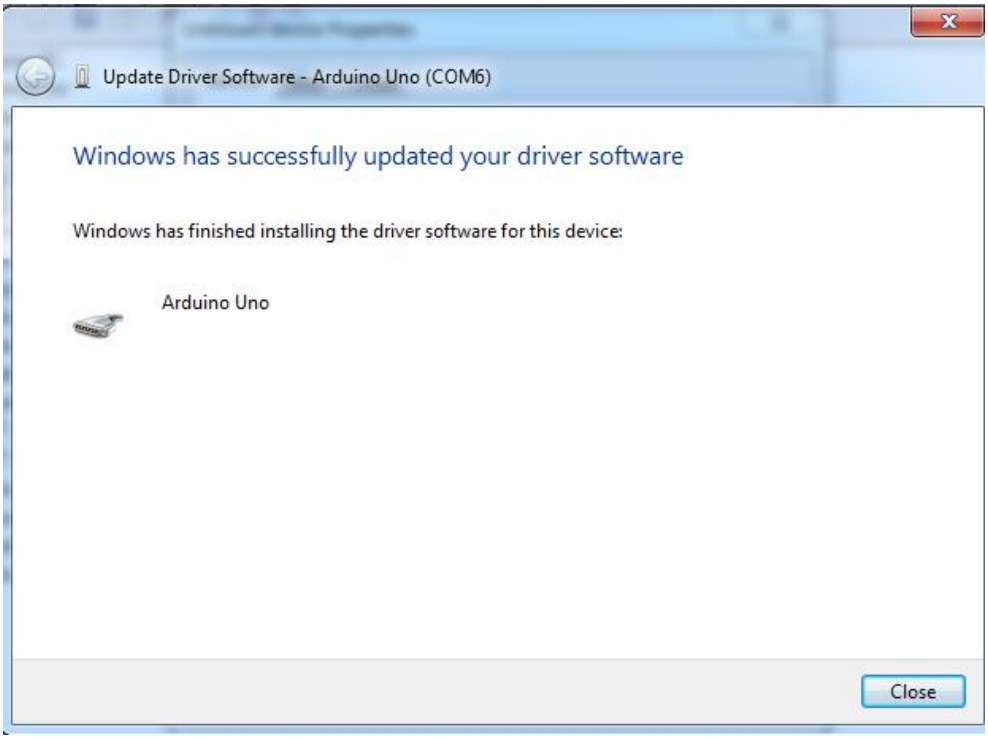
Step 3: click "browse my computer for updated driver software"



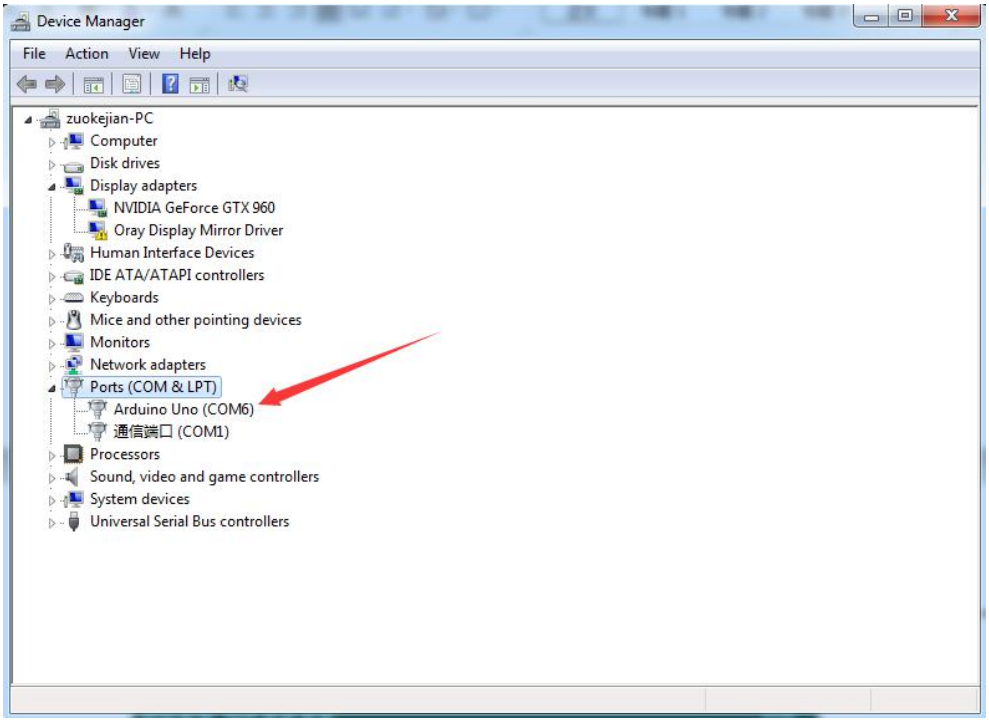
Step 4: find out the folder where the ARDUINO software is installed, click **drivers** folder and tap "Next"



Step 5: the driver is installed successfully.



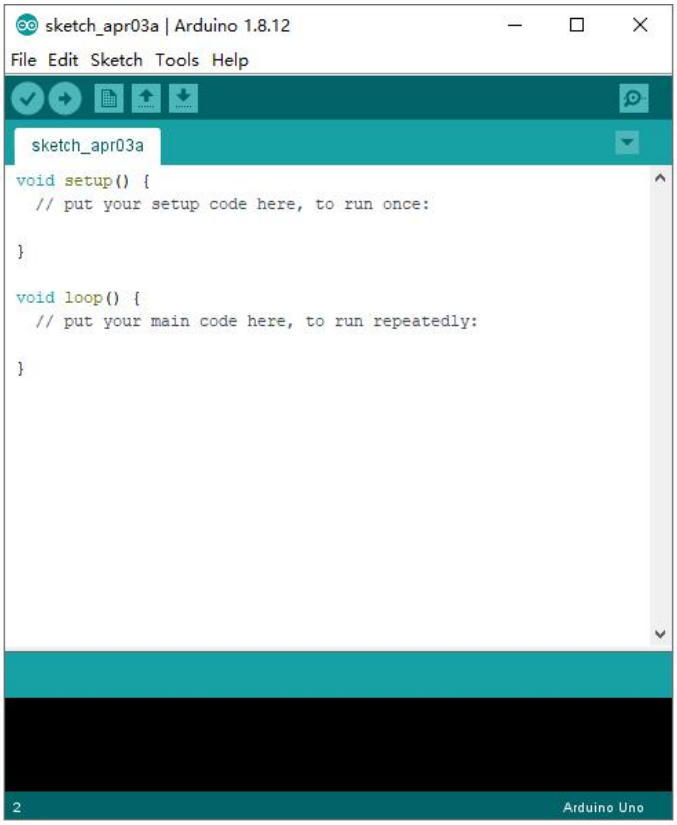
The device manager shows the serial port of Arduino.



(5) Arduino IDE Setting

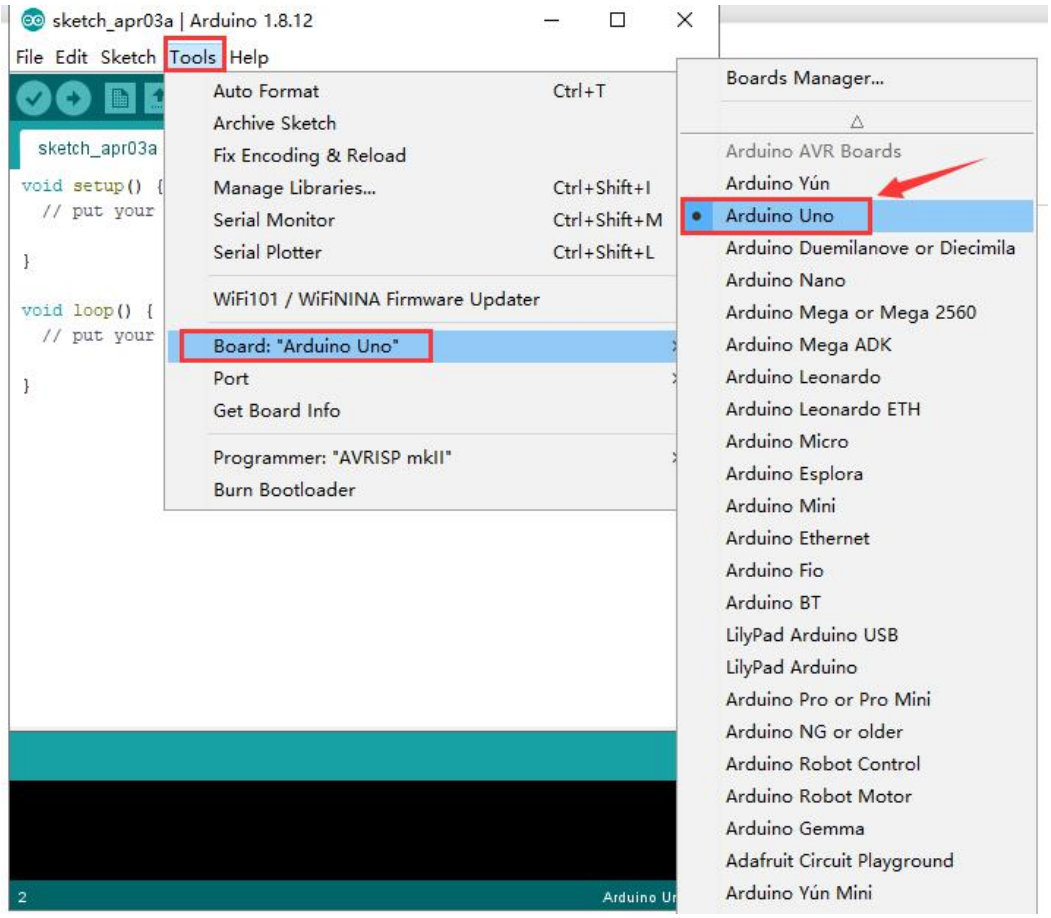
Click 

icon，open Arduino IDE.

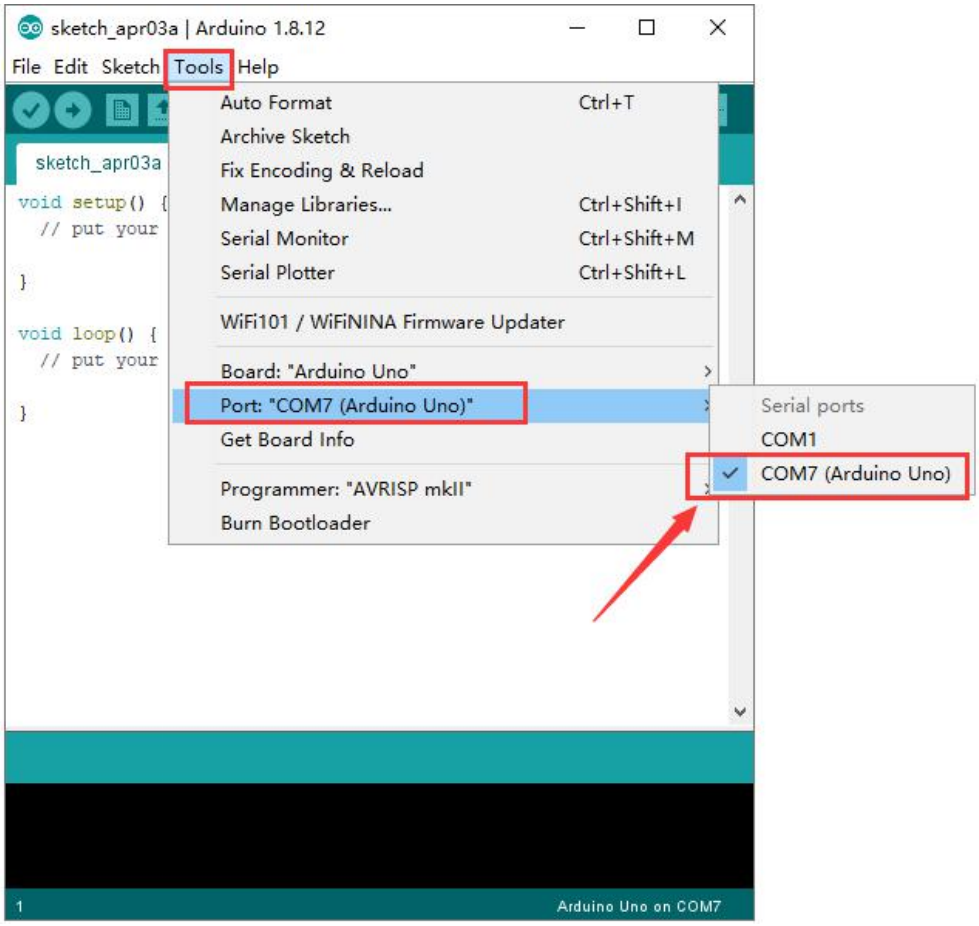


To avoid the errors when uploading the program to the board, you need to select the correct Arduino board that matches the board connected to your computer.

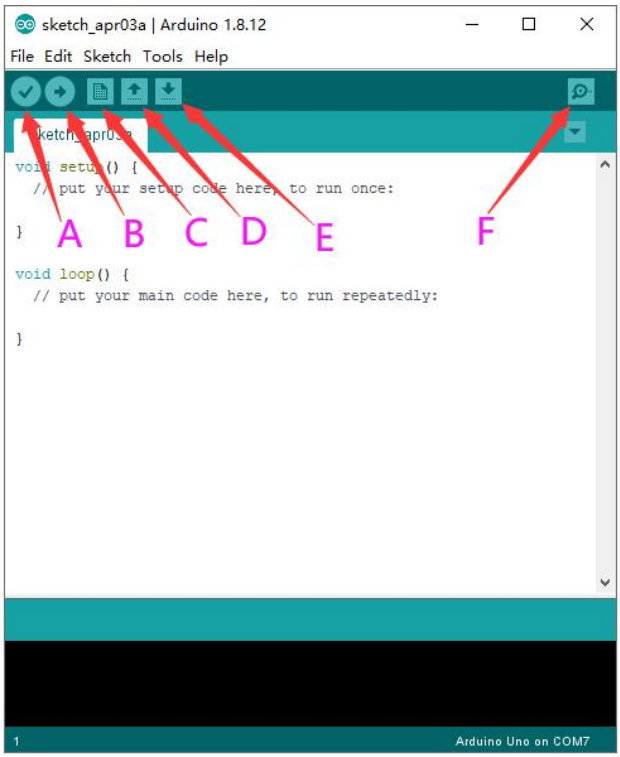
Then come back to the Arduino software, you should click Tools→Board, select the board. (as shown below)



Then select the correct COM port (you can see the corresponding COM port after the driver is successfully installed)



Before uploading the program to the board, let’ s demonstrate the function of each symbol in the Arduino IDE toolbar.



A- Used to verify whether there is any compiling mistakes or not.

B- Used to upload the sketch to your Arduino board.

C- Used to create shortcut window of a new sketch.

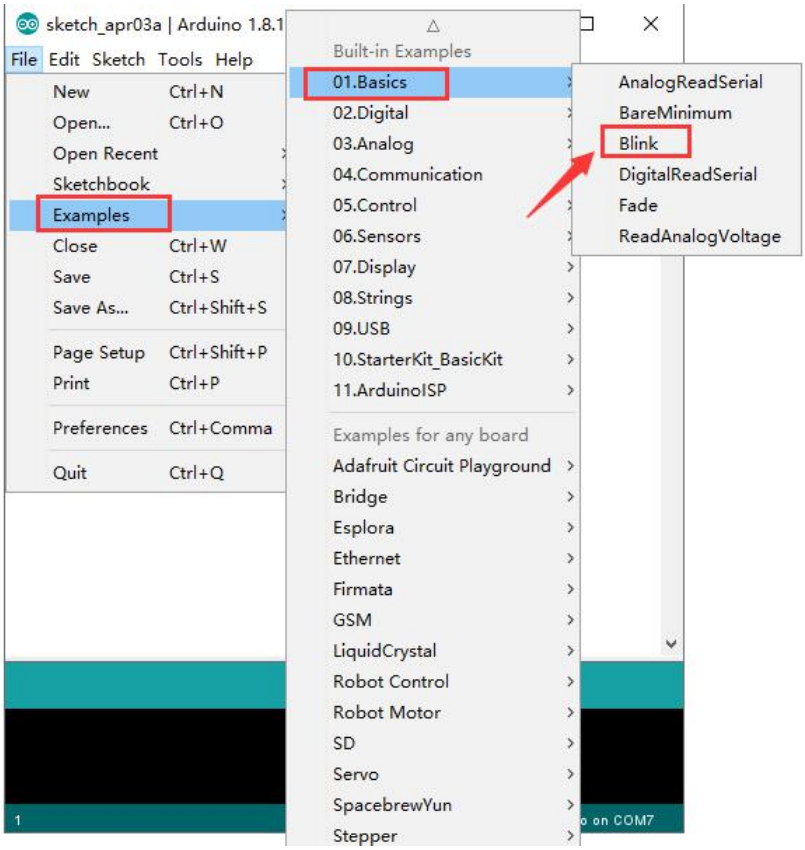
D- Used to directly open an example sketch.

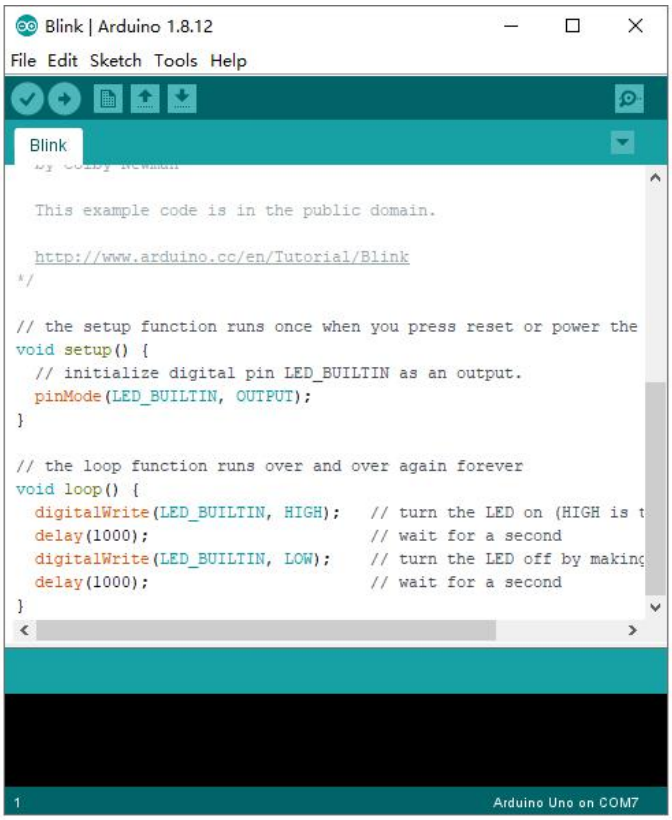
E- Used to save the sketch.

F- Used to send the serial data received from board to the serial monitor.

(6) Start First Program

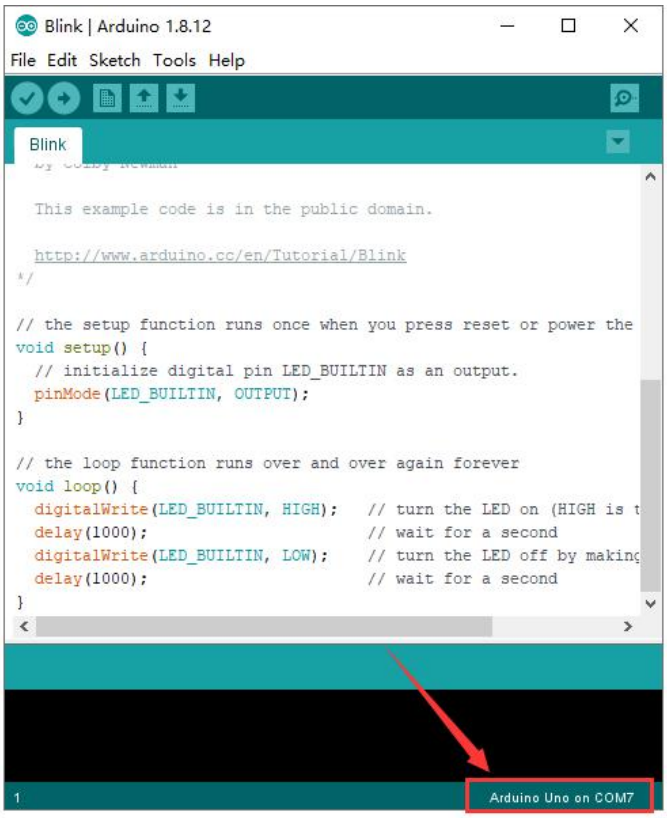
Open the file to select Example, choose BLINK from BASIC, as shown below:



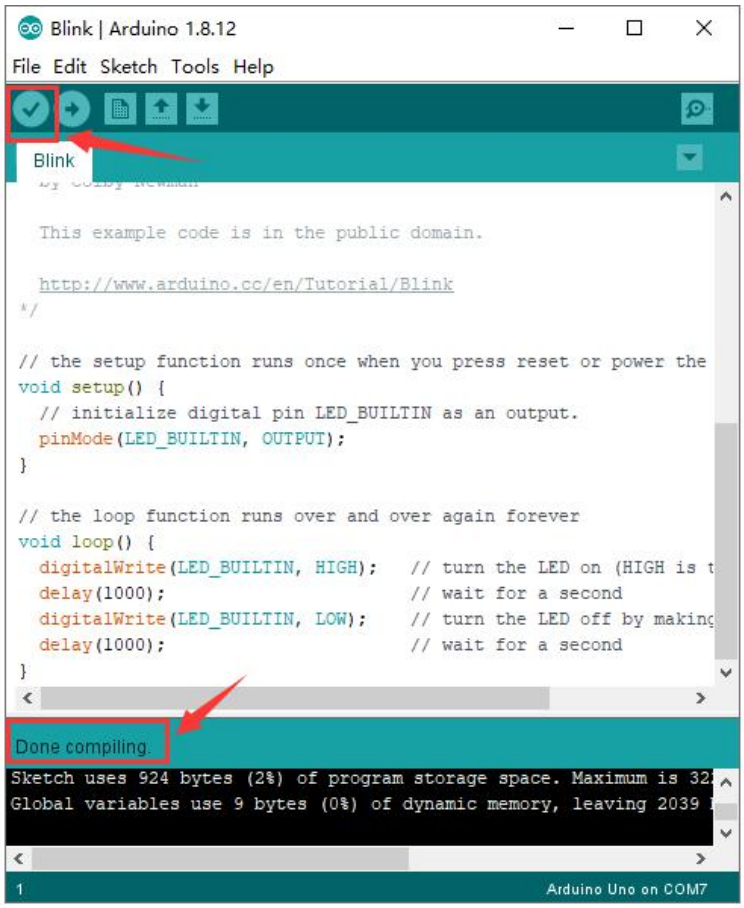


Set board and COM port, the corresponding board and COM port are

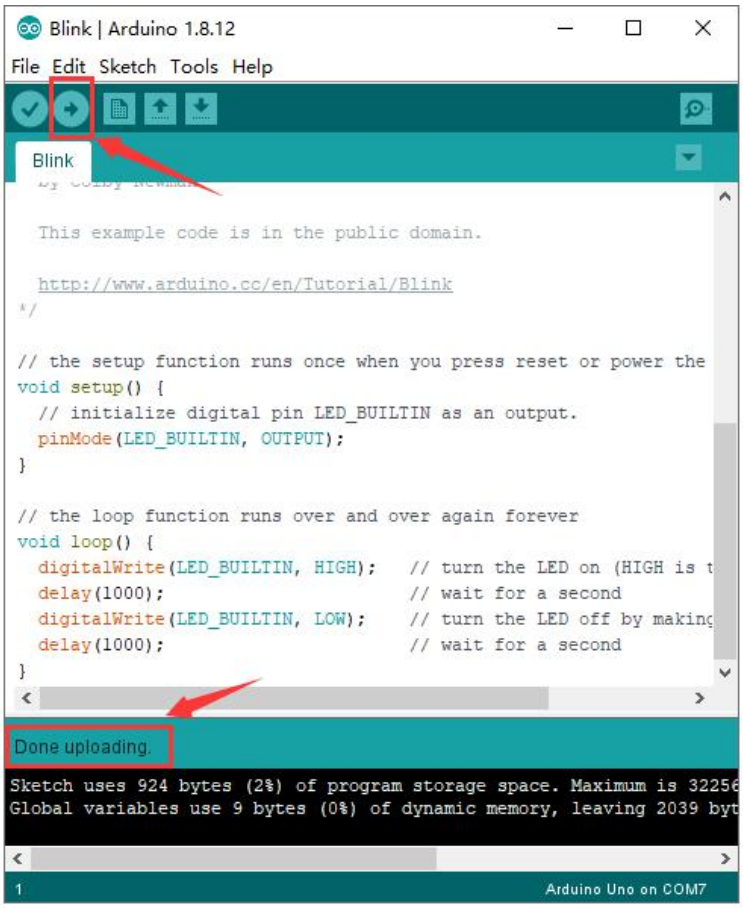
shown on the lower right of IDE.



Click  to start compiling the program, and check errors.



Click to upload the program, upload successfully.



Upload the program successfully, the onboard LED lights on for 1s, lights off for 1s. Congratulation, you finish the first program.

7. How to Add a Library?

(1) What are Libraries ?

Libraries are a collection of code that makes it easy for you to connect to a sensor,display, module, etc.

For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

The built-in libraries and some of these additional libraries are listed in the

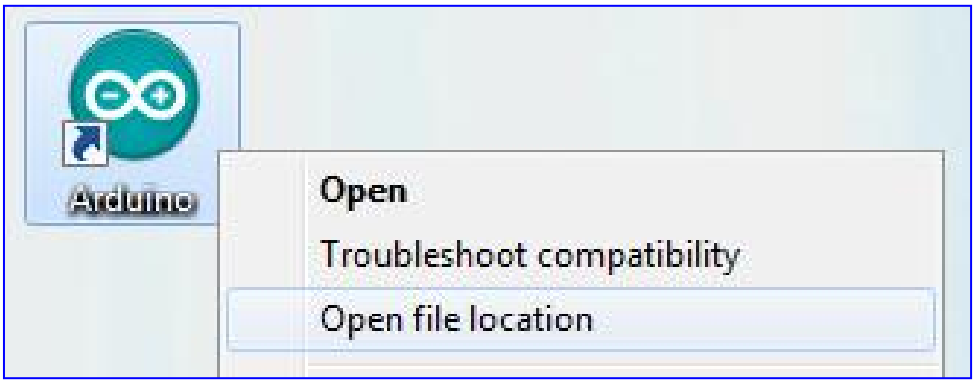
reference.

(2) How to Install a Library ?

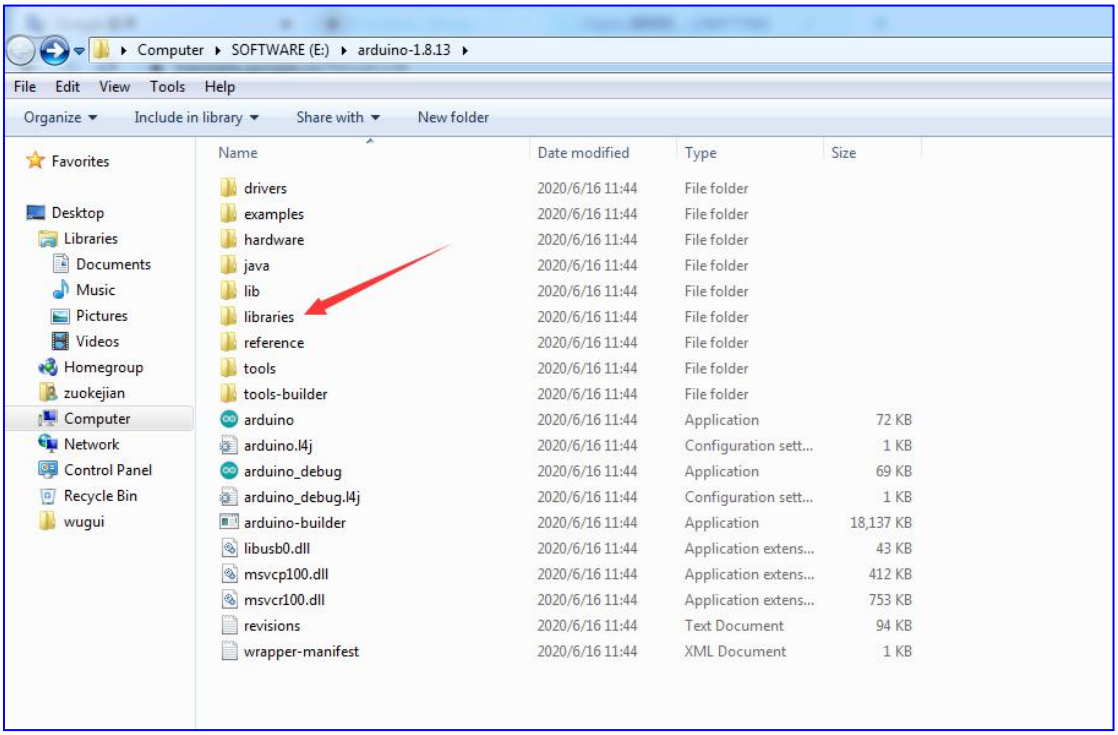
Here we will introduce the most simple way for you to add libraries .

Step 1 : After downloading well the Arduino IDE, you can right-click the icon of Arduino IDE.

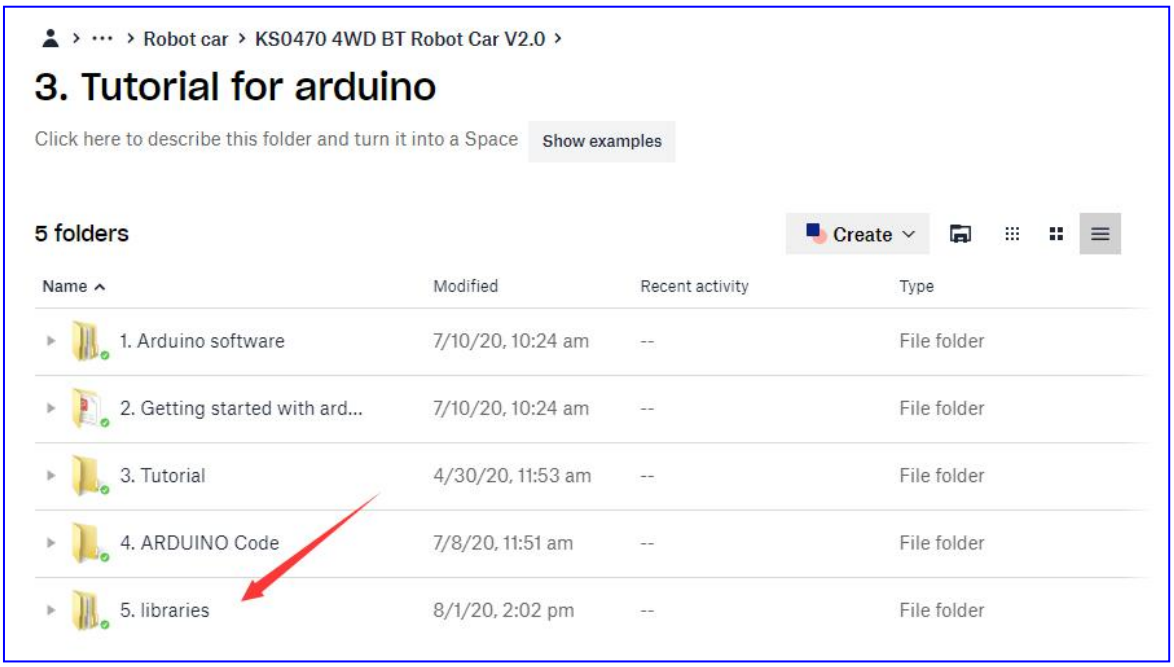
Find the option "Open file location" shown as below:

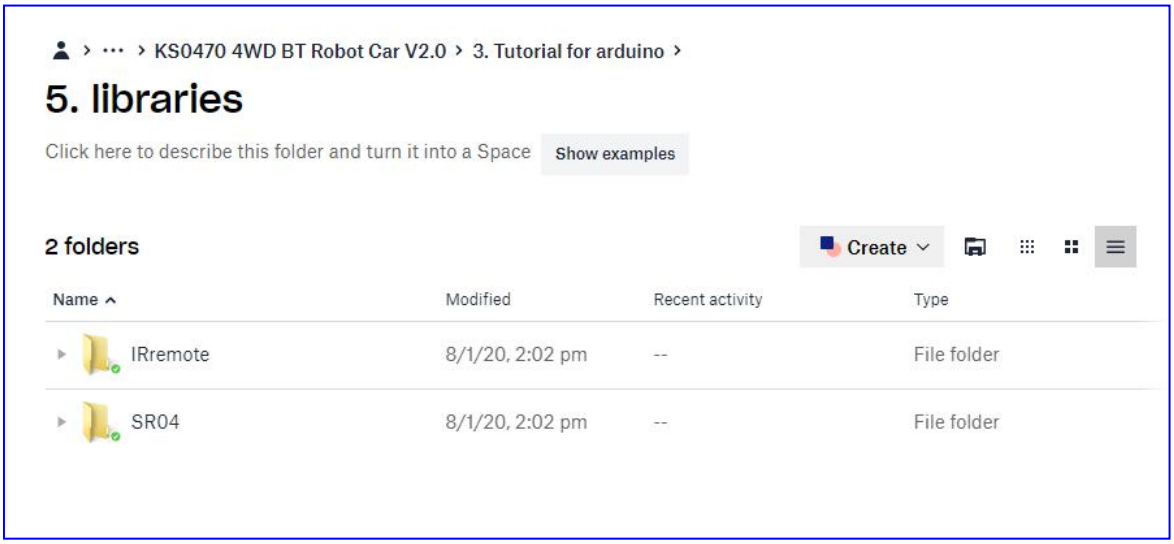


Step 2: Enter it to find out libraries folder, this folder is the library file of Arduino.



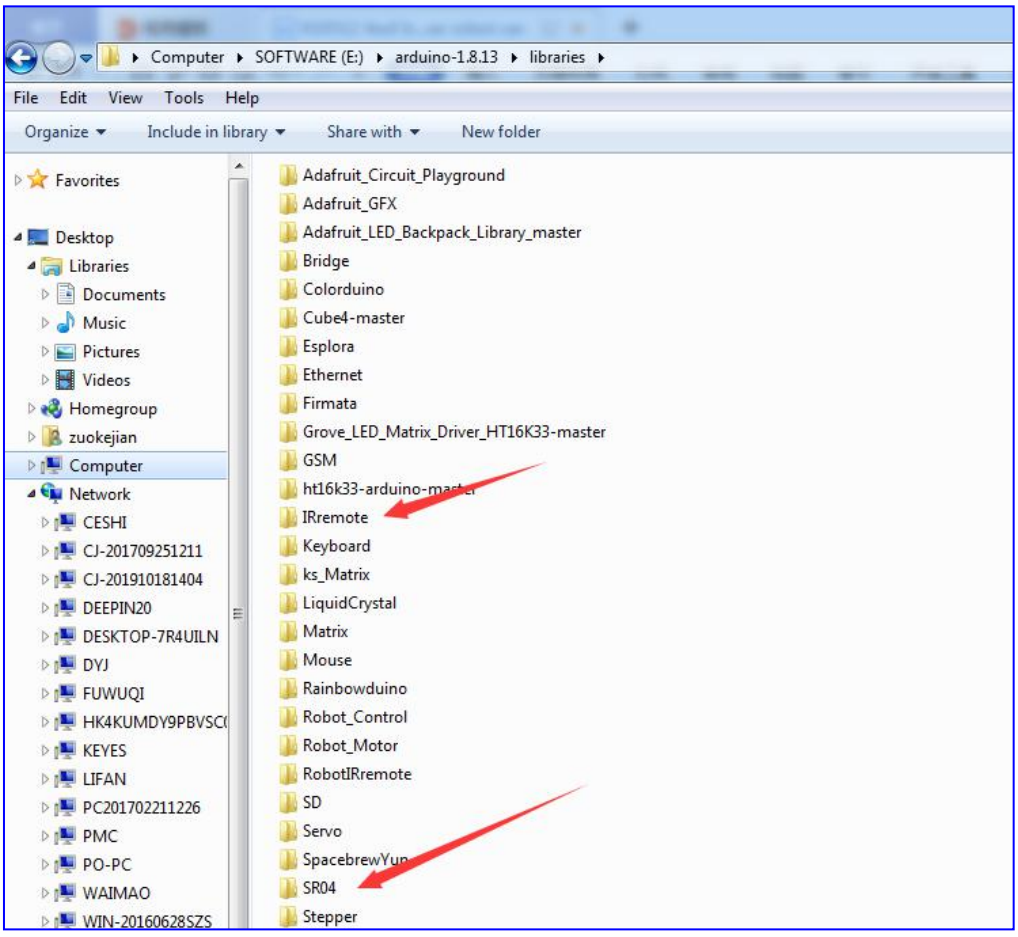
Step 3: Next to find out the "libraries" folder of 4WD robot car(seen in the link: https://fs.keyestudio.com/KS0470)





You just need to replicate and paste IRremove and SR04 folders into the libraries folder of Arduino IDE.

Then the libraries of 4wd robot car are installed successfully, as shown below:

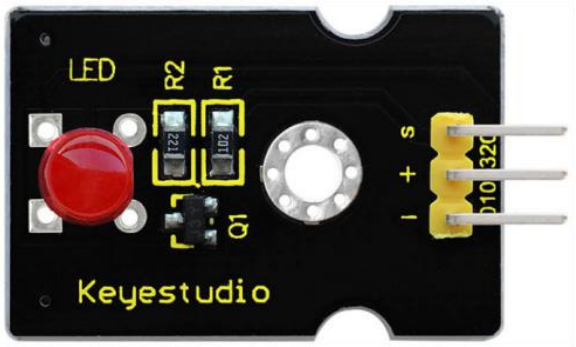


8. Projects

The whole project begins with basic program. Starting from simple to complex, the lessons will guide you to assemble robot car and absorb the knowledge of electronic and machinery step by step. I reckon that you could hardly sit still and itch to have a go, let’ s get started.

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

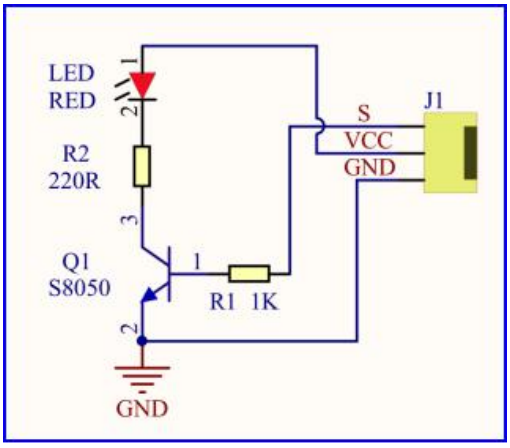
Project 1: LED Blink



(1) Description

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

(2) Specification



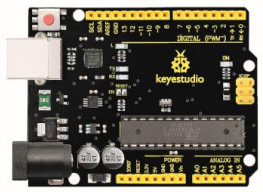
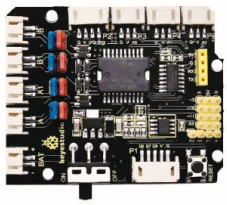
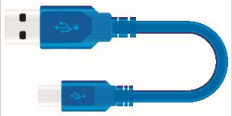
Control interface: digital port

Working voltage: DC 3.3-5V

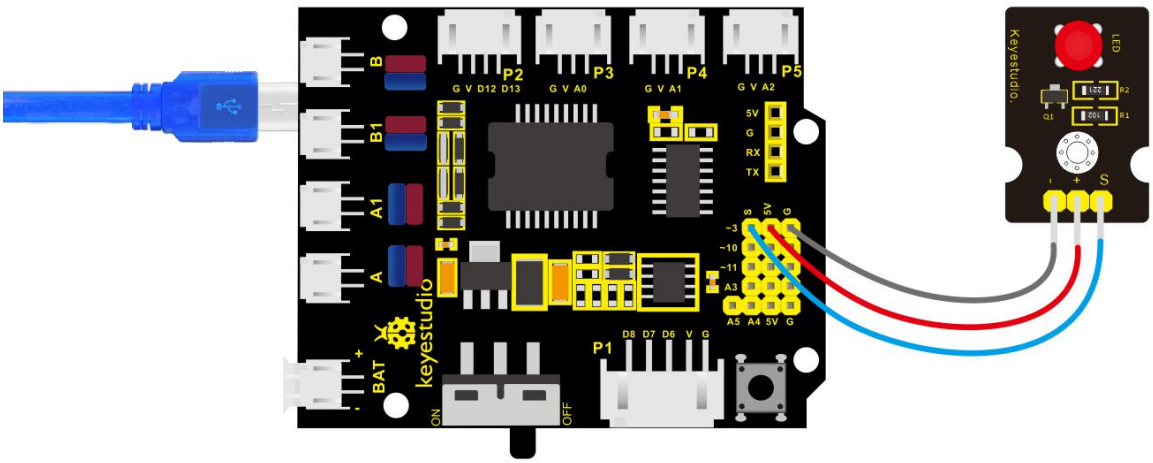
Pin spacing: 2.54mm

LED display color: red

(3) What You Need

Control Board *1	L298P Motor Shield*1	LED module *1	USB Cable *1	3pin Dupont line *1
				

(4) Wiring Diagram



The expansion board is stacked on development board, LED module is connected to G of shield, “+” is linked with 5V, S end is attached to D3.

(5) Test Code:

```
/*
 keyestudio 4wd BT Car V2
 lesson 1.1
 Blink
 http://www.keyestudio.com
*/
void setup()
 {
     pinMode(3, OUTPUT);// initialize digital pin 3 as an output.
}
void loop() // the loop function runs over and over again forever
{   digitalWrite(3, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(3, LOW); // turn the LED off by making the voltage LOW
    delay(1000); // wait for a second
}//*****************************************************************
```

(6) Test Result

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

(7) Code Explanation

pinMode(3， OUTPUT) - This function can denote that the pin is INPUT or OUTPUT

digitalWrite(3，HIGH) - When pin is OUTPUT, we can set it to HIGH(output 5V) or LOW(output 0V)

(8) Extension Practice

We succeed to blink LED. Next, let’ s observe what LED will change if we modify pins and delay time.

```
/*
 keyestudio 4wd BT Car V2
 lesson 1.2
 delay
 http://www.keyestudio.com
*/
void setup() {   // initialize digital pin 11 as an output.
    pinMode(3, OUTPUT);
}
// the loop function runs over and over again forever
void loop()
```

```
{ digitalWrite(3, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(100); // wait for 0.1 second
    digitalWrite(3, LOW); // turn the LED off by making the voltage LOW
    delay(100); // wait for 0.1 second
}//*****************************************************************
```

The LED flashes faster through the test result, therefore, pins and delay time affect flash frequency.

Project 2: Adjust LED Brightness

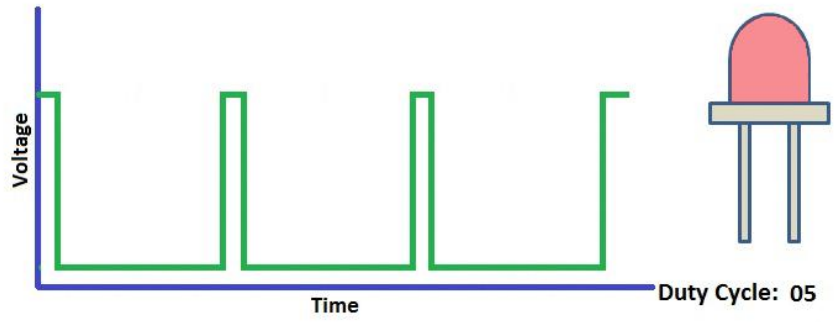
(1) Description

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

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

PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output.In general, the input voltage of port are 0V and 5V. What if the 3V is

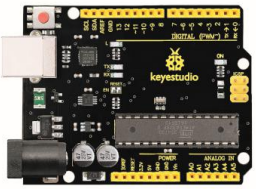
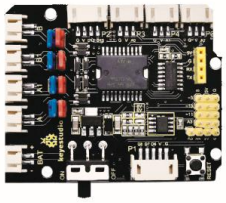
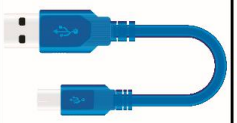
required? Or what if switch among 1V, 3V and 3.5V? We can’ t change resistor constantly. For this situation, we need to control by PWM.



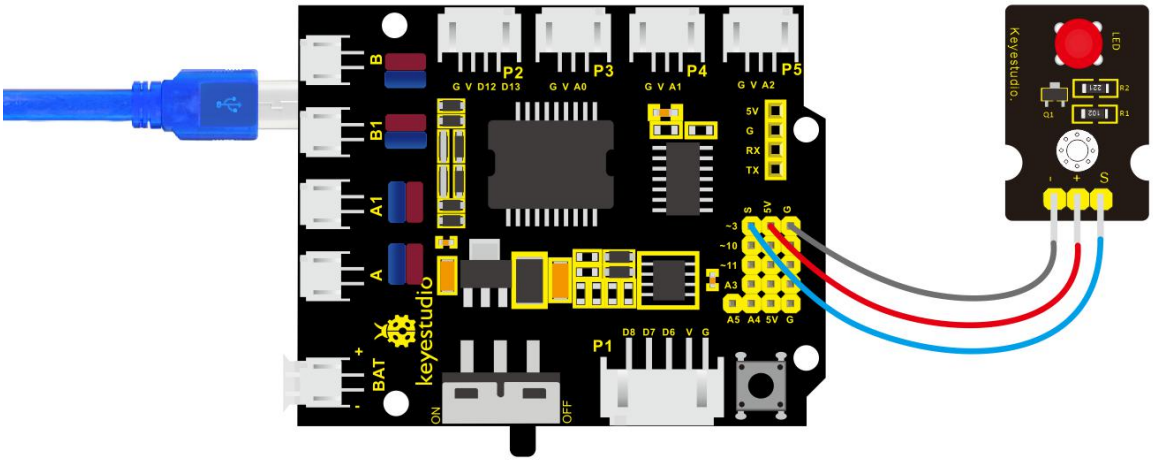
For the Arduino digital port voltage output, there are only LOW and HIGH, which correspond to the voltage output of 0V and 5V. You can define LOW as 0 and HIGH as 1, and let the Arduino output five hundred 0 or 1 signals within 1 second.

If output five hundred 1, that is 5V; if all of which is 1, that is 0V. If output 010101010101 in this way then the output port is 2.5V, which is like showing movie. The movie we watch are not completely continuous. It actually outputs 25 pictures per second. In this case, the human can’ t tell it, neither does PWM. If want different voltage, need to control the ratio of 0 and 1. The more 0,1 signals output per unit time, the more accurately control.

(2) What You Need

Control Board *1	L298P Motor Shield*1	LED module *1	USB Cable *1	3pin Dupont line *1
				

(3) Hook-up Diagram



(4) Test Code:

```
/*
  keyestudio 4wd BT Car V2
```

```
 lesson 2.1
 pwm
 http://www.keyestudio.com
*/
int ledPin = 3; // Define the LED pin at D3
int value;
void setup () {
   pinMode (ledPin, OUTPUT); // initialize ledpin as an output.
}
void loop () {
   for (value = 0; value <255; value = value + 1) {
      analogWrite (ledPin, value); // LED lights gradually light up
      delay (5); // delay 5MS
   }
   for (value = 255; value> 0; value = value-1) {
      analogWrite (ledPin, value); // LED gradually goes out
      delay (5); // delay 5MS
   }
}
```

(5) Test Result

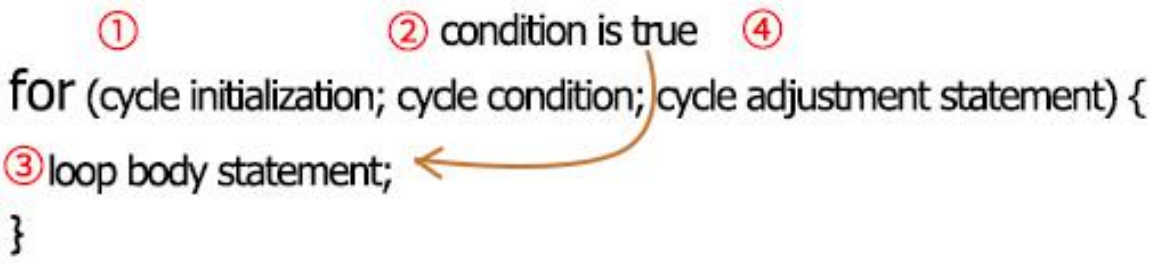
Upload test code successfully, LED gradually becomes brighter then darker,

like human breath, rather than light on and off immediately.

(6) Code Explanation

When we need to repeat some statements, we could use FOR statement.

FOR statement format is shown below:



FOR cyclic sequence:

Round 1: 1 → 2 → 3 → 4

Round 2: 2 → 3 → 4

…

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

After knowing this order, go back to code:

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

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

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

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

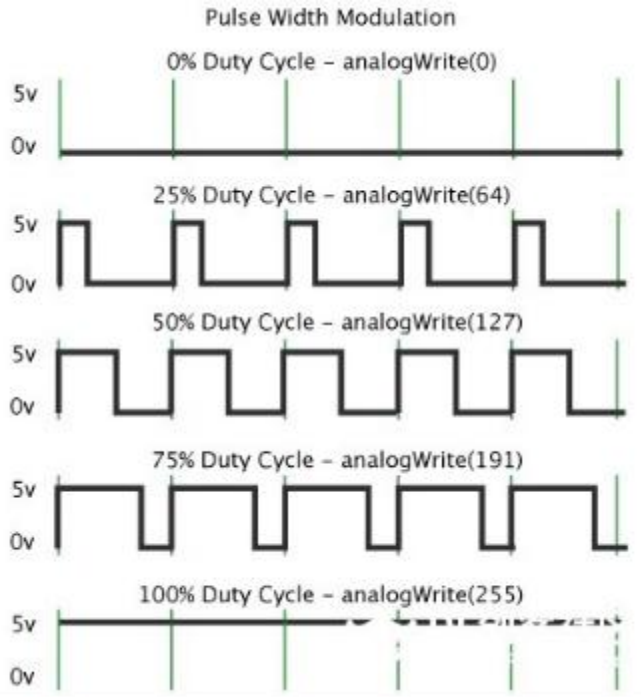
Function format as follows:

analogWrite(pin,value)

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

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

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



In the above figure, the green line represents a period, and value of analogWrite() corresponds to a percentage which is called Duty Cycle as well. Duty cycle implies that high-level duration is divided by low-level duration in a cycle. From top to bottom, the duty cycle of first square wave is 0% and its corresponding value is 0. The LED brightness is lowest, that is, turn off. The more time high level lasts, the brighter the LED. Therefore, the last duty cycle is 100%, which correspond to 255, LED is brightest. 25% means darker.

PWM mostly is used for adjusting the LED brightness or rotation speed of motor.

It plays vital role in controlling smart robot car. I believe that you can’ t wait to enter next project.

(7) Extension Practice:

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

```
/*
  keyestudio 4wd BT Car V2
  lesson 2.2
  pwm
  http://www.keyestudio.com
*/
int ledPin = 3; // Define the LED pin at D3
void setup(){
   pinMode (ledPin, OUTPUT); // initialize ledpin as an output.
}
void loop(){
   for (int value = 0; value <255; value = value + 1){
      analogWrite (ledPin, value); // LED lights gradually light up
      delay (30); // delay 30MS
   }
   for(int value=255; value>0;value=value-1){
      analogWrite (ledPin, value); // LED gradually goes out
```

```
    delay (30); // delay 30MS
  }
}//*********************************************************
```

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

Project 3 : The Working Principle of Line Tracking Sensor

(1) Description:

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

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

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

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

By rotating the adjustable potentiometer on the sensor, it can adjust the detection sensitivity of the sensor.

(2) Specification:

Operating Voltage: 3.3-5V (DC)

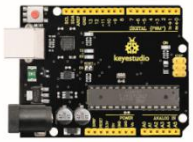
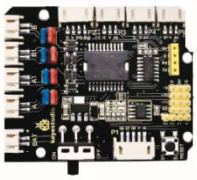
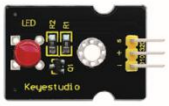
Interface: 5PIN

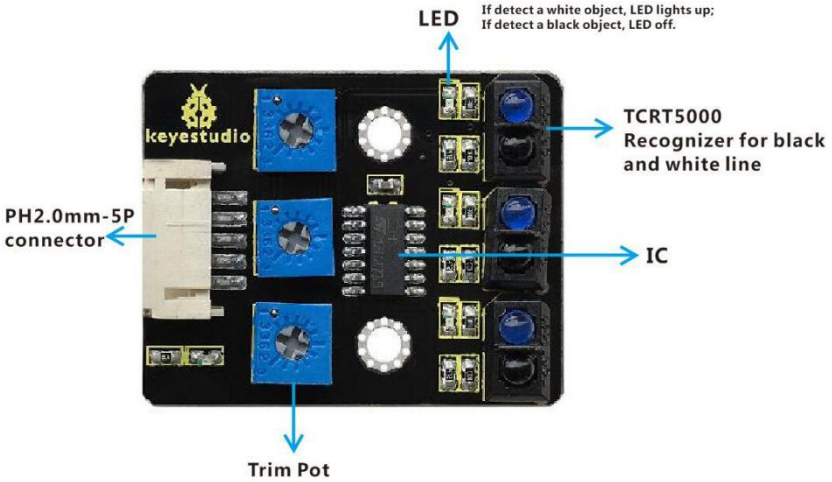
Output Signal: Digital signal

Detection Height: 0-3 cm

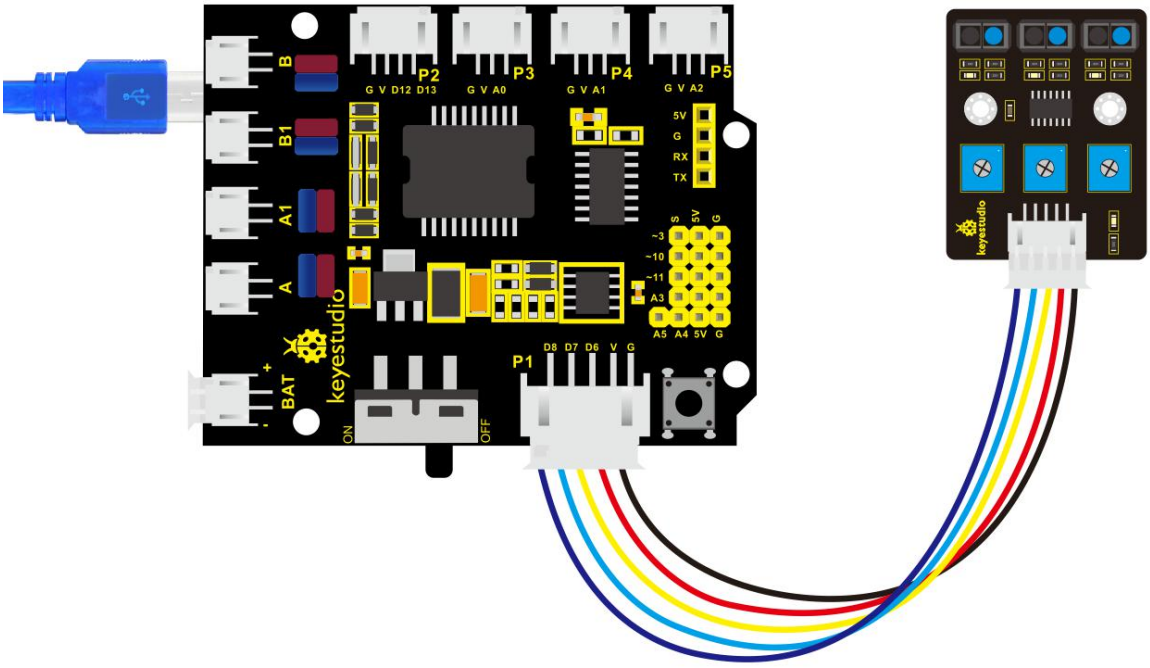
Special note: before testing, turn the potentiometer on the sensor to adjust the detection sensitivity. When adjust the LED at the threshold between ON and OFF, the sensitivity is the best.

(3) What You Need:

Control Board *1	L298P Motor Shield *1	Channel Line Tracking *1	LED module *1	USB Cable *1	5P Double-ended line *1
					 3pin Dupont line *1 



(4) Connection Diagram:



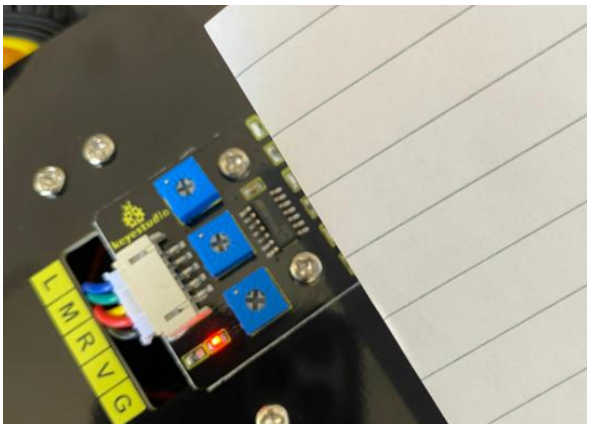
(5) Test Code:

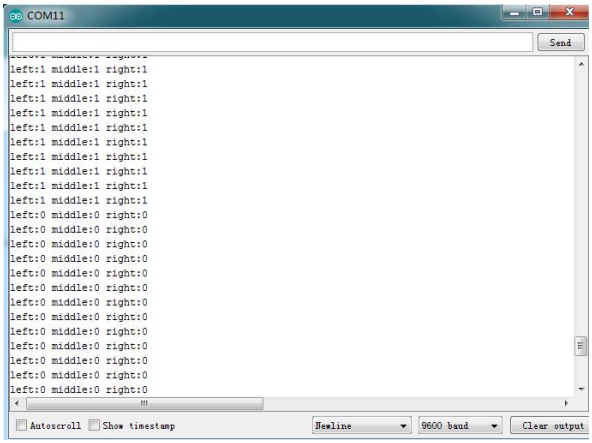
```
/*
keyestudio 4wd BT Car V2
lesson 3.1
  Line Track sensor
  http://www.keyestudio.com
*/
int L_pin = 6;   //pins of left line tracking sensor
int M_pin = 7;   //pins of middle line tracking sensor
int R_pin = 8;   //pins of right line tracking sensor
int val_L,val_R,val_M;// define these variables
```

```
void setup()
{
  Serial.begin(9600); // initialize serial communication at 9600 bits per
second
  pinMode(L_pin,INPUT); // make the L_pin as an input
  pinMode(M_pin,INPUT); // make the M_pin as an input
  pinMode(R_pin,INPUT); // make the R_pin as an input
}
void loop()
{
  val_L = digitalRead(L_pin);//read the L_pin:
  val_R = digitalRead(R_pin);//read the R_pin:
  val_M = digitalRead(M_pin);//read the M_pin:
  Serial.print("left:");
  Serial.print(val_L);
  Serial.print(" middle:");
  Serial.print(val_M);
  Serial.print(" right:");
  Serial.println(val_R);
  delay(500);// delay in between reads for stability
}//**************************************************************************
*
```

(6) Test Result:

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





(7) Code Explanation

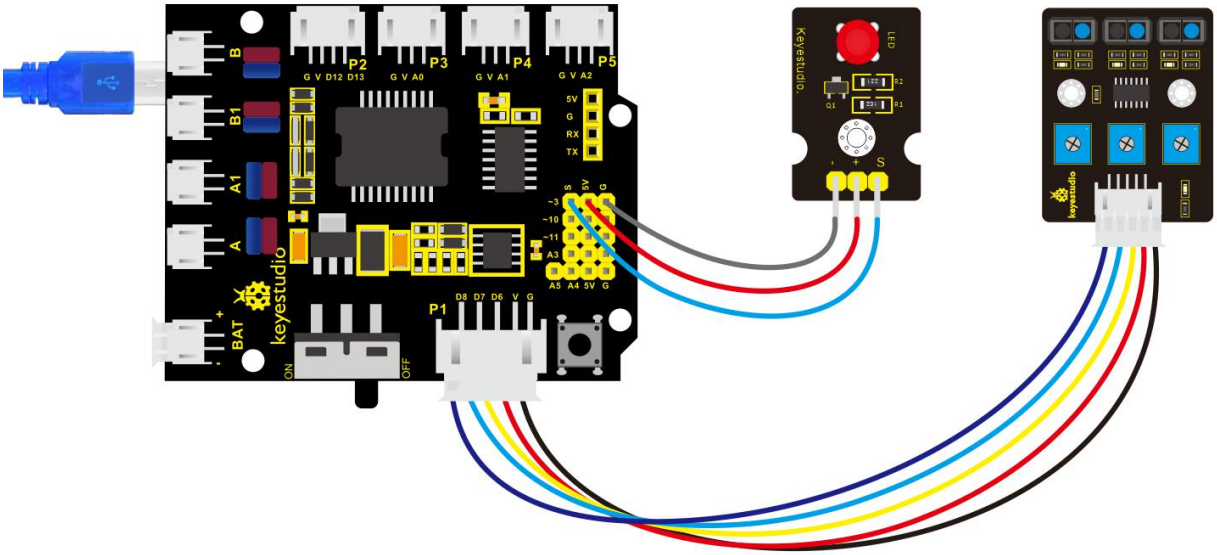
Serial.begin(9600)- Initialize serial port, set baud rate to 9600

pinMode- Define the pin as input or output mode

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

(8) Extension Practice

After knowing its working principle, connect an LED to D3. We could control LED by line tracking sensor.



Test Code

```
/*
keyestudio 4wd BT Car V2
lesson 3.2
  Line Track sensor
  http://www.keyestudio.com
*/
int L_pin = 6;   //pins of left line tracking sensor
int M_pin = 7;   //pins of middle line tracking sensor
int R_pin = 8;   //pins of right line tracking sensor
int val_L,val_R,val_M;// define the variables of three sensors
void setup()
{
   Serial.begin(9600); // initialize serial communication at 9600 bits per
```

```
second
  pinMode(L_pin,INPUT); // make the L_pin as an input
  pinMode(M_pin,INPUT); // make the M_pin as an input
  pinMode(R_pin,INPUT); // make the R_pin as an input
  pinMode(3, OUTPUT);
}
void loop()
{
  val_L = digitalRead(L_pin);//read the L_pin:
  val_R = digitalRead(R_pin);//read the R_pin:
  val_M = digitalRead(M_pin);//read the M_pin:
  Serial.print("left:");
  Serial.print(val_L);
  Serial.print(" middle:");
  Serial.print(val_M);
  Serial.print(" right:");
  Serial.println(val_R);

  if (val_L == HIGH)//if left line tracking sensor detects signals
  {
    digitalWrite(3, LOW);//LED is off
  }
```

```
else//if left line tracking sensor doesn’ t detect signals
{
  digitalWrite(3, HIGH);//LED lights up
  delay(2000);
}

if (val_R == HIGH)//if right line tracking sensor detects signals
{
  digitalWrite(3, LOW);//LED is off
}
else//if right line tracking sensor doesn’ t detect signals
{
  digitalWrite(3, HIGH);//LED lights up
  delay(2000);
}

if (val_M == HIGH)//if middle line tracking sensor detects signals
{
  digitalWrite(3, LOW);//LED is off
}
else//if middle line tracking sensor doesn’ t detect signals
{
```

```
        digitalWrite(3, HIGH);//LED lights up
        delay(2000);
    }
}
//*****************************************************************************
```

Upload the code to development board, we could observe the brightness of LED when covering the line tracking sensor or getting close to it by hand

Project 4: Servo Control

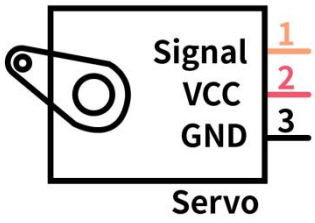
(1) Description

Servo motor is a position control rotary actuator. It mainly consists of housing, circuit board, core-less motor, gear and position sensor. Its working principle is that the servo receives the signal sent by MCU or

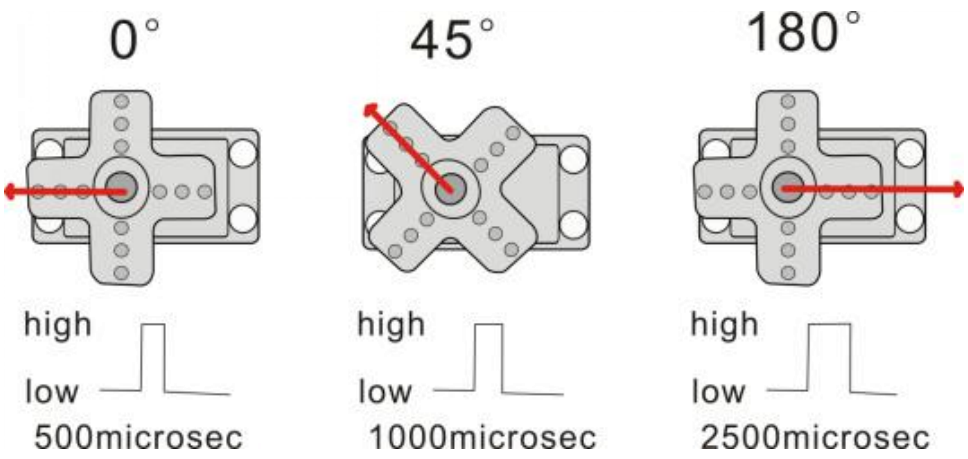
receiver and produces a reference signal with a period of 20ms and width of 1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and obtain the voltage difference output.

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

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



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



The corresponding servo angles are shown below:

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

(2) Specification

Working voltage: DC 4.8V ~ 6V

Operating angle range: about 180 ° (at 500 → 2500 μsec)

Pulse width range: 500 → 2500 μsec

No-load speed: 0.12 ± 0.01 sec / 60 (DC 4.8V) 0.1 ± 0.01 sec / 60 (DC 6V)

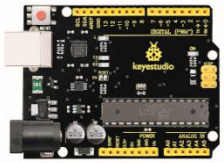
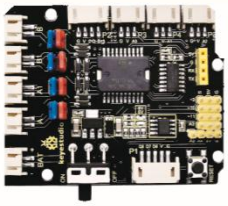
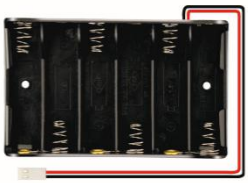
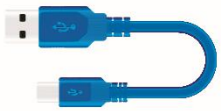
No-load current: 200 ± 20mA (DC 4.8V) 220 ± 20mA (DC 6V)

Stopping torque: 1.3 ± 0.01kg · cm (DC 4.8V) 1.5 ± 0.1kg · cm (DC 6V)

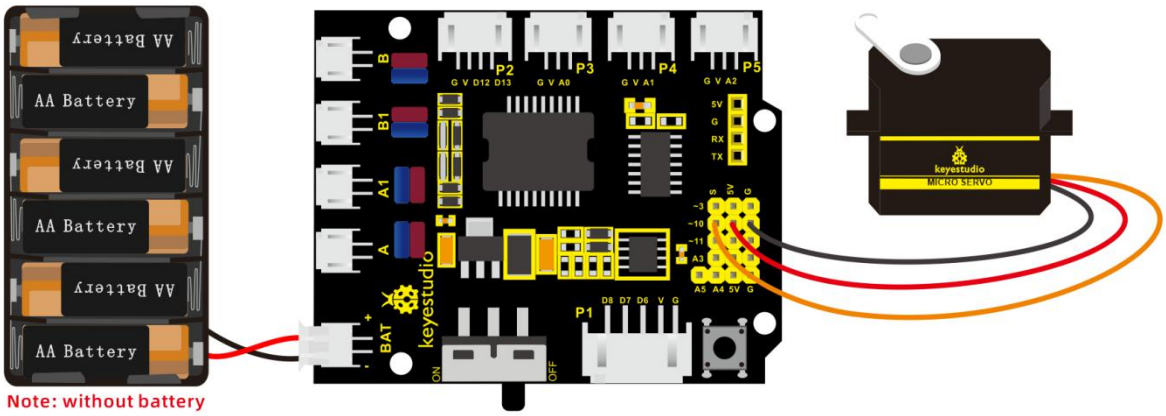
Stop current: ≦ 850mA (DC 4.8V) ≦ 1000mA (DC 6V)

Standby current: 3 ± 1mA (DC 4.8V) 4 ± 1mA (DC 6V)

(3) What You Need

Control Board *1	L298P Motor Shield *1	Servo motor *1	6 AA battery Holder *1	USB Cable *1
				

(4) Connection Diagram:



Wiring note: the brown line of servo is linked with Gnd(G), the red line is connected to 5v(V) and orange line is attached to digital 10.

The servo has to be connected to external power due to its high demand for driving servo current. Generally, the current of development board is not enough. If without connected power, the development board could be burnt.

(5) Test Code1

```
  /*
keyestudio 4wd BT Car V2
lesson 4.1
Servo
http://www.keyestudio.com
  */
#define servoPin 10   //servo Pin
int pos; //the angle variable of servo
int pulsewidth; // pulse width variable of servo
void setup() {
  pinMode(servoPin, OUTPUT);   //set the pins of servo to output
  procedure(0); // set the angle of servo to 0 degree
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
degrees
    // in steps of 1 degree
    procedure(pos);                  // tell servo to go to position in variable
'pos'
    delay(15);                       //control the rotation speed of servo
```

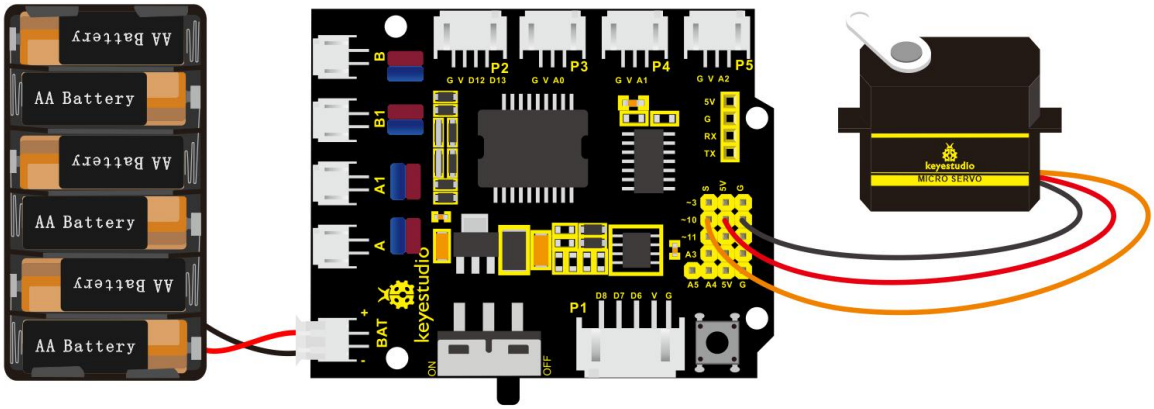
```
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
degrees
    procedure(pos);                    // tell servo to go to position in variable
'pos'
    delay(15);
  }}
// function to control servo
void procedure(int myangle) {
  pulsewidth = myangle * 11 + 500;   //calculate the value of pulse width
  digitalWrite(servoPin,HIGH);
  delayMicroseconds(pulsewidth);     //The duration of high level is pulse
width
  digitalWrite(servoPin,LOW);
  delay((20 - pulsewidth / 1000));   // the cycle is 20ms, the low level last
for the rest of time
}//*****************************************************************************
*******
```

Upload code successfully, servo swings forth and back in the range of 0° to 180°

There is another guide for restraining servo---- servo library file, the following link of official website is for your reference.

https://www.arduino.cc/en/Reference/Servo

The library file of servo is used in the following code



(6) Test Code2

```
/*
  keyestudio 4wd BT Car V2
  lesson 4.2
  servo
  http://www.keyestudio.com
*/
```

```
#include <Servo.h>
Servo myservo;   // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0;      // variable to store the servo position
void setup() {
  myservo.attach(10);   // attaches the servo on pin 9 to the servo object
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
degrees
    // in steps of 1 degree
    myservo.write(pos);                  // tell servo to go to position in
variable 'pos'
    delay(15);                           // waits 15ms for the servo to reach
the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
degrees
    myservo.write(pos);                  // tell servo to go to position in
variable 'pos'
    delay(15);                           // waits 15ms for the servo to reach
the position
```

```
  }}
//*********************************************************************
```

(7) Test Result

Upload code successfully and power on, servo swings in the range of 0° to 180°. The result is same. We usually control it by library file.

(8) Code Explanation

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

The following are some common statements of the servo function:

1. **attach (interface)** ——Set servo interface, port 9 and 10 are available

2. **write (angle)** ——The statement to set rotation angle of servo, the angle range is from 0° to 180°

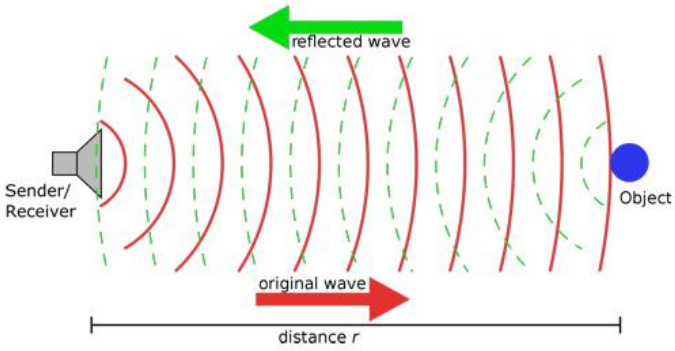
3. **read ()** ——The statement to read angle of servo, read the command value of "write()"

4. **attached ()** ——Judge if the parameter of servo is sent to its interface

Note: The above written format is "servo variable name, specific statement ()", for instance: myservo.attach(9)

Project 5: Ultrasonic Sensor

(1) Description



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

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

(2) Specification

Power Supply :+5V DC

Quiescent Current : <2mA

Working Current: 15mA

Effectual Angle: <15°

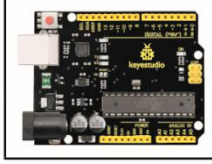
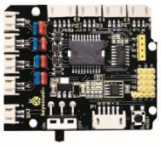
Ranging Distance : 2cm – 400 cm

Resolution : 0.3 cm

Measuring Angle: 30 degree

Trigger Input Pulse width: 10uS

(3) What You Need

Control Board *1	L298P Motor Shield *1	Ultrasonic module *1	LED module *1	USB Cable *1	4pin Dupont line *1
					 3pin Dupont line *1 

(4) The principle of ultrasonic sensor

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

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

The t is the time that emitting signal meets obstacle and returns.

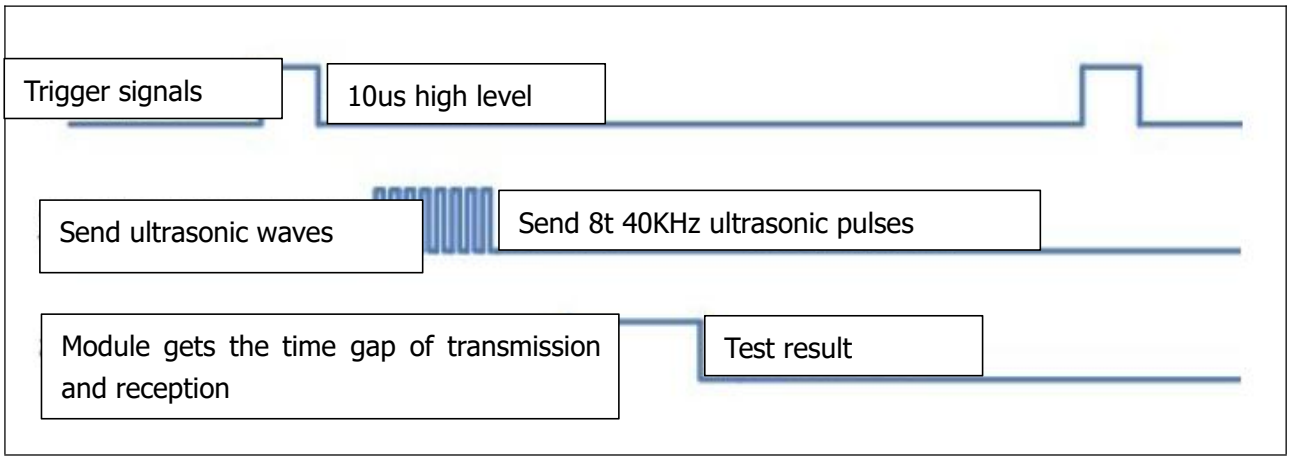
and the propagation speed of sound in the air is about 343m/s, therefore, distance = speed * time, because the ultrasonic wave emits and comes back, which is 2 times of distance, so it needs to be divided by 2, the distance measured by ultrasonic wave = (speed * time)/2

1. Use method and timing chart of ultrasonic module:

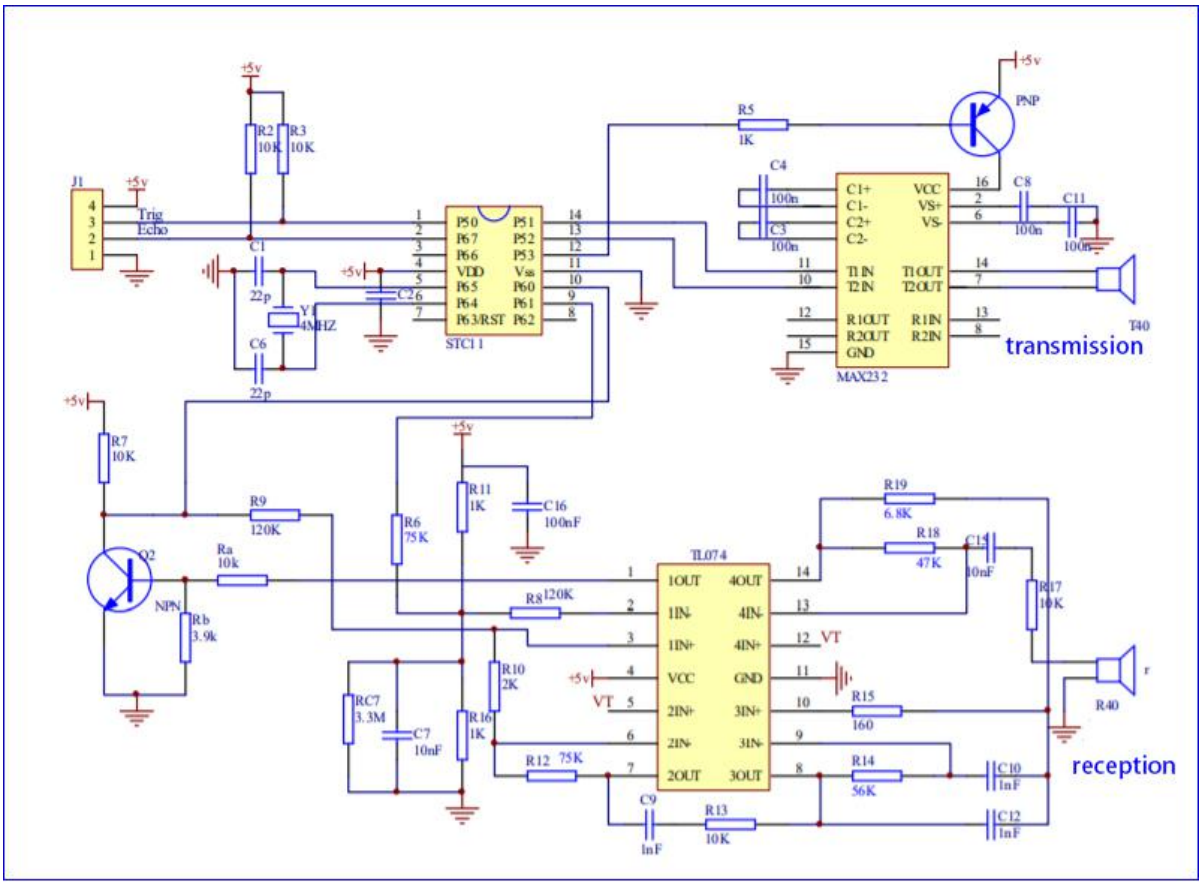
Setting the delay time of Trig pin of SR04 to 10μs at least, which can trigger it to detect distance.

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.

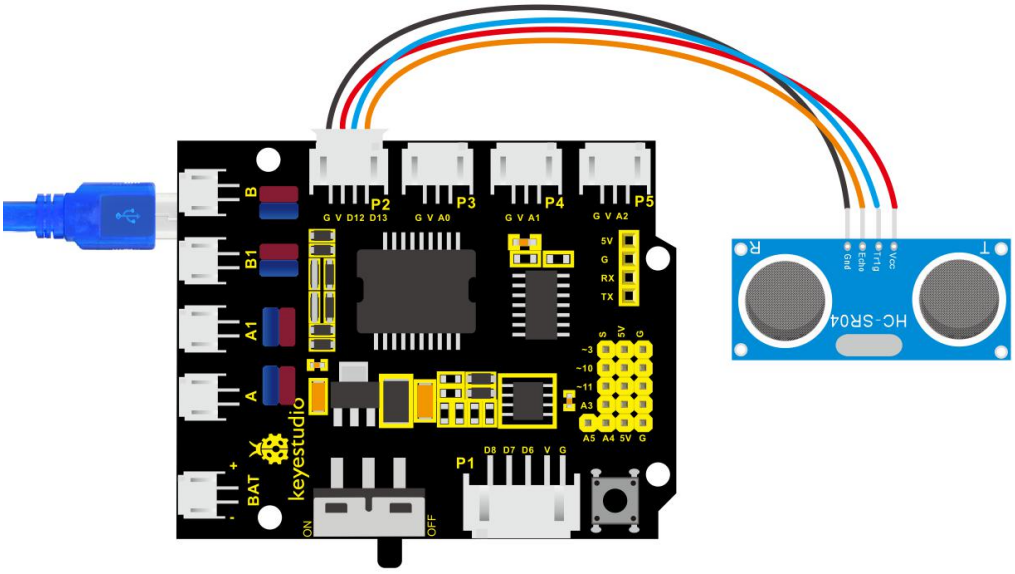
3. If the signal returns, the Echo pin will output a high level, and the duration of the high level is the time from the transmission of the ultrasonic wave to the return.



Circuit diagram of ultrasonic sensor:



(5) Connection Diagram



Wiring guide:

Ultrasonic sensor		keyestudio V5 Sensor Shield
VCC	→	5v(V)
Trig	→	12(S)
Echo	→	13(S)
Gnd	→	Gnd(G)

(6) Test Code

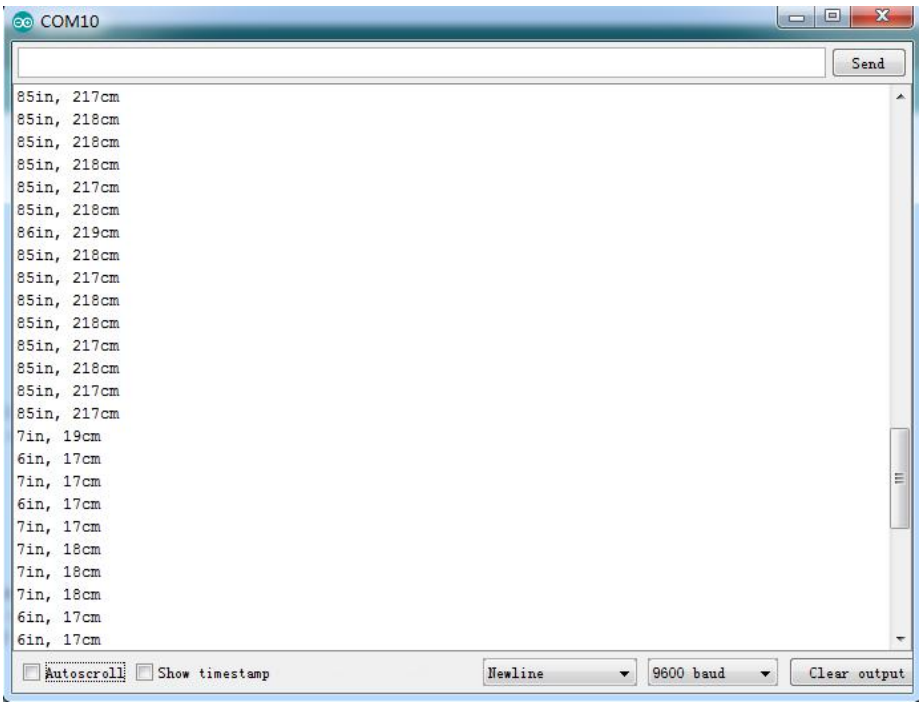
```
/*
  keyestudio 4wd BT Car V2
  lesson 5
  Ultrasonic sensor
  http://www.keyestudio.com
*/
int trigPin = 12;      // Trigger
int echoPin = 13;      // Echo
long duration, cm, inches;
  void setup() {
    //Serial Port begin
    Serial.begin (9600);
```

```
    //Define inputs and outputs
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}
void loop() {
    // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
    // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
     // Read the signal from the sensor: a HIGH pulse whose
    // duration is the time (in microseconds) from the sending
    // of the ping to the reception of its echo off of an object.
    duration = pulseIn(echoPin, HIGH);
     // Convert the time into a distance
    cm = (duration/2) / 29.1;         // Divide by 29.1 or multiply by 0.0343
    inches = (duration/2) / 74;     // Divide by 74 or multiply by 0.0135
       Serial.print(inches);
    Serial.print("in, ");
    Serial.print(cm);
```

```
    Serial.print("cm");
    Serial.println();
    delay(50);
}
//*************************************************************************
```

(7) Test Result

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



(8) Code Explanation

int trigPin- this pin is defined to transmit ultrasonic waves, generally output.

int echoPin - this is defined as the pin of reception, generally input

cm = (duration/2) / 29.1-unit is cm

inches = (duration/2) / 74-unit is inch

We can calculate the distance by using the following formula:

distance = (traveltime/2) x speed of sound

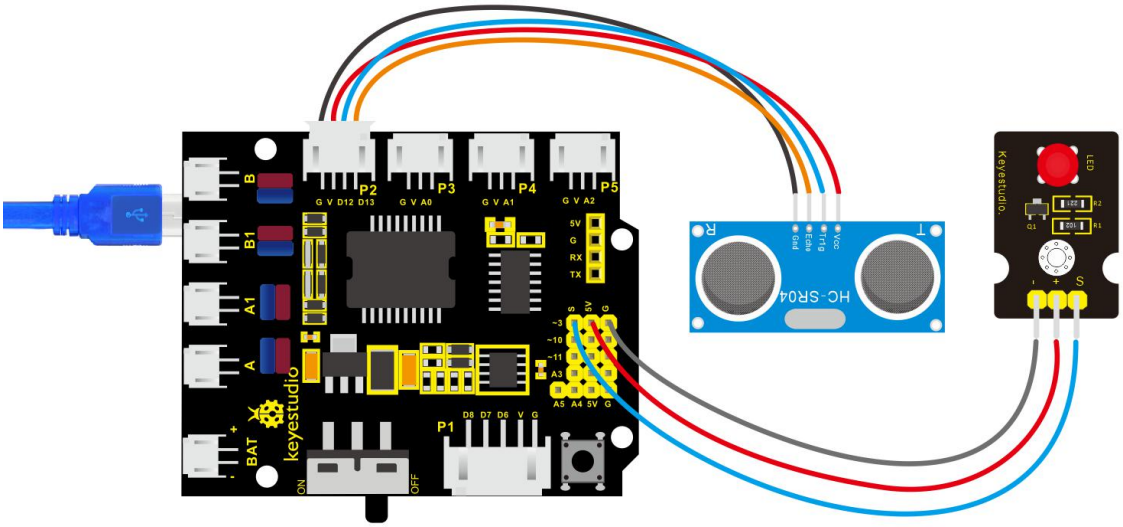
The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1 cm/uS

Or in inches: 13503.9in/s = 0.0135in/uS = 1/74in/uS

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

(9) Extension Practice:

We have just measured the distance displayed by the ultrasonic. How about controlling the LED with the measured distance? Let's try it, connect an LED light module to the D3 pin.



```
/*
  keyestudio 4wd BT Car V2
  lesson 5.2
  Ultrasonic LED
  http://www.keyestudio.com
*/
int trigPin = 12;      // Trigger
int echoPin = 13;      // Echo
long duration, cm, inches;
void setup() {
   Serial.begin (9600);   //Serial Port begin
   pinMode(trigPin, OUTPUT);   //Define inputs and outputs
   pinMode(echoPin, INPUT);
   pinMode(3, OUTPUT);
```

```
}
 void loop()
{
   // The sensor is triggered by a HIGH pulse of 10 or more microseconds.
   // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
   digitalWrite(trigPin, LOW);
   delayMicroseconds(2);
   digitalWrite(trigPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigPin, LOW);
   // Read the signal from the sensor: a HIGH pulse whose
   // duration is the time (in microseconds) from the sending
   // of the ping to the reception of its echo off of an object.
   duration = pulseIn(echoPin, HIGH);
   // Convert the time into a distance
   cm = (duration/2) / 29.1;         // Divide by 29.1 or multiply by 0.0343
   inches = (duration/2) / 74;     // Divide by 74 or multiply by 0.0135
   Serial.print(inches);
   Serial.print("in, ");
   Serial.print(cm);
   Serial.print("cm");
   Serial.println();
```

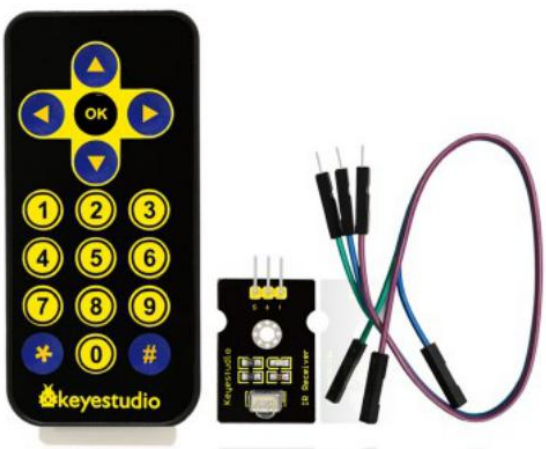
```
  delay(50);
if (cm>=2 && cm<=10)digitalWrite(3, HIGH);
else digitalWrite(3, LOW);
}//*****************************************************************
```

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

Project 6: IR Reception

(1) Description

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



The 38K infrared carrier signal emitted by remote controller is encoded by the encoding chip in the remote controller. It is composed of a section of pilot code, user code, user inverse code, data code, and data inverse code. The time

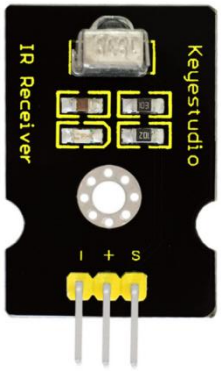
interval of the pulse is used to distinguish whether it is a 0 or 1 signal and the encoding is made up of these 0, 1 signals.

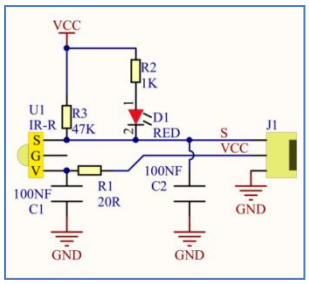
The user code of the same remote control is unchanged. The data code can distinguish the key.

When the remote control button is pressed, the remote control sends out an infrared carrier signal. When the IR receiver receives the signal, the program will decode the carrier signal and determines which key is pressed. The MCU decodes the received 01 signal, thereby judging what key is pressed by the remote control.

Infrared receiver we use is an infrared receiver module. Mainly composed of an infrared receiver head, it is a device that integrates reception, amplification, and demodulation. Its internal IC has completed demodulation, and can achieve from infrared reception to output and be compatible with TTL signals. Additionally, it is suitable for infrared remote control and infrared data transmission. The infrared receiving module made by the receiver has only three pins, signal line, VCC and GND. It is very convenient to communicate with arduino and other microcontrollers.

(2) Specification





Operating Voltage: 3.3-5V（DC）

Interface: 3PIN

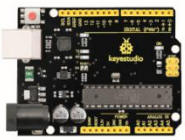
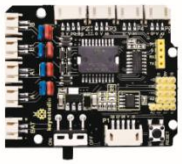
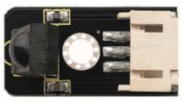
Output Signal: Digital signal

Receiving Angle: 90 degrees

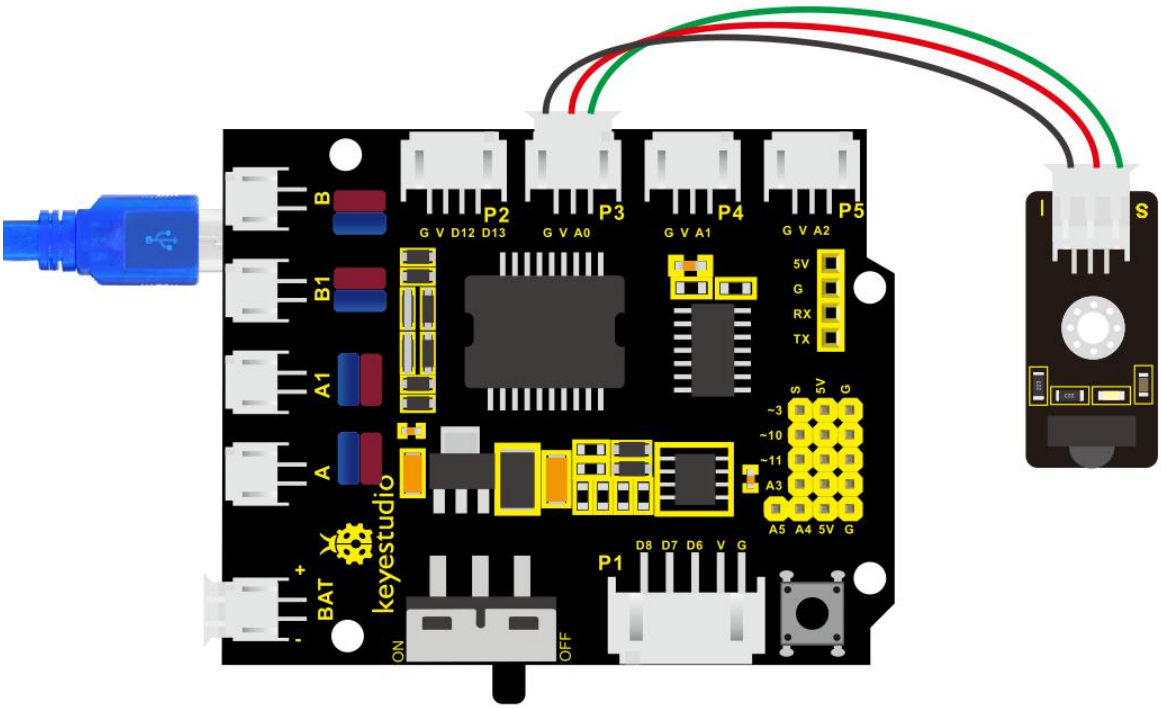
Frequency: 38khz

Receiving Distance: 10m

(3) What You Need

Control Board *1	L298P Motor Shield *1	IR receiver module *1	LED module *1	USB Cable *1	3pin Dupont line *1
					 3pin Dupont line *1 

(4) Connection Diagram



Respectively link "-" , "+" and S of IR receiver module with G(GND) , V (VCC) and A0 of keyestudio development board.

Attention: On the condition that digital ports are not available, analog ports can be regarded as digital ports. A0 equals to D14, A1 is equivalent to digital 15.

(5) Test Code

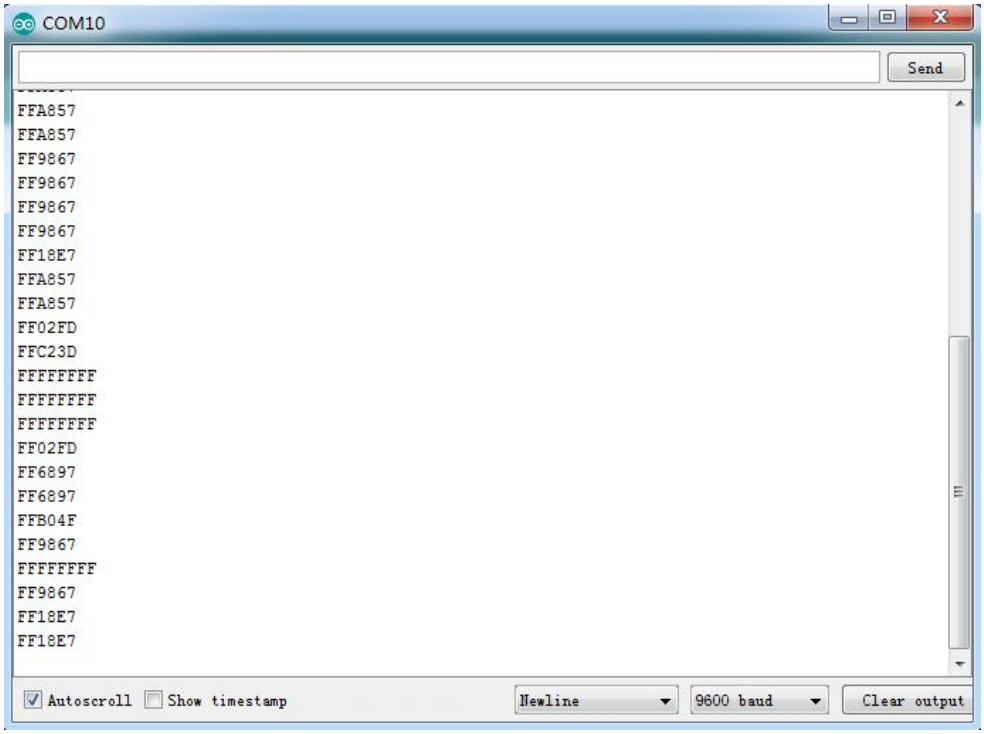
Firstly import library file of IR receiver module(refer to how to import Arduino library file) before designing code.

```
  /*
    keyestudio 4wd BT Car V2
    lesson 6.1
    IRremote
    http://www.keyestudio.com
*/
#include <IRremote.h>        // IRremote library statement
int RECV_PIN = A0;           //define the pins of IR receiver as A0
IRrecv irrecv(RECV_PIN);
decode_results results;     // decode results exist in the “result” of “decode
results”
void setup()
   {
       Serial.begin(9600);
       irrecv.enableIRIn(); // Enable receiver
   }
 void loop() {
       if (irrecv.decode(&results))//decode successfully, receive a set of
  infrared signals
      {
          Serial.println(results.value, HEX);//Wrap word in 16 HEX to output
```

```
and receive code
        irrecv.resume(); // Receive the next value
    }
    delay(100);
} //*******************************************************
```

(6) Test Result

Upload test code, open serial monitor and set baud rate to 9600, point remote control to IR receiver and the corresponding value will be shown, if pressing so long, the error codes will appear.



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

(7) Code Explanation

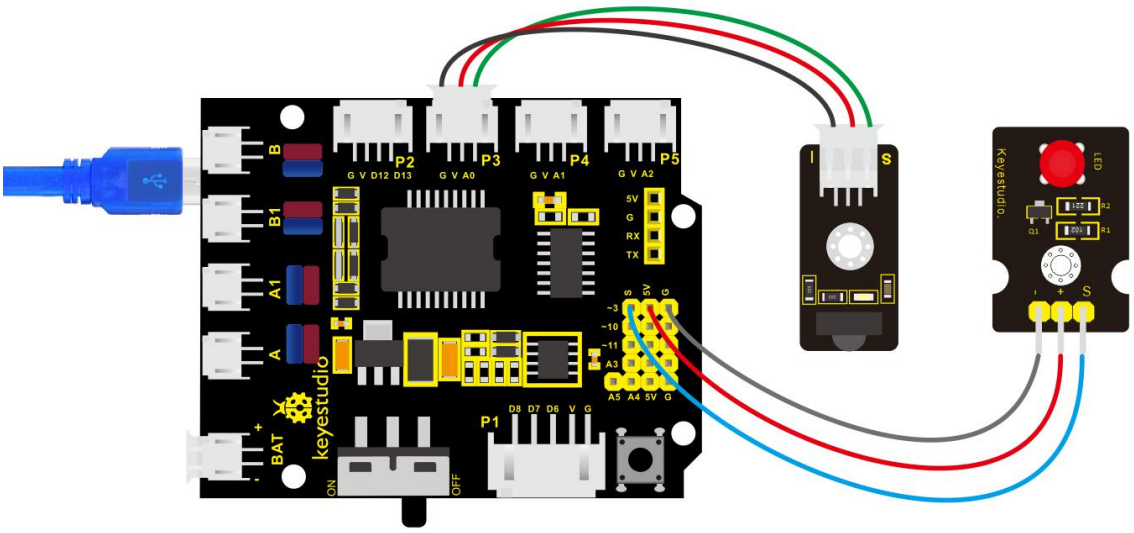
irrecv.enableIRIn(): after enabling IR decoding, the IR signals will be received, then function "decode()" will check continuously if decode successfully.

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

(8) Extension Practice

We decoded the key value of IR remote control. How about controlling LED by the measured value? We could operate an experiment to affirm. Attach an LED to D3, then press the keys of remote control to make LED light up

and off.




```
/* keyestudio 4wd BT Car V2
lesson 6.2
IRremote
http://www.keyestudio.com
  */
 #include <IRremote.h>
 int RECV_PIN = A0;//define the pin of IR receiver as A0
 int LED_PIN=3;// define the pin of LED as pin 3
 int a=0;
 IRrecv irrecv(RECV_PIN);
 decode_results results;
 void setup()
 {Serial.begin(9600);
```

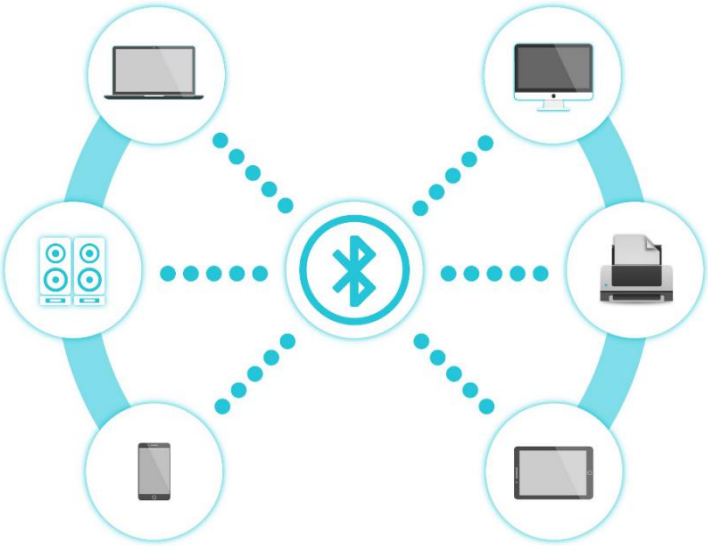
```
    irrecv.enableIRIn(); // Initialize the IR receiver
    pinMode(LED_PIN,OUTPUT);//set pin 3 of LED to OUTPUT
}
void loop() {
    if (irrecv.decode(&results)) {
if(results.value==0xFF02FD &a==0) //according to the above key value,
press “OK” on remote control , LED will be controlled
{digitalWrite(LED_PIN,HIGH);//LED will be on
a=1;
}
else if(results.value==0xFF02FD &a==1) //press again
{
digitalWrite(LED_PIN,LOW);//LED will go off
a=0;
}
        irrecv.resume(); // receive the next value
    }}//*******************************************************
```

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

Project 7: Bluetooth Remote Control

(1) Description

Bluetooth, a simple wireless communication module most popular since the last few decades and easy to use are being used in most of the battery-powered devices.



Over the years, there have been many upgrades of Bluetooth standard to keep fulfil the demand of customers and technology according to the need of time and situation.

Over the few years, there are many things changed including data transmission rate, power consumption with wearable and IoT Devices and Security System.

Here we are going to learn about HM-10 BLE 4.0 with Arduino Board. The HM-10 is a readily available Bluetooth 4.0 module. This module is used for establishing wireless data communication. The module is designed by using the Texas Instruments CC2540 or CC2541 Bluetooth low energy (BLE) System on Chip (SoC).

(2) Specification

Bluetooth protocol: Bluetooth Specification V4.0 BLE

No byte limit in serial port Transceiving

In open environment, realize 100m ultra-distance communication with iphone4s

Working frequency: 2.4GHz ISM band

Modulation method: GFSK(Gaussian Frequency Shift Keying)

Transmission power: -23dbm, -6dbm, 0dbm, 6dbm, can be modified by AT command.

Sensitivity: ≤-84dBm at 0.1% BER

Transmission rate: Asynchronous: 6K bytes ; Synchronous: 6k Bytes

Security feature: Authentication and encryption

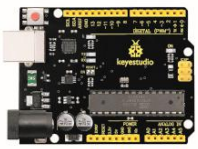
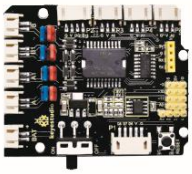
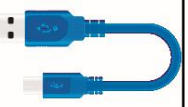
Supporting service: Central & Peripheral UUID FFE0, FFE1

Power consumption: Auto sleep mode, stand by current 400uA~800uA, 8.5mA during transmission.

Power supply: 5V DC

Working temperature: –5 to +65 Centigrade

(3) What You Need

Control Board *1	L298P Motor Shield *1	4.0 Bluetooth module *1	LED module *1	USB Cable *1	3pin Dupont line *1
					

(4) Connection Diagram

1. STATE: state test pins, connected to internal LED, generally keep it unconnected.

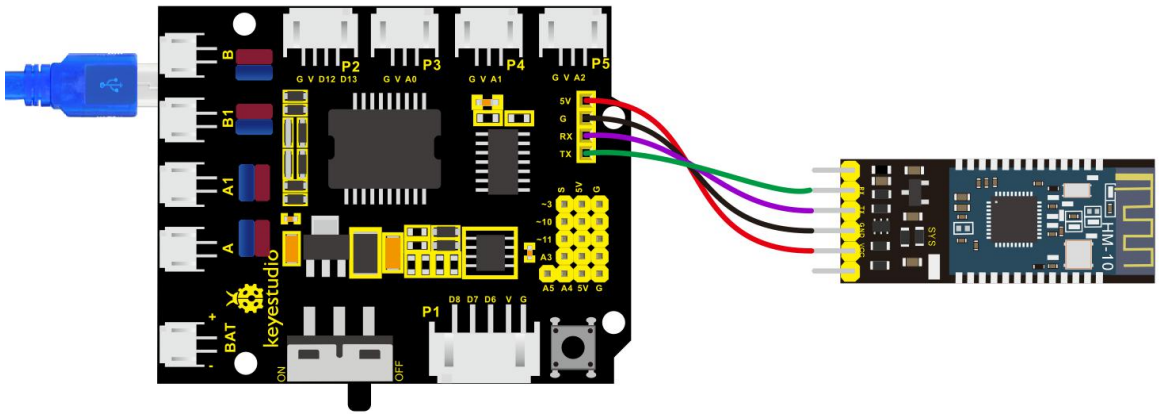
2. RXD: serial interface, receiving terminal.

3. TXD: serial interface, transmitting terminal.

4. GND: Ground.

5. VCC: positive pole of the power source.

6. EN/BRK: break connect, it means breaking the Bluetooth connection, generally, keep it unconnected.



Pay attention to the pin direction when inserting Bluetooth module, and don’ t insert it before uploading test code

(5) Test Code

```
/*
  keyestudio 4wd BT Car V2.0
  lesson 7.1
  bluetooth
http://www.keyestudio.com
*/

char ble_val; //character variable, used to store the value received by Bluetooth

void setup() {
  Serial.begin(9600);
}
void loop() {
  if(Serial.available() > 0)   //make sure if there is data in serial buffer

  {
    ble_val = Serial.read();   //Read data from serial buffer
    Serial.println(ble_val);   //Print
  }}
//*****************************************************************************
```

(There will be contradiction between serial communication of code and communication of Bluetooth when uploading code, therefore, don’ t link with Bluetooth module before uploading code.)

After uploading code on development board, then insert Bluetooth module, wait for the command from cellphone.

(6) Download APP

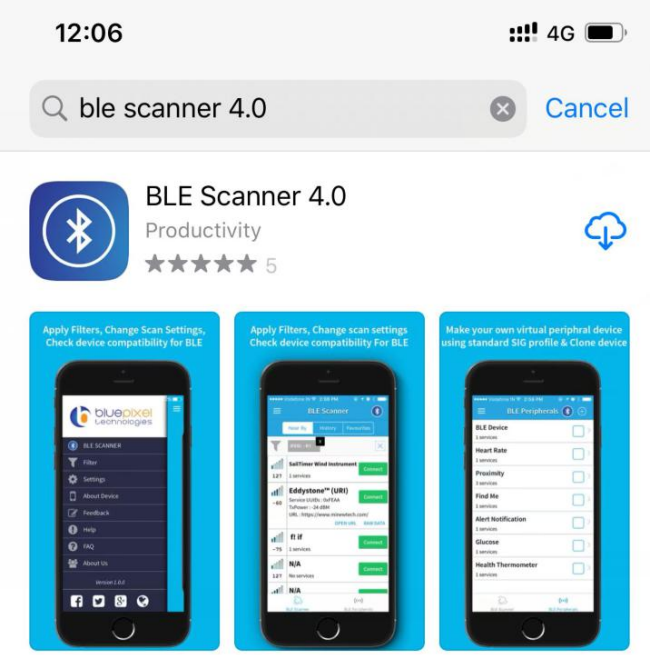
The code is for reading the received signal, and we also need a stuff to send signal. In this project, we send signal to control robot car via cellphone.

Then we need to download the APP.

1. iOS system

Note: Allow APP to access “location” in settings of your cellphone when connecting to Bluetooth module, otherwise, Bluetooth may not be connected.

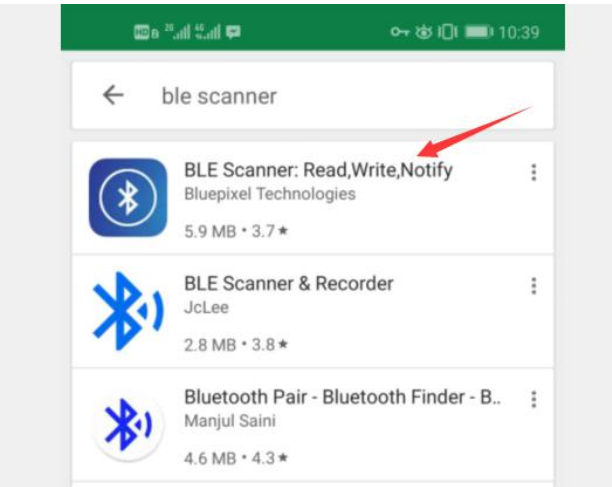
Enter APP STORE to search **BLE Scanner 4.0, then download it.**



2. Android system

Enter Google Play to find out **BLE Scanner, then download.**

And allow APP to access "location", you could enable "location" in settings of your cellphone.

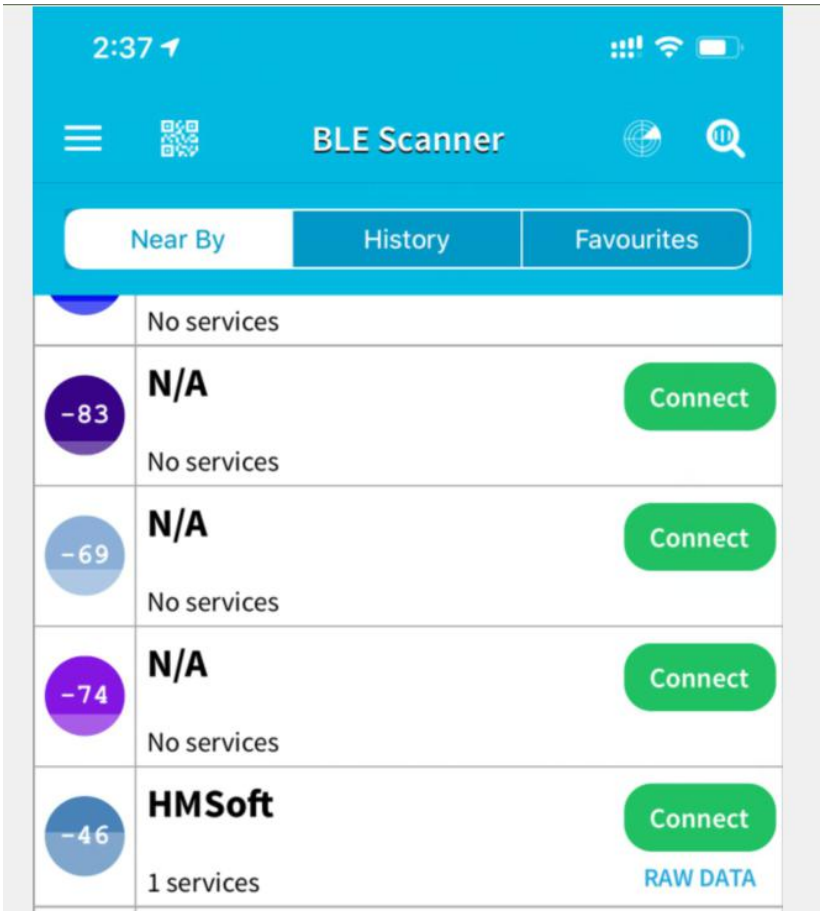


3. After installation, open App and enable "Location and Bluetooth"

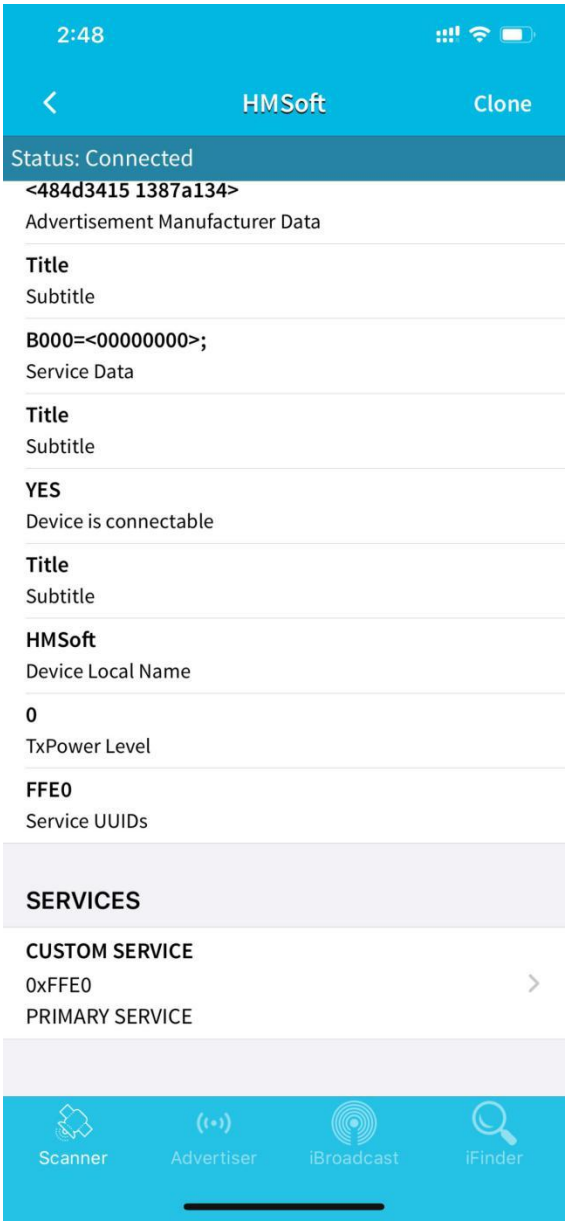
permission.

4. Open App, the name of Bluetooth module is HMSoft.

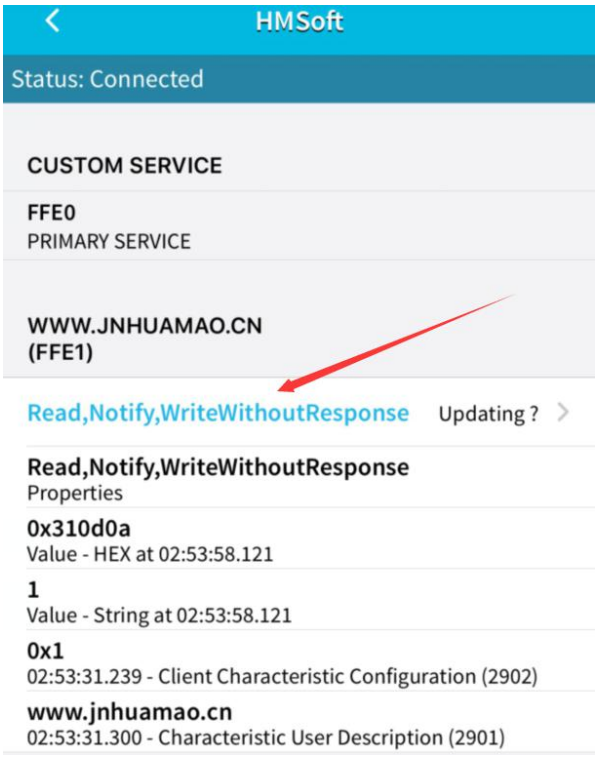
Then click "connect" to link with Bluetooth



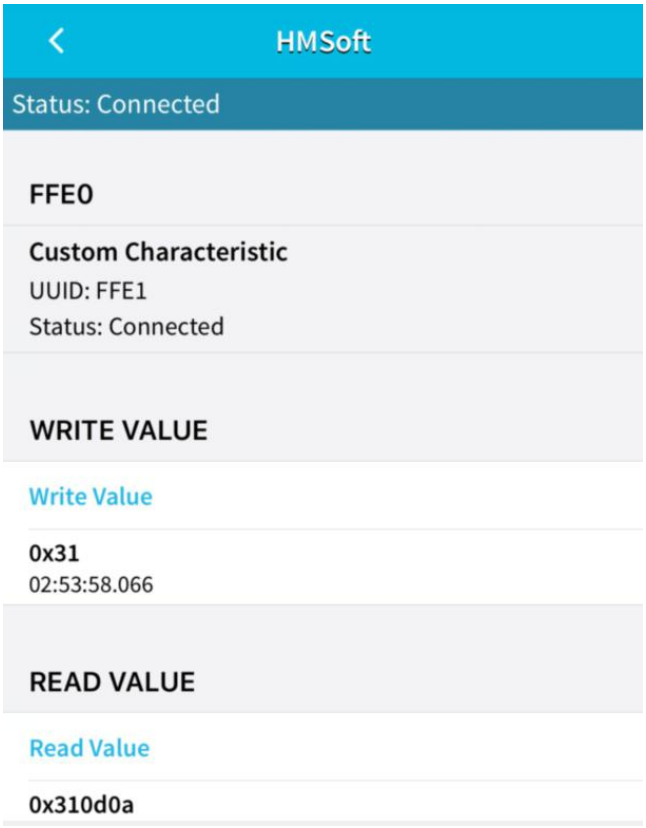
5. After connecting to HMSoft, click it to get multiple options, such as device information, access permission, general and custom service. Choose "CUSTOM SERVICE"



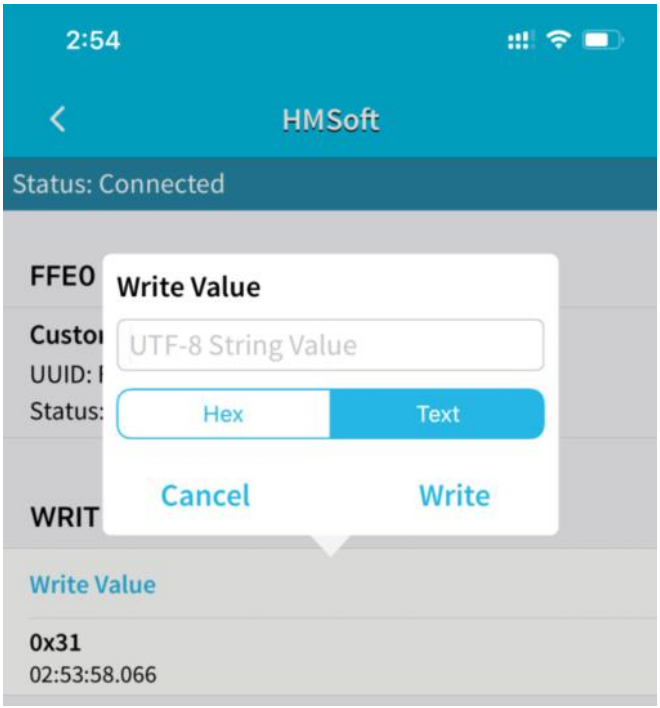
6. Then pop up the following page.



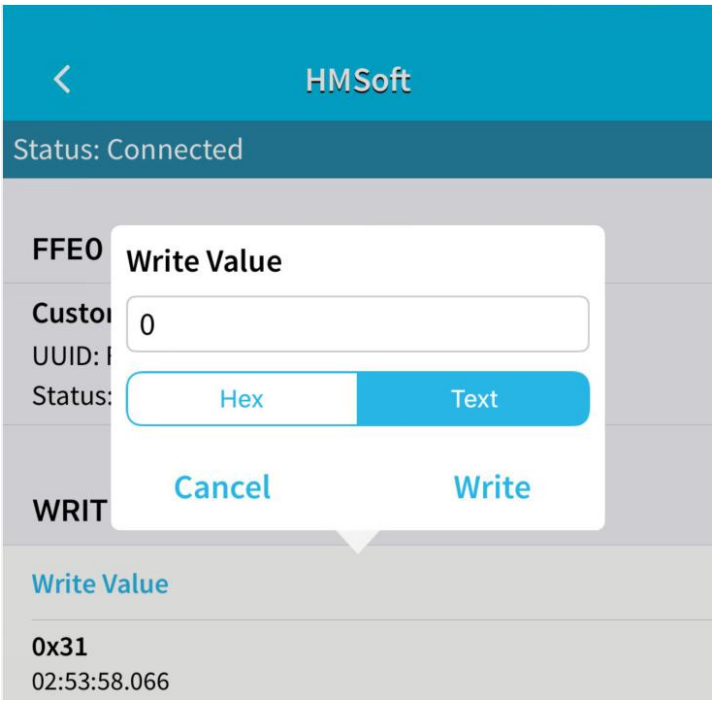
7. Click （Read,Notify,WriteWithoutResponse)to enter the following page



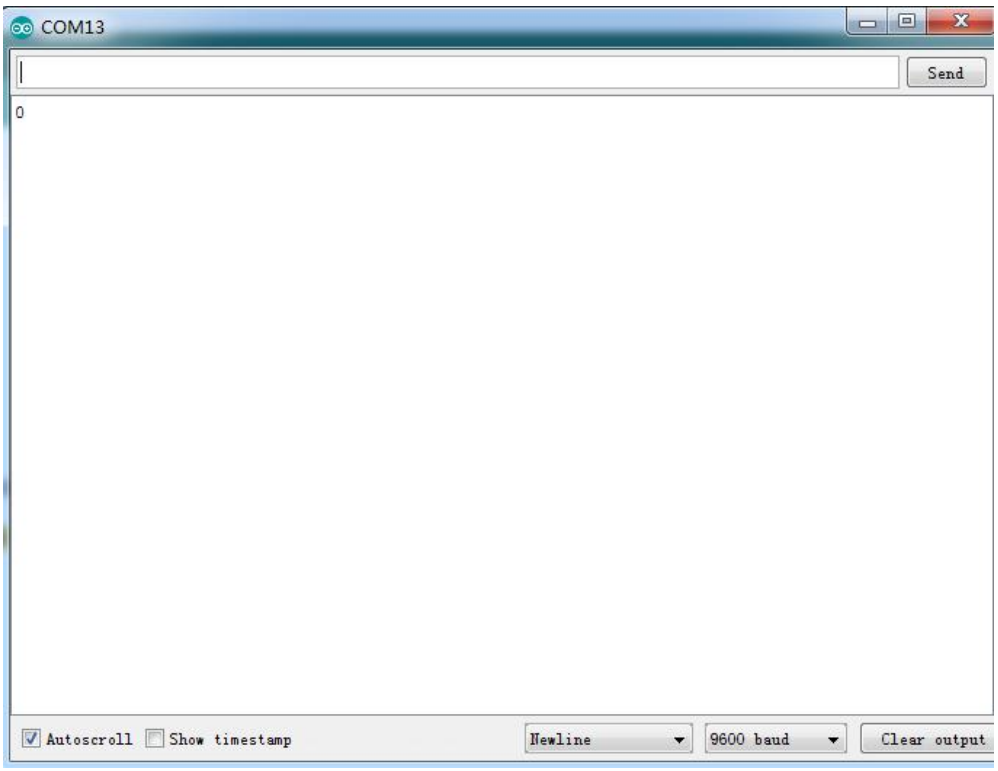
8. Click **Write Value to enter HEX or Text.**



9. Open the serial monitor on Arduino，enter a 0 or other character on Text interface.



10. Then click “Write”，open serial monitor to view if there is a “0” signal



(7) Code Explanation

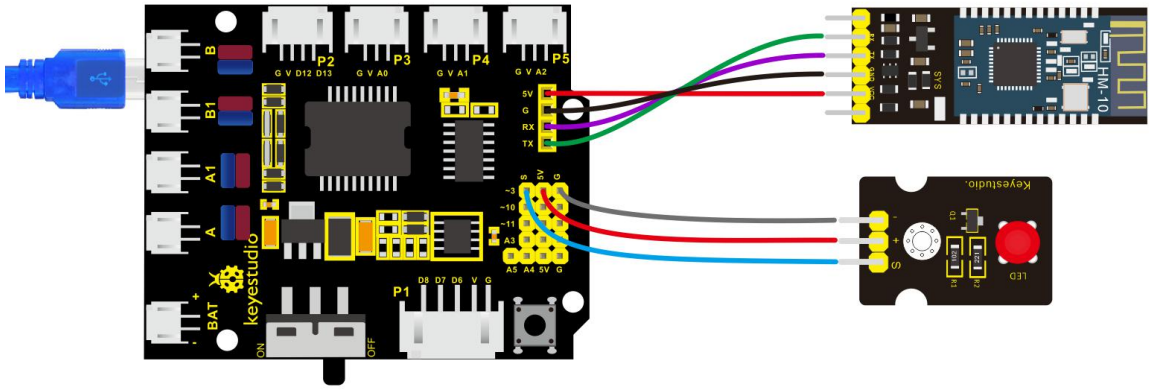
Serial.available() : The current rest characters when return to buffer area. Generally, this function is used to judge if there is data in buffer. When Serial.available()>0, it means that serial receives the data and can be read

Serial.read()： Read a data of a Byte in buffer of serial port, for instance, device sends data to Arduino via serial port, then we could read data by "Serial.read()"

(8) Extension Practice

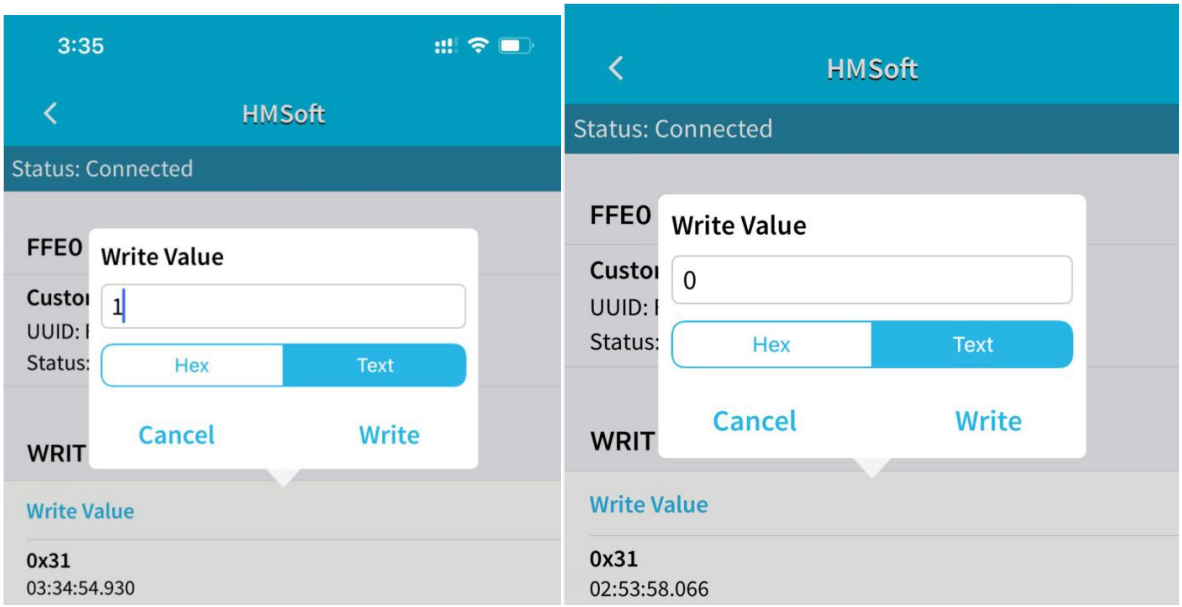
We could send a command via cellphone to turn on and off a LED.

D3 is connected to a LED, as shown below:




```
/*
  keyestudio 4wd BT Car V2.0
  lesson 7.2
  Bluetooth
  http://www.keyestudio.com
*/
int ledpin=3;
void setup()
{
   Serial.begin(9600);
   pinMode(ledpin,OUTPUT);
}
void loop()
```

```
{
  int i;
  if (Serial.available())
  {
    i=Serial.read();
    Serial.println("DATA RECEIVED:");
    if(i=='1')
    {
      digitalWrite(ledpin,1);
      Serial.println("led on");
    }
    if(i=='0')
    {
      digitalWrite(ledpin,0);
      Serial.println("led off");
    }}}
//*********************************************************************************
```



Click "Write" on APP, when you enter 1, LED will be on, when you input 0, LED will be off. (Remember to remove the Bluetooth module after finishing experiment, otherwise, burning code will be affected)

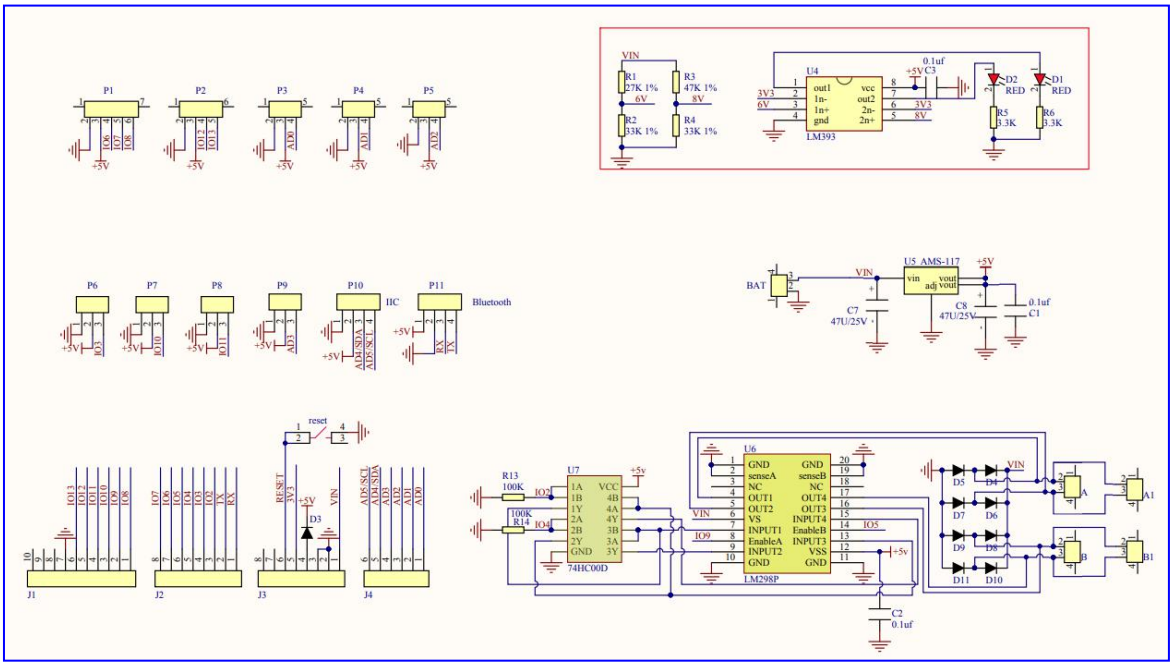
Project 8: Motor Driving and Speed Control

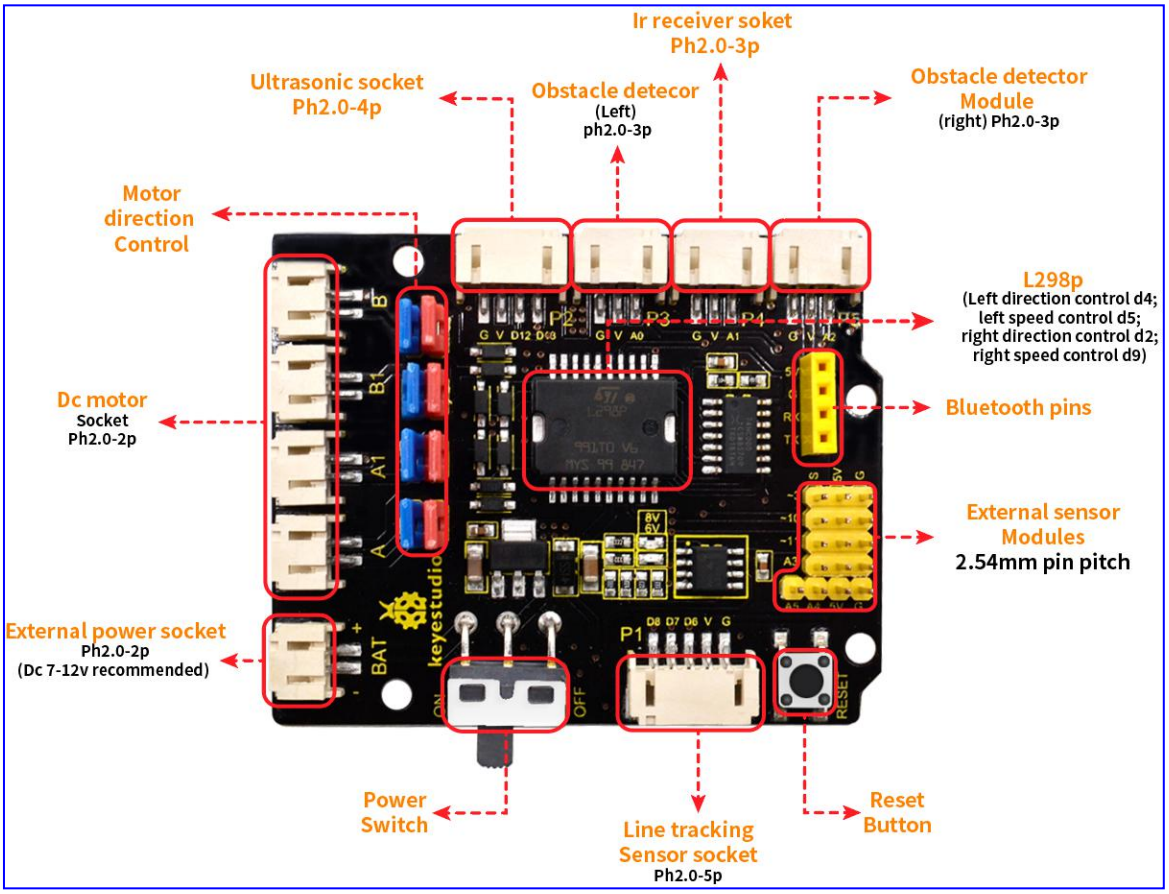
(1) Description

There are many ways to drive the motor. Our robot car uses the most commonly used L298P solution. L298P is an excellent high-power motor driver IC produced by STMicroelectronics. It can directly drive DC motors, two-phase and four-phase stepping motors. The driving current up to 2A, and output terminal of motor adopts eight high-speed Schottky diodes as protection.

We designed a shield based on the circuit of L298p.

The stacked design reduces the technical difficulty of using and driving the motor.





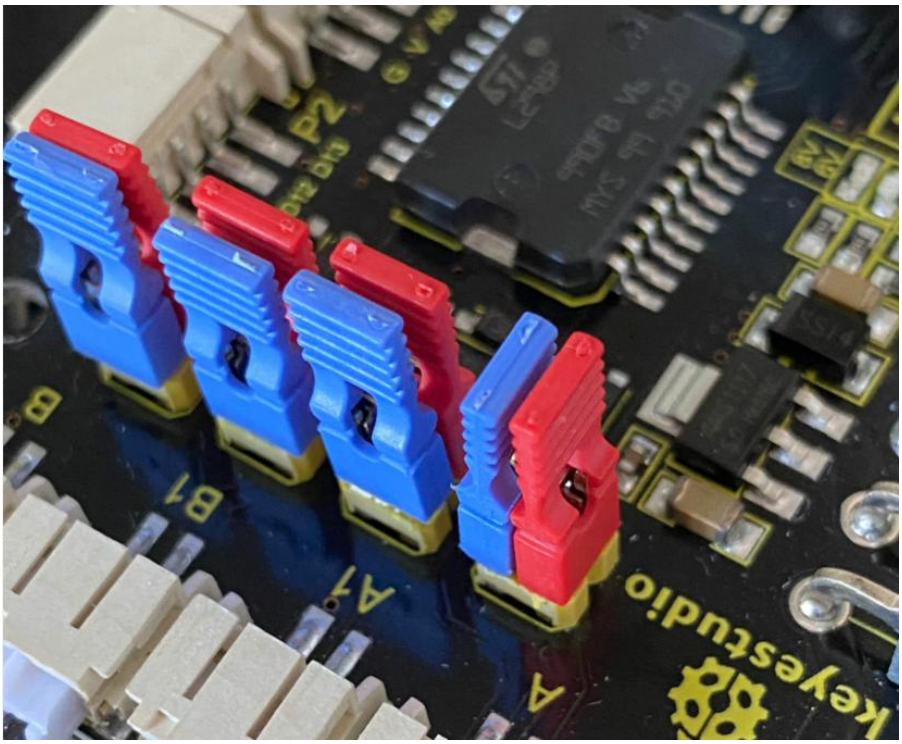
(2) Specification

Circuit Diagram for L298P Board

1) Logic part input voltage: DC5V
2) Driving part input voltage: DC 7-12V
3) Logic part working current: <36mA
4) Driving part working current: <2A
5) Maximum power dissipation: 25W (T=75℃)
6) Working temperature: -25℃～ +130℃
7) Control signal input level: high level 2.3V<Vin<5V, low level -0.3V<Vin<1.5V

(3) Drive Robot to Move

The driver of motor driver shield is in parallel connection. You could control the direction of motors by altering the orientation of jumper caps(seen in the picture).

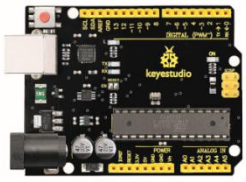
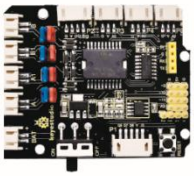
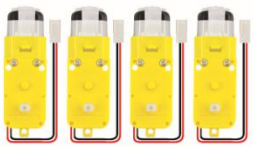
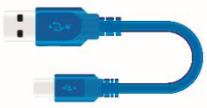
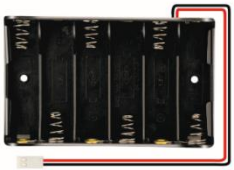


Through the above diagram, the direction pin of B motor is D4, and speed pin is D5; D2 is the direction pin of A motor, D9 is speed pin.

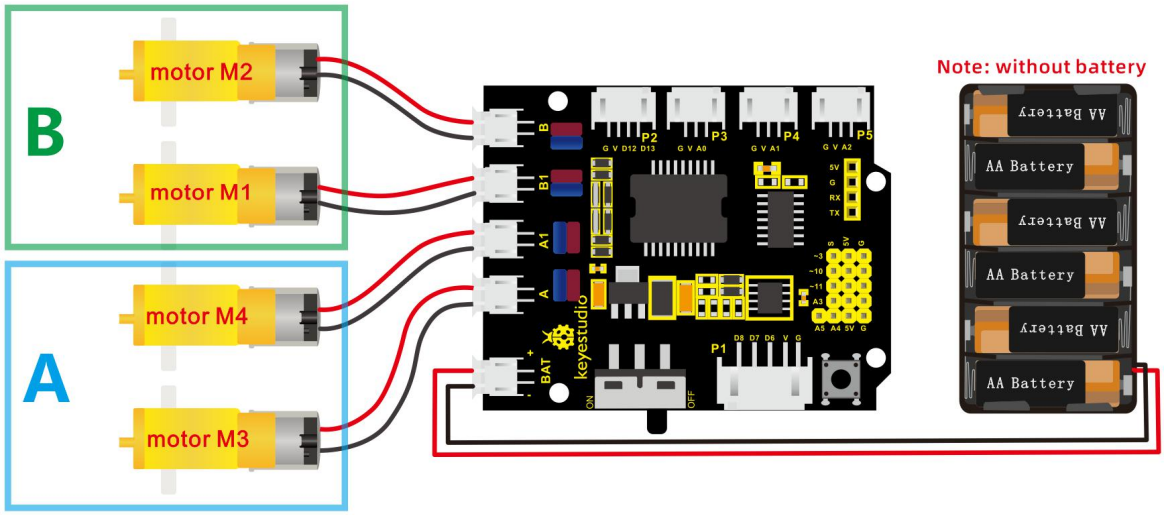
PWM decides 2 motors to rotate so as to drive robot car. The PWM value is in the range of 0-255, the larger the number, the faster the motor rotates

4WD Robot	Motor (A)	Motor (B)
Forward	Turn clockwise	
Backward	Turn anticlockwise	
Rotate to left	Turn anticlockwise	Turn clockwise
Rotate to right	Turn clockwise	Turn anticlockwise
Stop	Stop	Stop

(4) What You Need

Control Board *1	L298P Motor Shield *1	Motor *4	USB Cable *1	6 AA battery Holder *1
				

(5) Connection Diagram



Attention: connect motors in compliance with the above connection diagram

(6) Test Code

```
/*
  keyestudio 4wd BT Car V2.0
  lesson 8
  motor driver shield
  http://www.keyestudio.com
*/
#define ML_Ctrl 4       // define the direction control pin of B motor
#define ML_PWM 5     //define the PWM control pin of B motor
```

```
#define MR_Ctrl 2       //define direction control pin of A motor
#define MR_PWM 9     //define the PWM control pin of A motor
void setup()
{
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to output
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to output
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to output.
  pinMode(MR_PWM, OUTPUT);//set the PWM control pin of A motor to output
}
void loop()
{
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of A motor to HIGH
  analogWrite(MR_PWM,200);//set the PWM control speed of A motor to
```

```
200


  //front
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of B motor to
LOW
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to
200
  digitalWrite(MR_Ctrl,LOW);//set the direction control pin of A motor to
LOW
  analogWrite(MR_PWM,200);//set the PWM control speed of A motor to
200
   //back
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of B motor to
LOW
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to
200
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of A motor to
HIGH
  analogWrite(MR_PWM,200);// set the PWM control speed of A motor to
200
```

```
    //left
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set the PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);// set the direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set the PWM control speed of A motor to 200

   //right
  delay(2000);//delay in 2s
  analogWrite(ML_PWM,0);//set the PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set the PWM control speed of A motor to 0

    //stop
  delay(2000);//delay in 2s
}//*****************************************************************************
```

(7) Test Result

Hook up by connection diagram, upload code and power on, smart car goes forward and back for 2s, turns left and right for 2s, stops for 2s and alternately.

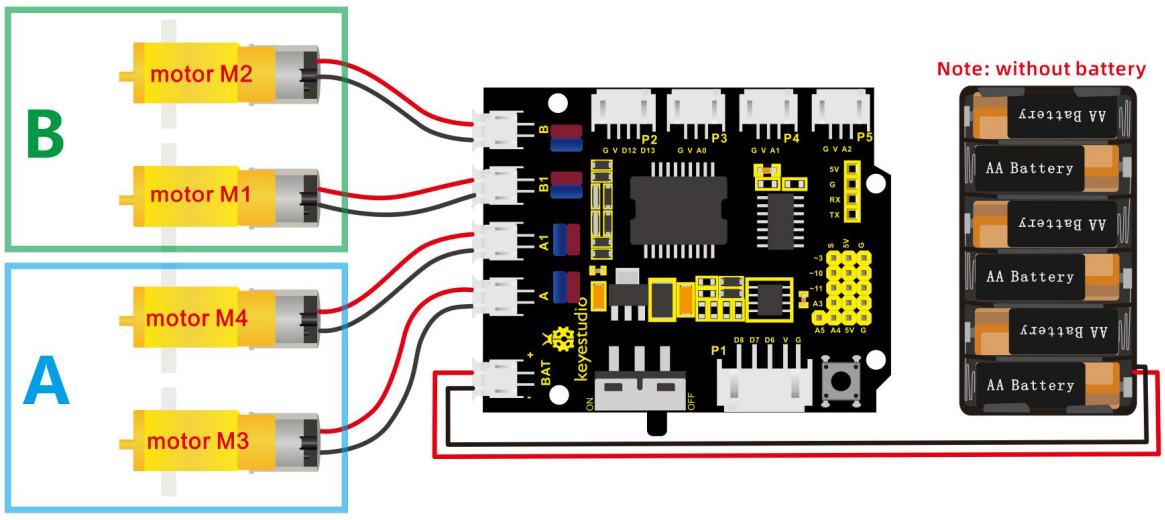
(8) Code Explanation

digitalWrite(ML_Ctrl,LOW): the rotation direction of motor is decided by the high/low level and and the pins that decide rotation direction are digital pins.

analogWrite(ML_PWM,200): the speed of motor is regulated by PWM, and the pins that decide the speed of motor must be PWM pins.

(9) Extension Practice

Adjust the speed that PWM controls the motor, hook up in same way



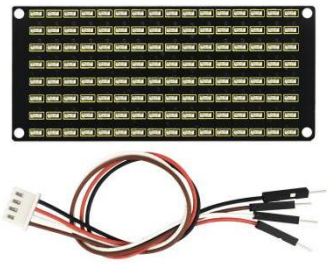
```
/*
 keyestudio 4wd BT Car V2.0
 lesson 8.2
 motor driver
 http://www.keyestudio.com
*/
#define ML_Ctrl 4        //define the direction control pin of B motor
#define ML_PWM 5     //define the PWM control pin of B motor
#define MR_Ctrl 2       //define the direction control pin of A motor
#define MR_PWM 9     //define the PWM control pin of A motor
void setup()
{
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to
OUTPUT
  pinMode(ML_PWM, OUTPUT);//set the PWM control pin of B motor to
OUTPUT
 pinMode(MR_Ctrl, OUTPUT);//set the direction control pin of A motor to
OUTPUT
  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to
OUTPUT
}
```

```
void loop()
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH level
  analogWrite(ML_PWM,250);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,250);//Set PWM control speed of A motor to 100
  //front
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,250);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,250);//Set PWM control speed of A motor to 100
   //back
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,250);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,250);//Set PWM control speed of A motor to 100
     //left
```

```
  delay(2000);//delay in 2s
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
level
  analogWrite(ML_PWM,250);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,250);//Set PWM control speed of A motor to 100
   //right
  delay(2000);//delay in 2s
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
     //stop
  delay(2000);//delay in 2s
}//*********************************************************************
```

After uploading the code successfully, do you find the motors rotate faster?

Project 9: 8*16 LED Board



(1) Description

If we add a 8*16 LED board to the robot, it will be amazing. Keyestudio's 8*16 dot matrix can meet your requirements. You can create facial emoticons, patterns or other interesting displays yourself. 8*16 LED light board comes with 128 LEDs. The data of the microprocessor (arduino) communicates with the AiP1640 through the two-wire bus interface, so as to control the 128 LEDs on the module, which produce the patterns you need on dot matrix. To facilitate wiring, we also provide a HX-2.54 4Pin wiring.

(2) Specification

Working voltage: DC 3.3-5V

Power loss: 400mW

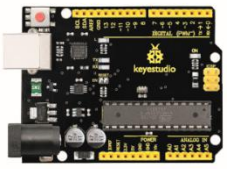
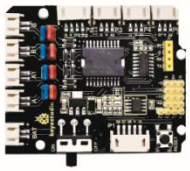
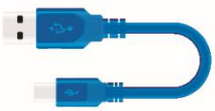
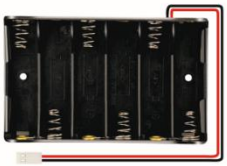
Oscillation frequency: 450KHz

Drive current: 200mA

Working temperature: -40~80℃

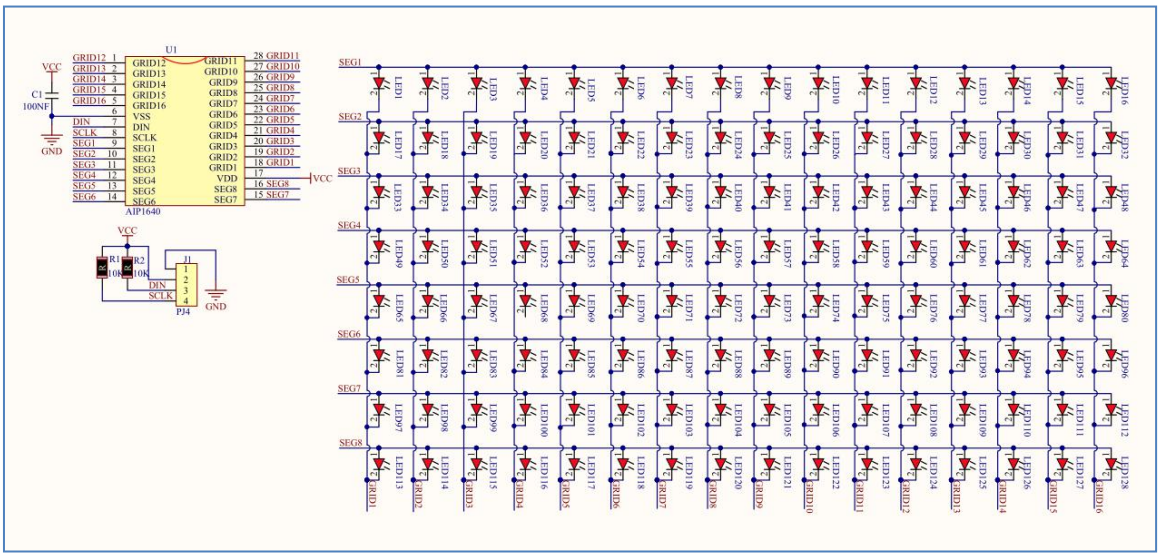
Communication method: two-wire bus

(3) What You Need

Control Board *1	L298P Motor Shield *1	8x16 LED panel board *1	USB Cable *1	6 AA battery Holder *1
				

(4) 8*16 Dot Matrix Display

Circuit Graph:



The principle of 8*16 dot matrix:

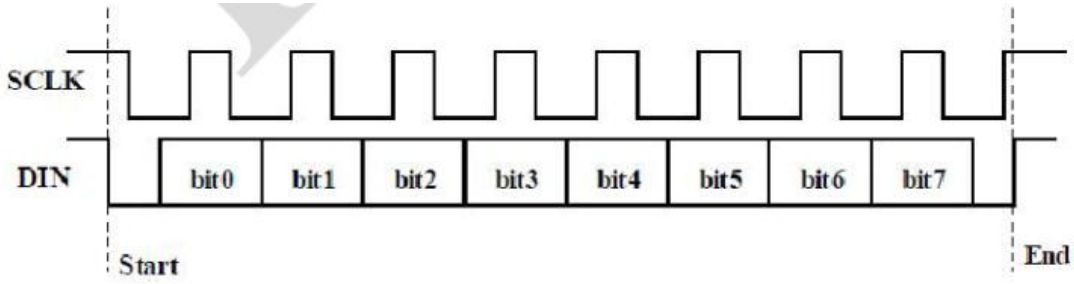
How to control each led light of 8*16 dot matrix? We know that a byte has 8 bits, each bit is 0 or 1. When a bit is 0, turn off LED and when a bit is 0, turn on LED. Thereby, one byte can control the LED in a row of dot matrix, so 16 bytes can control 16 columns of led lights, that is, 8*16 dot matrix.

Interface Description and Communication Protocol:

The data of the microprocessor (arduino) communicates with the AiP1640 through the two-wire bus interface.

The communication protocol diagram is shown below:

(SCLK) is SCL, (DIN) is SDA:



①The starting condition for data input: SCL is high level and SDA changes from high to low.

②For data command setting, there are methods as shown in the figure below

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

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

③For address command setting, the address can be selected as shown below.

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

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

④The requirement for data input is that SCL is high level when inputting data, the signal on SDA must remain unchanged. Only when the clock

signal on SCL is low level, the signal on SDA can be altered. The data input is low-order first, high-order is behind

⑤ The condition to end data transmission is that when SCL is low, SDA is low, and when SCL is high, the SDA level also becomes high.

⑥ Display control, set different pulse width, the pulse width can be selected as shown below

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

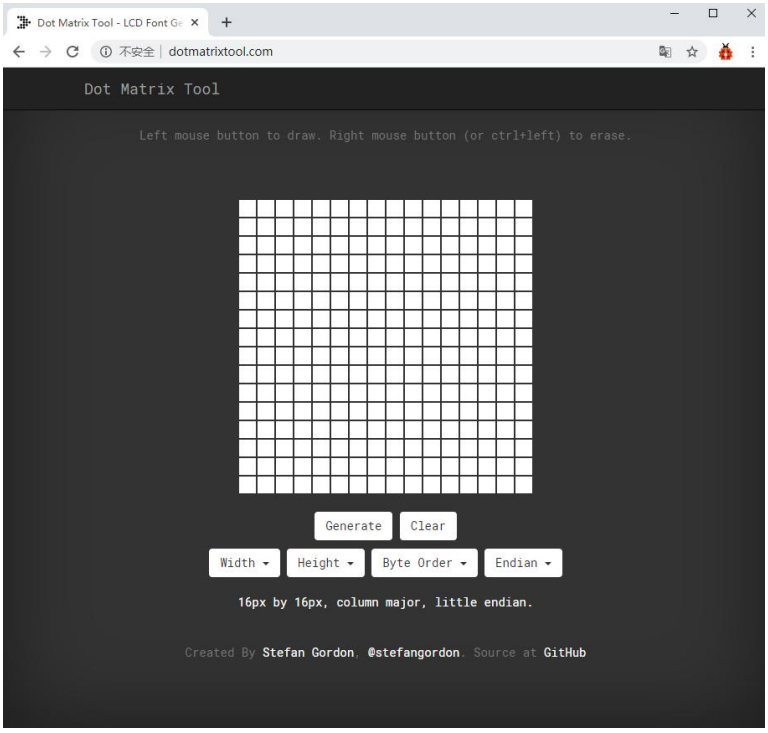
B7	B6	B5	B4	B3	B2	B1	B0	Function	Description
1	0	Irrelevant choice, fill in 0		1	0	0	0	Clear quantity setting (Brightness setting)	Set pulse width to 1/16
1	0			1	0	0	1		Set pulse width to 2/16
1	0			1	0	1	0		Set pulse width to 4/16
1	0			1	0	1	1		Set pulse width to 10/16
1	0			1	1	0	0		Set pulse width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

4. Introduction for Modulus Tool

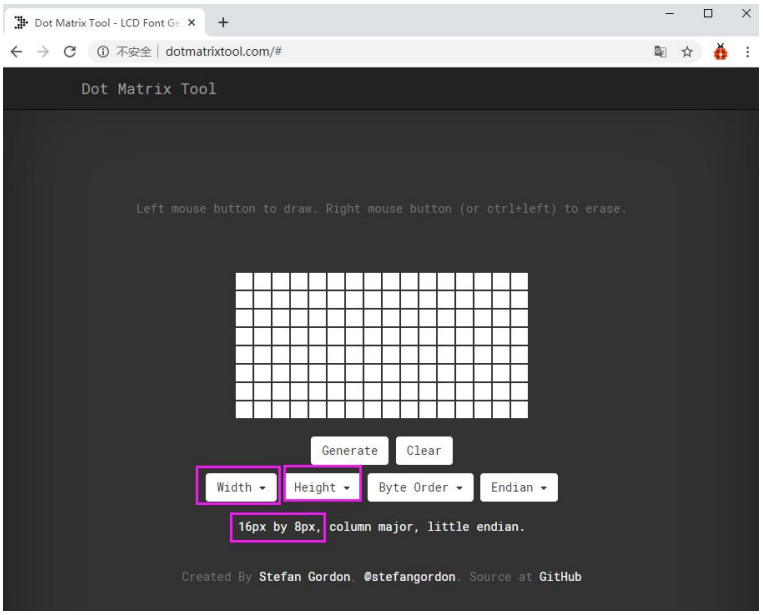
The online version of dot matrix modulus tool:

http://dotmatrixtool.com/#

①Open links to enter the following page.

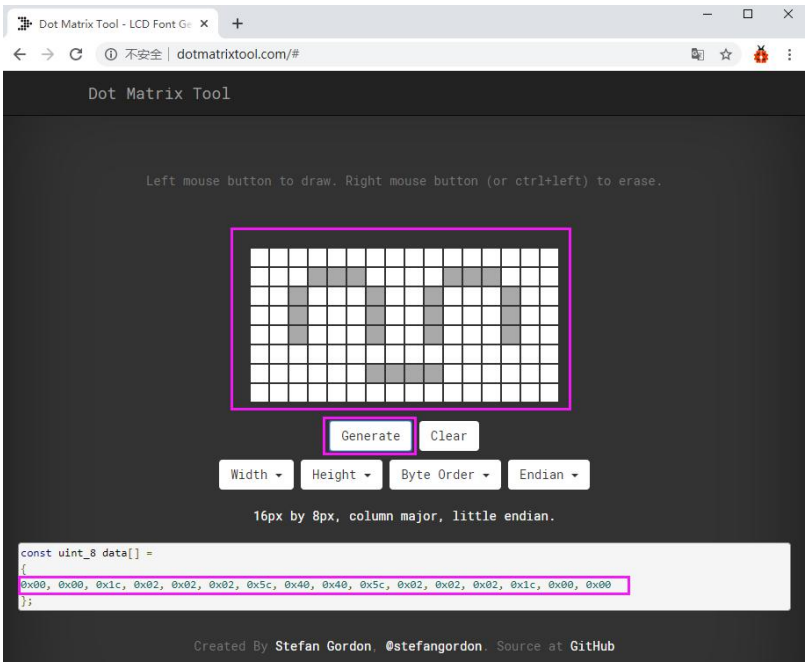


②The dot matrix is 8*16 in this project, so set the height to 8, width to 16, as shown below.

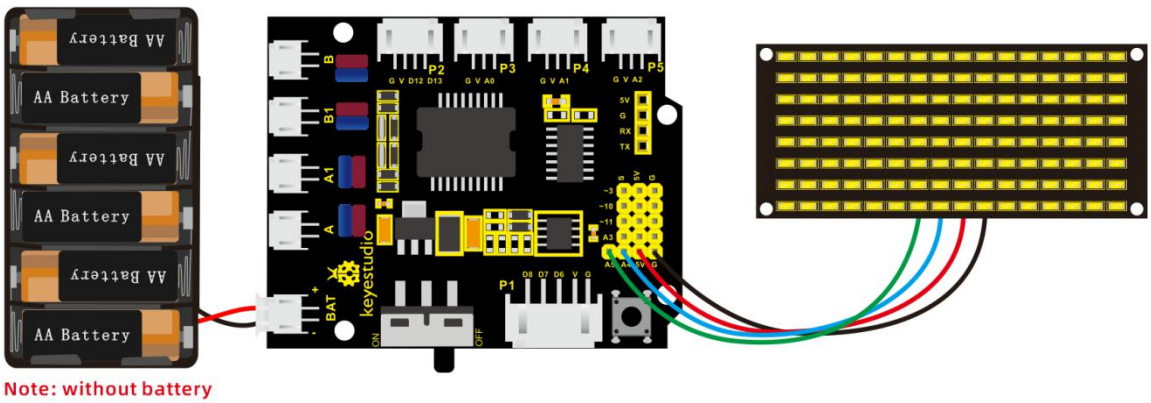


③ Generate hexadecimal data from the pattern

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



(5) Connection Diagram



Note: without battery

Wiring note: The GND, VCC, SDA, and SCL of the 8*16 LED panel are respectively connected to -(GND), + (VCC), A4 and A5 of the keyestudio sensor expansion board for two-wire serial communication. (Note: This pin is connected to arduino IIC, but this module is not IIC communication, it can be linked with any two pins.)

(6) Test Code

The code that shows smile face

```
 /*
 keyestudio 4wd BT Car V2.0
 lesson 9.1
 matrix
 http://www.keyestudio.com
*/
//get the data of smile pattern in the modulus tool
unsigned char smile[] = {0x00, 0x00, 0x1c, 0x02, 0x02, 0x02, 0x5c, 0x40, 0x40, 0x5c, 0x02, 0x02, 0x02, 0x1c, 0x00, 0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
void setup(){
```

```
    //Set pin to output
    pinMode(SCL_Pin,OUTPUT);
    pinMode(SDA_Pin,OUTPUT);
    //Clear the matrix display
    //matrix_display(clear);
}
void loop(){
    matrix_display(smile);    //display smile pattern
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start();    //the function to call the data transmission
    IIC_send(0xc0);    //Select address

    for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //data to convey patterns
    }
    IIC_end();     //end the transmission of patterns data
    IIC_start();
    IIC_send(0x8A);    //display control, set pulse width to 4/16
```

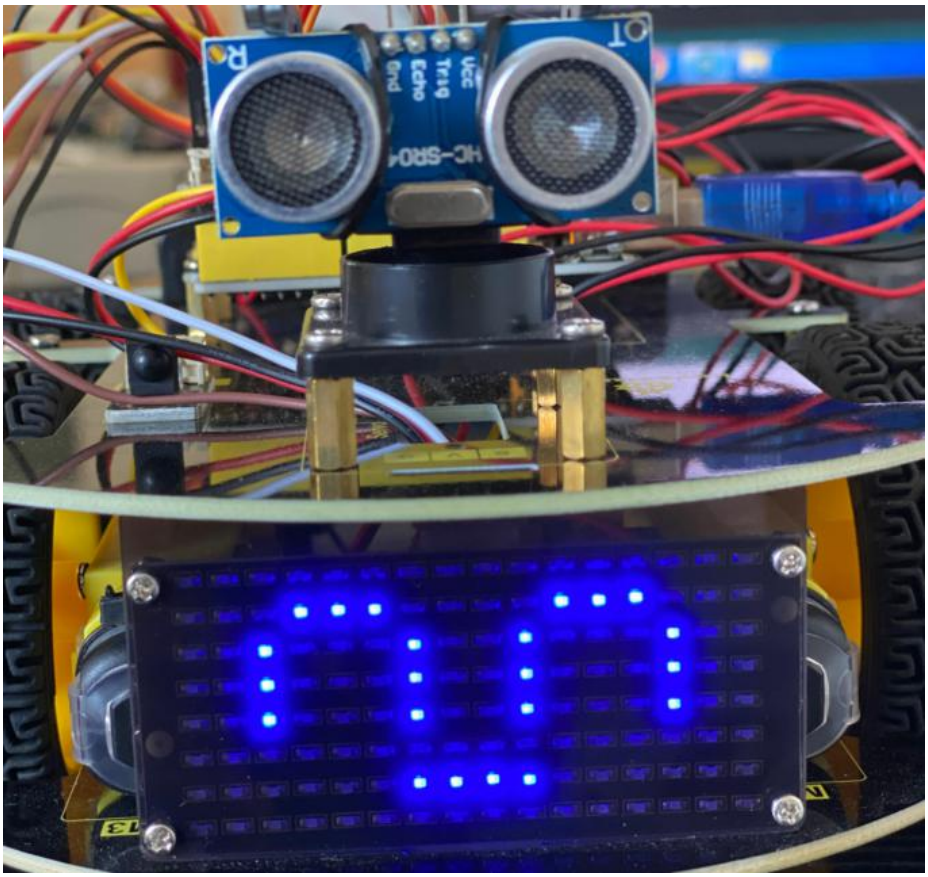
```
    IIC_end();
}
//    the condition that data transmission starts
void IIC_start()
{
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data)
{
    for(char i = 0;i < 8;i++)    //Every character has 8 bits
    {
        digitalWrite(SCL_Pin,LOW);    //pull down the SCL_Pin to change the
signal of SDA
        delayMicroseconds(3);
        if(send_data & 0x01)    //1 or 0 of byte    is used to set high and low
level of SDA_Pin
```

```
        {
            digitalWrite(SDA_Pin,HIGH);
        }
        else
        {
            digitalWrite(SDA_Pin,LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin,HIGH);  //Pull up SCL_Pin to stop data transmission
        delayMicroseconds(3);
        send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
    }
}
//the sign that data transmission ends
void IIC_end()
{
    digitalWrite(SCL_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
```

```
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
}//*****************************************************************************
*************
```

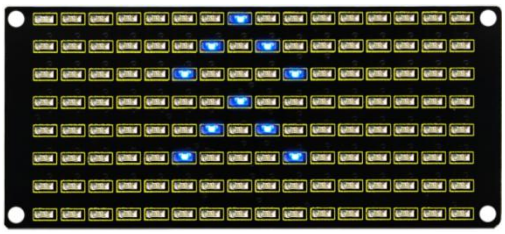
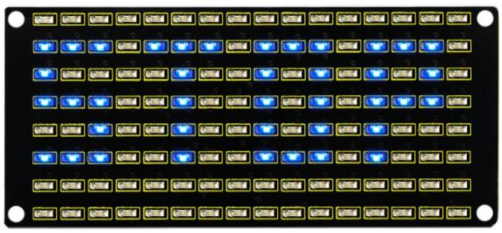
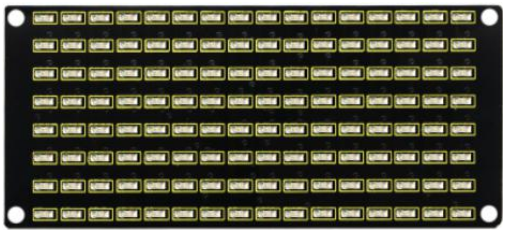
(7) Test Result

After uploading code on keyestudio V4.0 development board, hook up by the connection diagram, the DIP switch is dialed to right end, then a smile pattern is shown.



(8) Extension Practice

We use the modulo tool (http://dotmatrixtool.com/#)to make the dot matrix alternately display start, forward and stop patterns then clear the patterns, the time interval is 2000 milliseconds.

Get the graphical code to be displayed via modulus tool

Start :

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

Go front:

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

Go back:

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

Turn left:

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

Turn right:

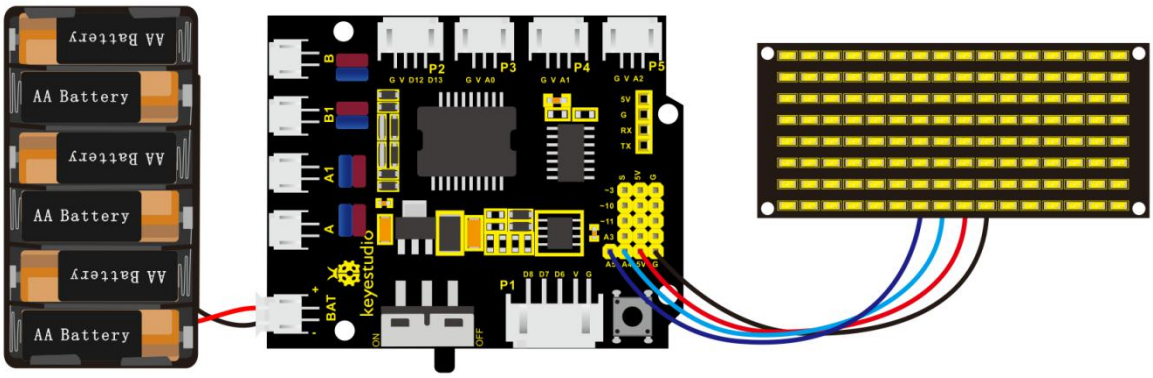
0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,
0x00,0x00

Stop:

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

Clear the matrix display:

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



The code that the multiple patterns shift:

```
/*
  keyestudio 4WD Robot v2.0
  lesson 9.2
  matrix
  http://www.keyestudio.com
*/
//Array, used to store the data of pattern, can be calculated by yourself or obtained from the modulus tool
unsigned char start01[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right[] =
```

```
{0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,
0x00,0x00};
unsigned char STOP01[] =
{0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,
0x0E,0x00};
unsigned char clear[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
void setup(){
  //Set pin to output
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the matrix display
  matrix_display(clear);
}
void loop(){
  matrix_display(start01);  //Display start pattern
  delay(2000);
  matrix_display(front);    ///Front pattern
  delay(2000);
```

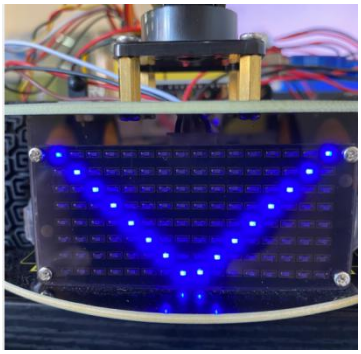
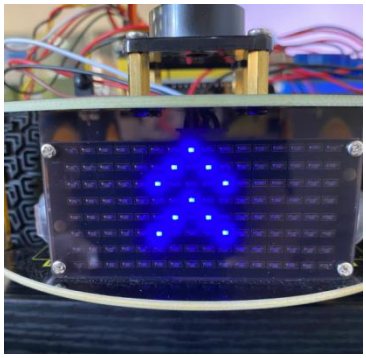
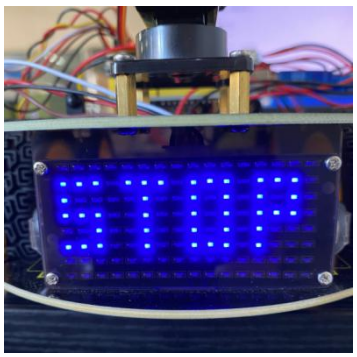
```
    matrix_display(STOP01);     //Stop pattern
    delay(2000);
    matrix_display(clear);      //Clear the matrix display
    delay(2000);
}

//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start();   //the function to call the data transmission
    IIC_send(0xc0);   //Select address
       for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //data to convey patterns
    }
    IIC_end();     //end the transmission of patterns data
    IIC_start();
    IIC_send(0x8A);   //display control, set pulse width to 4/16
    IIC_end();
}
//   the condition that data transmission starts
void IIC_start()
```

```
{
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data)
{
  for(char i = 0;i < 8;i++)  //Every character has 8 bits
  {
      digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the
signal of SDA
      delayMicroseconds(3);
      if(send_data & 0x01)  //1 or 0 of byte  is used to set high and low
level of SDA_Pin
      {
        digitalWrite(SDA_Pin,HIGH);
      }
      else
```

```
        {
            digitalWrite(SDA_Pin,LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin,HIGH);  //Pull up SCL_Pin to stop data transmission
        delayMicroseconds(3);
        send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
    }
}
//the sign that data transmission ends
void IIC_end()
{
    digitalWrite(SCL_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
```

```
}
//*****************************************************************************
```

Upload code on development board, 8*16 dot matrix display shows front , back and stop patterns, alternately.

Project 10: Line Tracking Robot

(1) Description

The previous projects are inclusive of the knowledge of multiple sensors and modules. Next, we will work on a little challenging task.

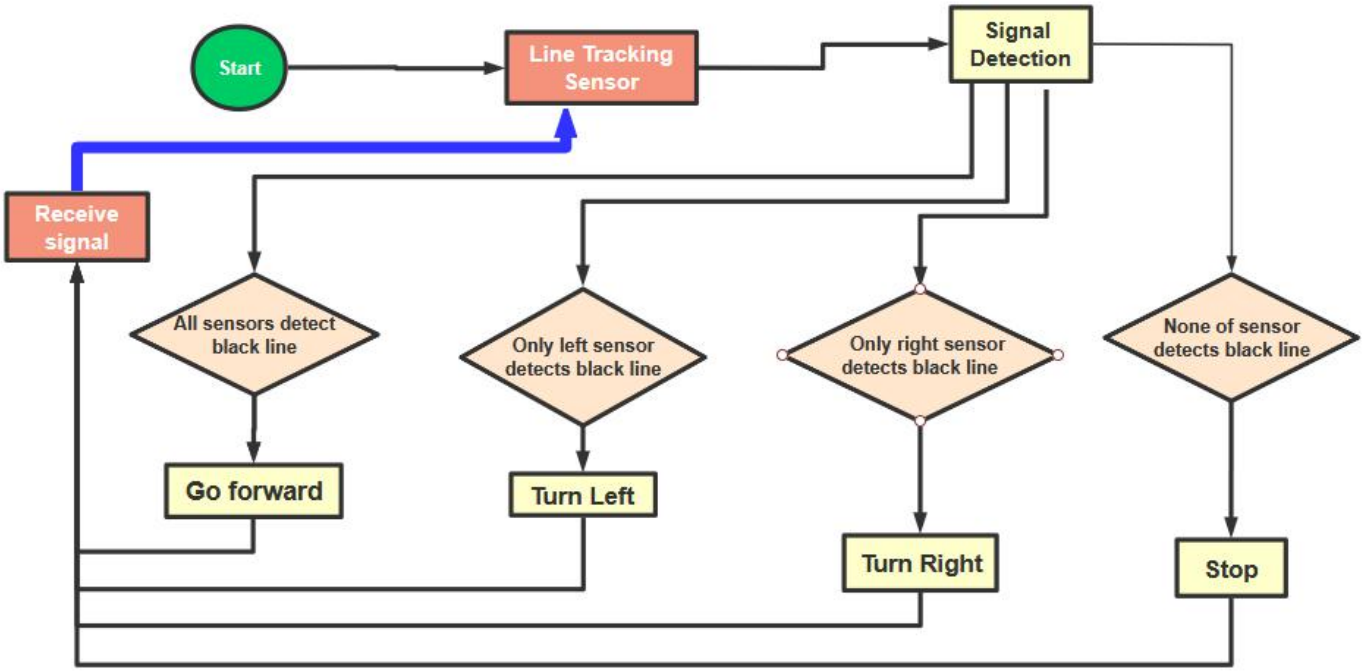
We could make a line tracking car on top of the working principle of line tracking sensor.

Line tracking robot car:

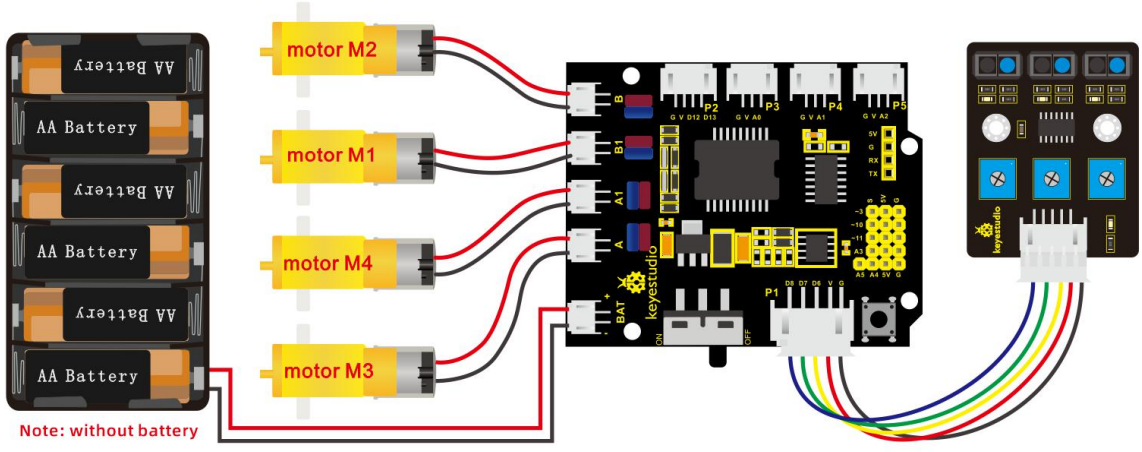
<table>
<tr><td rowspan="6">Detection</td><td rowspan="2">Left tracking sensor</td><td>detects black line：HIGH</td></tr>
<tr><td>detects white line：LOW</td></tr>
<tr><td rowspan="2">Middle tracking sensor</td><td>detects black line：HIGH</td></tr>
<tr><td>detects white line：LOW</td></tr>
<tr><td rowspan="2">Right tracking sensor</td><td>detects black line：HIGH</td></tr>
<tr><td>detects white line：LOW</td></tr>
<tr><td>Condition 1</td><td colspan="2">Status</td></tr>
<tr><td>Middle tracking sensor detects black line</td><td colspan="2">go front
(PWM set to 70)</td></tr>
<tr><td rowspan="4">Middle tracking sensor detects white line</td><td colspan="2">Status</td></tr>
<tr><td colspan="2">detecting the left and the right tracking sensor</td></tr>
<tr><td>Condition 2</td><td>Status</td></tr>
<tr><td>left tracking sensor detects black line; right sensor detects white line</td><td>Rotate to left
(PWM set to 200)</td></tr>
</table>

	left tracking sensor detects white line; right sensor detects black line	Rotate to right (PWM set to 200)
	left tracking sensor detects black line; right sensor detects black line	stop
	left tracking sensor detects white line; right sensor detects white line	stop

(2) Flow Chart



(3) Connection Diagram



(4) Test Code

```
/*
```

```
keyestudio 4wd BT Car V2.0
lesson 10
Line Tracking Robot
http://www.keyestudio.com
*/
#define ML_Ctrl 4        //define direction control pin of B motor
#define ML_PWM 5     //define PWM control pin of B motor
#define MR_Ctrl 2       //define direction control pin of A motor
#define MR_PWM 9     //define PWM control pin of A motor
const int sensor_l = 6;//define the pin of left line tracking sensor
const int sensor_c = 7;//define the pin of middle line tracking sensor
const int sensor_r = 8;//define the pin of right line tracking sensor
int l_val,c_val,r_val;//define these variables
void setup() {
  Serial.begin(9600);//start serial monitor and set baud rate to 9600
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to
OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to
OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to
OUTPUT
  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to
```

```
OUTPUT
  pinMode(sensor_l,INPUT);//set the pins of left line tracking sensor to
INPUT
  pinMode(sensor_c,INPUT);//set the pins of middle line tracking sensor to
INPUT
  pinMode(sensor_r,INPUT);//set the pins of right line tracking sensor to
INPUT
}
void loop()
{
  tracking(); //run main program
}

void tracking()
{
  l_val = digitalRead(sensor_l);//read the value of left line tracking sensor
  c_val = digitalRead(sensor_c);//read the value of middle line tracking
sensor
  r_val = digitalRead(sensor_r);//read the value of right line tracking sensor
  if(c_val == 1)//if the state of middle one is 1, which means detecting
black line
  {
```

```
      front();//car goes forward
    }
    else
    {
          if((l_val == 1)&&(r_val == 0))//if only left line tracking sensor
detects black trace
      {
        left();//car turns left
      }
     else if((l_val == 0)&&(r_val == 1))//if only right line tracking sensor
detects black trace
          {
        right();//car turns right
      }
      else// if line tracking sensors detect black trace or they don’ t
      {
        Stop();//car stops
      }
    }
}
void front()//define the status of going forward
{
```

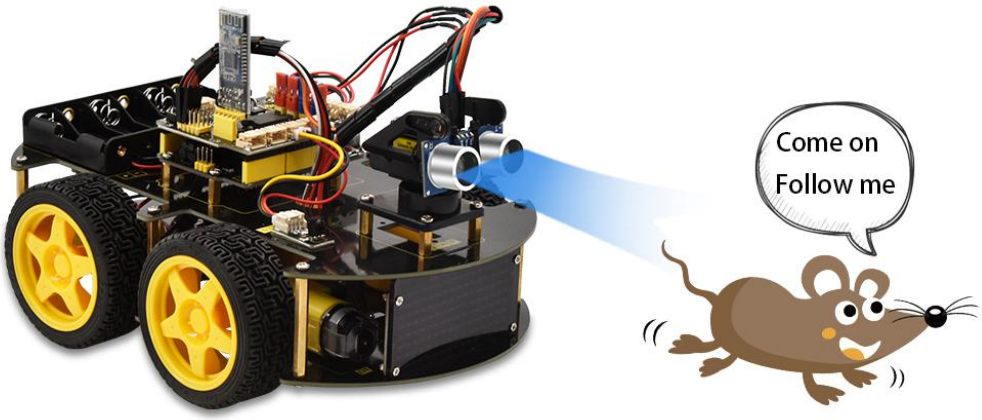
```
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,70);//set PWM control speed of B motor to 70
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH
  analogWrite(MR_PWM,70);//set PWM control speed of A motor to 70
}
void back()//define the state of going back
{
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void left()//car turns left
{
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to HIGH level
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void right()//define the right-turning state
```

```
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
level
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void Stop()//define the state of stop
{
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}//*********************************************************
```

(5) Test Result

Upload the code on the keyestudio V4.0 board successfully. Stack the expansion board on the keyestudio V4.0 board and wire it according to connection diagram. After power-on, the DIP switch will be dialed to the "ON" end, and the smart car can walk along the black line.

Project 11: Ultrasonic Follow Robot



(1) Description

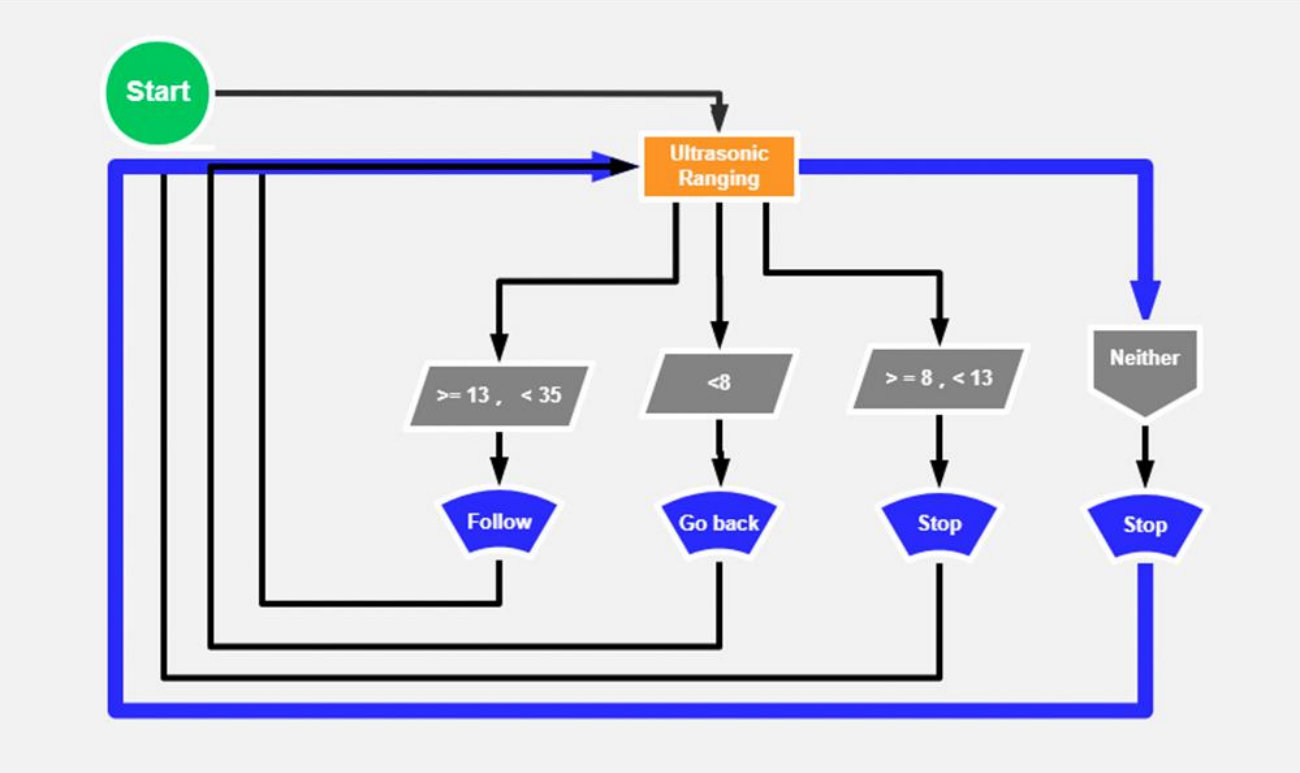
We combine the hardware knowledge -- various sensors, modules, motor drive, to build an ultrasonic follow robot car!

In the circuit process, we can make use of ultrasonic sensor to detect the distance between robot car and obstacles so as to control robot car to move by the measured distance value. And dot matrix shows smile face pattern.

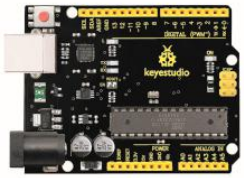
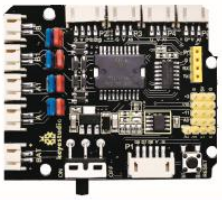
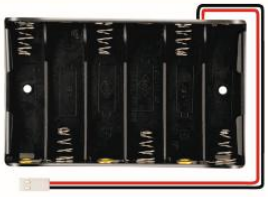
The specific logic of ultrasonic follow robot car is as shown below:

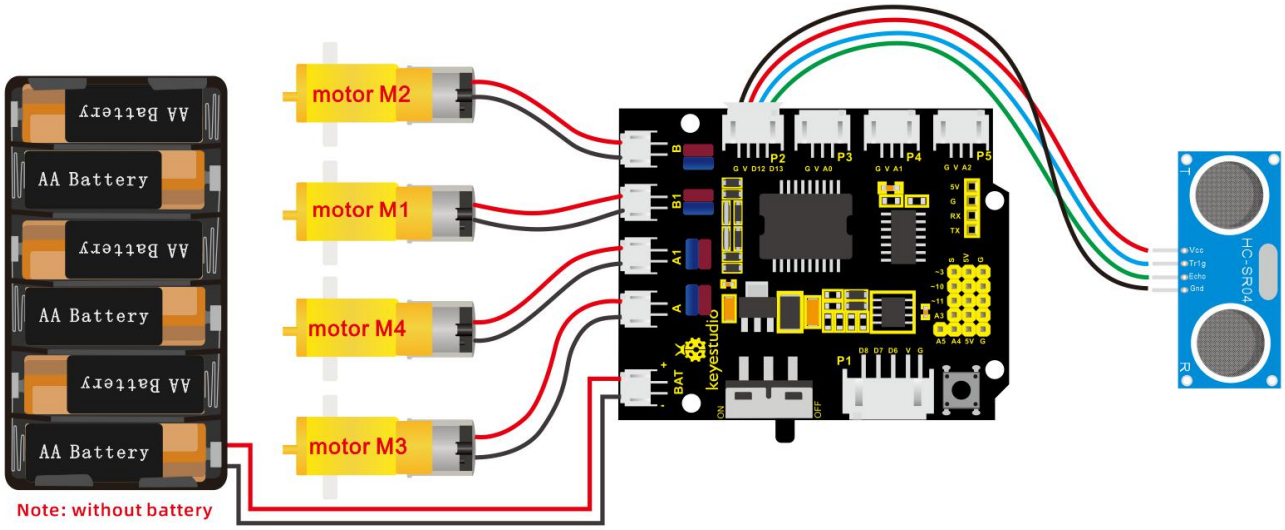
Detection	Measured distance of front obstacles	distance (unit: cm)
Condition	Distance < 8	
Status	Go back (PWM set to 100)	
Condition	distance≥8 and distance < 13	
Status	Stop	
Condition	distance≥13 and distance < 35	
Status	Go front (PWM set to 100)	
Condition	distance≥35	
Status	stop	

(2) Flow Chart



(3) Hook-up Diagram

Control Board *1	L298P Motor Shield *1	Ultrasonic module *1	Motor *4
			
USB Cable *1	6 AA battery Holder *1	4pin Dupont line *1	
			



(4) Test Code

```
/*
keyestudio 4wd BT Car V2.0
lesson 11
Ultrasonic Follow Robot
http://www.keyestudio.com
*/
#define ML_Ctrl 4        //define direction control pin of B motor
#define ML_PWM 5     //define PWM control pin of B motor
#define MR_Ctrl 2       //define direction control pin of A motor
#define MR_PWM 9     //define PWM control pin of A motor
#include "SR04.h" //define the function library of ultrasonic sensor
#define TRIG_PIN 12// set the signal input of ultrasonic sensor to D12
#define ECHO_PIN 13//set the signal output of ultrasonic sensor to D13
```

```
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance;
void setup() {
  Serial.begin(9600);//open serial monitor and set baud rate to 9600
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to
OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to
OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to
OUTPUT
  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to
OUTPUT
  pinMode(TRIG_PIN,OUTPUT);// set TRIG_PIN to OUTPUT
  pinMode(ECHO_PIN,INPUT);// set ECHO_PIN to INPUT
}
void loop() {
  distance = sr04.Distance();// the distance detected by ultrasonic sensor
   if(distance<8)//if distance is less than 8
  {
     back();//go back
  }
  else if((distance>=8)&&(distance<13))// if 8≤distance＜13
```

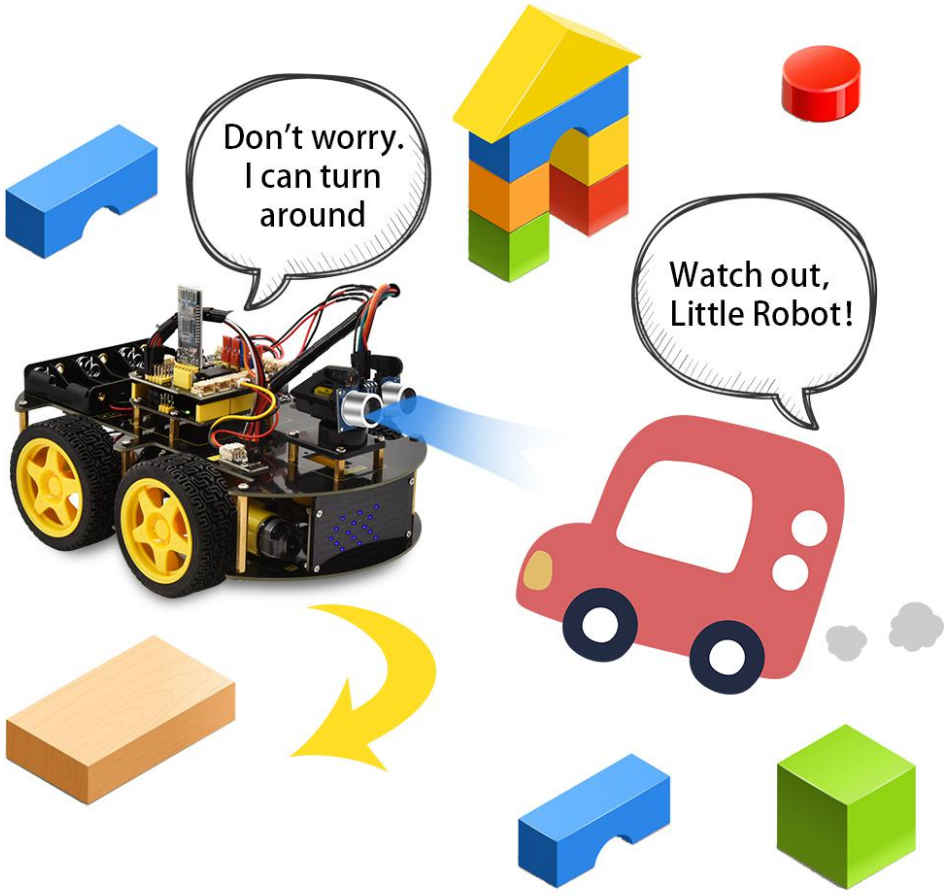
```
  {
    Stop();//stop
  }
  else if((distance>=13)&&(distance<35))//if 13≤distance < 35
  {
    front();//follow
  }
  else//otherwise
  {
    Stop();//stop
  }
}

void front()//go front
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,100);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH
  analogWrite(MR_PWM,100);//Set PWM control speed of A motor to 100
}
void back()//go back
```

```
{
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,100);//Set PWM control speed of B motor to 100
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,100);//Set PWM control speed of A motor to 100
}
void Stop()//stop
{
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}//*********************************************************
```

(5) Test Result

Upload the code to the development board, and plug in power，dot matrix will display smile face pattern and follow the obstacle to move.

Project 12: Ultrasonic Avoiding Robot



(1) Description

We' ve learned LED matrix, motor drive, ultrasonic sensor and servo in previous lessons. Next we could make an ultrasonic avoiding robot!

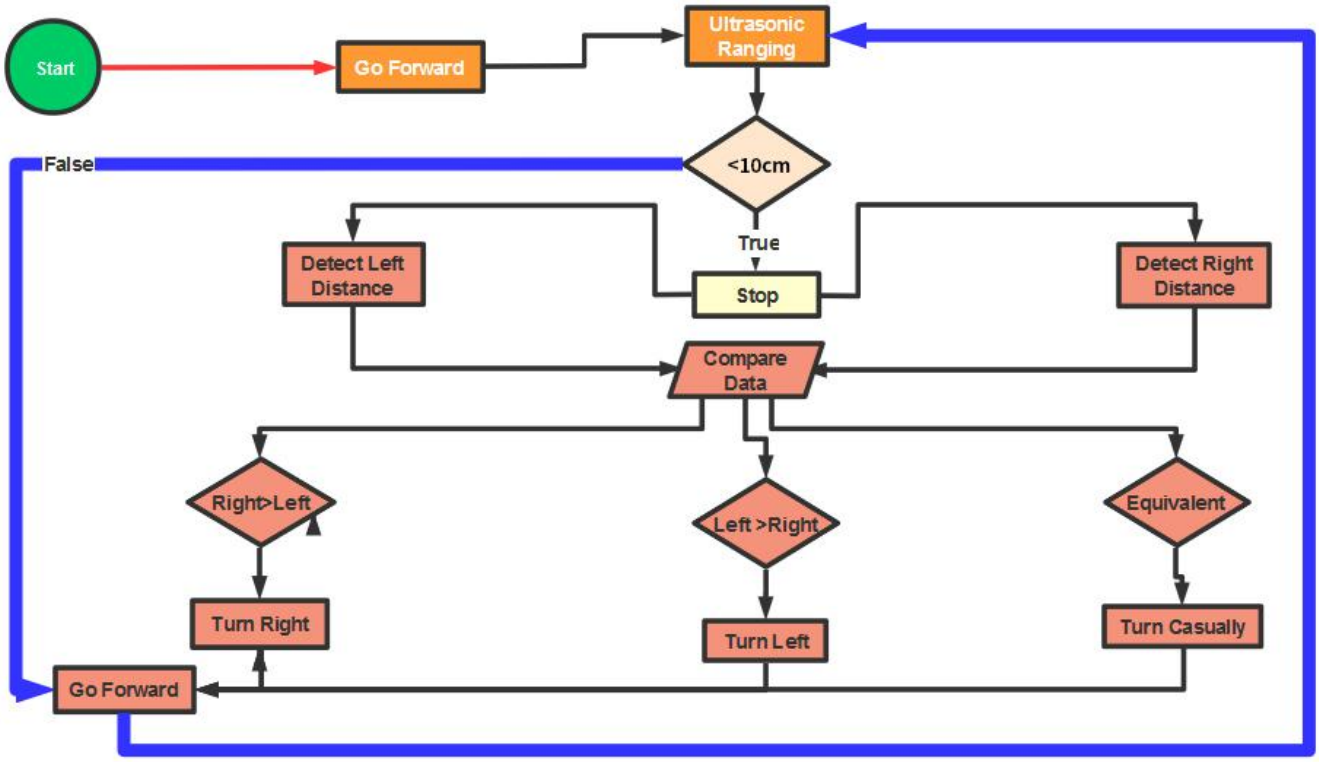
The measured distance between ultrasonic sensor and obstacle can be used to control servo to rotate so as to make robot car move.

The specific logic of ultrasonic avoiding smart car is as shown below:

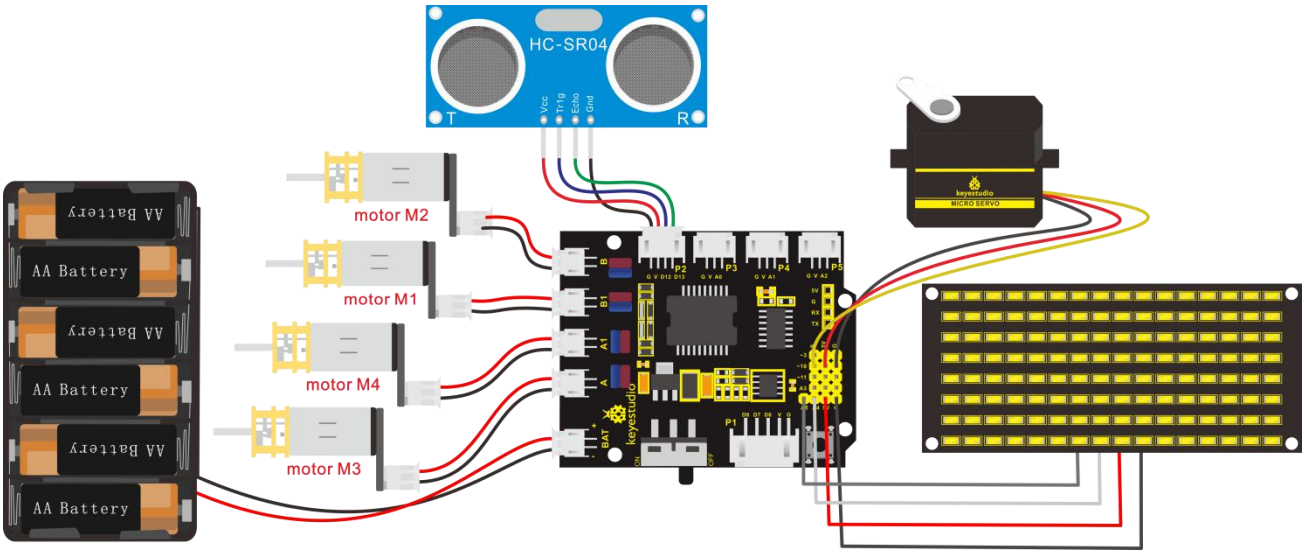
<table>
<tr><td>Initial Setup</td><td colspan="3">8x16 LED Matrix Clear</td></tr>
<tr><td></td><td colspan="3">Set servo to 90°</td></tr>
<tr><td rowspan="9">Loop program</td><td colspan="3">measured distance of front obstacle: distance (unit: cm)</td></tr>
<tr><td>Condition 1</td><td colspan="2">State</td></tr>
<tr><td rowspan="8">distance < 10</td><td colspan="2">Smart car stops</td></tr>
<tr><td colspan="2">8x16 LED matrix shows “stop” pattern</td></tr>
<tr><td>Set the servo to 180°</td><td>measured distance of obstacle: a1 (unit: cm)</td></tr>
<tr><td>Set the servo to 0°</td><td>measured distance of obstacle: a2 (unit: cm)</td></tr>
<tr><td>Condition 2</td><td>state</td></tr>
<tr><td rowspan="2">a1 < a2</td><td>rotate to right (PWM set to 200)</td></tr>
<tr><td>8x16 LED matrix shows “rightward” pattern</td></tr>
</table>

			Set the servo to 90°
		a1≥a2	rotate to left（PWM set to 200）
			8x16 LED matrix shows “ leftward ” pattern
			Set servo to 90°
	distance ≥ 10	8x16 LED matrix shows “forward” pattern	
		Go front（PWM set to 150）	

(2) Flow Chart



(3) Connection Diagram



(4) Test Code

```
/*
keyestudio 4wd BT Car V2.0
lesson 12
ultrasonic avoiding robot
http://www.keyestudio.com
*/
//Array, used to store the data of pattern, can be calculated by yourself or obtained from the modulus tool
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right[] = {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] = {0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
```

```
unsigned char clear[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
#define ML_Ctrl 4      //define direction control pin of B motor
#define ML_PWM 5    //define PWM control pin of B motor
#define MR_Ctrl 2     //define direction control pin of A motor
#define MR_PWM 9    //define PWM control pin of A motor
#include "SR04.h"//define the library of ultrasonic sensor
#define TRIG_PIN 12// set the signal input of ultrasonic sensor to D12
#define ECHO_PIN 13//set the signal output of ultrasonic sensor to D13
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance,a1,a2;//define three distance
const int servopin = 10;//set the pin of servo to D10

void setup() {
  Serial.begin(9600);//open serial monitor and set baud rate to 9600
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to
OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to
OUTPUT
```

```
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to
OUTPUT

  pinMode(MR_PWM, OUTPUT);//set PWM control pin of A motor to
OUTPUT
  servopulse(servopin,90);// the angle of servo is 90 degree
  delay(300);
  pinMode(SCL_Pin,OUTPUT);//   set clock pin to OUTPUT
  pinMode(SDA_Pin,OUTPUT);//set data pin to OUTPUT
  matrix_display(clear);// Clear the matrix display
}

void loop()
{
  avoid();//run the main program
}
void avoid()
{
  distance=sr04.Distance(); //obtain the value detected by ultrasonic
sensor
  if((distance < 20)&&(distance > 0))//if the distance is greater than 0 and
less than 20
```

```
{
    car_Stop();//stop
    matrix_display(STOP01);     //show stop pattern
    delay(100);
    servopulse(servopin,180);//servo rotates to 180°
    delay(500);
    a1=sr04.Distance();//measure the distance
    delay(100);
    servopulse(servopin,0);//rotate to 0 degree
    delay(500);
    a2=sr04.Distance();//measure the distance
    delay(100);
  if(a1 > a2)//if distance a1 is greater than a2
      {
     car_left();//turn left
     matrix_display(left);     //display left-turning pattern
     servopulse(servopin,90);//servo rotates to 90 degree
     delay(300);
     matrix_display(front);     //show forward pattern
    }
    else//if the right distance is greater than the left
    {
```

```
        car_right();// turn right
        matrix_display(right);     // display right-turning pattern
        servopulse(servopin,90);// servo rotates to 90 degree
        delay(300);
        matrix_display(front);     //show forward pattern
      }
    }
    else//otherwise
    {
      car_front();//go forward
      matrix_display(front);     // show forward pattern
    }
}
void servopulse(int servopin,int myangle)//the running angle of servo
{
    for(int i=0; i<30; i++)
    {
      int pulsewidth = (myangle*11)+500;
      digitalWrite(servopin,HIGH);
      delayMicroseconds(pulsewidth);
      digitalWrite(servopin,LOW);
      delay(20-pulsewidth/1000);
```

```
    }
}
void car_front()//car goes forward
{
    digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
level
    analogWrite(ML_PWM,150);//set PWM control speed of B motor to 150
    digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH level
    analogWrite(MR_PWM,150);//set PWM control speed of A motor to 150
}
void car_back()//go back
{
    digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
    analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
    digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
    analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_left()//car turns left
{
    digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
    analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
```

```
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_right()//car turns right
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_Stop()//car stops
{
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,150);
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,150);
  delay(50);
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
//this function is used for dot matrix display
```

```
void matrix_display(unsigned char matrix_value[])
{
  IIC_start();  //the function to call the data transmission
  IIC_send(0xc0);  //Select address
  for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
  {
    IIC_send(matrix_value[i]); //data to convey patterns
  }
  IIC_end();    //end the transmission of patterns data
  IIC_start();
  IIC_send(0x8A);  //display control, set pulse width to 4/16
  IIC_end();
}
//  the condition that data transmission starts
void IIC_start()
{
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
```

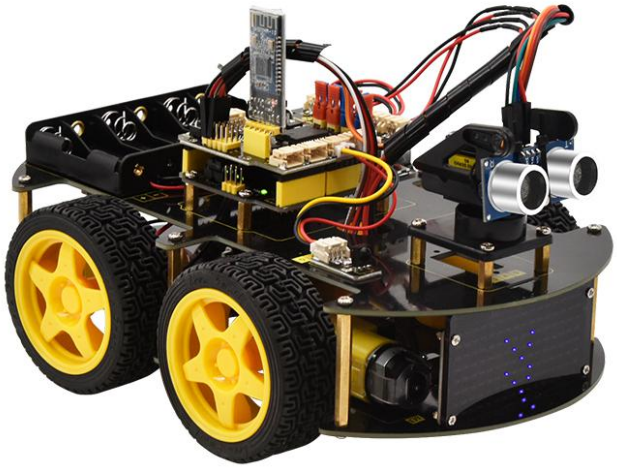
```
}
// transmit data
void IIC_send(unsigned char send_data)
{
  for(char i = 0;i < 8;i++)  //Every character has 8 bits
  {
    digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the signal of SDA
    delayMicroseconds(3);
    if(send_data & 0x01)  //1 or 0 of byte  is used to set high and low level of SDA_Pin
    {
      digitalWrite(SDA_Pin,HIGH);
    }
    else
    {
      digitalWrite(SDA_Pin,LOW);
    }
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH);  //Pull up SCL_Pin to stop data transmission
    delayMicroseconds(3);
```

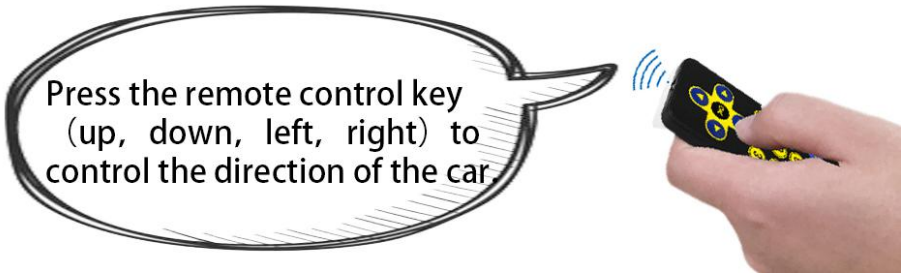
```
    send_data = send_data >> 1;    //Detect bit by bit, so move the data
right by one bit
  }
}
//the sign that data transmission ends
void IIC_end()
{
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
}//*********************************************************
```

(5) Test Result

After uploading the code on the keyestudio V4.0 board, wire according to connection diagram. After the DIP switch is dialed to the right end, the smart car can automatically avoid obstacles.

Project 13: IR Remote Control Robot





(1) Description

In this project, we will make IR remote control robot car!

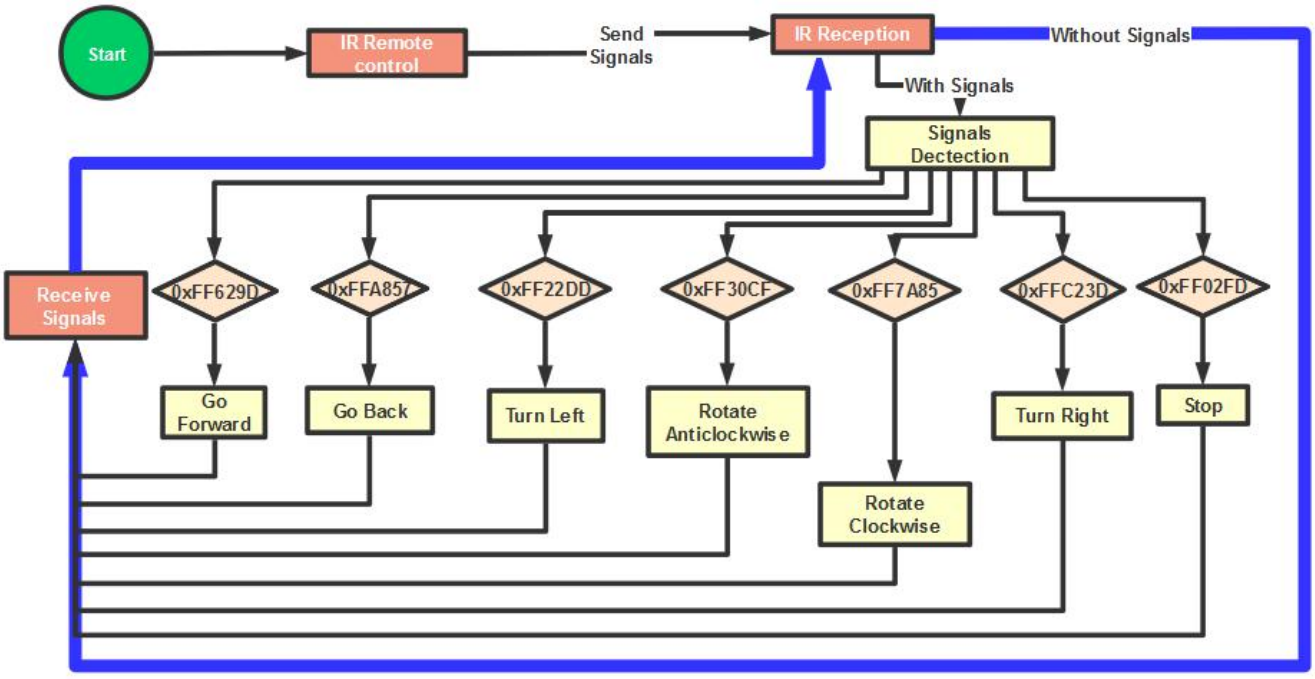
Press the button on IR remote control to drive robot car to move, and the corresponding state pattern is displayed on the 8*16 LED matrix.

(2) Flow Chart

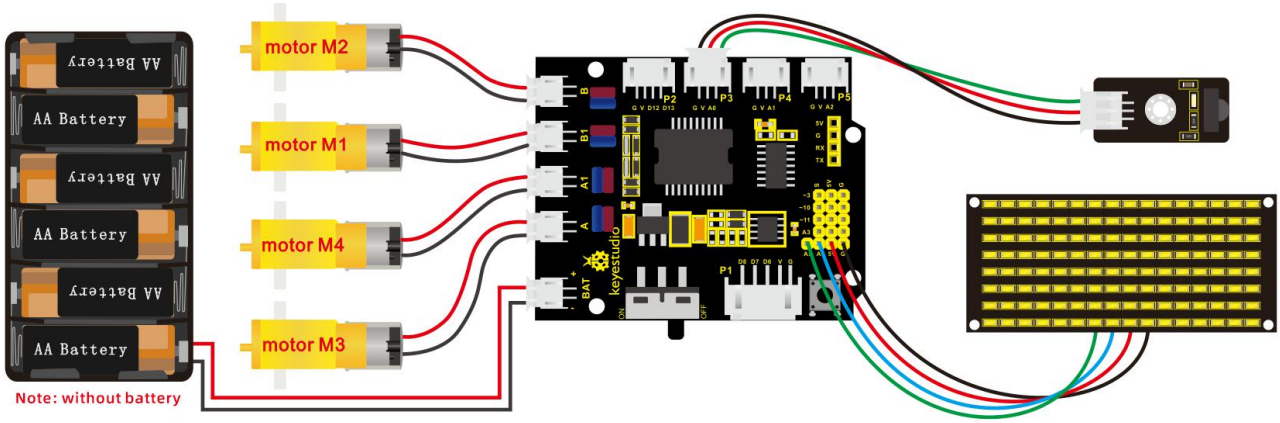
The specific logic of infrared remote control robot car is shown below:

Initial setup	8X16 LED matrix	
Remote control	Key Value	Key state
	FF629D	Go front（PWM set to 100）
		8*16 LED matrix shows front icon
	FFA857	Back（PWM set to 100）
		8*16 LED matrix shows back icon
	FF22DD	Rotate to left（PWM set to 200）
		8X16 LED matrix shows leftward icon
	FFC23D	Rotate to right（PWM set to 200）
		8X16 LED matrix shows rightward icon
	FF02FD	Stop
		8X16 LED matrix shows “STOP”

Based on the circuit design, we can start building our own remote control robot.



(3) Hook-up Diagram



(4) Test Code

```
/*
keyestudio 4wd BT Car V2.0
lesson 13
remote control robot
```

```
http://www.keyestudio.com
*/
//Array, used to store the data of pattern, can be calculated by yourself or obtained from the modulus tool
unsigned char start01[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right[] = {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] = {0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
```

```
unsigned char clear[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
#define ML_Ctrl 4      //define direction control pin of B motor
#define ML_PWM 5    //define PWM control pin of B motor
#define MR_Ctrl 2     //define direction control pin of A motor
#define MR_PWM 9    //define PWM control pin of A motor
#include <IRremote.h>//function library of IR remote control
int RECV_PIN = A0;// set the pin of IR receiver to A0
IRrecv irrecv(RECV_PIN);
long irr_val;
decode_results results;
void setup()
{
  pinMode(ML_Ctrl, OUTPUT);//define direction control pin of B motor to
OUTPUT
  pinMode(ML_PWM, OUTPUT);//define PWM control pin of B motor to
OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//define direction control pin of A motor to
OUTPUT
```

```
  pinMode(MR_PWM, OUTPUT);//define PWM control pin of A motor to
OUTPUT
    Serial.begin(9600);//Start serial printing, baud rate is 9600
  // In case the interrupt driver crashes on setup, give a clue
  // to the user what's going on.
  irrecv.enableIRIn(); // Start the receiver
  Serial.println("Enabled IRin");
  //Set pin to output
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the matrix display
  matrix_display(clear);
  matrix_display(start01);
}
void loop()
 {
  if (irrecv.decode(&results))
 {
    irr_val = results.value;
    Serial.println(irr_val, HEX);//serial reads the IR remote signals
    switch(irr_val)
    {
```

```
        case 0xFF629D : car_front(); matrix_display(front); break;
        case 0xFFA857 : car_back(); matrix_display(back); break;
        case 0xFF22DD : car_left(); matrix_display(left); break;
        case 0xFFC23D : car_right(); matrix_display(right); break;
        case 0xFF02FD : car_Stop(); matrix_display(STOP01); break;
      }
          irrecv.resume(); // Receive the next value
   }
}
void car_front()//car goes forward
{
   digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
level
   analogWrite(ML_PWM,200);//Set PWM control speed of B motor to 20
   digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH level
   analogWrite(MR_PWM,200);//Set PWM control speed of A motor to 20
}
void car_back()//car goes back
{
   digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
   analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
```

```
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_left()//car turns left
{
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH level
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_right()//car turns right
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
level
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_Stop()//car stops
{
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
```

```
    analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start();   //the function to call the data transmission
    IIC_send(0xc0);   //Select address
       for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //data to convey patterns
    }
    IIC_end();     //end the transmission of patterns data
    IIC_start();
    IIC_send(0x8A);   //display control, set pulse width to 4/16
    IIC_end();
}
//   the condition that data transmission starts
void IIC_start()
{
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
```

```
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data)
{
    for(char i = 0;i < 8;i++)   //Every character has 8 bits
    {
        digitalWrite(SCL_Pin,LOW);   //pull down the SCL_Pin to change the
signal of SDA
        delayMicroseconds(3);
        if(send_data & 0x01)   //1 or 0 of byte   is used to set high and low
level of SDA_Pin
        {
            digitalWrite(SDA_Pin,HIGH);
        }
        else
        {
            digitalWrite(SDA_Pin,LOW);
        }
        delayMicroseconds(3);
```

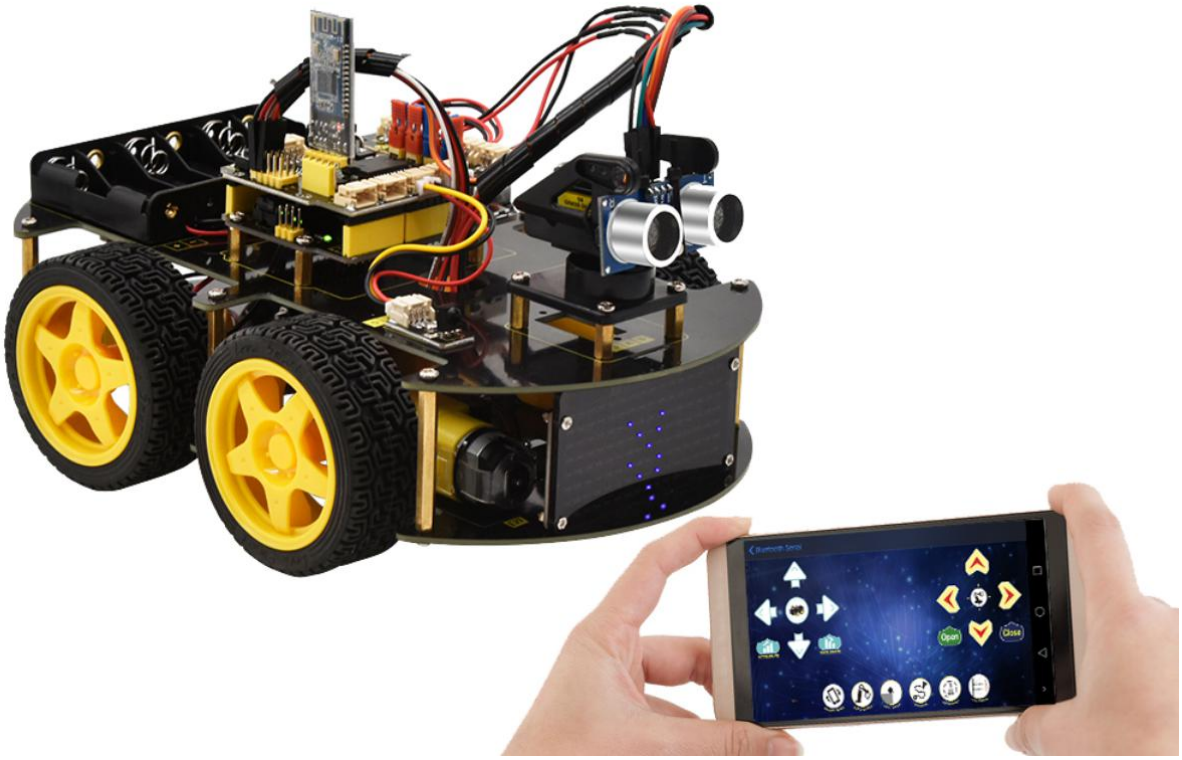
```
        digitalWrite(SCL_Pin,HIGH);  //Pull up SCL_Pin to stop data
transmission
        delayMicroseconds(3);
        send_data = send_data >> 1;   //Detect bit by bit, so move the data
right by one bit
    }
}
//the sign that data transmission ends
void IIC_end()
{
    digitalWrite(SCL_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
}//*******************************************************
```

(5) Test Result

After uploading the code successfully on the keyestudio V4.0 board, wire according to the connection diagram, after DIP switch is dialed to the right

end, we can use the infrared remote control to control the smart car movement. At the same time, the 8X16 LED light board displays the corresponding state pattern.

Project 14: Bluetooth Remote Control



(1) Description

We’ ve learned the basic knowledge of Bluetooth, in this lesson, we will make a Bluetooth remote smart car. In the experiment, we default the HM-10 Bluetooth module as a Slave and the cellphone as a Host.

keyes BT car is an APP rolled out by keyestudio team. You can control the

robot car by it readily.

(2) Test APP

Special note: Before uploading the test code, you need to remove the Bluetooth module, otherwise the test code will fail to upload. After the code is uploaded successful, then reconnect the Bluetooth module.

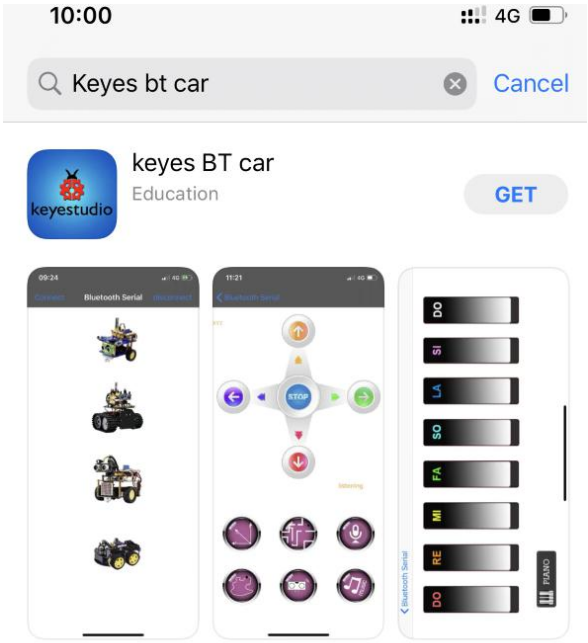
```
/*
keyestudio 4WD BT Car V2.0
lesson 14.1
Bluetooth test
http://www.keyestudio.com
*/
char BLE_val;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  if(Serial.available()>0)
  {
    BLE_val = Serial.read();
```

```
        Serial.println(BLE_val);
    }
}
//*************************************************************
```

Upload test code on V4.0 development board and insert the Bluetooth module. Then we need to download APP.

For iOS system

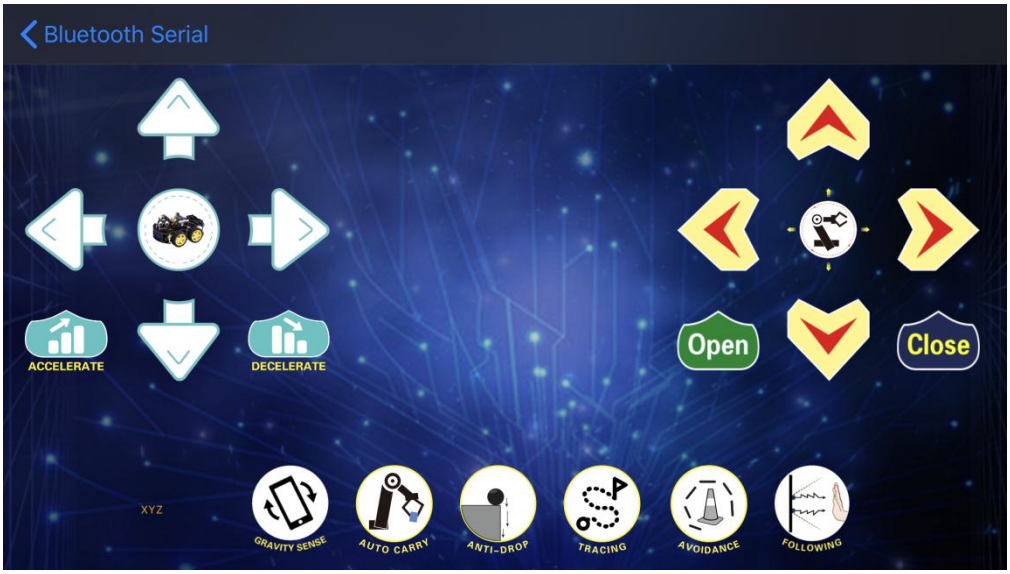
Search keyes BT car in App store



After installation, enter its interface.

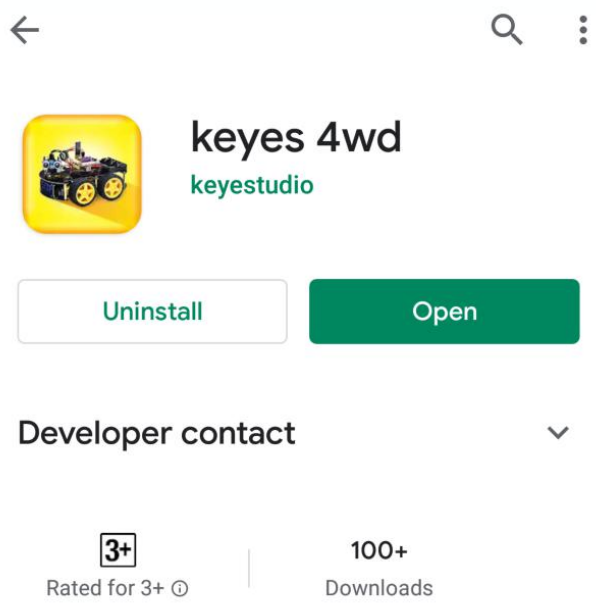
Click “Connect” to search and pair Bluetooth. After connecting well, click

to enter the main page of 4WD smart car.

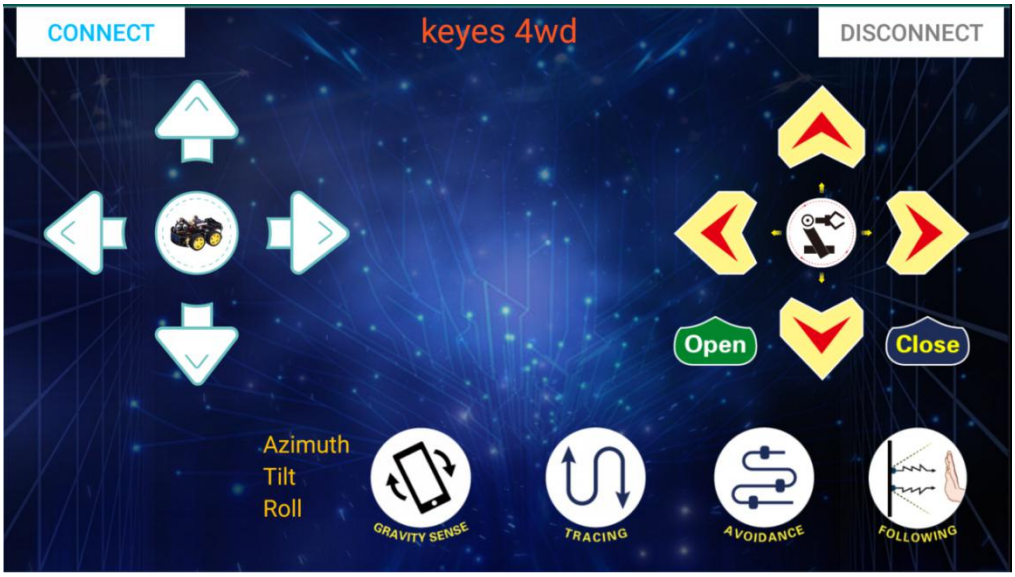


For Android System

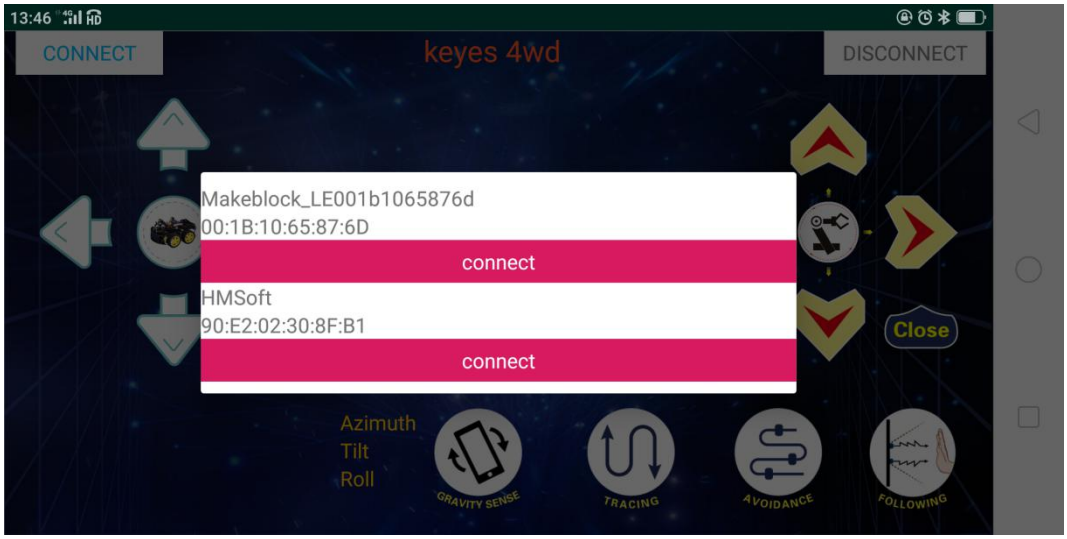
Enter Google play store to search keyes 4wd



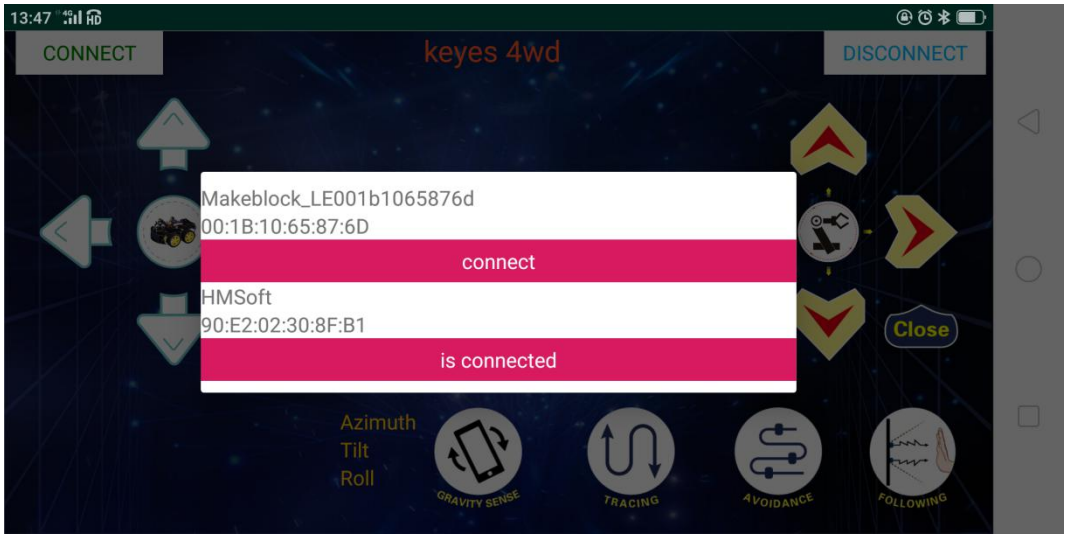
Its interface is shown below:



(3) Click on APP CONNECT icon to search Bluetooth.



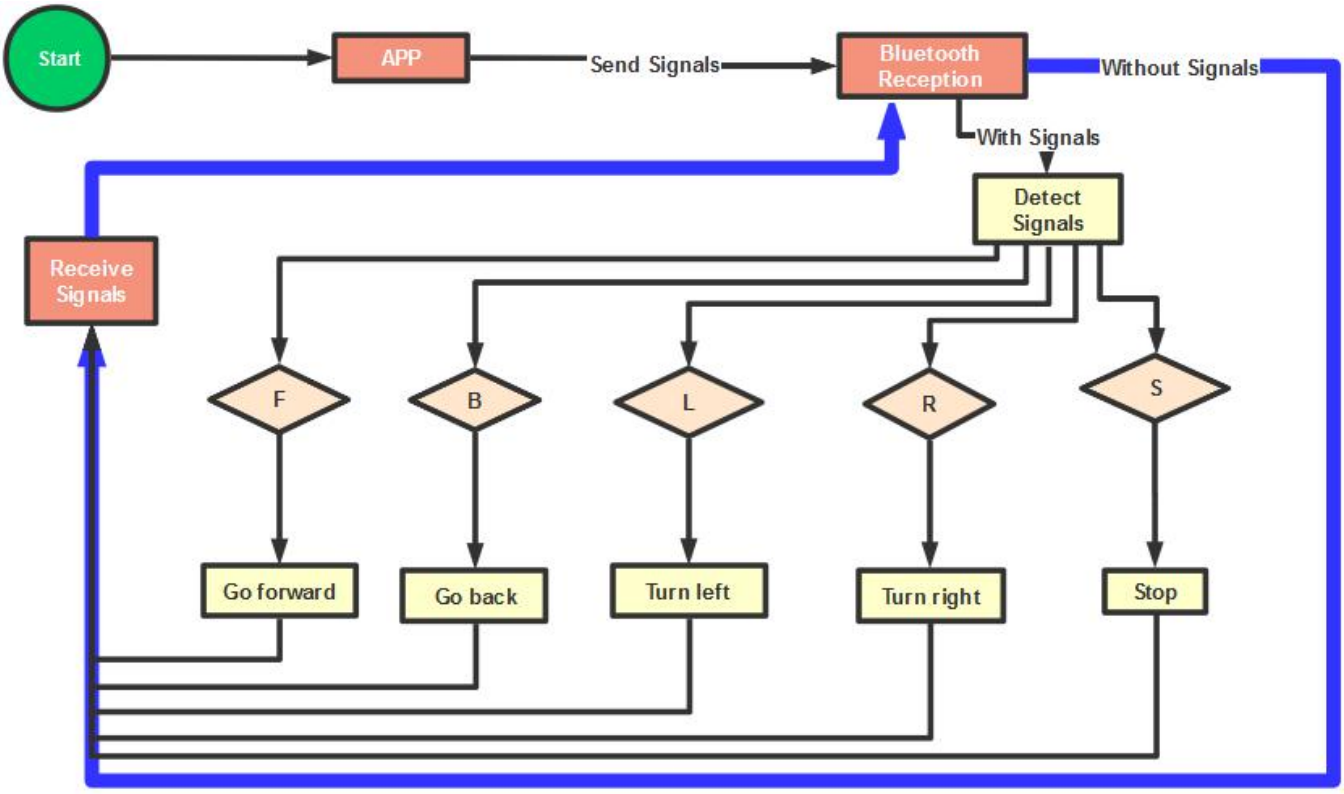
(4) Click "connect" below HMSoft, then the Bluetooth will be connected and its LED indicator will be always on.



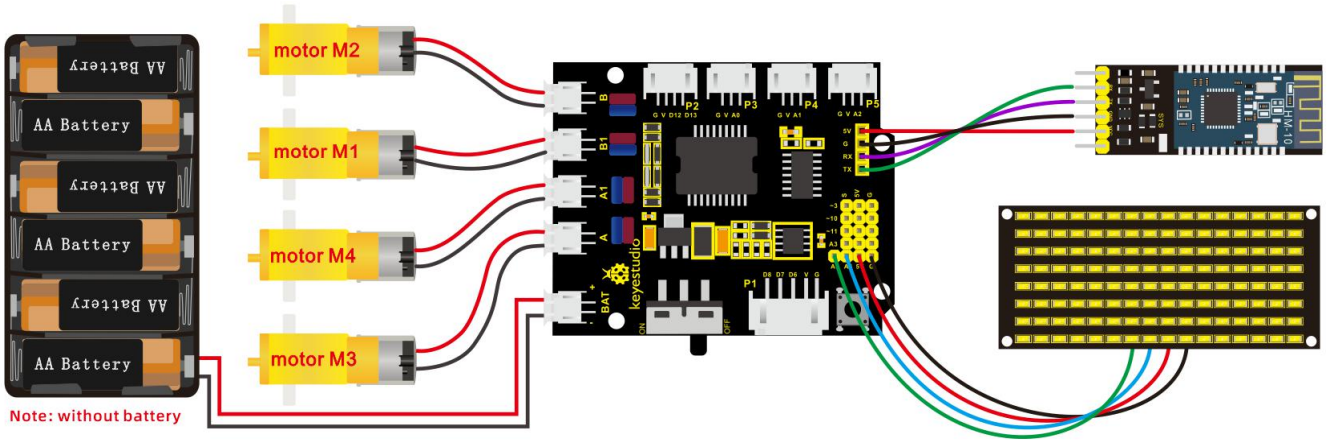
After successful connection, press the button of the Bluetooth APP, and the corresponding characters are displayed as shown below:

Key	Function	
CONNECT	match with connection HM-10 Bluetooth module	
DISCONNECT	disconnect Bluetooth	
	Control character	Function
	Press: F Release: S	robot car goes front; Release to stop
	Press: L Release: S	Robot car turns left; Release to stop
	Press: R Release: S	Robot car turns right; Release to stop
	Press: B Release: S	Robot car goes back; Release to stop
	Click to start the mobile gravity sensing; click again to end this function	
	Click to send “X”; Release to send “S”	Enable line tracking function; End this function
	Click to send “Y”; Release to send “S”	Start ultrasonic avoiding function; End this function
	Click to send “U” Release to send “S”	Start Ultrasonic follow function; End this function

(3) Flow Chart



(4) Hook-up Diagram



(5) Test Code

```
/*
keyestudio 4wd BT Car V2.0
lesson 14
Bluetooth Remote Control
http://www.keyestudio.com
*/
//Array, used to store the data of pattern, can be calculated by yourself or obtained from the modulus tool
unsigned char start01[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
```

```
unsigned char back[] =
{0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,
0x00,0x00};
unsigned char left[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,
0x10,0x00};
unsigned char right[] =
{0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,
0x00,0x00};
unsigned char STOP01[] =
{0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,
0x0E,0x00};
unsigned char clear[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};
#define SCL_Pin  A5  //Set clock pin to A5
#define SDA_Pin  A4  //Set data pin to A4
unsigned char data_line = 0;
unsigned char delay_count = 0;
#define ML_Ctrl 4      //define direction control pin of B motor
#define ML_PWM 5    //define PWM control pin of B motor
#define MR_Ctrl 2     //define direction control pin of A motor
```

```
#define MR_PWM 9     //define PWM control pin of A motor
char BLE_val;
void setup()
{
  Serial.begin(9600);
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(MR_PWM, OUTPUT);//Set PWM control pin of A motor to OUTPUT
//Set pin to output
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the matrix display
  matrix_display(clear);
  matrix_display(start01);
}

void loop()
```

```
{
  if(Serial.available()>0)
  {
    BLE_val = Serial.read();
    Serial.println(BLE_val);
  }
  switch(BLE_val)
  {
    case 'F': car_front(); matrix_display(front); break;
    case 'B': car_back(); matrix_display(back); break;
    case 'L': car_left(); matrix_display(left); break;
    case 'R': car_right(); matrix_display(right); break;
    case 'S': car_Stop();matrix_display(STOP01); break;
  }

}
void car_front()
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH
```

```
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_back()
{
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_left()
{
  digitalWrite(ML_Ctrl,LOW);//set direction control pin of B motor to LOW
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set direction control pin of A motor to
HIGH
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_right()
{
  digitalWrite(ML_Ctrl,HIGH);//set direction control pin of B motor to HIGH
  analogWrite(ML_PWM,200);//set PWM control speed of B motor to 200
  digitalWrite(MR_Ctrl,LOW);//set direction control pin of A motor to LOW
```

```
  analogWrite(MR_PWM,200);//set PWM control speed of A motor to 200
}
void car_Stop()
{
  analogWrite(ML_PWM,0);//set PWM control speed of B motor to 0
  analogWrite(MR_PWM,0);//set PWM control speed of A motor to 0
}
//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
  IIC_start();  //the function that calls the data transmission
  IIC_send(0xc0);  //Select address
    for(int i = 0;i < 16;i++) //Pattern data has 16 bytes
  {
      IIC_send(matrix_value[i]); //data to convey patterns
  }
  IIC_end();    //end the transmission of patterns data
  IIC_start();
  IIC_send(0x8A);  //display control, set pulse width to 4/16  IIC_end();
}
// the condition of data transmission starts
void IIC_start()
```

```
{
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data)
{
  for(char i = 0;i < 8;i++)  //Every character has 8 bits
  {
      digitalWrite(SCL_Pin,LOW);  //pull down the SCL_Pin to change the
signal of SDA
      delayMicroseconds(3);
      if(send_data & 0x01)  //1 or 0 of byte is used to set high and low
level of SDA_Pin
      {
        digitalWrite(SDA_Pin,HIGH);
      }
      else
```

```
        {
          digitalWrite(SDA_Pin,LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin,HIGH);  //Pull up SCL_Pin to stop data transmission
        delayMicroseconds(3);
        send_data = send_data >> 1;  //Detect bit by bit, so move the data right by one bit
    }
}
//the sign that data transmission ends
void IIC_end()
{
    digitalWrite(SCL_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
```

```
} //*************************************************************************
```

(6) Test Result

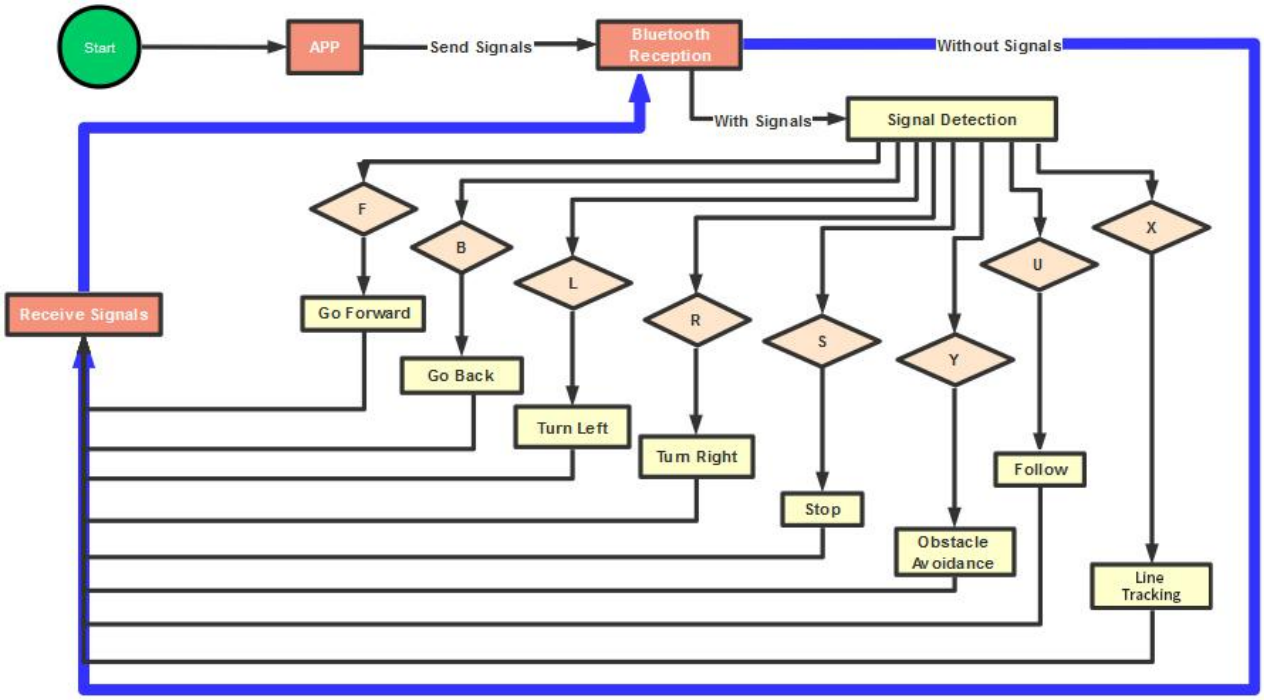
Uploading the code V4.0 board. We stack the expansion board on it and wire them according to the connection diagram. After power-on, the DIP switch will be dialed to the “ON” end. After connecting Bluetooth successfully, we can use APP to control the smart car to move.

Project 15: Multi-purpose Bluetooth Robot

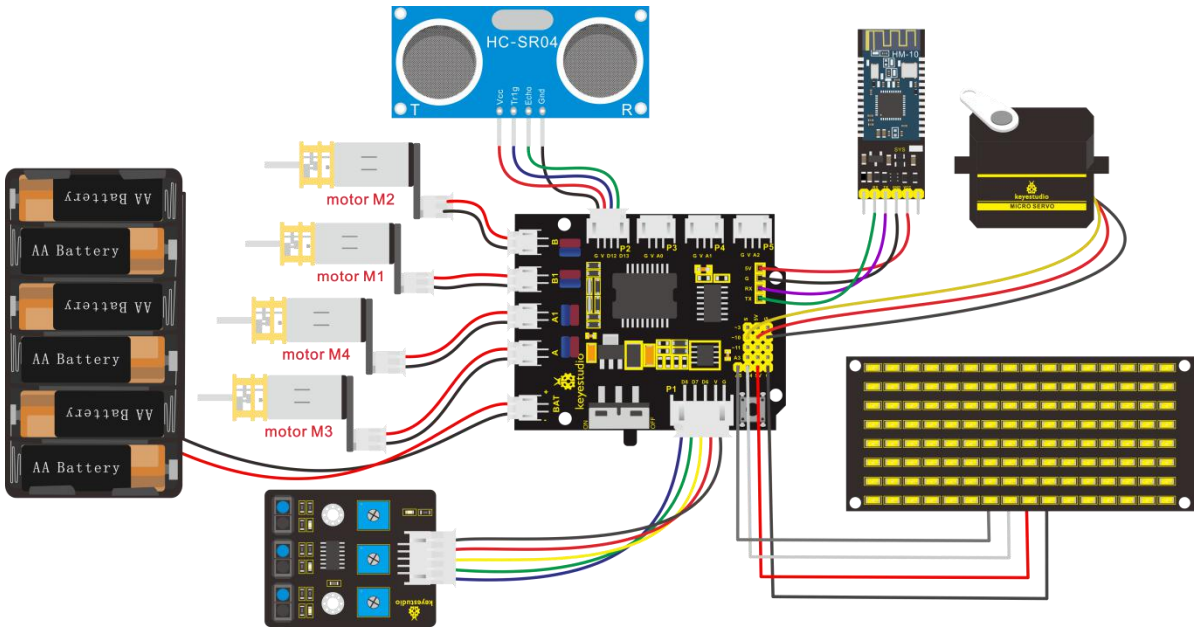
(1) Description

In previous projects, the robot car only performs single function, however, in this lesson, we integrate all of function to control smart car via Bluetooth control.

Here is a simple flow chart of multi-purpose robot car as for your reference.



(2) Connection Diagram



(3) Test Code

```
/*
keyestudio 4wd BT Car V2.0
lesson 15
Multifunctional Robot car
http://www.keyestudio.com
*/
unsigned char start01[] =
{0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,
0x02,0x01};
unsigned char front_matrix[] =
{0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,
0x00,0x00};
unsigned char back_matrix[] =
{0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,
0x00,0x00};
unsigned char left_matrix[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,
0x10,0x00};
unsigned char right_matrix[] =
{0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,
```

```
0x00,0x00};
unsigned char STOP01[] =
{0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,
0x0E,0x00};
unsigned char clear[] =
{0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00};

#define SCL_Pin  A5
#define SDA_Pin  A4

#include "SR04.h"
#define TRIG_PIN 12
#define ECHO_PIN 13
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance,distance1,distance2,distance3;

const int left_ctrl = 4;
const int left_pwm = 5;
const int right_ctrl = 2;
const int right_pwm = 9;
const int sensor_l = 6;
```

```
const int sensor_c = 7;
const int sensor_r = 8;
int l_val,c_val,r_val;
const int servopin = 10;
char BLE_val;

void setup() {
  Serial.begin(9600);
  //irrecv.enableIRIn(); // Start the receiver
  servopulse(servopin,90);
  pinMode(left_ctrl,OUTPUT);
  pinMode(left_pwm,OUTPUT);
  pinMode(right_ctrl,OUTPUT);
  pinMode(right_pwm,OUTPUT);
  pinMode(sensor_l,INPUT);
  pinMode(sensor_c,INPUT);
  pinMode(sensor_r,INPUT);
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  //Clear the screen
  matrix_display(clear);
  matrix_display(start01);
```

```
}

void loop() {
  if(Serial.available()>0)
  {
    BLE_val = Serial.read();
    Serial.println(BLE_val);
  }
  switch(BLE_val)
  {
    case 'F': front(); matrix_display(front_matrix); break;
    case 'B': back(); matrix_display(back_matrix); break;
    case 'L': left(); matrix_display(left_matrix); break;
    case 'R': right(); matrix_display(right_matrix); break;
    case 'S': Stop(); matrix_display(STOP01); break;
    case 'X': tracking(); break;
    case 'Y': avoid();break;
    case 'U': follow_car();break;
  }
}

void avoid()
```

```
{
  matrix_display(start01);
  int track_flag = 0;
  while(track_flag == 0)
  {
    distance1=sr04.Distance();
    if((distance1 < 20)&&(distance1 != 0))
    {
      Stop2();
      delay(100);
      servopulse(servopin,180);
      delay(500);
      distance2=sr04.Distance();
      delay(100);
      servopulse(servopin,0);
      delay(500);
      distance3=sr04.Distance();
      delay(100);
        if(distance2 > distance3)
      {
        left();
        servopulse(servopin,90);
```

```
        }
        else
        {
          right();
          servopulse(servopin,90);
        }
      }
      else
      {
        front();
      }
      if(Serial.available()>0)
      {
        BLE_val = Serial.read();
        if(BLE_val == 'S')
        {
          track_flag = 1;
        }
      }
    }
}
```

```
void follow_car()
{
  matrix_display(start01);
  servopulse(servopin,90);
  int track_flag = 0;
  while(track_flag == 0)
  {
    distance = sr04.Distance();

    if(distance<8)
    {
      back2();
    }
    else if((distance>=8)&&(distance<13))
    {
      Stop();
    }
    else if((distance>=13)&&(distance<35))
    {
      front();
    }
    else
```

```
      {
        Stop();
      }
      if(Serial.available()>0)
      {
        BLE_val = Serial.read();
        if(BLE_val == 'S')
        {
          track_flag = 1;
        }
      }
    }
}

void servopulse(int servopin,int myangle)
{
  for(int i=0;i<30;i++){
    int pulsewidth = (myangle*11)+500;
    digitalWrite(servopin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delay(20-pulsewidth/1000);
```

```
    }
}

void tracking()
{
    matrix_display(start01);
    int track_flag = 0;
    while(track_flag == 0)
    {
        l_val = digitalRead(sensor_l);
        c_val = digitalRead(sensor_c);
        r_val = digitalRead(sensor_r);

        if(c_val == 1)
        {
            front2();
        }
        else
        {
            if((l_val == 1)&&(r_val == 0))
            {
                left();
```

```
        }
        else if((l_val == 0)&&(r_val == 1))
        {
          right();
        }
        else
        {
          Stop();
        }
      }
      if(Serial.available()>0)
      {
        BLE_val = Serial.read();
        if(BLE_val == 'S')
        {
          track_flag = 1;
        }
      }
    }
}

void front()
```

```
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,220);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,190);
}
void front2()
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,75);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,70);
}
void back()
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,220);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,190);
}
void back2()
{
```

```
    digitalWrite(left_ctrl,LOW);
    analogWrite(left_pwm,110);
    digitalWrite(right_ctrl,LOW);
    analogWrite(right_pwm,90);
}
void left()
{
    digitalWrite(left_ctrl,LOW);
    analogWrite(left_pwm,220);
    digitalWrite(right_ctrl,HIGH);
    analogWrite(right_pwm,190);
}
void right()
{
    digitalWrite(left_ctrl,HIGH);
    analogWrite(left_pwm,220);
    digitalWrite(right_ctrl,LOW);
    analogWrite(right_pwm,190);
}
void Stop()
{
    analogWrite(left_pwm,0);
```

```
    analogWrite(right_pwm,0);
}
void Stop2()
{
    digitalWrite(left_ctrl,LOW);
    analogWrite(left_pwm,200);
    digitalWrite(right_ctrl,LOW);
    analogWrite(right_pwm,200);
    delay(50);
    analogWrite(left_pwm,0);
    analogWrite(right_pwm,0);
}

//this function is used for dot matrix display
void matrix_display(unsigned char matrix_value[])
{
    IIC_start();   // the function to transmit data
    IIC_send(0xc0);   //select address
       for(int i = 0;i < 16;i++) //pattern data has 16 bytes
    {
        IIC_send(matrix_value[i]); //data transmits patterns
    }
```

```
    IIC_end();     //end the transmission of patterns data
    IIC_start();
    IIC_send(0x8A);   //display the control, set pulse width to 4/16
    IIC_end();
}
// The condition of data transmission starts
void IIC_start()
{
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
}
// transmit data
void IIC_send(unsigned char send_data)
{
    for(char i = 0;i < 8;i++)   //Every character has 8 bits
    {
        digitalWrite(SCL_Pin,LOW);   //pull down the SCL_Pin to change the
signal of SDA
```

```
        delayMicroseconds(3);
        if(send_data & 0x01)    // 1 or 0 of byte is used to set high and low
level of SDA_Pin
        {
          digitalWrite(SDA_Pin,HIGH);
        }
        else
        {
          digitalWrite(SDA_Pin,LOW);
        }
        delayMicroseconds(3);
        digitalWrite(SCL_Pin,HIGH); //pull up the SCL_Pin to stop
transmitting data        delayMicroseconds(3);
        send_data = send_data >> 1;   //Detect bit by bit, so move the data
right by one bit detect bit by bit, move data
    }
}
//the sign that data ends transmitting
void IIC_end()
{
   digitalWrite(SCL_Pin,LOW);
   delayMicroseconds(3);
```

```
    digitalWrite(SDA_Pin,LOW);
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    digitalWrite(SDA_Pin,HIGH);
    delayMicroseconds(3);
}//*********************************************************
```

(4) Test Result

The 4WD robot car can go forward and back and turn left and right. After connecting to Bluetooth successfully, we can use the mobile APP to control the smart car to move.

9. Resources

Wiki page: https://wiki.keyestudio.com/Main_Page

Official website: https://keyestudio.com/

Assembly Video Link: http://video.keyestudio.com/ks0470/