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 cars and starter kits for Arduino, Raspberry Pi and BBC micro:bit, which can help customers at any level learn electronics and programming knowledge. Furthermore, all of our products comply with international quality standards and are greatly appreciated in a variety of different markets worldwide.

Welcome to check more contents from our official website:

http://www.keyestudio.com

*References and After-sales Service

1. Download Profile：https://fs.keyestudio.com/KS0558

2. If you find any parts missing or encounter any troubles, please feel free to contact us: **service@keyestudio.com.**

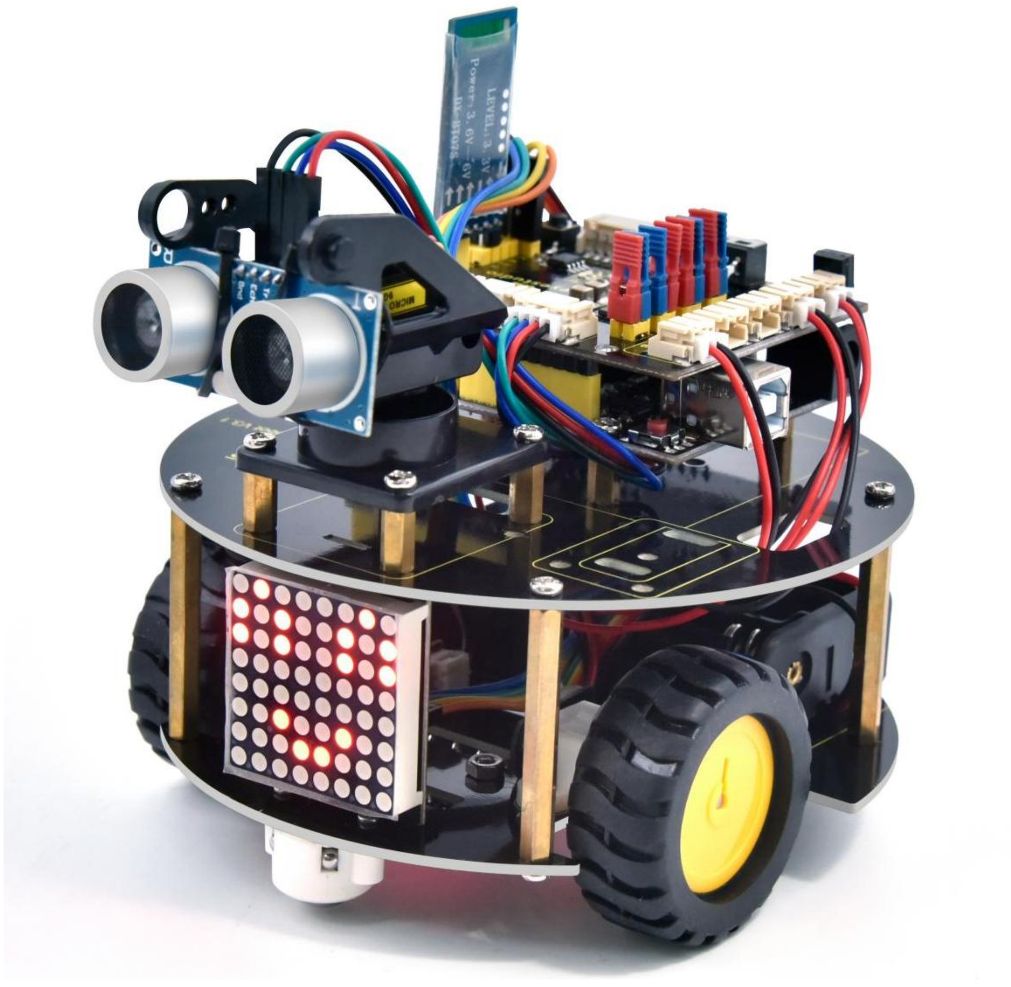
3. We will update projects and products continuously according to your sincere advice.

*Warning

1. This product contains tiny parts(screws, copper pillars). Therefore, keep it out of reach of children under 7 please.
2. This product consists of conductive parts (control board and electronic module). Please operate according to the requirements of tutorial. Otherwise, improper operation may cause parts to overheat and be damaged. Do not touch or immediately disconnect the circuit power.

*Copyright

Keyestudio Smart Little Turtle Robot V3.0



1. Introduction

Nowadays, technological education such as VR, kids programming, and artificial intelligence, has become a mainstream in educational industry. Thereby, people attach more importance to STEAM education. Arduino is notably famous for 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 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 updated a turtle robot V3.0.

It has a Keyestudio 4.0 board compatible with Arduino IDE and Scratch. The 4.0 board can read sensors and controls drivers by a driver expansion board.

This product boasts 16 learning projects, from simple to complex, which will guide you to make a smart turtle robot and introduce the detailed knowledge about sensors and modules.

2. Features

1. Multi-purpose function: Obstacle avoidance, following, IR remote control, Bluetooth control, confined with a circle, ultrasonic following and facial emoticons display.

2. Simple assembly: 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: 6-9V

Maximum output current: 2A

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

Motor speed: 5V 63 rpm

Motor drive mode: DRV8833 motor driver

Ultrasonic induction angle: <15 degrees

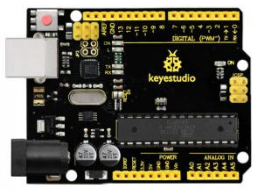
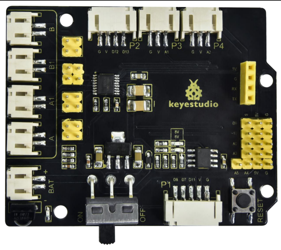
Ultrasonic detection distance: 2cm-400cm

Infrared remote control distance: 10M (measured)

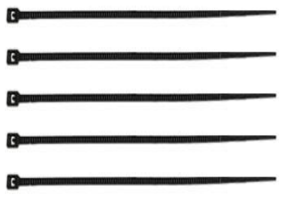
Bluetooth remote control distance: 50M(measured)

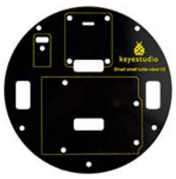
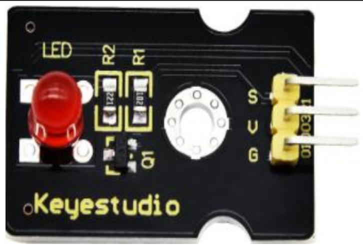
Bluetooth control: support Android and iOS system

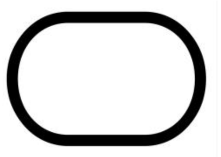
4. Product Kit

#	Product Name	QTY	Picture
1	Keyestudio V4.0 Board(UNO compatible)	1	
2	Keyestudio 8833 Driver Expansion Board	1	
3	Keyestudio Quick Connectors Line Tracking Sensor	1	
4	Keyestudio Quick Motor Connector A	1	
5	Keyestudio Quick Motor Connector B	1	
6	Keyestudio 8*8 Dot Matrix Module	1	
7	Keyestudio Ultrasonic Module	1	
8	DX-BT24 V5.1 BLE BT Module	1	

9	Keyestudio JMFP-4 17-Key Remote Control	1	
10	15CM JST-PH2.0MM-5P 24AWG Wire	1	
11	150MM PH2.0mm-4P to 2.54 Dupont Wire	1	
12	160mm 2P 24AW Wire	2	
13	Battery Holder	1	
14	4 AA Battery Holder	1	
15	M2*12MM Round Head Screws	4	
16	M2 Nuts	4	
17	M3*6MM Round Head Screws	27	
18	M3*10MM Flat Head Screws	2	
19	M3 Nuts	5	

20	M3*10MM Hexagon Copper Bush	8	
21	M3*40MM Hexagon Copper Bush	4	
22	Keyestudio 9G Servo	1	
23	N20 Motor Wheel	2	
24	N20 Motor U Type Mount	2	
25	Black Plastic Platform	1	
26	Arduino 3PI Universal Caster	2	
27	3*40MM Black-yellow Screwdriver	1	
28	1m Transparent Blue USB Cable	1	
29	3*100MM Black Ties	5	

30	Keyestudio Bottom Board	1	
31	Keyestudio Top Board	1	
32	F-F 20CM/40P Dupont Cable	4	
33	20cm 3pin F-F Dupont Cable	1	
34	Keyestudio Red LED Module	1	
35	Blue Jumper Caps	4	
36	Red Jumper Caps	4	
37	Decorative Board	1	
38	Winding Pipe	1	

39	Tracking Runway	1	

5. Installation

Step 1:Bottom motor wheel

- **Prepare the parts as follows:**

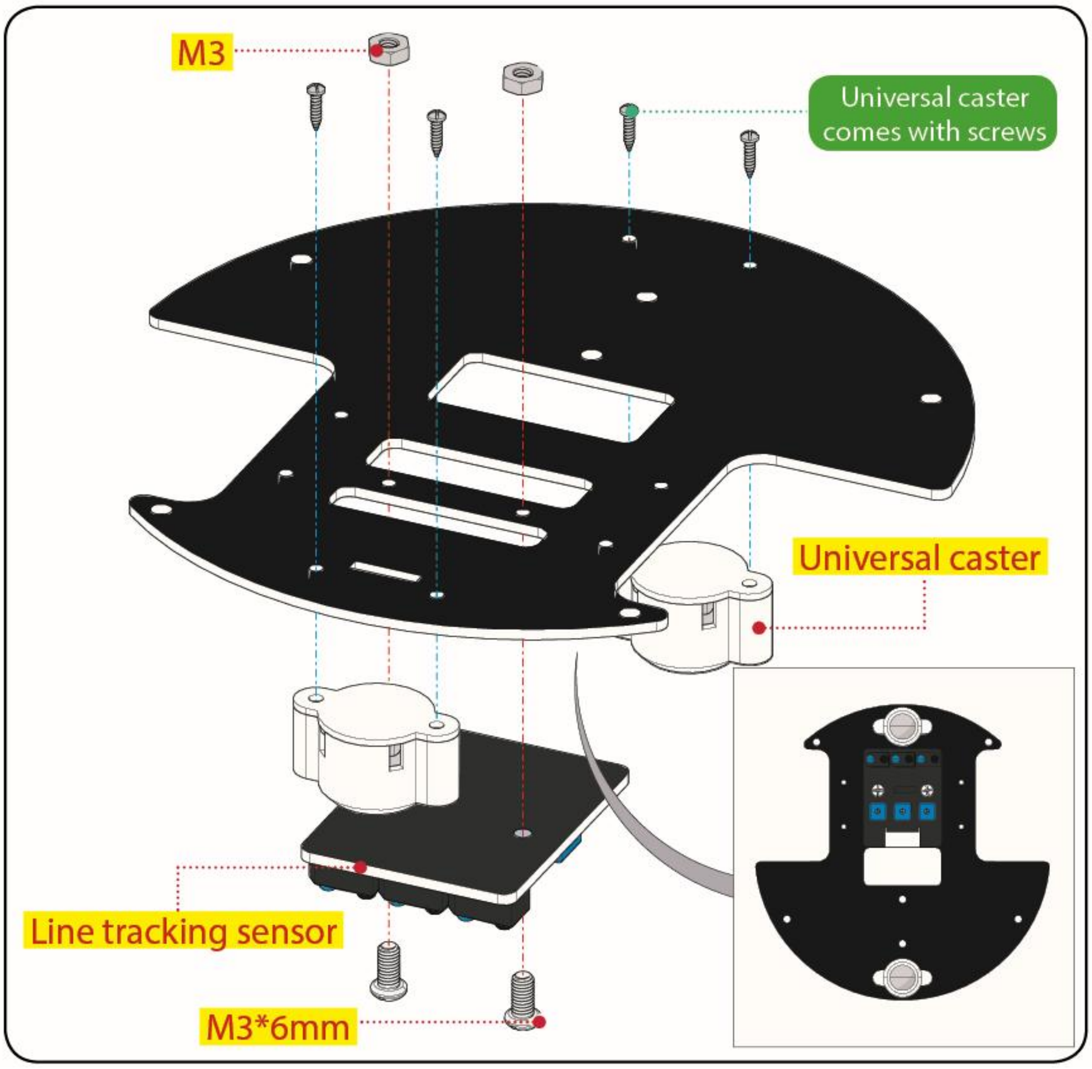
M3*6MM round-head screw *2

Nut M3 nickle plating *2

Bottom PCB*1

Tracking sensor *1

Universal caster *2



Step 2:Assemble Parts

- **Prepare the parts as follows:**

M2 Nut *4

12FN20 motor *2

U-type holder* 2

N20 motor wheel *2

2P wire *2

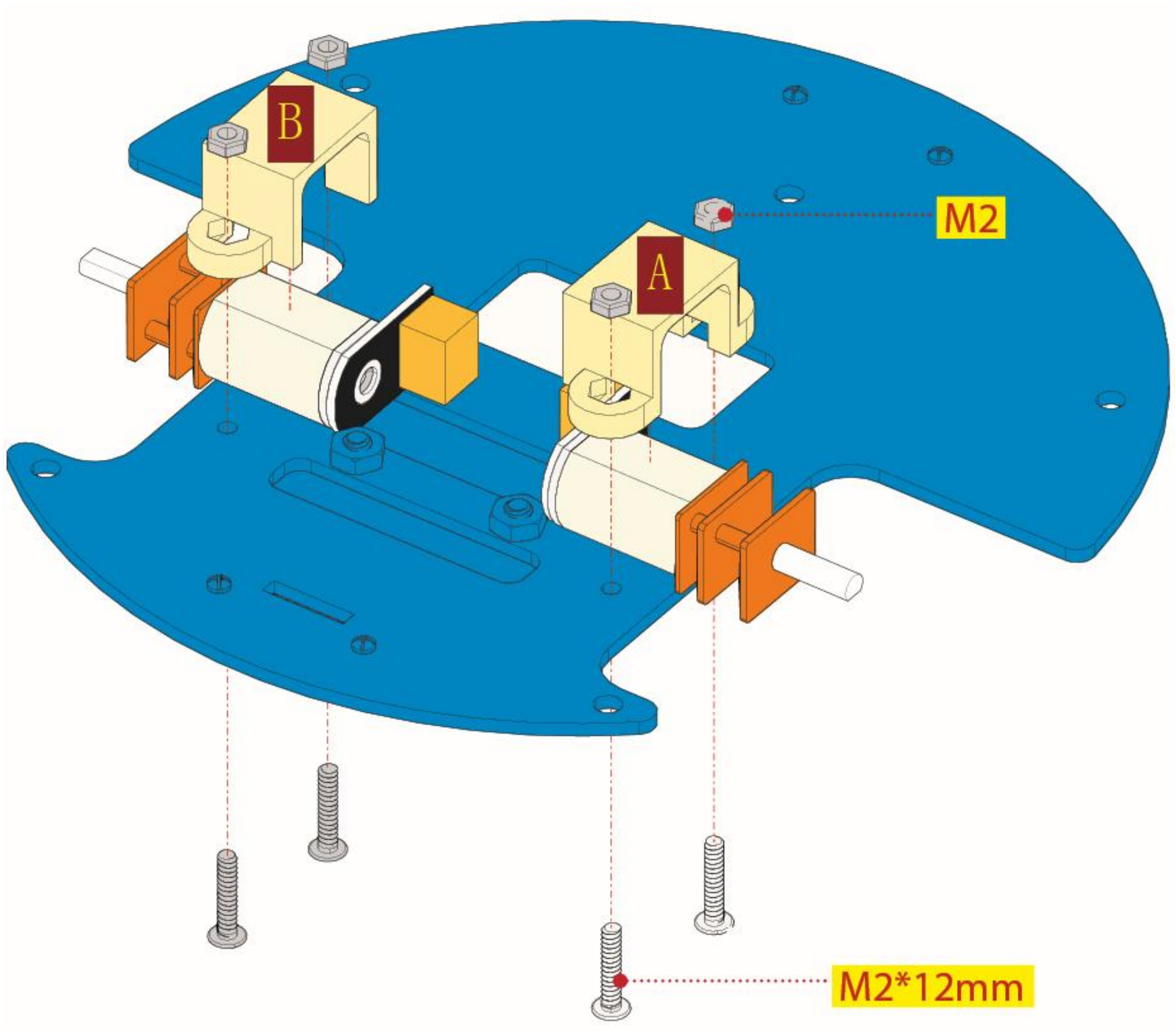
5P wire *1

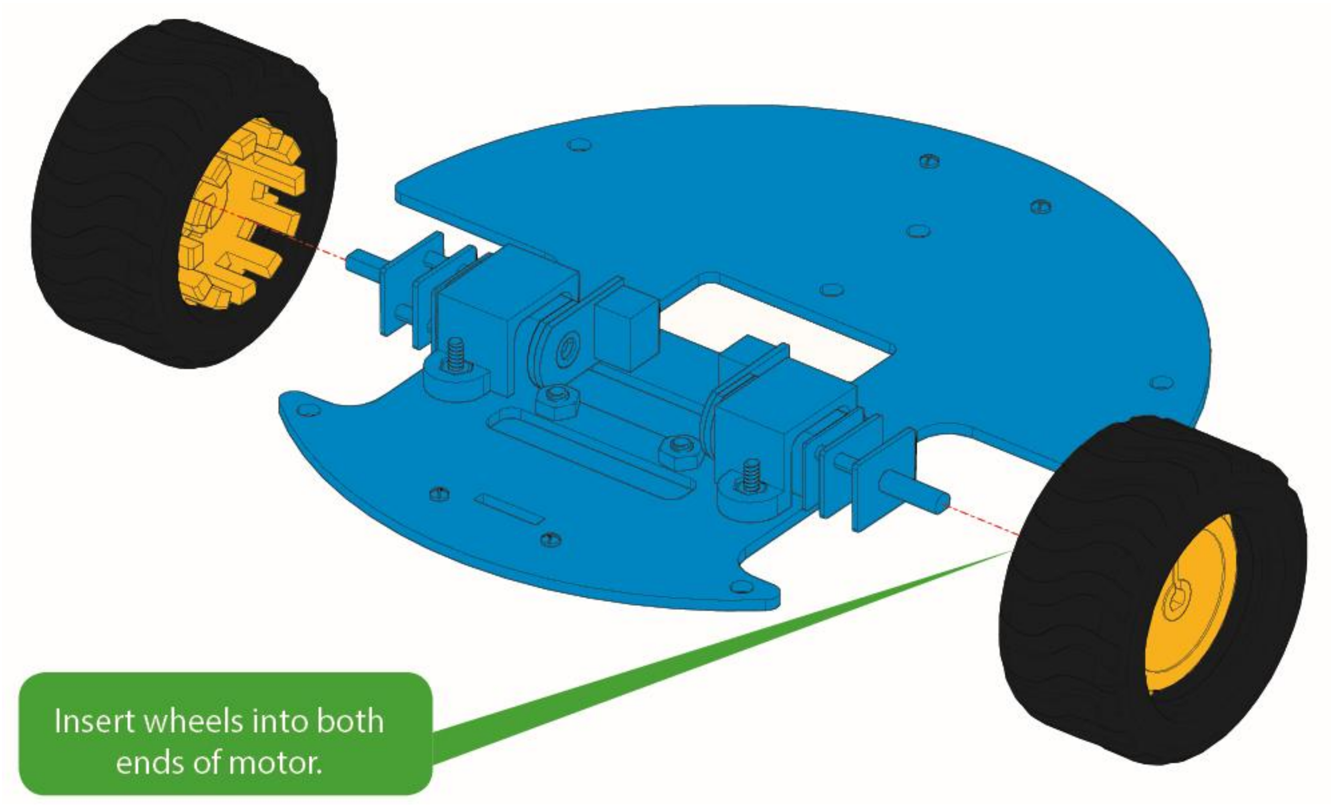
M2*12MM round-head screw *4

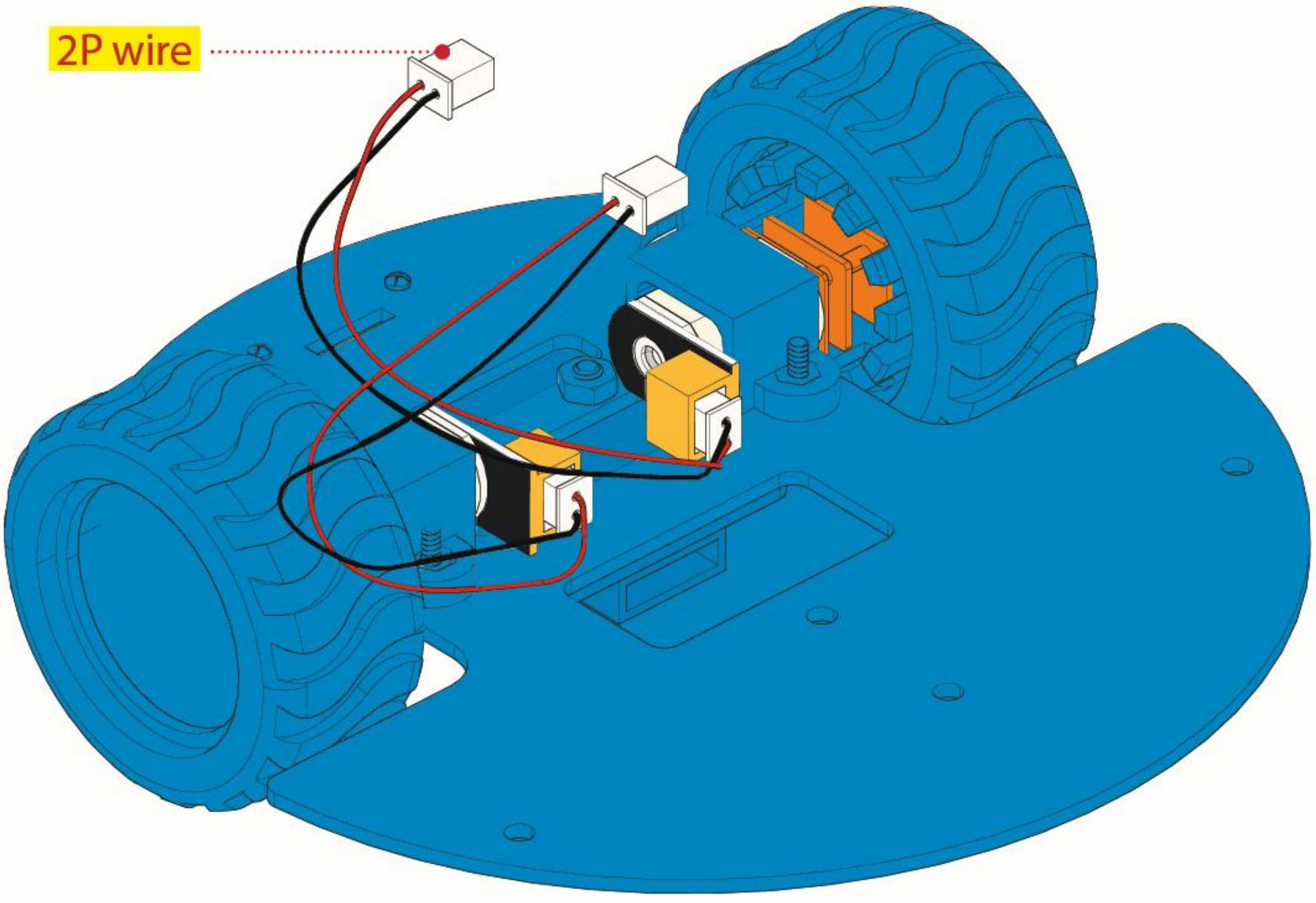
18650 battery holder *1

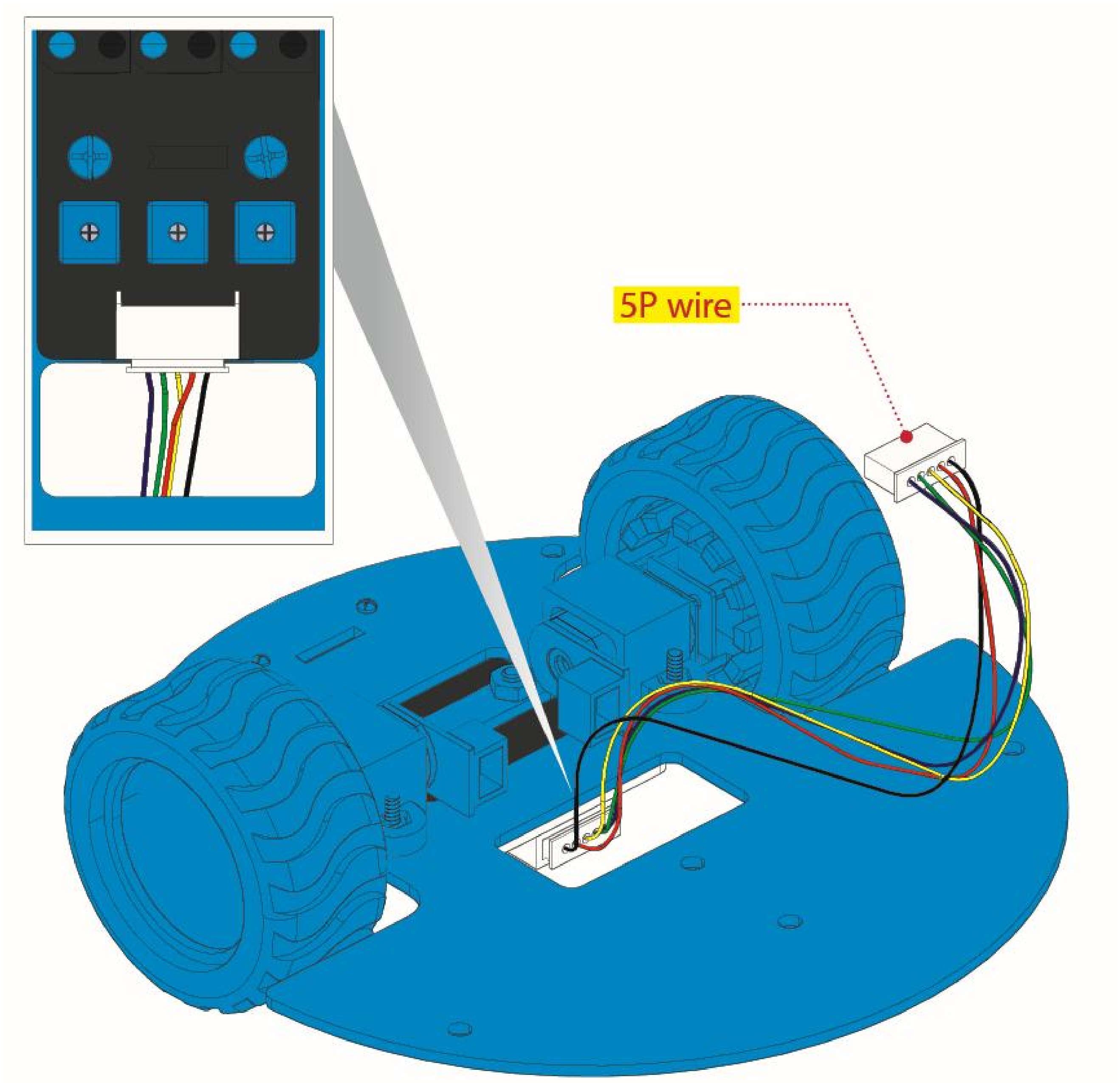
M3*10MM flat head screw *2

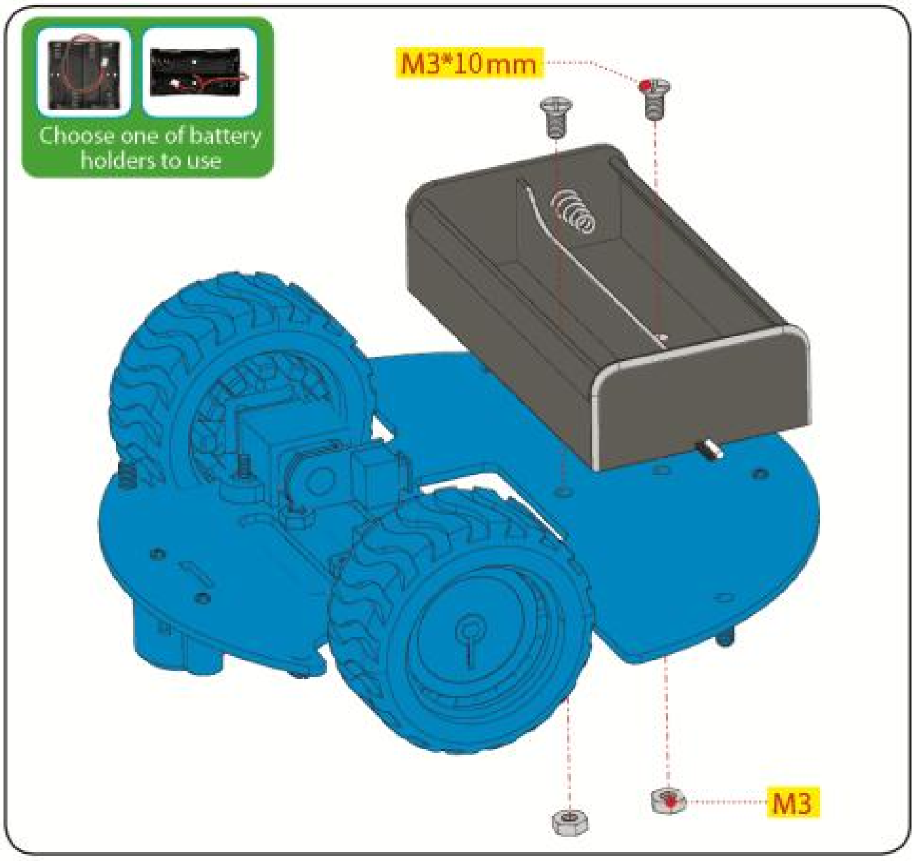
M3 nut *2


B
A
M2
M2*12mm

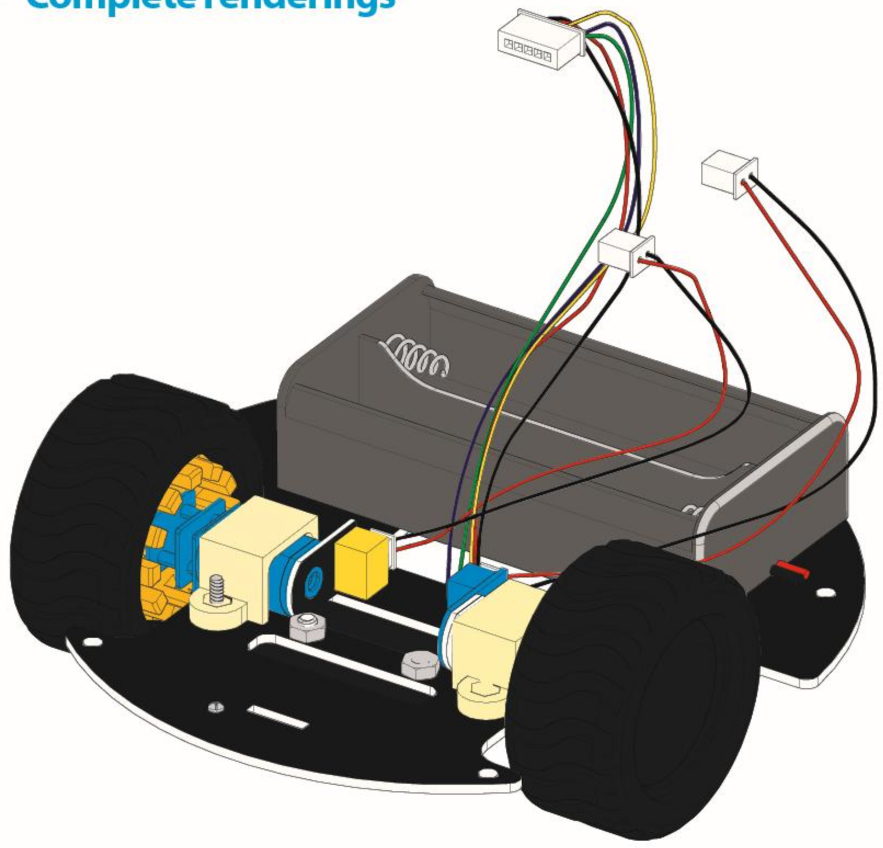

Insert wheels into both
ends of motor.


2P wire


5P wire



Complete renderings



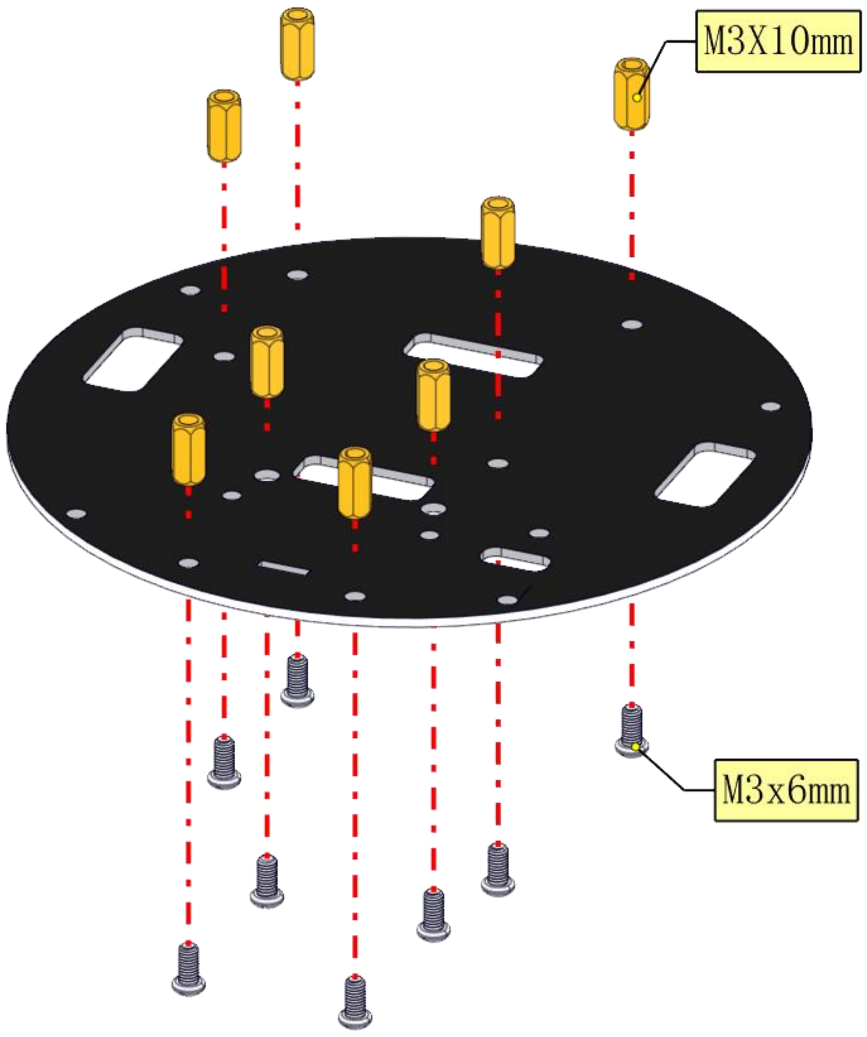
Step 3:Install Top PCB

- **Prepare the parts as follows:**

Top PCB *1

M3*6MM round-head screw *8

M3*10MM dual-pass copper pillar *8



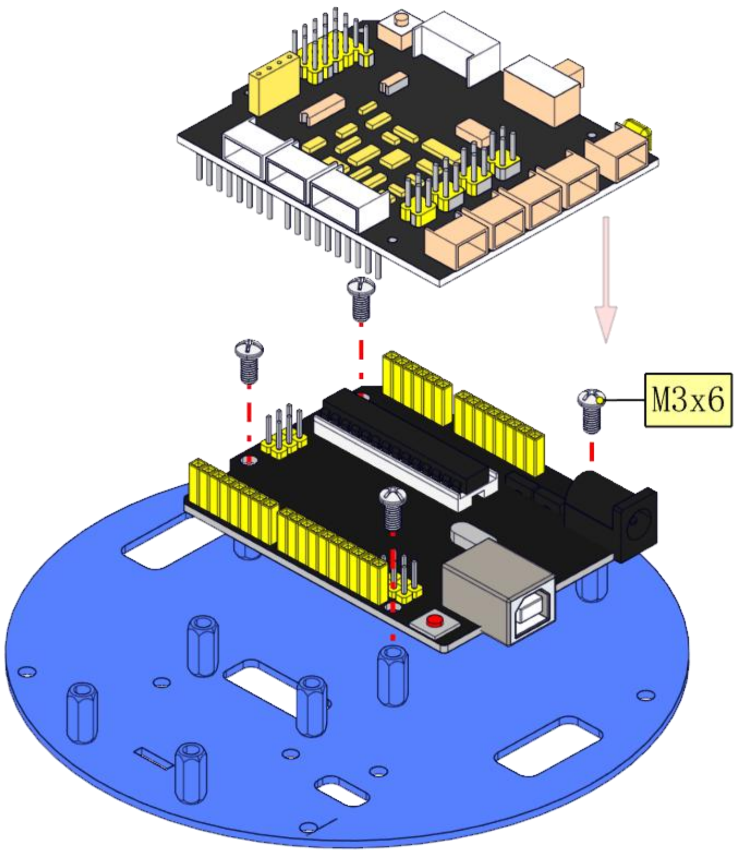
Step 4:Mount Control Board

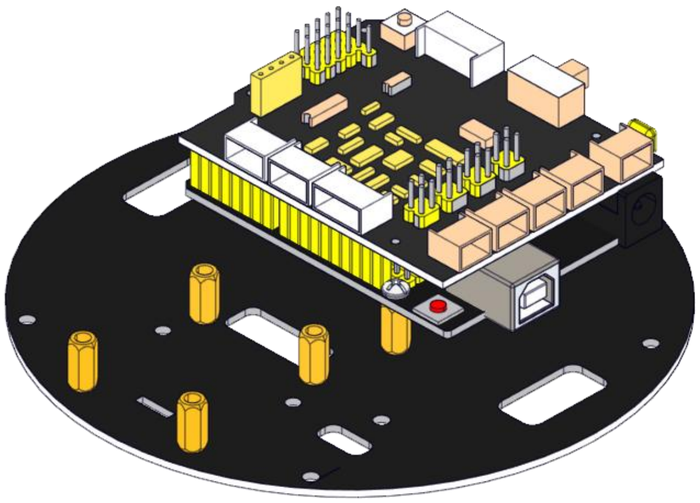
- Prepare the parts as follows:

V4.0 board*1

8833 Motor drive shield*1

M3*6MM round-head screw *4





Step 5:Servo plastic platform

- **Prepare the parts as follows:**

Servo *1

M2*4 screw *1

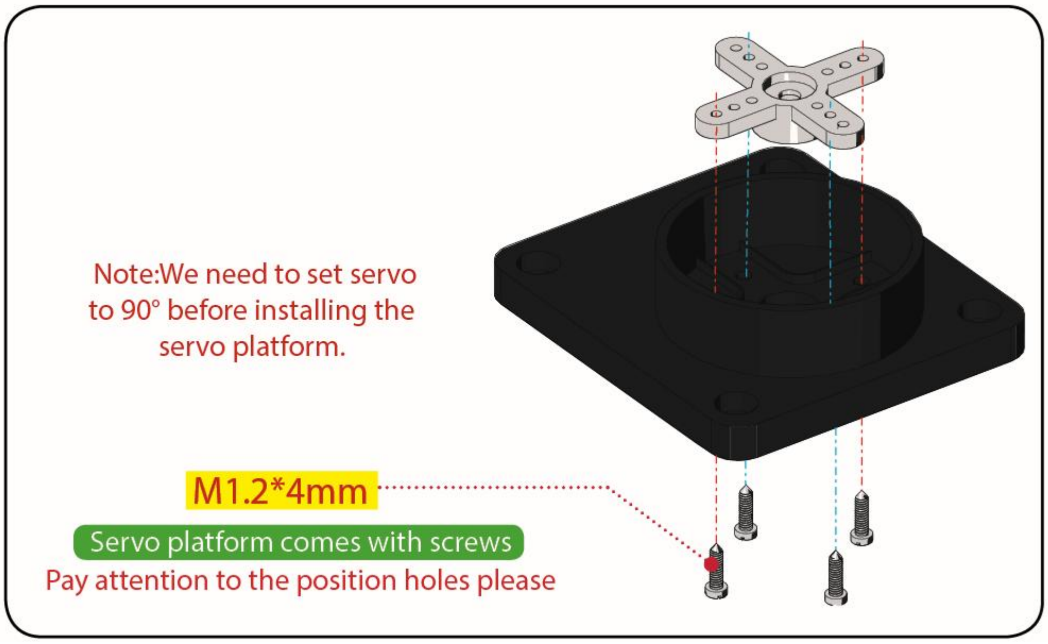
Black cable tie*2

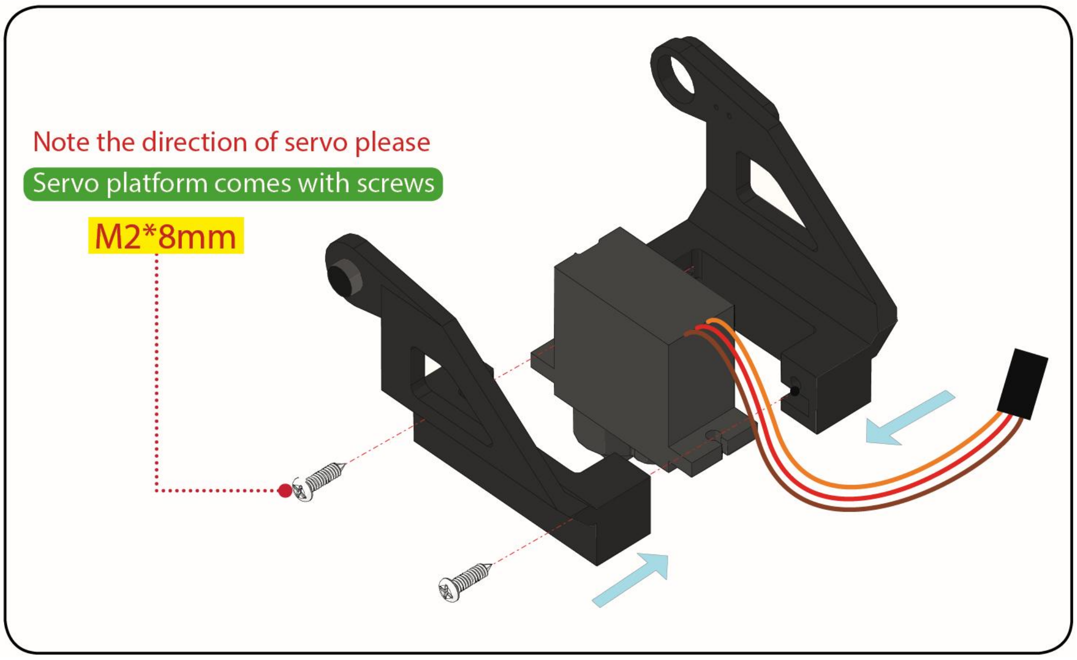
Ultrasonic sensor*1

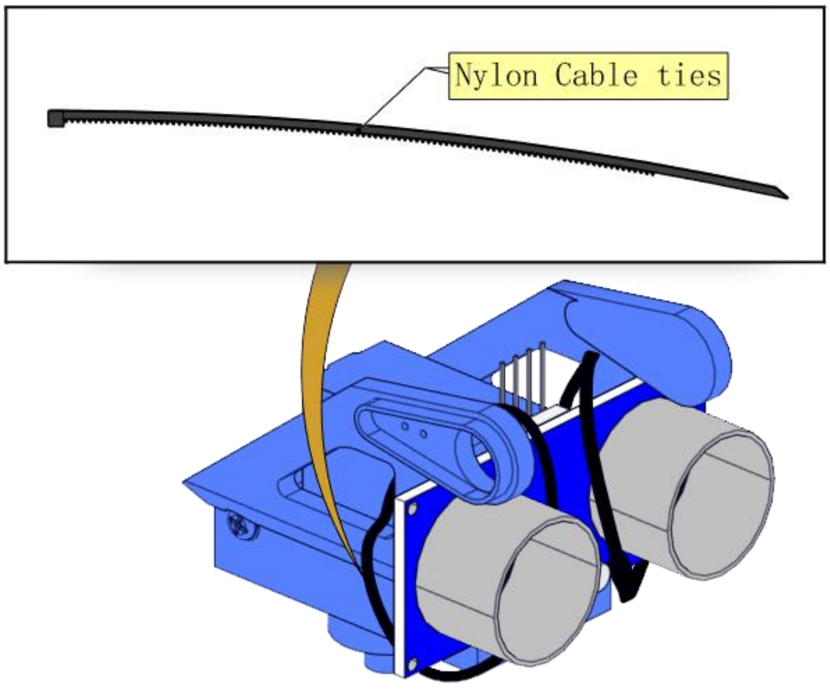
Black plastic platform *1

M1.2*4 tapping screw *4

M2*8 tapping screw *2

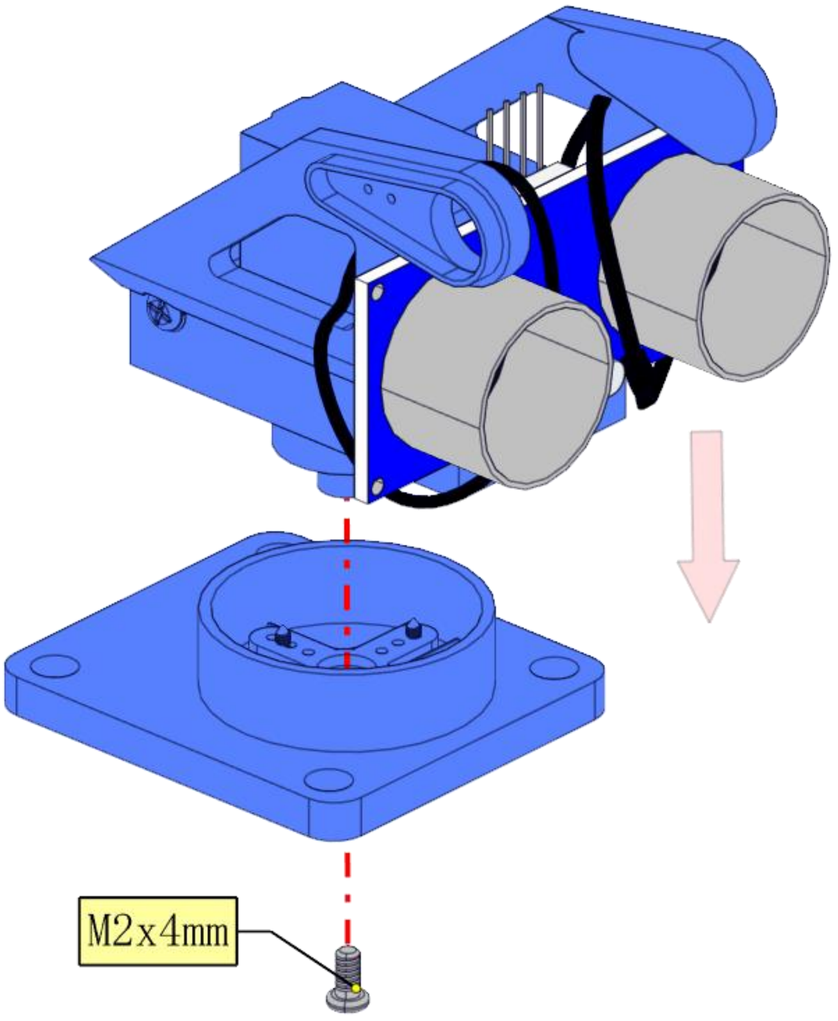

Note:We need to set servo to 90° before installing the servo platform.
M1.2*4mm
Servo platform comes with screws
Pay attention to the position holes please

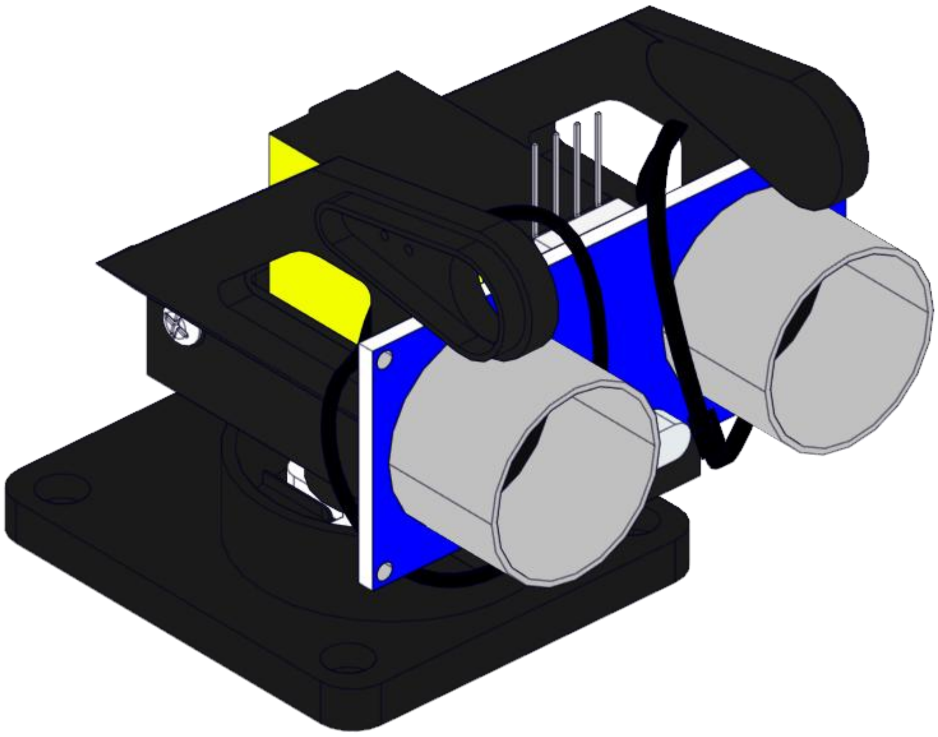

Note the direction of servo please
Servo platform comes with screws
M2*8mm


Nylon Cable ties

Initialize the steering gear, set the initial Angle of the steering gear to 90°, and the initialization code is as follows:

```
//*****************************************************************
/*
Set the 90-degree code,Copy the code and upload it to the development board. The steering gear connected to port D10 will rotate to 90 °
*/
#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 servo pin to OUTPUT
   procedure(0); //set the angle of servo to 0°
}
void loop() {

      procedure(90);                    // tell servo to go to position in variable 90°

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


M2x4mm



Step 6:Final Assembly

- **Prepare the parts as follows:**

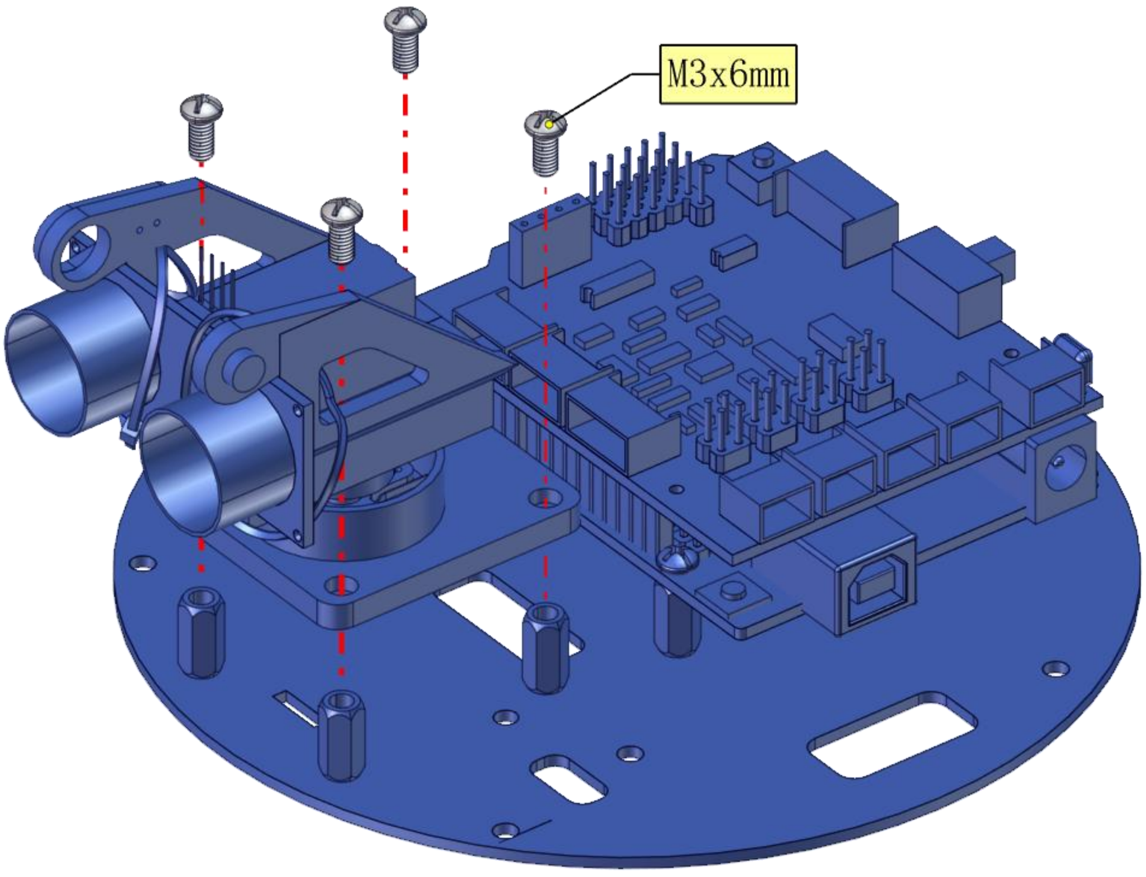
M3*6MM round-head screw *12

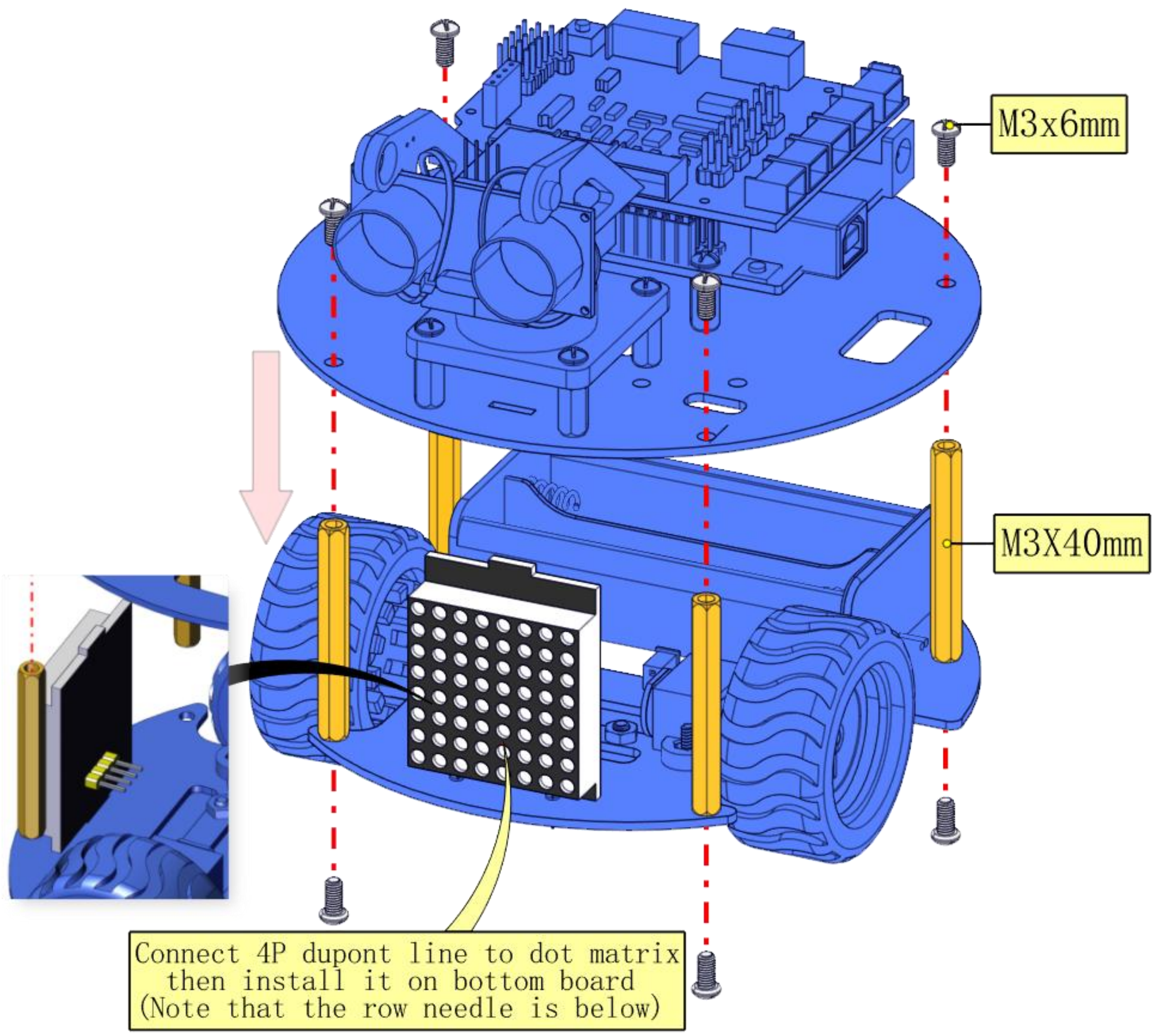
M3*40MM copper pillar *4

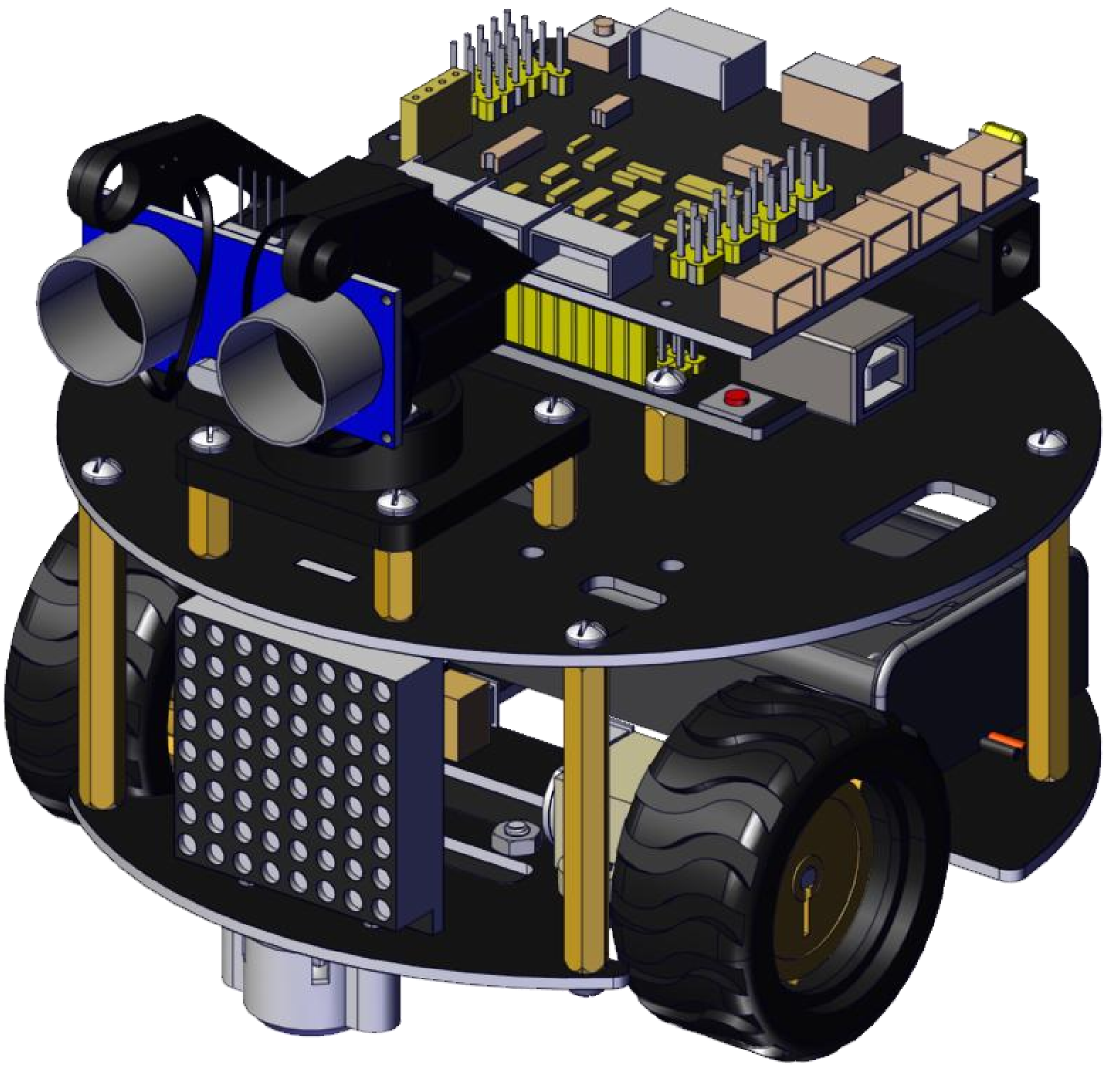
Bluetooth module *1

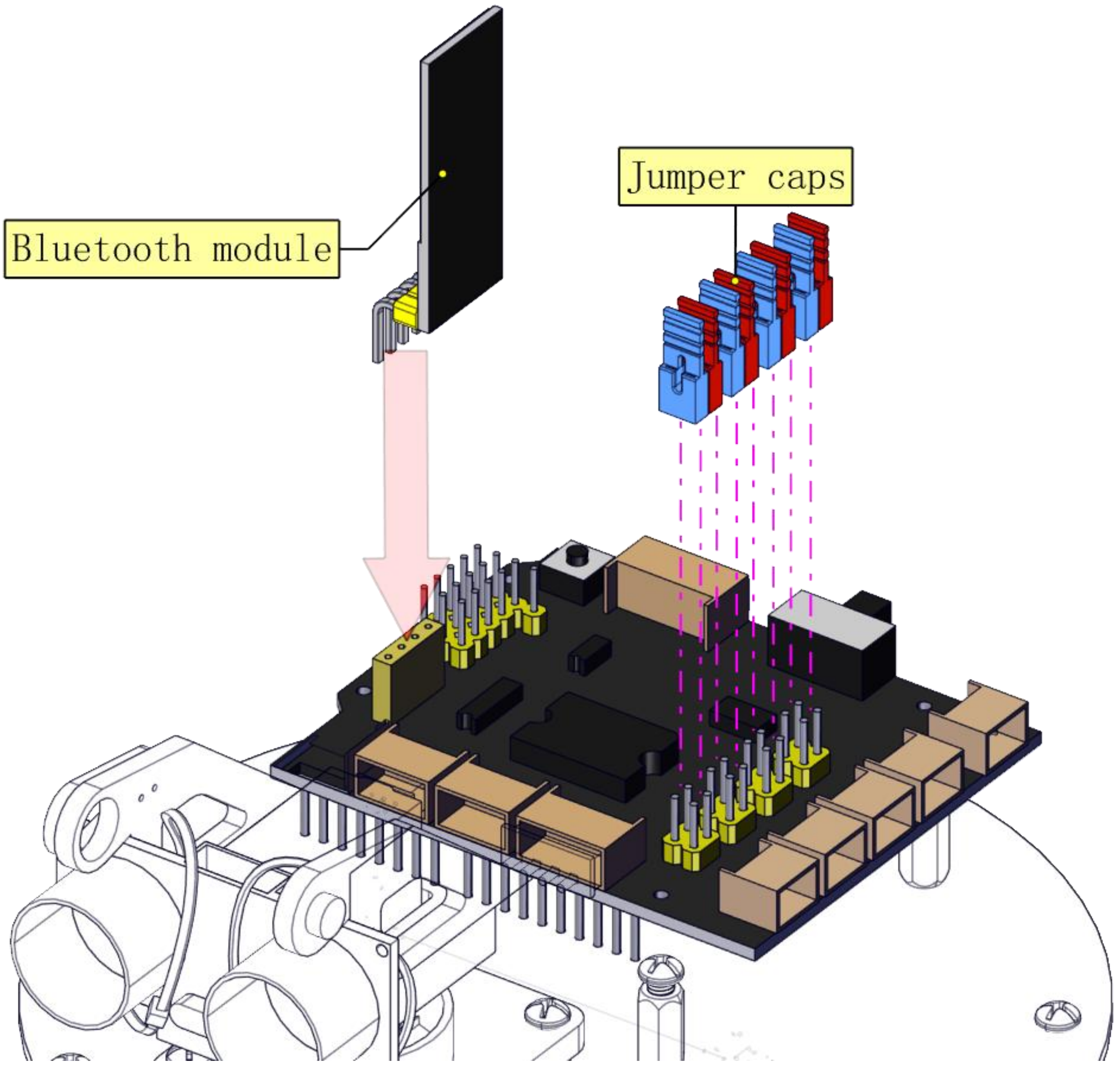
8x8 Dot Matrix *1

Jumper Cap *8

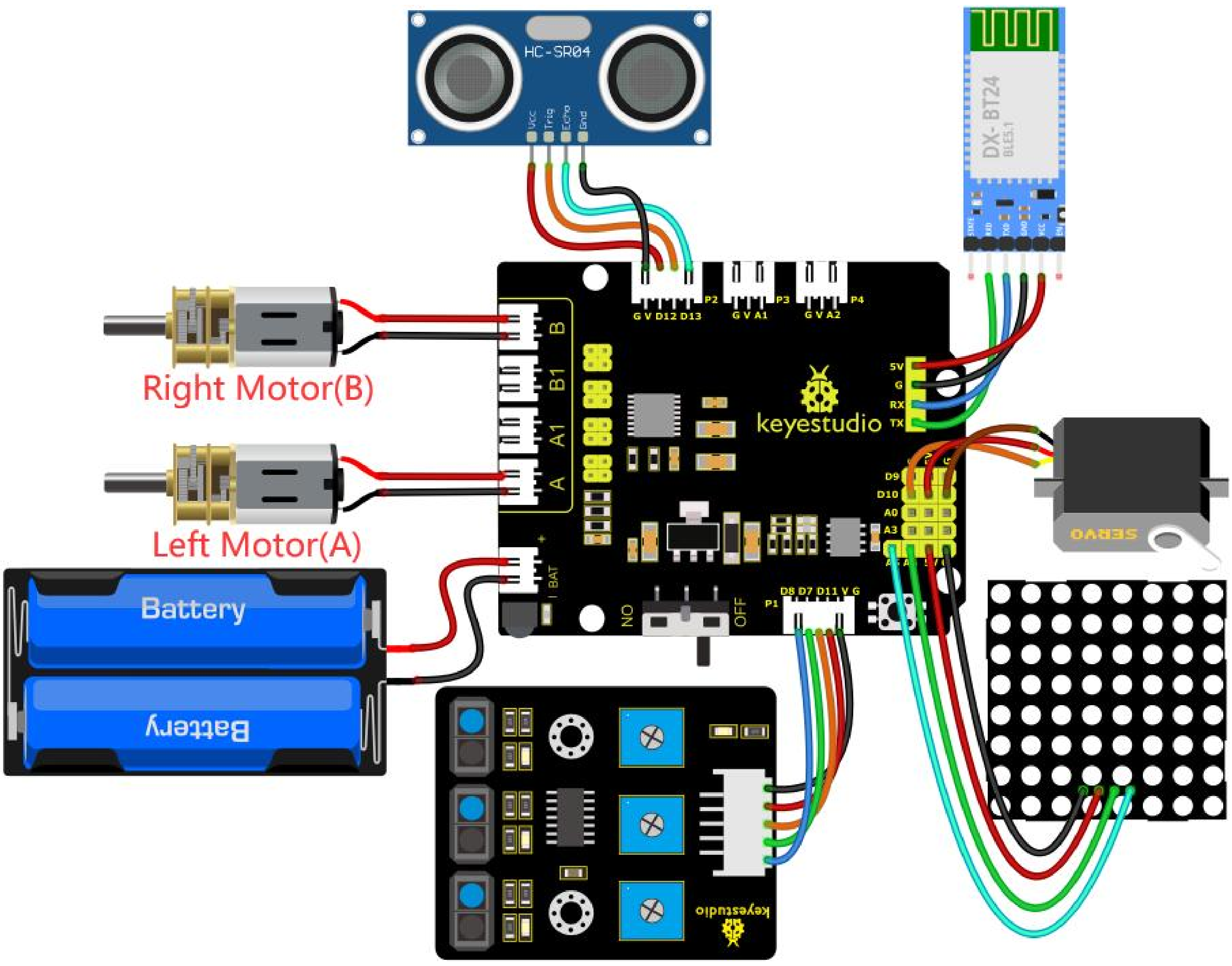


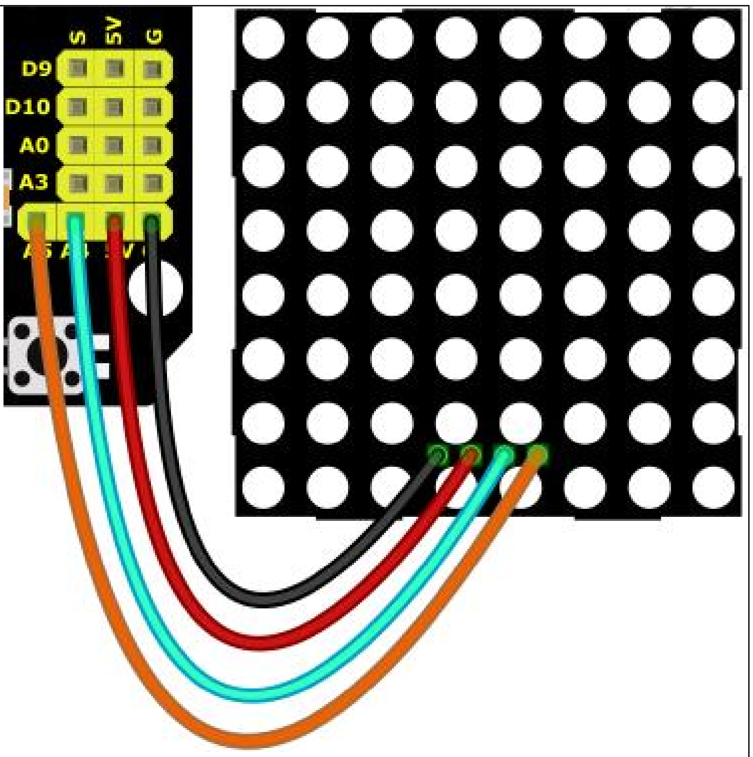

M3x6mm
M3X40mm
Connect 4P dupont line to dot matrix then install it on bottom board (Note that the row needle is below)

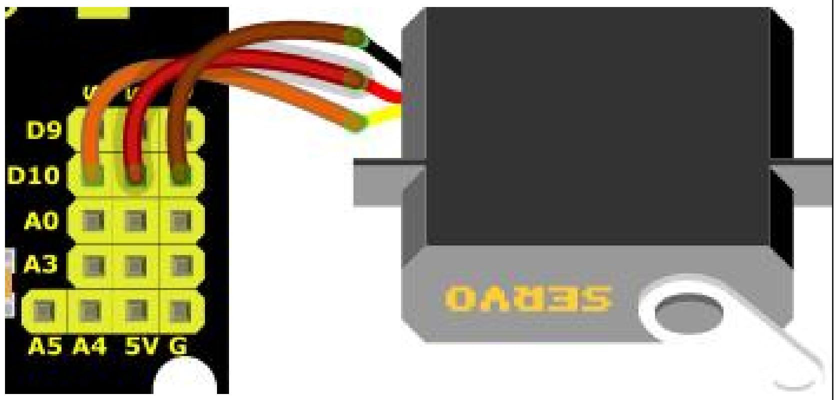


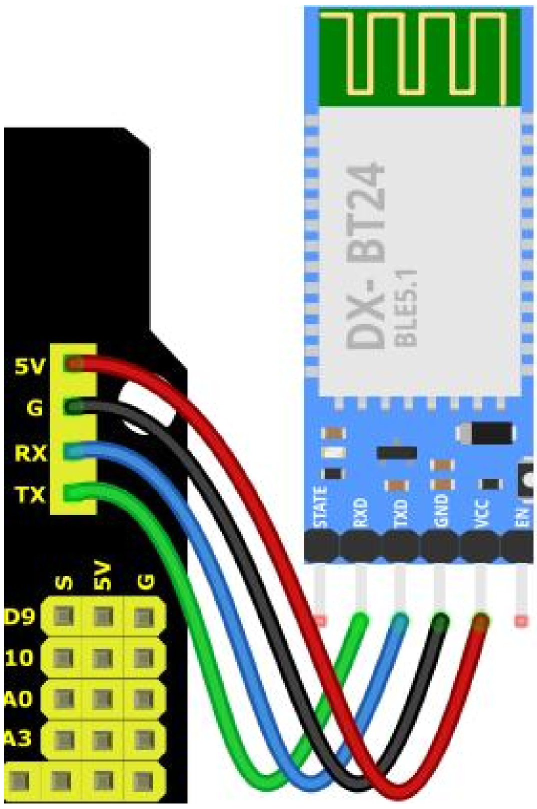

Bluetooth module
Jumper caps

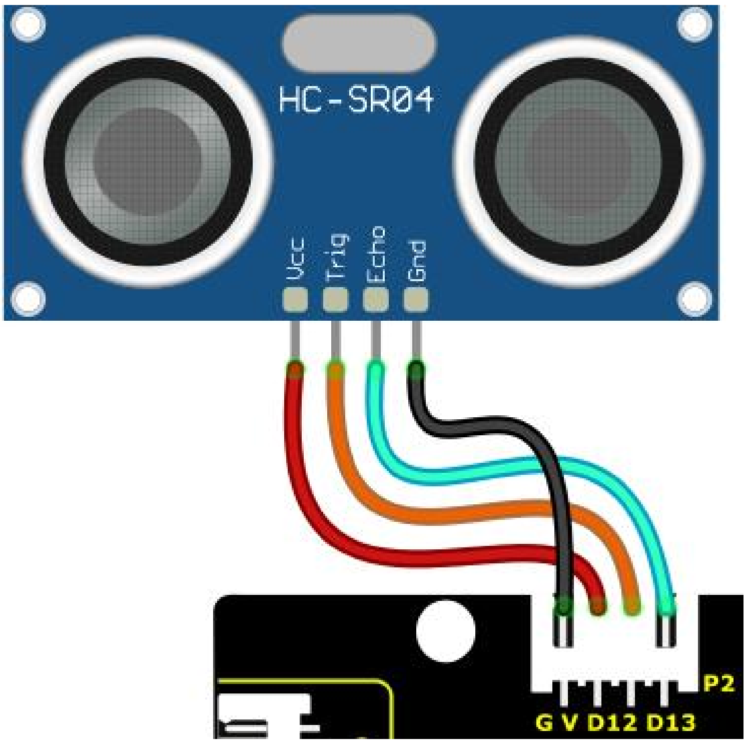
Step 7: Hook-up Guide

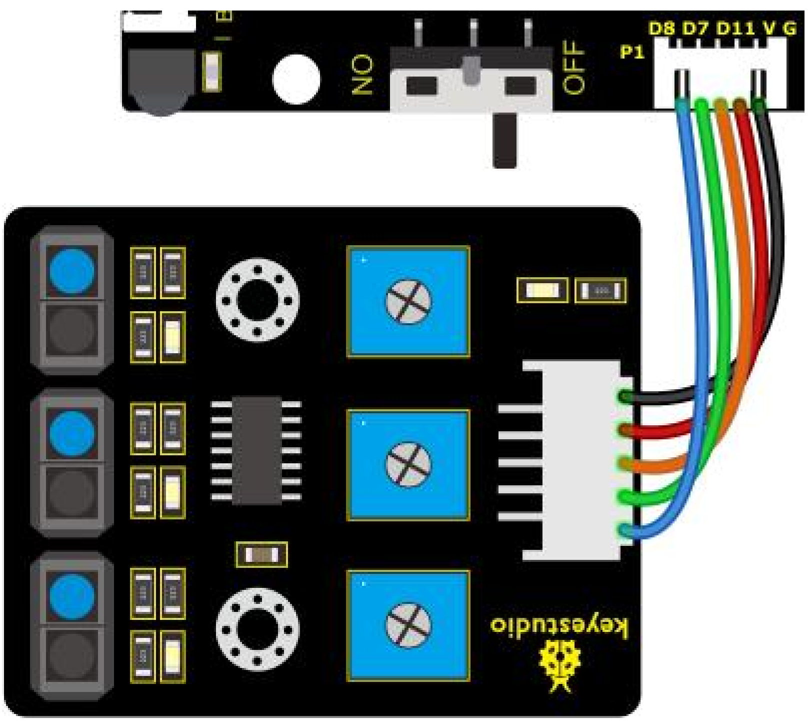


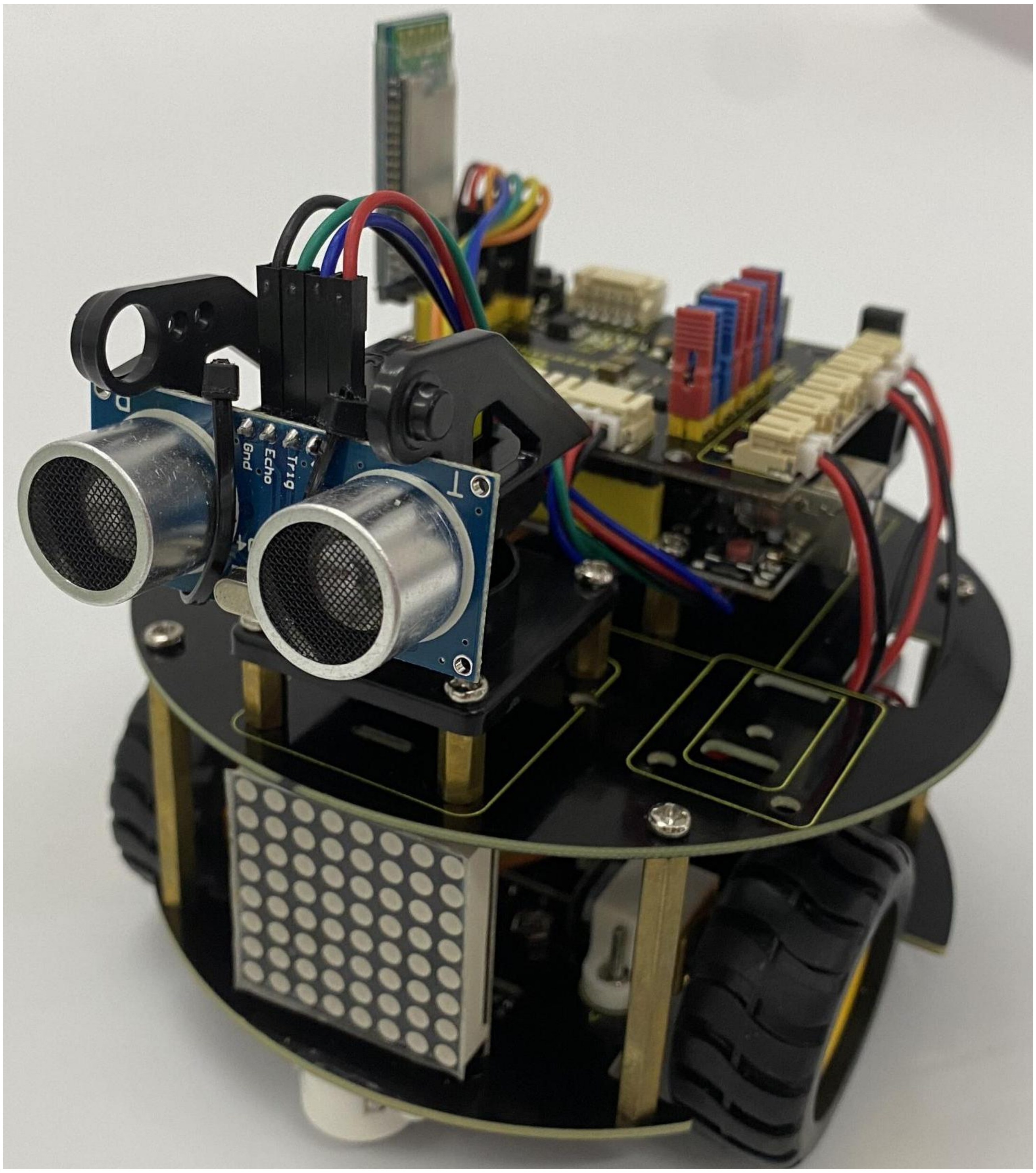
8*8 Dot Matrix	8833 Motor Driver Expansion Board	
G	G	
5V	5V	
SDA	A4	
SCL	A5	

Servo	8833 Motor Driver Expansion Board	
Brown	G	
Red	5V	
Orange	D10	

Bluetooth	8833 Motor Driver Expansion Board	
RXD	TX	
TXD	RX	
GND	G	
VCC	5V	

Ultrasonic Sensor	8833 Motor Driver Expansion Board	
Vcc	V	
Trig	D12	
Echo	D13	
Gnd	G	

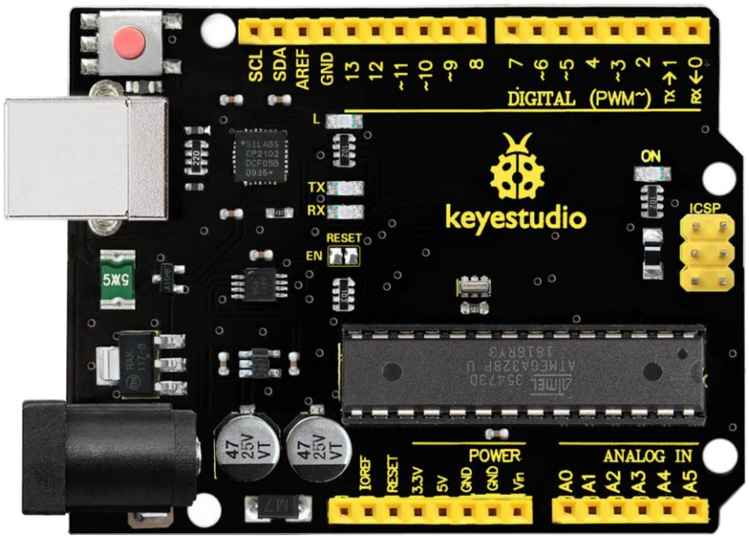
Line Tracking Sensor	8833 Motor Driver Expansion Board	
G	G	
V	V	
S1	D11	
S2	D7	
S3	D8	


Gnd
Echo
Trig
T

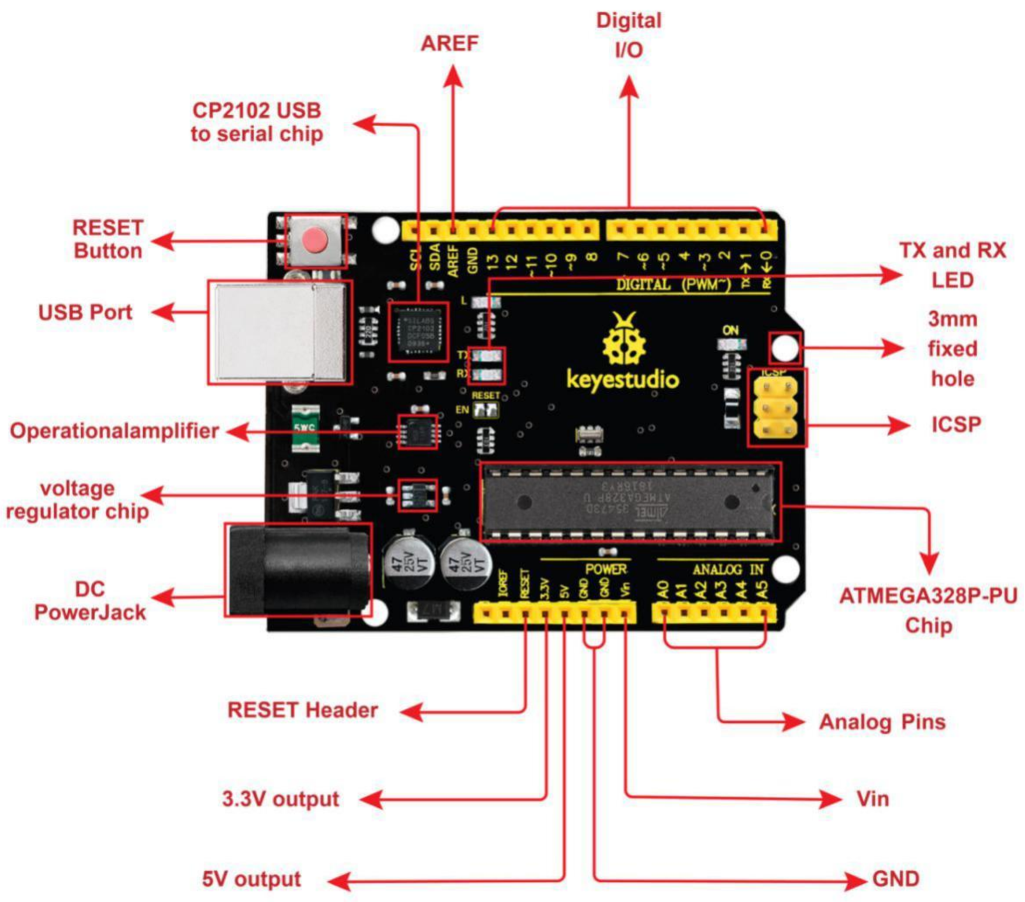
6. Getting started with Arduino

(1) Keyestudio V4.0 Development Board

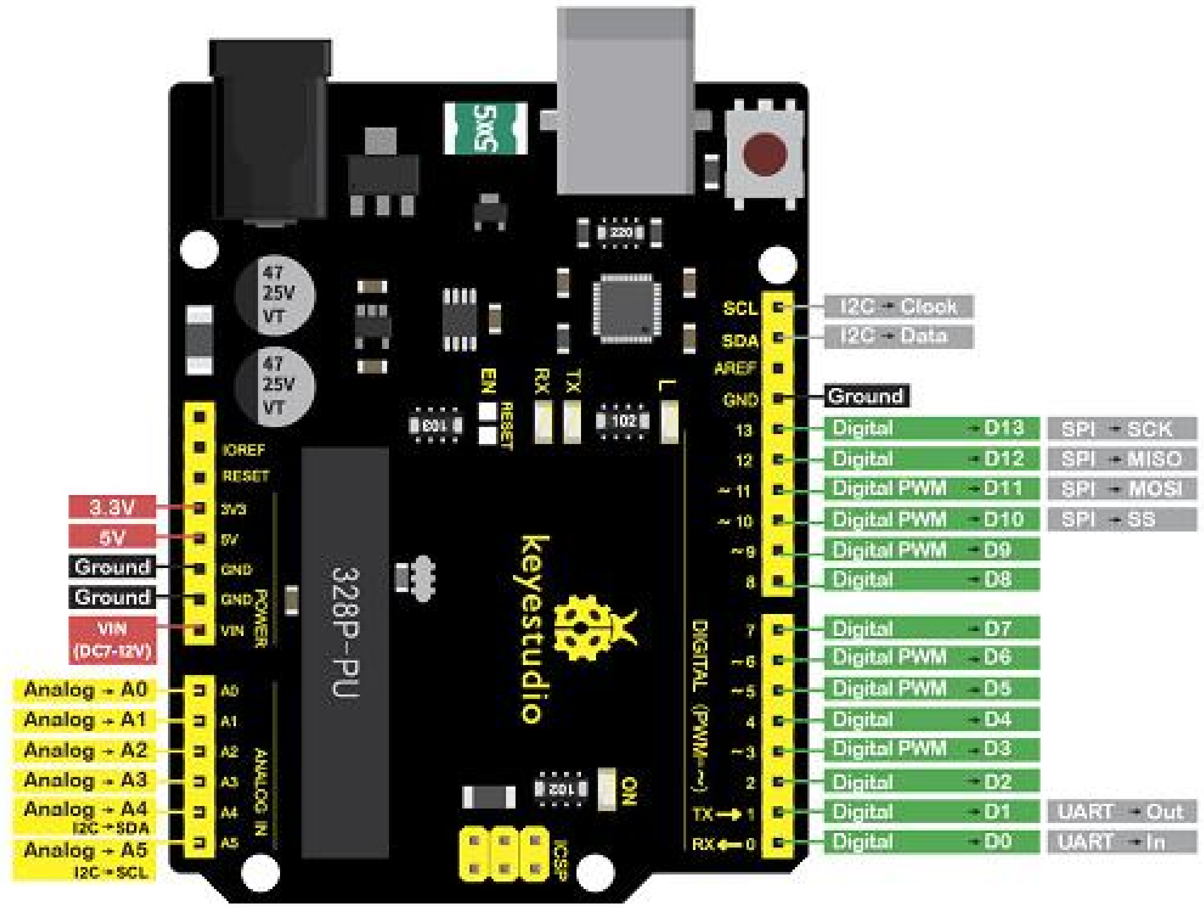
You need to know that keyestudio V4.0 development board is the core of this smart turtle robot .



Keyestudio V4.0 development board is an Arduino uno -compatible boar d, which is based on ATmega328P MCU, and with a cp2102 Chip as a UA RT-to-USB converter.



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



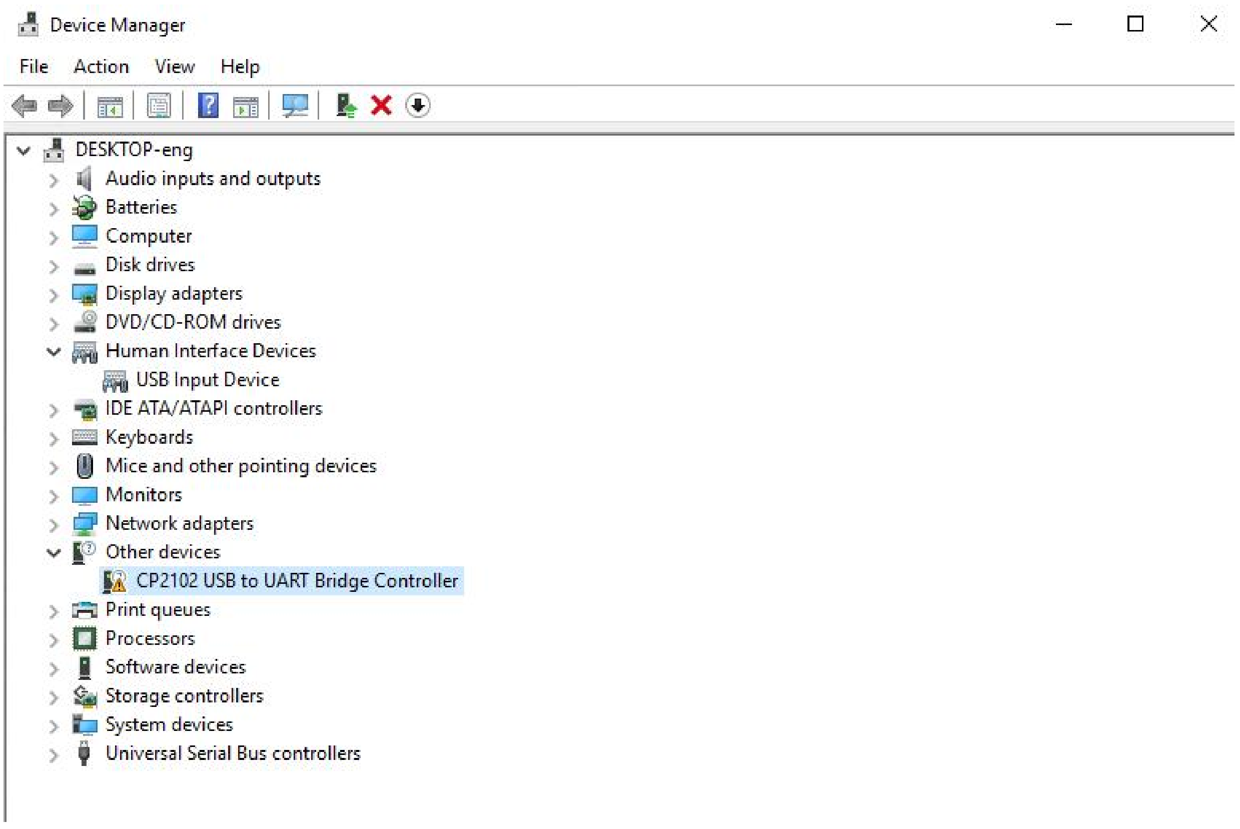
It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it via an external DC power jack (DC 7-12V) or via female headers Vin/ GND(DC 7-12V) to get started.

Microcontroller	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

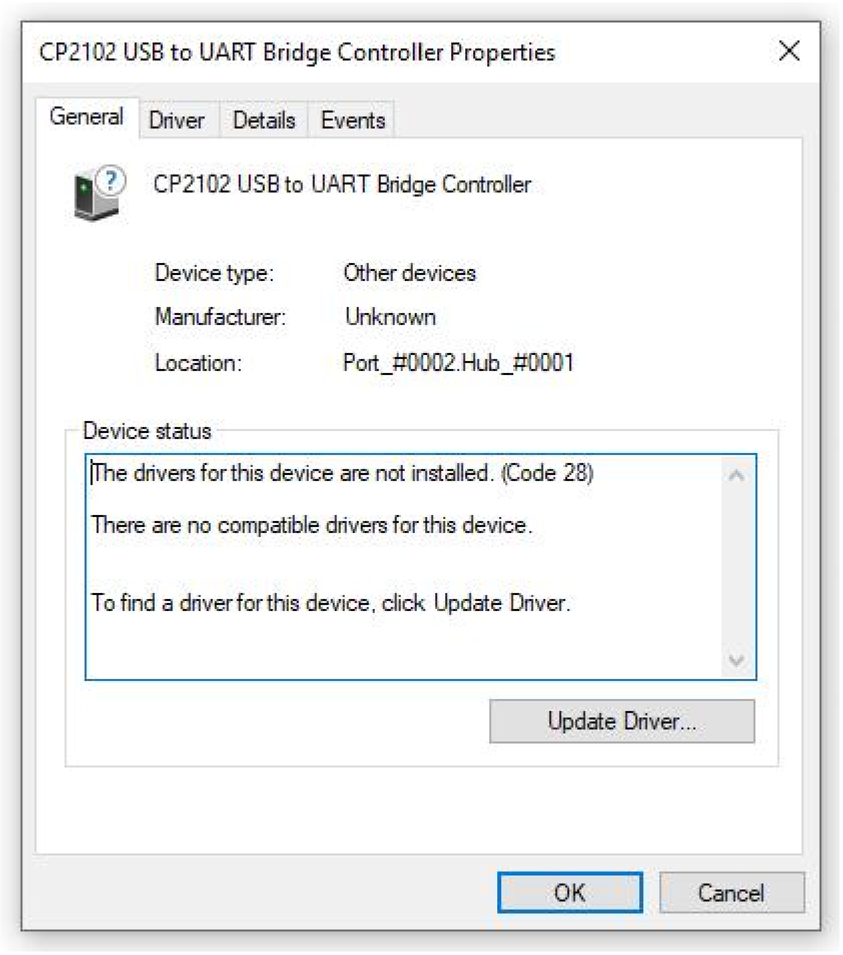
(2) Installing V4.0 board Driver

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



The yellow exclamation mark on the page implies an unsuccessful installation and you should double click the hardware and update the driver.



CP2102 USB to UART Bridge Controller Properties

CP2102 USB to UART Bridge Controller

Device type: Other devices
Manufacturer: Unknown
Location: Port_#0002.Hub_#0001

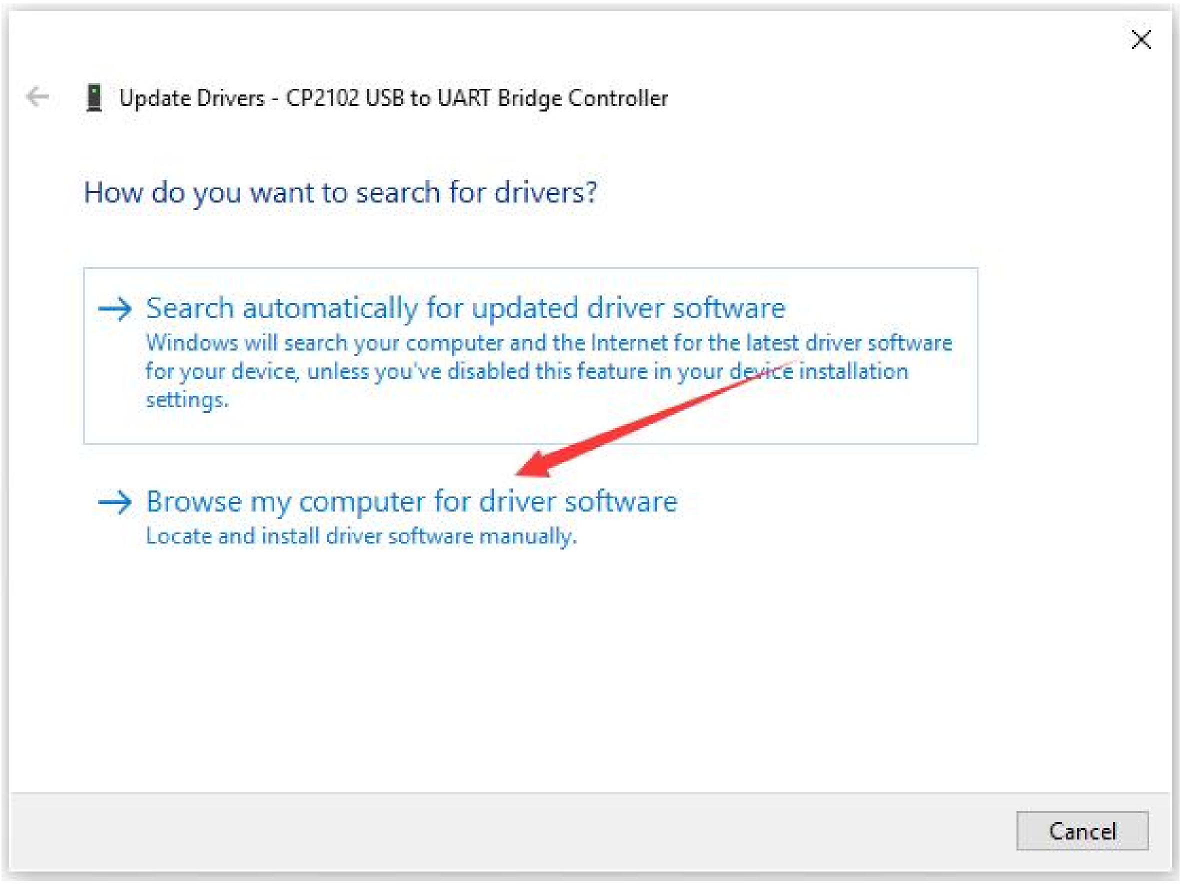
Device status

The drivers for this device are not installed. (Code 28)

There are no compatible drivers for this device.

To find a driver for this device, click Update Driver.

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


Update Drivers - CP2102 USB to UART Bridge Controller

How do you want to search for drivers?

→ Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

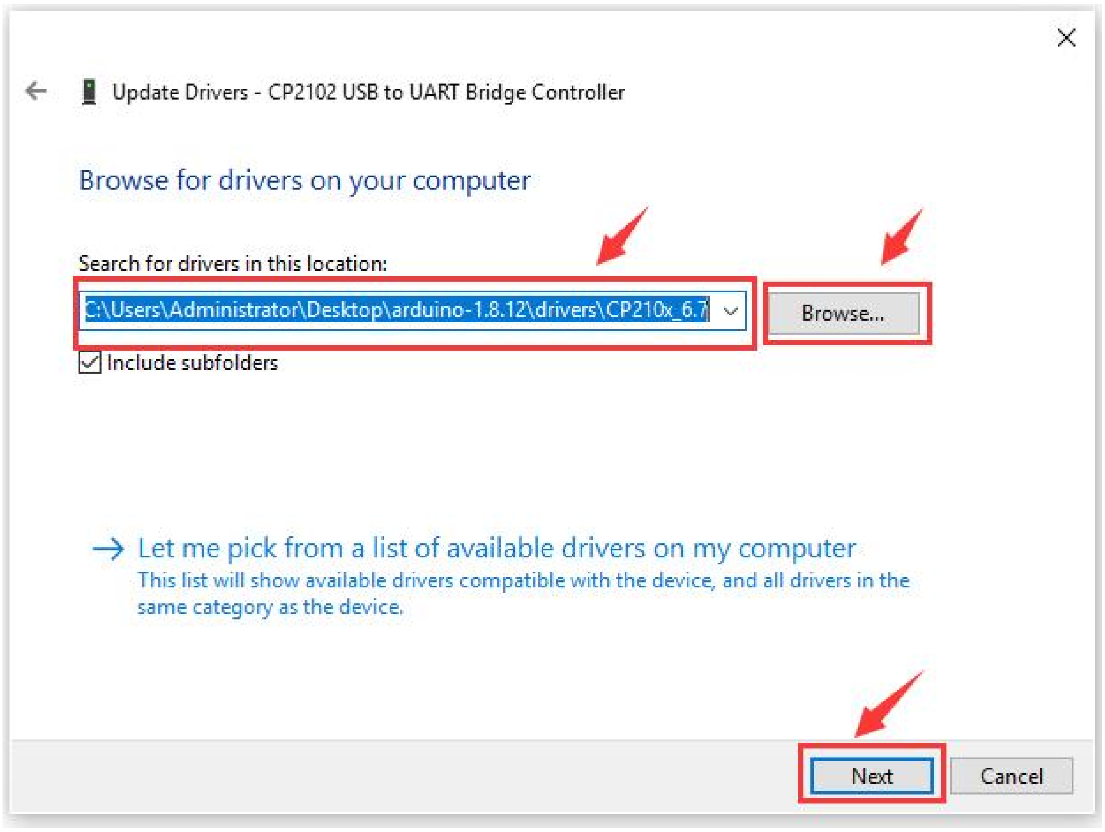
→ Browse my computer for driver software
Locate and install driver software manually.

There is a DRIVERS folder in Arduino software installed package

（ ）, open driver folder and you can see the driver of

CP210X series chips.

Click "Browse" , then find 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 suit.)


Update Drivers - CP2102 USB to UART Bridge Controller

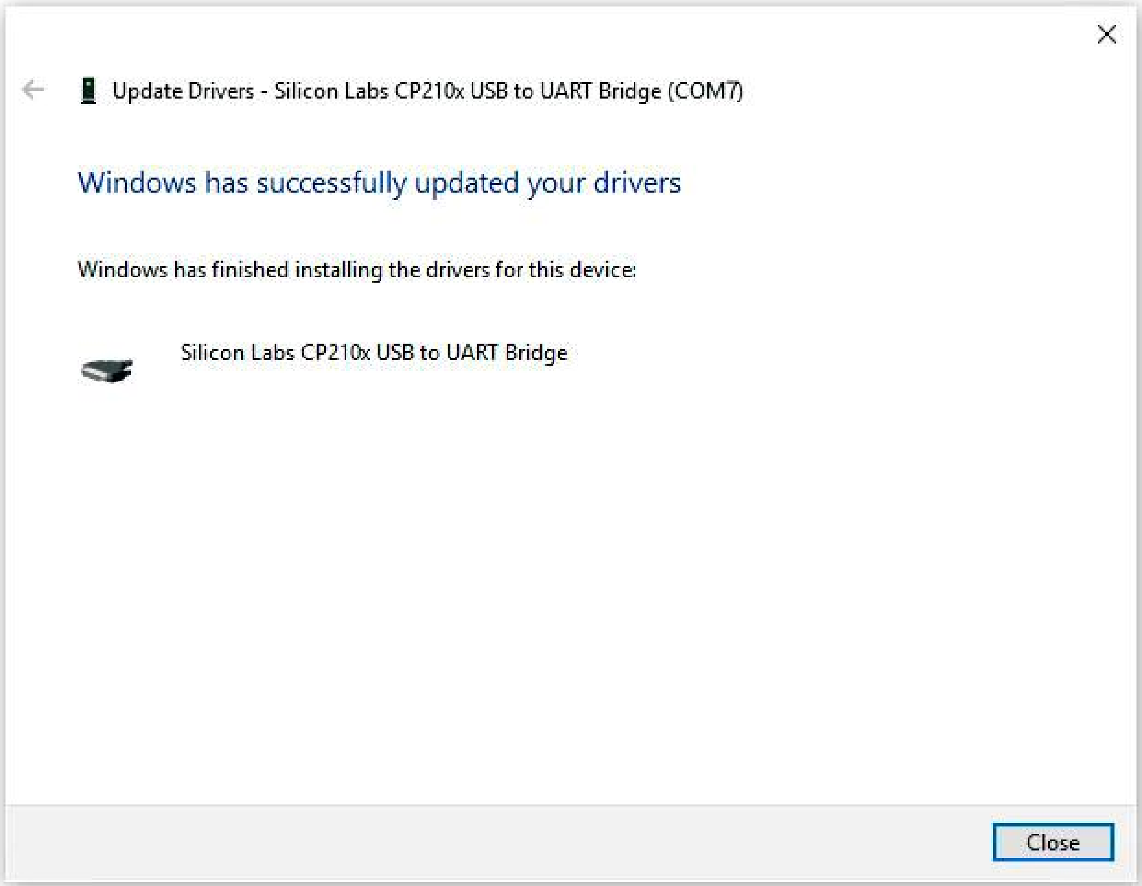
Browse for drivers on your computer

Search for drivers in this location:

C:\Users\Administrator\Desktop\arduino-1.8.12\drivers\CP210x_6.7

Browse...
☑ Include subfolders
→ Let me pick from a list of available drivers on my computer
This list will show available drivers compatible with the device, and all drivers in the same category as the device.
Next
Cancel

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

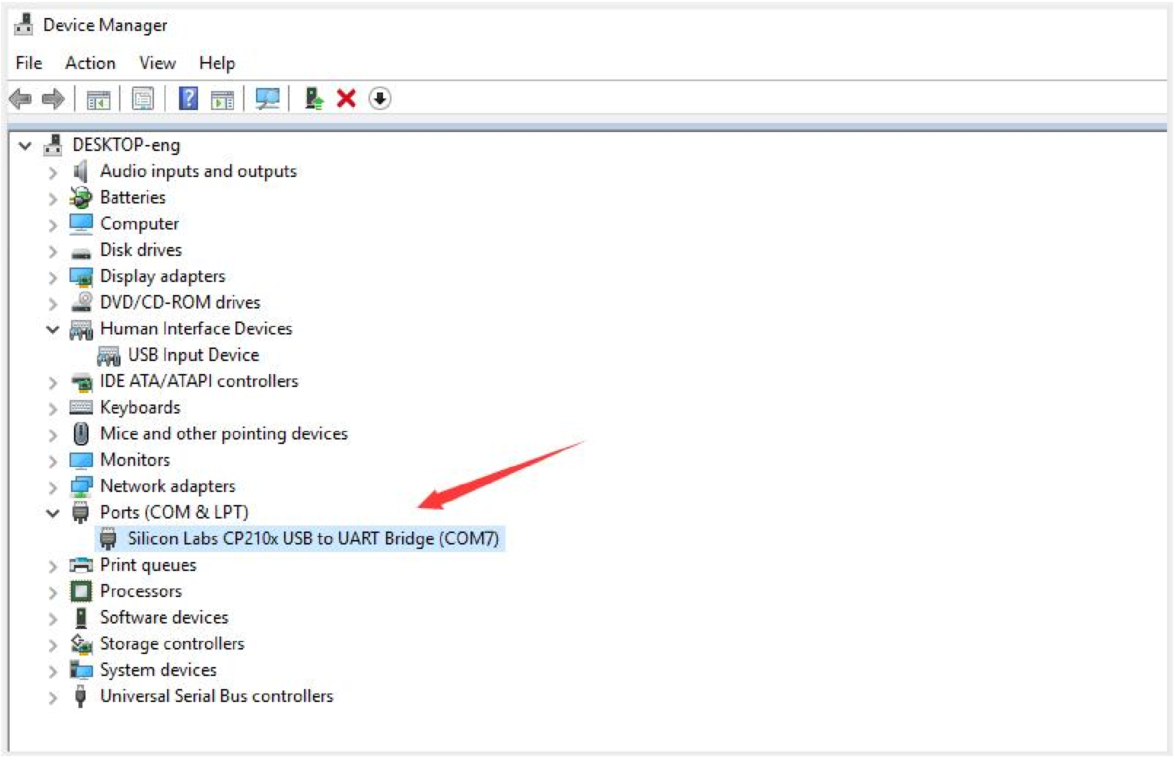


Update Drivers - Silicon Labs CP210x USB to UART Bridge (COM7)

Windows has successfully updated your drivers

Windows has finished installing the drivers for this device:

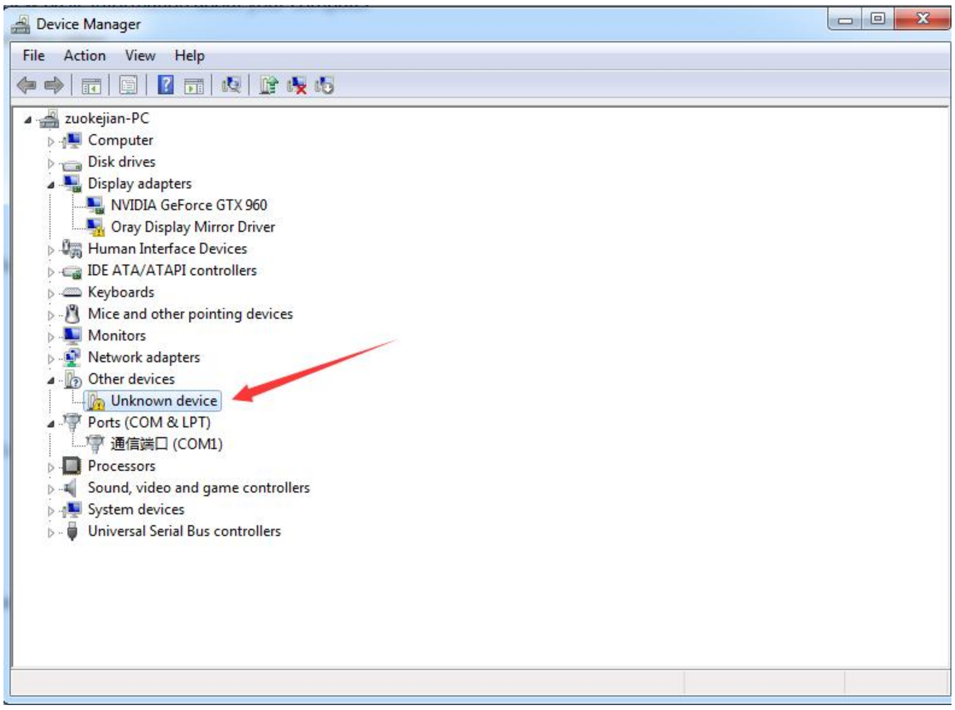
Silicon Labs CP210x USB to UART Bridge



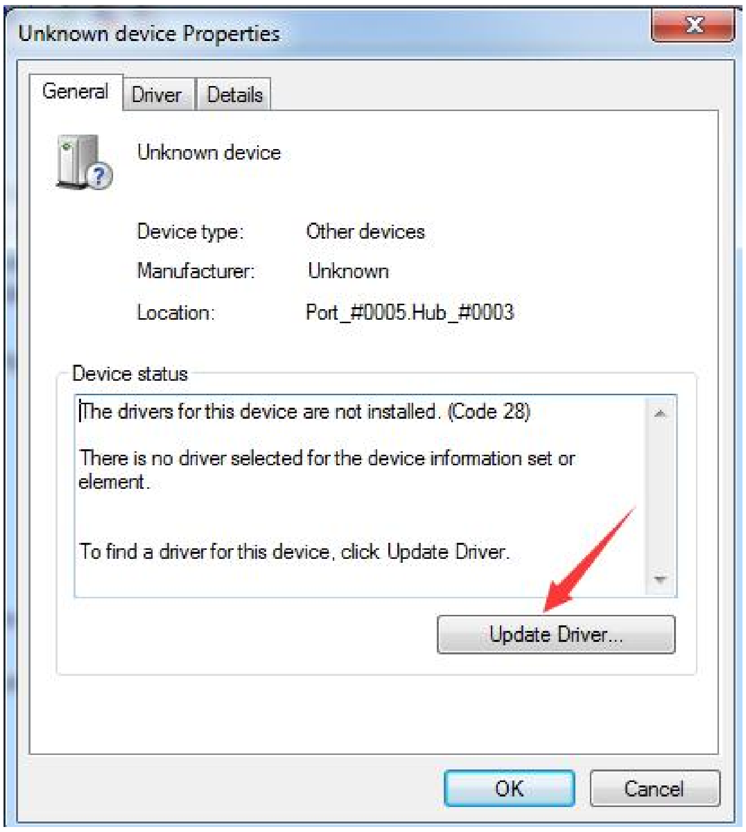
(3) Install other visions of driver

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


Unknown device Properties

Unknown device

Device type: Other devices
Manufacturer: Unknown
Location: Port_#0005.Hub_#0003

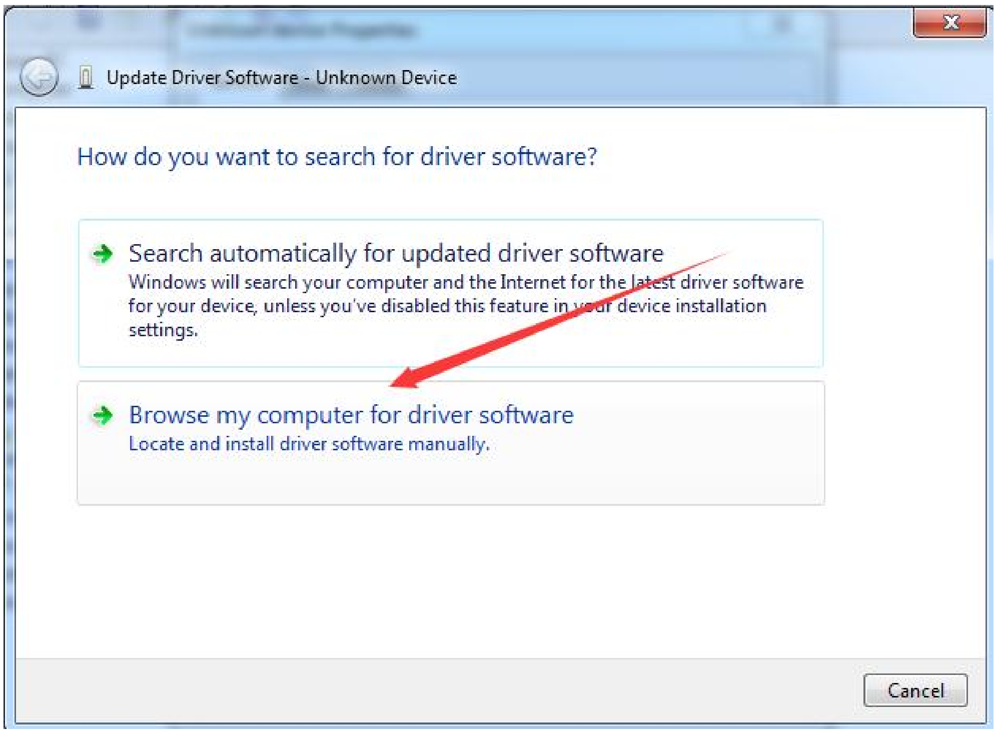
Device status

The drivers for this device are not installed. (Code 28)

There is no driver selected for the device information set or element.

To find a driver for this device, click Update Driver.

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

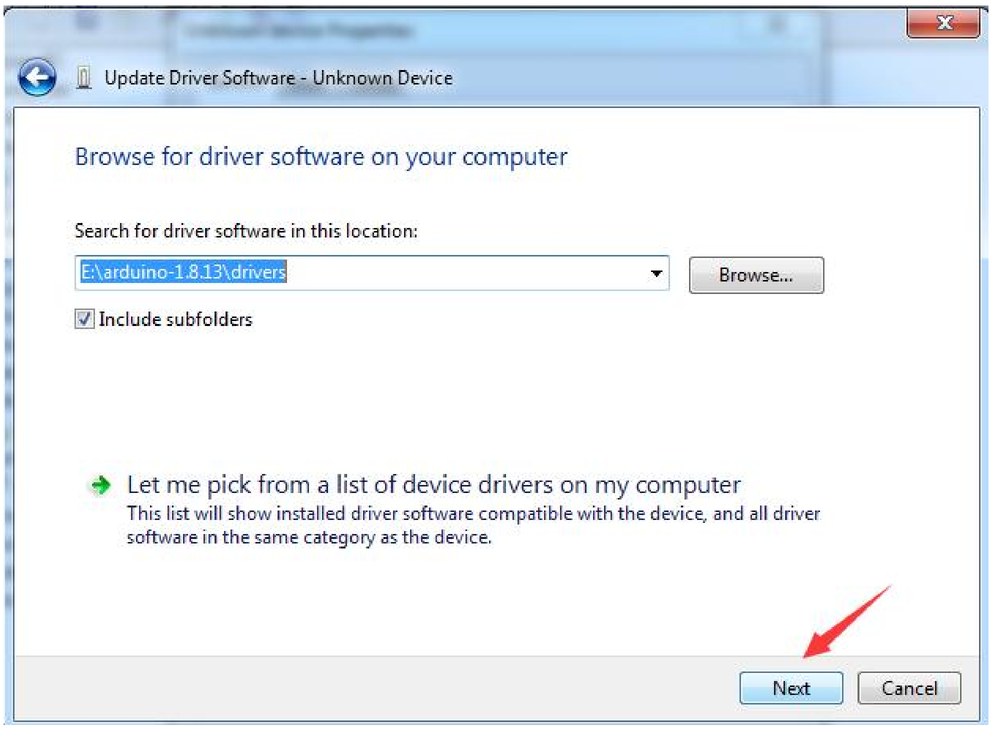

Update Driver Software - Unknown Device

How do you want to search for driver software?

Search automatically for updated driver software
Windows will search your computer and the Internet for the latest driver software for your device, unless you've disabled this feature in your device installation settings.

Browse my computer for driver software
Locate and install driver software manually.

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



Update Driver Software - Unknown Device

Browse for driver software on your computer

Search for driver software in this location:

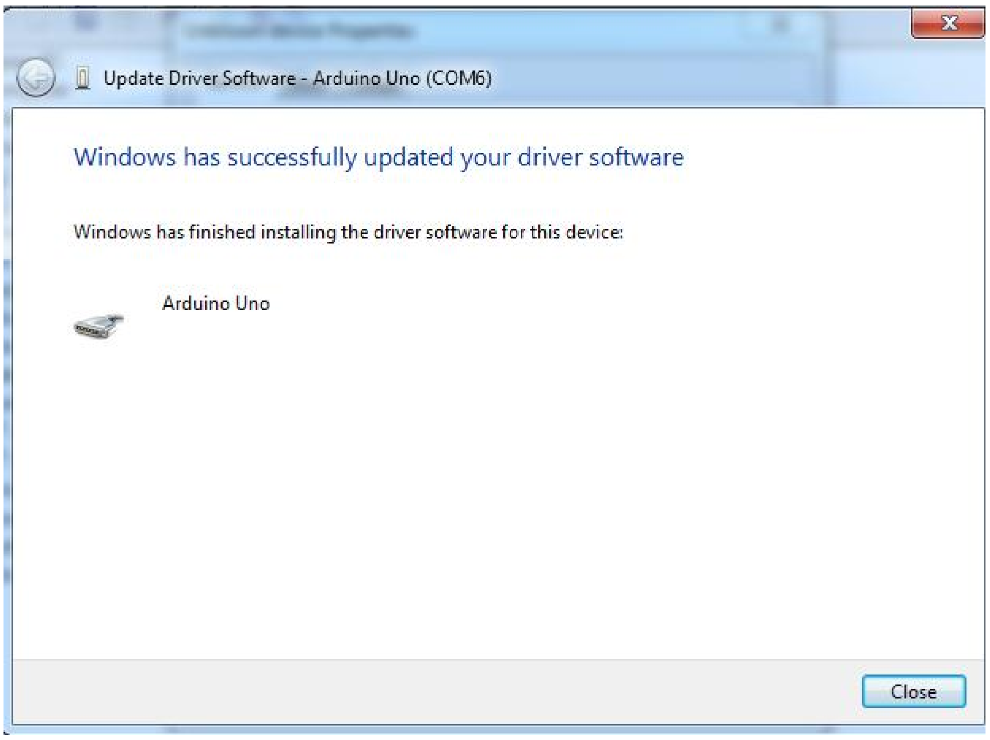
E:\arduino-1.8.13\drivers

[✓] Include subfolders

Let me pick from a list of device drivers on my computer

This list will show installed driver software compatible with the device, and all driver software in the same category as the device.

Step 5: the driver is installed successfully.



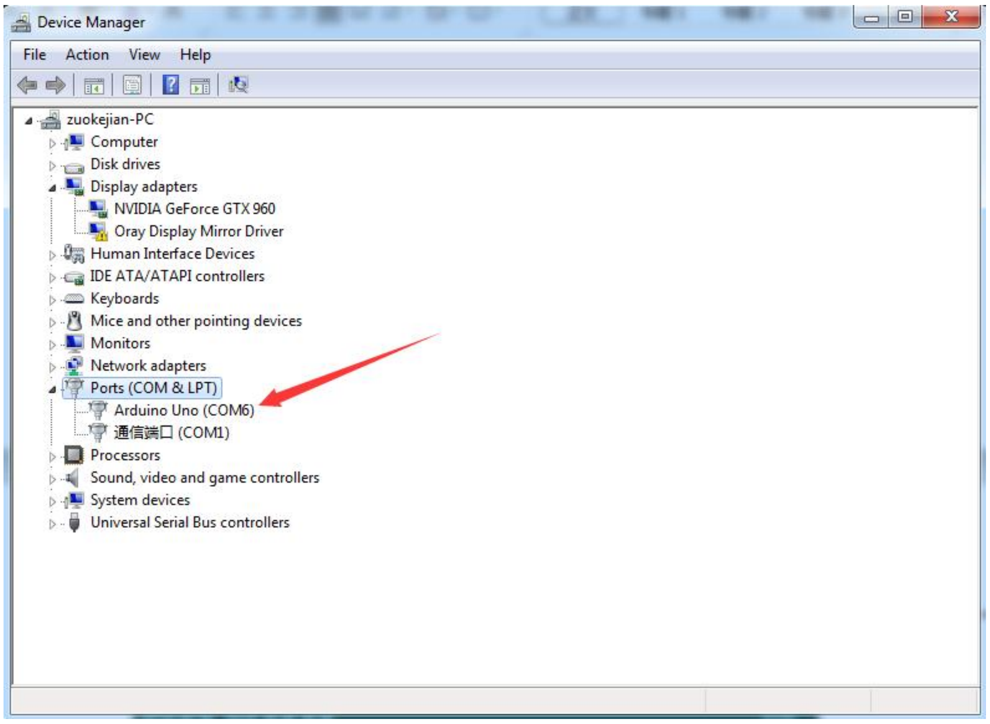
Update Driver Software - Arduino Uno (COM6)

Windows has successfully updated your driver software

Windows has finished installing the driver software for this device:

Arduino Uno

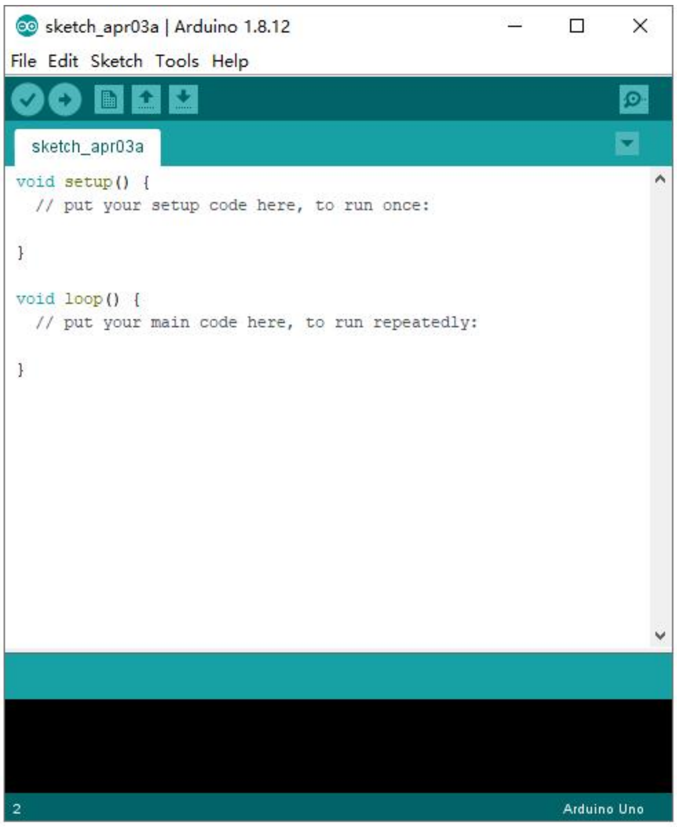
The device manager shows the serial port of Arduino.



zuokejian-PC
- Computer
- Disk drives
- Display adapters
  - NVIDIA GeForce GTX 960
  - Oray Display Mirror Driver
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - Arduino Uno (COM6)
  - 通信端口 (COM1)
- Processors
- Sound, video and game controllers
- System devices
- Universal Serial Bus controllers

(4) Arduino IDE Setting

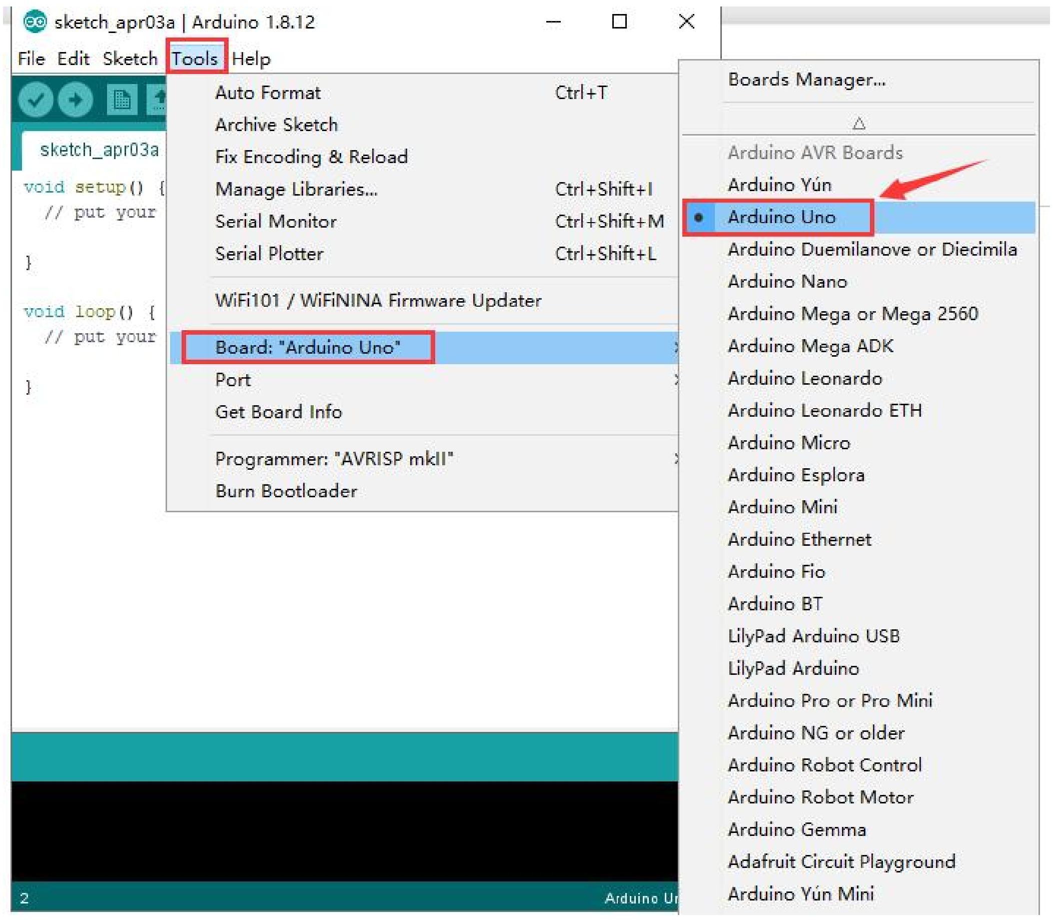
Click Arduino icon，open Arduino IDE.



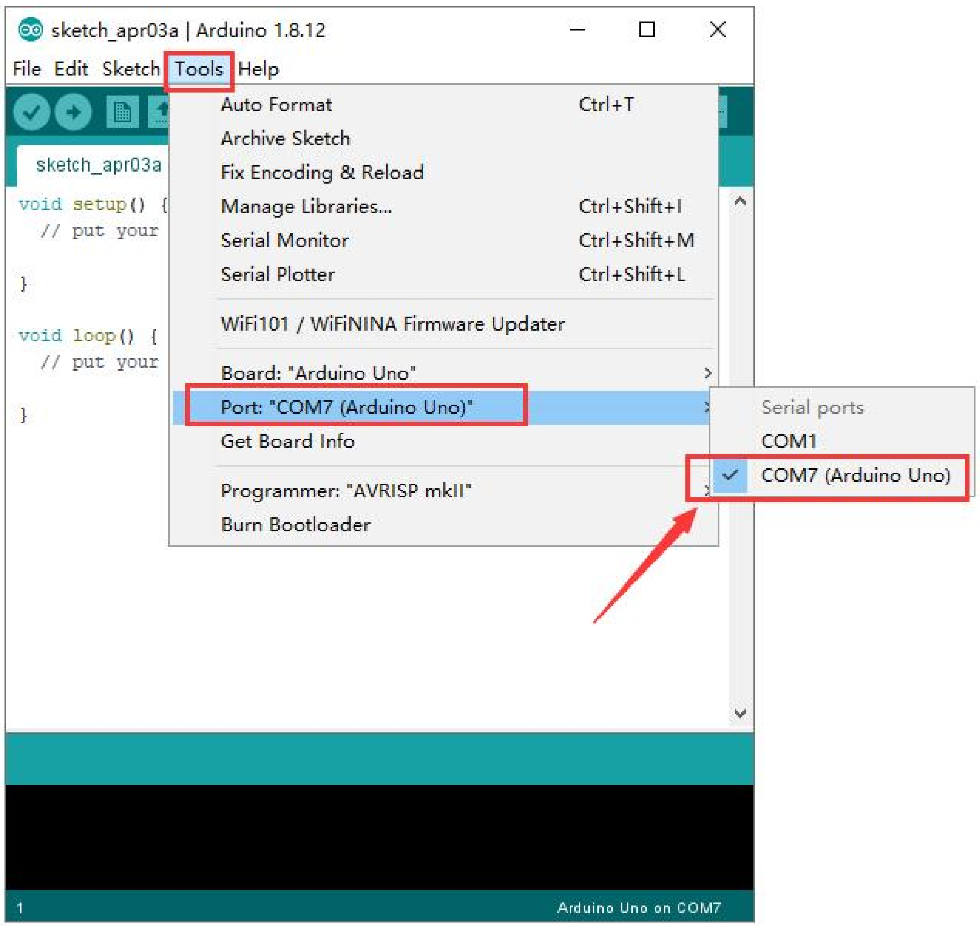
```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

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)

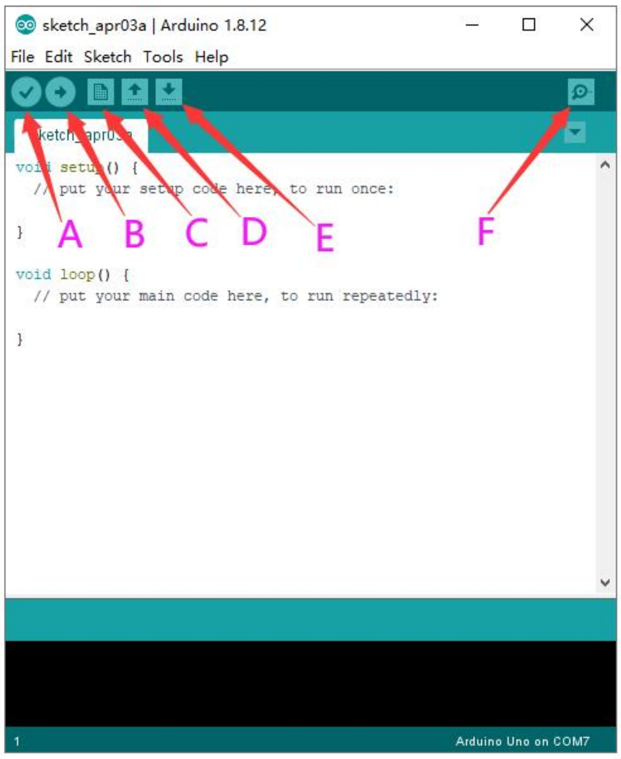


Then select the correct COM port (you can see the corresponding COM port after the driver is successfully installed)


sketch_apr03a | Arduino 1.8.12
File Edit Sketch Tools Help
sketch_apr03a
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM7 (Arduino Uno)"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM1
COM7 (Arduino Uno)
Arduino Uno on COM7

```
void setup() {
  // put your

}

void loop() {
  // put your

}
```

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



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

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.

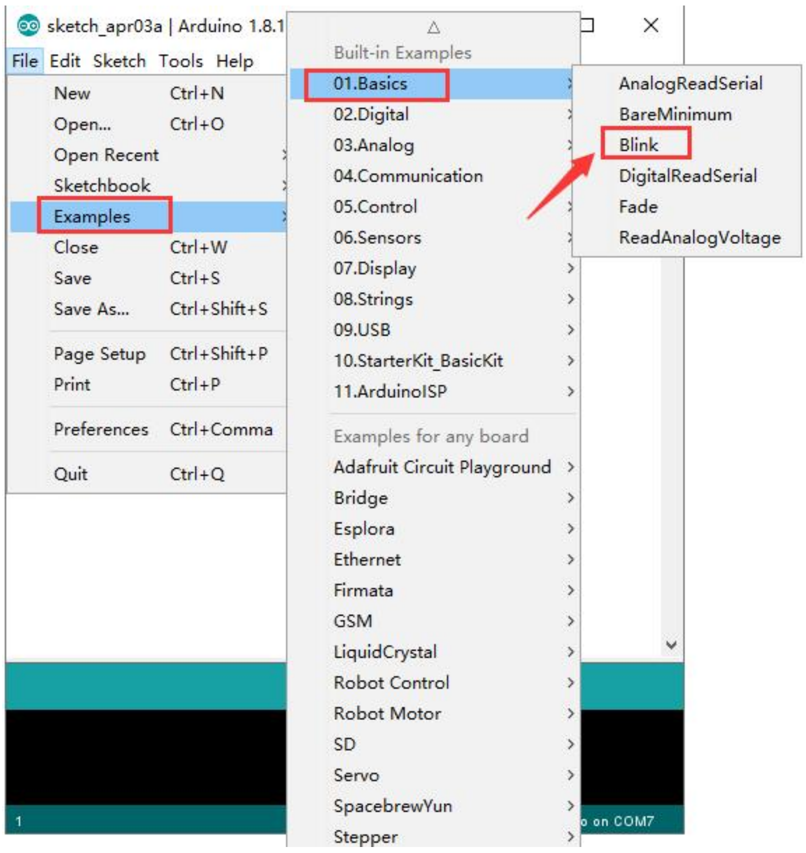
D- Used to directly open an example sketch.

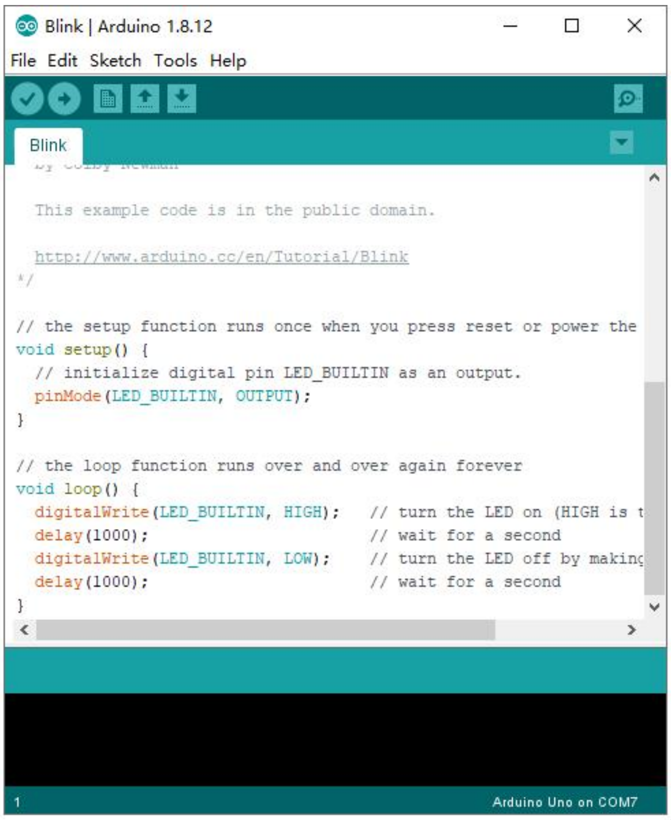
E- Used to save the sketch.

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

(5) Start First Program

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

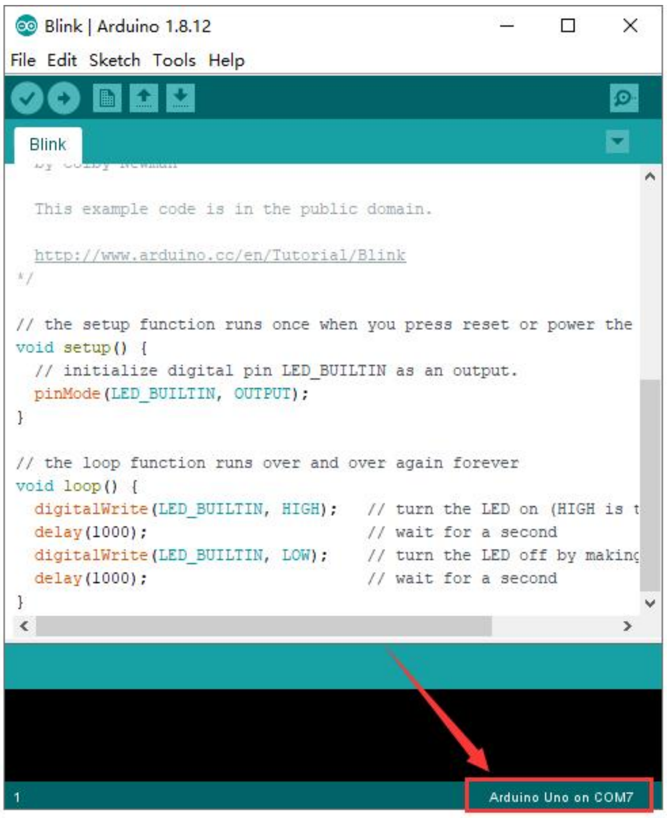




```
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is t
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by makinc
  delay(1000);                       // wait for a second
}
```

1   Arduino Uno on COM7

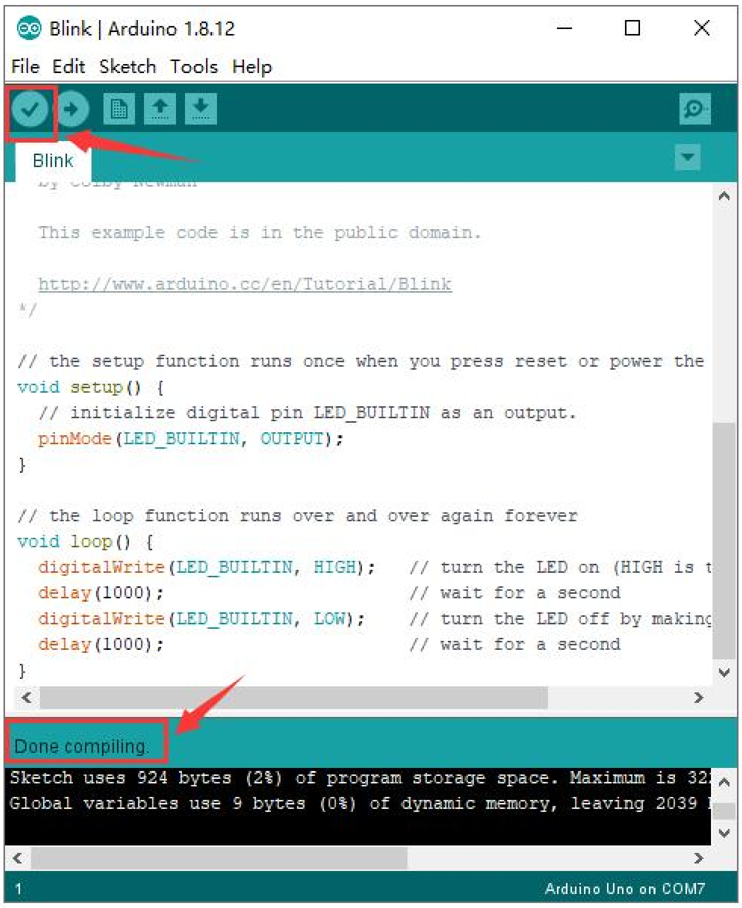
Set board and COM port, the corresponding board and COM port are shown on the lower right of IDE.



```
  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

Arduino Uno on COM7

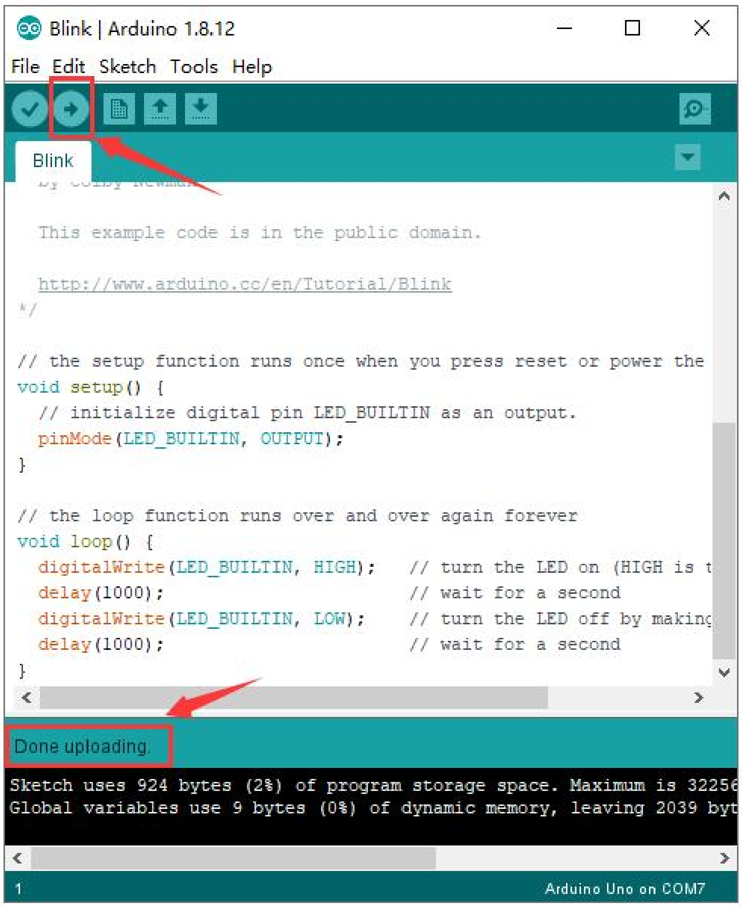
Click ✓ to start compiling the program, check errors.



```
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

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

```
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039
```

Click to upload the program, upload successfully.



```
  by Colby Newman

  This example code is in the public domain.

  http://www.arduino.cc/en/Tutorial/Blink
*/

// the setup function runs once when you press reset or power the
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is t
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by makinc
  delay(1000);                       // wait for a second
}
```

```
Done uploading.
Sketch uses 924 bytes (2%) of program storage space. Maximum is 32256
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039 byt
```

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

7. Add project Libraries

(1) What are Libraries ?

Libraries are a collection of code that makes it easy for you to drive 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.

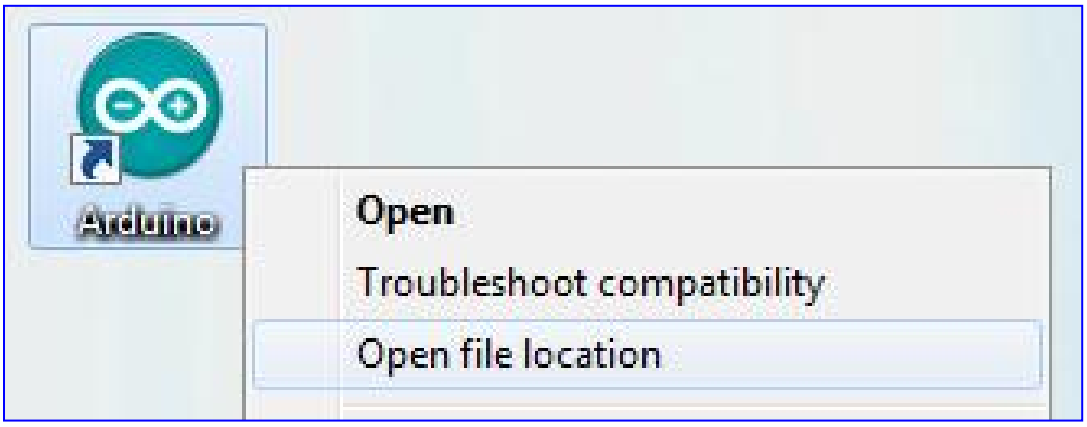
The built-in libraries and some of these additional libraries are listed in the reference.

(2) How to Install a Library ?

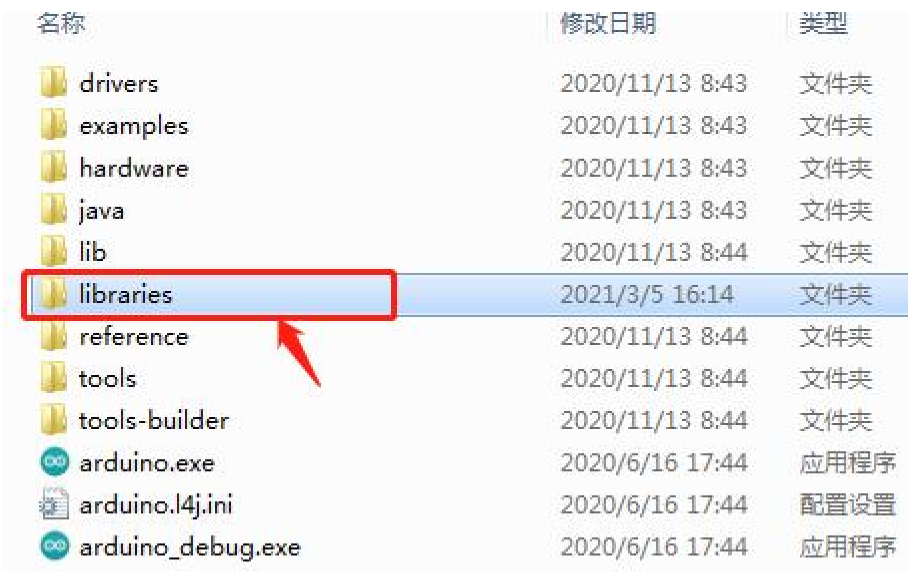
Here we will introduce the most simple way for you to add libraries .

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:

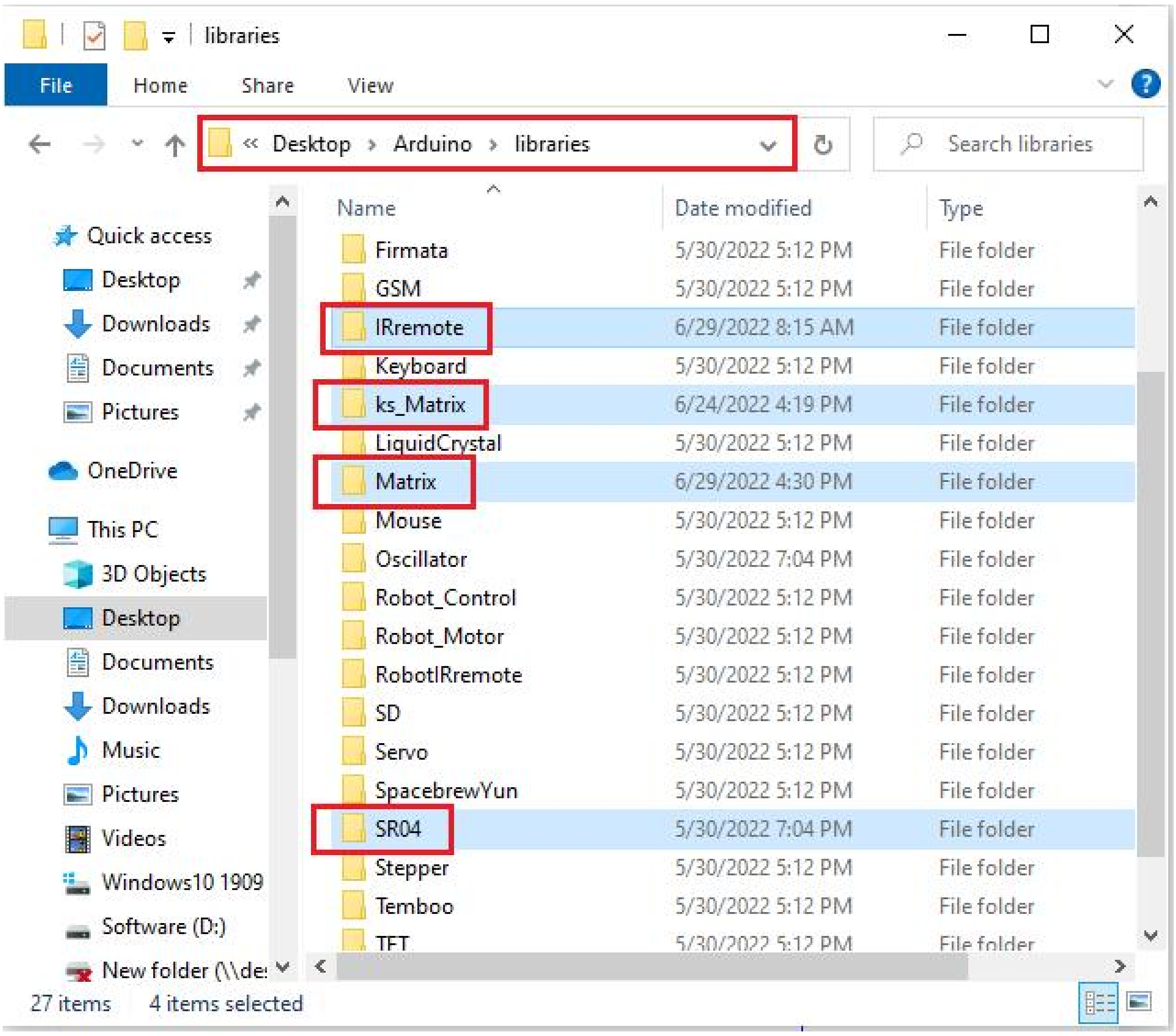


Step 2: Enter it to find out libraries folder, this folder is the library file of Arduino.



名称	修改日期	类型
drivers	2020/11/13 8:43	文件夹
examples	2020/11/13 8:43	文件夹
hardware	2020/11/13 8:43	文件夹
java	2020/11/13 8:43	文件夹
lib	2020/11/13 8:44	文件夹
libraries	2021/3/5 16:14	文件夹
reference	2020/11/13 8:44	文件夹
tools	2020/11/13 8:44	文件夹
tools-builder	2020/11/13 8:44	文件夹
arduino.exe	2020/6/16 17:44	应用程序
arduino.l4j.ini	2020/6/16 17:44	配置设置
arduino_debug.exe	2020/6/16 17:44	应用程序

Step 3: Next to find out the "libraries" folders of turtle robot(seen in the link: https://fs.keyestudio.com/KS0558), you just need to replicate and paste it into the libraries folder of Arduino IDE.



libraries

File Home Share View

« Desktop › Arduino › libraries

Search libraries

Quick access
Desktop
Downloads
Documents
Pictures
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows10 1909
Software (D:)
New folder (\\de:

Name	Date modified	Type
Firmata	5/30/2022 5:12 PM	File folder
GSM	5/30/2022 5:12 PM	File folder
IRremote	6/29/2022 8:15 AM	File folder
Keyboard	5/30/2022 5:12 PM	File folder
ks_Matrix	6/24/2022 4:19 PM	File folder
LiquidCrystal	5/30/2022 5:12 PM	File folder
Matrix	6/29/2022 4:30 PM	File folder
Mouse	5/30/2022 5:12 PM	File folder
Oscillator	5/30/2022 7:04 PM	File folder
Robot_Control	5/30/2022 5:12 PM	File folder
Robot_Motor	5/30/2022 5:12 PM	File folder
RobotIRremote	5/30/2022 5:12 PM	File folder
SD	5/30/2022 5:12 PM	File folder
Servo	5/30/2022 5:12 PM	File folder
SpacebrewYun	5/30/2022 5:12 PM	File folder
SR04	5/30/2022 7:04 PM	File folder
Stepper	5/30/2022 5:12 PM	File folder
Temboo	5/30/2022 5:12 PM	File folder
TFT	5/30/2022 5:12 PM	File folder

27 items 4 items selected

8.Projects

The whole project begins with basic programs. Starting from simple to complex, the lessons will guide you to assemble the 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 now. Let’ s get started.

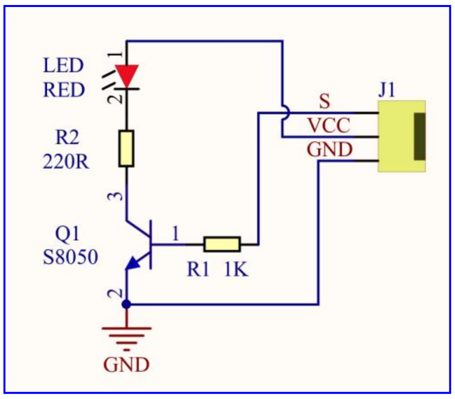
Note: (G), marked on each sensor and module, is the negative pole and connected to “G” , or “GND” on the sensor shield or control board ; (V) is the positive pole and linked with V , VCC or 5V on the sensor shield or control board.

Project 1: LED Blink

(1) Description

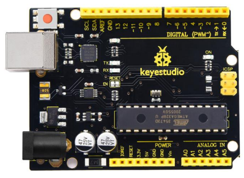
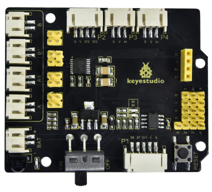
For starters and enthusiasts, LED Blink is a fundamental program. LED, the abbreviation of light emitting diodes, consists of Ga, As, P, N chemical compounds and so on. The LED can flash in diverse color by altering the delay time in the test code. When in control, power on GND and VCC, the LED will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

(2) Specification



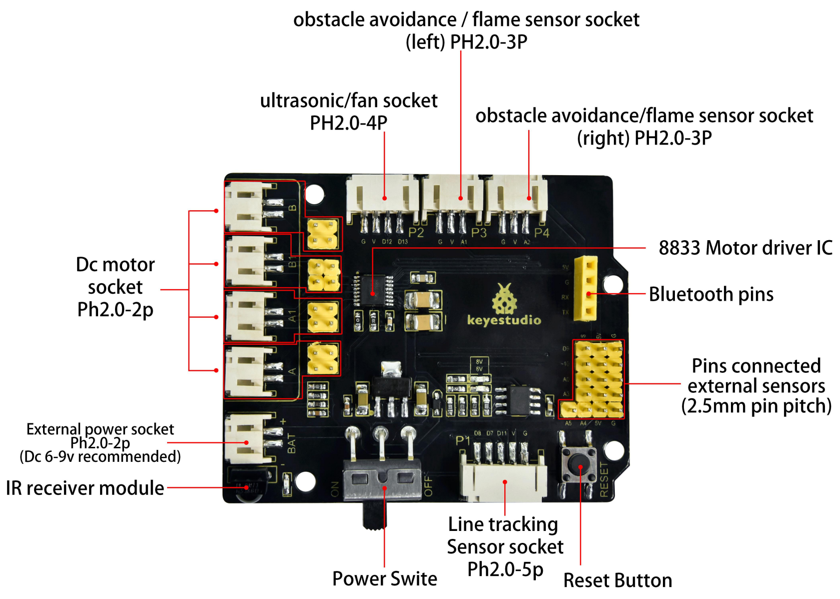
- Control interface: digital port
- Working voltage: DC 3.3-5V
- Pin spacing: 2.54mm
- LED display color: red

(3) Components

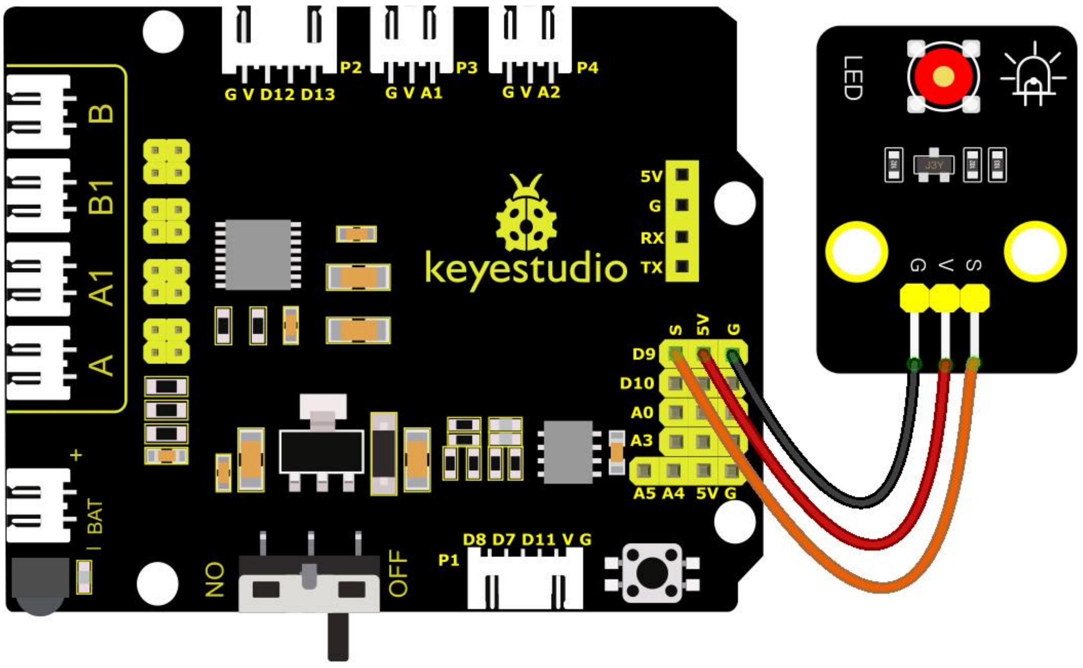
Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	Red LED Module*1
		
3P F-F Dupont Wire*1	USB cable*1	

(4) Keyestudio 8833 motor driver expansion board:

The Keyestudio 8833 motor driver expansion board is compatible with the keyestudio development board. Just stack it onto the development board


obstacle avoidance / flame sensor socket (left) PH2.0-3P
ultrasonic/fan socket PH2.0-4P
obstacle avoidance/flame sensor socket (right) PH2.0-3P
8833 Motor driver IC
Bluetooth pins
Dc motor socket Ph2.0-2p
Pins connected external sensors (2.5mm pin pitch)
External power socket Ph2.0-2p (Dc 6-9v recommended)
IR receiver module
Line tracking Sensor socket Ph2.0-5p
Power Swite
Reset Button

(5) Wiring Diagram



G, V and S of the LED module are connected to G, 5V and D9

(6) Test Code

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 1.1
 Blink
 http://www.keyestudio.com
*/
void setup()
{
  pinMode(9, OUTPUT);// initialize digital pin 9 as an output.
}

void loop() // the loop function runs over and over again forever
{
  digitalWrite(9, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(9, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
//*************************************************************************
```

Upload the code and power up. The LED connected to D9 will blink

(7) Code Explanation

pinMode(9, OUTPUT) - This function can denote that the pin is INPUT or OUTPUT

digitalWrite(9 , HIGH) - When pin is OUTPUT, we can set it to HIGH(output 5V) or LOW(output 0V)

(8) Extension Practice

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

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 1.2
 delay
 http://www.keyestudio.com
*/
void setup()
{
  // initialize digital pin 11 as an output.
  pinMode(9, OUTPUT);
}
// the loop function runs over and over again forever
void loop()
```

```
{
  digitalWrite(9, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(100); // wait for 0.1 second
  digitalWrite(9, LOW); // turn the LED off by making the voltage LOW
  delay(100); // wait for 0.1 second
}
//*****************************************************************
```

The test result shows that the LED flashes faster. Therefore, pins and time delaying affect flash frequency.

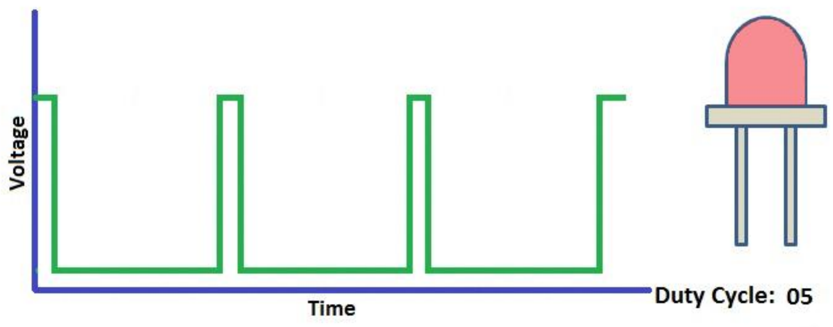
Project 2: Adjust LED Brightness

(1) Description

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

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

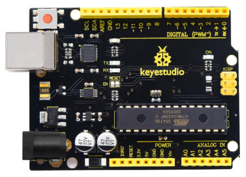
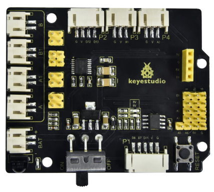
PWM is a means of controlling the analog output via digital means. Digital control is used to generate square waves with different duty cycles (a signal that constantly switches between high and low levels) to control the analog output.In general, the input voltages of ports are 0V and 5V. What if the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



For 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.

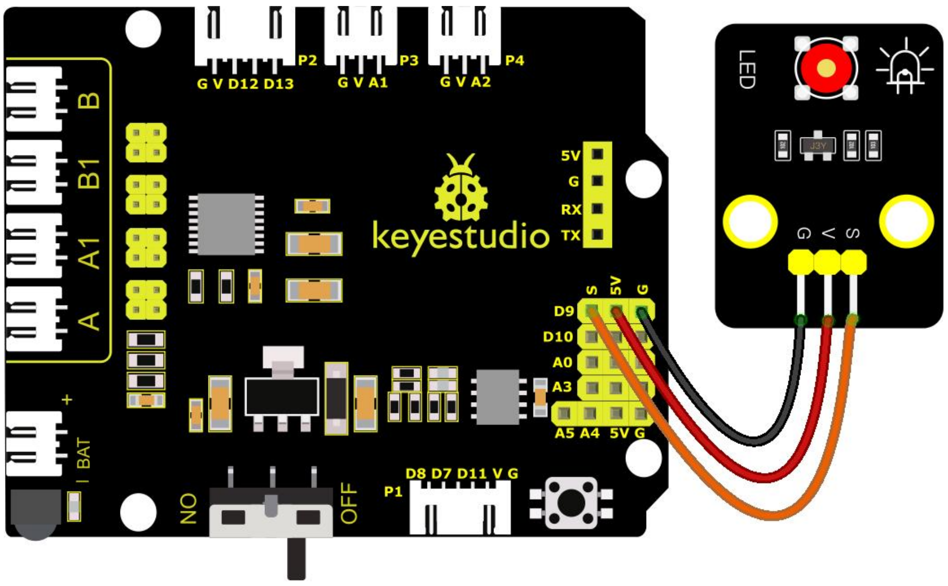
If output five hundred 1, that is 5V; if all of which is 1, that is 0V. If output 010101010101 in this way then the output port is 2.5V, which is like showing movie. The movie we watch are not completely continuous. It actually outputs 25 pictures per second. In this case, the human can’t tell it, neither 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) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	Red LED Module*1
		
3P F-F Dupont Wire*1	USB cable*1	

(3) Wiring Diagram

Keep the wiring-up unchanged.



(4) Test Code

```
//*******************************************************************
/*
 keyestudio smart turtle robot
 lesson 2.1
 pwm
 http://www.keyestudio.com
*/
int ledPin = 9; // Define the LED pin at D9
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
  }
}
//*******************************************************************
```

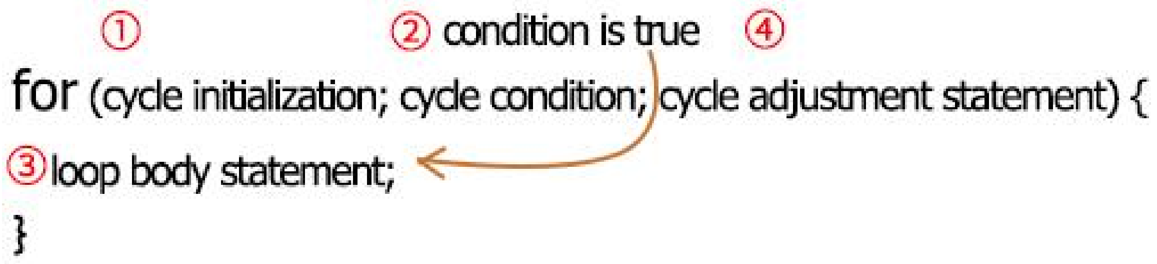
(5) Test Result:

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

(6) Code Explanation

o repeat some certain statements, we could use FOR statement.

FOR statement format is shown below:


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

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.

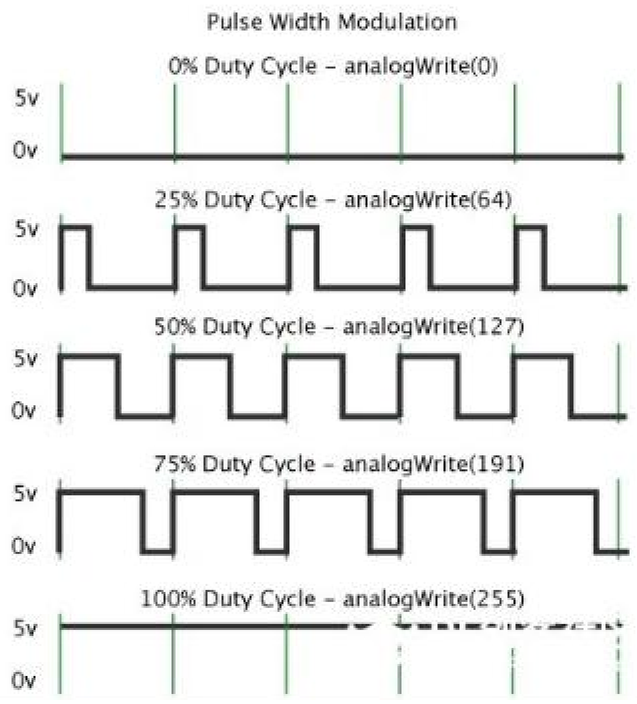
Function format as follows:

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.


Pulse Width Modulation
0% Duty Cycle - analogWrite(0)
25% Duty Cycle - analogWrite(64)
50% Duty Cycle - analogWrite(127)
75% Duty Cycle - analogWrite(191)
100% Duty Cycle - analogWrite(255)
5v
0v

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

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

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

(7) Extension Practice:

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

```
//*************************************************************
/*
 keyestudio smart turtle robot
 lesson 2.2
 pwm
 http://www.keyestudio.com
*/
int ledPin = 9; // Define the LED pin at D9
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, then LED blinks more slowly.

Project 3 : Line Tracking Sensor

(1) Description

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

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

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

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

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

(2) Specification:

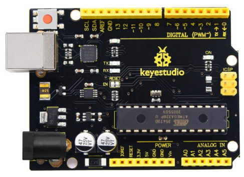
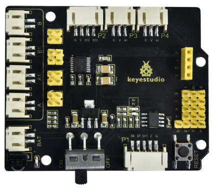
Operating Voltage: 3.3-5V (DC)

Interface: 5PIN

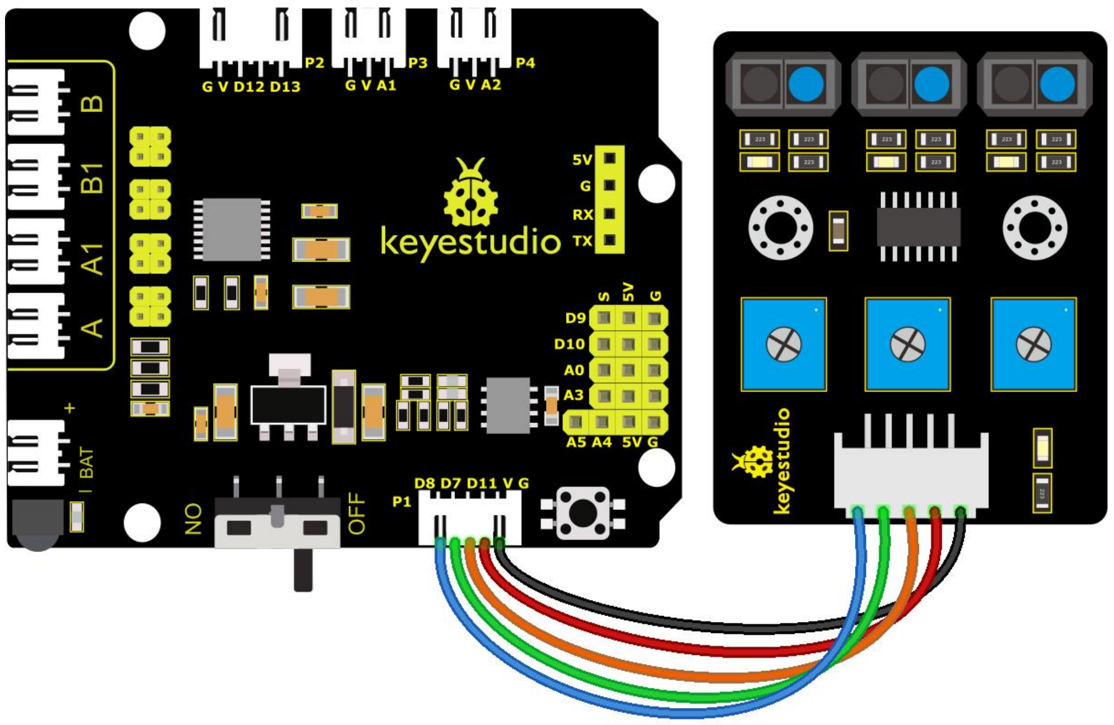
Output Signal: Digital signal

Detection Height: 0-3 cm

(3) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1		Red LED Module*1
			
5P Dupont wire*1	USB cable*1	3P F-F Dupont wire*1	Keyestudio line tracking sensor*1

(4) Wiring Diagram



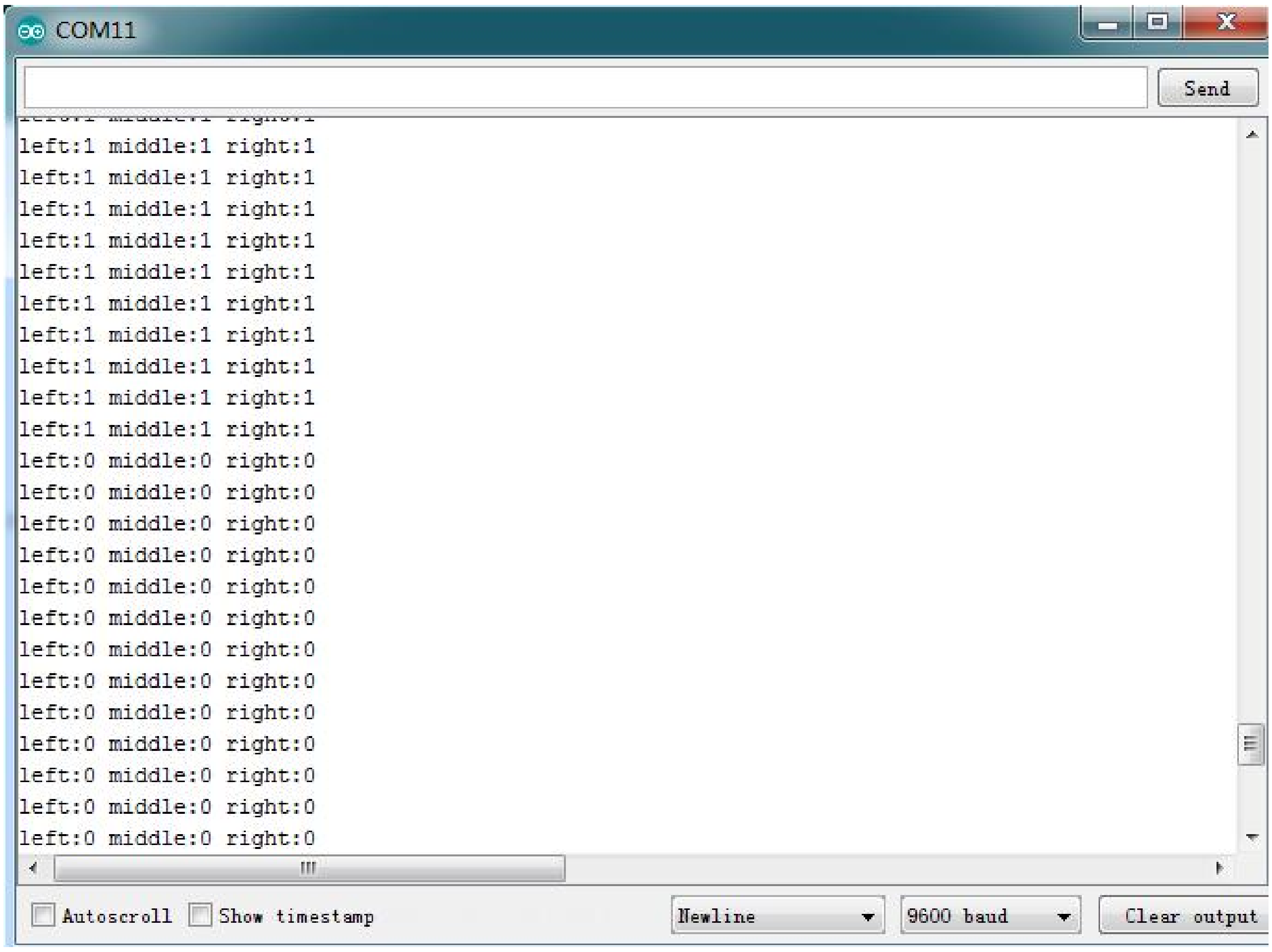
G, V, S1, S2 and S3 of the line tracking sensor are connected to G (GND), V (VCC), D11, D7 and D8 of the sensor expansion board.

(5) Test Code

```
//*****************************************************************************
/*
keyestudio smart turtle robot
lesson 3.1
 Line Track sensor
 http://www.keyestudio.com
*/
int L_pin = 11;  //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 variable value 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
}

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, wire up the 4.0 board with a USB cable and open the serial monitor. You can view status of three line tracking sensors. When not signals are received, the value is 1. If we cover the sensor with a white paper, the value will be 0


COM11

Send

```
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:1 middle:1 right:1
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
left:0 middle:0 right:0
```

Autoscroll Show timestamp Newline 9600 baud Clear output

(7) Code Explanation

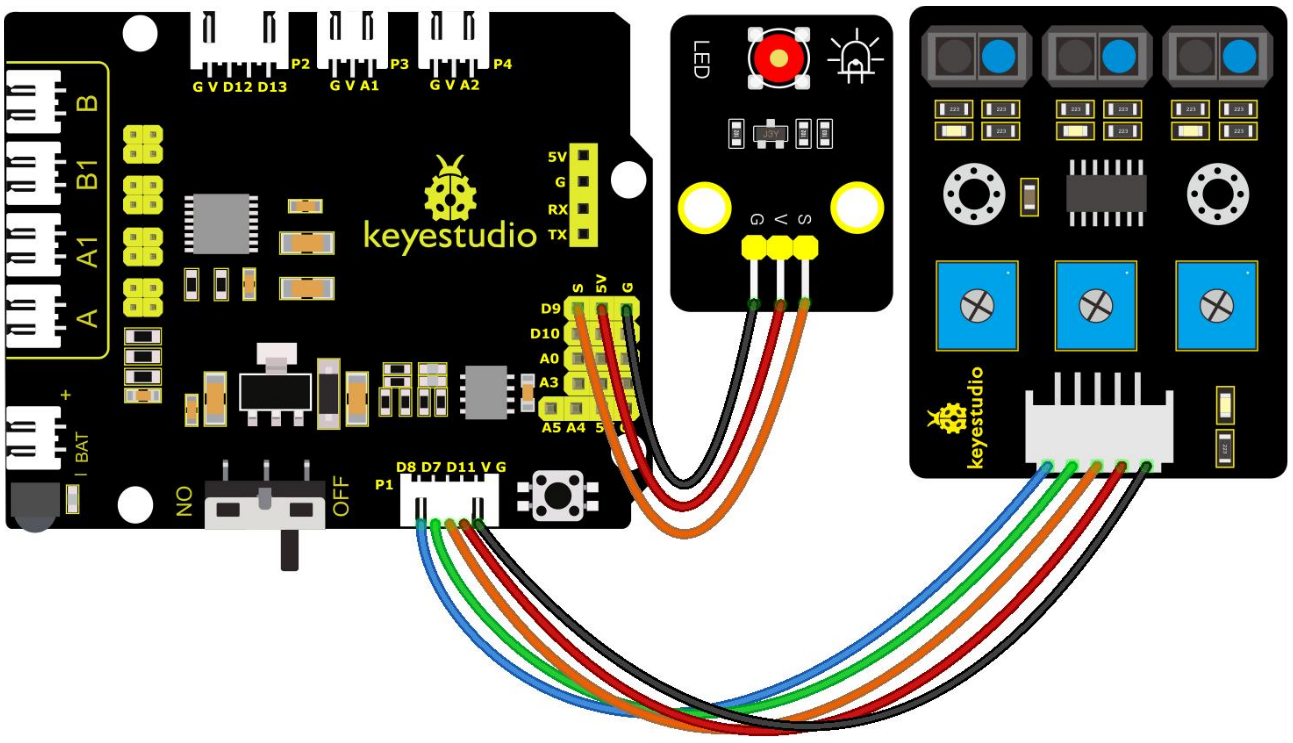
Serial.begin(9600)- Initialize serial port, set baud rate to 9600

pinMode- Define the pin as input or output mode

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

(8) Extension Practice

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



```
//*************************************************************************
/*
keyestudio smart turtle robot
lesson 3.2
 Line Track sensor
 http://www.keyestudio.com
*/
int L_pin = 11;  //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(9, 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);
  delay(500);// delay in between reads for stability
  if ((val_L == LOW) || (val_M == LOW) || (val_R == LOW))//if left line tracking sensor detects
signals
  {
    Serial.println("HIGH");
    digitalWrite(9, HIGH);//LED is off
  }
  else//if left line tracking sensor doesn't detect signals
  {
    Serial.println("LOW");
    digitalWrite(9, LOW);//LED is off
  }
 }
//*************************************************************************
```

Upload the code to development board and power it up with a USB cable.

Make a paper close to the sensor.

Then we can find LED light up when covering the line tracking sensor

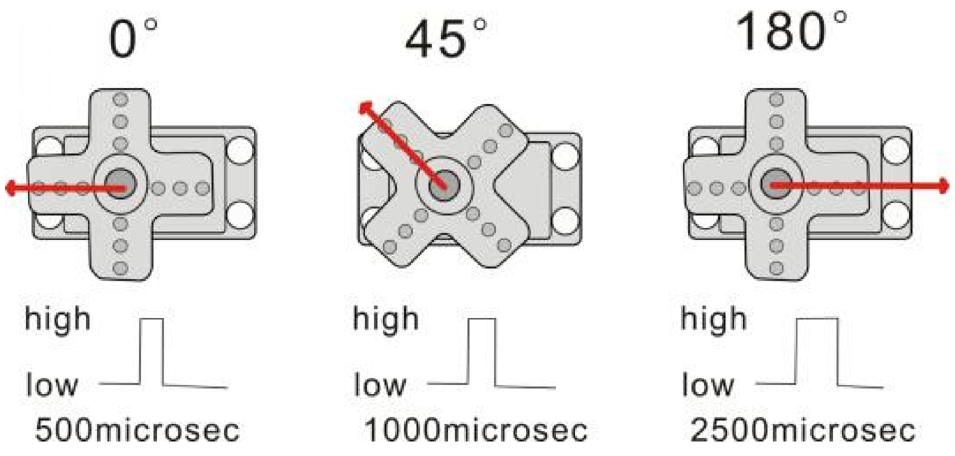
Project 4: Servo Control

(1) Description

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



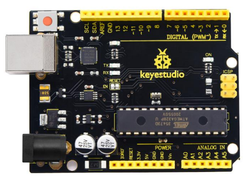
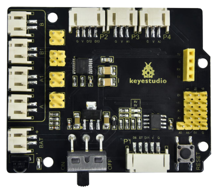
The corresponding servo angles are shown below:

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

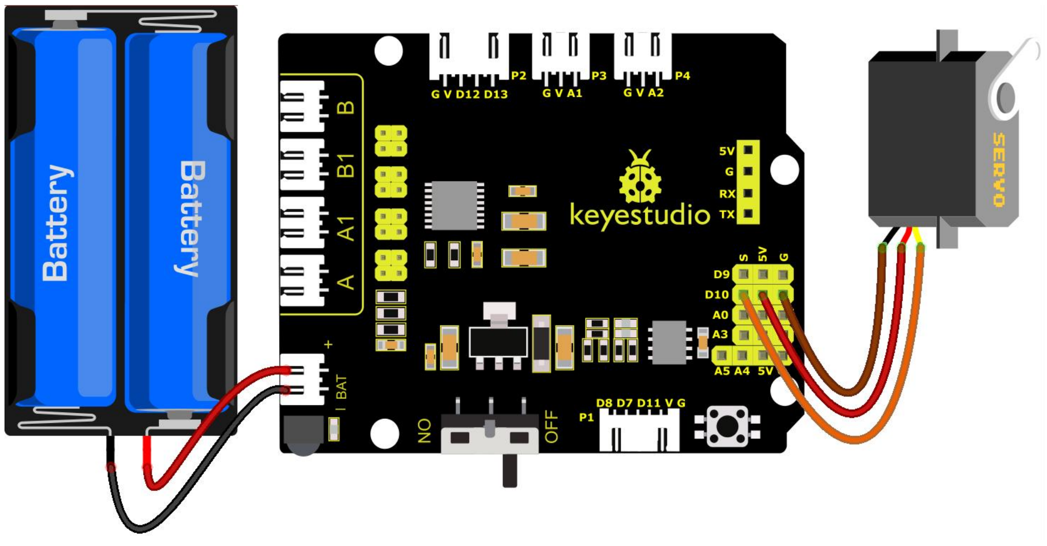
(2) Specification

- 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) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	Servo*1
		
18650 Battery Holder*1	USB cable*1	

(4) Wiring Diagram



Wiring note: the brown line of servo is linked with Gnd(G), the red one is connected to 5v(V) and the orange one 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 big enough. If without connected power, the development board could be burnt.

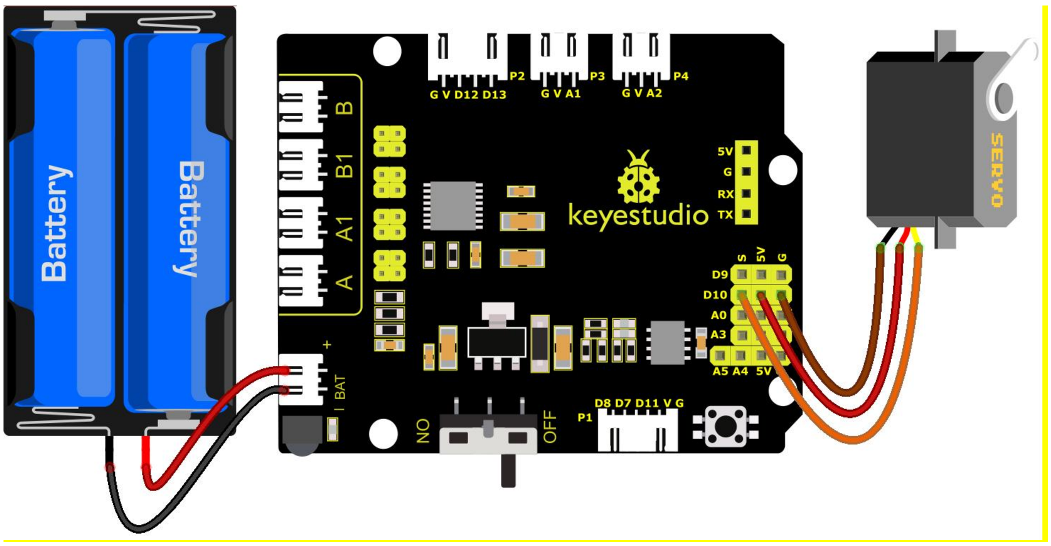
(5) Test Code1:

```
//*************************************************************************
/*
keyestudio smart turtle robot
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 and power the 4.0 board up with a USB cable, then servo will swing 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



(6) Test Code2:

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 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 with a USB cable, then the servo swings in the range of 0° to 180°. 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)**——Used for the statement to set rotation angle of servo, and the set angle range is from 0° to 180°

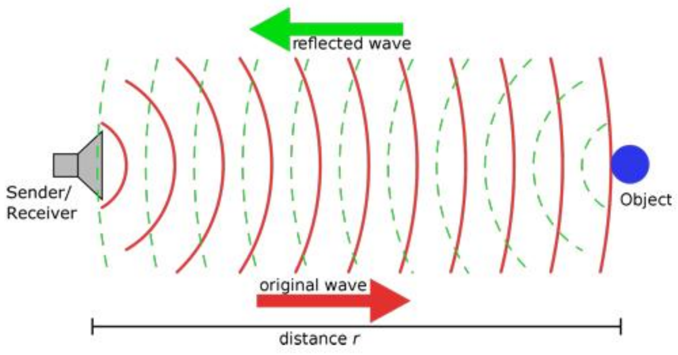
3. **read ()** ——used for the statement to read angle of servo, namely, reading the command value of "write()"

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 what bats do. It offers excellent non-contact range detection with high accuracy and stable readings in an easy-to-use package. It comes complete with ultrasonic transmitter and receiver modules.

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

(2) Parameters:

Power Supply :+5V DC

Quiescent Current : <2mA

Working Current: 15mA

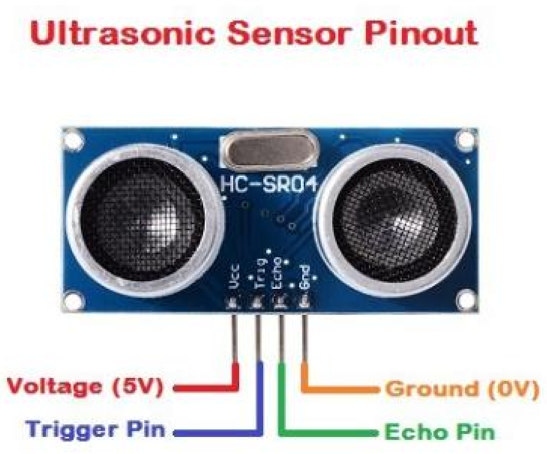
Effectual Angle: <15°

Ranging Distance : 2cm – 400 cm

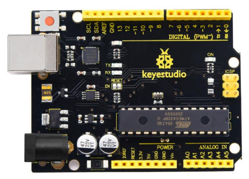
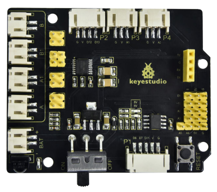
Resolution : 0.3 cm

Measuring Angle: 30 degree

Trigger Input Pulse width: 10uS



(3) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1		Red LED Module*1
			
4P Dupont Wire*1	USB cable*1	3P F-F Dupont wire*1	HC-SR04 Ultrasonic Sensor*1

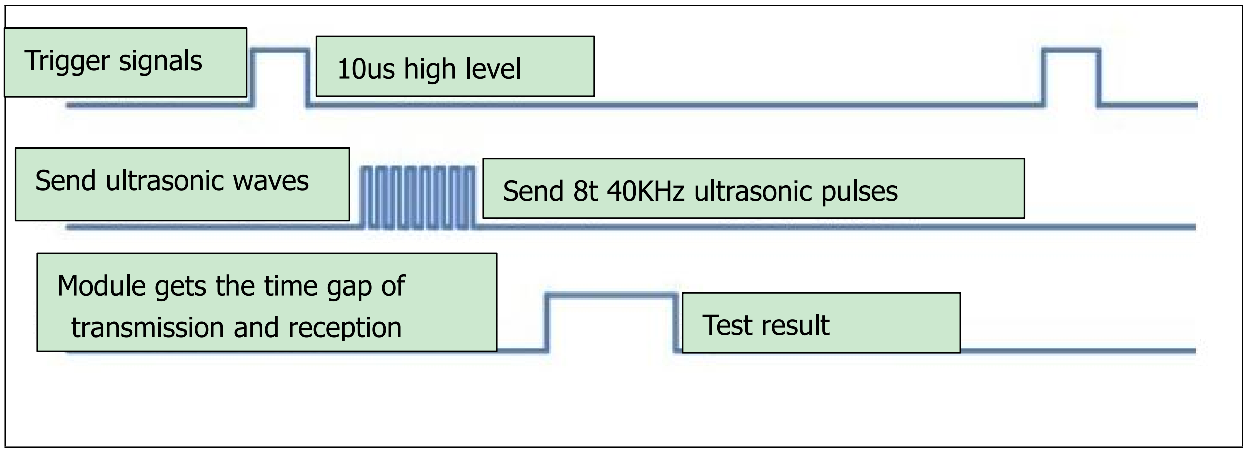
(4) The principle of ultrasonic sensor

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

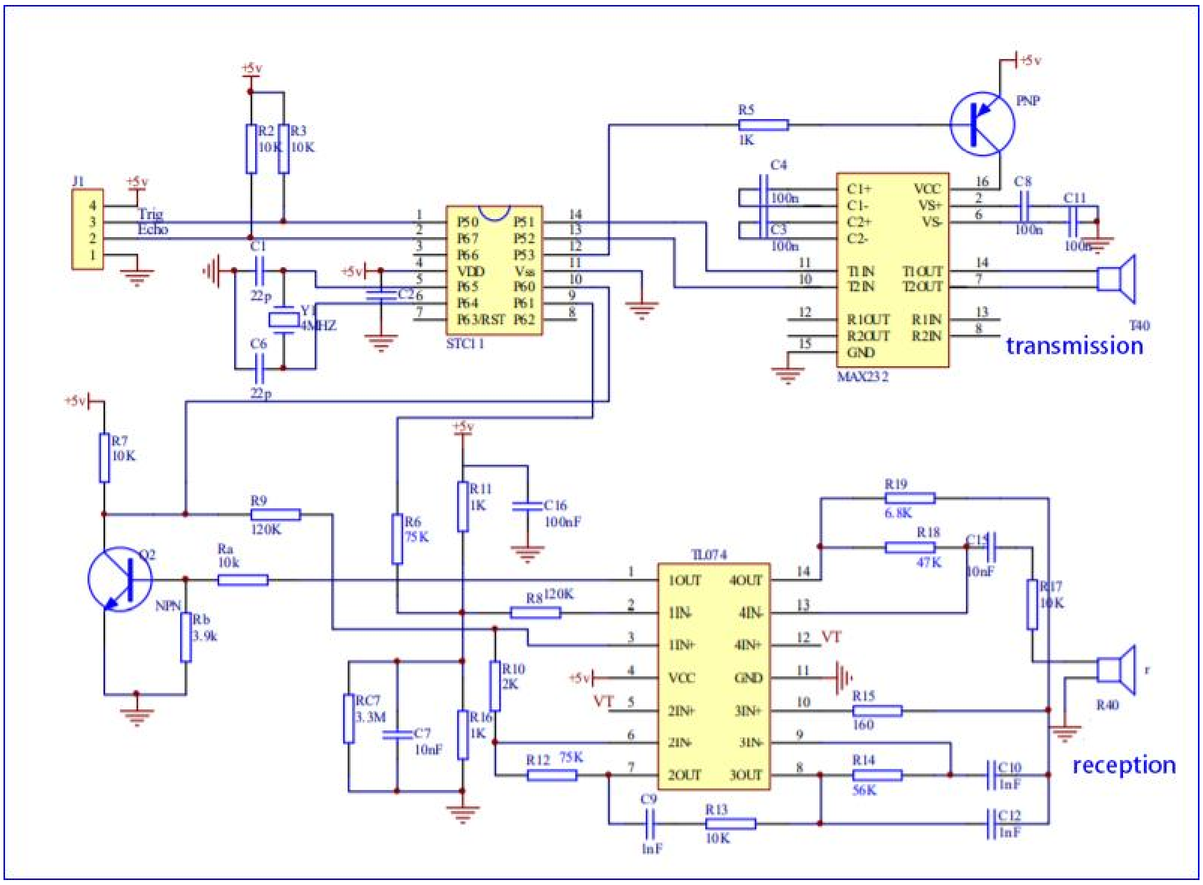
The ultrasonic module will emit the ultrasonic waves after triggering a signal. When the ultrasonic waves encounter the object and are reflected back, the module outputs an echo signal, so it can determine the distance of the object from the time difference between the trigger signal and echo signal.

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

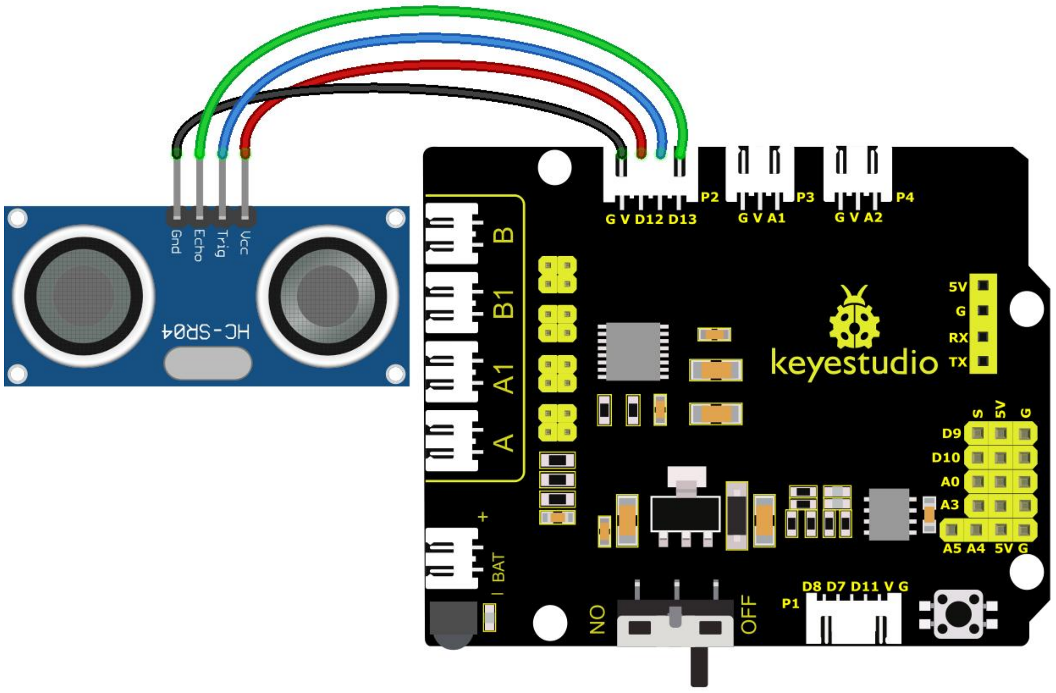
1. Use method and timing chart of ultrasonic module:
2. Setting the delay time of Trig pin of SR04 to 10μs at least, which can trigger it to detect distance.
3. 2. After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step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


Trigger signals
10us high level
Send ultrasonic waves
Send 8t 40KHz ultrasonic pulses
Module gets the time gap of transmission and reception
Test result

Circuit diagram of ultrasonic sensor:



(5) Connection Diagram



VCC, Trig, Echo and Gnd of the ultrasonic sensor are connected to 5V(V), D12(S), D13(S) and Gnd(G)

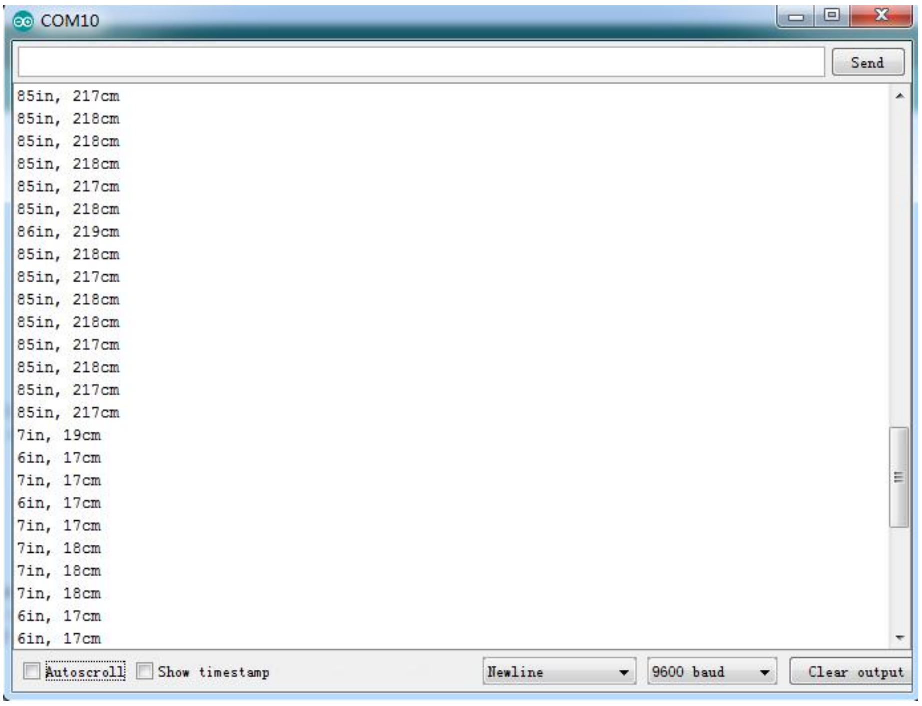
(6) Test Code

```
//*****************************************************************************
/*
 keyestudio smart turtle robot
 lesson 5.1
 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(250);
}
//*****************************************************************************
```

(7) Test Result

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



```
85in, 217cm
85in, 218cm
85in, 218cm
85in, 218cm
85in, 217cm
85in, 218cm
86in, 219cm
85in, 218cm
85in, 217cm
85in, 218cm
85in, 218cm
85in, 217cm
85in, 218cm
85in, 217cm
85in, 217cm
7in, 19cm
6in, 17cm
7in, 17cm
6in, 17cm
7in, 17cm
7in, 18cm
7in, 18cm
7in, 18cm
6in, 17cm
6in, 17cm
```

(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-

inches = (duration/2) / 74-

We can calculate the distance by using the following formula:

distance = (traveltime/2) x speed of sound

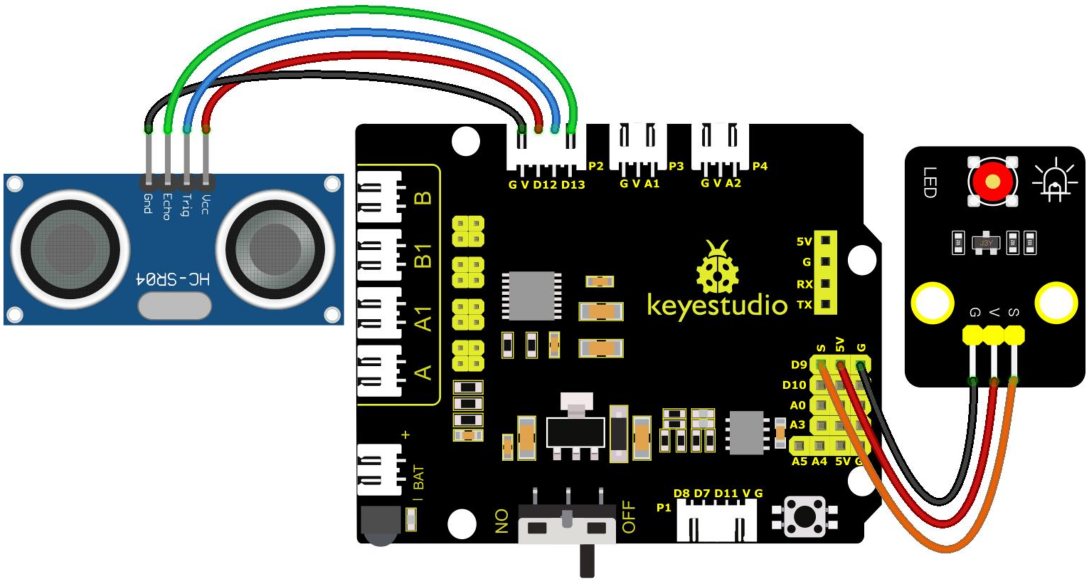
The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1 cm/uS

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 and connect an LED light module to the D9 pin.



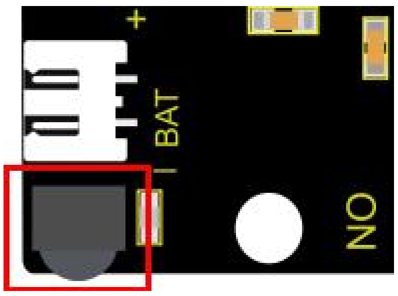
```
//*******************************************************************
/*
 keyestudio smart turtle robot
 lesson 5.2
 Ultrasonic LED
 http://www.keyestudio.com
*/
int trigPin = 12;    // Trigger
int echoPin = 13;    // Echo
long duration, cm, inches;

void setup() {
  Serial.begin (9600);  //Serial Port begin
  pinMode(trigPin, OUTPUT);  //Define inputs and outputs
  pinMode(echoPin, INPUT);
}
```

```
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(250);
  if (cm>=2 && cm<=10)
  {
    Serial.println("HIGH");
    digitalWrite(9, HIGH);
  }
  else
  {
    Serial.println("LOW");
    digitalWrite(9, LOW);
  }
}
//*****************************************************************
```

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

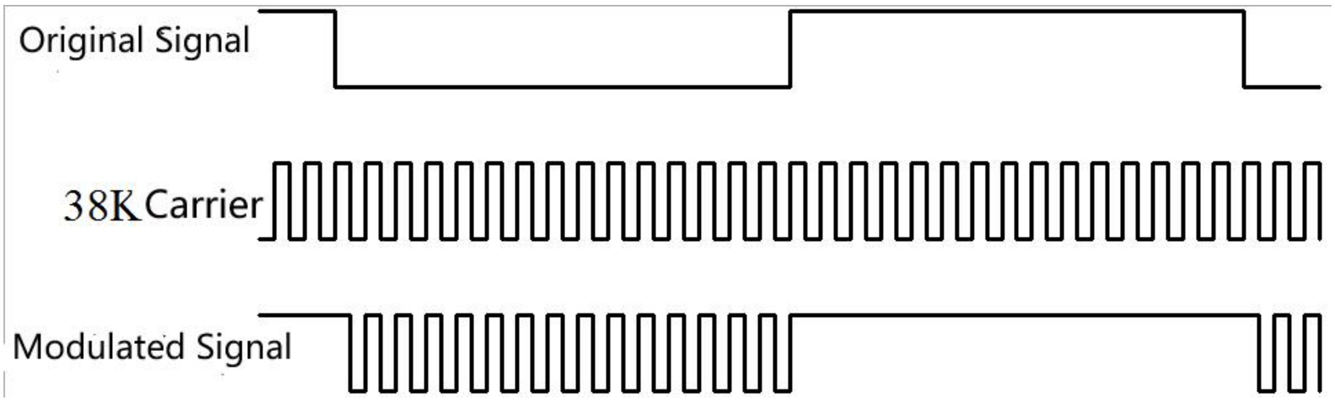
Project 6: IR Reception

(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 constant while the data code can distinguish the key.

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

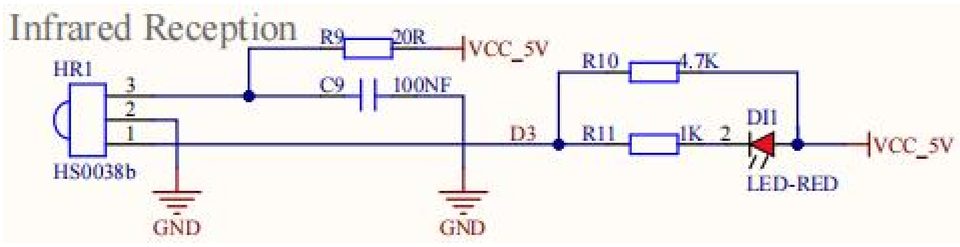
Infrared receiver we use is an infrared receiver module. Mainly composed of an infrared receiver head, which is a device that integrates reception, amplification, and demodulation. Its internal IC has completed demodulation, and can achieve from infrared reception to output and be compatible with TTL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

(2) Specification

Operating Voltage: 3.3-5V (DC)

Interface: 3PIN

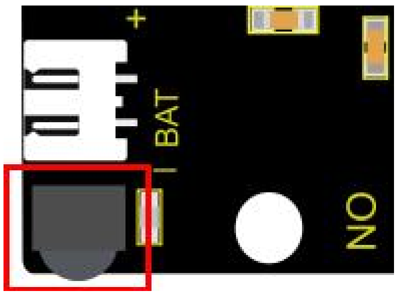
Output Signal: Digital signal



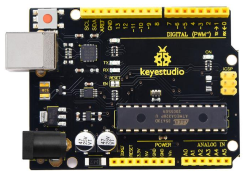
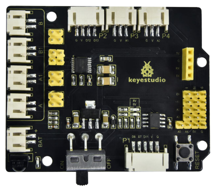
Receiving Angle: 90 degrees

Frequency: 38khz

Receiving Distance: 10m



(3) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	Red LED Module*1
		
3P F-F Dupont Wire*1	USB cable*1	

Since the 8833 board integrates with the IR receiver, it doesn’ t need wiring up.

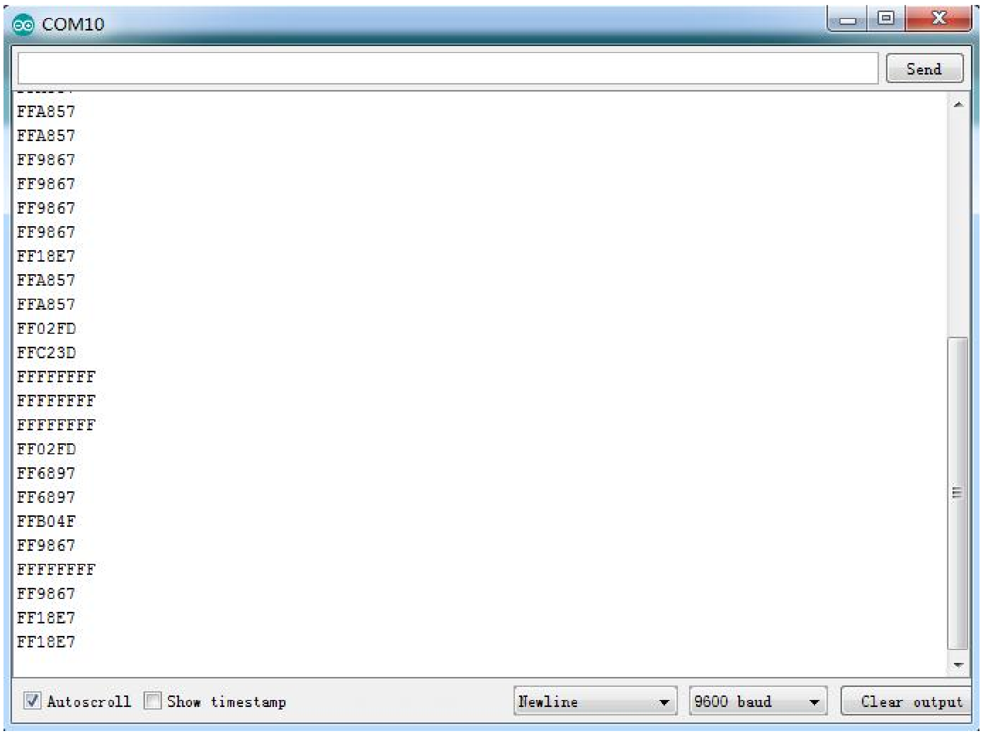
Pins of IR receiver module are G(GND) , V (VCC) and D3.

(4) Test Code

```
//*************************************************************************************
**
/*
  keyestudio smart turtle robot
 lesson 6.1
 IRremote
 http://www.keyestudio.com
*/
#include <IRremote.h>     //IRremote library statement
int RECV_PIN = 3;       //define the pins of IR receiver as D3
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);
 }
//*************************************************************************************
**
```

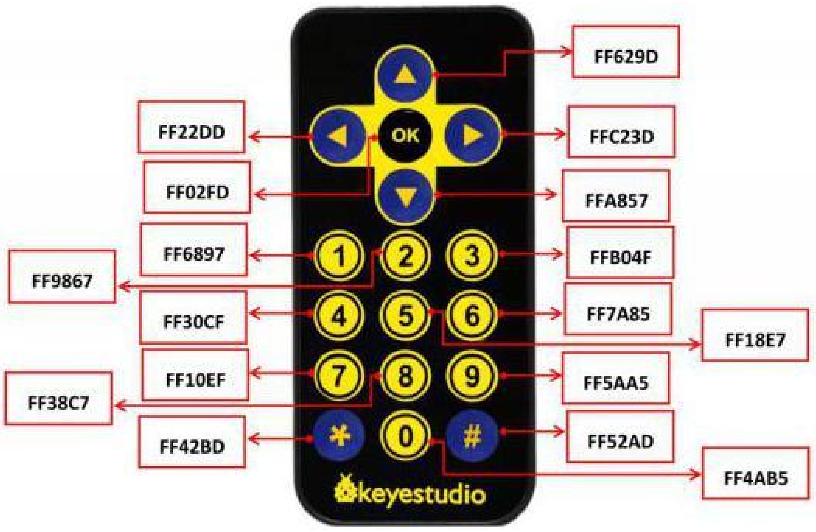
(5) Test Results:

Upload the test code. After uploading the code successfully, use the USB cable to connect the computer to supply power to the Keyestudio 4.0 development board. After powering on, open the serial monitor, set the baud rate to 9600, take out the remote control, and send the signal to the infrared receiving sensor. You can see the key value of the corresponding key, if the key time is too long, FFFFFFFF is prone to garbled characters.



```
FFA857
FFA857
FF9867
FF9867
FF9867
FF9867
FF18E7
FFA857
FFA857
FF02FD
FFC23D
FFFFFFFF
FFFFFFFF
FFFFFFFF
FF02FD
FF6897
FF6897
FFB04F
FF9867
FFFFFFFF
FF9867
FF18E7
FF18E7
```

The keys value of Keyestudio remote control are shown below.



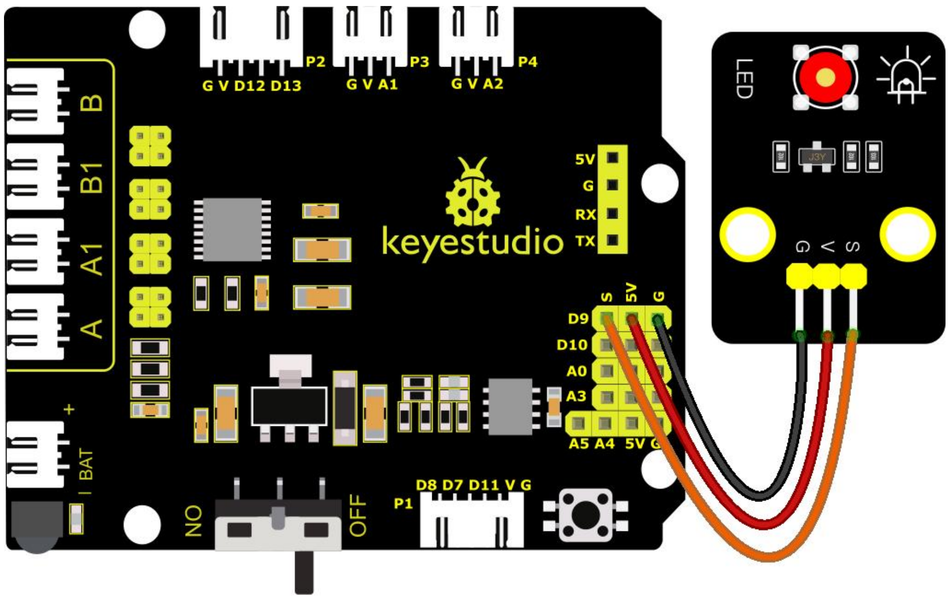
(6) Code Explanation

irrecv.enableIRIn(): after enabling IR decoding, the IR signals will be received, then function "decode()" will check continuously to make ure if decoding 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 continue to receive the next signal.

(7) Extension Practice:

We decoded the key value of IR remote control. How about controlling LED by the measured value? We could design an experiment.
Attach an LED to D9, then press the keys of remote control to make LED light on and off.



```
//**************************************************************************************
*
/*
keyestudio smart turtle robot
lesson 6.2
IRremote LED
http://www.keyestudio.com
*/
#include <IRremote.h>
int RECV_PIN = 3;//define the pin of IR receiver as D3
int LED_PIN = 9;//define the pin of LED as pin 9
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 9 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
    {
      Serial.println("HIGH");
      digitalWrite(LED_PIN,HIGH);//LED will be on
      a=1;
    }
    else if(results.value==0xFF02FD && (a==1)) //press again
    {
      Serial.println("LOW");
      digitalWrite(LED_PIN,LOW);//LED will go off
      a=0;
    }
    irrecv.resume(); // receive the next value
  }
}
//*************************************************************************************
*
```

Upload code to development board and power it up with a USB cable.

Press the OK key on remote control to make LED on and off.

Project 7: Bluetooth Remote Control

(1) Description

There is a DX-BT24 5.1 Bluetooth module in this kit. This bluetooth module comes with 256Kb space and complies with V5.1BLE bluetooth specification, which supports AT commands. Users can change parameters such as the baud rate and device name of the serial port as required.

Furthermore, it supports UART interface and bluetooth serial port transparent transmission, which also contains the advantages of low cost, small size, low power consumption and high sensitivity for sending and receiving. Notably, it solely needs a few peripheral components to realize its powerful functions.

(2) Parameters

- Bluetooth protocol: Bluetooth
- Specification V5.1 BLE

- Working distance: In an open environment, achieve 40m ultra-long distance communication Operating frequency: 2.4GHz ISM band
- Communication interface: UART
- Bluetooth certification: in line with FCC CE ROHS REACH certification standards
- Serial port parameters: 9600, 8 data bits, 1 stop bit, invalid bit, no flow control
- Power: 5V DC
- Operating temperature: –10 to +65 degrees Celsius

(3) Application:

The DX-BT24 module also supports the BT5.1 BLE protocol, which can be directly connected to iOS devices with BLE Bluetooth function, and supports resident running of background programs. Mainly used in the field of short-distance data wireless transmission. Avoid cumbersome cable connections and can directly replace serial cables. Successful application areas of BT24 modules:

※ Bluetooth wireless data transmission;

※ Mobile phone, computer peripheral equipment;

※ Handheld POS equipment;

※ Wireless data transmission of medical equipment;

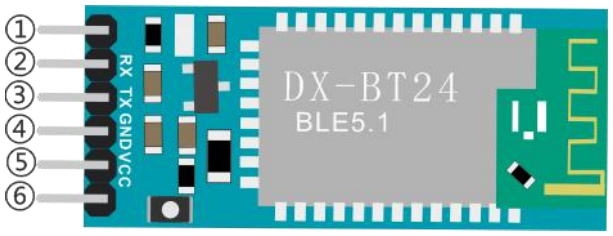
※ Smart home control;

※ Bluetooth printer;

※ Bluetooth remote control toys;

※ Shared bicycles;

(4) Ports



①STATE：Status pin

②RX：Receiving pin

③TX：sending pin

④GND：GND

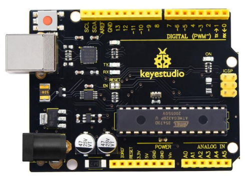
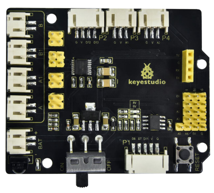
⑤VCC：Power

⑥EN：Enable pin

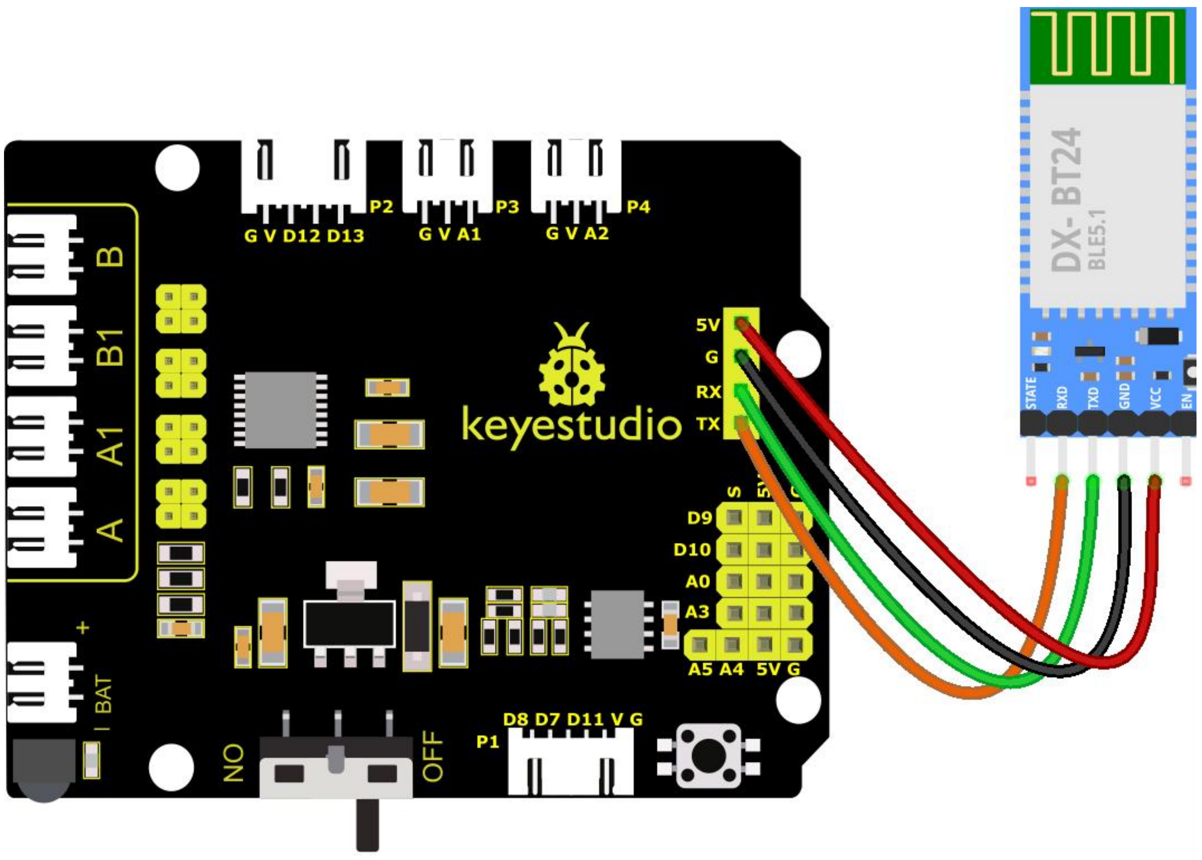
Connect the BT module to the development board.

Uno	BT24
TX	RX
RX	TX
VCC	5V
GND	GND

(5) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	Red LED Module*1
		
3P F-F Dupont Wire*1	USB cable*1	DX-BT24 BT Module*1

(6) Wiring Diagram



RXD, TXD, GND and VCC of the BT module are connected to TX, RX, G （GND) and 5V （VCC).

STATE and BRK of the BT module don’t need connection.

Note the direction of the BT module when inserting it onto the 8833 board. And don’t insert it before uploading the code.

(7) Test Code

```
//*************************************************************************
/*
keyestudio smart turtle robot
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
  }
}
//*************************************************************************
```

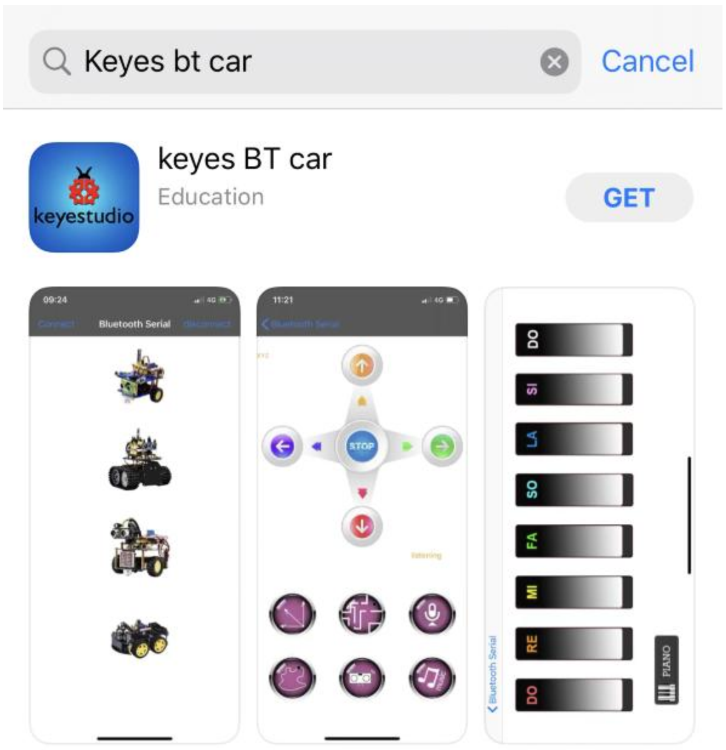
Don’ t connect the BT module when uploading the code because serial communication port will be occupied when uploading code and inserting the BT module.

Upload the code and power up the 4.0 board with a USB cable. Insert the BT module and download BT app.

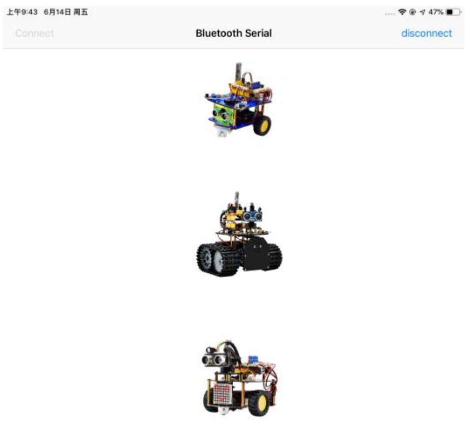
(8) Download app

For iOS system

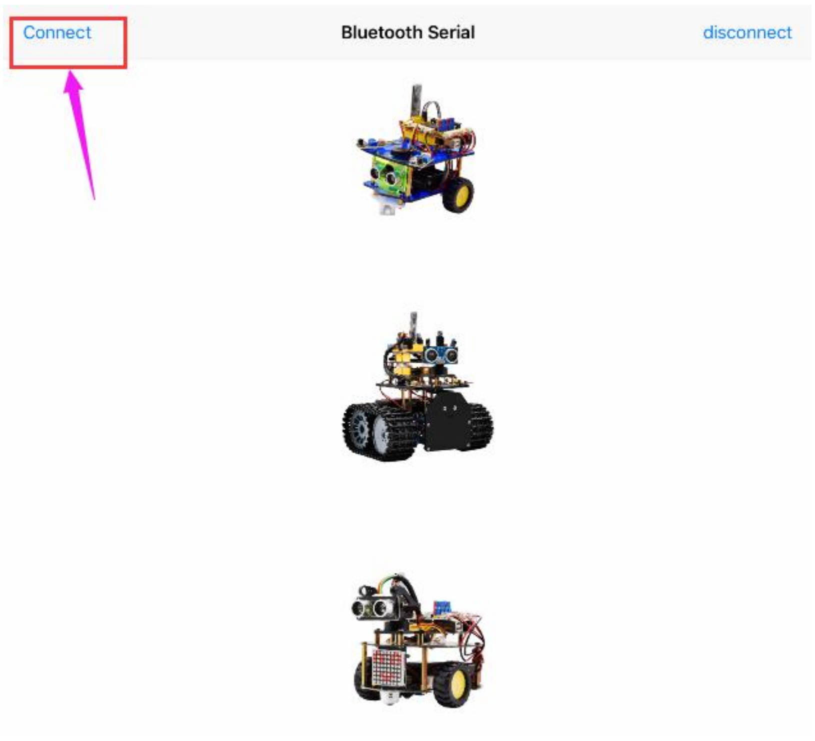
Search keyes BT car in App store



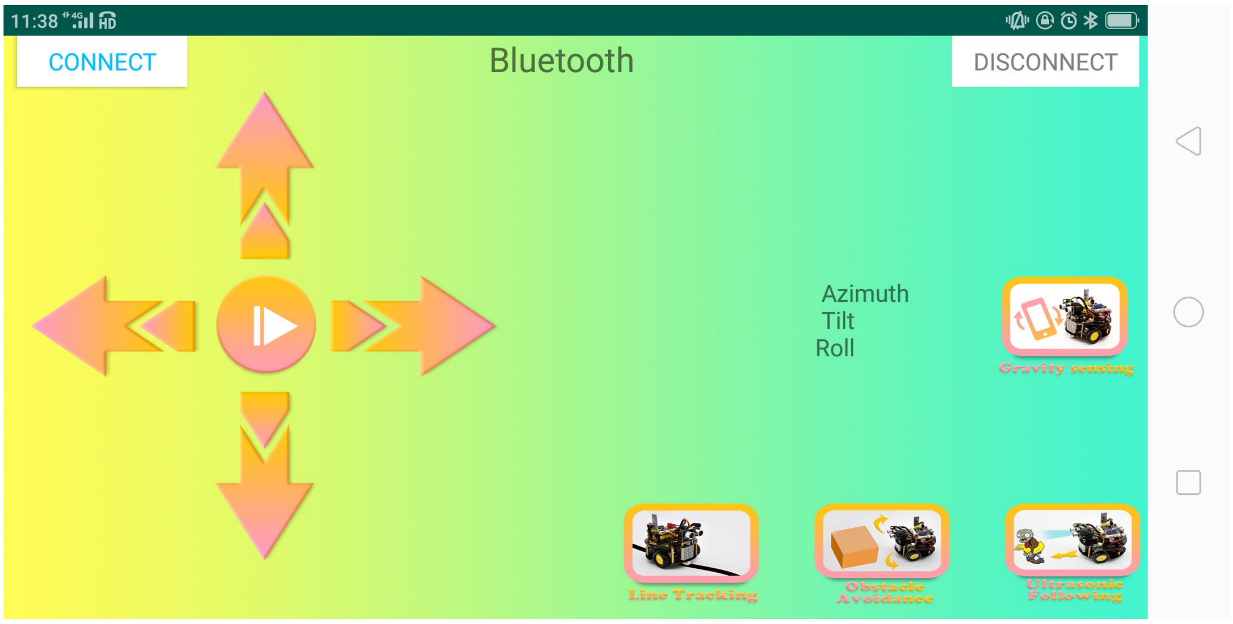
After installation, enter its interface.



Click “Connect” to search and pair Bluetooth.

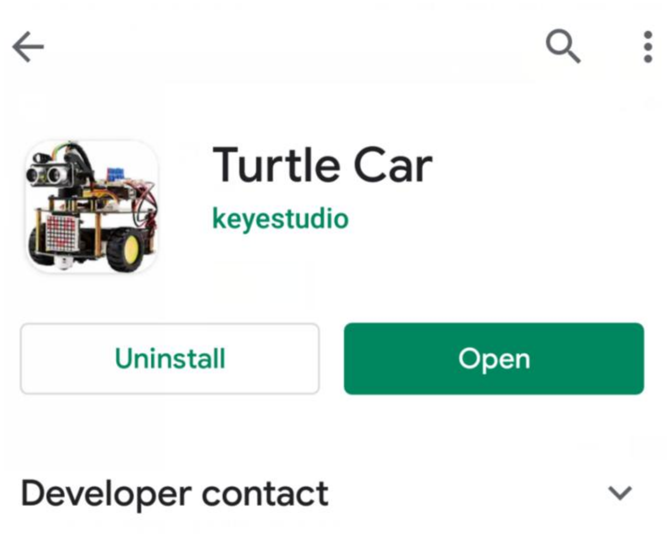


5.Click to enter the main page of turtle smart car.

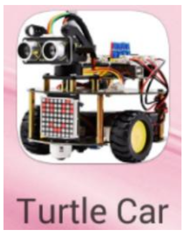


For Android System

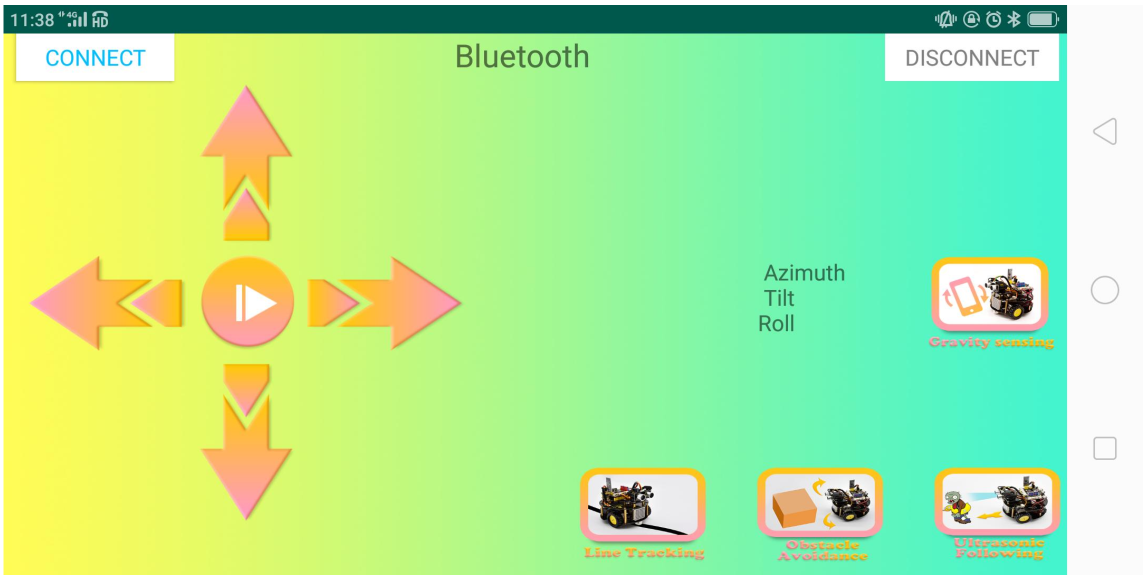
1. Enter Google play store to search for Turtle Car(**allow APP to access "location", you could enable "location"in settings of your cellphone.**



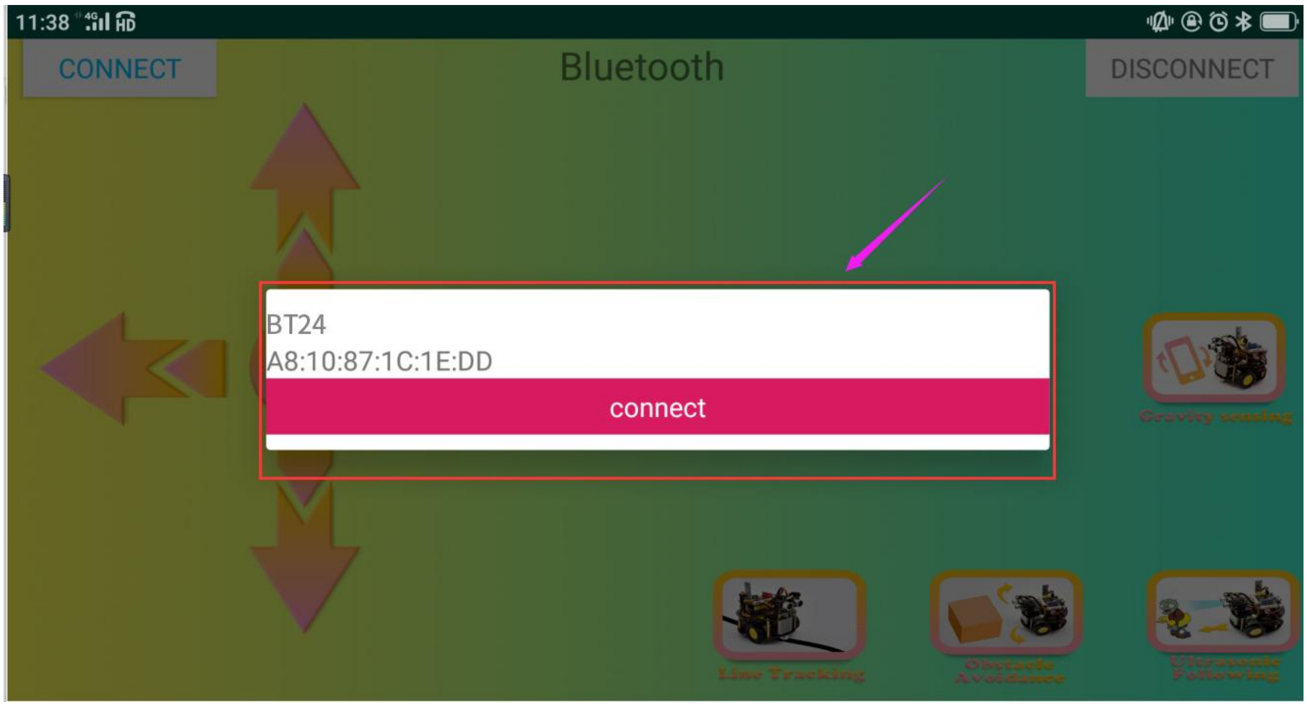
2. The app icon is shown below after installation.



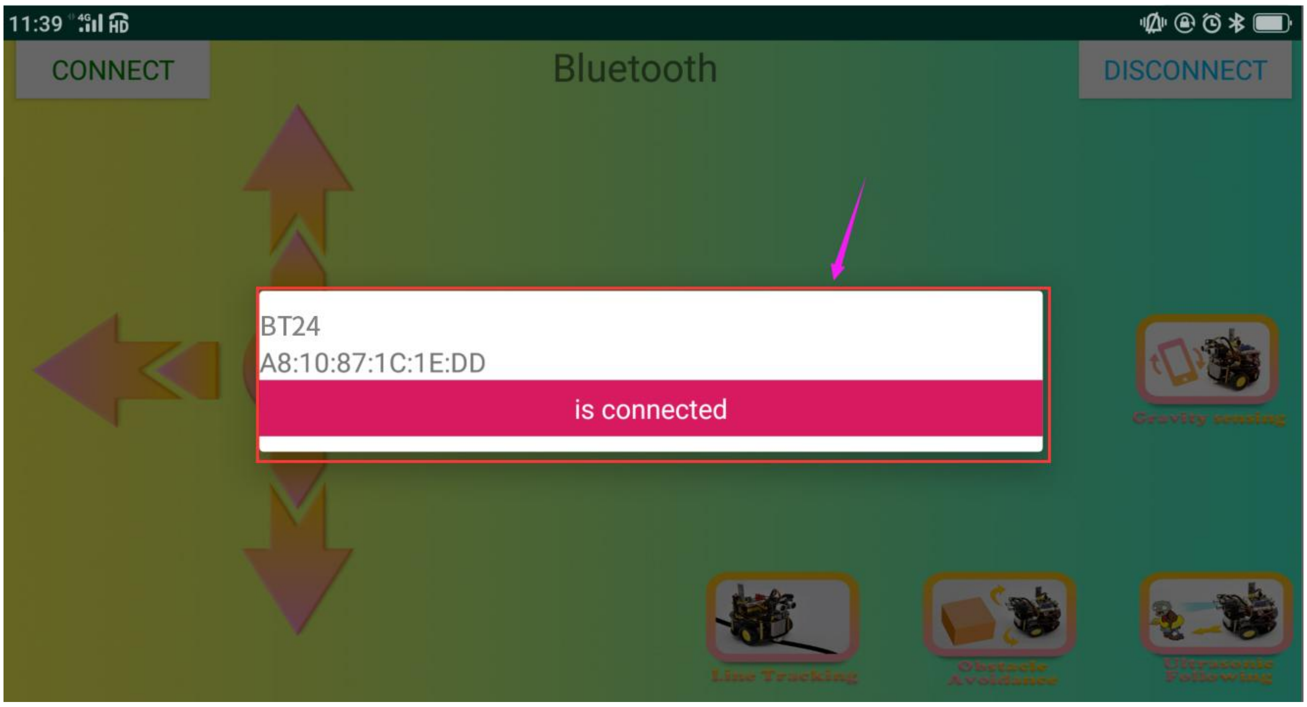
3. Click app to enter the following page.



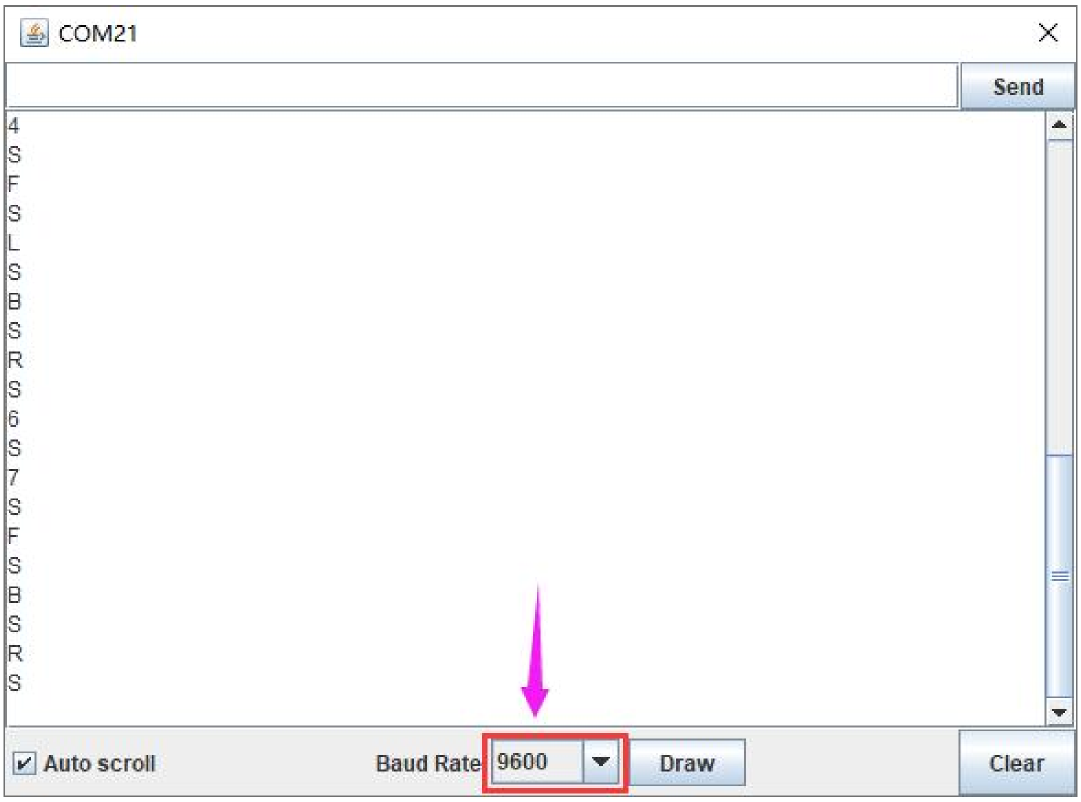
4. After connecting Bluetooth, plug in power and LED indicator of Bluetooth module will flicker. Tap CONNECT to search Bluetooth.



5. Click “connect” below HMSoft, then the Bluetooth will be connected and its LED indicator will stay on.

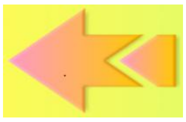
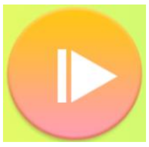
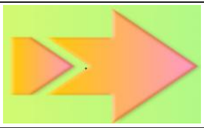


After connecting Bluetooth module, open serial monitor to set baud rate to 9600. Pressing the button of the Bluetooth APP, and the corresponding characters are displayed as shown below:



```
4
S
F
S
L
S
B
S
R
S
6
S
7
S
F
S
B
S
R
S
```

Baud Rate 9600

CONNECT	Pair DX-BT24 5.1 Bluetooth module	
Bluetooth	Enter control page of Bluetooth	
DISCONNECT	Disconnect Bluetooth	
	Press: F Release: S	Press the button, robot goes front; release to stop
	Press: L Release: S	Press the button, robot turns left; release to stop
	Click to send “S”	Stop
	Press: R	Press the button, robot turns

	Release: S	right; release to stop
⬇	Press: B Release: S	Press the button, robot goes back; release to stop
	Click to send “Y”	Start Ultrasonic follow function; click Stop to exit
	---	Click to start the mobile gravity sensing; click again to exit
	Click to send “U”	Start ultrasonic avoiding function; click Stop to exit
	Click to send "X"	Start line tracking function; click Stop to exit

We have read the characters of each key on mobile APP via serial port and know the function of those keys.

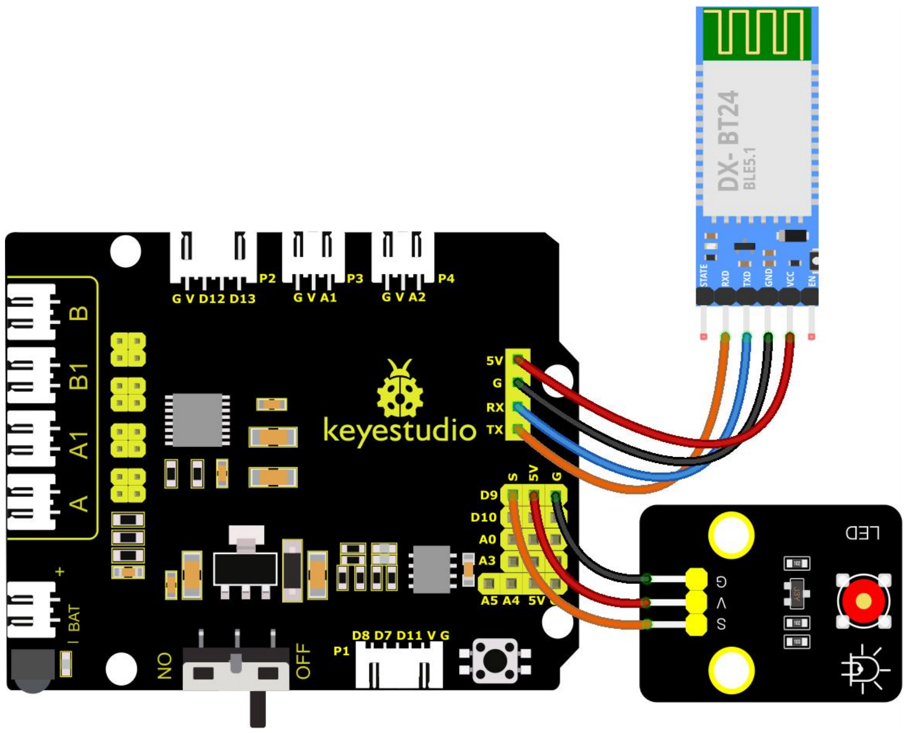
(8) Code Explanation:

Serial.available() : return the number of characters currently remaining in the serial port buffer. Generally, this function is used to judge whether there is data in the buffer of the serial port. When Serial.available()>0, it means that the serial port has received data and can be read;

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

(9) Project Expansion:

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



```
//*****************************************************************************
/*
 keyestudio smart turtle robot
 lesson 7.2
 Bluetooth LED
 http://www.keyestudio.com
*/
int ledpin=9;
```

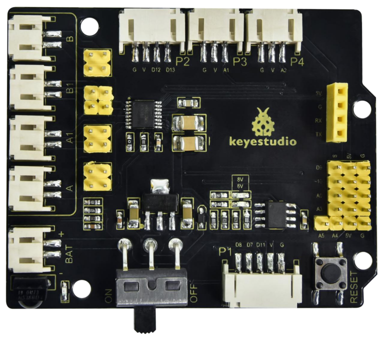
```
char ble_val;// An integer variable used to store the value received by Bluetooth

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

void loop()
{
  if (Serial.available() > 0) //Check whether there is data in the serial port cache
  {
    ble_val = Serial.read();  //Read data from the serial port cache
    Serial.print("DATA RECEIVED:");
    Serial.println(ble_val);
    if (ble_val == 'F') {
      digitalWrite(ledpin, HIGH);
      Serial.println("led on");
    }
    if (ble_val == 'B') {
      digitalWrite(ledpin, LOW);
      Serial.println("led off");
    }
   }
}
//**************************************************************************
```

Upload the code, connect to the 4.0 development board with a USB cable, power up and click

and to control the LED.

Project 8: Motor Driving and Speed Control

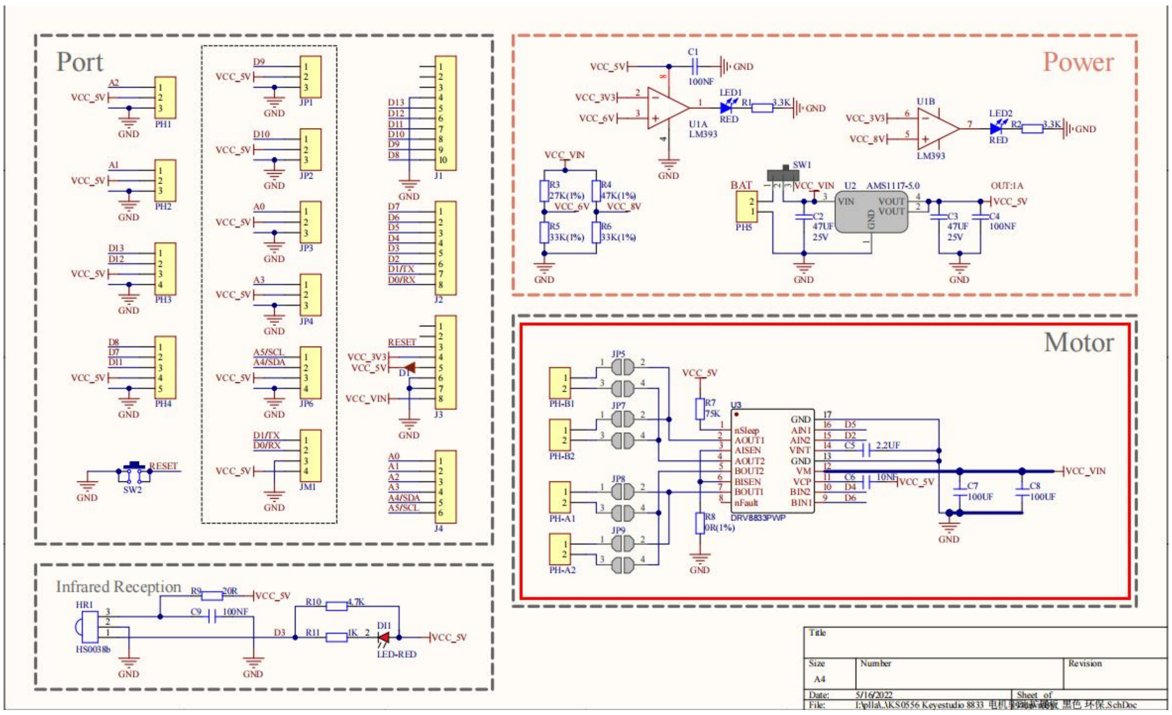
(1) Description

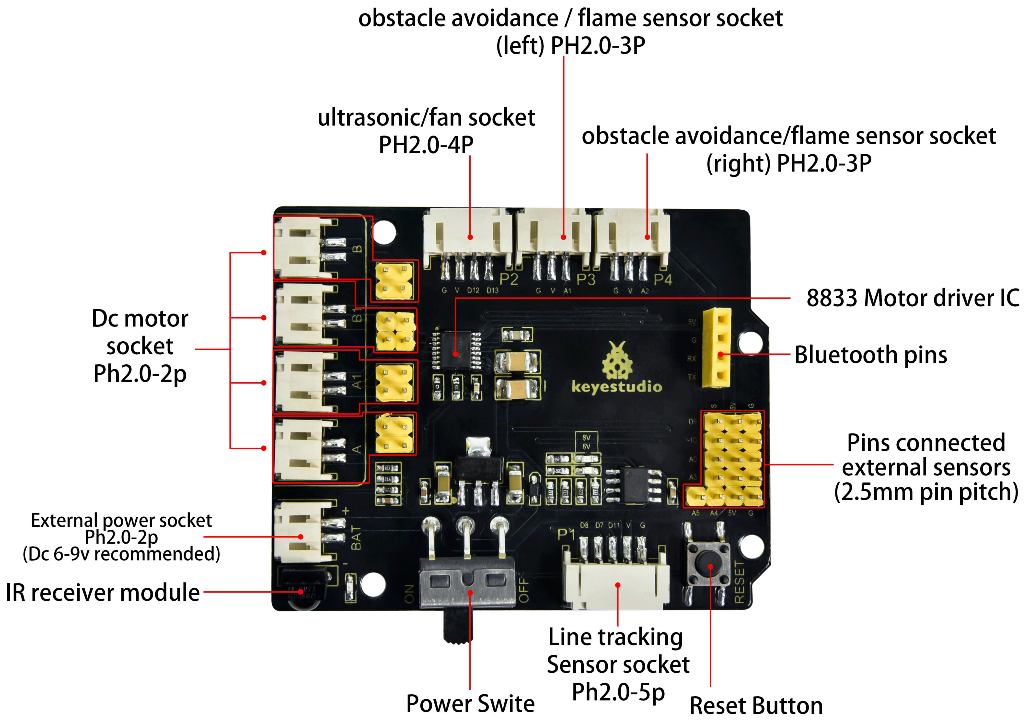


The 8833 motor driver expansion board uses PH2.0 terminals and an 8833 motor driver chip driven by the two-channel H bridge whose the largest current can be up to be 1.5A. It also integrates an IR receiver, ultrasonic sensor ports, analog ports, line tracking interfaces and pin headers for BT wifi and servo drivers

(2) Parameters

Keyestudio 8833 motor driver expansion board


Port
Power
Motor
Infrared Reception
DRV8833PWP
AMS1117-5.0
LM393
Title
Size
A4
Number
Revision
Date: 5/16/2022
Sheet of


obstacle avoidance / flame sensor socket
(left) PH2.0-3P
ultrasonic/fan socket
PH2.0-4P
obstacle avoidance/flame sensor socket
(right) PH2.0-3P
8833 Motor driver IC
Bluetooth pins
Pins connected
external sensors
(2.5mm pin pitch)
Dc motor
socket
Ph2.0-2p
External power socket
Ph2.0-2p
(Dc 6-9v recommended)
IR receiver module
Power Swite
Line tracking
Sensor socket
Ph2.0-5p
Reset Button

(3) Specification

Logic part input voltage: DC 5V

Input voltage of driving part: DC 6-9 V

Logic part operating current: <36mA

Operating current of driving part: <2A

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

Control signal input level: high level 2.3V<Vin<5V, low level -0.3V<Vin<1.5V

Working temperature: -25＋130℃

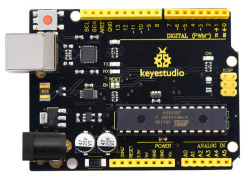
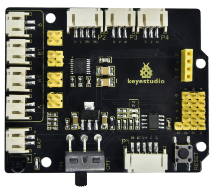
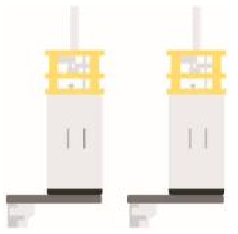
(3) Drive Robot to Move

From the above diagram, it is known that the direction pin of B motor is D2; a speed pin is D5; D4 is the direction pin of A motor; and D6 is speed pin.

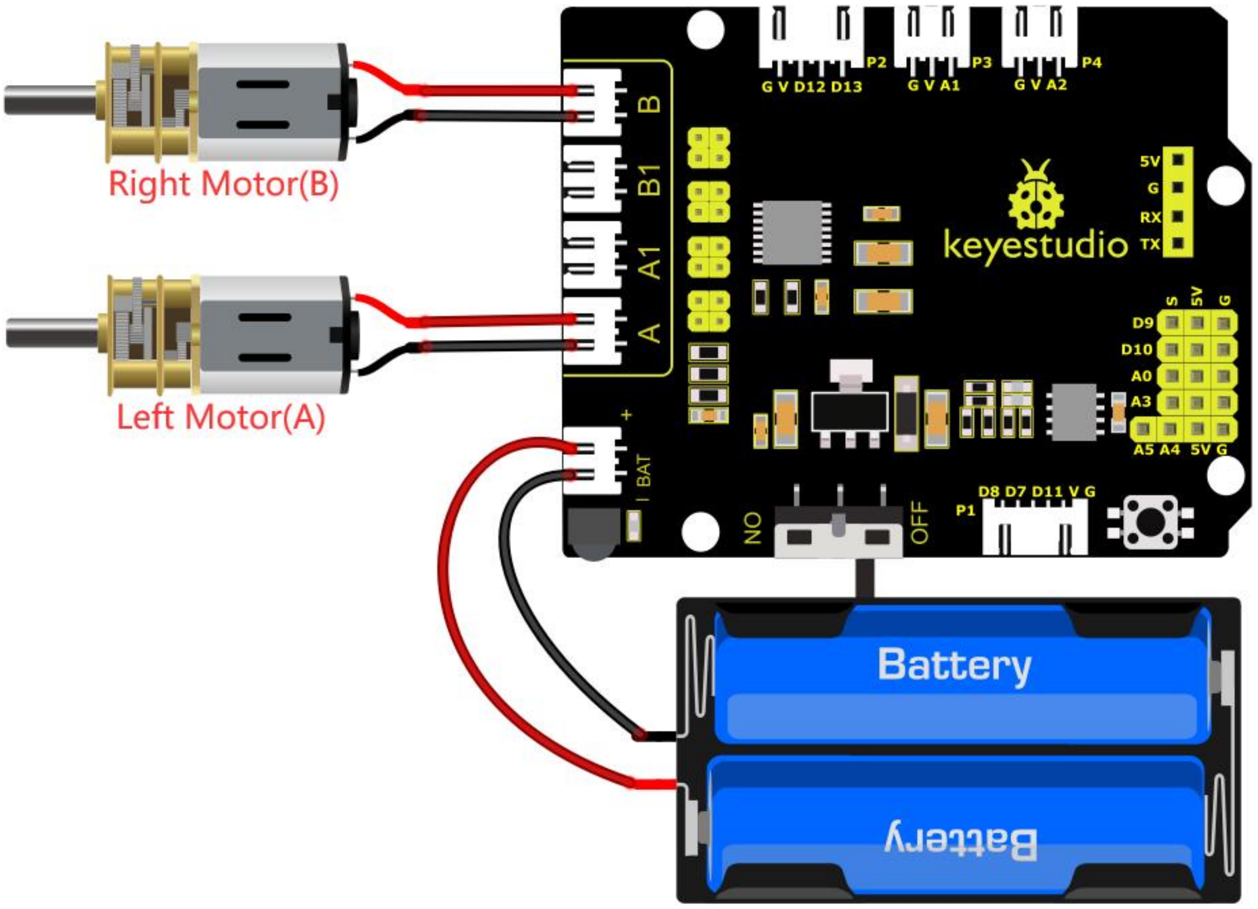
PWM drives the robot car. The PWM value is in the range of 0-255. The larger the number, the faster the rotation of the motor.

	D4	D6 (PWM)	Motor (L)	D2	D5 (PWM)	Motor (R)
Go forward	HIGH	255-200	Rotate clockwise	HIGH	255-200	Rotate clockwise
Go back	LOW	200	Rotate anticlockwise	LOW	200	Rotate anticlockwise
Rotate anticlockwise	LOW	200	Rotate anticlockwise	HIGH	255-200	Rotate clockwise
Rotate clockwise	HIGH	255-200	Rotate clockwise	LOW	200	Rotate anticlockwise

(4) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	Motor*1
		
18650 Battery Holder*1	2PDupont Wire*2	USB cable*1

(5) Wiring Diagram



(6) Test Code

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 8.1
 motor driver shield
 http://www.keyestudio.com
*/
#define ML_Ctrl 4     //define the direction control pin of A motor
#define ML_PWM 6   //define the PWM control pin of A motor
#define MR_Ctrl 2    //define the direction control pin of B motor
#define MR_PWM 5   //define the PWM control pin of B motor

void setup()
{
  pinMode(ML_Ctrl, OUTPUT);//set direction control pin of A motor to output
  pinMode(ML_PWM, OUTPUT);//set PWM control pin of A motor to output
  pinMode(MR_Ctrl, OUTPUT);//set direction control pin of B motor to output
  pinMode(MR_PWM, OUTPUT);//set PWM control pin of B motor to output
}
void loop()
{
  //front
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of A motor to HIGH
  analogWrite(ML_PWM,55);//set the PWM control speed of A motor to 55
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of B motor to HIGH
  analogWrite(MR_PWM,55);// set the PWM control speed of B motor to 55
  delay(2000);//delay in 2000ms
  //back
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of A motor to LOW level
  analogWrite(ML_PWM,200);// set the PWM control speed of A motor to 200
  digitalWrite(MR_Ctrl,LOW);//set the direction control pin of B motor to LOW level
  analogWrite(MR_PWM,200);//set the PWM control speed of B motor to 200
  delay(2000);//delay in 2000ms
  //left
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of A motor to LOW level
  analogWrite(ML_PWM,200);//set the PWM control speed of A motor to 200
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of B motor to HIGH level
  analogWrite(MR_PWM,55);//set the PWM control speed of B motor to 200
  delay(2000);//delay in 2000ms
  //right
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of A motor to HIGH level
  analogWrite(ML_PWM,55);//set the PWM control speed of A motor to 55
  digitalWrite(MR_Ctrl,LOW);// set the direction control pin of B motor to LOW level
  analogWrite(MR_PWM,200);//set the PWM control speed of B motor to 200
  delay(2000);//delay in 2000ms
  //stop
  digitalWrite(ML_Ctrl, LOW);// set the direction control pin of A motor to LOW level
  analogWrite(ML_PWM,0);//set the PWM control speed of A motor to 0
  digitalWrite(MR_Ctrl, LOW);// set the direction control pin of B motor to LOW level
  analogWrite(MR_PWM,0);//set the PWM control speed of B motor to 0
  delay(2000);// delay in 2000ms
}
//*****************************************************************************
```

(7) Test Result:

Upload code, power on the external power and turn the DIP switch to O. The turtle car will go for ward for 2s, back for 2s, turn left for 2s and right for 2s and stop for 2s

(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

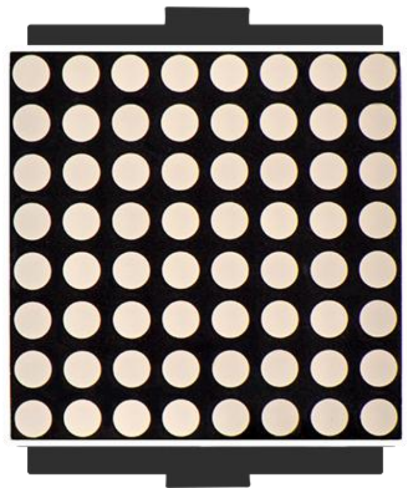
Adjust the speed that PWM controls the motor, hook up in the same way.

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 8.2
 motor driver
 http://www.keyestudio.com
*/
#define ML_Ctrl  4   //define the direction control pin of A motor
#define ML_PWM 6   //define the PWM control pin of A motor
#define MR_Ctrl  2   //define the direction control pin of B motor
#define MR_PWM 5   //define the PWM control pin of B motor

void setup()
{ pinMode(ML_Ctrl, OUTPUT);//set the direction control pin of A motor to OUTPUT
  pinMode(ML_PWM, OUTPUT);//set the PWM control pin of A motor to OUTPUT
  pinMode(MR_Ctrl, OUTPUT);//set the direction control pin of B motor to OUTPUT
  pinMode(MR_PWM, OUTPUT);//set the PWM control pin of B motor to OUTPUT
}

void loop()
{
 //front
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of A motor to HIGH
  analogWrite(ML_PWM,155);//set the PWM control speed of A motor to 155
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of B motor to HIGH
  analogWrite(MR_PWM,155);// set the PWM control speed of B motor to 155
  delay(2000);//delay in 2000ms
  //back
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of A motor to LOW level
  analogWrite(ML_PWM,100);// set the PWM control speed of A motor to 100
  digitalWrite(MR_Ctrl,LOW);//set the direction control pin of B motor to LOW level
  analogWrite(MR_PWM,100);//set the PWM control speed of B motor to 100
  delay(2000);//delay in 2000ms
  //left
  digitalWrite(ML_Ctrl,LOW);//set the direction control pin of A motor to LOW level
  analogWrite(ML_PWM,100);//set the PWM control speed of A motor to 100
  digitalWrite(MR_Ctrl,HIGH);//set the direction control pin of B motor to HIGH level
  analogWrite(MR_PWM,155);//set the PWM control speed of B motor to 155
  delay(2000);//delay in 2000ms
  //right
  digitalWrite(ML_Ctrl,HIGH);//set the direction control pin of A motor to HIGH level
  analogWrite(ML_PWM,155);//set the PWM control speed of A motor to 155
  digitalWrite(MR_Ctrl,LOW);// set the direction control pin of B motor to LOW level
  analogWrite(MR_PWM,100);//set the PWM control speed of B motor to 100
  delay(2000);//delay in 2000ms
  //stop
  digitalWrite(ML_Ctrl, LOW);// set the direction control pin of A motor to LOW level
  analogWrite(ML_PWM,0);//set the PWM control speed of A motor to 0
  digitalWrite(MR_Ctrl, LOW);// set the direction control pin of B motor to LOW level
  analogWrite(MR_PWM,0);//set the PWM control speed of B motor to 0
  delay(2000);// delay in 2000ms
}
//*************************************************************************
```

Upload code, power on the external power and turn the DIP switch to ON, do you find the motors rotate slower?

Note: low current will cause that the motor rotates slowly.

Project 9: 8*8 Dot Matrix

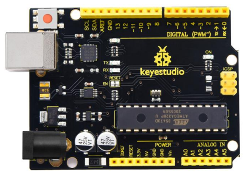
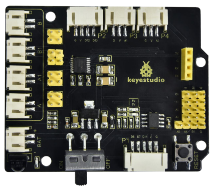
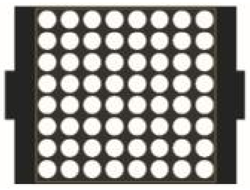


(1) Description

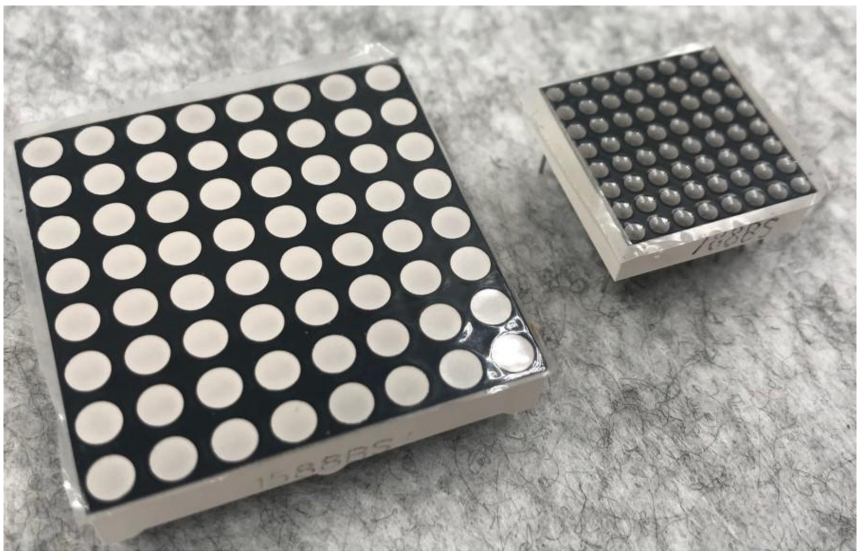
A fun way to make a small display is to use an 8x8 matrix or a 4-digit 7-segment display. Matrices like these are 'multiplexed' - to control 64 LEDs you need 16 pins. That's a lot of pins, and there are driver chips like the MAX7219 that can control a matrix for you. But there's a lot of wiring to set up and they take up a ton of space. After all, wouldn't it be awesome if you could control a matrix without tons of wiring?

We control and drive an 8*8 LED board by the HT16K33 chip, which is convenient for wiring and greatly save the resources of microcontroller.

(2) Components

Keyestudio 4.0 development board *1	Keyestudio 8833 motor driver expansion board *1	8*8 Dot Matrix Module*1
		
USB cable*1	F-F Dupont Wire*4	

(3) 8*8 Dot Matrix



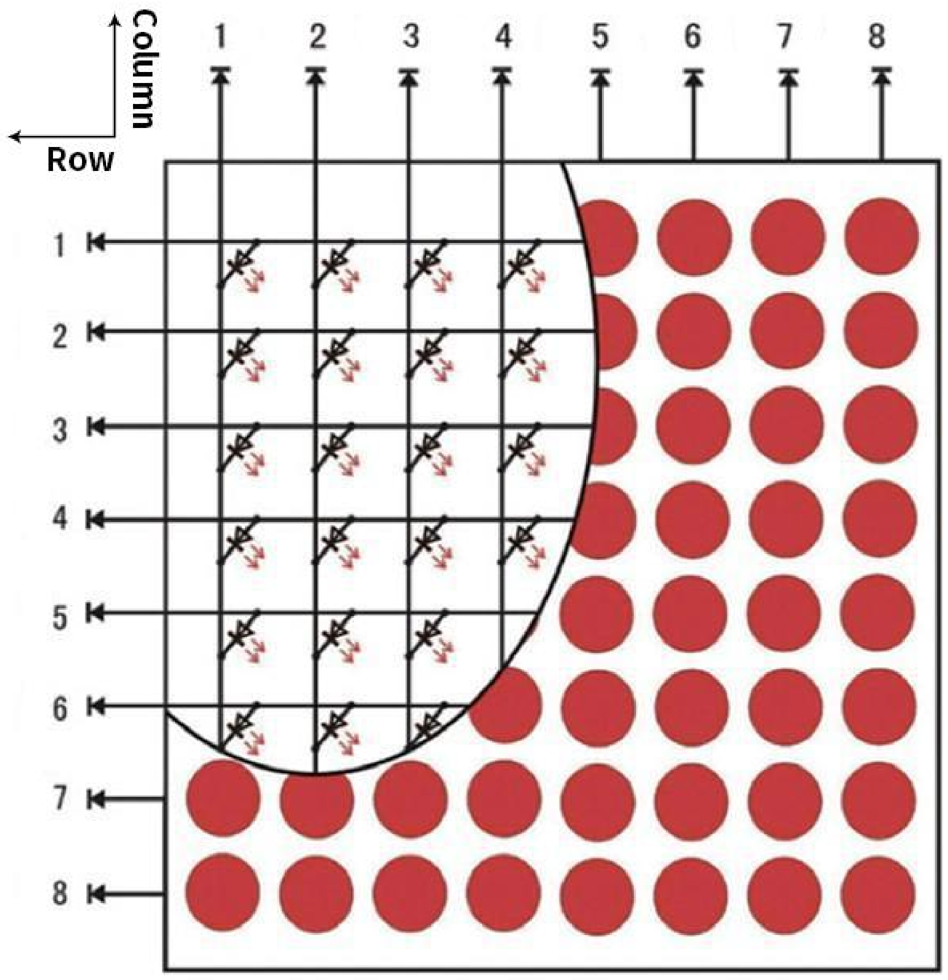
Composed of LED emitting tube diodes, LED dot matrix are applied widely to public information display like advertisement screen and bulletin board, by controlling LED to show words, pictures and videos, etc.

Divided into single-color, double-color, and three-color lights according to emitting color , LED dot matrix could show red, yellow, green and even true color.

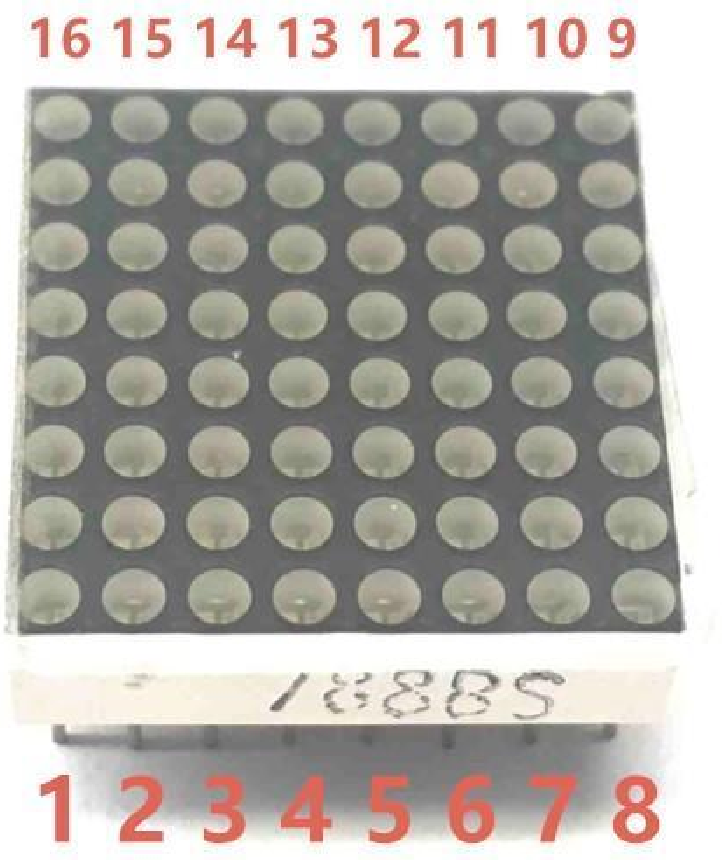
There are different types of matrices, including 4×4, 8×8 and 16×16 and etc.

The 8×8 dot matrix contains 64 LEDs.

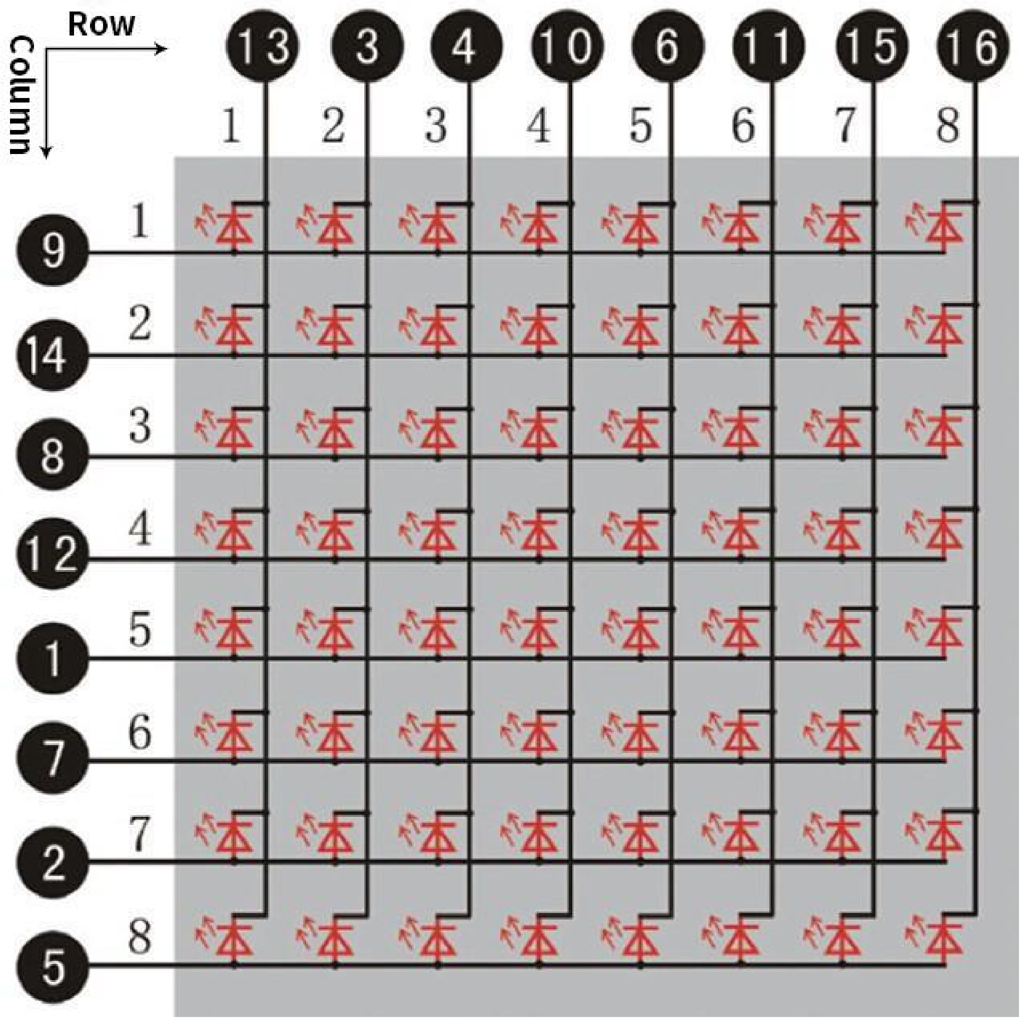
The inner structure of 8×8 dot matrix is shown below.



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



The definition inner pins are shown below:

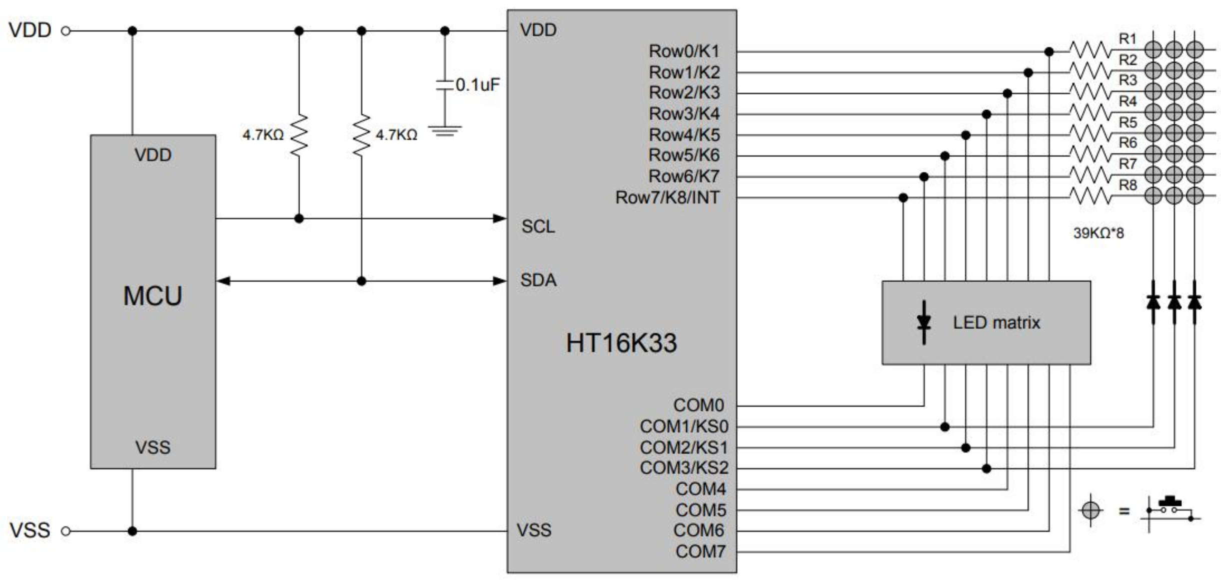


For instance, to light up the LED on row 1 and column 1, you should increase the voltage of pin 9 and reduce the voltage of pin 13.

(4) HT16K33 8X8 Dot Matrix

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

The picture below is the working schematic of HT16K33 chip



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

(5) Specification of 8*8 dot matrix

Input voltage: 5V

Rated input frequency: 400KHZ

Input power: 2.5W

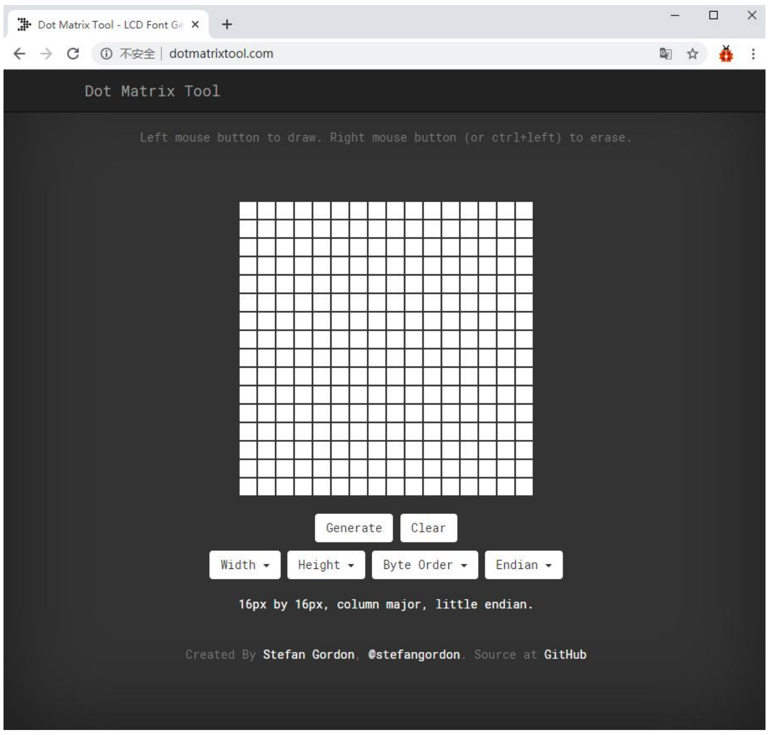
Input current: 500mA

4. Introduction for Modulus Tool

The online version of dot matrix modulus tool:

http://dotmatrixtool.com/#

① Open the link to enter the following page.

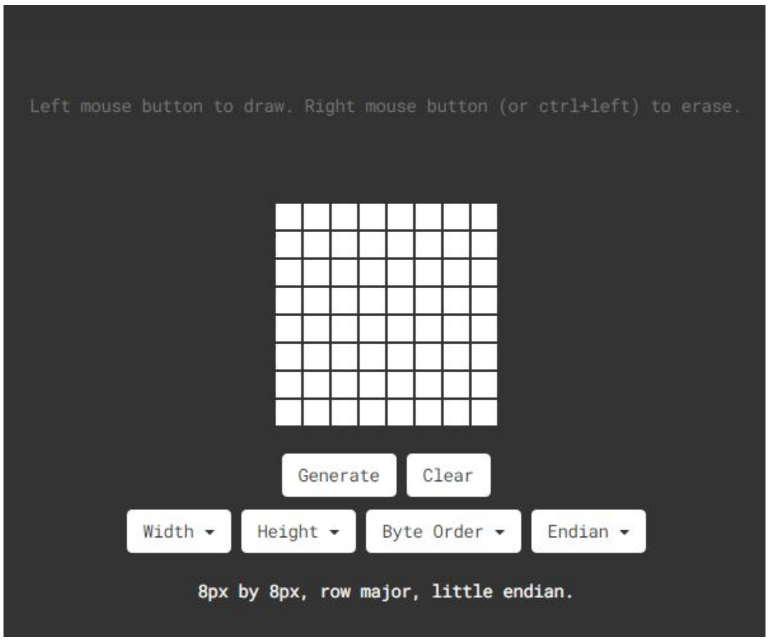


Left mouse button to draw. Right mouse button (or ctrl+left) to erase.

16px by 16px, column major, little endian.

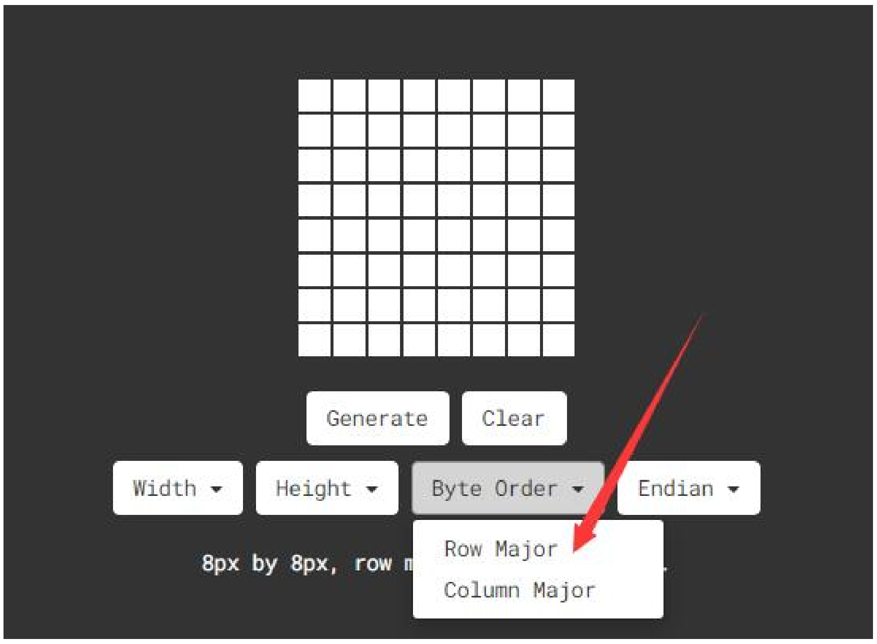
Created By Stefan Gordon, @stefangordon. Source at GitHub

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


Left mouse button to draw. Right mouse button (or ctrl+left) to erase.
Generate
Clear
Width
Height
Byte Order
Endian

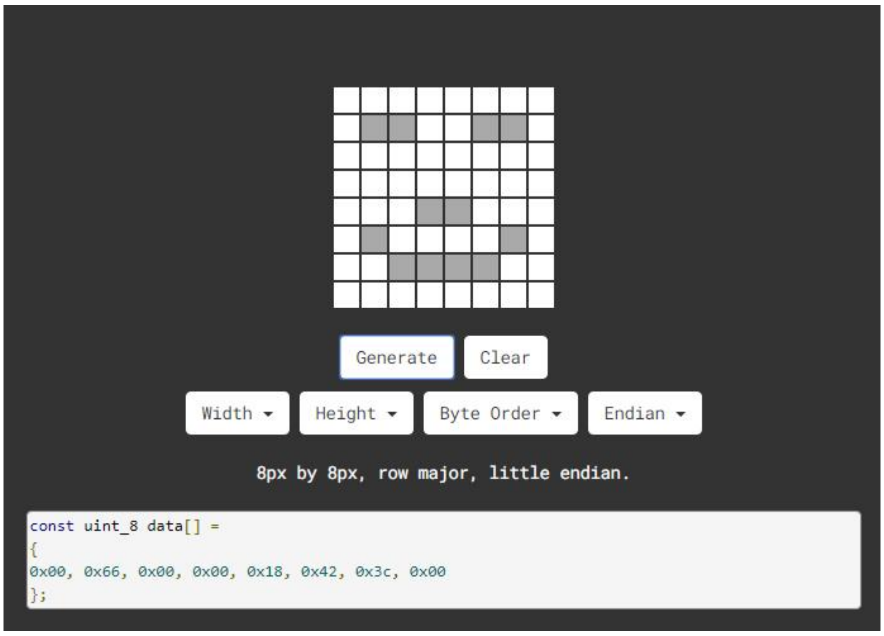
8px by 8px, row major, little endian.

Click Byte order to select “**Row major**”



③ Generate hexadecimal data from the pattern

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

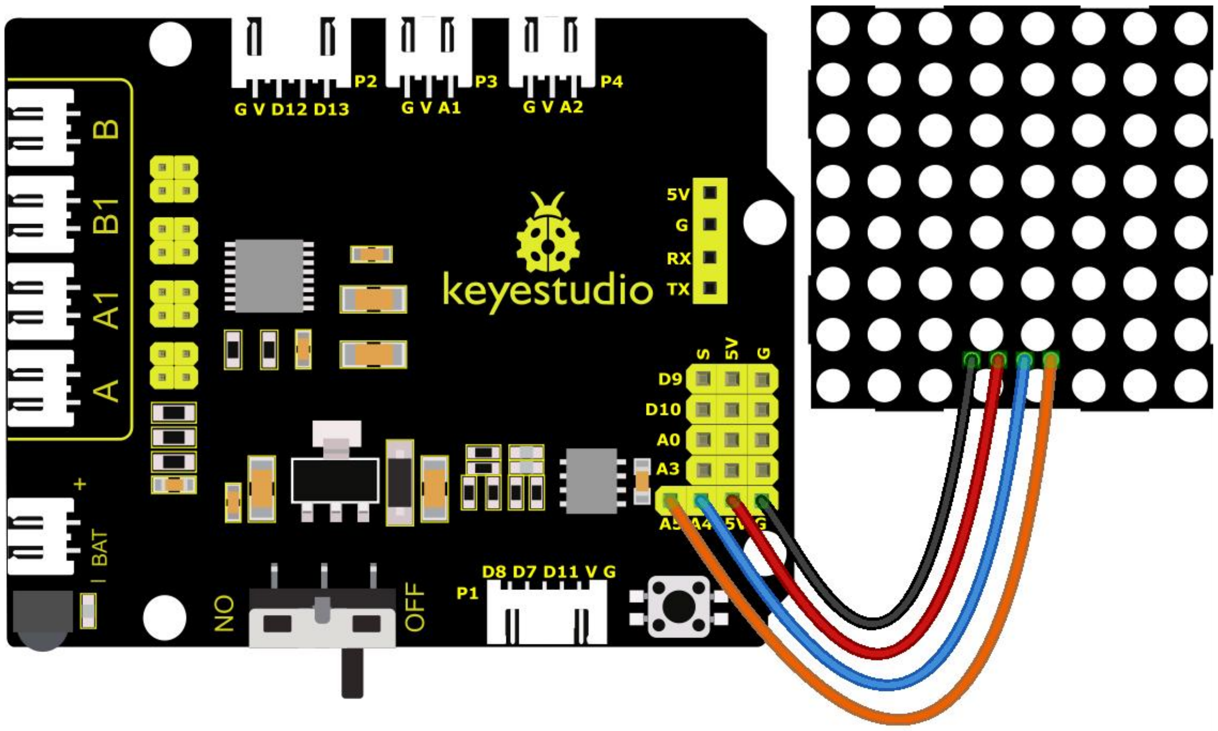


8px by 8px, row major, little endian.

```
const uint_8 data[] =
{
0x00, 0x66, 0x00, 0x00, 0x18, 0x42, 0x3c, 0x00
};
```

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

(6) Connection Diagram



GND, VCC, SDA and SCL of the 8*8 dot matrix module are connected to G (GND), V (VCC), A4 and A5 of the expansion board.

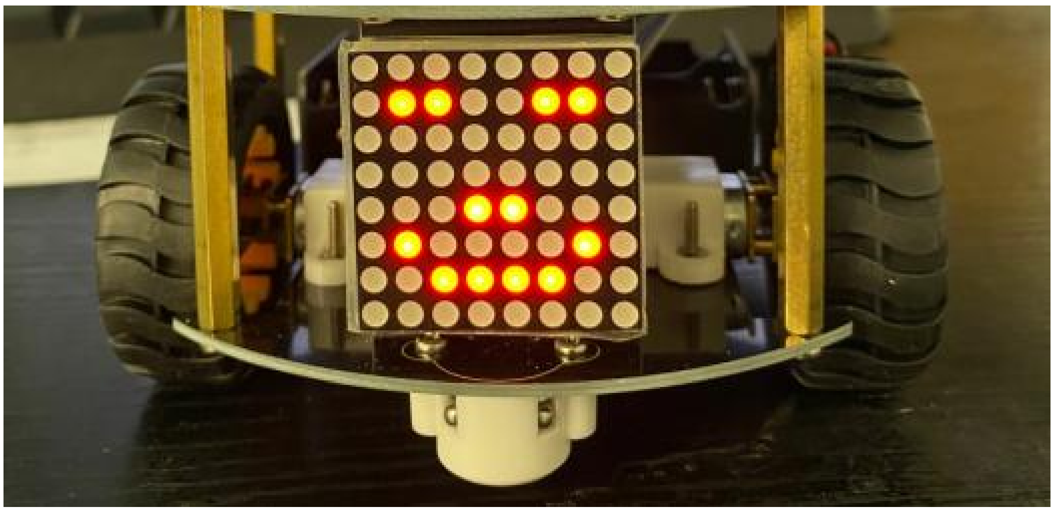
(7) Test Code

```
//**********************************************************************************
/*
 keyestudio smart turtle robot
 lesson 9.1
 Matrix
 http://www.keyestudio.com
*/
#include <Matrix.h>
Matrix myMatrix(A4,A5);   //set pins to communication pins
// define an array
uint8_t LedArray1[8]={0x00, 0x66, 0x00, 0x00, 0x18, 0x42, 0x3c, 0x00};
uint8_t  LEDArray[8]; //define an array(by modulus tool) without initial value

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

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

```
    }
  }
  myMatrix.writeDisplay();  // dot matrix shows
}
//*************************************************************************************
```

(8) Test Result

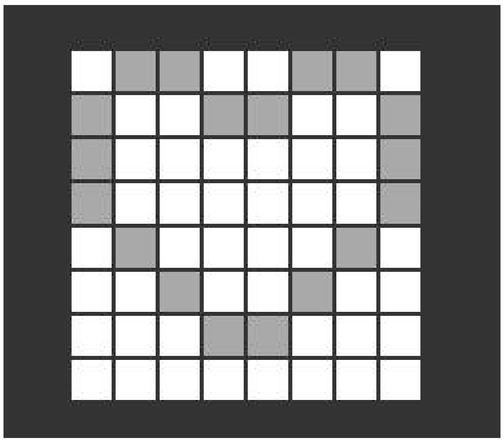
When uploading the code, power up the 4.0 board and turning on the robot car, the 8*8 dot matrix shows a smile facial pattern.



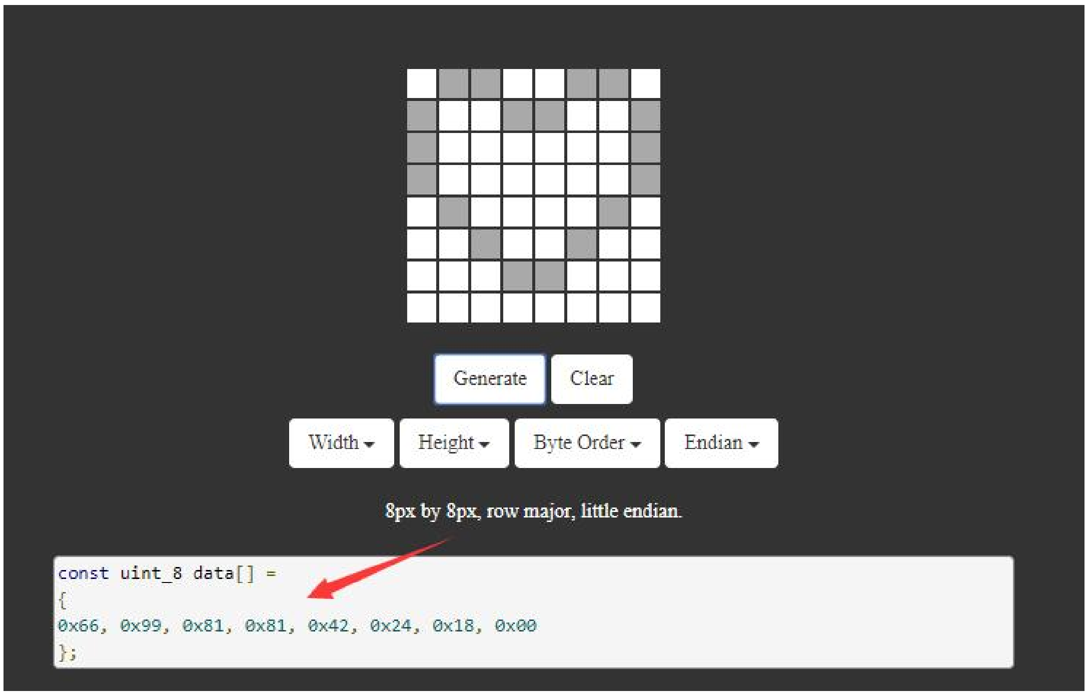
(9) Extension Practice:

Let’ s make dot matrix draw a heart-shaped pattern. What you need to do is entering the website and drawing the following pattern.

http://dotmatrixtool.com/#


,

Then we get the code of drawing the heart-shaped pattern.



8px by 8px, row major, little endian.

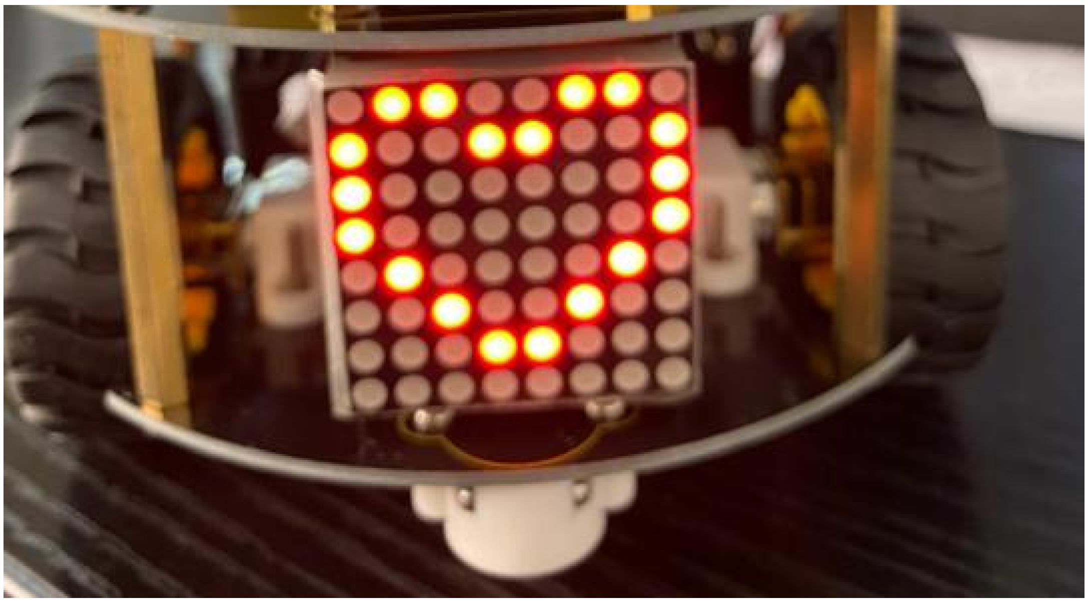
```
const uint_8 data[] =
{
0x66, 0x99, 0x81, 0x81, 0x42, 0x24, 0x18, 0x00
};
```

Replace the above code of heart-shaped pattern, then the complete code is shown below:

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 9.2
 Matrix
 http://www.keyestudio.com
*/
#include <Matrix.h>
```

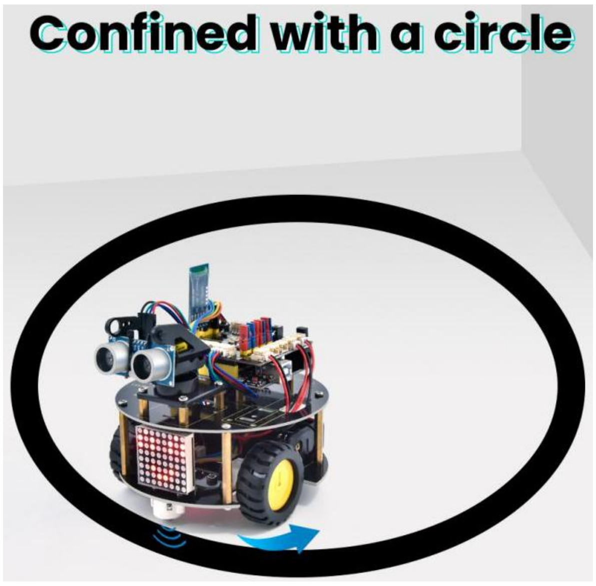
```
Matrix myMatrix(A4,A5);   //set pins to communication pins
//define an array
uint8_t LedArray1[8]={0x66,0x99,0x81,0x81,0x42,0x24,0x18,0x00};
uint8_t  LEDArray[8]; //define an array(by modulus tool) without initial value
void setup(){
  myMatrix.begin(0x70);  //communication address
  myMatrix.clear();    //Clear
}
void loop(){
  for(int i=0; i<8; i++)  // there is eight data, loop for eight times
  {
    LEDArray[i]=LedArray1[i];  //Call the emoticon array data in the subroutine LEDArray
    for(int j=7; j>=0; j--)  //Every data(byte) has 8 bits, therefore, loop for eight times

    {
      if((LEDArray[i]&0x01)>0) //judge if the last bit of data is greater than 0
      {
        myMatrix.drawPixel(j, i,1);  //light up the corresponding point
      }
      else  //otherwise
      {
        myMatrix.drawPixel(j, i,0);  //turn off the corresponding point
      }
      LEDArray[i] = LEDArray[i]>>1;  //LEDArray[i] moves right for one bit to judge the
previous one bit
    }
  }
  myMatrix.writeDisplay();  // dot matrix shows
}
//***************************************************************************
```

Upload the code, power up the 4.0 board with a USB cable. The 8X8 module will show the heartbeat pattern.



Project 10: Move-in-Confined-Space Tank

(1)Description:

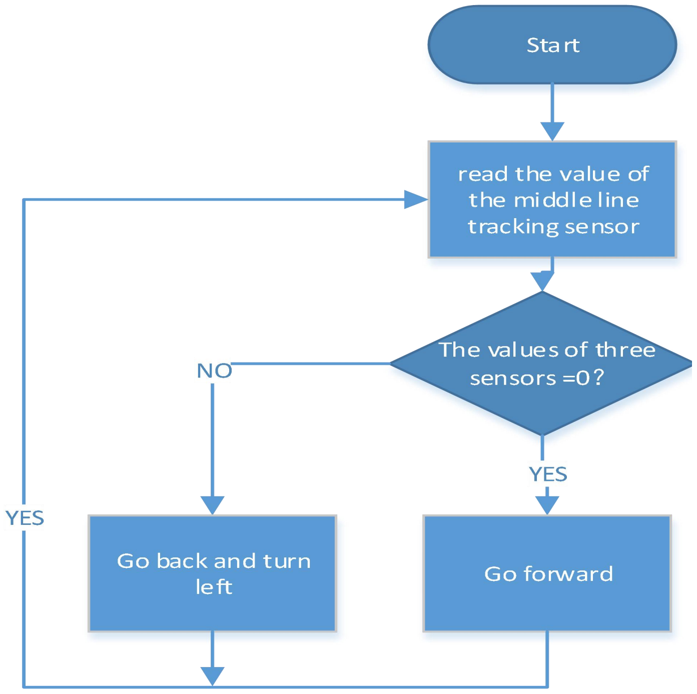


The ultrasonic sound-following and obstacle avoidance functions of the smart car have been introduced in previous projects. Here we intend to combine the knowledge in the previous courses to confine the smart car to move in a certain space. In the experiment, we use the line-tracking sensor to detect whether there is a black line around the smart car, and then control the rotation of the two motors according to the detection results, so as to lock the smart car in a circle drawn in black line.

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

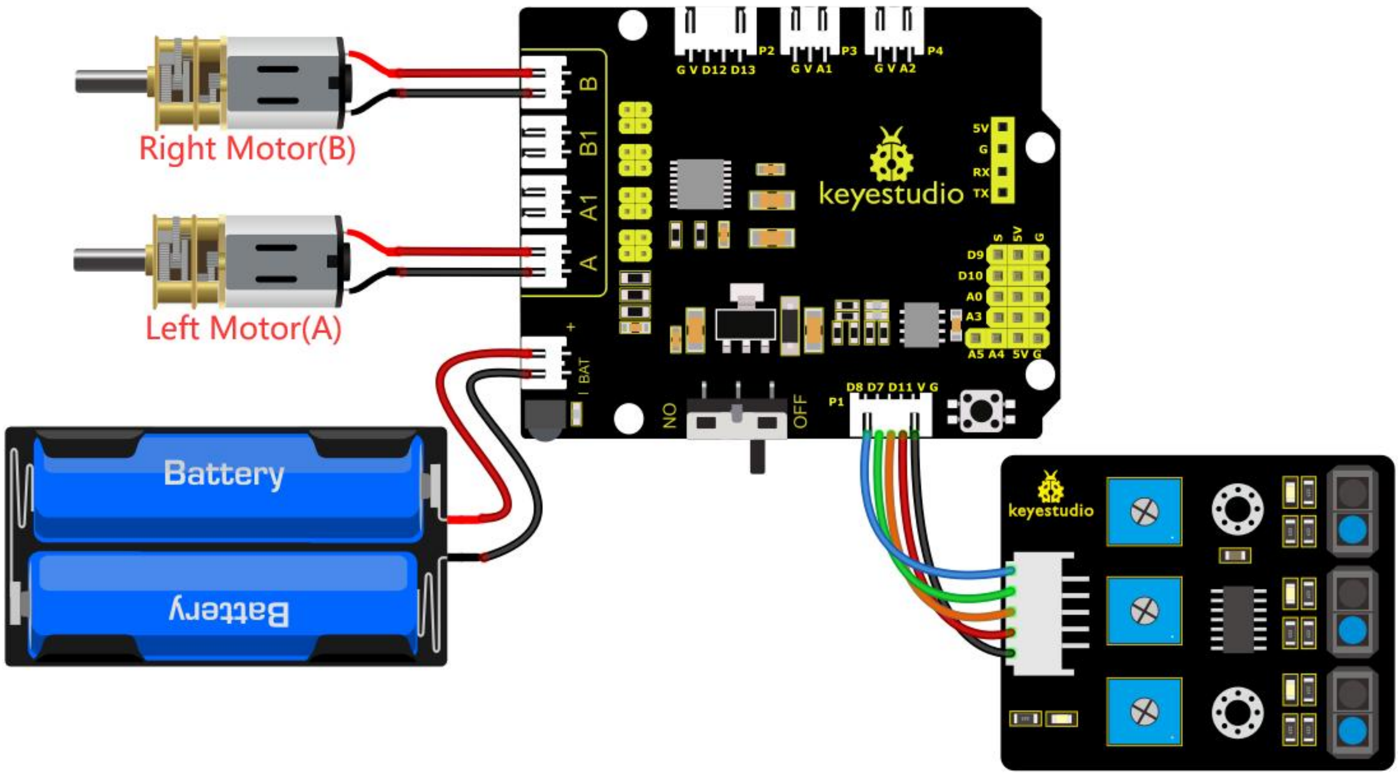
Detection	Line-tracking sensor in the middle	Black line detected: in high level
		White line detected: in low level
	Line-tracking sensor on the left	Black line detected: in high level
		White line detected: in low level
	Line-tracking sensor on the right	Black line detected: in high level
		White line detected: in low level

Condition	Movement
All the three line-tracking sensors detect no black lines	Move forward
Any of the three line-tracking sensors detects black lines	Step back Then rotate left



Start
read the value of the middle line tracking sensor
The values of three sensors =0?
NO
YES
Go back and turn left
Go forward
YES

(2) Wiring Diagram



G, V, S1, S2 and S3 of the line tracking sensor are connected to G (GND), V (VCC), D11, D7 and D8 of the sensor expansion board.

The power is connected to the BAT port

(3) Test Code

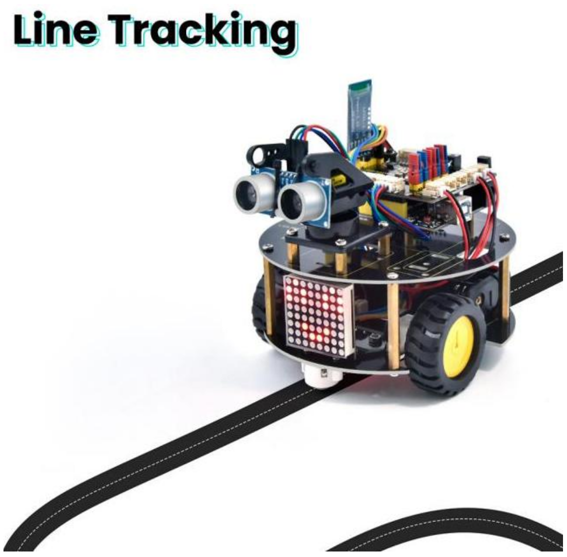
```
//*****************************************************************************
/*
 keyestudio smart turtle robot
 lesson 10
 Move in Confined Space turtle
 http://www.keyestudio.com
*/
int left_ctrl = 4;//define direction control pin of A motor
 int left_pwm = 6;//define PWM control pin of A motor
 int right_ctrl = 2;//define direction control pin of B motor
 int right_pwm = 5;//define PWM control pin of B motor
 int sensor_L = 11;//define the pin of left line tracking sensor
 int sensor_M = 7;//define the pin of middle line tracking sensor
 int sensor_R = 8;//define the pin of right line tracking sensor
int L_val,M_val,R_val;//define these variables

void setup() {
  Serial.begin(9600);//start serial monitor and set baud rate to 9600
  pinMode(left_ctrl,OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(left_pwm,OUTPUT);//set PWM control pin of A motor to OUTPUT
  pinMode(right_ctrl,OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(right_pwm,OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(sensor_L,INPUT);//set the pins of left line tracking sensor to INPUT
  pinMode(sensor_M,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
  M_val = digitalRead(sensor_M);//read the value of middle line tracking sensor
  R_val = digitalRead(sensor_R);//read the value of right line tracking sensor
  if ( L_val == 0 && M_val == 0 && R_val == 0 ) { //when no black lines are detected, turtle car
forward
    Car_front();
  }
  else { //Otherwise, if any of the patrol sensors detect a black line, back up and turn left
    Car_back();
    delay(700);
    Car_left();
    delay(800);
  }
}

void Car_front()
{
  digitalWrite(left_ctrl, HIGH);
  analogWrite(left_pwm, 100);
  digitalWrite(right_ctrl, HIGH);
  analogWrite(right_pwm, 100);
}
void Car_back()
{
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm, 155);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm, 155);
}
void Car_left()
{
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm, 155);
  digitalWrite(right_ctrl, HIGH);
  analogWrite(right_pwm, 100);
}
void Car_right()
{
  digitalWrite(left_ctrl, HIGH);
  analogWrite(left_pwm, 100);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm, 155);
}
void Car_Stop()
{
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm, 0);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm, 0);

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

(4) Test Result

Upload the code to the development board, power up and turn the DIP switch to ON. The turtle car will move in the circle.

Project 11: 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.

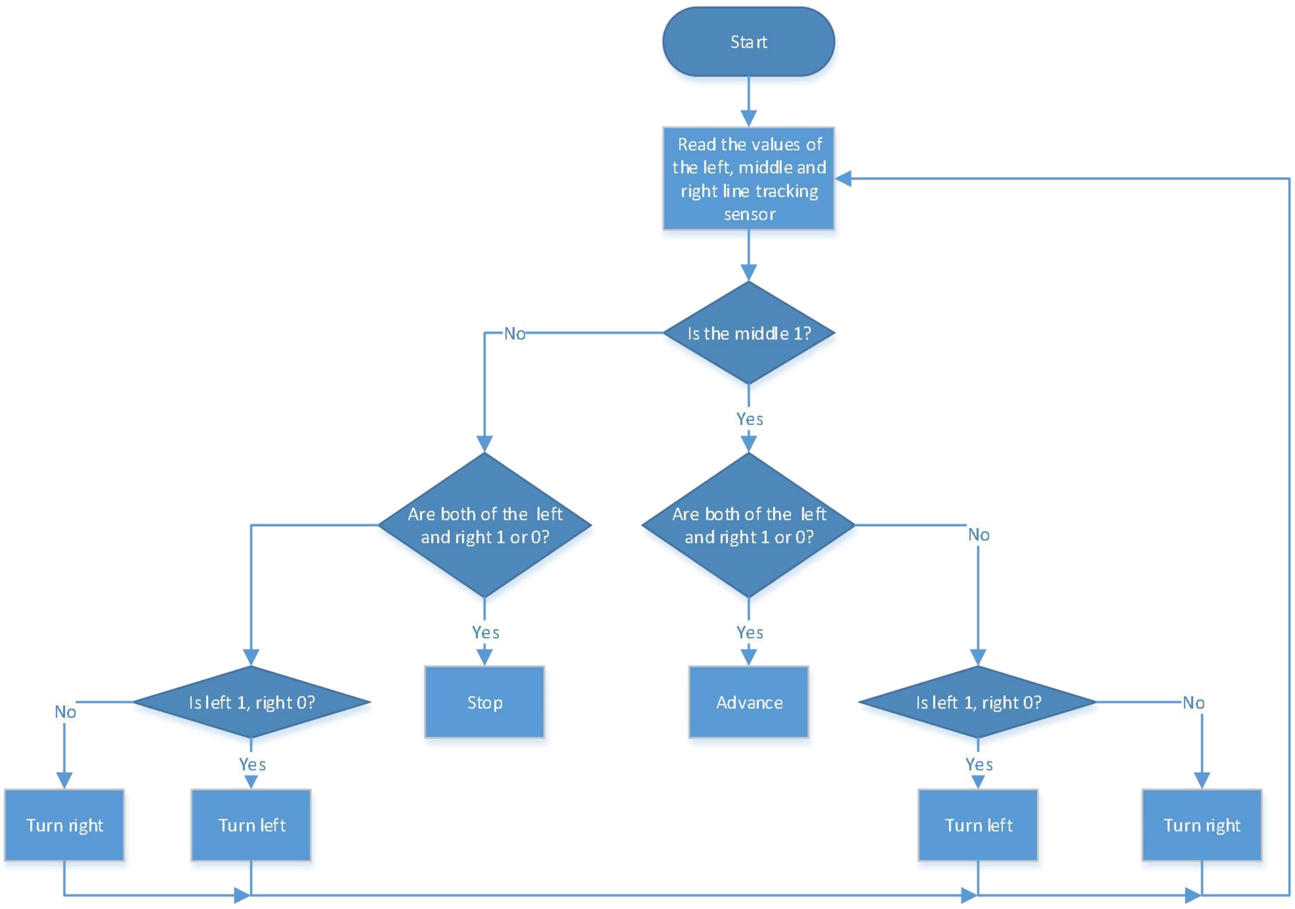
Built on the working principle of the line tracking sensor we could make a line tracking car.

Detection	Middle tracking sensor	detects black line：HIGH
		detects white line：LOW
	Left tracking sensor	etects black line：HIG

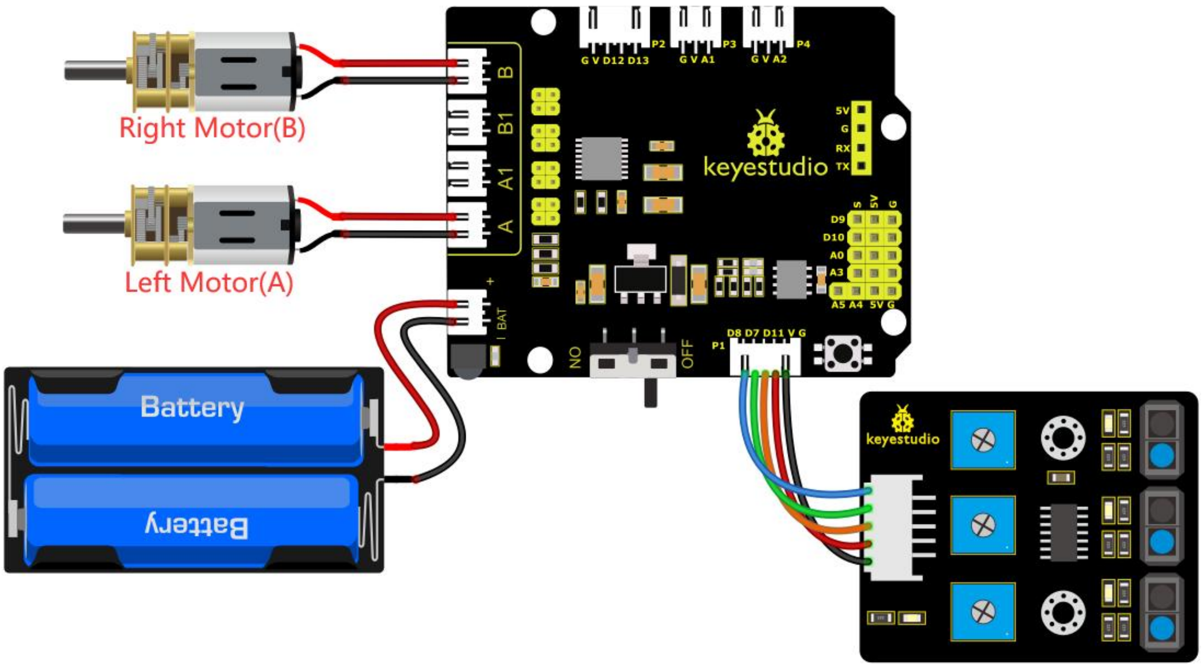
		detects white line：LOW
	Right tracking sensor	detects black line：HIGH
		detects white line：LOW

Condition 1	Status 2 detecting the left and the right tracking sensor	Status
Middle tracking sensor detects black line	left tracking sensor detects black line; right sensor detects white line	Rotate to left
	left tracking sensor detects white line; right sensor detects black line	Rotate to right
	left and right tracking sensor detect black line	Go front
	left and right tracking sensor detect white line	Go front
Middle tracking sensor detects white line	Only left line tracking sensor detects black line	Rotate to left
	Only right line tracking sensor detects black line	Rotate to right
	Left and right line tracking sensors	stop

	detect black line	
	Left and right line tracking sensors detect white line	stop


Start
Read the values of the left, middle and right line tracking sensor
Is the middle 1?
No
Yes
Are both of the left and right 1 or 0?
Are both of the left and right 1 or 0?
No
Yes
Yes
Stop
Advance
Is left 1, right 0?
Is left 1, right 0?
No
Yes
Yes
No
Turn right
Turn left
Turn left
Turn right

(2) Connection Diagram



Wiring up:

G, V, S1, S2 and S3 of the line tracking sensor are connected to G (GND), V (VCC), D11, D7 and D8 of the sensor expansion board.

The power is connected to the BAT port

(3) Test Code

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 11
 Tracking turtle
 http://www.keyestudio.com
*/
int left_ctrl = 4;//define direction control pin of A motor
 int left_pwm = 6;//define PWM control pin of A motor
 int right_ctrl = 2;//define direction control pin of B motor
 int right_pwm = 5;//define PWM control pin of B motor
 int sensor_L = 11;//define the pin of left line tracking sensor
 int sensor_M = 7;//define the pin of middle line tracking sensor
 int sensor_R = 8;//define the pin of right line tracking sensor
int L_val,M_val,R_val;//define these variables

void setup() {
  Serial.begin(9600);//start serial monitor and set baud rate to 9600
  pinMode(left_ctrl,OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(left_pwm,OUTPUT);//set PWM control pin of A motor to OUTPUT
  pinMode(right_ctrl,OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(right_pwm,OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(sensor_L,INPUT);//set the pins of left line tracking sensor to INPUT
  pinMode(sensor_M,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
  M_val = digitalRead(sensor_M);//read the value of middle line tracking sensor
  R_val = digitalRead(sensor_R);//read the value of right line tracking sensor

  if(M_val == 1){//if the state of middle one is 1, which means detecting black line
```

```
     if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not on the right, turn left
        left();
    }
     else if (L_val == 0 && R_val == 1) { //Otherwise, if a black line is detected on the right and not on the left, turn right
      right();
    }
     else { //Otherwise,forward
      front();
    }
  }
  else { //No black lines detected in the middle
    if (L_val == 1 && R_val == 0) { //If a black line is detected on the left, but not on the right, turn left
      left();
    }
    else if (L_val == 0 && R_val == 1) { //Otherwise, if a black line is detected on the right and not on the left, turn right
      right();
    }
    else { //Otherwise,stop
      Stop();
    }
  }
}
void front()//define the status of going forward
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,100);
}
void back()//define the state of going back
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,155);
}
void left()//define the left-turning state
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,100);
}
void right()//define the right-turning state
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,155);
}
void Stop()//define the state of stop
{
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm,0);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm,0);
}
//*************************************************************************
```

(4) Test Result

Uploading the code to the development board, powering up and turning the DIP switch to ON. The turtle car will walk along black lines.

Project 12: Ultrasonic Follow Robot

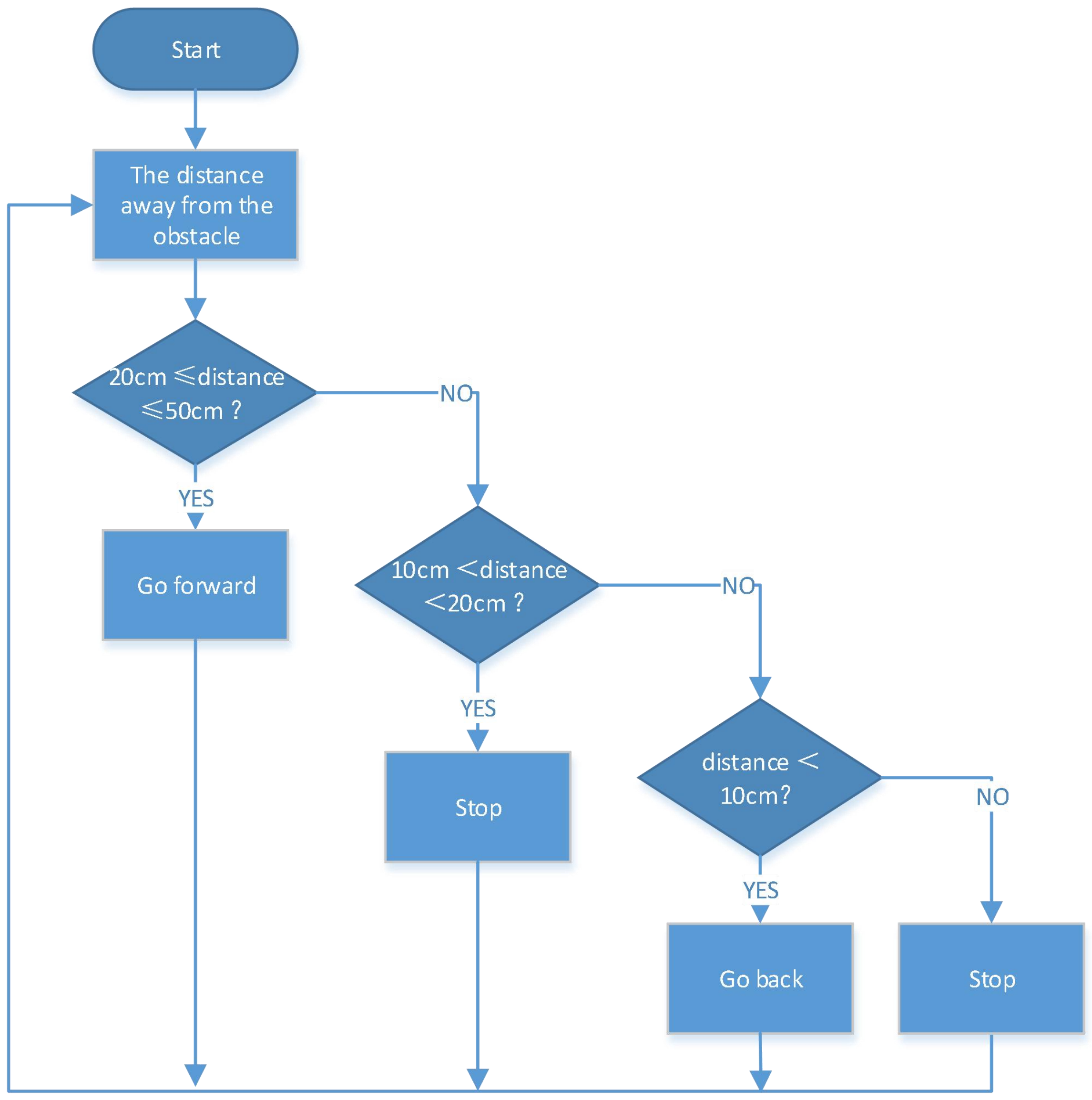
(1) Description

In this project, we detect the distance from the obstacle to drive two motors so as to make robot car move and 8*8 dot matrix shows smile facial pattern.

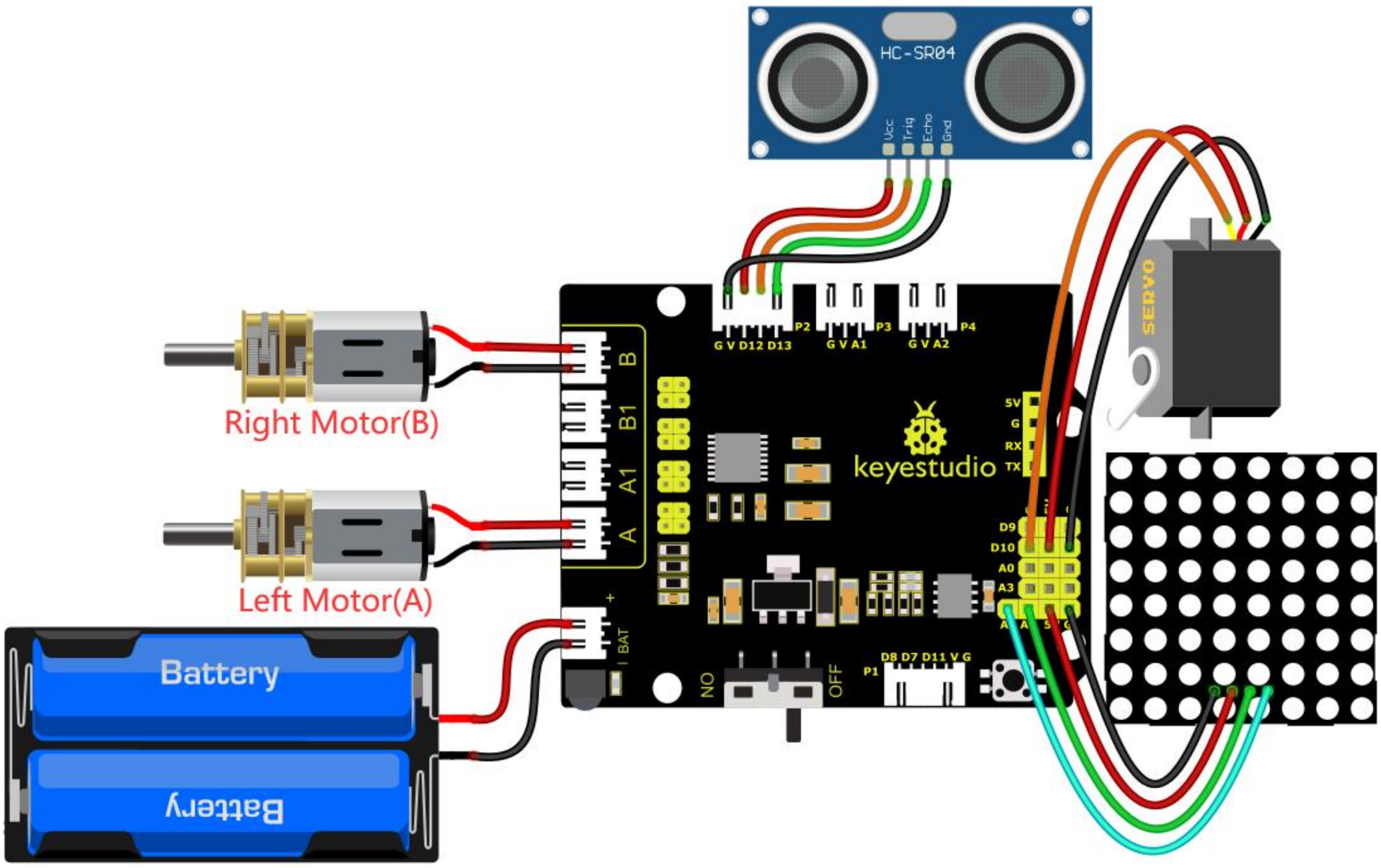
The specific logic of ultrasonic follow robot car is shown below:

Detection	Measured distance of front obstacles	distance (unit: cm)
Setting	Set servo to 90°	
	Make dot matrix show a smile face pattern	
Condition	distance≥20 and distance≤50	

Status	Go forward
Condition	distance > 10 and distance < 20
	distance > 50
Status	stop
Condition	distance≤10
Status	Go back



(2) Hook-up Diagram



Wiring up：

1. GND, VCC, SDA and SCL of the 8*8 dot matrix module are connected to G （GND), V （VCC), A4 and A5 of the expansion board.

2. VCC, Trig, Echo and Gnd of the ultrasonic sensor are connected to 5V(V), D12(S), D13(S) and Gnd(G)

3. The servo is connected to G, V and D10. The brown wire is interfaced with Gnd(G), the red wire is interfaced with 5V(V) and the orange wire is

interfaced with D10.

4. The power is connected to the BAT port

(3) Test Code

```
//*************************************************************************************
/*
 keyestudio smart turtle robot
 lesson 12
 flowing turtle
 http://www.keyestudio.com
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);// set the pins of dot matrix to A4 and A5.
//Array, used to store the data of pattern, can be calculated by yourself or obtained from 
the modulus tool
uint8_t matrix_smile[8]={0x42,0xa5,0xa5,0x00,0x00,0x24,0x18,0x00};
uint8_t  LEDArray[8];

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

#include "SR04.h" //define the function library of ultrasonic sensor
#define TRIG_PIN 12// set the signal of ultrasonic sensor to D12
#define ECHO_PIN 13// set the signal of ultrasonic sensor to D13
SR04 sr04 = SR04(ECHO_PIN,TRIG_PIN);
long distance;

int left_ctrl = 4;//define the direction control pin of A motor
int left_pwm = 6;//define the speed control pin of A motor
int right_ctrl = 2;//define the direction control pin of B motor
int right_pwm = 5;//define the speed control pin of B motor

void setup() {
  Serial.begin(9600);//open serial monitor and set baud rate to 9600
  pinMode(left_ctrl,OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(left_pwm,OUTPUT);//set PWM control pin of A motor to OUTPUT
  pinMode(right_ctrl,OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(right_pwm,OUTPUT);//set PWM control pin of B motor to OUTPUT
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  myservo.attach(10);  // attaches the servo on pin 10 to the servo object
  myservo.write(90);
  delay(500); //waits 500ms
  myMatrix.begin(112);
  myMatrix.clear();
  myMatrix.writeDisplay();//show stop pattern
  matrix_display(matrix_smile);  //show stop pattern

}

void loop() {
  distance = sr04.Distance();//the distance detected by ultrasonic sensor
   if(distance <= 10)//if distance is less than 10
  {
    back();//go back
  }
  else if((distance > 10)&&(distance< 20 ))//if 10<distance<20
  {
    Stop();//stop
  }
  else if((distance >= 20)&&(distance <= 50))//if 20≤distance<50
{
    front();//follow
  }
  else//otherwise
  {
    Stop();//stop
  }
}

void front()//define the status of going front
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,100);
}
void back()//define the status of going back
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,150);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,150);
}
```

```
void left()//define the status of turning left
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void right()//define the status of right turning
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}
void Stop()//define the state of stop
{
  digitalWrite(left_ctrl, LOW);
  analogWrite(left_pwm,0);
  digitalWrite(right_ctrl, LOW);
  analogWrite(right_pwm,0);
}

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

(4) Test Results:

After uploading the test code successfully, connecting according to the wiring diagram, dialing the DIP switch to the right end, powering it on, and setting the servo to 90°， the smart car will move with the obstacle and the matrix will show “smile”

Project 13: Ultrasonic Avoiding Robot

(1) **Description**

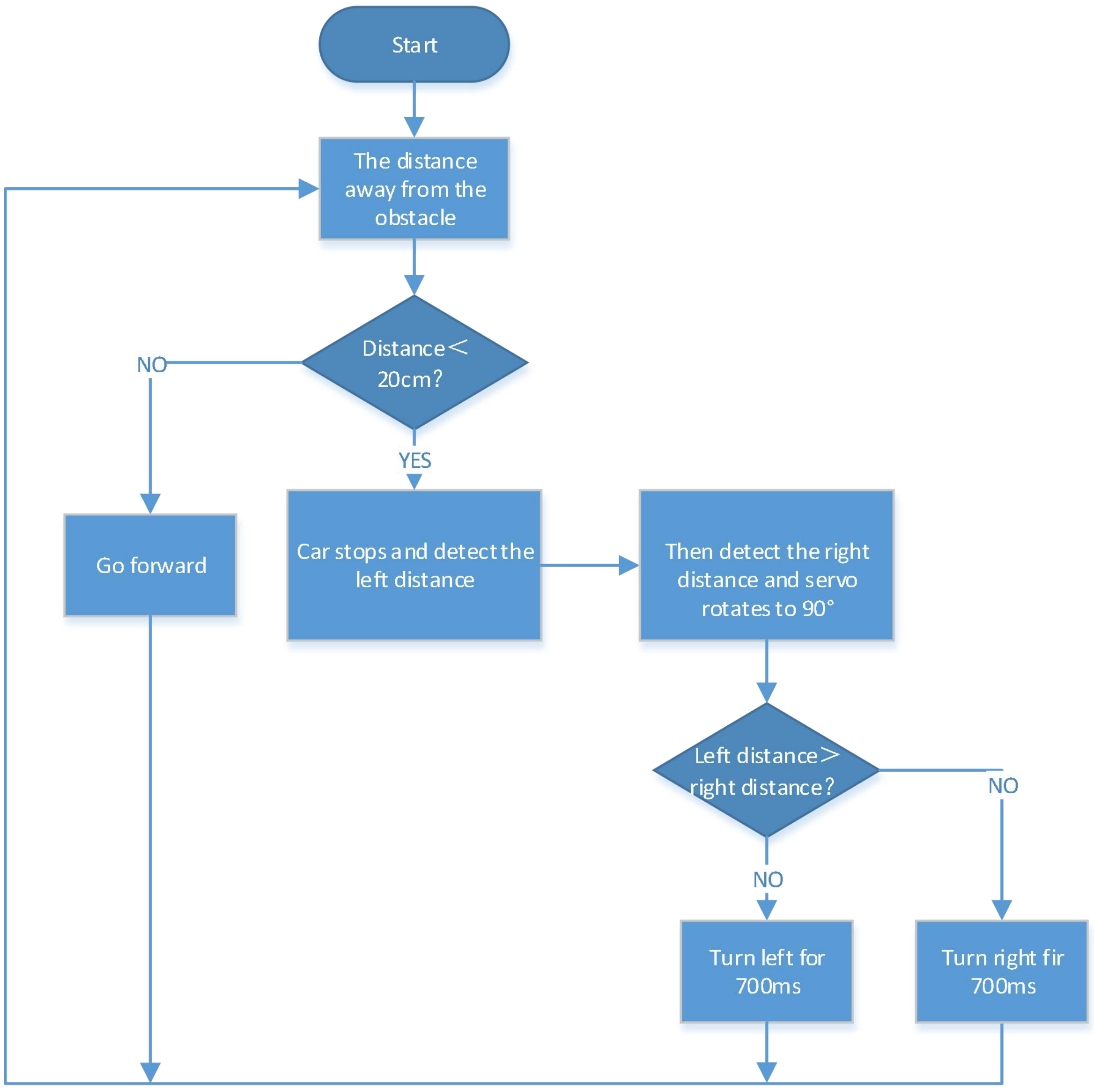
We’ ve learned LED matrix, motor drive, ultrasonic sensor and servo in previous lessons. Next, we could make an ultrasonic avoiding robot!

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

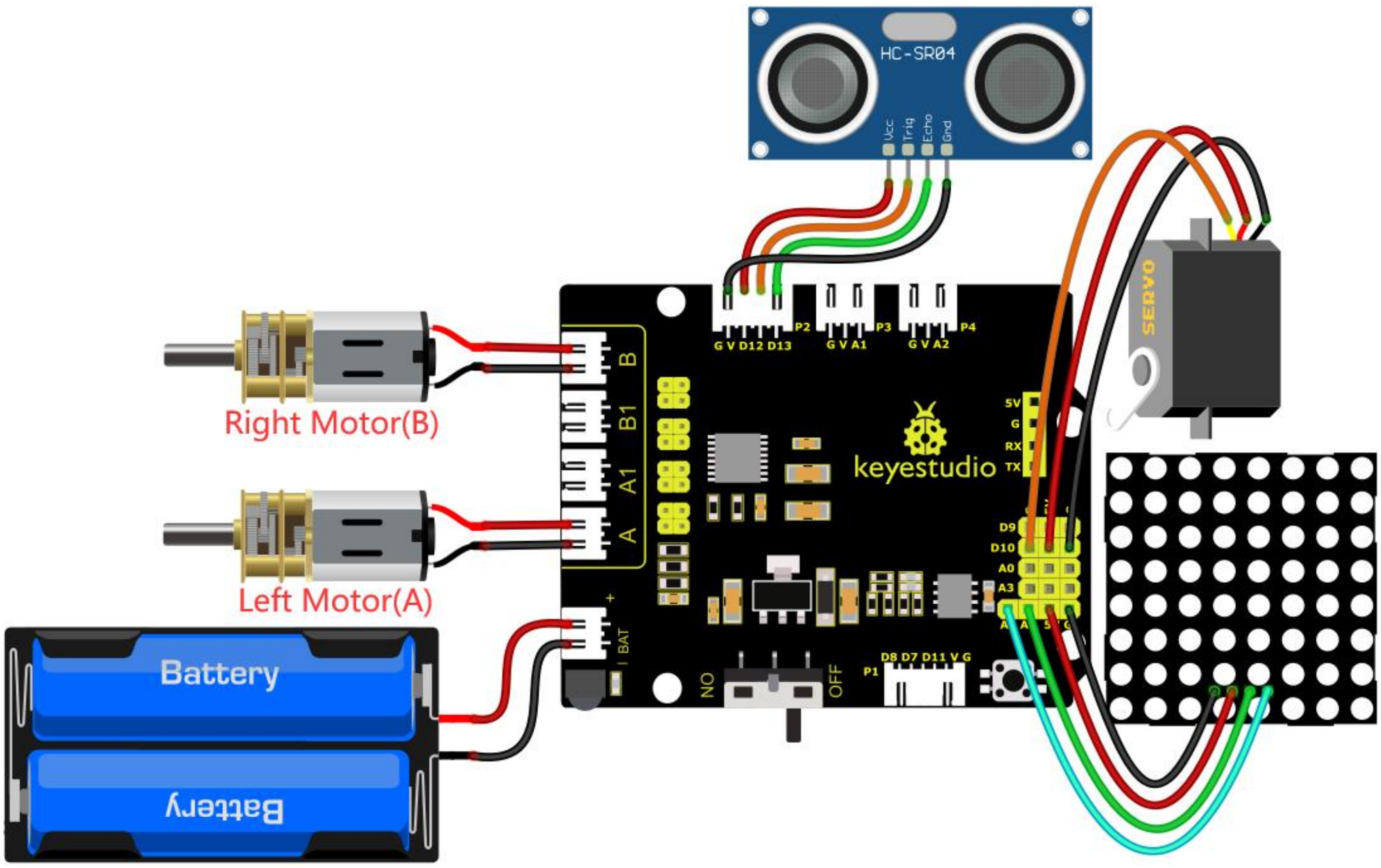
The specific logic of ultrasonic avoiding smart car is shown below:

Detection	measured distance of front obstacle set servo to 90°	distance1（unit：cm）
	measured distance of left obstacle（set servo to 160°）	distance2（unit：cm）
	measured distance of right obstacle （set servo to 20°）	distance3（unit：cm）
Setting	set the initial angle of servo to 90°	
Condition1	Status	
distance ＜ 20	Stop for 1000ms； set the angle of servo to 160°， read distance1， delay in 500ms； set the angle of servo to 20°，read distance2，delay in 500ms	
	Condition 2	Status
	distance1＞distance2	Set the angle of servo to 90°，rotate to left for 700ms and go front
	distance1≤distance2	Set the angle of servo to 90 °， rotate to right for 700ms, go front

Condition 1	Status
distance ≥ 20	Go front


Start
The distance away from the obstacle
Distance＜20cm?
NO
YES
Go forward
Car stops and detect the left distance
Then detect the right distance and servo rotates to 90°
Left distance＞right distance?
NO
NO
Turn left for 700ms
Turn right fir 700ms

(2) Connection Diagram



1. GND, VCC, SDA and SCL of the 8*8 dot matrix module are connected to G (GND), V (VCC), A4 and A5 of the expansion board.

2. VCC, Trig, Echo and Gnd of the ultrasonic sensor are connected to 5V(V), D12(S), D13(S) and Gnd(G)

3. The servo is connected to G, V and D10. The brown wire is interfaced with Gnd(G), the red wire is interfaced with 5V(V) and the orange wire is interfaced with D10.

4. The power is connected to the BAT port

(3) Test Code

```
//*************************************************************************
/*
 keyestudio smart turtle robot
 lesson 13
 avoiding turtle
 http://www.keyestudio.com
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);// set the pins of dot matrix to A4 and A5.
//Array, used to store the data of pattern, can be calculated by yourself or obtained from
the modulus tool
uint8_t matrix_heart[8]={0x66,0x99,0x81,0x81,0x42,0x24,0x18,0x00};
uint8_t matrix_smile[8]={0x42,0xa5,0xa5,0x00,0x00,0x24,0x18,0x00};
uint8_t matrix_front2[8]={0x18,0x24,0x42,0x99,0x24,0x42,0x81,0x00};
uint8_t matrix_back2[8]={0x00,0x81,0x42,0x24,0x99,0x42,0x24,0x18};
uint8_t matrix_left2[8]={0x12,0x24,0x48,0x90,0x90,0x48,0x24,0x12};
uint8_t matrix_right2[8]={0x48,0x24,0x12,0x09,0x09,0x12,0x24,0x48};
uint8_t matrix_stop2[8]={0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18};
uint8_t  LEDArray[8];
const int left_ctrl = 4;//define direction control pin of A motor
const int left_pwm = 6;//define PWM control pin of A motor
const int right_ctrl = 2;//define direction control pin of B motor
const int right_pwm = 5;//define PWM control pin of B 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 distance1,distance2,distance3;//define three distance
const int servopin = 10;//set the pin of servo to D10
int myangle;
int pulsewidth;
int val;

void setup() {
  Serial.begin(9600);//open serial monitor and set baud rate to 9600
  pinMode(left_ctrl,OUTPUT);//set direction control pin of A motor to OUTPUT
  pinMode(left_pwm,OUTPUT);//set PWM control pin of A motor to OUTPUT
  pinMode(right_ctrl,OUTPUT);//set direction control pin of B motor to OUTPUT
  pinMode(right_pwm,OUTPUT);//set PWM control pin of B motor to OUTPUT
  servopulse(servopin,90);//the angle of servo is 90 degree
  delay(300);
  myMatrix.begin(112);
  myMatrix.clear();
}

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

void avoid()
{
  distance1=sr04.Distance(); //obtain the value detected by ultrasonic sensor

  if((distance1 < 20)&&(distance1 != 0))//if the distance is greater than 0 and less than
10

  {
    car_Stop();//stop
    myMatrix.clear();
    myMatrix.writeDisplay();//show stop pattern
    matrix_display(matrix_stop2);  //show stop pattern
    delay(500);
    servopulse(servopin,160);//servo rotates to 160°
    delay(500);
    distance2=sr04.Distance();//measure the distance
    delay(100);
    servopulse(servopin,20);//rotate to 20 degree
    delay(500);
    distance3=sr04.Distance();//measure the distance
    delay(100);
    servopulse(servopin,90);  //Return to the 90 degree position
    delay(500);
    if(distance2 > distance3)//compare the distance, if left distance is more than right
distance
    {
      car_left();//turn left
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_left2);    //display left-turning pattern
      servopulse(servopin,90);//servo rotates to 90 degree
      delay(700); //turn left 700ms
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_front2);   //show forward pattern
    }
    else//if the right distance is greater than the left
    {
      car_right();//turn right
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_right2);   //display right-turning pattern
      servopulse(servopin,90);//servo rotates to 90 degree
      delay(700);
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_front2);  //show forward pattern
    }
  }
  else//otherwise
  {
    car_front();//go forward
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_front2);  // show forward pattern
  }
}

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

}

void car_front()//car goes forward
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_back()//go back
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}
void car_left()//car turns left
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_right()//car turns right
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}

void car_Stop()//stop
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,0);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,0);
}

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

(4) Test Result

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

Project 14: IR Remote Control Robot

(1) Description

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

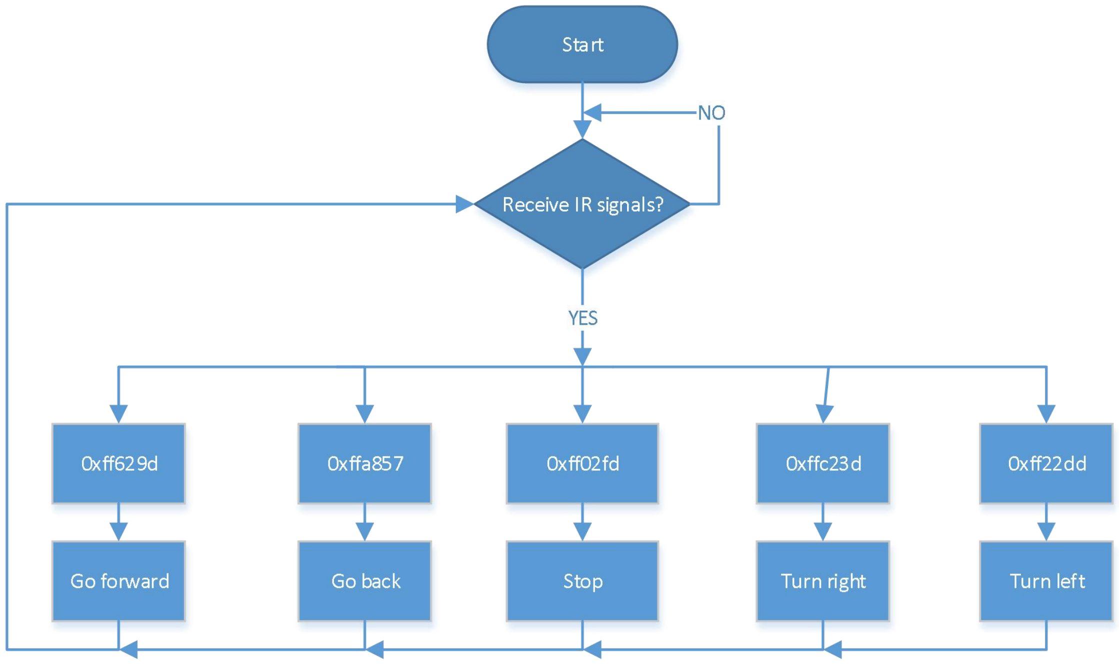
Press the button on IR remote control to drive robot car to move, and the corresponding state pattern is displayed on the 8*16 LED matrix.

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

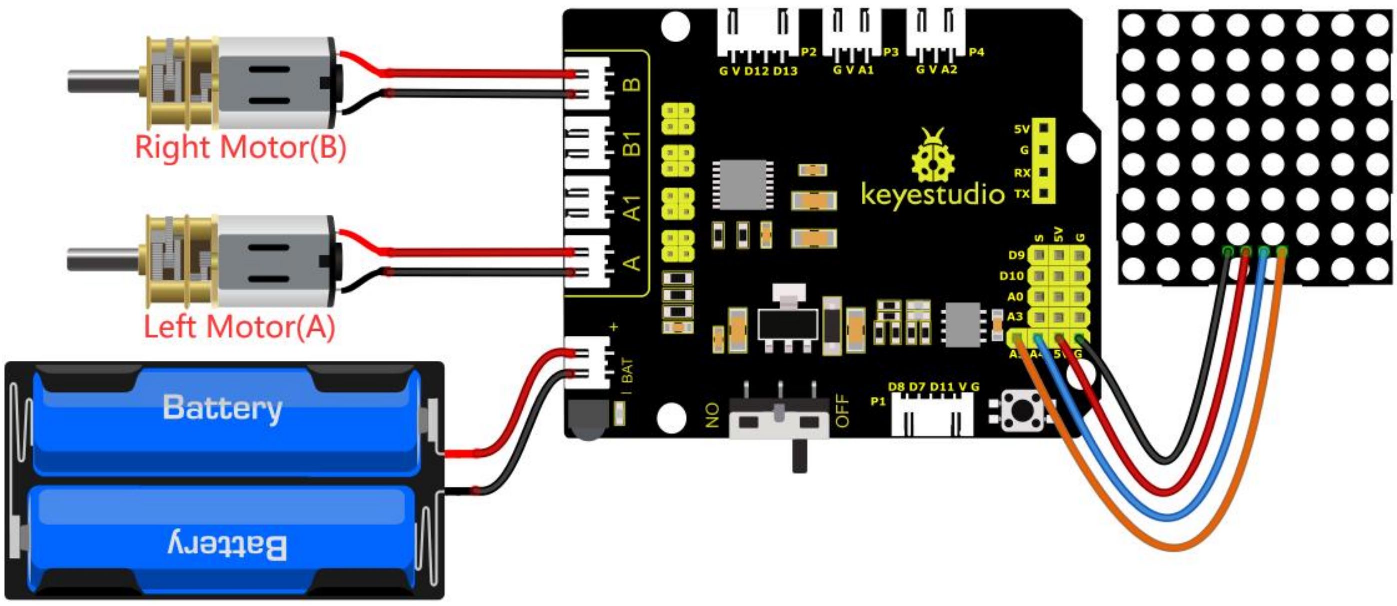
Initial setup	Dot matrix displays smile face	
Remote control	Key Value	Key state
▲	FF629D	Go front
		8*8 LED matrix shows front icon

▼	FFA857	Back
		8*8 LED matrix shows back icon
◀	FF22DD	Rotate to left
		8*8 LED matrix shows leftward icon
▶	FFC23D	Rotate to right
		8*8 LED matrix shows rightward icon
OK	FF02FD	Stop
		8*8 LED matrix shows “STOP”
4	FF30CF	Turn left
		8*8 LED matrix shows leftward icon
6	FF7A85	Turn right
		8*8 LED matrix shows rightward icon

Flow Chart



(2) Hook-up Diagram



1. GND, VCC, SDA and SCL of the 8*8 dot matrix module are connected to G（GND), V（VCC), A4 and A5 of the expansion board.

2. Since the Keyestudio 8833 motor driver expansion board integrates the IR receiver, pins of the IR receiver are G （GND), V （VCC） and D3

3. The power is connected to the BAT port

(3) Test Code

```
//*************************************************************************************
/*
keyestudio smart turtle robot
lesson 14
remote control turtle
http://www.keyestudio.com
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);
//Array, used to store the data of pattern, can be calculated by yourself or obtained from
the modulus tool
uint8_t matrix_heart[8]={0x66,0x99,0x81,0x81,0x42,0x24,0x18,0x00};
uint8_t matrix_smile[8]={0x42,0xa5,0xa5,0x00,0x00,0x24,0x18,0x00};
uint8_t matrix_front2[8]={0x18,0x24,0x42,0x99,0x24,0x42,0x81,0x00};
uint8_t matrix_back2[8]={0x00,0x81,0x42,0x24,0x99,0x42,0x24,0x18};
uint8_t matrix_left2[8]={0x12,0x24,0x48,0x90,0x90,0x48,0x24,0x12};
uint8_t matrix_right2[8]={0x48,0x24,0x12,0x09,0x09,0x12,0x24,0x48};
uint8_t matrix_stop2[8]={0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18};
uint8_t  LEDArray[8];
const int left_ctrl = 4;//define the direction control pin of A motor
const int left_pwm = 6;//define the speed control of A motor
const int right_ctrl = 2;//define the direction control pin of B motor
const int right_pwm = 5;//define the speed control pin of B motor
#include <IRremote.h>//function library of IR remote control
int RECV_PIN = 3;//set the pin of IR receiver to D3
IRrecv irrecv(RECV_PIN);
long irr_val;
decode_results results;


void setup()
{
  pinMode(left_ctrl,OUTPUT);//
  pinMode(left_pwm,OUTPUT);//
  pinMode(right_ctrl,OUTPUT);//
  pinMode(right_pwm,OUTPUT);//
    Serial.begin(9600);//
  // In case the interrupt driver crashes on setup, give a clue
  // to the user what's going on.
  Serial.println("Enabling IRin");
  irrecv.enableIRIn(); // Start the receiver
  Serial.println("Enabled IRin");
  myMatrix.begin(112);
  myMatrix.clear();
  myMatrix.writeDisplay();
  matrix_display(matrix_smile);
}
void loop()
 {
  if (irrecv.decode(&results))
 {
    irr_val = results.value;
    Serial.println(irr_val, HEX);//serial prints the read IR remote signals
    switch(irr_val)
    {
      case 0xFF629D : car_front();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_front2);
      break;

      case 0xFFA857 : car_back();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_back2);
      break;
      case 0xFF22DD : car_left();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_left2);
      break;

      case 0xFFC23D : car_right();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_right2);
      break;
      case 0xFF02FD : car_Stop();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_stop2);
      break;
    }
       irrecv.resume(); // Receive the next value
  }
}
void car_front()//define the state of going front
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_back()//define the status of going back
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}
void car_left()//set the status of left turning
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_right()//set the status of right turning
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}
void car_Stop()//define the state of stop
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,0);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,0);
}
```

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

(4) Test Result

Upload code power on the external power, turn the DIP switch to ON and press buttons on IR remote control. The turtle robot car will follow the obstacle to move.

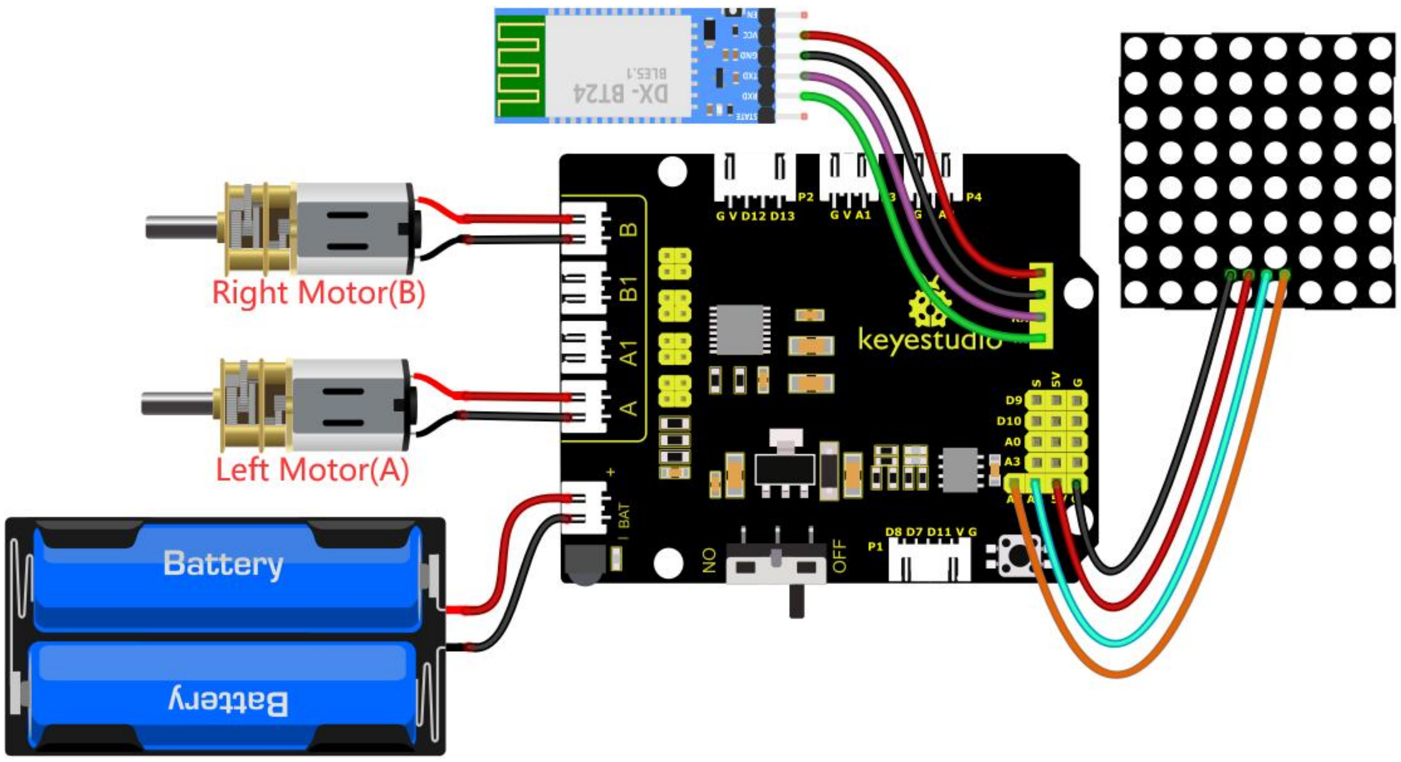
Project 15: Bluetooth Control Tank

(1) Description

We’ve learned the basic knowledge of Bluetooth. And in this lesson, we will make a Bluetooth remote smart car. In this 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) Wiring Diagram



Note:

1. RXD, TXD, GND and VCC of the BT module are connected to TX, RX, G (GND) and 5V (VCC).

STATE and BRK of the BT module don’ t need connection.

2. GND, VCC, SDA and SCL of the 8*8 dot matrix module are connected to G (GND), V (VCC), A4 and A5 of the expansion board.

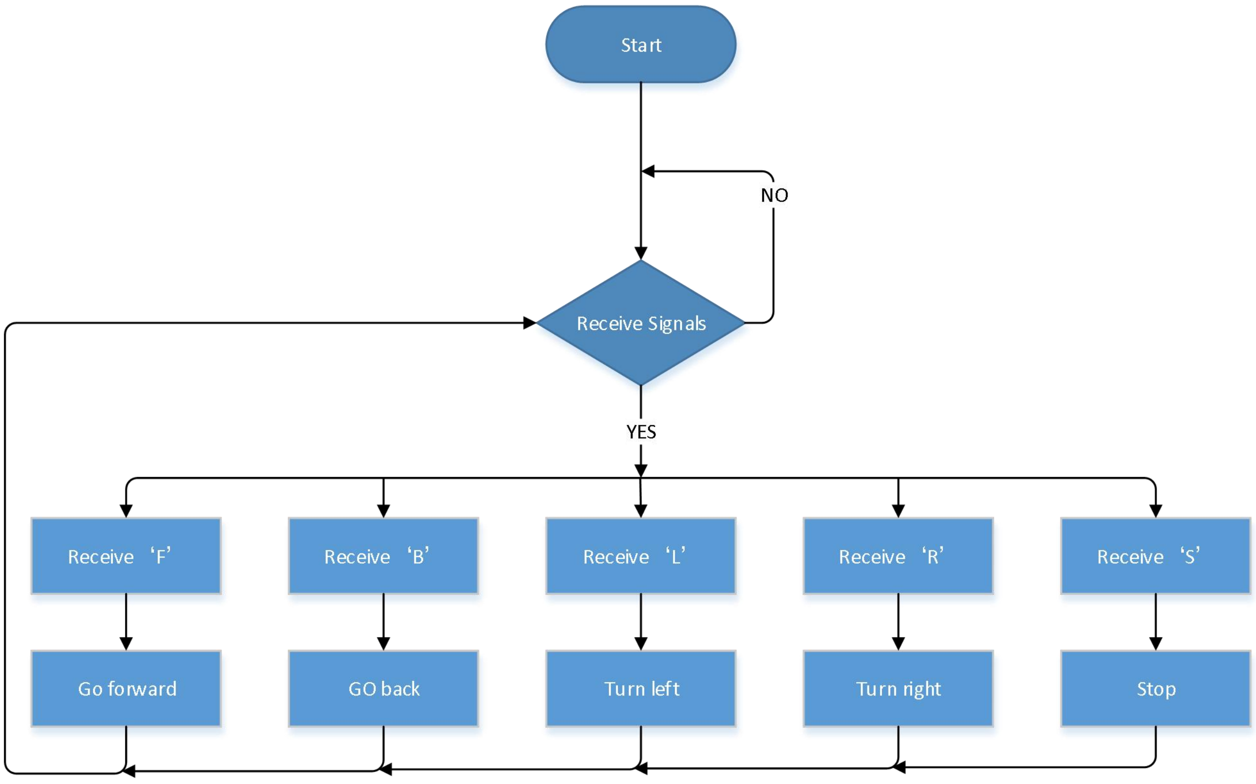
3. The power is connected to the BAT port

(3) APP interface

CONNECT	Pair the BT24 Bluetooth module	
Bluetooth	Enter control page of Bluetooth	
DISCONNECT	Disconnect Bluetooth	
↑	Press: F Release: S	Press the button, robot goes front; release to stop
←	Press: L Release: S	Press the button, robot turns left; release to stop
▶	Click to send "S"	Stop
→	Press: R Release: S	Press the button, robot turns right; release to stop
↓	Press: B Release: S	Press the button, robot goes back; release to stop
	Click to send "Y"	Start Ultrasonic follow function; click Stop to exit
	---	Click to start the mobile gravity sensing; click again to exit
	Click to send "U"	Start ultrasonic avoiding function; click Stop to exit

	Click to send "X"	Start line tracking function; click Stop to exit

Flow Chart



(4) Test Code

```
//*************************************************************************************
/*
keyestudio smart turtle robot
lesson 15
Bluetooth Control turtle
http://www.keyestudio.com
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);
//Array, used to store the data of pattern, can be calculated by yourself or obtained from
the modulus tool
uint8_t matrix_heart[8]={0x66,0x99,0x81,0x81,0x42,0x24,0x18,0x00};
uint8_t matrix_smile[8]={0x42,0xa5,0xa5,0x00,0x00,0x24,0x18,0x00};
uint8_t matrix_front2[8]={0x18,0x24,0x42,0x99,0x24,0x42,0x81,0x00};
uint8_t matrix_back2[8]={0x00,0x81,0x42,0x24,0x99,0x42,0x24,0x18};
uint8_t matrix_left2[8]={0x12,0x24,0x48,0x90,0x90,0x48,0x24,0x12};
uint8_t matrix_right2[8]={0x48,0x24,0x12,0x09,0x09,0x12,0x24,0x48};
uint8_t matrix_stop2[8]={0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18};
uint8_t  LEDArray[8];
unsigned char data_line = 0;
unsigned char delay_count = 0;
const int left_ctrl = 4;//define direction control pin of A motor
const int left_pwm = 6;//define PWM control pin of A motor
const int right_ctrl = 2;//define direction control pin of B motor
const int right_pwm = 5;//define PWM control pin of B motor
char BLE_val;
void setup()
{
  Serial.begin(9600);
  pinMode(left_ctrl,OUTPUT);
  pinMode(left_pwm,OUTPUT);
  pinMode(right_ctrl,OUTPUT);
  pinMode(right_pwm,OUTPUT);
  myMatrix.begin(112);
  myMatrix.clear();
  myMatrix.writeDisplay();
  matrix_display(matrix_smile);
}

void loop()
```

```
{
  if(Serial.available()>0)
  {
    BLE_val = Serial.read();
    Serial.println(BLE_val);
  }
  switch(BLE_val)
  {
    case 'F': car_front();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_front2);
    break;

    case 'B': car_back();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_back2);
    break;

    case 'L': car_left();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_left2);
    break;

    case 'R': car_right();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_right2);
    break;

    case 'S': car_Stop();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_stop2);
    break;
  }

}
void car_front()//go front
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_back()//go backward
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}
void car_left()//turn left
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_right()//turn right
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}

void car_Stop()//stop
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,0);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,0);
}
// the function that dot matrix shows patterns
void matrix_display(unsigned char matrix_value[])
{
  for(int i=0; i<8; i++)
    {
      LEDArray[i]=matrix_value[i];
      for(int j=7; j>=0; j--)
      {
        if((LEDArray[i]&0x01)>0)
        myMatrix.drawPixel(j, i,1);
        LEDArray[i] = LEDArray[i]>>1;
      }
    }
    myMatrix.writeDisplay();
}
//*************************************************************************************
```

(5) Test Result

Upload the cod and power up and turn the DIP switch to ON. Inset the BT module and open your cellphone to connect Bluetooth to control the turtle car.

The turtle can move forward, backward, turn left and right and so on. Also the 8*8 module will show corresponding patterns

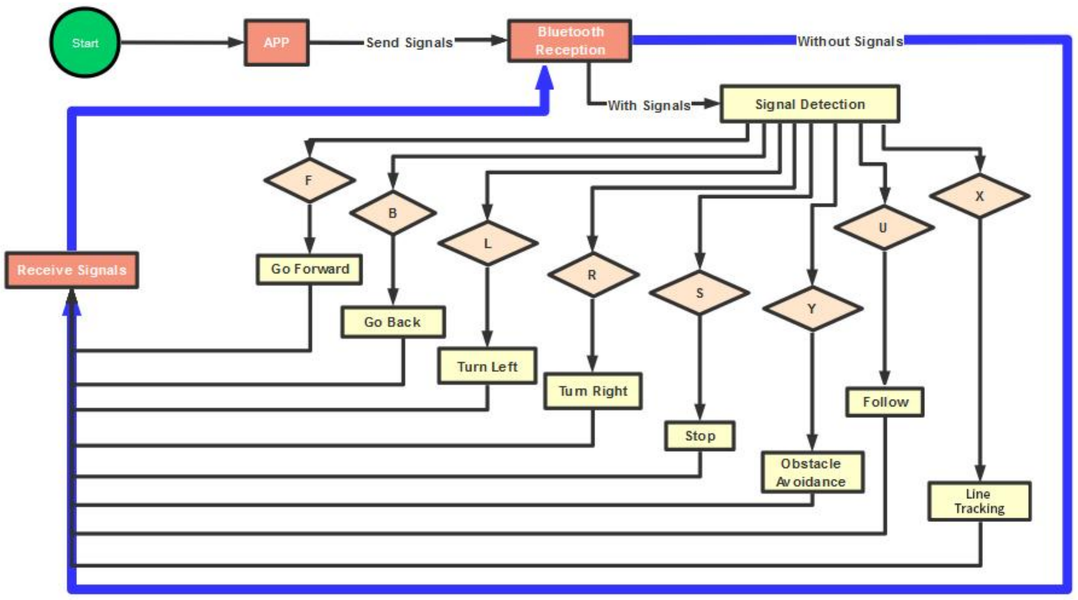
Remove the BT module when you are uploading the code, otherwise you will fail to upload it.

Project 16: Multi-purpose Bluetooth Robot

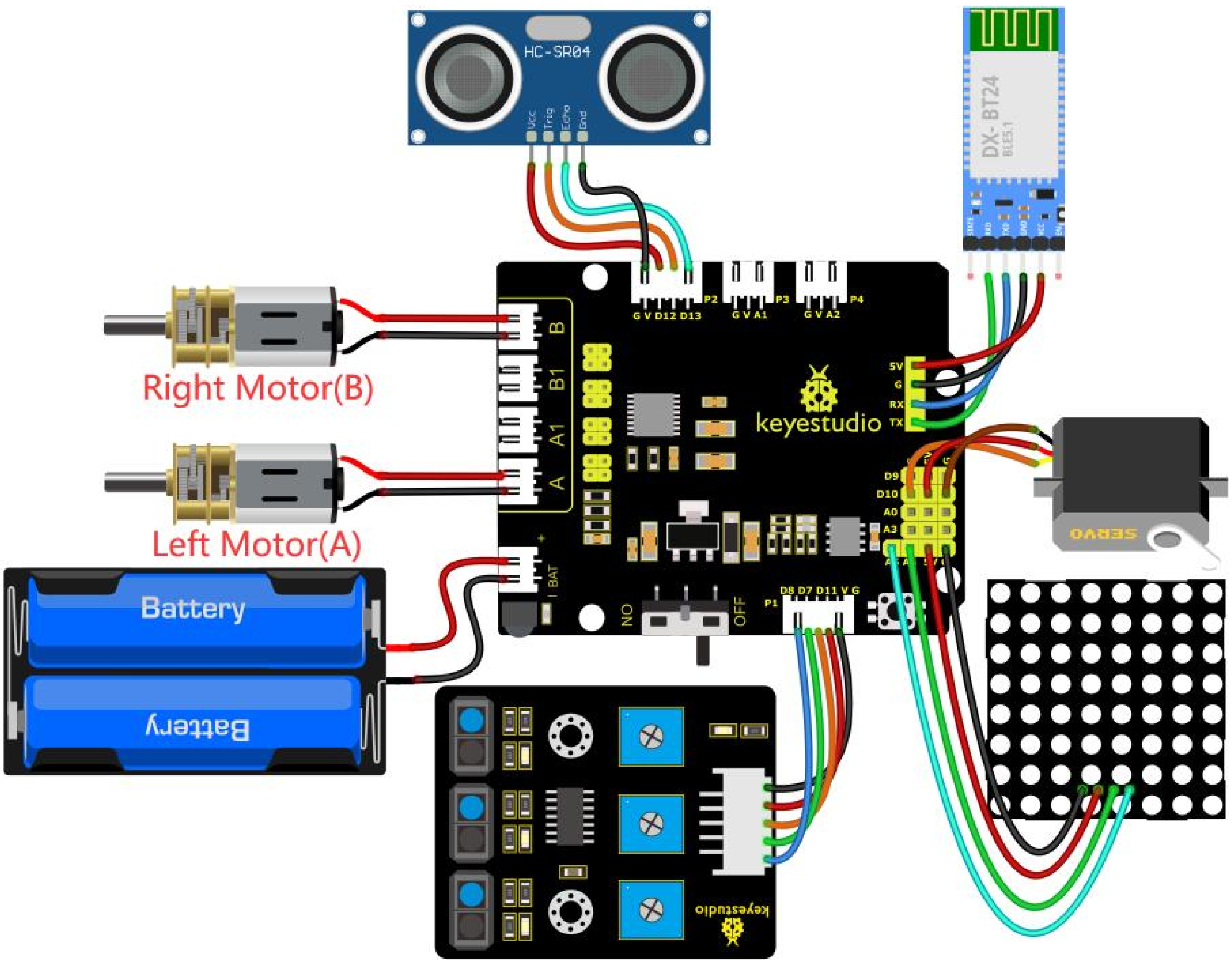
(1) Description

In previous projects, the turtle robot car only performs a single function. However, in this lesson, we will integrate all of its functions via Bluetooth control.

(2) Flow Chart



(3) Connection Diagram



Wiring up：

1. GND, VCC, SDA and SCL of the 8*8 dot matrix module are connected to G（GND), V（VCC), A4 and A5 of the expansion board.

2. VCC, Trig, Echo and Gnd of the ultrasonic sensor are connected to 5V(V), D12(S), D13(S) and Gnd(G)

3. The servo is connected to G, V and D10. The brown wire is interfaced with Gnd(G), the red wire is interfaced with 5V(V) and the orange wire is interfaced with D10.

4. RXD, TXD, GND and VCC of the BT module are connected to TX, RX, G（GND) and 5V（VCC).

STATE and BRK of the BT module don’t need connection.

5. Since the Keyestudio 8833 motor driver expansion board integrates the IR receiver, pins of the IR receiver are G（GND), V（VCC）and D3

6. G, V, S1, S2 and S3 of the line tracking sensor are connected to G（GND), V（VCC), D11, D7 and D8 of the sensor expansion board.

7. The power is connected to the AT port

(4) Test Code

```
//*******************************************************************************
/*
keyestudio smart turtle robot
lesson 16
Multifunctional turtle robot
http://www.keyestudio.com
*/
#include <ks_Matrix.h>
Matrix myMatrix(A4,A5);
//Array, used to store the data of pattern, can be calculated by yourself or obtained from
the modulus tool
uint8_t matrix_heart[8]={0x66,0x99,0x81,0x81,0x42,0x24,0x18,0x00};
uint8_t matrix_smile[8]={0x42,0xa5,0xa5,0x00,0x00,0x24,0x18,0x00};
uint8_t matrix_front2[8]={0x18,0x24,0x42,0x99,0x24,0x42,0x81,0x00};
uint8_t matrix_back2[8]={0x00,0x81,0x42,0x24,0x99,0x42,0x24,0x18};
uint8_t matrix_left2[8]={0x12,0x24,0x48,0x90,0x90,0x48,0x24,0x12};
uint8_t matrix_right2[8]={0x48,0x24,0x12,0x09,0x09,0x12,0x24,0x48};
uint8_t matrix_stop2[8]={0x18,0x18,0x18,0x18,0x18,0x00,0x18,0x18};
uint8_t  LEDArray[8];
#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 = 6;
const int right_ctrl = 2;
const int right_pwm = 5;
const int sensor_l = 11;
const int sensor_c = 7;
const int sensor_r = 8;
int l_val,c_val,r_val;
const int servopin = 10;
int myangle;
int pulsewidth;
int val;
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);
  myMatrix.begin(112);
  myMatrix.clear();
  myMatrix.writeDisplay();
  matrix_display(matrix_smile);
}

void loop() {
  if(Serial.available()>0)
  {
    BLE_val = Serial.read();
    Serial.println(BLE_val);
  }
  switch(BLE_val)
  {
    case 'F': car_front();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_front2);
    break;
    case 'B': car_back();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_back2);
    break;
    case 'L': car_left();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_left2);
    break;
    case 'R': car_right();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_right2);
    break;
    case 'S': car_Stop();
    myMatrix.clear();
    myMatrix.writeDisplay();
    matrix_display(matrix_stop2);
    break;
    case 'X': tracking();
    break;
    case 'Y': follow_car();
    break;
    case 'U': avoid();
    break;
  }
}

void avoid()
{
  myMatrix.clear();
  myMatrix.writeDisplay();
  matrix_display(matrix_smile);
  int track_flag = 0;
  while(track_flag == 0)
  {
    distance1=sr04.Distance();
    if((distance1 < 10)&&(distance1 != 0))
    {
      car_Stop();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_stop2);
      delay(100);
      servopulse(servopin,180);
      delay(100);
      distance2=sr04.Distance();
      delay(100);
      servopulse(servopin,0);
      delay(100);
      distance3=sr04.Distance();
      delay(100);
      if(distance2 > distance3)
      {
        car_left();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_left2);
        servopulse(servopin,90);
        //delay(100);
      }
      else
      {
        car_right();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_right2);
        servopulse(servopin,90);
        //delay(100);
      }
    }
    else
    {
      car_front();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_front2);
    }
    if(Serial.available()>0)
    {
      BLE_val = Serial.read();
      if(BLE_val == 'S')
      {
        track_flag = 1;
      }
    }
  }
}

void follow_car()
{
  servopulse(servopin,90);
  int track_flag = 0;
  while(track_flag == 0)
  {
    distance = sr04.Distance();
      if(distance<8)
    {
      car_back();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_back2);
    }
    else if((distance>=8)&&(distance<13))
    {
      car_Stop();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_stop2);
    }
    else if((distance>=13)&&(distance<35))
    {
      car_front();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_front2);
    }
    else
    {
      car_Stop();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_stop2);
    }
    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<20;i++)
  {
    pulsewidth = (myangle*11)+500;
    digitalWrite(servopin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servopin,LOW);
    delay(20-pulsewidth/1000);
  }
}
```

```
void tracking()
{
  myMatrix.clear();
  myMatrix.writeDisplay();
  matrix_display(matrix_smile);
  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)
    {
      car_front2();
      myMatrix.clear();
      myMatrix.writeDisplay();
      matrix_display(matrix_front2);
    }
    else
    {
      if((l_val == 1)&&(r_val == 0))
      {
        car_left();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_left2);
      }
      else if((l_val == 0)&&(r_val == 1))
      {
        car_right();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_right2);
      }
      else
      {
        car_Stop();
        myMatrix.clear();
        myMatrix.writeDisplay();
        matrix_display(matrix_stop2);
      }
    }
    if(Serial.available()>0)
    {
      BLE_val = Serial.read();
      if(BLE_val == 'S')
      {
        track_flag = 1;
      }
    }
  }
}
void car_front()
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,100);
}
void car_front2()
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,155);
}
void car_back()
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,100);
}

void car_left()
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,155);
  digitalWrite(right_ctrl,HIGH);
  analogWrite(right_pwm,100);
}
void car_right()
{
  digitalWrite(left_ctrl,HIGH);
  analogWrite(left_pwm,100);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,155);
```

```
}
void car_Stop()
{
  digitalWrite(left_ctrl,LOW);
  analogWrite(left_pwm,0);
  digitalWrite(right_ctrl,LOW);
  analogWrite(right_pwm,0);
}


//the function that dot matrix shows patterns
void matrix_display(unsigned char matrix_value[])
{
  for(int i=0; i<8; i++)
    {
      LEDArray[i]=matrix_value[i];
      for(int j=7; j>=0; j--)
      {
        if((LEDArray[i]&0x01)>0)
        myMatrix.drawPixel(j, i,1);
        LEDArray[i] = LEDArray[i]>>1;
      }
    }
    myMatrix.writeDisplay();
}
//*********************************************************************************
```

(5) Test Result

(Note: before uploading the test code, you need to remove the Bluetooth module. Otherwise the code will fail to upload. When the code uploading process is done, open the GPS on your phone, and then reconnect the Bluetooth module.)

Upload the code to the development board, power up and turn the DIP switch to ON. Connect the APP to Bluetooth and control the turtle car via the app. Multiple functions can be achieved.

9. Resources

https://www.arduino.cc/

Download test code, libraries and tutorials :

https://fs.keyestudio.com/KS0558

10. Troubleshooting

(1)No response on the turtle car

A： 1. Check batteries capacity

1. Check the wiring up

(2)Your PX can't identify the USB port

A：1. Refer the Chapter 7 How to import a library to install the driver of the CP2102.

2. Check the USB cable

(3) Fail to upload the code

A: 1. Upload the code with the Keyestudio V4.0 board(remove external sensors, modules or other electronic components).

2. Make sure the BT module is not interfaced with the expansion board when uploading the code, because it is connected to the RX\TX pins.

If the BT module is connected to the expansion board when uploading the code, the code may not uploaded.