Kit for arduino

keyestudio

Dec 18, 2023

KS0538 KS0539 KEYESTUDIO 2021 COMPLETE STARTER LEARNING KIT

CHAPTER

ONE

INTRODUCTION

Do you want to learn about programming?

As long as you are passionate about science and dare to explore new things, this kit is surely the best choice for you.

The kit is used to Arduino-based Scratch graphical programming and C language.

You can create numerous fascinating experiments with the PLUS mainboardsensorsmodules and electronic components.

As many as 32 project tutorials are provided, which contain detailed wiring diagrams, component knowledge, test code, and so on.

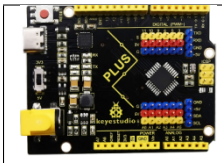
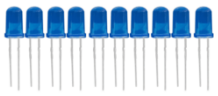
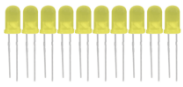
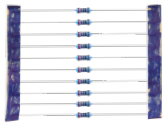
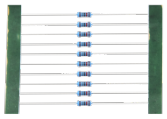
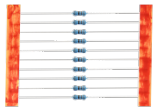
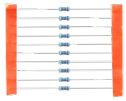
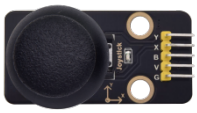
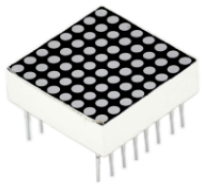
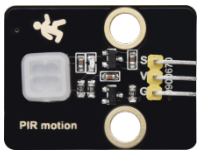
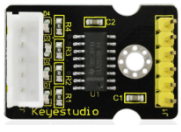
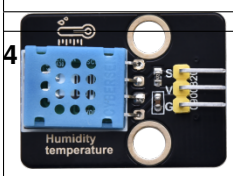
In addition, you can master the use of electronics, physics, science and programming by building up experiment with this kit.

CHAPTER

TWO

KIT LIST

(KS0538 includes Plus mainboard, KS0539 does not include Plus mainboard.)

				
Plus Mainboardx1	Blue LEDx10	Red LEDx10	Yellow LEDx10	Green LEDx10
				
RGBx1	220 Resistorx10	10K Resistorx10	1K Resistorx10	4.7K Resistorx10
				
10K Potentiometerx1	Active Buzzerx1	Passive Buzzerx1	Buttonx4	Tilt Switchx1
				
Photoresistorx2	Flame Sensorx1	10K thermistorx1	Yellow Capx4	IC 74HC595N x1
				
Joystick Modulex1	1-Digit Tube Displayx1	4-Digit Tube Displayx1	8x8 Dot Matrix Display x1	IC L293Dx1
				
LCD_128X32_DOT x1	IR Receiverx1	ESP8266WIFI ESP-01x1	PIR Motion Sensorx1	Fanx1
				
DC Motorx1	USB to ESP-01S WIFI Module Shieldx1	Stepper Motor Driver Boardx1	Stepper Motorx1	IR Remote Controlx1
				

CHAPTER

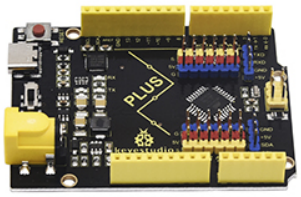
THREE

KEYESTUDIO PLUS BOARD

The Keyestudio PLUS motherboard, fully compatible with the Arduino IDE control board, which is the core of this kit

It incorporates all functions of Arduino UNO R3. Additionally, more improvements on the PLUS board makes it powerful.

It is your best choice to build up circuits and programme.



VS

Keyestudio PLUS

Arduino UNO R3

1. USB CP2102, stability and compatibility are better forto-turn chip		1. USB turnstile chip is 16 U2, Some systems are not compatible
2. Working voltage can be selected 3.3 V or 5 V, can connect 3.3 V sensor		2. Only 5 V working voltage, no way to connect 3.3 V sensor
3. Two more IO mouth, A6,A7, the best		3. Design did not leave these 2 IO ports, defective
4. Extended serial communication and I2C interface, can be easily connected to similar devices		4. No expansion port, more complex connection required
5. Unique DC-DC power supply design, working voltage 5 V, the current of 2 A, can directly drive some high current load, such as steering gear and motor		5. Working voltage 5 V, the current only 1 A, can not drive the equipment with high current.
6. Extension of 6 PWM ports and 6 analog port interfaces to connect sensors directly		6. No expansion port, more complex connection required
7. Extended serial communication and I2C interface, can be easily connected to similar devices		7. No expansion port, more complex connection required
8. Input voltage 6-15 V, wider voltage range, choose more		8. Input voltage 7-12 V, optional power supply mode is not
9. Choose the current more popular type-c interface, beautiful and generous, faster transmission speed		9. Adopt traditional square USB interface, more common

Specifications

Microcontroller: ATMEGA328P-AU

USB to serial chip: CP2102

Working voltage: DC 5V or 3.3V (DIP switch control)

External power supply: DC 6V to 15V (9V is recommended.)

Digital I/O pins: 14 (D0 to D13)

PWM channel: 6 (D3 D5 D6 D9 D10 D11)

Analog input channel (ADC): 8 (A0 to A7)

Each I/O port of DC output capacity: 20 mA

Output capacity of 3.3V port: 50 mA

Flash Memory: 32 KBof which the bootloader uses 0.5 KB

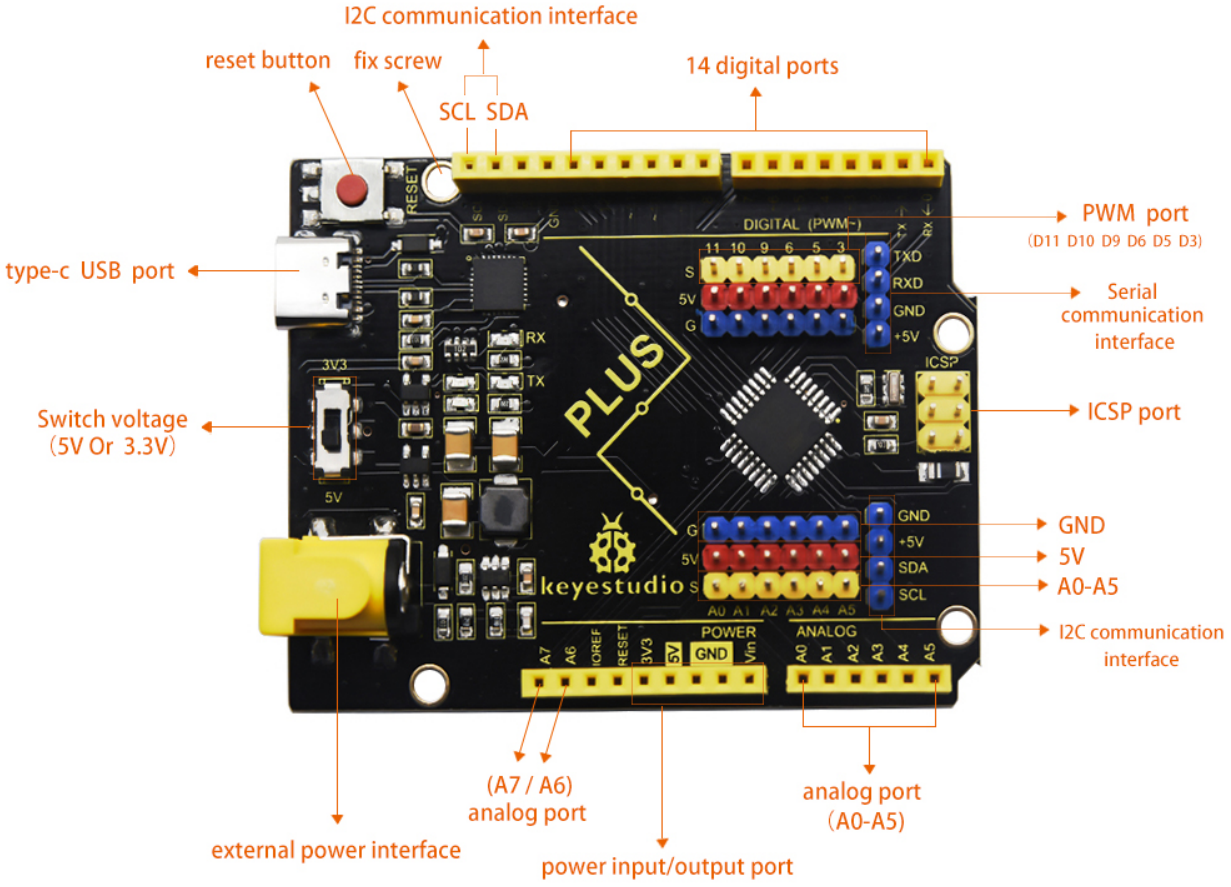
SRAM: 2 KB (ATMEGA328P-AU)

EEPROM:1 KB (ATMEGA328P-AU)

Clock speed: 16MHz

On-board LED pin: D13

Pinout



Specialized Functions of Pins:

- Serial communication interface: D0 is RX, D1 is TX
- PWM interface (pulse width modulation): D3 D5 D6 D9 D10 D11
- External interrupt interface: D2(interrupt 0) and D3 (interrupt 1)

- SPI communication interface: D10 is SS, D11 is MOSI, D12 is MISO, D13 is SCK
- IIC communication port: A4 is SDA, A5 is SCL

Note: The all experiments of this learning kit, the DIP switch on the Keyestudio PLUS control board is turned to the 5V terminal by default.

CHAPTER

FOUR

HOW TO DOWNLOAD ARDUINO,DRIVERS AND LIBRARY FILES

media/ide.png

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

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

CHAPTER

FIVE

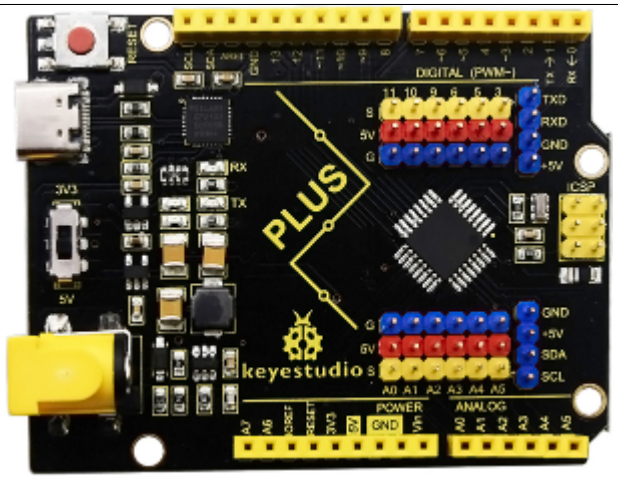
PROJECTS

5.1 Project 1: Hello World

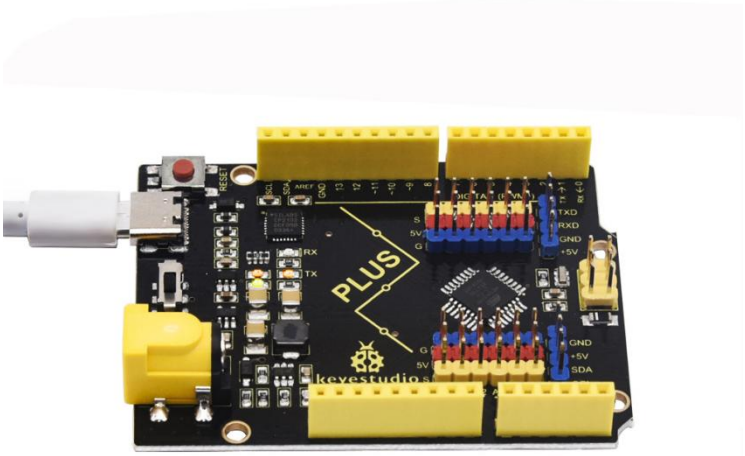
Introduction

For Arduino beginners, we will start with some simple things. In this project, you only need a PLUS mainboard and a USB cable to complete the "Hello World!" project.

Components Required

	
Keyestudio PLUS Mainboardx1	USB Cablex1

Connection



Code

Arduino uses a serial monitor to display information such as print statements and sensor data. This is a very powerful tool for debugging long code. Let's first learn the"if"statement, which is a control structure in Arduino programming.

```
/*

    Keyestudio 2021 Starter Kit

    Project 1

    Hello World

    http//www.keyestudio.com

    */

    char val;// defines variable "val"

    void setup()

    {

    Serial.begin(9600);// sets baudrate to 9600

    }

    void loop()

    {
```

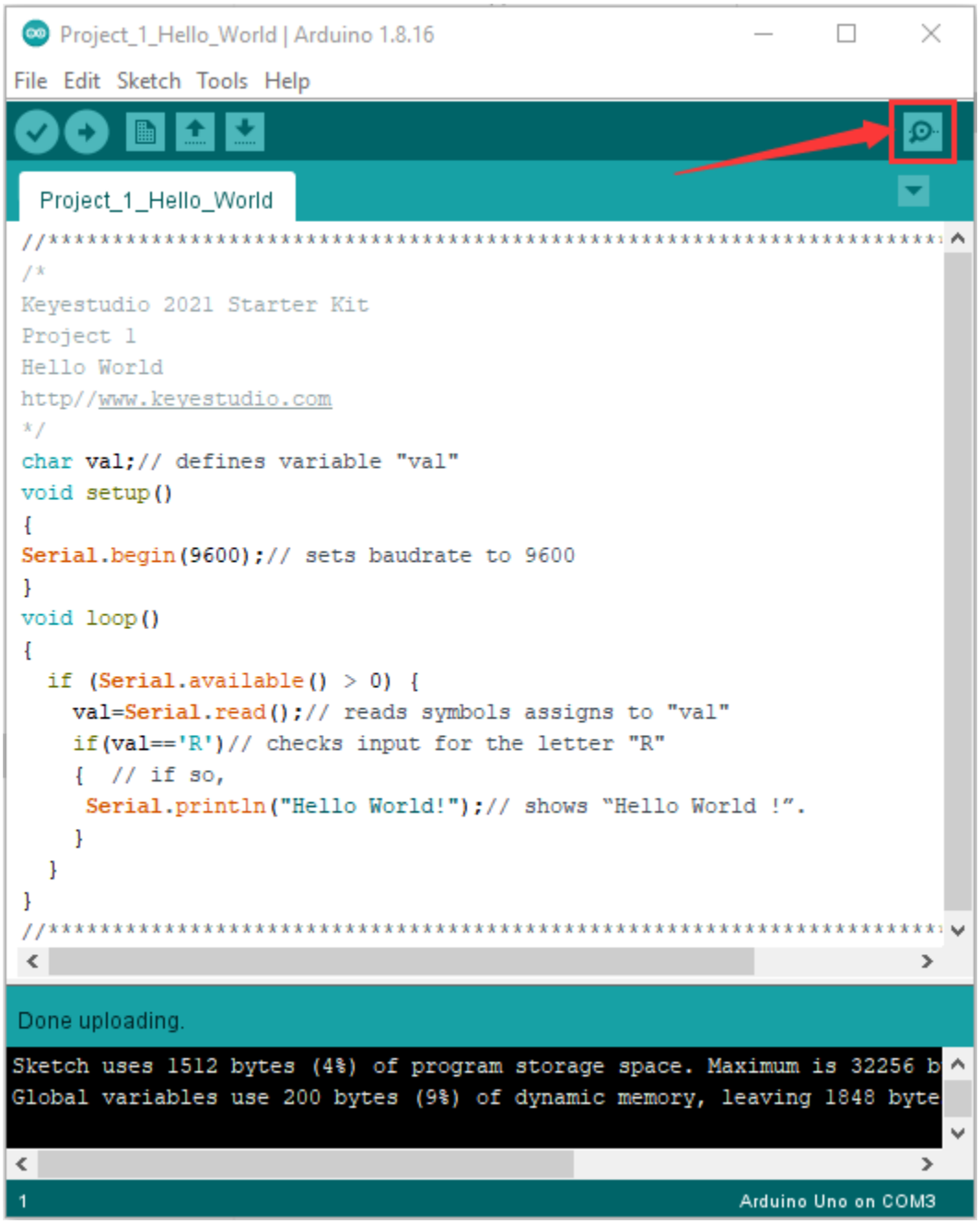
(continues on next page)

(continued from previous page)

```
if (Serial.available() \> 0) {

val=Serial.read();// reads symbols assigns to "val"

if(val=='R')// checks input for the letter "R"

{ // if so,

Serial.println("Hello World!");// shows “Hello World !”.

}

}

}
```

Result

Select the correct Arduino IDE mainboard type and COM port, and click the **button**  on the Arduino IDE to upload the code. After successfully uploading, click the icon  to enter the serial display.



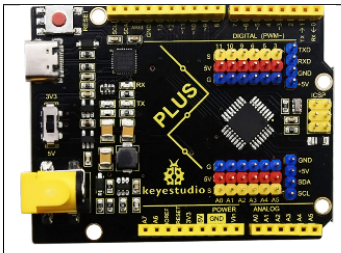
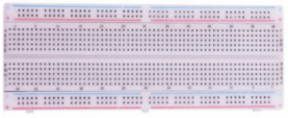
Whenever you enter an “R” in the text box and click “Send”, the serial monitor will display “Hello World!”.

5.2 Project 2: LED Blinking

Introduction

In this project, we will show you the LED flashing effect through Arduino's digital pins.

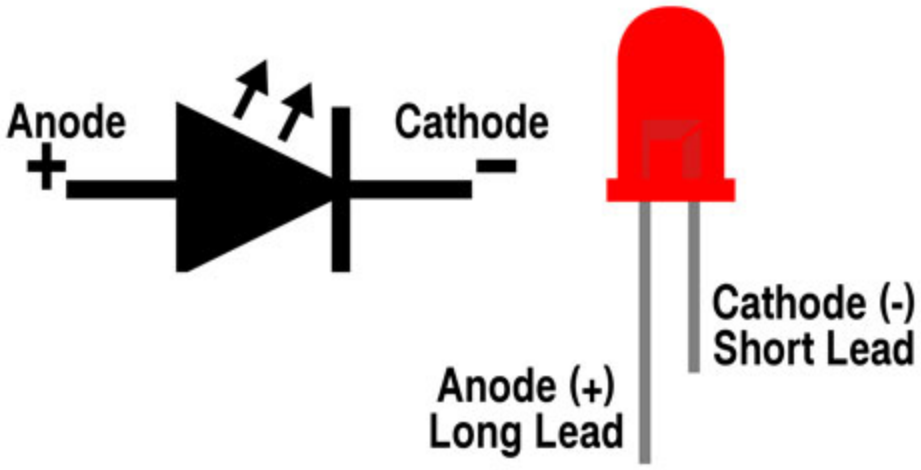
Components Required

					
Keyestudio PLUS Mainboardx1	Red LEDx1	220 Resistorx1	Breadboardx1	Jumper Wirex2	USB Cablex1

Component Knowledge

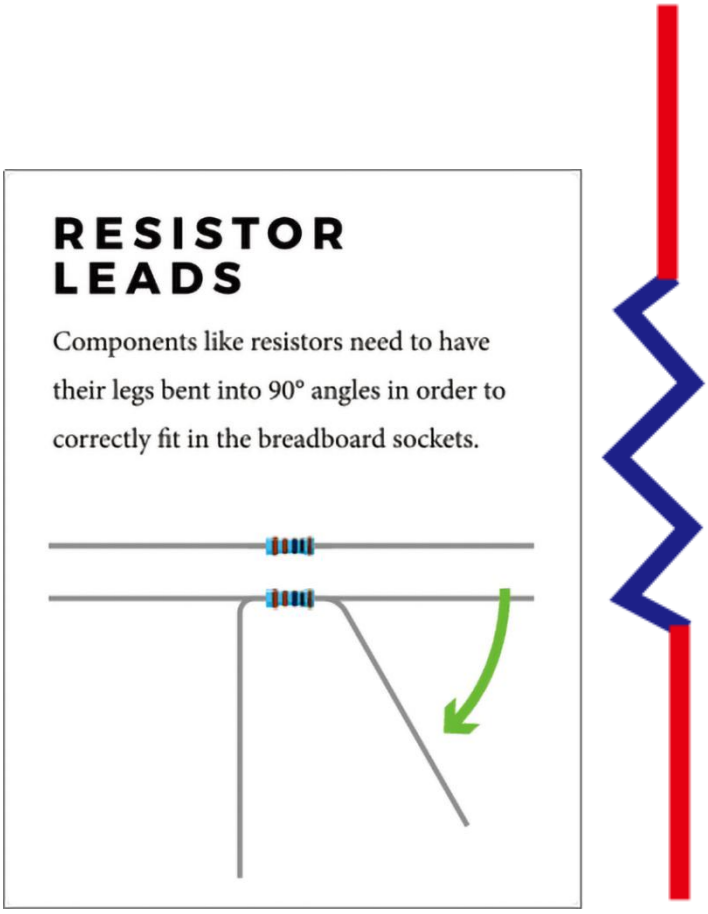
LED:

It is a kind of semiconductor called "light-emitting diode", which is an electronic device made of semiconductor materials (silicon, selenium, germanium, etc.). It has an anode and a cathode. The short lead (cathode) is grounded. The long lead (anode) is connected to 5V.



Resistor

A resistor is an electronic component in a circuit that restricts or regulates the flow current flow. Its unit is().



We can use resistors to protect sensitive components, such as LEDs. The strength of the resistance is marked on the body of the resistor with an electronic color code. Each color code represents a number, and you can refer to it in a resistance card.

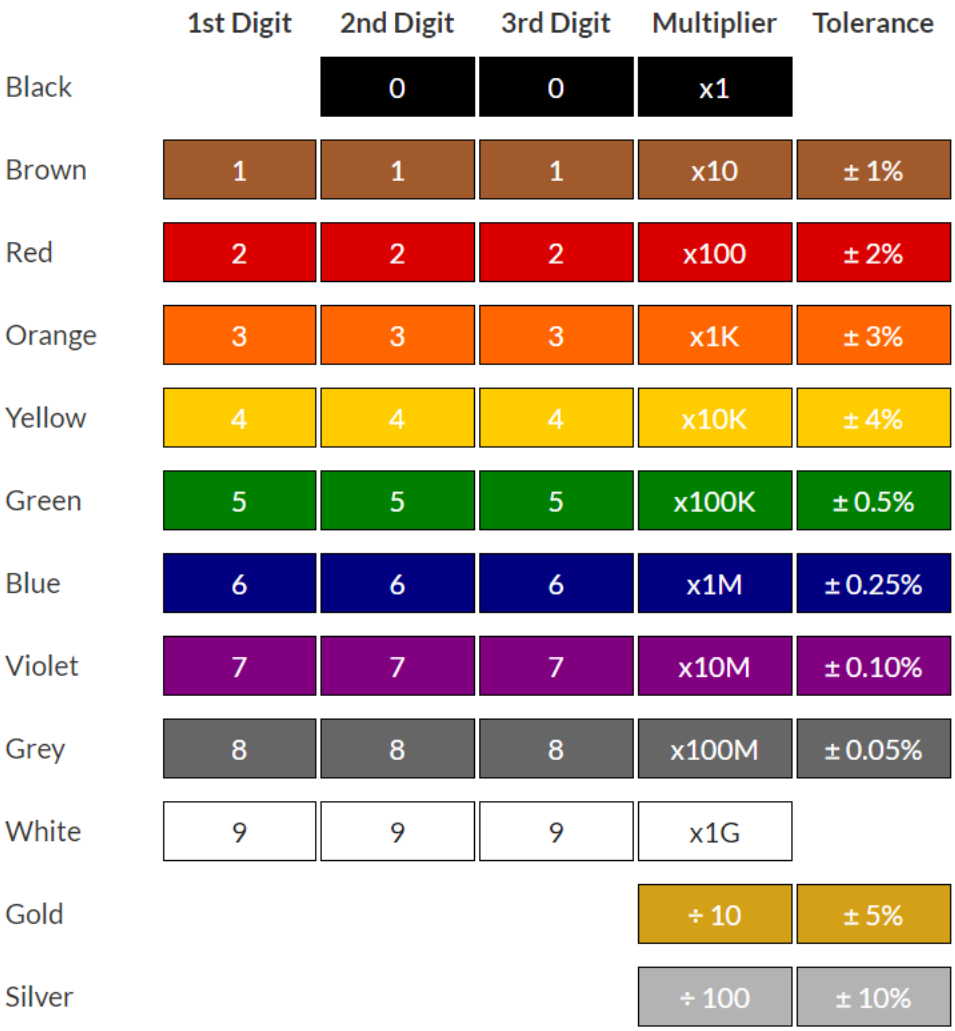
-Color 1 – 1st Digit

-Color 2 – 2nd Digit

-Color 3 – 3rd Digit

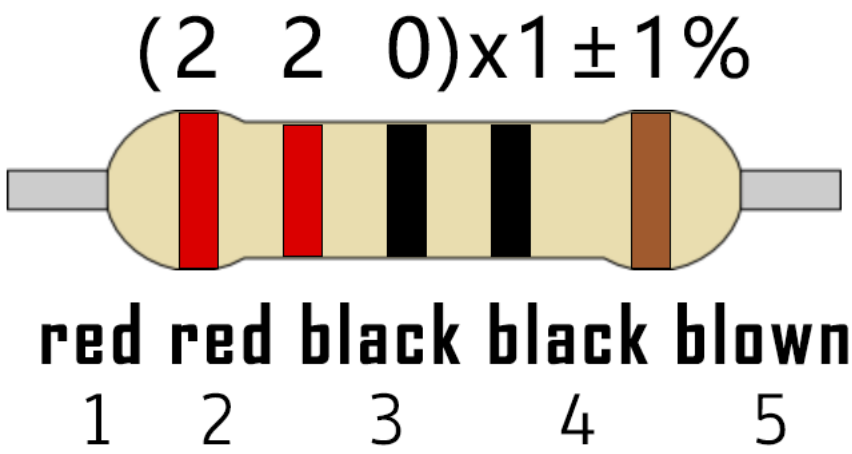
-Color 4 – Multiplier

-Color 5 – Tolerance

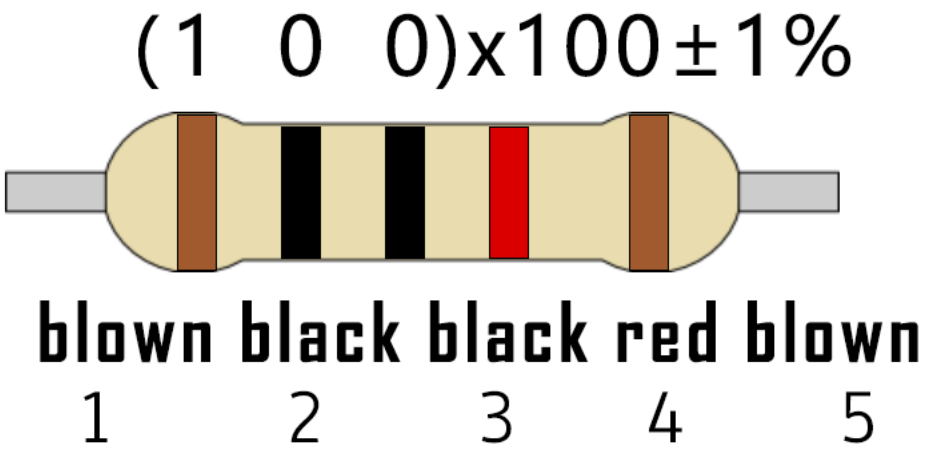


In this kit, we provide eight 5-band resistors with different resistance values. Take three 5-band resistors as an example.

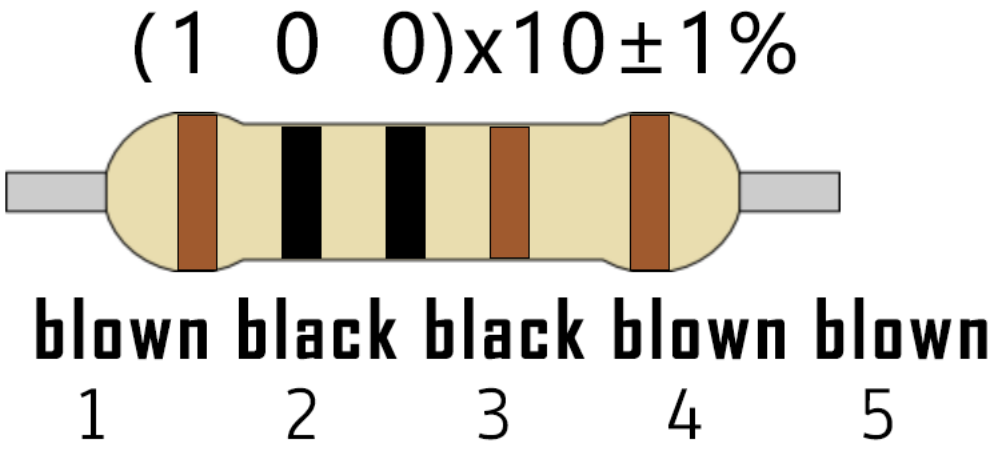
220 resistorx10



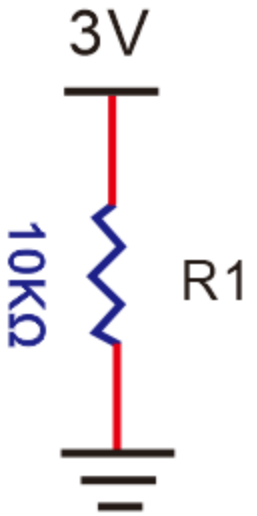
10K resistorx10



1K resistorx10



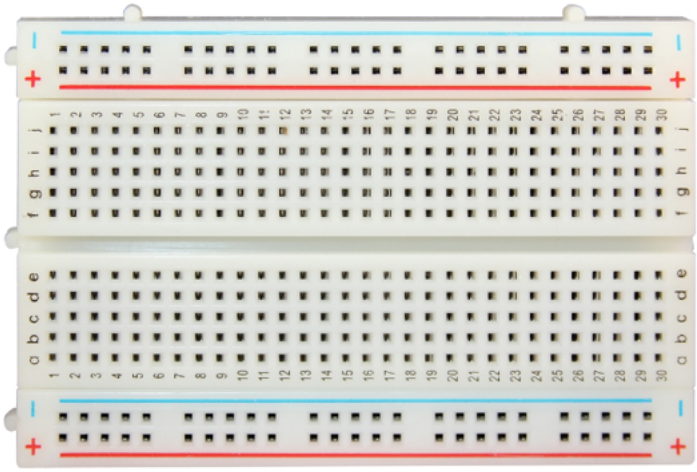
The connection between current, voltage, and resistance can be expressed by the formula: I=U/R.In the figure below, if the voltage is 3V, the current through R1 is: I = U / R = 3 V / 10 K= 0.0003A= 0.3mA.



Do not directly connect resistors with very low resistance to the two poles of the power supply, as this will cause excessive current to damage the electronic components. Resistors do not have positive and negative poles.

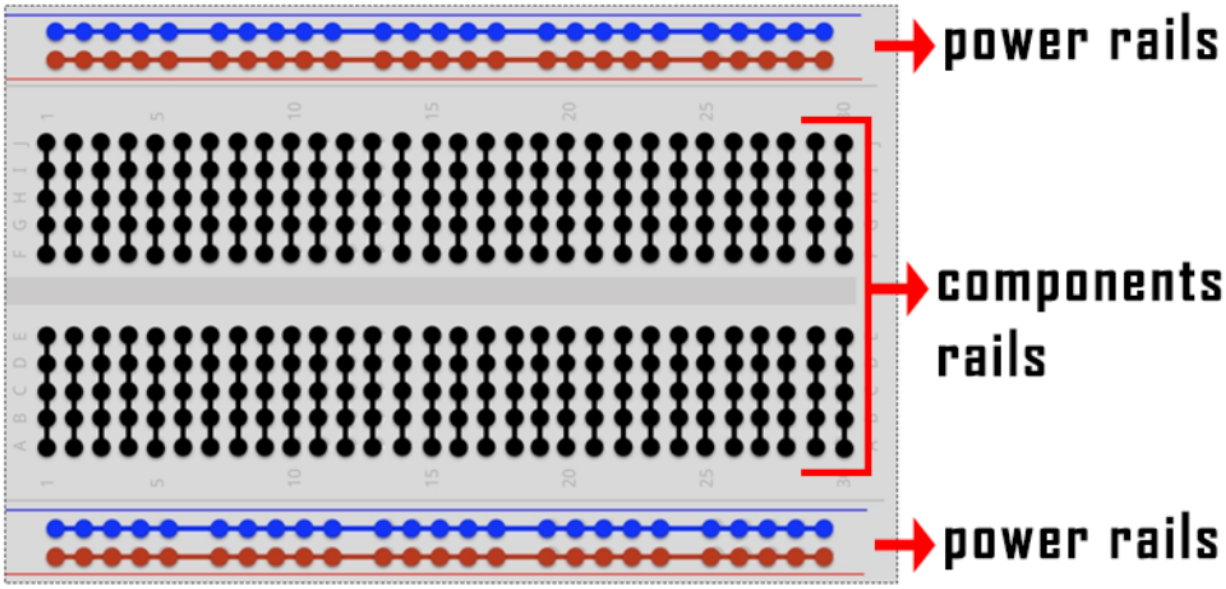
Breadboard

A breadboard is used to build and test circuits quickly before finalizing any circuit design. The breadboard has many holes into which circuit components like integrated circuits and resistors.A typical breadboard is as follows.

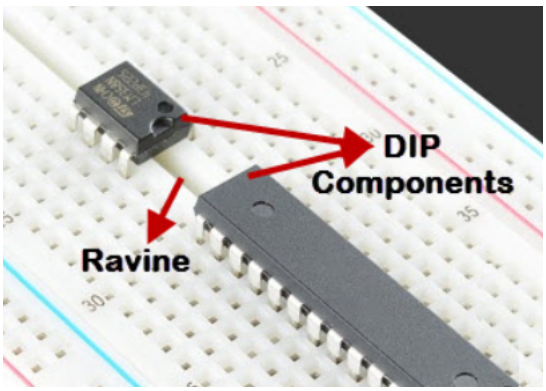


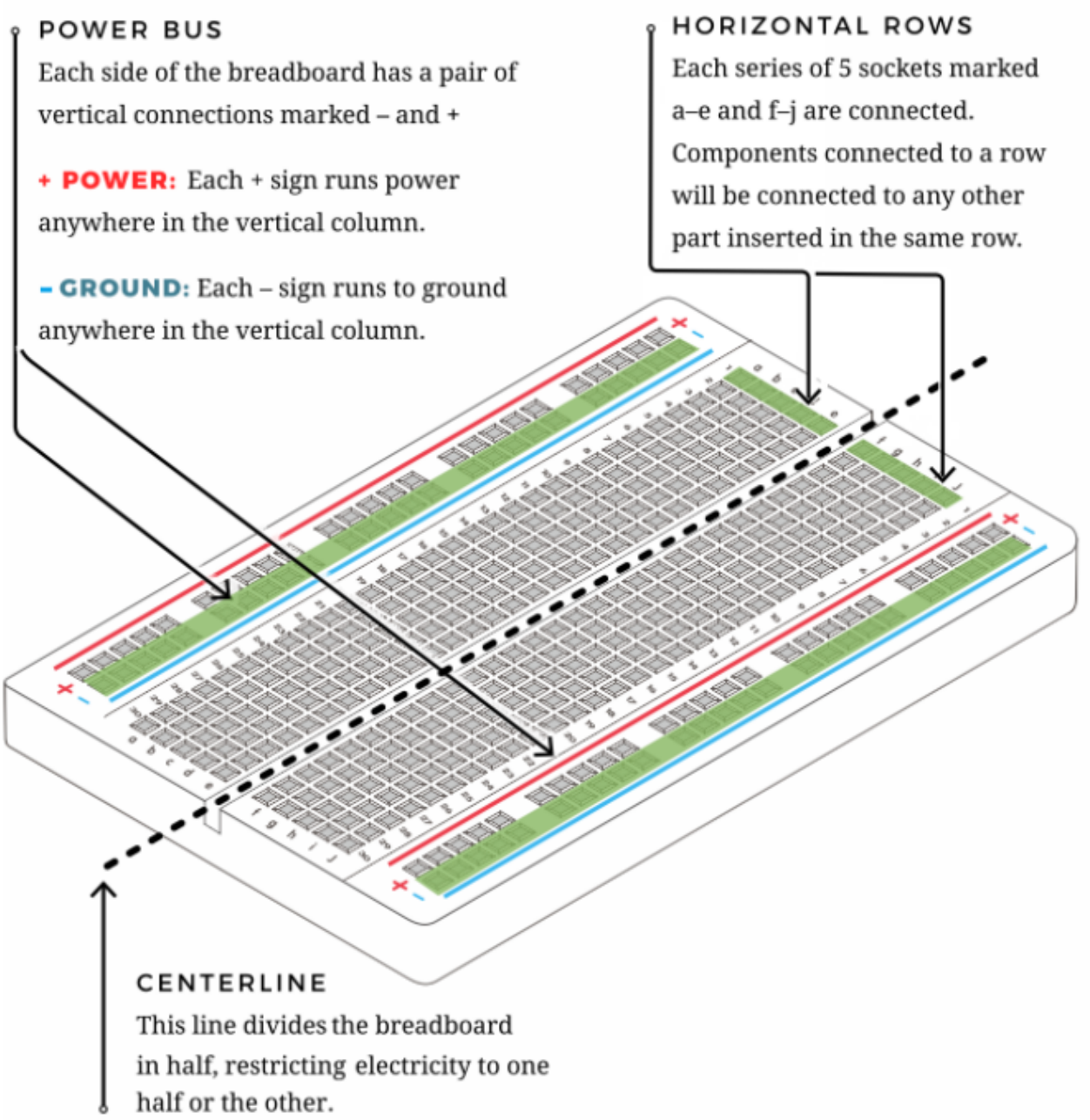
The bread board has strips of metal which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally while the remaining holes are connected vertically.

The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive (+) and negative (-) terminals of the power supply, respectively. The conductive layout of the breadboard is shown in the following diagram.



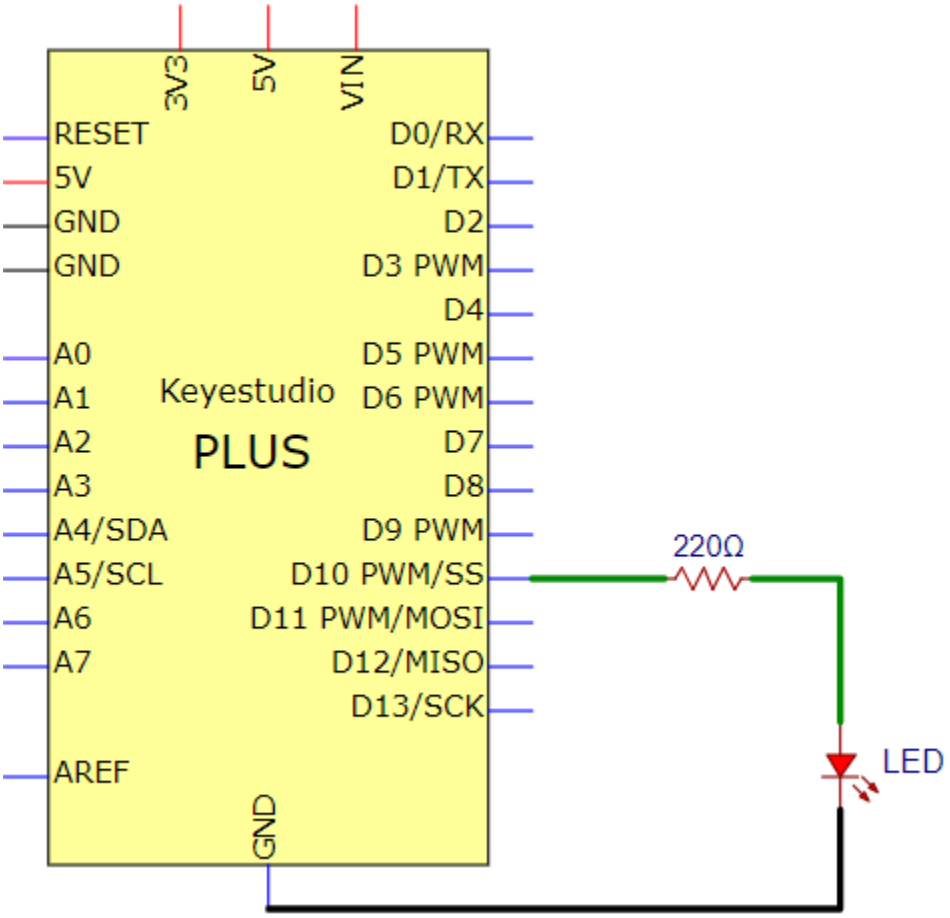
When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the figure below.



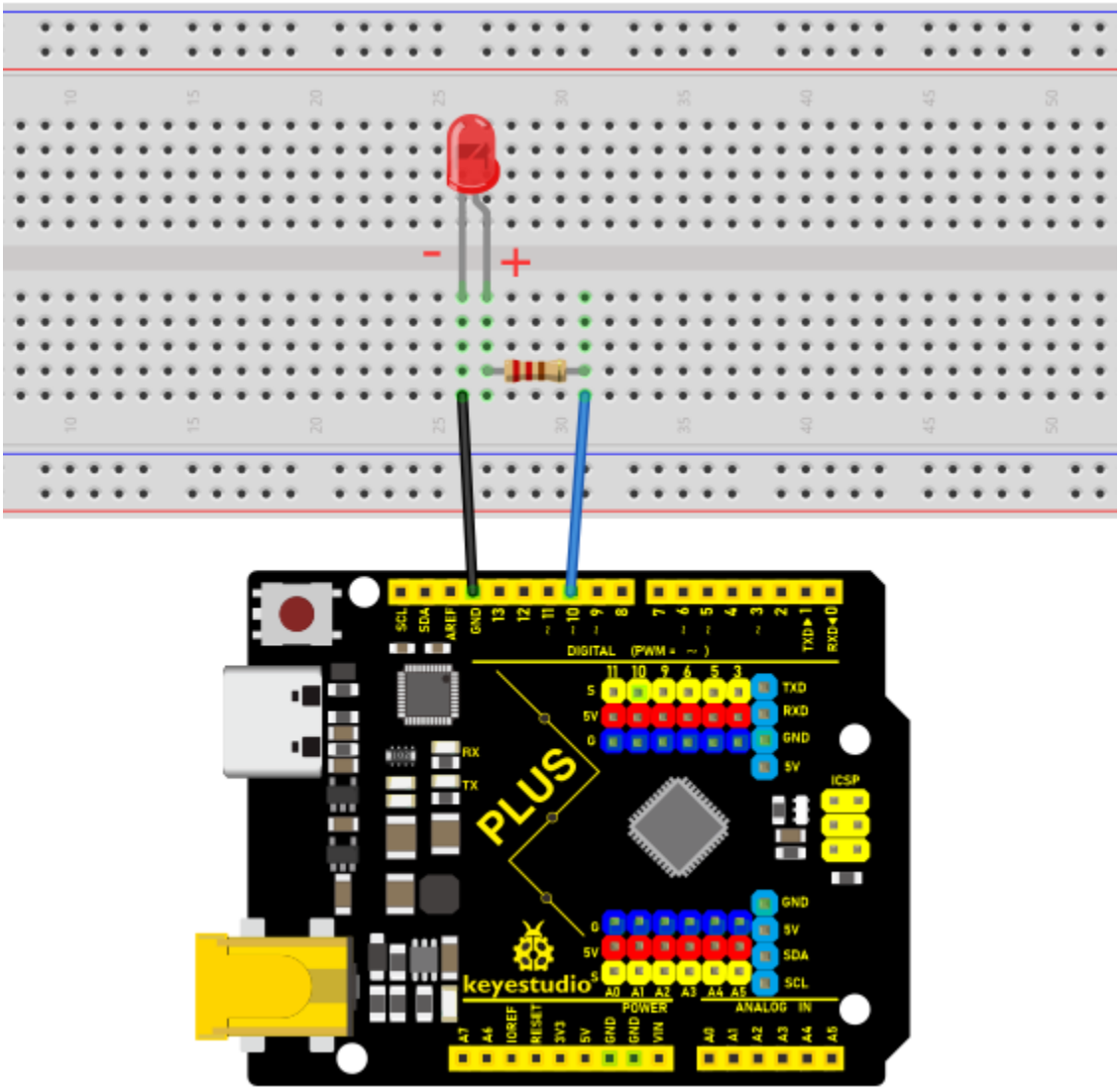


Circuit Diagram and Wiring Diagram

As shown in the diagram, we use digital pin 10 and connect one LED to a 220 ohm resistor to avoid high current damage to the LED.



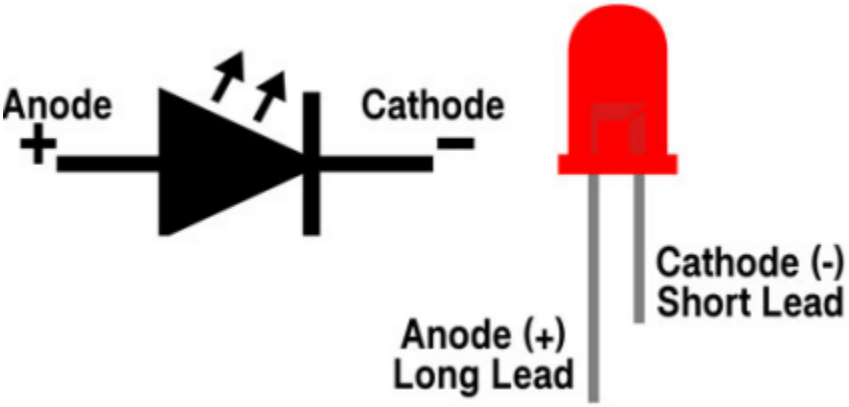
Circuit Diagram



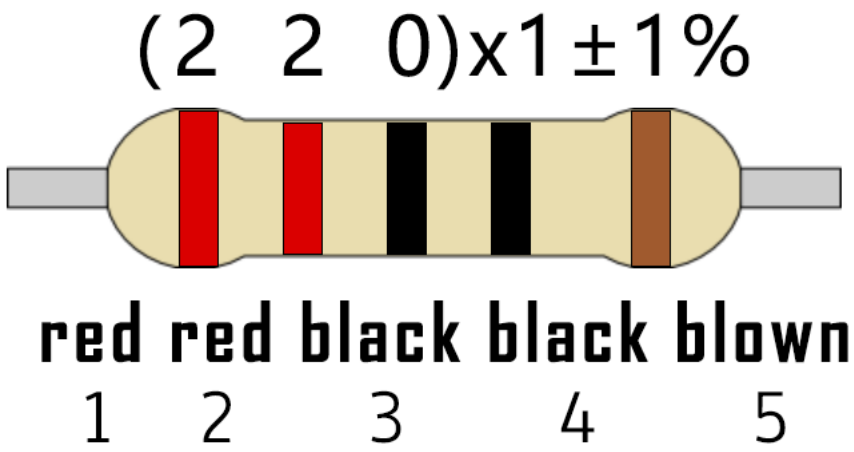
Wiring Diagram

Note:

How to connect an LED



How to identify the 220 five-band resistor



Code

```
/*

Keyestudio 2021 Starter Kit

Project 2

LED_Blinking

http//www.keyestudio.com

*/

int ledPin = 10; // defines numeric pin 10.

void setup()

{

pinMode(ledPin, OUTPUT);// defines PIN with connected LED as output

}

void loop()

{

digitalWrite(ledPin, HIGH); // turn on LED

delay(1000); // wait a second.

digitalWrite(ledPin, LOW); // turn off LED

delay(1000); // wait a second

}
```

Result

Upload the project code, wire up components according to the wiring diagram, and power on. The LED will blink.

Explanation

pinMode(ledPinOUTPUT): Before using the Arduino's pins, you need to tell the control board whether it is INPUT or OUTPUT. We use a built-in function "pinMode()" to do this.

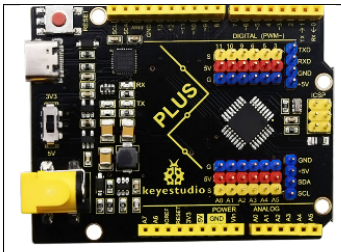
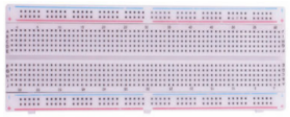
digitalWrite(ledPinHIGH) : When using a pin as an OUTPUT, it can be commanded as HIGH (output 5V) or LOW (output 0V).

5.3 Project 3: Breathing Led

Introduction

In this project, we will learn the PWM control of ARDUINO. PWM is Pulse Width Modulation, which is a technique that encodes analog signal levels into digital signal levels. We will use PWM to control the brightness of LED.

Components Required

					
Keyestudio Plus Mainboardx1	Red LEDx1	220 Resistorx1	Breadboardx1	Jumper Wirex2	USB Cablex1

Component Knowledge

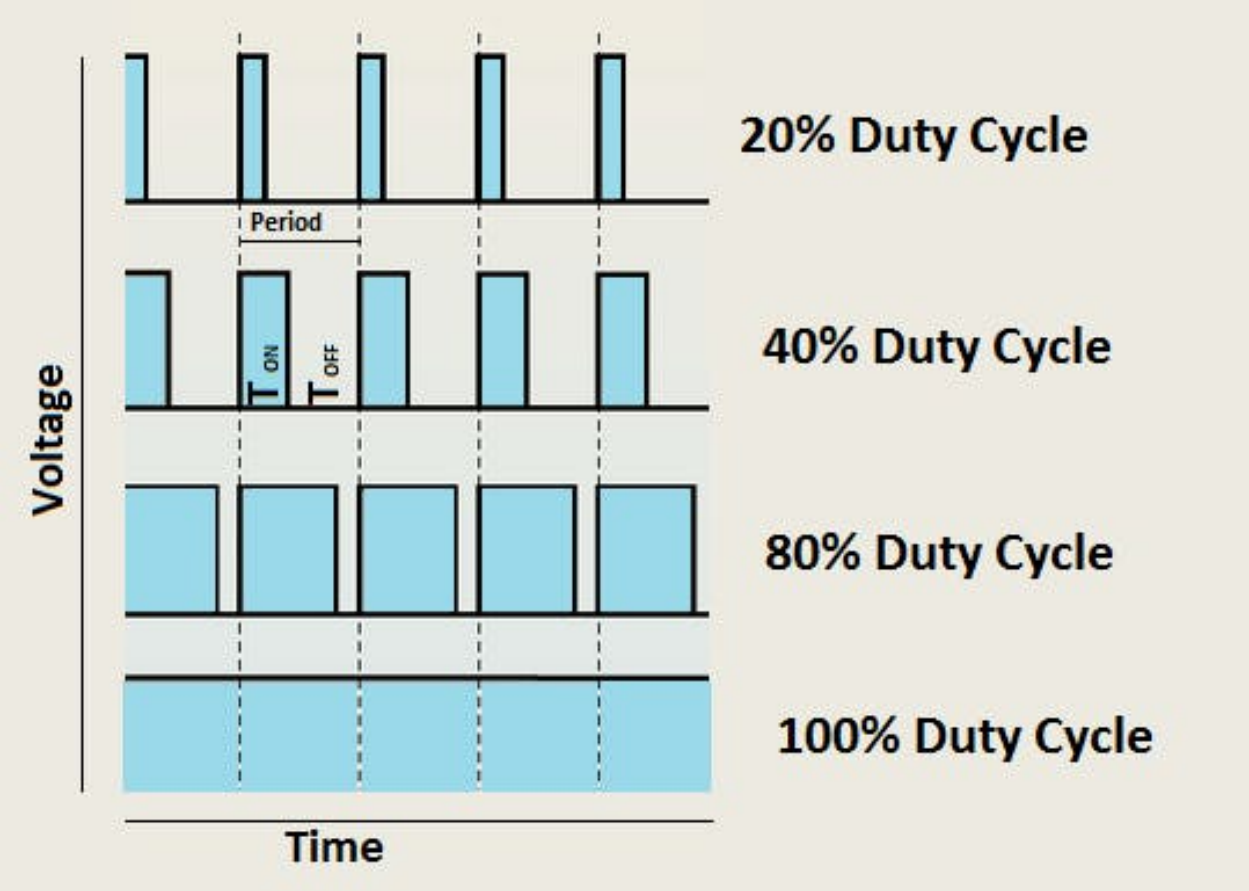
Working principle: It can control the brightness of LED, the speed of DC motors and Servo motors, and outputs square wave signal. If we want to dim the LED, we can change the ON(open) and OFF(close) time of the signal. When we change the time of ON and OFF fast enough, then the brightness of the LED will change. Here are some terms related to PWM as follows.

ON (open)When the signal is high.

OFF (close)When the signal is low.

Period: It is the sum of the time of On and Off.

Duty cycle: The percentage of time when the signal is at a high level for a certain period of time. At 50% duty cycle and 1Hz frequency, the LED will be on for half a second and off for the other half of a second.



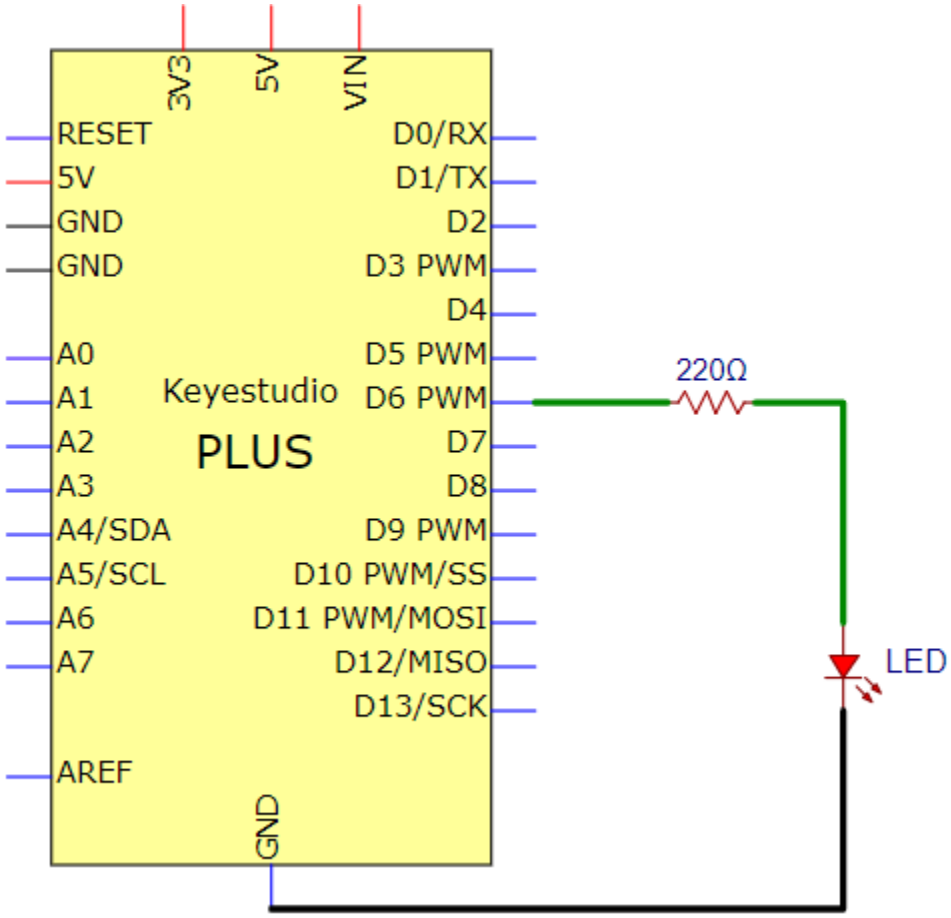
Arduino and PWM

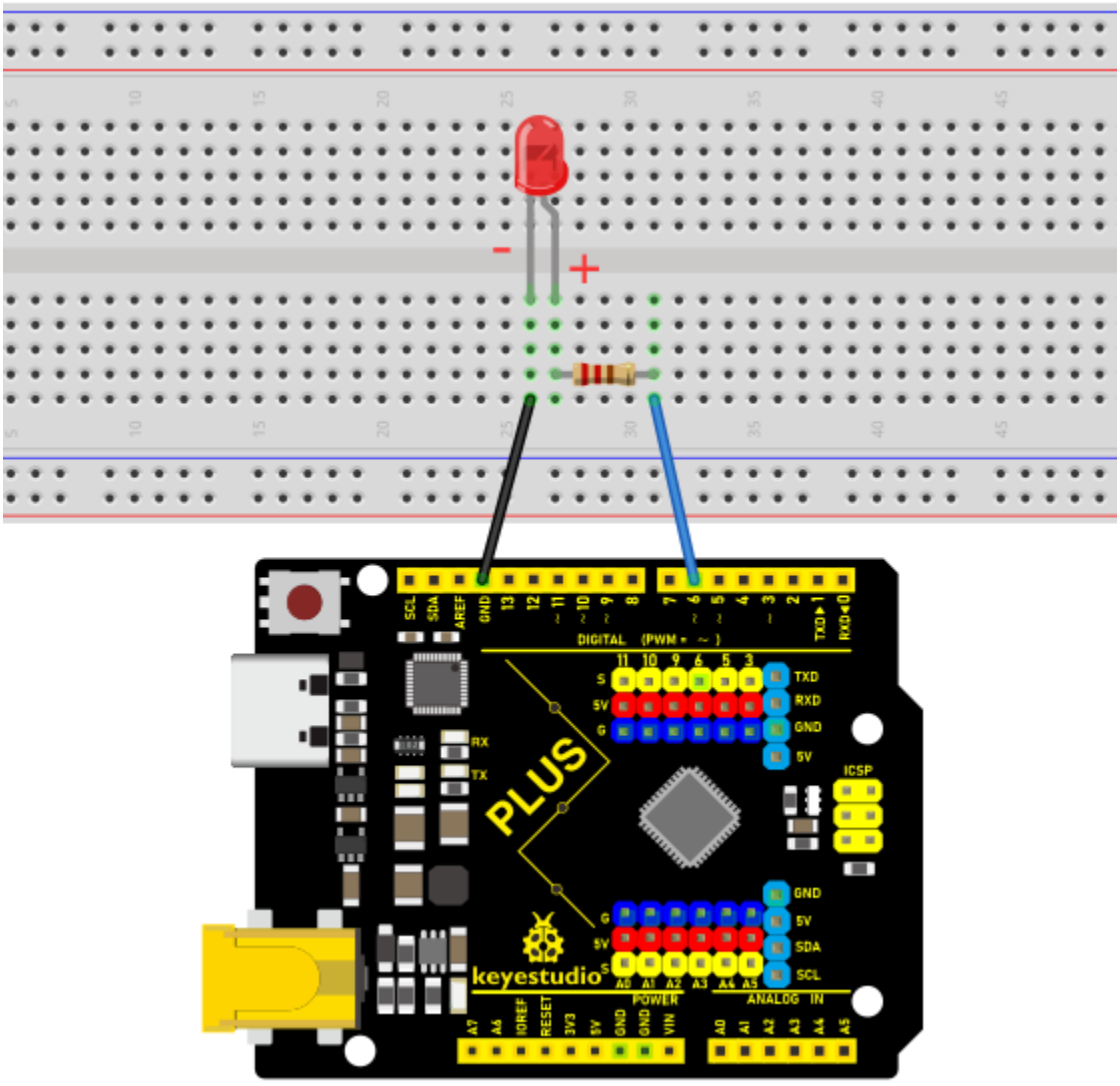
The Arduino IDE has a built-in function "analogWrite()" that can be used to generate PWM signals. Most of the pins generate signals with a frequency of about 490Hz and we can use this function to give values from 0 to 255.

"analogWrite(0)" indicates a signal with 0% duty cycle. "analogWrite(127)" indicates a signal with 50% duty cycle.

"analogWrite(255)" indicates a signal with 100% duty cycle. On the Plus control board, the PWM pins are 3, 5, 6, 9,10, and 11. PWM pins are marked with the "~"symbol. In this project, you will learn how to get the PWM output from the digital pins of the Plus control board and control the brightness of the LED by code.

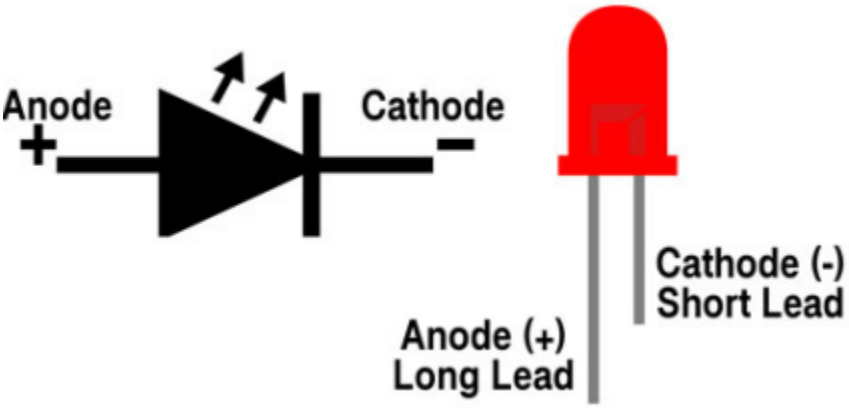
Circuit Diagram and Wiring Diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
Keyestudio
PLUS
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
220Ω
LED

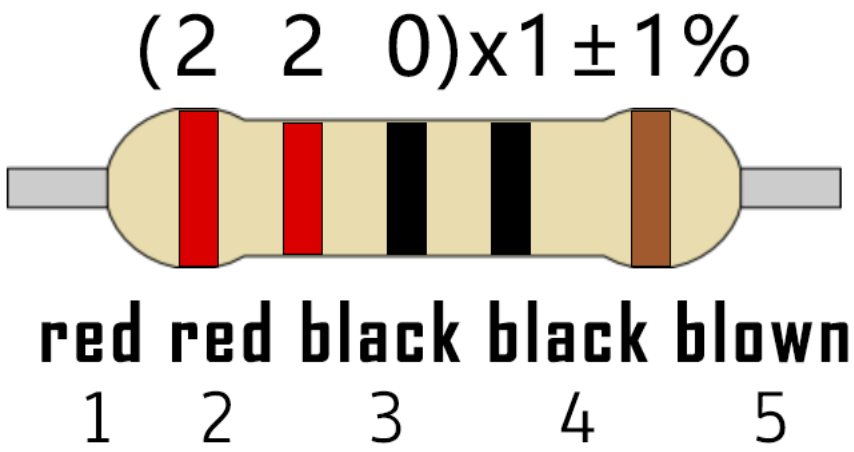


Note:

How to connect the LED



How to identify the 220 5-band resistor



Code

```
/*

Keyestudio 2021 Starter Kit

Project 3

Breathing_Led

http//www.keyestudio.com

*/

int ledPin = 6;

void setup()

{

pinMode(ledPin,OUTPUT);

}

void loop(){

for (int value = 0 ; value \< 255; value=value+1){

analogWrite(ledPin, value);

delay(5);

}

for (int value = 255; value \>0; value=value-1){

analogWrite(ledPin, value);
```

(continues on next page)

(continued from previous page)

```
delay(5);

} }
```

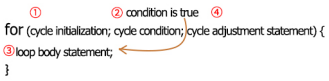
Result

After burning the project code, connecting the wires according to the wiring diagram, and powering on, the LED lights up gradually, and then gradually darkens.

Explanation

When we need to execute a sentence repeatedly, we can use the "for" statement.

The "for"statement format is as follows:



The loop sequence of"for"statement is as follows:

Round 1: 1 → 2 → 3 → 4

Round 2: 2 → 3 → 4

…

Until 2 does not hold and the "for" statement loop ends. Knowing this sequence, go back to the code.

for (int value = 0; value < 255; value=value+1){

…}

for (int value = 255; value >0; value=value-1){

…}

These two for statements, which realize that the value can continuously increases from 0 to 255, then decreases from 255 to 0. The loop continues indefinitely.

In the "for"statement, involving a new function "analogWrite()".

We know that the digital port has only two states, 0 and 1. How to send an analog value to a digital pin? This function will be used. Take a look at the Arduino board and look at the digital pins. You will find that 6 of the pins are marked with "~". These pins are different from other pins in that they can output PWM signals.

The format as follow:

analogWrite(pin,value)

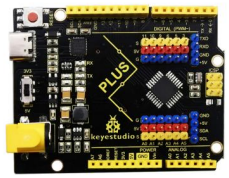
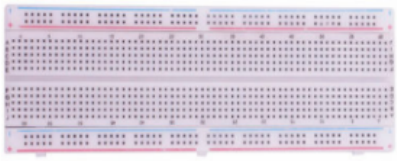
The "analogWrite()" function is used to input an analog value from 0 to 255 to the PWM port. Therefore, the value is between 0 and 255. Note that the "analogWrite()" function can only write to the digital pins with PWM function, that is, the 3, 5, 6, 9, 10, and 11 pins.

5.4 Project 4: Traffic Light

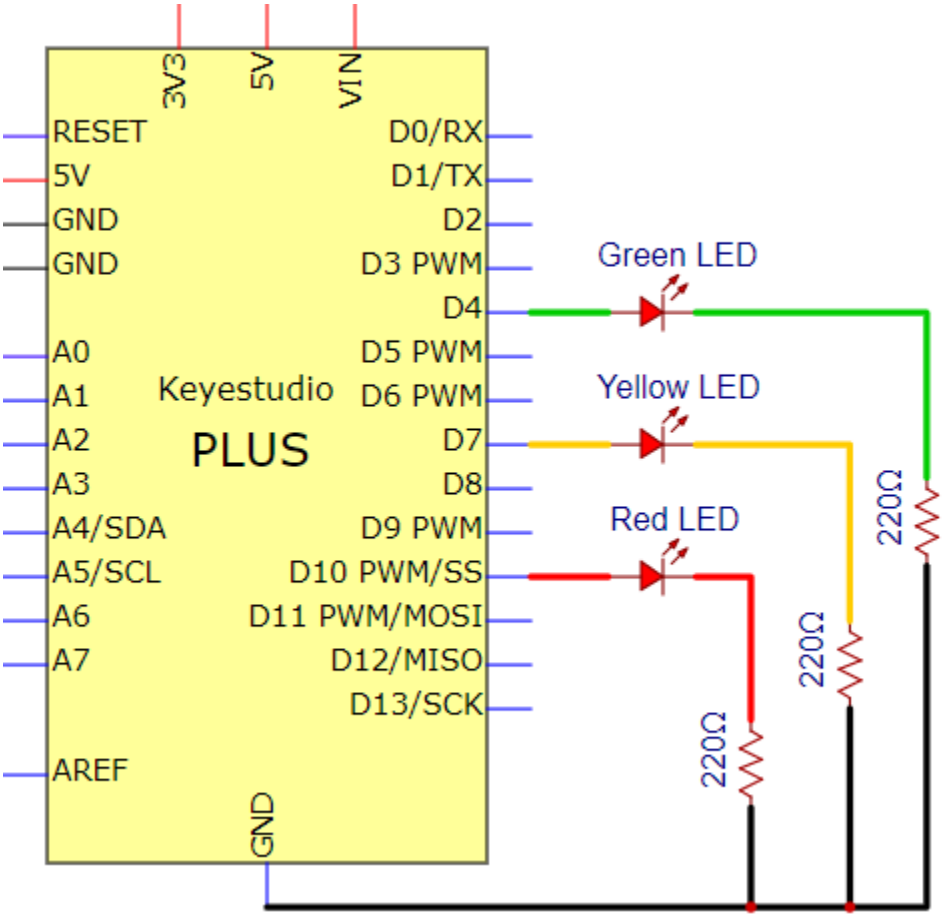
Introduction

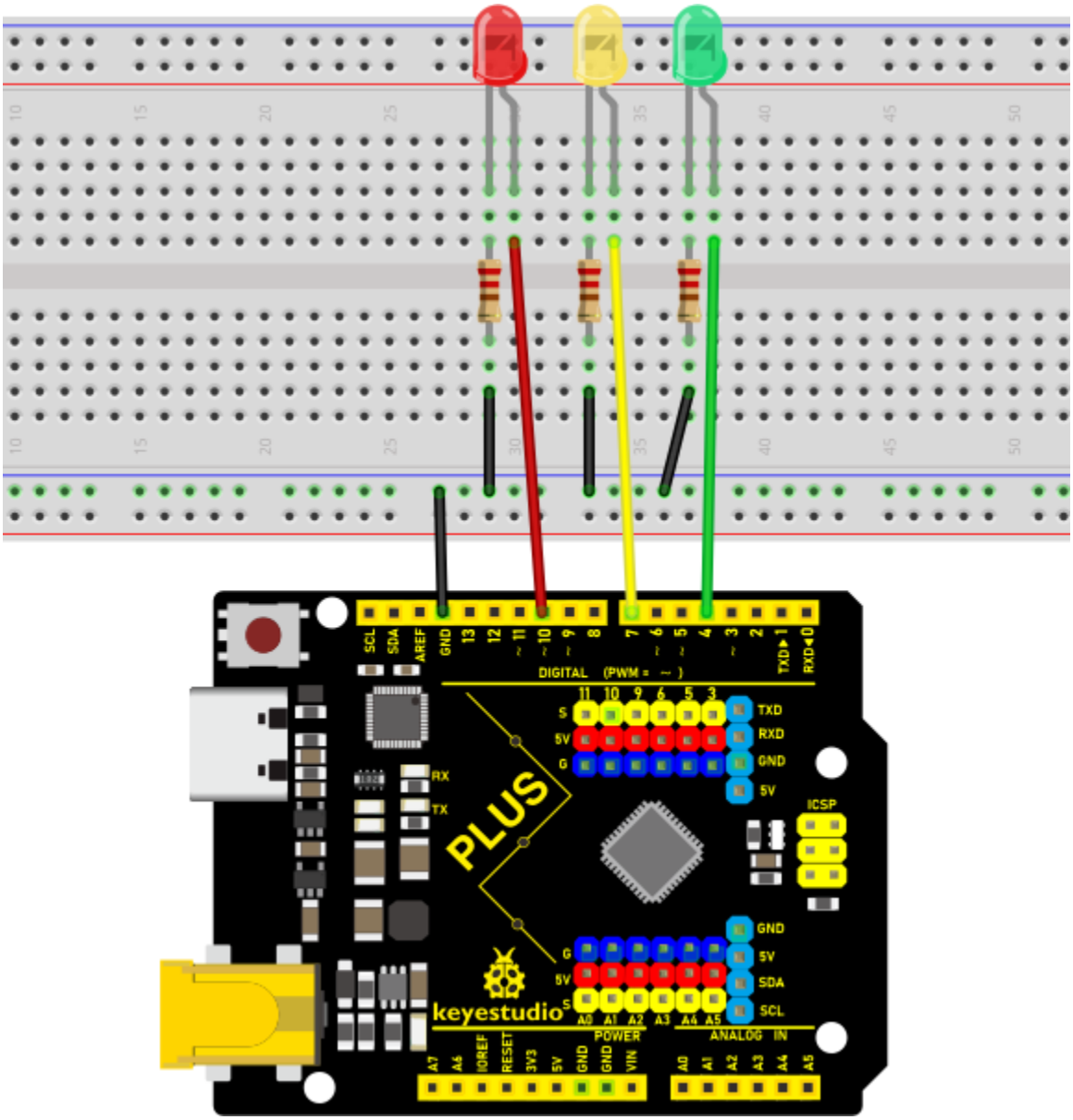
Traffic lights are closely related to people's daily lives. Traffic lights generally show red, yellow, and green. Everyone should obey the traffic rules, which can avoid many traffic accidents. In this project, we will use a PLUS board and some LEDs (red, green and yellow) to simulate the traffic lights.

Components Required

			
Keyestudio Plus Mainboardx1	Red LEDx1	Yellow LEDx1	Green LEDx1
			
USB Cablex1	220 Resistorx3	Breadboardx1	Jumper Wires

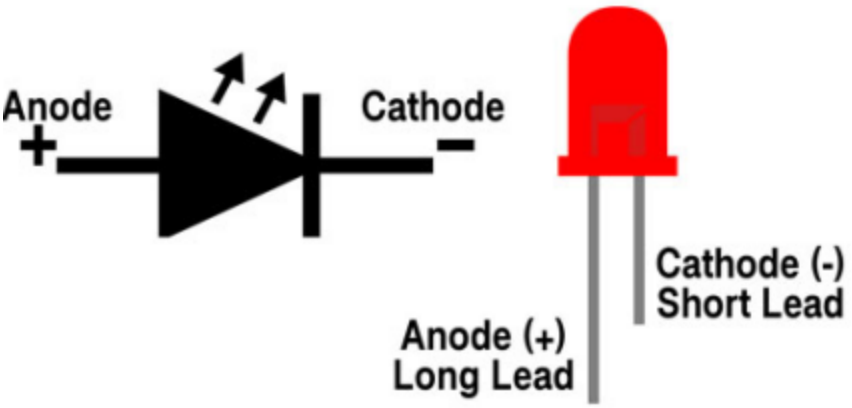
Circuit Diagram and Wiring Diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
Green LED
Yellow LED
Red LED
220Ω
220Ω
220Ω

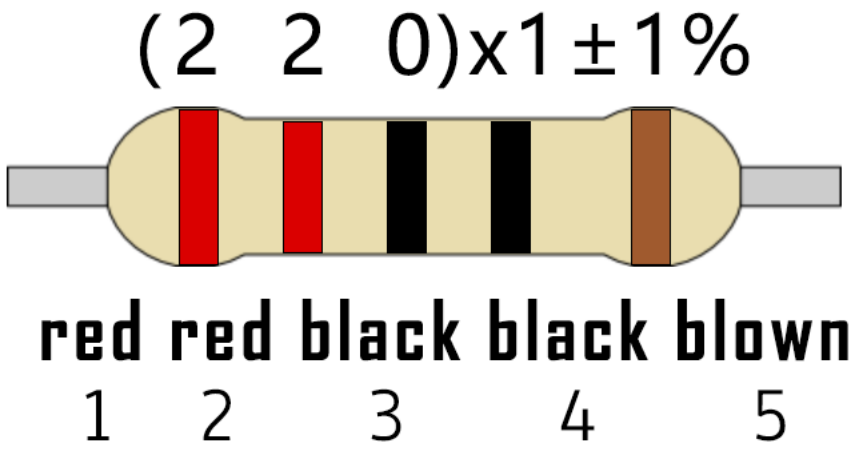


Note:

How to connect an LED



How to identify the 220 5-band resistor



Code

The flashing time of each LED should be the same as the traffic lights. In this program, we use "Arduino delay ()" to control the delay time.

```
/*

Keyestudio 2021 Starter Kit

Project 4

Traffic_Light

http//www.keyestudio.com

*/

int redled =10; // initializes digital PIN 10

int yellowled =7; // initializes digital PIN 7

int greenled =4; // initializes digital PIN 4
```

(continues on next page)

(continued from previous page)

```
void setup()

{

pinMode(redled, OUTPUT);// sets digital PIN 10 to “output”

pinMode(yellowled, OUTPUT); // sets digital PIN 7 to “output”

pinMode(greenled, OUTPUT); // sets digital PIN 4 to “output”

}

void loop()

{

digitalWrite(greenled, HIGH);// turns on LED

delay(5000);// delays 5 seconds

digitalWrite(greenled, LOW); // turns off LED

for(int i=0;i\<3;i++)// flashes 3 times.

{

delay(500);// delays 0.5 second

digitalWrite(yellowled, HIGH);// turns on LED

delay(500);// delays 0.5 second

digitalWrite(yellowled, LOW);// turns off LED

}

delay(500);// delays 0.5 second

digitalWrite(redled, HIGH);// turns on LED

delay(5000);// delays 5 second

digitalWrite(redled, LOW);// turns off LED

}
```

Result

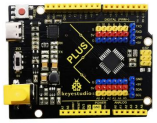
Upload the code and power on, the green LED will light up for 5s then go off. Next, the yellow one will blink for 3 times and red LED will be on for 5s then go off.

5.5 Project 5: RGB LED

Introduction

In this project, we will introduce the RGB LED and show you how to use the Plus control board to control the RGB LED. Even though RGB LED is very basic, it is also a great way to learn the fundamentals of electronics and coding.

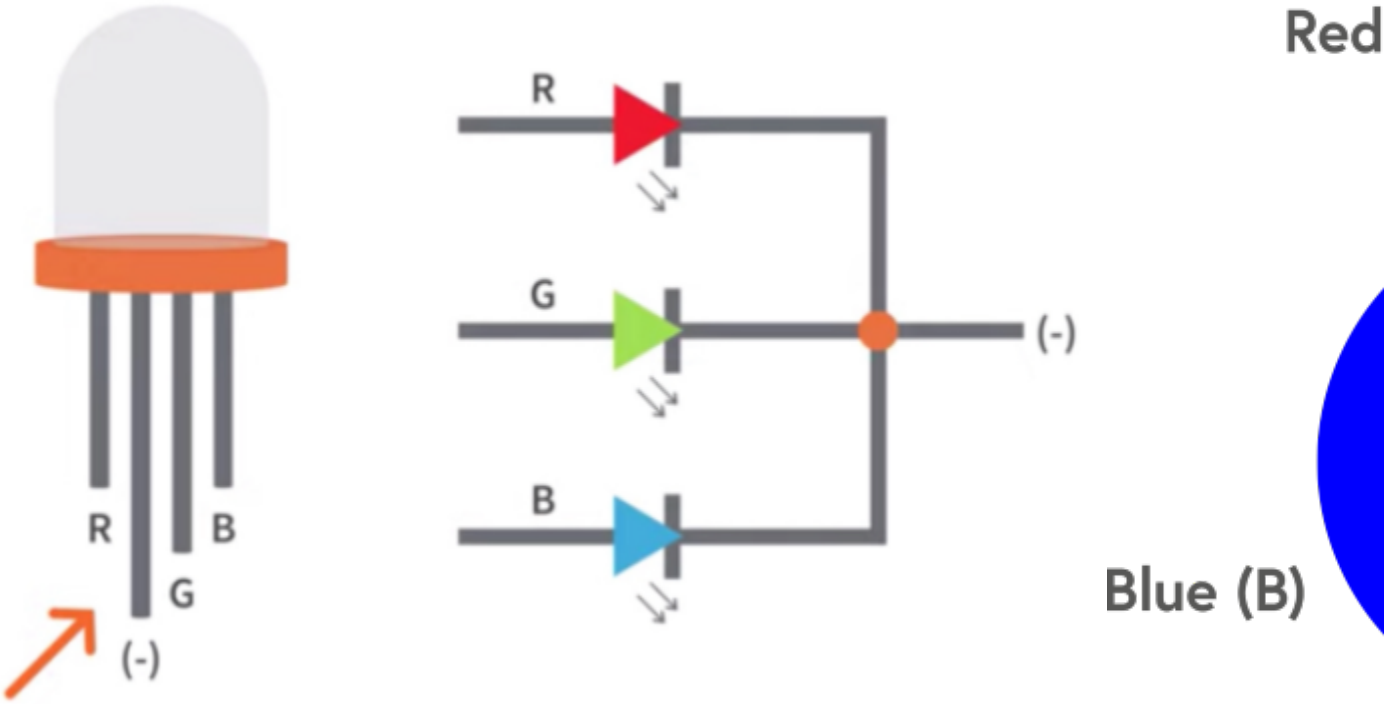
Components Required

					
Keyestudio Plus Mainboardx1	RGB LEDx1	220 Resistorx3	Breadboardx1	Jumper Wires	USB Ca-blex1

Component Knowledge

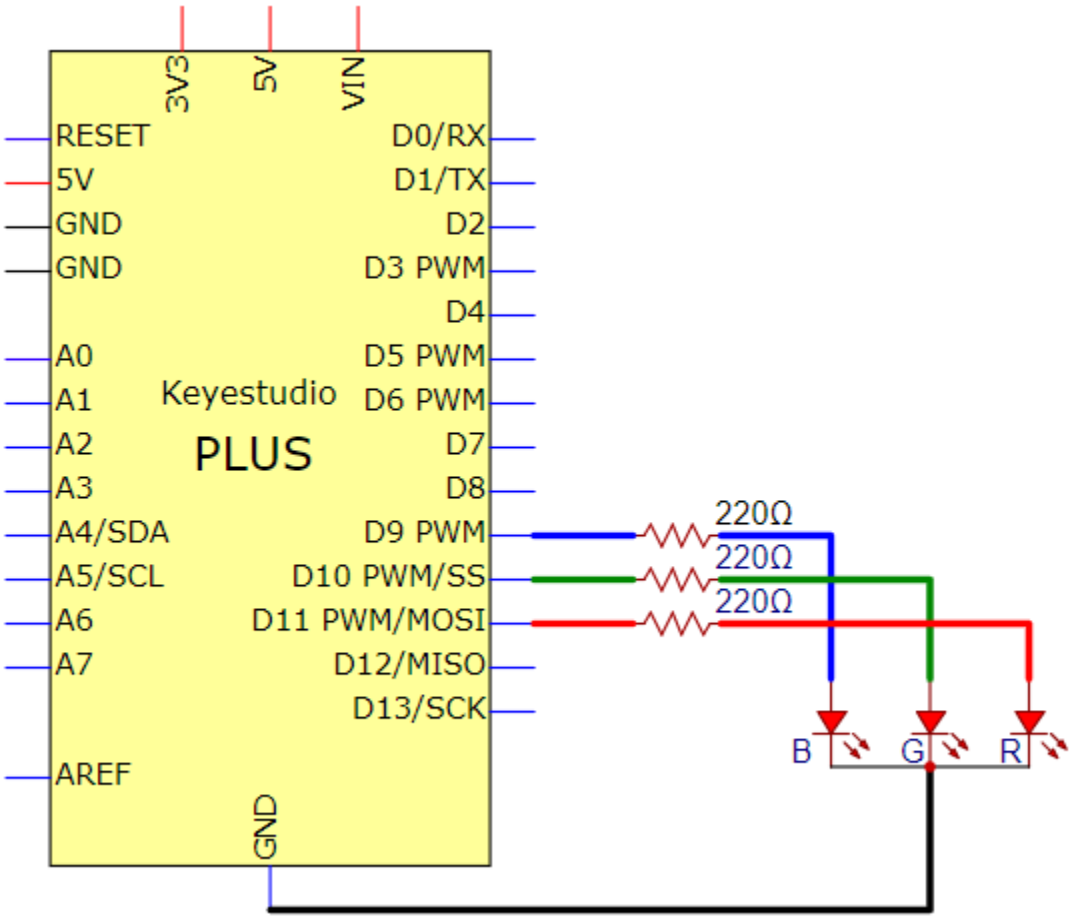
RGB LED

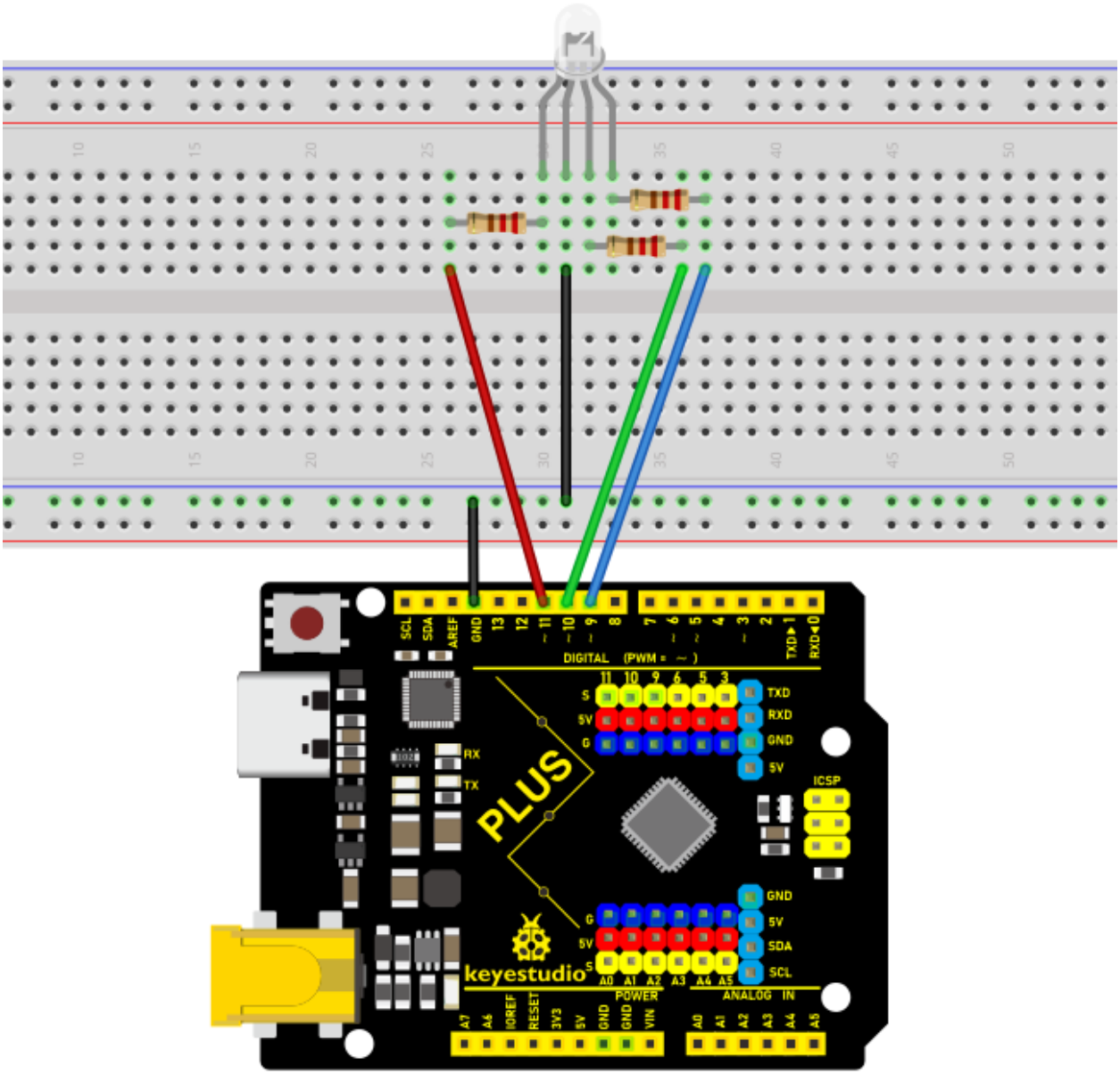
The monitors mostly adopt the RGB color standard, and all the colors on the computer screen are composed of the three colors of red, green and blue mixed in different proportions.



This RGB LED has pin R, G and B and a common cathode. To change its brightness, we can use the PWM pins which can give different duty cycle signals to the RGB LED to produce different colors.

Circuit Diagram and Wiring Diagram

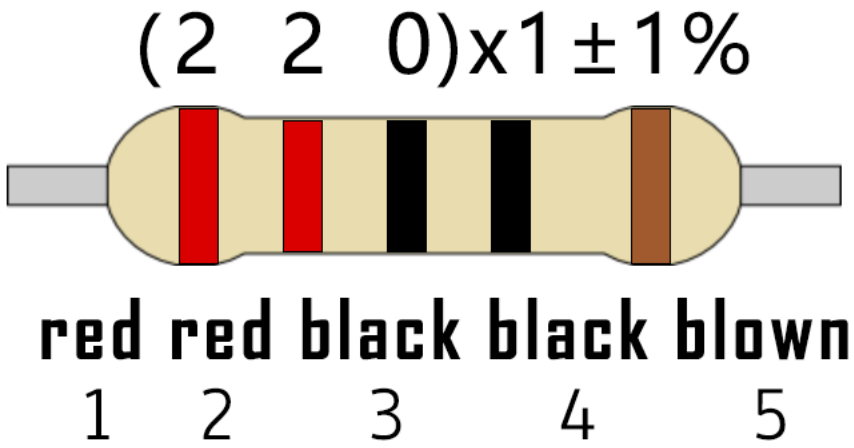

3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
220Ω
220Ω
220Ω
B
G
R



Note:

RGB LED longest pin (common cathode) connected to GND.

How to identify the 220 5-band resistor



Code

```
/*

Keyestudio 2021 starter learning kit

Project 5

RGB LED

http//www.keyestudio.com

*/

int redpin = 11; // select the pin for the red LED

int bluepin =9; // select the pin for the blue LED

int greenpin =10;// select the pin for the green LED
```

(continues on next page)

(continued from previous page)

```
int val;

void setup() {

pinMode(redpin, OUTPUT);

pinMode(bluepin, OUTPUT);

pinMode(greenpin, OUTPUT);

}

void loop()

{

for(val=255; val\>0; val--)

{

analogWrite(11, val);

analogWrite(10, 255-val);

analogWrite(9, 128-val);

delay(1);

}

for(val=0; val\<255; val++)

{

analogWrite(11, val);

analogWrite(10, 255-val);

analogWrite(9, 128-val);

delay(1);

}

}
```

Result

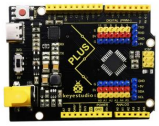
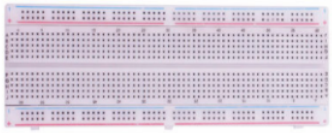
Upload the project code, wire up, power up and wait a few seconds, you will see a colorful LED.

5.6 Project 6: Flowing Light

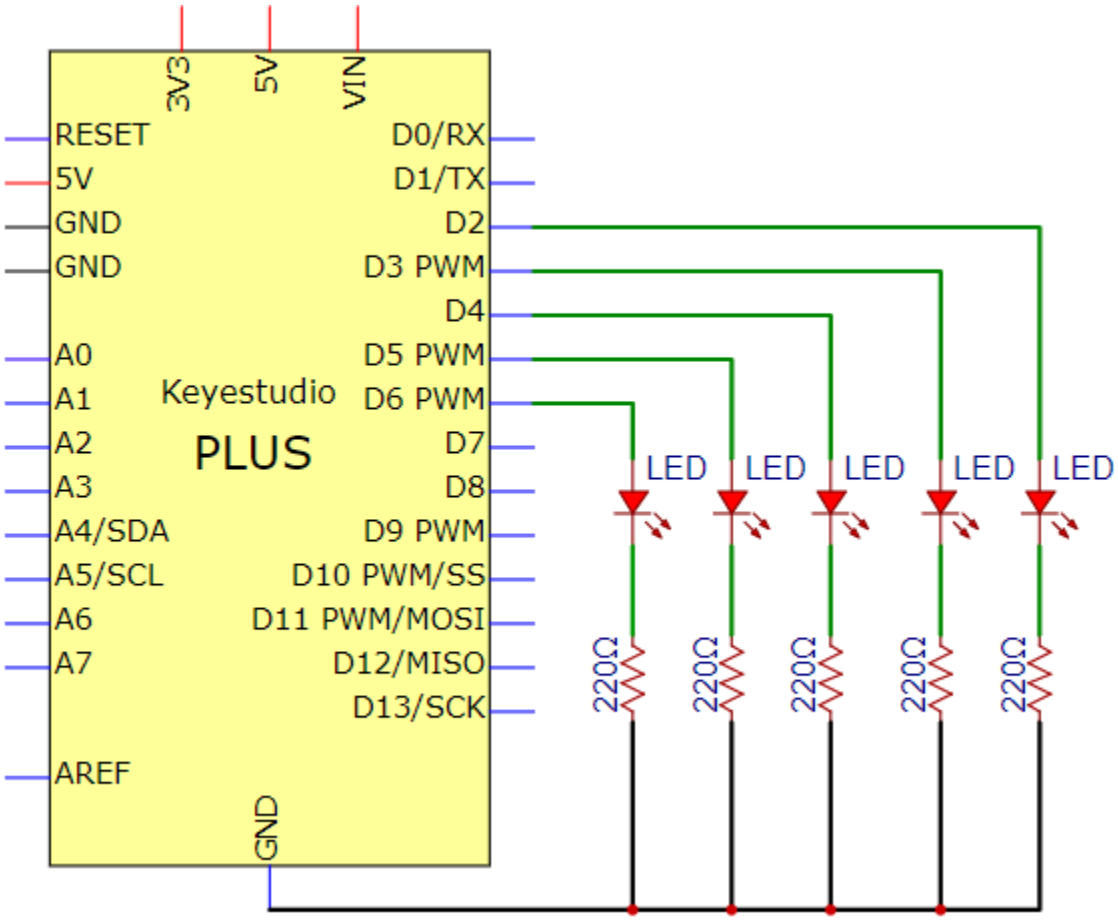
Introduction

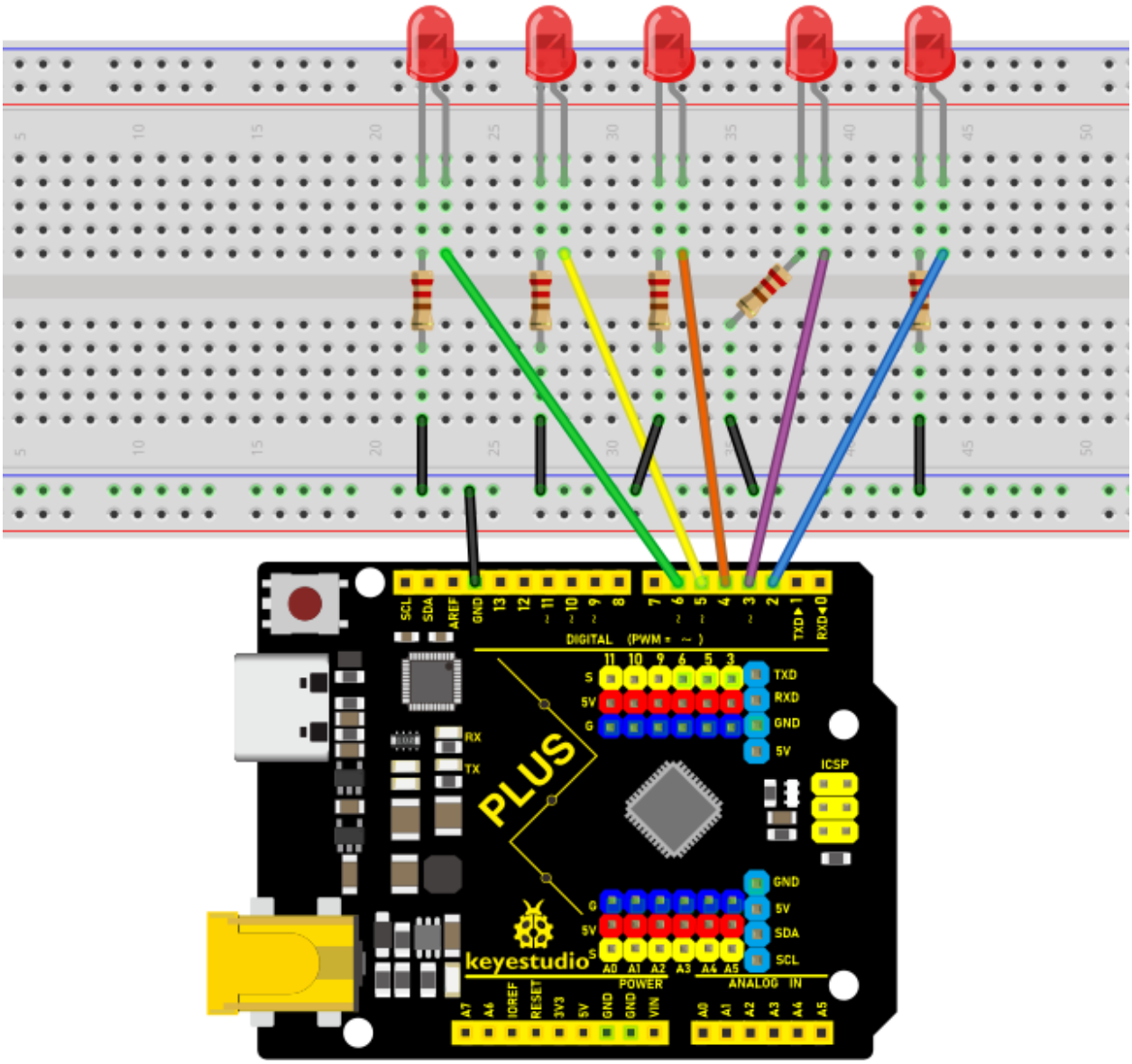
In our daily life, we can see many billboards made up of different colors of LED. They constantly change the light to attract the attention of customers. In this project, we will use Plus control board with 5 LEDs to achieve the effect of flowing water.

Components Required

					
Keyestudio Plus Mainboardx1	Red LEDx5	220 Resistorx5	Breadboard x1	Jumper Wires	USB Ca-blex1

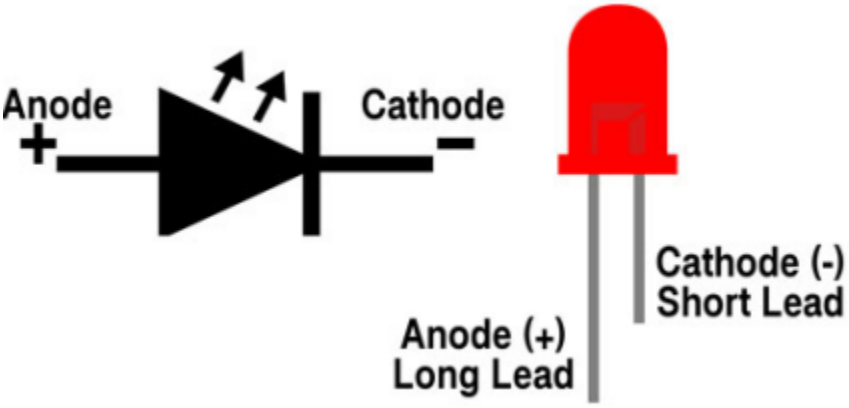
Circuit Diagram and Wiring Diagram



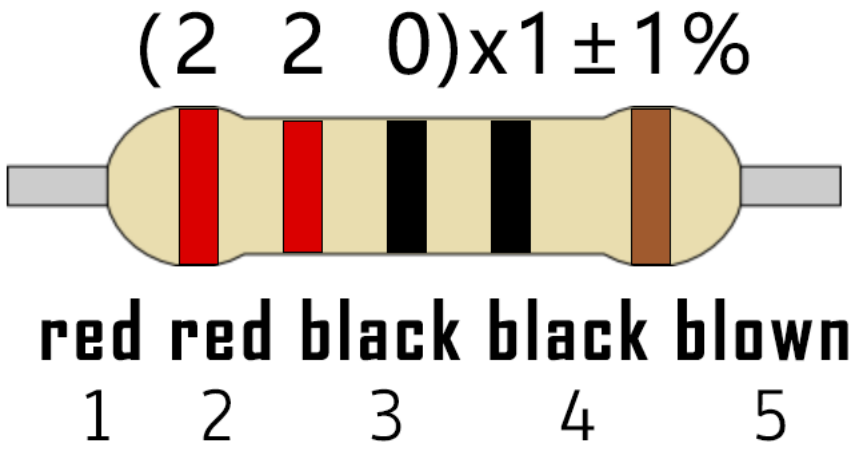


Note:

How to connect the LED



How to identify the 220 5-band resistor



Code

```
/*

Keyestudio 2021 Starter Kit

Project 6

Flowing_Light

http//www.keyestudio.com

*/

int BASE = 2 ;// I/O PIN for the first LED

int NUM = 5; // amount of LEDs

void setup()

{

for (int i = BASE; i \< BASE + NUM; i ++)

{

pinMode(i, OUTPUT); // sets I/O PIN to "output"

}

}

void loop()

{
```

(continues on next page)

(continued from previous page)

```
for (int i = BASE; i \< BASE + NUM; i ++)

{

digitalWrite(i, LOW); //sets I/O PIN to "low", turns off LEDs one after the
other

delay(200); // delay

}

for (int i = BASE; i \< BASE + NUM; i ++)

{

digitalWrite(i, HIGH); // sets I/O PIN to "high", turns on LEDs one after
the other

delay(200); // delay

}

}
```

Result

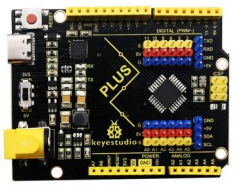
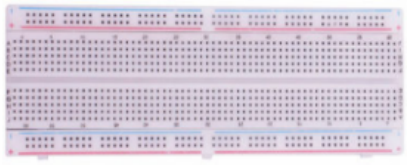
After burning the project code, connecting the wires and powering on, the 5 LEDs connected to the D2 to D6 pins of the development board will gradually light up and then gradually go off, just like a battery charge.

5.7 Project 7: Active Buzzer

Introduction

Active buzzer is a sound making element, widely used on computers, printers, alarms, electronic toys, telephones, timers, etc. It has an inner vibration source. In this project, we will use a PLUS control board to control the active buzzer to buzz.

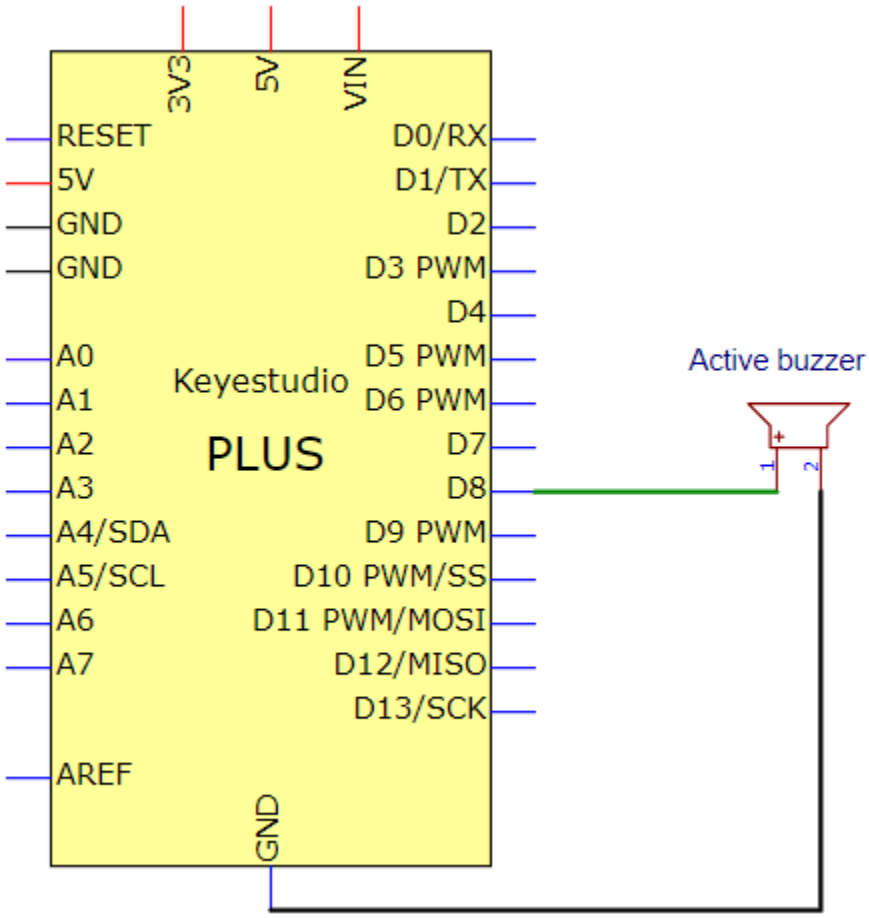
Components Required

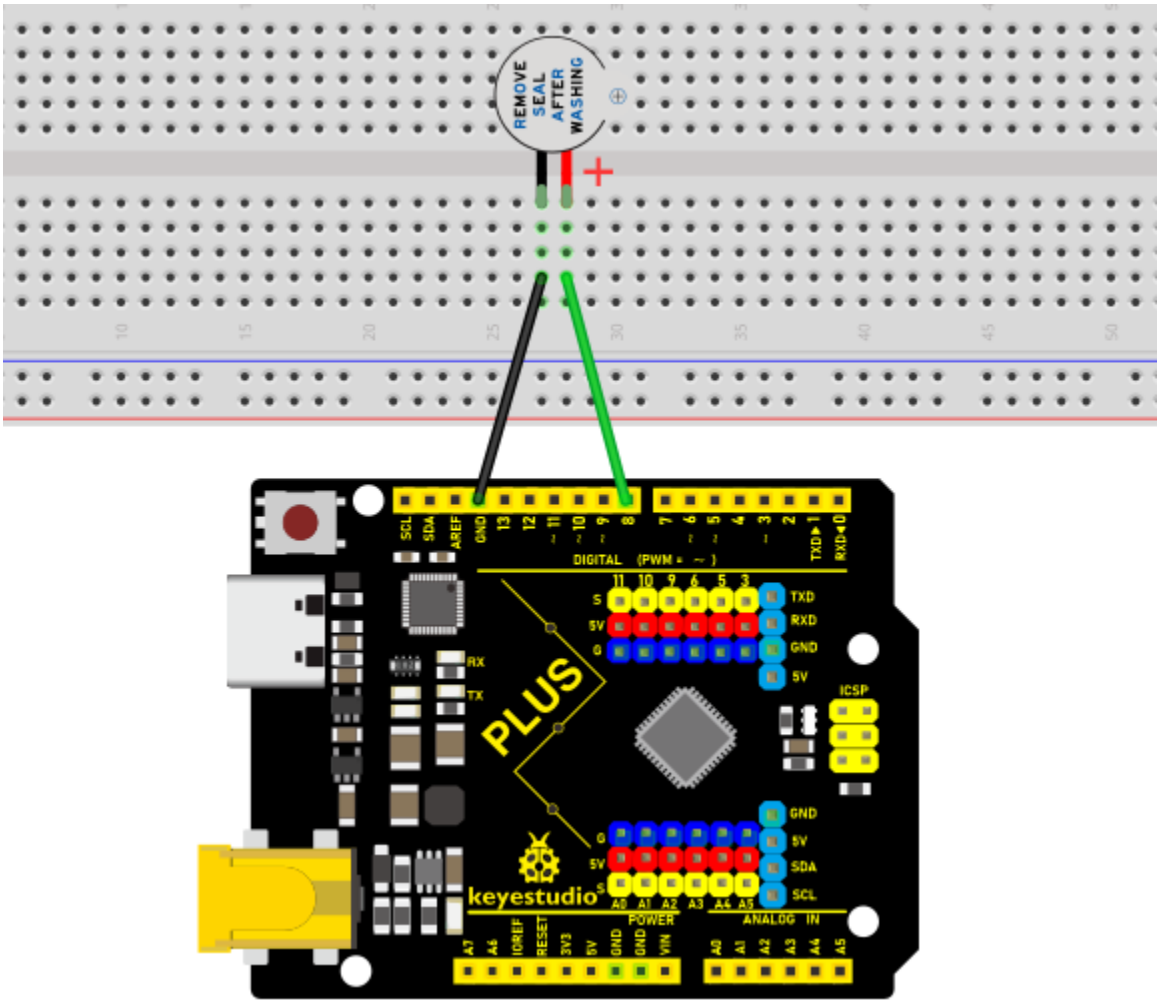
				
Keyestudio Plus Mainboardx1	Active Buzzerx1	Breadboard x1	Jumper Wires	USB Cablex1

Component Knowledge

The active buzzer inside has a simple oscillator circuit which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will sound. The passive buzzer is an integrated electronic buzzer with no internal vibration source. It must be driven by 2K to 5K square wave instead of a DC signal. The appearance of the two buzzers is very similar, but passive buzzers come with a green circuit board, and active buzzers come with a black tape. Passive buzzers don't have positive pole, but active buzzers have.

Circuit Diagram and Wiring Diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
Active buzzer
+
1
2



Note: The positive terminal ("+"/long pin) of the active buzzer is connected to pin 8, and the negative terminal (short pin) is connected to GND.

Code

```
/*

Keyestudio 2021 Starter Kit

Project 7

Active_buzzer

http//www.keyestudio.com

*/

int buzzerPin = 8;

void setup ()

{
```

(continues on next page)

(continued from previous page)

```
pinMode (buzzerPin, OUTPUT);

}

void loop ()

{

digitalWrite (buzzerPin, HIGH);

delay (500);

digitalWrite (buzzerPin, LOW);

delay (500);

}
```

Result

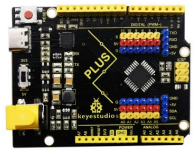
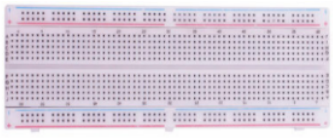
Upload the project code, wire up and power up, then the active buzzer buzzes.

5.8 Project 8: Passive Buzzer

Introduction

In this project, we will learn the passive buzzer and use the Plus control board to control the passive buzzer to play a song. Unlike an active buzzer, a passive buzzer can emit sounds of different frequencies.

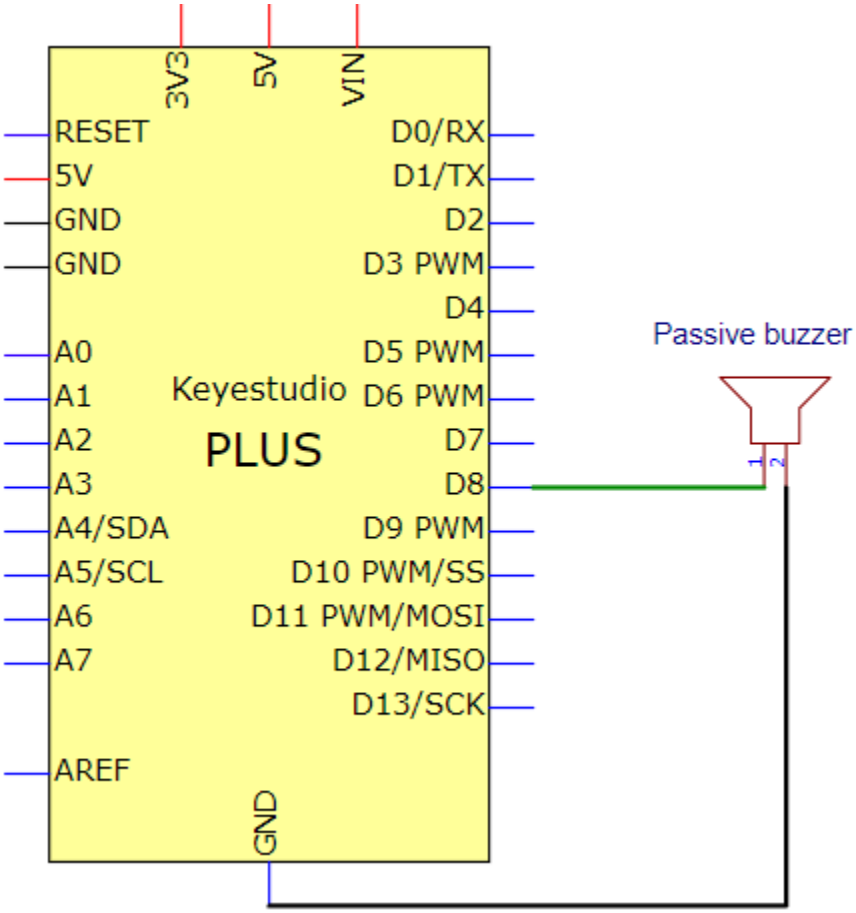
Components Required

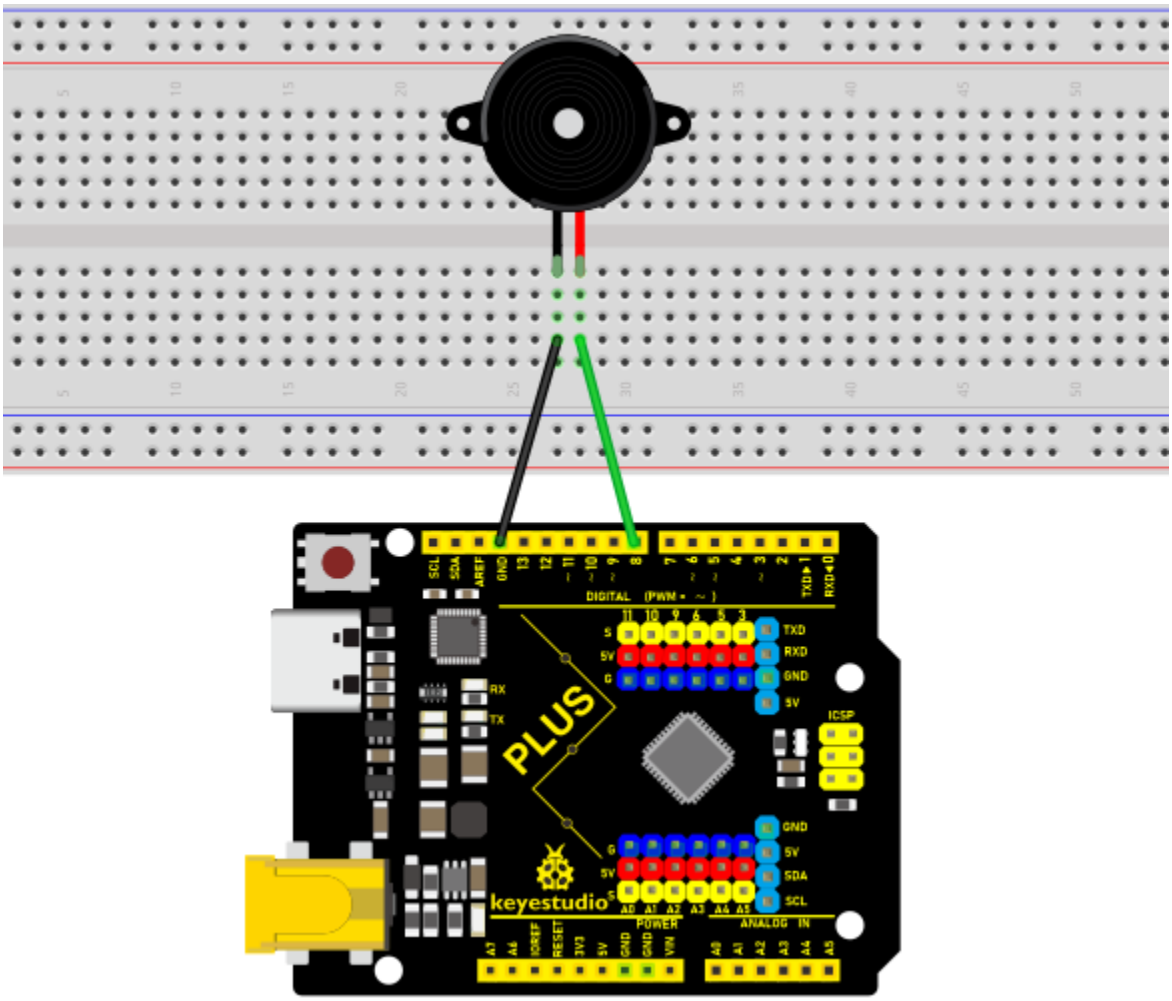
				
Keyestudio Plus Mainboardx1	Passive Buzzerx1	Breadboardx1	Jumper Wires	USB Cablex1

Component Knowledge

A passive buzzer is an integrated electronic buzzer with no internal vibration source. It must be driven by 2K to 5K square wave, not a DC signal. The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer, while the other with black tape is an active buzzer. Passive buzzers cannot distinguish between positive polarity while active buzzers can.

Circuit Diagram and Wiring Diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
Passive buzzer



Code

```
/*

Keyestudio 2021 Starter Kit

Project 8

Passive_buzzer

http//www.keyestudio.com

*/

\#define NOTE_B0 31

\#define NOTE_C1 33

\#define NOTE_CS1 35

\#define NOTE_D1 37
```

(continues on next page)

(continued from previous page)

```
\#define NOTE_DS1 39
\#define NOTE_E1 41
\#define NOTE_F1 44
\#define NOTE_FS1 46
\#define NOTE_G1 49
\#define NOTE_GS1 52
\#define NOTE_A1 55
\#define NOTE_AS1 58
\#define NOTE_B1 62
\#define NOTE_C2 65
\#define NOTE_CS2 69
\#define NOTE_D2 73
\#define NOTE_DS2 78
\#define NOTE_E2 82
\#define NOTE_F2 87
\#define NOTE_FS2 93
\#define NOTE_G2 98
\#define NOTE_GS2 104
\#define NOTE_A2 110
\#define NOTE_AS2 117
\#define NOTE_B2 123
\#define NOTE_C3 131
\#define NOTE_CS3 139
\#define NOTE_D3 147
\#define NOTE_DS3 156
\#define NOTE_E3 165
```

(continues on next page)

(continued from previous page)

```
\#define NOTE_F3 175
\#define NOTE_FS3 185
\#define NOTE_G3 196
\#define NOTE_GS3 208
\#define NOTE_A3 220
\#define NOTE_AS3 233
\#define NOTE_B3 247
\#define NOTE_C4 262
\#define NOTE_CS4 277
\#define NOTE_D4 294
\#define NOTE_DS4 311
\#define NOTE_E4 330
\#define NOTE_F4 349
\#define NOTE_FS4 370
\#define NOTE_G4 392
\#define NOTE_GS4 415
\#define NOTE_A4 440
\#define NOTE_AS4 466
\#define NOTE_B4 494
\#define NOTE_C5 523
\#define NOTE_CS5 554
\#define NOTE_D5 587
\#define NOTE_DS5 622
\#define NOTE_E5 659
\#define NOTE_F5 698
\#define NOTE_FS5 740
```

(continues on next page)

(continued from previous page)

```
\#define NOTE_G5 784
\#define NOTE_GS5 831
\#define NOTE_A5 880
\#define NOTE_AS5 932
\#define NOTE_B5 988
\#define NOTE_C6 1047
\#define NOTE_CS6 1109
\#define NOTE_D6 1175
\#define NOTE_DS6 1245
\#define NOTE_E6 1319
\#define NOTE_F6 1397
\#define NOTE_FS6 1480
\#define NOTE_G6 1568
\#define NOTE_GS6 1661
\#define NOTE_A6 1760
\#define NOTE_AS6 1865
\#define NOTE_B6 1976
\#define NOTE_C7 2093
\#define NOTE_CS7 2217
\#define NOTE_D7 2349
\#define NOTE_DS7 2489
\#define NOTE_E7 2637
\#define NOTE_F7 2794
\#define NOTE_FS7 2960
\#define NOTE_G7 3136
\#define NOTE_GS7 3322
```

(continues on next page)

(continued from previous page)

```
\#define NOTE_A7 3520

\#define NOTE_AS7 3729

\#define NOTE_B7 3951

\#define NOTE_C8 4186

\#define NOTE_CS8 4435

\#define NOTE_D8 4699

\#define NOTE_DS8 4978

\#define REST 0

int tempo=114; // change this to make the song slower or faster

int buzzer = 8;// initializes digital I/O PIN to control the buzzer

// notes of the moledy followed by the duration

// a 4 means a quarter note, 8 an eighteenth , 16 sixteenth, so on

// !!negative numbers are used to represent dotted notes

// so -4 means a dotted quarter note, a quarter plus an eighteenth

int melody[] = {

NOTE_E4,4, NOTE_E4,4, NOTE_F4,4, NOTE_G4,4,//1

NOTE_G4,4, NOTE_F4,4, NOTE_E4,4, NOTE_D4,4,

NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,

NOTE_E4,-4, NOTE_D4,8, NOTE_D4,2,

NOTE_E4,4, NOTE_E4,4, NOTE_F4,4, NOTE_G4,4,//4

NOTE_G4,4, NOTE_F4,4, NOTE_E4,4, NOTE_D4,4,

NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,

NOTE_D4,-4, NOTE_C4,8, NOTE_C4,2,

NOTE_D4,4, NOTE_D4,4, NOTE_E4,4, NOTE_C4,4,//8

NOTE_D4,4, NOTE_E4,8, NOTE_F4,8, NOTE_E4,4, NOTE_C4,4,

NOTE_D4,4, NOTE_E4,8, NOTE_F4,8, NOTE_E4,4, NOTE_D4,4,
```

(continues on next page)

(continued from previous page)

```
NOTE_C4,4, NOTE_D4,4, NOTE_G3,2,

NOTE_E4,4, NOTE_E4,4, NOTE_F4,4, NOTE_G4,4,//12

NOTE_G4,4, NOTE_F4,4, NOTE_E4,4, NOTE_D4,4,

NOTE_C4,4, NOTE_C4,4, NOTE_D4,4, NOTE_E4,4,

NOTE_D4,-4, NOTE_C4,8, NOTE_C4,2

};

//  each int value is composed of two bytes (16
bits)

// there are two values per note , so for each note there
are four bytes

int notes=sizeof(melody)/sizeof(melody[0])/2;

// this calculates the duration of a whole note in ms(60s/tempo)x4 beats

int wholenote = (60000 x 4) / tempo;

int divider = 0, noteDuration = 0;

void setup() {

// iterate over the notes of the melody

// remember, the array is twice the number of notes (notes + durations)

for (int thisNote = 0; thisNote \< notes x 2; thisNote = thisNote + 2) {

// calculates the duration of each note

divider = melody[thisNote + 1];

if (divider \> 0) {

noteDuration = (wholenote) / divider; // regular note, just proceed

} else if (divider \< 0) {

// dotted notes are represented with negative durations!!

noteDuration = (wholenote) / abs(divider);

noteDuration x= 1.5; // increases the duration in half for dotted notes

}
```

(continues on next page)

(continued from previous page)

```
// we only play the note for 90% of the duration, leaving 10% as a pause

tone(buzzer, melody[thisNote], noteDurationx0.9);

// Wait for the specief duration before playing the next note

delay(noteDuration);

noTone(buzzer); // stop the waveform generation before the next note

}

}

void loop() {

// if you want to repeat the song forever,

// just paste the setup code here instead.

}
```

Result

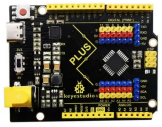
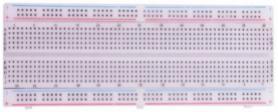
Upload the project code, wire up and power on, then the passive buzzer will play a song.

5.9 Project 9: 74HC595N Controls 7 LEDs

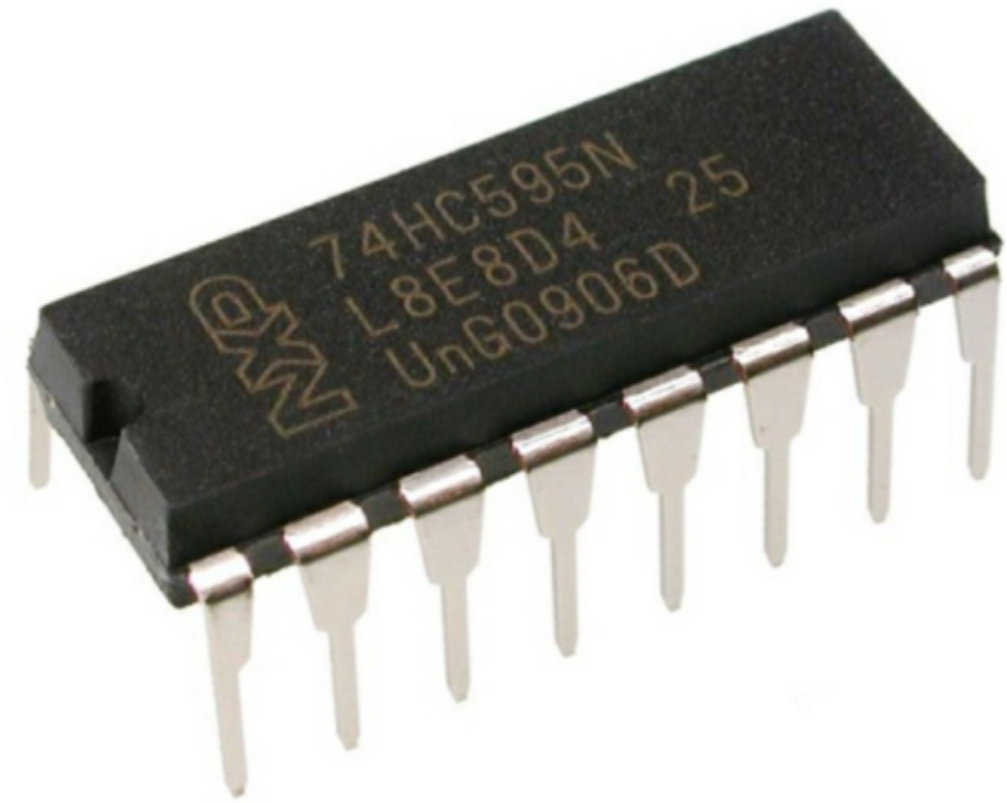
Introduction

For a PLUS mainboard, it has only 22 I/O ports, how do we light up a large number of LEDs? In this project, we will use 74HC595N to control 7 LEDs to save port resources.

Components Required

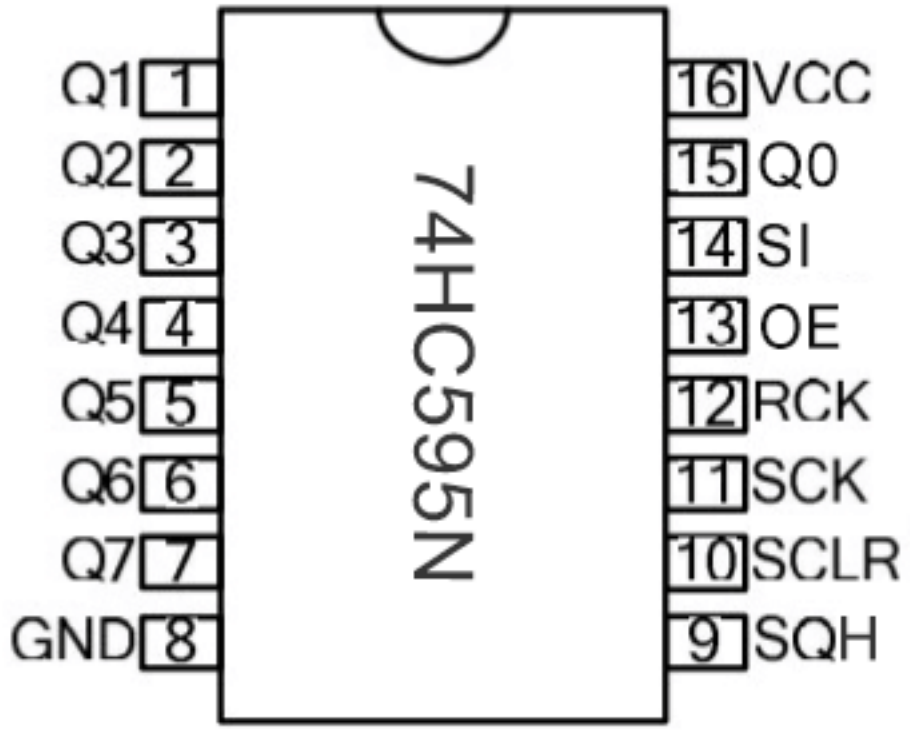
						
Keyestudio Plus Mainboardx1	Red LEDx7	74HC595N Chipx1	220 Resistorx7	Breadboardx1	Jumper Wires	USB Ca-blex1

Component Knowledge



74HC595N Chip: To put it simply, 74HC595N chip is a combination of 8-digit shifting register, memorizer and equipped with tri-state output.The shift register and the memorizer are synchronized to different clocks, and the data is input on the rising edge of the shift register clock SCK and goes into the memory register on the rising edge of the memory register clock RCK. If the two clocks are connected together, the shift register is always one pulse earlier than the storage register.

The shift register has a serial shift input (SI) and a serial output (SQH) for cascading. The 8-bit shift register can be reset asynchronously (low-level reset), and the storage register has an 8-bit Three-state parallel bus output, when the output enable (OE) is enabled (active low), the storage register is output to the 74HC595N pin (bus).

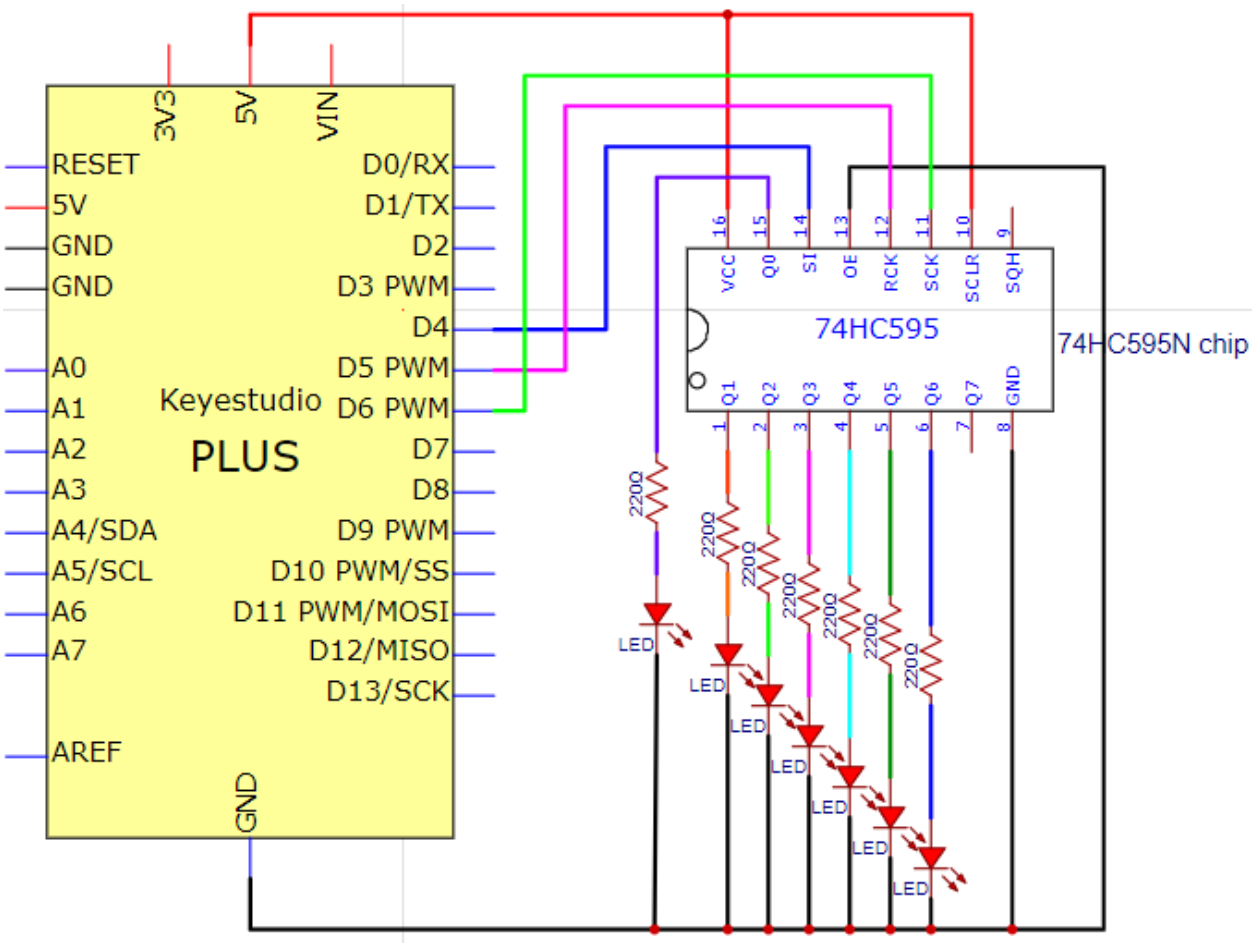


Pins

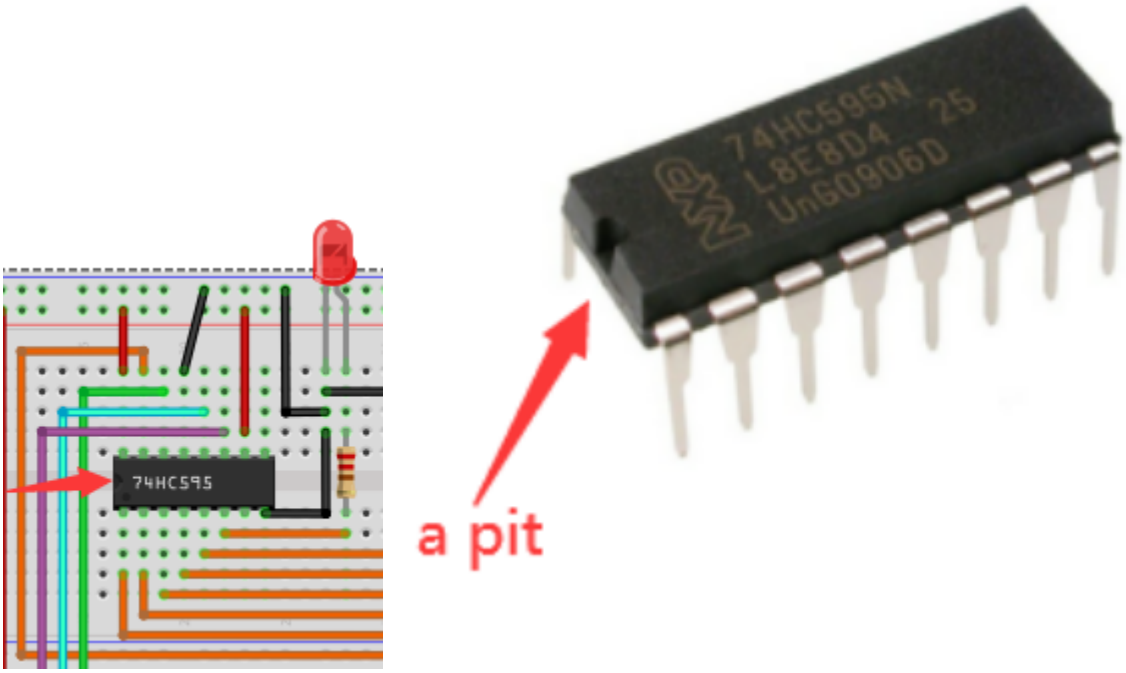
Pin13 OE	It is an output enable pin to ensure that the data of the latch is input to the Q0 to Q7 pins or not. When it is low, no high level is output. In this experiment, we directly connect to GND and keep the data output low.
Pin14 SI	This is the pin for 74HC595 to receive data, i.e. serial data input, only one bit can be input at a time, then 8 times in a row, it can form a byte.
Pin10 SCLR	A pin to initialize the storage register pins. It initializes the internal storage registers at a low level. In this experiment, we connect VCC to maintain a high level.
Pin11 SCK	The clock pin of the shift register. At the rising edge, the data in the shift register is shifted backward as a whole, and new data input is received.
Pin12 RCK	The clock input pin of the storage register . At the rising edge, the data is transferred from the shift register to the storage register. At this time, the data is output in parallel from the Q0 to Q7 ports.
Pin9 SQH	It is a serial output pin dedicated for chip cascading to the SI terminal of the next 74HC595.
Q0–Q7(Pin 15,Pin1-7)	Eight-bit parallel output, can directly control the 8 segments of the digital tube.

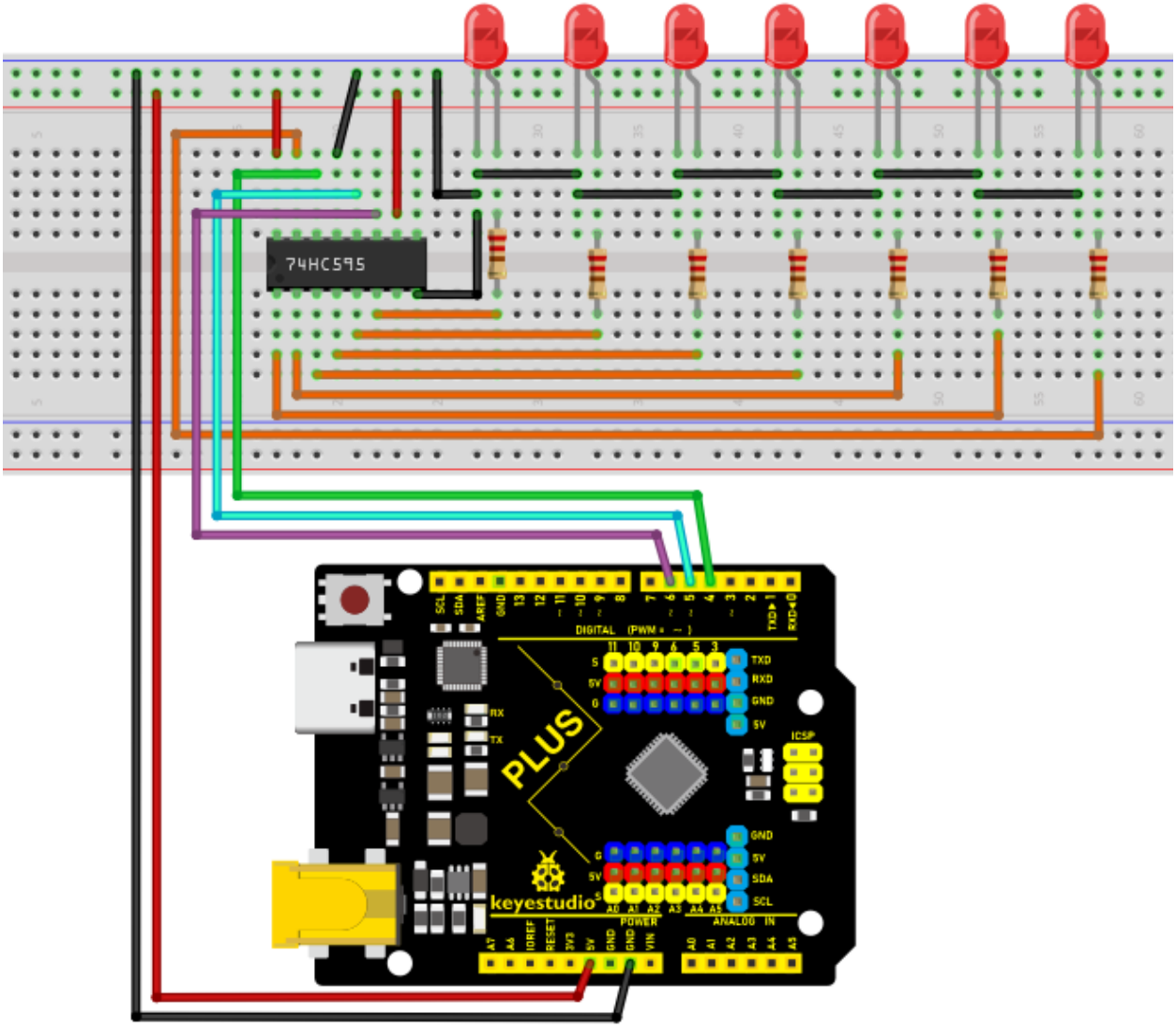
VCC and GND are used used for chip power supply, and the operating voltage is 5V.

Circuit Diagram and Wiring Diagram



Note: Pay attention to the direction in which the 74HC595N chip is inserted.


74HC595N
L8E8D4
UnG0906D
25
74HC595
a pit



Code

```
/*

Keyestudio 2021 Starter Kit

Project 9

74HC595N_control_7_LEDS

http//www.keyestudio.com

*/

int data = 4;// sets PIN 4 of the 74HC595 to datainput PIN SI

int clock = 6;// sets PIN 6 of the 74HC595 to clock PIN SCK
```

(continues on next page)

(continued from previous page)

```
int latch = 5;// sets PIN 5 of the 74HC595 to output latch RCK

int ledState = 0;

const int ON = HIGH;

const int OFF = LOW;

void setup()

{

pinMode(data, OUTPUT);

pinMode(clock, OUTPUT);

pinMode(latch, OUTPUT);

}

void loop()

{

for(int i = 0; i \< 256; i++)

{

updateLEDs(i);

delay(500);

}

}

void updateLEDs(int value)

{

digitalWrite(latch, LOW);//

shiftOut(data, clock, MSBFIRST, \~value);// shift out highbyte

digitalWrite(latch, HIGH);// lock

}
```

Result

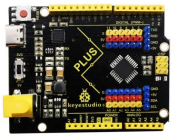
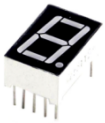
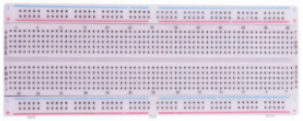
Upload project code, wire up and power on, then you can see the changes of 7 LEDs cyclically.

5.10 Project 10: 1-Digit Digital Tube

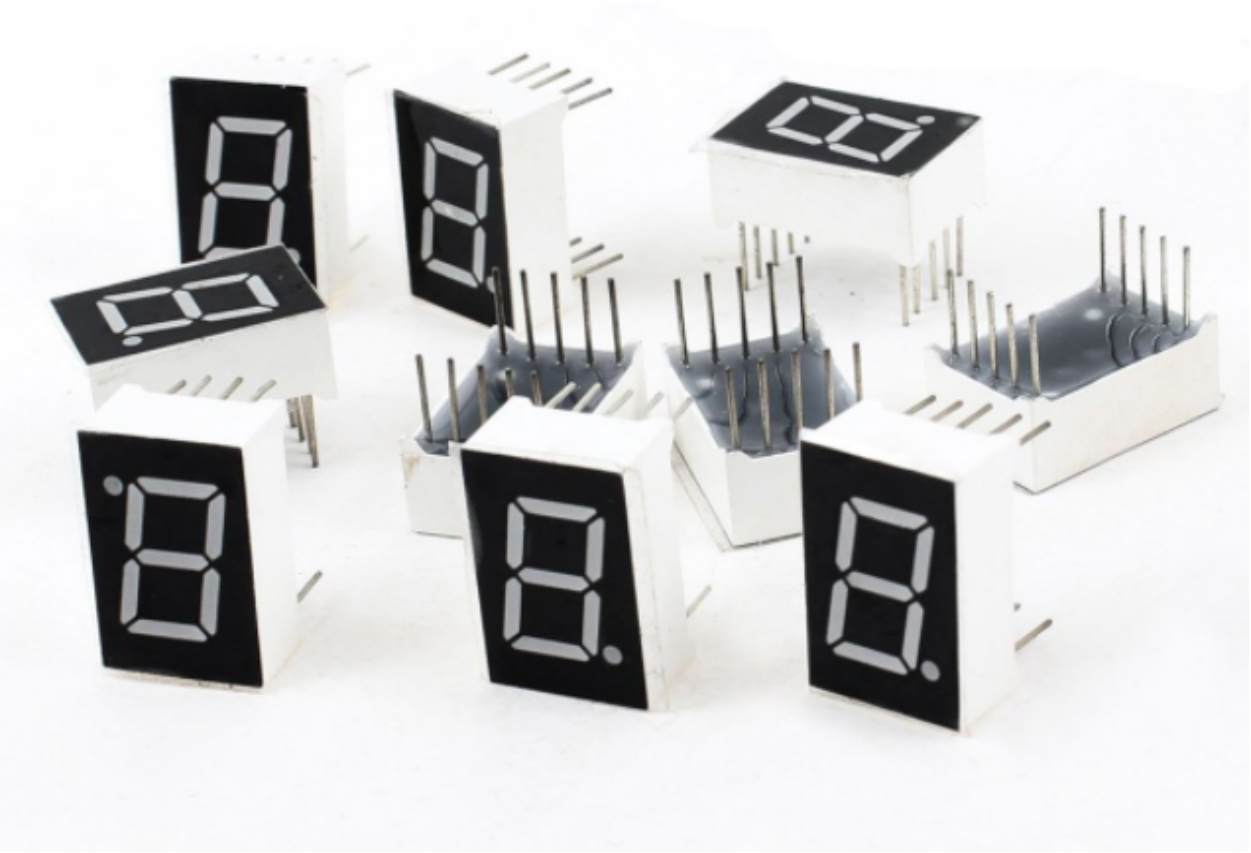
Introduction

The seven-segment digital tube is an electronic display device that displays decimal numbers. It is widely used in digital clocks, electronic meters, basic calculators and other electronic devices that display digital information. The tubes are an alternative to more complex dot-matrix displays that are easy to use in both limited light conditions and strong sunlight. In this project, we will use the PLUS control board to control 1-digit digital tube to display numbers.

Components Required

					
Keyestudio Plus Mainboardx1	1-digit Digital Tubex1	220 Resistorx8	Breadboardx1	Jumper Wires	USB Cablex1

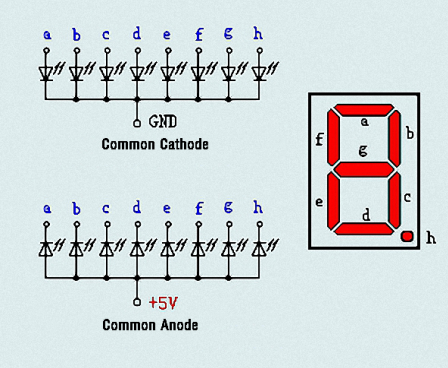
Component Knowledge



Display principle: the digital tube display is a semiconductor light-emitting device. Its basic unit is a light-emitting diode (LED). The digital tube display can be divided into 7-segment digital tube and 8-segment digital tube according to the number of segments. The 8-segment digital tube has one more LED unit than the 7-segment digital tube (used for decimal point display). Each segment of the 7-segment LED display is a separate LED. According to the connection mode of the LED unit, the digital tube can be divided into a common anode digital tube and a common cathode digital tube.

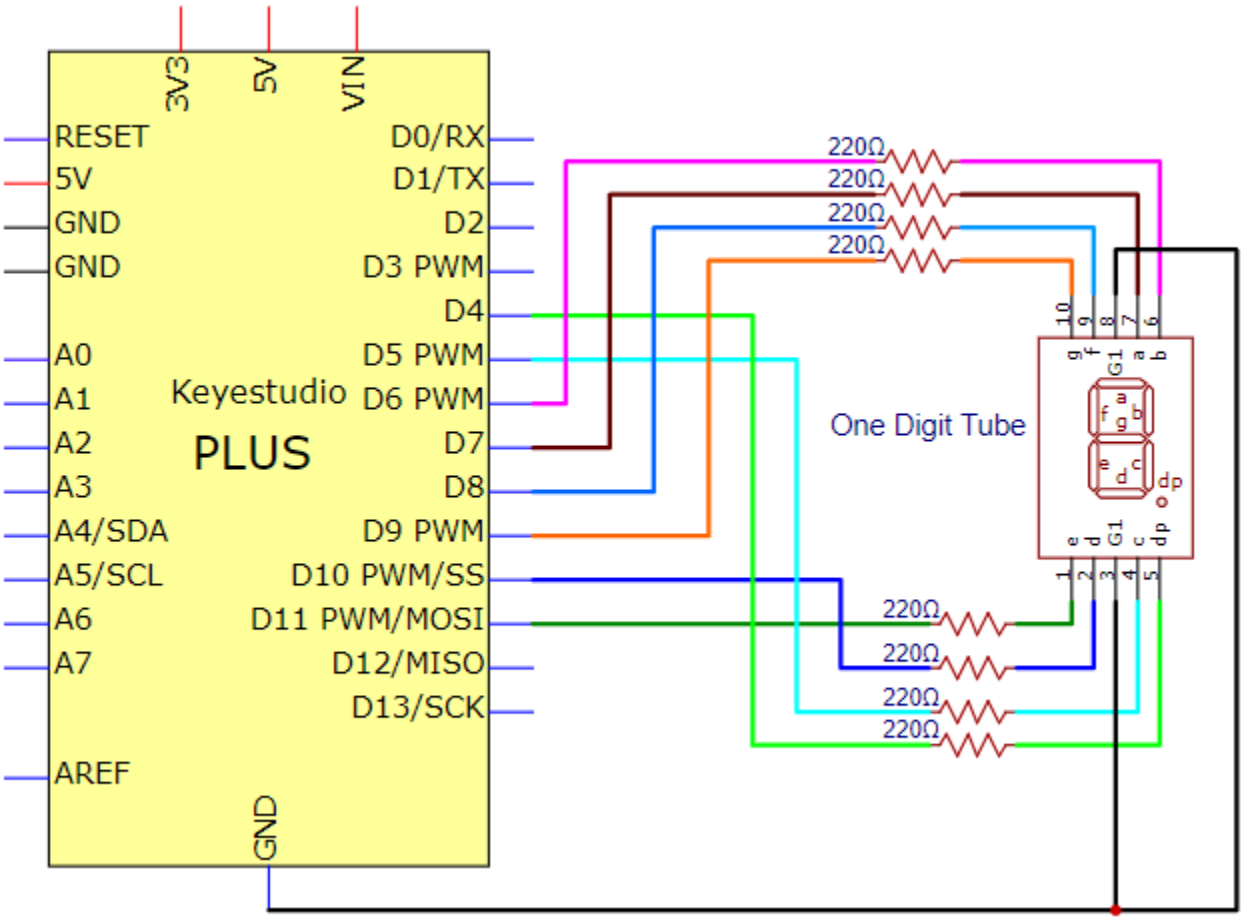
In the common cathode 7-segment digital tube, all the cathodes (or negative electrodes) of the segmented LEDs are connected together, so you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to “HIGH”.

In the common anode 7-segment digital tube, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to “+5V”. To light up a segmented LED, you can set its associated pin to “LOW”.

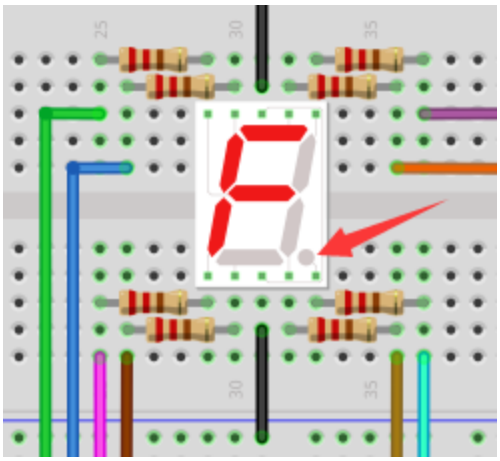


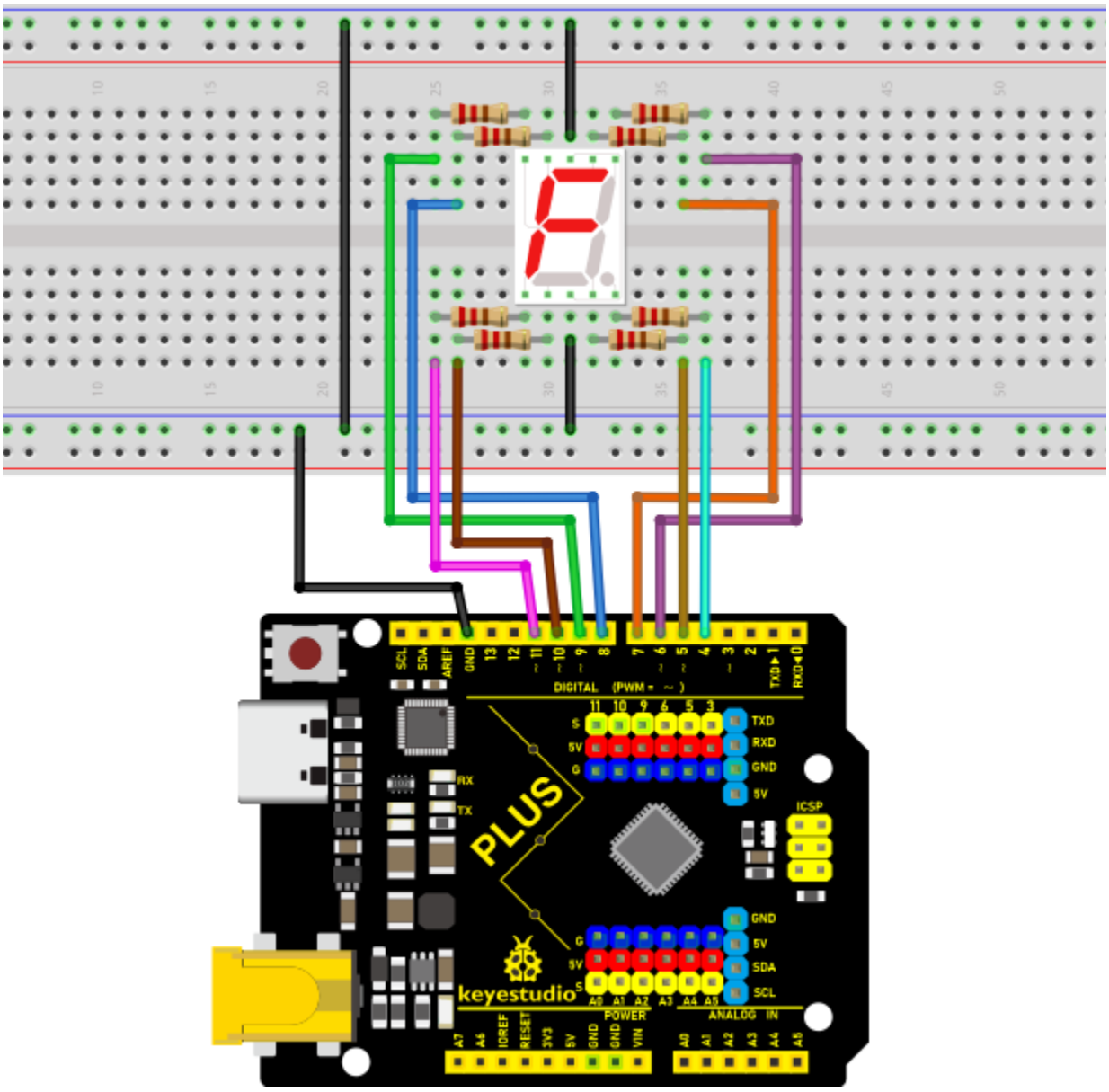
Each part of the digital tube is composed of an LED. So when you use it, you also need to use a current limiting resistor. Otherwise, the LED will be damaged. In this experiment, we use an ordinary common cathode one-bit digital tube. As we mentioned above, you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

Circuit Diagram and Wiring Diagram



Note: The direction of the 7-segment digital tube inserted into the breadboard is the same as the wiring diagram, and there is one more point in the lower right corner.




DIGITAL (PWM~)
TXD
RXD
GND
5V
ICSP
PLUS
keyestudio
POWER
ANALOG IN
SCL
SDA

Code

The digital display is divided into 7 segments, and the decimal point display is divided into 1 segment. When certain numbers are displayed, the corresponding segment will be illuminated. For example, when the number 1 is displayed, segments b and c will be opened. We compile a subroutine for each number, and compile the main program to display a number every 1 second, and display numbers 9 to 0 in cycles. The display time of each number depends on the delay time, the longer the delay time, the longer the display time.

```
/*

Keyestudio 2021 Stater Kit

Project 10

1-digit Digital Tube

http//www.keyestudio.com

*/

// sets the IO PIN for every segment

int a=7;// digital PIN 7 for segment a

int b=6;// digital PIN 6 for segment b

int c=5;// digital PIN 5 for segment c

int d=10;//digital PIN 10 for segment d

int e=11;//digital PIN 11 for segment e

int f=8;//digital PIN 8 for segment f

int g=9;//digital PIN 9 for segment g

int dp=4;//digital PIN 4 for segment dp

void digital_0(void) // displays number 0

{

unsigned char j;

digitalWrite(a,HIGH);

digitalWrite(b,HIGH);

digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(e,HIGH);

digitalWrite(f,HIGH);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_1(void) // displays number 1
{
unsigned char j;
digitalWrite(c,HIGH);// led sets level for PIN 5 to "high",turn on segment c
digitalWrite(b,HIGH);// turns on segment b
for(j=7;j\<=11;j++)// turns off other segments
digitalWrite(j,LOW);
digitalWrite(dp,LOW);// turns off segment dp
}
void digital_2(void) // displays number 2
{
unsigned char j;
digitalWrite(b,HIGH);
digitalWrite(a,HIGH);
for(j=9;j\<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
digitalWrite(c,LOW);
digitalWrite(f,LOW);
}
void digital_3(void) // displays number 3
{digitalWrite(g,HIGH);
digitalWrite(a,HIGH);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(dp,LOW);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
}
void digital_4(void) // displays number 4
{digitalWrite(c,HIGH);
digitalWrite(b,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
digitalWrite(a,LOW);
digitalWrite(e,LOW);
digitalWrite(d,LOW);
}
void digital_5(void) // displays number 5
{
unsigned char j;
digitalWrite(a,HIGH);
digitalWrite(b, LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e, LOW);
digitalWrite(f,HIGH);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_6(void) // displays number 6
{
unsigned char j;
for(j=7;j\<=11;j++)
digitalWrite(j,HIGH);
digitalWrite(c,HIGH);
digitalWrite(dp,LOW);
digitalWrite(b,LOW);
}
void digital_7(void) // displays number 7
{
unsigned char j;
for(j=5;j\<=7;j++)
digitalWrite(j,HIGH);
digitalWrite(dp,LOW);
for(j=8;j\<=11;j++)
digitalWrite(j,LOW);
}
void digital_8(void) // displays number 8
{
unsigned char j;
for(j=5;j\<=11;j++)
digitalWrite(j,HIGH);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(dp,LOW);

}

void digital_9(void) // displays number 9

{

unsigned char j;

digitalWrite(a,HIGH);

digitalWrite(b,HIGH);

digitalWrite(c,HIGH);

digitalWrite(d,HIGH);

digitalWrite(e, LOW);

digitalWrite(f,HIGH);

digitalWrite(g,HIGH);

digitalWrite(dp,LOW);

}

void setup()

{

int i;// declares a Variable

for(i=4;i\<=11;i++)

pinMode(i,OUTPUT);// sets PIN 4-11 to "output"

}

void loop()

{

while(1)

{

digital_9();// displays number 9

delay(1000); // waits a sencond
```

(continues on next page)

(continued from previous page)

```
    digital_8();// displays number 8

    delay(1000); // waits a sencond

    digital_7();// displays number 7

    delay(1000); // waits a sencond

    digital_6();// displays number 6

    delay(1000); // waits a sencond

    digital_5();// displays number 5

    delay(1000); // waits a sencond

    digital_4();// displays number 4

    delay(1000); // waits a sencond

    digital_3();// displays number 3

    delay(1000); // waits a sencond

    digital_2();// displays number 2

    delay(1000); // waits a sencond

    digital_1();// displays number 1

    delay(1000);// waits a sencond

    digital_0();// displays number 0

    delay(1000);// waits a sencond

    }}
```

Result

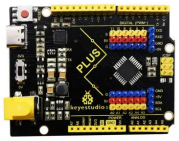
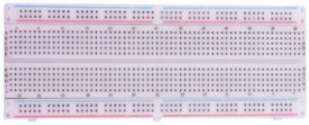
After burning the project code, connecting the wires and powering on, 1-digit digital tube will display numbers from 9 to 0.

5.11 Project 11: 4-Digit 7-Segment Tube Display

Introduction

The 4-digit tube display is low-cost and widely applied to electronic clocks, counters, countdown displays and so on. In this project, we will make the 4-digit 7-segment display show numbers from 0000-9999 through the PLUS MainBoard.

Components

					
Keyestudio PLUS MainBoardx1	Four digit tubex1	220Resistorx8	Jumper Wires	Breadboardx1	USB Ca-blex1

Component Knowledge

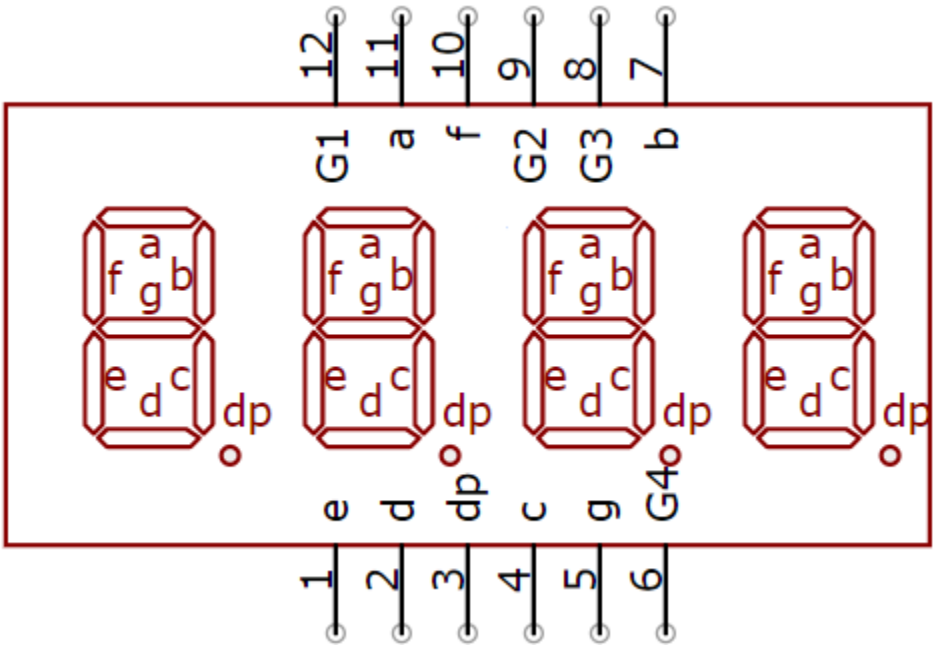
4-digit tube display

The 4-digit tube display is divided into the anode and cathode . Its working principle is similar to the 7-segment displays are really just seven LEDs lined up in a particular pattern. In this case, the number '8' shape we're all familiar with. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used for indication of a decimal point.

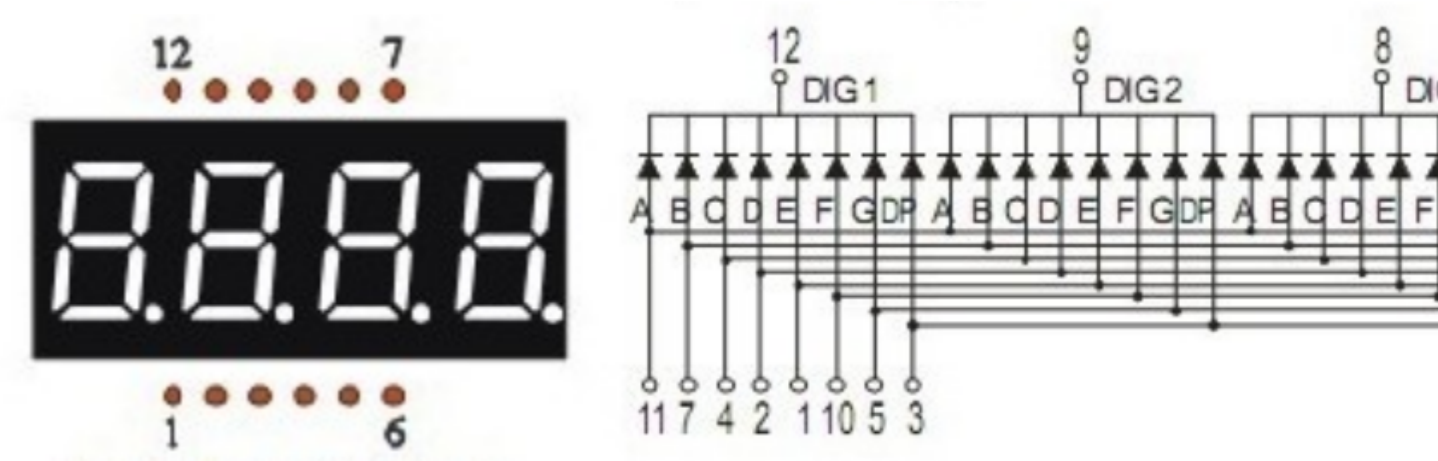
Our four - digit tubes have common cathodes.

Now have a look at the segment configuration so we know which pins light up which segments. The pinout for the 7-segment display is as follows.

The pin G1, G2, G3 and G4 are pins of the control bit.

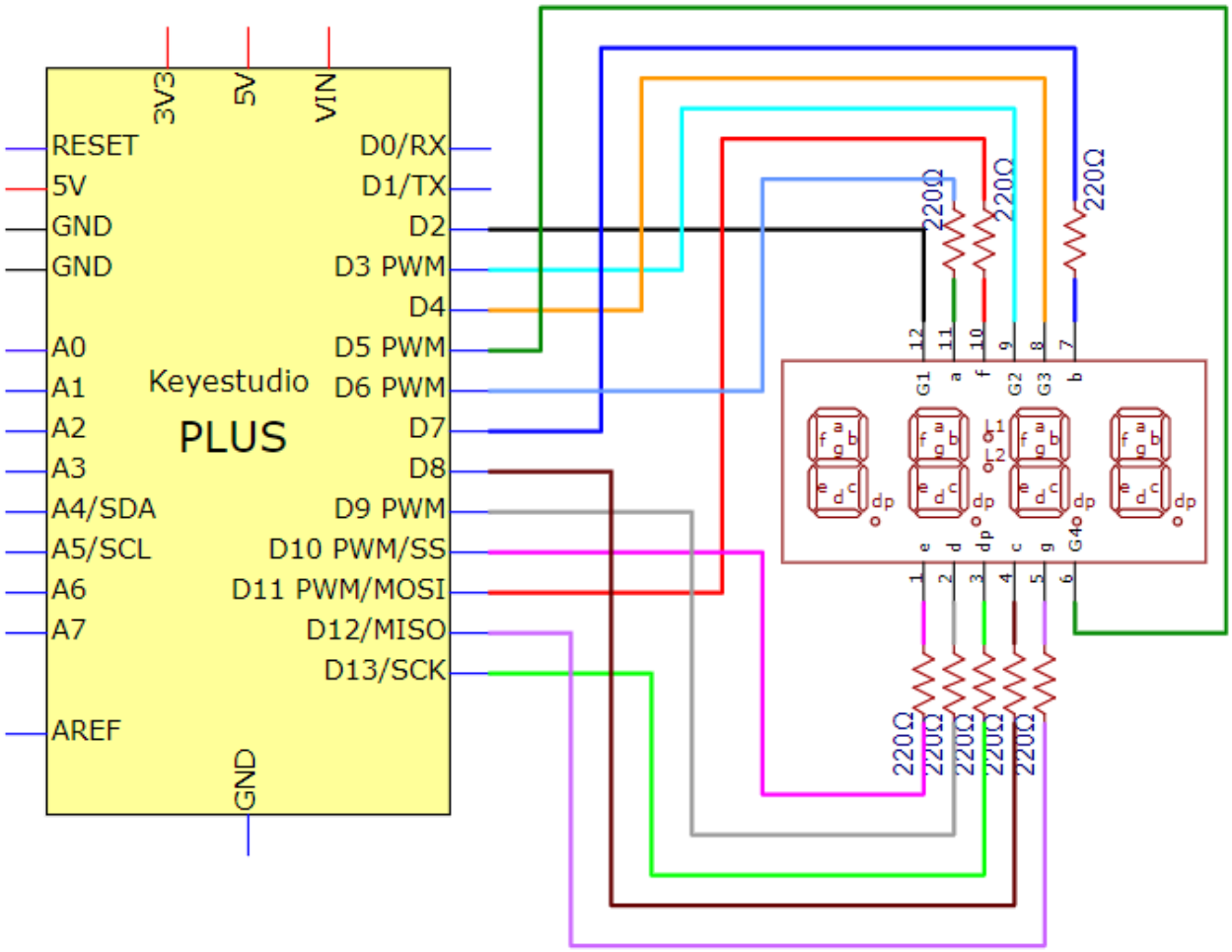


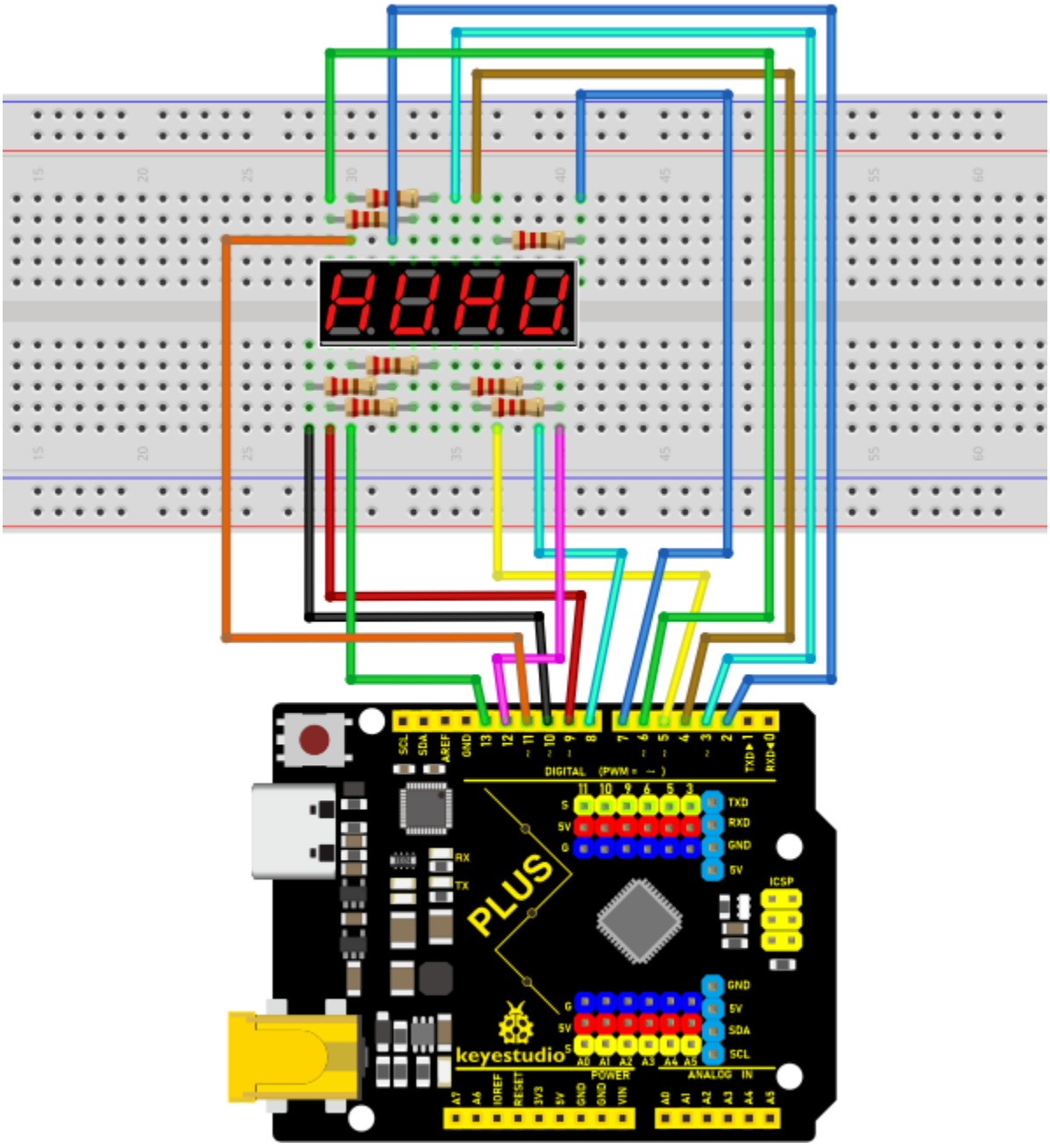
The following figure is the schematic diagram of the internal wiring of the 4-bit digital tube:



Connection Diagram

For a four digit tube, limiting resistors are essential and here we use eight resistors of 220


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
Keyestudio
PLUS
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
G1
G2
G3
G4
L1
L2
dp



Test Code

```
/*

Keyestudio 2021 starter kit

Project 11

Four_segment_display

http//www.keyestudio.com

```

(continues on next page)

(continued from previous page)

```
*/

int a = 6;

int b = 7;

int c = 8;

int d = 9;

int e = 10;

int f = 11;

int g = 12;

int dp = 13;

int g4 = 5;

int g3 = 4;

int g2 = 3;

int g1 = 2;

// set variables

long n = 1230;

int x = 100;

int del = 55; //

void setup()

{

pinMode(g1, OUTPUT);

pinMode(g2, OUTPUT);

pinMode(g3, OUTPUT);

pinMode(g4, OUTPUT);

pinMode(a, OUTPUT);

pinMode(b, OUTPUT);

pinMode(c, OUTPUT);
```

(continues on next page)

(continued from previous page)

```
pinMode(d, OUTPUT);

pinMode(e, OUTPUT);

pinMode(f, OUTPUT);

pinMode(g, OUTPUT);

pinMode(dp, OUTPUT);

}

//////////////////////////////////////////////////////////

void loop()

{

int a=0;

int b=0;

int c=0;

int d=0;

unsigned long currentMillis = millis();

while(d\>=0)

{

while(millis()-currentMillis\<10)

{

Display(1,a);

Display(2,b);

Display(3,c);

Display(4,d);

}

currentMillis = millis();

d++;

if (d\>9)
```

(continues on next page)

(continued from previous page)

```
{

c++;

d=0;

}

if (c\>9)

{

b++;

c=0;

}

if (b\>9)

{

a++;

b=0;

}

if (a\>9)

{

a=0;

b=0;

c=0;

d=0;

}

}

}

////////////////////////////////////////////////////////////

void WeiXuan(unsigned char n)//

{
```

(continues on next page)

(continued from previous page)

```
switch (n)

{

case 1:

digitalWrite(g1, LOW);

digitalWrite(g2, HIGH);

digitalWrite(g3, HIGH);

digitalWrite(g4, HIGH);

break;

case 2:

digitalWrite(g1, HIGH);

digitalWrite(g2, LOW);

digitalWrite(g3, HIGH);

digitalWrite(g4, HIGH);

break;

case 3:

digitalWrite(g1, HIGH);

digitalWrite(g2, HIGH);

digitalWrite(g3, LOW);

digitalWrite(g4, HIGH);

break;

case 4:

digitalWrite(g1, HIGH);

digitalWrite(g2, HIGH);

digitalWrite(g3, HIGH);

digitalWrite(g4, LOW);

break;
```

(continues on next page)

(continued from previous page)

```
default :

digitalWrite(g1, HIGH);

digitalWrite(g2, HIGH);

digitalWrite(g3, HIGH);

digitalWrite(g4, HIGH);

break;

}

}

void Num_0()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void Num_1()

{

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void Num_2()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, LOW);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num_3()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}
```

(continues on next page)

(continued from previous page)

```
void Num_4()

{

digitalWrite(a, LOW);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num_5()

{

digitalWrite(a, HIGH);

digitalWrite(b, LOW);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num_6()

{

digitalWrite(a, HIGH);

digitalWrite(b, LOW);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num_7()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void Num_8()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, HIGH);

digitalWrite(f, HIGH);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Num_9()

{

digitalWrite(a, HIGH);

digitalWrite(b, HIGH);

digitalWrite(c, HIGH);

digitalWrite(d, HIGH);

digitalWrite(e, LOW);

digitalWrite(f, HIGH);

digitalWrite(g, HIGH);

digitalWrite(dp, LOW);

}

void Clear() // clear screens

{

digitalWrite(a, LOW);

digitalWrite(b, LOW);

digitalWrite(c, LOW);

digitalWrite(d, LOW);

digitalWrite(e, LOW);

digitalWrite(f, LOW);

digitalWrite(g, LOW);

digitalWrite(dp, LOW);

}

void pickNumber(unsigned char n)// select numbers
```

(continues on next page)

(continued from previous page)

```
{
switch (n)
{
case 0: Num_0();
break;
case 1: Num_1();
break;
case 2: Num_2();
break;
case 3: Num_3();
break;
case 4: Num_4();
break;
case 5: Num_5();
break;
case 6: Num_6();
break;
case 7: Num_7();
break;
case 8: Num_8();
break;
case 9: Num_9();
break;
default: Clear();
break;
}
```

(continues on next page)

(continued from previous page)

```
}

void Display(unsigned char x, unsigned char Number)// Take x as the coordinate
and display numbers

{

WeiXuan(x);

pickNumber(Number);

delay(1);

Clear() ; // clear screens

}
```

Test Result

Upload the test code, wire up and power up. The 4-digital tube display shows numbers from 0000-9999.

5.12 Project 128x8 Dot Matrix Display

Introduction

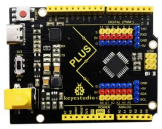
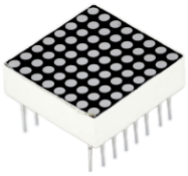
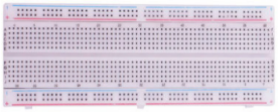
8x8 Dot matrix module can be used as display screen, like bus station display, advertising screens and bulletin boards.

On the screen there are 64 circles. And inside each circle has an LED light. There are 64 LEDs, pins on the side, 8 on each. You can see other models like 16x16 Dot matrix, 32x32 Dot matrix.

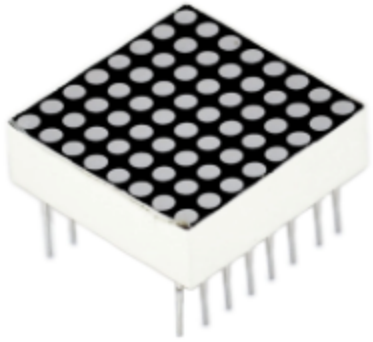
These 64 LEDs can be lit separately, or lit together. Lighten different LED to show different icons.

The single 8x8 dot matrix comes with 8 LEDs on each row and each cols. There are 16 pins on the side, 8 on each.

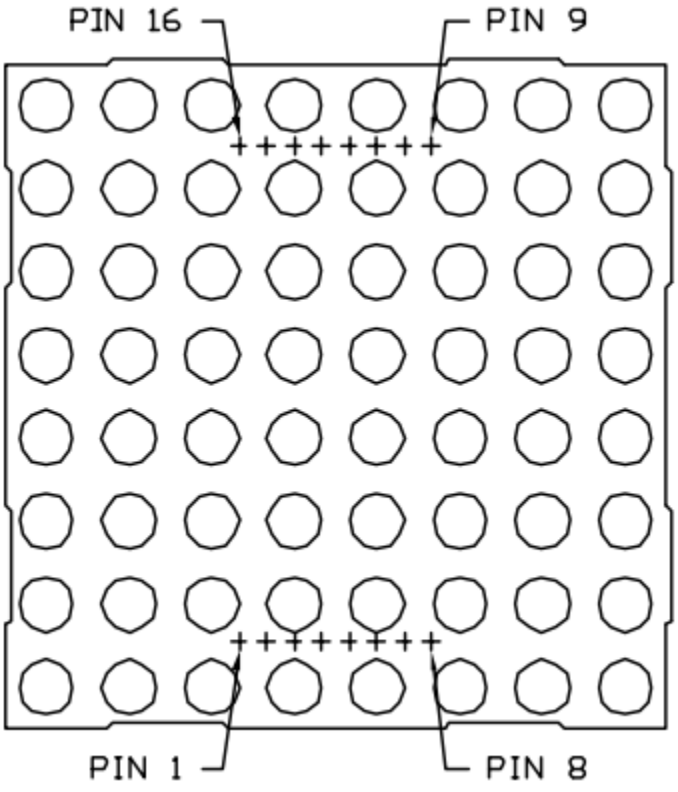
Components

					
Keyestudio PLUS MainBoardx1	8x8 Dot Matrix x1	220 Resistorx8	Jumper Wires	Breadboardx1	USB Cablex1

Component Knowledge



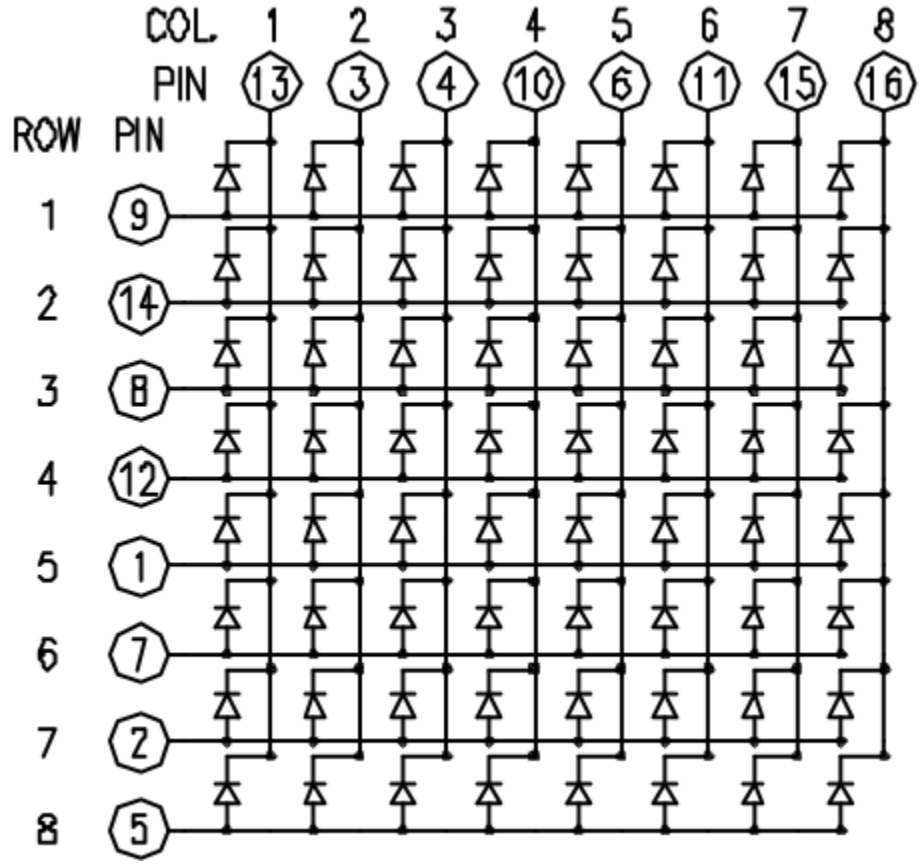
xx8x8 Dot Matrix DisplayxxThe 8x8 lattice consists of 64 leds, each LED placed at the intersection of a row and a column. The external view of the lattice screen is shown below.



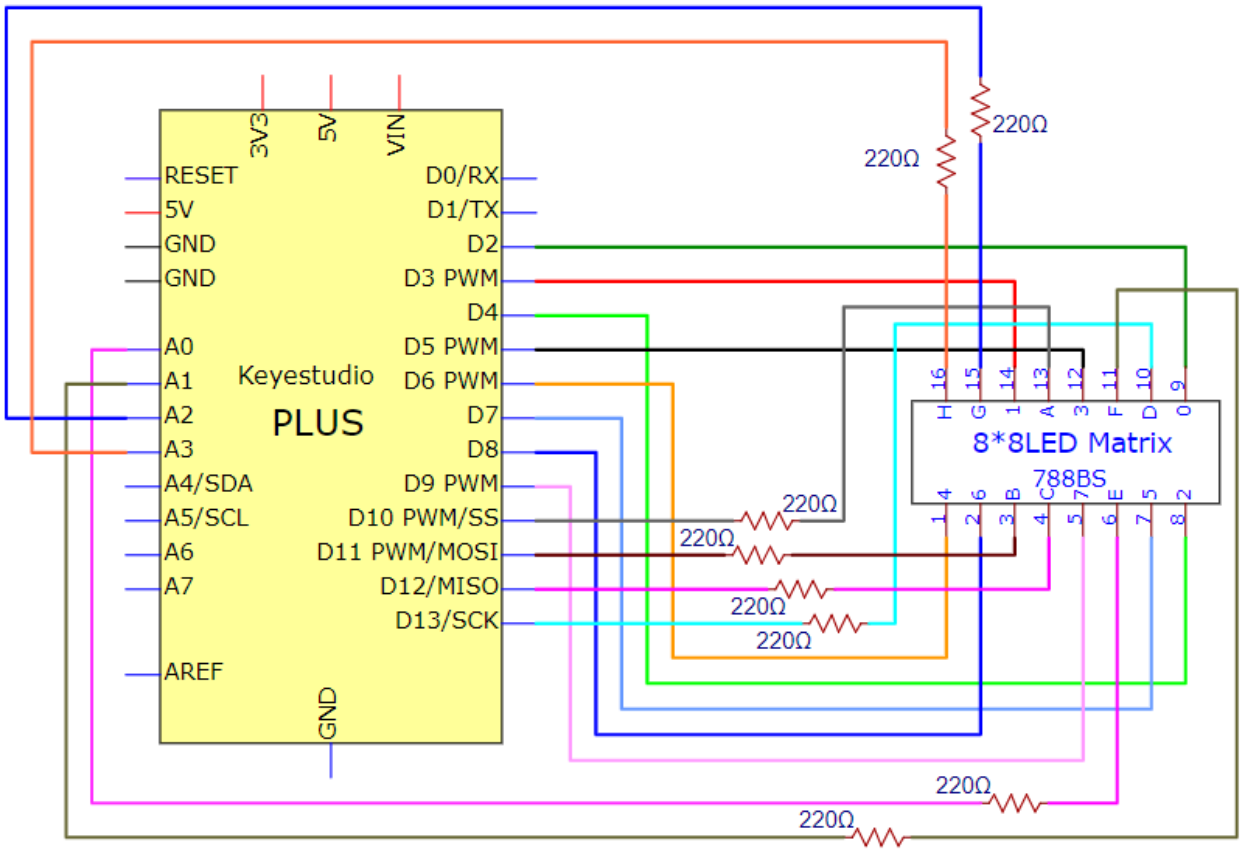
When the level on a row is 1 and the level on a column is 0 then the corresponding LED will light up.

If you want to light up an LED, for example, you can set the pin 9 to High and pin 13 to Low. If you want to make a row of LEDs light up, you should set the pin 9 to High and set pin , , , , , , and to Low. Equally, turning on a column of LEDs requires to set the pin 13 to Low, and set the pin , , , , , , and to High.

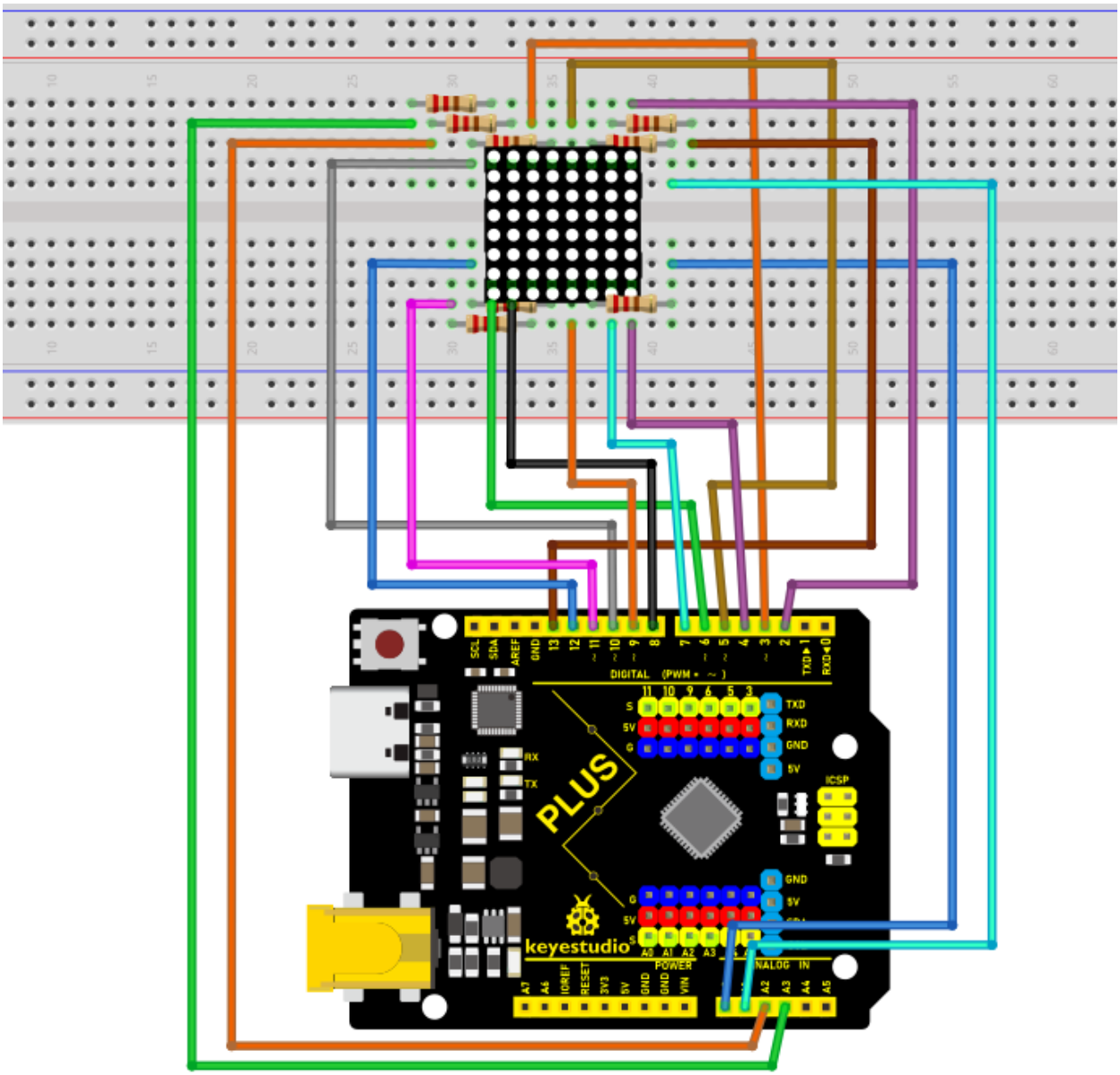
The internal view of the dot matrix screen is shown below



Circuit diagram and wiring diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
220Ω
8*8LED Matrix
788BS

Note: make the number 788BS on the dot matrix face up



Test Code

```
/*

Keyestudio 2021 starter kit

Project 12

12_8_8_Matrix

http//www.keyestudio.com

*/

int R[] = {2,3,4,5,6,7,8,9};
```

(continues on next page)

(continued from previous page)

```
int C[] = {10,11,12,13,A0,A1,A2,A3};
unsigned char data_0[8][8] =
{
{0,0,1,1,1,0,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,0,1,1,1,0,0,0}
};
unsigned char data_1[8][8] =
{
{0,0,0,0,1,0,0,0},
{0,0,0,1,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,1,1,0,0}
};
unsigned char data_2[8][8] =
{
{0,0,1,1,1,0,0,0},
```

(continues on next page)

(continued from previous page)

```
{0,1,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};
unsigned char data_3[8][8] =
{
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};
unsigned char data_4[8][8] =
{
{0,1,0,0,0,0,0,0},
{0,1,0,0,1,0,0,0},
{0,1,0,0,1,0,0,0},
{0,1,1,1,1,1,1,0},
{0,0,0,0,1,0,0,0},
```

(continues on next page)

(continued from previous page)

```
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,0,0,0,0}
};
unsigned char data_5[8][8] =
{
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};
unsigned char data_6[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};
```

(continues on next page)

(continued from previous page)

```
unsigned char data_7[8][8] =
{
{0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0}
};
unsigned char data_8[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};
unsigned char data_9[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
```

(continues on next page)

(continued from previous page)

```
{0,1,0,0,0,1,0,0},

{0,1,1,1,1,1,0,0},

{0,0,0,0,0,1,0,0},

{0,0,0,0,0,1,0,0},

{0,1,1,1,1,1,0,0},

{0,0,0,0,0,0,0,0}

};

void Display(unsigned char dat[8][8])

{

for(int c = 0; c\<8;c++)

{

digitalWrite(C[c],LOW);

for(int r = 0;r\<8;r++)

{

digitalWrite(R[r],dat[r][c]);

}

delay(1);

Clear();

}

}

void Clear()

{

for(int i = 0;i\<8;i++)

{

digitalWrite(R[i],LOW);

digitalWrite(C[i],HIGH);
```

(continues on next page)

(continued from previous page)

```
}

}

void setup(){

for(int i = 0;i\<8;i++)

{

pinMode(R[i],OUTPUT);

pinMode(C[i],OUTPUT);

}

}

void loop(){

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_0);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_1);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_2);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_3);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_4);

}

for (int i = 1; i \<= 100; i = i + (1)) {
```

(continues on next page)

(continued from previous page)

```
Display(data_5);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_6);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_7);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_8);

}

for (int i = 1; i \<= 100; i = i + (1)) {

Display(data_9);

}

}
```

Test Result

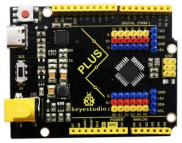
Upload the code, connect the wiring diagram and power up. Then 8x8 dot matrix will show numbers from 0 to 9.

5.13 Project 13: A Desk Lamp

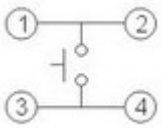
Introduction

Did you know that Arduino can light up an LED when you press a button? In this project, we will use the Plus Mainboard, a key switch and an LED to make a small desk lamp.

Components Required

				
Keyestudio Plus Mainboardx1	Buttonx1	Red LEDx1	10K Resistorx1	Button Capx1
				
Breadboardx1	220 Resistorx1	USB Ca-blex1	Jumper Wires	

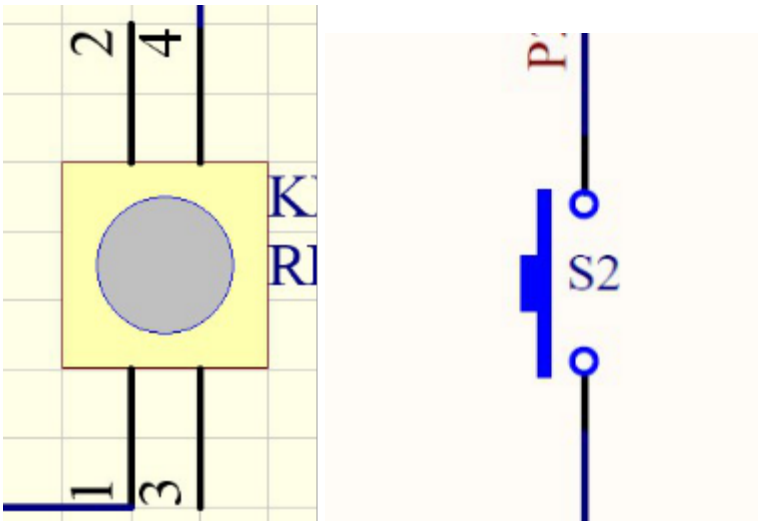
Component Knowledge

Button: The button can control the circuit on and off. The circuit is disconnected when the button is not pressed. But it breaks when you release it. Why does it only work when you press it? It starts from the internal structure of the button, which is shown in the figure: 

. Before the button is pressed, 1 and 2 are on, 3 and 4 are also on, but 1, 3 or 1, 4 or 2, 3 or 2, 4 are off (not working). Only when the button is pressed, 1, 3 or 1, 4 or 2, 3 or 2, 4 are on.

The key switch is one of the most commonly used components in circuit design.

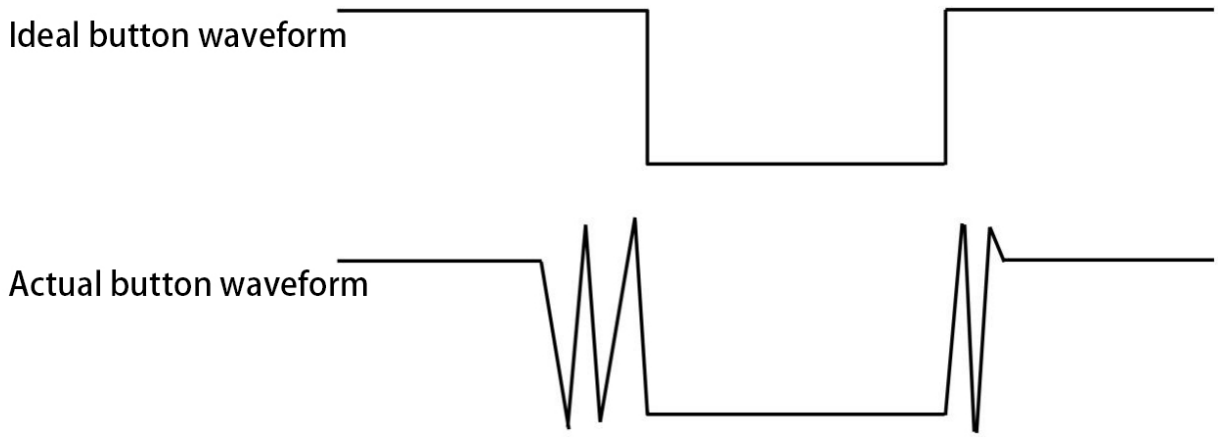
Schematic diagram of the button:



What is button jitter?

We think of the switch circuit as “press the button and turn it on imArduino/mediately”,”press it again and turn it off imArduino/mediately”. In fact, this is not the case.

The button usually uses a mechanical elastic switch, and the mechanical elastic switch will produce a series of jitter due to the elastic action at the moment when the mechanical contact is opened and closed (usually about 10ms). As a result, the button switch will not imArduino/mediately and stably turn on the circuit when it is closed, and it will not be completely and instantaneously disconnected when it is turned off.

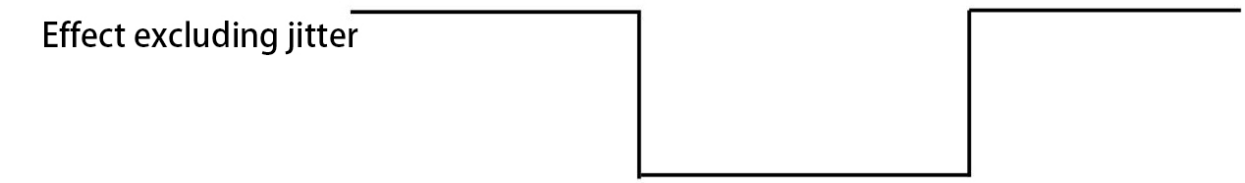


How to eliminate the jitter?

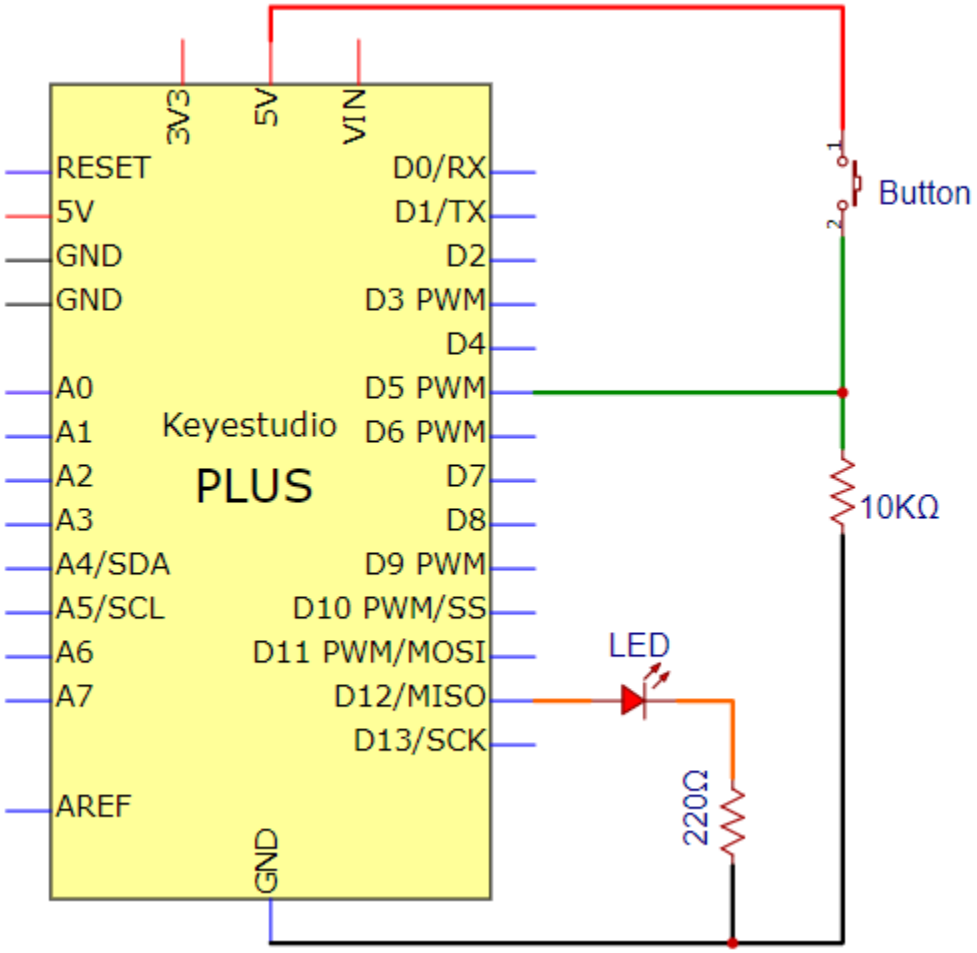
There are two common methods, namely fix jitter in the software and hardware. We only discuss the jitter removal in the software.

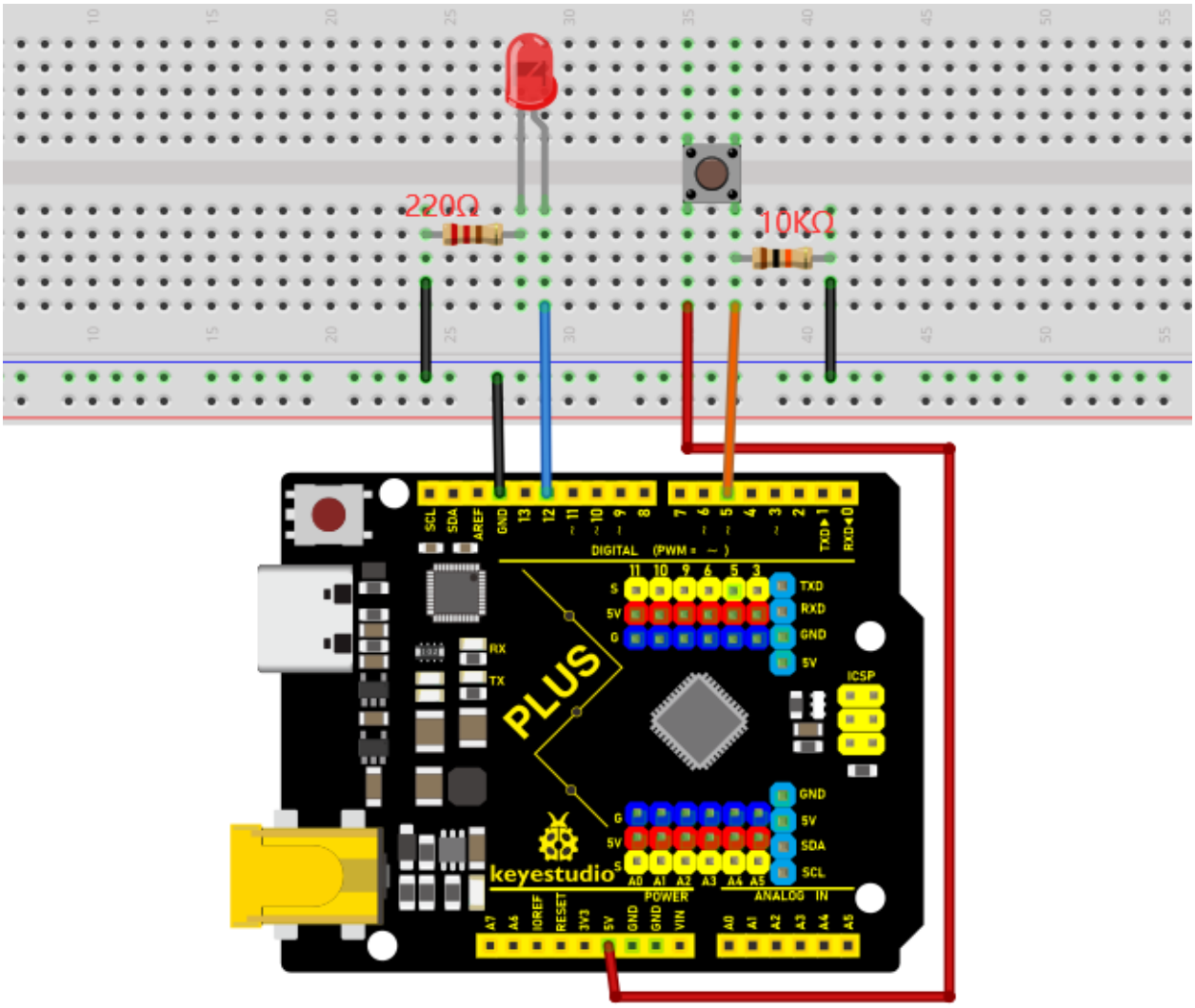
We already know that the jitter time generated by elasticity is about 10ms, and the delay command can be used to delay the execution time of the command to achieve the effect of jitter removal.

Therefore, we delay 0.05s in the code to achieve the key anti-shake function.



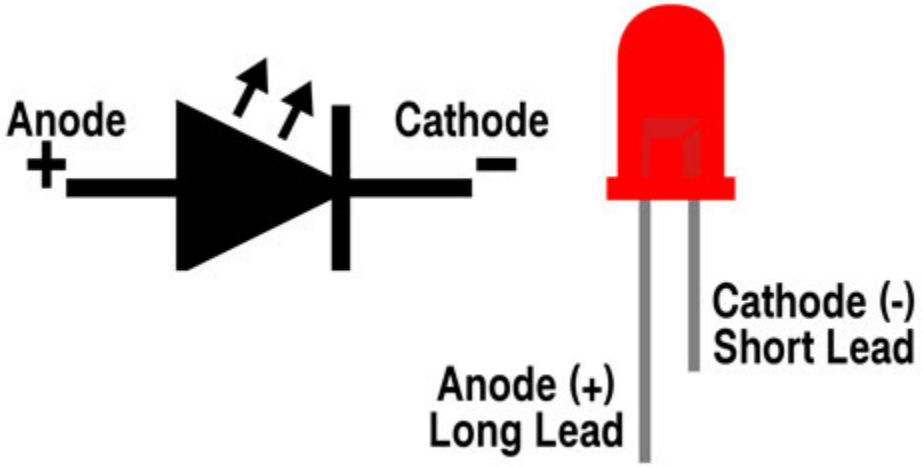
Circuit Diagram and Wiring Diagram



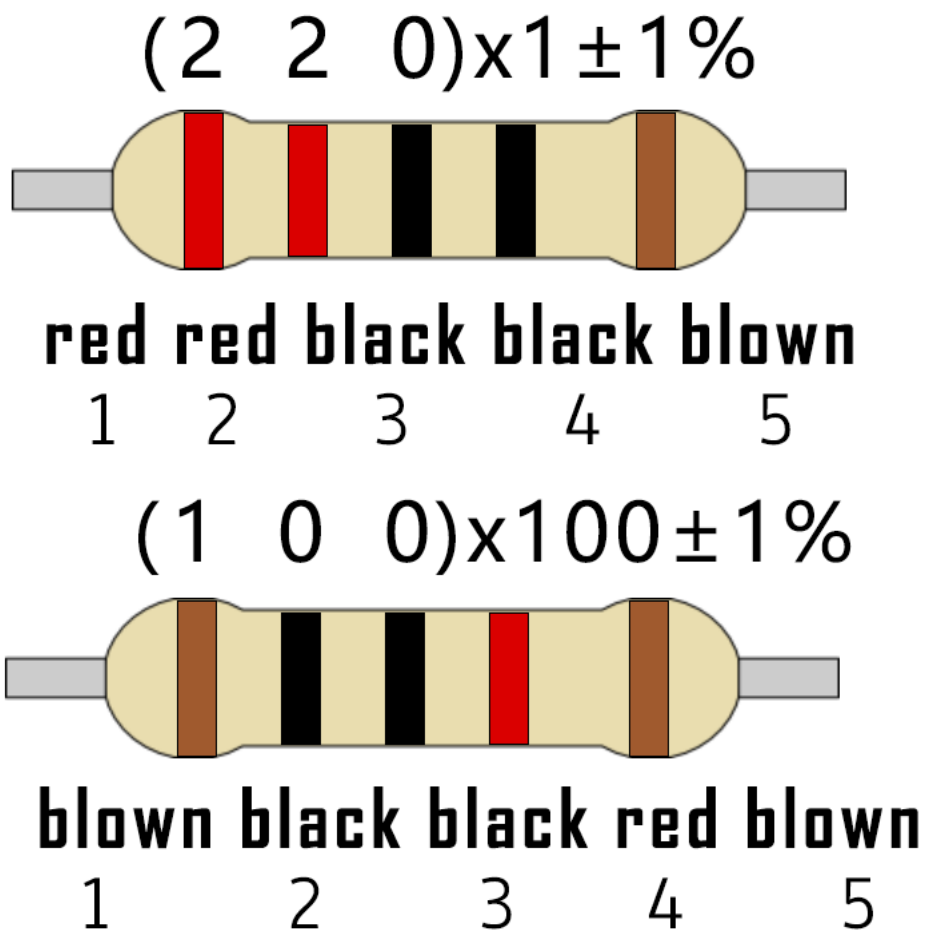


Note:

How to connect the LED



How to identify the 220 5-band resistor and 10K 5-band resistor



Code

```
/*

Keyestudio 2021 starter learning kit

Project 13

Small_Desk_Lamp

http//www.keyestudio.com

*/

int buttonPin = 5; //the button is connected to 5

int ledPin = 12; // LED is interfaced with 12

int ledState = LOW; // ledState records the state of the LED

int buttonState; // buttonState records the state of the button

int lastButtonState = LOW; // lastbuttonState the state that the button is
```

(continues on next page)

(continued from previous page)

```
pressed before

long lastDebounceTime = 0;

long debounceDelay = 50;

void setup() {

pinMode(buttonPin, INPUT);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, ledState);

}

void loop() {

//reading is used to save the data of the buttonPin

int reading = digitalRead(buttonPin);

//record the current timee once the data changes

if (reading != lastButtonState) {

lastDebounceTime= millis();

}

// wait for 50ms and determine again to make sure whether the state is as same as the␣
↪state of the button

// if not, change the state of the button

// at same time, if the state of the button is highpressedthen change the
state of the led

if ((millis() - lastDebounceTime) \>debounceDelay) {

if (reading != buttonState) {

buttonState = reading;

if (buttonState == HIGH) {

ledState= !ledState;

}

}
```

(continues on next page)

(continued from previous page)

```
}

digitalWrite(ledPin, ledState);

// chnage the previous state value of the button

lastButtonState = reading;

}
```

Result

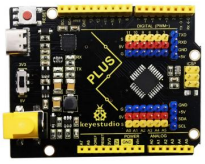
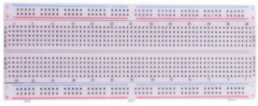
Burn the project code, connect the wires and power on first. Then press the button, the LED will turn on. Press the button again, the LED will turn off.

5.14 Project 14: Electronic Hourglass

Introduction

In this lesson, we will use a PLUS mainboard , a tilt switch and 4 LEDs to make an electronic hourglass.

Components Required

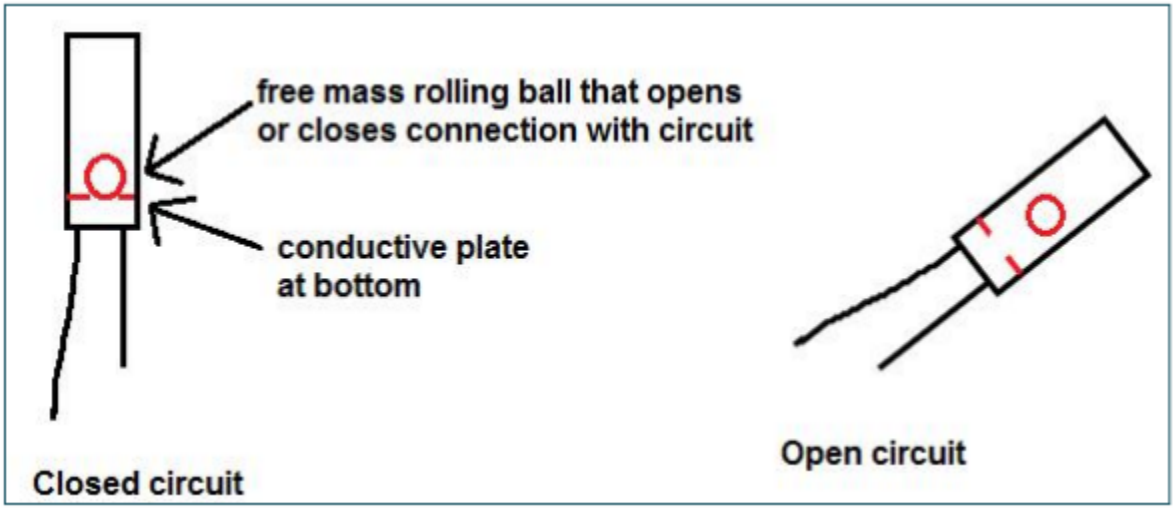
			
Keyestudio Plus Mainboardx1	Tilt Switchx1	Red LEDx4	10K Resistorx1
			
Breadboardx1	220 Resistorx4	USB Cablex1	Jumper Wires

Component Knowledge

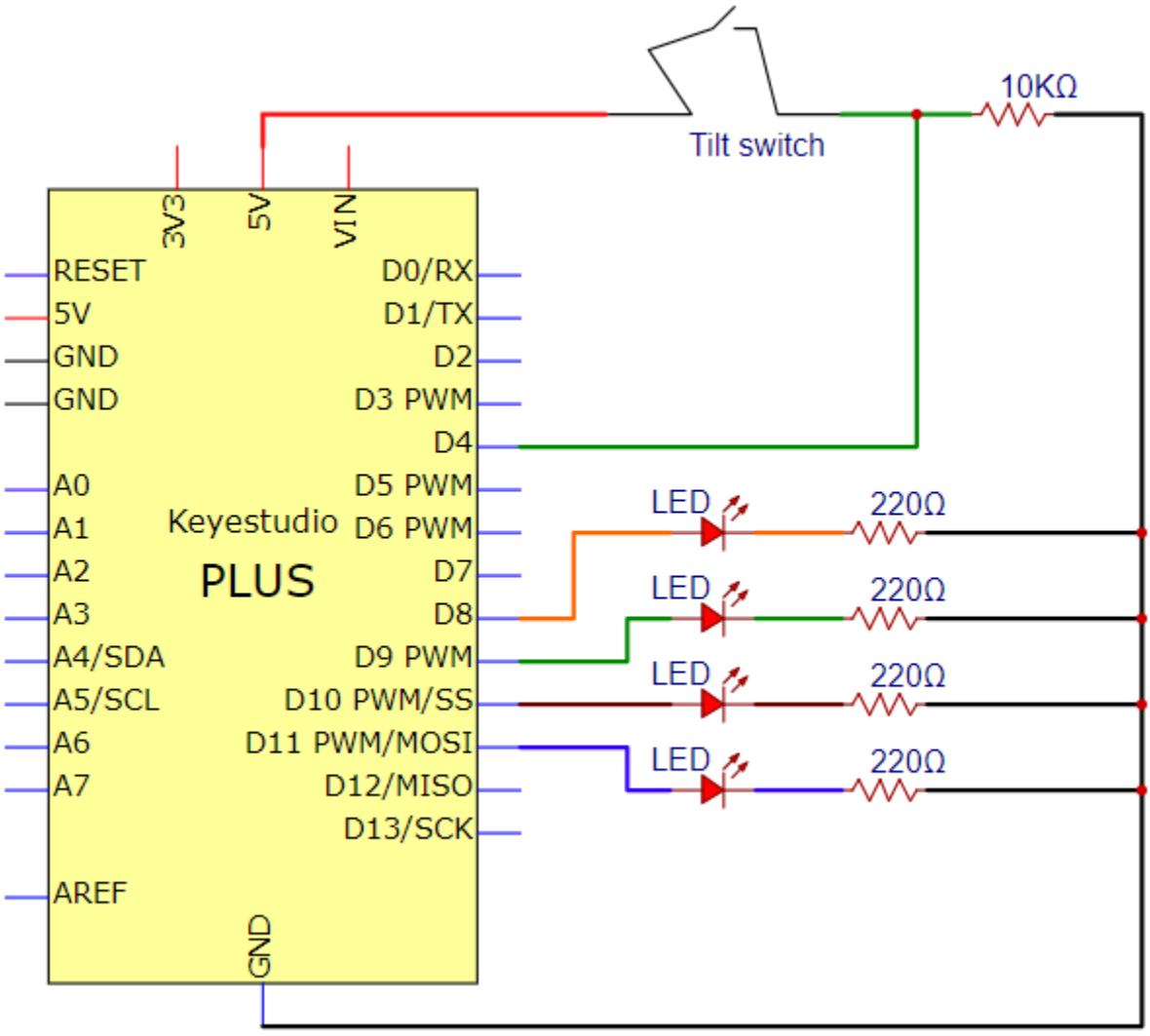
Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger, the tilt sensor is tilted toward the trigger end (two gold-plated pin ends), the tilt switch is in a closed circuit and the voltage at the analog port is about 5V(binary number is 1023). In this way, the LED will light up.

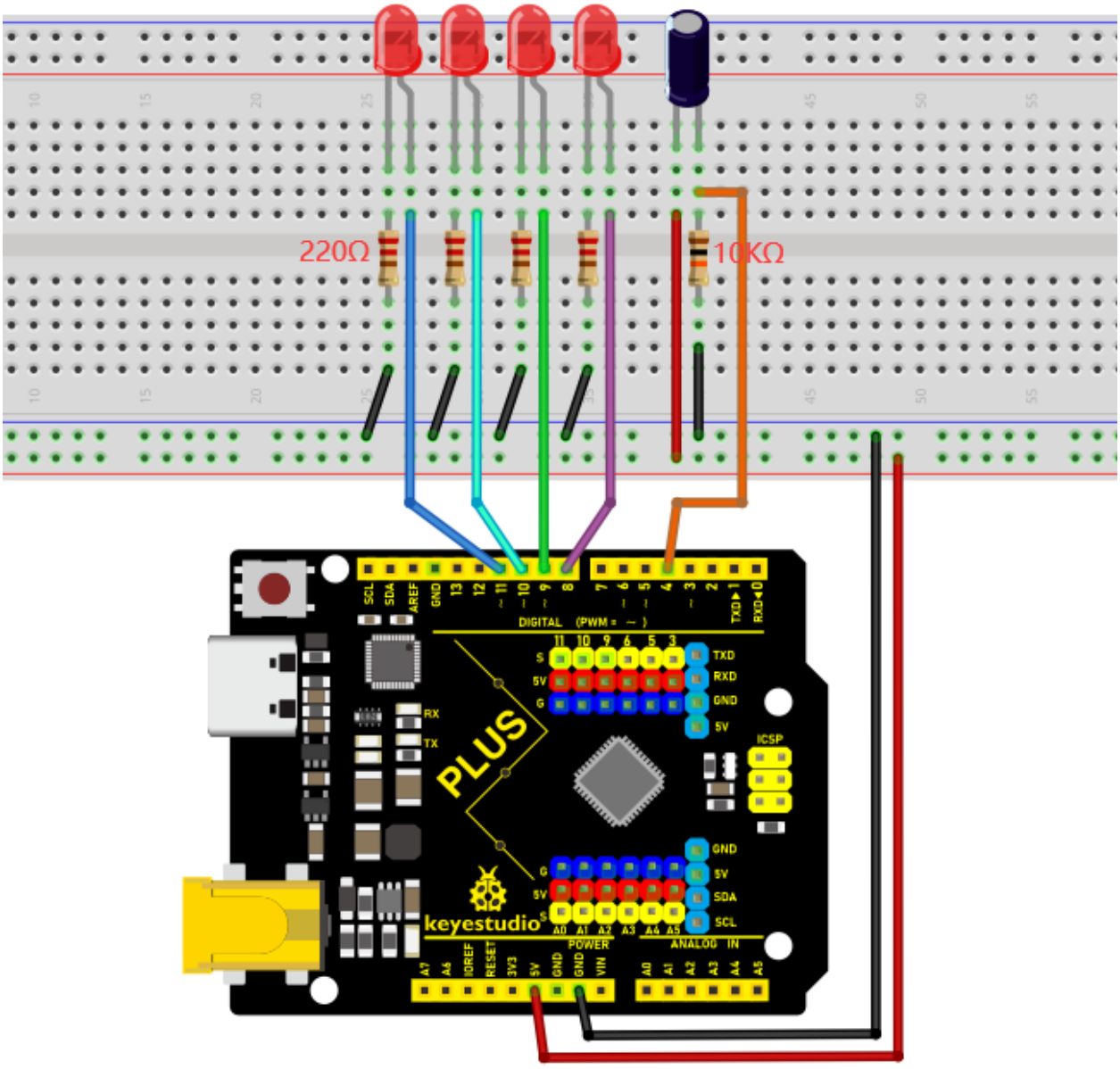
When the tilt switch is in a horizontal position or tilted to the other end, it is open and the voltage of the analog port is about 0V (binary number is 0), the LED will turn off. In the program, we judge the state of the switch based on whether the voltage value of the analog port is greater than 2.5V (binary number is 512).

As shown in the figure, use the internal structure of the tilt switch to illustrate how it works.



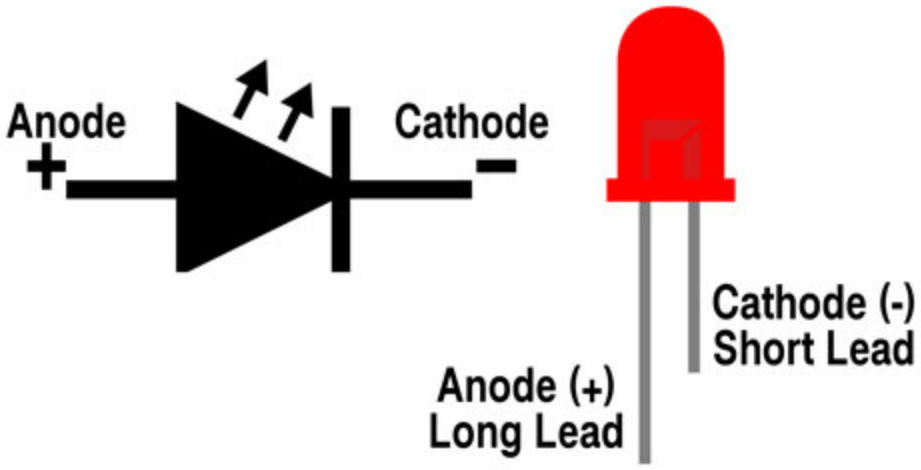
Circuit Diagram and Wiring Diagram



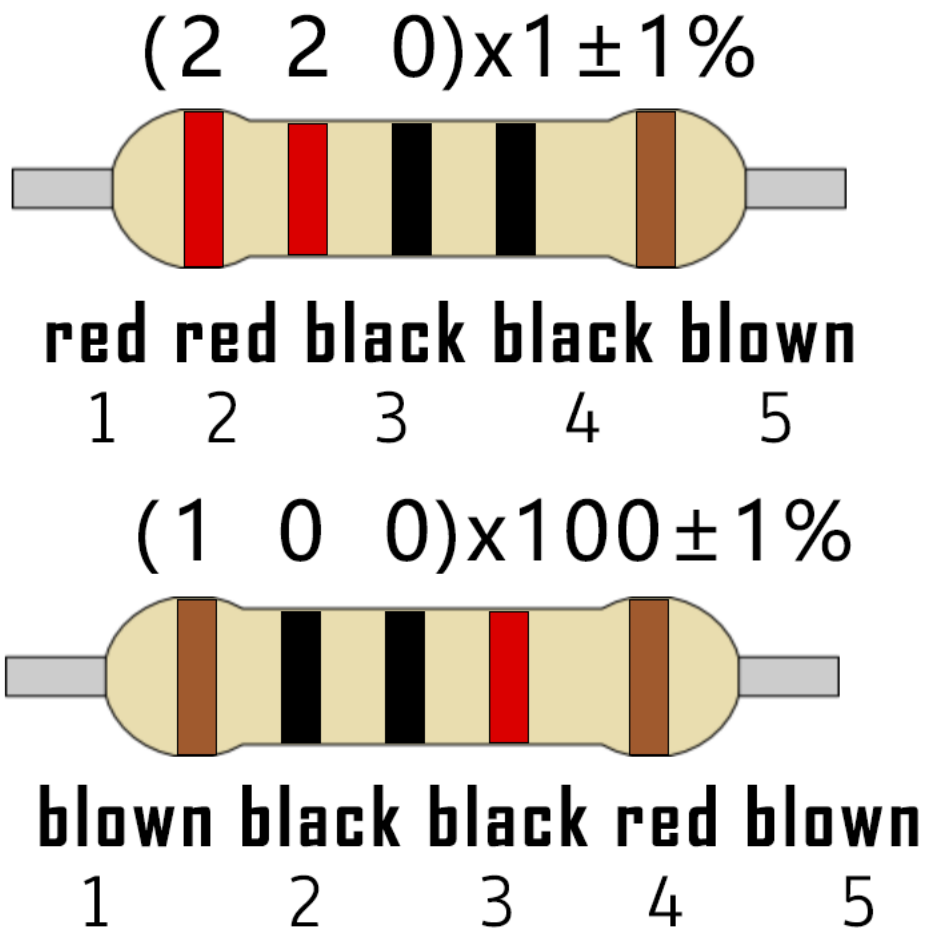


Note:

How to connect the LED



How to identify the 220 5-band resistor and 10K 5-band resistor



Code

```
/*

Keyestudio 2021 starter learning kit
```

(continues on next page)

(continued from previous page)

```
Project 14

Electronic_Hourglass

http//www.keyestudio.com

*/

const byte SWITCH_PIN = 4; // the tilt switch is connected to 4

byte switch_state = 0;

void setup()

{

for(int i=8;i\<12;i++)

{

pinMode(i, OUTPUT);

}

pinMode(SWITCH_PIN, INPUT);

for(int i=8;i\<12;i++)

{

digitalWrite(i,0);

}

Serial.begin(9600);

}

void loop()

{

switch_state = digitalRead(SWITCH_PIN);

Serial.println(switch_state);

if (switch_state == 0)

{

for(int i=8;i\<12;i++)
```

(continues on next page)

(continued from previous page)

```
{
digitalWrite(i,1);
delay(1000);
}
}
if (switch_state == 1)
{
for(int i=11;i\>7;i--)
{
digitalWrite(i,0);
delay(1000);
}
}
}
```

Result

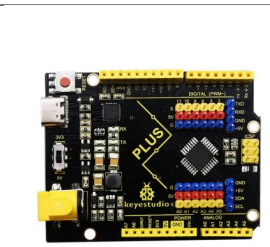
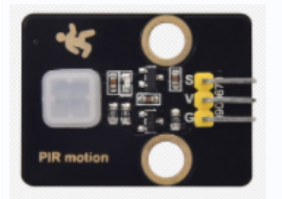
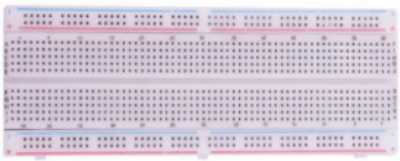
Upload project code, wire up and power up, hold the breadboard. When you tilt the breadboard to any angle, the LEDs will light up one by one. When you turn the breadboard to the original angle, the LEDs will turn off one by one.

5.15 Project 15: PIR Motion Sensor Controls the Buzzer

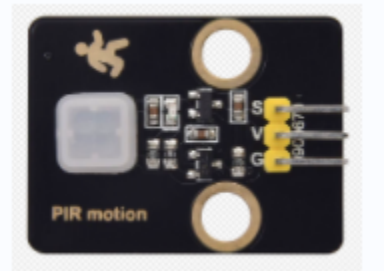
Introduction

PIR motion sensor measures the thermal infrared (IR) light emitted by moving objects. The sensor can detect the movement of people, animals, and cars to trigger safety alarms and lighting. They are used to detect movement and ideal for security such as burglar alarms and security lighting systems. In this project, we will use a PIR motion sensor and buzzer to detect sounds when people or animals approach.

Components Required

			
Keyestudio Plus Mainboardx1	PIR Motion Sensorx1	Active Buzzerx1	Breadboardx1
			
F-F Dupont Wires	USB Cablex1	Jumper Wires	

Component Knowledge



PIR motion sensor: The principle is that when certain crystals, such as lithium tantalate and triglyceride sulfate, are heated, the two ends of the crystal will generate an equal number of charges with opposite signs. These charges can be converted into voltage output by an amplifier. And the human body will release infrared light, although relatively weak, but still can be detected. When the PIR motion sensor detects the movement of a nearby person, the sensor signal terminal outputs a high level 1. Otherwise, it outputs a low level 0. Pay special attention that this sensor can detect people, animals and cars in motion. People, animals and cars at rest cannot be detected. The maximum detection distance is about 7 meters.

Note: Since vulnerable to radio frequency radiation and temperature changes, the PIR motion sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

Features:

Maximum input voltage: DC 3.3 ~ 5V

Maximum operating current: 50MA

Maximum power: 0.3W

Operating temperature: -20 ~ 85℃

Output high level is 3V, low level is 0V.

Delay time: about 2.3 to 3 seconds

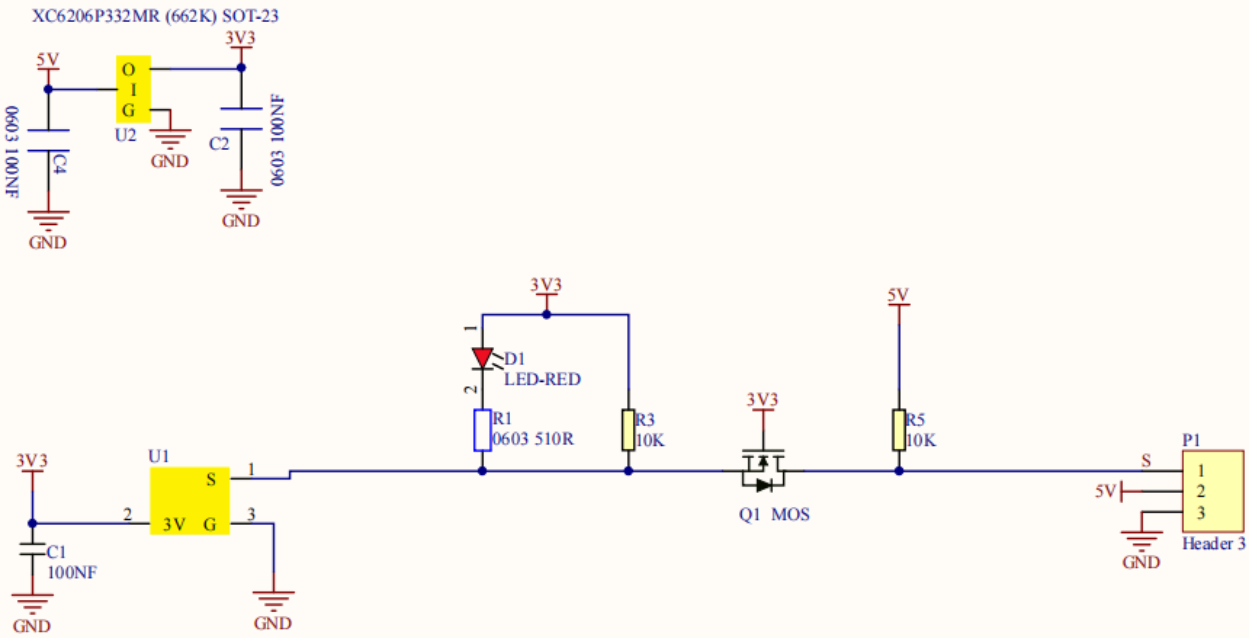
Detection Angle: about 100 degrees

Maximum detection distance: about 7 meters

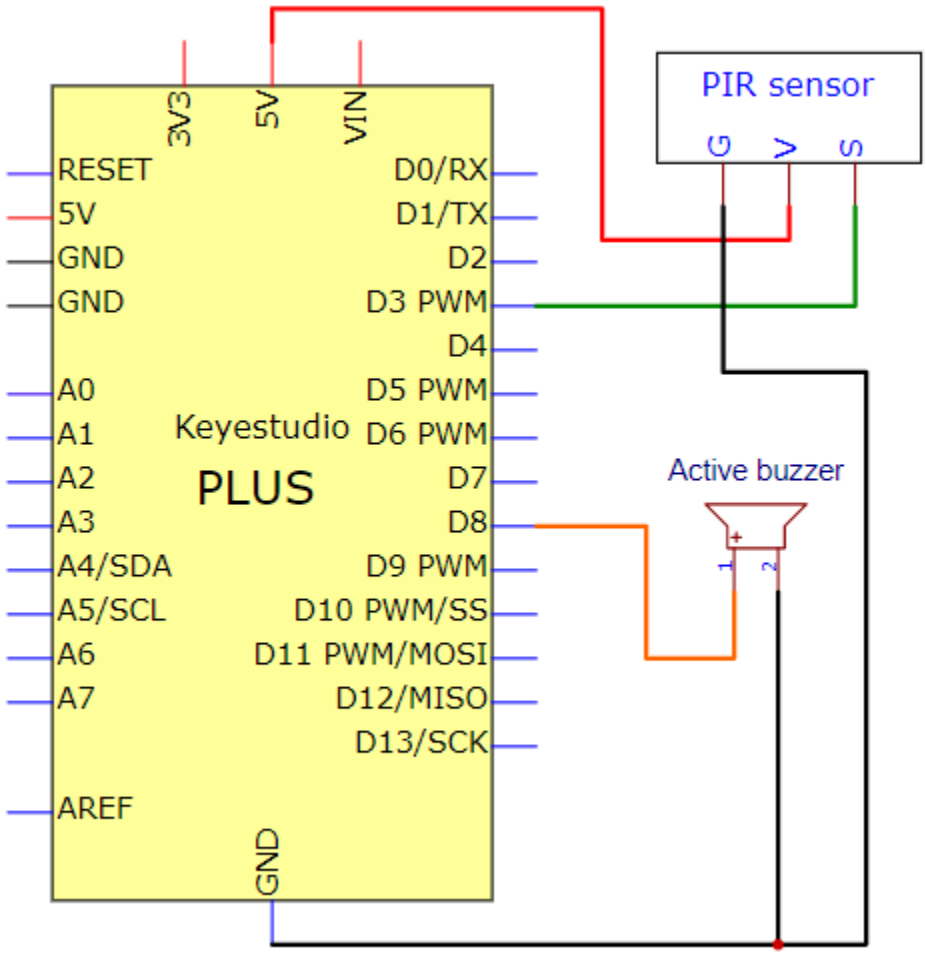
Indicator light output (when the output is high, it will light up)

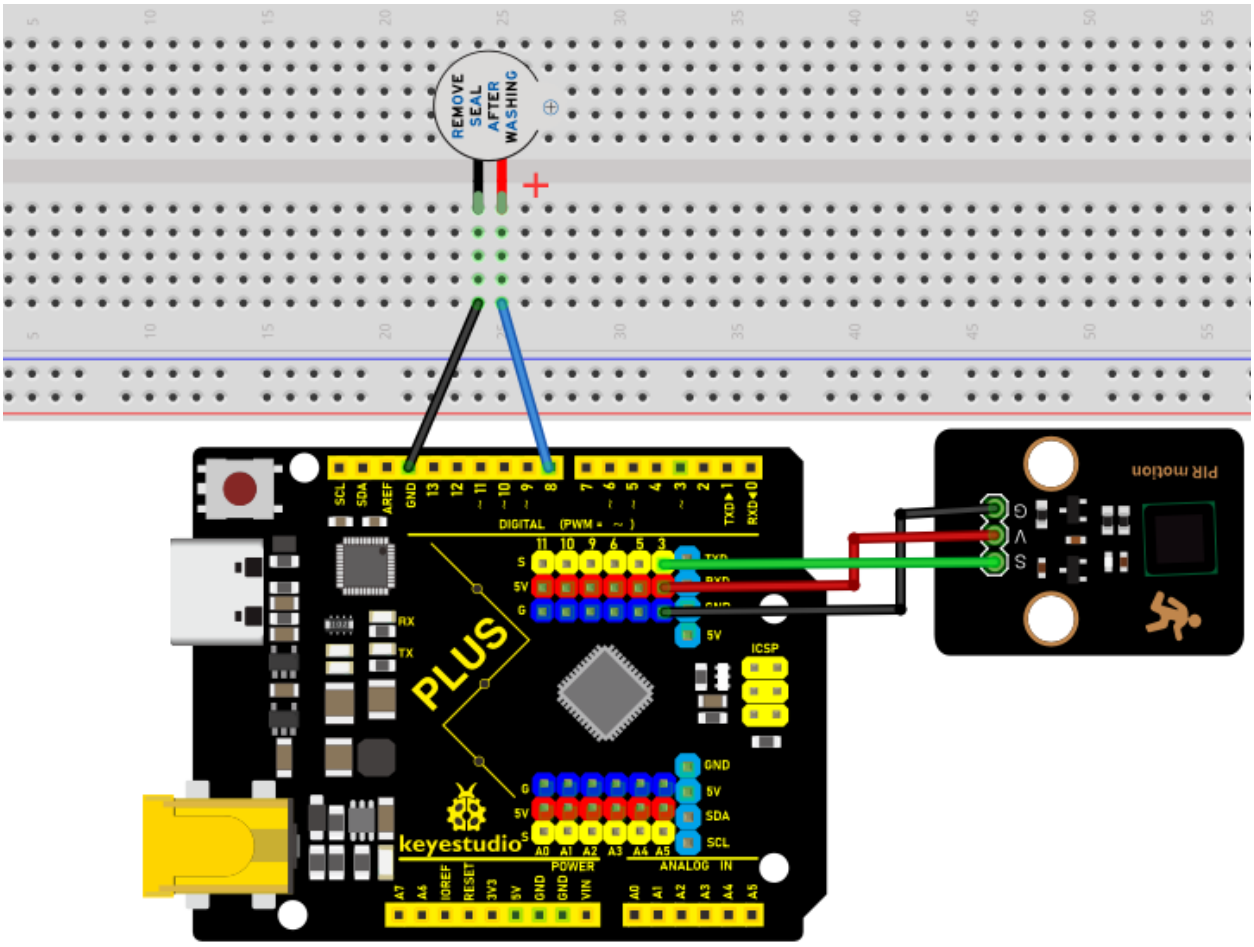
Pin limiting current: 50MA

Schematic diagram:



Circuit Diagram and Wiring Diagram


PIR sensor
G
V
S
3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
Keyestudio
PLUS
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Active buzzer
+
1
2



Code

```
/*

Keyestudio 2021 starter learning kit

Project 15

PIR_control_buzzer

http//www.keyestudio.com

*/

int buzzerpin = 8; // the pin of the buzzer

int pirPin = 3; // the pin of the PIR motion sensor

int pirStat = 0; // the state of the PIR motion sensor

void setup() {

pinMode(buzzerpin, OUTPUT);
```

(continues on next page)

(continued from previous page)

```
pinMode(pirPin, INPUT);

Serial.begin(9600);

}

void loop()

{

pirStat = digitalRead(pirPin);

if (pirStat == HIGH)

{ // if people or moving animals are detected

digitalWrite(buzzerpin, HIGH); // the buzzer chirps

Serial.println("Hey I got you!!!");

}

else {

digitalWrite(buzzerpin, LOW); //if people or moving animals are not detected
turn off buzzers

}

}
```

Result

Burn the project code, wire up and power on first. If the PIR motion sensor detects someone nearby, the buzzer will give an alarm. Click to open the serial monitor on the Arduino IDE, and you will see“Hey I got you ! !”.

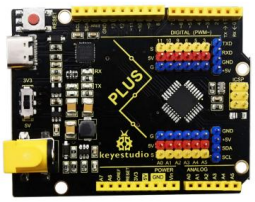
5.16 Project 16: I2C LCD_128X32_DOT

Introduction

We can use modules such as monitors to do various experiments in life. You can also DIY a variety of small objects. For example, you can make a temperature meter with a temperature sensor and display, or make a distance meter with an ultrasonic module and display.

In this project, we will use the LCD_128X32_DOT module as a display and connect it to the Plus control board. The Plus mainboard will be used to control the LCD_128X32_DOT display to show various English characters, common symbols and numbers.

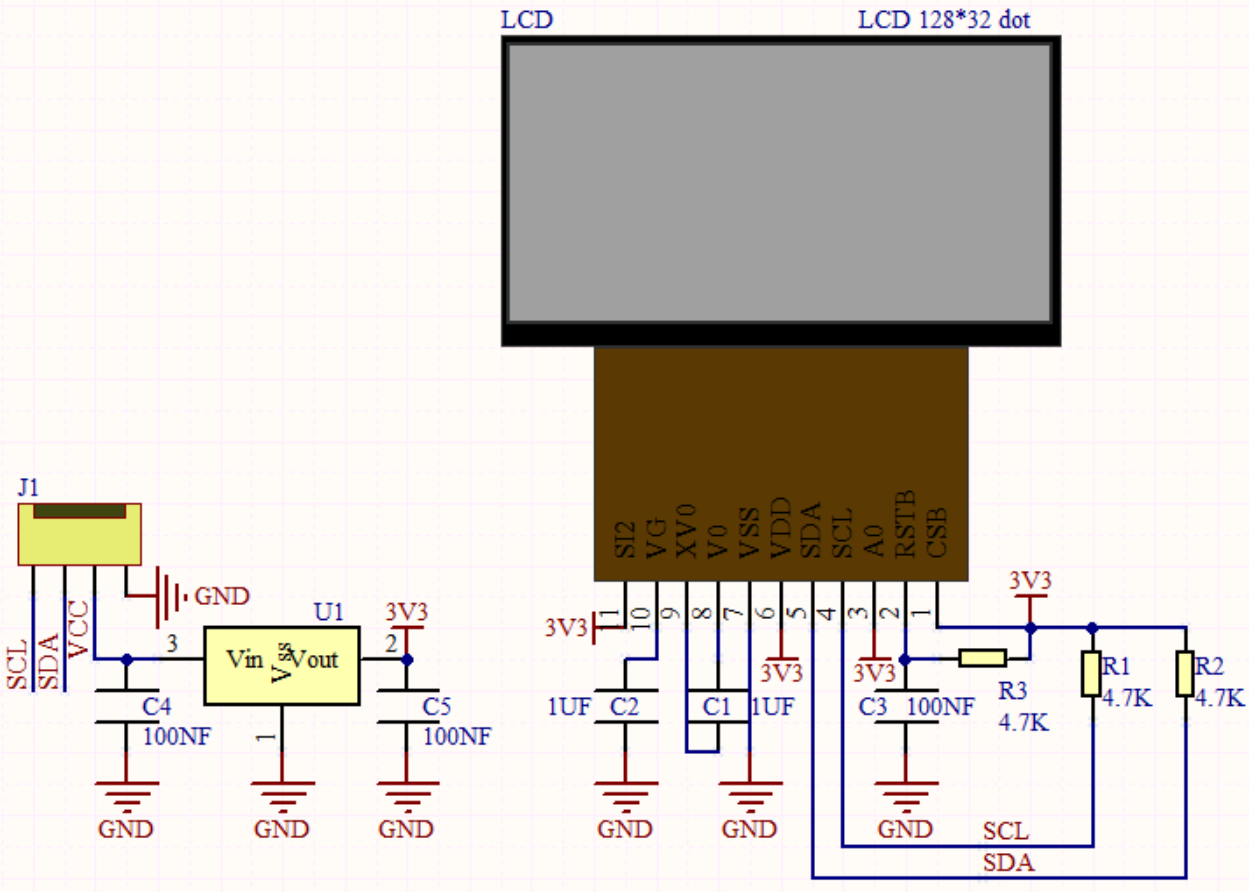
Components Required

			
Keyestudio Plus Mainboardx1	LCD_128X32_DOTx1	F-F Dupont Wires	USB Cablex1

Component Knowledge

LCD_128X32_DOT: It is an LCD module with 128x32 pixels and its driver chip is ST7567A. The module uses the IIC communication mode, while the code contains a library of all alphabets and common symbols that can be called directly. When using, we can also set it in the code so that the English letters and symbols show different text sizes.

Schematic diagram:



Features:

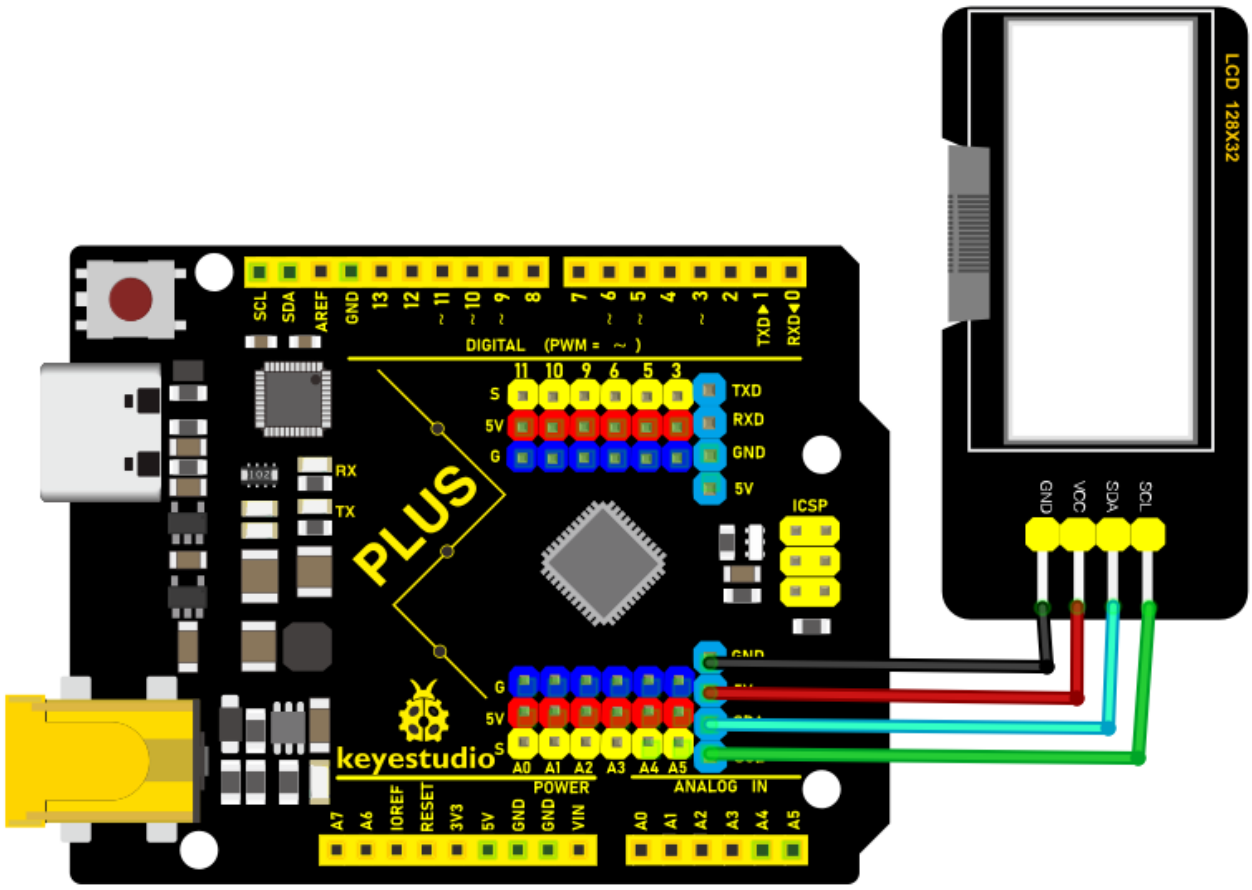
Pixel128x32 character

Operating voltage(chip)4.5V to 5.5V

Operating current100mA (5.0V)

Optimal operating voltage(module):5.0V

Connection Diagram



Code

xxNotexxThe library file is required in the code. If you have already added the "**lcd**"library file, you can ignore the process of adding library files.

Put the decompressed "**LCD_128X32**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

```
/*

Keyestudio 2021 starter learning kit

Project 16

I2C LCD_128X32_DOT

http//www.keyestudio.com

*/

\#include \<lcd.h\> //add library files

lcd Lcd; //define a Lcd class instance
```

(continues on next page)

(continued from previous page)

```
void setup() {

Lcd.Init(); //initialize

Lcd.Clear(); //clear

}

void loop() {

Lcd.Cursor(0, 4); //Set the first row and the eighth column to display,

Lcd.Display("KEYESTUDIO"); //Display KEYESTUDIO, same as below

Lcd.Cursor(1, 0);

Lcd.Display("ABCDEFGHIJKLMNOPQR");

Lcd.Cursor(2, 0);

Lcd.Display("123456789+-*/\<\>=\$@");

Lcd.Cursor(3, 0);

Lcd.Display("%\^&(){}:;'\|?,.\~\\\\[]");

}
```

Result

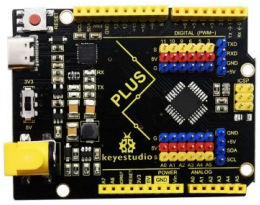
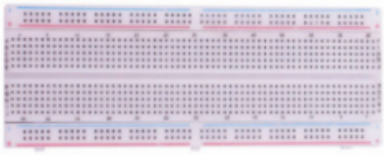
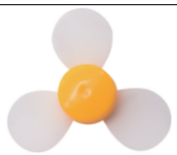
Upload the project code, wire up and power on, the LCD module display will show “KEYESTUDIO” at the first line. “ABCDEFGHIJKLMNOPQR” will be displayed at the second line. “123456789 + - x / <> = $ @ ” will shown at the third line and “% ^ & () { } :; ‘|?,. ~ \ [] ” will be displayed at the fourth line.

5.17 Project 17Small Fan

Introduction

In this lesson, we will make a small fan with a PLUS MainBoard and a DC motor.

Components

			
Keyestudio PLUS Main-Boardx1	L293D Chipx1	DC Motorx1	Breadboardx1
			
USB Cablex1	Jumper Wire	Fanx1	

Component Knowledge:

L293D Chip

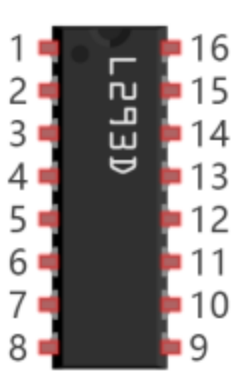
L293D is a direct current drive IC, which can be used to drive DC motor or stepper motor in some robot projects.

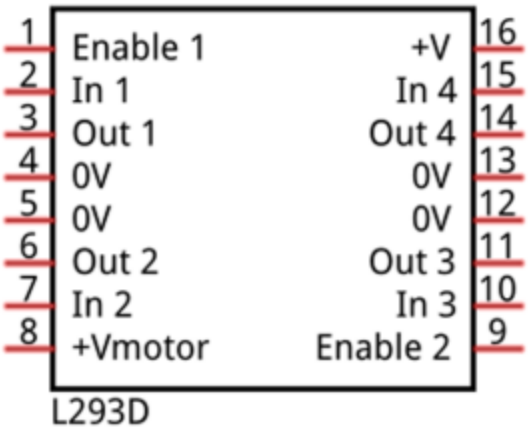
It has a total of 16 pins and can drive two-channel DC motors at the same time.

Its Input voltage range is 4.5 V ~ 36 V, the output current of per channel is MAX 600mA, which can drive inductive loads. What's more, its input end can be directly connected and controlled by the single-chip microcomputer.

When driving a small DC motor, the control of two-channel motors and the forward and reverse rotation can be realized by changing the high and low level of the input terminal. There are many motor drive boards using L293D chips on the market, of course, we can also use it via simply connecting.

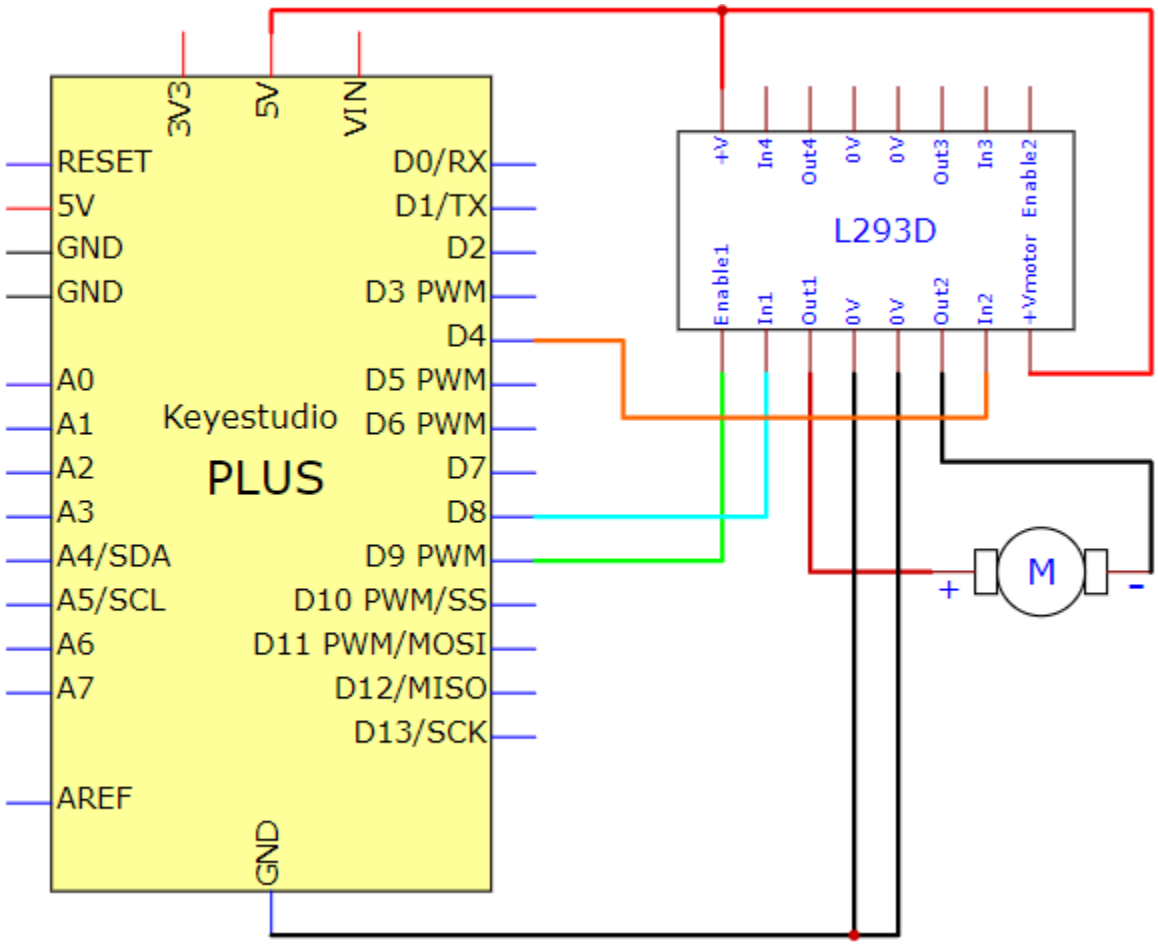
L293D Pin out

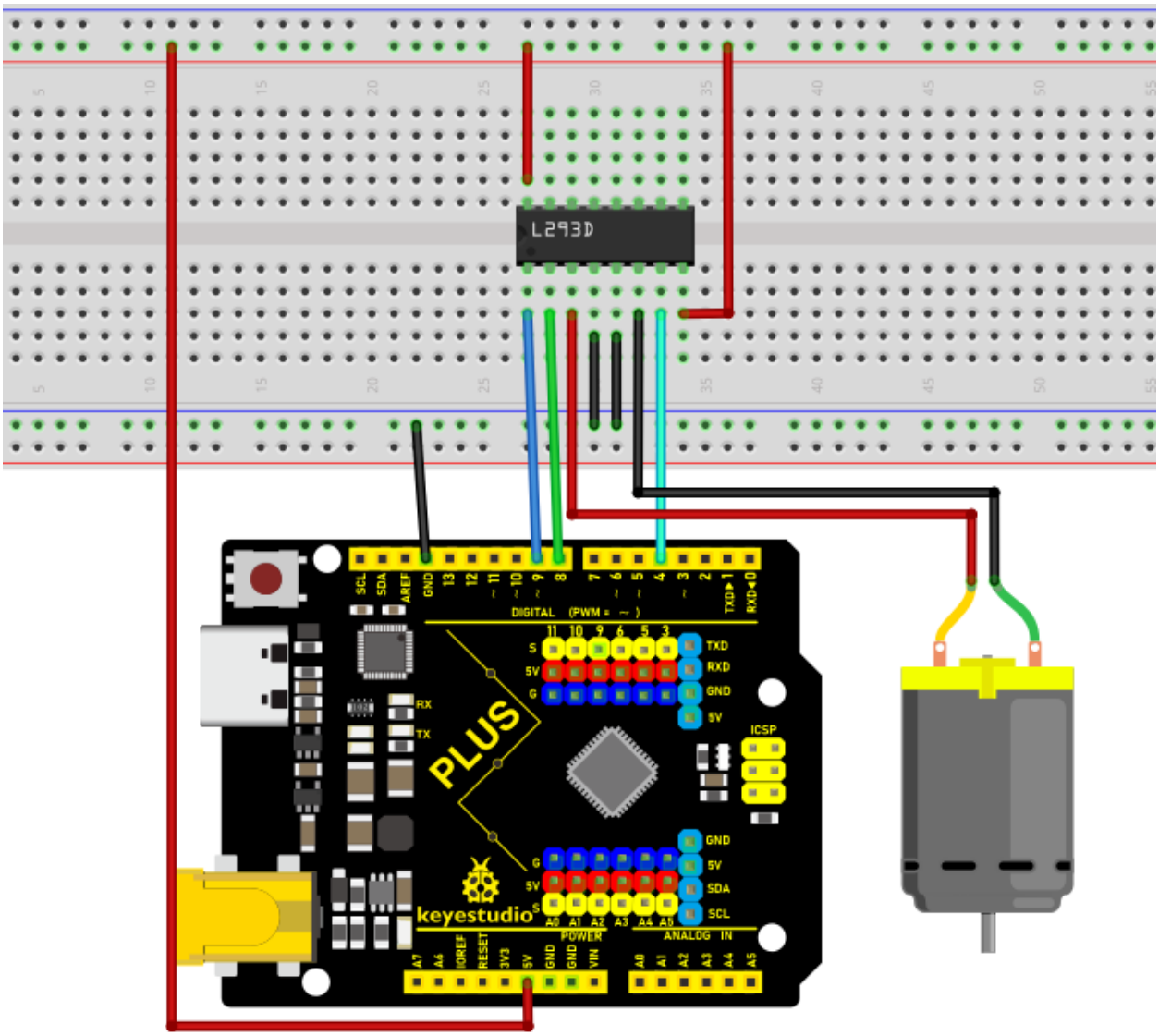




No	Name	Description
1	Enable1,2	Enable pin Input 1(2)and Input 2(7)
2	Input1	Directly input pin 1via digital circuit
3	Output1	Connected to one end of motor1
4	GND	Grounded(0V)
5	GND	Grounded(0V)
6	Output2	Connected to one end of motor1
7	Input2	Directly output pin 2 via digital circuit
8	Vcc2 (Vss)	Connected to voltage pin of motor(4.5V-36V)
9	Enable3,4	Enable pin 3(10)and 4(15)
10	Input3	Input3 pin, controlled by digital circuit
11	Output3	Connected to one end of motor2
12	GND	Grounded(0V)
13	GND	Grounded(0V)
14	Output4	Connected to one end of motor2
15	Input4	Input4 pin, controlled by digital circuit
16	Vcc1(Vss)	Connect + 5V to enable IC function

Circuit diagram and wiring diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
+V
In4
Out4
0V
0V
Out3
In3
Enable2
L293D
Enable1
In1
Out1
0V
0V
Out2
In2
+Vmotor
M
+
-



Test Code

```
/*
Keyestudio 2021 Starter Kit

Project 17

Small_Fan

http//www.keyestudio.com

*/

int IN1=8;

int IN2=4;

int ENA=9;
```

(continues on next page)

(continued from previous page)

```
void setup()
{
pinMode(IN1,OUTPUT);
pinMode(IN2,OUTPUT);
pinMode(ENA,OUTPUT);
}
void loop()
{
//rotate clockwise
digitalWrite(IN1,LOW);
digitalWrite(IN2,HIGH);
analogWrite(ENA,200);
delay(3000);
//delay in 3s
analogWrite(ENA,0);
delay(1000);
//rotate anticlockwise
digitalWrite(IN1,HIGH);
digitalWrite(IN2,LOW);
analogWrite(ENA,100);
delay(3000);
//delay in 3s
analogWrite(ENA,0);
delay(1000);
}
```

Test Result

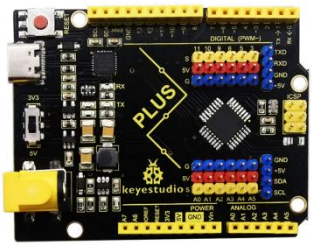
Upload the code to the control board, wire up, install the fan on the motor and power up. Then you can see the fan rotate clockwise for three seconds, stop for one second and anticlockwise for three seconds and stop for one second. Connect ENA to the PWM pin of the plus board, then the speed of the fan can be controlled by the PWM. In the experiment, the clockwise speed is faster than the anticlockwise speed.

5.18 Project 18: Servo Rotation

Introduction

Servo is a kind of motor that can rotate very precisely. It has been widely used in toy cars, RC helicopters, airplanes, robots, etc. In this project, we will use the PLUS mainboard to control the rotation of the servo.

Components Required

		
Keyestudio Plus Mainboardx1	Servox1	USB Cablex1

Component Knowledge

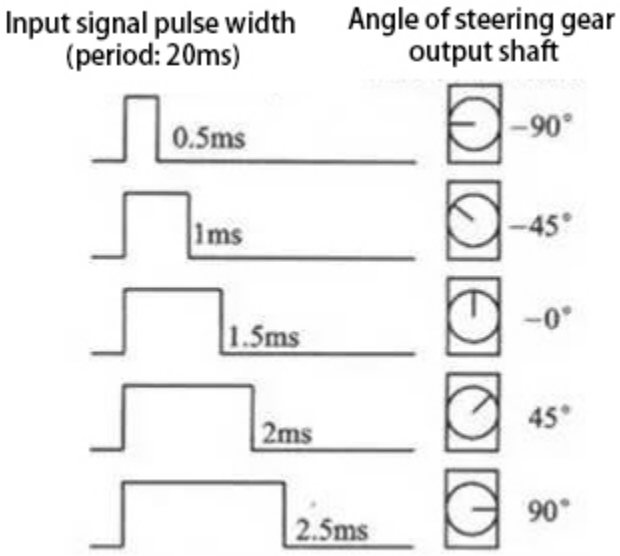
Servo:

The servo is a kind of position servo driver, which is mainly composed of housing, circuit board, coreless motor, gear and position detector. The working principle is that the receiver or microcontroller sends a signal to the servo, which has an internal reference circuit that generates a reference signal with a period of 20ms and a width of 1.5ms, and compares the DC bias voltage with the voltage of the potentiometer to output voltage difference.

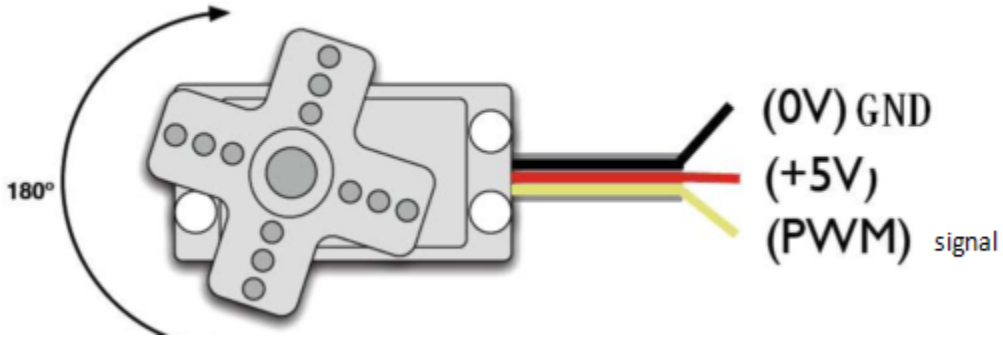
The IC on the circuit board determines the direction of rotation, and then drives the coreless motor to start rotation and transmits the power to the swing arm through the reduction gear, while the position detector sends back a signal to determine whether it has reached the positioning. It is suitable for those control systems that require constant change of angle and can be maintained.

When the motor rotates at a certain speed, the potentiometer is driven by the cascade reduction gear to rotate so that the voltage difference is 0 and the motor stops rotating. The angle range of general servo rotation is 0 to 180 degrees.

The pulse period for controlling the servo is 20ms, the pulse width is 0.5ms to 2.5ms, and the corresponding position is -90° to +90°. The following is an example of a 180 degree servo.

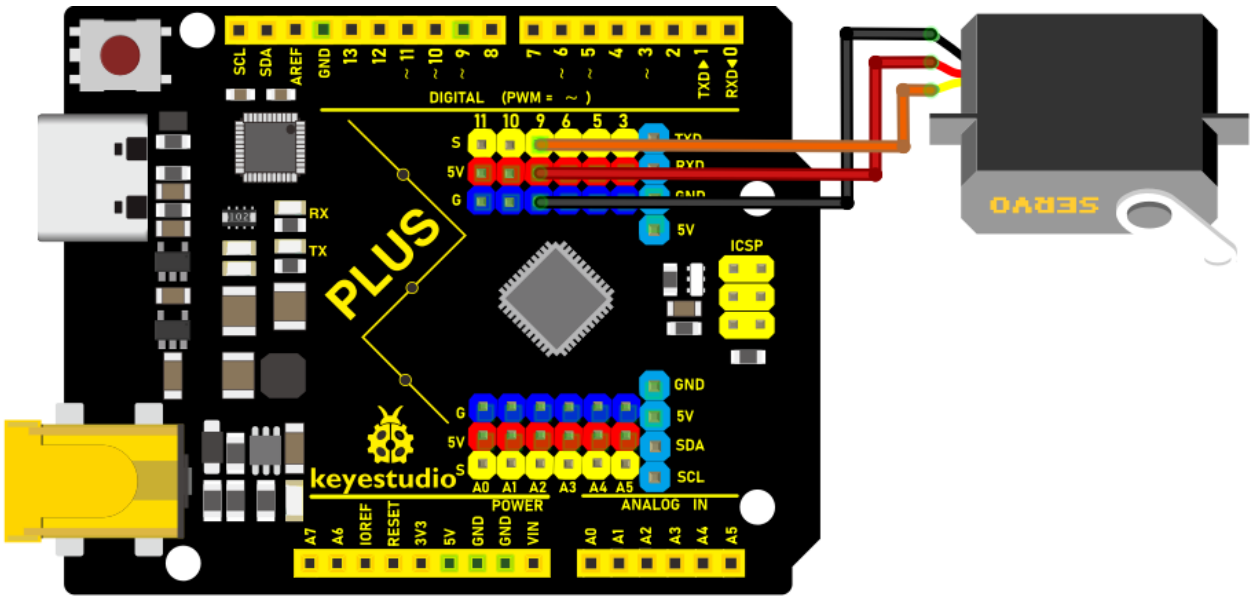


Servo motors have many specifications, but they all have three connecting wires, which are brown, red, and orange (different brands may have different colors). The brown is GND, the red is the positive power supply, and the orange is the signal line.



Wiring Diagram

Servo	Plus mainboard
Red line	5V
Brown line	G
Orange line	9(S)



Code

xxNotexxThe library files need to be installed in the code. If you have already added the Servo library files, ignore the process of adding the library files below.

Decompress the library files in the folder, that is, put the decompressed "**Servo**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

```
/*

Keyestudio 2021 starter learning kit

Project 18

Servo Rotation

http//www.keyestudio.com

*/

\#include \<Servo.h\>

Servo myservo;// define the name of the servo

void setup()

{

myservo.attach(9);// select the pin of the servo(9)

}
```

(continues on next page)

(continued from previous page)

```
void loop()

{

myservo.write(0);// set the rotation angle of the motor

delay(500);

myservo.write(45);// set the rotation angle of the motor

delay(500);

myservo.write(90);// set the rotation angle of the motor

delay(500);

myservo.write(135);// set the rotation angle of the motor

delay(500);

myservo.write(180);// set the rotation angle of the motor

delay(500);

}
```

Result

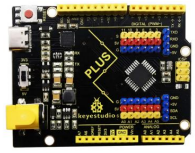
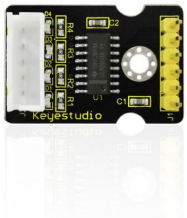
After upload the code to the Plus Mainboard, wire up and power on, the servo will rotate 0°, 45°, 90°, 135°, and 180°.

5.19 Project 19: Stepper Motor

Introduction

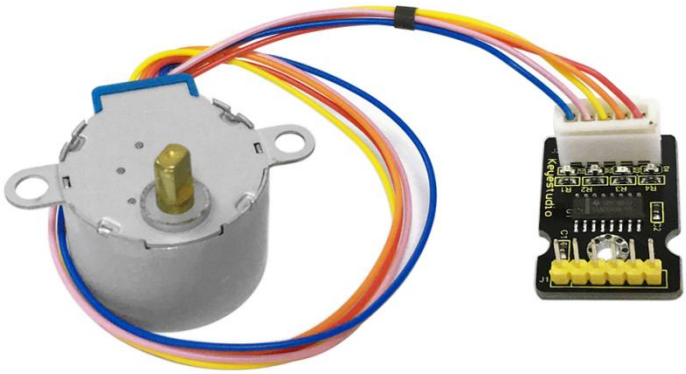
Stepper motors are accurately positioned and are the most important components in industrial robots, 3D printers, large lathes, and other mechanical devices. In this project, we will use a stepper motor and a clock paper card to make a clock model.

Components Required

				
Keyestudio Plus Mainboardx1	ULN2003 Stepper Motor Drive Boardx1	Stepper Motor x1	M-F Dupont Wires	USB Cablex1

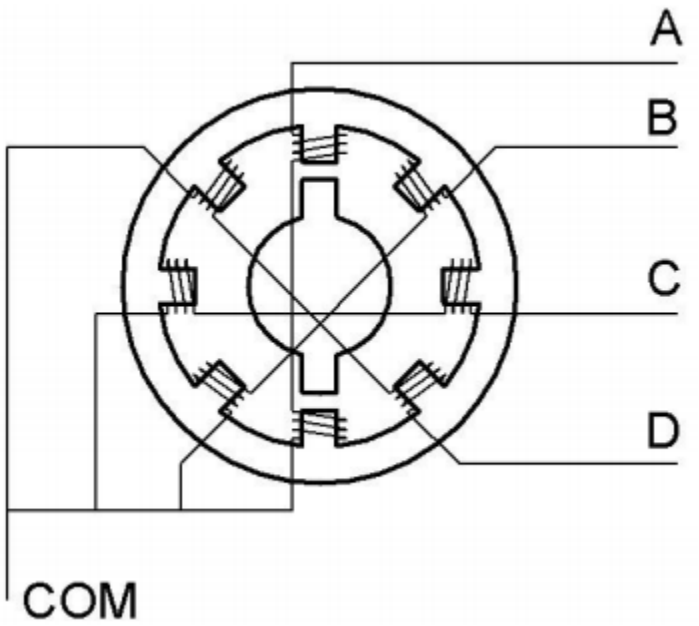
Component Knowledge

Stepper motor: It is a motor controlled by a series of electromagnetic coils. It can rotate by the exact number of degrees (or steps) needed, allowing you to move it to a precise position and keep it there. It does this by supplying power to the coil inside the motor in a very short time, but you must always supply power to the motor to keep it in the position you want. There are two basic types of stepping motors, namely unipolar stepping motor and bipolar stepping motor. In this project, we use a 28-BYJ48 unipolar stepper motor.



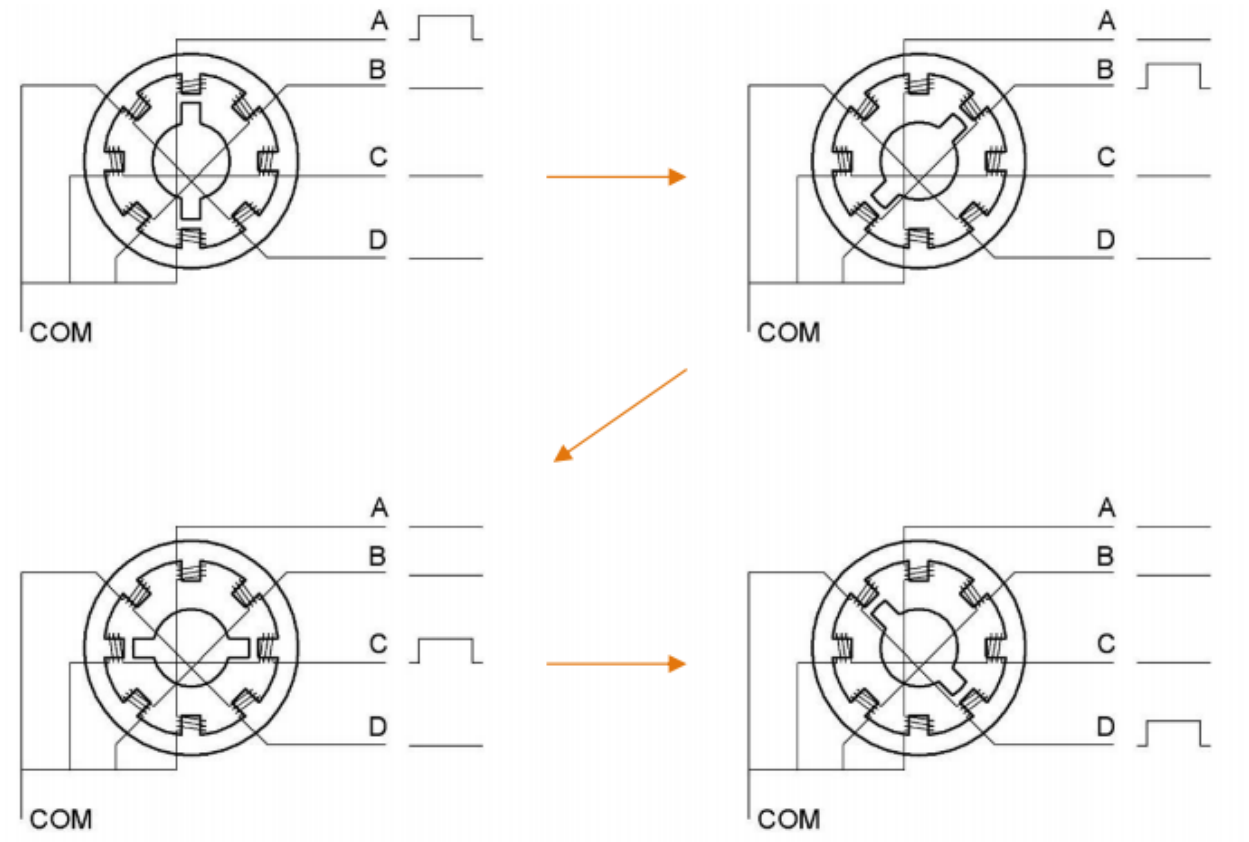
Working Principle:

The stepper motor is mainly composed of a stator and a rotor. The stator is fixed. As shown in the figure below, the part of the coil group A, B, C, and D will generate a magnetic field when the coil group is energized. The rotor is the rotating part. As follows, the middle part of the stator, two poles are permanent magnets.



Single -phase four beat: At the beginning, the coils of group A are turned on, and the poles of the rotor point at A coil. Next, the group A coil are disconnected, and the group B coils are turned on. The rotor will turn clockwise to the group B. Then, group B is disconnected, group C is turned on, and the rotor is turned to group C. After that, group C is disconnected, and group D is turned on, and the rotor is turned to group D. Finally, group D is disconnected, group A is turned on, and the rotor is turned to group A coils. Therefore, rotor turns 180° and continuously rotates B-C-D-A, which means it runs a circle (eight phase). As shown below, the rotation principle of stepper motor is A - B - C - D-A.

You make order inverse(D - C - B - A - D .....) if you want to make stepper motor rotate anticlockwise.



Half-phase and eight beat: 8 beat adopts single and dual beat wayA - AB - B - BC - C - CD - D - DA - A …… rotor will rotate half phase in this order. For example, when A coil is electrifiedrotor faces to A coil then A and B coil are connected, on this condition, the strongest magnetic field produced lies in the central part of AB coil, which means rotating half-phase clockwise.

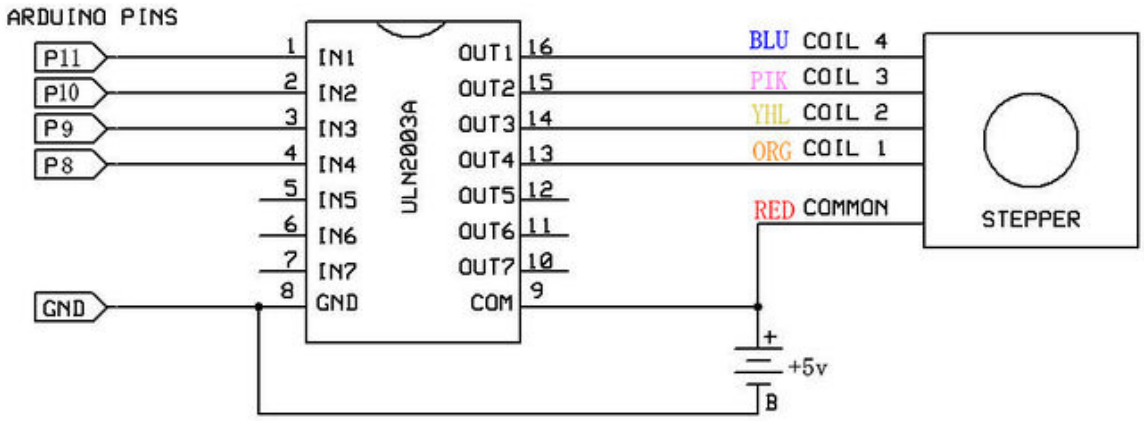
Stepper Motor Parameters:

The rotor rotates one circle when the stepper motor we provide rotates 32 phases and with the output shaft driven by 1:64 reduction geared set. Therefore the rotation (a circle) of output shaft requires 2048 phases.

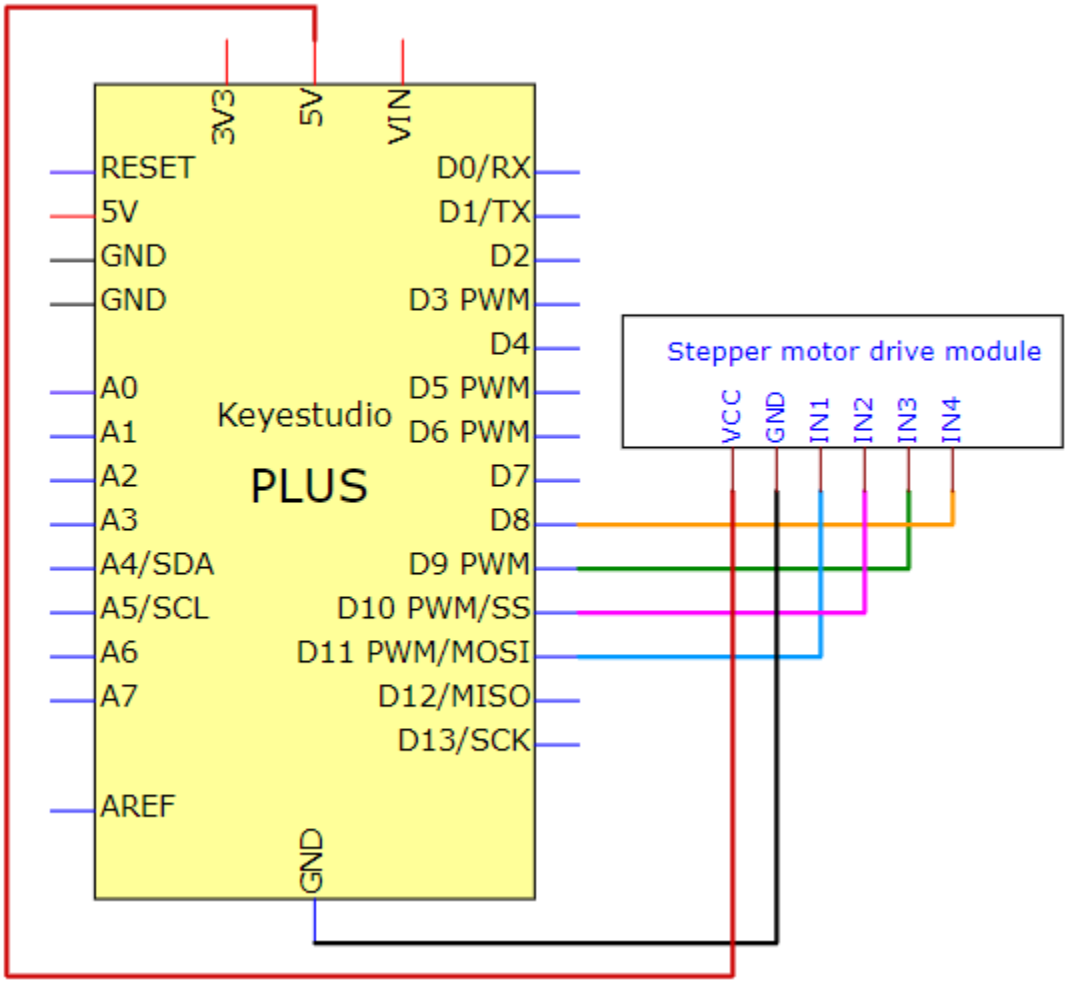
The step angle of 4-beat mode of 5V and 4-phase stepper motor is 11.25. And the step angle of 8-beat mode is 5.625, the reduction ratio is 1:64.

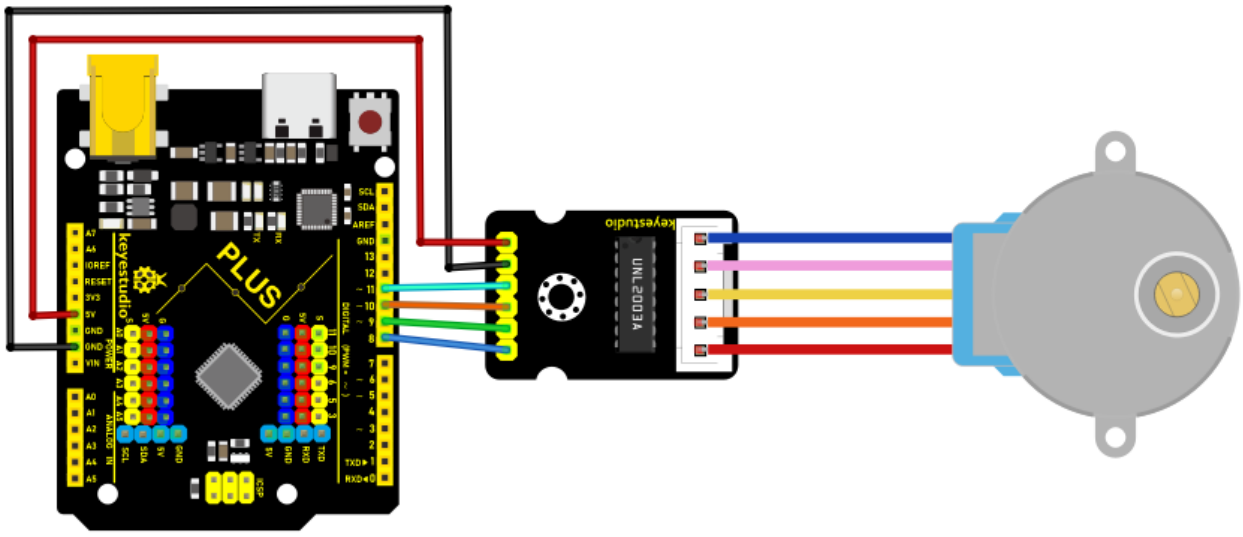
ULN2003Stepper Motor Drive Board: It is stepper motor driver.

The following schematic diagram shows how to use the ULN2003 stepper motor driver board interface to connect a unipolar stepper motor to the pins of the Plus control board, and shows how to use four TIP120 interfaces.



Schematic Diagram and Wiring Diagram





Code

```
/*

Keyestudio 2021 starter learning kit

Project 19

Stepper_motor

http//www.keyestudio.com

*/

// pins of the stepper motor

const int IN1_pin = 11;

const int IN2_pin = 10;

const int IN3_pin = 9;

const int IN4_pin = 8;

int val;

void setup() {

Serial.begin(9600);

// the setting of pins of stepper motor in Arduino

pinMode(IN1_pin,OUTPUT);

pinMode(IN2_pin,OUTPUT);
```

(continues on next page)

(continued from previous page)

```
pinMode(IN3_pin,OUTPUT);

pinMode(IN4_pin,OUTPUT);

}

void loop() {

int a = 1024;

int b = 1024;

val=Serial.read();

if(val=='A')

{

while(a--)

{

digitalWrite(IN1_pin, HIGH);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN4_pin, LOW);

delay(10);

digitalWrite(IN1_pin, LOW);

digitalWrite(IN2_pin, HIGH);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN4_pin, LOW);

delay(10);

digitalWrite(IN1_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN3_pin, HIGH);

digitalWrite(IN4_pin, LOW);

delay(10);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(IN1_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN4_pin, HIGH);

delay(10);

}

}

if(val=='C')

{

while(b--)

{

digitalWrite(IN4_pin, HIGH);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, LOW);

delay(10);

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, HIGH);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, LOW);

delay(10);

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, HIGH);

digitalWrite(IN1_pin, LOW);

delay(10);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, HIGH);

delay(10);

}

}

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, LOW);

}
```

Result

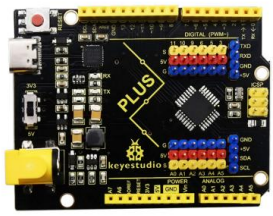
Upload the project code to the PLUS Mainboard, wire up and power on first. Then open the serial monitor, set the baud rate to 9600. We enter "A" in the serial monitor and click "send", the stepper motor rotates counterclockwise rotation. Enter "C" in the serial monitor and click "send", the stepper motor rotates clockwise.

5.20 Project 20: Relay

Introduction

In daily life, we generally use AC to drive electrical equipment, and sometimes we use switches to control electrical appliances. If the switch is directly connected to the AC circuit, once electricity leakage occurs, people are in danger. From a safety point of view, we specially designed this relay module with NO (normally open) and NC (normally closed) terminals. In this lesson we will learn a special and easy-to-use switch, which is the relay module.

Components Required

			
Keyestudio Plus MainBoardx1	Relay Modulex1	F-F Dupont Wires	USB Cablex1

Component Knowledge

Relay: It is an "automatic switch" that uses a small current to control the operation of a large current.

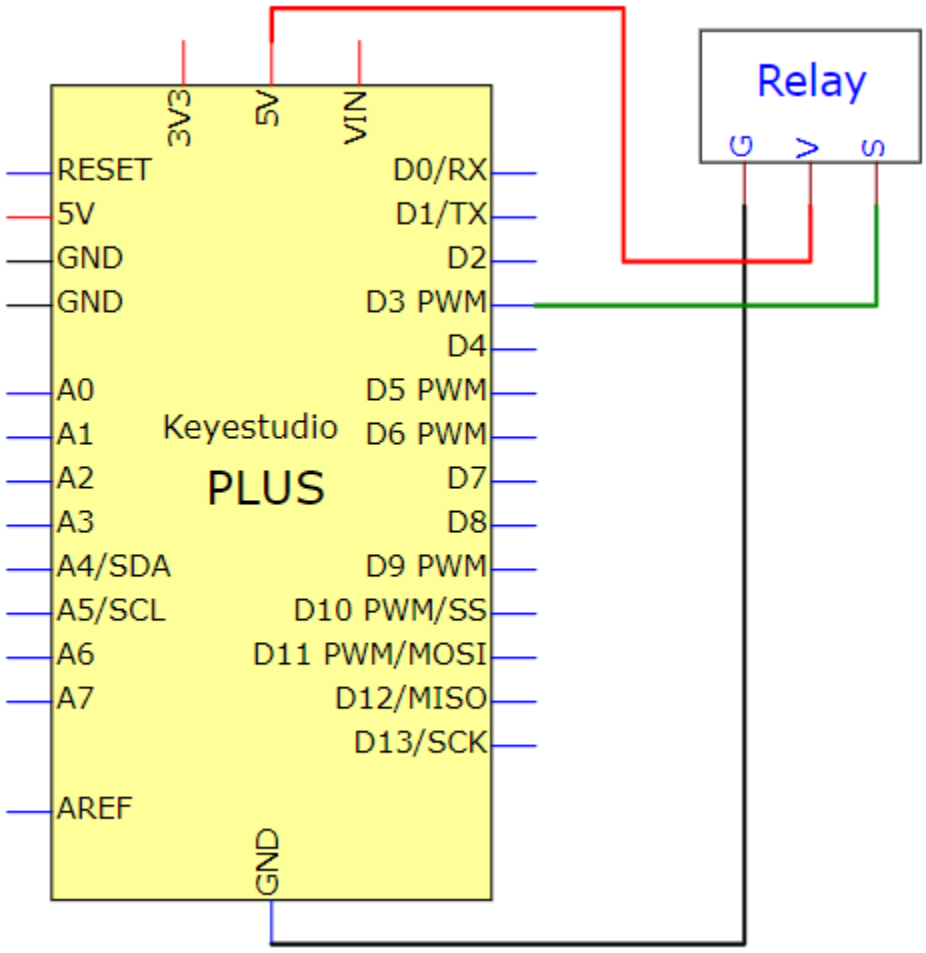
Input voltage5V

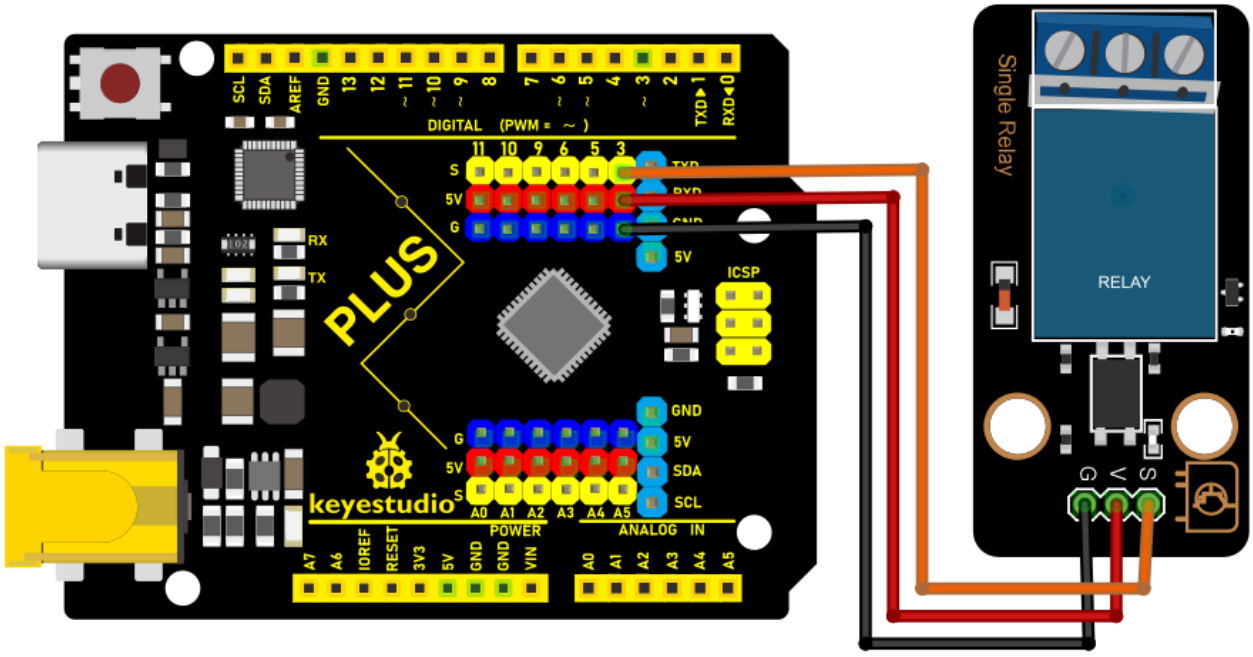
Rated load5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

The rated load means that a 5V Arduino can be used to control a device with a 24V DC voltage or a 250V AC voltage.

Schematic diagram of the Relay:

Schematic Diagram and Wiring Diagram


Relay
G
V
S
3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS


Single Relay
RELAY
G
V
S
PLUS
keyestudio
DIGITAL (PWM = ~)
POWER
ANALOG IN
ICSP
RX
TX
GND
5V
SDA
SCL

Code

```
/*

Keyestudio 2021 Starter Kit

Project 20

Relay

http//www.keyestudio.com

*/

int Relay = 3; // defines digital 3

void setup()

{

pinMode(Relay, OUTPUT); // sets "Relay" to "output"

}

void loop()

{

digitalWrite(Relay, HIGH); // turns on the relay

delay(2000); //delays 2 seconds

digitalWrite(Relay, LOW); // turns off the relay

delay(2000); // delays 2 seconds

}
```

Result

Upload the code to the mainboard successfully, wire up and power on, the relay will be turned on (ON end is connected) for 2 seconds, and stop (NC end is connected) for 2 seconds, circularly

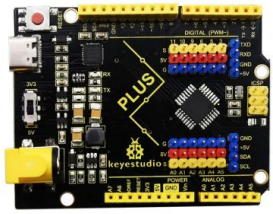
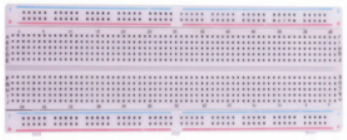
5.21 Project 21: Dimming Light

Introduction

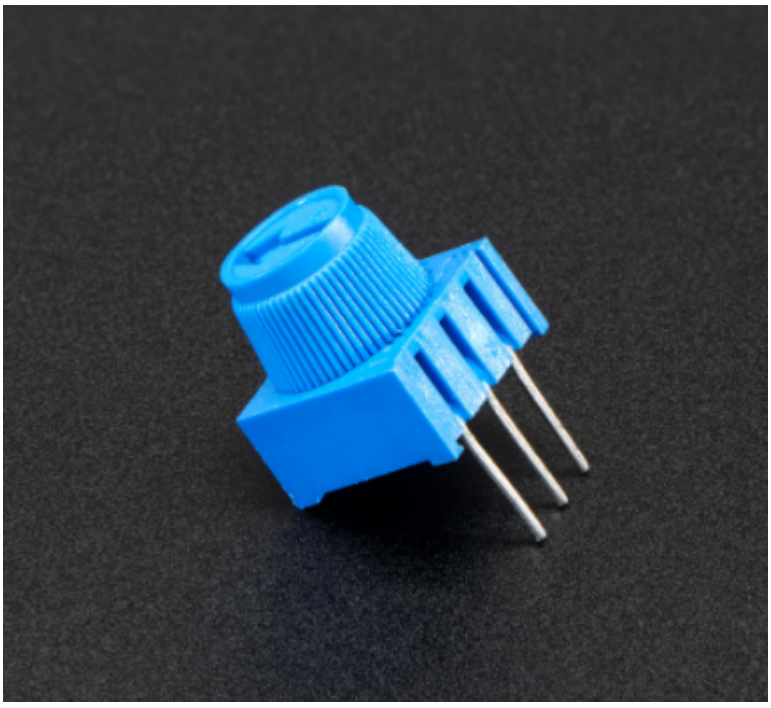
A potentiometer is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. It works by varying the position of a sliding contact across a uniform resistance. In a potentiometer, the entire input voltage is applied across the whole length of the resistor, and the output voltage is the voltage drop between the fixed and sliding contact.

In this project, we are going to learn how to use Arduino to read the values of the potentiometer, and make a dimming lamp.

Components Required

			
Keyestudio Plus Mainboardx1	Potentiometerx1	Red LEDx1	200 Resistorx1
			
Breadboardx1	USB Cablex1	Jumper Wires	

Component Knowledge

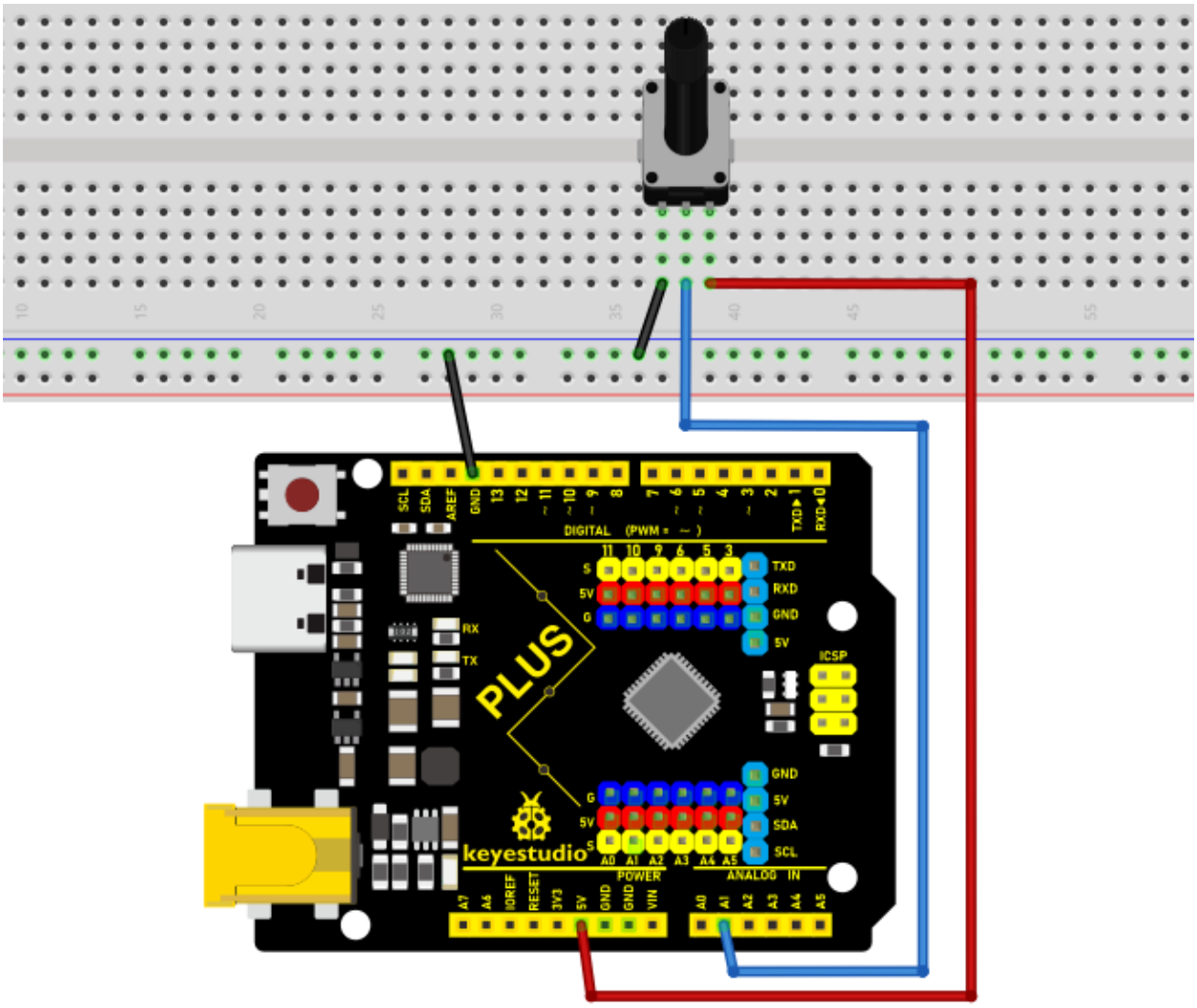


Adjustable potentiometer: It is a kind of resistor and an analog electronic component, which has two states of 0 and 1(high level and low level). The analog quantity is different, its data state presents a linear state such as 0 to 1023.

Read the Potentiometer Value

We connect the adjustable potentiometer to the analog pin of Arduino to read its value. Please refer to the following

wiring diagram for wiring.




```
/*

Keyestudio 2021 Starter Kit

Project 21.1

Read_the_adjustable_potentiometer_analog_value

http//www.keyestudio.com

*/

int potpin=A1;// initializes analog PIN A1 of the potentiometer

int val=0;// defines "val" with an initial value of 0

void setup()
```

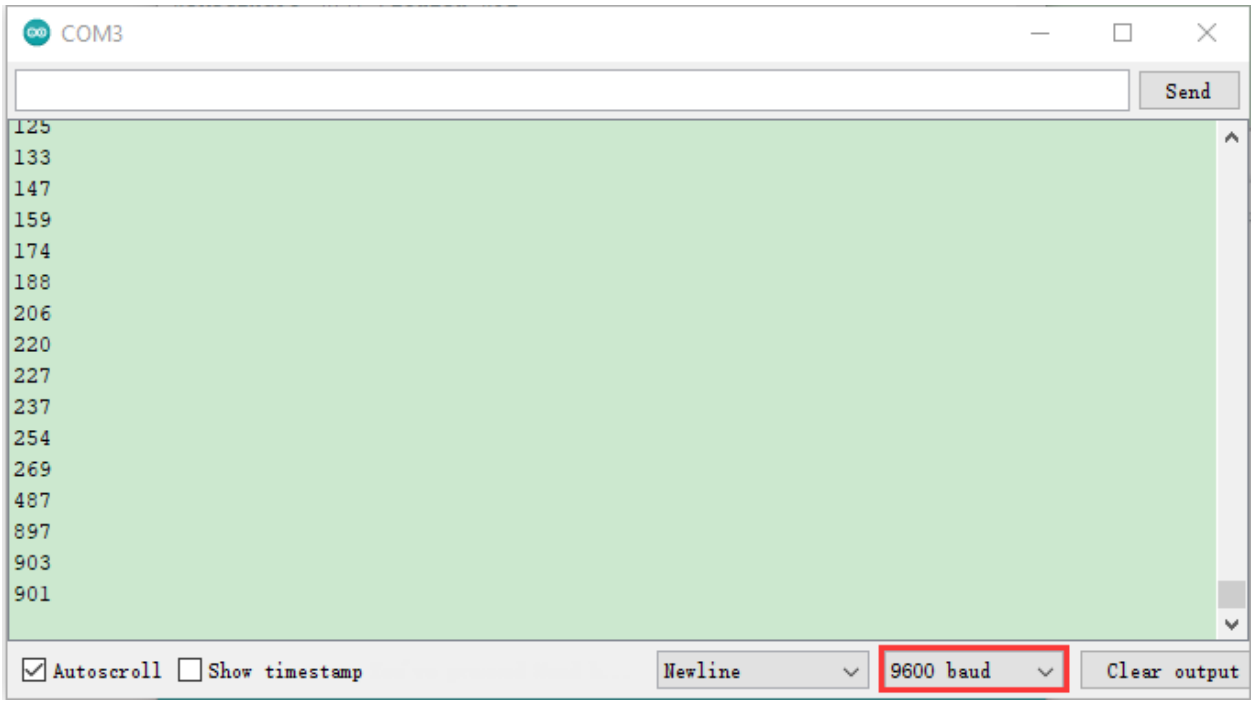
(continues on next page)

(continued from previous page)

```
{

Serial.begin(9600);// sets baudrate to 9600

}

void loop()

{

val=analogRead(potpin);// reads the analog value of analog PIN A1 and assigns it
to "val"

Serial.println(val);// displays the value of "val"

}
```

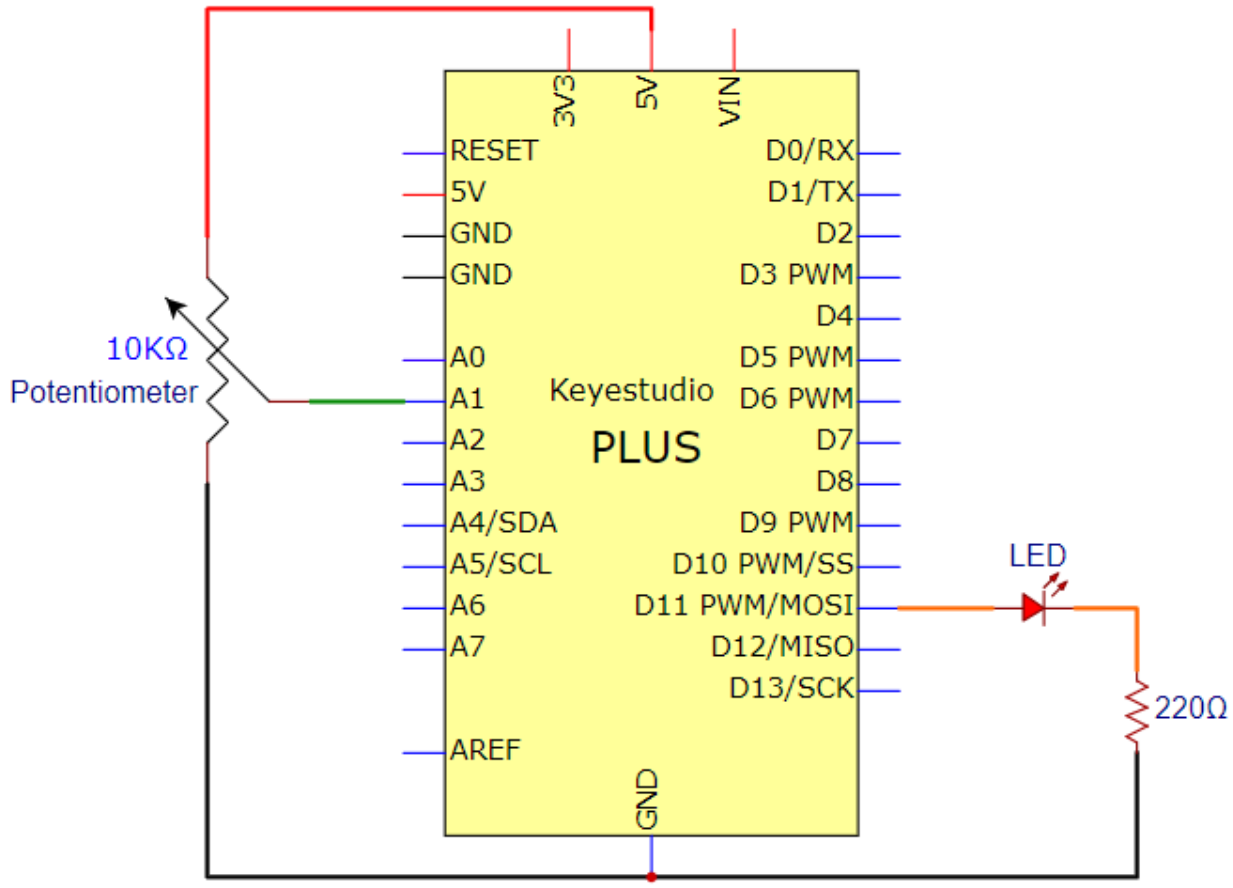
Upload the code to the Plus mainboard, connect the wires and power on first. Open the serial monitor, set the baud rate to 9600. When you rotate the potentiometer knob, you can see the displayed value change. After calculation, you can get the corresponding value you need.

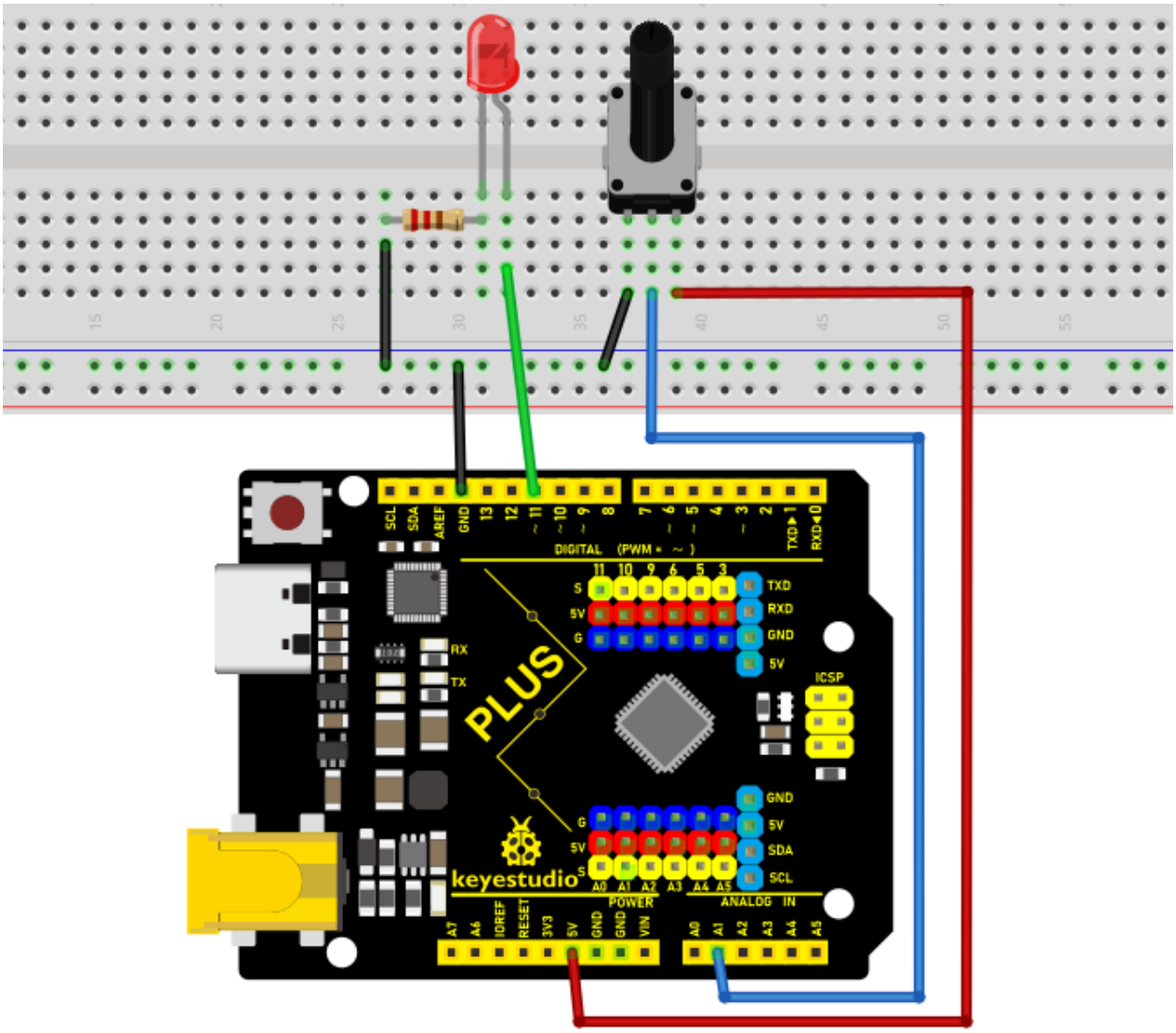
Below figure shows the analog value it reads.



Circuit Diagram and Wiring Diagram

In the last step, we read the value of the potentiometer, and now we need to convert the value of the potentiometer into the brightness of the LED to make a lamp that can adjust the brightness. The wiring diagram is as follows.


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
10KΩ
Potentiometer
LED
220Ω



Code

```
/*

Keyestudio 2021 Starter Kit

Project 21.2

Dimming_light

http//www.keyestudio.com

*/

int potpin=A1;// initializes analog PIN A1 of the adjustable potentiometer

int ledpin=11;// initializes the digital PIN 11
```

(continues on next page)

(continued from previous page)

```
int val=0;// defines "val" with an initial value of 0

void setup()

{

pinMode(ledpin,OUTPUT);// sets digital pin to "output"

Serial.begin(9600);// sets baudrate to 9600

}

void loop()

{

val=analogRead(potpin);// reads the analog value of analog PIN A1 and assigns it
to "val"

analogWrite(ledpin,val/4);

Serial.println(val);// displays the value of "val"

}
```

Result

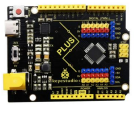
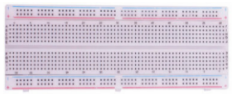
Upload the code to the Mainboard, connect the wires and power on first. Then open the serial monitor, set the baud rate to 9600, and the monitor will display the value of potentiometer. When we turn the knob of the potentiometer, the brightness of the LED will change.

5.22 Project 22: Flame Alarm

Introduction

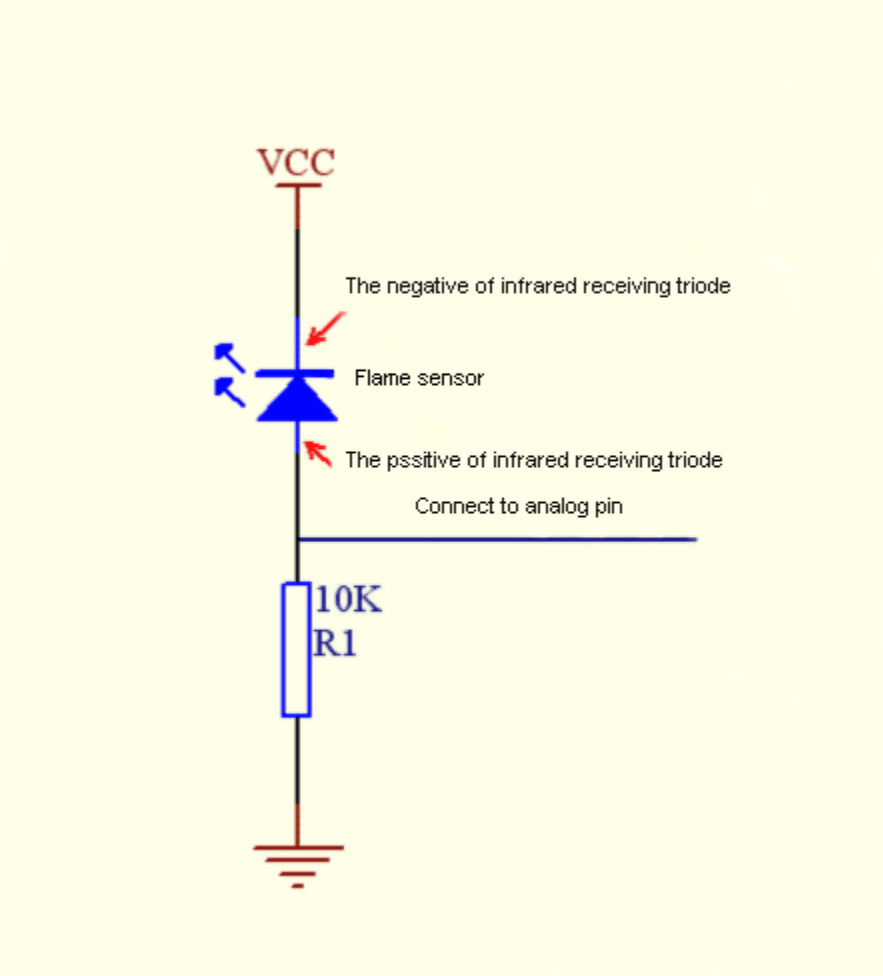
In this project, we will use the Plus mainboard, a flame sensor and a buzzer to make fire alarm devices.

Components Required

						
Keyestudio Plus Mainboardx1	Flame Sensorx1	Active Buzzerx1	Breadboardx1	Jumper Wires	USB Cablex1	10K Resistorx1

Component Knowledge

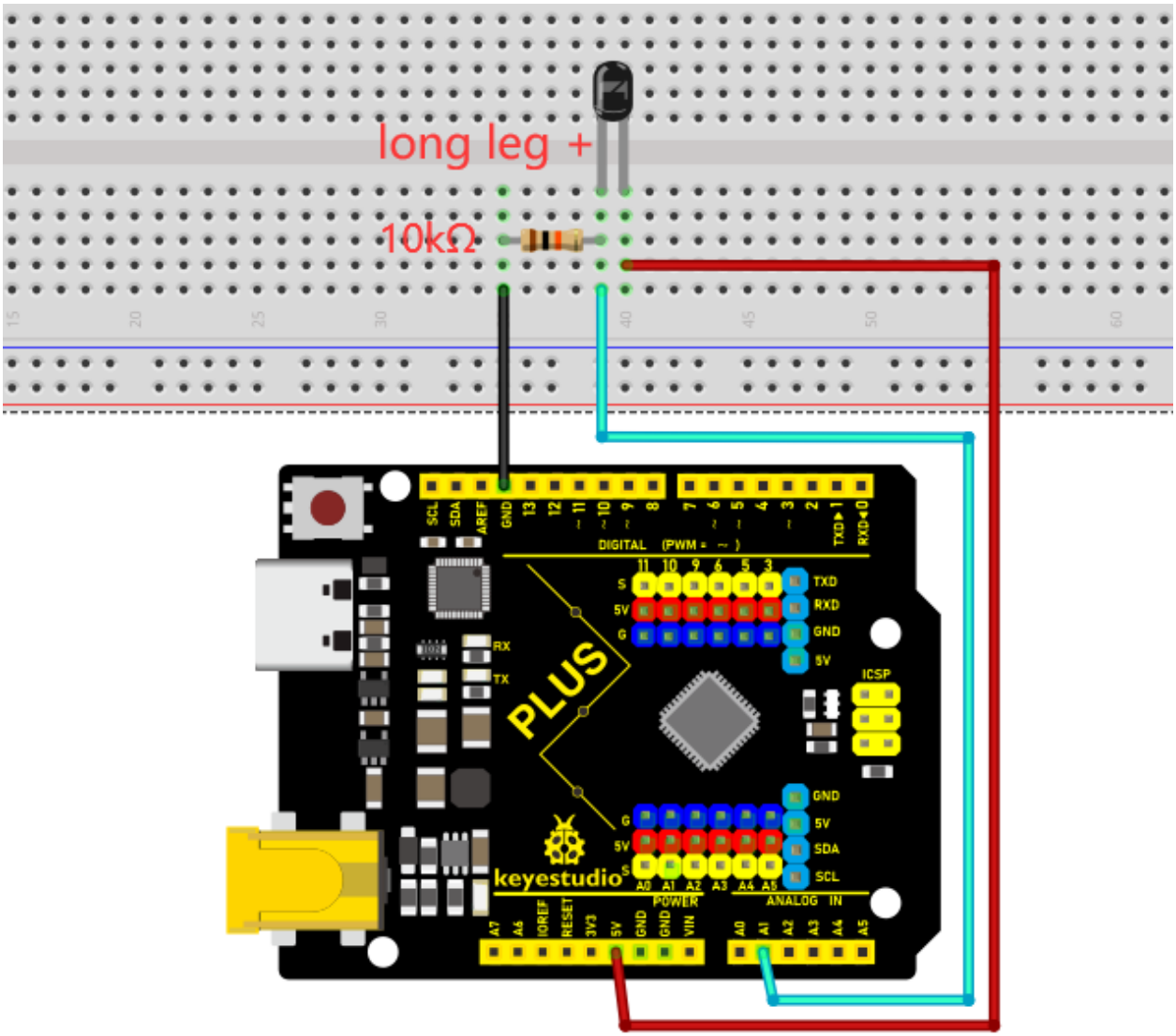
Flame Sensor The flame emits a certain degree of IR light, which is invisible to the human eye, but our flame sensor can detect it and alert the microcontroller. If the Arduino has detected a fire, it has a specially designed infrared receiver to detect the flame, and then convert the flame brightness into a fluctuating level signal. The short pin of the receiving triode is negative pole and the other long pin is positive pole. We should connect the short pin (negative pole) to 5V and the long pin (positive pole) to the analog pin, a resistor and GND. As shown in the figure below.



Note: Since vulnerable to radio frequency radiation and temperature changes, the flame sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

Read the Simulation Value

We start with a simple code to read the value of the flame sensor and print it on the serial monitor. For wiring, please refer to the following wiring diagram.



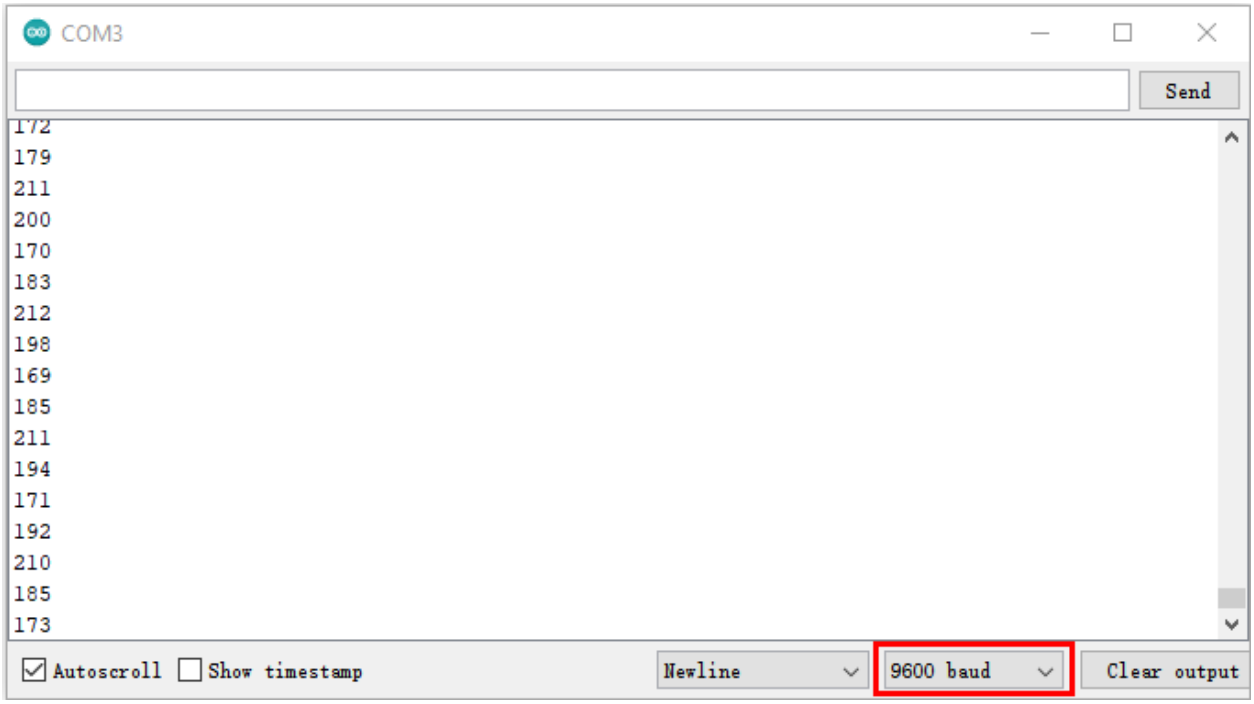

```
/*

Keyestudio 2021 Starter Kit

Project 22.1

Read_the_flame_sensor_analog_value

http//www.keyestudio.com

*/

int flamepin=A1;// initializes analog PIN A1

int val=0;// defines "val" with an initial value of 0

void setup()

```

(continues on next page)

(continued from previous page)

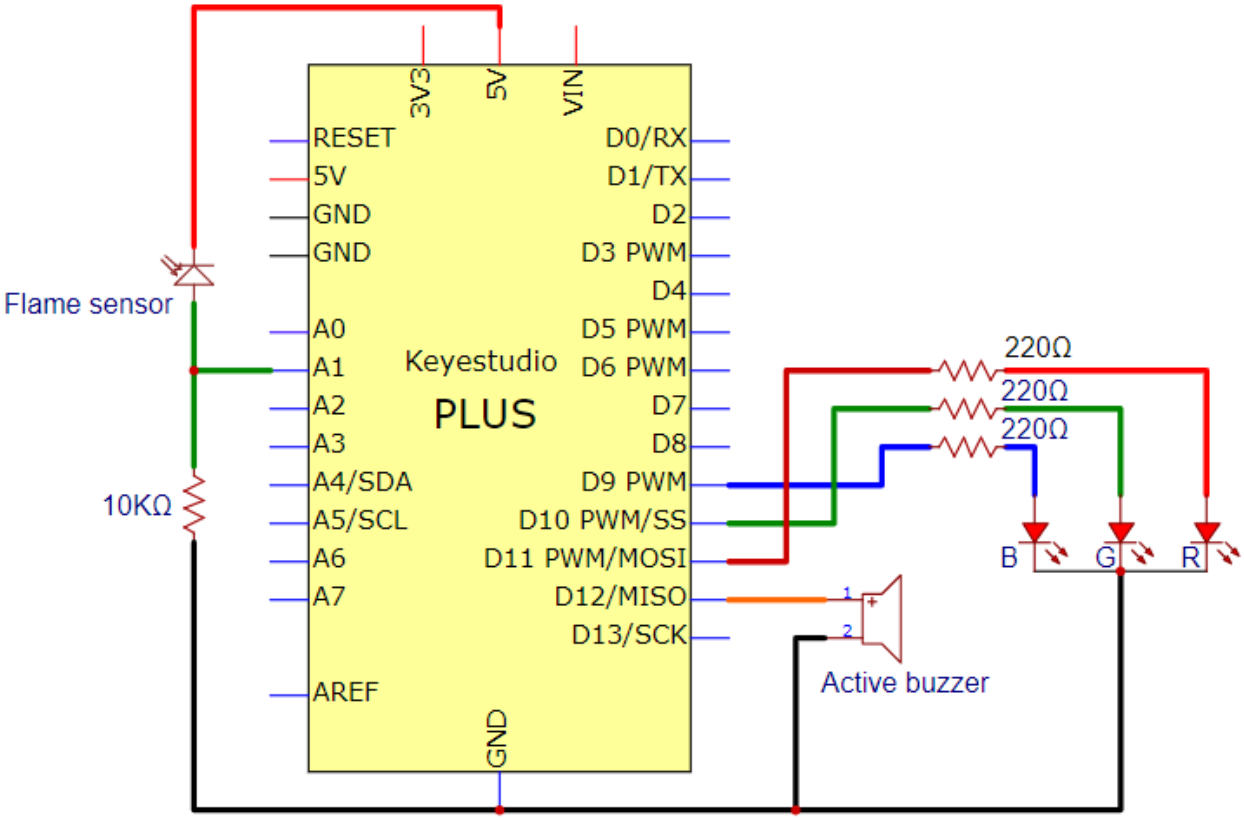
```
{

Serial.begin(9600);// sets baudrate to 9600

}

void loop()

{

val=analogRead(flamepin);// reads the analog value of analog PIN A1 and assigns
it to "val"

Serial.println(val);// displays the value of "val"

}
```

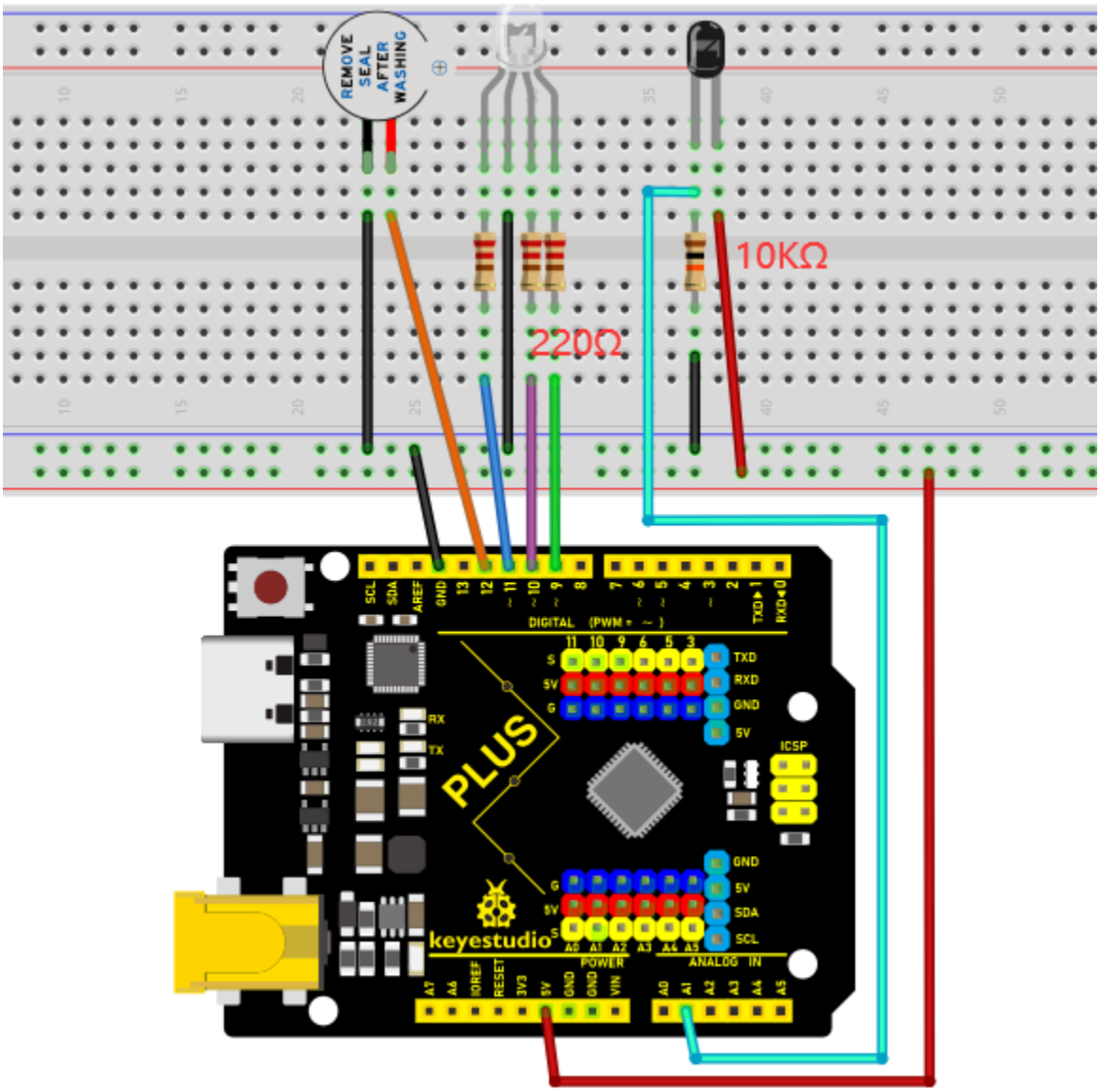
Upload the code to the Plus Mainboard, connect the wires and power on first. Turn on the serial monitor and set the baud rate to 9600, and approach the flame sensor with a lighter flame to see its analog value.



Circuit Diagram and Wiring Diagram

Next, we will use flame sensor and buzzer, RGB LED to make an interesting project, that is flame alarm. When flame is detected, RGB LED is red and buzzer alarms.


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
Flame sensor
10KΩ
220Ω
220Ω
220Ω
B
G
R
Active buzzer



Code

```
/*

Keyestudio 2021 Starter Kit

Project 22.2

Fire_alarm

http//www.keyestudio.com

*/

const int red = 11;
```

(continues on next page)

(continued from previous page)

```
const int green = 10;

const int blue= 9;

const int buzzer = 12;

const int flamepin = A1;

const int thereshold = 30;

void setup() {

// puts the setup code here and runs it once

Serial.begin(9600);

pinMode(red, OUTPUT);

pinMode(green, OUTPUT);

pinMode(blue, OUTPUT);

pinMode(buzzer, OUTPUT);

pinMode(flamepin, INPUT);

}

void setColor(int redValue, int greenValue, int blueValue)

{

analogWrite(red, redValue);

analogWrite(blue, blueValue);

analogWrite(green, greenValue);

}

void loop() {

// puts the main code here and repeats

int flamesenseval = analogRead(flamepin);

Serial.println(flamesenseval);

if (flamesenseval \>= thereshold) {

setColor(255, 0, 0); //red
```

(continues on next page)

(continued from previous page)

```
tone(buzzer, 1000);

delay(10);

}

else

{

setColor(0, 255, 0); // green

noTone(buzzer);

}

}
```

Result

Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor and set the baud rate to 9600, the monitor will display the value of the flame sensor. We use light the fire and keep it close to the flame sensor, the RGB LED will become red and the buzzer will alarm. Otherwise, the RGB LED will turn green and the buzzer doesn't emit sounds.

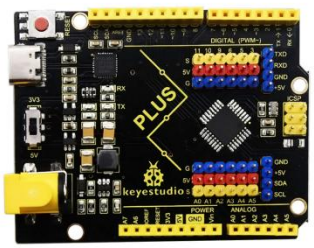
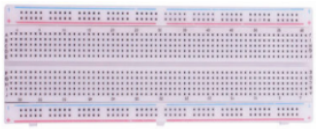
5.23 Project 23: Optic Control Lamp

Introduction

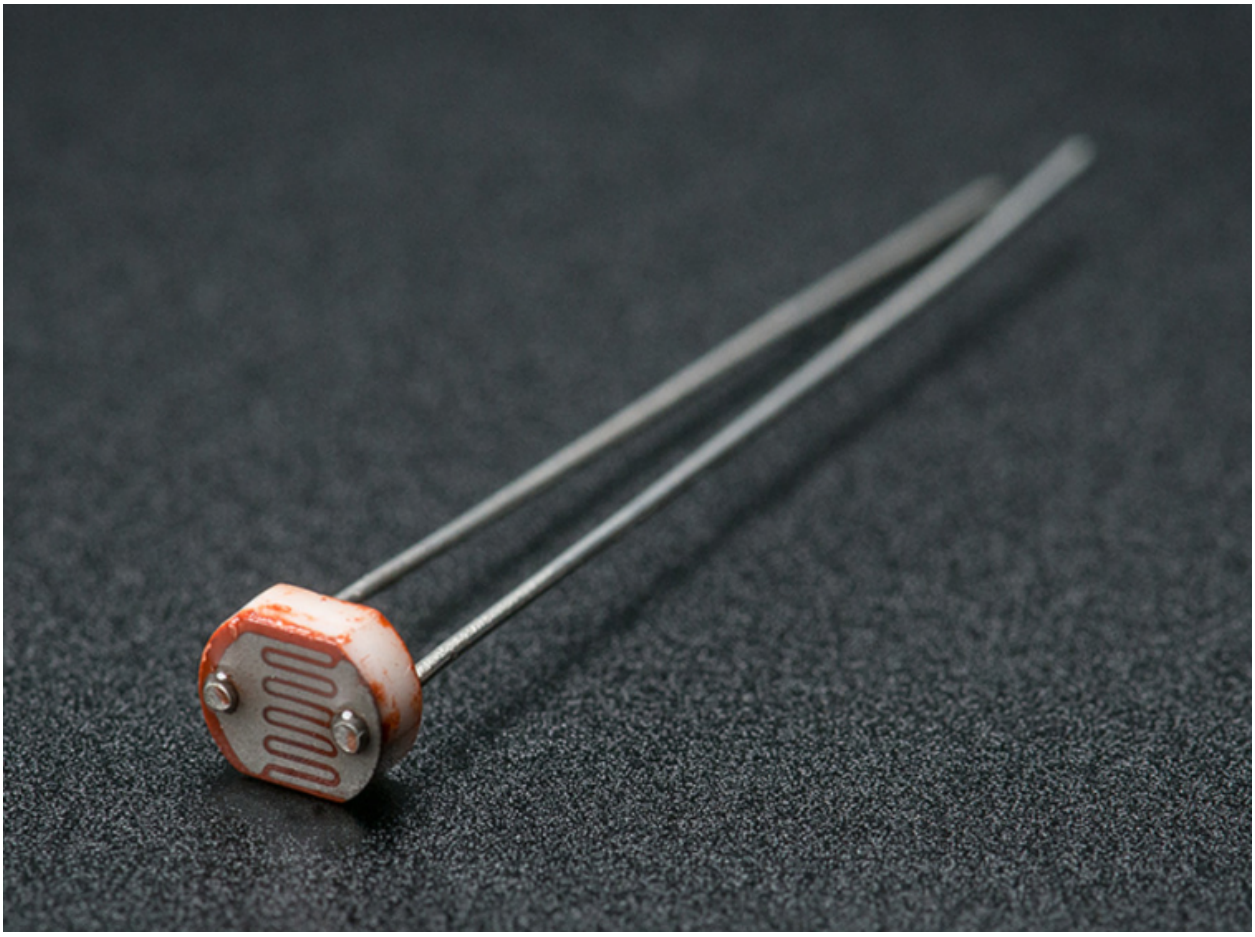
Sensors or components are ubiquitous in our daily life. For example, some public street lights turn on automatically at night and turn off automatically during the day. These make use of a photosensitive element that senses the intensity of external ambient light. When the outdoor brightness decreases at night, the street lights will automatically turn on. In the daytime, the street lights will automatically turn off. The principle of this is very simple.

In this lesson we will implement the function of this street light.

Components Required

			
Keyestudio Plus Mainboardx1	Photoresistorx1	Red LEDx1	220 Resistorx1
			
10K Resistorx1	Breadboardx1	Jumper Wires	USB Cablex1

Component Knowledge



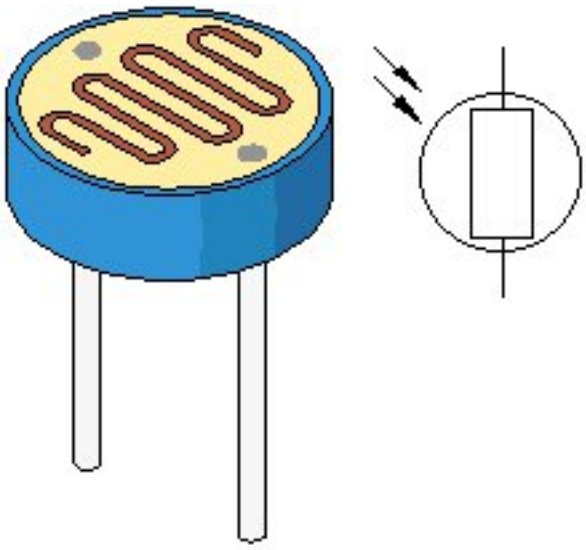
Photoresistor:

Photosensor is a kind of resistor made by using the photoelectric effect of semiconductor, the resistance value changes with the intensity of the incident light, also known as photoelectric detector. When the surrounding light becomes stronger, the resistance becomes smaller and the analog signal becomes larger. Conversely, when the light becomes weaker, the resistance increases and the analog signal becomes smaller.

The commonly used material for making photosensor is cadmium sulfide, in addition to selenium, aluminum sulfide, lead sulfide and bismuth sulfide and so on. These materials have the characteristic that their resistance decreases rapidly under the irradiation of light of a specific wavelength. This is because the carriers generated by the light are involved in the conduction and drift under the action of the applied electric field. The electrons rush to the positive electrode of the power supply, and the holes rush to the negative electrode of the power supply, so that the resistance of the photosensor drops rapidly.

Photoresistor is commonly applied in the measurement of light, light control and photovoltaic conversion (convert the change of light into the change of electricity).

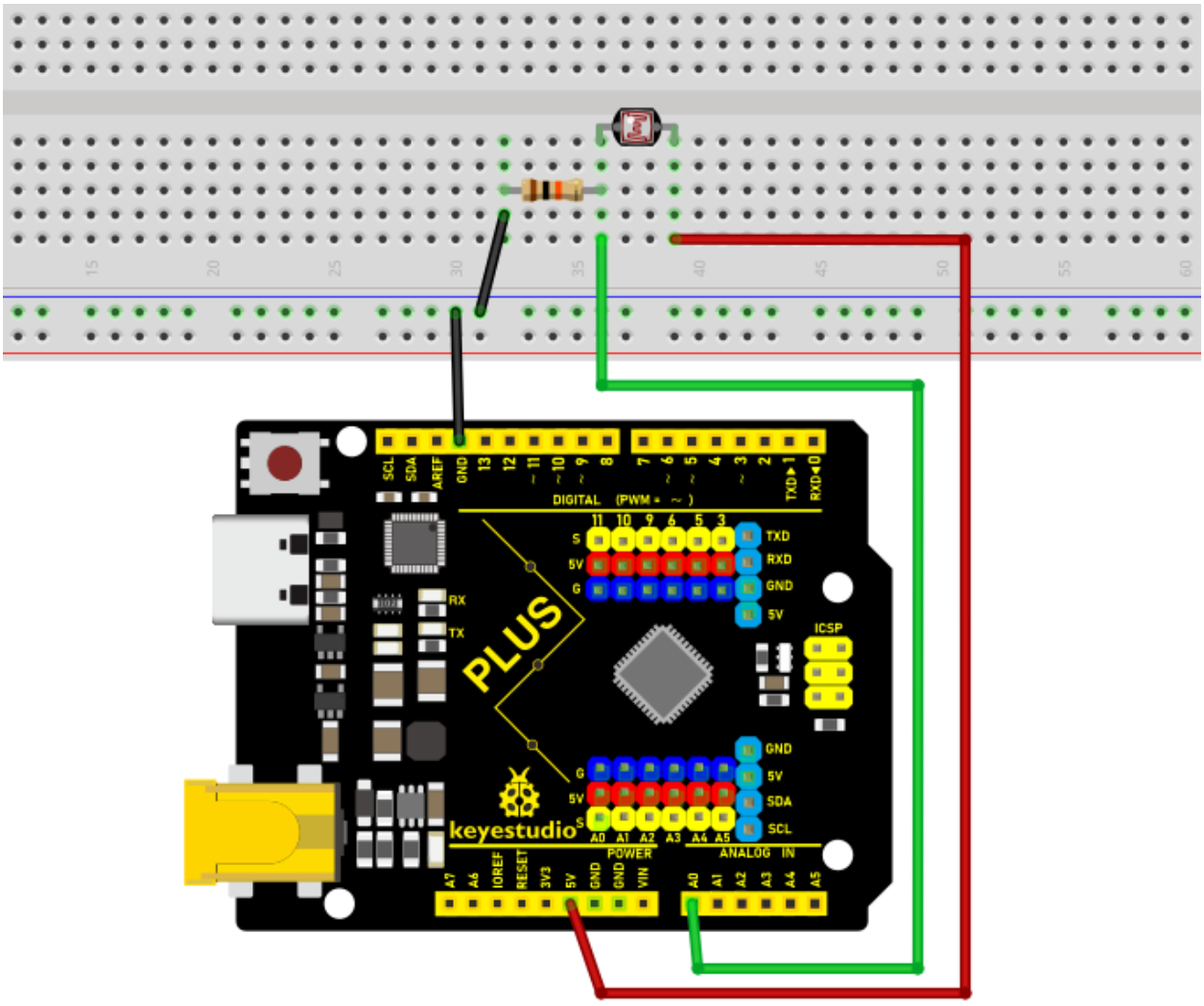
Photoresistor is also being widely applied to various light control circuit, such as light control and adjustment, optical switches, etc.



We will start with a relatively simple experiment regarding to photovaristor application.

Read the Analog Value

We first use a simple code to read the value of the photoresistor, print it in the serial monitor. For wiring, please refer to the following wiring diagram.



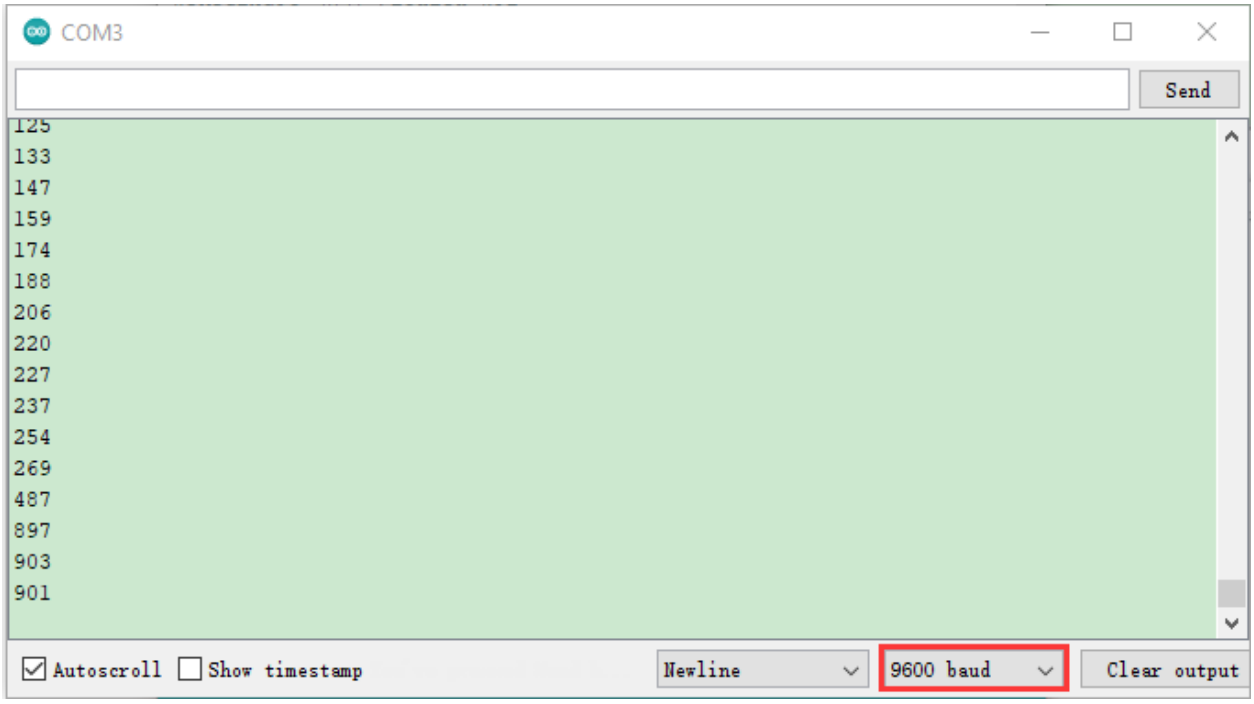

```
/*

Keyestudio 2021 Starter Kit

Project 23.1

Read_the_photosensitive_resistance_analog_value

http//www.keyestudio.com

*/

int photocellpin=A0;// initializes the analog PIN A0 connected to the
photoresistor

int val=0;// initializes the variable "val" with a value of 0

void setup()
```

(continues on next page)

(continued from previous page)

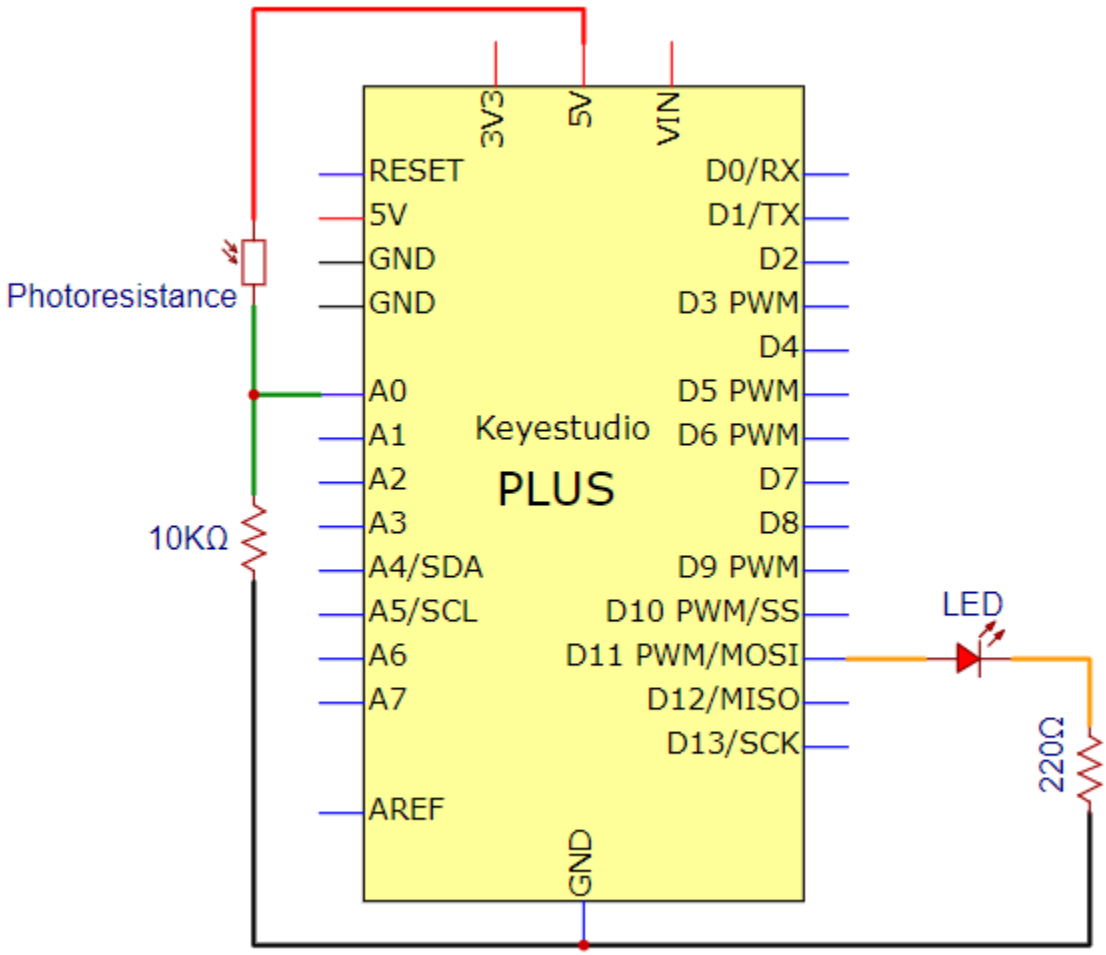
```
{

Serial.begin(9600);// sets baudrate to 9600

}

void loop()

{

val=analogRead(photocellpin);// reads the value of the sensor and assigns its
value to "val"

Serial.println(val);// displays the value of "val"

delay(200);// waits 0.2 second

}
```

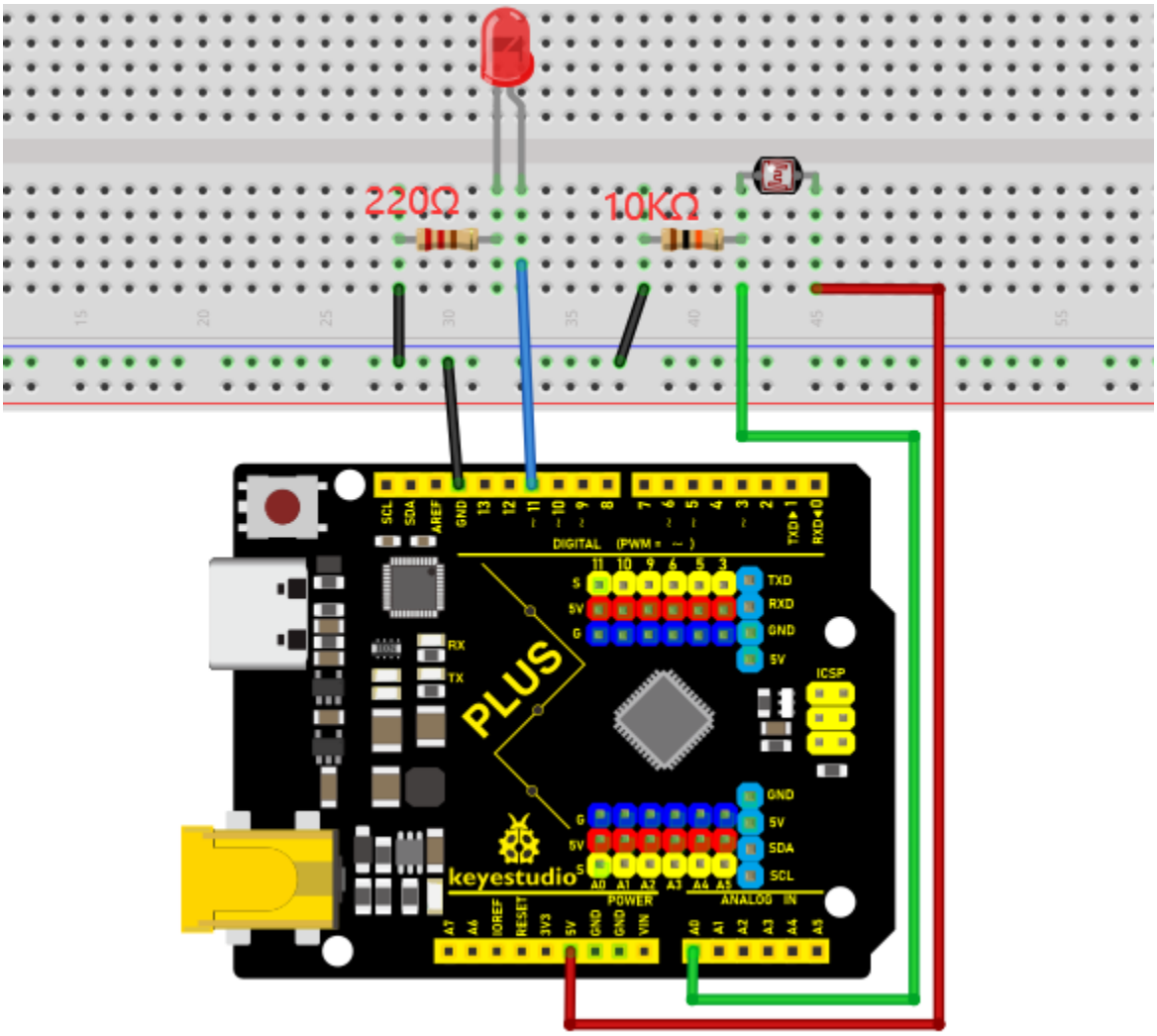
Upload the code to the Plus Mainboard, wire up and power up, open the serial monitor and set the baud rate to **9600**. Then you can read the analog value of photoresistor. When the light intensity around the sensor gets dim, the analog value displayed on the serial monitor will gradually reduce. On the contrary, the analog value will gradually increase.



Circuit Diagram and Wiring Diagram

Next, we make an optical control lamp.


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
Photoresistance
10KΩ
LED
220Ω



Code

```
/*

Keyestudio 2021 Starter Kit

Project 23.2

Optical Control Lamp

http//www.keyestudio.com

*/

int photocellpin=A0;// initializes the analog PIN A0 connected to the
photoresistor

int ledpin=11;// initializes digital PIN 11

int val=0;// initializes the variable "val" with a value of 0
```

(continues on next page)

(continued from previous page)

```
void setup()

{

pinMode(ledpin,OUTPUT);// sets digital PIN 11 to "output"

Serial.begin(9600);// sets baudrate to 9600

}

void loop()

{

val=analogRead(photocellpin);//reads the analog value of the sensor and assigns
its value to "val"

Serial.println(val);//displays the value of "val"

analogWrite(ledpin,val/4);//sets brightness (Max:255)

delay(10);// waits 0.01 second

}
```

Result

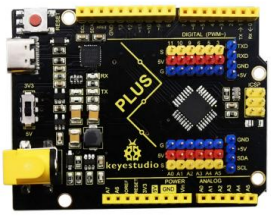
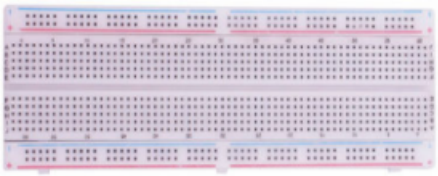
Upload the code to the Plus Mainboard, wire up and power up, open the serial monitor and set the baud rate to **9600**.

Then you can read the analog value of photoresistor. When the light intensity around the sensor gets dim, the analog value displayed on the serial monitor will gradually reduce. On the contrary, the analog value will gradually increase.

5.24 Project 24: Ultrasonic Ranger

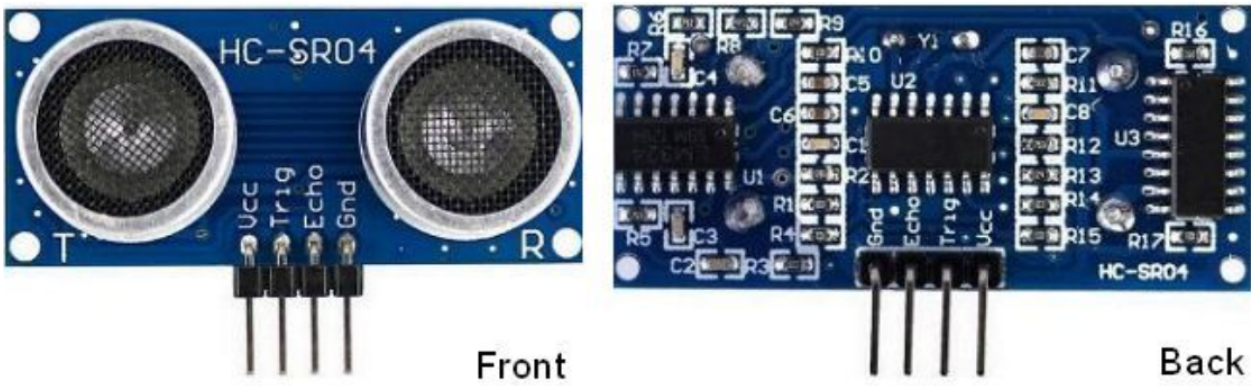
Introduction The HC-SR04 ultrasonic sensor is a very affordable distance sensor, mainly used for obstacle avoidance in various robotic projects. It is also used for water level sensing and even as a parking sensor. We treat the ultrasonic sensors as bat's eyes. In the dark, bats can still identify objects in front of them and directions through ultrasound.

Components Required

			
Keyestudio Plus Mainboardx1	Ultrasonic Sensorx1	220 Resistorx4	Red LEDx4
			
M-F Dupont Wires	USB Cablex1	Breadboardx1	Jumper Wires

Component Knowledge

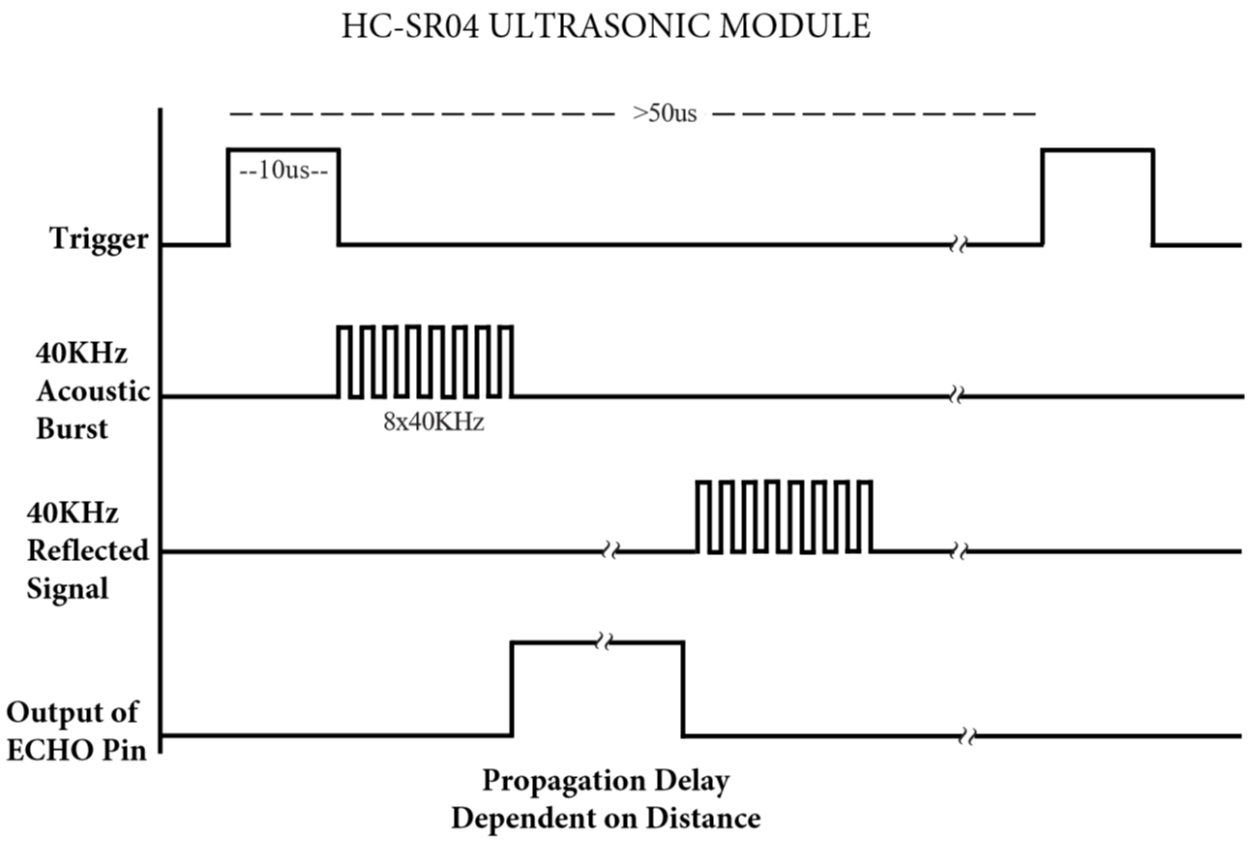
HC-SR04 ultrasonic sensor: Like bats, sonar is used to determine the distance to an object. It provides accurate non-contact range detection, high-precision and stable readings. Its operation is not affected by sunlight or black materials, just like a precision camera (acoustically softer materials like cloth are difficult to detect). It has an ultrasonic transmitter and receiver.



In front of the ultrasonic sensor are two metal cylinders, these are the converters. The converters convert the mechanical energy into an electrical signal. In the ultrasonic sensor, there are transmitting converters and receiving converters. The transmitting converter converts the electric signal into an ultrasonic pulse, and the receiving converter converts the reflected ultrasonic pulse back to an electric signal. If you look at the back of the ultrasonic sensor, you will see an IC behind the transmitting converter, which controls the transmitting converter. There is also an IC behind the receiving converter, which is a quad operational amplifier that amplifies the signal generated by the receiving converter into a signal large enough to be transmitted to the Arduino.

Sequence diagrams:

The figure shows the sequence diagram of the HC-SR04. To start the measurement, the Trig of SR04 must receive at least 10us high pulse (5V), which will activate the sensor to emit 8 cycles of 40kHz ultrasonic pulses, and wait for the reflected ultrasonic pulses. When the sensor detects ultrasound from the receiver, it sets the Echo pin to high (5V) and delays it by one cycle (width), proportional to the distance. To get the distance, measure the width of the Echo pin.



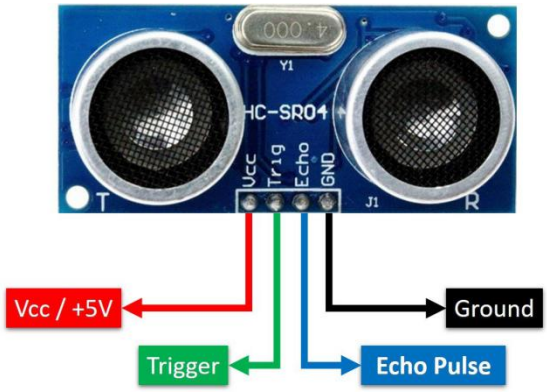
Time = Echo pulse width, its unit is “us” (microseconds)

Distance in centimeters = time / 58

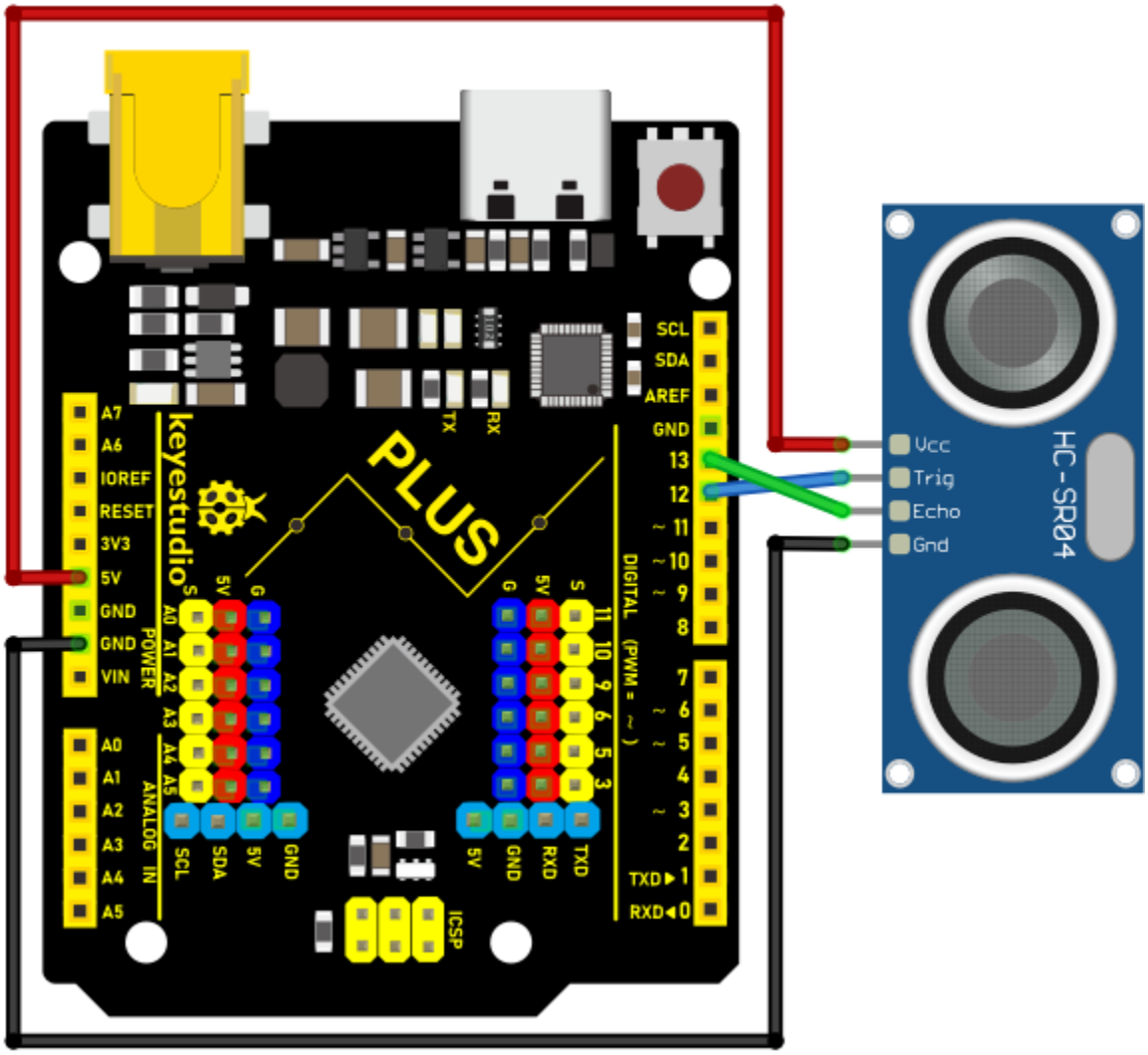
Distance in inches = time / 148

Read the Distance Value

We will start with a simple ultrasonic distance measurement and output the measured distance on the serial monitor.



The HC-SR04 ultrasonic sensor has four pins, they are Vcc, Trig, Echo and GND. The Vcc pin provides the power source for generating ultrasonic pulses and is connected to Vcc (+5V). The GND pin is grounded. The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse. The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse to the Plus control board. Wiring as shown below.



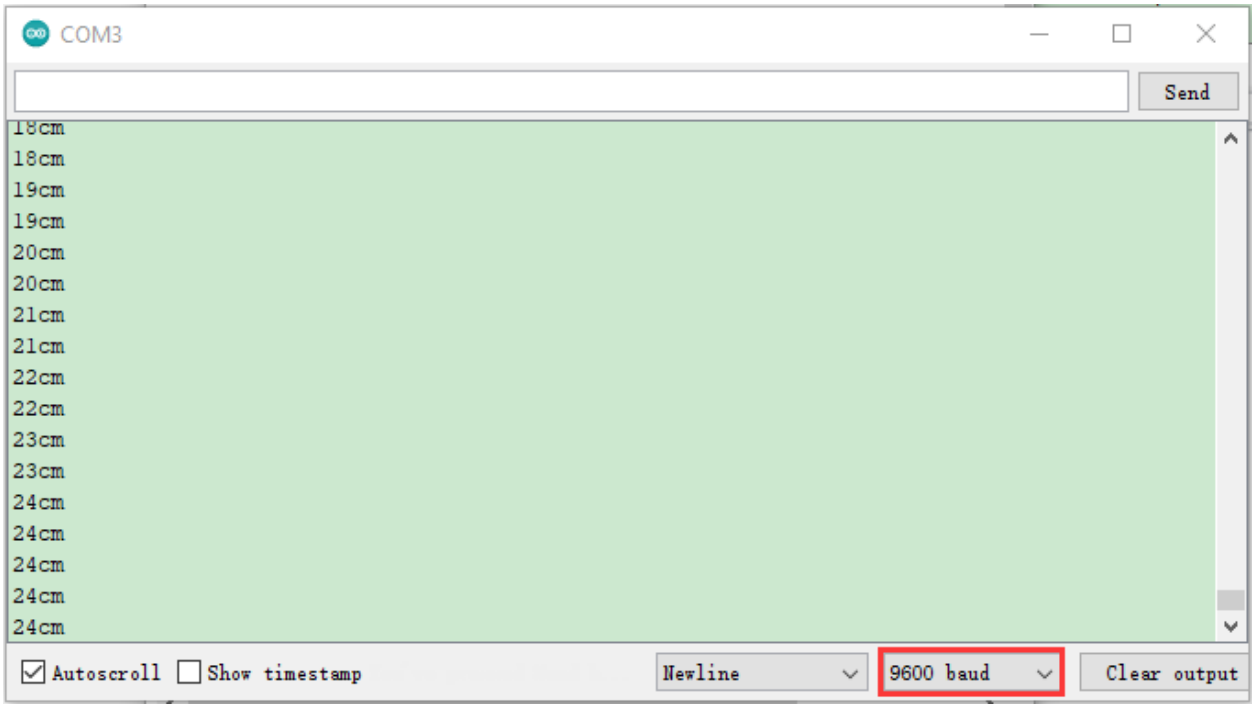

```
/*

Keyestudio 2021 starter learning kit

Project 24.1

Read_the_ultrasonic_distance

http//www.keyestudio.com

*/

const int trig = 12;
```

(continues on next page)

(continued from previous page)

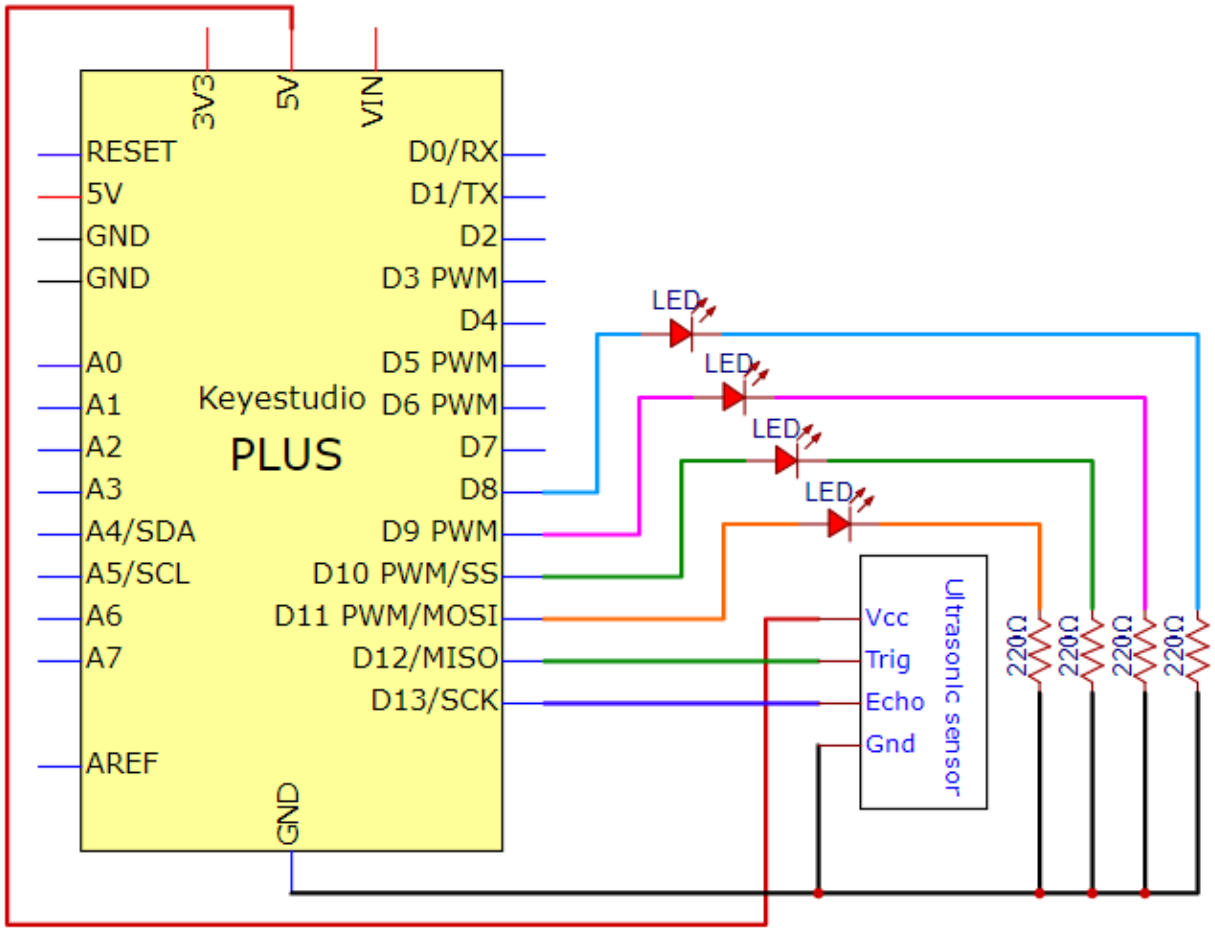
```
const int echo = 13;

int duration = 0;

int distance = 0;

void setup()

{

pinMode(trig , OUTPUT);

pinMode(echo , INPUT);

Serial.begin(9600);

}

void loop()

{

digitalWrite(trig , HIGH);

delayMicroseconds(1000);

digitalWrite(trig , LOW);

duration = pulseIn(echo , HIGH);

distance = (duration/2) / 28.5 ;

Serial.print(distance);

Serial.println("cm");

}
```

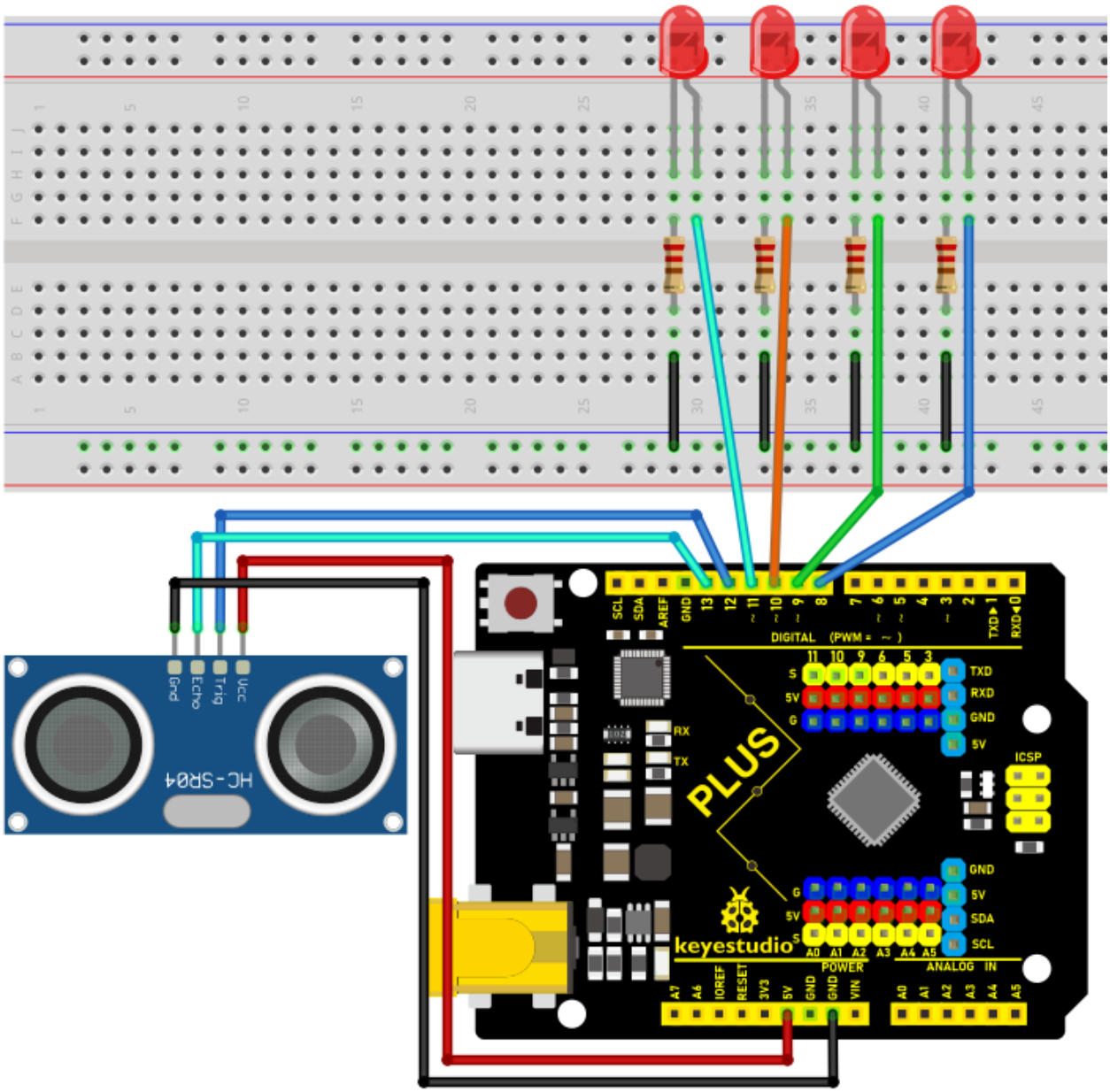
Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor, set the baud rate to 9600. When an object is placed in front of the ultrasonic sensor (near or far), it will detect the distance of the object and the value will be displayed on the monitor.



Circuit Diagram and Wiring Diagram

Next, we will make a simple ultrasonic ranger using an ultrasonic sensor and 4 LED lights. Connect the wires as shown below.


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
Keyestudio
PLUS
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
LED
LED
LED
LED
Vcc
Trig
Echo
Gnd
Ultrasonic sensor
220Ω
220Ω
220Ω
220Ω



Code

```
/*

Keyestudio 2021 starter learning kit

Project 24.2

Ultrasonic_Ranger

http//www.keyestudio.com

*/
```

(continues on next page)

(continued from previous page)

```
const int trig = 12;

const int echo = 13;

const int LED1 = 11;

const int LED2 = 10;

const int LED3 = 9;

const int LED4 = 8;

int duration = 0;

int distance = 0;

void setup()

{

pinMode(trig , OUTPUT);

pinMode(echo , INPUT);

pinMode(LED1 , OUTPUT);

pinMode(LED2 , OUTPUT);

pinMode(LED3 , OUTPUT);

pinMode(LED4 , OUTPUT);

Serial.begin(9600);

}

void loop()

{

digitalWrite(trig , HIGH);

delayMicroseconds(1000);

digitalWrite(trig , LOW);

duration = pulseIn(echo , HIGH);

distance = (duration/2) / 28.5;

Serial.println(distance);
```

(continues on next page)

(continued from previous page)

```
if ( distance \<= 7 )

{

digitalWrite(LED1, HIGH);

}

else

{

digitalWrite(LED1, LOW);

}

if ( distance \<= 14 )

{

digitalWrite(LED2, HIGH);

}

else

{

digitalWrite(LED2, LOW);

}

if ( distance \<= 21 )

{

digitalWrite(LED3, HIGH);

}

else

{

digitalWrite(LED3, LOW);

}

if ( distance \<= 28 )

{
```

(continues on next page)

(continued from previous page)

```
digitalWrite(LED4, HIGH);

}

else

{

digitalWrite(LED4, LOW);

}

}
```

Result

Upload the code to the PLUS Mainboard. After connecting the wires and powering on, the ultrasonic module can detect the distance of obstacles ahead. In addition, when we move our hands in front of the ultrasonic sensor, the corresponding LED will light up.

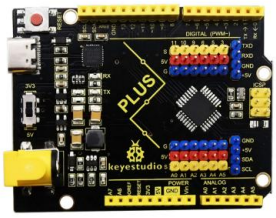
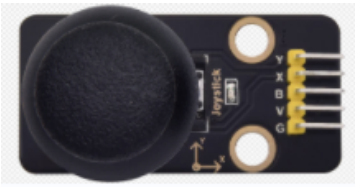
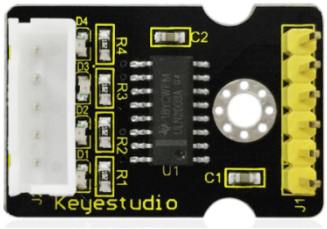
5.25 Project 25: Control Stepper Motor with Joystick

Introduction

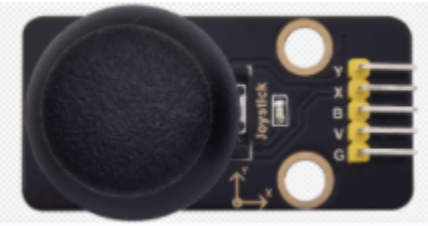
The joystick module is a component with two analog inputs and one digital input. It is widely used in game operation, robot control, drone control and other fields.

In this project, we use the Plus Mainboard and a joystick module to control the rotation of the stepper motor. You can have a deeper understanding of the principle and operation of the joystick module in practice.

Components Required

			
Keyestudio Plus Mainboardx1	Joystick Modulex1	Stepper Motorx1	USB Cablex1
			
ULN2003 Stepper Motor Drive Boardx1	M-F Dupont Wires	F-F Dupont Wires	

Component Knowledge



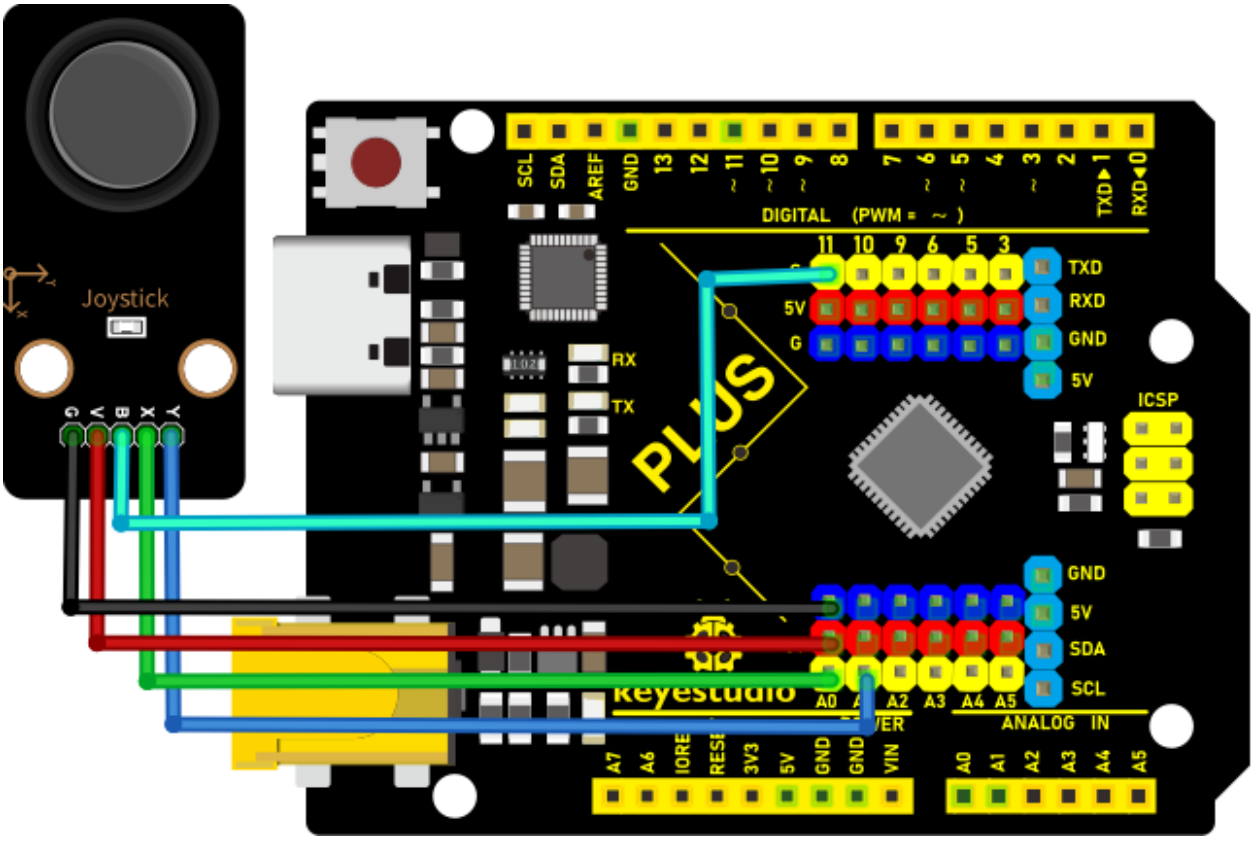
Joystick module: It mainly uses PS2 joystick components. In fact, the joystick module has 3 signal terminal pins, which simulate a three-dimensional space. The pins of the joystick module are GND, VCC, and signal terminals (B, X, Y). The signal terminals X and Y simulate the X-axis and Y-axis of the space. When controlling, the X and Y signal terminals of the module are connected to the analog port of the microcontroller. The signal terminal B simulates the Z axis of the space, it is generally connected to the digital port and used as a button.

VCC is connected to the microcontroller power output VCC (3.3V or 5V), GND is connected to the microcontroller GND, the voltage in the original state is about 1.65V or 2.5V. In the X-axis direction, when moving in the direction of the arrow, the voltage value increases, and the maximum voltage can be reached.

Moving in the opposite direction of the arrow, the voltage value gradually decreases to the minimum voltage. In the Y-axis direction, the voltage value decreases gradually as it moves in the direction of the arrow on the module, decreasing to the minimum voltage. As the arrow is moved in the opposite direction, the voltage value increases and can reach the maximum voltage. In the Z-axis direction, the signal terminal B is connected to the digital port and outputs 0 in the original state and outputs 1 when pressed. In this way, we can read the two analog values and the high and low level conditions of the digital port to determine the operating status of the joystick on the module.

Read the Value

We have to use analog Arduino pins to read the data from X or Y pins, and use digital pins to read the values of the button. Please follow the wiring diagram below for wiring.



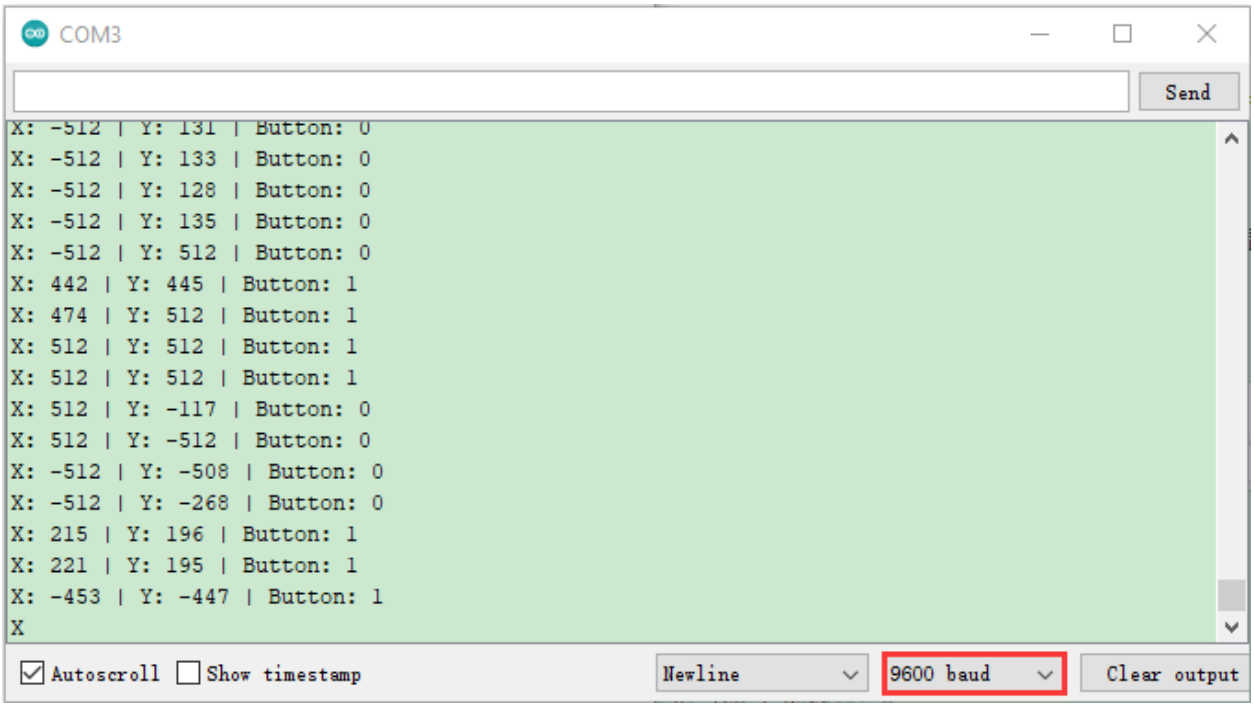

```
/*

Keyestudio 2021 starter learning kit

Project 25.1

Read_the_value_of_the_joystick_module

http//www.keyestudio.com

*/

int VRx = A0;

int VRy = A1;

int SW = 11;

int xPosition = 0;

int yPosition = 0;

int SW_state = 0;
```

(continues on next page)

(continued from previous page)

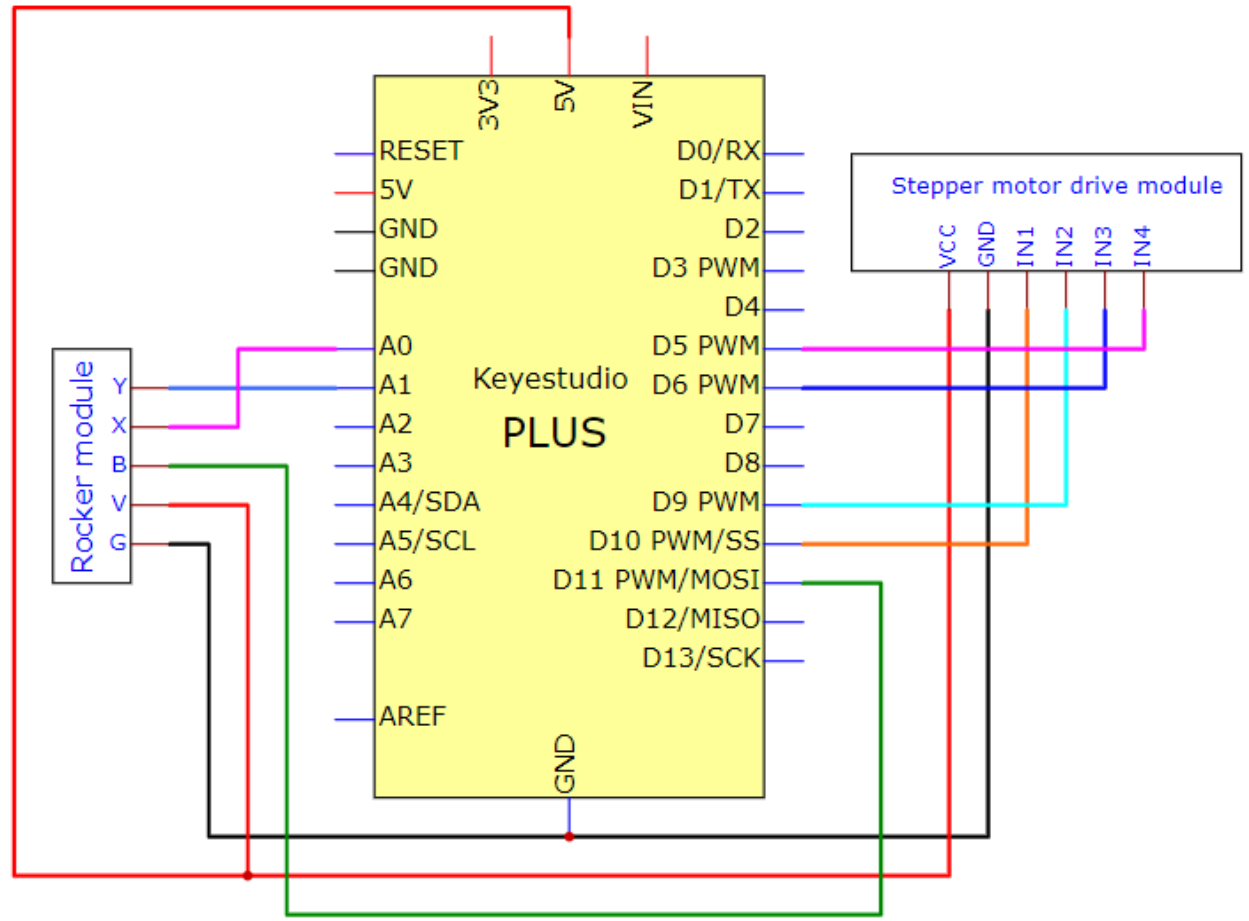
```
int mapX = 0;

int mapY = 0;

void setup() {

Serial.begin(9600);

pinMode(VRx, INPUT);

pinMode(VRy, INPUT);

pinMode(SW, INPUT_PULLUP);

}

void loop() {

xPosition = analogRead(VRx);

yPosition = analogRead(VRy);

SW_state = digitalRead(SW);

mapX = map(xPosition, 0, 1023, -512, 512);

mapY = map(yPosition, 0, 1023, -512, 512);

Serial.print("X: ");

Serial.print(mapX);

Serial.print(" \| Y: ");

Serial.print(mapY);

Serial.print(" \| Button: ");

Serial.println(SW_state);

delay(100);

}
```

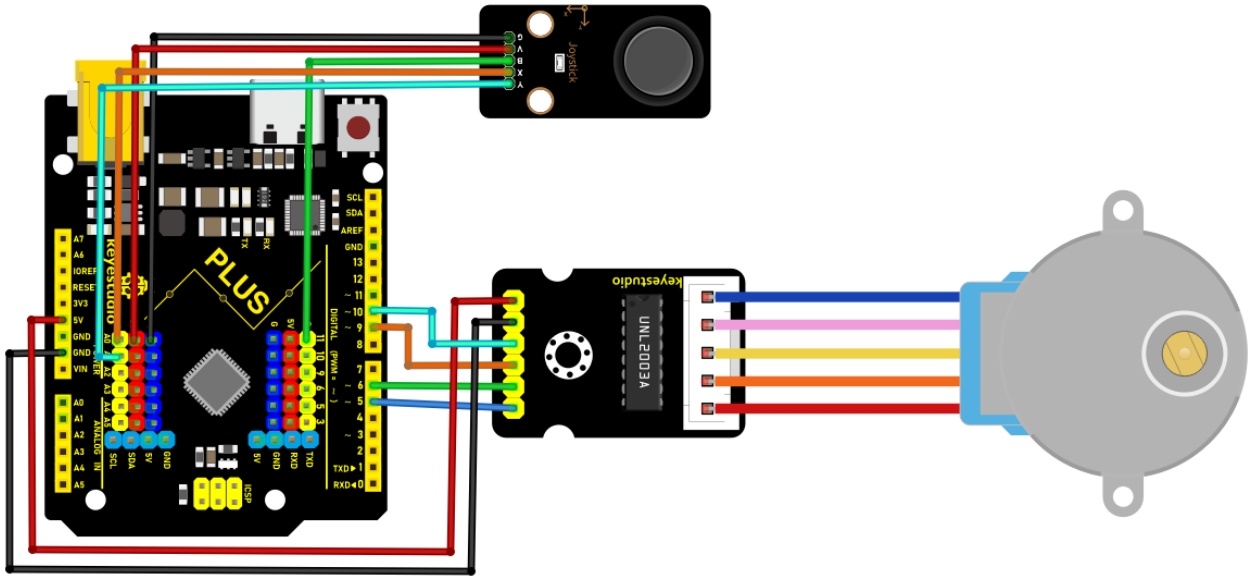
Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor, set the baud rate to 9600.When you shake the joystick or press the button, you can see their values on the serial monitor.



Circuit Diagram and Wiring Diagram

We just read the value of the joystick module. Now we need to do something with the joystick module and stepper motor, connected according to the following diagram.





Code

```
/*

Keyestudio 2021 starter learning kit
```

(continues on next page)

(continued from previous page)

```
Project 25.2

Control Stepper Motor with Joystick

http//www.keyestudio.com

*/

const int X_pin = 0; // analog pin A0 is connected to X

const int Y_pin = 1; // analog pin A1 is connected to Y

int SW_pin = 11;

int X_Rotate;

int Y_Rotate;

//the pin of the stepper motor

const int IN1_pin = 10;

const int IN2_pin = 9;

const int IN3_pin = 6;

const int IN4_pin = 5;

void setup() {

//set the pin of the joystic module

pinMode(SW_pin, INPUT);

digitalWrite(SW_pin, HIGH);

//set the pin of the stepper motor

pinMode(IN1_pin,OUTPUT);

pinMode(IN2_pin,OUTPUT);

pinMode(IN3_pin,OUTPUT);

pinMode(IN4_pin,OUTPUT);

}

void loop() {

X_Rotate = analogRead(X_pin);
```

(continues on next page)

(continued from previous page)

```
Y_Rotate = analogRead(Y_pin);
if (Y_Rotate \< 500) {
digitalWrite(IN1_pin, HIGH);
digitalWrite(IN2_pin, LOW);
digitalWrite(IN3_pin, LOW);
digitalWrite(IN4_pin, LOW);
delay((Y_Rotate/2)+2);
digitalWrite(IN1_pin, LOW);
digitalWrite(IN2_pin, HIGH);
digitalWrite(IN3_pin, LOW);
digitalWrite(IN4_pin, LOW);
delay((Y_Rotate/2)+2);
digitalWrite(IN1_pin, LOW);
digitalWrite(IN2_pin, LOW);
digitalWrite(IN3_pin, HIGH);
digitalWrite(IN4_pin, LOW);
delay((Y_Rotate/2)+2);
digitalWrite(IN1_pin, LOW);
digitalWrite(IN2_pin, LOW);
digitalWrite(IN3_pin, LOW);
digitalWrite(IN4_pin, HIGH);
delay((Y_Rotate/2)+2);
}
else if (Y_Rotate \> 550){
digitalWrite(IN4_pin, HIGH);
digitalWrite(IN3_pin, LOW);
```

(continues on next page)

(continued from previous page)

```
digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, LOW);

delay((1028-Y_Rotate)/2);

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, HIGH);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, LOW);

delay((1028-Y_Rotate)/2);

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, HIGH);

digitalWrite(IN1_pin, LOW);

delay((1028-Y_Rotate)/2);

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, HIGH);

delay((1028-Y_Rotate)/2);

}

else if (Y_Rotate \> 500 && Y_Rotate \< 550) {

digitalWrite(IN4_pin, LOW);

digitalWrite(IN3_pin, LOW);

digitalWrite(IN2_pin, LOW);

digitalWrite(IN1_pin, LOW);

}}
```

Result

Upload the code to the Plus Mainboard, connect the wires and power on first. Push the joystick along the positive direction of the Y axis, and the stepper motor will rotate forward. Conversely, if you push the joystick along the reverse direction of the Y axis, the stepper motor will reverse.

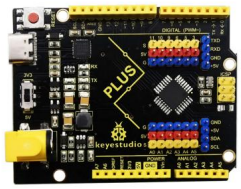
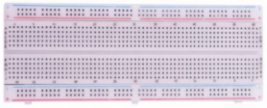
5.26 Project 26: IR Remote Control

Introduction

Infrared remote control is a low-cost, easy-to-use wireless communication technology. IR light is very similar to visible light, except that it has a slightly longer wavelength. This means that infrared rays cannot be detected by the human eye, which is perfect for wireless communication. For example, when you press a button on the TV remote control, an infrared LED will switch on and off repeatedly at a frequency of 38,000 times per second, sending information(such as volume or channel control) to the infrared sensor on the TV.

We will first explain how common IR communication protocols work. Then we will start this project with a remote control and an IR receiving component. We have prepared a home cartoon board. When we press the button of the remote control, the light on the house will be on, and when we press the button again, it will be off.

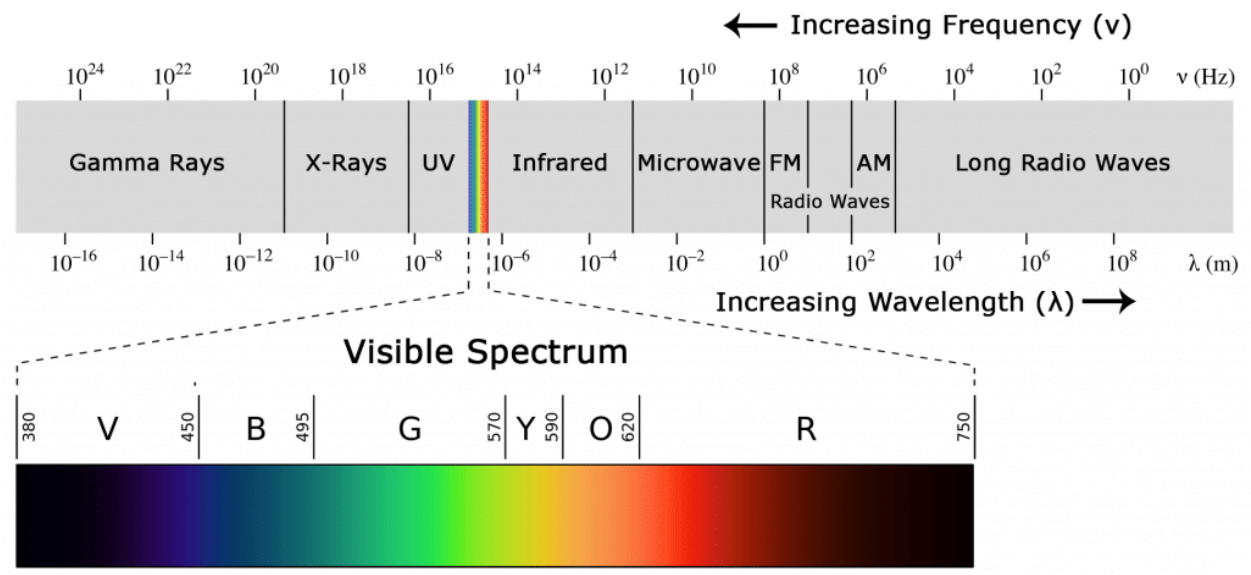
Components Required

				
Keyestudio Plus Mainboardx1	Red LEDx3	220 Resistorx3	Breadboardx1	Jumper Wires
				
IR Remote Controllerx1	IR Receiver x1	10K Resistorx1	USB Cablex1	

Component Knowledge

What is infrared?

Infrared radiation is a form of light similar to the light we see all around us. The only difference between IR light and visible light is the frequency and wavelength. Infrared radiation lies outside the range of visible light, so humans can't see it.



Because IR is a type of light, IR communication requires a direct line of sight from the receiver to the transmitter. It can't transmit through walls or other materials like WiFi or Bluetooth.

How IR and receiver work

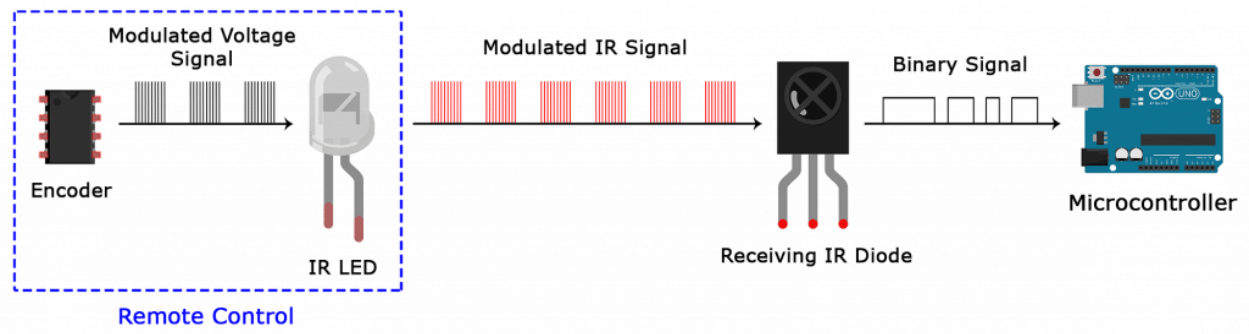
A typical infrared communication system requires an IR transmitter and an IR receiver. The transmitter looks just like a standard LED, except it produces light in the IR spectrum instead of the visible spectrum. If you have a look at TV remote, you'll see the IR transmitter.

LED: 

The IR receiver is a photodiode and pre-amplifier that converts the IR light into an electrical signal. IR receiver diodes typically look like this:

IR signal modulation

IR light is emitted by the sun, light bulbs, and anything else that produces heat. That means there is a lot of IR light noise all around us. To prevent this noise from interfering with the IR signal, a signal modulation technique is used. In IR signal modulation, an encoder on the IR remote controller converts a binary signal into a modulated electrical signal. This electrical signal is sent to the transmitting LED. The transmitting LED converts the modulated electrical signal into a modulated IR light signal. The IR receiver then demodulates the IR light signal and converts it back to binary before passing on the information to a microcontroller.



The modulated IR signal is a series of IR light pulses switched on and off at a high frequency known as the carrier frequency. The carrier frequency used by most transmitters is 38 kHz, because it is rare in nature and thus can be distinguished from ambient noise. This way the IR receiver will know that the 38 kHz signal was sent from the transmitter and not picked up from the surrounding environment. The receiver diode detects all frequencies of IR light, but it has a band-pass filter and only lets through IR at 38 kHz. It then amplifies the modulated signal with a pre-amplifier and converts it to a binary signal before sending it to a microcontroller.

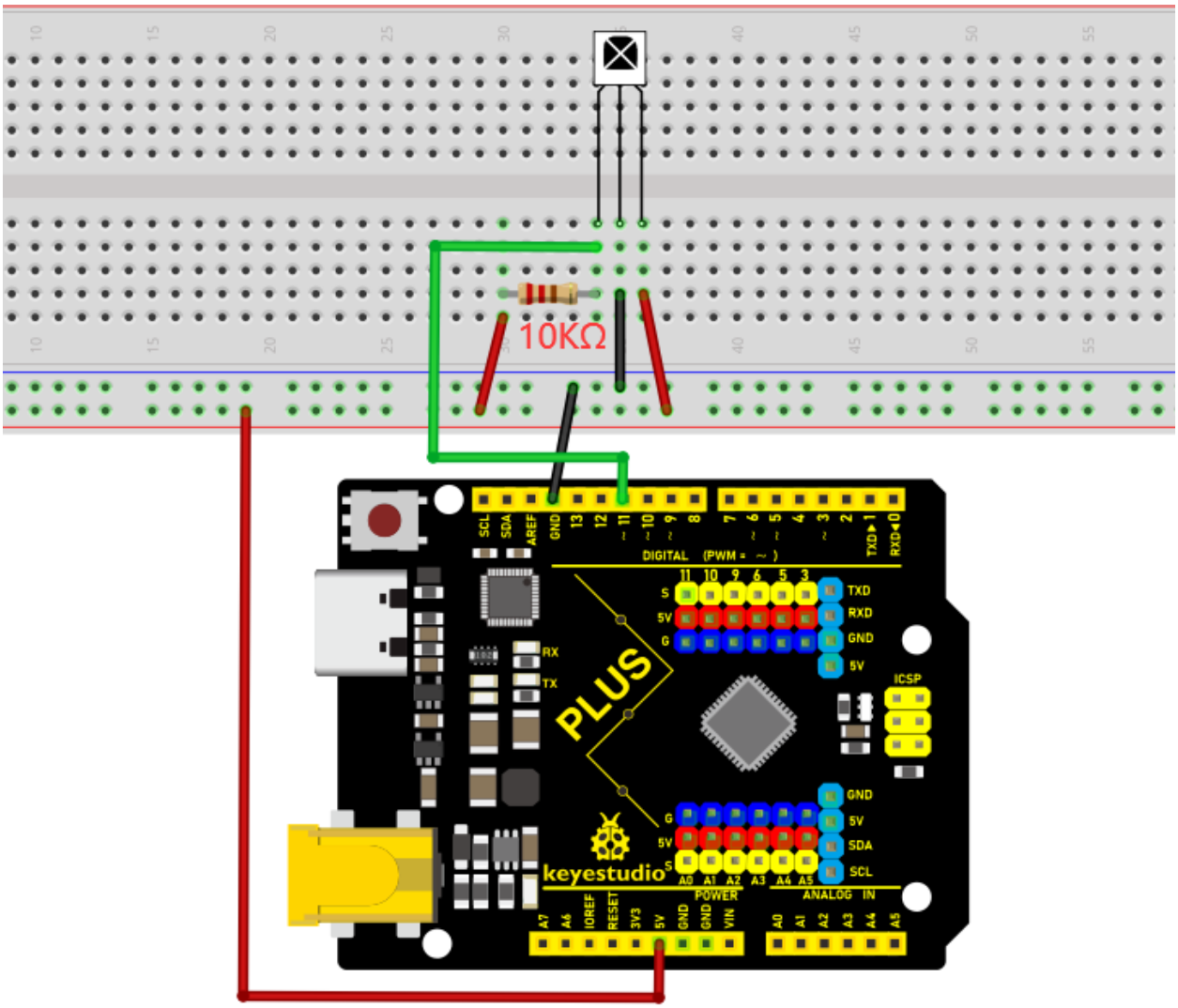
IR Codes:

This is the information that is modulated and sent over IR to the receiver. In order to decipher which key is pressed, the receiving microcontroller needs to know which code corresponds to each key on the remote.

Different remotes send different codes for the buttons, so you'll need to determine the code generated for each key on your particular remote. If you can find the datasheet, the IR key codes should be listed. If not though, there is a simple Arduino sketch that will read most of the popular remote controls and print the hexadecimal codes to the serial monitor when you press a key. I'll show you how to set up in a minute, but first we need to connect the receiver to the Plus mainboard.

Decode the IR Signals

We connect the infrared receiver module to the Plus mainboard according to the wiring diagram below.



Install the IR remote library:

We'll use the IR remote library for all of the code examples below.

NoteThe library file needs to be installed in the code.If the "**Arduino-IRremote-master**" library file has been added, ignore the process of adding the library file below.

Decompress the library files in the folder, that is, put the decompressed "**Arduino-IRremote-master**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

```
/*

Keyestudio 2021 starter learning kit

Project 26.1

Decoded_infrared_signal
```

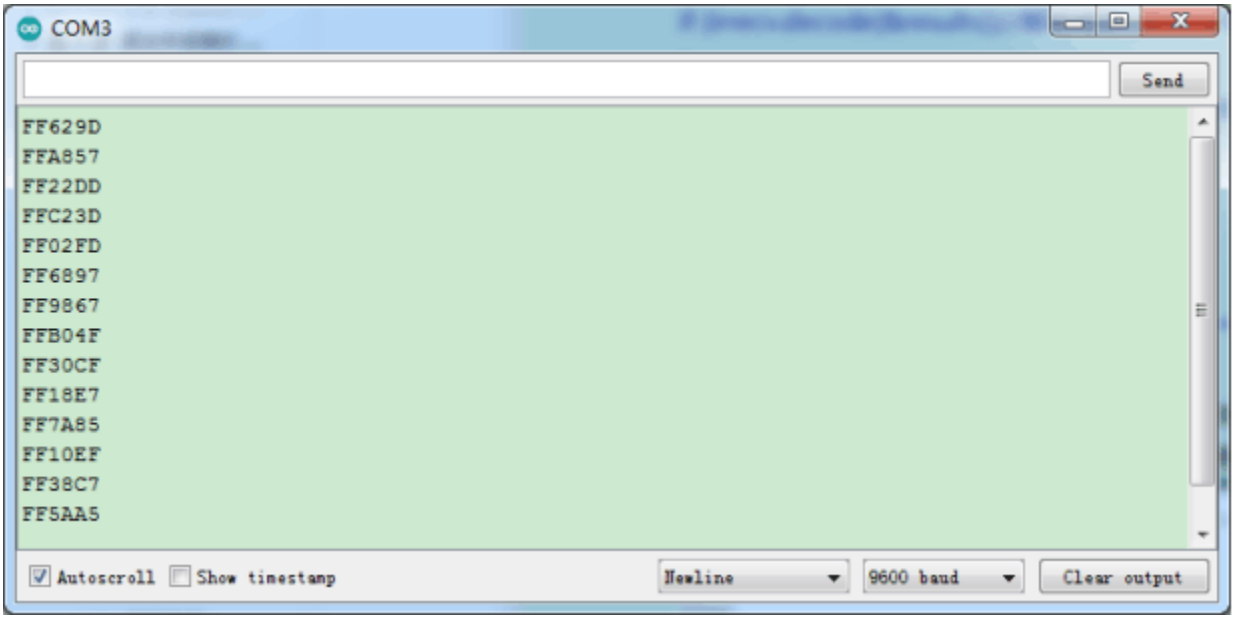
(continues on next page)

(continued from previous page)

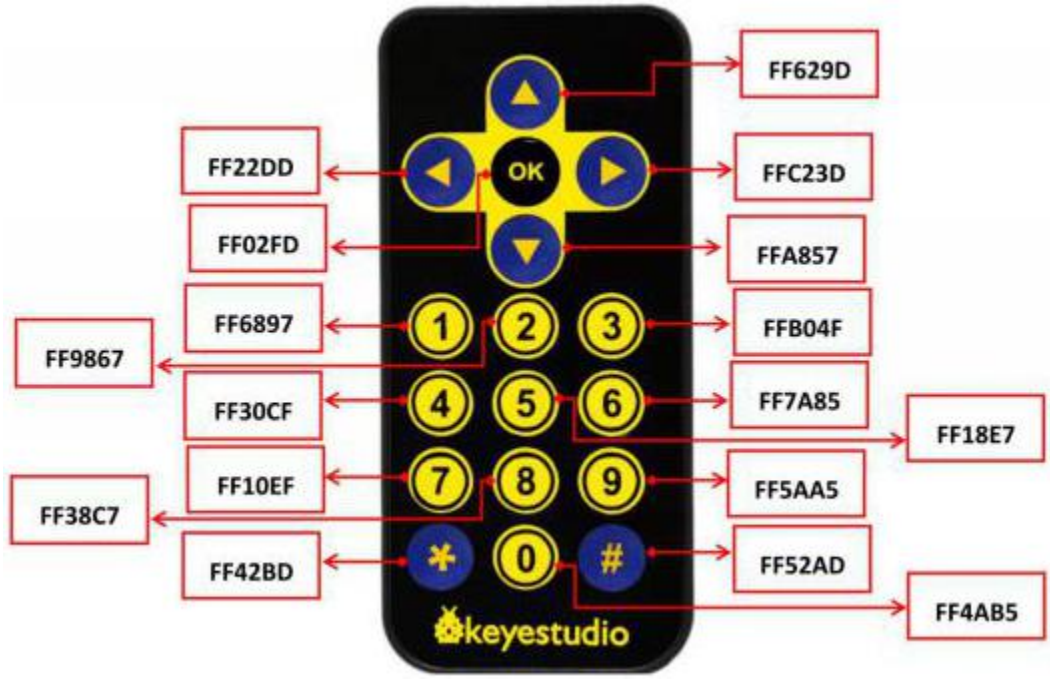
```
http//www.keyestudio.com

*/

\#include \<IRremote.h\>

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()

{

Serial.begin(9600);

irrecv.enableIRIn(); // start receiving signals

}

void loop() {

if (irrecv.decode(&results)) {

Serial.println(results.value, HEX);

irrecv.resume(); // receive the next value

}

delay(100);

}
```

Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor at a baud rate of 9600.

You will see a code on the serial monitor. Press the same button several times to make sure you have the right code for that button. If you see FFFFFFFF, just ignore it.



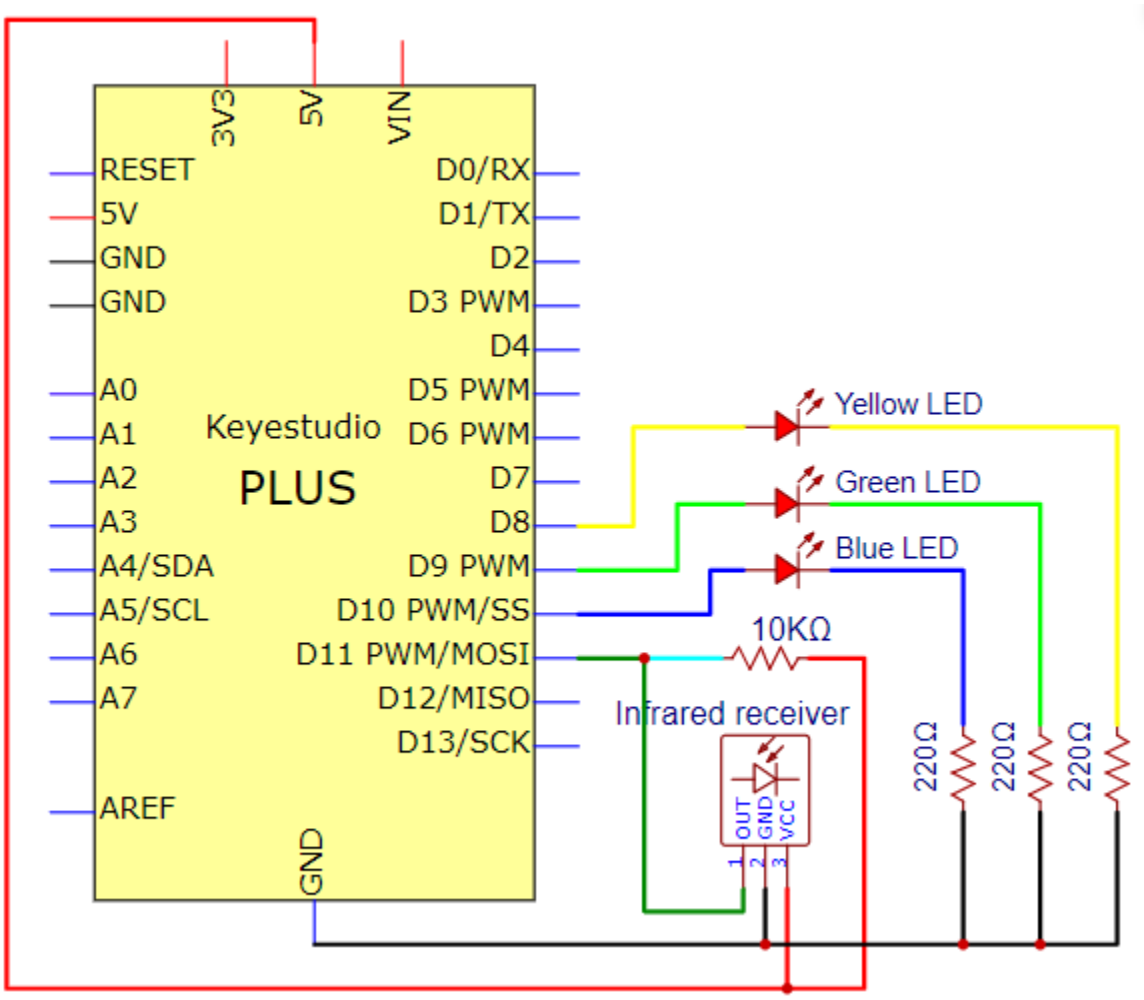
Write down the code associated with each button, because you'll need that information later.

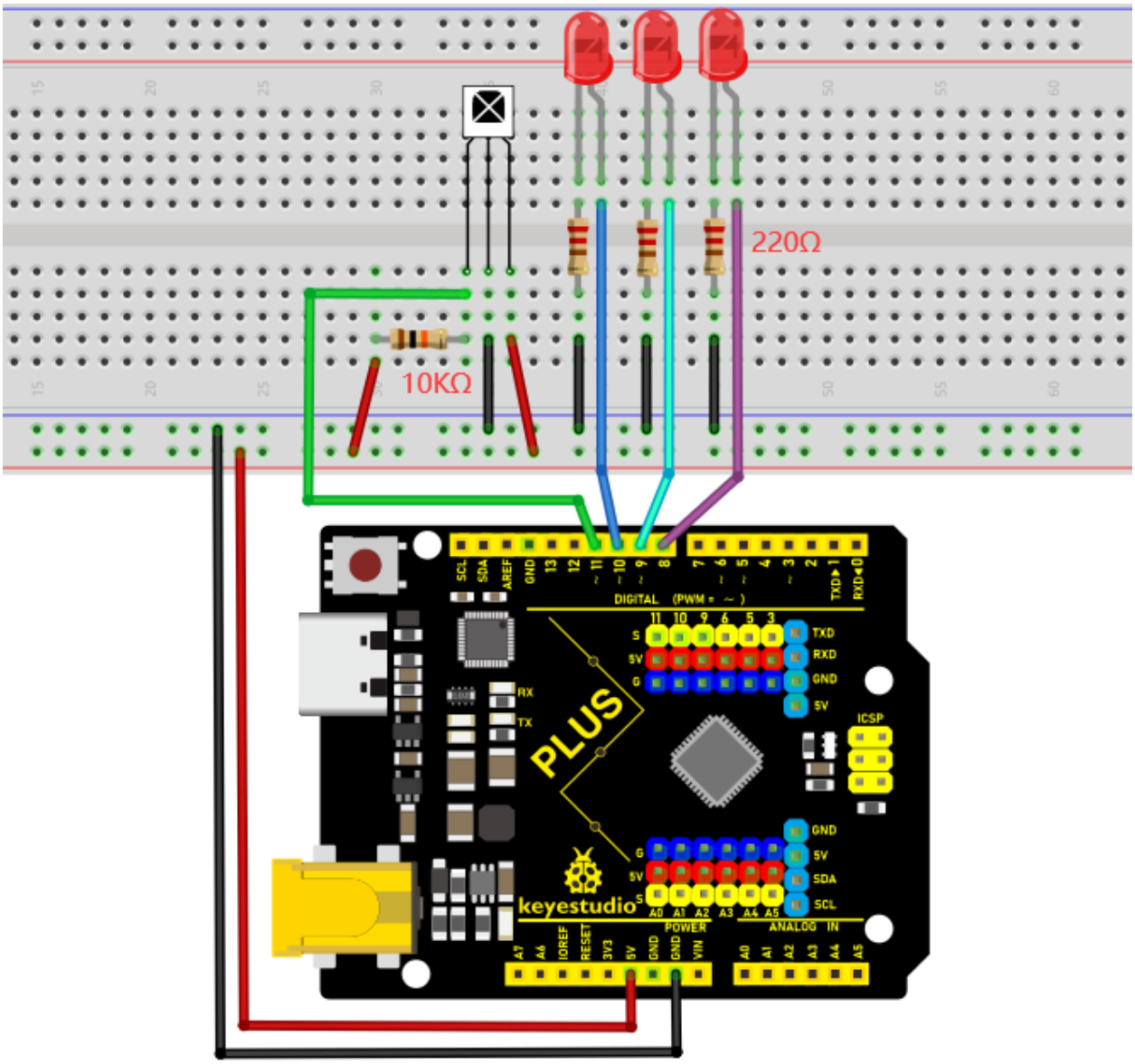


Circuit Diagram and Wiring Diagram

Now I'll show you how to control the Arduino's output pins using IR remote. In this project, we will light up an LED. You can easily modify the code to do things like control servo motors, or activate relays with any button press from the remote.

Connect the LEDs with resistors to pin 8, 9, 10.


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
Keyestudio
PLUS
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Yellow LED
Green LED
Blue LED
10KΩ
Infrared receiver
OUT
GND
VCC
1
2
3
220Ω
220Ω
220Ω



Code

```
/*

Keyestudio 2021 starter learning kit

Project 26.2

IR Remote Control

http//www.keyestudio.com

*/

\#include \<IRremote.h\>
```

(continues on next page)

(continued from previous page)

```
int IR_Recv = 11; //the pin of the IR receiver is 11

int bluePin = 10;

int greenPin = 9;

int yellowPin = 8;

IRrecv irrecv(IR_Recv);

decode_results results;

void setup(){

Serial.begin(9600); //start serial communication

irrecv.enableIRIn(); // start receiving

pinMode(bluePin, OUTPUT); // set the digital pin to OUTPUT

pinMode(greenPin, OUTPUT); // set the digital pin to OUTPUT

pinMode(yellowPin, OUTPUT); // set the digital pin to OUTPUT

}

void loop(){

// decode the IR signals input

if (irrecv.decode(&results)){

long int decCode = results.value;

Serial.println(results.value,HEX);

//switch to case to use the selected remote control button

switch (results.value){

case 0x00FF6897: //when you press the button 1

digitalWrite(bluePin, HIGH);

break;

case 0x00FF30CF: //when you press the button 4

digitalWrite(bluePin, LOW);

break;
```

(continues on next page)

(continued from previous page)

```
case 0x00FF9867: //when you press the button 2

digitalWrite(greenPin, HIGH);

break;

case 0x00FF18E7: //when you press the button 5

digitalWrite(greenPin, LOW);

break;

case 0x00FFB04F: //when you press the button 3

digitalWrite(yellowPin, HIGH);

break;

case 0x00FF7A85: //when you press the button 6

digitalWrite(yellowPin, LOW);

break;

}

irrecv.resume(); // receive the next value

}

delay(10);

}
```

Note: Add "IRremote" folder into installation directory Arduino compiler libraries, or you will fail to compile it.

Result

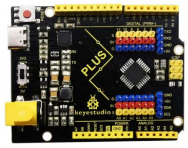
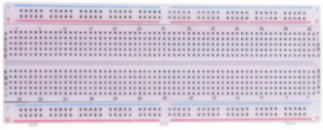
Upload the code to the Plus Mainboard, connect the wires and power on first. Press button 1 and 4 to turn on and off the first LED. Press button 2 and 5 to control the second LED. And press button 3 and 6 to control the state of the third LED.

5.27 Project 27: Temperature Instrument

Introduction

Thermistor is a kind of resistor whose resistance depends on temperature changes. Therefore, we can use this feature to make a temperature instrument.

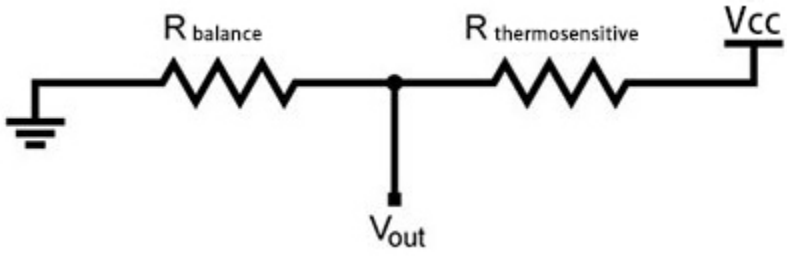
Components Required

			
Keyestudio Plus Mainboardx1	Thermistorx1	LCD_128X32_DOTx1	4.7K Resistorx1
			
M-M Dupont Wires	USB Cablex1	Breadboardx1	Jumper Wires

Component Knowledge

Thermistor: A thermistor is a temperature sensitive resistor. When it senses a change in temperature, the thermistor's resistance changes. We can use this feature to detect temperature intensity with thermistor. This is widely used in gardening, home alarm systems and other devices.

The NTC-MF52AT 10K thermistor is used here, where B is 3950 and it is connected in series with RS (RS=Rbalance=4.7K resistor). The resistance value of the thermistor changes as the temperature changes.



Calculation of NTC thermistor:
Calculation formula: Rt = RxEXP[Bx(1/T1-1/T2)]

Among them, T1 and T2 refer to K degrees, that is, Kelvin temperature.

Rt is the resistance of the thermistor at T1 temperature.

R is the nominal resistance value of the thermistor at T2 room temperature. The value of the 10K thermistor at 25℃ is 10K (R=10K). T2 = (273.15 + 25).

EXP[n] is the nth power of e.

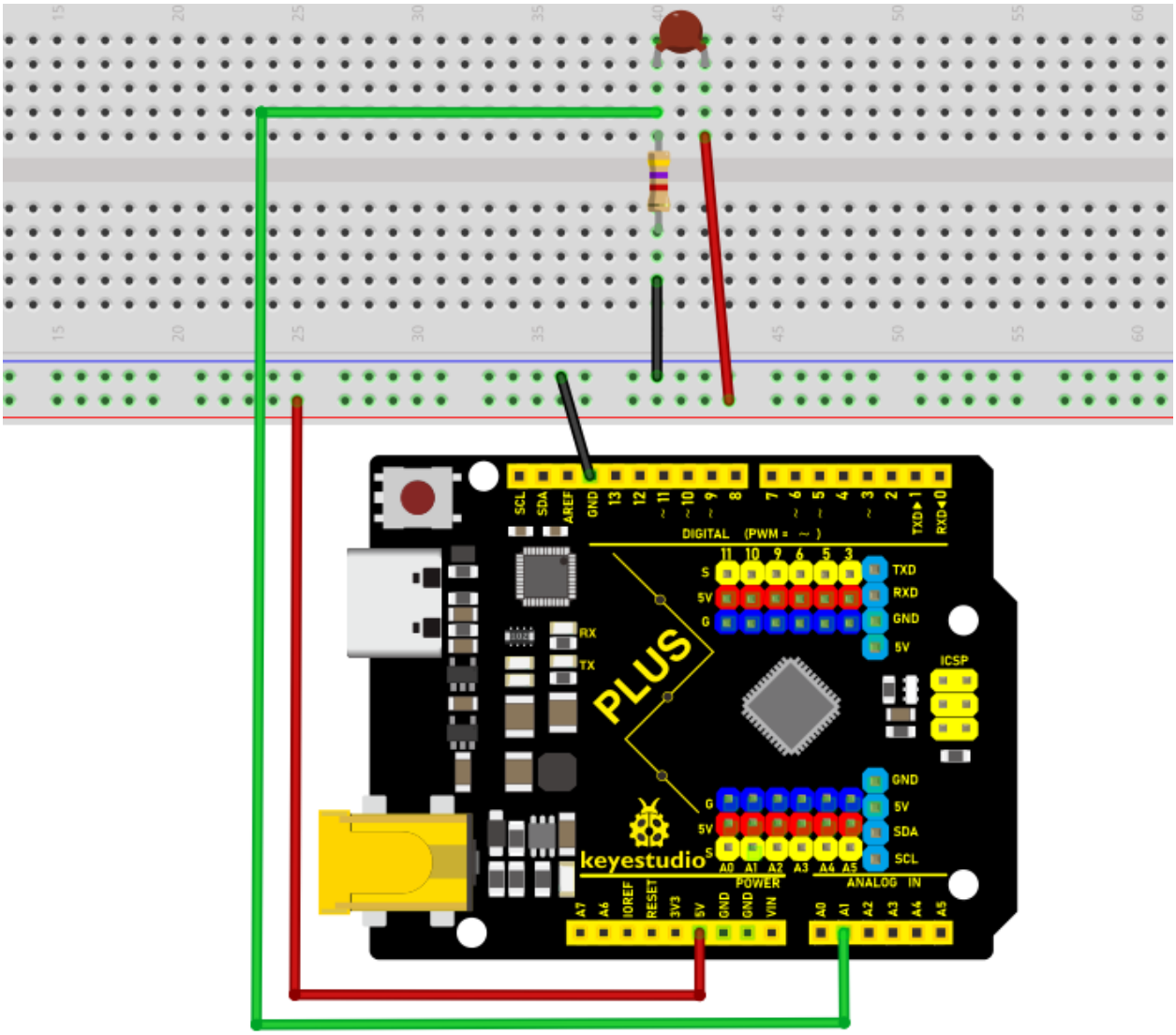
B is an important parameter of thermistor, B equals 3950.

We can use the value measured by the ADC converter to get the resistance value of the thermistor, and then use the formula to get the temperature value.

t=((T1xB)/(B+T1xln(Rt/R1)))-273.15"ln"can be converted to "log", that is , t=((T1xB)/(B+T1xlog(Rt/R1)))-273.15. Error is ±0.5.

Read the Values

First we learned how to use the serial monitor to print the thermistor values. Please connect the wires according to the following wiring diagram.



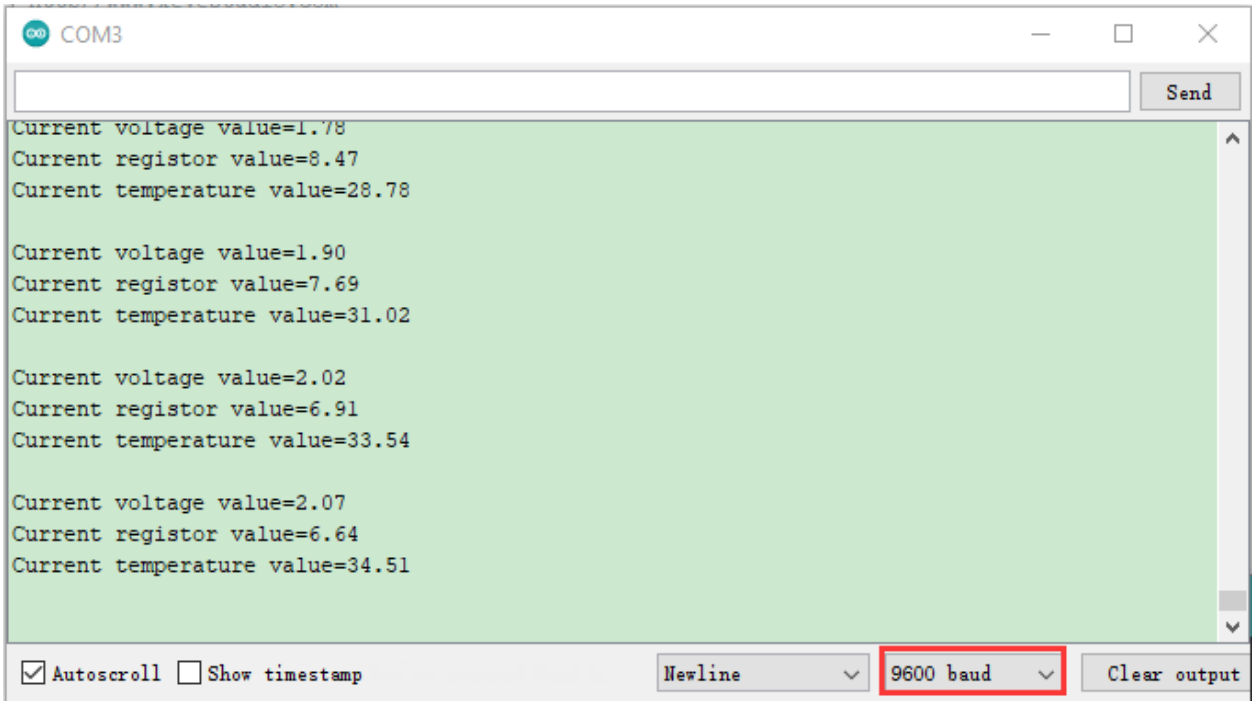

```
/*

Keyestudio 2021 starter learning kit

Project 27.1

Read_the_thermistor_analog_value

http//www.keyestudio.com

*/

\#include\<math.h\>

const float voltagePower=5.0;

const float Rs=4.7;//sample resistance is 4.7K
```

(continues on next page)

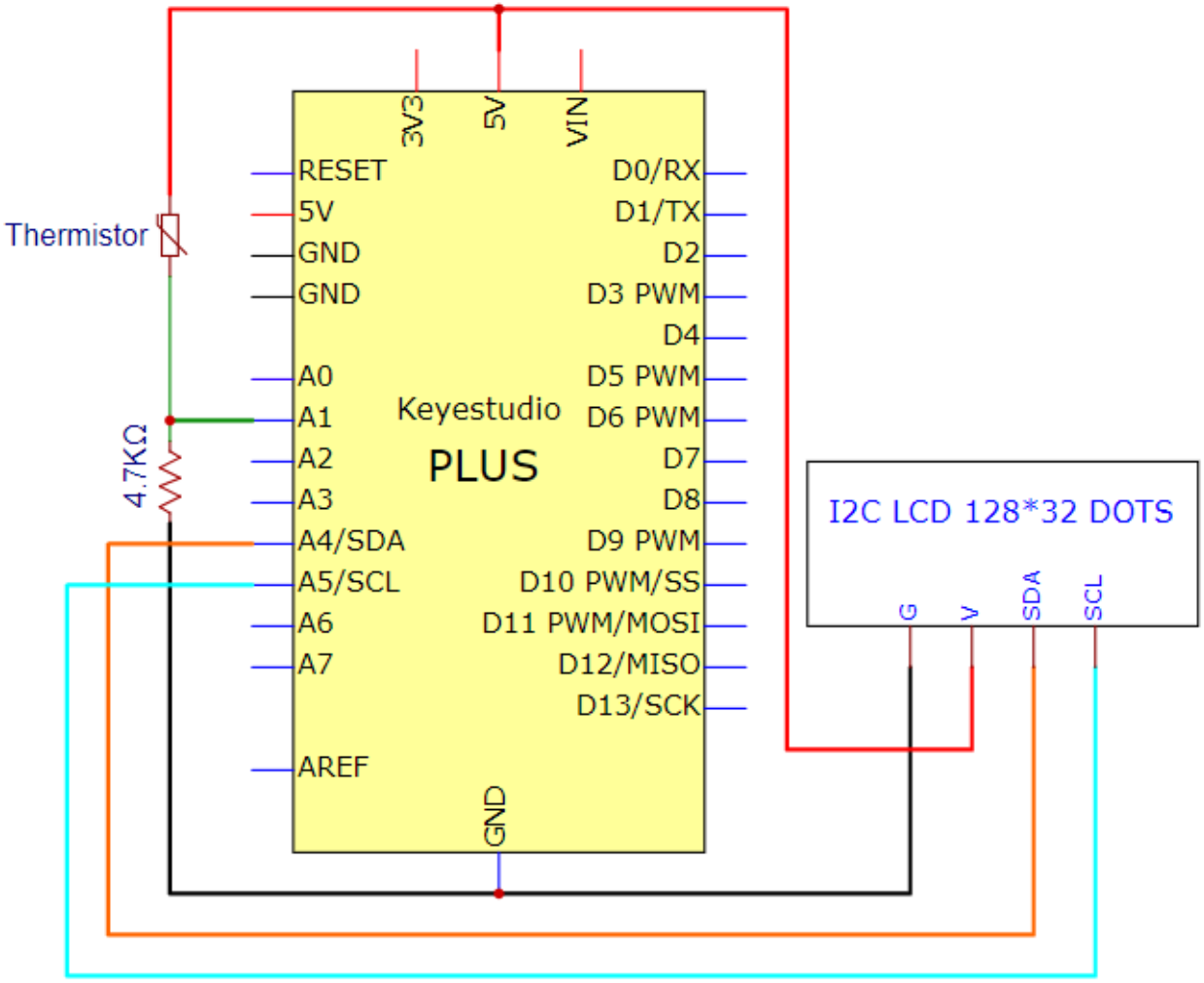
(continued from previous page)

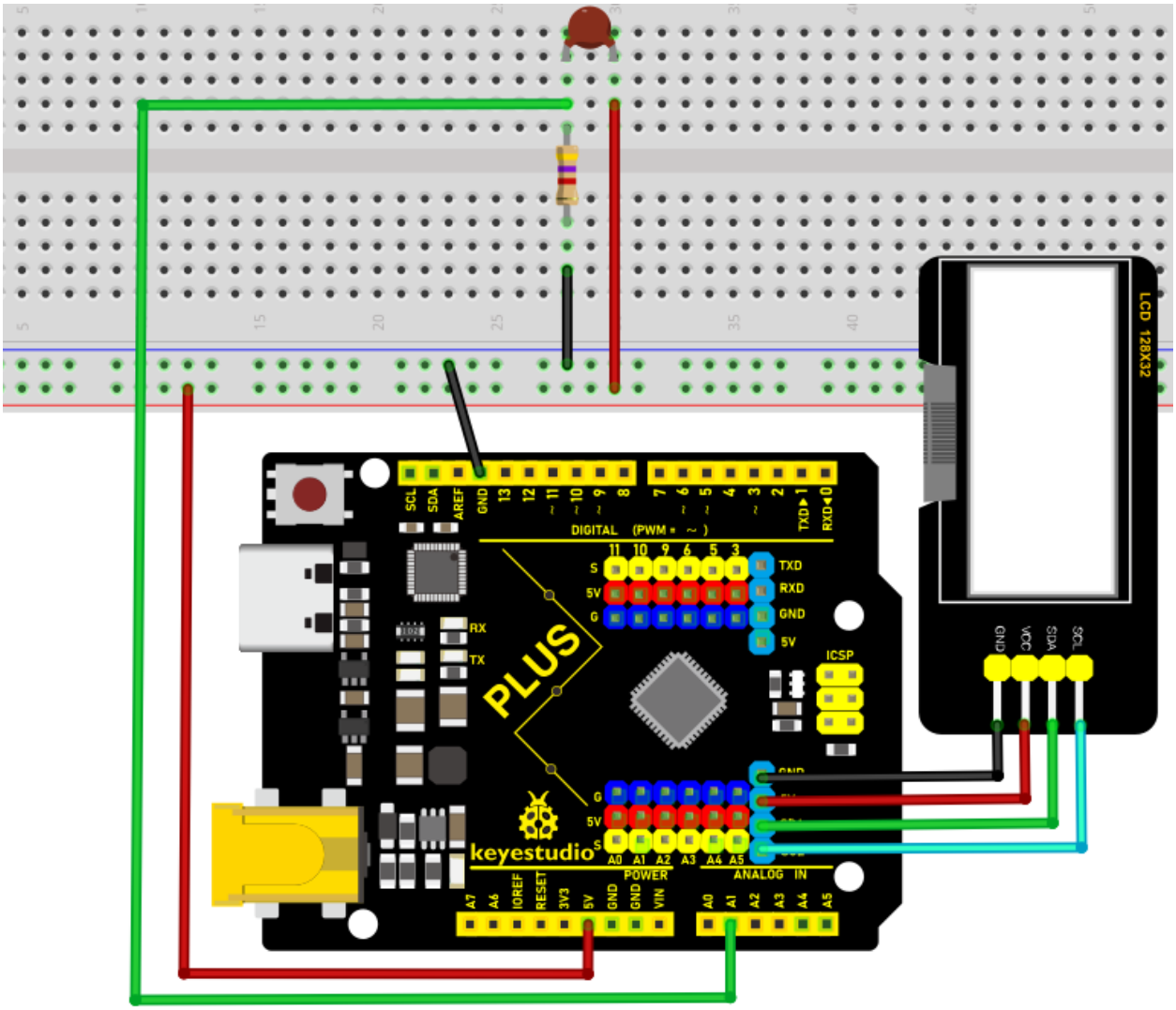
```
const int B=3950;

const double T1=273.15+25;//ordinary temperature

const double R1=10;//ordinary temperature corresponds the resistance value, unit
is K

void setup() {

Serial.begin(9600);

}

void loop() {

//Attain the voltage value at A1

double digitalValue=analogRead(1);

double voltageValue=(digitalValue/1023)x5;

Serial.print("Current voltage value=");

Serial.println(voltageValue);

//Obtain the resistance of the thermistor through the voltage divider ratio

double Rt=((voltagePower-voltageValue)xRs)/voltageValue;

Serial.print("Current registor value=");

Serial.println(Rt);

//Converted to get the temperature value

Serial.print("Current temperature value=");

Serial.println(((T1xB)/(B+T1xlog(Rt/R1)))-273.15);//

Serial.println();

//output for each 3 second, change the frequency

delay(3000);

}
```

Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor, the voltage value at thermistor pin A1 can be read, and the resistance value and temperature value of the thermistor can be obtained through the voltage division ratio. As shown below.



Circuit Diagram and Wiring Diagram


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
Thermistor
4.7KΩ
I2C LCD 128*32 DOTS
G
V
SDA
SCL



Code

xxNotexxThe "**LCD_128X32**"library file needs to be installed in the code. If the library file has been added, ignore the process of adding the library file below.

Project 16 contains the library files of I2C 128×32 LCD. Decompress the library files in the folder, that is, put the decompressed "**LCD_128X32**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

```
/*

Keyestudio 2021 starter learning kit

Project 27.2

Temperature_Instrument

http//www.keyestudio.com
```

(continues on next page)

(continued from previous page)

```
*/

\#include \<math.h\>

\#include \<lcd.h\> //add library files

lcd Lcd; //define a Lcd class instance

const float voltagePower=5.0;

const float Rs_val=4.7;//sample resistance is 4.7K

const int B=3950;

const double T1=273.15+25;//normal temperature

const double R1=10;//ordinary temperature corresponds the resistance value, unit
is K

char string[10];

void setup(){

Serial.begin(9600);

Lcd.Init(); //initialize

Lcd.Clear(); //clear

}

void loop(){

// attain the voltage value at A1

double digitalValue=analogRead(1);

double voltageValue=(digitalValue/1023)x5;

//Obtain the resistance of the thermistor through the voltage divider ratio

double Rt=((voltagePower-voltageValue)xRs_val)/voltageValue;

//Converted to get the temperature value

const float t=((T1xB)/(B+T1xlog(Rt/R1)))-273.15;

if(t\>-100.0) //If the temperature is greater than -100°C, the LCD display
voltage value,obtains the resistance value of the thermistor through the voltage
division ratio and temperature value
```

(continues on next page)

(continued from previous page)

```
{

Lcd.Cursor(0,0);

Lcd.Display("C v v=");

Lcd.Cursor(0,7);

Lcd.DisplayNum(voltageValue);

Lcd.Cursor(0, 10);

Lcd.Display("V");

Lcd.Cursor(1,0);

Lcd.Display("C r v=");

Lcd.Cursor(1,7);

Lcd.DisplayNum(Rt);

Lcd.Cursor(1, 10);

Lcd.Display("R");

Lcd.Cursor(2, 0);

Lcd.Display("C t v=");

Lcd.Cursor(2, 7);

Lcd.DisplayNum(t);

Lcd.Cursor(2, 10);

Lcd.Display("C");

}

delay(300);

}
```

Result

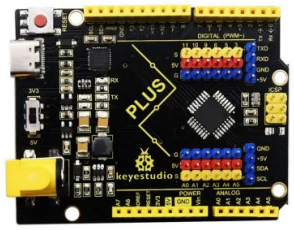
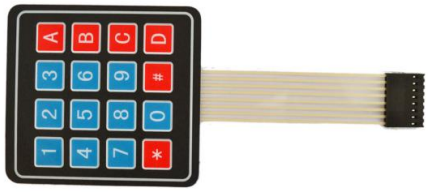
After uploading the code to the Plus Mainboard, connecting the wires and powering on, the LCD_128X32_DOT displays the voltage value of the corresponding A1 pin, obtain the resistance value of the thermistor through the voltage division ratio and the temperature value in the current environment .Only integers can be displayed, not decimals on the LCD_128X32_DOT

5.28 Project 28: Control the LED with 4x4 Matrix Keyboard

Introduction

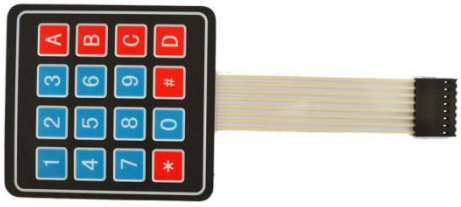
The commonly used digital button sensor uses an IO port for a button. Sometimes we need more buttons, it will take up too many IO ports. To reduce the use of IO ports, multiple buttons are made into matrix types. Through the control of the point of intersection, realize the control of multiple buttons with less IO ports. In this lesson, we will learn about the 4x4 thin-film matrix keyboard.

Components Required

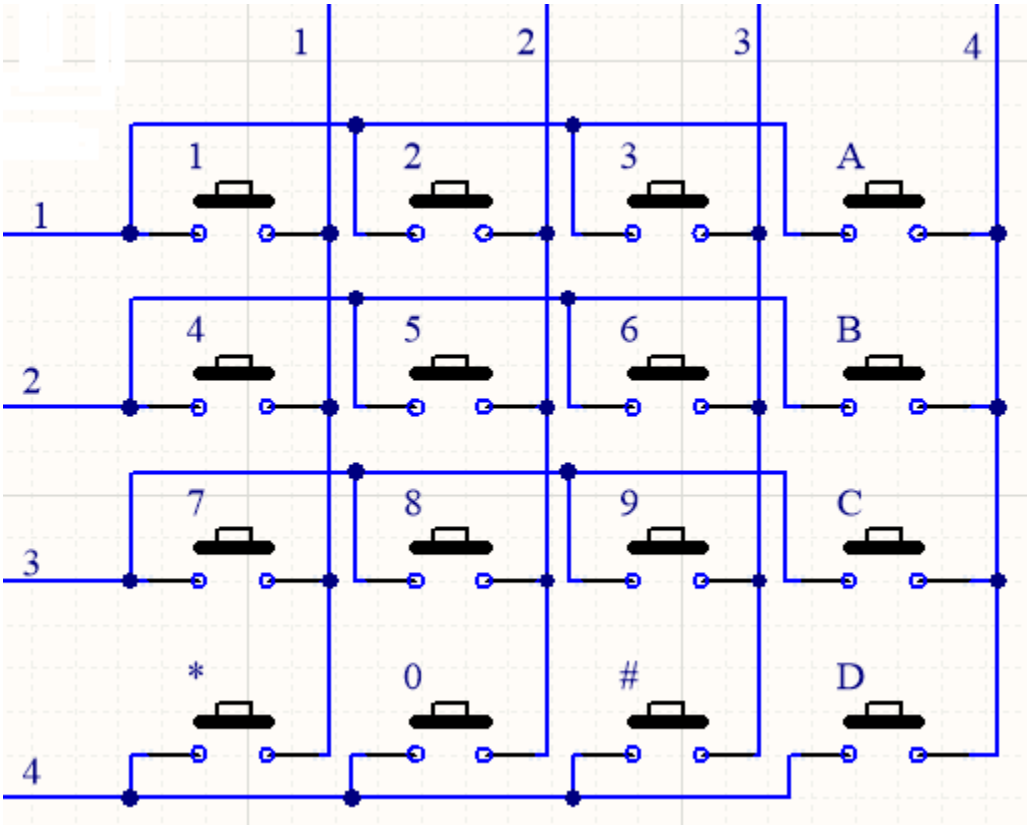
			
Keyestudio Plus Mainboardx1	4x4 Thin-film Matrix Keyboardx1	USB Cablex1	Jumper Wires

Component Knowledge

4x4 Matrix keyboard: The keyboard is a device that integrates many keys. As shown in the figure below, a 4x4 keyboard integrates 16 keys.



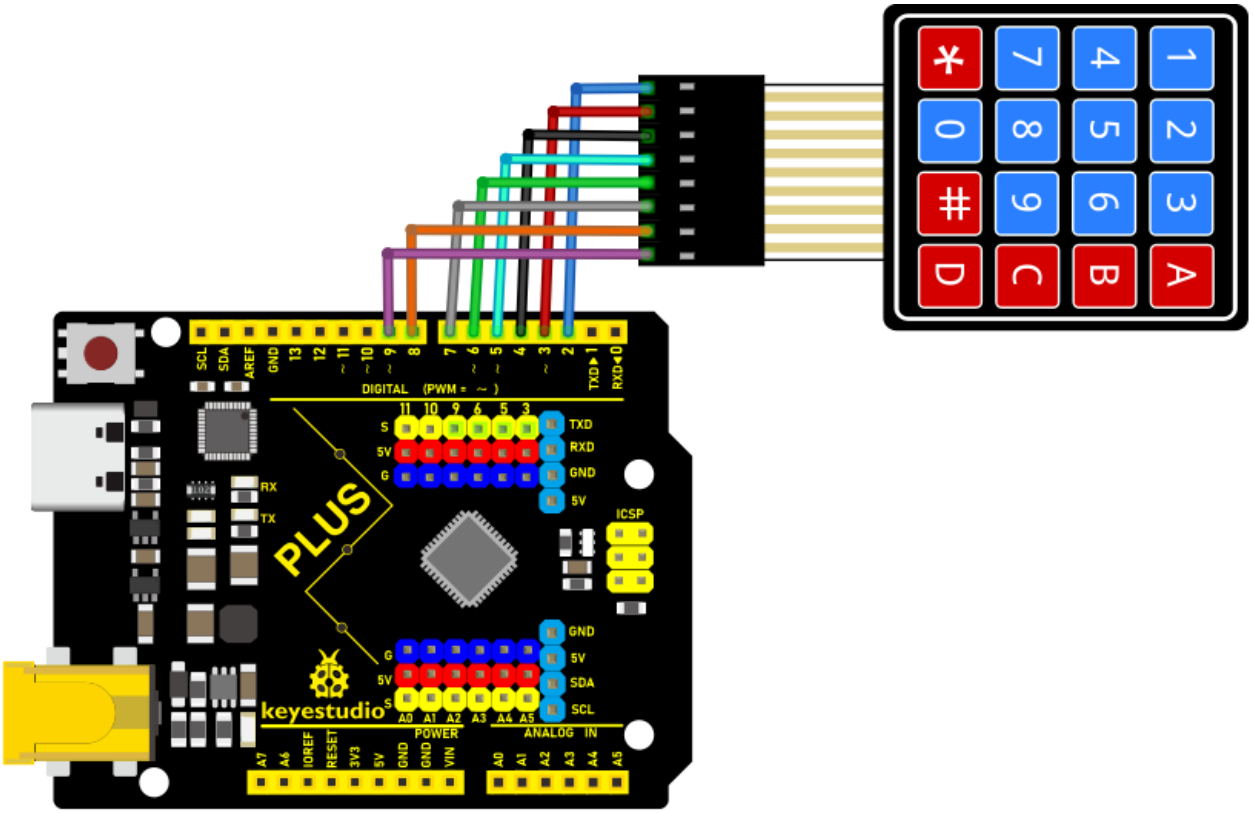
As with the LED matrix integration, in the 4x4 keyboard, each row of keys is connected to a pin, each column of keys is the same. This connection reduces the use of processor ports. The internal circuit is shown below.



The usage is similar to matrix LED, that is, using row scan or column scan methods to detect the state of the keys on each column or each line. Take the column scan method as an example. Send a low level to column 4 (Pin4), detect the state of rows 1, 2, 3 and 4, and determine whether the A, B, C and D keys are pressed. Then send the low level to columns 3, 2, 1 in turn, and detect whether other keys are pressed. Then you can get the state of all keys.

Read the Value

We start with a simple code to read the values of the 4x4 matrix keyboard and print them in the serial monitor. Its wiring diagram is shown below.



Install the Keypad Library:

We will use the Keypad library in all the following code examples.

xxNotexxThe library files are required in the code. If you have already added the "**Keypad**" library files ignore the process of adding the library file below.

Decompress the library files in the folder, that is, put the decompressed "**Keypad**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

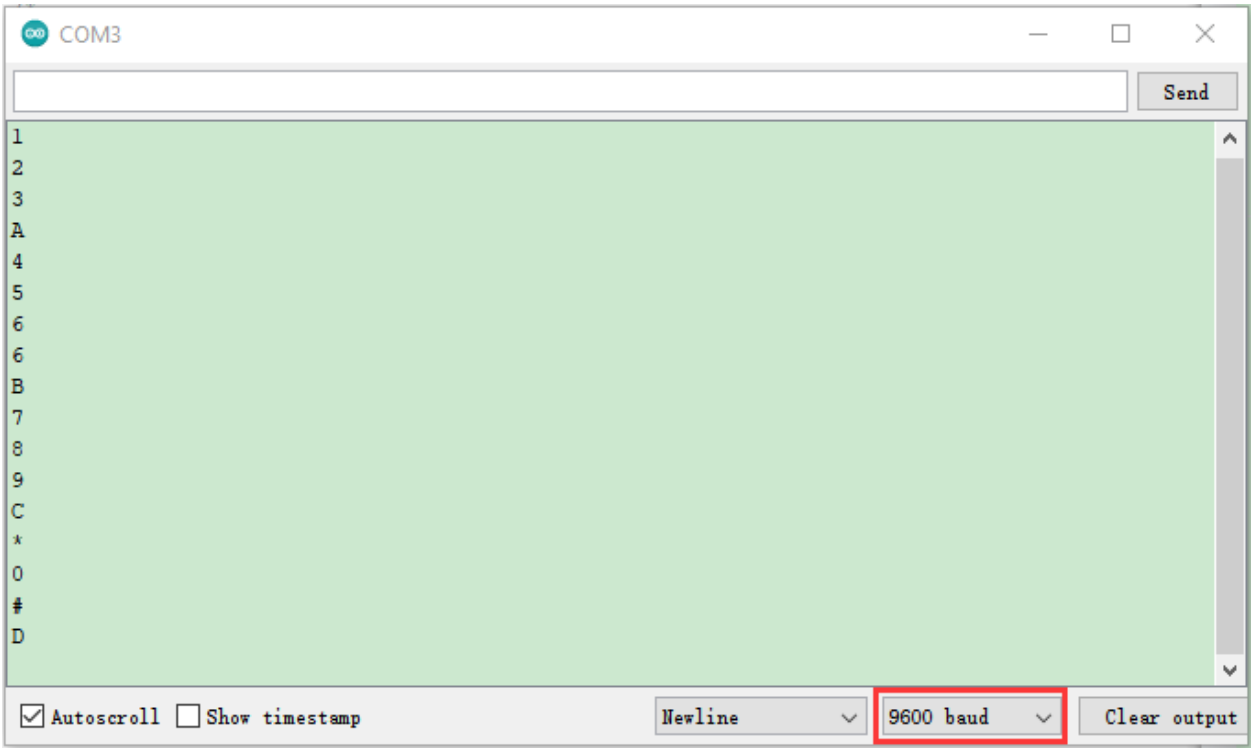
```
/*

Keyestudio 2021 starter learning kit

Project 28.1

4x4_Keypad_display

http//www.keyestudio.com

*/

\#include \<Keypad.h\>

const byte ROWS = 4; // define the row 4
```

(continues on next page)

(continued from previous page)

```
const byte COLS = 4; // define the row 4

char keys[ROWS][COLS] = {

{'1','2','3','A'},

{'4','5','6','B'},

{'7','8','9','C'},

{'x','0','\#','D'}

};

// connect the port of 4x4 keypad to the corresponding digital port of the
control board

byte rowPins[ROWS] = {2,3,4,5};

// connect the port of 4x4 keypad to the corresponding digital port of the
control board

byte colPins[COLS] = {6,7,8,9};

// call on the corresponding functions from libraries

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){

Serial.begin(9600);

}

void loop(){

char key = keypad.getKey();

if (key != NO_KEY){

Serial.println(key);

}

}
```

Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor, press the button of the 4x4 matrix keyboard to read the corresponding values in the open serial monitor. For example, we can see from the image below that when we press “#”, it will display “#”.



Code(4x4 Keyboard Control LED Experiment)

Use the above connection diagram. Now we use the 4 x 4 matrix keyboard to control the LED on the pin 13 of the Mainboard.

```
/*

Keyestudio 2021 starter learning kit

Project 28.2

Control the LED with 4x4 Matrix Keyboard

http//www.keyestudio.com

*/

\#include \<Keypad.h\>

const byte ROWS = 4; // define row 4

const byte COLS = 4; // define row 4

char keys[ROWS][COLS] = {

{'1','2','3','A'},

{'4','5','6','B'},

{'7','8','9','C'},
```

(continues on next page)

(continued from previous page)

```
{'x','0','\#','D'}

};

// connect the port of 4x4 keypad to the corresponding digital port of the
control board

byte rowPins[ROWS] = {2,3,4,5};

// connect the port of 4x4 keypad to the corresponding digital port of the
control board

byte colPins[COLS] = {6,7,8,9};

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

byte ledPin = 13;

boolean blink = false;

void setup(){

Serial.begin(9600);

pinMode(ledPin, OUTPUT); // set the digital pins to OUTPUT

digitalWrite(ledPin, HIGH); // set LED to light up

keypad.addEventListener(keypadEvent); //add EventListener to the keyboard

}

void loop(){

char key = keypad.getKey();

if (key != NO_KEY) {

Serial.println(key);

}

if (blink){

digitalWrite(ledPin,!digitalRead(ledPin));

delay(100);

}

}
```

(continues on next page)

(continued from previous page)

```
//

void keypadEvent(KeypadEvent key){

switch (keypad.getState()){

case PRESSED:

switch (key){

case '\#': digitalWrite(ledPin,!digitalRead(ledPin)); break;

case 'x':

digitalWrite(ledPin,!digitalRead(ledPin));

break;

}

break;

case RELEASED:

switch (key){

case 'x':

digitalWrite(ledPin,!digitalRead(ledPin));

blink = false;

break;

}

break;

case HOLD:

switch (key){

case 'x': blink = true; break;

}

break;

}

}
```

Note: Add the "Keypad" library folder we provided in the corresponding folder to the installation directory Arduino compiler library, otherwise the compilation will not work.

Result

Upload the code to the Plus Mainboard, connect the wires and power on first. When you press the x button, the LED of the pin 13 on the Plus Mainboard will always be on until you release it. When you press the # button and then release it, the LED will stay on until you press the button again.

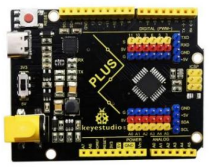
5.29 Project 29: Temperature and Humidity Meter

Introduction

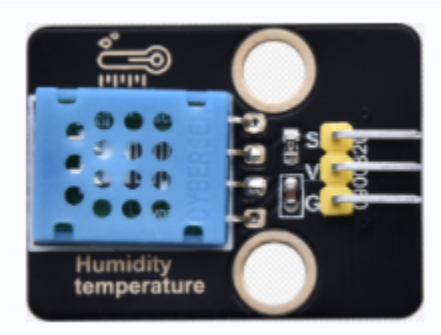
In winter, the humidity in the air is very low, that is, the air is very dry. Coupled with the cold, the human skin is prone to crack from excessive dryness. Therefore, you need to use a humidifier to increase the humidity of the air at home. But how do you know that the air is too dry? Then you need equipment to detect air humidity.

In this lesson, we will learn how to use the XHT11 temperature and humidity sensor. We use the XHT11 temperature and humidity sensor to create a thermometer and also combined with an LCD_128X32_DOT to display the temperature and humidity values.

Components Required

				
Keyestudio Plus Mainboardx1	Temperature and Tumidity sensorx1	LCD_128X32_DOTx1	F-F Dupont Wires	USB Cablex1

Component Knowledge



XHT11 temperature and humidity sensor: It is a temperature and humidity composite sensor with calibrated digital signal output. Its accuracy humidity is ±5%RH, temperature is ±2℃. Range humidity is 20 to 90%RH, and temperature is 0 to 50℃. The XHT11 temperature and humidity sensor applies dedicated digital module acquisition technology and temperature and humidity sensing technology to ensure extremely high reliability and excellent long-term stability of the product. The XHT11 temperature and humidity sensor includes a resistive-type humidity measurement and an NTC temperature measurement component, which is very suitable for temperature and humidity measurement applications where accuracy and real-time performance are not required.

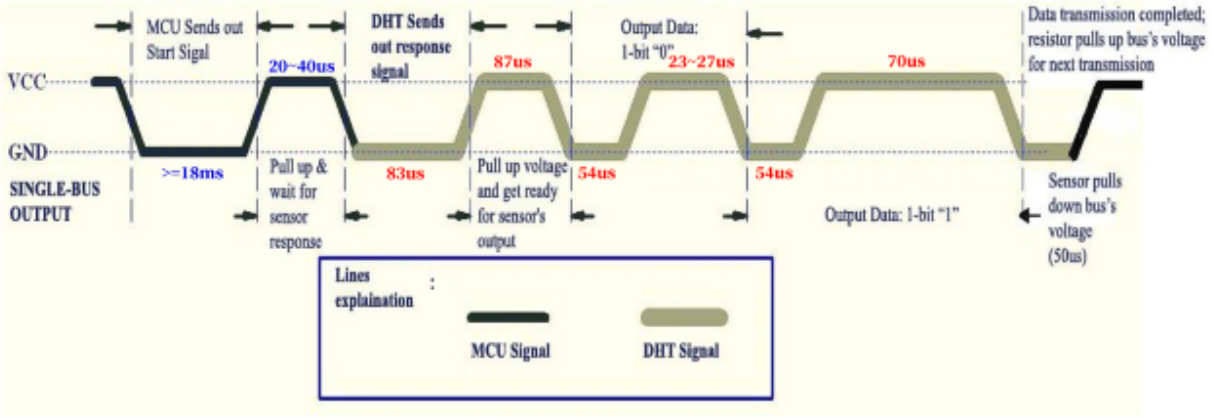
The operating voltage is in the range of 3.3V to 5.5V. XHT11 has three pins, which are VCC, GND, and S. S is the pin for data output, using serial communication.

Single bus format definition:

Description	Definition
Start signal	Microprocessor pulls data bus (SDA) down at least 18ms for a period of time(Maximum is 30ms), notifying the sensor to prepare data.
Response signal	The sensor pulls the data bus (SDA) low for 83μs, and then pulls up for 87μs to respond to the host's start signal.
Humidity	The high humidity is an integer part of the humidity data, and the low humidity is a fractional part of the humidity data.
Temperature	The high temperature is the integer part of the temperature data, the low temperature is the fractional part of the temperature data. And the low temperature Bit8 is 1, indicating a negative temperature, otherwise, it is a positive temperature.
Parity bit	Parity bit=Humidity high bit+ Humidity low bit+temperature high bit+temperature low bit

Data sequence diagram:

When MCU sends a start signal, XHT11 changes from the low-power-consumption mode to the high-speed mode, waiting for MCU completing the start signal. Once it is completed, XHT11 sends a response signal of 40-bit data and triggers a signal acquisition. The signal is sent as shown in the figure.



Combined with the code, you can understand better.

The XHT11 temperature and humidity sensor can easily add temperature and humidity data to your DIY electronic projects. It is perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.

Specification:

Working voltage: +5V

Temperature range: 0°C to 50°C, error of ± 2°C

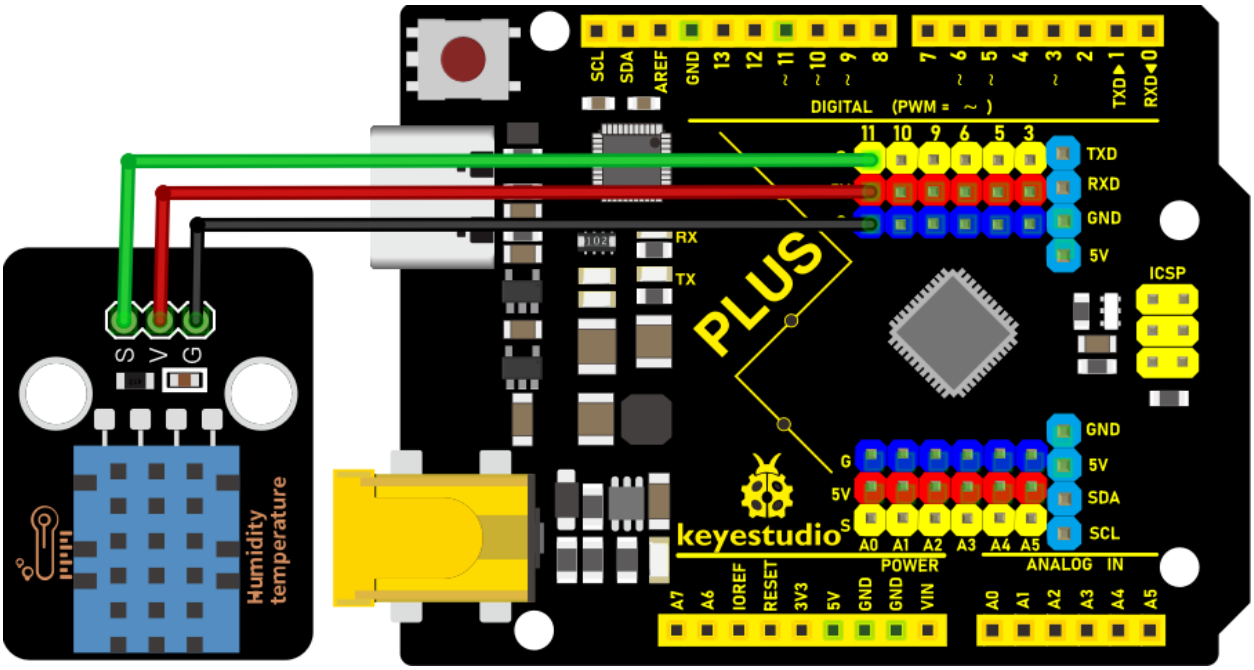
Humidity range: 20% to 90% RH,± 5% RH error

Digital interface

Schematic diagram:

Read the Value

First we learned how to use the serial monitor to print the values of the XHT11 sensor. Please connect the wires according to the wiring diagram below.



NoteThe library file needs to be installed in the code.If the "**Dht11**" library file has been added, ignore the process of adding the library file below.

Decompress the library files in the folder, that is, put the decompressed "**Dht11**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

```
/*

Keyestudio 2021 starter learning kit

Project 29.1

Read_the_temperature_and_humidity_values

http//www.keyestudio.com

*/

\#include "DHT.h"

\#define DHTPIN 11 // connect the DHT sensor to the digital pins
```

(continues on next page)

(continued from previous page)

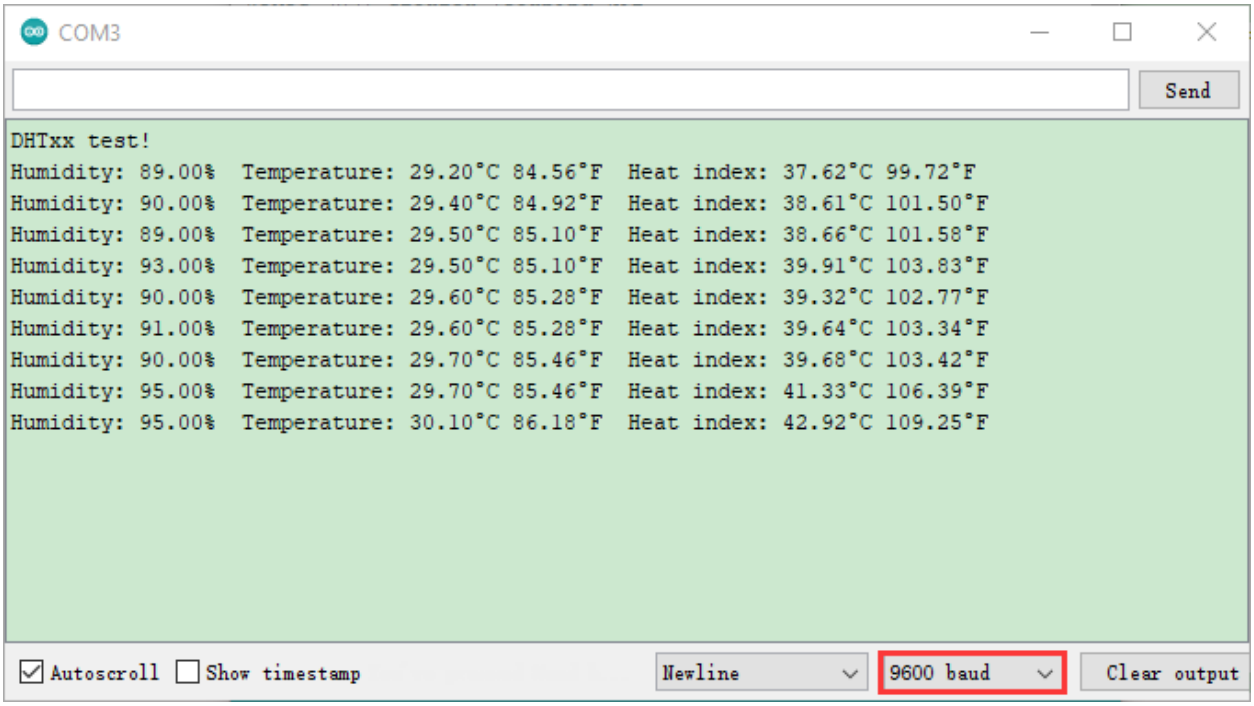
```
DHT dht(DHTPIN, DHT11);

void setup() {

Serial.begin(9600);

Serial.println(F("DHTxx test!"));

dht.begin();

}

void loop() {

// wait for a few seconds between two measurment

delay(2000);

// It takes about 250 milliseconds to read the temperature or humidity!

//

float h = dht.readHumidity();

// the temperature is Celsius (default value)

float t = dht.readTemperature();

// Calculate the Fahrenheit temperature (isFahrenheit = true)

float f = dht.readTemperature(true);

// fail to read or not, exit(try again).

if (isnan(h) \|\| isnan(t) \|\| isnan(f)) {

Serial.println(F("Failed to read from DHT sensor!"));

return;

}

// Calculate the Fahrenheit temperature index

float hif = dht.computeHeatIndex(f, h);

// (isFahreheit = false)

float hic = dht.computeHeatIndex(t, h, false);

Serial.print(F("Humidity: "));
```

(continues on next page)

(continued from previous page)

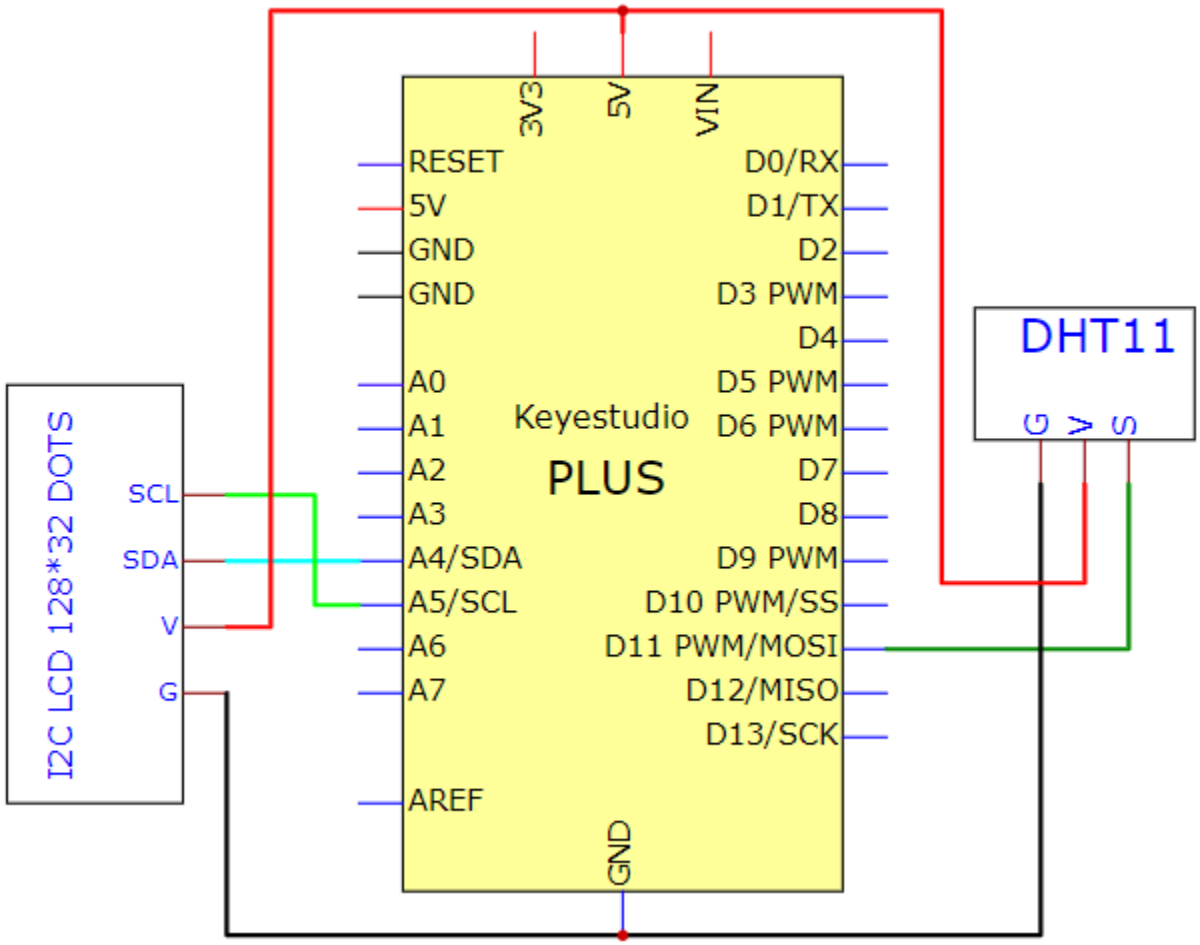
```
Serial.print(h);

Serial.print(F("% Temperature: "));

Serial.print(t);

Serial.print(F("°C "));

Serial.print(f);

Serial.print(F("°F Heat index: "));

Serial.print(hic);

Serial.print(F("°C "));

Serial.print(hif);

Serial.println(F("°F"));

}
```

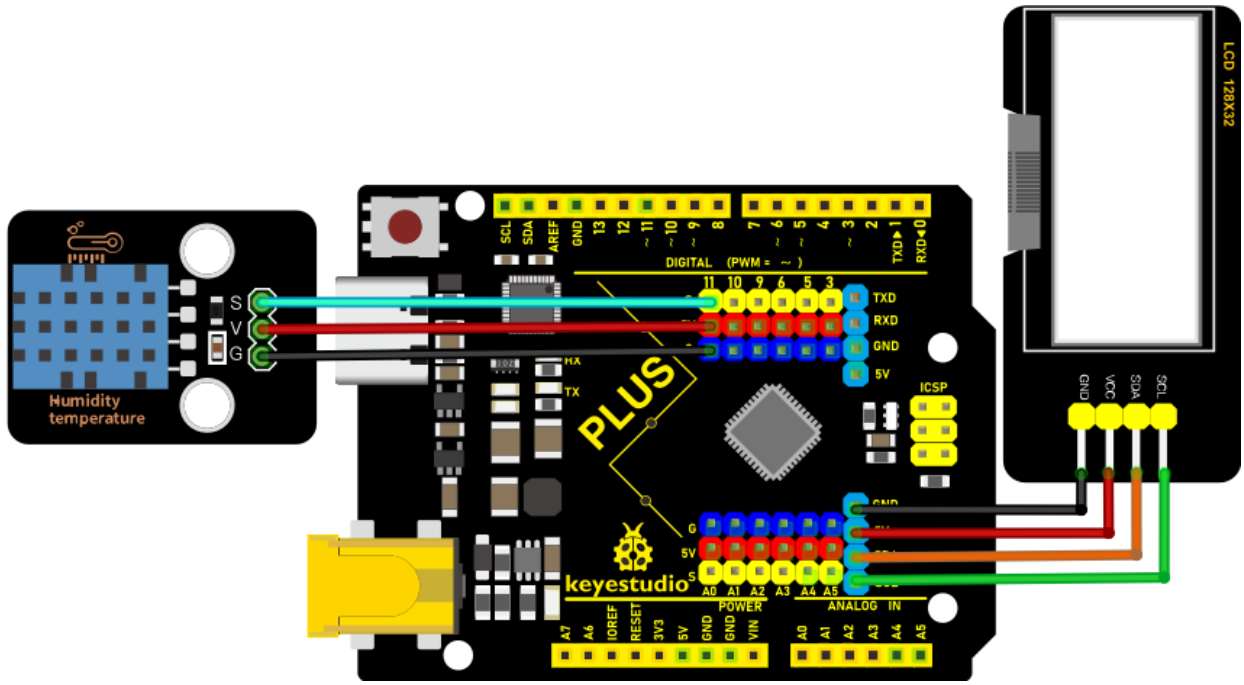
Upload the code to the Plus Mainboard, connect the wires and power on first. Then open the serial monitor, set the baud rate to 9600, you will see the current temperature and humidity value detected by the XHT11 sensor.



Circuit Diagram and Wiring Diagram

Now we start printing the value of the XHT11 temperature and humidity sensor with LCD screen. We will see the corresponding values on the LCD screen. Let's get started with this project. Please follow the wiring diagram below.


3V3
5V
VIN
RESET
5V
GND
GND
A0
A1
A2
A3
A4/SDA
A5/SCL
A6
A7
AREF
GND
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
Keyestudio
PLUS
I2C LCD 128*32 DOTS
SCL
SDA
V
G
DHT11
G V S


Humidity
temperature
S
V
G
PLUS
keyestudio
LCD 128X32
GND
VCC
SDA
SCL

Code

NoteThe library file needs to be installed in the code.If library files such as "**Dht11**"and "**lcd**" has been added, ignore the process of adding the library file below.

Decompress the library files in the folder, that is, put the decompressed "**Dht11**" and "**LCD_128X32**" folder into "\Arduino\libraries" under the compiler installation directory.

After successful placement, you need to restart the compiler, otherwise the compilation will not work.

e.g.C:\Program Files\Arduino\libraries

```
/*

Keyestudio 2021 starter learning kit

Project 29.2

Temperature_and_humidity_meter

http//www.keyestudio.com

*/

\#include "DHT.h"

\#define DHTPIN 11 // the pin connected the DHT sensor

DHT dht(DHTPIN, DHT11);

\#include \<lcd.h\> //add library files

lcd Lcd; //define a Lcd class instance

void setup() {

Lcd.Init(); //initialize

Lcd.Clear(); //clear

dht.begin();

}

char string[10];

//lcd displays humidity and temperature values

void loop() {

Lcd.Cursor(0,0);

Lcd.Display("Humidity:");

Lcd.Cursor(0,9);

Lcd.DisplayNum(dht.readHumidity());
```

(continues on next page)

(continued from previous page)

```
Lcd.Cursor(0,12);

Lcd.Display("%RH");

Lcd.Cursor(2,0);

Lcd.Display("Temperature:");

Lcd.Cursor(2,12);

Lcd.DisplayNum(dht.readTemperature());

Lcd.Cursor(2,15);

Lcd.Display("C");

delay(200);

}
```

Result

Upload the code to the Plus Mainboard, connect the wires and power on first. The LCD_128X32_DOT displays temperature and humidity in the current environment. We can use it as a real-time environmental monitoring tool.

5.30 Project 30: WiFi Test

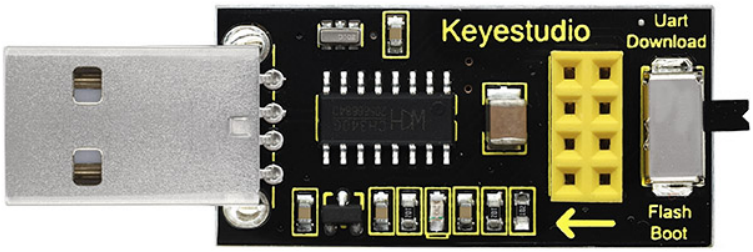
Introduction

ESP8266 serial WiFi ESP-01 module is an ultra-low-power UART-WiFi transparent transmission module and designed for mobile devices and IoT applications. The physical device of the user can be connected to Wi-Fi wireless network for Internet or LAN communication to realize networking functions.

Components Required

	
ESP8266 Serial WiFi ESP-01 Modulex1	USB to ESP-01S WiFi Module Serial Shieldx1

Component Knowledge



USB to ESP-01S WiFi module serial shield:

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

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

ESP8266 serial WiFi ESP-01: ESP8266 serial WiFi ESP-01 is an ultra-low-power UART-WiFi transparent trans-

mission module. It can be widely used in smart grids, intelligent transportation, smart furniture, handheld devices, industrial control and other fields.

Features

Supports wireless 802.11 b/g/n standards

Supports STA/AP/STA+AP three modes of operation

Built-in TCP/IP protocol stack to support multi-channel TCP Client connections

Supports many Socket AT commands

Supports UART / GPIO data communication interface

Supports Smart Link smart networking function

Supports remote firmware upgrades(OTA)

Built-in 32-bit MCU, can also be used as an application processor

Ultra-low-power and highly integrated Wi-Fi chip for battery-powered applications

Working temperature range: -40 ° C to + 125 ° C

3.3V single power supply

Parameters

Module	Type	ESP8266-01
	Main chip	ESP8266
Wireless parameters	**Wireless standard**	IEEE 802.11b/g/n
	Frequency range	2.412GHz-2.484GHz
	Transmit power	802.11b: +16 +/-2dBm (@11Mbps)
		802.11g: +14 +/-2dBm (@54Mbps)
		802.11n: +13 +/-2dBm (@HT20, MCS7)
	Receiving sensitivity	802.11b: -93 dBm (@11Mbps ,CCK)
		802.11g: -85dBm (@54Mbps, OFDM)
		802.11n: -82dBm (@HT20, MCS7)
	Antenna type	external stamp-hole interfaces
		external I-PEX connector and SMA connector
		Built-in onboard PCB antenna
Hardware parameters	**Hardware interfaces**	UARTIICPWMGPIOADC
	Operating voltage	3.3V
	GPIO drive capability	Max15ma
	Working current	Keep sending down=> Average~70mA Peak: 200mA Normal mode=> Average: ~12mA Peak: 200mA standby mode<200uA
	Operating temperature	-40℃~125℃
	Storage environment	Temperature<40℃ Relative humidity<90%R.H
	Size	Onboard PCB antenna14.3mmx24.8mmx1mm
Serial transparent transmission	**Transmission rate**	110-921600bps
	TCP Client	5
Software Parameters	**Wireless network types**	STA/AP/STA+AP
	Security mechanisms	WEP/WPA-PSK/WPA2-PSK
	Encryption types	WEP64/WEP128/TKIP/AES
	Firmware upgrade	Local serial port, OTA remote upgrade
	Network protocols	IPv4, TCP/UDP/FTP/HTTP
	User configuration	AT + instruction set, web page Android / iOS terminal, Smart Link intelligent configuration APP

About the Hardware

ESP8266 has many hardware interfaces, supporting UART, IIC, PWM, GPIO, ADC, etc., and suitable for a variety of IoT applications.

PIN	Function	Description
1	URXD	UART_RXD, receive General Purpose Input/OutputGPIO3
2	UTXD	UART_TXD, send 2General Purpose Input/Ou tput: GPIO1 3Do not pull down when power on
5	RESET-GPIO 16	External Reset signal, LOW reset, HIGH works(default is HIGH)
6	GND	GND
8	VCC	3.3V, power the module
9	ANT	WiFi Antenna
11	GPIO0	WiFi Status(Default)WiFi status indicator control signal Working mode selection: Suspend-Flash Bootworking mode Pull downUART Downloaddownload mode
12	ADC	ADC, input range: 0V-1V
13	GPIO15	Pull downwork mode
14	CH_PD	Working at HIGH level Power off at LOW level
15	GPIO2	It must be HIGH level when power on, do not pull down the hardware Internal is pulled up(default)

Modes	Min	Typ	Max	Unit
Tx 802.11b, CCK 1Mbps, Pout=+19.5dBm		215		mA
Tx 802.11b, CCK 11Mbps, Pout=+18.5dBm		197		mA
Tx 802.11g, OFDM54 Mbps, Pout=+16dBm		145		mA
Tx 802.11n, MCS7, Pout=+14dBm		135		mA
Rx 802.11b, 1024 bytes packet length, -80dBm		100		mA
Rx 802.11g, 1024 bytes packet length, -70dBm		100		mA
Rx 802.11n, 1024 bytes packet length, -65dBm		102		mA
Standby Mode		70		mA
Power Off		0.5		A

Power consumption

The above power consumption data is based on a 3.3V power supply at 25°ambient temperature.

1. All measurements are completed at the antenna interface.
2. All transmitted data is based on 90% duty cycle, which is measured in a continuous launch mode.
3. **Radio characteristic**

The following data were measured when the voltage is 3.3V at room temperature.

Description	Min	Typ	Max	Unit
Input frequency	2412		2484	MHz
Input resistance		50		
Input reflection			-10	dB
PA output power at 72.2 Mbps	14	15	16	dBm
PA output power in 802.11b mode	17.5	18.5	19.5	dBm
Sensitivity				
CCK 1Mbps		-98		dBm
CCK 11Mbps		-91		dBm
6Mbps(1/2BPSK)		-93		dBm
54Mbps(3/4 64-QAM)		-75		dBm
HT20MCS765Mbps72.2Mbps		-71		dBm
Adjacent channel rejection				
OFDM6Mbps		37		dB
OFDM54Mbps		21		dB
HT20MCS0		37		dB
HT20MCS7		20		dB

Note: 1. 72.2Mbps is measured in 802.11n mode with MCS equal to 7 and GI equal to 200uS.

Up to +19.5dBm output power in 802.11b mode.

Functions

A. Main functions

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

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

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

GPIO control: Control switch, relay, etc.

Working modes

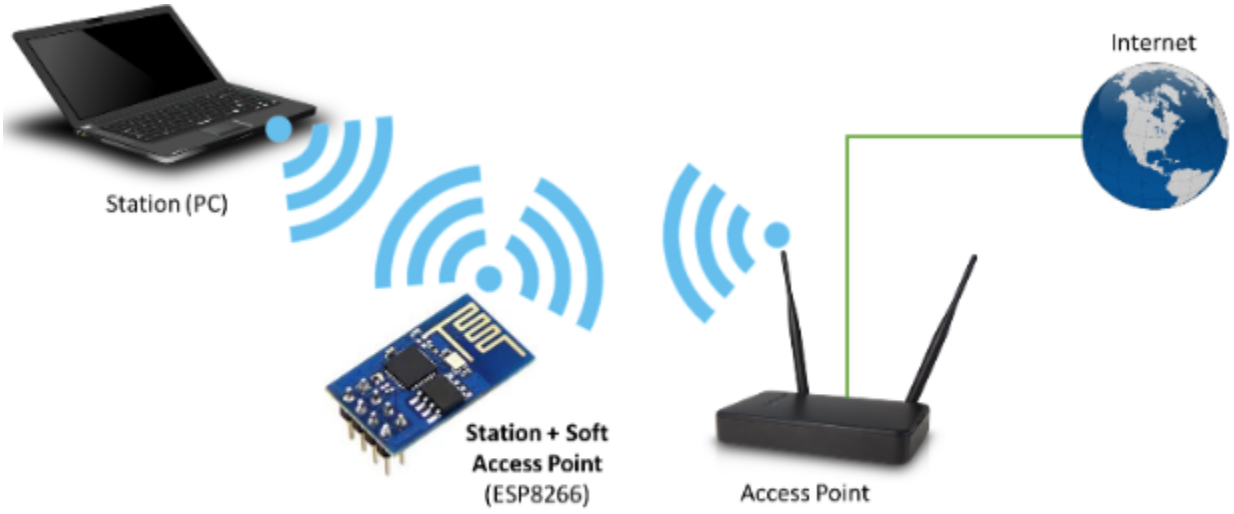
The ESP8266 module supports three operating modes, STA/AP/STA+AP.

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

ESP8266 operating in the **Station** mode

AP mode: The ESP8266 module acts as a hotspot to enable communication directly with the mobile phone or computer to achieve wireless control of the local area network (LAN).

STA+AP mode: The coexistence mode of the above two modes, that is, can achieve the seamless switching through the Internet control, more convenient for operation.



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

Applications

Serial CH340 to Wi-Fi

Industrial transparent transmission DTU

Wi-Fi remote monitoring/control Toy field Color LED control

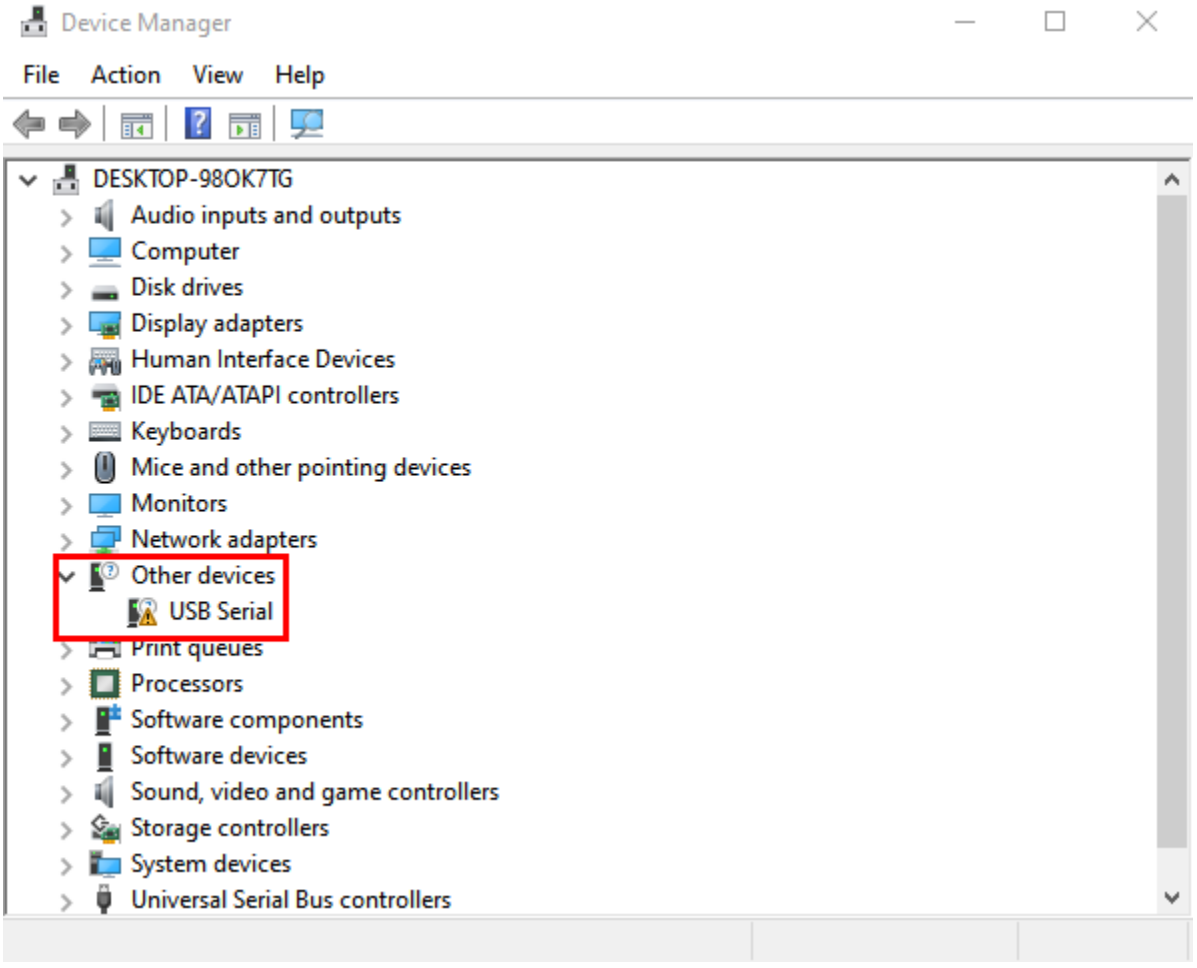
Integrated management of fire protection and security intelligence

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

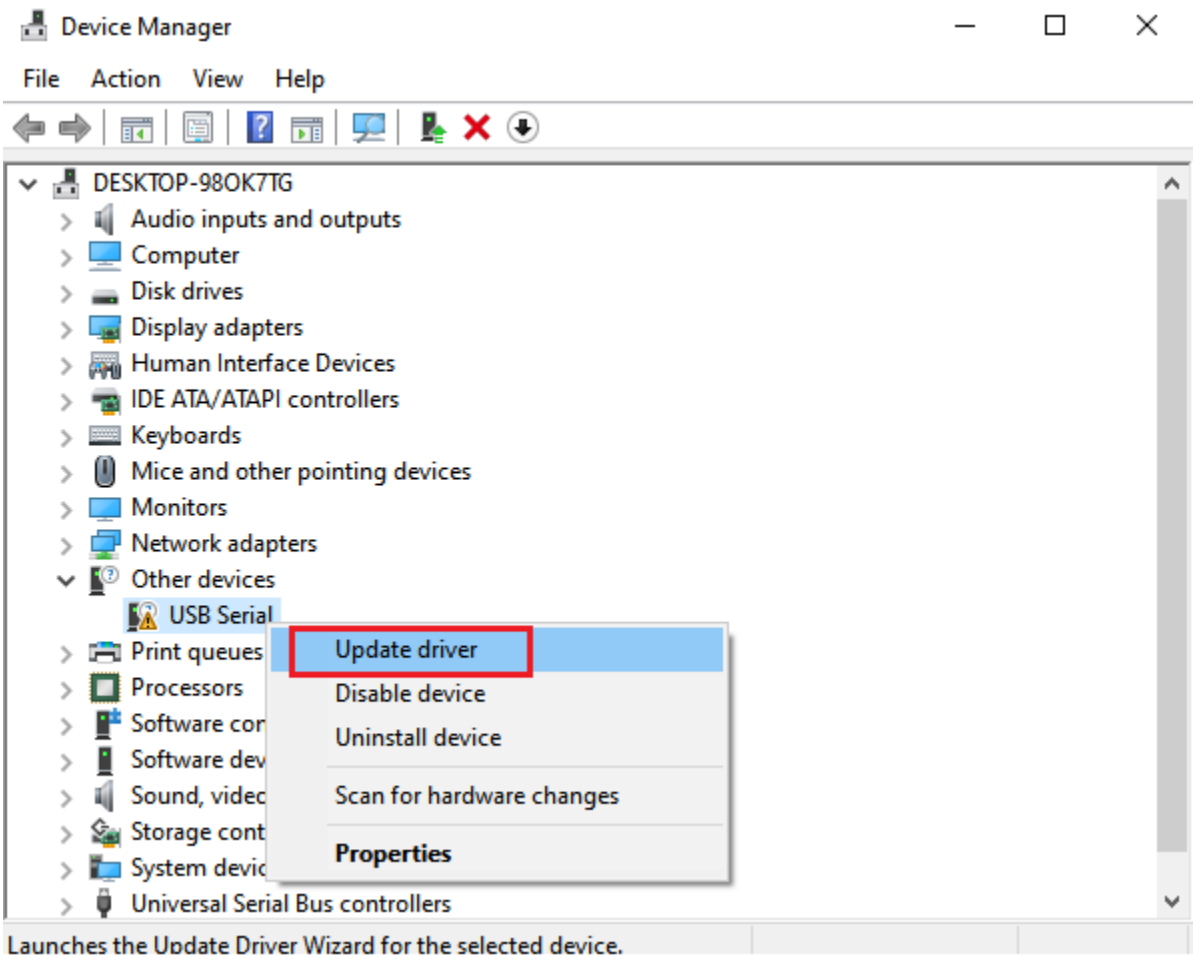
Install the Driver

The USB to serial port chip of this shield is CH340, We need to install the chip driver. The driver is usb_ch341_3.1.2009.06. We put this driver on the D: drive(i.e.: copy Driver File to D: drive). Then start installing the driver. The way to install drivers in different systems is pretty much the same, we will start installing drivers on the Windows 10.

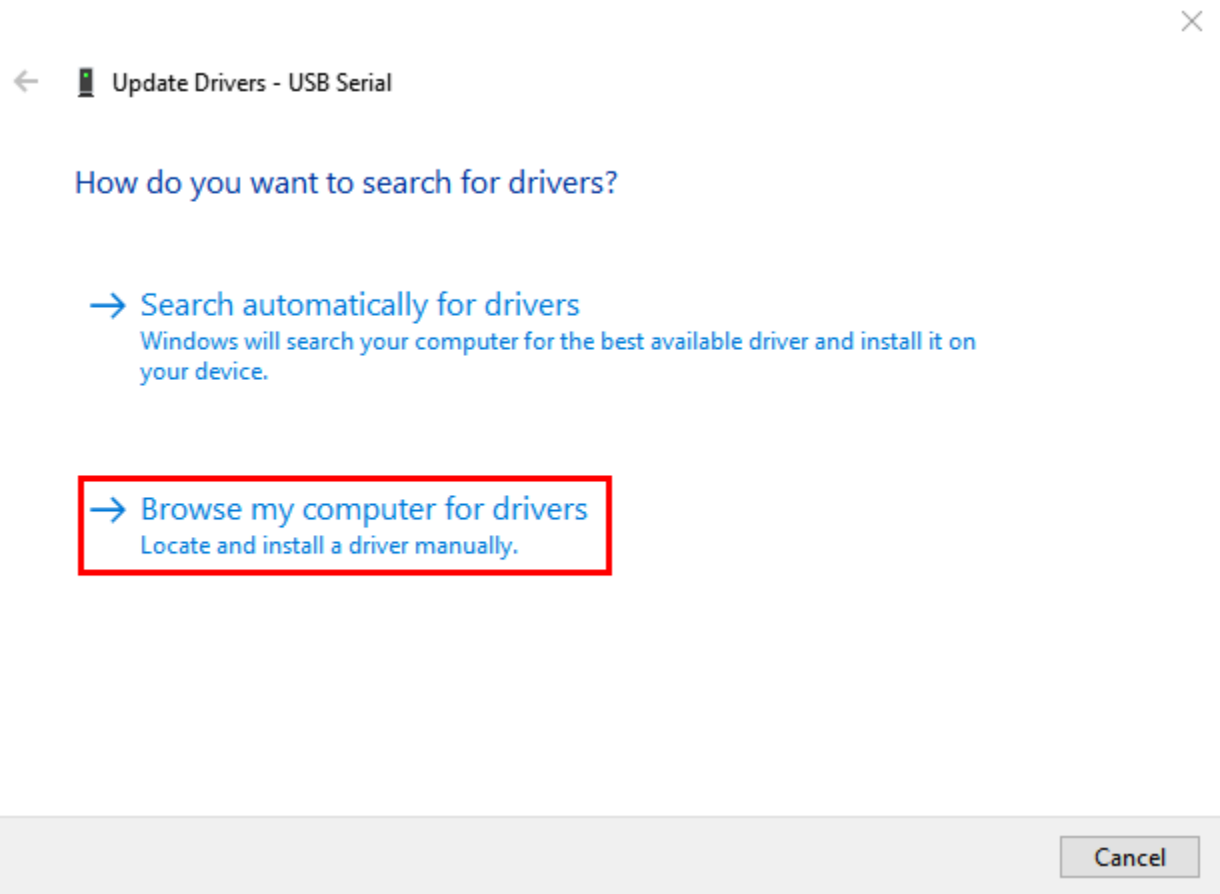
When you connect the shield to your computer at the first time, right click“Computer”—>“Properties”—>“Device manager”, you can see “**USB-Serial**”.



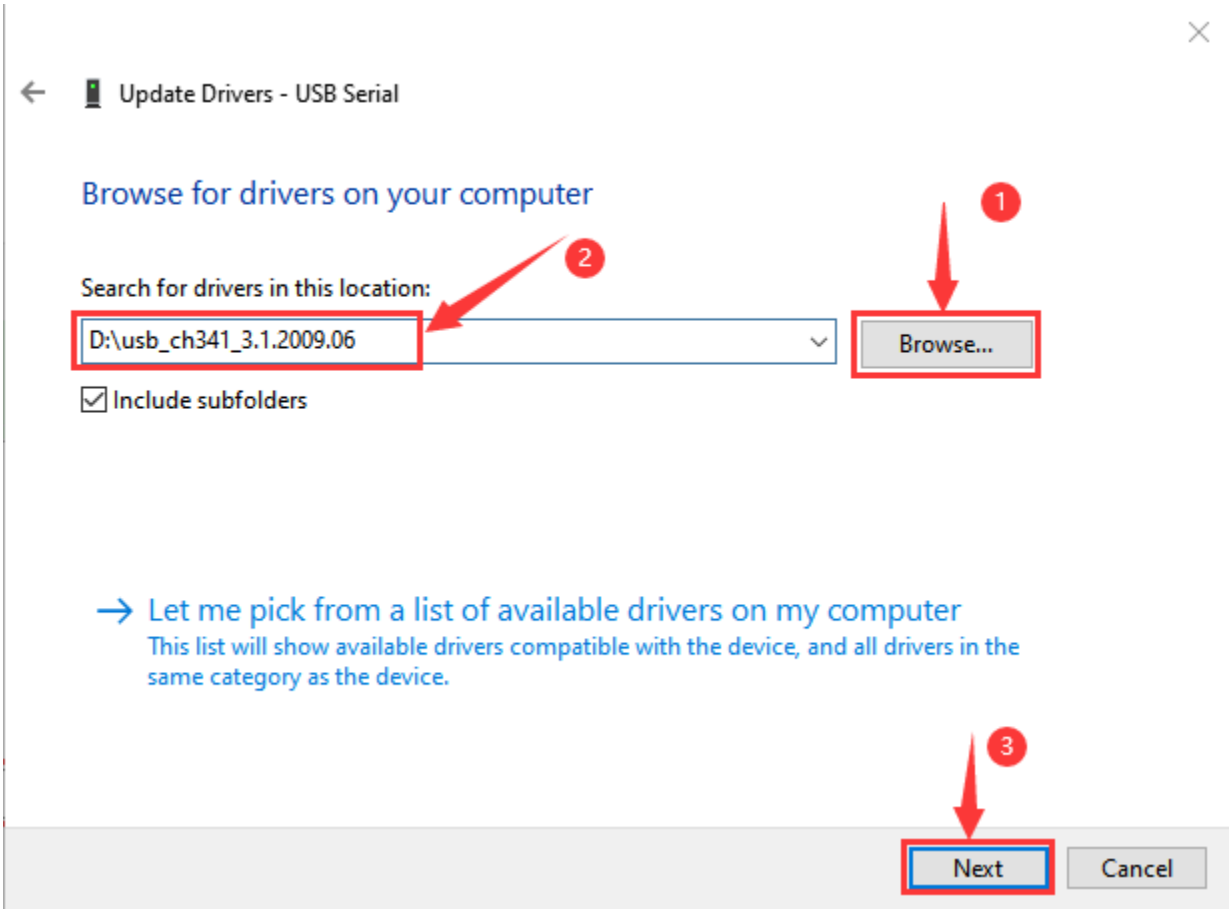
Click “**USB-Serial**”and select“**Update Driver** ”.



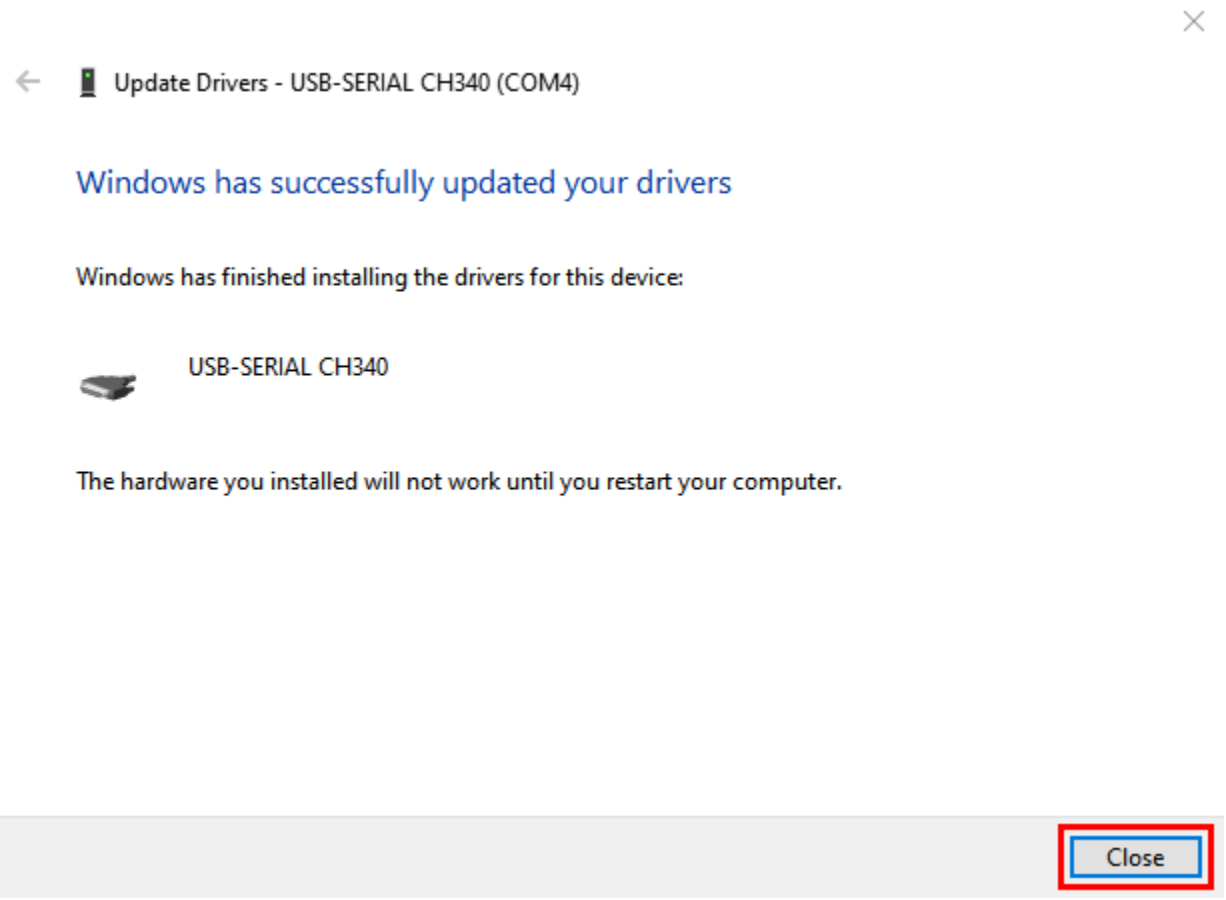
Then click on "**Browse my computer for drivers**".



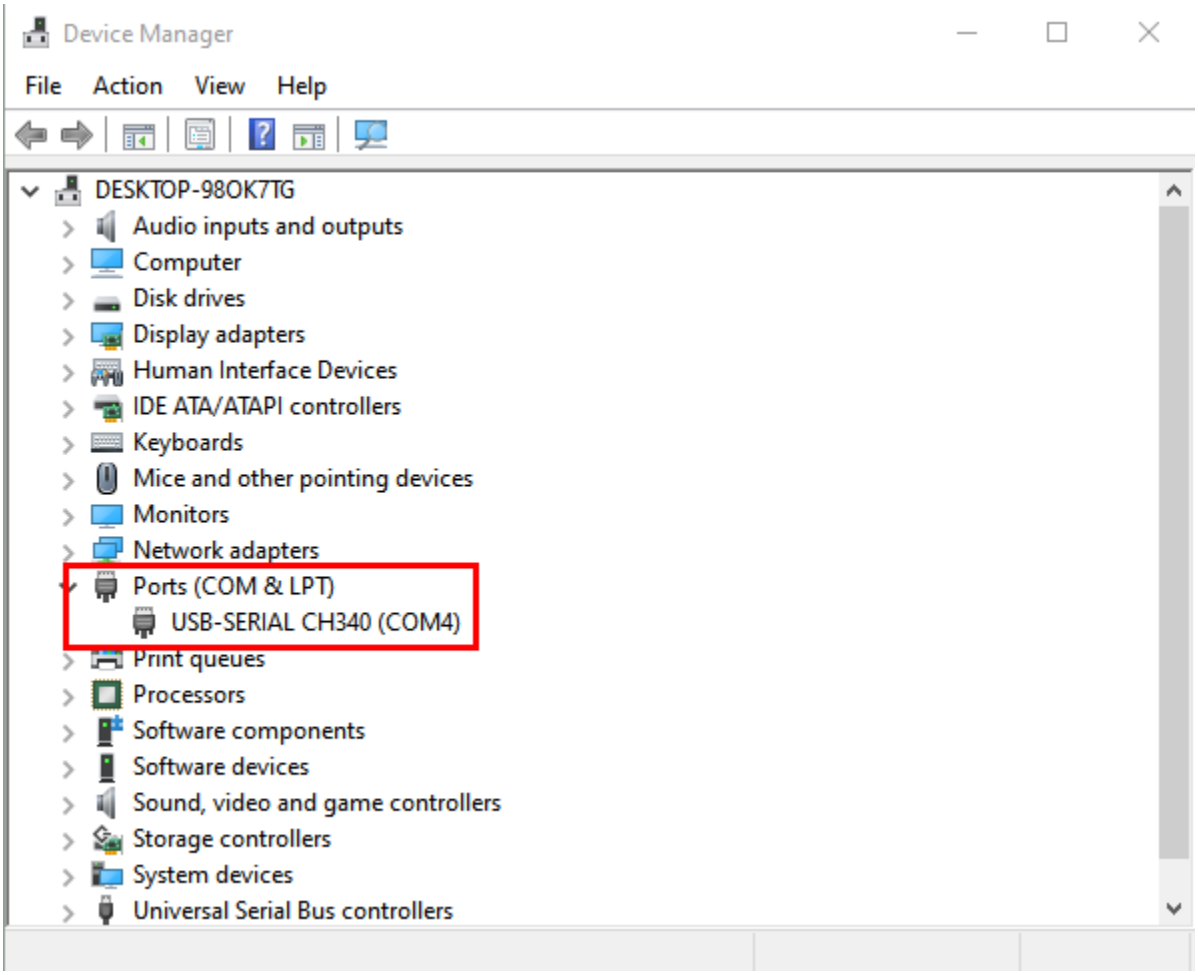
Find the "**Drive File**" folder provided.(Here I put the driver file USb_CH341_3.1.2009.06 on disk D)



Click "**Close**" when installation is complete.

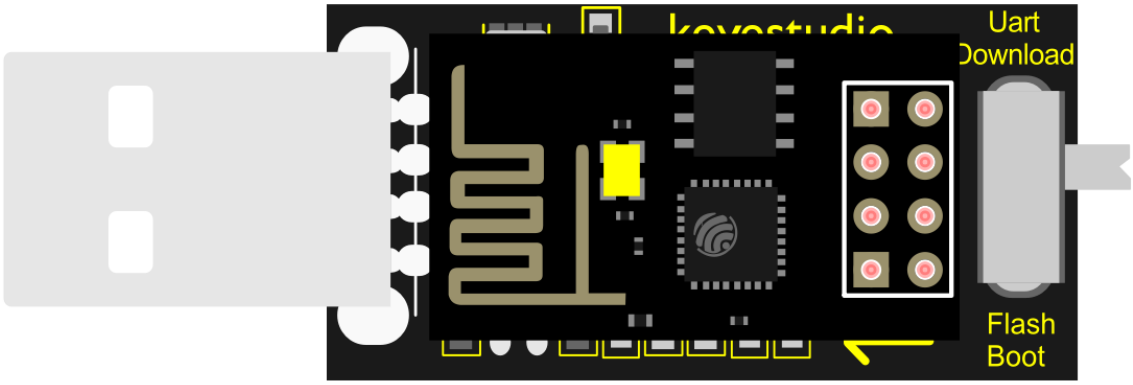


After the driver installation is complete, right click“Computer”—>“Properties”—>“Device manager”, you can see that the CH340 driver has been successfully installed on your computer, as follows.



Interface the Shield with the Computer

Insert the ESP8266 serial WiFi ESP-01 module in the correct orientation into the USB to ESP-01S WiFi module serial shield.



fritzing

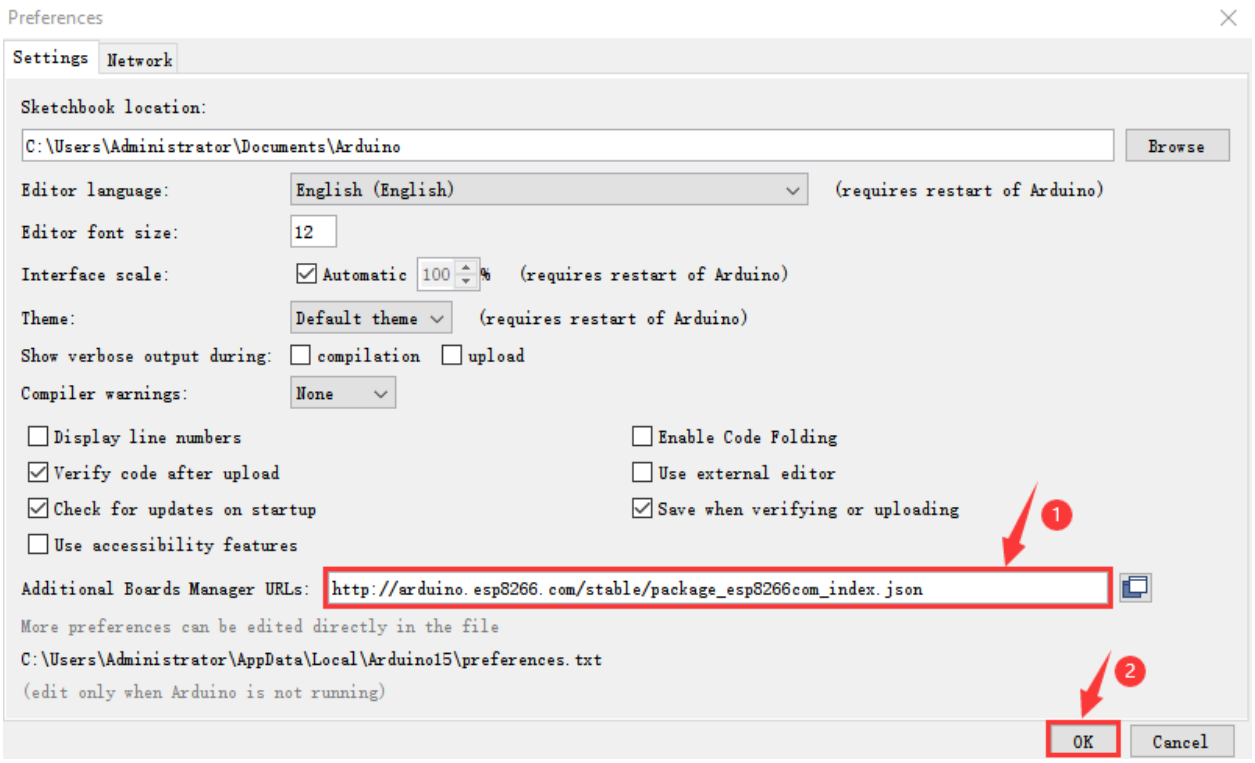
First, turn the DIP switch on the USB to ESP-01S WiFi module serial shield to the UartDownload, and then insert the shield into the USB port of the computer.

Set up the Development Environment

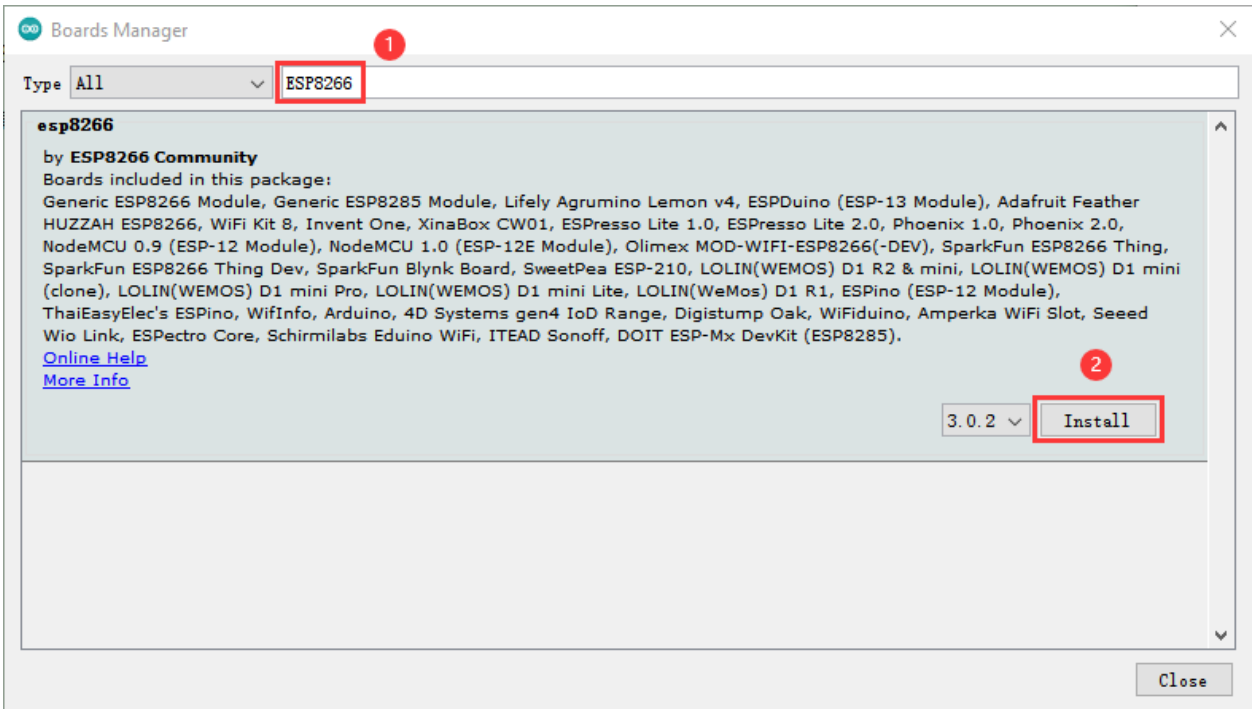
Insert the ESP8266 serial WiFi ESP-01 module into the USB to ESP-01S WiFi module serial shield correctly, and then plug the shield into the USB port of the computer. Click to enter the arduino-1.8.16 folder (you can also use the latest version). Find arduino.exe and click to enter the 1.8.16 version of the IDE interface.

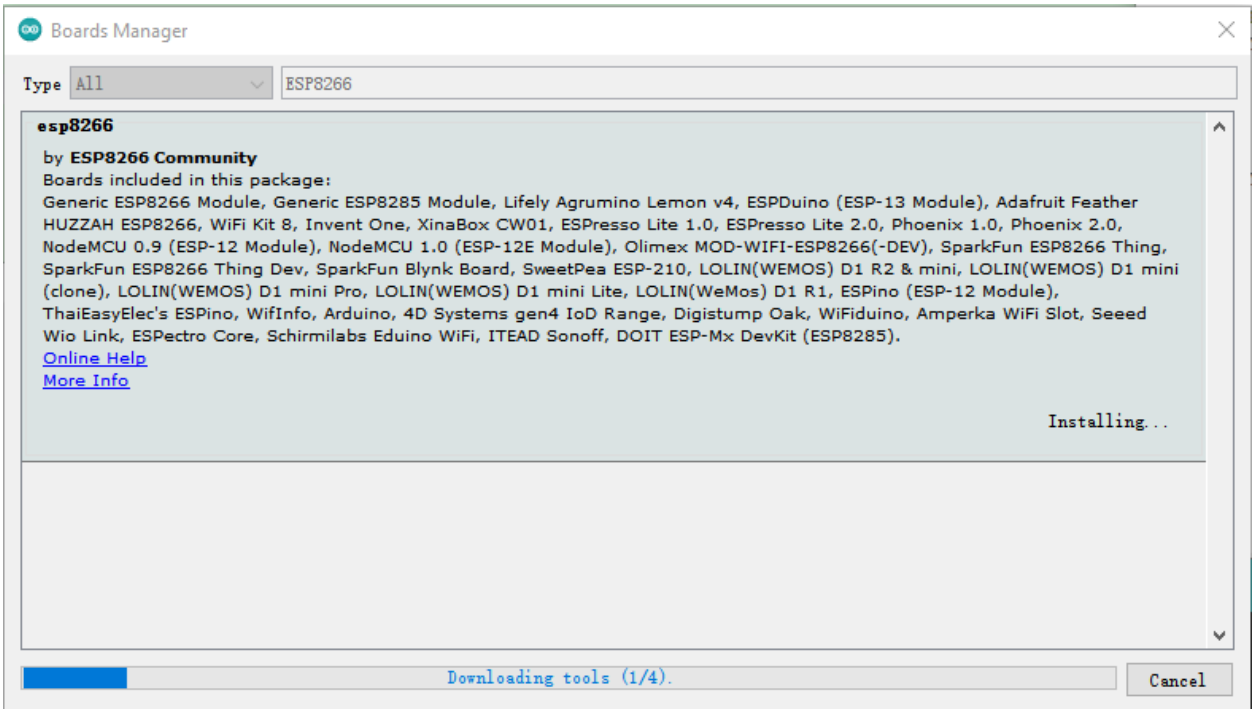
Download and install from the Arduino IDE

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

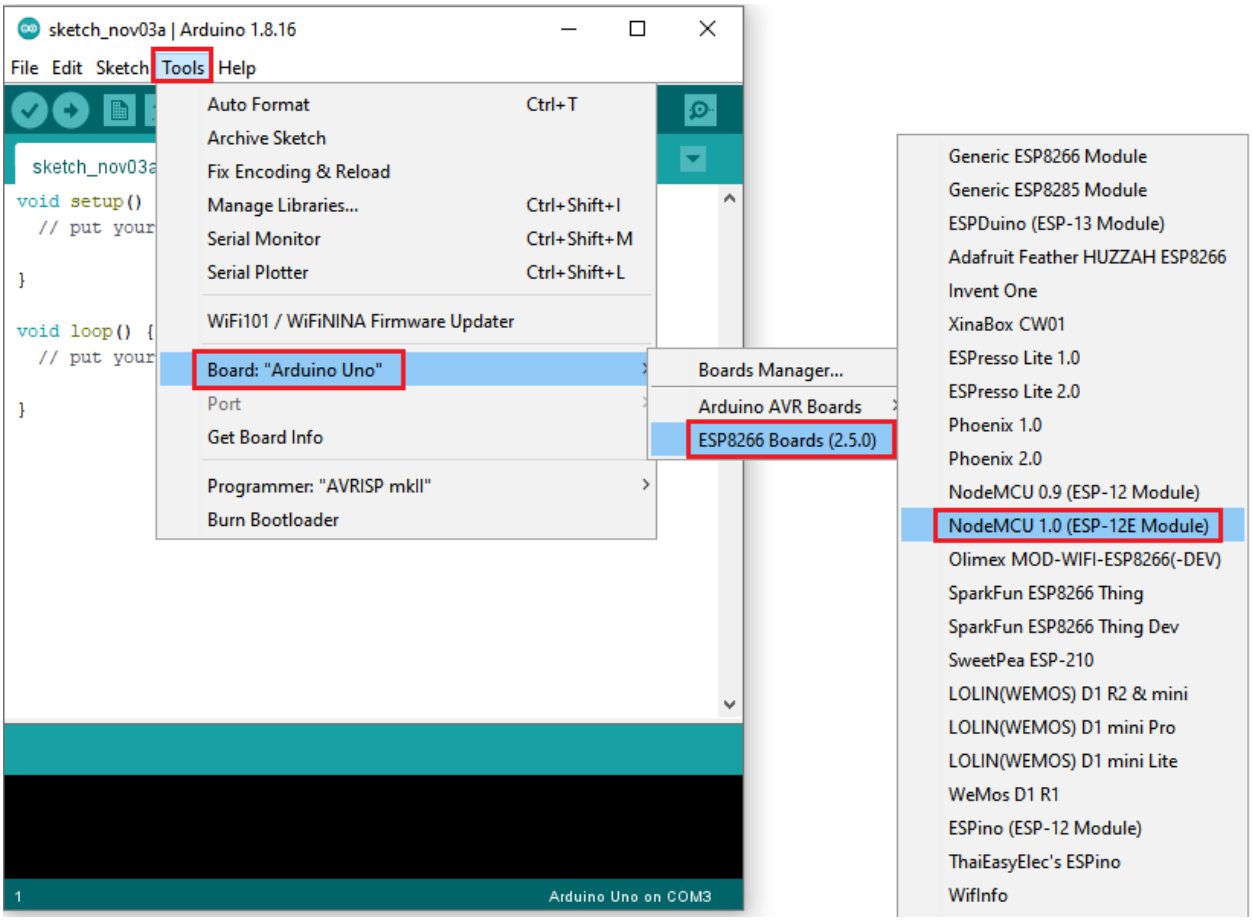


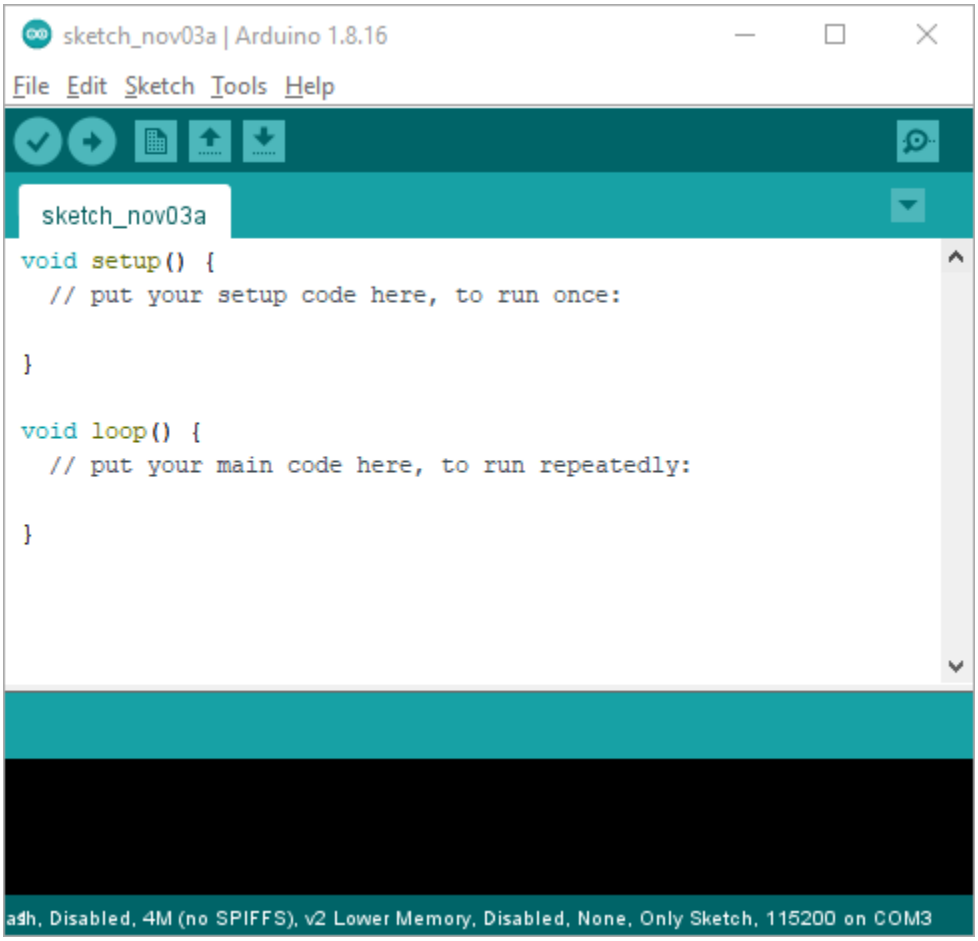
Click"**Tools**"→"**Board:**", then click on "**Board Manager…**" to enter the "**Board Manager**" page, type "ESP8266" in the space after "All". Then click the following search content, select the latest version to install. The installation package is not large, click "**Install**" to start to install the relevant plug-ins. (There may be an error in downloading and installing, possibly due to the server, so you need to click "Install" again. However, due to network reasons, most users may not be able to search esp8266 by esp8266 Community, so this method is not recommended for beginners, and **the following method 2 is recommended**.)

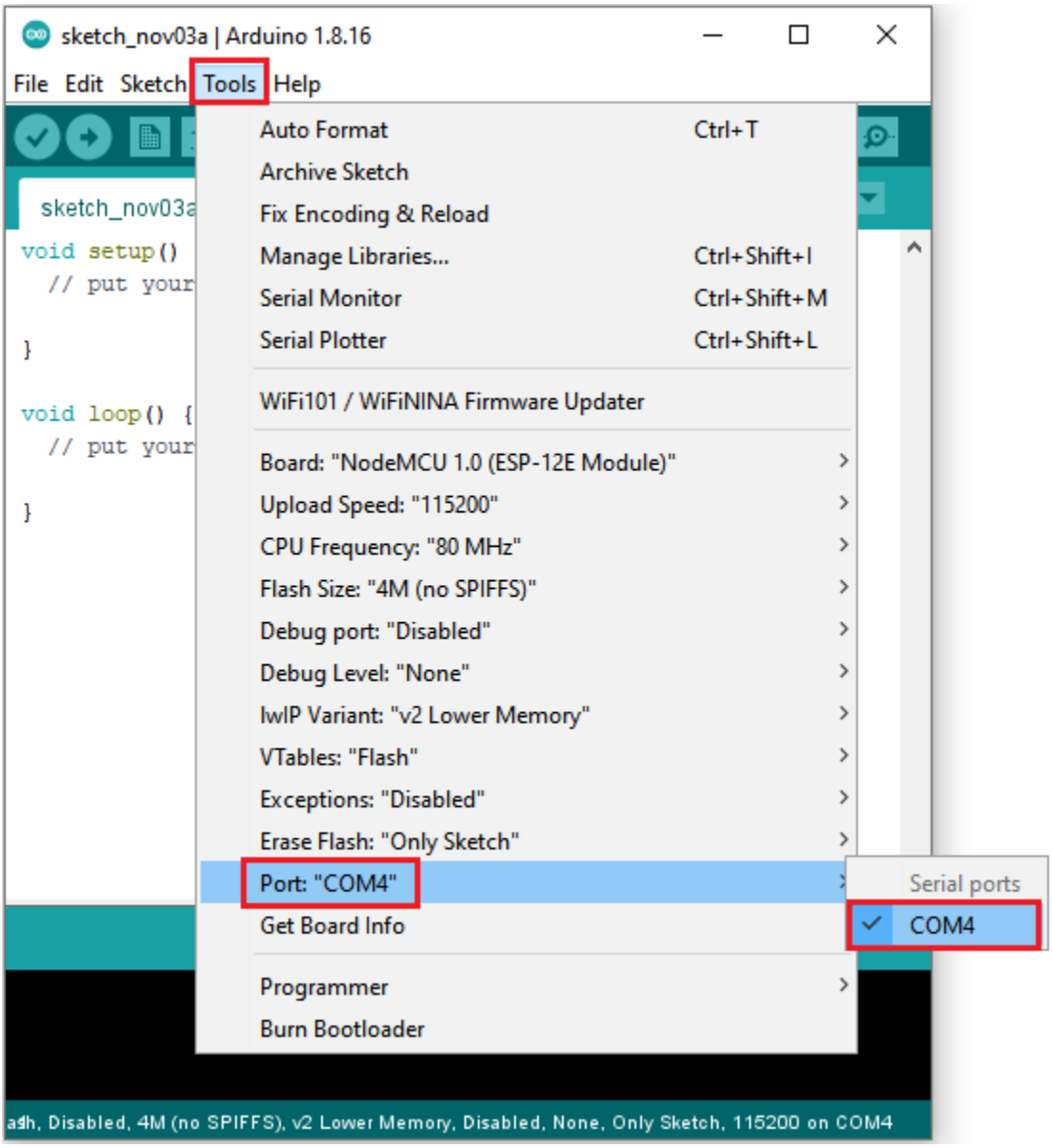




After successful installation, Click"**Close**"to close the page, and then click"**Tools**"→"**Board:**", you can view different models of ESP8266 development boards in it. Select the corresponding ESP8266 development board model and COM port to program ESP8266.

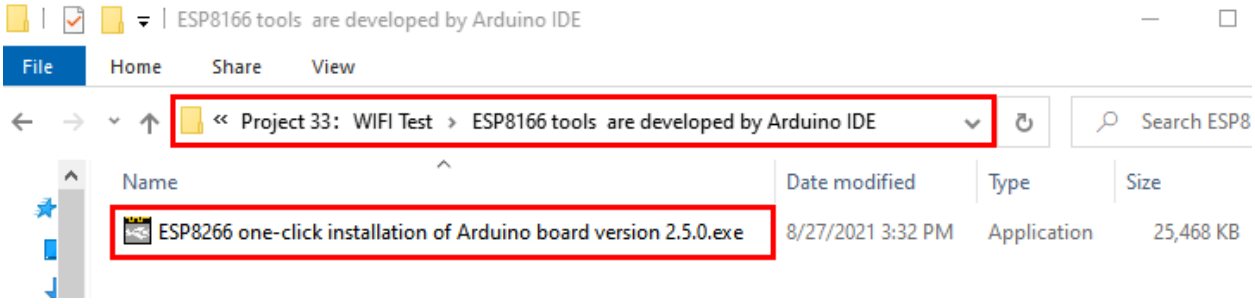

sketch_nov03a | Arduino 1.8.16
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP8266 Boards (2.5.0)
Generic ESP8266 Module
Generic ESP8285 Module
ESPDuino (ESP-13 Module)
Adafruit Feather HUZZAH ESP8266
Invent One
XinaBox CW01
ESPresso Lite 1.0
ESPresso Lite 2.0
Phoenix 1.0
Phoenix 2.0
NodeMCU 0.9 (ESP-12 Module)
NodeMCU 1.0 (ESP-12E Module)
Olimex MOD-WIFI-ESP8266(-DEV)
SparkFun ESP8266 Thing
SparkFun ESP8266 Thing Dev
SweetPea ESP-210
LOLIN(WEMOS) D1 R2 & mini
LOLIN(WEMOS) D1 mini Pro
LOLIN(WEMOS) D1 mini Lite
WeMos D1 R1
ESPino (ESP-12 Module)
ThaiEasyElec's ESPino
WifInfo
Arduino Uno on COM3


sketch_nov03a | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_nov03a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
ash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

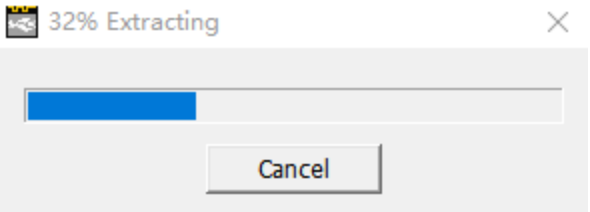


Installation of ESP8266 by tools (Recommended)

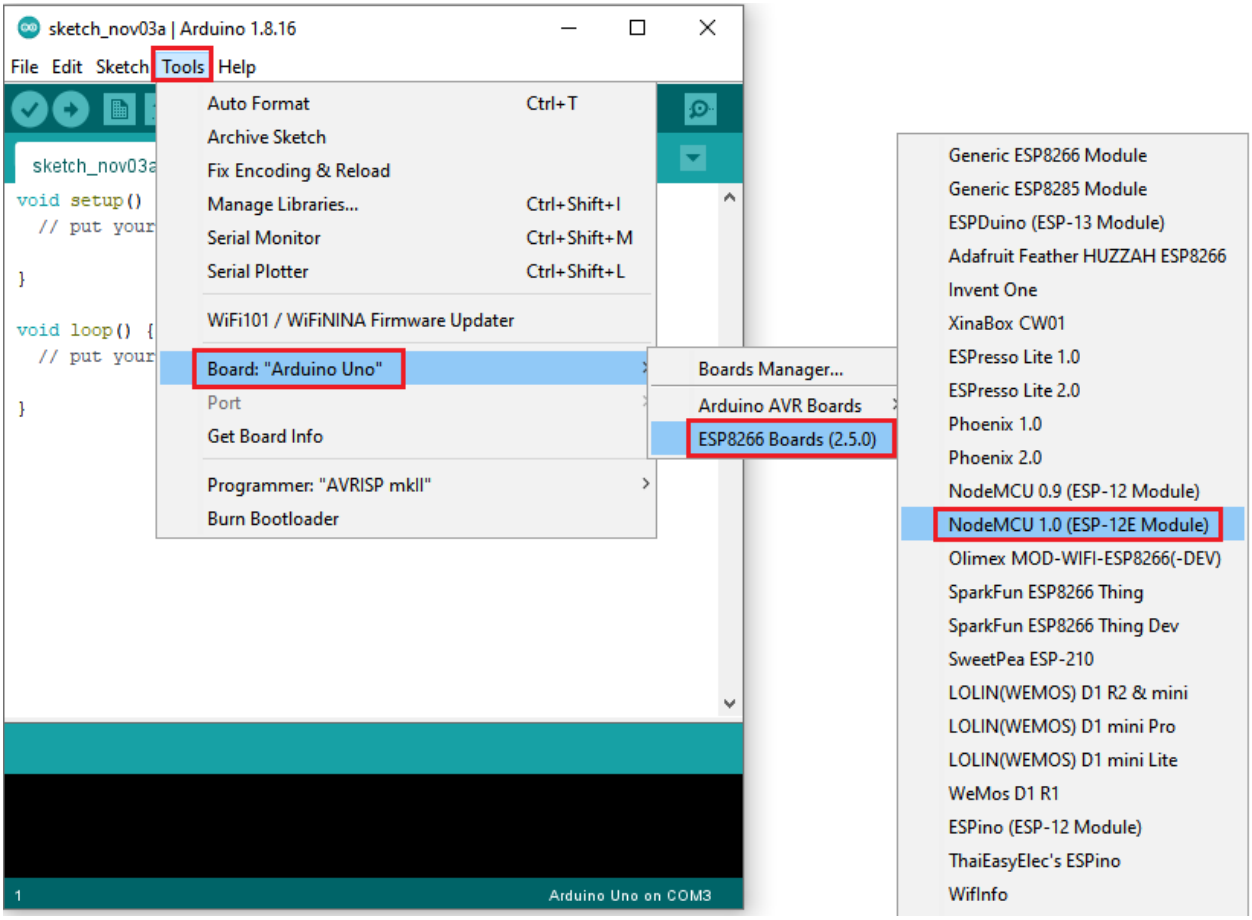
Use“ESP8266 one-click installation of Arduino board version 2.5.0.exe”to install ESP8266. This method is recommended because it is convenient and fast.

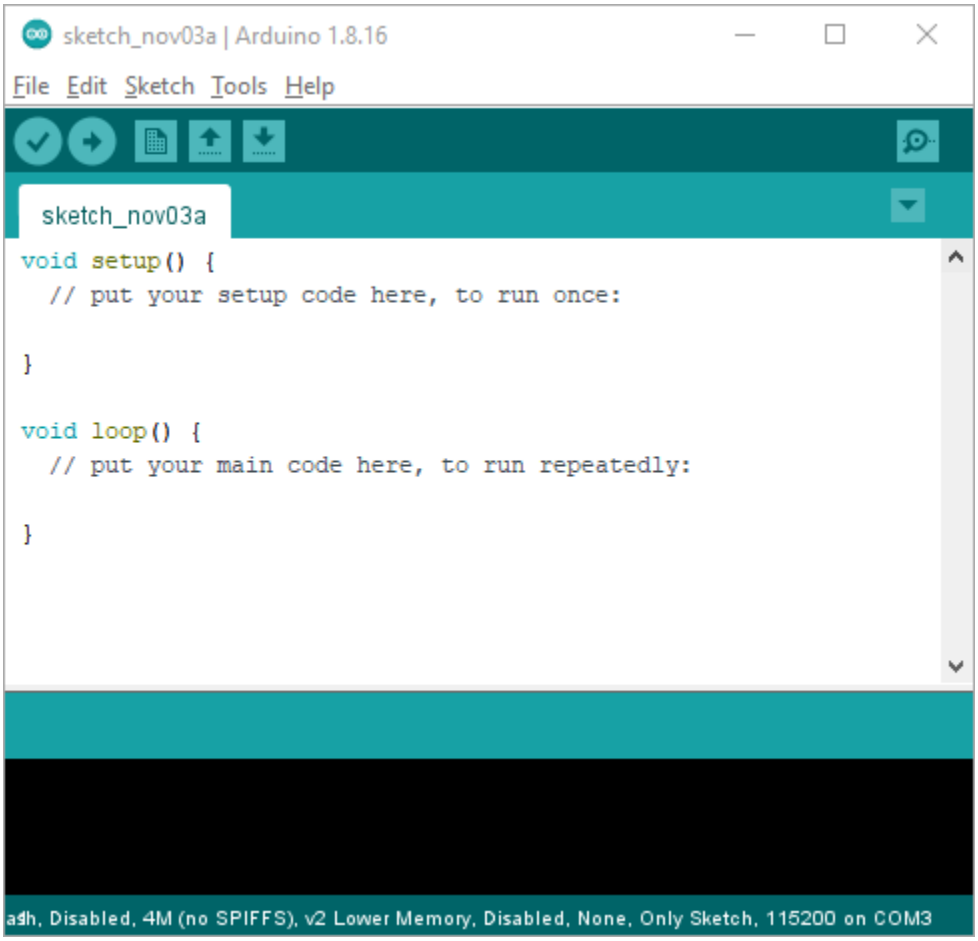


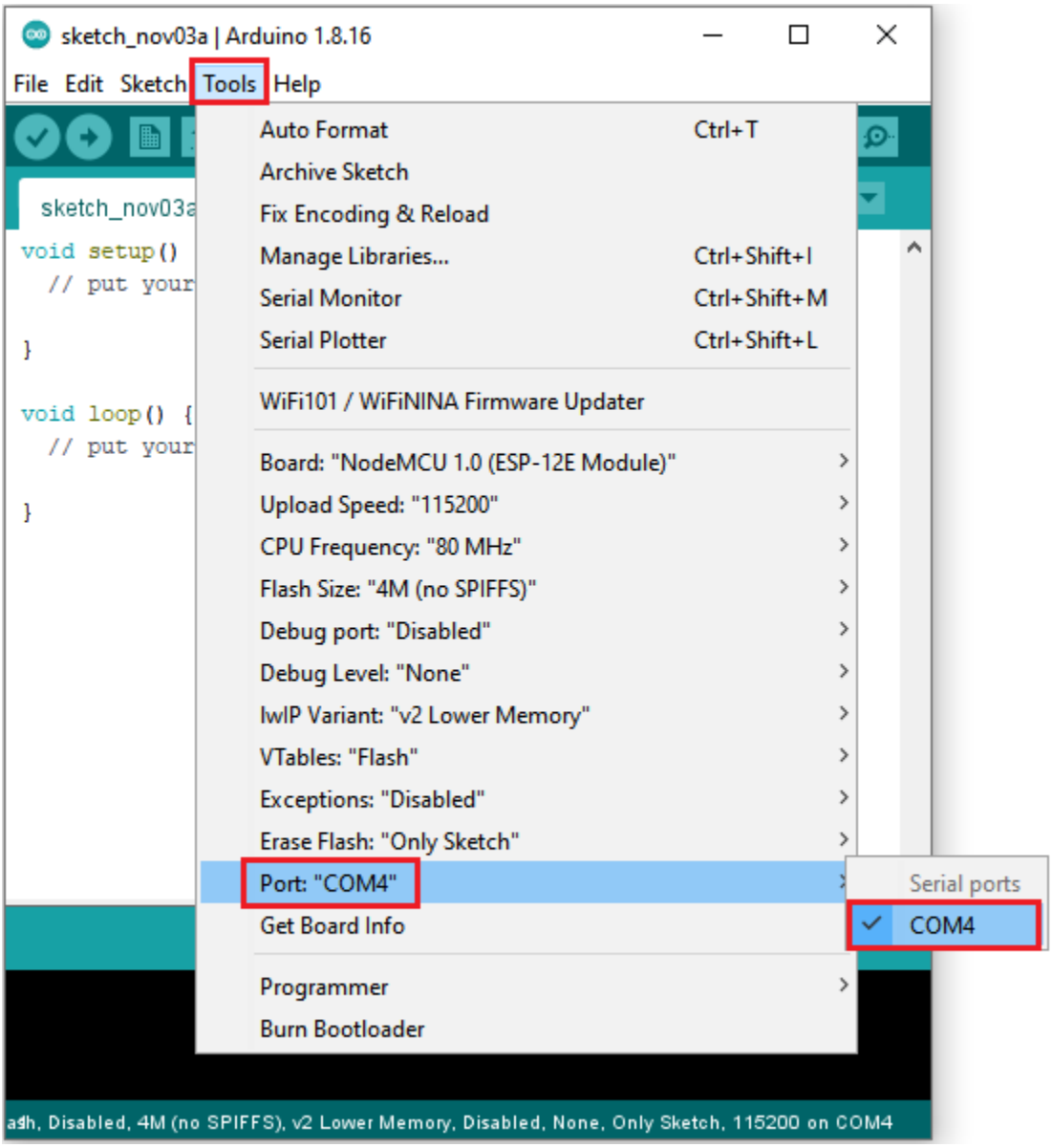
Double click“ESP8266 one-click installation of Arduino board version 2.5.0.exe”, then the installation is finished.



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




sketch_nov03a | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_nov03a
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
ash, Disabled, 4M (no SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3



WiFi Test Code

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

```
/*

Keyestudio 2021 starter learning kit

Project 30

WIFI test

http//www.keyestudio.com

*/
```

(continues on next page)

(continued from previous page)

```
\#include \<ESP8266WiFi.h\>

\#include \<ESP8266mDNS.h\>

\#include \<WiFiClient.h\>

\#ifndef STASSID

//\#define STASSID "your-ssid"

//\#define STAPSK "your-password"

\#define STASSID "ChinaNet-2.4G-0DF0" //the name of user's wifi

\#define STAPSK "ChinaNet@233" //the password of user's wifi

\#endif

const charx ssid = STASSID;

const charx password = STAPSK;

// TCP server at port 80 will response the HTTP requirement

WiFiServer server(80);

void setup(void) {

Serial.begin(115200);

// connect WiFi

WiFi.mode(WIFI_STA);

WiFi.begin(ssid, password);

Serial.println("");

// wait connection

while (WiFi.status() != WL_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.print("Connected to ");
```

(continues on next page)

(continued from previous page)

```
Serial.println(ssid);

Serial.print("IP address: ");

Serial.println(WiFi.localIP());

// set the mDNS responder::

// - in this example. the first parameter is domain name

// The fully qualified domain name is “esp8266.local”

// - the second parameter is IP address

// send the IP address via WiFi

if (!MDNS.begin("esp8266")) {

Serial.println("Error setting up MDNS responder!");

while (1) {

delay(1000);

}

}

Serial.println("mDNS responder started");

// activate TCP (HTTP) server

server.begin();

Serial.println("TCP server started");

// add the server to MDNS-SD

MDNS.addService("http", "tcp", 80);

}

void loop(void) {

MDNS.update();

// check the client side is connected or not

WiFiClient client = server.available();

if (!client) {
```

(continues on next page)

(continued from previous page)

```
return;

}

Serial.println("");

Serial.println("New client");

// wait the effective data from the client side

while (client.connected() && !client.available()) {

delay(1);

}

// read the first row of HTTP requirement

String req = client.readStringUntil('\\r');

// the first row of the HTTP requirement is shown below: "GET /path HTTP/1.1"

// Retrieve the "/path" part by finding the spaces

int addr_start = req.indexOf(' ');

int addr_end = req.indexOf(' ', addr_start + 1);

if (addr_start == -1 \|\| addr_end == -1) {

Serial.print("Invalid request: ");

Serial.println(req);

return;

}

req = req.substring(addr_start + 1, addr_end);

Serial.print("Request: ");

Serial.println(req);

client.flush();

String s;

if (req == "/") {

IPAddress ip = WiFi.localIP();
```

(continues on next page)

(continued from previous page)

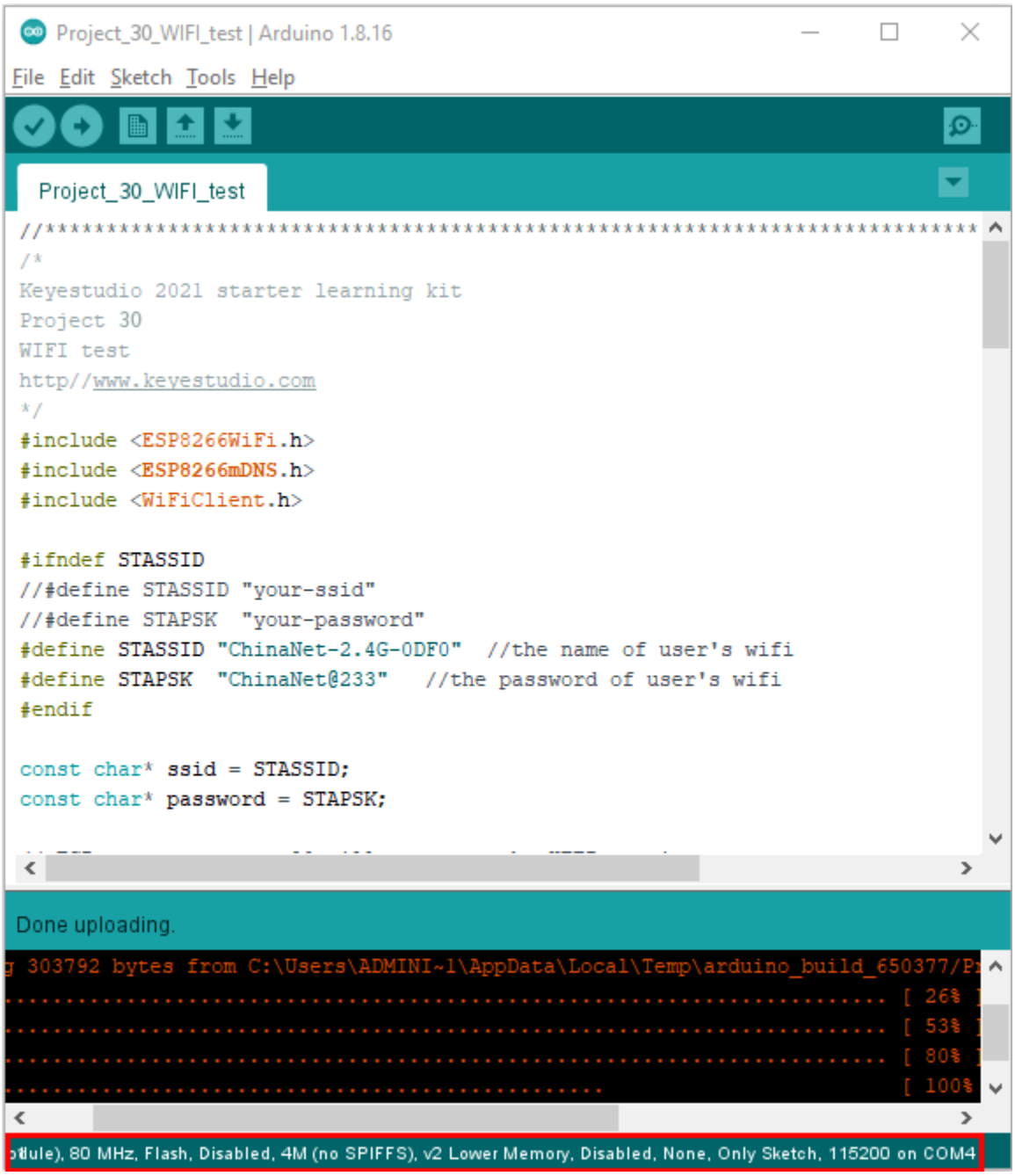
```
String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +
String(ip[3]);

s = "HTTP/1.1 200 OK\\r\\nContent-Type: text/html\\r\\n\\r\\n\<!DOCTYPE
HTML\>\\r\\n\<html\>Hello from ESP8266 at ";

s += ipStr;

s += "\</html\>\\r\\n\\r\\n";

Serial.println("Sending 200");

} else {

s = "HTTP/1.1 404 Not Found\\r\\n\\r\\n";

Serial.println("Sending 404");

}

client.print(s);

Serial.println("Done with client");

}
```

Result

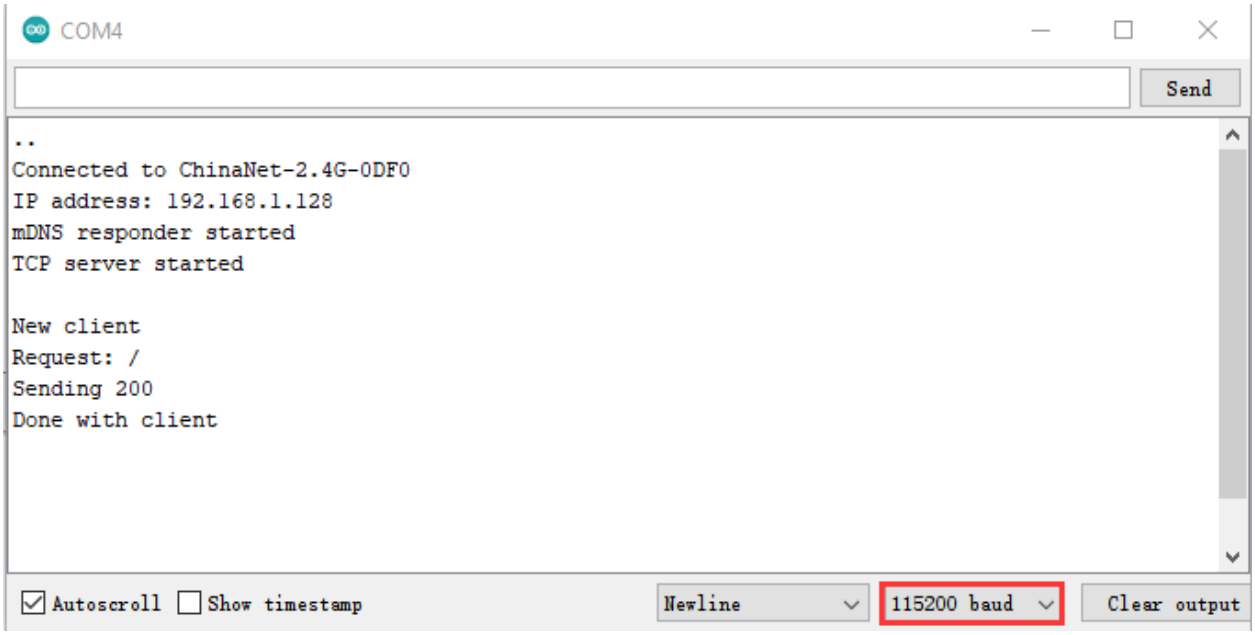
Note: You need to change the user WiFi name and user WiFi password in the project code to your own WiFi name and WiFi password.

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

After changing the WiFi name and WiFi password, ensure that the DIP switch on the shield has been turned to the UartDownload and the shield has been plugged into the USB port of the computer. Then set the board type and COM port according to the previous method, and the corresponding board type and COM port are displayed in the lower right corner of the IDE. Click  to upload the test code to the ESP8266 serial WiFi ESP-01 module, the upload is complete.Note: If the upload fails, unplug the shield and plug it into the computer's USB port again when the board type and COM port are OK.)



After the test code is uploaded successfully, first unplug the shield from the USB port of the computer, then turn DIP switch on the shield to the Flash Boot, and plugged into the USB port of the computer again. Open the serial monitor, set the baud rate to **115200**, and you can see your WiFi information, as shown below.

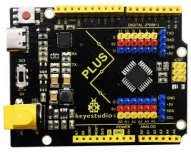
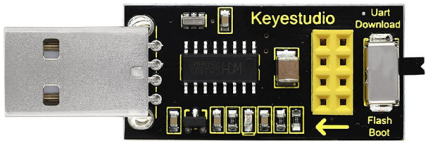


5.31 Project 31: Control LED With WiFi

Introduction

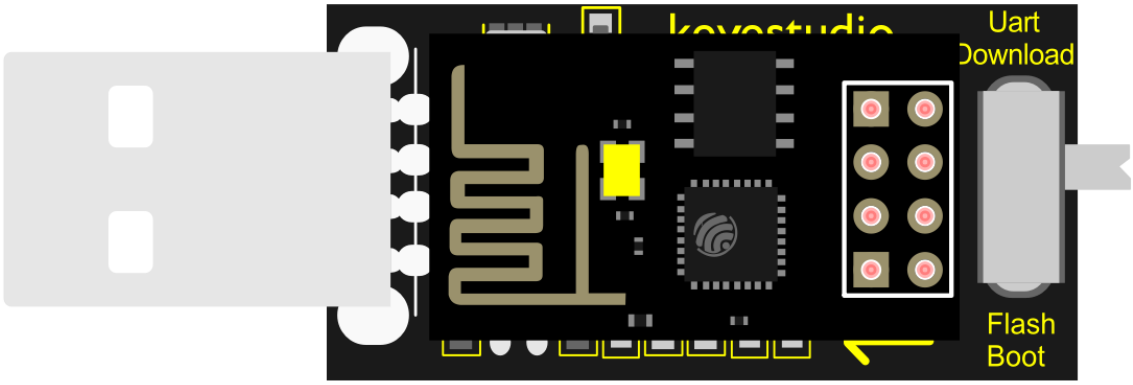
In the previous project 30, we already know that the ESP8266 serial WiFi ESP-01 module gets the relevant WiFi information through the WiFi test code. So in this project, we will use the ESP8266 serial WiFi ESP-01 module to control the effect of LED lighting up and off on the Plus control board through APP and WiFi.

Components Required

		
Keyestudio Plus Mainboardx1	USB to ESP-01S WiFi Module Serial Shieldx1	M-F Dupont Wires
	 	
ESP8266 Serial WiFi ESP-01 Modulex1	Smartphone/IPadx1	USB Cablex1

Plug the Shield into the USB port of the computer

Insert the ESP8266 serial WiFi ESP-01 module in the correct orientation into the USB to ESP-01S WiFi module serial shield.



First, turn the DIP switch on the USB to ESP-01S WiFi module serial shield to the UartDownload, and then insert the shield into the USB port of the computer.

ESP8266 Code

```
/*

Keyestudio 2021 starter learning kit

Project 31.1

ESP8266_Code

http//www.keyestudio.com

*/
```

(continues on next page)

(continued from previous page)

```
// generated by KidsBlock

\#include \<Arduino.h\>

\#include \<ESP8266WiFi.h\>

\#include \<ESP8266mDNS.h\>

\#include \<WiFiClient.h\>

\#ifndef STASSID

\#define STASSID "ChinaNet-2.4G-0DF0" //the name of user's Wifi

\#define STAPSK "ChinaNet@233" //the password of the user's wifi

\#endif

const charx ssid = STASSID;

const charx password = STAPSK;

WiFiServer server(80);

String unoData = "";

int ip_flag = 0;

int ultra_state = 1;

String ip_str;

void setup() {

Serial.begin(9600);

WiFi.mode(WIFI_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.print("IP ADDRESS: ");

Serial.println(WiFi.localIP());
```

(continues on next page)

(continued from previous page)

```
if (!MDNS.begin("esp8266")) {

//Serial.println("Error setting up MDNS responder!");

while (1) {

delay(1000);

}

}

// Serial.println("mDNS responder started");

server.begin();

//Serial.println("TCP server started");

MDNS.addService("http", "tcp", 80);

ip_flag = 1;

}

void loop() {

if(ip_flag == 1)

{

Serial.print("IP: ");

Serial.println(WiFi.localIP());

//Serial.print('\#');

delay(100);

}

MDNS.update();

WiFiClient client = server.available();

if (!client) {

return;

}

//Serial.println("");
```

(continues on next page)

(continued from previous page)

```
while (client.connected() && !client.available()) {

delay(1);

}

String req = client.readStringUntil('\\r');

int addr_start = req.indexOf(' ');

int addr_end = req.indexOf(' ', addr_start + 1);

if (addr_start == -1 \|\| addr_end == -1) {

//Serial.print("Invalid request: ");

//Serial.println(req);

return;

}

req = req.substring(addr_start + 1, addr_end);

client.flush();

String s;

if (req == "/") {

IPAddress ip = WiFi.localIP();

String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +
String(ip[3]);

s = "HTTP/1.1 200 OK\\r\\nContent-Type: text/html\\r\\n\\r\\n\<!DOCTYPE
HTML\>\\r\\n\<html\>Hello from ESP8266 at ";

s += ipStr;

s += "\</html\>\\r\\n\\r\\n";

//Serial.println("Sending 200");

Serial.println(WiFi.localIP());

Serial.write('x');

client.println(WiFi.localIP());

ip_flag = 0;
```

(continues on next page)

(continued from previous page)

```
}

else if(req == "/btn/0")

{

Serial.write('a');

client.println("turn on the relay");

}

else if(req == "/btn/1")

{

Serial.write('b');

client.println("turn off the relay");

}

else if(req == "/btn/2")

{

Serial.write('c');

client.println("Bring the steering gear over 180 degrees");

}

else if(req == "/btn/3")

{

Serial.write('d');

client.println("Bring the steering gear over 0 degrees");

}

else if(req == "/btn/4")

{

Serial.write('e');

client.println("esp8266 already turn on the fans");

}
```

(continues on next page)

(continued from previous page)

```
else if(req == "/btn/5")

{

Serial.write('f');

client.println("esp8266 already turn off the fans");

}

else if(req == "/btn/6")

{

Serial.write('g');

while(Serial.available() \> 0)

{

unoData = Serial.readStringUntil('\#');

client.println(unoData);

}

}

else if(req == "/btn/7")

{

Serial.write('h');

client.println("turn off the ultrasonic");

}

else if(req == "/btn/8")

{

Serial.write('i');

while(Serial.available() \> 0)

{

unoData = Serial.readStringUntil('\#');

client.println(unoData);
```

(continues on next page)

(continued from previous page)

```
//client.flush();

}

}

else if(req == "/btn/9")

{

Serial.write('j');

client.println("turn off the temperature");

}

else if(req == "/btn/10")

{

Serial.write('k');

while(Serial.available() \> 0)

{

unoData = Serial.readStringUntil('\#');

client.println(unoData);

//client.flush();

}

}

else if(req == "/btn/11")

{

Serial.write('l');

client.println("turn off the humidity");

}

else if(req == "/btn/12")

{

Serial.write('m');
```

(continues on next page)

(continued from previous page)

```
client.println(F("m"));

}

else if(req == "/btn/13")

{

Serial.write('n');

client.println(F("n"));

}

else if(req == "/btn/14")

{

Serial.write('o');

client.println(F("o"));

}

else if(req == "/btn/15")

{

Serial.write('p');

client.println(F("p"));

}

else if(req == "/btn/16")

{

Serial.write('q');

client.println(F("q"));

}

else if(req == "/btn/17")

{

Serial.write('r');

client.println(F("r"));
```

(continues on next page)

(continued from previous page)

```
}

else if(req == "/btn/18")

{

Serial.write('s');

client.println(F("s"));

}

else if(req == "/btn/19")

{

Serial.write('t');

client.println(F("t"));

}

else if(req == "/btn/20")

{

Serial.write('u');

client.println(F("u"));

}

else if(req == "/btn/21")

{

Serial.write('v');

client.println(F("v"));

}

else if(req == "/btn/22")

{

Serial.write('w');

client.println(F("w"));

}
```

(continues on next page)

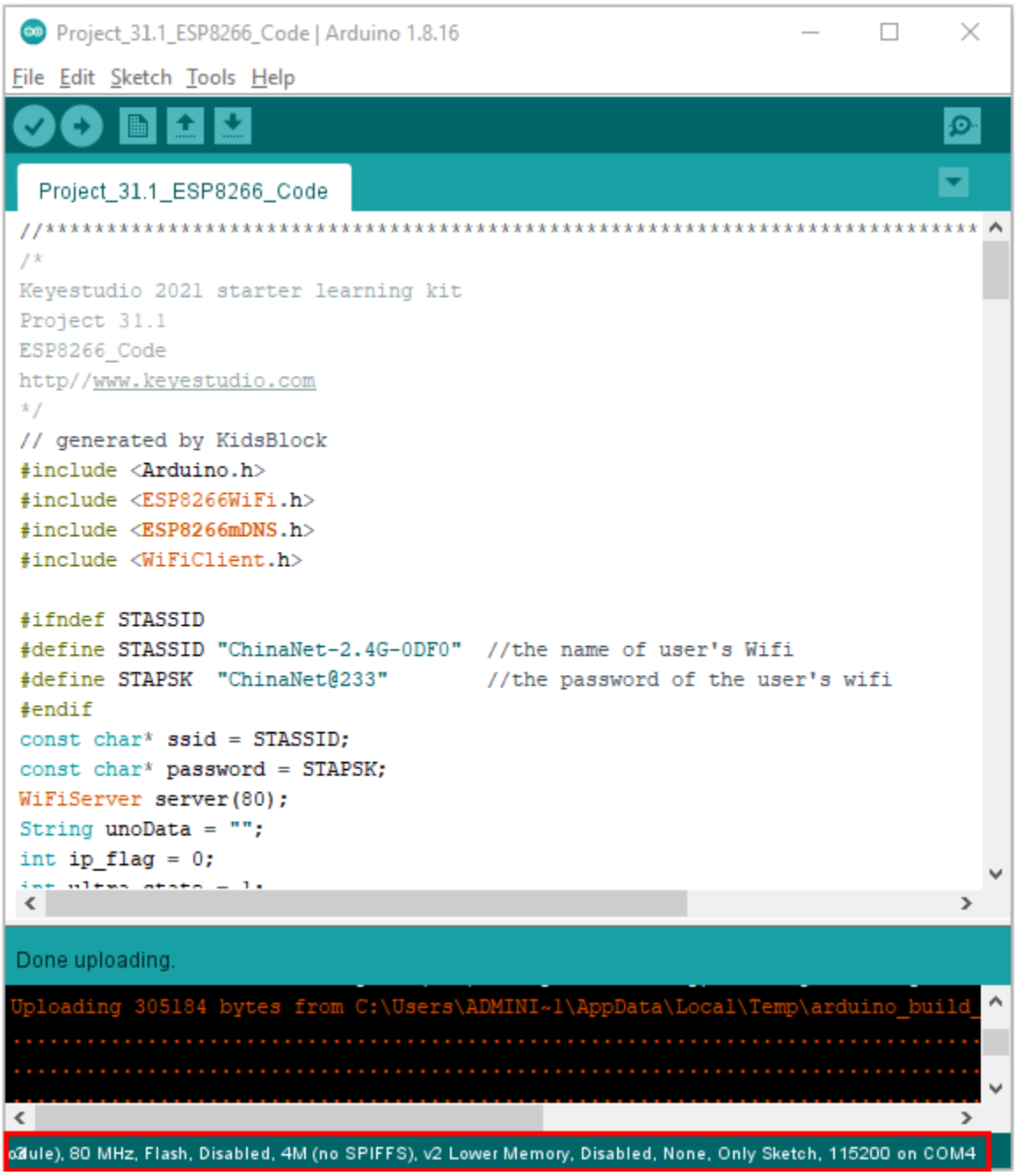
(continued from previous page)

```
else if(req == "/btn/23")

{

Serial.write('x');

client.println(F("x"));

}

else {

//s = "HTTP/1.1 404 Not Found\\r\\n\\r\\n";

//Serial.println("Sending 404");

}

client.print(F("IP : "));

client.println(WiFi.localIP());

}
```

Note: You need to change the user WiFi name and user WiFi password in the project code to your own WiFi name and WiFi password.

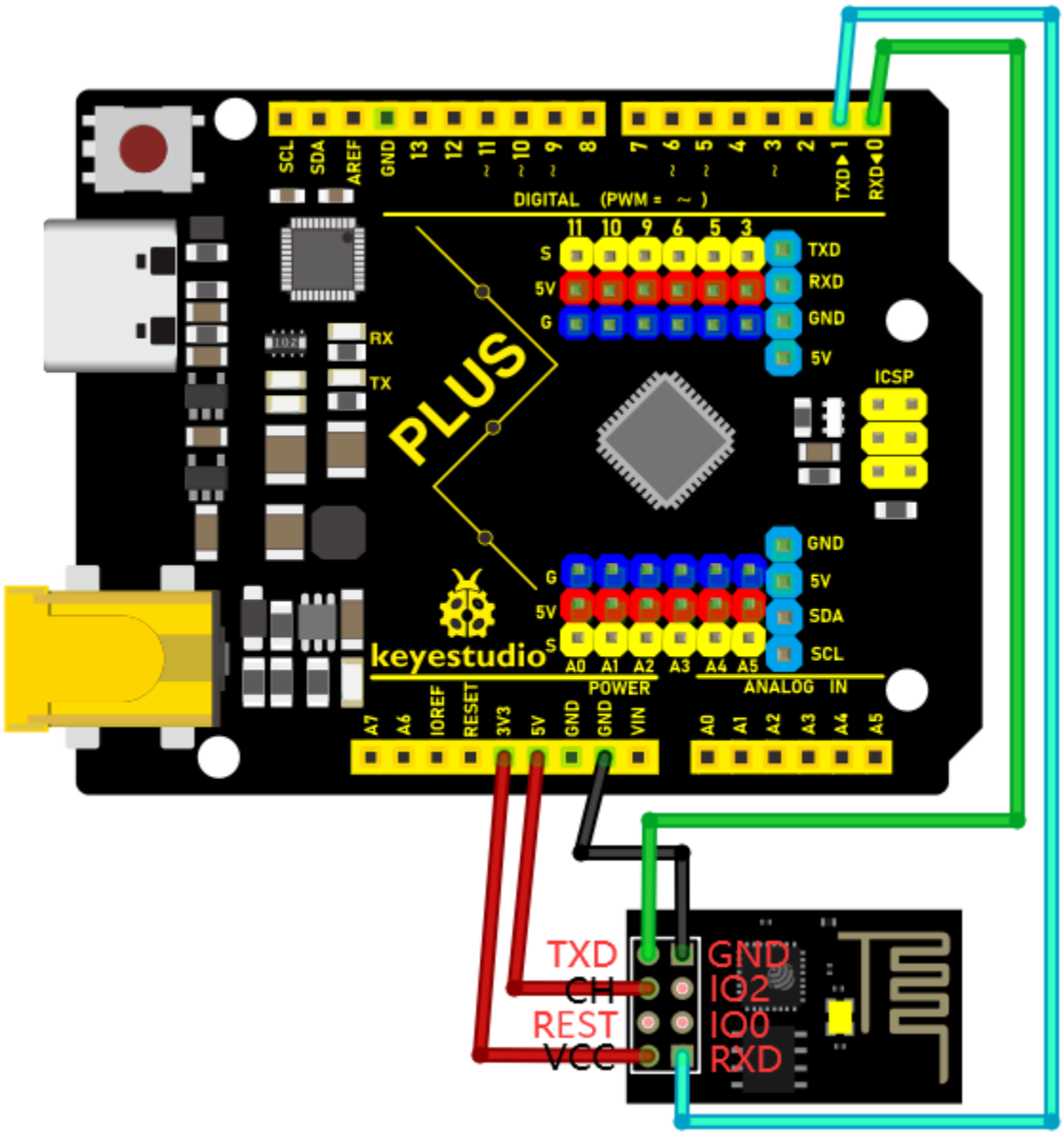
```
//#define STASSID "your-ssid"
//#define STAPSK  "your-password"
#define STASSID "ChinaNet-2.4G-0DF0"   //the name of user's wifi
#define STAPSK  "ChinaNet@233"        //the password of user's wifi
#endif
```

After changing the WiFi name and WiFi password, ensure that the DIP switch on the shield has been turned to the UartDownload and the shield has been plugged into the computer. Then set the board type and COM port according to the method in Project 30, and the corresponding board type and COM port are displayed in the lower right corner of the IDE. Click  to upload the test code to the ESP8266 serial WiFi ESP-01 module, then upload is complete.Note: If the upload fails, unplug the shield and plug it into the computer's USB port again when the board type and COM port are OK.)



After the test code is uploaded successfully, first unplug the shield from the USB port of the computer, and then unplug the ESP8266 serial WiFi ESP-01 module from the shield.

Wiring Diagram



Project Code

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

```
/*

Keyestudio 2021 starter learning kit

Project 31.2

Control LED With WiFi

http//www.keyestudio.com
```

(continues on next page)

(continued from previous page)

```
*/

const int ledPin = 13;

char wifiData;

void setup() {

Serial.begin(9600);

pinMode(ledPin, OUTPUT);

}

void loop() {

if(Serial.available() \> 0)

{

wifiData = Serial.read();

Serial.print(wifiData);

if(wifiData == '\#')

{

Serial.println("");

}

delay(100);

if(wifiData == 'a')

{

digitalWrite(ledPin, HIGH);

}

else if(wifiData == 'b')

{

digitalWrite(ledPin, LOW);

}

}
```

(continues on next page)

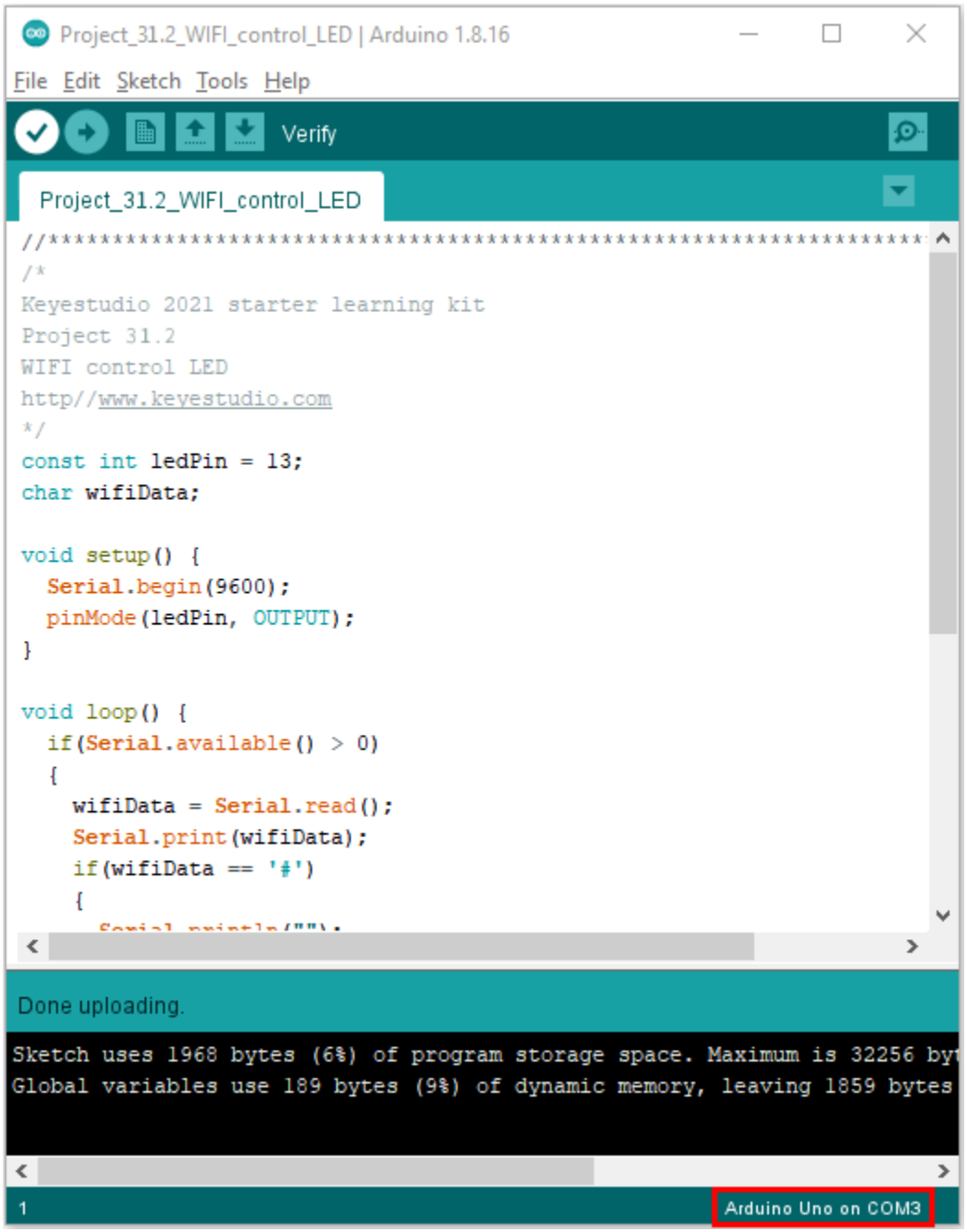
(continued from previous page)

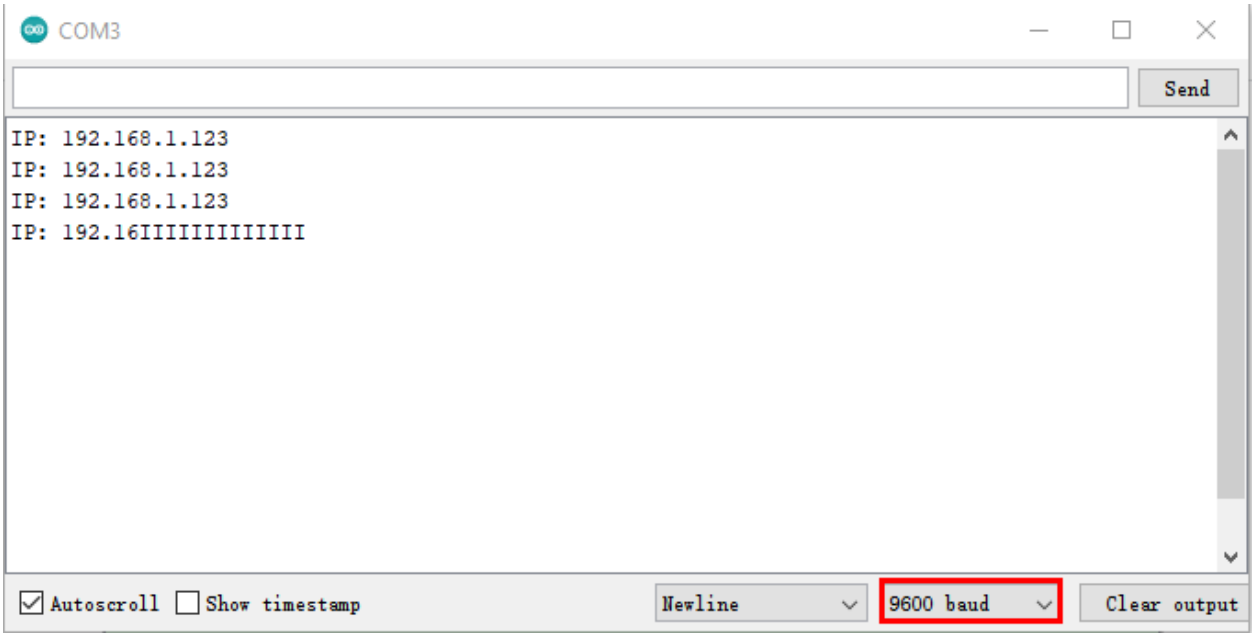
```
}
```

Result

Note: Before uploading the project code, you need to unplug the TX and RX cables connected to the Plus control board first, otherwise the code will not be uploaded successfully. Then click "Tools" → "Board:", select the Arduino UNO board and choose the correct COM port. Finally, upload the project code to the Plus mainboard. After uploading the code successfully, connect the other end of the TX Dupont wire on the ESP8266 serial WiFi ESP-01 module to the RX(0) pin on the Plus control board. The other end of RX Dupont wire is connected to the TX(1) pin on the PLUS control board. Click  to open serial monitor window and set the baud rate to 9600.

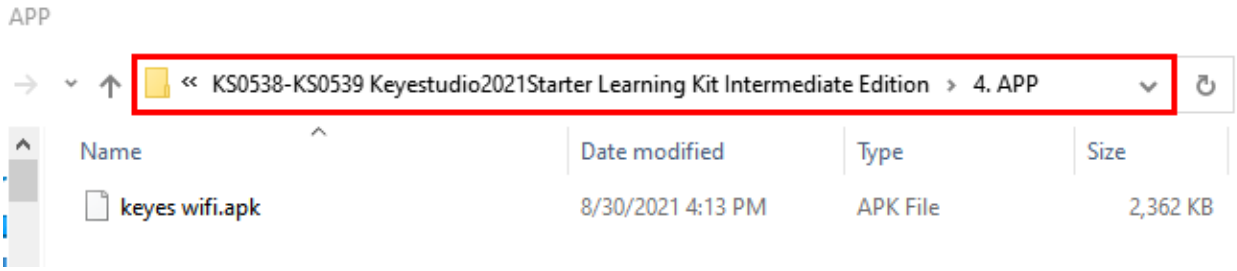
In this way, the serial monitor shows the IP address of your WiFi. (The IP address of WiFi sometimes changes. If the original IP address does not work, you need to detect the IP address again.)

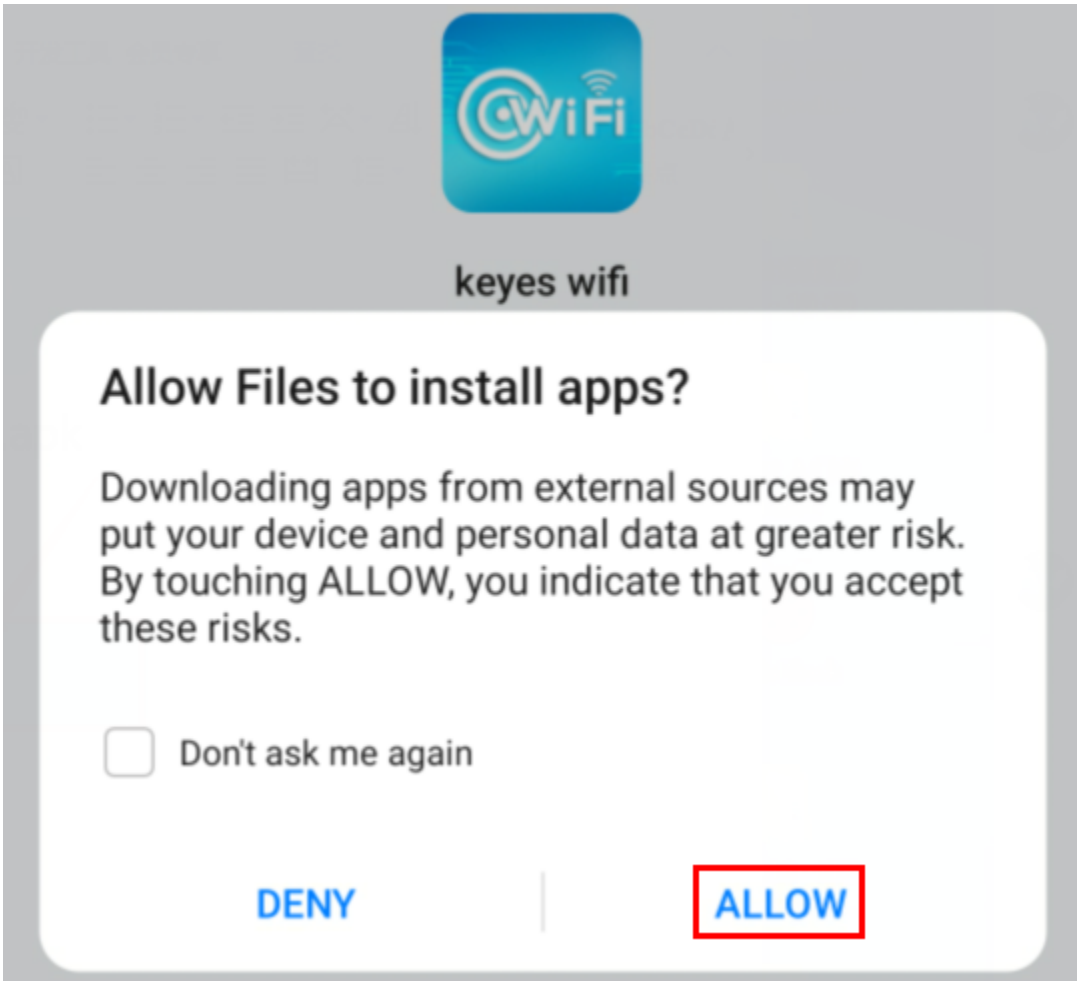

Project_31.2_WIFI_control_LED | Arduino 1.8.16
File Edit Sketch Tools Help
Verify
Project_31.2_WIFI_control_LED
//**********
/*
Keyestudio 2021 starter learning kit
Project 31.2
WIFI control LED
http//www.keyestudio.com
*/
const int ledPin = 13;
char wifiData;
void setup() {
Serial.begin(9600);
pinMode(ledPin, OUTPUT);
}
void loop() {
if(Serial.available() > 0)
{
wifiData = Serial.read();
Serial.print(wifiData);
if(wifiData == '#')
{
Done uploading.
Sketch uses 1968 bytes (6%) of program storage space. Maximum is 32256 byt
Global variables use 189 bytes (9%) of dynamic memory, leaving 1859 bytes
1
Arduino Uno on COM3

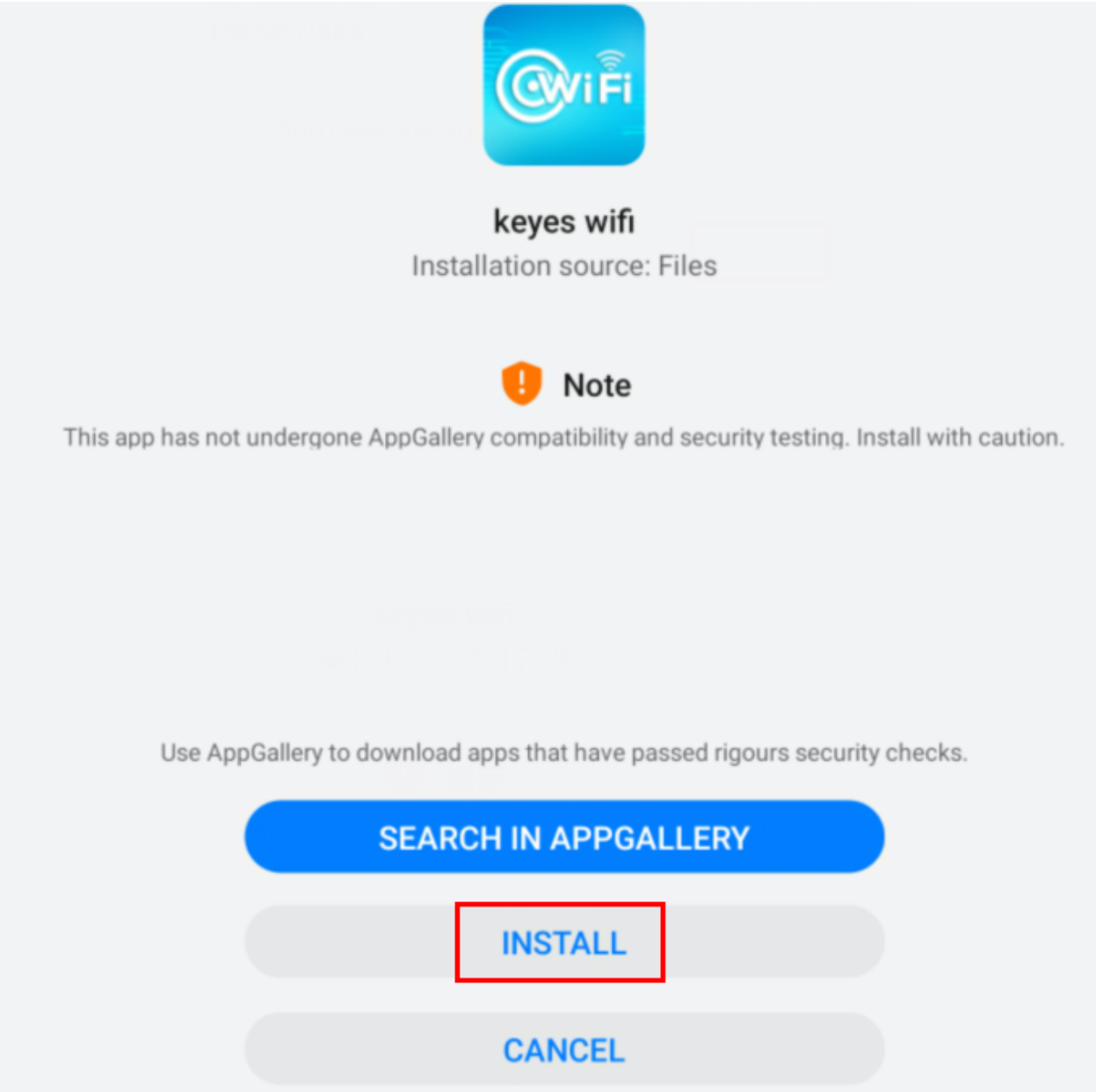


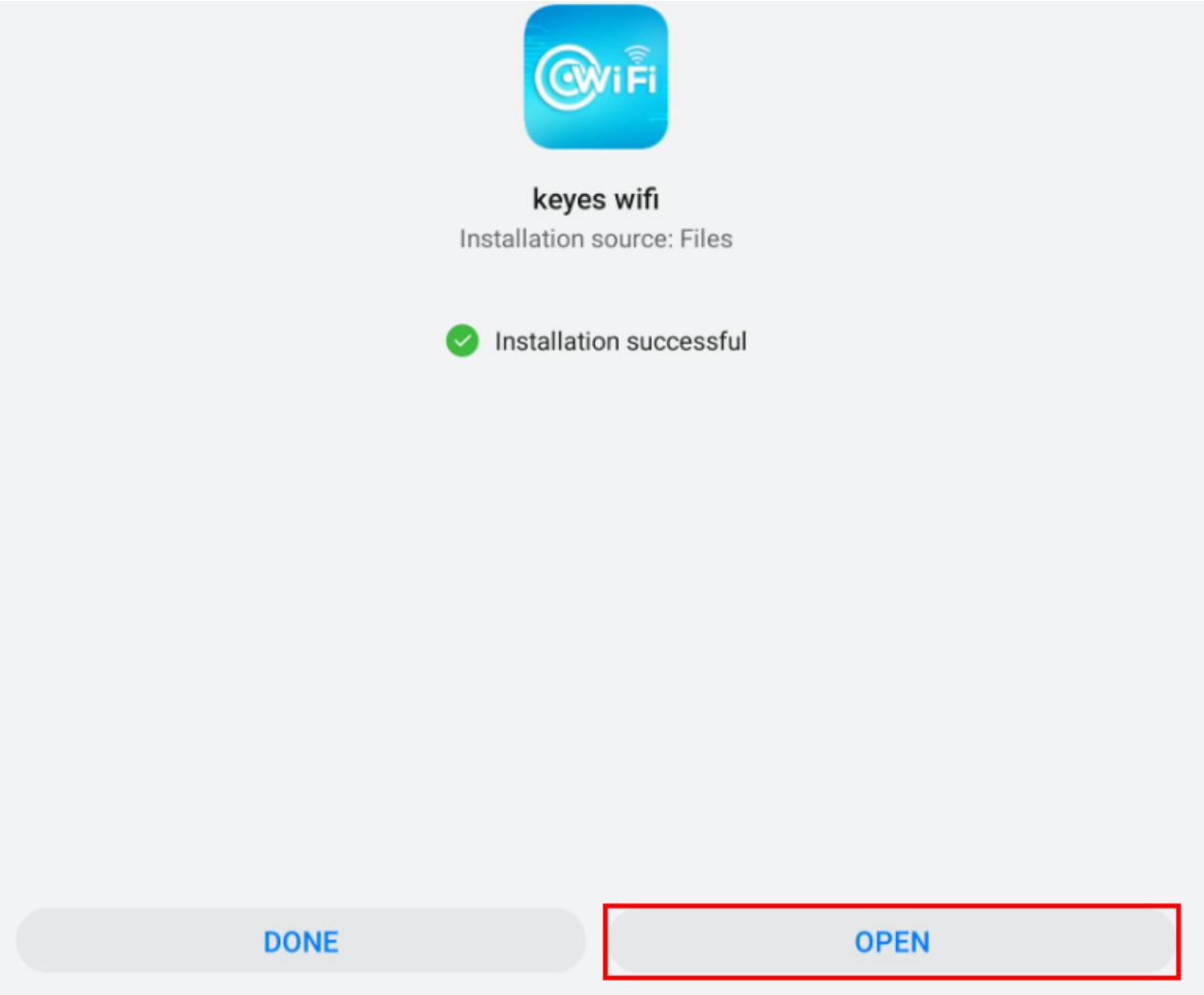
App for Android system devices(mobile phone/iPad)

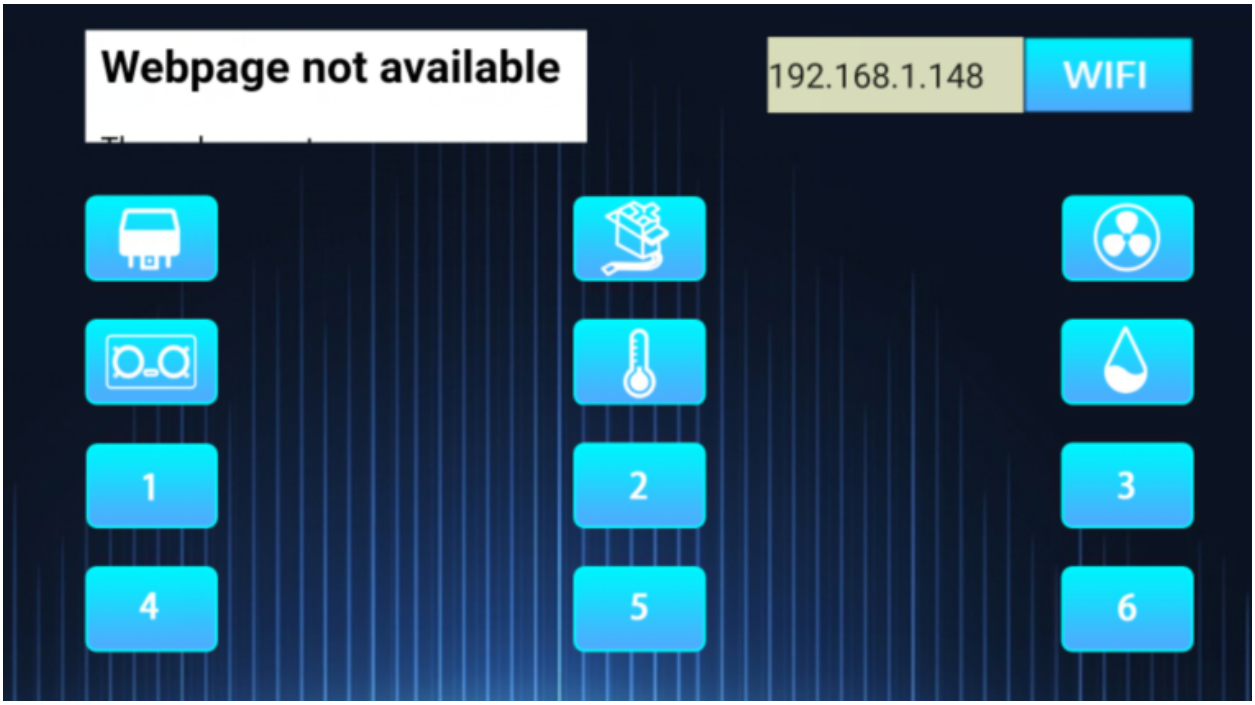
Now transfer the "keyes wifi.apk" file from the folder to your Android phone or iPad, click the "keyes wifi.apk" file to enter the installation page. Click the "**ALLOW**" button, and then click the "**INSTALL**" button. Click the "**OPEN**" button to enter the APP interface after the installation is completed.




keyes wifi
Allow Files to install apps?
Downloading apps from external sources may put your device and personal data at greater risk. By touching ALLOW, you indicate that you accept these risks.
Don't ask me again
DENY
ALLOW


keyes wifi
Installation source: Files
Note
This app has not undergone AppGallery compatibility and security testing. Install with caution.
Use AppGallery to download apps that have passed rigours security checks.
SEARCH IN APPGALLERY
INSTALL
CANCEL


keyes wifi
Installation source: Files
Installation successful
DONE
OPEN



Enter the detected WiFi IP address in the text box in front of the WiFi button(For example, the IP address detected by the serial monitor above is 192.168.1.123), then click the WiFi button, “Webpage not available” will become “192.168.1.123”.This shows that the App has been connected to WiFi.

App for IOS system devices (mobile phone/iPad)

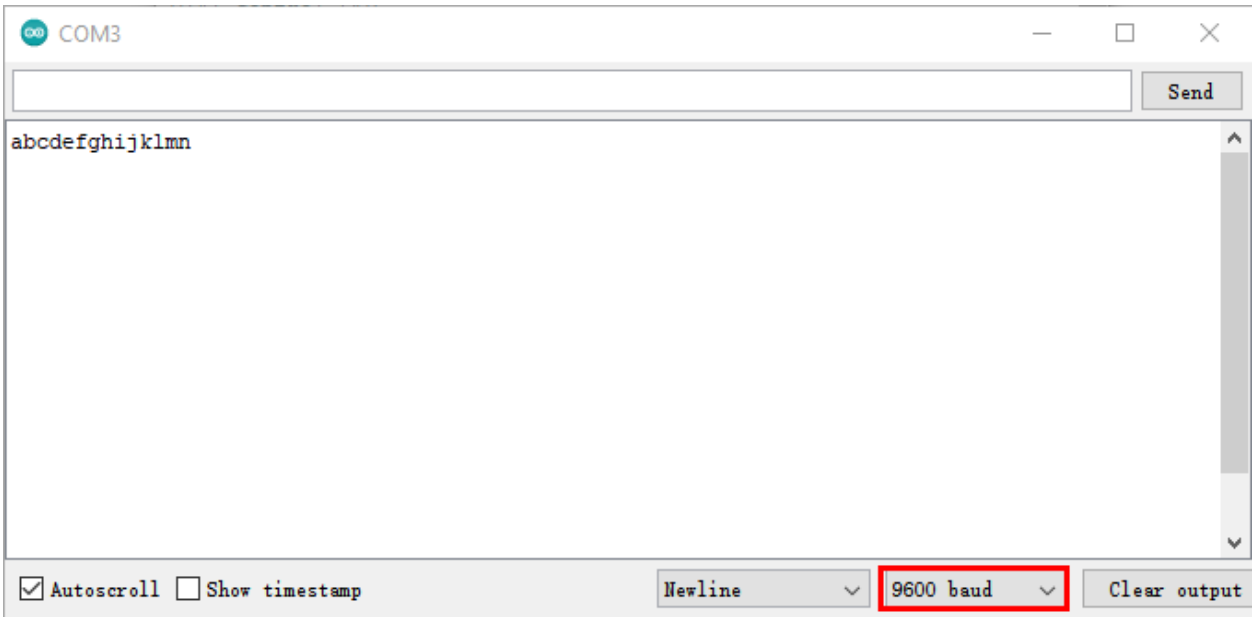
Open App Store.

Enter “keyes wifi” in the search box, search and the download screen will appear. Click“ ”, you can download and install keyes wifi APP. The following operations are similar to those of Android system. You can refer to the steps of Android system above for operation.

Note: Click the button on APP, the blue light on ESP8266 serial WiFi ESP-01 module will flash, indicating that APP has connected to WiFi.

After the APP has been connected to the WiFi, start the following operations.

Click the button on the APP, the serial monitor prints some corresponding control characters, as shown below.



Click  ,the LED on the Plus control board lights up. Click  again, the LED on the Plus control board goes out.

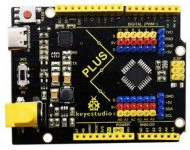
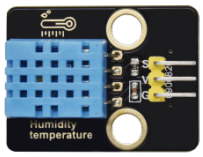
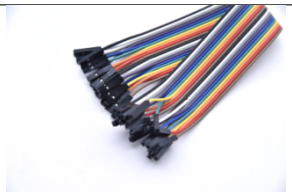
5.32 Project 32: WiFi Smart Home

Introduction

In the previous project 31, we already knew how to connect the APP to WiFi and also use the APP to control the LED on and off on the Plus control board through WiFi for a simple experiment.

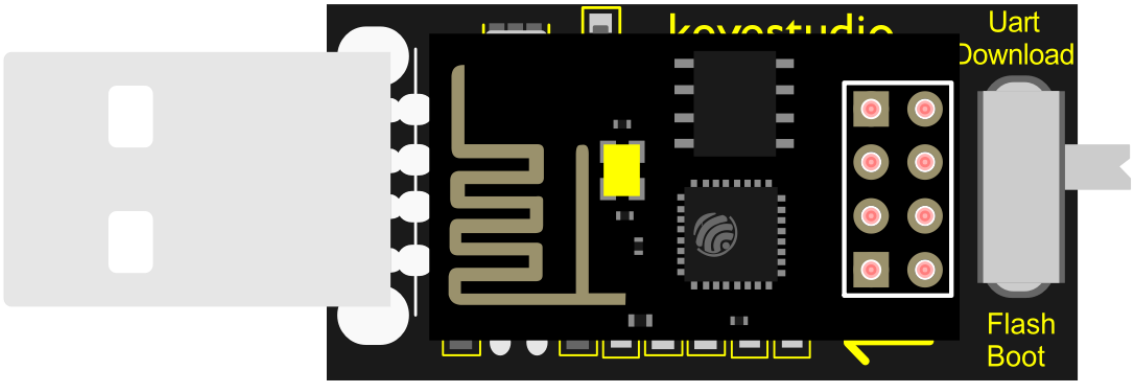
In this project, we will use APP to control multiple sensors or modules through WiFi to achieve the effect of smart home.

Components Required

			
Keyestudio Plus Main-boardx1	USB to ESP-01S WiFi Module Serial Shieldx1	ESP8266 Serial WiFi ESP-01 Modulex1	DHT11 Temperature and Humidity Sensorx1
			
Servox1	Ultrasonic Sensorx1	5V Relay Modulex1	USB Cablex1
			
Mobile Phone/iPadx1	M-M Dupont Wires	M-F Dupont Wires	

Plug the WiFi Module Serial Shield into the USB port of the computer

Insert the ESP8266 serial WiFi ESP-01 module in the correct orientation into the USB to ESP-01S WiFi module serial shield.



First, turn the DIP switch on the USB to ESP-01S WiFi module serial shield to the UartDownload, and then insert the shield into the USB port of the computer.

ESP8266 Code

```
/*

Keyestudio 2021 starter learning kit

Project 32.1

ESP8266_Code

http//www.keyestudio.com

*/
```

(continues on next page)

(continued from previous page)

```
// generated by KidsBlock

\#include \<Arduino.h\>

\#include \<ESP8266WiFi.h\>

\#include \<ESP8266mDNS.h\>

\#include \<WiFiClient.h\>

\#ifndef STASSID

\#define STASSID "ChinaNet-2.4G-0DF0" // the name of user's wifi

\#define STAPSK "ChinaNet@233" //the password of user's wifi

\#endif

const charx ssid = STASSID;

const charx password = STAPSK;

WiFiServer server(80);

String unoData = "";

int ip_flag = 0;

int ultra_state = 1;

String ip_str;

void setup() {

Serial.begin(9600);

WiFi.mode(WIFI_STA);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.print("IP ADDRESS: ");

Serial.println(WiFi.localIP());
```

(continues on next page)

(continued from previous page)

```
if (!MDNS.begin("esp8266")) {

//Serial.println("Error setting up MDNS responder!");

while (1) {

delay(1000);

}

}

// Serial.println("mDNS responder started");

server.begin();

//Serial.println("TCP server started");

MDNS.addService("http", "tcp", 80);

ip_flag = 1;

}

void loop() {

if(ip_flag == 1)

{

Serial.print("IP: ");

Serial.println(WiFi.localIP());

//Serial.print('\#');

delay(100);

}

MDNS.update();

WiFiClient client = server.available();

if (!client) {

return;

}

//Serial.println("");
```

(continues on next page)

(continued from previous page)

```
while (client.connected() && !client.available()) {

delay(1);

}

String req = client.readStringUntil('\\r');

int addr_start = req.indexOf(' ');

int addr_end = req.indexOf(' ', addr_start + 1);

if (addr_start == -1 \|\| addr_end == -1) {

//Serial.print("Invalid request: ");

//Serial.println(req);

return;

}

req = req.substring(addr_start + 1, addr_end);

client.flush();

String s;

if (req == "/") {

IPAddress ip = WiFi.localIP();

String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +
String(ip[3]);

s = "HTTP/1.1 200 OK\\r\\nContent-Type: text/html\\r\\n\\r\\n\<!DOCTYPE
HTML\>\\r\\n\<html\>Hello from ESP8266 at ";

s += ipStr;

s += "\</html\>\\r\\n\\r\\n";

//Serial.println("Sending 200");

Serial.println(WiFi.localIP());

Serial.write('x');

client.println(WiFi.localIP());

ip_flag = 0;
```

(continues on next page)

(continued from previous page)

```
}

else if(req == "/btn/0")

{

Serial.write('a');

client.println("turn on the relay");

}

else if(req == "/btn/1")

{

Serial.write('b');

client.println("turn off the relay");

}

else if(req == "/btn/2")

{

Serial.write('c');

client.println("Bring the steering gear over 180 degrees");

}

else if(req == "/btn/3")

{

Serial.write('d');

client.println("Bring the steering gear over 0 degrees");

}

else if(req == "/btn/4")

{

Serial.write('e');

client.println("esp8266 already turn on the fans");

}
```

(continues on next page)

(continued from previous page)

```
else if(req == "/btn/5")

{

Serial.write('f');

client.println("esp8266 already turn off the fans");

}

else if(req == "/btn/6")

{

Serial.write('g');

while(Serial.available() \> 0)

{

unoData = Serial.readStringUntil('\#');

client.println(unoData);

}

}

else if(req == "/btn/7")

{

Serial.write('h');

client.println("turn off the ultrasonic");

}

else if(req == "/btn/8")

{

Serial.write('i');

while(Serial.available() \> 0)

{

unoData = Serial.readStringUntil('\#');

client.println(unoData);
```

(continues on next page)

(continued from previous page)

```
//client.flush();

}

}

else if(req == "/btn/9")

{

Serial.write('j');

client.println("turn off the temperature");

}

else if(req == "/btn/10")

{

Serial.write('k');

while(Serial.available() \> 0)

{

unoData = Serial.readStringUntil('\#');

client.println(unoData);

//client.flush();

}

}

else if(req == "/btn/11")

{

Serial.write('l');

client.println("turn off the humidity");

}

else if(req == "/btn/12")

{

Serial.write('m');
```

(continues on next page)

(continued from previous page)

```
client.println(F("m"));

}

else if(req == "/btn/13")

{

Serial.write('n');

client.println(F("n"));

}

else if(req == "/btn/14")

{

Serial.write('o');

client.println(F("o"));

}

else if(req == "/btn/15")

{

Serial.write('p');

client.println(F("p"));

}

else if(req == "/btn/16")

{

Serial.write('q');

client.println(F("q"));

}

else if(req == "/btn/17")

{

Serial.write('r');

client.println(F("r"));
```

(continues on next page)

(continued from previous page)

```
}
else if(req == "/btn/18")
{
Serial.write('s');
client.println(F("s"));
}
else if(req == "/btn/19")
{
Serial.write('t');
client.println(F("t"));
}
else if(req == "/btn/20")
{
Serial.write('u');
client.println(F("u"));
}
else if(req == "/btn/21")
{
Serial.write('v');
client.println(F("v"));
}
else if(req == "/btn/22")
{
Serial.write('w');
client.println(F("w"));
}
```

(continues on next page)

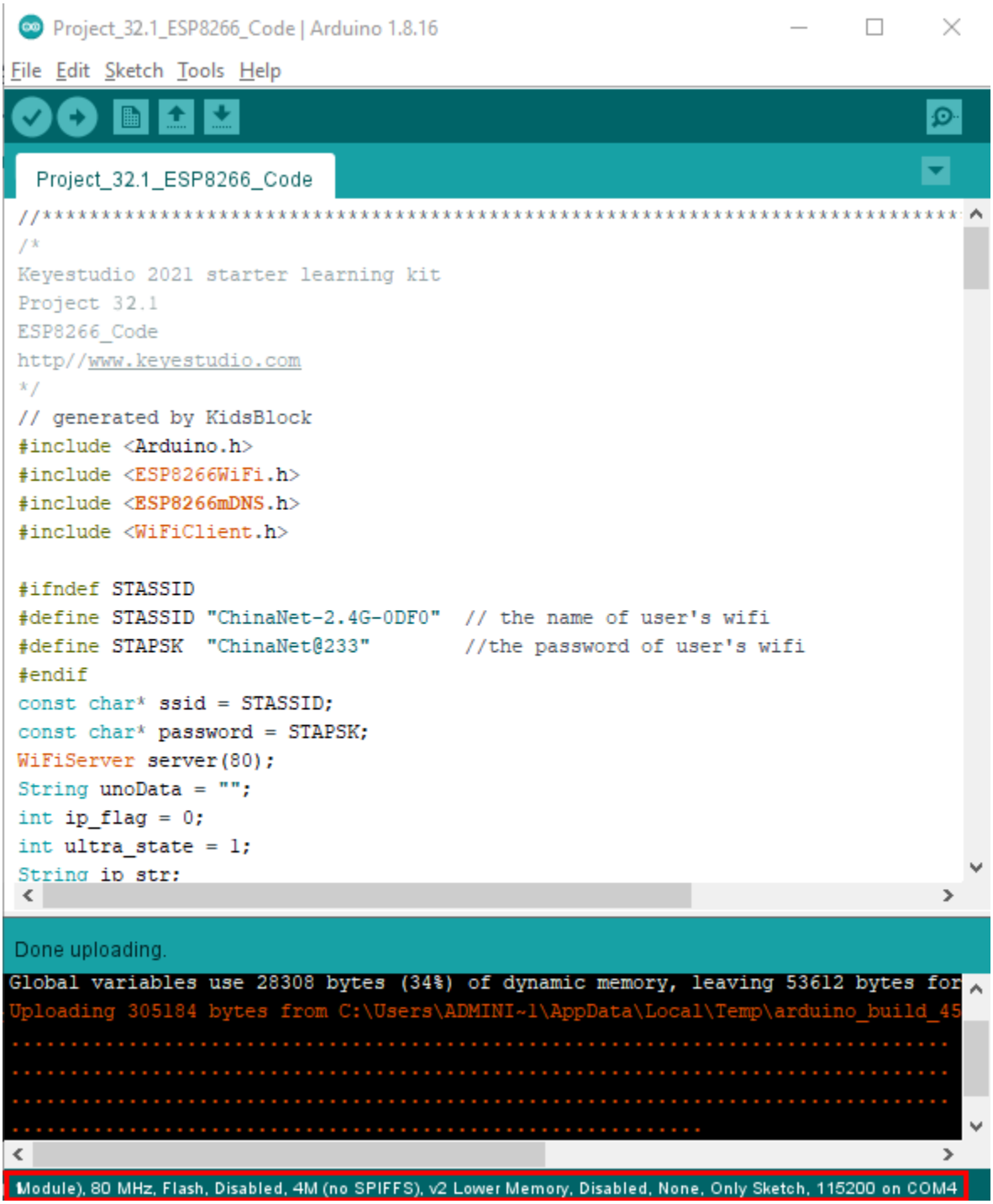
(continued from previous page)

```
else if(req == "/btn/23")

{

Serial.write('x');

client.println(F("x"));

}

else {

//s = "HTTP/1.1 404 Not Found\\r\\n\\r\\n";

//Serial.println("Sending 404");

}

client.print(F("IP : "));

client.println(WiFi.localIP());

}
```

Note: You need to change the user WiFi name and user WiFi password in the project code to your own WiFi name and WiFi password.

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

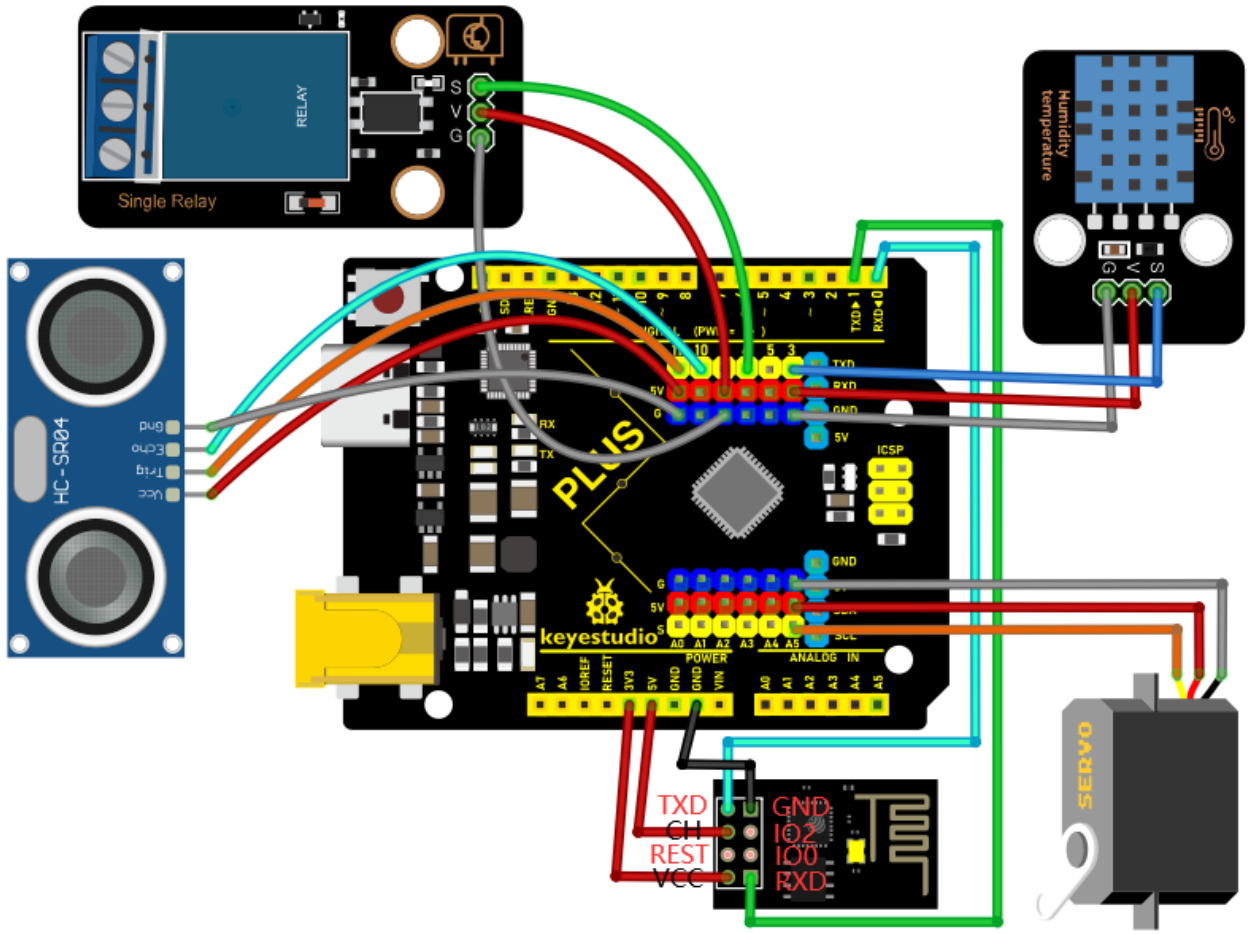
After changing the WiFi name and WiFi password, ensure that the DIP switch on the shield has been turned to the UartDownload and the shield has been plugged into the computer. Then set the board type and COM port according to the method in Project 30, and the corresponding board type and COM port are displayed in the lower right corner of the IDE. Click  to upload the test code to the ESP8266 serial WiFi ESP-01 module, the upload is complete. (Note: If the upload fails, unplug the shield and plug it into the computer's USB port again when the board type and COM port are OK.)



After the test code is uploaded successfully, first unplug the shield from the USB port of the computer, and then unplug the ESP8266 serial WiFi ESP-01 module from the shield.

Wiring Diagram

Relay module	Plus mainboard		DHT11 sensor	Plus mainboard
G	G		G	G
V	5V		V	5V
S	6(S)		S	3(S)
Ultrasonic sensor	Plus mainboard		Servo	Plus mainboard
Vcc	5V		Red line	5V
Trig	11		Brown line	G
Echo	10		Orange line	A5(S)
Gnd	G			
WIFI module	Plus mainboard			
VCC	3V3			
CH	5V			
TXD	RX(0)			
RXD	TX(1)			
GND	GND			



Project Code

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

```
/*

Keyes 2021 starter learning kit

Project 32.2

WIFI smart home

http//www.keyestudio.com

*/

\#include \<DHT.h\>

DHT dht(3, 11);

\#include\<Servo.h\>

Servo myservo;
```

(continues on next page)

(continued from previous page)

```
char wifiData;

int distance1;

String dis_str;

const int dhtPin = 3;

const int relayPin = 6;

const int trigPin = 11;

const int echoPin = 10;

const int servoPin = A5;

int ip_flag = 1;

int ultra_state = 1;

int temp_state = 1;

int humidity_state = 1;

void setup() {

Serial.begin(9600);

pinMode(dhtPin, INPUT);

pinMode(relayPin, OUTPUT);

pinMode(servoPin, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

digitalWrite(relayPin, LOW); //turn off the relay module

myservo.attach(A5);

dht.begin();

}

void loop() {

if(Serial.available() \> 0)

{
```

(continues on next page)

(continued from previous page)

```
wifiData = Serial.read();

// Serial.println(wifiData);

if(wifiData == 'x')

{

ip_flag = 0;

}

if(ip_flag == 1)

{

//String ip_addr = Serial.readStringUntil('\#');

Serial.print(wifiData);

if(wifiData == '\#')

{

Serial.println("");

}

delay(100);

}

}

switch(wifiData)

{

case 'a': digitalWrite(relayPin, HIGH); break;

case 'b': digitalWrite(relayPin, LOW); break;

case 'c': myservo.write(180); delay(200); break;

case 'd': myservo.write(0); delay(200); break;

case 'g': while(ultra_state\>0)

{

Serial.print("Distance = ");
```

(continues on next page)

(continued from previous page)

```
Serial.print(checkdistance());

Serial.println("\#");

ultra_state = 0;

}

break;

case 'h': ultra_state = 1; break;

case 'i': while(temp_state\>0)

{

Serial.print("Temperature = ");

Serial.print(dht.readTemperature());

Serial.println("\#");

temp_state = 0;

}

break;

case 'j': temp_state = 1; break;

case 'k': while(humidity_state \> 0)

{

Serial.print("Humidity = ");

Serial.print(dht.readHumidity());

Serial.println("\#");

humidity_state = 0;

}

break;

case 'l': humidity_state = 1; break;

}

}
```

(continues on next page)

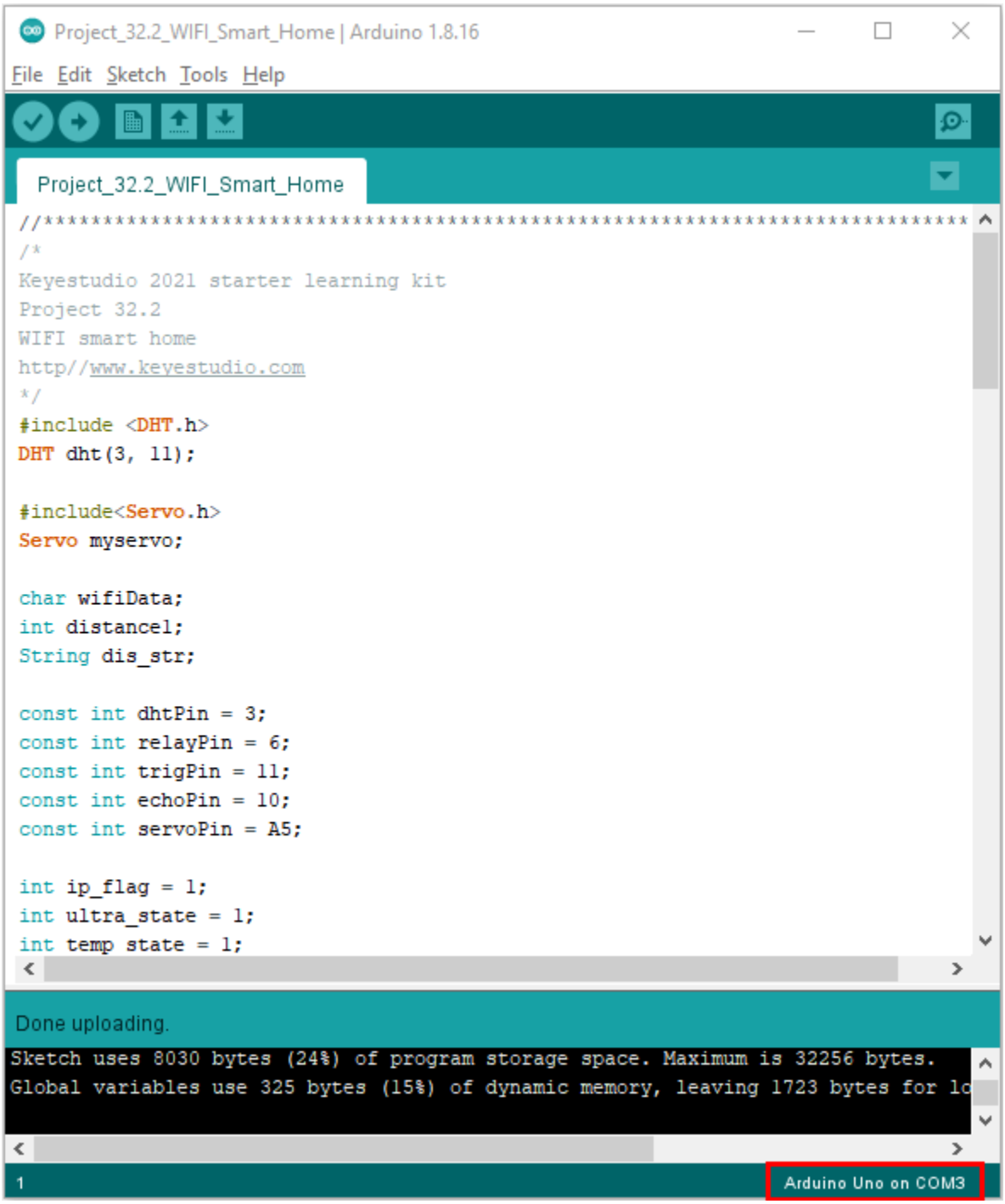
(continued from previous page)

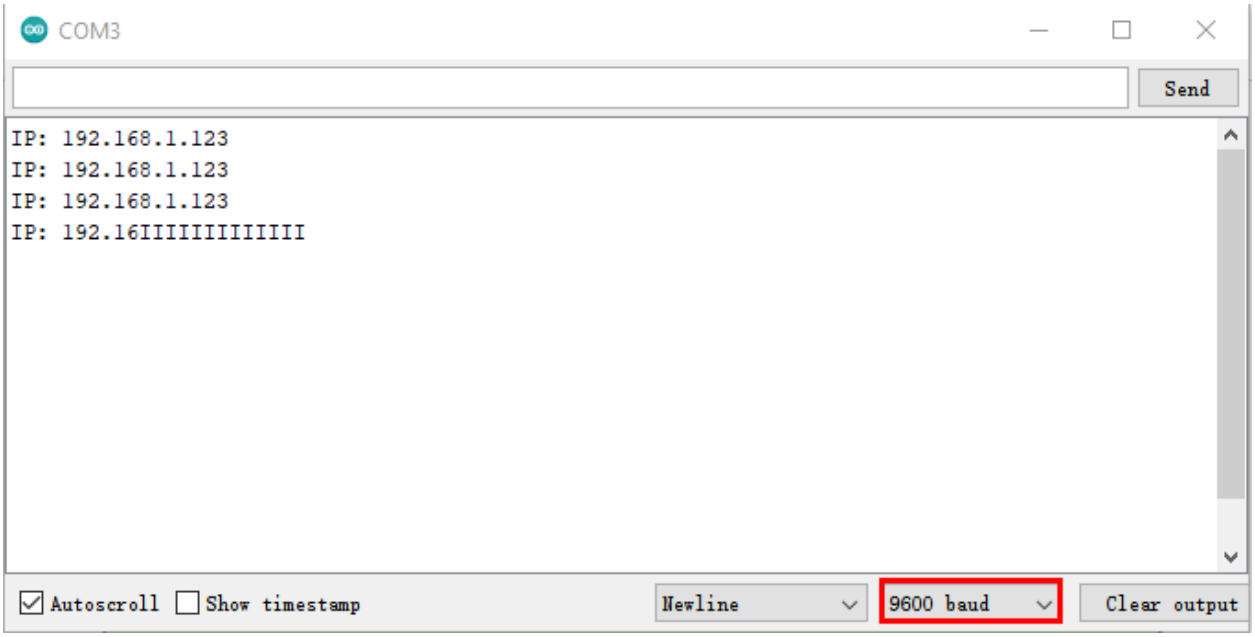
```
int checkdistance() {

digitalWrite(11, LOW);

delayMicroseconds(2);

digitalWrite(11, HIGH);

delayMicroseconds(10);

digitalWrite(11, LOW);

int distance = pulseIn(10, HIGH) / 58;

delay(10);

return distance;

}
```

Result

Note: Before uploading the project code, you need to unplug the TX and RX cables connected to the Plus control board first, otherwise the code will not be uploaded successfully. Then click "Tools" → "Board:", select the Arduino UNO board and choose the correct COM port. Finally, upload the project code to the Plus Mainboard. After uploading the code successfully, connect the other end of the TX Dupont wire on the ESP8266 serial WiFi ESP-01 module to the RX(0) pin on the Plus control board. The other end of RX Dupont wire is connected to the TX(1) pin on the PLUS control board. Click  to open serial monitor window and set the baud rate to 9600.

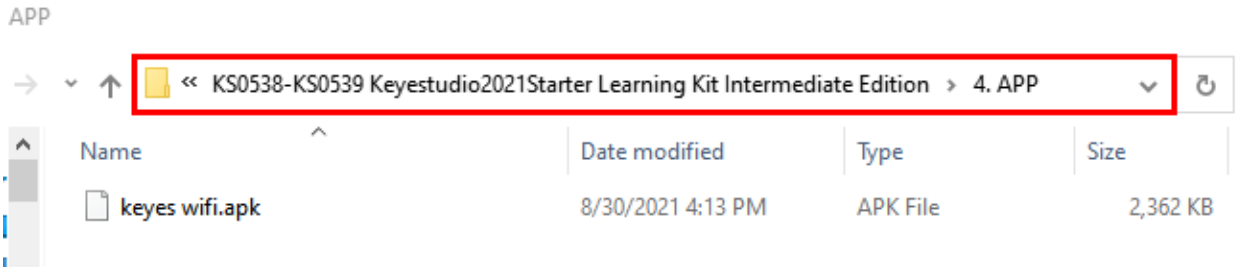
In this way, the serial monitor shows the IP address of your WiFi. (The IP address of WiFi sometimes changes. If the original IP address does not work, you need to detect the IP address again.)

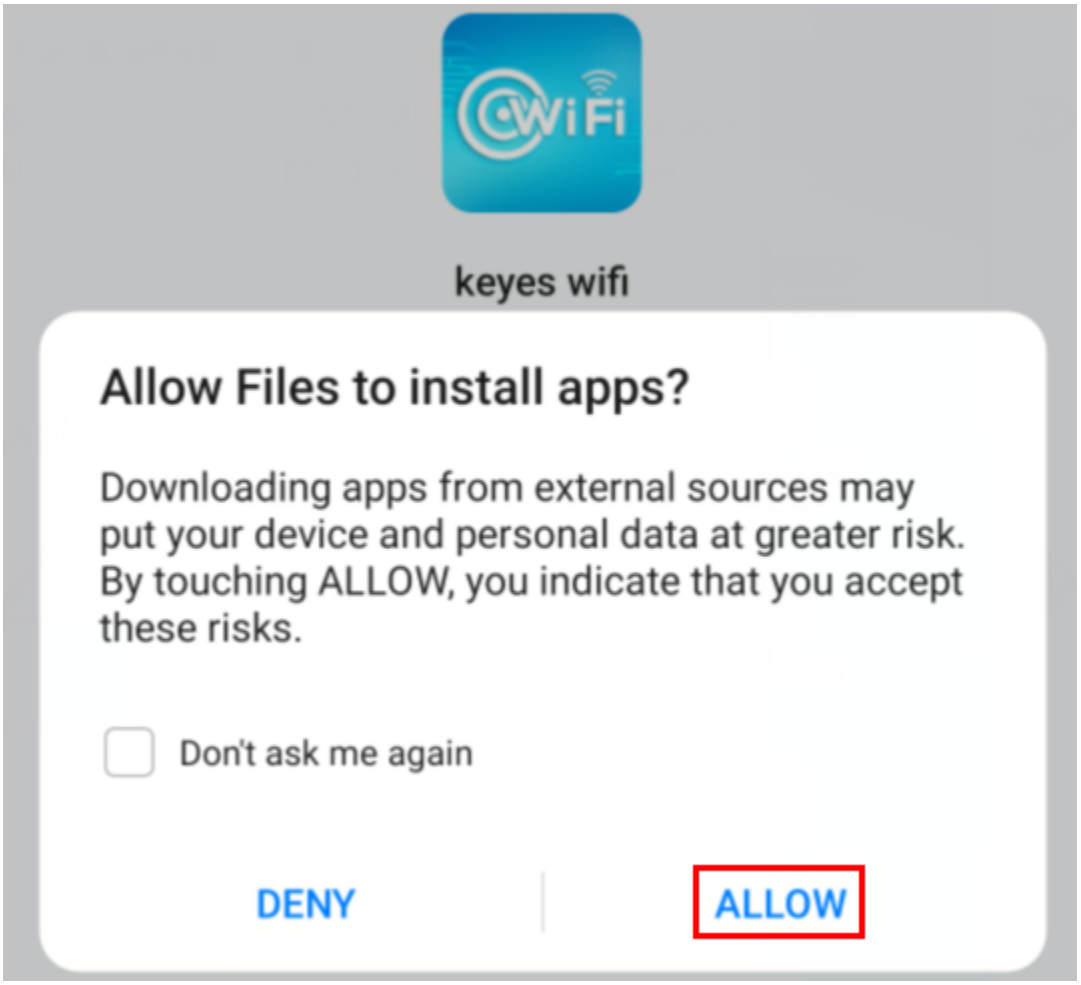

Project_32.2_WIFI_Smart_Home | Arduino 1.8.16
File Edit Sketch Tools Help
Project_32.2_WIFI_Smart_Home
//**********
/*
Keyestudio 2021 starter learning kit
Project 32.2
WIFI smart home
http//www.keyestudio.com
*/
#include <DHT.h>
DHT dht(3, 11);
#include<Servo.h>
Servo myservo;
char wifiData;
int distancel;
String dis_str;
const int dhtPin = 3;
const int relayPin = 6;
const int trigPin = 11;
const int echoPin = 10;
const int servoPin = A5;
int ip_flag = 1;
int ultra_state = 1;
int temp state = 1;
Done uploading.
Sketch uses 8030 bytes (24%) of program storage space. Maximum is 32256 bytes.
Global variables use 325 bytes (15%) of dynamic memory, leaving 1723 bytes for lo
1
Arduino Uno on COM3

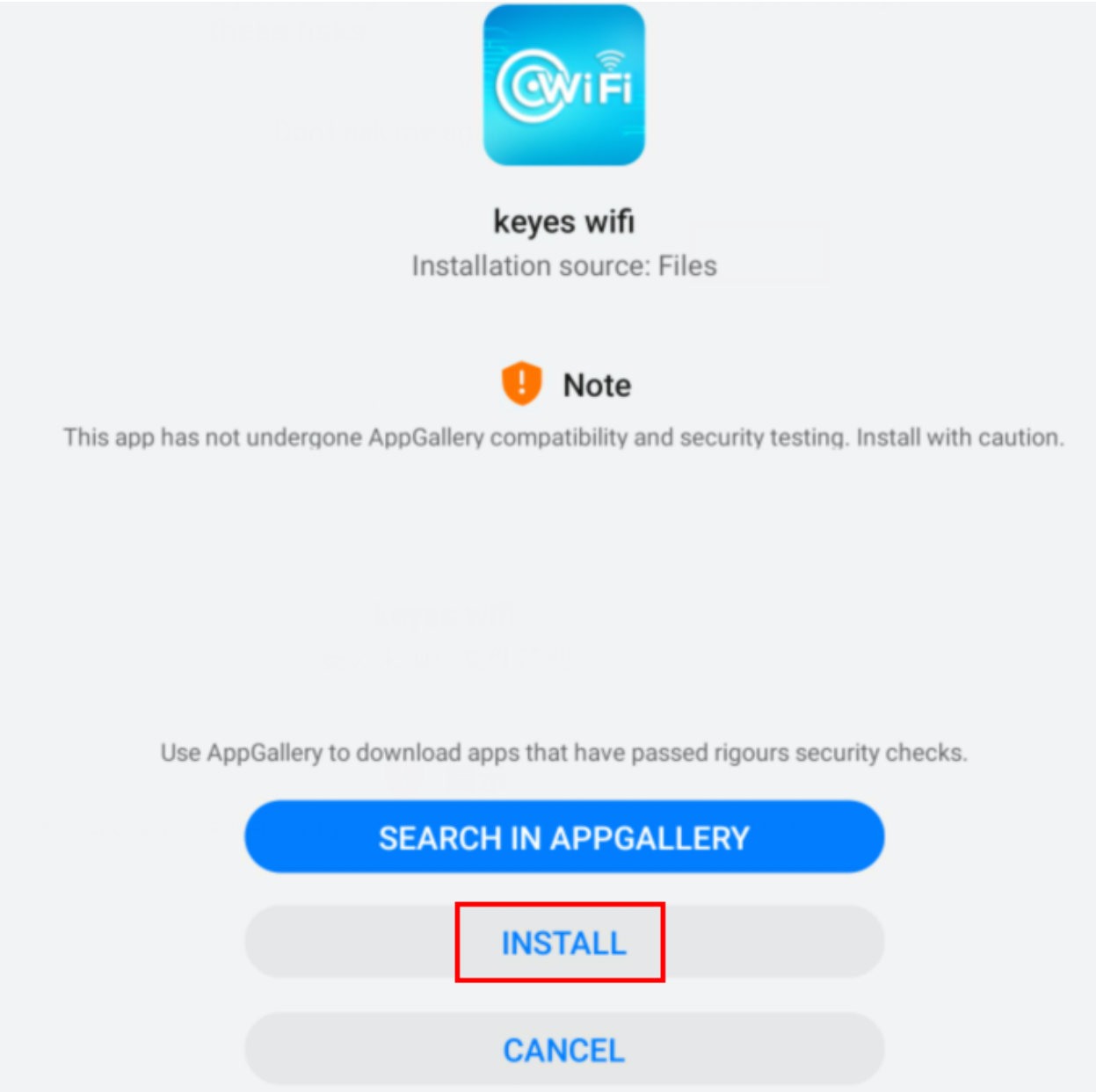


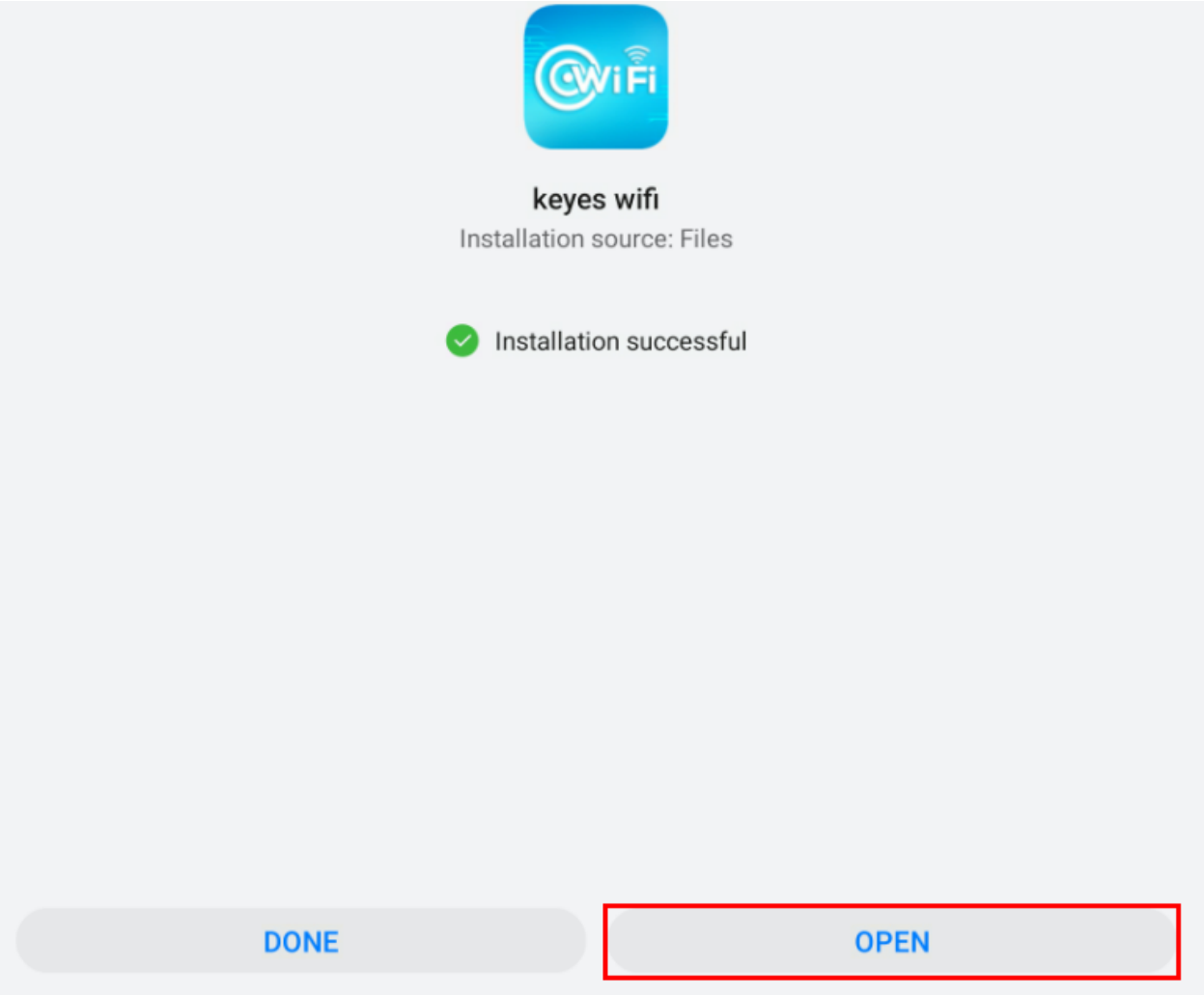
App for Android system devices(mobile phone/iPad)

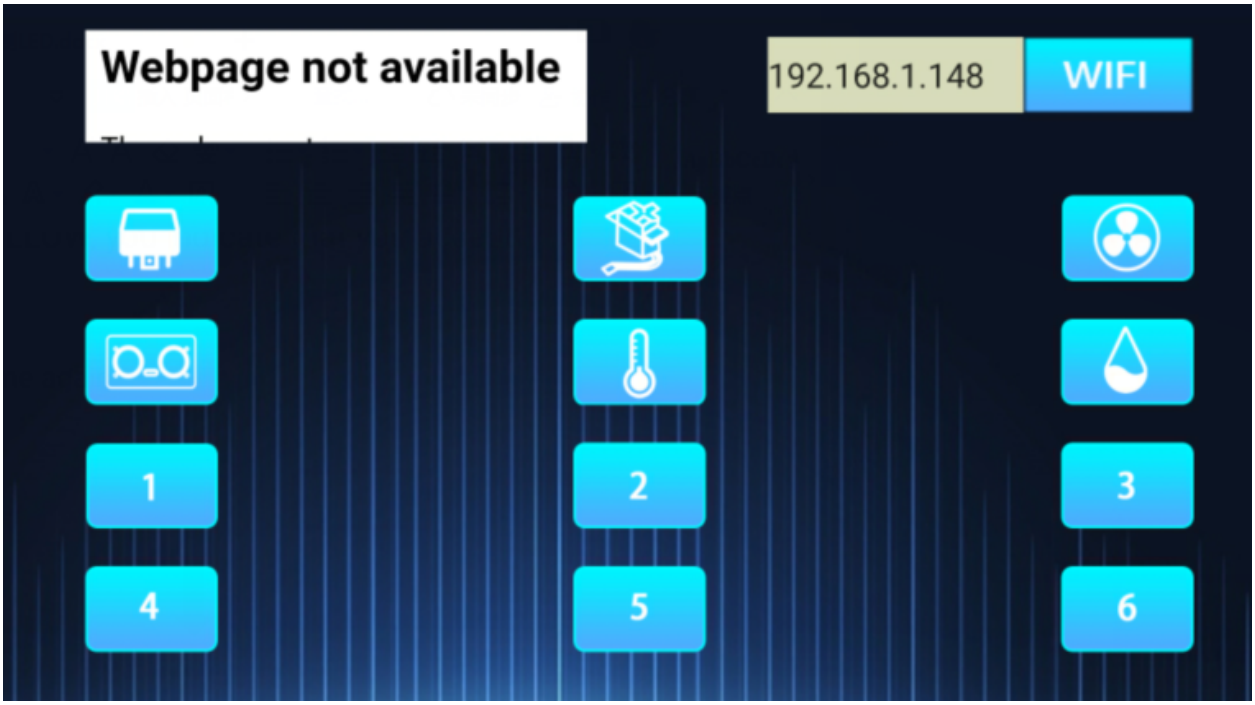
Now transfer the "keyes wifi.apk" file from the folder to your Android phone or iPad, click the "keyes wifi.apk" file to enter the installation page. Click the "**ALLOW**" button, and then click the "**INSTALL**" button. Click the "**Open**" button to enter the APP interface after the installation is completed.




keyes wifi
Allow Files to install apps?
Downloading apps from external sources may put your device and personal data at greater risk. By touching ALLOW, you indicate that you accept these risks.
Don't ask me again
DENY
ALLOW


keyes wifi
Installation source: Files
Note
This app has not undergone AppGallery compatibility and security testing. Install with caution.
Use AppGallery to download apps that have passed rigours security checks.
SEARCH IN APPGALLERY
INSTALL
CANCEL


keyes wifi
Installation source: Files
Installation successful
DONE
OPEN



Enter the detected WiFi IP address in the text box in front of the WiFi button (For example, the IP address detected by the serial monitor above is 192.168.1.123), then click the WiFi button, "Webpage not available" will become "192.168.1.123". This shows that the App has been connected to WiFi.

App for IOS system devices (mobile phone/iPad)

Open App Store.

Enter “keyes wifi” in the search box, search and the download screen will appear. Click“ ”, you can download and install keyes wifi APP. The following operations are similar to those of Android system. You can refer to the steps of Android system above for operation.

Note: Click the button on APP, the blue light on ESP8266 serial WiFi ESP-01 module will flash, indicating that APP has connected to WiFi.

After the APP has been connected to the WiFi, start the following operations. Click  .Relay opens and the APP shows `turn on the relay` , the indicator light on the module lights up. Click  again, relay is closed and the APP shows `turn off the relay` , indicator light on the module goes out.

Click  . The servo rotates 180° and the APP shows `Bring the steering gear over 180 degrees` .

Click  again, the APP shows `Bring the steering gear over 0 degrees` the servo rotates 0°.

Click  , ultrasonic sensor measures distance. Put an object in front of the ultrasonic sensor, and the APP shows `hgDistance = 14` Different distance shows different numbers). It means that the distance of the object from the ultrasonic sensor is 14cm at this time. Click  . Turn off ultrasound, theAPP shows `turn off the ultrasonic` .

Click  ,temperature and humidity sensor measures the temperature in the environment. The APP shows `jiTemperature = 28.00` , which means that the temperature in the environment is 28℃ at this time.

Click  , turn off the temperature and humidity sensor, the APP shows `turn off the temperature`.

Click  , temperature and humidity sensor measures the humidity in the environment. The APP shows `Humidity = 52.00`, it means that the humidity in the environment is 52% at this time. Click  turn off the temperature and humidity sensor, the APP shows `turn off the humidity`.