Welcome

Thank you for choosing Freenove products!

How to Start

When reading this, you should have downloaded the ZIP file for this product.
Unzip it and you will get a folder containing tutorials and related files. Please start with this PDF tutorial.

! Unzip the ZIP file instead of opening the file in the ZIP file directly.
! Do not move, delete or rename files in the folder just unzipped.

Unpack

Before taking out all the parts, please read the file “Unpack.pdf”.

Get Support

Encounter problems? Don't worry! Refer to “TroubleShooting.pdf” or contact us.

When there are packaging damage, quality problems, questions encountering in use, etc., just send us an email. We will reply to you within one working day and provide a solution.

support@freenove.com

Attention

Pay attention to safety when using and storing this product:

- This product is not suitable for children under 12 years of age because of small parts and sharp parts.
- Minors should use this product under the supervision and guidance of adults.
- This product contains small and sharp parts. Do not swallow, prick and scratch to avoid injury.
- This product contains conductive parts. Do not hold them to touch power supply and other circuits.
- To avoid personal injury, do not touch parts rotating or moving while working.
- The wrong operation may cause overheat. Do not touch and disconnect the power supply immediately.
- Operate in accordance with the requirements of the tutorial. Fail to do so may damage the parts.
- Store this product in a dry and dark environment. Keep away from children.
- Turn off the power of the circuit before leaving.

About

Freenove provides open source electronic products and services.

Freenove is committed to helping customers learn programming and electronic knowledge, quickly implement product prototypes, realize their creativity and launch innovative products. Our services include:

- Kits for learning programming and electronics
- Kits compatible with Arduino®, Raspberry Pi®, micro:bit®, etc.
- Kits for robots, smart cars, drones, etc.
- Components, modules and tools
- Design and customization

To learn more about us or get our latest information, please visit our website:

http://www.freenove.com

Copyright

Contents

Preface

If you want to make some interesting projects or want to learn electronics and programming, this document will greatly help you.

Projects in this document usually contains two parts: the circuit and the code. No experience at all? Don't worry, this document will show you how to start from scratch.

If you encounter any problems, please feel free to send us an email, we will try our best to help you.

Support email: support@freenove.com

To complete these projects, you need to use a control board and software to program it, as well as some commonly used components.

Control Board

The control board is the core of a circuit. After programming, it can be used to control other components in the circuit to achieve intended functions.
There are multiple versions of Freenove control board. Your purchase may be one of the following:

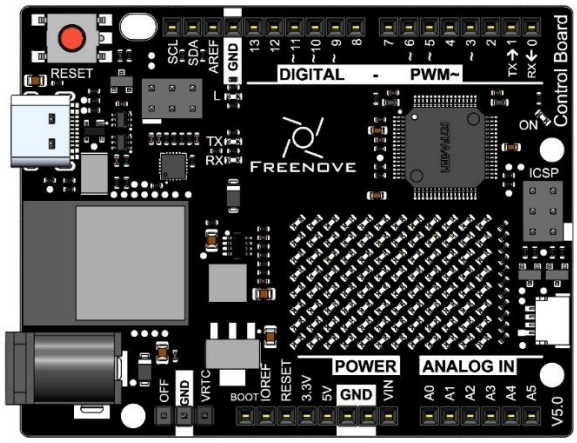
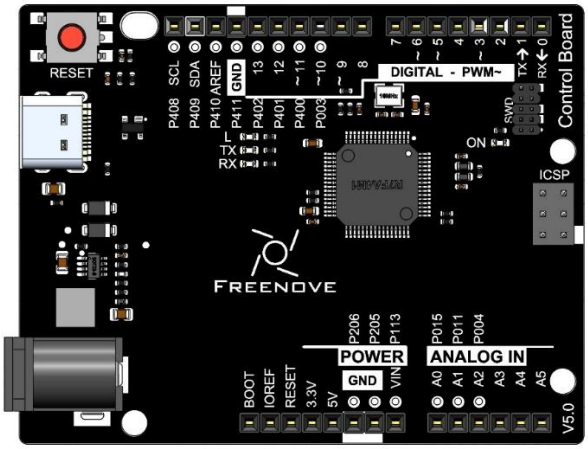
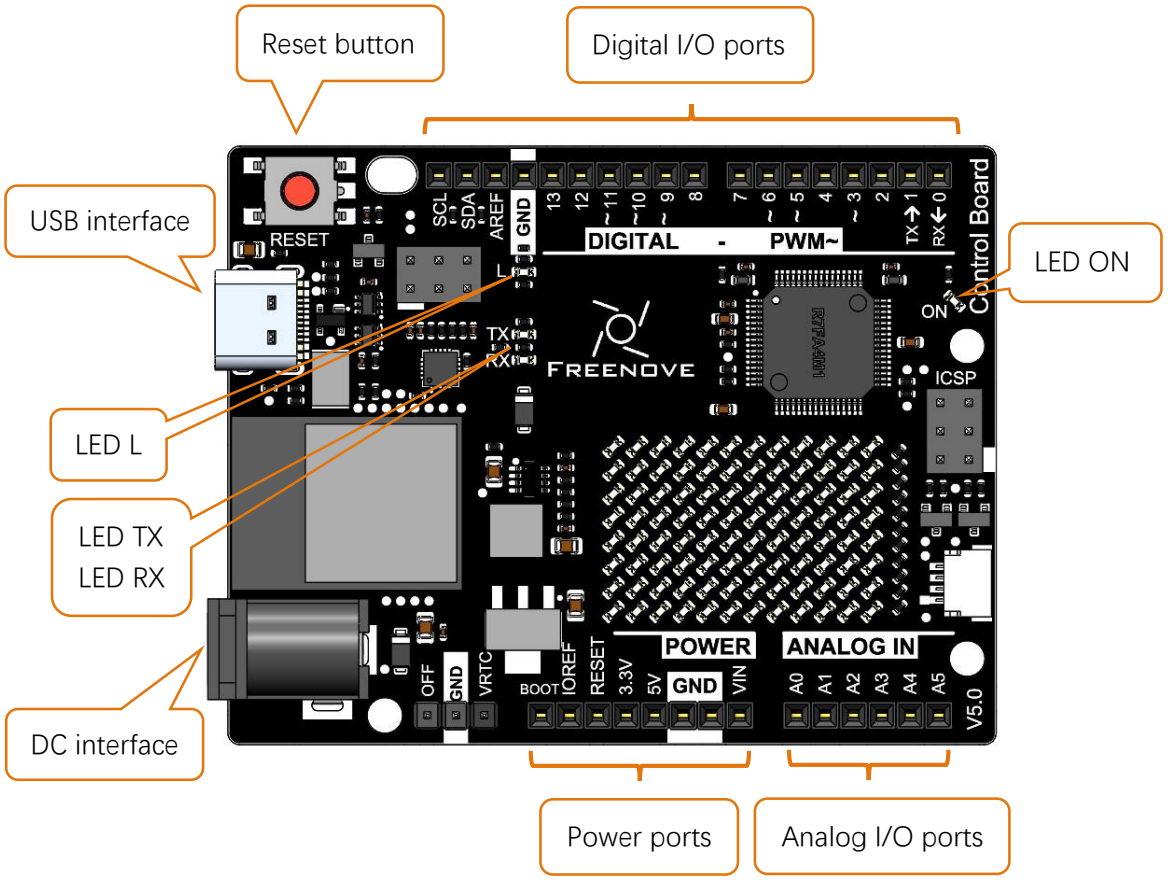
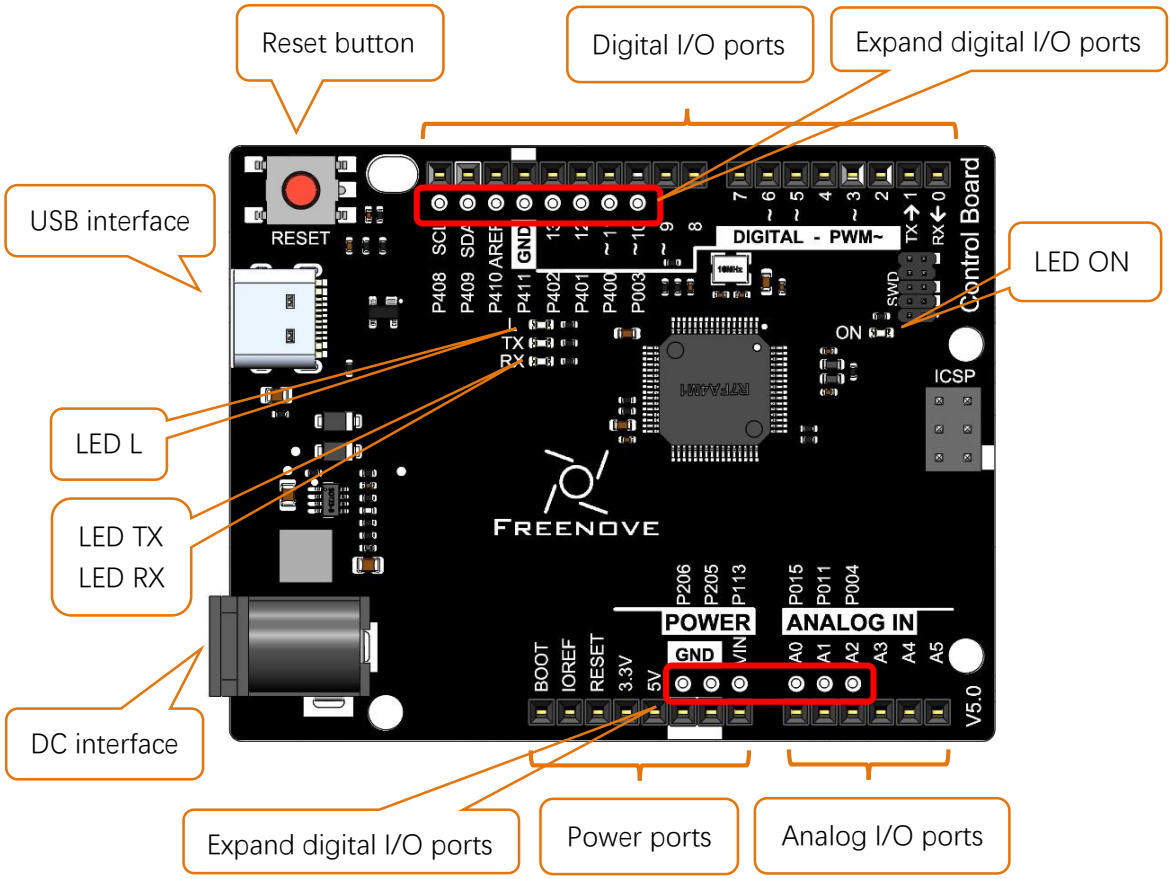
Freenove Control Board (WiFi)	Freenove Control Board (minina)
	

Diagram of the Freenove control board is shown below:





- Digital I/O ports is used to connect to other components or modules, to receive an input signal, or to send a control signal. Usually, we name it by adding a "D" in front of the number, such as D13 (pin 13).
- USB interface is used to provide power, upload code or communicate with PC.
- LED L is connected to digital I/O port 13 (pin 13).
- LED TX, RX is used to indicate the state of the serial communication.
- DC interface is connected DC power to provide power for the board.
- Power ports can provide power for electronic components and modules.
- Analog I/O ports can be used to measure analog signals.
- LED ON is used to indicate the power state.

Chapter 0 Getting Ready (Important)

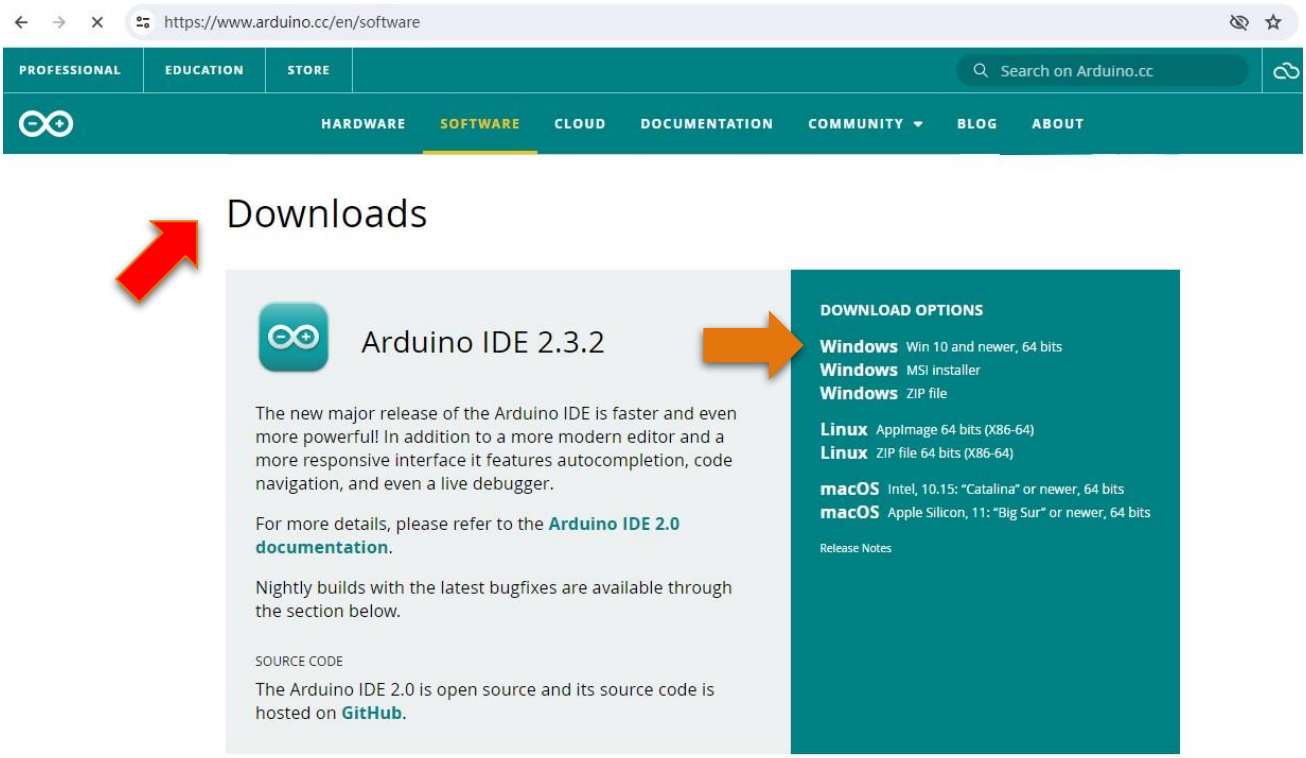
Before starting building the projects, you need to make some preparation first, which is so crucial that you must not skip.

Programming Software

We use Arduino® IDE to write and upload code for the control board, which is a free and open source. (Arduino® is a trademark of Arduino LLC.)

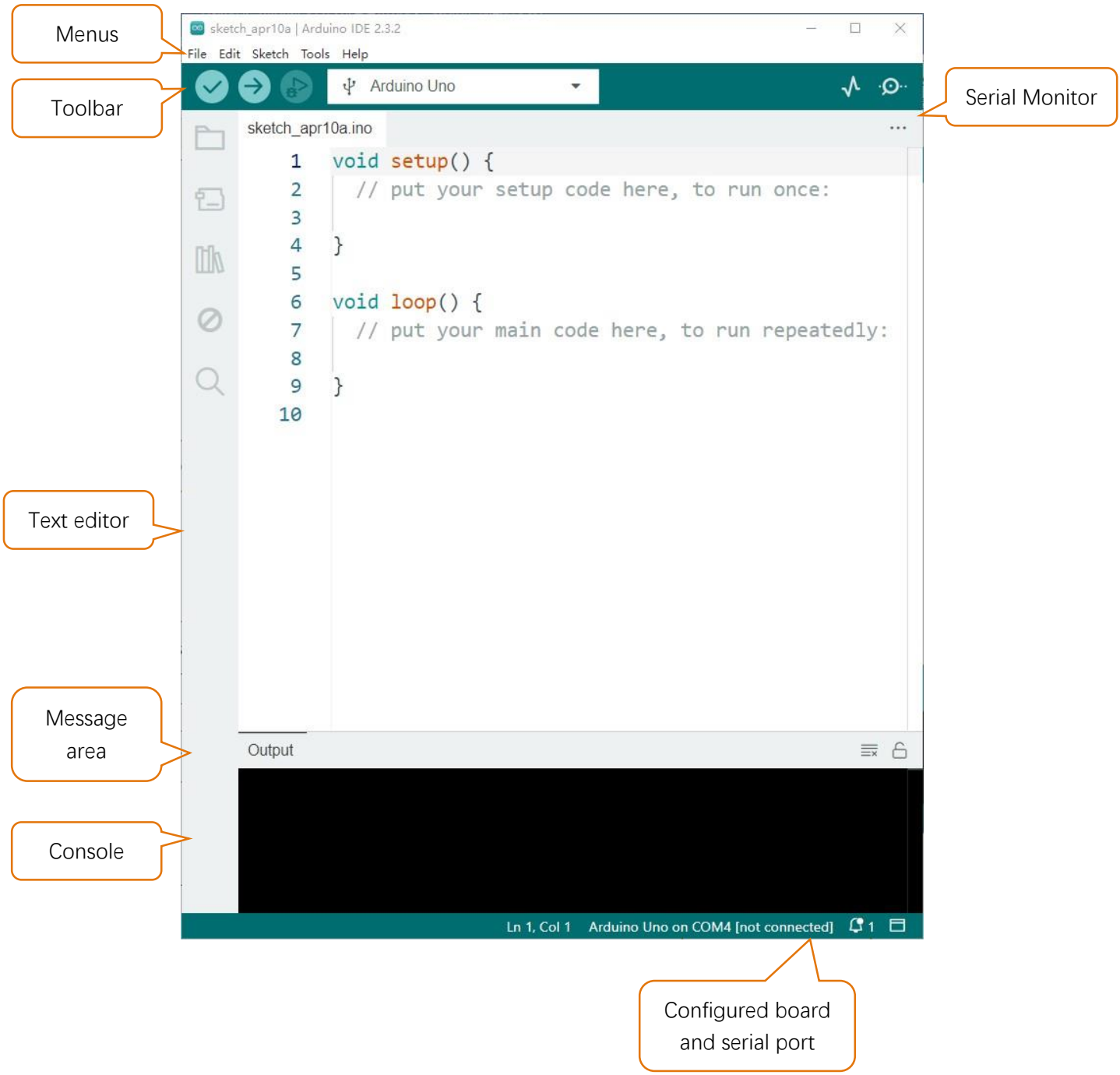
Arduino IDE uses C/C++ programming language. Don't worry if you have never used it, because this document contains programming knowledge and detailed explanation of the code.

First, install Arduino IDE. Visit https://www.arduino.cc/en/software. Scroll down and find **Arduino IDE (2.3.X)**. Then select and download corresponding installer according to your operating system. If you are a windows user, please select the "Windows Installer".



After the downloading completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it is popped up, please allow the installation. After installation is completed, an shortcut will be generated in the desktop.

Run it. The interface of the software is as follows:



Programs written with Arduino IDE are called **sketches**. These sketches are written in a text editor and are saved with the file extension**.ino**. The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino IDE, including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

Verify
Checks your code for errors compiling it.

Upload
Compiles your code and uploads it to the configured board.

New
Creates a new sketch.

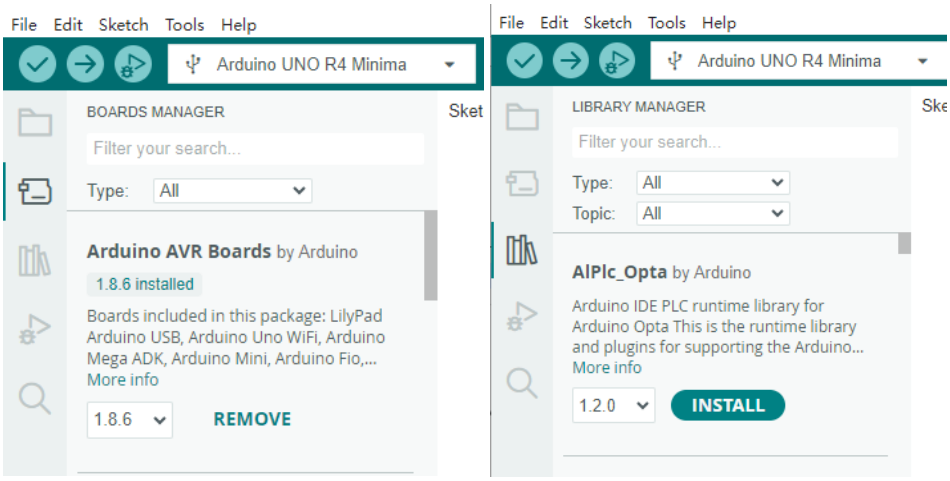
Open
Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Save
Saves your sketch.

Serial Monitor
Opens the serial monitor.

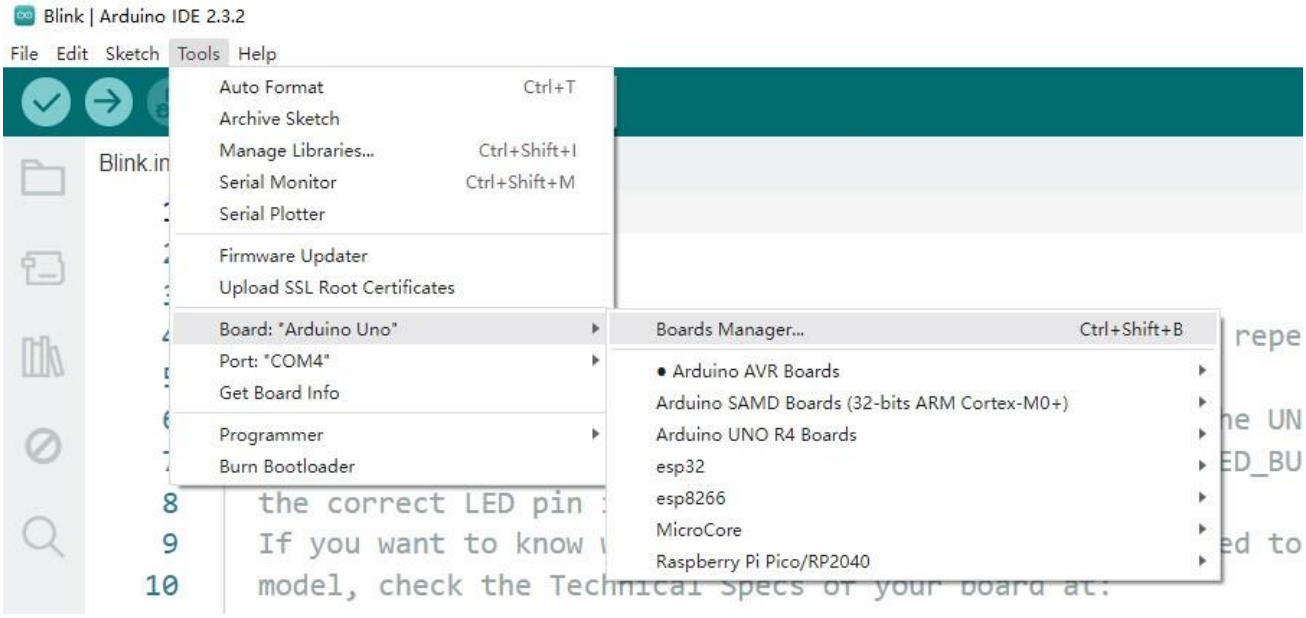
Additional commands are found within five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

In addition, the Arduino 2.X.X version provides a quick search bar for control board management and a quick search bar for libraries, so you can easily install the library files and control board support package you want.

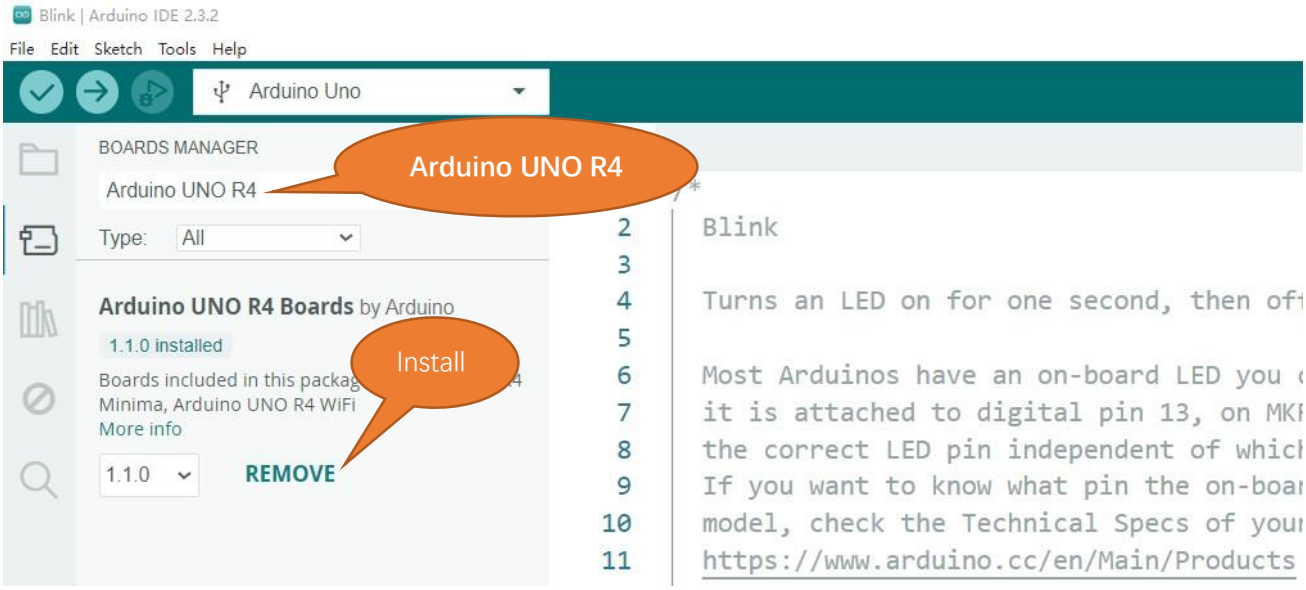


Installation of Development Board Support Package

1, Open Arduino IDE. Click Tools>**Board**>**Boards Manager...**on the menu bar.



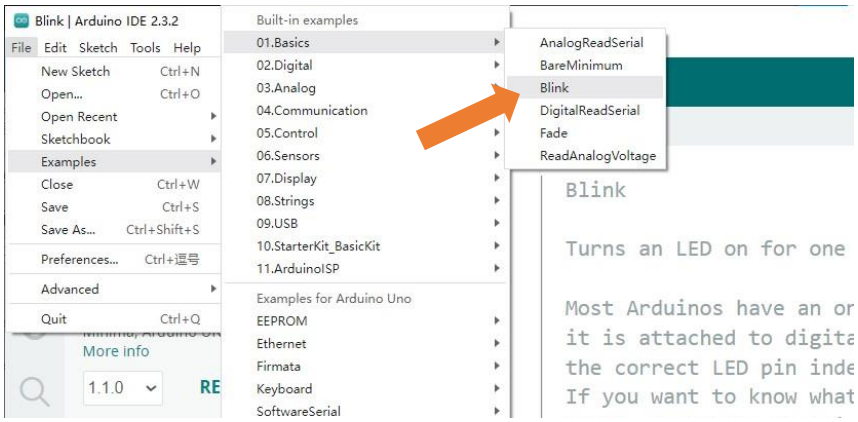
Enter **Arduino UNO R4** in the searching box, and select "**Arduino UNO R4**" and click on Install.



Click Yes in the pop-up "**dpinst-amd64.exe**"installation window. (Without it, you will fail to communicate with Arduino.) Thus far, we have finished installing the development support package.

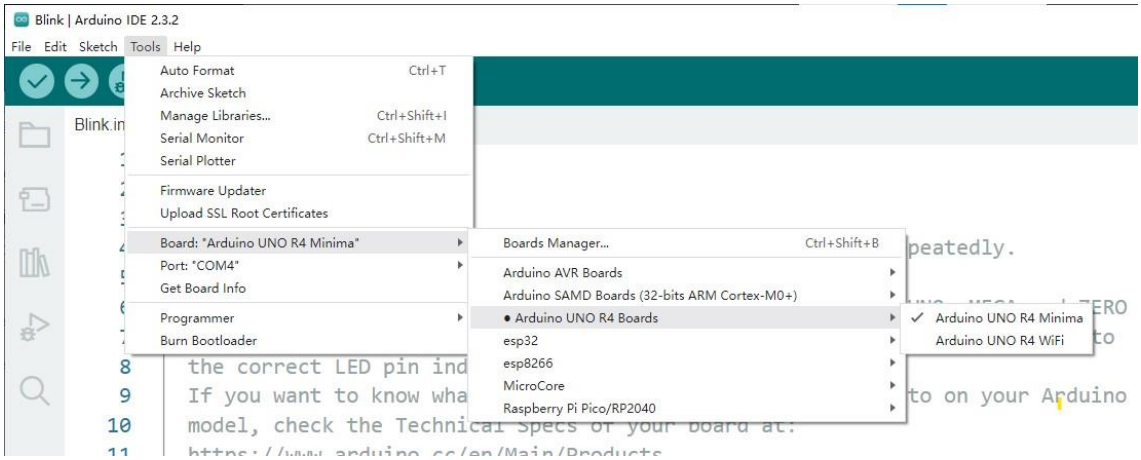
First Use

Open the example sketch "Blink".

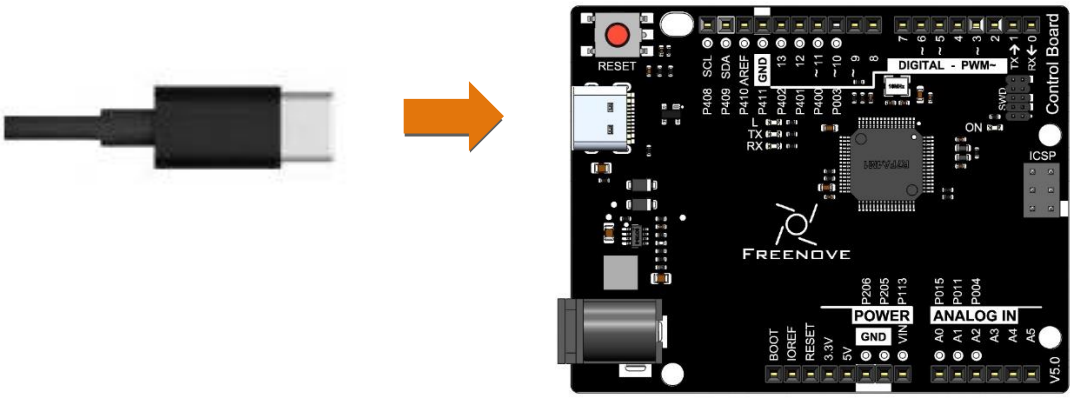


Select the board corresponding to the board you have in hands. Here we take Freenove Control Board (minina) as an example:

Select board "Arduino Uno R4 Minima". (Freenove control board is compatible with this board.)



Connect control board to your computer with USB cable.



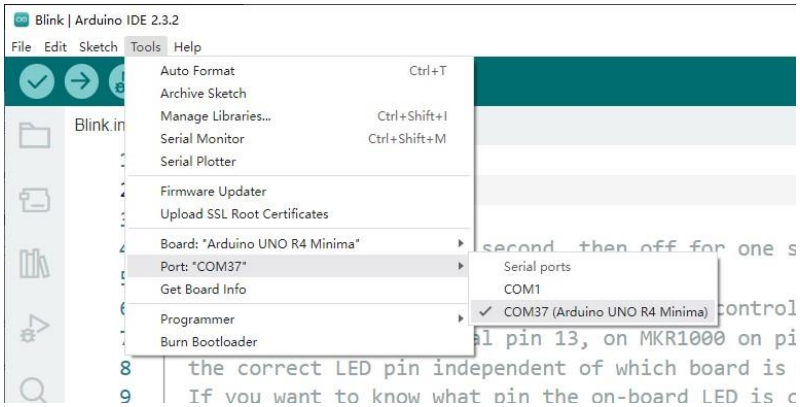
Select the port.

Note: Your port may be different from the following figure.

On Windows: It may be COM4, COM5 (Arduino Uno R4 Minima) or something like that.

On Mac: It may be /dev/cu.usbserial-710, /dev/cu.usemodem7101 (Arduino Uno R4 Minima) or something like that.

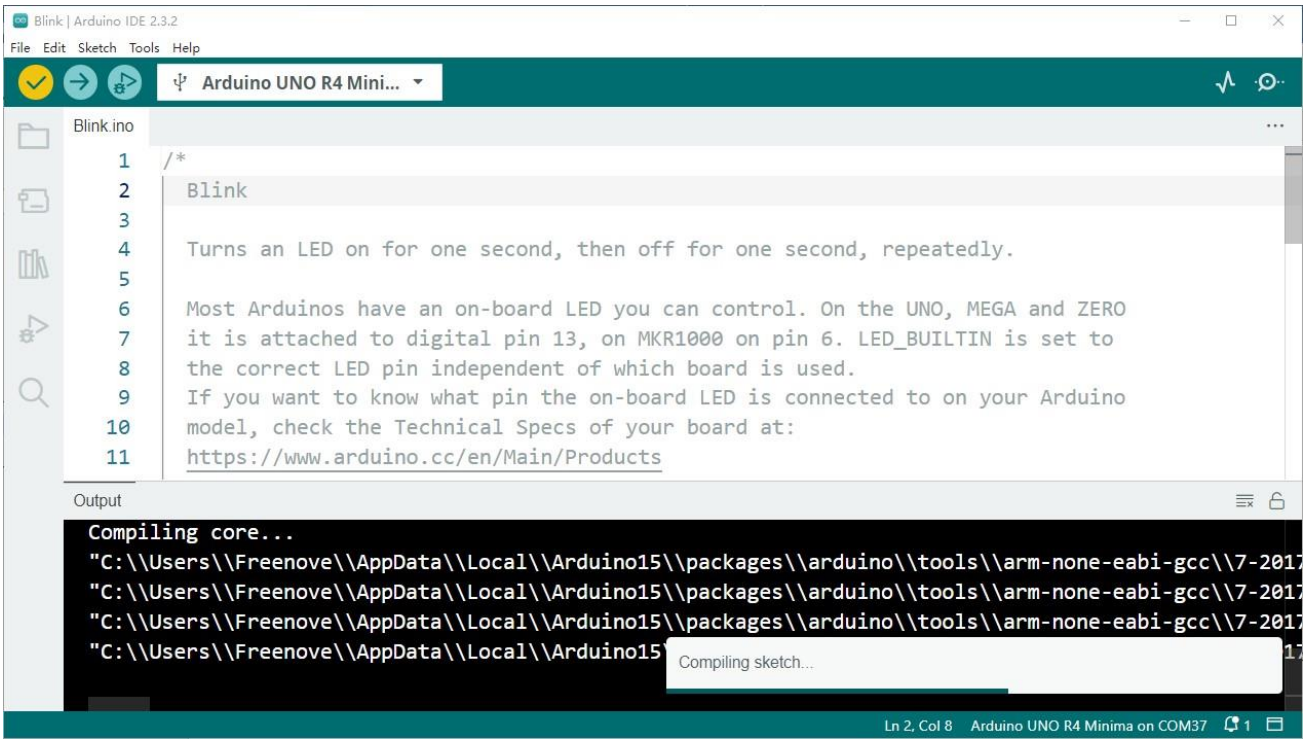
On Linux: It may be /dev/ttyUSB0, /dev/ttyACM0 or something like that.



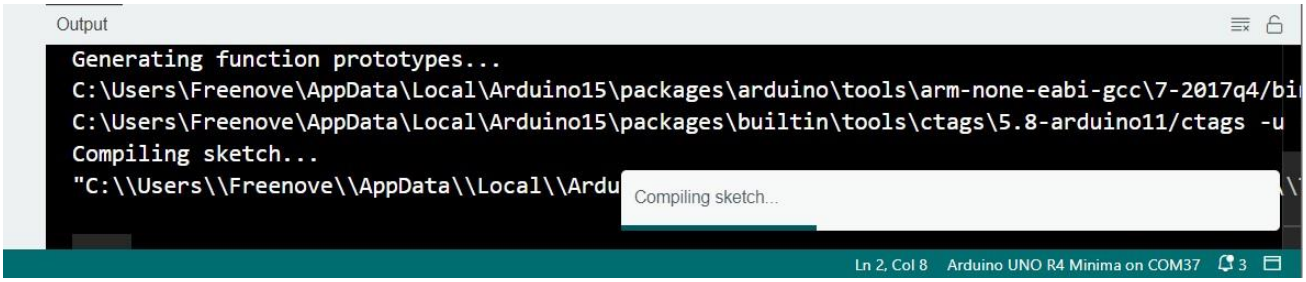
Note: If there is more than one port and you cannot decide which one to choose, disconnect the USB cable and check the port. Then connect the USB cable and check the port again. The new one is the correct port.

Having problems? Contact us for help! Send mail to: support@freenove.com

Click "Verify" button.

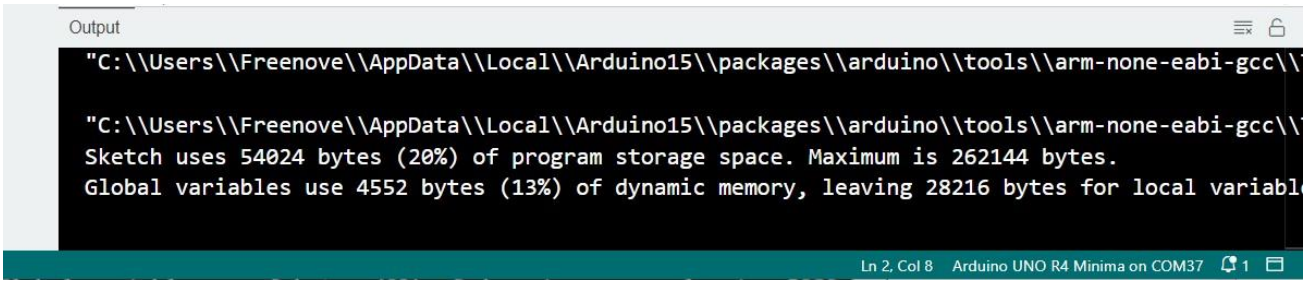


The following figure shows the code being compiled.



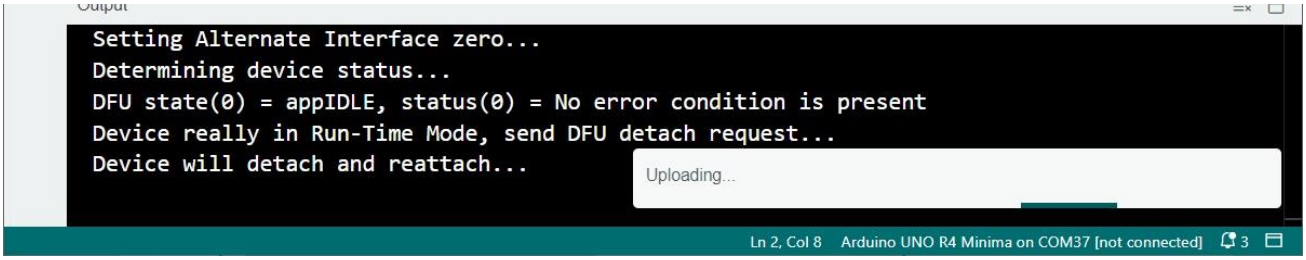
Wait a moment for the compiling to be completed. Figure below shows the code size and percentage of

space occupation. If there is an error in the code, the compilation will fail and the details are shown here.

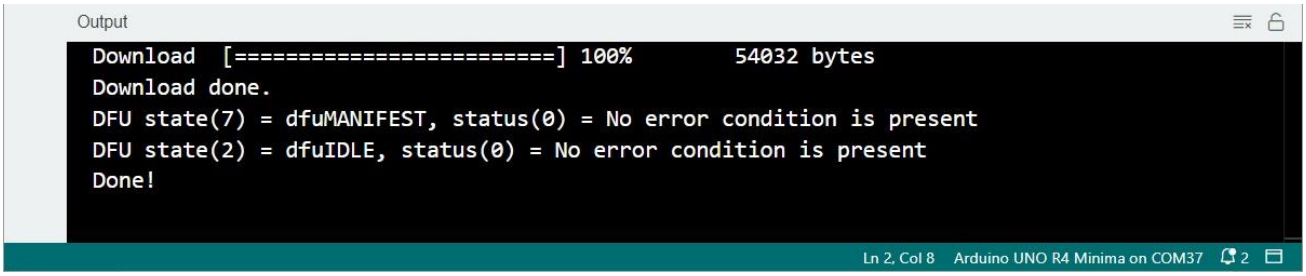


Click "Upload" button.

Figure below shows code are uploading.

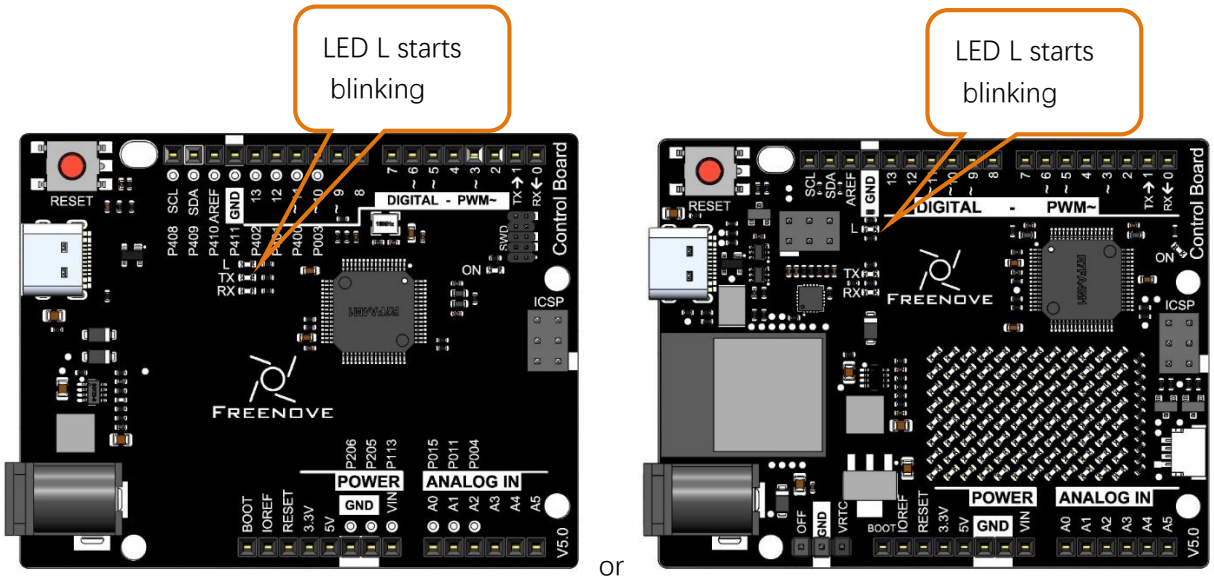


Wait a moment, and then the uploading is completed.



Having problems? Contact us for help! Send mail to: support@freenove.com

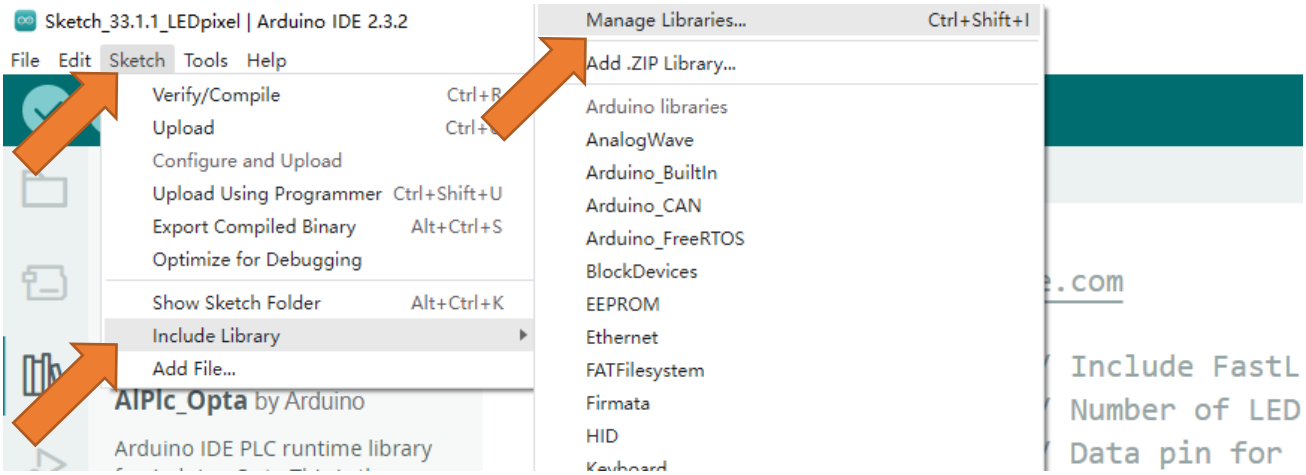
After that, we will see the LED marked with "L" on the control board start blinking. It indicates that the code is running now!



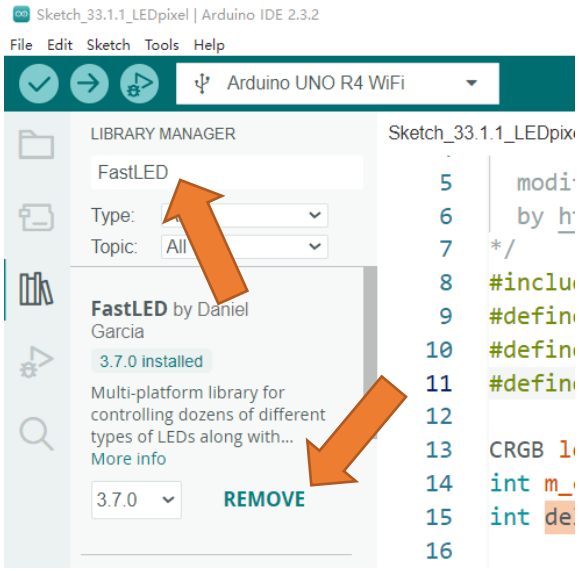
So far, we have completed the first use. I believe you have felt the joy of it. Next, we will carry out a series of projects, from easy to difficult, taking you to learn programming and the building of electronic circuit.

How to install the library

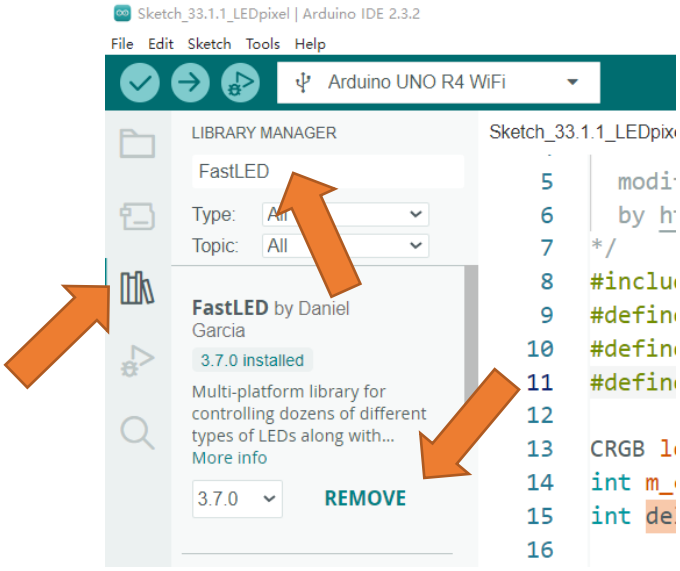
There are two ways to include libraries on Arduino IDE.
The first way, open the Arduino IDE, click Tools →Manager Libraries.



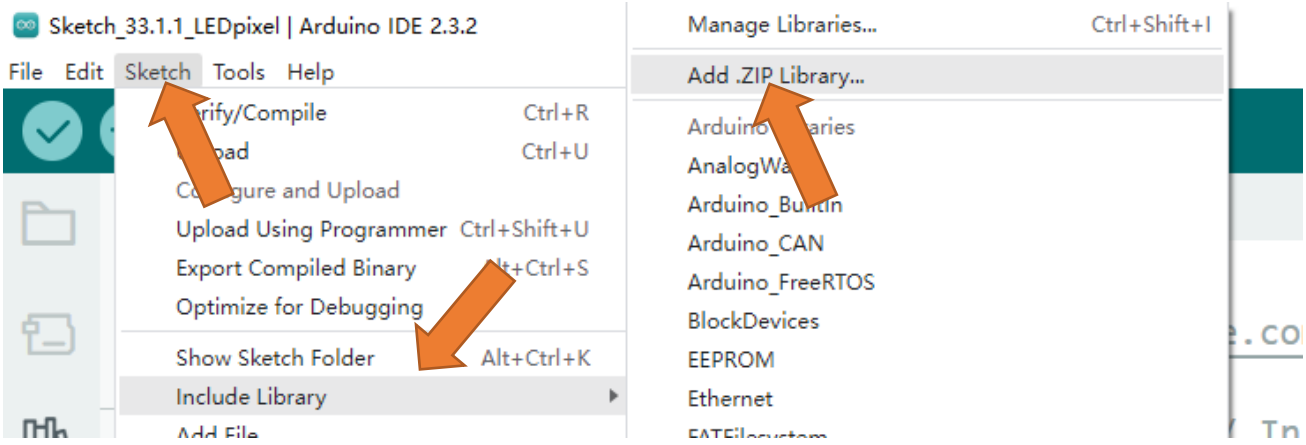
Here we take installing the “FastLED” library as an example. In the pop-up window, Library Manager, search for the name of the Library, “FastLED”. Then click Install.



Or, you can click the Library icon on the left of Arduino IDE, and type in ''FastLED' on the search bar to install.



The second way, open Arduino IDE, click Sketch→Include Library→Add .ZIP Library. In the pop-up window, find the file named "./Libraries/FastLED.Zip" which locates in this directory, and click OPEN.



Chapter 1 LED Blink

We have previously tried to make the LED marked with "L" blink on the control board. Now let us use electronic components and codes to reproduce the phenomenon, and try to understand their principle.

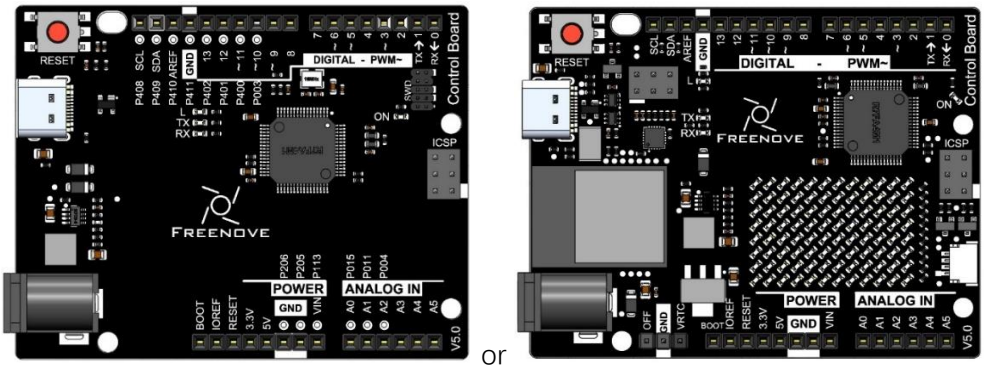
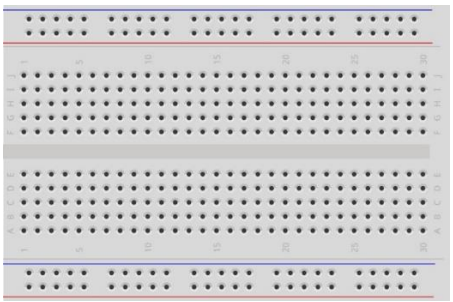
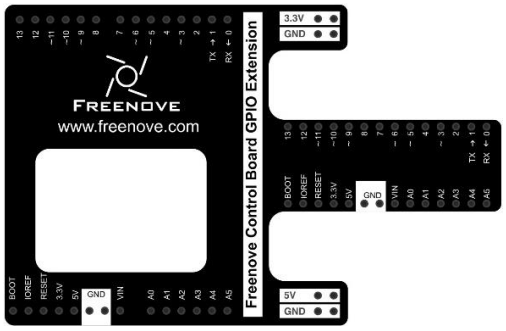
Project 1.1 Control LED with Manual Button

First, try using a push button switch to make the LED blink manually.

Component List

Note: It is worth noting that the board is compatible with the Arduino UNO R4 Minima and the Arduino UNO R4 WiFi board, and if you have one of them in hand, you can also use it for experiments in this tutorial.

Only the black control board is used to display the hardware connection in this document.

Control board x1 	
Breadboard x1 	GPIO Extension Board x1 

USB cable x1 	LED x1 	Resistor 220Ω x1 	Push button x1 
Jumper M/M x2 			

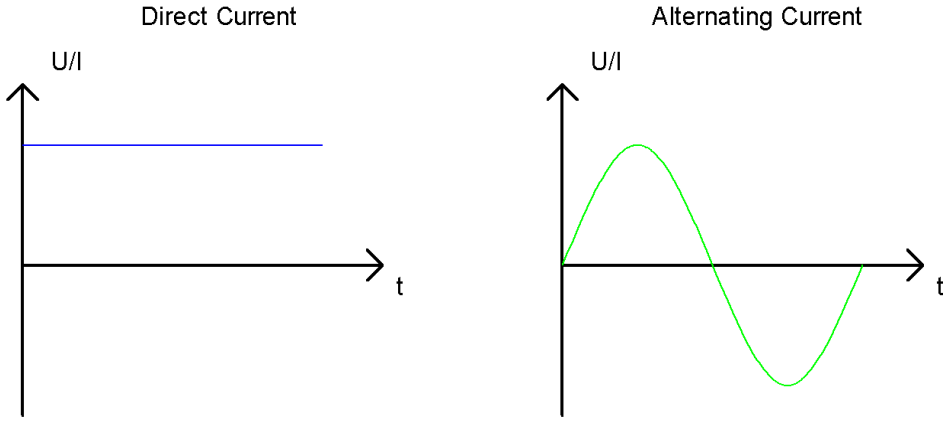
Circuit Knowledge

Power supply

Power supply provides energy for the circuit, and it is divided into DC power and AC power.

Voltage and current of DC power supply remains the same all the time, such as battery, power adapter.

Alternating Current (AC) describes the flow of charge that changes direction periodically. As a result, the voltage level also reverses along with the current. Its basic form is sinusoidal voltage(current). AC power is suitable for long-distance transmission of electric energy and it is used to supply power to houses.



Generally, electronic circuits use DC. Home appliances have rectifiers to convert AC into DC before they are used.

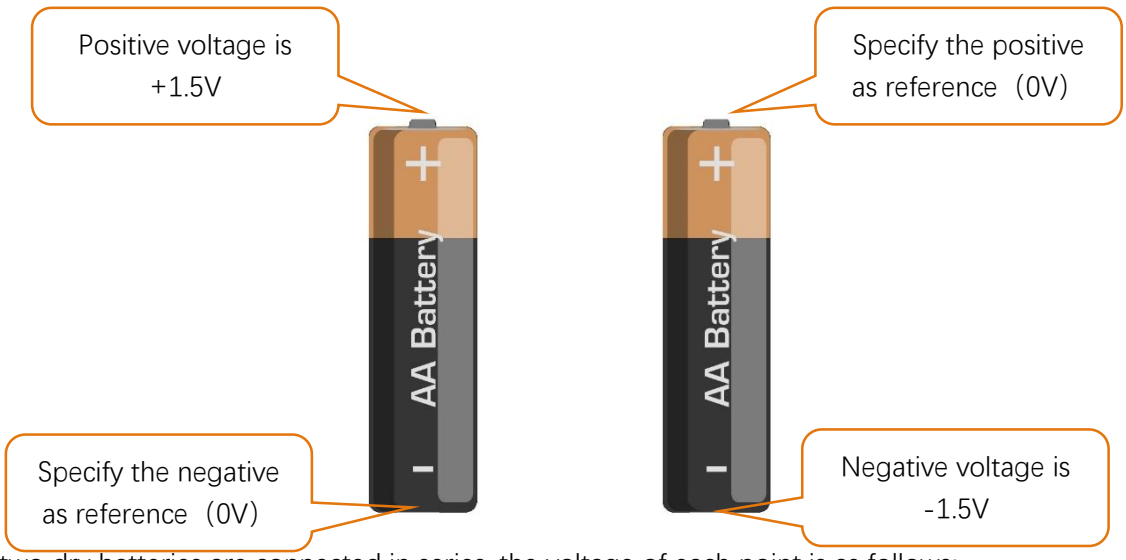
Battery or battery pack can be represented by the following symbols:

The positive and negative poles of the power supply must not be directly connected, otherwise it may scald you and cause damage to the battery.

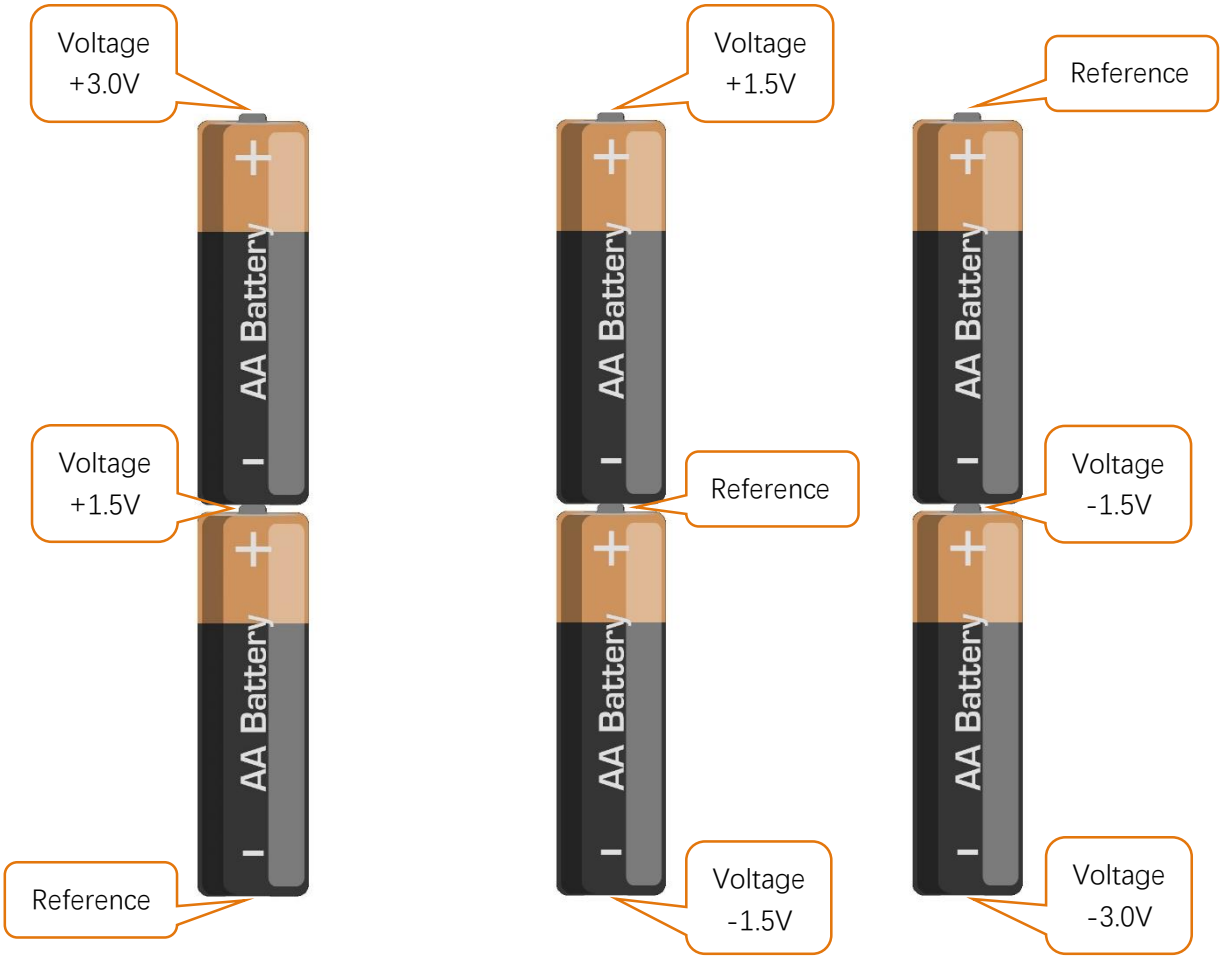
Voltage

The unit of voltage(U) is volt(V). 1kV=1000V, 1V=1000mV, 1mV=1000μv.

Voltage is relative. As to a dry battery marked with "1.5V", it's positive (+) voltage is 1.5V higher than the negative (-) voltage. If you specify the negative as reference （0V）, the positive voltage will be +1.5V.



When two dry batteries are connected in series, the voltage of each point is as follows:



In practical circuits, we usually specify negative as reference voltage (0V), which is called "Ground". The positive is usually called "VCC". The positive and negative poles of power supply is usually represented by the following two symbols:

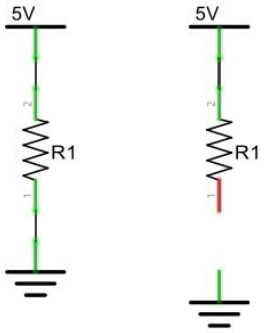
Positive(VCC) Negative(Ground, GND)

Current

The unit of current(I) is ampere(A). 1A=1000mA, 1mA=1000μA.

Closed loop consisting of electronic components is necessary for current to flow.

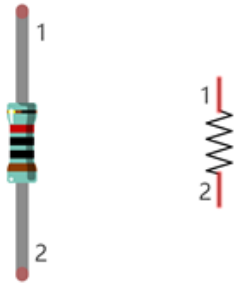
In the figures below: the left one is a loop circuit, so current flows through the circuit. The right one is not a loop circuit, so there is no current.



Resistor

Resistors use Ohms (Ω) as the unit of measurement of their resistance (R). 1MΩ=1000kΩ, 1kΩ=1000Ω.

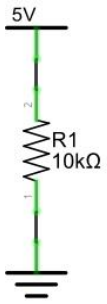
A resistor is a passive electrical component that limits or regulates the flow of current in an electronic circuit. On the left, we see a physical representation of a resistor, and the right is the symbol used to represent the presence of a resistor in a circuit diagram or schematic.



The bands of color on a resistor is a shorthand code used to identify its resistance value. For more details of resistor color codes, please refer to the card in the kit package.

With a fixed voltage, there will be less current output with greater resistance added to the circuit. The relationship between Current, Voltage and Resistance can be expressed by this formula: I=V/R known as Ohm's Law where I = Current, V = Voltage and R = Resistance. Knowing the values of any two of these allows you to solve the value of the third.

In the following diagram, the current through R1 is: I=U/R=5V/10kΩ=0.0005A=0.5mA.



WARNING: Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

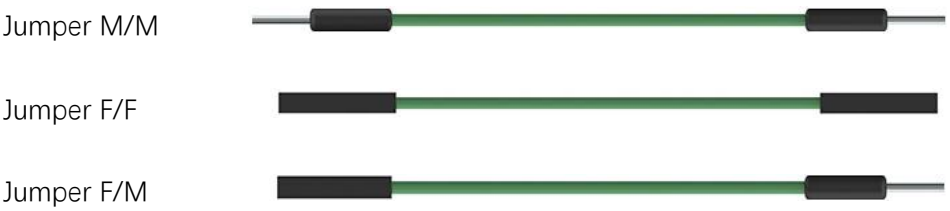
Component Knowledge

Let us learn about the basic features of components to use them better.

Jumper

Jumper is a kind of wire designed to connect the components together with its two terminals by inserting them onto breadboard or control board.

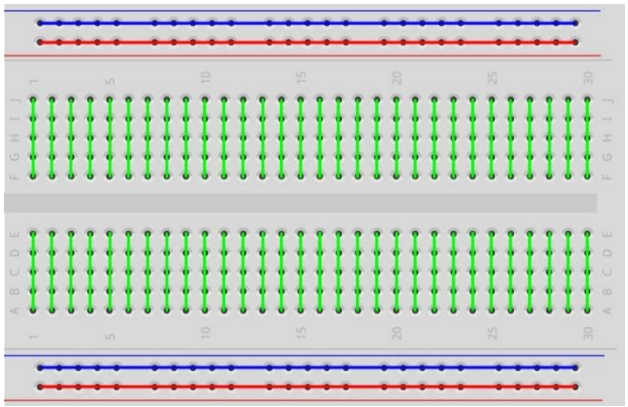
Jumpers have male end (pin) and female end (slot), so jumpers can be divided into the following 3 types.



Breadboard

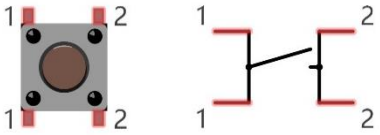
There are many small holes on breadboard to connect Jumpers.

Some small holes are connected inside breadboard. The following figure shows the inner links among those holes.



Push Button Switch

This type of Push Button Switch has 4 pins (2 Pole Switch). Two pins on the left are connected, and both left

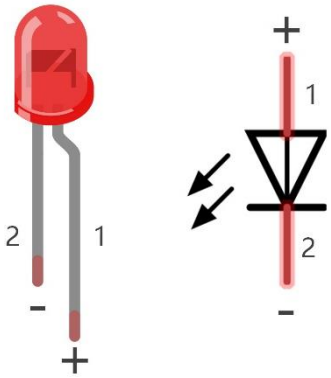
and right sides are the same per the illustration: 

When the button on the switch is pressed, the circuit is completed (your project is Powered ON).

LED

An LED is a type of diode. All diodes only work if current is flowing in the correct direction and have two Poles. An LED will only work (light up) if the longer pin (+) of LED is connected to the positive output from a power source and the shorter pin is connected to the negative (-) negative output also referred to as Ground (GND). This type of component is known as “Polar” (think One-Way Street).

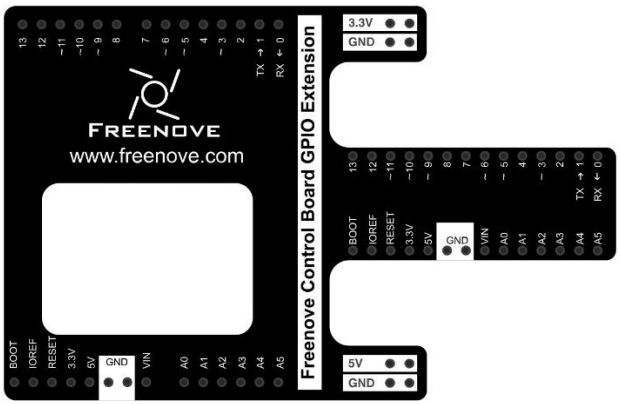
All common 2 lead diodes are the same in this respect. Diodes work only if the voltage of its positive electrode is higher than its negative electrode and there is a narrow range of operating voltage for most all common diodes of 1.9 and 3.4V. If you use much more than 3.3V the LED will be damaged and burn out.



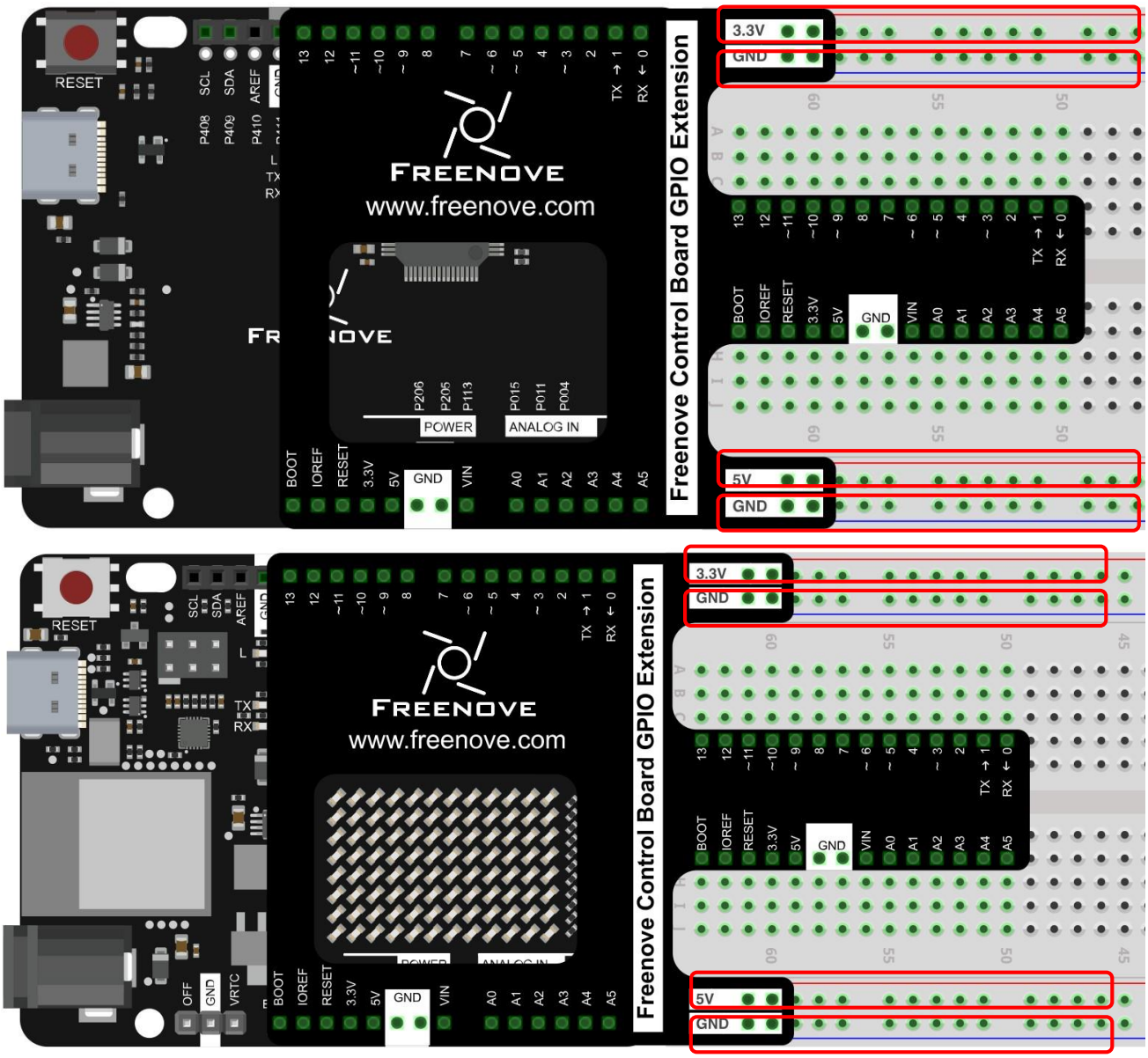
Note: LEDs cannot be directly connected to a power supply, which usually ends in a damaged component. A resistor with a specified resistance value must be connected in series to the LED you plan to use.

GPIO Extension Board

The GPIO extension board facilitates to connect the GPIOs of the microcontroller to the breadboard. The GPIO sequence on the extension board corresponds to that on the microcontroller.



Please pay attention to the outer lines of the breadboard after connecting the GPIO extension board. Now it has one row of 3.3V and one row of 5V power supply respectively. When building circuits, please be careful to avoid damaging the board due to incorrect wiring.



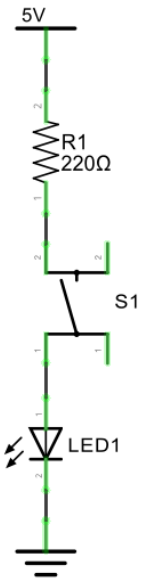
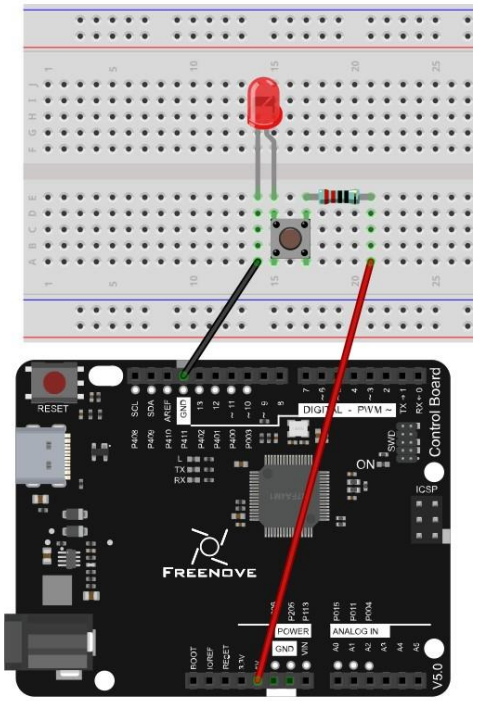
When connecting the circuit, the connection with short circuit should be avoided. If there is a short circuit, please disconnect the power supply, check the circuit again, and make sure the circuit is connected correctly before powering on the test.

Circuit

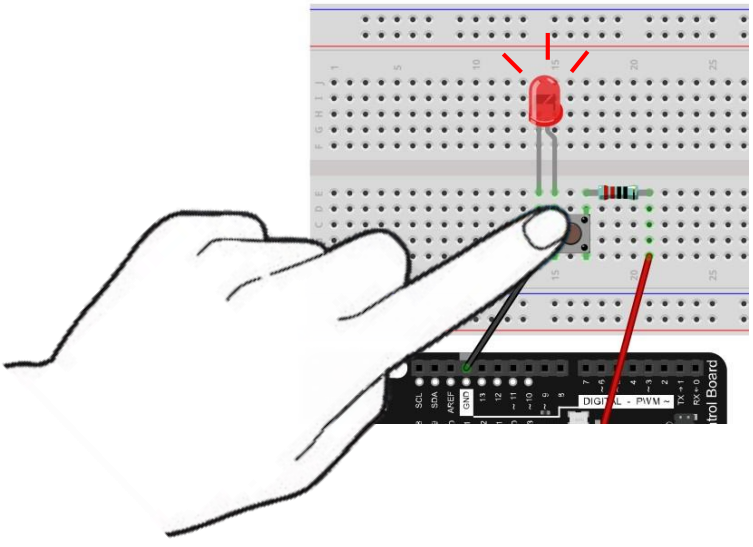
In this project, the LED is controlled by a push button switch, and the control board here only plays the role of power supply in the circuit.

Firstly, connect components with jumpers according to "hardware connection". Secondly, check the connection to confirm that there are no mistakes. Finally, connect the control board to computer with USB cable to avoid **short circuit caused by contacting the wires.**

Note: In this book, we use the regular board as an example to make the circuits. The connection is the same on the control board with WiFi function.

Schematic diagram	Hardware connection
	

LED lights up when you press the push button switch, and it lights off when you release the button.



Project 1.2 Control LED with Control Board

Now, try using control board to make LED blink through programing.

Component List

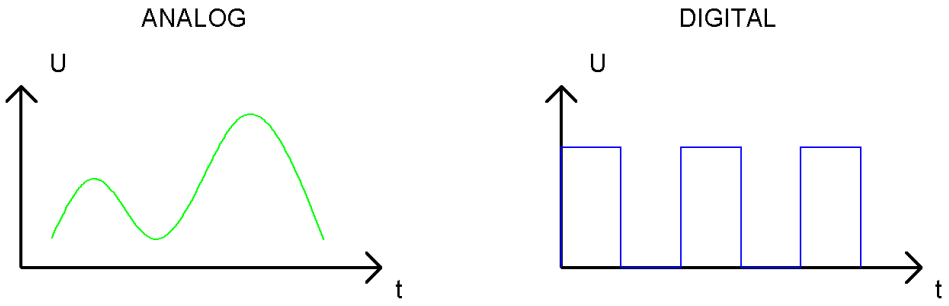
Components are basically the same with those in last section. Push button switch is no more needed.

Circuit Knowledge

Analog signal and Digital signal

An Analog Signal is a continuous signal in both time and value. On the contrary, a Digital Signal or discrete-time signal is a time series consisting of a sequence of quantities. Most signals in life are analog signals. A familiar example of an Analog Signal would be how the temperature throughout the day is continuously changing and could not suddenly change instantaneously from 0℃ to 10℃.

However, Digital Signals can instantaneously change in value. This change is expressed in numbers as 1 and 0 (the basis of binary code). Their differences can more easily be seen when compared when graphed as below.



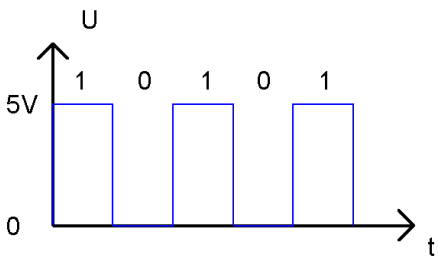
In practical applications, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1) it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

Low level and high level

In a circuit, the form of binary (0 and 1) is presented as low level and high level.

Low level is generally equal to ground voltage(0V). High level is generally equal to the operating voltage of components.

The low level of the control board is 0V and high level is 5V, as shown below. When IO port on control board outputs high level, components of small power can be directly lit, like LED.



Code Knowledge

Before start writing code, we should learn about the basic programming knowledge.

Comments

Comments are the words used to explain for the sketches, and they won't affect the running of code.
There are two ways to use comments of sketches.

1. Symbol "//"

Contents behind "//" comment out the code in a single line.

```
1  // this is a comment area in this line.
```

The content in front of "//" will not be affected.

```
1  delay(1000);              // wait for a second
```

2. Symbol "/*"and "*/"

Code can also be commented out by the contents starting with a "/*" and finishing with a "*/" and you can place it anywhere in your code, on the same line or several lines.

```
1  /* this is comment area. */
```

Or

```
1  /*
2     this is a comment line.
3     this is a comment line.
4  */
```

Data type

When programming, we often use digital, characters and other data. C language has several basic data types as follows:
int: A number that does not have a fractional part, an integer, such as 0, 12, -1;
float: A number that has a fractional part, such as 0.1, -1.2;
char: It means character, such as 'a', '@', '0';
For more about date types, please visit the website: https://www.Arduino.cc-Resources-Reference-Data Types.

Constant

A constant is a kind of data that cannot be changed, such as int type 0, 1, float type 0.1, -0.1, char type 'a', 'B'.

Variable

A variable is a kind of data that can be changed. It consists of a name, a value, and a type. Variables need to be defined before using, such as:

```
1  int i;
```

"int" indicates the type, ";" indicates the end of the statement. The statement is usually written in one single line; and these statements form the code.
After declaration of the variable, you can use it. The following is an assignment to a variable:

```
1  i = 0;                    // after the execution, the value of i is 0
```

"=" is used to pass the value of a variable or constant on the right side to the variable on the left.

A certain number of variables can be declared in one statement, and a variable can be assigned multiple times. Also, the value of a variable can be passed to other variables. For example:

```
int i, j;
i = 0;                    // after the execution, the value of i is 0
i = 1;                    // after the execution, the value of i is 1
j = i;                    // after the execution, the value of j is 1
```

Function

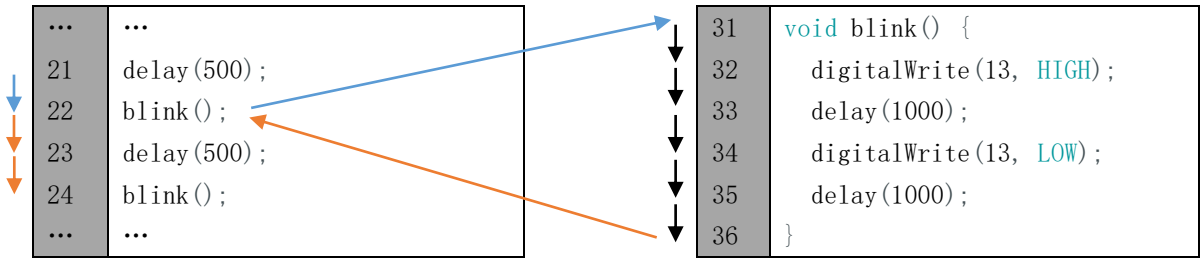
A function is a collection of statements with a sequence of order, which performs a defined task. Let's define a function void blink() as follows:

```
void blink() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

"void" indicates that the function does not return a value (Chapter 4 will detail the return value of functions);
"()" its inside is parameters of a function (Chapter 2 will detail the parameters of the functions). No content inside it indicates that this function has no parameters;
"{}" contains the entire code of the function.
After the function is defined, it is necessary to be called before it is executed. Let's call the function void blink(), as shown below.

```
blink();
```

When the code is executed to a statement calling the function, the function will be executed. After execution of the function is finished, it will go back to the statement and execute the next statement.

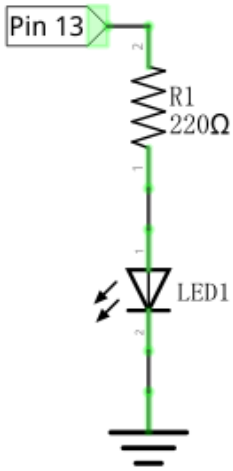
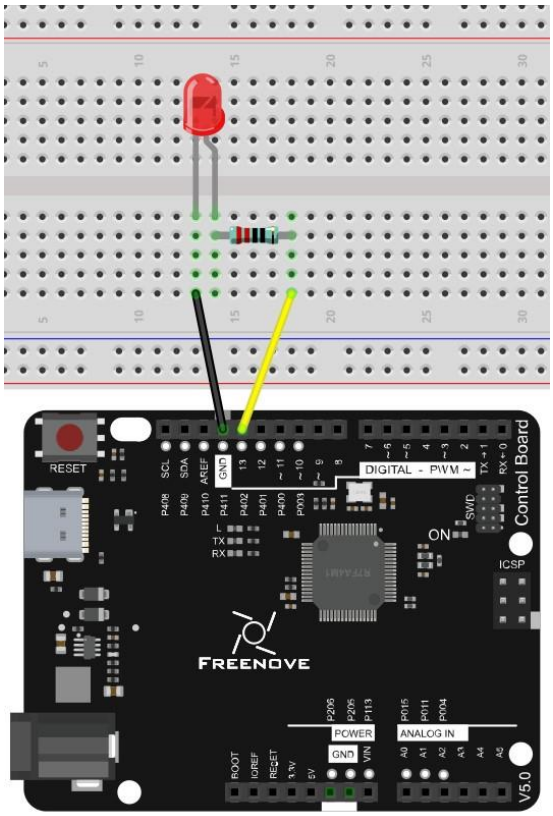


Some functions have one or more parameters. When you call such functions, you need to write parameters inside "()":

```
digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
delay(1000);              // wait for a second
```

Circuit

Now, we will use IO port of control board to provide power for the LED. Pin 13 of the control board is the digital pin. It can output high level or low level. In this way, control board can control the state of LED.

Schematic diagram	Hardware connection
	

Sketch

Sketch 1.2.1

In order to make the LED blink, we need to make pin 13 of the control board output high and low level alternately.

We highly recommend you type the code manually instead of copying and pasting, so that you can develop your coding skills and get more knowledge.

```
1  // the setup function runs once when you press reset or power the board
2  void setup() {
3    // initialize digital pin 13 as an output
4    pinMode(13, OUTPUT);
5  }
6
7  // the loop function runs over and over again forever
8  void loop() {
9    digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
10   delay(1000);              // wait for a second
11   digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
12   delay(1000);              // wait for a second
13 }
```

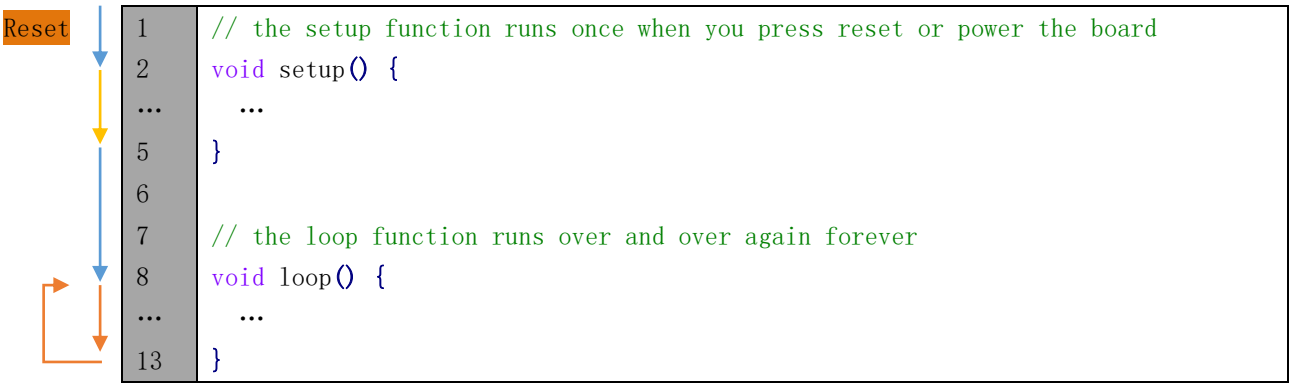
The code usually contains two basic functions: void setup() and void loop().

After control board is reset, the setup() function will be executed first, and then the loop() function will be executed.

setup() function is generally used to write code to initialize the hardware. And loop() function is used to write code to achieve certain functions. loop() function is executed repeatedly. When the execution reaches the end of loop(), it will jump to the beginning of loop() to run again.

Reset

Reset operation will lead the code to be executed from the beginning. Switching on the power, finishing uploading the code and pressing the reset button will trigger reset operation.



In the setup () function, first, we set pin 13 of the control board as output mode, which can make the port output high level or low level.

```
3   // initialize digital pin 13 as an output
4   pinMode(13, OUTPUT);
```

Then, in the loop () function, set pin 13 of the control board to output high level to make LED light up.

```
9   digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
```

Wait for 1000ms, which is 1s. delay() function is used to make control board wait for a moment before executing the next statement. The parameter indicates the number of milliseconds to wait for.

```
10  delay(1000);              // wait for a second
```

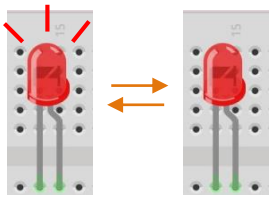
Then set the 13 pint to output low level, and LED light off. One second later, the execution of loop () function will be completed.

```
11  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
12  delay(1000);              // wait for a second
```

The loop() function is constantly being executed, so LED will keep blinking.

The functions called above are standard functions of the Arduino IDE, which have been defined in the Arduino IDE, and they can be called directly. We will introduce more common standard functions in later chapters. For more standard functions and the specific use method, please visit https://www.arduino.cc-Resources-Reference-Functions.

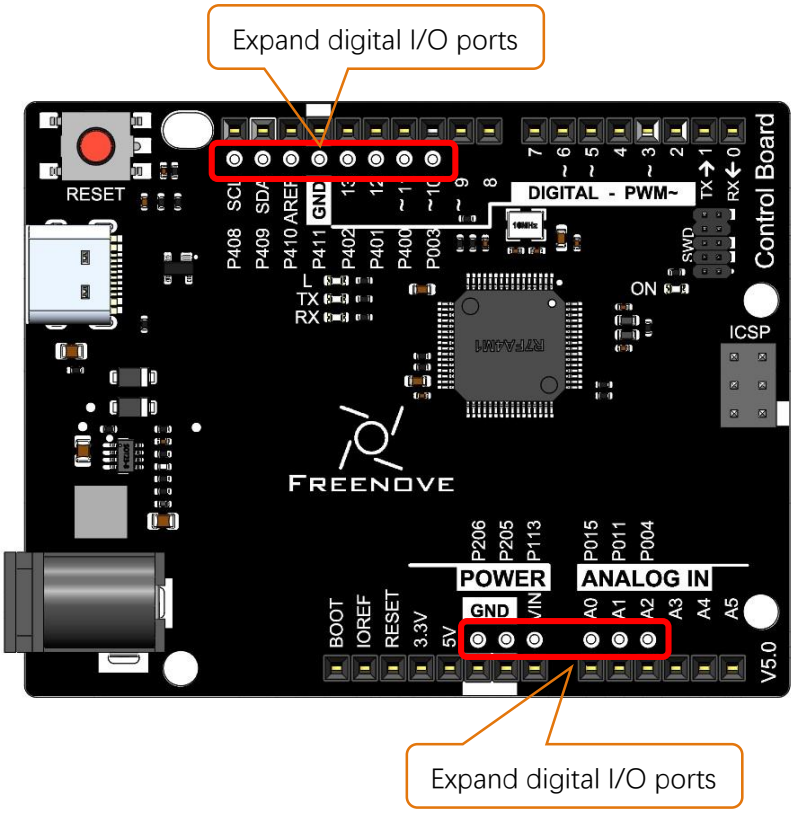
Verify and upload the code, then the LED starts blinking.



How to Use the Expanding GPIO Pins

In this section, we will learn to use the expanding GPIO pins. If you are working on the board with Bluetooth and WiFi functions, you can skip this section.

The board used in this section is as shown below:



Before using the expanding GPIO pins, please complete the following configuration first.

Copy "pins_arduino.h" and "variant.cpp" under the file path "/Libraries/Expanding_GPIO_Pins" to the file path "C:\Users\Freenove\AppData\Local\Arduino15\packages\arduino\hardware\renesas_uno\1.1.0\variants\MINIMA ".

The path may vary due to different versions of board package installed.

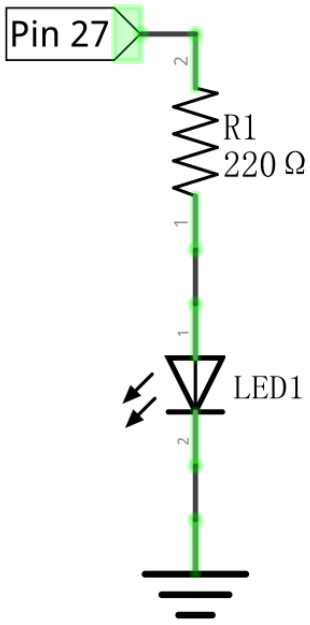
The configuration is completed onve the files are copied the files to the above path. The definition of the expanding GPIOs is as shown in the table below.

GPIOs on the control board	Definition of Arduino GPIO
P003	D27
P400	D28
P401	D29
P402	D30
P411	D31
P410	D32
P409	D33
P408	D34
P004	D35
P011	D36
P015	D37
P113	D38
P205	D39
P206	D40

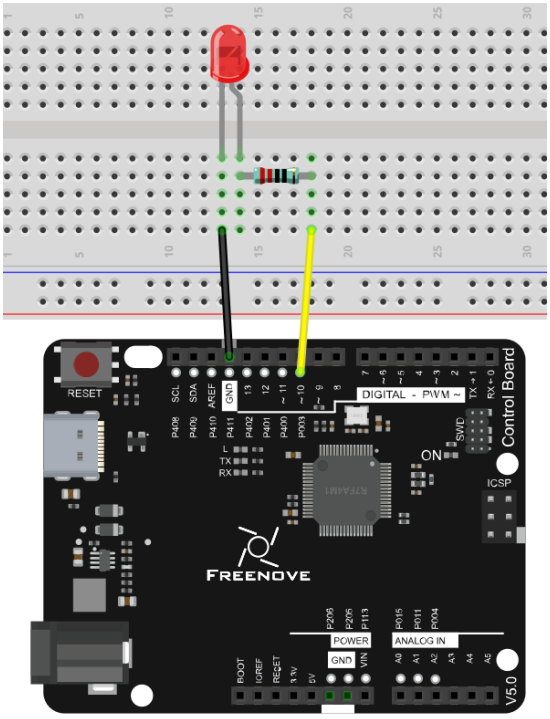
Now, let's try to use the expanding GPIO pins to light up an LED. Open the Blinking sketch, modify the control pin according to the pin definition shown in the above table. Here we take P003 on the control board as an example, so we modify the control pin of the LED to 27.

Build the circuit as below, and upload the sketch to the board.

Schematic diagram



Hardware connection



```
1  // the setup function runs once when you press reset or power the board
2  void setup() {
3    // initialize digital pin 13 as an output
4    pinMode(27, OUTPUT);
5  }
6
7  // the loop function runs over and over again forever
8  void loop() {
9    digitalWrite(27, HIGH);   // turn the LED on (HIGH is the voltage level)
10   delay(1000);              // wait for a second
11   digitalWrite(27, LOW);    // turn the LED off by making the voltage LOW
12   delay(1000);              // wait for a second
13 }
```

In this way, you can use more GPIOs to design more interesting projects. It is worth noting that all the expansion GPIOs are uniformly defined as ordinary digital I/O interfaces. We do not solder male or female headers to them. When you need to use these pins, you can solder headers yourself.

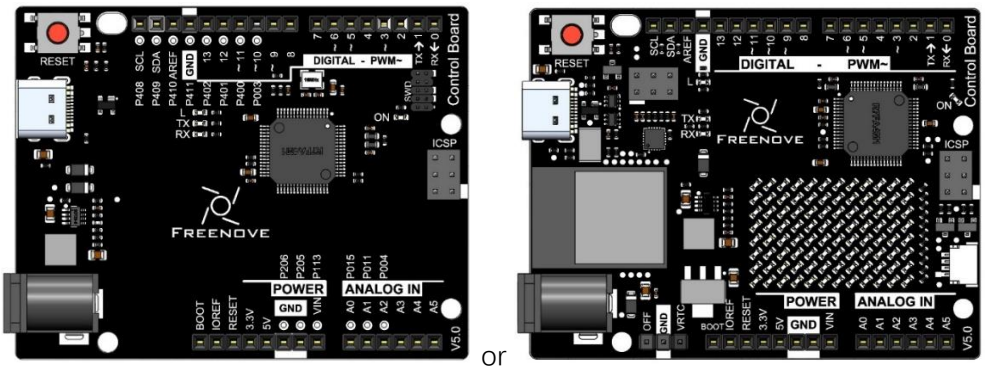
Chapter 2 Serial

In this chapter, we will get into serial communication, which is a more advanced means of communication.

Project 2.1 Send Data through Serial

We will use the serial port on control board to send data to computer.

Component List

Control board x1 
USB cable x1 

Code Knowledge

Bit and Byte

As mentioned earlier, computers use a binary signal. A binary signal is called 1 bit, and 8 bits organized in order is called 1 byte. Byte is the basic unit of information in computer storage and processing. 1 byte can represent 2^8=256 numbers, that is, 0-255. For example:

As to binary number 10010110, "0" usually presents the lowest value in code.

Sequence	7	6	5	4	3	2	1	0
Number	1	0	0	1	0	1	1	0

When a binary number need to be converted to decimal number, first, the nth number of it need be multiplied by n power of 2, then sum all multiplicative results. Take 10010110 as an example:

1*2^7+0*2^6+0*2^5+1*2^4+0*2^3+1*2^2+1*2^1+0*2^0=150

We can make a decimal number divided by 2 to convert it to binary number. Get the integer quotient for the next iteration and get the remainder for the binary digit. Repeat the steps until the quotient is equal to zero. Arrange all remainders from right to left in a line. Then we complete the conversion. For example:

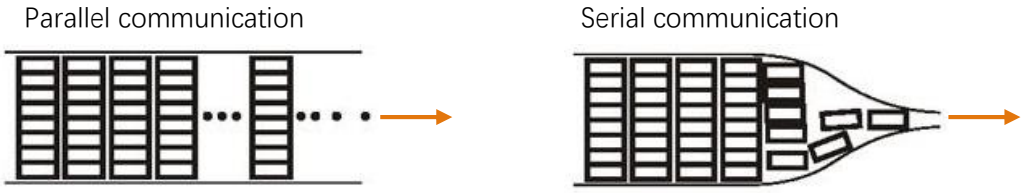
		Remainder	Sequence
2	150	0	0
2	75	1	1
2	37	1	2
2	18	0	3
2	9	1	4
2	4	0	5
2	2	0	6
2	1	1	7
	0		

The result is 10010110.

Circuit Knowledge

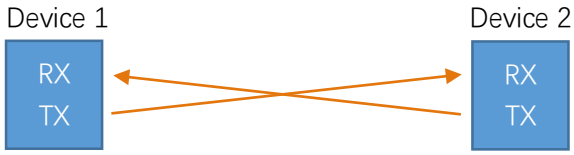
Serial and parallel communication

Serial communication uses one data cable to transfer data one bit by another in turn, while parallel communication means that the data is transmitted simultaneously on multiple cables. Serial communication takes only a few cables to exchange information between systems, which is especially suitable for computers to computer, long distance communication between computers and peripherals. Parallel communication is faster, but it requires more cables and higher cost, so it is not appropriate for long distance communication.



Serial communication

Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and the other for receiving data (RX line). The serial communication connections of two devices use is as follows:

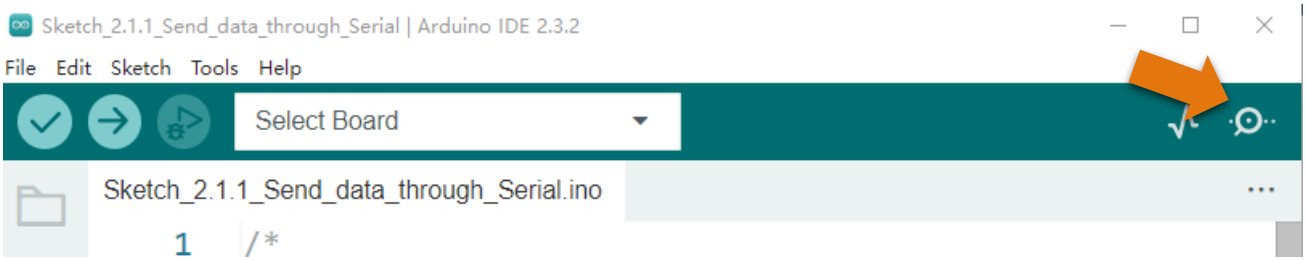


Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The baud rates commonly used are 9600 and 115200.

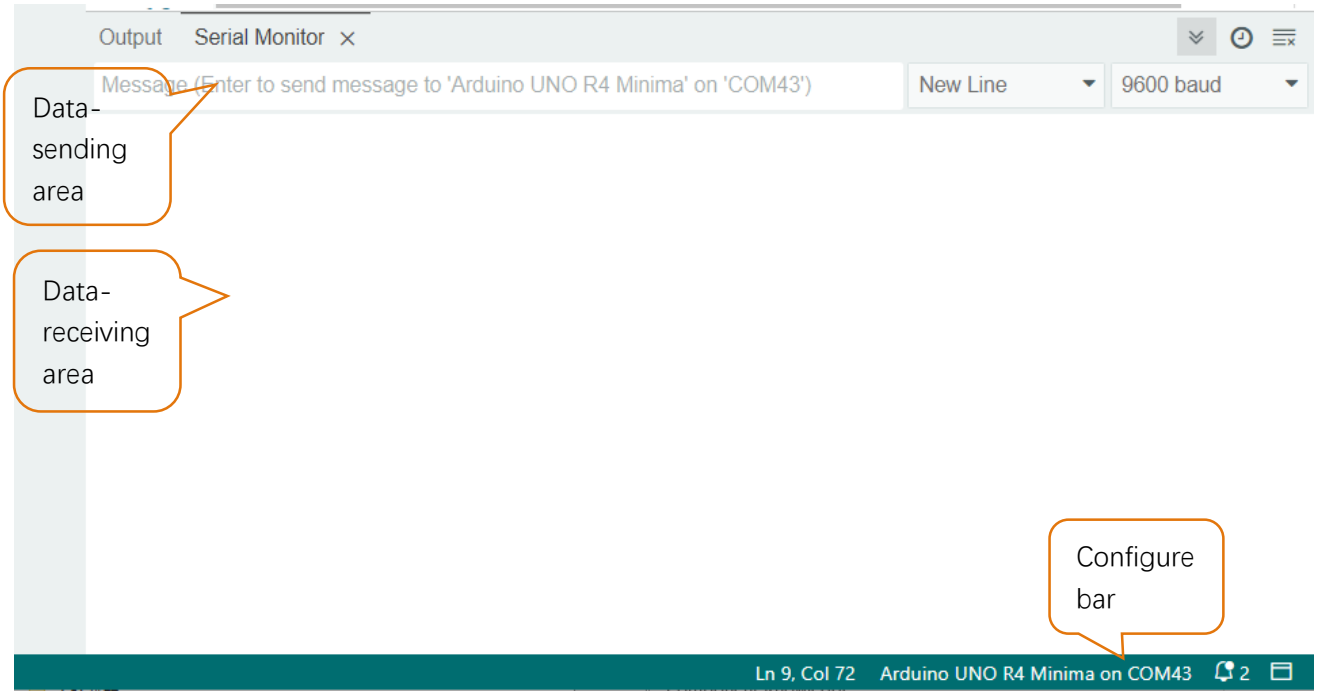
Serial port on Control board

Control board has integrated USB to serial transfer, so it can communicate with computer when USB cable get connected to it. Arduino IDE also uploads code to control board through the serial connection.

Computer identifies serial devices connected to your computer as COMx. We can use the Serial Monitor window of Arduino IDE to communicate with control board, connect control board to computer through the USB cable, choose the correct device, and then click the Serial Monitor icon to open the Serial Monitor window.

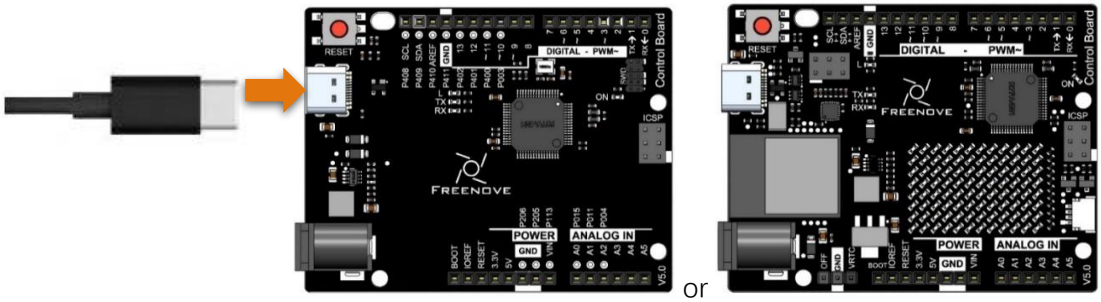


Interface of Serial Monitor window is as follows. If you can't open it, make sure control board had been connected to the computer, and choose the correct serial port in the menu bar "Tools".



Circuit

Connect control board to the computer with USB cable.



If you need any support, please feel free to contact us via: support@freenove.com

Sketch

Sketch 2.1.1

Now, write code to send some texts to the Serial Monitor window

```
1  int counter = 0;  // define a variable as a data sending to serial port
2
3  void setup() {
4    Serial.begin(9600);           // initialize the serial port, set the baud rate to 9600
5    Serial.println("UNO is ready!");    // print the string "UNO is ready!"
6  }
```

```
7
8   void loop() {
9     // print variable counter value to serial
10    Serial.print("counter:");    // print the string "counter:"
11    Serial.println(counter);    // print the variable counter value
12    delay(500);                 // wait 500ms to avoid cycling too fast
13    counter++;                  // variable counter increases 1
14  }
```

setup() function initializes the serial port.

And then continuously sends variable counter values in the loop () function.

Serial class

Class is a C++ language concept. Arduino IDE supports C++ language, which is a language extension. We don't explain specifically the concept here, but only describe how to use it. If you are interested in it, you can learn by yourself. Serial is a class name, which contains variables and functions. You can use the "." operational character to visit class variables and functions, such as:

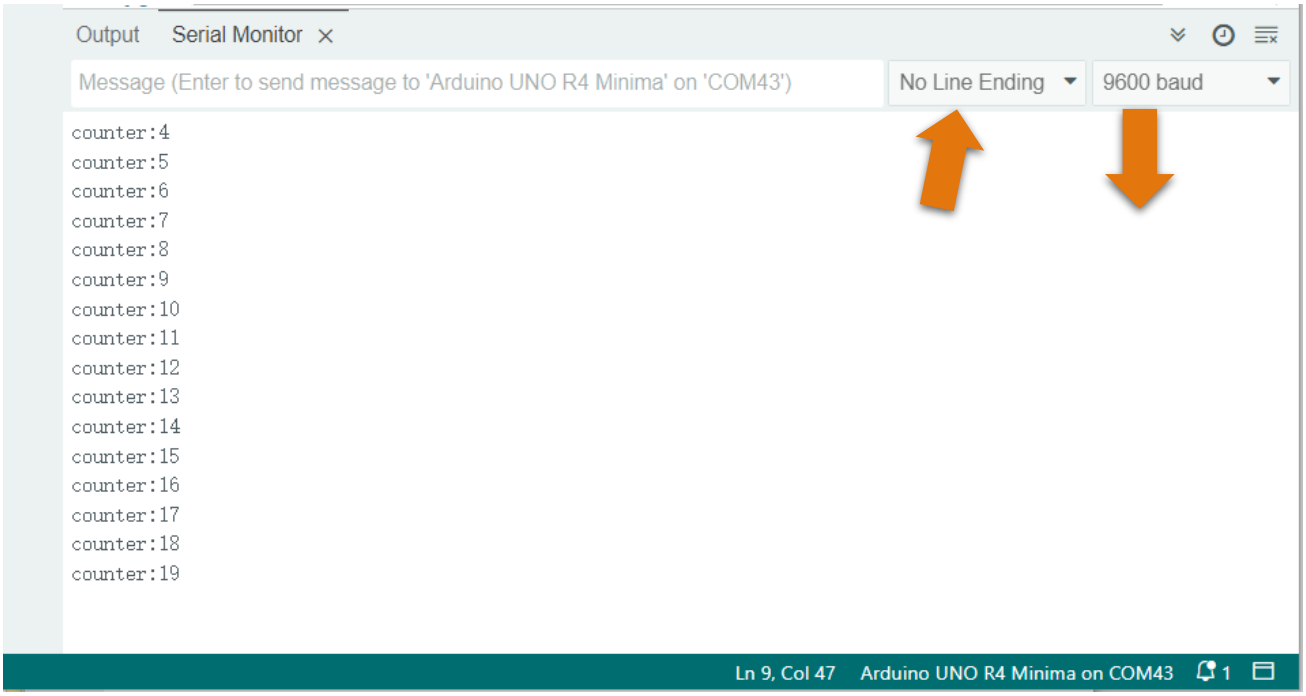
Serial.begin(speed): Initialize serial port, the parameter is the serial port baud rate;

Serial.print(val): Send string, the parameter here is what you want to send;

Serial.println(val): Send newline behind string.

Verify and upload the code, open the Serial Monitor, and then you'll see data sent from control board.

If it is not displayed correctly, check whether the configuration of the Serial Monitor in the lower right corner of the window is correct.



Project 2.2 Receive Data through Serial Port

In the previous section, we used Serial port on control board to send data to a computer, now we will use it to receive data from computer.

Component List

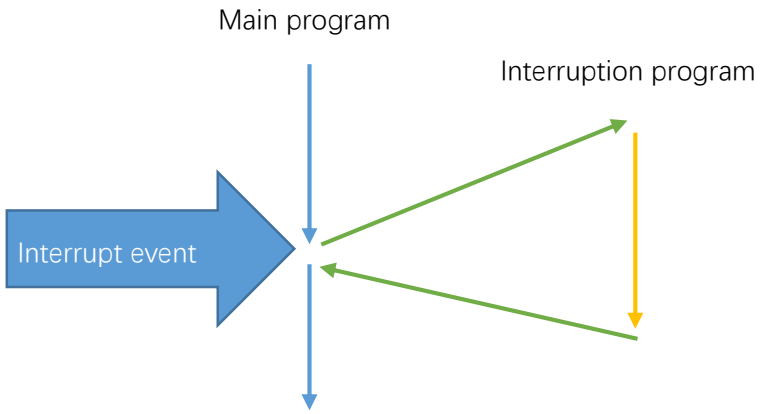
Same with the previous section.

Code Knowledge

Interrupt

An interrupt is a controller's response to an event. The event causing an interrupt is an interrupt source. We'll illustrate the interruption concept. For example, suppose you're watching TV while there is water in your kitchen heating, then you have to check whether the water is boiling or not from time to time, so you can't concentrate on watching TV. But if you have an interrupt, things will be different. Interrupt can work as a warning device for your kettle, which will beep when the water is about to boil. So before the water is boiling, you can focus on watching TV until a beep warning comes out.

Advantages of interrupt here: Processor won't need to check whether the event has happened every now and then, but when an event occurs, it informs the controller immediately. When an interrupt occurs, the processor will jump to the interrupt function to handle interrupt events, then return to where the interrupt occurs after finishing it and go on this program.



Circuit

Same with the previous section.

Sketch

Sketch 2.2.1

Now, write code to receive the characters from Serial Monitor window, and send it back.

```
1  char inChar;      // define a variable to store characters received from serial port
2
3  void setup() {
4    Serial.begin(9600);              // initialize serial port, set baud rate to 9600
5  }
6
7  void loop() {
8    if (Serial.available()) {        // judge whether data has been received
9      inChar = Serial.read();        // read one character
10     Serial.print("received:");     // print the string "received:"
11     Serial.println(inChar);        // print the received character
12   }
13 }
```

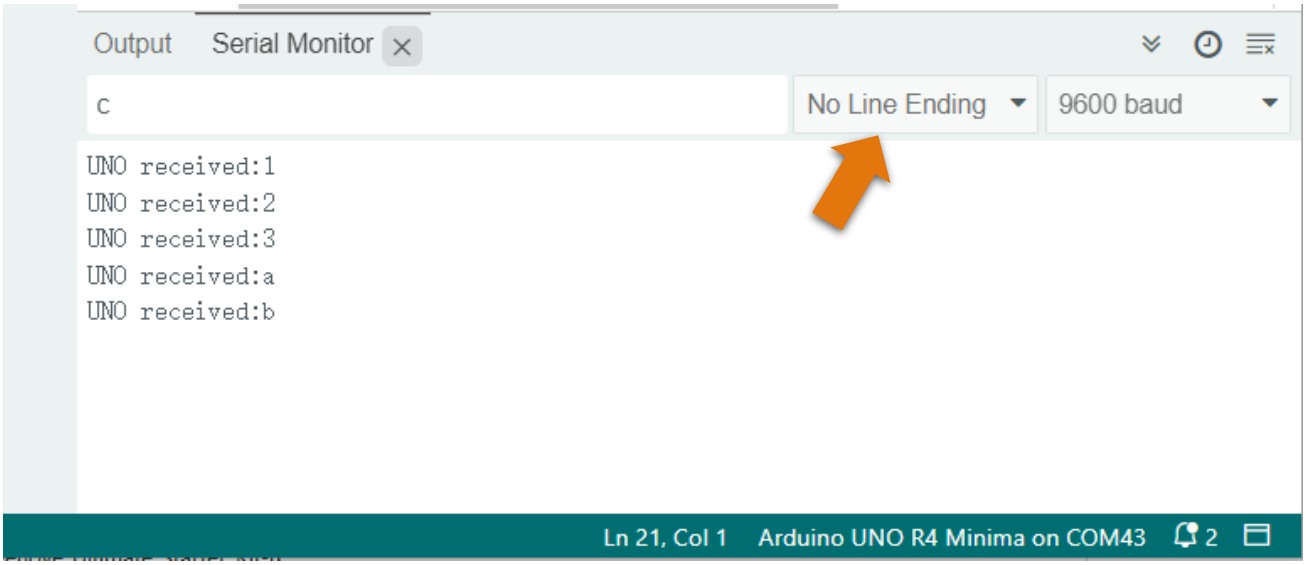
In the setup() function, we initialize the serial port. Then, the loop() function will continuously detect whether there are data to read. if so, it will read the character and send it back.

Serial Class

Serial.available(): return bytes of data that need to be read by serial port;

Serial.read(): return 1 byte of data that need to be read by serial port.

Verify and upload the code, open the Serial Monitor, write character in the sending area, click Send button, then you'll see information returned from control board.



char type

char type variable can represent a character, but it cannot store characters directly. It stores numbers to replace characters. char type occupies 1-byte store area, and use a value 0-127 to correspond to 128

characters. The corresponding relation between number and character is ruled by ASCII table. For more details of ASCII table, please refer to the appendix of this book.
Example: Define char aChar = 'a', bChar = '0', then the decimal value of aChar is 97, bChar will be 48.

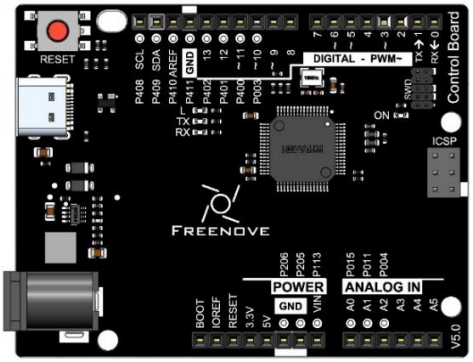
Project 2.3 Application of Serial

We will use the serial port on control board to control one LED.

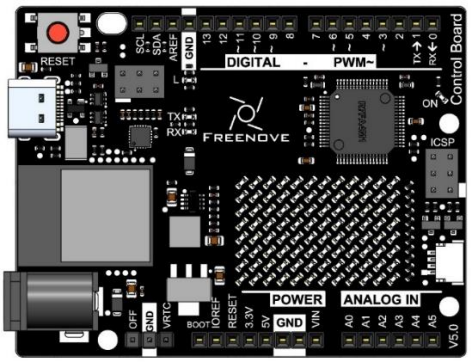
Component List

If the kit you bought does not include the following components, you can use only the control board and USB cable to finish this project.

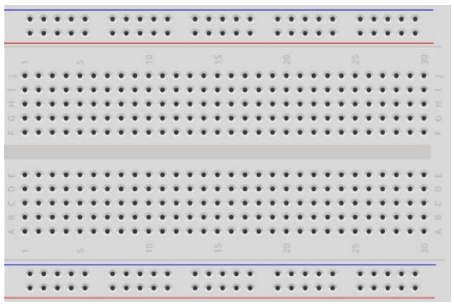
Control board x1



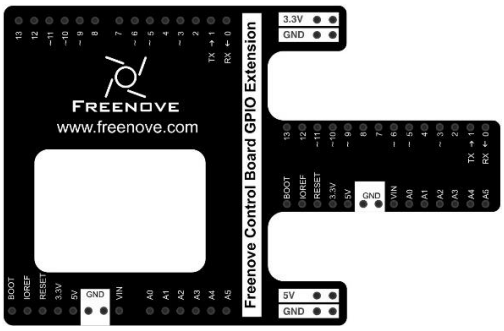
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

LED x1

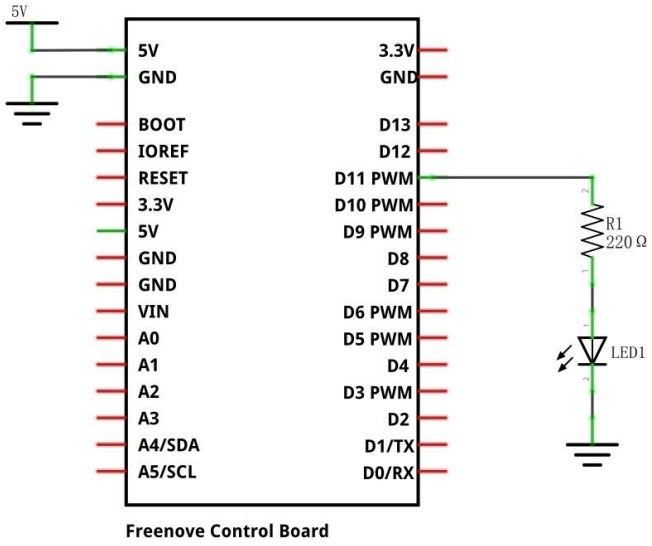
Resistor 220Ω x1

Jumper M/M x2

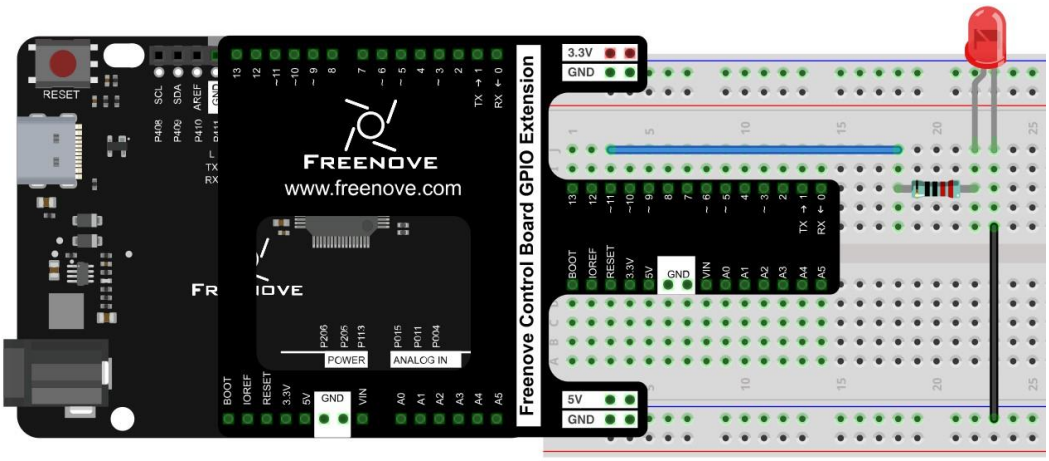
Circuit

Here we will use pin 11 of the control board to output PWM to drive 1 LED.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



If you only use the control board for this project, the LED being controlled is the onboard LED. You only need to change the LED pin to 13.

Sketch

Sketch 2.3.1

Code is basically the same with Sketch 2.2.1. But after receiving the data, control board will convert it into PWM duty cycle of output port.

```
1  int inInt;        // define a variable to store the data received from serial
2  int counter = 0;  // define a variable as the data sending to serial
3  int ledPin = 11;  // the number of the LED pin
4
```

```
5   void setup() {
6     pinMode(ledPin, OUTPUT);              // initialize the LED pin as an output
7     Serial.begin(9600);                   // initialize serial port, set baud rate to 9600
8     Serial.println("UNO is ready!");      // print the string "UNO is ready!"
9   }
10
11  void loop() {
12    if (Serial.available()) {             // judge whether the data has been received
13      inInt = Serial.parseInt();          // read an integer
14      Serial.print("UNO received:");      // print the string "UNO received:"
15      Serial.println(inInt);              // print the received character
16      // convert the received integer into PWM duty cycle of ledPin port
17      analogWrite(ledPin, constrain(inInt, 0, 255));
18    }
19  }
```

When serial receives data, it converts the data into PWM duty cycle of output port to make LED emit light with corresponding brightness.

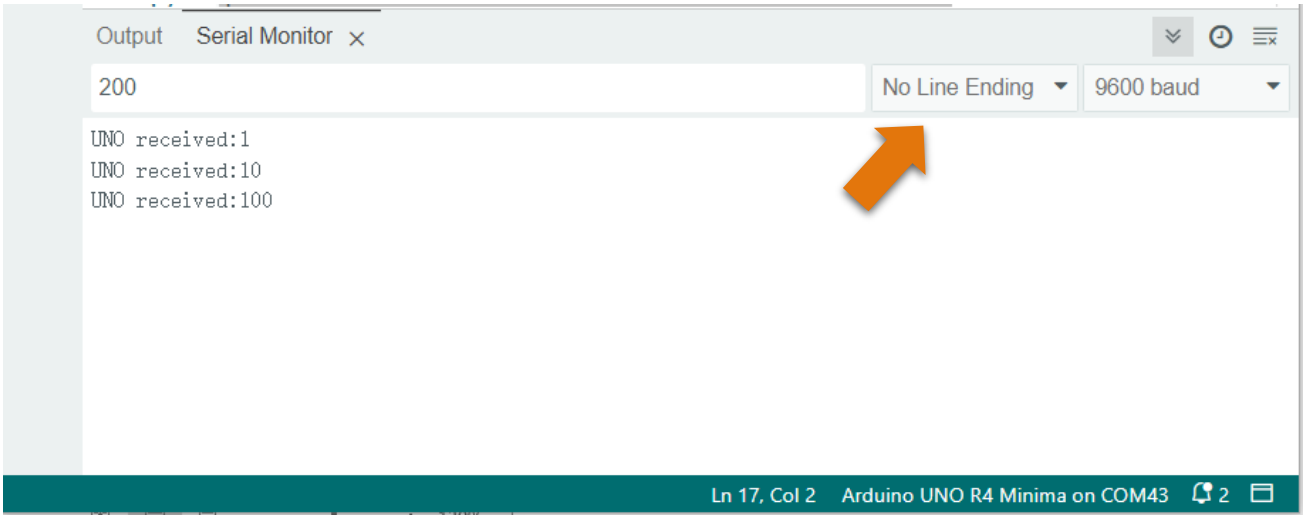
Serial Class

Serial.parseInt(): Receive an int type number as the return value.

constrain(x, a, b)

Limit x between a and b, if x<a, return a; if x>b, return b.

Verify and upload the code, open the Serial Monitor, and put a number in the range of 0-255 into the sending area and click the Send button. Then you'll see information returned from control board, meanwhile, LED can emit light with different brightness according to the number you send.



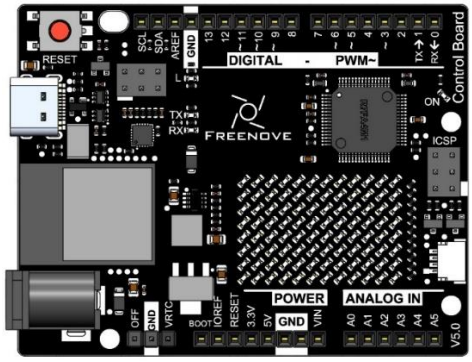
Chapter 3 Onboard LED Matrix (WiFi Board)

In this chapter, we utilize the LED matrix on the control board to display interesting patterns. The control board features a built-in 12x8 LED matrix that can be programmed to display graphics, animations, serve as an interface, and even play games.

Project 3.1 LED Matrix

In this section, we will use the LED matrix to display static graghics.

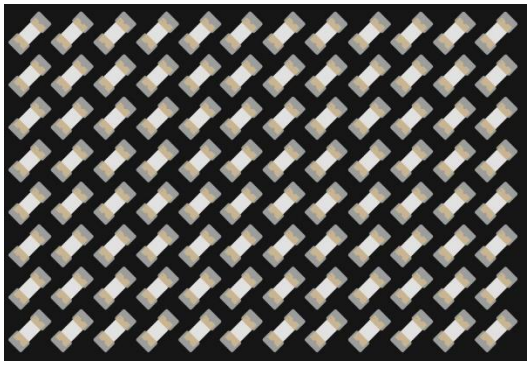
Component List

Control board x1 	USB cable x1 

Component Knowledge

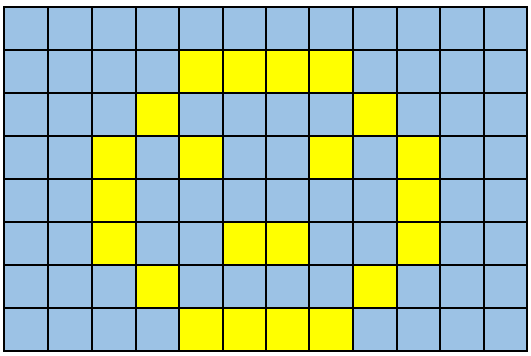
LED matrix

The LED matrix on the control board is a rectangular display module composed of a uniform grid of LEDs. Below is an 8x12 monochrome LED matrix, which includes 96 LEDs (8 rows by 12 columns).



You can call the LED matrix library to display any content you wish. With the LED matrix library, you can quickly display any graphics. For example, to display a smiling face, simply assign a value of 1 to the positions of the

LEDs that need to be lit. Conversely, assigning a value of 0 will turn off the corresponding LEDs, allowing you to set up a variety of interesting patterns.



1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	1	1	1	1	0	0	0	0
3	0	0	0	1	0	0	0	0	1	0	0	0
4	0	0	1	0	1	0	0	1	0	1	0	0
5	0	0	1	0	0	0	0	0	0	1	0	0
6	0	0	1	0	0	1	1	0	0	1	0	0
7	0	0	0	1	0	0	0	0	1	0	0	0
8	0	0	0	0	1	1	1	1	0	0	0	0

Line	Binary	Hexadecimal
1	0000 0000 0000	0x000
2	0000 1111 0000	0x0F0
3	0001 0000 1000	0x108
4	0010 0000 0100	0x204
5	0010 0000 0100	0x204
6	0010 0000 0100	0x204
7	0001 0000 1000	0x108
8	0000 1111 0000	0x0F0

To control the onboard 12x8 LED matrix, you will need a memory space of at least 96 bits in size, as shown below:

```
byte frame[8][12] = {
{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
{ 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 },
{ 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
{ 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0 },
{ 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0 },
{ 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0 },
{ 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
{ 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 }
};
```

Call the display function and you can see the LED display the above contents.

Sketch

Sketch 3.1.1

Upload the sketch to the control board and you should see the entire matrix lights up, turn off after a second, and then display the static expression resembling a smiley face.
The following is the program code:

```
#include "Arduino_LED_Matrix.h"  // Include the LED_Matrix library

ArduinoLEDMatrix matrix;  // Create an instance of the ArduinoLEDMatrix class

byte frame[8][12] = {
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 },
  { 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
  { 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0 },
  { 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0 },
  { 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0 },
  { 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
  { 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 }
};
151

const uint32_t full0n[] = {
  0xffffffff,
  0xffffffff,
  0xffffffff
};
const uint32_t full0ff[] = {
  0x00000000,
  0x00000000,
  0x00000000
};

void setup() {
  matrix.begin();  // Initialize the LED matrix
  matrix.loadFrame(full0n);
  delay(250);
  matrix.loadFrame(full0ff);
  delay(250);
}

void loop() {
```

```
36    matrix.renderBitmap(frame, 8, 12);
37    delay(250);
38  }
```

First, include the library “Arduino_LED_Matrix.h” at the beginning of the sketch, as shown below.

```
1   #include "Arduino_LED_Matrix.h"  // Include the LED_Matrix library
```

Create an object for the LED matric in the sketch.

```
3   ArduinoLEDMatrix matrix;  // Create an instance of the ArduinoLEDMatrix class
```

Activate the LED matrix by adding the line “matrix.begin();” under void setup() as shown below.

```
28    matrix.begin();  // Initialize the LED matrix
```

After the program initializes, it first lights up the entire LED matrix and then turns off the LED matrix, and subsequently continuously displays the smiling face graphic.

```
28    matrix.begin();  // Initialize the LED matrix
29    matrix.loadFrame(full0n);
30    delay(250);
31    matrix.loadFrame(full0ff);
32    delay(250);
...   ...
36    matrix.renderBitmap(frame, 8, 12);
37    delay(250);
```

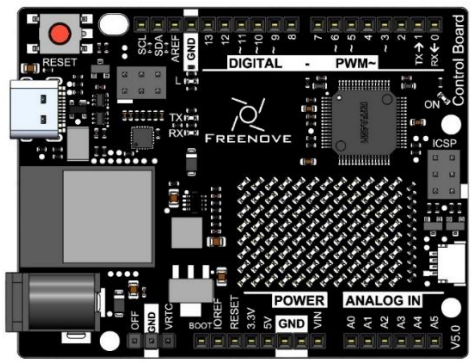
Arduino has also created an online tool called LED Matrix Editor to simplify frame creation. Each frame can be edited graphically and the duration specified. Once this process is complete, simply name the file and download it for use in your project.
To customize more graphics, you can design via this link: LED matrix editor (Arduino cc)

Project 3.2 LED Matrix

In this segment, we'll harness the onboard LED matrix to showcase dynamic visuals by scrolling the message "**Hello World!**" across the matrix.

Component List

Control board x1	USB cable x1 



Sketch

Sketch 3.2.1

When you upload the sketch to the control board, you will observe the onboard LED matrix displaying dynamic scenes. The text "**Hello World!**" will be presented in a scrolling manner.

The following is the program code:

```
#include "ArduinoGraphics.h"
#include "Arduino_LED_Matrix.h"

ArduinoLEDMatrix matrix;

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

void loop() {

  // Make it scroll!
  matrix.beginDraw();

  matrix.stroke(0xFFFFFFFF);
  matrix.textScrollSpeed(50);

  // add the text
  const char text[] = " Hello World! ";
  matrix.textFont(Font_5x7);
  matrix.beginText(0, 1, 0xFFFFFF);
  matrix.println(text);
  matrix.endText(SCROLL_LEFT);
```

```
25
26      matrix.endDraw();
27  }
```

First, include the library “Arduino_LED_Matrix.h” at the beginning of the sketch, as shown below.

```
1   #include "ArduinoGraphics.h"
2   #include "Arduino_LED_Matrix.h"
```

Create an object for the LED matric in the sketch.

```
4   ArduinoLEDMatrix matrix;  // Create an instance of the ArduinoLEDMatrix class
```

Activate the LED matrix by adding the line “matrix.begin();” under void setup() as shown below.

```
8     matrix.begin();  // Initialize the LED matrix
```

In the main function, print "**Hello World!**" on the LED matrix.

```
13    // Make it scroll!
14    matrix.beginDraw();
15
16    matrix.stroke(0xFFFFFFFF);
17    matrix.textScrollSpeed(50);
18
19    // add the text
20    const char text[] = " Hello World! ";
21    matrix.textFont(Font_5x7);
22    matrix.beginText(0, 1, 0xFFFFFF);
23    matrix.println(text);
24    matrix.endText(SCROLL_LEFT);
25
26    matrix.endDraw();
```

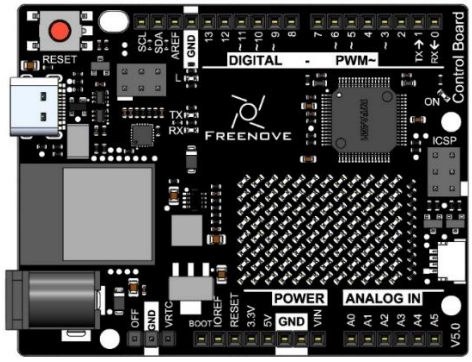
For more examples please refer to:
Using the Arduino UNO R4 WiFi LED Matrix | Arduino Documentation

Chapter 4 WiFi Working Modes

In this chapter, we'll focus on the WiFi infrastructure for control board.
Control board has 3 different WiFi operating modes: station mode, AP mode and AP+station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used.

Project 4.1 Station mode

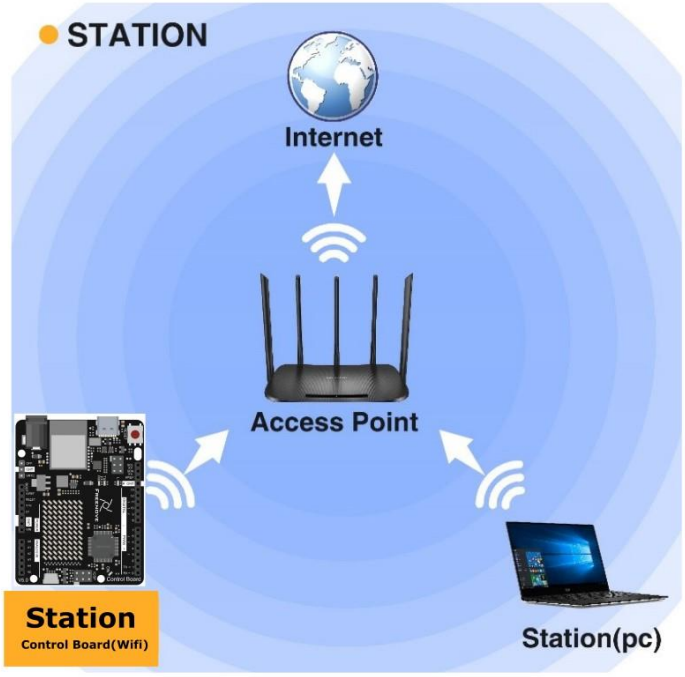
Component List

Control board x1 	USB cable x1 

Component knowledge

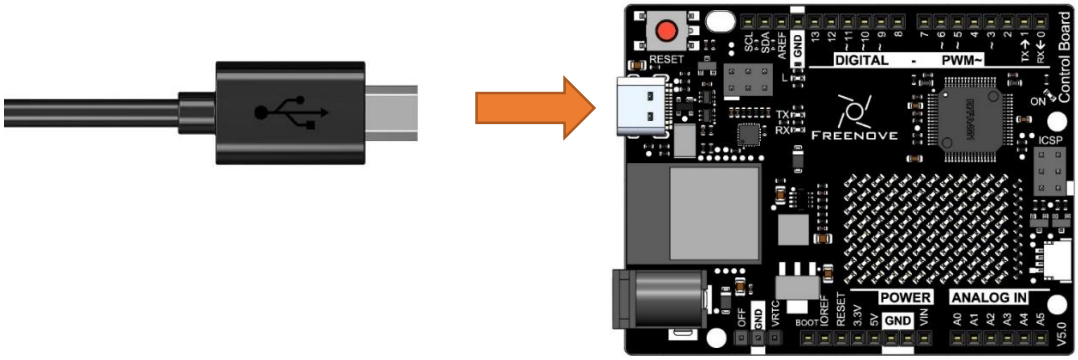
Station mode

When control board(wifi) selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if control board(wifi) wants to communicate with the PC, it needs to be connected to the router.



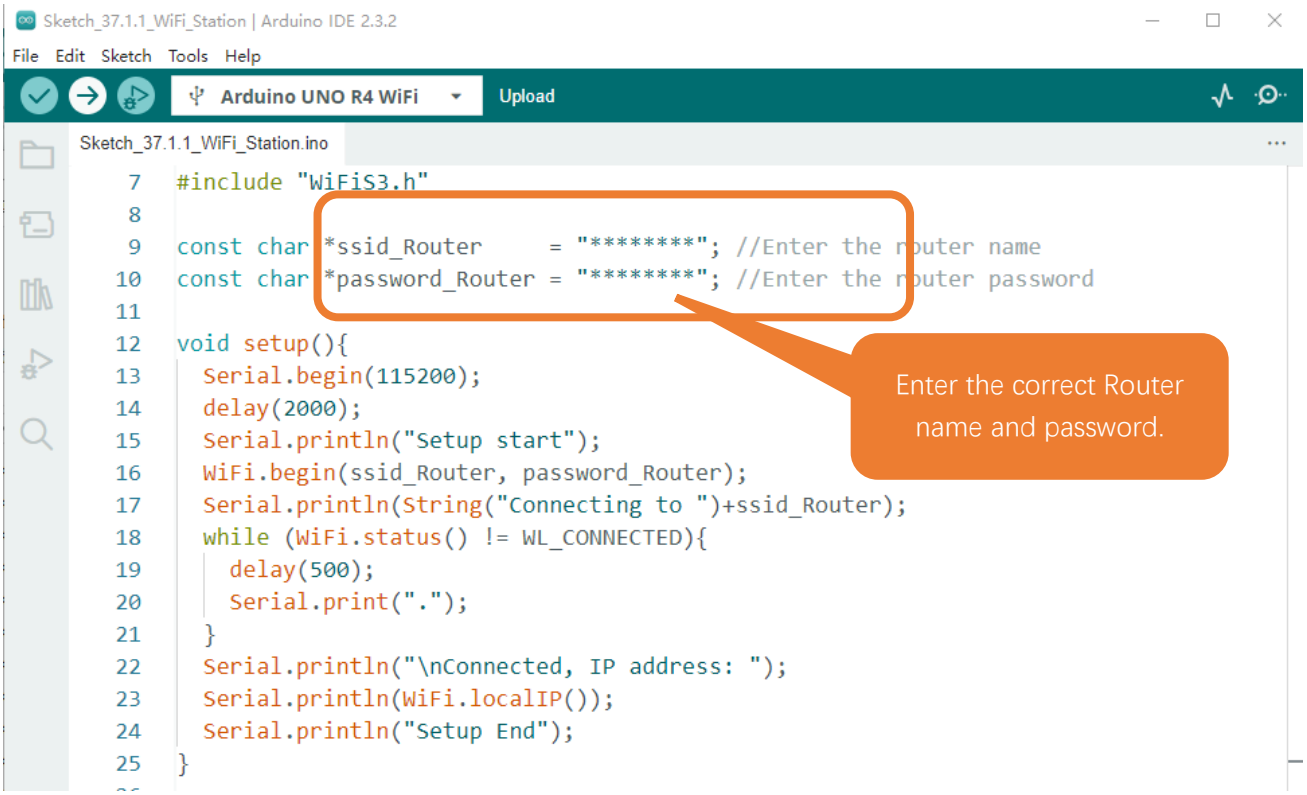
Circuit

Connect Freenove control board(wifi)to the computer using the USB cable.



Sketch

Sketch_4.1.1



Because the names and passwords of routers in various places are different, before the Sketch runs, users need to enter the correct router's name and password in the box as shown in the illustration above.

After making sure the router name and password are entered correctly, compile and upload codes to control board, open serial monitor and set baud rate to 115200. And then it will display as follows:

When control board successfully connects to "ssid_Router", serial monitor will print out the IP address assigned to control board by the router.

The following is the program code:

```
1  #include "WiFiS3.h"
```

```
2
3   const char *ssid_Router     = "********"; //Enter the router name
4   const char *password_Router = "********"; //Enter the router password
5
6   void setup(){
7     Serial.begin(115200);
8     delay(2000);
9     Serial.println("Setup start");
10    WiFi.begin(ssid_Router, password_Router);
11    Serial.println(String("Connecting to ")+ssid_Router);
12    while (WiFi.status() != WL_CONNECTED){
13      delay(500);
14      Serial.print(".");
15    }
16    Serial.println("\nConnected, IP address: ");
17    Serial.println(WiFi.localIP());
18    Serial.println("Setup End");
19  }
20
21  void loop() {
22  }
```

Include the WiFi Library header file of control board.

```
#include "WiFiS3.h"
```

Enter correct router name and password.

```
const char *ssid_Router     = "********"; //Enter the router name
const char *password_Router = "********"; //Enter the router password
```

Set control board in Station mode and connect it to your router.

```
  WiFi.begin(ssid_Router, password_Router);
```

Check whether control board has connected to router successfully every 0.5s.

```
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
```

Serial monitor prints out the IP address assigned to control board.

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

Reference

Class Station

Every time when using WiFi, you need to include header file "WiFiS3.h.".

begin(ssid, password,channel, bssid, connect): control board is used as Station to connect hotspot.

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: WiFi hotspot channel number; communicating through specified channel; optional parameter

bssid: mac address of WiFi hotspot, optional parameter

connect: blloean optional parameter, defaulting to true. If set as false, then control board won't connect

WiFi.
config(local_ip, gateway, subnet, dns1, dns2): set static local IP address.
local_ip: station fixed IP address.
subnet: subnet mask
dns1,dns2: optional parameter. define IP address of domain name server
status: obtain the connection status of WiFI
local IP(): obtian IP address in Station mode
disconnect(): disconnect wifi
setAutoConnect(boolen): set automatic connection Every time control board is power on, it will connect WiFi aitomatically.
setAutoReconnect(boolen): set automatic reconnection Every time control board disconnects WiFi, it will reconnect to WiFi automatically.

Project 4.2 AP mode

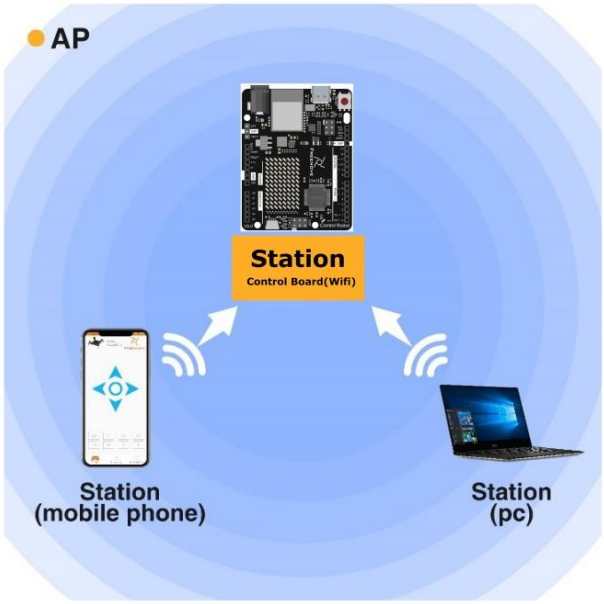
Component List & Circuit

Component List & Circuit are the same as in Section 30.1.

Component knowledge

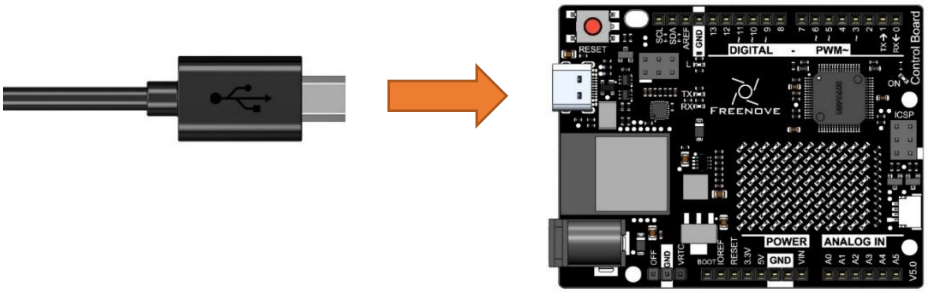
AP mode

When control board selects AP mode, it creates a hotspot network that is separate from the Internet and waits for other WiFi devices to connect. As shown in the figure below, Control board is used as a hotspot. If a mobile phone or PC wants to communicate with control board, it must be connected to the hotspot of control board. Only after a connection is established with control board can they communicate.



Circuit

Connect Freenove control board to the computer using the USB cable.

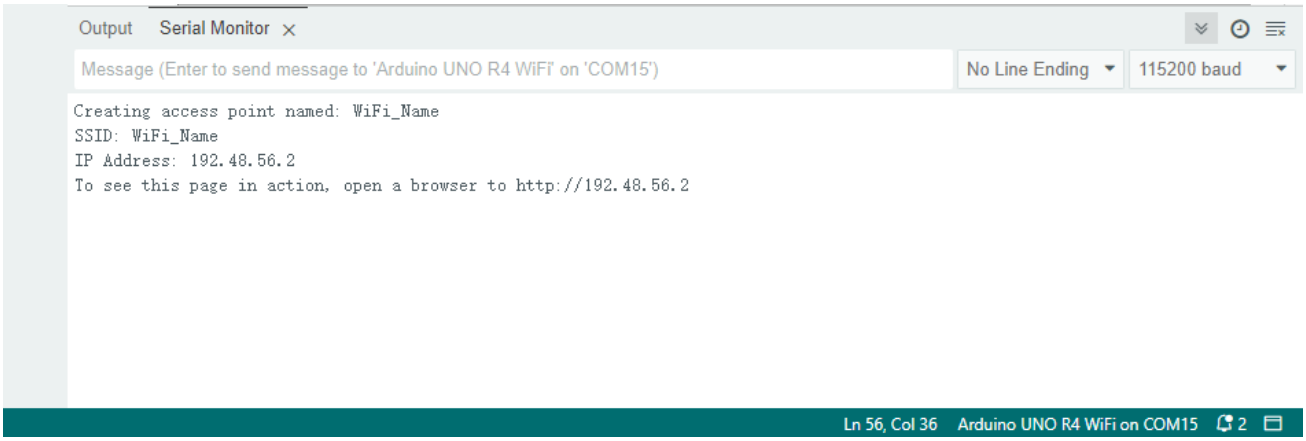


Sketch

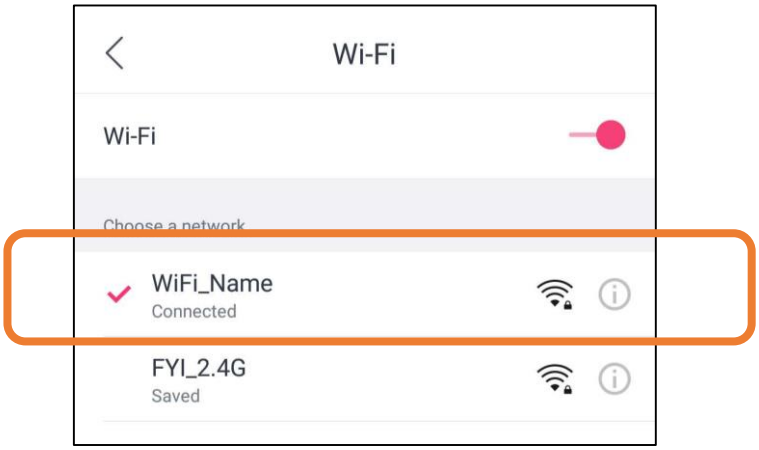
Sketch_4.2.1

Before the Sketch runs, you can make any changes to the AP name and password for control board in the box as shown in the illustration above. Of course, you can leave it alone by default.

Compile and upload codes to control board, open the serial monitor and set the baud rate to 115200. And then it will display as follows.



When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid_AP on control board, which is called "WiFi_Name" in this Sketch. You can enter the password "12345678" to connect it or change its AP name and password by modifying Sketch.



Sketch_37.2_AP_mode

The following is the program code:

```
#include "WiFiS3.h"

///////please enter your sensitive data in the Secret
tab/arduino_secrets.h
char ssid[] = "WiFi_Name";        // your network SSID (name)
char pass[] = "12345678";        // your network password (use for WPA, or
use as key for WEP)

int status = WL_IDLE_STATUS;
void setup() {
  //Initialize serial and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }
  // check for the WiFi module:
  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("Communication with WiFi module failed!");
    // don't continue
    while (true);
  }

  String fv = WiFi.firmwareVersion();
  if (fv < WIFI_FIRMWARE_LATEST_VERSION) {
    Serial.println("Please upgrade the firmware");
  }
  // by default the local IP address will be 192.168.4.1
  // you can override it with the following:
  WiFi.config(IPAddress(192,48,56,2));
  // print the network name (SSID);
  Serial.print("Creating access point named: ");
  Serial.println(ssid);

  // Create open network. Change this line if you want to create an WEP
network:
  status = WiFi.beginAP(ssid, pass);
  if (status != WL_AP_LISTENING) {
    Serial.println("Creating access point failed");
    // don't continue
    while (true);
  }

```

```
43    // wait 5 seconds for connection:
44    delay(5000);
45    // you're connected now, so print out the status
46    printWiFiStatus();
47  }
48
49  void loop() {
50  }
51
52  void printWiFiStatus() {
53    // print the SSID of the network you're attached to:
54    Serial.print("SSID: ");
55    Serial.println(WiFi.SSID());
56
57    // print your WiFi shield's IP address:
58    IPAddress ip = WiFi.localIP();
59    Serial.print("IP Address: ");
60    Serial.println(ip);
61
62    // print where to go in a browser:
63    Serial.print("To see this page in action, open a browser to http://");
64    Serial.println(ip);
65
66  }
```

Include WiFi Library header file of control board.

```
1   #include "WiFiS3.h"
```

Enter correct AP name and password.

```
5   char ssid[] = "WiFi_Name";          // your network SSID (name)
6   char pass[] = "12345678";          // your network password (use for WPA, or
    use as key for WEP)
```

Check whether the AP is turned on successfully. If yes, print out IP and MAC address of AP established by control board. If no, print out the failure prompt.

```
36    status = WiFi.beginAP(ssid, pass);
37    if (status != WL_AP_LISTENING) {
38      Serial.println("Creating access point failed");
39      // don't continue
40      while (true);
41    }
…   …
43    // wait 5 seconds for connection:
44    delay(5000);
45    // you're connected now, so print out the status
46    printWiFiStatus();
```

```
47
...  ...
52   void printWiFiStatus() {
53     // print the SSID of the network you're attached to:
54     Serial.print("SSID: ");
55     Serial.println(WiFi.SSID());
56
57     // print your WiFi shield's IP address:
58     IPAddress ip = WiFi.localIP();
59     Serial.print("IP Address: ");
60     Serial.println(ip);
61
62     // print where to go in a browser:
63     Serial.print("To see this page in action, open a browser to http://");
64     Serial.println(ip);
65
66   }
```

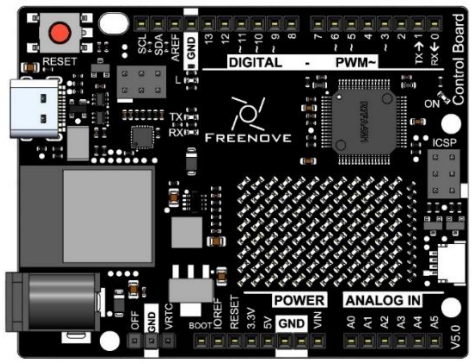
Chapter 5 TCP/IP

In this chapter, we will introduce how control board(wifi) implements network communications based on TCP/IP protocol. There are two roles in TCP/IP communication, namely Server and Client, which will be implemented respectively with two projects in this chapter.

Project 5.1 As Client

In this section, control board(wifi) is used as Client to connect Server on the same LAN and communicate with it.

Component List

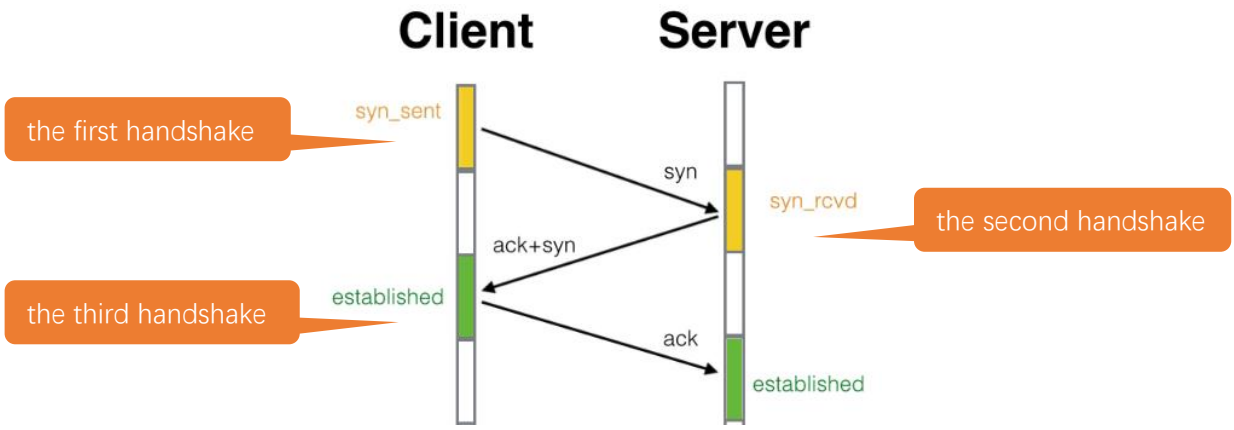
Control board x1 	USB cable x1 

Component knowledge

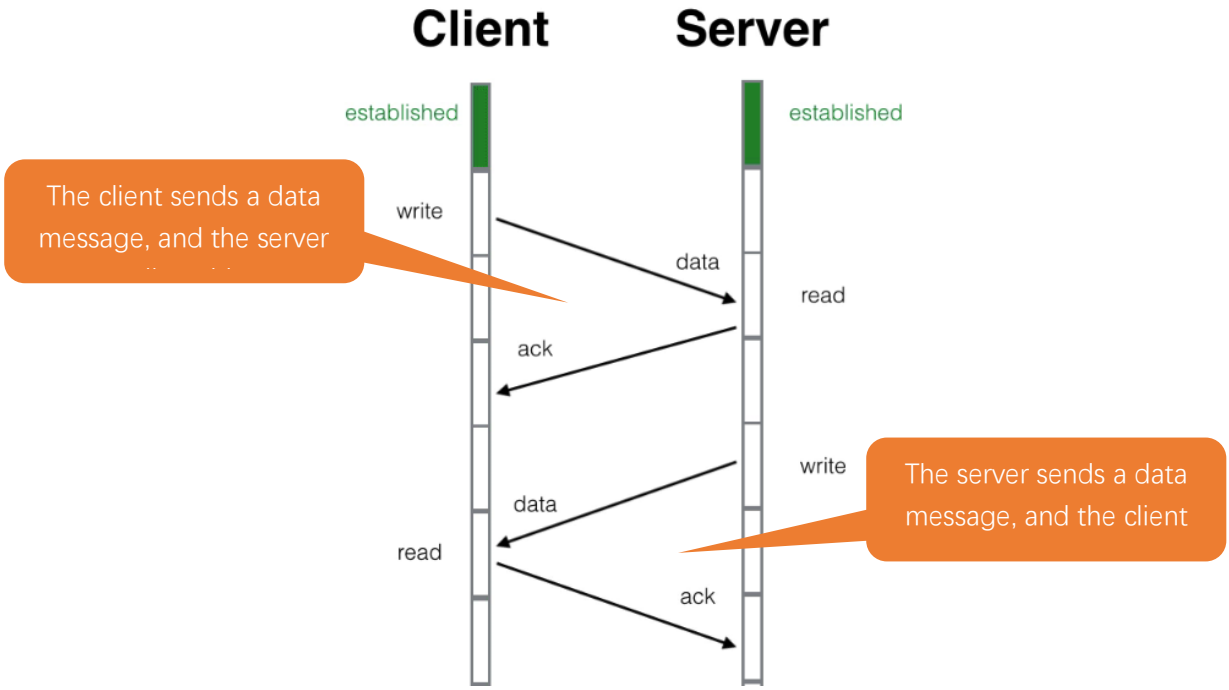
TCP connection

Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request to the server, and each time such a request is proposed, a "three-times handshake" is required.

Three-times handshake: In the TCP protocol, during the preparation phase of sending data, the client and the server interact three times to ensure the reliability of the connection, which is called "three-times handshake".
The first handshake, the client sends a connection request to the server and waits for the server to confirm.
The second handshake, the server sends a response back to the client informing that it has received the connection request.
The third handshake, the client sends a confirmation message to the server again to confirm the connection.



TCP is a connection-oriented, low-level transmission control protocol. After TCP establishes a connection, the client and server can send and receive messages to each other, and the connection will always exist as long as the client or server does not initiate disconnection. Each time one party sends a message, the other party will reply with an ack signal.



Install Processing

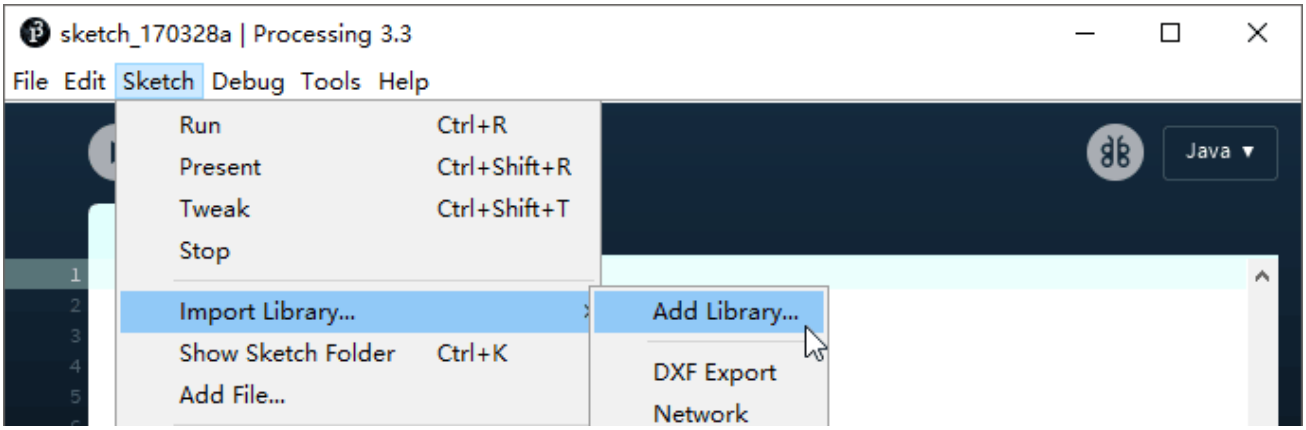
In this tutorial, we use Processing to build a simple TCP/IP communication platform.

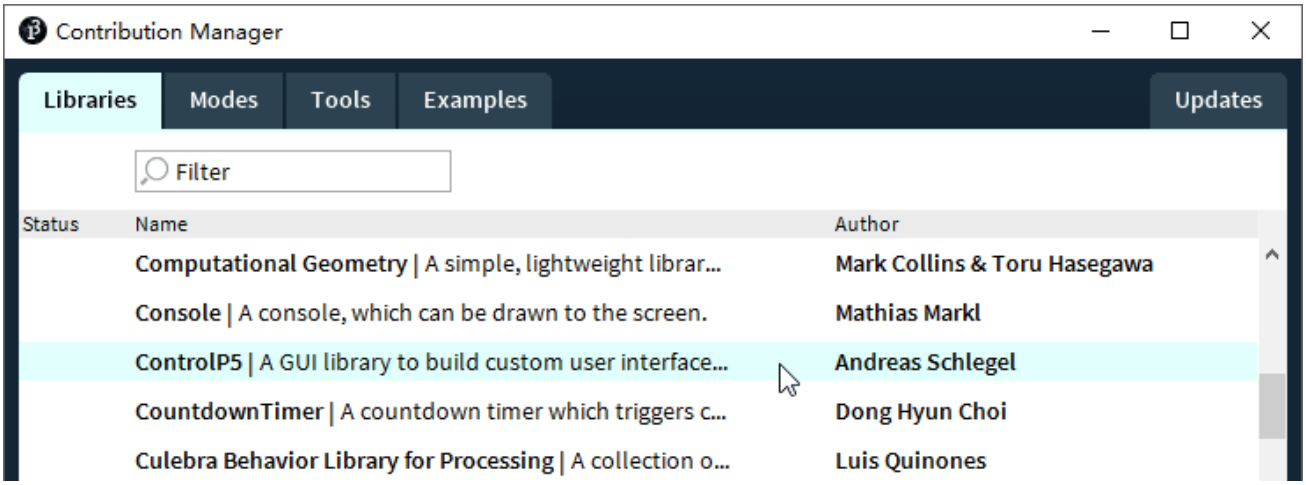
If you've not installed Processing, you can download it by clicking https://processing.org/download/. You can choose an appropriate version to download according to your PC system.

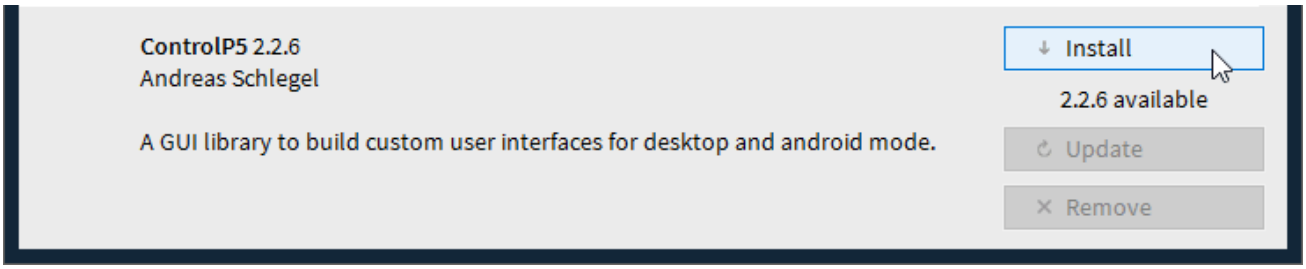
Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.

Use Server mode for communication

Install ControlP5.





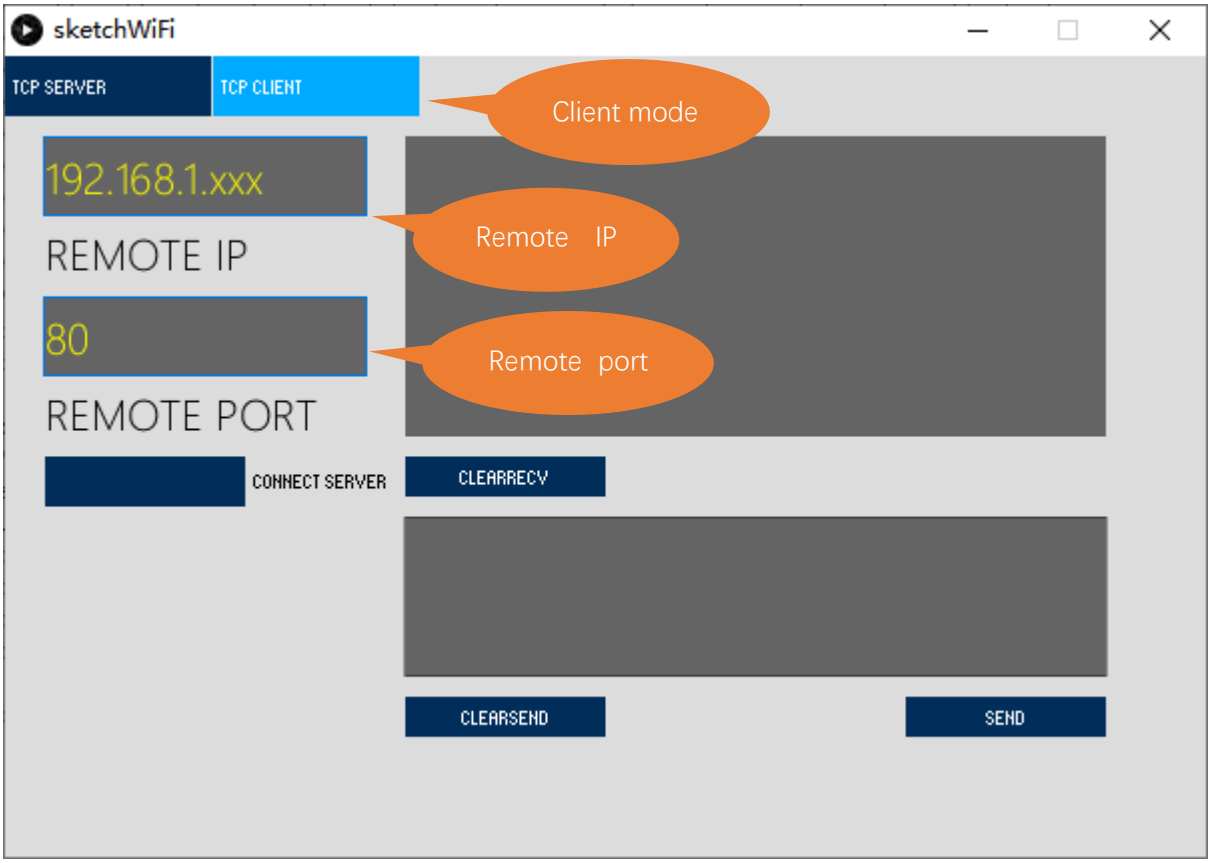


Open the "**Sketches\Sketches\Sketch_37.1_WiFiClient\sketchWiFi\sketchWiFi.pde**", and click "Run".

The new pop-up interface is as follows. If control board(wifi) is used as client, select TCP SERVER mode for sketchWiFi.

When sketchWiFi selects TCP SERVER mode, control board(wifi) Sketch needs to be changed according to sketchWiFi's displaying of LOCAL IP or LOCAL PORT.

If ESP32 serves as server, select TCP CLIENT mode for sketchWiFi.



When sketchWiFi selects TCP CLIENT mode, the LOCAL IP and LOCAL PORT of sketchWiFi need to be changed according to the IP address and port number printed by the serial monitor.

Mode selection: select **Server mode/Client mode**.

IP address: In server mode, this option does not need to be filled in, and the computer will automatically obtain the IP address.

In client mode, fill in the remote IP address to be connected.

Port number: In server mode, fill in a port number for client devices to make an access connection.

In client mode, fill in port number given by the Server devices to make an access connection.

Start button: In server mode, push the button, then the computer will serve as server and open a port number for client to make access connection. During this period, the computer will keep monitoring.

In client mode, before pushing the button, please make sure the server is on, remote IP address and remote port number is correct; push the button, and the computer will make access connection to the remote port number of the remote IP as a client.

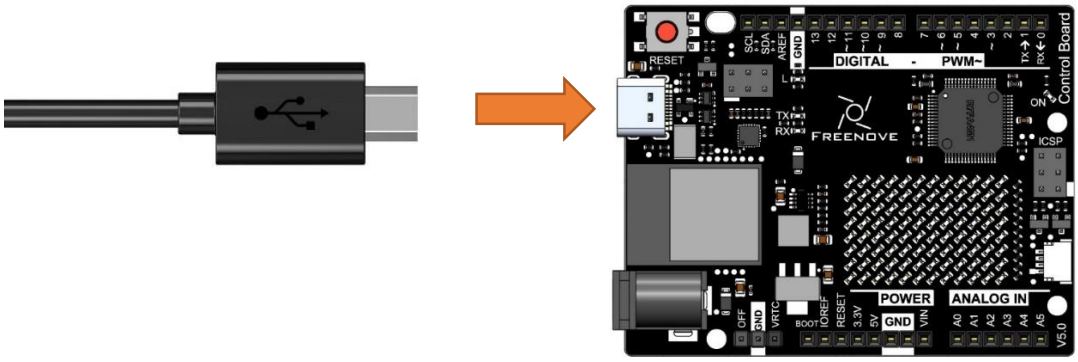
clear receive: clear out the content in the receiving text box

clear send: clear out the content in the sending text box

Sending button: push the sending button, the computer will send the content in the text box to others.

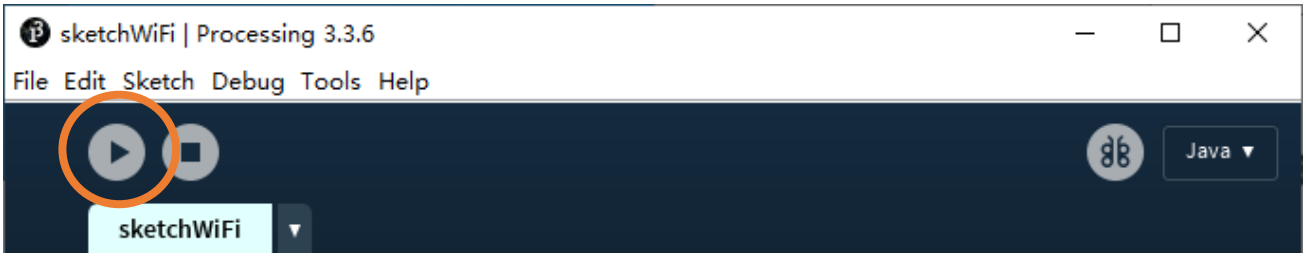
Circuit

Connect Freenove control board to the computer using USB cable.



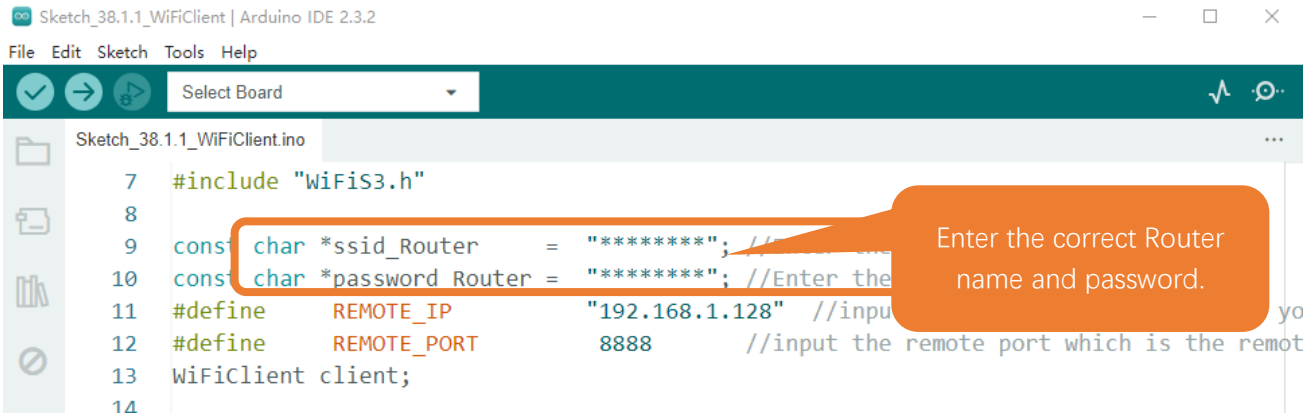
Sketch

Before running the Sketch, please open “sketchWiFi.pde.” first, and click “Run”.



The newly pop up window will use the computer’s IP address by default and open a data monitor port.

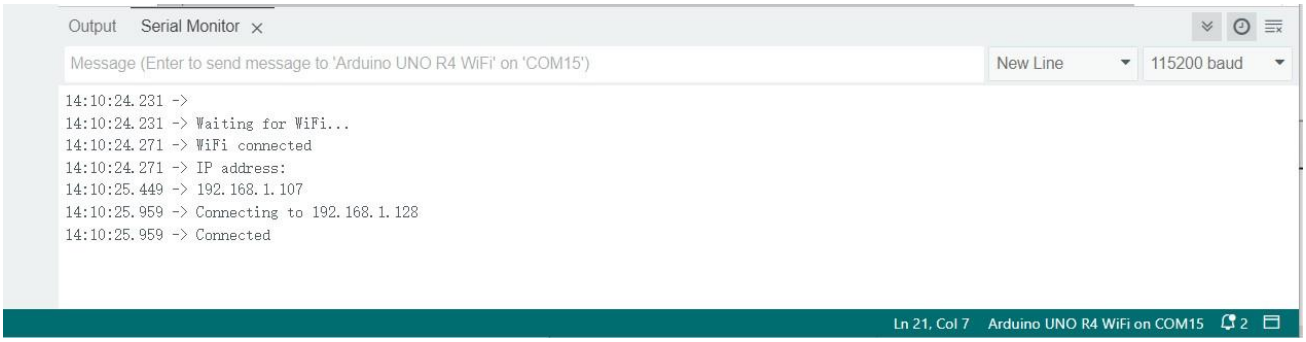
Next, open WiFiClient.ino. Before running it, please change the following information based on "LOCAL IP" and "LOCAL PORT" in the figure above.



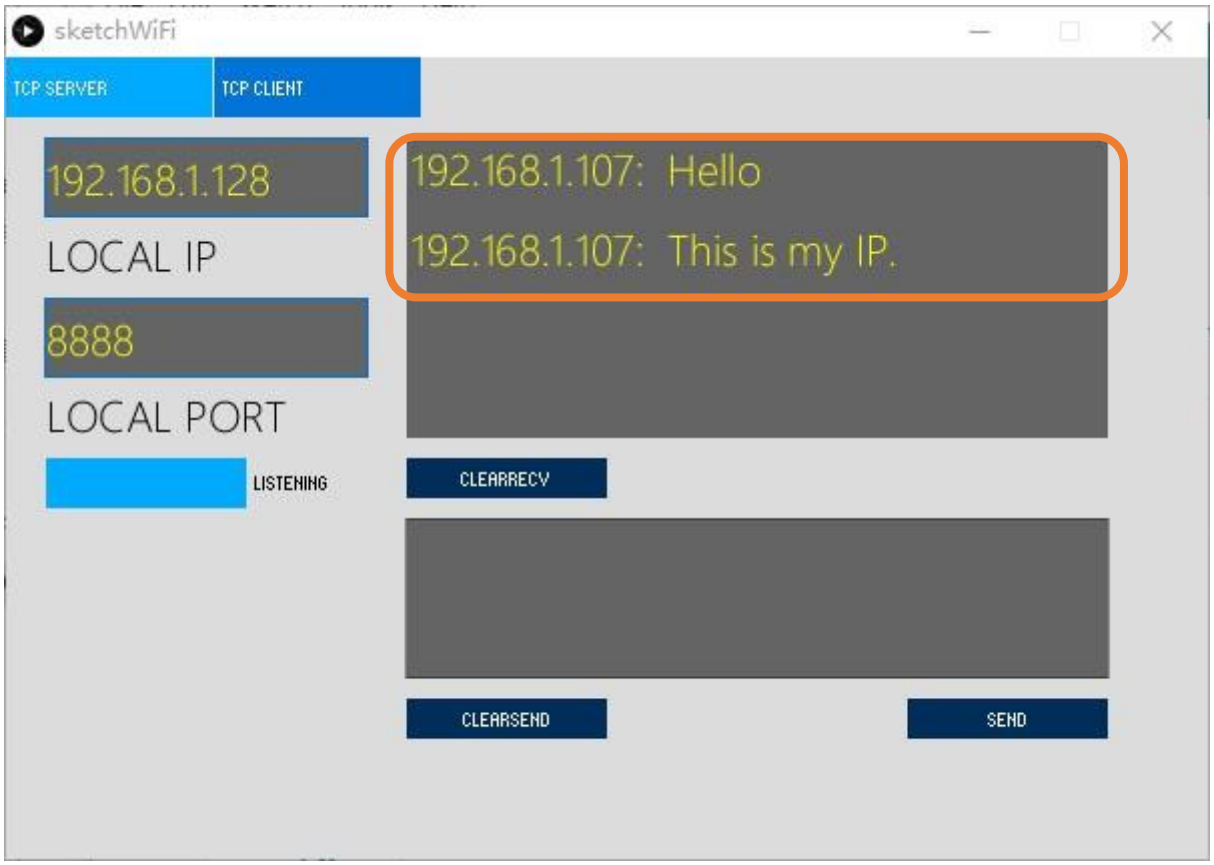
REMOTE_IP needs to be filled in according to the interface of sketchWiFi.pde. Taking this tutorial as an example, its REMOTE_IP is “192.168.1.128”. Generally, by default, the ports do not need to change its value.

Click LISTENING, turn on TCP SERVER's data listening function and wait for control board(wifi) to connect.

Compile and upload code to control board(wifi) , open the serial monitor and set the baud rate to 115200. control board(wifi) connects router, obtains IP address and sends access request to server IP address on the same LAN till the connection is successful. When connect successfully, control board(wifi) can send messages to server.


Output
Serial Monitor
Message (Enter to send message to 'Arduino UNO R4 WiFi' on 'COM15')
New Line
115200 baud
14:10:24.231 ->
14:10:24.231 -> Waiting for WiFi...
14:10:24.271 -> WiFi connected
14:10:24.271 -> IP address:
14:10:25.449 -> 192.168.1.107
14:10:25.959 -> Connecting to 192.168.1.128
14:10:25.959 -> Connected
Ln 21, Col 7
Arduino UNO R4 WiFi on COM15
2

Control board(wifi) connects with TCP SERVER, and TCP SERVER receives messages from control board(wifi), as shown in the figure below.



Sketch_38.1_As_Client

The following is the program code:

```
#include "WiFiS3.h"

const char *ssid_Router     =  "********"; //Enter the router name
const char *password_Router =  "********"; //Enter the router password
#define     REMOTE_IP       "********"  //input the remote server which is you want to connect
#define     REMOTE_PORT      8888       //input the remote port which is the remote provide
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);

  WiFi.begin(ssid_Router, password_Router);
  Serial.print("\nWaiting for WiFi... ");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  delay(500);

  Serial.print("Connecting to ");
  Serial.println(REMOTE_IP);

  while (!client.connect(REMOTE_IP, REMOTE_PORT)) {
    Serial.println("Connection failed.");
    Serial.println("Waiting a moment before retrying...");
  }
  Serial.println("Connected");
  client.print("Hello\n");
  client.print("This is my IP.\n");
}

void loop() {
  if (client.available() > 0) {
    delay(20);
    //read back one line from the server
    String line = client.readString();
    Serial.println(REMOTE_IP + String(":") + line);
```

```
  }
  if (Serial.available() > 0) {
    delay(20);
    String line = Serial.readString();
    client.print(line);
  }
  if (client.connected () == 0) {
    client.stop();
    WiFi.disconnect();
  }
}
```

Add WiFi function header file.

```
#include "WiFiS3.h"
```

Enter the actual router name, password, remote server IP address, and port number.

```
const char *ssid_Router     =  "********"; //Enter the router name
const char *password_Router =  "********"; //Enter the router password
#define     REMOTE_IP       "********"  //input the remote server which is you want to connect
#define     REMOTE_PORT      8888       //input the remote port which is the remote provide
```

Apply for the method class of WiFiClient.

```
WiFiClient client;
```

Connect specified WiFi until it is successful. If the name and password of WiFi are correct but it still fails to connect, please push the reset key.

```
  WiFi.begin(ssid_Router, password_Router);
  Serial.print("\nWaiting for WiFi... ");
  while (WiFi.status() ! = WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
```

Send connection request to remote server until connect successfully. When connect successfully, print out the connecting prompt on the serial monitor and send messages to remote server.

```
  while (!client.connect(REMOTE_IP, REMOTE_PORT)) {//Connect to Server
    Serial.println("Connection failed.");
    Serial.println("Waiting a moment before retrying...");
  }
  Serial.println("Connected");
  client.print("Hello\n");
```

When control board(wifi) receive messages from servers, it will print them out via serial port; Users can also send messages to servers from serial port.

```
if (client.available() > 0) {
    delay(20);
    //read back one line from the server
    String line = client.readString();
    Serial.println(REMOTE_IP + String(":") + line);
  }
```

```
43    if (Serial.available() > 0) {
44      delay(20);
45      String line = Serial.readString();
46      client.print(line);
47    }
```

If the server is disconnected, turn off WiFi of control board.

```
48    if (client.connected () == 0) {
49      client.stop();
50      WiFi.disconnect();
51    }
```

Reference

Class Client

Every time when using Client, you need to include header file "WiFiS3.h."

connect(ip, port, timeout)/connect(*host, port, timeout): establish a TCP connection.

ip, *host: ip address of target server

port: port number of target server

timeout: connection timeout

connected(): judge whether client is connecting. If return value is 1, then connect successfully; If return value is 0, then fail to connect.

stop(): stop tcp connection

print(): send data to server connecting to client

available(): return to the number of bytes readable in receive buffer, if no, return to 0 or -1.

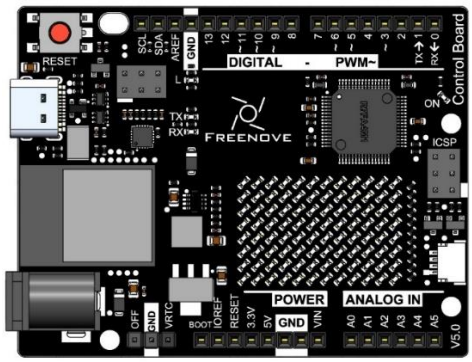
read(): read one byte of data in receive buffer

readString(): read string in receive buffer

Project 5.2 As Server

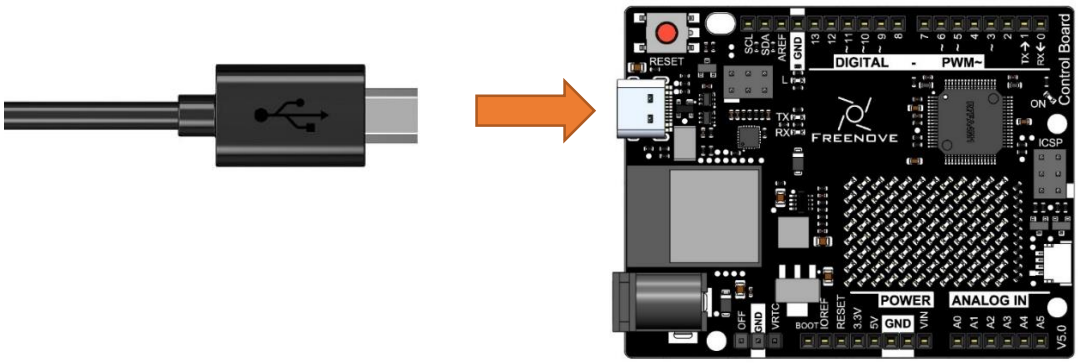
In this section, control board(wifi) is used as a server to wait for the connection and communication of client on the same LAN.

Component List

Control board x1 	USB cable x1 

Circuit

Connect Freenove control board(wifi) to the computer using a USB cable.

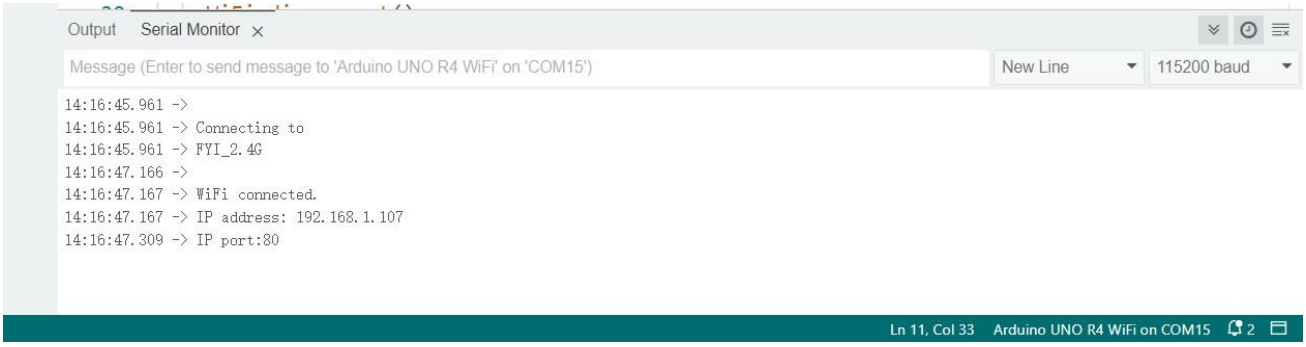


Sketch

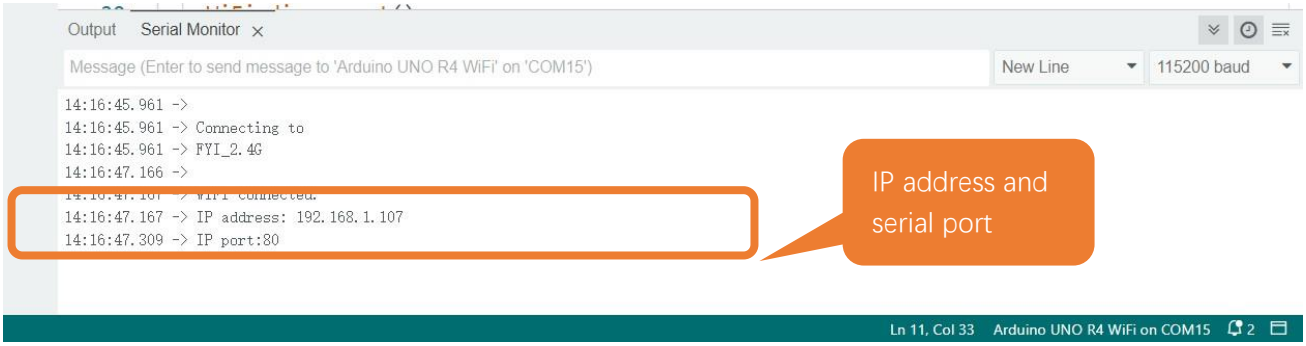
Before running Sketch, please modify the contents of the box below first.

Sketch_5.2.1

Compile and upload code to control board(wifi) , open the serial monitor and set the baud rate to 115200.
Turn on server mode for control board(wifi), waiting for the connection of other devices on the same LAN.
Once a device connects to server successfully, they can send messages to each other.
If the control board(wifi) fails to connect to router, press the reset button as shown below and wait for control board(wifi) to run again.



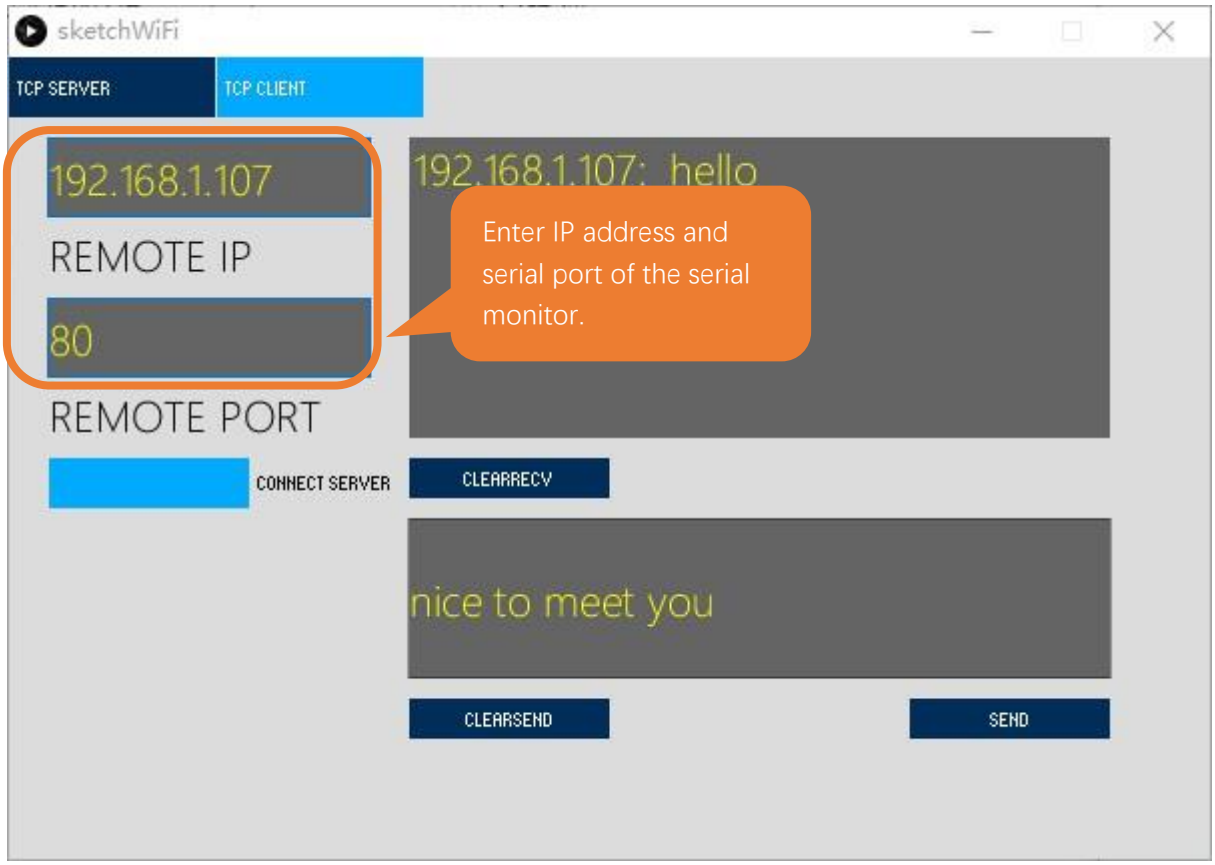
Serial Monitor



Processing:

Open the "**Sketches\Sketch_5.2.1_WiFiServer\sketchWiFi\sketchWiFi.pde**".

Based on the messages printed by the serial monitor, enter correct IP address and serial port in Processing to establish connection and make communication.



The following is the program code:

```
#include "WiFiS3.h"

#define port 80
const char *ssid_Router      = "********";  //input your wifi name
const char *password_Router  = "********";  //input your wifi passwords
WiFiServer  server(port);

void setup()
{
    Serial.begin(115200);
    Serial.printf("\nConnecting to ");
    Serial.println(ssid_Router);
    WiFi.disconnect();
    WiFi.begin(ssid_Router, password_Router);
    delay(1000);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");
```

```
22      Serial.print("IP address: ");
23      Serial.println(WiFi.localIP());
24      Serial.printf("IP port: %d\n",port);
25      server.begin(port);
26      WiFi.setAutoConnect(true);
27      WiFi.setAutoReconnect(true);
28  }
29
30  void loop(){
31   WiFiClient client = server.available();            // listen for incoming clients
32    if (client) {                                    // if you get a client
33      Serial.println("Client connected.");
34      while (client.connected()) {                   // loop while the client's connected
35        if (client.available()) {                    // if there's bytes to read from the
    client
36          Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
37          while(client.read()>0);                    // clear the wifi receive area cache
38        }
39        if(Serial.available()){                      // if there's bytes to read from the
    serial monitor
40          client.print(Serial.readStringUntil('\n')); // print it out the client.
41          while(Serial.read()>0);                    // clear the wifi receive area cache
42        }
43      }
44      client.stop();                                 // stop the client connecting.
45      Serial.println("Client Disconnected.");
46    }
47  }
```

Apply for method class of WiFiServer.

```
6   WiFiServer server(port);          //Apply for a Server object whose port number is 80
```

Connect specified WiFi until it is successful. If the name and password of WiFi are correct but it still fails to connect, please push the reset key.

```
13      WiFi.disconnect();
14      WiFi.begin(ssid_Router, password_Router);
15      delay(1000);
16      while (WiFi.status() != WL_CONNECTED) {
17          delay(500);
18          Serial.print(".");
19      }
20      Serial.println("");
21      Serial.println("WiFi connected.");
```

Print out the IP address and port number of control board(wifi).

```
22      Serial.print("IP address: ");
23      Serial.println(WiFi.localIP());                  //print out IP address of ESP32
```

```
24      Serial.printf("IP port: %d\n",port);              //Print out ESP32's port number
```

Turn on server mode of control board(wifi), start automatic connection and turn on automatic reconnection.

```
25      server.begin();                                  //Turn ON ESP32 as Server mode
26      WiFi.setAutoConnect(true);
27      WiFi.setAutoReconnect(true);
```

When control board(wifi) receive messages from servers, it will print them out via serial port; Users can also send messages to servers from serial port.

```
35      if (client.available()) {                        // if there's bytes to read from the
        client
36        Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
37        while(client.read()>0);                        // clear the wifi receive area cache
38      }
39      if(Serial.available()){                          // if there's bytes to read from the
        serial monitor
40        client.print(Serial.readStringUntil('\n')); // print it out the client.
41        while(Serial.read()>0);                        // clear the wifi receive area cache
42      }
```

Reference

Class Server

Every time use Server functionality, we need to include header file"WiFiS3.h".

WiFiServer(uint16_t port=80, uint8_t max_clients=4): create a TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 80.

max_clients: maximum number of clients with default number as 4.

begin(port): start the TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 0.

setNoDelay(bool nodelay): whether to turn off the delay sending functionality.

nodelay: true stands for forbidden Nagle algorithm.

close(): close tcp connection.

stop(): stop tcp connection.

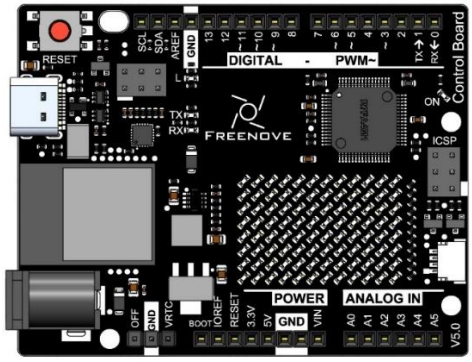
Chapter 6 Control LED with Web

In this chapter, we will use control board to make a simple smart home. We will learn how to control LED lights through web pages.

Project 6.1 Control the LED with Web

In this project, we need to build a Web Service and then use the control board to control the LED through the Web browser of the phone or PC. Through this example, you can remotely control the appliances in your home to achieve smart home.

Component List

Control board x1 	USB cable x1 

Component knowledge

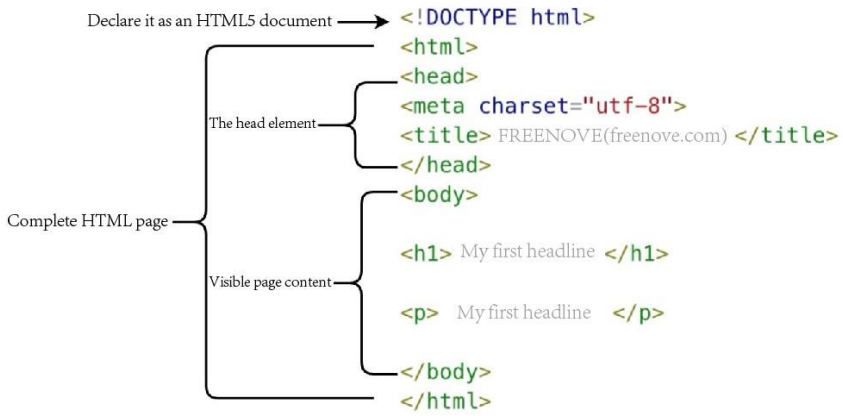
HTML

HyperText Markup Language (HTML) is a standard Markup Language for creating web pages.It includes a set of tags that unify documents on the network and connect disparate Internet resources into a logical whole.HTML text is descriptive text composed of HTML commands that describe text, graphics, animations, sounds, tables, links, etc.The extension of the HTML file is HTM or HTML.Hyper Text is a way to organize information.It uses hyperlinks to associate words and charts in Text with other information media.These related information media may be in the same Text, other files, or files located on a remote computer.This way of organizing information connects the information resources distributed in different places, which is convenient for people to search and retrieve information.

The nature of the Web is hypertext Markup Language (HTML), which can be combined with other Web technologies (e.g., scripting languages, common gateway interfaces, components, etc.) to create powerful Web pages. Thus, HYPERtext Markup Language (HTML) is the foundation of World Wide Web (Web) programming, that is, the World Wide Web is based on hypertext. Hypertext Markup Language is called hypertext Markup language because the text contains so-called "hyperlink" points.

You can build your own WEB site using HTML, which runs on the browser and is parsed by the browser.

Example analysis is shown in the figure below:



<!DOCTYPE html>: Declare it as an HTML5 document

<html>: Is the root element of an HTML page

<head>: Contains meta data for the document, such as < meta charset="utf-8"> Define the web page encoding format to UTF-8.

<title>: Notesthe title of the document

<body>: Contains visible page content

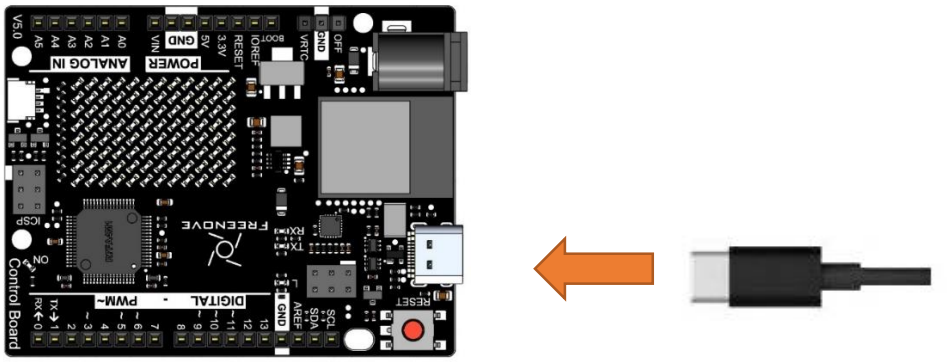
<h1>: Define a big heading

<p>: Define a paragraph

For more information, please visit: https://developer.mozilla.org/en-US/docs/Web/HTML

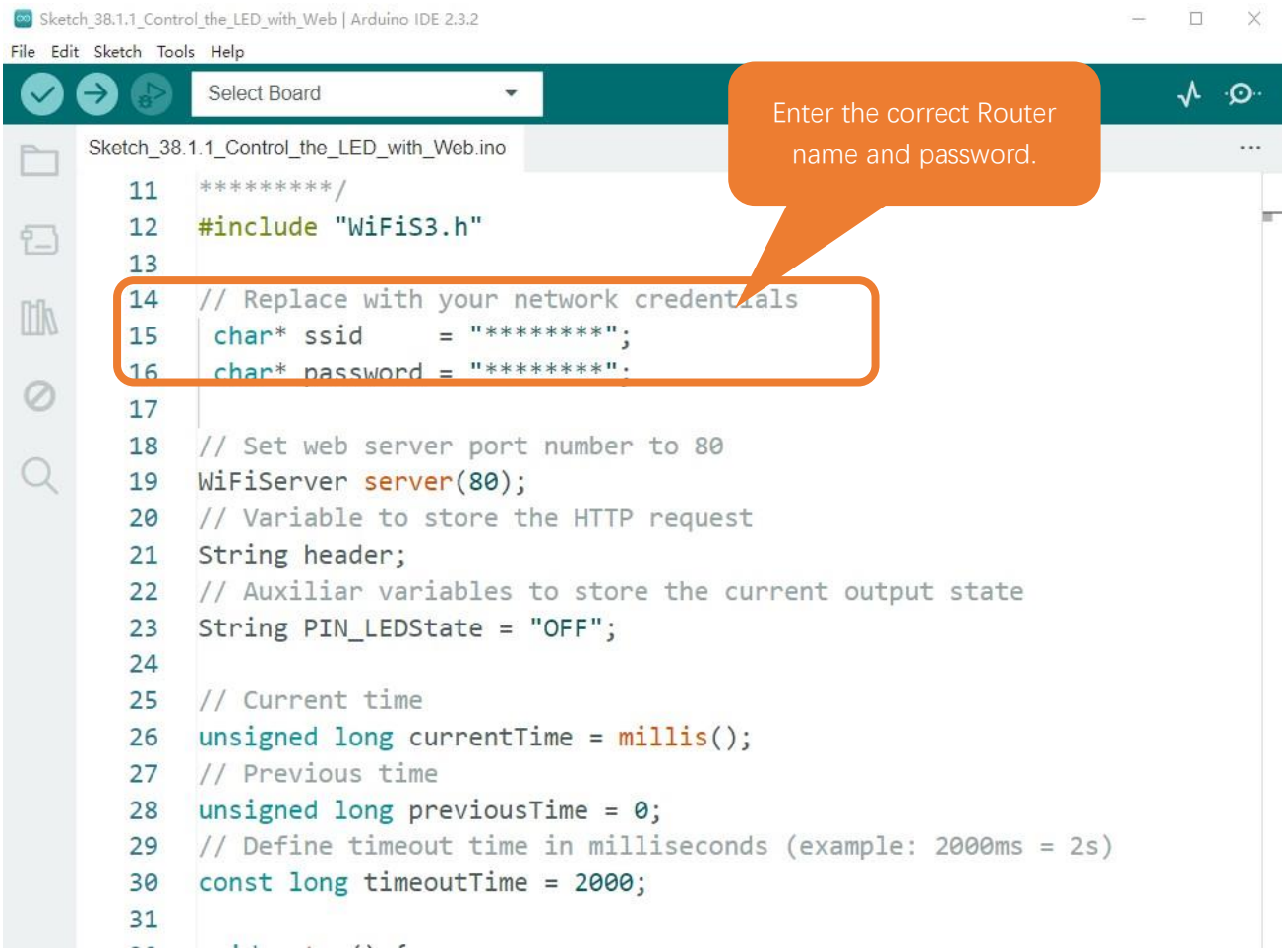
Circuit

Connect the board to the computer using the USB cable.

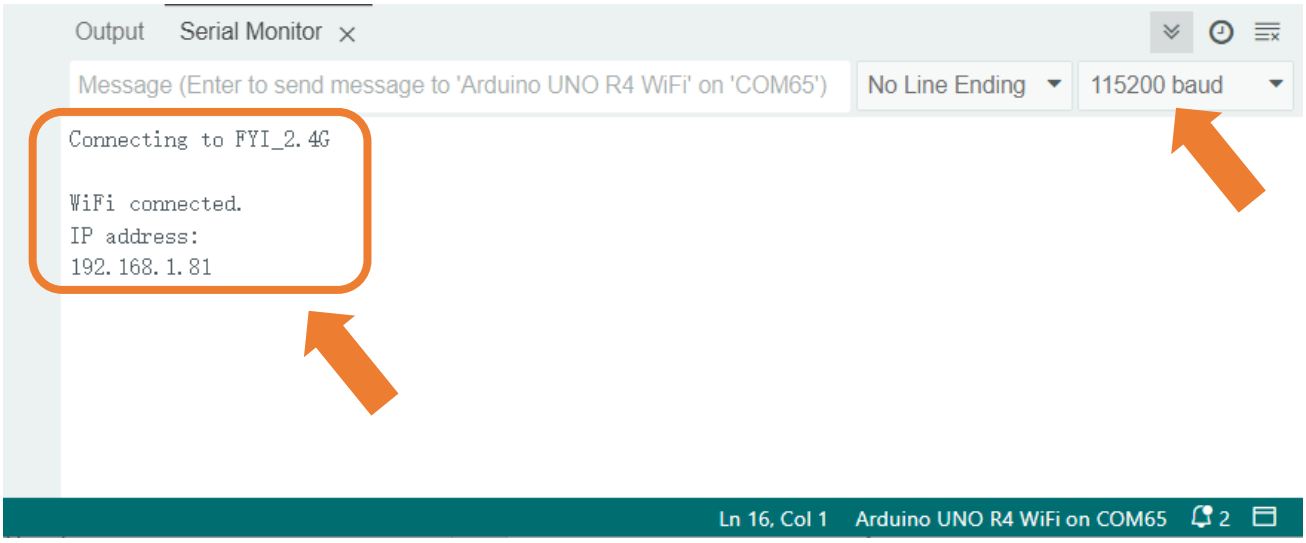


Sketch

Sketch_6.1.1



Upload the code to the control board, open the serial monitor and set the board rate to 115200. After the board connects to WiFi successfully, the IP address will be printed out, as shown below.



When Control Board successfully connects to "ssid_Router", serial monitor will print out the IP address assigned to Control Board by the router. Access http://192.168.1.26 in a computer browser on the LAN. As shown in the following figure:

You can click the corresponding button to control the LED on and off.
The following is the program code:

```
1  #include "WiFiS3.h"
2
3  // Replace with your network credentials
4   char* ssid     = "********";
```

```
5   char* password = "********";
6
7   // Set web server port number to 80
8   WiFiServer server(80);
9   // Variable to store the HTTP request
10  String header;
11  // Auxiliar variables to store the current output state
12  String PIN_LEDState = "OFF";
13
14  // Current time
15  unsigned long currentTime = millis();
16  // Previous time
17  unsigned long previousTime = 0;
18  // Define timeout time in milliseconds (example: 2000ms = 2s)
19  const long timeoutTime = 2000;
20
21  void setup() {
22    Serial.begin(115200);
23    // Initialize the output variables as outputs
24    pinMode(LED_BUILTIN, OUTPUT);
25    digitalWrite(LED_BUILTIN, LOW);
26
27    // Connect to Wi-Fi network with SSID and password
28    Serial.print("Connecting to ");
29    Serial.println(ssid);
30    WiFi.begin(ssid, password);
31    while (WiFi.status() != WL_CONNECTED) {
32      delay(500);
33      Serial.print(".");
34    }
35    // Print local IP address and start web server
36    Serial.println("");
37    Serial.println("WiFi connected.");
38    Serial.println("IP address: ");
39    Serial.println(WiFi.localIP());
40    server.begin();
41  }
42  void loop() {
43    WiFiClient client = server.available();  // Listen for incoming clients
44    if (client) {                           // If a new client connects,
45      Serial.println("New Client.");         // print a message out in the serial port
46      String currentLine = "";              // make a String to hold incoming data from the
    client
47      currentTime = millis();
```

```
48      previousTime = currentTime;
49      while (client.connected() && currentTime - previousTime <= timeoutTime) {  // loop while
    the client's connected
50        currentTime = millis();
51        if (client.available()) {  // if there's bytes to read from the client,
52          char c = client.read();  // read a byte, then
53          Serial.write(c);         // print it out the serial monitor
54          header += c;
55          if (c == '\n') {  // if the byte is a newline character
56            // if the current line is blank, you got two newline characters in a row.
57            // that's the end of the client HTTP request, so send a response:
58            if (currentLine.length() == 0) {
59              // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
60              // and a content-type so the client knows what's coming, then a blank line:
61              client.println("HTTP/1.1 200 OK");
62              client.println("Content-type:text/html");
63              client.println("Connection: close");
64              client.println();
65              // turns the GPIOs on and off
66              if (header.indexOf("GET /LED_BUILTIN/ON") >= 0) {
67                Serial.println("LED_BUILTIN ON");
68                PIN_LEDState = "ON";
69                digitalWrite(LED_BUILTIN, HIGH);
70              } else if (header.indexOf("GET /LED_BUILTIN/OFF") >= 0) {
71                Serial.println("LED_BUILTIN OFF");
72                PIN_LEDState = "OFF";
73                digitalWrite(LED_BUILTIN, LOW);
74              }
75              // Display the HTML web page
76              client.println("<!DOCTYPE html><html>");
77              client.println("<head> <title>Control Board Web Server</title> <meta
    name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
78              client.println("<link rel=\"icon\" href=\"data:,\">");
79              // CSS to style the on/off buttons
80              // Feel free to change the background-color and font-size attributes to fit your
    preferences
81              client.println("<style>html {font-family: Helvetica; display:inline-block; margin:
    0px auto; text-align: center;}");
82              client.println(" h1{color: #0F3376; padding: 2vh;} p{font-size: 1.5rem;}");
83              client.println(".button{background-color: #4286f4; display: inline-block; border:
    none; border-radius: 4px; color: white; padding: 16px 40px;text-decoration: none; font-size:
    30px; margin: 2px; cursor: pointer;}");
84
85
```

```
86              client.println(".button2{background-color: #4286f4;display: inline-block; border: none; border-radius: 4px; color: white; padding: 16px 40px;text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}</style></head>");
87              // Web Page Heading
88              client.println("<body><h1>Control Board Web Server</h1>");
89              client.println("<p>GPIO state: " + PIN_LEDState + "</p>");
90              client.println("<p><a href=\"/LED_BUILTIN/ON\"><button class=\"button button2\">ON</button></a></p>");
91              client.println("<p><a href=\"/LED_BUILTIN/OFF\"><button class=\"button button2\">OFF</button></a></p>");
92              client.println("</body></html>");
93              // The HTTP response ends with another blank line
94              client.println();
95              // Break out of the while loop
96              break;
97            } else {  // if you got a newline, then clear currentLine
98              currentLine = "";
99            }
100         } else if (c != '\r') {  // if you got anything else but a carriage return character,
101           currentLine += c;      // add it to the end of the currentLine
102         }
103       }
104     }
105     // Clear the header variable
106     header = "";
107     // Close the connection
108     client.stop();
109     Serial.println("Client disconnected.");
110     Serial.println("");
111   }
112 }
```

Include the WiFi Library header file of Control Board.

```
1   #include "WiFiS3.h"
```

Enter correct router name and password.

```
3   // Replace with your network credentials
4    char* ssid     = "********";
5    char* password = "********";
```

Set Control Board in Station mode and connect it to your router.

```
30    WiFi.begin(ssid, password);
```

Check whether Control Board has connected to router successfully every 0.5s.

```
31    while (WiFi.status() != WL_CONNECTED) {
32      delay(500);
```

```
33      Serial.print(".");
34    }
```

Serial monitor prints out the IP address assigned to Control Board.

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

Click the button on the web page to control the LED light on and off.

```
65              // turns the GPIOs on and off
66              if (header.indexOf("GET /LED_BUILTIN/ON") >= 0) {
67                Serial.println("LED_BUILTIN ON");
68                PIN_LEDState = "ON";
69                digitalWrite(LED_BUILTIN, HIGH);
70              } else if (header.indexOf("GET /LED_BUILTIN/OFF") >= 0) {
71                Serial.println("LED_BUILTIN OFF");
72                PIN_LEDState = "OFF";
73                digitalWrite(LED_BUILTIN, LOW);
74              }
```

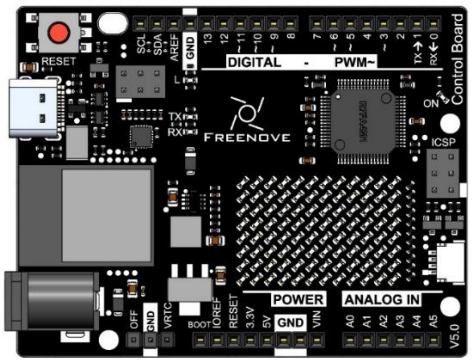
Chapter 7 Bluetooth

In this chapter, we engage with the onboard Bluetooth module. Bluetooth and Wi-Fi are common wireless communication technologies.

Project 7.1 Bluetooth Low Energy Data Passthrough

this section, we first learn how to make simple data transmission between the control board (Bluetooth) and our phone.

Component List

Control board x1 	USB cable x1 

Component knowledge

Control board's integrated Bluetooth function Bluetooth is a short-distance communication system, which can be divided into two types, namely Bluetooth Low Energy(BLE) and Classic Bluetooth. There are two modes for simple data transmission: master mode and slave mode.

Master mode

In this mode, works are done in the master device and it can connect with a slave device. And we can search and select slave devices nearby to connect with. When a device initiates connection request in master mode, it requires information of the other Bluetooth devices including their address and pairing passkey. After finishing pairing, it can connect with them directly.

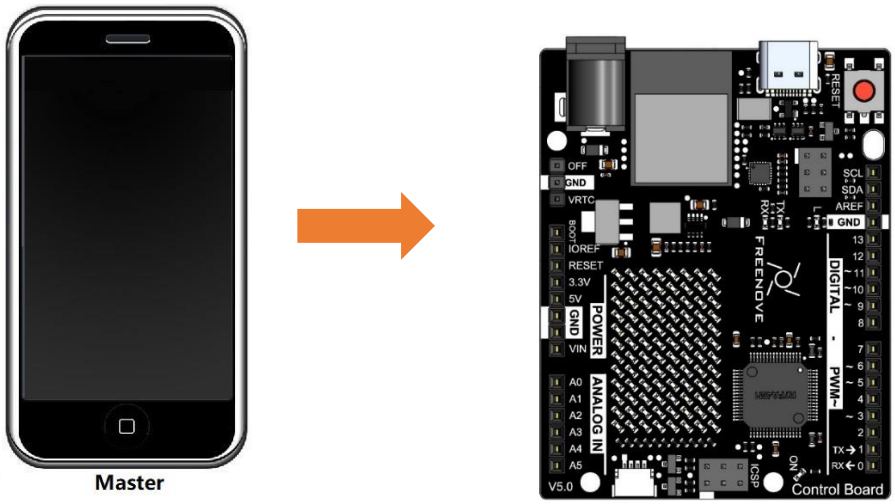
Slave mode

The Bluetooth module in slave mode can only accept connection request from a host computer, but cannot initiate a connection request. After connecting with a host device, it can send data to or receive from the host device.

Bluetooth devices can make data interaction with each other, as one is in master mode and the other in slave mode. When they are making data interaction, the Bluetooth device in master mode searches and selects devices nearby to connect to. When establishing connection, they can exchange data. When mobile phones exchange data with Control board, they are usually in master mode and Control board in slave mode.

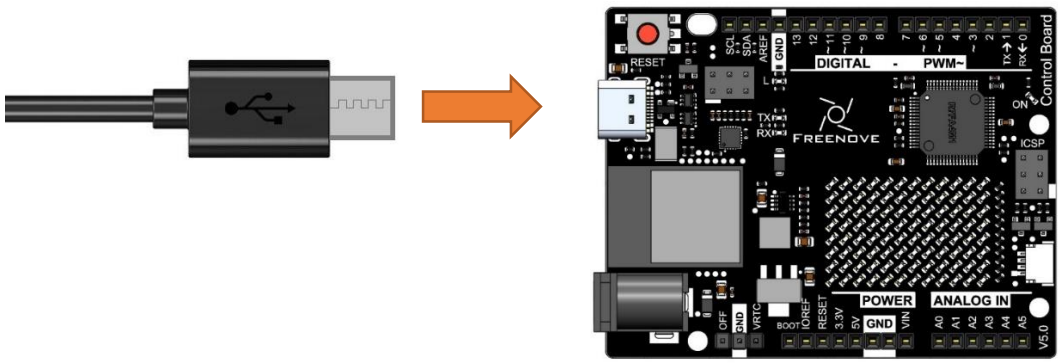
The control board is equipped with an ESP32 module, which provides Bluetooth and Wi-Fi functions, supporting speeds up to 2Mbps. The ESP32 module integrates a tracking antenna, which allows you to take advantage of the development board's connectivity without an external antenna.

However, It is worth noting that the Wi-Fi and Bluetooth modules share the same antenna, which means you cannot use Bluetooth and Wi-Fi at the same time.



Circuit

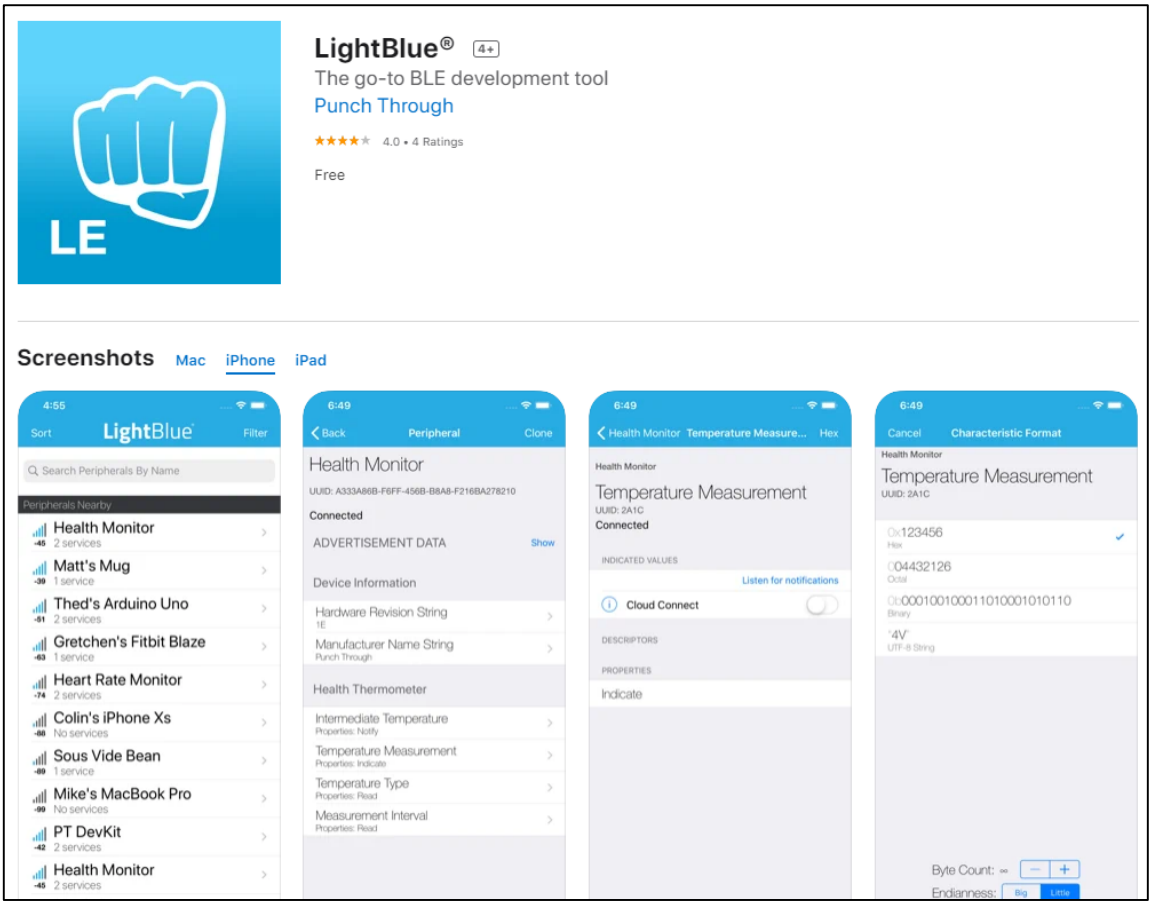
Connect Freenove Control board to the computer using the USB cable.



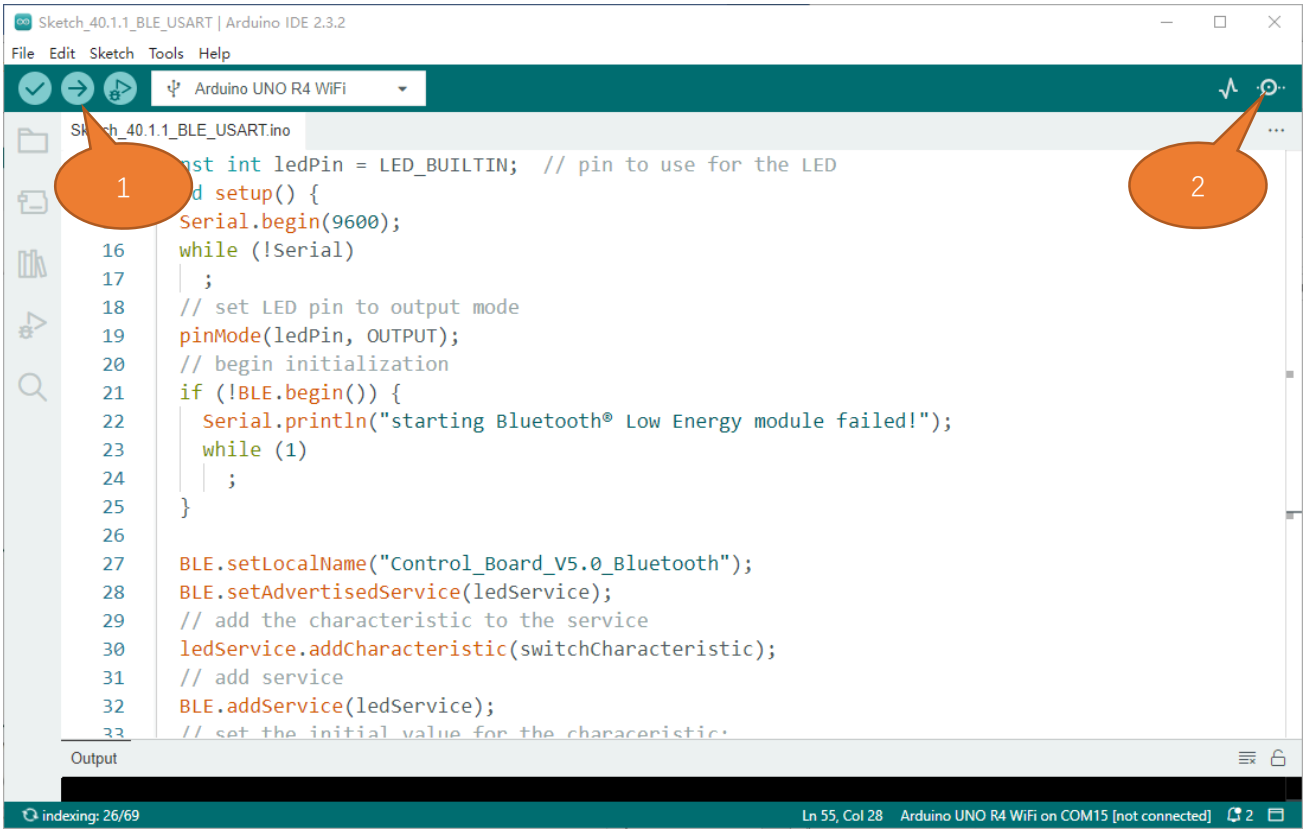
Sketch

Lightblue

If you can't install Serial Bluetooth on your phone, try LightBlue.If you do not have this software installed on your phone, you can refer to this link:
https://apps.apple.com/us/app/lightblue/id557428110#?platform=iphone.



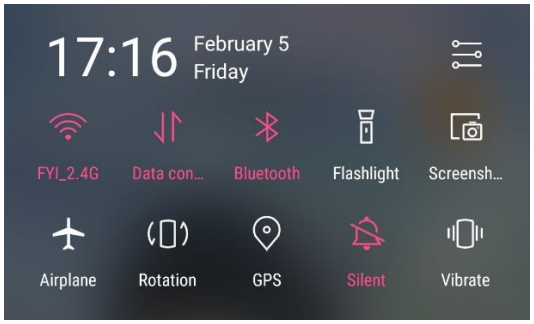
Step1. Upload the code of Project 7.1 to control board.
Step2. Click on serial monitor.



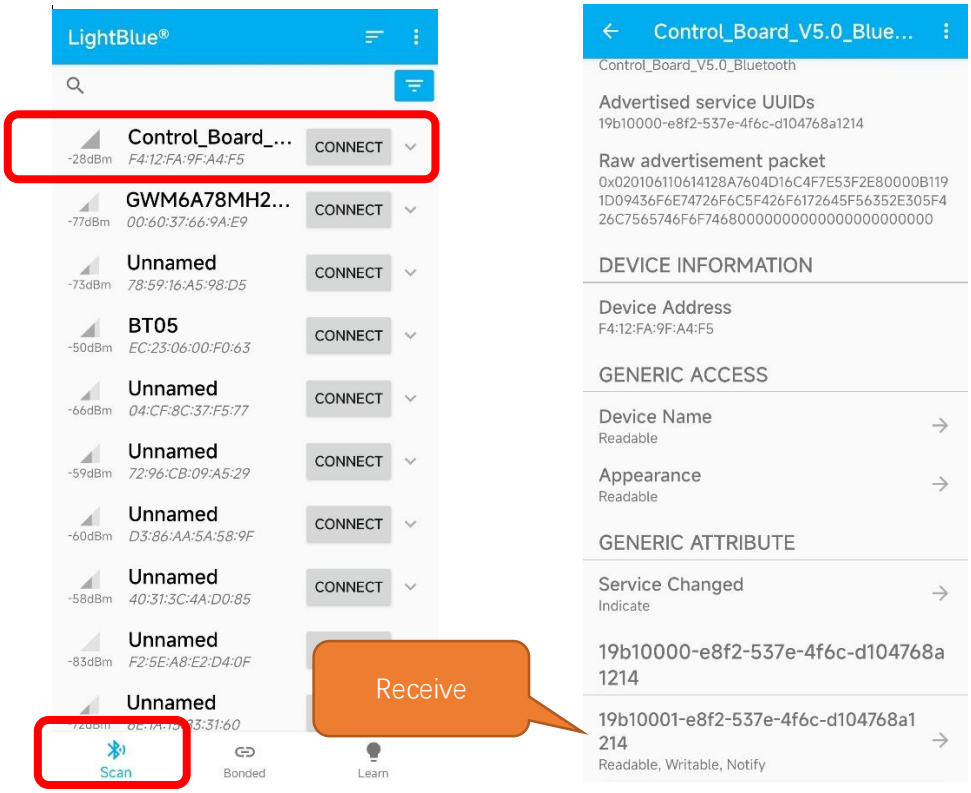

```
const int ledPin = LED_BUILTIN;  // pin to use for the LED
void setup() {
Serial.begin(9600);
while (!Serial)
  ;
// set LED pin to output mode
pinMode(ledPin, OUTPUT);
// begin initialization
if (!BLE.begin()) {
  Serial.println("starting Bluetooth® Low Energy module failed!");
  while (1)
    ;
}

BLE.setLocalName("Control_Board_V5.0_Bluetooth");
BLE.setAdvertisedService(ledService);
// add the characteristic to the service
ledService.addCharacteristic(switchCharacteristic);
// add service
BLE.addService(ledService);
```

Step3. Set baud rate to 9600.

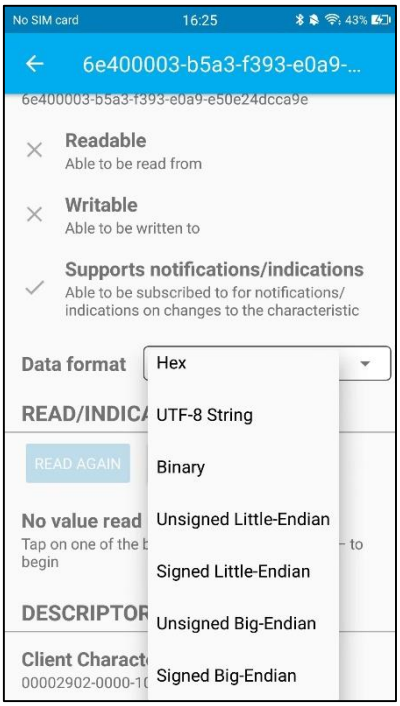
Turn ON Bluetooth on your phone, and open the Lightblue APP.

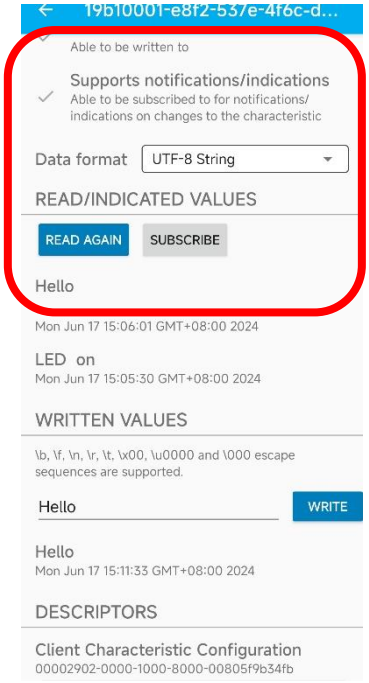


In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click Control_Board_V5.0_Bluetooth.



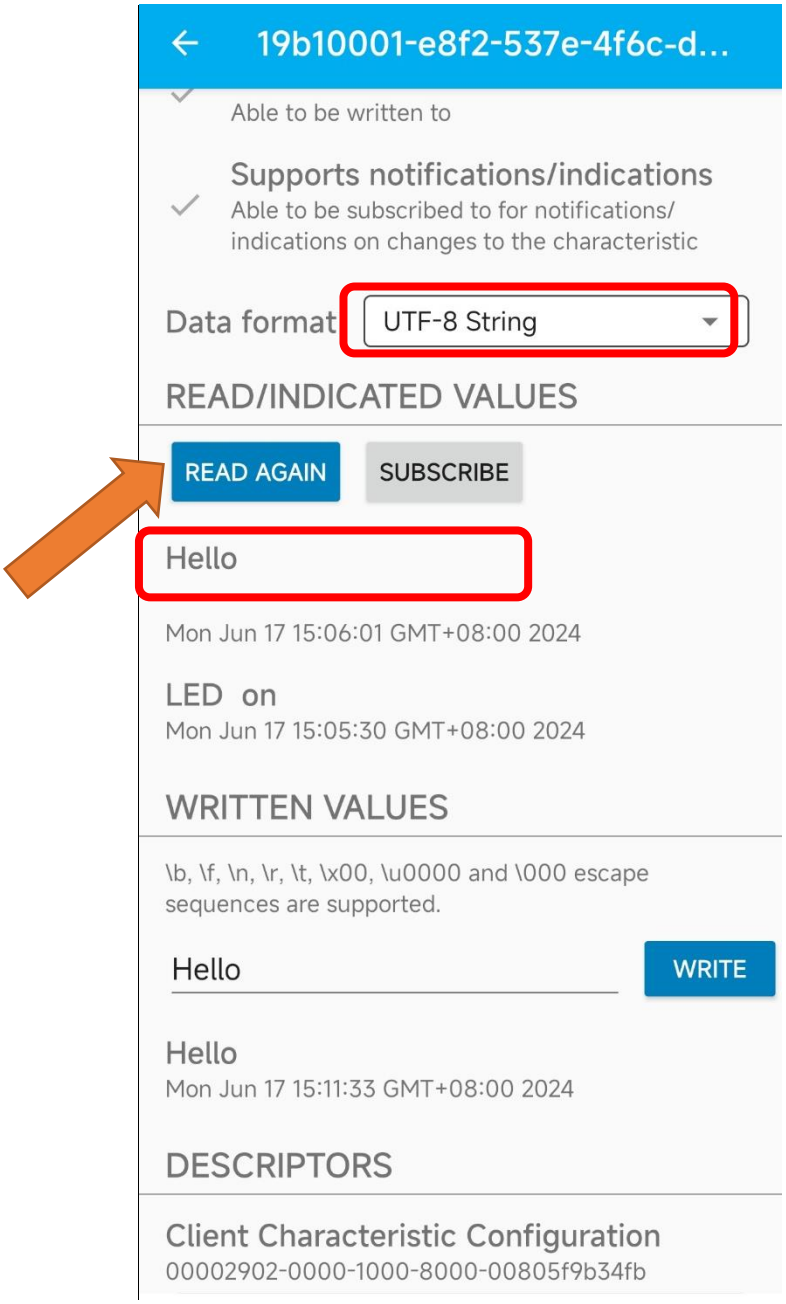
Click “Receive”. Select the appropriate Data format in the box to the right of Data Format. For example, HEX for hexadecimal, utf-string for character, Binary for Binary, etc. Then click SUBSCRIBE.



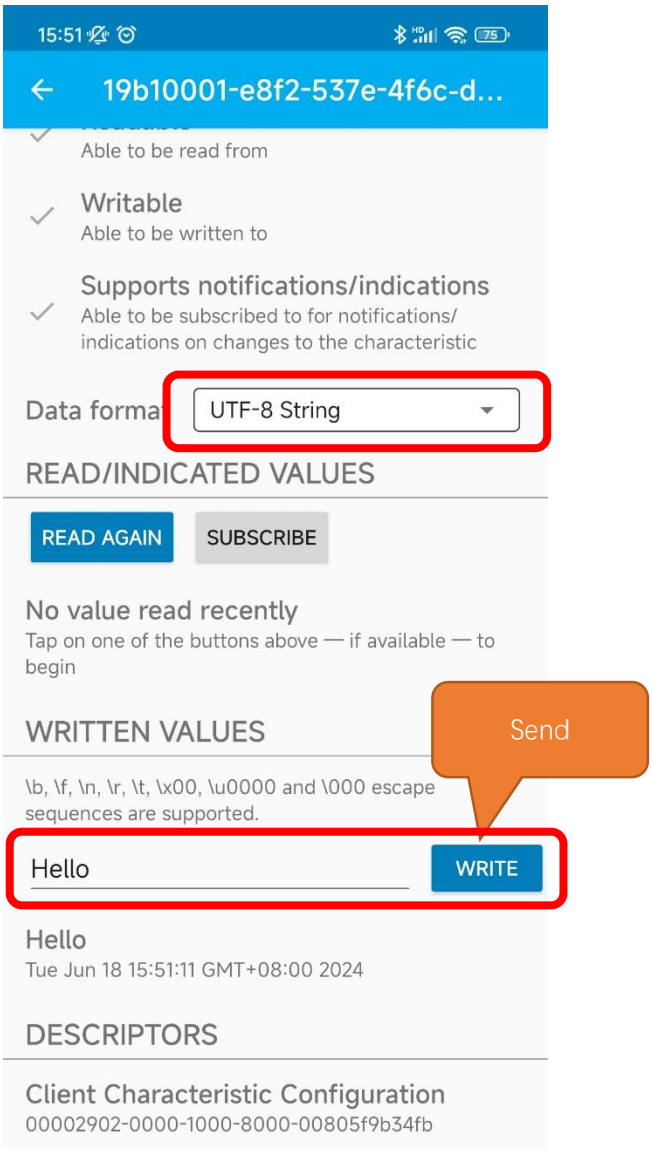


Back to the serial monitor on your computer. You can type anything in the left border of Send, and then click Send.

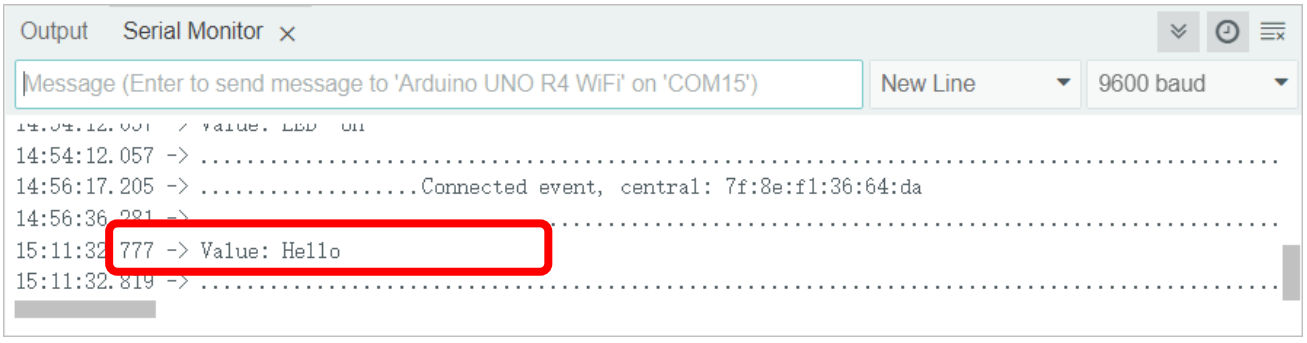
The last received data will be printed when the READ AGAIN button on the app is tapped.



Similarly, you can select “Send” on your phone. Set Data format, and then enter anything in the sending box and click Write to send.



And the computer will receive the message from the mobile Bluetooth.



And now data can be transferred between your mobile phone and computer via control board.
The following is the program code:

```
#include <ArduinoBLE.h>
#include "String.h"

BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214");  // Bluetooth® Low Energy LED Service
// Bluetooth® Low Energy LED Switch Characteristic - custom 128-bit UUID, read and writable by central
BLECharacteristic switchCharacteristic("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite | BLENotify | BLEBroadcast, 20);

String inputString = "";      // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete
long lasttime = 0;

const int ledPin = LED_BUILTIN;  // pin to use for the LED
void setup() {
  Serial.begin(9600);
  while (!Serial)
    ;
  // set LED pin to output mode
  pinMode(ledPin, OUTPUT);
  // begin initialization
  if (!BLE.begin()) {
    Serial.println("starting Bluetooth® Low Energy module failed!");
    while (1)
      ;
  }

  BLE.setLocalName("Control_Board_V5.0_Bluetooth");
  BLE.setAdvertisedService(ledService);
  // add the characteristic to the service
  ledService.addCharacteristic(switchCharacteristic);
  // add service
  BLE.addService(ledService);
  // set the initial value for the characeristic:
  switchCharacteristic.writeValue("LED");

  BLE.setEventHandler(BLEConnected, blePeripheralConnectHandler);
  BLE.setEventHandler(BLEDisconnected, blePeripheralDisconnectHandler);

  switchCharacteristic.setEventHandler(BLEWritten, switchCharacteristicWritten);

```

```
41    // start advertising
42    BLE.advertise();
43
44    Serial.println("Bluetooth® device active, waiting for connections...");
45  }
46
47  void loop() {
48    long nowtime = millis();
49    if (nowtime - lasttime > 1000) {
50      BLE.poll();
51      Serial.print(".");
52      if (Serial.available() > 0) {
53        String str = Serial.readString();
54        const char *newValue = str.c_str();
55        switchCharacteristic.writeValue(newValue);
56      }
57      lasttime = nowtime;
58    }
59  }
60
61  void blePeripheralConnectHandler(BLEDevice central) {
62    Serial.print("Connected event, central: ");
63    Serial.println(central.address());
64  }
65
66  void blePeripheralDisconnectHandler(BLEDevice central) {
67    Serial.print("Disconnected event, central: ");
68    Serial.println(central.address());
69  }
70
71  void switchCharacteristicWritten(BLEDevice central, BLECharacteristic characteristic) {
72    Serial.print("Characteristic event, written: ");
73    uint8_t characteristicValue[20];
74    int bytesRead = characteristic.readValue(characteristicValue, sizeof(characteristicValue));
75    Serial.print("Received bytes: ");
76    for (int i = 0; i < bytesRead; i++) {
77      Serial.print(characteristicValue[i], HEX);
78      Serial.print(" ");
79    }
80    Serial.println();
81    String receivedString = "";
82    for (int i = 0; i < bytesRead; i++) {
83      receivedString += (char)characteristicValue[i];
84    }
```

```
85    Serial.println("Value: " + receivedString);
86  }
```

Initialize the BLE function and name it.

```
27    BLE.setLocalName("Control_Board_V5.0_Bluetooth");
```

Write a Callback function for BLE server to manage connection of BLE.

```
61  void blePeripheralConnectHandler(BLEDevice central) {
62    Serial.print("Connected event, central: ");
63    Serial.println(central.address());
64  }
65
66  void blePeripheralDisconnectHandler(BLEDevice central) {
67    Serial.print("Disconnected event, central: ");
68    Serial.println(central.address());
69  }
```

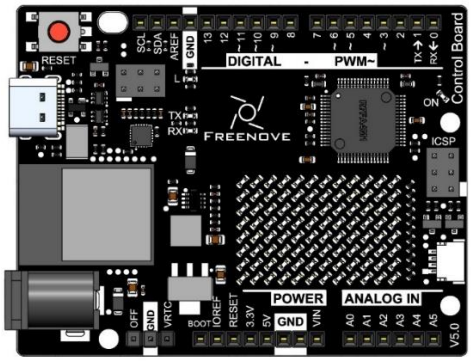
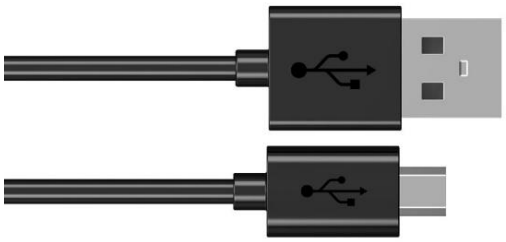
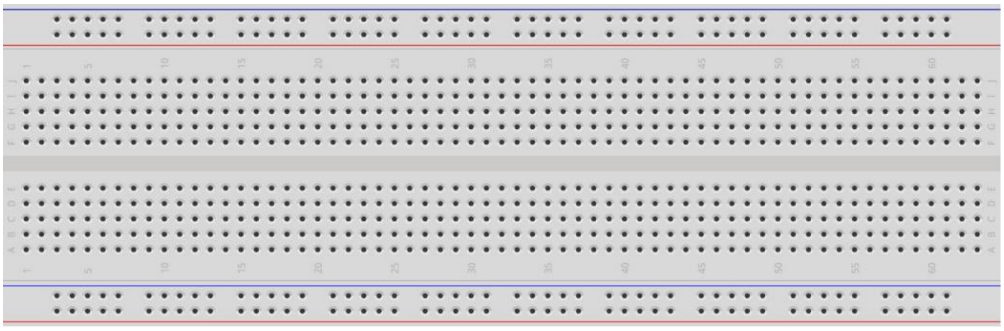
When the mobile phone send data to control board via BLE Bluetooth, it will print them out with serial port; When the serial port of control board receive data, it will send them to mobile via BLE Bluetooth.

```
52      if (Serial.available() > 0) {
53        String str = Serial.readString();
54        const char *newValue = str.c_str();
55        switchCharacteristic.writeValue(newValue);
56      }
...   ...
71  void switchCharacteristicWritten(BLEDevice central, BLECharacteristic characteristic) {
72    Serial.print("Characteristic event, written: ");
73    uint8_t characteristicValue[20];
74    int bytesRead = characteristic.readValue(characteristicValue, sizeof(characteristicValue));
75    Serial.print("Received bytes: ");
76    for (int i = 0; i < bytesRead; i++) {
77      Serial.print(characteristicValue[i], HEX);
78      Serial.print(" ");
79    }
80    Serial.println();
81    String receivedString = "";
82    for (int i = 0; i < bytesRead; i++) {
83      receivedString += (char)characteristicValue[i];
84    }
85    Serial.println("Value: " + receivedString);
86  }
```

Project 7.2 Control LED with Bluetooth

In this section, we will further learn how to use Bluetooth to control an LED.

Component List

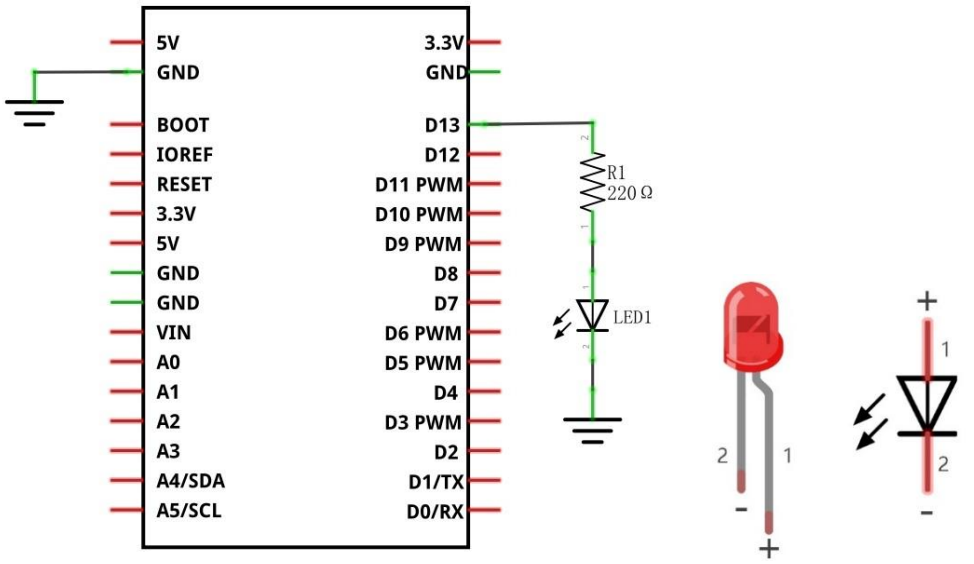
<table>
<tr><td>Control board x1
  </td><td colspan="3">USB cable x1
  </td></tr>
<tr><td>Micro USB Wire x1
  </td><td>LED x1
  </td><td>Resistor 220Ω x1
  </td><td>Jumper M/M x2
  </td></tr>
<tr><td colspan="4">Breadboard x1
  </td></tr>
</table>

If you do not have the above components in your kit, you can use the control board alone to finish the project. The control pin of the LED in this project is the same as the one controlling the onboard LED.

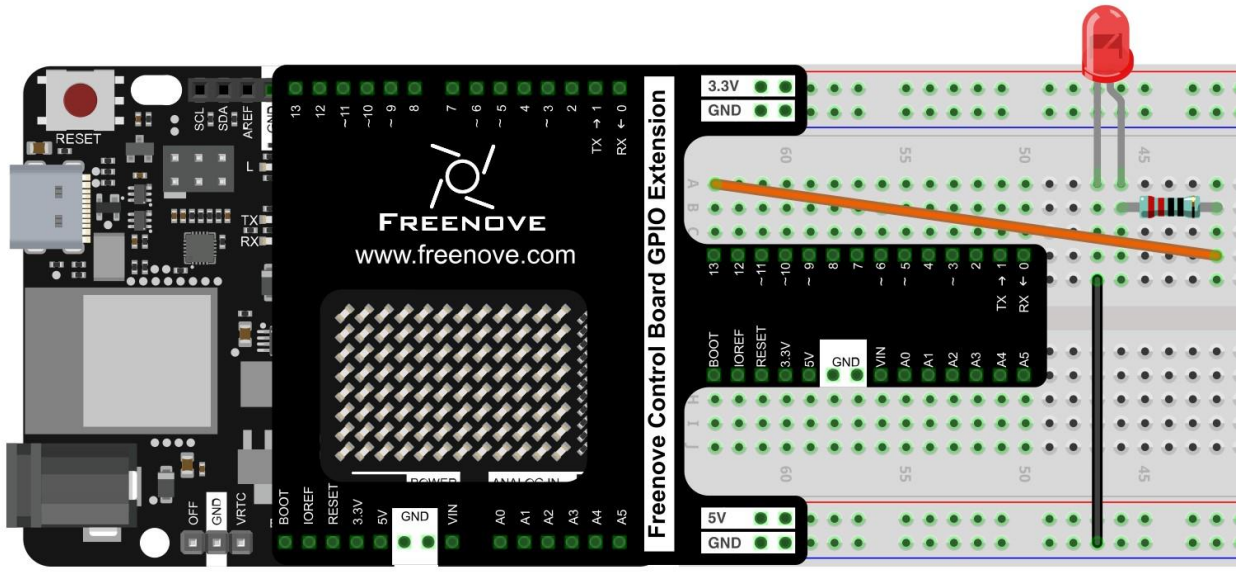
Circuit

Connect the control board to your computer with the USB cable.

Schematic diagram

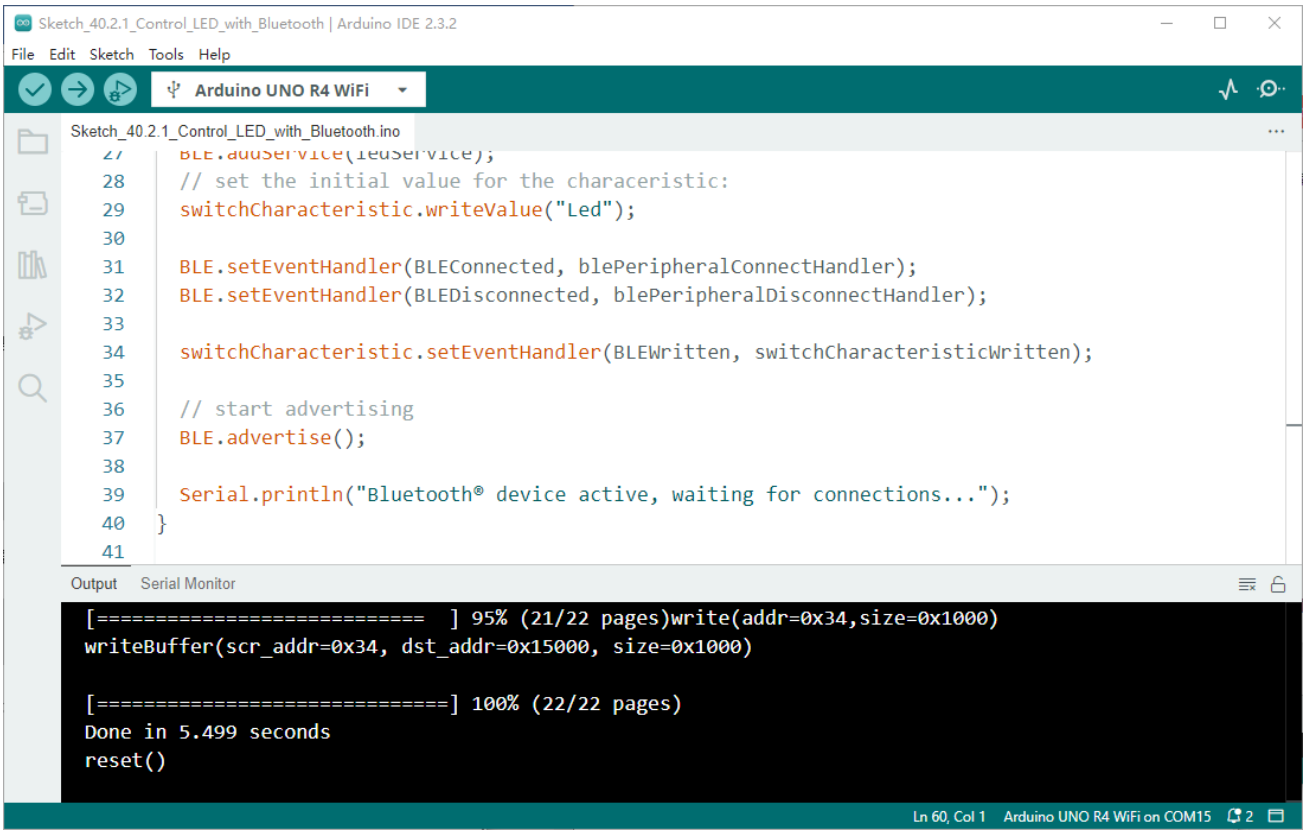


Hardware connection. **If you need any support, please contact us via: support@freenove.com**



Sketch

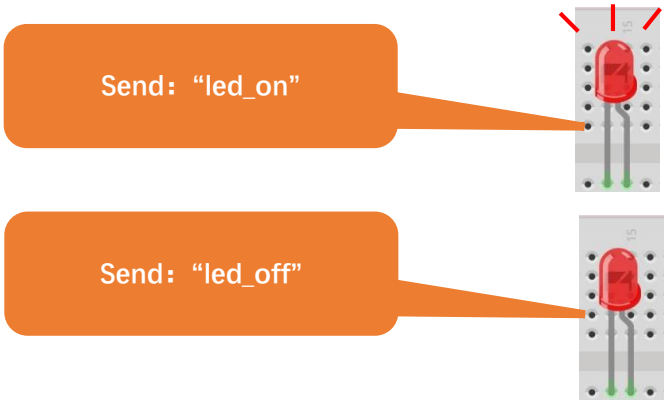
Sketch_7.2.1



Compile and upload code to Control board. The operation of the APP is the same as 40.1, you only need to change the sending content to "**led_on**" and "**led_off**" to operate LEDs on the Control board.
Data sent from mobile APP:

Display on the serial port of the computer:

The phenomenon of LED



Attention: If the sending content isn't "led-on' or "led-off", then the state of LED will not change. If the LED is on, when receiving irrelevant content, it keeps on; Correspondingly, if the LED is off, when receiving irrelevant content, it keeps off.

If you do not connect external LED circuit, you can check the status of the onboard LED.

With this example you can design more interesting projects.

The following is the program code:

```
#include <ArduinoBLE.h>
#include "String.h"

BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214");  // Bluetooth® Low Energy LED Service
// Bluetooth® Low Energy LED Switch Characteristic - custom 128-bit UUID, read and writable by central
BLECharacteristic switchCharacteristic("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite | BLENotify | BLEBroadcast, 20);

const int ledPin = LED_BUILTIN;  // pin to use for the LED
void setup() {
  Serial.begin(9600);
  while (!Serial)
    ;
  // set LED pin to output mode
  pinMode(ledPin, OUTPUT);
  // begin initialization
  if (!BLE.begin()) {
    Serial.println("starting Bluetooth® Low Energy module failed!");
    while (1)
      ;
  }

  BLE.setLocalName("Control_Board_V5.0_Bluetooth");
  BLE.setAdvertisedService(ledService);
  // add the characteristic to the service
  ledService.addCharacteristic(switchCharacteristic);
  // add service
  BLE.addService(ledService);
  // set the initial value for the characeristic:
  switchCharacteristic.writeValue("Led");

  BLE.setEventHandler(BLEConnected, blePeripheralConnectHandler);
  BLE.setEventHandler(BLEDisconnected, blePeripheralDisconnectHandler);

  switchCharacteristic.setEventHandler(BLEWritten, switchCharacteristicWritten);

  // start advertising
  BLE.advertise();

  Serial.println("Bluetooth® device active, waiting for connections...");
}
```

```
44
45  void loop() {
46    BLEDevice central = BLE.central();
47    if (central) {
48      central.poll();
49      delay(1000);
50      Serial.print(".");
51    }
52  }
53
54  void blePeripheralConnectHandler(BLEDevice central) {
55    Serial.print("Connected event, central: ");
56    Serial.println(central.address());
57  }
58
59  void blePeripheralDisconnectHandler(BLEDevice central) {
60    Serial.print("Disconnected event, central: ");
61    Serial.println(central.address());
62  }
63
64  void switchCharacteristicWritten(BLEDevice central, BLECharacteristic characteristic) {
65    Serial.print("Characteristic event, written: ");
66
67    uint8_t characteristicValue[20];
68    int bytesRead = characteristic.readValue(characteristicValue, sizeof(characteristicValue));
69
70    Serial.print("Received bytes: ");
71    for (int i = 0; i < bytesRead; i++) {
72      Serial.print(characteristicValue[i], HEX);
73      Serial.print(" ");
74    }
75    Serial.println();
76
77    String receivedString = "";
78
79    for (int i = 0; i < bytesRead; i++) {
80      receivedString += (char)characteristicValue[i];
81    }
82    Serial.println("Value: " + receivedString);
83    if (receivedString == "led_on") {
84      Serial.println("LED on");
85      digitalWrite(ledPin, HIGH);  // will turn the LED on
86    }
87    if (receivedString == "led_off") {  // a 0 value
```

```
88      Serial.println(F("LED off"));
89      digitalWrite(ledPin, LOW);  // will turn the LED off
90    }
91  }
```

Use character string to handle function header file.

```
2   #include "String.h"
```

Initialize the BLE Bluetooth and name it as "Control_Board_V5.0_Bluetooth"

```
25    BLE.setLocalName("Control_Board_V5.0_Bluetooth");
```

Compare the content in buffer array with "led_on" and "led_off" to see whether they are the same. If yes, execute the corresponding operation.

```
83  if (receivedString == "led_on") {
84      Serial.println("LED on");
85      digitalWrite(ledPin, HIGH);  // will turn the LED on
86    }
87    if (receivedString == "led_off") {  // a 0 value
88      Serial.println(F("LED off"));
89      digitalWrite(ledPin, LOW);  // will turn the LED off
90    }
```

What's Next?

THANK YOU for participating in this learning experience!

We have reached the end of this tutorial. If you find errors, omissions or you have suggestions and/or questions about this tutorial or component contents of this kit, please feel free to contact us:

support@freenove.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you are interesting in Processing, you can study the Processing.pdf in the unzipped folder.

If you want to learn more about Arduino, Raspberry Pi, micro:bit, robots, smart cars and other interesting products, please visit our website:

http://www.freenove.com/

We will continue to launch fun, cost-effective, innovative and exciting products.

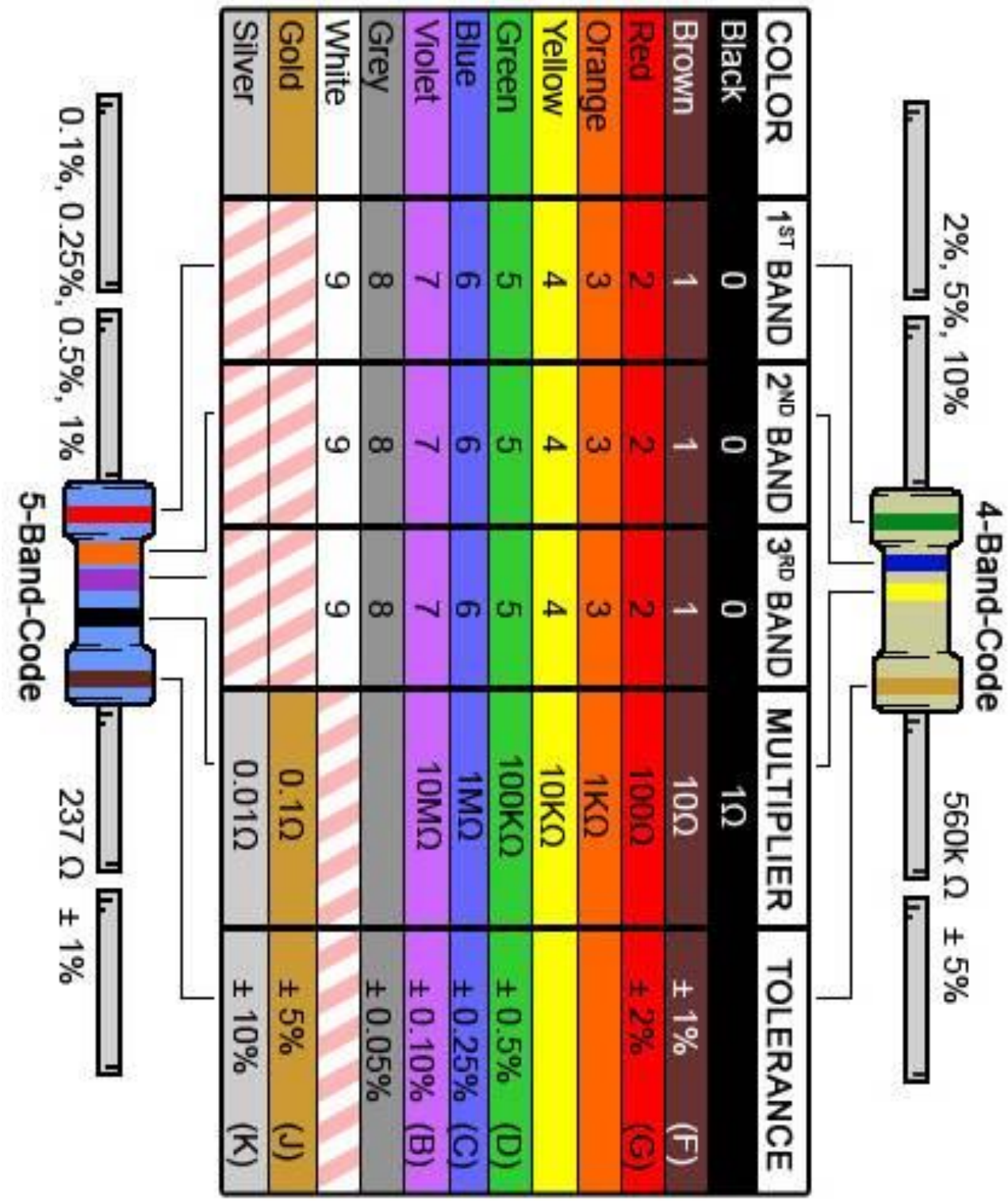
Thank you again for choosing Freenove products.

Appendix

ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Resistor Color Code



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)