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

All the files provided in the ZIP file are released under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License. You can find a copy of the license in the ZIP file.

It means you can use these files on your own derived works, in part or completely. But not for commercial use.

Freenove® brand and logo are trademarks of Freenove Creative Technology Co., Ltd. Must not be used without permission.

FREENOVE™

Other registered trademarks and their owners appearing in this document:

Arduino® is a trademark of Arduino LLC (https://www.arduino.cc/).
Raspberry Pi® is a trademark of Raspberry Pi Foundation (https://www.raspberrypi.org/).
micro:bit® is a trademark of Micro:bit Educational Foundation (https://www.microbit.org/).

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:

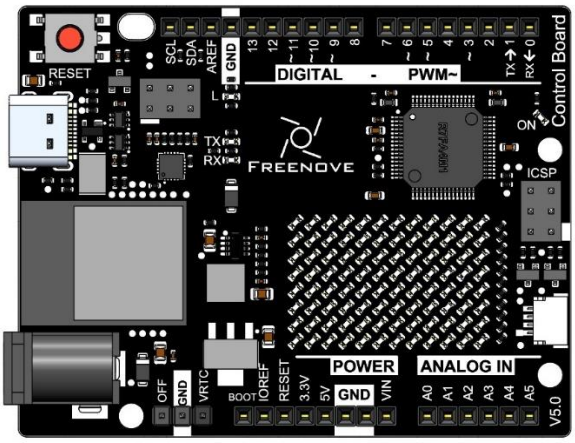
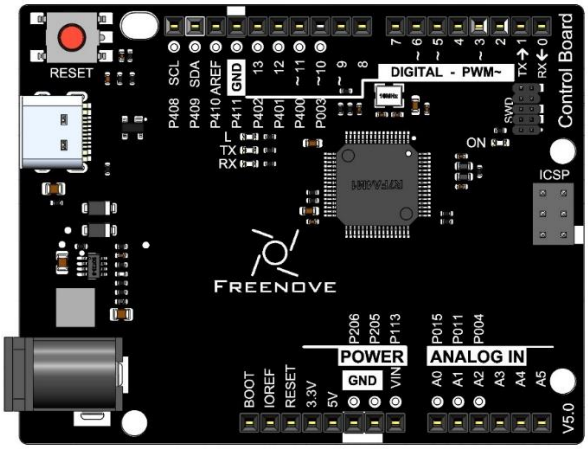
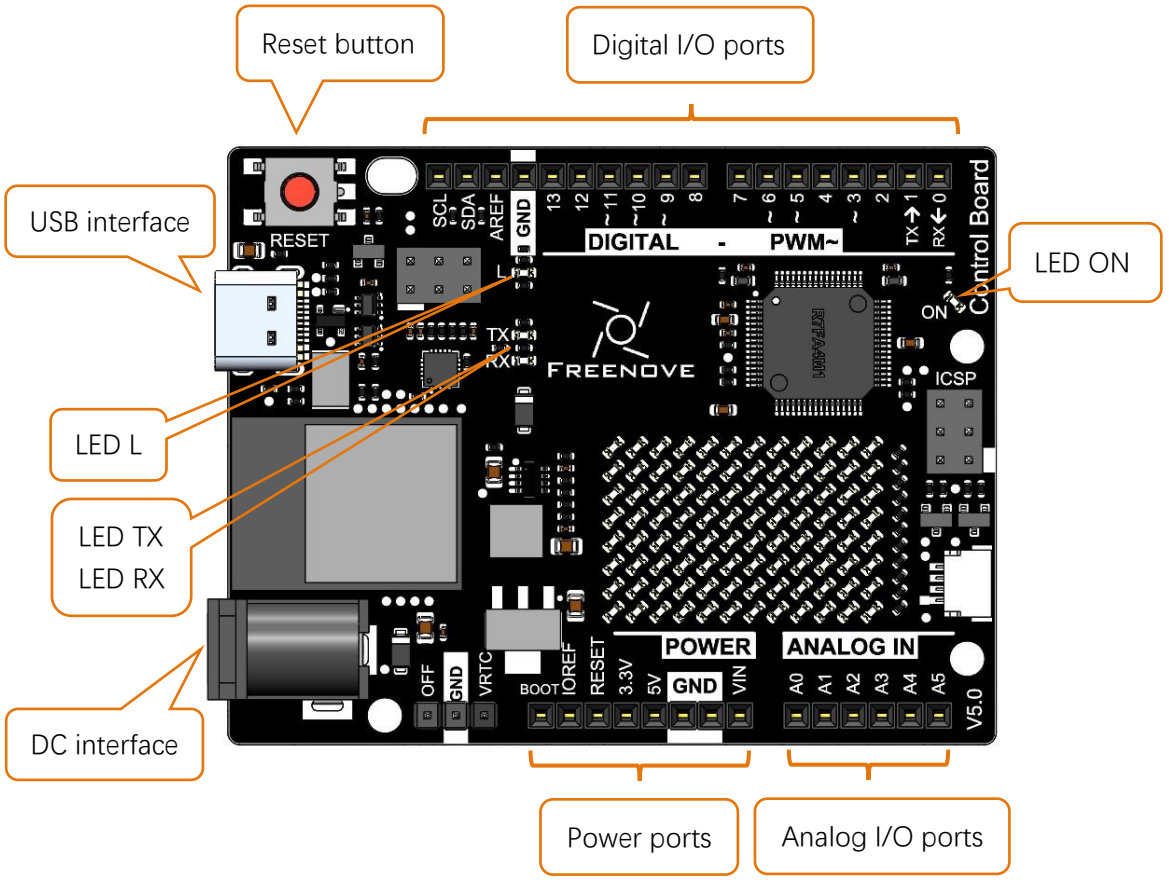
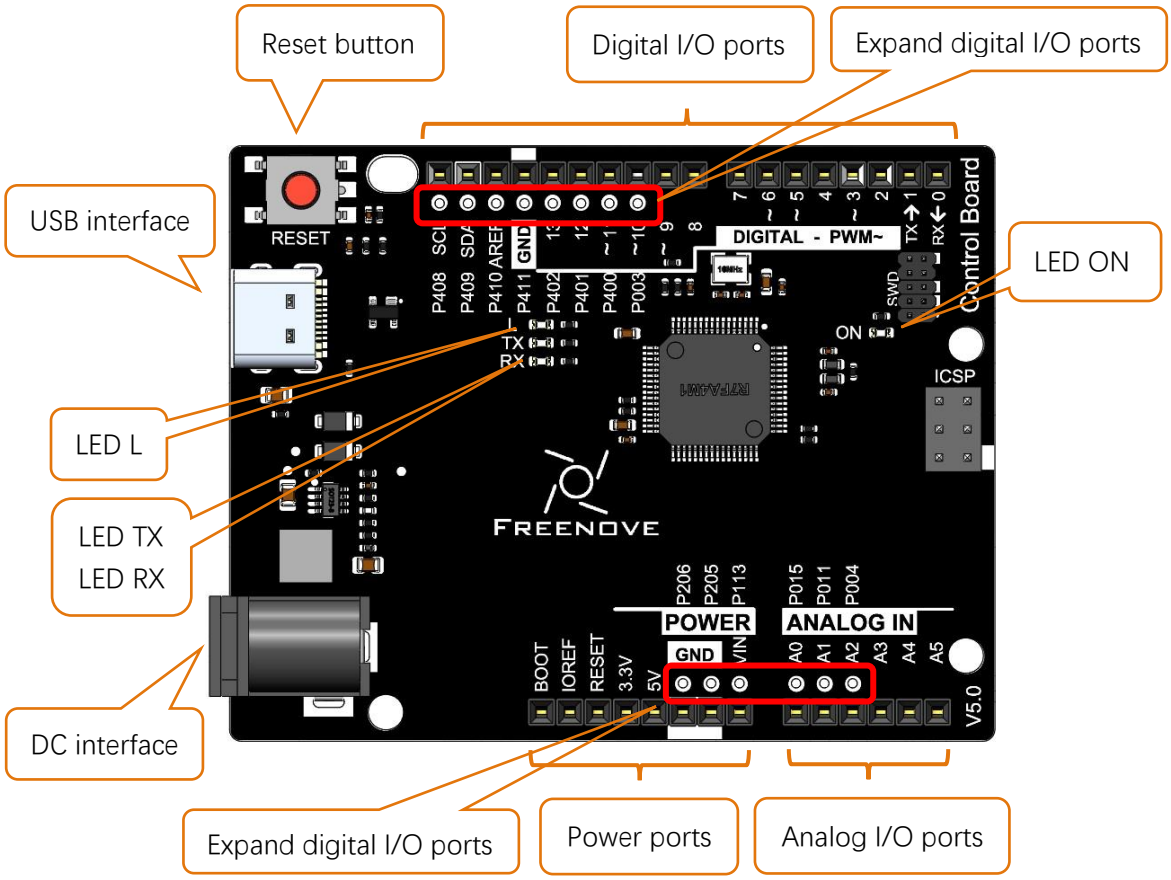
Freenove Control Board (WiFi)	Freenove Control Board (minina)
	

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)

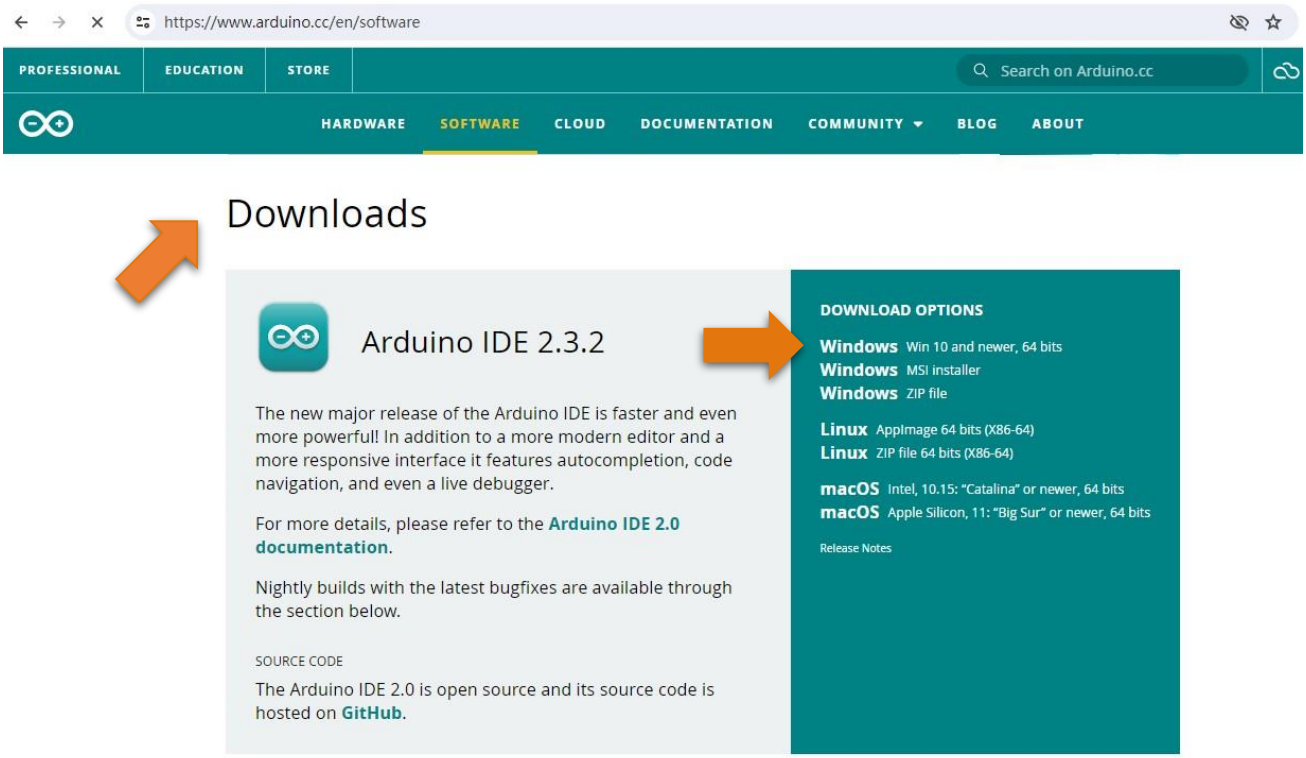
Before starting building the projects, you need to make some preparation first, which is so crucial that you must not skip.

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.)

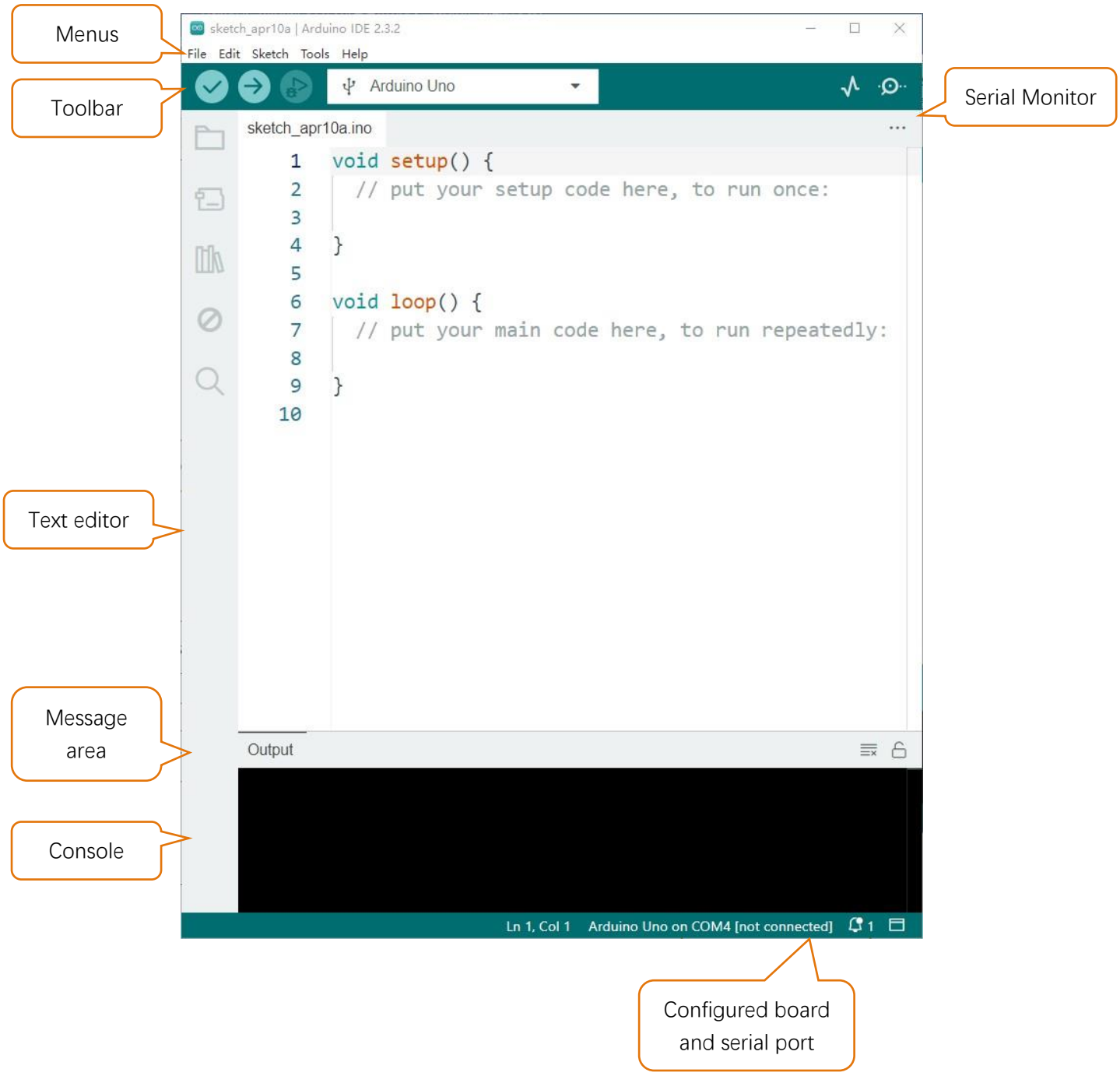
Arduino IDE uses C/C++ programming language. Don't worry if you have never used it, because this document contains programming knowledge and detailed explanation of the code.

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.

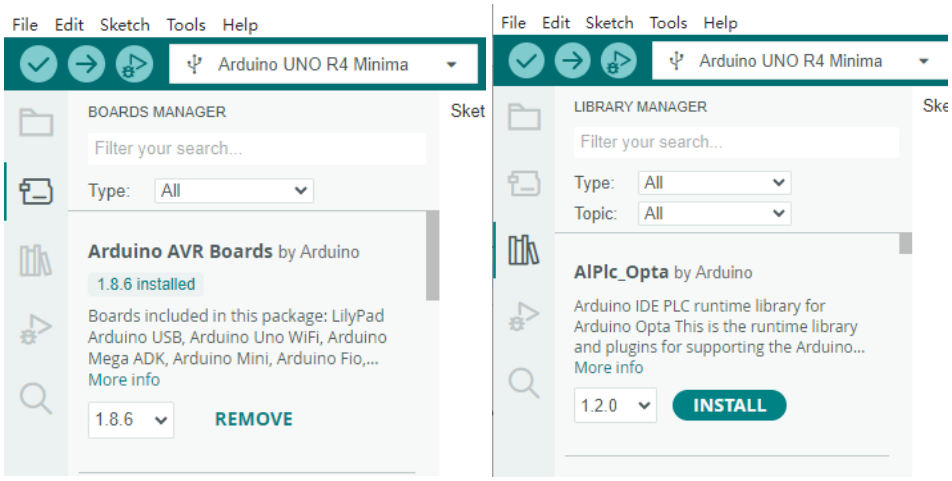
Open
Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Save
Saves your sketch.

Serial Monitor
Opens the serial monitor.

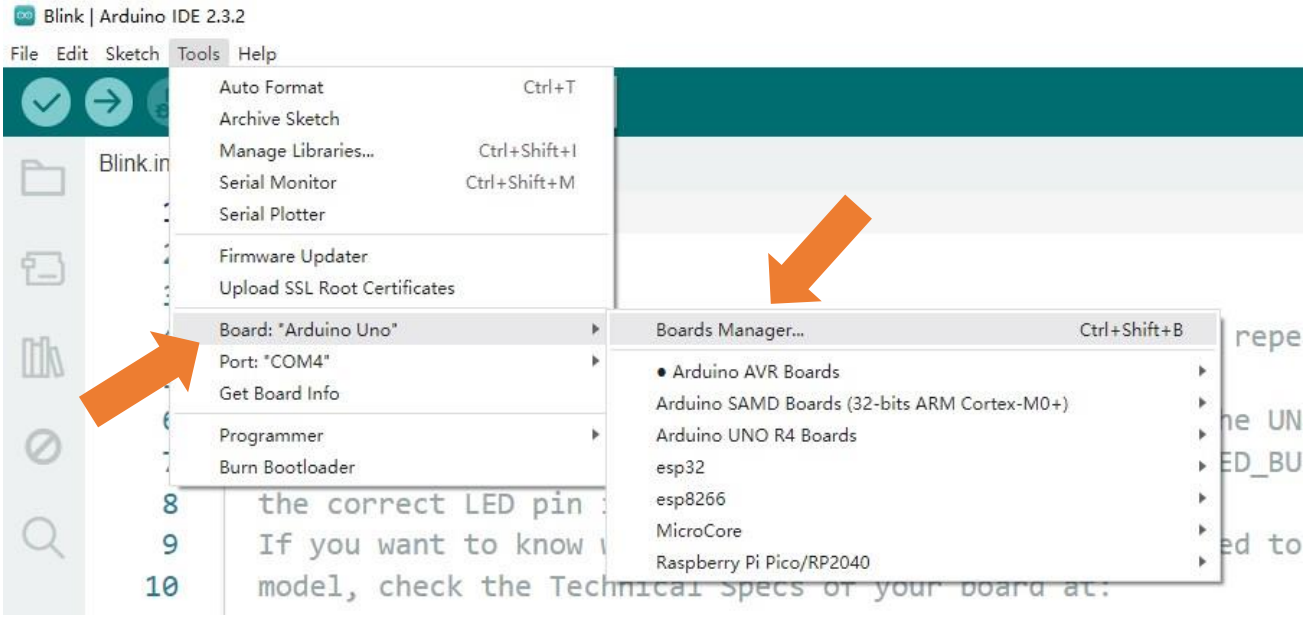
Additional commands are found within five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

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.

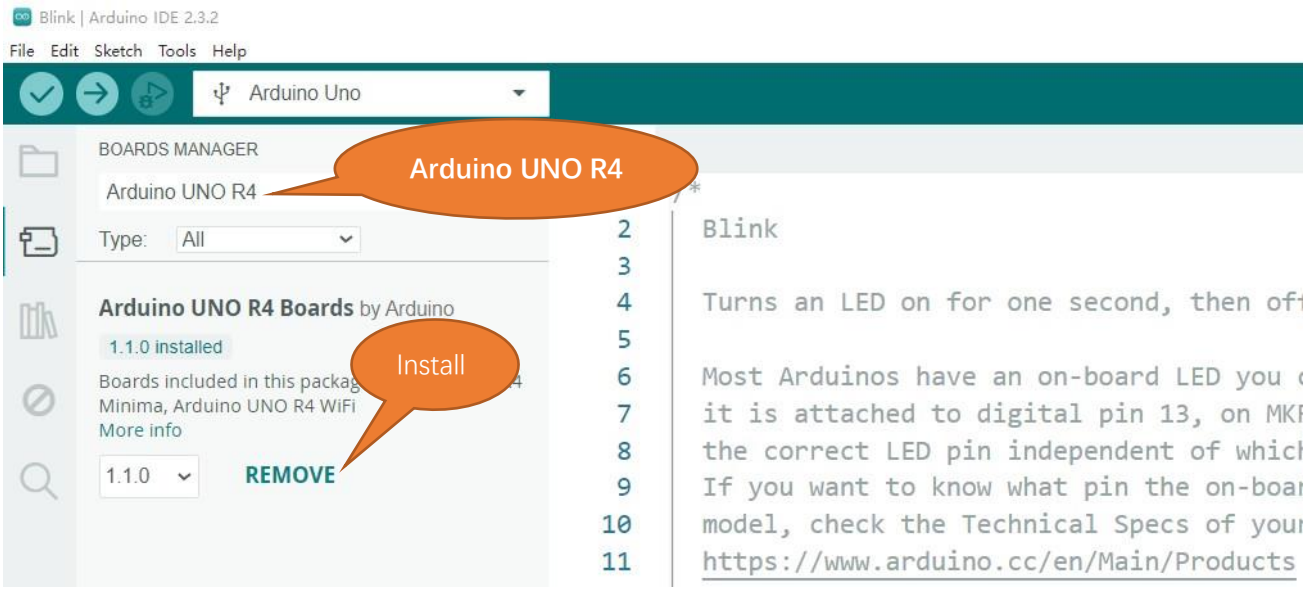


Installation of Development Board Support Package

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



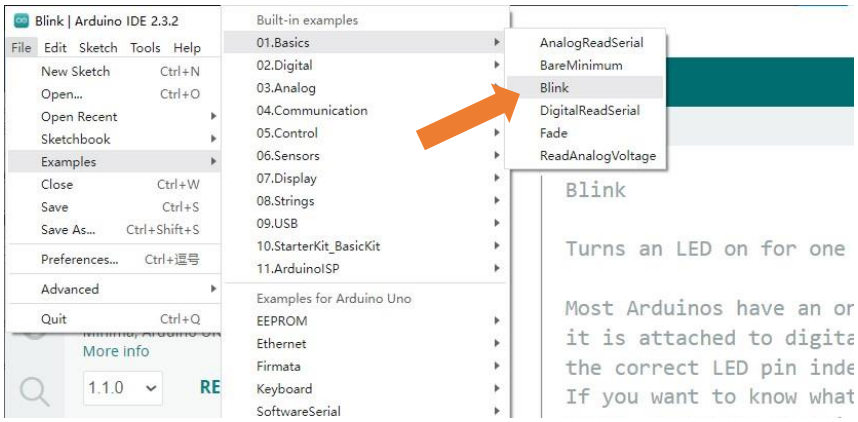
Enter **Arduino UNO R4** in the searching box, and select "**Arduino UNO R4**" and click on Install.



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.

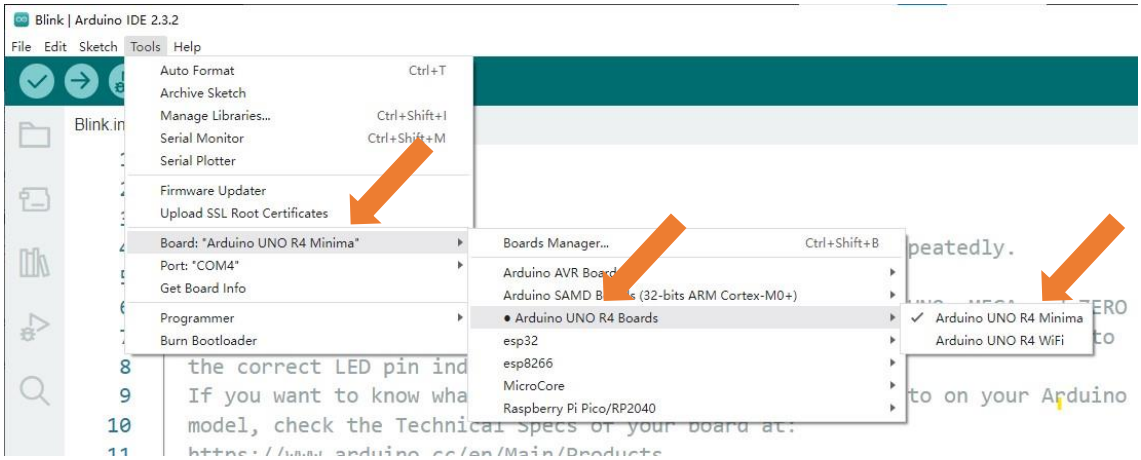
First Use

Open the example sketch "Blink".

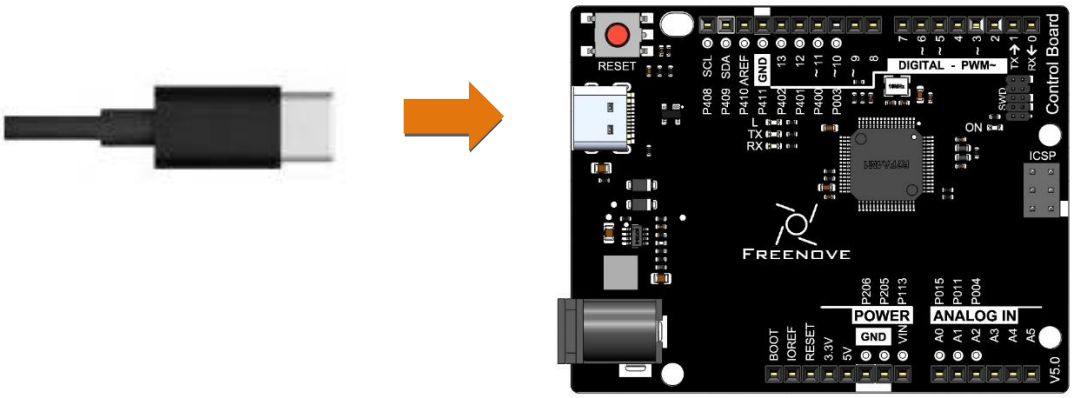


Select the board corresponding to the board you have in hands. Here we take Freenove Control Board (minina) as an example:

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



Connect control board to your computer with USB cable.



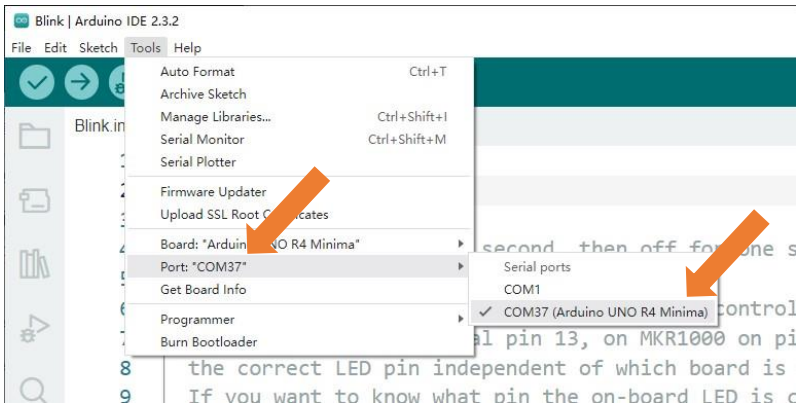
Select the port.

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.

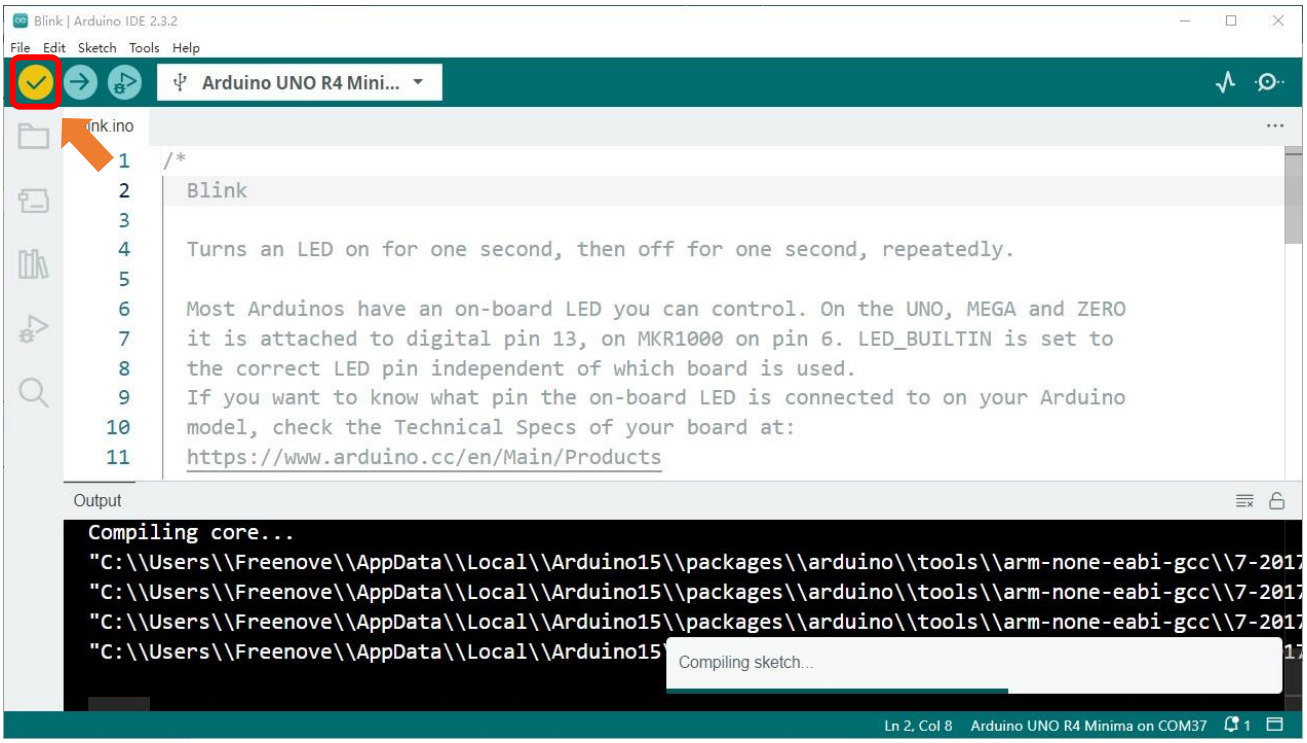
On Linux: It may be /dev/ttyUSB0, /dev/ttyACM0 or something like that.



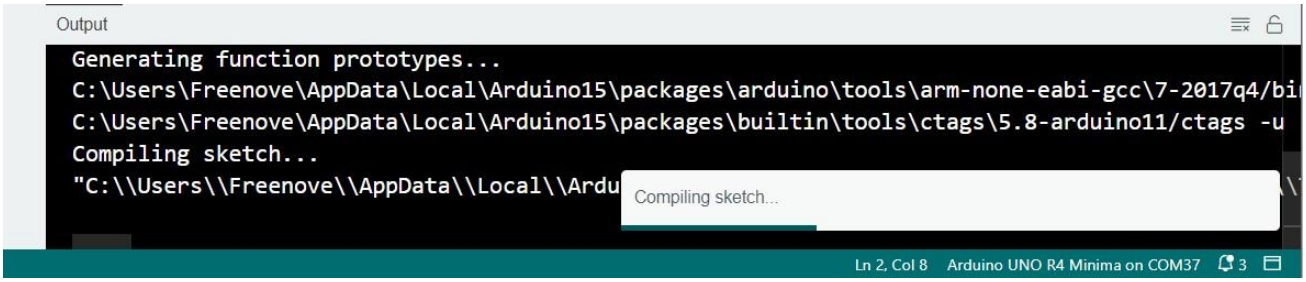
Note: If there is more than one port and you cannot decide which one to choose, disconnect the USB cable and check the port. Then connect the USB cable and check the port again. The new one is the correct port.

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

Click "Verify" button.

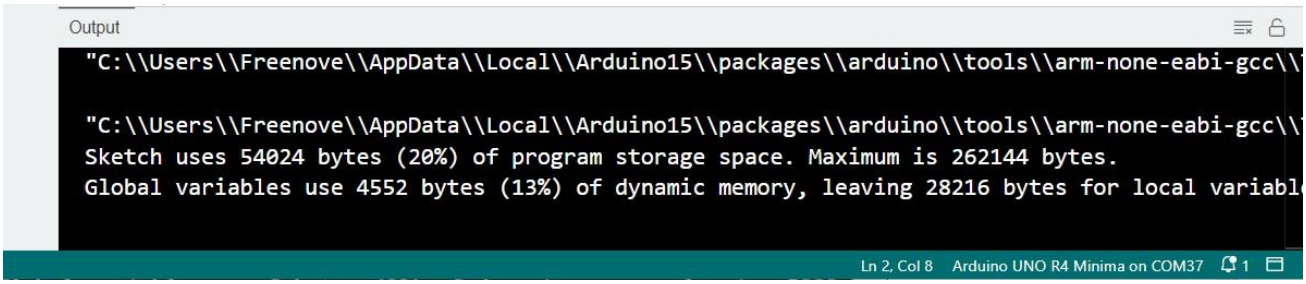


The following figure shows the code being compiled.



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.

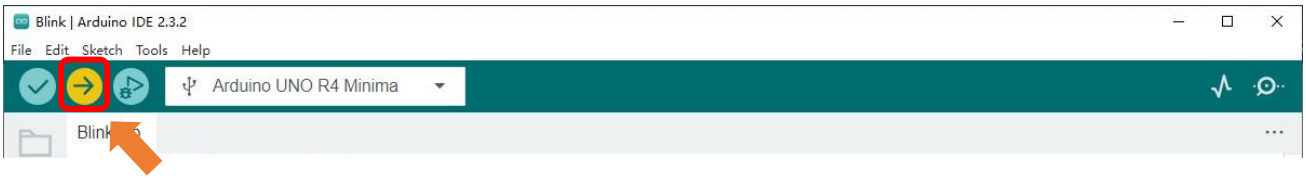
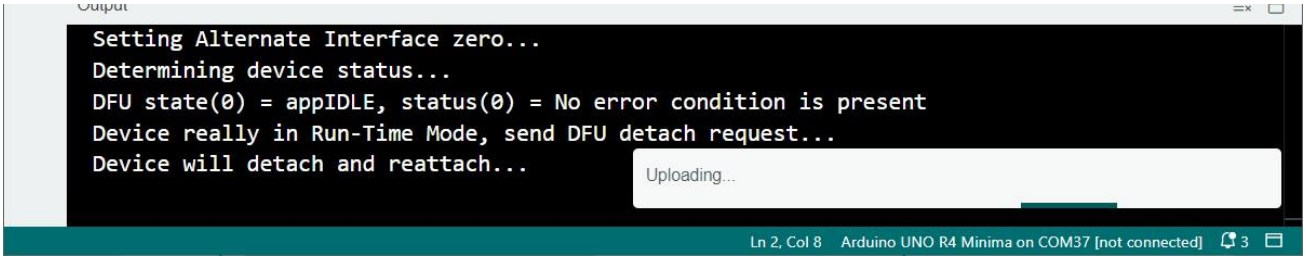
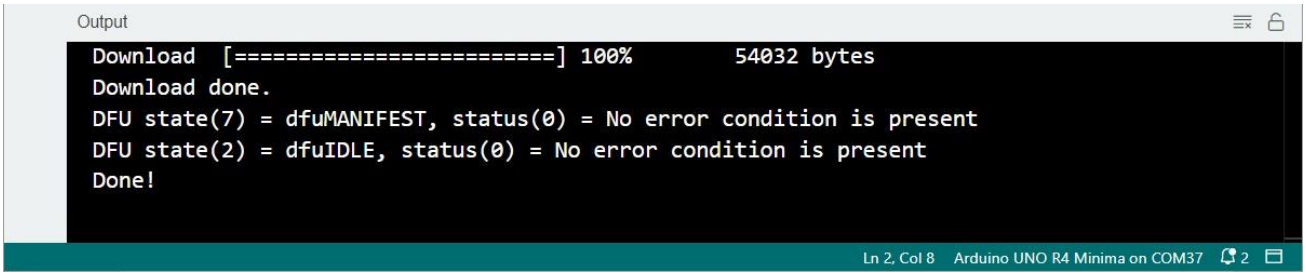


Figure below shows code are uploading.

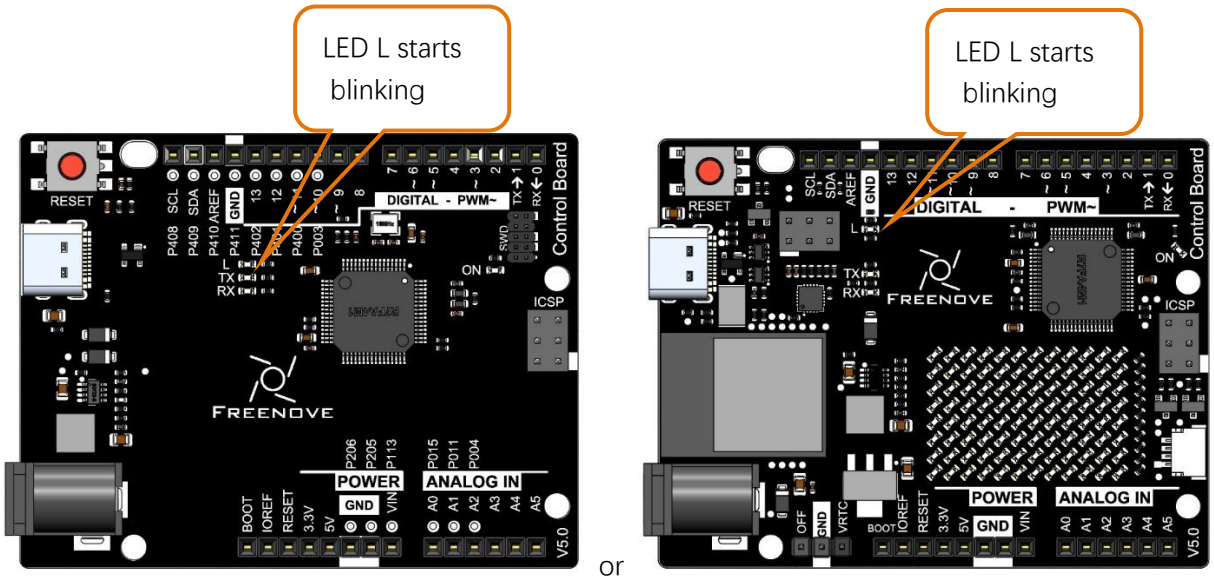


Wait a moment, and then the uploading is completed.



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

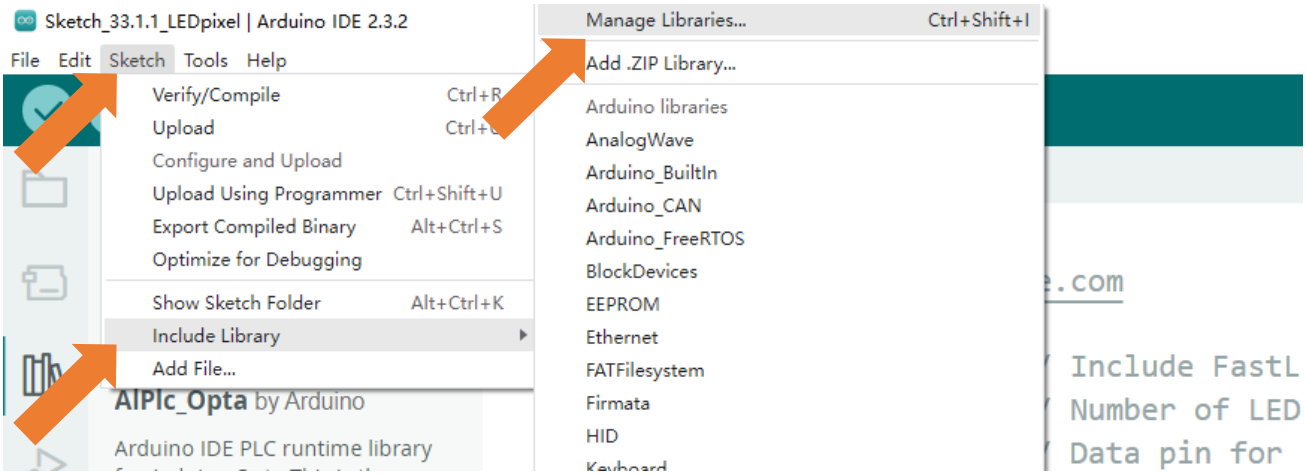
After that, we will see the LED marked with "L" on the control board start blinking. It indicates that the code is running now!



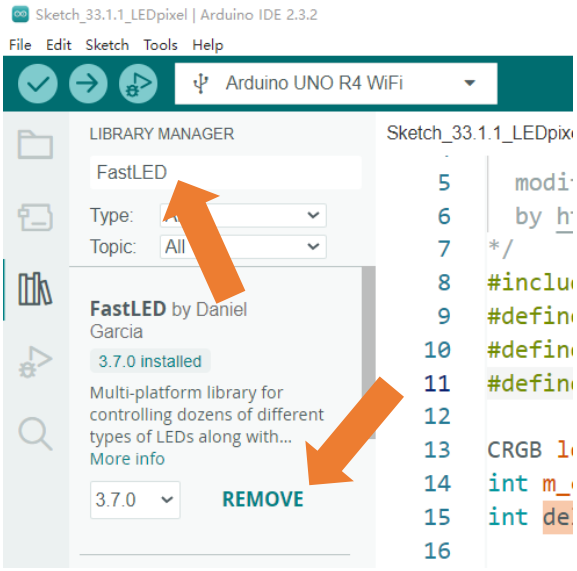
So far, we have completed the first use. I believe you have felt the joy of it. Next, we will carry out a series of projects, from easy to difficult, taking you to learn programming and the building of electronic circuit.

How to install the library

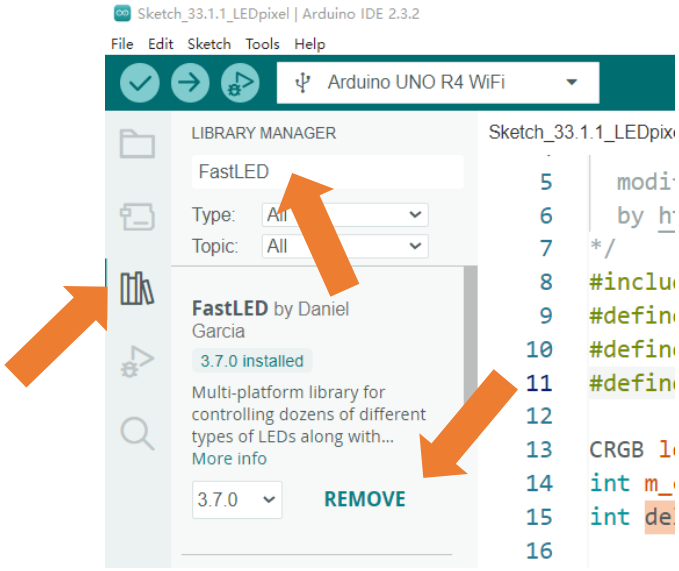
There are two ways to include libraries on Arduino IDE.
The first way, open the Arduino IDE, click Tools →Manager Libraries.



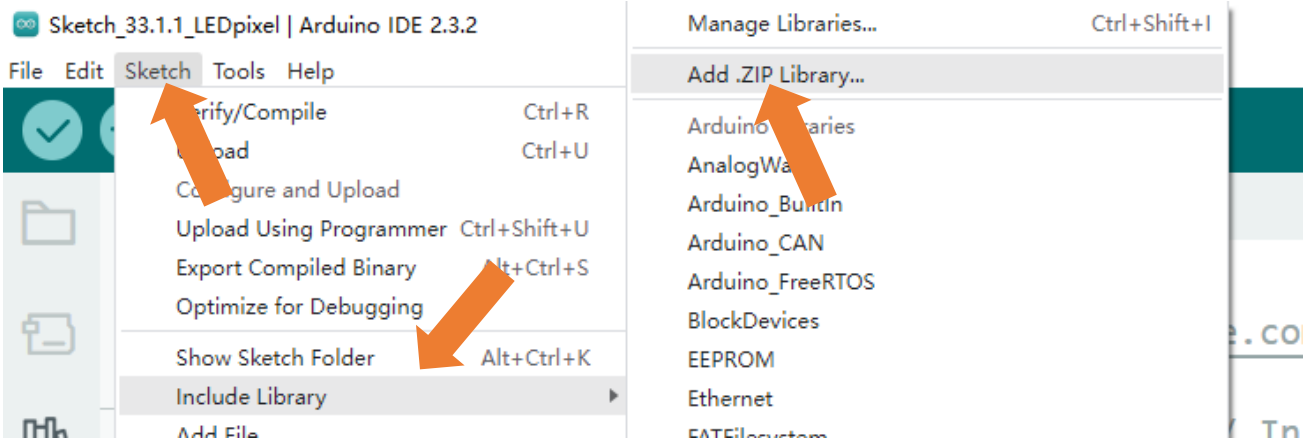
Here we take installing the “FastLED” library as an example. In the pop-up window, Library Manager, search for the name of the Library, “FastLED”. Then click Install.



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.

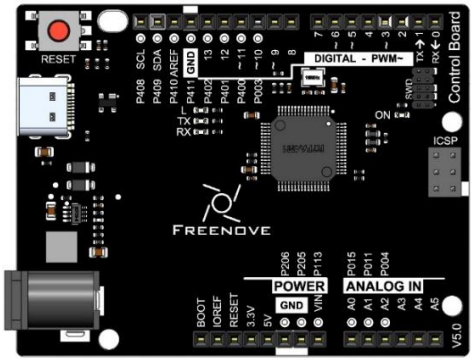
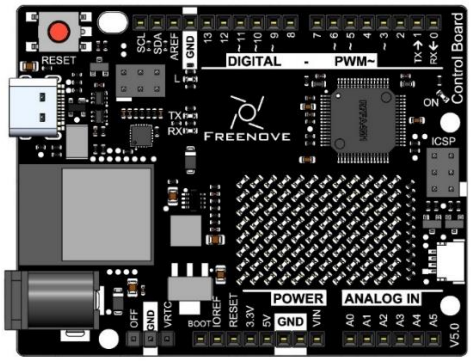
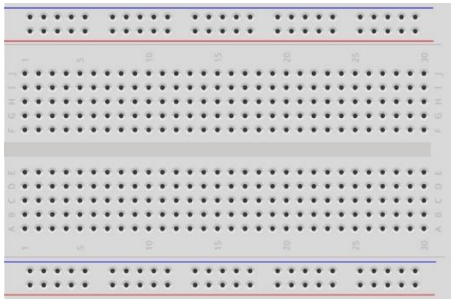
Project 1.1 Control LED with Manual Button

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  or 
Breadboard x1 

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

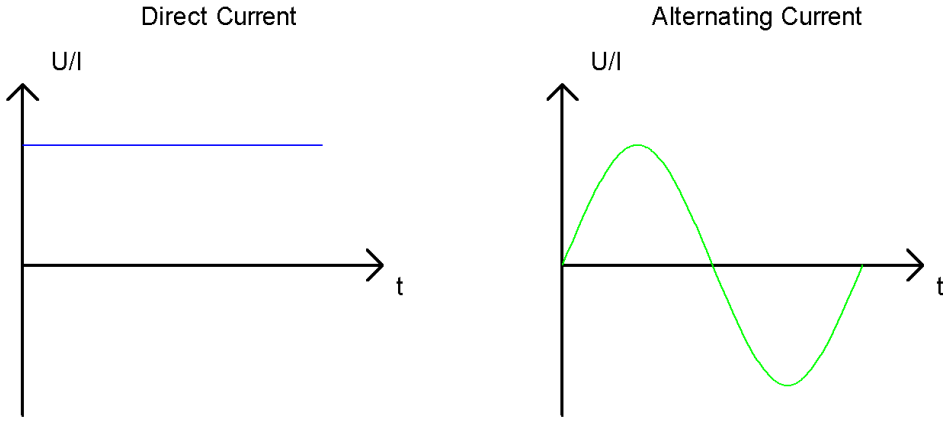
Circuit Knowledge

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.

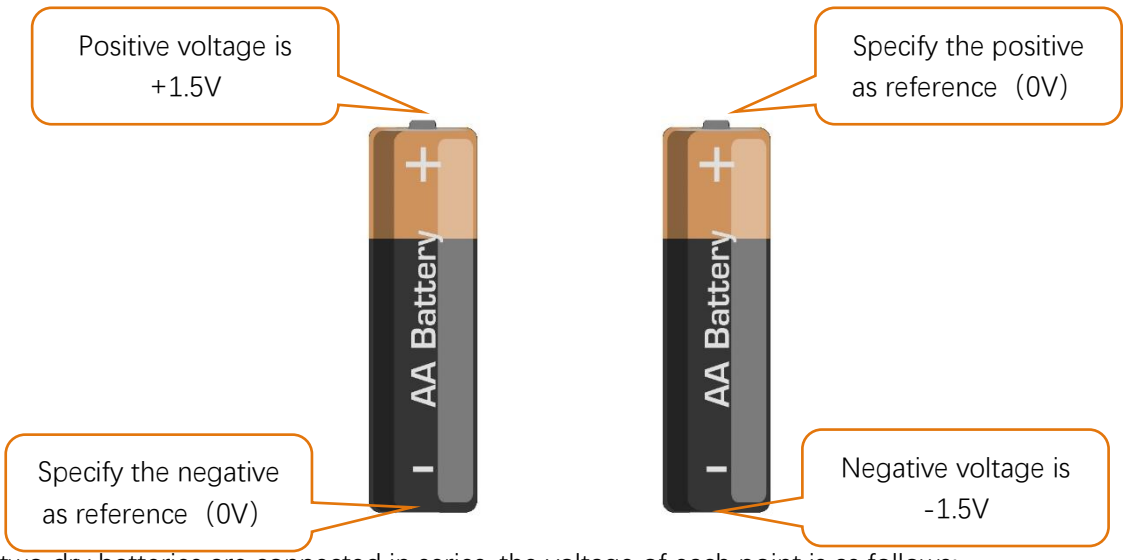
Battery or battery pack can be represented by the following symbols:

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.

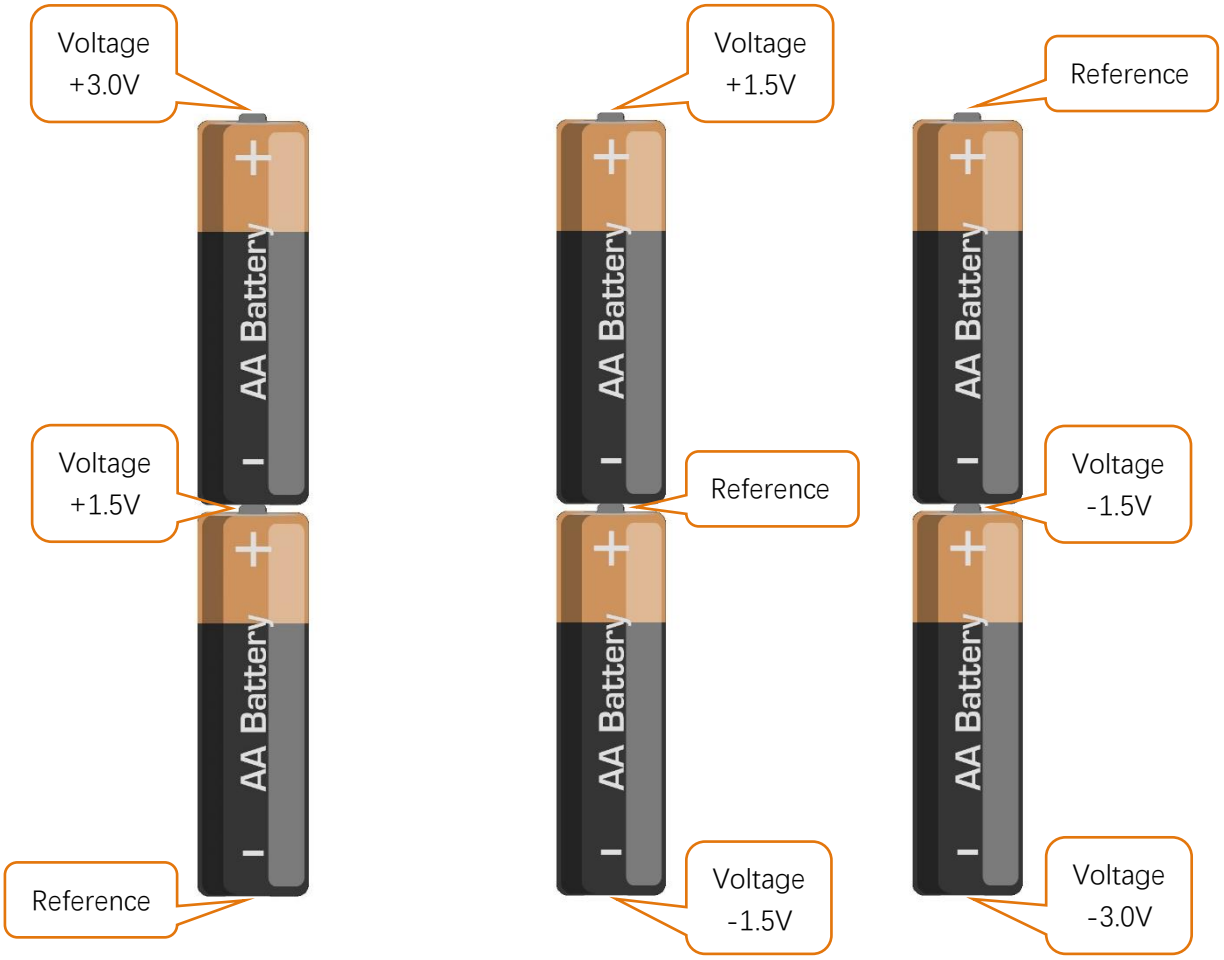
Voltage

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:

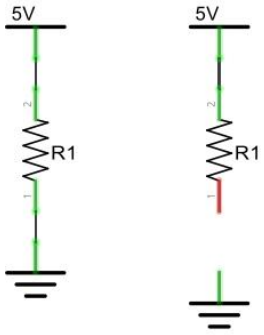
Positive(VCC) Negative(Ground, GND)

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.

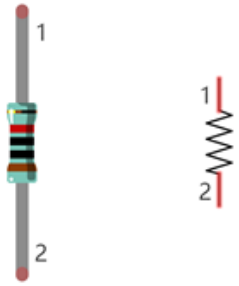
In the figures below: the left one is a loop circuit, so current flows through the circuit. The right one is not a loop circuit, so there is no current.



Resistor

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

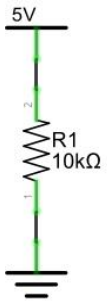
A resistor is a passive electrical component that limits or regulates the flow of current in an electronic circuit. On the left, we see a physical representation of a resistor, and the right is the symbol used to represent the presence of a resistor in a circuit diagram or schematic.



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.



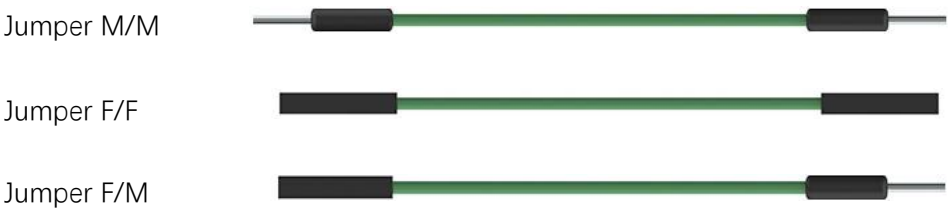
WARNING: Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

Component Knowledge

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

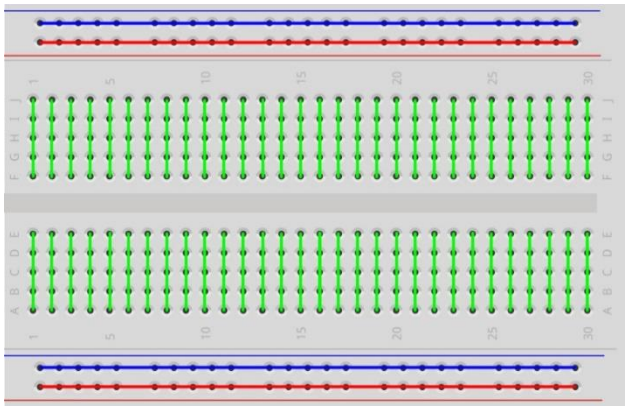
Jumper

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.



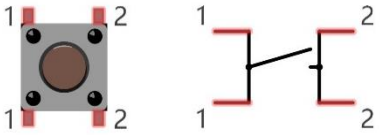
Breadboard

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

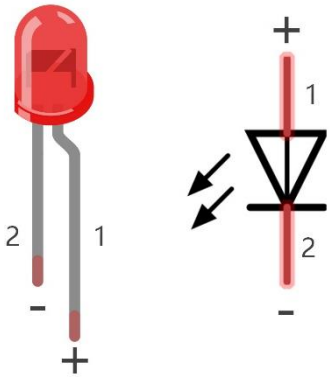
and right sides are the same per the illustration: 

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.



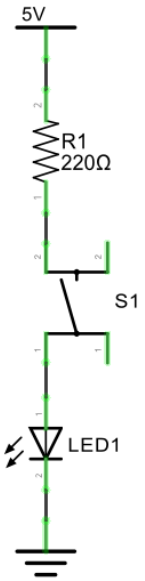
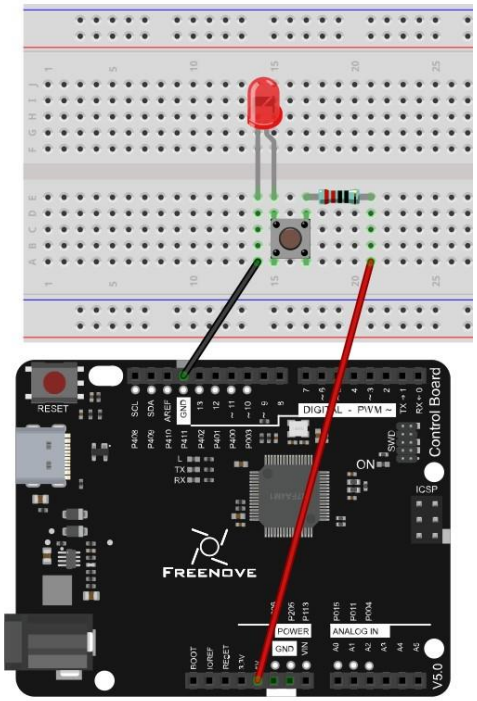
Note: LEDs cannot be directly connected to a power supply, which usually ends in a damaged component. A resistor with a specified resistance value must be connected in series to the LED you plan to use.

Circuit

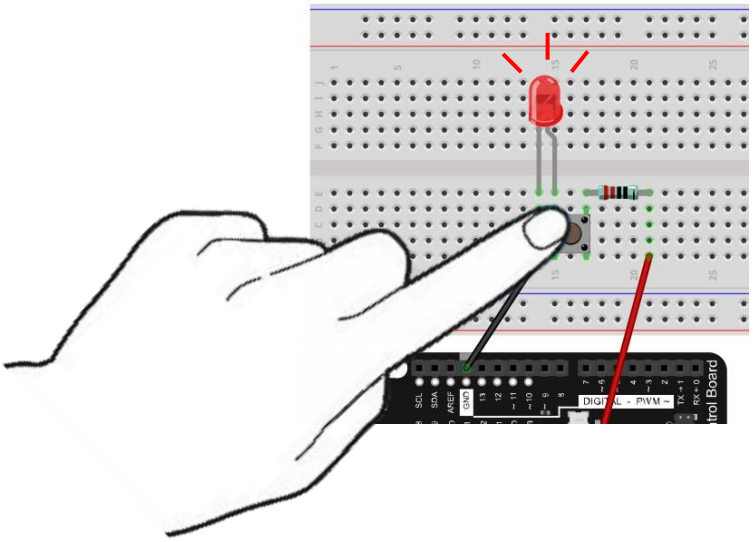
In this project, the LED is controlled by a push button switch, and the control board here only plays the role of power supply in the circuit.

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

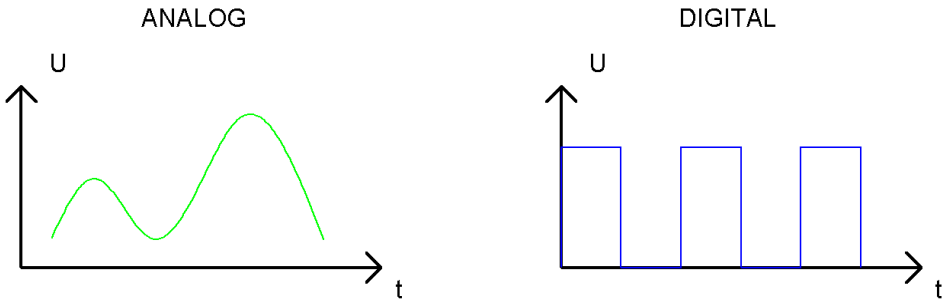
Components are basically the same with those in last section. Push button switch is no more needed.

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.



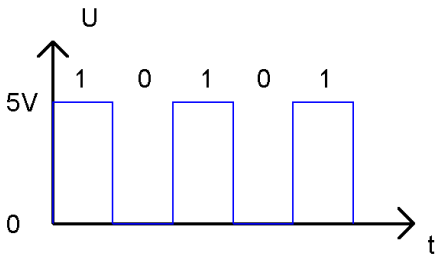
In practical applications, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1) it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

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

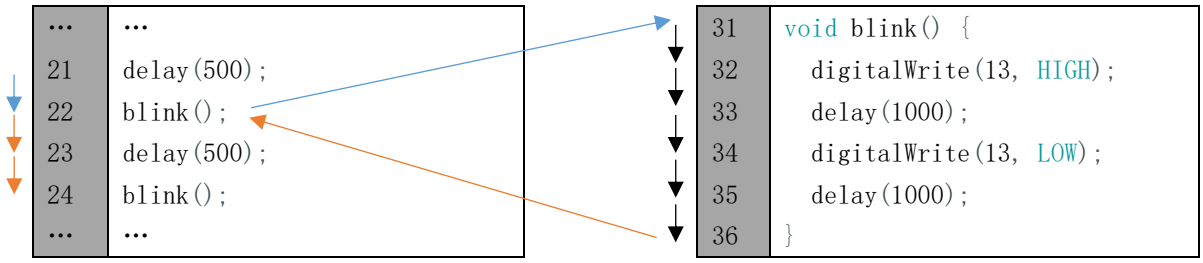
A function is a collection of statements with a sequence of order, which performs a defined task. Let's define a function void blink() as follows:

```
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.

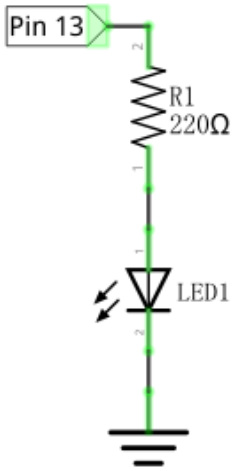
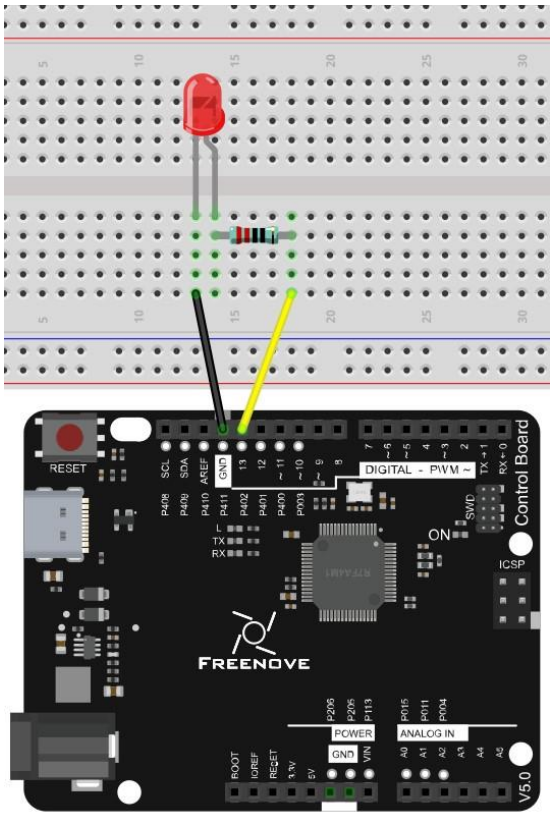


Some functions have one or more parameters. When you call such functions, you need to write parameters inside "()":

```
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 }
```

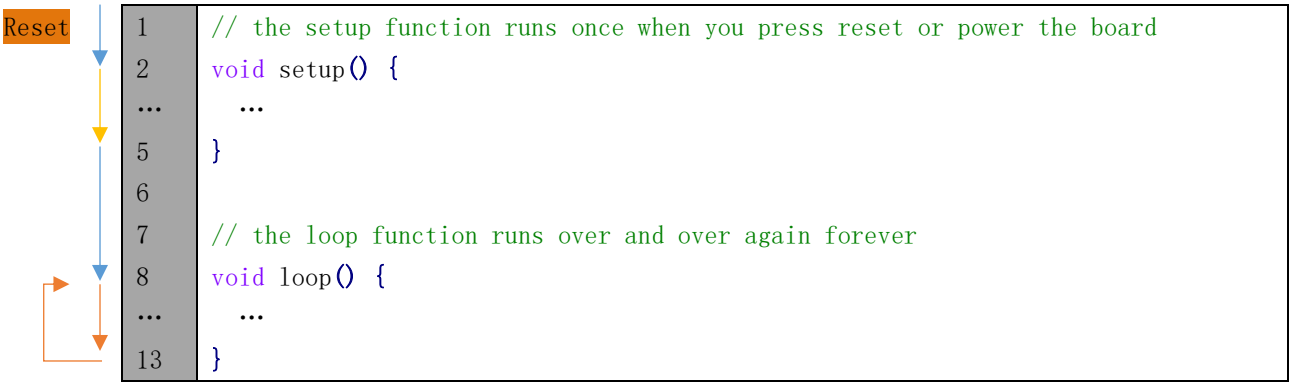
The code usually contains two basic functions: void setup() and void loop().

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
```

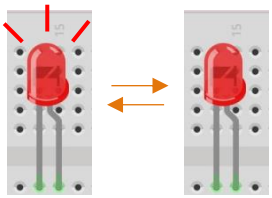
Then set the 13 pint to output low level, and LED light off. One second later, the execution of loop () function will be completed.

```
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.

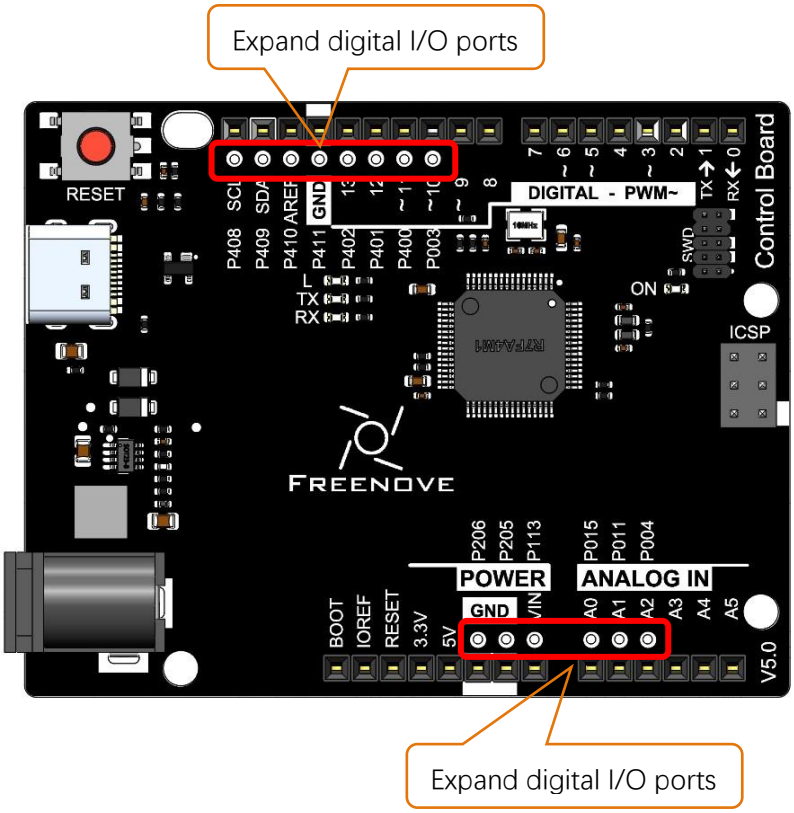
Verify and upload the code, then the LED starts blinking.



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 ".

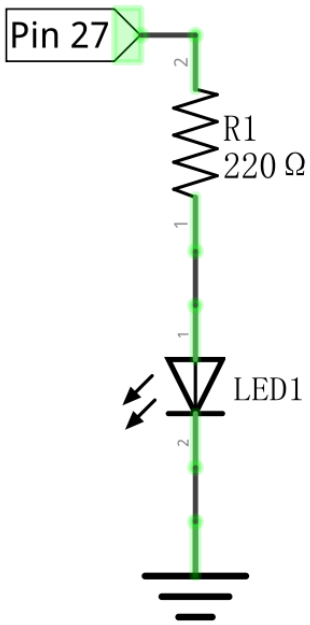
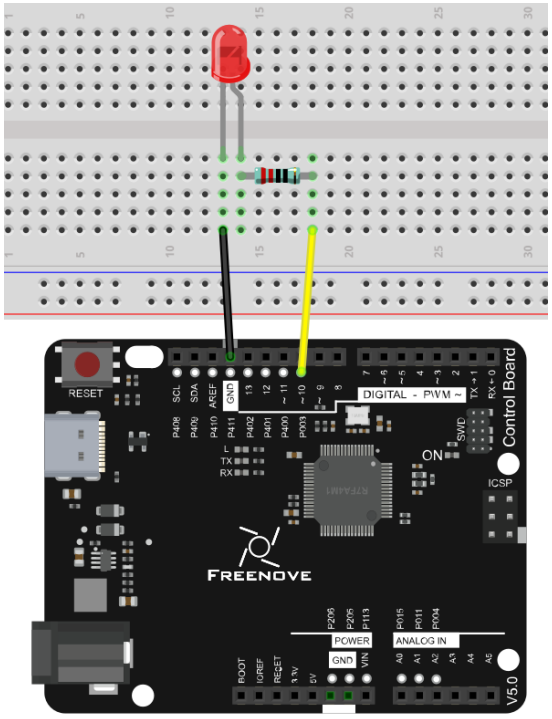
The path may vary due to different versions of board package installed.

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
	

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output
  pinMode(27, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(27, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(27, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

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 Two LEDs Blink

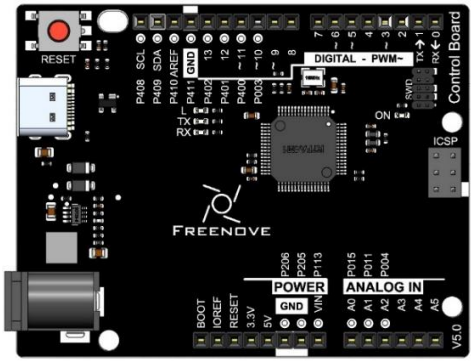
In last chapter, we have already written code to make 1 LED blink on the control board. And now, we will try to make 2 LEDs blink for further programming study.

Project 2.1 Two LEDs Blink

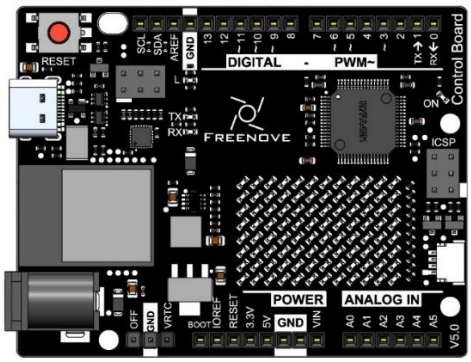
Now, try to make two LEDs blink on control board.

Component List

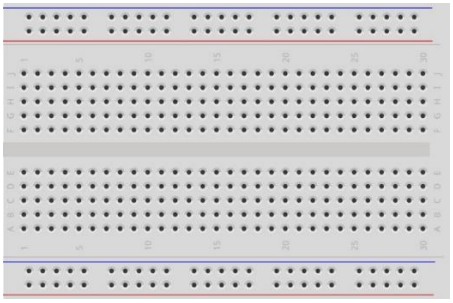
Control board x1



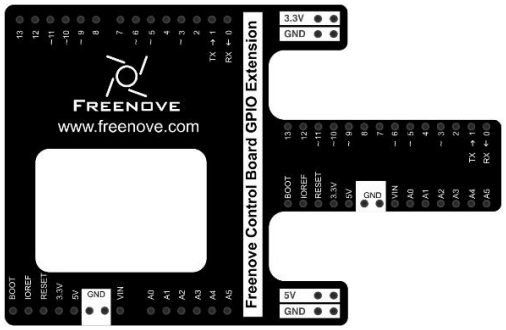
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

LED x2

Resistor 220Ω x2

Jumper M/M x4

Code Knowledge

In the last chapter, we have taken a brief look at programming. Now let us learn more about the basic programming knowledge.

Parameters of function

In the last chapter, we have used a function with a parameter, such as:

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

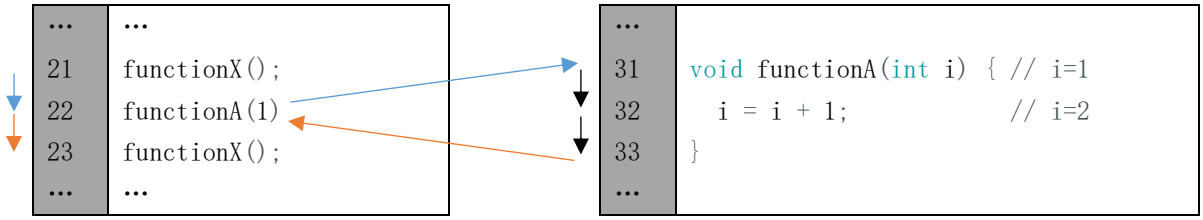
Next, we will define a function with a parameter as below:

```
1  void functionA(int i) {
2    i = i + 1;
3  }
```

"i" is the parameter of this function. "int" is the type of i. When calling this function, it is necessary to enter the parameter of int type:

```
1  functionA(1);
```

The input parameter will be assigned to "i" and involved in the calculation of the functionA(int i):



A function can define more than one parameter and the type of the parameters can be different:

```
1  void functionB(int i, char j) {
2    char k = 'a';
3    i = i + 1;
4    k = j;
5  }
```

Boolean data type

Data of Boolean type can only be assigned to "true" or "false".

"true" generally represents a certain relationship which is tenable and correct, and "false" is the opposite.

```
1  boolean isTrue;
2  isTrue = true;              // after the execution, "isTrue" is assigned to true.
3  isTrue = false;             // after the execution, "isTrue" is assigned to false.
```

In the code, the number 0 can be considered to be false, and nonzero numbers can be considered true.

Logical operator

The logic operators have "&&" (and), "||" (or), "!" (non), and the calculation object of them are boolean type. The result of logic operation is as follows:

&&	true	false
true	true	false
false	false	false

||	true	false
true	true	true
false	true	false

!	
true	false
false	true

For example:

```
1  boolean isTrue;
2  isTrue = true && false;   // after the execution, "isTrue"is assigned to false.
3  isTrue = true || false;   // after the execution, "isTrue"is assigned to true.
4  isTrue = !true;           // after the execution, "isTrue"is assigned to false.
```

Relation operator

Relational operator is used to judge whether the relationship of the two amount is tenable and correct. If the relationship is tenable, the result is true. Otherwise, the result is false.

For example, the results of "1<2" is true and the result of "1>2" is false:

```
1  boolean isTrue;
2  isTrue = 1 < 2;           // after the execution, "isTrue"is true.
3  isTrue = 1 > 2;           // after the execution, "isTrue"is false.
```

There are other relational operators such as "==" (equal to), ">=" (greater than or equal to), "<=" (less than or equal to) and "!=" (not equal to).

Conditional statement

Conditional statements are used to decide whether or not to execute the program based on the result of judgment statement.

```
1  int i = 0;
2  if (i == 0)    // true
3    i = 1;       // execution
4  functionX();
…  …
```

```
1  int i = 0;
2  if (i != 0)    // false
3    i = 1;       // no execution
4  functionX();
…  …
```

When there are many statements needed to be executed, we can put them into "{}":

```
1  int i = 0,  j = 0;
2  if (i == 0) { // true
3    i = 1;       // execution
4    j = 1;       // execution
5  }
6  functionX();
…  …
```

```
1  int i = 0,  j = 0;
2  if (i != 0) { // false
3    i = 1;       // no execution
4    j = 1;       // no execution
5  }
6  functionX();
…  …
```

Only the section of code in which conditions are tenable will be executed:

```
1  int i = 0;
2  if (i == 0)    // true
3    i = 1;       // execution
4  else
5    i = 1;       // no execution
6  functionX();
…  …
```

```
1  int i = 0;
2  if (i != 0)    // false
3    i = 1;       // no execution
4  else
5    i = 1;       // execution
6  functionX();
…  …
```

In addition, it can judge multiple conditions.

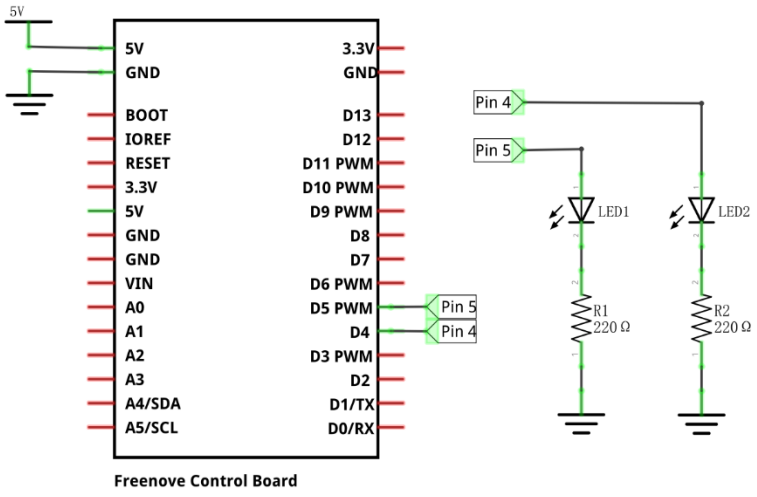
```
1   int i = 1;
2   if (i == 0)       // false
3     i = 1;          // no execution
4   else if (i == 1)// true
5     i = 2;          // execution
6   else if (i == 2)
7     i = 3;          // no execution
8   else
9     i = 4;          // no execution
10  functionX();
…   …
```

```
1   int i = 3;
2   if (i == 0)       // false
3     i = 1;          // no execution
4   else if (i == 1)// false
5     i = 2;          // no execution
6   else if (i == 2)// false
7     i = 3;          // no execution
8   else
9     i = 4;          // execution
10  functionX();
…   …
```

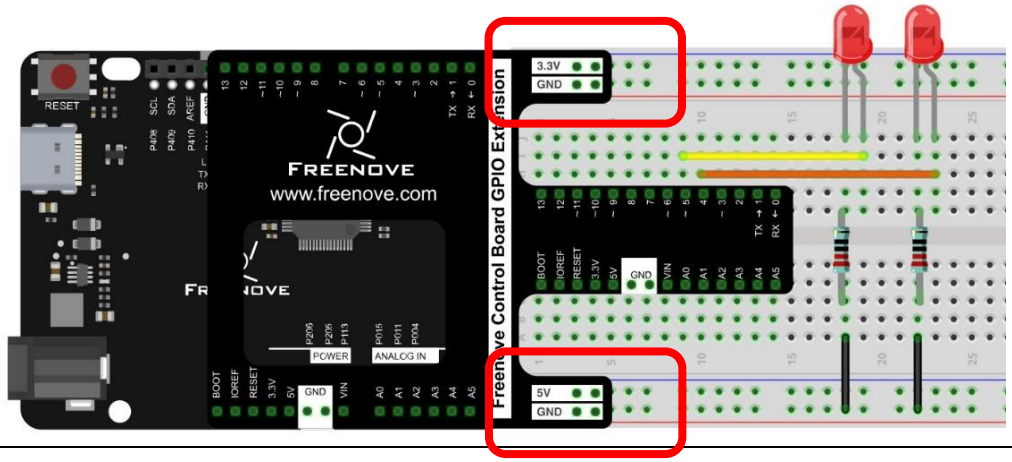
Circuit

Use pin 4 and pin 5 of the control board to drive these two LEDs respectively.

Schematic diagram



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



Sketch

In order to show the difference between using function and not using function, we will write two different sketches to make two LEDs blink.

Sketch 2.1.1

At first, use sketch without function to make two LEDs blink alternatively.

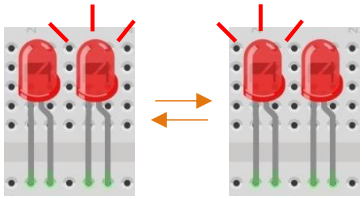
```
1  // set pin numbers:
2  int led1Pin = 5;            // the number of the LED1 pin
3  int led2Pin = 4;            // the number of the LED2 pin
4
5  void setup() {
6    // initialize the LED pin as an output:
7    pinMode(led1Pin, OUTPUT);
8    pinMode(led2Pin, OUTPUT);
9  }
10
11 void loop() {
12   digitalWrite(led1Pin, HIGH);   // turn the LED1 on
13   digitalWrite(led2Pin, LOW);    // turn the LED2 off
14   delay(1000);                   // wait for a second
15
16   digitalWrite(led1Pin, LOW);    // turn the LED1 off
17   digitalWrite(led2Pin, HIGH);   // turn the LED2 on
18   delay(1000);                   // wait for a second
19 }
```

This section of code is similar to the previous section. The difference is that the amount of LEDs is two, and the two LEDs blink alternatively.

Variable scope

In the 2nd and 3rd rows of the code above, we define two variables to store the pin number. These two variables defined outside the function are called "Global variable", which can be called by all other functions. Variables defined inside a function is called "local variable", which can be called only by the current function. When local variables and global variables have same names, the global variable is inaccessible within the function.

Verify and upload the code, then you will see the two LEDs blink alternatively.



Sketch 2.1.2

In the last sketch, we can see that the following two sections of the code are similar, so we will use one function to replace them to simplify the code.

```
12    digitalWrite(led1Pin, HIGH);   // turn the LED1 on
13    digitalWrite(led2Pin, LOW);    // turn the LED2 off
14    delay(1000);                   // wait for a second
```

```
16    digitalWrite(led1Pin, LOW);    // turn the LED1 off
17    digitalWrite(led2Pin, HIGH);   // turn the LED2 on
18    delay(1000);                   // wait for a second
```

Now, we will use a function to improve the above code.

```
1   // set pin numbers:
2   int led1Pin = 5;            // the number of the LED1 pin
3   int led2Pin = 4;            // the number of the LED2 pin
4
5   void setup() {
6     // initialize the LED pin as an output:
7     pinMode(led1Pin, OUTPUT);
8     pinMode(led2Pin, OUTPUT);
9   }
10
11  void loop() {
12    setLed(HIGH, LOW);      // set LED1 on, and LED2 off.
13    setLed(LOW, HIGH);      // set LED1 off, and LED2 on.
14  }
15
16  void setLed(int led1, int led2) {
17    digitalWrite(led1Pin, led1);   // the state of LED1 is determined by variable led1.
18    digitalWrite(led2Pin, led2);   // the state of LED2 is determined by variable led2.
19    delay(1000);                   // wait for a second
20  }
```

In the sketch above, we integrate the 2 LED statements into one function, void setLed(int led1, int led2), and control two LEDs through the parameters led1 and led2.

```
16  void setLed(int led1, int led2) {
17    digitalWrite(led1Pin, led1);   // the state of LED1 is determined by variable led1.
18    digitalWrite(led2Pin, led2);   // the state of LED2 is determined by variable led2.
19    delay(1000);                   // wait for a second
20  }
```

When the function above is called, we will control the two LEDs by using different parameters as below.

```
12    setLed(HIGH, LOW);      // set LED1 on, and LED2 off.
13    setLed(LOW, HIGH);      // set LED1 off, and LED2 on.
```

Verify and upload the code, then you will see the two LEDs blink alternatively.

HIGH and LOW

The macro is an identifier that represents a number, a statement, or a piece of code. HIGH and LOW are two macros. HIGH and LOW are defined in Arduino IDE as below:

```
#define HIGH 1
#define LOW 0
```

In the code, a macro is used as the content defined by itself. For example, setLed (HIGH, LOW) is equivalent to setLed (1, 0).

Using macros can simplify the code and enhance its readability, such as INPUT, OUTPUT.

Sketch 2.1.3

In the previous section of code, we used a function that integrates two similar paragraphs of code. And we control two LEDs to blink alternatively by using two parameters. Now, let us try to use one parameter to control these two LEDs, which is achieved by conditional statements.

Now, we'll use conditional statement to improve the code above.

```
// set pin numbers:
int led1Pin = 5;            // the number of the LED1 pin
int led2Pin = 4;            // the number of the LED2 pin

void setup() {
  // initialize the LED pin as an output:
  pinMode(led1Pin, OUTPUT);
  pinMode(led2Pin, OUTPUT);
}

void loop() {
  setLed1(HIGH);     // set LED1 on, and LED2 off.
  setLed1(LOW);      // set LED1 off, and LED2 on.
}

void setLed1(int led1) {
  digitalWrite(led1Pin, led1);   // the state of LED1 is determined by variable led1.

  if (led1 == HIGH)              // the state of LED2 is determined by variable led1.
    digitalWrite(led2Pin, LOW);  // if LED1 is turned on, LED2 will be turned off.
  else
    digitalWrite(led2Pin, HIGH); // if LED1 is turned off, LED2 will be turned on.

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

Here, we rewrite the function so that we only need to set the state of LED1, and the state of LED2 can be set automatically.

Verify and upload the code, and then two LEDs blink alternatively.

Chapter 3 LED Bar Graph

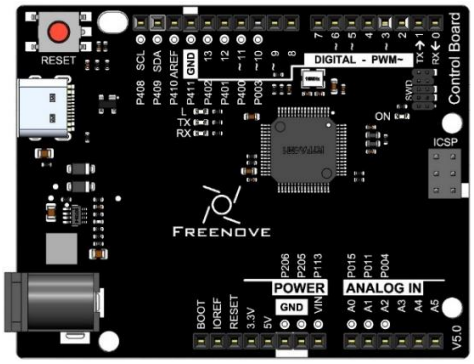
We have learned previously how to control 1 or 2 LEDs through Sketch on the control board and learned some basic knowledge of programming. Now let us try to control 10 LEDs and learn how to simplify the code.

Project 3.1 LED Bar Graph Display

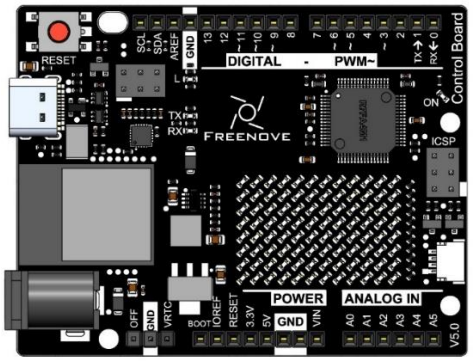
Let us use control board to control a bar graph LED consisting of 10 LEDs.

Component List

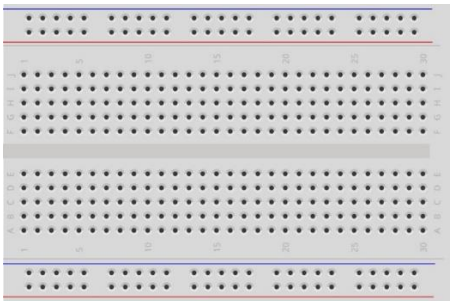
Control board x1



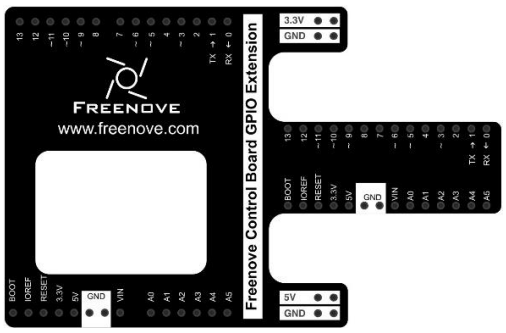
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

LED bar graph x1

Resistor 220Ω x10

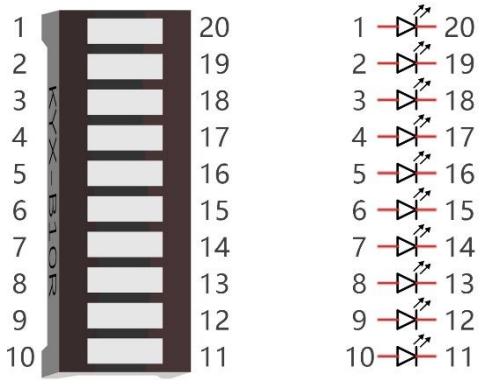
Jumper M/M x10

Component Knowledge

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

LED bar graph

LED bar graph is a component integration consisting of 10 LEDs. At the bottom of the LED bar graph, there are two rows of pins, corresponding to positive and negative pole separately. If the LED bar graph cannot work in the circuit, it is probably because the connection between positive and negative pole is wrong. Please try to reverse the LED bar graph connection.



Code Knowledge

This section will use new code knowledge.

Array

Array is used to record a set of variables. An array is defined as below:

```
1  int a[10];
```

"int" is the type of the array and "10" represents the amount of elements of the array. This array can store 10 int types of elements as below.

```
1  int a[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

Or there is another form that the number of elements is the size of the array:

```
1  int a[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
```

We can reference elements of an array as below:

```
1  int i, j;
2  i = a[0];
3  j = a[1];
4  a[0] = 0;
```

Among them, "[]" is the array index, with a[0] as the first elements in the array.

For example, now we define an array b[] below:

```
int b[] = {5, 6, 7, 8};
```

The value of each element in array b[] is as follows:

b[0]	b[1]	b[2]	b[3]
5	6	7	8

This is just the use of one-dimensional array. And there are two-dimensional arrays, three-dimensional arrays, and multi-dimensional arrays. Readers interested of this part can develop your own learning.

Loop

The loop statement is used to perform repetitive work such as the initialization to all the elements of an array.

```
while(expression)
  functionX();
```

When there is more than one statement to be executed, the form is as follows:

```
while(expression){
  functionX();
  functionY();
}
```

The first step of the execution is judging the expression inside "()". If the result is false, the statements inside "{}" will not be executed; if result is true, the statements will be executed.

```
int i = 0;
while (i < 2)
  i = i + 1;
i = 5;
```

First time: i<2, i=0 is tenable, execute i=i+1, then i=1;

Second time: i<2, i=1 is tenable, execute i=i+1, then i=2;

Third time: i<2, i=2 is not tenable, execution of loop statements is completed. Statement i=5 will be executed next.

"do while" and "while" is similar. The difference is that the loop statements of "do while" is executed before judging expression. The result of the judgment will decide whether or not to go on the next execution:

```
do {
  functionX();
} while (expression);
```

"for" is another loop statement, and its form is as follows:

```
for (expression1; expression2; expression 3)
  functionX();
```

When there is more than one statement to be executed, the form is as follows:

```
for (expression 1; expression 2; expression 3) {
  functionX();
  functionY();
}
```

Expression 1 is generally used to initialize variables; expression 2 is a judgement which is used to decide whether or not to execute loop statements; the expression 3 is generally used to change the value of variables.

For example:

```
int i = 0, j = 0;
for (i = 0; i < 2; i++)
  j++;
i = 5;
```

First time: i=0, i<2 is tenable, execute j++, and execute i++, then i=1, j=1;

Second time: i=1, i<2 is tenable, execute j++, and execute i++, then i=2, j=2;

Third time: i<2 is tenable, i=2 is not tenable. The execution of loop statements is completed. Statement i=5 will be executed next.

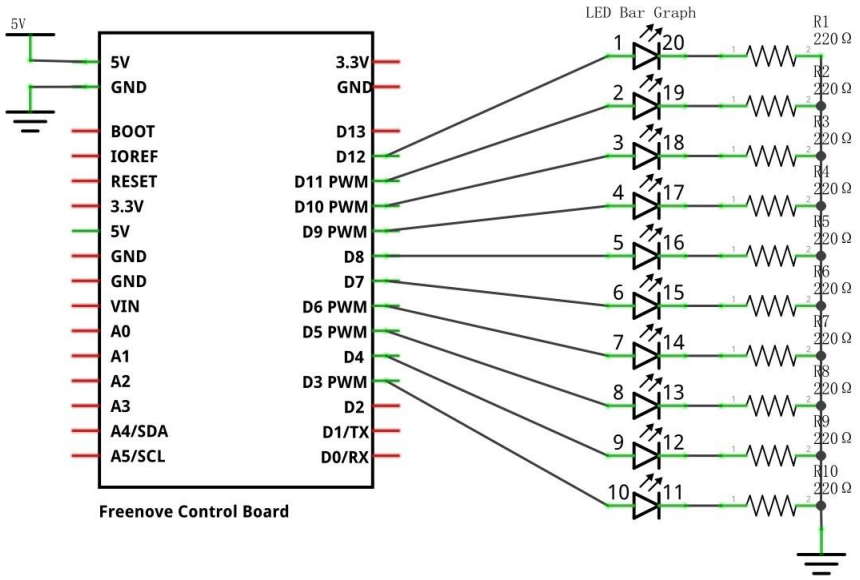
Operator ++, --

"i++" is equivalent to "i=i+1". And "i--" equivalent to "i=i-1".

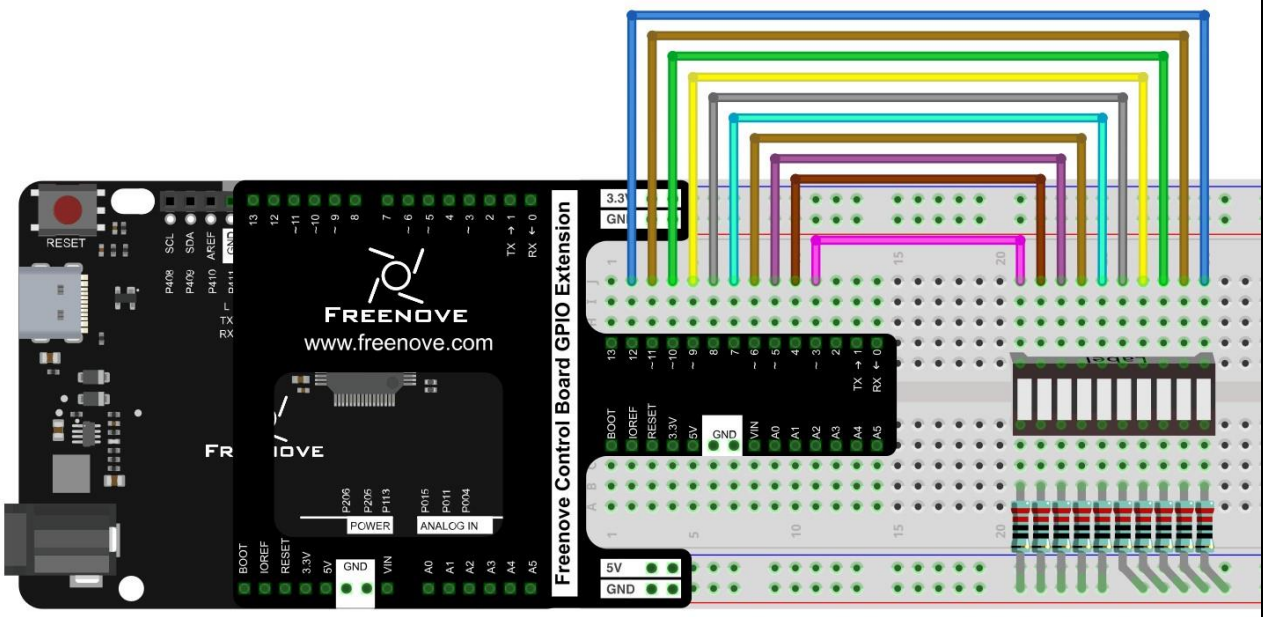
Circuit

Let us use pin 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 of the control board to drive LED bar graph.

Schematic diagram



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



Sketch

Now let us complete the sketch to control LED bar graph.

Sketch 3.1.1

First, write a sketch to achieve the LED light water.

```
const int ledCount = 10;    // the number of LEDs in the bar graph

// an array of pin numbers to which LEDs are attached
int ledPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };

void setup() {
  // loop over the pin array and set them all to output:
  for (int i = 0; i < ledCount; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  // the ith LED of LED bar graph will light up in turn
  for (int i = 0; i < ledCount; i++) {
    barGraphDisplay(i);
  }
}

void barGraphDisplay(int ledOn) {
  // make the "ledOn"th LED of LED bar graph on and the others off
  for (int i = 0; i < ledCount; i++) {
    if (i == ledOn)
      digitalWrite(ledPins[i], HIGH);
    else
      digitalWrite(ledPins[i], LOW);
  }
  delay(100);
}
```

Firstly, let us define a read-only variable to record the number of LEDs as the number of times in the loop.

```
const int ledCount = 10;    // the number of LEDs in the bar graph
```

Read-only variable

"const" keyword is used to define read-only variables, which is called in the same way as other variables. But read-only variables can only be assigned once.

Then we define an array used to store the number of pins connected to LED bar graph. So it is convenient to

manipulate arrays to modify the pin number.

```
// an array of pin numbers to which LEDs are attached
int ledPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

Use loop statement to set the pins to output mode in function setup().

```
void setup() {
  // loop over the pin array and set them all to output:
  for (int i = 0; i < ledCount; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}
```

Define a function to turn ON a certain LED on the LED bar graph and turn OFF the other LEDs.

```
void barGraphDisplay(int ledOn) {
  // make the "ledOn"th LED of LED bar graph on and the others off
  for (int i = 0; i < ledCount; i++) {
    if (i == ledOn)
      digitalWrite(ledPins[i], HIGH);
    else
      digitalWrite(ledPins[i], LOW);
  }
  delay(100);
}
```

Finally, when the above function is called cyclically, there will be a formation of flowing water lamp effect in LED bar graph.

```
void loop() {
  // the ith LED of LED bar graph will light up in turn
  for (int i = 0; i < ledCount; i++) {
    barGraphDisplay(i);
  }
}
```

Verify and upload the code, then you will see the LED bar graph flashing like flowing water.

Sketch 3.1.2

Then modify the code to create a reciprocating LED light water.

```
const int ledCount = 10;    // the number of LEDs in the bar graph

// an array of pin numbers to which LEDs are attached
int ledPins[] = { 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };

void setup() {
  // loop over the pin array and set them all to output:
  for (int i = 0; i < ledCount; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  // makes the "i"th LED of LED bar graph bright in turn
  for (int i = 0; i < ledCount; i++) {
    barGraphDisplay(i);
  }
  // makes the "i"th LED of LED bar graph bright in reverse order
  for (int i = ledCount; i > 0; i--) {
    barGraphDisplay(i - 1);
  }
}

void barGraphDisplay(int ledOn) {
  // make the "ledOn"th LED of LED bar graph on and the others off
  for (int i = 0; i < ledCount; i++) {
    if (i == ledOn)
      digitalWrite(ledPins[i], HIGH);
    else
      digitalWrite(ledPins[i], LOW);
  }
  delay(100);
}
```

We have modified the code inside the function loop() to make the LED bar graph light up in one direction, and then in a reverse direction repeatedly.

Verify and upload the code, then you will see a reciprocating LED water light on LED bar graph.

Chapter 4 LED Blink Smoothly

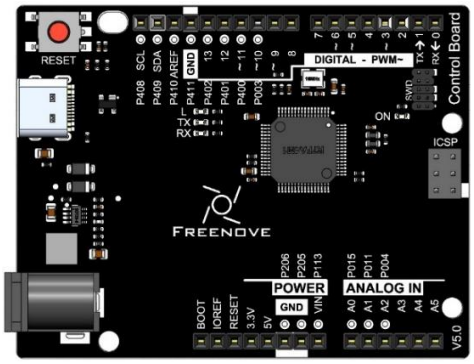
In the previous chapter, we have used Sketch to control up to 10 LEDs on the control board to blink and learned some basic knowledge of programming. Now, let us try to make LED emit different brightness of light.

Project 4.1 LEDs Emit Different Brightness

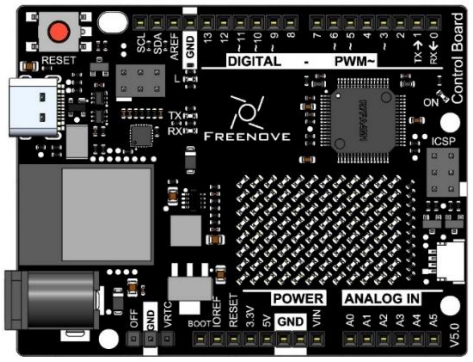
Now, let us use control board to make 4 LED emit different brightness of light.

Component List

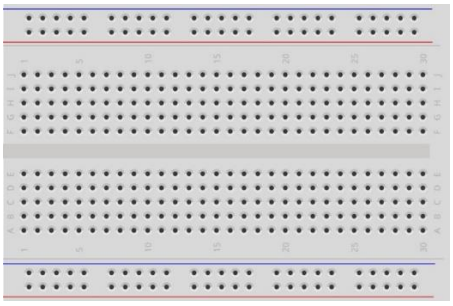
Control board x1



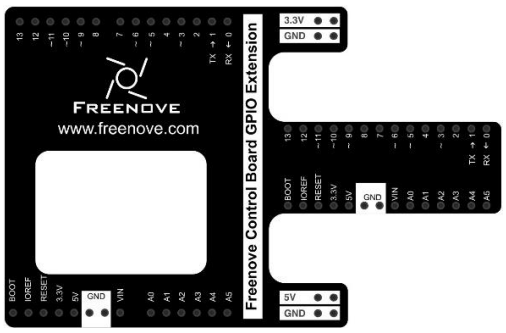
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

LED x4

Resistor 220Ω x4

Jumper M/M x8

Circuit Knowledge

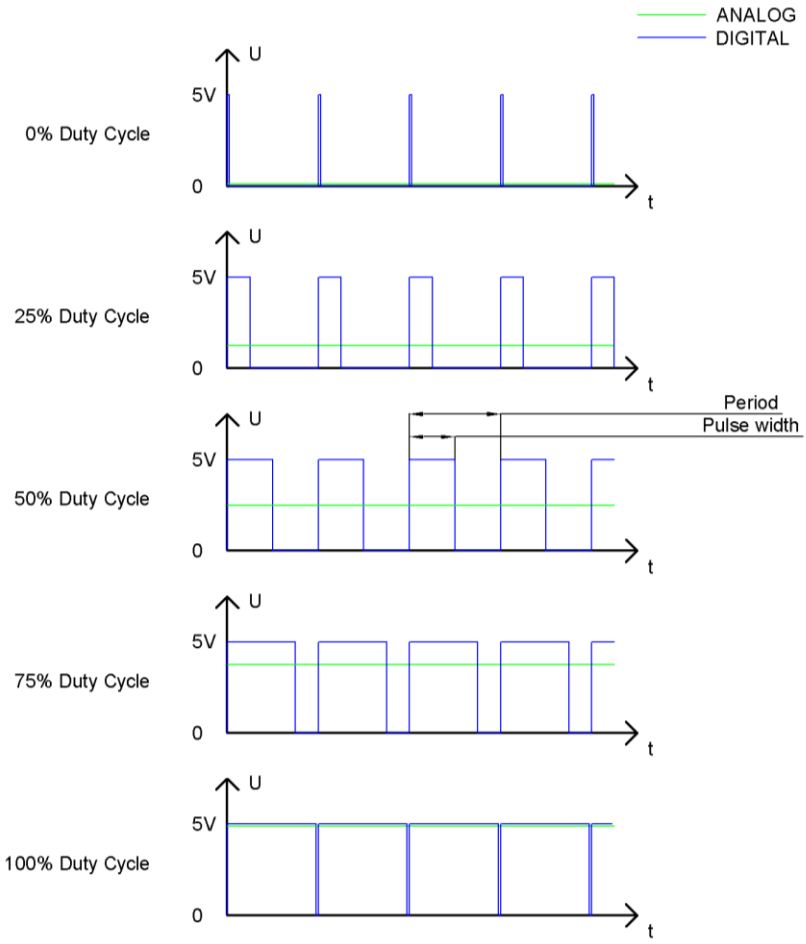
At first, let us learn how to use the circuit to make LED emit different brightness of light,

PWM

PWM, Pulse-Width Modulation, is a very effective method for using digital signals to control analog circuits. Digital processors cannot directly output analog signals. PWM technology makes it very convenient to achieve this conversion (translation of digital to analog signals).

PWM technology uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (Note: the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.

The longer the output of high levels last, the longer the duty cycle and the higher the corresponding voltage in the analog signal will be. The following figures show how the analog signal voltages vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:



The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on.

Code Knowledge

We will use new code knowledge in this section.

Return value of function

We have learned and used the function without return value, now we will learn how to use the function with return value. A function with return value is shown as follow:

```
1  int sum(int i, int j) {
2    int k = i + j;
3    return k;
4  }
```

"int" declares the type of return value of the function sum(int i, int j). If the type of the return value is void, the function does not return a value.

One function can only return one value. It is necessary to use the return statement to return the value of function.

When the return statement is executed, the function will return immediately regardless of code behind the return statement in this function.

A function with return value is called as follows:

```
1  int a = 1, b = 2, c = 0;
2  c = sum(1, 2);              // after the execution the value of c is 3
```

A function with a return value can also be used as a parameter of functions, for example:

```
1  delay(sum(100, 200));
```

It is equivalent to the following code:

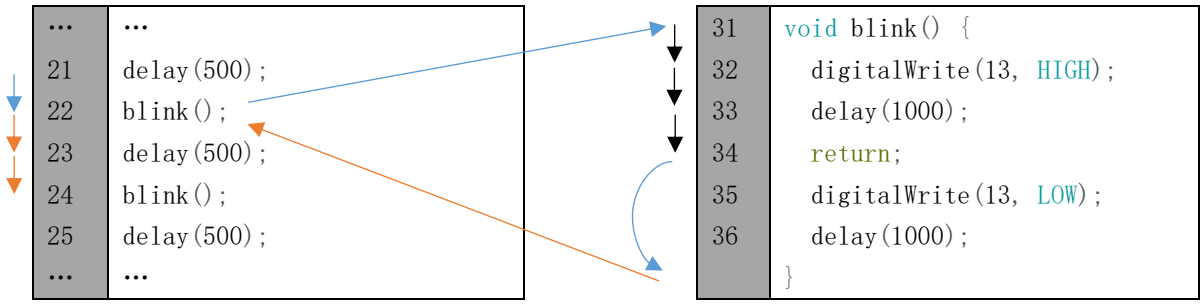
```
1  delay(300);
```

return

We have learned the role of the return statement in a function with a return value. It can also be used in functions without a return value, and there is no data behind the return keyword:

```
1  return;
```

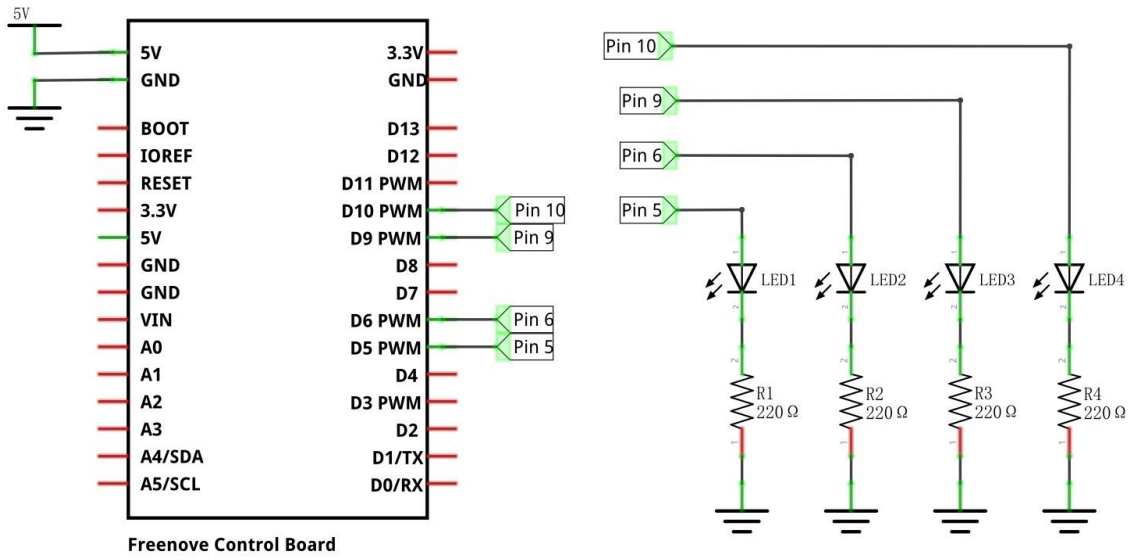
In this case, when the return statement is executed, the function will immediately end its execution rather than return to the end of the function. For example:



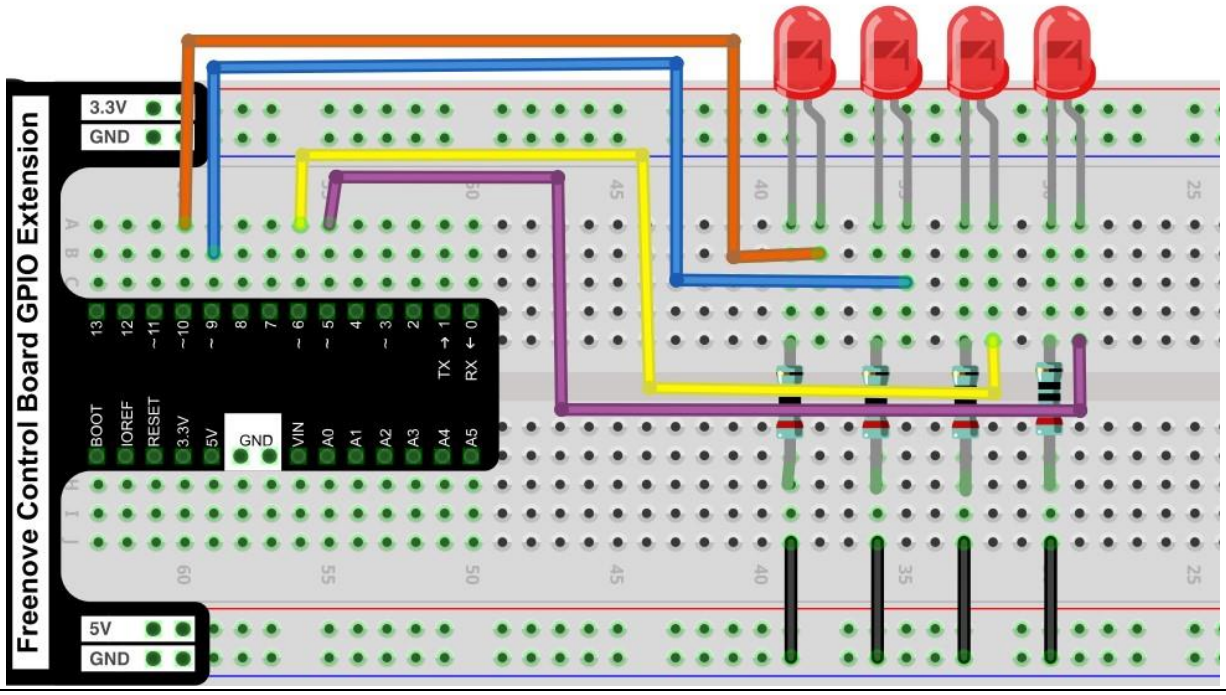
Circuit

Use pin 5, 6, 9, 10 on the control board to drive 4 LEDs.

Schematic diagram



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



Sketch

Sketch 4.1.1

Now let us use sketch to make 4 LEDs emit different brightness of light. We will transmit signal to make the 4

ports connected to LEDs output the PWM waves with duty cycle of 2%, 10%, 50%, and 100% to let the LEDs emit different brightness of the light.

```
// set pin numbers:
int ledPin1 = 5,            // the number of the LED1 pin
    ledPin2 = 6,            // the number of the LED2 pin
    ledPin3 = 9,            // the number of the LED3 pin
    ledPin4 = 10;           // the number of the LED4 pin

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(ledPin3, OUTPUT);
  pinMode(ledPin4, OUTPUT);
}

void loop()
{
  // set the ports output PWM waves with differrent duty cycle
  analogWrite(ledPin1, map(2, 0, 100, 0, 255));
  analogWrite(ledPin2, map(10, 0, 100, 0, 255));
  analogWrite(ledPin3, map(50, 0, 100, 0, 255));
  analogWrite(ledPin4, map(100, 0, 100, 0, 255));
}
```

After the initialization of the 4 ports, we set the ports to output PWM waves with different duty cycle. Take ledPin1 as an example, firstly map 2% to the range of 0-255, and then output the PWM wave with duty cycle of 2%,

```
  analogWrite(ledPin1, map(2, 0, 100, 0, 255));
```

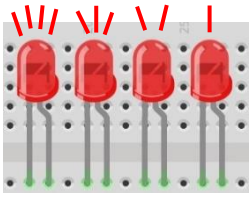
analogWrite(pin, value)

Arduino IDE provides the function, analogWrite(pin, value), which can make ports directly output PWM waves. Only the digital pin marked with "~" symbol on the control board can use this function to output PWM waves. In the function called analogWrite(pin, value), the parameter "pin" specifies the port used to output PWM wave. The range of value is 0-255, which represents the duty cycle of 0%-100%.
In order to use this function, we need to set the port to output mode.

map(value, fromLow, fromHigh, toLow, toHigh)

This function is used to remap a value, which will return a new value whose percentage in the range of toLow-toHigh is equal to the percentage of "value" in the range of fromLow-fromHigh. For example, 1 is the maximum in the range of 0-1 and the maximum value in the scope of 0-2 is 2, that is, the result value of map (1, 0, 1, 0, 2) is 2.

Verify and upload the code, and you will see the 4 LEDs emit light with different brightness.



Project 4.2 LED Blinking Smoothly

We will learn how to make a LED blink smoothly, that is, breathing light.

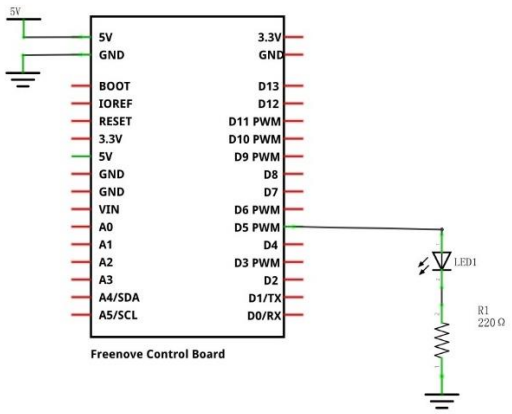
Component List

The Component list is basically the same as those in last section. And we need to get rid of a few LEDs and resistors.

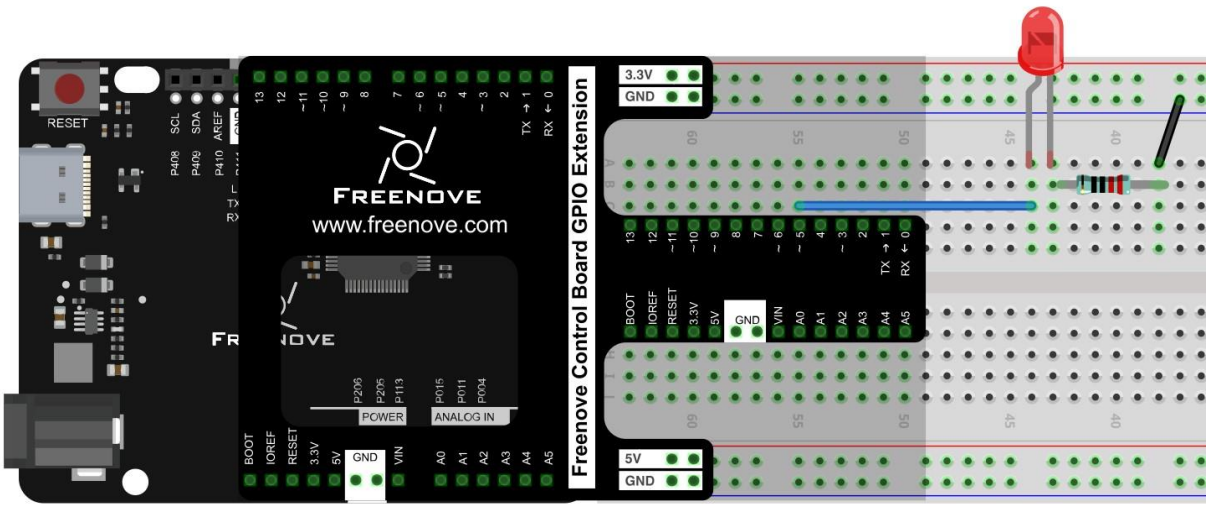
Circuit

Remove some LEDs and resistors connected to pin 6, 9, 10 on the control board in the circuit of the previous section.

Schematic diagram



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



Sketch

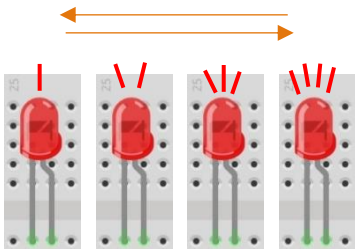
Sketch 4.2.1

Now complete the sketch to make brightness of LED change from dark to bright, and then from bright to dark. That is to make the duty cycle of the PWM wave change from 0%-100%, and then from 100%-0% cyclically.

```
// set pin numbers:
int ledPin = 5;             // the number of the LED pin

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // call breath() cyclically
  breath(ledPin, 6);
  delay(500);
}

void breath(int ledPin, int delayMs) {
  for (int i = 0; i <= 255; i++) {  // "i" change from 0 to 255
    analogWrite(ledPin, i);         // corespongding duty cycle change from 0%-100%
    delay(delayMs);                 // adjust the rate of change of brightness
  }
  for (int i = 255; i >= 0; i--) {  // "i" change from 255 to 0
    analogWrite(ledPin, i);         // corespongding duty cycle change from 0%-100%
    delay(delayMs);                 // adjust the rate of change in brightness
  }
}
```

Through two “for” loops, the duty cycle of the PWM wave changes from 0% to 100%, and then from 100% to 0% cyclically. delay(ms) function is used to control the change rate in the "for" loop, and you can try to modify the parameters to modify the change rate of brightness.

Verify and upload the code, then you will see that the brightness of the LED changes from dark to light, and from the light to dark cyclically.



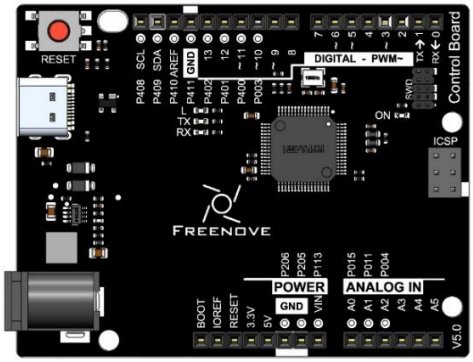
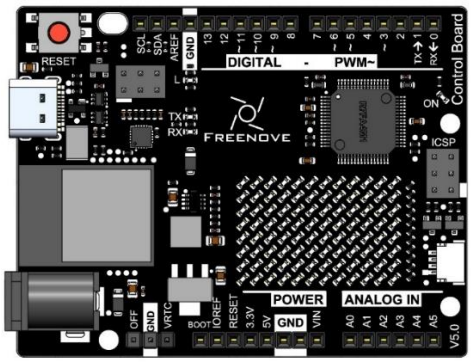
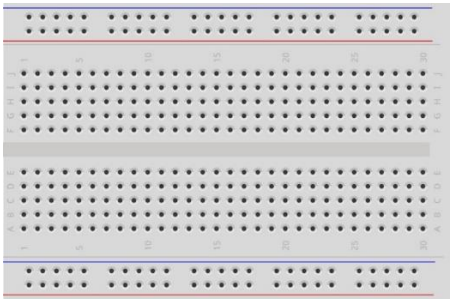
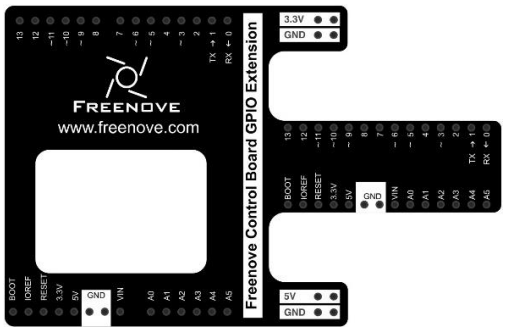
Chapter 5 Control LED with Push Button Switch

In the previous chapter, we have used the control board to output signals to make 10 LEDs flash, and make one LED emit different brightness. Now, let's learn how to get an input signal.

Project 5.1 Control LED with Push Button Switch

We will use the control board to get the status of the push button switch, and show it through LED.

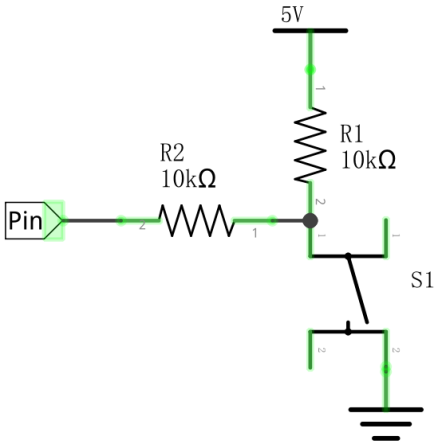
Component List

<table>
<tr><td colspan="4">Control board x1
  or  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td><td colspan="2">GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">USB cable x1
  </td><td>LED x1
  </td><td>Resistor
220Ω x1
  </td><td>Resistor
10kΩ x2
  </td><td>Push button
Switch x1
  </td></tr>
<tr><td colspan="2">Jumper M/M x4
</td><td colspan="4"></td></tr>
</table>

Circuit Knowledge

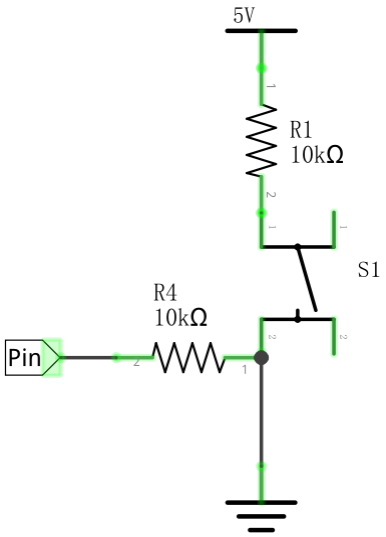
Connection of Push Button Switch

In Chapter 1, we connect push button switch directly to power up the circuit to control the LED to turn on or off. In digital circuits, we need to use the push button switch as an input signal. The recommended connection is as follows:



In the above circuit diagram, when the button is not pressed, 5V (high level) will be detected by control board port; and 0V (low level) when the button is pressed. The role of Resistor R2 here is to prevent the port from being set to output high level by accident. Without R2, the port could be connected directly to the cathode and cause a short circuit when the button is pressed.

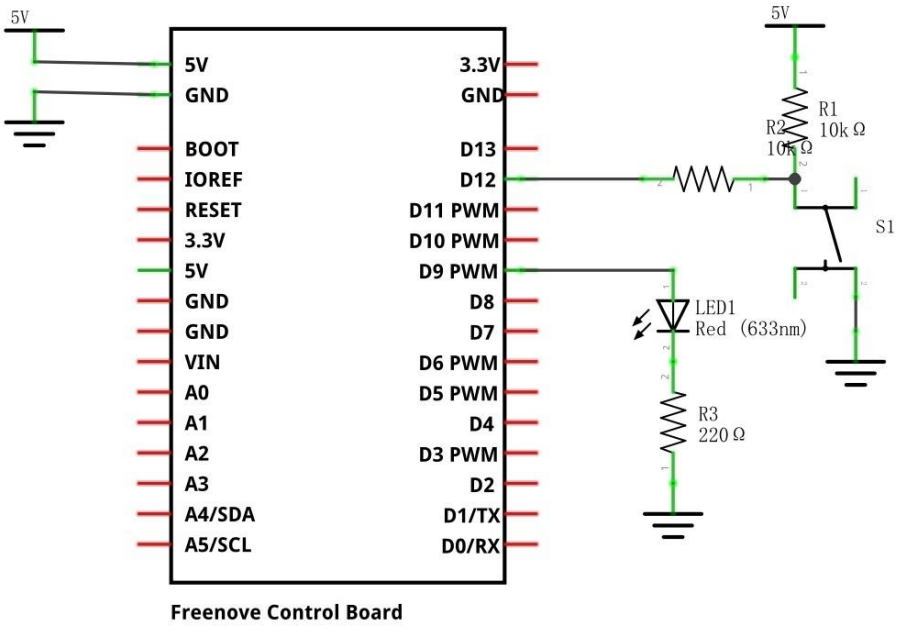
The following diagram shows another connection, in which the level detected by the control board port is opposite to the above diagram, whenever the button is pressed or not.



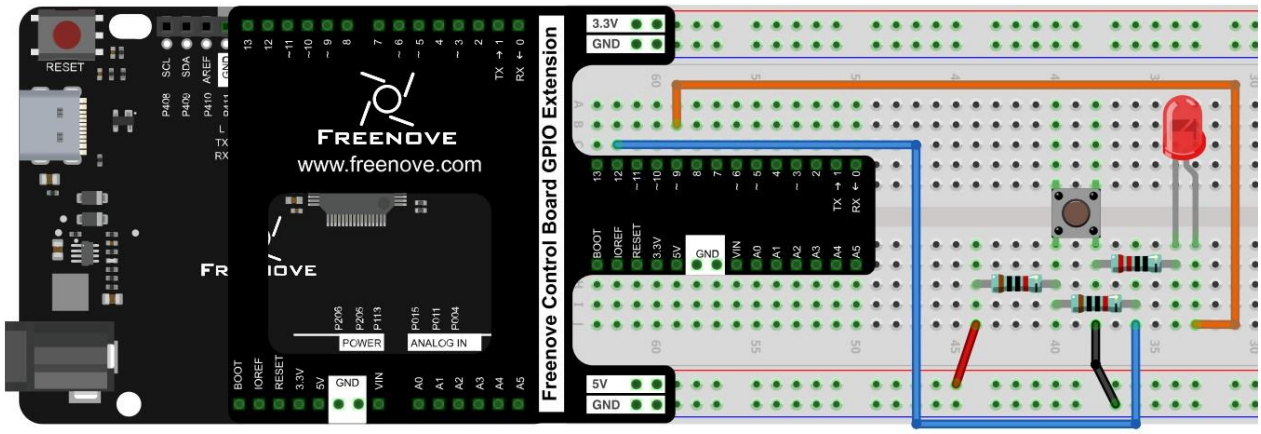
Circuit

Use pin 12 of control board to detect the status of push button, and pin 9 to drive LED.

Schematic diagram



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



Sketch

Sketch 5.1.1

Now, write code to detect the state of push button, and show it through LED.

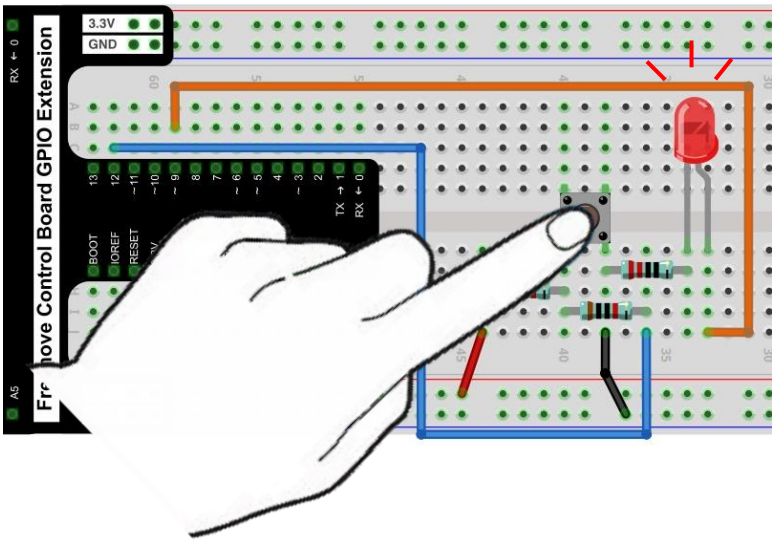
```
int buttonPin = 12; // the number of the push button pin
int ledPin = 9;     // the number of the LED pin

void setup() {
  pinMode(buttonPin, INPUT);  // set push button pin into input mode
  pinMode(ledPin, OUTPUT);    // set LED pin into output mode
}

void loop() {
  if (digitalRead(buttonPin) == HIGH) // if the button is not pressed
    digitalWrite(ledPin, LOW);        // switch off LED
  else                                // if the button is pressed
    digitalWrite(ledPin, HIGH);       // switch on LED
}
```

After the port is initialized, the LED will be turned on or off in accordance with the state of the pin connected to push button switch.

digitalRead(pin)

Arduino IDE provides a function digitalRead(pin) to obtain the state of the port pin. The return value is HIGH or LOW, that is, high level or low level.

Verify and upload the code, press the button, LED lights up; release the button, LED lights off.



Project 5.2 Change LED State with Push Button Switch

In the previous section, we have finished the experiment that LED lights ON when push button switch is pressed, and lights OFF as soon as it's released. Now, let's try something new: each time you press the button down, the state of LED will be changed.

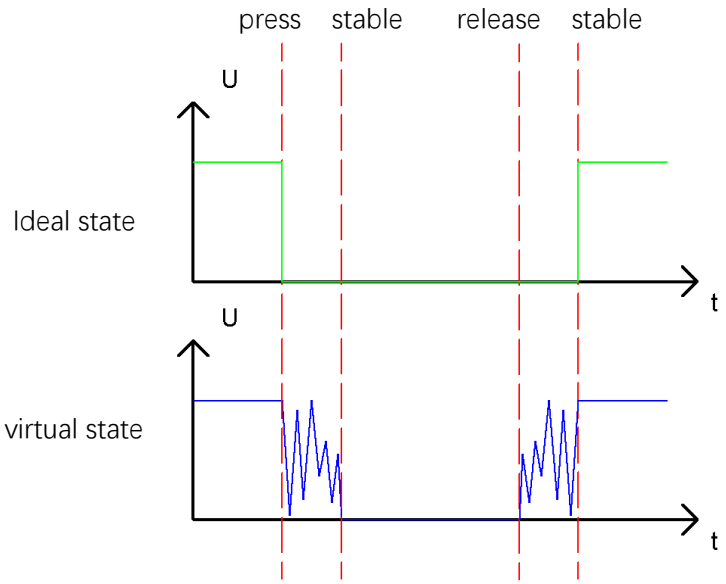
Component List

Same with the previous section.

Circuit Knowledge

Debounce a push button switch

When a Momentary Push Button Switch is pressed, it will not change from one state to another state immediately. Due to tiny mechanical vibrations, there will be a short period of continuous buffeting before it stabilizes in a new state too fast for Humans to detect but not for computer microcontrollers. The same is true when the push button switch is released. This unwanted phenomenon is known as “bounce”.



Therefore, if we can directly detect the state of the Push Button Switch, there are multiple pressing and releasing actions in one pressing cycle. This buffeting will mislead the high-speed operation of the microcontroller to cause many false decisions. Therefore, we need to eliminate the impact of buffeting. Our solution: to judge the state of the button multiple times. Only when the button state is stable (consistent) over a period of time, can it indicate that the button is actually in the ON state (being pressed).

Circuit

Same with the previous section.

Sketch

Sketch 5.2.1

Now, write a code to detect the state of the push button switch. Every time you pressed it, the state of LED will be changed.

```
int buttonPin = 12; // the number of the push button pin
int ledPin = 9;     // the number of the LED pin
boolean isLighting = false; // define a variable to save the state of LED

void setup() {
  pinMode(buttonPin, INPUT);  // set push button pin into input mode
  pinMode(ledPin, OUTPUT);    // set LED pin into output mode
}

void loop() {
  if (digitalRead(buttonPin) == LOW) {        // if the button is pressed
    delay(10);    // delay for a certain time to skip the bounce
    if (digitalRead(buttonPin) == LOW) {      // confirm again if the button is pressed
      reverseLED();                           // reverse LED
      while (digitalRead(buttonPin) == LOW);  // wait for releasing
      delay(10);  // delay for a certain time to skip bounce when the button is released
    }
  }
}

void reverseLED() {
  if (isLighting) {             // if LED is lighting,
    digitalWrite(ledPin, LOW);  // switch off LED
    isLighting = false;         // store the state of LED
  }
  else {                        // if LED is off,
    digitalWrite(ledPin, HIGH); // switch LED
    isLighting = true;          // store the state of LED
  }
}
```

Verify and upload the code, then each time you press the button, LED changes its state accordingly.

When judging the push button switch state, if it is detected as "pressed down", wait for a certain time to detect again to eliminate the effect of bounce. When the state is stable, released push button switch, and wait for a certain time to eliminate the effect of bounce after it is released.

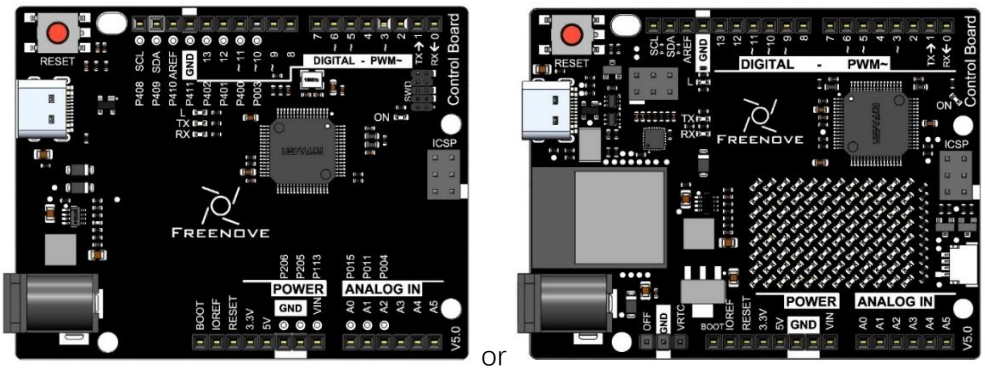
Chapter 6 Serial

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

Project 6.1 Send Data through Serial

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

Component List

Control board x1  or
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:

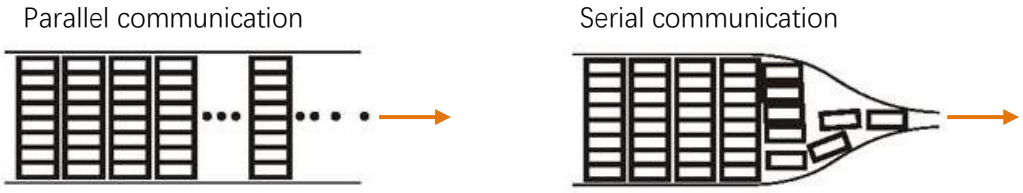
		Remainder	Sequence
2	150	0	0
2	75	1	1
2	37	1	2
2	18	0	3
2	9	1	4
2	4	0	5
2	2	0	6
2	1	1	7
	0		

The result is 10010110.

Circuit Knowledge

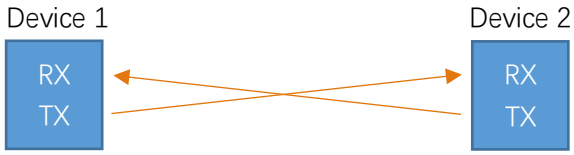
Serial and parallel communication

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:

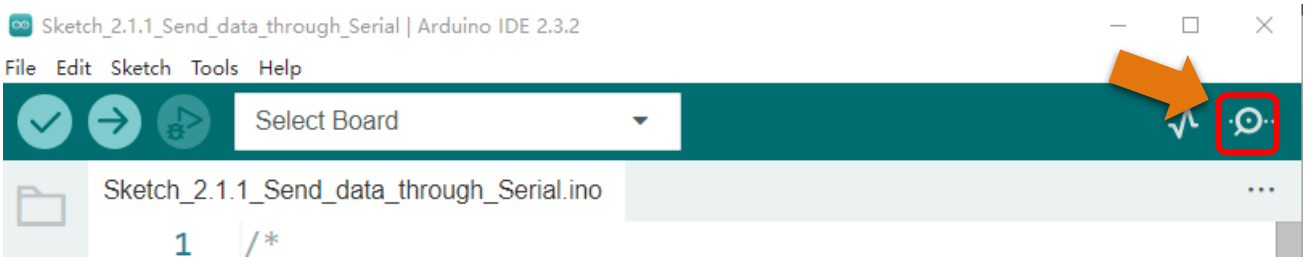


Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The baud rates commonly used are 9600 and 115200.

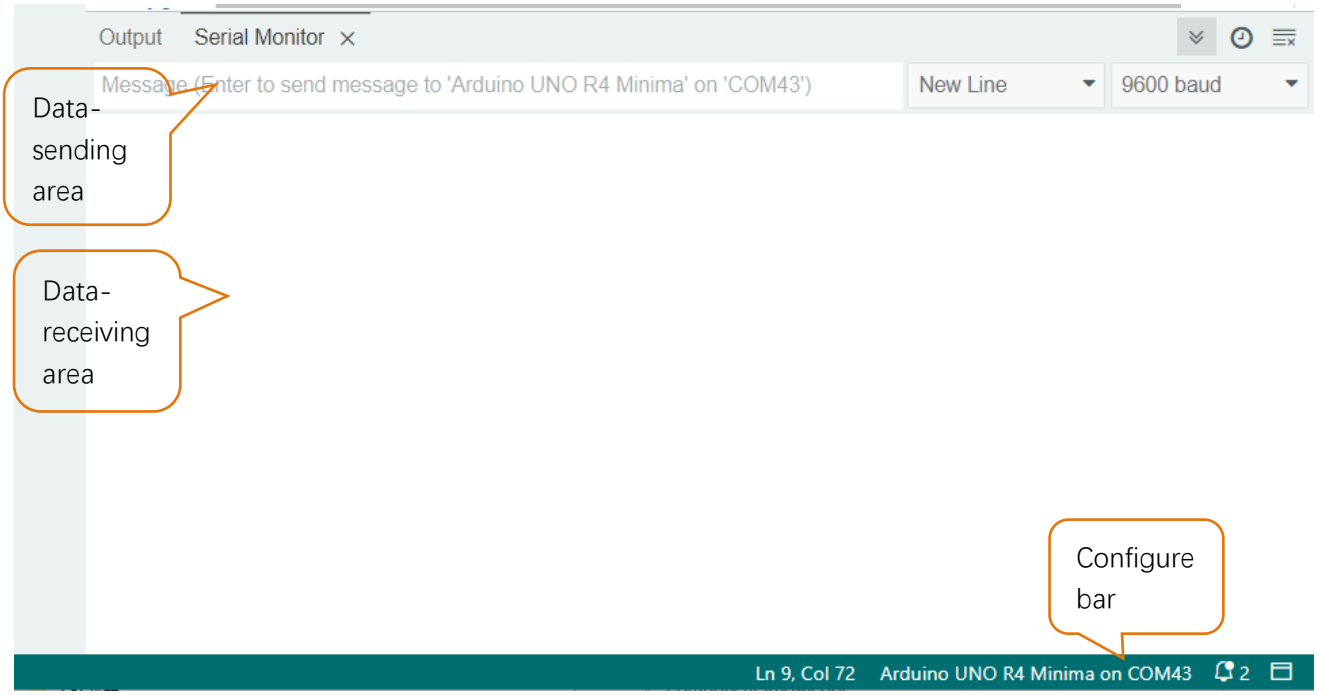
Serial port on Control board

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.

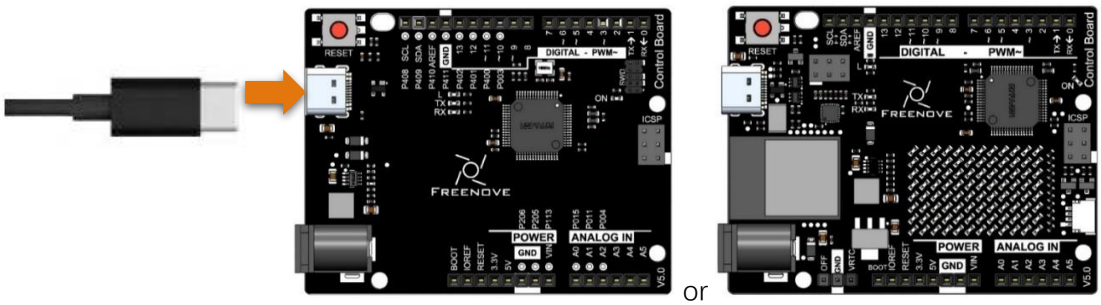


Interface of Serial Monitor window is as follows. If you can't open it, make sure control board had been connected to the computer, and choose the correct serial port in the menu bar "Tools".



Circuit

Connect control board to the computer with USB cable.



or

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

Sketch

Sketch 6.1.1

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;

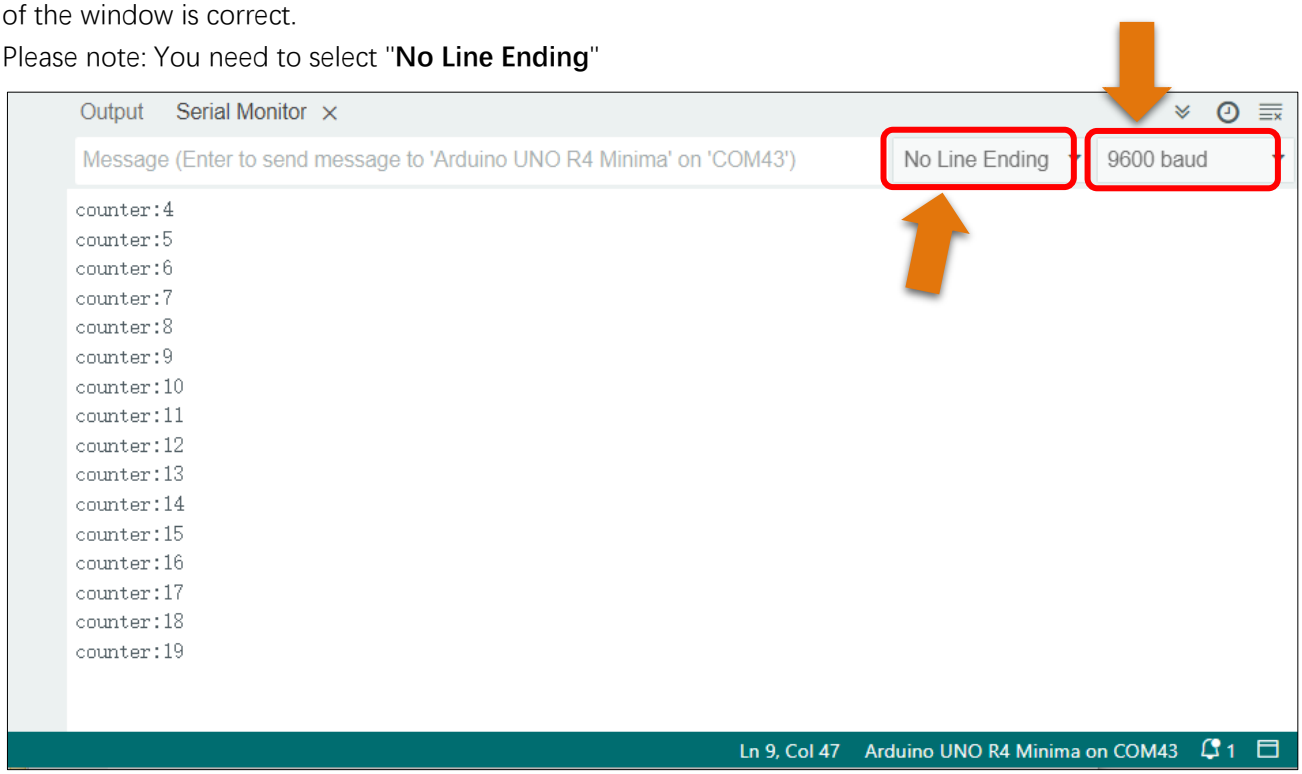
Serial.print(val): Send string, the parameter here is what you want to send;

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.

Please note: You need to select "**No Line Ending**"



Project 6.2 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

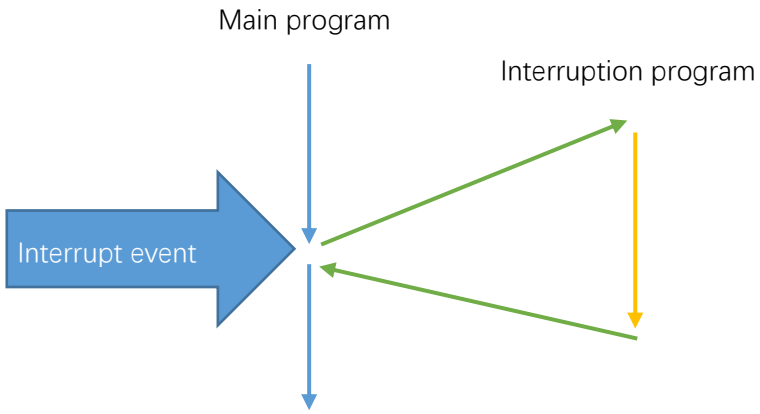
Same with the previous section.

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 6.2.1

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 }
```

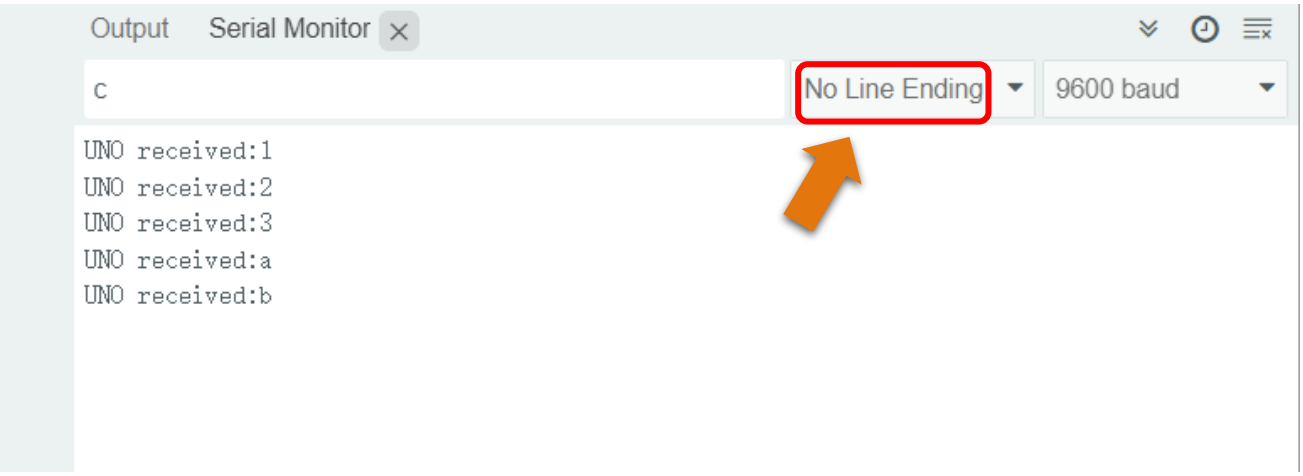
In the setup() function, we initialize the serial port. Then, the loop() function will continuously detect whether there are data to read. if so, it will read the character and send it back.

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.

Please note: You need to select "**No Line Ending**"



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.

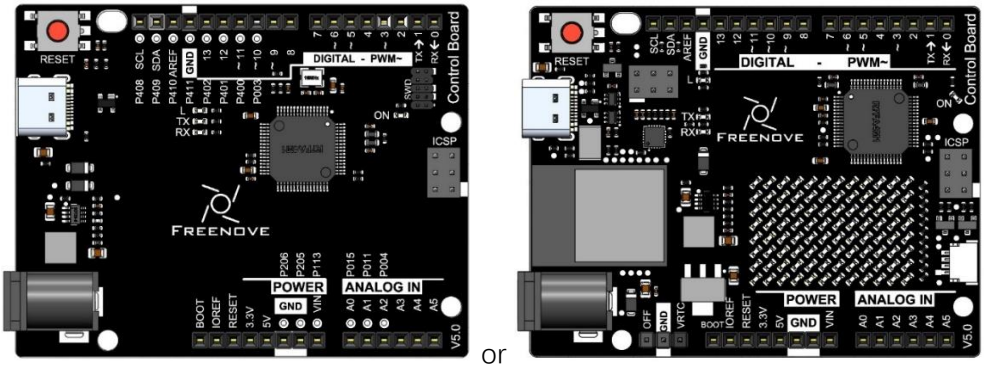
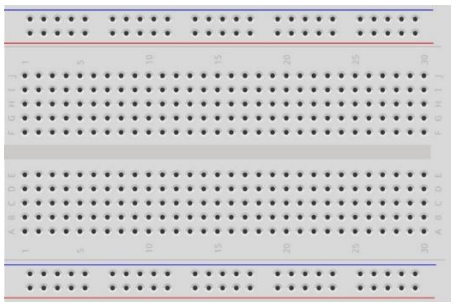
Example: Define char aChar = 'a', bChar = '0', then the decimal value of aChar is 97, bChar will be 48.

Project 6.3 Application of Serial

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

Component List

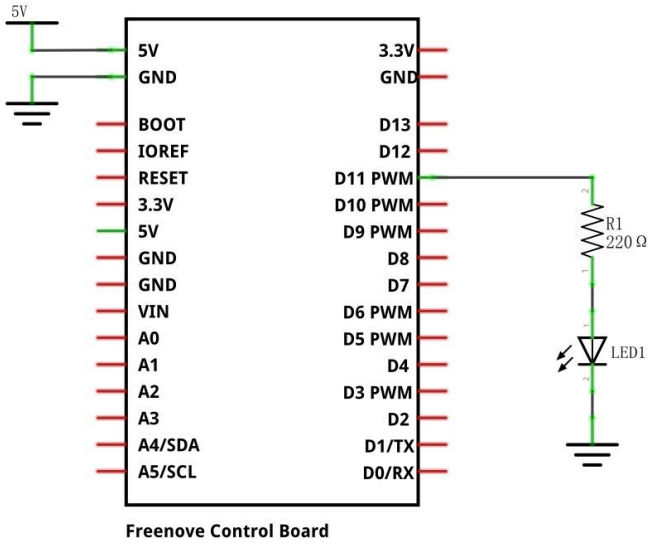
If the kit you bought does not include the following components, you can use only the control board and USB cable to finish this project.

<table>
<tr><td colspan="3">Control board x1
 
or</td></tr>
<tr><td colspan="3">Breadboard x1
  </td></tr>
<tr><td>USB cable x1
  </td><td rowspan="2">LED x1
  </td><td rowspan="2">Resistor 220Ω x1
  </td></tr>
<tr><td>Jumper M/M x2
</td></tr>
</table>

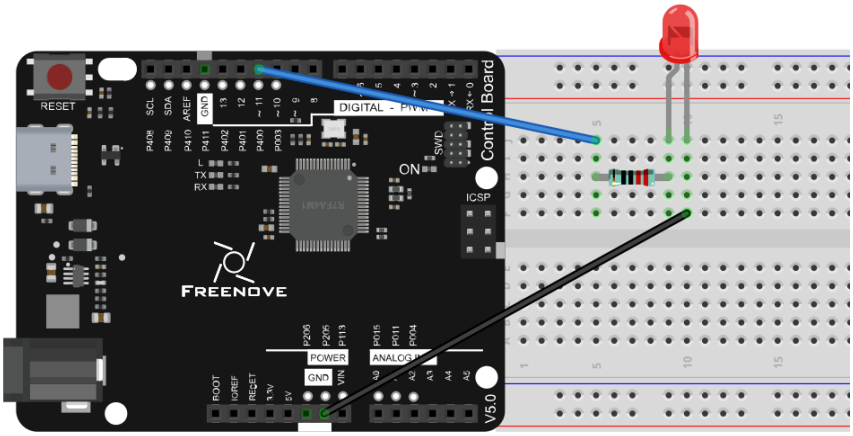
Circuit

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 6.3.1

Code is basically the same with Sketch 6.2.1. But after receiving the data, control board will convert it into PWM duty cycle of output port.

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

void setup() {
  pinMode(ledPin, OUTPUT);              // initialize the LED pin as an output
  Serial.begin(9600);                   // initialize serial port, set baud rate to 9600
  Serial.println("UNO is ready!");      // print the string "UNO is ready!"
}

void loop() {
  if (Serial.available()) {         // judge whether the data has been received
    inInt = Serial.parseInt();      // read an integer
    Serial.print("UNO received:");  // print the string "UNO received:"
    Serial.println(inInt);          // print the received character
    // convert the received integer into PWM duty cycle of ledPin port
    analogWrite(ledPin, constrain(inInt, 0, 255));
  }
}
```

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.

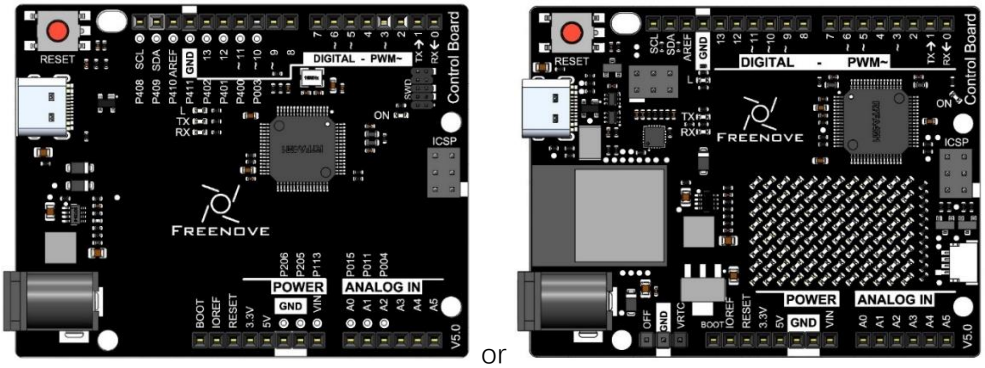
Please note: You need to select "**No Line Ending**"

Chapter 7 Timer

In this chapter, we will use the timer of the board, trigger the timer at intervals, thus allowing the serial port to print messages.

Project 7.1 Serial print using timer

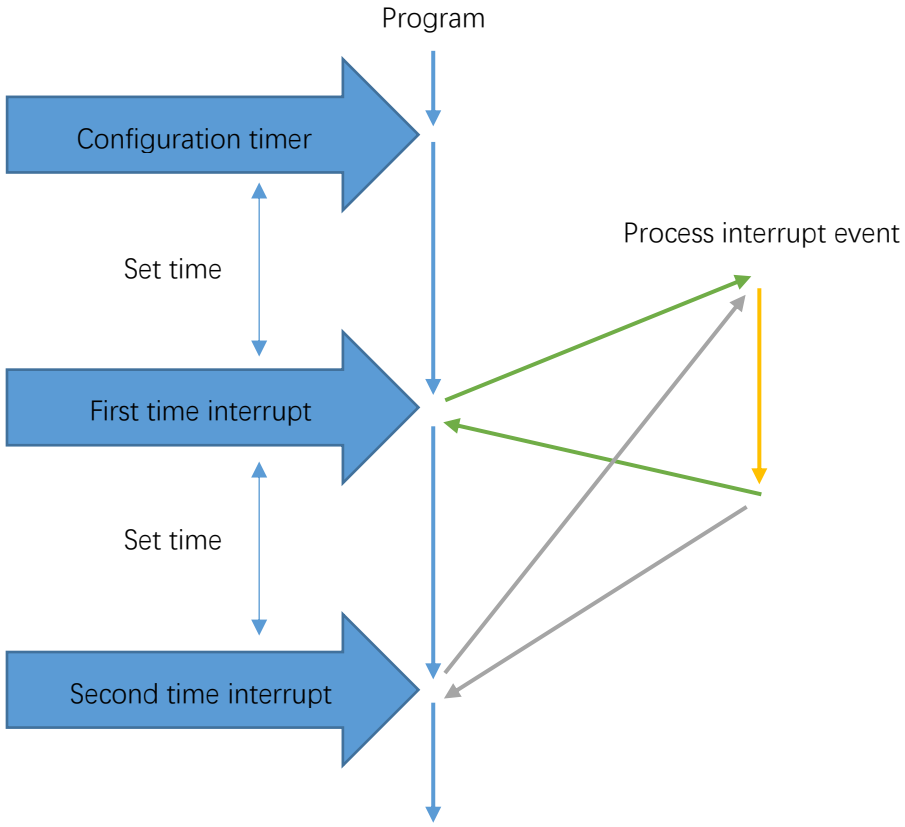
Component List

Control board x1 
USB cable x1 

Code Knowledge

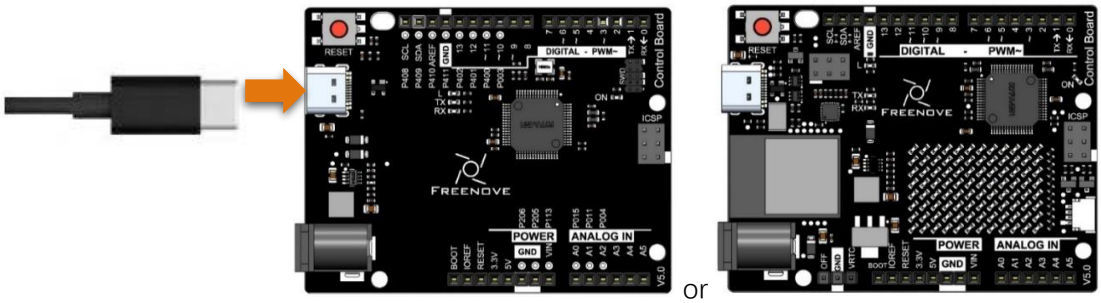
Timer

A Timer can be set to produce an interrupt after a period of time. When a timer interrupt occurs, the processor will jump to the interrupt function to process the interrupt event. And after completion the processing, execution will return to the interrupted location to go on. If you don't close the timer, interrupt will occur at the intervals you set.



Circuit

Connect control board to the computer with USB cable.



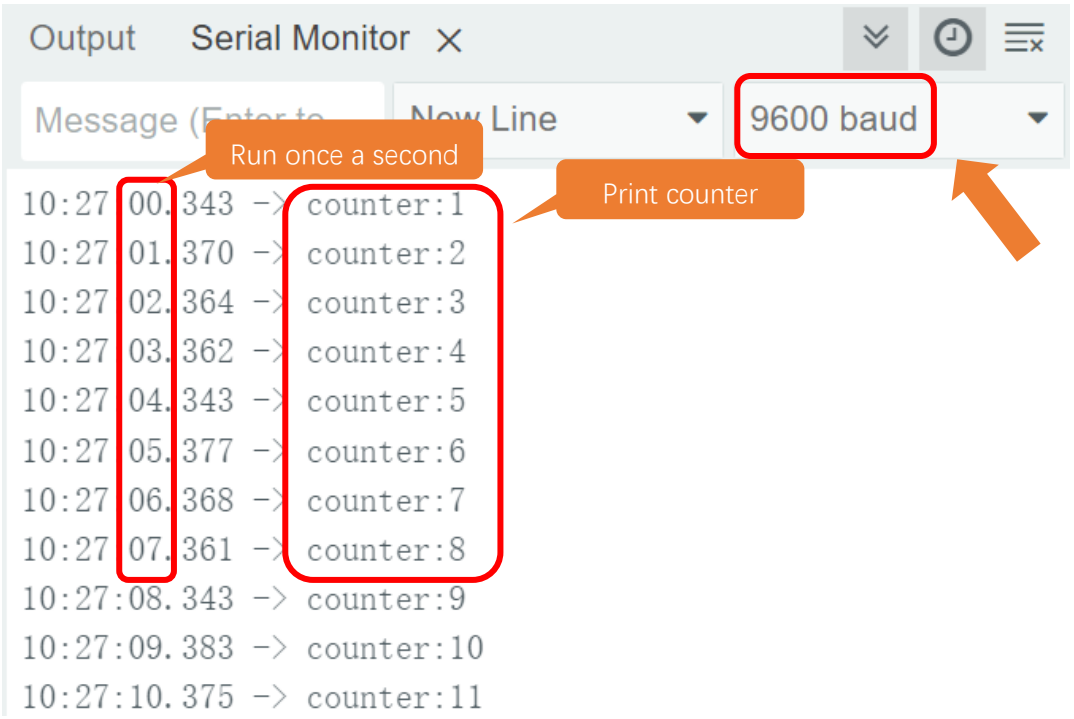
If you need any support, please feel free to contact us via: support@freenove.com

Sketch

Sketch 7.1.1

```
#include <CBTimer.h>    // Contains CallbackTimerR4 Library

int counter;
bool counter_flag;

void callback_func(void) {
  counter_flag = true;
  counter += 1;
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);               // initialize serial port, set baud rate to 9600
  static CBTimer timer;
  timer.begin(1000, callback_func);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(counter_flag == true)
  {
    counter_flag = false;
    Serial.print ("counter:");
    Serial.println(counter);
  }
}
```

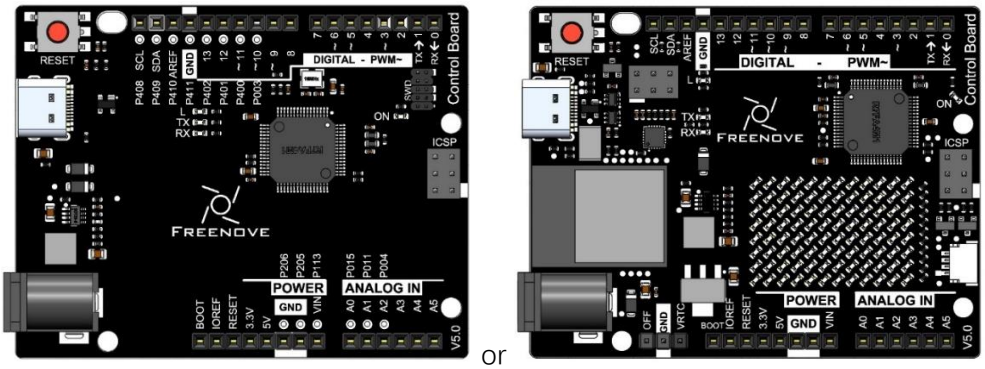
Verify and upload the code, open the Serial Monitor, and then you'll see data sent from control board.
If it is not displayed correctly, check whether the configuration of the Serial Monitor in the lower right corner of the window is correct.


Output
Serial Monitor
New Line
9600 baud
Run once a second
Print counter
10:27 00.343 -> counter:1
10:27 01.370 -> counter:2
10:27 02.364 -> counter:3
10:27 03.362 -> counter:4
10:27 04.343 -> counter:5
10:27 05.377 -> counter:6
10:27 06.368 -> counter:7
10:27 07.361 -> counter:8
10:27:08.343 -> counter:9
10:27:09.383 -> counter:10
10:27:10.375 -> counter:11

Project 7.2 Using timer to implement LED blinking

Component List

Control board x1

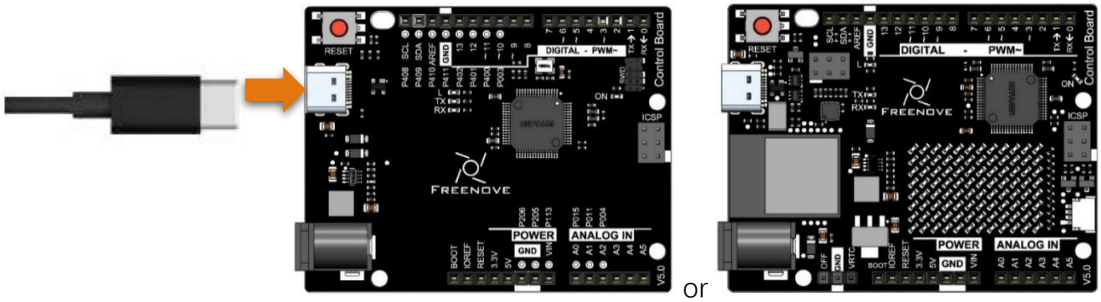


or

USB cable x1

Circuit

Connect control board to the computer with USB cable.



or

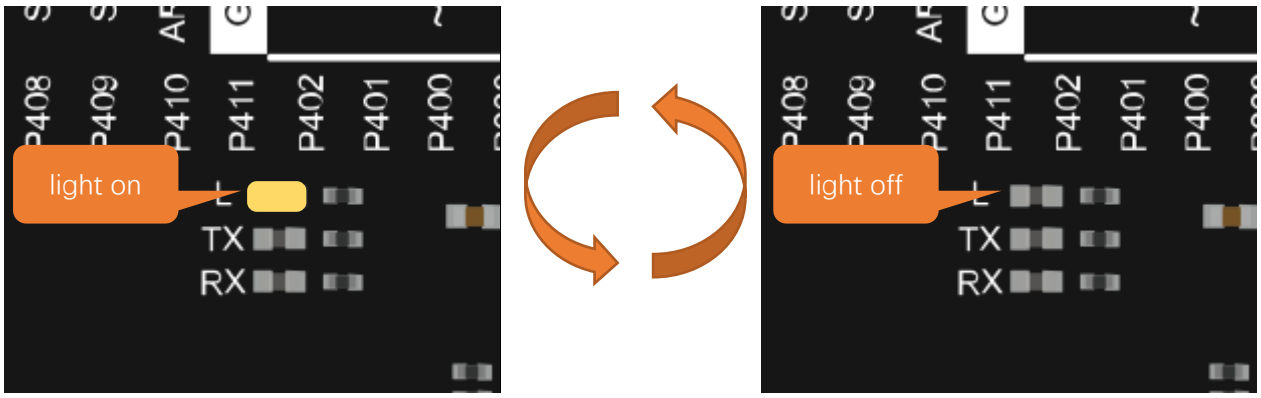
If you need any support, please feel free to contact us via: support@freenove.com

Sketch

Sketch 7.2.1

```
#include <CBTimer.h>    // Contains CallbackTimerR4 Library

const int ledpin = 13;

void callback_func(void) {
  int ledstate = digitalRead(ledpin);
  digitalWrite(ledpin, !ledstate);
}

void setup() {
  Serial.begin(9600);                 // initialize serial port, set baud rate to 9600
  pinMode(ledpin, OUTPUT);
  static CBTimer timer;
  timer.begin(1000, callback_func);
}

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

Verify and upload the code, then you will see the LED light starts to flash with a period of 1 second



Chapter 8 ADC

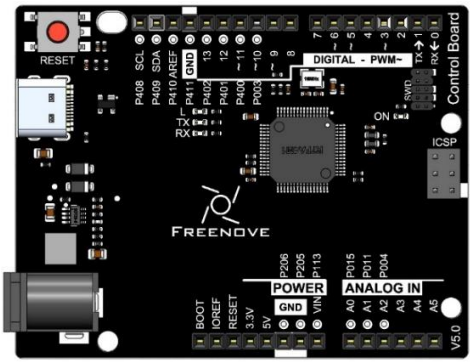
Previously we have learned about the digital ports of the control board and tried to output and input signals. Now, let's learn how to use the analog-to-digital converter (ADC) on the control board. By default, the resolution of the control board ADC is set to 10 bits, which can be updated to 12 bits (0-4096) and 14 bits (0-16383) to improve the accuracy of analog readings.

Project 8.1 ADC

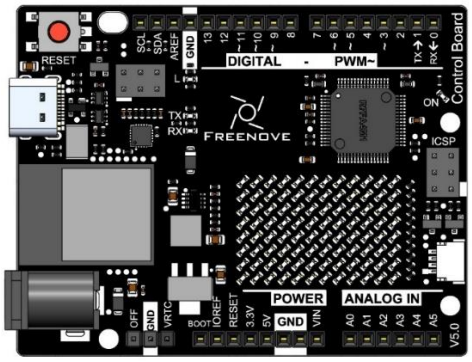
ADC is used to convert analog signals into digital signals. Control chip on the control board has integrated this function. Now let us try to use this function to convert analog signals into digital signals.

Component List

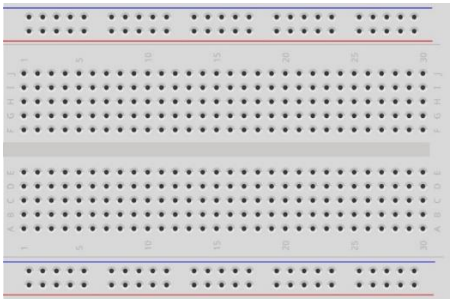
Control board x1



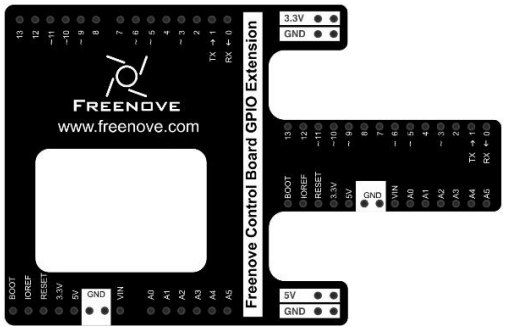
or

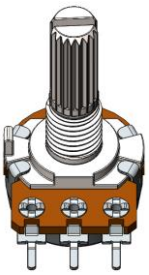


Breadboard x1



GPIO Extension Board x1

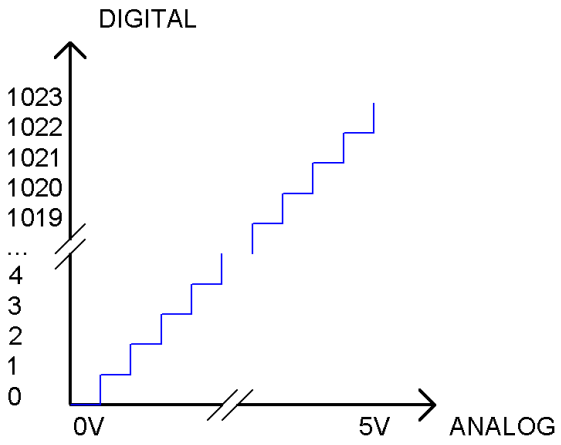


USB cable x1 	Rotary potentiometer x1 
Jumper M/M x3 	

Circuit Knowledge

ADC

An ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s.The onboard ADC module is set to 10 bits by default, which means the resolution is 2^10=1024, so its range (at 5V) will be evenly divided into 1024 parts. By default, the resolution of the onboard ADC is set to 10 bits, which can be updated to 12 bits (0-4096) and 14 bits (0-16383) resolution to improve the accuracy of the analog readings. Any analog value can be mapped to one digital value using the resolution of the converter. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V-5/1024V corresponds to digital 0;
Subsection 2: the analog in rang of 5 /1024V-2*5/1024V corresponds to digital 1;

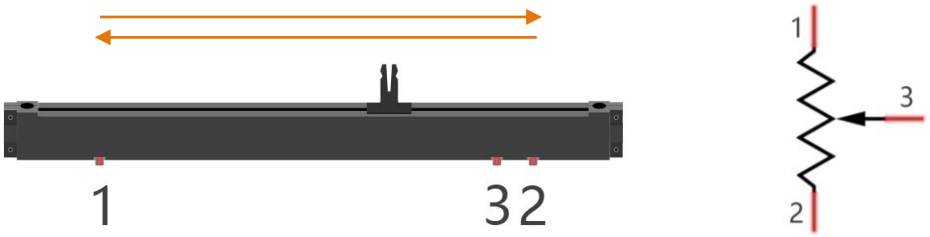
The following analog signal will be divided accordingly.

Component Knowledge

Potentiometer

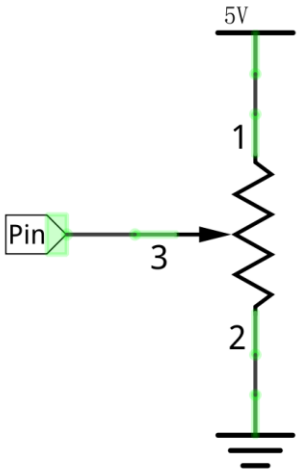
Potentiometer is a resistive element with three Terminal parts. Unlike the resistors that we have used thus far in our project which have a fixed resistance value, the resistance value of a potentiometer can be adjusted. A

potentiometer is often made up by a resistive substance (a wire or carbon element) and movable contact brush. When the brush moves along the resistor element, there will be a change in the resistance of the potentiometer's output side (3) (or change in the voltage of the circuit that is a part). The illustration below represents a linear sliding potentiometer and its electronic symbol on the right.



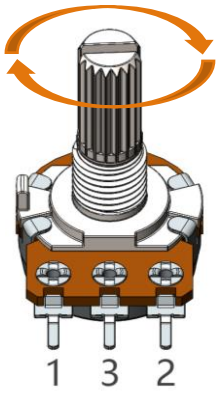
Between potentiometer pin 1 and pin 2 is the resistive element (a resistance wire or carbon) and pin 3 is connected to the brush that makes contact with the resistive element. In our illustration, when the brush moves from pin 1 to pin 2, the resistance value between pin 1 and pin 3 will increase linearly (until it reaches the highest value of the resistive element) and at the same time the resistance between pin 2 and pin 3 will decrease linearly and conversely down to zero. At the midpoint of the slider the measured resistance values between pin 1 and 3 and between pin 2 and 3 will be the same.

In a circuit, both sides of resistive element are often connected to the positive and negative electrodes of power. When you slide the brush "pin 3", you can get variable voltage within the range of the power supply.



Rotary potentiometer

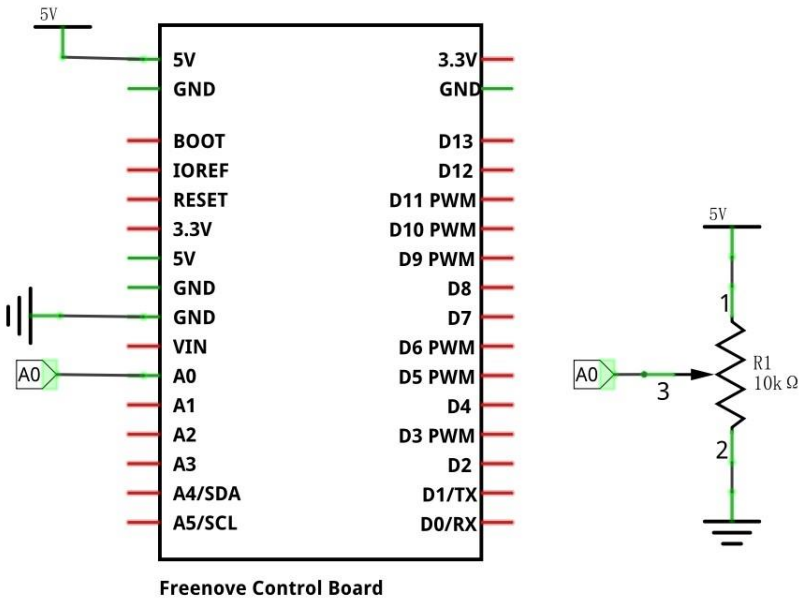
Rotary potentiometer and linear potentiometer have the same function; the only difference being the physical action being a rotational rather than a sliding movement.



Circuit

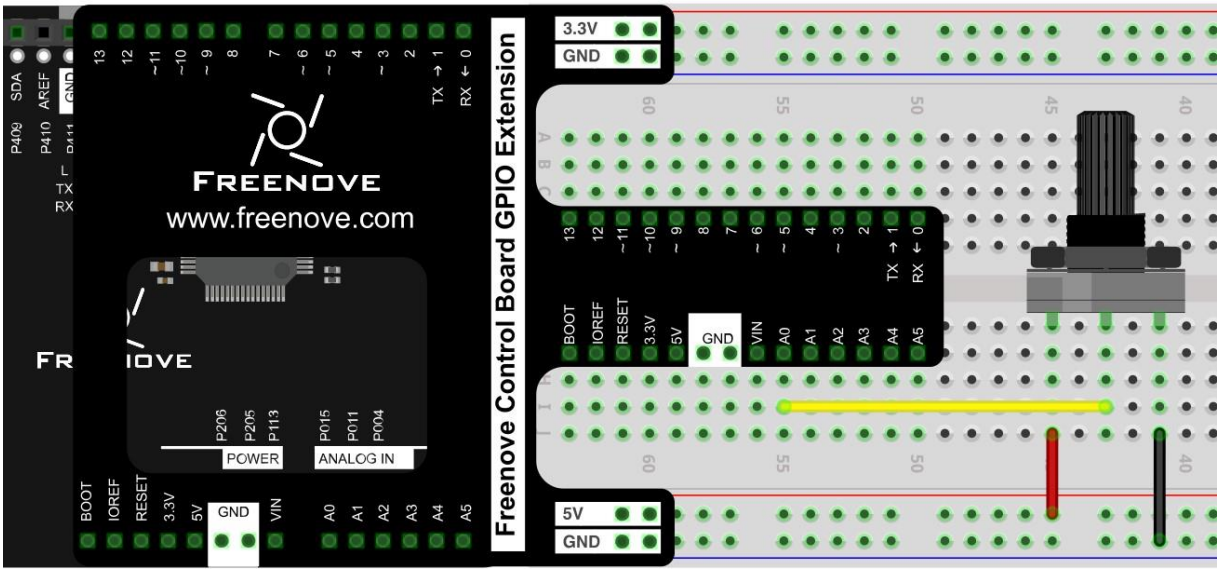
Use pin A0 on the control board to detect the voltage of rotary potentiometer.

Schematic diagram



Freenove Control Board

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



Sketch

Sketch 8.1.1

Now, write code to detect the voltage of rotary potentiometer, and send the data to Serial Monitor window of Arduino IDE through serial port. The code here demonstrates how to set the ADC to 14 bits precision.

```
int adcValue;   // Define a variable to save ADC value
float voltage;  // Define a variable to save the calculated voltage value

void setup() {
  Serial.begin(9600);  // Initialize the serial port and set the baud rate to 9600
  // analogReadResolution(10);  //change to 10-bit resolution
  // analogReadResolution(12);  //change to 12-bit resolution
  analogReadResolution(14);  //change to 14-bit resolution
}

void loop() {
  // int reading = analogRead(A0);          // returns a value between 0-1023
  // voltage = reading * (5.0 / 16383.0);  // Calculate voltage according to digital
  // int reading = analogRead(A0);          // returns a value between 0-4095
  // voltage = reading * (5.0 / 16383.0);  // Calculate voltage according to digital
  int reading = analogRead(A0);          // returns a value between 0-16383
  voltage = reading * (5.0 / 16383.0);  // Calculate voltage according to digital
  Serial.print("convertValue:");
  Serial.println(reading);
  Serial.print("Voltage:");
  Serial.println(voltage);
  delay(500);
}
```

From the code, we get the ADC value of pin A0, then convert it into voltage and sent to the serial port.

Verify and upload the code, open the Serial Monitor, and then you will see the original ADC value and converted voltage sent from control board.

Turn the rotary potentiometer shaft, and you can see the voltage change.

If you want to change the resolution of th ADC, you only need to modify the parameter of the function analogReadResolution(int bits); among which, bits is the bit number of the ADC. If the parameter is not set, it will be 10 bits by default. As this project applies 10-bit ADC, you do not need to set the parameter for this function.

```
1  // analogReadResolution(10);  //change to 10-bit resolution
2  // analogReadResolution(12);  //change to 12-bit resolution
3  analogReadResolution(14);  //change to 14-bit resolution
```

Read the ADC value and calculate the voltage based on the set ADC bit number.

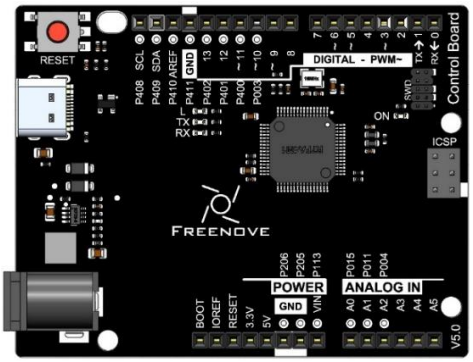
```
   // int reading = analogRead(A0);          // returns a value between 0-1023
   // voltage = reading * (5.0 / 16383.0);  // Calculate voltage according to digital
   // int reading = analogRead(A0);          // returns a value between 0-4095
   // voltage = reading * (5.0 / 16383.0);  // Calculate voltage according to digital
   int reading = analogRead(A0);          // returns a value between 0-16383
   voltage = reading * (5.0 / 16383.0);  // Calculate voltage according to digital
```

Project 8.2 Control LED by Potentiometer

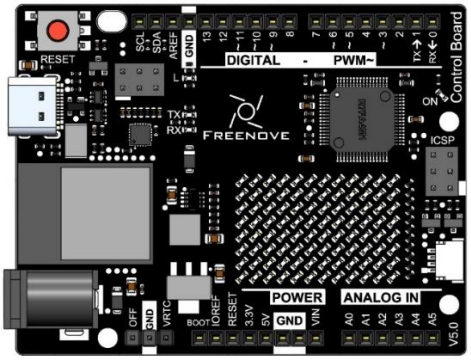
In the previous section, we have finished reading ADC value and converting it into voltage. Now, we will try to use potentiometer to control the brightness of LED.

Component List

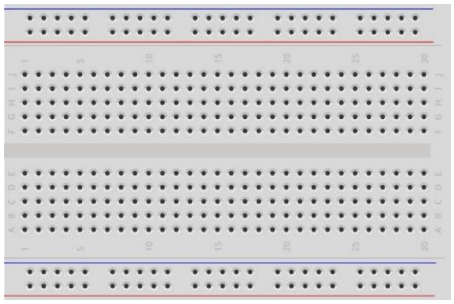
Control board x1



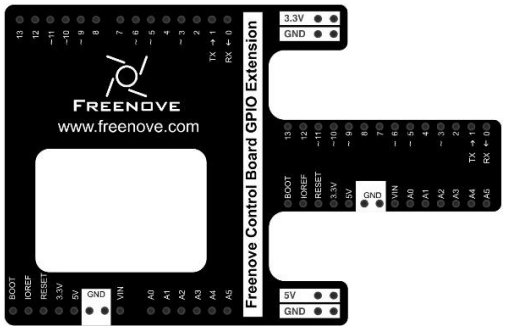
or



Breadboard x1

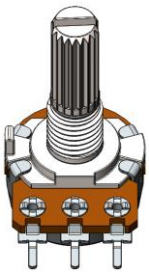


GPIO Extension Board x1



USB cable x1

Rotary potentiometer x1



LED x1

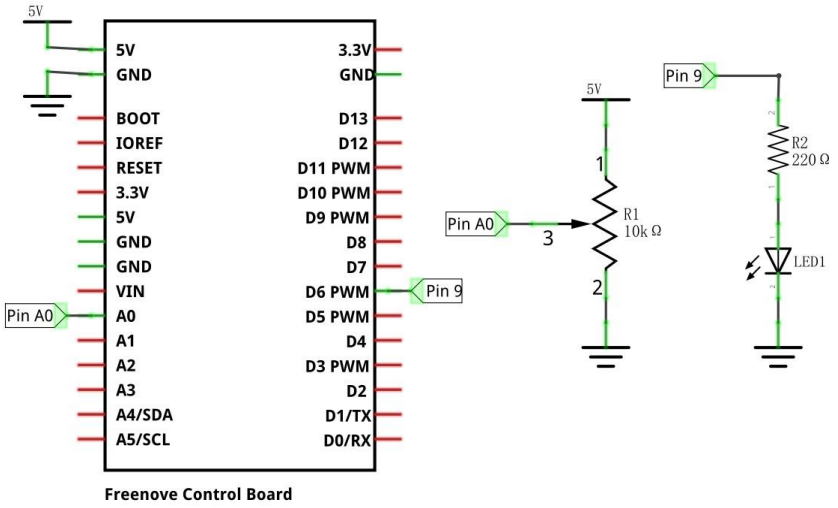
Resistor 220Ω x1

Jumper M/M x5

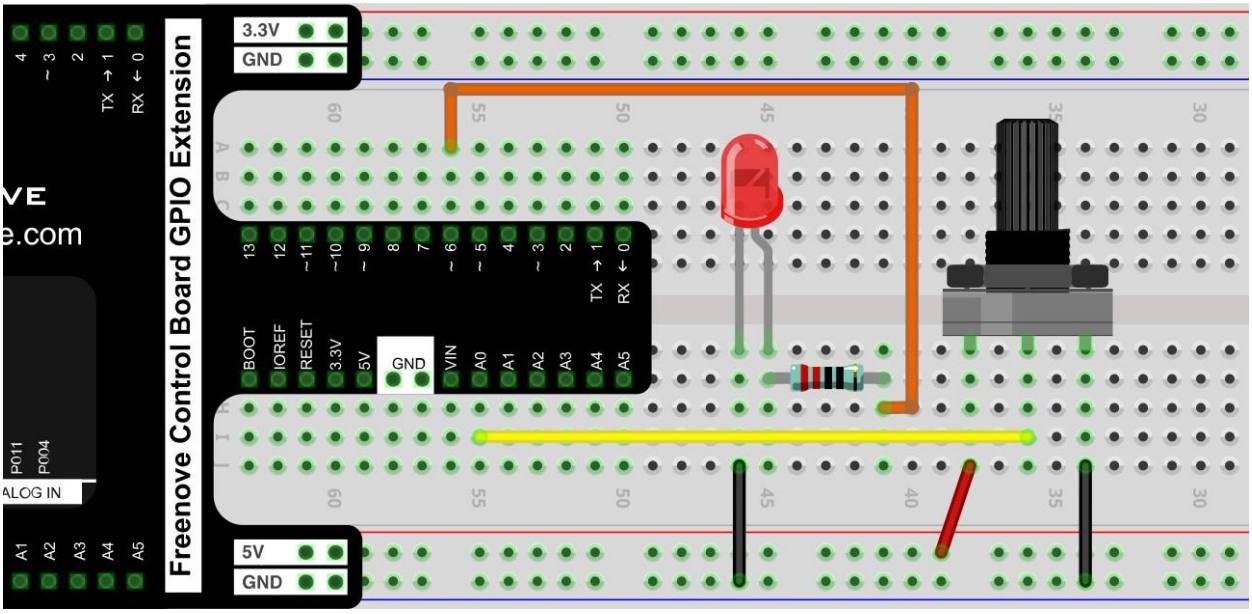
Circuit

Use pin A0 on control board to detect the voltage of rotary potentiometer, and use pin 9 to control one LED.

Schematic diagram



Hardware connection



Sketch

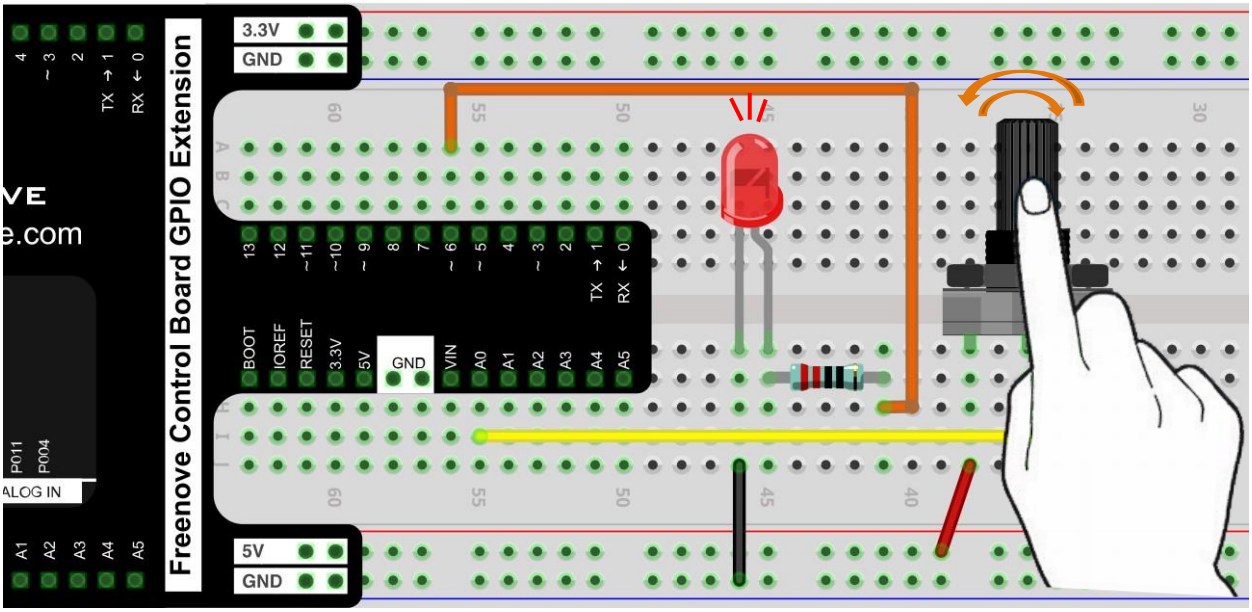
Sketch 8.2.1

Now, write the code to detect the voltage of rotary potentiometer, and control LED to emit light with different brightness according to that.

```
int adcValue;      // Define a variable to save the ADC value
int ledPin = 9;    // Use D9 on Freenove UNO to control the LED

void setup() {
  pinMode(ledPin, OUTPUT);              // Initialize the LED pin as an output
}

void loop() {
  adcValue = analogRead(A0);            // Convert the analog of A0 port to digital
  // Map analog to the 0-255 range, and works as PWM duty cycle of ledPin port
  analogWrite(ledPin, map(adcValue, 0, 1023, 0, 255));
}
```

In the code, we get the ADC value of pin A0 and map it to PWM duty cycle of LED pin port. According to different LED brightness, we can see the changes of voltage easily.

Verify and upload the code, turn the rotary potentiometer shaft, you will see the LED brightness change.



Chapter 9 RGB LED

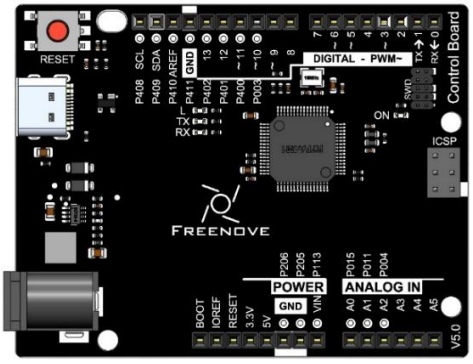
Earlier, we have learned to apply the analog port and ADC of the control board. Now, we'll use ADC to control RGB LED.

Project 9.1 Multicolored LED

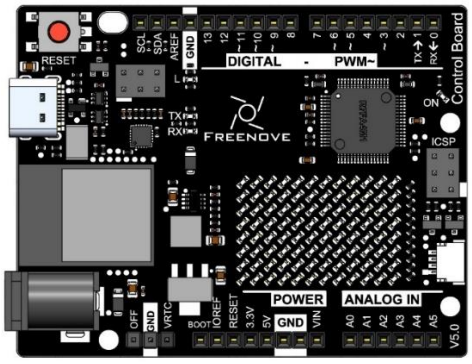
In the previous section, we have finished controlling the RGB LED to emit light with different color and brightness through three potentiometers. Now, we will try to make RGB LED emit multicolored lights automatically.

Component List

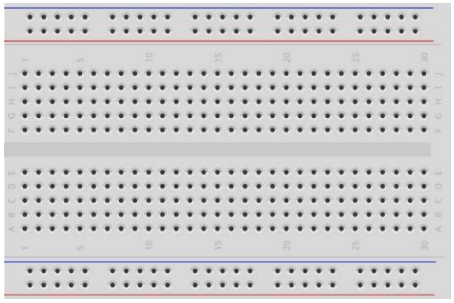
Control board x1



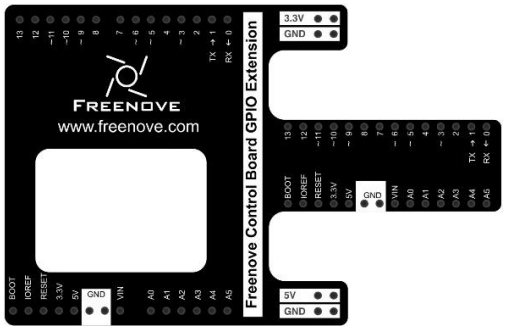
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

RGB LED x1

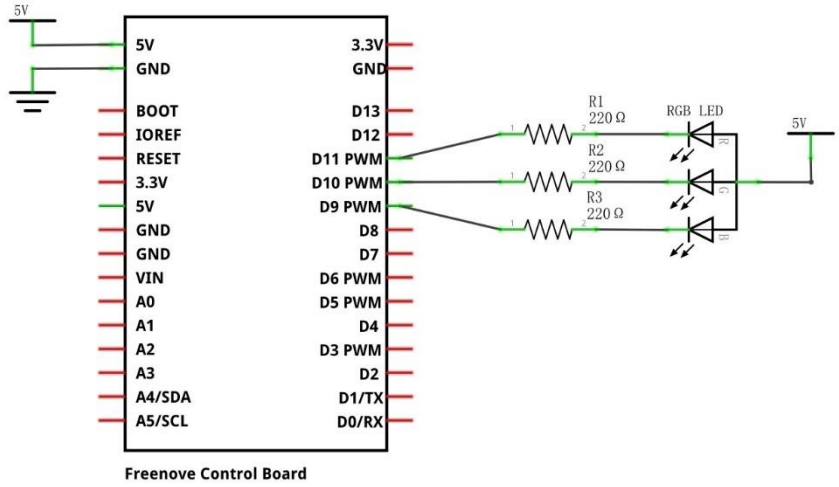
Resistor 220Ω x3

Jumper M/M x4		

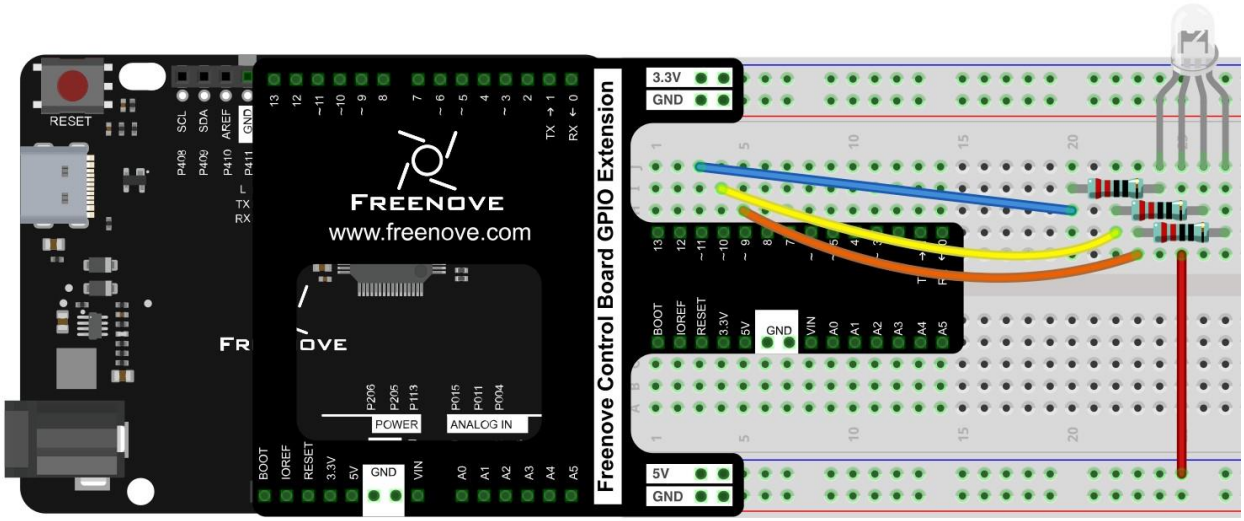
Circuit

Use pin 9, 10, 11 of the control board to control RGB LED.

Schematic diagram



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



Sketch

Sketch 9.1.1

Now, write code to generate three random numbers, and convert them into PWM duty cycle to control the three LEDs of RGB LED.

```
int ledPinR = 11;   // the number of the LED R pin
int ledPinG = 10;   // the number of the LED G pin
int ledPinB = 9;    // the number of the LED B pin

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPinR, OUTPUT);
  pinMode(ledPinG, OUTPUT);
  pinMode(ledPinB, OUTPUT);
}

void loop() {
  // Generate three random numbers between 0-255 as the output PWM duty cycle of ledPin
  rgbLedDisplay(random(256), random(256), random(256));
  delay(500);
}

void rgbLedDisplay(int red, int green, int blue) {
  // Set three ledPin to output the PWM duty cycle
  analogWrite(ledPinR, constrain(red, 0, 255));
  analogWrite(ledPinG, constrain(green, 0, 255));
  analogWrite(ledPinB, constrain(blue, 0, 255));
}
```

In the code, we create three random numbers, and convert them into PWM duty cycle to control the three LEDs of RGB LED to emit light with different brightness. At regular intervals, a new random number will be created, so RGB LED will start flashing light with different colors and brightness.

random(min, max)

random (min, max) function is used to generate random number, and it will return a random value in the range (min, Max-1).
You can also use random (max) function, the function set the minimum value into 0 by default, and returns a random value in the range (0, Max-1).

Verify and upload the code, and RGB LED starts flashing with different colors and brightness.

Chapter 10 Buzzer

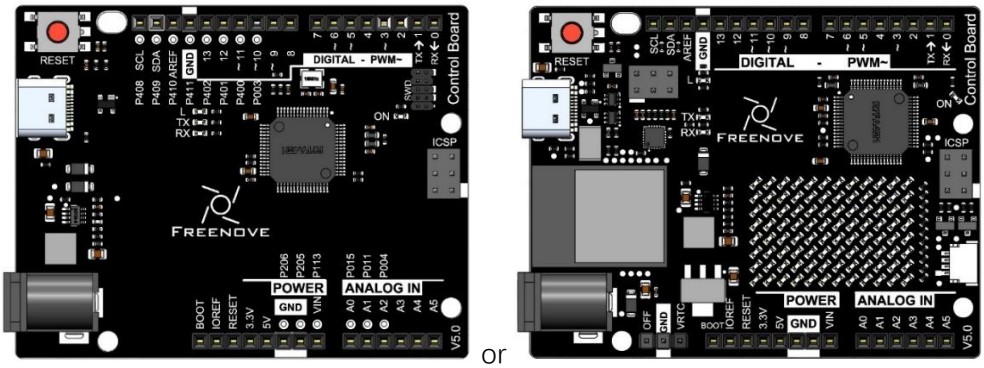
Earlier, we have used control board and basic electronic components to carry out a series of interesting projects. Now, let's learn how to use some integrated electronic components and modules. These modules are usually integrated with a number of electronic components, so they have special features and usages.
In this chapter, we'll use a sounding module, buzzer. It has two types: active and passive buzzer.

Project 10.1 Active Buzzer

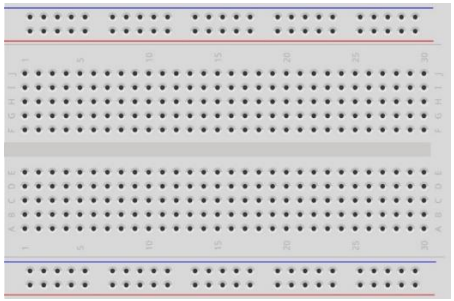
First, let's study some knowledge about active buzzer.

Component List

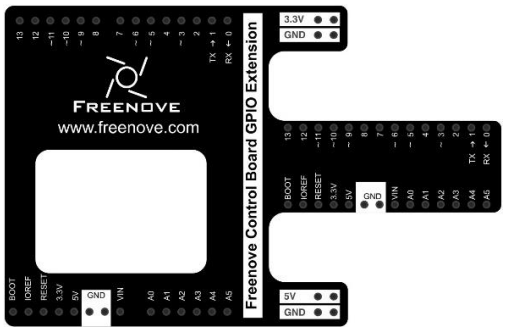
Control board x1



Breadboard x1



GPIO Extension Board x1



USB cable x1

Jumper M/M x6

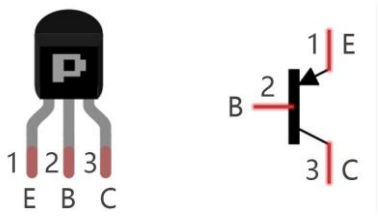
NPN transistorx1	Active buzzer x1	Push button x1	Resistor 1kΩ x1	Resistor 10kΩ x2
				

Component knowledge

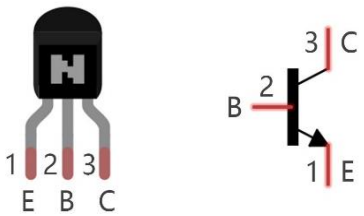
Transistor

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current (think of a transistor as an electronic “amplifying or switching device”). Transistors can be used to amplify weak signals, or to work as a switch. Transistors have three electrodes(PINs): base (b), collector (c) and emitter (e). When there is current passing between "be", then "ce" will have a several-fold current increase(transistor magnification), in this configuration the transistor acts as an amplifier. When current produced by "be" exceeds a certain value, "ce" will limit the current output. at this point the transistor is working in its saturation region and acts like a switch. Transistors are available as two types as shown below: PNP and NPN,

PNP transistor



NPN transistor



In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

Thanks to the transistors’ characteristics, they are often used as switches in digital circuits. As micro-controllers output current capacity is very weak, we will use a transistor to amplify its current in order to drive components requiring higher current.

Buzzer

A buzzer is an audio component. They are widely used in electronic devices such as calculators, electronic alarm clocks, automobile fault indicators, etc. There are both active and passive types of buzzers. Active buzzers have oscillator inside, these will sound as long as power is supplied. Passive buzzers require an external oscillator signal (generally using PWM with different frequencies) to make a sound.

Active buzzer Passive buzzer

(A white label is attached to the active buzzer)

Active buzzers are easier to use. Generally, they only make a specific sound frequency. Passive buzzers require an external circuit to make sounds, but passive buzzers can be controlled to make sounds of various frequencies. The resonant frequency of the passive buzzer in this Kit is 2kHz, which means the passive buzzer is the loudest when its resonant frequency is 2kHz.

How to identify active and passive buzzer?

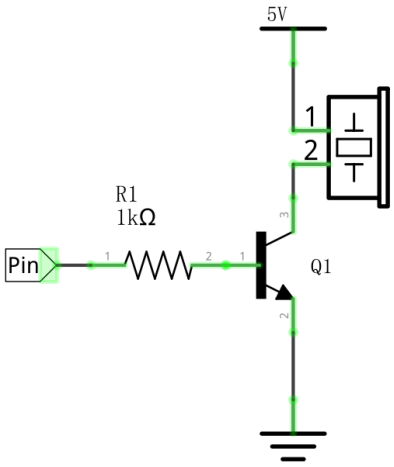
1. As a rule, there is a label on an active buzzer covering the hole where sound is emitted, but there are exceptions to this rule.
2. Active buzzers are more complex than passive buzzers in their manufacture. There are many circuits and crystal oscillator elements inside active buzzers; all of this is usually protected with a waterproof coating (and a housing) exposing only its pins from the underside. On the other hand, passive buzzers do not have protective coatings on their underside. From the pin holes, view of a passive buzzer, you can see the circuit board, coils, and a permanent magnet (all or any combination of these components depending on the model.

Active buzzer bottom

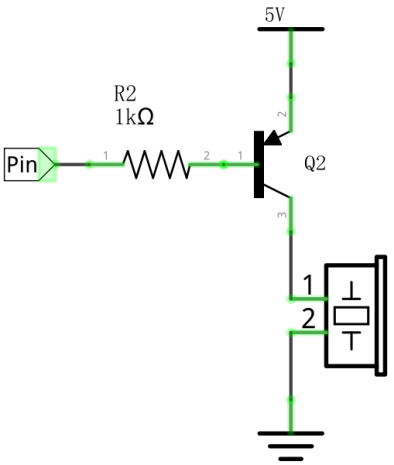
Passive buzzer bottom

Buzzers need relatively larger current when they work. But generally, microcontroller port can't provide enough current for them. In order to control buzzer by control board, transistor can be used to drive a buzzer indirectly.

When we use a NPN transistor to drive a buzzer, we often use the following method. If control board pin outputs high level, current will flow through R1 (Resistor 1), the transistor conducts current and the buzzer will make sounds. If control board pin outputs low level, no current will flow through R1, the transistor will not conduct current and buzzer will remain silent (no sounds).



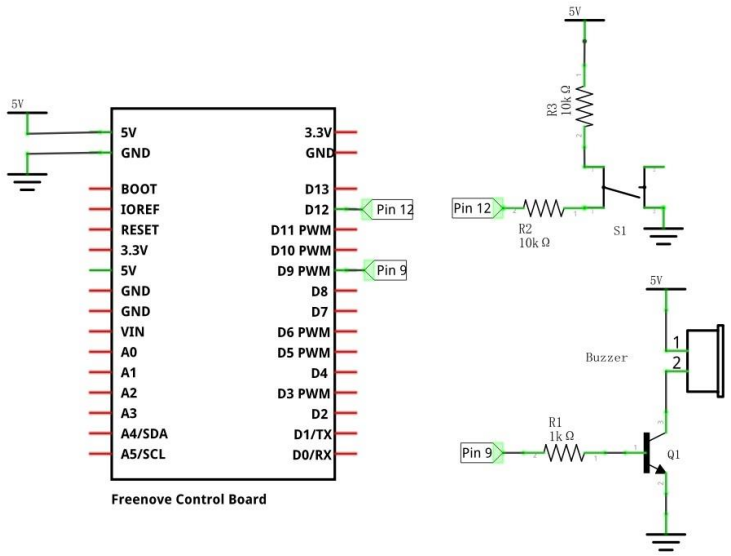
When we use a PNP transistor to drive a buzzer, we often use the following method. If control board pin outputs low level, current will flow through R1. The transistor conducts current and the buzzer will make sounds. If control board pin outputs high level, no current flows through R1, the transistor will not conduct current and buzzer will remain silent (no sounds). Below are the circuit schematics for both a NPN and PNP transistor to power a buzzer.



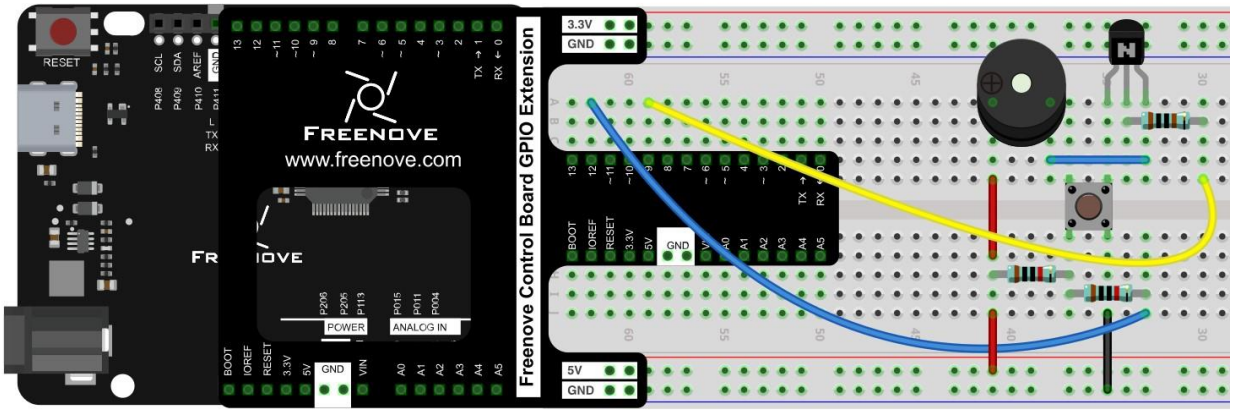
Circuit

Use pin 12 of control board to detect the state of push button switch, and pin 9 to drive active buzzer.

Schematic diagram



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



Sketch

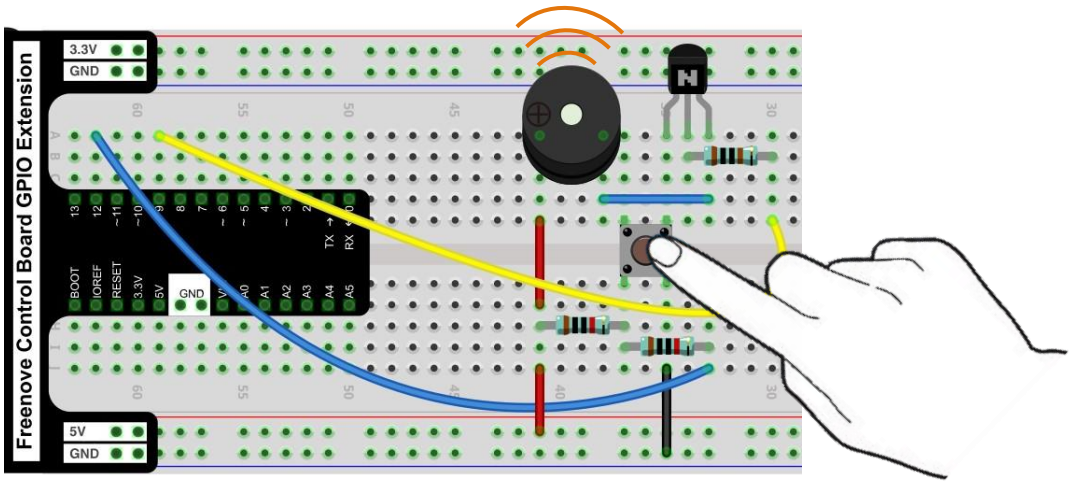
Sketch 10.1.1

Now, write code to detect the state of push button, and drive active buzzer to make a sound when it is pressed.

```
int buttonPin = 12;    // the number of the push button pin
int buzzerPin = 9;     // the number of the buzzer pin

void setup() {
  pinMode(buttonPin, INPUT);  // Set push button pin into input mode
  pinMode(buzzerPin, OUTPUT); // Set Buzzer pin into output mode
}

void loop() {
 if (digitalRead(buttonPin) == HIGH)// If the pin is high level,the button is not pressed.
    digitalWrite(buzzerPin, LOW);     // Turn off Buzzer
  else                                // The button is pressed, if the pin is low level
    digitalWrite(buzzerPin, HIGH);    // Turn on Buzzer
}
```

In the code, we check the state of push button switch. When it is pressed, the output high level controls transistor to get conducted, and drives active buzzer to make a sound.

Verify and upload the code, press the push button, and the active buzzer will make a sound.

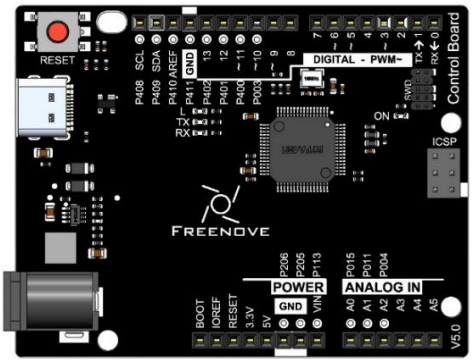


Project 10.2 Passive Buzzer

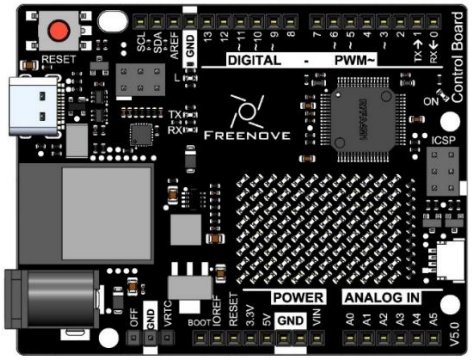
In the previous section, we have finished using transistor to drive an active buzzer to beep. Now, we will try to use a passive buzzer to make a sound with different frequency.

Component List

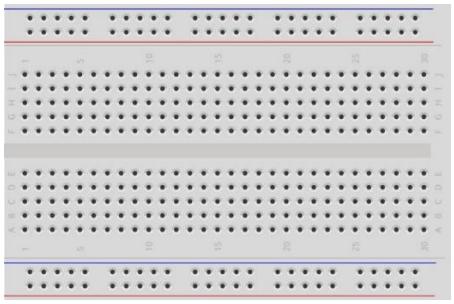
Control board x1



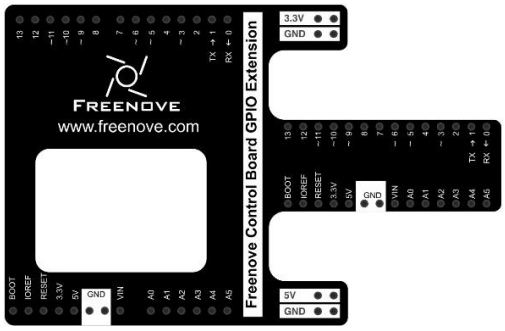
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

Jumper M/M x4

NPN transistor x1

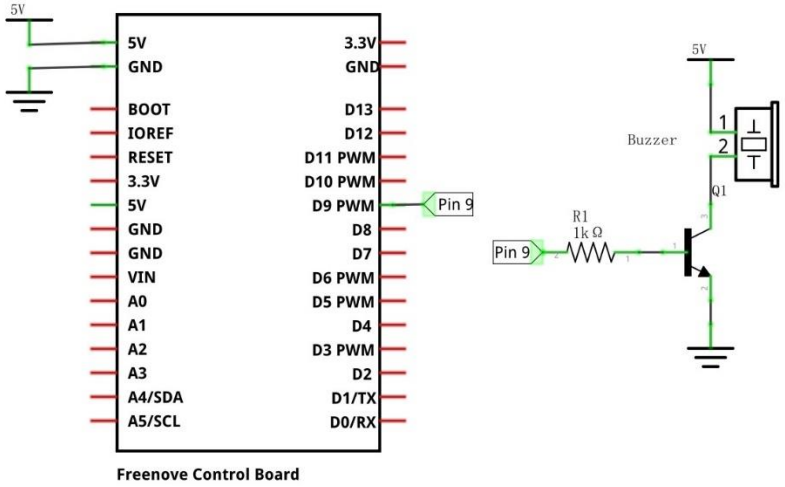
Passive buzzer x1

Resistor 1kΩ x1

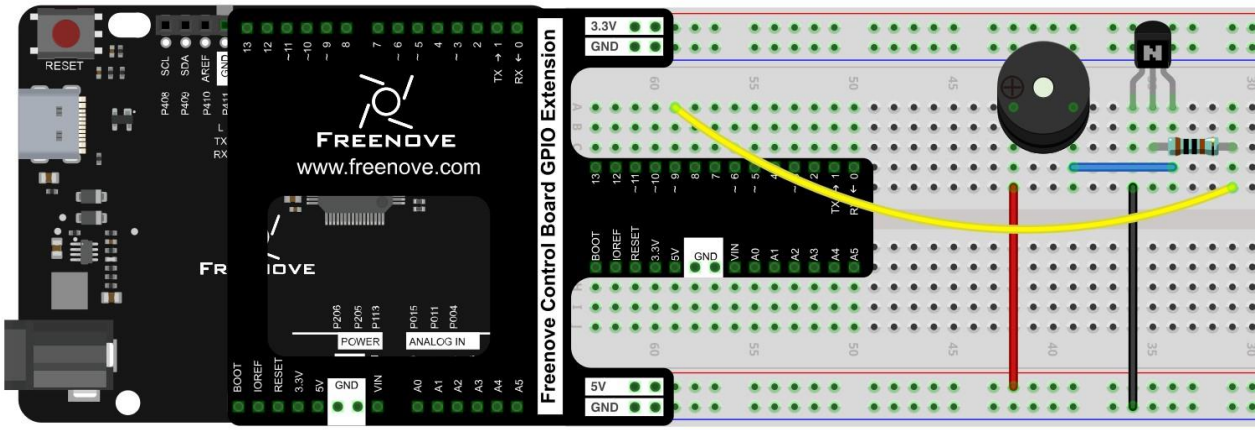
Circuit

Use pin 9 port of control board to drive a passive buzzer.

Schematic diagram



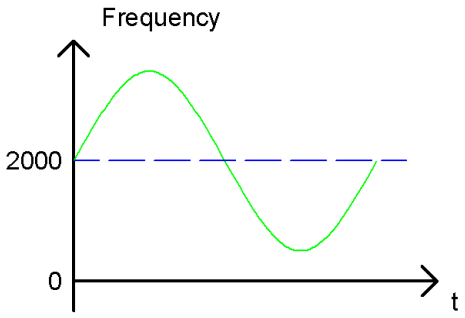
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch 10.2.1

Now, write code to drive a passive buzzer to make a warning sound. The frequency of the passive buzzer conforms roughly to the following sine curve over time:



Output PWM waves with different frequency to the port, which is connected to the transistor, to drive buzzer to make a sound with different frequency.

```
1  int buzzerPin = 9;     // the number of the buzzer pin
2  float sinVal;          // Define a variable to save sine value
3  int toneVal;           // Define a variable to save sound frequency
4
5  void setup() {
6    pinMode(buzzerPin, OUTPUT); // Set Buzzer pin to output mode
7  }
8
9  void loop() {
10   for (int x = 0; x < 360; x++) {        // X from 0 degree->360 degree
11     sinVal = sin(x * (PI / 180));        // Calculate the sine of x
12     toneVal = 2000 + sinVal * 500;       // Calculate sound frequency according to the sine of x
13     tone(buzzerPin, toneVal);            // Output sound frequency to buzzerPin
14     delay(1);
15   }
16 }
```

In the code, use one loop to control the sound frequency, varying according to sine curve in the range of 2000±500.

```
10   for (int x = 0; x < 360; x++) {        // X from 0 degree->360 degree
11     sinVal = sin(x * (PI / 180));        // Calculate the sine of x
12     toneVal = 2000 + sinVal * 500;       // Calculate sound frequency according to the sine of x
13     tone(buzzerPin, toneVal);            // Output sound frequency to buzzerPin
14     delay(1);
15   }
```

The parameter of sin() function is radian, so we need convert unit of π from angle to radian first .

tone(pin, frequency)
Generates a square wave of the specified frequency (and 50% duty cycle) on a pin.

Verify and upload the code, passive buzzer starts making a warning sound.

You can try using PNP transistor to complete the project of this chapter again.

Chapter 11 DAC

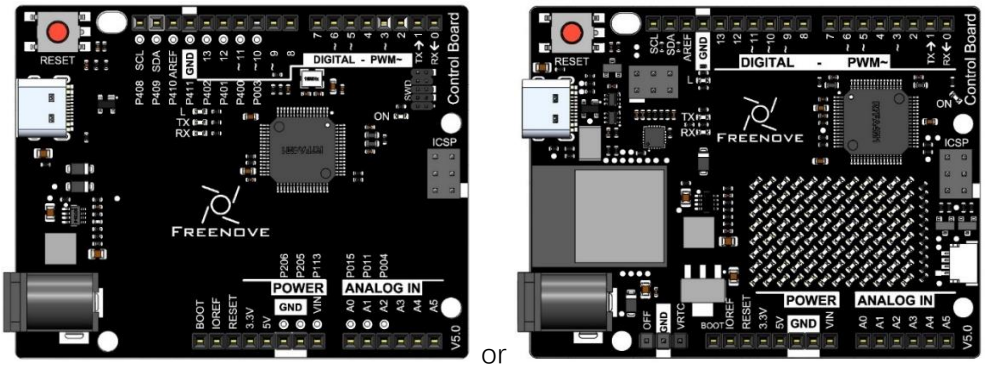
In this chapter, we learn about DAC (Digital to Analog Converter). The Control board (V5) has a built-in DAC (Digital to Analog Converter) which is used to convert digital signals into analog signals.

Project 11.1 DAC

In this project, let's learn the DAC function of the control board and use it to control a passive buzzer to play music.

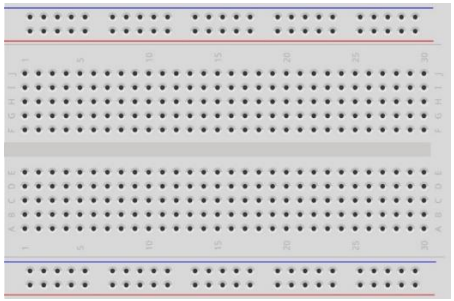
Component List

Control board x1

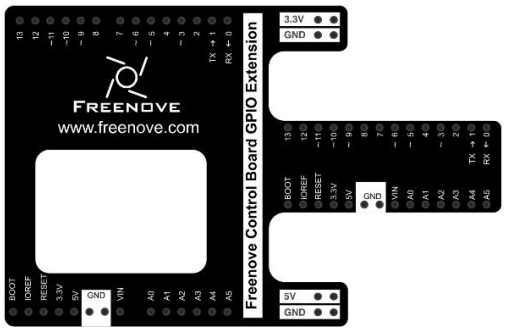


or

Breadboard x1



GPIO Extension Board x1



USB cable x1 	NPN transistor x1 	Speaker or Passive buzzer x1  or	Resistor 1kΩ x1 
Jumper M/M x6 			

If you don't have a speaker in your kit, use a passive buzzer instead.

Component Knowledge

DAC

Digital-to-Analog Converters (DACs) are capable of converting digital data into analog signals, making them suitable for a variety of applications. DACs, as the name suggests, are modules that convert digital codes into corresponding analog voltage outputs, functioning in a manner that is opposite to that of ADCs (Analog-to-Digital Converters).

Analog Output vs PWM

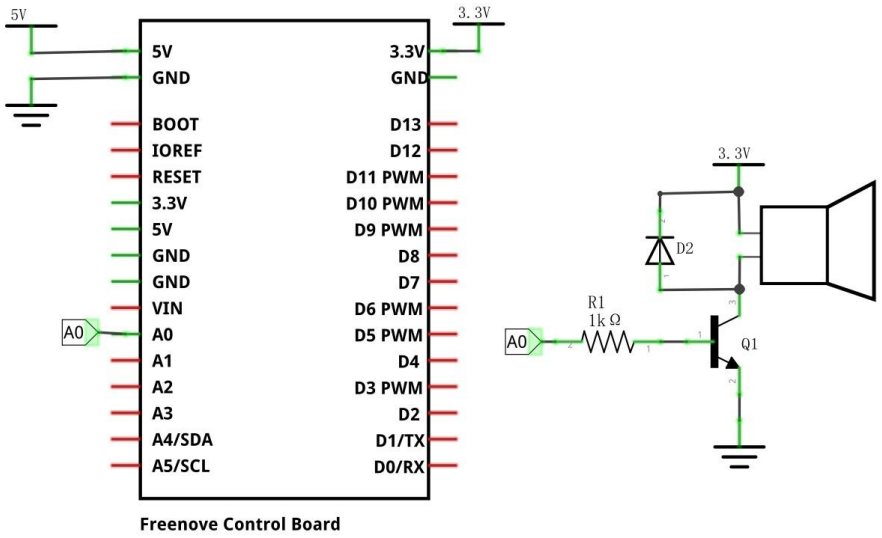
For many use cases when analog output is required, using PWM (Pulse Width Modulation) instead of genuine analog output will yield essentially the same results. A digital output pin can only either be fully on (HIGH) or fully off (LOW), but by turning it on and off very quickly with precise timings, the average voltage can be controlled and emulate an analog output. This method is called PWM.

For example when dimming an LED, you can freely use a PWM enabled digital pin as an analog output pin and the LED would dim just the same as if you'd be using a DAC output.

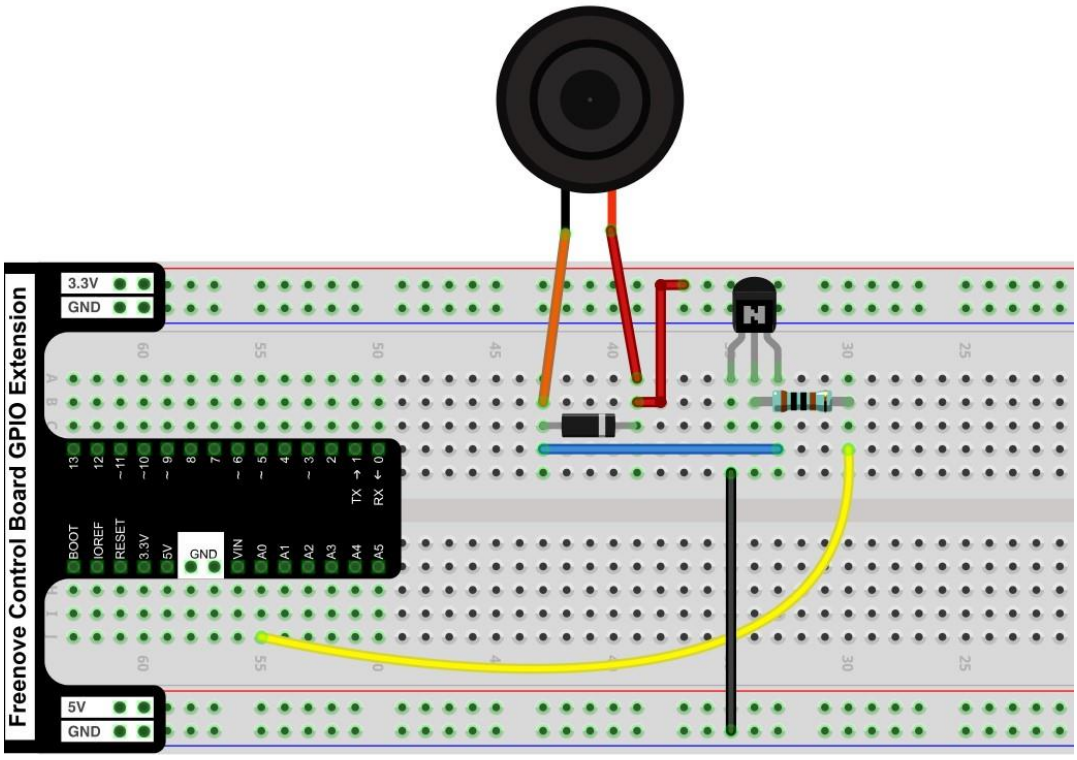
However this will not always be the case, and for many uses you will need to use a genuine analog output to get your desired results. One such case is in audio purposes, where a PWM output simply will not give the same quality of sound as a genuine analog output, and requires some fiddling to work in the first place.

Circuit

Schematic diagram



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



In addition, you can also refer to the above circuit to replace the speaker with a passive buzzer, the experimental effect is basically the same.

Sketch

Sketch 11.1.1

Upload the sketch to the board and you will hear the loudspeaker make sounds.

```
#include "analogWave.h"
#include "pitches.h"

// Melody and rhythm defined as an array
// Each note is followed by its duration (4 = quarter note, 8 = eighth note, etc.)
int melody[] = {
  // Super Mario Bros theme
  // Score available at https://musescore.com/user/2123/scores/2145
  // Theme by Koji Kondo
  NOTE_E5,8, NOTE_E5,8, REST,8, NOTE_E5,8, REST,8, NOTE_C5,8, NOTE_E5,8, //1
  NOTE_G5,4, REST,4, NOTE_G4,8, REST,4,
  NOTE_C5,-4, NOTE_G4,8, REST,4, NOTE_E4,-4, // 3
  NOTE_A4,4, NOTE_B4,4, NOTE_AS4,8, NOTE_A4,4,
  NOTE_G4,-8, NOTE_E5,-8, NOTE_G5,-8, NOTE_A5,4, NOTE_F5,8, NOTE_G5,8,
  REST,8, NOTE_E5,4,NOTE_C5,8, NOTE_D5,8, NOTE_B4,-4,
};

analogWave wave(DAC);  // Initialize the analogWave object for DAC(A0)

int noteCounter = 0;                  // Index to keep track of which note is being played
int tempo = 200;                      // Set the tempo in BPM (Beats Per Minute)
int wholenote = (60000 * 4) / tempo;  // Calculate the duration of a whole note in ms
int divider = 0, noteDuration = 0;    // Variables to hold note duration

void setup() {
  wave.sine(10);  // Initialize the sine wave generator with a frequency of 10 Hz
}

void loop() {
  // Calculate the duration of the current note
  divider = melody[noteCounter + 1];
  if (divider > 0) {
    // For regular notes
    noteDuration = wholenote / divider;
  } else if (divider < 0) {
    // For dotted notes (duration increased by 50%)
    noteDuration = wholenote / abs(divider);
    noteDuration *= 1.5;  // Increase the duration by 50% for dotted notes
  }
```

```
40
41    // Play the note
42    wave.freq(melody[noteCounter]);
43    delay(noteDuration * 0.85);  // Play the note for 85% of its duration
44    wave.stop();
45
46    // Pause between notes
47    delay(noteDuration * 0.15);  // Pause for 15% of the note duration
48
49    // Increment the note counter by 2 (because each note is followed by its duration)
50    noteCounter += 2;
51
52    // Reset the counter when reaching the end of the melody
53    int totalNotes = sizeof(melody) / sizeof(melody[0]);
54    noteCounter = noteCounter % totalNotes;
55
56    // Add a delay between repetitions of the melody
57    if (noteCounter == 0) {
58      delay(1000);
59    }
60  }
```

If you want to know more about the DAC Settings of the control panel, please refer to the connection: https://docs.arduino.cc/tutorials/uno-r4-minima/dac/

You can also use the above circuit to play an interesting audio clip by uploading Sketch_10.1.2_DAC_Jacques.ino to your development board. If you don't have a speaker, you can replace it with a passive buzzer.

Chapter 12 RTC

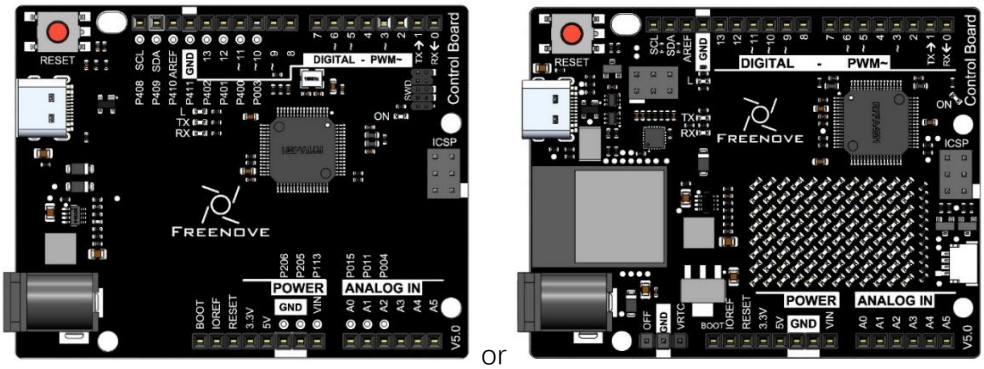
In this chapter, we explore how to access the Real-Time Clock (RTC) on the Control board (V5). The RTC is embedded in the microcontroller (RA4M1) of the Control board (V5).

Project 12.1 RTC

With this project, let's delve into controlling the RTC function of the control board, applying it to flash the indicator every two seconds.

Component List

Control board x1



USB cable x1

Component Knowledge

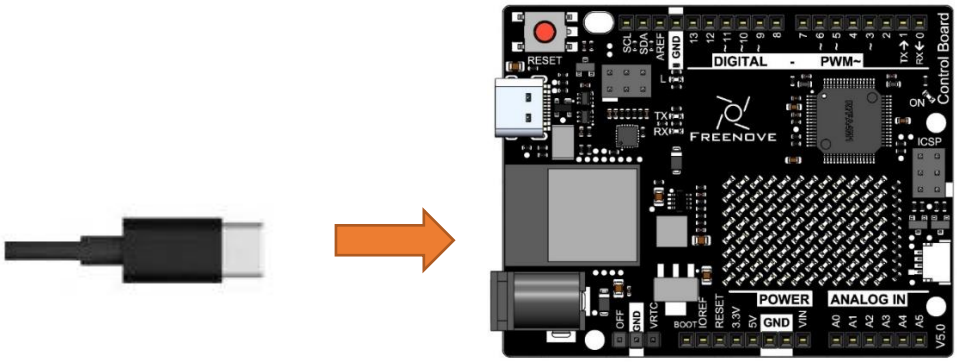
RTC

The Real-Time Clock (RTC) is integrated into the microcontroller (RA4M1) of the control board, serving as an independent timekeeping devices that continues to operate even when the primary source of power is off or unavailable. This is achieved thanks to a backup power source, such as a battery. Such feature significantly enhances the RTC's utility across a spectrum of applications, from automating tasks in home systems to timestamping data entries in logging applications.

It should be highlighted that the control board (WiFi) is equipped with a VRTC pin that sustains the RTC's functionality when the board is not powered. To engage this functionality, a voltage between 1.6 and 3.6 V must be supplied to the VRTC pin. It is crucial to stay within this voltage range to avoid damaging the board. It is also important to recognize that this advanced feature is exclusive to the control board with WiFi function and is not available in the standard models.

Circuit

Connect the board to the computer using the USB cable.



Sketch

Sketch 12.1.1

Upload the sketch to the board and you will see the onboard LED flash every two seconds.

```
#include "RTC.h"

volatile bool irqFlag = false;  // Flag to indicate a periodic callback.
bool ledState = false;          // Current LED state: true for ON and false for OFF
const int led = LED_BUILTIN;    // Pin number for the built-in LED

void setup() {
  pinMode(led, OUTPUT);  // Configure the LED pin as output

  Serial.begin(9600);

  // Initialize the RTC
  RTC.begin();
  // Set an arbitrary time to the RTC (required for RTC.setPeriodicCallback to work)
  RTCTime mytime(13, Month::MAY, 2024, 05, 13, 00, DayOfWeek::MONDAY,
SaveLight::SAVING_TIME_ACTIVE);
  RTC.setTime(mytime);

  // Set the periodic callback to trigger every 2 seconds
```

```
20    if (!RTC.setPeriodicCallback(periodicCallback, Period::ONCE_EVERY_2_SEC)) {
21      Serial.println("ERROR: periodic callback not set");
22    }
23  }
24
25  void loop() {
26    // Check if the periodic callback has been triggered
27    if (irqFlag) {
28      Serial.println("Timed CallBack");  // Output a debugging message
29      ledState = !ledState;              // Toggle the LED state
30      digitalWrite(led, ledState);       // Update the LED
31      irqFlag = false;                   // Reset the flag
32    }
33  }
34
35  // This is the callback function to be passed to RTC.setPeriodicCallback()
36  void periodicCallback() {
37    // Set the flag to true to indicate the callback has been triggered
38    irqFlag = true;
39  }
```

Initialize RTC.

```
13    RTC.begin();
```

Configure the time for RTC.

```
15    RTCTime mytime(13, Month::MAY, 2024, 05, 13, 00, DayOfWeek::MONDAY,
16  SaveLight::SAVING_TIME_ACTIVE);
17    RTC.setTime(mytime);
```

A periodic interrupt allows you to set a recurring callback. Set the periodcallback to triger once every two seconds.

```
20    if (!RTC.setPeriodicCallback(periodicCallback, Period::ONCE_EVERY_2_SEC)) {
21      Serial.println("ERROR: periodic callback not set");
22    }
```

Change the status of the LED every two seconds.

```
27    if (irqFlag) {
28      Serial.println("Timed CallBack");  // Output a debugging message
29      ledState = !ledState;              // Toggle the LED state
30      digitalWrite(led, ledState);       // Update the LED
31      irqFlag = false;                   // Reset the flag
32    }
```

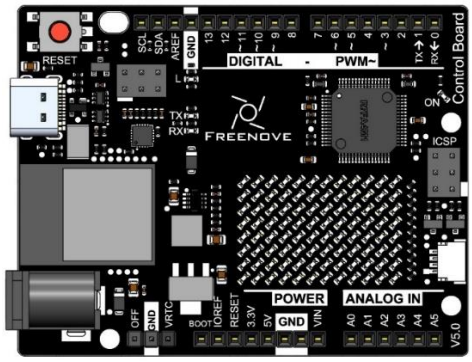
Chapter 13 Onboard LED Matrix (WiFi Board)

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 13.1 LED Matrix

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

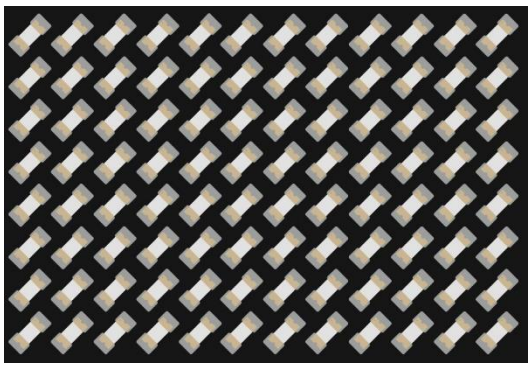
Component List

Control board x1 	USB cable x1 

Component Knowledge

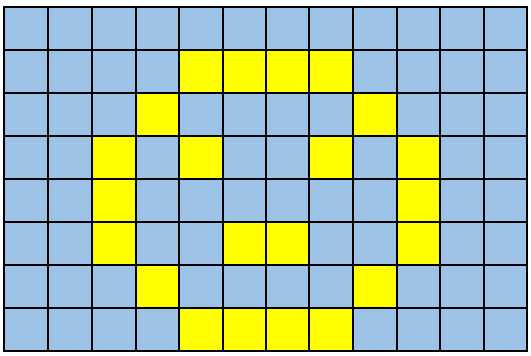
LED matrix

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

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 }};
```

The setup method mentioned above is easy to understand because you can visualize the image in the array pattern, and it is straightforward to edit while running. The elements in the array above form a smiley face, which is the image you see on the screen.

If you need to locate a single pixel, select its address and change the value. Remember to start counting from 0. Thus, the following lines will address the third pixel from the left and the second pixel from the top, and then activate it:

```
frame[2][1] = 1;


matrix.renderBitmap(frame, 8, 12);
```

Howevere, this method consumes more memory than required. Even though each LED only needs one bit to store its state, 8 bits (one byte) are used. There is a more memory-efficient approach for storing frames that employs an array of 32-bit integers.

Unsigned long variables store 32 bits, and dividing 96 by 32 gives us 3, indicating that 96 LEDs can be split into three segments. Thus, an array of unsigned long integers is an efficient way to store all the bits necessary for the LED matrix.

The following is the data of converting the binary values of each row of the LED matrix to hexadecimal from left to right:

Column	Binary	Hexadecimal
1	0000 0000 0000	0x000
2	0000 1111 0000	0x0F0
3	0001 0000 1000	0x108
4	0010 1001 0100	0x294
5	0010 0000 0100	0x204
6	0010 0110 0100	0x264
7	0001 0000 1000	0x108
8	0000 1111 0000	0x0F0

The above 8 sets of data are converted into 3 groups as follows:

```
0000 0000 0000 0000 1111 0000 0001 0000

1000 0010 1001 0100 0010 1001 0100 0010

0110 0100 0001 0000 1000 0000 1111 0000
```

```
0x0000F010

0x82942042

0x641080F0
```

From this we can conclude that the 32-bit integer array of a smiley face is as follows.

```
unsigned long frame[] = {
  0x0000F010,
  0x82942042,
  0x641080F0
};
```

Uploading the above data to the control board will display the corresponding pattern. Additionally, if you have multiple different frames, you can load and display them as below:

```
const uint32_t happy[] = {
    0x19819,
    0x80000001,
    0x81f8000
};
const uint32_t heart[] = {
     0x3184a444,
    0x44042081,
    0x100a0040
};
  matrix.loadFrame(happy);
  delay(500);

```

```
14    matrix.loadFrame(heart);
15    delay(500);
```

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

Sketch

Sketch 13.1.1

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:

```
1   #include "Arduino_LED_Matrix.h"  // Include the LED_Matrix library
2
3   ArduinoLEDMatrix matrix;  // Create an instance of the ArduinoLEDMatrix class
4
5   byte frame[8][12] = {
6     { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
7     { 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 },
8     { 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
9     { 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0 },
10    { 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0 },
11    { 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0 },
12    { 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
13    { 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 }
14  };
151
16  unsigned long frame2[] = {
17    0x0000F010,
18    0x82942042,
19    0x641080F0
20  };
21
22  const uint32_t full0n[] = {
23    0xffffffff,
24    0xffffffff,
25    0xffffffff
26  };
27  const uint32_t full0ff[] = {
28    0x00000000,
29    0x00000000,
30    0x00000000
```

```
31  };
32
33  void setup() {
34    matrix.begin();  // Initialize the LED matrix
35    matrix.loadFrame(full1On);
36    delay(250);
37    matrix.loadFrame(full1Off);
38    delay(250);
39  }
40
41  void loop() {
42    //matrix.loadFrame(frame2);
43    matrix.renderBitmap(frame, 8, 12);
44    delay(250);
45  }
```

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.

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

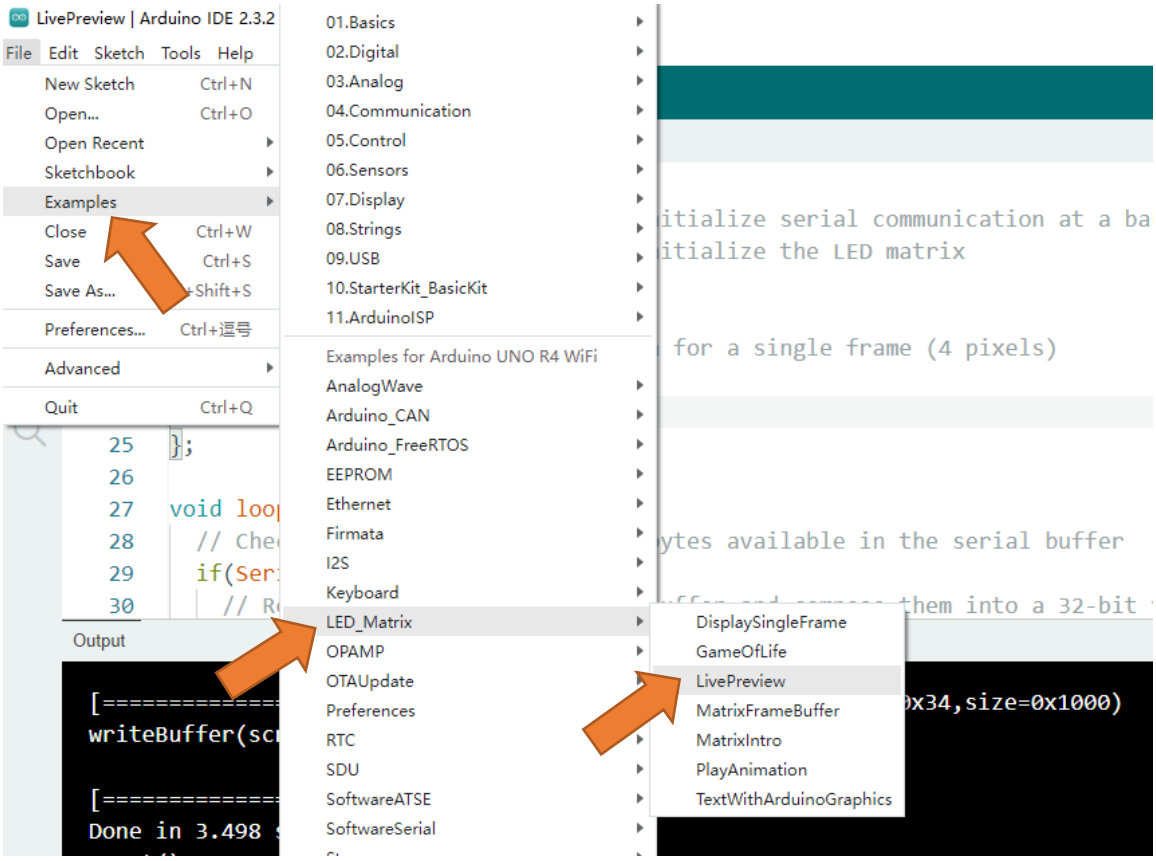
After the program initializes, it first lights up the entire LED matrix and then turns off the LED matrix, and subsequently continuously displays the smiling face graphic.

```
34    matrix.begin();  // Initialize the LED matrix
35    matrix.loadFrame(full1On);
36    delay(250);
37    matrix.loadFrame(full1Off);
38    delay(250);
...   ...
43    matrix.renderBitmap(frame, 8, 12);
44    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)

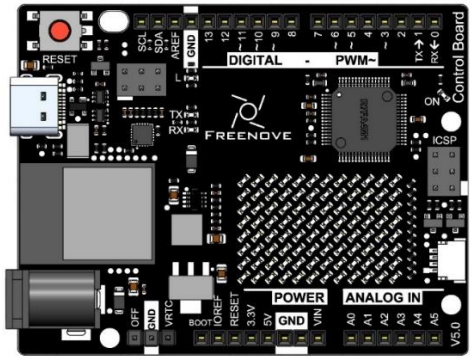
When using the above tools to customize the graphics, the following examples should be uploaded to the control board first. After uploading, the corresponding device can be connected to the real-time transmission of the graphics to the control board for display.



Project 13.2 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 13.1.1

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();

```

```
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();
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

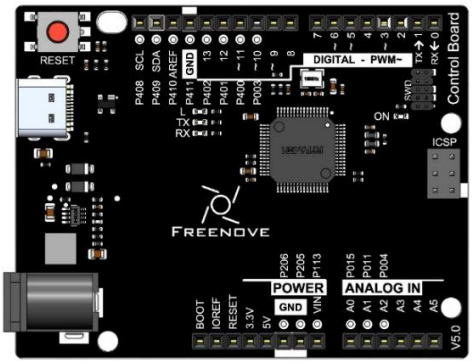
Project 13.3 Play the game with LED matrix

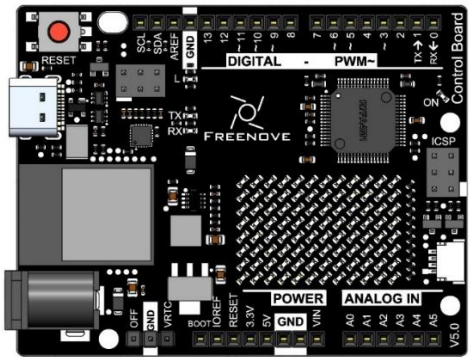
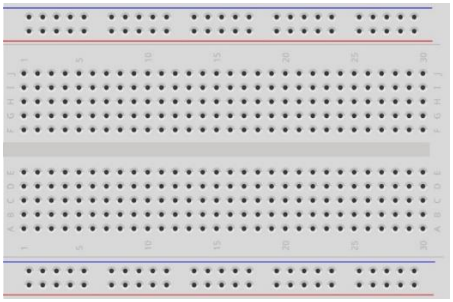
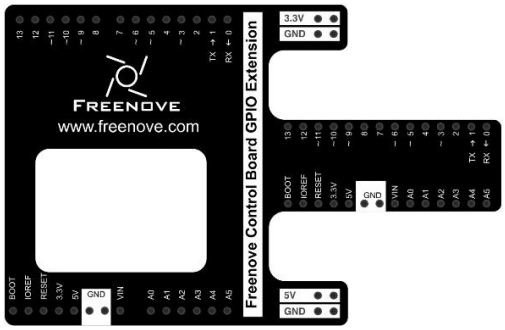
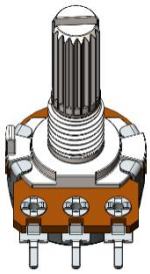
In this section, we play the game using the LED matrix.

Project 13.3.1 LED Matrix Bounce Game

Play a bounce game with an LED matrix.

Component List

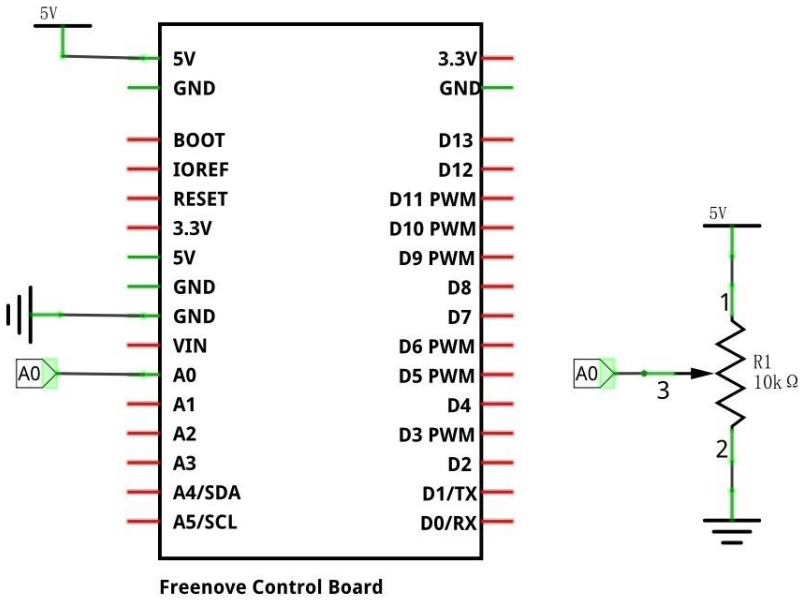
<table>
<tr><td colspan="2">Control board x1
 

or

</td></tr>
<tr><td>Breadboard x1
  </td><td>GPIO Extension Board x1
 
</td></tr>
<tr><td>USB cable x1
  </td><td rowspan="2">Rotary potentiometer x1
  </td></tr>
<tr><td>Jumper M/M x3
</td></tr>
</table>

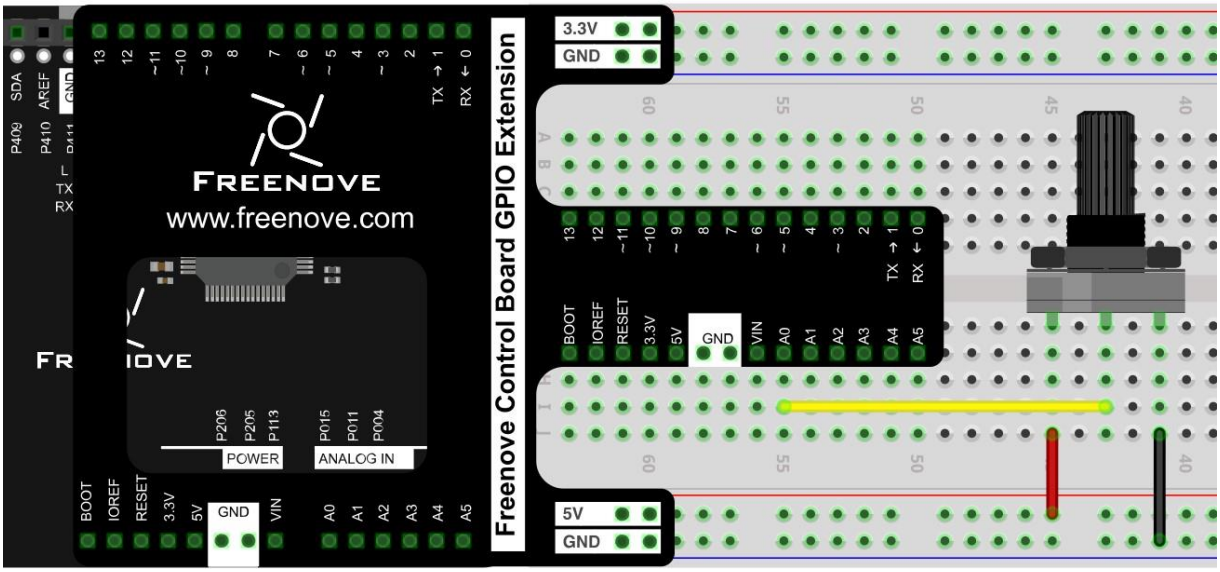
Circuit

The voltage of the rotary potentiometer is detected with the A0 pin on the control board to control the movement of the bat.

Schematic diagram



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



Sketch

Sketch 13.3.1

This is a simple ball bounce game, using a potentiometer to control the movement of the bat, when your bat fails to catch the ball, the game will end and the led matrix will show "OUT", and then start the game again.

The following is the program code:

```
#include "Arduino_LED_Matrix.h"

ArduinoLEDMatrix matrix;

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

uint8_t frame[8][12] = {
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};

uint8_t game_over[8][12] = {
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1 },
  { 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0 },
  { 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0 },
  { 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};

int y = 4;            // starting y position
int x = 4;            // starting x position
int b = 1;            // starting y direction
int a = 1;            // starting x direction
```

```
int anticipate = 5;  // variable used to anticipate whether ball hits bat
int speed = 300;     // initial delay which decreases each time ball hit


void loop() {
  int batcenter = (analogRead(A0) / 204);
  frame[batcenter][11] = 1;
  frame[batcenter + 1][11] = 1;
  frame[batcenter + 2][11] = 1;
  frame[y][x] = 1;
  matrix.renderBitmap(frame, 8, 12);
  delay(speed);
  frame[batcenter][11] = 0;
  frame[batcenter + 1][11] = 0;
  frame[batcenter + 2][11] = 0;
  frame[y][x] = 0;
  matrix.renderBitmap(frame, 8, 12);
  delay(1);

  if (y == 7) {
    b = -b;
  }
  if (y == 0) {
    b = -b;
  }
  if (x == 10) { // when ball reaches line in front of batcenter, checks if batcenter hit
    anticipate = (y + b);
    speed = (speed - 20);  // reduces delay each time batcenter hit
    if ((anticipate == batcenter) or (anticipate == (batcenter + 1)) or (anticipate ==
(batcenter + 2))) {
      a = -a;
    } else {  // game over grid
      matrix.renderBitmap(game_over, 8, 12);
      delay(2000);
      x = 4;
      y = 4;
      a = -1;
      b = -1;
      speed = 300;
    }
  }
  if (x == 0) {
    a = -a;
  }
```

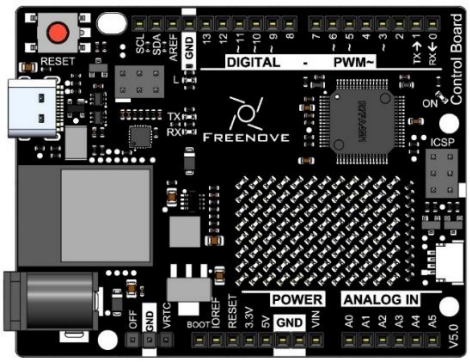
```
80      y = (y + b);  // adjusts ball position
81      x = (x + a);
82  }
83
```

Chapter 14 WiFi Working Modes (WiFi Board)

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 14.1 Station mode

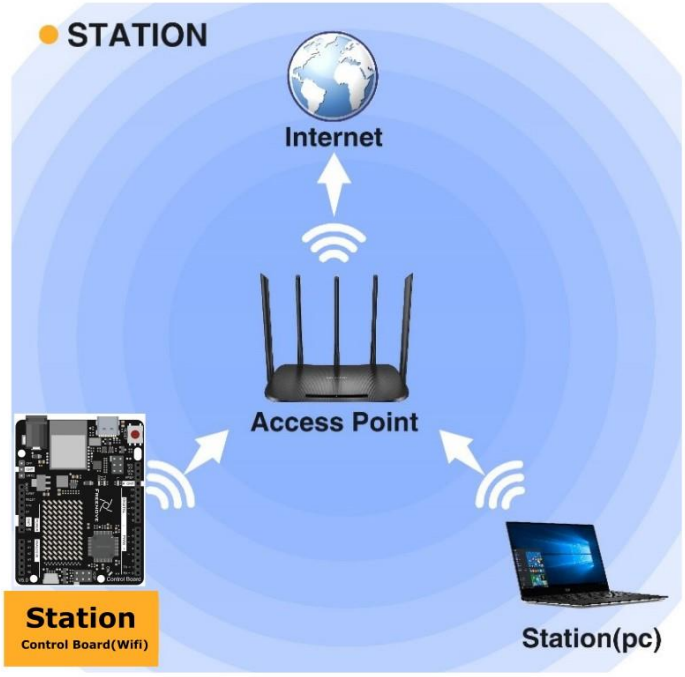
Component List

Control board x1 	USB cable x1 

Component knowledge

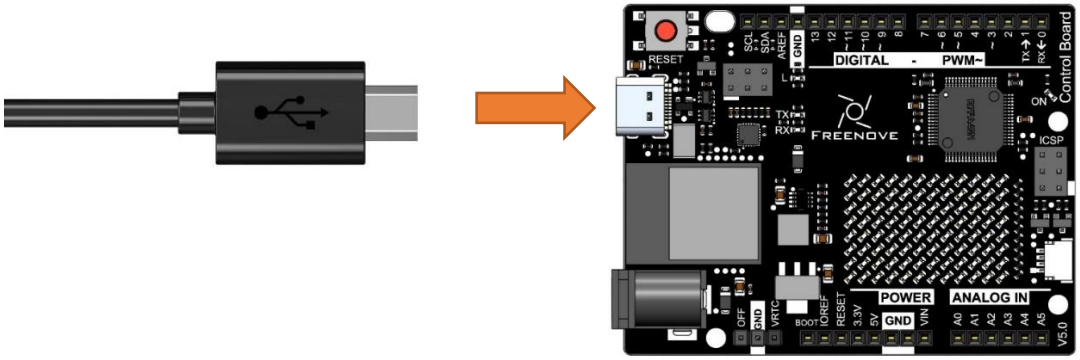
Station mode

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.



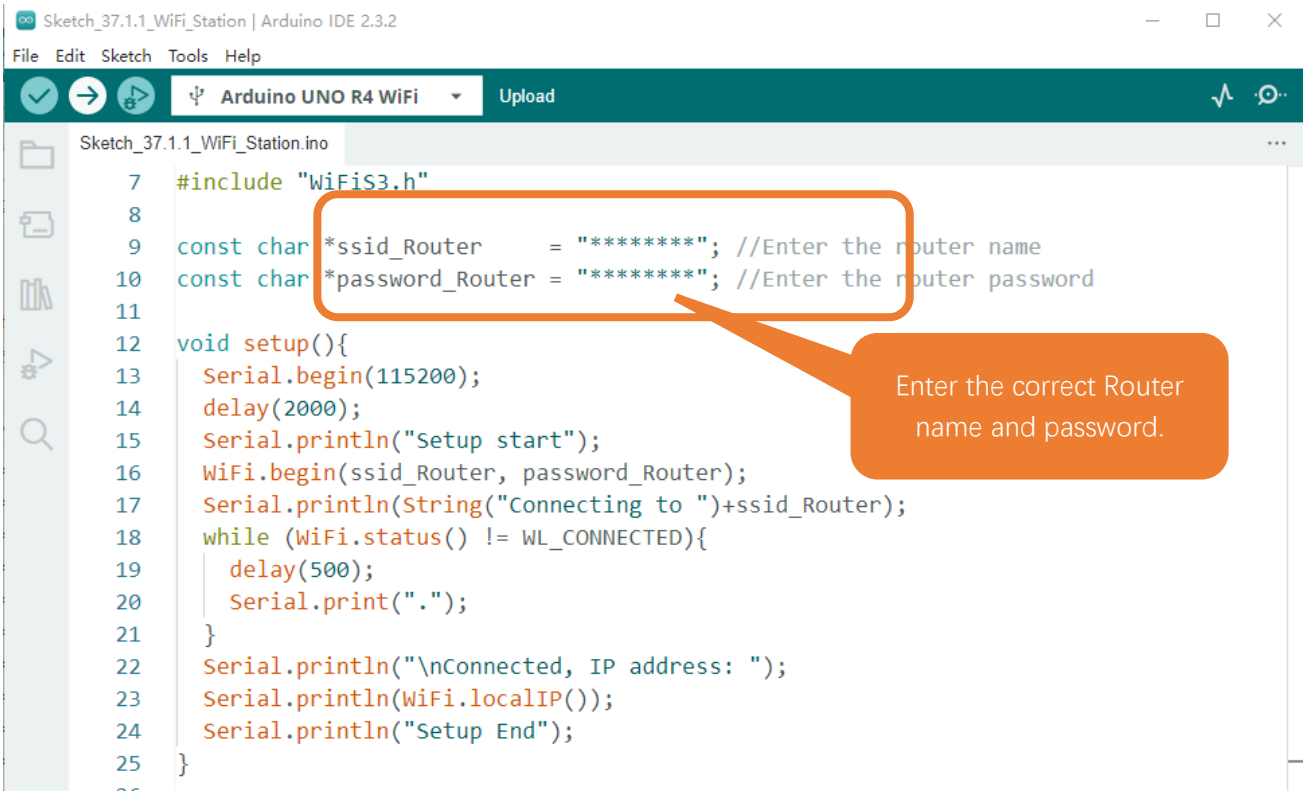
Circuit

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



Sketch

Sketch_14.1.1



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 14.2 AP mode

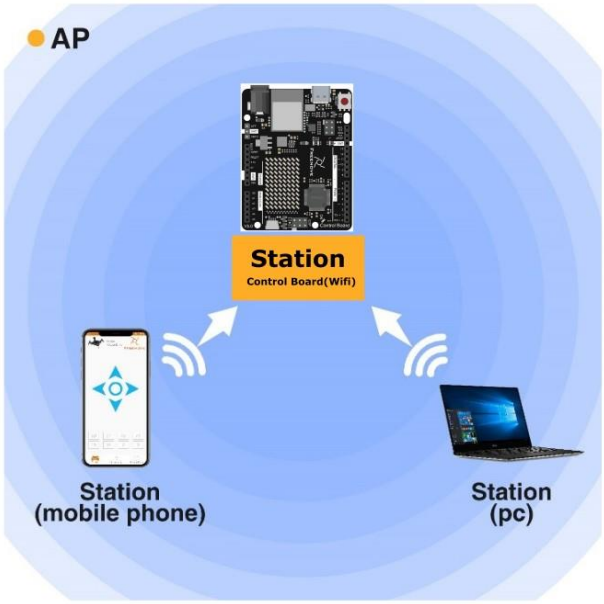
Component List & Circuit

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

Component knowledge

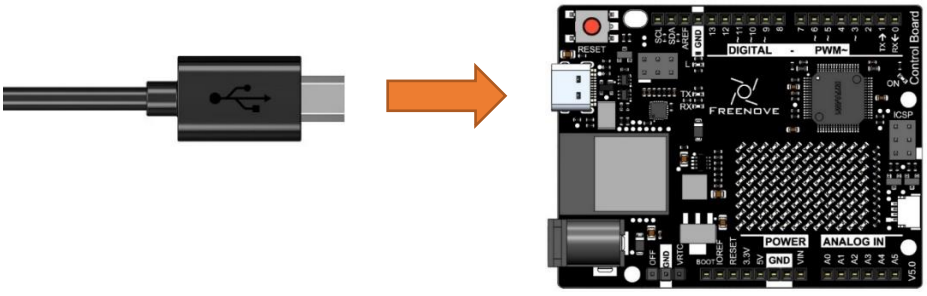
AP mode

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.

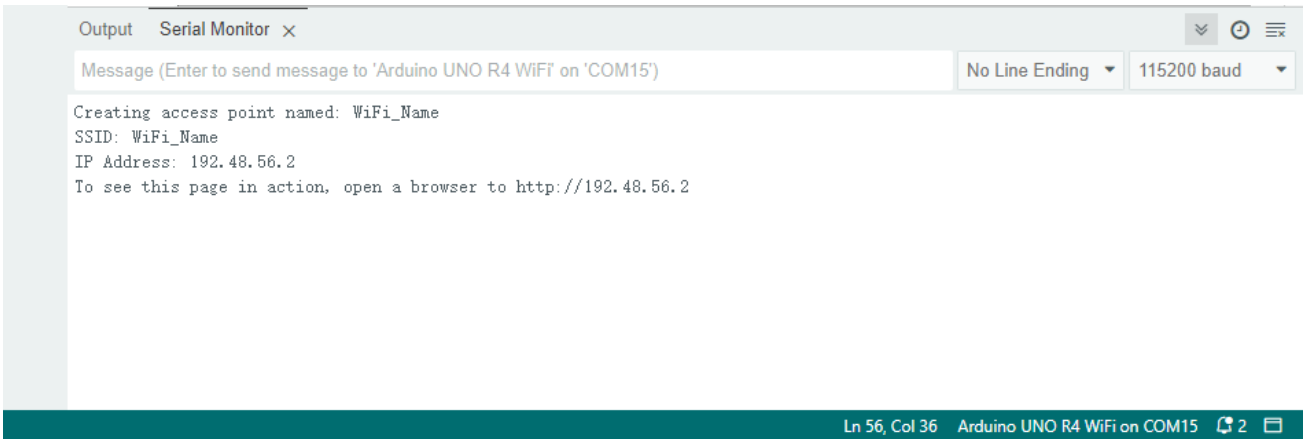


Sketch

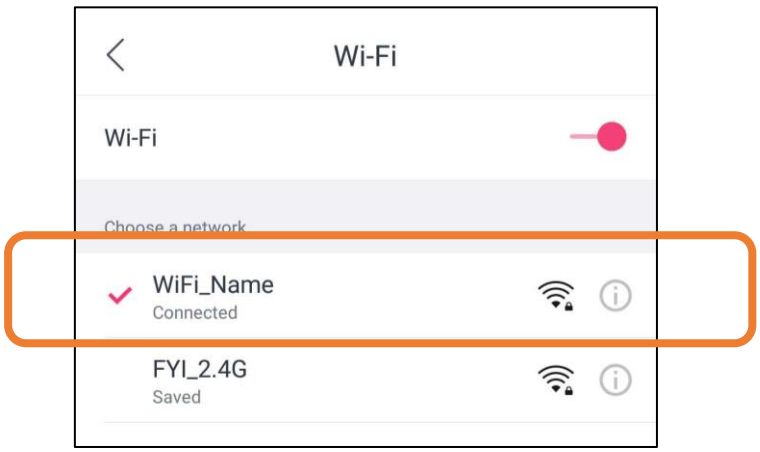
Sketch_14.2.1

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   }
```

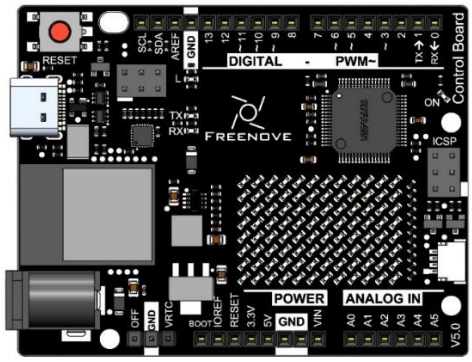
Chapter 15 TCP/IP (WiFi Board)

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 15.1 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

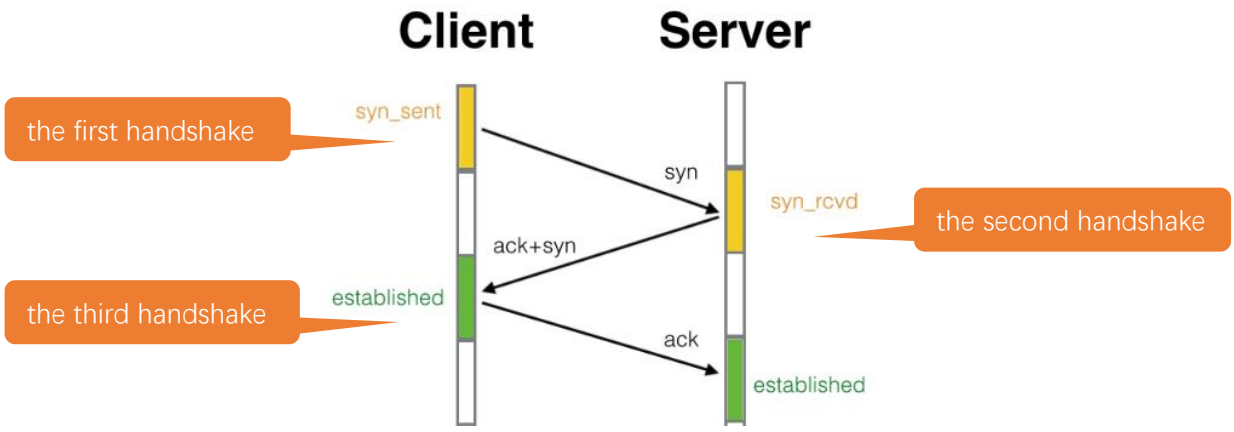
Control board x1 	USB cable x1 

Component knowledge

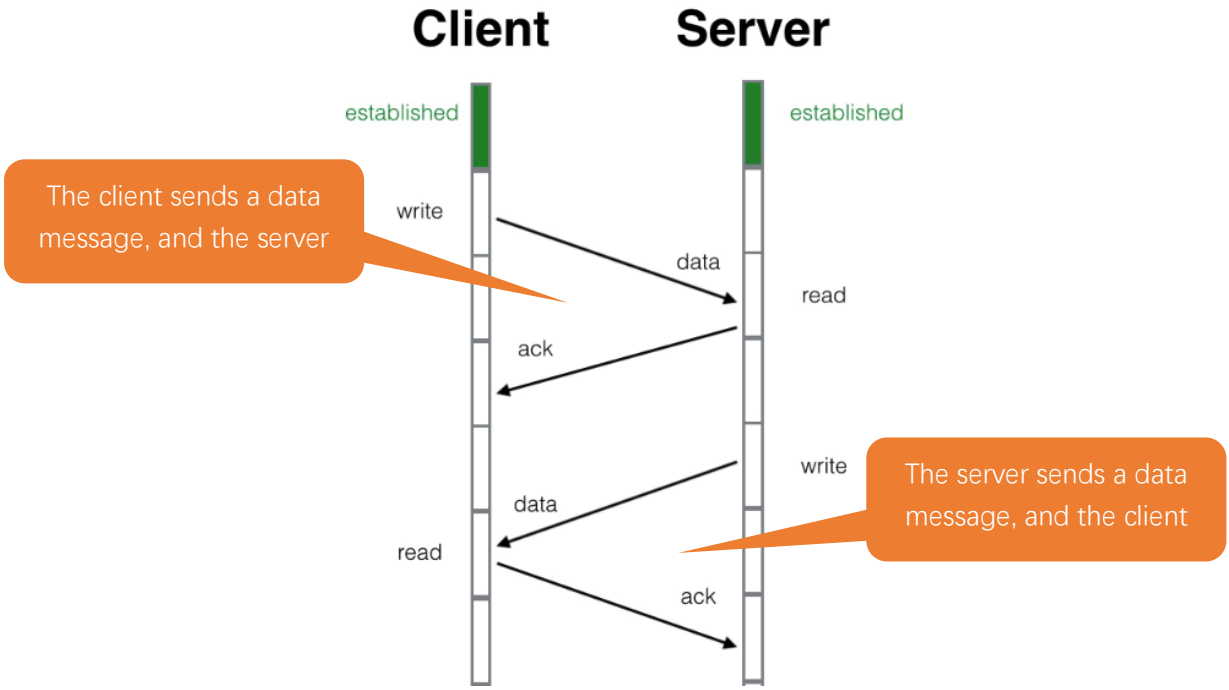
TCP connection

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

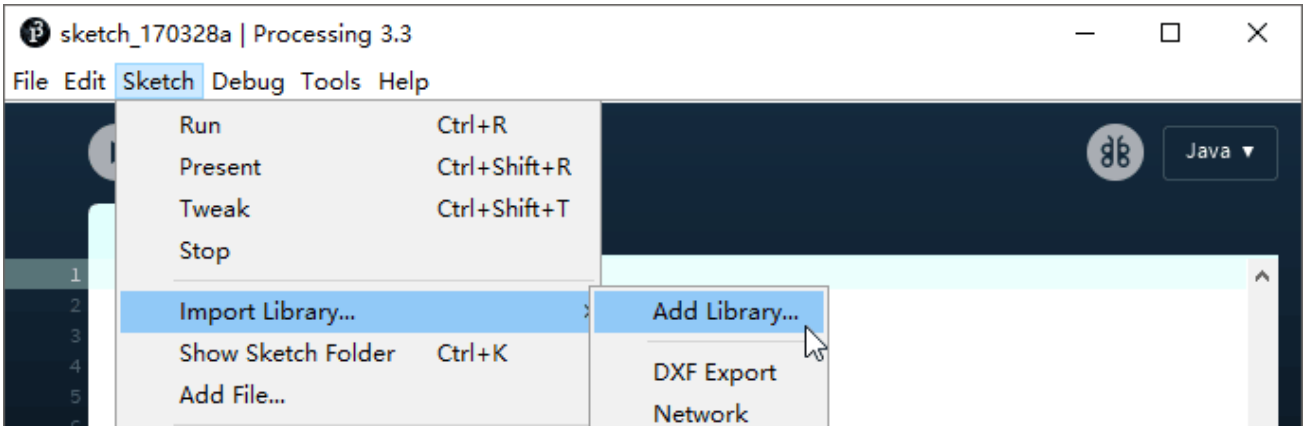
In this tutorial, we use Processing to build a simple TCP/IP communication platform.

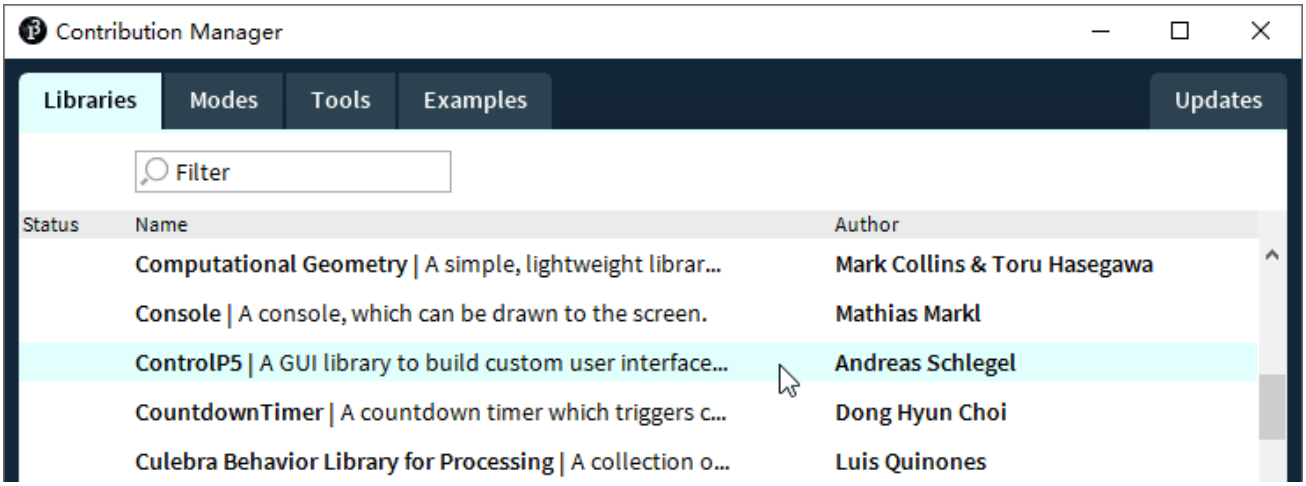
If you've not installed Processing, you can download it by clicking https://processing.org/download/. You can choose an appropriate version to download according to your PC system.

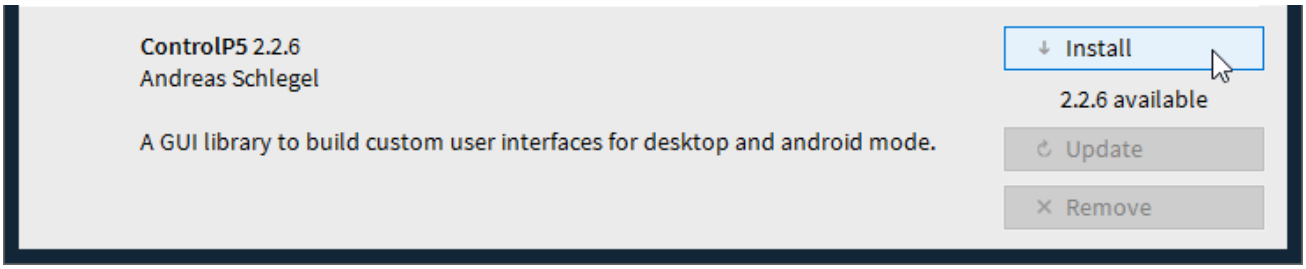
Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.

Use Server mode for communication

Install ControlP5.





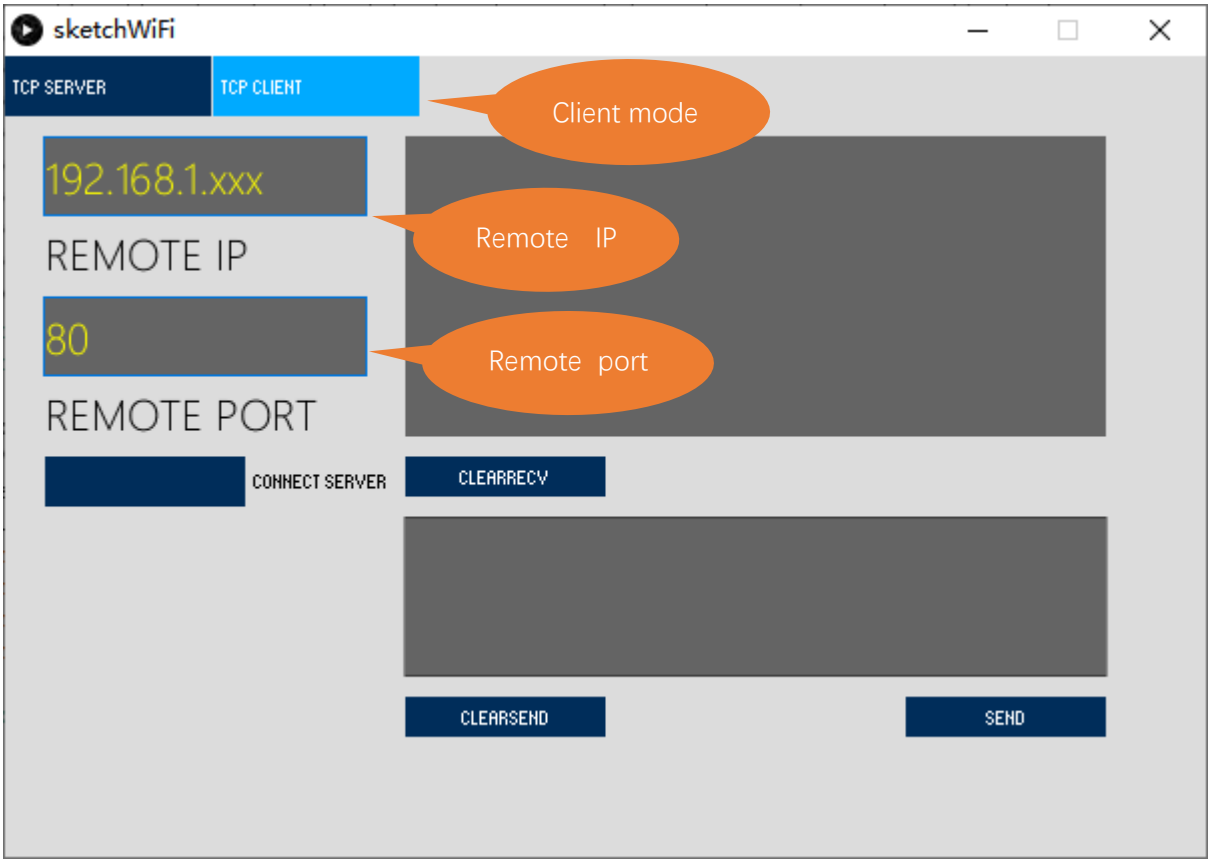


Open the "**Sketches\Sketches\Sketch_37.1_WiFiClient\sketchWiFi\sketchWiFi.pde**", and click "Run".

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.

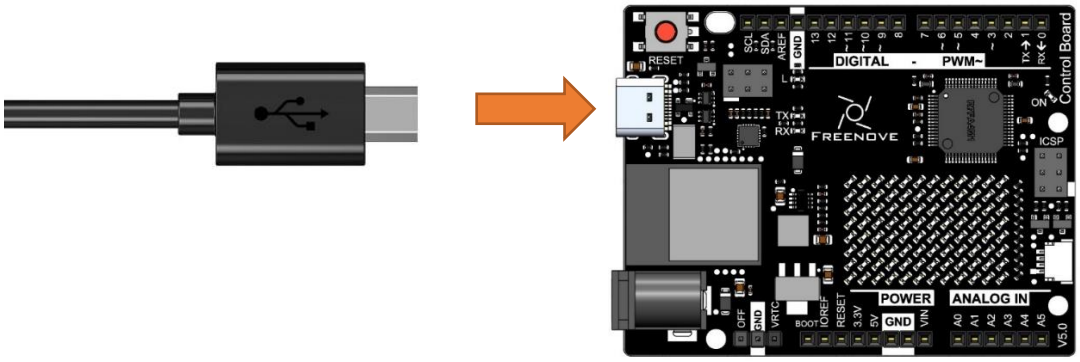
clear receive: clear out the content in the receiving text box

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.

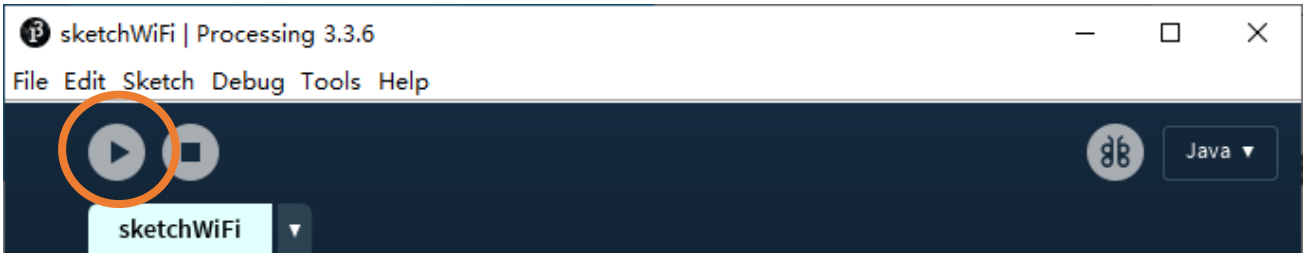
Circuit

Connect Freenove control board to the computer using USB cable.



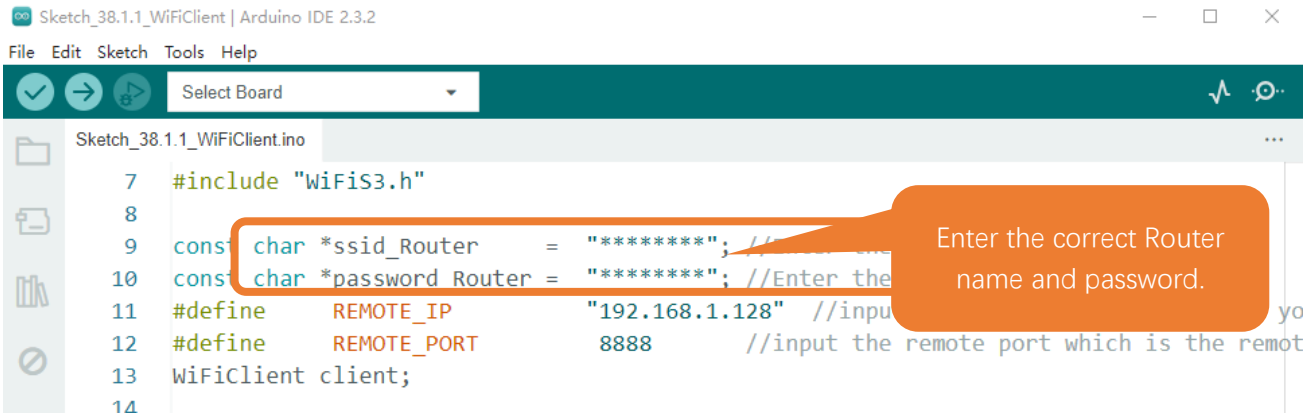
Sketch

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



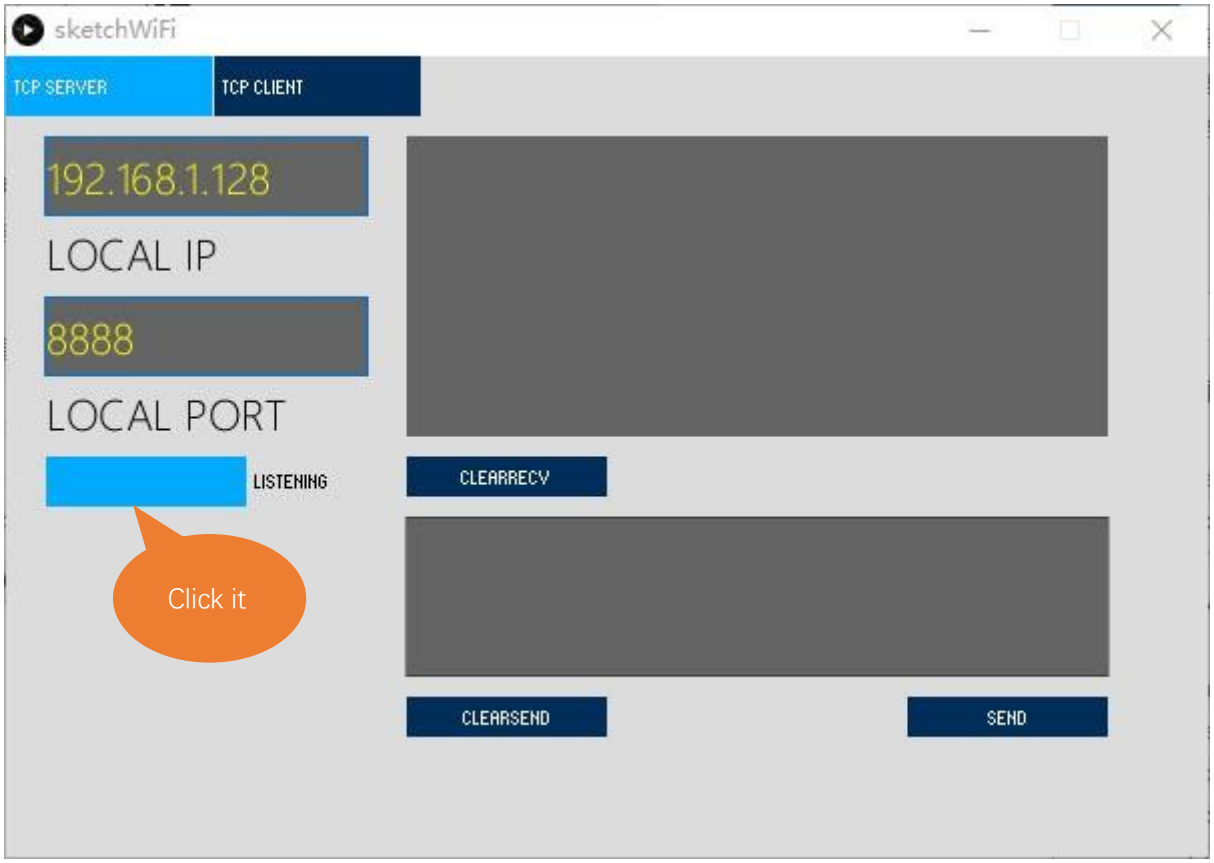
The newly pop up window will use the computer’s IP address by default and open a data monitor port.

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

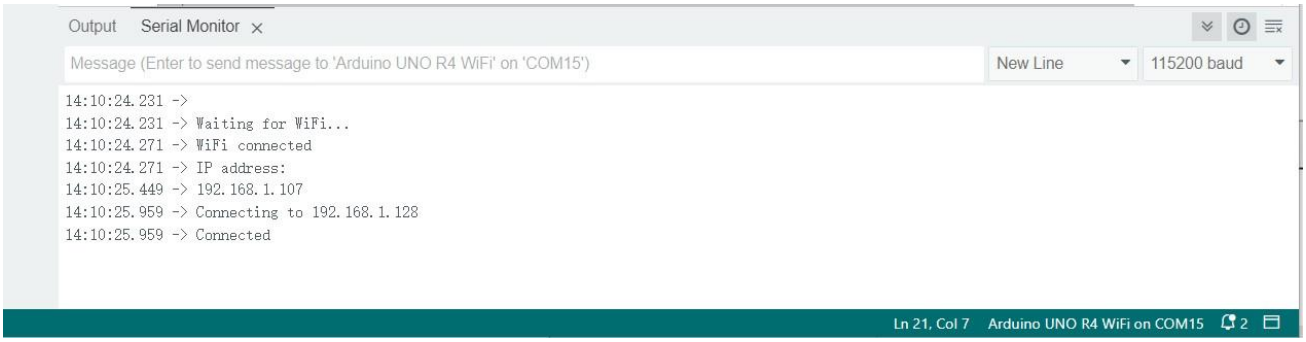


REMOTE_IP needs to be filled in according to the interface of sketchWiFi.pde. Taking this tutorial as an example, its REMOTE_IP is “192.168.1.128”. Generally, by default, the ports do not need to change its value.

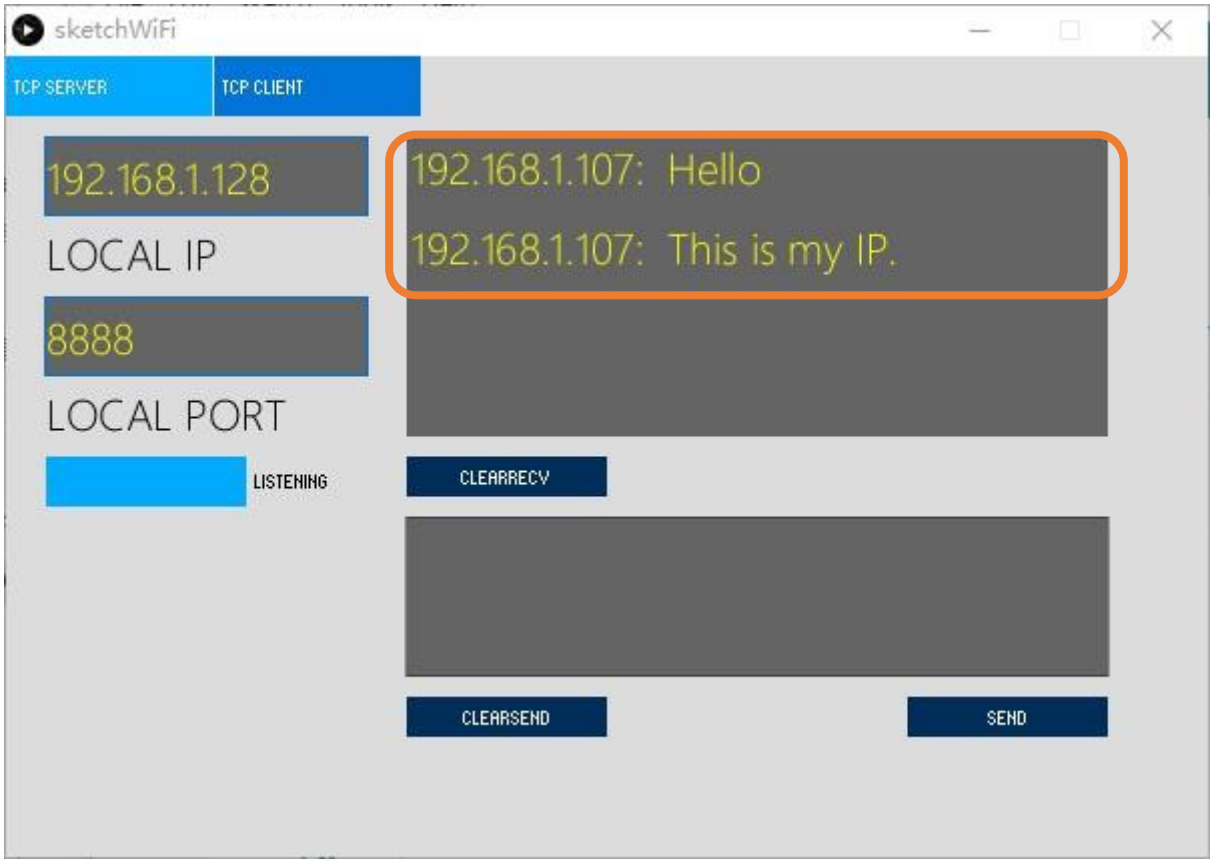
Click LISTENING, turn on TCP SERVER's data listening function and wait for control board(wifi) to connect.



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.



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.

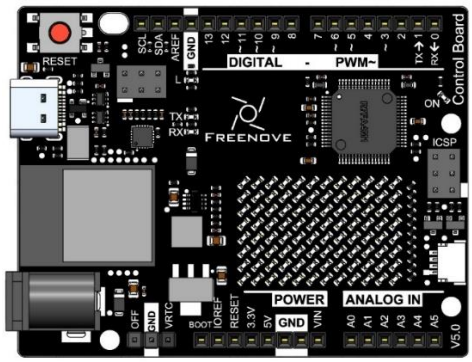
read(): read one byte of data in receive buffer

readString(): read string in receive buffer

Project 15.2 As Server

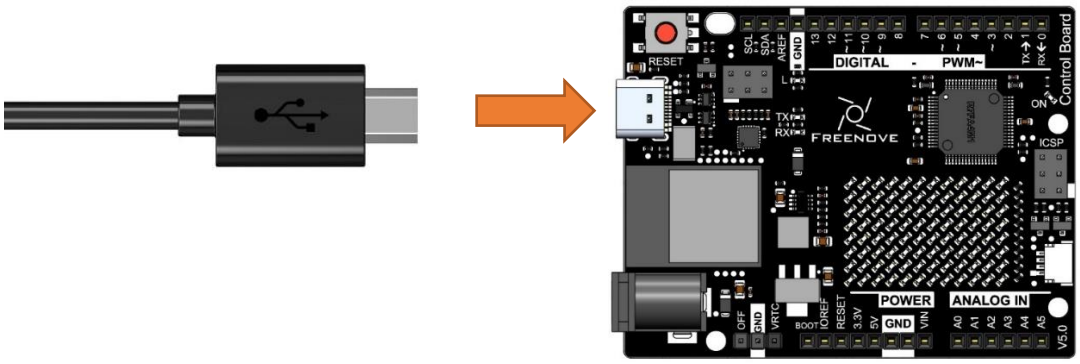
In this section, control board(wifi) is used as a server to wait for the connection and communication of client on the same LAN.

Component List

Control board x1 	USB cable x1 

Circuit

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

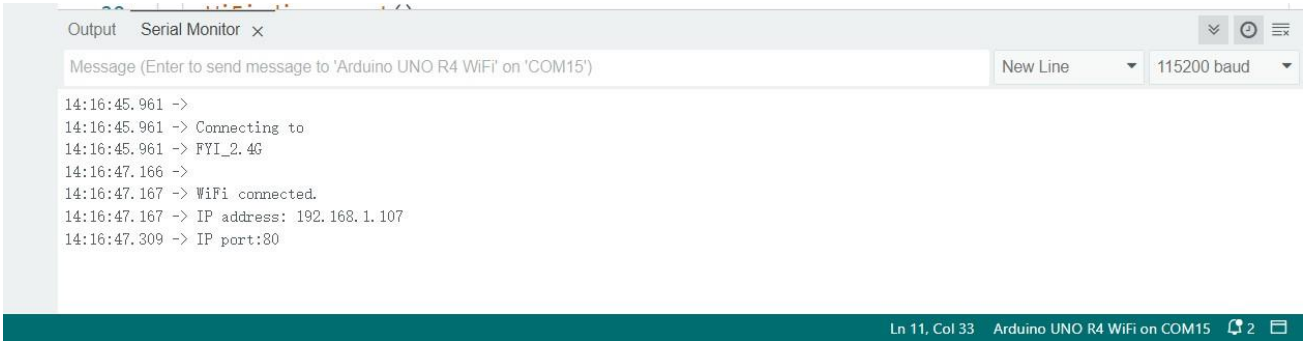


Sketch

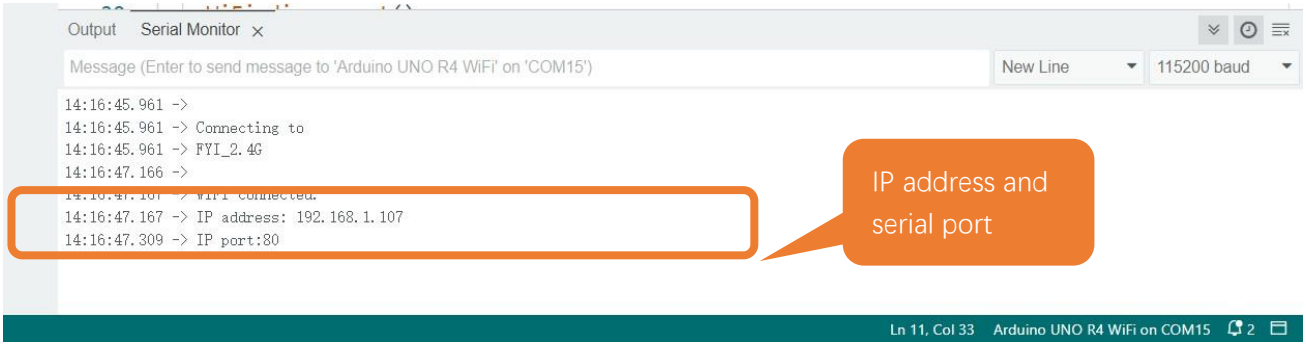
Before running Sketch, please modify the contents of the box below first.

Sketch_15.1.1

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.



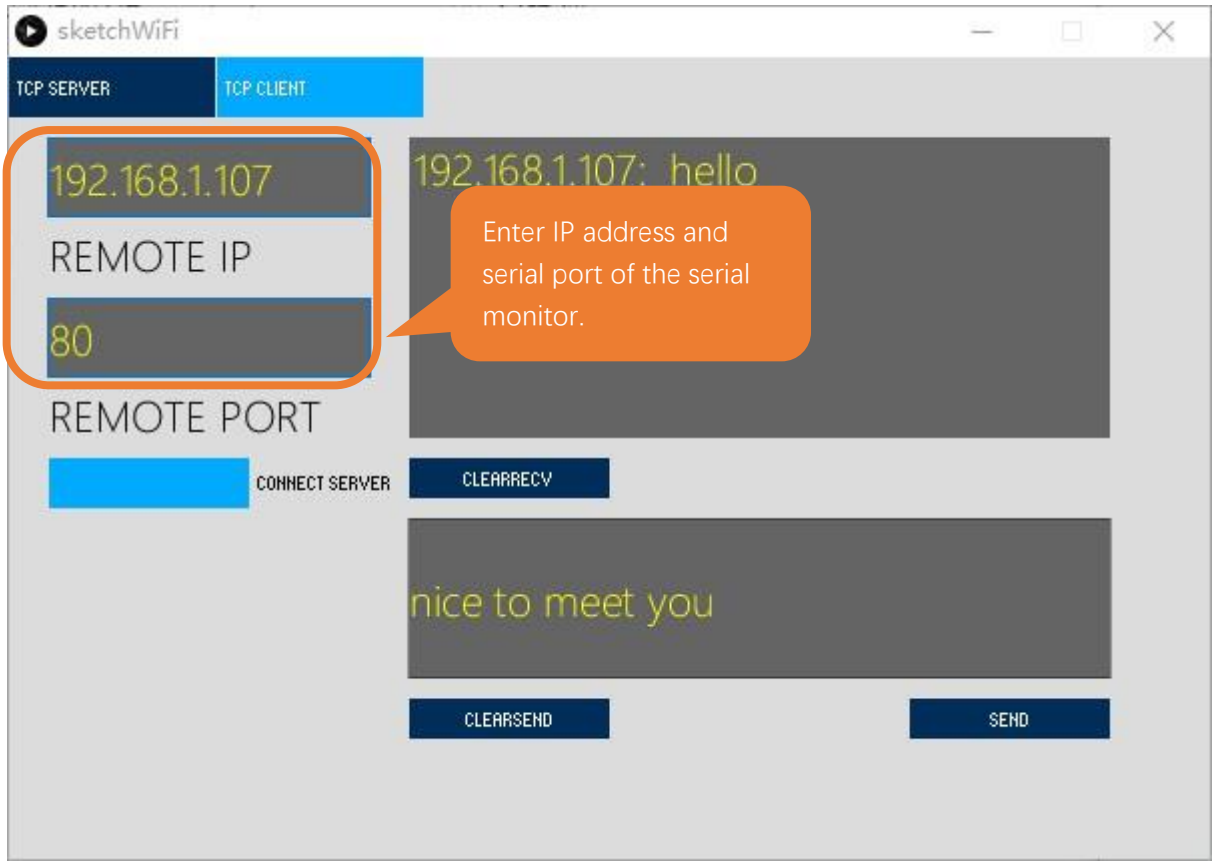
Serial Monitor



Processing:

Open the "**Sketches\Sketch_14.2.1_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.

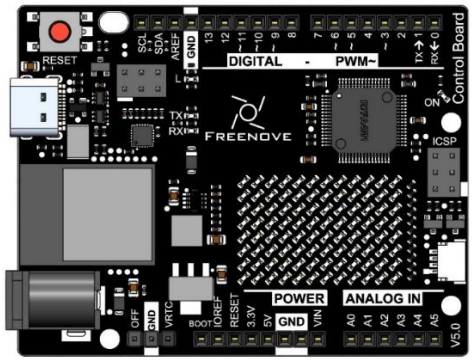
Chapter 16 Control LED with Web (WiFi Board)

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 16.1 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

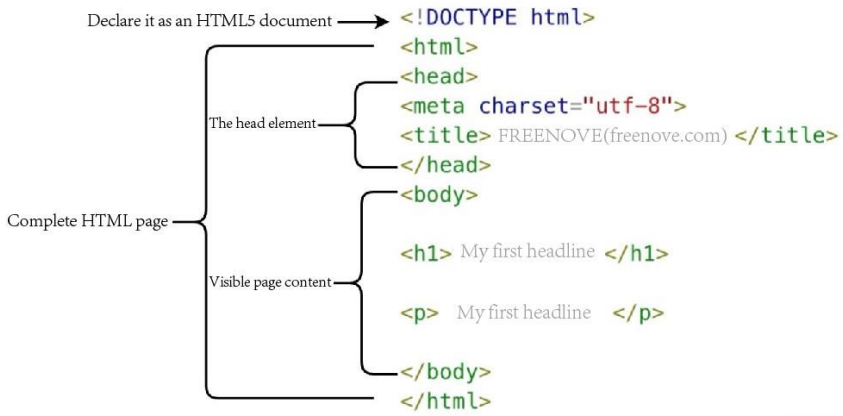
HTML

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

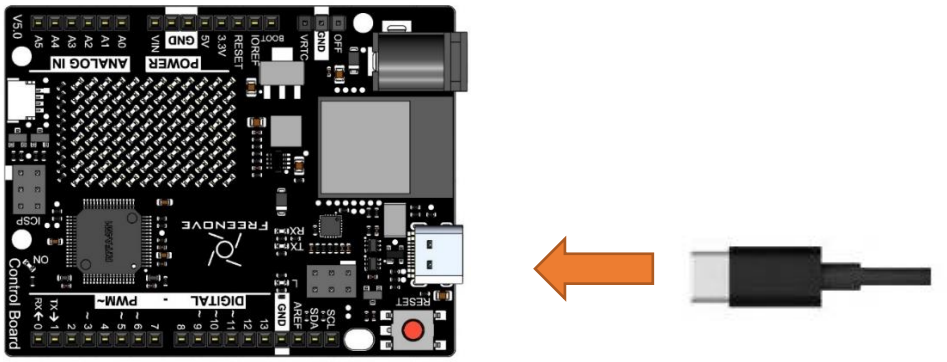
<h1>: Define a big heading

<p>: Define a paragraph

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

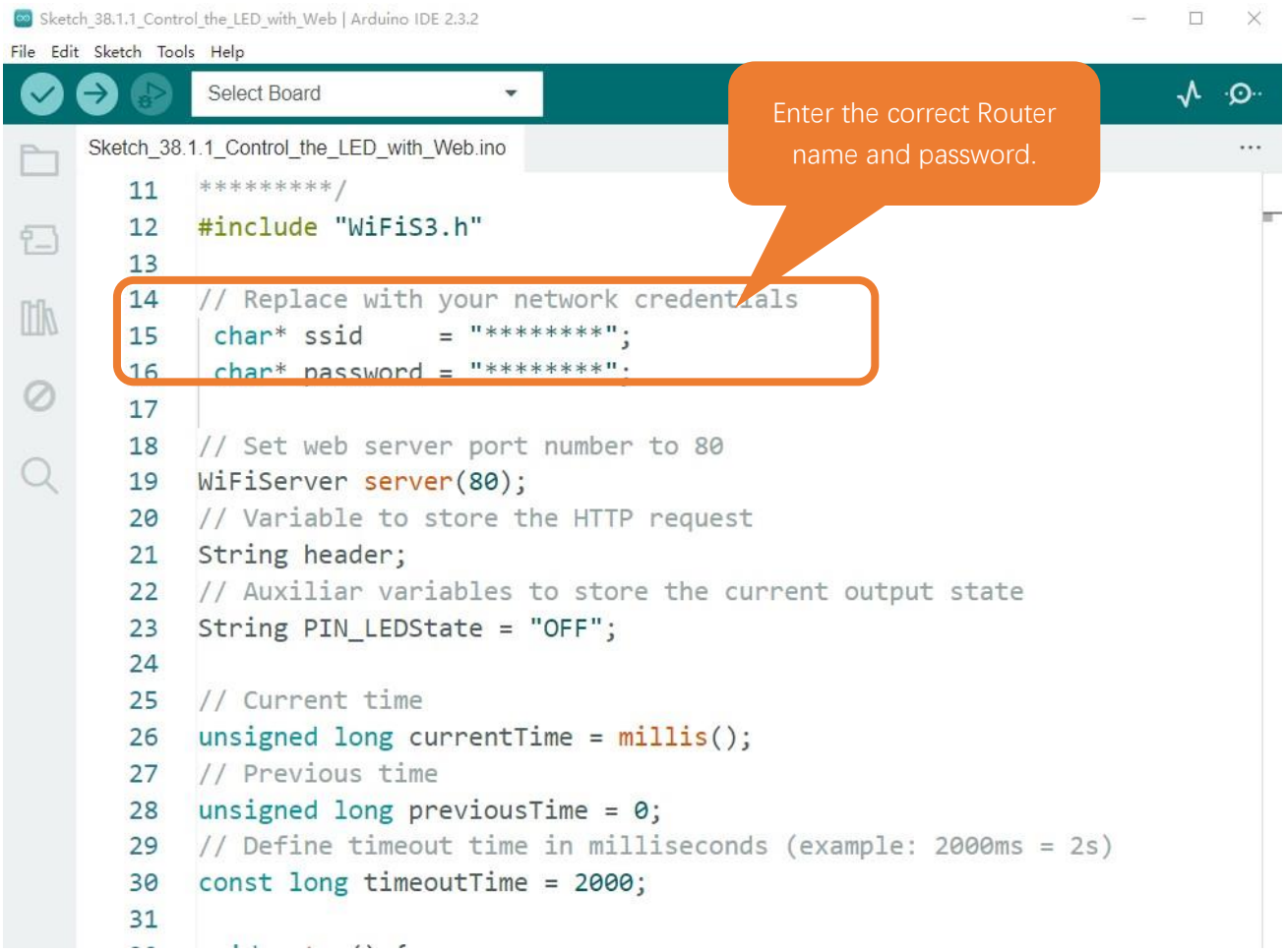
Circuit

Connect the board to the computer using the USB cable.

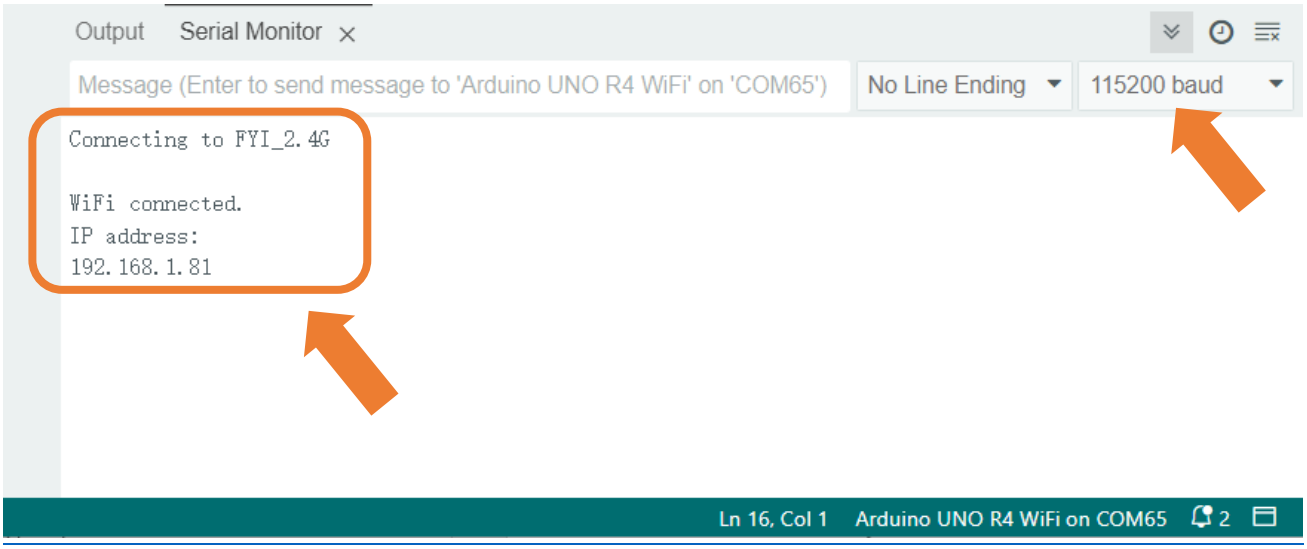


Sketch

Sketch_16.1.1



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:

```
#include "WiFiS3.h"

// Replace with your network credentials
 char* ssid     = "********";
 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              client.println(".button2{background-color: #4286f4;display: inline-block; border:
85  none; border-radius: 4px; color: white; padding: 16px 40px;text-decoration: none; font-size:
86  30px; margin: 2px; cursor: pointer;}</style></head>");
```

```
            // Web Page Heading
            client.println("<body><h1>Control Board Web Server</h1>");
87          client.println("<p>GPIO state: " + PIN_LEDState + "</p>");
88          client.println("<p><a href=\"/LED_BUILTIN/ON\"><button class=\"button
89  button2\">ON</button></a></p>");
90          client.println("<p><a href=\"/LED_BUILTIN/OFF\"><button class=\"button
    button2\">OFF</button></a></p>");
91          client.println("</body></html>");
            // The HTTP response ends with another blank line
92          client.println();
93          // Break out of the while loop
94          break;
95        } else {  // if you got a newline, then clear currentLine
96          currentLine = "";
97        }
98      } else if (c != '\r') {  // if you got anything else but a carriage return character,
99        currentLine += c;      // add it to the end of the currentLine
100     }
101   }
102 }
103 // Clear the header variable
104 header = "";
105 // Close the connection
106 client.stop();
107 Serial.println("Client disconnected.");
108 Serial.println("");
109 }
110 }
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              }
```

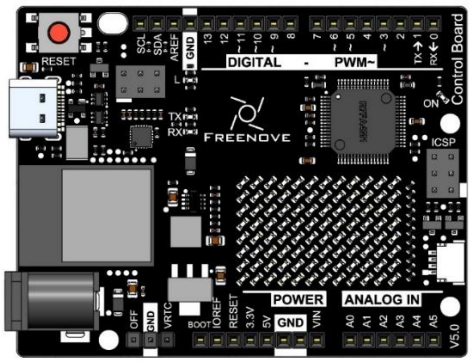
Chapter 17 Bluetooth (WiFi Board)

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

Project 17.1 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.

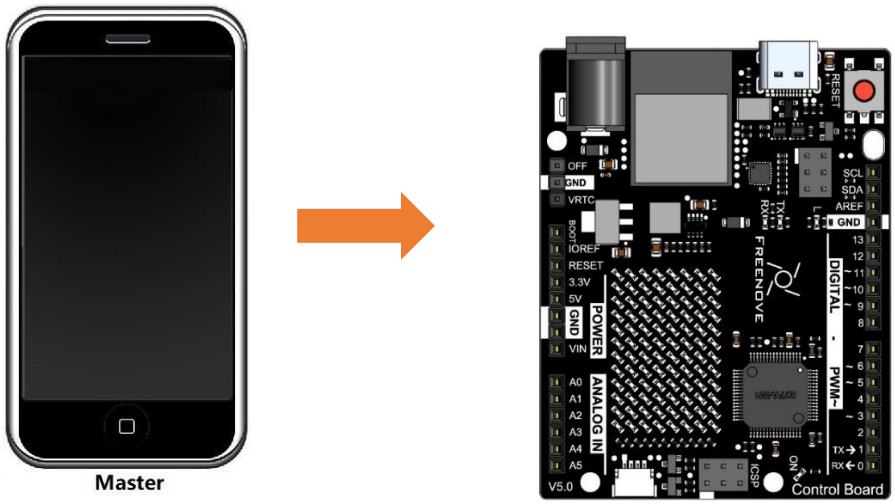
Slave mode

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.

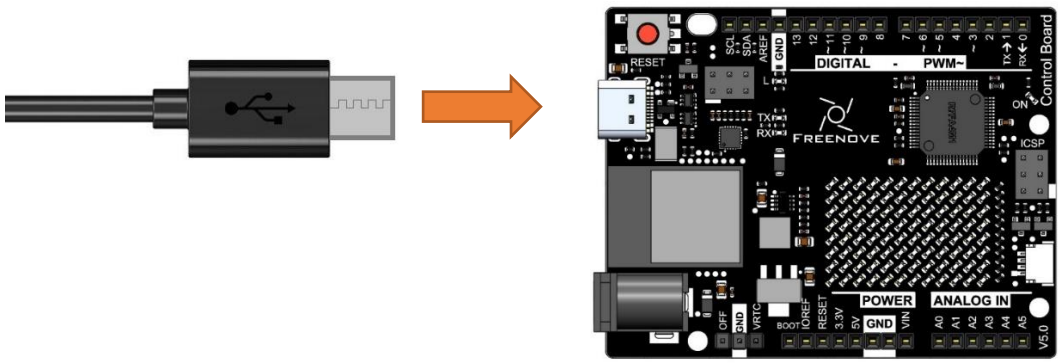
The control board is equipped with an ESP32 module, which provides Bluetooth and Wi-Fi functions, supporting speeds up to 2Mbps. The ESP32 module integrates a tracking antenna, which allows you to take advantage of the development board's connectivity without an external antenna.

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

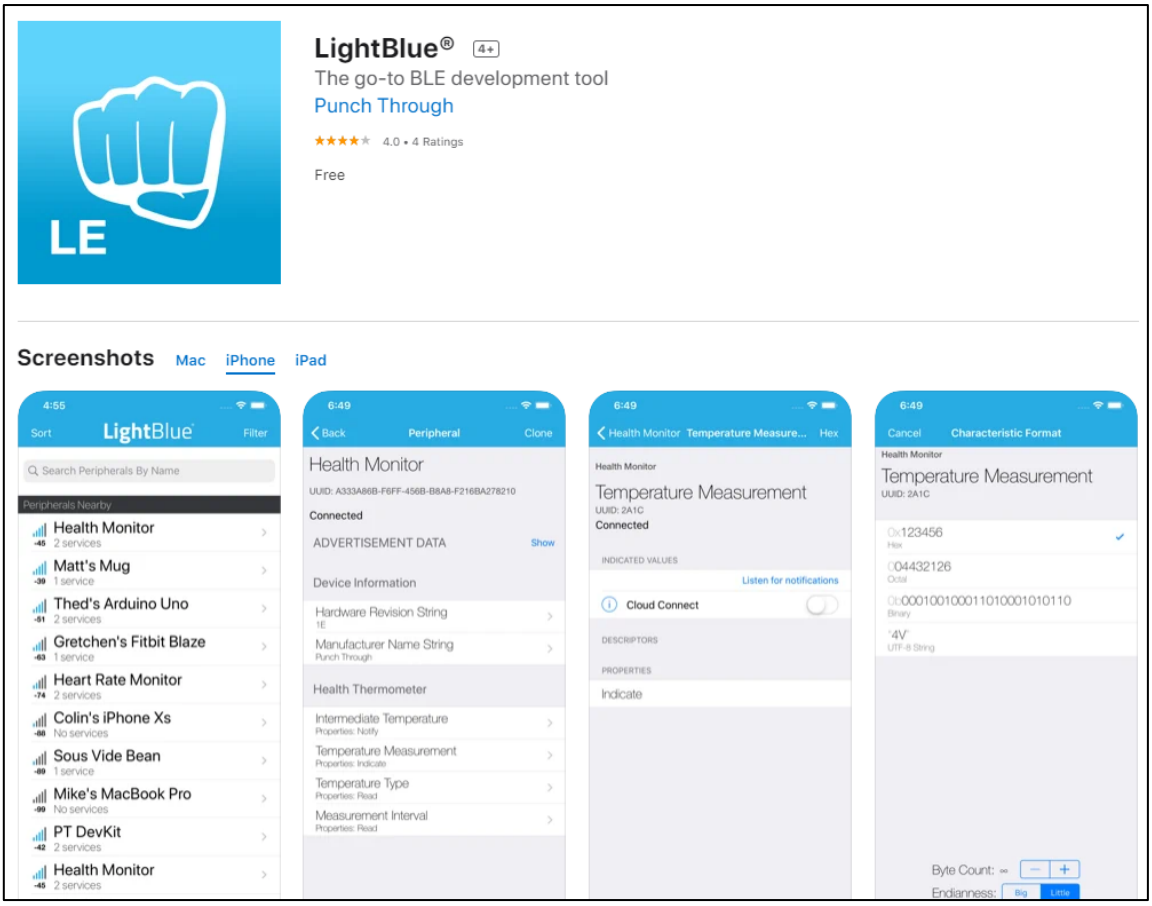
Connect Freenove Control board to the computer using the USB cable.



Sketch

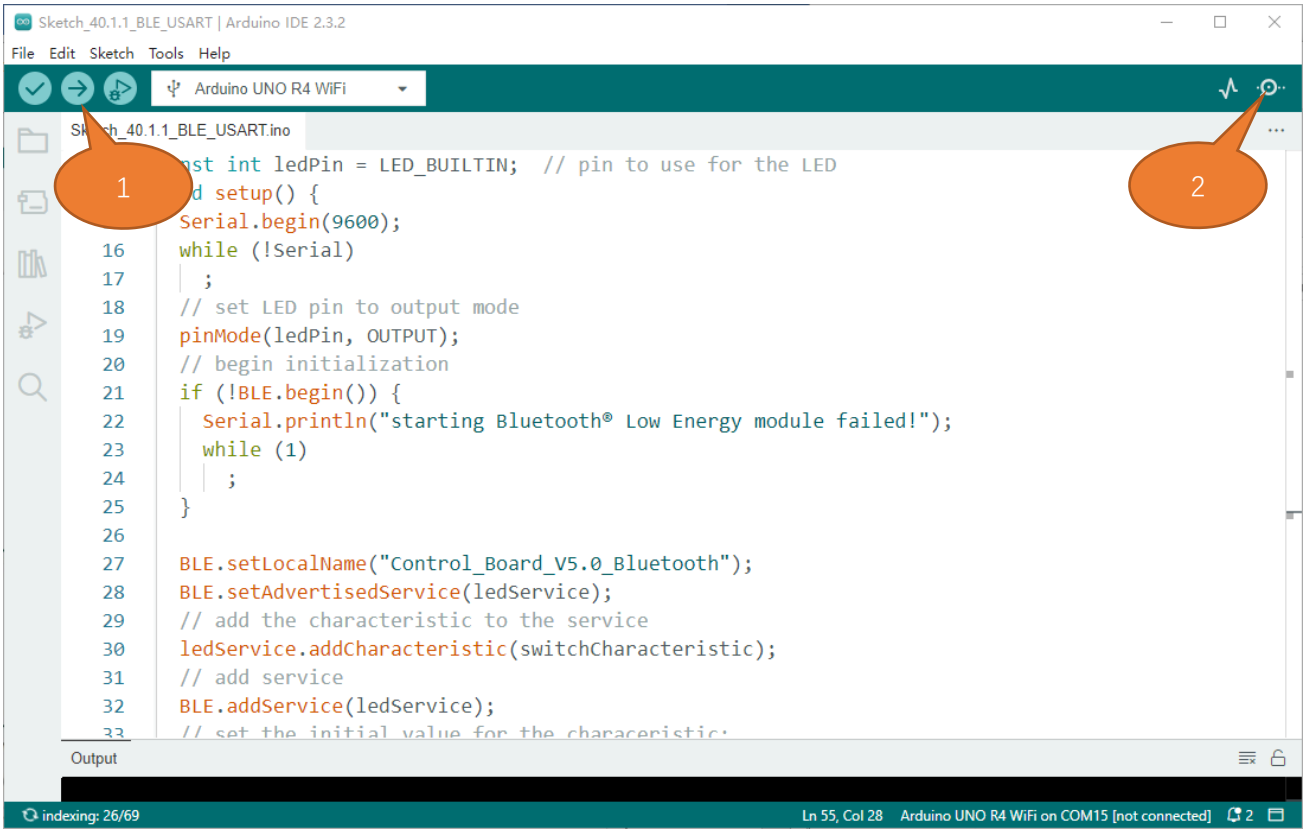
Lightblue

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.



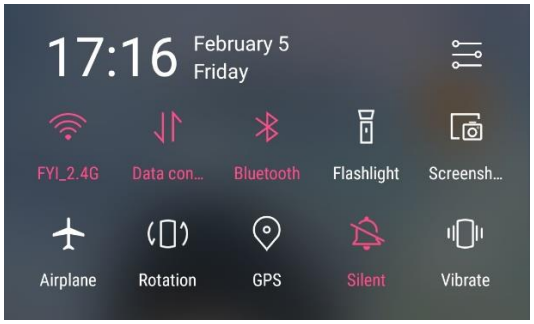
Step1. Upload the code of Project 16.1 to control board.

Step2. Click on serial monitor.

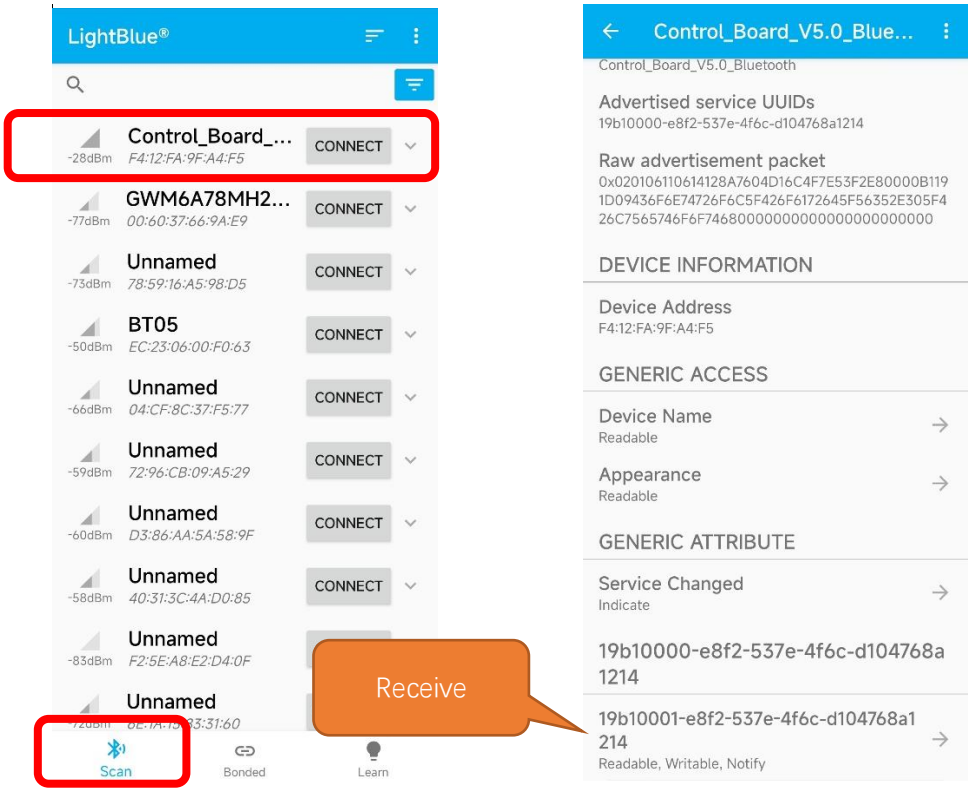


Step3. Set baud rate to 9600.

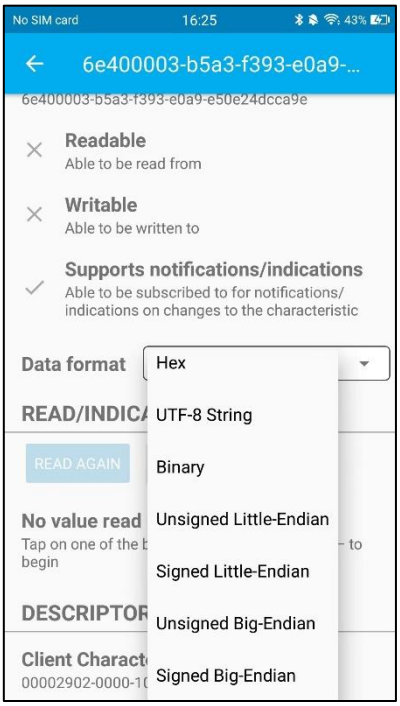
Turn ON Bluetooth on your phone, and open the Lightblue APP.

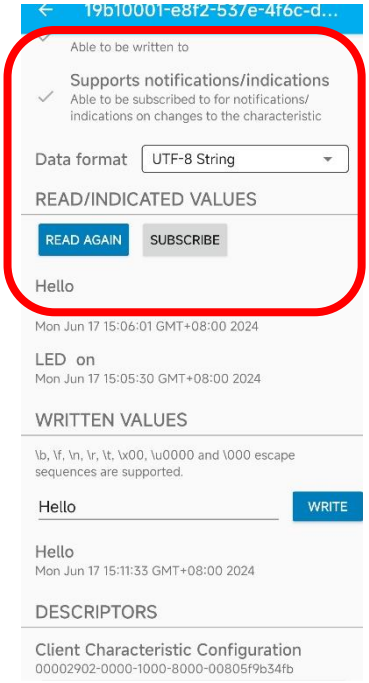


In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click Control_Board_V5.0_Bluetooth.



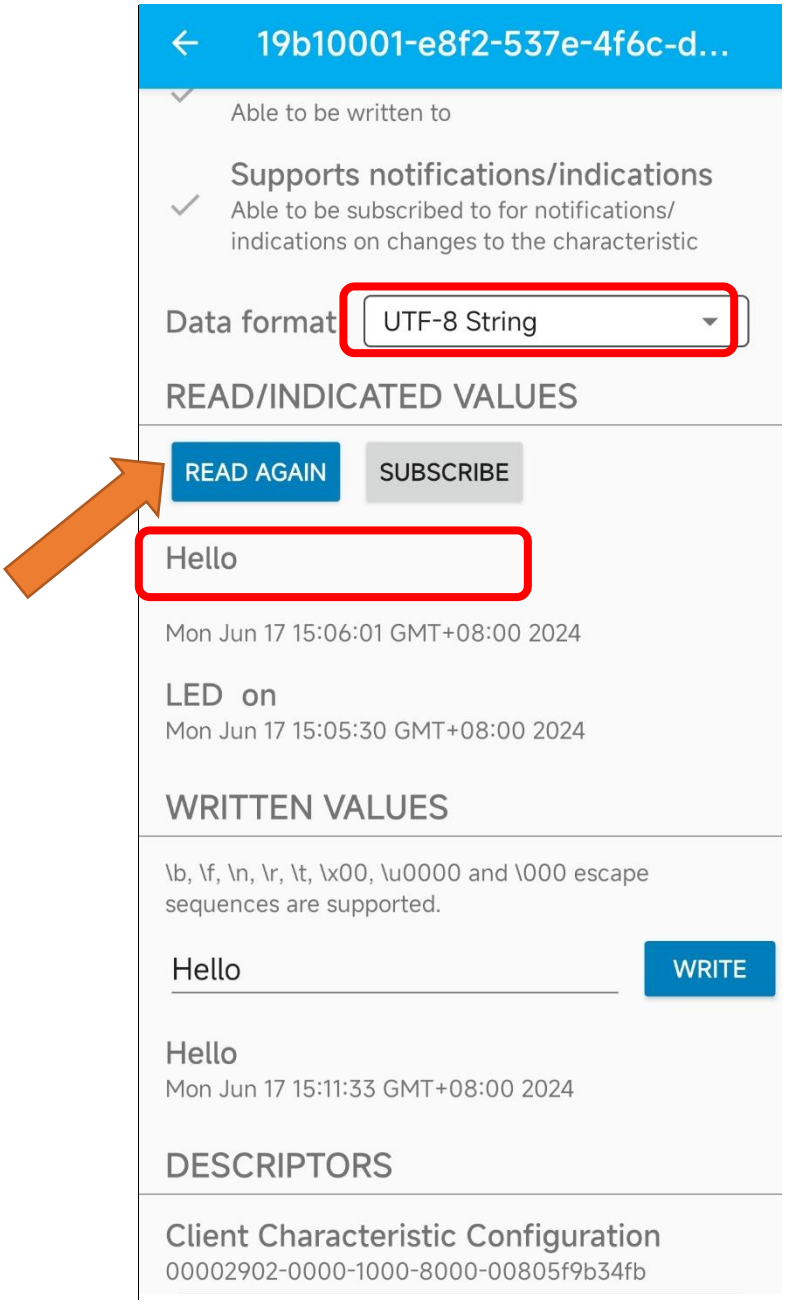
Click “Receive”. Select the appropriate Data format in the box to the right of Data Format. For example, HEX for hexadecimal, utf-string for character, Binary for Binary, etc. Then click SUBSCRIBE.



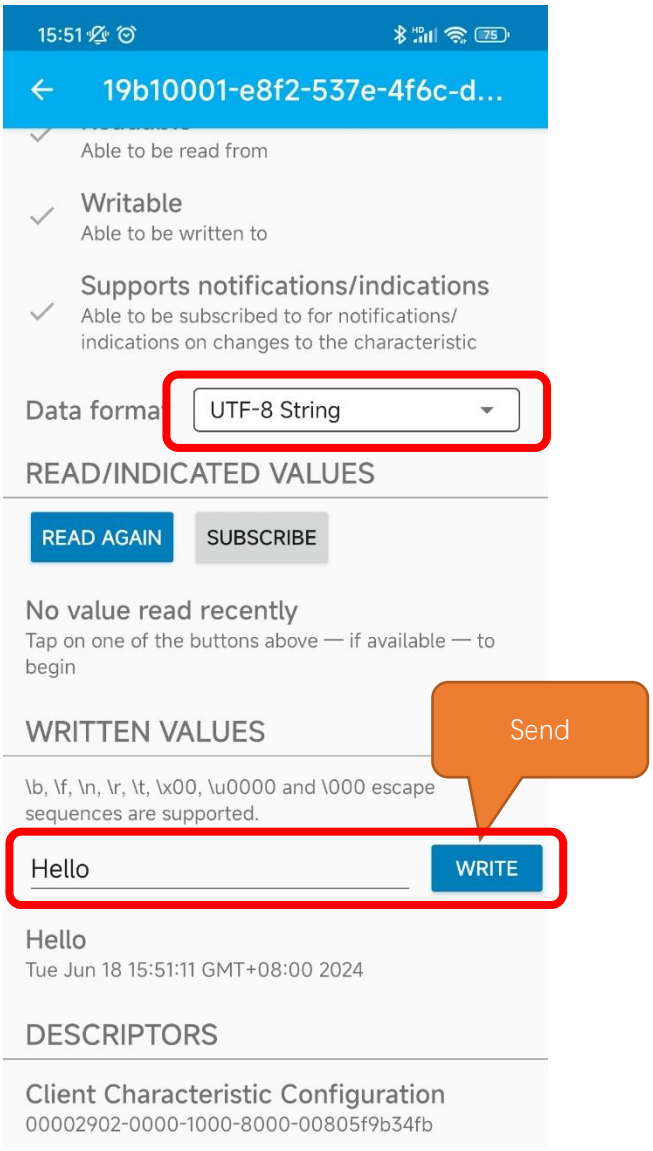


Back to the serial monitor on your computer. You can type anything in the left border of Send, and then click Send.

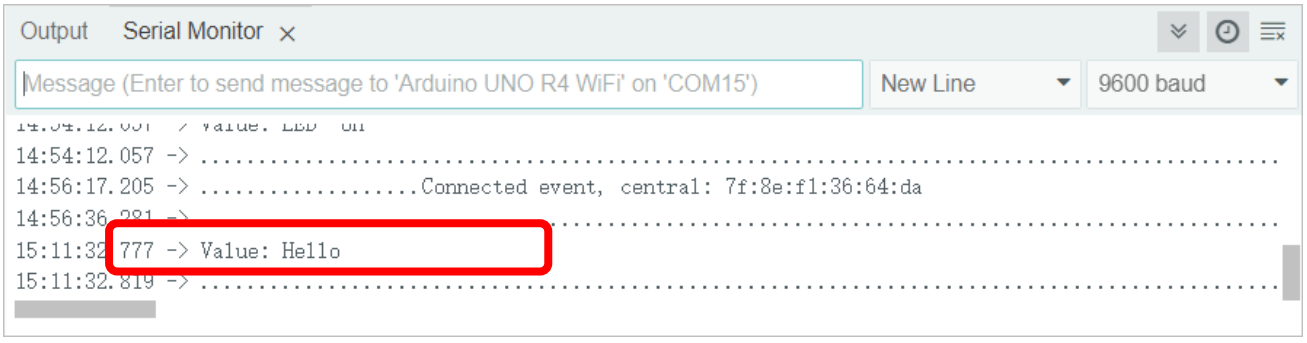
The last received data will be printed when the READ AGAIN button on the app is tapped.



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);

```

```
  // start advertising
  BLE.advertise();

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

void loop() {
  long nowtime = millis();
  if (nowtime - lasttime > 1000) {
    BLE.poll();
    Serial.print(".");
    if (Serial.available() > 0) {
      String str = Serial.readString();
      const char *newValue = str.c_str();
      switchCharacteristic.writeValue(newValue);
    }
    lasttime = nowtime;
  }
}

void blePeripheralConnectHandler(BLEDevice central) {
  Serial.print("Connected event, central: ");
  Serial.println(central.address());
}

void blePeripheralDisconnectHandler(BLEDevice central) {
  Serial.print("Disconnected event, central: ");
  Serial.println(central.address());
}

void switchCharacteristicWritten(BLEDevice central, BLECharacteristic characteristic) {
  Serial.print("Characteristic event, written: ");
  uint8_t characteristicValue[20];
  int bytesRead = characteristic.readValue(characteristicValue, sizeof(characteristicValue));
  Serial.print("Received bytes: ");
  for (int i = 0; i < bytesRead; i++) {
    Serial.print(characteristicValue[i], HEX);
    Serial.print(" ");
  }
  Serial.println();
  String receivedString = "";
  for (int i = 0; i < bytesRead; i++) {
    receivedString += (char)characteristicValue[i];
  }
```

```
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.

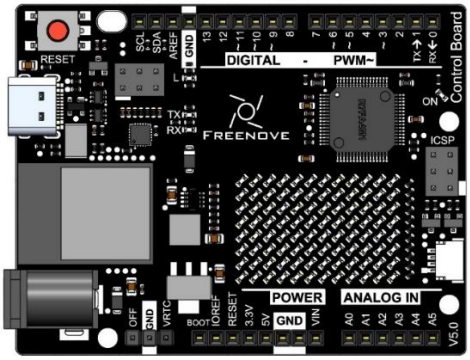
```
52      if (Serial.available() > 0) {
53        String str = Serial.readString();
54        const char *newValue = str.c_str();
55        switchCharacteristic.writeValue(newValue);
56      }
...   ...
71  void switchCharacteristicWritten(BLEDevice central, BLECharacteristic characteristic) {
72    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  }
```

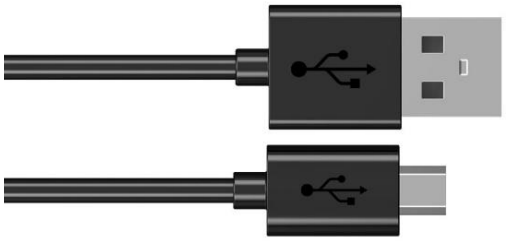
Project 17.2 Control LED with Bluetooth

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

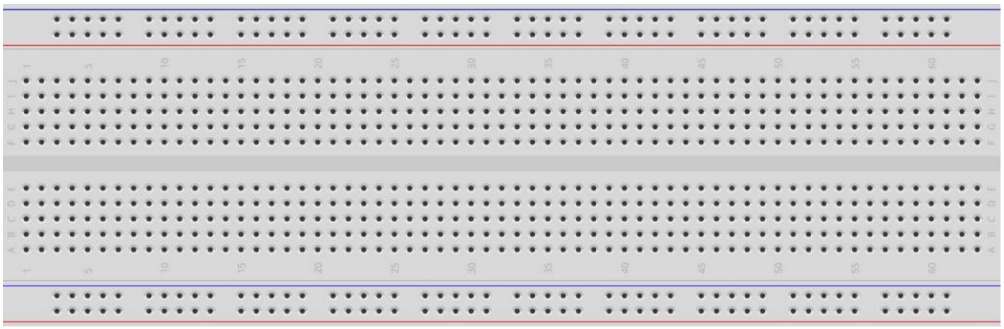
Component List

Control board x1



Micro USB Wire x1	LED x1	Resistor 220Ω x1	Jumper M/M x2
			

Breadboard x1

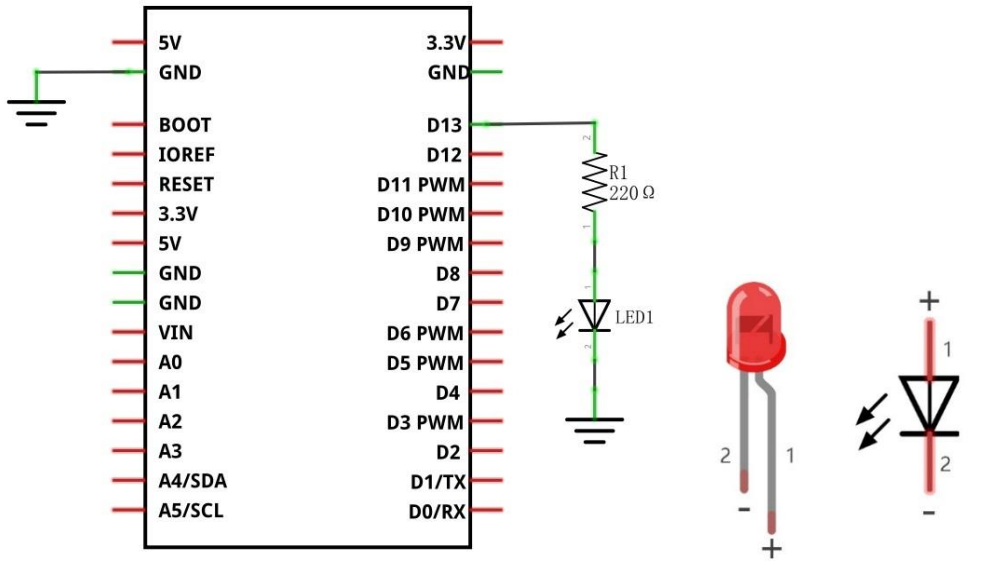


If you do not have the above components in your kit, you can use the control board alone to finish the project. The control pin of the LED in this project is the same as the one controlling the onboard LED.

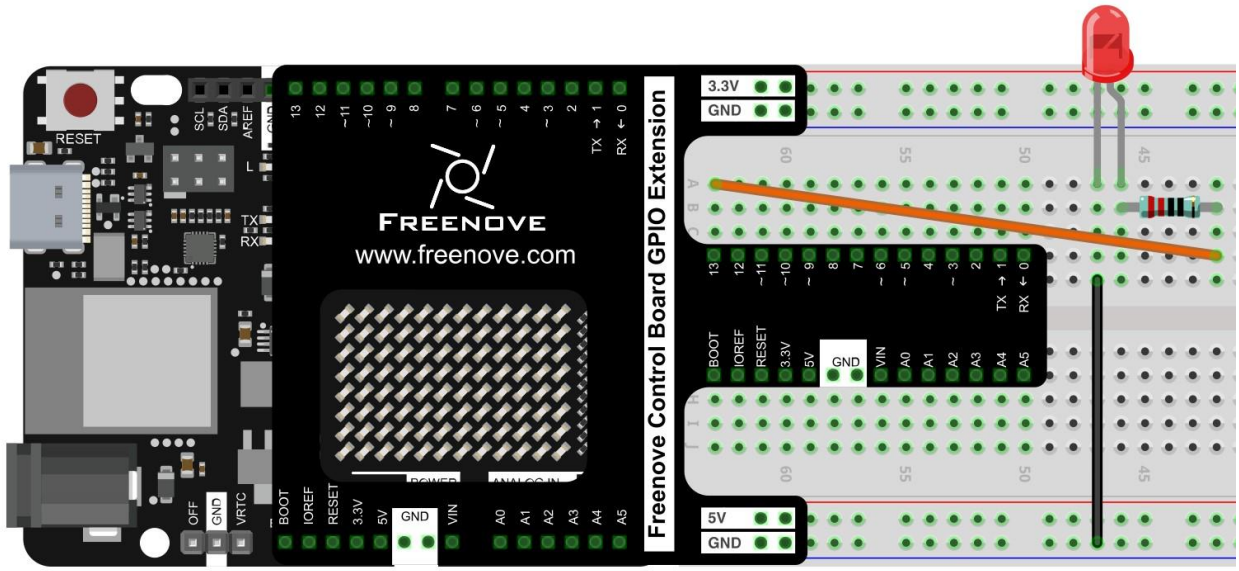
Circuit

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

Schematic diagram

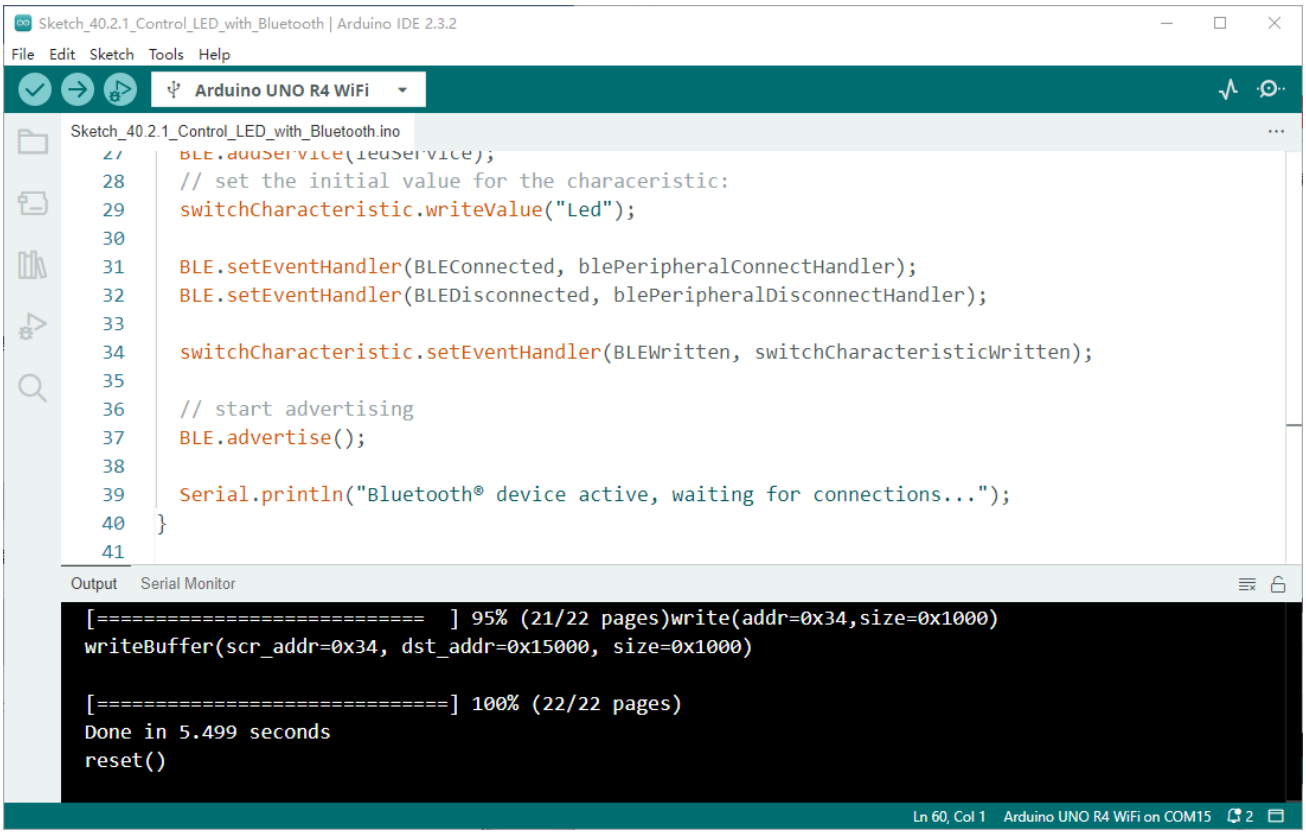


Hardware connection. **If you need any support, please contact us via: support@freenove.com**



Sketch

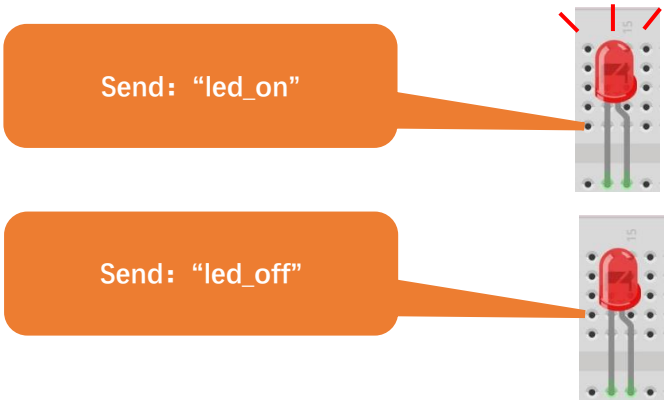
Sketch_17.2.1



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    }
```

Chapter 18 USB HID

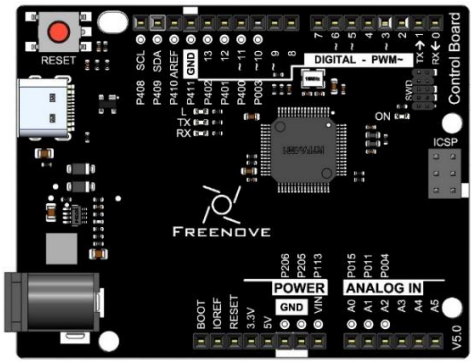
In this chapter, we will learn how to use a control board with keyboard and mouse APIs to emulate mouse and keyboard actions. This feature can be used to create game controllers, keyboard extensions, or other Human Interface Devices (HID).

Project 18.1 Mouse control

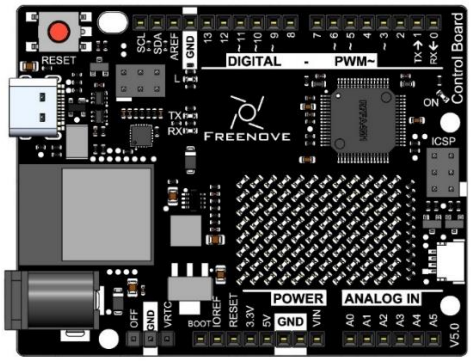
In this project, we learn how to emulate a mouse control.

Component List

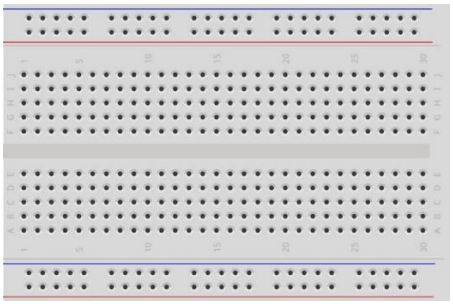
Control board x1



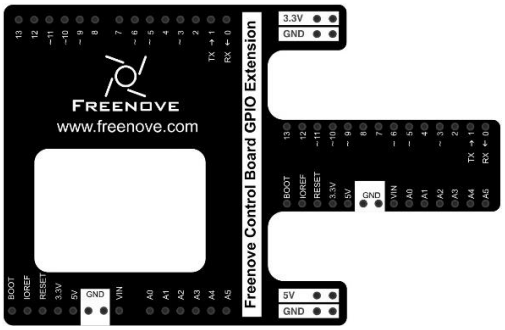
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

Resistor 10kΩ x8

Push button Switch x4

Jumper M/M x12

Component knowledge

Human Interface Device (HID)

Human Interface Devices (HID) are devices designed for human use (such as keyboards, mice, game controllers, etc.), which often send data to the computer via USB. When you press a key on the keyboard, you send data to the computer, the computer reads the data and then activates the corresponding key.

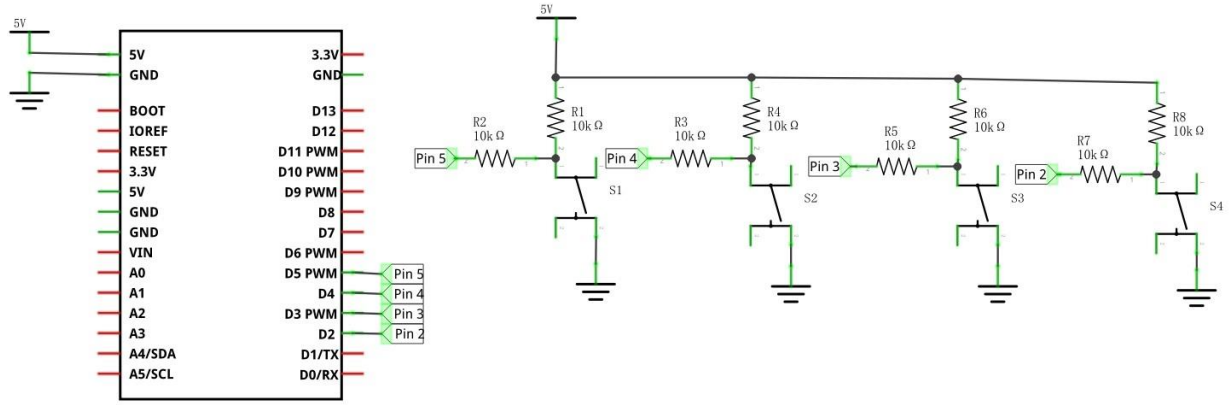
HID, or Human-Interface Devices, is a category of computer devices designed for direct interaction with humans and are typically used for input purposes. This category includes devices such as keyboards, mice, and game controllers. With the control board (V5), you can simulate these devices, unlocking a multitude of possibilities for DIY projects.

The control board (V5) has built-in support for HID, a feature found on most modern development boards but not on previous UNO versions.

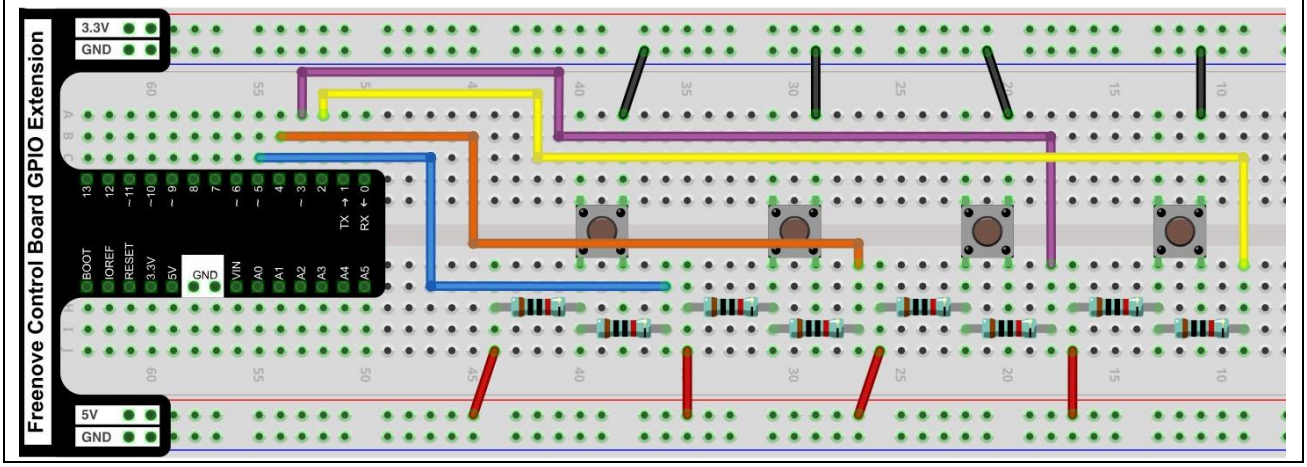
The control board (V5) is more than just a powerful development board; it also has built-in support for Human-Interface Devices (HID). This allows you to use the board to simulate devices such as mice and keyboards, adding a new level of interactivity to your projects.

Circuit

Schematic diagram



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



Sketch

Sketch_18.1.1

Upload the code to the control board, and you can easily control the movement of the mouse with the buttons. Here, four buttons are set up to control the mouse for moving up, down, left, and right.

The following is the program code:

```
#include <Mouse.h>  // Include the Mouse library for mouse control

const int upButton = 2;  // Define the pin where the button is connected
const int downButton  = 3;  // Define the pin where the button is connected
const int leftButton  = 4;  // Define the pin where the button is connected
const int rightButton = 5;  // Define the pin where the button is connected

void setup() {
  pinMode(upButton, INPUT_PULLUP);  // Set the button pin as an input
  pinMode(downButton, INPUT_PULLUP);  // Set the button pin as an input
  pinMode(leftButton, INPUT_PULLUP);  // Set the button pin as an input
  pinMode(rightButton, INPUT_PULLUP);  // Set the button pin as an input
  Mouse.begin();           // Initialize mouse control
  delay(1000);             // Wait for 1 second (1000 milliseconds) for hardware
initialization
}

void loop() {
  // Check if the button is pressed (HIGH)
  if (digitalRead(rightButton) == LOW) {
    //right
```

```
22      Mouse.move(10, 0);  // Move the mouse 10 units to the right
23      delay(200);         // Wait for 200 milliseconds to slow down mouse movement
24    }
25    if (digitalRead(leftButton) == LOW) {
26      //Left
27      Mouse.move(-10, 0);
28      delay(200);
29    }
30    if (digitalRead(downButton) == LOW) {
31      // Down
32      Mouse.move(0, 10);
33      delay(200);
34    }
35    if (digitalRead(upButton) == LOW) {
36      // Up
37      Mouse.move(0, -10);
38      delay(200);
39    }
40  }
```

Include the mouse library for mouse control.

```
1   #include <Mouse.h>  // Include the Mouse library for mouse control
```

Determine whether a button is press, and execute corresponding mouse movement.

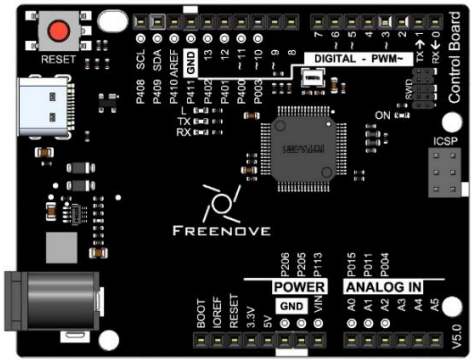
```
19    // Check if the button is pressed (HIGH)
20    if (digitalRead(rightButton) == LOW) {
21      //right
22      Mouse.move(10, 0);  // Move the mouse 10 units to the right
23      delay(200);         // Wait for 200 milliseconds to slow down mouse movement
24    }
25    if (digitalRead(leftButton) == LOW) {
26      //Left
27      Mouse.move(-10, 0);
28      delay(200);
29    }
30    if (digitalRead(downButton) == LOW) {
31      // Down
32      Mouse.move(0, 10);
33      delay(200);
34    }
35    if (digitalRead(upButton) == LOW) {
36      // Up
37      Mouse.move(0, -10);
38      delay(200);
39    }
```

Project 18.2 Keypad Control

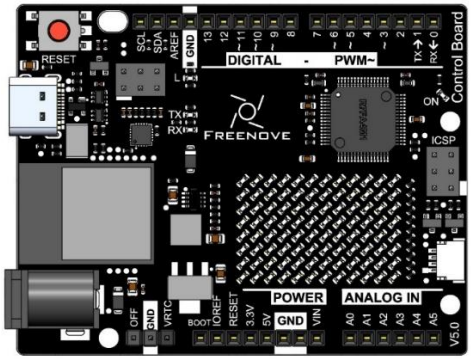
In this project, we will control keyboard input through buttons.

Component List

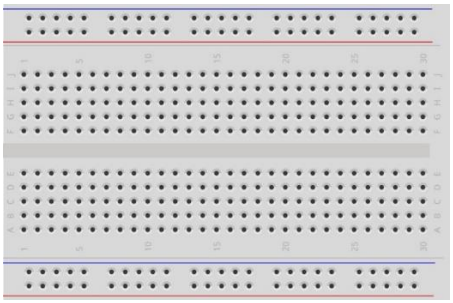
Control board x1



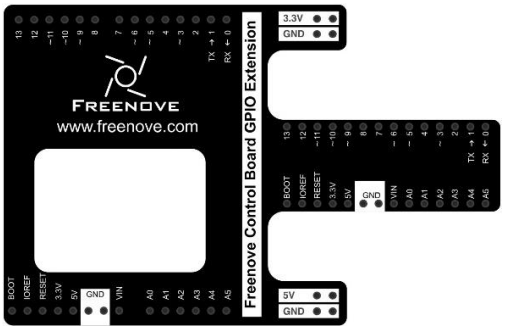
or



Breadboard x1



GPIO Extension Board x1



USB cable x1

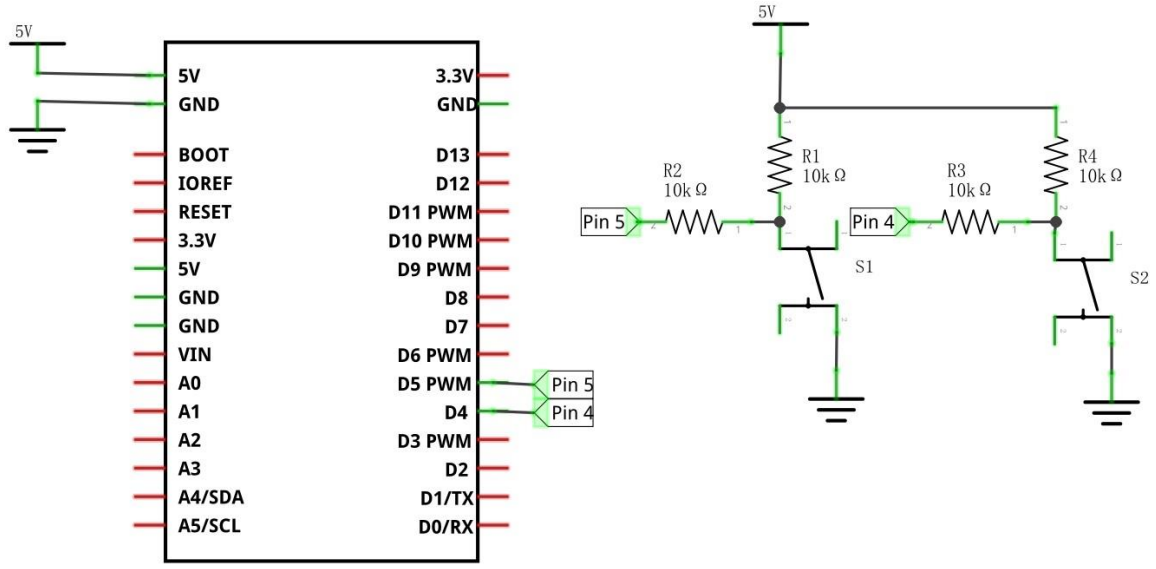
Resistor 10kΩ x2

Push button Switch x4

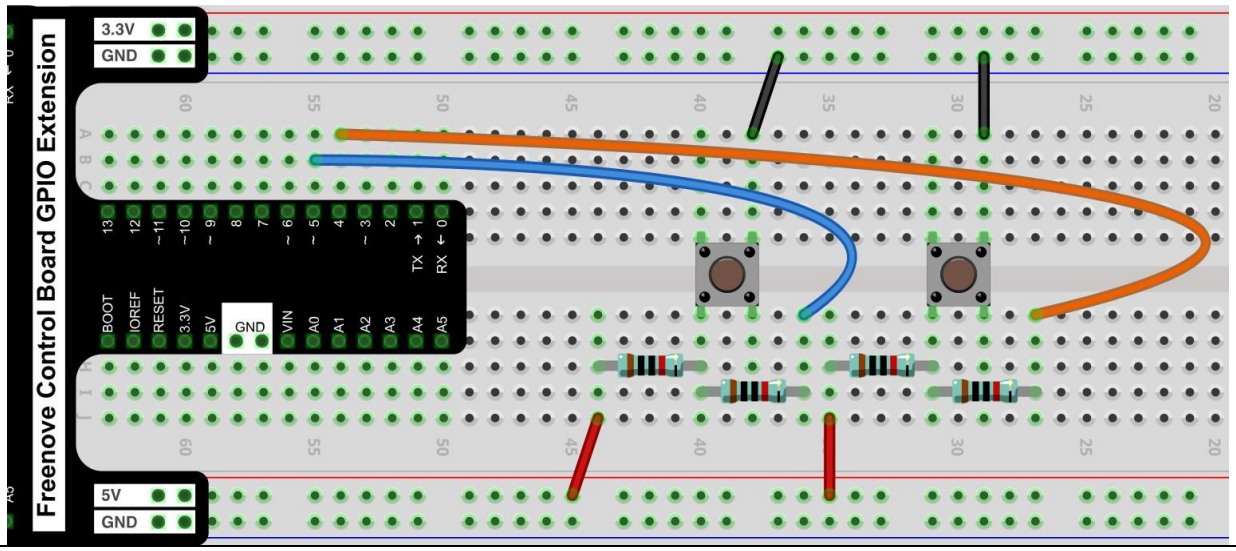
Jumper M/M x6

Circuit

Schematic diagram



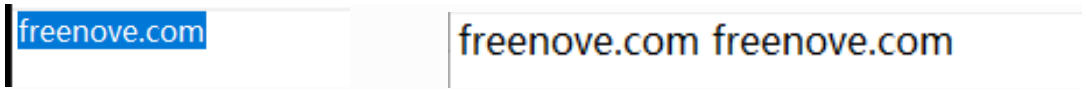
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_18.2.1

Upload the code to the control board, and you can easily control keyboard input through buttons. Here, two buttons are set up: one for copying (simultaneously pressing the CTRL and C keys on the keyboard), and the other for pasting (simultaneously pressing the CTRL and V keys on the keyboard). We can use these two buttons to complete the copy and paste of text. For example, to copy the text "Freenove.com" from the file "test.txt," use the mouse to select the text, press the copy button, and then press the paste button to perform the text copy.



Select the texts to copy Press the copy and paste buttons in turn

The following is the program code:

```
#include <Keyboard.h>  // Include the Keyboard library to enable keyboard functionalities

const int copyButtonPin = 4;  // Pin number for the copy button
const int pasteButtonPin = 5; // Pin number for the paste button

void setup() {
  Keyboard.begin();  // Initialize the Keyboard library
  pinMode(copyButtonPin, INPUT_PULLUP);  // Set the copy button pin as input
  pinMode(pasteButtonPin, INPUT_PULLUP); // Set the paste button pin as input
  delay(1000);       // Wait for 1 second to allow hardware initialization
}

void loop() {
  // Check if the copy button is pressed
  if (digitalRead(copyButtonPin) == LOW) {
    Keyboard.press(KEY_LEFT_CTRL);  // Press the Ctrl key
    Keyboard.press('c');            // Press the 'c' key
    Keyboard.releaseAll();          // Release all keys
    delay(150);                     // Wait for 200 milliseconds
  }
  // Check if the paste button is pressed
  else if (digitalRead(pasteButtonPin) == LOW) {
    Keyboard.press(KEY_LEFT_CTRL);  // Press the Ctrl key
    Keyboard.press('v');            // Press the 'v' key
    Keyboard.releaseAll();          // Release all keys
    delay(150);                     // Wait for 200 milliseconds
  }
}
```

Include the Keyboard library to enable keyboard functionalities

```
#include <Keyboard.h>  // Include the Keyboard library to enable keyboard functionalities
```

Determine whether a button is pressed, and execute corresponding key input values.

```
  if (digitalRead(copyButtonPin) == LOW) {
    Keyboard.press(KEY_LEFT_CTRL);  // Press the Ctrl key
    Keyboard.press('c');            // Press the 'c' key
    Keyboard.releaseAll();          // Release all keys
    delay(150);                     // Wait for 200 milliseconds
```

```
20    }
21    // Check if the paste button is pressed
22    else if (digitalRead(pasteButtonPin) == LOW) {
23      Keyboard.press(KEY_LEFT_CTRL);  // Press the Ctrl key
24      Keyboard.press('v');            // Press the 'v' key
25      Keyboard.releaseAll();          // Release all keys
26      delay(150);                     // Wait for 200 milliseconds
27    }
```

For more examples of USB HID, please refer to:

https://docs.arduino.cc/tutorials/uno-r4-minima/usb-hid/

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.

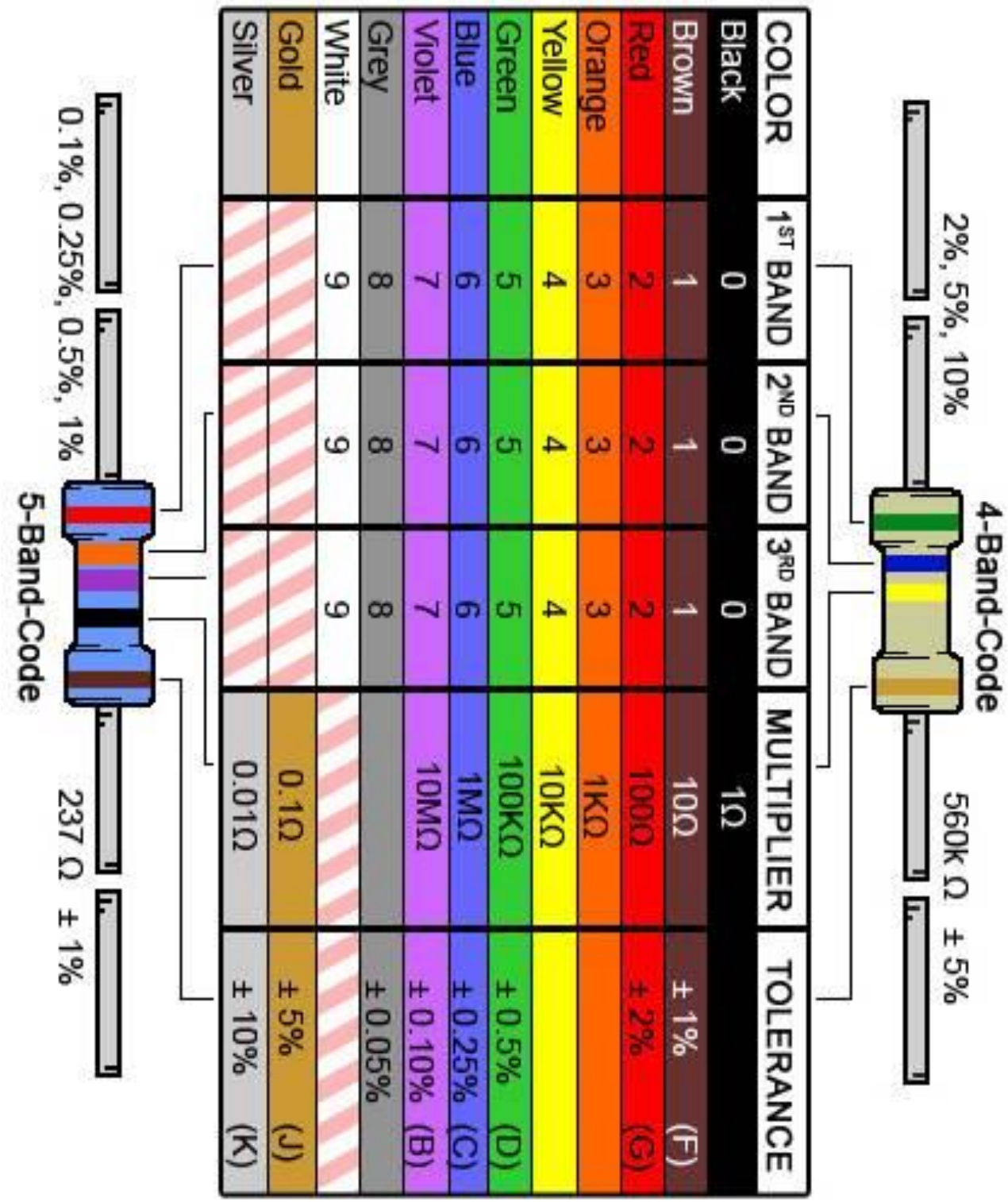
Thank you again for choosing Freenove products.

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)