Welcome

Thank you for choosing Freenove products!

How to Start

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

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

Unpack

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

Get Support

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

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

support@freenove.com

Attention

Pay attention to safety when using and storing this product:

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

About

Freenove provides open source electronic products and services.

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

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

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

http://www.freenove.com

Copyright

Contents

Preface

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

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

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

Support email: support@freenove.com

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

Control Board

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

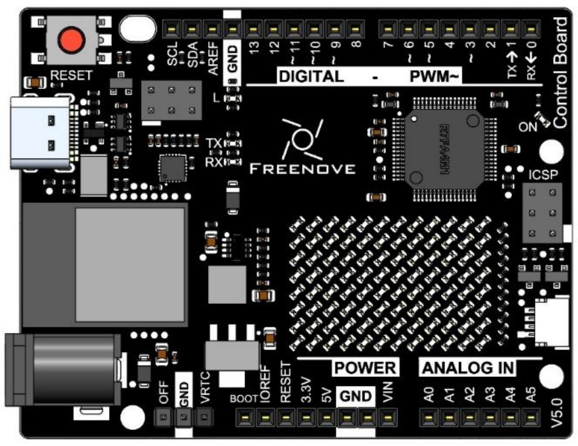
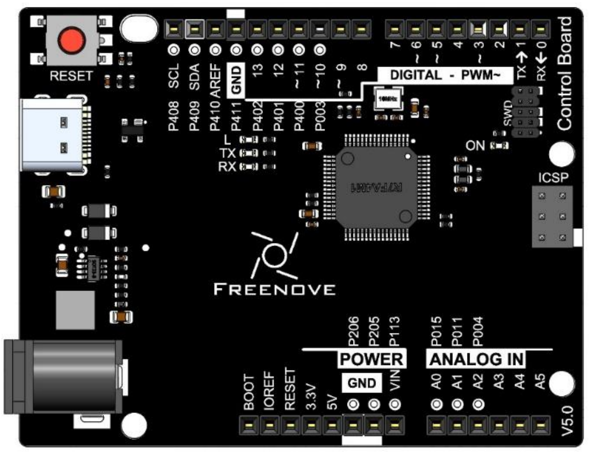
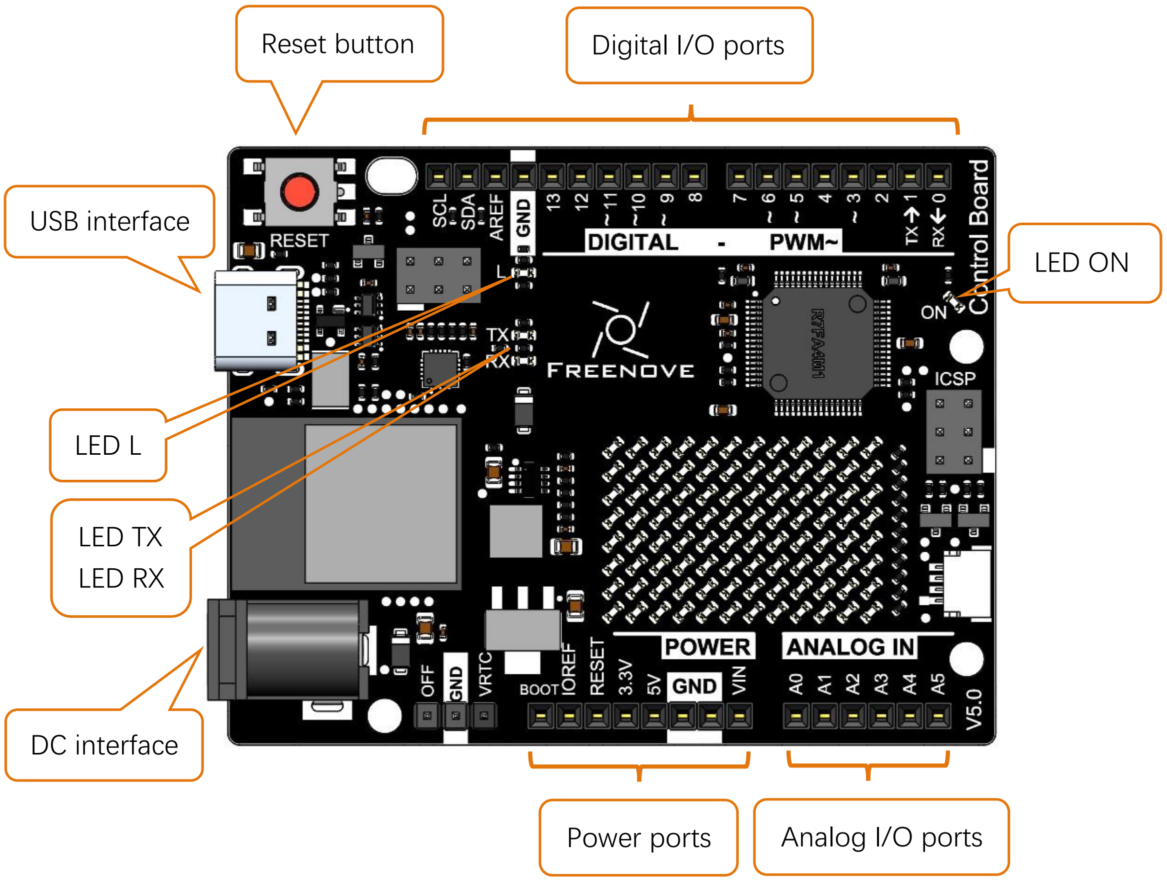
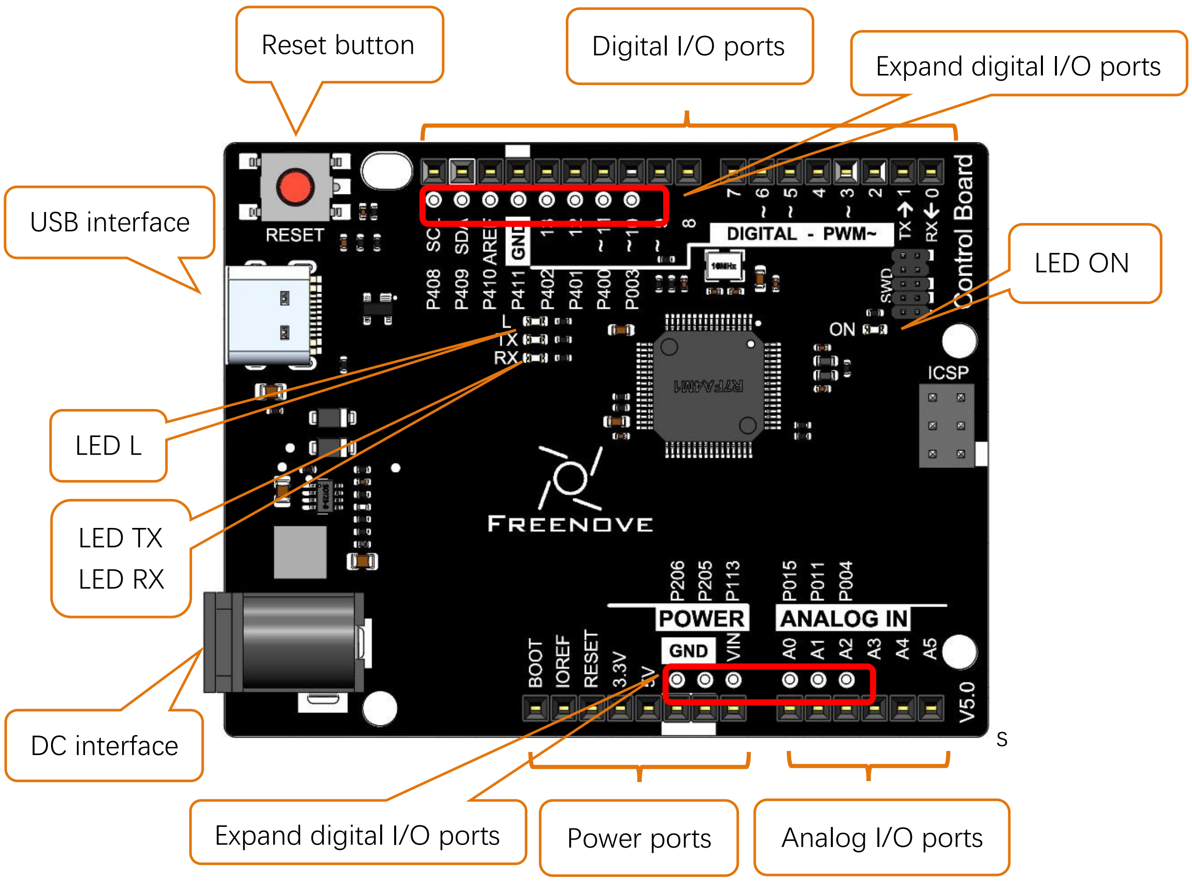
Freenove Control Board (WiFi)	Freenove Control Board (minina)
	

Diagram of the Freenove control board is shown below:





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

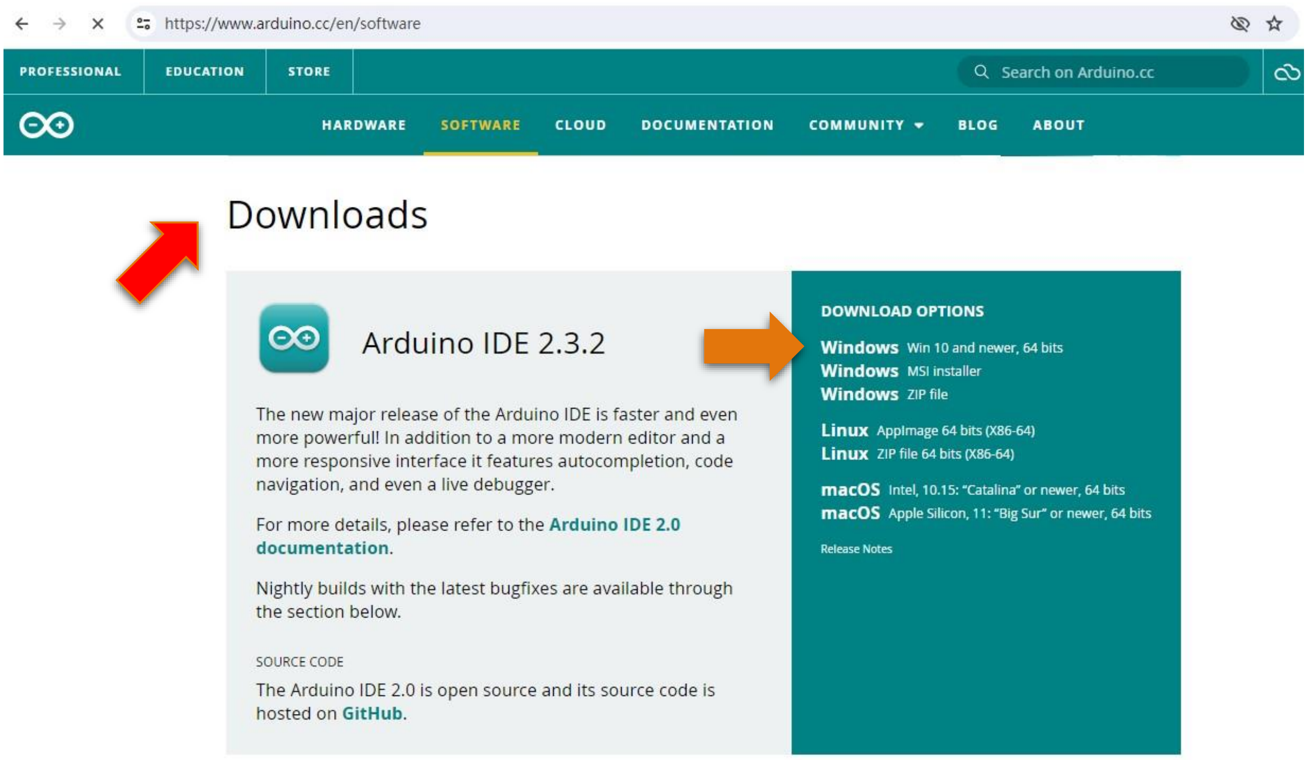
Chapter 0 Getting Ready (Important)

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

Programming Software

We use Arduino® IDE to write and upload code for the control board, which is a free and open source. (Arduino® is a trademark of Arduino LLC.)
Arduino IDE uses C/C++ programming language. Don't worry if you have never used it, because this document contains programming knowledge and detailed explanation of the code.

First, install Arduino IDE. Visit https://www.arduino.cc/en/software. Scroll down and find **Legacy 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".



Downloads

Arduino IDE 2.3.2

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

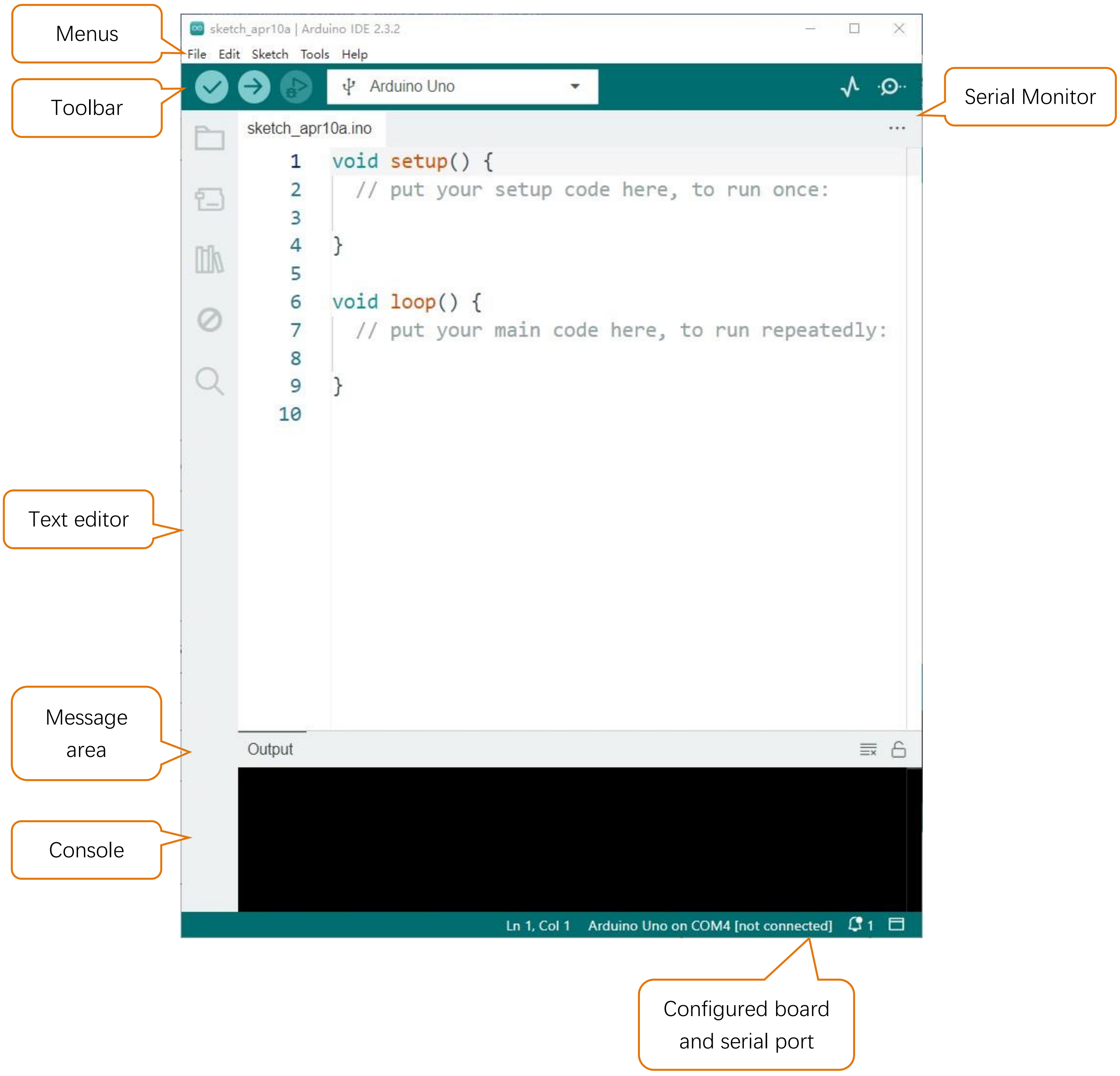
Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.15: "Catalina" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

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:


Menus
Toolbar
Serial Monitor
Text editor
Message area
Console
Configured board and serial port
sketch_apr10a | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
sketch_apr10a.ino
Output
Ln 1, Col 1 Arduino Uno on COM4 [not connected]

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

}

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

}

```

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.

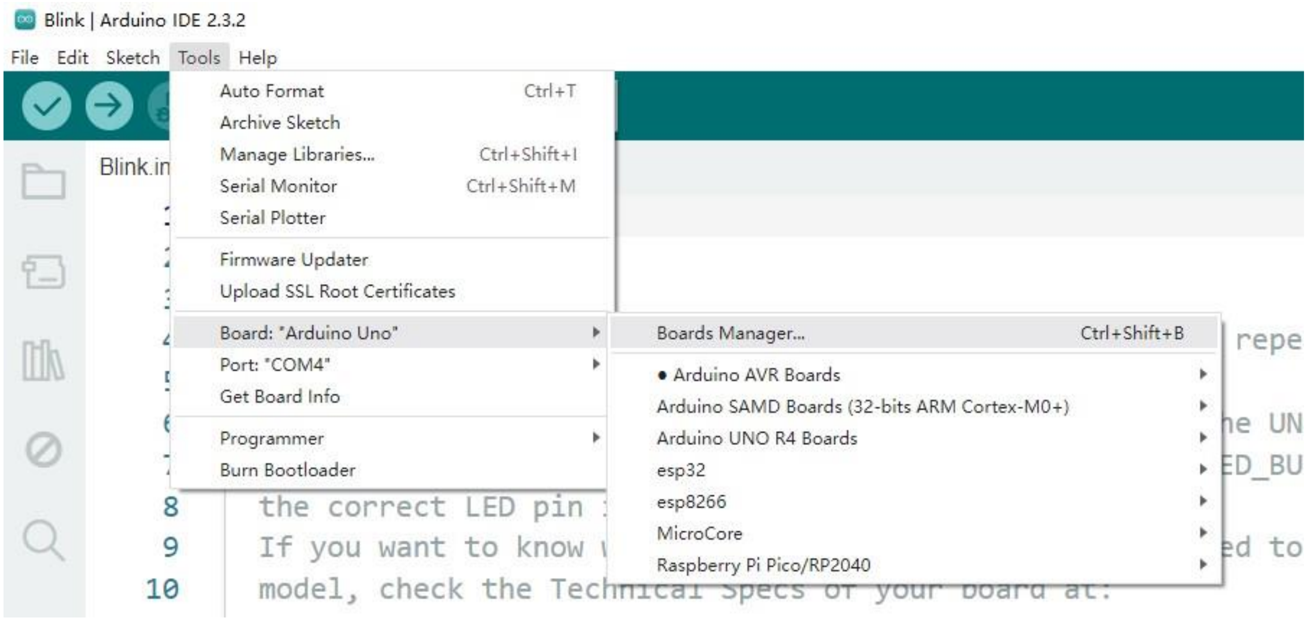
Save
Saves your sketch.

Serial Monitor
Opens the serial monitor.

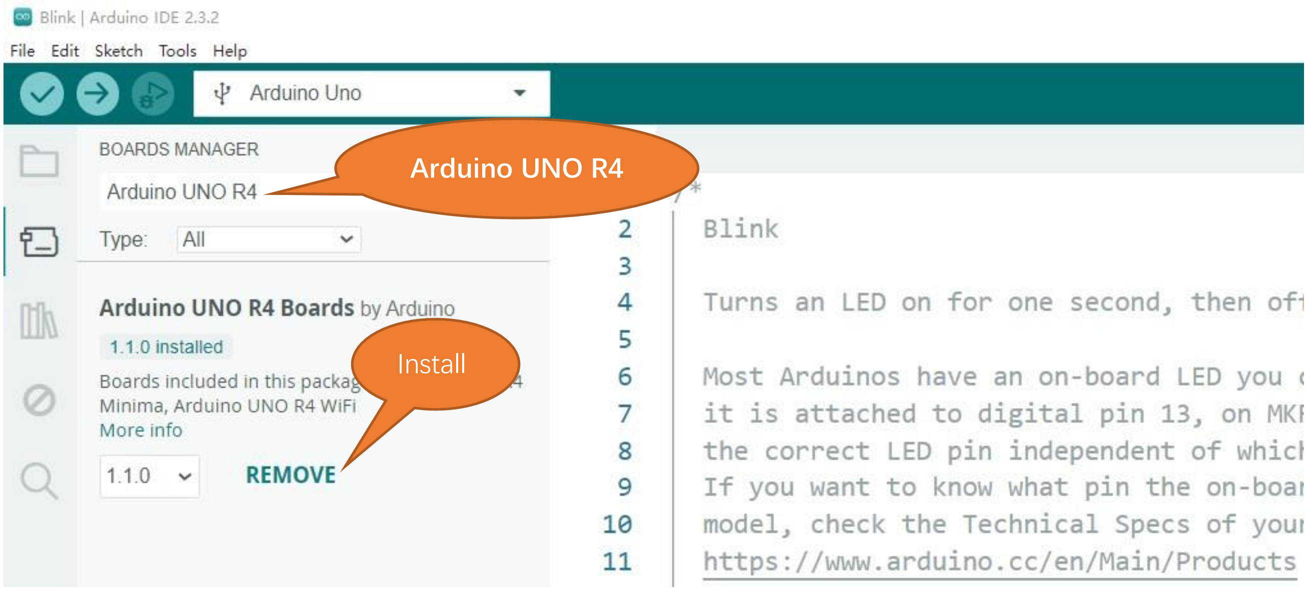
Additional commands are found within five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Installation of Development Board Support Package

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



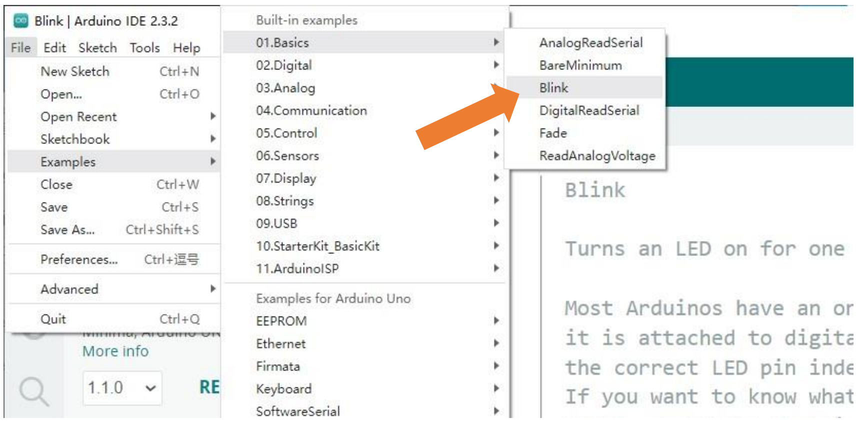
Enter **Arduino UNO R4** in the searching box, and select "**Arduino UNO R4**" and click on Install.


Blink | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino Uno
BOARDS MANAGER
Arduino UNO R4
Arduino UNO R4
Type: All
Arduino UNO R4 Boards by Arduino
1.1.0 installed
Boards included in this package: ... Minima, Arduino UNO R4 WiFi
More info
1.1.0
REMOVE
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.

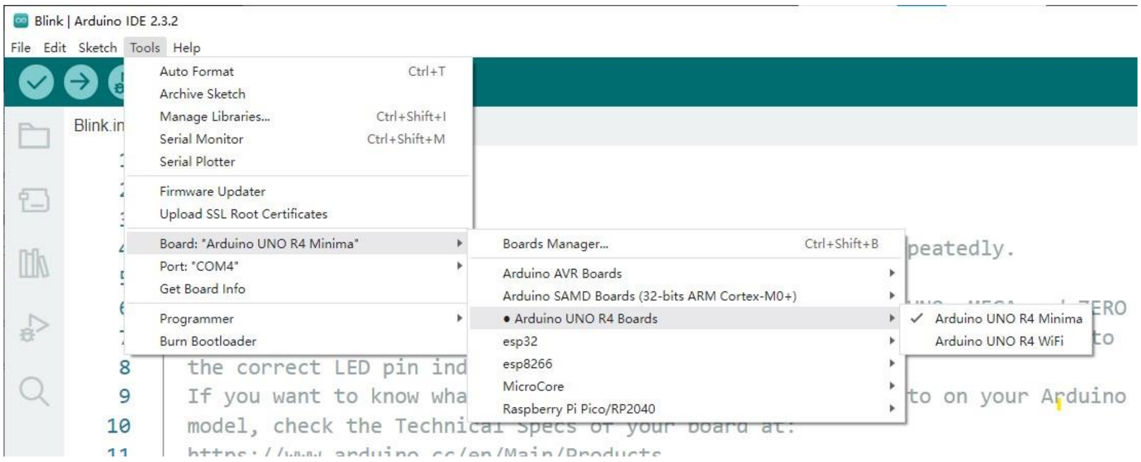
First Use

Open the example sketch "Blink".

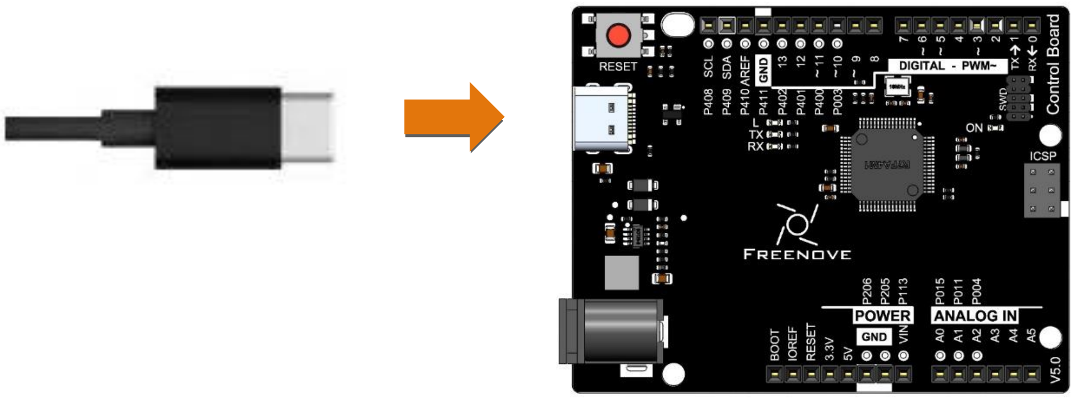


Choose the board corresponding to the model at hand. Here we take Freenove Control Board (minina) as an example.

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


Blink | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino UNO R4 Minima"
Port: "COM4"
Get Board Info
Programmer
Burn Bootloader
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
Arduino SAMD Boards (32-bits ARM Cortex-M0+)
● Arduino UNO R4 Boards
esp32
esp8266
MicroCore
Raspberry Pi Pico/RP2040
✓ Arduino UNO R4 Minima
Arduino UNO R4 WiFi

Connect control board to your computer with USB cable.



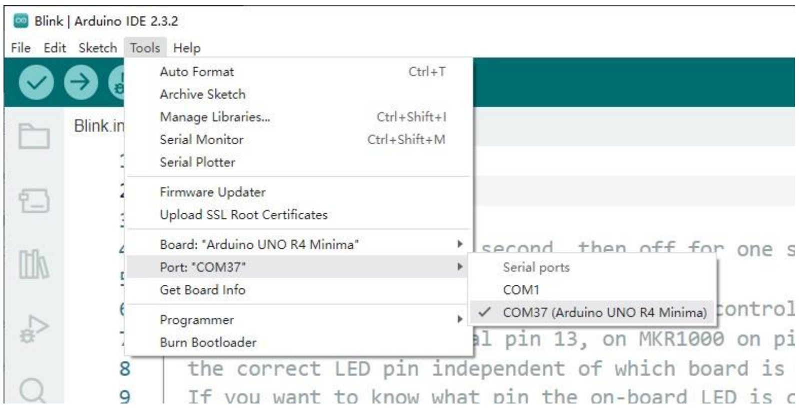
Select the port.

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.

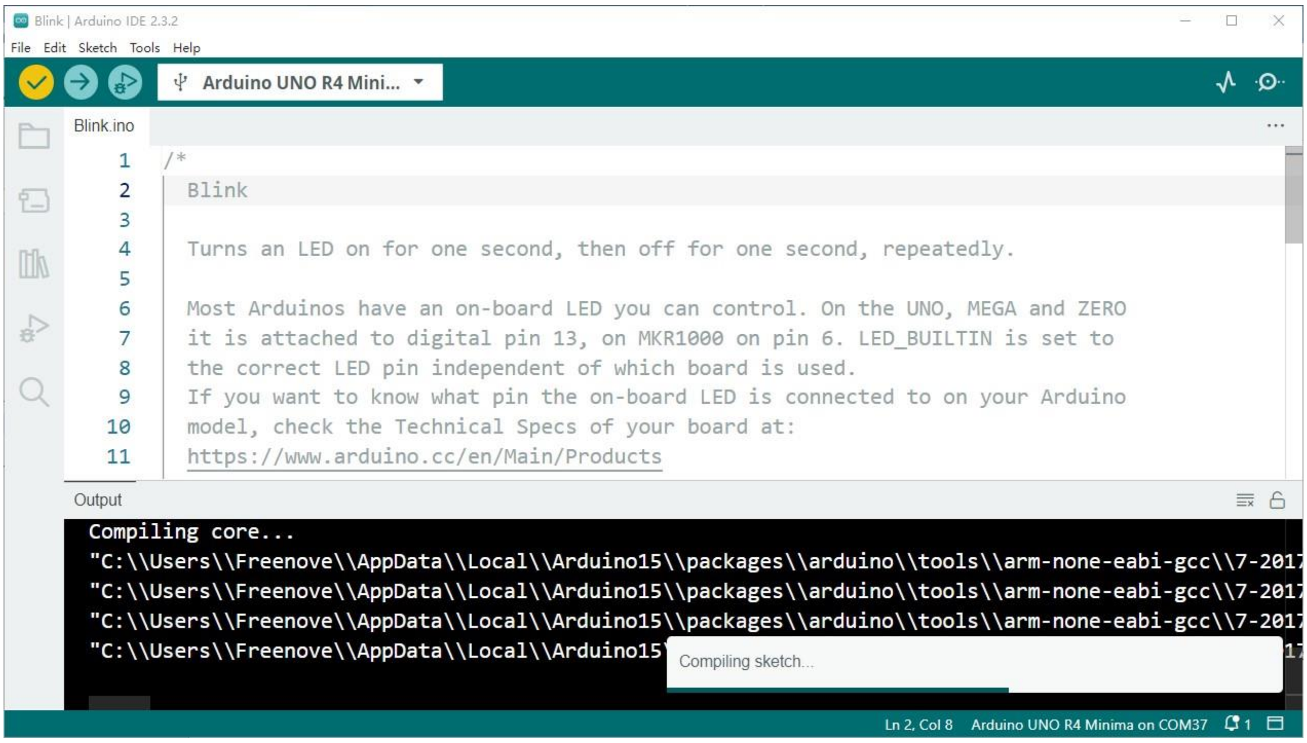
On Linux: It may be /dev/ttyUSB0, /dev/ttyACM0 or something like that.


Blink | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino UNO R4 Minima"
Port: "COM37"
Get Board Info
Programmer
Burn Bootloader
Serial ports
COM1
✓ COM37 (Arduino UNO R4 Minima)

Note: If there are more than one ports available 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 newly generated one is the correct port.

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

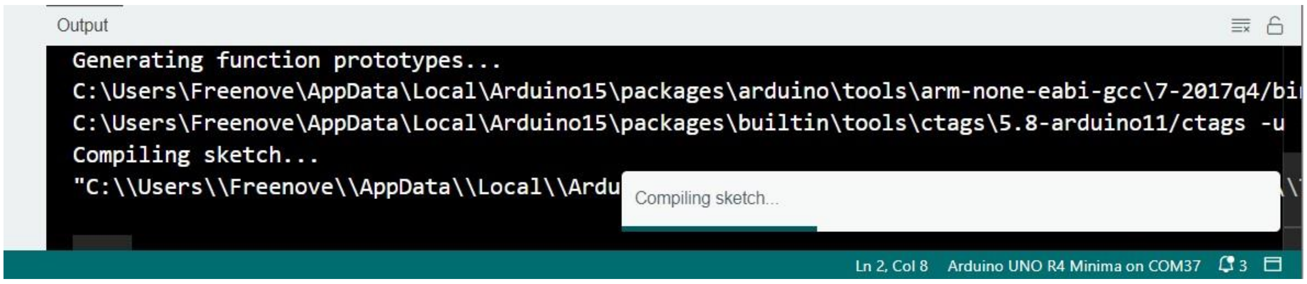
Click "Verify" button.


Blink | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Arduino UNO R4 Mini...
Blink.ino
Output
Compiling sketch...
Ln 2, Col 8 Arduino UNO R4 Minima on COM37 1

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products
```

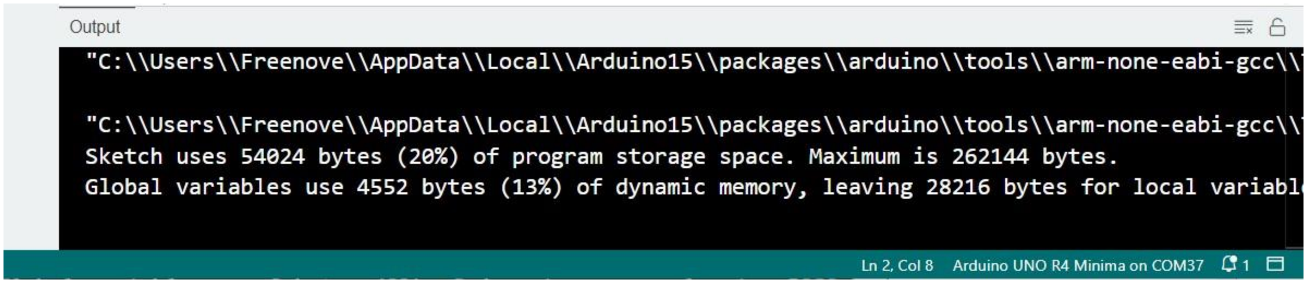
```
Compiling core...
"C:\\Users\\Freenove\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\arm-none-eabi-gcc\\7-2017
"C:\\Users\\Freenove\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\arm-none-eabi-gcc\\7-2017
"C:\\Users\\Freenove\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\arm-none-eabi-gcc\\7-2017
"C:\\Users\\Freenove\\AppData\\Local\\Arduino15
```

The following figure shows the code being compiled.



```
Generating function prototypes...
C:\Users\Freenove\AppData\Local\Arduino15\packages\arduino\tools\arm-none-eabi-gcc\7-2017q4/bi
C:\Users\Freenove\AppData\Local\Arduino15\packages\builtin\tools\ctags\5.8-arduino11/ctags -u
Compiling sketch...
"C:\\Users\\Freenove\\AppData\\Local\\Ardu
```

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.



```
"C:\\Users\\Freenove\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\arm-none-eabi-gcc\\

"C:\\Users\\Freenove\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\arm-none-eabi-gcc\\
Sketch uses 54024 bytes (20%) of program storage space. Maximum is 262144 bytes.
Global variables use 4552 bytes (13%) of dynamic memory, leaving 28216 bytes for local variabl
```

Ln 2, Col 8 Arduino UNO R4 Minima on COM37 1

Click "Upload" button.

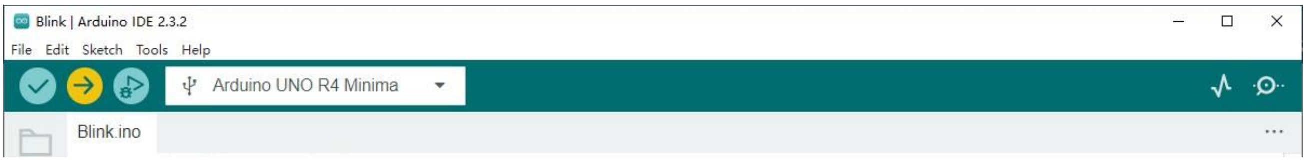
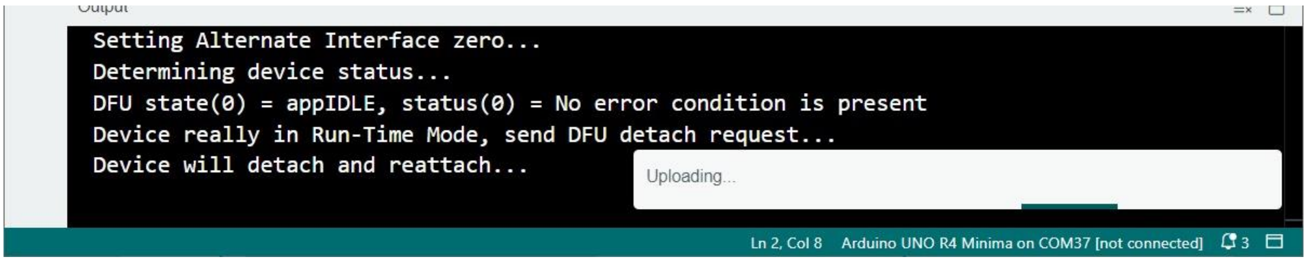


Figure below shows code is uploading.

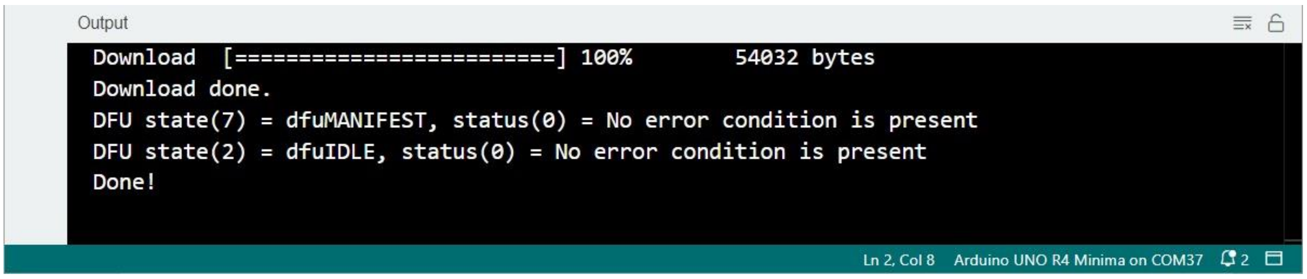


```
Setting Alternate Interface zero...
Determining device status...
DFU state(0) = appIDLE, status(0) = No error condition is present
Device really in Run-Time Mode, send DFU detach request...
Device will detach and reattach...
```

Uploading...

Ln 2, Col 8 Arduino UNO R4 Minima on COM37 [not connected] 3

Wait for the code to finish uploading.

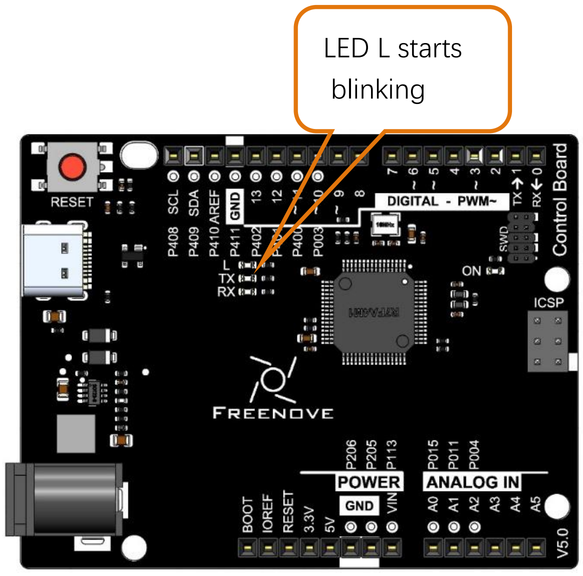


```
Download   [=========================] 100%        54032 bytes
Download done.
DFU state(7) = dfuMANIFEST, status(0) = No error condition is present
DFU state(2) = dfuIDLE, status(0) = No error condition is present
Done!
```

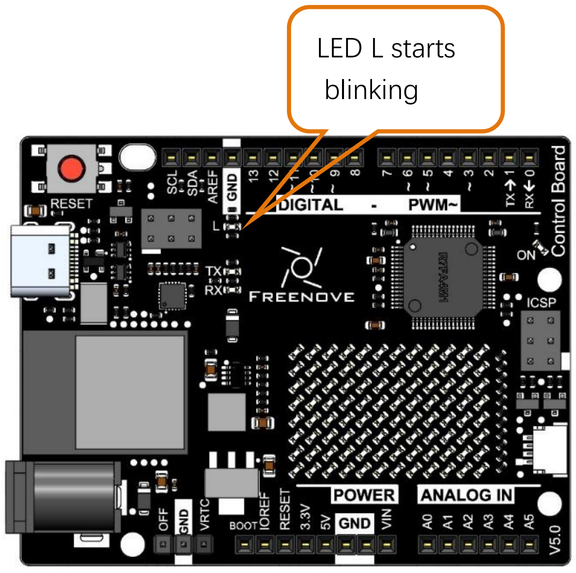
Ln 2, Col 8 Arduino UNO R4 Minima on COM37 2

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!



or



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.

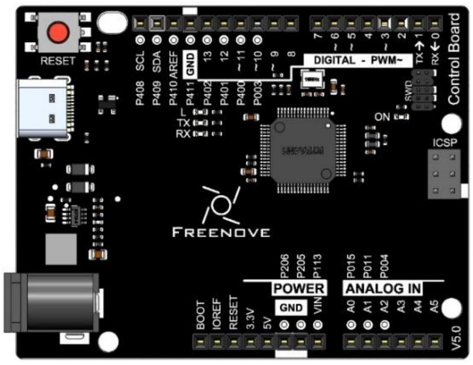
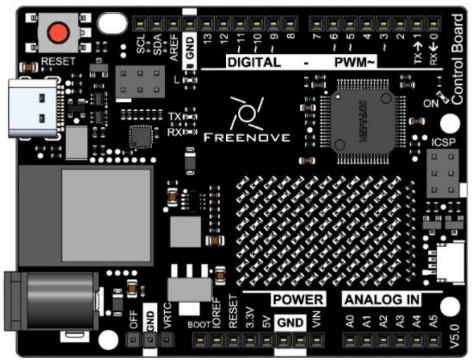
Chapter 1 LED Blink

Make the onboard LED marked "L" blink.

Project 1.1 Control the onboard LED

In this section, we will light up the an onboard LED.

Component List

Control board x1  or 
USB cable x1

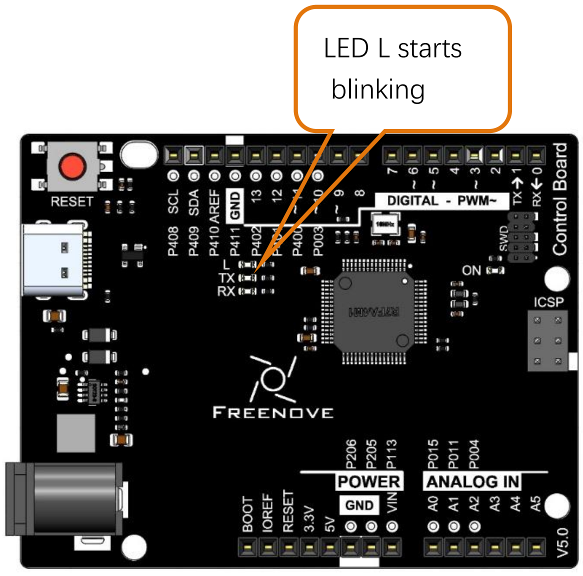
Sketch

The onboard LED is controlled by Pin13. When the pin outputs high, the LED lights up; when it outputs low, the LED is OFF.

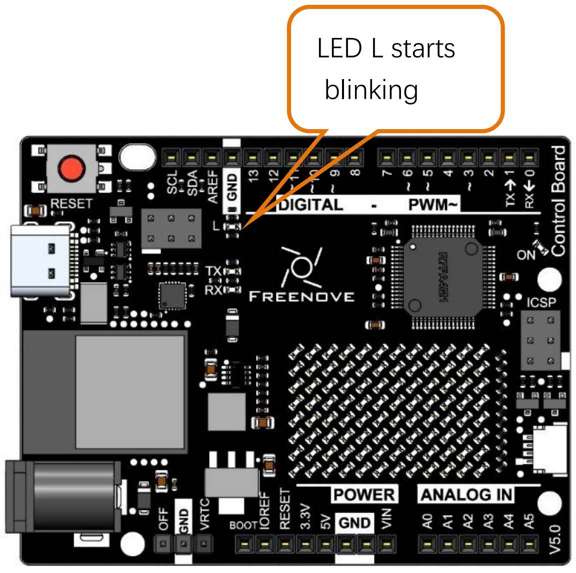
Sketch_01.1_Blink.

After the code uploads, the LED will blink at intervals of 1s.

```
void setup() {
  // initialize digital pin 13 as an output
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```



or



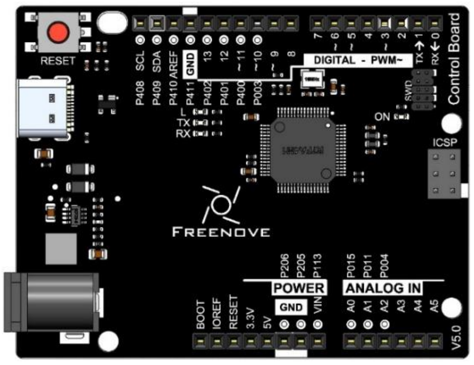
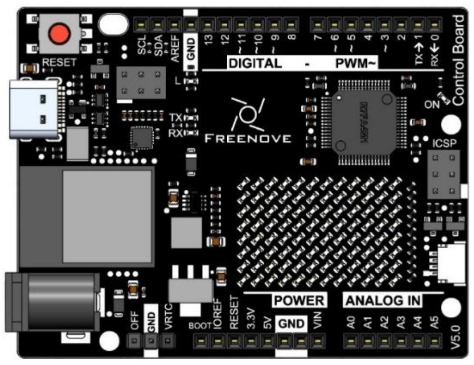
Chapter 2 Serial

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

Project 2.1 Send Data through Serial

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

Component List

Control board x1  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:

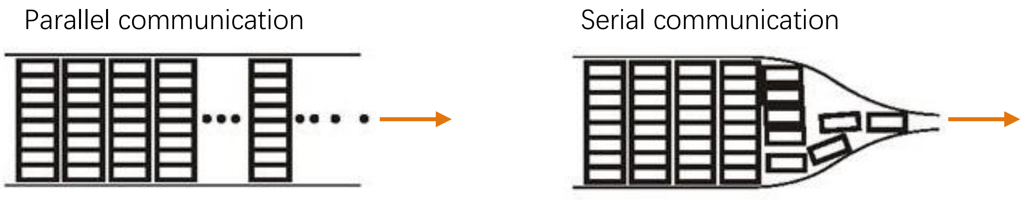
		Remainder	Sequence
2	150	………… 0	0
2	75	………… 1	1
2	37	………… 1	2
2	18	………… 0	3
2	9	………… 1	4
2	4	………… 0	5
2	2	………… 0	6
2	1	………… 1	7
	0		

The result is 10010110.

Circuit Knowledge

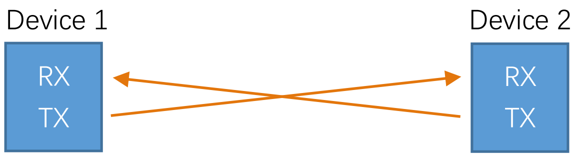
Serial and parallel communication

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:

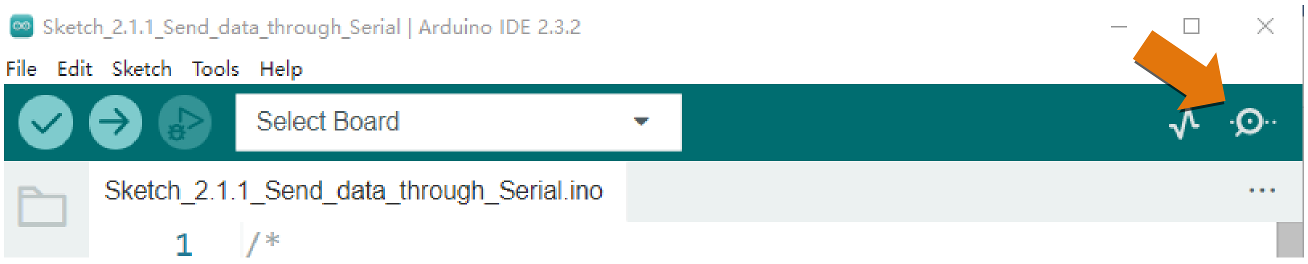


Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The baud rates commonly used are 9600 and 115200.

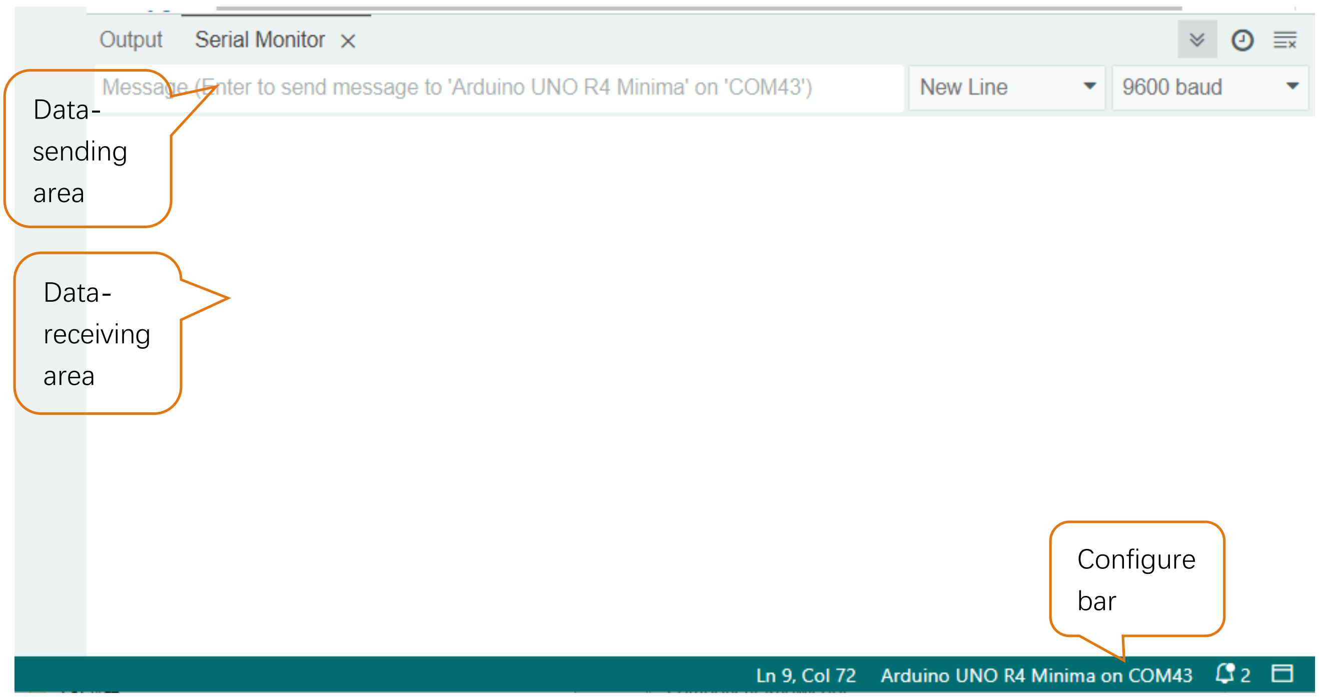
Serial port on Control board

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.

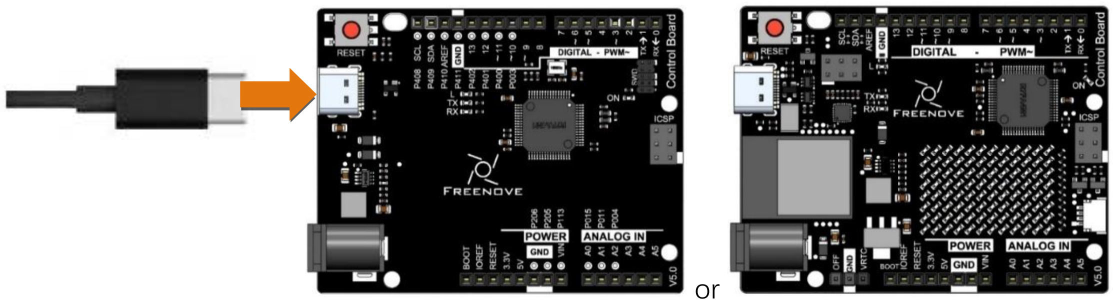

Sketch_2.1.1_Send_data_through_Serial | Arduino IDE 2.3.2
File Edit Sketch Tools Help
Select Board
Sketch_2.1.1_Send_data_through_Serial.ino
/*

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".


Output
Serial Monitor
Message (Enter to send message to 'Arduino UNO R4 Minima' on 'COM43')
New Line
9600 baud
Data-sending area
Data-receiving area
Configure bar
Ln 9, Col 72
Arduino UNO R4 Minima on COM43

Circuit

Connect control board to the computer with USB cable.



Sketch

Sketch 2.1.1

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

```
int counter = 0;  // define a variable as a data sending to serial port

void setup() {
  Serial.begin(9600);                  // initialize the serial port, set the baud rate to 9600
}

```

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

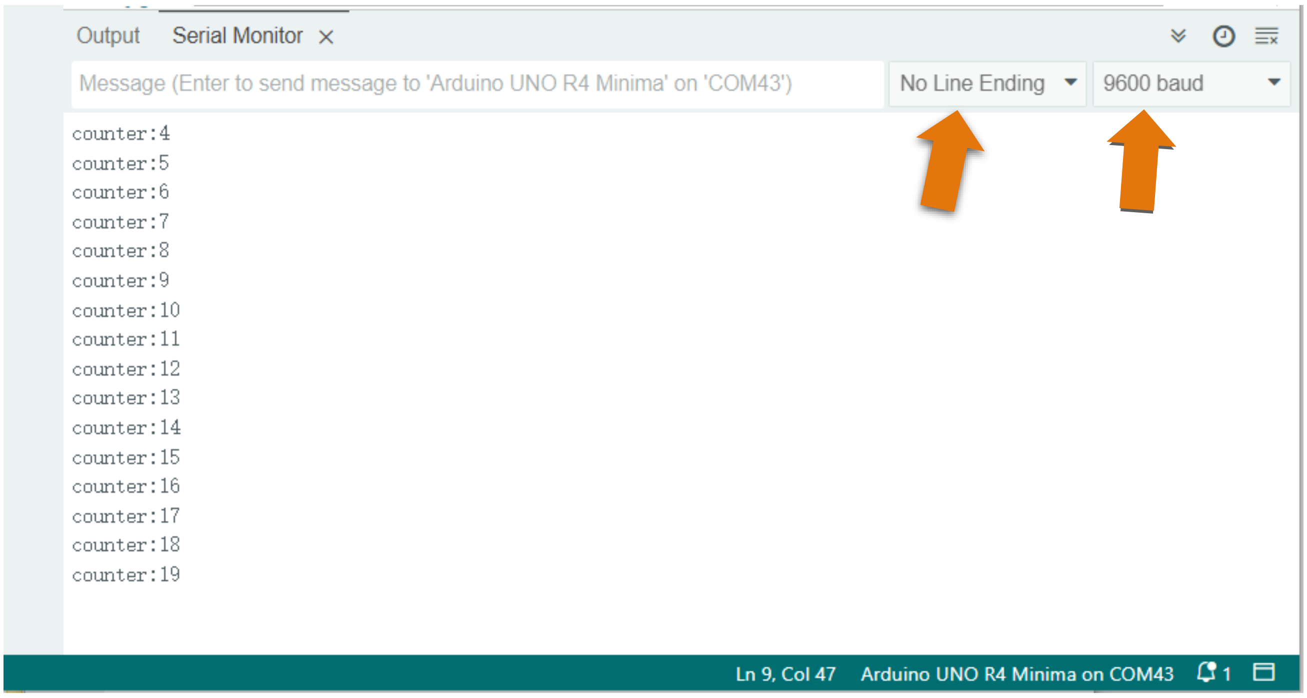
setup() function initializes the serial port.
And then continuously sends variable counter values in the loop () function.

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.


Output
Serial Monitor
Message (Enter to send message to 'Arduino UNO R4 Minima' on 'COM43')
No Line Ending
9600 baud
Ln 9, Col 47
Arduino UNO R4 Minima on COM43

```
counter:4
counter:5
counter:6
counter:7
counter:8
counter:9
counter:10
counter:11
counter:12
counter:13
counter:14
counter:15
counter:16
counter:17
counter:18
counter:19
```

Project 2.2 Receive Data through Serial Port

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

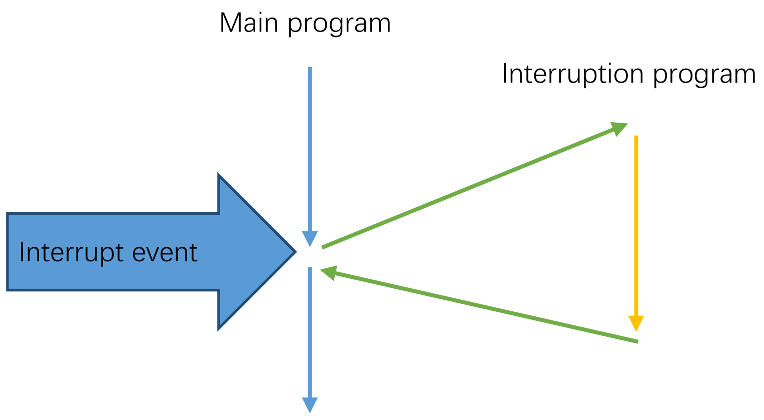
Component List

Same with the previous section.

Code Knowledge

Interrupt

An interrupt is a controller's response to an event. The event causing an interrupt is an interrupt source. We'll illustrate the interruption concept. For example, suppose you're watching TV while there is water in your kitchen heating, then you have to check whether the water is boiling or not from time to time, so you can't concentrate on watching TV. But if you have an interrupt, things will be different. Interrupt can work as a warning device for your kettle, which will beep when the water is about to boil. So before the water is boiling, you can focus on watching TV until a beep warning comes out.
Advantages of interrupt here: Processor won't need to check whether the event has happened every now and then, but when an event occurs, it informs the controller immediately. When an interrupt occurs, the processor will jump to the interrupt function to handle interrupt events, then return to where the interrupt occurs after finishing it and go on this program.



Circuit

Same with the previous section.

Sketch

Sketch 2.2.1

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

```
char inChar;      // define a variable to store characters received from serial port

```

```
void setup() {
  Serial.begin(9600);               // initialize serial port, set baud rate to 9600
}

void loop() {
  if (Serial.available()) {         // judge whether data has been received
    inChar = Serial.read();         // read one character
    Serial.print("received:");      // print the string "received:"
    Serial.println(inChar);         // print the received character
  }
}
```

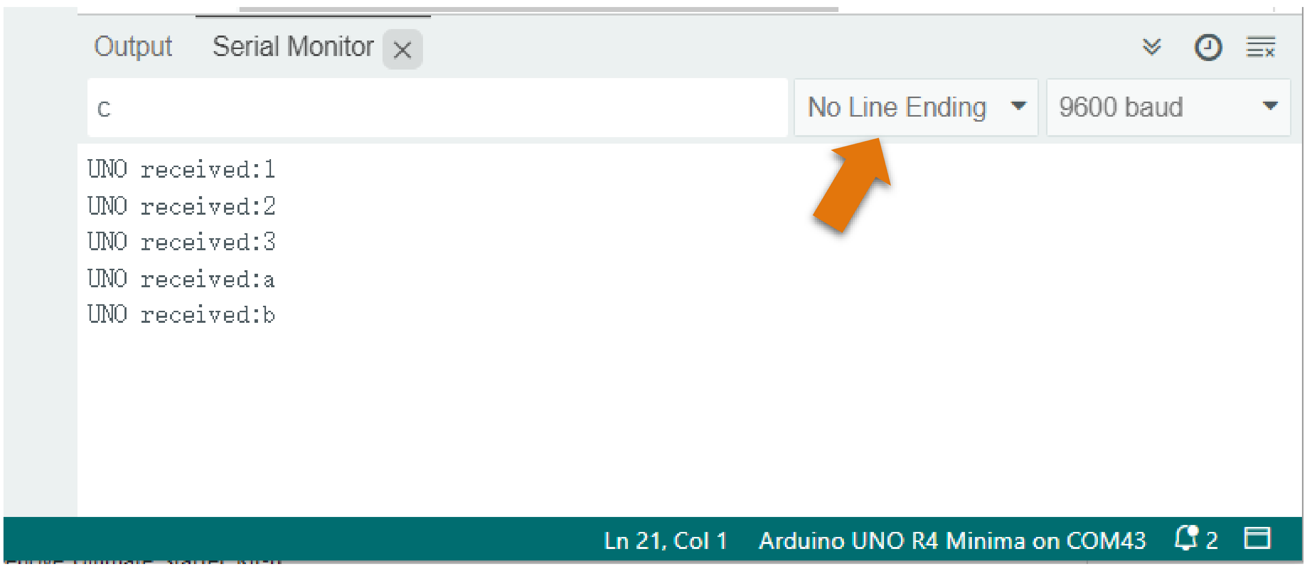
In the setup() function, we initialize the serial port. Then, the loop() function will continuously detect whether there are data to read. if so, it will read the character and send it back.

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.



```
c
```

```
UNO received:1
UNO received:2
UNO received:3
UNO received:a
UNO received:b
```

char type

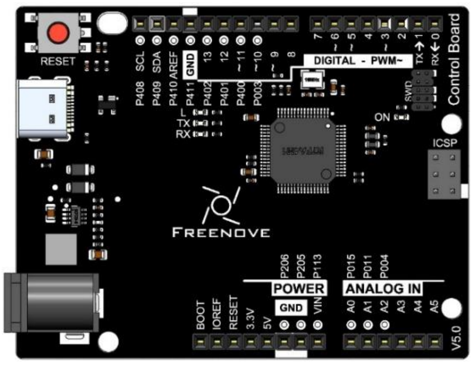
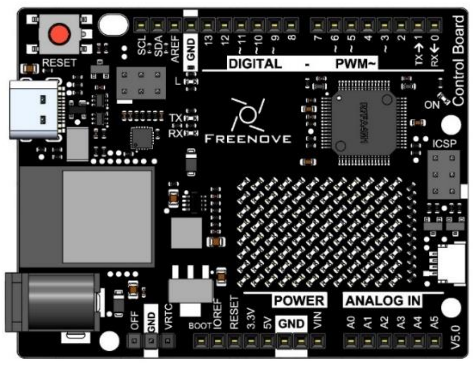
char type variable can represent a character, but it cannot store characters directly. It stores numbers to replace characters. char type occupies 1-byte store area, and use a value 0-127 to correspond to 128 characters. The corresponding relation between number and character is ruled by ASCII table. For more details of ASCII table, please refer to the appendix of this book.

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

Project 2.3 Application of Serial

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

Component List

Control board x1
 or 
USB cable x1

Sketch

Sketch 2.3.1

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

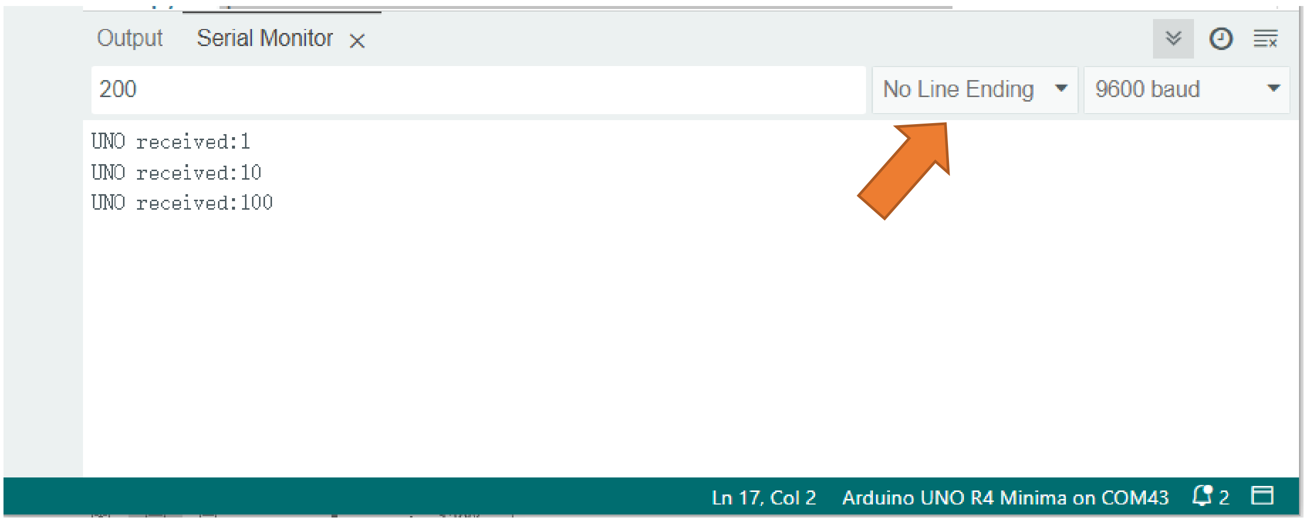
```
int inInt;        // define a variable to store the data received from serial
int counter = 0;  // define a variable as the data sending to serial
int ledPin = 13;  // the number of the LED pin

void setup() {
  pinMode(ledPin, OUTPUT);            // initialize the LED pin as an output
  Serial.begin(9600);                 // initialize serial port, set baud rate to 9600
}

void loop() {
  if (Serial.available()) {         // judge whether the data has been received
    inInt = Serial.parseInt();      // read an integer
    Serial.print("received:");      // print the string " 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.



200
No Line Ending
9600 baud

```
UNO received:1
UNO received:10
UNO received:100
```

Ln 17, Col 2   Arduino UNO R4 Minima on COM43

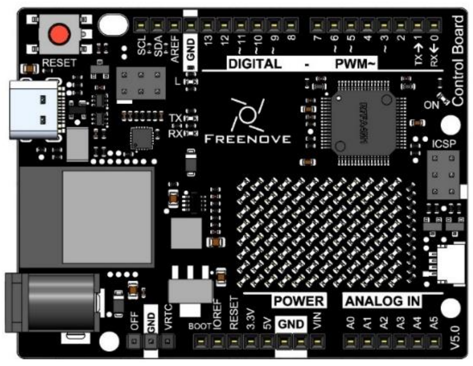
Chapter 3 Using the Onboard LED Matrix

The control board has a built-in 12x8 LED matrix, which can be programed to display graphics, animate, act as an interface, and even play games.

Project 3.1 LED Matrix

For this project, we will turn ON the LED matrix and turn it off.

Component List

Control board x1	USB cable x1
	

Sketch

Sketch 3.1

Upload the sketch to the control board and you'll see all LEDs of the matrix light up simultaneously and turn them off after one second.

The following is the program code:

```
#include "Arduino_LED_Matrix.h"  // Include the LED_Matrix library
ArduinoLEDMatrix matrix;  // Create an instance of the ArduinoLEDMatrix class
void setup() {
  matrix.begin();  // Initialize the LED matrix
}

const uint32_t fullOn[] = {
  0xffffffff,
  0xffffffff,
  0xffffffff
};
```

```
const uint32_t fullOff[] = {
  0x00000000,
  0x00000000,
  0x00000000
};
void loop() {
  matrix.loadFrame(fullOff);
  delay(250);
  matrix.loadFrame(fullOn);
  delay(250);
}
```

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

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

Create an object for the LED matric in the sketch.

```
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.

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

the main function, light up all LEDs of the matrix and turn them off.

```
void loop() {
  matrix.loadFrame(fullOff);
  delay(250);
  matrix.loadFrame(fullOn);
  delay(250);
}
```

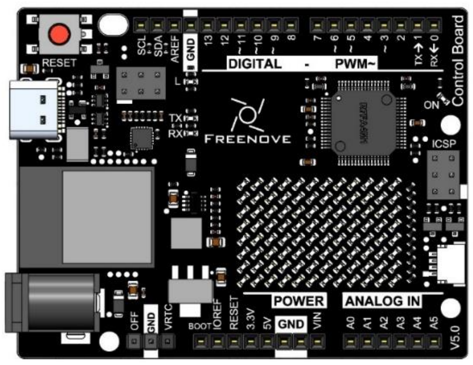
Chapter 4 Control LED with Web

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

Project 4.1 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

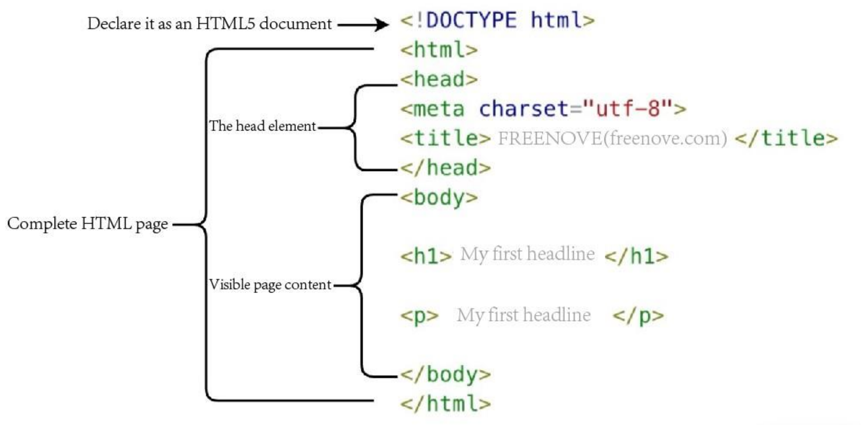
HTML

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>
<html>
<head>
<meta charset="utf-8">
<title> FREENOVE(freenove.com) </title>
</head>
<body>

<h1> My first headline </h1>

<p>  My first headline  </p>

</body>
</html>
```

<!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 &lt; meta charset="utf-8"&gt; Define the web pa ge encoding format to UTF-8.

<title>: Notesthe title of the document

<body>: Contains visible page content

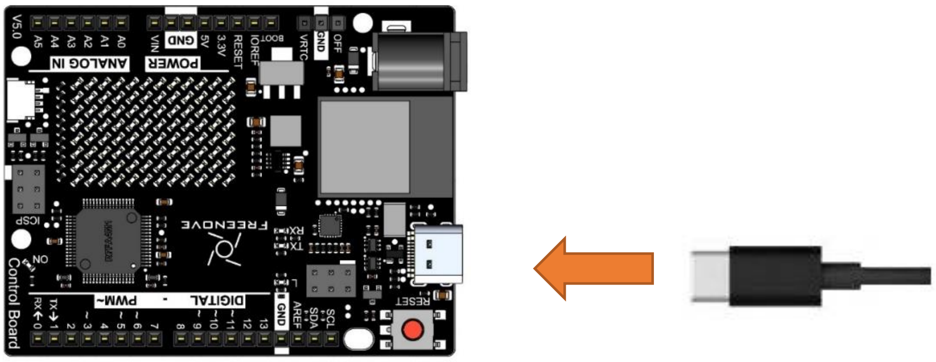
<h1>: Define a big heading

<p>: Define a paragraph

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

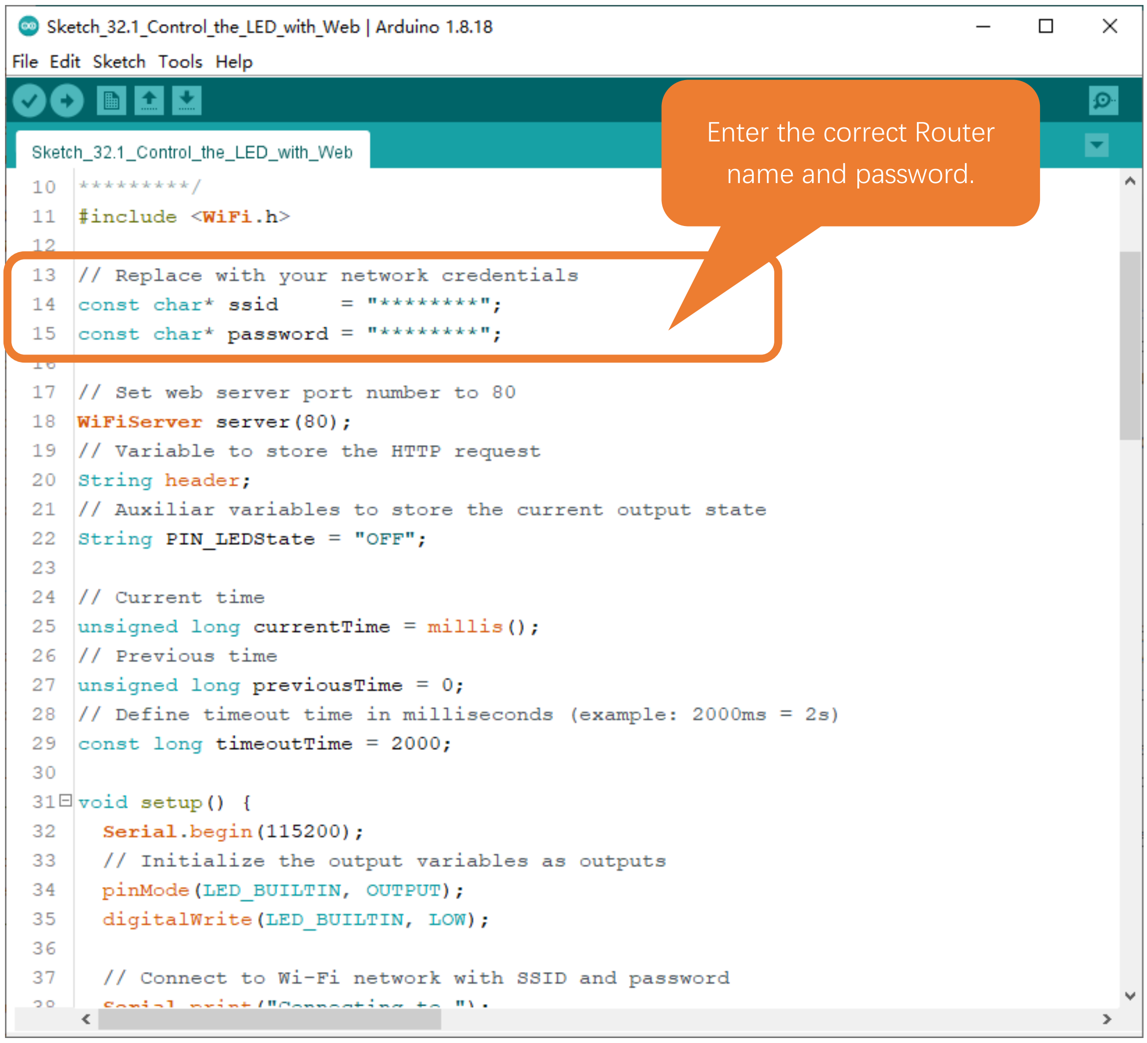
Circuit

Connect the board to the computer using the USB cable.



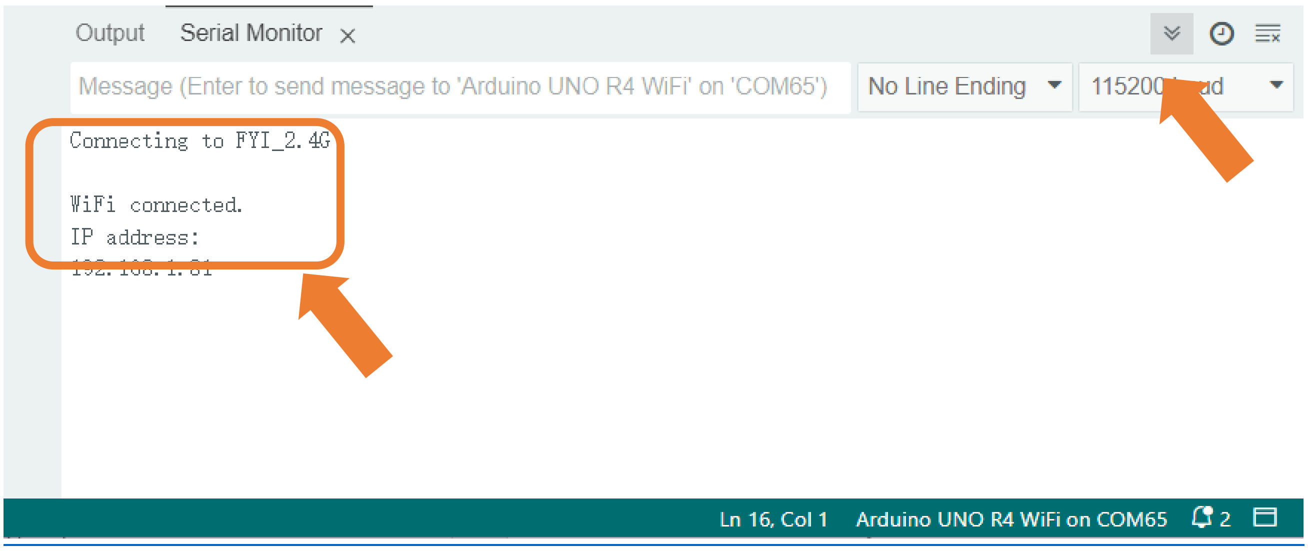
Sketch

Sketch_32.1_Control_the_LED_with_Web


Sketch_32.1_Control_the_LED_with_Web | Arduino 1.8.18
File Edit Sketch Tools Help
Sketch_32.1_Control_the_LED_with_Web
Enter the correct Router name and password.

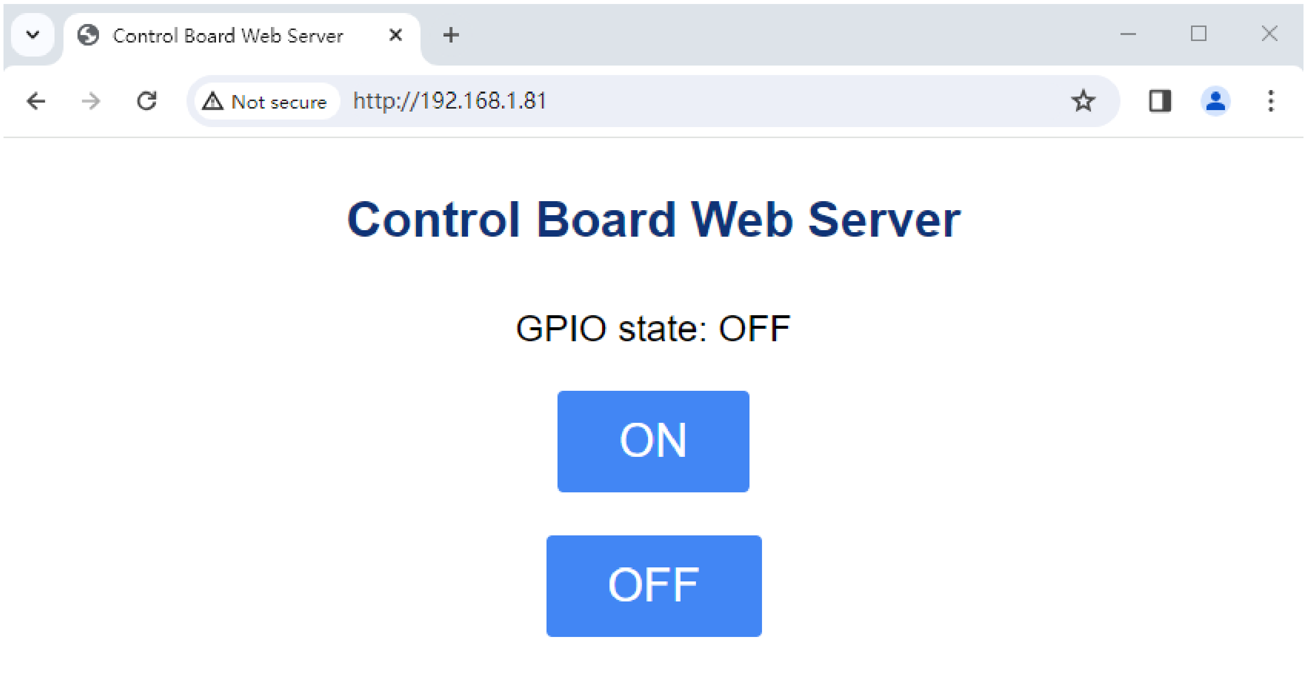
```
*********/
#include <WiFi.h>

// Replace with your network credentials
const char* ssid     = "********";
const char* password = "********";

// Set web server port number to 80
WiFiServer server(80);
// Variable to store the HTTP request
String header;
// Auxiliar variables to store the current output state
String PIN_LEDState = "OFF";

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // Connect to Wi-Fi network with SSID and password
```

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.


Output
Serial Monitor
Message (Enter to send message to 'Arduino UNO R4 WiFi' on 'COM65')
No Line Ending
115200 baud
Ln 16, Col 1
Arduino UNO R4 WiFi on COM65

```
Connecting to FYI_2.4G

WiFi connected.
IP address:
192.168.1.81
```

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 <WiFi.h>

// Replace with your network credentials
const char* ssid     = "********";
const char* password = "********";

```

```
// Set web server port number to 80
WiFiServer server(80);
// Variable to store the HTTP request
String header;
// Auxiliar variables to store the current output state
String PIN_LEDState = "OFF";

// Current time
unsigned long currentTime = millis();
// Previous time
unsigned long previousTime = 0;
// Define timeout time in milliseconds (example: 2000ms = 2s)
const long timeoutTime = 2000;

void setup() {
  Serial.begin(115200);
  // Initialize the output variables as outputs
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  // Connect to Wi-Fi network with SSID and password
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // Print local IP address and start web server
  Serial.println("");
  Serial.println("WiFi connected.");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
void loop() {
  WiFiClient client = server.available();  // Listen for incoming clients
  if (client) {                            // If a new client connects,
    Serial.println("New Client.");         // print a message out in the serial port
    String currentLine = "";               // make a String to hold incoming data from the
client
    currentTime = millis();
    previousTime = currentTime;

    while (client.connected() && currentTime - previousTime <= timeoutTime) {  // loop while the client's connected
      currentTime = millis();
      if (client.available()) {  // if there's bytes to read from the client,
        char c = client.read();  // read a byte, then
        Serial.write(c);         // print it out the serial monitor
        header += c;
        if (c == '\n') {  // if the byte is a newline character
          // if the current line is blank, you got two newline characters in a row.
          // that's the end of the client HTTP request, so send a response:
          if (currentLine.length() == 0) {
            // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
            // and a content-type so the client knows what's coming, then a blank line:
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();
            // turns the GPIOs on and off
            if (header.indexOf("GET /LED_BUILTIN/ON") >= 0) {
              Serial.println("LED_BUILTIN ON");
              PIN_LEDState = "ON";
              digitalWrite(LED_BUILTIN, HIGH);
            } else if (header.indexOf("GET /LED_BUILTIN/OFF") >= 0) {
              Serial.println("LED_BUILTIN OFF");
              PIN_LEDState = "OFF";
              digitalWrite(LED_BUILTIN, LOW);
            }
            // Display the HTML web page
            client.println("<!DOCTYPE html><html>");
            client.println("<head> <title>Control Board Web Server</title> <meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            // CSS to style the on/off buttons
            // Feel free to change the background-color and font-size attributes to fit your preferences
            client.println("<style>html {font-family: Helvetica; display:inline-block; margin: 0px auto; text-align: center;}");
            client.println(" h1{color: #0F3376; padding: 2vh;} p{font-size: 1.5rem;}");
            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;}");
            client.println(".button2{background-color: #4286f4;display: inline-block; border: none; border-radius: 4px; color: white; padding: 16px 40px;text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}</style></head>");
            // Web Page Heading
            client.println("<body><h1>Control Board Web Server</h1>");
            client.println("<p>GPIO state: " + PIN_LEDState + "</p>");
            client.println("<p><a href=\"/LED_BUILTIN/ON\"><button class=\"button button2\">ON</button></a></p>");
            client.println("<p><a href=\"/LED_BUILTIN/OFF\"><button class=\"button button2\">OFF</button></a></p>");
            client.println("</body></html>");
            // The HTTP response ends with another blank line
            client.println();
            // Break out of the while loop
            break;
          } else {  // if you got a newline, then clear currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // if you got anything else but a carriage return character,
          currentLine += c;      // add it to the end of the currentLine
        }
      }
    }
    // Clear the header variable
    header = "";
    // Close the connection
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Include the WiFi Library header file of Control Board.

```
#include <WiFi.h>
```

Enter correct router name and password.

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

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

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

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());
```

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

```
            // turns the GPIOs on and off
            if (header.indexOf("GET /LED_BUILTIN/ON") >= 0) {
              Serial.println("LED_BUILTIN ON");
              PIN_LEDState = "ON";
              digitalWrite(LED_BUILTIN, HIGH);
            } else if (header.indexOf("GET /LED_BUILTIN/OFF") >= 0) {
              Serial.println("LED_BUILTIN OFF");
              PIN_LEDState = "OFF";
              digitalWrite(LED_BUILTIN, LOW);
            }
```

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	□