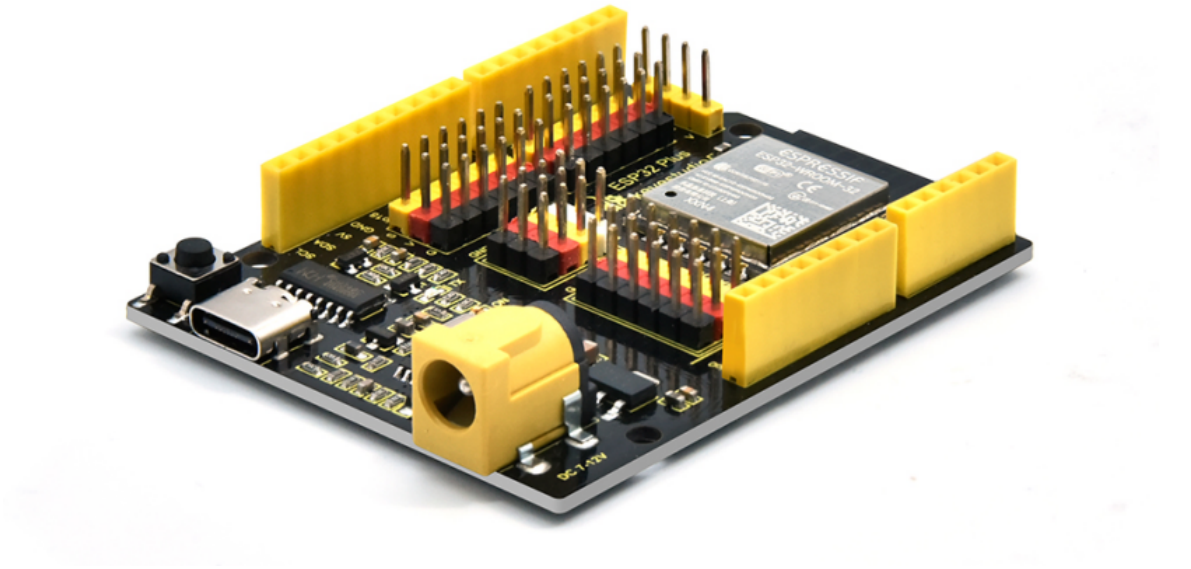
Arduino tutorial

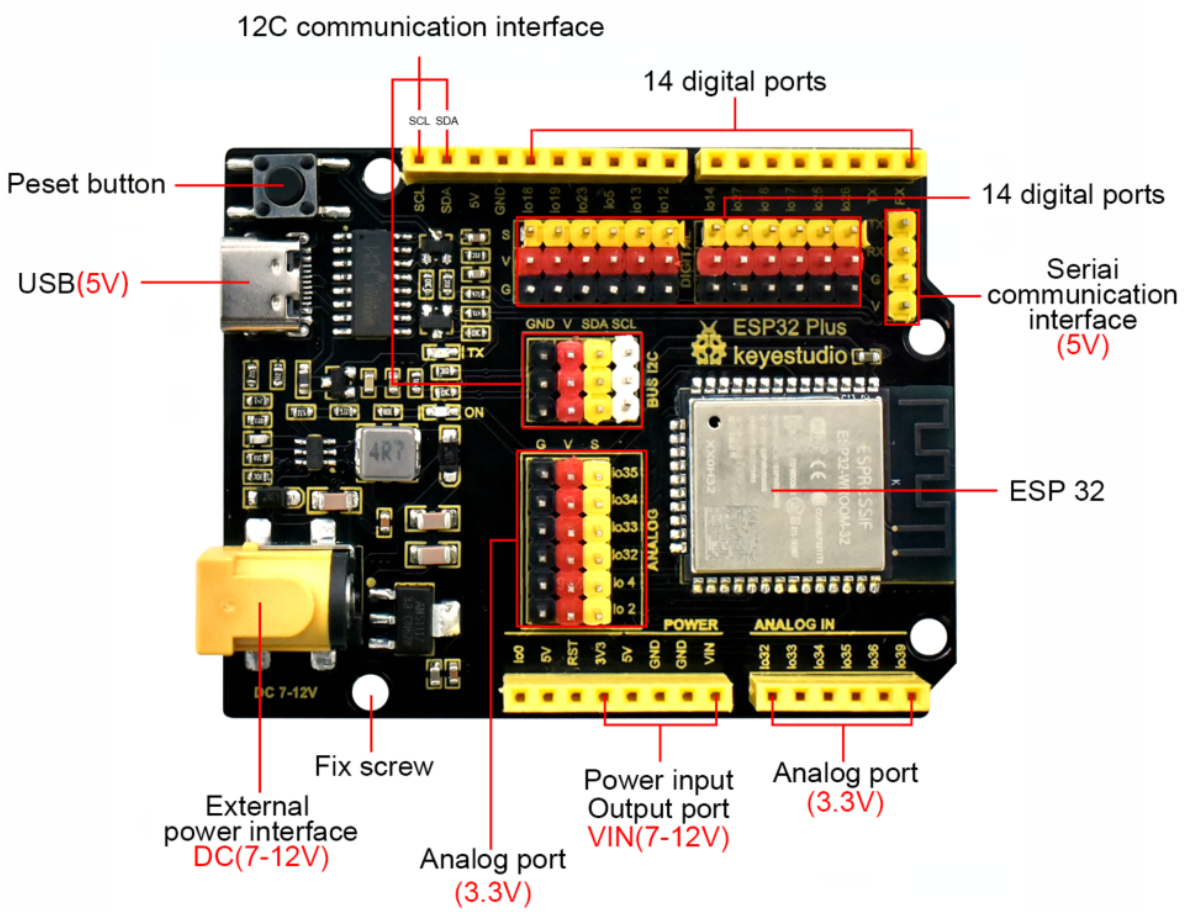
Getting started with Arduino

1. ESP32 PLUS Development board



ESP32PLUS is a universal WIFI plus Bluetooth development board based on ESP32, integrated with ESP32-WOROOM-32 module and compatible with Arduino.

It has a hall sensor, high-speed SDIO/SPI, UART, I2S as well as I2C. Furthermore, equipped with freeRTOS operating system, which is quite suitable for the Internet of things and smart home.

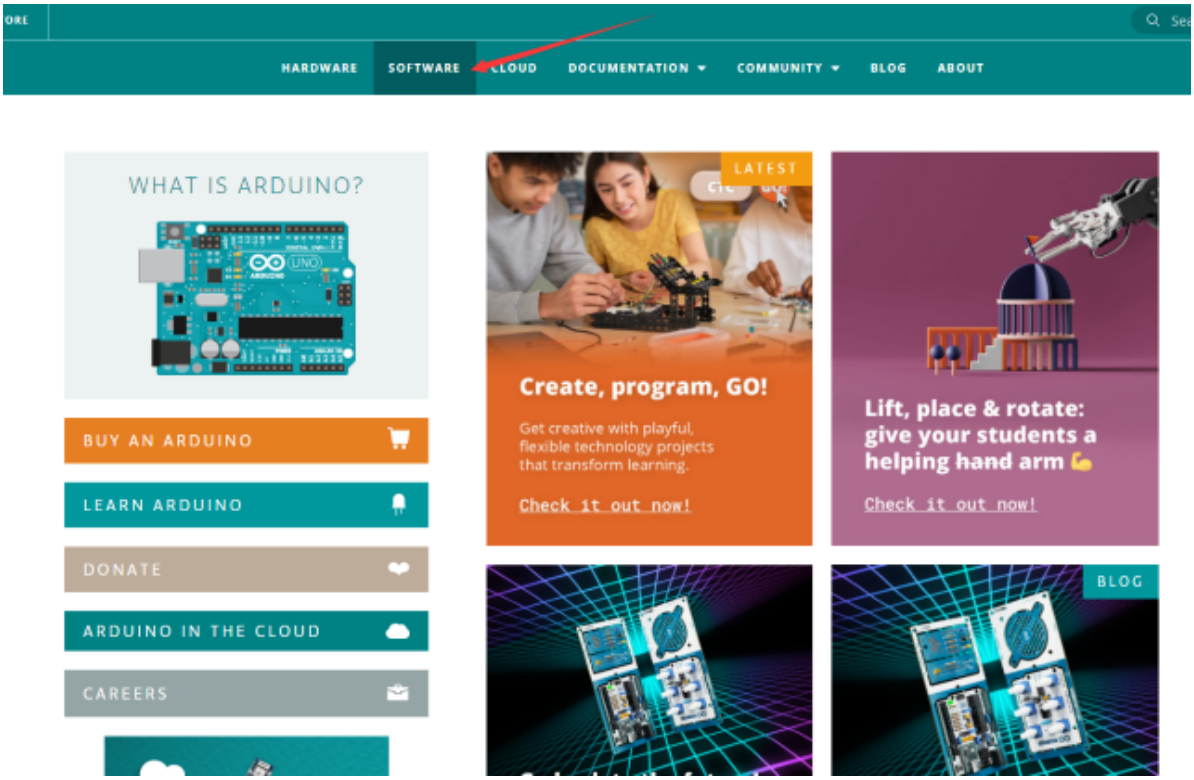


2. Windows System

2.1 Installing Arduino IDE

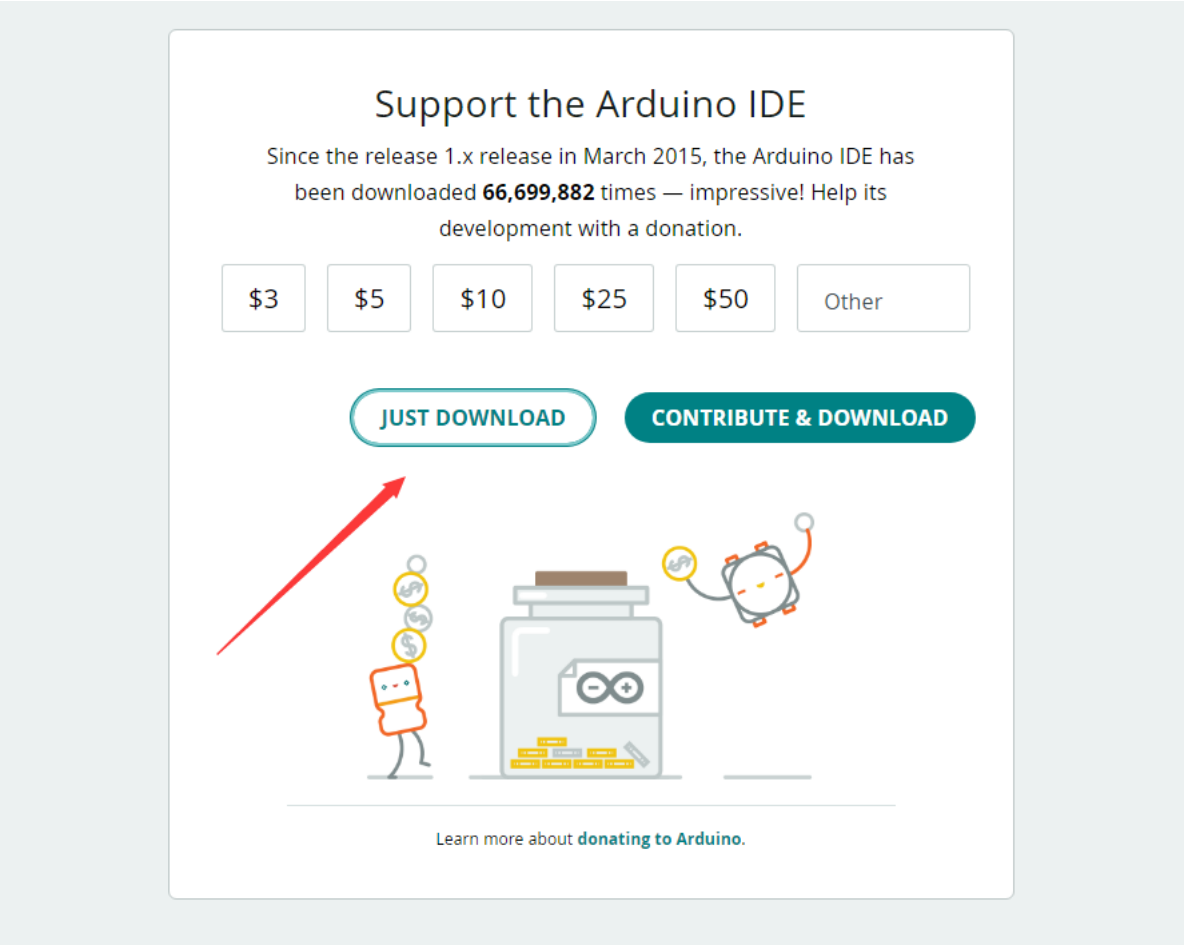
When you get control board, you need to download Arduino IDE and driver firstly.

You could download Arduino IDE from the official website:https://www.arduino.cc/, click the **SOFTWARE** on the browse bar to enter download page, as shown below:



There are various versions of IDE for Arduino. Just download a version compatible with your system. Here we will show you how to download and install the windows version of Arduino IDE.

You can choose between the Installer (.exe) and the Zip packages. We suggest you use the first one that installs directly everything you need to use the Arduino Software (IDE), including the drivers. With the Zip package you need to install the drivers manually. The Zip file is also useful if you want to create a portable installation.



You just need to click JUST DOWNLOAD.

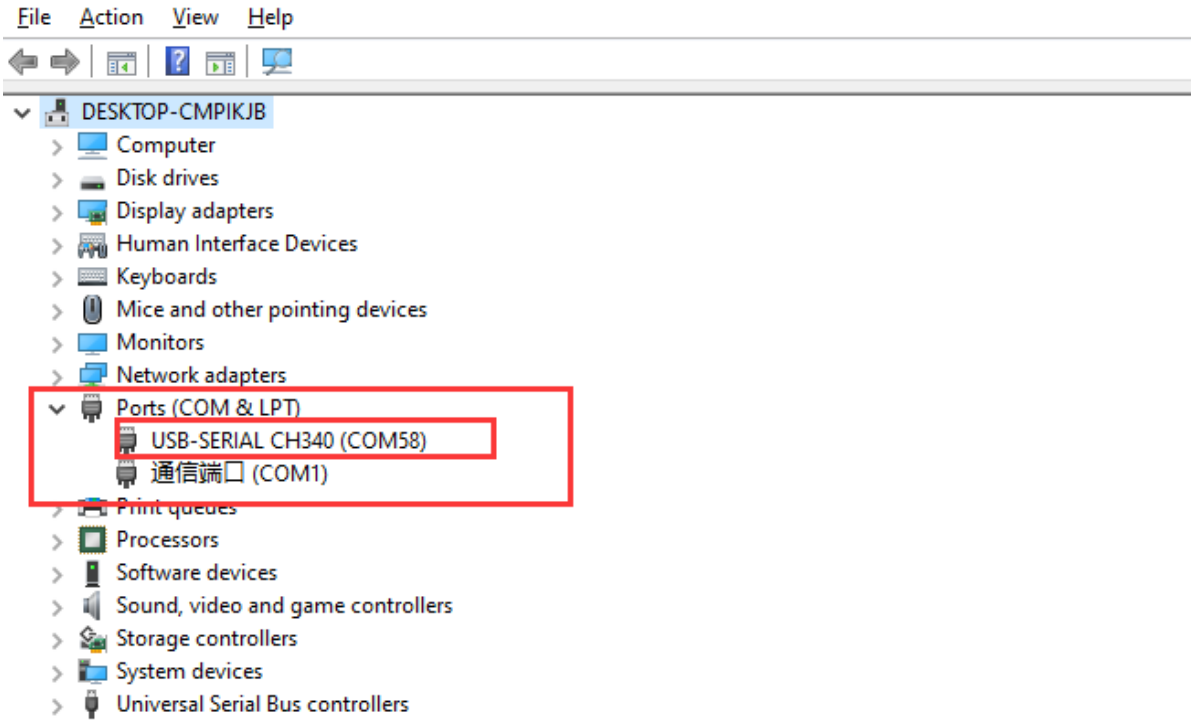
2.2 Install a driver

If you have installed the CH340 driver, just skip it.

Connect the main control board to your computer with a USB cable, and the driver will be installed automatically on MacOS and Windows10 system. If the driver installation process fails, you need to install the driver manually.

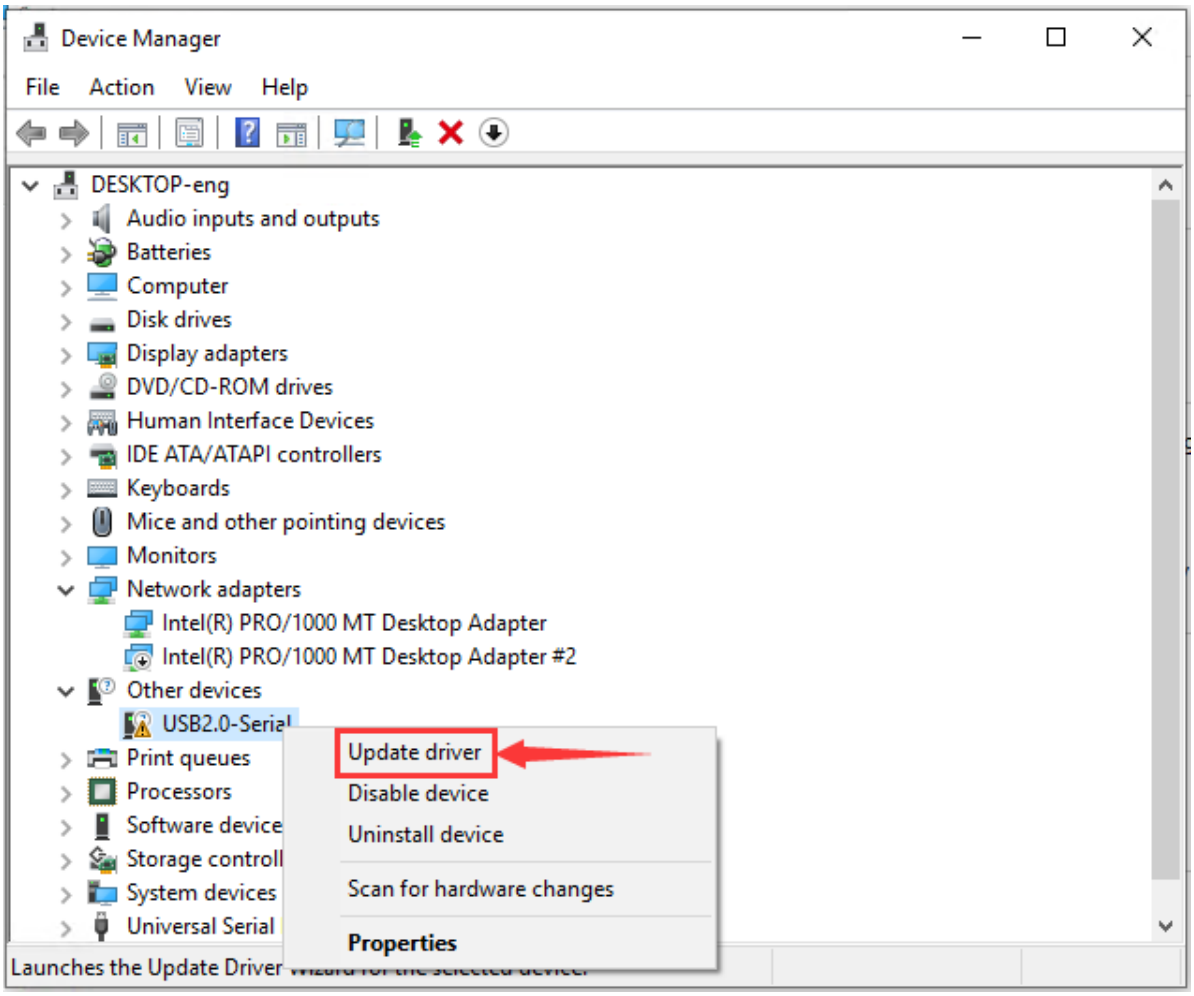
(1) Check whether the computer automatically installs the driver:

Right click Computer----- Click Properties-----Click Device Manager, the following picture shows the successful installation:

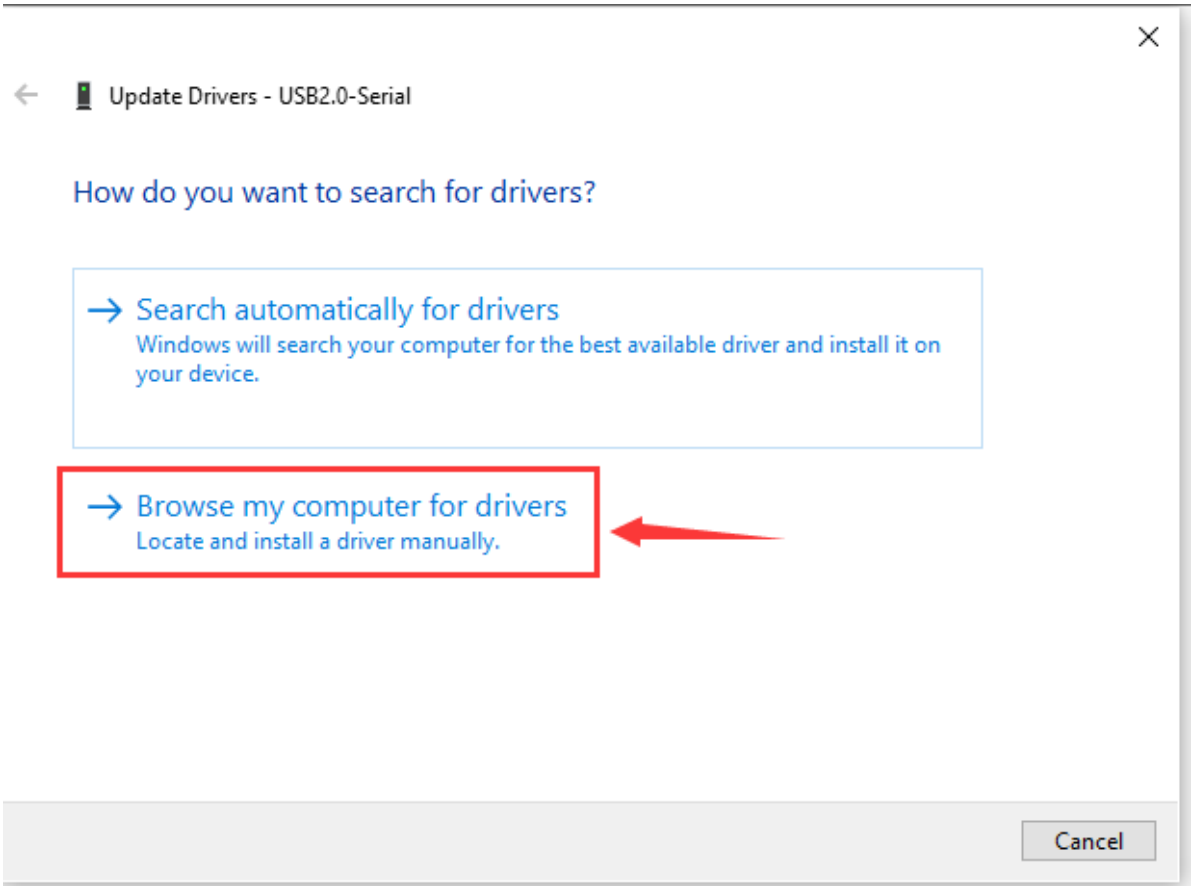


(2) Manual installation:

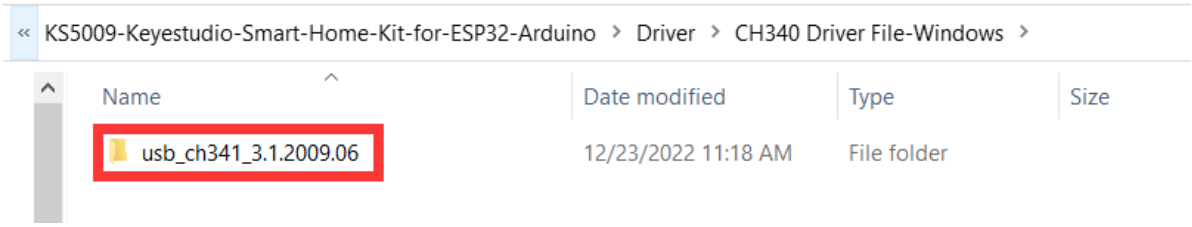
Right-click "**USB2.0-Serial**" and click "**Update drive...**"



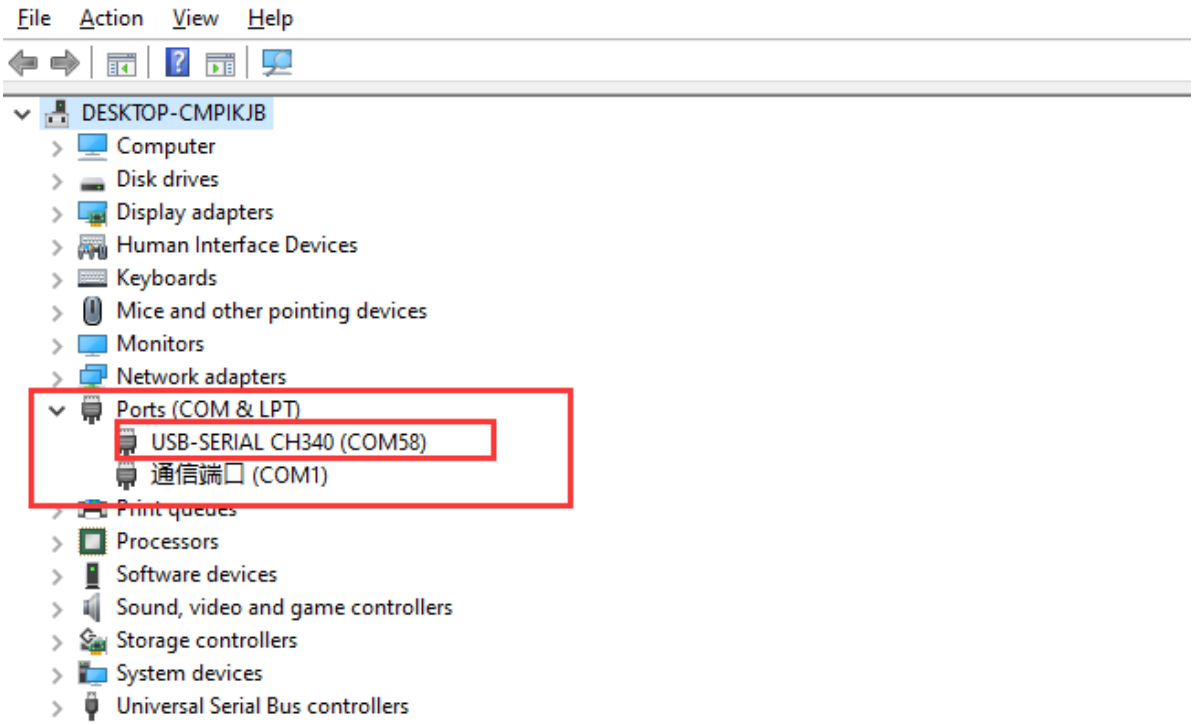
Click "**Browse my computer for driver software**"



Click“**Browse...**”and select the“**usb_ch341_3.1.2009.06 folder**”.

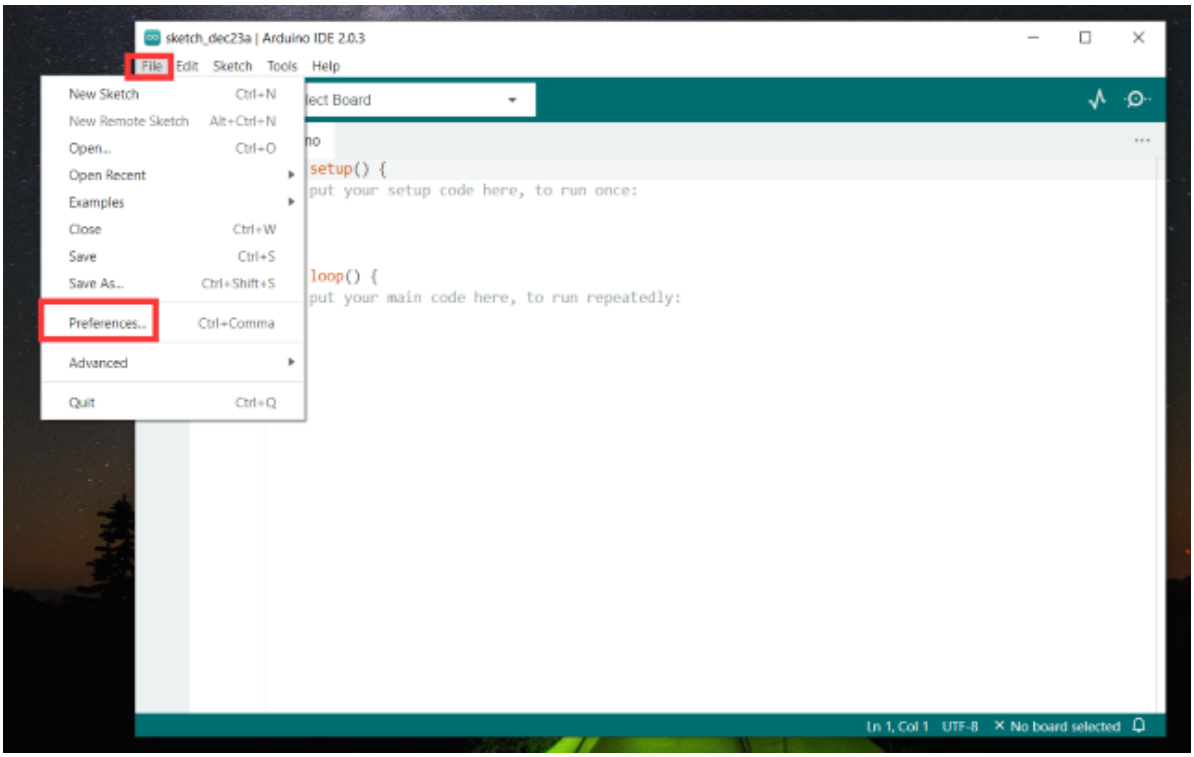


Check the serial port connection status again, as shown in the following figure, the driver is successfully installed.



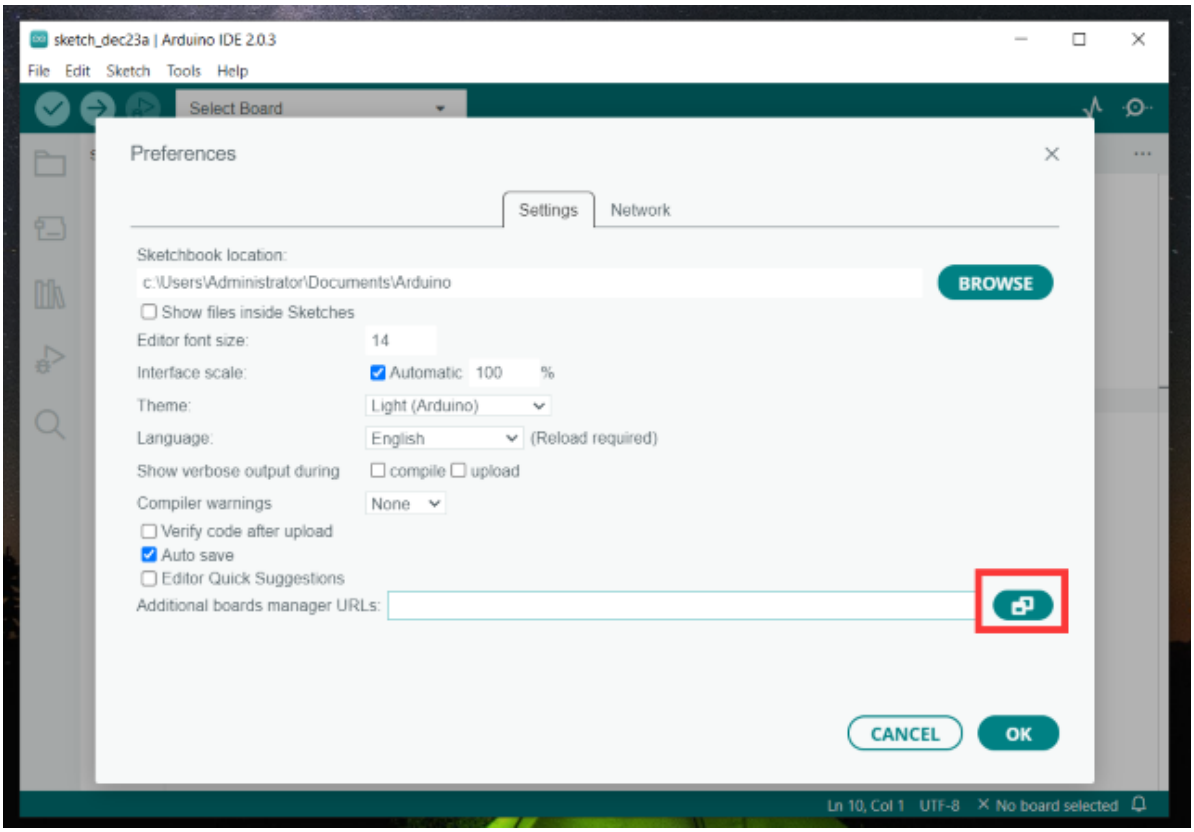
2.3 Add the ESP32 Environment(add version 3

(1) Open the arduino IDE，click File > Preferences，as shown below:

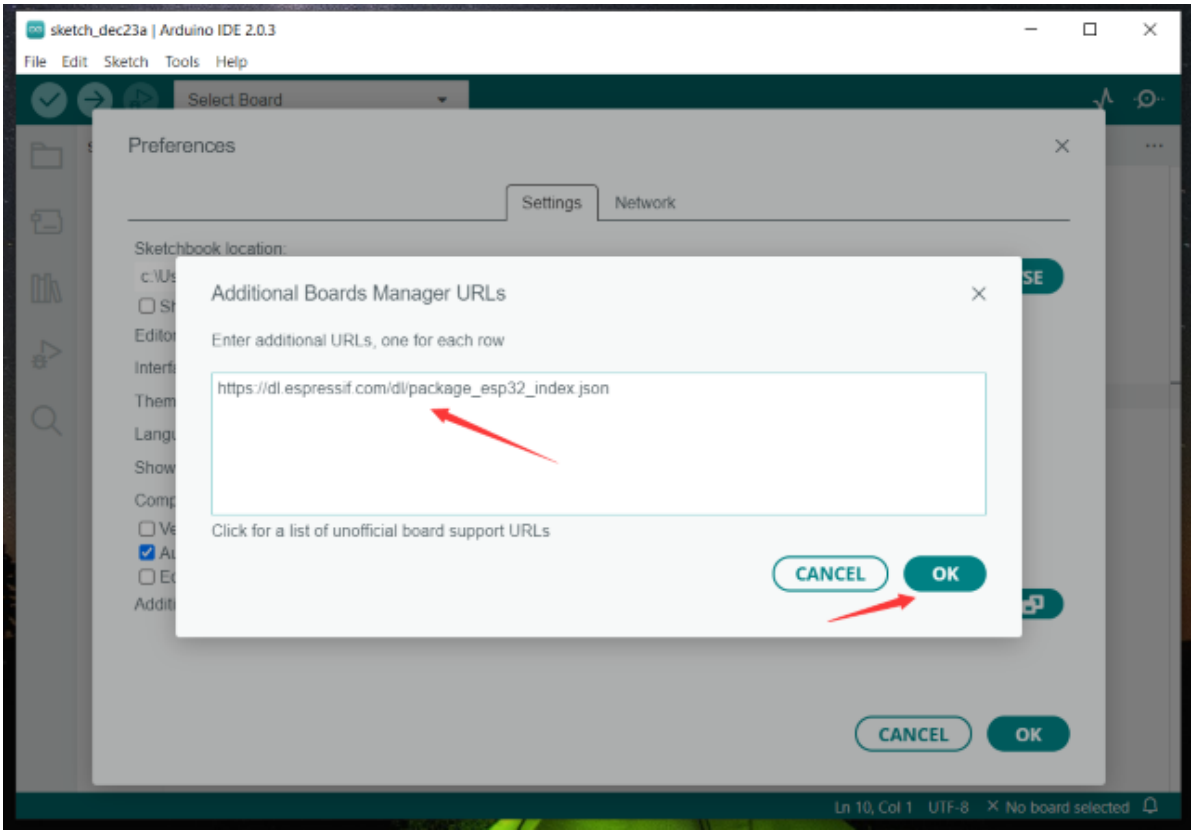


(2) Copy the link：https://dl.espressif.com/dl/package_esp32_index.json

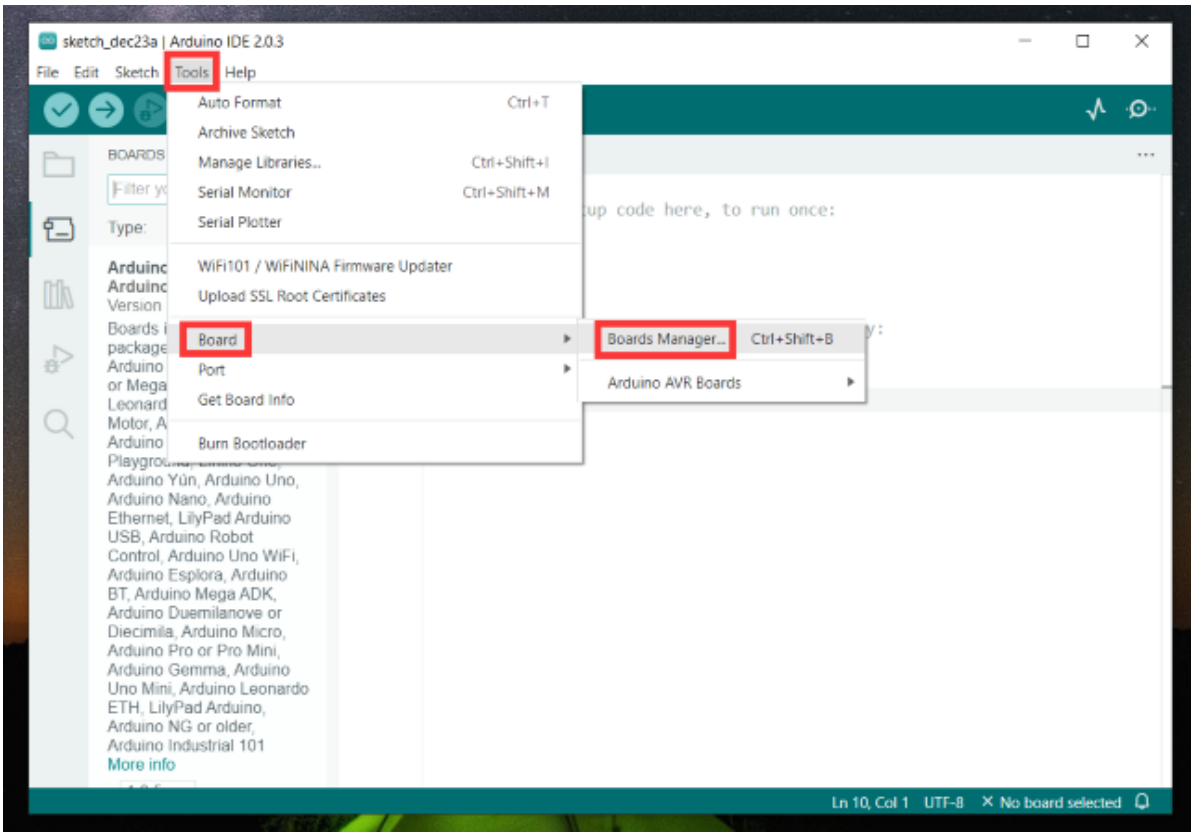
(3) Open the button marked below:



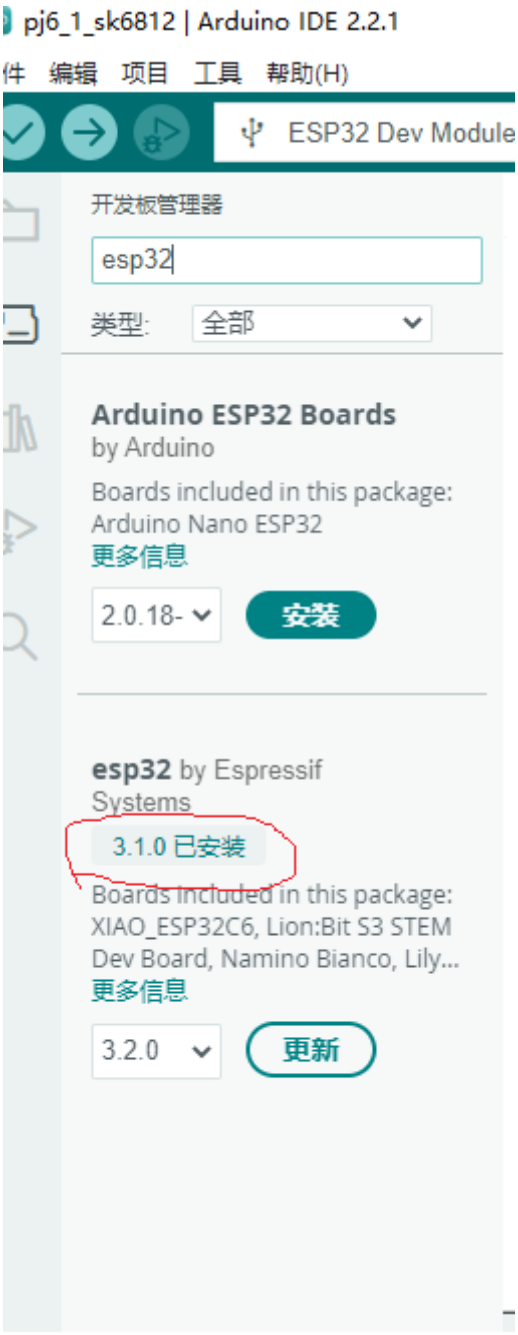
(4) Paste it inside and click OK, as shown below



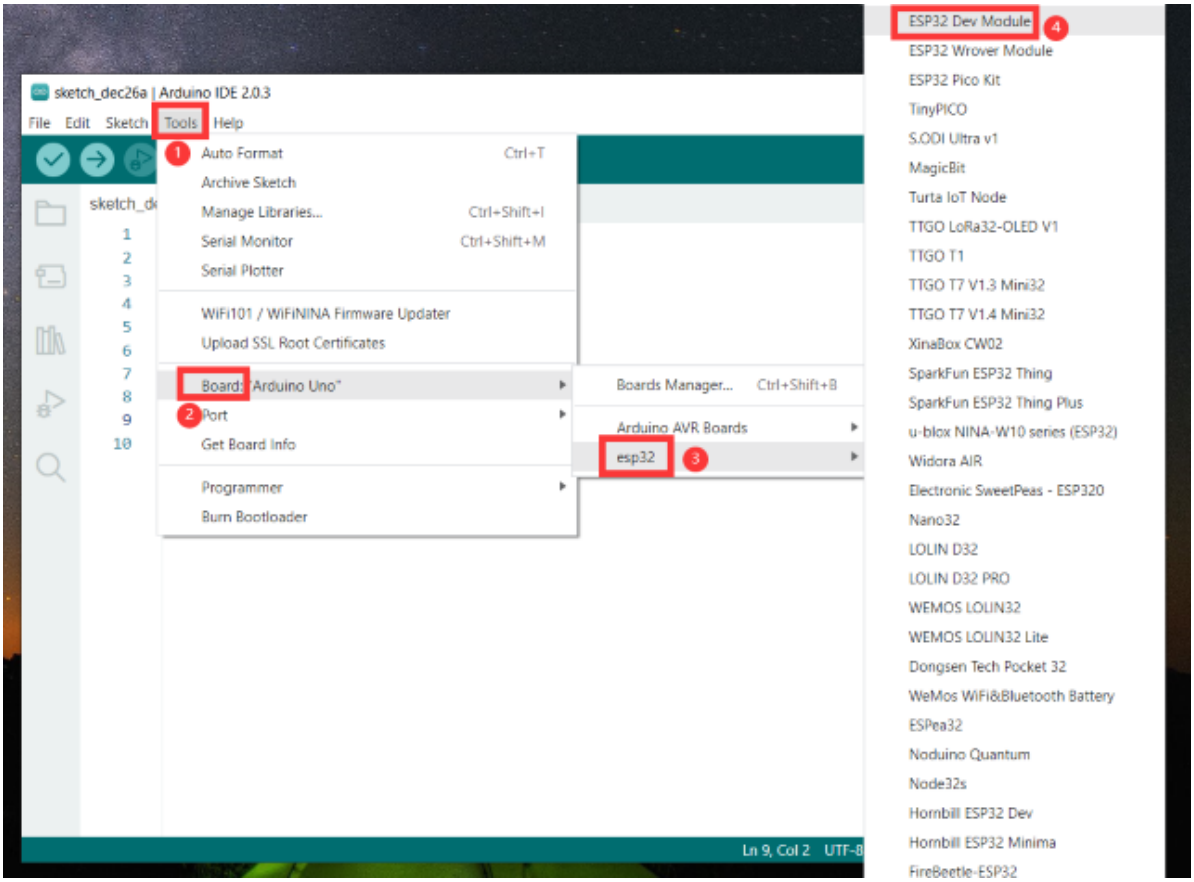
(5) Click Tools > Board > Boards Manager



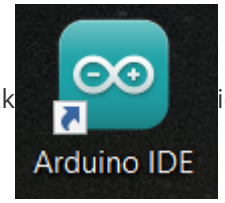
(6) Find the ESP32 from the pop up Boards Manager and then click install. (add version 3.1.0)!!!Very important



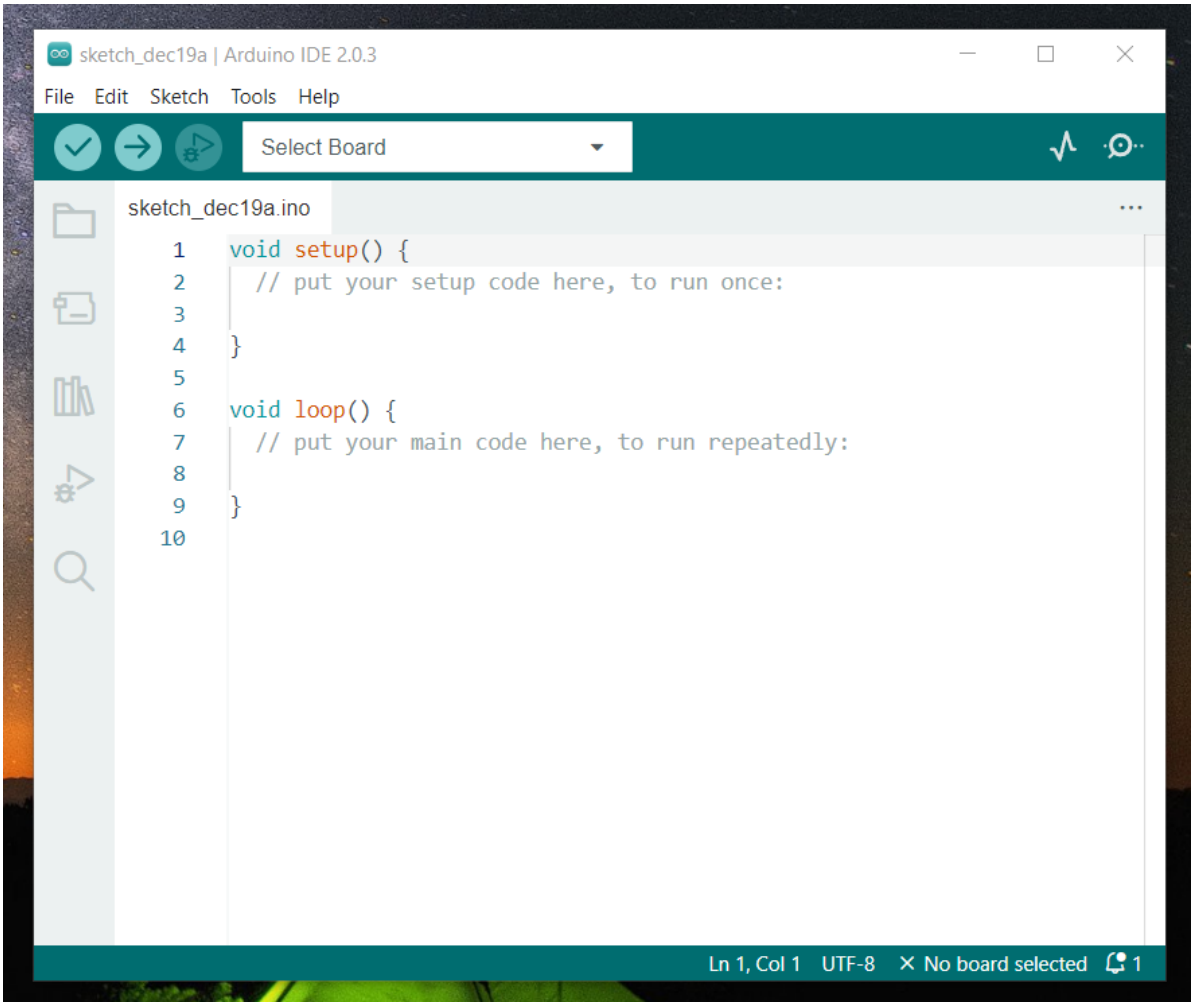
(7) Click Tools > Board >esp32 to choose the ESP32 Dev Module.



2.4 Arduino IDE Setting

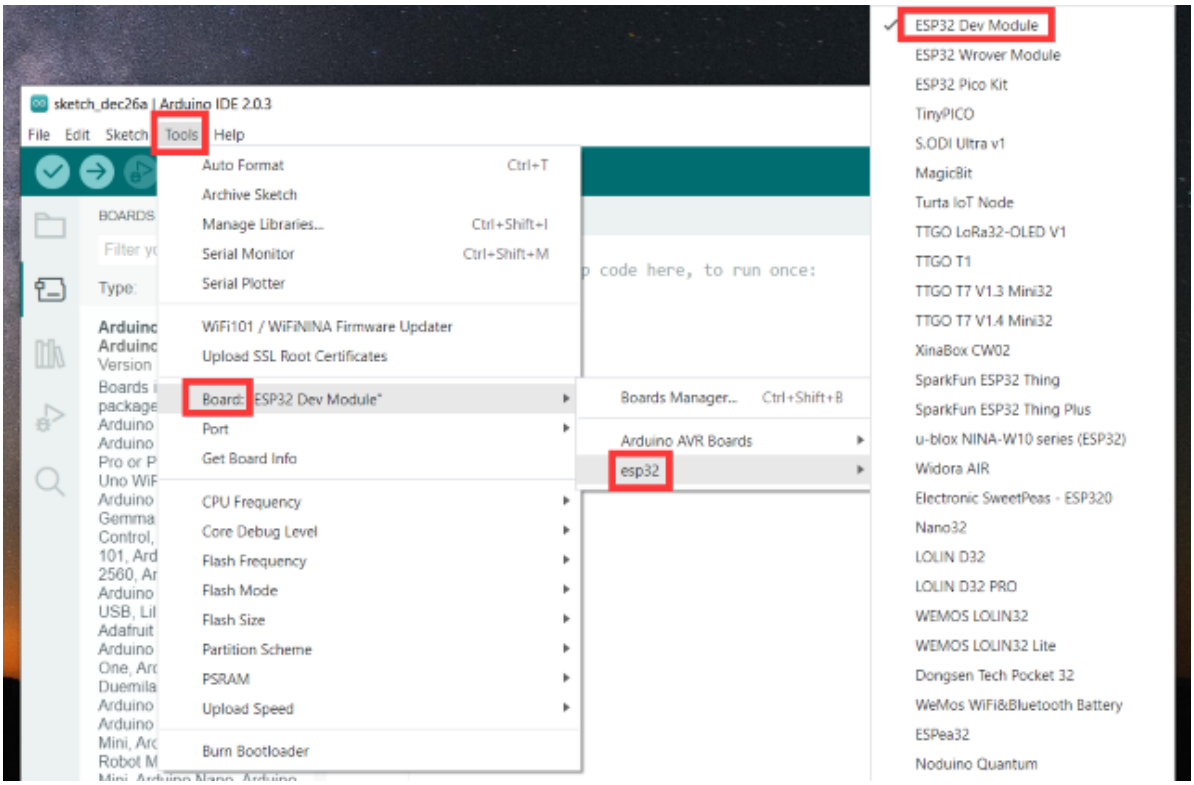
Click 

icon，open Arduino IDE.

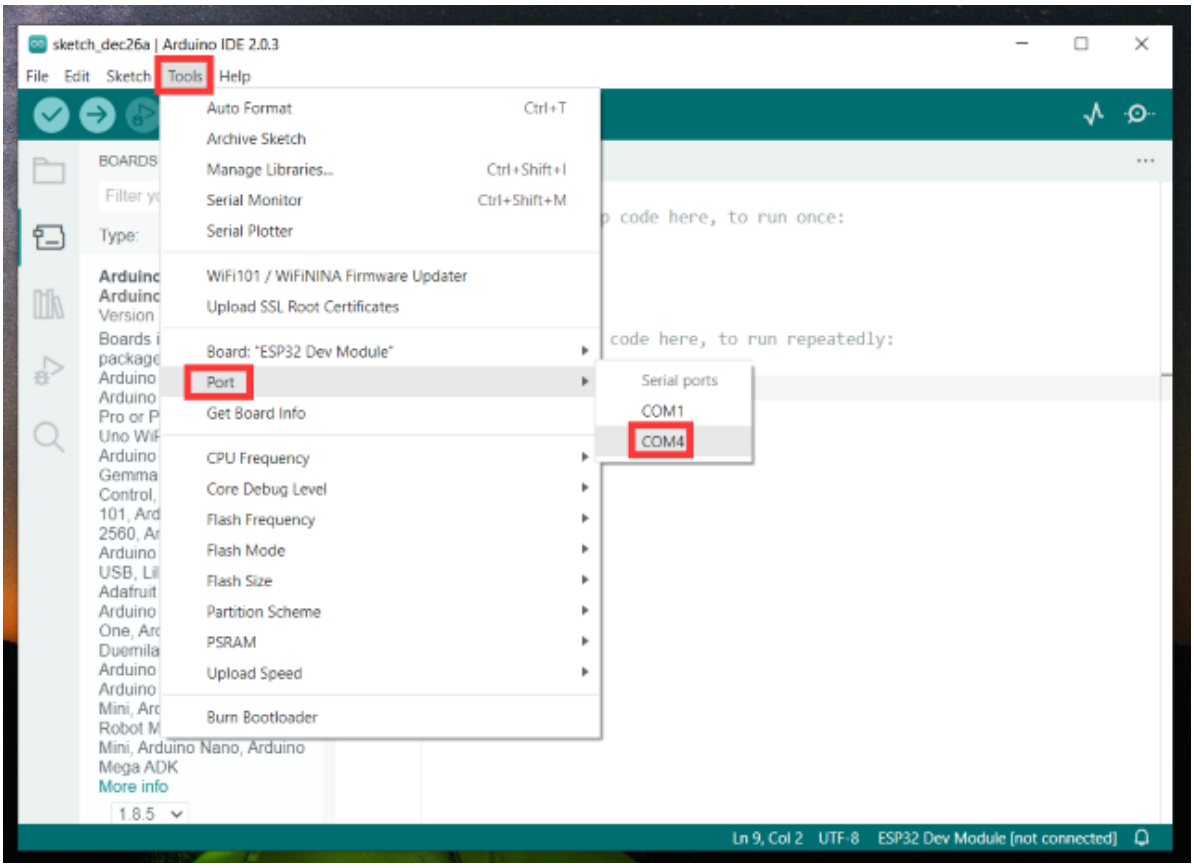


To avoid the errors when uploading the program to the board, you need to select the correct Arduino board that matches the board connected to your computer.

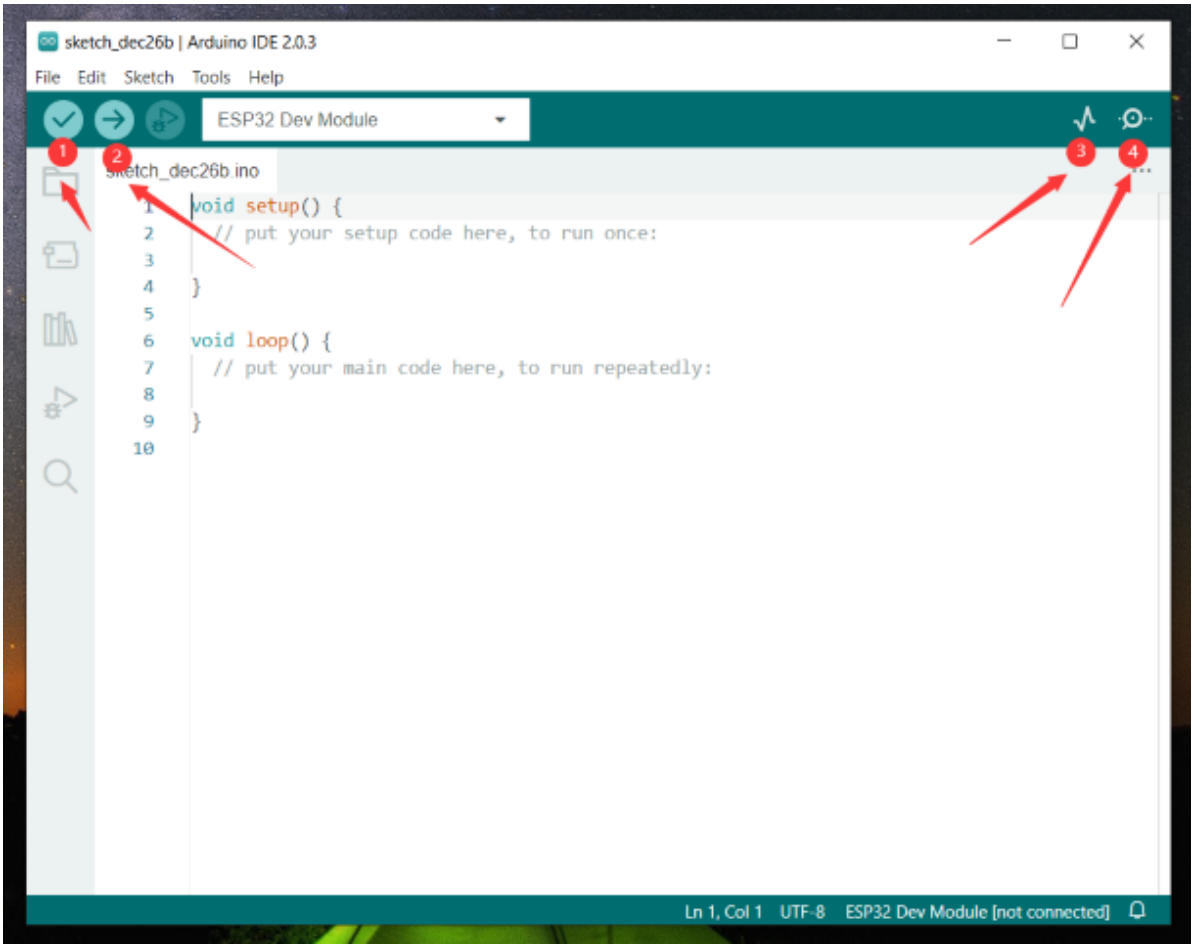
Then come back to the Arduino software, you should click Tools→Board, select the board. (as shown below)



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



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



1- Used to verify whether there is any compiling mistakes or not.

2- Used to upload the sketch to your ESP32 board.

3- Used to send the serial data received from board to the serial plottle.

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

3.Mac System

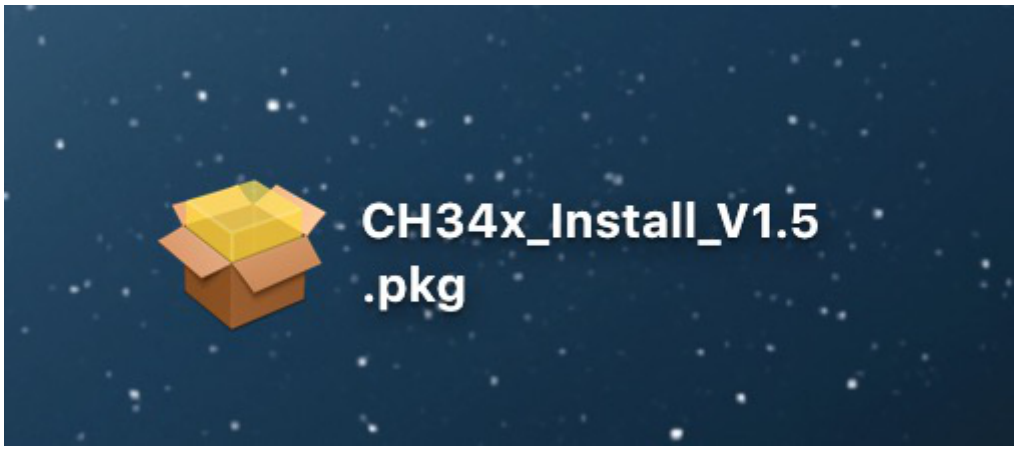
3.1 Download Arduino IDE

3.2 Download the CH340 driver

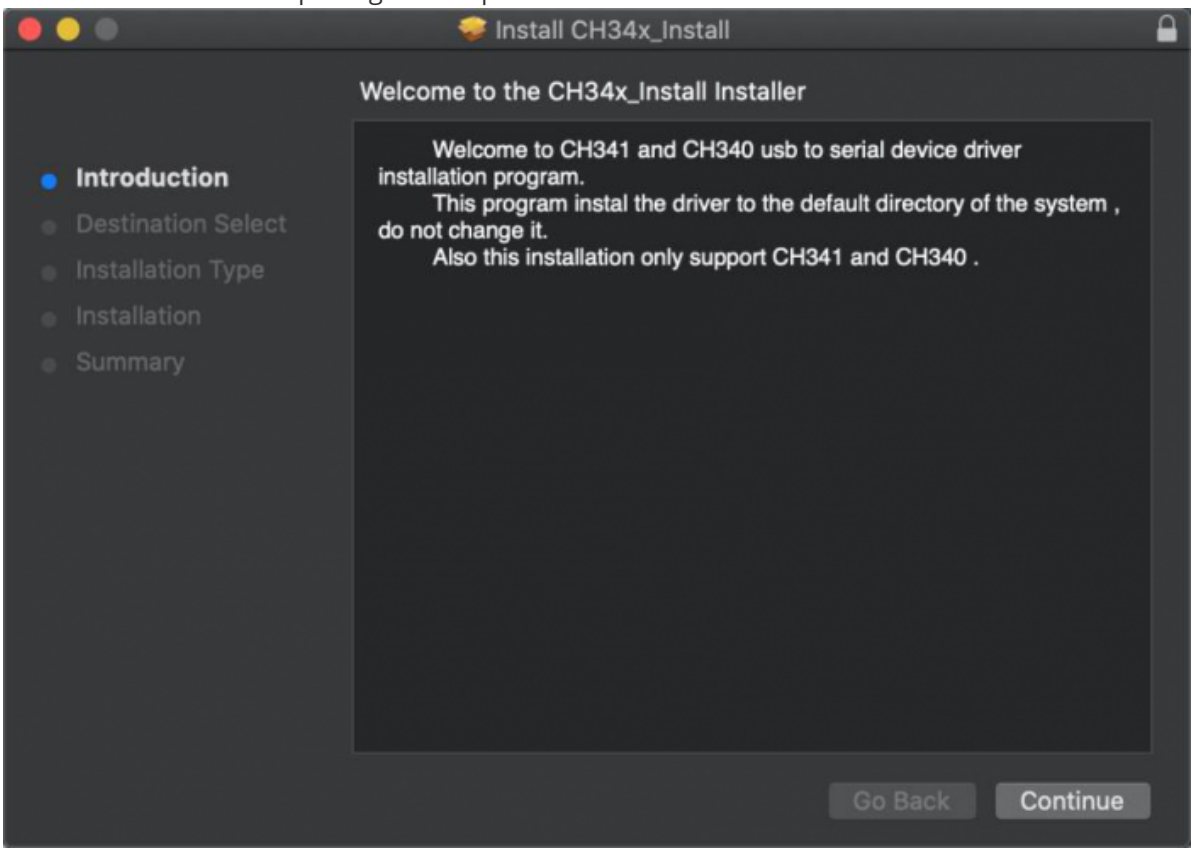
CH340 chip driver for MAC

3.3 How to install the CH340 driver

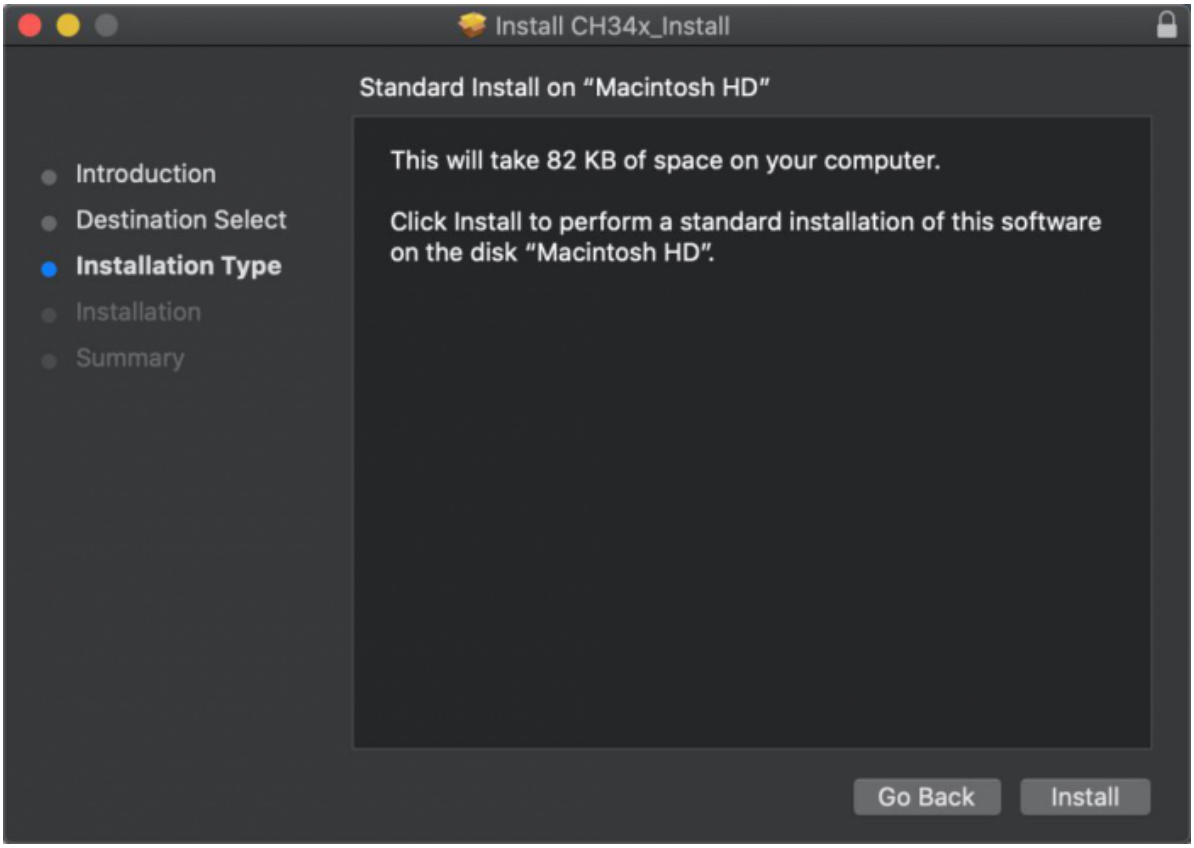
After the download, seen as below:



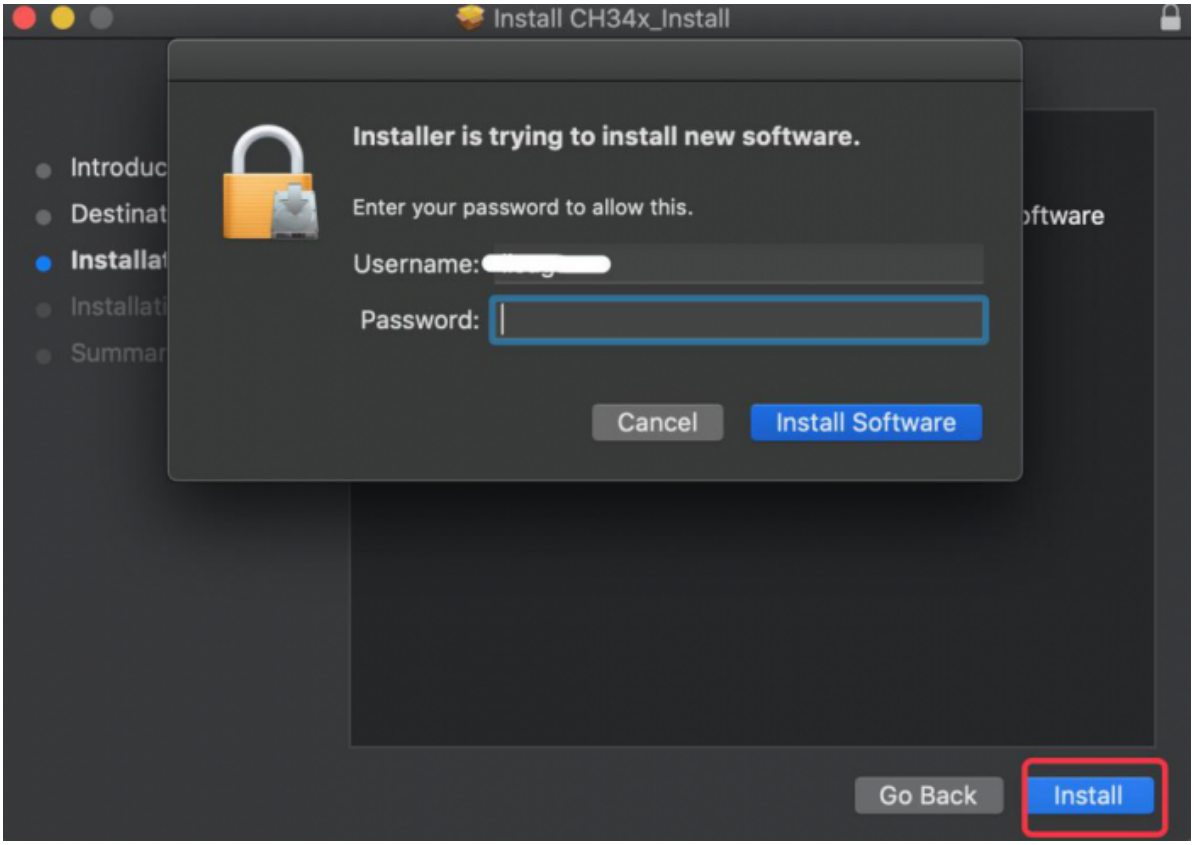
Double-click installation package and tap Continue



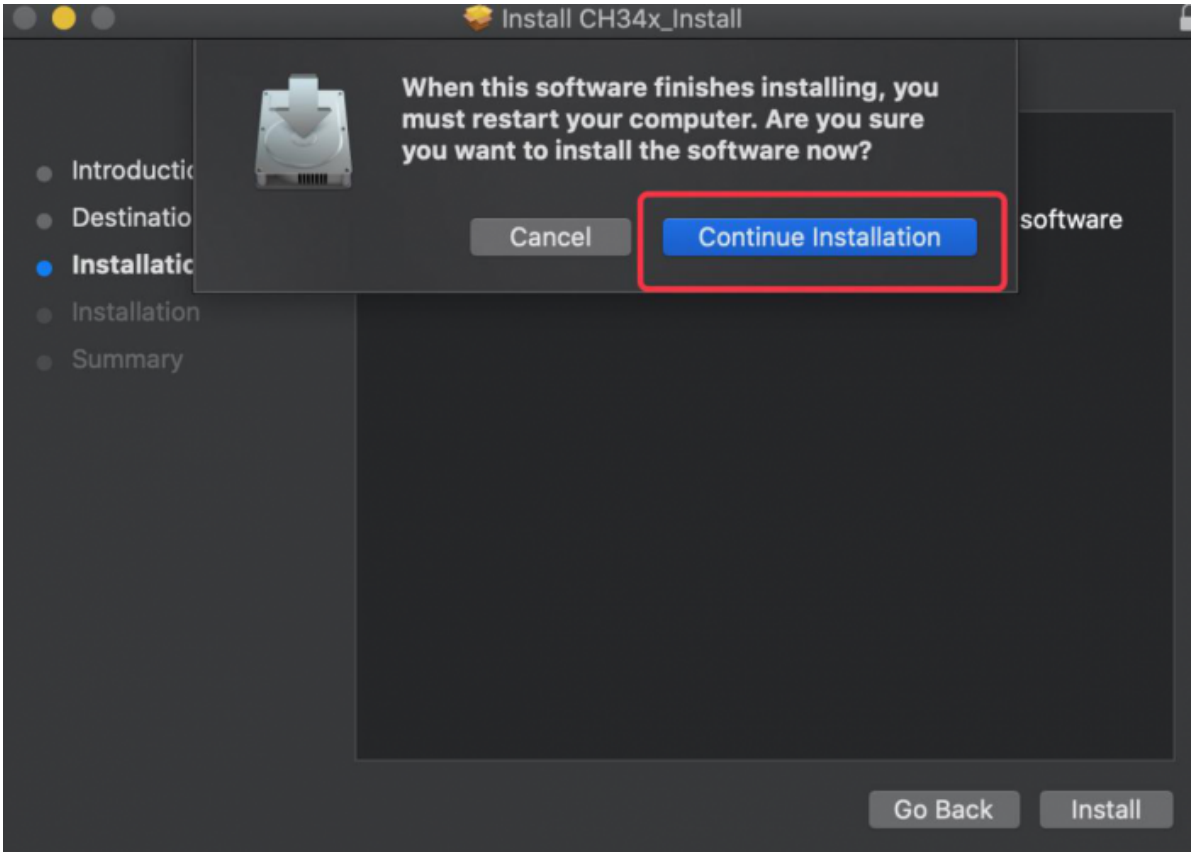
Click Install



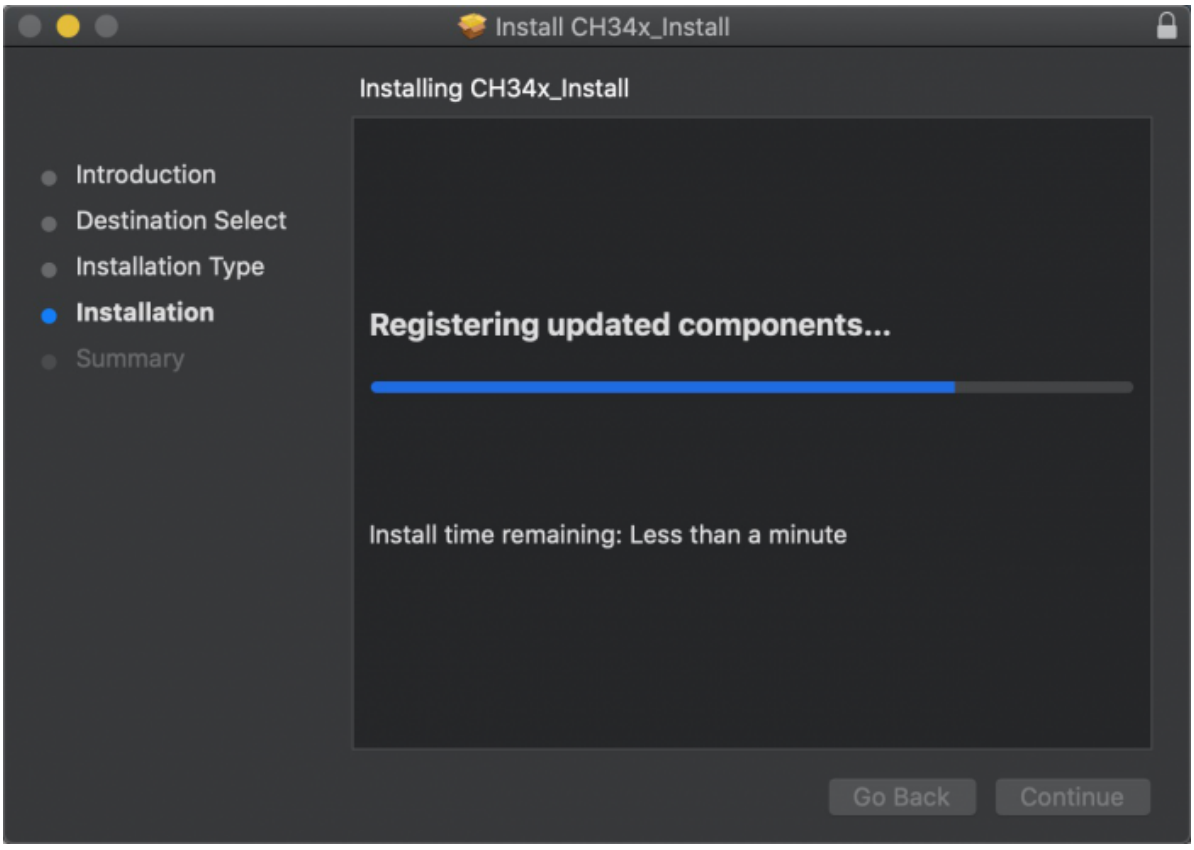
Input your user password and click Install Software



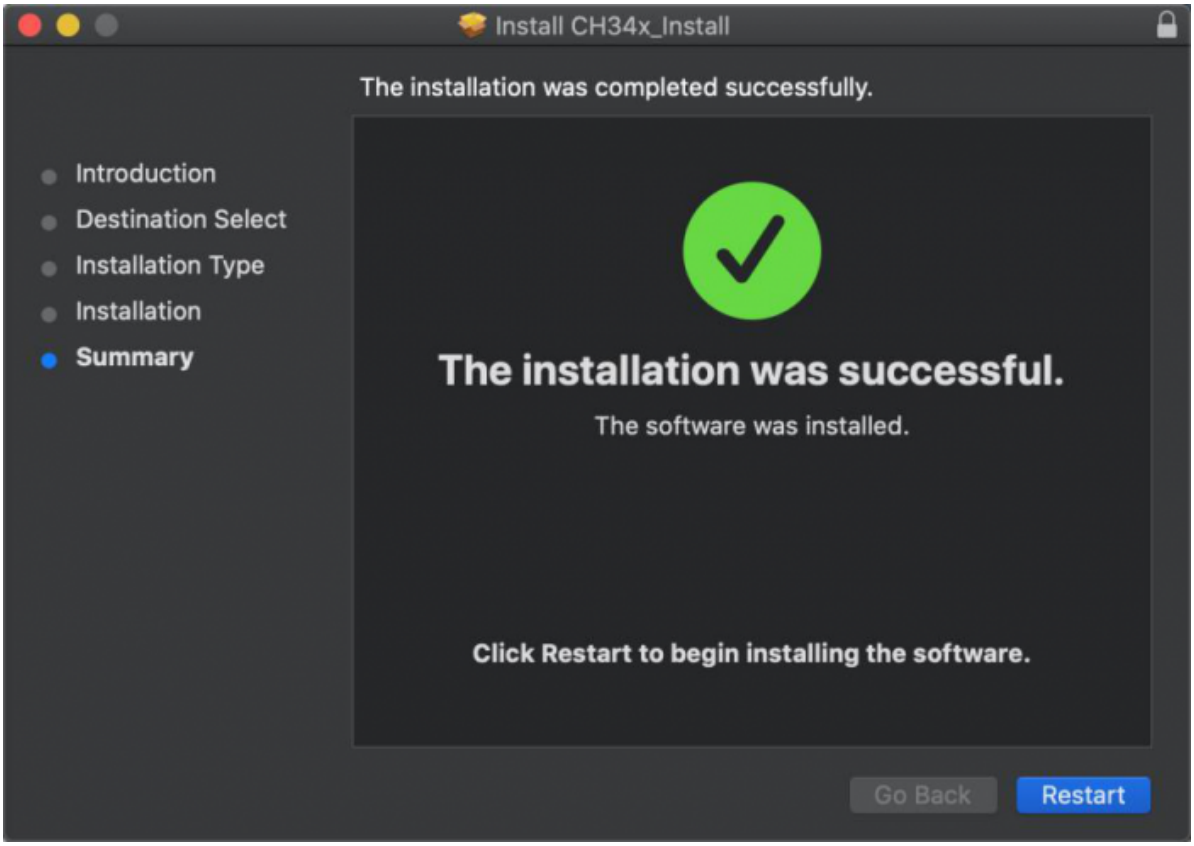
Tap Continue Installation



Wait to install

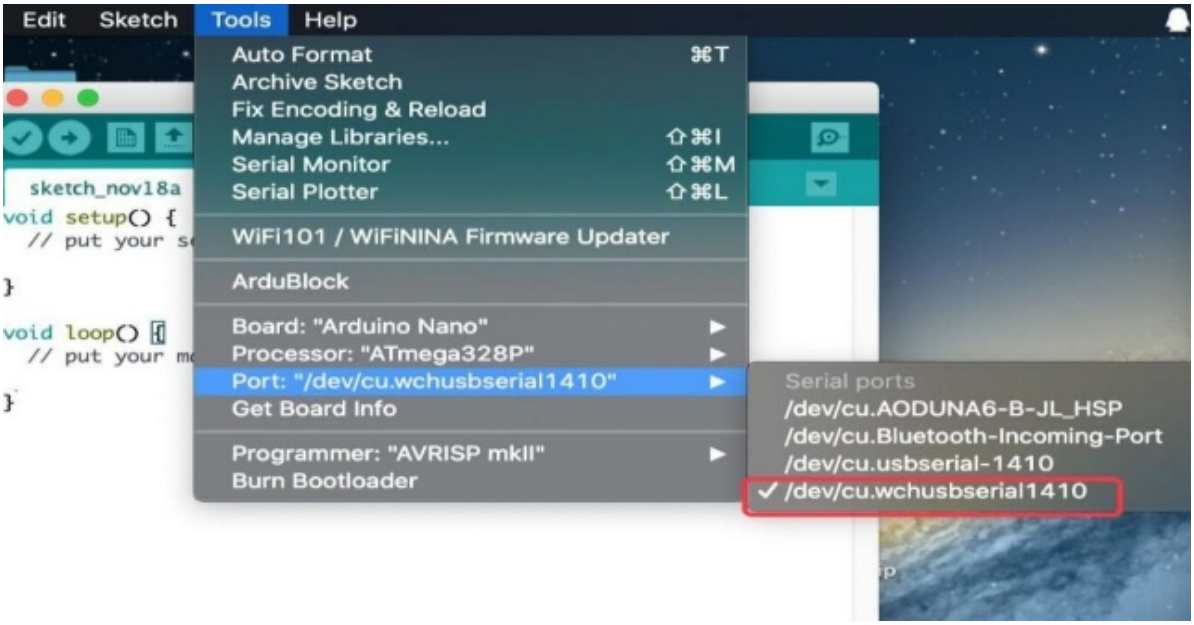


Click Restart after the installation is finished



3.4 Arduino IDE Setting:

Except for COM ports, the setting method is the same as in chapter 1.4:



How to Add Libraries?

What are Libraries ?

Libraries are a collection of code that makes it easy for you to drive a sensor,display, module, etc.

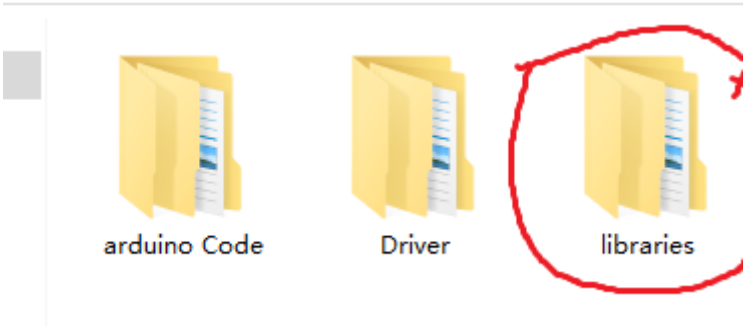
For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

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

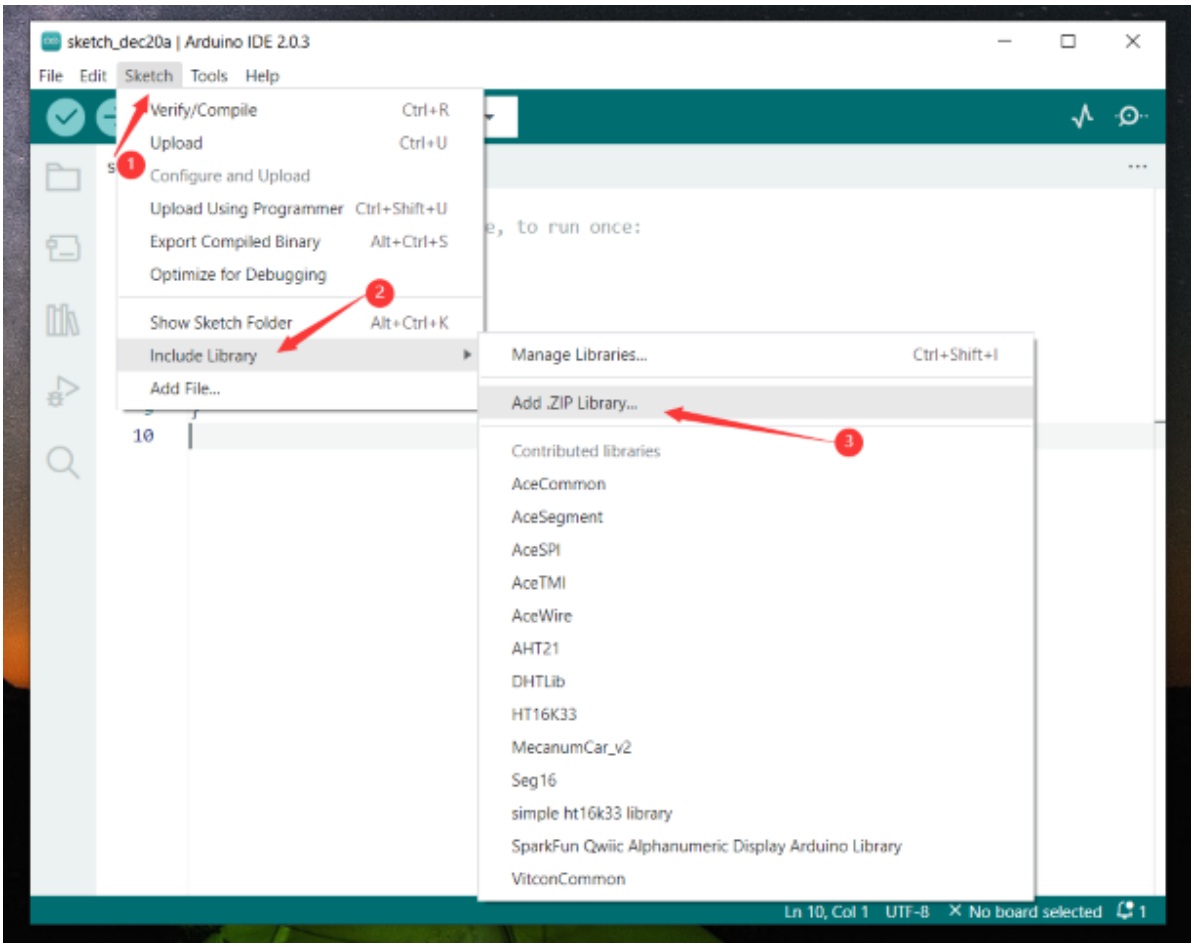
https://www.arduino.cc/en/Reference/Libraries

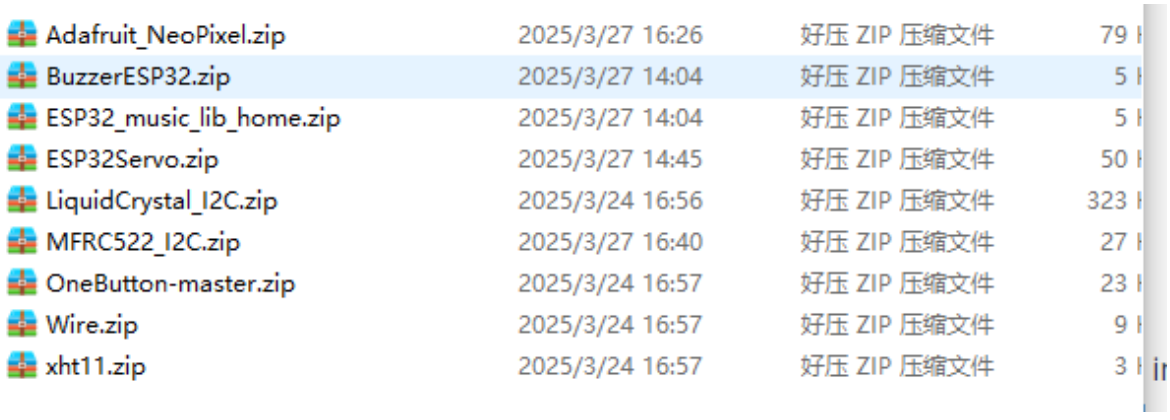
Add ZIP Libraries

When you want to add a zip library, you need to download it as a ZIP file, put in the proper directory. The Libraries needed to run the mini tank can be found on：

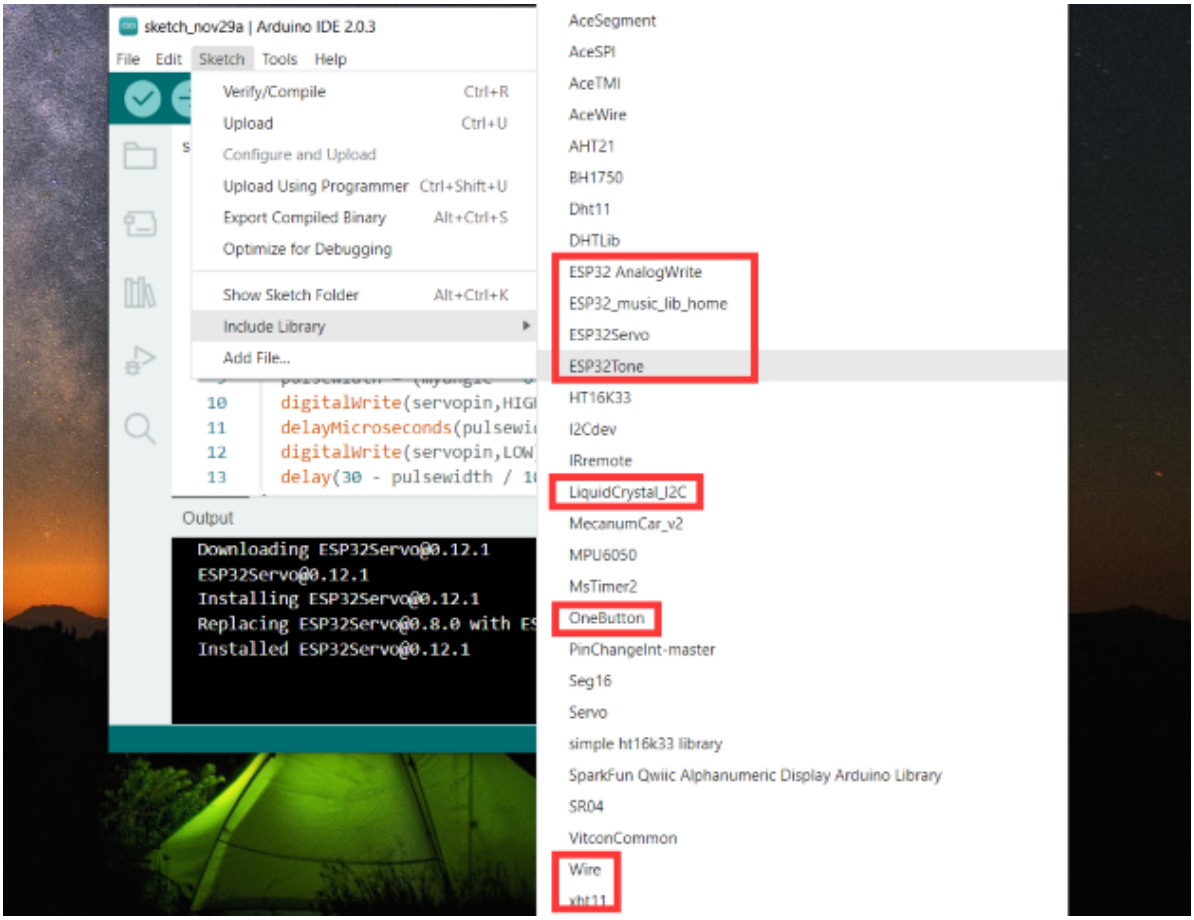


Click Sketch---->Include Library—>Add.ZIP Library，then Then navigate to the library file you downloaded and click "open."





Import the library. You can find it in the include library list.



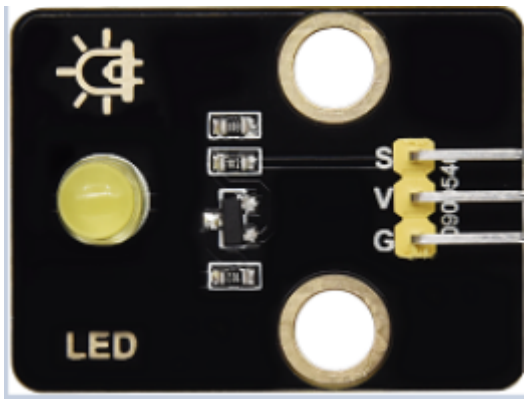
Arduino Projects

Alright, let's get straight to our projects. We will make you know the smart home deeply from the simple sensor.

Note: In this course, the interface of each sensor / module marked with (G,-, GND) indicates the negative pole, G is connected to G, - or GND of sensor shield or control board; "V" is positive pole and connected with V, VCC or 5V.

Project 1.1 LED Blink

1. Description



We've installed the driver of ESP32 PLUS development board.

In the first lesson, we will conduct an experiment to make LED blink.

Let's connect GND and VCC to power. The LED will be on when signal end S is high level, on the contrary, LED will turn off when signal end S is low level.

In addition, the different blinking frequency can be presented by adjusting the delayed time.

2. Working Principle

LED is also the light-emitting diode, which can be made into an electronic module. It will shine if we control pins to output high level, otherwise it will be off.

3. Parameters

Working voltage	**DC 3~5V**
Working current	<20mA
Power	0.1W

4. Control Pin

Yellow LED	**12**

5. Test Code

```
#define led_y 12  //Define the yellow led pin to 12

void setup() {    //The code inside the setup function runs only once
  pinMode(led_y, OUTPUT);  //Set pin to output mode
}

void loop() {     //The code inside the loop function will always run in a loop
  digitalWrite(led_y, HIGH);  //Light up the LED
  delay(200);     //Delay statement, in ms
  digitalWrite(led_y, LOW);   //Close the LED
  delay(200);
}
```

6.Test Result

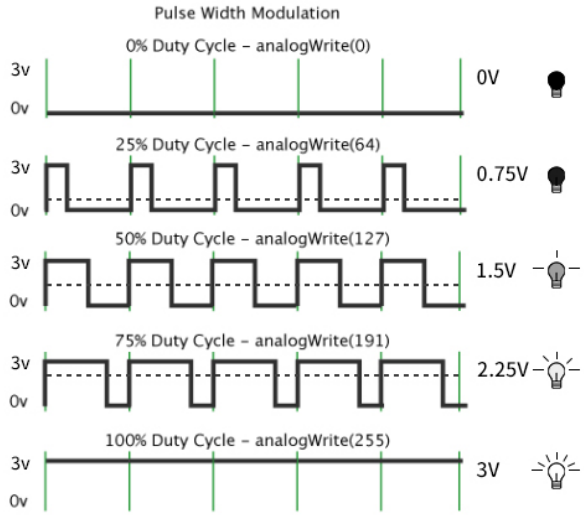
After uploading the code , you can see white and yellow LEDs flashing together.

Project 1.2 Breathing LED

1. Description

A“breathing LED”is a phenomenon where an LED's brightness smoothly changes from dark to bright and back to dark, continuing to do so and giving the illusion of an LED“breathing. However, how to control LED’s brightness?

It makes sense to take advantage of PWM. Output the number of high level and low level in unit time, the more time the high level occupies, the larger the PWM value, the brighter the LED.



We provide the PWM output library file < analogwrite.h > for ESP32, therefore solely a simple statement analogWrite(); can control the PWM output.

2. Test Code

```
#include <Arduino.h>
#define led_y 12    // Define LED pin

void setup()
{
  pinMode(led_y, OUTPUT);  // Set pin as output mode
}

void loop()
{
  for(int i = 0; i < 255; i++)  // For loop: increment variable i until it
reaches 255
  {
    analogWrite(led_y, i);  // PWM output to control LED brightness
    delay(3);
  }

  for(int i = 255; i > 0; i--)  // For loop: decrement variable i until it
reaches 0
  {
    analogWrite(led_y, i);
    delay(3);
  }
}
```

3. Test Result

The LED gradually gets dimmer then brighter, cyclically, like human breathe.

Project 2.1 Read the Button

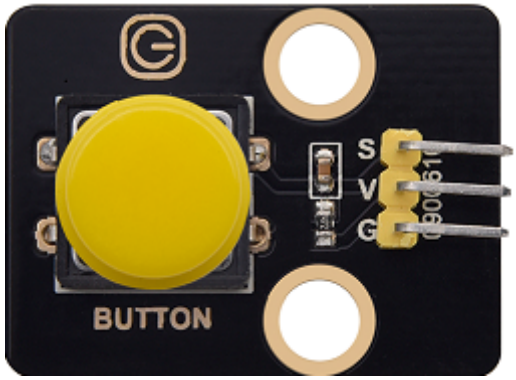
1. Description

The common table lamp uses LED lights and buttons, which can control the light on and off pressing the button.

We will work to read the status value of the button and display it on the serial monitor, so as to see it intuitively.

2. Button Principle

The button module is a digital sensor, which can only read 0 or 1. When the module is not pressed, it is in a high level state, that is, 1, when pressed, it is a low level 0.



3. Pins of the Button

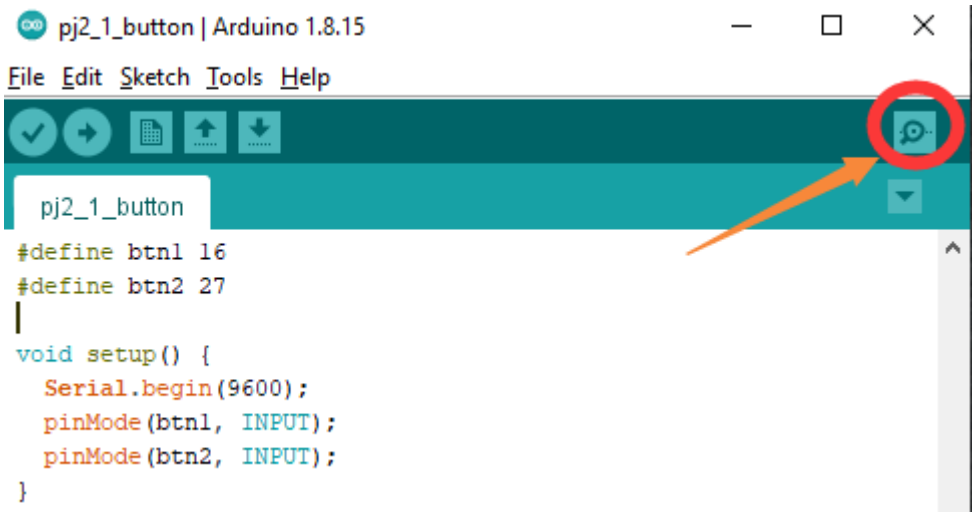
Button 1	**16**
Button 2	27

4. Test Code

```
#define btn1 16
#define btn2 27

void setup() {
  Serial.begin(9600);
  pinMode(btn1, INPUT);
  pinMode(btn2, INPUT);
}

void loop() {
  boolean btn1_val = digitalRead(btn1);
  boolean btn2_val = digitalRead(btn2);
  Serial.print("button1 = ");
  Serial.print(btn1_val);
  Serial.print("   ");
  Serial.print("button2 = ");
  Serial.println(btn2_val);
  delay(100);
}
```

5. Test Result

Open the serial monitor of the arduino IDE



Press the button again to see the change of the button state value, as shown below:

Project 2.2. Table Lamp

1. Description

For common simple table lamp, click the button it will be opened, click it again, the lamp will be closed.

2. Test Code

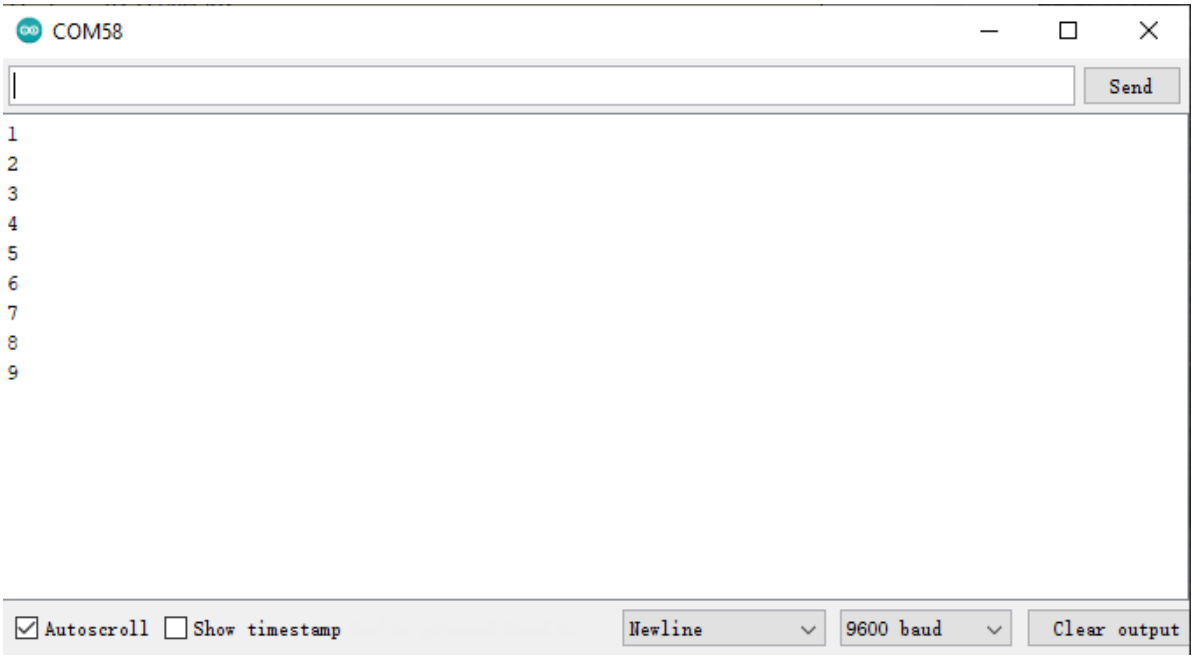
Calculate the clicked button times and take the remainder of 2, you can get 0 or 1 two state values.

```
#define btn1 16
#define led_y 12
int btn_count = 0; // Counter for button presses

void setup()
{
```

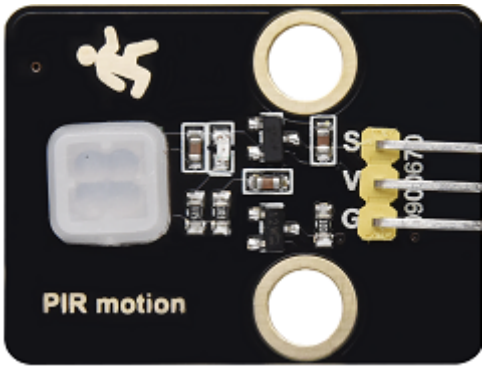
```
  Serial.begin(9600);
  pinMode(btn1, INPUT);
  pinMode(led_y, OUTPUT);
}

void loop()
{
  boolean btn1_val = digitalRead(btn1);
  if(btn1_val == 0) // If button is pressed
  {
    delay(10);  // 10ms delay for debouncing
    if(btn1_val == 0) // Confirm button is still pressed
    {
      boolean btn_state = 1;
      while(btn_state == 1) // Loop until button is released
      {
        boolean btn_val = digitalRead(btn1);
        if(btn_val == 1)  // If button is released
        {
          btn_count++;    // Increment press counter
          Serial.println(btn_count);
          btn_state = 0;  // Exit loop
        }
      }
    }
    boolean value = btn_count % 2; // Modulo operation (0 or 1)
    if(value == 1)
    {
      digitalWrite(led_y, HIGH); // Turn LED on
    }
    else
    {
      digitalWrite(led_y, LOW); // Turn LED off
    }
  }
}
```

3. Test Result

Open the serial monitor and print out the clicked button times, then click the button once, the LED will be on, click it again, it will be off.



Project 3.1 Read the PIR Motion Sensor



1. Description

The PIR motion sensor has many application scenarios in daily life, such as automatic induction lamp of stairs, automatic induction faucet of washbasin, etc.

It is also a digital sensor like buttons, which has two state

values 0 or 1. And it will be sensed when people are moving.

We will print out the value of the PIR motion sensor through the serial monitor.

2. Control Pin

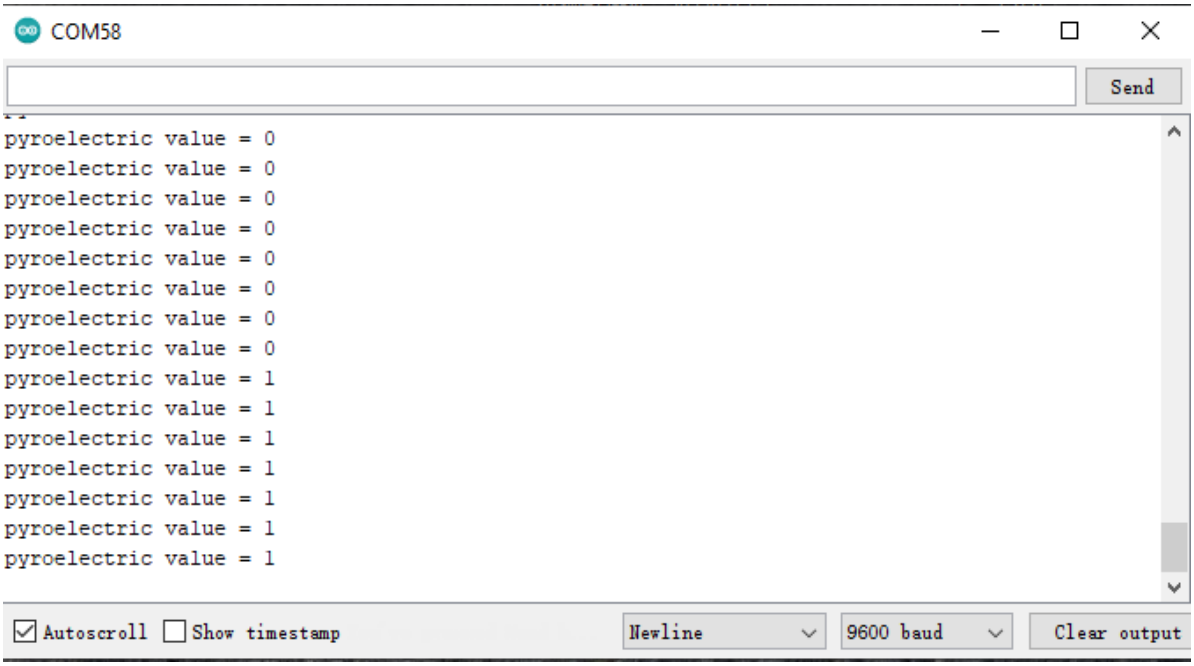
PIR motion sensor	14

3. Test Code

```
#define pyroelectric 14

void setup() {
  Serial.begin(9600);
  pinMode(pyroelectric, INPUT);
}

void loop() {
  boolean pyroelectric_val = digitalRead(pyroelectric);
  Serial.print("pyroelectric value = ");
  Serial.println(pyroelectric_val);
  delay(200);
}
```

4. Test Result

When you stand still in front of the sensor, the reading value is 0, move a little, it will change to 1.



Project 3.2 PIR Motion Sensor

If someone moves in front of the sensor, the LED will light up.

1. Test Code

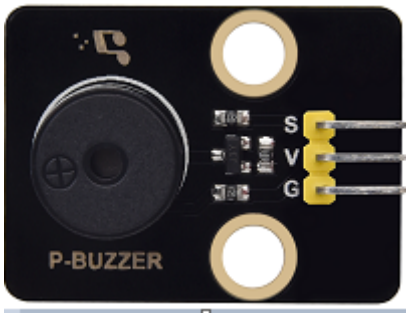
```
#define pyroelectric 14
#define led_y 12  // Yellow LED pin definition

void setup()
{
  Serial.begin(9600);
  pinMode(pyroelectric, INPUT);
  pinMode(led_y, OUTPUT);  // Set pin as output mode
}

void loop()
{
```

```
  boolean pyroelectric_val = digitalRead(pyroelectric);

  Serial.print("pyroelectric value = ");
  Serial.println(pyroelectric_val);

  delay(200);

  if(pyroelectric_val == 1)
  {
    digitalWrite(led_y, HIGH);  // Turn LED on when motion detected
  }
  else
  {
    digitalWrite(led_y, LOW);   // Turn LED off when no motion
  }
}
```

2. Test Result

Move your hand in front of the sensor, the LED will turn on. After 5s of immobility, the LED lights will turn off.

Project 4.1 Play Happy Birthday



1. Description

There is a audio power amplifier element in the car expansion board, which is as an external amplification equipment to play music.

In this project, we will work to play a piece of music by using it.

2. Component Knowledge

Passive Buzzer: The audio power amplifier (like the passive buzzer) does not have internal oscillation. When controlling, we need to input square waves of different frequencies to the positive pole of the component and ground the negative pole to control the power amplifier to chime sounds of different frequencies.

3. Control Pin

Passive Buzzer	25

4. Test Code

```
#include <BuzzerESP32.h>

BuzzerESP32 buzzer(25); // Initialize buzzer on GPIO25

void setup()
{
  buzzer.setTimbre(30); // Set timbre (sound quality)
  birthday();           // Play birthday melody
}

void loop()
{
  // Empty loop as melody plays only once at startup
}

void birthday()
{
  // Play birthday melody - parameters are (frequency, duration)
  buzzer.playTone(294, 250);  // D4
  buzzer.playTone(440, 250);  // A4
  buzzer.playTone(392, 250);  // G4
  buzzer.playTone(532, 250);  // C5 (slightly sharp)
  buzzer.playTone(494, 250);  // B4
  buzzer.playTone(392, 250);  // G4
  buzzer.playTone(440, 250);  // A4
  buzzer.playTone(392, 250);  // G4
  buzzer.playTone(587, 250);  // D5
  buzzer.playTone(532, 250);  // C5 (slightly sharp)
  buzzer.playTone(392, 250);  // G4
  buzzer.playTone(784, 250);  // G5
  buzzer.playTone(659, 250);  // E5
  buzzer.playTone(532, 250);  // C5 (slightly sharp)
  buzzer.playTone(494, 250);  // B4
  buzzer.playTone(440, 250);  // A4
  buzzer.playTone(698, 250);  // F5
  buzzer.playTone(659, 250);  // E5
  buzzer.playTone(532, 250);  // C5 (slightly sharp)
  buzzer.playTone(587, 250);  // D5
  buzzer.playTone(532, 500);  // C5 (slightly sharp) - longer duration
  buzzer.playTone(0, 0);      // Turn off buzzer
}
```

5. Test Result

The passive buzzer will play happy Birthday.

Project 4.2 Music Box

we will make a music box and switch tunes by pressing buttons.

1. Test Code

```
#include <musicESP32_home.h>
music Music(25);  // Initialize music player on GPIO25
#define btn1 16    // Button pin
int btn_count = 0; // Counter for button presses
boolean music_flag = 0; // Flag to trigger music playback

void setup()
{
  Serial.begin(9600);
  pinMode(btn1, INPUT);
  // Available music options:
  // Music.tetris();
  // Music.birthday();
  // Music.Ode_to_Joy();
  // Music.christmas();
  // Music.star_war_tone();
}

void loop()
{
  boolean btn1_val = digitalRead(btn1);

  if(btn1_val == 0) // If button is pressed
  {
    delay(10);  // 10ms delay for debouncing

    if(btn1_val == 0) // Confirm button is still pressed
    {
      boolean btn_state = 1;

      while(btn_state == 1) // Wait until button is released
      {
        boolean btn_val = digitalRead(btn1);

        if(btn_val == 1)  // If button is released
        {
          music_flag = 1;
          btn_count++;    // Increment press counter
          Serial.println(btn_count);

          // Cycle through 1-3 count
          if(btn_count == 4)
          {
            btn_count = 1;
          }

          // Play different song based on press count
          switch(btn_count)
          {
            case 1:
              if(music_flag == 1)
              {
                Music.Ode_to_Joy();
                music_flag=0;
              }
```

```
              break;

            case 2:
              if(music_flag == 1)
              {
                Music.christmas();
                music_flag=0;
              }
              break;

            case 3:
              if(music_flag == 1)
              {
                Music.tetris();
                music_flag=0;
              }
              break;
          }

          btn_state = 0;  // Exit wait loop
        }
      }
    }
  }
}
```

2. Test Result

Click button 1 once, it will play a Tetris, then click it again, it will play *Ode to Joy*, after playing, click the button 1 for the third time, it will play Christmas.

Project 5.1 Control the Door

1. Description

Automatic doors and windows need power device, which will become more automatic with a 180 degree servo and some sensors. Adding a raindrop sensor, you can achieve the effect of closing windows automatically when raining. If adding a RFID, we can realize the effect of swiping to open the door and so on.

2. Component Knowledge

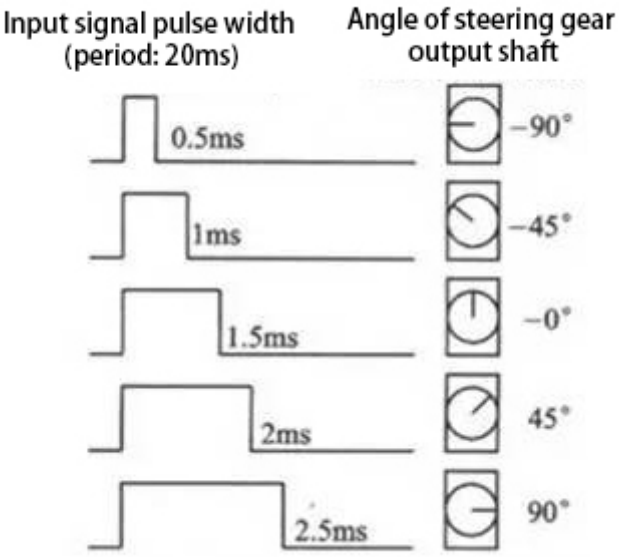
Servo: Servo is a position servo driver device consists of a housing, a circuit board, a coreless motor, a gear and a position detector.

Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of 1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and obtain the voltage difference output.

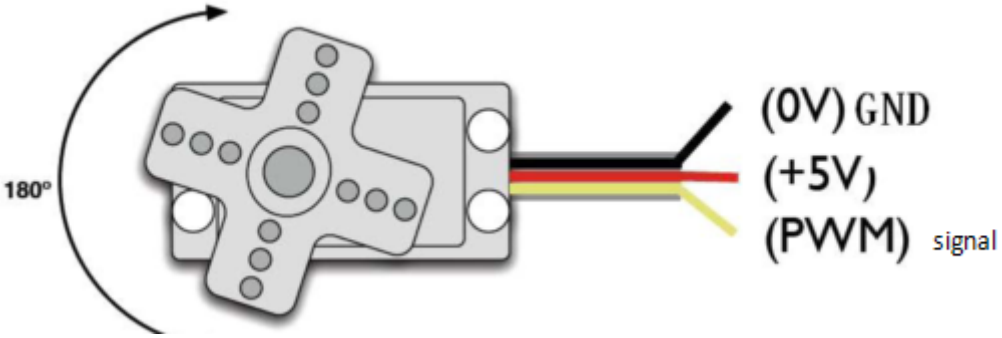
The IC on the circuit board judges the direction of rotation, and then drives the coreless motor to start rotation. The power is transmitted to the swing arm through the reduction gear, and the signal is sent back by the position detector to judge whether the positioning has been reached, which is suitable for those control systems that require constant angle change and can be maintained.

When the motor speed is constant, the potentiometer is driven to rotate through the cascade reduction gear, which leads that the voltage difference is 0, and the motor stops rotating. Generally, the angle range of servo rotation is 0° --180 °.

The pulse period of the control servo is 20ms, the pulse width is 0.5ms ~ 2.5ms, and the corresponding position is -90°~ +90°. Here is an example of a 180° servo:



In general, servo has three lines in brown, red and orange. The brown wire is grounded, the red one is a positive pole line and the orange one is a signal line.



3. Pin

The servo of the window	5
The servo of the door	13

4. Test Code

```
#include <ESP32Servo.h>
Servo myservo;  // create servo object to control a servo
                // 16 servo objects can be created on the ESP32

int pos = 0;    // variable to store the servo position
// Recommended PWM GPIO pins on the ESP32 include 2,4,12-19,21-23,25-27,32-33
int servoPin = 13;

void setup() {
    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);
    myservo.setPeriodHertz(50);    // standard 50 hz servo
    myservo.attach(servoPin, 1000, 2000); // attaches the servo on pin 18 to the
servo object
    // using default min/max of 1000us and 2000us
    // different servos may require different min/max settings
    // for an accurate 0 to 180 sweep

}

void loop() {
    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
        // in steps of 1 degree
        myservo.write(pos);    // tell servo to go to position in variable 'pos'
        delay(15);             // waits 15ms for the servo to reach the position
    }
    for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
        myservo.write(pos);    // tell servo to go to position in variable 'pos'
        delay(15);             // waits 15ms for the servo to reach the position
    }

}
```

5. Test Result

The servo of the door turns with the door, back and forth

Project 5.2 Close the Window

1. Description

We will work to use a servo and a raindrop sensor to make an device closing windows automatically when raining.

2. Component Knowledge

Raindrop Sensor: This is an analog input module, the greater the area covered by water on the detection surface, the greater the value returned (range 0~4096).

3. Test Code

```
#include <ESP32Servo.h>

#define servoPin 5
#define waterPin 34
Servo myservo;


void setup() {
  Serial.begin(9600);
  pinMode(waterPin, INPUT);

    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);
    myservo.setPeriodHertz(50);    // standard 50 hz servo
    myservo.attach(servoPin, 1000, 2000); // attaches the servo on pin 18 to the
servo object
    // using default min/max of 1000us and 2000us
    // different servos may require different min/max settings
    // for an accurate 0 to 180 sweep

  delay(200);
}

void loop() {
  int water_val = analogRead(waterPin);
  Serial.println(water_val);
  if(water_val > 1500) {
    myservo.write(0);
    delay(200);
  }
  else {
    myservo.write(176);
    delay(200);
  }
}
```

4. Test Result

At first, the window opens automatically, and when you touch the raindrop sensor with your hand (which has water on the skin), the window will close.

Project 6.1 Control SK6812

1. Description

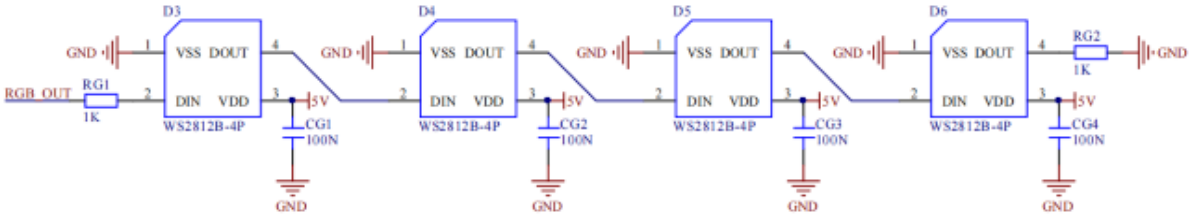
The atmosphere lamp of smart home is 4 SK6812RGB LEDs. RGB LED belongs to a simple luminous module, which can adjust the color to bring out the lamp effect of different colors. Furthermore, it can be widely used in buildings, bridges, roads, gardens, courtyards, floors and other fields of decorative lighting and venue layout, Christmas, Halloween, Valentine's Day, Easter, National Day as well as other festivals during the atmosphere and other scenes.

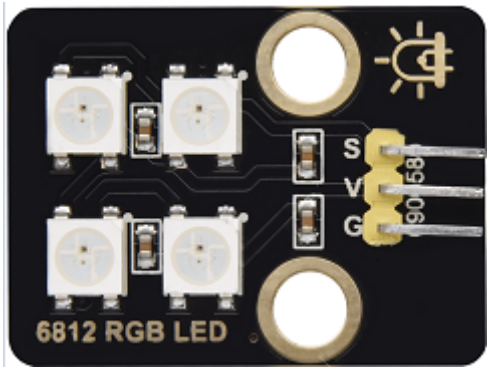
In this experiment, we will make various lighting effects.

2. Component Knowledge

From the schematic diagram, we can see that these four RGB LEDs are all connected in series. In fact, no matter how many they are, we can use a pin to control a RGB LED and let it display any color. Each RGBLED is an independent pixel, composed of R, G and B colors, which can achieve 256 levels of brightness display and complete the full true color display of 16777216 colors.

What's more, the pixel point contains a data latch signal shaping amplifier drive circuit and a signal shaping circuit, which effectively ensures the color of the pixel point light is highly consistent.





3. Pin

SK6812	26

4. Test Code

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h>                                  // Required for 16 MHz
Adafruit Trinket
#endif
#define LED_PIN    26                                   // Which pin on the Arduino
is connected to the NeoPixels?
#define LED_COUNT 4                                     // How many NeoPixels are
attached to the Arduino?
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800); // Declare our
NeoPixel strip object:

void setup() {
#if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
  clock_prescale_set(clock_div_1);                      // These lines are
specifically to support the Adafruit Trinket 5V 16 MHz.
#endif
  strip.begin();                                        // INITIALIZE NeoPixel strip
object (REQUIRED)
  strip.show();                                         // Turn OFF all pixels ASAP
  strip.setBrightness(50);                              // Set BRIGHTNESS to about
1/5 (max = 255)
}

void loop() {
  colorWipe(strip.Color(255,   0,   0), 50);         // Red
  colorWipe(strip.Color(  0, 255,   0), 50);         // Green
  colorWipe(strip.Color(  0,   0, 255), 50);         // Blue

  theaterChase(strip.Color(127, 127, 127), 50);      // White, half brightness
  theaterChase(strip.Color(127,   0,   0), 50);      // Red, half brightness
  theaterChase(strip.Color(  0,   0, 127), 50);      // Blue, half brightness

  rainbow(10);                                          // Flowing rainbow cycle
along the whole strip
  theaterChaseRainbow(50);                              // Rainbow-enhanced
theaterChase variant
}

void colorWipe(uint32_t color, int wait) {
  for(int i=0; i<strip.numPixels(); i++) {          // For each pixel in
strip...
    strip.setPixelColor(i, color);                  // Set pixel's color (in
RAM)
    strip.show();                                   // Update strip to match
    delay(wait);                                    // Pause for a moment
  }
}

void theaterChase(uint32_t color, int wait) {
  for(int a=0; a<10; a++) {                         // Repeat 10 times...
    for(int b=0; b<3; b++) {                        // 'b' counts from 0 to 2...
      strip.clear();                                // Set all pixels in RAM to 0
(off)
```

```
      for(int c=b; c<strip.numPixels(); c += 3) {    // 'c' counts up from 'b' to
end of strip in steps of 3...
        strip.setPixelColor(c, color);               // Set pixel 'c' to value
'color'
      }
      strip.show();                                  // Update strip with new
contents
      delay(wait);                                   // Pause for a moment
    }
  }
}

void rainbow(int wait) {
  for(long firstPixelHue = 0; firstPixelHue < 5*65536; firstPixelHue += 256) {
    for(int i=0; i<strip.numPixels(); i++) {        // For each pixel in
strip...
      int pixelHue = firstPixelHue + (i * 65536L / strip.numPixels());
      strip.setPixelColor(i, strip.gamma32(strip.ColorHSV(pixelHue)));
    }
    strip.show();                                   // Update strip with new
contents
    delay(wait);                                   // Pause for a moment
  }
}

void theaterChaseRainbow(int wait) {
  int firstPixelHue = 0;                           // First pixel starts at red
(hue 0)
  for(int a=0; a<30; a++) {                        // Repeat 30 times...
    for(int b=0; b<3; b++) {                       // 'b' counts from 0 to 2...
      strip.clear();                               // Set all pixels in RAM to 0
(off)
      for(int c=b; c<strip.numPixels(); c += 3) {  // 'c' counts up from 'b' to
end of strip in increments of 3...
        int      hue   = firstPixelHue + c * 65536L / strip.numPixels();
        uint32_t color = strip.gamma32(strip.ColorHSV(hue)); // hue -> RGB
        strip.setPixelColor(c, color);             // Set pixel 'c' to value
'color'
      }
      strip.show();                                // Update strip with new
contents
      delay(wait);                                // Pause for a moment
      firstPixelHue += 65536 / 90;                // One cycle of color wheel over
90 frames
    }
  }
}
```

5. Test Result

The atmosphere lamps of the smart home will display a variety of colors and light effects.

Project 6.2 Button

1. Description

There are two buttons to switch the color of the atmosphere lamp.

2. Test Code

```
#define btn1 16    // Button 1 pin
#define btn2 27    // Button 2 pin

#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
 #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#endif

#define LED_PIN    26    // NeoPixel data pin
#define LED_COUNT 4      // Number of NeoPixels
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);

int btn_count = 0; // Counter for button presses

void setup()
{
  Serial.begin(9600);
  pinMode(btn1, INPUT);
  pinMode(btn2, INPUT);

  #if defined(__AVR_ATtiny85__) && (F_CPU == 16000000)
    clock_prescale_set(clock_div_1);
  #endif

  strip.begin();           // Initialize NeoPixel strip
  strip.show();            // Turn off all pixels
  strip.setBrightness(50); // Set brightness (max 255)
}

void loop()
{
  boolean btn1_val = digitalRead(btn1);
  boolean btn2_val = digitalRead(btn2);

  // Button 1 (Decrement) handling
  if(btn1_val == 0) // If button is pressed
  {
    delay(10);  // Debounce delay
    if(btn1_val == 0) // Confirm button press
    {
      boolean btn_state = 1;
      while(btn_state == 1) // Wait for button release
      {
        boolean btn_val = digitalRead(btn1);
        if(btn_val == 1)  // If button released
        {
          btn_count--;    // Decrement counter
          if(btn_count <= 0) // Limit minimum value
          {
```

```
            btn_count = 0;
          }
          Serial.println(btn_count);
          btn_state = 0;  // Exit loop
        }
      }
    }
  }

  // Button 2 (Increment) handling
  if(btn2_val == 0) // If button is pressed
  {
    delay(10);  // Debounce delay
    if(btn2_val == 0) // Confirm button press
    {
      boolean btn_state2 = 1;
      while(btn_state2 == 1) // Wait for button release
      {
        boolean btn2_val = digitalRead(btn2);
        if(btn2_val == 1)  // If button released
        {
          btn_count++;    // Increment counter
          if(btn_count >= 6) // Limit maximum value
          {
            btn_count = 6;
          }
          Serial.println(btn_count);
          btn_state2 = 0;  // Exit loop
        }
      }
    }
  }

  // Change LED color based on button count
  switch(btn_count)
  {
    case 0: colorWipe(strip.Color(0,   0,   0), 50); break;    // Off
    case 1: colorWipe(strip.Color(255,  0,   0), 50); break;   // Red
    case 2: colorWipe(strip.Color(0,   255,   0), 50); break;  // Green
    case 3: colorWipe(strip.Color(0,   0,   255), 50); break;  // Blue
    case 4: colorWipe(strip.Color(255, 255,   0), 50); break;  // Yellow
    case 5: colorWipe(strip.Color(255, 0,   255), 50); break;  // Magenta
    case 6: colorWipe(strip.Color(255, 255, 255), 50); break;  // White
  }
}

// Fill strip with one color
void colorWipe(uint32_t color, int wait)
{
  for(int i=0; i<strip.numPixels(); i++)
  {
    strip.setPixelColor(i, color); // Set pixel color
    strip.show();                  // Update strip
    delay(wait);                   // Pause
  }
}
```

3. Test Result

We can switch the color of the atmosphere lamp by clicking buttons 1 and 2.

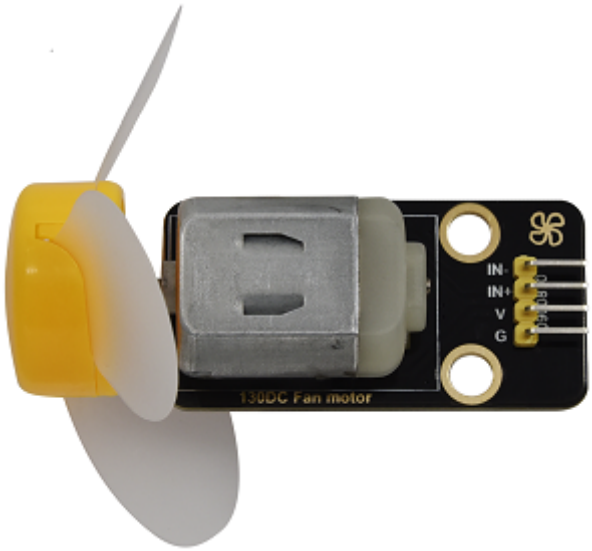
Project 7.1 Control the Fan

1. Description

In this project, we will learn how to make a small fan.

2. Component Knowledge

The small fan uses a 130 DC motor and safe fan blades. You can use PWM output to control the fan speed.



3. Control Method

Two pins are required to control the motor of the fan, one for INA and two for INB. The PWM value range is 0~255. When the PWM output of the two pins is different, the fan can rotate.

INA - INB <= -45	**Rotate clockwise**
INA - INB >= 45	Rotate anticlockwise
INA == 0, INB == 0	Stop

4. Control Pins

INA	**19**
INB	18

5. Test Code

```
#define fanPin1 19
#define fanPin2 18

void setup() {
  pinMode(fanPin1, OUTPUT);
```

```
  pinMode(fanPin2, OUTPUT);
}

void loop() {
  digitalWrite(fanPin1, LOW); //pwm = 0
  analogWrite(fanPin2, 180);
  delay(3000);
  digitalWrite(fanPin1, LOW);
  digitalWrite(fanPin2, LOW);
  delay(1000);
  digitalWrite(fanPin1, HIGH); //pwm = 255
  analogWrite(fanPin2, 210);
  delay(3000);
  digitalWrite(fanPin1, LOW);
  digitalWrite(fanPin2, LOW);
  delay(1000);

}
```

6. Test Result

The fan will rotate clockwise and anticlockwise at different speeds.

Project 7.2 Switch On or Off the Fan

One button switches the fan on and the other button controls the speed of the fan.

1. Test Code

```
#define fanPin1 19    // Fan control pin 1
#define fanPin2 18    // Fan control pin 2
#define btn1 16       // Button 1 pin
#define btn2 27       // Button 2 pin

int btn_count = 0;    // Counter for button 1 presses
int btn_count2 = 0;   // Counter for button 2 presses
int speed_val = 130;  // Initial fan speed (PWM value)

void setup() {
  Serial.begin(9600);
  pinMode(btn1, INPUT);
  pinMode(btn2, INPUT);
  pinMode(fanPin1, OUTPUT);
  pinMode(fanPin2, OUTPUT);
}

void loop() {
  boolean btn1_val = digitalRead(btn1);

  // Button 1 (Power/Speed Control) handling
  if(btn1_val == 0) // If button is pressed
  {
    delay(10);  // Debounce delay
    if(btn1_val == 0) // Confirm button press
    {
      boolean btn_state = 1;
```

```
    while(btn_state == 1) // Wait for button release
    {
      boolean btn_val = digitalRead(btn1);
      if(btn_val == 1)  // If button released
      {
        btn_count++;    // Increment press counter
        Serial.println(btn_count);
        btn_state = 0;  // Exit loop
      }
    }
  }

  boolean power_state = btn_count % 2; // Toggle power state (0 or 1)

  while(power_state == 1) // While fan is on
  {
    digitalWrite(fanPin1, LOW);  // Set direction
    analogWrite(fanPin2, speed_val); // Set speed

    // Button 2 (Speed Adjustment) handling
    boolean btn2_val = digitalRead(btn2);
    if(btn2_val == 0) // If speed button pressed
    {
      delay(10); // Debounce delay
      if(btn2_val == 0) // Confirm press
      {
        boolean btn_state2 = 1;
        while(btn_state2 == 1) // Wait for release
        {
          boolean btn2_val = digitalRead(btn2);
          if(btn2_val == 1) // If released
          {
            btn_count2++; // Increment speed level
            if(btn_count2 > 3) // Cycle through 1-3
            {
              btn_count2 = 1;
            }

            // Set speed based on count
            switch(btn_count2)
            {
              case 1:
                speed_val = 130; // Low speed
                Serial.println(speed_val);
                break;
              case 2:
                speed_val = 180; // Medium speed
                Serial.println(speed_val);
                break;
              case 3:
                speed_val = 230; // High speed
                Serial.println(speed_val);
                break;
            }
            btn_state2 = 0;
          }
        }
      }
    }
```

```
      }

      // Check for power off
      btn1_val = digitalRead(btn1);
      if(btn1_val == 0) // If power button pressed
      {
        delay(10); // Debounce delay
        if(btn1_val == 0) // Confirm press
        {
          digitalWrite(fanPin1, LOW); // Stop fan
          analogWrite(fanPin2, 0);
          power_state = 0;  // Exit fan control loop
        }
      }
    }
  }
}
```

2. Test Result

Click button 1, the fan starts to rotate, click button 2, the speed can be adjusted(there are three different speeds), press the button 1 again, the fan stops.

Project 8.1 Display Characters

1. Description

As we all know, screen is one of the best ways for people to interact with electronic devices.

2. Component Knowledge

1602 is a line that can display 16 characters. There are two lines, which use IIC communication protocol.

3. Control Pins

SDA	SDA
SCL	SCL

4. Test Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);

void setup(){
  mylcd.init();
  mylcd.backlight();
}

void loop(){
  mylcd.setCursor(0, 0);
  mylcd.print("hello");
  mylcd.setCursor(0, 1);
  mylcd.print("keyestudio");
  //mylcd.clear();
}
```

5. Test Result

The first line of the LCD1602 shows hello and the second line shows keyestudio.

Project 8.2 Dangerous Gas Alarm

1. Description

When a gas sensor detects a high concentration of dangerous gas, the buzzer will sound an alarm and the display will show dangerous.

2. Component Knowledge

MQ2 Smoke Sensor: It is a gas leak monitoring device for homes and factories, which is suitable for liquefied gas, benzene, alkyl, alcohol, hydrogen as well as smoke detection. Our sensor leads to digital pin D and analog output pin A, which is connected to D as a digital sensor in this project.

3. Test Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Initialize LCD with I2C address 0x27, 16 columns and 2 rows
LiquidCrystal_I2C mylcd(0x27, 16, 2);

#define gasPin 23    // Gas sensor input pin
```

```
#define buzPin 25    // Buzzer output pin

// State flags for LCD display updates
boolean dangerDisplayed = 1;
boolean safetyDisplayed = 1;

void setup() {
  Serial.begin(9600);

  // Initialize LCD
  mylcd.init();
  mylcd.backlight();

  // Set pin modes
  pinMode(buzPin, OUTPUT);
  pinMode(gasPin, INPUT);

  // Display initial message
  mylcd.setCursor(0, 0);
  mylcd.print("safety");
}

void loop() {
  boolean gasVal = digitalRead(gasPin);  // Read gas sensor value
  Serial.println(gasVal);

  if(gasVal == 0)  // If dangerous gas detected
  {
    while(dangerDisplayed == 1)  // Update display if needed
    {
      mylcd.clear();
      mylcd.setCursor(0, 0);
      mylcd.print("dangerous");
      dangerDisplayed = 0;
      safetyDisplayed = 1;
    }

    // Sound alarm buzzer (short pulses)
    digitalWrite(buzPin, HIGH);
    delay(1);
    digitalWrite(buzPin, LOW);
    delay(1);
  }
  else  // No dangerous gas detected
  {
    digitalWrite(buzPin, LOW);  // Ensure buzzer is off

    while(safetyDisplayed == 1)  // Update display if needed
    {
      mylcd.clear();
      mylcd.setCursor(0, 0);
      mylcd.print("safety");
      dangerDisplayed = 1;
      safetyDisplayed = 0;
    }
  }
}
```

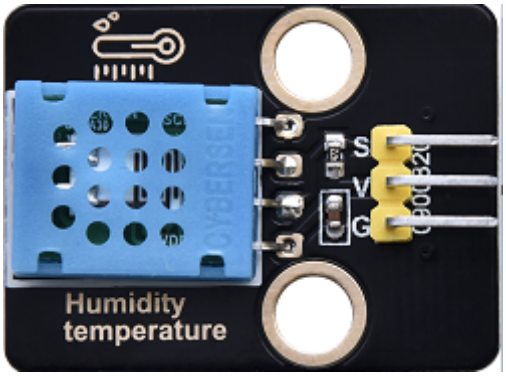
4. Test Result

The screen displays "safety" in normal state. However, when the gas sensor detects some dangerous gases, such as carbon monoxide, at a certain concentration, the buzzer will sound an alarm and the screen displays "dangerous".

Project 9 Temperature and Humidity Tester

1. Component Knowledge

Its communication mode is serial data and single bus. The temperature measurement range is -20 ~ +60℃, accuracy is ±2℃. However, the humidity range is 5 ~ 95%RH, the accuracy is ±5%RH.



2. Control Pin

Temperature and Humidity Sensor	17

3. Test Code

```
//*****************************************************************************
****
/*
 * Filename    : xht11
 * Description : Read the temperature and humidity values of XHT11.
 * Auther      : http//www.keyestudio.coml
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);
#include "xht11.h"
xht11 xht(17);

unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are
received, not the parity bits
void setup() {
  Serial.begin(9600);//Start the serial port monitor and set baud rate to 9600
  mylcd.init();
  mylcd.backlight();
}

void loop() {
  if (xht.receive(dht)) { //Returns true when checked correctly
    Serial.print("RH:");
```

```
    Serial.print(dht[0]); //The integral part of humidity, DHT [1] is the
fractional part
    Serial.print("%  ");
    Serial.print("Temp:");
    Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the
fractional part
    Serial.println("C");

    mylcd.setCursor(0, 0);
    mylcd.print("T = ");
    mylcd.print(dht[2]);
    mylcd.setCursor(0, 1);
    mylcd.print("H = ");
    mylcd.print(dht[0]);
    //mylcd.clear();
    delay(200);
  } else {    //Read error
    Serial.println("sensor error");
  }
  delay(1000);  //It takes 1000ms to wait for the device to read
}
//*************************************************************************************
****
```

4. Test Result

The LCD1602 displays the temperature (T = ** ° C) and humidity (H = ** %RH). When you breathe into the T/H sensor, you can see that the humidity rises.

Project 10 Open the Door

1. Component Knowledge

Radio frequency identification, the card reader is composed of a radio frequency module and a high-level magnetic field. The Tag transponder is a sensing device, which doesn't contain a battery. It only contains tiny integrated circuit chips and media for storing data and antennas for receiving and transmitting signals.

To read the data in the tag, first put it into the reading range of the card reader. The reader will generate a magnetic field, which can produce electricity according to Lenz's law, then the RFID tag will supply power, thereby activating the device.

2. Control Pins

Use IIC communication

SDA	SDA
SCL	SCL

3. Test Code

```
//*****************************************************************************
****
/*
 * Filename    : RFID
 * Description : RFID reader UID
 * Auther      : http//www.keyestudio.com
*/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);
#include <ESP32Servo.h>
Servo myservo;
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPIO21 and GPIO22 of ESP32
// 0x28 is the i2c address of SDA, if doesn't match, please check your address
with i2c.
MFRC522 mfrc522(0x28);   // create MFRC522.
#define servoPin  13
#define btnPin 16
boolean btnFlag = 0;

String password = "";

void setup() {
  Serial.begin(115200);           // initialize and PC's serial communication
  mylcd.init();
  mylcd.backlight();
  Wire.begin();                   // initialize I2C
  mfrc522.PCD_Init();             // initialize MFRC522
  ShowReaderDetails();            // dispaly PCD - MFRC522 read carder
  Serial.println(F("Scan PICC to see UID, type, and data blocks..."));

    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
    ESP32PWM::allocateTimer(3);
    myservo.setPeriodHertz(50);    // standard 50 hz servo
    myservo.attach(servoPin, 1000, 2000); // attaches the servo on pin 18 to the
servo object
    // using default min/max of 1000us and 2000us
    // different servos may require different min/max settings
    // for an accurate 0 to 180 sweep

  mylcd.setCursor(0, 0);
  mylcd.print("Card");
```

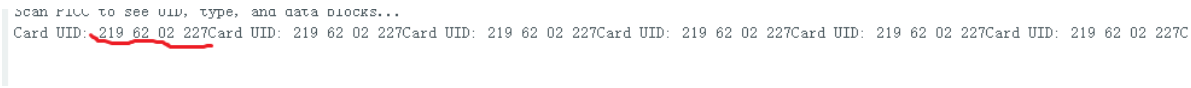
```
}

void loop() {
  //
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
    delay(50);
    password = "";
    if(btnFlag == 1)
    {
      boolean btnVal = digitalRead(btnPin);
      if(btnVal == 0)  //If door close button is pressed (active-low)
      {
        Serial.println("close");
        mylcd.setCursor(0, 0);
        mylcd.print("close");
        myservo.write(0);
        btnFlag = 0;
      }
    }
    return;
  }

  // select one of door cards. UID and SAK are mfrc522.uid.

  // save UID
  Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    //Serial.print(mfrc522.uid.uidByte[i], HEX);
    Serial.print(mfrc522.uid.uidByte[i]);
    password = password + String(mfrc522.uid.uidByte[i]);
  }
  if(password == "")  //Card number is correct,open the door
  {
    Serial.println("open");
    mylcd.setCursor(0, 0);
    mylcd.clear();
    mylcd.print("open");
    myservo.write(180);
    password = "";
    btnFlag = 1;
  }
  else   //Card number error,dispaly error
  {
    password = "";
    mylcd.setCursor(0, 0);
    mylcd.print("error");
  }
  //Serial.println(password);
}

void ShowReaderDetails() {
  //  attain the MFRC522 software
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
    Serial.print(F(" = v1.0"));
```

```
  else if (v == 0x92)
    Serial.print(F(" = v2.0"));
  else
    Serial.print(F(" (unknown)"));
  Serial.println("");
  // when returning to 0x00 or 0xFF, may fail to transmit communication signals
  if ((v == 0x00) || (v == 0xFF)) {
    Serial.println(F("WARNING: Communication failure, is the MFRC522 properly
connected?"));
  }
}
//**************************************************************************************
****
```

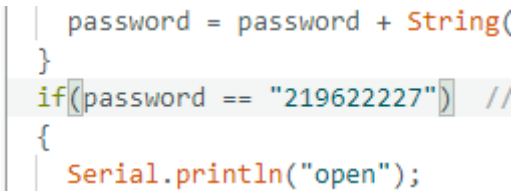
4. Test Result

Upload the code, display "Card" on the LCD1602, open the serial monitor, and set the baud rate to "115200".

Close the provided card to the RFID induction area, display "error" on the LCD1602,but the serial monitor output is as shown in the figure:



Input the "Card UID" from the image into the position shown in the figure (remove spaces in "Card UID" and in the serial monitor's **Card UID**, remove leading **0** only if it appears **before any digits** (e.g., `" 0123"` → `"123"`), but keep **0** if it follows a number (e.g., `"601"` remains `"601"`).):



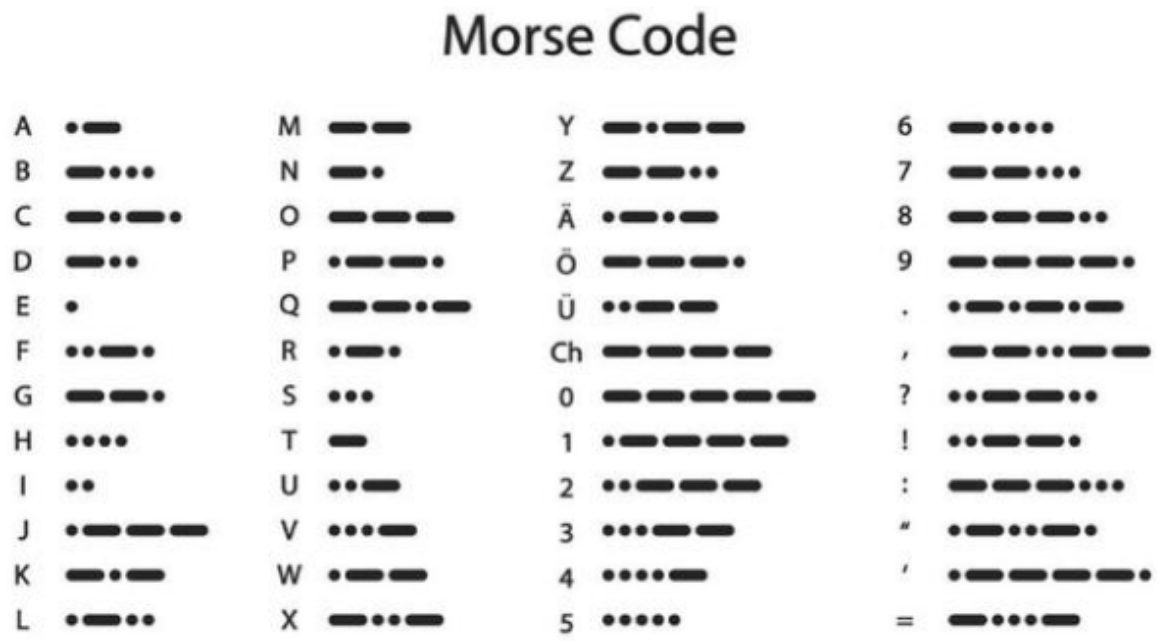
Upload the code,close the provided card to the RFID induction area,the door will turn and open, and LCD1602 shows "open".

Click button 1 and the door turns and closes. However, when swiping another blue induction block, the LCD1602 shows "Error".

Project 11 Morse Code Open the Door

Morse code, also known as Morse password, is an on-again, off-again signal code that expresses different letters, numbers, and punctuation marks in different sequences. Now we use it as our password gate.

The Morse code corresponds to the following characters:



1. Description

We use K —•— as the correct password. What's more, there is a button library file OneButton, which is very simple to click, double click, long press and other functions. For Morse password, click is".", long press and release is "-".

2. Test Code

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);
#include "OneButton.h"
// Setup a new OneButton on pin 16.
OneButton button1(16, true);
// Setup a new OneButton on pin 27.
OneButton button2(27, true);
#include <ESP32Servo.h>
Servo myservo;
int servoPin = 13;
String password = "";
String correct_p = "-.-";  //password

// setup code here, to run once:
void setup() {
  Serial.begin(115200);
  mylcd.init();
  mylcd.backlight();
  // link the button 1 functions.
  button1.attachClick(click1);
  button1.attachLongPressStop(longPressStop1);
  // link the button 2 functions.
  button2.attachClick(click2);
  button2.attachLongPressStop(longPressStop2);

    // Allow allocation of all timers
    ESP32PWM::allocateTimer(0);
    ESP32PWM::allocateTimer(1);
    ESP32PWM::allocateTimer(2);
```

```
    ESP32PWM::allocateTimer(3);
    myservo.setPeriodHertz(50);    // standard 50 hz servo
    myservo.attach(servoPin, 1000, 2000); // attaches the servo on pin 18 to the
servo object
    // using default min/max of 1000us and 2000us
    // different servos may require different min/max settings
    // for an accurate 0 to 180 sweep

  mylcd.setCursor(0, 0);
  mylcd.print("Enter password");
}

void loop() {
  // keep watching the push buttons:
  button1.tick();
  button2.tick();
  delay(10);
}

// ----- button 1 callback functions
// This function will be called when the button1 was pressed 1 time (and no 2.
button press followed).
void click1() {
  Serial.print(".");
  password = password + '.';
  mylcd.setCursor(0, 1);
  mylcd.print(password);
} // click1

// This function will be called once, when the button1 is released after beeing
pressed for a long time.
void longPressStop1() {
  Serial.print("-");
  password = password + '-';
  mylcd.setCursor(0, 1);
  mylcd.print(password);
} // longPressStop1

// ... and the same for button 2:
void click2() {
  Serial.println(password);
  if(password == correct_p)
  {
    myservo.write(180);  //open the door if the password correct
    mylcd.clear();
    mylcd.setCursor(0, 0);
    mylcd.print("open");
  }
  else
  {
    mylcd.clear();
    mylcd.setCursor(0, 0);
    mylcd.print("error");
    delay(2000);
    mylcd.clear();
    mylcd.setCursor(0, 0);
    mylcd.print("input again");
  }
```

```
  password = "";
} // click2

void longPressStop2() {
  //Serial.println("Button 2 longPress stop");
   myservo.write(0);  //open door
   mylcd.clear();
   mylcd.setCursor(0, 0);
   mylcd.print("close");
} // longPressStop2
```

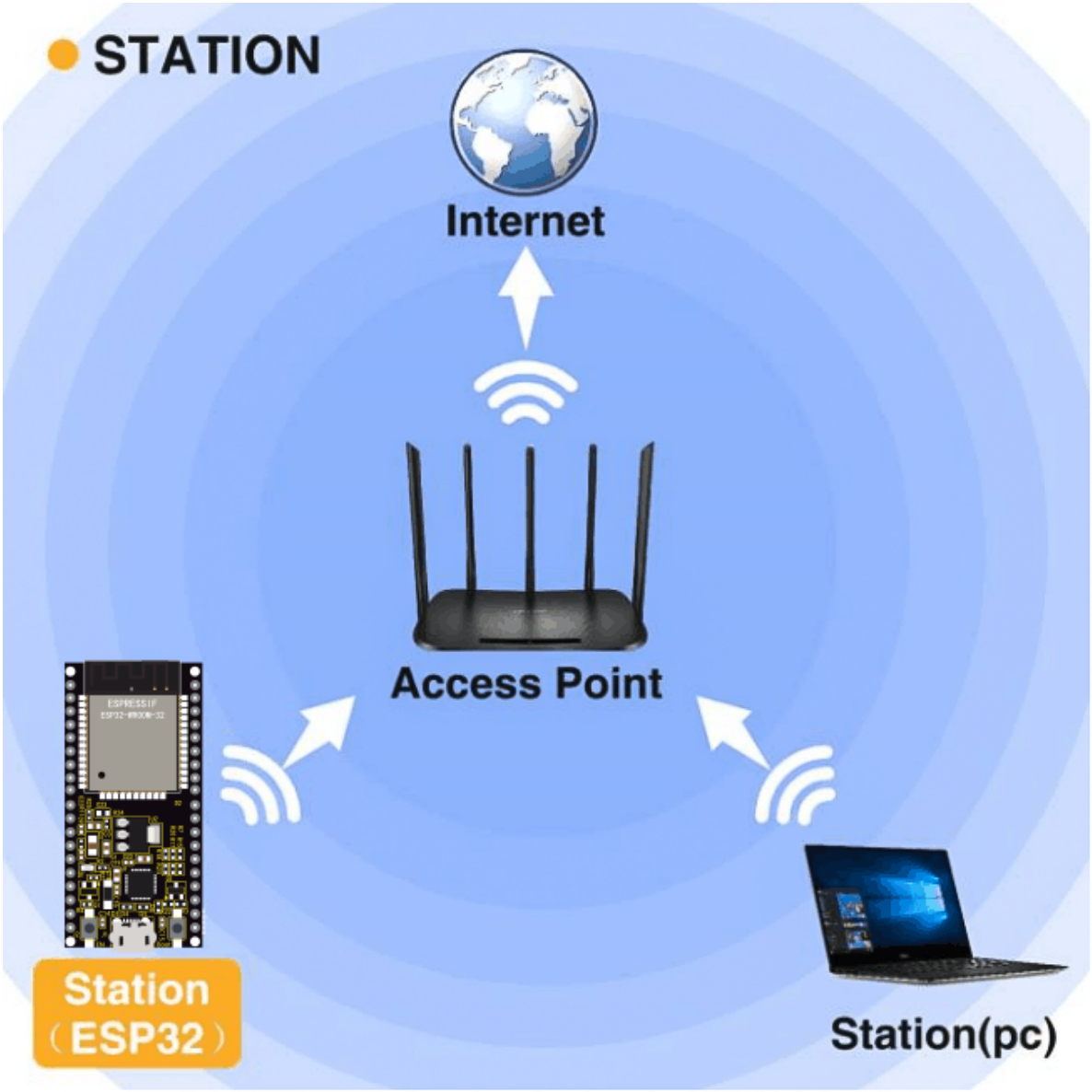
3. Test Result

At first, the LCD1602 displays "Enter password", then click or long press button 1 to tap the password. If we input the correct password "-.-", then click button 2, the door will open, and the LCD1602 will display "open".

If other incorrect passwords are entered, the door will not move, the LCD1602 will display “error” and then “enter again” 2s later. Furthermore, long press button 2 can close the door.

Project 12.1 Smart Home

The easiest way to access the Internet is to use a WiFi to connect. The ESP32 main control board comes with a WiFi module, making our smart home accessible to the Internet easily.



1. Description

We connect the smart home to a LAN, which is the WiFi in your home or the hot spot of your phone. After the connection is successful, an address will be assigned, which can be used for communication. We will print the assigned address in the serial monitor.

2. Test Code

Note: ssiD and password in the code should be filled with your own WiFi name and password.


```
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>

// Network Configuration
const char* ssid = "LieBaoWiFi359";
const char* password = "wmbd315931";
WiFiServer server(80);

// Global Variables
String requestPath = "/";  // Stores the HTTP request path

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

  // Connect to WiFi
  Serial.println("\nConnecting to WiFi...");
  WiFi.begin(ssid, password);

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

  // Network information
  Serial.println("\nWiFi connected");
```

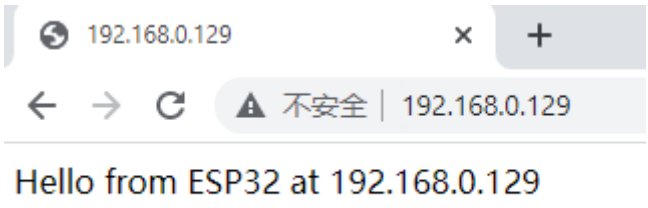
```
  printNetworkInfo();

  // Start server and mDNS
  server.begin();
  if (!MDNS.begin("esp32")) {
    Serial.println("Error setting up MDNS responder!");
  }
  MDNS.addService("http", "tcp", 80);
  Serial.println("HTTP server started");
}

void loop() {
  WiFiClient client = server.available();

  if (!client) {
    return;
  }

  // Wait for client data
  while (client.connected() && !client.available()) {
    delay(1);
  }

  // Read HTTP request
  String request = client.readStringUntil('\r');
  parseHttpRequest(request);

  // Handle request
  String response;
  if (requestPath == "/") {
    response = buildHomepageResponse();
    Serial.println("Serving homepage");
  } else {
    response = buildNotFoundResponse();
    Serial.println("Unknown request: " + requestPath);
  }

  // Send HTTP response
  client.println(response);
  client.stop();

  // Small delay between requests
  delay(100);
}

// Helper Functions
void printNetworkInfo() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void parseHttpRequest(String req) {
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);

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

```
    Serial.print("Invalid request: ");
    Serial.println(req);
    requestPath = "/404";
    return;
  }

  requestPath = req.substring(addr_start + 1, addr_end);
  Serial.println("Requested path: " + requestPath);
}

String buildHomepageResponse() {
  IPAddress ip = WiFi.localIP();
  String ipStr = String(ip[0]) + '.' + ip[1] + '.' + ip[2] + '.' + ip[3];

  String html = "HTTP/1.1 200 OK\r\n";
  html += "Content-Type: text/html\r\n";
  html += "Connection: close\r\n";
  html += "\r\n";
  html += "<!DOCTYPE HTML>\n";
  html += "<html><head><title>ESP32 Web Server</title></head>\n";
  html += "<body><h1>Hello from ESP32</h1>\n";
  html += "<p>IP Address: " + ipStr + "</p>\n";
  html += "</body></html>\n";

  return html;
}

String buildNotFoundResponse() {
  String html = "HTTP/1.1 404 Not Found\r\n";
  html += "Content-Type: text/html\r\n";
  html += "Connection: close\r\n";
  html += "\r\n";
  html += "<!DOCTYPE HTML>\n";
  html += "<html><head><title>404 Not Found</title></head>\n";
  html += "<body><h1>404</h1><p>Page not found</p></body></html>\n";

  return html;
}
```

3. Test Result

If the WiFi is connected successfully, the serial monitor will print out the assigned IP address.

Open a browser to access the IP address, then we will read the contents of the string S sent out by the client.println(s); in the code.



Project 12.2 Control Smart Home

1. Description

In this project, we will learn how to realize different functions of the smart home through accessing different strings under the address. There is a LCD screen that can print out the IP address, which is much more convenient.

2. Test Code

```
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>

String item = "0";
const char* ssid = "LieBaoWiFi359";
const char* password = "wmbd315931";
WiFiServer server(80);

#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);
//#include <analogWrite.h>
#define fanPin1 19
#define fanPin2 18
#define led_y 12  //Define yellow LED pin as 12

void setup() {
  Serial.begin(115200);
  mylcd.init();
  mylcd.backlight();
  pinMode(led_y, OUTPUT);
  pinMode(fanPin1, OUTPUT);
  pinMode(fanPin2, OUTPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
```

```
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
  mylcd.setCursor(0, 0);
  mylcd.print("ip:");
  mylcd.setCursor(0, 1);
  mylcd.print(WiFi.localIP());  //LCD displays IP address
}

void loop() {
  WiFiClient client = server.available();
  if (!client) {
      return;
  }
  while(client.connected() && !client.available()){
      delay(1);
  }
  String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1) {
      Serial.print("Invalid request: ");
      Serial.println(req);
      return;
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")  //Browser can read the information sent by client.println(s)
when accessing the address
  {
      IPAddress ip = WiFi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) +
'.' + String(ip[3]);
      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>Hello from ESP32 at ";
      s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);  //Send the content of string S. When accessing the E-
smart home address using a browser, the information can be read.
  }
  if(req == "/led/on") //Browser accesses IP address/led/on
  {
    client.println("turn on the LED");
    digitalWrite(led_y, HIGH);
  }
  if(req == "/led/off") //Browser accesses IP address/led/off
  {
    client.println("turn off the LED");
    digitalWrite(led_y, LOW);
  }
  if(req == "/fan/on") //Browser accesses IP address/fan/on
  {
    client.println("turn on the fan");
    digitalWrite(fanPin1, LOW); //pwm = 0
    analogWrite(fanPin2, 180);
  }
```

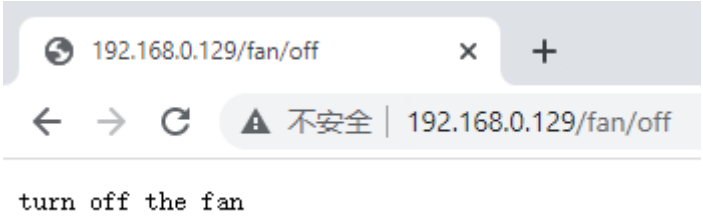
```
  if(req == "/fan/off") //Browser accesses IP address/fan/on
  {
    client.println("turn off the fan");
    digitalWrite(fanPin1, LOW); //pwm = 0
    analogWrite(fanPin2, 0);
  }
  //client.print(s);
  client.stop();
}
```

3. Test Result

If the smart home is successfully connected to WiFi, the LCD screen will display the assigned address.

Accessing address must add / led/on when using the browser, such as my address is 192.168.0.129/ led/on. Then the smart home LED lights will be turned on, if accessing 192.168.0.129/ led /off, then the LED lights will be off.

When the browser accesses 192.168.0.129/fan/ on, the fan of the smart home will be turned on and at 192.168.0.129/fan/ off will be turned off.

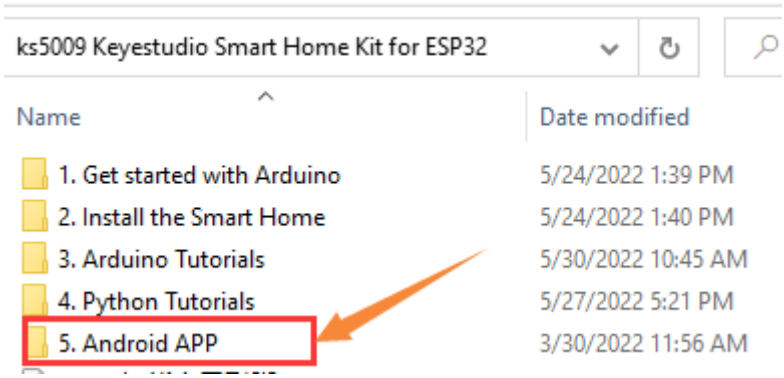


Project 13.1: Mobile Phone APP test

Download APP

Android APP:

The Android apk installation package is available in our resource pack, as shown below:

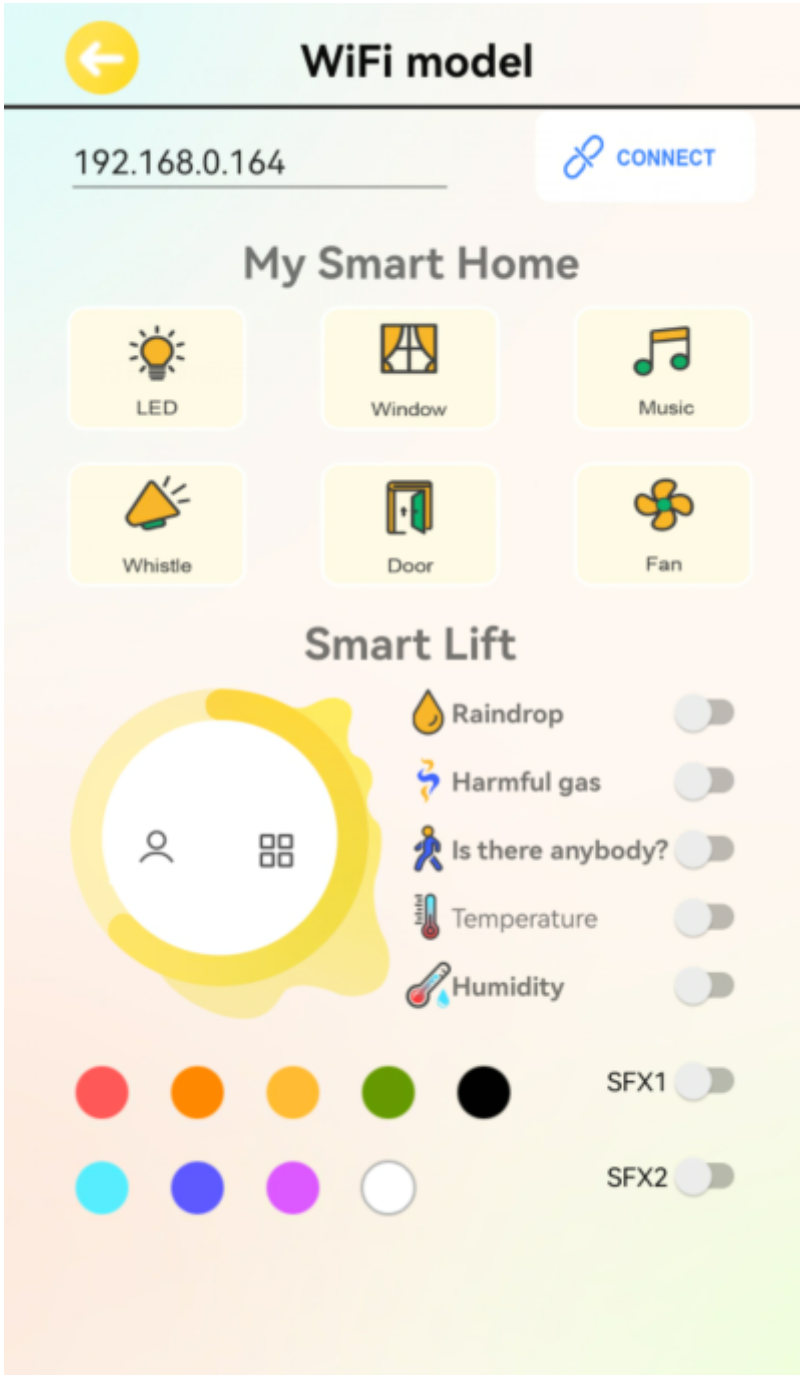


Download from Google play:

Please search for keyes IoT home on Google play to download it.

Icon:

APP Interface



Download iOS APP

Please search for keyes IoT home on APP Store to download it.

1. Description

We will use APP to control the smart home LED lights and fan switches.

2. Test Code

```
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>

String item = "0";
const char* ssid = "LieBaoWiFi359";
const char* password = "wmbd315931";
WiFiServer server(80);
```

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);
//#include <analogWrite.h>
#define fanPin1 19
#define fanPin2 18
#define led_y 12                                        // Define yellow LED pin as
12

void setup()
{
  Serial.begin(115200);
  mylcd.init();
  mylcd.backlight();
  pinMode(led_y, OUTPUT);
  pinMode(fanPin1, OUTPUT);
  pinMode(fanPin2, OUTPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
  mylcd.setCursor(0, 0);
  mylcd.print("ip:");
  mylcd.setCursor(0, 1);
  mylcd.print(WiFi.localIP());                          // LCD displays IP address
}

void loop()
{
  WiFiClient client = server.available();
  if (!client)
  {
      return;
  }
  while(client.connected() && !client.available())
  {
      delay(1);
  }
  String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1)
  {
      Serial.print("Invalid request: ");
      Serial.println(req);
      return;
```

```
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")                                     // Browser can read the
information sent by client.println(s) when accessing the address
  {
      IPAddress ip = WiFi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) +
'.' + String(ip[3]);
      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>Hello from ESP32 at ";
      s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);                              // Send the content of
string S. When accessing the E-smart home address using a browser, the
information can be read.
  }
  if(req == "/led/on")                                // Browser accesses IP
address/led/on
  {
    client.println("turn on the LED");
    digitalWrite(led_y, HIGH);
  }
  if(req == "/led/off")                               // Browser accesses IP
address/led/off
  {
    client.println("turn off the LED");
    digitalWrite(led_y, LOW);
  }
  if(req == "/fan/on")                                // Browser accesses IP
address/fan/on
  {
    client.println("turn on the fan");
    digitalWrite(fanPin1, LOW);                       // pwm = 0
    analogWrite(fanPin2, 180);
  }
  if(req == "/fan/off")                               // Browser accesses IP
address/fan/on
  {
    client.println("turn off the fan");
    digitalWrite(fanPin1, LOW);                       // pwm = 0
    analogWrite(fanPin2, 0);
  }
  //client.print(s);
  client.stop();
}
```

3. Test Result

1. Open the APP and select WIFI

2. APP controls LED and the fan

The mobile phone and the smart home must share the same WiFi, or the smart home connects to the hotspot of the mobile phone.

APP input IP address (LCD1602 displays the assigned IP address), then click connect, the connection is successful if ESP32 IP: 192.168..... is displayed.

Next, you can click the LED, then the smart home LED will be turned on. Click the fan button and the fan will be turned on, as shown below:

Project 13.2 IoT Smart Home

1. Description

The IOT smart home connects to the family WiFi through

WiFi, and the mobile phone used for operation should also be connected to the same WiFi.

What's more, the smart home also can connect to the hotspot of the mobile phone. If the connection is successful, the LCD1602 will display the IP address. Using the phone APP to input the corresponding IP for communication is enable to realize the APP control of various functions of the smart home.

2. Test Code

```
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>
#include <Adafruit_NeoPixel.h>
#define LED_PIN    26
#define LED_COUNT 4                                    // Number of NeoPixels
attached
Adafruit_NeoPixel strip(LED_COUNT, LED_PIN, NEO_GRB + NEO_KHZ800);
```

```
String item = "0";
const char* ssid = "LieBaoWiFi359";
const char* password = "wmbd315931";
WiFiServer server(80);

#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C mylcd(0x27,16,2);
#include "xht11.h"
xht11 xht(17);

#include <BuzzerESP32.h>
#define buzzer_pin 25
BuzzerESP32 buzzer(buzzer_pin);                              // GPIO25

#define waterPin 34
#define fanPin1 19
#define fanPin2 18
#define led_y 12                                             // Yellow LED pin
definition
#define gasPin 23
#define pyroelectric 14

unsigned char dht[4] = {0, 0, 0, 0};                        // Only first 32 bits
received (no parity bits)

// Servo channels
int channel_PWM = 13;
int channel_PWM2 = 10;
int freq_PWM = 50;
int resolution_PWM = 10;
const int PWM_Pin1 = 5;
const int PWM_Pin2 = 13;

void setup()
{
  Serial.begin(115200);
  mylcd.init();
  mylcd.backlight();
  pinMode(led_y, OUTPUT);
  pinMode(fanPin1, OUTPUT);
  pinMode(fanPin2, OUTPUT);
  pinMode(waterPin, INPUT);

  buzzer.setTimbre(30);                                      // Set timbre
  buzzer.playTone(0,0);                                      // Turn off buzzer

  pinMode(gasPin, INPUT);
  pinMode(pyroelectric, INPUT);

  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
```

```
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
  mylcd.setCursor(0, 0);
  mylcd.print("ip:");
  mylcd.setCursor(0, 1);
  mylcd.print(WiFi.localIP());                              // Display IP on LCD
}

void loop()
{
  WiFiClient client = server.available();
  if (!client)
  {
      return;
  }
  while(client.connected() && !client.available())
  {
      delay(1);
  }
  String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1)
  {
      Serial.print("Invalid request: ");
      Serial.println(req);
      return;
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")                                          // Browser can read
information sent by client.println(s)
  {
      IPAddress ip = WiFi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) +
'.' + String(ip[3]);
      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML>\r\n<html>ESP32 ip:";
      s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);                                   // Send string content -
readable when accessing smart home address
  }
  if(req == "/led/on")                                     // IP address/led/on
  {
    client.println("turn on the LED");
    digitalWrite(led_y, HIGH);
  }
  if(req == "/led/off")                                    // IP address/led/off
  {
    client.println("turn off the LED");
```

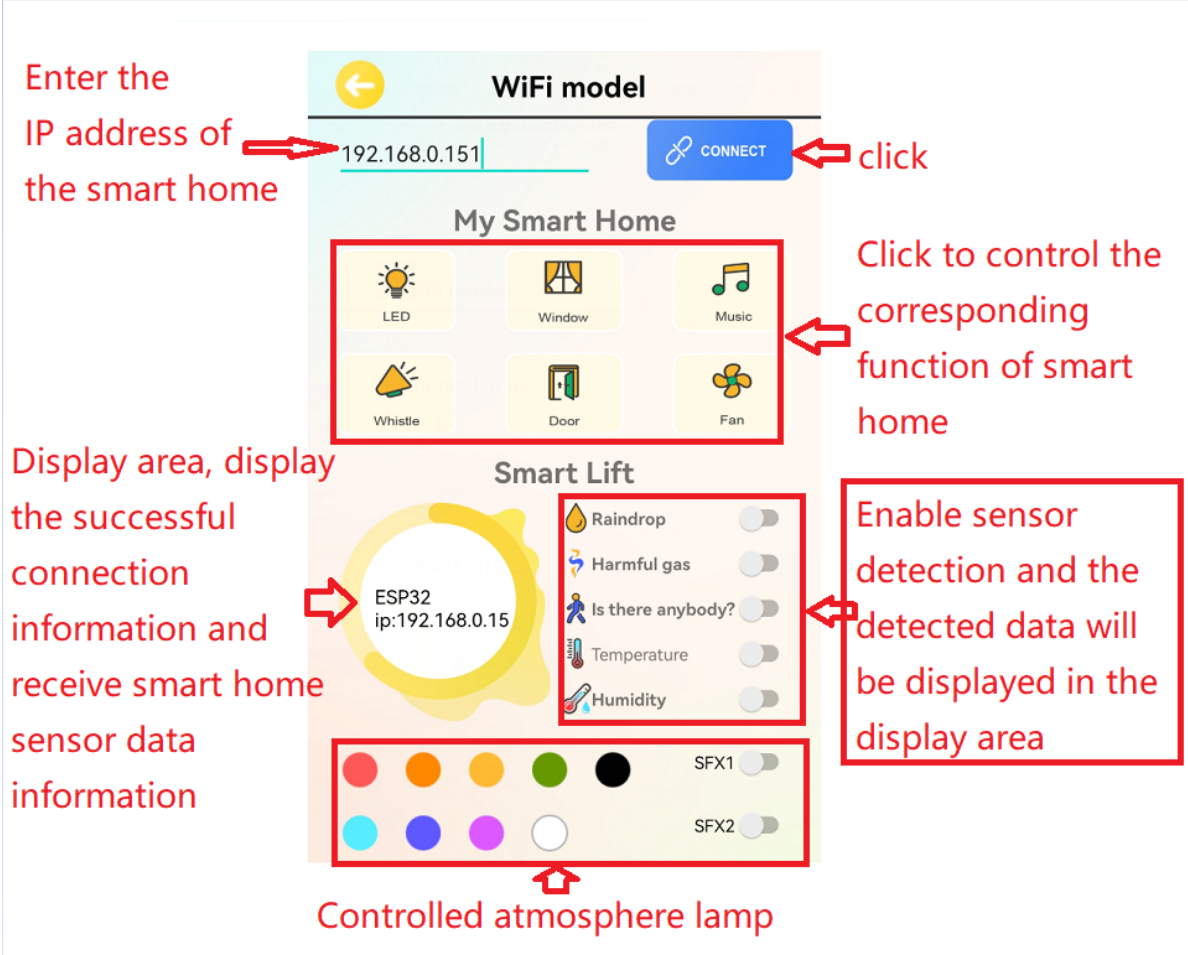
```
    digitalWrite(led_y, LOW);
  }
  if(req == "/window/on")
  {
    client.println("open the window");
    ledcWrite(channel_PWM, 100);                          // 2.5ms high pulse
(2.5/20*1024) = 180° servo angle
  }
  if(req == "/window/off")
  {
    client.println("close the window");
    ledcWrite(channel_PWM, 60);                           // 0.5ms high pulse = 0°
servo angle
  }
  if(req == "/music/on")
  {
    //client.println("play music");
  }
  if(req == "/music/off")
  {
    client.println("play music");
    birthday();
    buzzer.playTone(0,0);
  }
  if(req == "/buz/on")
  {
    client.println("buzzer");
    buzzer.playTone(392,250);
    Serial.println("1");
  }
  if(req == "/buz/off")
  {
    client.println("off");
    buzzer.playTone(0,0);
  }
  if(req == "/door/on")
  {
    client.println("open the door");
    ledcWrite(channel_PWM2, 120);
  }
  if(req == "/door/off")
  {
    client.println("close the door");
    ledcWrite(channel_PWM2, 20);
  }
  if(req == "/fan/on")
  {
    client.println("turn on the fan");
    digitalWrite(fanPin1, LOW);                           // pwm = 0
    ledcWrite(5, 100);                                    // LEDC channel 1 PWM
output = 100
  }
  if(req == "/fan/off")
  {
    client.println("turn off the fan");
    digitalWrite(fanPin1, LOW);                           // pwm = 0
    ledcWrite(5, 0);                                      // LEDC channel 1 PWM
output = 0
```

```
  }
  // Color control endpoints follow same pattern...
  if(req == "/rain/on")
  {
    int rainVal = analogRead(waterPin);
    client.println(rainVal);
  }
  if(req == "/rain/off")
  {
    client.println("off");
  }
  if(req == "/gas/on")
  {
    boolean gasVal = analogRead(gasPin);
    if(gasVal == 0)
    {
      client.println("safety");
    }
    else
    {
      client.println("dangerous");
    }
  }
  if(req == "/gas/off")
  {
    client.println("off");
  }
  if(req == "/body/on")
  {
    boolean pyroelectric_val = digitalRead(pyroelectric);
    if(pyroelectric_val == 1)
    {
      client.println("someone");
    }
    else
    {
      client.println("no one");
    }
  }
  if(req == "/body/off")
  {
    client.println("off");
  }
  if(req == "/temp/on")
  {
    if (xht.receive(dht))                                  // Returns true on
successful read
    {
      Serial.print("Temp:");
      Serial.print(dht[2]);                                // Integer part of
temperature
      Serial.println("C");
      delay(200);
    }
    else                                                   // Read error
    {
      Serial.println("sensor error");
    }
```

```
    client.println(dht[2]);
    delay(1000);                                    // Wait 1000ms for device
read
  }
  if(req == "/temp/off")
  {
    client.println("off");
  }
  if(req == "/humidity/on")
  {
    if (xht.receive(dht))                           // Returns true on
successful read
    {
      Serial.print("Humidity:");
      Serial.print(dht[0]);                         // Integer part of
humidity
      Serial.println("%");
      delay(200);
    }
    else                                            // Read error
    {
      Serial.println("sensor error");
    }
    client.println(dht[0]);
    delay(1000);                                    // Wait 1000ms for device
read
  }
  if(req == "/humidity/off")
  {
    client.println("off");
  }
}

// Birthday melody function
void birthday()
{
  buzzer.playTone(294,250);                         // Parameters: frequency,
duration
  buzzer.playTone(440,250);
  buzzer.playTone(392,250);
  buzzer.playTone(532,250);
  buzzer.playTone(494,250);
  buzzer.playTone(392,250);
  buzzer.playTone(440,250);
  buzzer.playTone(392,250);
  buzzer.playTone(587,250);
  buzzer.playTone(532,250);
  buzzer.playTone(392,250);
  buzzer.playTone(784,250);
  buzzer.playTone(659,250);
  buzzer.playTone(532,250);
  buzzer.playTone(494,250);
  buzzer.playTone(440,250);
  buzzer.playTone(698,250);
  buzzer.playTone(659,250);
  buzzer.playTone(532,250);
  buzzer.playTone(587,250);
  buzzer.playTone(532,500);
```

```
  buzzer.playTone(0,0);                                    // Turn off
}

// NeoPixel effect functions (colorWipe, rainbow, theaterChaseRainbow remain
unchanged)
```

3. Test Result



Resources

Download code, libraries and more details, please refer to the following link: https://fs.keyestudio.com/KS5009