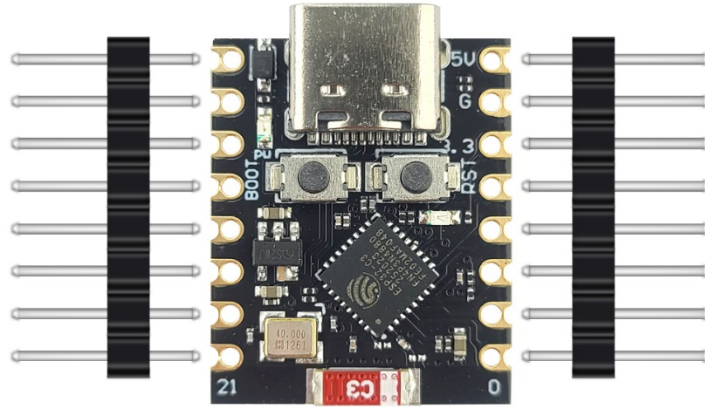


MINI ESP32 C3 Development Board



The ESP32C3SuperMini is an IoT mini development board based on the Espressif ESP32-C3 WiFi/BT dual-mode chip. The ESP32-C3 is a 32-bit RISC-V CPU containing a FPU (Floating Point Unit) for 32-bit single-precision arithmetic with powerful computational capabilities. It has excellent RF performance and supports IEEE 802.11 b/g/n WiFi and BT 5 (LE) protocols. The board comes with an external antenna to enhance signal strength for wireless applications. It also features a small and compact form factor combined with a single-sided surface mount design. It is equipped with a rich set of interfaces, with 11 digital I/Os that can be used as PWM pins and 4 analog I/Os that can be used as ADC pins, and it supports four serial interfaces, including UART, I2C, and SPI. There is also a small reset button and a bootloader mode button on the board.

Combining these features, the ESP32C3SuperMini is positioned as a high-performance, low-power, cost-effective IoT mini development board for low-power IoT applications and wireless wearable applications.

Product Parameters:

Powerful CPU: ESP32-C3, 32-bit RISC-V single-core processor, running at up to 160 MHz.

WiFi: 802.11b/g/n protocol, 2.4GHz, supports Station Mode, SoftAP Mode, SoftAP+Station Mode, and Mixed Mode.

BT: BT 5.0

Ultra-low power consumption: deep sleep power consumption of about 43μA

Rich board resources: 400KB SRAM, 384KB ROM, built-in 4Mflash.

Chip Model: ESP32C3FN4

Ultra-small size: as small as a thumb (22.52x18mm) in a classic form factor for wearables and small projects.

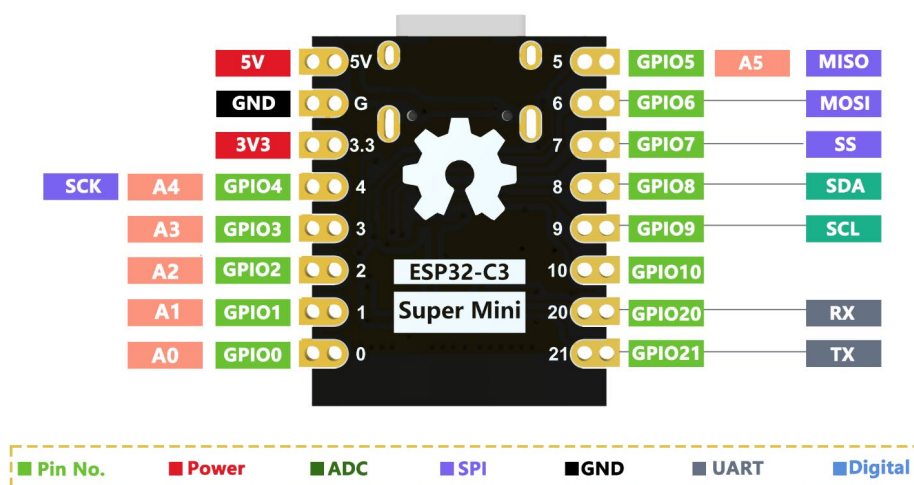
Reliable security features: cryptographic hardware gas pedal supporting AES-128/256, hash, RSA, HMAC, digital signatures and secure booting

Rich interfaces: 1xI2C, 1xSPI, 2xUART, 11xGPIO(PWM), 4xADC

Single-sided components, surface mount design

On-board LED blue light: GPIO8 pinout

Pin Definition Diagram:



Top view of the PCB showing dimensions and component labels. The overall width is 18.00mm and the overall height is 22.52mm. The width of the central component area is 15.24mm. The PCB is populated with various components including a USB connector, a microcontroller (QFN32 5x5), a crystal oscillator (SOT23-5), and several passive components. The pin headers are labeled 0 to 21. The USB connector is labeled 5V, G, 3.3, RST, and IO8. The microcontroller is labeled QFN32 5x5. The crystal oscillator is labeled SOT23-5. The passive components are labeled PWR, BOOT, and IO8.

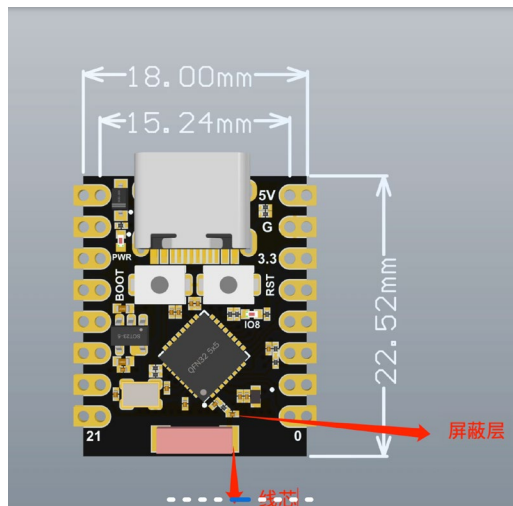
External power supply:

If you need external power supply just connect the external power supply + level to the 5V position and GND to the negative terminal. (Support 3.3 ~ 6V power supply). Remember that when connecting the external power supply, you can not access the USB, USB and external power supply can only choose one.

Note:Please be careful not to short-circuit the positive and negative poles and burn the battery and equipment when welding.

WIFI antenna:

If you want to use an external antenna, you can follow the picture below to connect an external antenna.

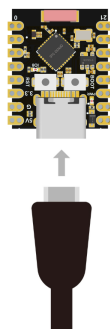


Hardware Setup:

You need to prepare the following:

- 1 ESP32C3SuperMini
- 1 PC
- 1 USB Type-C cable

Step 1. Connect the ESP32C3SuperMini to your computer via USB Type-C cable.



software setup:

Step 1. Download and install the latest version of Arduino IDE according to your operating system.

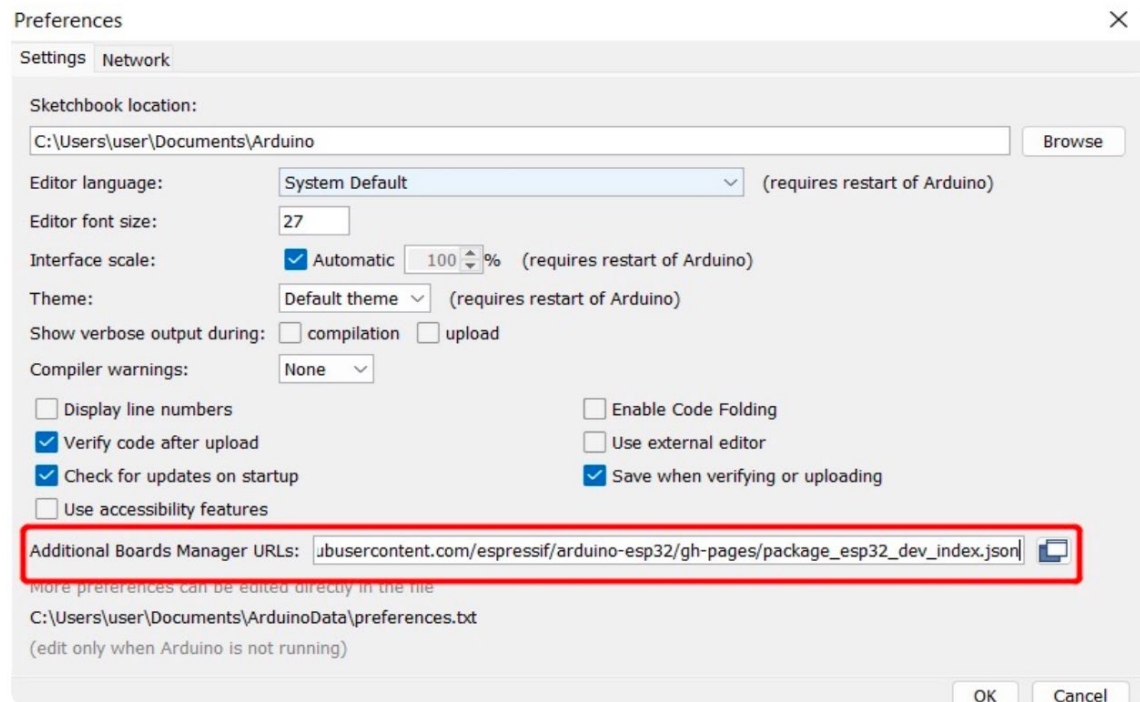


Step 2. Launch the Arduino application.

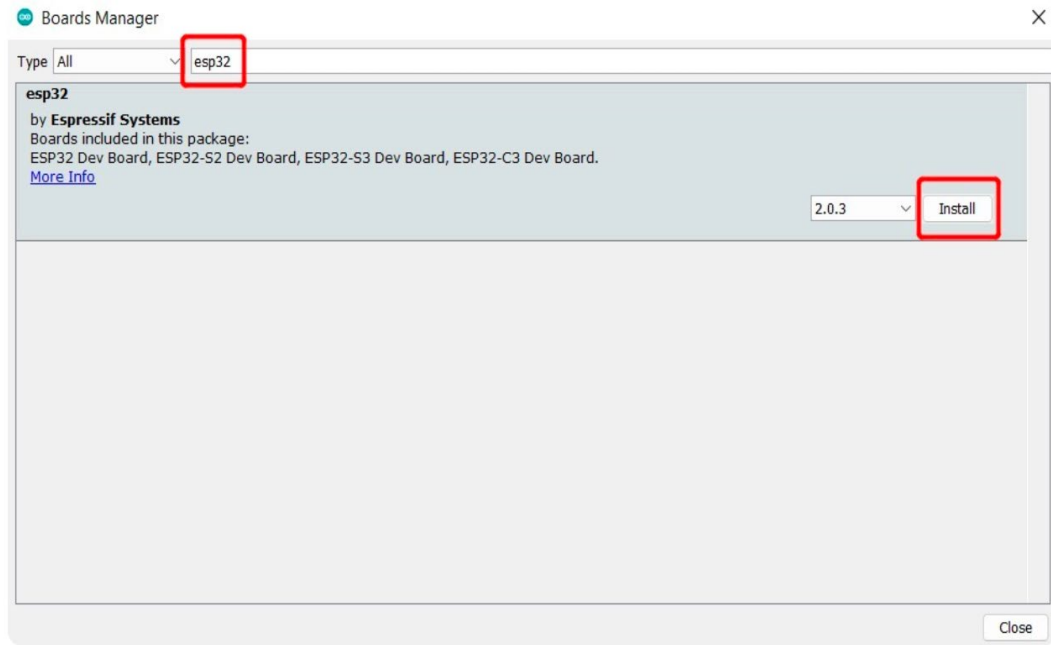
Step 3. Add the ESP32 board package to the Arduino IDE.

Navigate to File > Preferences and fill in the "Additional Boards Manager URL" using the following url:

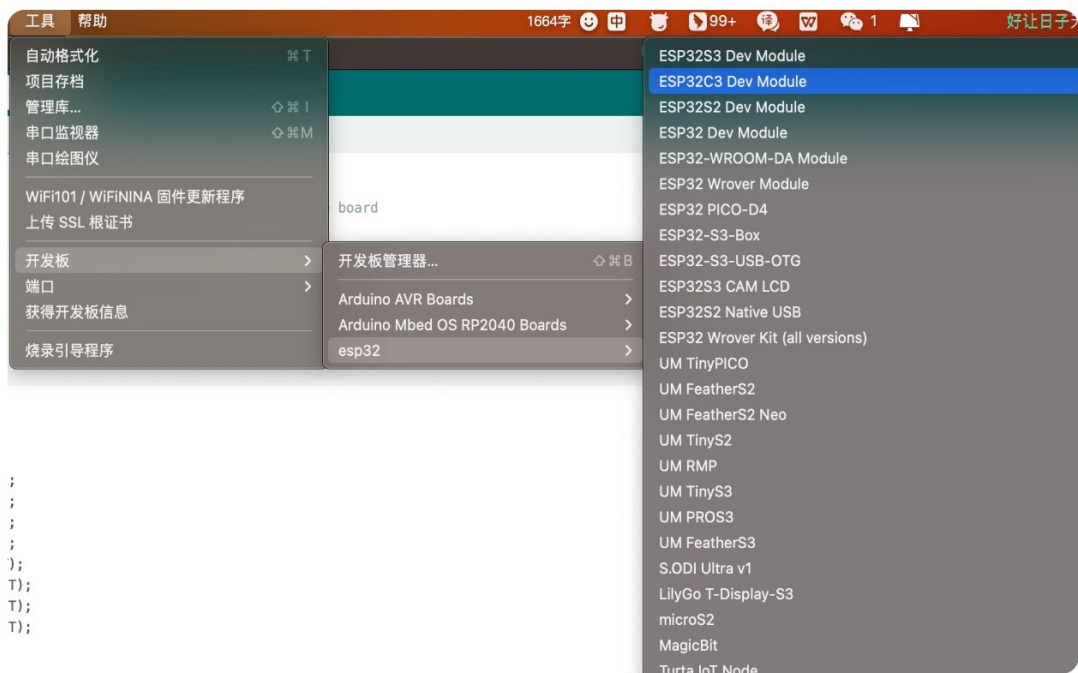
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json



Navigate to Tools > Board > Boards Manager... , enter the keyword " esp32 " in the search box, select the latest version of esp32 and install it.



Navigate to Tools > Development Boards > ESP32 Arduino and select "ESP32C3 Dev Module". The list of boards is a bit long and you will need to scroll to the bottom to get to it.



Navigate to Tools > Ports and select the serial port name of the connected ESP32C3SuperMini. This may be COM3 or later (COM1 and COM2 are usually reserved for hardware serial ports).

Flashing LED

```
1 // define led according to pin diagram
2 int led = 8;
3
4 void setup() {
5     // initialize digital pin led as an output
6     pinMode(led, OUTPUT);
7 }
8
9 void loop() {
10    digitalWrite(led, HIGH); // turn the LED off
11    delay(1000);             // wait for a second
12    digitalWrite(led, LOW);  // turn the LED on
13    delay(1000);             // wait for a second
14 }
15
```

After uploading, you will see the LED on the board blinking with a 1 second delay between each blink.

WIFI controlled LED:

```
1  /*
2   WiFi Web Server LED Blink
3
4   A simple web server that lets you blink an LED via the web.
5   This sketch will print the IP address of your WiFi Shield (once connected)
6   to the Serial monitor. From there, you can open that address in a web browser
7   to turn on and off the LED on pin 5.
8
9   If the IP address of your shield is yourAddress:
10  http://yourAddress/H turns the LED on
11  http://yourAddress/L turns it off
12
13  This example is written for a network using WPA2 encryption. For insecure
14  WEP or WPA, change the Wifi.begin() call and use Wifi.setMinSecurity() accordingly
15
16  Circuit:
17  * WiFi shield attached
18  * LED attached to pin 5
19
20  created for arduino 25 Nov 2012
21  by Tom Igoe
22
23  ported for sparkfun esp32
24  31.01.2017 by Jan Hendrik Berlin
25
26  */
27
```

```
28 #include <WiFi.h>
29
30 const char* ssid    = "Onion3"; // 设置wifi名称
31 const char* password = "asdfghjkl"; // 设置wifi密码
32
33 WiFiServer server(80);
34
35 void setup()
36 {
37     Serial.begin(115200);
38     pinMode(8, OUTPUT); // set the LED pin mode
39
40     delay(10);
41
42     // We start by connecting to a WiFi network
43
44     Serial.println();
45     Serial.println();
46     Serial.print("Connecting to ");
47     Serial.println(ssid);
48
49     WiFi.begin(ssid, password);
50
51     while (WiFi.status() != WL_CONNECTED) {
52         delay(500);
53         Serial.print(".");
54     }
```

```
55
56     Serial.println("");
57     Serial.println("WiFi connected.");
58     Serial.println("IP address: ");
59     Serial.println(WiFi.localIP());
60
61     server.begin();
62
63 }
```

```

65 void loop(){
66   WiFiClient client = server.available(); // listen for incoming clients
67
68   if (client) { // if you get a client,
69     Serial.println("New Client."); // print a message out the serial port
70     String currentLine = ""; // make a String to hold incoming data
71     while (client.connected()) { // loop while the client's connected
72       if (client.available()) { // if there's bytes to read from the client
73         char c = client.read(); // read a byte, then
74         Serial.write(c); // print it out the serial monitor
75         if (c == '\n') { // if the byte is a newline character
76
77           // if the current line is blank, you got two newline characters in a row.
78           // that's the end of the client HTTP request, so send a response:
79           if (currentLine.length() == 0) {
80             // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
81             // and a content-type so the client knows what's coming, then a blank line
82             client.println("HTTP/1.1 200 OK");
83             client.println("Content-type:text/html");
84             client.println();
85
86             // the content of the HTTP response follows the header:
87             client.print("Click <a href=\"/H\">here</a> to turn the LED on pin 8 on");
88             client.print("Click <a href=\"/L\">here</a> to turn the LED on pin 8 of");
89
90             // The HTTP response ends with another blank line:

```

```
90      // The HTTP response ends with another blank line:
91      client.println();
92      // break out of the while loop:
93      break;
94  } else {    // if you got a newline, then clear currentLine:
95      currentLine = "";
96  }
97  } else if (c != '\r') { // if you got anything else but a carriage return
98      currentLine += c;    // add it to the end of the currentLine
99  }
100
101  // Check to see if the client request was "GET /H" or "GET /L":
102  if (currentLine.endsWith("GET /H")) {
103      digitalWrite(8, HIGH);    // GET /H turns the LED on
104  }
105  if (currentLine.endsWith("GET /L")) {
106      digitalWrite(8, LOW);    // GET /L turns the LED off
107  }
108  }
109  }
110  // close the connection:
111  client.stop();
112  Serial.println("Client Disconnected.");
113  }
114  }
115
```