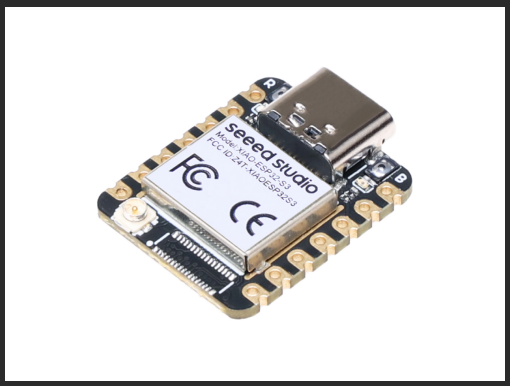
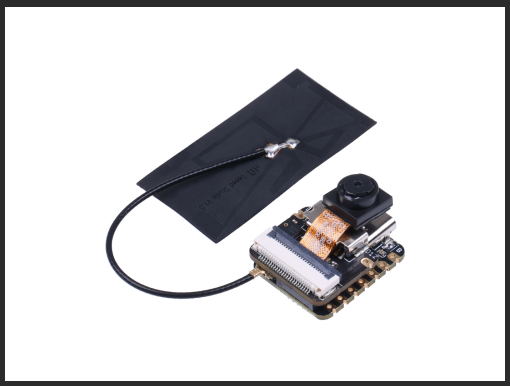
Getting Started with Seeed Studio XIAO ESP32S3 (Sense)

Seeed Studio XIAO ESP32S3	Seeed Studio XIAO ESP32S3 Sense
	
Get One Now	Get One Now

Introduction

Seeed Studio XIAO Series are diminutive development boards, sharing a similar hardware structure, where the size is literally thumb-sized. The code name "XIAO" here represents its half feature "Tiny", and the other half will be "Puissant". Seeed Studio XIAO ESP32S3 Sense integrates camera sensor, digital microphone and SD card supporting. Combining embedded ML computing power and photography capability, this development board can be your great tool to get started with intelligent voice and vision AI.

Specification

Item	Seeed Studio XIAO ESP32S3	Seeed Studio XIAO ESP32S3 Sense
Processor	ESP32-S3R8 Xtensa LX7 dual-core, 32-bit processor that operates at up to 240 MHz	
Wireless	Complete 2.4GHz Wi-Fi subsystem BLE: Bluetooth 5.0, Bluetooth mesh	
Built-in Sensors	-	OV2640 camera sensor for 1600*1200 Digital microphone
Memory	On-chip 8M PSRAM & 8MB Flash	On-chip 8M PSRAM & 8MB Flash Onboard SD Card Slot, supporting 32GB FAT
Interface	1x UART, 1x IIC, 1x IIS, 1x SPI, 11x GPIOs (PWM), 9x ADC, 1x User LED, 1x Charge LED 1x Reset button, 1x Boot button	1x UART, 1x IIC, 1x IIS, 1x SPI, 11x GPIOs (PWM), 9x ADC, 1x User LED, 1x Charge LED, 1x B2B Connector (with 2 additional GPIOs) 1x Reset button, 1x Boot button
Dimensions	21 x 17.5mm	21 x 17.5 x 15mm (with expansion board)
Power	Input voltage (Type-C): 5V Input voltage (BAT): 4.2V	
	Circuit operating Voltage (ready to operate): - Type-C: 5V@**19mA** - BAT: 3.8V@**22mA**	Circuit operating Voltage (ready to operate): - Type-C: 5V@**38.3mA** - BAT: 3.8V@**43.2mA** (with expansion board)

		-	Webcam Web application: - Type-C: - - Average power consumption: 5V/**138mA** - - Photo moment: 5V/**341mA** - Battery: - - Average power consumption: 3.8V/**154mA** - - Photo moment: 3.8V/**304mA**
		-	Microphone recording & SD card writing: - Type-C: - - Average power consumption: 5V/**46.5mA** - - Peak power consumption: 5V/**89.6mA** - Battery: - - Average power consumption: 3.8V/**54.4mA** - - Peak power consumption: 3.8V/**108mA**
		Charging battery current: **100mA**	Charging battery current: **100mA**
	Low Power Consumption Model	Modem-sleep Model: **3.8V/25 mA** Light-sleep Model: **3.8V/2 mA** Deep Sleep Model: **3.8V/14 µA**	Without any peripherals: - Modem-sleep Model: **3.8V/25.5 mA** - Light-sleep Model: **3.8V/2.4 mA** - Deep Sleep Model: **3.8V/63.768 µA** Connect the camera: - Modem-sleep Model: **3.8V/44.57 mA** - Light-sleep Model: **3.8V/5.47 mA** - Deep Sleep Model: **3.8V/3.00 mA** Connecting an SD Card: - Modem-sleep Model: **3.8V/32.8 mA**

		- Light-sleep Model: **3.8V/3.48 mA** - Deep Sleep Model: **3.8V/1.08 mA** Simultaneously connect the camera and the SD card: - Modem-sleep Model: **3.8V/55.72 mA** - Light-sleep Model: **3.8V/6.56 mA** - Deep Sleep Model: **3.8V/3.98 mA**
Wi-Fi Enabled Power Consumption	Active Model: **~ 100 mA**	Active Model: **~ 110 mA** (with expansion board)
BLE Enabled Power Consumption	Active Model: **~ 85 mA**	Active Model: **~ 102 mA** (with expansion board)
Working Temperature	-40°C ~ 65°C	

Features

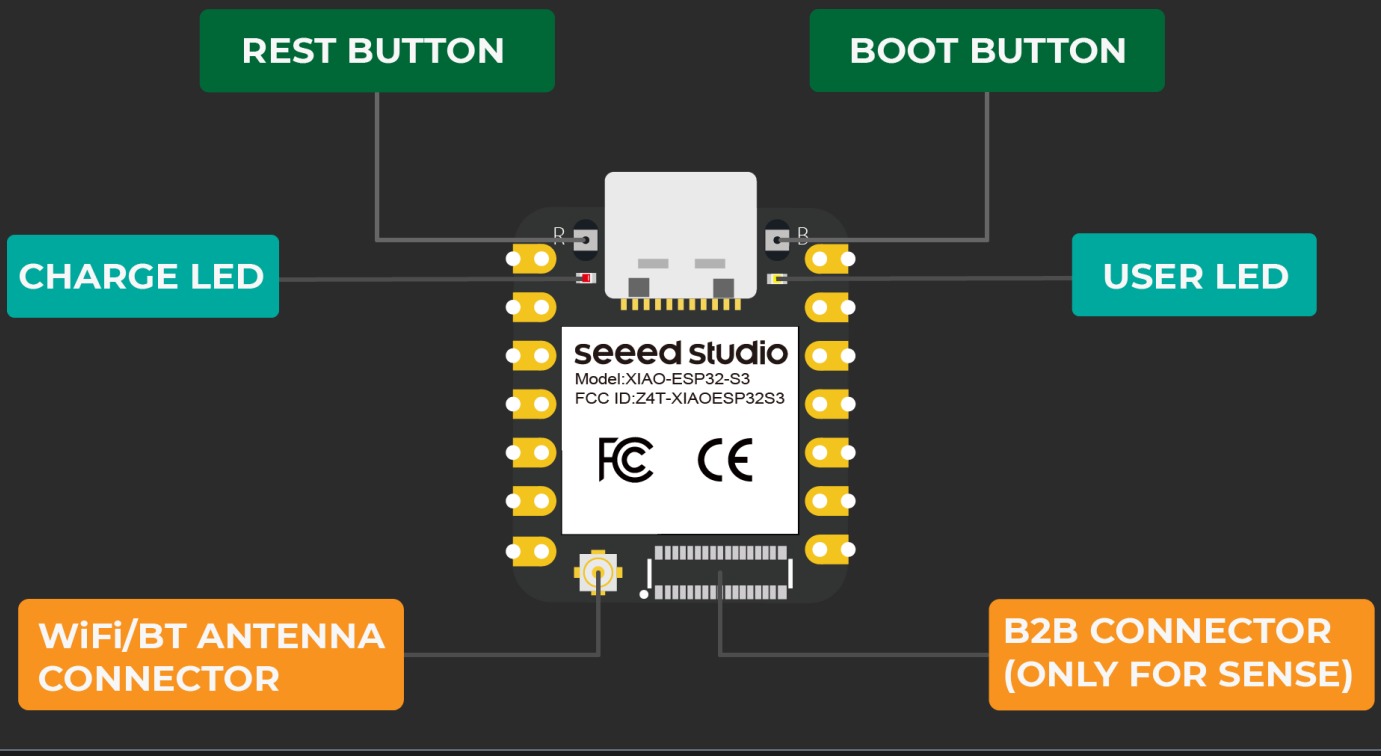
- **Powerful MCU Board**: Incorporate the ESP32S3 32-bit, dual-core, Xtensa processor chip operating up to 240 MHz, mounted multiple development ports, Arduino / MicroPython supported
- **Advanced Functionality** (for Sense): Detachable OV2640 camera sensor for 1600*1200 resolution, compatible with OV5640 camera sensor, intergating additional digital microphone
- **Elaborate Power Design**: Lithium battery charge management capability, offer 4 power consumption model which allows for deep sleep mode with power consumption as low as 14μA
- **Great Memory for more Possibilities**: Offer 8MB PSRAM and 8MB FLASH, supporting SD card slot for external 32GB FAT memory
- **Outstanding RF performance**: Support 2.4GHz Wi-Fi and BLE dual wireless communication, support 100m+ remote communication when connected with U.FL a ntenna

- **Thumb-sized Compact Design**: 21 x 17.5mm, adopting the classic form factor of XIAO, suitable for space limited projects like wearable devices

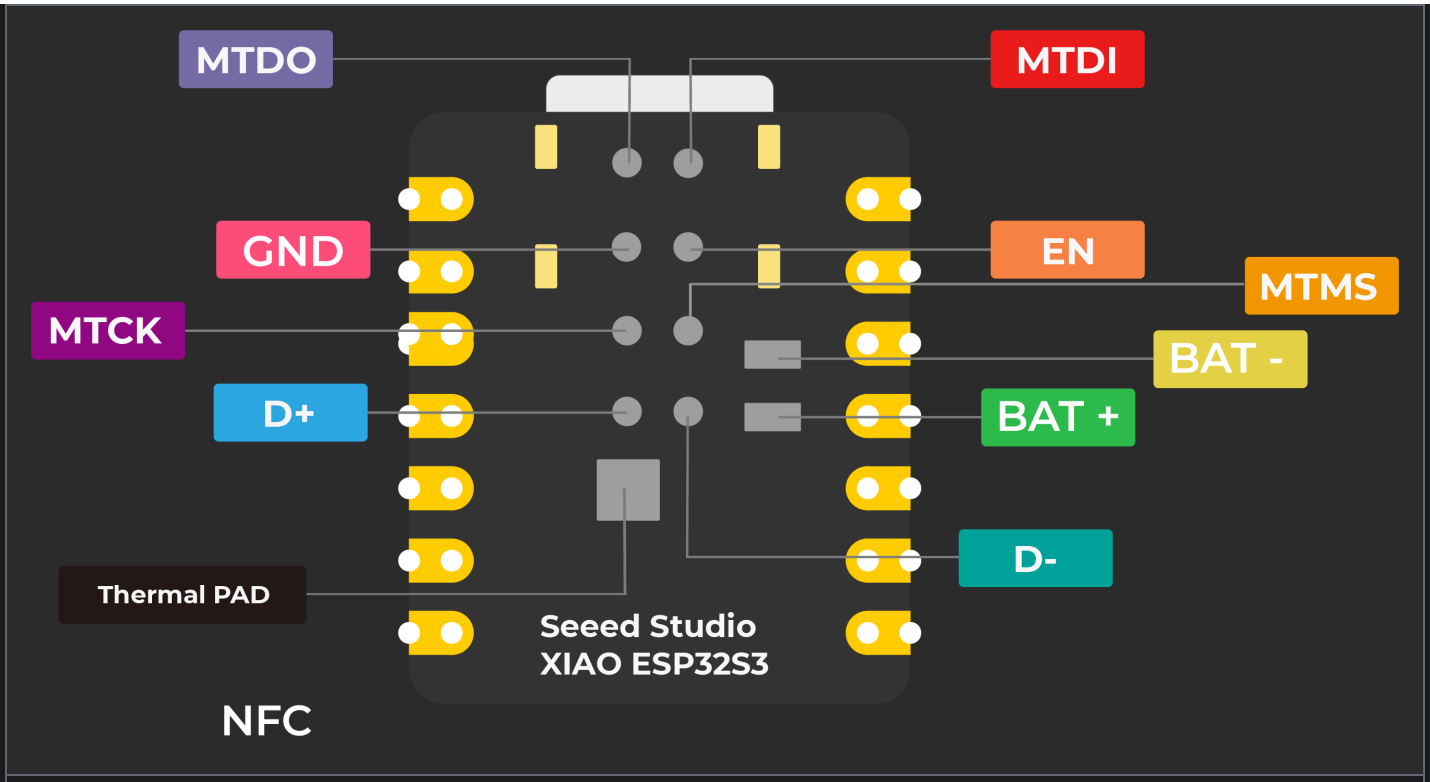
Hardware Overview

Before everything starts, it is quite essential to have some basic parameters of the product. The following table provides information about the characteristics of Seeed Studio XIAO ESP32S3.

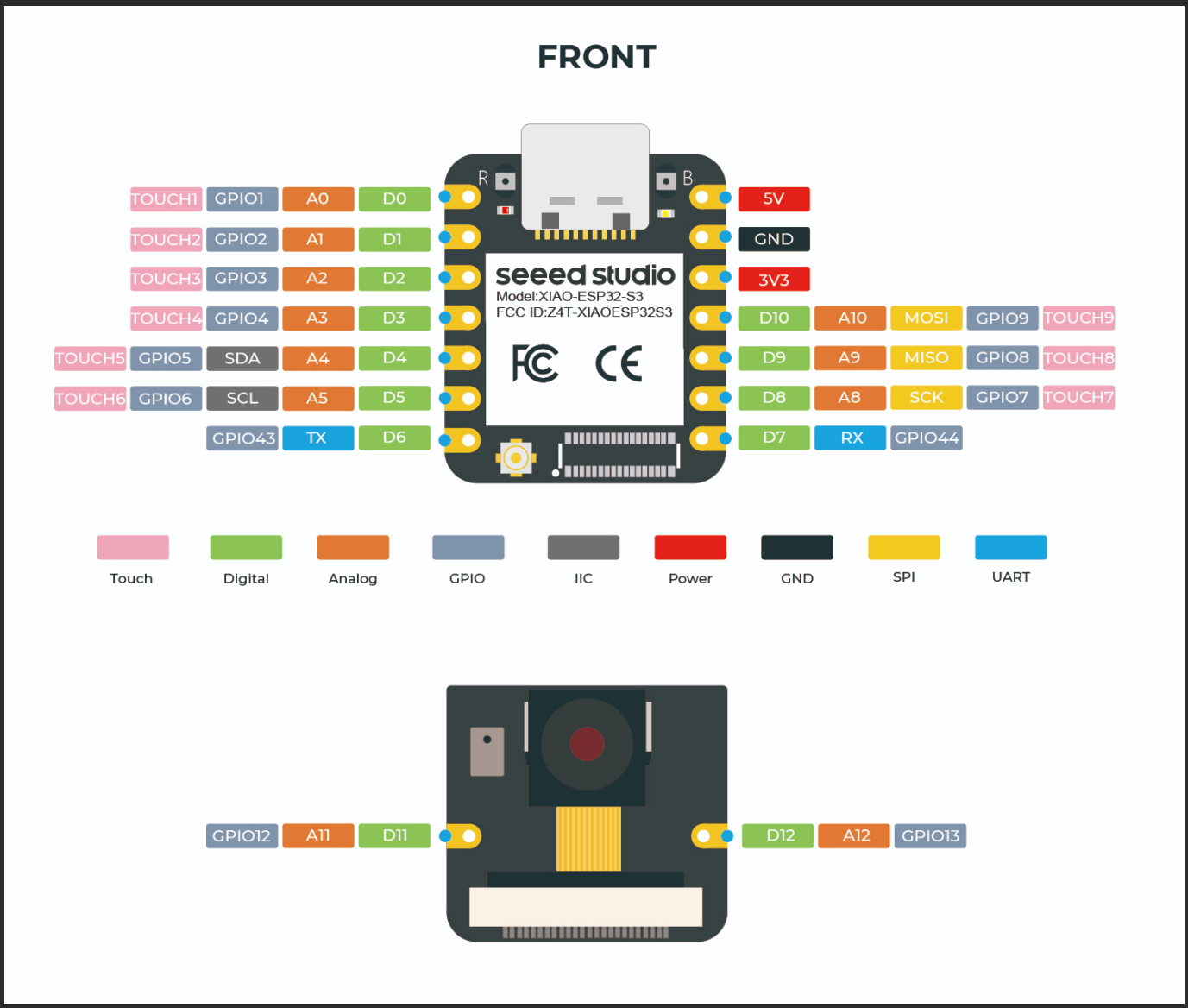
XIAO ESP32S3/XIAO ESP32S3 Sense front indication diagram



XIAO ESP32S3/XIAO ESP32S3 Sense back indication diagram



XIAO ESP32S3/XIAO ESP32S3 Sense Pin List



- 5V - This is 5v out from the USB port. You can also use this as a voltage input but you must have some sort of diode (schottky, signal, power) between your external power source and this pin with anode to battery, cathode to 5V pin.
- 3V3 - This is the regulated output from the onboard regulator. You can draw 700mA
- GND - Power/data/signal ground

Strapping Pins

At each startup or reset, a chip requires some initial configuration parameters, such as in which boot mode to load the chip, voltage of flash memory, etc. These parameters are passed over via the strapping pins. After reset, the strapping pins operate as regular IO pins.

The parameters controlled by the given strapping pins at chip reset are as follows:

- **Chip boot mode** – GPIO0 and GPIO46
- **VDD_SPI voltage** – GPIO45
- **ROM messages printing** – GPIO46
- **JTAG signal source** – GPIO3

GPIO0, GPIO45, and GPIO46 are connected to the chip's internal weak pull-up/pull-down resistors at chip reset. These resistors determine the default bit values of the strapping pins. Also, these resistors determine the bit values if the strapping pins are connected to an external high-impedance circuit.

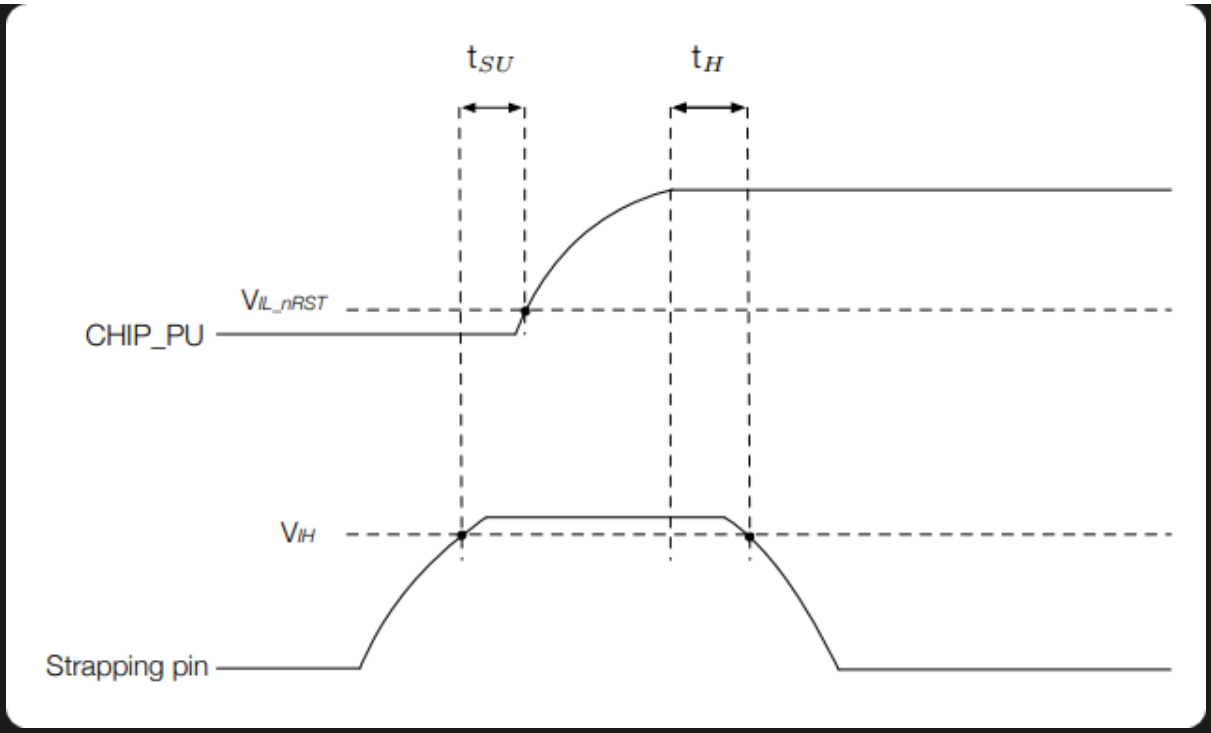
Strapping Pin	Default Configuration	Bit Value
GPIO0	Pull-up	1
GPIO3	Floating	–
GPIO45	Pull-down	0
GPIO46	Pull-down	0

To change the bit values, the strapping pins should be connected to external pull-down/pull-up resistances. If the ESP32-S3 is used as a device by a host MCU, the strapping pin voltage levels can also be controlled by the host MCU.

All strapping pins have latches. At system reset, the latches sample the bit values of their respective strapping pins and store them until the chip is powered down or shut down. The states of latches cannot be changed in any other way. It makes the strapping pin values available during the entire chip operation, and the pins are freed up to be used as regular IO pins after reset.

Regarding the timing requirements for the strapping pins, there are such parameters as setup time and hold time.

Parameter	Description	Min (ms)
t_{SU}	*Setup time* is the time reserved for the power rails to stabilize before the CHIP_PU pin is pulled high to activate the chip.	0
t_{H}	*Hold time* is the time reserved for the chip to read the strapping pin values after CHIP_PU is already high and before these pins start operating as regular IO pins.	3



Getting Started

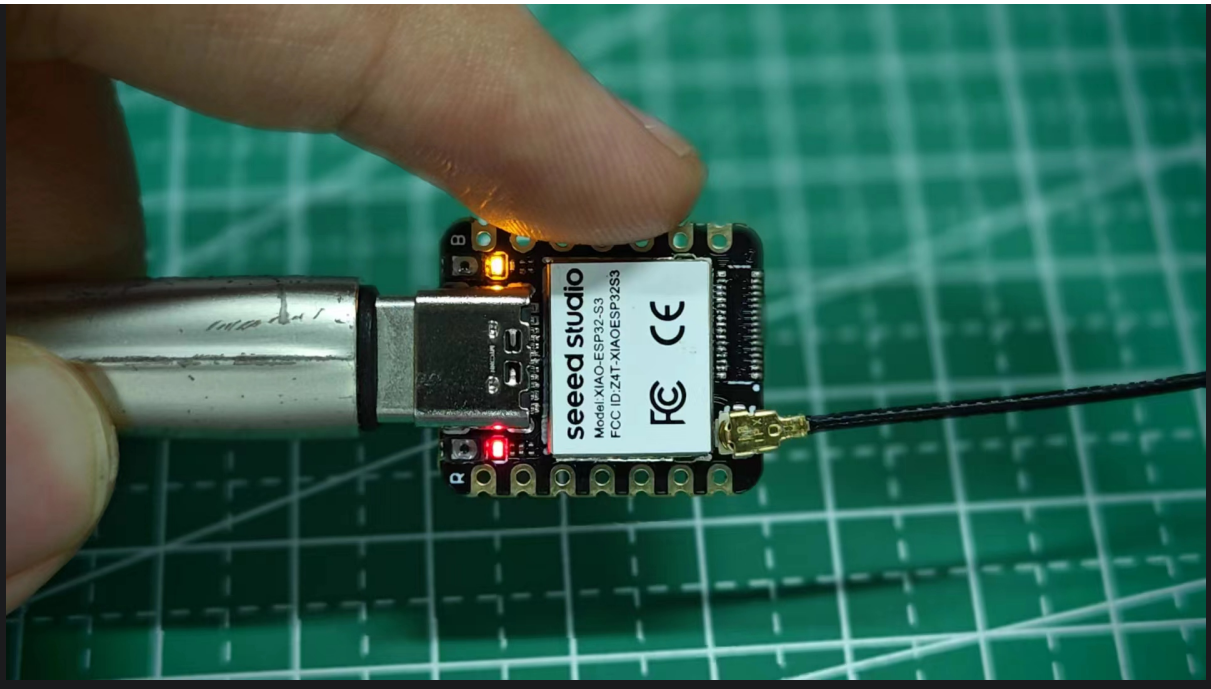
To enable you to get started with the XIAO ESP32S3 faster, please read the hardware and software preparation below to prepare the XIAO.

Factory procedure

We pre-program each new XIAO ESP32S3 and XIAO ESP32S3 Sense with a simple factory program.

1. **XIAO ESP32S3**

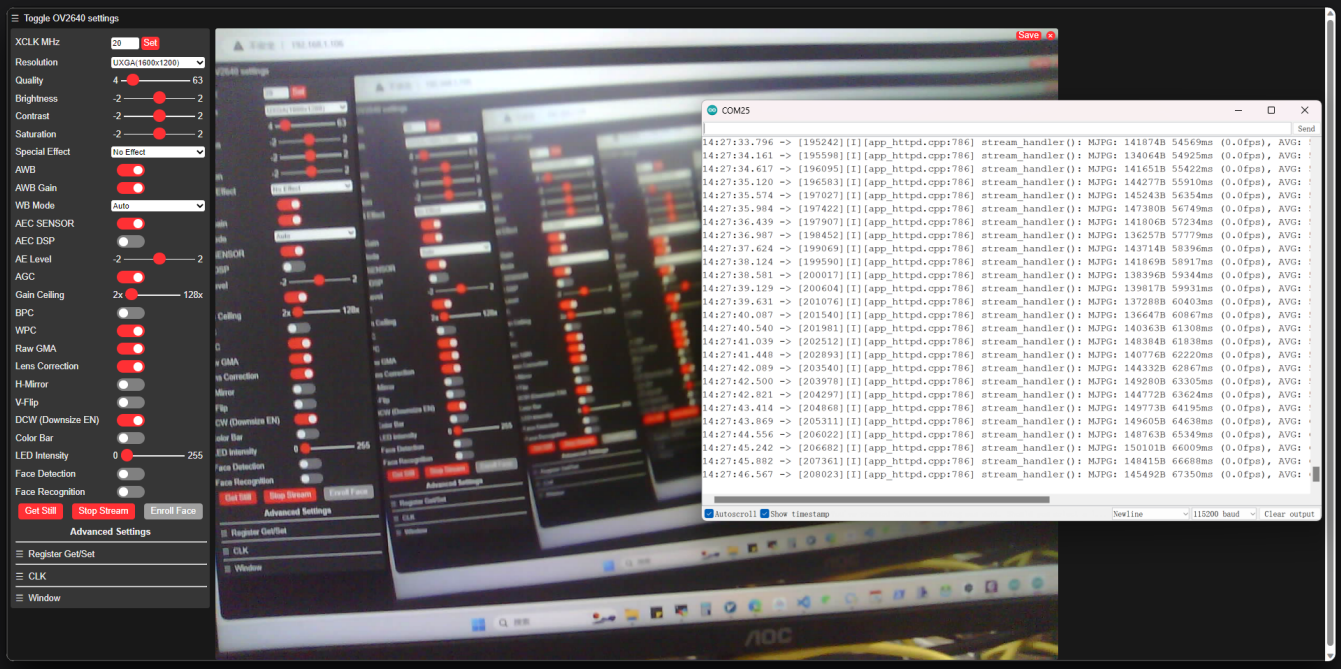
The factory program preset in the regular version is the touch pin light-up program. When you power up the XIAO, touch some of its pins and the orange user indicator will light up.



2. **XIAO ESP32S3 Sense**

The XIAO ESP32S3 Sense is shipped with the WebCam sample program pre-installed. You can use this program by giving the XIAO a good antenna installation and powering it up. For details, you can read the Wiki about this program.

- Video Streaming

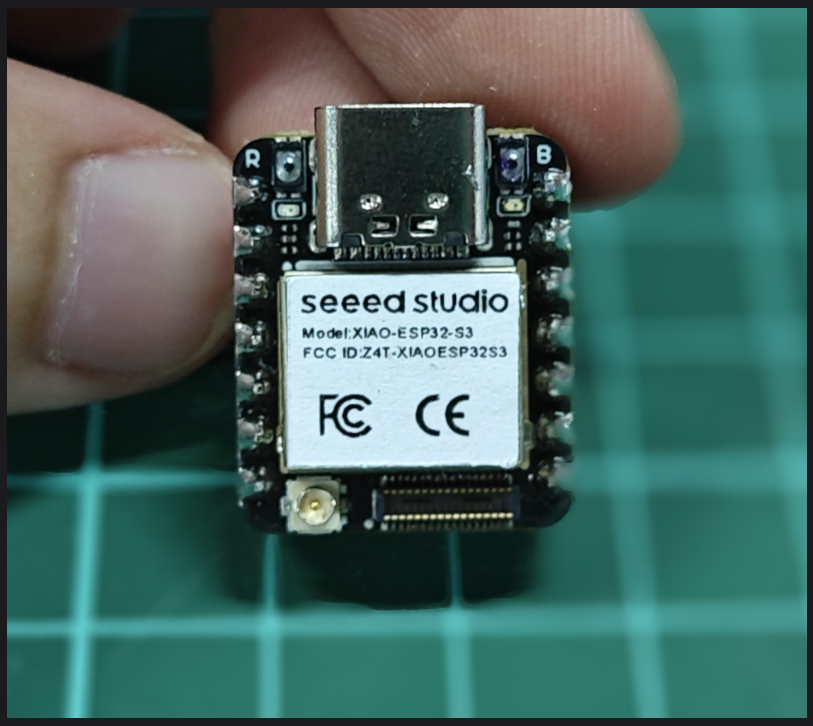


Hardware Preparation

Solder header

XIAO ESP32S3 is shipped without pin headers by default, you need to prepare your own pin headers and solder it to the corresponding pins of XIAO so that you can connect to the expansion board or sensor.

Due to the miniature size of XIAO ESP32S3, please be careful when soldering headers, do not stick different pins together, and do not stick solder to the shield or other components. Otherwise, it may cause XIAO to short circuit or not work properly, and the consequences caused by this will be borne by the user.

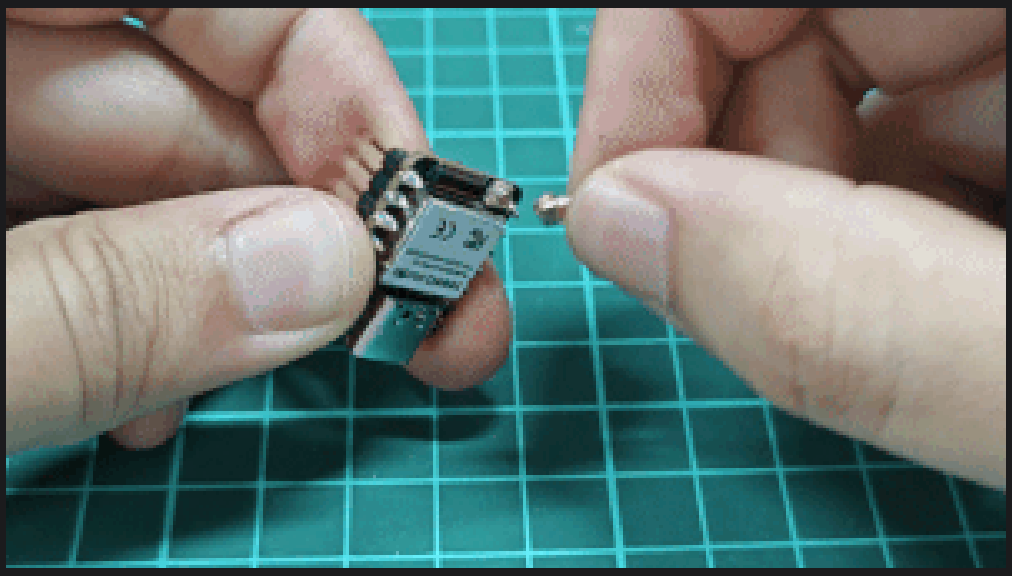


Installation of antenna

On the bottom left of the front of XIAO ESP32S3, there is a separate "WiFi/BT Antenna Connector". In order to get better WiFi/Bluetooth signal, you need to take out the antenna inside the package and install it on the connector.

There is a little trick to the installation of the antenna, if you press down hard on it directly, you will find it very difficult to press and your fingers will hurt! The correct way to install the antenna is to put one side of the antenna connector into the connector block first, then press down a little on the other side, and the antenna will be installed.

Remove the antenna is also the case, do not use brute force to pull the antenna directly, one side of the force to lift, the antenna is easy to take off.

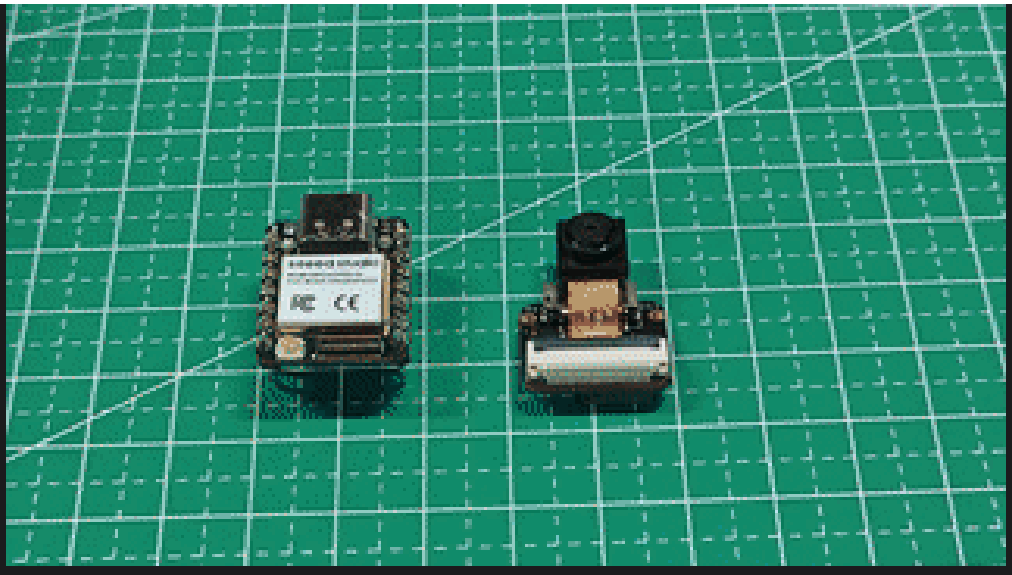


Installation of expansion boards (for Sense)

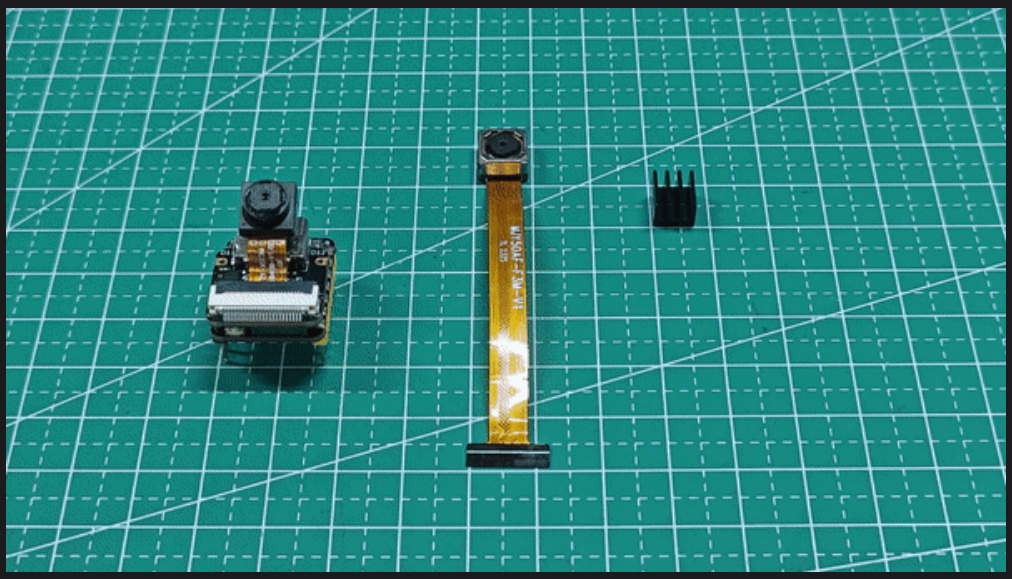
If you are shopping for the XIAO ESP32S3 Sense, then you should also include an expansion board. This expansion board has a 1600*1200 OV2640 camera sensor, Onboard SD Card Slot and digital microphone.

By installing the expansion board with XIAO ESP32S3 Sense, you can use the functions on the expansion board.

Installing the expansion board is very simple, you just need to align the connector on the expansion board with the B2B connector on the XIAO ESP32S3, press it hard and hear a "click", the installation is complete.

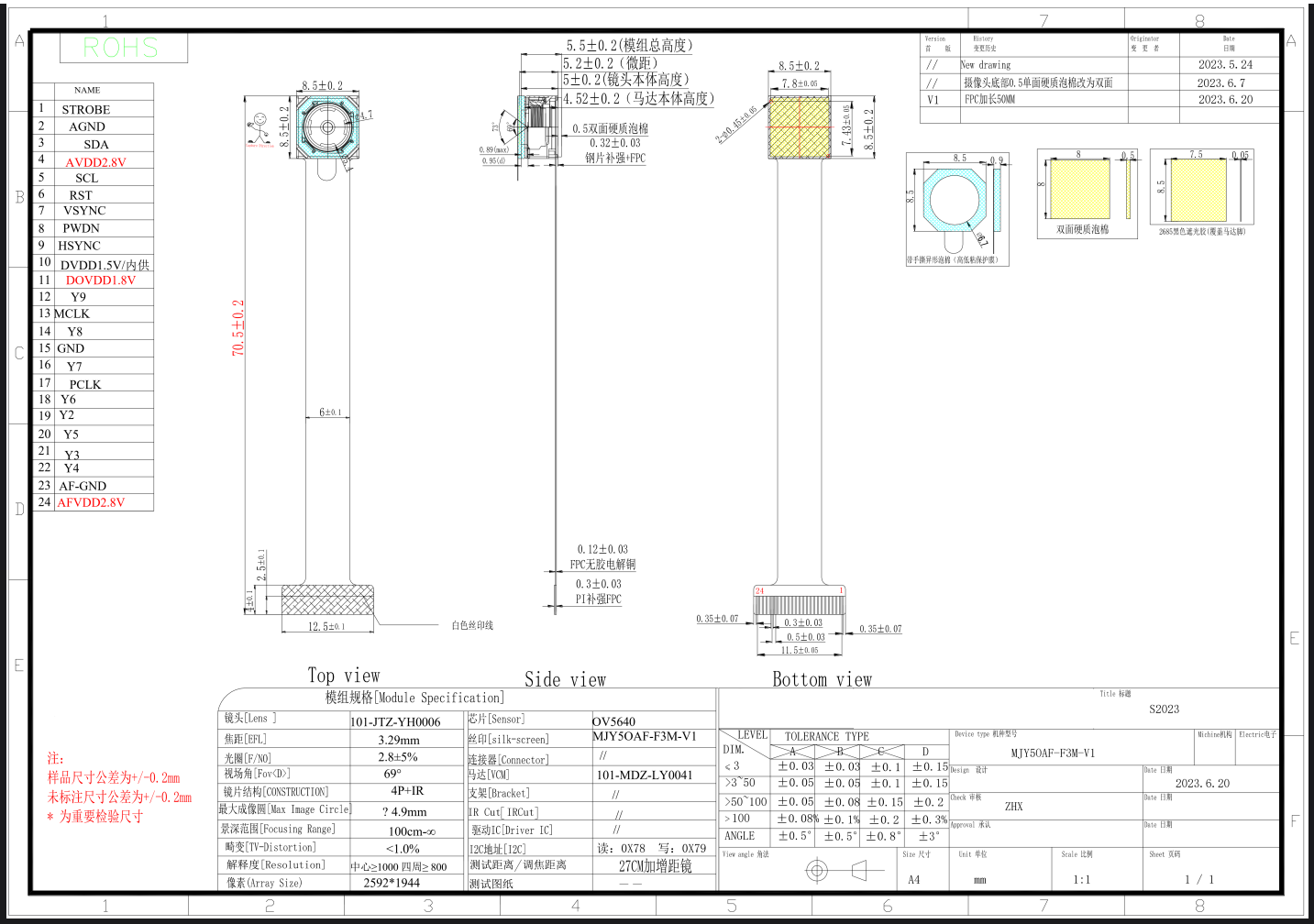


We now have a new fully XIAO ESP32S3 Sense-compatible powerful camera, the OV5640, on our shelves, and if you purchase it, you can replace the camera to use it.



Get One Now

If you need to know the detailed parameter information of ov5640, you can refer to the following chart.



> **TIP**
>
> All the programs about cameras in the Wiki are compatible with both OV5640 and OV2640 cameras.

Software Preparation

The recommended programming tool for the XIAO ESP32S3 is the Arduino IDE, so as part of the software preparation, you will need to complete the Arduino installation.

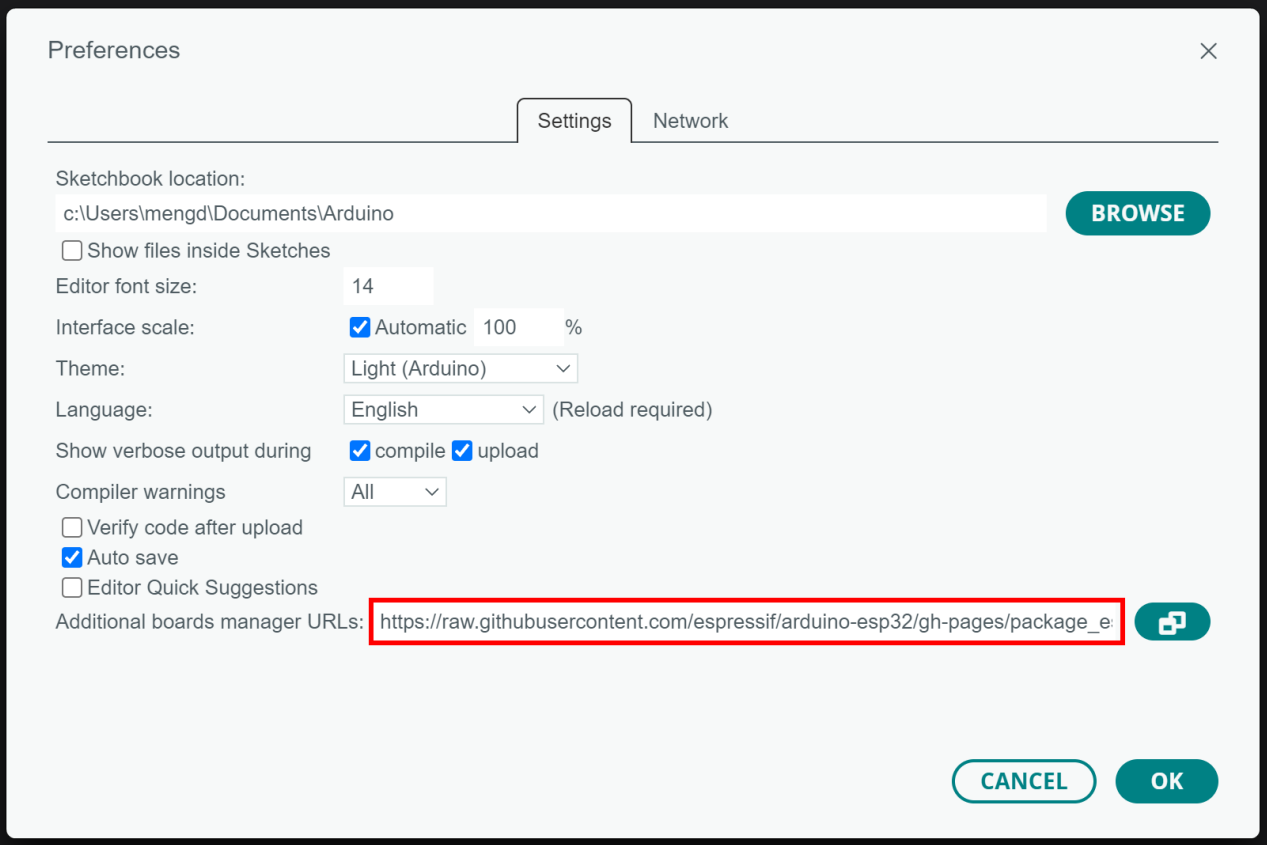
> **TIP**
>
> If this is your first time using Arduino, we highly recommend you to refer to Getting Started with Arduino.

- **Step 1.** Download and Install the stable version of Arduino IDE according to your operating system.

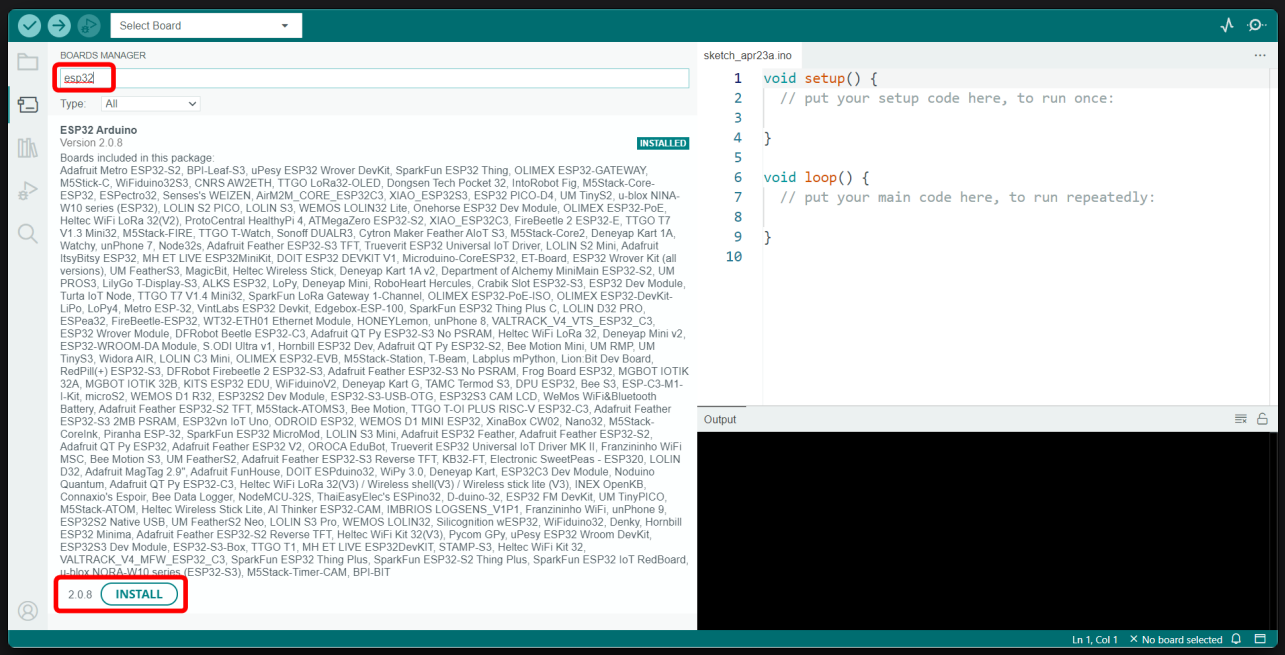
Download Arduino IDE

- **Step 2.** Launch the Arduino application.
- **Step 3.** Add ESP32 board package to your Arduino IDE.

  Navigate to **File > Preferences**, and fill **"Additional Boards Manager URLs"** with the url below: *https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json*



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

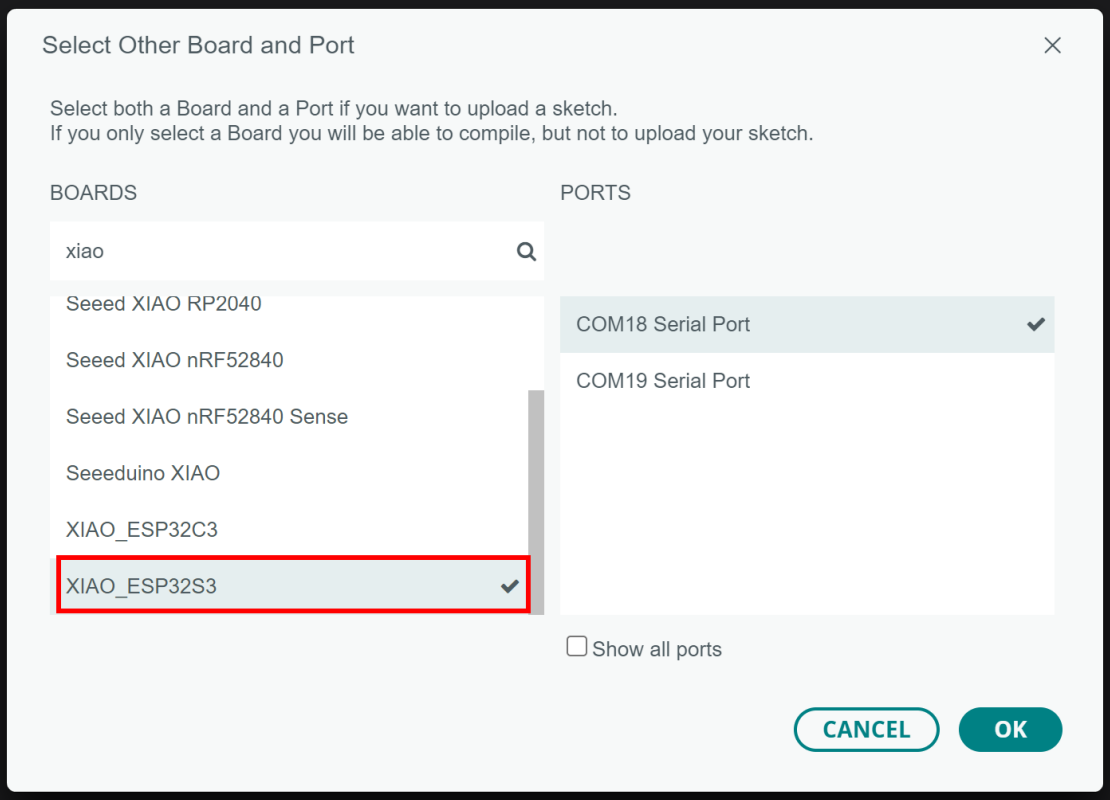


> **CAUTION**
>
> The on-board package for XIAO ESP32S3 requires at least version **2.0.8** to be available.

- **Step 4.** Select your board and port.

On top of the Arduino IDE, you can select the port directly. This is likely to be COM3 or higher (**COM1** and **COM2** are usually reserved for hardware serial ports).

Also, search for **xiao** in the development board on the left. select **XIAO_ESP32S3**.



With this preparation, you can start writing programs for XIAO ESP32S3 to compile and upload.

BootLoader Mode

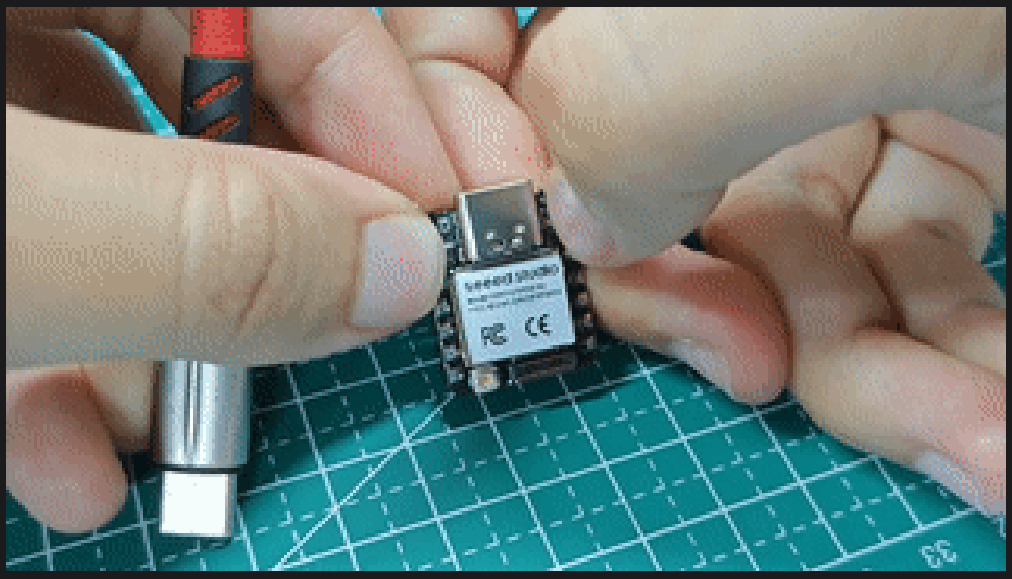
There are times when we use the wrong program to make XIAO appear to lose ports or not work properly. The specific performance is:

- Connected to computer, but no port number found for XIAO.
- The computer is connected and the port number appears, but the upload program fails.

When you encounter the above two situations, you can try to put XIAO into BootLoader mode, which can solve most of the problems of unrecognized devices and failed uploads. The specific method is:

- **Step 1**. Press and hold the BOOT button on the XIAO ESP32S3 without releasing it.
- **Step 2**. Keep the BOOT button pressed and then connect to the computer via the data cable. Release the BOOT button after connecting to the computer.

- **Step 3**. Upload the **Blink** program to check the operation of the XIAO ESP32S3.



Reset

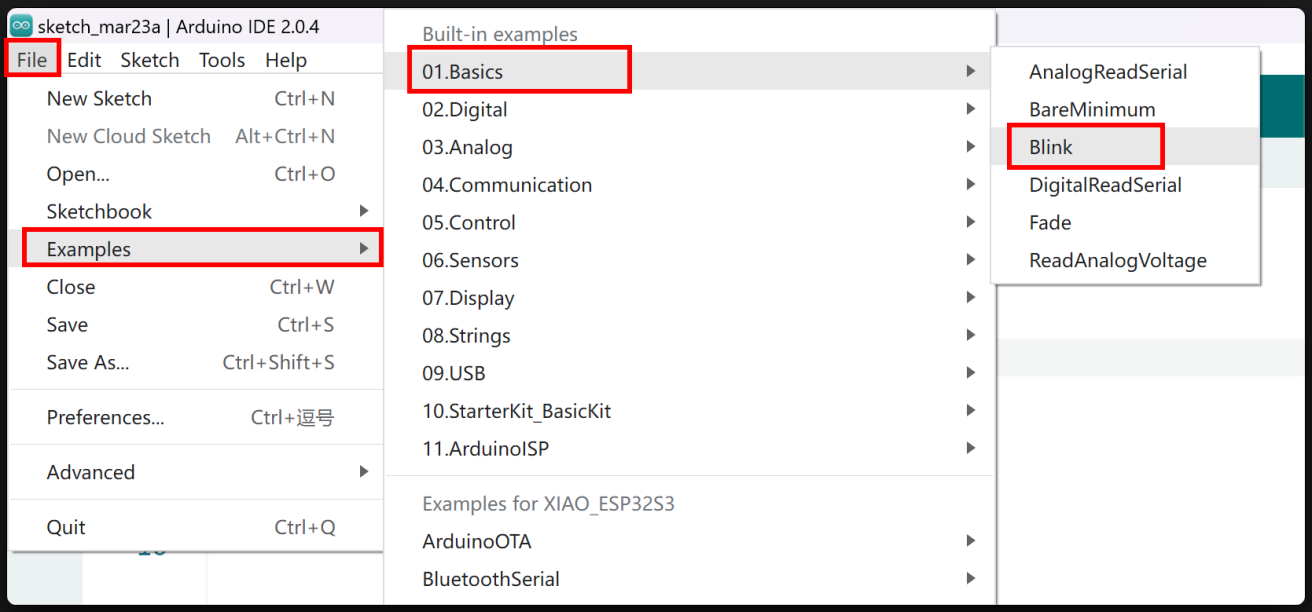
When the program runs abnormally, you can press Reset once during power-up to let XIAO re-execute the uploaded program.

When you press and hold the BOOT key while powering up and then press the Reset key once, you can also enter BootLoader mode.

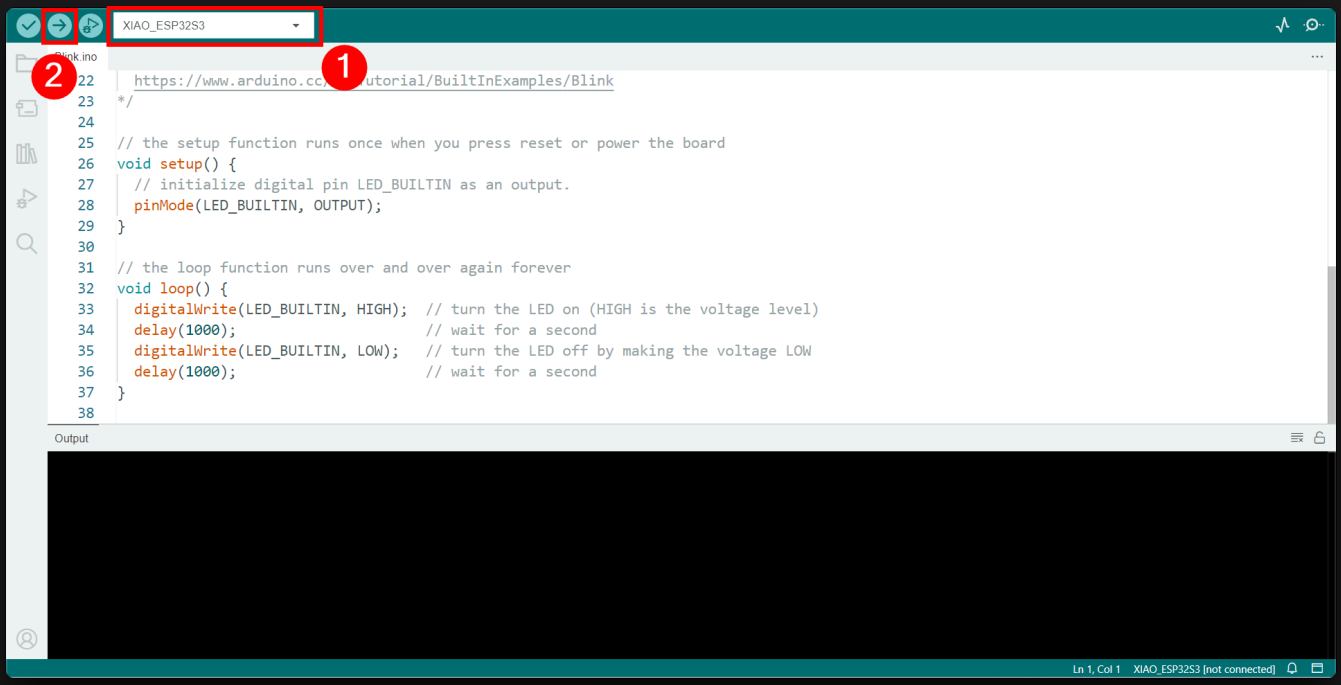
Run your first Blink program

By now, I believe you have a good understanding of the features and hardware of the XIAO ESP32S3. Next, let's take the simplest Blink program as an example and perform the first blink for your XIAO ESP32S3!

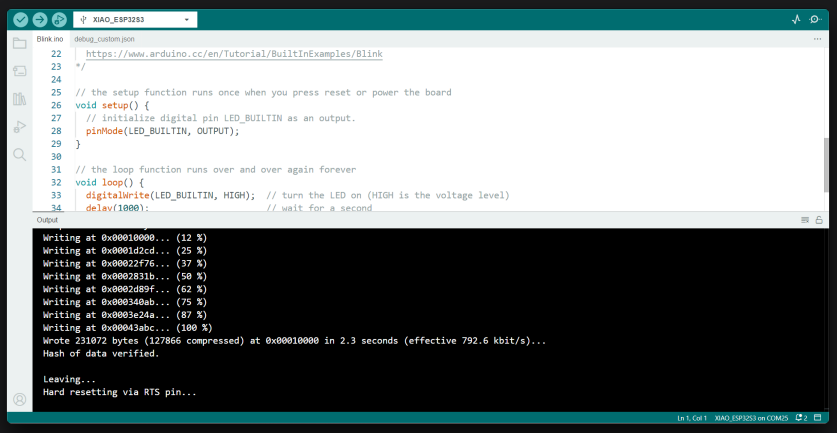
- **Step 1.** Launch the Arduino application.
- **Step 2.** Navigate to **File > Examples > 01.Basics > Blink**, open the program.

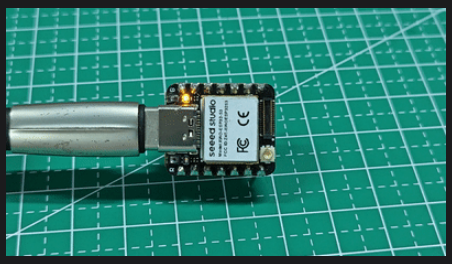


- **Step 3.** Select the board model to **XIAO ESP32S3**, and select the correct port number to upload the program.



Once the program is successfully uploaded, you will see the following output message and you can observe that the orange LED on the right side of the XIAO ESP32S3 is blinking.





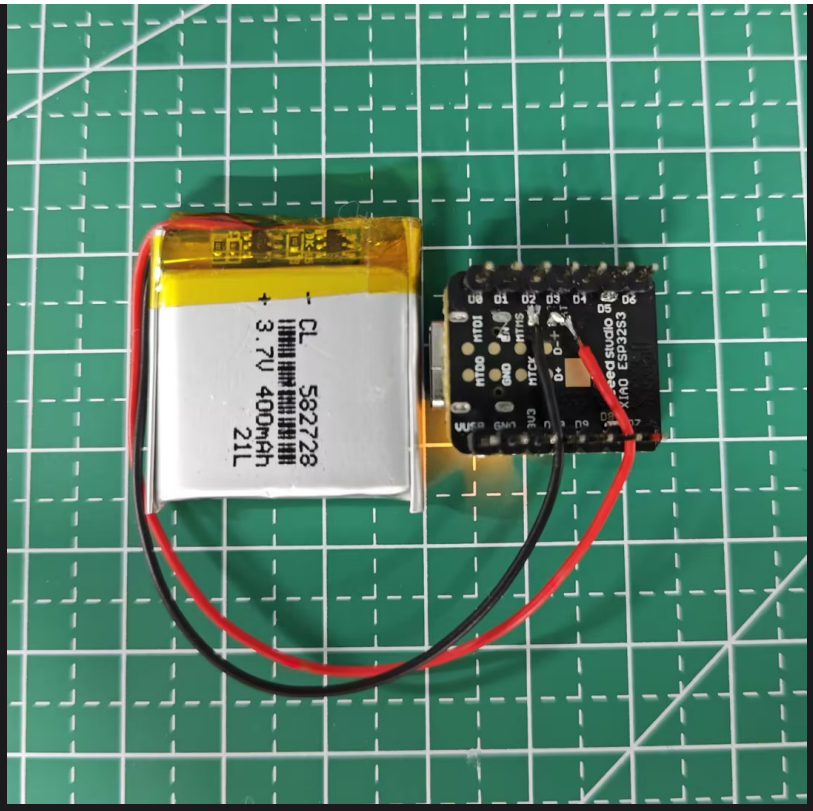
Congratulations, you've learned how to write and upload programs for the XIAO ESP32S3!

> **NOTE**
>
> The LED will only turn off when the user LED pin on the XIAO ESP32S3 is set to a high level, and it will only turn on when the pin is set to a low level.

Battery Usage

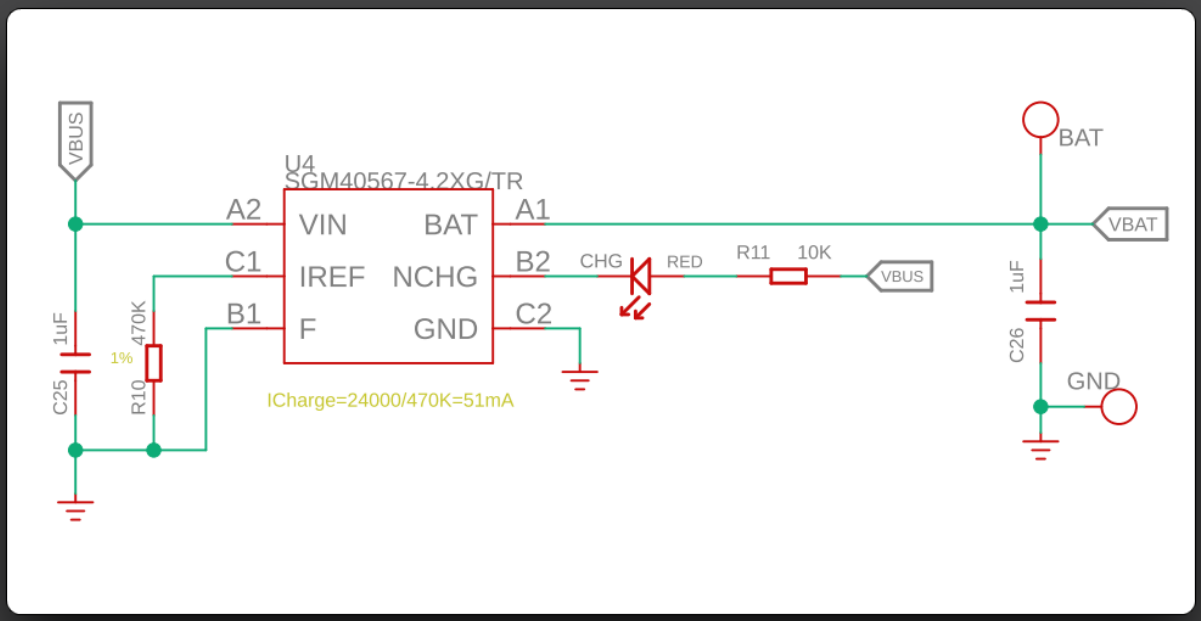
The XIAO ESP32S3 series has a built-in power management chip that allows the XIAO ESP32S3 to be powered independently by using a battery or to charge the battery through the XIAO ESP32S3's USB port.

If you want to connect the battery for XIAO, we recommend you to purchase qualified rechargeable 3.7V lithium battery. When soldering the battery, please be careful to distinguish between the positive and negative terminals. The negative terminal of the power supply should be the side closest to the USB port, and the positive terminal of the power supply is the side away from the USB port.



NOTE

Since all GPIO pins of the XIAO ESP32S3 are assigned their own functions, we do not have a GPIO configured for the battery pin. this means that we cannot get the battery voltage at the software level by reading the analog value of one of the GPIOs. If necessary, you can consider connecting the positive and negative terminals of the battery to two of the pins to measure the battery voltage.

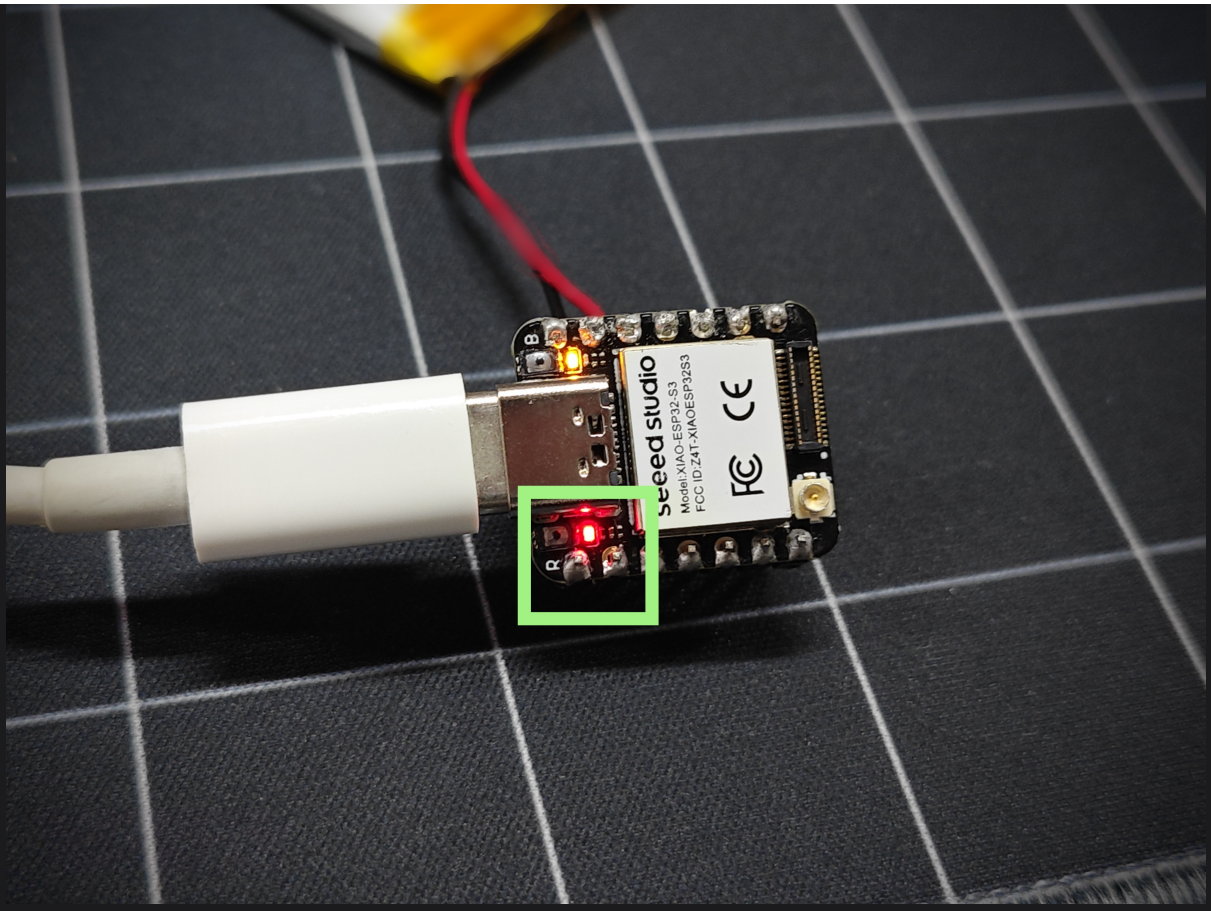


CAUTION

When you use battery power, there will be no voltage on the 5V pin.

At the same time, we designed a red indicator light for battery charging, through the indicator light display to inform the user of the current state of the battery in the charge.

1. When XIAO ESP32S3 is not connected to the battery, the red light comes on when the Type-C cable is connected and goes off after 30 seconds.
2. The red light flashes when the battery is connected and the Type-C cable is connected for charging.
3. When connecting Type-C to charge the battery fully, the red light turns off.



Deep sleep mode and wake-up

The XIAO ESP32S3 has a complete deep sleep mode and wake-up function. Here we will show two of the more common examples offered by the ESP.

Demo1: Deep Sleep with External Wake Up

This code displays how to use deep sleep with an external trigger as a wake up source and how to store data in RTC memory to use it over reboots.

```
/*
======================================
This code is under Public Domain License.

Hardware Connections
======================
Push Button to GPIO 33 pulled down with a 10K Ohm
resistor
```

```
NOTE:
======
Only RTC IO can be used as a source for external wake
source. They are pins: 0,2,4,12-15,25-27,32-39.

Author:
Pranav Cherukupalli <cherukupallip@gmail.com>
*/

#define BUTTON_PIN_BITMASK 0x200000000 // 2^33 in hex

RTC_DATA_ATTR int bootCount = 0;

/*
Method to print the reason by which ESP32
has been awaken from sleep
*/
void print_wakeup_reason(){
  esp_sleep_wakeup_cause_t wakeup_reason;

  wakeup_reason = esp_sleep_get_wakeup_cause();

  switch(wakeup_reason)
  {
    case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Wakeup caused by external
signal using RTC_IO"); break;
    case ESP_SLEEP_WAKEUP_EXT1 : Serial.println("Wakeup caused by external
signal using RTC_CNTL"); break;
    case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Wakeup caused by timer");
break;
    case ESP_SLEEP_WAKEUP_TOUCHPAD : Serial.println("Wakeup caused by
touchpad"); break;
    case ESP_SLEEP_WAKEUP_ULP : Serial.println("Wakeup caused by ULP
program"); break;
    default : Serial.printf("Wakeup was not caused by deep sleep:
%d\n",wakeup_reason); break;
  }
}

void setup(){
  Serial.begin(115200);
  delay(1000); //Take some time to open up the Serial Monitor
```

```
  //Increment boot number and print it every reboot
  ++bootCount;
  Serial.println("Boot number: " + String(bootCount));

  //Print the wakeup reason for ESP32
  print_wakeup_reason();

  /*
  First we configure the wake up source
  We set our ESP32 to wake up for an external trigger.
  There are two types for ESP32, ext0 and ext1 .
  ext0 uses RTC_IO to wakeup thus requires RTC peripherals
  to be on while ext1 uses RTC Controller so doesnt need
  peripherals to be powered on.
  Note that using internal pullups/pulldowns also requires
  RTC peripherals to be turned on.
  */
  esp_sleep_enable_ext0_wakeup(GPIO_NUM_33,1); //1 = High, 0 = Low

  //If you were to use ext1, you would use it like

//esp_sleep_enable_ext1_wakeup(BUTTON_PIN_BITMASK,ESP_EXT1_WAKEUP_ANY_HIGH);

  //Go to sleep now
  Serial.println("Going to sleep now");
  esp_deep_sleep_start();
  Serial.println("This will never be printed");
}

void loop(){
  //This is not going to be called
}
```

Demo2: Deep Sleep with Timer Wake Up

ESP32 offers a deep sleep mode for effective power saving as power is an important factor for IoT applications. In this mode CPUs, most of the RAM, and all the digital peripherals which are clocked from APB_CLK are powered off. The only parts of the chip which can still be powered on are: RTC controller, RTC peripherals ,and RTC memories.

This code displays the most basic deep sleep with a timer to wake it up and how to store data in RTC memory to use it over reboots.

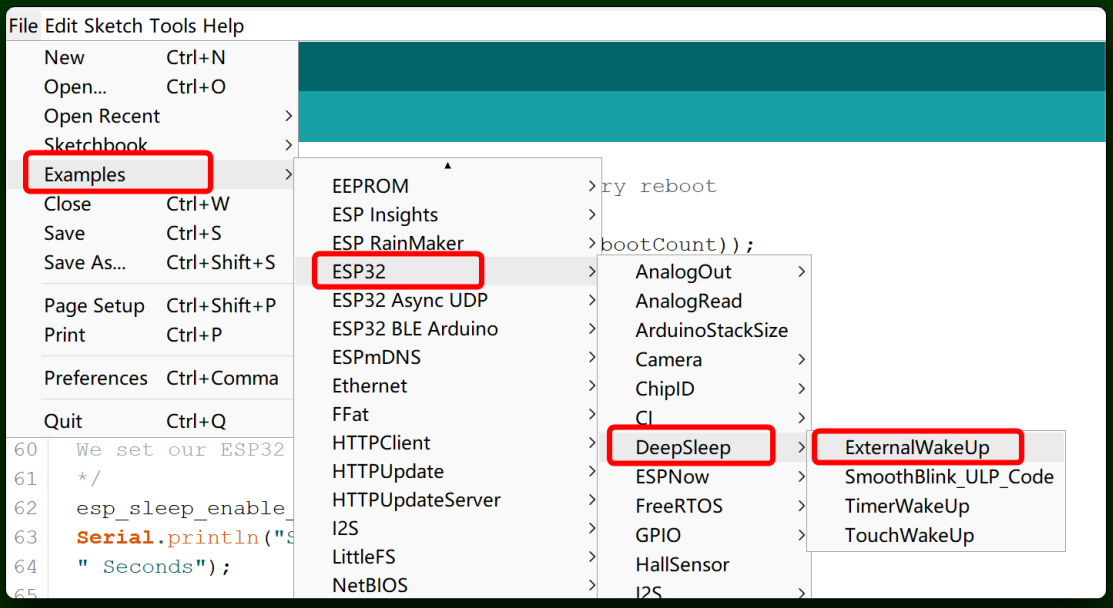
```
/*
Simple Deep Sleep with Timer Wake Up
=====================================
This code is under Public Domain License.

Author:
Pranav Cherukupalli <cherukupallip@gmail.com>
*/

#define uS_TO_S_FACTOR 1000000ULL  /* Conversion factor for micro seconds
to seconds */
#define TIME_TO_SLEEP  5        /* Time ESP32 will go to sleep (in
seconds) */

RTC_DATA_ATTR int bootCount = 0;

/*
Method to print the reason by which ESP32
has been awaken from sleep
*/
void print_wakeup_reason(){
  esp_sleep_wakeup_cause_t wakeup_reason;

  wakeup_reason = esp_sleep_get_wakeup_cause();

  switch(wakeup_reason)
  {
    case ESP_SLEEP_WAKEUP_EXT0 : Serial.println("Wakeup caused by external
signal using RTC_IO"); break;
    case ESP_SLEEP_WAKEUP_EXT1 : Serial.println("Wakeup caused by external
signal using RTC_CNTL"); break;
    case ESP_SLEEP_WAKEUP_TIMER : Serial.println("Wakeup caused by
timer"); break;
    case ESP_SLEEP_WAKEUP_TOUCHPAD : Serial.println("Wakeup caused by
touchpad"); break;
    case ESP_SLEEP_WAKEUP_ULP : Serial.println("Wakeup caused by ULP
program"); break;
    default : Serial.printf("Wakeup was not caused by deep sleep:
%d\n",wakeup_reason); break;
```

```
  }
}

void setup(){
  Serial.begin(115200);
  delay(1000); //Take some time to open up the Serial Monitor

  //Increment boot number and print it every reboot
  ++bootCount;
  Serial.println("Boot number: " + String(bootCount));

  //Print the wakeup reason for ESP32
  print_wakeup_reason();

  /*
  First we configure the wake up source
  We set our ESP32 to wake up every 5 seconds
  */
  esp_sleep_enable_timer_wakeup(TIME_TO_SLEEP * uS_TO_S_FACTOR);
  Serial.println("Setup ESP32 to sleep for every " + String(TIME_TO_SLEEP)
+
  " Seconds");

  /*
  Next we decide what all peripherals to shut down/keep on
  By default, ESP32 will automatically power down the peripherals
  not needed by the wakeup source, but if you want to be a poweruser
  this is for you. Read in detail at the API docs
  http://esp-idf.readthedocs.io/en/latest/api-
reference/system/deep_sleep.html
  Left the line commented as an example of how to configure peripherals.
  The line below turns off all RTC peripherals in deep sleep.
  */
  //esp_deep_sleep_pd_config(ESP_PD_DOMAIN_RTC_PERIPH, ESP_PD_OPTION_OFF);
  //Serial.println("Configured all RTC Peripherals to be powered down in
sleep");

  /*
  Now that we have setup a wake cause and if needed setup the
  peripherals state in deep sleep, we can now start going to
  deep sleep.
  In the case that no wake up sources were provided but deep
  sleep was started, it will sleep forever unless hardware
```

```
  reset occurs.
  */
  Serial.println("Going to sleep now");
  Serial.flush();
  esp_deep_sleep_start();
  Serial.println("This will never be printed");
}

void loop(){
  //This is not going to be called
}
```

TIP

If you want to learn to use more of the deep sleep mode and wake-up functions, you can find more sample programs officially written for the chip by ESP in the Arduino IDE.



UF2 BootLoader

We understand that some users are looking to flash UF2 files directly to XIAO, which will enable the process of batch flashing programs. Here we will describe this method.

Method I

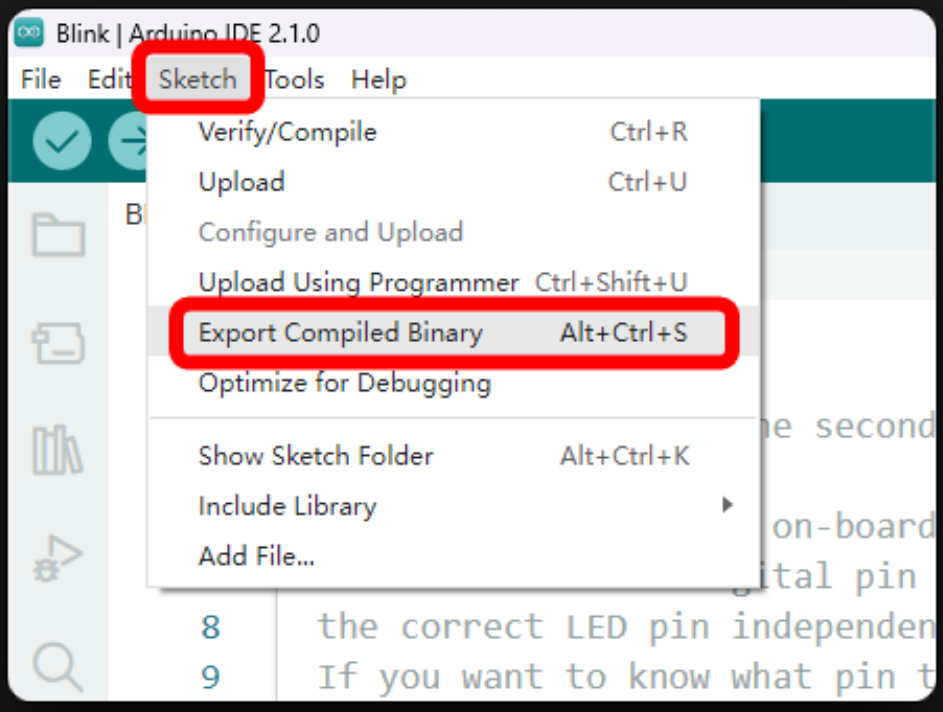
> **NOTE**
>
> Currently this method can only be used on Windows systems.

Step 1. Download the required script zip. And extract it to your local machine.

https://files.seeedstudio.com/wiki/SeeedStudio-XIAO-ESP32S3/res/xiaos3-bin2uf2.zip

Step 2. Converting BIN files into UF2 files.

Once you have compiled and saved an Arduino program, you can export the binary file BIN file directly. This file will then be generated in your Arduino project folder.

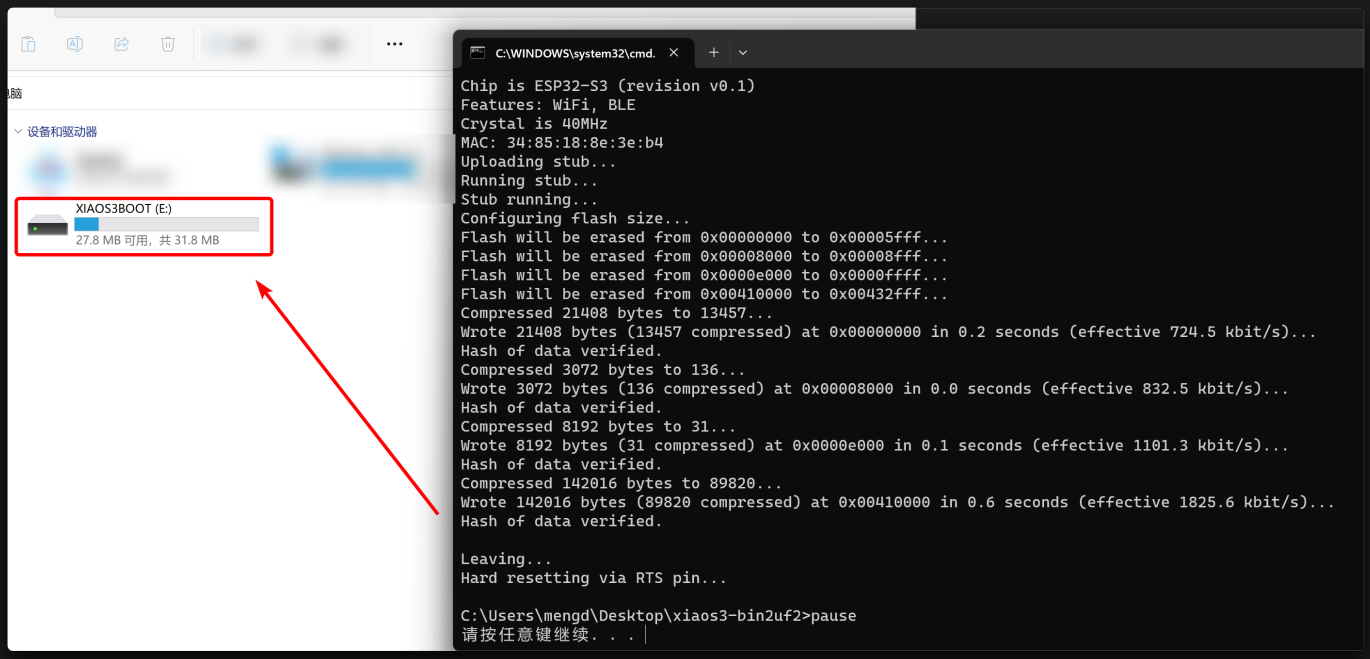


At this point, all you need to do is copy this BIN file to the **xiaos3-bin2uf2** directory that you just extracted in the first step, and then execute the **convert_uf2.bat** script to generate a UF2 file

directly.

Step 3. Put the XIAO into UF2 BootLoader mode.

Then please connect the XIAO to the computer and then run the **boot_uf2.bat** script again, the XIAO will appear in the computer as a USB stick, which means it has successfully entered the UF2 BootLoader mode.



Step 4. Copy the UF2 file to XIAO ESP32S3.

Next you can access the XIAO ESP32S3's USB stick and copy the converted UF2 to the USB stick. Once the copying is complete, the XIAO USB stick will automatically disappear and the program will start to execute.

TIP

1. Please ensure that your program is compiled and executed without problems, otherwise the UF2 program may not run as expected.
2. The sample UF2 file for Blink is provided in the **xiaos3-bin2uf2** folder. When this program is uploaded, the orange LED on the XIAO ESP32S3 will flash and you can use this UF2 file as a test.

Step 5. Enter the UF2 BootLoader again.

Once you have performed the above steps and you still want the XIAO ESP32S3 to access the UF2 BootLoader to upload other UF2 files, you need to quickly press the **Reset** button first and then the **Boot** button afterwards. And there is no need to execute the boot_uf2.bat script again.

> **NOTE**
>
> Press Reset then Boot, and be quick!

Method II

The project is composed of customizing the 2nd stage bootloader from IDF and UF2 factory application as 3rd stage bootloader. Note: since IDF is actively developed and change very often, it is included as submodule at lib/esp-idf, please run export script there to have your environment setup correctly.

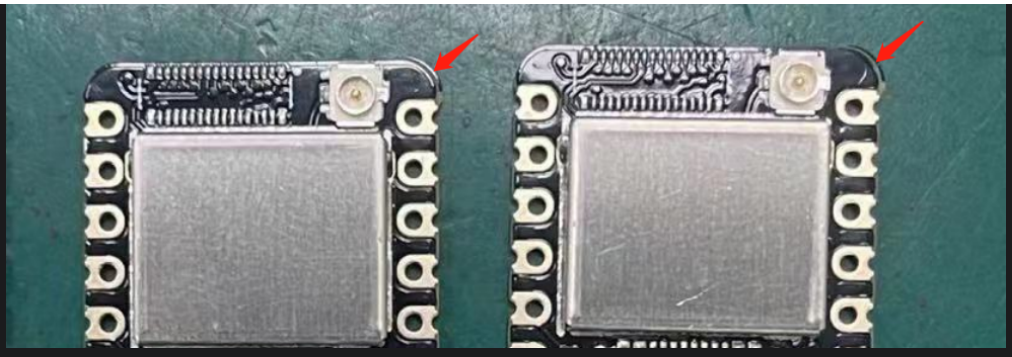
Learn More

Troubleshooting

Q1: What should I do if the upload program fails/the program runs abnormally/the device port is not found?

If you encounter the above problem, it is recommended that you first try pressing the reset button on the XIAO ESP32S3 to try to get the program running again. If the problem persists, please recheck your program and read the methods provided in **BootLoader Mode** to restore the device.

Q2: Why does my XIAO have the problem of not being flush at the rounded corners? Is this a quality problem?



First of all, it should be noted that this is not a quality issue and will not affect the normal function of XIAO.

XIAO ESP32S3 is the most complex one in all XIAO because of its high integration, and the PCB needs to be put together in factory production. Due to the high level of integration, the splicing board connection can only be placed at the four rounded corners, which will lead to the problem of uneven rounded corners on the picture. We will try to improve the process to ensure that this problem will be solved in the subsequent production.

Resources

[PDF] **ESP32-S3 Datasheet**

For Seeed Studio XIAO ESP32S3

- **[PDF]** Seeed Studio XIAO ESP32S3 Schematic
- **[ZIP]** Seeed Studio XIAO ESP32S3 Eagle Libraries
- **[DXF]** Seeed Studio XIAO ESP32S3 Dimension in DXF
- **[LBR]** Seeed Studio XIAO ESP32S3 Eagle footprint
- **[ZIP]** Seeed Studio XIAO ESP32S3 Factory firmware
- **[XLSX]** Seeed Studio XIAO ESP32S3 pinout sheet
- **[STEP]** Seeed Studio XIAO ESP32S3 3D Model

For Seeed Studio XIAO ESP32S3 Sense

- **[PDF]** Seeed Studio XIAO ESP32S3 Sense Schematic
- **[ZIP]** Seeed Studio XIAO ESP32S3 Sense KiCAD Libraries
- **[ZIP]** Seeed Studio XIAO ESP32S3 Sense Eagle Libraries
- **[DXF]** Seeed Studio XIAO ESP32S3 Sense Dimension in DXF (top)
- **[DXF]** Seeed Studio XIAO ESP32S3 Sense Dimension in DXF (bottom)
- **[ZIP]** Seeed Studio XIAO ESP32S3 Sense Factory firmware
- **[XLSX]** Seeed Studio XIAO ESP32S3 Sense pinout sheet
- **[STEP]** Seeed Studio XIAO ESP32S3 Sense 3D Model

Other

- **[STP]** XIAO ESP32S3 Sense housing design (top)
- **[STP]** XIAO ESP32S3 Sense housing design (bottom)

The remaining open source material is being compiled, so stay tuned!

Tech Support & Product Discussion

Thank you for choosing our products! We are here to provide you with different support to ensure that your experience with our products is as smooth as possible. We offer several communication channels to cater to different preferences and needs.

Edit this page

Last updated on ***Mar 23, 2023*** *by* ***Citric***

4 reactions

1 1 1 1

31 comments · 61+ replies *– powered by giscus*

Oldest Newest