# ESP32-S3-Touch-LCD-4.3

## Overview

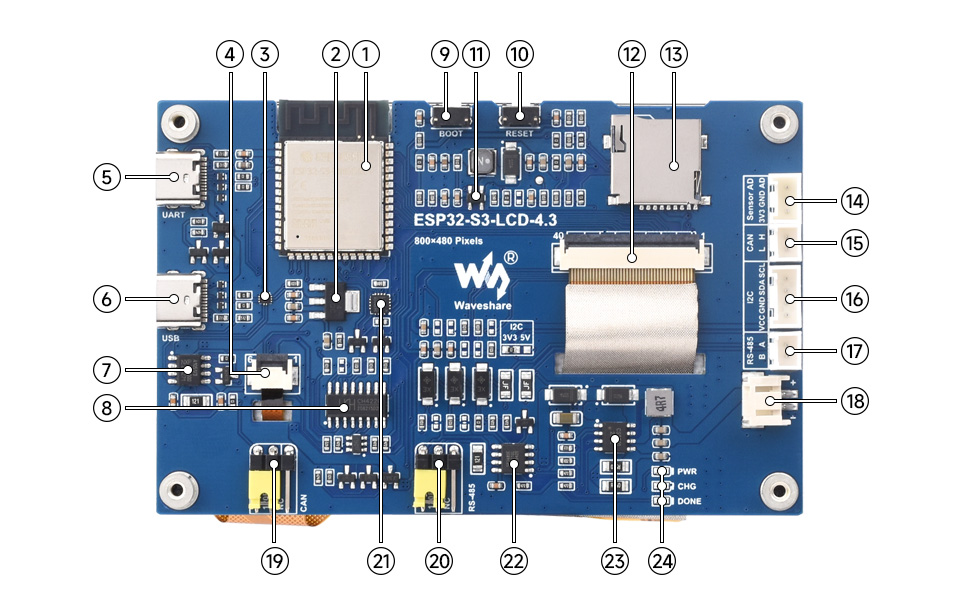
### Introduction

This is a low-cost, high-performance MCU development board designed by Waveshare. It supports 2.4GHz WiFi and BLE 5, integrates high-capacity Flash and PSRAM, and has a 4.3-inch wide capacitive touch LCD screen on board to smoothly run GUI interface programs such as LVGL. It combines a variety of peripheral interfaces (e.g., CAN, I2C, RS485, etc.) to quickly develop applications such as HMIs for ESP32-S3. With a wide range of functions and interfaces, it can meet power consumption requirements in Internet of Things (IoT), mobile devices, smart home and other applications.

### Features

* Equipped with high-performance Xtensa 32-bit LX7 dual-core processor, up to 240MHz main frequency
* Supports 2.4GHz Wi-Fi (802.11 b/g/n) and Bluetooth 5 (BLE), with onboard antenna
* Built-in 512KB SRAM and 384KB ROM, with onboard 16MB Flash and 8MB PSRAM
* Onboard 4.3inch LCD display, 800×480 resolution, 65K color
* Supports capacitive touch control via I2C interface (optional), with 5-point touch, and supports interrupts
* Onboard CAN, RS485, I2C interface and TF card slot, integrates full-speed USB
* Supports flexible clock, module power supply independent setting and other controls to realize low power consumption in different scenarios

### Onboard Resources

[](https://www.waveshare.com/wiki/File:ESP32-S3-Touch-LCD-4.3-details-intro.jpg)

**1. ESP32-S3-WROOM-1-N16R8**   
WiFi Bluetooth SoC module with 240MHz operating frequency   
Package 8MB PSRAM and 16MB Flash

**2. SGM2212-3.3**   
800mA low-noise LDO

**3. FSUSB42UMX**   
For USB pin multiplexing

**4. Touch panel connector**   
For non-touch, it is not connected

**5. USB TO UART Type-C port**

**6. USB Type-C interface**

**7. TJA1051T/3/1J**   
CAN interface chip

**8. CH422G**   
IO expansion chip

**9. BOOT button**   
Press and hold to power on for program flashing

**10. RESET button**   
Press to restart controller

**11. MP3302DJ-LF-Z**   
Screen backlight boost chip **12. 4.3inch display panel connector**

**13. TF card slot**

**14. Sensor screw terminal**

**15. CAN interface**

**16. I2C terminal**

**17. RS485 terminal**

**18. 3.7V single lithium battery PH2.0 interface**

**19. CAN terminal resistor selection interface**

**20. RS485 terminal resistor selection interface**

**21. CH343P**   
USB to UART chip

**22. SP3485**   
RS485 transceiver chip

**23. CS8501**   
Lithium battery charge manager chip

**24. Status indicator**   
PWR: Power indicator   
CHG: Lithium battery charging indicator   
DONE: Lithium battery charging completed indicator

## Pinouts

* **LCD interface**: The interface to connect the LCD cable

|  |  |  |
| --- | --- | --- |
| **ESP32-S3** | **LCD** | **Description** |
| GPIO0 | G3 | Green data 3rd place |
| GPIO1 | R3 | Red data 3rd place |
| GPIO2 | R4 | Red data 4th place |
| GPIO3 | VSYNC | Vertical synchronization signal |
| GPIO5 | DE | Data enable signal |
| GPIO7 | PCLK | Clock signal |
| GPIO10 | B7 | Blue 7th place |
| GPIO14 | B3 | Blue 3rd place |
| GPIO17 | B6 | Blue 6th place |
| GPIO18 | B5 | Blue 5th place |
| GPIO21 | G7 | Green 7th place |
| GPIO38 | B4 | Blue 4th place |
| GPIO39 | G2 | Green 2nd place |
| GPIO40 | R7 | Red 7th place |
| GPIO41 | R6 | Red 6th place |
| GPIO42 | R5 | Red 5th place |
| GPIO45 | G4 | Green 4th place |
| GPIO46 | HSYNC | Horizontal synchronization signal |
| GPIO47 | G6 | Green 6th place |
| GPIO48 | G5 | Green 5th place |
| CH422G | LCD | - |
| EXIO2 | DISP | Backlight enable pin |

* **Touch screen interface**: The interface used to connect the touch cable

|  |  |  |
| --- | --- | --- |
| ESP32-S3 | Touch | Description |
| GPIO4 | TP\_IRQ | Touch interrupt pin |
| GPIO8 | TP\_SDA | Touch data pin |
| GPIO9 | TP\_SCL | Touch clock pin |
| CH422G | Touch | - |
| EXIO1 | TP\_RST | Touch reset pin |

* **USB interface**: Used for power supply and flashing

|  |  |  |
| --- | --- | --- |
| ESP32-S3 | USB | Description |
| GPIO19 | USB\_DN | Data cable D- |
| GPIO20 | USB\_DP | Data cable D+ |
| CH422G | USB | - |
| EXIO5 | USB\_SEL | Pull down to USB mode, otherwise CAN mode |

* **TF card interface**: The interface used to connect the TF card. The pin connections are shown in the table below.

|  |  |  |
| --- | --- | --- |
| ESP32-S3 | TF | Description |
| GPIO11 | MOSI | TF card input pin |
| GPIO12 | SCK | TF card clock pin |
| GPIO13 | MISO | TF card output pin |
| CH422G | TF | - |
| EXIO4 | SD\_CS | TF card enable pin, active low |

* **RS485 interface**: The development board is equipped with an RS485 interface, allowing direct connection for device communication, with automatic switching of the circuit's transmit and receive modes

|  |  |  |
| --- | --- | --- |
| ESP32-S3 | RS485 | Description |
| GPIO16 | RS485\_RXD | Data input |
| GPIO15 | RS485\_TXD | Data output |

* **CAN interface**: Implements the transmission and reception control, data analysis, acquisition and monitoring of the CAN bus network

|  |  |  |
| --- | --- | --- |
| ESP32-S3 | CAN | Description |
| GPIO15 | CANTX | Data output |
| GPIO16 | CANRX | Data input |
| CH422G | CAN | - |
| EXIO5 | CAN\_SEL | Pull up to CAN mode, otherwise USB mode |

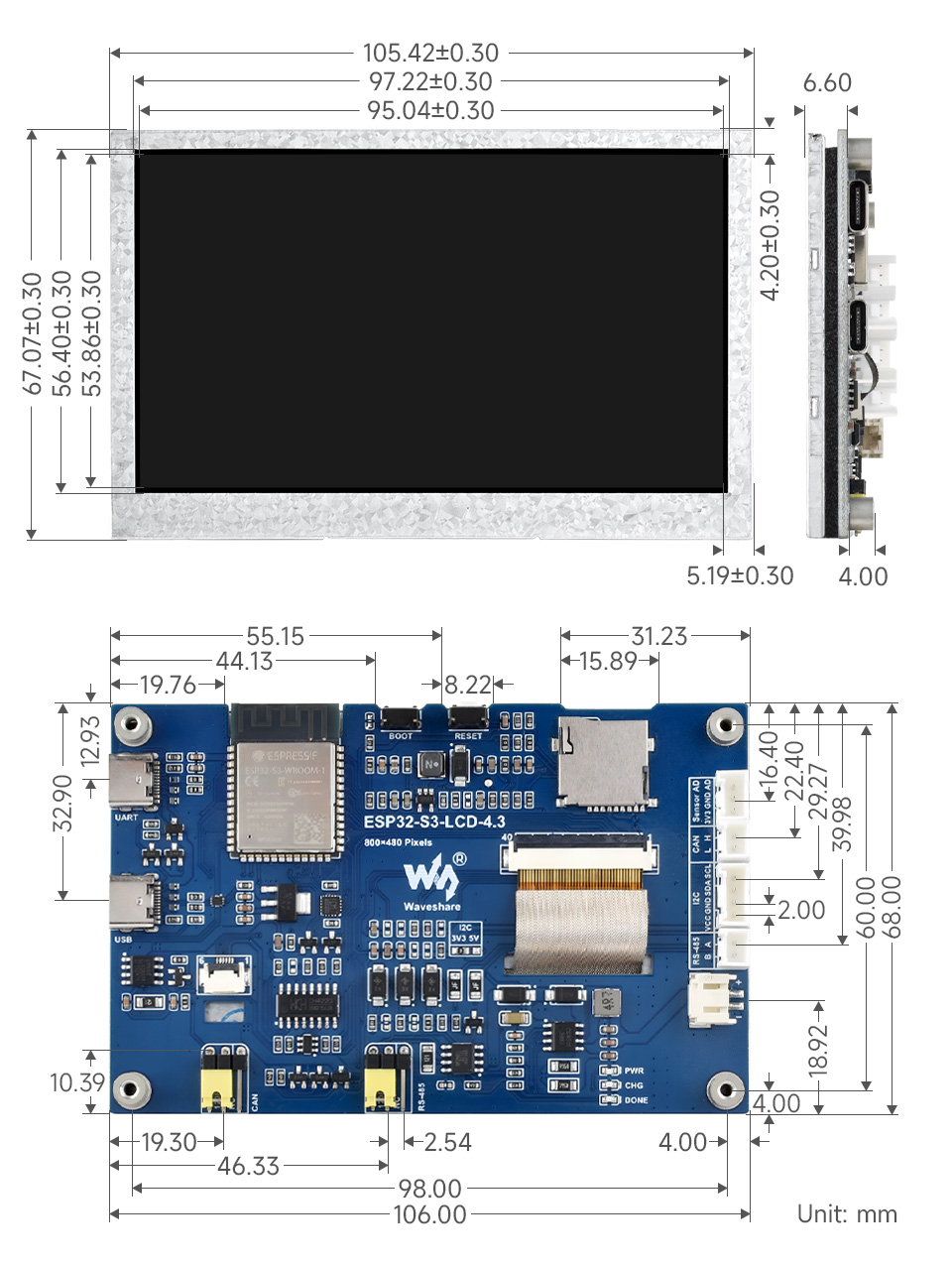
* **I2C interface**: ESP32-S3 provides multi-channel hardware I2C, currently using GPIO8 (SDA) and GPIO9 (SCL) pins for I2C bus   
  It is for mounting the IO expansion chip, touch interface, I2C external interface

|  |  |  |
| --- | --- | --- |
| ESP32-S3 | I2C | Description |
| GPIO8 | SDA | I2C data pin |
| GPIO9 | SCL | I2C clock pin |

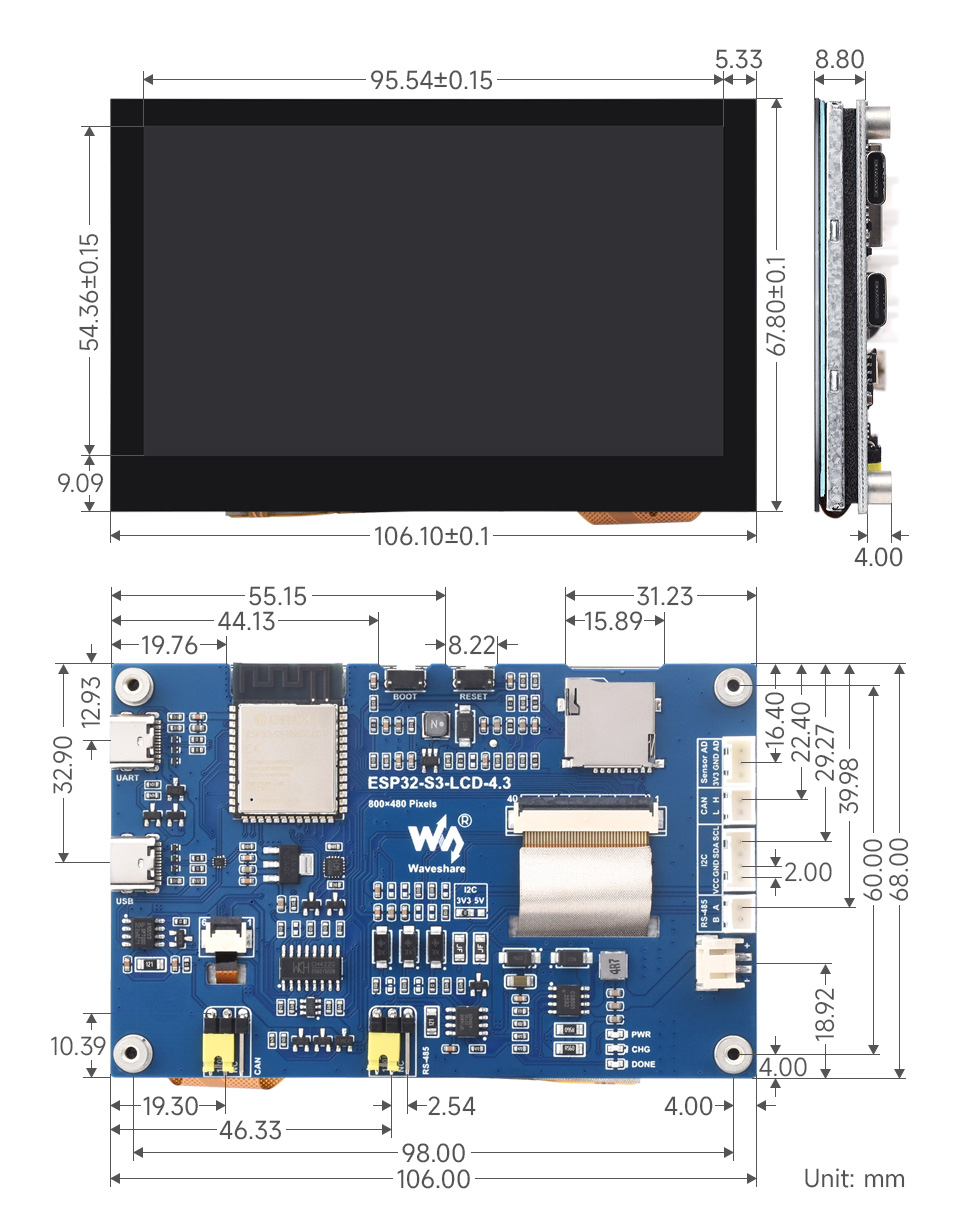
* **PH2.0 battery interface**: The development board uses a high-efficiency charge/discharge management chip CS8501, which can boost a single lithium battery to 5V, the current charging current is 580mA, and the user can change the charging current by replacing the R45 resistor, please refer to [ESP32-S3-Touch-LCD-4.3 Schematic](https://files.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3/manual/ESP32-S3-Touch-LCD-4.3-Sch.pdf) for details.

### Dimensions

#### Without Touch Version

[](https://www.waveshare.com/wiki/File:ESP32-S3-LCD-4.3-details-size.jpg)

#### With Touch Version

[](https://www.waveshare.com/wiki/File:Esp32-s3-touch-lcd-4.3-003.jpg)

### Specifications

|  |  |
| --- | --- |
| **Basic parameters** | |
| Processor | High-performance 32-bit Xtensa LX7 dual-core processor with a frequency up to 240MHz |
| Wifi/Bluetooth | Supports 2.4 GHz Wi-Fi (802.11 b/g/n) and Bluetooth 5 (LE) with onboard antennas |
| Flash | 16MB Flash |
| PSRAM | 8MB PSRAM |
| Power supply range | TypeC 5V |
| **Screen parameters** | |
| Resolution | 800 x 480 |
| Display interface | RGB |
| Display panel | IPS |
| Viewing angle | 160° |
| Screen brightness | 270 Cd/m² |
| Touch type | Capacitive |
| Touch panel | toughened glass |
| **Peripheral interface** | |
| Communication interface | CAN, RS485, I2C, USB |
| **Other** | |
| Power consumption | 5V 450mA |
| Operating temperature | 0℃ ~ 65℃ |
| Product size (L×W) | Non-touch version: 105.4×67.1mm  Touch version: 106.1×67.8mm |

## Usage Instructions

Currently there are two development tools and frameworks, **Arduino IDE** and **ESP-IDF**, providing flexible development options, you can choose the right development tool according to your project needs and personal habits.

### Development Tools

|  |  |
| --- | --- |
|  | Arduino IDE Arduino IDE is an open source electronic prototyping platform, convenient and flexible, easy to get started. After a simple learning, you can start to develop quickly. At the same time, Arduino has a large global user community, providing an abundance of open source code, project examples and tutorials, as well as rich library resources, encapsulating complex functions, allowing developers to quickly implement various functions. |
|  | ESP-IDF ESP-IDF, or full name Espressif IDE, is a professional development framework introduced by Espressif Technology for the ESP series chips. It is developed using the C language, including a compiler, debugger, and flashing tool, etc., and can be developed via the command lines or through an integrated development environment (such as Visual Studio Code with the Espressif IDF plugin). The plugin offers features such as code navigation, project management, and debugging, etc. |

Each of these two development approaches has its own advantages, and developers can choose according to their needs and skill levels. Arduino are suitable for beginners and non-professionals because they are easy to learn and quick to get started. ESP-IDF is a better choice for developers with a professional background or high performance requirements, as it provides more advanced development tools and greater control capabilities for the development of complex projects.

### Components Preparation

* ESP32-S3-Touch-LCD-4.3 x1
* TF card x 1
* USB cable (Type-A to Type-C) x 1
* USB TO RS485 Bidirectional Converter x1
* USB to CAN adapter analyzer x1

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-47.png)

## Precautions

* The development board has an onboard automatic download circuit, the Type C port at the UART silk screen is used for program download and log printing. After downloading the program, press the RESET button to run the program
* Please pay attention to the PCB antenna area when using, and avoid attaching other metals or plastic parts to the PCB antenna
* The TF card uses SPI/MMC to communicate, note that the SD\_CS pin needs to be driven by the EXIO4 of the CH422G
* The development board uses PH2.0 header to lead out ADC, CAN, I2C, RS485 peripheral pins, and connects sensor components using PH2.0 to 2.54mm male connector
* The CAN and RS485 peripherals use jumpers to connect the 120 ohm resistor by default, and NC is optional to cancel terminal resistor connection
* The 4.3inch screen occupies the vast majority of the GPIO, and the development board uses the [CH422G](https://files.waveshare.com/wiki/common/CH422DS1_EN.pdf) chip to expand the IO for resetting, turning off and on the backlight, etc.
* **The development board uses USB to download the demo. If the port cannot be recognized, please enter boot mode (press and hold the boot button, then connect to the computer, and then release the boot button). After downloading the demo, press the RESET button to run the demo.**
* Currently, under ESP-IDF v5.3, the average frame rate limit for running the LVGL benchmark demo with a single core is 26 frames per second, corresponding to an interface frame rate of 41 (PCLK 21 MHz). Before compilation, ESP32 and LVGL need to be configured through menuconfig:

CONFIG\_FREERTOS\_HZ=1000

CONFIG\_ESP\_DEFAULT\_CPU\_FREQ\_MHZ\_240=y

CONFIG\_ESPTOOLPY\_FLASHMODE\_QIO=y

CONFIG\_ESPTOOLPY\_FLASHFREQ\_120M=y [Need to be consistent with PSRAM]

CONFIG\_SPIRAM\_MODE\_OCT=y

CONFIG\_IDF\_EXPERIMENTAL\_FEATURES=y and CONFIG\_SPIRAM\_SPEED\_120M=y [Need to be consistent with FLASH]

CONFIG\_SPIRAM\_FETCH\_INSTRUCTIONS=y

CONFIG\_SPIRAM\_RODATA=y

CONFIG\_ESP32S3\_DATA\_CACHE\_LINE\_64B=y

CONFIG\_COMPILER\_OPTIMIZATION\_PERF=y

#The following LVGL configuration items are helpful for frame rate improvement (LVGL v8.3):

#define LV\_MEM\_CUSTOM 1 or CONFIG\_LV\_MEM\_CUSTOM=y

#define LV\_MEMCPY\_MEMSET\_STD 1 or CONFIG\_LV\_MEMCPY\_MEMSET\_STD=y

#define LV\_ATTRIBUTE\_FAST\_MEM IRAM\_ATTR or CONFIG\_LV\_ATTRIBUTE\_FAST\_MEM=y

* For detailed LCD and LVGL performance descriptions, please refer to [Documentation](https://files.waveshare.com/wiki/common/Performance.pdf)
* PH1.25 lithium battery holder only supports a single 3.7V lithium battery, do not use multiple sets of battery packs to connect to charge and discharge at the same time, and it is recommended that the capacity of a single battery is less than 2000mAH
* The CH422G and touch of the board occupy the following slave address, please do not use I2C devices with the same address:

0 1 2 3 4 5 6 7 8 9 a b c d e f

00: - - - - - - - - - - - - - - - -

10: - - - - - - - - - - - - - - - -

20: 20 21 22 23 24 25 26 27 - - - - - - - -

30: 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f

40: - - - - - - - - - - - - - - - -

50: - - - - - - - - - - - - - - 5d -

60: - - - - - - - - - - - - - - - -

70: - - - - - - - - - - - - - - - -

Before operating, it is recommended to browse the table of contents to quickly understand the document structure. For smooth operation, please read the [FAQ](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#FAQ) carefully to understand possible problems in advance. All resources in the document are provided with hyperlinks for easy download.

# Working with Arduino

This chapter introduces setting up the Arduino environment, including the Arduino IDE, management of ESP32 boards, installation of related libraries, program compilation and downloading, as well as testing demos. It aims to help users master the development board and facilitate secondary development.

## Environment Setup

### Download and Install Arduino IDE

* Click to visit the [Arduino official website](https://www.arduino.cc/en/software), select the corresponding system and system bit to download The version of the Arduino IDE needs to be ≥ 1.8, and the path of installation must not be Chinese, otherwise there will be an error when compiling.   
  [](https://www.waveshare.com/wiki/File:ESP32-S3-AMOLED-1.91-Ar-software-01.png)
* Run the installer and install all by default

The environment setup is carried out on the Windows 10 system, Linux and Mac users can access [Arduino-esp32 environment setup](https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html) for reference

### Install Arduino-ESP32

* Regarding ESP32-related motherboards used with the Arduino IDE, the **esp32 by Espressif Systems** library must be installed first.
* It is generally recommended to use **Install Online**. If online installation fails, use **Install Offline**.
* To install the Arduino-ESP32 tutorial, please refer to [Arduino board manager tutorial](https://www.waveshare.com/wiki/Arduino_Board_Managers_Tutorial)
* The ESP32-S3-Touch-LCD-4.3 development board comes with an offline package. Click here to download: [esp32\_package\_3.0.7\_arduino offline package](https://drive.google.com/drive/folders/1Pcs_A4FKWvdSHnz9lEBYqOpr-noTMbIv?usp=sharing)

|  |  |  |
| --- | --- | --- |
| * ESP32-S3-Touch-LCD-4.3 Development board installation instructions | | |
| **Board name** | **Board installation requirement** | **Version number requirement** |
| ESP32-S3-Touch-LCD-4.3 | "Install Offline" / "Install Online" | 3.0.0 and above |

### Install Libraries

* When installing Arduino libraries, there are usually two ways to choose from: **Install online** and **Install offline**. **If the library installation requires offline installation, you must use the provided library file**   
  For most libraries, users can easily search and install them through the online library manager of the Arduino software. However, some open-source libraries or custom libraries are not synchronized to the Arduino Library Manager, so they cannot be acquired through online searches. In this case, users can only manually install these libraries offline.
* For library installation tutorial, please refer to [Arduino library manager tutorial](https://www.waveshare.com/wiki/Arduino_Library_Manager_Tutorial)
* ESP32-S3-Touch-LCD-4.3 library file is stored in the sample program, click here to jump:[ESP32-S3-Touch-LCD-4.3 Demo](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#Resources)

|  |  |  |  |
| --- | --- | --- | --- |
| * ESP32-S3-Touch-LCD-4.3 library file installation instructions | | | |
| **Library Name** | **Description** | **Version** | **Library Installation Requirement** |
| ESP32\_Display\_Panel | ESP32 Microcontroller's specific display panel control library | v0.1.4 and above | "Install Online" or "Install Offline" |
| ESP32\_IO\_Expander | ESP32's I/O expansion library | v0.0.4 or later | "Install Online" or "Install Offline" |
| lvgl | LVGL graphical library | v8.4.0 | "Install Offline" |
| lv\_conf.h | LVGL configuration file | —— | "Install Offline" |

## Run the First Arduino Demo

If you are just getting started with ESP32 and Arduino, and you don't know how to create, compile, flash, and run Arduino ESP32 programs, then please expand and take a look. Hope it can help you!

#### New Project

* Run the Arduino IDE and select **File** -> **New Sketch**
* Enter the code:

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

}

void loop() {

// put your main code here, to run repeatedly:

Serial.println("Hello, World!");

delay(2000);

}

* Save the project and select **File** -> **Save As...**. In the pop-up menu, select the path to save the project, and enter a project name, such as **Hello\_World**, click **Save**

### Compile and Flash Demos

* Select the corresponding development board, take the ESP32S3 motherboard as an example:

①. Click to select the dropdown menu option **Select Other Board and Port**;   
②. Search for the required development board model **esp32s3 dev module** and select;   
③. Select **COM Port**;   
④. **Save** the selection.

* If the ESP32S3 mainboard only has a USB port, you need to enable **USB CDC**, as shown in the following diagram:
* Compile and upload the program:

①. Compile the program; ②. Compile and download the program; ③. Download successful.

* Open the **Serial Monitor** window, and the demo will print "Hello World!" every 2 seconds, and the operation is as follows:

### Demo

|  |  |  |
| --- | --- | --- |
| * ESP32-S3-Touch-LCD-4.3 Demo | | |
| **Demo** | **Basic Description** | **Dependency Library** |
| 01\_I2C\_Test | Test I2C header | - |
| 02\_RS485\_Test | Test RS485 header | - |
| 03\_SD\_Test | Test TF card slot | - |
| 04\_Sensor\_AD | Test ADC header | - |
| 05\_UART\_Test | Test UART | - |
| 06\_TWAItransmit | Test CAN header | - |
| 07\_TWAIreceive | Test CAN header | - |
| 08\_DrawColorBar | Test RGB screen | ESP32\_Display\_Panel |
| 09\_lvgl\_Porting | Test LVGL porting | LVGL, ESP32\_Display\_Panel |

* **If the ESP32 version number is 3.0.6 or above, the ESP32-S3-Touch-LCD-4.3 supports direct model selection. After selecting the model directly, some parameters do not need to be modified by default**
  + Take ESP32-S3-LCD-1.69 as an example

[](https://www.waveshare.com/wiki/File:ESP32-S3-LCD-1.69-demo-02.png)

### 01\_I2C\_Test

#### Hardware connection

* Connect the board to the computer using a USB cable

#### Code analysis

* **loop()**:

loop()The function is the main loop part of the program, and its core function is to scan devices on the I2C bus.   
First, variables were defined to store error codes, device addresses, and to record the number of devices found.   
Then, iterate through possible I2C device addresses from location 0x01 to 0x7f using a loop. For each address, use Wire.beginTransmission(address) to start the transmission to the device at the specific address, and then use Wire.endTransmission() to end the transmission and get the error code.   
If the error code is 0, an I2C device was found at that address, print the device address and increase the count of the number of devices. If the error code is not 2 (indicating that the device is not responding), the error code and the corresponding address are printed.   
Finally, if no I2C devices are found, print the appropriate message and use delay(5000) to pause the program for 5 seconds and scan again.

#### Demo flashing

* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

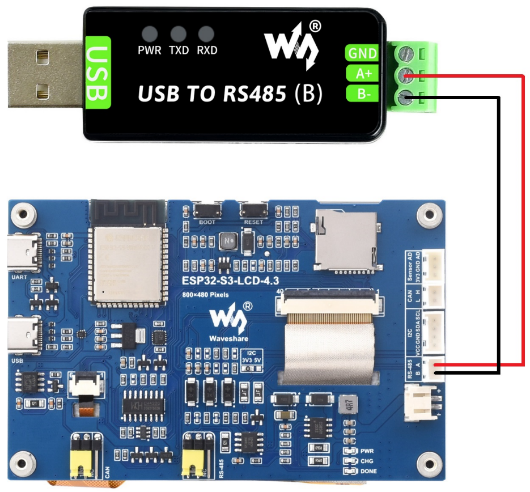
* The serial monitor prints the device address on the I2C bus

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-6.png)

### 02\_RS485\_Test

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the development board to [USB to RS485 converter](https://www.waveshare.com/USB-TO-RS485-B.htm), as shown in the figure

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-43.png)

#### Code analysis

* **setup()**:

setup function is primarily used for initializing serial communication   
Use the RS485.begin function to initialize serial port Serial1, set the baud rate, data format, and specify the receive and transmit pins. Then, through a loop, ensure the serial port initialization is successful.

* **loop()** :

loop function is the main loop part of the program, and its main function is to implement simple 485 communication data return   
By checking whether there is data available at the serial port, if there is data, it reads a byte and sends it back immediately, so that the received 485 data can be sent back intact

#### Demo flashing

* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

* Open the serial port debugging assistant to send a message to the ESP32-S3-Touch-LCD-4.3 device, and the device will return the received message to the serial port debugging assistant

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-32.png)

### 03\_SD\_Test

#### Hardware connection

* Connect the board to the computer using a USB cable
* Insert the TF card into the board

#### Code analysis

* **setup()**:
* setup function mainly performs a series of initialization operations and tests on the TF card
  + First, it initializes serial port communication and sets the baud rate to 115200. Then create a ESP\_IOExpander\_CH422G object to manage the extended IO pins, initialize and set multiple pins as output modes, and control the status of pins such as touchscreen reset (TP\_RST), LCD backlight (LCD\_BL), LCD reset (LCD\_RST), TF card select (SD\_CS), and USB select (USB\_SEL).
  + Next, use the extended GPIO pins to handle the TF card, initialize SPI communication, and attempt to mount the TF card. If the mount fails, an error message will be output and returned. If the mount is successful, the type of TF card will be detected and output, and information about the size of the TF card will also be output.
  + After that, perform a series of tests on file system operations, including listing directory contents, creating directories, deleting directories, writing to files, appending content to files, reading files, deleting files, renaming files, and testing file input and output, and output the total space and used space size of the TF card.

#### Demo flashing

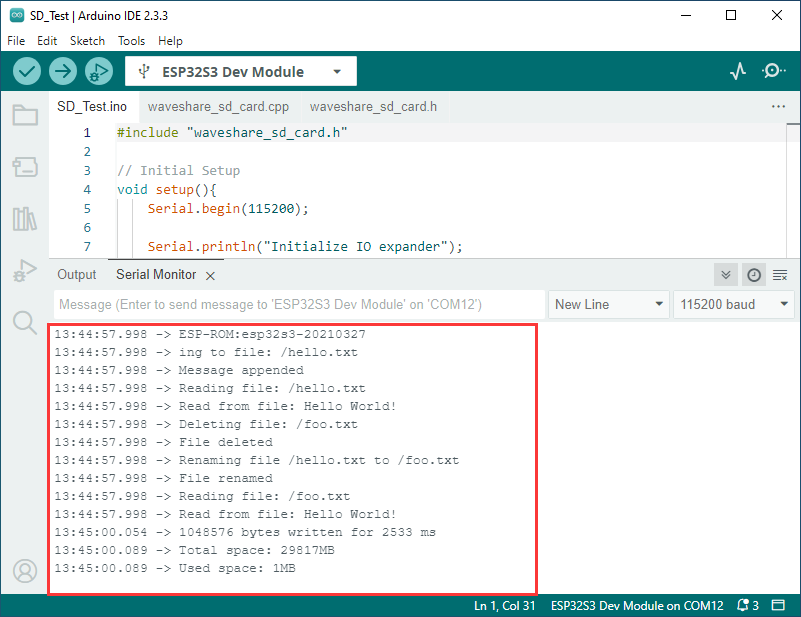
* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

* The ESP32-S3-Touch-LCD-4.3 can identify the type and size of an TF card, and then perform file operations such as adding, deleting, modifying, and querying.

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-12.png)

### 04\_Sensor\_AD\_Test

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the PH2.0 to 2.54mm male connector to the Sensor AD interface of the board

#### Code analysis

* **setup()**:

The setup function first initializes the serial communication, then initializes the ADC, and sets the ADC resolution to 12 bits (0-4096).

* **loop()** :

The loop function reads the current AD value every 100ms and prints the current AD value.

#### Demo flashing

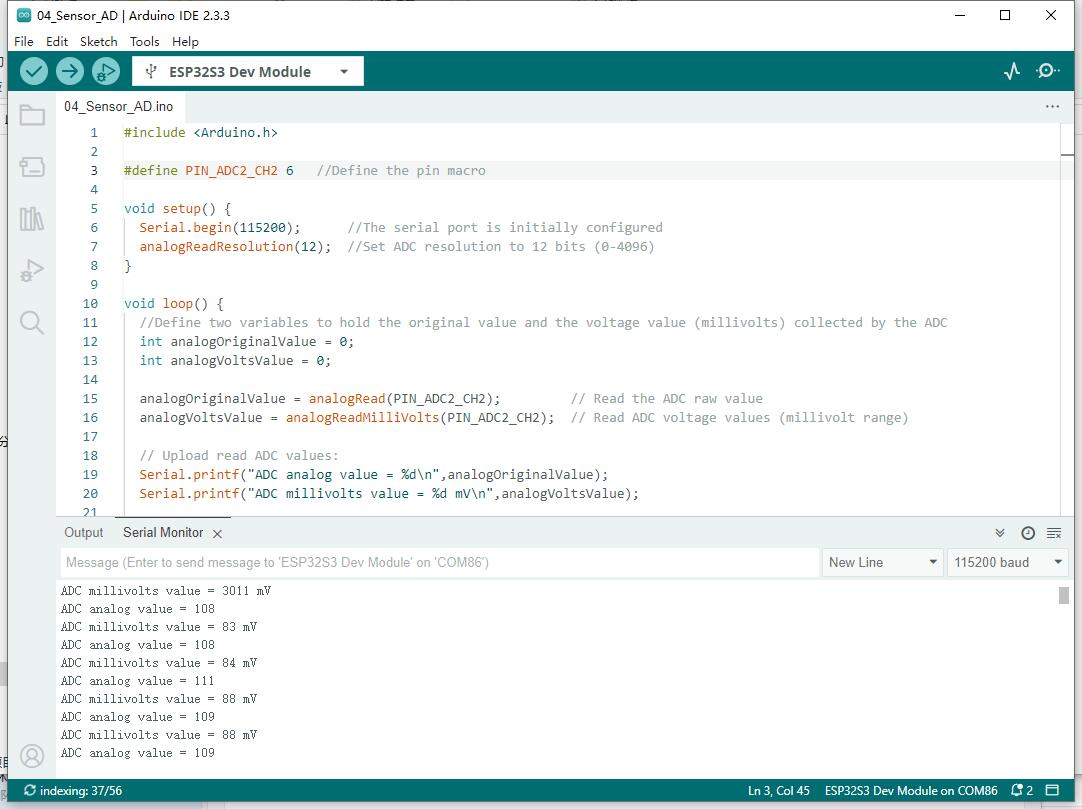
* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

* Result demonstration: ESP32-S3-Touch-LCD-4.3 will set the ADC resolution, read the current AD value, and print to the serial port terminal

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-40.jpg)

### 05\_UART\_Test

#### Hardware connection

* Connect the UART port of the board to the computer using a USB cable

#### Code analysis

* **setup()**:

setup function is primarily used for initializing serial communication   
Use the UART.begin function to initialize serial port Serial , set the baud rate, data format, and specify the receive and transmit pins. Then, through a loop, ensure the serial port initialization is successful.

* **loop()** :

loop function is the main loop part of the program, and its main function is to implement simple UART communication data return   
By checking whether there is data available at the serial port, if there is data, it reads a byte and sends it back immediately, so that the received UART data can be sent back intact

#### Demo flashing

* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

* Open the serial port debugging assistant to send a message to the ESP32-S3-Touch-LCD-4.3 device, and the device will return the received message to the serial port debugging assistant

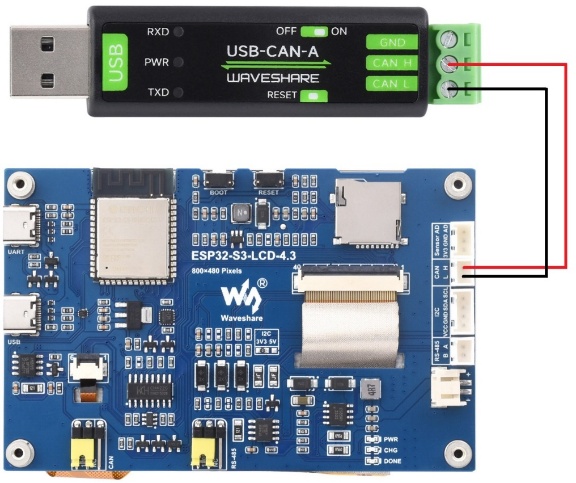
[Immagine che contiene testo, schermata, software, Pagina Web

Il contenuto generato dall'IA potrebbe non essere corretto.](https://www.waveshare.com/wiki/File:ESP32-Arduino-45.png)

### 06\_TWAItransmit

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the development board to [USB-CAN-A](https://www.waveshare.com/USB-CAN-A.htm), as shown in the figure

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-44.png)

#### Code analysis

* **waveshare\_twai\_transmit()**:
* waveshare\_twai\_transmit() function primarily handles the transmission and alarm processing for TWAI (an interface similar to CAN bus)
  + First, it checks if any alarms have occurred. Read the triggered alerts by calling twai\_read\_alerts and obtain the TWAI status information into a twai\_status\_info\_t structure. Then, according to the different alarms triggered, perform corresponding processing. For example, if an error passive alarm, a bus error alarm, a transmission failure alarm, or a transmission success alarm is triggered, print the corresponding message and output some status information such as the bus error count, the number of messages to be sent, the transmission error counter, and the transmission failure counter.
  + It then takes the current time (in milliseconds) and checks if it's time to send the message. If the difference between the current time and the last time a message was sent is greater than or equal to the set transmission time interval TRANSMIT\_RATE\_MS , then update the last send time to the current time and call the send\_message function to send a message. send\_message function configures and queues a message containing a specific identifier, data length, and data content for transmission. If the queuing is successful, it prints a success message; otherwise, it prints a failure message. After sending, it clears the message data array.

#### Demo flashing

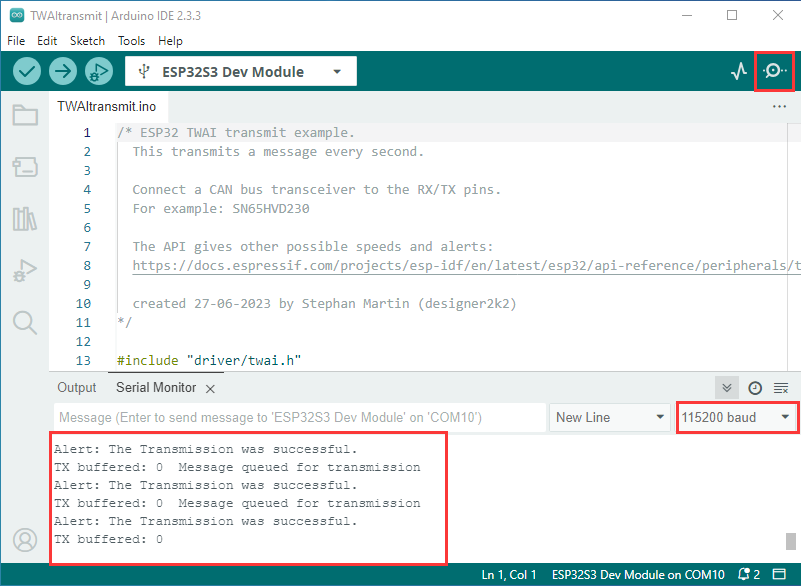
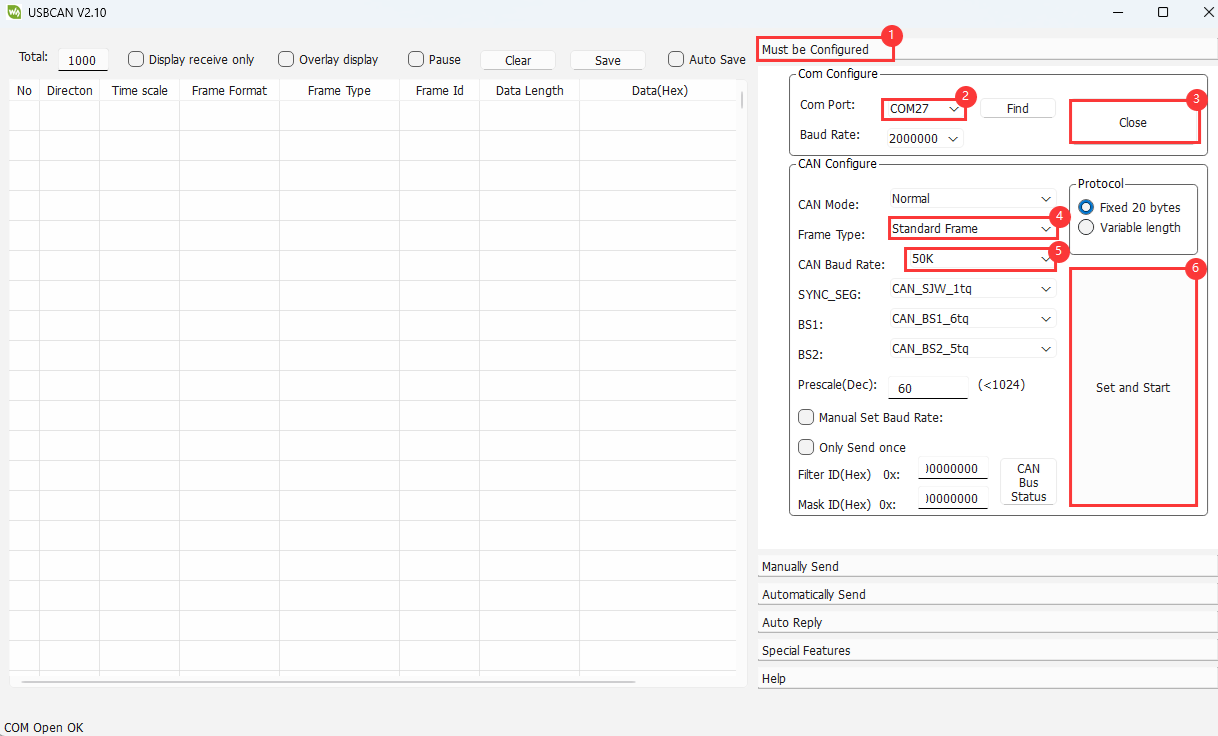
* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

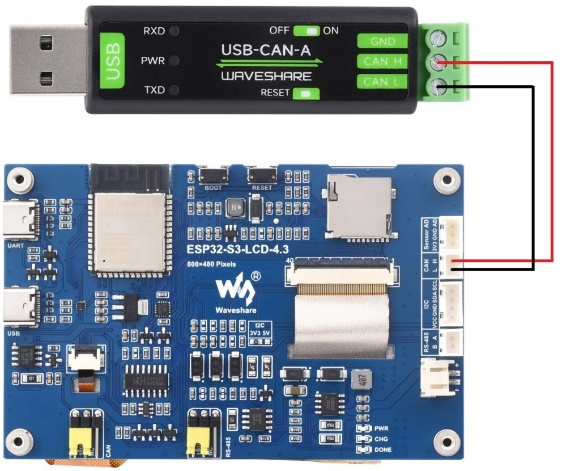
* Serial port print indicates successful CAN message transmission. After configuring [USB-CAN-A\_TOOL](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#Resources), you can observe the CAN messages sent by the ESP32-S3-Touch-LCD-4.3 upon startup.

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-20.png)[](https://www.waveshare.com/wiki/File:ESP32-Arduino-CN-1.png)

### 07\_TWAIreceive

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the development board to [USB-CAN-A](https://www.waveshare.com/USB-CAN-A.htm), as shown in the figure

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-44.png)

#### Code analysis

* **waveshare\_twai\_receive()**:
  + First, the triggered alarm is read and the status information is obtained, and the corresponding processing is carried out according to different alarm conditions, such as printing error passive, bus error, receiving queue full and other alarm information and related counts.
  + When a data alert is triggered, the loop receives messages and calls the handle\_rx\_message function to process them. This function determines the message format and prints the message identifier and data content (excluding remote transmission requests), effectively handling received messages on the TWAI bus and responding to alerts.

#### Demo flashing

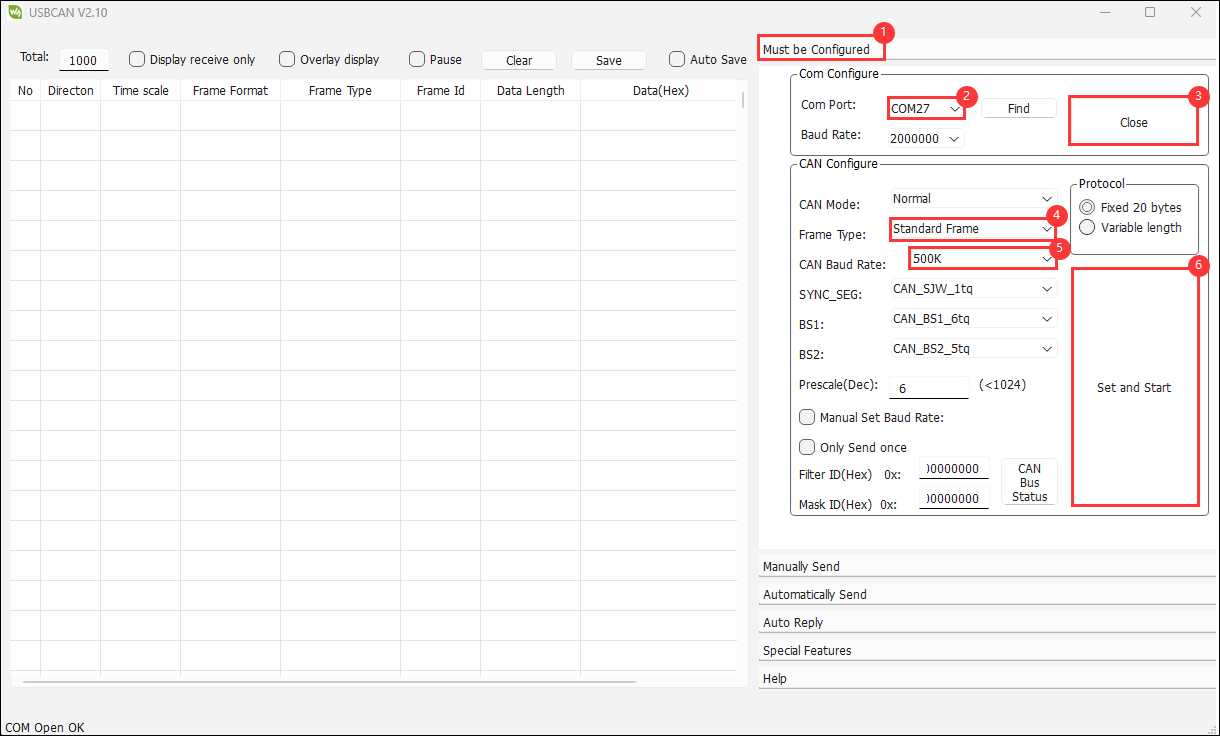
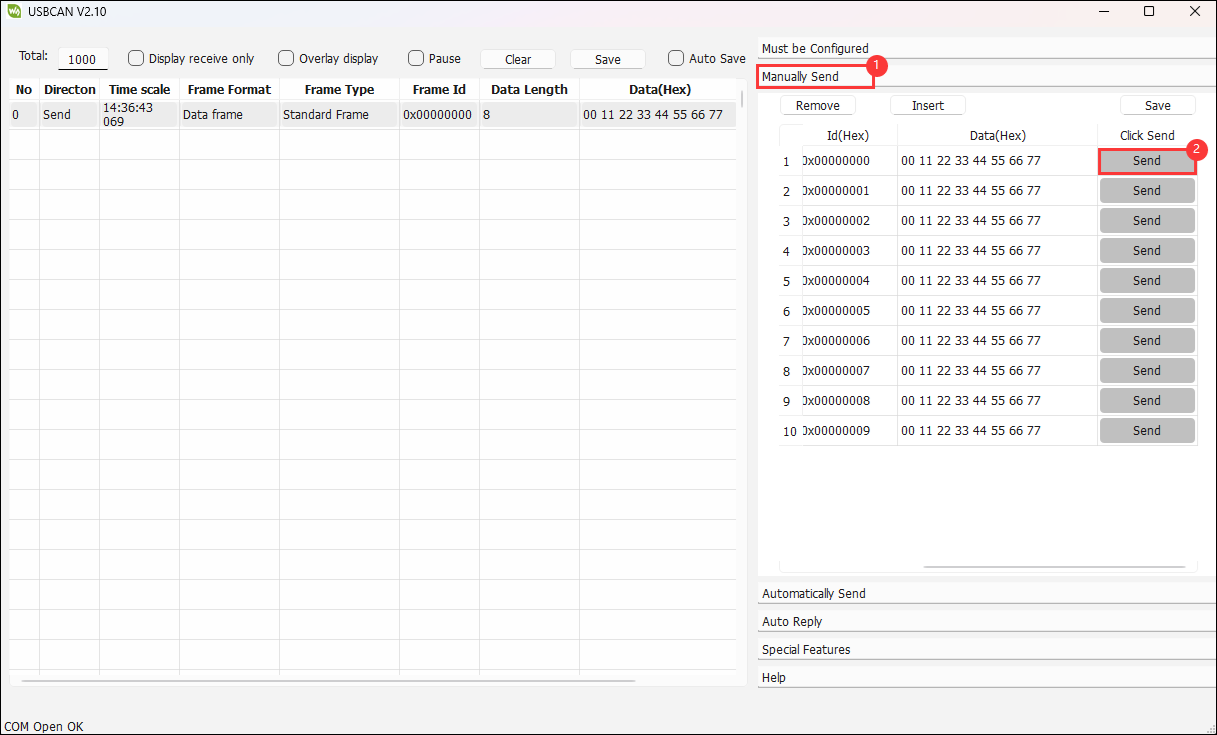
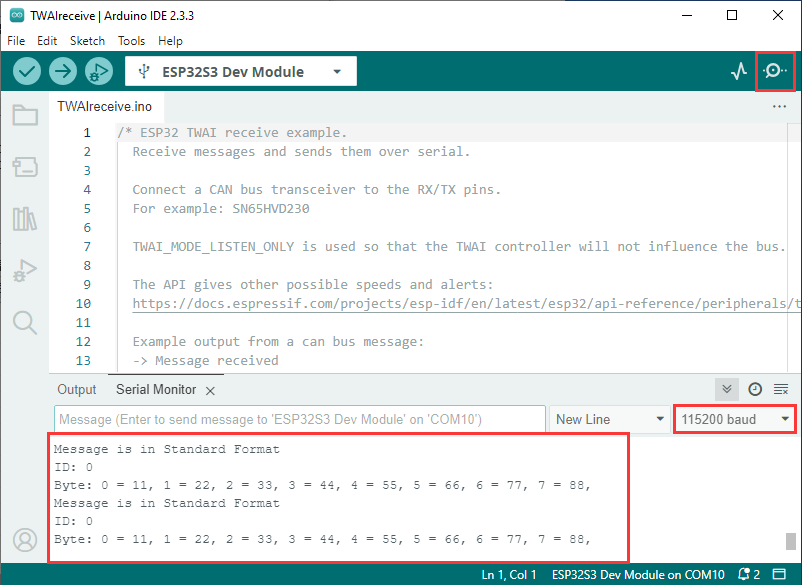
* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-41.png)

* Flash the demo

#### Result demonstration

* ESP32-S3-Touch-LCD-4.3 waits [USB-CAN-A\_TOOL](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#Resources) to send a message. If the message is received successfully, it will be printed to the serial port

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-CN-2.png)[](https://www.waveshare.com/wiki/File:ESP32-Arduino-23.png)[](https://www.waveshare.com/wiki/File:ESP32-Arduino-22.png)

### 08\_DrawColorBar

#### Hardware connection

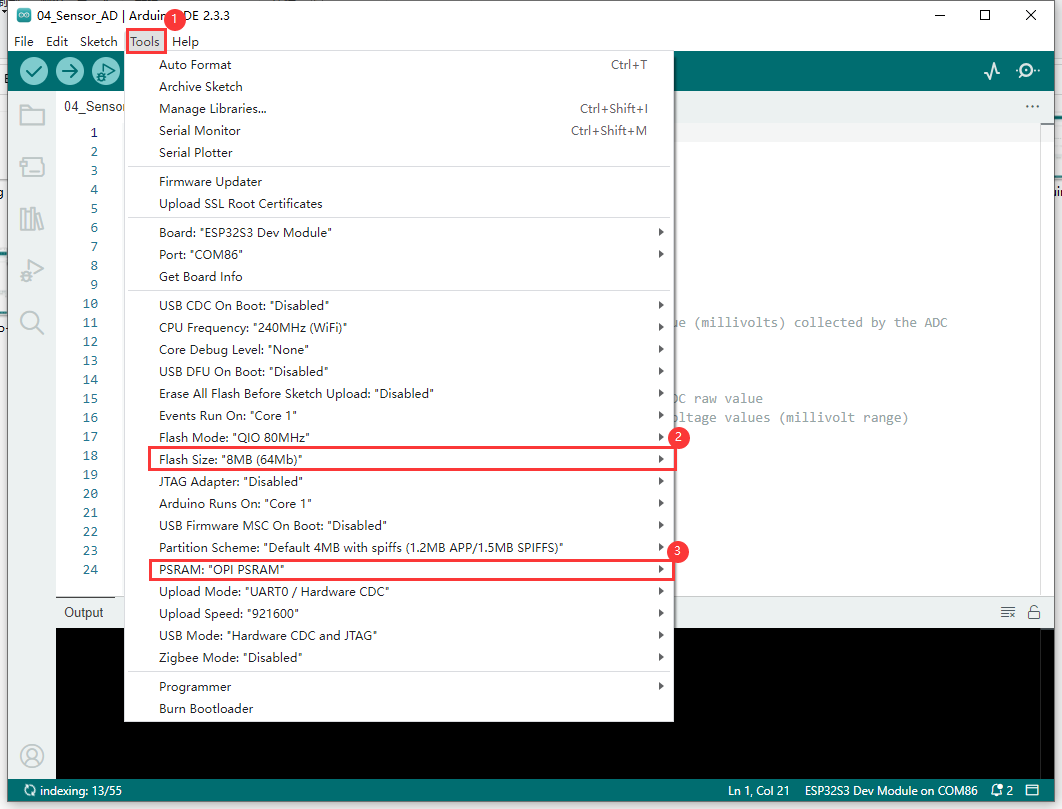
* Connect the board to the computer using a USB cable

#### Code analysis

* **waveshare\_lcd\_init()**:
  + First, it prints out "Initialize IO expander" to indicate the start of initializing the IO expander. Then create a ESP\_IOExpander\_CH422G instance, initialize it, and start its operation. Set IO0 - IO7 pins to output mode, and set the touch screen reset pin (TP\_RST) and the LCD reset pin (LCD\_RST) to high level while turning off the LCD backlight (LCD\_BL) and wait 100 ms.
  + Next, print "Create RGB LCD bus", create an RGB panel bus object ESP\_PanelBus\_RGB, configure its pins, width, height, RGB timing frequency and timing parameters, etc., set the bounce buffer size and display the active low flag, and then start the panel bus operation.
  + Then, print "Create LCD device", create an LCD object ESP\_PanelLcd, pass in parameters such as panel bus object, number of color bits, and reset pins to initialize, reset, and start the operation. If EXAMPLE\_ENABLE\_PRINT\_LCD\_FPS is defined, the callback function ending with VSync is attached to the LCD object.
  + Finally, print "Draw color bar from top left to bottom right, the order is B - G - R", call the colorBarTest function to draw the color bar on the LCD, and turn on the LCD backlight.

#### Demo flashing

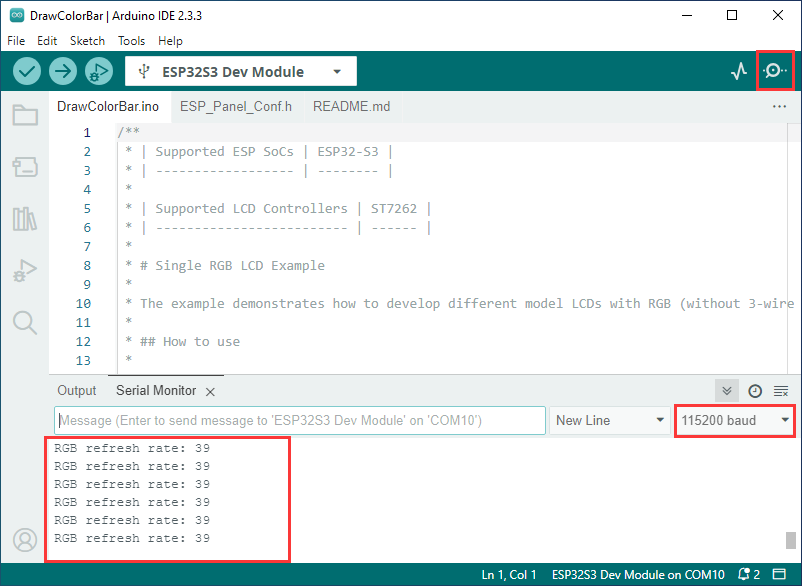
* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-42.png)

* Flash the demo

#### Result demonstration

* Serial port prints log, and the screen lights up

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-24.png)[](https://www.waveshare.com/wiki/File:800px-ESP32-Arduino-25.png)

### 09\_lvgl\_Porting

#### Hardware connection

* Connect the board to the computer using a USB cable

#### Code analysis

* **setup()**:

Initialize serial communication at a baud rate of 115200. Next, create and initialize the IO expander, set the pin mode and state, and initialize the GT911 touch screen. Then create and initialize the panel device, configure the RGB bus as needed. After that, initialize LVGL, create a simple label or an example or demo function that optionally calls LVGL, and release the mutex lock at the end

* **loop()** :

Only prints "IDLE loop" and waits for 1 second without any other substantial action. The overall goal is to build an LVGL-based user interface environment.

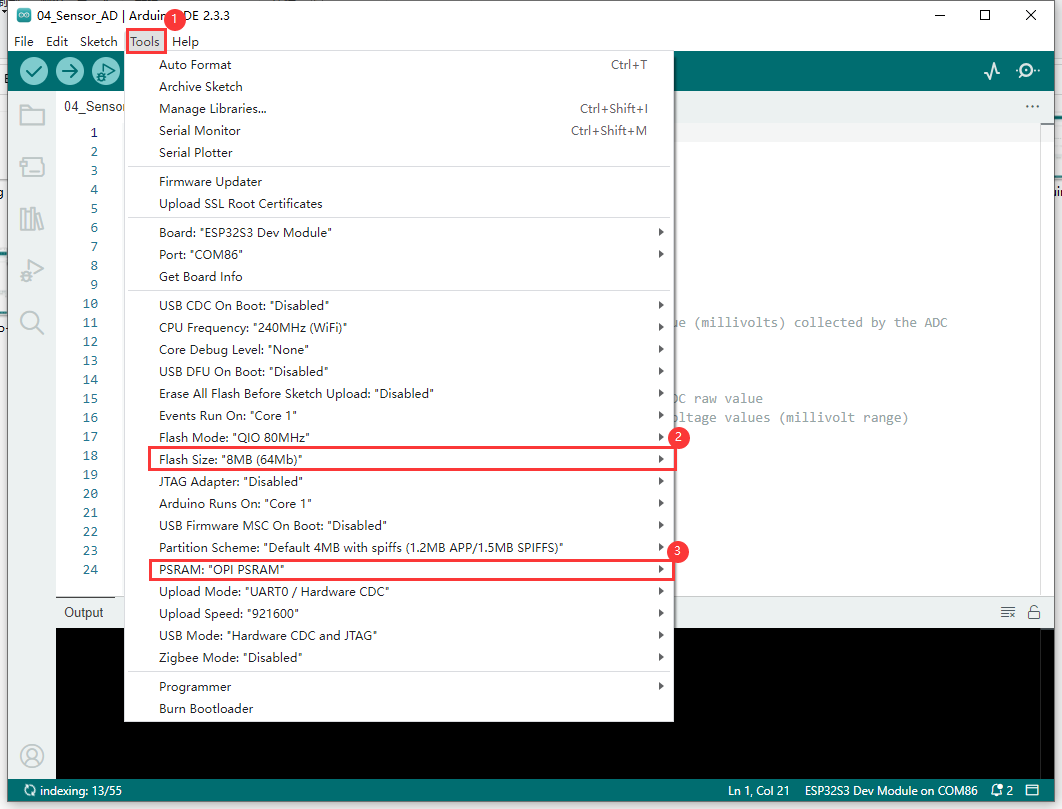
#### Code modification

* In ESP\_Panel\_Board\_Custom.h there is a macro definition for selecting whether to turn on the touch function, with a value of 0 corresponds to turning off touch, and a value of 1 corresponds to turning on touch, which can be selected according to the model purchased

#define ESP\_OPEN\_TOUCH 0 // 1 initiates the touch, 0 closes the touch.

#### Demo flashing

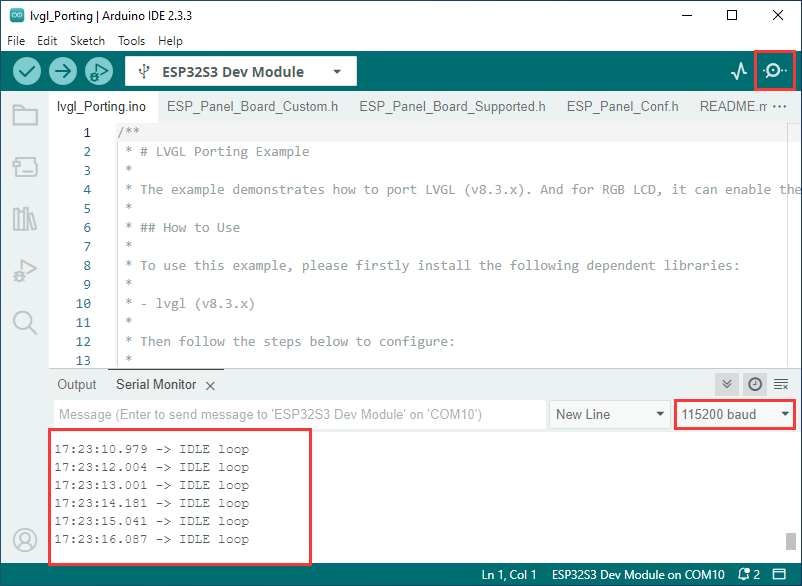
* Select the development board ESP32S3 Dev Module and port
* Set development board parameters

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-42.png)

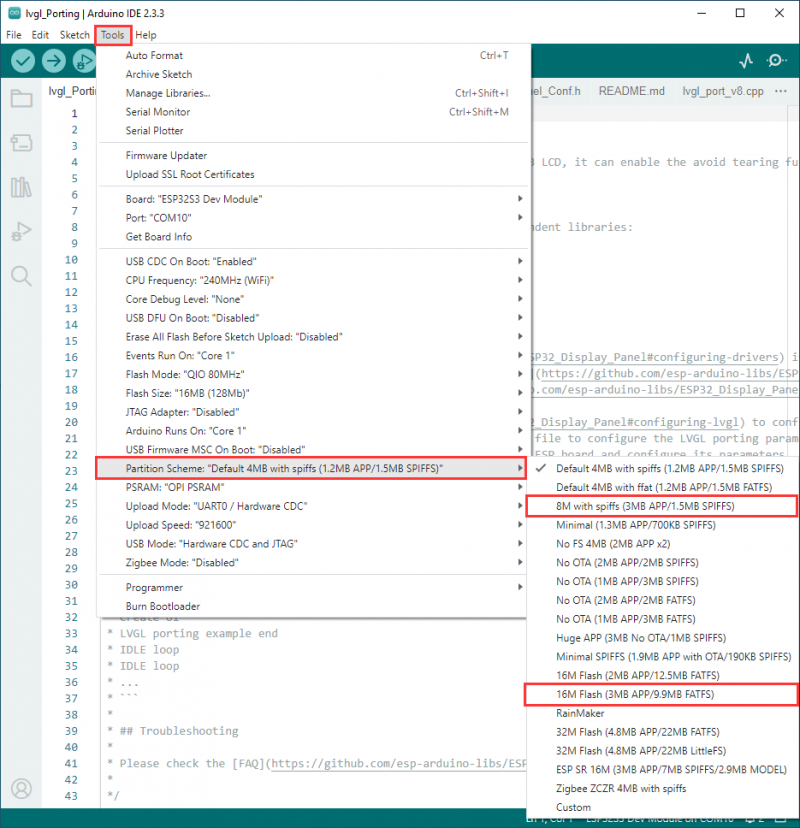
* Flash the demo

#### Result demonstration

* Serial port prints the screen refresh rate, and the screen lights up

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-26.png)   
[Screen lights up\_video](https://www.youtube.com/watch?v=luICguq9cjo)

### Other instructions

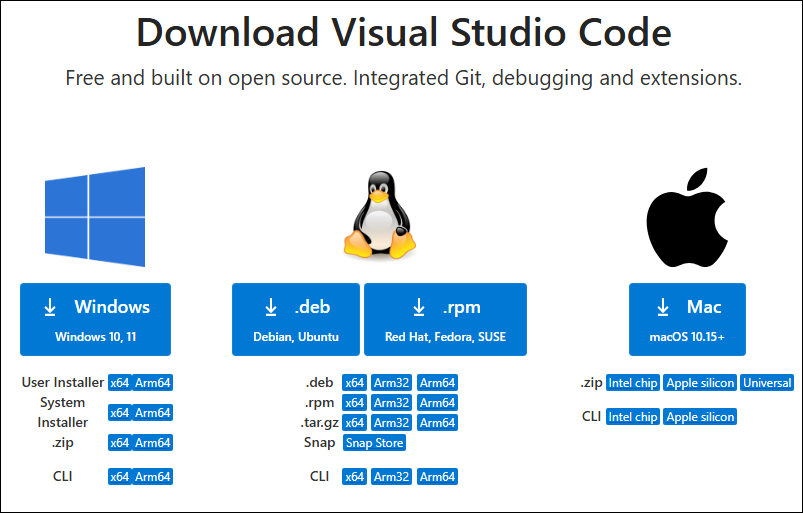
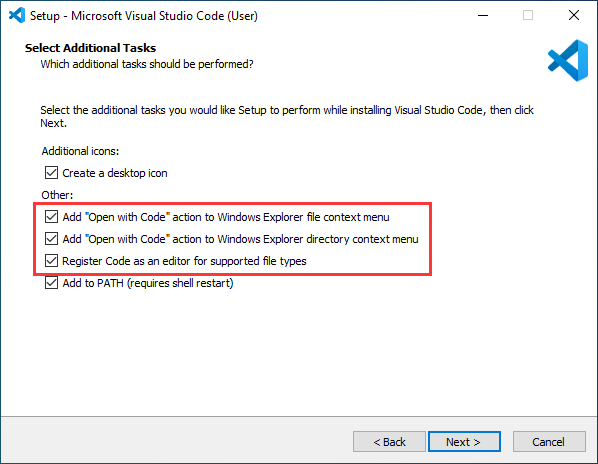
* Screen drifting occurs during use, please refer to [ESP official FAQ](https://docs.espressif.com/projects/esp-faq/en/latest/software-framework/peripherals/lcd.html#why-do-i-get-drift-overall-drift-of-the-display-when-esp32-s3-is-driving-an-rgb-lcd-screen)
* When using your own UI program, there is a lack of memory, you can click Tools to select a larger partition table   
  [](https://www.waveshare.com/wiki/File:800px-ESP32-Arduino-28.png)
* The version of lvgl used is 8.4, and you can query and use the LVGL API through the following documentation
  + [LVGL documentation](https://docs.lvgl.io/8.4/)

# Working with ESP-IDF

This chapter introduces setting up the ESP-IDF environment setup, including the installation of Visual Studio and the Espressif IDF plugin, program compilation, downloading, and testing of demos, to assist users in mastering the development board and facilitating secondary development.

## Environment Setup

### Download and Install Visual Studio

* Open the download page of [VScode official website](https://code.visualstudio.com/download), choose the corresponding system and system bit to download   
  [](https://www.waveshare.com/wiki/File:ESP32-S3-AMOLED-1.91-VScode-01.png)
* After running the installation package, the rest can be installed by default, but here for the subsequent experience, it is recommended to check boxes 1, 2, and 3   
  [](https://www.waveshare.com/wiki/File:ESP32-S3-AMOLED-1.91-VScode-02.png)
  + After the first two items are enabled, you can open VSCode directly by right-clicking files or directories, which can improve the subsequent user experience.
  + After the third item is enabled, you can select VSCode directly when you choose how to open it

The environment setup is carried out on the Windows 10 system, Linux and Mac users can access [ESP-IDF environment setup](https://docs.espressif.com/projects/esp-idf/en/v5.1.4/esp32s3/get-started/windows-setup.html) for reference

### Install Espressif IDF Plugin

* It is generally recommended to use **Install Online**. If online installation fails due to network factor, use **Install Offline**.
* For more information about how to install the Espressif IDF plugin, see [Install Espressif IDF Plugin](https://www.waveshare.com/wiki/Install_Espressif_IDF_Plugin_Tutorial)

## Run the First ESP-IDF Demo

If you are just getting started with ESP32 and ESP-IDF, and you don't know how to create, compile, flash, and run ESP-IDF ESP32 programs, then please expand and take a look. Hope it can help you!

### New Project

### Create Demo

* Using the shortcut F1, enter esp-idf:show examples projects
* Select your current IDF version
* Take the Hello world demo as an example

①Select the corresponding demo   
②Its readme will state what chip the demo applies to (how to use the demo and the file structure are described below, omitted here)   
③Click to create the demo

* Select the path to save the demo, and require that the demos cannot use folders with the same name

### Modify COM Port

* The corresponding COM ports are shown here, click to modify them
* Please select the COM ports according to your device (You can view it from the device manager)
* In case of a download failure, please press the Reset button for more than 1 second or enter download mode, and wait for the PC to recognize the device again before downloading once more

### Modify Driver Object

* Select the object we need to drive, which is our main chip ESP32S3
* Choose the openocd path, it doesn't affect us here, so let's just choose one

### Other Status Bar Functions

①.ESP-IDF Development Environment Version Manager, when our project requires differentiation of development environment versions, it can be managed by installing different versions of ESP-IDF. When the project uses a specific version, it can be switched to by utilizing it   
②.Device flashing COM port, select to flash the compiled program into the chip   
③.Select set-target chip model, select the corresponding chip model, for example, ESP32-P4-NANO needs to choose esp32p4 as the target chip   
④.menuconfig, click it to Modify sdkconfig configuration file [Project configuration details](https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/api-reference/kconfig.html)   
⑤.fullclean button, when the project compilation error or other operations pollute the compiled content, you can clean up all the compiled content by clicking it   
⑥.Build project, when a project satisfies the build, click this button to compile   
⑦.Current download mode, the default is UART   
⑧.flash button, when a project build is completed, select the COM port of the corresponding development board, and click this button to flash the compiled firmware to the chip   
⑨.monitor enable flashing port monitoring, when a project passes through Build --> Flash, click this button to view the log of output from flashing port and debugging port, so as to observe whether the application works normally   
⑩.Debug   
⑪.Build Flash Monitor one-click button, which is used to continuously execute Build --> Flash --> Monitor, often referred to as "little flame"

### Compile, Flash and Serial Port Monitor

* Click on the all-in-one button we described before to compile, flash and open the serial port monitor
* It may take a long time to compile especially for the first time
* During this process, the ESP-IDF may take up a lot of CPU resources, so it may cause the system to lag
* If it is the first time to flash the program for a new project, you will need to select the download method, and select UART
* This can also be changed later in the Download methods section (click on it to pop up the options)
* As it comes with the onboard automatic download circuit, it can be downloaded automatically without manual operation
* After successful download, it will automatically enter the serial monitor, you can see the chip output the corresponding information and be prompted to restart after 10S

### Use the IDF Demos

The following takes **ESP32-S3-LCD-1.47-Demo** as an example to introduce the two opening methods of the project and the general steps of use, and the detailed explanation of the ESP-IDF project. If you use other projects, the operation steps can be applied similarly.

### Open In the Software

* Open VScode software and select the folder to open the demo
* Select the provided ESP-IDF example and click to select the file (located in the /Demo/ESP-IDF path under demo)

#### Open from Outside the Software

* Select the project directory correctly and open the project, otherwise it will affect the compilation and flashing of subsequent programs
* After connecting the device, select the COM port and model, click below to compile and flash to achieve program control

#### ESP-IDF Project Details

* Component: The components in ESP-IDF are the basic modules for building applications, each component is usually a relatively independent code base or library, which can implement specific functions or services, and can be reused by applications or other components, similar to the definition of libraries in Python development.
  + Component reference: The import of libraries in the Python development environment only requires to "import library name or path", while ESP-IDF is based on the C language, and the importing of libraries is configured and defined through CMakeLists.txt.
  + The purpose of CmakeLists.txt: When compiling ESP-IDF, the build tool CMake first reads the content of the top-level CMakeLists.txt in the project directory to read the build rules and identify the content to be compiled. When the required components and demos are imported into the CMakeLists.txt, the compilation tool CMake will import each content that needs to be compiled according to the index. The compilation process is as follows:

## Demo

|  |  |
| --- | --- |
| * ESP32-S3-Touch-LCD-4.3 Demo | |
| **Demo** | **Basic Description** |
| 01\_I2C\_Test | Test I2C header |
| 02\_RS485\_Test | Test RS485 header |
| 03\_SD\_Test | Test TF card slot |
| 04\_Sensor\_AD | Test ADC header |
| 05\_UART\_Test | Test UART |
| 06\_TWAItransmit | Test CAN seat |
| 07\_TWAIreceive | Test CAN seat |
| 08\_lvgl\_Porting | Test UART porting |

* Dependency libraries are automatically downloaded at compile time via IDF component.yml
  + Refer to [IDF Component Manager](https://docs.espressif.com/projects/esp-idf/en/latest/esp32s3/api-guides/tools/idf-component-manager.html) for more learning links

### 01\_I2C\_Test

#### Hardware connection

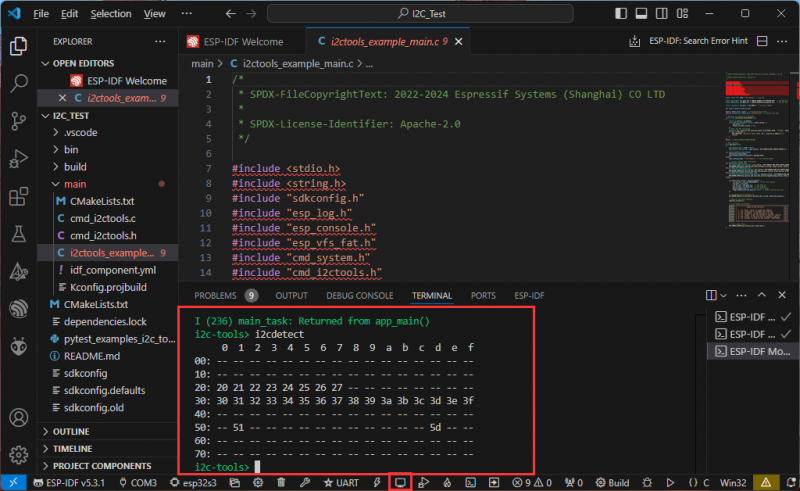
* Connect the board to the computer using a USB cable

#### Code analysis

* **app\_main()**:
  + Firstly, constants and variables related to I2C are defined, such as log tags, the SDA and SCL pins for I2C, and port numbers.
  + Next, install the console REPL environment for user interaction based on different configuration options. Then configure the I2C bus, including the clock source, ports, pins, and enable internal pull-up resistors, and initialize the I2C master bus.
  + A series of I2C utility commands are then registered, such as commands for device detection, register reads and writes, and so on. Instructions for use are also printed to instruct the user on how to use these commands.
  + Finally, start the console REPL, which allows users to interact with the application through the command line and perform various I2C operations, providing users with a convenient way to operate the I2C bus through the command line.

#### Result demonstration

* After the flashing is successful, open the serial terminal, enter the command, and press enter to run:

[](https://www.waveshare.com/wiki/File:800px-ESP32-IDF-14.png)

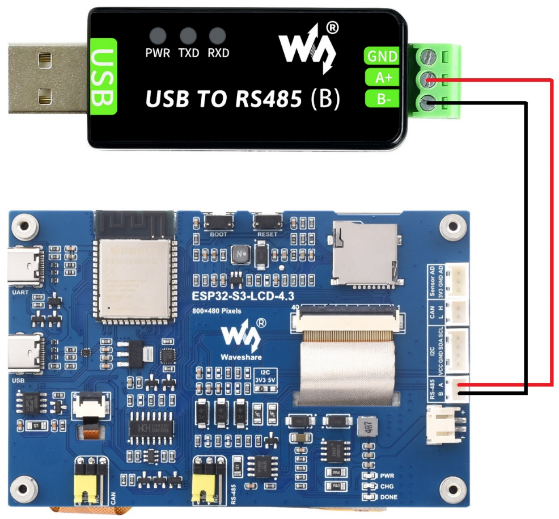
* The steps are as follows:

1. Use "help" to check all supported commands
2. Use "i2cconfig" to configure your I2C bus
3. Use "i2cdetect" to scan the devices on the bus
4. Use "i2cget" to retrieve the content of a specific register
5. Use "i2cset" to set the value of a specific register
6. Use "i2cdump" to dump all registers (experiment)

### 02\_RS485\_Test

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the development board to [USB to RS485 converter](https://www.waveshare.com/USB-TO-RS485-B.htm), as shown in the figure

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-43.png)

#### Code analysis

* **echo\_task()**:
  + Firstly, UART parameters were configured, including baud rate, data bits, parity bits, stop bits, and hardware flow control, etc.
  + Then install the UART driver, set the UART pins, and allocate a temporary buffer for receiving data.
  + In an infinite loop, data is read from the UART, the read data is written back to the UART, and log information is recorded when data is received.

#### Result demonstration

* Open the serial port debugging assistant to send a message to the ESP32-S3-Touch-LCD-4.3 device, and the device will return the received message to the serial port debugging assistant

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-32.png)

### 03\_SD\_Test

#### Hardware connection

* Connect the board to the computer using a USB cable

#### Code analysis

* **waveshare\_sd\_card\_init()**:

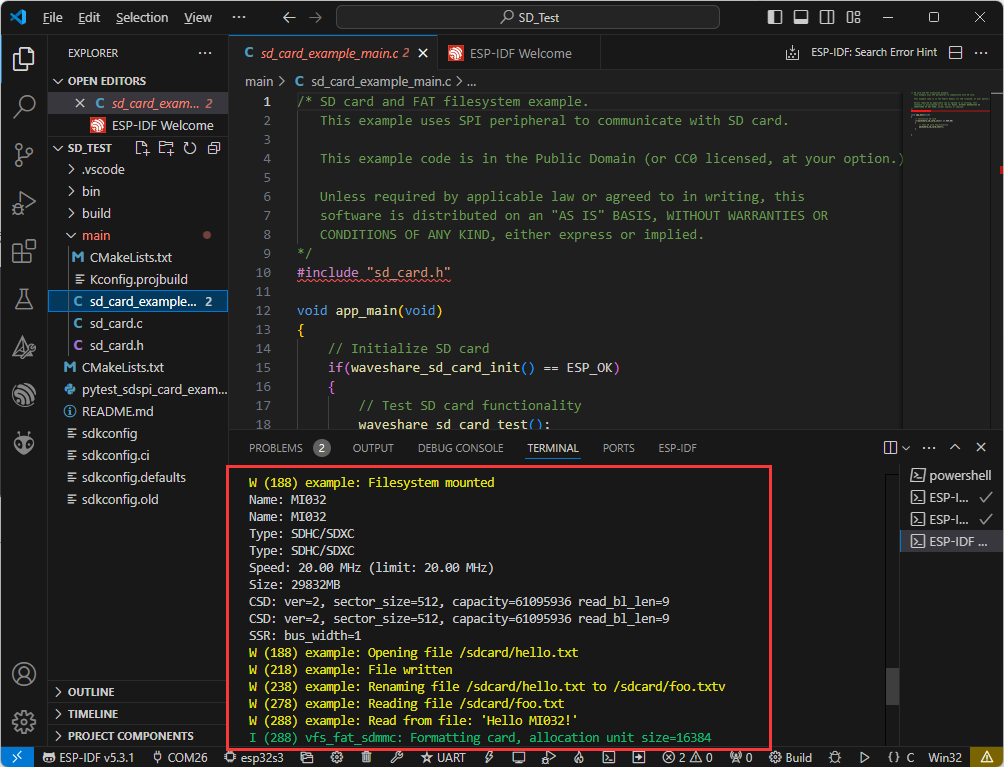
This function is mainly used to initialize the TF card. First initialize I2C, pull down the CS pin of the TF card through I2C control chip. Next, configure the TF card mounting options, including whether to format when mounting fails, the maximum number of files, and the allocation unit size, etc. After that, initialize the SPI bus and mount the TF card file system using the configured SPI bus and mount options. If mounting is successful, return ESP\_OK, indicating that the TYF card initialization is complete.

* **waveshare\_sd\_card\_test()**:

This function is used to test the functionality of the TF card. First print the information of the initialized TF card. Then create a file, write data into it, rename the file, and read the content of the renamed file. Next, format the file system and check if the file has been deleted after formatting. Finally, create a new file and read its content, unmount the TF card and free up SPI bus resources when the test is complete.

#### Result demonstration

* After successful programming, the serial port will print information about the storage card, such as its name, type, capacity, and maximum supported frequency. Then, it will create a file, write to the file, rename the file, and read the renamed file:

[](https://www.waveshare.com/wiki/File:ESP32-IDF-18.png)

### 04\_Sensor\_AD\_Test

#### Hardware connection

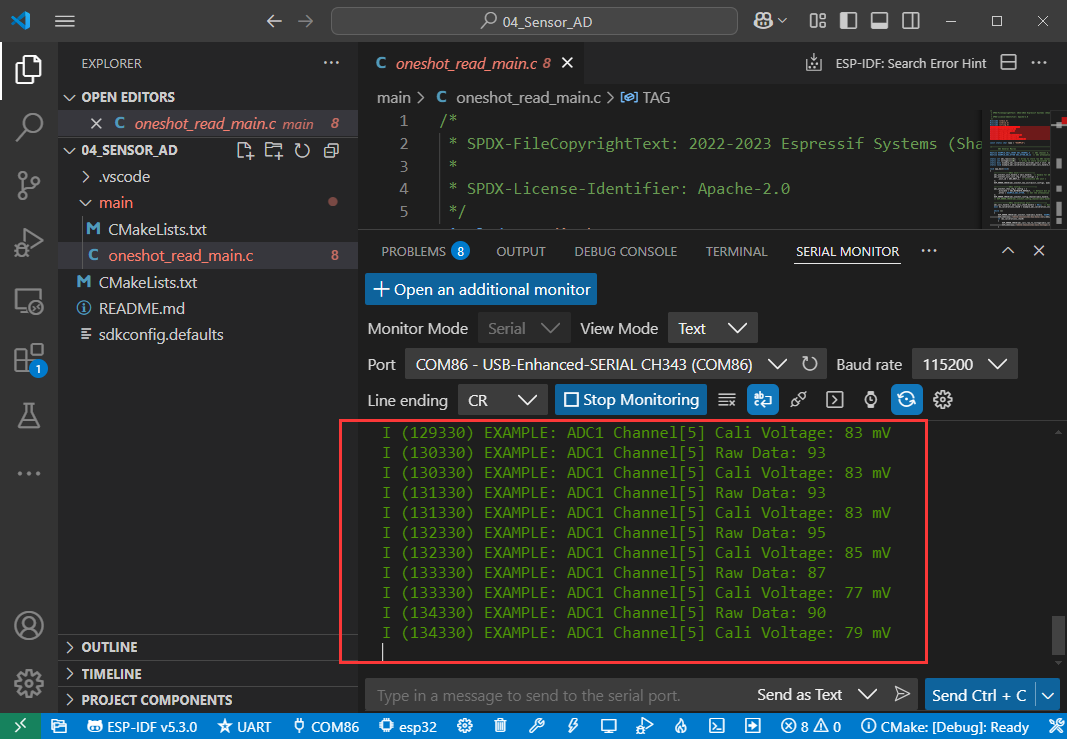
* Connect the board to the computer using a USB cable
* Connect the PH2.0 to 2.54mm male connector to the Sensor AD interface of the board

#### Code analysis

* **app\_main()**:
  + First, some variables used to store the current ADC are defined, and the calibration function is declared.
  + Next initialize the ADC, set its resolution and attenuation, and then create the ADC. In an infinite loop, there is also a 1-second delay in the print loop after the current ADC value is read.

#### Result demonstration

* Result demonstration: ESP32-S3-Touch-LCD-4.3 will set the ADC resolution, read the current AD value, and print to the serial port terminal

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-46.png)

### 05\_UART\_Test

#### Hardware connection

* Connect the UART port of the board to the computer using a USB cable

#### Code analysis

* **echo\_task()**:
  + Firstly, UART parameters were configured, including baud rate, data bits, parity bits, stop bits, and hardware flow control, etc.
  + Then install the UART driver, set the UART pins, and allocate a temporary buffer for receiving data.
  + In an infinite loop, data is read from the UART, the read data is written back to the UART, and log information is recorded when data is received.

#### Result demonstration

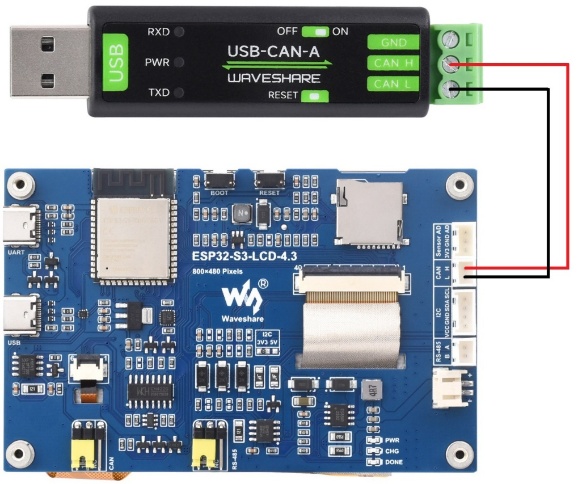
* Open the serial port debugging assistant to send a message to the ESP32-S3-Touch-LCD-4.3 device, and the device will return the received message to the serial port debugging assistant

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-32.png)

### 06\_TWAItransmit

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the development board to [USB-CAN-A](https://www.waveshare.com/USB-CAN-A.htm), as shown in the figure

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-44.png)

#### Code analysis

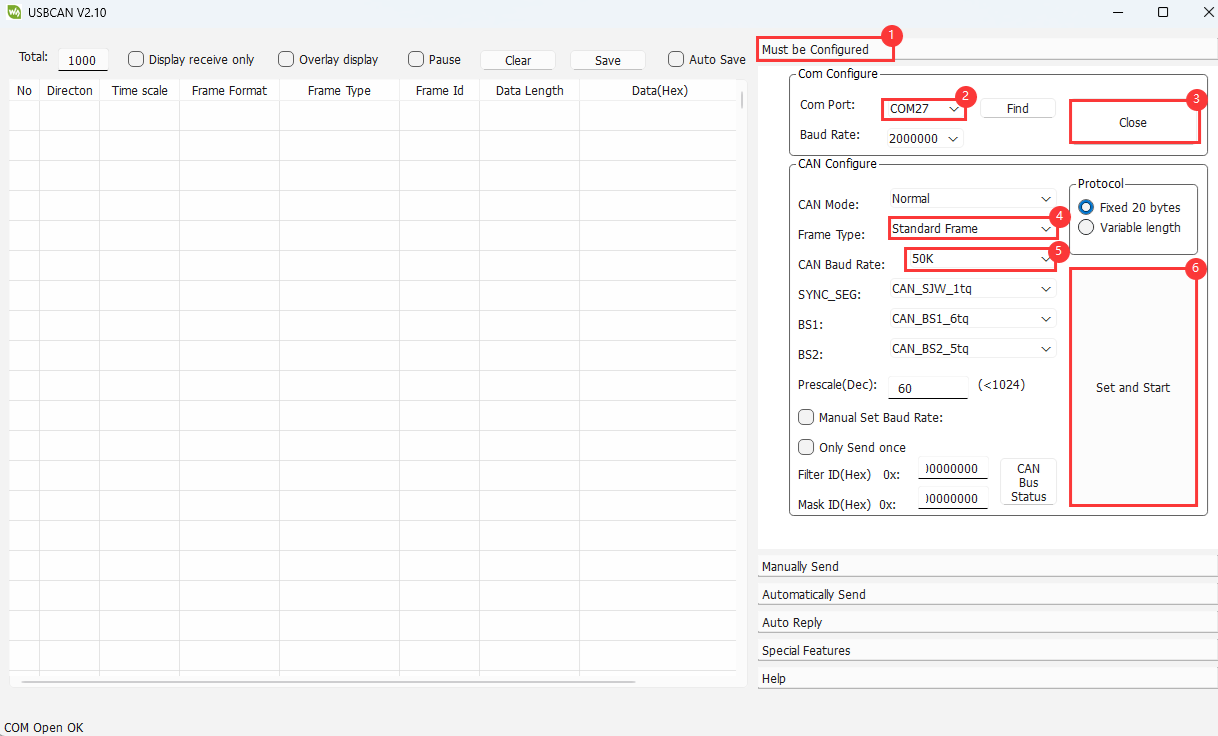
* **waveshare\_twai\_transmit()**:
  + If the driver is not installed, it will return to the failed state after waiting for a period of time
  + Read triggered alerts and get TWAI status information
  + Print the corresponding log information according to different alarm types, including passive error alarms, bus error alarms, transmission failure alarms, and transmission success alarms, and print relevant status information
  + Determine whether it is time to send a message, if so, send a message and update the last time it was sent

#### Result demonstration

* Serial port print indicates successful CAN message transmission. After configuring [USB-CAN-A\_TOOL](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#Resources), you can observe the CAN messages sent by the ESP32-S3-Touch-LCD-4.3 upon startup.

[](https://www.waveshare.com/wiki/File:ESP32-IDF-20.png)

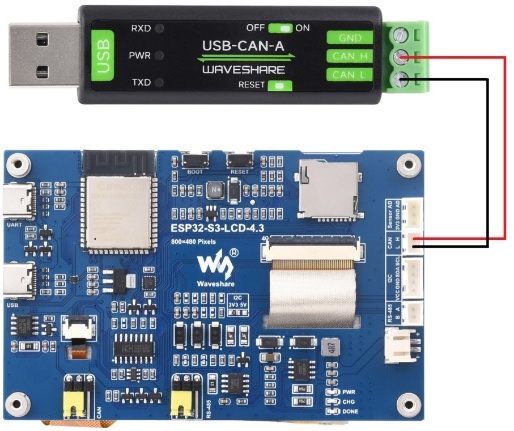
* Observe USB-CAN-A\_TOOL further, and you will see the CAN messages sent by ESP32-S3-Touch-LCD-4.3

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-CN-1.png)

### 07\_TWAIreceive

#### Hardware connection

* Connect the board to the computer using a USB cable
* Connect the development board to [USB-CAN-A](https://www.waveshare.com/USB-CAN-A.htm), as shown in the figure

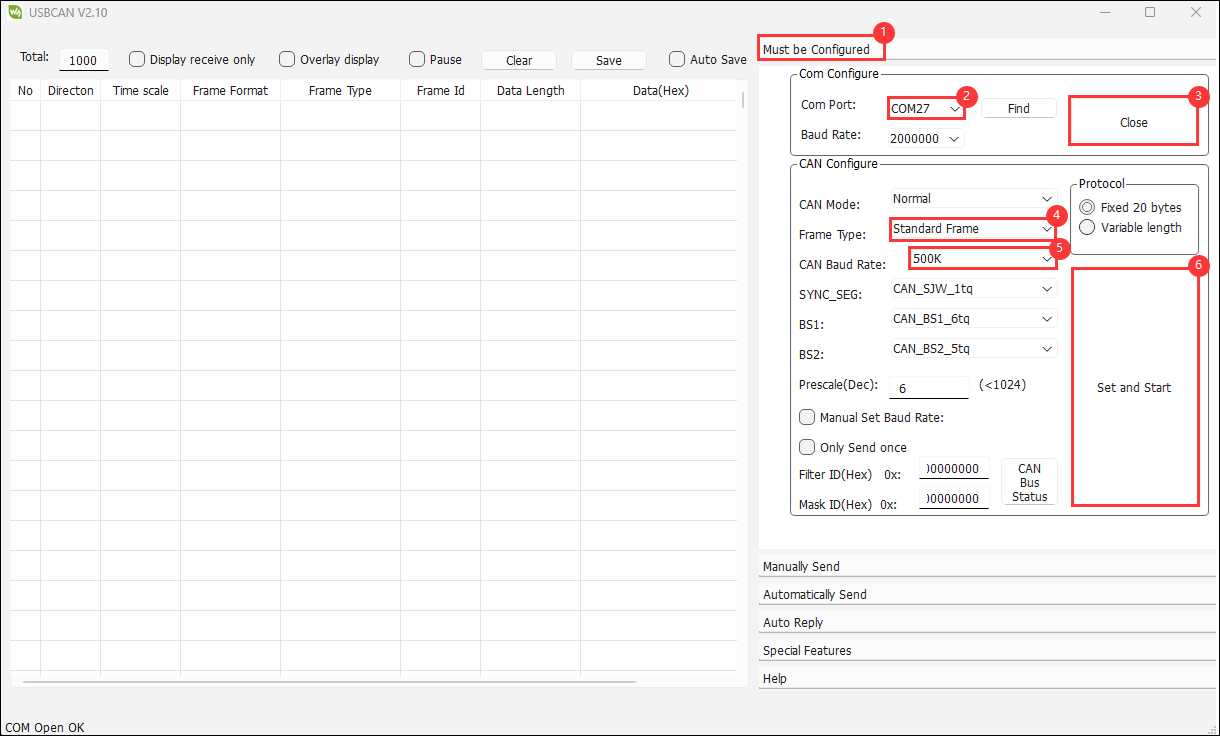
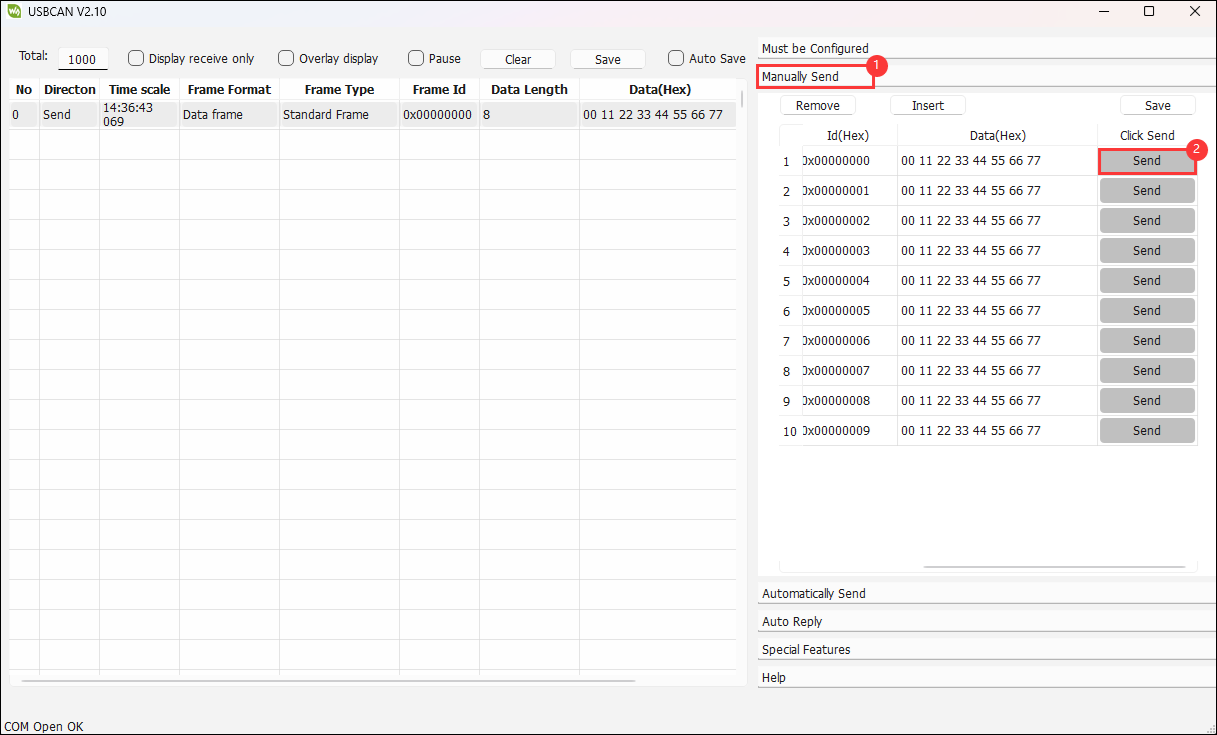
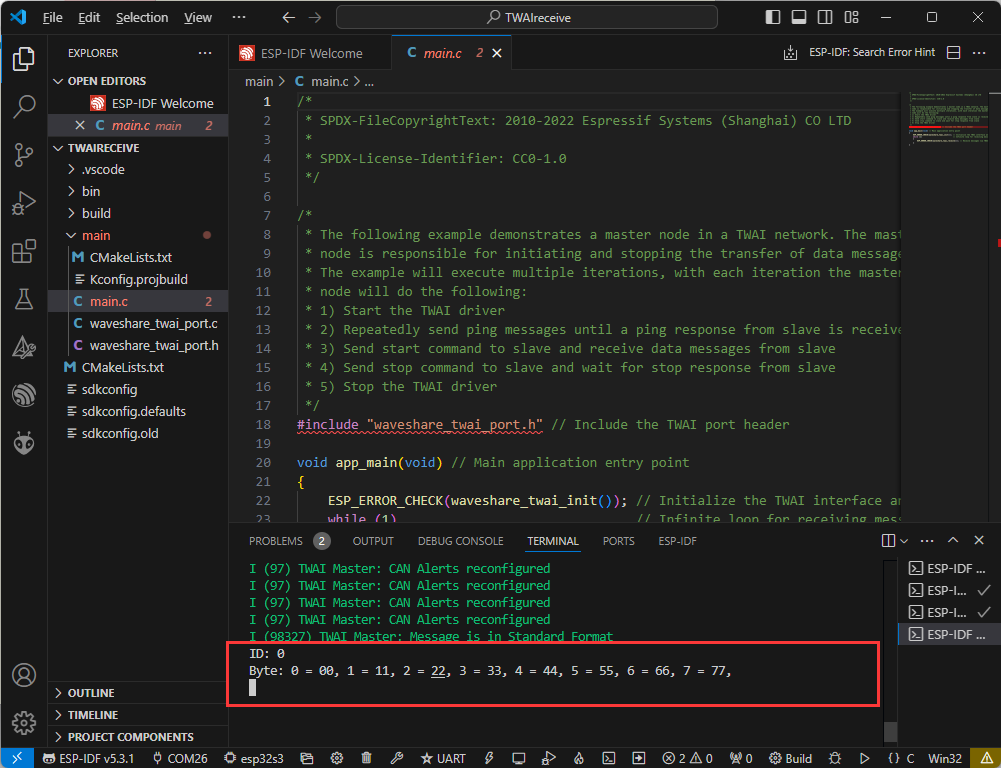
[](https://www.waveshare.com/wiki/File:ESP32-Arduino-44.png)

#### Code analysis

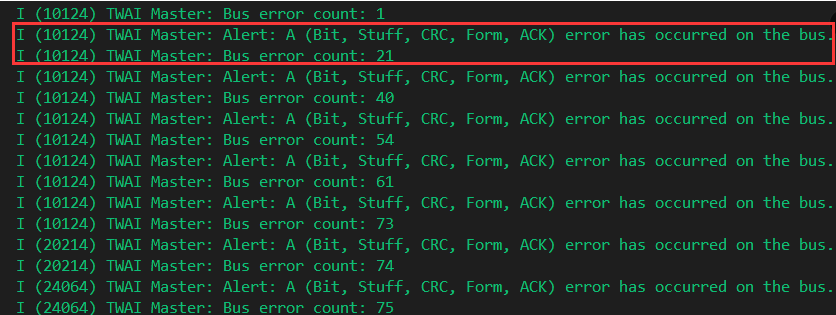
* **waveshare\_twai\_receive()**:
  + If the driver is not installed, it will return to the failed state after waiting for a period of time
  + Read triggered alerts and get TWAI status information
  + Print corresponding log information based on different alarm types triggered, including error passive alarms, bus error alarms, and receive queue overflow alarms, and print related status information
  + If a received data alert is triggered, the messages are received in a loop and the handle\_rx\_message function is called to process each received message. Finally, the success status is returned

#### Result demonstration

* ESP32-S3-Touch-LCD-4.3 waits [USB-CAN-A\_TOOL](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#Resources) to send a message. If the message is received successfully, it will be printed to the serial port

[](https://www.waveshare.com/wiki/File:ESP32-Arduino-CN-2.png)[](https://www.waveshare.com/wiki/File:ESP32-Arduino-23.png)[](https://www.waveshare.com/wiki/File:ESP32-IDF-21.png)

* If the following error occurs, click on the serial monitor and use the debugging tool to resend the data. (If you press Reset, sometimes you need to click the serial monitor again):

[](https://www.waveshare.com/wiki/File:ESP32-IDF-22.png)

### 08\_lvgl\_Porting

#### Hardware connection

* Connect the board to the computer using a USB cable

#### Code analysis

* **app\_main()**:
  + Initialize the Waveshare ESP32-S3 RGB LCD, and then you can choose to turn the screen backlight on or off.
  + Then print a message indicating that you want to display the demonstration content of LVGL. Since the LVGL API is not thread-safe, the mutex is locked first.
  + You can then choose to run different LVGL demos such as lv\_demo\_stress, lv\_demo\_benchmark, lv\_demo\_music, lv\_demo\_widgets, or example\_lvgl\_demo\_ui etc.
  + Finally release the mutual exclusion lock.

#### Code modification

* In ESP\_Panel\_Board\_Custom.h there is a macro definition for selecting whether to turn on the touch function, with a value of 0 corresponds to turning off touch, and a value of 1 corresponds to turning on touch, which can be selected according to the model purchased

#define CONFIG\_EXAMPLE\_LCD\_TOUCH\_CONTROLLER\_GT911 0 // 1 initiates the touch, 0 closes the touch.

Result demonstration

* After successful flashing, press the reset button to run the demo

[**Screen lights up\_video**](https://www.youtube.com/watch?v=luICguq9cjo)

* If you want to further increase the frame rate, you can refer to this [link](https://docs.espressif.com/projects/esp-faq/en/latest/software-framework/peripherals/lcd.html#how-can-i-improve-the-display-frame-rate-of-lcd-screens) for configuration.
* For RGB LCD drivers, please refer to this [link](https://docs.espressif.com/projects/esp-idf/en/release-v5.3/esp32s3/api-reference/peripherals/lcd/rgb_lcd.html)
* For GT911 drivers, you can refer to this [link](https://github.com/espressif/esp-bsp/tree/master/components/lcd_touch/esp_lcd_touch_gt911)
* The version of lvgl used is 8.3, and you can query and use the LVGL API through the following documentation
  + [LVGL documentation](https://docs.lvgl.io/8.3/)

# Flash Firmware Flashing and Erasing

* **The current demo provides test firmware, which can be used to test whether the onboard device functions properly by directly flashing the test firmware**
* **bin file path:**

...\ESP32-S3-Touch-LCD-4.3-Demo\Firmware

[Flash firmware flashing and erasing](https://www.waveshare.com/wiki/Flash_Firmware_Flashing_and_Erasing) for reference

## Resources

### Schematic Diagram

* [Schematic](https://files.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3/manual/ESP32-S3-Touch-LCD-4.3-Sch.pdf)

### Project Diagram

* [3D Drawing](https://files.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3/ESP32-S3-Touch-LCD-4in3_3D_Drawing.zip)

### Demo

* [ESP32-S3-Touch-LCD-4.3 Demo](https://files.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3/ESP32-S3-Touch-LCD-4.3-Demo.zip)

## Datasheets

### ESP32-S3

* [ESP32-S3 Datasheet](https://files.waveshare.com/wiki/common/Esp32-s3_datasheet_en.pdf)
* [ESP32-S3 Technical reference manual](https://files.waveshare.com/wiki/common/Esp32-s3_technical_reference_manual_en.pdf)
* [ESP32-S3-WROOM-1 datasheet](https://files.waveshare.com/wiki/common/Esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf)

### Other Components

* [ST7262 Datasheet](https://files.waveshare.com/wiki/common/ST7262.pdf)
* [GT911 Datasheet](https://files.waveshare.com/wiki/common/GT911_EN_Datasheet.pdf)
* [CH343 Datasheet](https://files.waveshare.com/wiki/common/CH343DS1-en.pdf)
* [TJA1051 Datasheet](https://files.waveshare.com/wiki/common/TJA1051.pdf)
* [CH422G Datasheet](https://files.waveshare.com/wiki/common/CH422DS1_EN.pdf)

## Software Tools

### Arduino

* [Arduino IDE Official download link](https://www.arduino.cc/en/software/)

### VScode

* [VScode official website](https://code.visualstudio.com/download)

### Debugging tool

* [SSCOM Serial port and network debugging assistant](https://files.waveshare.com/wiki/common/Sscom5.13.1.zip)
* [USB-CAN-A\_TOOL](https://files.waveshare.com/wiki/USB-CAN-A/Tool/USBCANV2.10.zip)

## Other Resource Links

* [ESP32-Arduino official documentation](https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html)
* [ESP32-Arduino official resources](https://github.com/espressif/arduino-esp32)
* [ESP-IDF official resources](https://github.com/espressif/esp-idf)
* [LVGL official documentation](https://docs.lvgl.io/master/intro/introduction.html)

## FAQ

[Question: Why did ESP32-S3-Touch-LCD-4.3 CAN reception fail?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion1)

**Answer:**

* Restart the COM port in the UCANV2.0.exe and press the Reset button of ESP32-S3-Touch-LCD-4.3 several times
* Remove the check for "DTR" on the serial port debugging assistant

[Question: ESP32-S3-Touch-LCD-4.3 shows no response after flashing an Arduino demo for RGB screen displaying?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion2)

**Answer:**

If there's no screen response after programming the code, check whether the correct configurations are set in Arduino IDE -> Tools, select the corresponding Flash (16MB) and enable PSRAM (8MB OPI))

[Question: ESP32-S3-Touch-LCD-4.3 flashing Arduino's RGB screen program fails to compile and reports error?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion3)

**Answer:**

To check if the library is installed, refer to [library installation procedure](https://www.waveshare.com/wiki/Arduino_Library_Manager_Tutorial)

[Question: Why are all libraries installed, and the flashing LVGL program reports an error missing lv\_cong.h?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion4)

**Answer:**

The path to install the library contains Chinese characters, resulting in the inability to find the library file

[Question: Why does the screen not display after successful flashing with Arduino?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion5)

**Answer:**

You can refer to the following steps to run the demo for comparison:

1. Before running the program, refer to [Arduino library manager tutorial](https://www.waveshare.com/wiki/Arduino_Library_Manager_Tutorial) for library installation
2. To install library, please refer to [video reference](https://files.waveshare.com/wiki/common/ESP32-S3-7-lib.zip)
3. Run and flash [Demo](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#Resources)

[Question: Why does error "fatal error:esp\_ memory\_ utils.h:No such file or directory" occur when compiling demos using the Arduino IDE?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion6)

**Answer:**

Please install Arduino esp32 version ≥v3.0.2, this can solve the issue

[Question:Can CAN communication use debugging tools to send continuous frames?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion7)

**Answer:**

Yes, you can also customize the frequency of consecutive frames. When the frequency is too high and the computer freezes, it may cause bus errors

[Question: What should I do if ESP-IDF flashing fails?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion8)

**Answer:**

1. You can set the board to download mode to solve this issue. Power off completely, press and hold the Boot button and power it on again, then release it, enter the download mode, re-flash the program, reset and run.
2. Please try to press the fullclean button in the status bar and recompile the flashing, this function is to clean up all the compiled content by clicking when the project compilation error or other operations pollute the compiled content

[Question: What should I do if I can't find the AppData folder?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion9)

**Answer:**

Some AppData folders are hidden by default and can be set to show

* + English system: Explorer->View->Check "Hidden items"
  + Chinese system: File Explorer -> View -> Display -> Check "Hidden Items"

[Question: How to check the COM port?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion10)

**Answer:**

Press the Windows + R keys to open the "Run" dialog box, input devmgmt.msc and press Enter to open the Device Manager; Expand the "Port (COM and LPT)" section, here it will list all the COM ports and their current status.

[Question: How to deal with the first compilation of the program being extremely slow?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion11)

**Answer:**

It's normal for the first compilation to be slow, just be patient

[Question: How to solve the problem that the program is flashed successfully but there is no display on LCD?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion12)

**Answer:**

If there is a reset button on the development board, press the reset button; if there is no reset button, please power it on again

[Question: Why does the program flashing fail when using a MAC device?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion13)

**Answer:**

Install [MAC Driver](https://files.waveshare.com/wiki/common/CH34XSER_MAC.7z) and flash again.

[Question: How to use SquareLine Studio to design interfaces?](https://www.waveshare.com/wiki/ESP32-S3-Touch-LCD-4.3#accordion14)

**Answer:**

* Please refer to [SquareLine Studio tutorial](https://www.waveshare.com/wiki/Waveshare_SquareLine_Studio)

## Support

### Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.   
Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)