

# ESP32-S3-Touch-LCD-4.3

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

## ESP32-S3-Touch-LCD-4.3



ESP32-S3-N16R8, Type C USB

## ESP32-S3-LCD-4.3



ESP32-S3-N16R8, Type C USB

## Overview

---

## 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



## 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](#) 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](#)
- 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](#) 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](#), select the corresponding system and system bit to download. The version of the Arduino IDE needs to be  $\geq 1.8$ , and the path of installation must not be Chinese, otherwise there will be an error when compiling.



**Arduino IDE 2.3.3**

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

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

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

**SOURCE CODE**

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

**DOWNLOAD OPTIONS**

- Windows** Win 10 and newer, 64 bits
- Windows** MSI installer
- Windows** ZIP file
- Linux** ApplImage 64 bits (X86-64)
- Linux** ZIP file 64 bits (X86-64)
- macOS** Intel, 10.15: "Catalina" or newer, 64 bits
- macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

- 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](#) 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](#)
- 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](#)

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

ESP32-S3-Touch-LCD-4.3 Development board installation instructions

## 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](#)
- 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](#)

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"

ESP32-S3-Touch-LCD-4.3 library file installation instructions

## 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!

## Demo

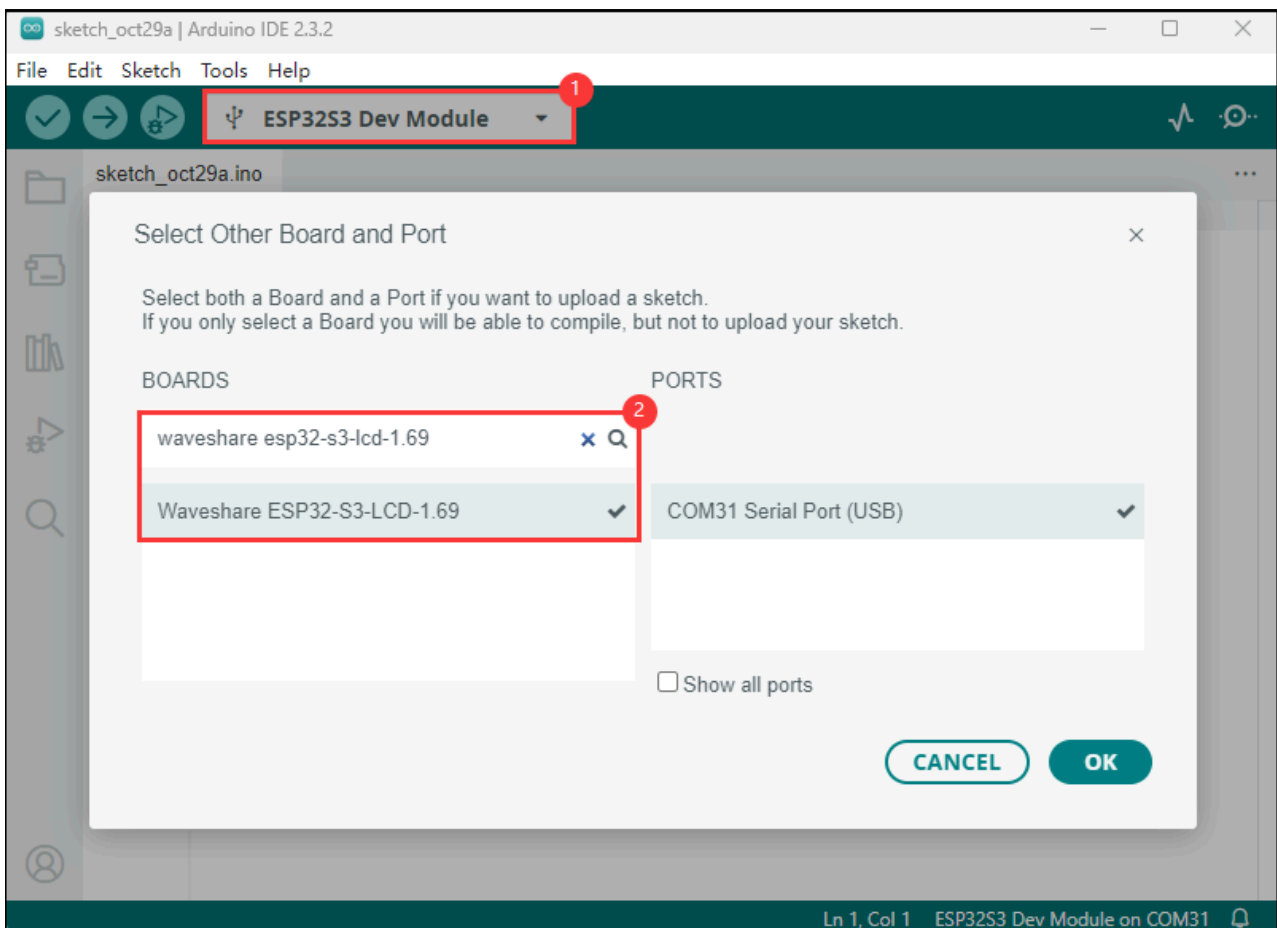
Demo	Basic Description	Dependency Library
01_I2C_Test	Test I2C header	-
02_RS485_Test	Test RS485 header	-
03_SD_Test	Test TF card slot	-
04_Sensor_AD	Test ADC header	-

05_UART_Test	Test UART	-
06_TWItransmit	Test CAN header	-
07_TWIreceive	Test CAN header	-
08_DrawColorBar	Test RGB screen	ESP32_Display_Panel
09_lvgl_Porting	Test LVGL porting	LVGL, ESP32_Display_Panel

### ESP32-S3-Touch-LCD-4.3 Demo

**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



## 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.

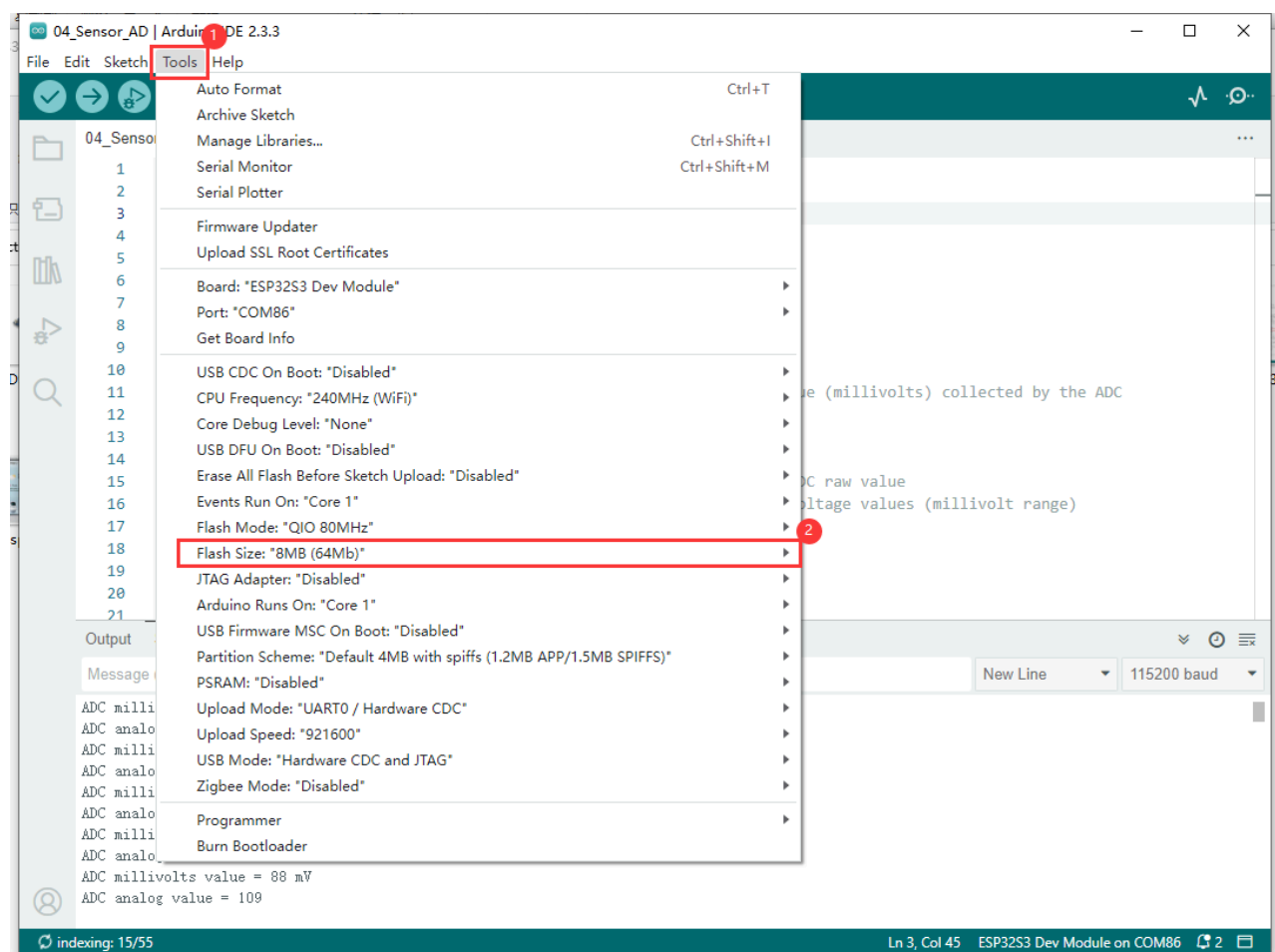
First, variables were defined to store error codes, device addresses, and to record the number of devices found.

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

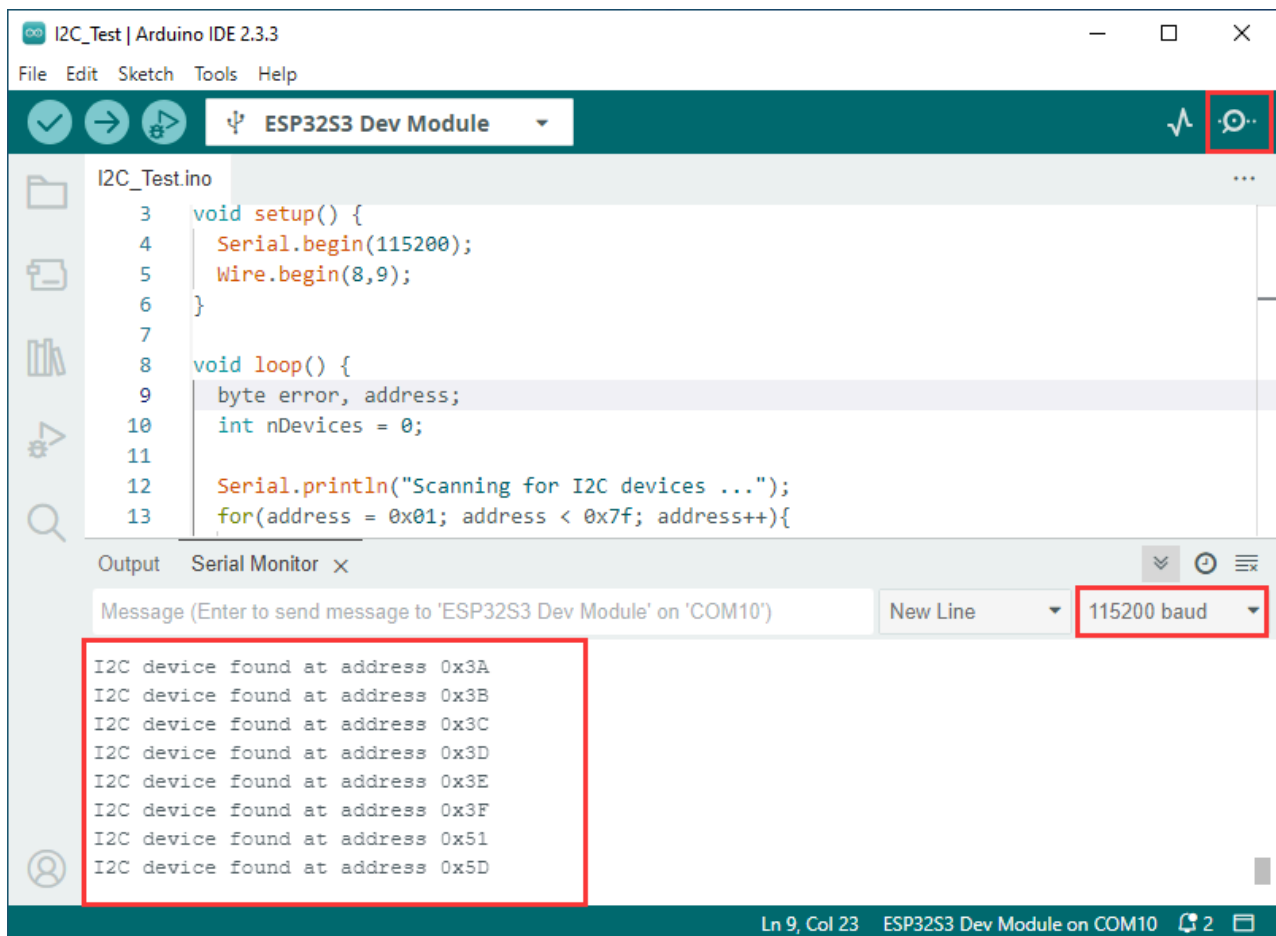


Flash the demo

### Result demonstration



The serial monitor prints the device address on the I2C bus



## 02\_RS485\_Test

### Hardware connection

- Connect the board to the computer using a USB cable
- Connect the development board to [USB to RS485 converter](#), as shown in the figure

 [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

 [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

---

 [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_IOWExpander_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

 [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.

---

 [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



---

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

 [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

 [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





---

## 06\_TWAltransmit

---

### Hardware connection

- Connect the board to the computer using a USB cable
- Connect the development board to [USB-CAN-A](#), as shown in the figure



---

## 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

 [ESP32-Arduino-41.png](#)

---

Flash the demo

### **Result demonstration**

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

 [ESP32-Arduino-20.png](#)





---

## 07\_TWAIreceive

---

### Hardware connection

- Connect the board to the computer using a USB cable
- Connect the development board to [USB-CAN-A](#), as shown in the figure



---

## 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

 [ESP32-Arduino-41.png](#)

---

Flash the demo

### **Result demonstration**

ESP32-S3-Touch-LCD-4.3 waits [USB-CAN-A\\_TOOL](#) to send a message. If the message is received successfully, it will be printed to the serial port









---

 [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




Flash the demo

## **Result demonstration**

Serial port prints log, and the screen lights up

 [ESP32-Arduino-24.png](#)

 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





---

Flash the demo

### **Result demonstration**

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

---


 [ESP32-Arduino-26.png](#)

---

[Screen lights up\\_video](#)

## Other instructions

- Screen drifting occurs during use, please refer to [ESP official FAQ](#)
- When using your own UI program, there is a lack of memory, you can click Tools to select a larger partition table

 [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](#)

## 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](#), choose the corresponding system and system bit to download

 [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

 [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](#) 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](#)

## 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!

### 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

ESP32-S3-Touch-LCD-4.3 Demo

Dependency libraries are automatically downloaded at compile time via IDF component.yml

Refer to [IDF Component Manager](#) 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:

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](#), as shown in the figure

---

 [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



---

## 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:

 [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

 [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

 [ESP32-Arduino-32.png](#)

---

## 06\_TWAltransmit

---

### Hardware connection

- Connect the board to the computer using a USB cable
- Connect the development board to [USB-CAN-A](#), as shown in the figure

 [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](#), you can observe the CAN messages sent by the ESP32-S3-Touch-LCD-4.3 upon startup.

 [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



---

## 07\_TWAIreceive

---

### Hardware connection

- Connect the board to the computer using a USB cable
- Connect the development board to [USB-CAN-A](#), as shown in the figure

 [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](#) to send a message. If the message is received successfully, it will be printed to the serial port









 [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):



---

## 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](#)

- If you want to further increase the frame rate, you can refer to this [link](#) for configuration.
- For RGB LCD drivers, please refer to this [link](#)
- For GT911 drivers, you can refer to this [link](#)
- The version of lvgl used is 8.3, and you can query and use the LVGL API through the following documentation

[LVGL documentation](#)

## 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](#) for reference



## Resources

---

### Schematic Diagram

---

[Schematic](#)

### Project Diagram

---

[3D Drawing](#)

### Demo

---

[ESP32-S3-Touch-LCD-4.3 Demo](#)

### Datasheets

---

### ESP32-S3

---

### Other Components

---

### Software Tools

---

#### Arduino

---

[Arduino IDE Official download link](#)

#### VScode

---

[VScode official website](#)

### Debugging tool

---

### Other Resource Links

---

### FAQ

---

#### 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

#### 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))

**Answer:**

To check if the library is installed, refer to [library installation procedure](#)

**Answer:**

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

**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](#) for library installation
2. To install library, please refer to [video reference](#)
3. Run and flash [Demo](#)

**Answer:**

Please install Arduino esp32 version  $\geq v3.0.2$ , this can solve the issue

**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

**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

**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"

**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.

**Answer:**

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

**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

**Answer:**

Install [MAC Driver](#) and flash again.

**Answer:**

Please refer to [SquareLine Studio tutorial](#)

## 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)

[Submit Now](#)