ES3C40P

4.0inch IPS Arduino

Demo Instructions

Contents

1. Software and hardware platform description

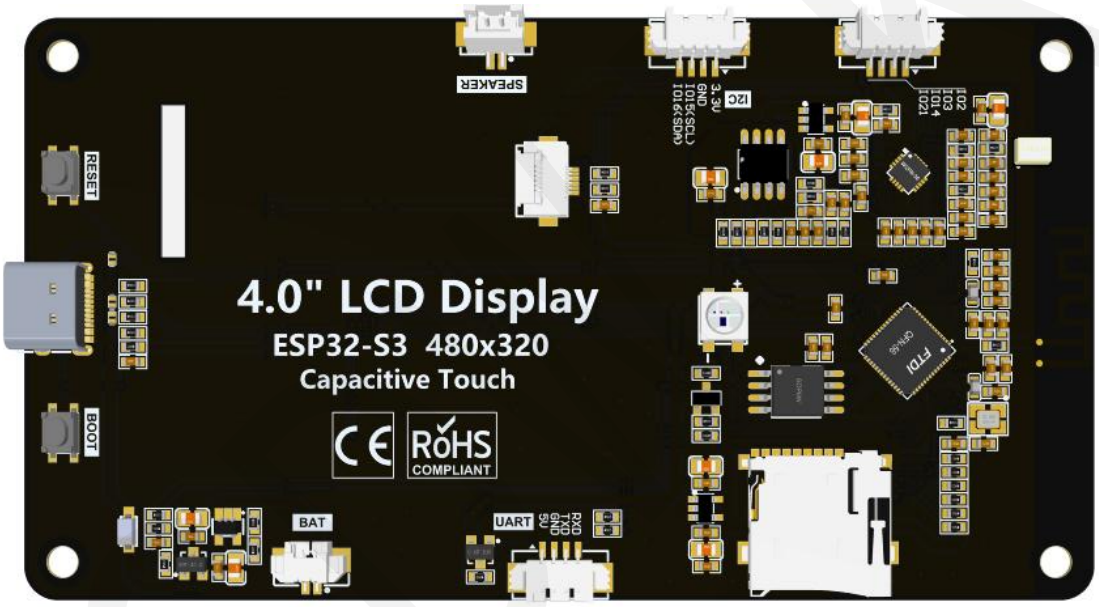
Module：4.0inch ESP32-S3 IPS LCD Display，Resolution of 480x320, with ST7796S screen driver IC

Main control chip：ESP32-S3 chip,Maximum clock frequency of 240MHz, supporting 2.4G WIFI+Bluetooth;

Arduino IED version：2.3.4

ESP32 Ardunio core library software versions：3.2.0

2. Pin assignment description



Picture 2.1 Back view of the 4.0 inch ESP32-S3 display module

The main control of the 4.0-inch ESP32-S3 display module is ESP32-S3, and its GPIO allocation for onboard peripherals is shown in the following table:

ESP32-S3 chip pin allocation instructions			
On board device	**On board device pins**	**ESP32-S3 connection pin**	**description**
LCD	**TFT_CS**	IO10	LCD screen chip selection control signal, low level effective
	TFT_RS	IO46	LCD screen command/data selection control signal.High level: data, low level: command

	TFT_SCK	IO12	LCD SPI bus clock signal
	TFT_MOSI	IO11	LCD SPI bus writes data signals
	TFT_MISO	IO13	LCD SPI bus reading data signal
	TFT_RST	CHIP_PU	LCD screen reset control signal, low level reset (shared reset pin with ESP32-S3 main control)
	TFT_BL	IO45	LCD screen backlight control signal (high level lights up the backlight, low level turns off the backlight)
CTP	**TP_SDA**	IO16	Capacitive touch screen I2C bus data signal
	TP_SCL	IO15	Capacitive touch screen I2C bus clock signal
	TP_RST	IO18	Capacitive touch screen reset control signal, low level reset
	TP_INT	IO17	Capacitive touch screen interrupts the input signal, and when a touch event occurs, it inputs a low level
LED	**RGB_INT**	IO42	Single line RGB three color LED light, which can light up the internal red, green, and blue light beads according to different signals
SDCARD	**SD_CLK**	IO38	SD card SDIO bus clock signal
	SD_CMD	IO40	SD card SDIO bus command signal
	SD_D0	IO39	SD card SDIO bus data signal (DATA0~DATA3 four data lines)
	SD_D1	IO41	
	SD_D2	IO48	
	SD_D3	IO47	
BATTERY	**BAT_ADC**	IO9	Battery voltage ADC value acquisition input signal
Audio	**Audio_EN**	IO1	Audio output enable signal, low-level enable, high-level disable
	I2S_MCK	IO4	Audio I2S bus master clock signal
	I2S_SCK	IO5	Audio I2S bus bit clock signal
	I2S_DO	IO6	Audio I2S bus bit output data signal
	I2S_LRC	IO7	Audio I2S bus left and right channel

			selection signals. High level: right channel; Low level: Left channel
	I2S_DI	IO8	Audio I2S bus bit input data signal
KEY	**BOOT_KEY**	IO0	Download mode selection button (press and hold the button to power on, then release it to enter download mode)
	RESET_KEY	EN	ESP32-23E reset button, low level reset (shared with LCD screen reset)
USB	**USB_N**	IO19	USB bus differential signal data line negative pole
	USB_P	IO20	Positive pole of USB bus differential signal data line
Serial Port	**TX0**	IO43	ESP32-S3 serial port 0 sending signal
	RX0	IO44	ESP32-S3 serial port 0 receiving signal
POWER	**TYPE-C_POWER**	/	Type-C power interface, connected to 5V voltage

Table 2.1 Pin allocation instructions for ESP32-S3 onboard peripherals

3. Example program description

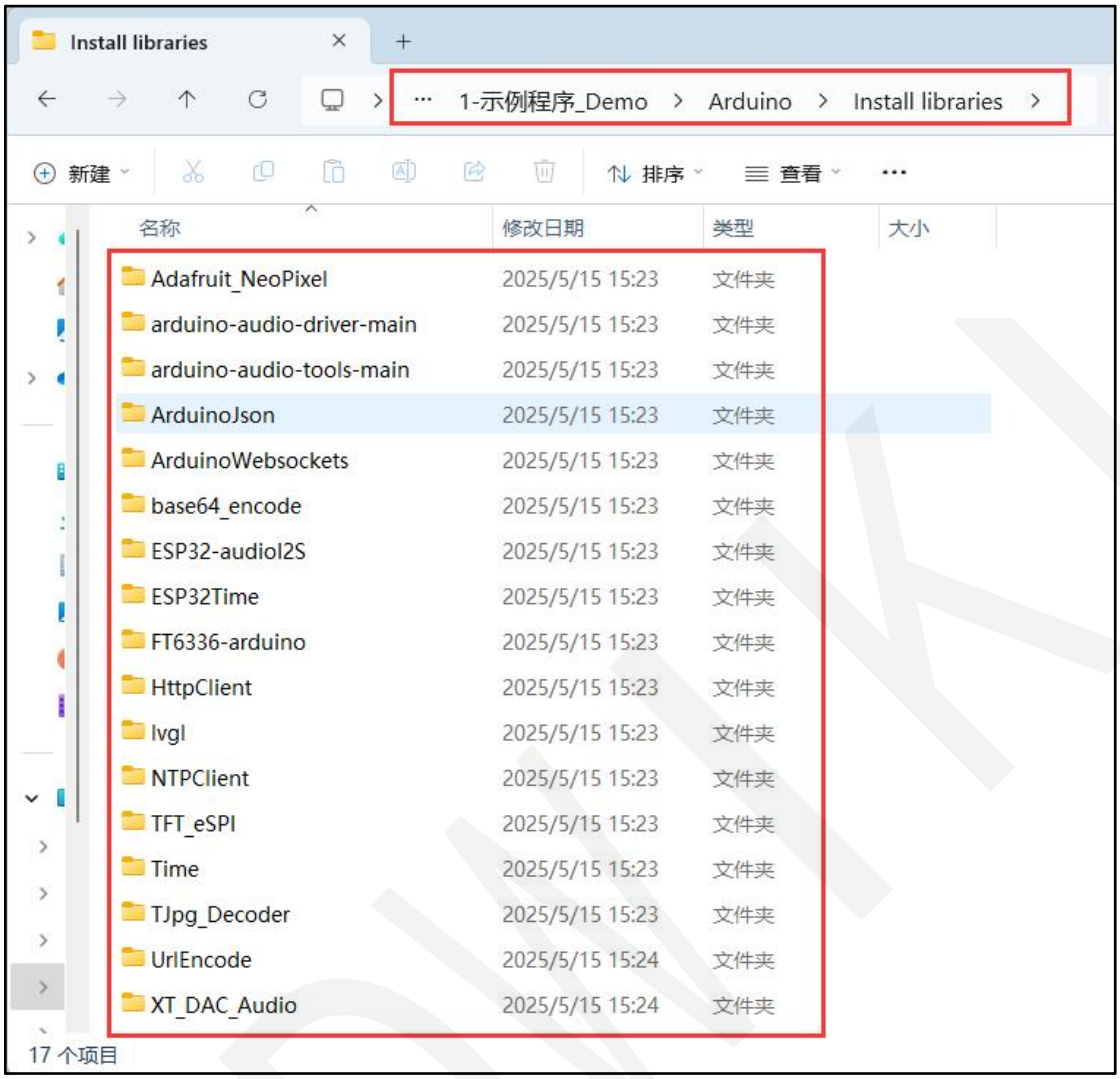
3.1. Set up ESP32 Arduino development environment

Detailed instructions for setting up the ESP32 Arduino development environment can be found in the "**Arduino_IDE_development_environment_construction_for_ESP32**" document the resource pack.

3.2. Install other software libraries

After setting up the development environment, you need to install the third-party libraryused by the sample program. The steps are as follows:

A. Open the "**1-示例程序_Demo \Arduino\Install libraries"** directory in the resource package, find the third-party library as shown in the figure below:



Picture 3.1 Example program third-party software library

Adafruit_Neopixel: Library to provide control functionality for RGB lights.

arduino-audio-driver: Audio codec driver library

arduino-audio-tools: Audio signal processing library

ArduinoJson：Library to provide IoT JSON functionality for Arduino.

ArduinoWebsockets: A library for writing modern websockets applications with Arduino

base64_encode：Library for base64 encoding of data

ESP32-audioI2S：Library to provide audio decoding function for ESP32, which can use the ESP32's I2S bus to play audio files in mp3, ma, and mav formats stored on the SD card through external audio devices.

ESP32Time：Arduino library for setting and retrieving the internal RTC time on ESP32 boards

FT6336-arduino：Provide a library for capacitive touch drive

HttpClient：Http client library for interacting with the Arduino web server

lvgl：Highly customizable, low-resource-occupancy, beautiful and easy-to-use embedded system graphics software library.

NTPClient：NTP client software library connecting to NTP server

TFT_eSPI：Arduino library for TFT-LCD displays, supporting a variety of platforms and multiple LCD driver IC

Time: Software library providing timing functions for Arduino.

TJpg_Decoder：Arduino platform JPG format picture decoding library, which can decode JPG files in SD card or Flash and then display them on LCD.

UrlEncode：This library can restore strings encoded by the urlencode function into their original information so that the client and server can correctly understand the transmitted data.。

XT_DAC_Audio：ESP32 XTronical DAC audio software library, which can support audio files in WAV format.

B. Copy these software libraries to the library directory under the project folder. The library directory of the project folder defaults to **"C:\Users\Administrator\Documents\Arduino\libraries"** (The red part is the a ctual username of the computer). If you change project folder path, you need to copy it into the library directory of the modified project folder.

C. The third-party software library in the above package has been configured and can be used directly. After installation, open the sample program to compile and run it, and you can see the effect.

If you want to use the latest version of the library, you can download it from GitHub or install it using the Arduino IDE library management tool. The latest version of the library requires configuration of lvgl and TFT_eSPI. The steps are as follows:

A. Download the latest libraries from GitHub or install them using the Arduino IDE library management tool. The GitHub download link is as follows:

lvgl: https://github.com/lvgl/lvgl/tree/release/v8.3(**Only V8. x version can be used, V9. x version cannot be used**)

TFT_eSPI: https://github.com/Bodmer/TFT_eSPI

Attached are the download links for other software packages that do not need to be configured:

ArduinoJson：https://github.com/bblanchon/ArduinoJson.git

ESP32Time：https://github.com/fbiego/ESP32Time

HttpClient：http://github.com/amcewen/HttpClient

NTPClient：https://github.com/arduino-libraries/NTPClient.git

Time：https://github.com/PaulStoffregen/Time

TJpg_Decoder: https://github.com/Bodmer/TJpg_Decoder

TFT_Touch : https://github.com/Bodmer/TFT_Touch

B、After the library download is complete, extract it (you can rename the extracted library folder for easy identification), and then copy it to the library directory under the project. The default is:"**C:\Users\Administrator\Documents\Arduino\libraries"**（The red part is the actual username of the computer）。 Next, perform library configuration. Open the "**1-示例程序_Demo \Arduino\Replaced files**" directory in the resource package, locate the file, as shown in the following figure:

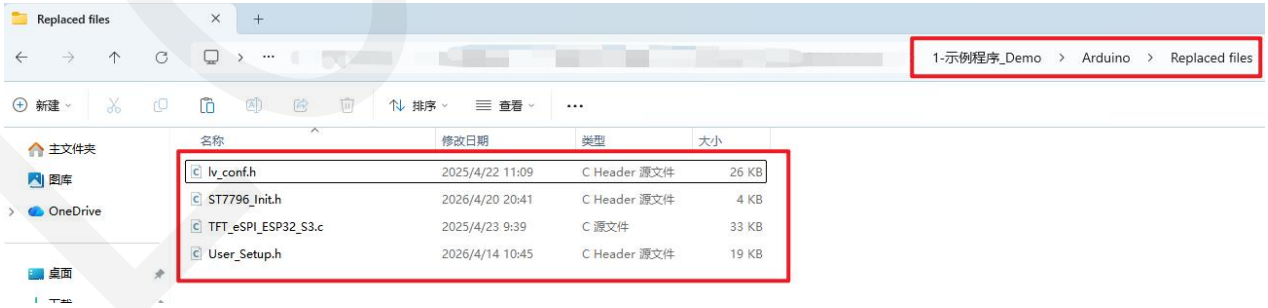


Figure 3.2 Third party software library replacement file

C. Configure LVGL library:

Copy the **lv_conf. h** file from the Replaced files directory to the top-level directory of the lvgl library in the project library directory, as shown in the

following figure:

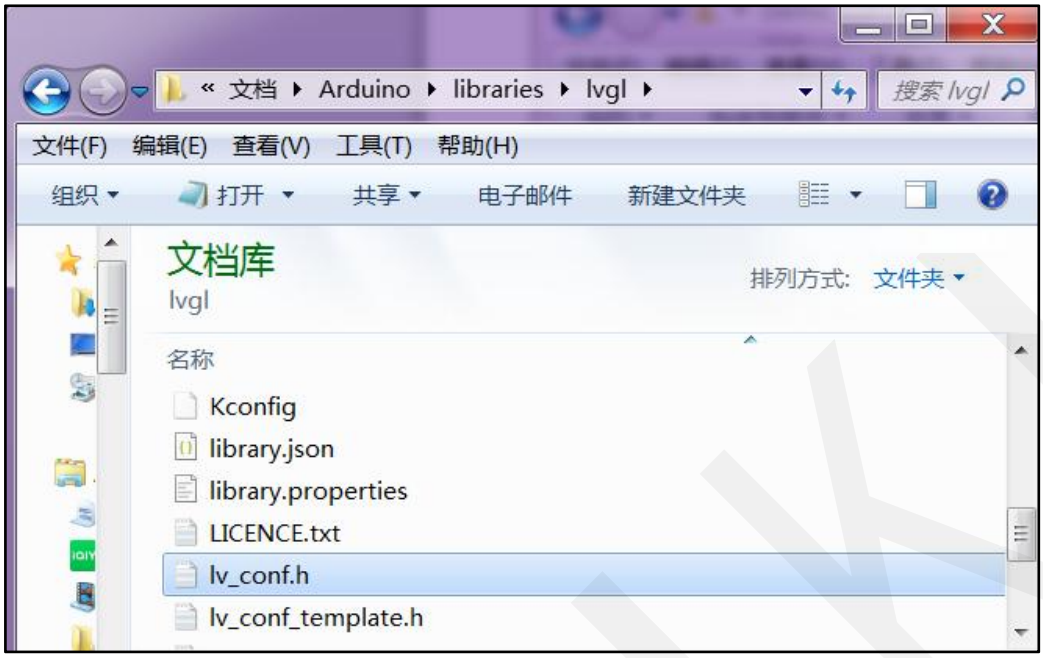


Figure 3.3 Configuring LVGL Library 1

Open the **lv_conf_internal. h** file in the **src** directory of the lvgl library under the engineering library directory, as shown in the following figure:

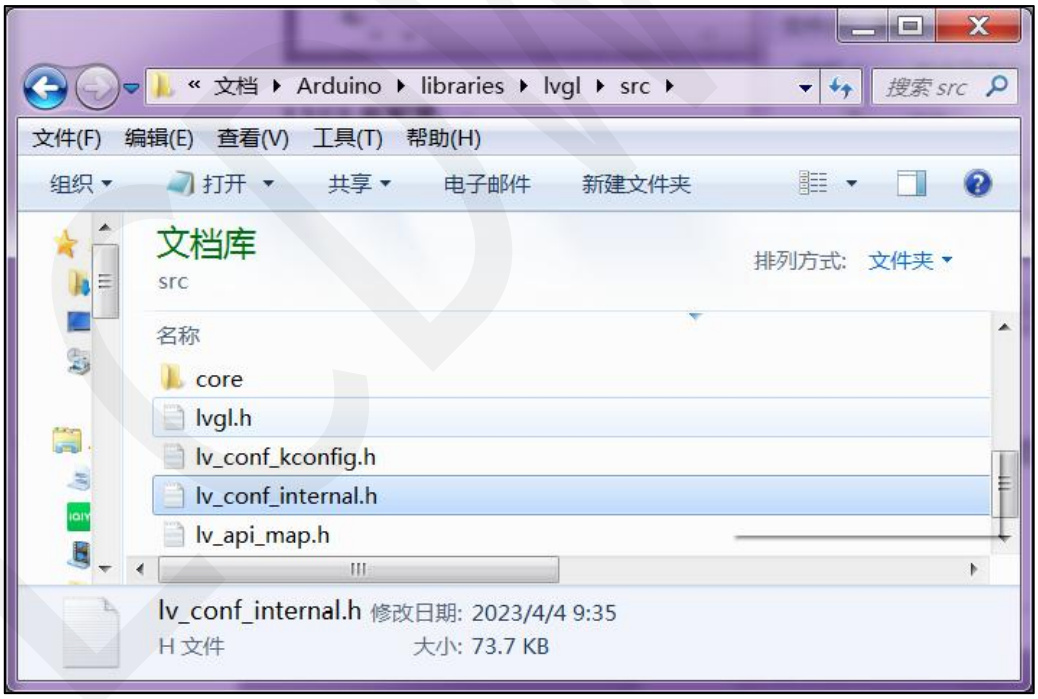


Figure 3.4 Configuring LVGL Library 2

After opening the file, modify the contents of line 41 as shown below (by "**.. /.. /lv_conf.h** Change the value to**.. /lv_conf.h** "), and save the modification.

```
/*If lv_conf.h is not skipped include it*/
#ifndef LV_CONF_SKIP
    #ifdef LV_CONF_PATH                         /*If there is a path defined for lv_conf.h (
        #define __LV_TO_STR_AUX(x) #x
        #define __LV_TO_STR(x) __LV_TO_STR_AUX(x)
        #include __LV_TO_STR(LV_CONF_PATH)
        #undef __LV_TO_STR_AUX
        #undef __LV_TO_STR
    #elif defined(LV_CONF_INCLUDE_SIMPLE)       /*Or simply include lv_conf.h is enabled*/
        #include "lv_conf.h"
    #else
        #include "../lv_conf.h"                 /*Else assume lv_conf.h is next to the lvgl fo
    #endif
    #if !defined(LV_CONF_H) && !defined(LV_CONF_SUPPRESS_DEFINE_CHECK)
        /* #include will sometimes silently fail when __has_include is used */
        /* https://gcc.gnu.org/bugzilla/show_bug.cgi?id=80753 */
        #pragma message("Possible failure to include lv_conf.h, please read the comment in th
    #endif
#endif
```

Figure 3.5 Configuring LVGL library 3

Copy **examples** and **demos** from lvgl in the project library to **src** in lvgl, as shown below:

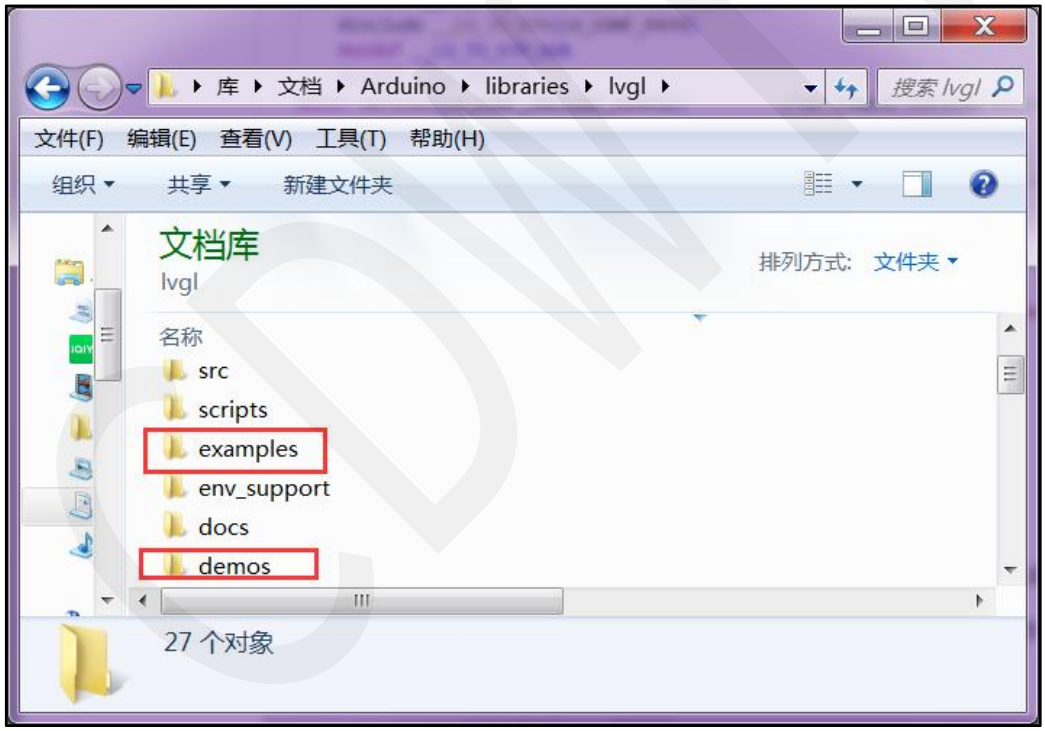


Figure 3.6 Configuring LVGL Library 4

Copy directory status:

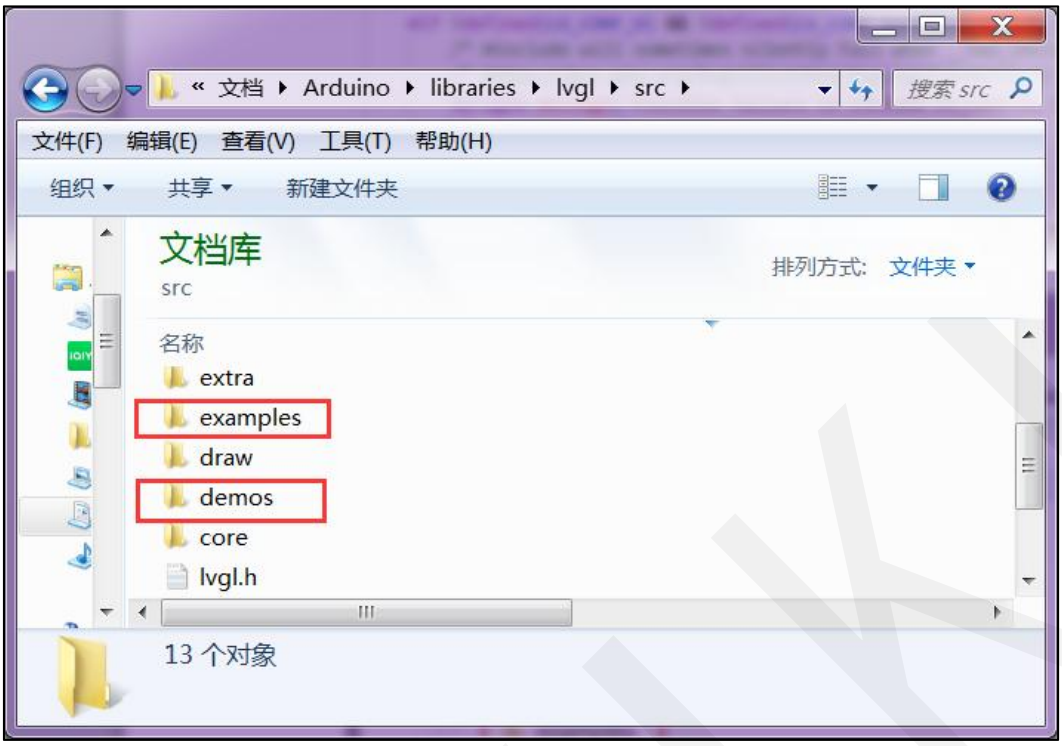


Figure 3.7 Configuring LVGL Library 5

D. Configure TFT_eSPI library:

Firstly, rename the **User_Setup. h** file in the top-level directory of the TFT_eSPI library under the project folder library directory to **User_Setup_bak. h**. Then, copy the **User_Setup. h** file from the Replaced files directory to the top-level directory of the TFT_eSPI library under the project library directory, as shown in the following figure:

Figure 3.8 Configuring TFT_eSPI Library 1

Next, rename **ST7796_Init.h** in the TFT_eSPI library TFT_Drivers directory under the project folder directory to **ST7796_Init. bak. h**, and then copy **ST7796_Init.h** in the Replaced files directory to the TFD_eSPI library **TFT_Drivers** directory under the project folder library directory, as shown in the following figure:

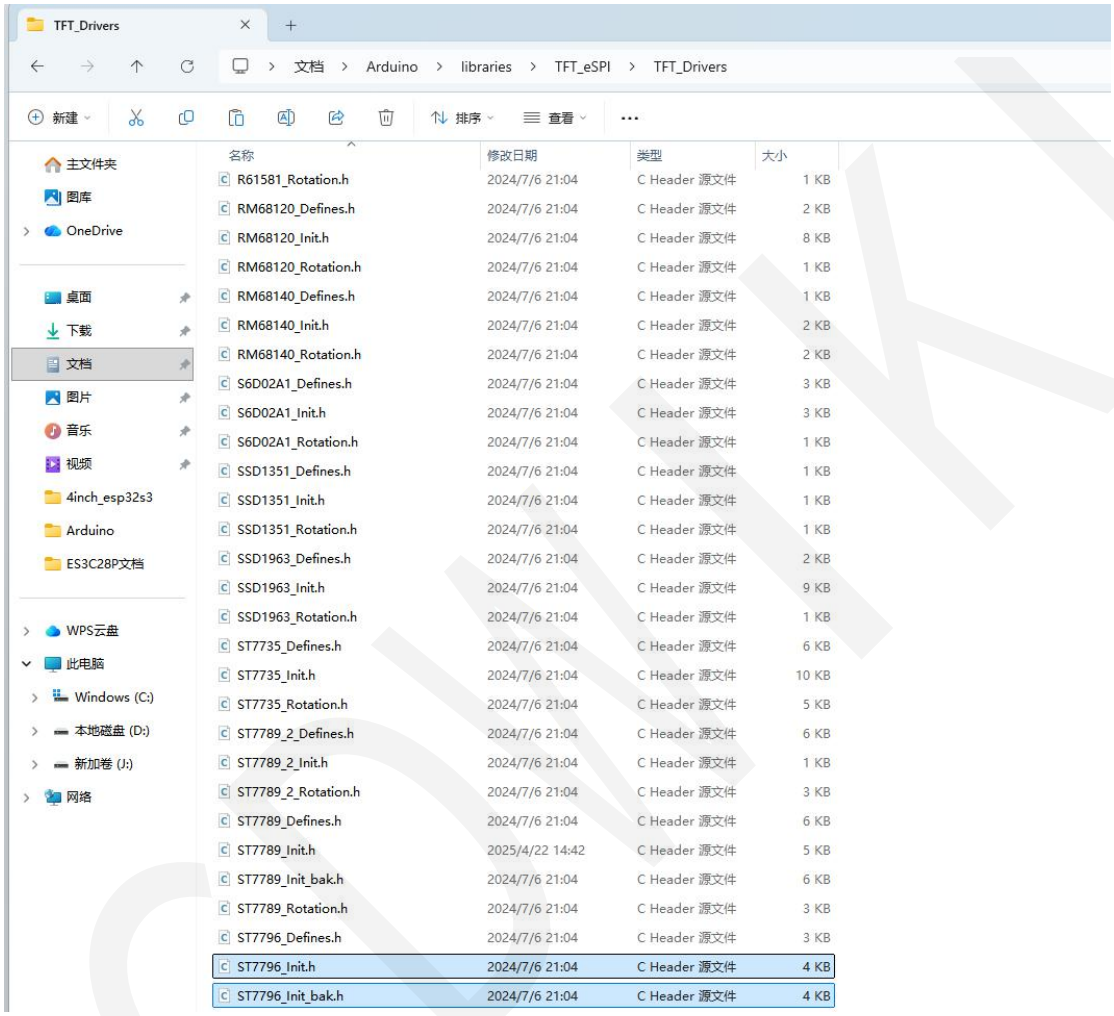


Figure 3.9 Configuring TFT_eSPI Library 2

Replace the **TFT_eSPI_SSP32_S3. c** file in the **TFT_eSPI** library **Processor** directory with the **TFT_eSPI_SSP32_S3. c** file in the Replaced files directory. Alternatively, you can directly modify the **TFT_eSPI-ESP32_S3. c** file in the Processor directory of the TFT_eSPI library, as shown in the following figure:

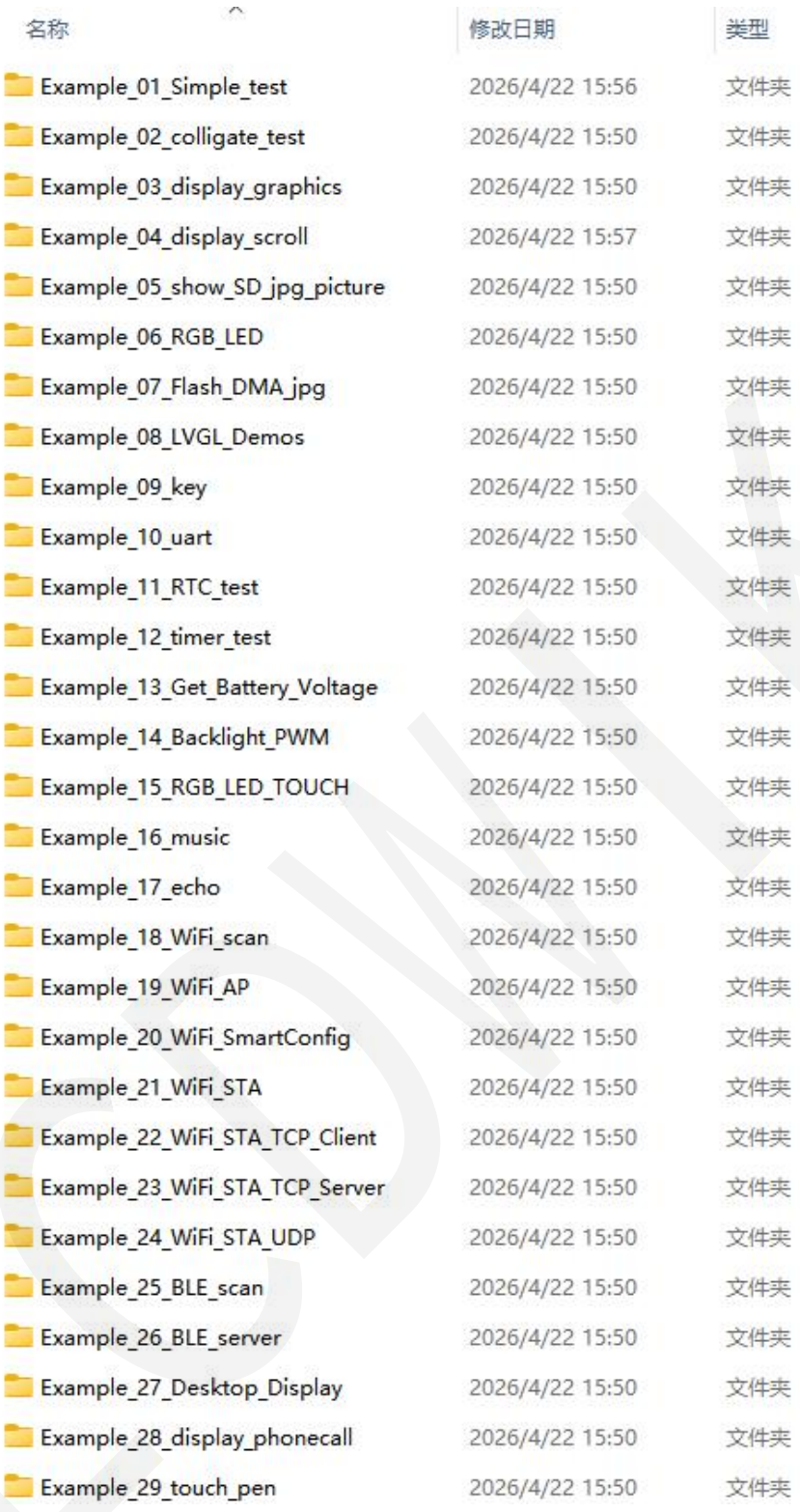
```
/*****************************************************************************************
** Function name:           dma_end_callback
** Description:             Clear DMA run flag to stop retransmission loop
*****************************************************************************************/
extern "C" void dma_end_callback();

void IRAM_ATTR dma_end_callback(spi_transaction_t *spi_tx)
{
  //WRITE_PERI_REG(SPI_DMA_CONF_REG(spi_host), 0);
  WRITE_PERI_REG( SPI_DMA_CONF_REG(SPI_DMA_CH_AUTO), 0b11);
}
```

Figure 3.10 Configuring TFT_eSPI Library 3

3.3. Example program description

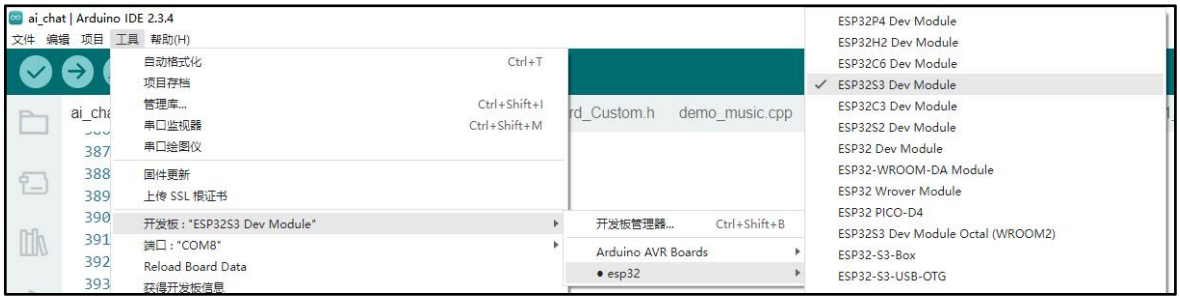
The sample program is located in the "**1-示例程序_Demo \Arduino\demos**" directory under the data package, as shown the following figure:



名称	修改日期	类型
Example_01_Simple_test	2026/4/22 15:56	文件夹
Example_02_colligate_test	2026/4/22 15:50	文件夹
Example_03_display_graphics	2026/4/22 15:50	文件夹
Example_04_display_scroll	2026/4/22 15:57	文件夹
Example_05_show_SD_jpg_picture	2026/4/22 15:50	文件夹
Example_06_RGB_LED	2026/4/22 15:50	文件夹
Example_07_Flash_DMA_jpg	2026/4/22 15:50	文件夹
Example_08_LVGL_Demos	2026/4/22 15:50	文件夹
Example_09_key	2026/4/22 15:50	文件夹
Example_10_uart	2026/4/22 15:50	文件夹
Example_11_RTC_test	2026/4/22 15:50	文件夹
Example_12_timer_test	2026/4/22 15:50	文件夹
Example_13_Get_Battery_Voltage	2026/4/22 15:50	文件夹
Example_14_Backlight_PWM	2026/4/22 15:50	文件夹
Example_15_RGB_LED_TOUCH	2026/4/22 15:50	文件夹
Example_16_music	2026/4/22 15:50	文件夹
Example_17_echo	2026/4/22 15:50	文件夹
Example_18_WiFi_scan	2026/4/22 15:50	文件夹
Example_19_WiFi_AP	2026/4/22 15:50	文件夹
Example_20_WiFi_SmartConfig	2026/4/22 15:50	文件夹
Example_21_WiFi_STA	2026/4/22 15:50	文件夹
Example_22_WiFi_STA_TCP_Client	2026/4/22 15:50	文件夹
Example_23_WiFi_STA_TCP_Server	2026/4/22 15:50	文件夹
Example_24_WiFi_STA_UDP	2026/4/22 15:50	文件夹
Example_25_BLE_scan	2026/4/22 15:50	文件夹
Example_26_BLE_server	2026/4/22 15:50	文件夹
Example_27_Desktop_Display	2026/4/22 15:50	文件夹
Example_28_display_phonecall	2026/4/22 15:50	文件夹
Example_29_touch_pen	2026/4/22 15:50	文件夹

Picture 3.11 Example program

Open the routine and then select ESP32S3 development board in Tools -> Development Board



Picture 3.12

Then select the port to access Other settings are as follows:

Picture 3.13

The following are the introductions to the sample programs：

01_Simple_test

This example is a basic example program, it does not depend on any third-party libraries. Hardware is needed to use LCD display, the display content is fullscreen color filling and random rectangle filling. This example can be used directly to check whether the display is normal.

02_colligate_test

This example needs to depend on the TFT_eSPI software library, and the hardware needs to use the LCD display. The content to be displayed includes drawing, drawing lines, various graphic displays, and running time statistics, which is a relatively comprehensive display example.

03_display_graphics

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display. The display content includes various graphical drawings and fills

04_display_scroll

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display. The display content includes Chinese characters and image display, text display, negative display, and four-directional rotation display.

05_show_SD_jpg_picture

This example requires the TFT_eSPI and TJpg_Decoder library, and the hardware needs to use LCD display and MicroSD card. This example is to read the JPG image in the MicroSD card and parse it, and then display the image on the LCD. The steps of the example are as：

A. Copy the JPG images in the "PIC_320x480" directory under the sample folder to the root directory of the MicroSD card the computer.

B. Insert the MicroSD card into the SD card slot of the display module.

C. Power up the display module, compile and download this example program, you can see the pictures on the LCD screen rotating display.

06_RGB_LED

This example requires the Adafruit_Neopixel library. The hardware needs to use RGB tricolor lights. This example shows how to cycle through the red, green, and blue colors of RGB tricolor lights by pressing a button.

07_Flash_DMA_jpg

This example requires the TFT_eSPI and TJpg_Decoder software libraries. Hardware requirements include an LCD display. This example demonstrates reading JPG image from the internal Flash of the ESP32 module, parsing it, and

then displaying the image on the LCD. Here are the steps to use this example：

A. Take a template of the jpg image to be displayed through the online template tool. Online template tool website：

http://tomeko.net/online_tools/file_to_hex.php?lang=en

B. After successful scanning, copy the scanned data into the array in the "image.h" file in the example folder (you can rename the array, and the example should also be modified accordingly）

C. Power up the display module, compile and download this example program, you can see the picture display on the LCD screen.

8_LVGL_Demos

This example needs to depend on TFT_eSPI, lvgl software library, and the hardware needs to use LCD screen, capacitive touch screen. example shows the 5 built-in Demo functions of lvgl embedded UI system. Through this example, you can learn how to transplant lvgl on ESP32 platform how to configure the underlying devices such as display and touch screen. In the example program, only one demo can be compiled at a time, the demo that needs to be compiled should uncommented, and other demos should be commented, as shown in the figure below:

```
111   // uncomment one of these demos
112    lv_demo_widgets();
113   // lv_demo_benchmark();
114   // lv_demo_keypad_encoder();
115   // lv_demo_music();
116   // lv_demo_stress();
```

Picture 3.14 choose lvgl demo

lv_demo_widgets：various small control test demo

lv_demo_benchmark：Performance benchmark demo

lv_demo_keypad_encoder：Keyboard encoder test demo

lv_demo_music：Music player test demo

lv_demo_stress：Stress test demo

Note: The first compilation of this example takes a long time, about 15 minutes.

09_key_interrupt

This example requires the Adafruit_Neopixel library. Hardware required: BOOT button and RGB LED. This example shows how to detect key events in an-driven way, and how to control the RGB LED on and off by operating the button.

10_uart

This example needs to depend on the TFT_eSPI software library, and the hardware needs to use the serial port and LCD display. This example shows ESP2 interacting with the computer end through the serial port. ESP32 sends information to the computer end through the serial port, and the computer end also sends information to ESP32 the serial port, and ESP32 receives and displays the information on the LCD screen.

11_RTC_test

This example requires the TFT_eSPI and ESP32Time library, and the hardware needs to use an LCD display. This example shows how to set the-time time and date using the ESP32's RTC module and display the time and date on the LCD display.

12_timer_test

This example requires the Adafruit_Neopixel library. The hardware needs to use RGB three-color lights. This example shows the use of ESP32ers, by setting a 1-second timer to control the green LED light to turn on and off.

13_Get_Battery_Voltage

This example relies on the TFT_eSPI software library. The hardware requires an LCD display and a 3.7V lithium battery. This example demonstrates how use the ESP32's ADC feature to obtain the voltage of an external lithium battery and display it on the LCD screen.

14_Backlight_PWM

This example needs to depend on the TFT_eSPI library, and the hardware needs to use LCD display and capacitive touch screen. This example shows how adjust the backlight brightness of the display screen by touching and sliding the display module, and display the brightness value change.

15_RGB_LED_TOUCH

This example needs to depend on TFT_eSPI, Adafruit_Neopixel software library, and the hardware needs to use LCD screen, capacitive screen and RGB tricolor light. This example shows the RGB light control of on and off, flashing and brightness adjustment through touch buttons.

16_music

This example requires the TJpg_Decoder and ESP32-audioI2S software libraries, and the hardware includes an LCD display, capacitive touch, speaker, and MicroSD card. This example demonstrates how to read an audio file in mp3 format from an SD card and play it in a loop. The steps to use example are as follows：

A. After opening the example, you must first set the Tools->Partition Scheme to the Huge APP (3MB No OTA / 1MB SPIFF) option, otherwise you will get an error that the memory is insufficient during compilation.

B. Copy all the mp3 audio files in the "mp3" directory in the example folder to the MicroSD card. Of course, you can also not use the files in this directory and find some other mp3 audio files.

C. Insert the MicroSD card into the SD card slot of the display module.

D. Power on the display module, compile and download this example program, and the external speaker will play the sound.

17_echo

This example needs to depend on the ESP32-audioI2S software library, and the hardware needs to use a speaker and microphone. This example shows that you speak into the microphone, it causes the speaker to sound.

18_WiFi_scan

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display and ESP32 WIFI module. This example shows ESP32 WIFI module scanning surrounding wireless network information in STA mode. And display the scanned wireless network information on the LCD display.

The wireless network information includes SSID RSSI, CHANNEL, ENC_TYPE and other content. When the wireless network information scanning is completed, the number of scans will be displayed, and only the first17 scanned wireless network information will be displayed at most.

19_WiFi_AP

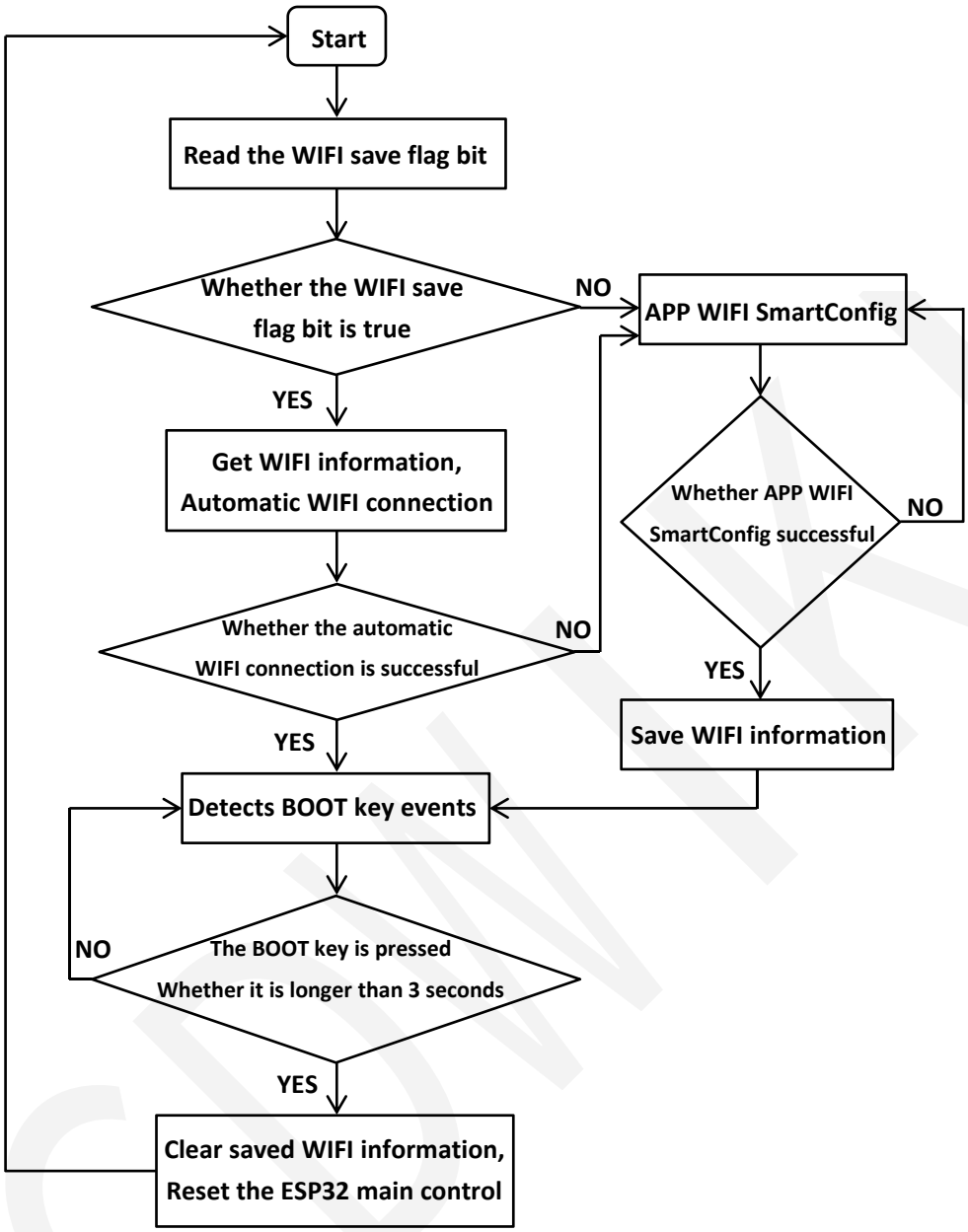
This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display and ESP32 WIFI module. This example the ESP32 WIFI module set to AP mode for WIFI terminal connection. On the display, it will display the SSID, password, host IP address, host MAC, and other information set under the AP mode of the ESP32 WIFI module. Once a terminal is successfully connected, the display will show the number of terminals connected. Set SSID and password in the "ssid" and "password" variables at the beginning of the example program, as shown in the figure below :

```
19
20 //AP mode SSID and PWD
21 const char *ssid = "ESP32_AP";
22 const char *password = "0123456789";
```

Picture 3.15 Set SSID and password under AP mode

20_WiFi_SmartConfig

This example requires the TFT_eSPI software library, and the hardware needs to use LCD display, ESP32 WIFI module and BOOT button. example shows the process of intelligent network configuration of ESP32 WIFI module in STA mode through EspTouch mobile APP. The whole example program operation flow chart is as:



Picture 3.16 The flowchart of the smart distribution network example program running

The operation steps of this example program are as follows:

A. Download the **EspTouch** application on your mobile phone, or copy the installation program **"esptouch-v2.0.0.apk"** from **“7-工具软件_Tool_software”** folder under the data package (only the installation program for the Android system, iOS system applications can only be installed from the device), and you can download the installation program from the official website. The official website download address is as follows:

https://www.espressif.com.cn/en/support/download/apps

B. Power on the display module, compile and download this example program, if there is no saved WIFI information in the ESP32, it will directly enter smart configuration mode, at this time, open the EspTouch application on the mobile phone, enter the SSID and password of the WIFI connected to the mobile phone, and then the relevant information in the form of UDP broadcast, once the ESP32 receives this information, it will connect to the network according to the SSID and password in the information and once the network is successfully connected, it will be displayed on the display screen. SSID, password, IP address and MAC address, etc., and save WIFI. It should be noted that the success rate of this configuration method is not very high, if it fails, you need to try several times.

C. If the ESP32 has saved WIFI information, it will automatically connect to the network according to the saved WiFi information when it is turned on. If the fails, it will enter the smart configuration mode. After the network connection is successful, press and hold the BOOT for more than 3 seconds to clear the saved WIFI information reset the ESP32 to re-enter the smart configuration.

21_WiFi_STA

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display, ESP32 WIFI module. This example program the process of ESP32 connecting to WIFI in STA mode according to the provided SSID and password. The steps of this example program are as follows:

A. Write the WIFI information to be connected in the "**ssid**" and "**password**" variables at the beginning of the example program, as shown in the figure below:

```
17 #include <TFT_eSPI.h>
18 #include <WiFi.h>
19
20 //Manually modifying parameters
21 const char *ssid = "yourssid";
22 const char *password = "yourpwd";
23
```

Picture 3.17 Write the WIFI information

B. Power on the display module, compile and download this example program,you can see the ESP32 start to connect to WIFI on the display. If the WIFI connection is successful, the display will show the success prompt, SSID, IP address, MAC address and other information; if the connection time exceeds 3 minutes, the connection fails and the failure prompt is displayed.

22_WiFi_STA_TCP_Client

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display, ESP32 WIFI module. This example program shows the process of ESP32 connecting to TCP server as a TCP client after connecting to WIFI in STA mode. The steps of this example program are as follows:

A. Write the SSID, password, server IP, and server port variables at the beginning of the example program with the WIFI information to be connected, the IP of the TCP server (the IP address of the computer) and the port number, as shown in the following figure:

```
//Manually modifying parameters
const char *ssid = "yourssid";
const char *password = "yourpwd";

const IPAddress serverIP(192,168,4,52); //The server address to b
uint16_t serverPort = 8080;          //Server port number

char t_buf[100] = {0};
```

Picture 3.18 Write WIFI information and TCP server information 1

B. Open the **"TCP&UDP Test Tool"** or **"Network Debug Assistant"** and other test tools on the computer (the installation package is in the**"7 工具软件_software"** directory of the data package), create a TCP server in the

tool, and the port number should be consistent with the setting in the example program.

C. Power on the display module, compile and download this example program, you can see the ESP32 start to connect to WIFI on the display. If theIFI connection is successful, the display will show the success prompt, SSID, IP address, MAC address, TCP server port number and other information, and then start to connect to TCP server. After the connection is successful, there will be a corresponding prompt. At this time, you can communicate with the server.

23_WiFi_STA_TCP_Server

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display, ESP32 WIFI module. This example program the process of ESP32 being connected to TCP clients as a TCP server in STA mode after connecting to WIFI. The operation steps of this example program are as:

A. Write the WIFI information to be connected and the TCP server port number in the "**ssid**", "**password**", "port" variables at the beginning of example program, as shown in the figure below:

```
//Manually modifying parameters
const char *ssid = "yourssid";
const char *password = "yourpwd";

char t_buf[100] = {0};
int port = 10000;

WiFiServer server(port); //Declare server objects
```

Picture 3.19 Write WIFI information and TCP server information 2

B. Power on the display module, compile and download this example program you can see the ESP32 start to connect to WIFI on the display. If theIFI connection is successful, the display will show the success prompt, SSID, IP address, MAC address, TCP server port number and other information, and then create a TCP server wait for the TCP client to connect.

C. Open **"TCP&UDP Test Tool"** or **"Network Debug Assistant"** and other test tools on the computer (the installation package is in the **"7 工具软件_software"** directory of the data package), create a TCP client in the tool (note that the IP address and port number must be consistent with the content displayed on display), and then start connecting to the server. If the connection is successful, there will be a corresponding prompt, and you can communicate with the server at this time.

24_WiFi_STA_UDP

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display, ESP32 WIFI module. This example program the process of ESP32 being connected by a UDP client in STA mode after connecting to WIFI and acting as a UDP server. The steps of this example program as follows:

A. Write the WIFI information to be connected and the UDP server port number in the "**ssid**", "**password**", "**localUdpPort**" variables at the beginning of the example program, as shown in the figure below :

```
//Manually modifying parameters
const char *ssid = "yourssid";
const char *password = "yourpwd";

char t_buf[100] = {0};

AsyncUDP udp;                          //Creating UDP Objects
unsigned int localUdpPort = 10000;  //Local port number
```

Picture 3.20 Write WIFI information and UDP server information

B. Power on the display module, compile and download this example program,you can see the ESP32 start to connect to WIFI on the display. If theIFI connection is successful, the display will show the success prompt, SSID, IP address, MAC address, local port number and other information, and then create the UDP server waiting for the UDP client to connect.

C. Open **"TCP&UDP Test Tool"** or **"Network Debug Assistant"** and other test tools on the computer (the installation package is in the **"7 工具软件_software"** directory of the data package), create a UDP client in the

tool (note that the IP address and port number must be consistent with the content displayed the display), and then start connecting to the server. If the connection is successful, there will be a corresponding prompt, and you can communicate with the server at this time

25_BLE_scan

This example requires the TFT_eSPI software library, and the hardware needs to use an LCD display, ESP32 Bluetooth module. This example shows the32 Bluetooth module scanning the surrounding BLE Bluetooth devices, and displays the names and RSSI of the scanned BLE Bluetooth devices with names on the LCD display.

26_BLE_server

This example requires the TFT_eSPI software library and can only be used with the 2.0 version of the core software library for Arduino-ES32 (e.g., version 2.0.17). The hardware required includes an LCD display and an ESP32 Bluetooth module. This example demonstrates process of creating a Bluetooth BLE server endpoint with the ESP32 Bluetooth module, which is then connected to and communicates with a Bluetooth BLE client. The steps for this are as follows:

A. Install Bluetooth BLE debugging tools on your phone, such as "**BLE** Debug Assistant", "**LightBlue**", etc.

B. Power on the display module, compile and download this example program, you can see the Bluetooth BLE client running prompt on the display screen. If you want to the Bluetooth BLE server device name by yourself, you can modify it in the parameter of the "**BLEDevice::init**" function in the example program, as shown in figure below :

```
68 void setupBLE()
69 {
70   BLEDevice::init("ESP32_BT_BLE");    //Create BLE device
71   pServer = BLEDevice::createServer();    //Create BLE server
72   pServer->setCallbacks(new MyServerCallbacks());    //Set the d
```

Picture 3.21 Set Bluetooth BLE service end device name

C. Open the Bluetooth and Bluetooth BLE debugging tools on the mobile phone, search for the Bluetooth BLE service end device name (the default is

"ESP3_BT_BLE"), and then click on the name to connect. After the connection is successful, the ESP32 display module will prompt. Next, you can perform communication.

27_Desktop_Display

This example program needs to depend on the ArduinoJson, Time, HttpClient, TFT_eSPI, TJpg_Decoder, NTPClient library. Hardware needs to use LCD screen, ESP32 WIFI module. This example shows a weather clock desktop, which can display the city weather conditions (including temperature, humidity weather icon, and scroll to display other weather information), the current time and date, and a spaceman animation. Weather information is obtained from the Internet from the Weather Network, time information is updated from the NTP server. The steps of this example program are as follows:

A. After opening the example, you must first set the **Tools->Partition Scheme to the Huge APP (3MB No OTA / 1MB SPIFFS)** option, otherwise you will get an error that the memory is insufficient during compilation.

B. B、Write the WIFI information to be connected in the "ssid" and "passwd" variables at the beginning of the example program, as shown in the below. If not set, Smart Networking will be performed (for details on Smart Networking, please refer to the Smart Delivery example program).

```
42 //---------------wifi information---------------------
43 const char* ssid = "yourssid";      //WIFI name
44 const char* passwd = "yourpasswd";    //WIFI password
45 //----------------------------------------------------
```

Picture 3.22 Set up WIFI information

C. Power the display module, compile and download this example program, and you can see the weather clock desktop on the display.

28_display_phonecall

This example needs to depend on TFT_eSPI, TFT_Touch software library. Hardware needs to use LCD display and resistive touch screen. This example a simple mobile phone dial interface, which inputs content through touch buttons.

29_touch_pen

This example needs to depend on TFT_eSPI, TFT_Touch software library. Hardware needs to use LCD display and resistive touch screen. This example that the line is drawn on the display through touch, which can detect whether the touch screen function is normal.