5.0inch RGB Arduino Demo Program Description

Contents

1. Software and Hardware Platform Description

Module: 5.0-inch ESP32-S3 RGB Display Module with 800x480 resolution.

Module Main Controller: ESP32-S3 chip, maximum frequency 240MHz, supporting 2.4G WIFI + Bluetooth.

Arduino IDE Version: 2.3.4.

ESP32 Arduino Core Library Version: 3.2.0。

2. Pin Assignment Description

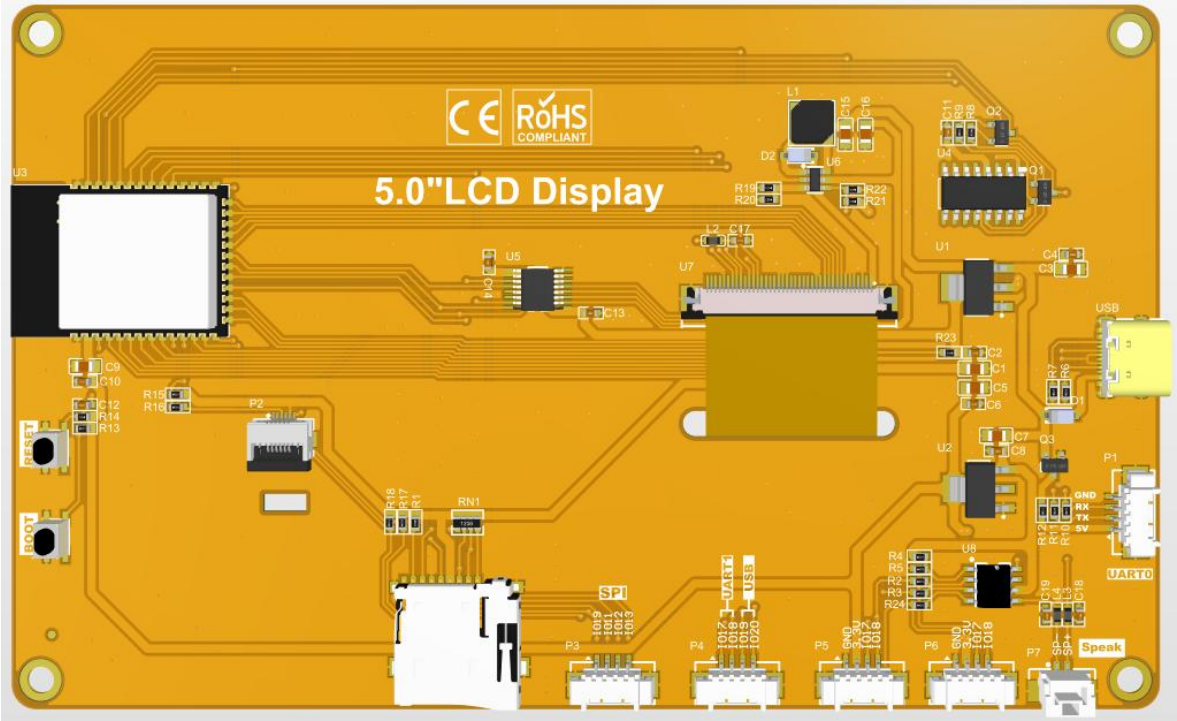

Figure 2.1 Back View of 5.0-inch ESP32-S3 Display Module

The main controller of the 5.0-inch ESP32-S3 display module is ESP32-S3, and the GPIO assignment for connecting on-board peripherals is shown in the following table:

ESP32-S3 Pin Assignment Description			
On-board Device	On-board Device Pin	ESP32-S3 Connected Pin	Description
LCD	LCD_DE	IO40	Pixel Clock Control Pin
	LCD_VSYNC	IO41	5-bit RED Data Pin
	LCD_HSYNC	IO39	6-bit GREEN Data Pin
	LCD_PCLK	IO42	Pixel Clock Control Pin
	PIN_R0	IO45	5-bit RED Data Pin
	PIN_R1	IO48	
	PIN_R2	IO47	
	PIN_R3	IO21	
	PIN_R4	IO14	
	PIN_G0	IO5	6-bit GREEN Data Pin
	PIN_G1	IO6	
	PIN_G2	IO7	
	PIN_G3	IO15	
	PIN_G4	IO16	
	PIN_G5	IO4	
	PIN_B0	IO8	5-bit BLUE Data Pin
	PIN_B1	IO3	
	PIN_B2	IO46	
	PIN_B3	IO9	
	PIN_B4	IO1	
RTP	TP_SCK	IO12	Resistive Touch Screen SPI Bus Clock Control Pin
	TP_MISO	IO13	Resistive Touch Screen SPI Bus Data Write Control Pin
	TP_MOSI	IO11	Resistive Touch Screen SPI Bus Data Read Control Pin

	TP_CS	IO38	Resistive Touch Screen SPI Bus Chip Select Control Pin
	TP_INT	IO18	Resistive Touch Screen SPI Bus Interrupt Control Pin
SD	SD_CS	IO10	SD Card SPI Bus Chip Select Control Pin
	SD_MOSI	IO11	SD Card SPI Bus Data Read Control Pin
	SD_MISO	IO13	SD Card SPI Bus Data Write Control Pin
	SD_SCLK	IO12	SD Card SPI Bus Clock Control Pin
KEY	KEY_BOOT	IO0	Download Mode Selection Button (Press this button to power on, then release to enter download mode)
	KEY_RESET	EN	ESP32-S3 Reset Button, Active Low Reset
UART0	TX0	IO43	ESP32-S3 UART0 Transmit Signal Pin
	RX0	IO44	ESP32-S3 UART0 Receive Signal Pin
AUDIO	I2S_BCLK	IO0	
	I2S_SDATA	IO17	
	I2S_LRCLK	IO18	

Table 2.1 ESP32-S3 On-board Peripheral Pin Assignment Description

3. Demo Program Usage Instructions

3.1. Set up the ESP32 Arduino Development Environment

For detailed instructions on setting up the ESP32 Arduino development environment, refer to the "ESP32_Arduino_IDE Development Environment Setup" document in the data package.

3.2. Install Third-Party Software Libraries

After setting up the development environment, you first need to install the third-party software libraries used by the demo programs. The steps are as follows:

A. Open the **“1-示例程序_Demo\Arduino\Install libraries”** directory in the data package and find the third-party software libraries, as shown in the following figure:

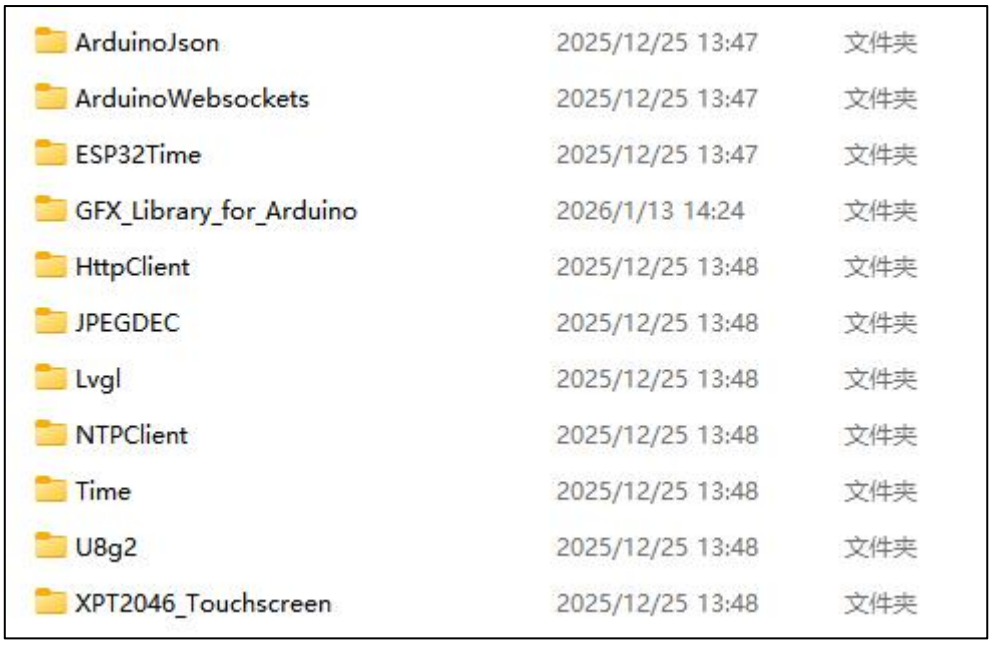

Figure 3.1 Third-Party Software Libraries for Demo Programs

Among them:

- **ArduinoJson:** A C++ JSON software library for Arduino and the Internet of Things.
- **ArduinoWebsockets:** A library for writing modern websockets applications with Arduino.
- **ESP32Time:** An Arduino software library for setting and retrieving the internal RTC time on ESP32 boards.
- **HttpClient:** An HTTP client software library for interacting with web servers from Arduino.
- **lvgl:** A highly customizable, low-resource, aesthetically pleasing and easy-to-use embedded system graphics software library.
- **NTPClient:** An NTP client software library for connecting to NTP servers.
- **GFX_Library_for_Arduino:** An Arduino graphics library for LCD screens, supporting multiple platforms and multiple LCD driver ICs.
- **Time:** A software library that provides timing functions for Arduino.
- **JPEGDEC:** A JPG image decoding library for the Arduino platform, which can decode JPG files from SD cards or Flash and display them on LCD screens.
- **U8g2:** A font library for the Arduino platform that can display Chinese characters on LCD screens.

- **XT_DAC_Audio:** A resistive touch driver library that enables touch functionality.

B、Copy these software libraries to the library directory of the project folder. The default library directory of the project folder is **"C:\Users\Administrator\Documents\Arduino\libraries"** (the red part is the actual user name of the computer). If you have modified the project folder path, you need to copy them to the library directory of the modified project folder.

C、The third-party software libraries in the above data package have been configured and can be used directly. After installation, open the demo program, compile and run it to see the effect.

3.3. Demo Program Usage Instructions

The demo programs are located in the **"1-Demo Programs_Demo \Arduino\demos"** directory of the data package, as shown in the following figure:

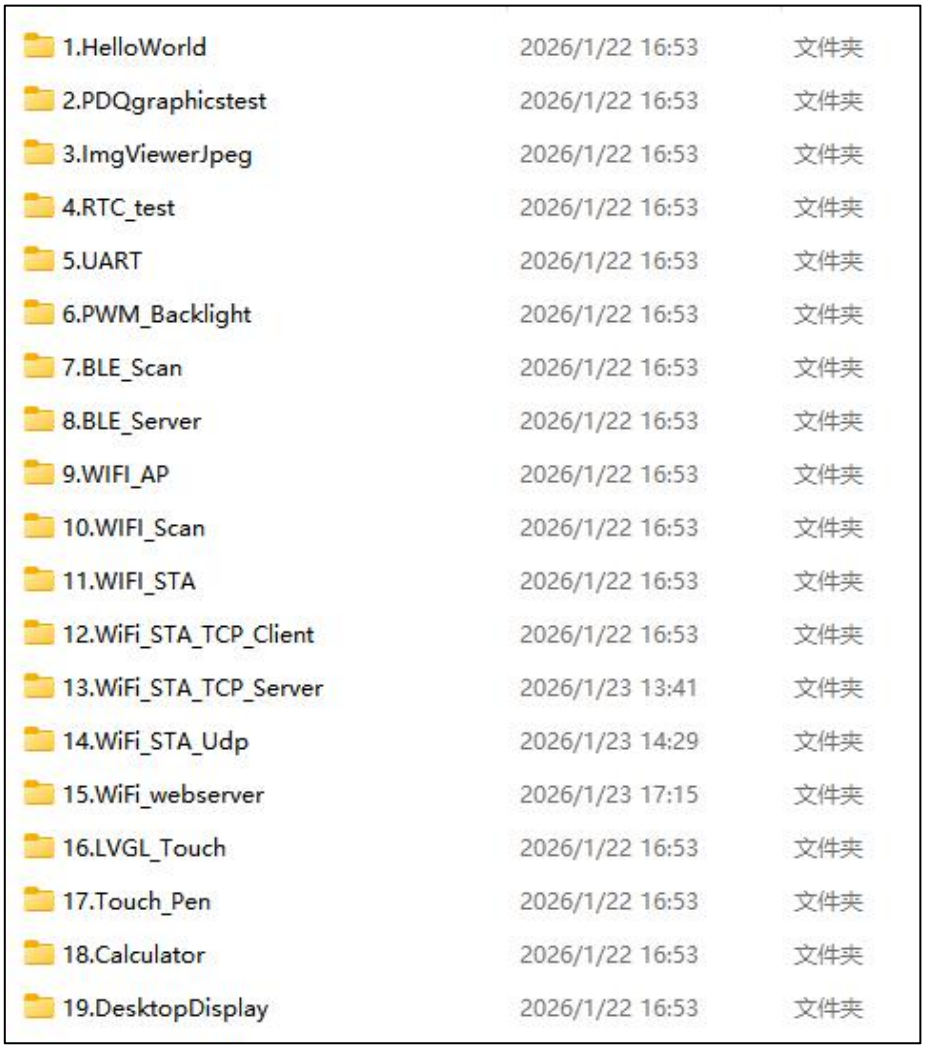

Figure 3.2 Demo Programs

After opening the routine, select the ESP32S3 development board in Tools -> Board

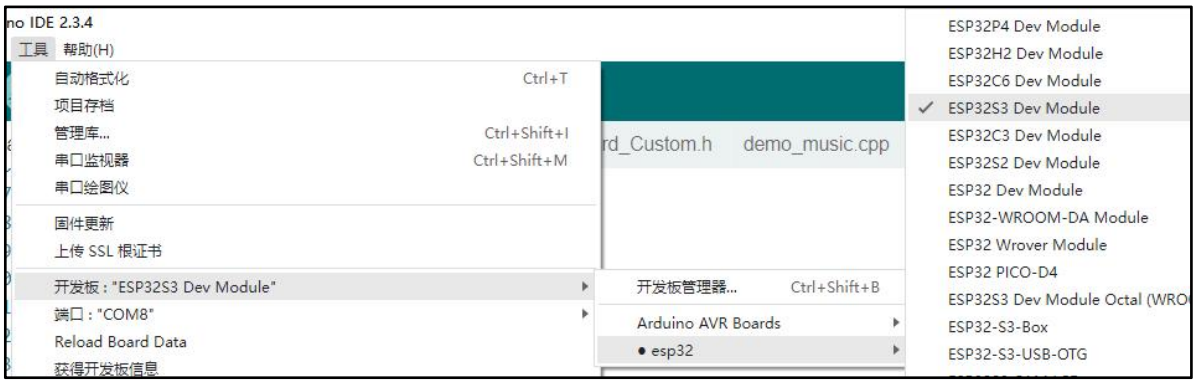

Figure 3.3

Then select the connected port

The remaining settings are as follows

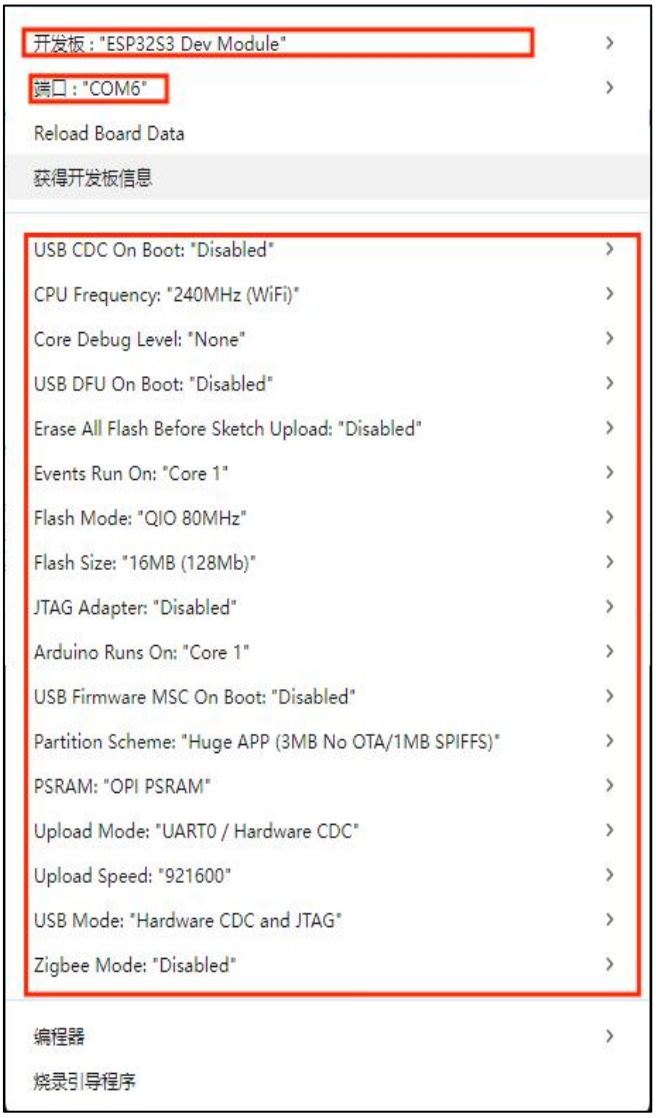

Figure 3.4

Introductions to each demo program are as follows:

01_HelloWorld

This demo is a basic example program that requires the GFX_Library_for_Arduino library. The hardware requires an LCD display, and the displayed content is randomly filled with "Helloworld" characters. This demo is used to check if the display is working properly.

02_PDQgraphicstest

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display. The displayed content includes full-screen color filling, dot drawing, line drawing, and various graphic displays, making it a comprehensive display demo.

03_ImgViewerJpeg

This demo requires the GFX_Library_for_Arduino library and the JPEGDEC software library, and the hardware requires an LCD display and a MicroSD card. The function of this demo is to read and parse JPG pictures from the MicroSD card, then display the pictures on the LCD. The steps to use the demo are as follows:

- A. Copy the JPG pictures in the "DATA" directory of the demo folder to the root directory of the MicroSD card via a computer.
- B. Insert the MicroSD card into the SD card slot of the display module;
- C. Power on the display module, compile and download the demo program, and you can see pictures rotating on the LCD screen.

04_RTC_test

This demo requires the GFX_Library_for_Arduino library and the ESP32Time software library, and the hardware requires an LCD display. This demo shows how to set the time and date using the RTC module of the ESP32 and display the time and date on the LCD display.

05_Uart

This demo requires the GFX_Library_for_Arduino library, and the hardware requires a UART and an LCD display. This demo shows the interaction between the ESP32 and the computer via UART. The ESP32 sends information to the computer via UART, and the computer also sends information to the ESP32 via UART, which the ESP32 receives and displays on the LCD screen.

06_PWM_Backlight

This demo requires the GFX_Library_for_Arduino library and the XT_DAC_Audio library, and the hardware requires an LCD display and a resistive touch screen. This demo shows how to adjust the backlight brightness of the display through touch sliding operations on the display module, and display the change of the brightness value at the same time.

07_BLE_Scan

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display. This demo shows the ESP32 Bluetooth module scanning surrounding BLE Bluetooth devices and displaying the names and RSSI of the scanned BLE Bluetooth devices with names on the LCD display.

08_BLE_server

This demo requires the GFX_Library_for_Arduino library and the U8g2 font library, and the hardware requires an LCD display. This demo shows the process of the ESP32 Bluetooth module creating a BLE server, being connected by a BLE client, and communicating. The steps to use this demo are as follows:

- A. Install a BLE debugging tool on your mobile phone, such as "BLE Debug Assistant", "LightBlue", etc.
- B. Power on the display module, compile and download the demo program, and you can see the BLE client running prompt on the display. If you want to modify the BLE server device name yourself, you can modify it in the parameter passed by the "BLEDevice::init" function in the demo program, as shown in the following figure:

```
// ====================== BLE初始化函数 ======================
void setupBLE()
{
    // 初始化BLE设备
    BLEDevice::init("ESP32_BT_BLE");
    // 创建BLE服务器
    pServer = BLEDevice::createServer();
    pServer->setCallbacks(new MyServerCallbacks());
    // 创建BLE服务
```

Figure 3.6 Set BLE Server Device Name

- C. Turn on Bluetooth and the BLE debugging tool on the mobile phone, search for the BLE server device name (default is "ESP32_BT_BLE"), then click the name to connect. After successful connection, the ESP32 display module will have a prompt. Then you can perform Bluetooth communication.

09_WiFi_scan

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo shows the ESP32 WIFI module scanning surrounding wireless network information in STA mode. The scanned wireless network information is displayed on the LCD display. The wireless network information includes SSID, RSSI, CHANNEL, ENC_TYPE and other contents. After the wireless network information scanning is completed, the number of scans will be displayed, and only the first 17 scanned wireless network information will be displayed at most.

10_WiFi_AP

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo shows the ESP32 WIFI module set to AP mode for WIFI terminal connection. The display will show the SSID, password, host IP address, host MAC address and other information set by the ESP32 WIFI module in AP mode. Once a terminal is successfully connected, the display will show the number of connected terminals. You can set the SSID and password yourself in the "ssid" and "password" variables at the beginning of the demo program, as shown in the following figure:

```
// ====================== WiFi AP配置 ======================
#define AP_SSID "ESP32-S3_AP"     // AP热点名称
#define AP_PASSWORD "12345678"    // AP密码（至少8位，否则无法启动AP）
#define AP_CHANNEL 6              // AP信道（1~13）
#define AP_MAX_CONNECT 4          // 最大连接数（ESP32最多支持4个）
```

图 3.7 设置 AP 模式下 SSID 和密码

11_WiFi_STA

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo program shows the process of the ESP32 connecting to WIFI with the provided SSID and password in STA mode. The operation steps of this demo program are as follows:

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

```
// ====================== WiFi STA配置 ======================
#define WIFI_SSID "ESP"     // 要连接的WiFi名称
#define WIFI_PASSWORD "12345678" // 要连接的WiFi密码
#define WIFI_CONNECT_RETRY 5          // 连接失败重试次数
#define WIFI_CONNECT_DELAY 3000       // 重试间隔（毫秒）
```

Figure 3.8 Write WIFI Information

- **B.** Power on the display module, compile and download the demo program, and you can see the ESP32 starting to connect to WIFI on the display. If the WIFI connection is successful, the success prompt, SSID, IP address, MAC address and other information will be displayed on the display; if the connection fails more than 5 times, the connection will fail and a failure prompt will be displayed.

12_WiFi_STA_TCP_Client

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo program shows the process of the ESP32, after connecting to WIFI in STA mode, acting as a TCP client to connect to a TCP server. The operation steps of this demo program are as follows:

- **A.** Write the WIFI information to be connected, the IP address of the TCP server (computer's IP address) and the port number into the "ssid", "password", "serverIP", "serverPort" variables at the beginning of the demo program, as shown in the following figure:

```
// ====================== WiFi STA配置 ======================
#define WIFI_SSID         "ESP"         // 参考代码的WiFi名称
#define WIFI_PASSWORD     "12345678"    // 参考代码的WiFi密码
#define WIFI_CONNECT_RETRY 5            // WiFi连接失败重试次数
#define WIFI_CONNECT_DELAY 3000         // WiFi重试间隔（毫秒）

// ====================== TCP Client配置 ======================
#define TCP_SERVER_IP     "192.168.1.126"// 参考代码的TCP服务器IP
#define TCP_SERVER_PORT   8080          // 参考代码的TCP服务器端口
#define TCP_CONNECT_RETRY 3             // TCP连接失败重试次数
#define TCP_CONNECT_DELAY 2000          // TCP重试间隔（毫秒）
#define TCP_RECV_BUFFER_SIZE 1024       // 接收缓冲区大小
```

Figure 3.9 Write WIFI Information and TCP Server Information 1

- B. Open a test tool such as **“TCP&UDP 测试工具”** or **“网络调试助手”** on the computer (the installation package is in the **“7-工具软件_Tool_software”** directory of the data package), and create a TCP server in the tool with the same port number as set in the demo program.

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

13_WiFi_STA_TCP_Server

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo program shows the process of the ESP32, after connecting to WIFI in STA mode, acting as a TCP server to be connected by a TCP client. The operation steps of this demo program are as follows:

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

```
// ====================== WiFi STA配置 ======================
#define WIFI_SSID         "yourssid"         // WiFi名称
#define WIFI_PASSWORD     "yourpass"    // WiFi密码
#define WIFI_CONNECT_RETRY 5            // WiFi连接失败重试次数
#define WIFI_CONNECT_DELAY 3000         // WiFi重试间隔（毫秒）

// ====================== TCP Server配置 ======================
#define TCP_SERVER_PORT   10000          // TCP服务器监听端口
#define TCP_CLIENT_TIMEOUT 5000         // 客户端超时时间（毫秒）
#define TCP_RECV_BUFFER_SIZE 1024       // 接收缓冲区大小
#define MAX_CLIENTS 1                   // 最大支持客户端数
```

Figure 3.11 Write WIFI Information and TCP Server Information 2

- B. Power on the display module, compile and download the demo program, and you can see the ESP32 starting to connect to WIFI on the display. If the WIFI connection is successful, the success prompt, SSID, IP address, MAC address, TCP server port number and other information will be displayed on the display, then a TCP server will be created to wait for TCP client connection.

- C. Open a test tool such as "TCP&UDP Test Tool" or "Network Debug Assistant" on the computer (the installation package is in the "7-Tool Software_Tool_software" directory of the data package), and create a TCP client in the tool (note that the IP address and port number must be consistent with those displayed on the display), 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.

14_WiFi_STA_UDP

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo program shows the process of the ESP32, after connecting to WIFI in STA mode, acting as a UDP server to be connected by a UDP client. The operation steps of this demo program are as follows:

- A. Write the WIFI information to be connected and the UDP server port number into the "ssid", "password", "localUdpPort" variables at the beginning of the demo program, as shown in

the following figure:

```
// ====================== WiFi配置 ======================
const char *ssid = "yourssid";
const char *password = "yourpass";

// ====================== UDP配置 ======================
WiFiUDP udp;                        // 同步UDP对象
unsigned int localUdpPort = 10000; // 本地监听端口
unsigned int broadcastPort = localUdpPort;
const char *broadcastData = "broadcast UDP data";
char udpRecvBuffer[256] = {0};     // 接收缓冲区（避免内存溢出）
```

Figure 3.12 Write WIFI Information and UDP Server Information

- B. Power on the display module, compile and download the demo program, and you can see the ESP32 starting to connect to WIFI on the display. If the WIFI connection is successful, the success prompt, SSID, IP address, MAC address, local port number and other information will be displayed on the display, then a UDP server will be created to wait for UDP client connection.

- C. Open a test tool such as "TCP&UDP Test Tool" or "Network Debug Assistant" on the computer (the installation package is in the "7-Tool Software_Tool_software" directory of the data package), and create a UDP client in the tool (note that the IP address and port number must be consistent with those displayed on the display), 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.

15_WIFI_Webserver

This demo requires the GFX_Library_for_Arduino library, and the hardware requires an LCD display and an ESP32 WIFI module. This demo shows the ESP32 WIFI module set to AP mode for WIFI terminal connection. The display will show the SSID, password, host IP address, host MAC address and other information set by the ESP32 WIFI module in AP mode. Once a terminal is successfully connected, the display will show the number of connected terminals. After connecting to the hotspot, enter the following URL "192.168.4.1" to access the webpage, where you can control the LCD brightness and backlight color. The steps to use the demo are as follows:

- A: Set the SSID and password yourself in the "ssid" and "password" variables at the beginning of the demo program, as shown in the following figure:

```
// ====================== WiFi AP 配置 ======================
const char *ap_ssid = "ESP";           // 设置热点的的SSID
const char *ap_password = "12345678";  // 设置热点的密码
WebServer server(80);
```

Figure 3.13

- B: After burning the code, use a mobile phone to connect to the hotspot. After successful

connection, enter the URL "192.168.4.1" to control the LCD brightness and backlight color on the webpage.

16_LVGL_Touch

This demo requires the lvgl software library, GFX_Library_for_Arduino library and XT_DAC_Audio library, and the hardware requires an LCD display and a resistive touch screen. This demo shows 4 built-in Demo functions of the lvgl embedded UI system. Through this demo, you can learn how to port lvgl on the ESP32 platform and how to configure underlying devices such as displays and touch screens. Only one demo can be compiled at a time in the demo program. Uncomment the demo you need to compile and add comments to other demos, as shown in the following figure:

```
lv_demo_widgets();
//lv_demo_benchmark();
//lv_demo_music();
//lv_demo_stress();
```

Figure 3.14 Select lvgl Demo

- lv_demo_widgets: Various small control test demo
- lv_demo_benchmark: Performance benchmark test demo
- lv_demo_music: Music player test demo
- lv_demo_stress: Stress test demo

17_Touch_pen

This demo requires the GFX_Library_for_Arduino library and the XT_DAC_Audio library. The hardware requires an LCD display and a resistive touch screen. This demo shows drawing lines on the display through touch, which can detect whether the touch screen function is normal.

18_Calculator

This demo requires the GFX_Library_for_Arduino library, XT_DAC_Audio library and U8g2 software library, and the hardware requires an LCD display and a resistive touch screen. This demo shows a simple touch calculator that can perform simple addition, subtraction, multiplication and division operations.

19_Desktop_Display

This demo program requires ArduinoJson, Time, HttpClient, GFX_Library_for_Arduino library,

U8g2 software library, and NTPClient library. The hardware requires an LCD display and an ESP32 WIFI module. This demo shows a weather clock desktop that can display city weather conditions (including temperature, humidity, and other weather information), current time and date. Weather information is obtained from the weather website through the network, and time information is updated from the NTP server. The operation steps of this demo program are as follows:

- A. After opening the demo, you first need to set Tools->**Partition Scheme** to the **Huge APP (3MB No OTA /1MB SPIFFS) option**, otherwise a memory insufficient error will occur during compilation.

- B. Write the WIFI information to be connected into the "ssid" and "passwd" variables at the beginning of the demo program, as shown in the following figure.

```
// ====================== 网络配置（请修改为你的WiFi名称和密码）
======================
const char* WIFI_SSID = "ESP";     // WiFi名称
const char* WIFI_PASSWORD = "12345678"; // WiFi密码
const uint32_t CONNECT_TIMEOUT = 10000; // 连接超时时间：10秒（10000毫秒）
const uint32_t PROGRESS_UPDATE_INTERVAL = 100; // 进度条更新间隔：100毫秒
```

Figure 3.15 Set WIFI Information

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

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