EC5R28T
2.8inch TN Arduino program description

Contents

1. Software and Hardware Platform Description

Module: 2.8-inch ESP32-C5 TN display module, 240x320 resolution, ST7789 LCD driver IC.

Module MCU: ESP32-C5 chip, maximum frequency 240MHz, supports 2.4G/5G WiFi + Bluetooth.

Arduino IDE version: 2.3.4

ESP32 Arduino core library version: 4.0.0

2. Pin Assignment Description

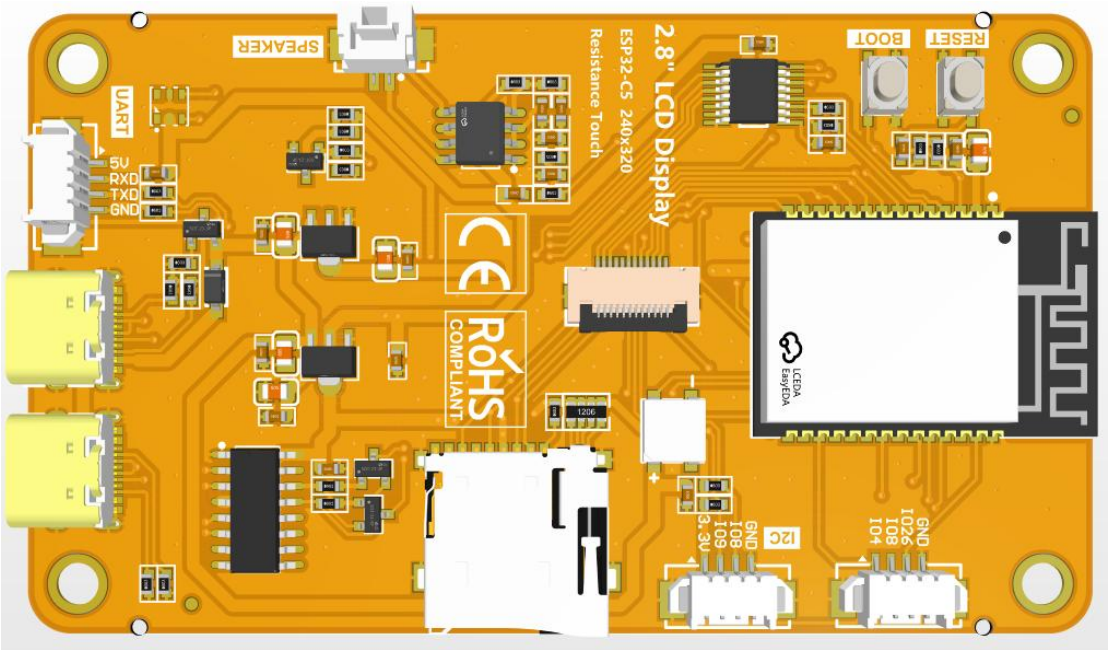

Figure 2.1 Back of the 2.8-inch ESP32-C5 TN display module

The 2.8-inch ESP32-C5 display module uses the ESP32-C5 as its MCU. The GPIO assignments for onboard peripherals are shown in the table below:

ESP32-C5 Chip Pin Assignment Description			
Onboard Device	**Device Pin**	**ESP32-C5 Connection Pin**	**Description**
LCD	**TFT_CS**	IO23	LCD chip select control signal, active low
	TFT_RS	IO24	LCD command/data selection control signal. High level: data; Low level: command
	TFT_SCK	IO6	LCD SPI bus clock signal
	TFT_MOSI	IO7	LCD SPI bus write data signal

	TFT_MISO	IO2	LCD SPI bus read data signal
	TFT_RST	EN	LCD reset control signal, active low reset (shared with ESP32-C5 MCU reset pin)
	TFT_BL	IO25	LCD backlight control signal (high level: backlight on; low level: backlight off)
RTP	TP_CLK	IO6	Touch screen SPI bus clock signal
	TP_CS	IO1	Touch screen chip select control signal, active low
	TP_DIN	IO7	SPI MOSI (Master Output / Write)
	TP_DOUT	IO2	SPI MISO (Master Input / Read)
	TP_INT	IO3	Touch screen interrupt control signal
SDCARD	SD_CLK	IO6	SD card SPI bus clock signal
	SD_CS	IO10	SD card chip select control signal, active low
	SD_DIN	IO7	SPI MOSI (Master Output / Write)
	SD_DOUT	IO2	SPI MISO (Master Input / Read)
LED	RGB_INT	IO27	Single-wire RGB tri-color LED; can independently control the red, green, and blue LEDs via different signals
Audio	Audio_DO	IO26	Audio output signal
KEY	BOOT_KEY	IO28	Download mode selection button (hold this button during power-on, then release to enter download mode)
	RESET_KEY	EN	ESP32-C5 reset button, active low reset (shared with LCD reset)
USB	USB_N	IO13	USB bus differential signal data line negative
	USB_P	IO14	USB bus differential signal data line positive
Serial Port	RX0	IO12	ESP32-C5 UART0 receive signal interface
	TX0	IO11	ESP32-C5 UART0 transmit signal interface
LP_Serial	LP_UART_RXD	IO4	Extended external LP_UART receive signal interface
	LP_UART_TXD	IO5	Extended external LP_UART transmit signal interface
I2C	I2C_SDA	IO8	Extended external I2C data signal interface
	I2C_SCL	IO9	Extended external I2C clock signal interface

Table 2.1 ESP32-C5 Onboard Peripheral Pin Assignment Description

3. Sample Program Usage Instructions

3.1. Setting Up ESP32 Arduino Development Environment

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

3.2 Installing Third-Party Libraries

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

A. Open the "Install libraries" directory in the "2.8inch_ESP32-C5 Example Code" folder from the resource package to find the third-party libraries, as shown in the figure below:

Figure 3.1 Third-party libraries for the sample programs

The libraries include:

Adafruit_Neopixel: Library providing control functions for RGB LEDs.

ArduinoJson: C++ JSON library for Arduino and IoT applications.

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

TFT_Touch: Resistive touch screen driver library.

HttpClient: HTTP client library for interacting with Arduino web servers.

lvgl: Highly customizable, low-resource, visually appealing, and easy-to-use embedded graphics library.

NTPClient: NTP client library for connecting to NTP servers.

TFT_eSPI: Arduino graphics library for TFT-LCD screens, supporting multiple platforms and LCD driver ICs.

Time: Library providing timing functions for Arduino.

JPEGDEC: JPG format image decoding library that can decode JPG files and display them on the LCD.

B. Copy these libraries to the library directory of your project folder. The default project folder library directory is "C:\Users\Administrator\Documents\Arduino\libraries" (replace "Administrator" with your actual computer username). If you have modified the project folder path, copy them to the modified project folder library directory.

C. After installing the third-party libraries, you can open and use the sample programs.

The lvgl and TFT_eSPI libraries in the third-party library collection need to be configured before use. The libraries in the resource package have already been configured and can be used directly.

3.3. Sample Program Usage Instructions

The sample programs are located in the "Demo" directory of the "2.8inch_ESP32-C5 Example Code" folder in the resource package, as shown in the figure below:

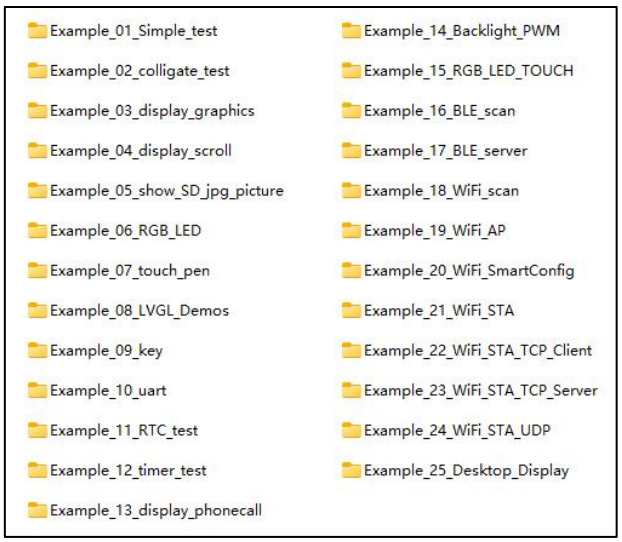

Figure 3.10 Sample programs

After opening the example, select the ESP32C5 board in Tools -> Board:

Figure 3.11

Then select the connected port， Flash Size -> 16MB； Partition Scheme -> 16M Flash(3MB APP/9.9MB FATFS):

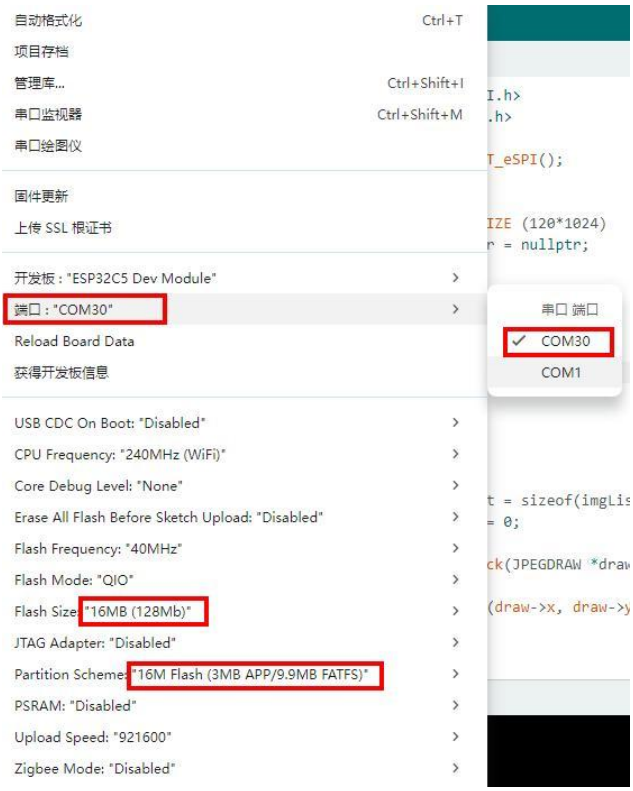

Figure 3.12

The descriptions of each sample program are as follows:

01_Simple_test

This is a basic sample program that does not depend on any third-party libraries. The hardware requires an LCD display screen. The display content includes full-screen color fill and random rectangle fill. This example can be used directly to check whether the display screen is working properly.

02_colligate_test

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen. The display content includes drawing points, lines, various graphics, and runtime statistics. It is a relatively comprehensive display example.

03_display_graphics

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen. The display content includes various graphic drawings and fills.

04_display_scroll

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen. The display content includes text and image display, scrolling text display, inverted color display, and four-direction rotation display.

05_show_SD_jpg_picture

This example depends on the TFT_eSPI and JPEGDEC libraries. The hardware requires an LCD display screen and a MicroSD card. This example reads JPG images from the MicroSD card, parses them, and displays the images on the LCD. The usage steps are as follows:

A. Copy the JPG images from the "PIC_320x480" directory in the example folder to the root directory of the MicroSD card via computer.

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

C. Power on the display module, compile and download this example program, and you will see images cycling on the LCD screen. Note that you need to set Tools->PASRAM to Enabled, otherwise it will not display.

06_RGB_LED

This example depends on the Adafruit_Neopixel library. The hardware requires an RGB tri-color LED. This example demonstrates cycling through the red, green, and blue colors of the RGB tri-color LED using a button.

07_touch_pen

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and a resistive touch screen. This example demonstrates drawing lines by touching the screen, which can be used to test whether the touch screen function is working properly.

08_LVGL_Demos

This example depends on the TFT_eSPI and lvgl libraries. The hardware requires an LCD display

screen and a resistive touch screen. This example demonstrates 5 built-in demos of the lvgl embedded UI system. Through this example, you can learn how to port lvgl to the ESP32 platform and how to configure the display screen, touch screen, and other underlying devices. Only one demo can be compiled at a time in the example program. Uncomment the demo you want to compile, and comment out the other demos, 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();
```

Figure 3.13 Select lvgl demo

- lv_demo_widgets: Various widget test demo
- lv_demo_benchmark: Performance benchmark test demo
- lv_demo_keypad_encoder: Keypad 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, approximately 15 minutes.

09_key

This example depends on the Adafruit_Neopixel library. The hardware requires a BOOT button and an RGB tri-color LED. This example demonstrates detecting button events and controlling the RGB tri-color LED on/off through button operations.

10_uart

This example depends on the TFT_eSPI library. The hardware requires a serial port and an LCD display screen. This example demonstrates the ESP32 interacting with a computer via serial port. The ESP32 sends messages to the computer via serial port, and the computer also sends messages to the ESP32 via serial port. After receiving, the ESP32 displays the messages on the LCD screen.

11_RTC_test

This example depends on the TFT_eSPI and ESP32Time libraries. The hardware requires an LCD display screen. This example demonstrates using the ESP32 RTC module to set real-time and date, and displaying the time and date on the LCD screen.

12_timer_test

This example depends on the Adafruit_Neopixel library. The hardware requires an RGB tri-color LED. This example demonstrates using the ESP32 timer, setting a 1-second timer to control the green LED on/off.

13_display_phonecall

This example depends on the TFT_eSPI and TFT_Touch libraries. The hardware requires an LCD display screen and a resistive touch screen. This example demonstrates a simple phone dialing interface, where content is input through touch buttons.

14_Backlight_PWM

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and a resistive touch screen. This example demonstrates adjusting the display screen backlight brightness through touch sliding operations on the display module, while displaying the brightness value changes.

15_RGB_LED_TOUCH

This example depends on the TFT_eSPI and Adafruit_Neopixel libraries. The hardware requires an LCD display screen, a resistive touch screen, and an RGB tri-color LED. This example demonstrates controlling the RGB LED on/off, blinking, and brightness adjustment through touch buttons.

16_BLE_scan

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 Bluetooth module. This example demonstrates the ESP32 Bluetooth module scanning surrounding BLE Bluetooth devices and displaying the names and RSSI of scanned named BLE Bluetooth devices on the LCD screen.

17_BLE_server

The hardware requires an LCD display screen and an ESP32 Bluetooth module. This example demonstrates the ESP32 Bluetooth module creating a Bluetooth BLE server, being connected by a Bluetooth BLE client, and communicating. The usage steps are as follows:

A. Install a Bluetooth BLE debugging tool on your phone, such as "BLE Debugging Assistant", "LightBlue", etc.

B. Power on the display module, compile and download the example program. You can see the

Bluetooth BLE client running prompt on the display screen. If you want to modify the Bluetooth BLE server device name, you can modify it in the "BLEDevice::init" function parameter in the example program, as shown in the 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 c
```

Figure 3.14 Set the BLE server device name

C. Turn on Bluetooth on your phone and open the Bluetooth BLE debugging tool. Search for the Bluetooth BLE server device name (default: "ESP32_BT_BLE"), then click the name to connect. After successful connection, the ESP32 display module will show a prompt. You can then start Bluetooth communication.

18_WiFi_scan

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates the ESP32 WiFi module scanning surrounding wireless network information in STA mode and displaying the scanned wireless network information on the LCD screen. The wireless network information includes SSID, RSSI, CHANNEL, ENC_TYPE, etc. After the wireless network information scan is complete, the number of scans will be displayed, with a maximum of the first 17 scanned wireless network entries shown.

19_WiFi_AP

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates setting the ESP32 WiFi module to AP mode for WiFi terminals to connect. The display screen will show the SSID, password, host IP address, host MAC address, and other information set in AP mode. Once a terminal connects successfully, the display screen will show the number of connected terminals. Set the SSID and password yourself 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";
```

Figure 3.15 Set SSID and password in AP mode

20_WiFi_SmartConfig

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen, an ESP32 WiFi module, and a BOOT button. This example demonstrates the ESP32 WiFi module in STA mode performing smart network configuration via the EspTouch mobile app. The flow chart of the entire example program is as

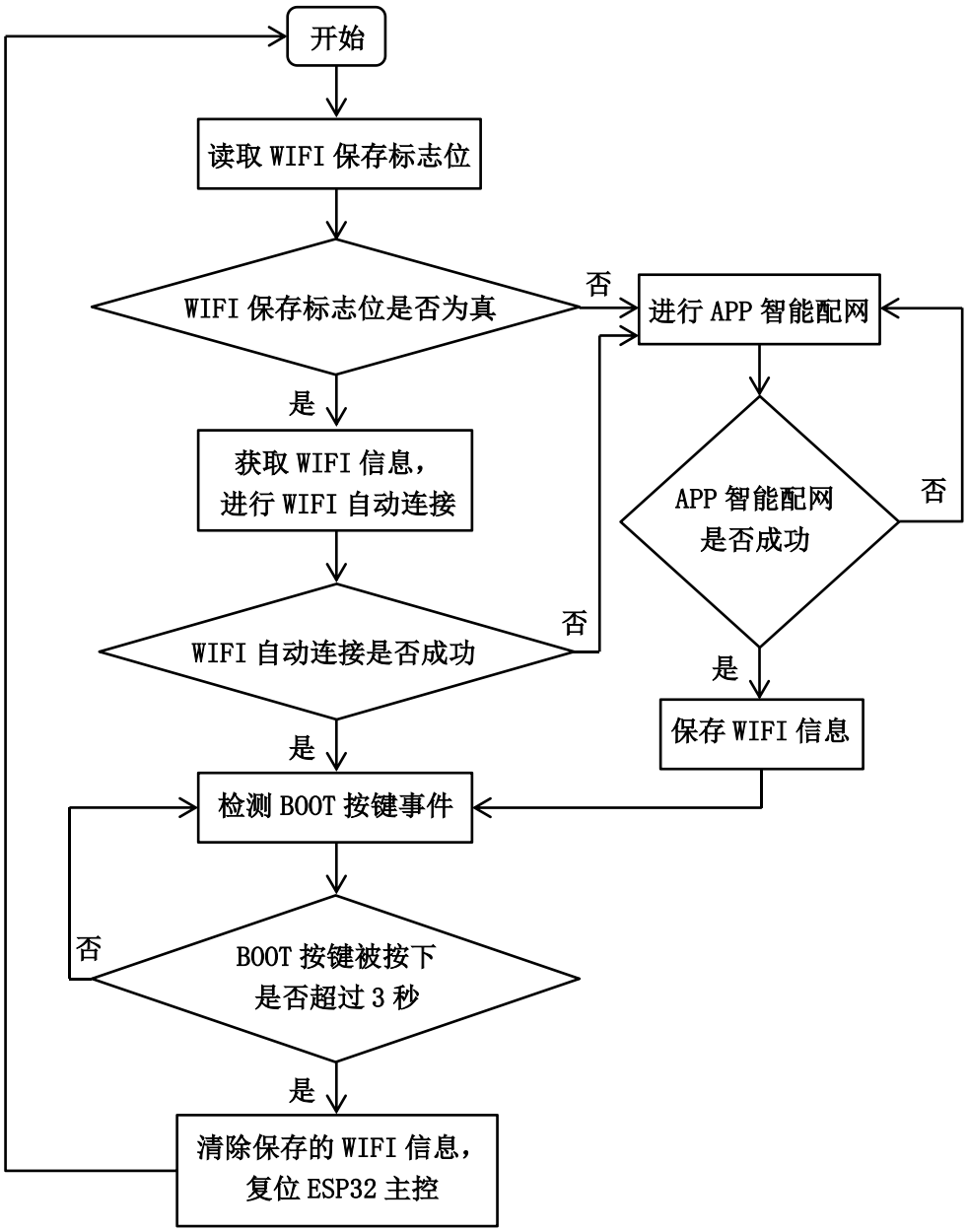

Figure 3.16 Smart network configuration example program flow chart

A. Download the EspTouch application on your phone, or copy the installer "esptouch-v2.0.0.apk" from the "7-Tool Software_Tool_software" folder in the resource package (only the Android installer is available; the iOS application can only be installed from the device).

You can also download the installer from the official website at:

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

B. Power on the display module, compile and download the example program. If the ESP32 does not have any WiFi information saved, it will directly enter smart network configuration mode. At this point, open the EspTouch application on your phone, enter the SSID and password of the WiFi your phone is connected to, and then broadcast the relevant information via UDP. Once the ESP32 receives this information, it will connect to the network using the SSID and password in the information. After successful network connection, the SSID, password, IP address, and MAC address will be displayed on the screen, and the WiFi information will be saved. Note that the success rate of this configuration method is not very high; if it fails, you may need to try multiple times.

C. If the ESP32 has saved WiFi information, it will automatically connect to the network using the saved WiFi information on startup. If the connection fails, it will enter smart network configuration mode. After successful network connection, long-press the BOOT button for more than 3 seconds to clear the saved WiFi information and reset the ESP32 to restart smart network configuration.

21_WiFi_STA

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates the ESP32 connecting to WiFi in STA mode using the provided SSID and password. The operation steps are as follows:

A. Write the WiFi information to connect 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
```

Figure 3.17 Write WiFi information

B. Power on the display module, compile and download the example program. You can see the ESP32 start connecting to WiFi on the display screen. If the WiFi connection is successful, the display screen will show a success prompt, SSID, IP address, MAC address, and other information.

If the connection time exceeds 3 minutes, the connection fails and a failure prompt is displayed.

22_WiFi_STA_TCP_Client

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates the ESP32, after connecting to WiFi in STA mode, acting as a TCP client connecting to a TCP server. The operation steps are as follows:

A. Write the WiFi information to connect, the TCP server IP address (computer IP address), and port number in the "ssid", "password", "serverIP", and "serverPort" 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";

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

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

Figure 3.18 Write WiFi and TCP server information (1)

B. Open a "TCP & UDP Test Tool" or "Network Debugging Assistant" or similar testing tool on your computer (installer packages are in the "7-Tool Software_Tool_software" directory of the resource package). Create a TCP server in the tool with the port number matching the one set in the example program.

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

23_WiFi_STA_TCP_Server

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates the ESP32, after connecting to WiFi in STA mode, acting as a TCP server being connected by a TCP client. The operation steps are as follows:

A. Write the WiFi information to connect and the TCP server port number in the "ssid", "password", and "port" 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};
int port = 10000;

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

Figure 3.19 Write WiFi and TCP server information (2)

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

C. Open a "TCP & UDP Test Tool" or "Network Debugging Assistant" or similar testing tool on your computer (installer packages are in the "7-Tool Software_Tool_software" directory of the resource package). Create a TCP client in the tool (note that the IP address and port number must match the content displayed on the screen), then start connecting to the server. If the connection is successful, there will be a corresponding prompt, and you can communicate with the server.

24_WiFi_STA_UDP

This example depends on the TFT_eSPI library. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates the ESP32, after connecting to WiFi in STA mode, acting as a UDP server being connected by a UDP client. The operation steps are as follows:

A. Write the WiFi information to connect and the UDP server port number in the "ssid", "password", and "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
```

Figure 3.20 Write WiFi and UDP server information

B. Power on the display module, compile and download the example program. You can see the ESP32 start connecting to WiFi on the display screen. If the WiFi connection is successful, the display screen will show a success prompt, SSID, IP address, MAC address, local port number, and other information, then create a UDP server and wait for UDP client connection.

C. Open a "TCP & UDP Test Tool" or "Network Debugging Assistant" or similar testing tool on your computer (installer packages are in the "7-Tool Software_Tool_software" directory of the resource package). Create a UDP client in the tool (note that the IP address and port number must match the content displayed on the screen), then start connecting to the server. If the connection is successful, there will be a corresponding prompt, and you can communicate with the server.

25_Desktop_Display

This example depends on the ArduinoJson, Time, HttpClient, TFT_eSPI, and NTPClient libraries. The hardware requires an LCD display screen and an ESP32 WiFi module. This example demonstrates a weather clock desktop that can display city weather information (including temperature, humidity, weather icons, and scrolling display of other weather information), current time and date, and an astronaut animation. Weather information is obtained from a weather network, and time information is updated from an NTP server. The usage steps are as follows:

A. Write the WiFi information to connect in the "ssid" and "passwd" variables at the beginning of the example program, as shown in the figure below. If not set, smart network configuration will be performed (for smart network configuration instructions, please refer to the smart network configuration example program).

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

Figure 3.21 Set WiFi information

B. Power on the display module, compile and download the example program. You can see the weather clock desktop on the display screen.