Welcome to the JCZN Workshop!

·····Table of contents·····

Getting Started

Introduction

The objective of this post is to explain how to upload an Arduino program to the ESP32-2432S028R module, from JCZN .

http://www.jczn1688.com/filedownload/534855

The ESP32 WiFi and Bluetooth chip is the latest generation of Espressif products. It has a dual-core 32-bit MCU, which integrates WiFi HT40 and Bluetooth/BLE 4.2 technology inside.
ESP wroom 32 has a significant performance improvement. It is equipped with a high-performance dual-core Tensilica LX6 MCU. One core handles high speed connection and the other for standalone application development. The dual-core MCU has a 240 MHz frequency and a computing power of 600 DMIPS.
In addition, it supports Wi-Fi HT40, Classic Bluetooth/BLE 4.2, and more GPIO resources.

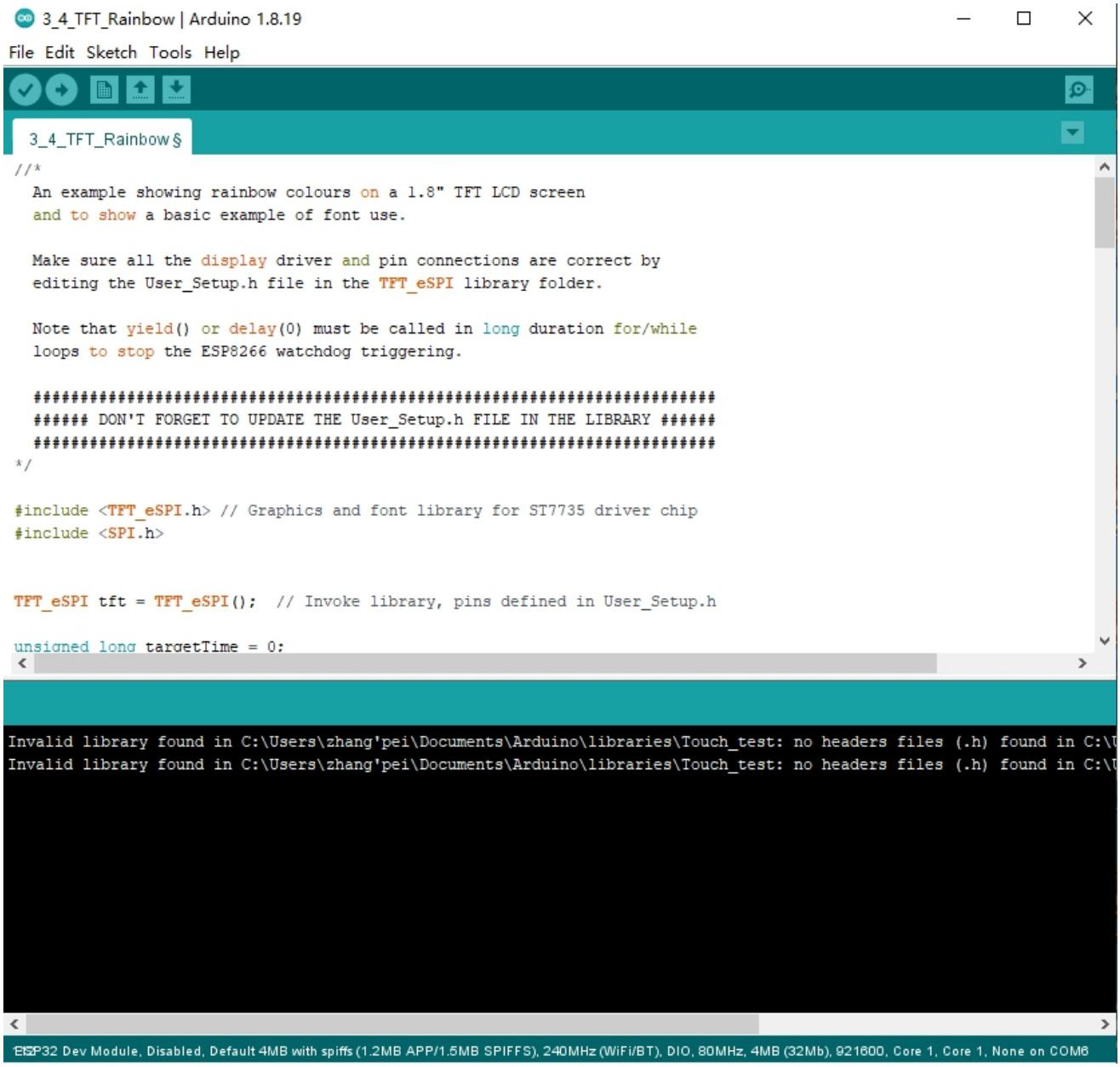
Installing using Arduino IDE

Programming the ESP32
An easy way to get started is by using the familiar Arduino IDE. While this is not necessarily the best environment for working with the ESP32, it has the advantage of being a familiar application, so the learning curve is flattened.
We will be using the Arduino IDE for our experiments.

1, Installing using Arduino IDE
we first need to install version 1.8.19 of the Arduino IDE (or greater),for example, the Arduino installation was in “C/Programs(x86)/Arduino”.
download release link:
https://downloads.arduino.cc/arduino-1.8.19-windows.exe

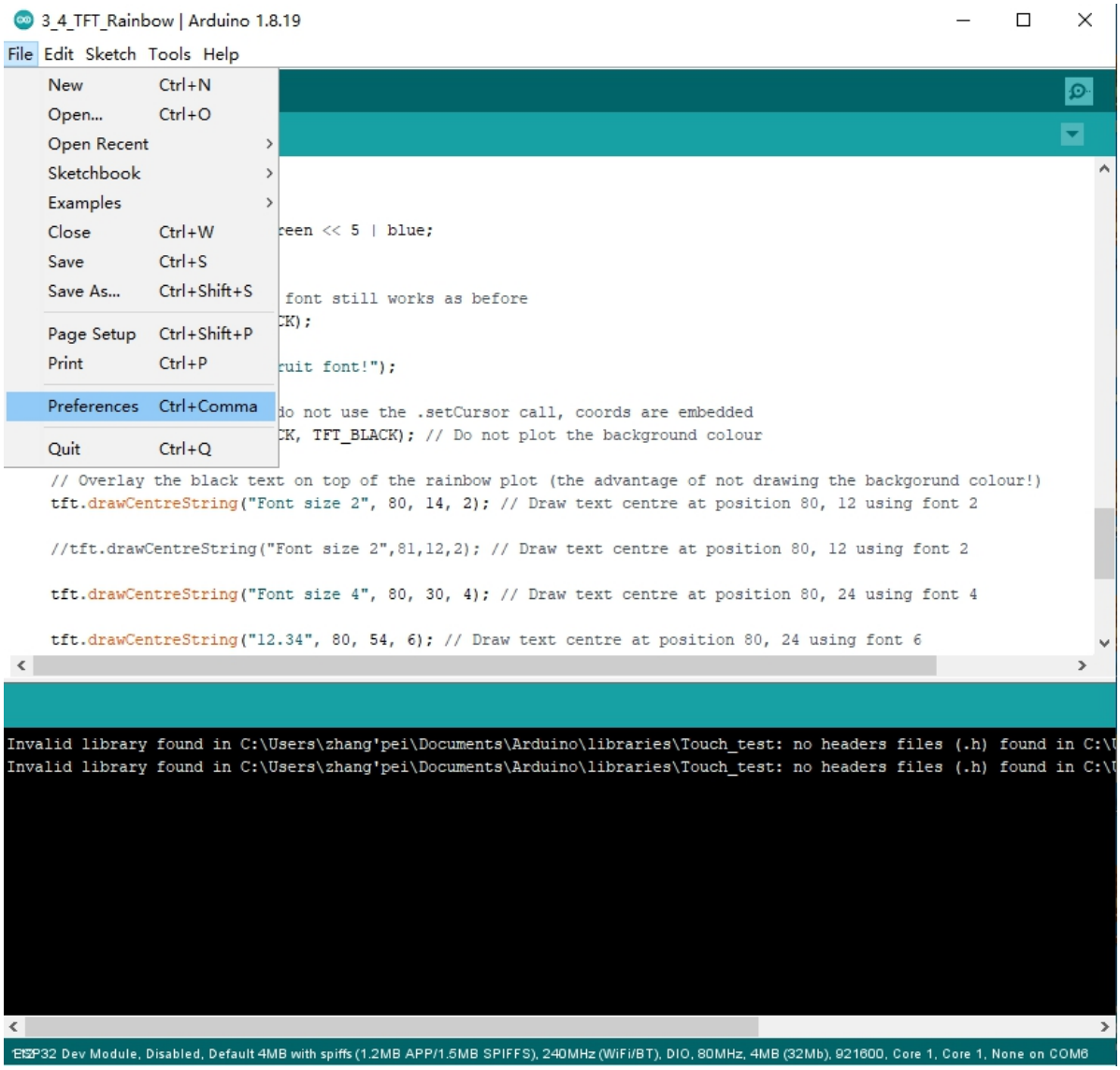
2, This is the way to install Arduino-ESP32 directly from the Arduino IDE.

Add Boards Manager Entry
Here is what you need to do to install the ESP32 boards into the Arduino IDE:
(1) Open the Arduino IDE.



(2) Click on the File menu on the top menu bar.

(3) Click on the Preferences menu item. This will open a Preferences dialog box.



（4）You should be on the Settings tab in the Preferences dialog box by default.

（5）Look for the textbox labeled “Additional Boards Manager URLs”.

（6）If there is already text in this box add a coma at the end of it, then follow the next step.

（7）Paste the following link into the text box ：

Stable release link:

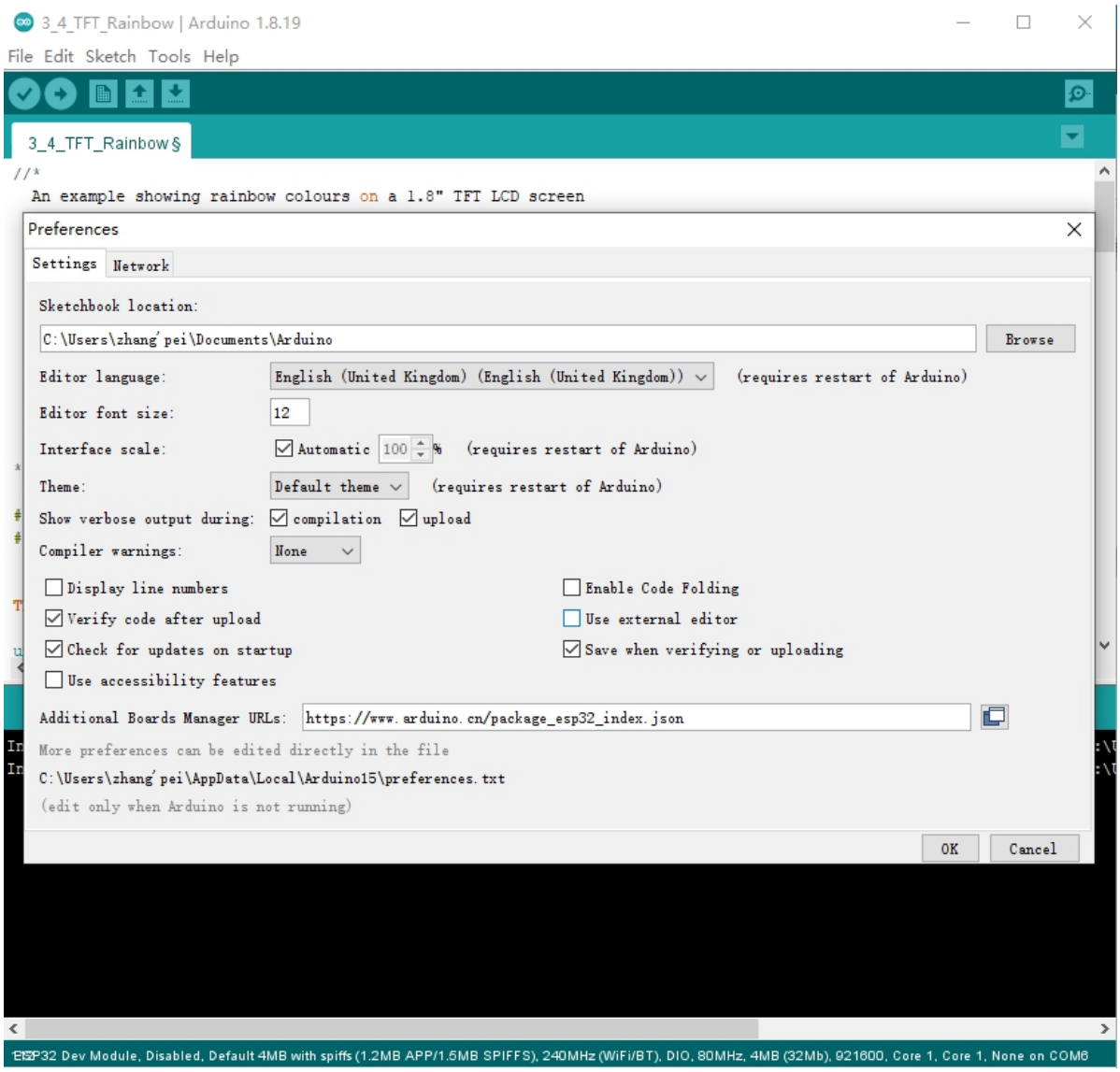
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

Development release link:

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

（8）Click the OK button to save the setting.

The textbox with the JSON link in it is illustrated here:



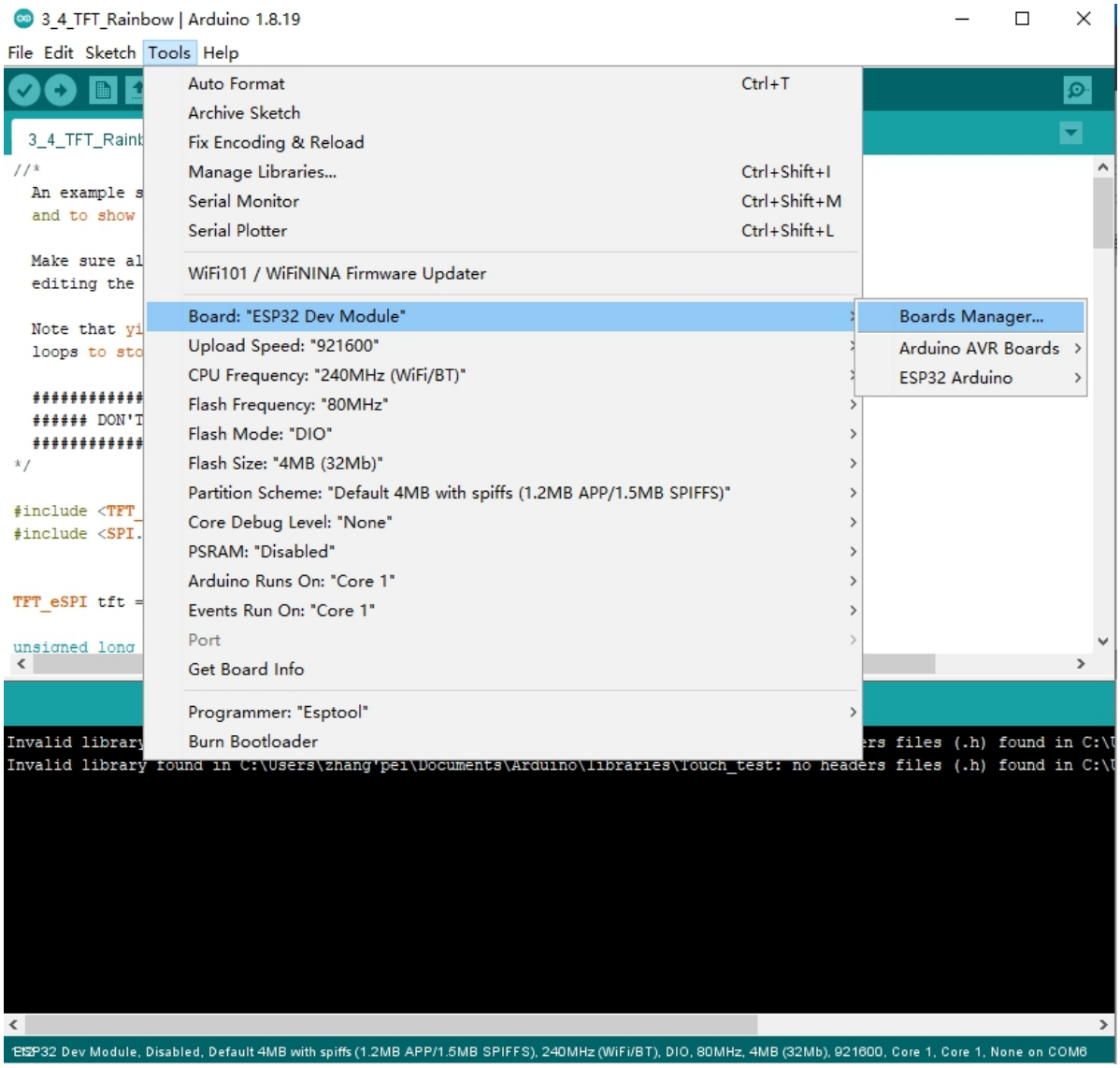
(9) In the Arduino IDE click on the Tools menu on the top menu bar.

(10) Scroll down to the Board: entry

(11) A submenu will open when you highlight the Board: entry.

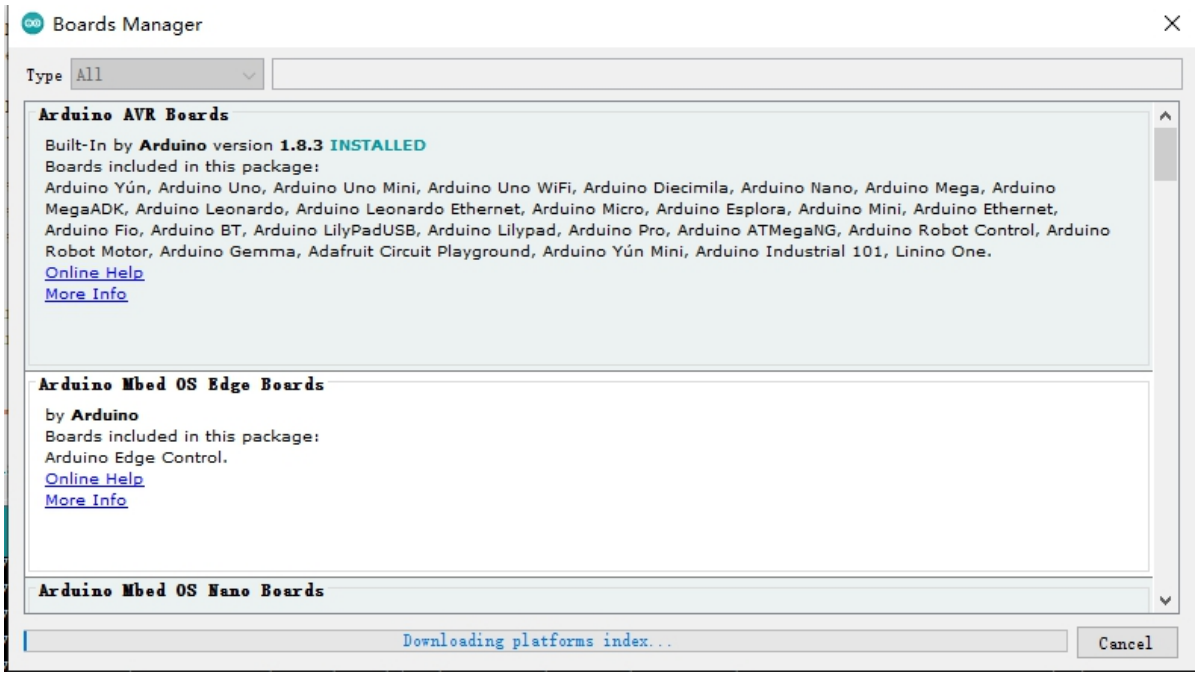
(12) At the top of the submenu is Boards Manager. Click on it to open the Boards Manager dialog box.

(13)I n the search box in the Boards Manager enter “esp32”.

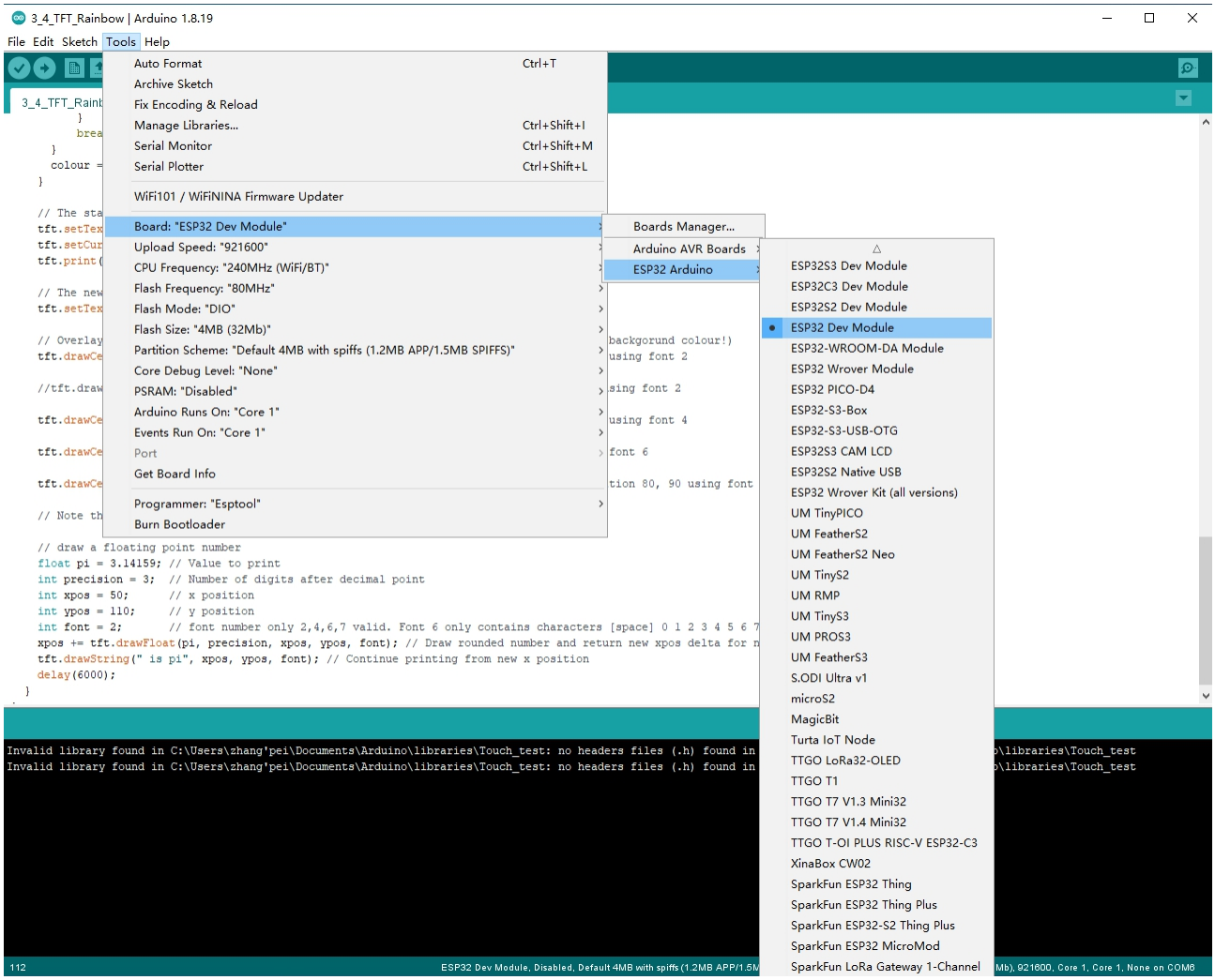


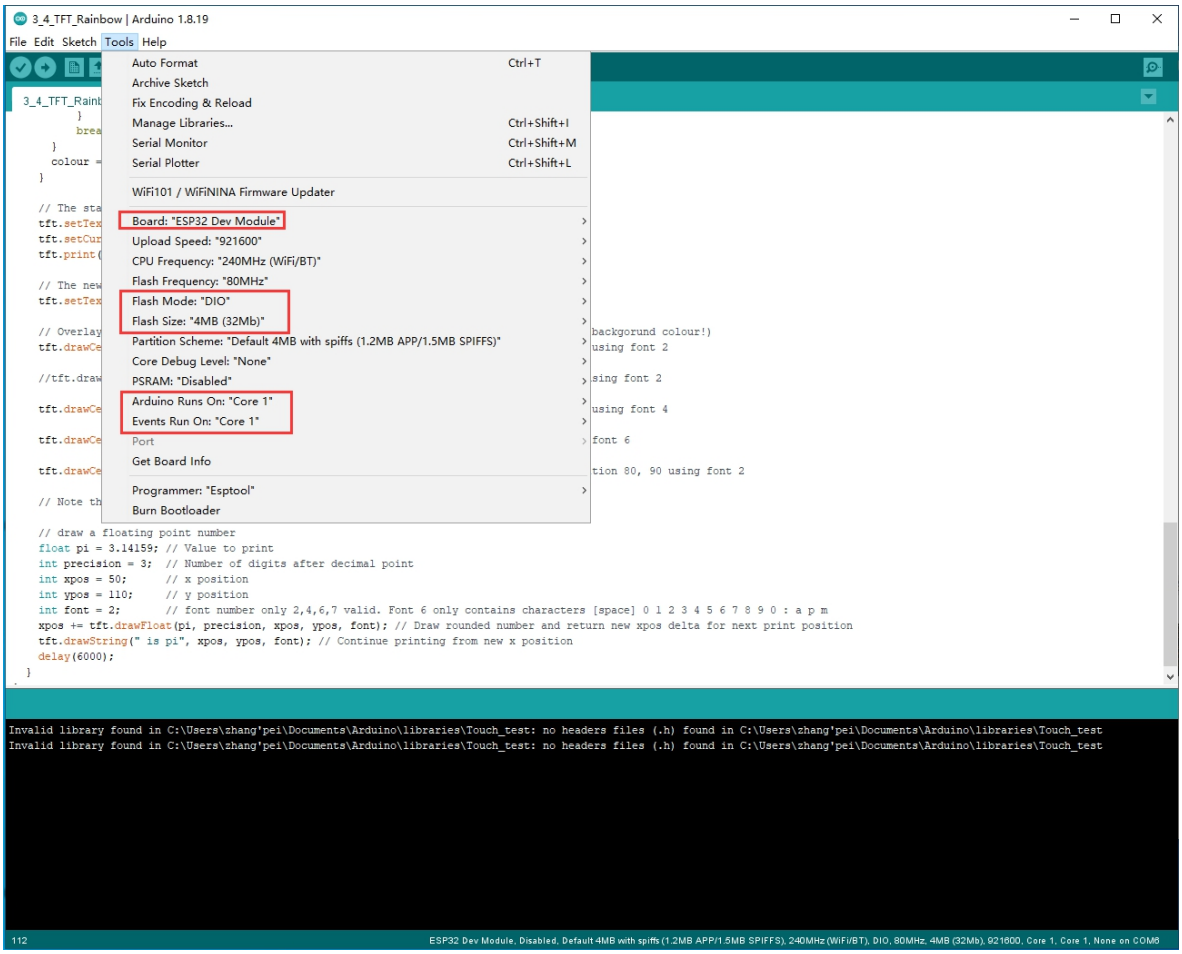
(14) You should see an entry for “esp32 by Espressif Systems”. Highlight this entry and click on the Install button.

This will install the ESP32 boards into your Arduino IDE

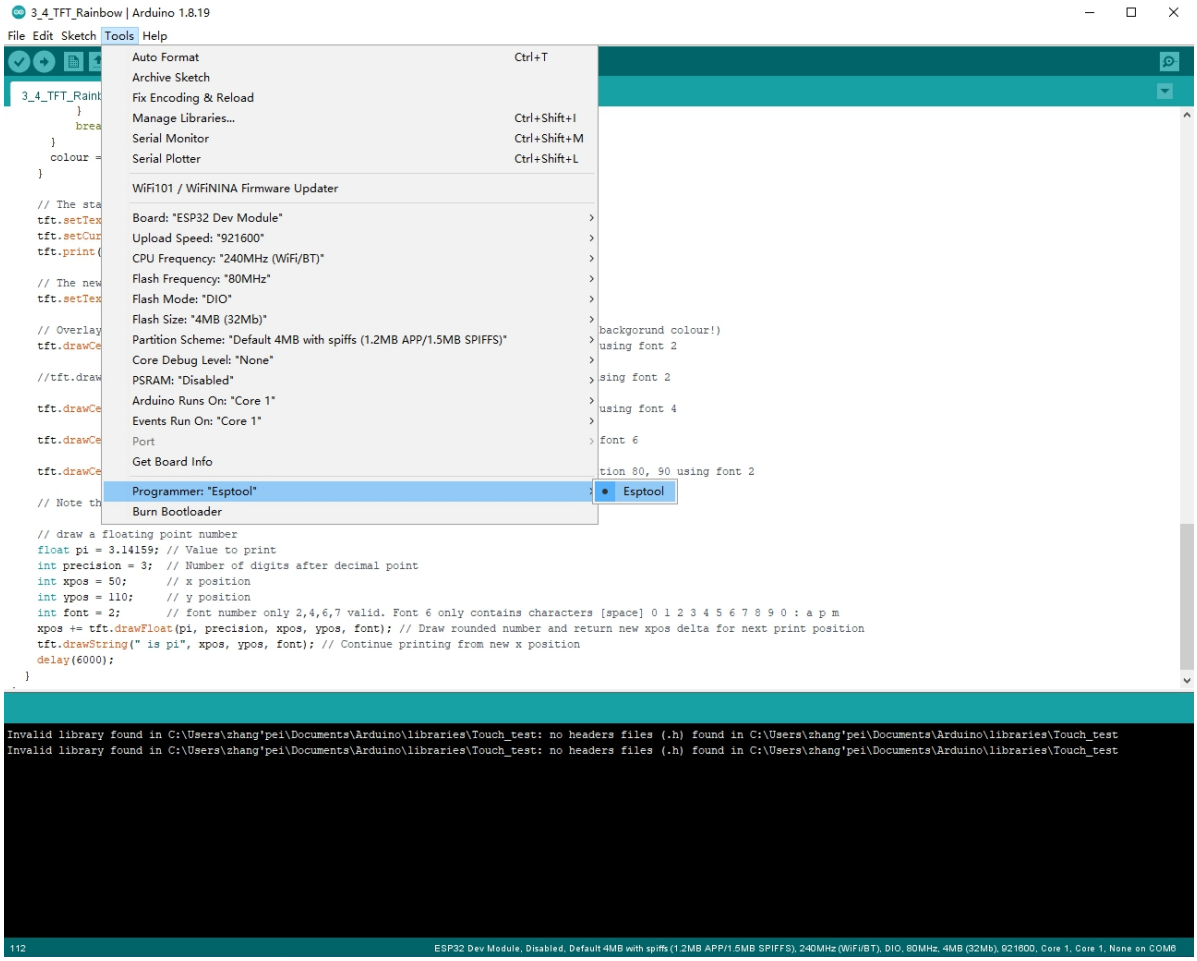


Once the installation completes, we need to select the correct board options for the "ESP32 Arduino" board. In the board type, in the tools tab, we choose “ESP32 Dev Module”.


3_4_TFT_Rainbow | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "ESP32 Dev Module"
Upload Speed: "921600"
CPU Frequency: "240MHz (WiFi/BT)"
Flash Frequency: "80MHz"
Flash Mode: "DIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
PSRAM: "Disabled"
Arduino Runs On: "Core 1"
Events Run On: "Core 1"
Port
Get Board Info
Programmer: "Esptool"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
ESP32 Arduino
ESP32S3 Dev Module
ESP32C3 Dev Module
ESP32S2 Dev Module
ESP32 Dev Module
ESP32-WROOM-DA Module
ESP32 Wrover Module
ESP32 PICO-D4
ESP32-S3-Box
ESP32-S3-USB-OTG
ESP32S3 CAM LCD
ESP32S2 Native USB
ESP32 Wrover Kit (all versions)
UM TinyPICO
UM FeatherS2
UM FeatherS2 Neo
UM TinyS2
UM RMP
UM TinyS3
UM PROS3
UM FeatherS3
S.ODI Ultra v1
microS2
MagicBit
Turta IoT Node
TTGO LoRa32-OLED
TTGO T1
TTGO T7 V1.3 Mini32
TTGO T7 V1.4 Mini32
TTGO T-OI PLUS RISC-V ESP32-C3
XinaBox CW02
SparkFun ESP32 Thing
SparkFun ESP32 Thing Plus
SparkFun ESP32-S2 Thing Plus
SparkFun ESP32 MicroMod
SparkFun LoRa Gateway 1-Channel
// draw a floating point number
float pi = 3.14159; // Value to print
int precision = 3; // Number of digits after decimal point
int xpos = 50; // x position
int ypos = 110; // y position
int font = 2; // font number only 2,4,6,7 valid. Font 6 only contains characters [space] 0 1 2 3 4 5 6 7
xpos += tft.drawFloat(pi, precision, xpos, ypos, font); // Draw rounded number and return new xpos delta for n
tft.drawString(" is pi", xpos, ypos, font); // Continue printing from new x position
delay(6000);
Invalid library found in C:\Users\zhang'pei\Documents\Arduino\libraries\Touch_test: no headers files (.h) found in
Invalid library found in C:\Users\zhang'pei\Documents\Arduino\libraries\Touch_test: no headers files (.h) found in
112
ESP32 Dev Module, Disabled, Default 4MB with spiffs (1.2MB APP/1.5M
Mb), 921600, Core 1, Core 1, None on COM6



Set and In the programmer entry of the same tab, we choose “esptool”.



It's important to note that after the code is uploaded, the device will start to run it. So, if we want to upload a new program, wee need to reset the power of the device, in order to guarantee that it enters flashing mode again.

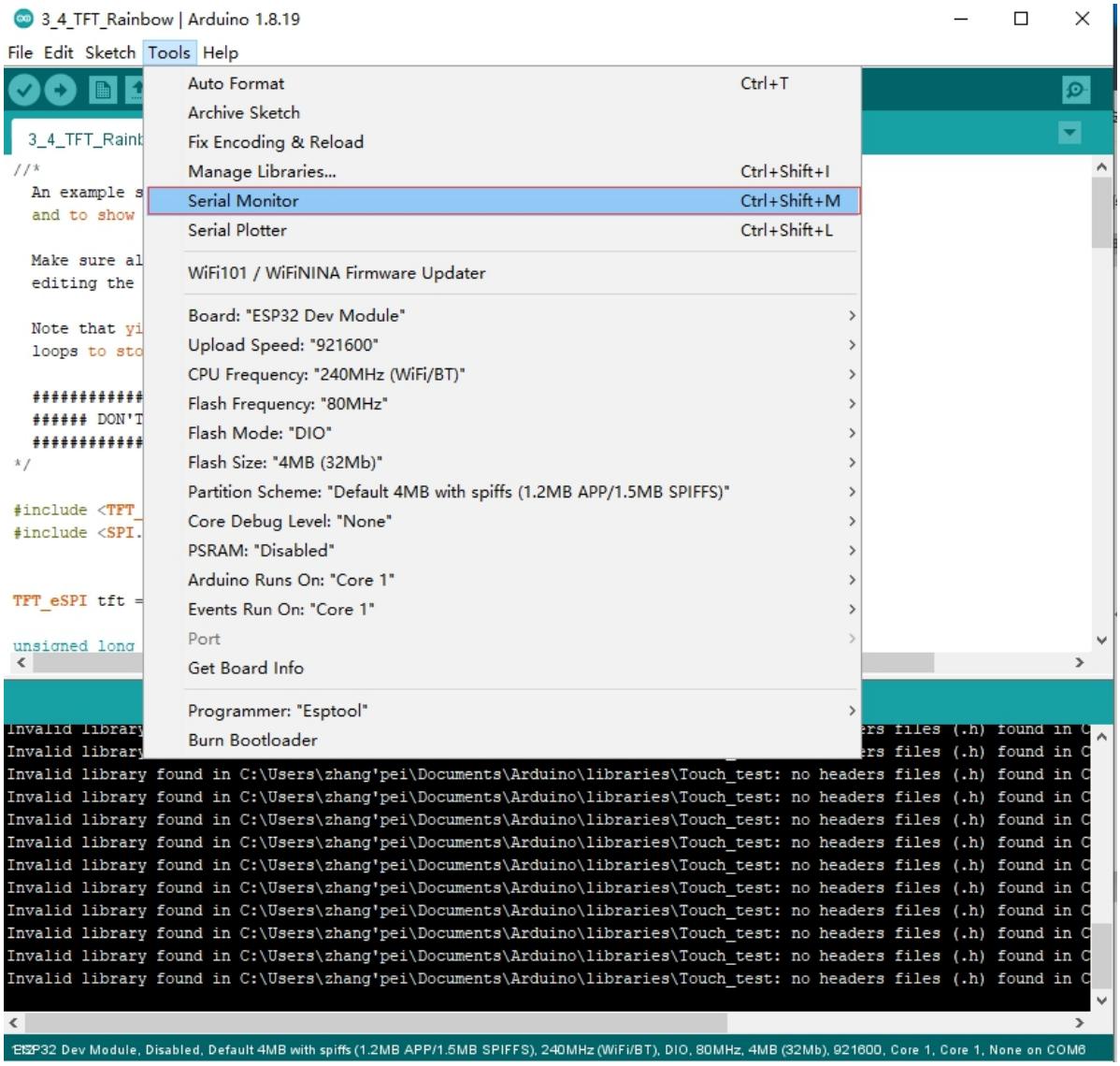
First program

Since this platform is based on Arduino, we can use many of the usual functions. As an example for the first program, the code bellow starts the Serial port and prints "hello from ESP32" every second.

```
void setup() {
  Serial.begin(115200);
}

void loop() {
  Serial.println("hello from ESP32");
  delay(1000);
}
```

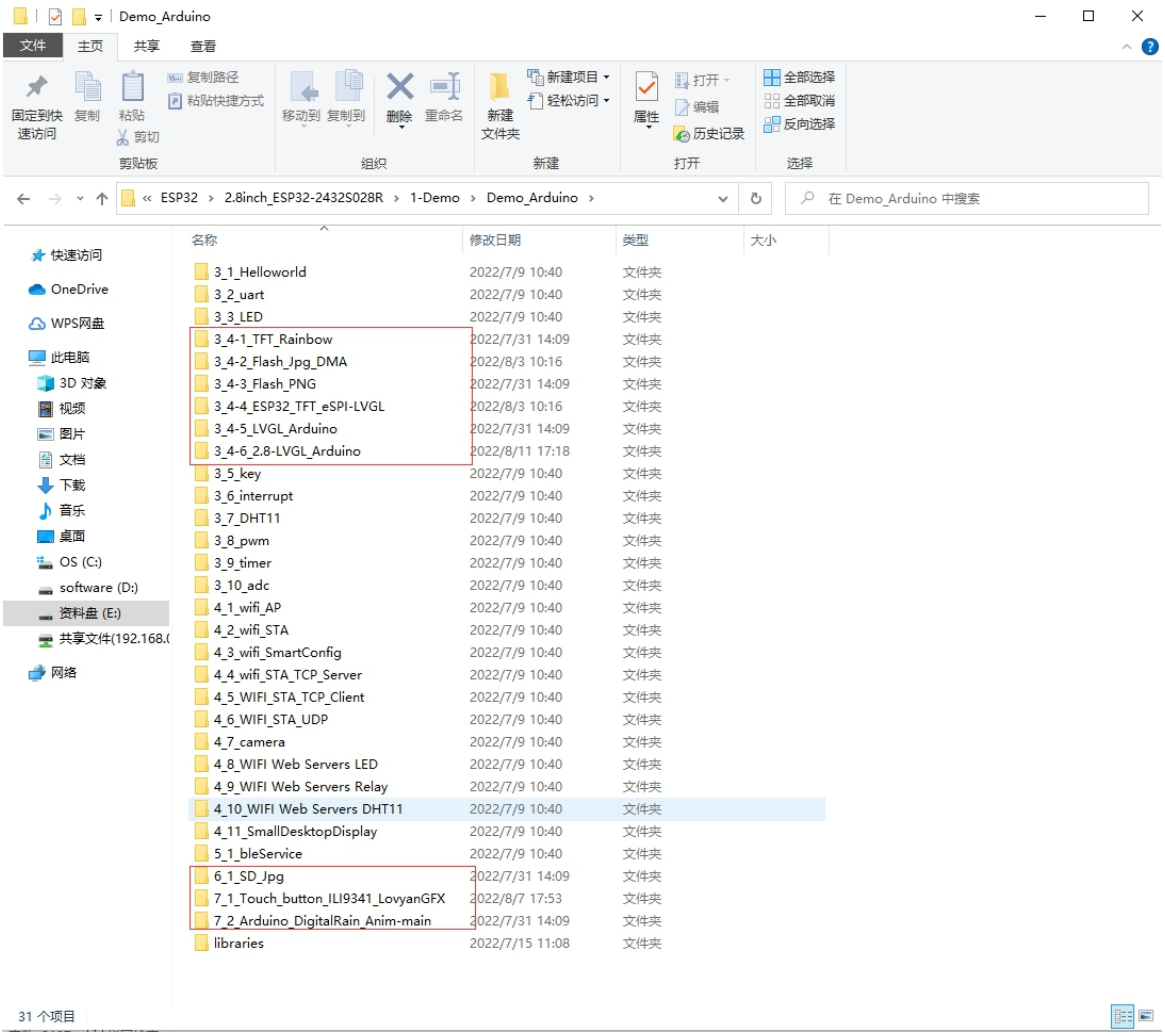
If everything is working fine, we will see the output in the serial console shown.



Again thank you for so much concern.. Hopefully, it's the beginning of a wonderful relationship!

Sample program usage

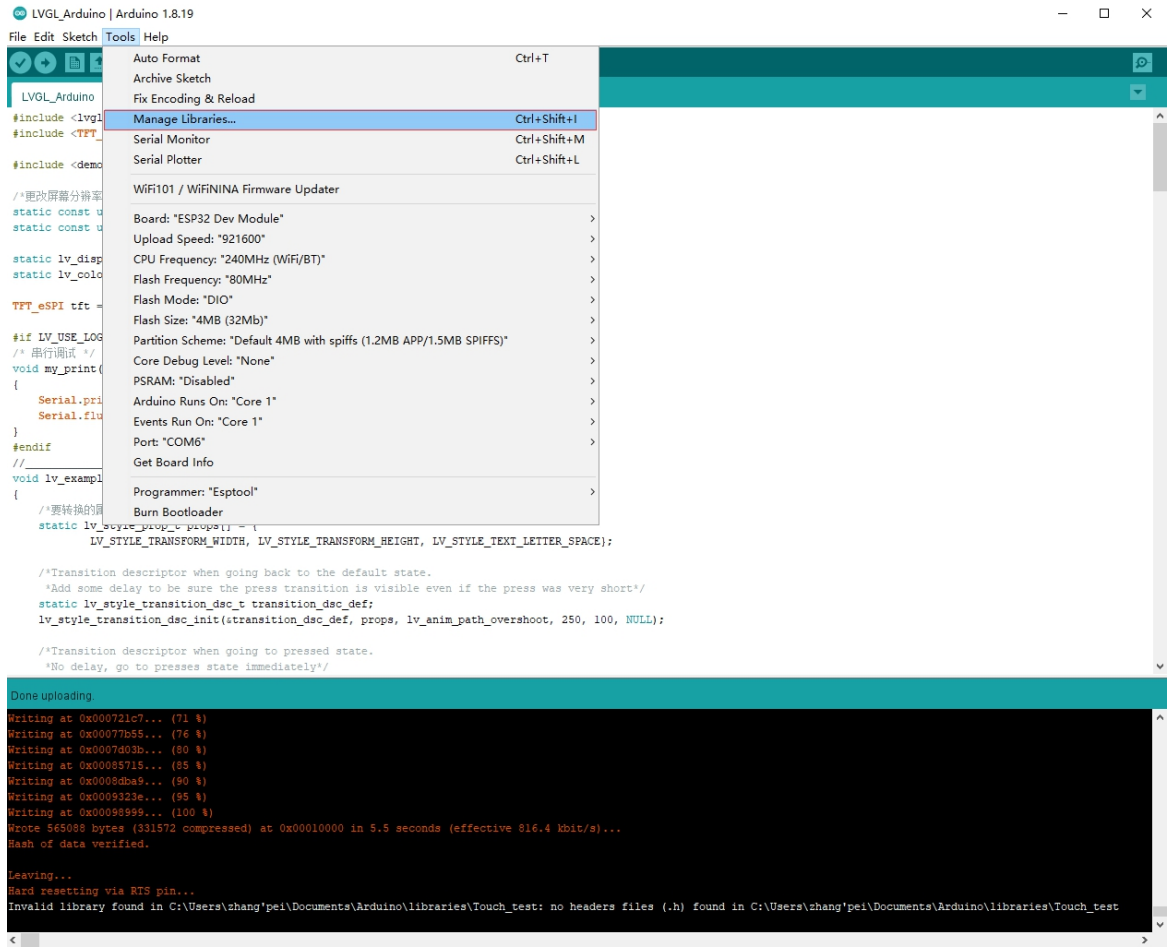
At present, only a preliminary explanation and introductory use are given to the samples displayed on the screen, and the corresponding examples in the data center are found, as shown in the figure:

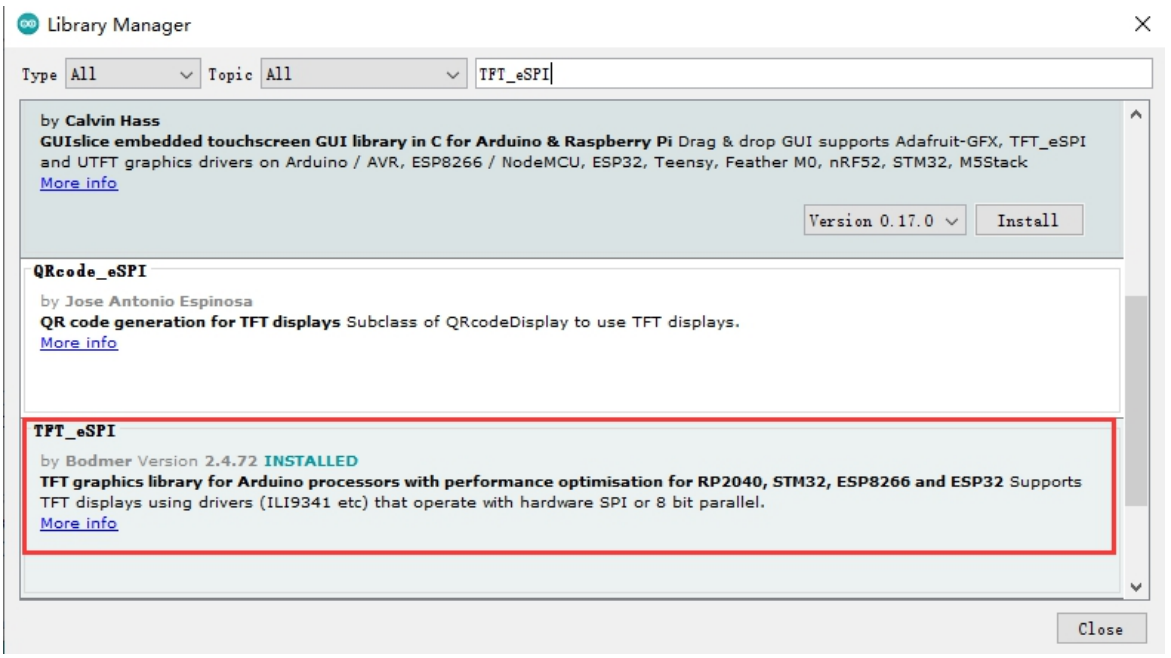


The examples in the red circle are all based on the TFT_eSPI library as the basic application. This library supports various commonly used driver chips, such as ST7735, ST7789, ILI9341, etc., and has good compatibility.

TFT_eSPI library file installation:

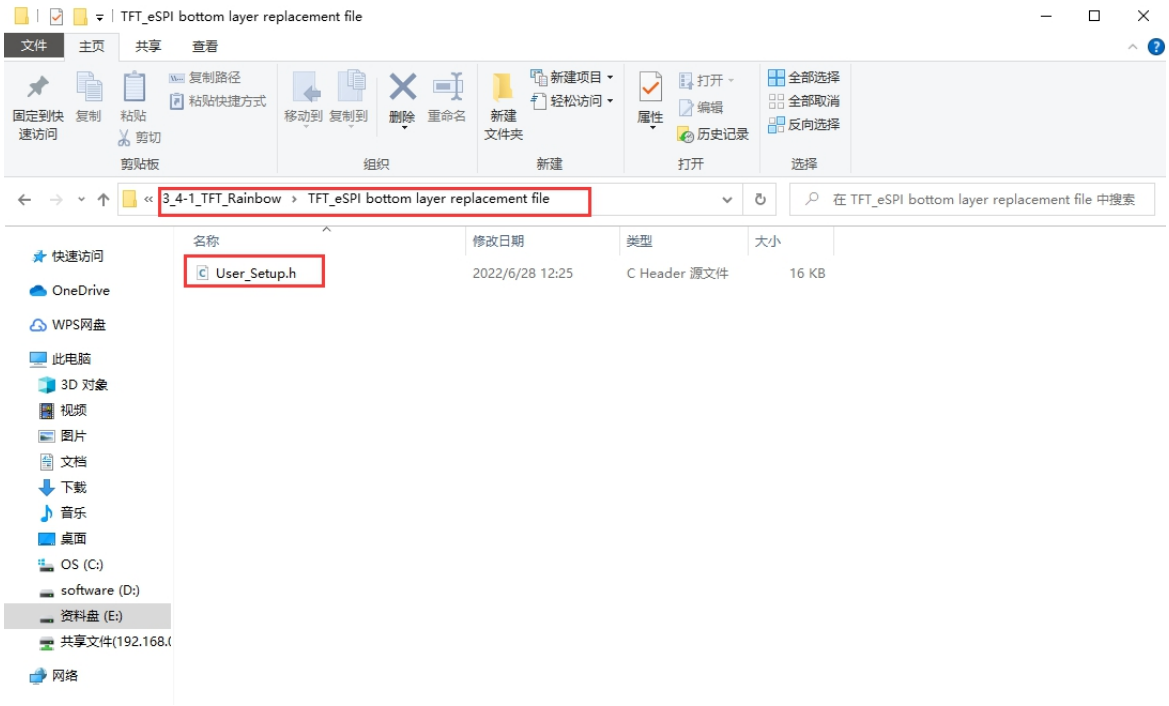
Open the library manager in Arduino, search for TFT_eSPI, and click instal .





Although the TFT_eSPI library has many advantages, it may also have a troublesome place for ordinary users, that is, after the installation

It needs to be configured separately, I already have a configured file, as shown in the figure:

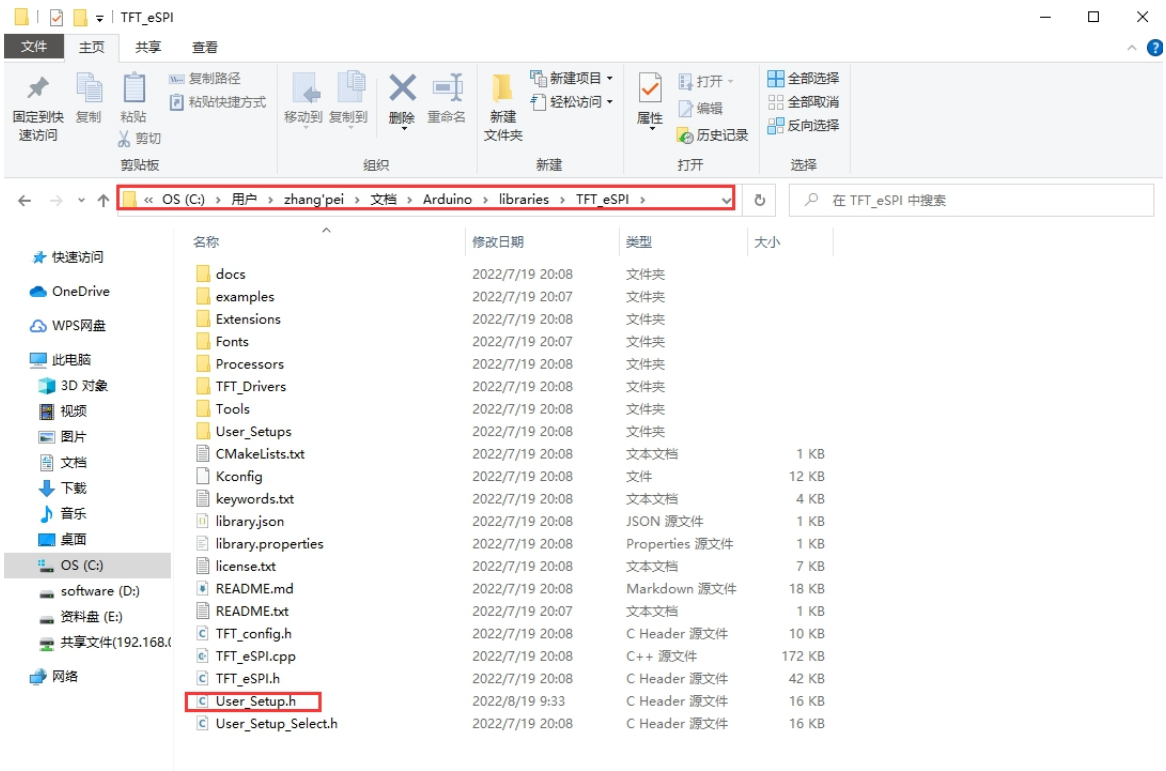


Copy the file and replace User_Setup.h in TFT_eSPI. The library location is generally C:\Users\<username>\sketchbook\libraries \TFT_eSPI .

If you want to take a closer look at the various setting options, you can follow my tutorial and set it up. Go to the Arduino library file installation directory and open the location of the TFT_eSPI library. Taking Windows as an example, the library installation directory is generally:

C:\Users\<username>\ sketchbook\libraries \TFT_eSPI .

As shown:



Then open the User_Setup.h file in the library file directory, and make corresponding settings according to your own screen type and driver chip type. Here is an example of the 2.8-inch ILI9341 TFT LCD color screen I used.

First, we open User_Setup.h.

Step 1: Modify the custom driver file. Among the many driver files, choose the one that suits your screen, and comment out the unused ones.

As shown:

```
// Only define one driver, the other ones must be commented out
//#define ILI9341_DRIVER       // Generic driver for common displays
#define ILI9341_2_DRIVER     // Alternative ILI9341 driver, see https://github.com/Bodmer/TFT_eSPI/issues/1172
//#define ST7735_DRIVER      // Define additional parameters below for this display
//#define ILI9163_DRIVER     // Define additional parameters below for this display
//#define S6D02A1_DRIVER
//#define RPI_ILI9486_DRIVER // 20MHz maximum SPI
//#define HX8357D_DRIVER
//#define ILI9481_DRIVER
//#define ILI9486_DRIVER
//#define ILI9488_DRIVER     // WARNING: Do not connect ILI9488 display SDO to MISO if other devices share the SPI bus (TFT SDO does NOT tristate when CS is high)
//#define ST7789_DRIVER      // Full configuration option, define additional parameters below for this display
//#define ST7789_2_DRIVER    // Minimal configuration option, define additional parameters below for this display
//#define R61581_DRIVER
//#define RM68140_DRIVER
//#define ST7796_DRIVER
//#define SSD1351_DRIVER
//#define SSD1963_480_DRIVER
//#define SSD1963_800_DRIVER
//#define SSD1963_800ALT_DRIVER
//#define ILI9225_DRIVER
//#define GC9A01_DRIVER
```

Set the width and height, for ILI9341, set the width and height.

As shown:

```
76
77  // For ST7789, ST7735, ILI9163 and GC9A01 ONLY, define the pixel width and height in portrait orientation
78  // #define TFT_WIDTH  80
79  // #define TFT_WIDTH  128
80  // #define TFT_WIDTH  128 // ST7789 240 x 240 and 240 x 320
81  #define TFT_WIDTH  240
82  //#define TFT_WIDTH  320
83  // #define TFT_HEIGHT 160
84  // #define TFT_HEIGHT 128
85  //#define TFT_HEIGHT 160 // ST7789 240 x 240
86  #define TFT_HEIGHT 320 // ST7789 240 x 320
87  // #define TFT_HEIGHT 240 // GC9A01 240 x 240 //#define TFT_HEIGHT 480
88  //#define TFT_HEIGHT 480 //
89
```

Step 2: Pin definition, comment out other definitions, define your own pins .

As shown:

```
209  #define TFT_MISO 12
210  #define TFT_MOSI 13 // In some display driver board, it might be written as "SDA" and so on.
211  #define TFT_SCLK 14
212  #define TFT_CS   15  // Chip select control pin
213  #define TFT_DC    2  // Data Command control pin
214  #define TFT_RST  -1  // Reset pin (could connect to Arduino RESET pin)
215  #define TFT_BL   21  // LED back-light
216
217  #define TOUCH_CS 33     // Chip select pin (T_CS) of touch screen
218
```

Step 3: Turn on the Backlight Pins .

As shown:

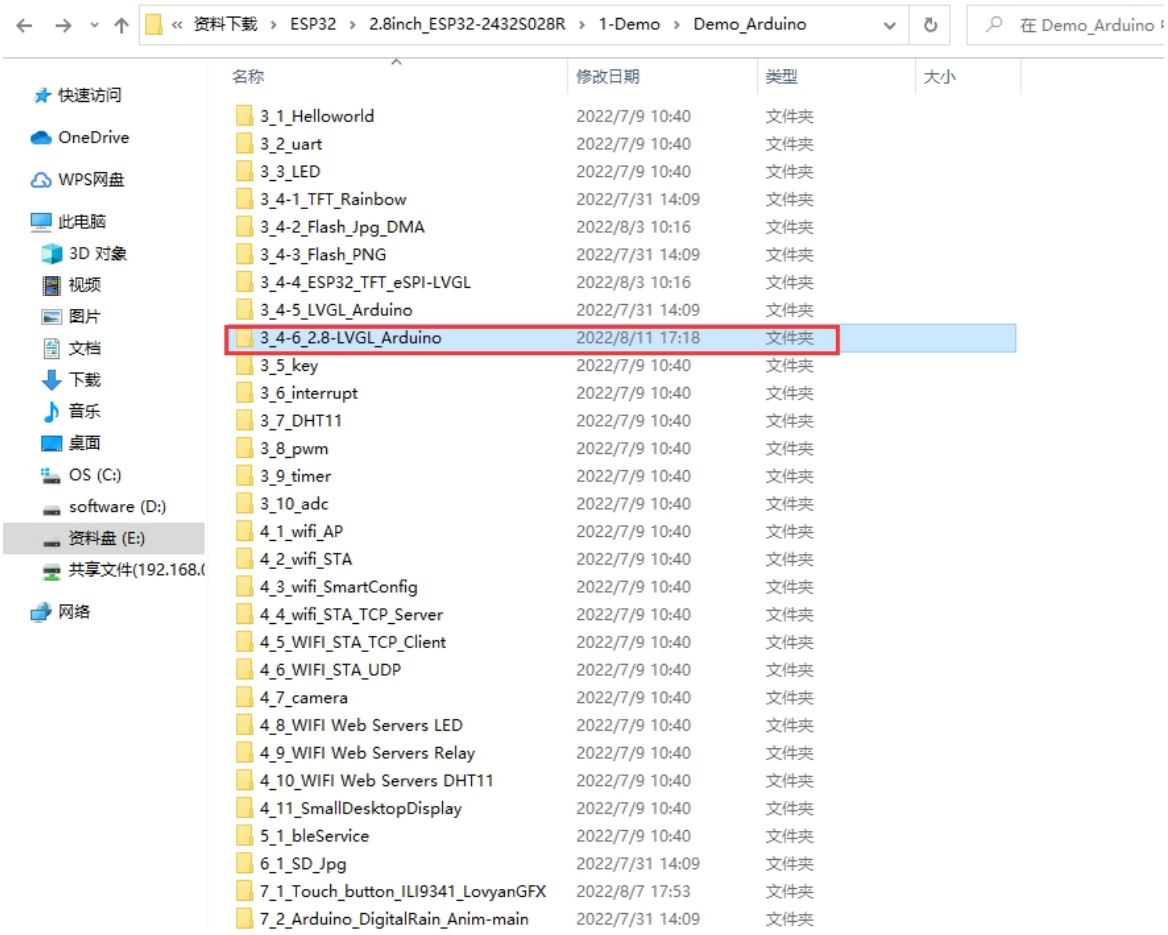
```
125
126  #define TFT_BL   21            // LED back-light control pin
127  #define TFT_BACKLIGHT_ON HIGH  // Level to turn ON back-light (HIGH or LOW)
128
129
```

After configuring these, compile the arduino function in 3_4-1_TFT_Rainbow to light up the screen .

About the use of touch and LVGL:

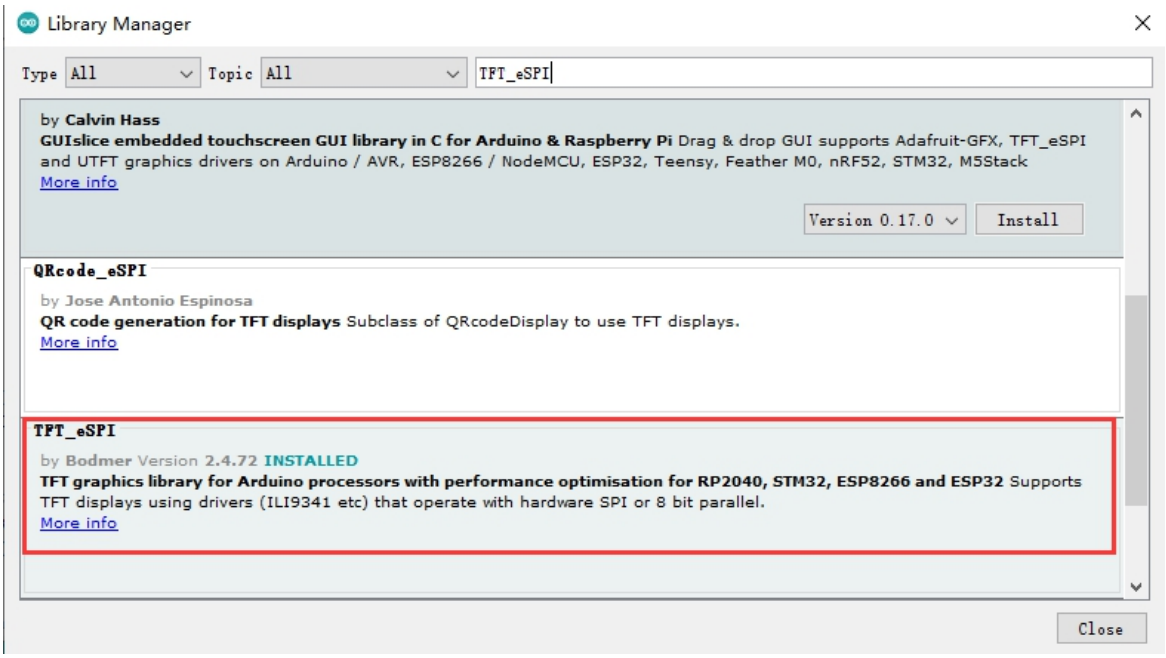
Find the data center 3_4-6_2.8-LVGL_Arduino

As shown:

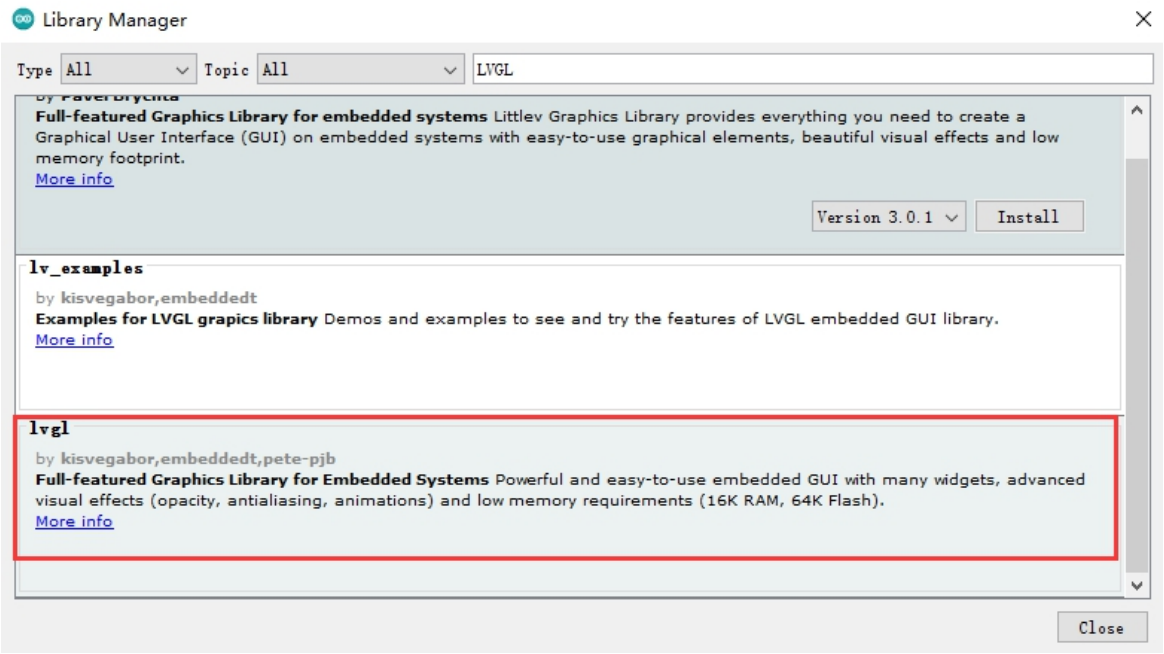


Download two library files .

One -TFT_e SPI

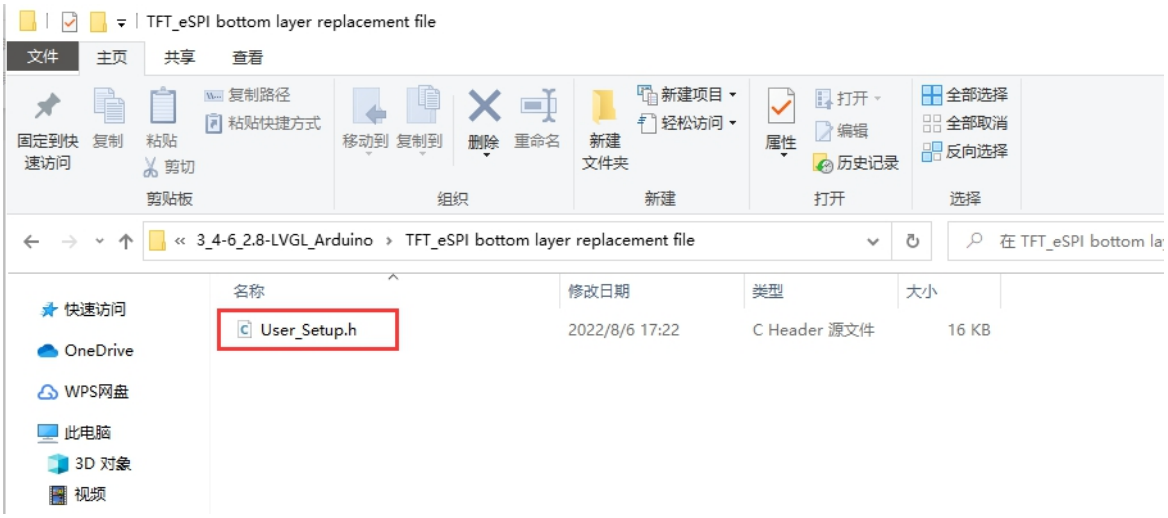


Two -Lvgl



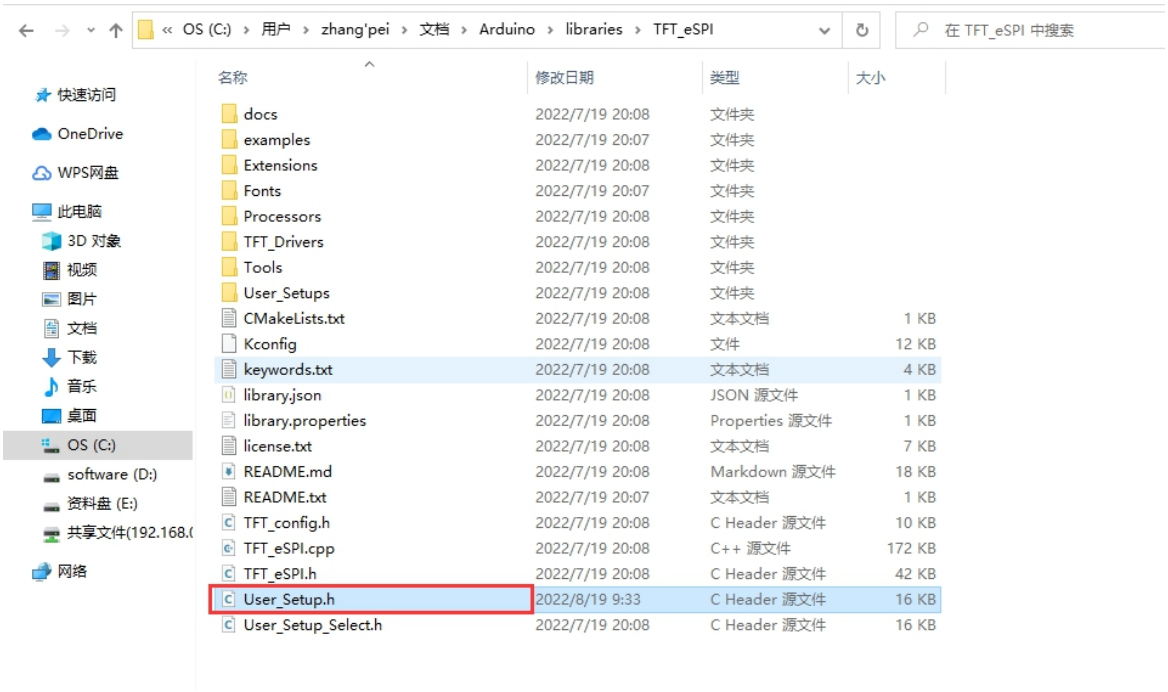
Copy the User_Setup.h of the data center .

As shown:



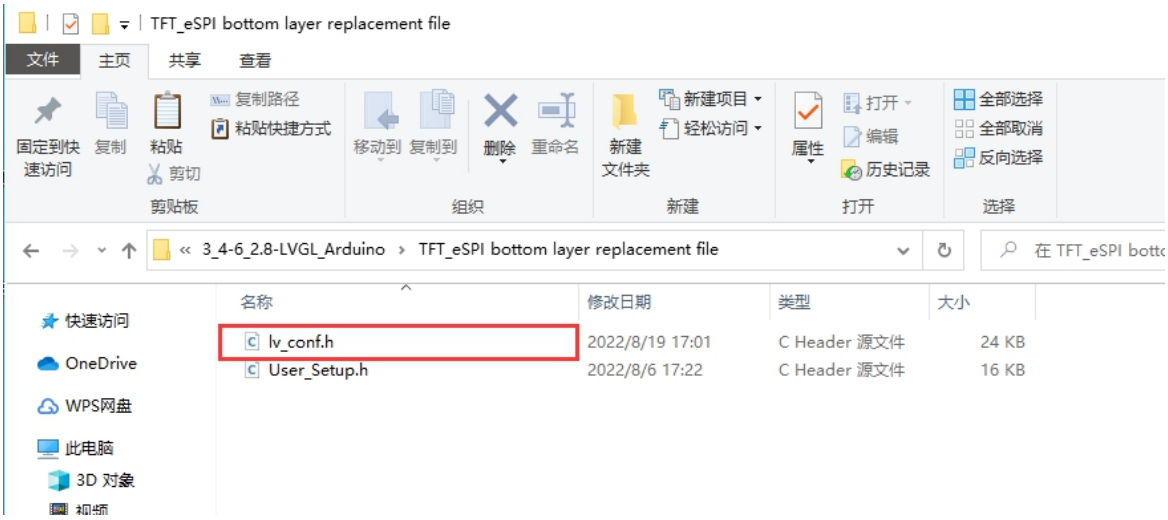
Replace the corresponding User_Setup.h file in TFT_eSPI with this file .

As shown:



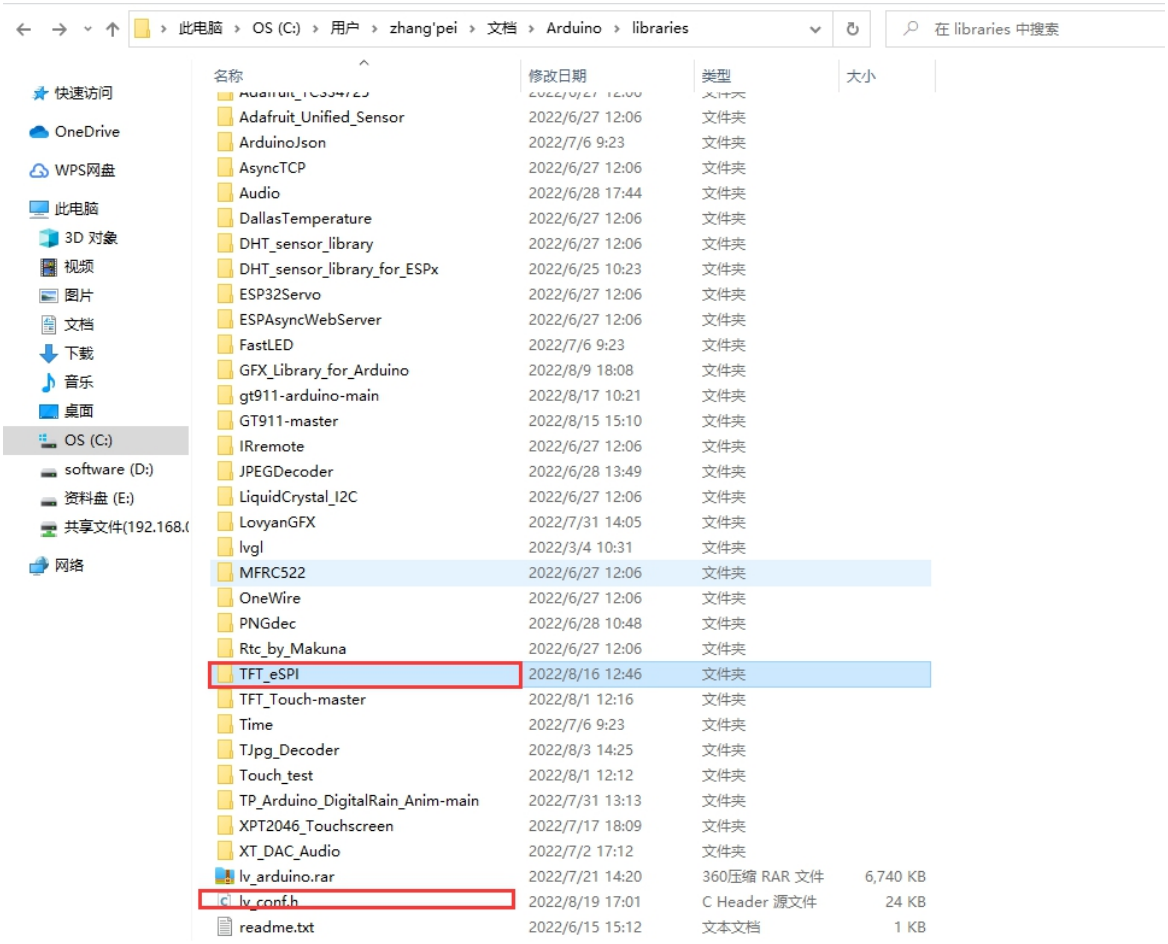
复制资料中心的 lv_conf.h

如图：



Put this file under the arduino library file, it must be in the same root directory as the library TFT_eSPI .

As shown：



After compiling, you can run LVGL and touch normally.

Function introduction

一、Basic settings

1. tft.init(); //Initialization

Initialize the screen, if it is ST7735, you can pass a parameter to it, and see when it is used .

2. tft.fillScreen(TFT_BLACK); //fill full screen fill full screen, followed by color values.

tft.fillScreen(uint32_t color);

3. Screen rotation

// Set the rotation angle of the screen display, the parameters are: 0, 1, 2, 3

// Represent 0°, 90°, 180°, 270°

void setRotation(uint8_t r);

4. Screen inversion

//Invert display colors i = 1 invert, i = 0 normal

tft.invertDisplay(bool i);

二、Text related API

1. tft.setCursor(20, 10, 4); //Set the starting coordinate position and font size of typing

// Set the text display coordinates. By default, the upper left corner of the text is used as the reference point. The reference point can be changed.

void setCursor(int16_t x, int16_t y);

// Set the text display coordinates, and the font of the text

void setCursor(int16_t x, int16_t y, uint8_t font);

2. tft.setTextColor(2); //Set font color

// Set text color

void setTextColor(uint16_t color);

// Set text color and background color

void setTextColor(uint16_t fgcolor, uint16_t bgcolor);

//Setting the background color can effectively prevent numbers from overlapping

3. tft.setTextSize(2); //Set font size

Setting the text size can enlarge the display of the font, but the "resolution" of the font will not change

// Set the text size, the text size range is an integer from 1 to 7

void setTextSize(uint8_t size);

4. tft.print("Hello World!");

// Display font

tft.print("Hello World!");

5. tft.printf, tft.println //Display font

Special Note: Font 7 is an imitation of a 7-segment digital screen

三、APIs related to drawing text

1. Draw the string (left)

int16_t drawString(const String &string, int32_t x, int32_t y)

int16_t drawString(const char * string, int32_t x, int32_t y)

int16_t drawString(const String &string, int32_t x, int32_t y, uint8_t font)

int16_t drawString(const char * string, int32_t x, int32_t y, uint8_t font)

2. Draw the string (centered)

int16_t drawCentreString(const char * string, int32_t x, int32_t y, uint8_t font)

int16_t drawCentreString(const String &string, int32_t x, int32_t y, uint8_t font)

3. Draw the string (right)

int16_t drawRightString(const char * string, int32_t x, int32_t y, uint8_t font)

int16_t drawRightString(const String &string, int32_t x, int32_t y, uint8_t font)

4. Drawing characters

int16_t drawChar(uint16_t uniCode, int32_t x, int32_t y)

int16_t drawChar(uint16_t uniCode, int32_t x, int32_t y, uint8_t font)

void drawChar(int32_t x, int32_t y, uint16_t c, uint32_t color, uint32_t bg, uint8_t size)

5. Plot floating point numbers

int16_t TFT_eSPI::drawFloat(float floatNumber, uint8_t decimal, int32_t x, int32_t y)

int16_t TFT_eSPI::drawFloat(float floatNumber, uint8_t decimal, int32_t x, int32_t y, uint8_t font)

tft.drawFloat(3.124, 4, 0,0,4);

6. Draw the numbers

int16_t drawNumber(long intNumber, int32_t x, int32_t y)

int16_t drawNumber(long intNumber, int32_t x, int32_t y, uint8_t font)

四、Drawing geometric figures

1. Draw the dots

void drawPixel(int32_t x, int32_t y, uint32_t color)

2. Draw lines

void drawLine(int32_t xs, int32_t ys, int32_t xe, int32_t ye, uint32_t color)

3. Draw a horizontal line (quick)

void drawFastHLine(int32_t x, int32_t y, int32_t w, uint32_t color)

4. Draw a vertical line (quick)

void drawFastVLine(int32_t x, int32_t y, int32_t h, uint32_t color)

5. Draw the hollow circle

tft.drawCircle(100, 100,50,TFT_RED);

6. Draw a filled circle

void fillCircle(int32_t x, int32_t y, int32_t r, uint32_t color)

7. Draw a hollow ellipse

tft.drawEllipse(100, 100, 100,60,TFT_GREENYELLOW);

8. Draw a solid ellipse

void drawRect(int32_t x, int32_t y, int32_t w, int32_t h, uint32_t color)

9. Draw a hollow rectangle

void drawRect(int32_t x, int32_t y, int32_t w, int32_t h, uint32_t color)

10. Draw a solid rectangle

void fillRect(int32_t x, int32_t y, int32_t w, int32_t h, uint32_t color)

11. Draw a hollow rounded rectangle

void drawRoundRect(int32_t x, int32_t y, int32_t w, int32_t h, int32_t radius, uint32_t color)

12. Draw a solid rounded rectangle

void fillRoundRect(int32_t x, int32_t y, int32_t w, int32_t h, int32_t radius, uint32_t color)

13. Draw Hollow Triangles

void drawTriangle(int32_t x1, int32_t y1, int32_t x2, int32_t y2, int32_t x3, int32_t y3, uint32_t color)

14. Draw Solid Triangles

void fillTriangle(int32_t x1, int32_t y1, int32_t x2, int32_t y2, int32_t x3, int32_t y3, uint32_t color)

五、Image display related

1. Display BMP picture

void drawBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t w, int16_t h, uint16_t fgcolor)

void drawBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t w, int16_t h, uint16_t fgcolor, uint16_t bgcolor)

2. XBM

xbm is a simple two-color image bitmap format, which was widely used in early cgi and is currently used in counters. Here TFT_eSPI recommends an online XBM production toolxbm is a simple two-color image bitmap format, which was widely used in early cgi and is currently used in counters. Here TFT_eSPI recommends an online XBM production tool

https://www.online-utility.org/image/convert/to/XBM

3. Test is very useful

void drawXBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t w, int16_t h, uint16_t fgcolor)

void drawXBitmap(int16_t x, int16_t y, const uint8_t *bitmap, int16_t w, int16_t h, uint16_t fgcolor, uint16_t bgcolor)

Display pictures

void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, const uint16_t *data) void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, uint16_t *data)

void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, const uint16_t *data, uint16_t transparent)

void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, uint16_t *data, uint16_t transparent)

void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, uint8_t *data, bool bpp8 =

true, uint16_t *cmap = (uint16_t *)nullptr)

void pushImage(int32_t x, int32_t y, int32_t w, int32_t h, uint8_t *data, uint8_t transparent, bool bpp8 = true, uint16_t *cmap = (uint16_t *)nullptr)