ESP32-S3-AMOLED-1.91

Overview

Usage instructions

ESP32-S3-AMOLED-1.91 currently provides two development tools and frameworks, Arduino IDE and ESP-IDF, providing flexible development options, you can choose the right development tool according to your project needs and personal habits.

Development tools

Arduino IDE

Arduino IDE is an open source electronic prototyping platform, convenient and flexible, easy to get started. After a simple learning, you can start to develop quickly. At the same time, Arduino has a large global user community, providing an abundance of open source code, project examples and tutorials, as well as rich library resources, encapsulating complex functions, allowing developers to quickly implement various functions.

ESP-IDF

ESP-IDF, or full name Espressif IDE, is a professional development framework introduced by Espressif Technology for the ESP series chips. It is developed using the C language, including a compiler, debugger, and flashing tools, etc., and can be developed via the command lines or through an integrated development environment (such as Visual Studio Code with the Espressif IDF plugin). The plugin offers features such as code navigation, project management, and debugging.

Each of these two development approaches has its own advantages, and developers can choose according to their needs and skill levels. Arduino are suitable for beginners and non-professionals because they are easy to learn and quick to get started. ESP-IDF is a better choice for developers with a professional background or high performance requirements, as it provides more advanced development tools and greater control capabilities for the development of complex projects.

Before operating, it is recommended to browse the table of contents to quickly understand the document structure. For smooth operation, please read the FAQ carefully to understand possible problems in advance. All resources in the document are provided with hyperlinks for easy download.

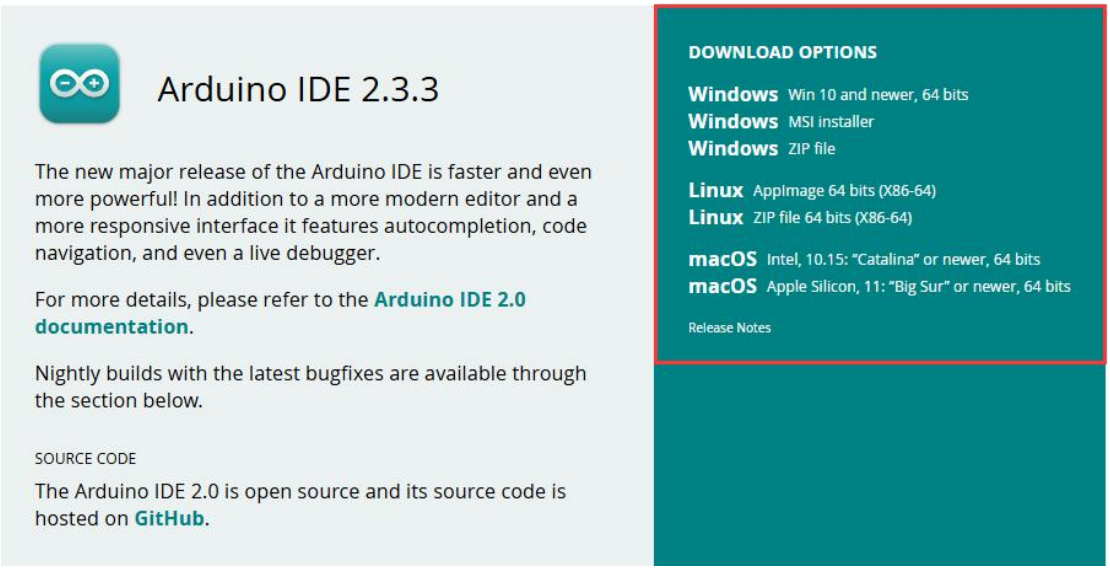
Working with Arduino

This chapter introduces setting up the Arduino environment, including the Arduino IDE, management of ESP32 boards, installation of related libraries, program compilation and downloading, as well as testing demos. It aims to help users master the development board and facilitate secondary development.

Environment setup

Download and install Arduino IDE

Click to visit the official website, select the corresponding system and system bit to download.



Run the installer and install all by default.

The environment setup is carried out on the Windows 10 system, Linux and Mac users can access Arduino-esp32 environment setup for reference

Install Arduino-ESP32

Regarding ESP32-related motherboards used with the Arduino IDE, the esp32 by Espressif Systems library must be installed first.

It is generally recommended to use Install Online. If online installation fails, use Install Offline.

To install the Arduino-ESP32 tutorial, please refer to Arduino board manager tutorial

The ESP32-S3-AMOLED-1.91 development board comes with an offline package. Click here to download: esp32_package_3.0.7_arduino offline package

ESP32-S3-AMOLED-1.91 Development board installation instructions

Board name	Board installation requirements	Version number requirements
ESP32-S3-AMOLED-1.91	"Install Offline" / "Install Online"	3.0.7 and above

Install library

When installing Arduino libraries, there are usually two ways to choose from: Install online and Install offline. If the library installation requires offline installation, you must use the provided library file

- For most libraries, users can easily search and install them through the online library manager of the Arduino software. However, some open-source libraries or custom libraries are not synchronized to the Arduino Library Manager, so they cannot be acquired through online searches. In this case, users can only manually install these libraries offline.
- For library installation tutorial, please refer to Arduino library manager tutorial
- ESP32-S3-AMOLED-1.91 library file is stored in the sample program

ESP32-S3-AMOLED-1.91 library installation description

Library Name	Description	Version	Library Installation Requirements
LVGL	Graphical library	v8.4.0	"Install Offline"

Run the first Arduino demo

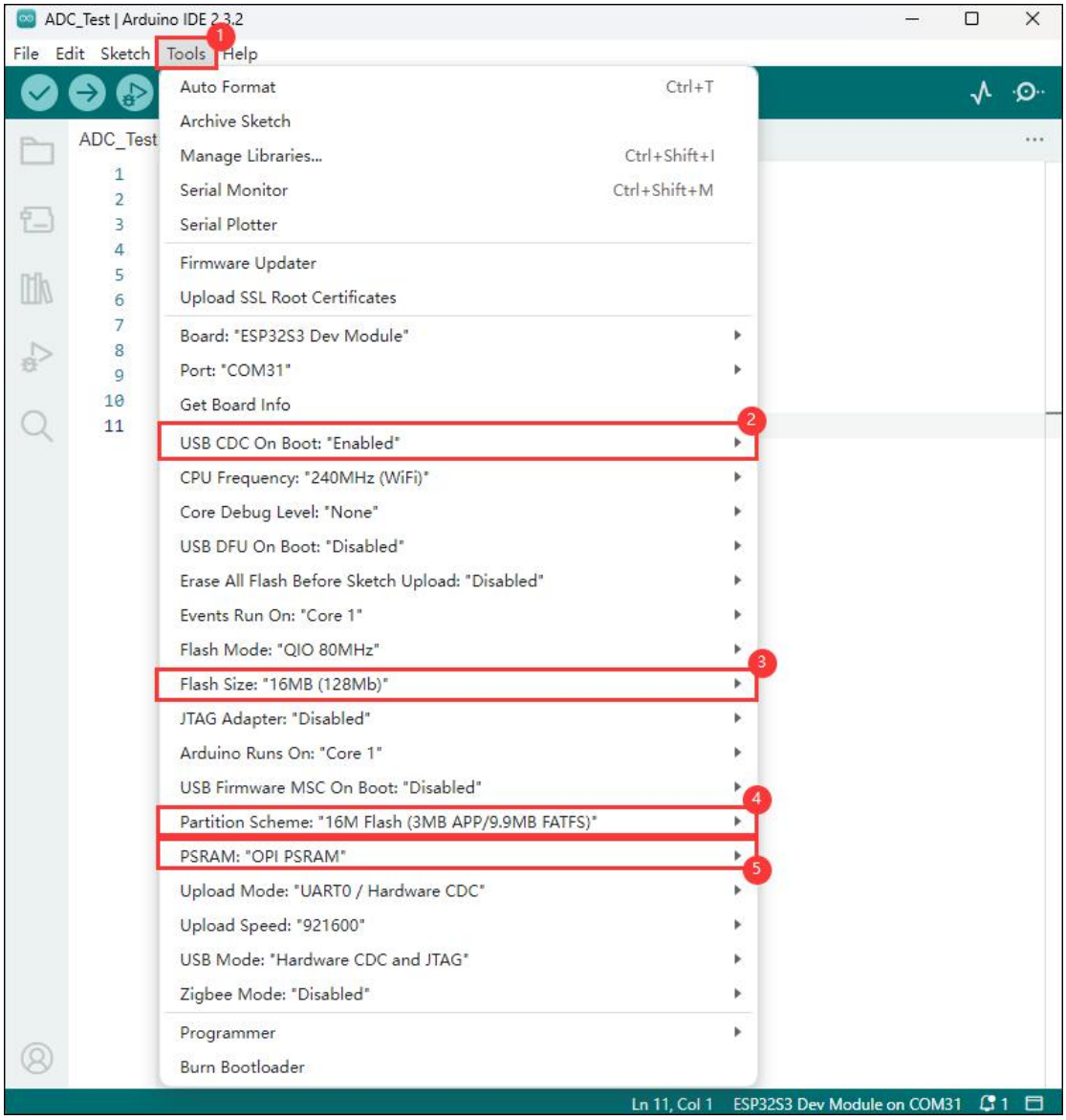
If you are just getting started with ESP32 and Arduino, and you don't know how to create, compile, flash, and run Arduino ESP32 programs, then please expand and take a look. Hope it can help you!

Demos

ESP32-S3-AMOLED-1.91 demos

Demo	Basic Description	Dependency Library
ADC	Read the current voltage value of the system	-
I2C_QMI8658	Print the original data sent by the IMU	-
LVGL	LVGL demo	LVGL
SPI_SD	Load and display the information of the TF card	-
WIFI_STA	Set to STA mode to connect to WiFi and obtain an IP address	-
WIFI_AP	Set to AP mode to obtain the IP address of the access device	-
Arduino_Playablity	Playability program demo	-

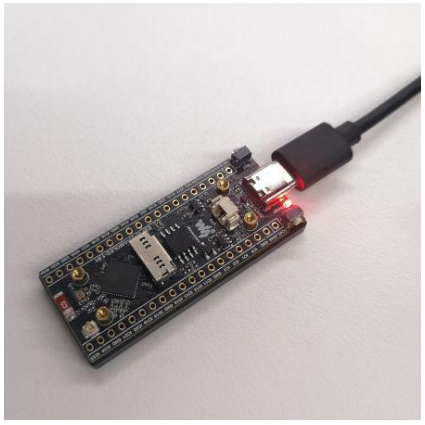
Arduino project parameter settings



ADC

Hardware connection

Connect the board to the computer using a USB cable

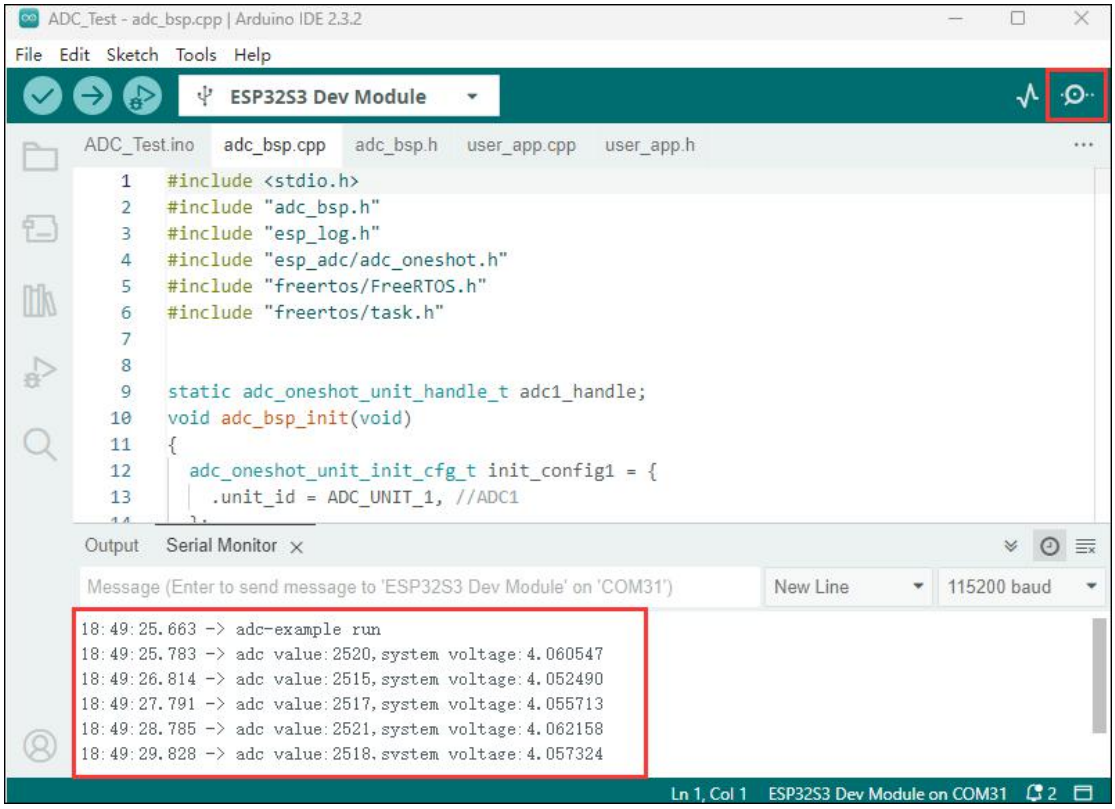


Code analysis

- adc_bsp_init(void): Initializes ADC1, including creating an ADC one-time trigger unit and configuring channel 0 for ADC1.
- adc_get_value(float *value,int *data): Reads the value of ADC1 channel 0 and calculates the corresponding voltage value based on the reference voltage and resolution, stores it at the position where the incoming pointer points to, and stores 0 if the read fails.
- adc_example(void* parmeter): After initializing the ADC, in an infinite loop, read the value of the ADC channel on GPIO1, print the original ADC value, and calculate the system voltage value, performing this operation once every second.

Result demonstration

Open the serial port monitoring, you can see the ADC value and voltage in the printed output, as shown in the following figure:



The ADC sampling value is around 2518, and the actual voltage is greater than 3.3V, which is due to the voltage boost through the battery chip. For a detailed analysis, you can refer to the schematic diagram

I2C_QMI8658

Hardware connection

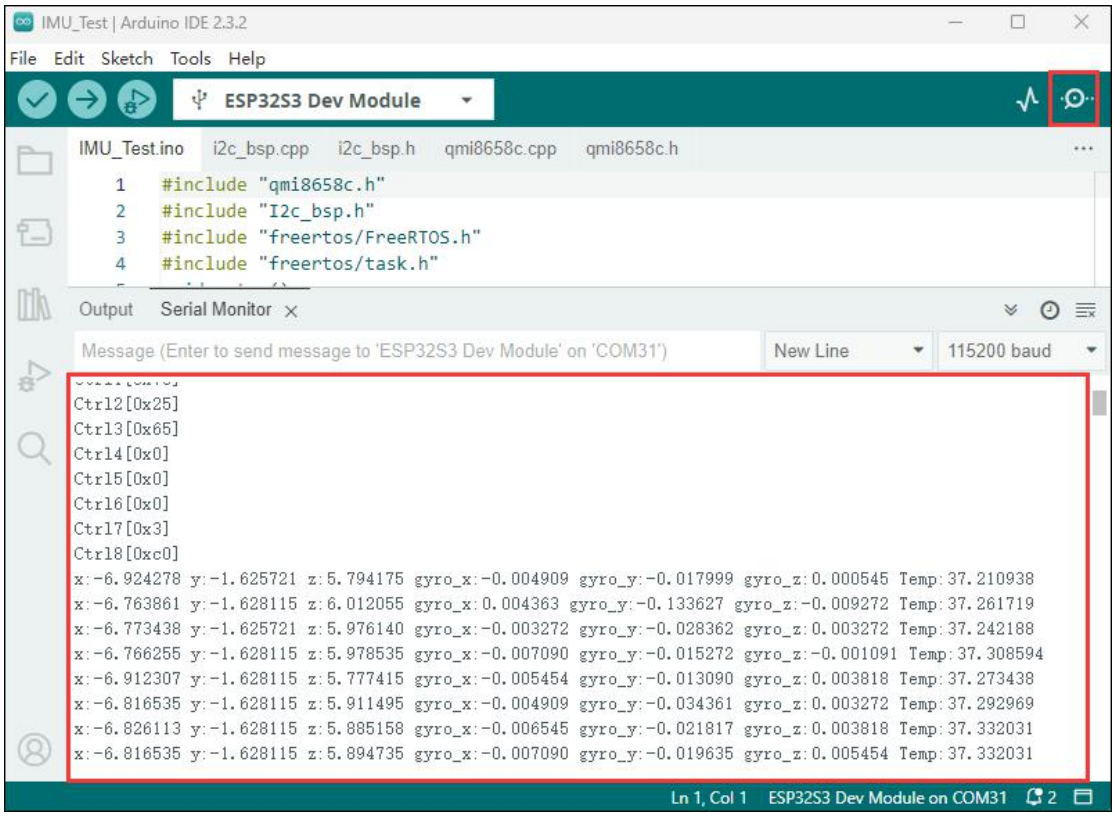
Connect the board to the computer using a USB cable

Code analysis

qmi8658c_example(void* parmeter): The function initializes the QMI8658 device, reading and printing accelerometer data, gyroscope data, and temperature data in an infinite loop, once every second. During the rotation of the board, the gyroscope data increases with greater rotation speed, and the accelerometer calculates the corresponding acceleration based on the current position.

Result demonstration

Open the serial port monitoring, and you can see the original data output from the IMU (Euler angles need to be converted by yourself), as shown in the following figure:



Data is output once every second. If you need to modify or refer to it, you can directly access the qmi source file for operations.

LVGL

Hardware connection

Connect the board to the computer using a USB cable

Code analysis

For LVGL, lvgl_conf.h is its configuration file, and below are explanations for some commonly used contents.

```
/*Color depth: 1 (1 byte per pixel), 8 (RGB332), 16 (RGB565), 32
(ARGB8888)*/#define LV_COLOR_DEPTH 16//Color depth, a macro defi
nition that must be concerned with porting LVGL


#define LV_MEM_CUSTOM 0#if LV_MEM_CUSTOM == 0

   /*Size of the memory available for `lv_mem_alloc()` in bytes
(>= 2kB)*/

   #define LV_MEM_SIZE (48U * 1024U)          /*[bytes]*/


   /*Set an address for the memory pool instead of allocating it
 as a normal array. Can be in external SRAM too.*/

   #define LV_MEM_ADR 0     /*0: unused*/

   /*Instead of an address give a memory allocator that will be
called to get a memory pool for LVGL. E.g. my_malloc*/

   #if LV_MEM_ADR == 0

      #undef LV_MEM_POOL_INCLUDE

      #undef LV_MEM_POOL_ALLOC

   #endif

#else      /*LV_MEM_CUSTOM*/

   #define LV_MEM_CUSTOM_INCLUDE <stdlib.h>   /*Header for the d
ynamic memory function*/

   #define LV_MEM_CUSTOM_ALLOC   malloc

   #define LV_MEM_CUSTOM_FREE    free

   #define LV_MEM_CUSTOM_REALLOC realloc#endif     /*LV_MEM_CUS
TOM*///The above section is mainly for LVGL memory allocation, /
/which defaults to lv_mem_alloc() versus lv_mem_free().
```

There are also some LVGL demos and file systems that can be set in the conf configuration file.

Code modification

For boards without touch, you need to find **`EXAMPLE_USE_TOUCH`** in the file where the main function is located, and change 1 to 0

For boards with touch, you need to find **`EXAMPLE_USE_TOUCH`** in the file where the main

function is located, and change 0 to 1

```
#define EXAMPLE_USE_TOUCH             0
```

Result demonstration

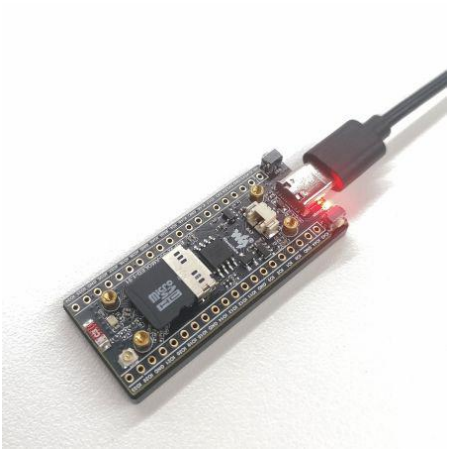
The LVGL demo has high requirements for RAM and ROM, so it is necessary to configure the demo according to the requirements of environment setup, and after the program is flashed, the running effect of the device is as follows (if it is a board with touch, you need to modify the macro definition EXAMPLE_USE_TOUCH to 1):

For more learning and use of LVGL, please refer to LVGL official documentation

SPI_SD

Hardware connection

Install a TF card on the board (you must insert a TF card with a capacity of less than 64G first), and use a USB cable to connect the board to the computer (if it is a V1 board, you need to comment out the macro definition VersionControl_V2, which is enabled by default)

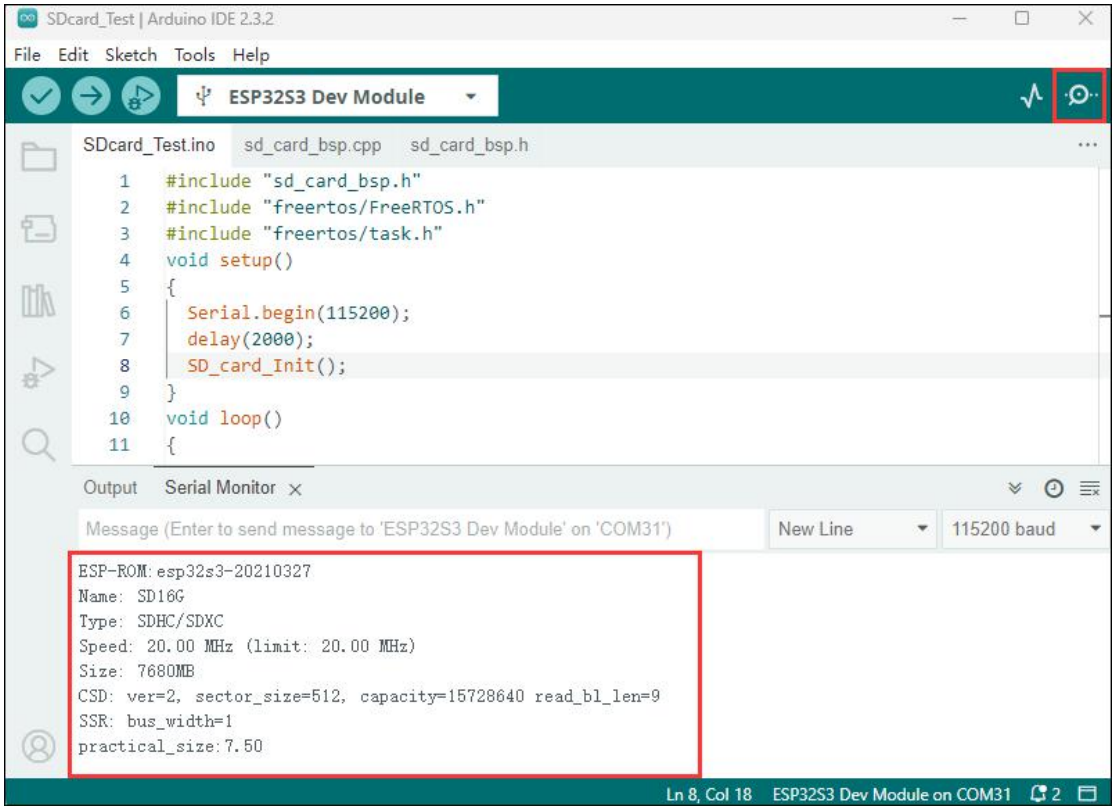


Code analysis

`SD_card_Init(void)`: This function initializes the TF card according to different configurations, including configuring the mounting parameters, host and card slot parameters, and then tries to mount the TF card, if successful, the card information and capacity are printed.

Result demonstration

Click on the serial port monitoring device, you can see the information of the output TF card, practical_size is the actual capacity of the TF card, as shown in the figure below:



How to know more about the Arduino ESP32 libraries which explain the use of TF card? Please refer to [Arduino ESP32 library TF card use].

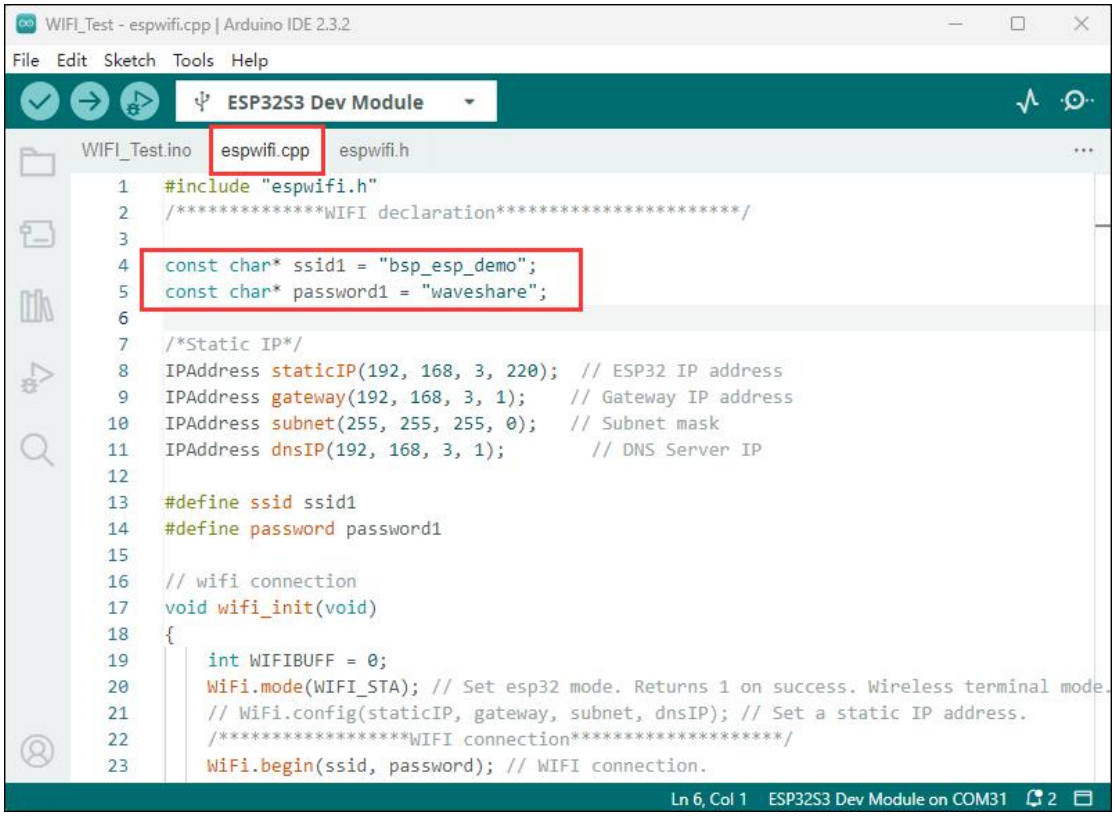
WIFI_STA

Hardware connection

Connect the board to the computer using a USB cable

Code modification

The project realizes that the chip is connected to WIFI in STA mode and obtains the IP address, before compiling and downloading the firmware, some code needs to be modified, specifically changing the name and password of the WIFI router to those suitable for the environment.

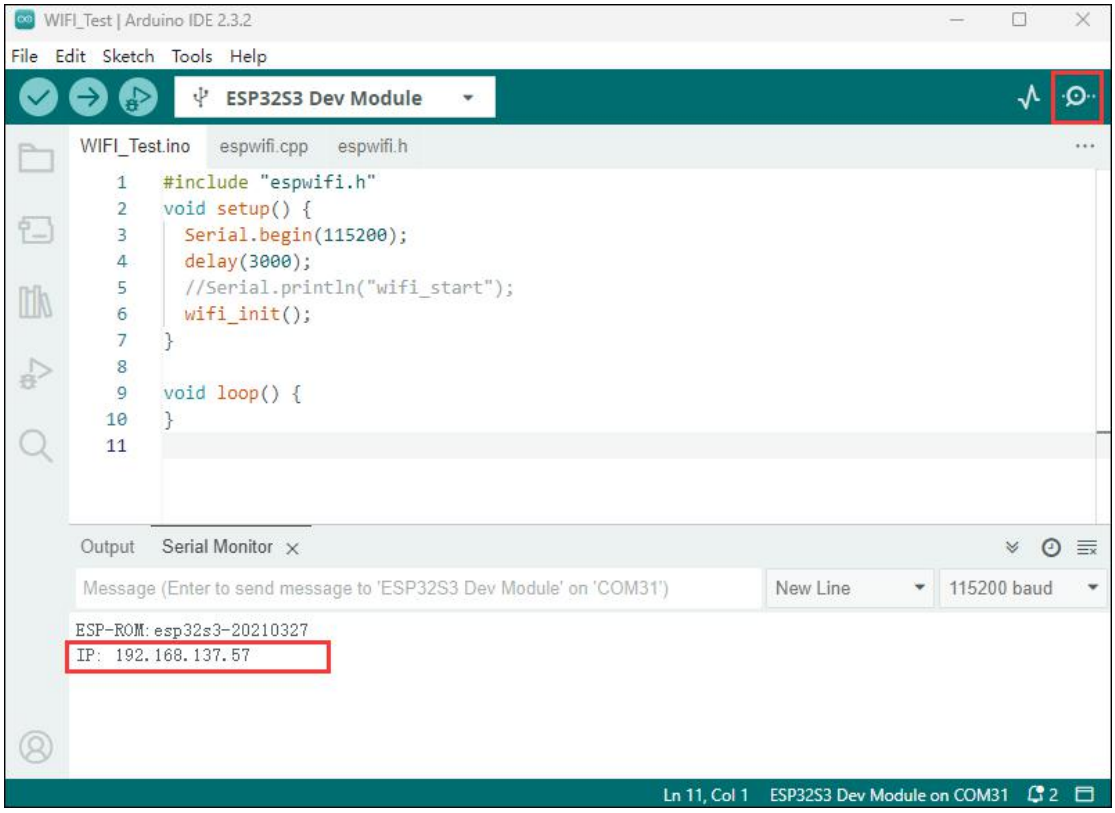


Code analysis

wifi_init(void): This function is used to initialize the Wi-Fi connection of the ESP32. It sets the ESP32 to Wi-Fi site mode and tries to connect to the specified Wi-Fi network (via the ssid and password). If the connection is successful, it prints the local IP address; if the connection fails within a certain period (20 * 500 milliseconds), it prints the connection failure message. At the same time, the function can also set the auto-connection and auto-reconnect functions.

Result demonstration

The chip is successfully connected to WIFI in STA mode, and you can see the obtained IP address by clicking on the serial port monitoring device.



WIFI_AP

Hardware connection

Connect the board to the computer using a USB cable

Code analysis

The code initializes serial communication on the ESP32 and then creates a WiFi access point named "bsp_esp_demo" with the password "waveshare", and no other continuous operations are performed in a loop while the program runs.

```
const char* ssid     = "bsp_esp_demo";
const char* password = "waveshare";

WiFi.softAP(ssid,password);
```

Result demonstration

Use your mobile phone or other device to connect to WIFI, the WiFi name is "bsp_esp_demo", and the password is "waveshare"

Arduino_Playablity

We also provide some playable demos for your reference, but it should be noted that the

following demos are based on ESP32_Arduino versions below V3.0. Environment setup can be referred to below

Install the 2.0.9 environment directly by downloading the Arduino2.0.9 offline package, double-click to install it after the download is completed

Click to download the required library files, the library file is installed in the same way as the Arduino library file installation above

Hardware connection

Connect the board to the computer using a USB cable

Result demonstration

RGBpalette AnimationDemo SliderControl

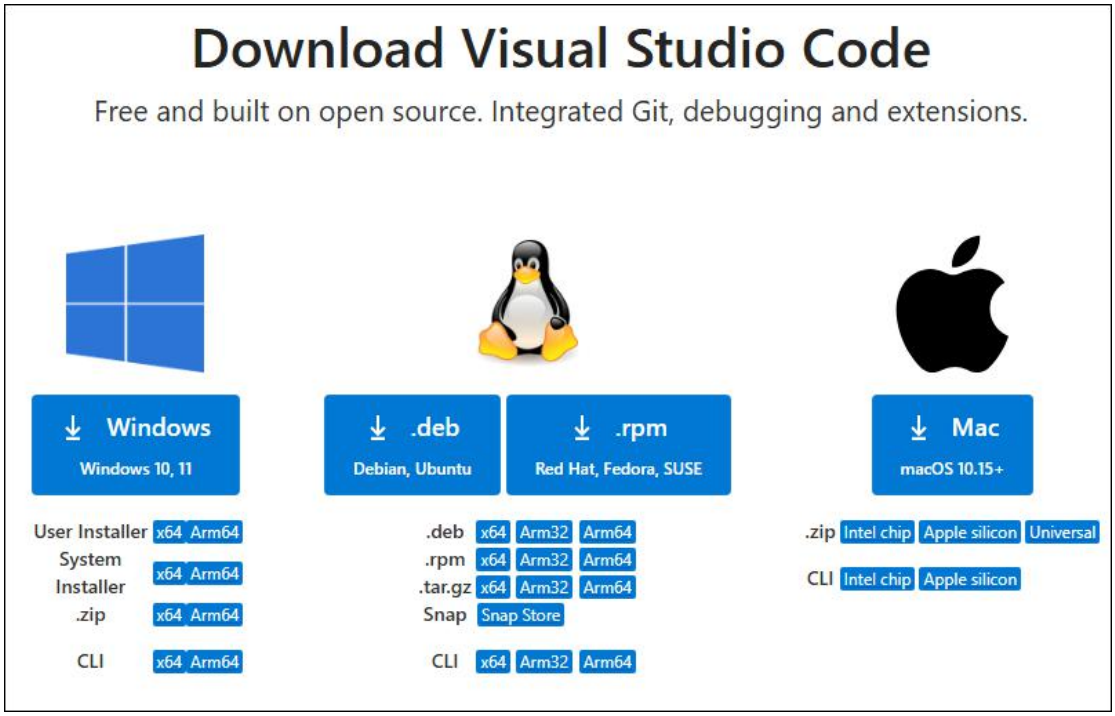
Working with ESP-IDF

This chapter introduces setting up the ESP-IDF environment setup, including the installation of Visual Studio and the Espressif IDF plugin, program compilation, downloading, and testing of example programs, to assist users in mastering the development board and facilitating secondary development.

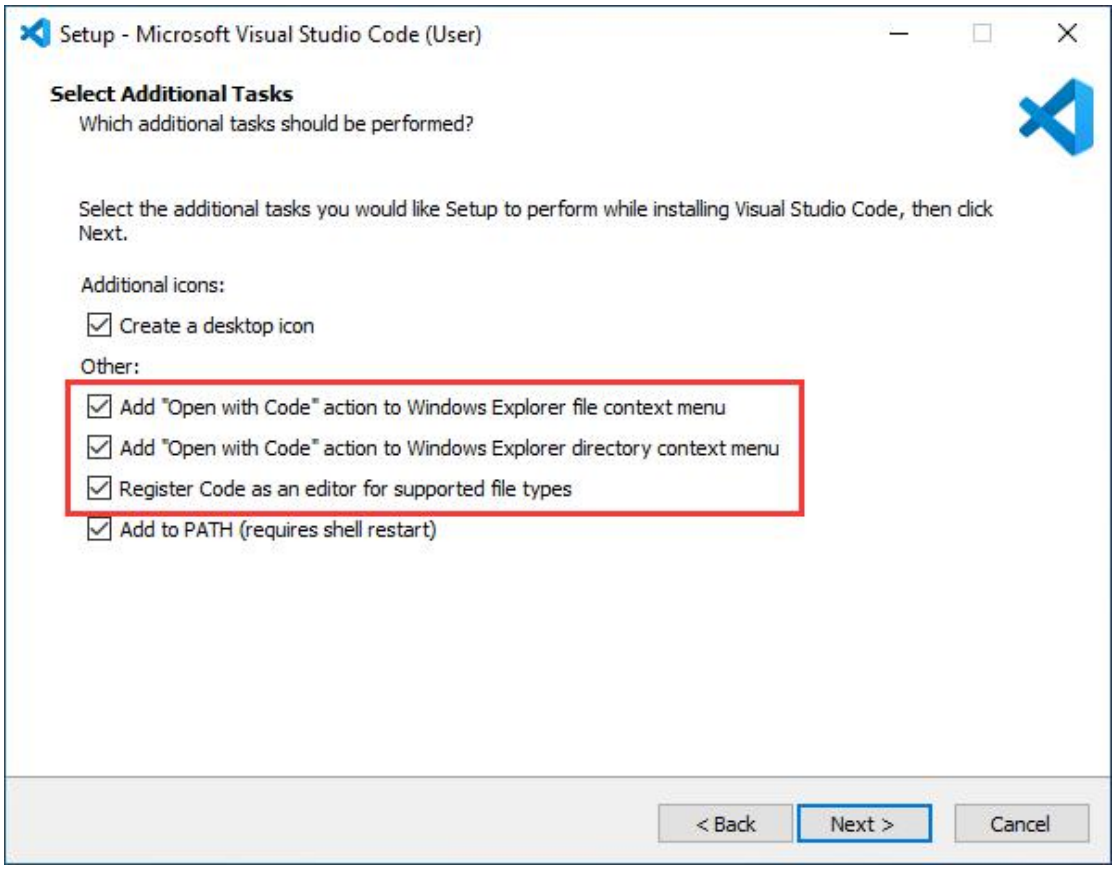
Environment setup

Download and install Visual Studio

Open the download page of VScode official website, choose the corresponding system and system bit to download



After running the installation package, the rest can be installed by default, but here for the subsequent experience, it is recommended to check boxes 1, 2, and 3



After the first two items are enabled, you can open VSCode directly by right-clicking files or directories, which can improve the subsequent user experience.

After the third item is enabled, you can select VSCode directly when you choose how to open it.

The environment setup is carried out on the Windows 10 system, Linux and Mac users can access ESP-IDF environment setup for reference

Install Espressif IDF Plug-in

It is generally recommended to use Install Online. If online installation fails due to network factor, use Install OIffline.
For more information about how to install the Espressif IDF plugin, see Install Espressif IDF Plugin

Run the first ESP-IDF demo

If you are just getting started with ESP32 and ESP-IDF, and you don't know how to create, compile, flash, and run ESP-IDF ESP32 programs, then please expand and take a look. Hope it can help you!

Demos

ESP32-S3-AMOLED-1.91 demos

Demo	Basic Description	Dependency Library
ADC	Read the current voltage value of the system	-
I2C_QMI8658	Print the original data sent by the IMU	-
LVGL	LVGL demo	LVGL
SPI_SD	Load and display the information of the TF card	-
WIFI_STA	Set to STA mode to connect to WiFi and obtain an IP address	-
WIFI_AP	Set to AP mode to obtain the IP address of the access device	-
FactoryProgram	Comprehensive project	LVGL

ADC

Hardware connection

Connect the board to the computer using a USB cable

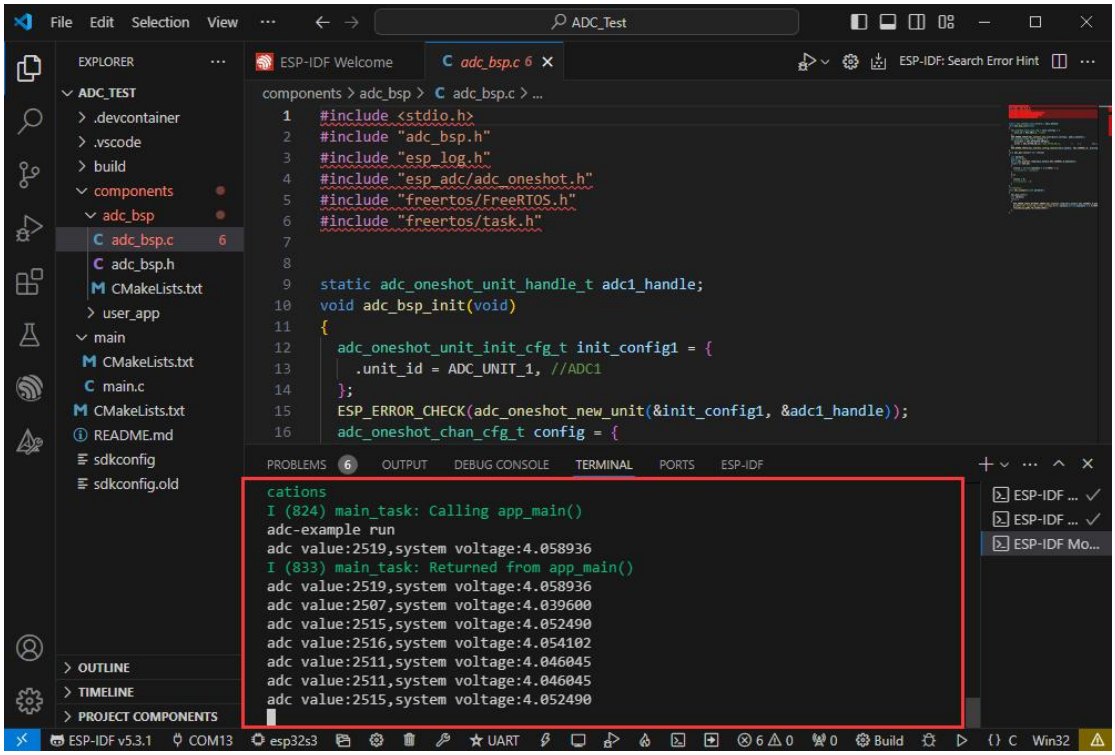
Code analysis

- `adc_bsp_init(void)`: Initializes ADC1, including creating an ADC one-time trigger unit and configuring channel 0 for ADC1.
- `adc_get_value(float *value)`: Reads the value of ADC1 channel 0 and calculates the corresponding voltage value based on the reference voltage and resolution, stores it at the position where the incoming pointer points to, and stores 0 if the read fails.
- `adc_example(void* parmeter)`: After initializing the ADC, in an infinite loop, read the value of the ADC channel on GPIO1, print the original ADC value, and calculate the system voltage value, performing this operation once every second.

Result demonstration

After the demo is flashed, the running result of the device is as follows:

Open the serial port monitoring, you can see the output ADC value and voltage, as shown in the following figure:



It can be seen that the ADC sampling value is about 2511, and the actual voltage is greater than 3.3V, because this is the voltage after the battery chip is boosted. For specific analysis, you can click to view the schematic diagram

I2C_QMI8658

Hardware connection

Connect the board to the computer using a USB cable

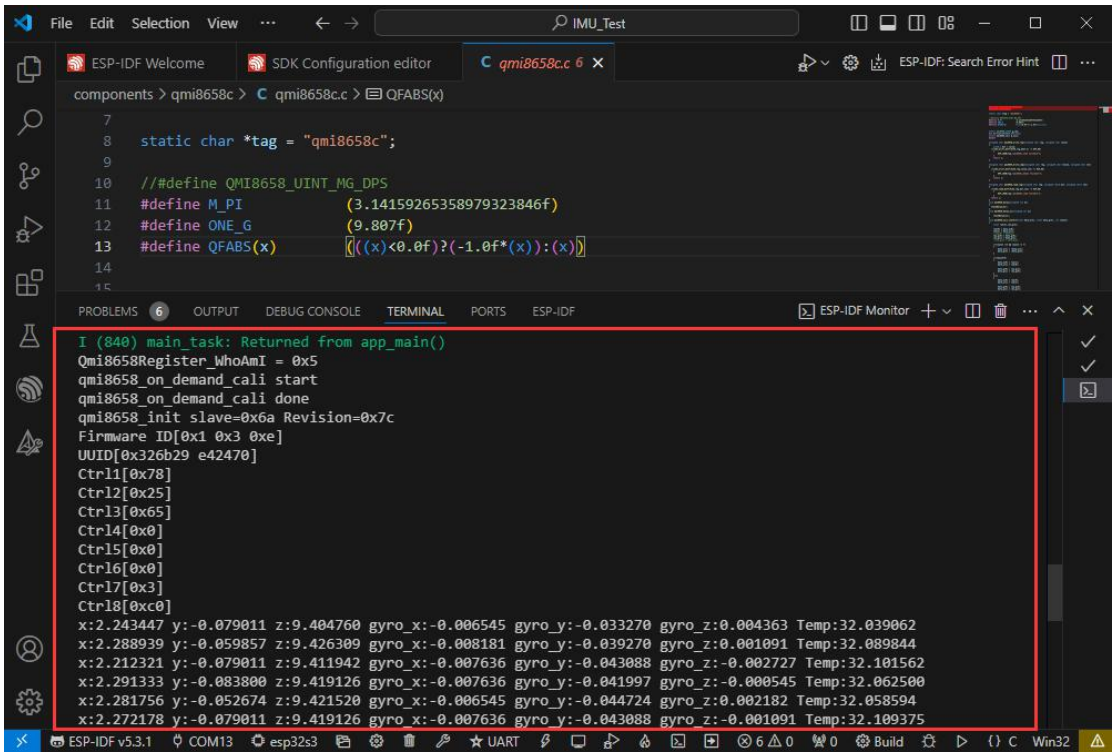
Code analysis

`qmi8658c_example(void* parmeter)`: The function initializes the QMI8658 device, reading and printing accelerometer data, gyroscope data, and temperature data in an infinite loop, once every second.

Result demonstration

After the demo is flashed, the running result of the device is as follows:

Open the serial port monitoring, and you can see the original data output from the IMU (Euler angles need to be converted by yourself), as shown in the following figure:



You can see that it is output every 1 second. If you need to modify or refer to it, you can directly go to the qmi source file to modify it

LVGL

Hardware connection

Connect the board to the computer using a USB cable

Code analysis

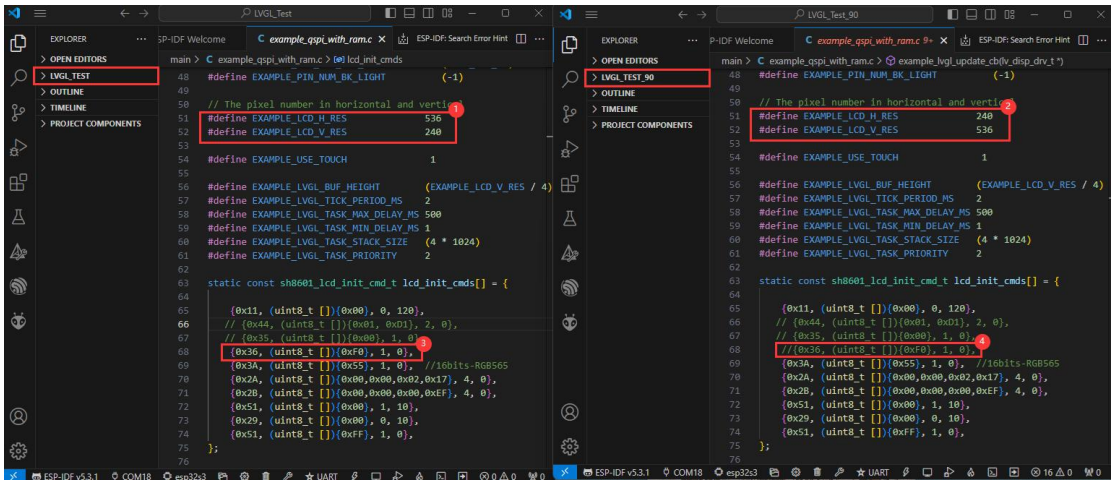
There are two projects in the LVGL catalog, one is the original display, and the other is the 90-degree rotation display, which can be selected according to your needs.

The original display and the 90-degree rotation display are mainly different from two points:

① and ②: There is a difference in the number of pixels in the horizontal and vertical directions

③ and ④: The register at the 0x36 address under the lcd_init_cmds array is responsible for rotation, and modifying its value can achieve a rotation effect. If no rotation is needed, you can

directly delete {0x36,(uint8_t []){0xF0,1,0}}



Code modification

For boards without touch, you need to find **EXAMPLE_USE_TOUCH** in the file where the main function is located, and change 1 to 0

For boards with touch, you need to find **EXAMPLE_USE_TOUCH** in the file where the main function is located, and change 0 to 1

```
#define EXAMPLE_USE_TOUCH                0
```

Result demonstration

lvgl (Original display) *lvgl_90 (90-degree rotation display)*

Want to know the description of the screen driver IC register? Please refer to [RM67162 Datasheet]

SPI_SD

Hardware connection

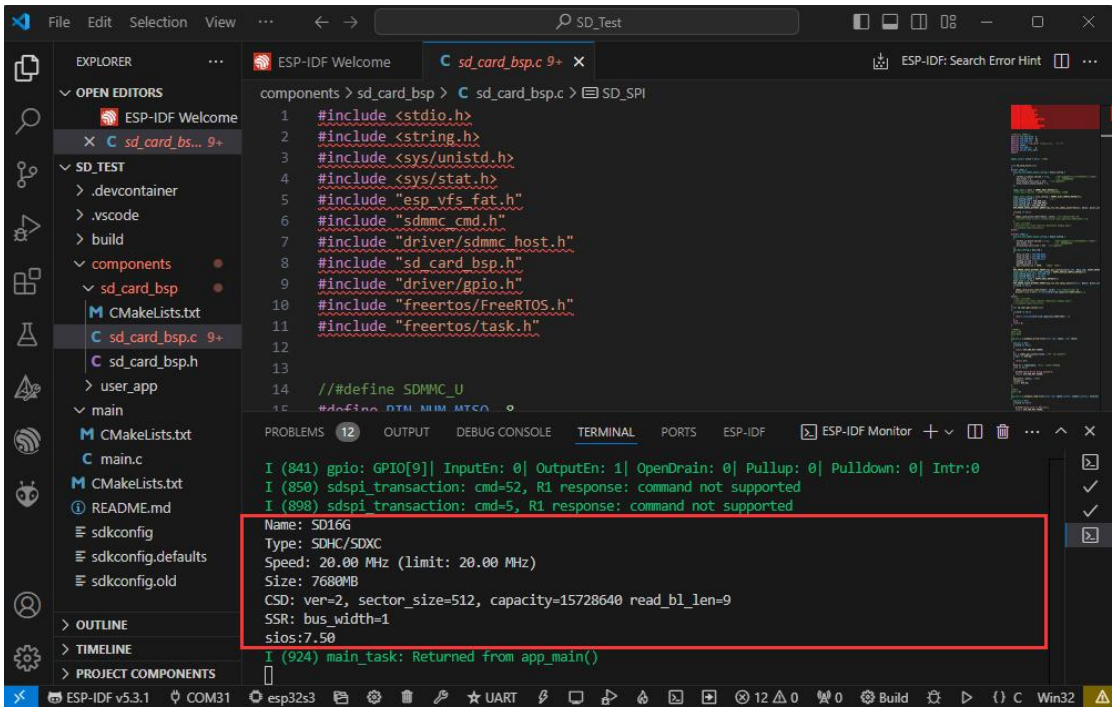
Install a TF card on the board (you must insert a TF card with a capacity of less than 64G first), and use a USB cable to connect the board to the computer (if it is a V1 board, you need to comment out the macro definition VersionControl_V2 in sd_card_bsp.c, which is enabled by default)

Code analysis

SD_card_Init(void): This function initializes the TF card according to different configurations, including configuring the mounting parameters, host and card slot parameters, and then tries to mount the TF card, if successful, the card information and capacity are printed.

Result demonstration

You can see the output TF card information



How to know more about the Arduino ESP32 libraries which explain the use of TF card? Please refer to Arduino ESP32 library TF card use.

WIFI_STA

Hardware connection

Connect the board to the computer using a USB cable

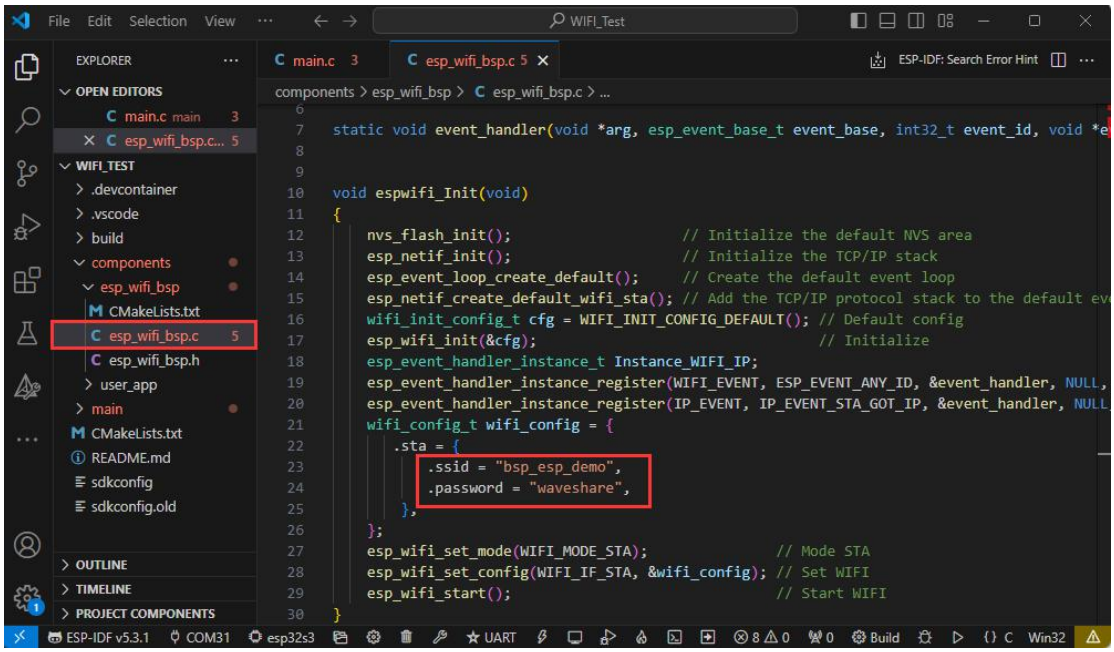
Code analysis

espwifi_Init(void): This function is used for WiFi initialization on ESP32. It sequentially initializes

non-volatile storage, the TCP/IP stack, creates a default event loop and a default WiFi site network interface, initializes WiFi with the default configuration, registers event handlers to handle WiFi and IP-related events, sets WiFi connection parameters, and starts WiFi.

Code modification

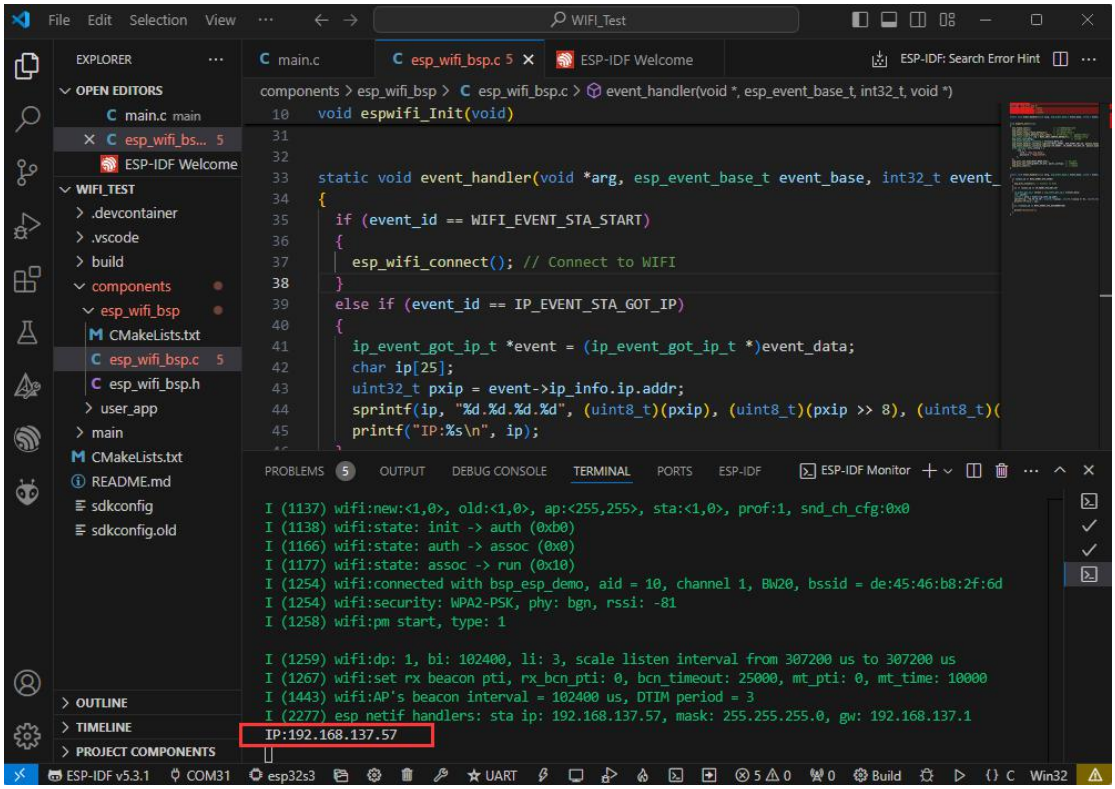
The project realizes that the chip is connected to WIFI in STA mode and obtains the IP address, before compiling and downloading the firmware, some code needs to be modified, specifically changing the name and password of the WIFI router to those suitable for the environment.



Result demonstration

After the demo is flashed, the running result of the device is as follows:

```
* The chip successfully connects to WIFI and obtains an IP addre
ss while in STA mode.
```



WIFI_AP

Hardware connection

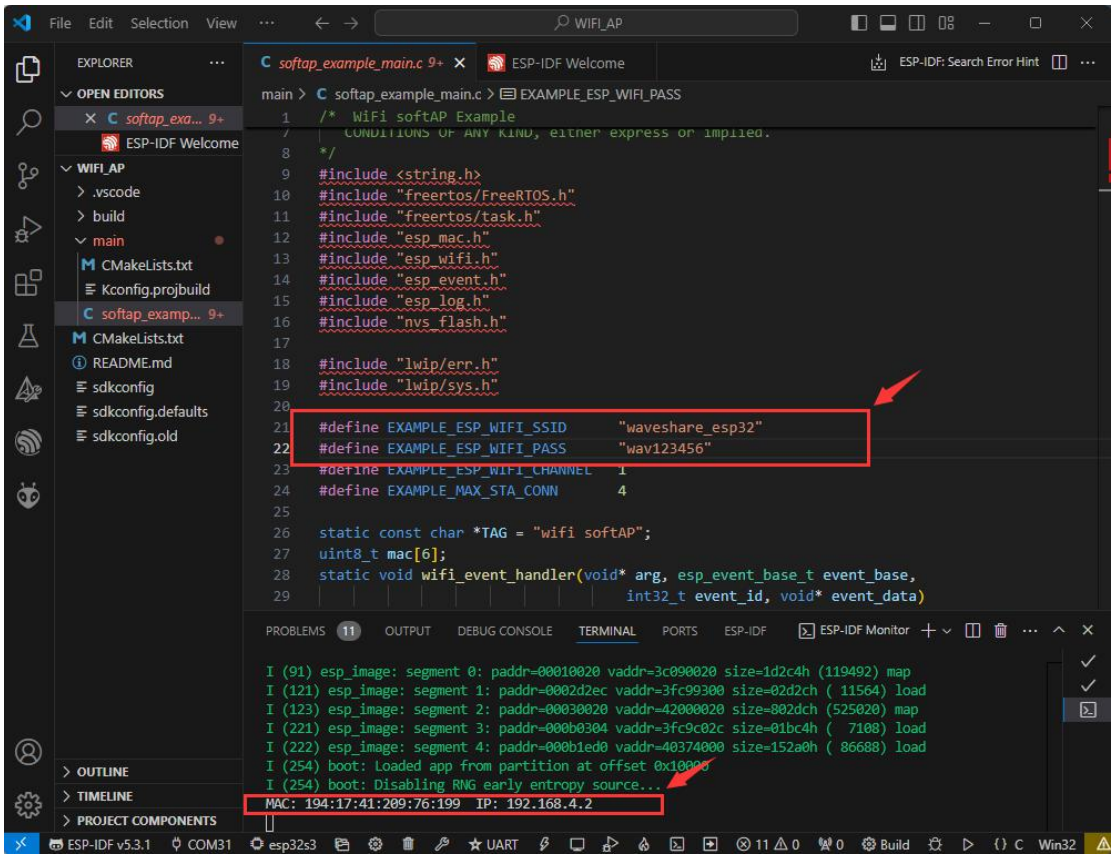
Connect the board to the computer using a USB cable

Code analysis

wifi_init_softap(void): This function is used to initialize the ESP32's Wi-Fi soft access point, including setting up the network interface, registering event handling functions, configuring soft AP parameters, and starting the soft AP.

Result demonstration

When the chip is in AP mode, use the mobile phone to successfully connect to WIFI and the serial port will print the MACA address of the connected device and the IP address assigned to the device.



FactoryProgram

Hardware connection

Use a USB cable to connect the board to the computer (if it is a V1 version, you need to comment out the CMake statement target_compile_definitions which is enabled by default in the CMakeLists.txt file under the ui_bsp directory)

Code modification

For boards without touch, you need to find EXAMPLE_USE_TOUCH in the file where the main function is located, and change 1 to 0

For boards with touch, you need to find EXAMPLE_USE_TOUCH in the file where the main function is located, and change 0 to 1

```
#define EXAMPLE_USE_TOUCH                0
```

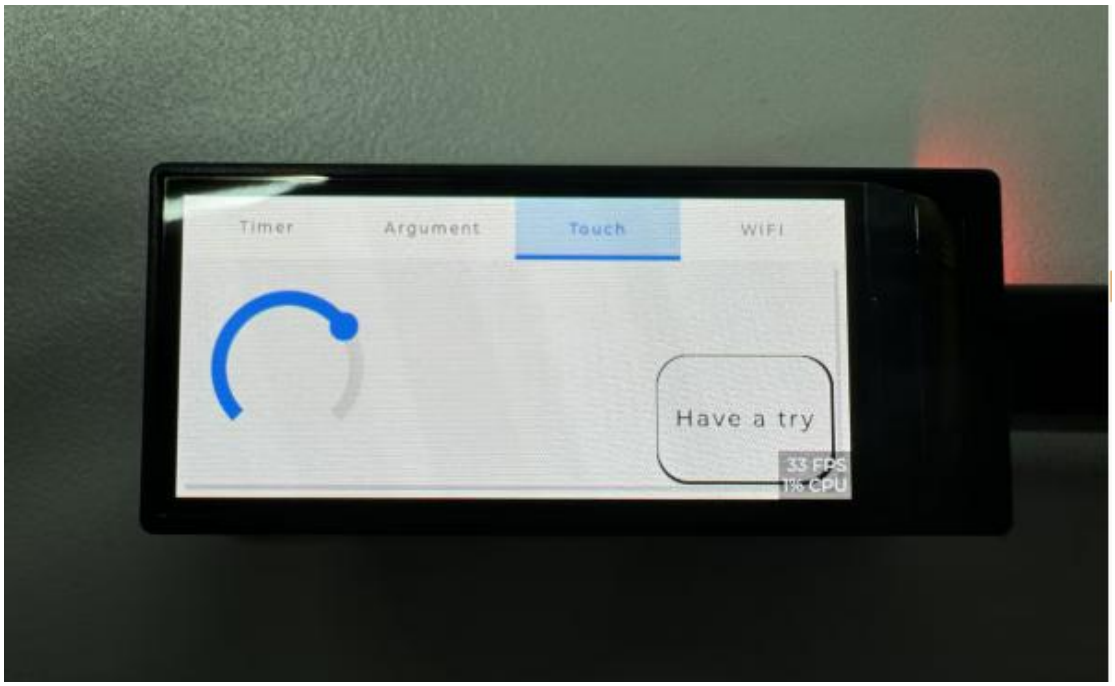
Result demonstration

The touch version supports BOOT button and touch switching

Switch pages by using the BOOT button on the side of the board to display the analog clock interface first

Switch to the Argument interface by pressing the BOOT button, where you can see a page with information on the onboard hardware

Click the BOOT button again to jump to the Touch interface, which is the component interface



Click the BOOT button again to jump to the WIFI interface

Resources

Schematic diagram

ESP32-S3-AMOLED-1.91 Schematic diagram

Demos

ESP32-S3-AMOLED-1.91 Demo

ESP32-S3-AMOLED-1.91-Arduino_Playablity

Datasheets

ESP32-S3
ESP32-S3 Datasheet
ESP32-S3 Technical reference manual

Other components

QMI8658 Datasheet
RM67162 Datasheet

Software tools

Arduino

Arduino IDE Official download link
ESP32-Arduino official documentation
esp32_package_3.0.7_arduino offline package
esp32_package_2.0.9_arduino offline package

VScode

VScode official website

Other resource link

LVGL official documentation

FAQ

Question1: After the module downloads the demo and re-downloads it, why sometimes it can't connect to the serial port or the flashing fails?

Answer1:

Long press the BOOT button, press RESET at the same time, then release RESET, then release the BOOT button, at this time the module can enter the download mode, which can solve most of the problems that can not be downloaded.

Question2: Failed to set up the VSCode environment?

Answer2:

First consider the network issue, try switching to another network

Question3: Error when compiling an Arduino program?

Answer3:

Check if the Arduino IDE -> Tools is correctly configured

Question4: Is it stuck when sliding pictures displayed?

Answer4:

Modify LVGL display cache to the full screen size

Modify the LV_IMG_CACHE_DEF_SIZE option in the configuration to 1000 to achieve some optimization

Question5: Can't display Chinese?

Answer5:

Basic Chinese can be displayed, but if it is a rare character, it cannot be displayed

You can transcode the required rare characters through the transcoding software, and then add them to the project font library

Question6: How to deal with the first compilation of the program being extremely slow?

Answer6:

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

Question7: How to handle the display "waiting for download..." on the serial port after successfully ESP-IDF flashing?

Answer7:

If there is a reset button on the development board, press the reset button; if there is no reset button, please power it on again

Question8: What should I do if I can't find the AppData folder?

Answer8:

Some AppData folders are hidden by default and can be set to show.

English system: Explorer->View->Check "Hidden items"

Chinese system: File Explorer -> View -> Display -> Check "Hidden Items”

Question9: How do I check the COM port I use?

Answer9:

Windows system:

①View through Device Manager: Press the Windows + R keys to open the "Run" dialog box; input devmgmt.msc and press Enter to open the Device Manager; expand the "Ports (COM and LPT)" section, where all COM ports and their current statuses will be listed.

②Use the command prompt to view: Open the Command Prompt (CMD), enter the “mode” command, which will display status information for all COM ports.

③Check hardware connections: If you have already connected external devices to the COM port, the device usually occupies a port number, which can be determined by checking the connected hardware.

Linux system:

①Use the dmesg command to view: Open the terminal.

① Use the ls command to view: Enter ls /dev/ttyS* or ls /dev/ttyUSB* to list all serial port devices.

③ Use the setserial command to view: Enter setserial -g /dev/ttyS* to view the configuration information of all serial port devices.

Question10: Why does the program flashing fail when using a MAC device?

Answer10:

Install MAC Driver and flash again.