

1. Introduction to Testing Platform

Development Board : ESP32-WROOM-32E devKit

MCU : ESP32-32E module

Frequency : 240MHz

2. Pin connection instructions

The module can be directly plugged into the ESP32-32E development board, as shown in the following figure:

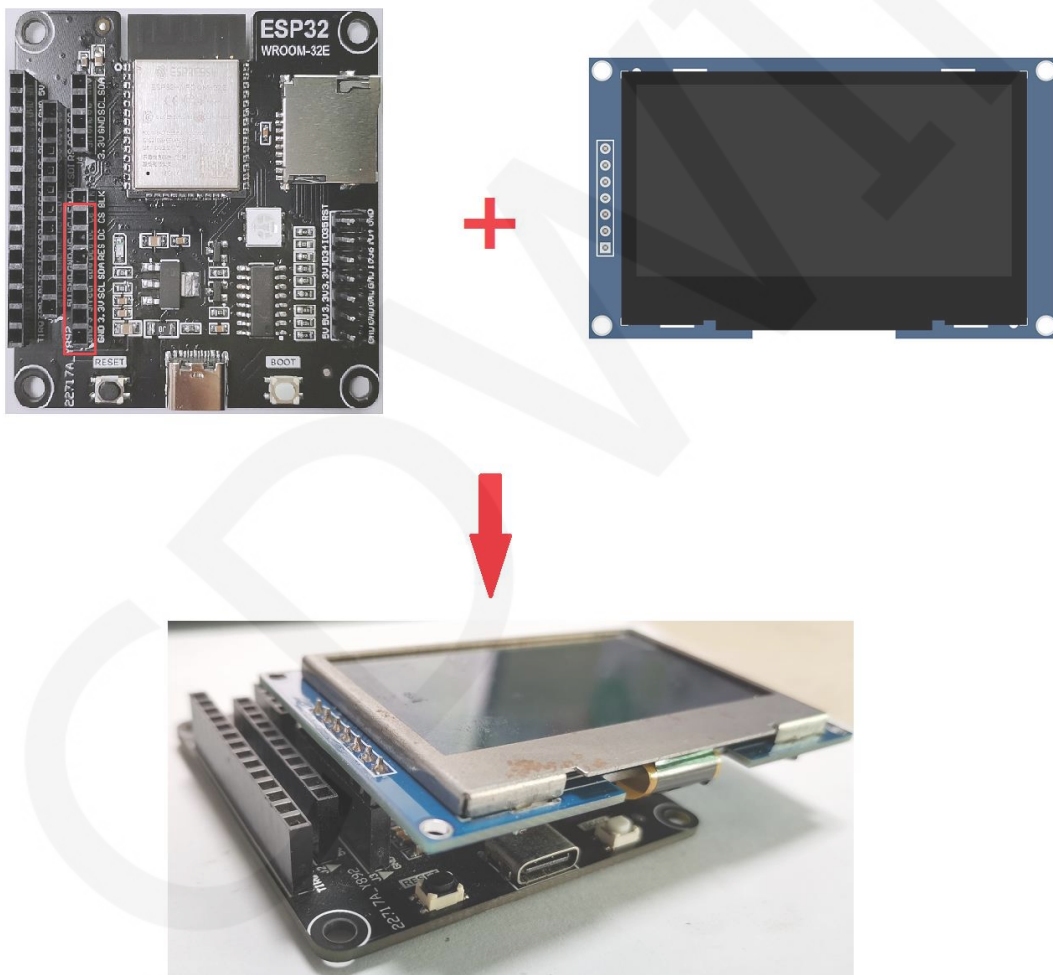


Figure 1: Module Inline ESP 32-32E Development Board

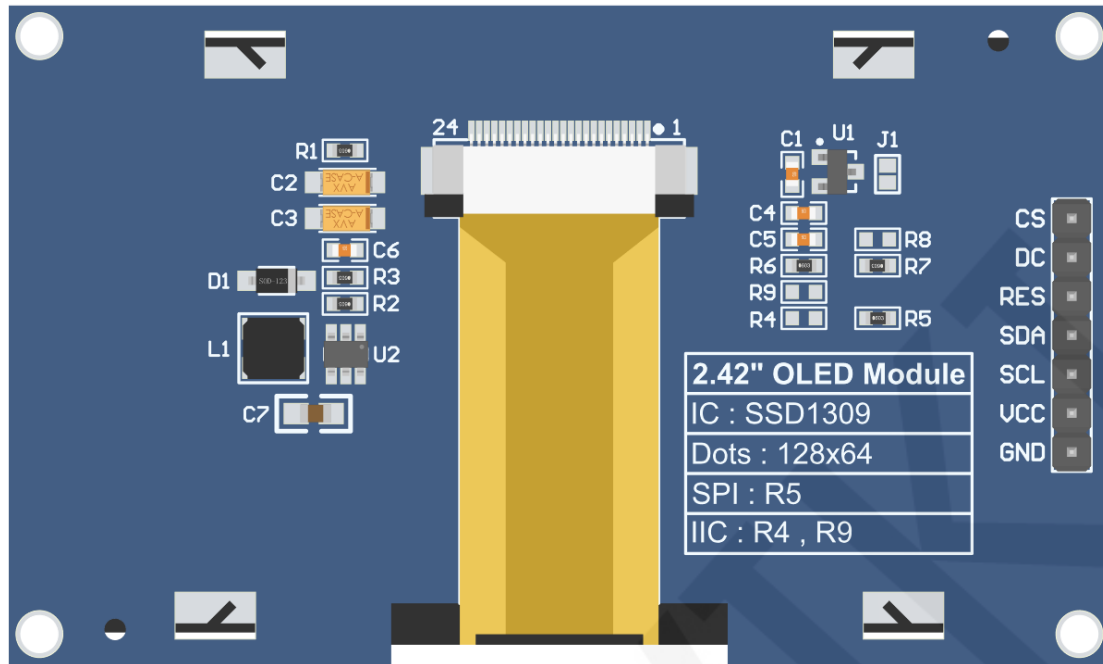


Figure 2 Module Back Pins

NOTE:

- A. Connect to a 5V microcontroller, which can short circuit J1 to keep the IO voltage and IO high level consistent;
- B. R8 is not soldered by default. If there is no need to control the CS pin, R8 solders the 0R resistor to keep the CS signal grounded;
- C. If SPI communication mode is selected, R5 will weld 0R resistor, and R4 and R9 will be disconnected;
- D. If IIC communication mode is selected, R4 and R9 will be welded with 0R resistor, and R5 will be disconnected;

ESP32-32E SPI Test Program Pin Direct Insertion Instructions			
Number	Module pins	Corresponding ESP32-32E development board wiring pins	Remarks
1	GND	GND	OLED screen power supply ground
2	VCC	5V/3.3V	OLED screen power supply positive
3	SCL	IO14	SPI bus clock signal
4	SDA	IO13	SPI bus write data signal

5	RES	IO27	OLED screen reset control signal, low-level reset
6	DC	IO2	OLED screen command/data selection control signal High level: data, low level: command
7	CS	IO15	OLED screen chip selection control signal, effective at low level (if welding R8, CS pin may not be connected)

ESP32-32E IIC Test Program Pin Direct Insertion Instructions

Number	Module pins	Corresponding ESP32-32E development board wiring pins	Remarks
1	GND	GND	OLED screen power supply ground
2	VCC	5V/3.3V	OLED screen power supply positive
3	SCL	IO14	IIC bus clock signal
4	SDA	IO13	IIC bus data signal
5	RES	IO27/3.3V	OLED screen reset control signal, low-level reset (if no control is required, the RES pin can be connected to a high-level (3.3V))
6	DC	IO2/GND/3.3V	IIC bus selects signal from device address When connecting to the IO2 pin, IO2 is low level: 0x78, and IO2 is high level: 0x7A Low level (connected to GND): 0x78, high level (connected to 3.3V): 0x7A
7	CS	IO15/GND	OLED screen chip selection control signal, effective at low levels When using IIC communication, there is no need for control. When connecting to IO15, it must be set to low level or GND can be connected (such as welding R8, CS pin can not be connected)

3. Demo Function Description

This sample program uses the ESP32 hardware HSPI bus and includes SPI and IIC test programs. Each test program includes hardware and software functional testing, which is located in **Demo_ESP32** directory, as shown in the following figure:



✧ Description of sample program content

The testing program includes the following test items:

- A. Example01-graph_Test is a graphical display test
- B. Example02 string_Test is a character display test;
- C. Example03 show_BMP is a BMP bitmap display test;

✧ Example program IIC slave device address modification instructions (only for IIC test programs)

Open any IIC sample program and locate the **setup** function. If using the 0x7A slave device address, there is no need to annotate the two lines of code

digitalWrite(OLED_DC, HIGH) and **u8g2.setI2CAddress(0x7A)** (to make them effective). If using the 0x78 slave device address, the two lines of code

digitalWrite(OLED_DC, HIGH) and **u8g2.setI2CAddress(0x7A)** need to be annotated (to make them ineffective), as shown in the following figure:

```
void setup(void) {  
  pinMode(OLED_DC, OUTPUT);  
  digitalWrite(OLED_DC, LOW);  
  /*When using 0x7A slave device address, please use the following definition*/  
  //digitalWrite(OLED_DC, HIGH);  
  //u8g2.setI2CAddress(0x7A);  
  pinMode(OLED_CS, OUTPUT);  
  digitalWrite(OLED_CS, LOW);  
  Wire.begin(/*SDA*/ SDA, /*SCL*/ SCL);  
  u8g2.begin();  
}
```

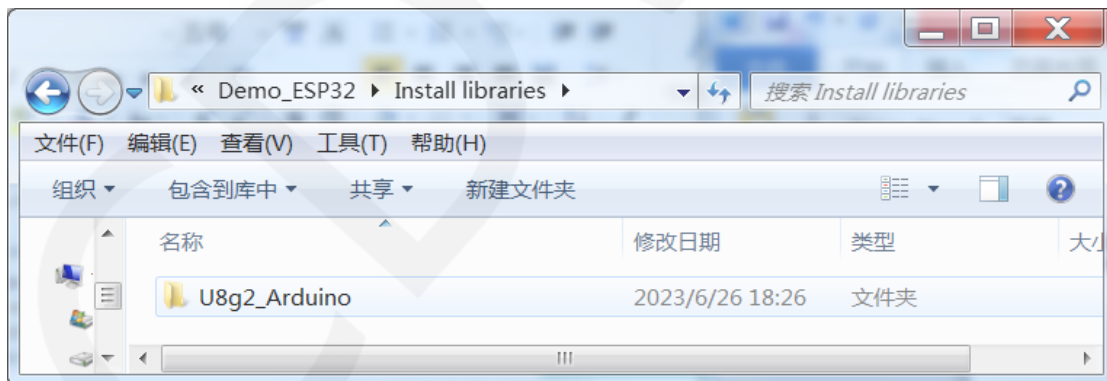
4. Demo Usage Instructions

✧ Building Development Environment

For specific methods of building a development environment, please refer to the "Arduino_development_environment_construction_for-ESP32-EN" document in this directory.

✧ Installing software library

After the development environment is set up, the software library used by the sample program needs to be copied to the project library directory so that the sample program can be called. The software library is located in the **Install libraries** directory, as shown in the following figure:

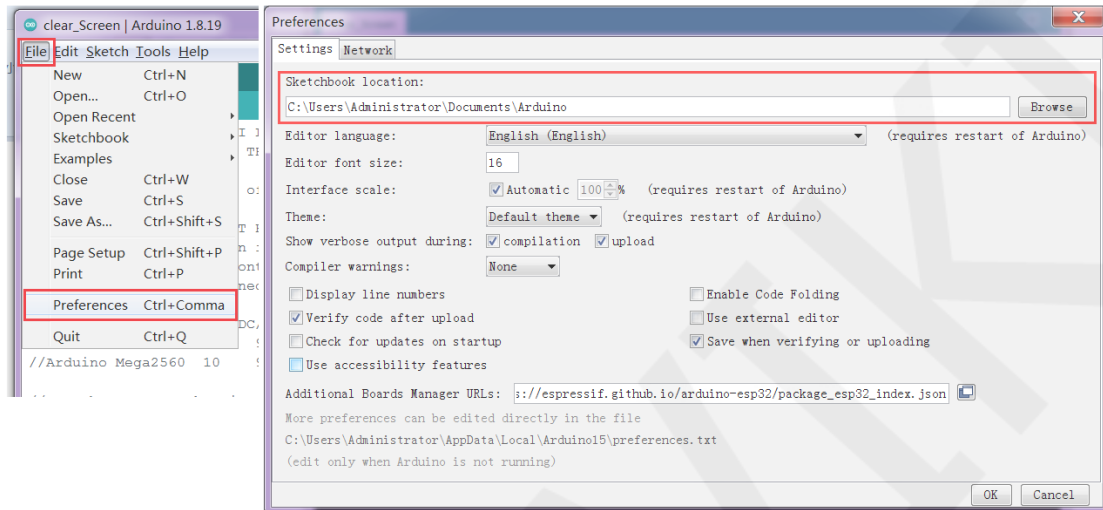


You can also download the latest software library from Github and unzip it (for easy differentiation, you can rename the unzipped folder, as shown in the **Install libraries** directory), and then copy it to the engineering library directory. The download address is as follows:

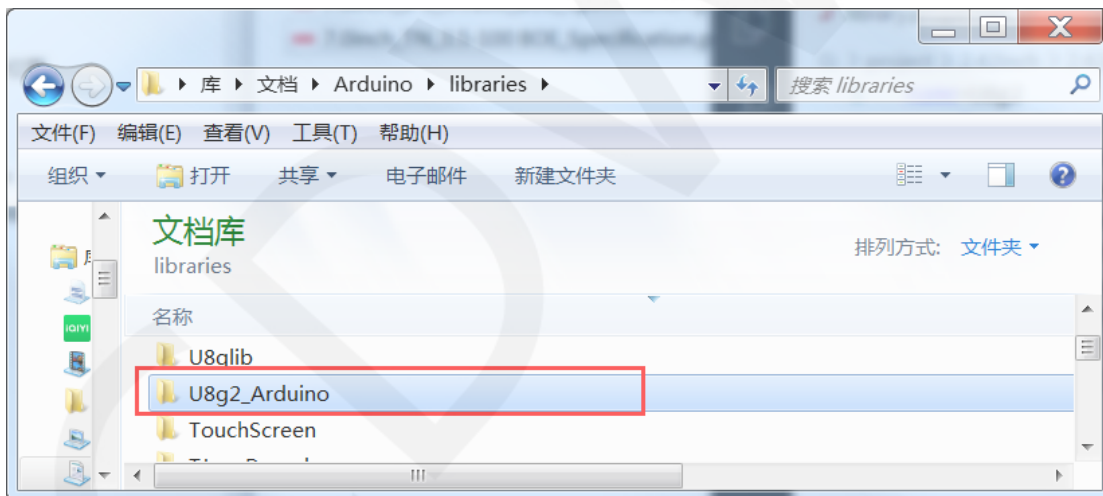
https://github.com/olikraus/U8g2_Arduino

These software library have been configured and can be directly copied to the project

library directory for use. The default path for the engineering library directory is **C:\Users\Administrator\Documents\Arduino\libraries**. You can also change the project library directory: open the Arduino IDE software, click **File -> Preferences**, and reset the **Sketchbook location** in the pop-up interface, as shown in the following figure:



Copy the software library to the project library directory, as shown in the following figure:



❖ Compile and Run Programs

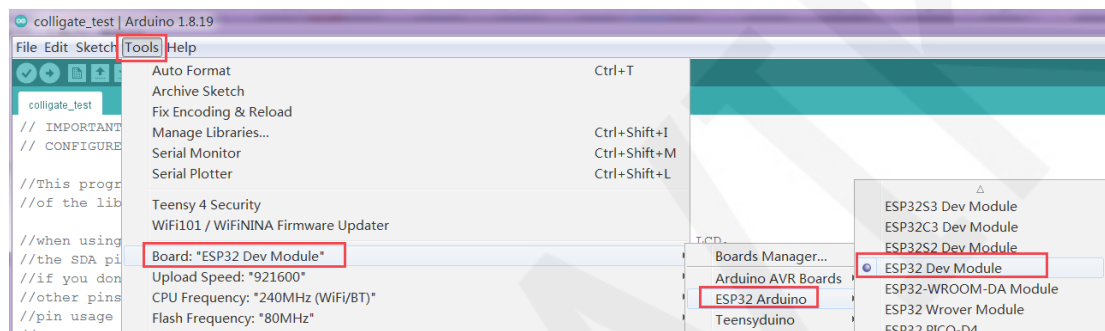
After the library installation is completed, the sample program can be compiled and run as follows:

- A. Plug the display module directly into the ESP32 development board, and connect the development board to a PC to power on;
- B. Open Any sample program in the **Demo_ESP32** directory, as shown in the following figure (Here is the Example01-graph_test of the hardware SPI testing

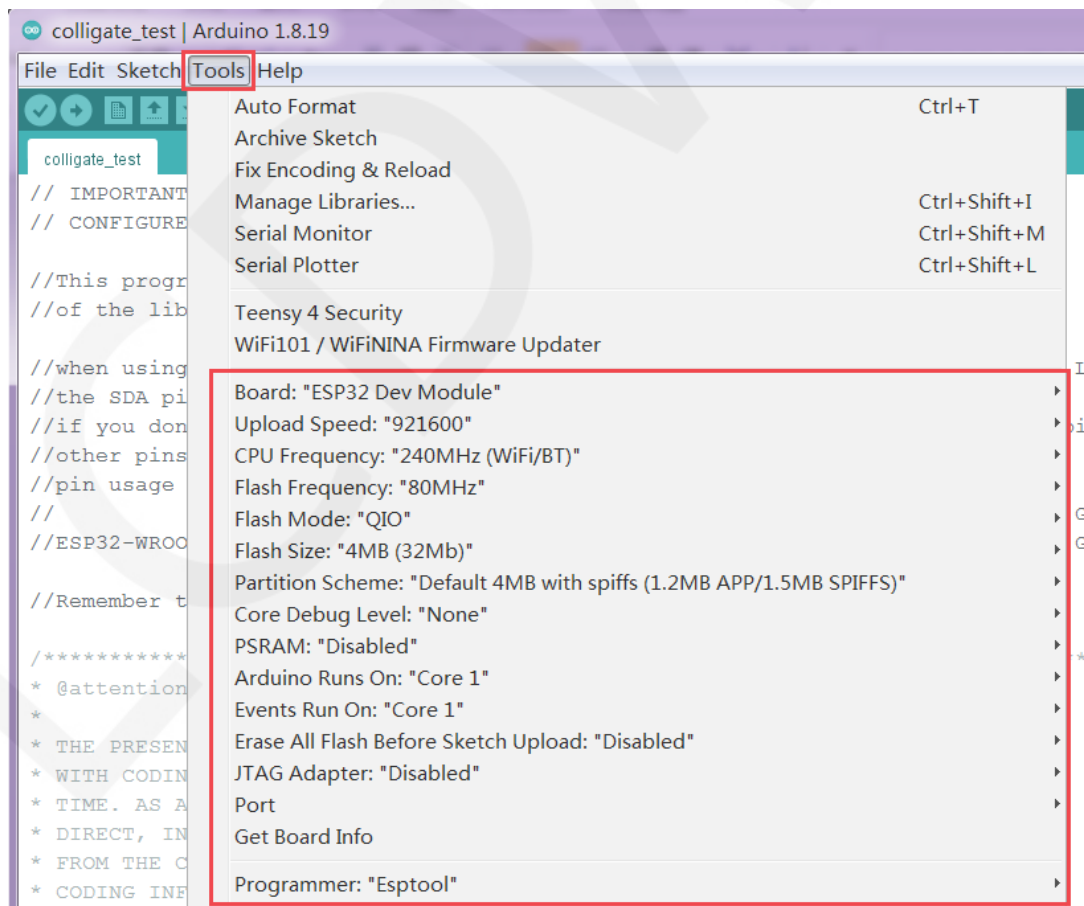
program as an example) as shown in the following figure::



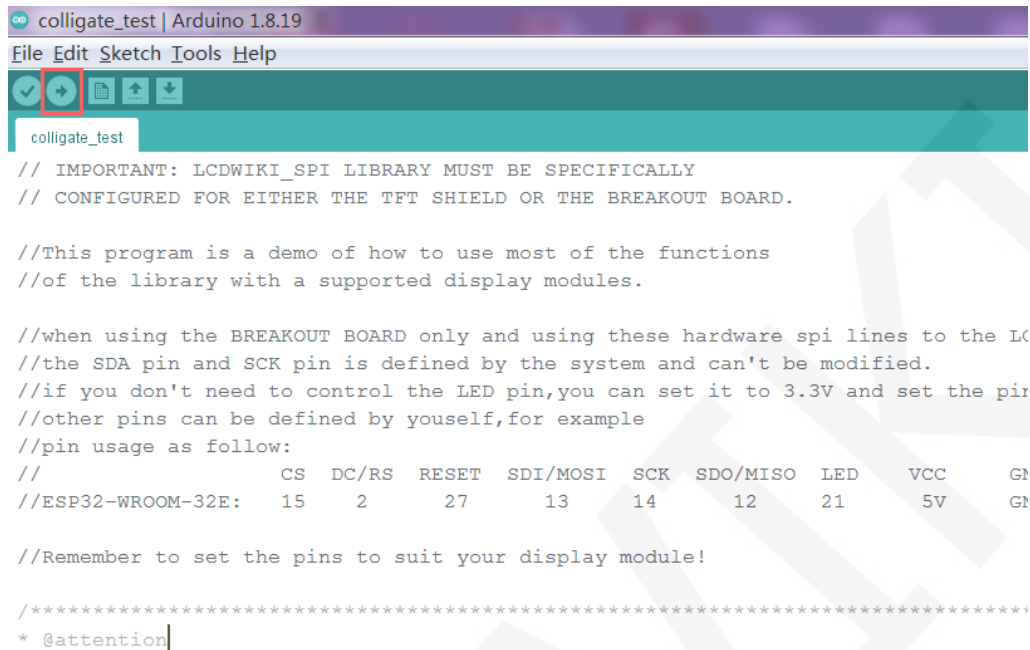
C. After opening the sample program, select the ESP32 device, as shown in the following figure:



D. Configure ESP32 Flash, PSRAM, ports, etc. as shown in the following figure:



- E. Click the **upload** button to compile and download the program, as shown in the following figure:



```

colligate_test | Arduino 1.8.19
File Edit Sketch Tools Help
colligate_test
// IMPORTANT: LCDWIKI_SPI LIBRARY MUST BE SPECIFICALLY
// CONFIGURED FOR EITHER THE TFT SHIELD OR THE BREAKOUT BOARD.

//This program is a demo of how to use most of the functions
//of the library with a supported display modules.

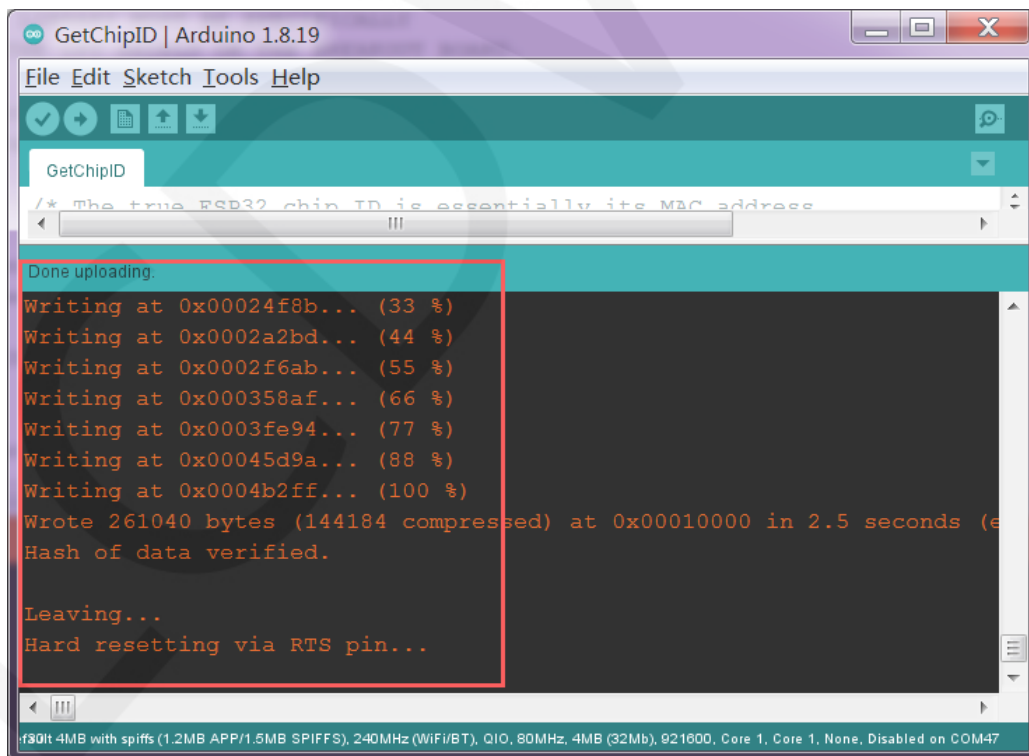
//when using the BREAKOUT BOARD only and using these hardware spi lines to the IC
//the SDA pin and SCK pin is defined by the system and can't be modified.
//if you don't need to control the LED pin, you can set it to 3.3V and set the pin
//other pins can be defined by yourself, for example
//pin usage as follow:
//
//          CS DC/RS  RESET  SDI/MOSI  SCK  SDO/MISO  LED   VCC   GND
//ESP32-WROOM-32E:  15   2    27    13      14   12      21   5V    GND

//Remember to set the pins to suit your display module!

/*****
* @attention

```

- F. If the following prompt appears, it indicates that the program has been compiled and downloaded successfully, and has already been run:



```

GetChipID | Arduino 1.8.19
File Edit Sketch Tools Help
GetChipID
/* The true ESP32 chip ID is essentially its MAC address

Done uploading.
Writing at 0x00024f8b... (33 %)
Writing at 0x0002a2bd... (44 %)
Writing at 0x0002f6ab... (55 %)
Writing at 0x000358af... (66 %)
Writing at 0x0003fe94... (77 %)
Writing at 0x00045d9a... (88 %)
Writing at 0x0004b2ff... (100 %)
Wrote 261040 bytes (144184 compressed) at 0x00010000 in 2.5 seconds (effective 102400 bytes/s)
Hash of data verified.

Leaving...
Hard resetting via RTS pin...

f30it 4MB with spiiffs (1.2MB APP/1.5MB SPIFFS), 240MHz (WiFi/BT), QIO, 80MHz, 4MB (32Mb), 921600, Core 1, Core 1, None, Disabled on COM47

```

- G. If the display module displays content, it indicates that the program has run successfully.