

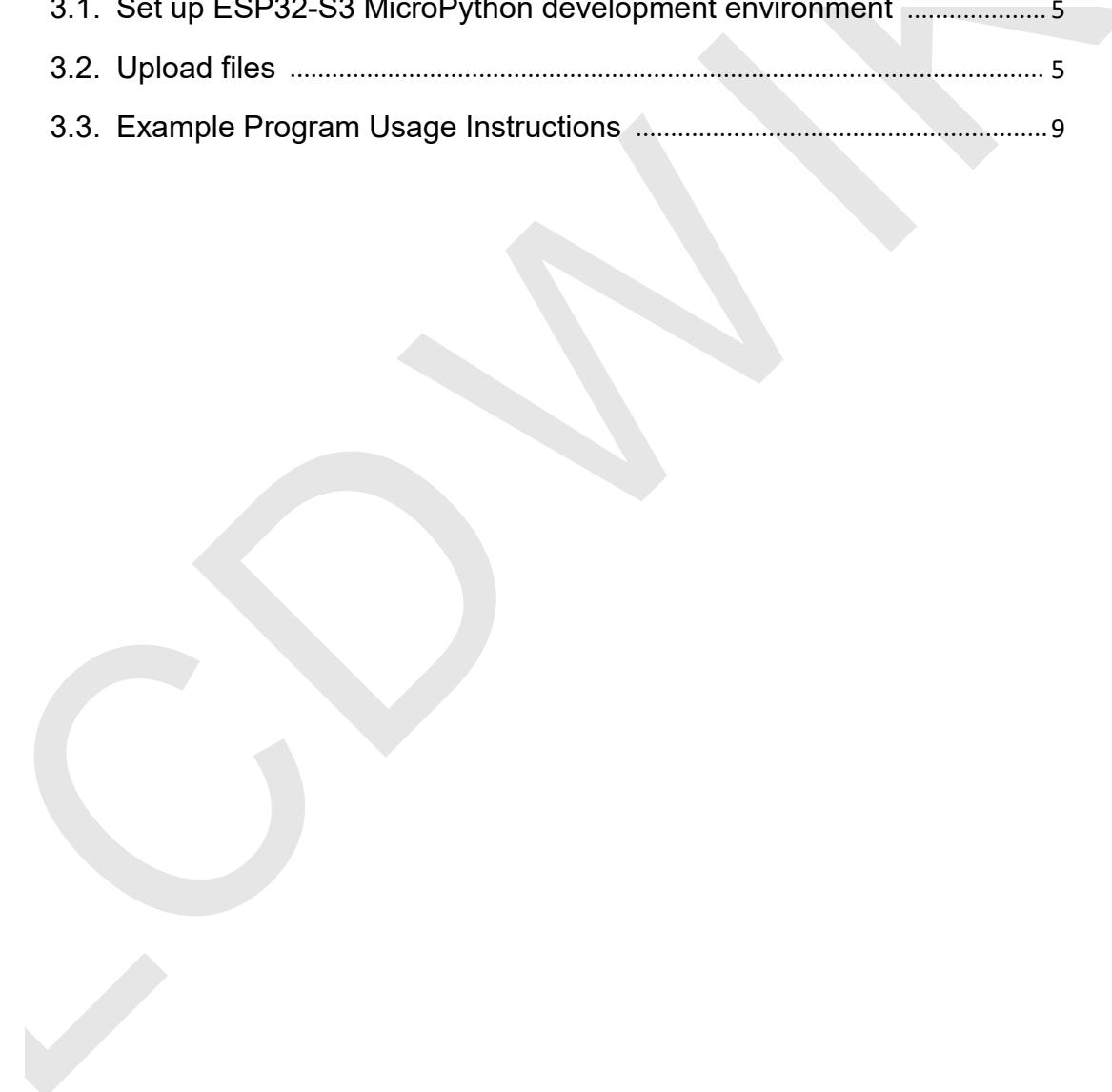
ES3C28P&ES3N28P

2.8inch MicroPython

Demo Instructions

CONTENTS

1. Software and hardware platform description	3
2. Pin allocation instructions	3
3. Instructions for the example program	5
3.1. Set up ESP32-S3 MicroPython development environment	5
3.2. Upload files	5
3.3. Example Program Usage Instructions	9



1. Software and hardware platform description

Module: 2.8-inch ESP32-S3 display module with 240x320 resolution and ILI9341V screen driver IC.

Module master: ESP32-S3 chip, the highest main frequency 240MHz, support 2.4G WIFI+ Bluetooth.

Thonny version: 4.1.6

ESP32 MicroPython firmware version: 1.25.0

2. Pin allocation instructions

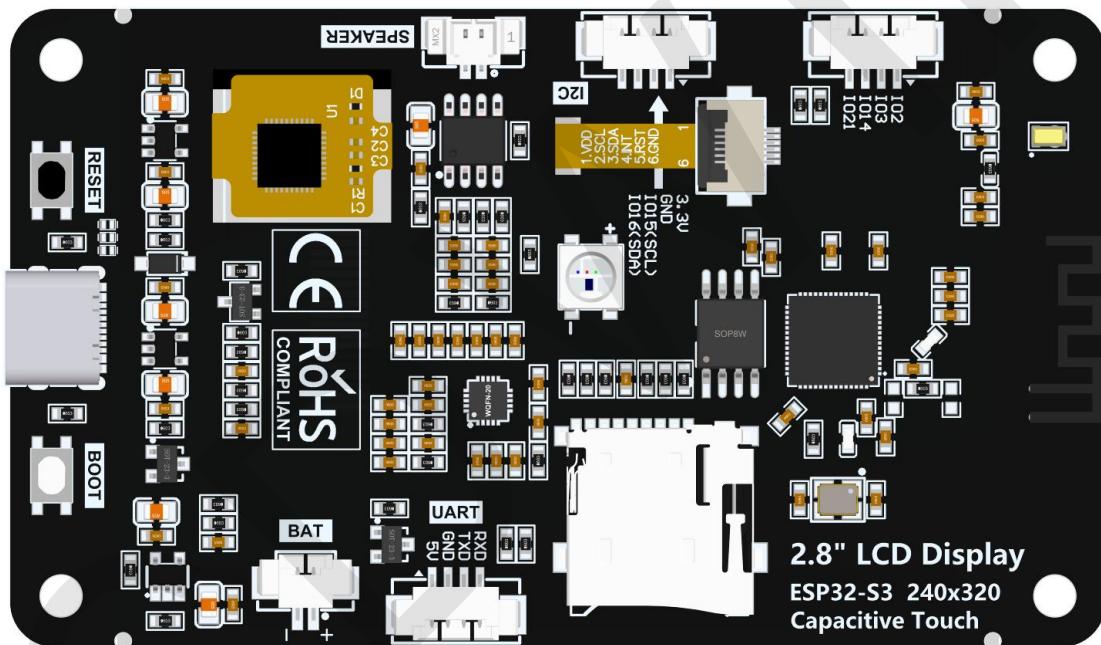


Figure 2.1 Rear view of 2.8-inch ESP32-S3 display module

The main controller of the 2.8-inch ESP32-S3 display module is ESP32-S3 chip, and the GPIO allocation for its onboard peripherals is shown in the table below:

ESP32-S3 chip pin allocation instructions			
On board device	On board device pins	ESP32-S3 connection pin	description
LCD	TFT_CS	IO10	LCD screen chip selection control signal, low level effective
	TFT_RS	IO46	LCD screen command/data selection control signal. High level: data, low level: command

	TFT_SCK	IO12	LCD SPI bus clock signal
	TFT_MOSI	IO11	LCD SPI bus writes data signals
	TFT_MISO	IO13	LCD SPI bus reading data signal
	TFT_RST	CHIP_PU	LCD screen reset control signal, low level reset (shared reset pin with ESP32-S3 main control)
	TFT_BL	IO45	LCD screen backlight control signal (high level lights up the backlight, low level turns off the backlight)
CTP	TP_SDA	IO16	Capacitive touch screen I2C bus data signal
	TP_SCL	IO15	Capacitive touch screen I2C bus clock signal
	TP_RST	IO18	Capacitive touch screen reset control signal, low level reset
	TP_INT	IO17	Capacitive touch screen interrupts the input signal, and when a touch event occurs, it inputs a low level
LED	RGB_INT	IO42	Single line RGB three color LED light, which can light up the internal red, green, and blue light beads according to different signals
SDCARD	SD_CLK	IO38	SD card SDIO bus clock signal
	SD_CMD	IO40	SD card SDIO bus command signal
	SD_D0	IO39	SD card SDIO bus data signal (DATA0~DATA3 four data lines)
	SD_D1	IO41	
	SD_D2	IO48	
	SD_D3	IO47	
BATTERY	BAT_ADC	IO9	Battery voltage ADC value acquisition input signal
Audio	Audio_EN	IO1	Audio output enable signal, low-level enable, high-level disable
	I2S_MCK	IO4	Audio I2S bus master clock signal
	I2S_SCK	IO5	Audio I2S bus bit clock signal
	I2S_DO	IO6	Audio I2S bus bit output data signal
	I2S_LRC	IO7	Audio I2S bus left and right channel

			selection signals. High level: right channel; Low level: Left channel
	I2S_DI	IO8	Audio I2S bus bit input data signal
KEY	BOOT_KEY	IO0	Download mode selection button (press and hold the button to power on, then release it to enter download mode)
	RESET_KEY	EN	ESP32-23E reset button, low level reset (shared with LCD screen reset)
USB	USB_N	IO19	USB bus differential signal data line negative pole
	USB_P	IO20	Positive pole of USB bus differential signal data line
Serial Port	TX0	IO43	ESP32-S3 serial port 0 sending signal
	RX0	IO44	ESP32-S3 serial port 0 receiving signal
POWER	TYPE-C_POWER	/	Type-C power interface, connected to 5V voltage

Table 2.1 Pin allocation instructions for ESP32-S3 onboard peripherals

3. Instructions for the example program

3.1. Set up ESP32-S3 MicroPython development environment

For detailed instructions on setting up the

["MicroPython_development_environment_construction_for_ESP32-S3"](#), please refer to the document

3.2. Upload files

After the development environment is set up, the relevant files need to be uploaded to the ESP32-S3 device in order to run the testing program.

Before uploading the file, please familiarize yourself with the directory contents of the MicroPython sample program. Open the "1-示例程序_Demo\MicroPython" directory in the package, as shown in the following figure:

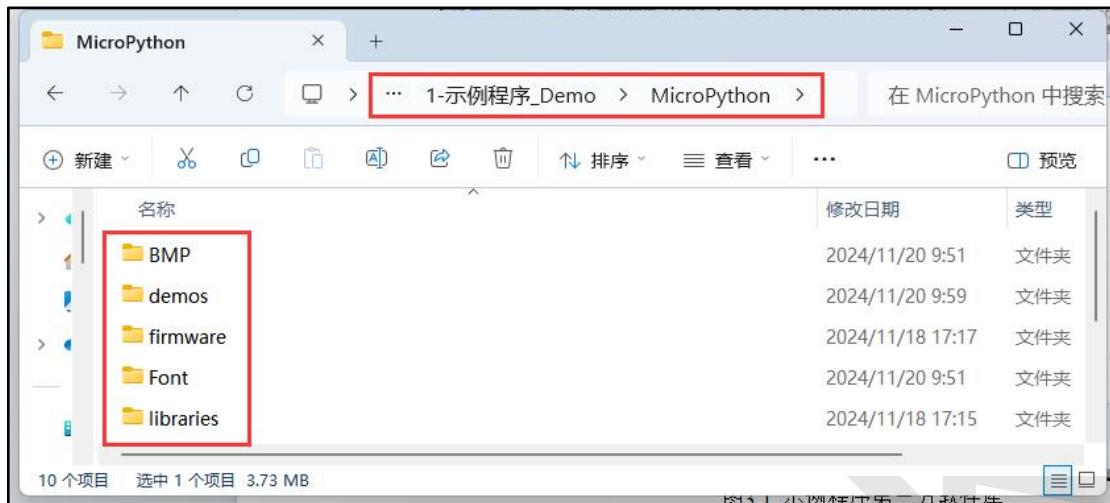


Figure 3.1 MicroPython Example Program Catalog

The contents of each folder are described as follows:

BMP: Stores BMP format images that sample programs need to use.

demos: Contains sample programs

firmware: Stores MicroPython firmware (needs to be burned when setting up the development environment)

Font: Stores the Chinese and English character modulo data that the sample program needs to use.

libraries: Stores MicroPython library files that sample programs need to use

After understanding the directory contents of the MicroPython sample program, the next step is to upload the program file to the ESP32-S3 device. The steps are as follows:

- A. Connect the ESP32-S3 display module to the computer and power it on using a USB cable.
- B. Open the Thonny software and configure the MicroPython interpreter for ESP32-S3, as shown in the following figure:
(If already configured, this step can be omitted)

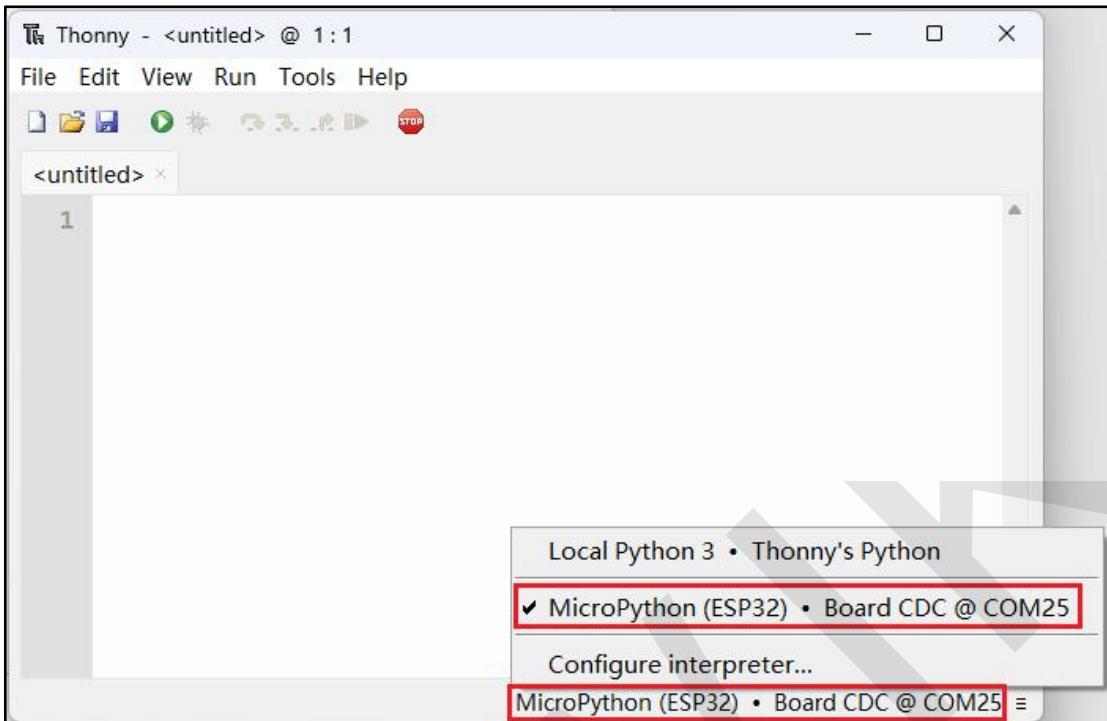


Figure 3.2 Selecting MicroPython interpreter

C. Click the toolbar button to connect the ESP32-S3 device. If the following prompt appears in the shell information bar, it indicates that the device connection is successful.

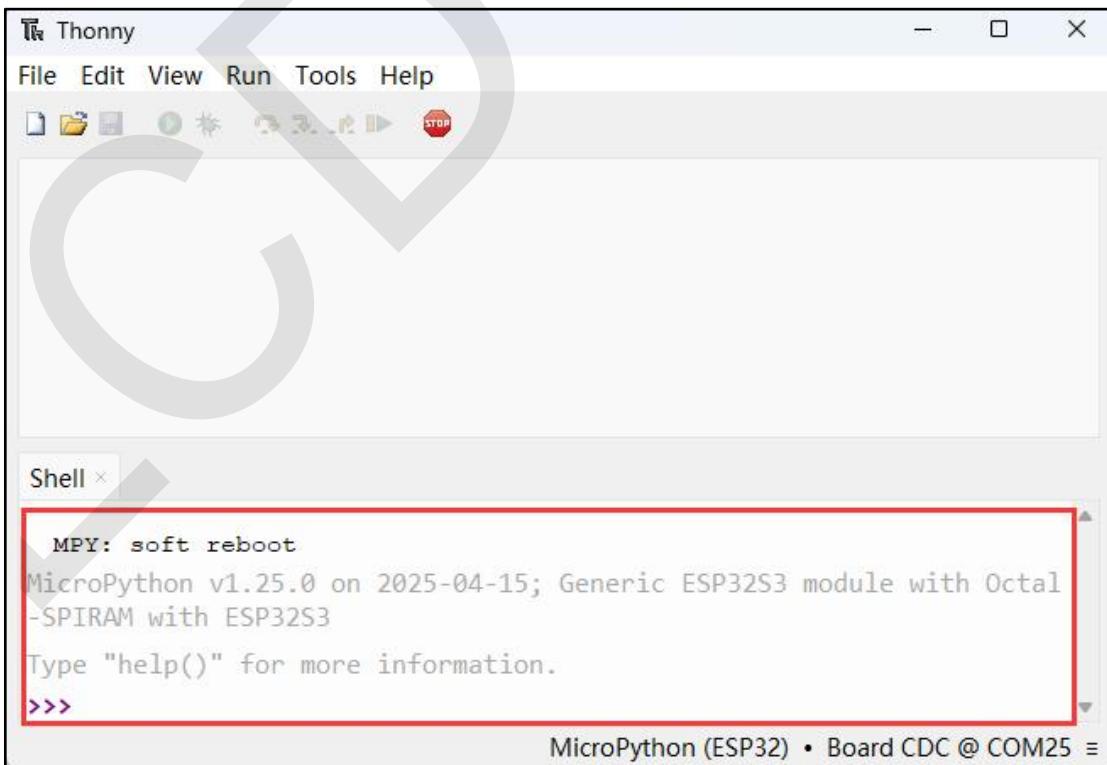


Figure 3.3 Connecting ESP32-S3 devices

D. Click the "View ->Files" button to open the file window (ignore this operation if it is already open). Find the "1-示例程序_Demo\MicroPython" directory in the package in the window, left click the mouse to select the target file in the directory, and right-click on the standalone mouse to select "Upload to /" to upload the target file. As shown in the following figure:

Please note that when uploading files, ESP32-S3 cannot run any programs, otherwise the upload will fail

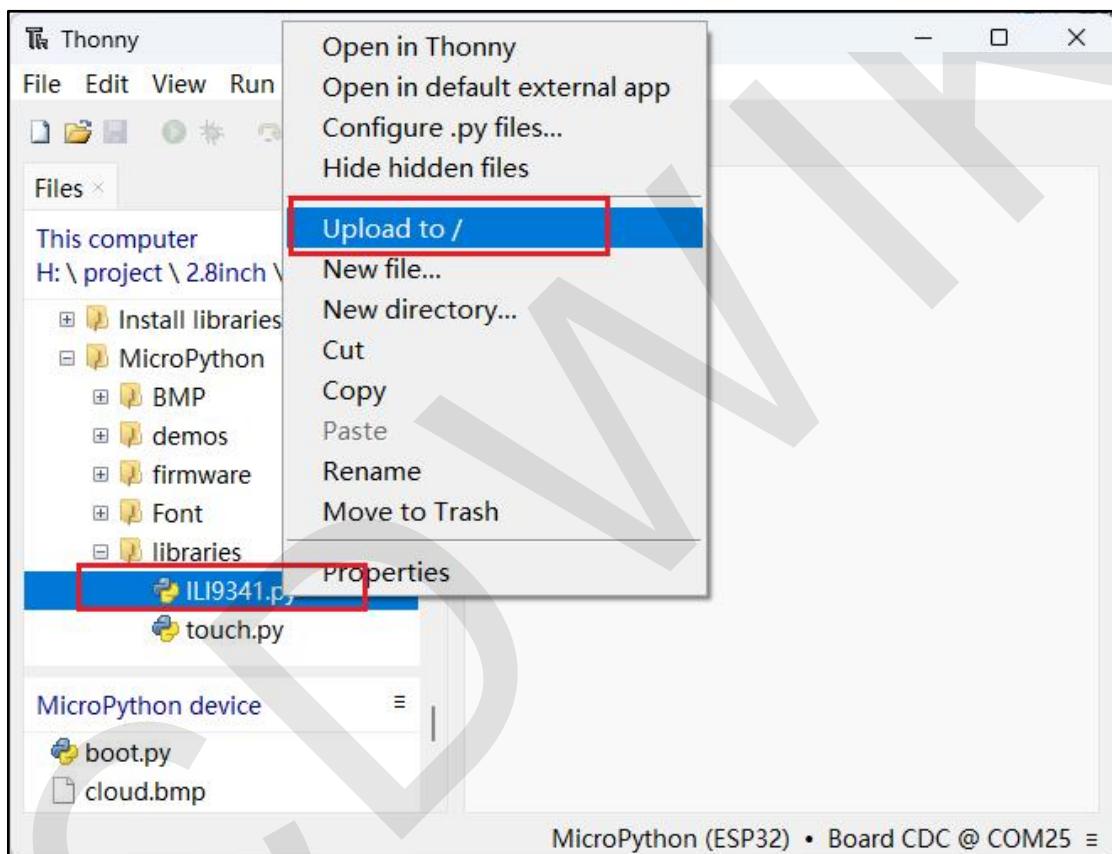


Figure 3.4 Uploading Files to ESP32-S3 Devices

E. Upload the files from the "BMP", "Font", and "libraries" directories to the ESP32 device using the above method. The files in the 'demos' directory can be transferred or not. As shown in the following figure:

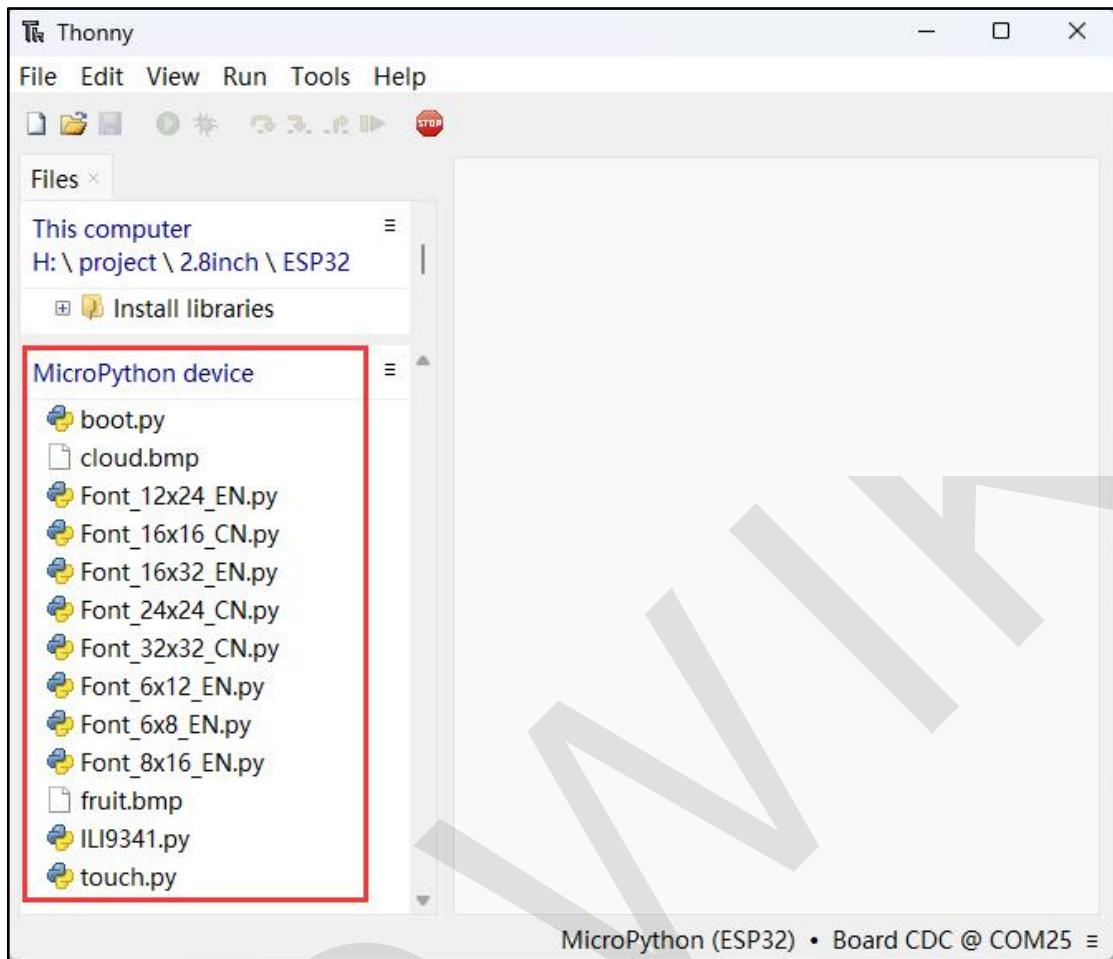


Figure 3.5 Completed file upload

3.3. Example Program Usage Instructions

The sample program is located in the "1-示例程序_Demo\MicroPython\demos"

directory of the package, as shown in the following figure:

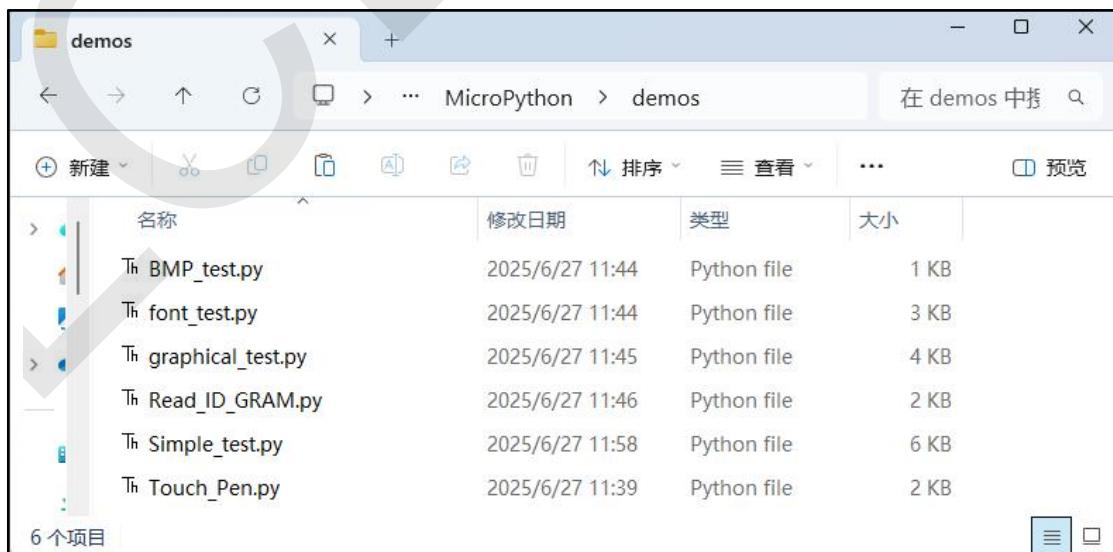


Figure 3.6 Example Program

The sample program can be uploaded to an ESP32-S3 device to open and run, or it can be opened and run on a local computer. If you need to power on the ESP32-S3 display module to run automatically, you need to change the sample program name to "main.py" and upload it to the ESP32-S3 display module.

In the Python software, open the target sample program, click the menu bar



button, and you can run it. If the operation fails, the ESP32-S3 device needs to be reconnected. The introduction of each example program is as follows:

BMP_test.py

This example program relies on the ILI9341.py library to display images in BMP format

font_test.py

This example program relies on the ILI9341.py library to display Chinese and English characters of various sizes. The font modeling data needs to be saved in the font file according to the relevant format. For instructions on character casting, please refer to the following website:

http://www.lcdwiki.com/Chinese_and_English_display_modulo_settings

graphical_test.py

This example program relies on the ILI9341.py library to display graphics such as points, lines, rectangles, rounded rectangles, triangles, circles, ellipses, etc. for drawing and filling, as well as setting display orientation.

Read_ID_GRAM.py

This example program relies on the ILI9341.py library to display LCD ID and RGAM color value readings.

Simple_test.py

This example does not rely on any software libraries and displays simple screen scrolling content.

Touch_Pen.py

This example relies on the ILI9341.py library and the touch.py library, displaying the operation of drawing dots and lines on the touch screen.