

FireBeetle 2 ESP32-C5开发板Arduino环境多库点屏大作战

作者：HonestQiao / 乔楚
日期：2025-10-03
首发：<https://mc.dfrobot.com.cn/thread-398279-1-1.html>

本文从硬件认识到软件编码，详细介绍了在FireBeetle 2 ESP32-C5开发板上，基于Arduino环境进行点屏的多种方式。

本文以FireBeetle 2 ESP32-C5开发板和GDI显示屏为基础展开，但本质上可用于任何Arduino环境中ESP32和显示屏的使用参考。

另外，需要说明的是，我当前使用的Arduino IDE是2.3.6版本的，但本文中使用Arduino IDE的方法，在Arduino 1.8.19上基本类似。

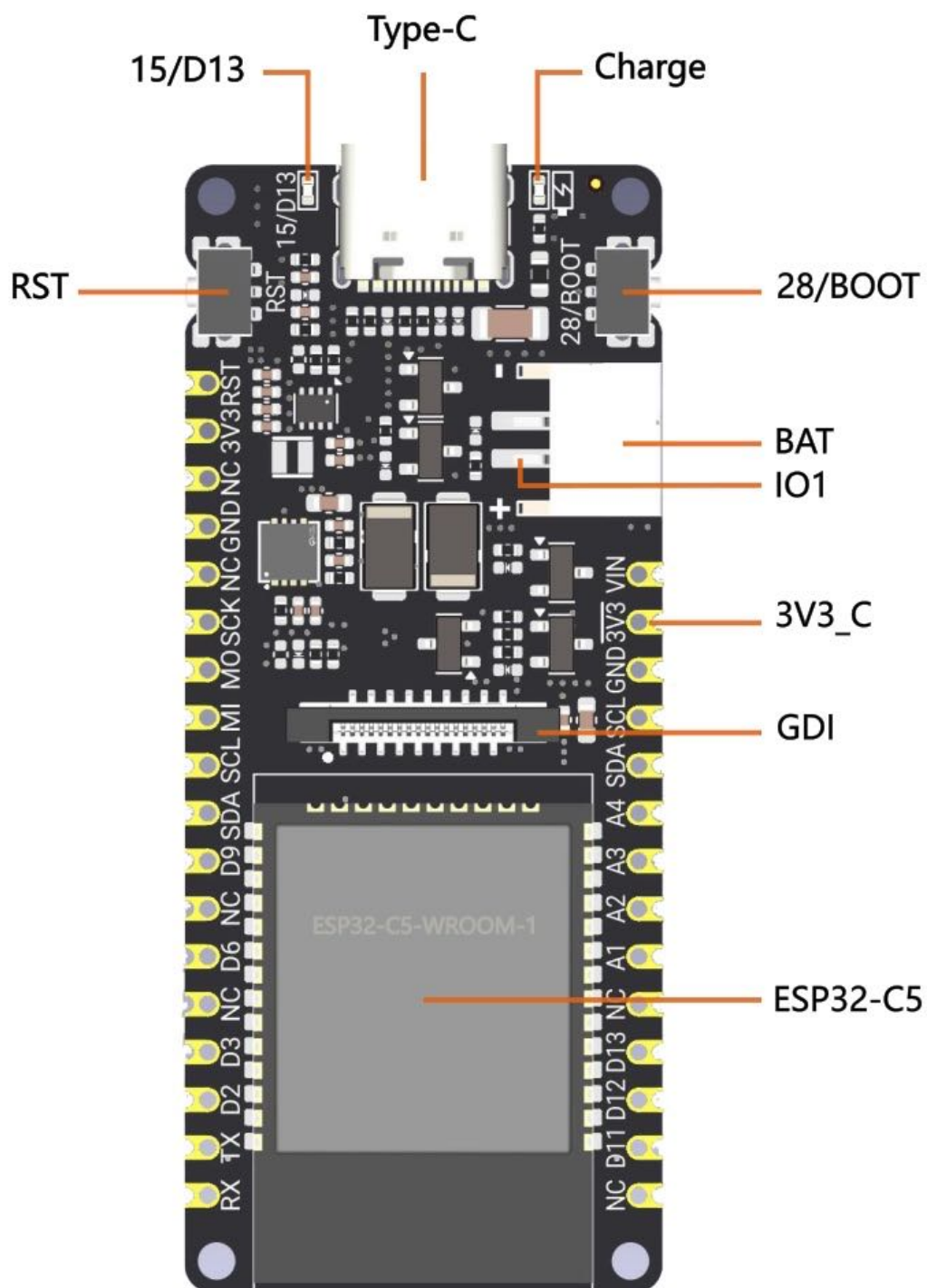
一、硬件了解

1. GDI接口了解

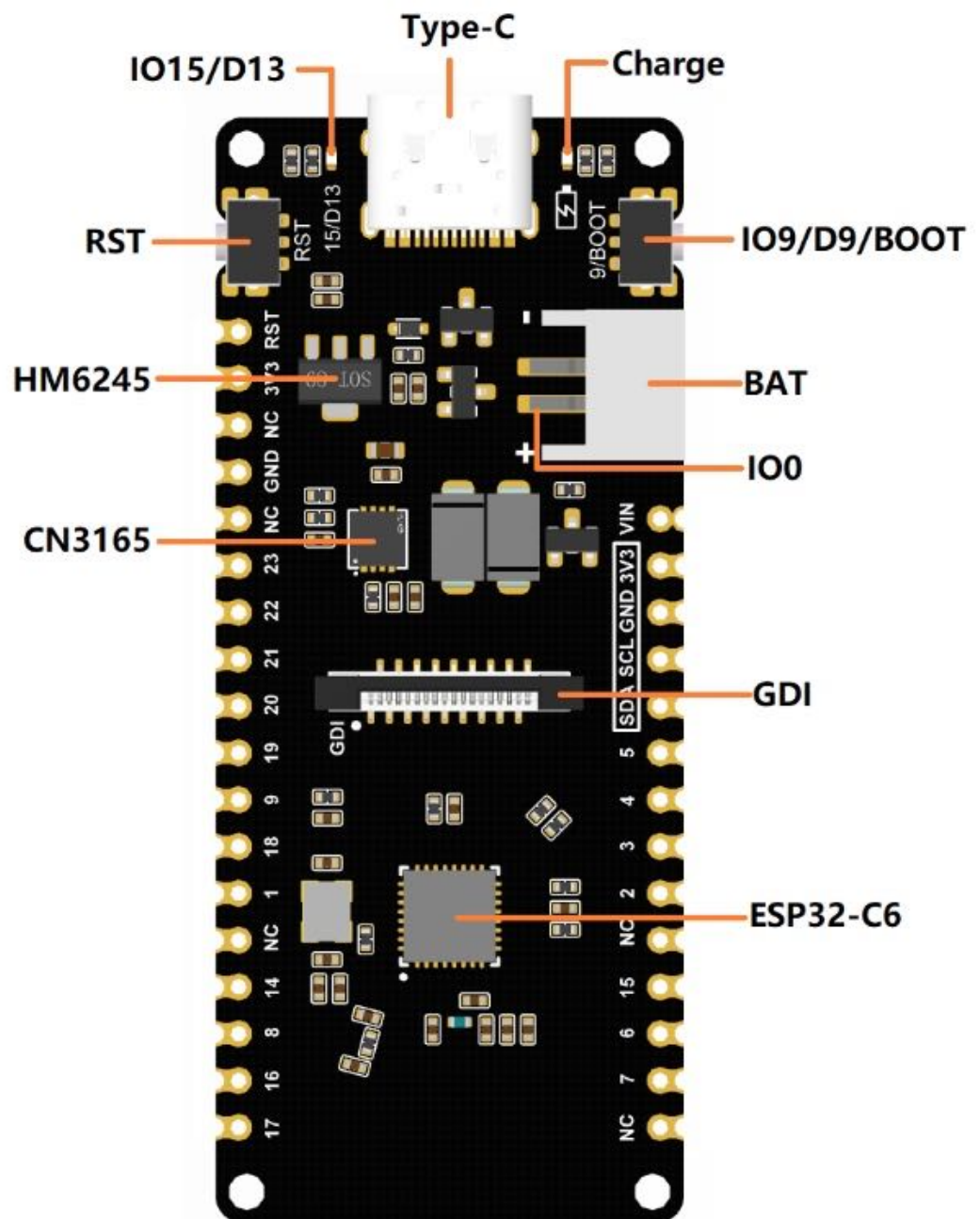
在很多DFRobot的FireBeetle和Beetle系列开发板或扩展板上，都提供了一个名为的GDI接口：

开发板	图片
Beetle ESP32 C3	

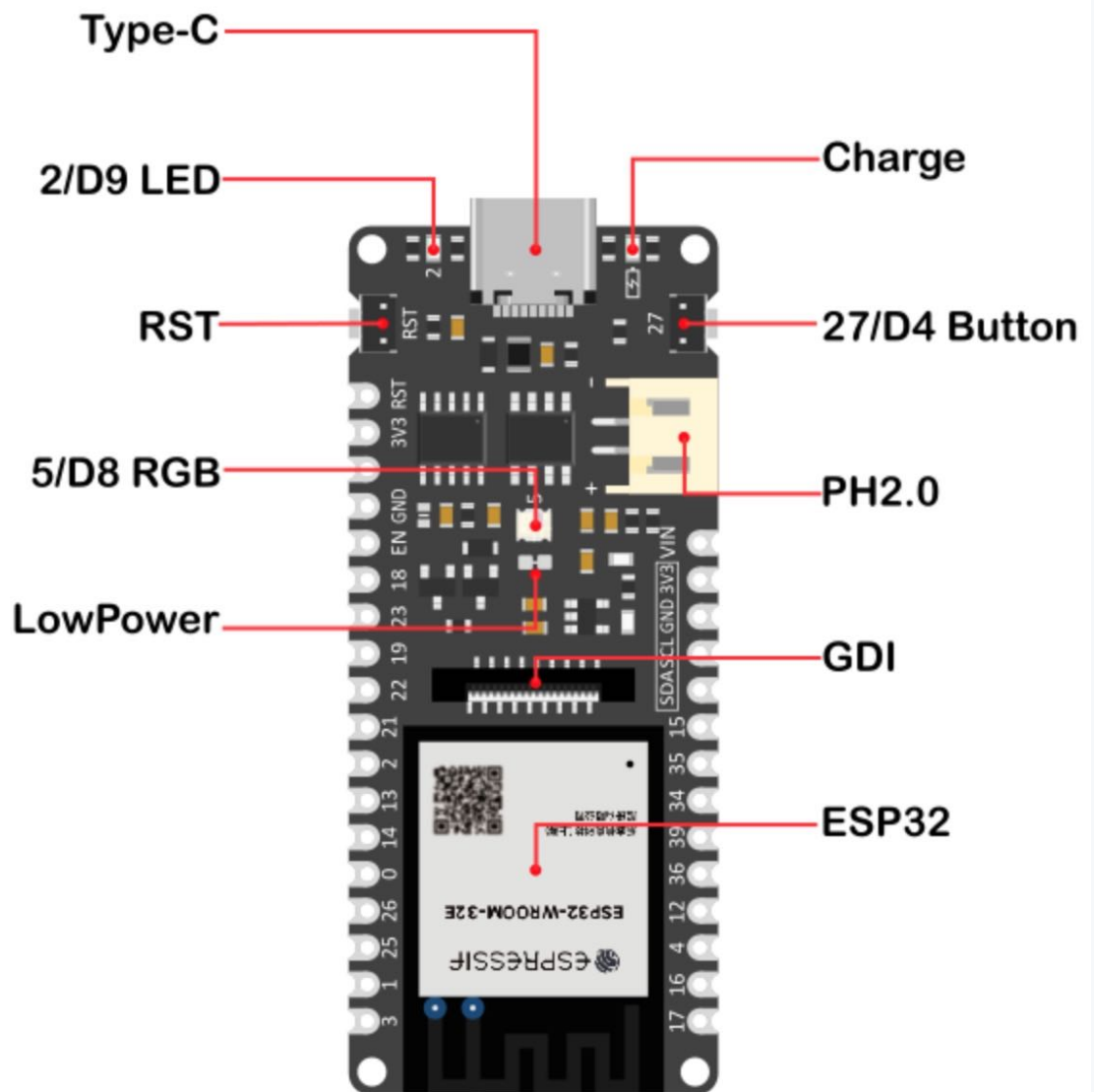
FireBeetle
2 ESP32
C5



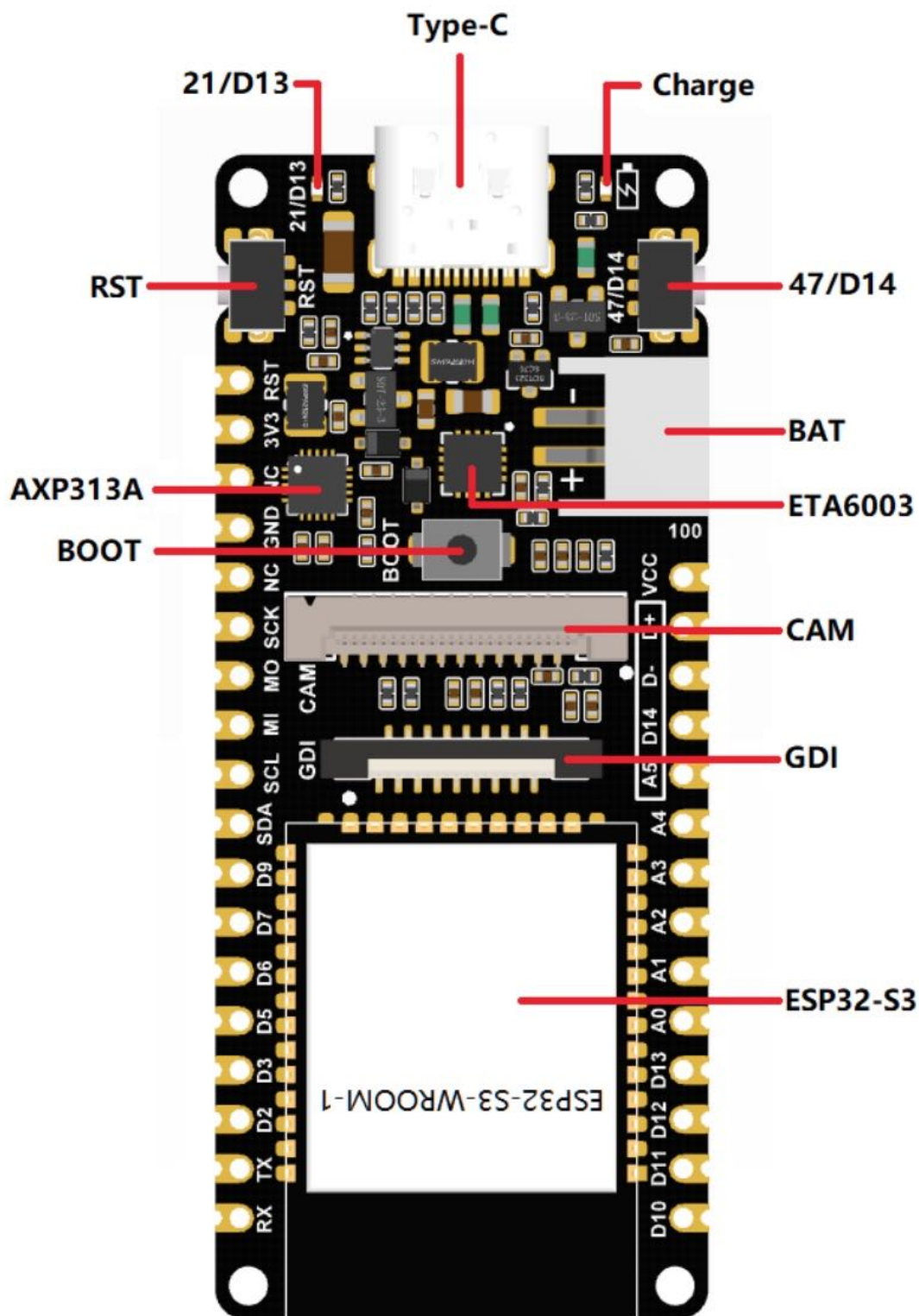
FireBeetle
2 Board
ESP32
C6



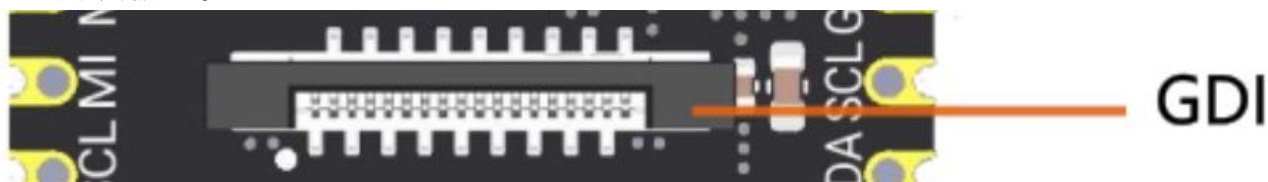
FireBeetle
Board
ESP32 E



FireBeetle
2 Board



GDI接口全名叫GDI显示接口，此接口为DFRobot专用GDI显示屏接口，使用18pin-FPC线连接屏幕，连接屏幕更加便捷，同样一块支持GDI接口的显示屏，可以连接到任意一块提供GDI接口的开发板上。



使用FPC线连接支持GDI接口的屏幕时，通常只需要根据不同主控配置对应的引脚即可。

这里拿较早的FireBeetle Board ESP32 E和最新的FireBeetle 2 ESP32 C5(本文写作时)进行对比：

FPC PINS	FireBeetle ESP32 PINS	Description	FPC PINS	Firebeetle 2 ESP32-C5 PINS	Description
VCC	3V3	3.3V	VCC	3V3	3V3
BLK (PWM调光)	12/D13	背光	LCD_BL	15/D13	背光
GND	GND	GND	GND	GND	GND
SCLK	18/SCK	SPI时钟	SCLK	23/SCLK	SPI时钟
MOSI	23/MOSI	主机输出, 从机输入	MOSI	24/MOSI	主机输出, 从机输入
MISO	19/MISO	主机输入, 从机输出	MISO	25/MISO	主机输入, 从机输出
DC	25/D2	数据/命令	LCD_DC	8/D2	数据/命令
RES	26/D3	复位	LCD_RST	26/D3	复位
CS	14/D6	TFT片选	LCD_CS	27/D6	TDT片选
SDCS	13/D7	SD卡片选	SD_CS	3/A2	SD卡片选
FCS	0/D5	字库	FCS	NC	字库片选
TCS	4/D12	触摸	TCS	6/D12	触摸片选
SCL	22/SCL	I2C时钟	SCL	10/SCL	I2C时钟
SDA	21/SDA	I2C数据	SDA	9/SDA	I2C数据
INT	16/D11	INT	INT	7/D11	INT
BUSY-TE	17/D10	防撕裂引脚	BUSY	NC	防撕裂引脚
X1	NC	自定义引脚1	X1	NC	自定义引脚1
X2	NC	自定义引脚2	X2	NC	自定义引脚2

从上图可以看到，不同开发板上的GDI接口，每个FPC线引脚的顺序的功能定义，都是相同的，但对应到具体开发板上的引脚，就根据实际开发板确定了。

2. GDI显示屏了解

DFRobot也提供多款支持GDI接口的显示屏：

显示屏	驱动芯片	说明
1.54" 240x240 IPS广视角TFT显示屏	ST7789	彩色TFT显示屏
1.8" 128x160 IPS TFT LCD 显示屏	ST7735S	彩色TFT显示屏
2.0" 320x240 IPS广视角TFT显示屏	ST7789	彩色TFT显示屏
2.8" 320x240 IPS TFT电阻触摸显示屏	ILI9341	彩色TFT带触摸显示屏
3.5" 480x320 IPS TFT电容触摸显示屏	ILI9488	彩色TFT带触摸显示屏
1.4" 172x320 IPS TFT LCD高清显示屏	ST7789V3	高清彩色TFT显示屏
1.51" 128x64 OLED 透明屏幕	SSD1309	透明单色OLED显示屏，蓝色

需要注意的是，最后一款 1.51" 128x64 OLED 透明屏幕，不是彩色屏幕，是单色屏幕的，在

本文的后面会单独说明用法。

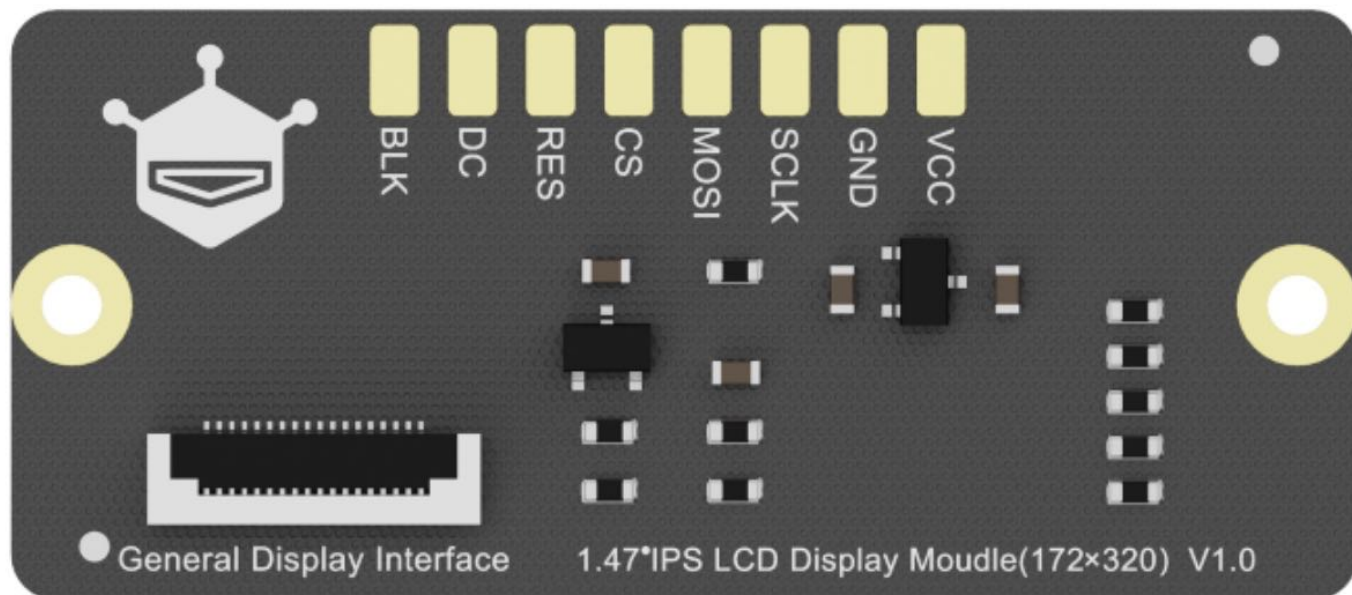
我平时最喜欢用的支持GDI接口的显示屏是：[1.4" 172×320 IPS TFT LCD高清显示屏](#)



这块屏的官方产品简介，那是相当的霸气：

这是一款1.47英寸，分辨率为172×320的彩色高清IPS显示屏。采用驱动芯片ST7789V3、SPI通讯接口，显示屏显示区域占比大，边框宽度仅 1mm，圆弧边角，外形小巧美观，显示效果精细、清晰，还原真实色彩。显示屏上可高清显示各种文字、图像、动画、甚至是视频，基于 Arduino 的图像显示 GDL 库和 LVGL 库，可以做出酷炫的动态效果，适合DIY电子项目。可广泛应用于迷你游戏机、迷你气象站、背包挂件、迷你时钟、迷你视频播放、礼物制作、小型仪表显示等场景。

在这款显示屏的背后，提供了GDI接口，还提供焊点用于焊接2.54的排针：



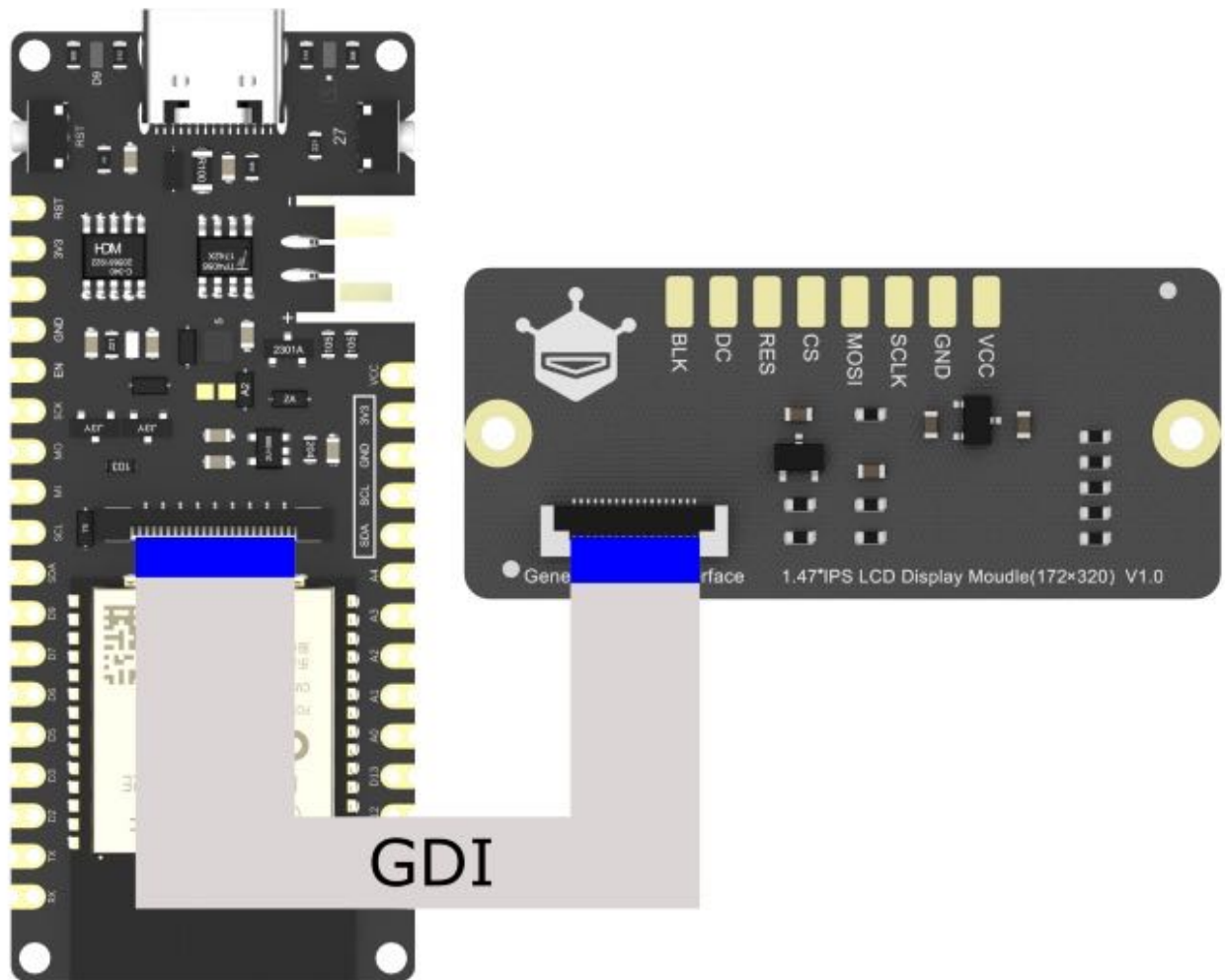
后续的TFT LCD的点屏操作和例子，都将以这块屏为基础展开。当然，如果使用其他的屏，文章中也会有说明如何处理。

3. 硬件连接

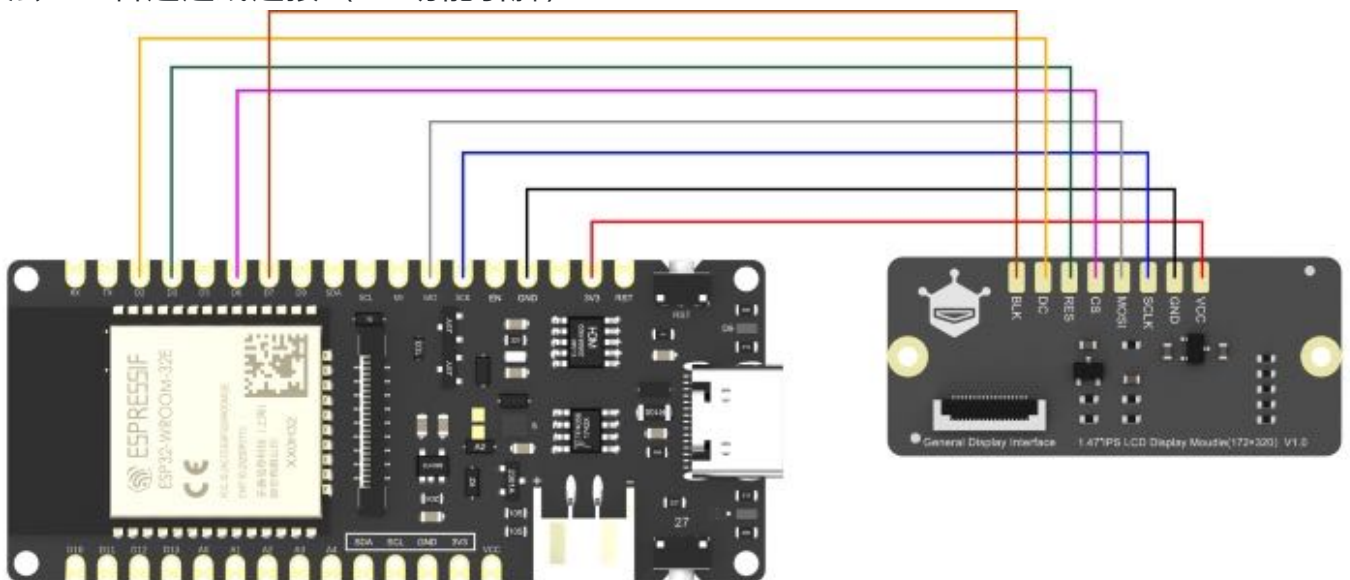
要使用支持GDI接口的显示屏，有两种方法，一种是用FPC线连接GDI接口，另一种则是连接SPI功能引脚。

具体连接如下图所示：

方法一：FPC线连接



方法二：普通连线连接（SPI功能引脚）



虽然这里的示例说的是1.4" 172×320 IPS TFT LCD高清显示屏，但其他支持GDI接口的显示屏，连接方式都是一样的，既可以用FPC线连接，也可以用普通连线连接。所以，支持GDI接口的显示屏，不仅可以用在提供GDI接口的开发板上，也可以用在其他没用GDI接口的开发板上。除非是这个显示屏，只提供了GDI接口。

FireBeetle 2 ESP32-C5开发板通过GDI接口连接[1.4" 172×320 IPS TFT LCD高清显示屏](#)，连接后具体如下：



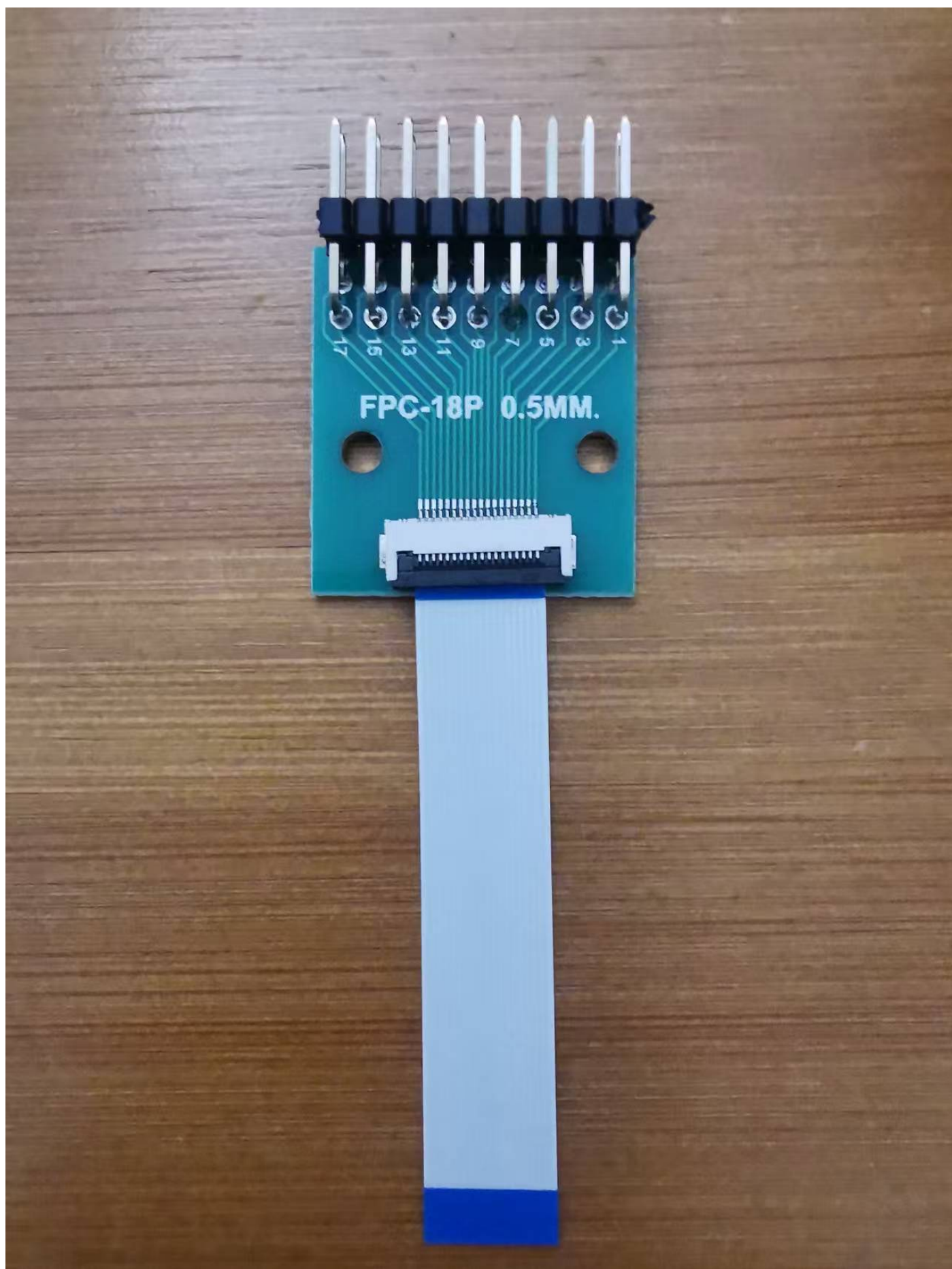
另外，需要说明的是，GDI接口并不是专用接口，只是定义了1个18pin-FPC线的线序，这里

面的引脚，不只是用于连接GDI屏幕，还可以当作通用IO口使用(根据各IO口自身功能定义确定)。

例如FireBeetle 2 ESP32-C5开发板的GDI接口定义如下：

FPC PINS	Firebeetle 2 ESP32-C5 PINS	Description
VCC	3V3	3V3
LCD_BL	15/D13	背光
GND	GND	GND
SCLK	23/SCLK	SPI时钟
MOSI	24/MOSI	主机输出，从机输入
MISO	25/MISO	主机输入，从机输出
LCD_DC	8/D2	数据/命令
LCD_RST	26/D3	复位
LCD_CS	27/D6	TDT片选
SD_CS	3/A2	SD卡片选
FCS	NC	字库片选
TCS	6/D12	触摸片选
SCL	10/SCL	I2C时钟
SDA	9/SDA	I2C数据
INT	7/D11	INT
BUSY	NC	防撕裂引脚
X1	NC	自定义引脚1
X2	NC	自定义引脚2

当不连接GDI显示屏的时候，可以用一个下面的转接板，来连接到开发板的GDI接口：



连接后，各引脚本来是什么功能的，转接后，就还是什么功能。例如其中LCD_BL对应的D13接口，你就可以连接一个LED，然后在程序中控制LED的亮灭。你还可以把支持GDI接口的显示屏，仅通过普通连线连接接到这个转接板上使用。

你甚至可以将非GDI接口的显示屏接在转接板上，变相“支持”GDI接口 :) 除了使用SPI通信的屏幕，你也可以连接I2C接口的屏幕，如128x64的OLED，毕竟GDI接口上有I2C引脚的定义，何况还可以自定义I2C引脚的。

4. 引脚配置

GDI接口是18Pin的，但点亮屏幕，我们通常只需要关注下面的引脚即可。在FireBeetle 2 ESP32-C5开发板上使用GDI接口显示屏，具体如下：

功能	引脚	说明
SCLK	23	SPI时钟
MOSI	24	SPI主机输出，从机输入
LCD_BL	15	显示屏背光
LCD_DC	8	LCD数据/命令
LCD_RST	26	LCD复位
LCD_CS	27	TFT片选

如果你使用其他开发板，则根据具体连接的方式来确定对应的引脚即可。

在Arduino中，我们要将上面的引脚定义转换为代码中的宏定义：

```
#define TFT_SCLK      23      // SPI时钟
#define TFT_MOSI      24      // SPI主机输出，从机输入
#define TFT_BL        15      // 显示屏背光
#define TFT_DC        8       // LCD数据/命令
#define TFT_RST       26      // LCD复位
#define TFT_CS        27      // TFT片选
```

在后续的不同点屏演示中，我们将统一使用上面的宏定义。

二、Arduino-ESP32安装

因为是要在Arduino开发环境中进行点屏，所以先需要在Arduino中安装ESP32的支持库Arduino-ESP32。

1. ECO1版本说明

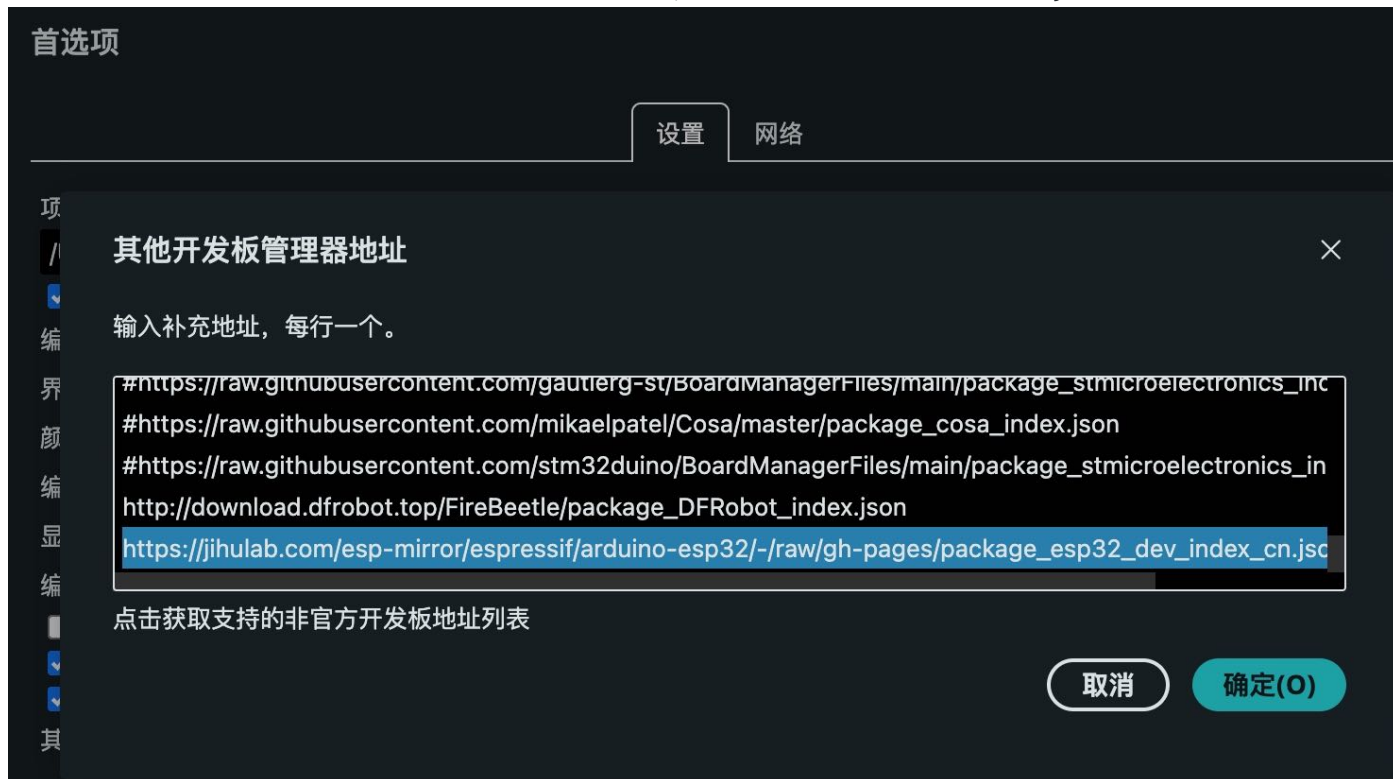
由于DFRobot提供的FireBeetle 2 ESP32-C5开发板的C5模组是ESP32-C5 revision v0.1，即

ECO1版本，必须使用Arduino-ESP32的3.3.0-alpha1版本才能提供支持，即使提示升级也不要升级，否则无法继续支持了。

如果是之后的C5模组，如v1.0、v1.1、v1.2等版本的，或者称呼为ECO2的，就可以直接安装Arduino-ESP32的最新版本，或者提示升级的时候升级到最新版本即可。

2. json配置文件

在Arduino IDE中安装Arduino-ESP32支持库，需要先添加一条对应的json配置文件地址：



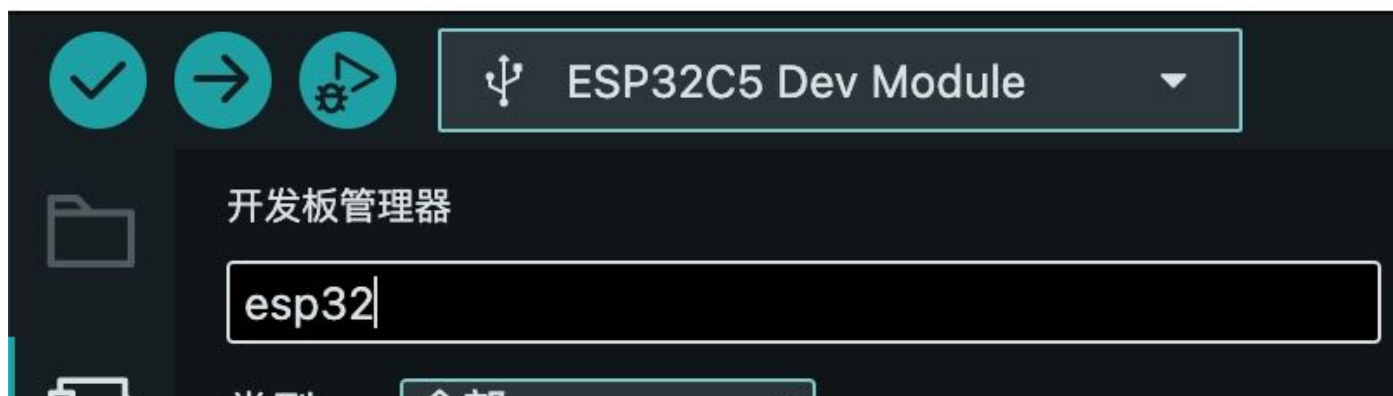
因为网络原因，安装Arduino-ESP32有时会非常费力，好在乐鑫已经贴心的在 [Arduino ESP32 / Getting Started / Installing](#) 提供了中国区专用地址：





https://jihulab.com/esp-mirror/espressif/arduino-esp32/-/raw/gh-pages/package_esp32_dev_index_cn.json

将上面的json文件网址，拷贝到Arduino中的开发板管理器地址设置中即可。

3. 安装支持库

在开发板管理中，搜索esp32，找到名称只有"esp32"的进行操作：





类型: 全部

1

Arduino ESP32 Boards by Arduino

Boards included in this package: Arduino Nano ESP32

[更多信息](#)

2.0.18-4 ▼

安装

DFRobot ESP32 Boards by DFRobot

Boards included in this package: FireBettle ESP32-E Borad, FireBettle ESP32 Borad

[更多信息](#)

0.2.1 ▼

安装

esp32 by Espressif Systems

3.3.0-alpha1-cn 已安装

Boards included in this package: UM FeatherS2, microS2, u-blox NORA-W10 series (ESP32-S3), Adafruit Feather ESP32-S2 TFT, OLIMEX ESP32-GATEWAY, OLIMEX ESP32-...

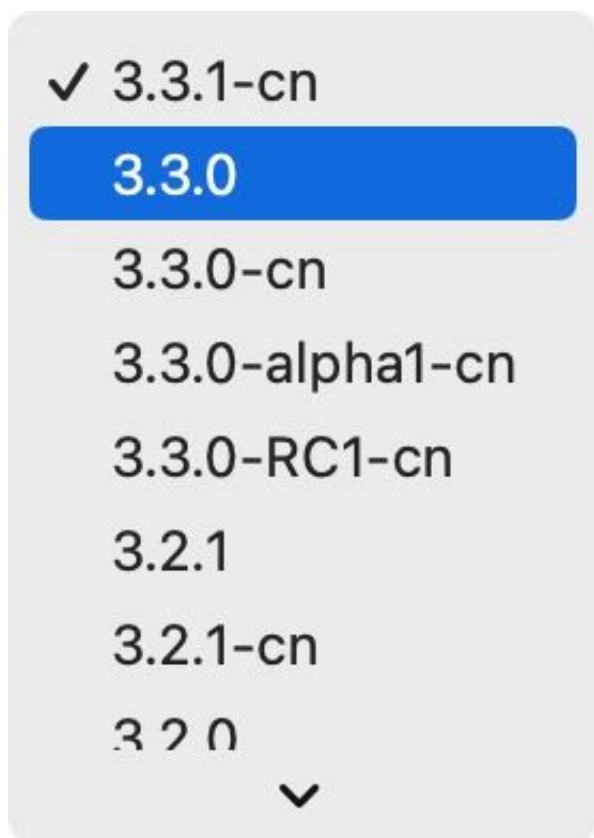
[更多信息](#)

3.3.1-cr ▼

更新

2

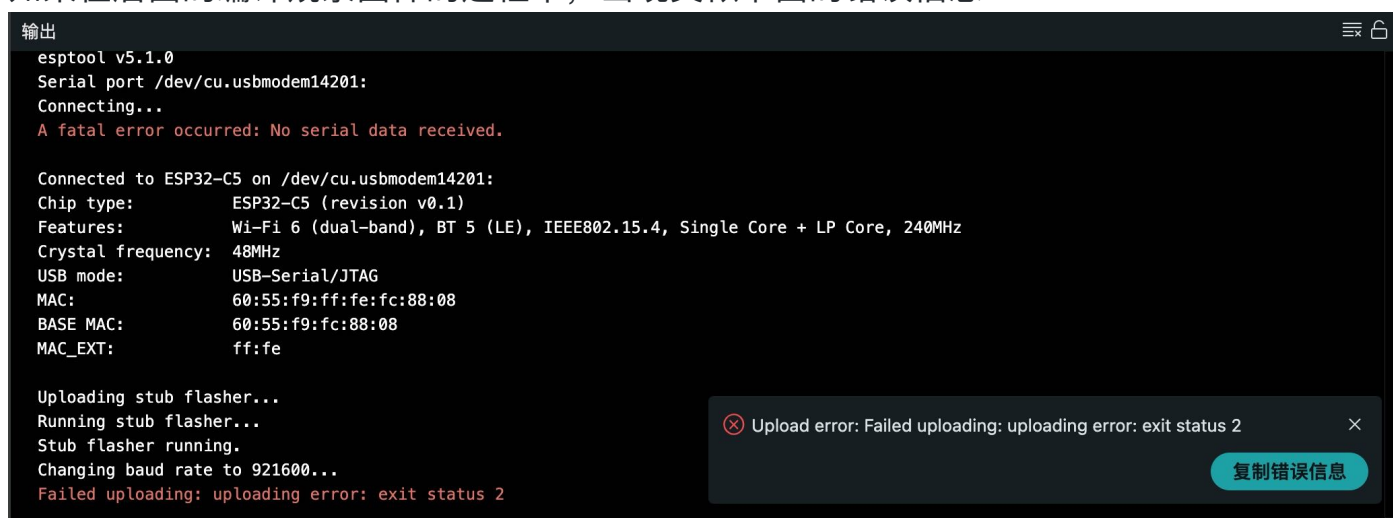
点击esp32下的版本列表，会详细列出当前可用的版本：



在中国区安装的话，一定要选择带有-cn后缀的版本，下载速度快又好！！

记住，如果是 **ESP32-C5 revision v0.1 / ECO1**，那么就必须选择 **3.3.0-alpha1-cn**，否则直接安装最新的版本即可。

如果在后面的编译烧录固件的过程中，出现类似下面的错误信息：

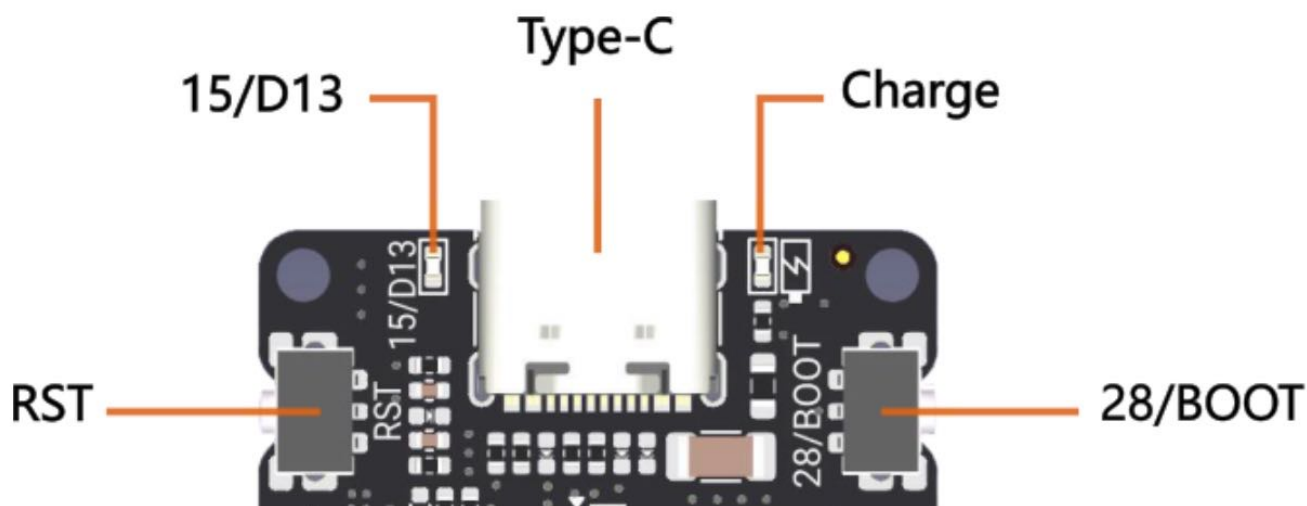


这一般就是版本不正确导致的。

4. 编写blink测试

安装好开发板支持库以后，编写一个最简单的Blink(闪灯)程序。

首先从FireBeetle 2 ESP32-C5开发板的引脚定义，可以得知板载LED的引脚编号为15：



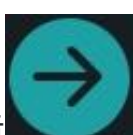
然后，编写对应的Blink程序：

```
int led = 15;
bool status = false;
void setup() {
  pinMode(led,OUTPUT);
}

void loop() {
  digitalWrite(led,status?HIGH:LOW);
  status = !status;
  delay(100);
}
```



编写完成后，点击



烧录到开发板，输出界面出现如下的结果，表示烧录成功：

输出

```
Writing at 0x0002ab2a... (40 %)
Writing at 0x00031eb7... (50 %)
Writing at 0x00038ef4... (60 %)
Writing at 0x0003f583... (70 %)
Writing at 0x00045a15... (80 %)
Writing at 0x0004bc2a... (90 %)
Writing at 0x0005242a... (100 %)
Wrote 272320 bytes (148086 compressed) at 0x00010000 in 1.2 seconds (effective 1870.0 kbit/s)...
Hash of data verified.

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

如果烧录失败，请查看上一小节检查是否出现同样的错误信息，并进行处理。

烧录成功后，FireBeetle 2 ESP32-C5开发板上的绿色LED灯就会闪烁（无声的“吧嗒吧

嗒”）。

至此，万事俱备，只待“点屏”。

三、DFRobot_GDL显示库

DFRobot官方提供了一个DFRobot_GDL显示库，可以很方便点屏，显示中英文、图片、动画、折线等。

1. 安装DFRobot_GDL库

在Arduino的库管理界面中，搜索DFRobot_GDL，会搜索出来一个DFRobot_GDL库：



这个库并非DFRobot官方提交，经实测不支持ESP32-C5，因此请勿安装。
如已安装，请立即移除。实际上，若不移除，后续安装官方DFRobot_GDL库时也会自动覆盖。

要安装DFRobot官方最新版本DFRobot_GDL，正确的方法是先访问

https://gitee.com/dfrobot/DFRobot_GDL，然后点击 克隆/下载：

gitee

开源企业版高校版私有云模力方舟我的

搜索开源

DFRobot/DFRobot_GDL

<> 代码

Issues 0

Pull Requests 0

Wiki

统计

流水线

服务

master

分支 5

标签 3

克隆/下载

lbx-8023 update 744b08d 1个月前

210 次提交

examples

update

1个月前

image

V1.0.3

3个月前

src

Update DFRobot_Touch.cpp

1个月前

.DS_Store

preparing for Ard Lib Registry

2年前

.gitignore

preparing for Ard Lib Registry

2年前

LICENCE

V1.0.3

3个月前

README.md

V1.0.3

3个月前

keywords.txt

change keywords

3年前

library.json

added the last edge case chinese fonts for the sp project

1年前

library.properties

V1.0.3

3个月前

然后在下载界面，点击 下载ZIP：

gitee

开源企业版高校版私有云模力方舟我的

搜索开源

DFRobot/DFRobot_GDL

<> 代码

Issues 0

Pull Requests 0

master

分支 5

标签 3

克隆/下载

lbx-8023 update 744b08d 1个月前

210 次提交

examples

update

1个月前

image

V1.0.3

3个月前

src

Update DFRobot_Touch.cpp

1个月前

.DS_Store

preparing for Ard Lib Registry

2年前

.gitignore

preparing for Ard Lib Registry

2年前

LICENCE

V1.0.3

3个月前

README.md

V1.0.3

3个月前

keywords.txt

change keywords

3年前

library.json

added the last edge case chinese fonts for the sp project

1年前

library.properties

V1.0.3

3个月前

克隆/下载

HTTPS

SSH

SVN

SVN+SSH

git@gitee.com:dfrobot/DFRobot_GDL.git

提示

下载代码请复制以下命令到终端执行

git clone git@gitee.com:dfrobot/DFRobot_GDL.git

为确保你提交的代码身份被 Gitee 正确识别，请执行以下命令完成配置

git config --global user.name 'HonestQiao'

git config --global user.email 'honestqiao@163.com'

初次使用 SSH 协议进行代码克隆、推送等操作时，需按下述提示完成 SSH 配置

1 生成 RSA 密钥

ssh-keygen -t rsa

2 获取 RSA 公钥内容，并配置到 SSH公钥 中

cat ~/.ssh/id_rsa.pub

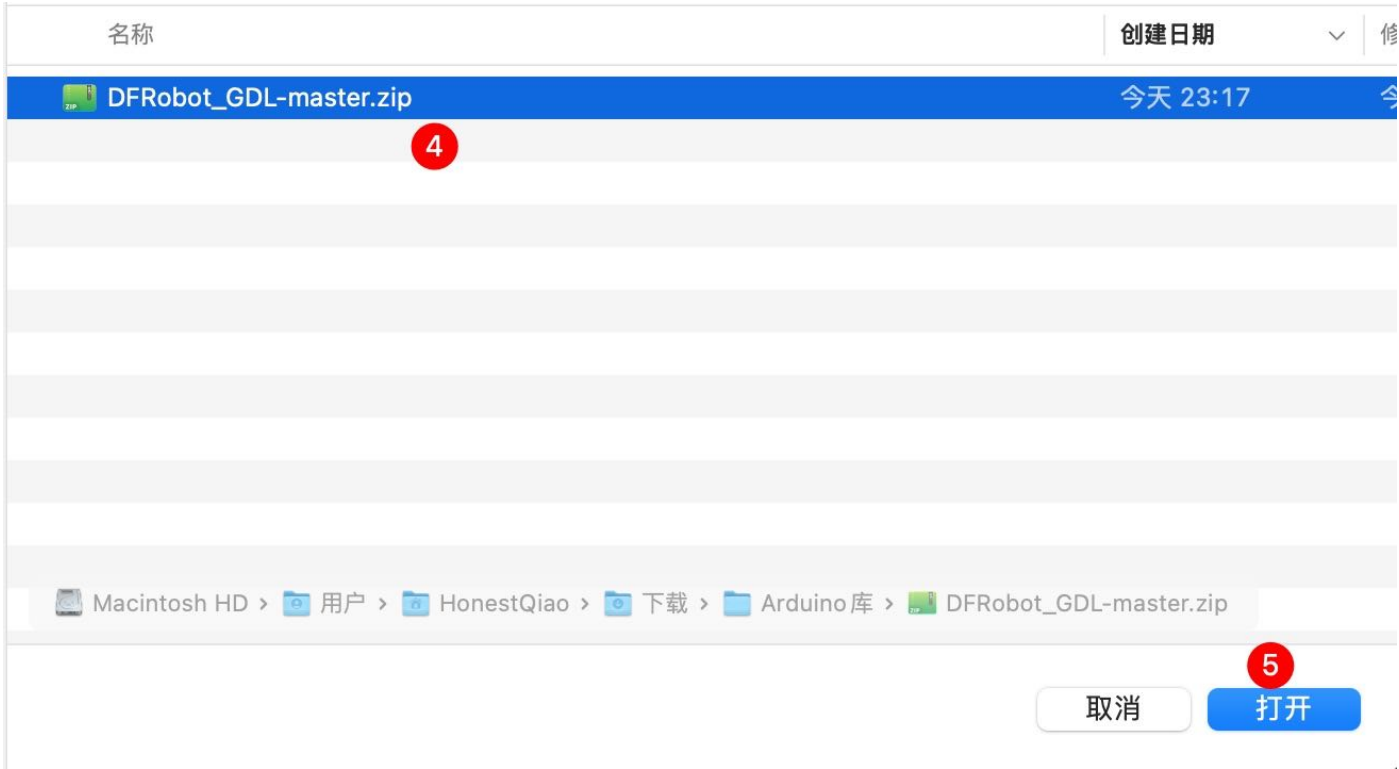
浏览器会将DFRobot_GDL库下载回来：



然后，在Arduino IDE中，按照下面的方式，添加**.ZIP库**：



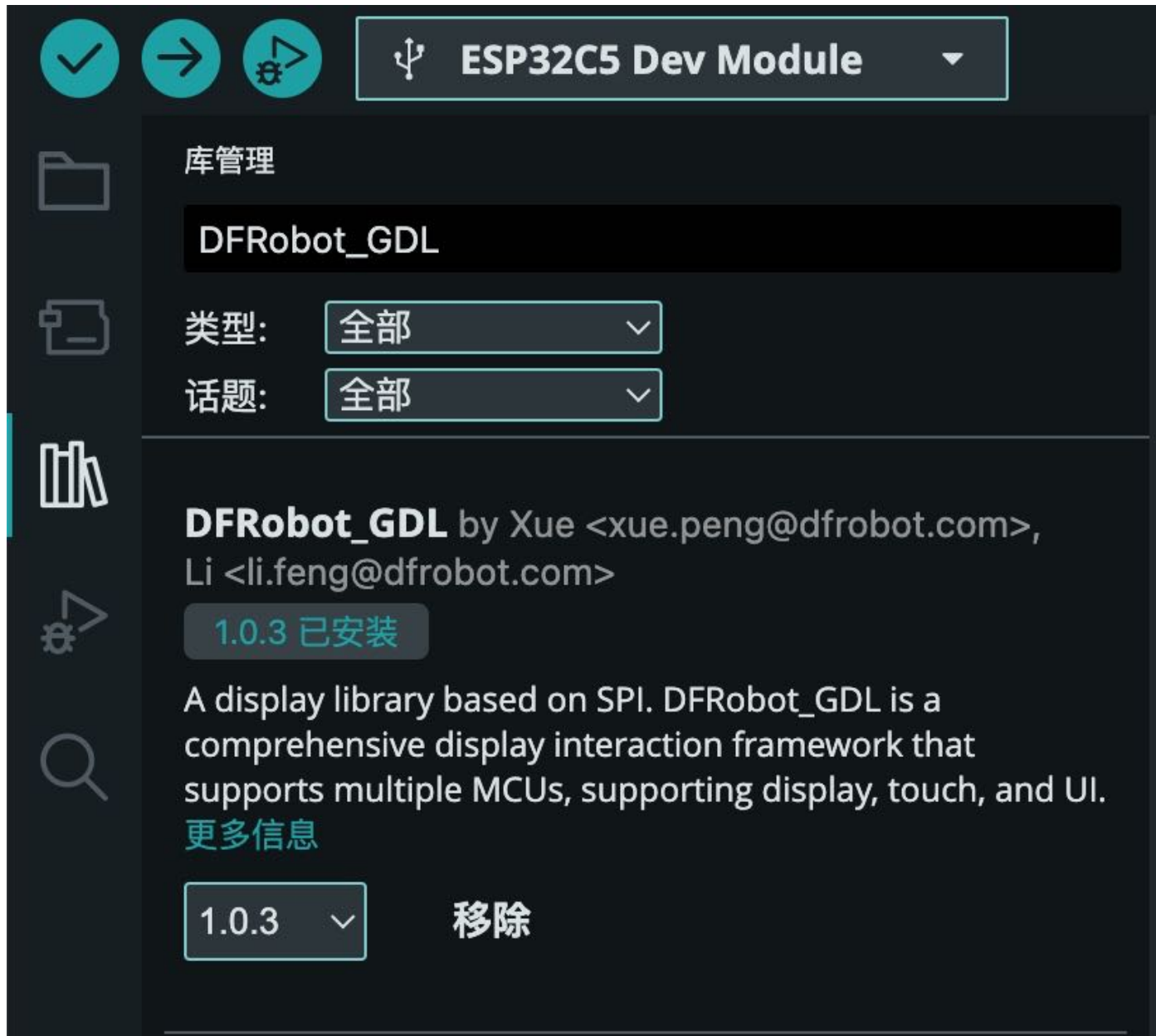
然后在打开的文件选择界面，选择刚才下载的 DFRobot_GDL-master.zip 即可：



点击打开后，等待Arduino IDE安装库文件：

正在处理中 DFRobot_GDL-master.zip

安装完成后，搜索到的DFRobot_GDL，也会显示 1.0.3已安装：

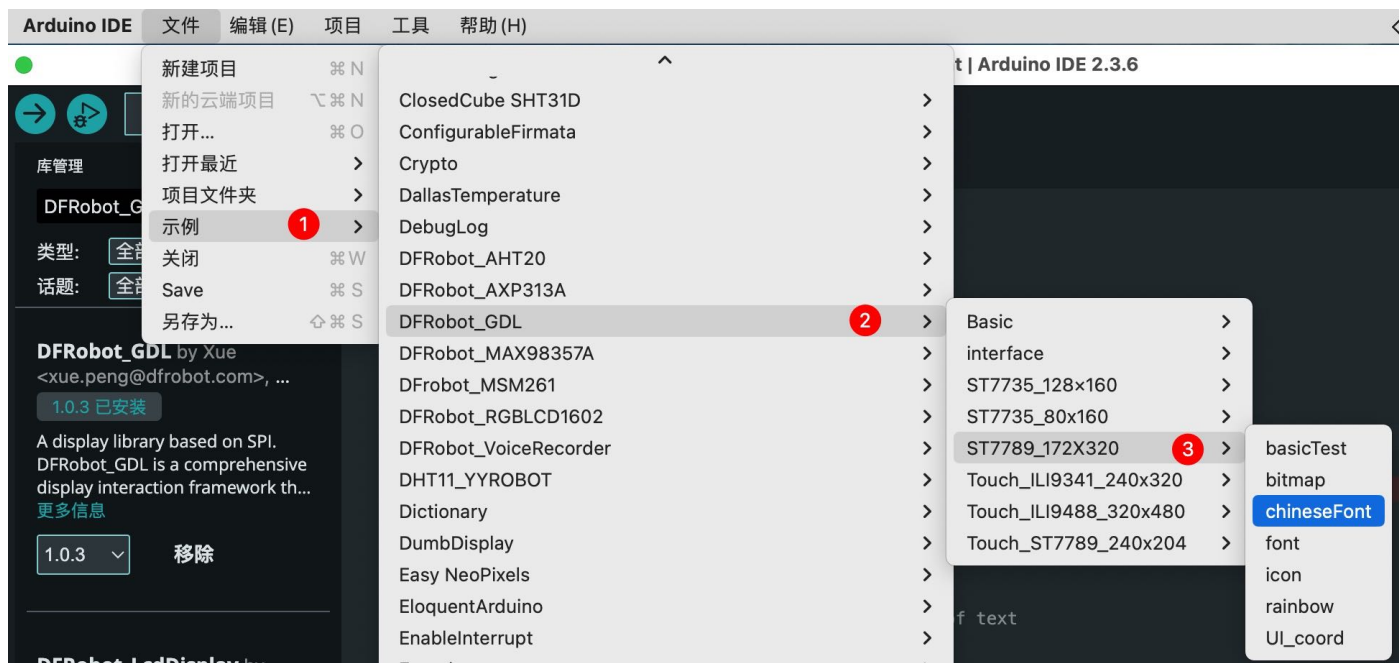


这次就不用点移除，可以正常使用了。

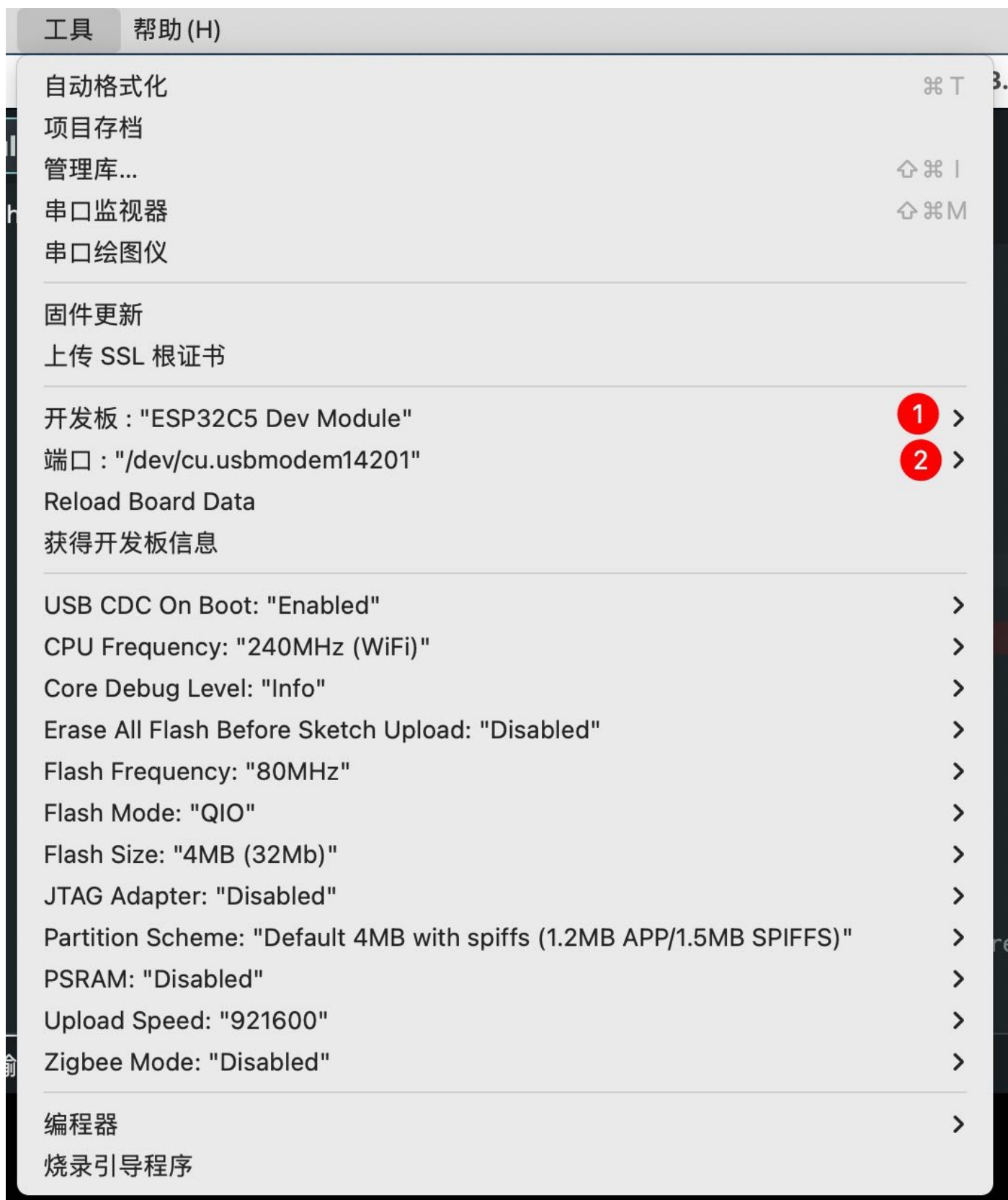
2. 演示代码

DFRobot_GDL提供了很多例子可供参考，帮助我们快速了解具体用法。

通过下面的途径，我们选择一个[1.4" 172x320 IPS TFT LCD高清显示屏](#)对应的显示中文的例子：



打开例子程序代码后，通过工具菜单，设置开发板的参数：



在上述界面中，仅需正确选择开发板为**ESP32C5 Dev Module**，端口为**开发板连接后对应的串口设备端口**即可。

一般我还会设置下面两个选项：

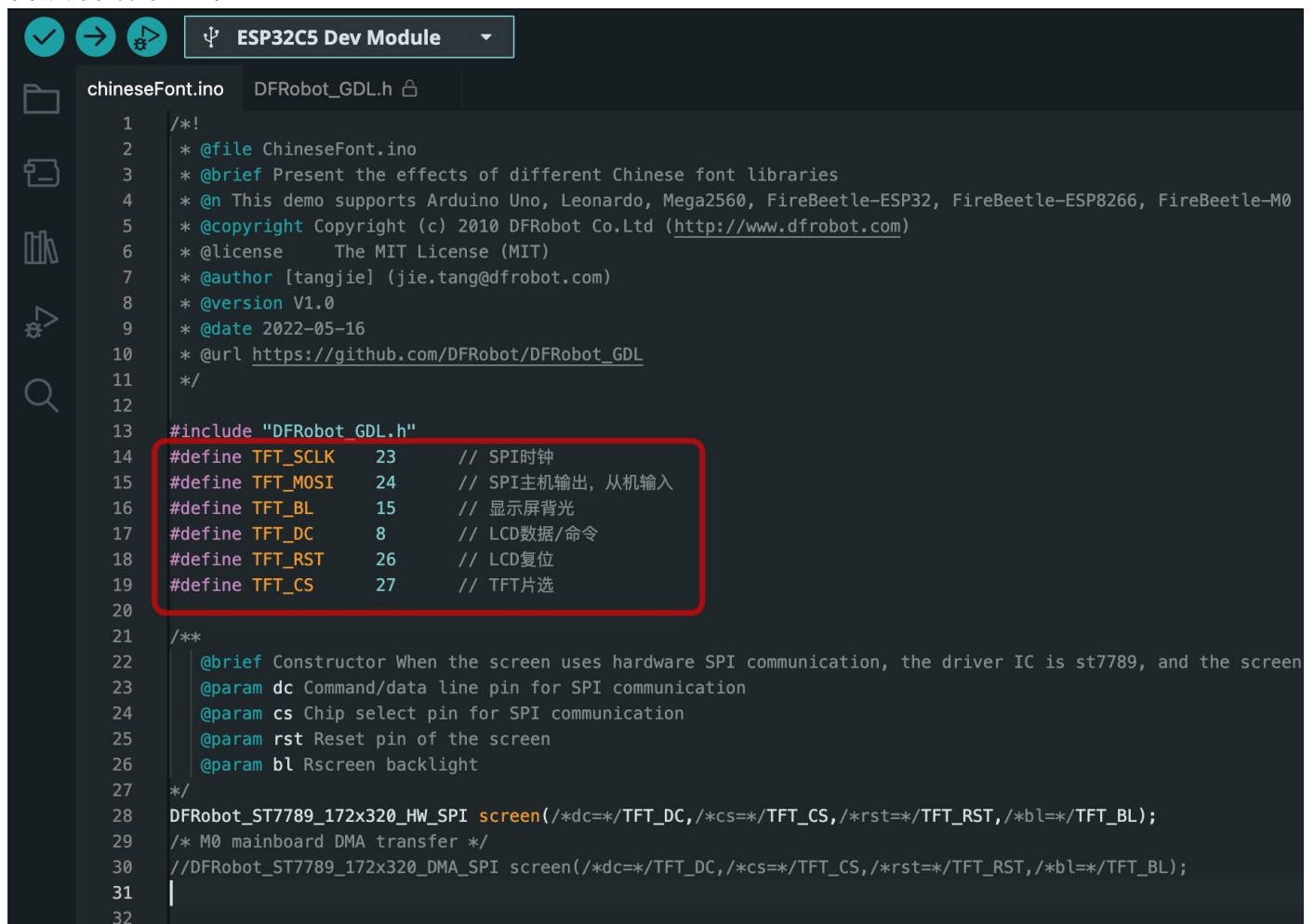
- **USB CDC On Boot: "Enable"**：日志信息通过USB连接的串口输出
- **Core Debug Level: "Info"**：显示开发板运行过程中输出的信息和运行出错的信息

3. 引脚设置

在打开的例子代码中，将原有的引脚定义的部分，替换为之前的引脚配置中的宏定义：

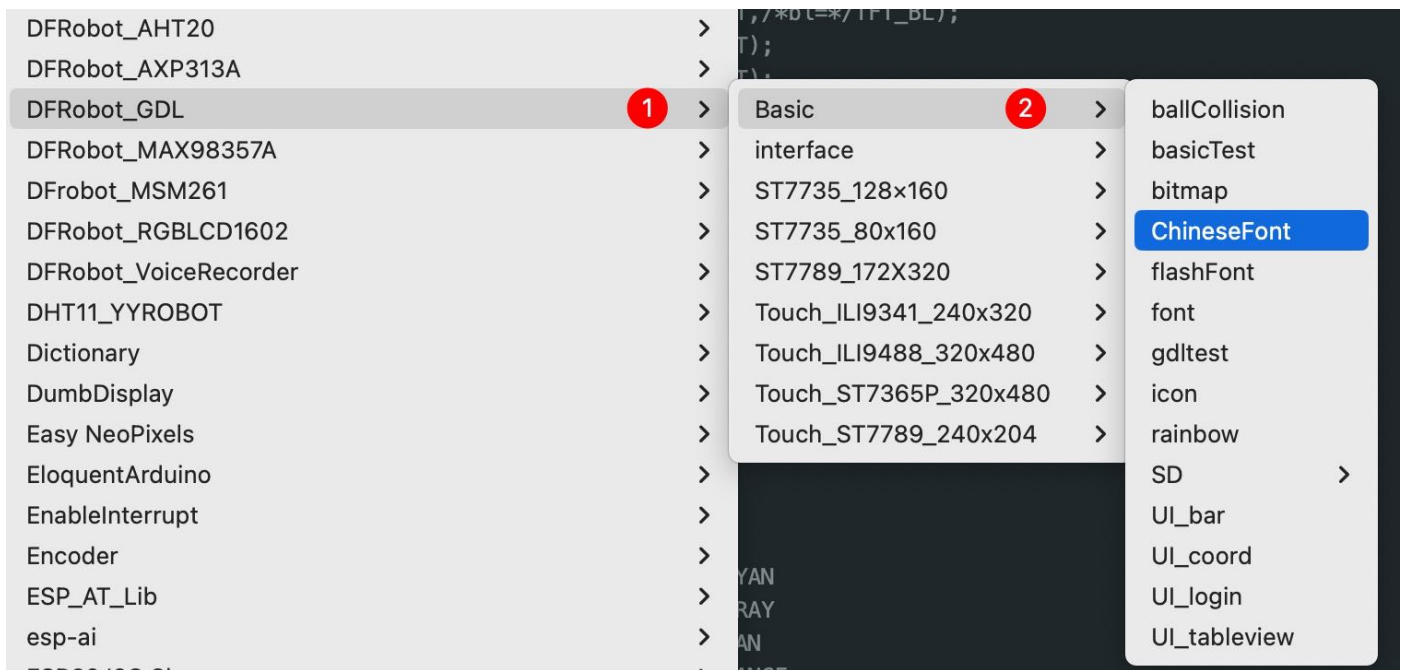
```
#define TFT_SCLK      23      // SPI时钟
#define TFT_MOSI      24      // SPI主机输出，从机输入
#define TFT_BL        15      // 显示屏背光
#define TFT_DC         8      // LCD数据/命令
#define TFT_RST       26      // LCD复位
#define TFT_CS        27      // TFT片选
```

替换后结果如下：



```
chineseFont.ino  DFRobot_GDL.h
1  /*!
2  * @file ChineseFont.ino
3  * @brief Present the effects of different Chinese font libraries
4  * @n This demo supports Arduino Uno, Leonardo, Mega2560, FireBeetle-ESP32, FireBeetle-ESP8266, FireBeetle-M0
5  * @copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
6  * @license The MIT License (MIT)
7  * @author [tangjie] (jie.tang@dfrobot.com)
8  * @version V1.0
9  * @date 2022-05-16
10 * @url https://github.com/DFRobot/DFRobot\_GDL
11 */
12
13 #include "DFRobot_GDL.h"
14 #define TFT_SCLK 23 // SPI时钟
15 #define TFT_MOSI 24 // SPI主机输出，从机输入
16 #define TFT_BL 15 // 显示屏背光
17 #define TFT_DC 8 // LCD数据/命令
18 #define TFT_RST 26 // LCD复位
19 #define TFT_CS 27 // TFT片选
20
21 /**
22  * @brief Constructor When the screen uses hardware SPI communication, the driver IC is st7789, and the screen
23  * @param dc Command/data line pin for SPI communication
24  * @param cs Chip select pin for SPI communication
25  * @param rst Reset pin of the screen
26  * @param bl Rscreen backlight
27  */
28 DFRobot_ST7789_172x320_HW_SPI screen(/*dc=*/TFT_DC, /*cs=*/TFT_CS, /*rst=*/TFT_RST, /*bl=*/TFT_BL);
29 /* M0 mainboard DMA transfer */
30 //DFRobot_ST7789_172x320_DMA_SPI screen(/*dc=*/TFT_DC, /*cs=*/TFT_CS, /*rst=*/TFT_RST, /*bl=*/TFT_BL);
31
32
```

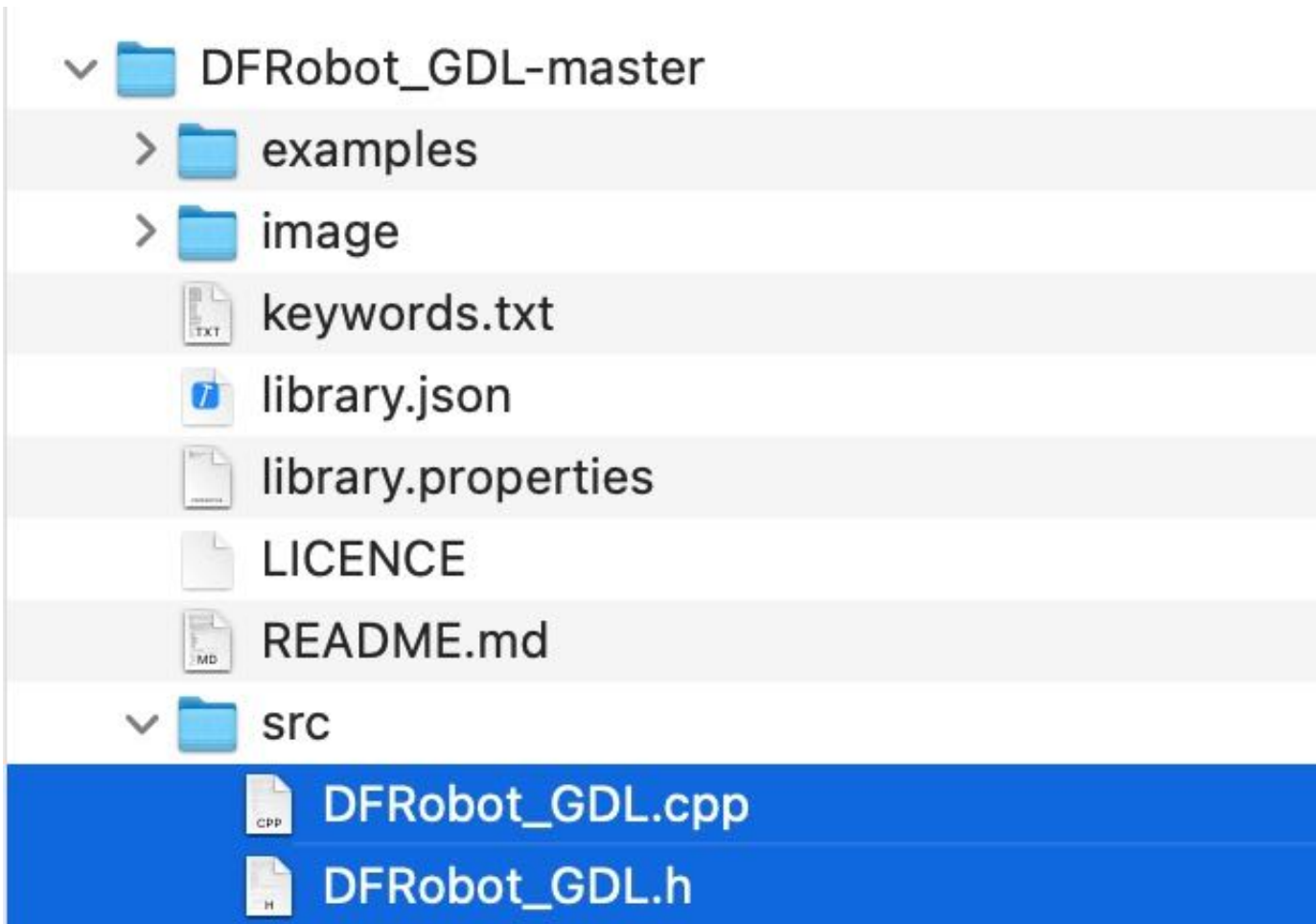
如果你的屏幕，在上面列出的例子中没有对应的，那么可以点击下面的例子：



在打开的例子代码中，列出了可以直接支持的显示屏：

```
/**
 * @brief Constructor Constructor of hardware SPI communication
 * @param dc Command/data line pin for SPI communication
 * @param cs Chip select pin for SPI communication
 * @param rst reset pin of the screen
 */
//DFRobot_ST7789_240x204_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST,/*bl=*/TFT_BL);
//DFRobot_ST7789_240x240_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ST7789_240x320_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ILI9341_240x320_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ILI9488_320x480_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ST7365P_320x480_HW_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST,/*bl=*/TFT_BL);
/* M0 mainboard DMA transfer */
//DFRobot_ST7789_240x204_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST,/*bl=*/TFT_BL);
//DFRobot_ST7789_240x240_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ST7789_240x320_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ILI9341_240x320_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ILI9488_320x480_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST);
//DFRobot_ST7365P_320x480_DMA_SPI screen(/*dc=*/TFT_DC,/*cs=*/TFT_CS,/*rst=*/TFT_RST,/*bl=*/TFT_BL);
```

如果还没有的话，你可参考列表中某一款屏幕的定义，自行修改DFRobot_GDL.h和DFRobot_GDL.cpp中的配置：



具体怎么定义，大家可以打开这两个文件研究，并不复杂，这里就不展开了。

4. 编译烧录



修改完成后，点击  编译，然后点击  烧录到开发板，输出界面出现如下的结果，表示烧录成功：

```
输出
Writing at 0x0003e039... (63 %)
Writing at 0x000444d9... (72 %)
Writing at 0x0004ab6d... (81 %)
Writing at 0x0005136b... (90 %)
Writing at 0x0005779c... (100 %)
Wrote 320224 bytes (178652 compressed) at 0x00010000 in 1.5 seconds (effective 1714.8 kbit/s)...
Hash of data verified.

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

如果出现下面的错误：


```
输出
Multiple Libraries were found for "SD.h"
  Used: /Users/HonestQiao/Library/Arduino15/packages/esp32/hardware/esp32/3.3.0-alpha1-cn/libraries/SD
  Not used: /Users/HonestQiao/Library/Arduino15/libraries/SD
  Not used: /Users/HonestQiao/Documents/Arduino/libraries/SD
Using library DFRobot_GDL at version 1.0.3 in folder: /Users/HonestQiao/Documents/Arduino/libraries/DFRobot_GDL
Using library SPI at version 3.3.0 in folder: /Users/HonestQiao/Library/Arduino15/packages/esp32/hardware/esp32/3.3.0-alpha1-cn/libraries/SPI
Using library Wire at version 3.3.0 in folder: /Users/HonestQiao/Library/Arduino15/packages/esp32/hardware/esp32/3.3.0-alpha1-cn/libraries/Wire
Using library SD at version 3.3.0 in folder: /Users/HonestQiao/Library/Arduino15/packages/esp32/hardware/esp32/3.3.0-alpha1-cn/libraries/SD
Using library FS at version 3.3.0 in folder: /Users/HonestQiao/Library/Arduino15/packages/esp32/hardware/esp32/3.3.0-alpha1-cn/libraries/FS
exit status 1
Compilation error: 'SIMKAIFont12pt' was not declared in this scope
复制错误信息
```

这说明需要往前看，先移除DFRobot_GDL，再安装DFRobot_GDL-master.zip

如果烧录后，屏幕没有正确显示，可以把开发板从电脑断开，然后重新连线上电。
如果还是没有正确显示的话，那么需要添加一行到如下图所示位置：

```
42 void setup() {
43     // put your setup code here, to run once:
44     Serial.begin(115200);
45     SPI.begin(TFT_SCLK, -1, TFT_MOSI, -1); // 添加的行，用于启动SPI通信
46     screen.begin();
47 }
```

修改后的代码如下：

```
void setup() {
    // put your setup code here, to run once:
    Serial.begin(115200);
    SPI.begin(TFT_SCLK, -1, TFT_MOSI, -1); // 添加的行，用于启动SPI通信
    screen.begin();
}
```

修改完成后，再次编译烧录，就能正常显示。

5. 运行效果

显示中文的实际效果如下：【动图】



我们再选择一个例子Rainbow:

DFRobot_AHT20	>	
DFRobot_AXP313A	>	
DFRobot_GDL	>	Basic >
DFRobot_MAX98357A	>	interface >
DFrobot_MSM261	>	ST7735_128x160 >
DFRobot_RGBLCD1602	>	ST7735_80x160 >
DFRobot_VoiceRecorder	>	ST7789_172X320 > basicTest
DHT11_YYROBOT	>	Touch_ILI9341_240x320 > bitmap
Dictionary	>	Touch_ILI9488_320x480 > chineseFont
DumbDisplay	>	Touch_ST7365P_320x480 > font
Easy NeoPixels	>	Touch_ST7789_240x204 > icon
EloquentArduino	>	
EnableInterrupt	>	rainbow
Encoder	>	UI_coord

选择后，按照前面的步骤，修改引脚配置，添加SPI.begin(...), 编译烧录，运行效果如下：



四、Adafruit_ST7789显示库

Adafruit也为Arduino提供了很多库，方便我们点屏，对驱动芯片为ST7789和ST7735的屏都能很好的支持。

1. 安装Adafruit ST7789显示库

在本文“一、硬件了解”中关于GDI显示屏的部分，已经列出了 [1.4" 172x320 IPS TFT LCD高清显示屏](#) 驱动芯片为ST7789V3，安装**Adafruit ST7789**显示库就可以使用。

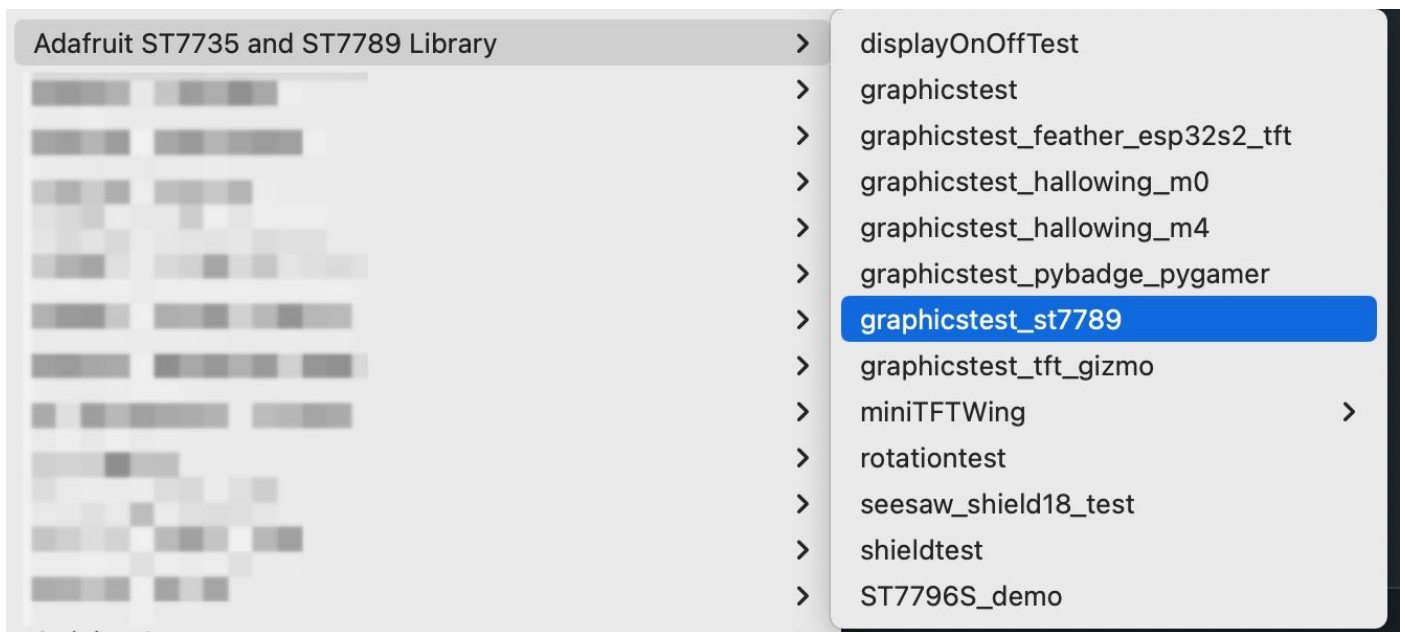
Adafruit还提供了**Adafruit ILI9341**显示库，可以根据所使用的显示屏的驱动芯片进行选择。

在Arduino IDE的库管理界面中，搜索**Adafruit ST7789**，安装 **Adafruit ST7735 and ST7789 Library** 最新版本即可：



2. 演示代码

Adafruit ST7735 and ST7789 Library显示库也提供了很多例子可供参考，帮助我们快速了解具体用法：



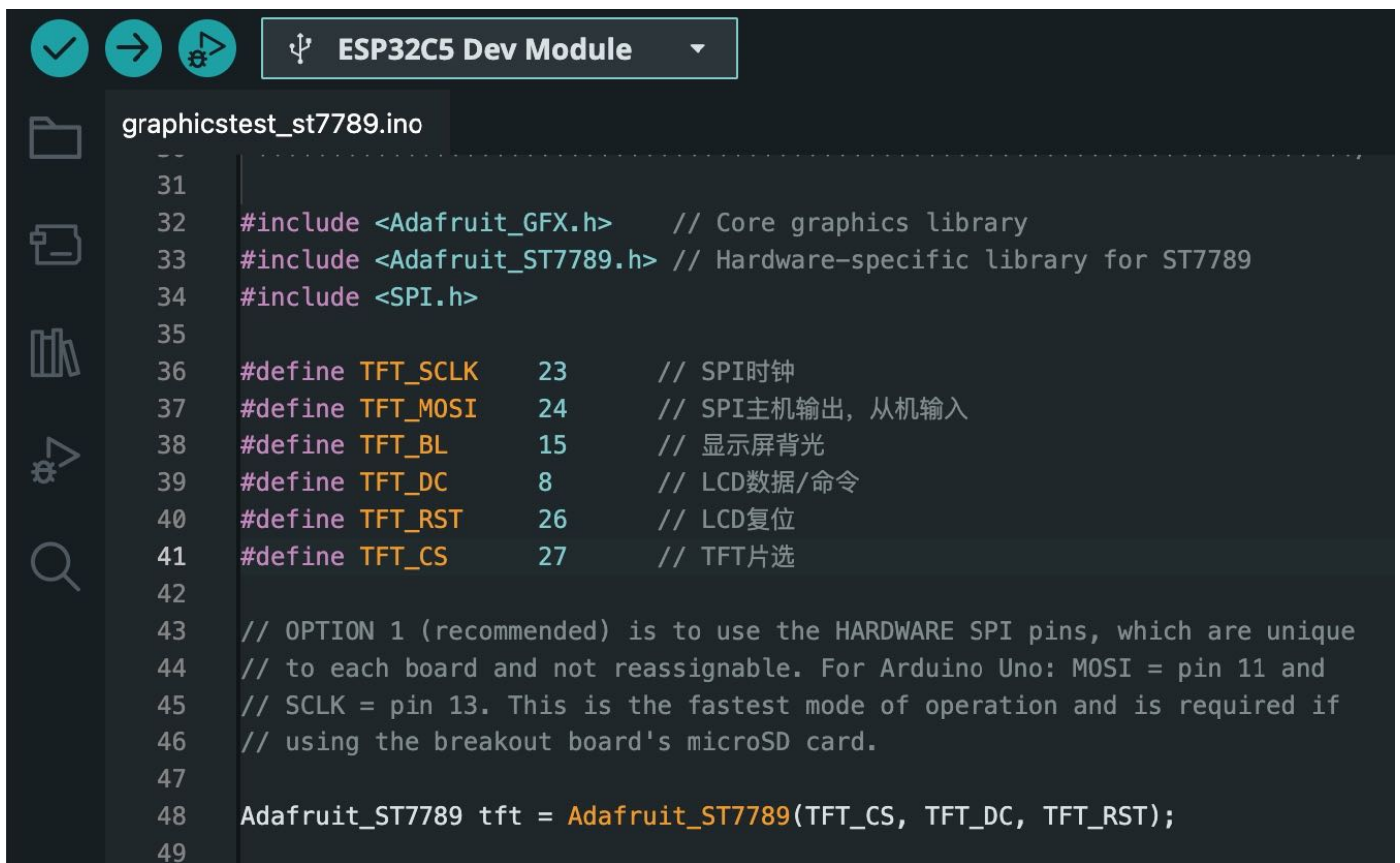
我们选择其中的graphicstest_st7789例子，其提供了多种显示效果展示。

3. 引脚设置

在打开的例子代码中，将原有的引脚定义的部分，替换为之前的引脚配置中的宏定义：

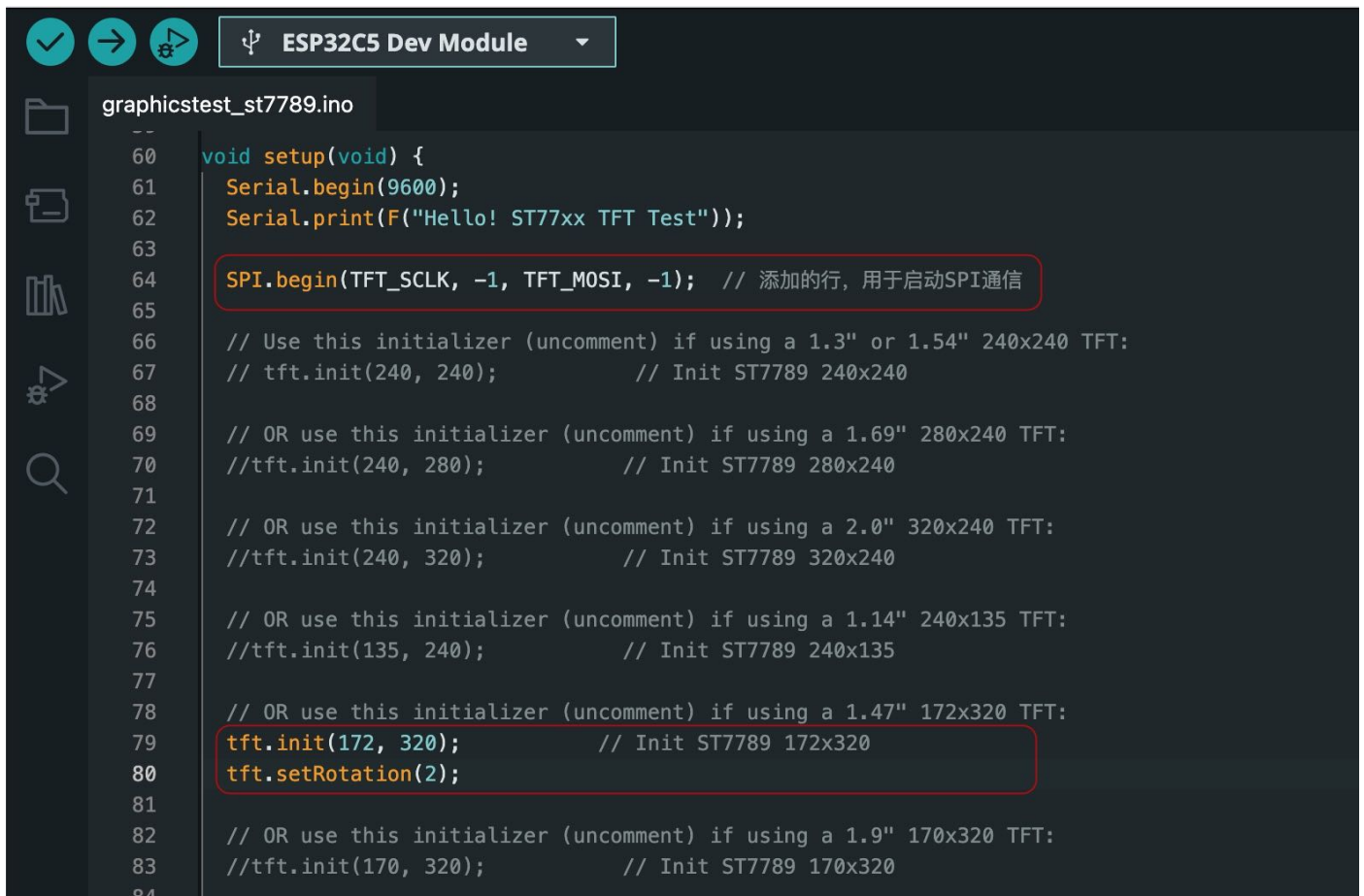
```
#define TFT_SCLK      23      // SPI时钟
#define TFT_MOSI      24      // SPI主机输出，从机输入
#define TFT_BL        15      // 显示屏背光
#define TFT_DC         8      // LCD数据/命令
#define TFT_RST       26      // LCD复位
#define TFT_CS        27      // TFT片选
```

替换后结果如下：



```
31
32 #include <Adafruit_GFX.h> // Core graphics library
33 #include <Adafruit_ST7789.h> // Hardware-specific library for ST7789
34 #include <SPI.h>
35
36 #define TFT_SCLK 23 // SPI时钟
37 #define TFT_MOSI 24 // SPI主机输出, 从机输入
38 #define TFT_BL 15 // 显示屏背光
39 #define TFT_DC 8 // LCD数据/命令
40 #define TFT_RST 26 // LCD复位
41 #define TFT_CS 27 // TFT片选
42
43 // OPTION 1 (recommended) is to use the HARDWARE SPI pins, which are unique
44 // to each board and not reassignable. For Arduino Uno: MOSI = pin 11 and
45 // SCLK = pin 13. This is the fastest mode of operation and is required if
46 // using the breakout board's microSD card.
47
48 Adafruit_ST7789 tft = Adafruit_ST7789(TFT_CS, TFT_DC, TFT_RST);
49
```

另外, 还需要进行下面的修改:



```
60 void setup(void) {
61   Serial.begin(9600);
62   Serial.print(F("Hello! ST77xx TFT Test"));
63
64   SPI.begin(TFT_SCLK, -1, TFT_MOSI, -1); // 添加的行, 用于启动SPI通信
65
66   // Use this initializer (uncomment) if using a 1.3" or 1.54" 240x240 TFT:
67   // tft.init(240, 240); // Init ST7789 240x240
68
69   // OR use this initializer (uncomment) if using a 1.69" 280x240 TFT:
70   //tft.init(240, 280); // Init ST7789 280x240
71
72   // OR use this initializer (uncomment) if using a 2.0" 320x240 TFT:
73   //tft.init(240, 320); // Init ST7789 320x240
74
75   // OR use this initializer (uncomment) if using a 1.14" 240x135 TFT:
76   //tft.init(135, 240); // Init ST7789 240x135
77
78   // OR use this initializer (uncomment) if using a 1.47" 172x320 TFT:
79   tft.init(172, 320); // Init ST7789 172x320
80   tft.setRotation(2);
81
82   // OR use this initializer (uncomment) if using a 1.9" 170x320 TFT:
83   //tft.init(170, 320); // Init ST7789 170x320
84
```

其中:

- SPI.begin(): 表示启动SPI通信

- tft.init(): 表示初始化TFT LCD，选项分别为宽和高，可以根据显示屏的实际情况设置宽度和高度
- tft.setRotation(): 表示旋转方向，用0、1、2、3分别表示旋转0°、90°、180°、270°

4. 运行效果

修改完成后，编译烧录到开发板，最终的运行结果如下：【动图】

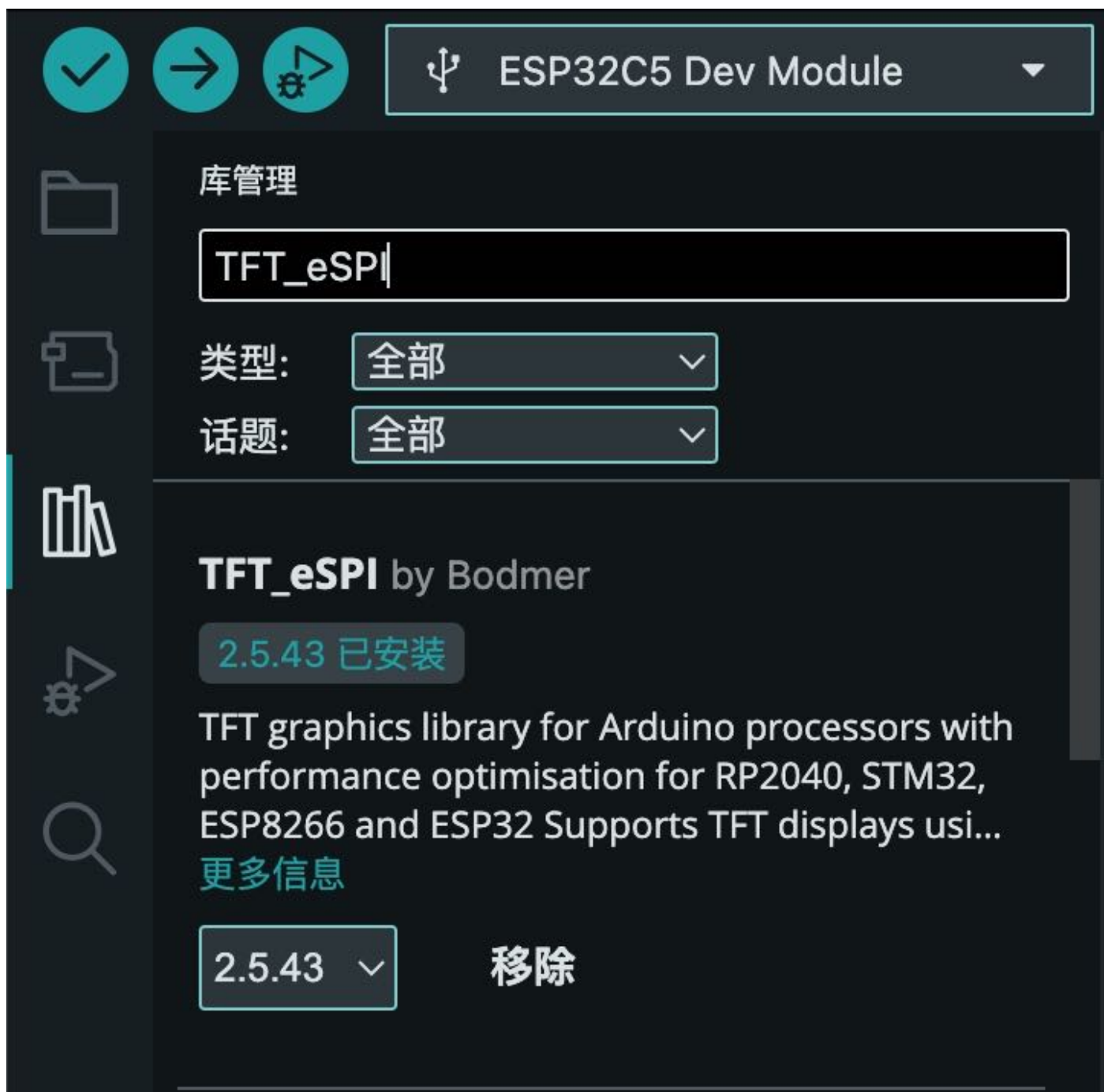


五、TFT_eSPI显示库

前面的两个显示库，使用都比较简单。在Arduino环境中，还有一个让人如雷贯耳的显示库，可以说是显示库中的六边形战士，那就是TFT_eSPI，只是使用的难度，稍微要提升一点。

1. 安装TFT_eSPI定制库

在Arduino IDE的库管理界面中，搜索TFT_eSPI，可以找到对应的库：



可是呢，这个TFT_eSPI的官方版本，还没有支持ESP32-C5。

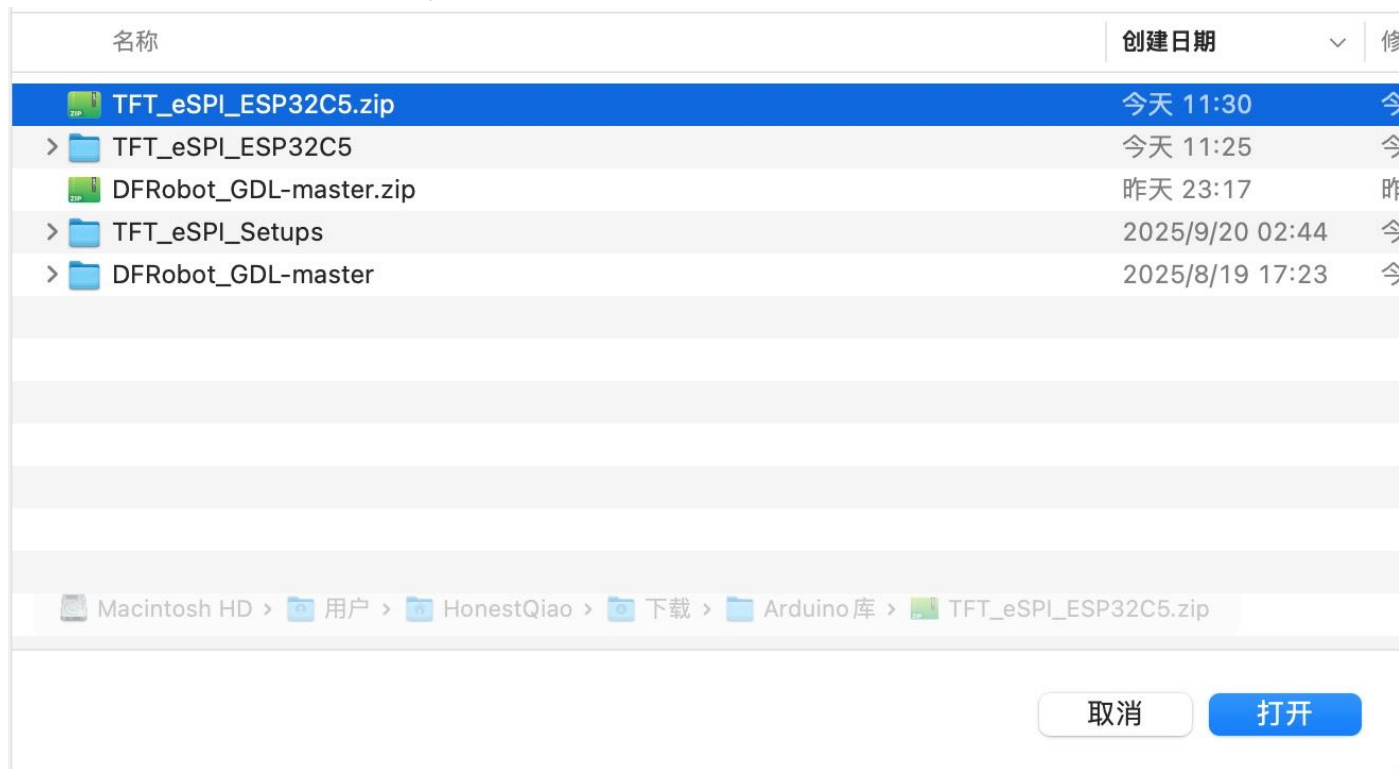
在TFT_eSPI官方git仓库的issue中，有人提出了 [Support For ESP32-C5 ECO2 · Issue #3751](#)，我查看后，经过简单修改处理，也能支持ESP32-C5 ECO1了。

先下载这个压缩包：[TFT_eSPI_ESP32C5.zip](#)

然后，在Arduino IDE中，按照下面的方式，添加**.ZIP库**：

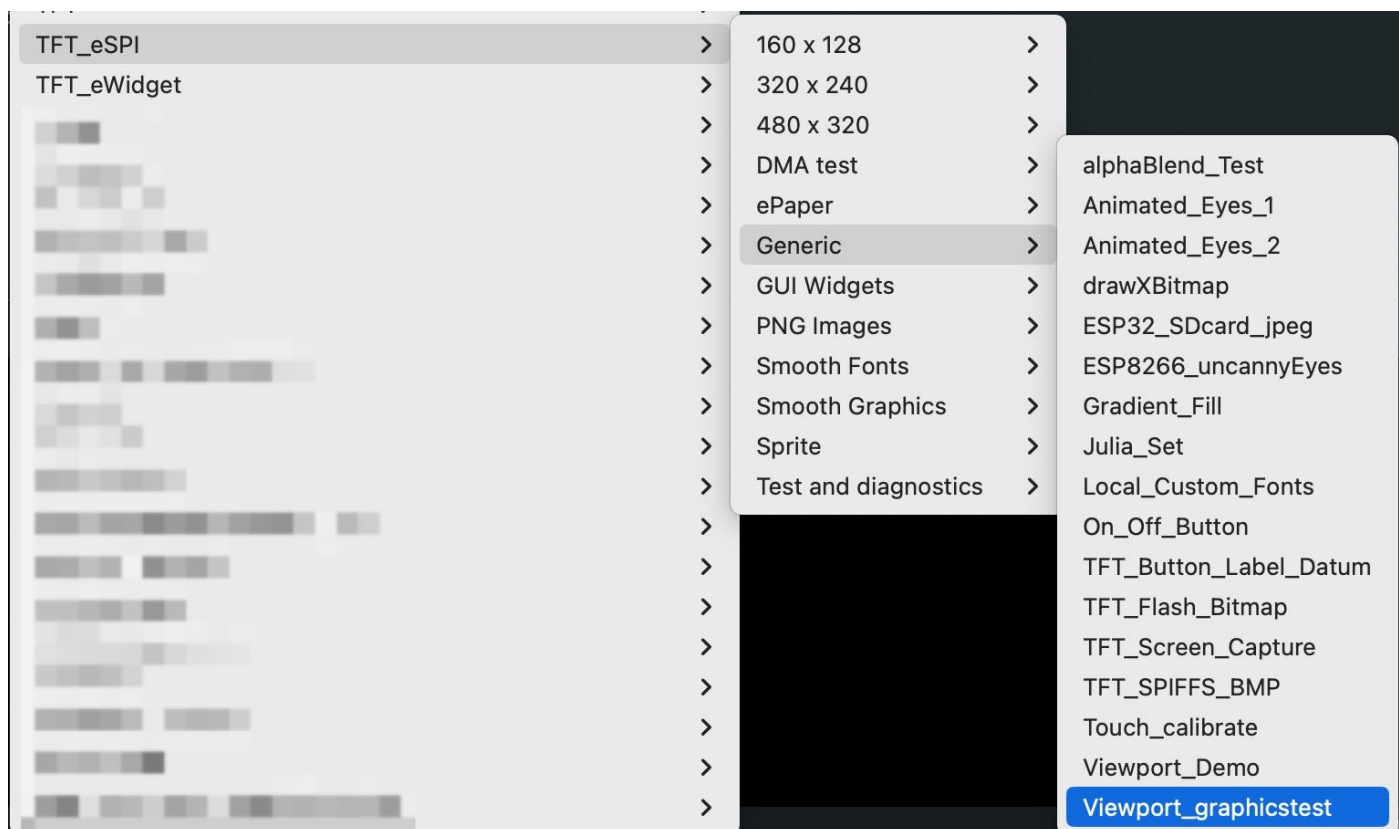


然后在打开的文件选择界面，选择刚才下载的ZIP文件即可安装：



2. 演示代码

TFT_eSPI显示库也提供了很多例子可供参考，帮助我们快速了解具体用法：



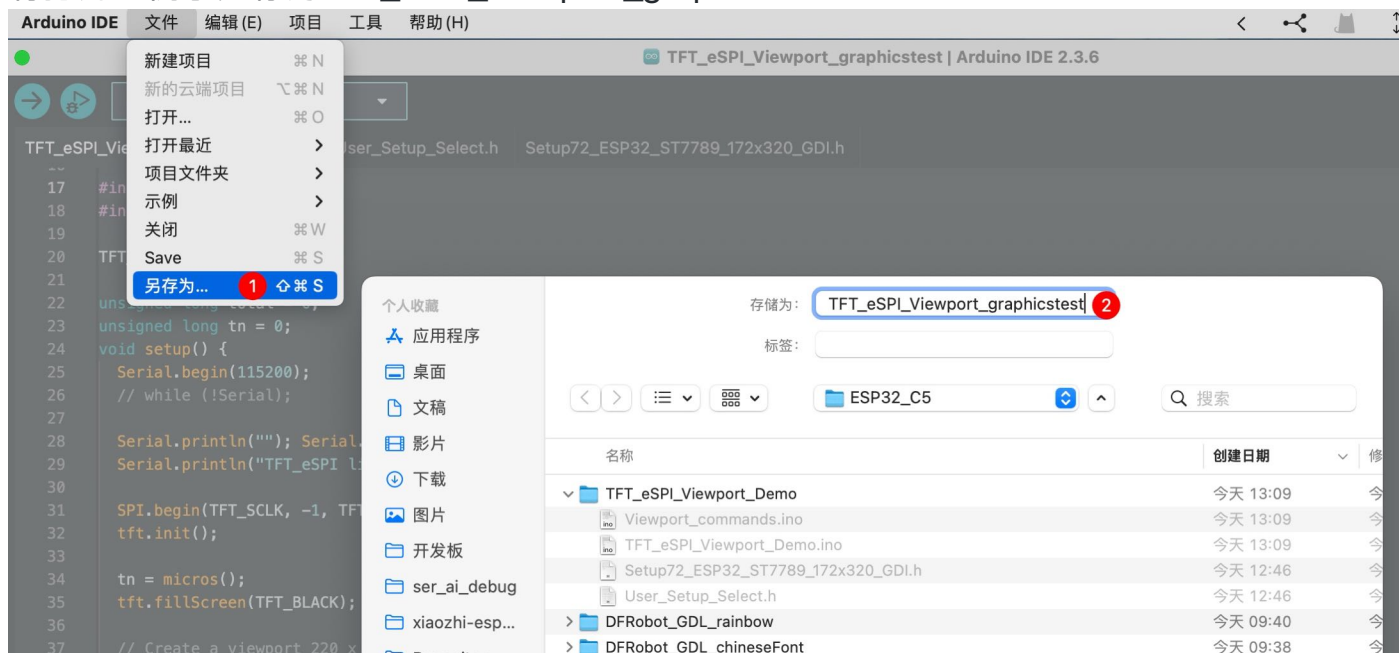
我们选择其中的Viewport_graphicstest例子，其提供了多种显示效果展示。

3. 代码修改

在TFT_eSPI中，也需要进行引脚功能的设置，才能正确使用，不过设置的方法，和前面的两个显示库会有不同，需要按照下面的步骤进行：

1) 保存例子代码

将打开的例子另存为TFT_eSPI_Viewports_graphicstest：



2) 拷贝配置文件

将 TFT_eSPI_ESP32C5.zip 压缩包中的如下文件：

▼ TFT_eSPI_ESP32C5

▼ User_Setups

Setup72_ESP32_ST7789_172x320.h

Setup71_ESP32_S2_ST7789.h

User_Setup.h

User_Setup_Select.h

拷贝到代码保存后的目录，并将 Setup72_ESP32_ST7789_172x320.h 更名为 Setup72_ESP32C5_ST7789_172x320_GDI.h：

▼ TFT_eSPI_Viewports_graphicstest

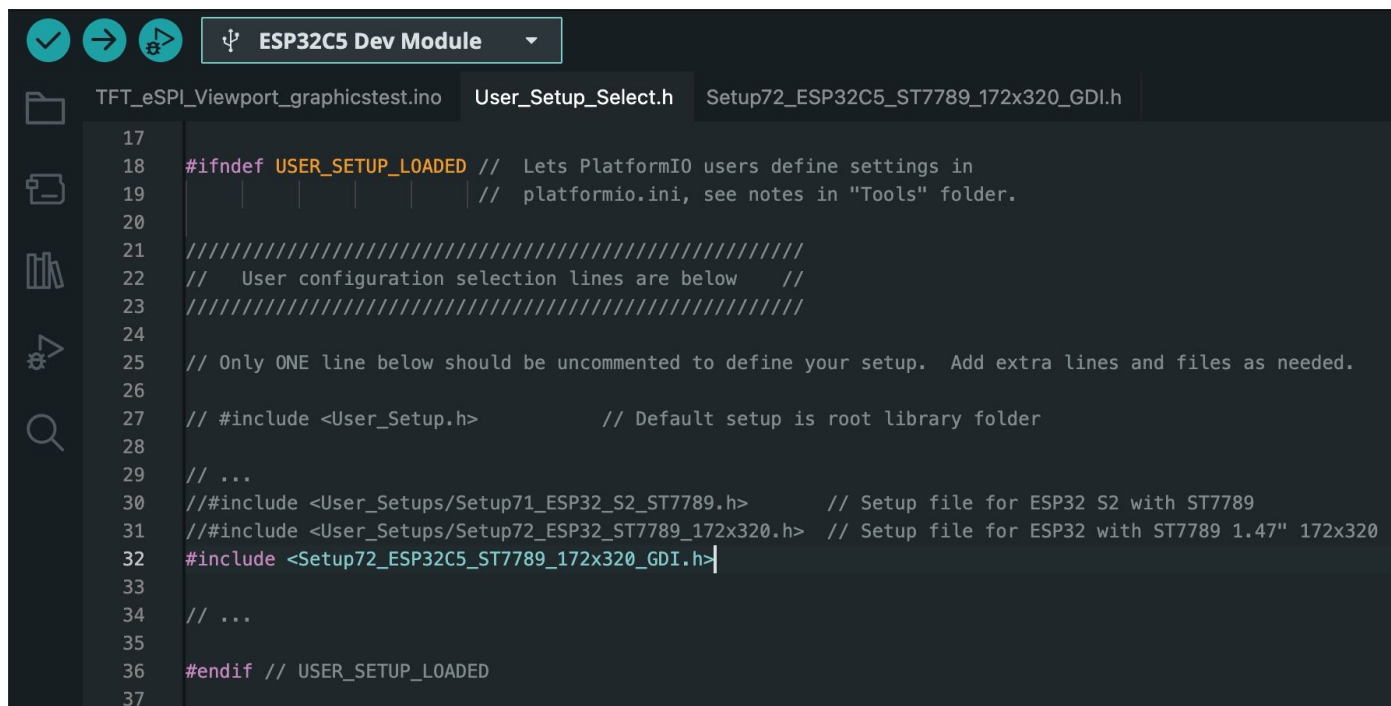
TFT_eSPI_Viewports_graphicstest.ino

User_Setup_Select.h

Setup72_ESP32C5_ST7789_172x320_GDI.h

3) 调用屏幕配置

然后，在 Arduino IDE 中，打开 User_Setup_Select.h 文件，按照下图修改：



```
17
18 #ifndef USER_SETUP_LOADED // Lets PlatformIO users define settings in
19 // platformio.ini, see notes in "Tools" folder.
20
21 //////////////////////////////////////////////////
22 // User configuration selection lines are below //
23 //////////////////////////////////////////////////
24
25 // Only ONE line below should be uncommented to define your setup. Add extra lines and files as needed.
26
27 // #include <User_Setup.h> // Default setup is root library folder
28
29 // ...
30 // #include <User_Setups/Setup71_ESP32_S2_ST7789.h> // Setup file for ESP32 S2 with ST7789
31 // #include <User_Setups/Setup72_ESP32_ST7789_172x320.h> // Setup file for ESP32 with ST7789 1.47" 172x320
32 #include <Setup72_ESP32C5_ST7789_172x320_GDI.h>
33
34 // ...
35
36 #endif // USER_SETUP_LOADED
37
```

也就是注释掉 USER_SETUP_LOADED 区块内所有的语句，并添加：

```
#include <Setup72_ESP32C5_ST7789_172x320_GDI.h>
```

实际上，在 TFT_eSPI 的 User_Setups 目录中，提供了近 100 种常见屏幕的配置：

```

85
86 // #include <User_Setups/Setup51_LilyPi_ILI9481.h> // Setup file for LilyGo LilyPi with ILI9481 display
87 // #include <User_Setups/Setup52_LilyPi_ST7796.h> // Setup file for LilyGo LilyPi with ST7796 display
88
89 // #include <User_Setups/Setup60_RP2040_ILI9341.h> // Setup file for RP2040 with SPI ILI9341
90 // #include <User_Setups/Setup61_RP2040_ILI9341_PIO_SPI.h> // Setup file for RP2040 with PIO SPI ILI9341
91 // #include <User_Setups/Setup62_RP2040_Nano_Connect_ILI9341.h> // Setup file for RP2040 with SPI ILI9341
92
93 // #include <User_Setups/Setup66_Seeed_XIAO_Round.h> // Setup file for Seeed XIAO with GC9A01 240x240
94
95 // #include <User_Setups/Setup70_ESP32_S2_ILI9341.h> // Setup file for ESP32 S2 with SPI ILI9341
96 // #include <User_Setups/Setup70b_ESP32_S3_ILI9341.h> // Setup file for ESP32 S3 with SPI ILI9341
97 // #include <User_Setups/Setup70c_ESP32_C3_ILI9341.h> // Setup file for ESP32 C3 with SPI ILI9341
98 // #include <User_Setups/Setup70d_ILI9488_S3_Parallel.h> // Setup file for ESP32 S3 with SPI ILI9488
99
100 // #include <User_Setups/Setup71_ESP32_S2_ST7789.h> // Setup file for ESP32 S2 with ST7789
101 // #include <User_Setups/Setup72_ESP32_ST7789_172x320.h> // Setup file for ESP32 with ST7789 1.47" 172x320
102
103 // #include <User_Setups/Setup100_RP2040_ILI9488_parallel.h> // Setup file for Pico/RP2040 with 8-bit parallel ILI9488
104 // #include <User_Setups/Setup101_RP2040_ILI9481_parallel.h> // Setup file for Pico/RP2040 with 8-bit parallel ILI9481
105 // #include <User_Setups/Setup102_RP2040_ILI9341_parallel.h> // Setup file for Pico/RP2040 with 8-bit parallel ILI9341
106 // #include <User_Setups/Setup103_RP2040_ILI9486_parallel.h> // Setup file for Pico/RP2040 with 8-bit parallel ILI9486
107 // #include <User_Setups/Setup104_RP2040_ST7796_parallel.h> // Setup file for Pico/RP2040 with 8-bit parallel ST7796
108
109 // #include <User_Setups/Setup105_RP2040_ST7796_16bit_parallel.h> // Setup file for RP2040 16-bit parallel display
110 // #include <User_Setups/Setup106_RP2040_ILI9481_16bit_parallel.h> // Setup file for RP2040 16-bit parallel display
111 // #include <User_Setups/Setup107_RP2040_ILI9341_16bit_parallel.h> // Setup file for RP2040 16-bit parallel display
112 // #include <User_Setups/Setup108_RP2040_ST7735.h> // Setup file for WAVESHARE RP2040 board with onboard ST7735 0.96" 160x80 display
113
114 // #include <User_Setups/Setup135_ST7789.h> // Setup file for ESP8266 and ST7789 135 x 240 TFT
115
116 // #include <User_Setups/Setup136_LilyGo_TTV.h> // Setup file for ESP32 and Lilygo TTV ST7789 SPI bus TFT 135x240
117 // #include <User_Setups/Setup137_LilyGo_TDisplay_RP2040.h> // Setup file for Lilygo T-Display RP2040 (ST7789 on SPI bus with 135x240 TFT)
118

```

你可以根据驱动芯片和屏幕的大小，选择对应的配置文件，或者复制一个进行简单的修改即可。

4) 修改屏幕配置

在 Arduino IDE 中，打开 Setup72_ESP32C5_ST7789_172x320_GDI.h，按照下图修改：

```

1 // Support for 1.47" 320x172 Round Rectangle Color IPS TFT Display
2 #define USER_SETUP_ID 71
3
4 #define ST7789_DRIVER // Full configuration option, define additional parameters below for this display
5
6 #define TFT_RGB_ORDER TFT_BGR // Colour order Blue-Green-Red
7
8 #define TFT_WIDTH 172 // ST7789 172 x 320
9 #define TFT_HEIGHT 320 // ST7789 240 x 320
10
11 #define TFT_BL 15 // 显示屏背光
12 #define TFT_BACKLIGHT_ON HIGH // Level to turn ON back-light (HIGH or LOW)
13
14 #define TFT_SCLK 23 // SPI时钟
15 #define TFT_MOSI 24 // SPI主机输出, 从机输入
16 #define TFT_DC 8 // LCD数据/命令
17 #define TFT_RST 26 // LCD复位
18 #define TFT_CS 27 // TFT片选
19

```

将原有的引脚定义的部分，替换为之前的引脚配置中的宏定义：

```

#define TFT_SCLK 23 // SPI时钟
#define TFT_MOSI 24 // SPI主机输出, 从机输入
#define TFT_BL 15 // 显示屏背光
#define TFT_DC 8 // LCD数据/命令

```

```
#define TFT_RST      26      // LCD复位
#define TFT_CS       27      // TFT片选
```

5) 修改主要代码

接着，在Arduino IDE中，打开TFT_eSPI_ViewPort_graphicstest.ino，按照下图修改：



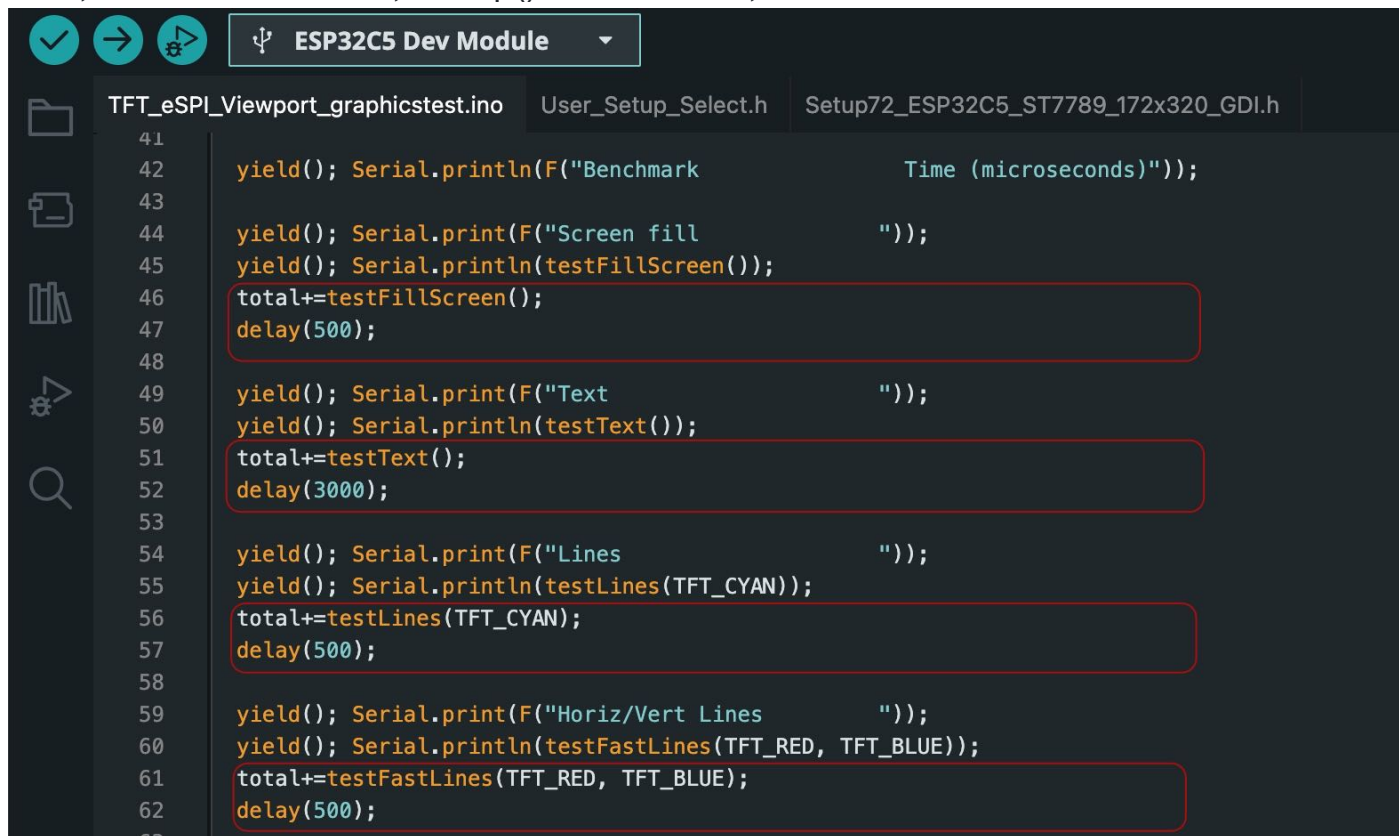
```
TTF_eSPI_ViewPort_graphicstest.ino  User_Setup_Select.h  Setup72_ESP32C5_ST7789_172x320_GDI.h

22  unsigned long total = 0;
23  unsigned long tn = 0;
24  void setup() {
25      Serial.begin(115200);
26      // while (!Serial);
27
28      Serial.println(""); Serial.println("");
29      Serial.println("TFT_eSPI library test!");
30
31      SPI.begin(TFT_SCLK, -1, TFT_MOSI, -1); // 添加的行，用于启动SPI通信
32      tft.init();
33
34      tn = micros();
35      tft.fillScreen(TFT_BLACK);
36
37      // Create a viewport 220 x 300 pixels
38      tft.setViewport(10,10,TFT_WIDTH-11,TFT_HEIGHT-11);
39
40      tft.frameViewport(TFT_RED, -1); // 1 pixel wide frame around viewport
41
42      yield(); Serial.println(F("Benchmark          Time (microseconds)"));
43
108
109  void loop(void) {
110      for (uint8_t rotation = 0; rotation < 4; rotation++) {
111          tft.setRotation(rotation);
112          tft.resetViewport(); // reset viewport to whole screen
113          tft.fillScreen(TFT_BLACK); // so it can be cleared
114
115          // Create a viewport 220 x 300 pixels
116          tft.setViewport(10,10,TFT_WIDTH-11,TFT_HEIGHT-11);
117          tft.frameViewport(TFT_RED, -1); // 1 pixel wide frame around viewport
118
119          testText();
120          delay(2000);
121      }
122  }
123
```

其中修改如下：

- SPI.begin(): 表示启动SPI通信
- tft.setViewport(): 根据屏幕大小，设置对应的显示区域

最后，建议把这个文件中，setup()中的如下区域，按照下图修改：



```
41
42   yield(); Serial.println(F("Benchmark                               Time (microseconds)"));
43
44   yield(); Serial.print(F("Screen fill                               "));
45   yield(); Serial.println(testFillScreen());
46   total+=testFillScreen();
47   delay(500);
48
49   yield(); Serial.print(F("Text                                       "));
50   yield(); Serial.println(testText());
51   total+=testText();
52   delay(3000);
53
54   yield(); Serial.print(F("Lines                                       "));
55   yield(); Serial.println(testLines(TFT_CYAN));
56   total+=testLines(TFT_CYAN);
57   delay(500);
58
59   yield(); Serial.print(F("Horiz/Vert Lines                         "));
60   yield(); Serial.println(testFastLines(TFT_RED, TFT_BLUE));
61   total+=testFastLines(TFT_RED, TFT_BLUE);
62   delay(500);
```

也就是把类似红框圈起来的部分，全部去掉注释，上图在这个区域还有很多类似的代码，请一一去掉注释；否则显示速度太快，你还来不及看清楚怎么了，对应的展示就已经以迅雷不及掩耳之势完成了。

3. 运行效果

修改完成后，编译烧录到开发板，最终的运行结果如下：【动图】



从上图可以看出，刷新的速度，那真是电光火石啊。
这也正是TFT_eSPI的强大之处：极致的刷新速度！

六、LVGL图形显示库

LVGL，无需多言，享誉佛道魔三界，是一款在嵌入式界广受好评的轻量级多功能图形库，官方介绍如是说：

LVGL 是最流行的免费开源嵌入式图形库，可为任何 MCU、MPU 和显示类型创建漂亮的 UI。

从消费电子产品到工业自动化，任何应用程序都可以利用 LVGL 的 30 多个内置小部件、100 多个样式属性、受网络启发的布局以及支持多种语言的排版系统。

这款图形库的强大，就连唐僧来了也得连声称赞：“好！好！好！”
连小米都使用，并且破天荒的没有用小字重新定义LVGL：

“小米一直使用 LVGL 来满足我们的嵌入式图形需求，它已应用于大量设备。我们对它的性能和轻量级特性非常满意。我们也非常重视与 LVGL 开源社区的合作。”

谷瑶瑶，小米，小米
移动软件部高级总监



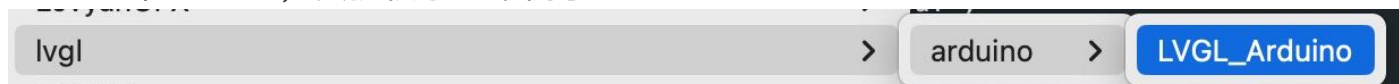
1. 安装LVGL库

在Arduino IDE中安装LVGL很简单，一搜一安就好了：

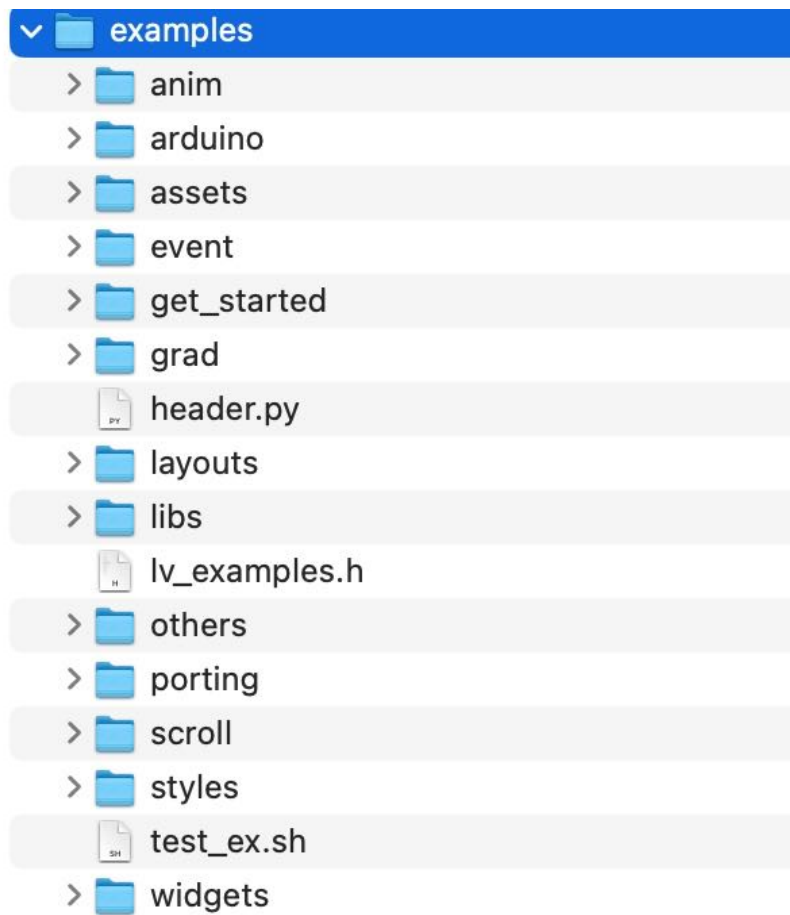
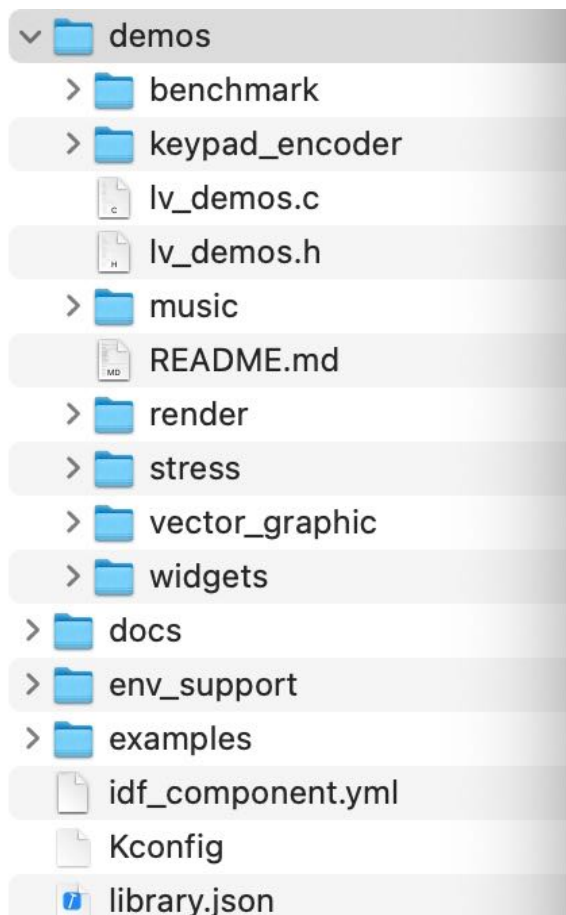


2. 演示代码

Arduino中的LVGL，只提供了一个例子：



不过，这个例子，只是一个入口文件，实际上LVGL本身带有丰富的examples(功能展示)和 demos(完整项目展示)，通过这个入口文件，都可以很方便的调用起来：



感兴趣的话，也可以查看官方的 [Examples](#) 和 [Demos](#) 展示进行学习。

3. 代码修改

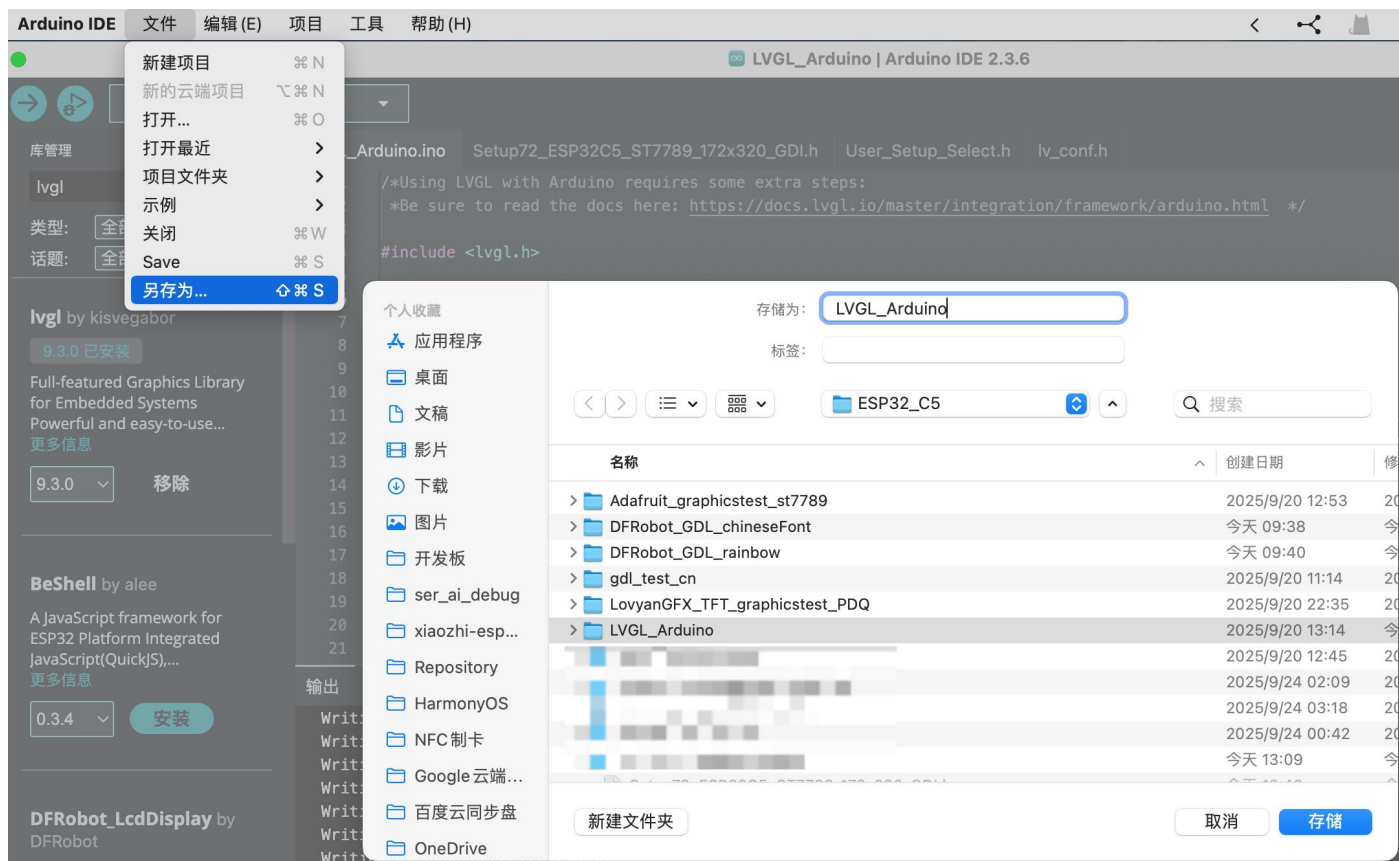
在Arduino环境中，LVGL本身不负责屏幕的驱动，需要使用其他库来驱动屏幕，或者自己写代码来驱动屏幕。

LVGL的Arduino版本与TFT_eSPI深度集成，只需要简单配置，就能完成驱动工作。

现在，按照以下步骤来操作：

1) 保存例子代码

将LVGL例子另存到LVGL_Arduino目录：



2) 拷贝TFT_eSPI配置文件

将本文中TFT_eSPI章节部分对应的配置文件User_Setup_Select.h、Setup72_ESP32C5_ST7789_172x320_GDI.h拷贝到LVGL_Arduino目录中：

名称

- LVGL_Arduino
 - LVGL_Arduino.ino
 - User_Setup_Select.h
 - Setup72_ESP32C5_ST7789_172x320_GDI.h

3) 拷贝LVGL配置文件

打开 我的电脑 - 我的文档 - Arduion目录 - libraries目录：

- Arduino
 - libraries
 - lvgl
 - Kconfig
 - lv_conf_template.h
 - README.md
 - lvgl.h
 - lvgl_private.h

将其中的lv_conf_template.h拷贝到LVGL_Arduino目录中，并更名为lv_conf.h

名称

名称
LVGL_Arduino.ino
lv_conf.h
User_Setup_Select.h
Setup72_ESP32C5_ST7789_172x320_GDI.h

4) 修改lv_conf.h

在Arduino IDE中，打开lv_conf.h，修改如下的行：

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h
13		
14	/* clang-format off */	
15	#if 1 /* Set this to "1" to enable content */	
16		
17	#ifndef LV_CONF_H	
18	#define LV_CONF_H	
19		

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h
1248		
1249	/** Interface for TFT_eSPI */	
1250	#define LV_USE_TFT_ESPI	1
1251		

这两处都设置为1，表示使用TFT_eSPI驱动屏幕。

另外，我一般还会开启如下的配置：

4.1 开启字体配置：

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h
584	/* Montserrat fonts with ASCII range and some symbols using	
585	* https://fonts.google.com/specimen/Montserrat */	
586	#define LV_FONT_MONTSEERRAT_8 0	
587	#define LV_FONT_MONTSEERRAT_10 0	
588	#define LV_FONT_MONTSEERRAT_12 1	
589	#define LV_FONT_MONTSEERRAT_14 1	
590	#define LV_FONT_MONTSEERRAT_16 1	
591	#define LV_FONT_MONTSEERRAT_18 0	
592	#define LV_FONT_MONTSEERRAT_20 0	
593	#define LV_FONT_MONTSEERRAT_22 0	
594	#define LV_FONT_MONTSEERRAT_24 0	

可以根据需要开启字体配置

4.2 打开监控配置

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h	User_Se
1009	/** 1: Enable system monitor component */		
1010	#define LV_USE_SYSMON 1		
1011	#if LV_USE_SYSMON		
1012	/** Get the idle percentage. E.g. uint32_t my_get_idle(void);		
1013	#define LV_SYSMON_GET_IDLE lv_os_get_idle_percent		
1014			
1015	/** 1: Show CPU usage and FPS count.		
1016	* - Requires `LV_USE_SYSMON = 1` */		
1017	#define LV_USE_PERF_MONITOR 1		
1018	#if LV_USE_PERF_MONITOR		
1019	#define LV_USE_PERF_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT		
1020			
1021	/** 0: Displays performance data on the screen; 1: Prints		
1022	#define LV_USE_PERF_MONITOR_LOG_MODE 0		
1023	#endif		
1024			
1025	/** 1: Show used memory and memory fragmentation.		
1026	* - Requires `LV_USE_STDLIB_MALLOC = LV_STDLIB_BUILTIN`		
1027	* - Requires `LV_USE_SYSMON = 1` */		
1028	#define LV_USE_MEM_MONITOR 1		
1029	#if LV_USE_MEM_MONITOR		
1030	#define LV_USE_MEM_MONITOR_POS LV_ALIGN_BOTTOM_LEFT		
1031	#endif		
1032	#endif /*LV_USE_SYSMON*/		

开启选项后，会在屏幕上面显示当前运行的资源消耗和性能情况。

4.3 关闭examples和demos编译

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h
1314	/*=====	
1315	* BUILD OPTIONS	
1316	*=====*/	
1317		
1318	/** Enable examples to be built with the library. */	
1319	#define LV_BUILD_EXAMPLES 0	
1320		
1321	/** Build the demos */	
1322	#define LV_BUILD_DEMOS 0	
1323		

如果不是为了专门要调用examples和demos代码，一般设置为0关闭这两个选项，这样会减少需要编译的文件，加快编译速度。

5) 修改主要代码

在Arduino IDE中，打开LVGL_Arduino.ino，修改下面的各处。

5.1设置分辨率

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h	User_Setup_Select.h
17			
18	/*Set to your screen resolution and rotation*/		
19	#define TFT_HOR_RES TFT_WIDTH		
20	#define TFT_VER_RES TFT_HEIGHT		
21	#define TFT_ROTATION LV_DISPLAY_ROTATION_270		
22			
23	/*LVGL draw into this buffer, 1/10 screen size usually works well. The size is		
24	#define DRAW_BUF_SIZE (TFT_HOR_RES * TFT_VER_RES / 10 * (LV_COLOR_DEPTH / 8))		
25	uint32_t draw_buf[DRAW_BUF_SIZE / 4];		
26			

将TFT_HOR_RES和TFT_VER_RES从原有的数字，修改为在TFT_eSPI中的定义即可。
TFT_ROTATION为旋转方向，根据实际情况调整，LV_DISPLAY_ROTATION_xx的xx可取值为0、90、180、270。

5.2 添加SPI启动代码

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h	User_Setup_Sel
<pre>77 void setup() 78 { 79 String LVGL_Arduino = "Hello Arduino! "; 80 LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_m 81 82 Serial.begin(115200); 83 Serial.println(LVGL_Arduino); 84 85 SPI.begin(TFT_SCLK, -1, TFT_MOSI, -1); // 添加的行，用于启动SPI通信 86 lv_init(); 87 88 /*Set a tick source so that LVGL will know how much time elapsed. */ 89 lv_tick_set_cb(my_tick); 90</pre>			

3. 运行效果

修改完成后，编译烧录到开发板，最终的运行结果如下：



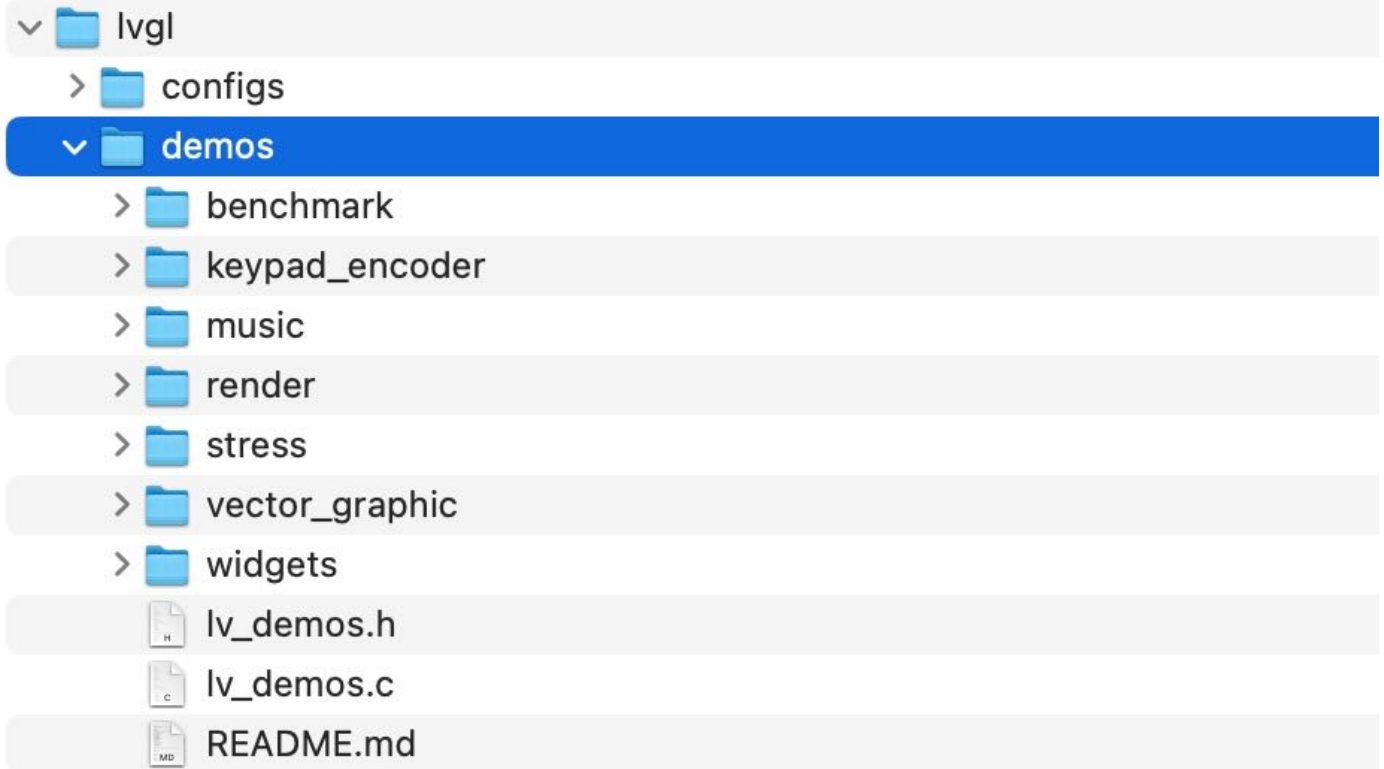
4. music界面效果演示

前面说了，LVGL_Arduino这个例子，只是一个入口文件，那我们再进一步，通过这个入口

文件，把LVGL本身的music demo给跑起来。

1) 拷贝music demo

打开 我的电脑 - 我的文档 - Arduion目录 - libraries目录 - lvgl目录：

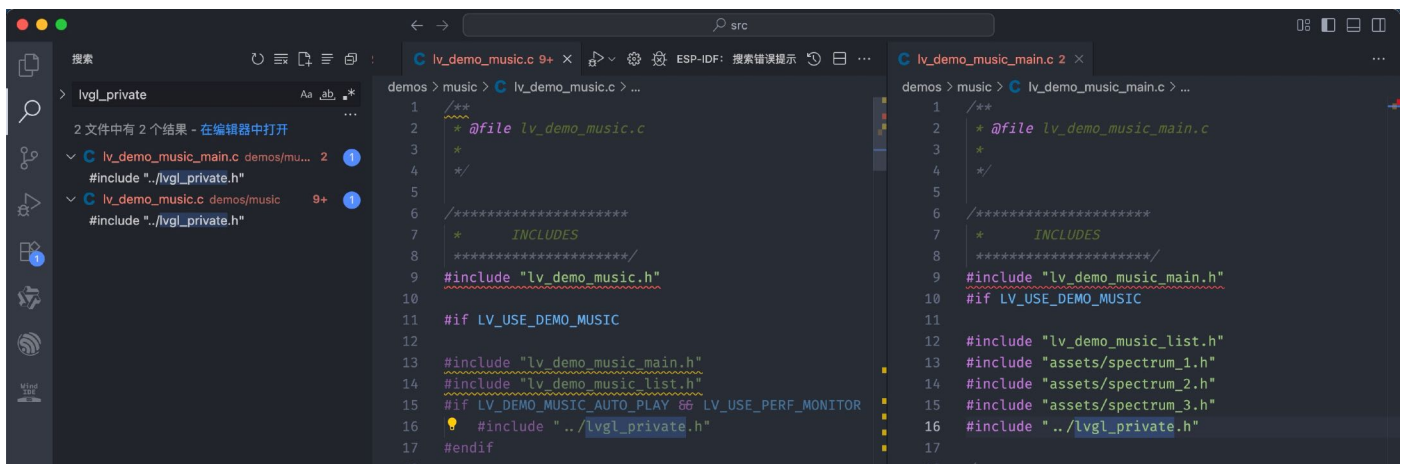


将其中的demos目录，整个拷贝到LVGL_Arduino目录中的src目录下，然后把拷贝后的demo目录中，仅保留music目录和lv_demos.h、lv_demos.c，其他的都删除：



2) 修改music源码

用VSCode等编辑工具，打开LVGL_Arduino目录，然后搜索lvgl_private，然后按照下图修改：



将所有涉及到 lvgl_private.h 的路径，都修改为 ../lvgl_private.h

3) 修改lv_conf.h

在Arduino IDE中，打开lv_conf.h，按照下图修改：

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h	User_Setup_Select.h
------------------	-----------	--------------------------------------	---------------------

```
1320
1321 /** Build the demos */
1322 #define LV_BUILD_DEMOS 1
1323
1324 /*=====
1325  * DEMO USAGE
1326  =====*/
1327
1328 #if LV_BUILD_DEMOS
1329     /** Show some widgets. This might be required to increase `LV_MEM_SIZE`. */
1330     #define LV_USE_DEMO_WIDGETS 0
1331
1332     /** Demonstrate usage of encoder and keyboard. */
1333     #define LV_USE_DEMO_KEYPAD_AND_ENCODER 0
1334
1335     /** Benchmark your system */
1336     #define LV_USE_DEMO_BENCHMARK 0
1337
1338     #if LV_USE_DEMO_BENCHMARK
1339         /** Use fonts where bitmaps are aligned 16 byte and has Nx16 byte stride */
1340         #define LV_DEMO_BENCHMARK_ALIGNED_FONTS 0
1341     #endif
1342
1343     /** Render test for each primitive.
1344     * - Requires at least 480x272 display. */
1345     #define LV_USE_DEMO_RENDER 0
1346
1347     /** Stress test for LVGL */
1348     #define LV_USE_DEMO_STRESS 0
1349
1350     /** Music player demo */
1351     #define LV_USE_DEMO_MUSIC 1
1352     #if LV_USE_DEMO_MUSIC
1353         #define LV_DEMO_MUSIC_SQUARE 0
1354         #define LV_DEMO_MUSIC_LANDSCAPE 0
1355         #define LV_DEMO_MUSIC_ROUND 0
1356         #define LV_DEMO_MUSIC_LARGE 0
1357         #define LV_DEMO_MUSIC_AUTO_PLAY 1
1358     #endif
1359
```

上述修改，是打开music demo，并开启自动演示。因为使用的不是可触摸的显示器，所以需要打开自动演示。

4) 修改主代码

在Arduino IDE中，打开LVGL_Arduino.ino，按照下图修改：

LVGL_Arduino.ino	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h	User_Setup_Select.h
------------------	-----------	--------------------------------------	---------------------

```

14
15 // #include <examples/lv_examples.h>
16 #include <src/demos/lv_demos.h>
17
18 /*Set to your screen resolution and rotation*/
19 #define TFT_HOR_RES    TFT_WIDTH
20 #define TFT_VER_RES    TFT_HEIGHT
21 #define TFT_ROTATION  LV_DISPLAY_ROTATION_270
22

```

LVGL_Arduino.ino ●	lv_conf.h	Setup72_ESP32C5_ST7789_172x320_GDI.h	User_Setup_Select.h
--------------------	-----------	--------------------------------------	---------------------

```

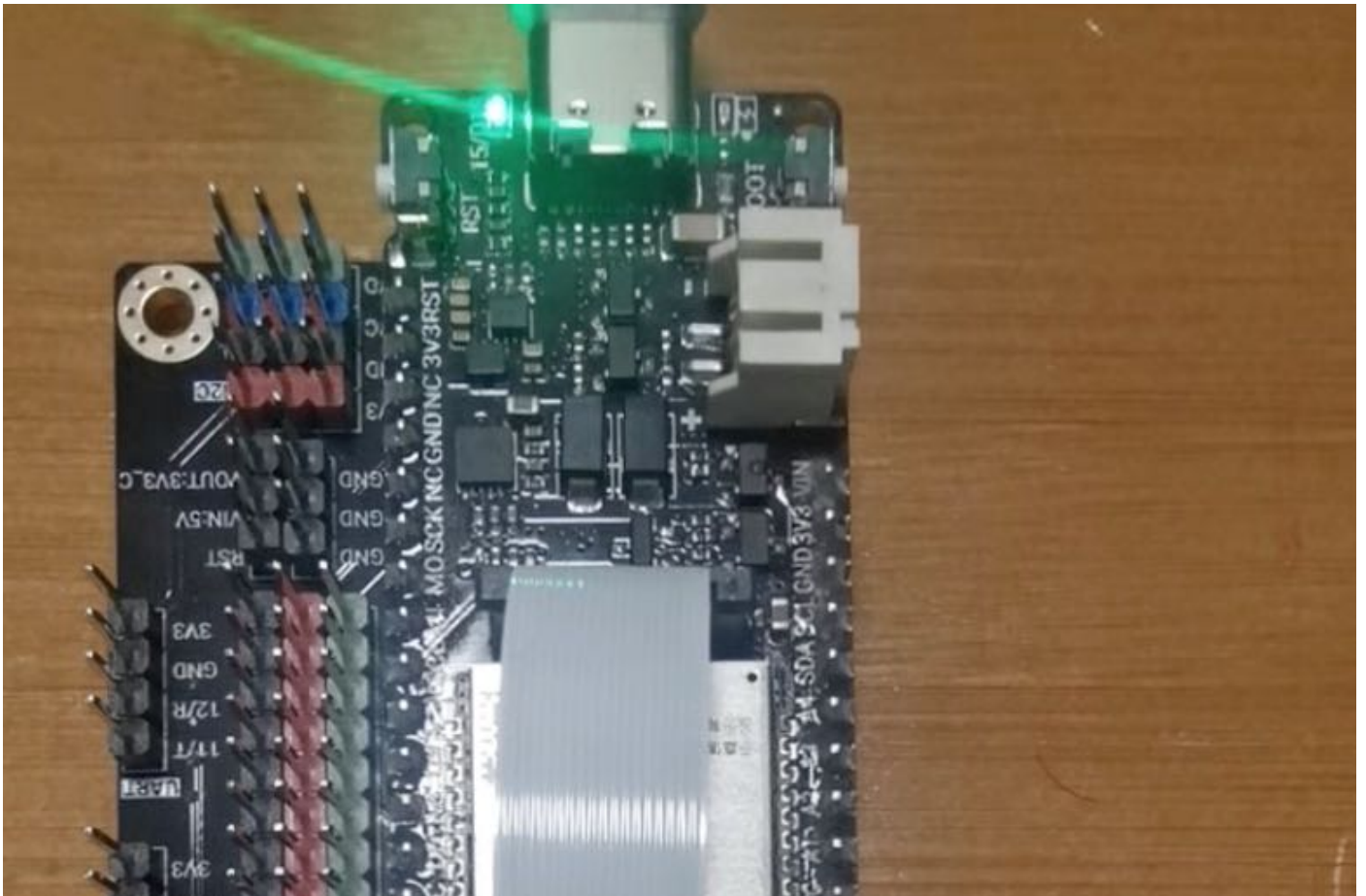
132
133 lv_obj_t *label = lv_label_create( lv_screen_active() );
134 lv_label_set_text( label, "Hello Arduino, I'm LVGL!" );
135 lv_obj_align( label, LV_ALIGN_CENTER, 0, 0 );
136
137 delay(3000);
138 lv_demo_music();
139
140 Serial.println( "Setup done" );
141 }
142

```

上述修改调用demos的头文件，并调用music的入口lv_demo_music()。

5) 界面效果展示

编译烧录后，最终的运行结果如下： [【点击查看视频】](#)





七、OLED点屏

在硬件了解部分，我已经指出过[1.51" 128x64 OLED 透明屏幕](#)不是彩色屏幕，是单色屏幕的。

在DFRobot官方给出的[文档](#)中说明了驱动芯片是SSD1309，使用DFRobot定制的U8g2显示库提供支持，不过U8g2的官方最新版本，已经支持了SSD1309，所以直接使用U8g2最新版本即可。

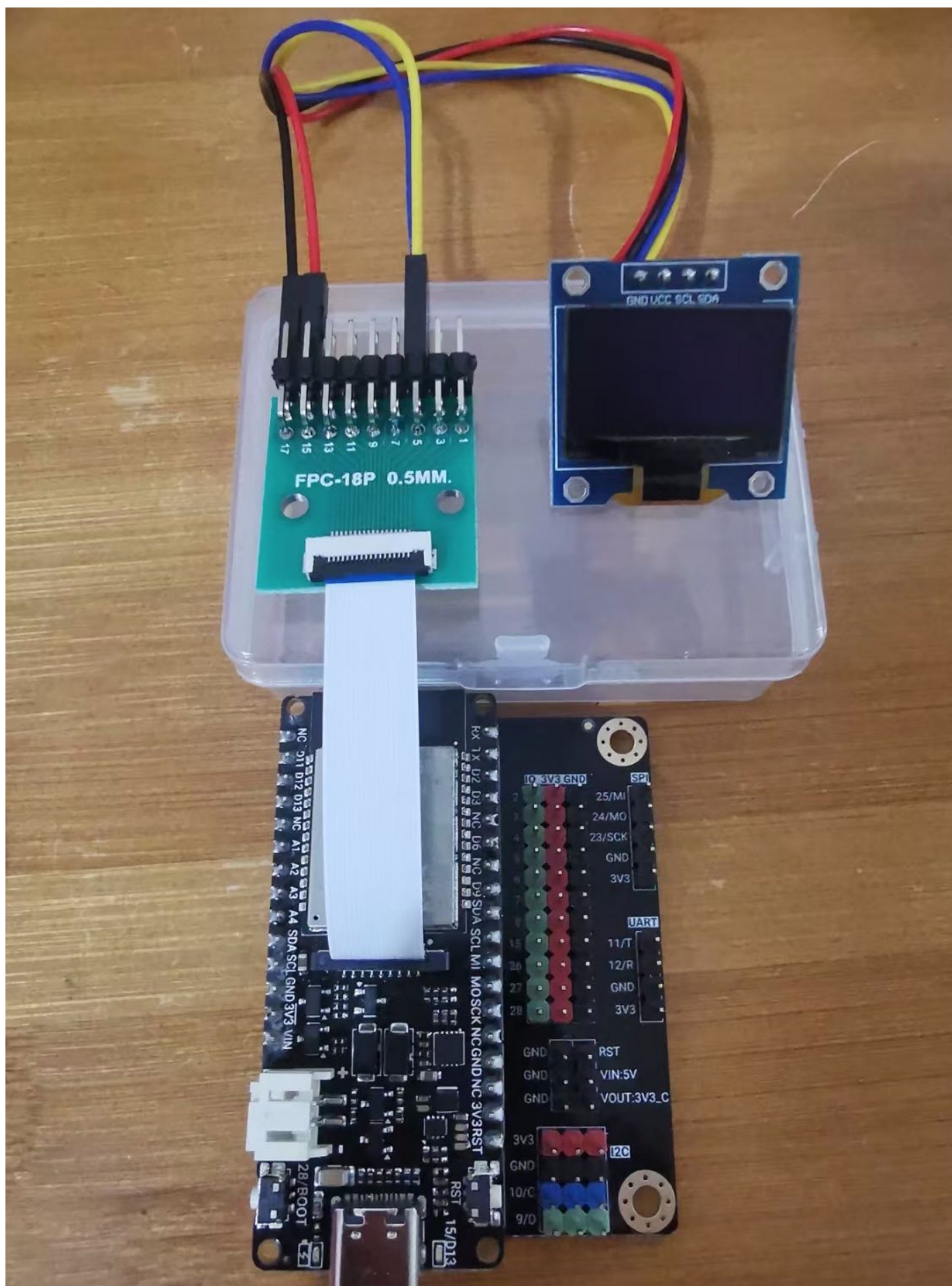
前面说过，TFT_eSPI是Arduino环境中显示库中的六边形战士，那U8g2则是嵌入式单色屏显示库中的小霸王。注意，U8g2不只是在Arduino环境中使用，而是可以移植到几乎所有嵌入式环境中使用，你能看到的单色屏，几乎都被U8g2支持。

我没有[1.51" 128x64 OLED 透明屏幕](#)，但我有一个类似[0.96" 128x64 I2C/SPI OLED单色显示屏](#)的OLED显示屏，使用的是I2C接口。虽然[1.51" OLED 透明屏幕](#)使用的是SPI接口，但是通过U8g2提供的封装，SPI接口和I2C接口两者使用的方法，除了原始接口的定义有差别，其他都是一模一样的。

所以，这一章，以[0.96" 128x64 I2C OLED单色显示屏](#)为例来进行讲解。

通过转接板和杜邦线，我成功把这块[0.96" 128x64 I2C OLED单色显示屏](#)连接到GDI接口的I2C引脚上：

SCL	10/SCL	I2C时钟
SDA	9/SDA	I2C数据



现在我可以宣布：这块0.96英寸128x64 I2C OLED单色显示屏也“支持”GDI接口了：)

1. 安装U8g2显示库

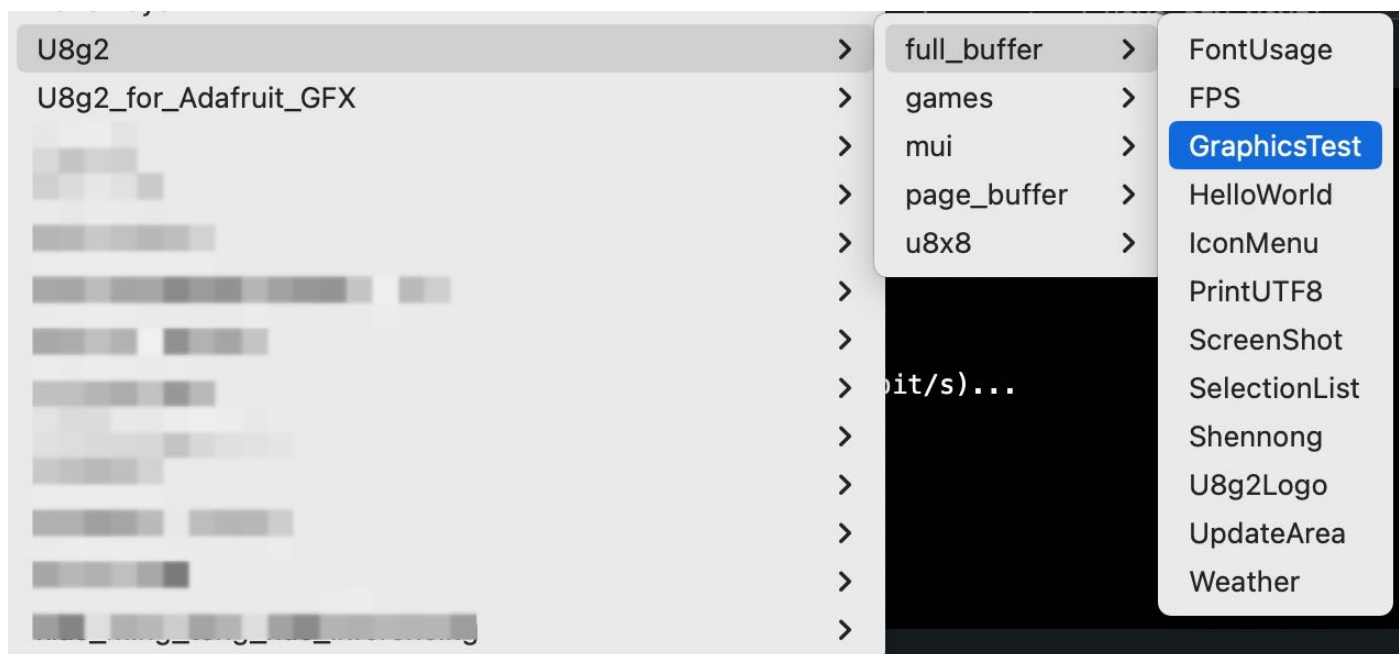
在Arduino IDE的库管理界面中，搜索U8g2，可以找到对应的库：



直接点击安装，或者升级，安装最新版本即可。

2. 演示代码

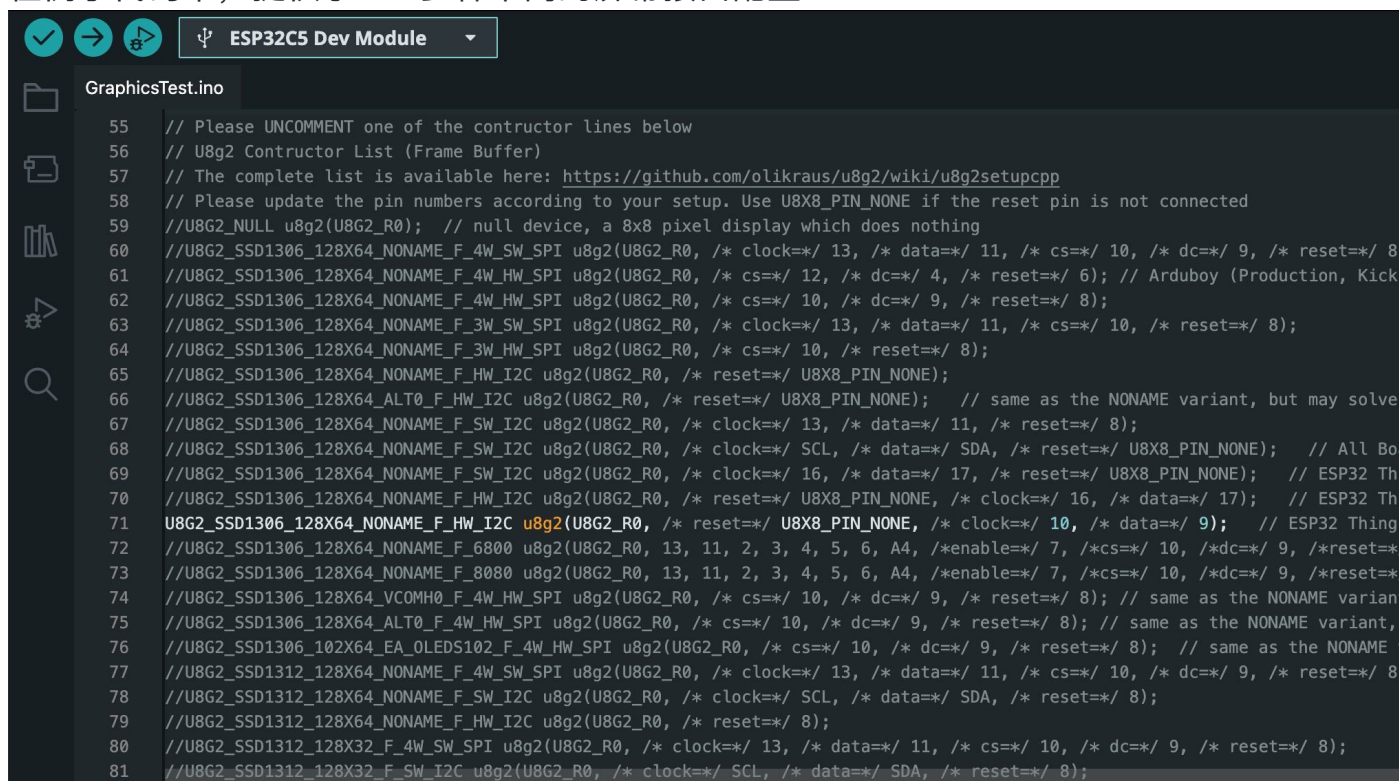
U8g2显示库也提供了很多例子可供参考，帮助我们快速了解具体用法：



我们选择其中的GraphicsTest例子，其提供了多种显示效果展示。

3. 修改接口配置

在例子代码中，提供了300多种不同的屏幕接口配置：



如果使用0.96" 128x64 I2C OLED单色显示屏，则可以使用上图中的配置：

```
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/  
U8X8_PIN_NONE, /* clock=*/ 10, /* data=*/ 9);
```

如果使用1.51" 128x64 OLED 透明屏幕，则可以使用下图中的配置：


```
ESP32C5 Dev Module
GraphicsTest.ino
166 //U8G2_SSD1305_128X32_NONAME_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
167 //U8G2_SSD1305_128X32_ADAFRUIT_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
168 //U8G2_SSD1305_128X32_ADAFRUIT_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
169 //U8G2_SSD1305_128X64_ADAFRUIT_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
170 //U8G2_SSD1305_128X64_ADAFRUIT_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
171 //U8G2_SSD1305_128X64_RAYSTAR_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
172 //U8G2_SSD1305_128X64_RAYSTAR_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
173 //U8G2_SSD1309_128X64_NONAME0_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
174 //U8G2_SSD1309_128X64_NONAME0_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
175 //U8G2_SSD1309_128X64_NONAME2_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
176 //U8G2_SSD1309_128X64_NONAME2_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
177 U8G2_SSD1309_128X64_NONAME2_1_4W_HW_SPI u8g2(/* rotation=*/U8G2_R0, /* cs=*/ 27, /* dc=*/ 8, /* reset=*/ 26);
178 //U8G2_SSD1316_128X32_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
179 //U8G2_SSD1316_128X32_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
180 //U8G2_SSD1316_96X32_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
181 //U8G2_SSD1316_96X32_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
182 //U8G2_SSD1317_96X96_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8); // not tested,
183 //U8G2_SSD1317_96X96_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8); // not tested, not confirmed
184 //U8G2_SSD1318_128X96_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 13, /* data=*/ 11, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
185 //U8G2_SSD1318_128X96_F_4W_HW_SPI u8g2(U8G2_R0, /* cs=*/ 10, /* dc=*/ 9, /* reset=*/ 8);
186 //U8G2_LD7032_60X32_F_4W_SW_SPI u8g2(U8G2_R0, /* clock=*/ 11, /* data=*/ 12, /* cs=*/ 9, /* dc=*/ 10, /* reset=*/ 8); // SW SPI Nano Bo
187 //U8G2_LD7032_60X32_F_4W_HW_SPI u8g2(U8G2_R0, /* clock=*/ 11, /* data=*/ 12, /* cs=*/ 9, /* dc=*/ 10, /* reset=*/ 8); // NOT TESTED
```

```
U8G2_SSD1309_128X64_NONAME2_1_4W_HW_SPI u8g2(/* rotation=*/U8G2_R0, /*
cs=*/ 27, /* dc=*/ 8, /* reset=*/ 26);
```

如果你使用的是其他规格的屏幕，可以选择对应的配置使用，并将其他的配置全部注释掉。

做好上述对应接口的设置以后，后续操作，就完全一样了。

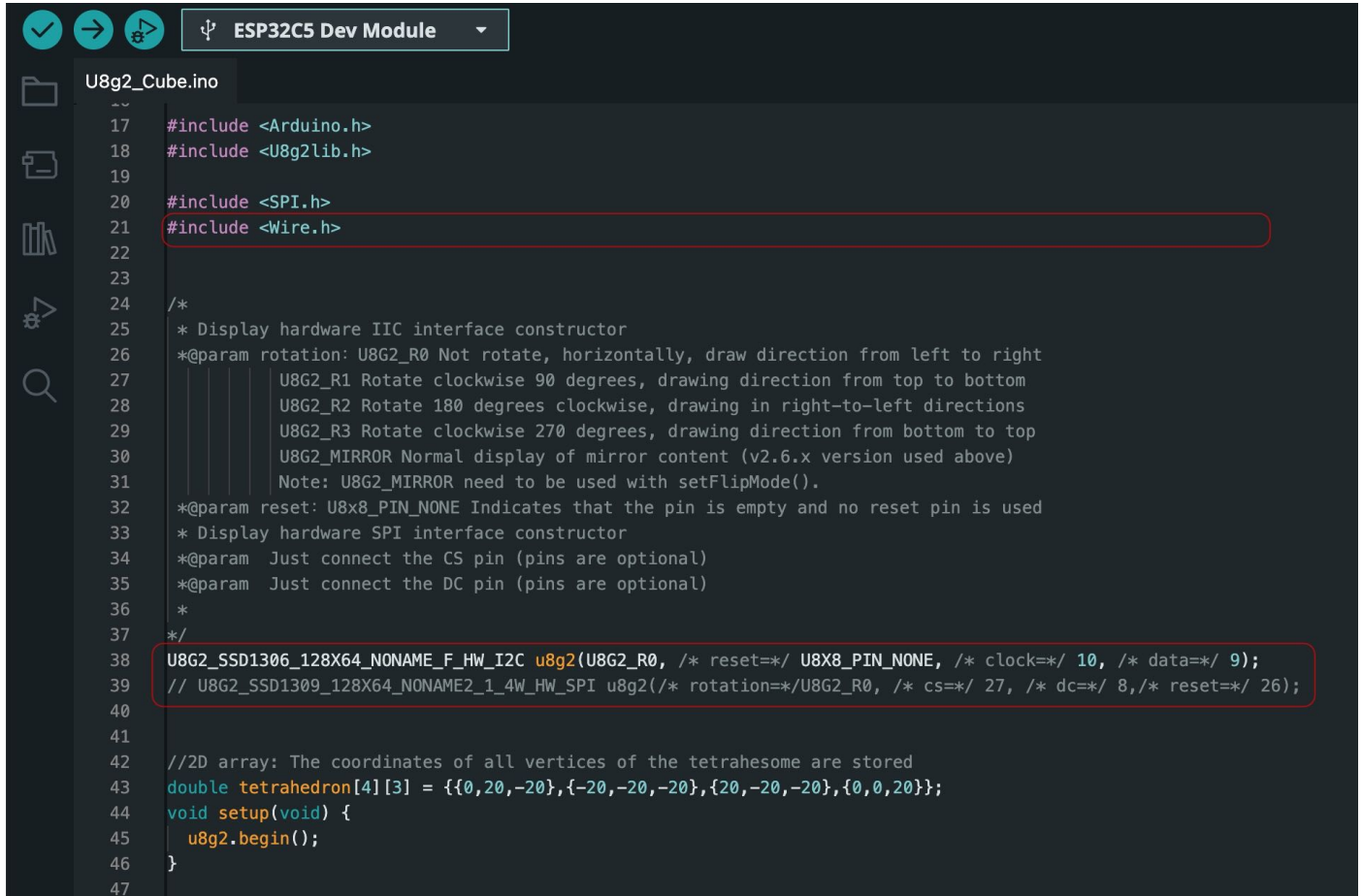
4. 运行效果

修改完成后，编译烧录到开发板，最终的运行结果如下：【动图】



5. cube(三维几何体立方体)效果展示

在DFRobot定制的U8g2显示库中，有一个Cube的例子，可以从查看文件 [DFRobot_Demo/1.51 inch OLED12864-SSD1309/Cube/Cube.ino](#)，点击一键复制，然后在Arduino IDE中新建一个文件，并拷贝进去：



```
U8g2_Cube.ino
17 #include <Arduino.h>
18 #include <U8g2lib.h>
19
20 #include <SPI.h>
21 #include <Wire.h>
22
23
24 /*
25  * Display hardware IIC interface constructor
26  * @param rotation: U8G2_R0 Not rotate, horizontally, draw direction from left to right
27  *                U8G2_R1 Rotate clockwise 90 degrees, drawing direction from top to bottom
28  *                U8G2_R2 Rotate 180 degrees clockwise, drawing in right-to-left directions
29  *                U8G2_R3 Rotate clockwise 270 degrees, drawing direction from bottom to top
30  *                U8G2_MIRROR Normal display of mirror content (v2.6.x version used above)
31  *                Note: U8G2_MIRROR need to be used with setFlipMode().
32  * @param reset: U8X8_PIN_NONE Indicates that the pin is empty and no reset pin is used
33  * Display hardware SPI interface constructor
34  * @param Just connect the CS pin (pins are optional)
35  * @param Just connect the DC pin (pins are optional)
36  *
37  */
38 U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE, /* clock=*/ 10, /* data=*/ 9);
39 // U8G2_SSD1309_128X64_NONAME2_1_4W_HW_SPI u8g2(/* rotation=*/ U8G2_R0, /* cs=*/ 27, /* dc=*/ 8, /* reset=*/ 26);
40
41
42 //2D array: The coordinates of all vertices of the tetrahedron are stored
43 double tetrahedron[4][3] = {{0,20,-20},{-20,-20,-20},{20,-20,-20},{0,0,20}};
44 void setup(void) {
45     u8g2.begin();
46 }
47
```

按照上图，添加Wire.h的调用，并根据使用的具体屏幕，做好对应的接口设置。

然后，修改下图中的数值为1:

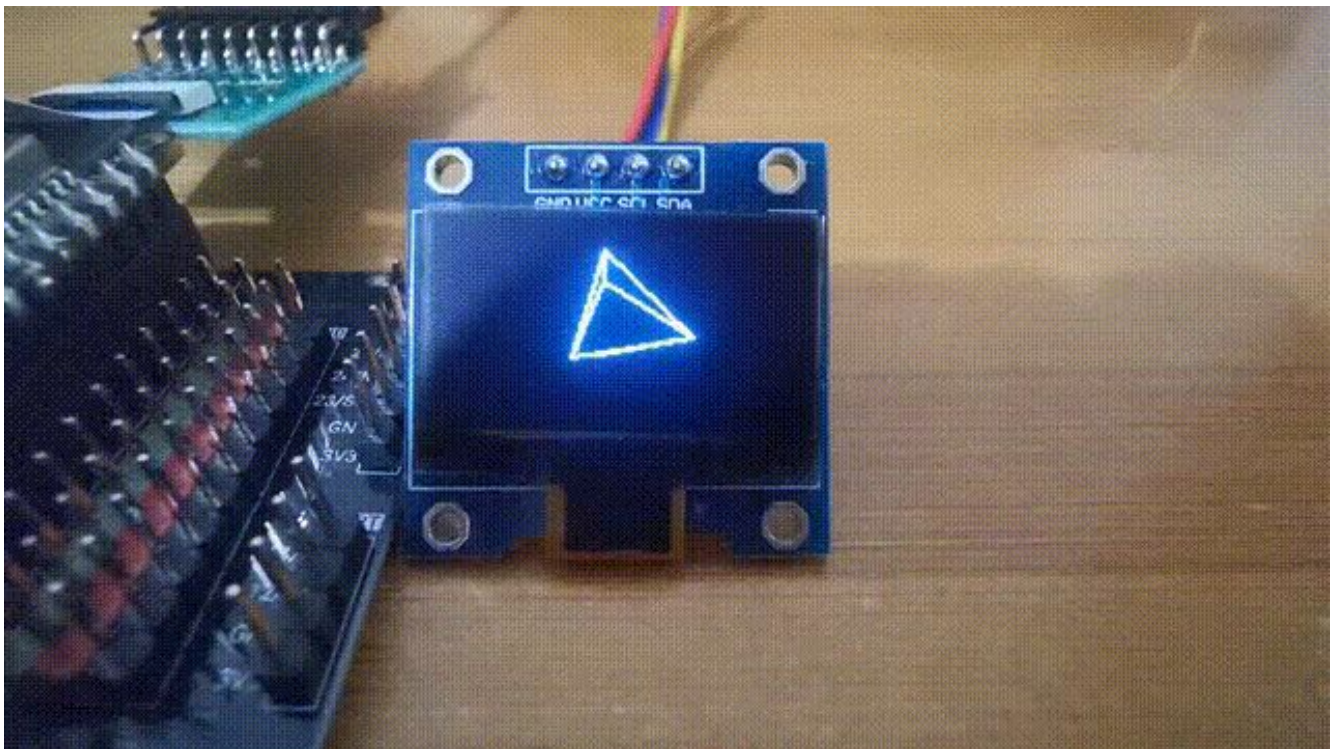

```
ESP32C5 Dev Module

U8g2_Cube.ino

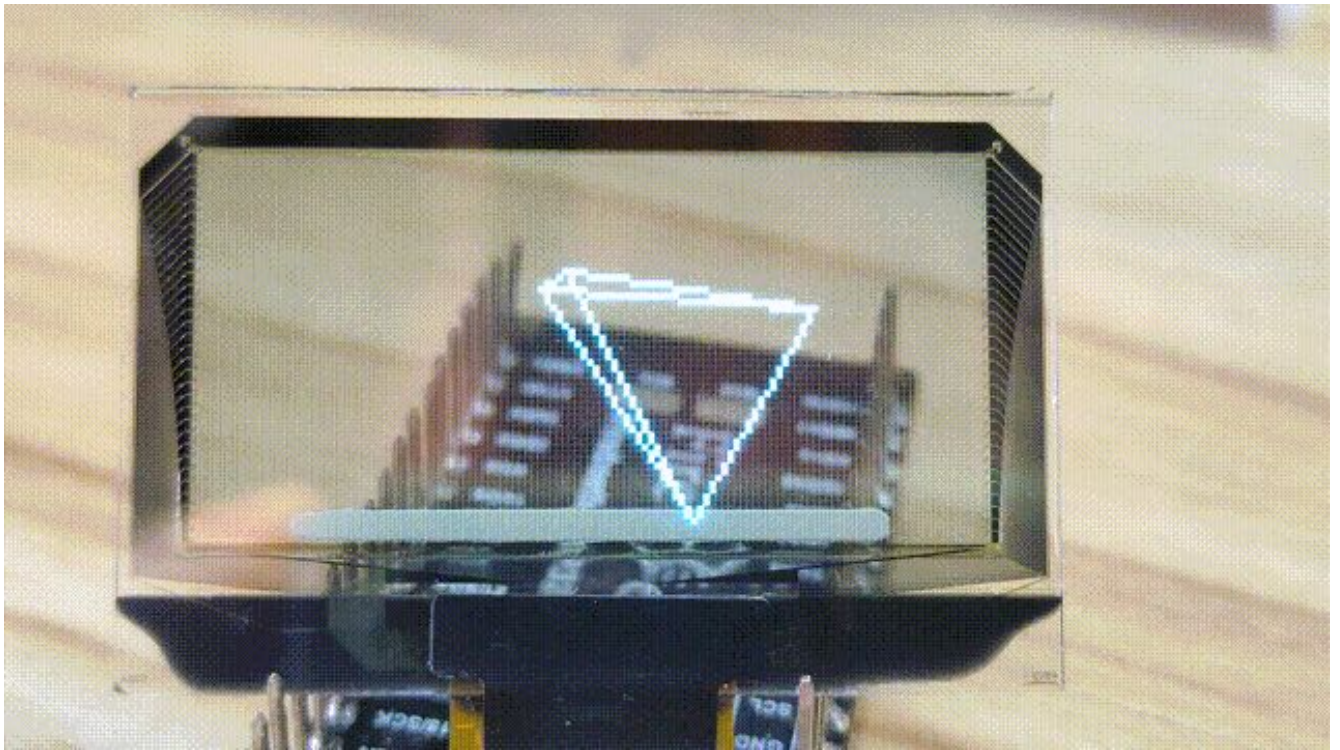
47
48 void loop(void) {
49     /*
50      * firstPage will change the current page number position to 0
51      * When modifications are between firstpage and nextPage, they will be re-rendered at e
52      * This method consumes less ram space than sendBuffer
53      */
54     u8g2.firstPage();
55     do {
56         //Connect the corresponding points inside the tetrahedron together
57         u8g2.drawLine(0xyzTo0u(tetrahedron[0][0], tetrahedron[0][2]), 0xyzTo0v(tetrahedron[0][1], t
58         u8g2.drawLine(0xyzTo0u(tetrahedron[1][0], tetrahedron[1][2]), 0xyzTo0v(tetrahedron[1][1], t
59         u8g2.drawLine(0xyzTo0u(tetrahedron[0][0], tetrahedron[0][2]), 0xyzTo0v(tetrahedron[0][1], t
60         u8g2.drawLine(0xyzTo0u(tetrahedron[0][0], tetrahedron[0][2]), 0xyzTo0v(tetrahedron[0][1], t
61         u8g2.drawLine(0xyzTo0u(tetrahedron[1][0], tetrahedron[1][2]), 0xyzTo0v(tetrahedron[1][1], t
62         u8g2.drawLine(0xyzTo0u(tetrahedron[2][0], tetrahedron[2][2]), 0xyzTo0v(tetrahedron[2][1], t
63         // Rotate 0.1°
64         rotate(1);
65
66     } while ( u8g2.nextPage() );
67     //delay(50);
68 }
69 /*!
70  * @brief Convert xz in the three-dimensional coordinate system 0xyz
71  * into the u coordinate inside the two-dimensional coordinate system 0uv
72  * @param x in 0xyz
```

修改完成后，编译烧录到开发板，最终的运行结果如下：【动图】

- 0.96" 128x64 I2C OLED单色显示屏



- 1.51" 128x64 OLED 透明屏幕



八、总结

在DFRobot支持GDI接口的开发板上，连接使用带有GDI接口的显示屏，真的是非常、非常、非常的方便，一线直连(FPC线)，再也不用换了块板子又要繁琐的连线了。

同时，得益于Arduino环境的开放，我们能够使用多种显示库，来方便的驱动我们所使用的屏幕，可以把精力放在好好做项目和界面呈现上。

这篇文章，算是对Arduino+ESP32环境点屏的一次个人总结，分享给大家希望能够给大家带来帮助。我所拥有的显示屏也非常有限，所以并不能面面俱到。还有很多有特色的显示屏及显示库，我未能一一体验，或限于篇幅，未在本文中全部罗列。

另外，在编写本文的过程中，参考了很多的文章和开源资料，也得到多位技术大佬的指点，在此深表感谢。

如果大家在点屏的过程中，有心得、有体会、有想法、有疑问，随时欢迎一起交流，让我们一起点亮屏幕，探索更多可能吧！