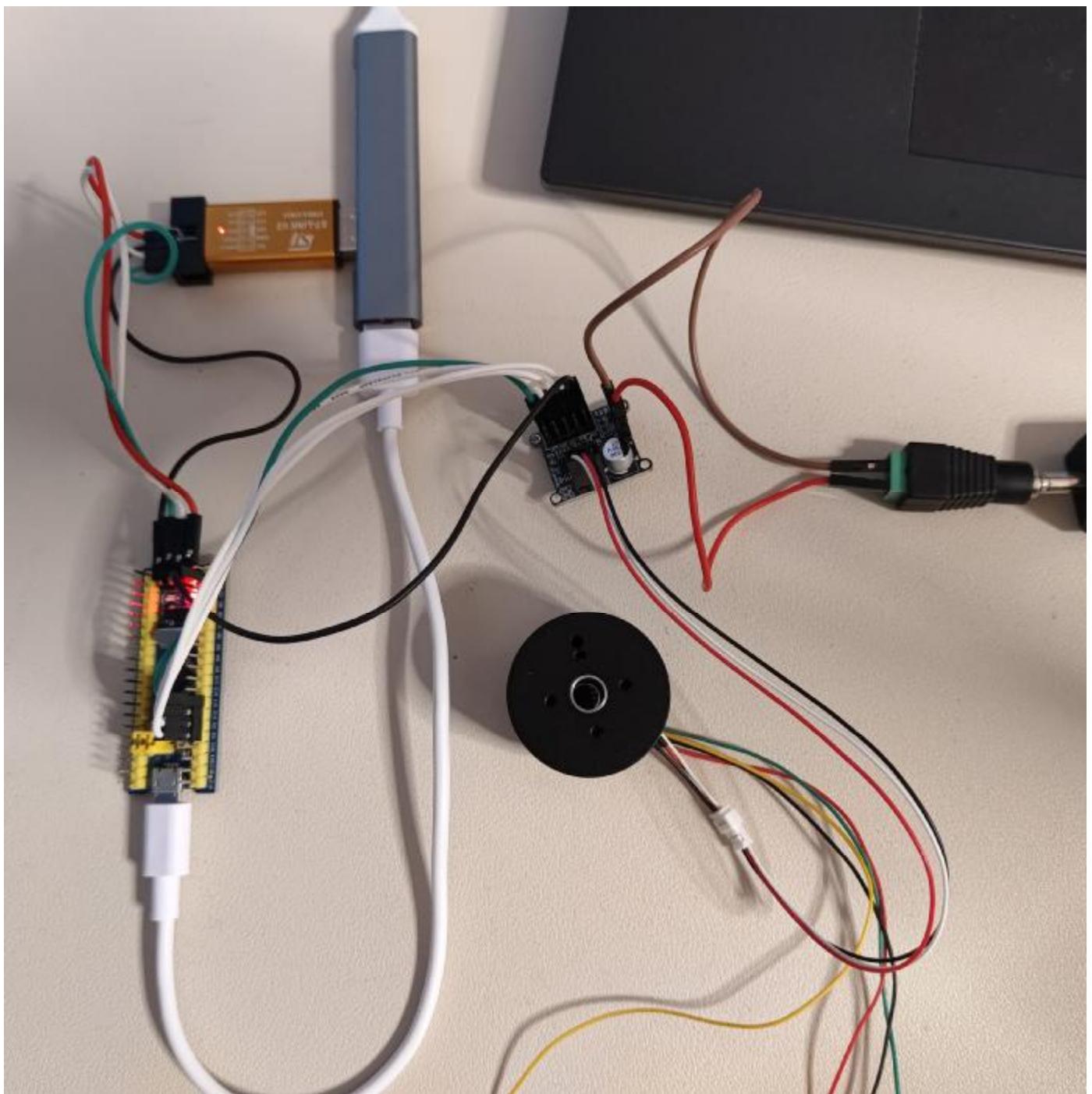


STM32 引用 SimpleFOC 库开环驱动直流无刷电机

M 创动工坊提供 mcdgf.taobao.com

一、硬件准备



STM32 核心板, 2804 电机和 SimpleFOC mini 驱动板 (M 创动工坊提供), ST-link, 12V 直流电源, USB 线等

二、软件准备

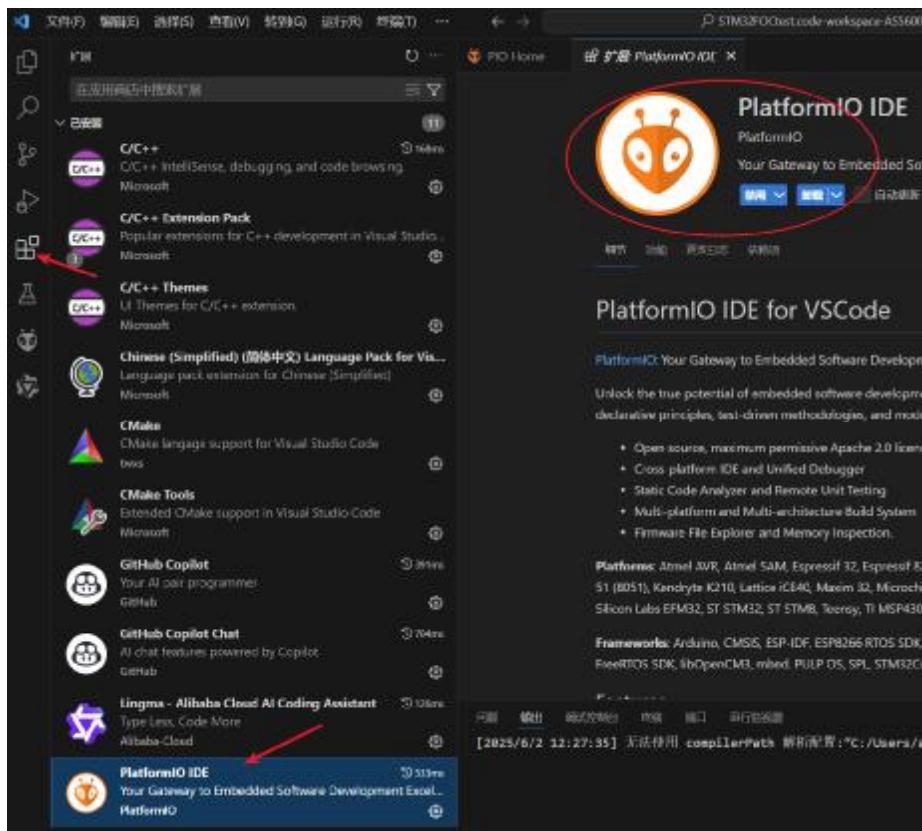
1. 安装微软的 Vscode, 网上很多教程, 且有说明书



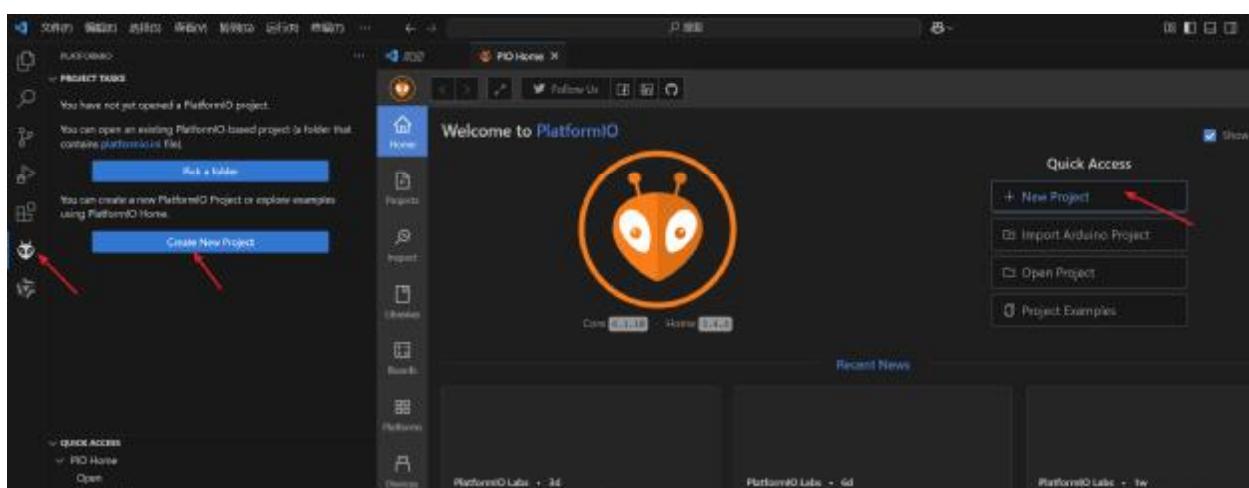
安装好 stlink 驱动



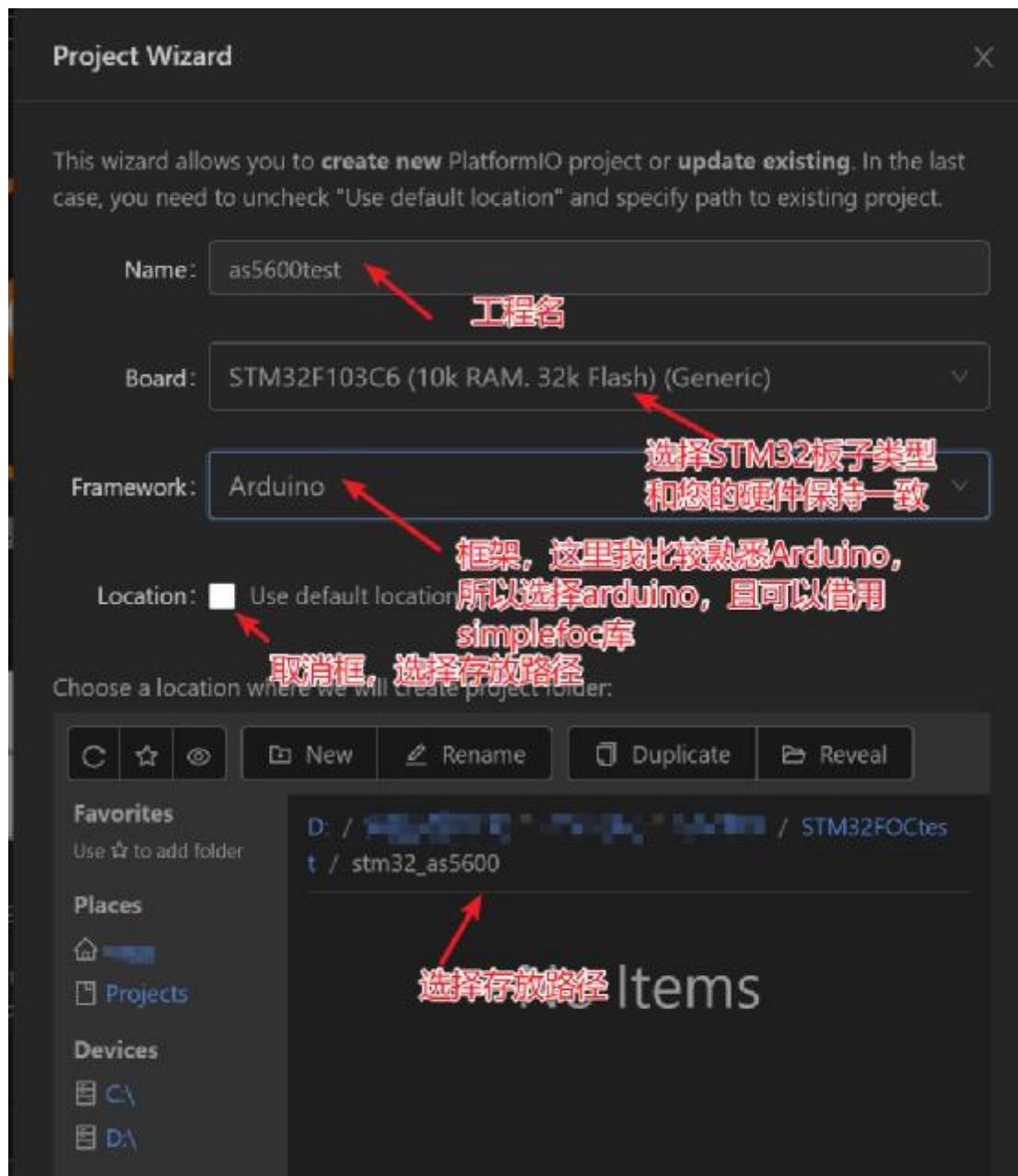
2. 打开 vscode 并安装 PlatformIO 插件。其他的一些中文插件、AI 插件等等，根据需要安装



3. 打开插件，新建工程



4. 重要的设置



5. Ini 文件设置, 这是关键

```

[env:genericSTM32F103C6]
platform = ststm32
board = genericSTM32F103C6
framework = arduino

```

Ini文件设置

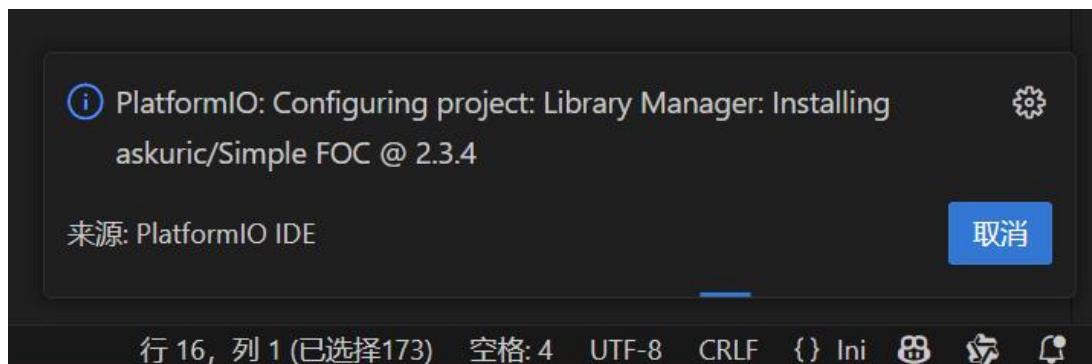
The screenshot shows the PlatformIO IDE interface. On the left is a tree view of the project structure under 'ASS5600TEST': .pio, .vscode, include, lib, src, test, .gitignore, and platformio.ini. The right pane displays the contents of the 'platformio.ini' file:

```
platformio.ini
1 ; PlatformIO Project Configuration file
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:genericSTM32F103C6]
12 platform = ststm32
13 board = genericSTM32F103C6
14 framework = arduino
15
16 lib_deps =
17     askuric/Simple FOC@ 2.3.4
18 board_build.mcu = stm32f103c6t6
19 board_upload.maximum_size = 32768
20 upload_protocol = stlink
21 debug_tool = stlink
22 build_flags = -Os
23
```

A red box highlights the 'lib_deps' section, and three numbered callouts point to it: 1. 用SimpleFOC库, 2. 定义板子, 3. 定义使用stlink协议。

```
lib_deps =
askuric/Simple FOC@ 2.3.4
board_build.mcu = stm32f103c6t6
board_upload.maximum_size = 32768
upload_protocol = stlink
debug_tool = stlink
build_flags = -Os
```

写好后，记得点保存，就开始自动下载库，右下角



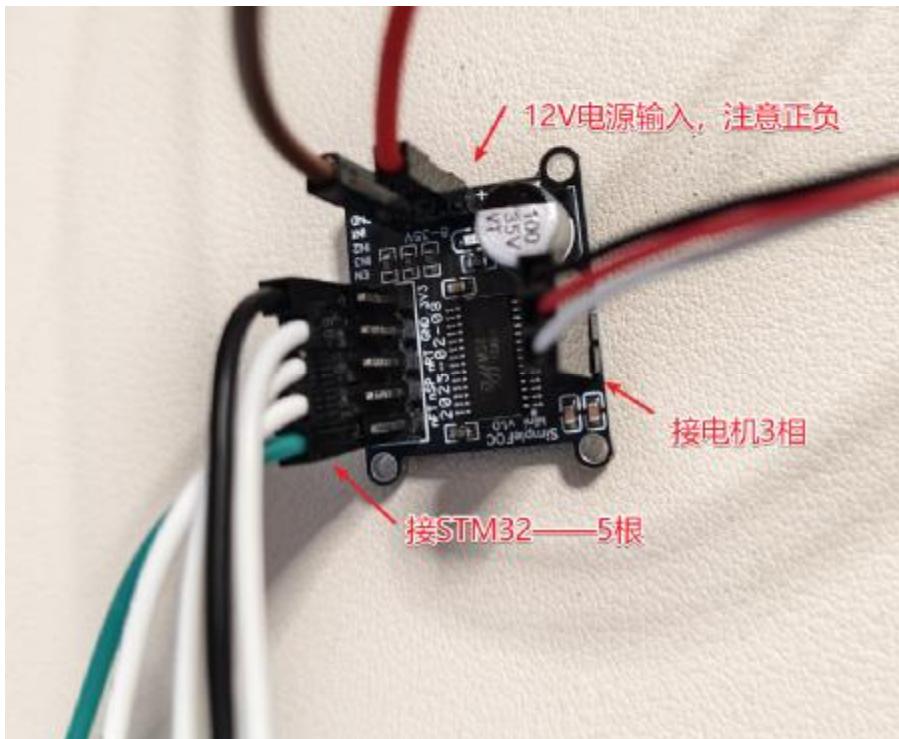
设置到这里就结束了。

三、接线

根据程序定义，接线

Simplefoc mini 板与 STM32 接线：

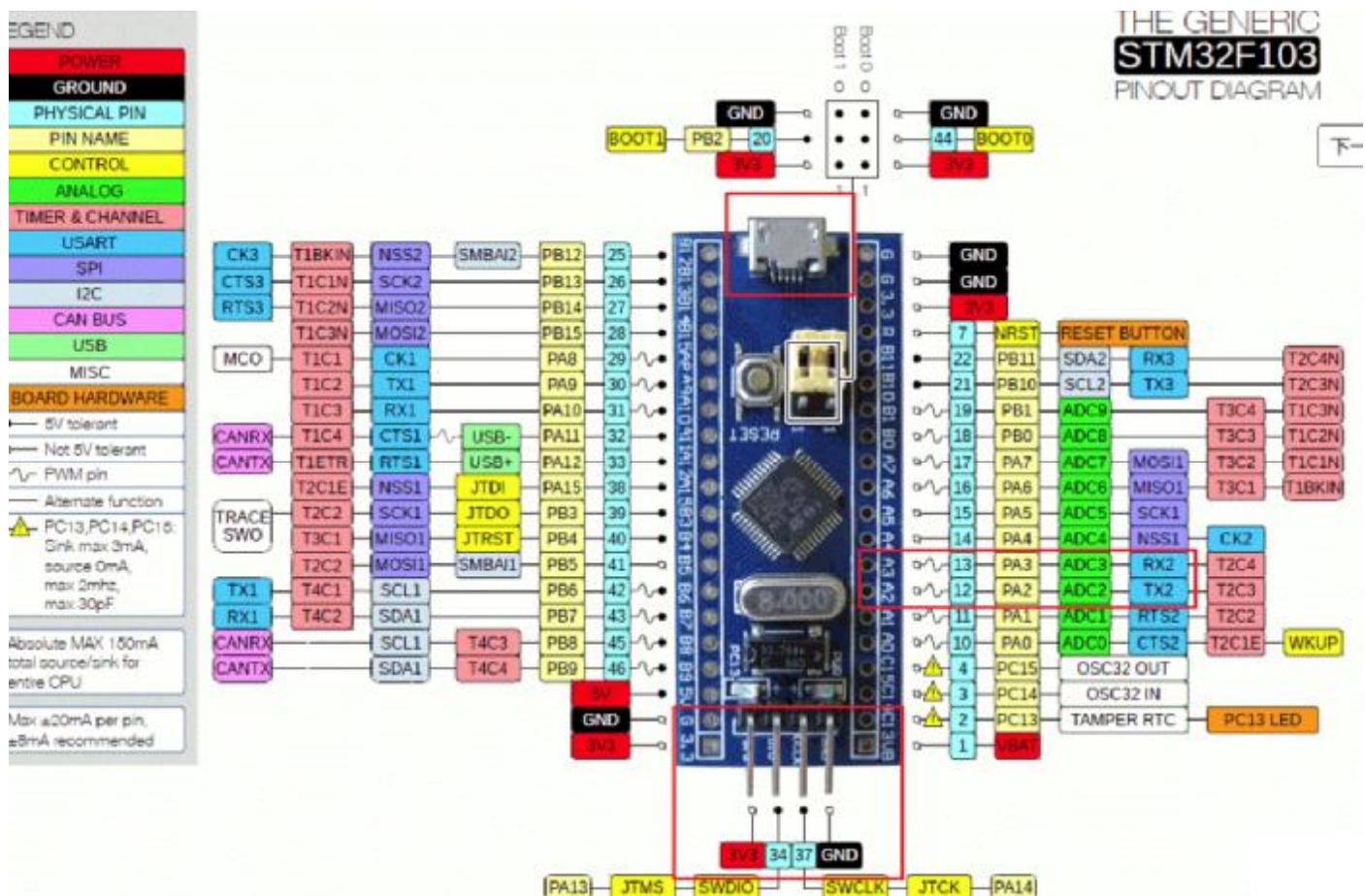
1. In1-2-3 分别接 STM32 板子的 A8,A9,A10，这是三个 PWM 引脚
2. en 接 STM32 板子的 A11，这是使能引脚
3. GND 接 GND



STM32 与 ST-Link 接线

按 STM32 和 ST-link 的丝印接即可

STM32 与 USB 转串口模块接线



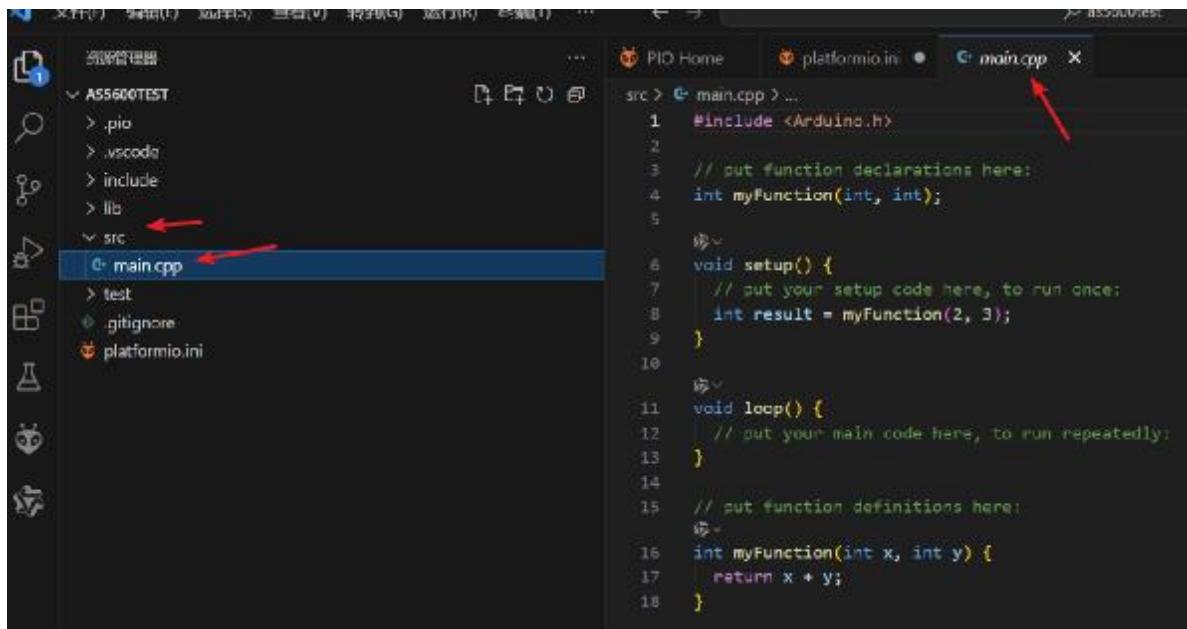
USB 供电， microUSB 接口。

到此，线已接完。

四、 编程

1. 开始编程

打开默认是这样的



The screenshot shows the VS Code interface with the following details:

- Project Explorer (Left):** Shows the project structure for "A55600TEST". It includes a ".pio" folder, a ".vscode" folder, an "include" folder, a "lib" folder, and a "src" folder containing a "main.cpp" file.
- Code Editor (Right):** Displays the "main.cpp" file content. The code is a simple Arduino-style C++ program with comments and function definitions.
- Status Bar:** Shows the path "PIO Home > platformio.ini > main.cpp" and a red arrow pointing to the file name.

2. 按 arduino 格式 C++语言，写入以下代码，代码就不做注释了，可以直接复制粘贴到 AI，如 DEEPSEEK 上，让它帮忙逐行解析。

```
3. #include <SimpleFOC.h>
4.
5. #define PIN_PWM_U PA8
6. #define PIN_PWM_V PA9
7. #define PIN_PWM_W PA10
8. #define PIN_ENABLE PA11
9.
10. BLDCDriver3PWM driver = BLDCDriver3PWM(PIN_PWM_U, PIN_PWM_V, PIN_PWM_W,
PIN_ENABLE);
11. BLDCMotor motor = BLDCMotor(7); // 极对数为 7 的无刷电机
12. void setup() {
13.     // 1. 先把 EN 拉到“失能”
14.     pinMode(PIN_ENABLE, OUTPUT);
15.     digitalWrite(PIN_ENABLE, LOW);
16.     // 2. 驱动器参数
17.     driver.voltage_power_supply = 12; // 实际 12V 电源
18.     driver.voltage_limit = 4.0; // 限制到 4V，减少电流与发热
19.     driver.pwm_frequency = 20000; // 20kHz
20.     driver.enable_active_high = true; // PA11 拉高时才使能 MS8313
21.     driver.init(); // 初始化驱动器
22.     // 4. 连接电机、初始化
23.     motor.linkDriver(&driver);
24.     motor.voltage_limit = 2.0; // 电机端相电压限幅到 2V
25.     motor.controller = MotionControlType::velocity_openloop;
26.     motor.init();
27.     // 5. 打开 EN
28.     digitalWrite(PIN_ENABLE, HIGH);
29. }
30.
```

```
31. void loop() {  
32.   motor.move(15); // 15 rad/s 开环  
33. }
```

The screenshot shows the PlatformIO IDE interface with the following details:

- File Menu:** 文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 帮助(H) ...
- Project Manager:** STM32FOCTest.CODE.WORKSPACES-VELOCITY (179)
- STM32FOCTest CODE WORKSPACE-VELOCITY (179) -> STM32FOCTest - velocity-OPEN**
- Files:** history, pio, vscode, include, lib, .gitignore, platformio.ini, STM32-F746Control.docx, STM32FOCTest.CODE.WORKSPACE-VELOCITY.CODE-WORKSPACE
- Code Editor:** main.cpp (selected tab)
- Code Content (main.cpp):**

```
#include "SimpleFOC.h"

#define PIN_PWM_U PA8
#define PIN_PWM_V PA9
#define PIN_PWM_W PA10
#define PIN_ENABLE PA13

BLDCDriverPMW driver = BLDCDriverPMW(PIN_PWM_U, PIN_PWM_V, PIN_PWM_W, PIN_ENABLE);
BLDCMotor motor = BLDCMotor(7); // 电机数为 7 的无刷电机

void setup() {
    // 1. 处理 EN 电源使能
    pinMode(PIN_ENABLE, OUTPUT);
    digitalWrite(PIN_ENABLE, LOW);
    // 2. 配置参数
    driver.voltage_power_supply = 12; // 电源 12V 电源
    driver.voltage_limit = 4.0; // 限制到 4V，减少电流发热
    driver.pwm_frequency = 20000; // 20kHz
    driver.enable.active_high = true; // PA11 低电平才启动 MG0313
    driver.init(); // 初始化驱动器
    // 3. 选择电机。初始化
    motor.linkDriver(&driver);
    motor.voltage_limit = 2.0; // 电机端相电压限制到 2V
    motor.controller = MotionControlType::velocity_openloop;
    motor.init();
    // 4. 11引脚 EN
    digitalWrite(PIN_ENABLE, HIGH);
}

void loop() {
    motor.move(15); // 15 rad/s 开环
}
```

- Terminal:** ** Verify Started **
*** Verified OK ***
*** Resetting Target ***
shutdown command invoked
- Status Bar:** SUCCESS Took 9.96 seconds
- Bottom Bar:** 按端口号任务重用，按按键重用。

就这么多了！

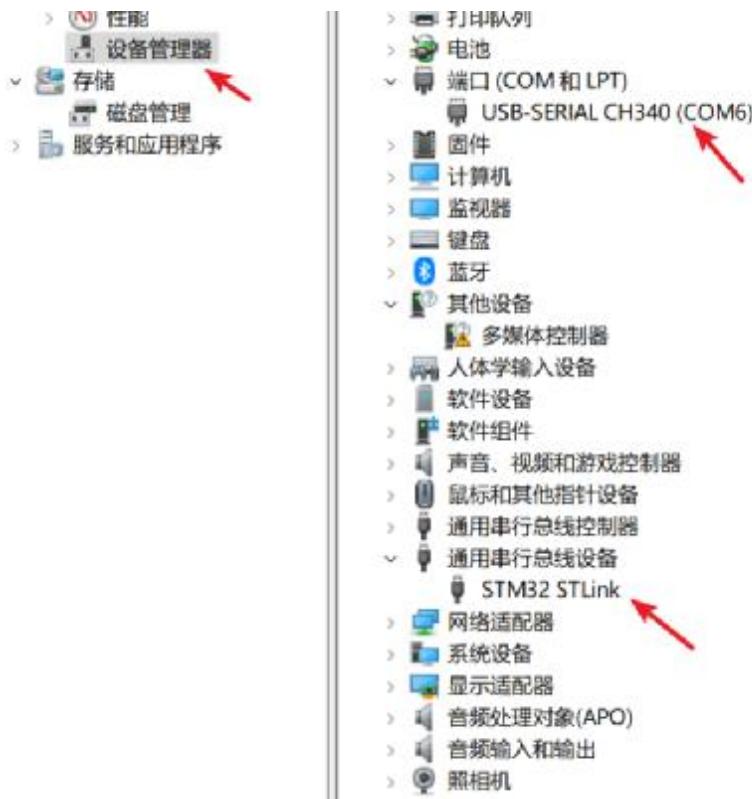
五、 实现

1. 编译烧录

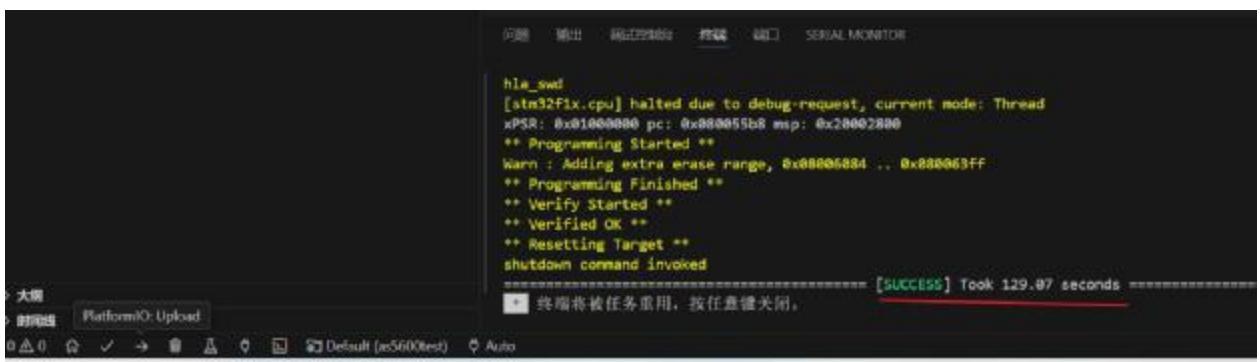


2. 插上 USB，开始烧录

确定驱动安装正确，设备管理器中，可以看到这两个。



3. 烧录成功



接上电源测试，电机就开始转动了。

不过这样电机会有**异响，会异响，会异响！会发烫，会发烫，会发烫**，这里只是演示一下，验证硬件是否**有故障，接线是否正确**。

这不是硬件故障的问题，是控制的缘故。

在 STM32 板上出现「明显噪声」的问题，通常并非 SimpleFOC 算法本身所致，而是底层 PWM 输出和驱动芯片（DRV8313）之间的时序、死区（Dead-time）或使能（ENABLE）逻辑没有完全匹配导致的。以下几点是最常见的原因，并附上相应的修改建议：

1. STM32 PWM 定时器的死区（Dead-time）没有配置

- 现象：DRV8313 是三相半桥驱动芯片，需要给每一高高、低低 MOSFET 之间留一定死区时间，否则在高低侧 MOSFET 切换时会出现短路风险或不完整切换，从而产生电流抖动、声音。
- ESP32 上通常已经由库帮你把死区算好，而在 STM32 上，如果你只是直接把 PA8/PA9/PA10 三脚接到 TIM1_CH1/CH2/CH3，上面代码里调用 `driver.setFrequency(30000); driver.init();` 并不会主动地在 TIM1 的 BDTR 寄存器里写入死区（deadtime）。
- 解决方式：在调用 `driver.init();` 之后，显式给 TIM1 加上死区。比如如果你希望死区为 1.0 微秒，可以这样写（以 STM32F1xx 为例，F4/F7 的寄存器名略有变化）：

```

// ...
// ... 代码示例：给 TIM1_BDTR 与死区值（1us 无关） ...
// 1. 死区设置时钟是 72MHz，而 TIM1 的时钟是时钟一稳定后 72MHz (APB2v2)

```

开环的解决方案，暂时不会，有知道的朋友，欢迎告诉我，谢谢！

接下来试用闭环，应该可以解决这个问题。

到此，测试结束！本文档主要针对 M 创动工坊淘宝店提供硬件, mcdgf.taobao.com