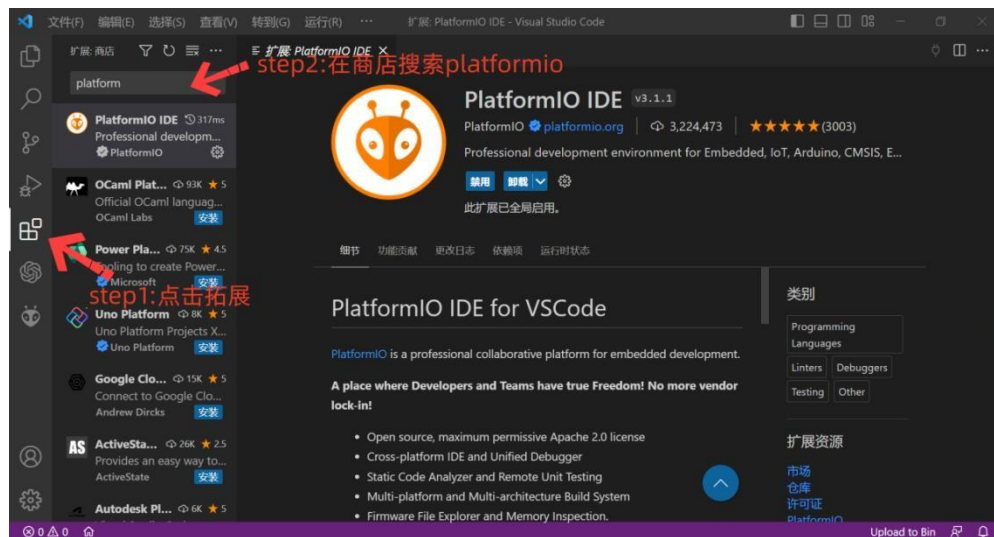


一、环境配置

第一步：下载 Visual Studio Code：



第二步：打开 Visual Studio Code，在其扩展商店中搜索并下载 PlatformIO IDE



(图中软件即为 PlatformIO IDE)

点击下载后他会下载其他需要使用的拓展，有时会缺少 MinGW，无法新建文件夹，我们需要下载一个 MinGW-w64。

第三步：下载 MinGW-w64

查看资料包“8 工具”里的“mingw64”，将其解压到任意位置

(文件下载后解压，解压后安装的路径需要全英文，不能有中文)

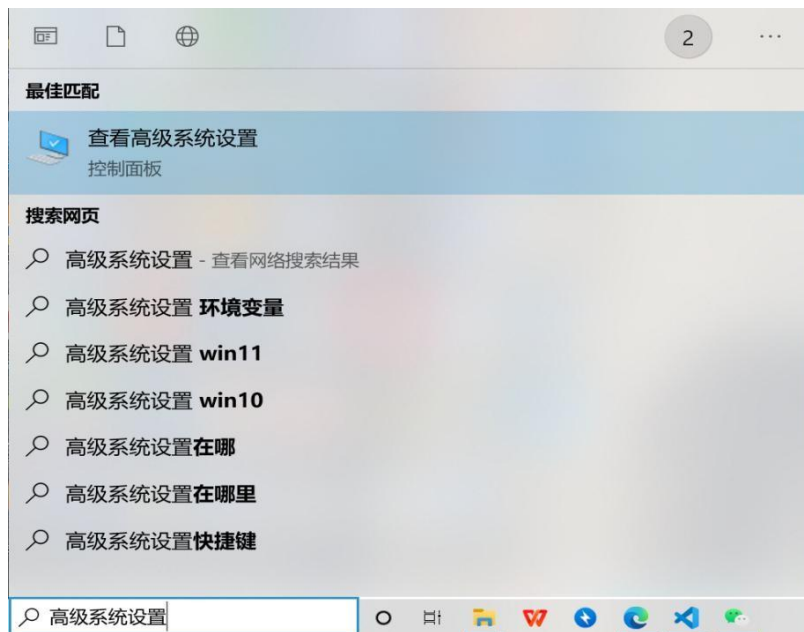
3.1 打开解压后的文件→打开 bin 文件→复制 bin 文件的地址
(例：D:\ming\mingw64\bin)



3.2 配置环境：

步骤：电脑搜索打开“查看高级系统设置”→环境变量→系统变量→Path

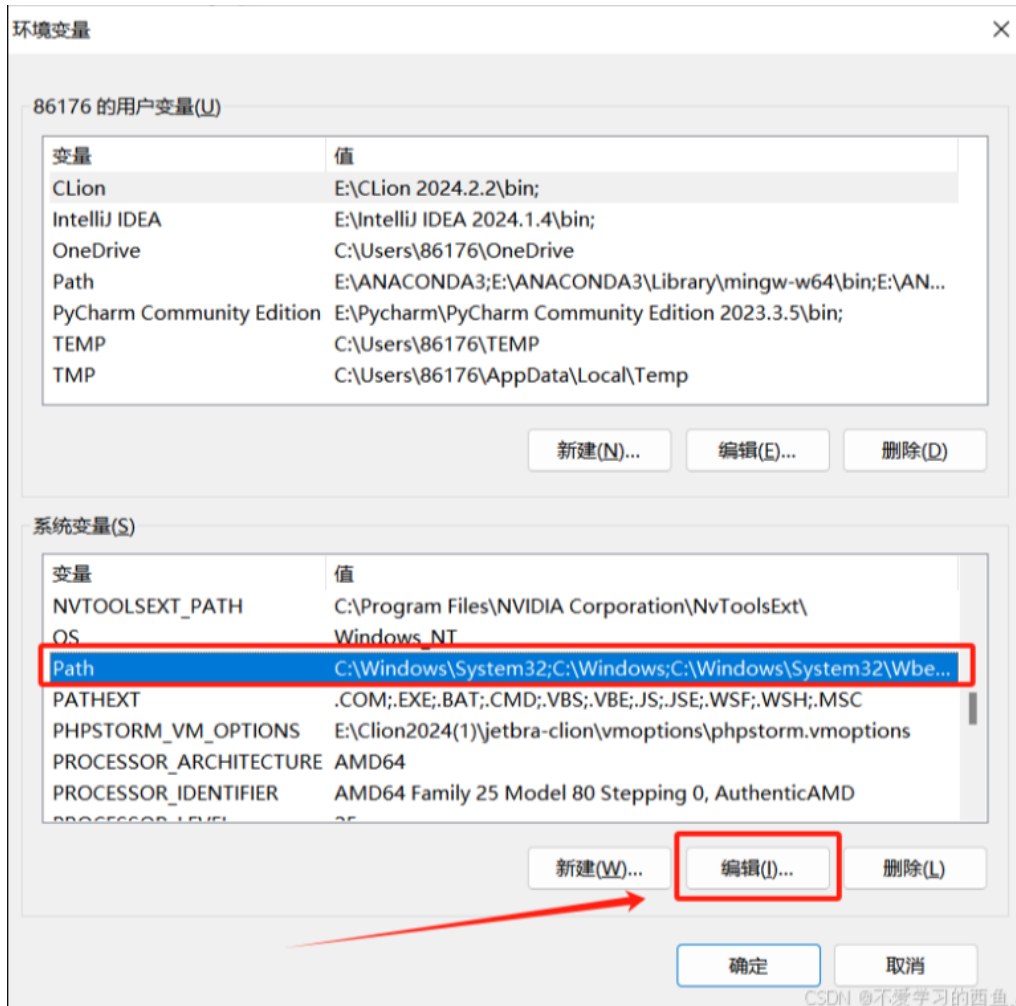
→新建→粘贴上一小步的 bin 文件的地址，点击确定。



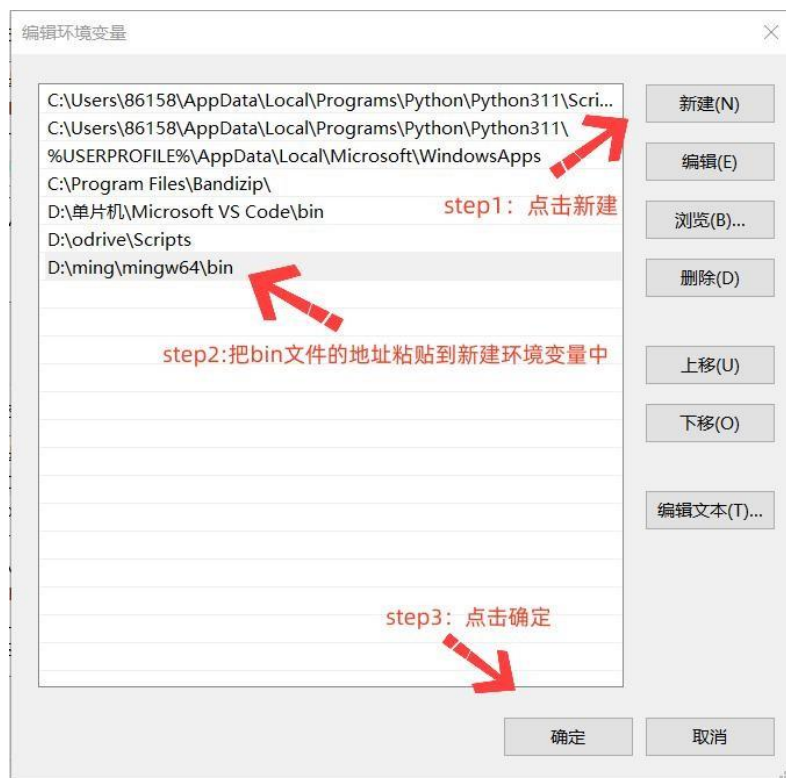
(1: 电脑搜索打开“查看高级系统设置”)



(2: 点击环境变量)

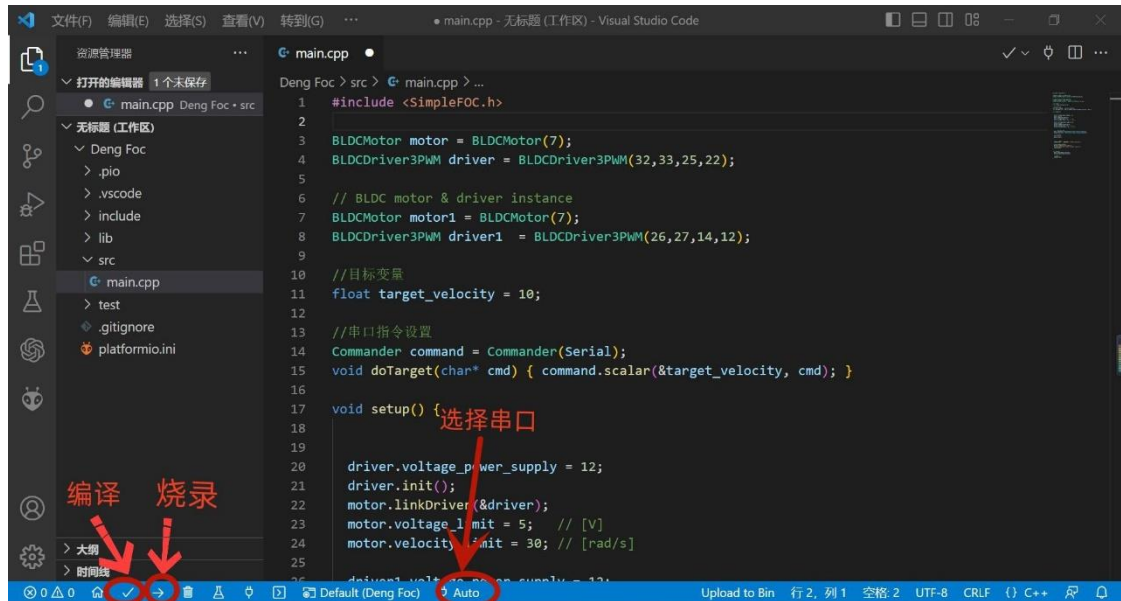


(3: 双击打开 Path，注意是要选择系统变量)



(4: 新建环境变量)

第四步：编译及烧录按键说明：



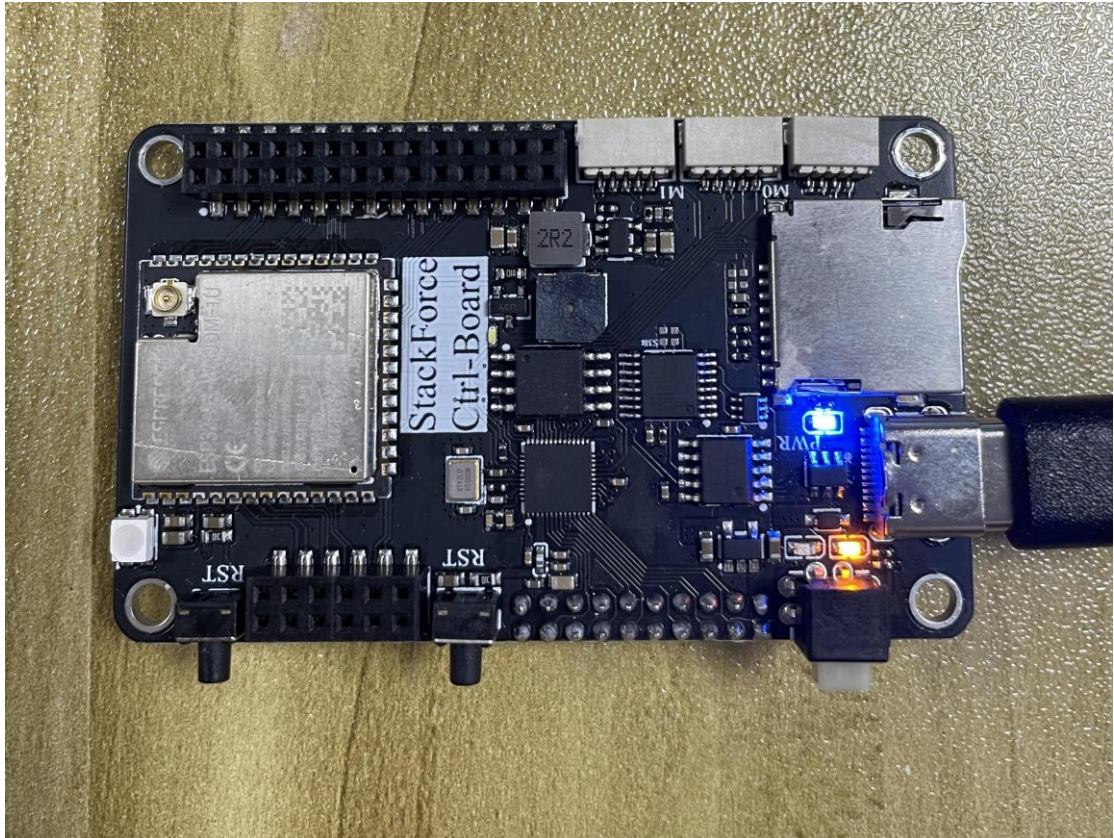
(√：编译程序 →：烧录程序到硬件 Auto：选择串口)

下面是烧录按键说明，先不烧录程序

电脑与硬件连接、点击 Auto (可忽略)，会自动检测并推荐串口。选择串口后点击编译 (可忽略)、烧录，即可将程序烧录至硬件。

二、S1 烧录和调试

1.连接 USB，USB 有缝隙一边朝下，无缝一边朝上,松开白色按键，切换至 S1 芯片（黄灯亮）



备注：为什么要分上下

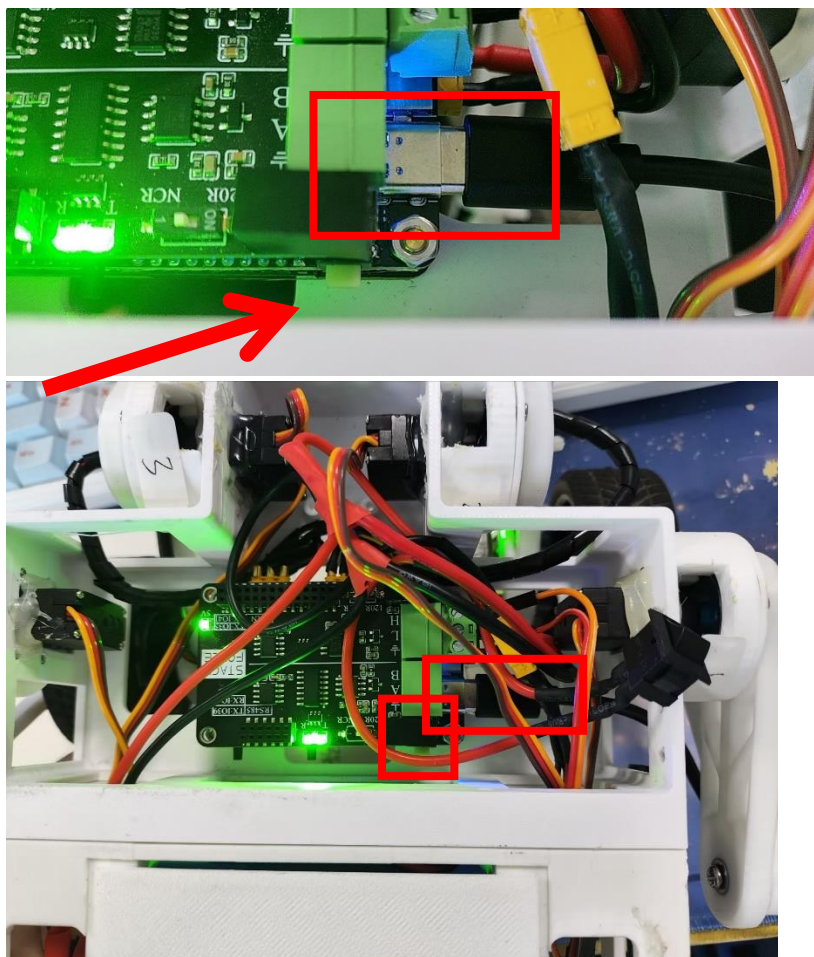
因为我们的板子有两个芯片，S1 芯片负责电机程序的运行，S3 负责舵机控制程序的运行，typec 线有上下个两排排针分别通信，我们的主控板设计两个芯片分别占用 typec 的一排用来烧录程序，通过白色按键来在硬件上控制电脑要把程序烧录到哪个芯片

2.前驱动烧录 S1 程序

方法一：前驱动和后驱动的主控板我们都已经烧录电机控制程序了，完成下方接线后就跳到下方“4.Vofa 串口助手下载与使用”

将 USB 线插到前驱动的主控板上，注意 usb 无缝隙一面朝上

松开白色按钮，具体如下图所示

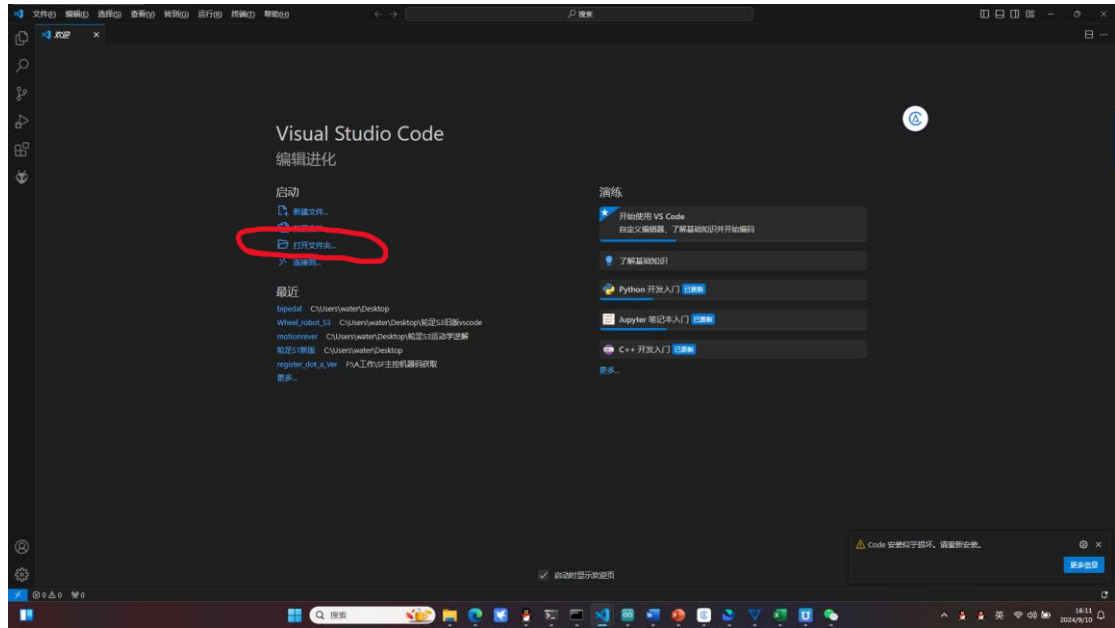


方法二：按照下方烧录 S1 电机控制程序（不推荐）

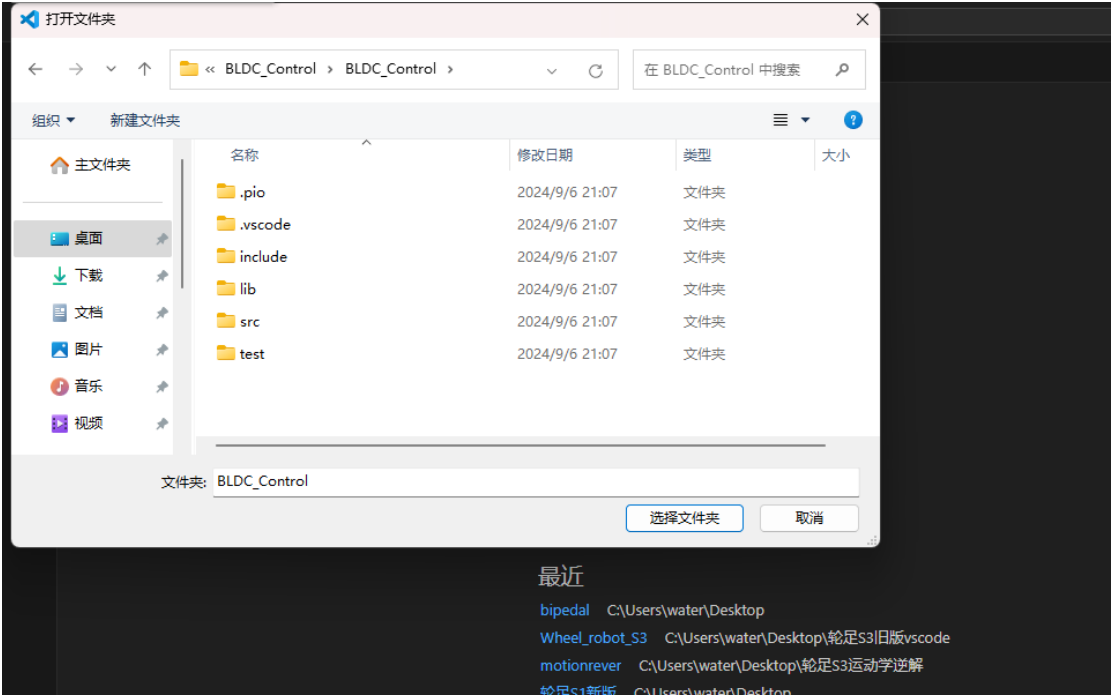
在 vscode 打开工程 BLDC_Control 文件夹（每个程序都要按照下方说明操作）

程序在资料包里的程序文件夹中

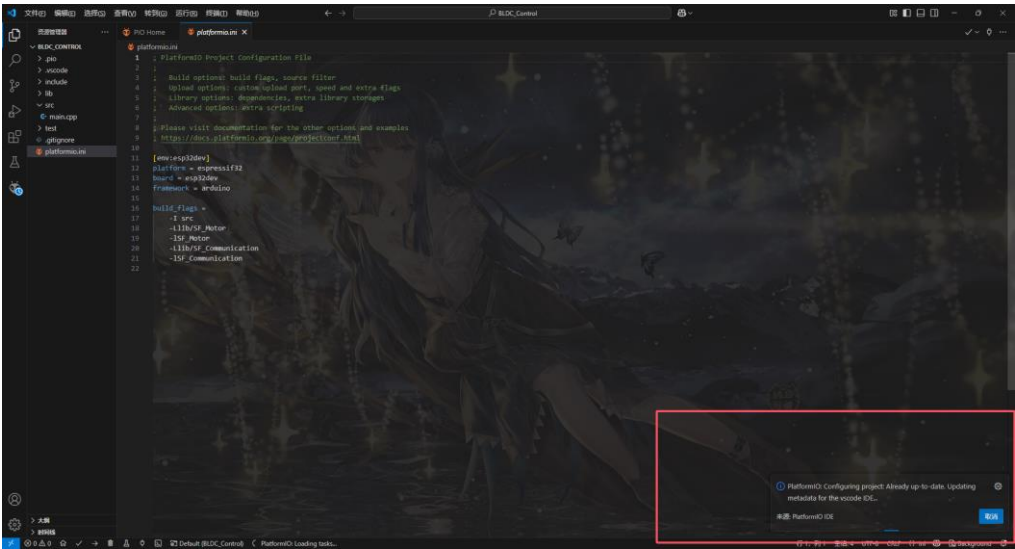
这样操作可以让 platformio 自动安装库，所以不能直接将项目文件拖进 vscode
打开新的 vscode 软件，打开文件夹（或者是在文件打开文件夹）



找到 S1 程序保存的位置，点击选择文件夹（不能有中文路径，且一定要打开到当前位置）



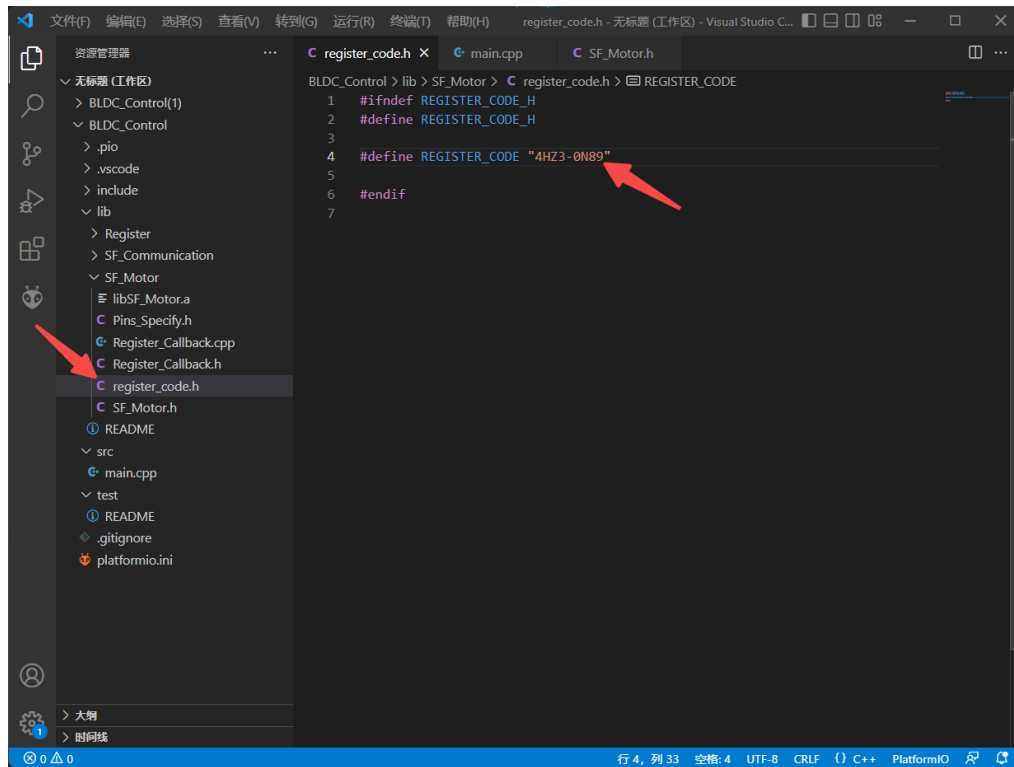
等待右下方项目加载完成



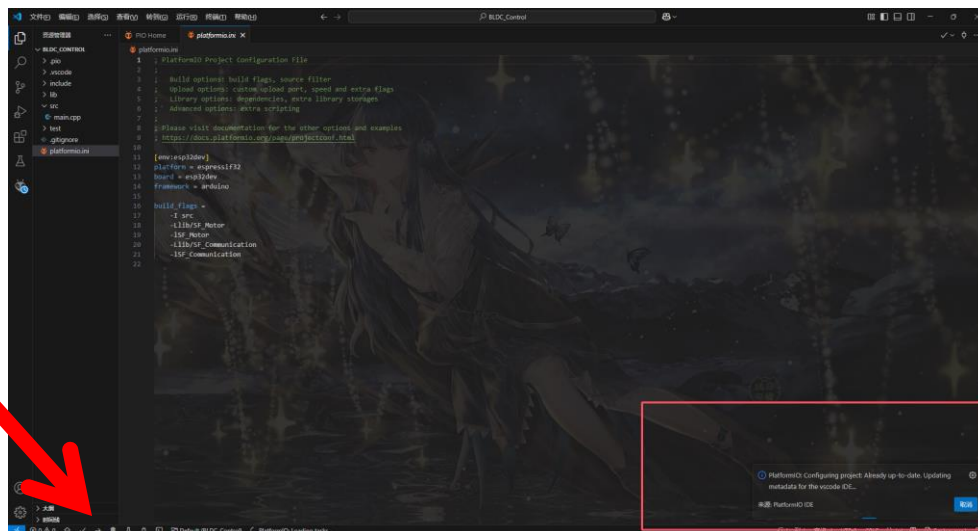
修改注册码，将前驱动的主控板的注册码写到这个位置，

如果和后驱动主控板弄混了，可以随机选一个注册码继续往后操作，如果后面操作出现注册码错误就选择另一个注册码

每个板子的注册码都是不一样的，您的注册码我们将他贴在了主控包上的标签纸，
将其输入到这个位置。



点击程序下方的箭头烧录程序到前驱动的 S1 芯片上



3.烧录过程可能会遇到的问题及解决办法

1、成功烧录效果

```
Writing at 0x00037c8d... (46 %)
Writing at 0x0003d5ab... (53 %)
Writing at 0x00042b8d... (61 %)
Writing at 0x00047e45... (69 %)
Writing at 0x0004d24f... (76 %)
Writing at 0x00055215... (84 %)
Writing at 0x0005c42d... (92 %)
Writing at 0x00063e1c... (100 %)
Wrote 346528 bytes (198507 compressed) at 0x00010000 in 4.6 seconds (effective 600.8 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
===== [SUCCESS] Took 14.38 seconds =====
* 终端将被任务重用，按任意键关闭。
```

2、这个主控芯片是 S3，与 S1 程序不匹配

```
CURRENT: upload_protocol = esptool
Looking for upload port...
Auto-detected: COM18
Uploading .pio\build\esp32dev\firmware.bin
esptool.py v4.5.1
Serial port COM18
Connecting...

A fatal error occurred: This chip is ESP32-S3 not ESP32. Wrong --chip argument?
*** [upload] Error 2
===== [FAILED] Took 5.24 seconds =====

* 终端进程"C:\Users\water\.platformio\penv\Scripts\platformio.exe 'run', '--target', 'upload'"已终止，退出代码：1。
* 终端将被任务重用，按任意键关闭。
```

A fatal error occurred: This chip is ESP32-S3 not ESP32. Wrong --chip argument?
*** [upload] Error 2

检查 USB 是否插反，要求无缝朝上；

检查主控白色按键是否有松开，松开时按键旁边亮黄灯，主控处于 S1 芯片烧录状态

3、串口被占用了

```
jlink, minimodule, olimex-arm-usb-ocd, olimex-arm-usb-ocd-h, olimex-arm-u
sb-tiny-h, olimex-jtag-tiny, tumpa
CURRENT: upload_protocol = esptool
Looking for upload port...
Auto-detected: COM15
Uploading .pio\build\esp32dev\firmware.bin
esptool.py v4.5.1
Serial port COM15

A fatal error occurred: Could not open COM15, the port doesn't exist
*** [upload] Error 2
===== [FAILED] Took 4.50 seconds =====

* 终端进程"C:\Users\water\.platformio\penv\Scripts\platformio.exe 'run',
'--target', 'upload'"已终止，退出代码：1。
* 终端将被任务重用，按任意键关闭。
```

A fatal error occurred: Could not open COM15, the port doesn't exist

*** [upload] Error 2

检查有没有其他软件占用了串口，

检查 vofa 的串口监视器是否关闭

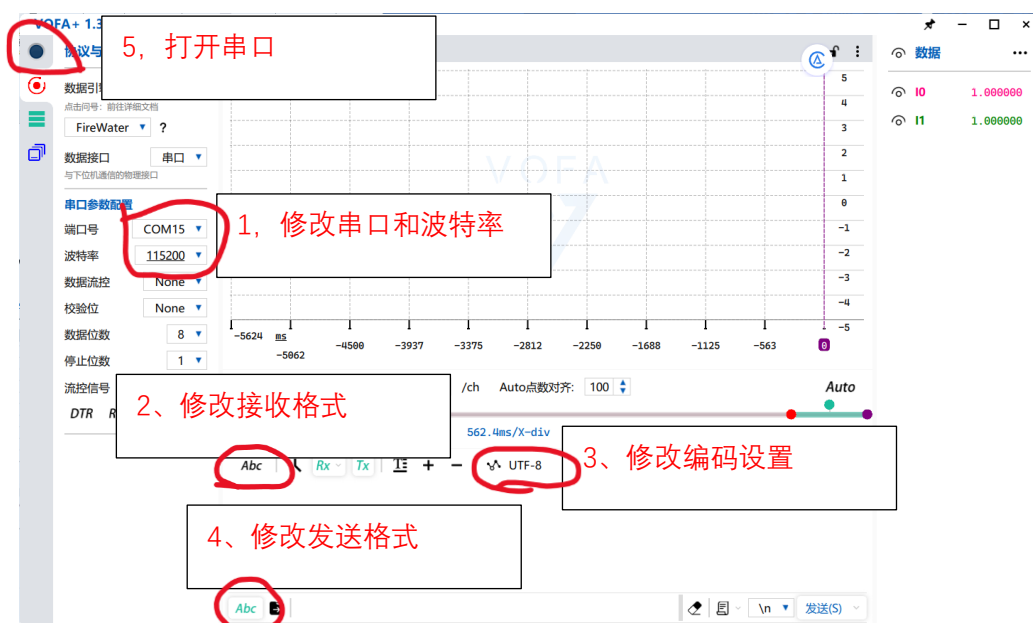
检查其他串口助手是否关闭串口

4.Vofa 串口助手下载与使用教程

Vofa 使用教程：按照下方图片顺序设置 vofa 格式，除了串口号和波特率，其他都设置成图片中一样的符号和颜色

烧录后，接上 usb，打开串口助手，波特率为 115200，端口号在插上 usb 后选择有 CH340 字样的端口号，如果没有就是 CH340 驱动没有安装，可以查看工具包里的工具查看教程安装 ch340 驱动；

Vofa 串口助手下载网址：[下载中心](#) | [VOFA-Plus 上位机](#)



5.S1 电机控制程序调试，极对数校准

在完成上述 vofa 连接后；用手扶着机器人，轮子离开地面，**按一下 S1 复位键（或者点两次 vofa 上的 RTS 按键，RTS 亮灭之后开始复位）**，此时 vofa 上显示有极对数信息，等待轮子自检转动完成，如极对数被辨别后是 7，则表示校准成功，则如下图所示，**如果极对数不是 7 请看下方**

若极对数为 inf 或其它英文字符串，则请 1 检查是否打开了电源，2 检查线路是否接错，3 检查小电流板是否接上 12V 电源 3 检查磁铁安装

如果极对数为 6 或者 8 或者是非 7 数值，可能是车轮安装过紧，（可查看轮足安装文档搜索“电机轴承安装”，重新安装这个位置）

或者是轮子与地面有摩擦（每次上电或 S1 复位都要让机器人离地或者轮子不要碰到其他地方，自检才能正常运行）

可重新调试再 S1 复位直到极对数显示为 7 为止。



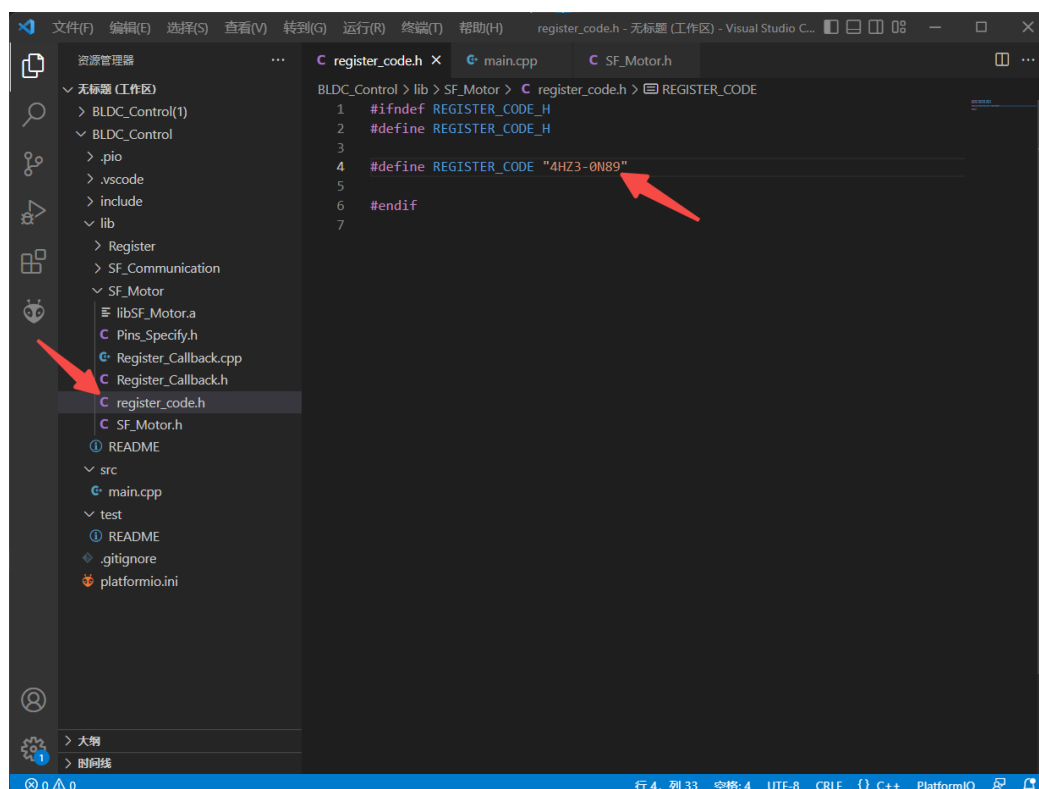
前后驱动的极对数都调完后就可以跳到“三、后驱动舵机偏置值获取”

6、后驱动烧录 S1 电机控制程序，将 USB 插到后驱动的主控板上，注意 USB 有缝一侧朝下，白色开关松开（箭头指向处）

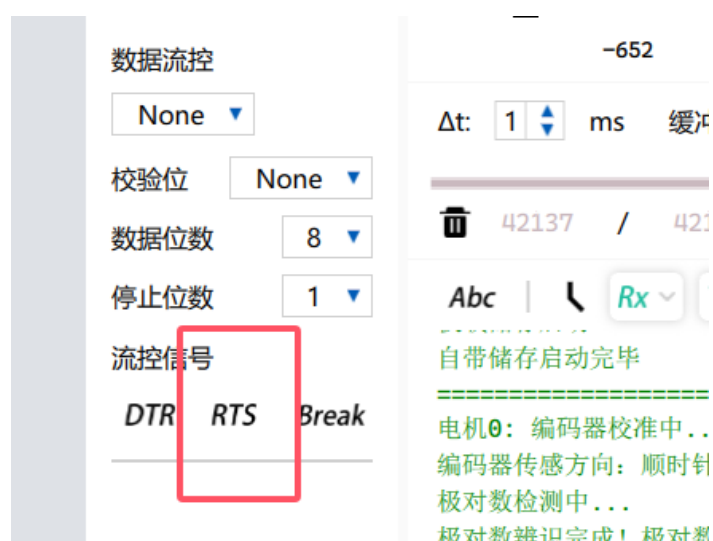
后驱动板的 S1 程序已经烧录了，跳到上方 4、5 点调极对数就可以



7、打开刚刚的 BLDC 程序，将后驱动主控板的注册码写到下图位置,然后烧录就可以

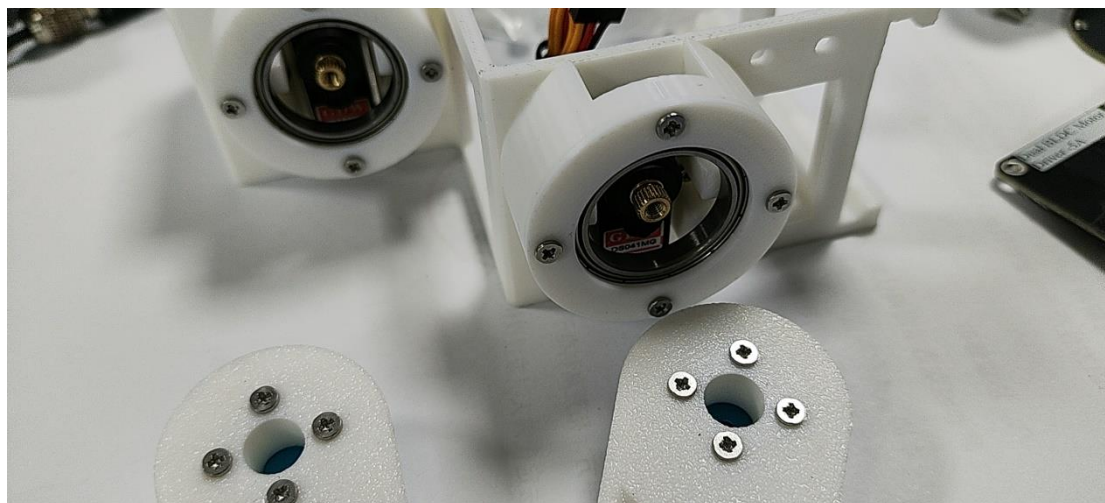


8、剩下的部分参考上面 3、4、5 这三个步骤，由于 S1 复位键按不到，操作步骤 5 的时候可以双击 vofa 中的 RTS 通过软件复位

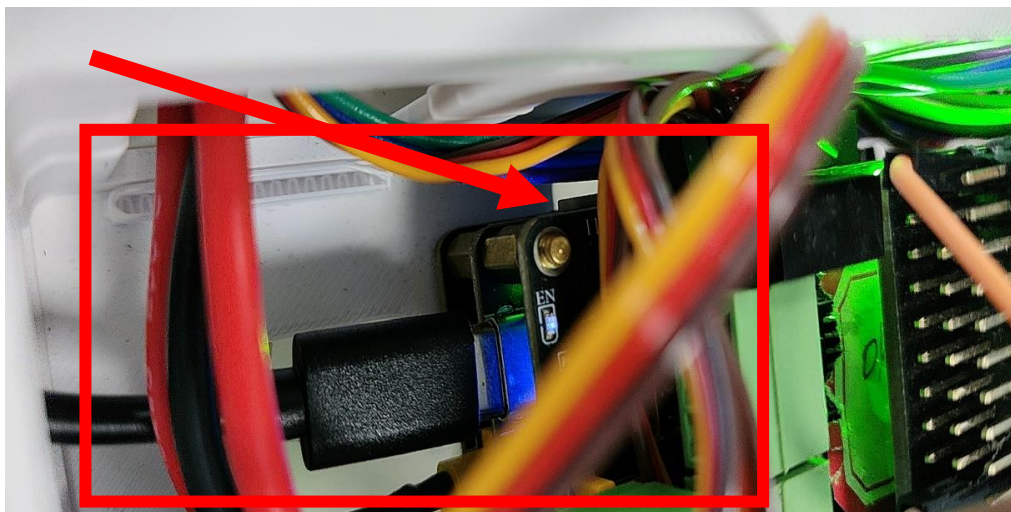


三、后驱动 S3 舵机偏置值获取

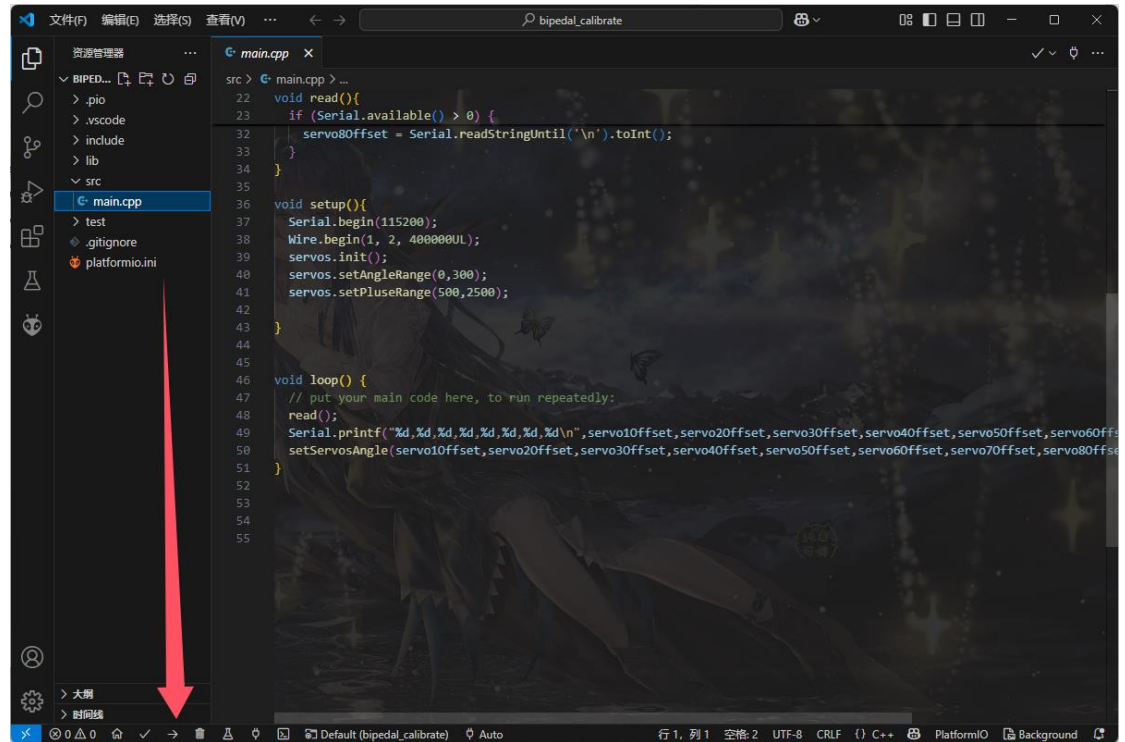
在烧录程序前一定要将大腿拆下



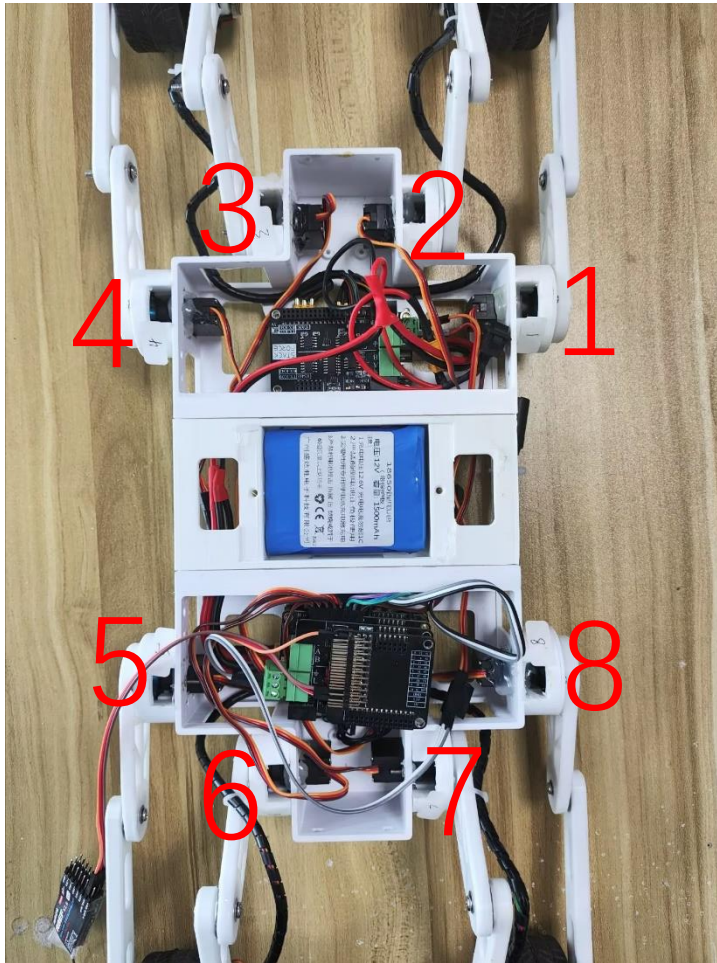
1、将 USB 插在后驱动的主控板上，并按下白色按键



2.在 vsocce 打开 bipedal_calibrate 文件夹(偏置值获取程序),
直接烧录程序,



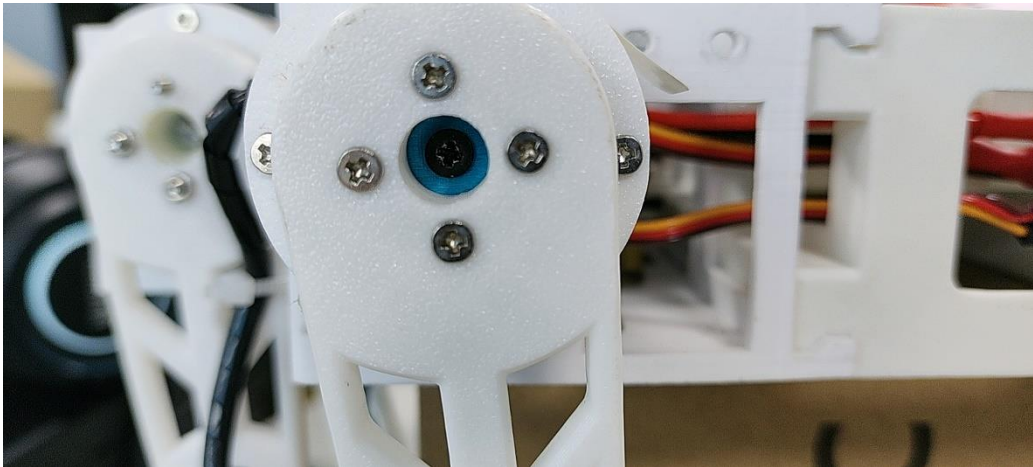
3、打开 vofa, 选择端口号, 波特率设置 115200, 打开串口, 可以看到串口信息为 0,0,0,0,0,0,0,0;分别代表 1,2,3,4,5,6,7,8 号舵机的偏置值



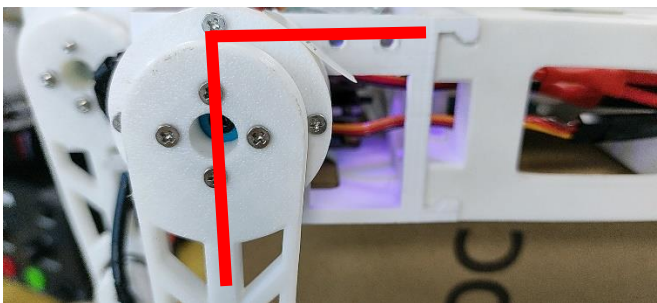
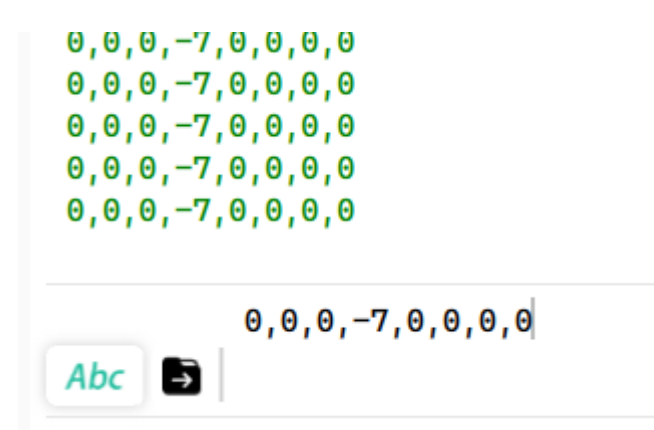
4、电池上电，等待舵机完成转动后安装腿部尽量垂直于机器人机身，并拿出舵机盒子的黑色螺丝将大腿固定在机身上



4、在串口输入 1,2,3,4,5,6,7,8 等指令控制舵机转动直到腿部完全垂直水平面
方向解释：腿部面向自己，顺时针为负，逆时针为正，下面以 4 号舵机的为例
下图是安装时 4 号大腿位置，稍微往右偏了一点



在 vofa 中输入 0,0,0,-7,0,0,0,0 然后发送，让舵机顺时针旋转 7 度，使大腿垂直机身

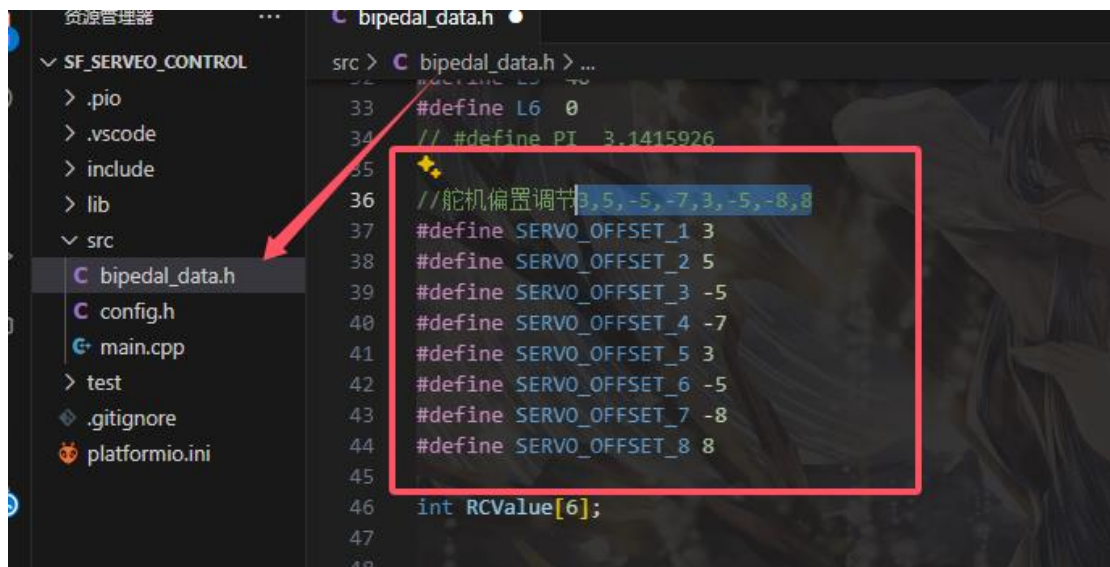


其他大腿同理，最终得到偏置值 3,5,-5,-7,3,-5,-8,8
保存当前偏置值，要写到下面的 S3 舵机控制程序

四、后驱动舵机调试

1、修改偏置值

在 vscode 打开 SF_serveyo_control 文件夹（机器人运动控制程序），
在 bipedal_data.h 文件下修改偏置值 OFFSET，具体看如下，将上面程序获取到的偏置值输入到下面对应位置，然后将程序烧录到后驱动的 S3 芯片上（跟前面偏置值获取程序一样的烧录方法）



```
src > C bipedal_data.h > ...
33 #define L6 0
34 // #define PI 3.1415926
35
36 //舵机偏置调节 3,5,-5,-7,3,-5,-8,8
37 #define SERVO_OFFSET_1 3
38 #define SERVO_OFFSET_2 5
39 #define SERVO_OFFSET_3 -5
40 #define SERVO_OFFSET_4 -7
41 #define SERVO_OFFSET_5 3
42 #define SERVO_OFFSET_6 -5
43 #define SERVO_OFFSET_7 -8
44 #define SERVO_OFFSET_8 8
45
46 int RCValue[6];
47
48
```

2、舵机调试



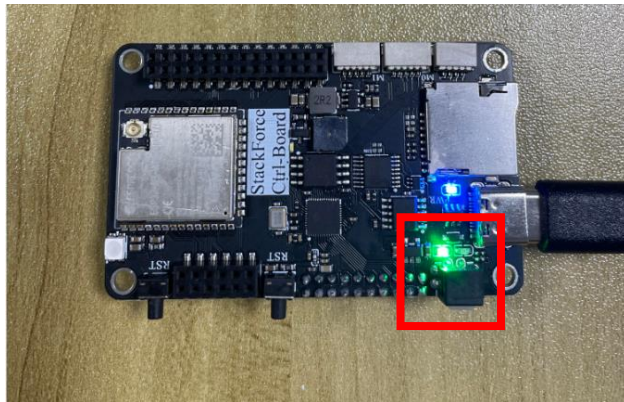
烧录完程序后拔开 usb，然后打开“0 整机操作说明”按照里面的“**遥控器对频**”对频遥控器，将遥控器的挡位打到和图中所示的一样，将机器人拿起等待 10 秒左右，前后左右滑动 B 可以看到机器人左右翻滚和腿高变化，具体如下

A 打到上面（无陀螺仪车子模式），C 打到最下（车子模式），E 往上打开启遥控器电源

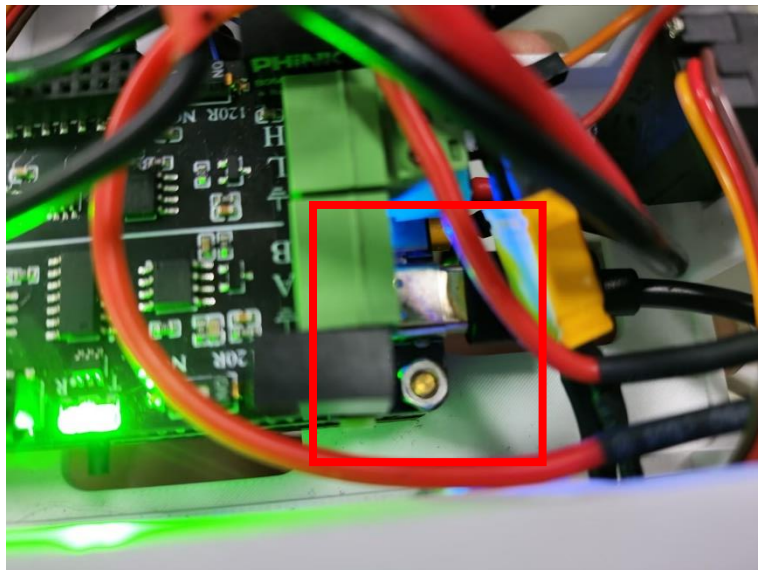
没问题的话就继续往下操作

五、前驱动 S3 程序烧录

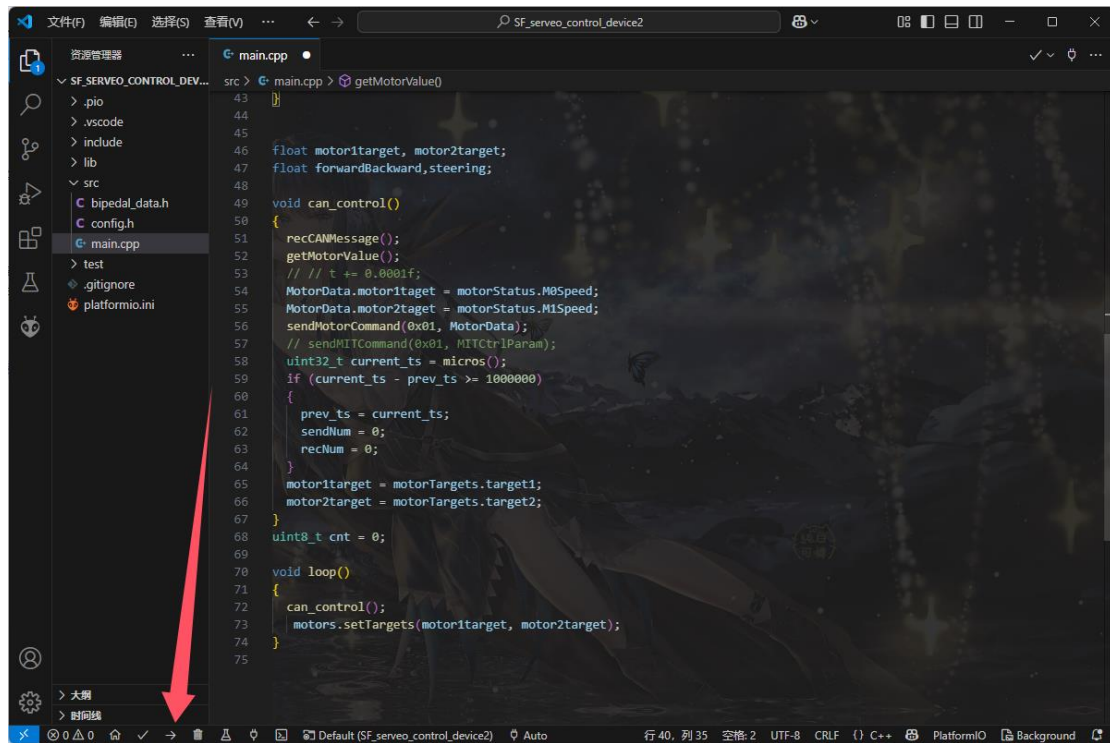
1、烧录完 S1 后，切换至 S3 芯片，并烧录 S3 程序（USB 无縫隙朝上，按下图所示白色按钮，灯为绿色则切换至 S3



将 usb 插到前驱动的主控板上，注意 USB 不要插反，同时按下白色按钮



2、打开 SF_servec_control_device2 程序，注意按照上面打开 bldc 电机控制程序的方式打开这个程序，然后直接烧录就可以



```
43  
44  
45  
46 float motor1target, motor2target;  
47 float forwardBackward, steering;  
48  
49 void can_control()  
50 {  
51     recCANMessage();  
52     getMotorValue();  
53     // t += 0.0001f;  
54     MotorData.motor1target = motorStatus.M0Speed;  
55     MotorData.motor2target = motorStatus.M1Speed;  
56     sendMotorCommand(0x01, MotorData);  
57     // sendMIICommand(0x01, MIICtrlParam);  
58     uint32_t current_ts = micros();  
59     if (current_ts - prev_ts >= 1000000)  
60     {  
61         prev_ts = current_ts;  
62         sendNum = 0;  
63         recNum = 0;  
64     }  
65     motor1target = motorTargets.target1;  
66     motor2target = motorTargets.target2;  
67 }  
68 uint8_t cnt = 0;  
69  
70 void loop()  
71 {  
72     can_control();  
73     motors.setTargets(motor1target, motor2target);  
74 }  
75
```

六、后驱动设置 dir

1、设置轮子 Dir

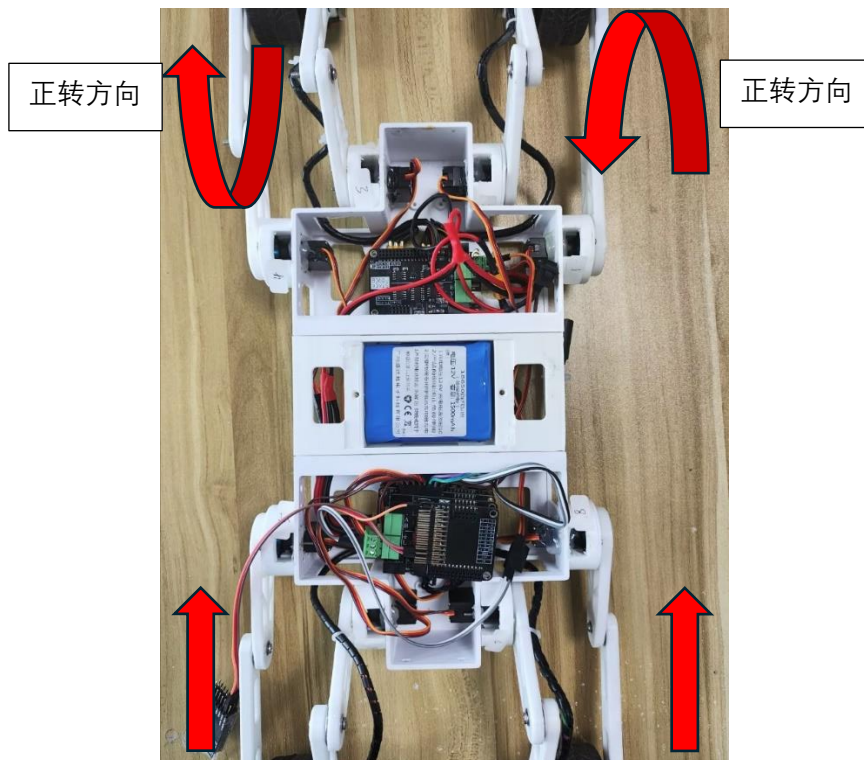
在 SF_serveo_control 程序中打开 main.cpp 文件往下滑一下找到 setRobotparam()函数，在这里设置轮子极性

```
int flat = 0; //模式切换标志位
//设置轮子方向dir
void setRobotparam(){
    //后面两个电机速度控制方向
    motorStatus.M0Dir = -1; //左后电机控制方向
    motorStatus.M1Dir = 1; //右后电机控制方向

    flat = 1; //电机转速反馈方向校准模式，将电机设置为固定转速
    // flat = 0; //电机取消校准模式
    //后面两个电机速度反馈方向
    motorStatus.M0SpdDir = 1; //左后电机反馈方向
    motorStatus.M1SpdDir = -1; //右后电机反馈方向

    //前面两个电机速度控制方向
    motorStatus.M3Dir = 1; //左前电机控制方向
    motorStatus.M4Dir = -1; //右前电机控制方向
}
```

按照上图上的红色框框，将 flat 设置为 1，下面的 flat=0 注释掉；然后烧录程序，此时后面两个电机以固定转速固定方向转动（上电后电机才会转动）
打开 vofa，波特率设置为 115200



观察左后腿的电机转动方向，电机向前转则速度为正，（看上图，板少的一侧是前）
如果 vofa 上打印的第二个数据 M0 速度为负，则需要将 motorStatus.M0SpdDir 取反

(取反的意思是原本是 1 写成 -1; 原本 -1 写成 1)

右后腿电机向后转则速度为负,

如果 vofa 上打印的第二个数据 M1 速度为负, 则不需要修改 motorStatus.M1SpdDir

```
int flat = 0; //模式切换标志位
//设置轮子方向dir
void setRobotparam(){
    //后面两个电机速度控制方向
    motorStatus.M0Dir = -1; //左后电机控制方向
    motorStatus.M1Dir = 1; //右后电机控制方向

    flat = 1; //电机转速反馈方向校准模式, 将电机设置为固定转速
    // flat = 0; //电机取消校准模式
    //后面两个电机速度反馈方向
    motorStatus.M0SpdDir = 1; //左后电机反馈方向
    motorStatus.M1SpdDir = -1; //右后电机反馈方向

    //前面两个电机速度控制方向
    motorStatus.M3Dir = 1; //左前电机控制方向
    motorStatus.M4Dir = -1; //右前电机控制方向
}
```

改完后将程序烧录, 观察电机转动方向与 vofa 中的数据方向是否一致

2、设置控制 dir

返回调试函数，设置 **flat=0**;取消电机反馈方向校准，然后烧录

```
int flat = 0; //模式切换标志位
//设置轮子方向dir
void setRobotparam(){
    //后面两个电机速度控制方向
    motorStatus.M0Dir = -1; //左后电机控制方向
    motorStatus.M1Dir = 1; //右后电机控制方向

    // flat = 1; //电机转速反馈方向校准模式，将电机设置为固定转速
    flat = 0; //电机取消反馈方向校准模式
    //后面两个电机速度反馈方向
    motorStatus.M0SpdDir = 1; //左后电机反馈方向
    motorStatus.M1SpdDir = -1; //右后电机反馈方向

    //前面两个电机速度控制方向
    motorStatus.M3Dir = 1; //左前电机控制方向
    motorStatus.M4Dir = -1; //右前电机控制方向
}
```

机器人上电，遥控器 A 打上，C 打下，D 往上推，这个时候机器人四个轮子都应该往前转，观察机器人轮子情况，（取反的意思是原本是 1 写成 -1；原本 -1 写成 1）

如果左后电机往后转，motorStatus.M0Dir 的值取反；

如果右后电机往前转，motorStatus.M1Dir 的值不取反；

如果左前电机往前转，motorStatus.M3Dir 的值不取反；

如果右前电机往前转，motorStatus.M4Dir 的值不取反；



到这里就已经完成四足机器人的调试了，
接下来看资料包“0 整机操作说明去操作机器人”