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

可以参考网站[下载安装 MinGW-w64 详细步骤(c/c++的编译器 gcc 的 windows 版，win10 真实可用）-CSDN 博客]下载，也可以看下方教程安装

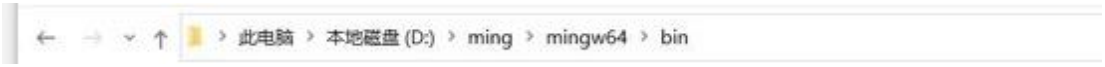
3.1 打开 MinGW 官网官方下载网站:

“ https://sourceforge.net/projects/mingw-w64/files/mingw-w64/ming w-w64-release/”

3.2 往下翻，下载“x86_64-posix-sjlj”版本

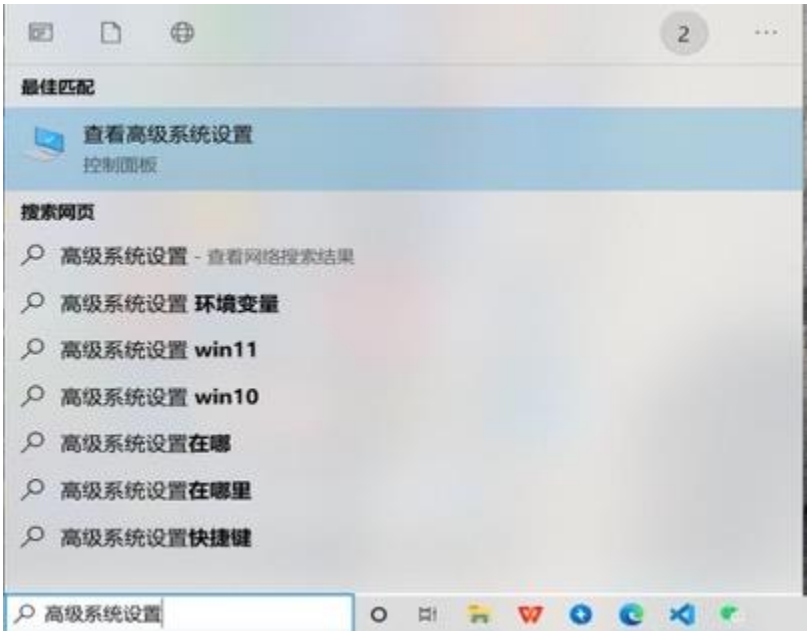
（文件下载后解压，解压后安装的路径需要全英文，不能有中文）

3.3　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）



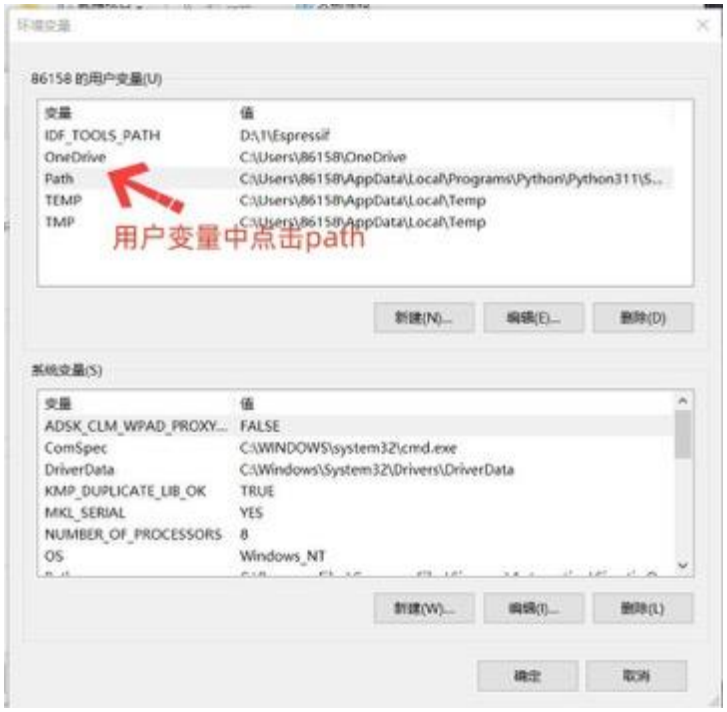
3.4　配置环境：

步骤：电脑搜索打开“查看高级系统设置”→环境变量→用户变量→Path→新建→粘贴上一小步的 bin 文件的地址，点击确定。

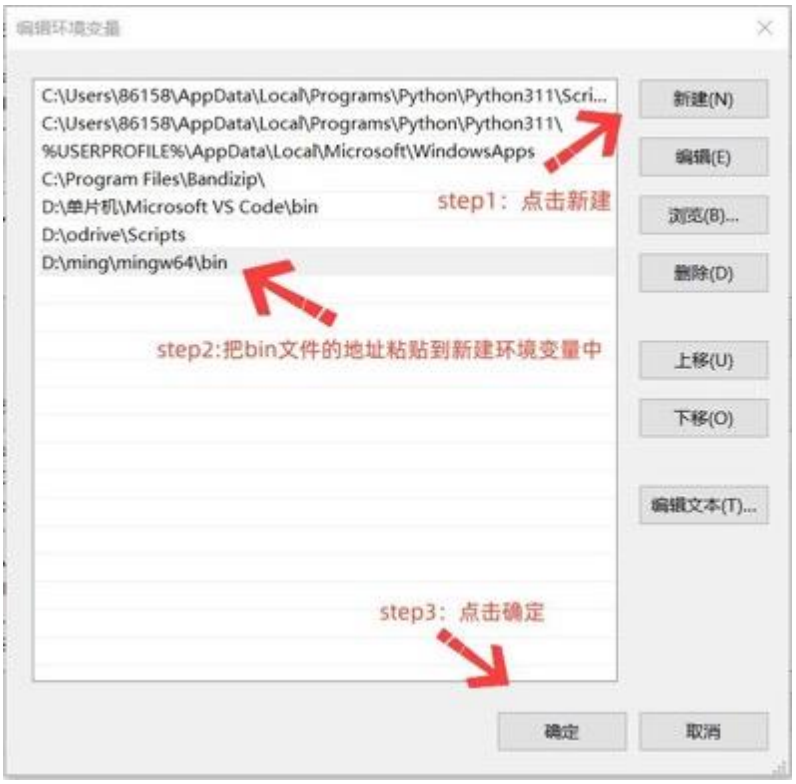


（1：电脑搜索打开“查看高级系统设置”）

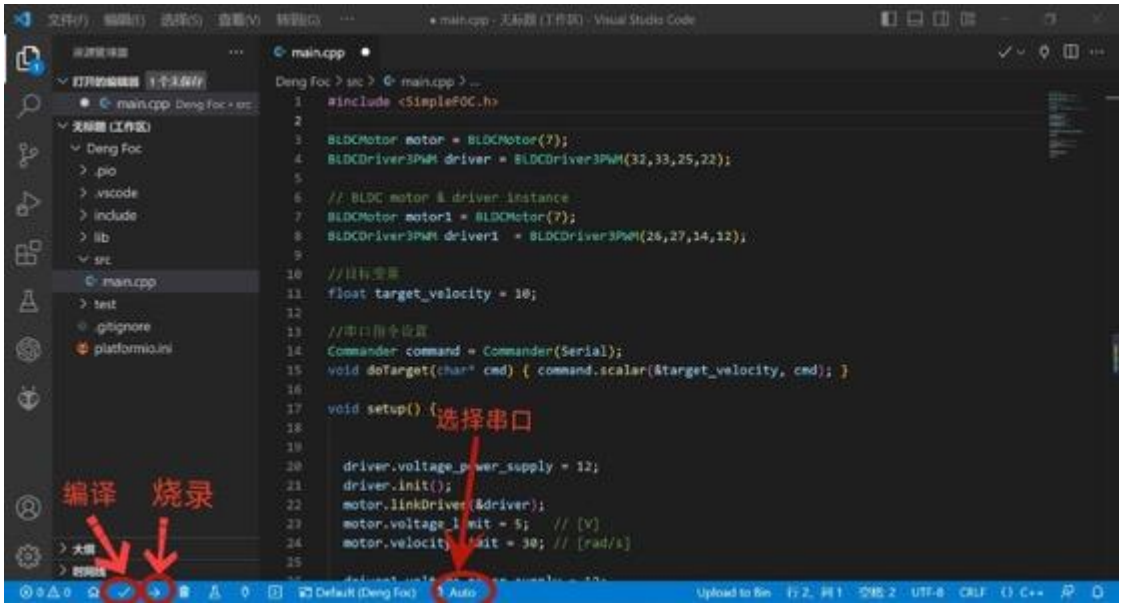
（2：点击环境变量）



（3：双击打开 Path）


编辑环境变量
C:\Users\86158\AppData\Local\Programs\Python\Python311\Scri...
C:\Users\86158\AppData\Local\Programs\Python\Python311\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Program Files\Bandizip\
D:\单片机\Microsoft VS Code\bin
D:\odrive\Scripts
D:\ming\mingw64\bin
新建(N)
编辑(E)
浏览(B)...
删除(D)
上移(U)
下移(O)
编辑文本(T)...
step1：点击新建
step2:把bin文件的地址粘贴到新建环境变量中
step3：点击确定
确定
取消
（4：新建环境变量）

第四步：编译及烧录：

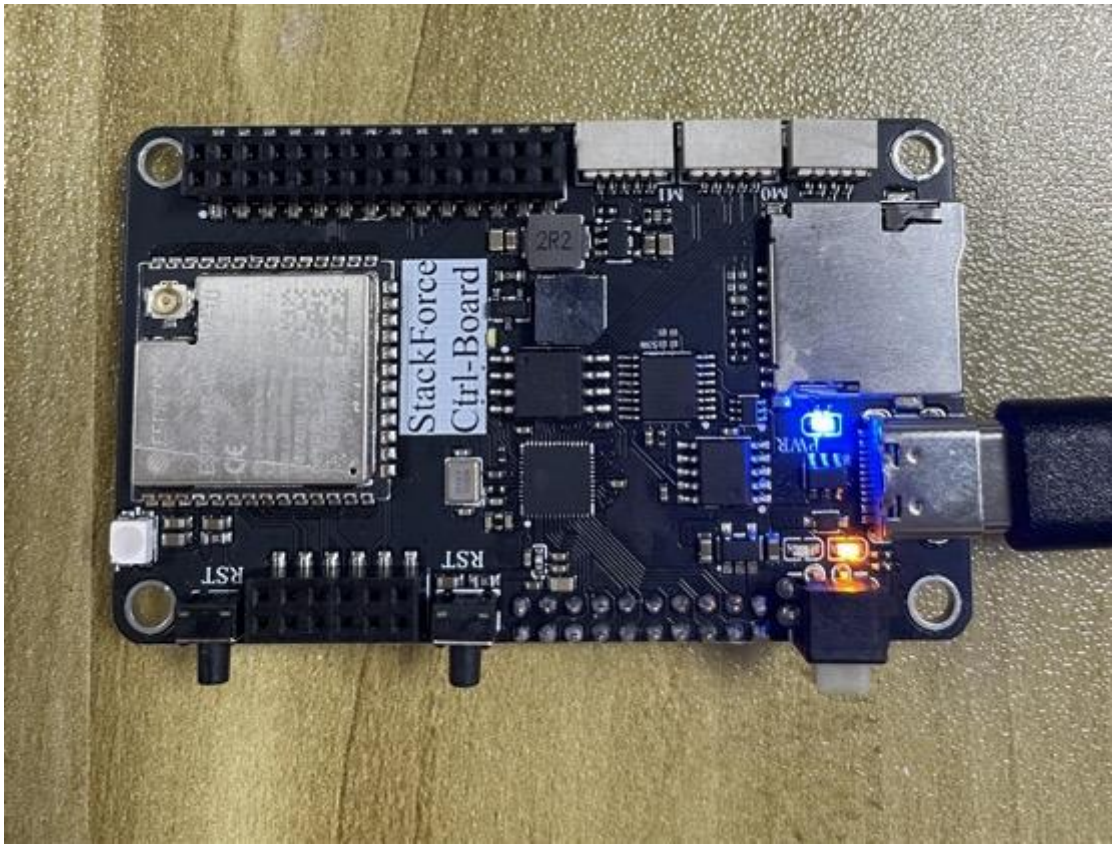


（√：编译程序　→：烧录程序到硬件　**Auto**：选择串口）

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）



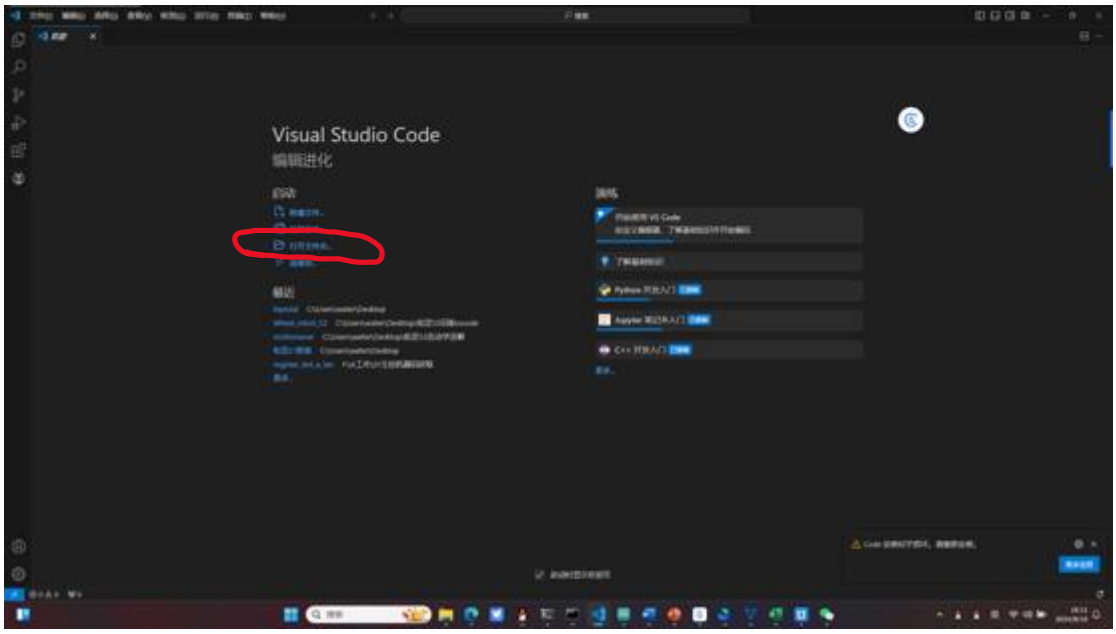
备注：为什么要分上下

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

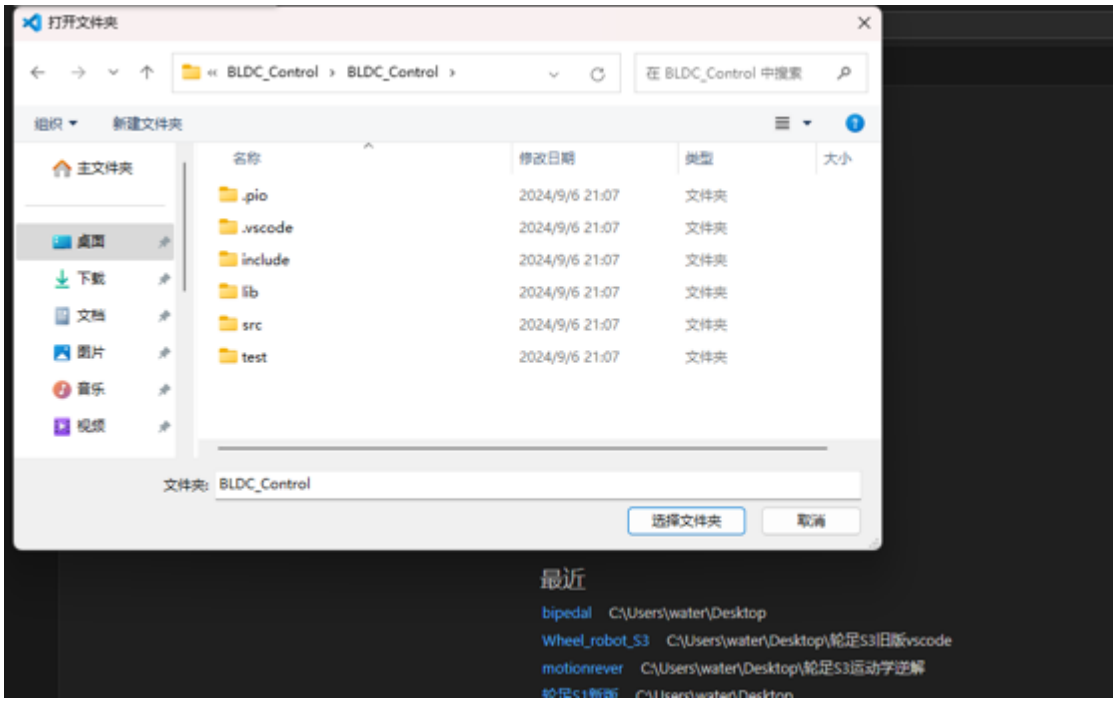
2.烧录 S1 程序

在 vscode 打开工程 BLDC_Control

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）



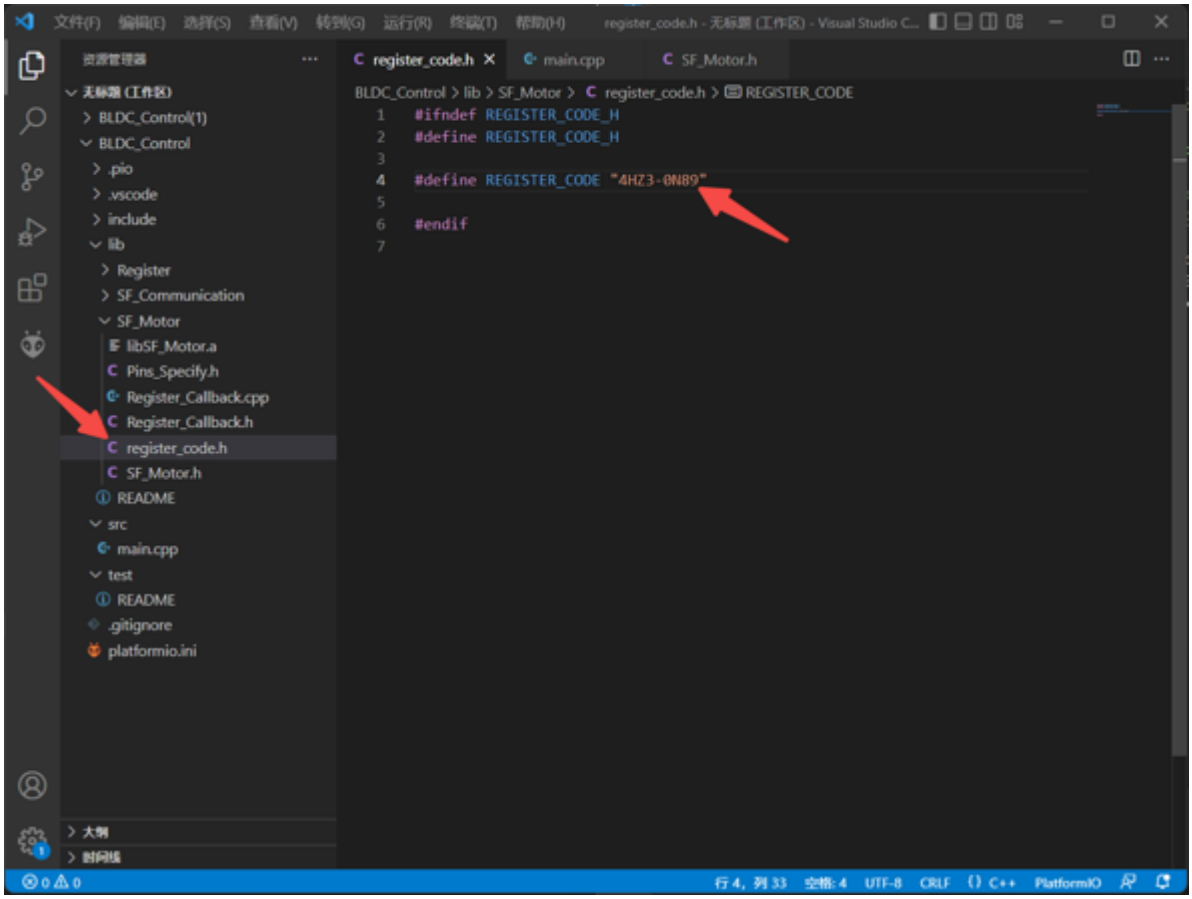
找到 S1 程序保存的位置，点击选择文件夹（不能有中文路径，且一定要打开到当前位置）



烧录前需要修改注册码与芯片的通讯方式：

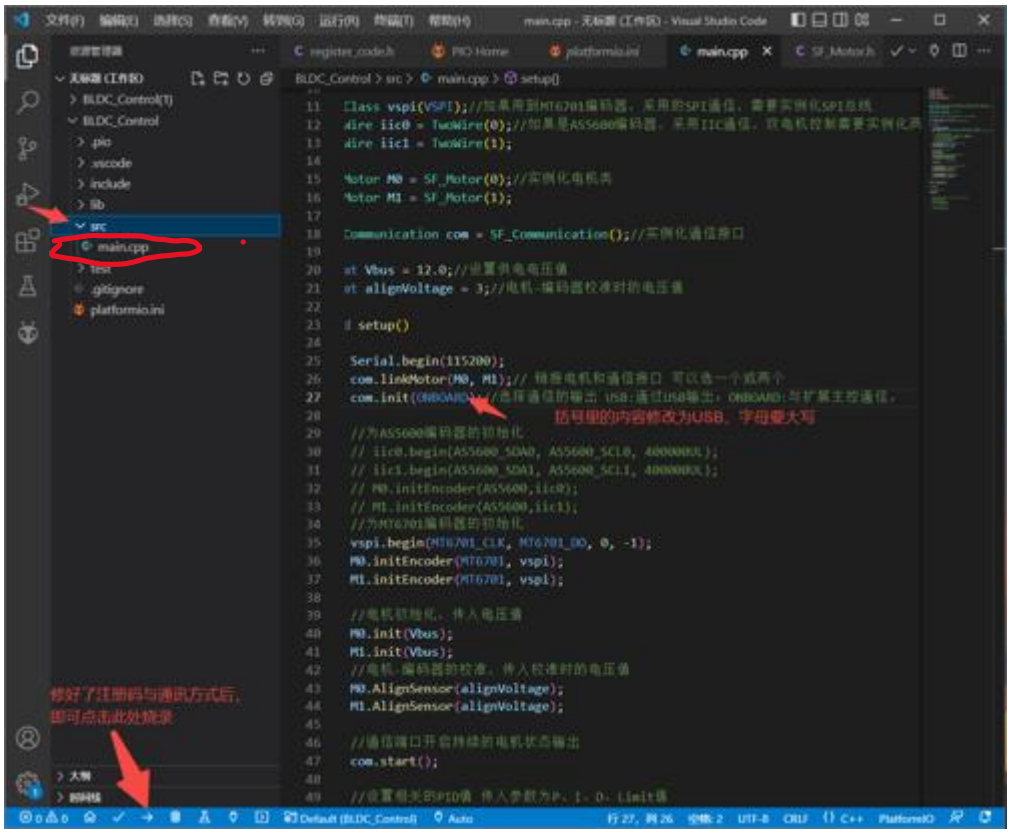
修改注册码：

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。



修改通信方式，在图中位置修改为 USB，

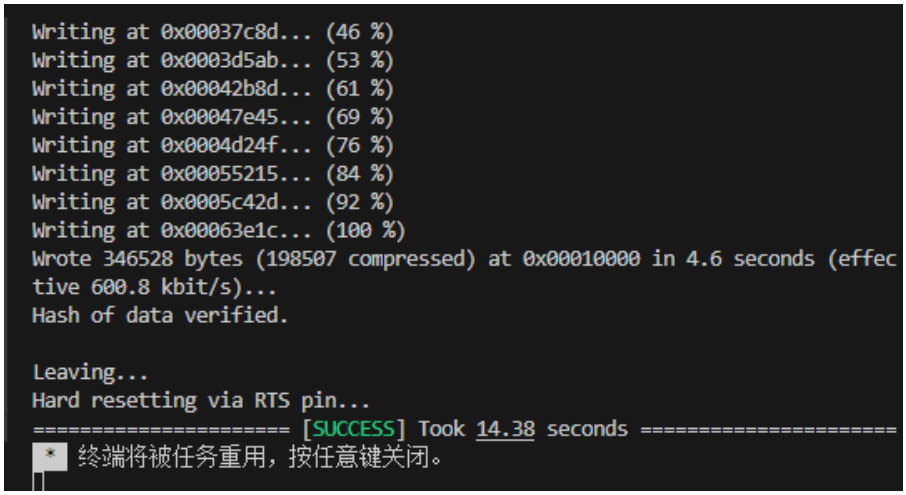
（USB 为主控板的 S1 芯片与电脑之间进行串口通信，ONBOARD 为 S1 芯片与 S3 芯片之间通信）



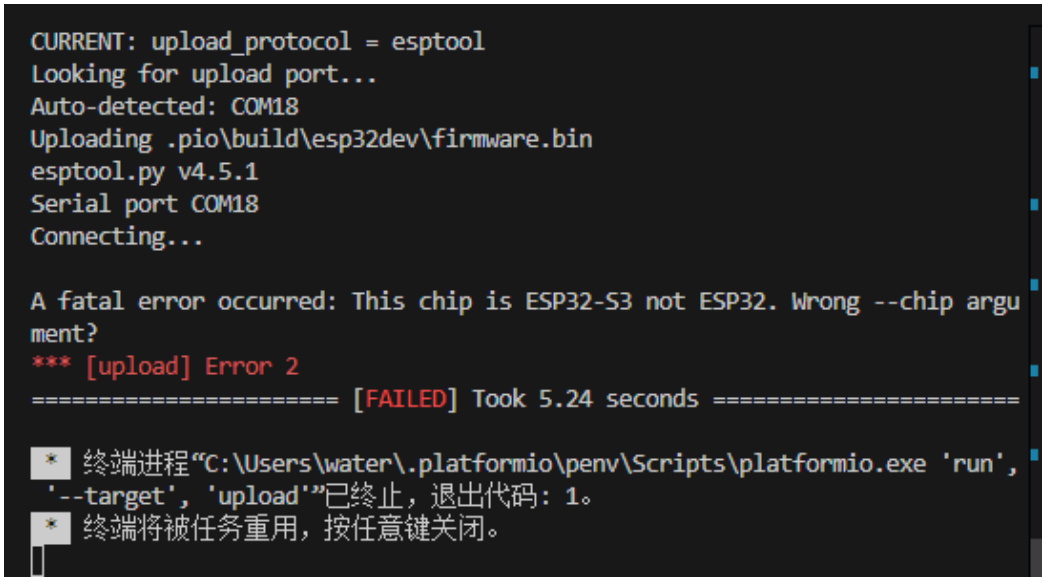
修改完成后点击左下角向右的箭头即可烧录程序

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



2、这个主控芯片是 S3，与 S1 程序不匹配



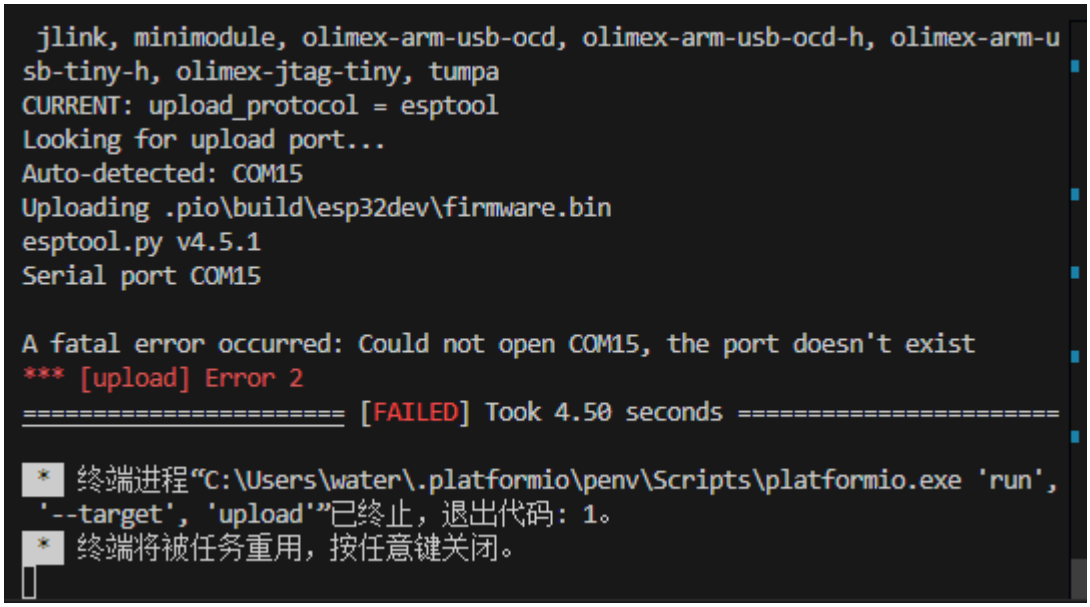
A fatal error occurred: This chip is ESP32-S3 not ESP32. Wrong --chip argument?

*** [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



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

*** [upload] Error 2

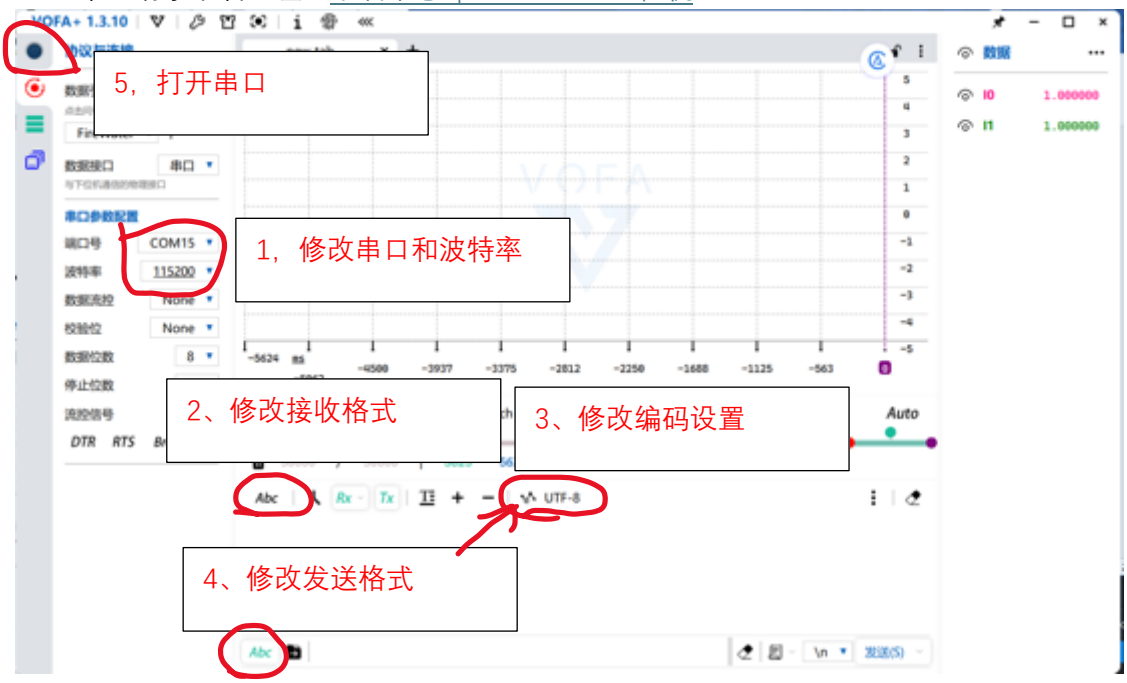
检查有没有其他软件占用了串口，

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

烧录后，打开串口助手，波特率为 115200，查看串口信息
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

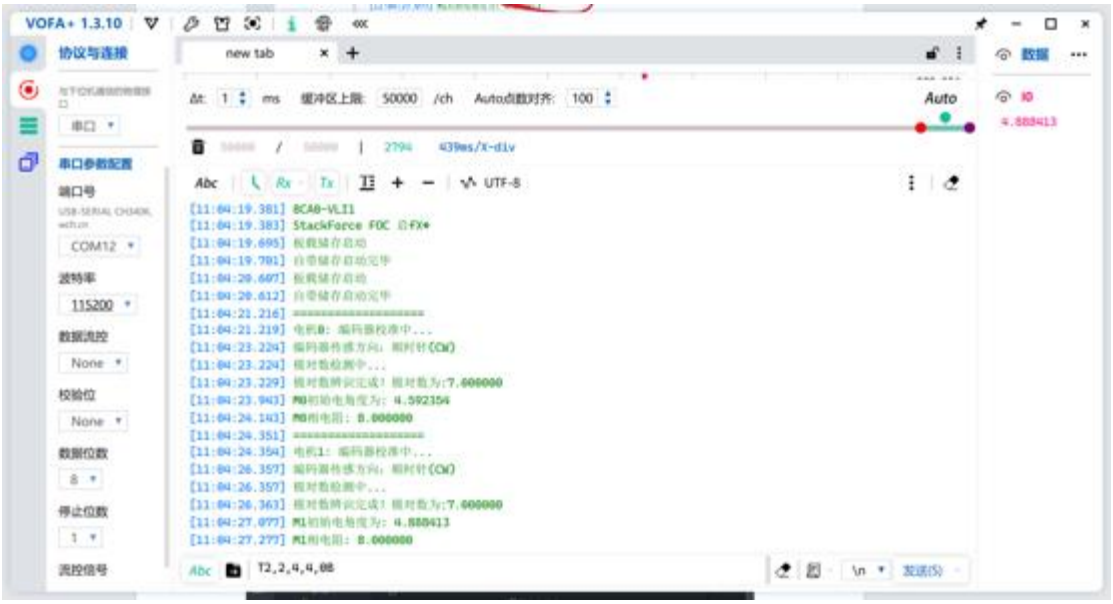
用手扶着机器人，轮子离开地面，复位 S1，等待轮子自检转动完成，
如极对数被辨别后是 7，则表示校准成功，则如下图所示，
若极对数为 inf 或其它英文字符串，则请检查是否打开了电源，检查线路是否接错
如果极对数为 6 或者 8，可能是车轮安装过紧，（可查看轮足安装文档“轮子轴承安装”调试）
或者是轮子与地面有摩擦（每次上电或 S1 复位都要让机器人离地自检才能正常运行）
可重新调试再 S1 复位直到极对数显示为 7 为止。

5.S1 电机控制程序调试，极对数校准

往串口输入数据 T2,2,4,4,0B,观察车轮转动情况，轮子能够丝滑转动就是没有问题的（方向在 S3 舵机控制程序调节，在这不用管），若存在问题，则输入 T0,0,4,4,0B 停止转动（关电源也能停），若转动不正常，请重新进行步骤 4 的内容。

T2,2,4,4,0B 为电机控制指令，前四个数字分别代表

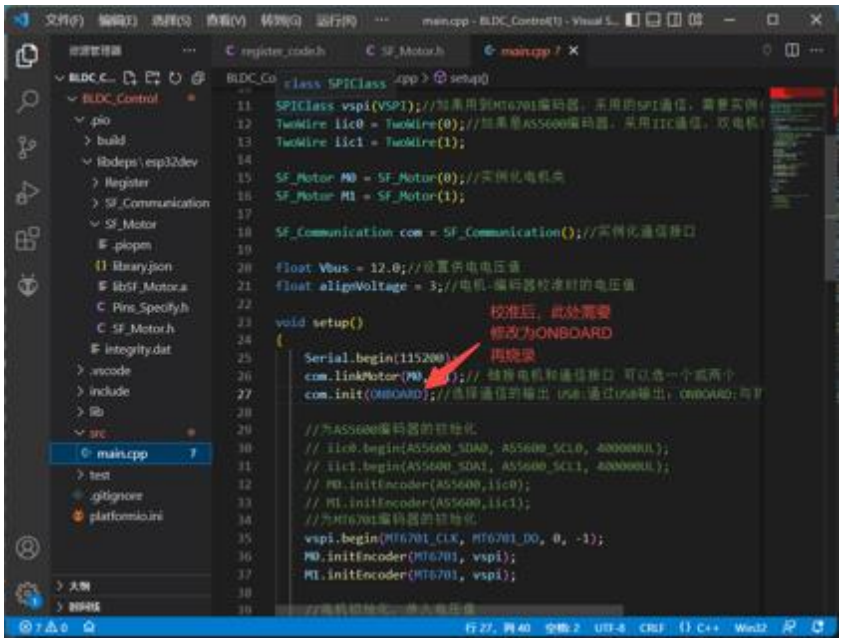
电机 0 目标力矩；电机 1 目标力矩；电机 0 模式为力矩模式；电机 1 模式为力矩模式



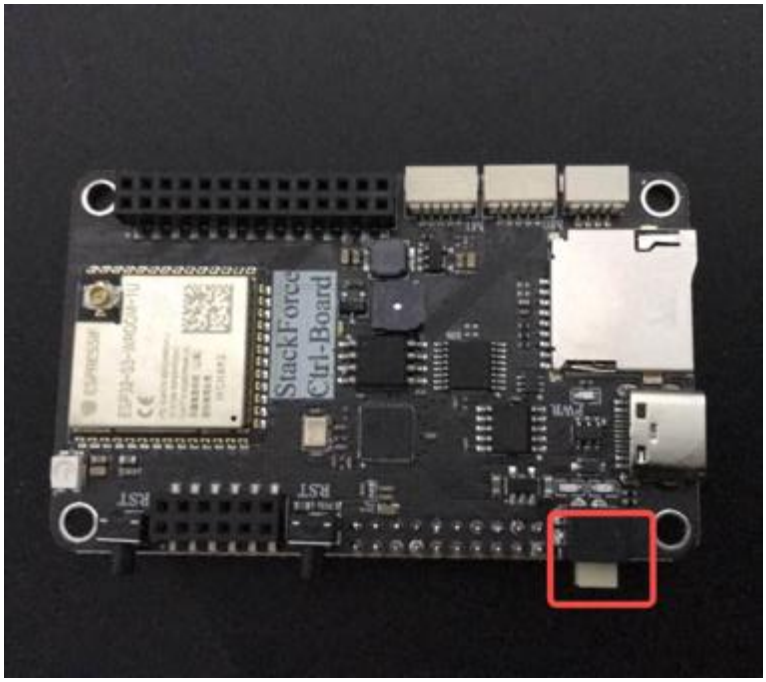
完成以上步骤后就是完成了电机调试，需要把芯片的通讯方式改为板载通讯模式“ONBOARD”，再将程序烧录进 S1

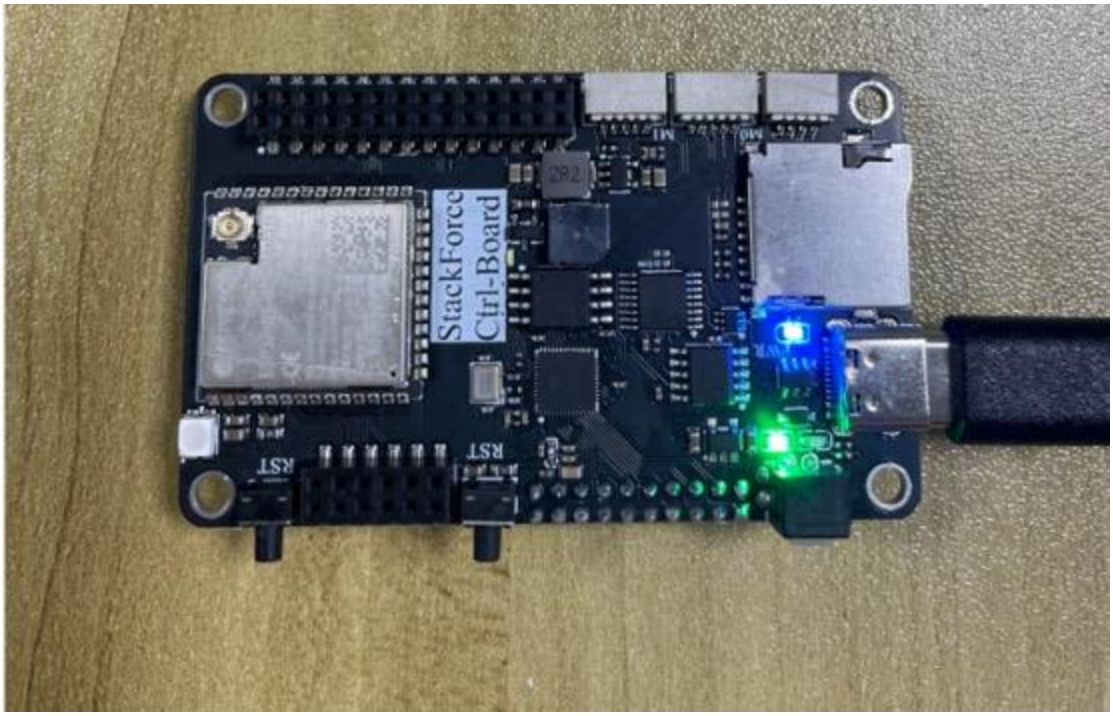
备注：usb 模式是通过串口助手发送 T2,2,4,4,0B 电机控制指令控制电机转动；

ONBOARD 模式是通过 S3 芯片上舵机控制程序发送电机控制指令控制电机转动



1. 烧录完 S1 后，切换至 S3 芯片，并烧录 S3 程序（USB 无缝隙朝上，按下图所示按钮，灯为绿色则切换至 S3）

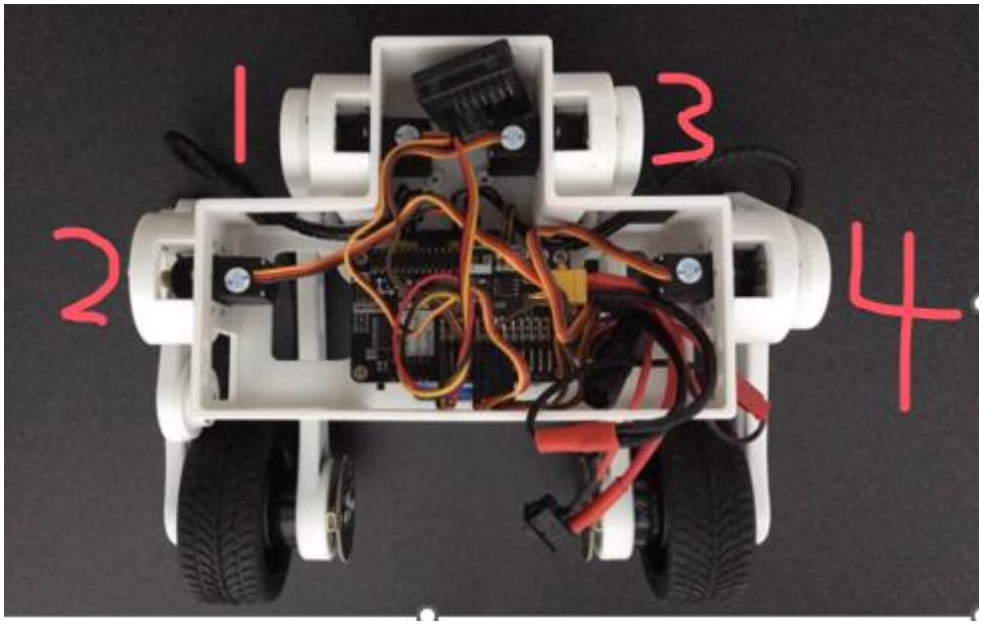




三、S3 偏置值获取

在烧录程序前一定要将大腿拆下

在 vsoce 打开 bipedal_calibrate 文件夹（偏置值获取程序），直接烧录程序，打开 vofa，波特率设置 115200，可以看到串口信息为 0,0,0,0;分别代表 1,2,3,4 号舵机的偏置值



这里调试可以看同文件目录下的校准视频，大致操作如下

1、等待舵机完成转动后安装腿部尽量垂直于水平面

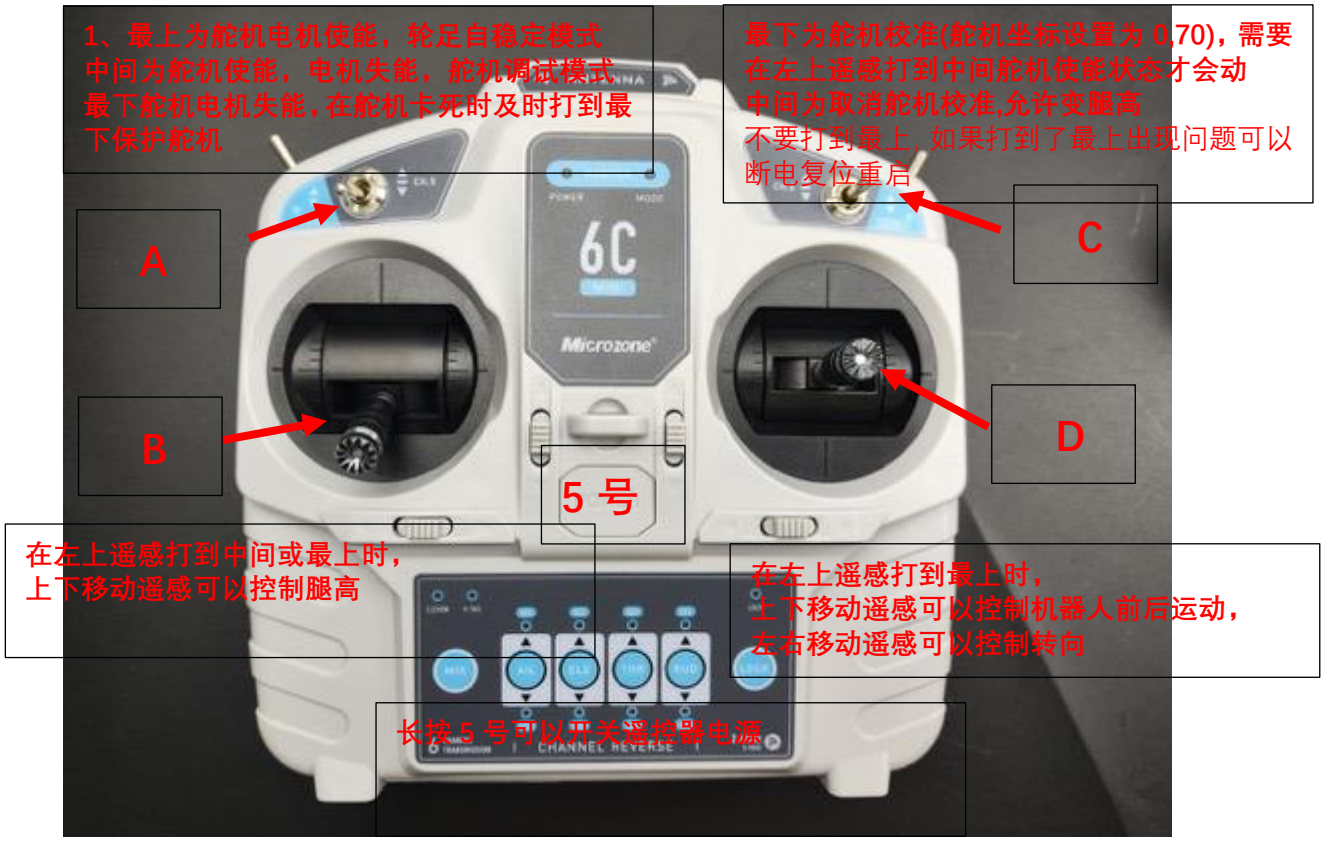
2、在串口输入 1,2,3,4 等指令控制舵机转动直到腿部完全垂直水平面

方向解释：腿部面向自己，顺时针为负，逆时针为正

3、舵机盒子里的黑色螺丝拧紧

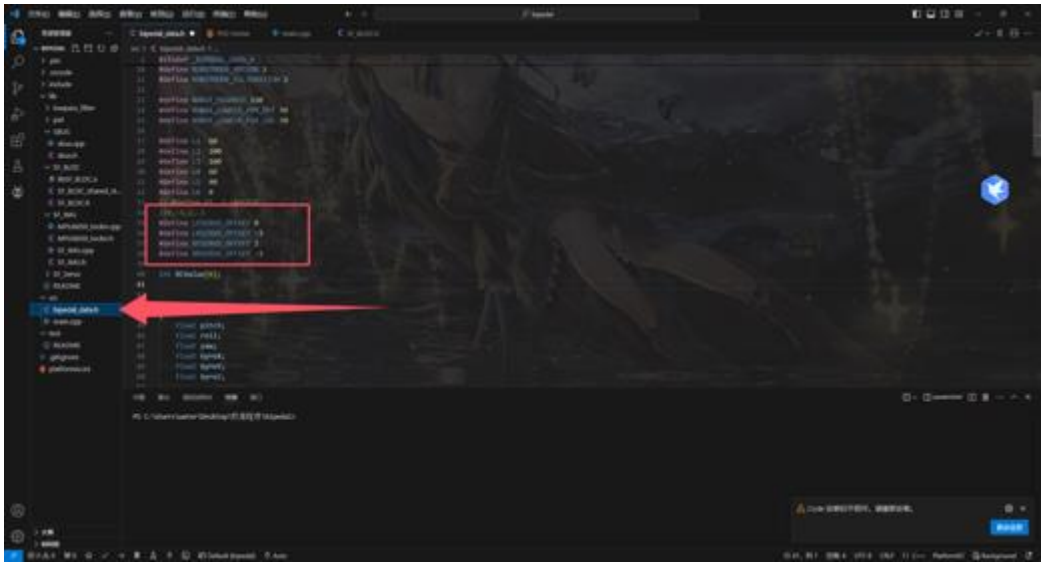
保存当前偏置值，要写到下面的 S3 舵机控制程序

四、S3 校准和调试



第一步：修改偏置值

在 vscode 打开 bipedal 文件夹（舵机控制程序），
在 bipedal_data.h 文件下修改偏置值 OFFSET，具体看如下



第二步：舵机调试

烧录完程序后拔开 usb，然后长按 5 打开遥控器遥控器 A 打下（失能），B 打下，C 打下
用手扶着机器人，并且让机器人保持水平陀螺仪校准和电机校准，按下按钮机器人上电，
自检完成后 A 打中舵机使能，等待舵机转回初始位置，C 打中（不能打上）取消舵机校准，
滑动 B 控制腿高移动 y 坐标，滑动 D 控制轮子前后移动 x 坐标

第三步：设置 SpdDir

设置电机固定转动方向，在 main 文件下搜索 motors.setTargets,
设置电机 0 和电机 1 的力矩固定为 2,

```
if (robotMode.motorEnable == 1 && robotPose.pitch <= 40 && robotPose.pitch >= -35) {
  motorsTarget.motorLeft = _constrain(motorsTarget.motorLeft, -5.7, 5.7);
  motorsTarget.motorRight = _constrain(motorsTarget.motorRight, -5.7, 5.7);
  // motors.setTargets(motorsTarget.motorLeft,motorsTarget.motorRight);
  motors.setTargets(2,2);
} else {
  motors.setTargets(0,0);
}
```

看下图 Print 出电机速度，loop 函数下

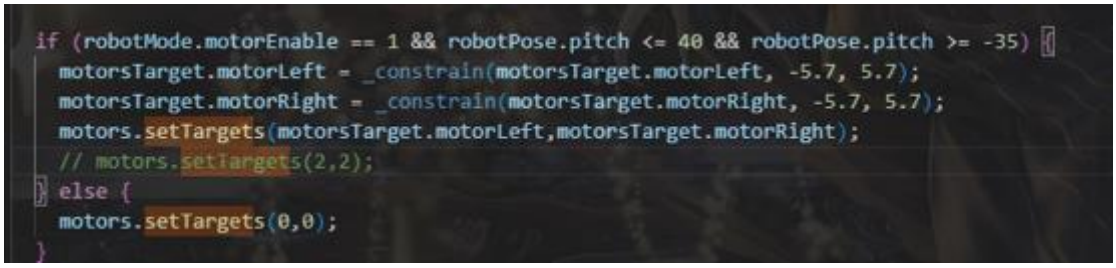
```
if (Ts > 1.0) {
  last_time = now_time;
  //轮子x坐标，左轮y坐标，右轮y坐标，机器人左右横滚角，机器人前后俯仰角，平均速度，右轮速度，左轮速度
  // Serial.printf("%f,%f,%f,%f,%f,%f,%f,%f\n",coordTarget.x,coordTarget.yLeft,coordTarget.yRight,robotPose.roll,robotPose.pitch,robotPose.speedAvg,motorSta
  //平均速度，右轮速度，左轮速度
  Serial.printf("%f,%f,%f\n",robotPose.speedAvg,motorStatus.M0SpdDir*motorStatus.M0Speed,motorStatus.M1SpdDir*motorStatus.M1Speed);
  //解释motorStatus.M0Speed是S1反馈的电机速度（轮子向前转动为正），如果反馈的速度与实际相反需要手动设置motorStatus.M0SpdDir取反，反之不用
}
```

烧录程序到 S3，然后打开 vofa 设置波特率为 961200,
遥控器 A 打上电机使能,
上电并用手扶着让电机自检完成（可以先用 vofa 与 S1 通信看看是否自检通过),
观察电机转动方向，左电机向前转则速度为正,
如果 vofa 上打印的第三个数据 M1 速度为负，则需要将 motorStatus.M1SpdDir 取反
右电机向后转则速度为负,
如果 vofa 上打印的第二个数据 M0 速度为负，则不需要修改 motorStatus.M0SpdDir

```
//电机速度控制方向
motorStatus.M0Dir = 1;
motorStatus.M1Dir = -1;
//电机速度反馈方向
motorStatus.M0SpdDir = -1;
motorStatus.M1SpdDir = 1;
```

第四步：设置电机控制 dir

按照下图取消电机控制的注释，注释下一行，烧录程序



```
if (robotMode.motorEnable == 1 && robotPose.pitch <= 40 && robotPose.pitch >= -35) {
  motorsTarget.motorLeft = _constrain(motorsTarget.motorLeft, -5.7, 5.7);
  motorsTarget.motorRight = _constrain(motorsTarget.motorRight, -5.7, 5.7);
  motors.setTargets(motorsTarget.motorLeft,motorsTarget.motorRight);
  // motors.setTargets(2,2);
} else {
  motors.setTargets(0,0);
}
```

给轮足上电，复位完后让轮足机器人向前倾斜，观察轮子转动方向
如果全部向前转动则可以平衡
如果左边向后则需要将 M1Dir 取反，右轮同理

```
//电机速度控制方向
motorStatus.M0Dir = 1;
motorStatus.M1Dir = -1;
//电机速度反馈方向
motorStatus.M0SpdDir = -1;
motorStatus.M1SpdDir = 1;
```