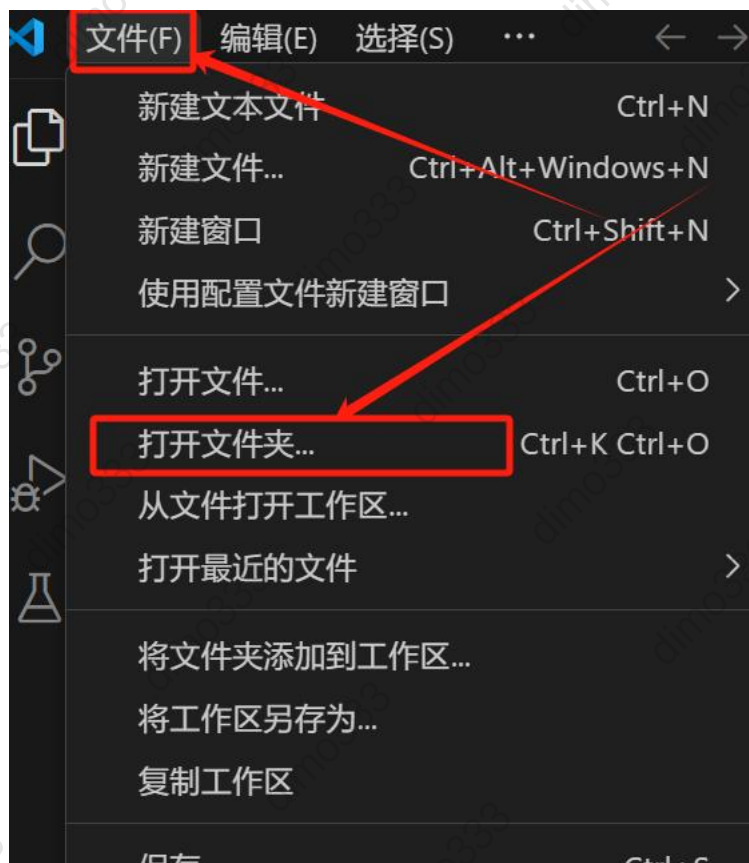
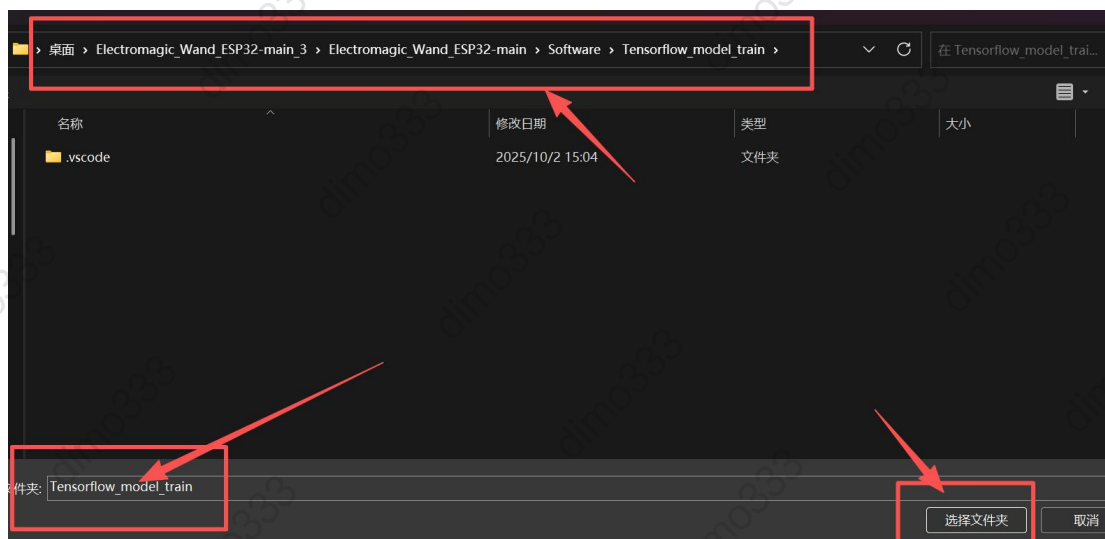


以下部分教程会**异常**的难，如果你畏惧了，可以在看完这一段话后开始放弃。开始前，请确保你成功运行并将烧录至魔杖，否则后续执行时可能会出现问题。

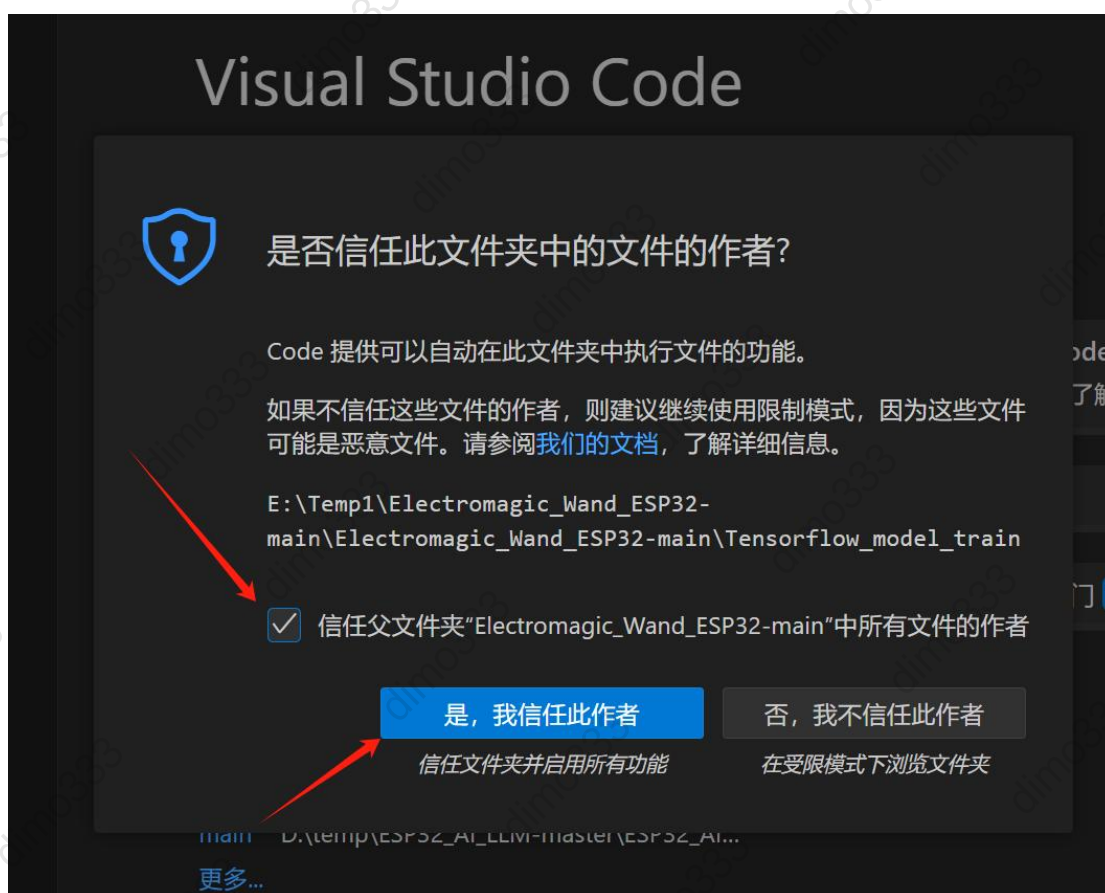
13、打开 vscode，点击左上角 文件->打开文件夹：



14、找到从 github 上下载的代码的文件夹，选择并打开它：

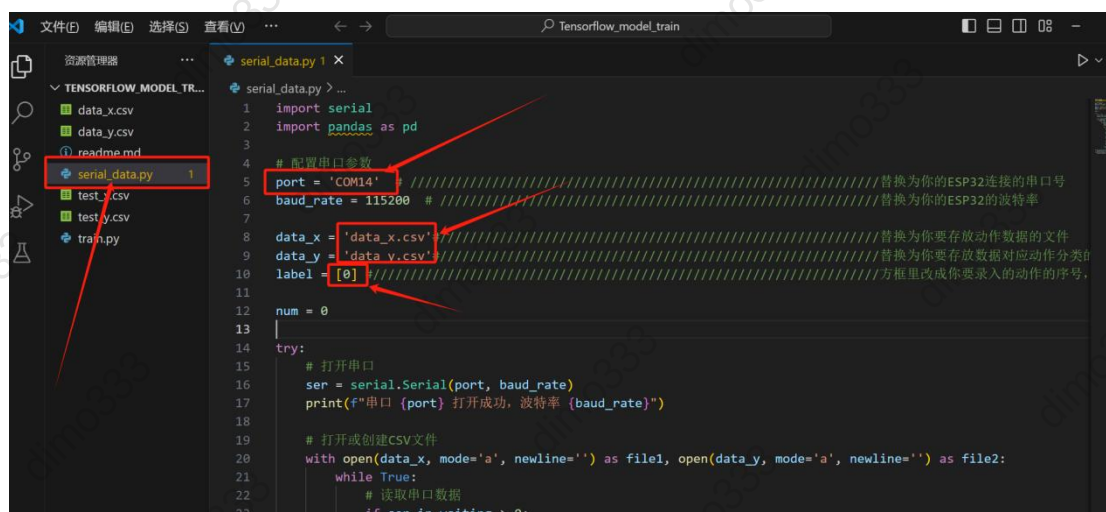


15、会弹出提示。选择信任我，好吗



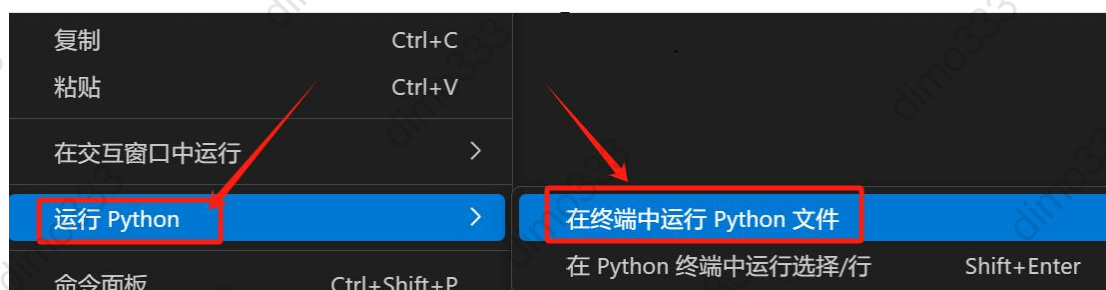
16、左边选择 **serial_data** 文件，修改你 **esp** 连上电脑的串口号，然后选择一个存放录入数据的 **csv** 文件（这里如果不想用我的数据，可以左边打开 **date_x.csv** 和 **data_y.csv** 删除里面的数据），然后在 **label** 的方框里填写需要录制的动作序号（这里我以动作 **0** 举例，并且已经清除了 **date_x.csv** 和 **data_y.csv** 的所有数据）。

（这一步记得关闭 **arduinoIDE**，避免串口被占用）



```
1 import serial
2 import pandas as pd
3
4 # 配置串口参数
5 port = 'COM14' // 替换为你的ESP32连接的串口号
6 baud_rate = 115200 // 替换为你的ESP32的波特率
7
8 data_x = 'data_x.csv' // 替换为你要存放动作数据的文件
9 data_y = 'data_y.csv' // 替换为你要存放数据对应动作分类
10 label = [0] // 方框里改成你要录入的动作的序号。
11
12 num = 0
13
14 try:
15     # 打开串口
16     ser = serial.Serial(port, baud_rate)
17     print(f"串口 {port} 打开成功, 波特率 {baud_rate}")
18
19     # 打开或创建CSV文件
20     with open(data_x, mode='a', newline='') as file1, open(data_y, mode='a', newline='') as file2:
21         while True:
22             # 读取串口数据
23             if ser.in_waiting > 0:
```

17、右键点击 运行 **python->**在终端中运行 **python** 文件



18、出现以下界面即为成功打开（这一步可能会随机录一个值进去，可以去 data_x、data_y 里面删掉这一组数据，删除完后记得保存）

```
问题 输出 调试控制台 终端 端口
DLL from .libs:
C:\Users\dim03\.conda\envs\TF2.1\lib\site-packages\numpy\
.libs\libopenblas.NOIJJG62EMASZI6NYURL6JBKM4EVBGM7.gfortr
an-win_amd64.dll
C:\Users\dim03\.conda\envs\TF2.1\lib\site-packages\numpy\
.libs\libopenblas.WCDJNK7YVMPZQ2ME2ZZHJJRJ3JIKNDB7.gfortr
an-win_amd64.dll
stacklevel=1)
串口 COM14 打开成功, 波特率 115200
读取到数据: -0.39,-0.11,-0.37,-0.09,-0.34,-0.08,-0.32,-0.
05,-0.28,-0.05,-0.26,-0.03,-0.24,-0.03,-0.22,-0.03,-0.20,
-0.01,-0.19,-0.00,-0.18,0.01,-0.16,0.01,-0.15,0.00,-0.13,
0.00,-0.12,0.00,-0.12,0.01
数据追加到CSV文件成功:0
```

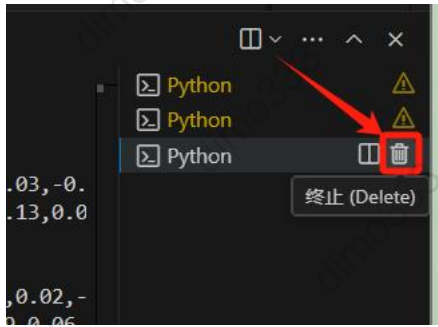
19、短按按键并松开，然后开始绘制自己想要录制的动作

20、出现以下界面即为成功录制

```
问题 输出 调试控制台 终端 端口

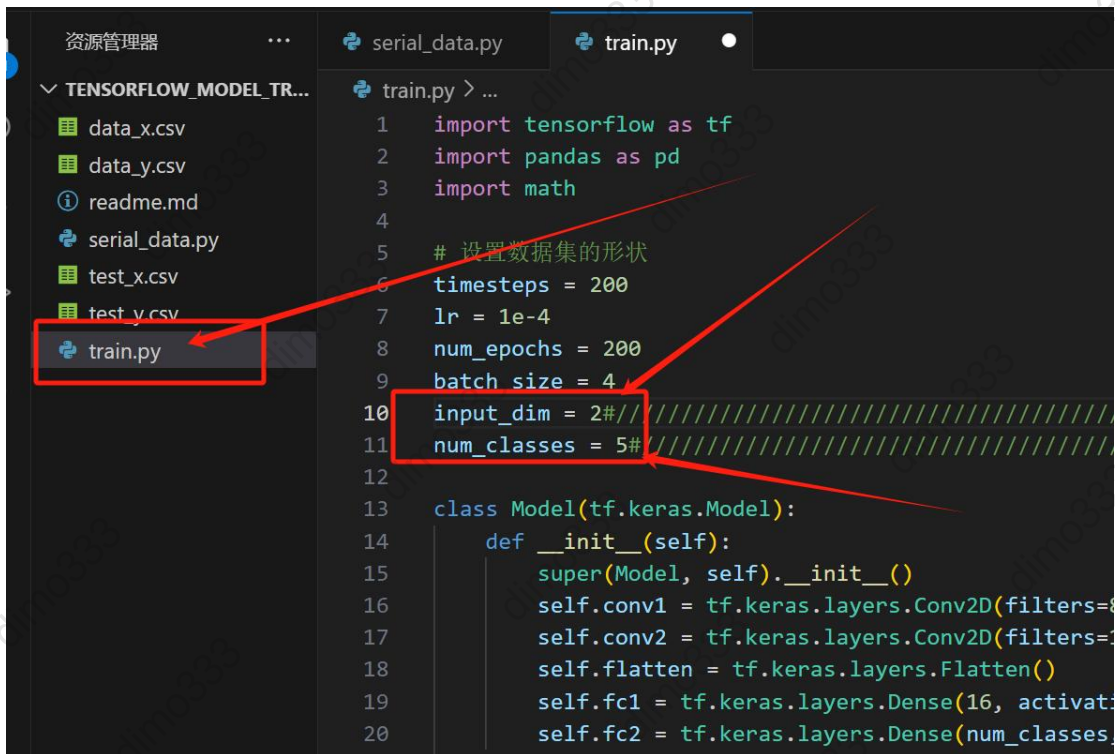
R3JJIKNDB7.gfortran-win_amd64.dll
stacklevel=1)
串口 COM14 打开成功, 波特率 115200
读取到数据: -0.39,-0.11,-0.37,-0.09,-0.34,-0.08,-0.32,-0.05,-0.28,-0.05,-0.26,-0.03,-0.24,-0.03,-
0.22,-0.03,-0.20,-0.01,-0.19,-0.00,-0.18,0.01,-0.16,0.01,-0.15,0.00,-0.13,0.00,-0.12,0.00,-0.12,0
.01
数据追加到CSV文件成功:0
读取到数据: -0.01,0.00,0.03,-0.01,-0.01,-0.02,-0.05,-0.01,-0.04,0.01,-0.02,0.02,-0.00,0.01,
-0.02,0.01,0.01,0.00,0.01,-0.03,-0.03,0.02,0.07,-0.05,-0.09,-0.06,-0.09,0.06,-0.11,0.03,0.03,-0.0
3,0.02,-0.07,0.01,0.09,0.03,0.03,0.13,-0.05,0.01,-0.06,-0.04,-0.05,-0.05,-0.02,-0.01,-0.02,0.01,-
0.01,0.04,-0.03,0.01,0.03,0.02,0.02,0.01,0.03,0.02,0.02,0.02,0.04,0.00,0.05,0.00,0.05,0.00,0.00,0
.05,0.00,-0.02,0.05,-0.00,0.13,0.13,0.06,0.05,0.09,0.04,0.01,0.09,-0.13,0.16,-0.18,0.10,-0.20,
0.02,-0.06,0.04,-0.18,-0.12,0.22,0.02,0.18,0.08,0.24,-0.01,0.16,-0.12,0.12,-0.18,0.06,-0.13,-0.02
,-0.08,-0.05,-0.06,0.01,-0.02,-0.02,-0.04,0.06,-0.02,0.06,0.00,0.12,0.03,0.09,0.03,0.05,0.01,0.03
,0.01,-0.01,0.03,-0.04,-0.04,0.00,0.02,-0.13,0.01,0.06,0.02,-0.02,0.02,0.05,-0.06,0.08,-0.03,0.03
,0.01,-0.01,0.02,-0.00,-0.01,0.04,-0.01,0.03,-0.00,0.02,0.00,0.01,-0.01,0.03,0.00,0.02,-0.02,0.01
,-0.00,0.02,-0.00,0.02,-0.01,0.02,-0.00,0.03,-0.00,0.03,0.07,0.01,0.04,-0.02,-0.02,0.02,-0.05,0.0
3,-0.04,0.01,0.04,0.02,0.00,0.01,-0.00,0.02,0.01,0.01,0.01,0.01,-0.01,0.03,-0.03,0.05,-0.02,0.02,
0.01,-0.00,0.01,-0.00,-0.01,0.04
数据追加到CSV文件成功:1
```


21、动作 0 录制完成后，关闭当前终端



22、如果需要继续录制动作 1，请返回第 16 步继续操作

23、如果需要训练当前录制完成的动作模型文件，左边文件中选择 **train** 文件，修改代码当中 **input_dim** 参数（如果你只使用了两个轴，比如 **x** 轴、**z** 轴，这里就需要填写 2），修改当前代码当中 **num_classes** 参数（几个手势就写几，1 个就写 1，2 个就写 2）

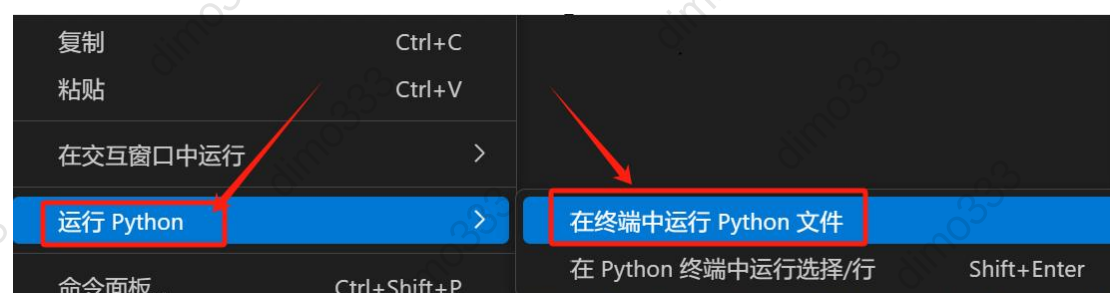


24、往下找到文件定义部分，如果不需要 test_x、test_y 可以替换成 data_x、data_y（亦或把 data_x、data_y 中的数据抽取部分数据填写至 test_x、test_y）

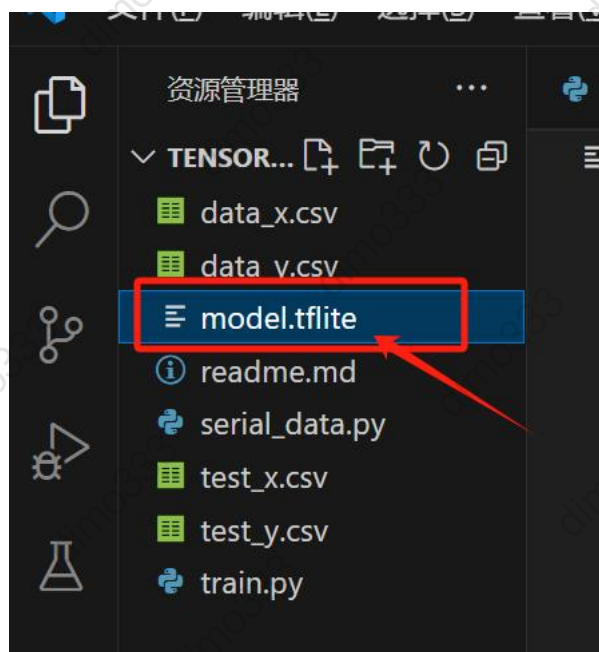
我使用时是一个动作录制 50 组数据，随机抽取 10 个到 test_x、test_y，仅供参考

```
27     x = self.flatten(x)
28     x = self.fc1(x)
29     x = self.fc2(x)
30
31     return x
32
33 if __name__ == '__main__':
34
35     train_x_pd = pd.read_csv('data_x.csv', header=None)#####
36     train_y_pd = pd.read_csv('data_y.csv', header=None)#####
37     test_x_pd = pd.read_csv('test_x.csv', header=None)#####
38     test_y_pd = pd.read_csv('test_y.csv', header=None)#####
39
40     train_x = tf.convert_to_tensor(train_x_pd.to_numpy(), dtype=tf
41     train_x = tf.reshape(train_x, [-1, timesteps, input_dim])
42     train_y = tf.convert_to_tensor(train_y_pd.to_numpy(), dtype=tf
43
```

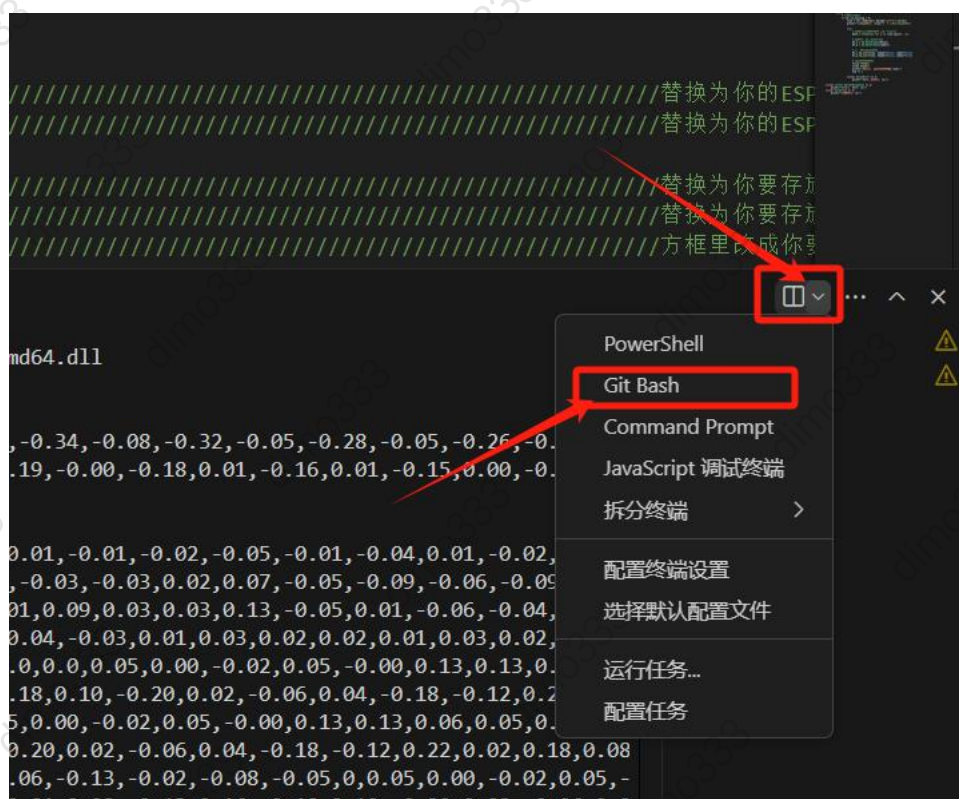
25、一切准备就绪后右键点击 运行 python->在终端中运行 python 文件



26、成功运行完后左边会出现一个 model.tflite 文件



27、打开新的终端，选择 git bash（如果你的电脑是 window）



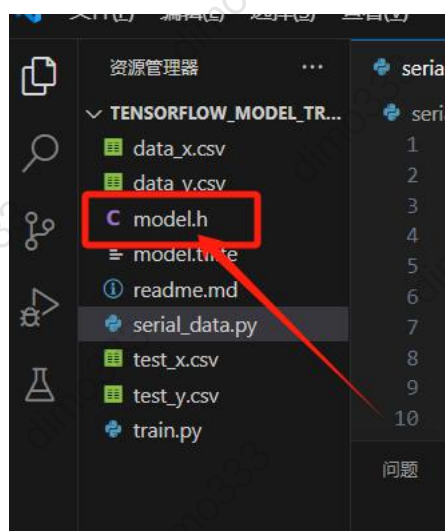
28、在新的 gitbash 终端中输入指令并回车：

```xxd -i model.tflite > model.h```



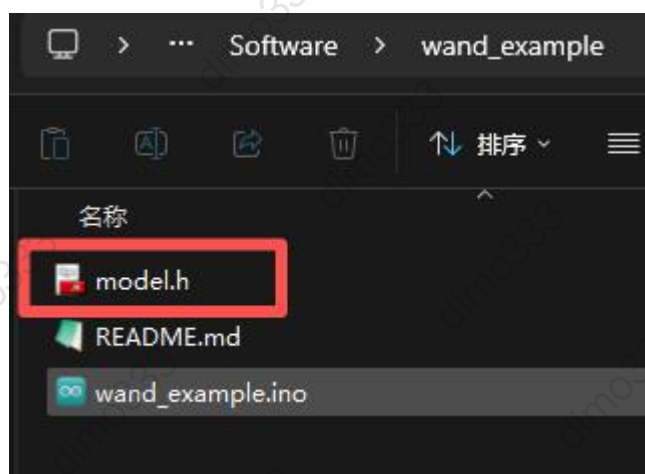
A screenshot of a terminal window with a dark background. The prompt is 'dimo3@dimo MINGW64 /e/Temp1/Electromagic\_Wand\_ESP32-main/Electromagic\_Wand\_ESP32-main/ensorflow\_model\_train'. The command '\$ ``xxd -i model.tflite > model.h```' is entered and highlighted with a red rectangle. A red arrow points from the command to the next line, which shows the prompt '\$' again.

29、此时左上角会生成一个新的 model.h 文件

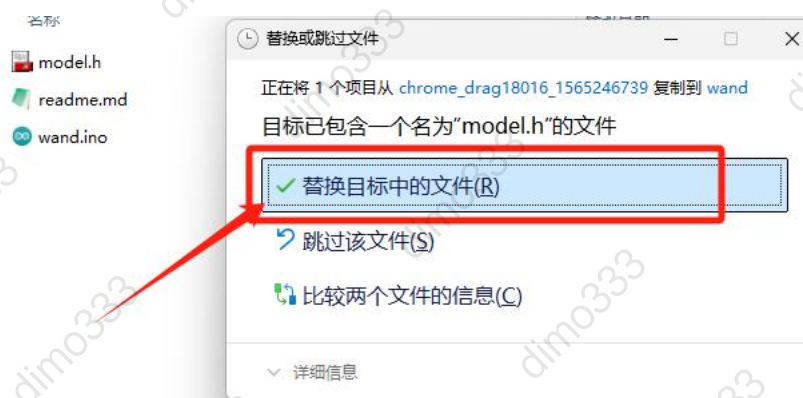




30、把这个文件复制，然后打开 wand 文件夹，与 wand 文件同级存放，以下以 wand\_example 为例



31、替换掉原有的.h 模型文件



**32、**此时再打开 wand 文件，调整一下你的代码（按键、动作对应的控制等）

```
const int buttonPin = 4; // 按钮引脚
const int ledPin = 12; // LED 引脚
int buttonState; // 当前按钮状态
int lastButtonState = HIGH; // 上一次按钮状态

// 主函数
void setup() {
 pinMode(buttonPin, INPUT);
 pinMode(ledPin, OUTPUT);
}

void loop() {
 if (max_index == 0) { // 第一次按下按钮
 digitalWrite(ledPin, HIGH);
 delay(200);
 digitalWrite(ledPin, LOW);
 delay(200);
 } else if (max_index == 1) { // 第二次按下按钮
 // ...
 } else if (max_index == 2) { // 第三次按下按钮
 // ...
 } else {
 // ...
 }
}
```

**33、**左上角点击上传



**34、**此时，你的 魔杖 就注魔完成了。

你可以开始使用你的魔杖了：点击按键->松开按键->在两秒内做动作->魔杖识别并执行设置好的代码。