遇到的问题：

Usb相关

1、汇报描述符在usbd\_custom\_hid\_if.c文件中，修改了结构体内容后如果数量有变化，对应的结构体大小宏USBD\_CUSTOM\_HID\_REPORT\_DESC\_SIZE也要记得修改。

2、CUSTOM\_HID\_FS\_BINTERVAL宏定义了发送的时间间隔，默认是0x20，比较大，如果实时性要求高要改小。

3、定义的结构体要注意变量大小和顺序，不然编译器会自动对齐导致实际大小和期望的大小不一致。

4、usb会用到比较多的空间，cubemx默认的0x200不够，需要弄大一点。这个问题的现象就是下载完程序不执行，注释掉usb相关又OK了。

Platformio相关：

1. cubemx生成的工程用platformio打开，编译的时候默认使用了platformio的库文件，包括一些头文件，比如stm32f4xx\_hal\_conf.h，默认用了

C:\Users\<yourname>\.platformio\packages\framework-stm32cubef4\Drivers\STM32F4xx\_HAL\_Driver\Inc\stm32f4xx\_hal\_conf.h的头文件，对于f4芯片，这里的外部时钟HSE\_VALUE配置为了25000000U，如果我们用的是8Mhz的晶振的话，就会导致整个系统时钟错误，表现为HAL\_Delay函数严重不准。

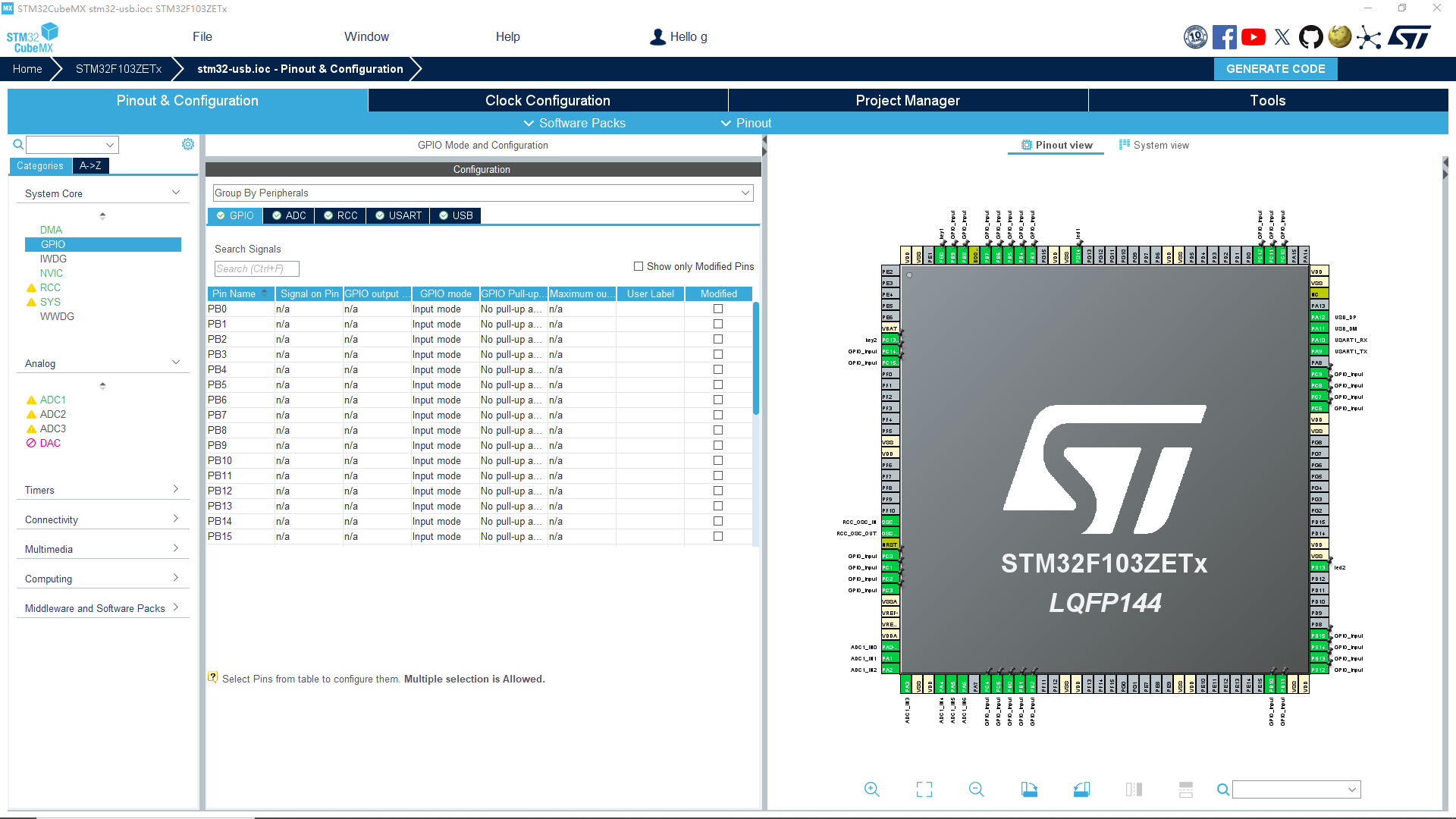
解决办法是在platformio.ini文件中添加board\_build.stm32cube.custom\_config\_header = yes配置项，这样就会使用本地定义的头文件。

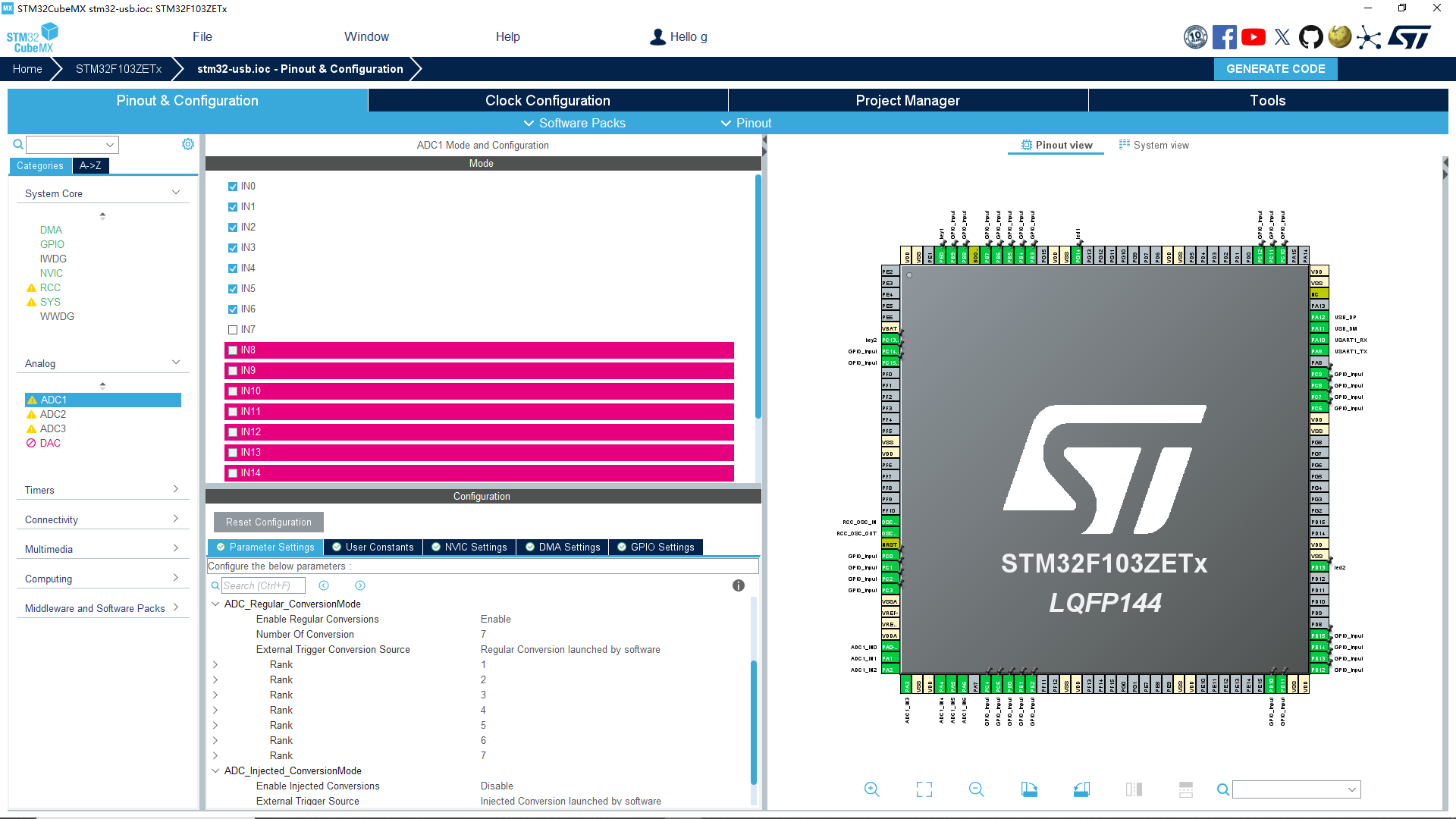
参考链接：

<https://docs.platformio.org/en/latest/frameworks/stm32cube.html#:~:text=board_build.stm32cube.custom_config_header%20%3D%20yes>

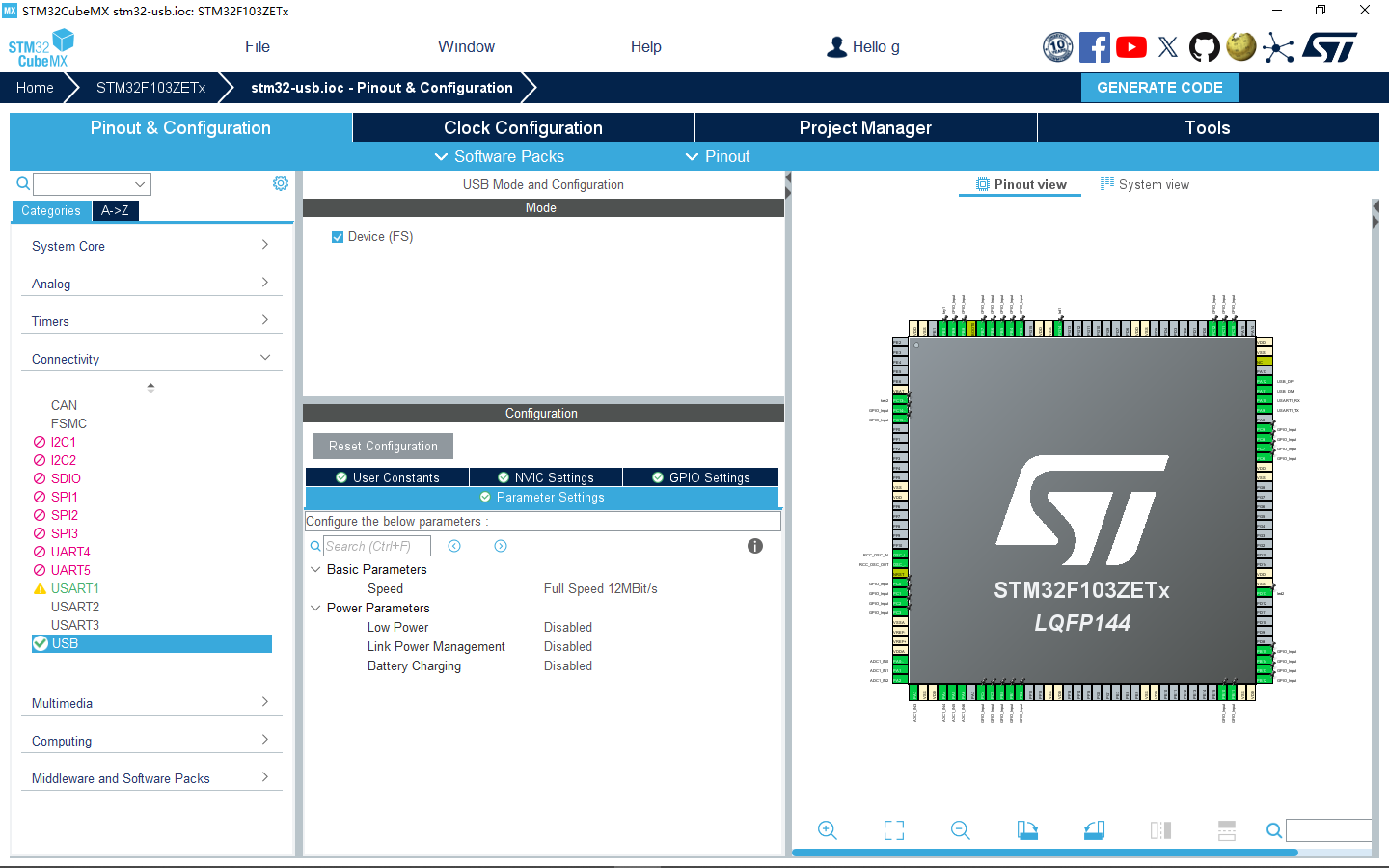
用stm32的usb hid实现游戏控制器

1. 用cubemx生成基础工程，游戏控制器一般就是按钮和旋钮的组合，按钮就是gpio，旋钮是adc。具体配置就不罗嗦了。

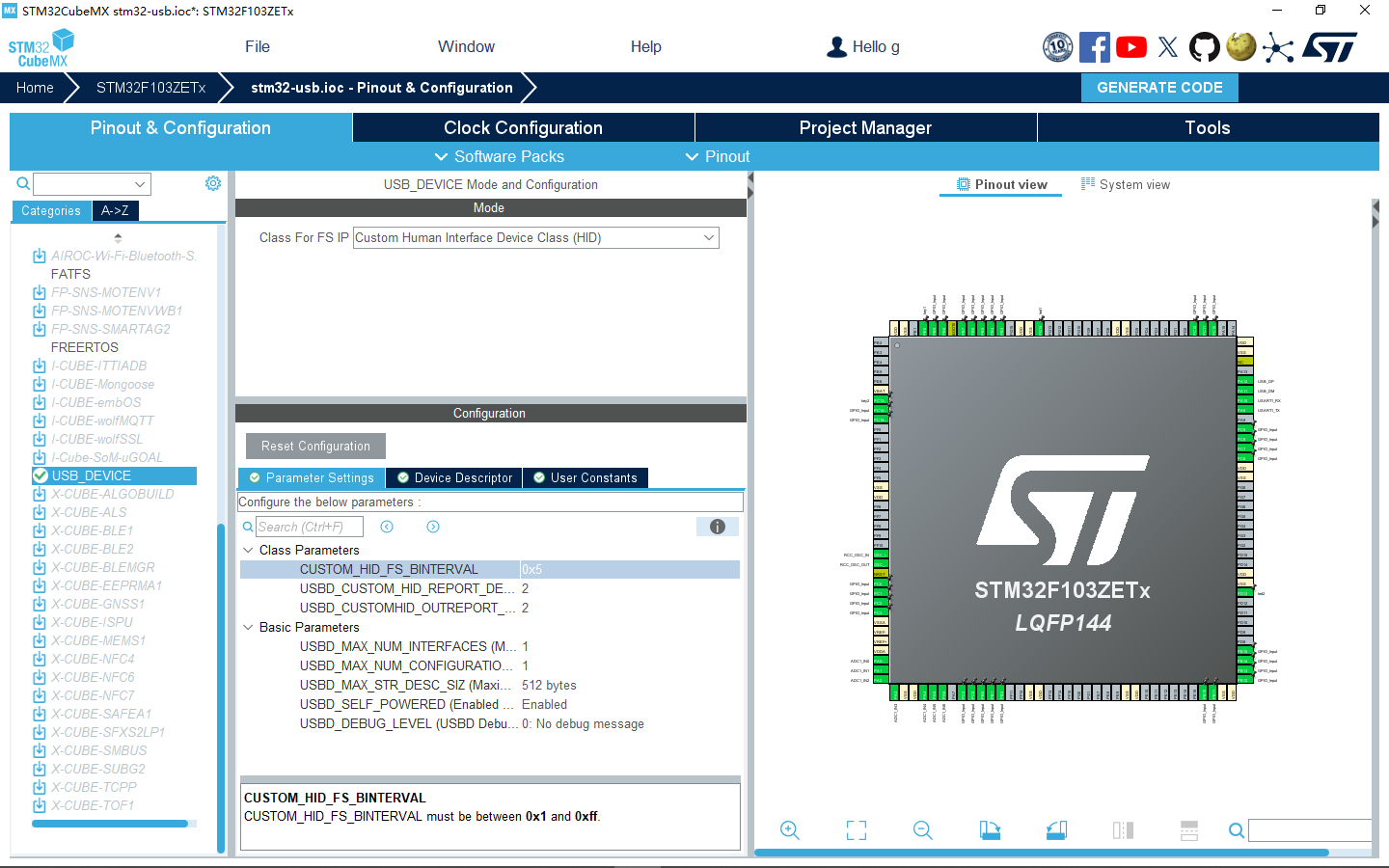




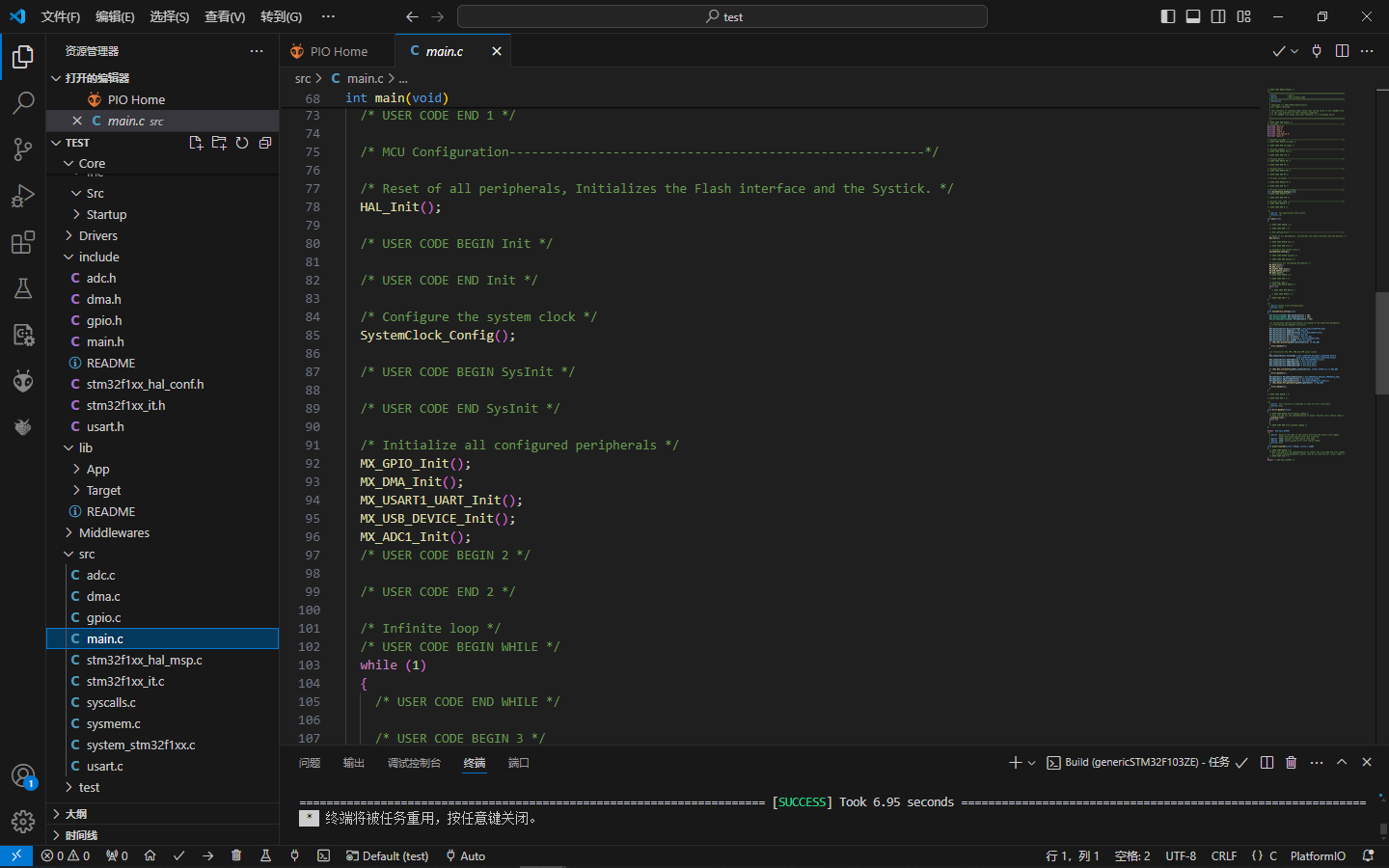
connectivity这里打开usb，勾选Device，参数保持默认即可。



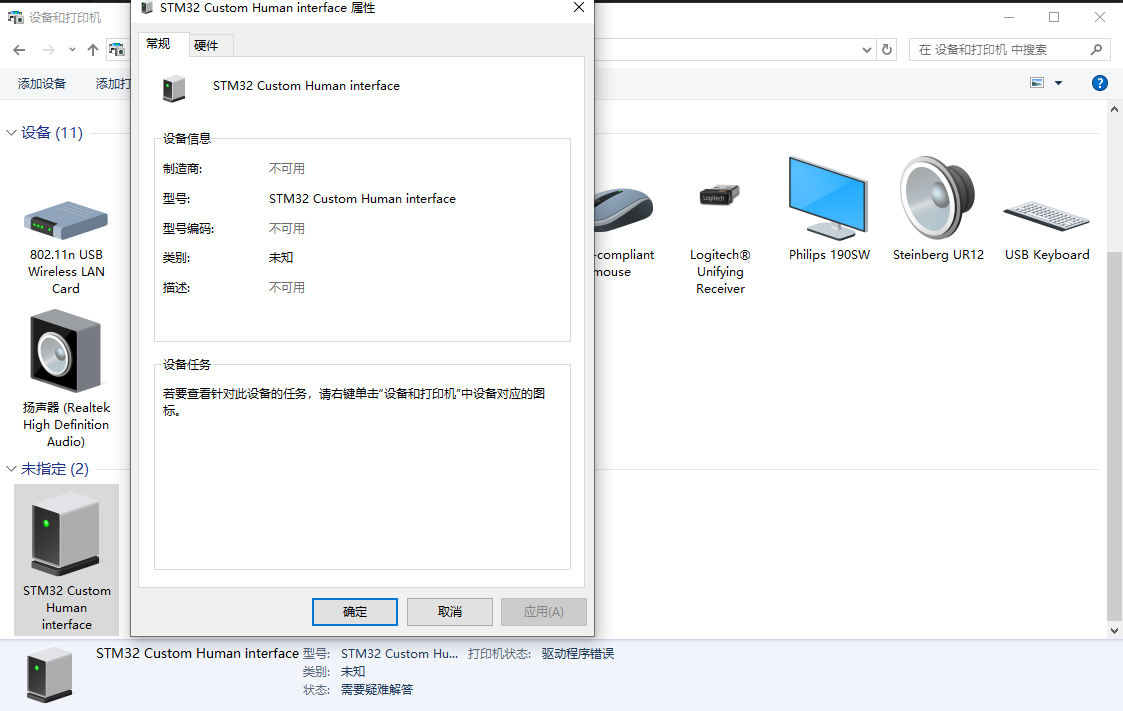
Middleware and Software Packs这里选择USB\_DEVICE，Mode这里选择Custom HID，参数如果懂可以根据需求改，不懂保持默认，后边我们在代码里改也可以。



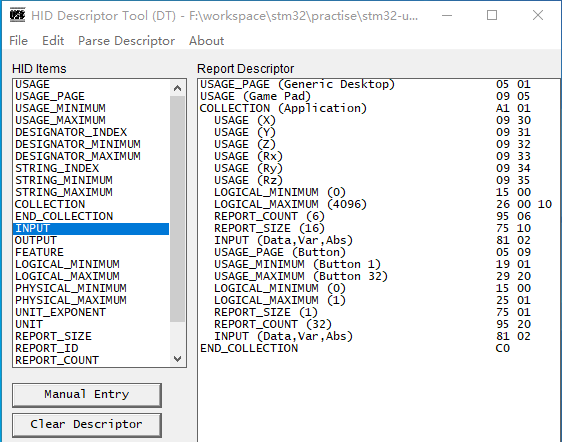
之后配置好时钟和其他设置，生成代码即可。

1. 我用的vscode+platformio开发环境，新建工程并打开导出的代码。Cubemx和platformio的文件夹结构不同，需要挪一下文件，主要是将Core/Src下的文件放到src下，将Core/Inc下的头文件放到include下，将USB\_DEVICE下的两个文件夹放到lib下。当然也可以在platformio.ini中配置添加这些文件的路径，不过我还是喜欢直接拖拽。

此时编译下载到板子应该是可以被电脑识别为hid设备了。

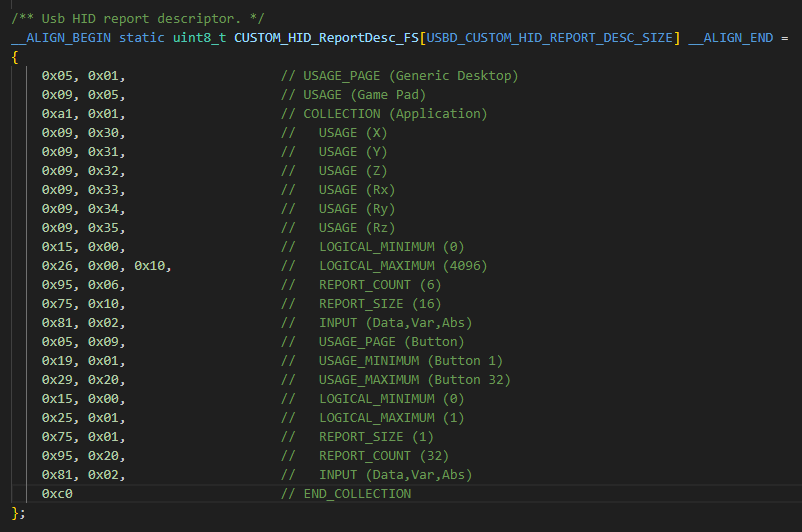


此时汇报描述符没配置，所以识别类别未知。下面就填充汇报描述符，这是hid进行通信的基础。填充这个结构体需要参考《[HID Usage Tables](https://usb.org/sites/default/files/hut1_5.pdf)》这个文件，这里定义了各种hid设备的用法。另外官方还提供了一个工具[Dt.exe](https://usb.org/sites/default/files/documents/dt2_4.zip)构建汇报描述符。

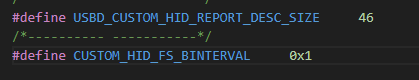


前三行和最后一行是必须要的，第一行表明是一般桌面应用，第二行指明设备是game pad类型，第三行的Application是规定必须有的。中间部分是实际的按键和旋钮，根据自己的实际需求定义。详细语法规则就不罗嗦了，可以参考《USB Complete\_ The Developer’s Guide 4th Edition》这本书。我演示的这个也不规范，不过实测可以用。

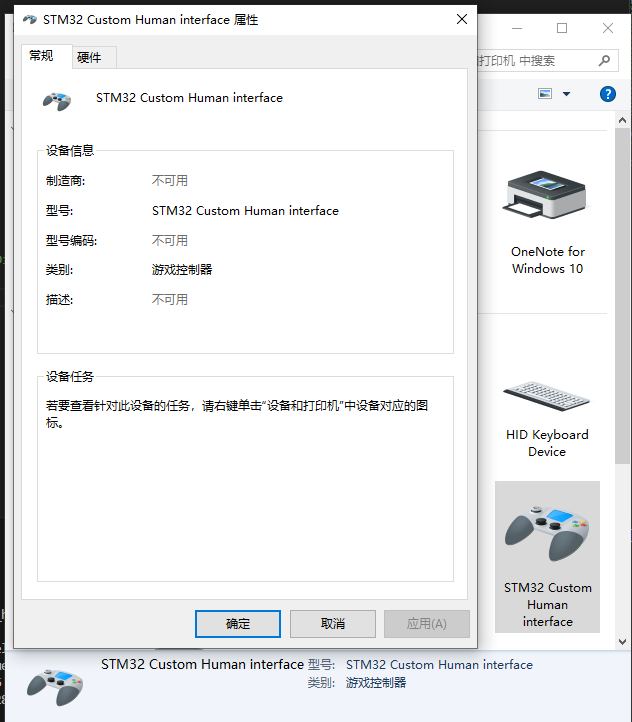
配置好后导出为头文件，就生成了我们需要的汇报描述符结构体。将结构体的内容复制到usbd\_custom\_hid\_if.c文件中的CUSTOM\_HID\_ReportDesc\_FS结构体中



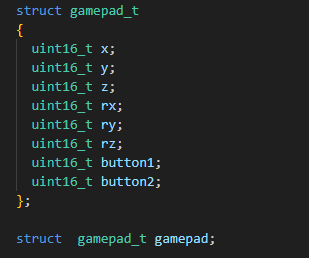
记得修改结构体大小的宏USBD\_CUSTOM\_HID\_REPORT\_DESC\_SIZE为实际结构体大小，不然会导致识别出错。这个宏下边的宏决定了汇报速度，想要汇报的快就要改小，最小为1。



到此，编译下载电脑应该可以正常识别为一个游戏控制器了。

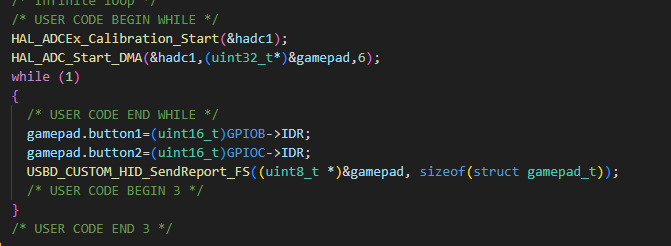


接下来就是汇报实时数据了，我们需要构建一个结构体，其内容大小和顺序要跟汇报描述符里边定义的一致。

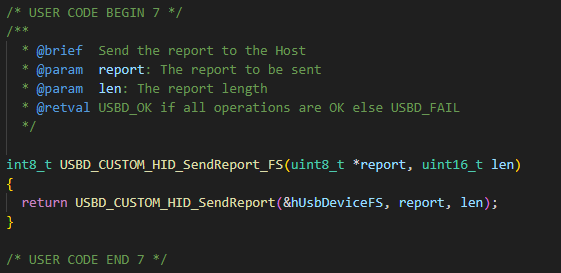


定义结构体要注意结构体对齐的问题，不然会出现汇报数据错位的问题，因为系统会自动对齐结构体成员，可能会添加额外的无用字节占位。

然后在主程序里开启ad转换并填充结构体，最后调用汇报函数。

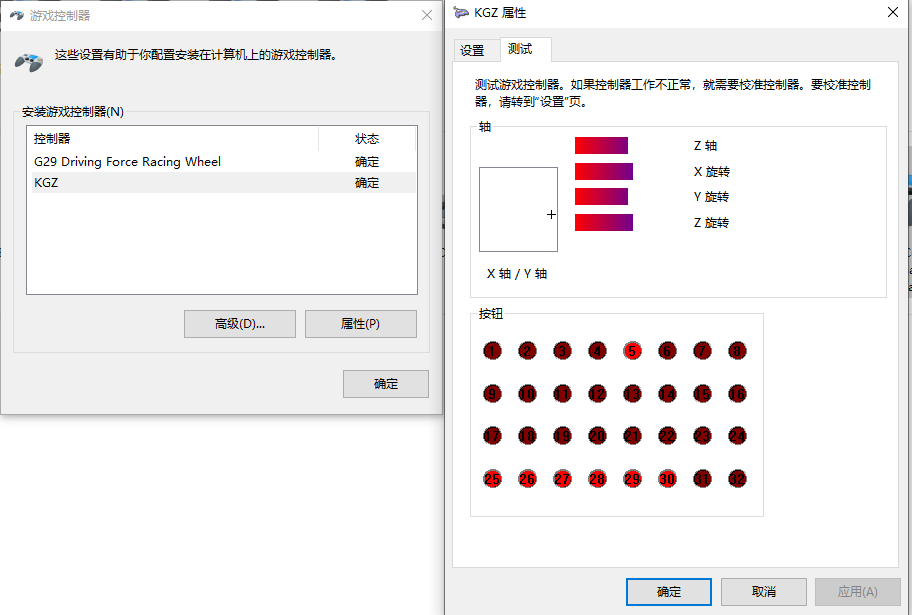


汇报函数在usbd\_custom\_hid\_if.c文件中最后，默认是注释掉的，要用的话需要取消注释，并将static关键字去掉，将函数在usbd\_custom\_hid\_if.h中声明一下就OK了。



到此，在游戏控制器中的属性就可以看到轴和按键的情况了。





我发现stm32的adc各个通道之间有影响，目前还没有找到解决办法。以上只是一个演示，要想实际应用还需要进一步优化，要根据自己实际的硬件情况修改引脚配置。