从0开始配置自瞄代码环境(不难，也就几个小时能配好的，部分截取coding2022版从0配置)

工具（后续会整合）:

Vscode1.71.2

Foxglov-studio-1.65.0(有库需要配置，详细看官网ros2部分)

Linuxqq

linuxSDKV2.1.0.31（迈德相机的配置，readme很详细，结合coding配置）（还有海康的，会提供相应代码启动，亲自尝试）

pigcha（科学上网）

cutecom (查看串口)

相机标定代码（camera\_caliberation）

配置环境：

C++环境配置（或者用clangd，参考coding）

Cmake（重点，需要长时间学习）

Eigen3

Ceres

Fmt(以上几个参考coding，fmt需要注意静态链接安装，cmake -DCMAKE\_POSITION\_INDEPENDENT\_CODE=TRUE ... 看readme，不然后续使用会报错)

Opencv-4.5.5（找博客，很详细，需要花时间）

（以上部分的环境配置好后，包括相机，可以使用2022版自瞄进行测试）

Ros-humble(需要换源，不然下载很慢)

ros-humble-serial-driver（问GPT详细配置）

ros2 nav2（不确定是否需要）

gazebo

camera\_info\_manager（代码使用时报无这个hpp文件的错误，问GPT解决，下载后重开vscode或者重启解决）

还有很多库忘了（包括可视化工具的使用，看官网，下面有链接），后续会进行修改，有新的nuc会让学弟亲自尝试，直到代码正常使用，启动过程中会肯定会有问题的，有问题就问。

Linux遇到的问题如下（很多博客有解决办法，一个不行多试几个）:

中文输入法

Deb的安装，使用dpkg

清华源

Vim的使用

等

代码使用注意事项

相机标定

目标：确定识别的目标距离准确度

过程：

（1）相机标定获取相机参数，方式可用ros或者matlab

Ros: 通过发布节点，启动代码对节点传输的图片进行标定

code： camera\_caliberation

文档：coding相机标定



迈德节点：ros2 run mindvision\_camera mindvision\_camera\_node

启动代码标定：ros2 run camera\_calibration cameracalibrator --size 8x6 --square 0.20 image:=/image\_raw camera:=/camera 其中8和6分别的长宽黑白格数量减1（即格子之间的点）

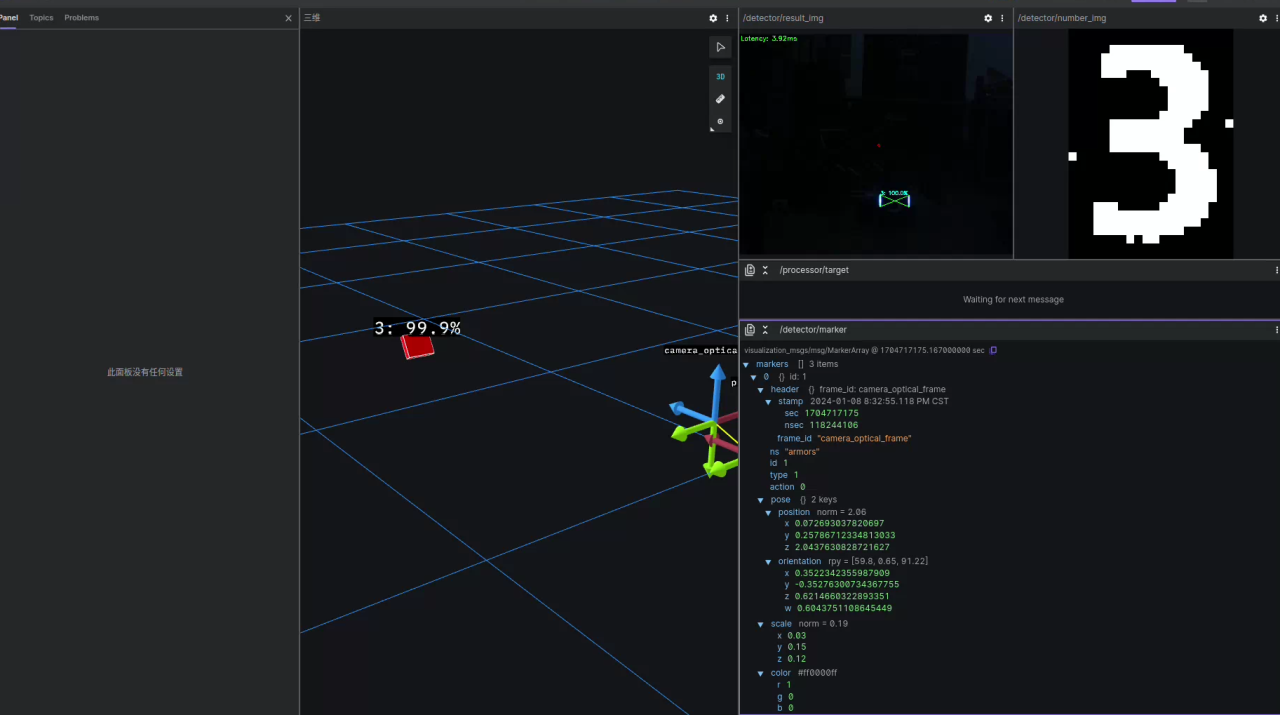
标定完后会获取一系列相机参数，在ros2\_mindvision\_camera中的camera\_info.yaml进行修改。

最后进行测距，代码启动：

ros2 run mindvision\_camera mindvision\_camera\_node

ros2 launch rm\_vision\_bringup no\_hardware.launch.py

ros2 launch rosbridge\_server rosbridge\_websocket\_launch.xml(Foxglove Studio，有两种方法启动, 链接：<https://docs.foxglove.dev/docs/connecting-to-data/frameworks/ros2/>)



通过相机测距与真实值进行对比，保证数据准确。

（2）matlab:暂无

串口

目标：串口数据接收正确

过程：

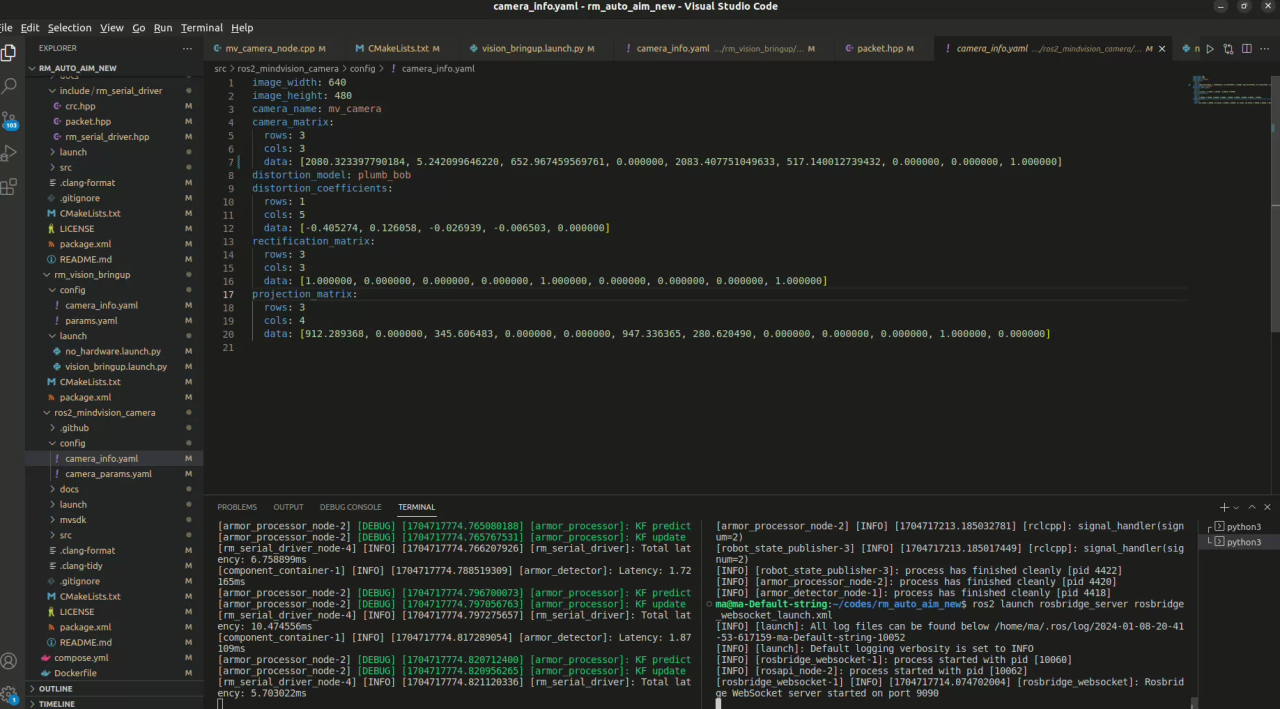
连接串口，蓝光表示已经接收数据，绿光表示已经发送数据。

首先查看是否存在串口（接了串口后）（ls /dev/tty\*） , 查看ttyACM0或者ttyUSB0，存在后记得赋权（sudo chmod 777 /dev/tty\*），使用cutecom进行查看是否接收正确。

正确接收并且换届配置好后，启动代码

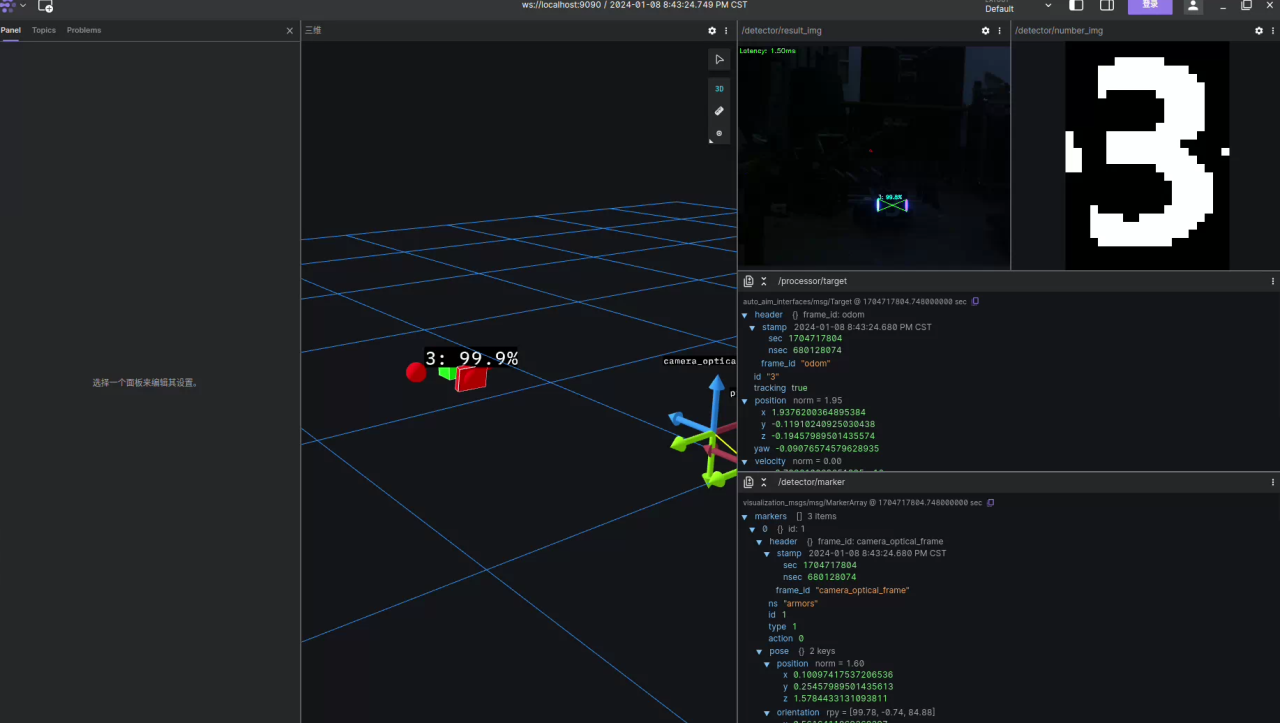
ros2 launch rm\_vision\_bringup vision\_bringup.launch.py

ros2 launch rosbridge\_server rosbridge\_websocket\_launch.xml



绿色打印表示识别到目标并且成功KFC预测

启动Foxglove Studio,



具体串口数据看rm\_serial\_driver

弹道解算

目标：保证弹丸能够精准击打目标

过程：

具体代码参照algorithm\_SolveTrajectory.cpp

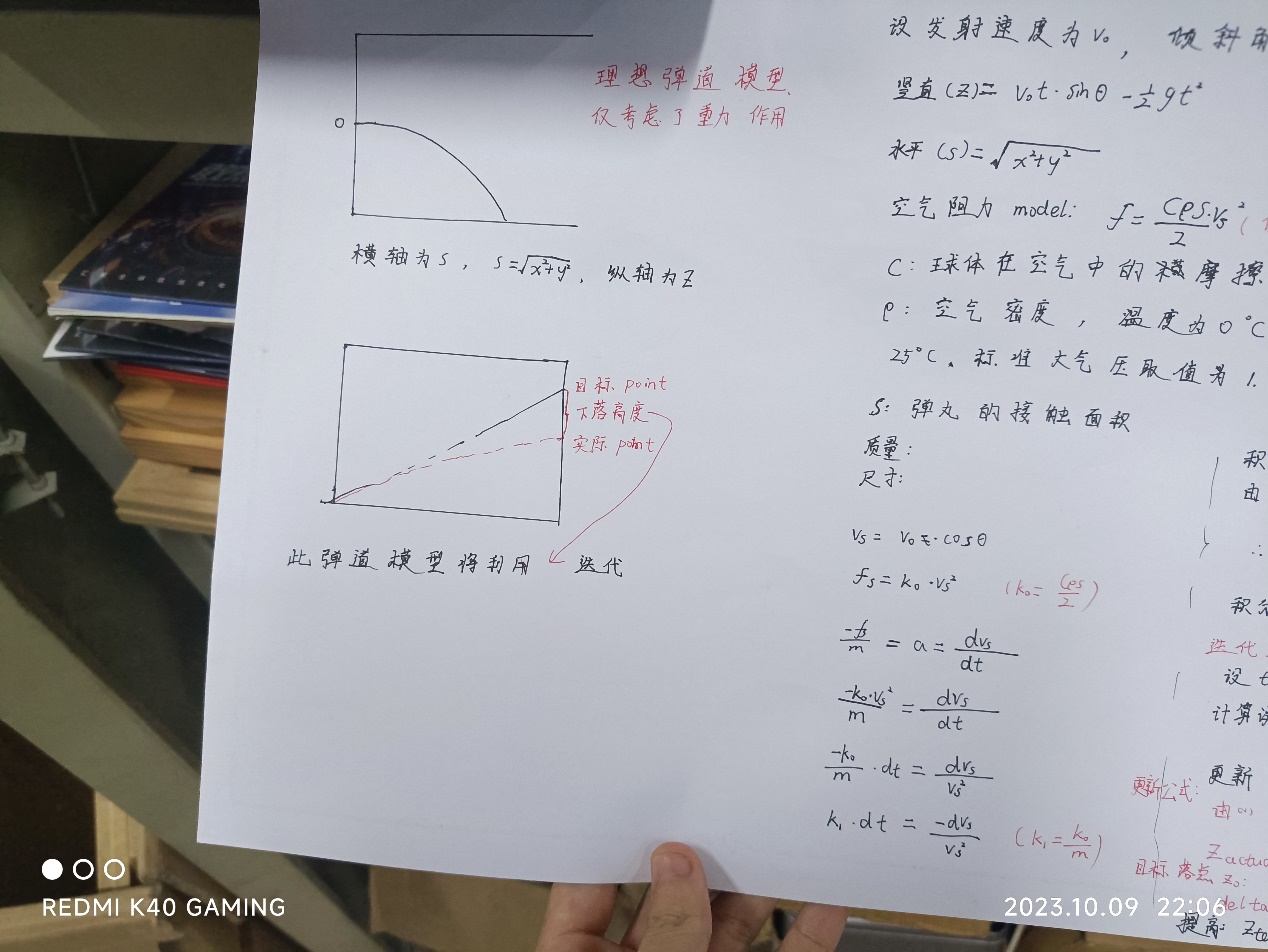
首先确保algorithm\_SolveTrajectory.h中的参数正确，包括弹道系数k，弹速v，偏置时间bias\_times, s\_bias和z\_bias。

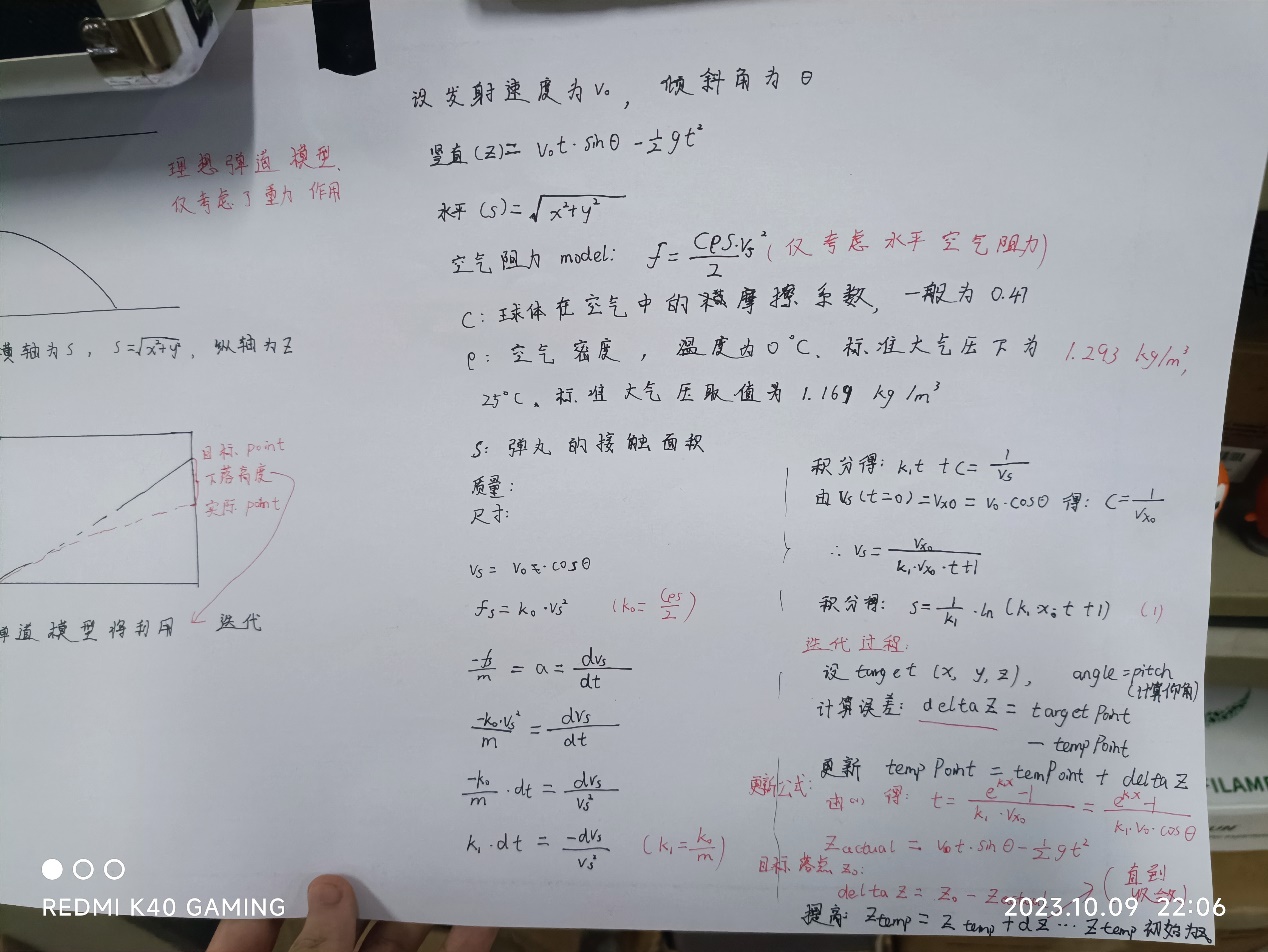
弄懂弹道解算的原理及其公式，参考小黑板的解算图，理解代码逻辑。

理解电控端弹道解算中的特别参数，如Gyro->Yaw\_angle是陀螺仪yaw轴值等。

理解最终解算的误差值的原理及其逻辑，重点是yaw。

调整静态补偿，保证误差值New\_Visual\_Yaw和New\_Visual\_Pitch在清零的情况下能够精确击打目标，包括远近高低，如果近距离打的中而远距离打不中的情况，要么就是相机标定出问题，要么就是陀螺仪飘了，要么就是弹道解算的逻辑或者是基本参数有问题，需要仔细看，统一套参数不是适配所有自瞄调参，需要调整出适合的参数。





陀螺仪检查

目标：保证pitch轴和yaw轴的数据尽可能准确，需要注意yaw轴或者pitch轴会不会相互影响，以及有没有飘了导致大幅度变动。

过程：

代码查看：ros2代码使用topic去查看，单独摆动yaw轴或者pitch轴，观察是否正常使用，不应该存在动pitch轴而yaw轴也在移动的情况或者在不动的情况下yaw轴持续变动的情况等，由于陀螺仪的特性（拉），需要长期进行检测。

原因：陀螺仪的数据准确会影响弹道解算的准确度。

自瞄使用

目标：电控端开启自瞄模式时可以正常使用（误差值清零的情况下精准击打，如果没有回到弹道解算那一步）

过程：

弹道解算（电控和视觉，最好都要懂逻辑，不能单方面的付出）

调PID（电控部分）

陀螺仪数据的融合，保证yaw轴和pitch轴数据准确（电控部分）

自瞄模式的准确使用（电控和视觉）

简洁版的，后续需要进行改进，有问题就问（肯定会有问题的，问题是发现的，只会越来越多），发信息也行，打电话也行，有求必应。