## 封皮

## 修改说明

## 一、环境搭建

### 1、安装ubuntu 20.04

### 2、配置aerocar

#### (1)先安装git

指令：sudo apt install git

#### (2)安装aerocar代码

在你指定的文件夹下安装aerocar代码

指令：git clone https://e.coding.net/ifccq/aerocar/AeroCar.git --recursive

【提示】

这一步可能出现有子模块无法下载的现象，无需担心

#### 运行bash脚本

进入根目录，运行下面指令，完善运行环境

指令：bash ./Tools/setup/ubuntu.sh

1. 更新团队仓库里的子模块

1、更新相关的子模块

(1)在项目里找到相关的子模块

(2)在源代码里添加相关子模块的仓库

(3)更新子模块

#### 6、重启计算机

#### 7、安装工具

安装GCC

指令：sudo apt-get install -y gcc-arm-linux-gnueabihf g++-arm-linux-gnueabihf

指令：sudo apt install make

指令：sudo apt install python3-pip

指令：sudo apt install ninja-build exiftool ninja-build protobuf-compiler libeigen3-dev genromfs xmlstarlet libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev python3-pip

### 3、安装ROS

参照官网教程：http://wiki.ros.org/cn/Installation/Ubuntu

输入命令

sudo apt install ros-noetic-desktop-full

遇到下面这样的问题

下列软件包有未满足的依赖关系：

ros-neotic-desktop-full :

依赖: ros-neotic-desktop 但是它将不会被安装

依赖: ros-neotic-perception 但是它将不会被安装

依赖: ros-neotic-simulators 但是它将不会被安装

E: 无法修正错误，因为您要求某些软件包保持现状，就是它们破坏了软件包间的依赖关系。

尽量不要使用sudo apt install aptitude下载aptitude 再用sudo aptitude install这个命令。

这个是会自动配置依赖问题，但有的时候他的方案，在安装一些依赖的同时会删除其他依赖，导致其他依赖丢失，使好多东西不能用了，上一次我就这么做了导致重装系统。

以上问题需要将软件和更新里的更新-订阅，设置为所有更新，然后关闭更新后再次输入命令即可。

### 4、安装QGC

参照官网连接：

<https://docs.qgroundcontrol.com/master/en/getting_started/download_and_install.html>

### 5、仿真环境搭建

1、安装仿真器

aerocar支持多种仿真器，包括Gazebo、jMAVSim、MavROS等。这里以安装Gazebo为例，可以使用以下命令安装：

sudo apt-get install gazebo9 libgazebo9-dev libopencv-dev

编译aerocar并运行仿真器

在aerocar目录下运行以下命令，编译aerocar：

cd AeroCar（进入工程根目录）

make px4\_sitl gazebo

## 二、开发流程

1、分支管理（分支模式、远程仓库、本地仓库、审核与合并）

2、代码测试（静态测试、单元测试、仿真测试）

3、下载固件

## aerocar代码编写调试

### 1、代码编写规范

在aerocar文件的根目录下运行./Tools/astyle/check\_code\_style\_all.sh，也即是在Tools这个文件夹这一层运行；

### 2、代码更改记录

### 3、仿真测试

## 代码的编写及管理

### 代码的编写

#### (1)根据自己的任务创建本地分支，使用以下指令：

git branch -b 分支名 //创建并切换到新建的分支上

#### (2)每次更新代码后需保存提交推送仓库（哪怕只是改动一行）

git add .

git commit -m “改动的内容介绍”

git branch --set -upstream -to = 仓库名/分支名 //第一次推送使用这个命令，后续不再使用此命令

git fetch //下拉 //除了第一次推送外，后续需下拉合并后再推送

git rebase //合并 //除了第一次推送外，后续需下拉合并后再推送

git push 仓库名 分支名

#### (3)代码编写完后在仓库创建合并请求

在CODING的项目中，选择“合并请求”标签页，点击“创建合并请求”按钮，选择需要合并的源分支和目标分支，填写合并请求的标题和描述信息，可以选择相应的合并请求的负责人和审核成员，最后点击“提交”按钮即可创建合并请求。

#### (4)审核合并请求

合并请求的负责人可以在合并请求中查看和审核合并请求，可以选择“同意”、“需修改”或“拒绝”等审核结果，同时可以添加评论和附件等信息，审核人员审核完成后，提交审核结果即可。

#### (5)代码审核

##### 创建代码审核任务

在CODING的项目中，选择“代码审核”标签页，点击“新建代码审核任务”按钮，填写代码审核任务的名称、描述、审核成员、审核分支等信息，点击“提交”按钮即可创建代码审核任务。

##### 提交代码审核请求

在代码审核任务中，点击“提交审核请求”按钮，选择需要审核的代码分支和提交的代码版本，填写审核请求的标题和描述信息，可以选择相应的审核成员和审核分组，最后点击“提交审核请求”按钮即可将审核请求提交给审核人员。

##### 审核代码请求

审核人员可以在代码审核任务中查看和审核代码请求，可以选择“同意”、“需修改”或“拒绝”等审核结果，同时可以添加评论和附件等信息，审核人员审核完成后，提交审核结果即可。

##### 审核完成后的操作

如果审核结果为“同意”，则可以选择合并代码到指定分支或删除审核请求；如果审核结果为“需修改”，则开发人员需要修改代码后重新提交审核请求；如果审核结果为“拒绝”，则需要重新进行代码编写和审核。

##### 合并代码

如果审核结果为“同意”，则可以点击“合并”按钮将代码合并到目标分支，同时可以选择是否保留源分支，是否启动自动化测试等选项；如果审核结果为“需修改”，则开发人员需要修改代码后重新提交合并请求；如果审核结果为“拒绝”，则需要重新进行代码编写和审核。

### 2、代码管理

#### （1）注册CODING账户并创建项目

首先需要注册一个CODING账户并创建一个新的项目。进入CODING官网（https://coding.net/），点击注册按钮进行账户注册，完成注册后进入CODING控制台，点击新建项目按钮创建一个新的项目。可以设置项目名称、描述、可见性、权限等参数。

#### （2）代码托管

将代码提交到CODING中进行托管。可以使用CODING提供的Git、SVN等版本控制工具，管理代码的版本、分支、标签等。下面以Git为例介绍代码托管的步骤：

在控制台页面上点击项目名称进入项目主页，选择版本管理 -> Git仓库 -> 克隆，复制项目的SSH地址。

在本地计算机上安装Git客户端，使用Git命令行工具或者图形化界面工具（如SourceTree、GitKraken等）克隆远程仓库到本地。可以使用以下命令进行克隆：

$ git clone <CODING项目SSH地址>

在本地进行代码的开发和修改，并使用以下命令提交代码：

$ git add .

$ git commit -m "提交信息"

$ git push origin <分支名称>

这里的分支名称可以是主分支（如master），也可以是自己创建的分支。提交代码后可以在CODING项目页面的版本管理界面查看提交记录、分支情况等。

#### （3）任务管理

CODING提供了一套任务管理系统，可以方便地跟踪和管理项目中的各个任务，包括需求、Bug、任务等。

进入CODING项目页面，选择任务管理 -> 任务，点击新建任务按钮创建一个新的任务，可以设置任务类型、任务描述、任务负责人、任务优先级等参数。

在任务列表页面，可以看到所有的任务列表，可以按照不同的状态（如待处理、处理中、已完成）进行筛选。点击任务名称可以进入任务详情页面，进行任务的编辑、指派、跟踪等操作。

#### （4）协作工具

CODING提供了一系列的协作工具，可以方便地进行团队协作和交流。

讨论：可以在项目页面上选择讨论板块，创建新的讨论主题，并进行讨论。

文档共享：可以在项目页面上选择文档板块，创建新的文档，对文档进行编辑、共享和版本管理。

在线编辑器：可以在项目页面上选择代码编辑器板块，使用CODING提供的在线代码编辑器进行代码编辑和修改。

代码审查：可以在项目页面上选择代码审查板块，进行代码审查和评审。

#### （5）持续集成

CODING提供了一套持续集成系统，可以方便地进行自动化构建、测试和部署等操作，保证代码的质量和稳定性。

在CODING项目页面上选择持续集成板块，创建新的持续集成任务。

配置构建、测试和部署的步骤和流程，可以选择使用CODING提供的预设模板，也可以自定义。

执行持续集成任务，查看构建和测试结果，如果全部通过，可以选择自动部署或手动部署。

#### （6）安全管理

CODING提供了一套安全管理系统，可以保证代码和数据的安全性和隐私性。

在CODING项目页面上选择安全管理板块，进行安全配置和设置。

可以设置访问权限、敏感数据的保护、代码审查等功能，保证项目的安全性。

#### （7）自查

为了保证代码质量，可以定期进行自查和审核。

在CODING项目页面上选择代码审查板块，进行代码审查和评审。

可以查看代码的变更记录、提交信息、代码质量指标等，对代码进行评审和反馈。

## 编译烧写程序源代码

### 1、在程序根目录下打开终端，输入：make clean

### 2、输入：make cuav\_nora\_default

### 3、下载程序的步骤：连接飞控，再输入：make cuav\_nora\_default upload