**2017 RoboMaster夏令营**

**技术报告**

**2017.8**

**目录**

**[目录](#_Toc490648079)** [2](#_Toc490648079)

[1 机械部分 5](#_Toc490648080)

[1.1设计动机 5](#_Toc490648081)

[1.2设计需求 5](#_Toc490648082)

[1.3设计方案 5](#_Toc490648083)

[1.3.1 底盘机构 5](#_Toc490648084)

[1.3.3 机械爪设计 6](#_Toc490648085)

[1.4方案的优点与不足 7](#_Toc490648086)

[2 嵌入式部分 8](#_Toc490648087)

[2.1任务分析 8](#_Toc490648088)

[2.2嵌入式工程模块划分 8](#_Toc490648089)

[2.3云台控制部分 8](#_Toc490648090)

[2.3.1射击角度定位 8](#_Toc490648091)

[2.3.2拨弹模块 9](#_Toc490648092)

[2.3.3扫描状态控制 9](#_Toc490648093)

[2.4底盘控制部分(全场定位设计) 9](#_Toc490648094)

[2.5 难点与不足 10](#_Toc490648095)

[2.5.1待解决问题 10](#_Toc490648096)

[2.5.2嵌入式技术总结 10](#_Toc490648097)

[3 算法部分(无人机) 10](#_Toc490648098)

[3.1 开发环境介绍 10](#_Toc490648099)

[3.1.1 硬件环境 10](#_Toc490648100)

[3.1.2 算法环境 10](#_Toc490648101)

[3.2整体技术方案概述 11](#_Toc490648102)

[3.2.1技术原理介绍 11](#_Toc490648103)

[3.3 算法整体框架设计 12](#_Toc490648104)

[3.4 算法功能模块说明 13](#_Toc490648105)

[3.4.1 Guidance位姿读取 13](#_Toc490648106)

[3.4.2位置环PID控制 14](#_Toc490648107)

[3.4.3逻辑控制 15](#_Toc490648108)

[3.4.4取弹机构控制 16](#_Toc490648109)

[3.5测试结果 16](#_Toc490648110)

[3.6 可优化方案 16](#_Toc490648111)

[4 算法部分（战车部分） 17](#_Toc490648112)

[4.1 开发环境介绍 17](#_Toc490648113)

[4.1.1 硬件环境 17](#_Toc490648114)

[4.1.2 算法环境 18](#_Toc490648115)

[4.2整体技术方案概述 20](#_Toc490648116)

[4.2.1技术原理介绍 20](#_Toc490648117)

[4.3 算法功能模块说明 21](#_Toc490648118)

[4.3.1定位算法 21](#_Toc490648119)

[4.3.2导航算法 21](#_Toc490648120)

[4.3.3跟踪射击 23](#_Toc490648121)

[4.3.4单兵逻辑 24](#_Toc490648122)

[4.3.4多兵作战 25](#_Toc490648123)

[4.4测试结果 25](#_Toc490648124)

[4.5 可优化方案 25](#_Toc490648125)

[5 夏令营感想、总结 27](#_Toc490648126)

# 1 机械部分

## 1.1设计动机

简单快速稳定的完成子弹的抓取以及交接任务，保证波轮盘不卡弹。

## 1.2设计需求

1. 取弹机构设计制作
2. 接弹机构设计制作
3. 拨弹机构优化

取弹机构与接弹机构优先级最高，拨弹机构优化其次。

取弹机构设计与接弹机构的设计并行，时间大约用了20天，拨弹机构的优化在之后，时间大约用了13天。

## 1.3设计方案

### 1.3.1 底盘机构

如激光雷达、摄像头等传感器如何安放

激光雷达放置的位置不变，摄像头安装在枪管前段，用一个镜头夹夹在枪管上。

有悬挂和无悬挂的区别，为什么要这样设计？

前轮联合悬挂后轮没有悬挂，对于本次比赛来说这样的悬挂配置决定了保证了激光雷达平面的水平，使得激光雷达能够更为稳定的工作。本次比赛的场地较为平整，没有什么不规则凸起，相对来说独立悬挂没有特别大的意义1.3.2 云台设计

云台采用下供弹的方式，将拨盘和预装载区垫高，使得预装载的子弹整齐有序而且更多，再在拨盘顶部加上一个搅拌装置，使得拨弹机构不出现空弹和漏弹的情况，拨弹机构就很少出现卡顿的情况。在拨弹机构底面上贴满铁氟龙，减小拨轮、拨弹机构与子弹间的摩擦，有效防止了子弹卡到拨轮下方。

提高子弹的一致性，采用了使用两个硬度不一的摩擦轮，使得子弹发射时的偏移保持在同一方向上。

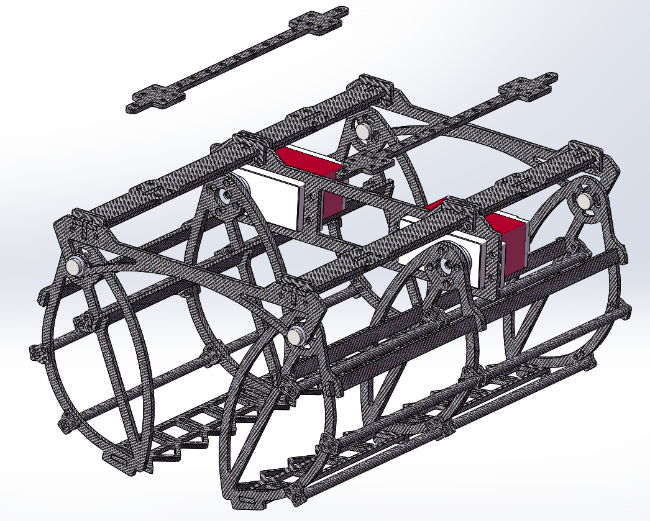
### 1.3.3 机械爪设计

飞机抓球机构如何设计

官方给予每队一台M100作为空中机器人，根据官网数据查得M100的自身重量为2355g，最大起飞重量为3600g，极限起飞重量为4500g（估测），折算出可供我们使用的重量为1245g。飞机需要挂在的物品有下视guidance（约43g），guidance主控（约64g），VBUS线缆（约11.6g），妙算 Manifold（约180g），连接线等重量这算为100g，则留给取弹机构以及子弹的重量仅为846g。设计装载200枚子弹，每粒子弹重量为2.6g，则子弹总重为520g，机构重量仅为326g。

|  |  |
| --- | --- |
| 对称电机轴距 | 650 mm |
| 重量（含TB47D电池） | 2355 g |
| 重量（含TB48D电池） | 2431 g |
| 最大起飞重量 | 3600 g |
| 扩展架重量 | 45 g |
| 电池仓重量 | 160 g |

通过前期测试的机械爪与涵道方案对比，涵道吸球方案优点：吸子弹时的适应能力强，全部子弹都能被吸取上； 缺点：重量较大，吸取子弹数量不可控制等。机械爪方案优点：取弹速度快，质量较涵道轻，子弹数量较为好控制；缺点：平台上有一层子弹无法被吸取，会出现卡舵机的情况。主要在重量方面涵道方案无法得以有效控制，最后选择了又轻又快的机械爪方案。

机械爪方案在前期一直坚持使用单舵机加齿轮传动的驱动方式，但是转轴的扭转强度不够，打印件直接扭断了，就放弃了使用单舵机加齿轮传动的驱动方式，转而使用两个舵机来驱动。从一代机械爪开始使用碳板加碳管拼接的结构，保证强度又能保持质量轻。顶部使用部分弹簧悬架的结构，能够抵消机械爪活动是在纵向方向上的偏移，同时又能给机械爪提供足够的压紧力，防止机械爪抓空。

最后实测效果很好，一直用到比赛最后。

陆地机器人和空中机器人对接的机构怎么设计的

采用伞结构作为伸展机构，伞结构的主要特点是能够充分收展，展开是面积可以达到很大，而且能够向四面八方展开，收起来面积很小空间很小，正好适用于陆地机器人和空中机器人对接的机构。

## 1.4方案的优点与不足

方案优点：能够稳定快速的完成交接任务，结构形式新颖，做工精致。

不足：设计的机构细节很复杂，加工制作流程较为长久。

# 2 嵌入式部分

## 2.1任务分析

此次夏令营的主要的任务方向在于识别与决策，识别任务主要是飞机识别己方车辆，战车识别装甲板、识别环境并进行定位，决策任务主要是不同状态下的路径规划，单兵作战决策。嵌入式的主要任务为对目标点的精确打击、底盘运动的驱动与反馈、对自身血量弹药量监测并决策。

## 2.2嵌入式工程模块划分

基于提供的工程模板，我们将任务划分为多块：

1) 云台控制

2) 底盘控制

3) 上下位机通信

4) 裁判系统数据读取

5) 状态机决策

其中状态机决策分别被并入云台控制、底盘控制的线程中，没有划分单独的线程**。**

## 2.3云台控制部分

### 2.3.1射击角度定位

角度定位模块我们用了双环pid控制。上位机视觉提供的xy坐标，经过解算得到角度偏差，与当前角度求和，作为位置环输入。由于视觉算法和硬件的限制，采集的频率不高，并且具有滞后性，尝试过降低位置环频率与视觉算法同步、使用编码器历史数据与视觉采集数据进行解算等方法提高输入精度，均因为震荡而失败。最终方案采用视觉数据以较小的比例与当前角度求和，消除了震荡，并且动态性能也较好。我们还根据检测球速、敌方距离，对pitch轴瞄准做了矫正，提高了命中率。最终在5米范围内，定点射击命中率达到80%，移动情况下也能有30%命中率。

### 2.3.2拨弹模块

我们采用了双环pid的拨弹控制，可直接控制射击子弹数。我们改进了防卡弹的机械设计，增加挡片长度，使拨弹轮可以回退，当有卡弹情况出现时可用超时等待检测，检测到后进行回退，再继续射击。我们比赛时出现过多次卡弹情况，但由于回退机制的存在及时恢复战斗力，没有因此造成丧失战斗力情况。我们还做了空弹检测，通过射击时实时检测射频检测是否还有子弹，并作出逃跑规划。

### 2.3.3扫描状态控制

当失去敌方目标时，云台会持续循环扫描，扫描模式下云台pitch轴固定，yaw轴以1秒的频率左右往复运动，为摄像头提供视野。yaw轴运动以时间的三角函数作为位置环输入，可以做到平滑精准的扫描。

## 2.4底盘控制部分(全场定位设计)

对于全场定位我们尝试了两种方式，第一种为编码器定位+uwb融合的的方式，第二种为上位机位置控制，下位机速度控制。对于第一种，下位机通过四轮电机自带的编码器解算平移的坐标，再通过陀螺仪获取角度信息，以固定频率积分即可得到当前的相对位置，uwb数据读取后使用互补滤波进行融合，消除积分误差，再通过位置闭环进行位置控制，到达上位机制定位置。对于第二种，上位机进行位置控制并通过激光雷达与下位机信息融合进行定位。通过测试和比较，第二种对于路径规划方面有明显优势，最终采取第二种。

## 2.5 难点与不足

### 2.5.1待解决问题

没有对敌方移动速度作预测，由于识别、控制的滞后，敌方横向移动时命中率很低，尝试做过预测但效果很差，只能放弃。

## 2.5.2嵌入式技术总结

在这次比赛中我们遇到了不少问题，但都在不断想方设法解决，虽然不能做到完美但我们一直在不断测试改进，一点点提高性能，虽然最终成绩没有达到预期但我们依然很有收获。

# 3 算法部分(无人机)

## 3.1 开发环境介绍

### 3.1.1 硬件环境

使用了M100无人机，搭载Manifold妙算作为机载计算机，搭载guidance视觉传感导航系统用于室内定位。另外，安装了一个官方提供的摄像头模组作为下视相机，用于识别和辅助定位。其中，摄像头模块焦距为2.8mm，保证无人机有较大的视角。

取弹机构方面，小组赛使用涵道吸弹的方案，联盟赛使用机械爪抓弹。

### 3.1.2 算法环境

1.机载计算机manifold运行ubuntu 14.04系统，在ROS上搭建机器人框架进行程序开发。ROS的搭建过程可参考：

wiki.ros.org/indigo/Installation/UbuntuARM。

注意manifold应安装ROS（indigo）for Ubuntu ARM

2.几个比较重要的ros包：

dji-sdk/Onboard-SDK-ROS：通过串口与M100的A1飞控通信，读取飞控信息并对无人机进行控制。（<https://developer.dji.com/onboard-sdk/>）

dji-sdk/Guidance-SDK-ROS：读取guidance数据。

Usb-cam & image-proc：读取摄像头图像以及完成图像的畸变矫正。

## 3.2整体技术方案概述

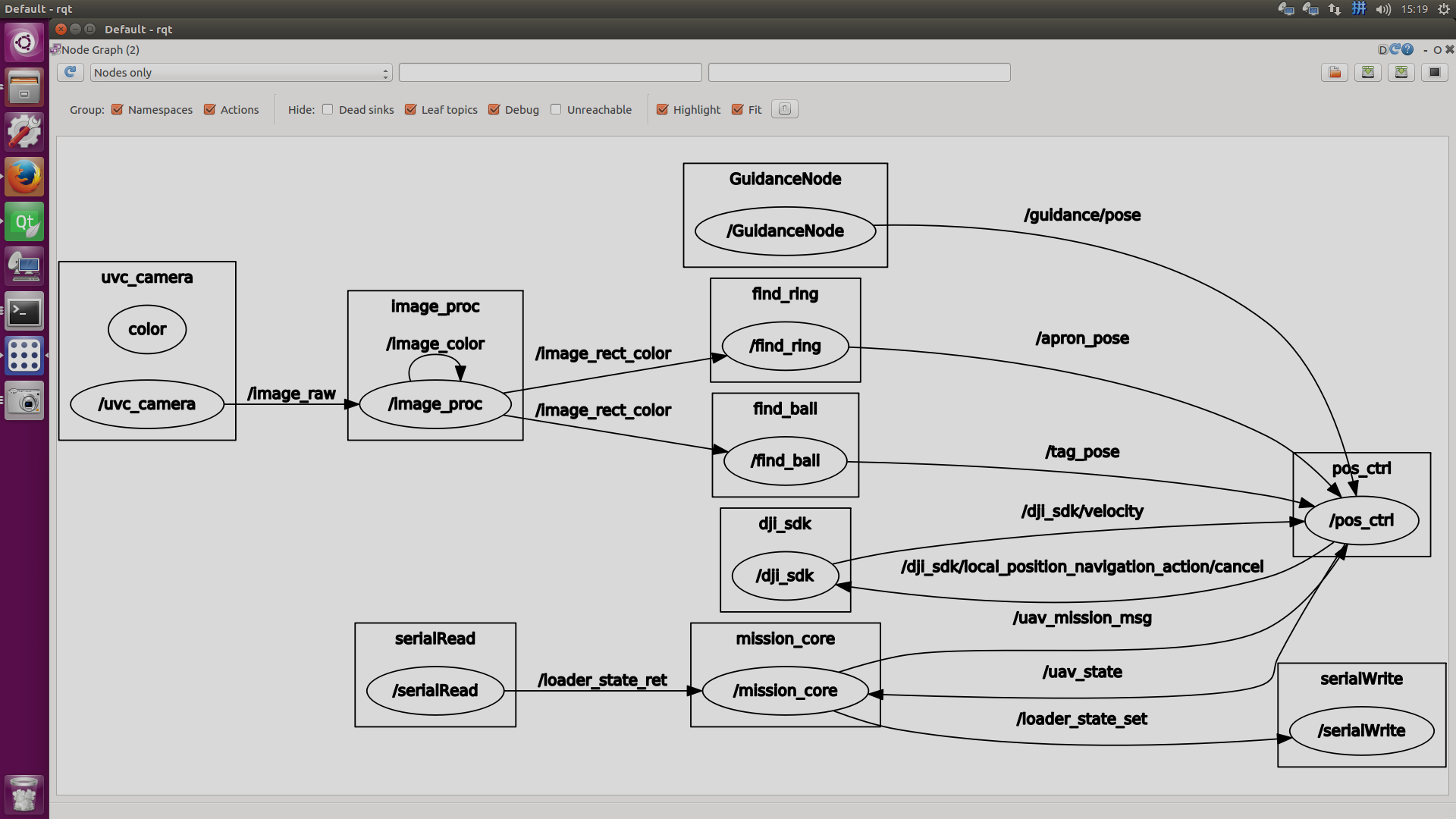
### 3.2.1技术原理介绍

本项目中，无人机需要自主取弹，完成起飞，之后以空投的形式完成对步兵车的补给，完成补给后需要自主返回并降落。因此，任务的关键点在于如何控制无人机精确地投弹以及降落。

无人机的位置控制是该部分的核心，此处使用PID控制器对位置进行控制。其反馈来源于guidance反馈的无人机全场位姿或下视相机反馈的无人机相对于停机坪/步兵车的位置；其输出的控制量为无人机的速度（x,y,z,yaw），通过Onboard-sdk的velocityAndYawRateCtrl接口将指令发给飞控;位置环的控制周期为20Hz。

取弹机构控制方面，使用串口与取弹机构通信，通过状态查询/切换的方法控制取弹机构。

## 3.3 算法整体框架设计

系统的ROS节点图如下,其中：

* GuidanceNode用于读取guidance数据，获取无人机的位姿信息。值得注意的是， Guidance-SDK-ROS中并没有提供读取无人机位姿的接口，需要自己写一个读取发布motion消息的发布器。/guidance/pose消息类型为PoseStamped，频率为50Hz
* uvc\_camera获取相机图像后，通过Image\_proc完成畸变矫正，之后发布给find\_ring节点（识别停机坪）和find\_ball节点（识别步兵车上的标志物），这两个节点发布类型为PoseStamped的消息/arpon\_pose & /tag\_pose——无人机相对与标志/停机坪的位置。频率30Hz。
* mission\_core读取无人机状态，指定无人机当前任务。并且通过串口节点serialread&serialwrite控制取弹机构。
* pos\_ctrl为系统的核心节点，接收guidance和摄像头反馈无人机位姿，完成pid控制，通过dji\_sdk节点操控无人机。

比赛流程的一次取弹投弹流程如上：

* 由于guidance的全局位置会漂移，每次起飞的时候，利用停机坪作一个矫正。
* 飞机飞到补弹点后，利用相机图像作精确的定位，悬停在步兵车上方后投弹。

## 3.4 算法功能模块说明

### 3.4.1 Guidance位姿读取

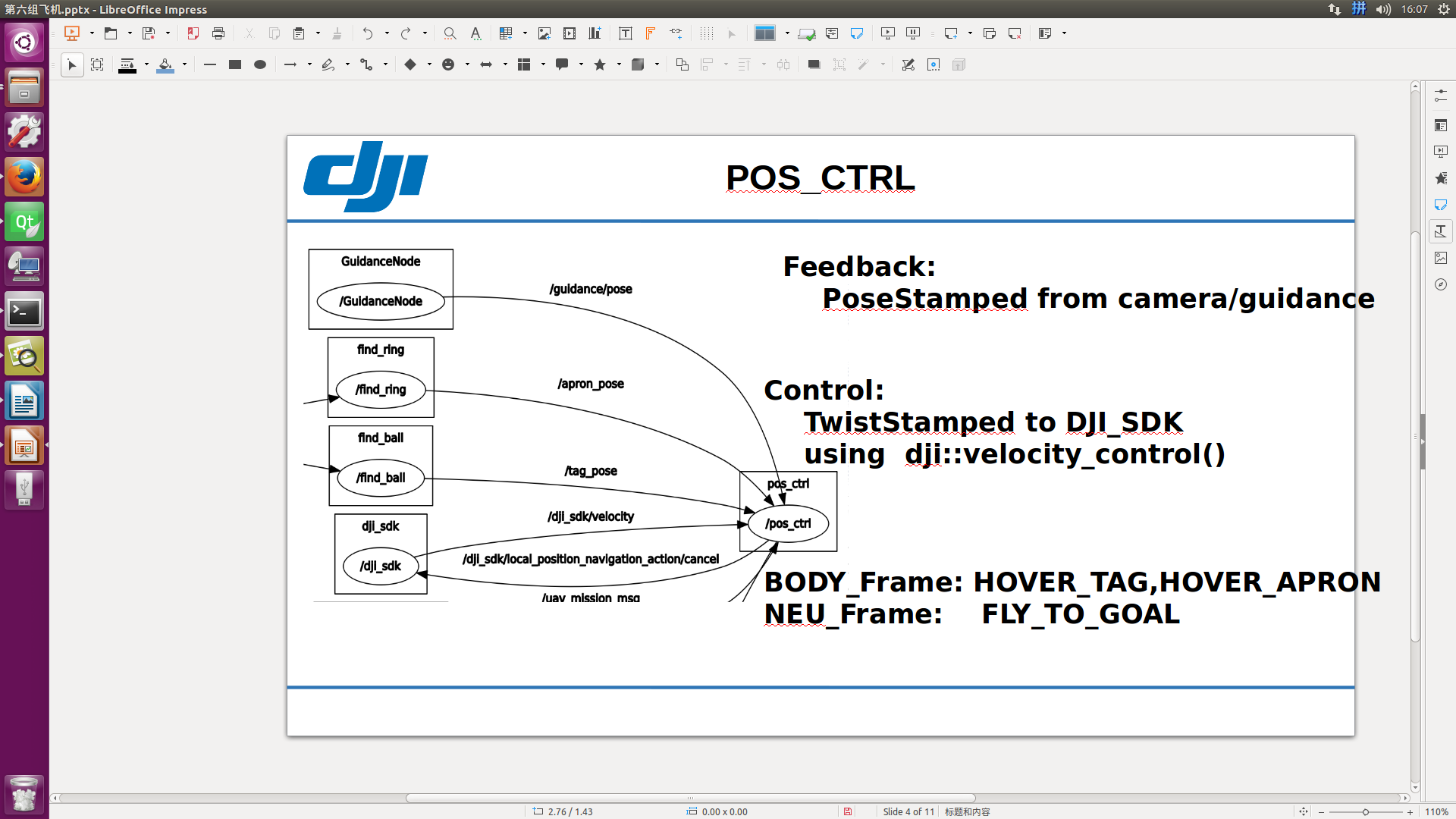
代码如上。读取了全局位置和四元数表示的姿态。

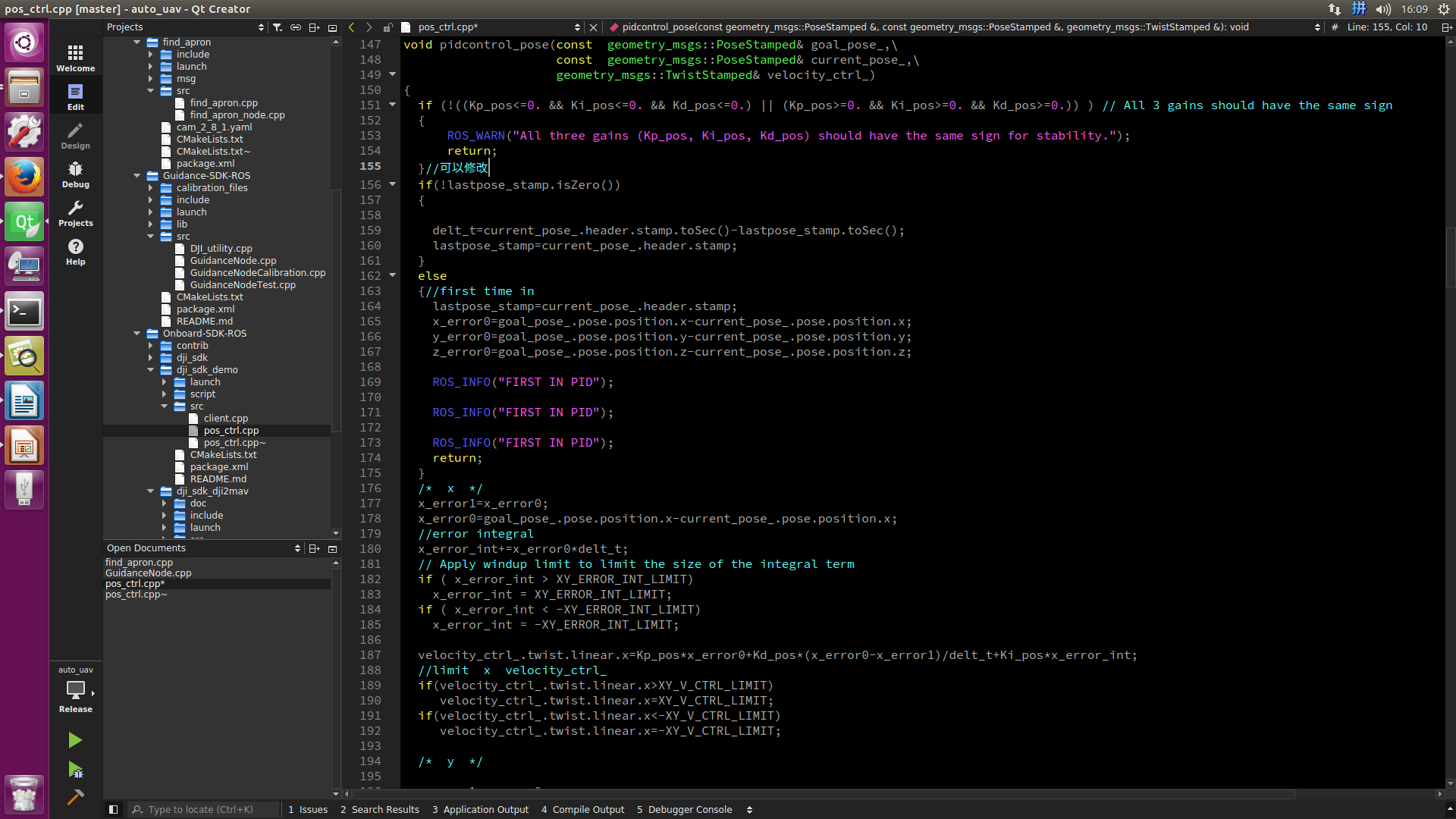
### 3.4.2位置环PID控制

该部分实现于Pos\_Ctrl节点内。主要的函数为pidcontrol\_pose，该函数接收两个带时间戳的姿态变量，一个为goal\_pose\_,一个为来自反馈的current\_pose\_，输出为Twist类型的速度控制量velocity\_ctrl\_。

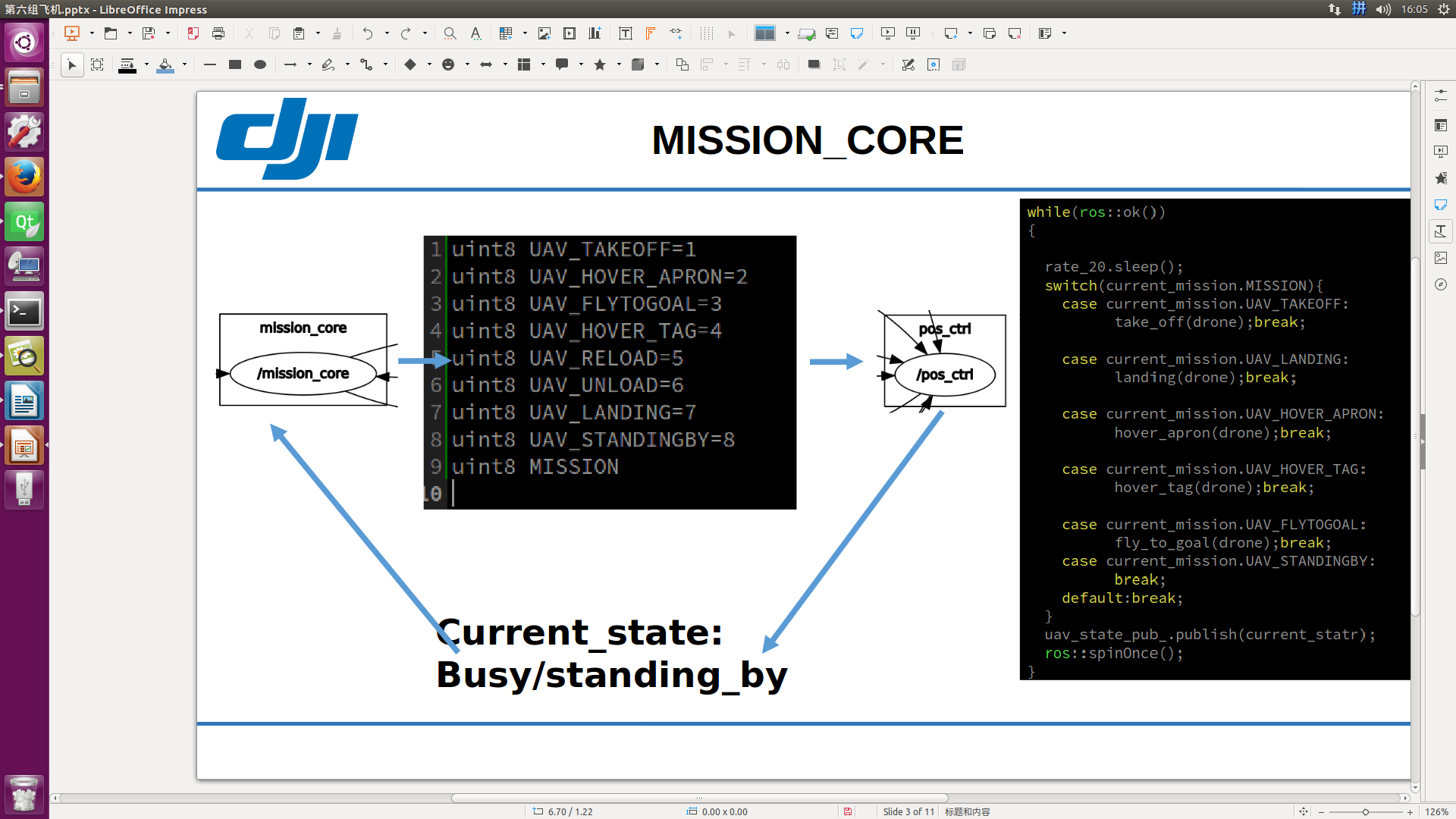
在pid控制算法的实现中，velocity=Kp\*error0+Kd\*(error0-error1)/delt\_t+Ki\*error\_int。

其中：

* delt\_t取自两次反馈姿态的时间戳之差，这样如果图像出现丢帧使得姿态反馈丢帧，控制器依然可以工作良好。
* 输出做了限幅，避免控制量过大引起炸机。
* 机体坐标系和NEU坐标系下的位置控制复用该函数，因此使用前要调用pid\_reinit以清空历史误差error和误差积分error\_int。‘
* 使用velocity\_control\_body和velocity\_control\_NEU给飞控发送机体坐标系和NEU坐标系下的速度控制量。在本系统的位置控制中，步兵车标志上方悬停和停机坪上方悬停使用的是机体坐标系，飞往目标点的位置控制为NEU坐标系。

下图是pid控制函数的部分代码：

### 3.4.3逻辑控制

Mission\_core节点用于掌控无人机当前状态并给无人机发布任务。

无人机的任务类型主要有：

STANDINGBY/HOVER\_APRON/TAKE\_OFF/HOVER\_TAG/RELOAD/UBLOAD/FLYTOHOME/FLYTOGOAL/FLYTOORIGIN/LANDING

Pos\_ctrl节点中，以20hz的频率读取无人机任务并执行，完成标志物、停机坪上方的悬停，以及到目标点的飞行。

### 3.4.4取弹机构控制

通过串口发送状态指令，接受反馈。其状态转换如下：



## 3.5测试结果

系统的关键在于位置控制环，需要多次调整pid参数以取得比较好的控制效果。本系统在比赛中工作稳定，位置控制精准。

## 3.6 可优化方案

1.无人机的投弹的目标位置可以由步兵车来发布，由于本次夏令营无线网络条件比较差，我们采用的是指定目标区域的方法。优化方案可以由步兵车发布位置，无人机导航过去。

2.如果加入敌车识别的模块，可以完成侦查敌车的功能。

# 4 算法部分（战车部分）

## 4.1 开发环境介绍

### 4.1.1 硬件环境

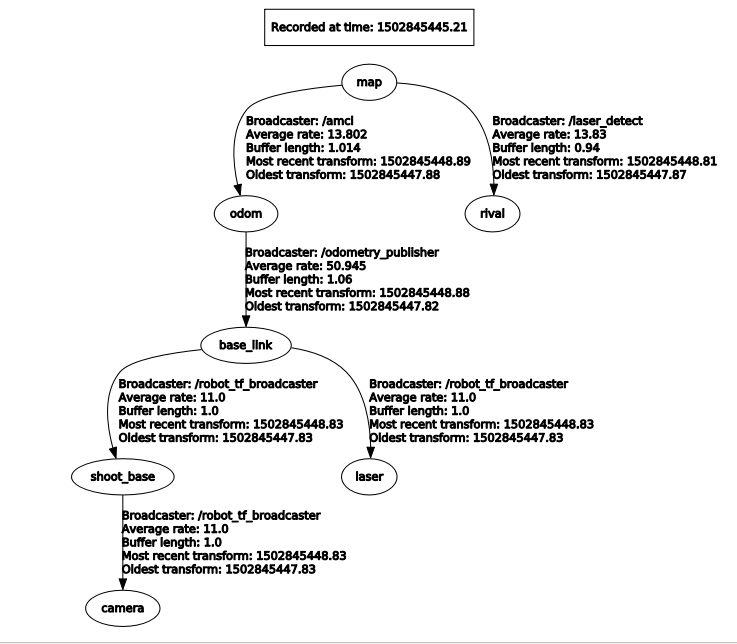
计算设备、传感器：

系统先前使用的为TX1,因为TX1电气损坏的原因，性能不稳定，后期使用的为i7NUC。传感器为标准配置：思岚R2激光雷达，最远探测有效距离可达8米;卷帘门CMOS配6mm长焦镜头，可增大视觉检测的有效距离，但也因此视角较小，通过云台摆动弥补这一缺陷。

硬件框架：

激光雷达硬件上是一个CP210X模块，与TX1或者PC通信。因为使用了NUC，不再有TX1上单板集成的串口功能。需要增加一个USB转串口模块作NUC与信仰板之间的通信，尝试过使用CP2102模块，但驱动不太兼容，无论是在NUC还是在笔记本电脑上只能接收，不能发送，后来使用CH340解决了不能发送的问题。

整个系统的坐标系：



### 4.1.2 算法环境

最开始嵌入式底层做了比较多的代码重构，造成上下层无法正常的对接。一部分主要是坐标系朝向的问题：

激光雷达的官方手册及源码文件夹中的附图都明确了坐标系顺时针为正，实际上SDK对数据做了反向处理，并将400点的激光雷达扫描数据通过角度补偿缩放至-180～180度逆时针，每一度一个读数。

最开始官方给的底盘坐标系有多个版本，其实坐标系的设定只要保证和ROS一样，采用右手系就可以。另外最初使用的单轴陀螺仪测量带宽偏小，动态响应慢，造成地盘快速转动时测量不准，嵌入式较难debug小车运动过程中的状态，将角度信息转化为角度制打印出来，才定位至这一硬件故障。

官方的上层串口协议做了帧头、帧尾、帧长的校验，出错的概率较小，但是它要求下层发送频率和上层读取linux串口缓冲区的频率要保持一致。目前的配置为一个数据包36字节左右，115200的配置，数据发送时间小于3ms，但如果在3ms之内读取数据就易造成同步出现错误。上下层以50Hz的速率通行，同步出错的概率小于15%，因为最开始异步的读取了一次数据做角度的初始化，之后数据同步的压力变小。但是往往存在角度初始化不成功的现象。

将代码移植到NUC上之后，发现数据同步的压力急剧增加，往往是初始时数据接受正常，之后丢包的几率越来越大，怀疑是NUC自动升频，无法正常同步，之后将串口协议做了分包聚合，抛弃了同步的机制，才解决这一问题。

另外官方版本的代码上层读取串口的节点不管是否有新数据更新，一直以固定频率发送，掩盖了实际可能存在的丢包情况。将代码更改为只有读取到新数据才发布的方式。

## 4.2整体技术方案概述

### 4.2.1技术原理介绍

定位部分：

采用slam方法（gmapping）建立地图。在生成完毕的地图上使用AMCL方法做定位。

导航部分：

根据生成的地图维护一个代价地图，代价地图采用inflation、voxel等层，可以做到实时更新周围障碍物。全局路径规划在代价地图上做Dijkstra算法，搜索Potential区域，采用梯度下降的方式在Potential中构建一条路径，采用角度线性插样的方式得到每一导航点的姿态信息。局部路径规划使用TEBlocalPlanner，修改全向轮配置参数的权重，并配置底盘运动性能的参数。

视觉部分：

灯条检测→矩形框选定->字符区域白色比例判定->pnp解算

逻辑部分：

定点巡逻 、 搜索敌人 、瞄准设计 、伤害转向

## 4.3 算法功能模块说明

### 4.3.1定位算法

使用AMCL 最大似然度拟合的方法。该算法对底盘根据码盘、陀螺仪测量自身位置的要求较高，当车辆被障碍物阻塞，底盘打滑最为严重时，易出现AMCL校准偏慢，最终失效的情况。同时高度依赖激光雷达的数据，但在有斜坡的情况下，雷达易扫至地面，造成部分数据不可用。

尝试过使用ORB\_SLAM单目做定位，尺度的方法可以通过保存地图，限定启动位置解决。但不是特别稳定，受光线影响较大，在场地较暗的位置，易出现检测不到特征的情况;支持不了快速旋转的运动方式;视野中有较大且移动缓慢的障碍物时失效;摄像头只有单方向检测。综合以上原因，抛弃了在步兵车上做视觉SLAM的方案。

### 4.3.2导航算法

使用的导航算法为Dijkstra算法，由于实际的地图区域按照0.05m的分辨率，只有400×400的大小，正常搜索时搜索范围也只有相应大小；多源路径的方法计算压力比较小。与A\*算法一样，都是完备解法。Dijkstra算法不受启发函数的影响，绝对满足代价函数的单调性，而A\*算法为了能够单调，往往将启发函数的因子设置过大，更偏向于靠近目标点，往往不是很好的解。

采用RDP类的算法具有减少计算量的优势，但是场地总体来说比较宽阔，与房间走廊布置不相似，容易造成规划出的路径比较冗余。

**Navigation默认导航算法分析**

A\* 与 Dijkstra

算法原理：

Navigation Dijkstra算法思路（搜索效果最好的配置为 二次拟合 + 梯度下降重构）

**1.依次从Cur[ ]中取点作为当前点，以当前点作四邻域搜索**

**2.将代价小于这一轮搜索条件的点依次加入Next[ ]数组**

**3.Cur[ ]数组为空，提高搜索条件的阈值**

**4.将Next[ ]作为Cur[ ]，进行下一轮搜索**

算法缺陷：

上述算法忽略了Dijkstra算法的要求：每次计算都应当取Next[ ]中代价最小值计算。

当周围出现较多障碍物时，搜索区域内易出现局部最小值，特别是采用八邻域最小值重构时无法找到路径。

**重写Dijkstra算法(A\* )**

function Dijkstra(Graph, source): create vertex set Q

for each vertex v in Graph: // Initialization

dist[v] ← INFINITY // Unknown distance from source to v

prev[v] ← UNDEFINED // Previous node in optimal path from source

add v to Q // All nodes initially in Q (unvisited nodes)

dist[source] ← 0 // Distance from source to source

while Q is not empty:

u ← vertex in Q with min dist[u] // Node with the least distance will be selected first

remove u from Q

for each neighbor v of u: // where v is still in Q.

alt ← dist[u] + length(u, v)

if alt < dist[v]: // A shorter path to v has been found

dist[v] ← alt

prev[v] ← u

return dist[], prev[]

简单寻找最小代价的过程使算法复杂度为O(V\*4)，采用优先队列方式加快速度。

### 4.3.3跟踪射击

实现跟踪射击：

**激光雷达 目标位置检测（单兵）**

（初始节点启动时 请求 mapserver 获得代价地图，由costmap生成每一点距边界点距离的地图）

1.订阅激光雷达数据

2.监听tf\_tree 获取 激光雷达相对于地图的坐标

3.将激光雷达的二维点云数据映射至地图

4.根据点云所在坐标与边界距离做判别 是否是新出现的障碍物

5.对检测出的障碍物的点云做滤波归类处理

**视觉装甲位置检测(多兵模式)**

1.坐标计算

base\_link -> shoot\_base -> camera -> rival

订阅嵌入式底层云台转角的信息，获得云台的位置，得到枪管上云台的位置

2.当检测到敌人装甲时，利用 摄像头返回 目标的深度信息、Y方向的偏移

得到敌人的目标位置

3.将激光雷达的数据与视觉得到目标位置作匹配，得到匹配数组

4.若在误差范围内失配， 则继续以先前点作跟踪，并对丢失次数计数

5.在匹配数组中寻找距先前跟踪点最近的点，更新跟踪点

6.丢失次数超过阈值，重新以最近检测到的新的位置作跟踪

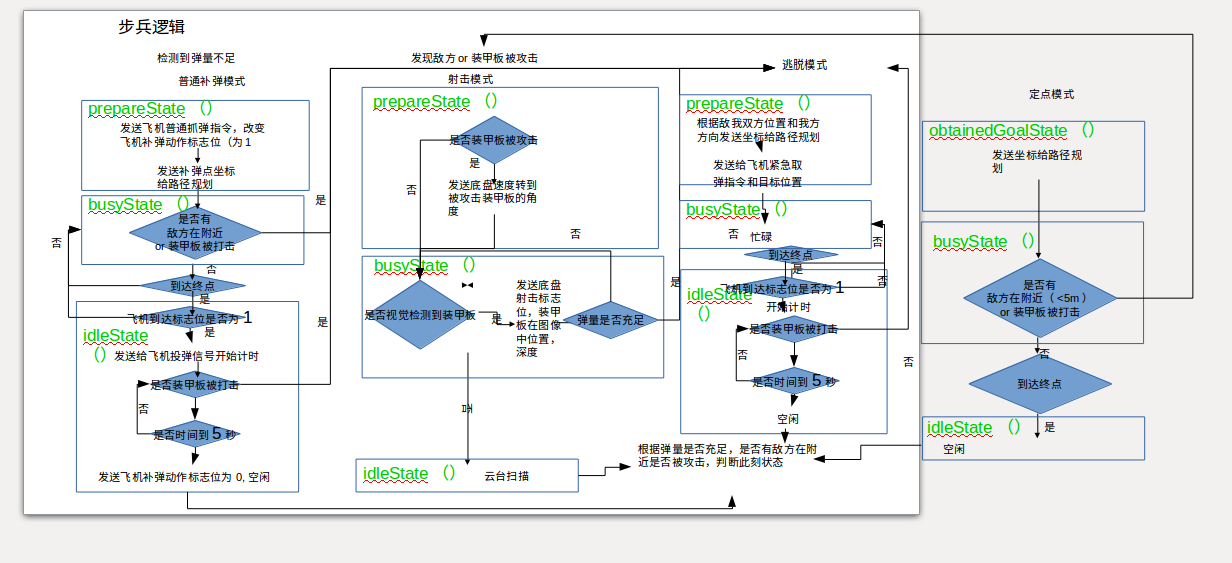
算法维护三个vector（采用时间戳 erase时间较久的数据） ：

激光雷达检测的可能区域

摄像头检测出现的位置

匹配成功的位置

### 4.3.4单兵逻辑

步兵逻辑：

整个控制逻辑只控制底盘运动，云台在接收到armor\_detect数据后由嵌入式底层全程全权控制。

基于官方的logical\_core包，我们做了如上扩展，步兵车分为四种模式，分别是补弹，射击，巡逻，定点模式。每种模式有三种状态，分别是准备状态忙碌状态和空闲状态。定点模式为二次开发的拓展模式，为后面更多的可能战术模式留出接口。补弹模式的触发条件是检测到弹量不足，射击模式的触发条件是发现敌方或者装甲板被打，当不满足以上条件时，自动切换到巡逻模式。

在比赛中，我们采用命令行控制的方式，写多个节点在巡逻模式下运行，使车能够一个命令行到达场地中的不同位置。具体到达位置是一阶段比赛1,2,3,4点。红蓝双方各四个点，共8个点。单兵比赛中我们先在本测起始区完成补弹。然后运动到对方起始区附近的123点位置，在这三个点上往复巡逻。云台收到上层发送的装甲板位置信息后自动打击。测试效果良好。

### 4.3.4多兵作战

多兵比赛时，各个车通过无线串口通信，发送彼此的位置坐表信息，时间有限这个信息没有应用的比赛策略中，延时0.3秒。

多兵比赛时，我们三个车分别在起始区，1点，4点补弹。起始时等待飞机补弹，当飞机补弹完毕后开始向敌方基地前进。三辆车分别从两侧向敌方基地前进。再到达敌方较远的点停止，进行打击。

## 4.4测试结果

实际测试时：1组、6组的车跑动效果相较其他组更加灵活，不容易出现RECOVERY状态。

利用激光雷达对敌人作位置检测将得到的坐标信息发不至地图上与激光雷达的数据点重合，误差主要为激光雷达的精度及AMCL定位算法的误差。

激光雷达融合视觉作多兵检测未来得及测试。

## 4.5 可优化方案

1.激光雷达检测受地形干扰比较严重，当车辆在斜坡位置，激光雷达扫至地面，可采用modelProb模型消除定位问题，但是障碍物层仍会添加新的障碍物，有必要修改mapserver部分。

2.在地图中根据敌方位置自主地设定目标点，规划路径作绕行、跟踪等

3.摄像头误检测的处理：采用激光雷达的信息，可在一定程度上对误检做判断

4.敌方位置的检测利用多个传感器的数据，也只是简单的作误差限制，可以参考AMCL的粒子滤波的方法

5.局部路径规划与全局路径规划相互协调的问题

局部路径规划条件设置过严，无法正常执行全局路径规划的路线

局部路径规划条件设置过松，运行可以比较流畅，但易进入全局路径规划认为危险的区域，造成全局 路径规划失败。全局路径规划起始点重新构建。

# 5 夏令营感想、总结

我们很有幸能在大疆robomaster夏令营度过充实又难忘的一个月。大家是来自全国各地的不同高校的一群机器人狂热爱好者。我们从一开始的陌生，到后来一起讨论方案，一起刷夜赶进度，再到最后比赛的齐心协力，这一切都将成为我们宝贵的回忆。成为优秀的工程师，开发出瞩目的产品，是每一个工科生的梦想。而夏令营刚好把有着同样梦想，志同道合的人聚在一起，给我们提供了一个广阔的舞台，对我们的成长都有很大帮助。

从个人技术层面来讲，这次夏令营要求我们做出一套自主的无人战车和无人机系统。整个项目富有挑战，难度较大，所以每个成员都经受了不小的历练。算法组的同学们一开始对ROS操作系统完全不了解，后来能在TX1上实现战车的自主导航和射击，并且对机器人导航和视觉识别相关算法都有了一定深度的理解。机械组的同学们设计了多套取弹机构，有涵道和机械爪，并进行了大量实际测试对方案进行评比，这一过程在他们以后的实际工作中会十分适用。嵌入式和硬件的同学是非常辛苦的，没日没夜的调试和维护是他们的日常生活。很高兴在夏令营结束的时候都能看的每个组员身上轻松的笑容，一份付出一份收获。

从团队意识层面来讲，夏令营的工作需要各个不同分工方向密切配合，并且需要团队成员对整体方案和进度安排有清晰的认识。无论是物资、经费和任务分工管理都不是件容易的事情。一开始我们队伍对比赛的理解有些偏差，在前期没有结合工作进度，提了很多不切实际的方案构想，这一度导致我们工作停滞，效率低下。好在后来我们及时调整态度，才在比赛中取得不俗的成绩。这个经验可以算是不成功的经验，但是对于我们来说反而比成功的经验更有价值。这教会了我们在面对实际工程问题时，如何能将所学的理论知识合理应用，而不是纸上谈兵，华而不实。

总而言之，我们度过了充实的一个暑假，收获了知识和友情，也非常感谢大疆，感谢RM能给我们提供一次这么好的机会。



RoboMaster大赛组委会

邮箱：[robomaster@dji.com](mailto:robomasters@dji.com)

官方论坛：[http://bbs.robomaster.com](http://bbs.robomasters.com/)

官方网站：[http://www.robomasters.com](http://www.robomasters.com/)

电话：075536383255（周一至周五10:00-19:00）

地址：广东省深圳市南山区西丽镇茶光路1089号集成电路设计应用产业园2楼202



**微信**  **微博**

ROBOMASTERTM 是大疆创新的商标。

Copyright © 2017 大疆创新 版权所有