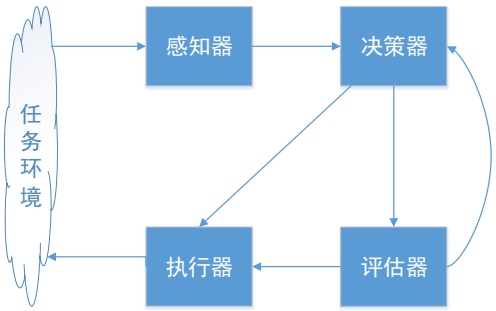
逻辑与控制部分技术报告

飞行器逻辑架构以及控制部分，和大多数一样，我主要采取 ROS 与 Linux 系统调用相关库函数的使用以及 C/C++编程实现飞行框架的搭建。首先我们采用普适的智能机器人控制模型：

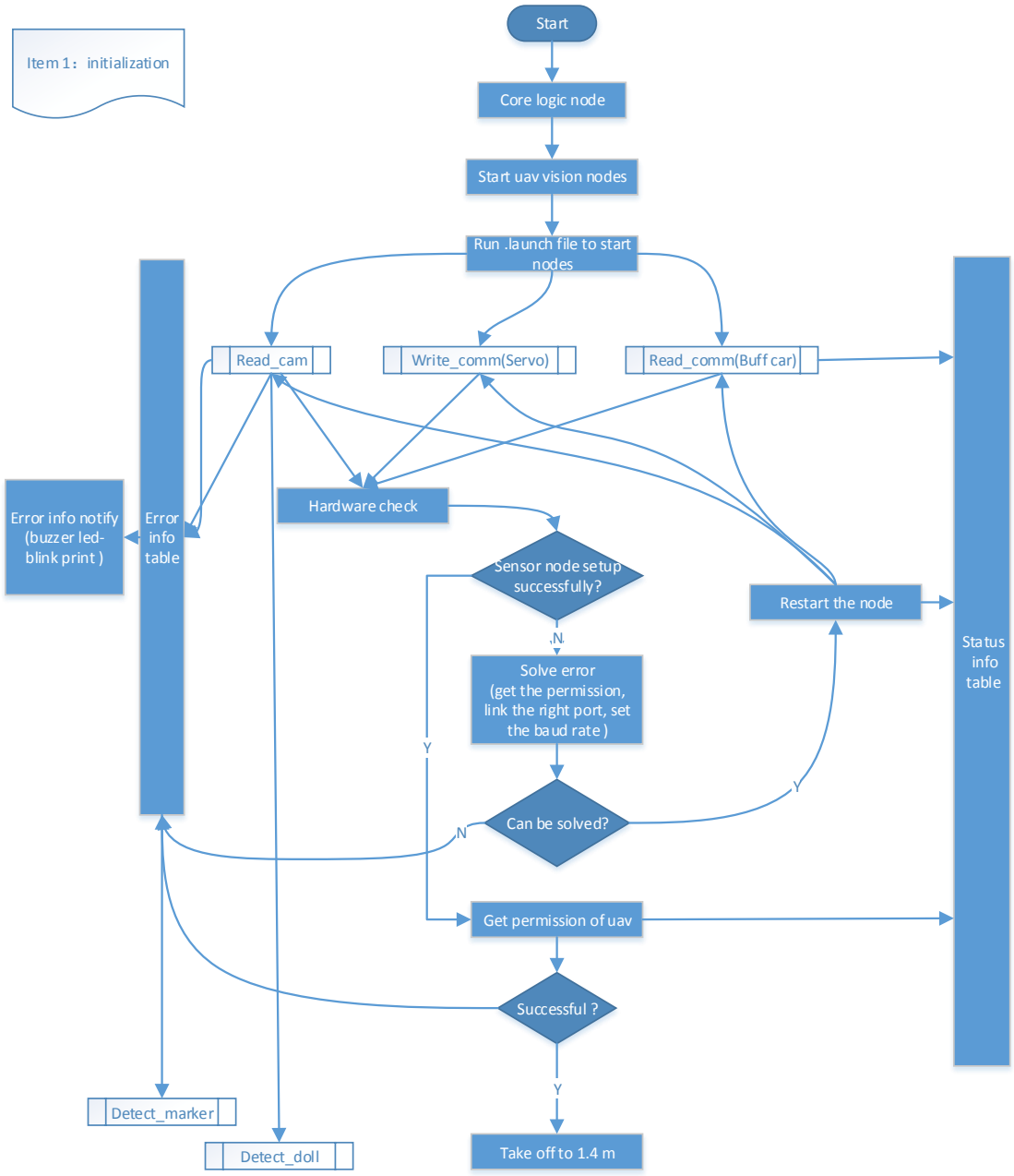


【图一：智能机器人控制模型】

为了使评估系统具有灵活性以及消除其不确定性以及复杂性，我们将其分布到整个任务的不同阶段状态中实现整个调度的实现及其优化。

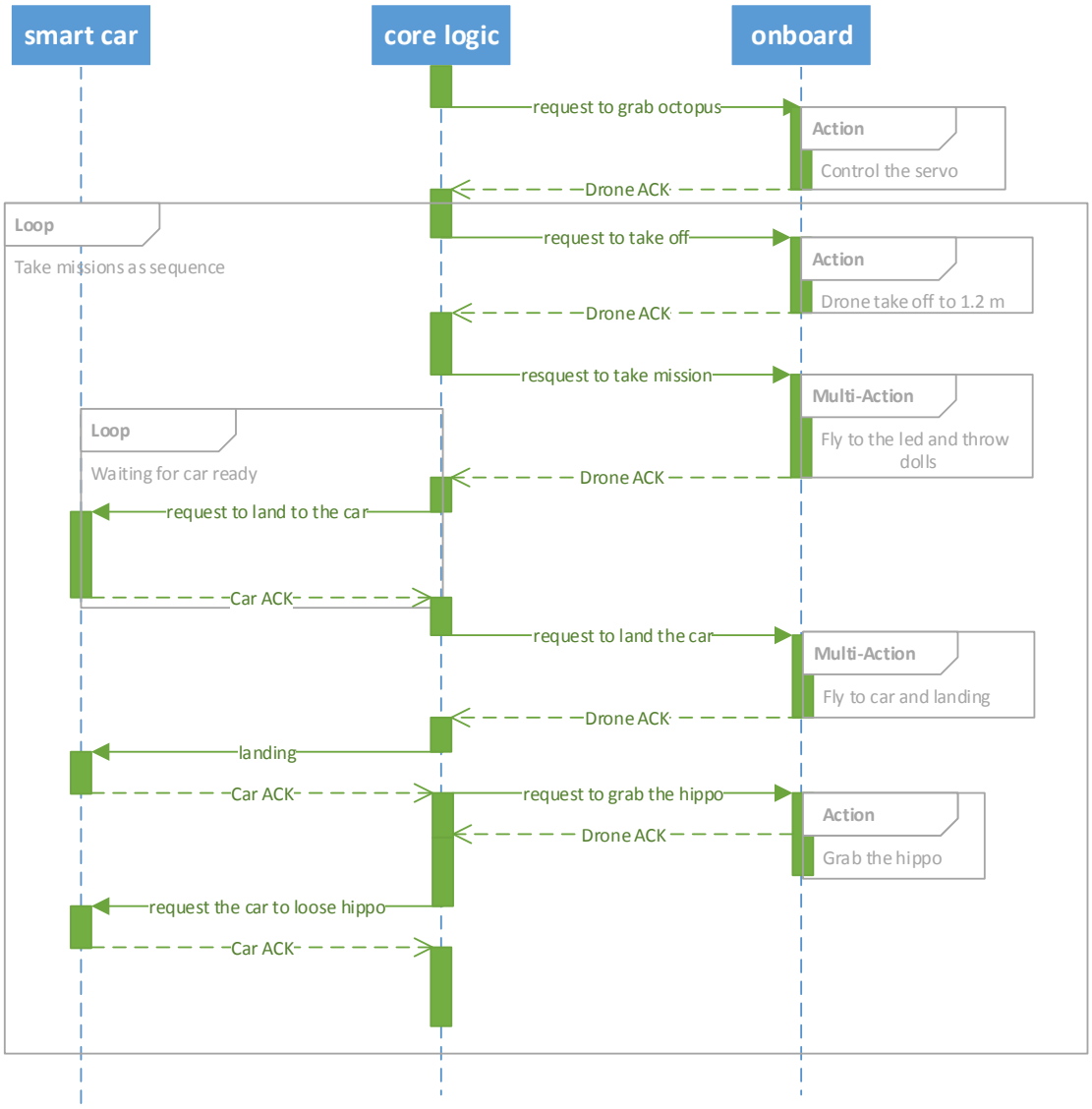
版本更新与控制

关于整个飞行器的飞行控制架构，我们主要经历过三个重要的改版，每一个版本都是对飞行控制的不断迭代与磨合。其中初版采用比较高频的节点之间通信以实现不同节点之间的协同控制，其中初始化飞行器步骤如下：



【图二 drone 启动初始化】

经过控制实现的测试以及上下层之间的相互调整，最终通过 ROS service 的方式实现不同层次的控制：

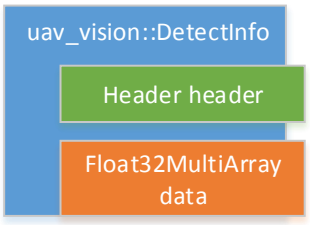


【图三：控制流程任务调度】

ROS 控制与通信模型

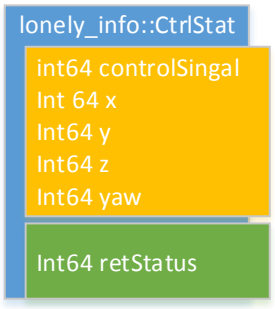
在通信控制中，主要采取两种通信模式： Topic 与 Service: Topic 用于信息采集传感器，用于大量并且具有固定波特率发送大量信息的节点间通信，例如 Guidance 以及 SLAM 采取的地理位置信息以及 Monopoly 采集的图像信息，通过发布相关信息来实现多节点订阅并行处理数据；Service 用于节点两两之间的少量数据交换需要可靠的通信，我们用于主逻辑节点与各个执行器例如 onboard，servo 之间的指令发送与状态转换。

其中 topic 与 service 我们统一了数据结构，其中 topic 数据结构如下：



header 用于记录数据包的顺序、id 以及时间戳用于同步以及确认信息有效性，data 主要存储数据信息。

Service 信息主要通过下面的数据结构来实现：



其中黄色部分为 request 部分的数据，其中包括控制信号以及给飞行器发送的大地全局位置信息，onboard 返回小车或飞行器的任务状态。