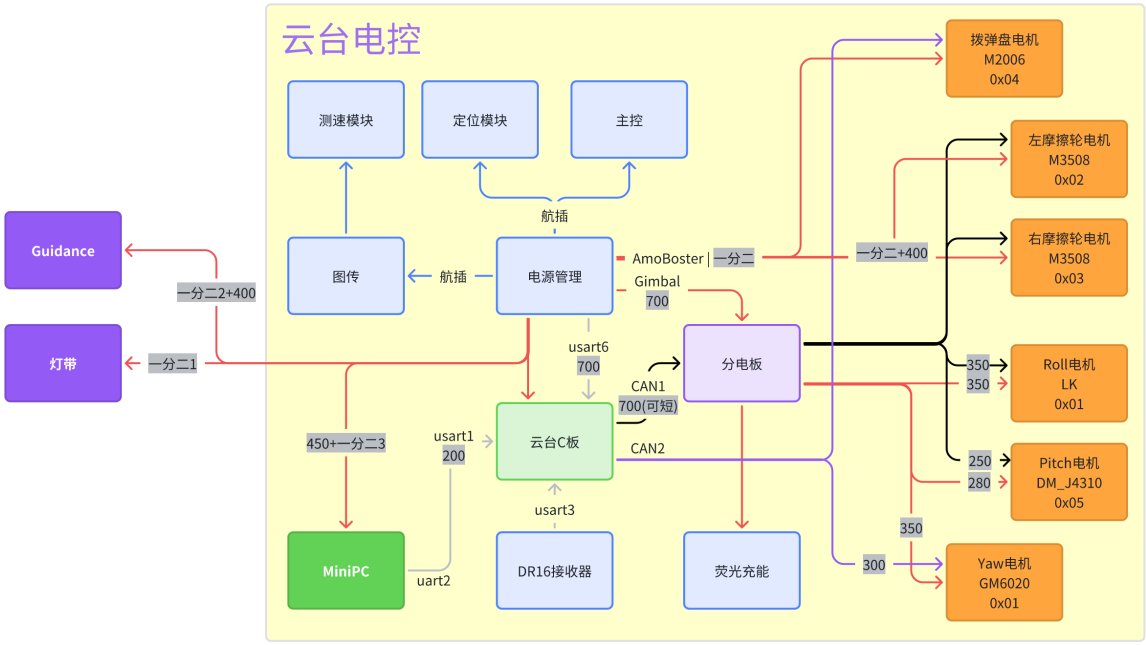
无人机2025电控技术方案

硬件布置与连接

电路拓扑



- 拓扑图中红色线路代表电源，其他颜色代表不同种类的信号线
- 线路中的数字标注为对应的线材焊接的大致长度，未标记的主要为成品线

注意事项

1. 对于当前的三轴无人机云台机械结构，多数从电管延伸到云台上的线材需要为活动空间保留一定的长度冗余，所以在布线时，建议提前测量好合理活动空间内各个线材所需的大致长度，并在此基础上多补充50-100（mm）作为冗余，用于布线固定时可能造成的长度消耗；同时适当的长度冗余既可以减缓线材下对于接口的拉扯、保证接口的稳固与信号/电源稳定，也能减小线材对于云台三轴控制可能造成的非线性影响。
2. 完成初步线材安排之后，建议将各个线材的长度做好记录（例如上图中在拓扑图中的各长度数据）。既方便焊接备用线材，避免重复测量。也能在线材出现质量问题时辅助分析不合理的线材长度，并加以修改。
3. 当前的机械结构由于没有导电滑环之类的线材引导结构，从上层弹舱处电管引出到云台的多数线材具有较大的活动空间，不便于对其进行充分、美观的保护。主要的保护和固定措施应着重于：
    1. 使用套线管、编织网套保护较细的信号线。
    2. 使用打胶或扎带等固定手段，防止XT30等接口脱落。
    3. 对于连接到云台上的线材，在云台上的接口附近寻找合适的机械附着点进行固定，使得从接口到附着点的长度相对不变。从而将云台运动时，对线材可能的拉扯效果尽量转移到未被固定的中间部分，避免接口受力造成的不良影响。

4. 当前多数的硬件供电电压为12V，而目前购买的灯带多为5V供电，需要请硬件组帮忙制作对应降压板。尤其注意提前告知所使用灯带的灯珠型号以及预计的灯珠数量，提醒硬件在设计时对于额定电流保留一定的冗余，避免过流烧板。同时在降压板背面通过绝缘材料进行保护，避免接触碳管引发短路等安全事故。
5. 在供弹链路镂空处加装荧光充能灯珠时，建议先套一圈扎带用于结构保护，防止灯带内陷阻塞弹路，引发的卡弹等结果。

软件设计

基本架构

25赛季电控组的各成员基本统一使用了基于状态机的嵌入式架构，便于开发、调试和维护。

无人机电控主要负责的模块为云台三轴和发射机构，这里进行基本框架的组成分析

HW-Components

电控组组件库，封装了主要的硬件设备驱动、常用的算法、裁判系统通信协议、主要模块的基本功能等，便于重复使用。

Instance

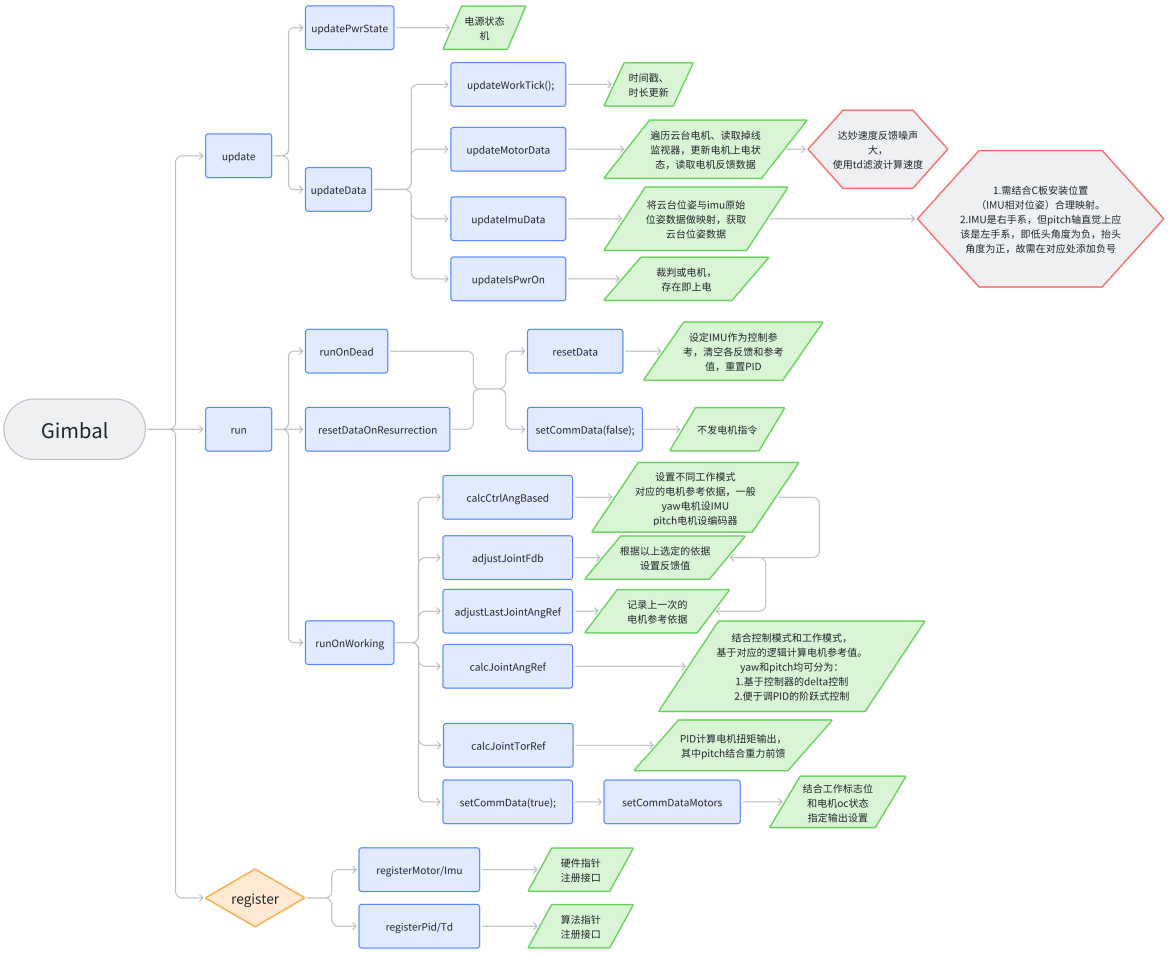
基本模块对象的实例化，创建并初始化对应组件（主要是组件库中封装好的对象）的唯一指针，便于在各个状态机中进行独立的调用。同时避免对象重复、内存泄露和地址冲突之类的常见bug。

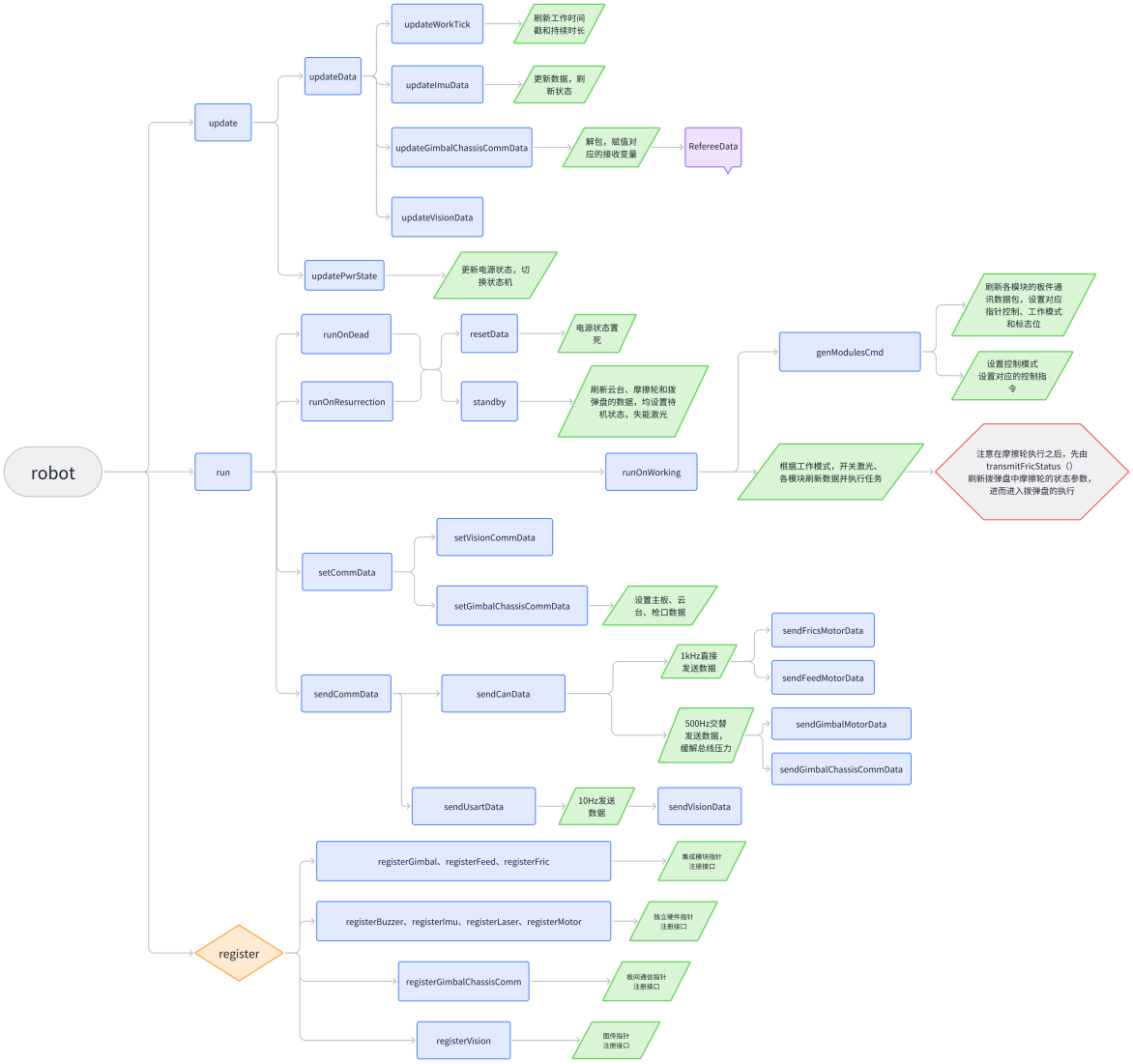
RobotModules

主要模块状态机的核心逻辑部分，各个模块的类通过引用具体的硬件和组件指针，进行控制逻辑的设计，配合完成系统的总体控制。

以下是舵轮步兵、云台部分核心代码（gimbal、robot）的框架梳理， 无人机云台代码在细节实现上略有调整，但整体架构的理解可以直接参考

#####

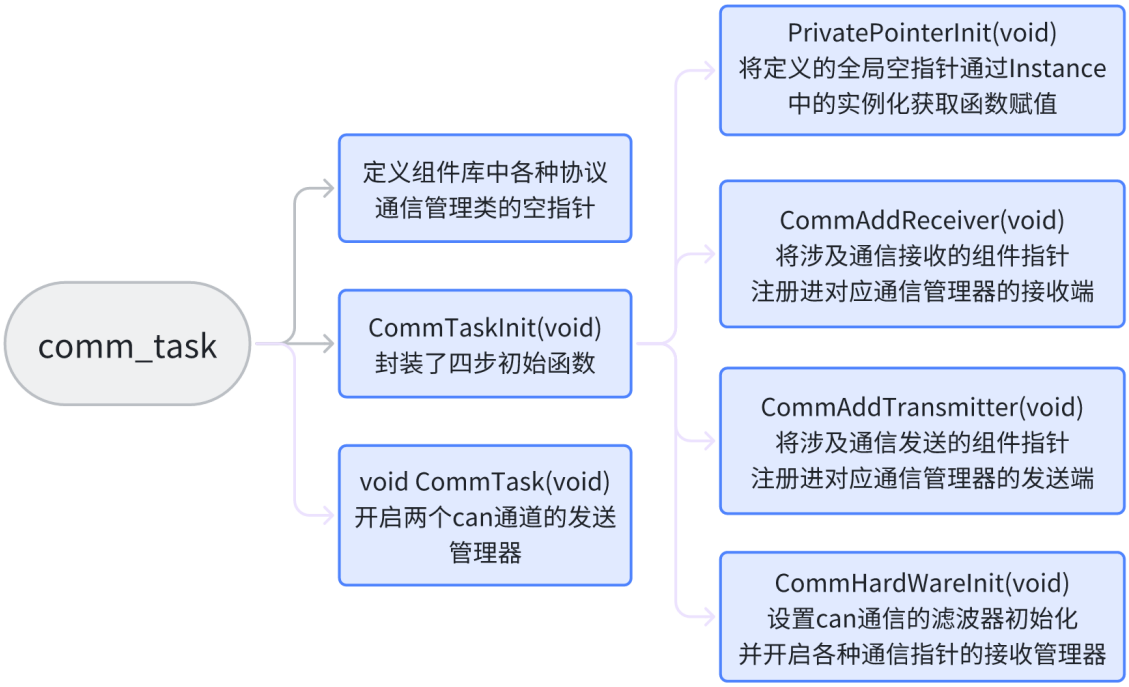

Gimbal
update
updatePwrState
电源状态机
updateData
updateWorkTick();
时间戳、时长更新
updateMotorData
遍历云台电机、读取掉线监视器，更新电机上电状态，读取电机反馈数据
达妙速度反馈噪声大，使用td滤波计算速度
updateImuData
将云台位姿与imu原始位姿数据做映射，获取云台位姿数据
1.需结合C板安装位置（IMU相对位姿）合理映射。
2.IMU是右手系，但pitch轴直觉上应该是左手系，即低头角度为负，抬头角度为正，故需在对应处添加负号
updateIsPwrOn
裁判或电机，存在即上电
run
runOnDead
resetData
设定IMU作为控制参考，清空各反馈和参考值，重置PID
resetDataOnResurrection
setCommData(false);
不发电机指令
runOnWorking
calcCtrlAngBased
设置不同工作模式对应的电机参考依据，一般yaw电机设IMU pitch电机设编码器
adjustJointFdb
根据以上选定的依据设置反馈值
adjustLastJointAngRef
记录上一次的电机参考依据
calcJointAngRef
结合控制模式和工作模式，基于对应的逻辑计算电机参考值。
yaw和pitch均可分为：
1.基于控制器的delta控制
2.便于调PID的阶跃式控制
calcJointTorRef
PID计算电机扭矩输出，其中pitch结合重力前馈
setCommData(true);
setCommDataMotors
结合工作标志位和电机oc状态指定输出设置
register
registerMotor/Imu
硬件指针注册接口
registerPid/Td
算法指针注册接口



Task

一般包含**comm_task**和**main_task**

如下图所示，comm_task负责通信相关任务的处理，包括回调函数的设计、通信组件库的调用等。



main_task用于封装唯一接口，在main.c中进行调用。通过特定的语法，允许C语言文件可以完成对各自定义的Cpp文件的调用。

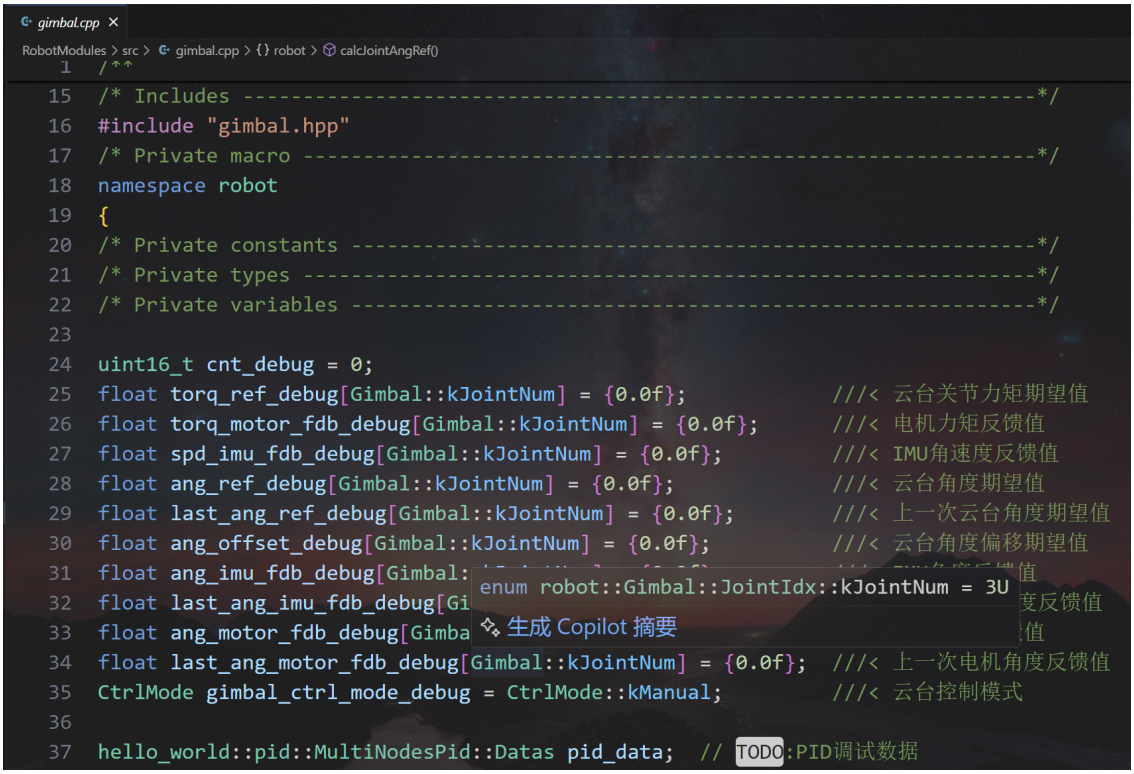
注意事项

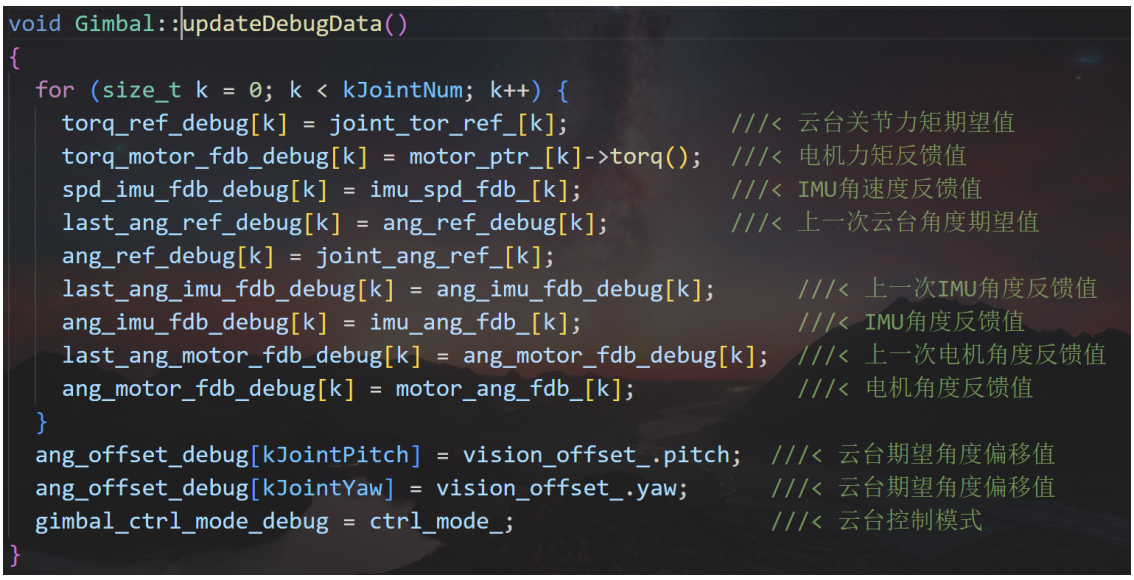
以下内容是个人调试和测试过程中发现的一些问题和注意事项，而非对基本组件库使用的讲解，**建议阅读者对于常用组件库的基本使用及代码框架有了初步的认识与理解**，请知悉！

1. 拨弹盘组件库初始化参数中，零位偏置角度的设置建议通过测试实际单发效果进行设置，而非通过机械理论参数或手拨后读取编码器等方式来参考。因为弹链的总体长度会受单发限位的位置、弹丸公差、安装公差等综合因素影响，使得后者测出的数值并非实际使用中的最优参数。
2. 无人机云台控制完全基于IMU，IMU的零漂问题对云台控制和自瞄的影响尤其明显。通过测试，使用传统的采样取平均的方案易受外界干扰的影响，表现不稳定；推荐使用人为测试固定的零漂参数，尽量减小零漂现象。同时参数的设置受温度影响较大，在改变调试环境时，建议及时测试微调，以取得更好的效果。

   另外IMU的加速度计有助于缓解零漂，但是在摩擦轮开启时，加速度计的数据易产生较大的噪声，对IMU的测量数值产生一定影响。因此需要合理参考加速度计，例如可以考虑在开启摩擦轮时关闭加速度器，更优的方案仍需要更多的控制思路和测试来确定。目前认为手动设置固定零漂数值的策略下，开关加速度计对反馈值的影响不大。

3. 使用Ozone进行调试，每次重新烧录程序时，被查看数值或变化波形的一般变量会被自动取消，但全局变量会保留。为了方便调试，建议为需要持续观察的变量在相关程序文件开头处创建全局变量接口，主动在程序中批量赋值，通过查看这些变量的波形，可以避免调试参数（如PID）时反复重新烧录再指定观察对应的变量。





4. pitch角度在直观考虑上正下负（遵循左手系），但是视觉自瞄统一遵循右手系，对于pitch角度即为下正上负。

   相关的正负处理会在gimbal的关节角度赋值，以及与视觉的通信协议中做出对应处理，在使用时应注意辨析区分。

```cpp
void Gimbal::updateImuData()
{
  HW_ASSERT(imu_ptr_ != nullptr, "pointer %d to imu %d is nullptr", imu_ptr_);
  // IMU是右手系，但pitch轴直觉上应该得是左手系，即低头角度为负，抬头角度为正，故在此处加负号
  imu_ang_fdb_[kJointRoll] = imu_ptr_->roll();  // TODO:是否需要roll轴的反馈
  imu_ang_fdb_[kJointPitch] = (-1.0f) * imu_ptr_->pitch();
  imu_ang_fdb_[kJointYaw] = imu_ptr_->yaw();

  imu_spd_fdb_[kJointRoll] = imu_ptr_->gyro_roll();  // TODO:是否需要roll轴的反馈
  imu_spd_fdb_[kJointPitch] = (-1.0f) * imu_ptr_->gyro_pitch();
  imu_spd_fdb_[kJointYaw] = imu_ptr_->gyro_yaw();
};
```

ins_vision.cpp | ins_vision.hpp | vision.hpp

HW-Components > devices > vision > inc > vision.hpp > ...

```cpp
#ifndef HW_COMPONENTS_DEVICES_VISION_VISION_HPP_
namespace hello_world
namespace vision
class Vision : public comm::Receiver, public comm::Transmitter
  TargetColor getTargetColor(void) const { return tx_data_.tgt_color; }

  void setPose(
      float roll, float pitch, float yaw, bool is_left_hand_system = false)
  {
    tx_data_.roll = roll;
    tx_data_.pitch = is_left_hand_system ? pitch : -pitch;
    tx_data_.yaw = yaw;
  };
  float getPoseRoll(bool is_left_hand_system = false) const
  {
    return tx_data_.roll;
  }
  float getPosePitch(bool is_left_hand_system = false) const
  {
    return is_left_hand_system ? tx_data_.pitch : -tx_data_.pitch;
  }
  float getPoseYaw(bool is_left_hand_system = false) const
  {
    return tx_data_.yaw;
  }
```

5. 调试roll轴PID参数时，可以将其他轴的控制指令映射到roll轴上，调试完成后再恢复。
6. 由于无人机的pitch轴重心距离轴心较近，且存在线材拉扯等非线性因素的影响，简单基于刚体模型的重力前馈对于pitch轴控制效果不好。目前的方案是通过测量pitch限位范围内，维持多个不同角度（基本均匀分布）所需要的电机力矩，通过数值计算的高次拟合获得一条角度-力矩曲线，作为前馈映射关系。

   测试发现，云台上仰时所需的力矩前馈较大，下降时较小。基于此，在云台下降时使用了一定的比例系数关系，保证前馈连贯的同时，一定程度优化了响应速度和超调表现。

```cpp
// pitch轴重力前馈
if (joint_idx == kJointPitch) {
  float ang = joint_ang_fdb_[joint_idx];
  forward_toq[0] = 9.75f * ang * ang * ang + 6.5868f * ang * ang + 2.0765f * ang + 0.7248f;  ///< 前馈力矩，单位 N·m
  if (joint_ang_ref_[joint_idx] - joint_ang_fdb_[joint_idx] < -0.025f) {
    if (joint_ang_fdb_[joint_idx] > 0.0f) {
      float percent = joint_ang_fdb_[joint_idx] / cfg_.max_ang[kJointPitch];
      forward_toq[0] *= (0.36f + percent * 0.64f);  // 前馈力矩随角度变化
    } else {
      forward_toq[0] *= 0.36f;
    }
  }
}
```

7. UI绘制一些由特殊事件触发的元素时，应该基于相关数据的特征加以处理。如果是只在触发时较短时间内（如一个控制周期、一个通信周期）有效的标志位（如血量的扣除），对应元素显示不能简单地通过标志位的情况来判定（否则可能因为显示时间极短而失去意义），可以通过标志位的变化刷新一个内置的计时器，来保证元素被有效显示。

```
void UiDrawer::updateEnemyHitX(StraightLine &g_line_enemy_1, StraightLine &g_line_enemy_2)
{
  static uint16_t show_tick = 0;
  if ((vis_mode_ == VisionMode::Outpost || vis_mode_ == VisionMode::Normal) && is_enemy_outpost_hit_) {
    show_tick += 500;
  } else if ((vis_mode_ == VisionMode::Base || vis_mode_ == VisionMode::Normal) && is_enemy_base_hit_) {
    show_tick += 500;
  }
  if (show_tick > 1000) {
    show_tick = 1000;  // 限制最大显示时间
  }

  if (show_tick > 0) {
    g_line_enemy_1.setVisibility(true);
    g_line_enemy_2.setVisibility(true);
    show_tick--;
  } else {
    g_line_enemy_1.setVisibility(false);
    g_line_enemy_2.setVisibility(false);
  }
```