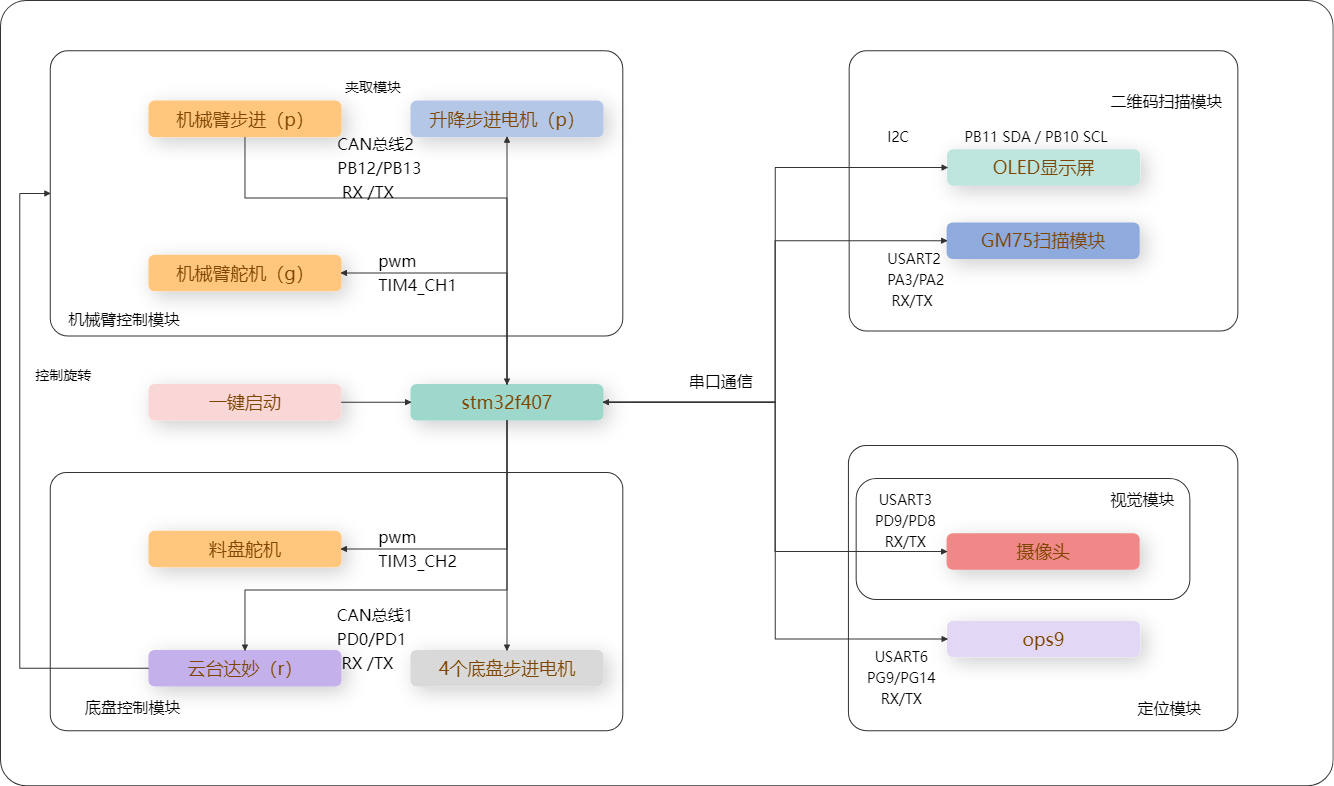
# **运输赛道一组\_ 架构与接口设计说**

说明：

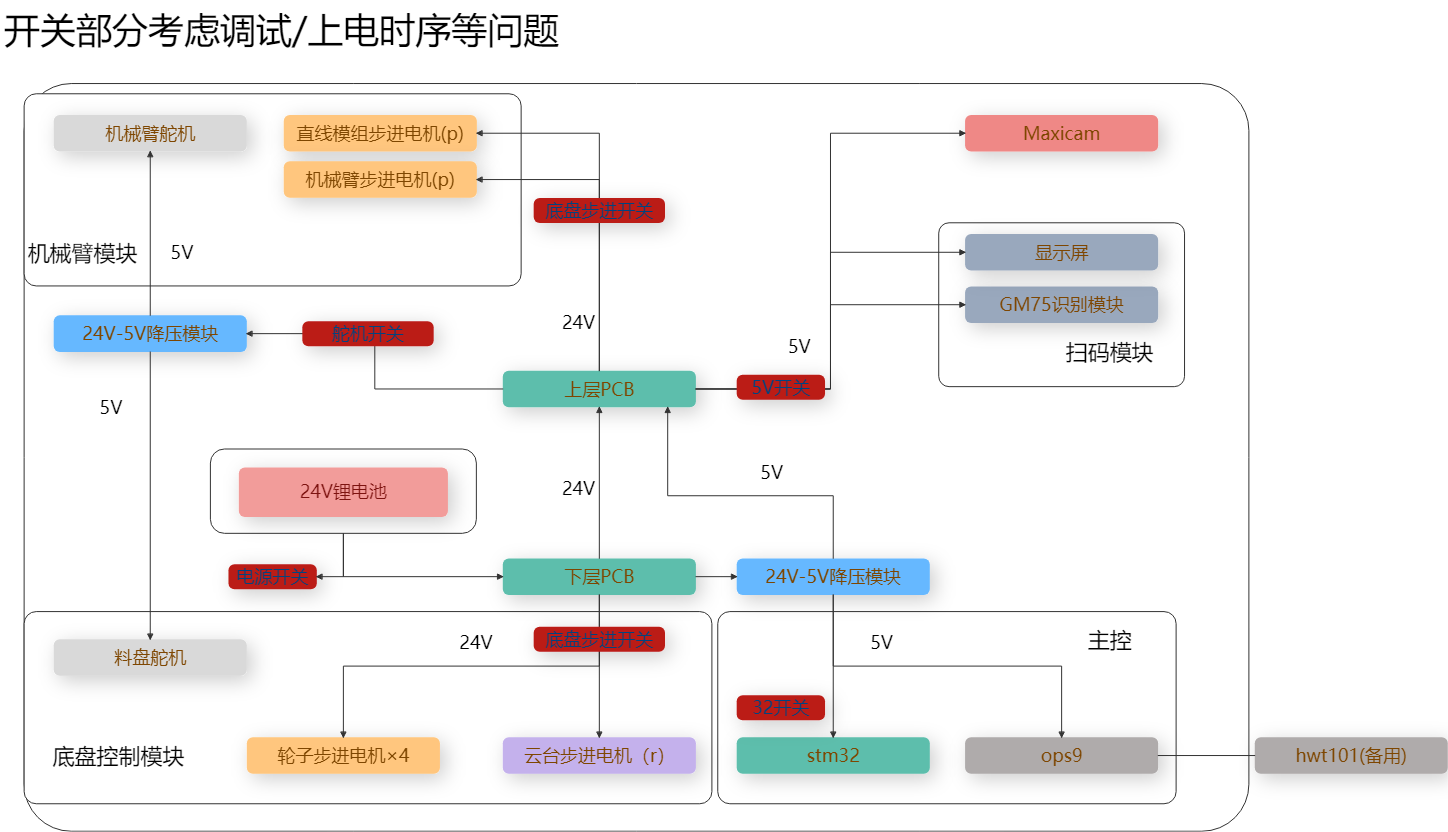
1. 上交审核时删除所有<>内的内容和 ”说明“ 部分的内容
2. 所有图表使用钉钉自带的画图功能绘制，接口定义使用钉钉自带的代码显示功能

# **硬件架构**

## **信号线路**



## **电源线路**



# **任务分析**

## **任务简述**

* **Task 1: 启动与任务获取** maixcam接收启动指令，指挥stm32控制底盘前往二维码区。stm32通过GM75模块扫描并上传任务码给maixcam进行决策，同时根据maixcam的指令在OLED屏上显示任务信息。
* **Task 2: 第一批物料搬运 (原料区 → 粗加工区 → 暂存区)** maixcam规划完整路径并分步下达指令。在原料区，maixcam进行动态追踪识别（颜色、位置），stm32则控制底盘与机械臂协同抓取物料。随后，stm32将物料运送并放置在“粗加工区”，再从“粗加工区”二次搬运，最终放置到“暂存区”。放置前maixcam通过视觉识别圆心与直线进行精准位姿校准。
* **Task 3: 第二批物料搬运与码垛 (原料区 → 粗加工区 → 暂存区码垛)** maixcam指挥stm32返回原料区，重复Task 2的取料和在“粗加工区”的放置流程。在最终搬运至“暂存区”时，执行“码垛”操作，在maixcam的视觉引导下，stm32控制机械臂将第二批物料精准地堆叠在第一批物料之上。
* **Task 4: 任务完成并返航** maixcam在确认所有搬运及码垛任务完成后，向stm32发送返航指令，stm32控制底盘自动行驶，最终准确停靠在“启停区”，比赛结束。

## **模块分工**

* **感知模块**
  + maixcam pro是核心视觉传感器，负责开赛前的位姿校准，以及在抓取过程中识别物料颜色、在放置时进行视觉伺服（直线+圆心校准）。
  + ops9是核心定位传感器，负责在机器人整个运行过程中，提供连续、精准的定位信息。
  + GM75二维码扫描器是任务获取传感器，专用于扫描二维码，获取比赛任务序列。
* **一级控制模块** PC机是远程监控与调试终端，负责系统的整体规划、开发阶段的调试和赛前状态的确认。
* **二级控制模块** maixcam是**决策中心**，负责处理所有视觉信息，进行复杂的解算和决策，并将具体、高层次的指令（例如“移动到目标点”、“开始动态追随”、“抓取红色物料”）发送给stm32。
* **三级控制模块** stm32是**执行核心**，根据maixcam发来的指令，进行实时的硬件控制，包括驱动底盘电机、控制机械臂和云台的步进电机与舵机，以及更新OLED显示屏。
* **执行模块** 底盘、机械臂、云台、OLED显示屏是物理执行单元，负责执行抓取物料、放置物料、显示任务编号等所有实际工作。

graph TD  
 %% 定义功能模块 (系统架构)   
   
 subgraph "一键控制按钮"  
 按钮[("按钮<br>开机启动")]  
 end  
  
 subgraph "智能车系统"  
 subgraph "主控 (maixcam) - 决策大脑"  
 Master["算力与视觉中心"]  
 end  
   
 subgraph "下位机 (STM32) - 执行核心"  
 Slave[("STM32<br>硬件控制中枢")]  
   
 subgraph "板载感知单元"  
 direction LR  
 GM75["GM75<br>二维码扫描"]  
 OPS9["OPS9<br>定位"]  
 end  
   
 subgraph "物理执行单元"  
 direction LR  
 Chassis["底盘"]  
 Arm["机械臂/云台"]  
 OLED["OLED显示屏"]  
 end  
   
 GM75 -- "任务码" --> Slave  
 OPS9 -- "定位数据" --> Slave  
 Slave -- "驱动" --> Chassis & Arm & OLED  
 end  
  
 %% 主控与下位机核心通信  
 Master <-->|高层指令/状态反馈| Slave  
 end  
  
 %% 定义任务流程   
  
 %% --- Task 1: 启动与任务获取 ---  
 按钮 -- "<b>Task 1.1</b><br>一键启动指令" --> Master  
 Master -- "<b>Task 1.2</b><br>指令: 前往二维码区" --> Slave  
 Slave -- "<b>Task 1.3</b><br>反馈: 任务码已由GM75获取" --> Master  
 Master -- "<b>Task 1.4</b><br>指令: 在OLED上显示任务" --> Slave  
   
 %% --- Task 2: 第一批物料搬运 ---  
 Master -- "<b>Task 2</b><br>指令: 执行第一批物料完整流程<br>(取料 → 粗加工 → 暂存)" --> Slave  
   
 %% --- Task 3: 第二批物料搬运与码垛 ---  
 Master -- "<b>Task 3</b><br>指令: 执行第二批物料完整流程<br>(取料 → 粗加工 → 码垛)" --> Slave  
  
 %% --- Task 4: 返航 ---  
 Master -- "<b>Task 4</b><br>指令: 任务完成, 返回启停区" --> Slave

## **功能接口**

## 主控 → 下位机 (命令下发接口)

// 接口：I\_CommandExecutor (由 stm32 实现)  
  
interface I\_CommandExecutor   
{  
 /\*\*  
 \* @brief 指挥底盘移动到场地的绝对坐标。  
 \* @param targetX 目标点的X坐标 (mm)。  
 \* @param targetY 目标点的Y坐标 (mm)。  
 \* @param targetAngle 最终的目标姿态角度。  
 \* @return CommandStatus 表示指令是否成功接收。  
 \*/  
 CommandStatus MoveTo(float targetX, float targetY, float targetAngle);  
  
 /\*\*  
 \* @brief 执行一次完整的抓取动作。  
 \* @param materialType 要抓取的物料标识（如颜色）。  
 \* @param trackingData 主控提供的动态追踪辅助数据。  
 \* @return CommandStatus 表示指令是否成功接收。  
 \*/  
 CommandStatus ExecuteGrasp(MaterialType materialType, object trackingData);  
  
 /\*\*  
 \* @brief 执行一次完整的放置动作。  
 \* @return CommandStatus 表示指令是否成功接收。  
 \*/  
 CommandStatus ExecutePlace();  
  
 /\*\*  
 \* @brief 执行一次码垛放置动作（放置在已有物料上）。  
 \* @return CommandStatus 表示指令是否成功接收。  
 \*/  
 CommandStatus ExecuteStack();  
  
 /\*\*  
 \* @brief 更新OLED显示屏上的内容。  
 \* @param line1 第一行要显示的文本。  
 \* @param line2 第二行要显示的文本。  
 \* @return CommandStatus 表示指令是否成功接收。  
 \*/  
 CommandStatus UpdateDisplay(string line1, string line2);  
}

## **下位机 → 主控 (事件与数据上报接口)**

// 接口：I\_StatusReporter   
  
interface I\_StatusReporter  
{  
 /\*\*  
 \* @brief 当下位机完成一个MoveTo指令后调用。  
 \* @param finalPose 机器人最终的精确位姿。  
 \*/  
 void OnMoveCompleted(Pose finalPose);  
  
 /\*\*  
 \* @brief 当一个动作（抓取/放置/码垛）完成后调用。  
 \* @param actionType 完成的动作类型。  
 \* @param success 动作是否成功。  
 \*/  
 void OnActionCompleted(ActionType actionType, bool success);  
  
 /\*\*  
 \* @brief 当GM75模块成功扫描到二维码后调用。  
 \* @param qrCodeContent 二维码包含的字符串任务码。  
 \*/  
 void OnQRCodeScanned(string qrCodeContent);  
  
 /\*\*  
 \* @brief 下位机周期性上报由OPS9提供的当前位置信息。  
 \* @param currentPose 机器人当前的精确位姿。  
 \*/  
 void OnPoseUpdated(Pose currentPose);  
}

## **下位机 (stm32) 与 板载外设 接口**

// 接口：I\_ChassisDriver (底盘驱动)  
interface I\_ChassisDriver  
{  
 // 输入: X、Y方向速度和角速度。输出: 无。  
 void SetVelocity(float vx, float vy, float angular\_v);  
}  
  
// 接口：I\_ArmDriver (机械臂驱动)  
interface I\_ArmDriver  
{  
 // 输入: 预设的宏指令名称。输出: 无。  
   
 void ExecuteMacro(string macroName);  
}  
  
// 接口：I\_QRCodeReader (GM75驱动)  
interface I\_QRCodeReader  
{  
 // 输入: 无。输出: 扫描到的字符串，若未扫到则返回null或空。  
 string ReadData();  
}  
  
// 接口：I\_Localizer (OPS9驱动)  
interface I\_Localizer  
{  
 // 输入: 无。输出: 包含位置和姿态的Pose对象。  
 Pose GetCurrentPose();  
}  
  
// 接口：I\_Display (OLED驱动)  
interface I\_Display  
{  
 // 输入: 两行字符串。输出: 无。  
 void Show(string line1, string line2);  
}