全国大学生物联网设计竞赛

**基于OneOS的远程机械臂系统**

|  |  |
| --- | --- |
| 学校名称： | 华南理工大学 |
| 团队名称： | 青云队 |
|  |  |
| 队长： | 石海鹏 |
| 队员1： | 魏烁堃 |
| 队员2： | 林标伟 |

全国大学生物联网设计竞赛组委会

2019年5月

**基于OneOS的远程机械臂系统**

# 摘 要

本文主要介绍了基于OneOS的远程机械臂系统的主要功能、优势和实现细节。

机械臂常见，而跑操作系统的少，拥有从底层到客户端的整套框架、支持远程控制的机械臂少。为此我们设计了这套基于OneOS的远程机械臂系统。

本系统由用户控制界面、ESP32-CAM图像传输与通信以及机械臂部分组成。用户控制界面提供实时图像传输和交互式控制按钮，ESP32-CAM负责将实时图像通过RTSP协议传输给用户控制界面，并通过MQTT接收用户控制信号，机械臂部分根据接收到的指令执行相应动作。在软件开发技术方面，客户端采用Flutter框架，实现实时图像传输和交互式控制。控制层技术使用MQTT协议进行用户控制信号传输，通过RTSP推流实现视频传输。在机械臂部分，实现了底层控制和运动学代码构建。路径规划方面采用了五次曲线插补的方法，以实现机械臂在运动过程中的平滑轨迹。

本作品在机械臂控制系统领域具备三大独特的特色与创新：一是基于OneOS系统设计的完善底层控制代码，使机械臂的开发变得更加简洁高效；二是进行了完善的运动学建模，实现了机械臂的精准控制；三是搭建了从云到端的完整框架，实现了远程控制系统，用户能够通过界面上的按钮远程操控机械臂进行操作。

**关键词：机械臂控制系统，OneOS，机械臂运动学，RTSP，MQTT**

**目 录**

[摘 要 I](#_Toc141203875)

[1. 设计需求分析 1](#_Toc141203876)

[2. 特色与创新 4](#_Toc141203877)

[3. 功能设计 5](#_Toc141203878)

[3.1 机械臂代码 5](#_Toc141203879)

[3.1.1 操作系统层 5](#_Toc141203880)

[3.1.2 运动学代码 5](#_Toc141203881)

[3.2 用户控制界面 6](#_Toc141203882)

[3.2.1 实时图像传输 6](#_Toc141203883)

[3.2.2 交互式控制 6](#_Toc141203884)

[3.2.3 远程通信传输 6](#_Toc141203885)

[3.3 ESP32-CAM图像传输与通信部分 6](#_Toc141203886)

[3.3.1 实时图像传输 6](#_Toc141203887)

[3.3.2远程控制接收 7](#_Toc141203888)

[3.3.3指令传输到机械臂 7](#_Toc141203889)

[4. 系统实现 8](#_Toc141203890)

[4.1 机械臂控制程序 8](#_Toc141203891)

[4.1.1 进程关系以及各自任务 8](#_Toc141203892)

[4.1.2 插补与路径规划 9](#_Toc141203893)

[4.2 用户控制界面 10](#_Toc141203894)

[4.2.1软件开发技术 10](#_Toc141203895)

[4.2.2控制层技术 12](#_Toc141203896)

[4.3 ESP32-CAM图像传输与通信部分 15](#_Toc141203897)

[4.3.1感知层技术 15](#_Toc141203898)

[4.3.2传输层技术 16](#_Toc141203899)

[4.3.3控制层技术 17](#_Toc141203900)

[5. 其他内容 18](#_Toc141203901)

[5.1 机械臂误差分析 18](#_Toc141203902)

# 设计需求分析

远程控制的家用机械臂为家庭生活带来了多方面的便利与用途。首先，它可成为家务助手，能轻松搬运重物、整理物品、清洁家居，从而减轻了家人的劳动负担，节省时间和精力。其次，远程控制功能让用户能够随时通过智能设备远程操控机械臂，无论身在何地，这对于频繁旅行或长时间不在家的人尤其有用。同时，家用机械臂还可以为教育和娱乐提供支持，帮助孩子学习编程和机器人技术，同时可以编程进行各种有趣的娱乐活动。此外，对于老年人或有特殊需求的家庭成员，机械臂的应用可以帮助他们照料日常生活需求，增加独立性，减轻家人的照顾压力。而对于喜爱创意制作和科技体验的用户，机械臂是一个绝佳的工具，可以参与各种创意活动并了解现代科技的应用。然而，在使用机械臂时，我们也要注意安全性和隐私保护，确保正确使用和维护设备，从而充分发挥家用机械臂的实用性与乐趣。

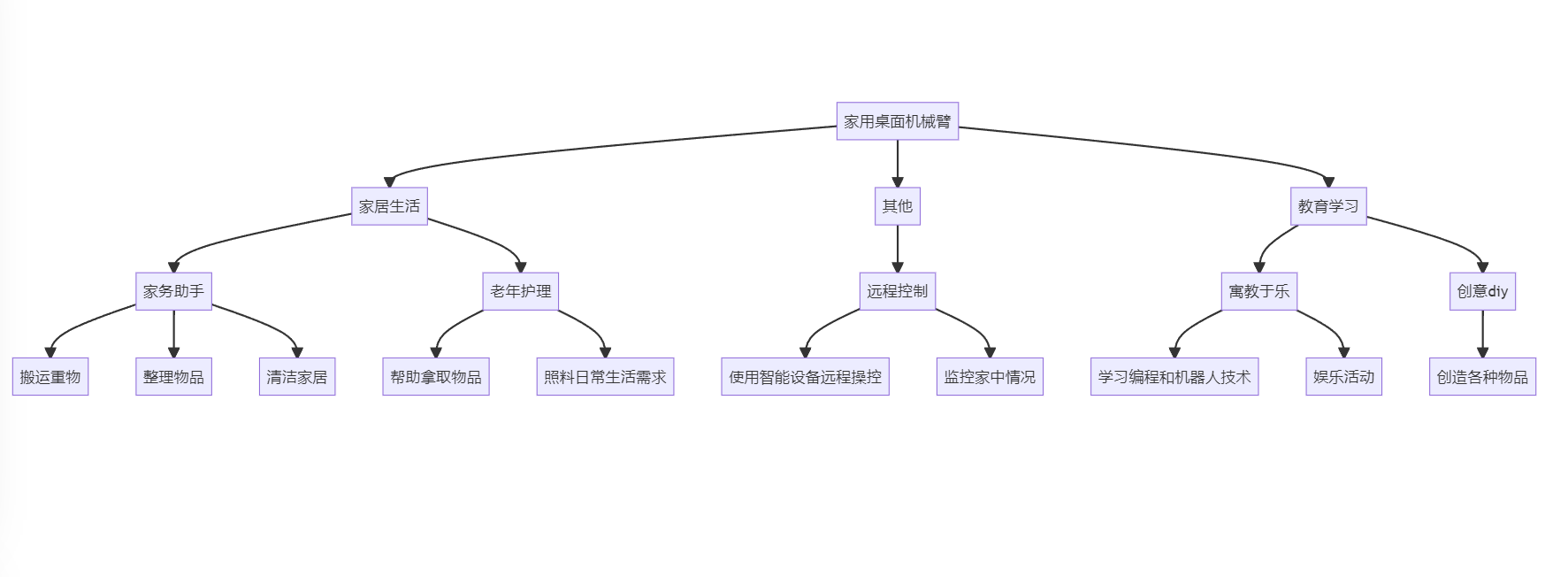


图1.1 家用机械臂的应用

我们发现，市面上常见的家用机械臂大多存在精度欠佳、操作方式简陋、难以调试、可扩展性差等诸多痛点，无法满足用户的实际使用需求。针对以上问题，我们设计出了基于oneos架构的机械臂控制系统，瞄准家用机械臂控制系统的蓝海，聚焦结构化、平台化设计，实现了远程控制客户端、机械臂运动学与oneos操作系统的有机结合。

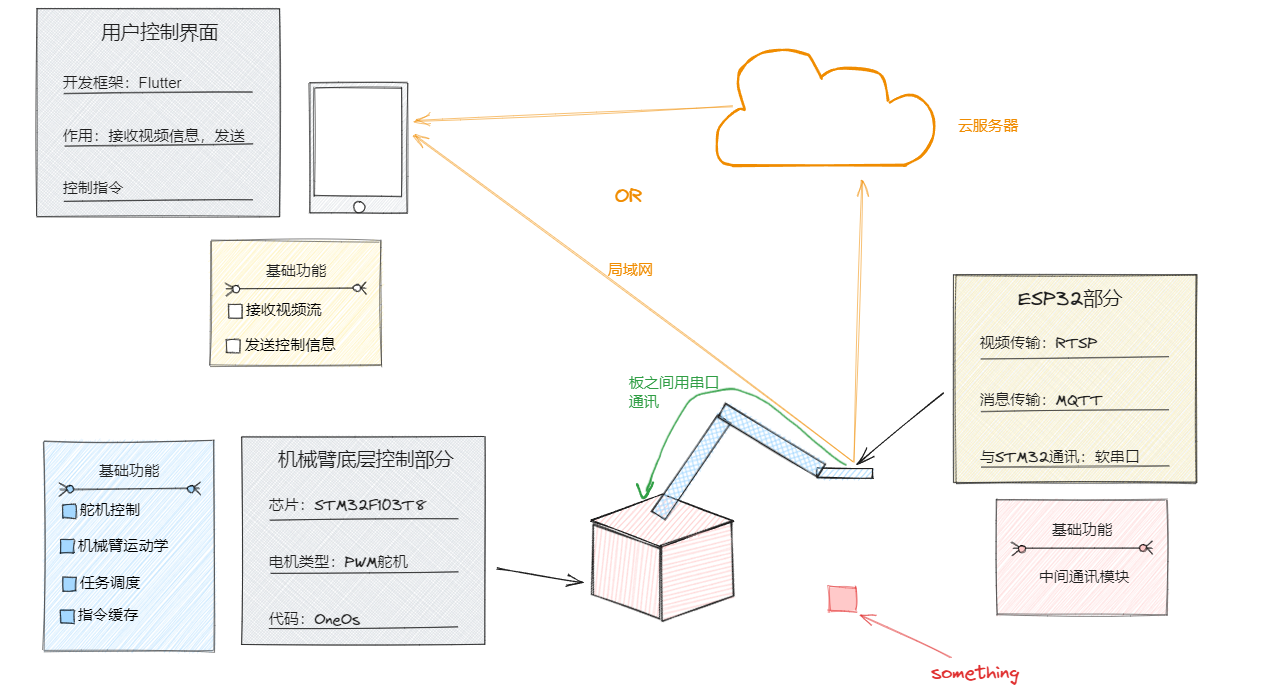


图1.2 整体方案初稿

在远程控制客户端，用户需要获取架设于机械臂关节末端的摄像机回传的视频，并能通过客户端向机械臂发出远程指令。视频传输部分的首要考量是其实时性和易扩展性，综合考量各项协议后我们使用了rtsp协议。rtsp协议的优势在于它提供了一种有效的实时流媒体控制方式，支持灵活的多媒体数据传输和交互性操作。远程指令传输使用了mqtt协议，其具有轻量级、可靠性和灵活性等多项优势。它适用于物联网和分布式系统，能够实现高效的消息传递，减少网络开销，同时支持灵活的通信模式和QoS级别，确保消息的可靠传递。

底层控制上，我们选用了stm32f103t8芯片，其仅有64kb flash与20 kb ram,这为控制系统的设计带来了不小的挑战。这个部分的首要考量便是精简代码，在OneOS内核裁剪的基础上还需要在程序中省下每一点内存。

机械臂运动学上，我们设计了一套完善的三轴码垛机械臂的运动学算法。设计之初的首要考量便是可扩展性。以坐标正逆解为基础逐层搭建涵盖基础舵机控制、运动范围检测、定位点插补、路径规划的整套框架。

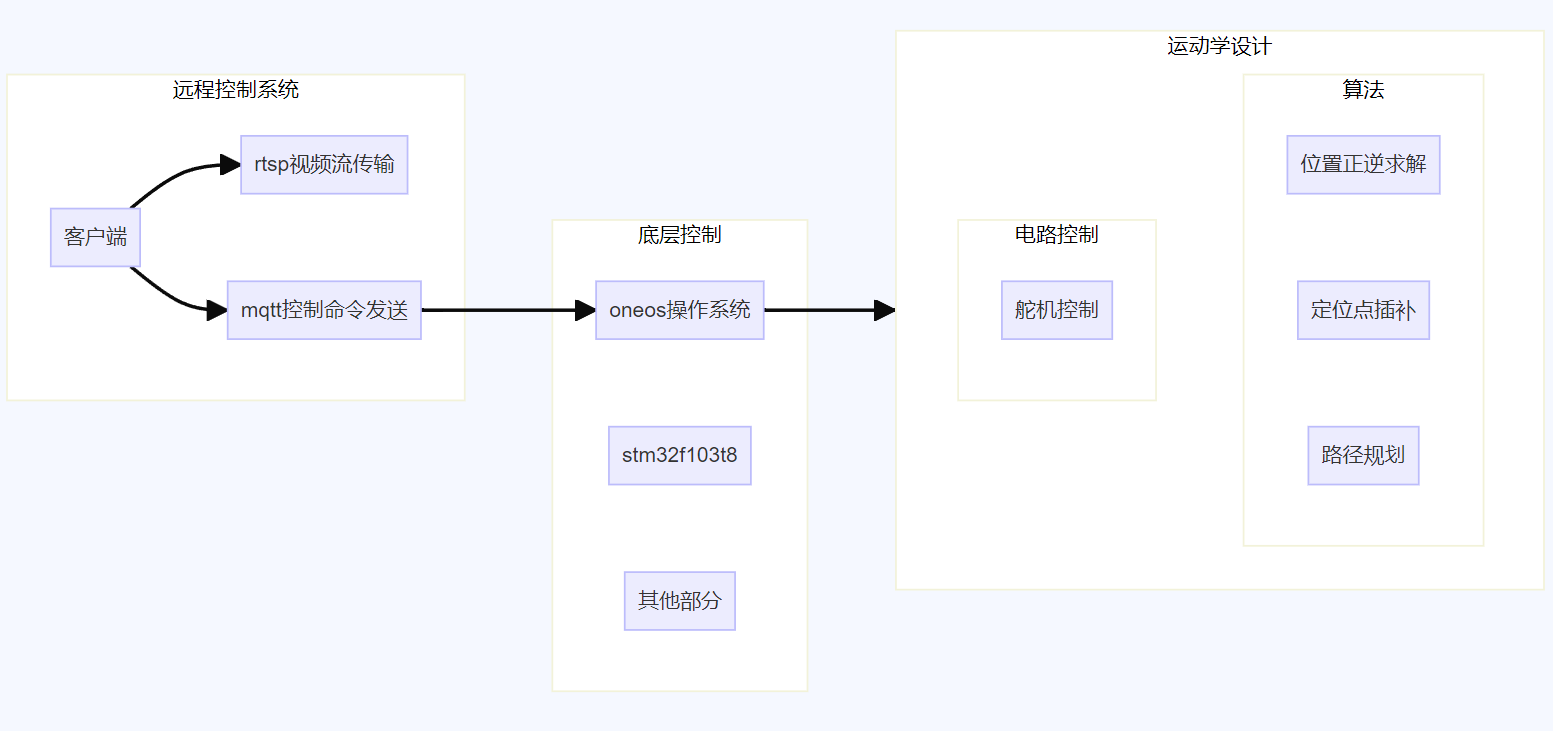


图1.3 机械臂控制系统框架

# 特色与创新

本作品在机械臂控制系统领域具备三大独特的特色与创新，为实现机械臂的智能化控制和远程操控提供了全新的解决方案。

首先，本作品基于OneOS系统设计了完善的底层控制代码。在设计过程中，我们充分发挥了OneOS可自由裁剪的优势，精心地对系统进行了最小化处理，同时保留调试接口和shell功能，使得OneOS能够最大限度地适配各种微控制器。通过在OneOS系统的基础上进行二次封装，我们成功地将用户从繁杂的硬件代码中解脱出来，使得机械臂的开发变得更加简洁高效。

其次，本作品进行了完善的运动学建模。通过精确的舵机控制、位置正逆求解、定位点插补以及路径规划等环节，我们实现了对机械臂的精准控制，其定点定位精度可达1mm的理论值。当然，在实际应用中，受到重心偏移、机械臂关节静摩擦、舵机精度等多方面因素的影响，实际精度约为5mm内，仍然非常令人满意。通过精心设计与优化的运动学模型，我们使得机械臂能够在各种复杂场景下高效运动，灵活应对各类任务，为机械臂的智能化控制奠定了坚实基础。

最后，本作品搭建了从云到端的完整框架，实现了远程控制系统。我们通过RTSP视频流传输和MQTT控制命令发送，将实时图像传输与机械臂控制命令的传递在客户端上实现。我们精心设计了客户端界面，使用户可以通过手机或电脑实时查看机械臂周围的场景，并通过界面上的按钮轻松发送控制指令。这种完整的云到端框架不仅方便用户实时监控机械臂的运动，更为远程控制提供了强大的支持，使得用户能够随时随地远程操控机械臂进行操作，极大地拓展了机械臂应用的可能性。

综上所述，本作品凭借基于OneOS系统的底层控制代码、精细化的运动学建模和完整的云到端框架，为机械臂控制系统领域带来了独特的贡献和创新，为实现智能化控制和远程操控提供了全新的解决方案。

# 功能设计

用户控制界面、ESP32-CAM图像传输与通信以及机械臂部分共同构成了这个项目的核心，实现了远程控制机械臂并查看周围环境的目标。用户控制界面为用户提供了实时图像传输和交互式控制按钮，让用户可以远程监控机械臂周围的场景，并通过界面上的按钮发送控制指令。ESP32-CAM负责将实时图像通过RTSP协议传输给用户控制界面，并通过MQTT网络通信接收用户的控制信号，再通过串口将指令传输给机械臂。机械臂部分则根据来自ESP32-CAM的控制指令执行相应动作，实现机械臂的移动和抓取物品功能。

## 机械臂代码

### 操作系统层

机械臂主要由两个进程控制，分别为指令输入进程和指令处理和运行进程。在设计之初，考虑到机械臂所追求的实时性，如果运行过程中出错，例如目标点无法到达、路径中某个点无法到达等，应当立刻输出错误信息，故将指令的处理和运行归为一个进程是必要的。而将指令输入进程和处理隔离成两个进程，主要目的是实现指令缓存的功能。代码中设计了指令环形队列，接收到的指令将依序存储在队列中。指令在输入进程中入队，在处理进程出队，是标准的生产者消费者问题，使用信号量作为同步机制。

### 运动学代码

本次使用的机械臂由舵机驱动。舵机区别于步进电机，特点在于精度较低、运动速度块。精度上，考虑理论误差，一般舵机的精度0.15°-0.35°，机械臂臂长140mm，140mm x 0.3° = 0.733mm，即理论不可能做到高于±1mm的定位精度。舵机另一大特点在于转动速度快。由此可以推出机械臂的运动特性为：

一、只能到达某些特定点。由于精度限制，基本可以看成机械臂可以准确到达的每个坐标点之间相隔大概1mm，实际表现为相隔较近的几个坐标实际都只能到一个点，几乎无法做到毫米级的精确移动。

二、点间运动时间极短。当设定依次抵达一系列插补点时，如果插补点间停留的时间过长，会很明显观察到抖动的情况。而过短的等待时间则无法接收准确的pwm波形。

在设计机械臂运动代码时，充分考虑了以上固有缺陷，做了诸多适配性调整。例如通过调整插补点间等待时间减轻抖动现象，动态调整舵机起转时间优化功率消耗等。

路径点插补上主要使用了五次曲线插补，路径规划使用bezier曲线实现沿任意路径的移动。

## 用户控制界面

用户控制界面是该项目的前端，使用Flutter框架开发，旨在提供一个直观、实时的机械臂控制界面，使用户能够轻松监控机械臂周围环境并远程控制其动作。通过与ESP32-CAM建立通信，实现了以下关键功能：

### 3.2.1 实时图像传输

用户界面与ESP32-CAM之间建立连接，接收来自相机的实时图像流。这样，用户可以通过手机或电脑实时查看机械臂周围的场景，确保操作的安全性和准确性。图像传输采用流式处理，确保实时性和流畅性。

### 3.2.2 交互式控制

用户界面提供交互按钮和控制面板，使用户可以轻松选择和发送指令。按钮直观易懂，控制面板提供多种运动和动作选项，用户只需点击按钮或选择选项，即可实现对机械臂的远程控制。

### 3.2.3 远程通信传输

用户在界面上的操作指令通过网络传输到ESP32-CAM，实现与机械臂的无线通信。通过MQTT协议建立稳定连接，确保控制指令的可靠传输，使用户能够实时与机械臂进行交互。

## 3.3 ESP32-CAM图像传输与通信部分

ESP32-CAM模块是这个项目中的重要组成部分，负责实现图像传输和通信功能。通过其强大的处理能力和丰富的通信接口，它实现了以下关键功能：

### 3.3.1 实时图像传输

ESP32-CAM采集摄像头的实时图像，并通过RTSP（Real-Time Streaming Protocol）协议将图像流推送给Flutter用户控制界面。这样，用户可以在实时的图像流中远程监视机械臂周围的环境，保证了用户对机械臂操作的实时感知。

### 3.3.2远程控制接收

ESP32-CAM通过MQTT（Message Queuing Telemetry Transport）协议与Flutter用户界面建立连接，接收用户发送的控制指令。这些指令经由MQTT传输到ESP32-CAM，为实现远程控制提供了可靠的通信机制。

### 3.3.3指令传输到机械臂

接收到来自用户界面的控制指令后，ESP32-CAM通过串口将指令传输给机械臂控制器。这将机械臂的动作控制从用户界面传递到机械臂实际执行层面，实现了远程控制机械臂的关键中转作用。

# 系统实现

（本章节主要描述实现功能所采用物联网技术架构，包括感知层技术、传输层技术、控制层技术、软件开发技术、云应用、数据挖掘和可视化应用等。）

## 机械臂控制程序

### 进程关系以及各自任务

指令输入进程负责从串口2接收esp32传向stm32的控制指令，处理后存入指令环形队列中。任务核心如下：

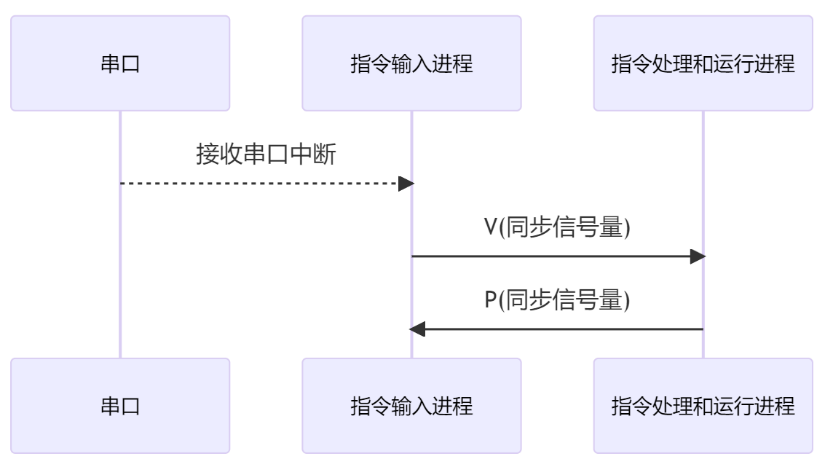
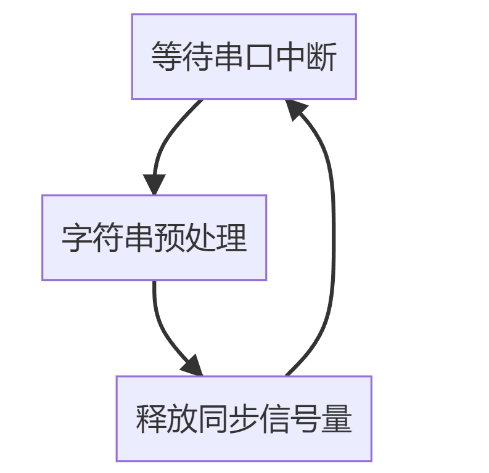


图4.1 指令输入进程流程图

指令处理和运行进程从队列中取出待执行的指令，处理并决定执行或抛弃。核心任务如下：

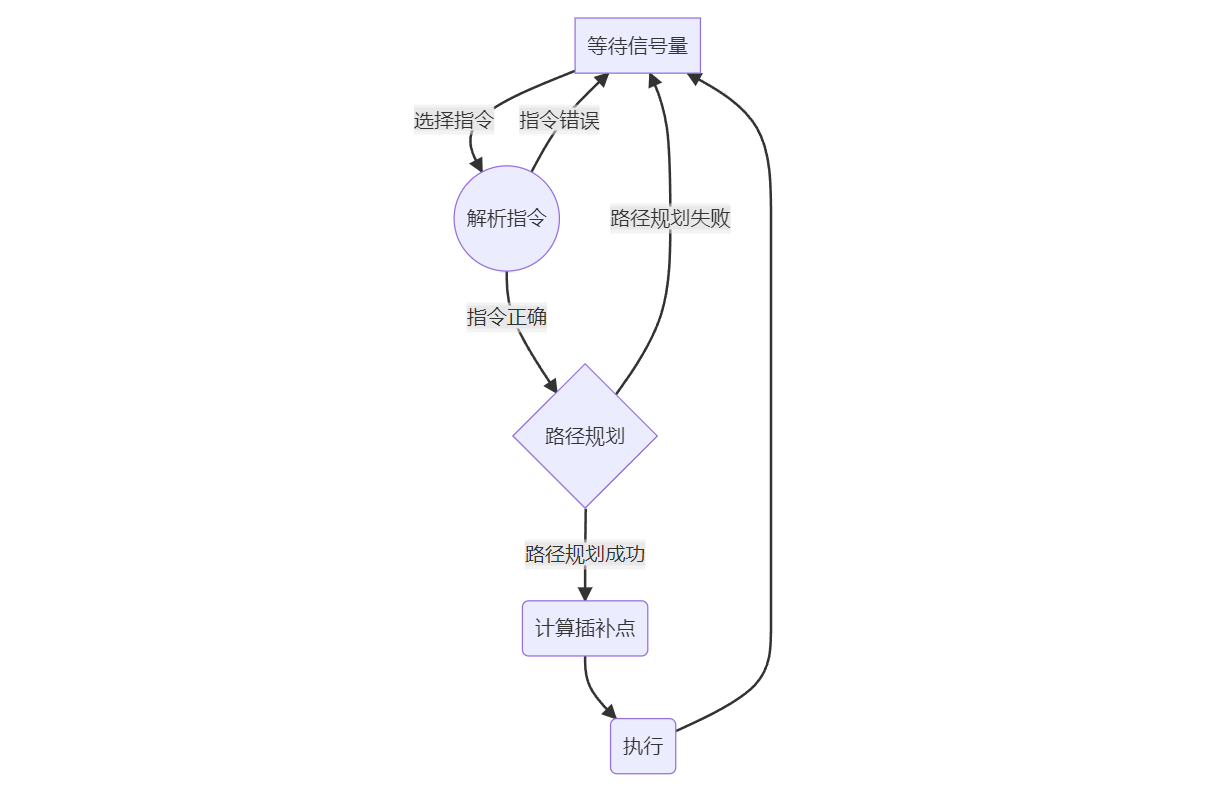


图4.2 指令处理和运行进程流程图

### 插补与路径规划

插补采用了五次曲线插补的方法。五次曲线插补具有插值计算得到的轨迹连续、插补生成的轨迹曲线光滑且无折点、易调整适等多项优点，是一种在数值计算和控制系统中常用的插补方法，用于平滑地连接两个数据点之间的曲线。它通过计算一条平滑的五次多项式曲线来实现插值。在机械臂控制系统中，五次曲线插补通常用于规划运动轨迹，使机械臂在运动过程中实现平滑、连续且符合预期的轨迹。其原理如下。

1. 确定起始点和终止点：首先确定待连接的两个数据点，分别称为起始点和终止点。

2．构造五次多项式：根据起始点和终止点已知，初末速度、加速度为0，构造一个五次多项式函数f(x) = a0 + a1 \* x + a2 \* x^2 + a3 \* x^3 + a4 \* x^4 + a5 \* x^5。其中由于初末速度、加速度为0，故可化简后用通项公式直接算出a0、a1、a2、a3、a4和a5，无需模糊逼近。

3. 插值计算：使用求解得到的五次多项式函数，在两个数据点之间的插值点上计算出中间的数据点。这些中间数据点构成了机械臂在运动过程中的平滑轨迹。

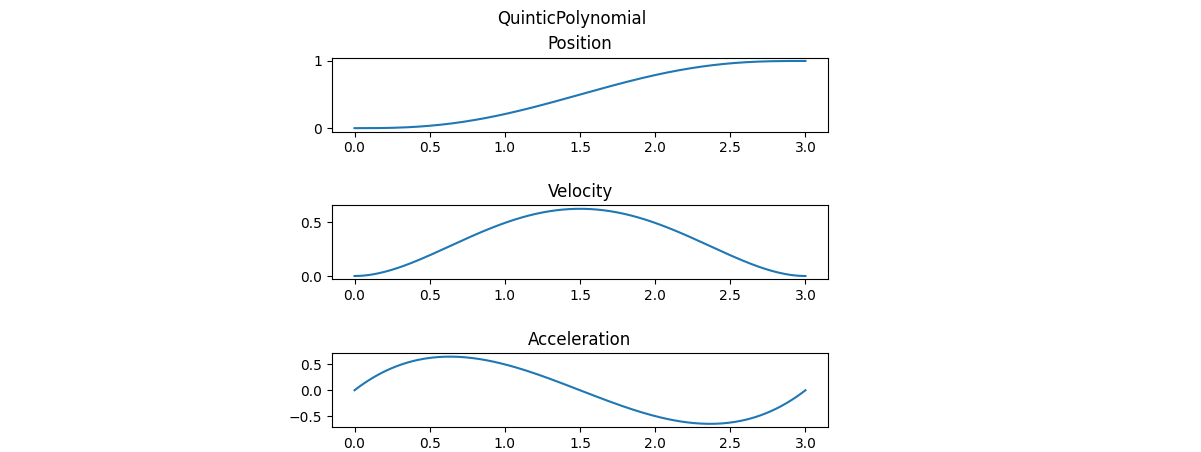


图4.3 五次曲线插补效果

路径规划沿用了五次插值的思路。首先需要得到路径在三维坐标下的函数表达式，直线运动为一个空间直线，任意曲线则由bezier曲线进行分段拟合，并计算得到需要插补点的个数。随后根据数目将一条拐点分别为(0,0)和(1,1)的五次曲线均匀划分，在该五次曲线上截取点，得到一系列0到1之间的向量。最后将该向量带入到路径的空间坐标函数中，即可实现均匀运动、慢入慢出的插补点计算。

## 用户控制界面

用户控制界面为用户提供了实时图像传输和交互式控制按钮。涉及到软件开发技术与控制层技术。

### 4.2.1软件开发技术

用户控制界面采用Flutter技术进行开发，Flutter是一个基于Dart语言的开发框架，它通过自绘UI的方式来实现跨平台开发，它具有以下特点：

1. 高度灵活： Flutter提供丰富的组件库和自定义组件功能，使得界面设计可以高度灵活，满足项目的定制化需求。
2. 快速开发： Flutter的热重载功能允许开发者快速看到界面的变化，加速应用开发过程。
3. 跨平台支持： Flutter支持iOS、Android等主流平台，稍加修改就能通过一套代码在多个平台上运行，方便以后在不同平台上进行部署。
4. 响应式布局： Flutter采用响应式布局，能够自动适应不同设备尺寸，提供一致的用户体验。

Flutter使用自己的渲染引擎绘制UI元素，而不是依赖于平台本身的UI组件。这种设计使得Flutter能够在不同平台上保持一致的UI外观和行为，从而实现了跨平台的统一性。同时Flutter采用的Dart语言是一种灵活、高效的编程语言，具备了跨平台开发的特性。它支持AOT（Ahead of Time）和JIT（Just in Time）两种编译方式，这使得Dart可以在不同平台上运行。在AOT编译模式下，代码会被预先编译成机器码，这样可以在iOS和Android等平台上获得更好的性能。而在JIT编译模式下，代码会在运行时即时编译成本地代码，这在开发过程中提供了热重载功能，支持快速迭代和调试。

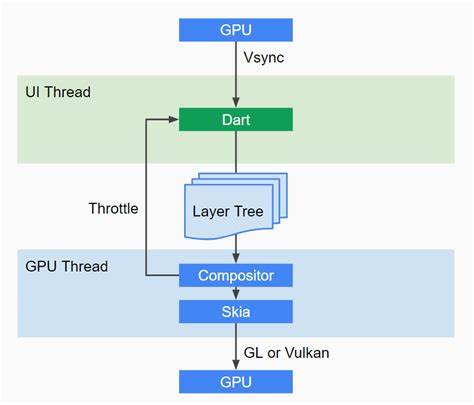


图4.4 Flutter底层原理

在软件开发技术上，用户控制界面主要有两个部分：视频显示部分和用户交互部分。我们将它们分别拆分成两个组件VideoScreen和MoveBox。

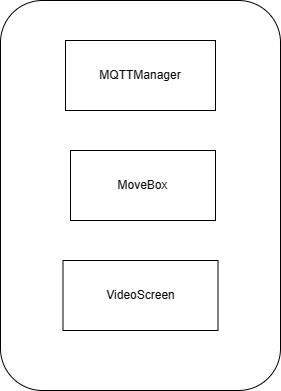


图4.5 用户控制界面结构

视频显示部分我们使用了flutter\_vlc\_player插件来实现视频播放功能。视频播放器界面是一个StatefulWidget，其中包含了一个VlcPlayer小部件用于展示视频内容。视频播放器界面还包含了一些与视频播放相关的逻辑，例如初始化视频播放器、控制视频播放等。整个界面是响应式的，会根据屏幕尺寸自动调整视频的宽高比。

用户交互部分我们使用内置的Material Design来设计布局，它提供了丰富的UI组件和工具，可以快速构建漂亮的用户界面。界面中使用了一些图片资源来表示机器人的移动方向。这些图片资源被加载为AssetImage对象，并通过Image小部件进行展示。

### 4.2.2控制层技术

在控制层技术层面，我们采用MQTT协议来传输用户控制信号。MQTT（Message Queuing Telemetry Transport）是一种轻量级、开放式、基于发布/订阅模式的消息传输协议。它被设计用于在低带宽、不稳定或网络延迟较高的环境中传输数据，特别适用于物联网和远程传感器应用。它具有以下特点：

1. 发布/订阅模式： MQTT采用发布/订阅模式，其中消息发布者（发布者）将消息发送到特定的主题（Topic），而订阅者（订阅者）通过订阅特定的主题来接收消息。这种松耦合的通信模式允许多个订阅者同时接收同一主题的消息。
2. 轻量级： MQTT协议设计简洁，消息头部较小，传输开销较小。它适用于资源有限的设备，如传感器、嵌入式系统和移动设备等，保证了高效的通信。
3. 可靠传输： MQTT支持三种消息传输质量等级，能对不同重要性的信息进行服务质量控制。
4. 连接性： MQTT支持长连接和保持会话，客户端和服务端之间可以保持持久的通信连接。这样，在连接中断后，客户端可以重新连接并恢复先前的订阅状态，确保消息不会丢失。
5. 遗嘱消息： 客户端可以设置遗嘱消息，在客户端异常断开连接时，服务器会将遗嘱消息发布到指定的主题。这样，可以实现客户端的在线状态监测。

在用户控制界面中，我们使用mqtt\_client库进行MQTT开发。首先初始化MQTT client，然后检测与MQTT服务器是否连接成功。如果连接失败，则会尝试重新连接。连接成功后，则会发布预先设定好的主题，用于与esp32-cam进行通信，并设置好QoS等级为0（最高），确保传输信息质量。

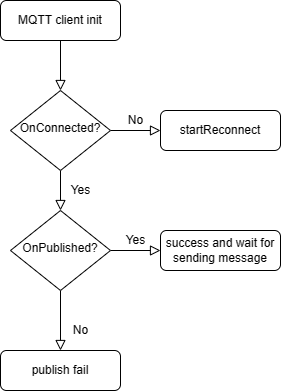


图4.6 用户控制界面MQTT流程

核心代码如下：

1. 对MQTT client的各种状态的回调函数进行绑定



1. 进行MQTT服务鉴权设置



1. 进行MQTT连接



1. 发布MQTT主题



## ESP32-CAM图像传输与通信部分

ESP32-CAM负责将实时图像通过RTSP协议传输给用户控制界面，并通过MQTT网络通信接收用户的控制信号，再通过串口将指令传输给机械臂。

### 4.3.1感知层技术

首先我们在代码中定义了一系列的参数，用于配置系统的各种设置。这些参数包括板子类型、帧持续时间、帧大小等。这些参数可以通过iotWebConf库进行配置和管理。

然后我们使用了OV2640库来初始化摄像头。通过cam对象，可以根据已设置的参数对摄像头进行各种设置和操作，如亮度、对比度、饱和度、特殊效果、白平衡、自动曝光等。

ESP32-CAM模块搭载的高性能摄像头能让我们通过摄像头采集机械臂周围环境的实时图像。该图像数据经过图像处理和编码后，通过RTSP（Real-Time Streaming Protocol）协议推流给Flutter用户控制界面。

### 4.3.2传输层技术

在视频传输方面，我们主要通过RTSP（Real-Time Streaming Protocol）来实现视频流的传输。我们使用MICRO-rtsp库来协助提供RTSP服务器的功能。推送RTSP视频流的流程如下：

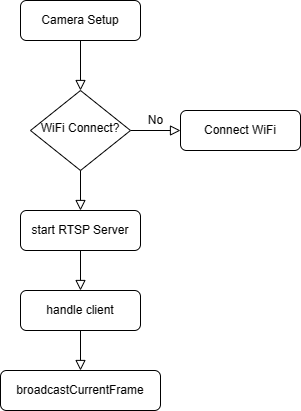


图4.7 RTSP服务器启动流程

1. 相机和WiFi配置：启用相机，并检查WiFi是否成功连接。
2. RTSP服务器初始化：创建了一个RTSP服务器对象camera\_server。通过这个对象，我们可以配置和管理RTSP服务器。rtsp\_server类是一个继承自WiFiServer的子类，用于创建和管理RTSP服务器。它接受一个摄像头对象作为参数，并在指定的端口上启动RTSP服务器。同时它还设置了一个定时器，用于处理客户端连接和请求。
3. 监听客户端连接：检查是否有新的客户端连接，并将其添加到客户端列表中。并处理来自客户端的请求
4. 视频流传输：调用broadcastCurrentFrame函数发送当前帧的视频流给客户端。

### 4.3.3控制层技术

在控制信息传输方面，我们使用MQTT协议来传输用户控制信号，总体流程与用户控制界面相似。但是用户控制界面是输入端，esp32-cam则是接收端。因此esp32-cam不需要发布主题，只需要订阅用户控制界面部分发布的主题即可，esp32-cam的MQTT流程如下：

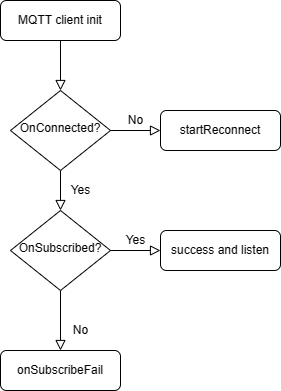


图4.8 Esp32-cam的MQTT流程

# 其他内容

## 机械臂误差分析

前文已在3.1.2章节处论证得受限于固有精度，机械臂静态定位理论精度约为1mm；同时也通过分析舵机运动特性指出了“抖动”现象的产生原因。不过这些都是理论得到的，实际机械臂调试中，机械结构也是重要因素。当前机械臂由于设计缺陷，转轴间摩擦力过大导致运动不畅。在10至30 mm的短距离运动过程中，静摩擦力占主导，稍远一些则主要是克服动摩擦力。静摩擦力略微大于动摩擦力，所以短距离运动从物理学上确实是更加吃力，也更容易出现“抖动”现象。当然最大的问题还是舵机本身。舵机底层采用PID控制，由电位器读取角度值。小角度下，一是电位器可能不太灵敏，二是由于PID原理输出功率会下降，空载固然还行，阻力矩很大时的小角度位移便很可能不甚理想。

除此之外，机械臂的重心偏移也会导致定位精度下降，毕竟本来空载和满载不同扭矩下，舵机可能发生滑牙甚至堵转的现象。加之重心偏移导致的底部转盘轻微倾斜，十分容易导致z轴上定位偏差高于预期。虽然代码有专门对z轴做了补正，但实际效果还是得因环境而定。最小偏差可能仅1mm，最大可能有2cm