

《物联网与嵌入式技术》

大作业课题报告

手势识别控制柔性臂——远程通讯

学 生 姓 名：_____赵思橙_____

学科、 专业：_____机械工程_____

学 号：_____22304125_____

完 成 日 期：_____2024. 4. 29_____

大连理工大学

Dalian University of Technology

目 录

| | | |
|-------|------------------|---|
| 1 | 技术路线..... | 1 |
| 2 | 开发软件介绍..... | 2 |
| 2.1 | EMQX 概述..... | 2 |
| 2.2 | MQTTBOX 概述..... | 2 |
| 3 | Python 编程设计..... | 4 |
| 3.1 | 数据发送端设计..... | 4 |
| 3.1.1 | 连接服务器..... | 4 |
| 3.1.2 | 创建独立的循环通讯线程..... | 4 |
| 3.1.3 | 消息发送..... | 4 |
| 3.2 | 数据接收端设计..... | 4 |
| 3.2.1 | 回调函数..... | 4 |
| 3.2.2 | 关联发送端主题名..... | 5 |

1 技术路线

本设计通过远程摄像头获取手势图像，经手势识别程序识别和分析用户的手部动作，通过网络开源的物联网云平台传递数据发送给柔性臂端，根据接收到的数据控制柔性臂的动作。技术路线如图 1.1 所示：

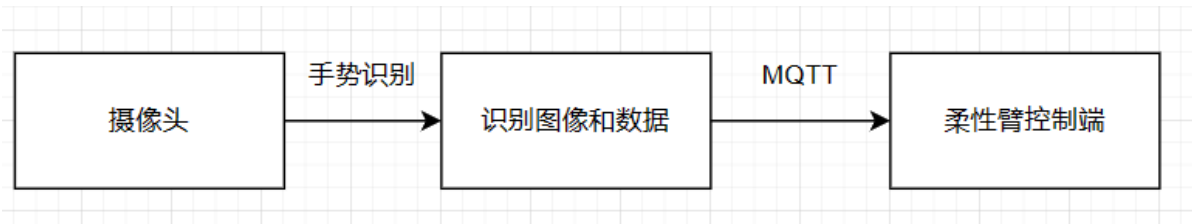


图 1.1 手势识别控制柔性臂技术设计总路线

本文利用 EMQX 作为服务器实现不同设备间的数据传输。

2 开发软件介绍

2.1 EMQX 概述

EMQX 是一款开源的大规模分布式 MQTT 消息服务器，功能丰富，专为物联网和实时通信应用而设计。EMQX 支持 MQTT 并发连接数高达 1 亿条，单服务器的传输与处理吞吐量可达每秒百万级 MQTT 消息，同时保证毫秒级的低时延。EMQX 支持多种协议，包括 MQTT、HTTP、QUIC 和 WebSocket 等，保证各种网络环境和硬件设备的可访问性。EMQX 还提供了全面的 SSL/TLS 功能支持，比如双向认证以及多种身份验证机制，为物联网设备和应用程序提供可靠和高效的通信基础设施。内置基于 SQL 的规则引擎，EMQX 可以实时提取、过滤、丰富和转换物联网数据。此外，EMQX 采用了无主分布式架构，以确保高可用性和水平扩展性，并提供操作友好的用户体验和出色的可观测性。EMQX 支持 Ubuntu、CentOS、Linux、macOS 和 Windows 等多个系统的下载和使用，同时 EMQX 开源版可以免费使用，可以使没有系统学习过的用户可以快速搭建 MQTT 服务器。

EMQX 作为物联网应用开发和物联网平台搭建必须用到的基础设施软件，主要在边缘和云端实现物联网设备互联与设备上云，提供物联网设备接入、协议处理、消息路由、数据存储、流数据处理等核心能力。EMQX 消息传输流程如图 2.1 所示：

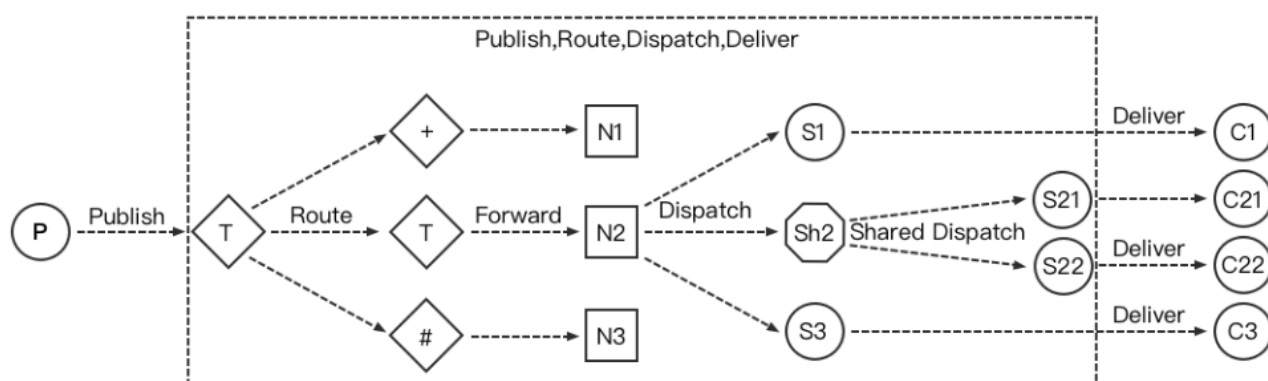


图 2.1 EMQX 消息传输流程

2.2 MQTTBOX 概述

MQTTBOX 是 Sathya Vikram 个人开发的 MQTT 客户端工具，采用了 Electron 跨平台技术，界面简单直接，支持多个客户端同时在线，但客户端之间的切换、互发消息等

交互还是有一定不便。MQTT Box 借助 Chrome 有很强大的跨平台特性和简单的负载测试功能。利用 MQTT 实现的不同设备间的简单通讯如图 2.2 所示：

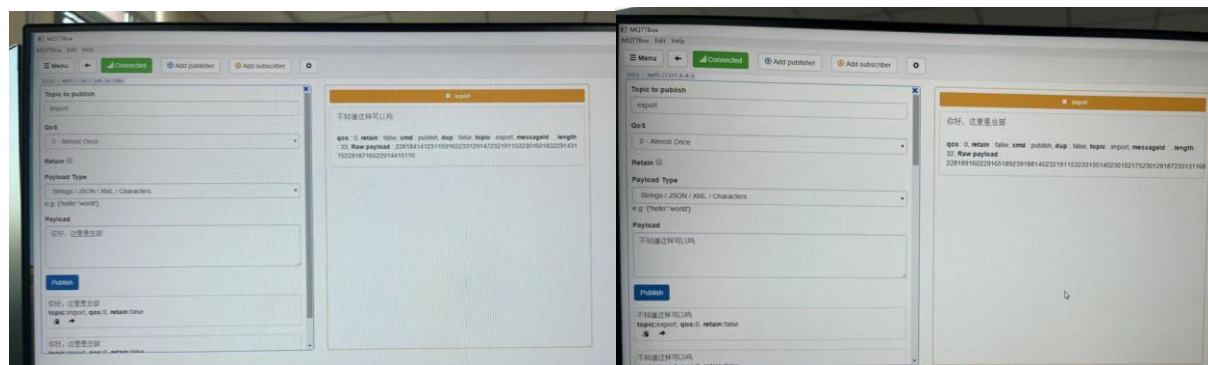


图 2.2 利用 MQTTBOX 实现两个设备间的通讯

简单负载测试结果如图 2.3 所示：

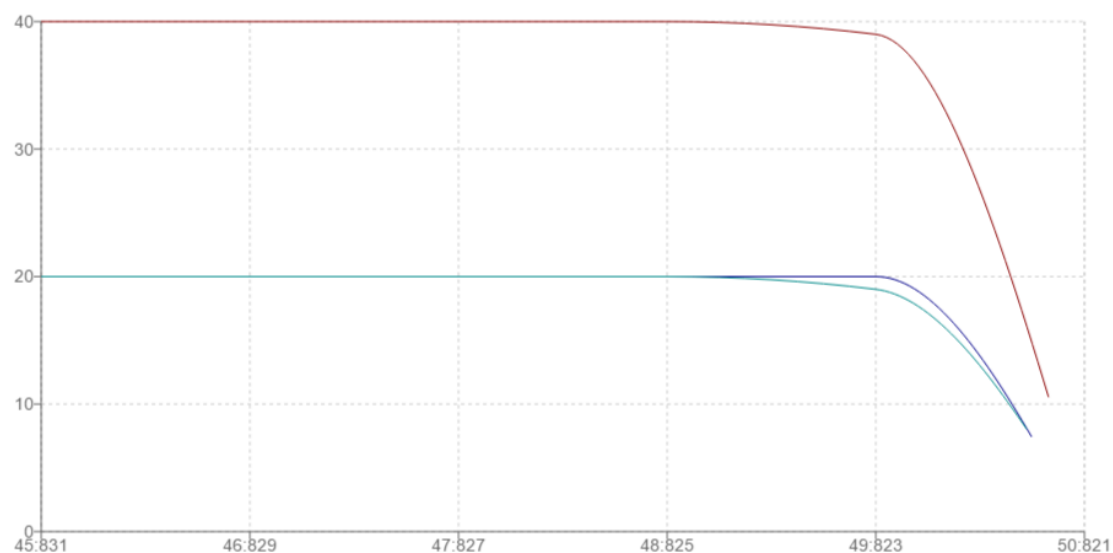


图 2.3 MQTTBOX 简单负载测试

3 Python 编程设计

利用 Python 编程首先需要安装 paho.mqtt 库。如图 3.1 所示

```
import paho.mqtt.client as mqtt
```

图 3.1 使用 paho.mqtt 库

3.1 数据发送端设计

3.1.1 连接服务器

连接服务器代码如图 3.2 所示，其中 broker_ip 为服务器的 ip 地址，broker_port 为服务器的端口号，一般为 1883，timeout 为延迟，一般默认为 60。

```
self.client.connect(self.broker_ip, self.broker_port, self.timeout)
```

图 3.2 连接服务器

3.1.2 创建独立的循环通讯线程

利用内置的 loop_start 函数可以开启一个独立的循环通讯线程去监控网络通讯状态。代码实现如图 3.3 所示

```
self.client.loop_start() #开启一个独立的循环通讯线程。
```

图 3.3 建立独立的循环通讯线程

创建通讯线程还可以使用 loop 函数，但是该函数仅实现一次通讯线程，需要反复调用。Loop_start 函数可以实现堵塞式的通讯线程循环，但是会无法进行其他工作。

3.1.3 消息发送

消息发送的实现如图 3.4 所示。Topic 是和服务器连接的主题名称，payload 是本次发送的信息，qos 和 retain 在 MQTT 的开源版本下使用默认值即可。

```
self.client.publish(topic, payload=payload, qos=qos, retain=retain)
```

图 3.4 消息发送

3.2 数据接收端设计

3.2.1 回调函数

每接受一次数据就执行一次回调函数，本设计只需要输出接收到的数据即可。

```
def default_on_message(self, client, userdata, msg):  
    print(msg.payload.decode('utf-8'))
```

图 3.5 回调函数

3.2.2 关联发送端主题名

设置和发送端相同的主题名才可以接收到数据。

```
self.client.subscribe(self.topic)
```

图 3.6 设置接收端主题名