

**THE HONG KONG UNIVERSITY OF**

**SCIENCE AND TECHNOLOGY**

**ENGG 3960Z**

**Part 1**

**作者: Zoe Han (Han xinyi) ； Fallengold (GUO, Zilin); Lawrence Ruan;**

**指导/提议：Li Gaoyang**

**⾹港科技⼤学ENTERPRIZE战队** [**robomasterhkust@gmail.com**](mailto:robomasterhkust@gmail.com)

**Github链接：** [**https://github.com/hkustenterprize/RM2024-RosComm**](https://github.com/hkustenterprize/RM2024-RosComm)

**下位机开源链接：https://github.com/hkustenterprize/RM2024-SerialDriver-STM32**

目录

1概述

* 1. 背景&目标

2.软件设计

2.1系统架构

2.2主要思路

2.3重点功能

3．研发迭代过

3.1测试记录

3.2重点问题解决

## **1.1** 背景&目标

在赛场上对于机器人的控制有很高的实时性的控制要求，上位机进行视觉解算的时候也应该即使有及时的反馈。在之前赛季中，视觉算法的上下位机间通讯的频率并不是很理想，有限的通讯带宽限制了数据的更新和反馈，从而一定程度地影响了控制的实时性。这个赛季对上下位机间通讯做了重新设计。希望能够达到更高的通信速率并且稳定通讯质量，使可用数据的提高。

需要注意的是本套上位机代码需要和合适的下位机代码配套使用。具体可以参考队伍的下位机代码开源，合适的下位机处理才能准确接受大量数据。具体可以参考：

<https://github.com/hkustenterprize/RM2024-SerialDriver-STM32>

## **2.**软件设计

## 2.1系统架构

图示

描述已自动生成

**1. 底层串口驱动（Serial Driver）**

**功能描述：**

串口驱动层的主要功能是与物理串口设备进行低层次交互。其设计目标是将串口通信细节与上层ROS2逻辑隔离出来，确保串口操作的独立性和可移植性。该层提供的功能包括：

* **串口读写操作**：通过指定的串口接口（如 /dev/ttyUSB0）进行数据的读取和写入。
* **串口配置管理**：支持对串口的配置进行动态修改，包括波特率、起始位、数据位、停止位、校验位、是否启用硬件流控等参数。
* **串口打开/关闭控制**：提供接口打开和关闭串口设备，并对设备状态进行监控。
* **数据统计与日志记录**：在运行时对收发的数据进行统计，并记录相关日志以便调试和性能评估。
* **错误处理**：包括对串口超时、数据丢失、通信错误等事件的处理。

**设计特点：**

* **硬件无关性**：通过封装硬件相关的串口操作，使得驱动层可以在不同的硬件平台上运行（例如不同的Linux发行版或硬件架构）。
* **线程安全**：考虑到串口读写可能是异步的，设计时需确保该层的操作是线程安全的，避免数据读写冲突。
* **接口封装**：提供统一的接口供上层调用，隐藏底层串口通信的复杂性。

**2. 上层ROS2节点（RM Serial Driver）**

**功能描述：**

上层的ROS2节点（RM Serial Driver）是系统与ROS2框架的接口部分。其核心职责是收集底层串口驱动层传递的原始数据，并根据ROS2的通信机制与其他节点进行交互。具体职责包括：

* **处理底层数据**：从串口驱动接收的原始数据进行解析、处理，并根据需要将其转换为ROS2消息格式。
* **与其他ROS2节点交互**：通过ROS2的发布/订阅机制，与其他节点进行数据交换。发送频率至少支持四个1k的发送频率（即每秒至少发布4000次消息），并通过callback机制接收其他节点的数据。
* **定时发布数据**：通过ROS2的定时器机制（rclcpp::Timer），为串口数据设定发布的频率。
* **实时数据处理**：通过订阅其他节点的消息，实时处理反馈数据，并通过串口驱动层发送到串口设备。

**设计特点：**

* **高频发布**：为满足高频率的数据发布需求，节点需要支持频率控制，并保证在高负载情况下的性能稳定性。
* **实时性要求**：由于串口通信的实时性要求较高，节点需要设计高效的数据处理流程，避免阻塞或延迟。
* **错误处理与恢复机制**：在串口通信异常时，节点需要能够检测并尝试自动恢复，确保系统的鲁棒性。

2.2主要思路

在接受端的设计中，采用了类似于**环形缓冲区**（circular buffer）的思想，并在此基础上引入了**双重 CRC 校验**（Cyclic Redundancy Check）。这样设计的目的是为了确保数据传输的完整性和可靠性，同时提高数据处理的效率。下面我们详细分析这种设计，并进一步拓展其概念。

**环形缓冲区的概念**

环形缓冲区是一种特殊的缓冲区结构，它的特点是：当缓冲区的尾部被写满时，新的数据可以从缓冲区的头部开始覆盖旧的数据（如果需要）。这种结构特别适合处理**流式数据**，如串口通信中不断接收的字节流。环形缓冲区在处理数据时，不需要频繁移动数据，只需更新指针（读指针和写指针），这使得它在性能和空间利用率方面具有很大的优势。

**接收数据的流程**

**数据接收并存入缓冲区**：

当串口接收到数据时，第一时间将这些数据存储到环形缓冲区中，而不立即进行处理。这保证了数据不会丢失，并且可以最大限度地提高接收效率，避免处理数据时阻塞新的数据进入。

**处理数据流程**：

缓冲区中的数据是按顺序进行处理的，首先会检查剩余数据是否满足数据包头（header）的长度要求。如果剩余数据足够长，可以包含一个完整的包头，则进行解包操作。包头中通常包含数据包的长度、协议标识符等信息，通过解析包头，能够知道完整数据包的长度。

**双重 CRC 校验**：

CRC 校验是一个非常重要的步骤，用于确保数据在传输过程中没有发生错误。这里采用了双重 CRC 校验机制：

* + 1. **第一个 CRC 校验**：在解包包头后，首先对包头部分进行 CRC 校验，确保包头信息的完整性。如果包头的 CRC 校验失败，说明包头损坏，那么当前数据包无效，直接丢弃这个包头部分的数据，等待新的数据包。
    2. **第二个 CRC 校验**：如果包头的 CRC 校验通过，那么接下来会判断缓冲区中是否有足够的剩余数据来组成一个完整的数据包。如果有足够的数据，就对整个数据包（包括包头和有效载荷）进行 CRC 校验，以确保整个数据包的完整性。

**数据包处理与分流**：

如果整个数据包的 CRC 校验通过，说明该数据包是完整且有效的。此时，数据包会被解析，解码出有效的数据，并根据数据包的类型、协议标识符等信息，将数据分流到对应的 ROS 2 话题（topic）中，通过 ROS 2 的发布者（publisher）发送给其他节点进行处理。

处理完的数据会从环形缓冲区中移除，以便腾出空间给新的数据。

**处理不完整的数据包**：

如果当前缓冲区中的数据不够长，无法组成一个完整的数据包（即剩余数据长度不足以覆盖包头或有效载荷），则会**保留这些数据**，等待下一轮新的数据进入缓冲区后，再次进行处理。这样设计的目的是避免由于数据包分片导致的数据丢失。

2.3重点功能

3.1节点初始化 (SerialDriverNode 构造函数)：

使用特定的配置（波特率、数据位、停止位、校验位）初始化串口。

设置 ROS 2 的发布者、订阅者和客户端，用于与其他节点通信。

一个持续循环检查串口是否打开，如果没有打开则尝试打开串口。

3.2接收数据 (SerialDriverNode::rx)：

rx 方法在一个单独的线程中持续从串口读取数据。

尝试通过 decode 方法解码接收到的数据包，该方法会验证 CRC 校验并处理数据。

数据处理的主体部分，对于上位机串口接收的数据进行高速的处理。

* 1. 解码数据 (SerialDriverNode::decode)该方法解析接收到的数据提取数据包头和有效载荷。

通过 CRC16 校验来验证数据的完整性。

如果数据有效，则对其进行分类和处理；否则，记录错误日志并丢弃损坏的数据。

3.3分类数据 (SerialDriverNode::classify)：

classify 方法处理解码后的数据，并根据目标颜色和其他字段更新参数或通过 ROS 2 发布者发布消息。

使用 tf2\_ros::TransformBroadcaster 广播坐标变换。

1. 发送数据 (SerialDriverNode::GimbalCommand\_CB 和 SerialDriverNode::transmit)：
   * 节点根据接收到的消息向云台系统发送控制命令。
   * GimbalCommand\_CB 方法准备数据，附加 CRC 校验并将其添加到发送缓冲区。
   * transmit 方法从缓冲区中取出数据并通过串口发送。

3.4错误处理与恢复：

如果串口连接出现问题（例如读取/写入失败或 CRC 错误），节点会尝试关闭并重新打开串口。

ROS 2 服务客户端：节点通过服务客户端与 armor\_detector 服务进行交互设置参数，并通过 /tracker/reset 服务重置追踪器。

**3.** 研发迭代过程（ROS communication）

研发过程中，不同阶段或者版本所完成的功能、发现的问题、问题定位分析以及改进解决方案说明。

以下格式仅作为参考，可自行更改格式。

简介：上位机NUC到下位机MCU通讯基于usb转uart串口进行通讯。接收端使用环形缓冲提高数据接收速率。通过报文头和报文尾的十六位CRC校验来核实数据正确与否。

第一阶段：

基于ch343芯片，最开始使用Linux本身的CDC-ACM串口模组，可以直接使用ROS2官方serial driver 的通讯包，波特率上限局限在460800，并且有大量的CRC错误无法消除，结点的收发频率大约500hz，有效信息的收发频率最低有80-90hz。将问题定位到底层串口驱动的有过多的线程锁导致效率下降，以及第二阶段：

更换思路，放弃使用官方的串口通讯库，能够操作的空间太小了。更换了不可调整串口设置的CDC-ACM串口驱动，改成自由度更高的VCM驱动。对于串口的配置进行更加详细的代码控制。参考Linux的serial.h 等系统文件重写了Linux的串口调用的底层驱动。目标是精简优化底层驱动之后，提高基础带宽到2M，并且增加硬件流控，使用RTS和CTS信号稳定通讯。

第三阶段：

在针对第二阶段的压力测试的时候发现，由于芯片质量问题，RTS和CTS电平信号处理不当，容易触发芯片的自我保护导致线路瘫痪。放弃使用硬件流控，转向更改应用层的程序设计来实现最大化接收数据。前后分别使用双缓冲和环形缓冲进行数据处理。分成两个进程分别负责接收和发送，并且使用 pipe 进行通信来保障环形缓冲的安全。最后达到了UART通讯双工2M波特率，节点收发频率1khz,CRC错误率百分之零点三。

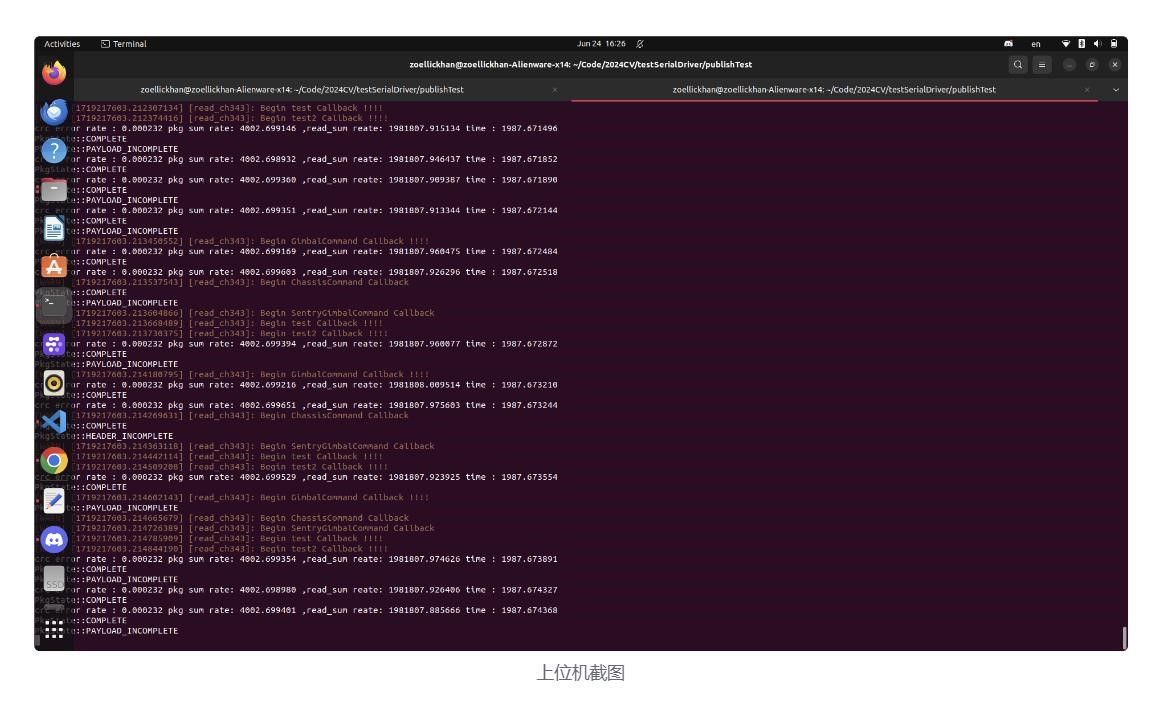
## **3.1** 测试记录

对机器人各项功能点进行测试验收，记录测试环境、测试设备等信息。



**3.2**

版本迭代过程记录



迭代过程的时间颗粒度参考兵种、技术点里程碑时间，推荐以月为单位，避免时间跨度太小或过大。

|  |  |  |
| --- | --- | --- |
| 版本号或阶段 | 功能或性能详细说明 | 完成时间 |
| V1.0 | 基于ch343芯片的ROS COMM 底层驱动 | 2024.2.26 |
| V1.1 | 尝试加入硬件流控，双工3M波特率，在ROS2架构运行 | 2024.4.2 |
| V1.1 | Serial Driver 无流控，环形缓冲，双工2M，1KHz完成压力测试 | 2024.6.24 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## **3.2** 重点问题解决记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 问题描述 | 问题产生原因 | 问题解决方案  &实际解决效果 | 机器人版本号或阶段 | 解决人员 |
| 1 | 中途测试实际收发包有百分之四十缺失 | 环形缓冲没有彻底解码，每次收发有冗余数据留在缓冲区 | 增加循环检查，每次收发都确保无冗余数据 | V1.0 | 嵌入式工程师： 郭子霖算法工程师：韩欣颐 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 问题描述 | 问题产生原因 | 问题解决方案  &实际解决效果 | 机器人版本号或阶段 | 解决人员 |
| 2 | 有时下位机接收数据有半秒的延时 | 上层代码问题，导致只有缓冲区满数据才会进行打包发送 | 更改了 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 参考文献

参考文献的规范

1. 书籍、期刊论文等适用通用论文引用规范，格式参考【1】
2. RM论坛开源文档等电子文献适用论文引用规范中的网页部分（要求附上链接），格式参考【2】
3. 格式参考：
4. [《机械设计与研究》征稿启事[](https://www.qk.sjtu.edu.cn/mdr/CN/10.13952/j.cnki.jofmdr.a4399)J]. 机械设计与研究, 2017, 33(05): 3-.
5. 王明亮.关于中国学术期刊标准化数据库系统工程的进展 ［EB/OL］. http://www. cajcd.edu.cn/pub/wml.txt/980810-2.html, 1998-08-16/199810-04.

