doi:10.3772/j.issn.1002-0470.2019.09.006

基于 ROS 和 IgH EtherCAT 主站的 SCARA 机器人控制系统①

徐建明② 吴蜀魏 吴小文 张文安 俞 立

(浙江工业大学信息工程学院 杭州 310023)

摘 要 针对在开源软件基础上研制网络化选择顺应性装配机器手臂(SCARA)控制器的需求,设计了一种 Linux 系统下基于开源机器人操作系统(ROS)和以太网控制自动化技术(EtherCAT)的 SCARA 机器人控制系统。基于开源 IgH EtherCAT 主站,开发了 IgH 用户层程序并移植到 Xenomai 的实时微内核中,实现 EtherCAT 通讯。在 ROS 平台下,通过建立统一机器人描述格式(URDF)模型,结合 moveit! 实现运动规划和求解,根据 ros_control 的配置开发了硬件抽象节点,利用 QT 插件设计开发了人机交互界面节点,主要包含人机界面通讯协议、数据库界面、编程界面等模块,并根据不同坐标系下的控制命令开发了相应的命令处理节点。最后,通过对实时任务周期和同步信号抖动时间的测量,以及机器人控制实验,验证了所设计的 SCARA 机器人控制系统的实用性。

关键词 机器人控制系统(ROS),以太网控制自动化技术(EtherCAT),IgH,Linux

0 引 言

工业机器人发展至今已比较成熟,工业机器人技术正朝着模块化、协同化、网络化的方向发展。随着装备制造业的迅猛发展以及科学技术的不断突破,对工业机器人的要求也越来越高,加快工业机器人技术的发展对推动“中国制造 2025”具有重要意义[1]。

目前国内外的选择顺应性装配机器手臂(selective compliance assembly robot arm, SCARA)控制系统主要可以分为两类,一类为库卡、ABB、发那科、安川等传统工业机器人厂商研发的控制器,它们大多数是采用专用硬件、软件和机器人控制语言,具有相对封闭的系统结构,与外部设备连接的开放性不够[2]。另一类为基于德国 3S 的 CODESYS、中国固高的 Otostudio、美国 Keil 的 RTX 等商用软件平台上开发的工业机器人控制系统,如李辰[3]采用 CODESYS 开发了 SCARA 机器人控制系统。杨帆[4]利用 CPAC 和 Otostudio 开发的 SCARA 机器人控制系统并对控制策略做了相关改进。毕鲁雁等人[5]基于 RTX 实时操作系统设计了工业机器人控制系统,这类控制系统需要支付昂贵的授权费用。随着工业机器人应用越来越广泛,有必要研发一个采用开源平台的成本低、开放性好的 SCARA 机器人控制系统。

近年来,开源操作系统(尤其是 Linux)的实时性得到了显著的提升,出现了许多 Linux 实时改造方案,如 Xenomai、RT Linux、RTAI 和 AquoSA,其中的 Xenomai 是一种采用双内核机制的 Linux 内核的开源实时扩展,将一个实时微内核嵌套到 Linux 内核中,并且优先级高于 Linux 内核,微内核里的实时任务会被优先执行[6,7]。实时工业以太网控制自动化技术(Ethernet control automation technology, EtherCAT)是由德国倍福公司提出的,它具有拓扑灵活、应用简易、高性能等特点,是目前速度最快的实时工业以太网技术之一,被广泛应用于工业机器人控制系统[8]。IgH 是 EtherLab 推出的一套集成在

① 国家自然科学基金-浙江省自然科学基金联合基金两化融合(U1709213),国家自然科学基金面上(61374103)和浙江省自然科学基金重点(LZ15F030003)资助项目。

② 男,1970 年生,博士,教授;研究方向:迭代学习控制,电机伺服控制技术,机器人控制技术等;联系人,E-mail:xujm@zjut.edu.cn

(收稿日期:2019-01-03)

Linux 内核上的 EtherCAT 主站开源框架,该框架对于 EtherCAT 协议实现更完整[9]。开源机器人操作系统(robot operating system, ROS)是由 Willow Garage 公司发布的一款开源机器人操作系统,其控制体系能够与机器人建模与运动学解算良好地结合在一起,并且通过统一平台,使机器人的功能模块化,减少了工业机器人控制系统的开发工作量[10-12]。

本文以 4 自由度的 SCARA 机器人为研究对象,基于 EtherCAT 技术和 ROS 开发平台设计了集控制、人机交互一体的 SCARA 机器人控制系统。EtherCAT 主站使用 IgH 开源框架,IgH 用户层运行在 Xenomai 实时微内核中并通过线程通信与 ROS 进行数据交互,Xenomai 为 IgH 主站提供了实时保障。使用 ROS 平台提供的 moveit! 实现运动规划、运动学求解,以及使用 ros _ control 框架实现插值计算。通过 ros _ qtc _ plugin[13]插件在 QT 中设计开发了带人机界面的 ROS 节点,人机界面节点包含数据库、编程、点动等模块,最后根据界面控制命令设计了人机界面通讯协议和界面命令处理节点。

1 控制系统组成

1.1 控制系统硬件组成

控制系统硬件通过 EtherCAT 总线将上位机直接与支持 EtherCAT 通信的工业机器人伺服驱动器相连。上位机为装有 Linux 系统的带触摸屏工控机,工业机器人为 SCARA 机器人,机械臂末端带有夹爪。控制系统硬件组成如图 1 所示。

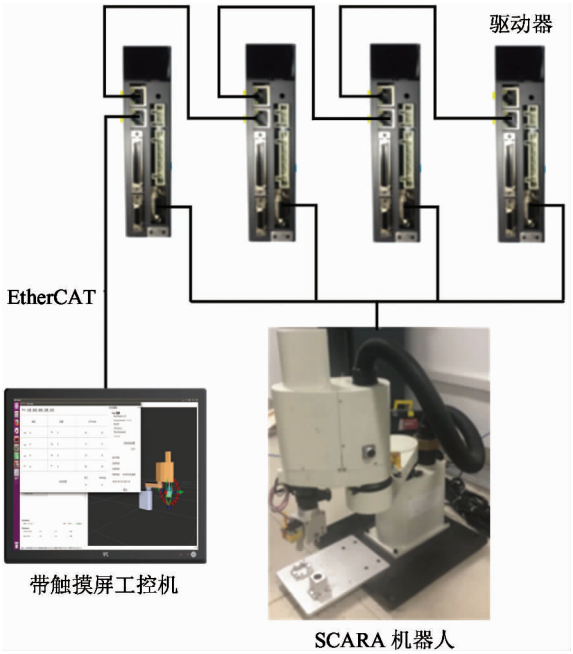


图 1 控制系统硬件组成

1.2 控制系统软件框架

控制系统软件框架运行在 Linux 系统下,包括 ROS 和 IgH 2 个部分,ROS 包含人机界面、命令处理节点、moveit! 和 ros _ control 4 个部分,IgH 通过运行在 Xenomai 实时微内核中实现 EtherCAT 主站,利用 EtherCAT 总线实现与驱动器的网络通信。控制系统软件整体框架如图 2 所示。

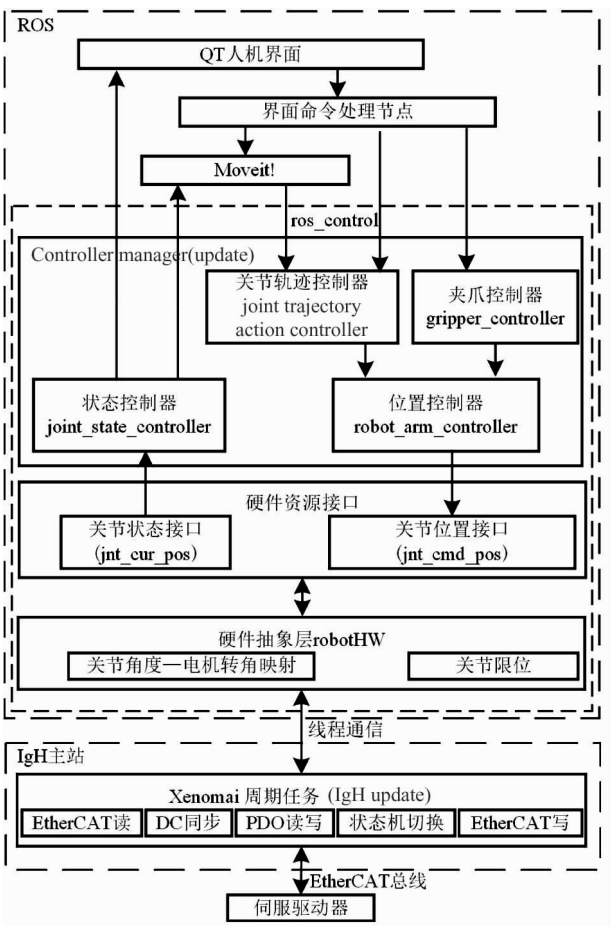


图 2 控制系统软件框架

用户通过 QT 人机界面对机器人发出控制指令或机器人编程语言指令,经过界面命令处理节点解析、处理后,笛卡尔坐标系下的控制指令由 moveit! 进行运动学规划、求解,规划好的轨迹利用 controller manager启动关节轨迹控制器(joint trajectory action controller,JTAC)进行插值计算,关节坐标系下的控制指令直接传给 JTAC 或夹爪控制器(gripper _ con-

troller)计算。控制器计算后得到每个关节的关节角度通过关节位置接口中 jnt _ cmd _ pos 变量传给硬件抽象 robotHW,接着将关节角度(jnt _ cmd _ pos)映射成电机转角,利用线程通信传给 IgH,最后通过 EtherCAT 总线将控制命令发给驱动器执行。

2 EtherCAT 通信实现

EtherCAT 数据帧是由主站发起,接着遍历各个驱动器,每当数据帧到达各个驱动器的从站控制器(EtherCAT slave control,ESC)时,ESC 都会根据数据帧内容判断是否进行读写数据操作,当最后一个驱动器的操作完成后数据帧将会返回主站,主站担负整个 EtherCAT 网络中的控制任务。IgH 提供了专用网卡的实时驱动,本文引入了开源 Xenomai 实时扩展,将 IgH 用户层中的周期性任务移植到 Xenomai 的实时微内核中,运行在微内核实时任务中的 IgH 主站执行过程如图 3 所示。

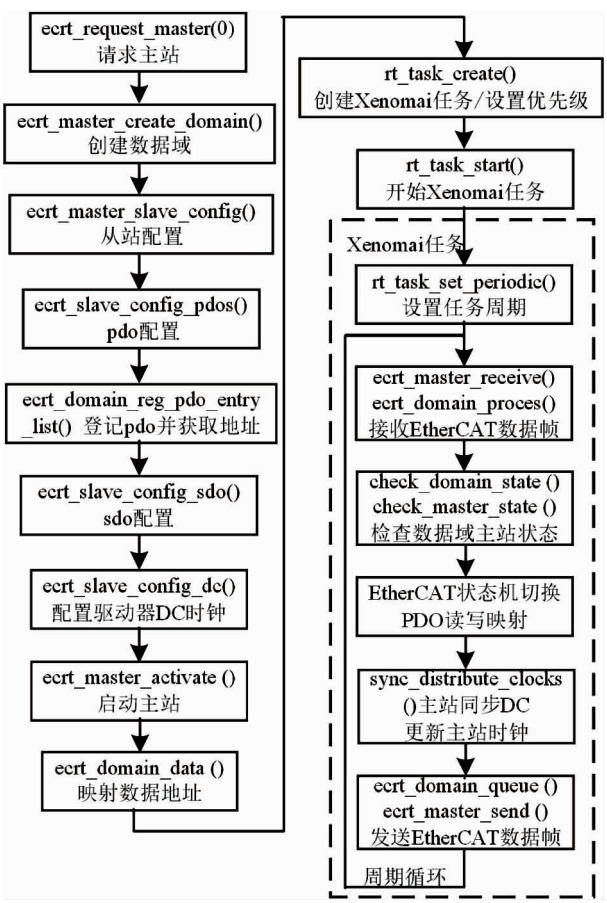


图 3 基于 Xenomai 的 EtherCAT 主站流程

EtherCAT 应用层使用 CoE 协议(CANopen over EtherCAT)实现伺服控制的周期性任务。主站通过服务数据(service data objects, SDO)配置函数 ecrt _ slave _ config _ sdo ()往对象字典 0x6060 写入 8,即将驱动器配置为周期同步位置模式(cyclic synchronous position mode, CSP)。4 个伺服驱动器实现周期性任务所需要的过程数据(process data objects, PDO)和对应的对象字典如表 1 所示,其中第 4 个驱动器中的数字输出对象字典 0x60FE. 1 中还包含机械臂末端夹爪 IO 信号。

表 1 PDO 配置

索引	对象字典	名称	类型
0x1600 RXPDO	0x6040. 00	控制字	UINT16
	0x607A. 00	目标位置	DINT32
	0x60FE. 01	数字输出	UINT32
	0x60FF. 00	目标速度	DINT32
0x1A00 TXPDO	0x6041. 00	状态字	UINT16
	0x6064. 00	实际位置	DINT32
	0x606C. 00	实际速度	DINT32
	0x60FD. 00	数字输入	UINT32

3 控制系统软件实现

控制系统软件的实现包含机器人模型的建立、ros _ control 的硬件抽象节点设计和控制器配置、moveit! 的配置、基于 QT 插件的人机交互界面设计以及界面命令处理节点的设计。通过 rqt _ graph 工具查看当前节点关系图,节点关系图直观地反映了各个节点数据间的关系,本文主要的节点关系如图 4所示。各个节点之间通过消息机制或者 action 来通讯,节点功能如表 2 所示。

3.1 机器人运动学模型建立

实现机器人控制需要建立机器人模型[14],使用统一机器人描述格式(unified robot description format,URDF)[15]来描述机器人模型,可以将机器人的各个结构抽象成一个个连杆结构相连,还包含机器人运动学动力学信息。首先在 SolidWorks 中创建机器人模型,然后将机器人各个关节参数导入 URDF 文件中,ROS 通过 TF 功能库计算出各个关节坐标系之间的转换。本文的SCARA机器人的连杆坐标

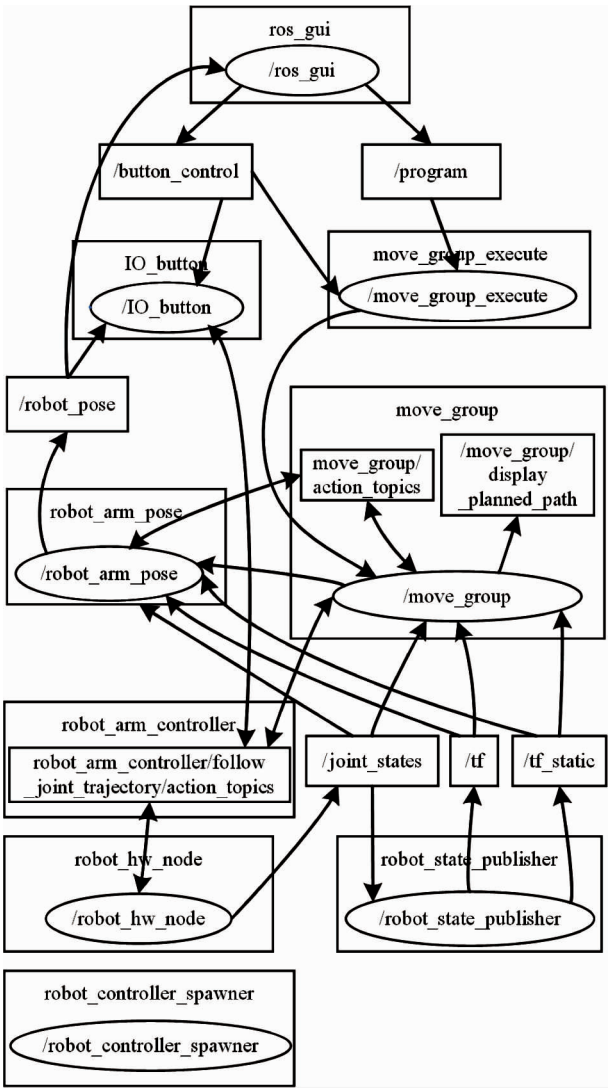


图 4　控制系统节点关系

表 2　节点功能

节点名称	功能
ros _ gui	人机界面节点,包括多个界面功能,接收 robot _ arm _ pose 通过 robot _ pose 话题发布的位姿、状态并显示
IO _ button	界面命令处理节点,接收 ros _ gui 通过 button _ control 话题发布的数据,负责处理关节坐标系下的控制命令
move _ group _ execute	界面命令处理节点,接收 ros _ gui 通过 button _ control 和 program 话题发布的数据,负责处理笛卡尔坐标系下的控制命令
robot _ arm _ controller	ros _ control 中的控制模块,接收 move _ group 和 IO _ button 节点通过 action 传输的轨迹,负责对轨迹进行插值计算

（表 2 续）

move _ group	moveit! 核心部分,负责接收 move _ group _ execute 所需要规划的轨迹点,并进行轨迹规划
robot _ arm _ pose	实时获取机器人末端位姿和系统运行状态并发布
robot _ hw _ node	ros _ control 中的机器人硬件抽象节点,接收 robot _ arm _ controller 模块插值计算后的关节角度,将经过映射后的电机转角传给 IgH

系布局如图 5 所示,创建好的机器人模型在 moveit! 可视化工具 RVIZ 中的显示效果如图 6 所示。SCARA 机器人在图 5 所示坐标系下的 D-H 参数如表 3 所示。

3.2　ros _ control 与 moveit! 实现

3.2.1　ros _ control

ros _ control 包含控制器管理器、控制器、硬件资源接口、机器人硬件抽象。robot _ hw _ node 主要

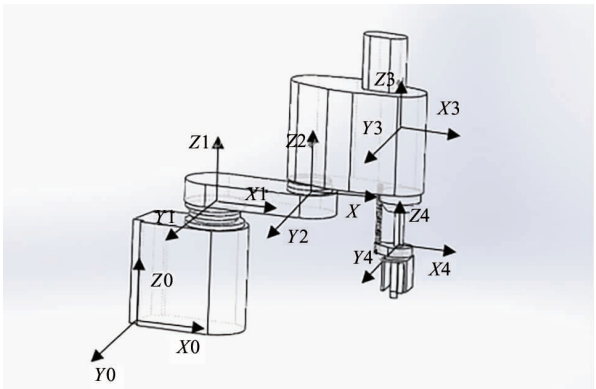


图 5　SCARA 机器人的连杆坐标系布局

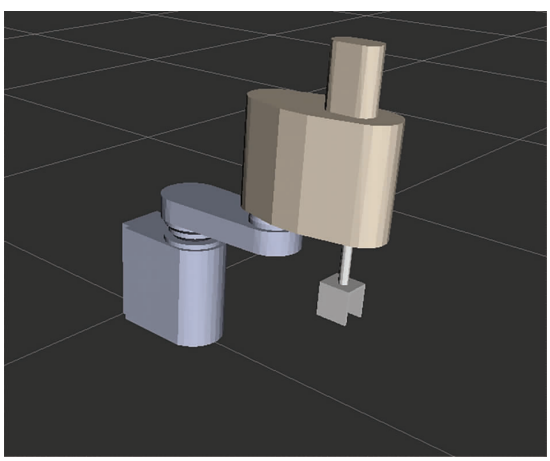


图 6　SCARA 机器人显示效果图

表 3　SCARA 机器人连杆参数

连杆 i	a_i (mm)	α_i (deg)	d_i (mm)	θ_i (deg)
1	160	0	210	$\theta_1(0)$
2	250	0	70	$\theta_2(0)$
3	150	0	120	0
4	0	0	d	$\theta_4(0)$

负责初始化硬件资源接口和 IgH、关节角度与电机转角映射以及 ROS 与 IgH 数据交互,程序流程图如图 7 所示。其中与 IgH 采用多线程共享全局变量实现数据交互,通过定义中间全局变量,当 ROS 下发的目标电机转角数据更新时,将新数据写入中间全局变量,接着等待 IgH 读取,同理,IgH 上传的实际电机转角写入中间全局变量并等待 ROS 端读取。

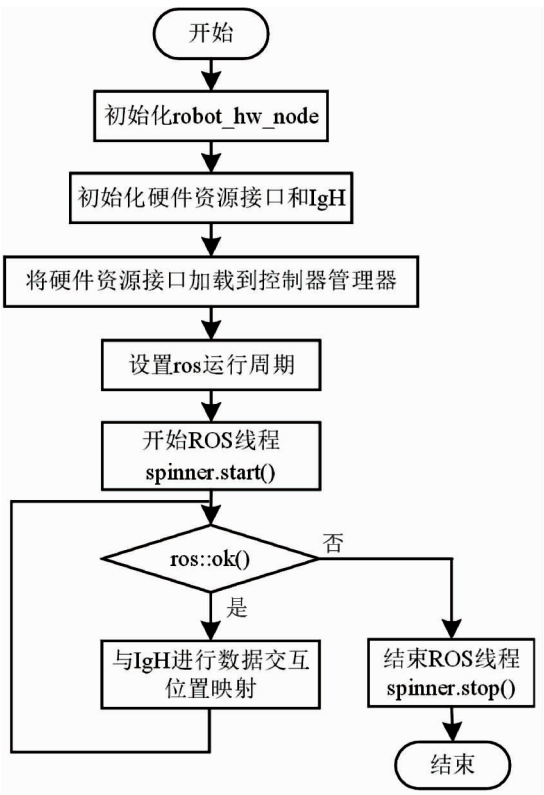


图 7　robot _ hw _ node 节点流程图

robot _ arm _ controller 为 ros _ control 的控制器模块,控制器管理器根据 action 接口类型启动对应的控制器,控制器在控制器管理器中根据运行状态分为初始化、开始、更新、结束 4 个过程。本文通过 ArbotiX 功能包中的 JTAC 和夹爪控制器实现插值计算和夹爪功能,其中 JTAC 使用 5 次样条插值方案[16],控制器计算出来的关节角度通过硬件资源接口传给 robot _ hw _ node 节点。

3.2.2　moveit!

move _ group 为 moveit! 的核心部分,集成各个功能包和插件。moveit! 使用 Moveit Setup Assistant 工具配置,其中运动规划使用开源运动规划库(open motion planning library, OMPL)中的算法,以及利用 OROCOS 提供的 KDL 插件进行机器人正逆运动学求解。还需通过 moveit _ simple _ controller _ manager 插件配置控制器,将控制器与 moveit! API 之间的通信抽象化,本文的配置为:

controller _ list:

- name: robot _ arm _ controller　##控制器名字

action _ ns: follow _ joint _ trajectory　##控制器发布 action 消息的命名空间

default: True　##是否是该规划组的默认控制器插件

type: FollowJointTrajectory　##action 的类型

joints:　##该规划组所包含的关节

- joint _ 1##关节 1
- joint _ 2##关节 2
- joint _ 3##关节 3
- joint _ 4##关节 4

move _ group 通过调用插件接口 sendTrajectory()将规划好的轨迹发送给该插件,接着插件读取 controller 的配置,根据 action 的类型创建 action client,利用 client- > sendGoal()将轨迹发给 JTAC 的 server 处理。

3.3　人机交互界面设计

人机交互界面的设计基于 QT 插件 ros _ qtc _ plugin 来实现,该插件可以在 QT 中导入、创建、编译、调试 ROS 项目,ros _ gui 节点启动时会同时启动一个 QT 界面窗口。整体结构如图 8 所示,后续可以对界面功能进行扩展,实现更丰富的功能。

人机交互界面通过两个窗口来实现,在登录窗口输入正确的用户名和密码即可进入操作窗口,操作窗口使用 Tab Widget 控件来实现多个操作页面,Dock Widget 控件实现 ROS 连接界面与状态显示。整个人机交互界面主要完成了以下几种设计,包括

人机界面通信协议的设计、数据库界面的设计、编程界面的设计。

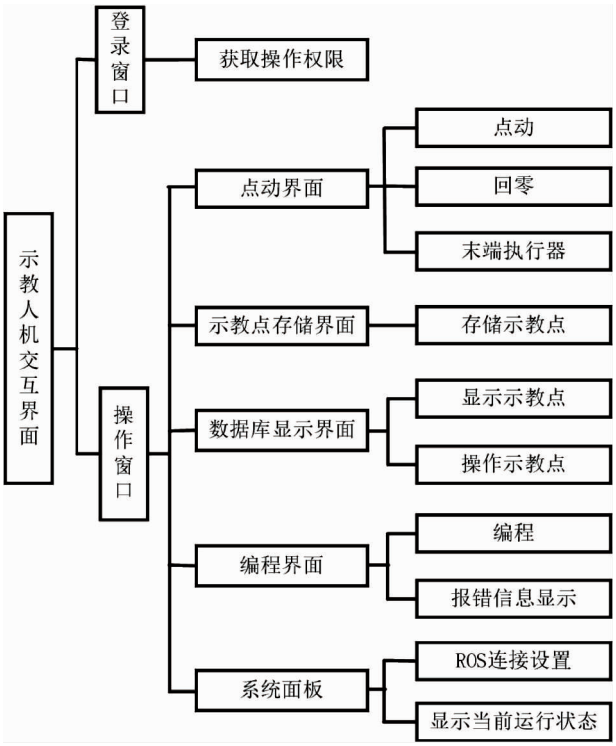


图 8　人机交互界面结构图

(1)人机界面通信协议

本文基于 ROS 的消息机制并根据界面功能设计了一种人机界面通信协议,后续可以根据功能需求对其进行扩展,消息结构如表 4 所示,其中√为对应话题所使用的数据消息类型,×表示没有使用该数据类型。

表 4　消息定义

	button _ control	program	robot _ pose
功能码 INT32	√	×	×
数据量 INT32	√	√	√
数据 Float64	√	√	√
指令码 INT32	×	√	×
行号 INT32	×	√	×
速度 INT32	×	√	√
状态 INT32	×	×	√

当 ros _ gui 节点收到其他节点发来的实时状态、位姿时,将在界面显示相应内容,命令处理节点接收到 ros _ gui 节点的控制命令后,根据消息内的功能码或指令码进行解析并运行,功能码和指令码定义如表 5 所示。

表 5　功能码/指令码定义

功能/指令	命令	执行节点
0x01 功能	关节 X +	IO _ button
……	……	……
0x09 功能	回零	IO _ button
0x0A 功能	夹爪开关	IO _ button
0x0B 功能	V +	IO _ button
0x0C 功能	V-	IO _ button
0x10 功能	笛卡尔 X +	move _ group _ execute
0x11 功能	笛卡尔 X-	move _ group _ execute
……	……	……
0x24 指令	movel	move _ group _ execute
0x25 指令	close	move _ group _ execute
0x26 指令	执行	move _ group _ execute

(2)数据库界面

数据库界面通过 QT 默认集成的 SQLite 数据库实现,利用 QT 封装好的 API 对数据库中的数据进行操作和管理。通过 QSqlTableModel 类设置模型、关联创建好的各坐标系数据表,以及使用 TableView 控件关联模型并显示数据库数据,最后利用按钮槽函数来响应数据库按钮信号,设计流程如图 9 所示。

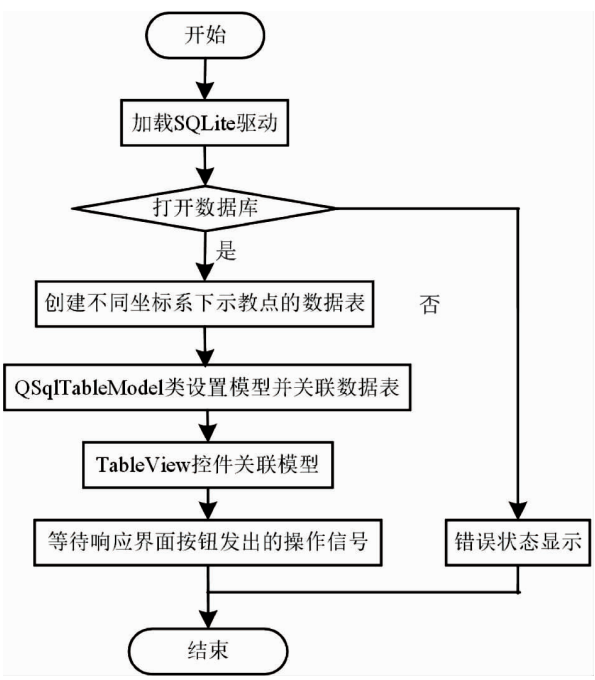


图 9　数据库流程图

数据库界面包含存储、导出、排序、添加、删除、刷新等功能。存储:负责将选中的坐标系下当前的坐标存入数据库中;导出:把数据库中需要使用的示教点导出,用于编程时匹配以及数据提取;排序:根据表头给示教点排序;添加:往数据库添加一个自定义示教点;删除:删除当前选中的示教点;刷新:刷新数据库。

(3)编程界面

编程界面的机器人语言解释器是基于 QT 正则表达式,将机器人语言解释成界面命令处理节点能够识别的指令系统。解释器设计流程如图 10 所示,包含以下 2 个步骤。

第 1 步 利用 QT 正则表达式设计机器人语言的语句表,可直接添加修改语句方便扩展,本文实现 SCARA 机器人编程控制所用的语句表如下。

```
QString Movel("^(movel)(\\s*)([A-Z]+[0-9]+)(\\s*)(\\d+)(;)(\\s*)$"); //movel 笛卡尔坐标系走直线
QString Movej("^(movej)(\\s*)([A-Z]+[0-9]+)(\\s*)(\\d+)(;)(\\s*)$"); //movej 关节坐标系走直线
QString Open("^(open)(;)(\\s*)$"); //夹爪关闭
QString Close("^(close)(;)(\\s*)$"); //夹爪打开
QString Delay("^(delay)([/(])(\\d+)([/)])(;)(\\s*)$"); //延时
QString P("^(" + P + ")([/(])(\\s*)(([-?])?)(\\d+)(((/.])(\\d+))?)(\\s*[,]\\s*)(([-?])?)(\\d+)(((/.])(\\d+))?)(\\s*[,]\\s*)(([-?])?)(\\d+)(((/.])(\\d+))?)(\\s*[,]\\s*)(([-?])?)(\\d+)(((/.])(\\d+))?)(\\s*[/)]\\s*)(;)(\\s*)$"); //示教点匹配。
```

第 2 步 利用 exactMatch()函数根据上文建立的语句表进行首次匹配,匹配通过后记录指令码、行号、延时时间等数据,如果是 movej 或 movel 指令则进行第 2 次匹配。第 2 次匹配将根据最后一句语句表对示教点匹配,匹配成功之后将通过 cap()函数对示教点各个坐标进行捕获,记录需要运行的示教点坐标以及速度,若匹配失败将根据错误原因显示报错信息。最后将数据赋值给 move _ group _ execute. msg 中的变量并通过 program 话题发布,由 move _ group _ execute 节点接收解析。

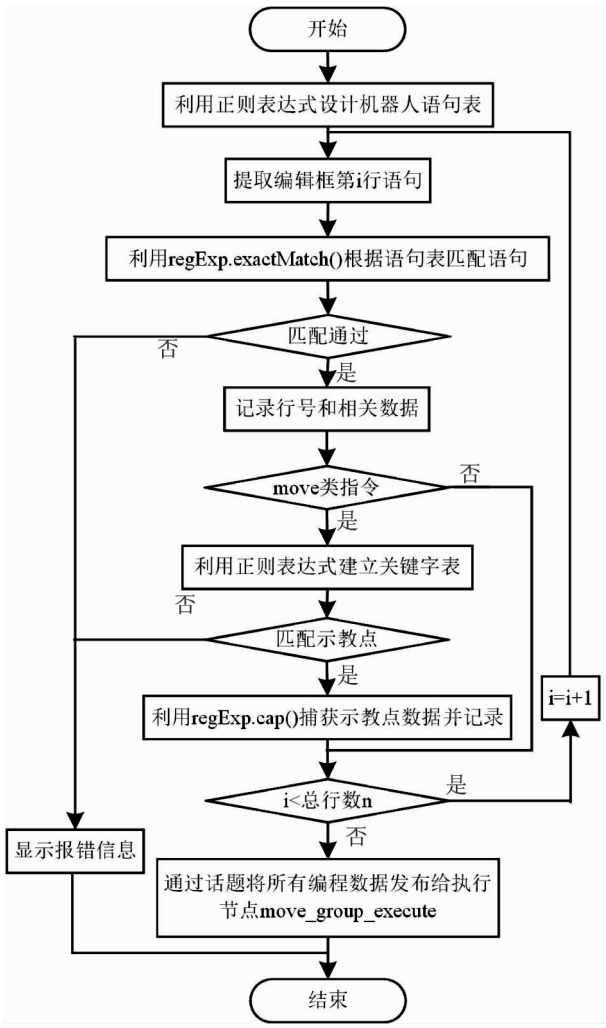


图 10　解释器流程图

3.4　命令处理节点设计

3.4.1　move _ group _ execute 节点

move _ group _ execute 用于处理人机界面节点下发的笛卡尔坐标系下点动、回零和直线命令,在初始化一些接口后,首先订阅 ros _ gui 发布的话题,获取机械臂当前位姿和控制命令,接着根据功能码或者指令码判断具体的控制命令,执行相应的函数,最后把期望的目标点添加到 waypoints 中,通过调用 API 函数 compute _ cartesian _ path()尝试规划一条包含目标路点的轨迹,规划成功后将通过 execute()

控制机械臂运动,否则打印错误信息,设计流程图如图 11 所示。

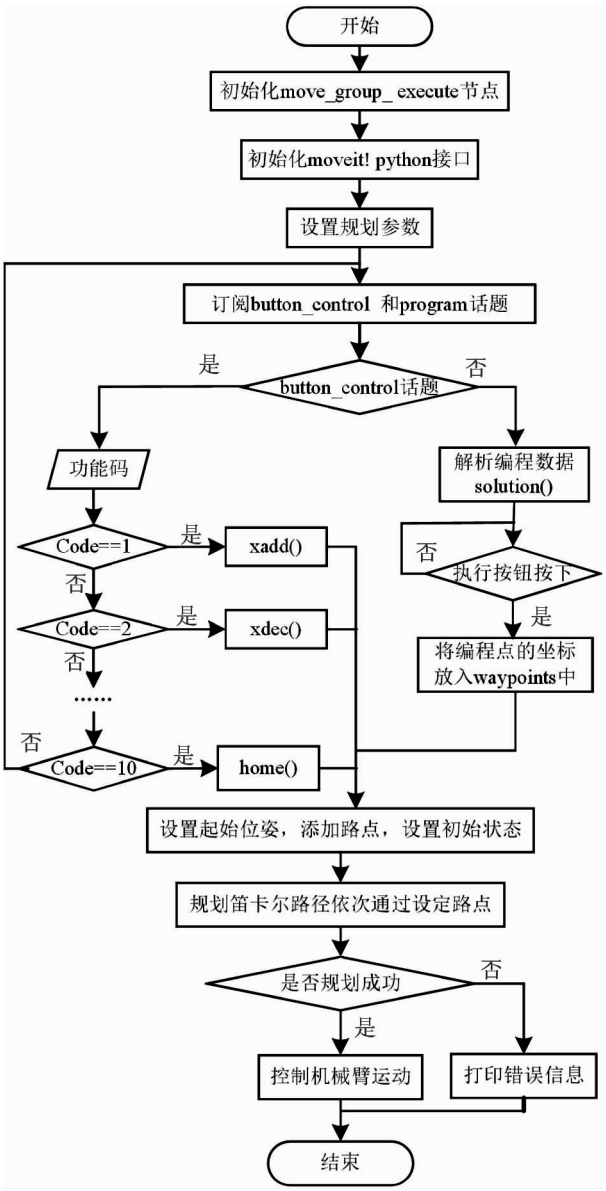


图 11　move _ group _ execute 节点流程图

3.4.2　IO _ button 节点

IO _ button 负责处理关节坐标系下的控制命令,通过 action 直接与 controller 连接,使用 SimpleAction Client 创建 2 个客户端,轨迹命令的客户端 action 消息名为 robot _ arm _ controller/follow _ joint _ trajectory,类型为 FollowJointTrajectoryAction;夹爪命令的客户端 action 消息名为 robot _ gripper _ controller/gripper _ action,类型为 GripperCommand。接着等待与服务端连接,连接成功后就可以将目标位置或者夹爪开关信号发送到服务端进行处理,设计流程图如图 12 所示。

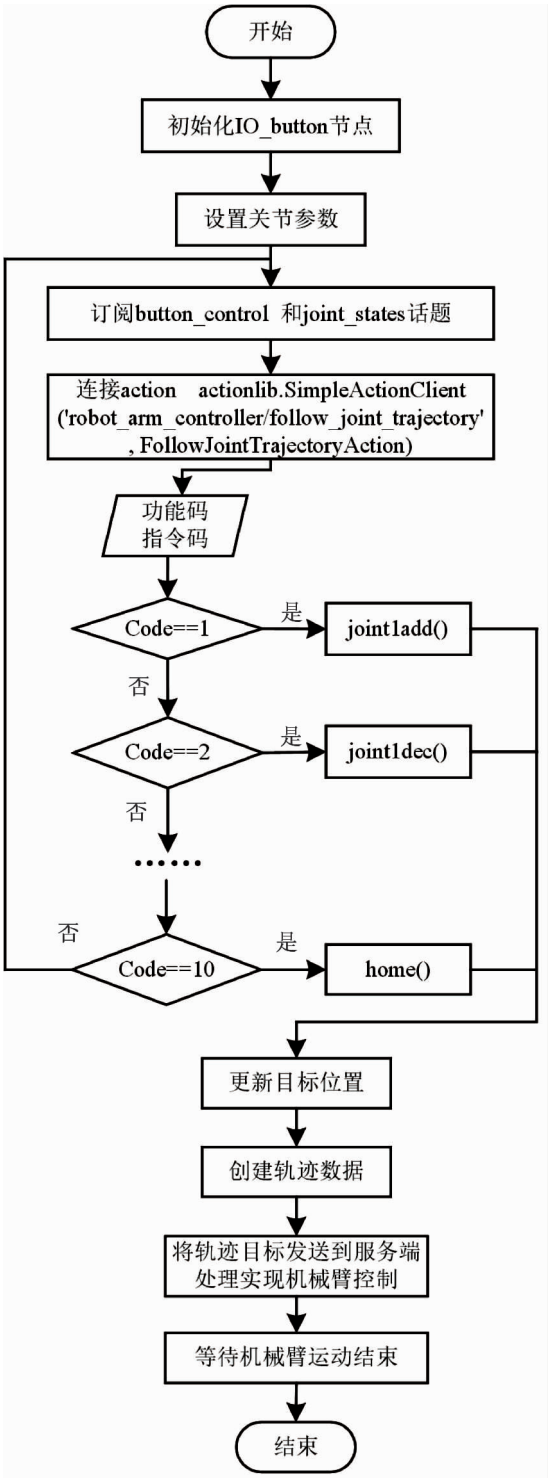


图 12　IO _ button 节点流程图

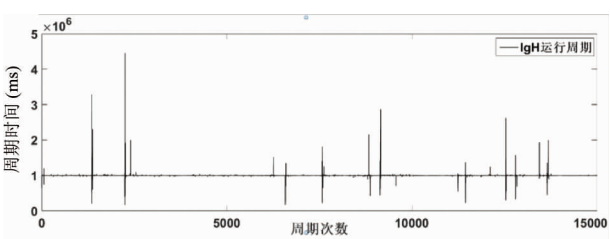
4　实　验

EtherCAT 主站和 ROS 平台安装在一台工控机

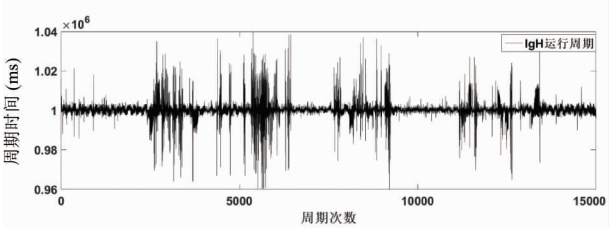
上,结合配备实验室设计的伺服驱动器的 SCARA 机器人,对开发的机器人控制系统进行测试。

(1) 通信系统是控制系统核心结构之一,保证了控制系统的可靠性、实时性和同步性。通过测量每个 IgH 运行周期时间,运行在 Linux 内核内和 Xenomai 实时微内核的 IgH 周期时间如图 13 所示,结果表明运行在 Linux 内核内的周期在 0.2 ~ 4.5 ms 之间波动,运行在 Xenomai 实时微内核里的周期在 1 ± 0.04 ms 内波动,周期抖动过大将会导致 EtherCAT 通信报错。多轴同步通过每个驱动器响应 EtherCAT 同步中断信号来实现,利用 ROHDE&SCHWARZ 示波器测量伺服驱动器之间的同步中断信号延时,共测量了 100 000 组数据并画出分布图,如图 14 所示,其中横坐标表示抖动时间,纵坐标表示样本次数,结果表明同步时钟抖动峰值在 25 ns 以内,均值为 - 67.173 ps,并且为正态分布,达到了工业控制的基本网络抖动精度要求。

(2) 人机交互界面与 SCARA 机器人机械本体的联机调试,通过点动界面控制机器人移动到指定位置,并将示教点的位置信息保存到数据库中,接着使用编程功能编写机器人程序以实现对机器人的控制。数据库界面如图 15 所示,编程界面如图 16所示,SCARA 机器人根据编写的程序执行过程如图 17 所示,将金属块从右边搬运至左边。



(a) 运行在Linux内核



(b) 运行在Xenomai实时微内核

图 13　周期时间分布图

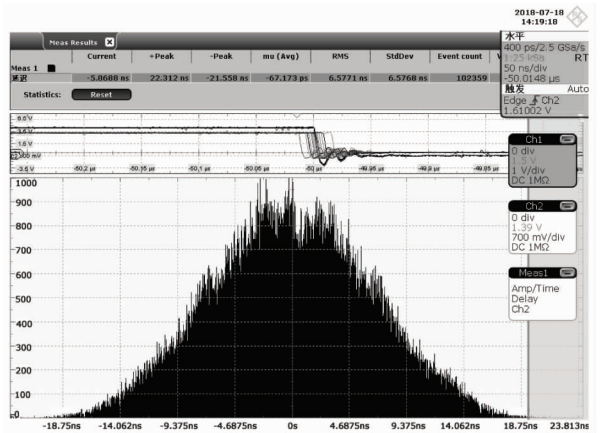


图 14　同步信号抖动时间

图 15　数据库界面

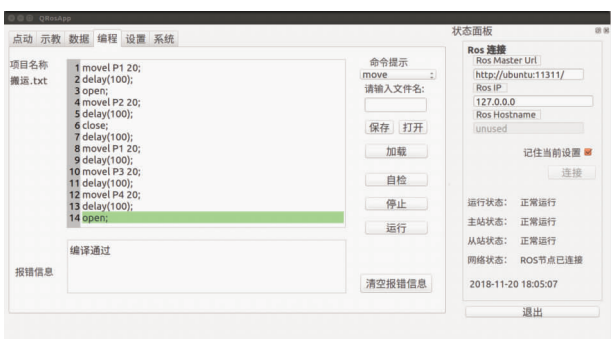
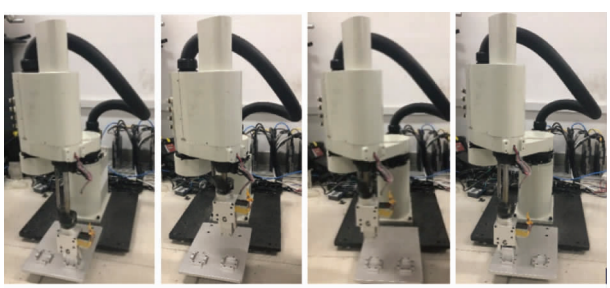


图 16　编程界面



(a) 位姿1　(b) 位姿2　(c) 位姿3　(d) 位姿4

图 17　机器人运行状态

5 结 论

本文研究并设计了基于 EtherCAT 和 ROS 的 SCARA 机器人控制系统,采用 EtherCAT 总线技术、Xenomai 实时扩展、ros _ control 插值计算、moveit! 运动规划和求解、ROS 通讯机制、QT 开发框架等实现了 ROS 平台下对 SCARA 机器人的控制、示教。本控制系统所使用的平台、框架皆为开源的,具有低成本、易于扩展、操作方便等优点,满足工业控制的基本需求,对实现不同类型的工业机器人控制的开发具有较大的参考价值。

参考文献

[1] 孟明辉,周传德,陈礼彬,等. 工业机器人的研发及应用综述[J]. 上海交通大学学报,2016,50(1):98-101

[2] Martínez-Prado M, Rodríguez J, Gómez-Loenzo R, e t al. An FPGA-based open architecture industrial robot controller[J]. *IEEE Access*, 2018, 6: 13407-13417

[3] 李辰. SCARA 型多轴工业机器人控制技术研究[D]. 济南:山东大学控制科学与工程学院,2017. 1-8

[4] 杨帆. 基于 CPAC 的 SCARA 机器人控制系统的研究[D]. 武汉:湖北工业大学机械电子工程学院,2016. 1-5

[5] 毕鲁雁,刘立生. 基于 RTX 的工业机器人控制系统设计与实现[J]. 组合机床与自动化加工技术,2013(3):87-89

[6] Cereia M, Bertolotti I C, Scanzio S. Performance of a real-time EtherCAT master under Linux[J]. *IEEE Transactions on Industrial Informatics*, 2011, 7(4): 679-687

[7] Delgado R, Choi B W. On the in-controller performance of an open source EtherCAT master using open platforms [C]. In: International Conference on Ubiquitous Robots and Ambient Intelligence, Jeju, Korea, 2017. 744-748

[8] 张群. 一种实时以太网 EtherCAT 介绍[J]. 电子世界,2012(22):15-16

[9] 高恩博. 基于 Linux 嵌入式平台的 EtherCAT 主站系统研究与设计[D]. 杭州:浙江大学电气工程学院,2017. 8-11

[10] 胡春旭. ROS 机器人开发实践[M]. 北京:机械工业出版社,2018. 1-4

[11] ROS(Robot Operating System)[EB/OL]. http::://www. ros. org:ROS, 2008

[12] Hoske, Mark T. ROS industrial aims to open, unify advanced robotic programming[J]. *Control Engineering*, 2013,60(2):20-21

[13] Armstrong L. ros _ qtc _ plugin[EB/OL]. https://github. com/ros-industrial/ros _ qtc _ plugin: GitHub, 2015

[14] 熊有伦. 机器人技术基础[M]. 武汉:华中理工大学出版社,1996. 33-51

[15] 曹正万,平雪良,陈盛龙,等. 基于 ROS 的机器人模型构建方法研究[J]. 组合机床与自动化加工技术,2015(8):51-54

[16] 曹狄, 陈玮,林浩志,等. 基于 ROS 的羽毛球机器人上体控制系统设计[J]. 机电一体化,2018,24(2):54-59

A SCARA robot control system based on ROS and IgH EtherCAT master

Xu Jianming, Wu Shuwei, Wu Xiaowen, Zhang Wenan, Yu Li

(College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023)

Abstract

According to requirments of developing a networked selection of compliant assembly robotic arm (SCARA) controller based on open source softwares, a SCARA robot control system based on robot operating system (ROS) and Ethernet control automation technology (EtherCAT) is designed under Linux system. EtherCAT communication is implemented by developing the IgH user layer program and migrating to Xenomai's real-time microkernel based on the IgH EtherCAT master. Under the ROS platform, the motion planning and solving is achieved by building an unified robot description format (URDF) model and combining with moveit! A hardware abstraction node is developed according to the configuration of ros _ control, then the human-computer interaction interface node is designed and developed by using the QT plug-in, mainly including human-machine interface communication protocol, database interface, programming interface and other modules. The corresponding command processing nodes are developed according to the control commands in different coordinate systems. Finally, through measuring the real-time task cycle and synchronous signal jitter time, as well as robot control experiments, the results verify the practicability of the designed SCARA robot control system.

Key words: robot control system (ROS), Ethernet control automation technology (EtherCAT), IgH, Linux