doi:10.3772/j.issn.1002-0470.2020.09.008

基于云平台的机器人监控系统设计①

徐建明② 俞俊铭 董建伟 俞 立

(浙江工业大学信息工程学院 杭州 310023)

摘 要 本文设计了一种基于阿里云平台的SCARA机器人监控系统,包括本地客户端与Web远程监控端。本地客户端基于TCP/IP协议和C#语言进行搭建,包括基于OPC统一架构(OPC UA)的数据采集模块,基于MySQL的数据存储模块和基于消息队列遥测传输(MQTT)与云平台的数据交互模块。Web远程监控端通过前后端分离的方式进行搭建,基于Django rest framework框架搭建Web后端服务程序,可视化界面由Vue前端框架搭建,在此基础上,采用WebGL和Three.js技术在界面上搭建SCARA机器人3维模型。最后基于OPC UA协议实现了本地客户端与SCARA机器人的运动状态、数据指令的交互,通过MQTT与位于阿里云的MySQL数据库进行数据交互,通过3D可视化界面实现对SCARA机器人监控。

关键词 阿里云;OPC统一架构(OPC UA);消息队列遥测传输(MQTT);Django框架;Web远程监控

0 引 言

随着德国“工业4.0”的推出,工业生产进入到使用信息化技术促进产业变革的时代,同时转向云计算是互联网发展面临的一个重大改变,而云平台则是该改变中重要的一个环节,因此研究基于云平台的机器人监控系统显得越来越重要。

目前机器人监控系统多数采用客户端/服务端的模式进行搭建,如张爱民等人[1]基于TCP/IP协议设计工业机器人远程监控与诊断系统。这种模式下,远程端需要安装客户端,系统扩展性降低;另一种模式是通过访问浏览器获取数据并查看设备运行状态。Sallinen等人[2]提出基于Web用户界面的工业机器人远程监控与维护的框架。骆晓娟等人[3]设计基于AJAX和浏览器/服务器(browser/server,B/S)构架的实时监测系统。徐建明等人[4]设计基于Web的工业机器人3D虚拟动态监控系统。同时随着云平台和云计算的发展,基于云平台的机器人监控系统成为物联网未来的发展方向之一。Gubbi等人[5]研究物联网未来的发展方向。Ji等人[6]提出基于云的物联网城市停车系统。Douzis[7]提出基于云的模块化和通用物联网管理系统。Baker等人[8]研究基于云的物联网系统的组合算法。Dinh等人[9]提出云与面向移动云计算应用的物联网基于位置的交互模型。同时,消息队列遥测传输(message queuing telemetry transport, MQTT)作为即时消息协议逐渐被广泛用于物联网。Schmitt等人[10]提出通过MQTT协议作为物联网数据交换的桥梁。在此基础上,机器人3维展示被逐渐运用于监控系统中。Mostefa等人[11]设计一种基于虚拟现实的移动机器人远程操作系统。杨硕等人[12]设计基于虚拟现实的一对多远程康复训练机器人监控系统。

随着数据量不断扩大,基于本地服务器的监控系统对硬件资源的需求会不断增大,本文以客户端/服务端(client/server,C/S)模式与B/S模式相结合

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项目(U1709213)和国家自然科学基金面上项目(61374103)资助。

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:2019-09-23)

的方式设计一种基于阿里云的机器人监控系统,相较于上述传统方式,云平台具有更好的稳定性、容量扩展性,可按照需求为用户定制服务资源。本系统由本地客户端和 Web 远程监控端组成,其中本地客户端基于 TCP/IP 协议和 OPC 统一架构(OPC unified architecture, OPC UA)协议与机器人控制器进行数据通讯,通过搭建 MySQL 客户端存储数据,在此基础上,基于 MQTT 协议与云端进行数据交互。Web 远程监控端采用前后端分离的方式搭建,相较于传统机器人监控系统的开发方式,前后端分离的方式具有开发周期短暂的优势,同时更容易发现系统运行时出现的错误。后端基于 Django rest framework 框架搭建,前端采用 Vue 框架搭建,大屏展示界面基于阿里云的 DataV 模块搭建,3 维画面展示界面基于 WebGL[13] 和 Three. js 技术搭建[14]。

1 系统整体框架

本系统以基于 CoDeSys 开发的 SCARA 机器人控制系统为研究对象进行搭建,包括本地客户端和 WEB 远程监控端,系统框架图如图 1 所示。

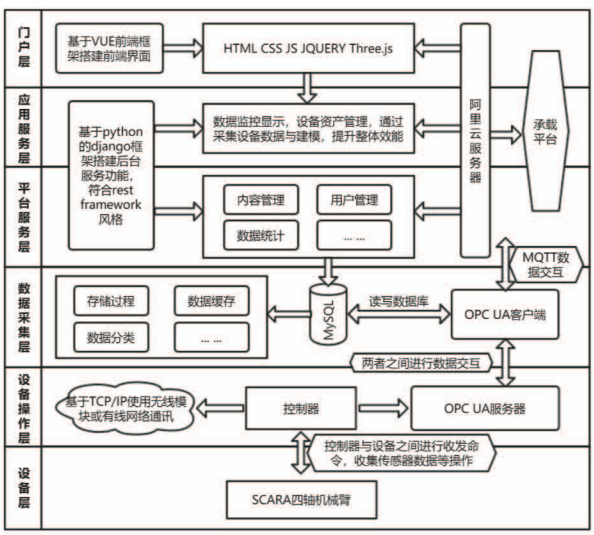

图 1 系统框架图

图 1 中的设备层与设备操作层通过机器人控制器进行衔接,机器人控制器的运动控制程序以及 OPC UA 服务器由 Codesys PLC 软件编程工具进行编写,通过工业以太网(EtherCAT)解决采集机器人相关物理数据[15],通过搭建 OPC UA 客户端来解决 SCARA 机器人的数据交互。

本地客户端基于 TCP/IP 协议搭建,通过 OPC UA 协议解决与控制器的数据交互问题,基于 MySQL 的嵌入式应用程序解决数据存储问题,其中数据库作为监控系统前后端的信息载体,存储 SCARA 机器人运动过程中产生的各种信息,同时基于 MQTT 协议解决与云端的数据交互问题,MQTT 协议具有功耗低、轻量级及易于实现的优点,是物联网的重要组成部分。

WEB 远程监控端由图 1 中的平台服务层,应用服务层和门户层组成,采用 nginx[16] 与 uwsgi 技术将远程监控程序部署于阿里云服务器,程序部署完成后,用户可以通过浏览器随时随地查看机器人历史数据并进行实时监控。

2 本地客户端设计

本地监控端采用 C/S(客户端/服务器端)模式进行搭建,主要由 3 部分组成,即基于 OPC UA Client[17] 的数据采集模块、基于 MySQL 的数据存储模块和基于 MQTT 与云平台的数据交互模块。

2.1 基于 OPC UA Client 的数据采集模块

传统的工业自动化解决方案由于在设备间通信上采用不兼容和不可互操作的差别化标准,因而存数据交互的困难。OPC UA 架构采用客户端/服务器模式和发布者/订阅者模式为数据交互提供框架,在地址空间定义节点类并实例化,通过层次结构进行访问,以简化客户端访问。C#语言可以实现 OPC UA 的自动化接口,提供自动配置、过程控制和数据存取的接口,在窗口界面中搭建用户输入和响应事件模块,在工作线程中通过订阅方式对数据进行采集。在功能上实现浏览 OPC UA 服务器、修改数据、订阅数据,具体实现方式如下。

(1)浏览 OPC UA 服务器。在窗口界面拖入 panel 容器控件,在主程序中引用“Opc. Ua. Client”库,在工作线程中使用 FormBrowseServer form = new FormBrowseServer( )函数,该函数通过结构层次的方式访问服务器,查看服务器的节点状态,从而获

取服务器的内容，使用 panel. Controls. Add(form) 函数，该函数是 C#语言的内置函数，用于将服务器的内容显示在 panel 容器中，供用户浏览和使用。

(2)修改数据。在工作线程中使用 private OpcUaClient opcUaClient = new OpcUaClient( ) 函数，该函数为 opcua. dll 函数库的内置函数，用于将 OPC UA 客户端实例化，工作线程根据该实例对象进行节点操作、节点查阅、节点订阅以及其他操作，线程中使用 opcUaClient. writeNode (this. NodeId. Text, this. writeNodeId. Text) 函数，该函数通过结构层次查找节点，依赖 TCP/IP 协议修改节点数据，从而实现修改机械臂运动参数等数据，并控制机械臂。

(3)订阅数据。在工作线程中编写 SubCallback( ) 函数，该函数通过调用者(Caller) 向回调函数(Callee) 发出调用，被调用函数启动后，不需要被调函数执行完毕，程序执行流立即返回到调用者继续执行，从而实现节点订阅，并对订阅的节点绑定 listView1 _ DragDrop( ) 函数，该函数根据 C#控件内容的可拖动原理，实现对数据节点名称的拖动，编写 AddSubscription( ) 函数，用于添加节点，并将每个节点绑定上文中的 SubCallback( ) 函数，实现对批量节点的订阅。工作程序流程图如图 2 所示。

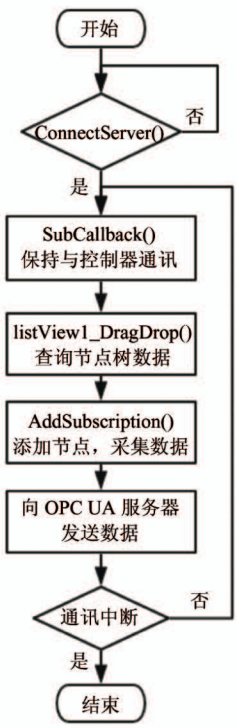

图 2　数据订阅程序流程图

2.2　基于 MySQL 的数据存储模块

MySQL[18] 是一个关系型数据库管理系统，关系数据库将数据保存在不同的表中以增加速度并提高灵活性，并为 C#编程语言提供了 API 接口，同时能够作为一个库嵌入到其他软件中，使嵌入客户端的方式实现数据的存储得以实现，不仅可以提高数据的实时性，也能降低本地客户端的内存空间。

本数据存储模块通过 C#语言进行搭建，并嵌入到本地客户端，数据库表格采用树状关系图的方法进行设计，使用多表关联方式建立机器人对象表、设备表、采集点数据表，该设计方法优势在于扩展性高，无需频繁修改数据库。在功能上实现数据解析，数据筛选和数据存储，如图 3 所示，具体实现方式如下。

(1)数据解析。将从控制器接收到的数据按照通讯协议解析成当前系统实际的物理值，在工作线程中使用 value. WrappedValue. TypeInfo. BuiltInType 函数，该函数有 2 个属性，数组和基础类型，通过返回值判断数据类型、数据格式和数据量，从而得到该数据包含的物理量和数值单位。

(2)数据筛选。在窗口界面中拖入 comboBox 控件和 ListBox 列表框控件，用于存放采集点信息，在工作线程中使用 comboBox. Items. Add(sArray[s-1]) 函数，该函数用于在 comboBox 控件中添加新的采集点，同时使用 infos. Add(new database( ) { data = "" }) 函数，该函数通过创建新的数据对象，将采集的数据筛选后添加至列表框控件中，便于用户对数据的管理和分类。

(3)数据存储。在本地客户端主程序中引用“MySQL. Data. DLL”库，在工作线程中使用 MySqlConnection conn = new MySqlConnection( ) 函数，该函数用于创建 MySQL Client 实例对象，对实例对象调用 conn. Open ( ) 函数创建 MySQL 通道，编写 MySqlCommand cmd1 = new MySqlCommand(" select * from users _ userprofile " , conn) 函数，该函数根据上文创建的实例对象(conn) 连接 MySQL 数据库，编写 MySqlCommand cmd1 = new MySqlCommand("select * from device _ device order by id DESC limit 1" , conn) 函数，该函数用于将数据存储至

云端 MySQL 数据库,同时添加存储时间,用户信息等,函数内部参数包括实例对象,采集点对象和采集点数据。

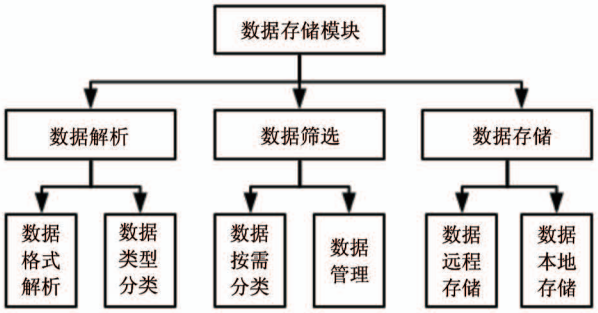

图 3　数据存储模块功能图

2.3　基于 MQTT 与云平台的数据交互模块

目前基于 Http 协议的物联网系统开发的方式[19]被频繁采用,但在嵌入式系统中或网络带宽昂贵的环境下,Http 协议并不适用,MQTT 做为一种低开销、低带宽占用的即时通讯协议,MQTT 在工业物联网、小型设备等方面有广泛的应用。因此本客户端选择采用 MQTT 协议搭建数据交互模块,具体实现方式如下。

(1)在云端搭建 MQTT 服务器,用于主题消息接收、发布。使用"apollo create brokerServer"创建服务器实例,使用"apollo-broker. cmd run"启动服务器,并且 Apollo 提供后台管理页面,方便管理和调试。

(2)建立 MQTT 服务,在客户端程序中添加引用"MQTTnet",用于激活 MQTTnet 功能。

(3)在工作线程中使用 private MqttClient mqttClient 函数,该函数用于创建 MQTT 实例对象,连接 MySQL 数据库。

(4)利用 C#自带控件库在窗口界面中拖入 textBox 等控件,用于在客户端中搭建 MQTT 用户操作界面。

MQTT 模块的后台工作线程为编写 Task. Run(async() = > {awaitConnectMqttServerAsync();})函数,该函数用于连接位于云端的 MQTT 服务器;通过编写 MqttApplicationMessage()函数,该函数用于将采集到的数据实时地传输到位于云端的 MQTT 服务器;同时通过 mqttClient. PublishAsync()函数,该函数用于订阅所需要的数据,程序流程如图 4所示。

整个工作流程如图 5 所示,当客户端连接到机器人控制器后,通过 OPC 客户端将采集到的数据以 MQTT 的形式实时地传输到云端,位于云端的 MQTT 服务器接收到数据后,做出处理,发布该主题,同时位于 WEB 端的 MQTT 客户端订阅该主题,获取数据。

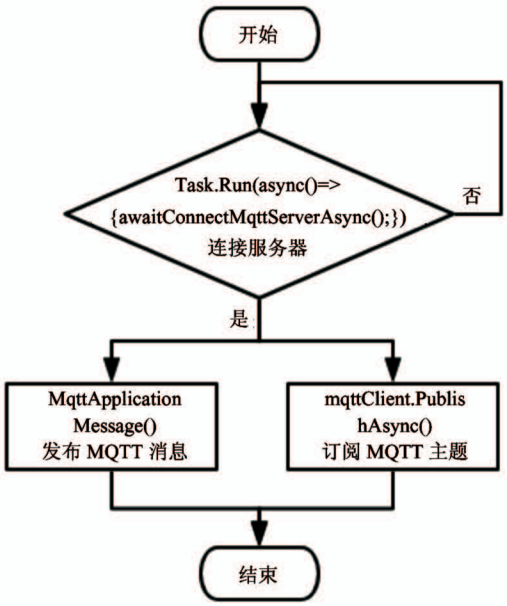

图 4　MQTT 程序流程图

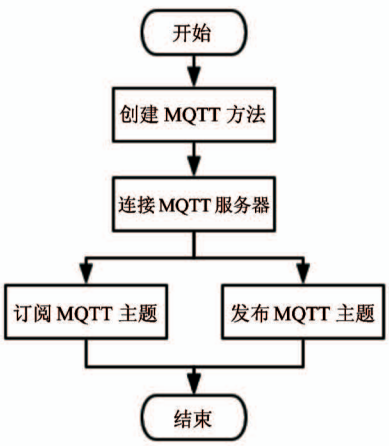

图 5　MQTT 工作流程图

3　Web 远程监控端设计

Web 远程监控端采用 B/S(浏览器、服务器)架构的方式实现,使用阿里云作为 Web 服务器。传统的 Web 框架过于复杂,难以满足快速开发的要求,

而 Django 框架以其便捷、快速、高效的特点而深受好评。综上所述，后端基于 Django rest framework 框架[20]进行搭建，以 JSON[21]的格式提供数据接口。前端基于 Vue 框架进行搭建，Vue 是一套用于构建用户界面的渐进式 JavaScript 框架[22]。在此基础上，Web 远程监控端针对 SCARA 机器人，搭建 3 维动画界面和绘制曲线图。Web 系统分为 4 个层面：

（1）门户层。即网页浏览，通过统一认证授权提供业务服务统一注册、统一登录入口。

（2）API 接口输出层。该层主要包括设备资产管理，通过设备数据采集与建模，以 API 接口的形式提供给前端使用。

（3）业务逻辑层。该层主要为各业务应用系统的构建和运行提供技术支撑，并为各应用服务提供计算、数据的调度及数据管理服务。

（4）数据库操作层。该层主要是获取数据库中的数据。通过 Django 框架中的 model 操作与 MySQL 数据库进行交互，使用 ModelViewset 类对数据库进行增删改查操作。Web 系统框架如图 6 所示。

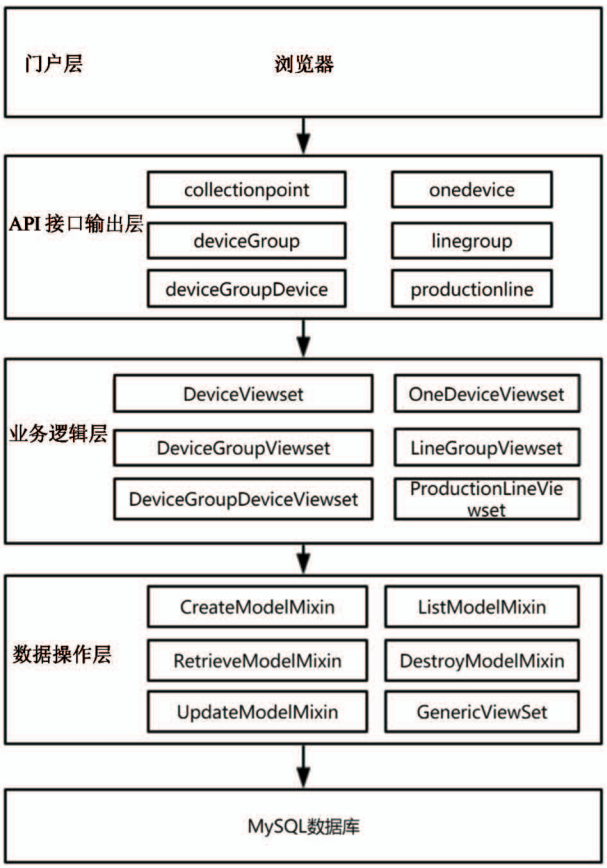

图 6　Web 系统框架图

图 6 中的门户层即前端可视化界面；图 6 中的 API 接口输出层、业务逻辑层和数据操作层组成后端服务程序，用于数据验证、数据查询，用户验证等操作。整个 Web 系统包括用户管理功能块，数据查询功能块，实时监控功能块和 3 维动画展示功能块。

3.1　用户管理

用户管理采用目前较流行的 session 与 token 机制进行验证，使用 JSON Web Token（JWT）作为跨域身份验证的解决方案，在双方之间使用 JSON 对象进行数据传输，该种方式通过使用密钥（HMAC 算法），RSA 或 ECDSA 的公钥/密钥键值对进行认证和信任，具有较高的安全性，原理如图 7 所示。用户管理模块实现用户短信验证注册和用户验证登录功能，具体实现方式如下。

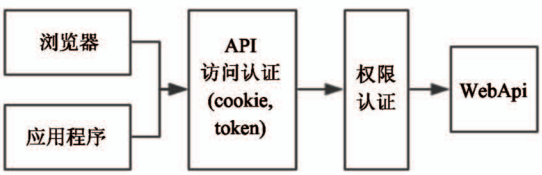

图 7　JWT 原理图

（1）用户短信验证注册。后端 Django 程序中编写 YunPian（）类方法，该函数调用底层数据传输协议，并依赖第 3 方短信服务商，实现短信发送功能，编写 generate _ code（）函数，调用随机数生成函数，生成4位验证码，编写UserRegSerializer（）类方法，该函数实现的原理是通过获取用户传输过来的验证码，与上文中生成的验证码作比较，从而验证短信验证码和密码格式的正确性，编写 SmsCodeViewset（）类方法，将通过验证的用户名和密码存入数据库；前端 VUE 程序[23]中编写 register 组件，用于搭建用户注册界面，同时使用 Html5，CSS3，JavaScript，jquery[24]美化界面。编写 register（）函数，通过后端生成的 api 接口，上传数据，实现用户注册功能。

（2）用户验证登录。编写 UserViewset（）类方法，用于获取前端上传的用户名和密码信息，编写 UserDetailSerializer（）类方法，用于验证用户名和密码正确性，若正确，使用 Response（re _ dict）函数返

回 session 与 token 给前端用户界面,同时在 settings 配置文件中导入 JWT,用于启用该登录验证功能块,通过 JWT 自带的登录验证的 api 接口 url(r^login/', obtain _ jwt _ token)完成用户登录功能;前端 Vue 程序中编写 login 组件,搭建用户登录界面,建立 store 状态管理器,用于存储用户登录的信息,如 session 与 token,实现用户登录功能。

3.2 数据查询

数据查询页面以节点树的形式展现,该方式可以简洁明了地展示所有设备和采集点,页面上通过查询设备,查看设备下的所有采集点数据,通过曲线图的形式展示数据。数据查询模块在功能上设计实现用户数据绑定,数据筛选和数据展示等 3 大功能,如图 8 所示,具体实现方式如下。

(1)用户数据绑定。编写 DeviceViewset( )类方法,该函数通过遍历 url 配置中的 as _ view 字典参数,获取元组类型的 items( ),从而获取用户信息,编写 SensorSerializer( )类方法,该函数用于获取该用户下的设备数据和采集点数据,以 JSON 格式提供给前端,同时在 url. py 文件使用路由注册函数,给前端提供 api 接口;前端 VUE 程序中编写 device 组件,搭建用户数据可视化界面,通过编写 getDeviceInfo( )函数,该函数用于获取相关数据,实现用户数据绑定功能。

(2)数据筛选。编写 CollectionPointViewset( )类方法,该函数通过遍历所有字典参数获取所有采集点数据,编写 collectionFilter( )类方法,该函数通过接收"|"前的变量值(value)和":"后的参数(args),返回一个值。从而解决时间筛选的问题,在该基础上,编写 filter _ backends( )类方法,用于实现数据排序,数据过滤和数据查询等功能。前端 VUE 程序中编写 deviceBrowses 组件,搭建用户数据筛选界面,如时间段筛选,设备名称筛选等,在此基础之上编写 getCollectionName = params = > { return, axios. get(`${local _ host}/collectionpoint/^, { params: params })}函数,将筛选条件发送到后端程序,后端获取筛选条件信息后,实现数据筛选功能。

(3)数据展示。在 Vue 框架中引入 jqplot 功能包,该功能包用于在前端页面绘制曲线图,通过调用 MYM. jqplot 函数,在该函数中添加数据列表,axes、axesDefaults、highlighter、cursor、title 等参数来绘制曲线图;服务端使用流行的 rest api 接口的形式给前端提供数据。前端页面通过 axios( )函数向服务端请求数据,服务端首先查看客户端是否已经登录,如果有则提供对用户开放的 api 接口。随后继续监听是否有具体请求某个 api 接口,如果有则将采集到的机器人最新数据封装成 JSON 格式发送到客户端,实现数据展示功能。

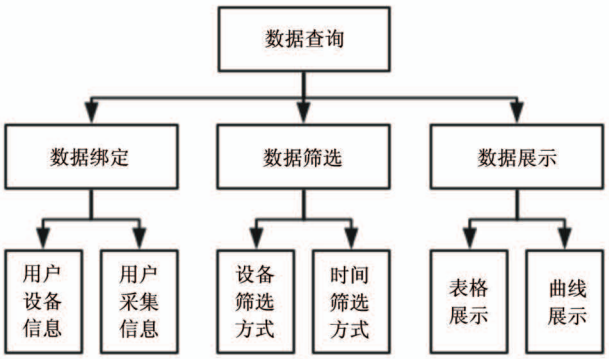

图 8 数据查询模块功能图

3.3 数据实时监控

数据实时监控展示主要分为自定义 Web 界面的数据监控与阿里云 DataV 的数据监控。

自定义 Web 界面数据监控是通过 MQTT 协议实现,实现与 MQTT 服务器交互、采集点筛选和数据实时显示等功能,如图 9 所示,具体实现方式如下。

(1)与 MQTT 服务器交互。在前端 Vue 项目中引入 Paho 功能包,用于实现 MQTT 客户端功能,生成 MQTT 客户端前先编写用于连接 MQTT 服务器的参数列表,如服务器 IP 地址、端口号等,具体为 ServerUri = ' mq. tongxinmao. com ' ServerPort = 18832;TimeOut = 5;KeepAlive = 100;CleanSession = false;SSL = false;在此基础上调用 new Paho. MQTT. Client 函数,创建 MQTT Client 实例,编写 onConnect( )函数和 onConnectionLost( )函数用于登录和断开 MQTT 服务器;编写 WriteToStatus( )函数,用于描述 MQTT 状态;编写 onMessageArrived( )函数,用于接收 MQTT 服务器的消息,实现与 MQTT 服务器数据交互功能。

(2)采集点筛选。在前端 Vue 项目中编写 getChartAllData( )函数,从 MQTT 服务器获取所有上传的数据,编写 editChartData( )函数,用于移除绘制到曲线图上的部分曲线,编写 collectData:function( )函数,用于获取实时数据,实现采集点筛选功能。

(3)数据实时显示。搭建在数据采集客户端的 MQTT Client,不断将数据以 0.5 s 间隔上传到 MQTT 服务器,位于前端界面的 MQTT 客户端订阅 MQTT 服务器上该主题,获取数据,编写 this. plot = MYM. jqplot('UserChart', SeriesData)函数,将采集到的数据实时显示在表格和曲线图中,通过修改 seriesDefaults、axes、legend、highlighter 等参数,更改曲线样式,实现数据实时显示功能。

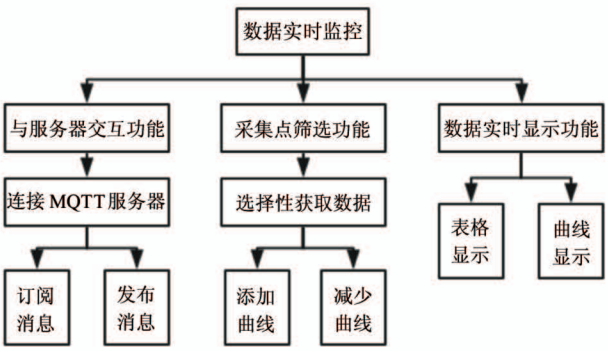

图 9　数据监控模块功能图

阿里云 DataV 的数据监控界面是通过使用阿里云 DataV 中的组件,搭建整个 UI 界面,DataV 搭建页面方便简洁,使用阿里云所提供的组件,拖拽到页面即可形成一个界面,界面酷炫。数据接口有多种方式,比如 api 接口,静态数据等。本方案使用最常见的 api 数据接口的形式。后端服务器只需要将数据以 json 的数据格式,以 api 接口方式开放,然后通过 DataV 获取数据,通过使用曲线表,数据列表等工具形成大屏界面,如图 10 所示。

3.4　三维动画展示

本模块使用 soildworks 对 SCARA 机器人进行 3D 建模。soildworks[25]与其他主流的 3 维制图软件相比,做机械机构设计,钣金设计等具有很大的优势。建模方案是将 SCARA 机器人分为 5 个部分进行建模,如底座,连杆等。图 11 所示是机器人底座 3 维模型图。对机器人的各部件进行建模后,将 3D 模型以 stl 的文件格式导出。

图 10　DataV 大屏展示图

图 11　底座 3 维模型图

在 3D 界面的开发中,需要使用到 WebGL 类库 three. js[26],three. js 在 WebGL 的基础上进行了进一步地封装和简化开发过程。在此基础上,需要导入 STLLoader. js 类库,该库功能是将 stl 文件导入到前端页面当中,具体实现方式如下。

(1)初始化场景。场景是所有物体的容器,在 Three. js 中场景就只有一种,用 Three. Scene 来表示,要构件一个场景需要 new 一个对象,代码为 var scene = new Three. Scene( )。

(2)初始化相机。相机类型选择透视相机,该类型所展示的效果更接近人眼视觉效果,代码是 This. camera = Three. PerspectiveCamera (70, 800/600, 0.1,10)。

(3)初始化渲染器。渲染器决定了渲染的结果并且以怎样的方式来绘制,代码如下:

```
var renderer = new Three. WebGLRenderer( )
renderer. setSize(800,600)
```

this. renderer. render(this. scene, this. camera)

在完成一个 3 维场景的基本框架后,添加 3 维物体、光源,实例化模型并加载对象,导入 SCARA 机器人各部件的 3 维模型,对模型对象的材质、网格、大小等进行定义,图 12 为机器人 3 维展示界面。

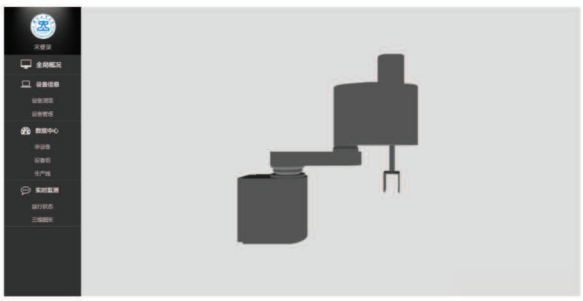

图 12　3 维展示界面

4　实验测试与分析

通过实验验证本系统的可行性,首先测试本地客户端的数据采集模块,界面如图 13 所示,上半部分用于设置机器人控制器的 IP 地址,连接控制器。下半部分用于查看节点树数据,数据操作和节点订阅等。当本地客户端运行时监测诊断如图 14 所示,从图中可看出当客户端开始采集数据时,进程内存会变大,但整个采集阶段一直处于平稳阶段,其中 CPU 占有率一直处于低消耗状态,具有较高的效率。

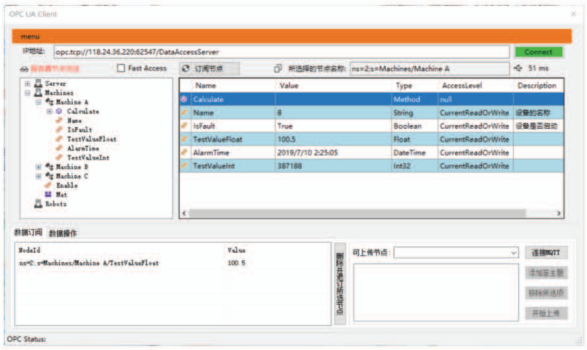

图 13　OPC UA 数据采集界面

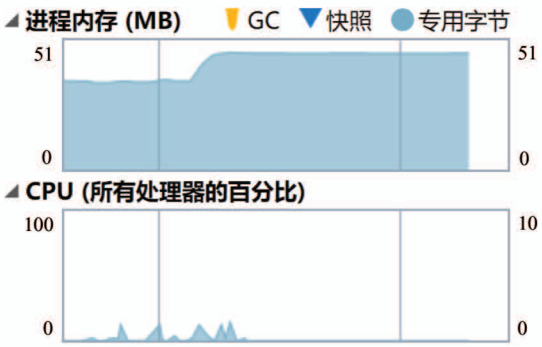

图 14　客户端运行监测诊断图

其次测试本地客户端的数据存储模块,界面如图 15 所示,从操作界面可知,数据有多种存储路径,用户可对不同种类的机器人进行数据采集以及存储,系统不需要额外设计数据库表进行数据存储,从而提高了系统的可扩展性。同时,通信效率达到每 50 ms 发送一个数据包,客户端采集的数据量为

图 15　数据存储模块界面

2 000,经过 MySQL 模块存储后,数据库接收到的数据量为 1 982,丢包率不超过 1%。

本地客户端的 MQTT 数据交互模块测试界面如图 16 所示,从操作界面可知,用户可根据自己的需求上传数据,对数据进行筛选。网页端的 MQTT 客户端测试界面如图 17 所示,主要包括连接 MQTT 服务器,查看数据节点,绘制波形图,移除曲线等操作。其中本地客户端的 MQTT 客户端数据上传间隔设置为 25 ms,网页端的 MQTT 客户端运行时监测诊断如图 18所示,由图可知,接收到数据的时间间隔在 25 ~30 ms 之间,延时较低,可以满足系统实时性的要求。

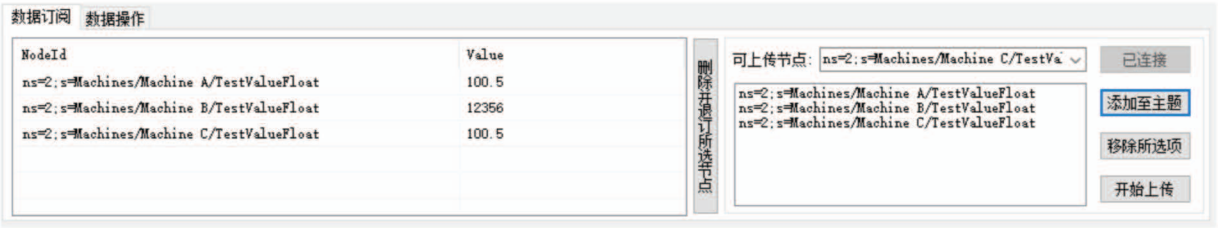

图 16　数据交互模块界面

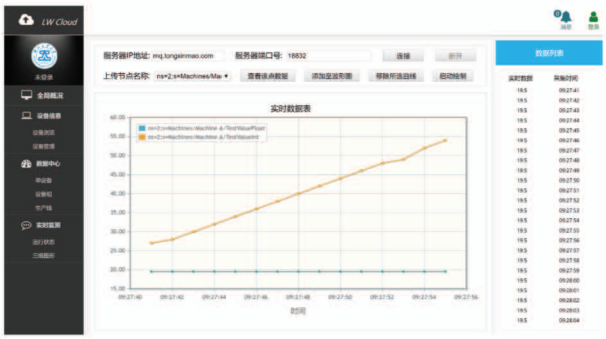

图 17　用户登录界面

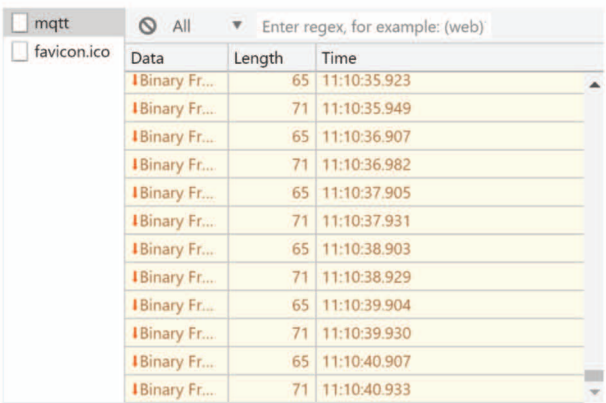

图 18　MQTT 客户端运行监测诊断图

Web 远程监控端的用户登录测试界面如图 19 所示，系统实现了用户的登录操作，与后端的数据交互信息如图 20 所示，由图可知，通过验证后只返回无规律的 token 值，保证了用户个人信息与数据的安全性。

Web 远程监控端的设备浏览测试界面如图 21 所示，设备列表栏用来显示设备以及采集点信息。设备浏览界面的右侧是用户所选择的设备的所有数据，该模块上半部分显示用户选取的单设备下的所有采集点信息，包括采集点名称，采集的数量，初次采集时间以及最后一次采集时间；下半部分显示用户选取的采集点的所有数据，包括数据的名称和采集时间。

图 19　用户登录界面

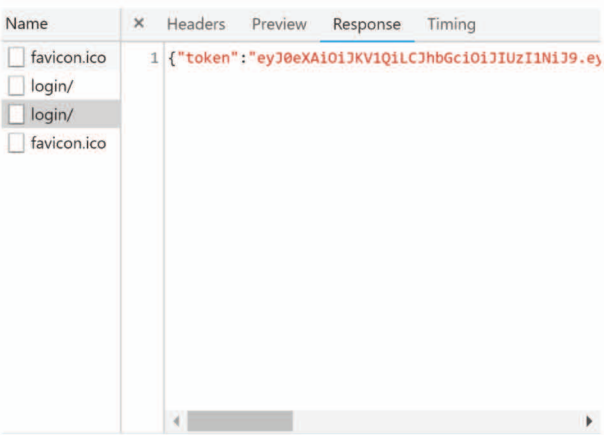

图 20　数据交互信息图

图 21　设备浏览测试界面

Web 远程监控端的数据查询测试界面如图 22 所示,设备数据查询界面的中间部分的顶部用来选择筛选条件,后端系统根据前端提供的筛选条件返回采集点数据和信息;中间部分的中部用来显示经过筛选查询后采集点信息,包括该采集点的名称,采集点总数,初次采集时间以及末次采集时间;中间部分的底部用来将采集点的数据曲线化的形式展示出来。页面的右侧部分用来显示所选取的采集点的所有数据以及采集时间。

图 22　数据查询测试界面

实验结果表明,系统分块化设计有利于整个系统的稳定性、高效性和可扩展性,同时前后端分离的方式使系统出现的错误更容易查找,维护更加容易。

5　结　论

本文研究了一种基于云平台的机器人监控系统,基于 OPC UA 协议、MySQL 数据库技术及前后端分离技术相结合进行搭建,同时,该系统挂载于阿里云平台。在数据采集过程中,不需要考虑底层系统的差异性,提高了开发效率,降低了开发成本。Web 系统采用前后端分离的方式,该方式开发周期短,系统稳定性高。系统针对 SCARA 机器人进行实时采集数据,通过 MQTT 客户端将数据传送到位于云端的 MQTT 服务器,位于可视化界面的 MQTT 客户端将 MQTT 服务器接收到数据进行实时展示和 3 维展示。目前市面上的机器人都具有以太网接口,且 OPC UA 协议使通讯有统一的标准,因此本系统对于开发机器人监控系统,具有一定的参考价值。

参考文献

[1] 张爱民, 孔得鹏, 王倩. 工业机器人的远程监控与诊断系统设计[J]. 机械, 2010, 37(10):45-47

[2] Sallinen M, Heikkila T, Koskinen J. Sensor based flexibility for robotics in manufacturing applications[C]// IEEE International Symposium on Industrial Electronics, Seoul, Korea, 2009: 1422-1427

[3] 骆晓娟,许力. 基于 AJAX 和 B/S 构架的实时监测系统[J]. 工业控制计算机,2013,26(4):64-65

[4] 徐建明, 吕汉泰, 张贵军,等. 基于 Web 的工业机器人 3D 虚拟动态监控系统[J]. 高技术通讯, 2017, 27(3):254-260

[5] Gubbi J, Buyya R, Marusic S, et al. Internet of things (IoT): a vision, architectural elements, and future directions[J]. *Future Generation Computer Systems*, 2013, 29(7):1645-1660

[6] Ji Z, Ganchev I, Máirtín O, et al. A cloud-based car parking middleware for IoT-based smart cities: design and implementation[J]. *Sensors*, 2014, 14(12): 22372-22393

[7] Douzis K, Sotiriadis S, Petrakis E G M, et al. Modular and generic IoT management on the cloud[J]. *Future Generation Computer Systems*, 2018, 78(1):369-378

[8] Baker T, Asim M, Tawfik H, et al. An energy-aware service composition algorithm for multiple cloud-based IoT applications[J]. *Journal of Network and Computer Applications*, 2017, 89(C):96-108

[9] Dinh T, Kim Y, Lee H. A Location-Based interactive model of internet of things and cloud (IoT-Cloud) for mobile cloud computing applications[J]. *Sensors*, 2017, 17(3):489

[10] Schmitt A, Carlier F, Renault V. Dynamic bridge generation for IoT data exchange via the MQTT protocol[J]. *Procedia Computer Science*, 2018, 130:90-97

[11] Mostefa M, Boudadi L K E, Loukil A, et al. Design of mobile robot teleoperation system based on virtual reality[C]// Proceedings of the 3rd International Conference on Control, Engineering and Information Technology, Tlemcen, Algeria, 2009: 2024-2029

[12] 杨硕, 彭思, 宋爱国,等. 基于虚拟现实的一对多远程康复训练机器人监控系统[J]. 高技术通讯, 2011, 21(2):191-195

[13] Králik M, Žáková K. Interactive WebGL model of hydraulic plant[J]. *IFAC-Papers OnLine*, 2015, 48(29): 146-151

[14] Pettit J B, Marioni J C. bioWeb3D: an online WebGL 3D data visualization tool[J]. *BMC Bioinformatics*, 2013, 14(3): 506-511

[15] 许锐炮. Socket 在 C#程序中的应用[J]. 科技视界, 2014, 21: 64-64

[16] 周敏. Nginx[J]. 程序员, 2007, 10: 115-116

[17] 刘洋, 刘明哲, 徐皑冬, 等. 基于消息代理的 OPC UA 发布/订阅模式研究与实现[J]. 高技术通讯, 2018, 28(6): 553-559

[18] 黄传禄. 基于 Python 的 MYSQL 数据库访问技术[J]. 现代信息科技, 2017, 1(4): 73-75

[19] 龚永罡, 付俊英, 汪昕宇. MQTT 协议在物联网中的应用研究[J]. 电脑与电信, 2017, 11: 89-91

[20] 齐金刚, 李滔, 李晋军. Django 框架 Web 数据查询分页技术研究[J]. 电子设计工程, 2014, 22(5): 33-37

[21] 高静, 段会川. JSON 数据传输效率研究[J]. 计算机工程与设计, 2011, 32(7): 2267-2270

[22] Wandschneider M. Learning node. js: a hands-on guide to building web applications in JavaScript, 2nd Edition [J]. *Pearson Schweiz Ag*, 2017, 58(2): 277-307

[23] 朱二华. 基于 Vue. js 的 Web 前端应用研究[J]. 科技与创新, 2017, 20: 119-121

[24] 周玲余. 基于 jQuery 框架的页面前端特效的设计与实现[J]. 计算机与现代化, 2013, 1: 61-63

[25] Vavro J, Kováčiková P, Bezdedová R. Kinematic and dynamic analysis of planar mechanisms by means of the solid works software[J]. *Procedia Engineering*, 2017, 177: 476-481

[26] 任宏康, 祝若鑫, 李风光等. 基于 Three. js 的真实三维地形可视化设计与实现[J]. 测绘与空间地理信息, 2015, 10: 51-54

Design of robot monitoring system based on cloud

Xu Jianming, Yu Junming, Dong Jianwei, Yu Li

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

Abstract

SCARA robot monitoring system is designed based on Alibaba Cloud in this paper, including local client and Web remote monitoring terminal. The local client is built based on the TCP/IP protocol and the C#, including data collect module based on OPC unified architecture (OPC UA), data storage module based on MySQL and data interaction module based on message queuing telemetry transport (MQTT) and cloud platform. The Web remote monitoring terminal is built by separating the front and rear ends. The Web backend service program is compiled based on Django rest framework, and the visual interface is compiled by the Vue framework. On this basis, WebGL and Three. js technologies are used to compiled the SCARA robot Web3D model on the interface. Finally, based on the OPC UA protocol, the interaction between the local client and the SCARA robot's motion state and data commands is realized, the data exchange with the MySQL database located in Alibaba Cloud is realized through MQTT, and the SCARA robot is monitored through the 3D visual interface.

Key words: Alibaba Cloud, OPC unified architecture (OPC UA), message queuing telemetry transport (MQTT), Django framework, Web remote monitoring