文章编号：2096-1472(2020)-12-15-04
DOI:10.19644/j.cnki.issn2096-1472.2020.12.005

工业物联网环境下设备数据采集研究与实现

欧阳旻[1,2]，郭玉超[1]，王 桓[3]，王志兵[1,2]

(1.湖南工业大学计算机学院，湖南 株洲 412007；
2.智能信息感知与处理技术湖南省重点实验室，湖南 株洲 412007；
3.电子科技大学中山学院，广东 中山 528402)
✉ouyangmin16@163.com; ihidchaos@qq.com; 349285070@qq.com; wzb@hut.edu.cn

摘 要：数据的采集和云端存储是工业物联网的重要应用，文章介绍了利用Modbus工业总线协议进行数据采集的方法，开发了基于Go语言的数据采集系统，解决工业设备海量数据的采集和存储问题，该系统使工业数据的采集不依赖于网关等特定硬件设备，减少了企业进行智能化改造的成本，数据可视化的呈现也为企业进行生产优化和决策提供了依据。

关键词：工业物联网；Go语言；数据采集
中图分类号：TP399 **文献标识码**：A

Research and Implementation of Device Data Acquisition based on the Environment of Industrial Internet of Things

OUYANG Min[1,2], GUO Yuchao[1], WANG Huan[3], WANG Zhibing[1,2]

(1.*College of Computer*, *HuNan University of Technology, Zhuzhou* 412007, *China*;
2. *Key Laboratoryof Intelligent Information Perception and Processing Technology*, *Zhuzhou* 412007, *China*;
3.School of Electron and Information Engineering, *University of Electronic Science and Technology of China*, *Zhongshan Institute*, *Zhongshan* 528402, *China*)
✉ouyangmin16@163.com; ihidchaos@qq.com; 349285070@qq.com; wzb@hut.edu.cn

Abstract: Data acquisition and cloud storage are important applications of industrial Internet of things. This paper introduces method of data acquisition using Modbus industrial bus protocol, and proposes a data acquisition system based on Go language to solve the problem of acquisition and storage of massive data of industrial devices. This system not only makes acquisition of industrial data independent of gateway and other specific hardware devices, but also reduces the cost of intelligent transformation of enterprises. Visualization of data also provides a basis for enterprises to optimize production and make decisions.

Keywords: industrial Internet of things; Go language; date acquisition

1 引言(Introduction)

随着工业物联网的发展，工业设备的智能化程度越来越高，然而设备的通讯受限于不同设备的物理链路、各种不同的协议，因此大多数数据不能互联互通。同时，现有的SQL(Structured Query Language)数据库也很难适应工业物联网中时间序列数据的存储特点，使得数据的存储和查询效率很低。工业物联网主要集中在生产和服务方面的应用，往往会涉及能源、交通、工业控制等质量较高的设备和资产，对运行安全提出了更高的要求[1,2]。文章介绍了一种新型的利用Modbus工业总线现场协议[3-5]进行数据采集的方法，实现了在云端对边缘设备进行数据采集的功能。借助这种方法，工业数据的采集可以不依赖于具体硬件网关，减少使用成本，

基金项目：“2018年湖南省自然科学基金项目”(2018JJ2097)；“2019年教育厅科学研究项目”(19C057)；国家重点研发项目(2018YFB1700204)；湖南省重点研发计划(2019GK2133)“2016年湖南省教育厅科学研究项目”(16C0482)“2018年教育厅科学研究项目”(18K077)的研究成果.

对于企业及时参与工业物联网的发展有一定的启示意义。

2 背景(Background)

目前，国内外厂商对于工业物联网数据采集[6]措施，主要有两种方案。一是利用嵌入式工业网关接入工业现场，或通过以太网，或通过串口，或通过OPC(OLE for Process Control)服务器等介质进行数据采集。这种方式需要购买厂商的硬件设备，往往价格高昂，并且与设备厂商的平台进行强绑定，不利于用户开发，难以满足个性化和定制化需求。优点是对于要求不多的客户，使用省心，不需要考虑很多，并且售后服务比较有保障。二是利用数据传输单元DTU(Data Terminal Unit)进行透明传输，将现场设备接入厂商云端，通过Socket方式实现数据采集。这种方式不需要购买或替换厂商的硬件网关，能直接利用用户已有的设备，减少了硬件成本开销。缺点是云服务价格取决于厂商，用户的话语权减弱，且迁移数据受限制。

3 需求及解决方案(Requirements and solutions)

该系统主要实现在云端[7]对边缘设备进行采集的功能，同时进行可视化的呈现。国内做数据采集工业网关的厂家非常多，证明这个技术方向有一定的研究价值。本采集系统使用Modbus+MQTT+InfluxDB的技术方案，经过验证，技术上具有一定的可行性，可以满足用户的数据采集需求。

3.1 数据采集

用户使用本数据采集系统时，首要的需求便是采集现场数据。这要求系统能够支持用户设备的通信协议[8]。经过分析研究，Modbus TCP(Modbus Transmission Control Protocol)与Modbus RTU(Modbus Remote Terminal Unit)协议已经能够满足用户的基础需求。另外，为了支持一些原始的串口协议，系统也加入了对于DTU透传模块的支持，经过正确配置后也可以通过DTU模块读取下属设备的数据。只有对协议的支持是远远不够的。为了能够方便用户的操作配置，系统的配置文件一定要清晰明了，不使用户感到迷惑。因此系统选用的JSON格式作为配置格式，既方便用户阅读和修改配置，也便于程序读取和解析。

3.2 数据存储

在满足用户的采集需求后，另一个问题便是数据存储问题。由于时序数据的天然特殊性，SQL类数据库并不适合存储该类数据。因此本系统选用时序数据库中性能较为优异的InfluxDB数据库作为存储方案，为如何收集数据，如何存储数据，如何处理和监视数据，以及如何可视化数据提出了合适的解决方案。另外，系统还提供了备选方案，如存储到MQTT(Message Queuing Telemetry Transport)中，后面接入消息队列，可以进一步处理；用户也可以选择直接存入云厂商数据库，例如，百度云天工TSDB数据库。

3.3 数据可视化

用户对于数据的分析和处理有着强烈的需求。例如，用户想要知道某台设备数据的波动情况，来判断这台设备是否稳定；或者根据数据的趋势来对后续数据的情况进行预测分析；这些都要求系统能够对数据进行可视化展示，或者对数据转发至具有可视化能力的平台中。为解决这个问题，系统提供多种方案，如Chronograf展示或Grafana展示，也可以借助百度云天工的物可视进行展示。

4 系统设计(System design)

本系统使用的开发软件为Jetbrains Goland，使用GO语言作为开发语言。系统采用Modbus作为数据采集的协议，MQTT作为配置文件下发和数据存储方案，InfluxDB和OpenTSDB作为时序数据库进行存储，用GO(Golang)语言进行代码编写，JSON(JavaScript Object Notation)协议作为配置文件和API(Application Programming Interface)交互的格式，使用Docker容器虚拟化平台进行系统的整体部署和安装。系统整体框架可分为三部分：主函数模块、采集驱动模块、存储驱动模块。

4.1 主函数模块

主函数模块为该模块程序的入口。程序主要执行的动作有：加载配置文件、加载存储驱动、加载采集驱动、开始采集任务、注册系统监听事件、阻塞主函数等待等。程序执行的流程为：首先从JSON文件中读取并加载配置文件，然后将存储模块的配置文件交给存储模块去加载，将采集模块的配置交给采集模块去加载。待加载完成后，循环遍历驱动字典，为每个采集驱动实例开辟线程，并开始采集任务。接着，程序注册了对于系统的监听事件，当按下键盘停止程序时，程序会循环遍历驱动字典并停止每个驱动实例。此外，程序还在主线程中一直循环等待阻塞，使得其他采集线程能一直执行。主函数模块流程图如图1所示。

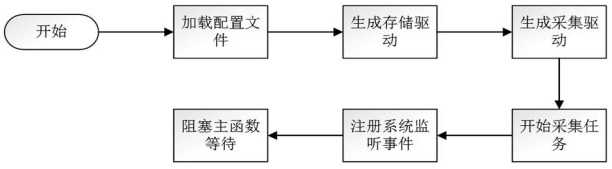

图1 主函数模块流程图

Fig.1 Flow chart of main function module

主函数模块代码如下：

```
func main() {
    config := loadConfig()
    S := drivers.NewStorage(config.Info.Storage)
     driverMap := drivers.NewDriver(config.Info.
Gateway, S)
```

```
for k := range driverMap {
    driver := driverMap[k]
    go driver.Start()
}
```

监听系统事件，有停止信号时关闭程序并销毁资源。

```
var stopLock sync.Mutex
stop := false
stopChan := make(chan struct{}, 1)
signalChan := make(chan os.Signal, 1)
go func() {
    <-signalChan
    stopLock.Lock()
    stop = true
    stopLock.Unlock()
    log.Println("Cleaning before stop...")
```

程序终止之前，先要关闭所有正在运行的任务，并销毁对应的资源。

```
    for k := range driverMap {
        driver := driverMap[k]
        go driver.Stop()
    }
    stopChan <- struct{}{}
    os.Exit(0)
}()
signal.Notify(signalChan, syscall.SIGINT, syscall.SIGTERM)
```

使用select一直阻塞主程序，等待其他函数执行，或者等待中断信号发送select {}。

4.2 采集驱动模块

该模块是程序的核心，该模块程序主要执行的动作有：解析存储配置、生成采集驱动、调度采集程序、开始和停止采集任务、校验数据类型、转换数据格式、监听网络端口等。程序执行的流程是：首先解析配置文件，判断每个驱动的采集协议，根据不同协议生成对应的采集驱动，然后开始采集。采集时，如果协议是Modbus，首先判断函数功能码，然后去调用对应的函数。采集到的结果是原始值，与用户的需求值不相吻合。因此调用转换函数。转换函数首先判断传进来的采集结果是否为空，为空则直接返回。如果采集到的数据不为空，首先判断采集点表中数据类型是否正确。如果正确，利用反射机制调用对应转换函数，将传进来的原始值转换为对应的需求值。如果协议是原始串口类型，则启动网络监听程序，等待客户端连接。如果有客户端请求连接，判断客户端合法性，合法则接受客户端连接，否则拒绝。服务器校验通过并接受连接后，程序发送采集指令到客户端，等待客户端返回结果，然后同样的将数据进行转换操作后返回。采集驱动模块流程图如图2所示。

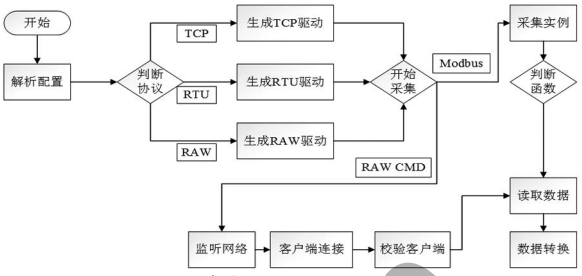

图2 采集驱动模块流程图

Fig.2 Flow chart of the acquisition drive module

4.3 存储驱动模块

这一部分是程序的结尾部分，也是程序向上传输的通道。当每个采集驱动采集到数据并转换完成后，都会调用存储模块，对数据进行写入。存储驱动的流程是：获取到数据后，根据对应的存储驱动启用与否，调用对应的存储函数进行写入，可以是写入数据库，也可以是发送到MQTT的Topic里面去。数据发送后，如果失败则打印相应的日志，如果成功，返回并进行下一次数据的发送。

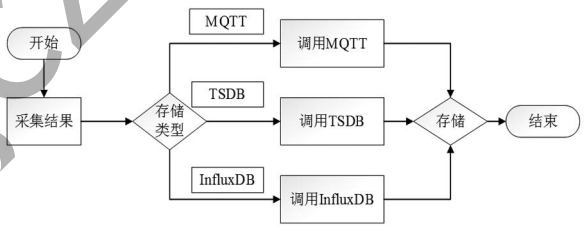

图3 存储驱动模块流程图

Fig.3 Flow chart of storage drive module

5 系统实现(System implementation)

系统使用Go语言编写，在Goland里面添加Docker远程部署，链接到服务器，点击运行即可将系统部署到服务器。开发中需要使用的代码文件和用途描绘如表1所示。

表1 项目部分GO代码说明

Tab.1 Description of GO codes in the project

文件名称	文件说明
main.go	程序入口
collect.go	采集程序
define.go	配置文件的结构体定义
dispatcher.go	采集驱动调度
driver.go	定义采集驱动的生成、开始、停止方法
format.go	定义数据转换方法
listen.go	Socket监听程序
storage.go	定义存储驱动生成和写入的方法
influxdb.go	生成InfluxDB数据库实例
mqtt.go	生成MQTT实例
tsdb.go	生成百度云TSDB实例

系统使用Docker部署。需要编写Dockerfile和entrypoint.sh代码。Dockerfile代码内容如下：

```
FROM alpine:latest
MAINTAINER GuoYuchao <ihidchaos@qq.com>
ENV LANG C.UTF-8
WORKDIR /root/
COPY ./bin bin
COPY ./entrypoint.sh entrypoint.sh
ENTRYPOINT ./entrypoint.sh
```

entrypoint.sh代码内容如下：

```
#!/bin/sh
cd ~/bin/ || exit
./linux_amd64 &
status=$?
if [ $status -ne 0 ]; then
  exit $status
fi
while true ; do
    sleep 1
done
```

在Goland里面添加Docker远程部署配置，连接到服务器，点击运行即可将本系统部署到服务器。部署完后进行系统测试，先查看服务器是否存在在Docker容器中，显示系统已经成功部署，如图4所示。

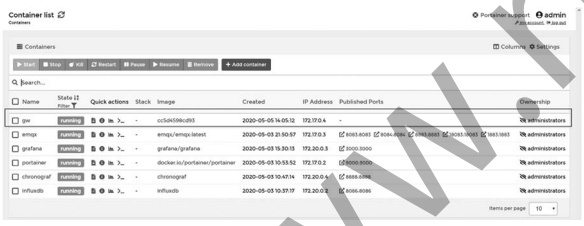

图4 系统部署结果

Fig.4 System deployment results

部署成功后进行数据采集结果测试、数据存储测试，以及数据可视化测试。先看采集程序是否采集到了数据，接着检查InfluxDB和OPEN_TSDB数据库，再检查数据是否正确的发送到了数据库中，最后检查TSDB数据可视化，数据可视化结果如图5所示。

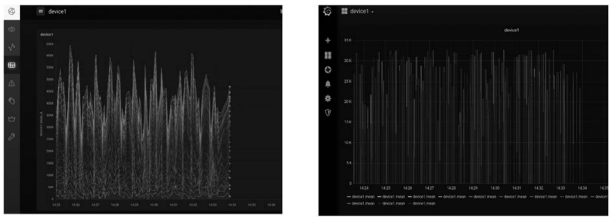

图5 Chronograf数据可视化和Grafana数据可视化

Fig.5 Chronograf data visualization & Grafana data visualization

6 结论(Conclusion)

本系统解决了传统的数据采集系统使用C语言和一些单片机产品组合开发存在的开发效率低下，产品价格昂贵，部署、维护难度高花费大等缺点，实现了数据采集、数据存储、数据可视化的功能，满足了工业物联网环境下企业对数据采集和智能化生产的需求，借助这种方法，工业数据的采集可以不依赖于具体的硬件网关，减少了企业智能化改造的成本，同时数据可视化的呈现为企业进行生产优化和决策提供了依据，助力企业智能化的发展。

参考文献(References)

[1] 王鹏.基于Modbus协议的数据采集系统的研究[D].合肥工业大学,2019:6-7.

[2] Y. Peng, P. Liu, T. Fu. Performance analysis of edge-PLCs enabled Industrial internet of things[J]. Peer-to-Peer Networking and Applications, 2020,13(5):1830-1838.

[3] 何建忠.基于MODBUS协议的工业网关设计与实现[D].内蒙古大学,2013:5-10.

[4] 刘立博,丁茹.基于Modbus协议的台达DVP系列PLC位姿监控功能扩展[J].中国设备工程,2020(21):184-186.

[5] 阎坤.基于JAVA实现的Modbus通讯模块在温测系统中的应用[J].工业控制计算机,2019(11):33-36.

[6] 刘金.大规模集群状态时序数据采集、存储与分析[D].北京邮电大学,2018:3-4.

[7] 王妙琼,魏凯,姜春宇.工业互联网中时序数据处理面临的新挑战[J].信息通信技术与政策,2019,5:4-9.

[8] 胡存,骆德汉,童怀.基于Modbus与MQTT融合工业能耗网关系统设计[J].物联网技术,2019,9(04):49-54.

作者简介：

欧阳旻(1975-)，男，硕士，讲师.研究领域：软件开发，网络技术.

郭玉超(1998-)，男，本科生.研究领域：软件开发，网络技术.

王 桓(1980-)，男，博士，副教授.研究领域：工业时间序列预测及非线性系统分析.

王志兵(1974-)，男，硕士，副教授.研究领域：大数据，软件工程.