ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.19, No.20, July 2023

E-mail: xsjl@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

DOI:10.14004/j.cnki.ckt.2023.1025

基于Qt5开发的面向工业设备的状态监视及控制软件

刘佳梁

(中国科学院长春光学精密机械与物理研究所,吉林 长春 130000)

摘要:文章介绍了如何使用Qt Creator软件开发一种工业设备状态监视及控制软件。选择Qt Creator作为开发环境,采用C++编程语言进行软件开发。软件功能主要包含监视系统内各分系统故障状态、状态显示及告警,并对设备关键节点进行控制,如继电器、步进电机等。软件通过网络通信收集各分系统状态并向控制器发送控制指令,网络通信基于UDP/IP协议并采用组播的方式,数据库使用QSQLite,存储设备关键故障及状态信息,便于回放及分析。通过参考文中的通信模块、数据库模块及界面显示模式,读者可以快速掌握基于Qt Creator开发环境开发一种工业设备状态监视及控制软件。

关键词: Qt Creator;C++;状态监视;网络通信;数据库

中图分类号:TP311 **文献标识码:**A

文章编号:1009-3044(2023)20-0070-03

开放科学(资源服务)标识码(OSID):

21世纪以来,我国工业化进程快速发展,工业自动化设备呈现高精度、高集成的发展趋势,与此同时,由于电子系统故障引发的灾难性事故时有发生,也因此造成大量的人力、物力损失,针对现代设备故障状态监控、设备管理及远程控制,研发一款设备状态监视及控制软件,提供设备健康监视平台,实现设备远程管理及控制,符合当前设备管理的发展方向[1]。因此,提出了一种基于Qt Creator开发环境,使用C++编程语言开发的设备状态监视及管理软件,软件编写过程中使用的Qt Creator版本为5.9.1,MinGw版本为5.3.0,运行平台为X86架构计算机,操作系统为Windows 10专业版。

1 软件工作流程

软件基于UDP/IP协议并采用组播的方式与系统内各设备进行通信,将接收到的来自各设备数据包解码后分发给数据管理线程和界面管理线程,数据管理线程负责对数据进行分类、提取并存储,便于事后数据回放及分析[2]。界面管理线程实时更新设备状态显示,对于异常状态进行告警提示,同时响应界面用户操作,实现对设备远程控制。软件工作流程图如图1所示。

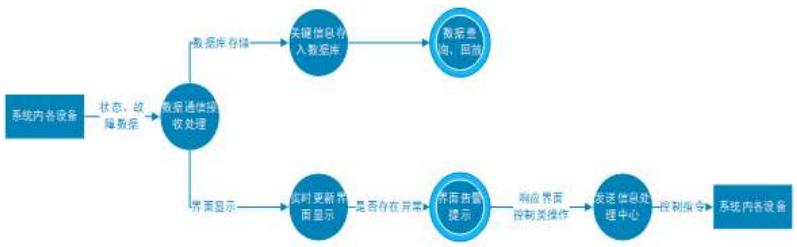

图1 软件工作流程图

2 软件界面

Qt Designer(界面设计师)提供了非常丰富的界面控件,使用鼠标简单拖拽控件即可创建出程序界面框架,同时运用Qt布局管理系统可以对程序界面实现布局、美化。软件主界面如图2所示。

图2 软件主界面

Qt提供了信号与槽的机制,其中信号会在特定情况下被触发,槽函数则可以理解为与信号相对应的响应函数,使用connect函数将信号与槽函数进行连接,也可以通过disconnect函数断开信号与槽函数之间的连接关系,当界面中控件状态发生变化时,系统会自动发射相应信号,只需将处理函数与信号连接,便可实现界面操作响应。

信号与槽函数有三个特点:1)信号之间可以互相连接;2)一个信号可以连接多个槽函数;3)多个信号可以连接同一个槽函数。这三个特点决定了信号与槽函数的应用非常灵活、便利,同时能够极大降低对象之间的耦合度[3]。

收稿日期:2023-02-12

作者简介:刘佳梁(1994—),男,黑龙江哈尔滨人,研究实习员,硕士研究生,主要研究方向为信息处理。

当界面中存在较多同类型控件需要响应点击操作时，可以通过findChildren函数遍历界面中同类型控件，在for循环中执connect操作，减少重复代码并且易于维护[4]，如图3所示。

```
//找到界面中所有同类型控件
QList<QPushButton *> pbtns = ui->frame->findChildren<QPushButton *>();
foreach (QPushButton *pbtn, pbtns)
{
    pbtn->setIconSize(icoSize);
    pbtn->setMinimumWidth(icoWidth);
    connect(pbtn, SIGNAL(clicked()), this, SLOT(buttonClick()));
}
```

图3　遍历界面中同类型控件

findChildren函数找到了MainWindow下的所有QPushButton类型按钮。然后给每一个按钮设置图标和尺寸，通过connect函数连接到buttonClick()槽函数[5]。槽函数代码如图4所示。

```
void MainWindow::buttonClick()
{
    QPushButton *b = (QPushButton *)sender();
    //获取当前点击事件按钮的名称
    QString name = b->objectName();
    if (name == "mainwindow") {
        ui->stackedWidget->setCurrentIndex(0);
    } else if (name == "infoquery") {
        ui->stackedWidget->setCurrentIndex(1);
    } else if (name == "logquery") {
        ui->stackedWidget->setCurrentIndex(2);
    } else if (name == "sysseting") {
        ui->stackedWidget->setCurrentIndex(3);
    } else if (name == "exut") {
        exit(0);

    }
}
```

图4　槽函数代码

3 网络通信

系统采用udp组播的方式进行网络数据收发。Qt提供QUdpSocket类用于实现udp通信，进行udp通信前需要通过以下步骤对套接字进行初始化[6]。

3.1 指定网卡

通常计算机中存在多块网卡，QNetworkInterface::allInterfaces()可以获得计算机中所有网卡，通过MAC地址识别出指定网卡，使用setMulticastInterface函数指定用该网卡用于组播通信。

3.2 生存时间

设置组播数据的生存时间TTL(Time-To-Live)。TTL指一个数据报到达目的地址之前跳过网络的最大次数，可以理解为数据报每跨1个路由生存时间就会减1，程序中将生存时间设置为1，表示数据报只能在同一路由下的局域网内传播。

3.3 禁止回环

通过setSocketOption(QAbstractSocket::MulticastLoopbackOption,0)函数，设置禁止回环，表示本机不能接收自身发出的数据。

3.4 绑定IP地址及端口

通过bind函数绑定IP地址和端口，从而进行网络数据报收发。当有数据报传入时会自动发射readyRead()信号，通过connect()函数将readyRead()信号与ReceiveData()槽函数连接，即可在ReceiveData()函数中读取并处理接收到的数据报。

3.5 加入组播组

使用joinMulticastGroup()函数加入指定组播组地址，便可以接受该组播组中的数据报。网络初始化函数代码如图5所示。

```
void NetUdp::Init()
{
    m_UdpSocket = new QUdpSocket;
    foreach (QNetworkInterface oNetInterface, QNetworkInterface::allInterfaces())
    {
        QNetworkInterface::InterfaceFlags flags = oNetInterface.flags();
        if(!(flags.testFlag(QNetworkInterface::IsRunning) && !flags.testFlag(QNetworkInterface::IsLoopBack))
        {
            continue;
        }
        if(oNetInterface.hardwareAddress() == "11:11:11:11:11:11")//找到指定网卡
        {
            oNet = oNetInterface;

        }
    }
    m_UdpSocket->setMulticastInterface(oNet);//指定网卡
    m_UdpSocket->setSocketOption(QAbstractSocket::MulticastTtlOption,1);//组播的数据的生存期
    m_udpSocket->setSocketOption(QAbstractSocket::MulticastLoopbackOption,0);//禁止本机反环数据
    m_udpSocket->joinMulticastGroup(m_McastAddr);
    if(m_UdpSocket->bind(QHostAddress(m_bindIP), m_bindPort, QAbstractSocket::ShareAddress | QAbstractSocket::ReuseAddressHint))
    {
        Writelog::Instance()->addLog(LogLevel_normalLevel, QString::fromLocal8Bit("组播网络初始化为:  IP: %1  Port: %2").arg(m_bindIP).arg(m_bindPort));
    }
    else
    {
        Writelog::Instance()->addLog(LogLevel_normalLevel, QString::fromLocal8Bit("组播网络初始化为:  IP: %1  Port: %2").arg(m_bindIP).arg(m_bindPort));
    }
    connect(m_UdpSocket, SIGNAL(readyRead()), this, SLOT(ReceiveData()));
}
```

图5　网络初始化函数代码

网络数据接收函数代码如图7所示，当套接字中存在未读取数据时，hasPendingDatagrams()函数返回true。pendingDatagramSize()的值表示第一个数据报的长度，同时初始化一个对应长度的QbyteArray变量，将readDatagram()读取的数据报内容存入变量中，在processData()函数中对读取的数据报进行解码等处理[7]。网络数据接收函数代码如图6所示。

```
void NetUdp::ReceiveData()
{
    while (m_UdpSocket->hasPendingDatagrams())
    {
        QByteArray datagram;
        int bsize = m_UdpSocket->pendingDatagramSize();
        datagram.resize(bsize);
        m_udpSocket->readDatagram(datagram.data(), bsize);
    }
    processData(datagram);//处理数据
}
```

图6　网络数据接收函数代码

网络数据发送函数代码如图7所示。

```
void NetUdp::SendData(QByteArray data)
{
    QHostAddress destAddr;
    destAddr.setAddress(m_DestIP);
    m_UdpSocket->writeDatagram(data, data.size(), destAddr, m_DestPort);
}
```

图7　网络数据发送函数代码

4 数据库

本软件包含历史状态信息查询及事后分析功能，选择使用数据库实现历史信息存储及查询操作。Qt为数据库操作提供了QtSql模块，其中包含一套无关于开发平台和数据库类型的调用接口，通过运用该模块，实现数据库和应用程序的无缝衔接。开发人员只需掌握基本的SQL语句，即可实现数据库应用程序开发。结合本软件对数据库的性能需求，最终选择使用QSQLite数据库[8]。

qt_sql_default_connection为数据库的默认连接名称，实际应用中，可以通过QSqlDatabase::addDatabase()函数的第二个参数指定连接名称。第一个参数QSQLite表示使用SQLite数据库。如果默认连接不存在，则创建连接并添加数据库。数据库打开函数代码如图8所示。

```
bool MyDB::openDb()
{
    QSqlDatabase db;
    if (QSqlDatabase::contains("qt_sql_default_connection"))
    {
        db = QSqlDatabase::database("qt_sql_default_connection");
    }
    else
    {
        db = QSqlDatabase::addDatabase("QSQLITE");
        db.setDatabaseName(QApplication::applicationDirPath() + "/database/" + "data.db");
        db.setUserName("datainfo");
        db.setPassword("123456");
    }
    if(!db.open())
    {
        qDebug()<<"error: failed to connect sqlite3 database."<< db.lastError();
        return false;
    }
    else {
        qDebug()<<"success to connect sqlite3 database.";
        return true;
    }
}
```

图8　数据库打开函数代码

SQLite的创建表语句为CREATE TABLE，同时还可以判断表是否已经存在。创建表函数代码如图9所示，创建了一个名为temp_data的数据表，包含7列，第一列是id，类型是整型，设置为主键且自增长。第二列是TIME，类型是可变字符串，最长20个字符，五至六列为info，数据类型是浮点值。第七列为STATUS，数据类型是带符号的整数[9]。

```
void MyDB::createTable(void)
{
    //用于执行sql语句的对象
    QSqlQuery query;
    //构建创建数据库的sql语句字符串
    QString str = QString("CREATE TABLE IF NOT EXISTS temp_data( \
                          id INTEGER PRIMARY KEY AUTOINCREMENT, \
                          TIME          VARCHAR(20) NOT NULL, \
                          Info1         REAL     NOT NULL, \
                          Info2         REAL     NOT NULL, \
                          Info3         REAL     NOT NULL, \
                          Info4         REAL     NOT NULL, \
                          STATUS        INTEGER      NOT NULL \
                          );");
    query.exec(str);
}
```

图9　创建表函数代码

SQLite的插入语句是INSERT INTO，插入函数代码如图10所示。

```
bool MyDB::insertRecord(const InfoData &infodata)
{
    QSqlQuery query;
    query.prepare("INSERT INTO temp_data(id, TIME, Info1, Info2, Info3, Info4, STATUS)" \
                  "VALUES(NULL, :TIME, :Info1, :Info2, :Info3, :Info4, :STATUS)");
    query.bindValue(":id", infodata.id);
    query.bindValue(":TIME", infodata.dateTime);
    query.bindValue(":Info1", infodata.info1);
    query.bindValue(":Info2", infodata.info2);
    query.bindValue(":Info3", infodata.info3);
    query.bindValue(":Info4", infodata.info4);
    query.bindValue(":STATUS", infodata.status);
    if(!query.exec())
    {
        qDebug() << "insertRecord error: " << query.lastError();
        return false;
    }
    return true;
}
```

图10　数据库插入函数代码

QSqlQueryModel类为SQL结果集提供了一个只读数据模型，是用于执行SQL语句和遍历结果集的高级接口，可用于为QTableView等视图类提供数据。从数据库中提取数据操作如图11所示，通过canFetchMore()、fetchMore()函数，提取数据库中整个结果集[10]。数据库数据获取函数代码如图11所示。

```
QSqlQueryModel* MyDB::getData()
{
    QSqlQueryModel *model = new QSqlQueryModel;
    model->setQuery("SELECT * FROM temp_data", db);
    while(model->canFetchMore())
        model->fetchMore();
    return model;
}
```

图11　数据库数据获取函数代码

5 结束语

Qt是一个跨平台的C++图形用户界面应用程序框架，提供给应用程序开发者建立图形用户界面所需的所用功能，同时，Qt还具有优良的跨平台特性，并提供丰富的API。在工程项目开发中，使用Qt进行程序开发极大降低了使用者的学习成本。

主要介绍了使用Qt软件进行界面搭建、网络数据通信模块以及数据库的建立和使用，在实际应用过程中，软件同时接收20余个分系统的实时状态信息，对故障状态进行提示告警，同时将关键信息纳入数据库，并对设备关键节点进行控制。基于文中介绍的程序模型，可快速开发出适用于类似场景的状态监视及控制软件。

参考文献：

[1] 许丽佳.电子系统的故障预测与健康管理技术研究[D].成都:电子科技大学,2009.
[2] 陆文周.Qt 5开发及实例[M].3版.北京:电子工业出版社,2017:292-294.
[3] 李全虎.交互界面开发工具-Qt[J].中国科技信息,2005(5):33.
[4] 安峰.QT平台上的动态可定制界面设计[J].单片机与嵌入式系统应用,2014,14(3):24-25,28.
[5] 缪雨润.基于Qt的图形用户界面的研究与实现[D].南京:东南大学,2015.
[6] 尹圣雨.TCP/IP网络编程[M].金国哲,译.北京:人民邮电出版社,2014:103-109.
[7] 杜召辉,刘安东.基于Qt的移动机器人上位机软件设计与实现[J].计算机测量与控制,2018,26(5):107-111.
[8] 李帅,刘全利,王伟.基于QT的车载监控系统主控单元软件设计及实现[C].中国过程控制会议,2014.
[9] 杨中华.基于Qt/Embedded的SQLite数据库研究及应用[D].成都:西华大学,2008.
[10] 闵孝忠,朱林立.QT环境下通用数据库组件技术的研究与设计[J].计算机应用与软件,2015,32(12):231-234.

【通联编辑:梁书】