6k 1绪论

--研究背景

--国内外研究现状

--研究目标

3k 2康复理论基础

--临床治疗

--传统康复治疗

--机器人辅助治疗

3k 3系统总体设计方案

--机械部分

--控制部分

--软件设计

8k 4系统详细设计与实现

----数据存储

基于MySQL数据库结构设计，查论文

参考吴刘海师兄论文

----串口通信

查论文

串口通信是上下位机进行通信的一种常用的通信协议，大部分单片机中都有一到多个串口资源经过简单的配置就可以实现上下位机的通信，下图是串口通信协议中的一种形式。常用的是8位数据位加起始位加停止位，因此串口是只能发送0-255区间的数据。因此想要发送int或者float型的数据需要按高地位或者到内存中取出数据来发送，此时就需要通信协议来确保数据的准确性。

qt中集成了QSerialport类，可以直接使用该类实现串口通信。在开始使用这部分的时候也遇到不少困难，现在一步步说起如何通过QT实现通过串口资源进行通信。在建工程的时候记得勾选QSerialport选项，如果建工程的时候没选可以按图中的方式再设置或者通过添加附加依赖项添加也可以。

首先在头文件中实例化串口。

[cpp] view plain copy print?

#include <QtSerialPort/QSerialPort>

#include <QtSerialPort/QSerialPortInfo>

QSerialPort serialPort;

然后需要查看系统上的串口资源，然后将串口资源显示在界面上。

[cpp] view plain copy print?

foreach(const QSerialPortInfo &info, QSerialPortInfo::availablePorts())

{

qDebug() << "Name : " << info.portName();

qDebug() << "Description : " << info.description();

qDebug() << "Manufacturer: " << info.manufacturer();

//将串口资源显示在界面

if (ui.comboBox->currentText() != info.portName())

ui.comboBox->addItem(info.portName());

}

找到串口资源后需要对其一些属性进行配置，包括波特率等等，通过槽函数实现。

[cpp] view plain copy print?

void uart::ComOpen()

{

ui.pushButton->setText("Open");

serialPort.setPortName(ui.comboBox->currentText());//选取串口

serialPort.open(QIODevice::ReadWrite); //打开串口

serialPort.setBaudRate(ui.BaudRate->currentText().toInt());

//qDebug() << ui.BaudRate->currentText().toInt() << endl;

serialPort.setDataBits(QSerialPort::Data8);

serialPort.setParity(QSerialPort::NoParity);

serialPort.setStopBits(QSerialPort::OneStop);

serialPort.setFlowControl(QSerialPort::NoFlowControl);

timer.start(5000);//500ms定时器

}

这样串口部分已经可以正常工作了，串口发来数据怎么检测呢？QSerialport中提供了readRead()信号，当有数据发来的时候就调用ReadData函数。这样就可以实现串口数据的读取了。需要对数据进行解码等操作都可以在该函数中进行，稍后会稍微说下简单的和校验。

[cpp] view plain copy print?

QObject::connect(&serialPort, SIGNAL(readyRead()), this, SLOT(ReadData()));//发来数据就读取

void uart::ReadData()

{

ui.buffSize->setText(QString::number(serialPort.bytesAvailable()));

int buffersize = ui.bufferSize->value();

if (serialPort.bytesAvailable()>buffersize){ //更改过滤个数，提高通信速率

//requestData = serialPort.readAll().toHex();//转成 hex

requestData = serialPort.readAll();//字符型

ui.textEdit->append(requestData);

}

if (!timer.isActive())

timer.start(5000);//500ms定时器

}

当然还需要对数据的发送指令，这部分很简单，直接上代码。

[cpp] view plain copy print?

void uart::dataSend()

{

QString data = ui.lineEdit->text();

QByteArray DATA;

DATA = data.toLatin1();

serialPort.write(DATA);

}

----Socket通信

查论文

QTcpSocket 类提供一个TCP套接字

TCP是一个面向连接，可靠的的通信协议，非常适合于连续不断的数据传递

QTcpSocket 是QAbstractSocket类非常方便的一个子类，让你创建一个TCP连接和数据流交流。

注意：TCP套接字不能以QIODevice::Unbuffered模式来打开

在Symbian系统上，程序想用这个类的话必须拥有NetworkServices平台支持，如果客户机缺少这个能力，将会导致"恐惧"（不明白为什么这么翻译）

成员函数：

QTcpSocket::QTcpSocket ( QObject \* parent = 0 )

以UnconnectedState态创建一个QTcpSocket对象

QTcpSocket::~QTcpSocket () [virtual]‘

析构函数，销毁对象

QTcpServer类

提供一个TCP基础服务类继承自QObject

这个类用来接收到来的TCP连接，可以指定TCP端口或者用QTcpServer自己挑选一个端口，可以监听一个指定的地址或者所有的机器地址。

调用listen()来监听所有的连接，每当一个新的客户端连接到服务端就会发射信号newConnection()

调用nextPendingConnection()来接受待处理的连接。返回一个连接的QTcpSocket()，我们可以用这个返回的套接字和客户端进行连接

如果有错误，serverError()返回错误的类型。调用errorString()来把错误打印出来。

当监听连接时候，可以调用serverAddress()和serverPort()来返回服务端的地址和端口。

调用close()来关闭套接字，停止对连接的监听。‘

尽管QTcpServer大多时候设计使用事件循环，也可以不适用事件循环，可以使用waitForNewConnection()，会一直阻塞，知道一个连接可以用或者超时。

在Symbian平台上，同样的需要NetworkServices平台支持。

成员函数

------------------------------

void close() 关闭服务，然后服务器讲不再监听任何连接

QString errorString()const错误时候返回错误的字符串

------------------------------

bool hasPendingConnections()const如果服务端有一个待处理的连接，就返回真，否则返回假

QTcpSocket\* nextPendingConnection()

返回一个套接字来处理一个连接，这个套接字作为服务端的一个子对象，意味着当QTcpServer对象销毁时候，这个套接字也自动删除，当使用完后明确的删除这个套接字也好，这样可以避免内存浪费。当没有可处理的连接时候，这个函数返回0。

注意：返回的套接字不能再其他线程中使用。如果想在其他线程中使用，那么你需要重载incomingConnection()

-------------------------------------

void incomingConnection(int socketDescriptor)[virtual protected]

当QTcpServer有一个新连接时候调用这个虚函数，socketDescriptor参数是新连接的套接字描述符

这个函数新建一个QTcpSocket套接字，建立套接字描述符，然后存储套接字在一个整形的待连接链表中。最后发射信号newConnection()

重写这个函数，当一个新连接时候，来调整这个函数的行为。

当服务端使用QNetworkProxy服务器代理时候，使用一般的套接字函数套接字描述符可能不可以用，这时候应该使用QTcpSocket::setSocketDescriptor()来设置描述符

--------------------------------------

bool isListening()const

当服务端正在监听连接时候返回真，否则返回假

bool listen( const QHostAddress & address = QHostAddress::Any,quint16 port = 0 )

告诉服务端监听所有来自地址为address端口为Port的连接，如果Port为0，那么会自动选择，如果address是QHostAddress::Any,那么服务端监听所有连接，成功返回1，否则返回0

int maxPendingConnections()const

返回最大允许连接数。默认是30

void setMaxPendingConnections(int numConnections)

设定待处理的连接最大数目为numConnections,当超过了最大连接数后，客户端仍旧可以连接服务端，但是服务端不在接受连接，操作系统会把这些链接保存在一个队列中。

----QThread

二、QThread线程

QThread是Qt线程中有一个公共的抽象类，所有的线程都是从QThread抽象类中派生的，要实现QThread中的纯虚函数run(),run()函数是通过start()函数来实现调用的。

创建线程对象的实例，调用QThread::start()，在子线程类run()里出现的代码将会在新建线程中被执行。

QCoreApplication::exec()总是在主线程(执行main()的那个线程)中被调用，不能从一个QThread中调用。在GUI程序中，主线程也被称为GUI线程，是唯一允许执行GUI相关操作的线程。另外，必须在创建一个QThread之前创建QApplication(or QCoreApplication)对象。

QThread的两种使用方法:

A、不使用事件循环。

a. 子类化 QThread

b. 重载 run 函数，run函数内有一个 while 或 for 的死循环

c. 设置一个标记为来控制死循环的退出。

B、使用事件循环。

a. 子类化 QThread，

b. 重载 run 使其调用 QThread::exec()

c. 为子类定义信号和槽，由于槽函数并不会在新开的 thread 运行，很多人为了解决这个问题在构造函数中调用 moveToThread(this)

Bradley T. Hughes 给出说明是： QThread 应该被看做是操作系统线程的接口或控制点，而不应该包含需要在新线程中运行的代码。需要运行的代码应该放到一个QObject的子类中，然后将该子类的对象moveToThread到新线程中。

6k 5系统界面实现

流程图

-用户管理

-康复评估

-康复训练

----计划制定

---游戏设计

3k 6实验

2k 7结论