QT界面实现重要代码部分：

客户端代码：

#include "clientwidget.h"

#include "ui\_clientwidget.h"

#include "qdebug.h"

clientWidget::clientWidget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::clientWidget)

{

ui->setupUi(this);

socket = new QTcpSocket(this);

connect(ui->pushButton\_connect,SIGNAL(clicked()),this,SLOT(onConnect())); //点击相关按钮触发相关槽函数

connect(ui->pushButton\_disconnect,SIGNAL(clicked()),this,SLOT(onDisconnect()));

connect(ui->pushButton\_send,SIGNAL(clicked()),this,SLOT(onSend()));

connect(socket,SIGNAL(connected()),this,SLOT(imcoming())); //管道建立后触发相关函数

ui->pushButton\_send->setShortcut(tr("Ctrl+R")); //设置发送相关快捷键

}

clientWidget::~clientWidget()

{

delete socket;

delete ui;

sum=0;

}

/\*连接按钮点击后的函数\*/

void clientWidget::onConnect()

{

ui->pushButton\_connect->setDisabled(true); //设置相关按键使能与否

socket->connectToHost(QString(ui->lineEdit\_socket->text()),9090); //这里可以在lineEdit里写相应ip，如果只是测试只要输入localhost即可，且配置好端口

if(!socket->waitForConnected(3000)) //等待3s，如果管道没接通那么则行{}内容

{

ui->pushButton\_connect->setEnabled(true);

ui->textEdit->append("connect failed!");

return;

}

ui->pushButton\_disconnect->setEnabled(true); //设置相关按键使能与否

ui->pushButton\_send->setEnabled(true);

ui->textEdit->append("connected to host");

// connect(socket,SIGNAL(readyRead()),this,SLOT(onread()));

}

/\*点击Disconnect按钮后执行函数\*/

void clientWidget::onDisconnect()

{

socket->abort(); //强行断开管道

ui->textEdit->append("disconnected!"); //设置相关按键使能与否

ui->pushButton\_connect->setEnabled(true);

ui->pushButton\_disconnect->setDisabled(true);

ui->pushButton\_send->setDisabled(true);

// connect(socket,SIGNAL(readyRead()),this,SLOT(onread()));

}

/\*点击Send按钮后触发函数\*/

void clientWidget::onSend()

{

socket->write(ui->lineEdit->text().toUtf8()); //写信息进管道

// socket->write("Hello",strlen("Hello")); //写信息进管道

socket->flush(); //推信息

ui->lineEdit->clear();

}

/\*读信息函数\*/

void clientWidget::onread()

{

QSqlQuery query;

char bufclient[100]; //缓冲区

char w[5];

char s[5];

QString da;

QString tt;

memset(bufclient,0,sizeof(bufclient)); //清空缓冲区

int lengthc = socket->bytesAvailable(); //从管道中计算传送数据长度

if(lengthc>0)

{

socket->read(bufclient,lengthc); //把指定长度数据从管道中放入缓冲区buf

ui->textEdit->insertPlainText("\n");

ui->textEdit->insertPlainText("server : "); //显示相关发信息人，以及时间等

ui->textEdit->insertPlainText(QDateTime::currentDateTime().toString("yyyy-MM-dd "));

ui->textEdit->insertPlainText(QTime::currentTime().toString("hh:mm:ss"));//textEdit文本框显示当前时间

ui->textEdit->insertPlainText("\n");

ui->textEdit->append(bufclient); //显示信息

ui->textEdit->insertPlainText("\n");

da=QDateTime::currentDateTime().toString("yyyy-MM-dd");

tt=QTime::currentTime().toString("hh:mm:ss");

memset(w,0,sizeof(w));

memset(s,0,sizeof(s));

w[0]=bufclient[0];

w[1]=bufclient[1];

s[0]=bufclient[3];

s[1]=bufclient[4];

/\*bool rtn = query.exec(QString("insert into table values('%1', '%2', '%3', %4, %5, 0, %6)")

.arg(id).arg(brand).arg(type).arg(price).arg(num).arg(num));\*/

query.exec(QString("insert into type values('%1','%2','%3','%4')").arg(da).arg(tt).arg(w).arg(s));

}

//ui->textEdit->append(bufclient);

}

/\*管道建立后，触发的相关函数\*/

void clientWidget::imcoming()

{

ui->textEdit->append("connected.");

connect(socket,SIGNAL(readyRead()),this,SLOT(onread())); //准备，然后触发读信息槽

connect(socket,SIGNAL(disconnected()),this,SLOT(enServerButton()));//管道断开，触发相关槽

}

/\*断开后，各按键相关使能函数\*/

void clientWidget::enServerButton()

{

ui->pushButton\_connect->setEnabled(true); //各按键使能

ui->pushButton\_disconnect->setDisabled(true);

ui->pushButton\_send->setDisabled(true);

// socket->abort();

}

void clientWidget::on\_opensqlbutton\_clicked()

{

QSqlQuery query;

// int r=0;

query.exec("select \* from type");

while(query.next())

{sum++;}

/\* query.exec("select \* from type");

QStandardItemModel \*model=new QStandardItemModel(sum,4,this);

int k=0;

qDebug()<<query.value(k).toString();

while(query.next())

{

QStandardItem \*item=new QStandardItem(query.value(0).toString());

model->setItem(r,0,item);

QStandardItem \*item1=new QStandardItem(query.value(1).toString());

model->setItem(r,1,item1);

QStandardItem \*item2=new QStandardItem(query.value(2).toString());

model->setItem(r,2,item2);

QStandardItem \*item3=new QStandardItem(query.value(3).toString());

model->setItem(r,3,item3);

r++;

\*/

/\* qDebug()<<query.value(0).toString()<<query.value(1).toString()<<query.value(2).toString()<<query.value(3).toString();

ui->textEdit\_2->append(query.value(0).toString());

ui->textEdit\_2->append(query.value(1).toString());

ui->textEdit\_2->append("wendu:");

ui->textEdit\_2->append(query.value(2).toString());

ui->textEdit\_2->append("shidu:");

ui->textEdit\_2->append(query.value(3).toString());

ui->textEdit\_2->append("\n");

\*/

/\*

}

this->hide();

tableView=new QTableView;

tableView->setModel(model);

tableView->show();

this->show();

sum=0;

\*/

Dialog2 dialog;

dialog.exec();

}

void clientWidget::on\_cleanbutton\_clicked()

{

QSqlQuery query;

query.exec("delete from type");

sum=0;

}

void clientWidget::on\_pushBotton\_clear\_clicked()

{

ui->lineEdit\_socket->clear();

}

终端代码：

#include "widget.h"

#include "ui\_widget.h"

Widget::Widget(QWidget \*parent) :

QWidget(parent),

ui(new Ui::Widget)

{

ui->setupUi(this);

//connect(t,SIGNAL(ab()),this,SLOT(change()));

myCom= new Posix\_QextSerialPort("/dev/ttyUSB0",QextSerialBase::Polling);

//readTimer->start(100);

myCom->open(QIODevice::ReadWrite);

//以读写方式打开串口

myCom->setBaudRate(BAUD9600);

//波特率设置，我们设置为9600

myCom->setDataBits(DATA\_8);

//数据位设置，我们设置为8位数据位

myCom->setParity(PAR\_NONE);

//奇偶校验设置，我们设置为无校验

myCom->setStopBits(STOP\_1);

//停止位设置，我们设置为1位停止位

myCom->setFlowControl(FLOW\_OFF);

//数据流控制设置，我们设置为无数据流控制

myCom->setTimeout(10);

readTimer=new QTimer(this);

readTimer->start(1000);

connect(readTimer,SIGNAL(timeout()),this,SLOT(readMyCom()));

ui->textEdit->insertPlainText("1");

}

void Widget::change()

{

ui->startthread\_button->setEnabled(true);

ui->stopthread\_button->setEnabled(true);

}

void delay(unsigned int a)

{

unsigned int i;

while (--a!=0)

{

for(i=0;i<600;i++);

}

}

Widget::~Widget()

{

delete ui;

}

void Widget::readMyCom()//读取串口数据并显示出来

{

ui->textEdit->clear();

ui->textEdit\_2->clear();

QString temp = myCom->readAll();

//ui->textBrowser->insertPlainText(temp);

//读取串口缓冲区的所有数据给临时变量temp

int index = temp.indexOf(":");

//int index=2;

wendu= temp.mid(index-2,2);

//delay(1000);

shidu= temp.mid(index+1,2);

// delay(1000);

ui->textEdit->insertPlainText(wendu);

// ui->textEdit->insertPlainText(temp);

//delay(1000);

ui->textEdit\_2->insertPlainText(shidu);

//将串口的数据显示在窗口的文本浏览器中

/\*char sendbuf[100]={0};

QByteArray ba = temp.toLatin1();

strcpy(sendbuf,ba.data());

socket->write(sendbuf,strlen(sendbuf)); //把相关数据送入管道

socket->flush(); //防止数据较少不能自动发送，所以推一把

\*/

thread.add(temp);

delay(100);

temp.clear();

}

void Widget::on\_pushButton\_clicked()

{

}

void Widget::on\_startthread\_button\_clicked()

{

thread.start();

ui->startthread\_button->setEnabled(false);

ui->stopthread\_button->setEnabled(true);

}

void Widget::on\_stopthread\_button\_clicked()

{

if(thread.isRunning())

{

thread.stop();

ui->startthread\_button->setEnabled(true);

ui->stopthread\_button->setEnabled(true);

}

}