Mainwindow.h文件

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QSerialPort>

#include <QSerialPortInfo>

#include <QSqlDatabase>

#include <QTimer>

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~*MainWindow*();

private slots:

void on\_btnOpen\_clicked();

void ReadData();

void ReadData\_N();

void timeUpdate();

void on\_startButton\_clicked();

void on\_stopButton\_clicked();

void on\_btnOpen\_n\_clicked();

private:

Ui::MainWindow \*ui;

QSerialPort \*serial;

QSerialPort \*esp\_serial;

QSqlDatabase db;

QTimer \*timer;

QByteArray Combuf\_n;

QString wen,shi,hw;

int iwen,ishi;

};

#endif // MAINWINDOW\_H

Main.cpp

#include "mainwindow.h"

#include <QApplication>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

MainWindow w;

w.show();

return a.exec();

}

Mainwindow.cpp

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <Qserialport>

#include <QSerialPortInfo>

#include <QSqlError>

#include <QMessageBox>

#include <QSqlDatabase>

#include <QSqlQuery>

#include <QDebug>

#include <QTimer>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

//数据库连接代码

db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("localhost");

db.setDatabaseName("zigbee");

db.setUserName("root");

db.setPassword("");

if(!db.open())

{

QMessageBox::critical(0,QObject::tr("无法打开数据库"),

"无法创建数据库连接！",QMessageBox::Cancel);

return;

}

else {

qDebug()<<"suceess Connect!";

}

timer = new QTimer(); //定时器

//遍历获取串口信息

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

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

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

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

QSerialPort serial;

serial.setPort(info);

if(serial.*open*(QIODevice::ReadWrite))

{

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

ui->cmbPortName\_n->addItem(info.portName());

serial.*close*();

}

}

ui->cmbBaudRate->setCurrentIndex(0);

ui->cmbBaudRate\_n->setCurrentIndex(0);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

void MainWindow::on\_btnOpen\_clicked()

{

if(ui->btnOpen->text() == tr("打开数据串口"))

{

serial = new QSerialPort;

//设置串口名

serial->setPortName(ui->cmbPortName->currentText());

//打卡串口

serial->*open*(QIODevice::ReadWrite);

//设置波特率

serial->setBaudRate(ui->cmbBaudRate->currentText().toInt());

//设置数据位数

switch (ui->cmbDataBits->currentIndex())

{

case 8:

serial->setDataBits(QSerialPort::Data8);

break;

default:

break;

}

//设置校验位

switch (ui->cmbParity->currentIndex())

{

case 0:

serial->setParity(QSerialPort::NoParity);

break;

default:

break;

}

//设置停止位

switch (ui->cmbStopBits->currentIndex())

{

case 1:

serial->setStopBits(QSerialPort::OneStop);

break;

case 2:

serial->setStopBits(QSerialPort::TwoStop);

default:

break;

}

//设置流控制

serial->setFlowControl(QSerialPort::NoFlowControl);

//关闭设置菜单使能

ui->cmbPortName->setEnabled(false);

ui->cmbBaudRate->setEnabled(false);

ui->cmbDataBits->setEnabled(false);

ui->cmbParity->setEnabled(false);

ui->cmbStopBits->setEnabled(false);

ui->btnOpen->setText(tr("关闭数据串口"));

//连接信号槽

QObject::connect(serial,&QSerialPort::readyRead,this,&MainWindow::ReadData);

}

else

{

//关闭串口

serial->clear();

serial->*close*();

serial->deleteLater();

//恢复设置使能

ui->cmbPortName->setEnabled(true);

ui->cmbBaudRate->setEnabled(true);

ui->cmbDataBits->setEnabled(true);

ui->cmbParity->setEnabled(true);

ui->cmbStopBits->setEnabled(true);

ui->btnOpen->setText(tr("打开数据串口"));

}

}

void MainWindow::on\_btnOpen\_n\_clicked()

{

if(ui->btnOpen\_n->text() == tr("打开网络串口"))

{

esp\_serial = new QSerialPort;

//设置串口名

esp\_serial->setPortName(ui->cmbPortName\_n->currentText());

//打卡串口

esp\_serial->*open*(QIODevice::ReadWrite);

//设置波特率

esp\_serial->setBaudRate(ui->cmbBaudRate\_n->currentText().toInt());

//设置数据位数

switch (ui->cmbDataBits\_n->currentIndex())

{

case 8:

esp\_serial->setDataBits(QSerialPort::Data8);

break;

default:

break;

}

//设置校验位

switch (ui->cmbParity\_n->currentIndex())

{

case 0:

esp\_serial->setParity(QSerialPort::NoParity);

break;

default:

break;

}

//设置停止位

switch (ui->cmbStopBits\_n->currentIndex())

{

case 1:

esp\_serial->setStopBits(QSerialPort::OneStop);

break;

case 2:

esp\_serial->setStopBits(QSerialPort::TwoStop);

default:

break;

}

//设置流控制

esp\_serial->setFlowControl(QSerialPort::NoFlowControl);

//关闭设置菜单使能

ui->cmbPortName\_n->setEnabled(false);

ui->cmbBaudRate\_n->setEnabled(false);

ui->cmbDataBits\_n->setEnabled(false);

ui->cmbParity\_n->setEnabled(false);

ui->cmbStopBits\_n->setEnabled(false);

ui->btnOpen\_n->setText(tr("关闭网络串口"));

//连接信号槽

QObject::connect(esp\_serial,&QSerialPort::readyRead,this,&MainWindow::ReadData\_N);

QObject::connect(timer,&QTimer::timeout,this,&MainWindow::timeUpdate);

}

else

{

//关闭串口

esp\_serial->clear();

esp\_serial->*close*();

esp\_serial->deleteLater();

//恢复设置使能

ui->cmbPortName\_n->setEnabled(true);

ui->cmbBaudRate\_n->setEnabled(true);

ui->cmbDataBits\_n->setEnabled(true);

ui->cmbParity\_n->setEnabled(true);

ui->cmbStopBits\_n->setEnabled(true);

ui->btnOpen\_n->setText(tr("打开网络串口"));

}

}

//数据串口读取算法

void MainWindow::ReadData()

{

QByteArray Combuf;

if(serial->*bytesAvailable*()>7)

{

Combuf = serial->readAll();

qDebug()<<Combuf;

}//因为QT接收串口数据会断层，故用bytesAvailable函数，作用是当接收数据大于5时才调用readall接收数据

if(!Combuf.isEmpty())

{

QByteArray allArray,jingArray,qianArray,ywArray;

allArray = Combuf;

if(allArray.contains("#") && allArray.contains("$"))

{

QString checkData;

QString checkData2,checkData3;

checkData = allArray.data();

checkData2 = checkData.mid(0,1);

checkData3 = checkData.mid(5,1);

if(checkData2 == "$" && checkData3 == "#")

//yw&hw

{

QString data;

bool hok;

data = allArray.data();

hw = data.mid(7,1);

int ihw = hw.toInt(&hok,10);

qDebug()<<ihw;

switch(ihw)

{

case 1:ui->hwLabel->setText("有人,已报警!");break;

case 0:ui->hwLabel->setText("无人");break;

}

//wenshi

jingArray = allArray.split('#').at(0);

qDebug()<<jingArray;

for(int i=1;i<jingArray.length();i++)

{

qianArray[i-1]=jingArray[i];

}

qDebug()<<qianArray;

//QByteArray转int

QString str\_data;

str\_data=qianArray.data();

wen = str\_data.mid(0,2);//温度

shi = str\_data.mid(2,2);//湿度

bool wenok,shiok;

iwen = wen.toInt(&wenok,10);

ishi = shi.toInt(&shiok,10);

ui->wdlabel->setText(wen);

ui->sdlabel->setText(shi);

qDebug()<<wen;

qDebug()<<shi;

QByteArray w\_buf,s\_buf;

w\_buf = wen.toLatin1();

s\_buf = shi.toLatin1();

if(ui->btnOpen\_n->text() == tr("关闭网络串口"))

{

//当网络串口按下时才执行if里面的语句，建立esp8266和手机的tcp连接 esp\_serial->write("AT+CIPSTART=0,\"TCP\",\"192.168.4.2\",8080\r\n");

}

//数据库插入部分；

QSqlQuery query(db);

//使用占位符：使得能在sql语句用变量

query.prepare("insert into cgq(wd,sd,hw) values(:iwen,:ishi,:ihw)");

//数字表示变量在sql语句的位置，后面即对应变量，这个函数实现变量和占位符的绑定。

query.bindValue(0,iwen);

query.bindValue(1,ishi);

query.bindValue(2,ihw);

//这一句一定要写！

query.exec();

}

}

}

Combuf.clear();

}

void MainWindow::ReadData\_N()

{

timer->start(100);

Combuf\_n.append(esp\_serial->readAll());

}

//延时函数

void MainWindow::timeUpdate()

{

timer->stop();

qDebug()<<Combuf\_n;

if(Combuf\_n.length()!=0 && !Combuf\_n.contains("OK") && Combuf\_n.contains(":")){

QString code = Combuf\_n.data();

QStringList list = code.split(':');

QString str = list[1];

qDebug()<<list[1];

if(str.mid(0,5) == "start"){

on\_startButton\_clicked();

}

else if(str.mid(0,4) == "stop"){

on\_stopButton\_clicked();

}

}

if(Combuf\_n.contains("OK")){

QByteArray w\_buf,s\_buf;

w\_buf = wen.toLatin1();

s\_buf = shi.toLatin1();

esp\_serial->write("Tmperature:");

esp\_serial->write(w\_buf);

esp\_serial->write("Humidity:");

esp\_serial->write(s\_buf);

if(hw == "1"){

esp\_serial->write("有人");

}

else if(hw == "0"){

esp\_serial->write("无人");

}

}

//如果成功建立tcp连接，esp8266设置指令，开始向手机发送数据

if(Combuf\_n.contains("CONNECT")){

if(iwen > 10 && ishi > 10)

{

esp\_serial->write("AT+CIPSEND=0,30\r\n");

}

else if (iwen < 10 || ishi < 10)

{

esp\_serial->write("AT+CIPSEND=0,29\r\n");

}

else if (iwen < 10 && ishi< 10)

{

esp\_serial->write("AT+CIPSEND=0,28\r\n");

}

}

qDebug()<<Combuf\_n;

Combuf\_n.clear();

}

void MainWindow::on\_startButton\_clicked()

{

serial->write("start");

QMessageBox::information(this,"TIPS","开始检测",QMessageBox::Cancel);

}

void MainWindow::on\_stopButton\_clicked()

{

serial->write("stop");

QMessageBox::information(this,"TIPS","停止检测",QMessageBox::Cancel);

}