

**课 程 实 验 报 告**

**课程名称： 射频识别技术原理及应用**

**专业班级： 物联网1501**

**学 号： U201514888**

**姓 名： 陈艺欣**

**指导教师： 甘早斌**

**报告日期： 2018.6.26**

**计算机科学与技术学院**

**目 录**

[1 实验一 低频读写器实验 3](#_Toc514840089)

[1.1 实验目的 3](#_Toc514840090)

[1.2 实验内容及结果 3](#_Toc514840091)

[1.3 实验体会与总结 4](#_Toc514840092)

[1.4 核心源码说明 5](#_Toc514840093)

[2 实验二 高频读写器实验ISO14443A 9](#_Toc514840094)

[2.1 实验目的 9](#_Toc514840095)

[2.2 实验内容及结果 9](#_Toc514840096)

[2.3 实验体会与总结 14](#_Toc514840097)

[2.4 核心源码说明 15](#_Toc514840098)

[3 实验三 高频读写器实验ISO15693 29](#_Toc514840099)

[3.1 实验目的 29](#_Toc514840100)

[3.2 实验内容及结果 29](#_Toc514840101)

[3.3 实验体会与总结 32](#_Toc514840102)

[3.4 核心源码说明 33](#_Toc514840103)

[4 实验四 超高频读写器实验 53](#_Toc514840104)

[4.1 实验目的 53](#_Toc514840105)

[4.2 实验内容及结果 53](#_Toc514840106)

[4.3 实验体会与总结 55](#_Toc514840107)

[4.4 开发实例源码 55](#_Toc514840108)

实验五 RFID综合应用实验

1.1 实验目的

本次实验属于综合性应用实验，要求用户能够灵活应用 RFID 技术原理，解决实际生活中遇到的应用问题，培养用户分析问题、解决问题的能力以及综合知识的应用能力。由于RFID技术应用范围非常广泛，本次实验限定应用13.56M读写器、基于ISO14443A协议的电子标签、基于ISO15693协议电子标签开发两套综合应用系统。

1.2 需求分析

1.2.1 功能分析

采用北京博创 RFID 实验箱模拟图书管理系统的读卡设备、支持 ISO15693 协议的 S50 卡（5 张）模拟图书，一张卡作为用户身份的唯一识别卡，其他四张 卡与唯一的一本图书关联。

用户首次申请领用该卡（称用户卡）时，保存个人信息（姓名，性别，年龄）至数据库中，并对卡进行初始化，即写入个人信息，写至块0。将卡与个人姓名关联起来（采用实名制）。另外四张卡初始化与四本图书关联，在数据库中录入卡号、书名、作者、出版社、总本数、剩余本数、可借时间，标识图书的唯一性。将最近的 5 次借阅信息同时存储在数据库中。

不同的图书可供借阅的时间长短不一样，用户利用该用户卡借阅不同图书，在卡内记录最近五条借还明细，借还明细同时写入系统数据库表中。当有借阅书籍超期未还时，将不能借阅新的书籍。

读写器设备与上位机始终保持联系，上位机与数据库服务器始终保持联系。

1.2.2 数据信息

1)数据信息描述

(1) 用户基本信息表（卡号，姓名，性别，年龄）；

(2) 用户借/还信息表（卡号，姓名，电子标签Id，借阅时间）；

(3) 图书基本信息表（电子标签Id，图书名称，作者，出版社，总本数，剩余本数，可借时间）；

2）数据字典

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 主/外键 | 字段值约束 | 说明 |
| cardID | vchar | 主键 |  | 卡的编号 |
| name | vchar |  |  | 持卡人姓名 |
| gender | vchar |  |  | 持卡人性别 |
| age | int |  |  | 持卡人年龄 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 字段类型 | 主/外键 | 字段值约束 | 说明 |
| booksID | vchar | 主键 |  | 书的编号 |
| name | vchar |  |  | 书的名称 |
| author | vchar |  |  | 书的作者 |
| publishing\_house | vchar |  |  | 书的出版社 |
| count | int |  |  | 书的总共数量 |
| residue | int |  |  | 书的剩余数量 |

1.3 实验内容及结果

基于 RFID 技术的图书管理系统设计与实现。

1. 系统功能
2. 用户发卡管理；

刷卡获取卡号，填写用户姓名、性别、年龄，将信息添加到数据库表中，并将姓名信息写入卡内0块。

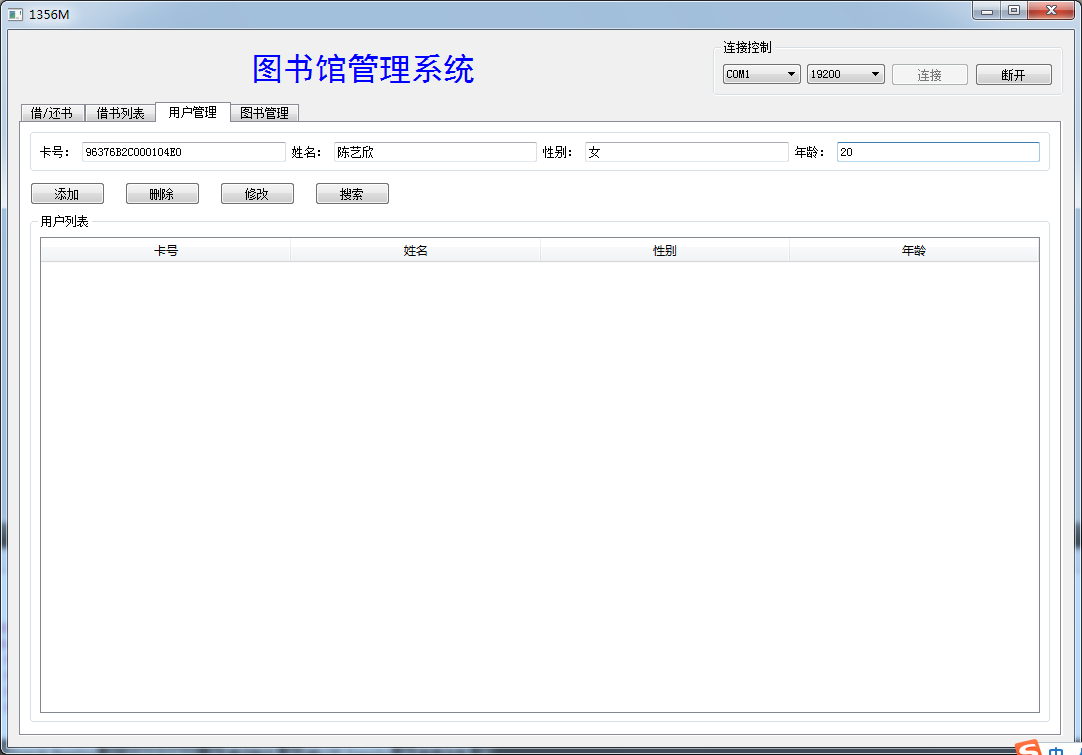


图 1 输入用户信息

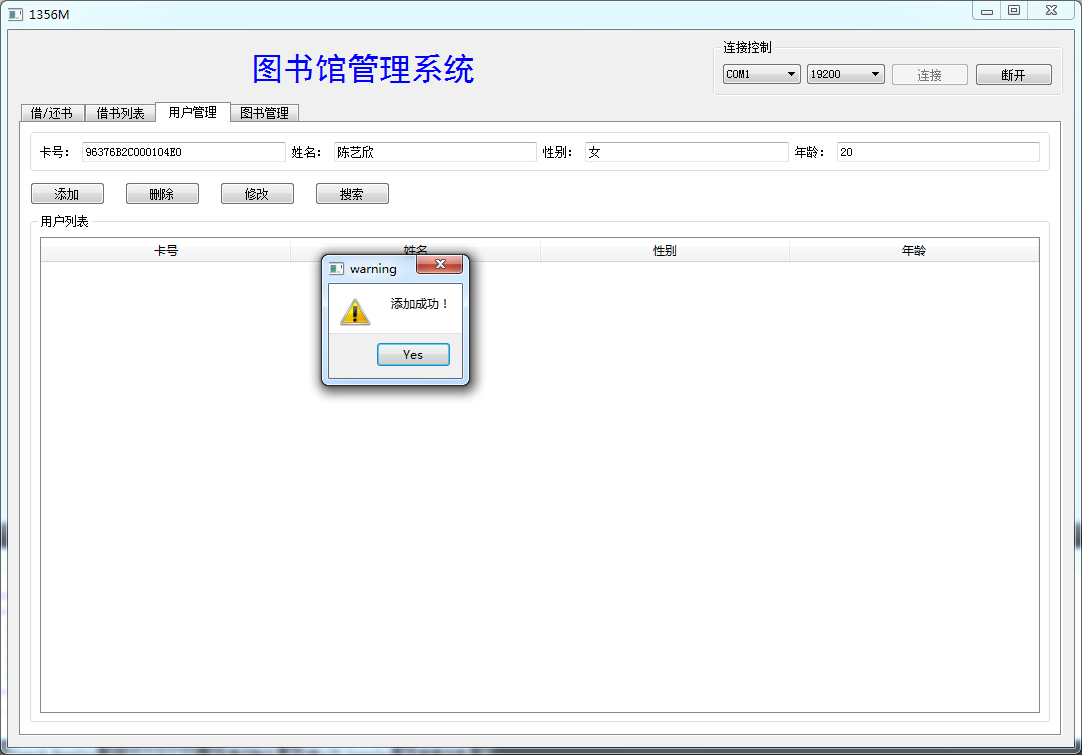


图 2 添加成功

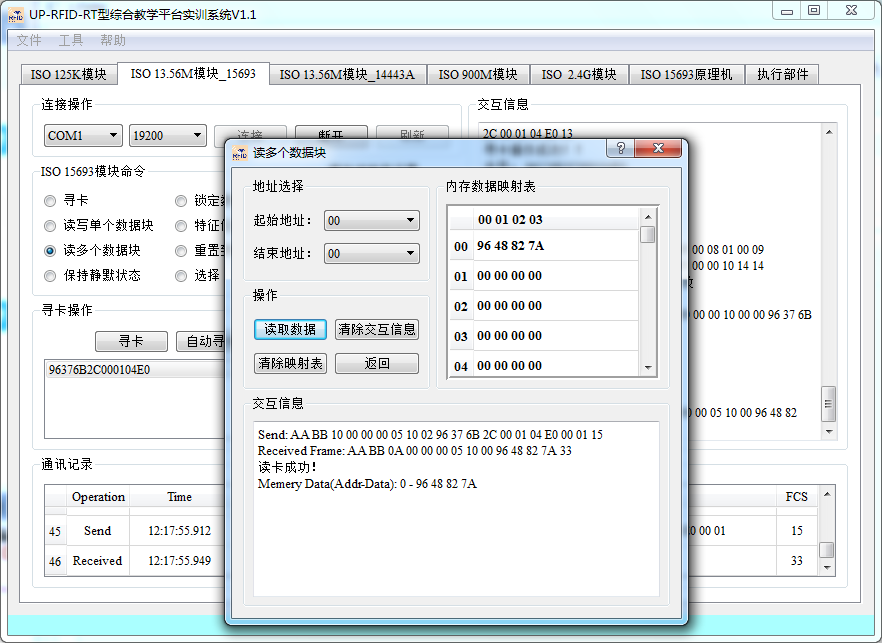


图 3 成功写入数据

1. 图书与电子标签关联管理；

获取卡号，输入信息包括书名、作者、出版社、总本书、剩余本数和可借时长，将数据写入数据库表中。

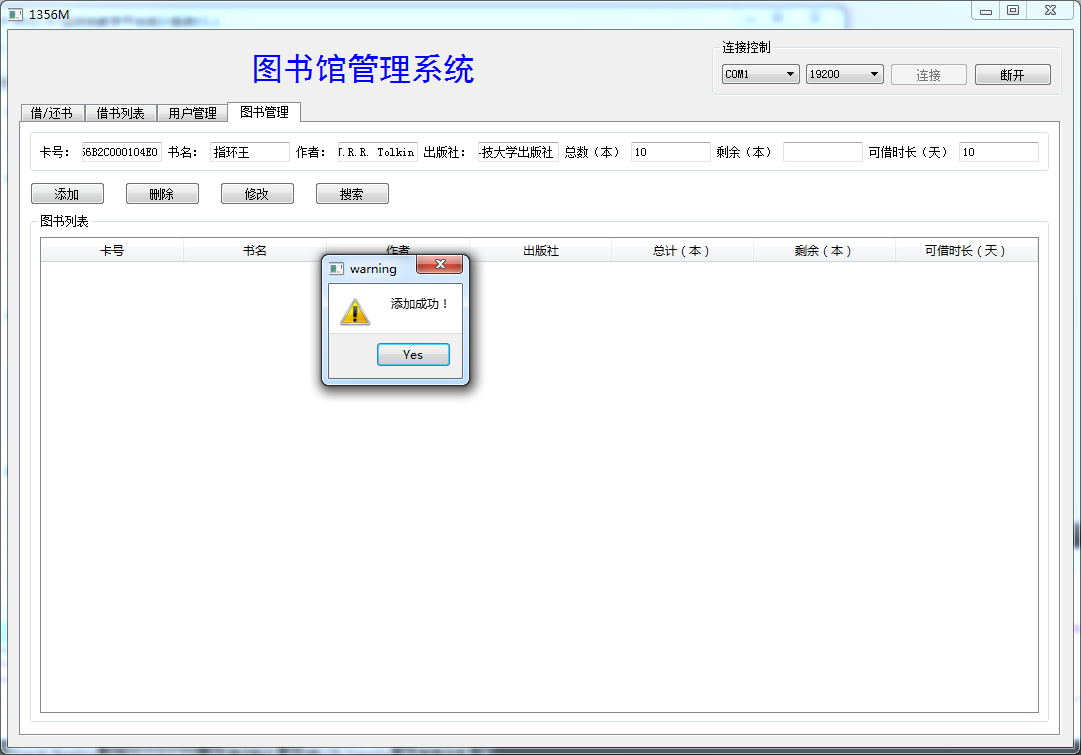


图 4 将信息与卡相关联



图 5 信息写入数据库

1. 用户毕业时的销卡管理，清除卡内借/还数据以及个人数据；

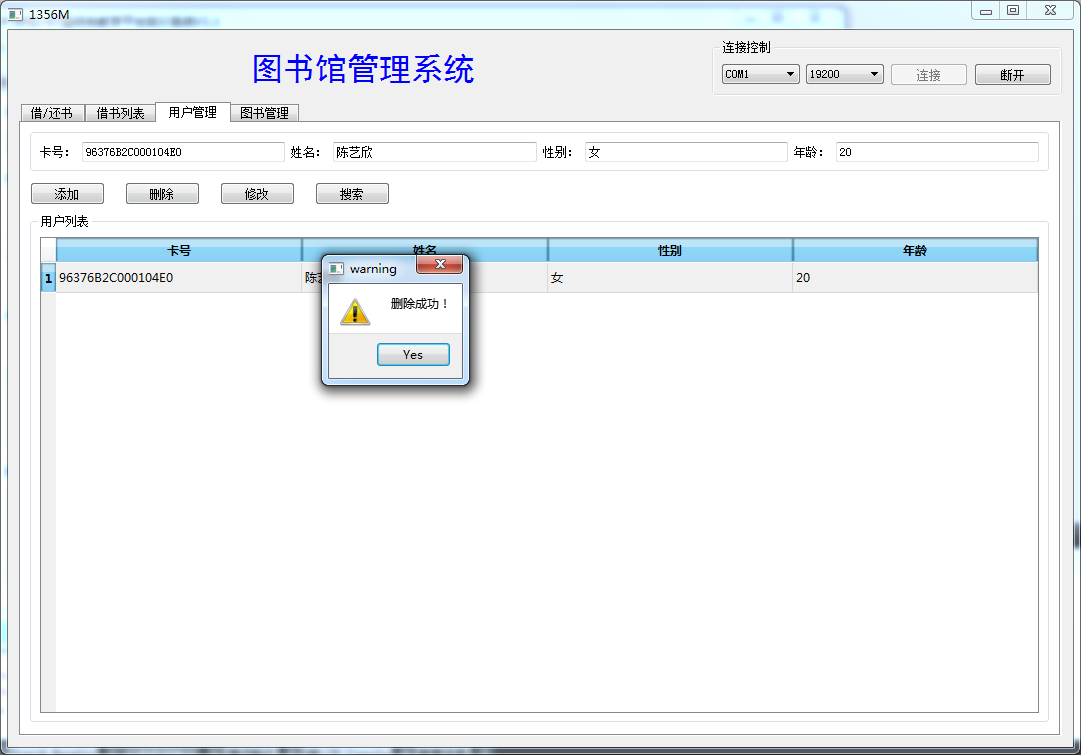


图 6 用户销卡

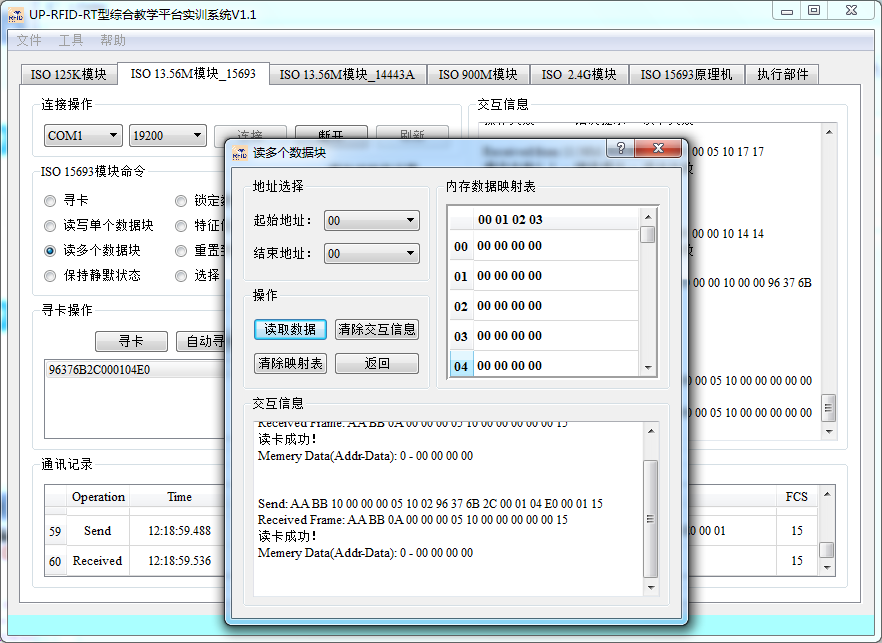


图 7 数据清零

1. 用户借或还图书时，在数据库中保存借阅记录信息，假定记录信息不超过5条；



图 8 借书并记录



图 9 借书记录

1. 若借书时有过期借阅书籍，则需先归还书籍再借阅。

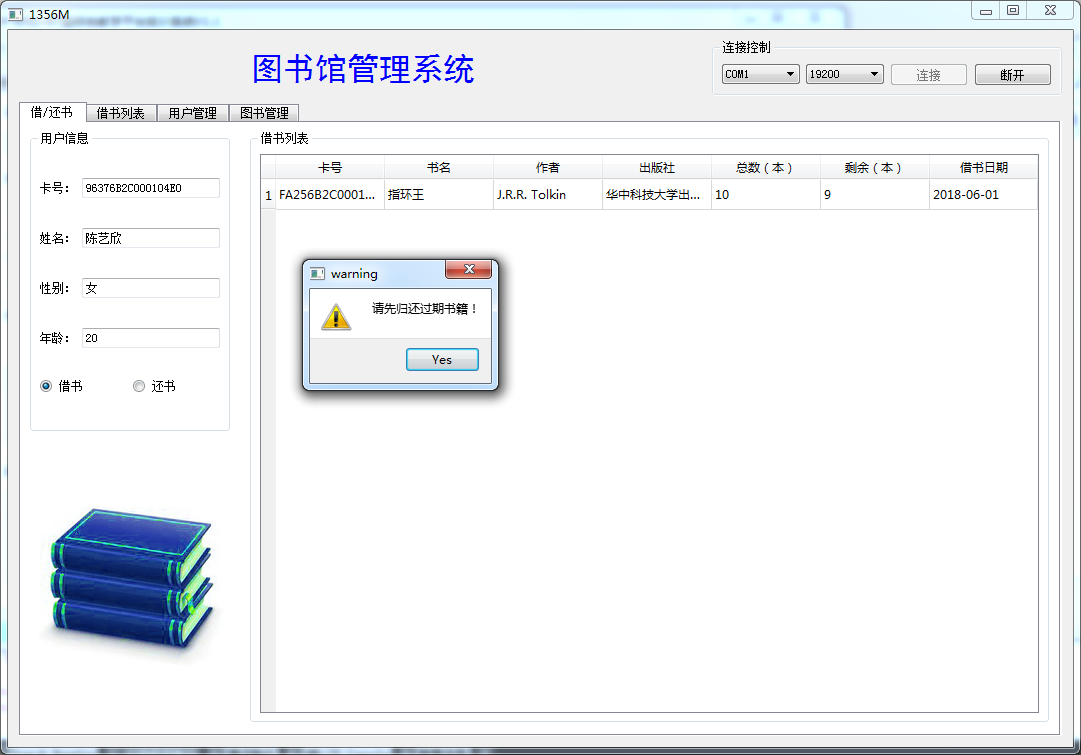


图 10 请先归还过期书籍



图 11 归还书籍后成功借阅

1.3 实验体会与总结

思考题：

1. 通过试验箱，反复循环读取十张低频电子标签。在读取过程中可能会遇到哪些问题或发生哪些现象，并分析遇到的这些问题或现象的原因；

2、在利用低频读写器模拟门禁系统中，如何获取读写器发送过来的卡号？请写出相应的函数体（含注释），并说明函数的调用方法

|  |
| --- |
| */\*\**  *\* @brief MainWindow::readData*  *\* 读取串口数据*  *\*/*  void MainWindow::readData()  {  **if**(serialPort->bytesAvailable() < 5)  **return**;  QByteArray data = serialPort->readAll();  **if**(m125dll->LF125K\_FrameAnalysis((uint8 \*)(data.data())) == 0)  {  QString tagId = CharStringtoHexString(tr(" "),data.data(),data.length());  }  }  QByteArray data = serialPort->readAll();*//读取串口发来的数据*  *//Qt有自带的封装好的函数，可以直接调用，但是读取到的数据QByteArray型，要想转化成QString，还要经过进一步调用CharStringtoHexString的函数，能使卡号显示出来。* |

总结：

通过这次实验，我大概了解了低频读写器API函数的调用方法。

1.4 核心源码说明

|  |
| --- |
| #include "mainwindow.h"  #include "ui\_mainwindow.h"  #include "recordtablemodel.h"  #include <QMessageBox>  #include <QDebug>  */\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *\*作者: jianghj@up-tech.com*  *\*日期: 2016-09-30*  *\*描述: 125K演示程序主要代码,此处模拟的人员通道,进出需要刷卡,*  *\* 125K在实际应用中主要也是这个功能,比如小区的门禁卡.*  *\* 注意:人为主动刷卡,2.4G是被动刷卡*  *\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*  MainWindow::MainWindow(QWidget \*parent) :  QMainWindow(parent),  ui(**new** Ui::MainWindow)  {  ui->setupUi(**this**);  **this**->fillPortsParameters(ui->baudRateBox);*//波特率填充*  **this**->serialPort = **new** QSerialPort(**this**);  db = **new** Database(**this**);*//连接数据库*  model = **new** RecordTableModel(**this**);  ui->tableView->setModel(model);  ui->tableView->resizeColumnsToContents();  ui->tableView->horizontalHeader()->setStretchLastSection(true);  intValidator = **new** QIntValidator(0, 4000000,**this**);  ui->btn\_connect->setEnabled(true);  ui->btn\_refresh->setEnabled(true);  ui->btn\_disconnect->setEnabled(false);  **this**->on\_btn\_refresh\_clicked();  m125dll = **new** M125Dll();  *//关联相关槽函数*  connect(ui->baudRateBox, SIGNAL(currentIndexChanged(int)),**this**, SLOT(checkCustomBaudRatePolicy(int)));  connect(serialPort, SIGNAL(error(QSerialPort::SerialPortError)), **this**, SLOT(handleError(QSerialPort::SerialPortError)));*//收到串口错误信息*  connect(serialPort, SIGNAL(readyRead()), **this**, SLOT(readData()));*//收到串口信息*  }  MainWindow::~MainWindow()  {  model->submitAll();  **delete** model;  **delete** db;  **delete** m125dll;  **delete** intValidator;  **delete** serialPort;  **delete** ui;  }  *//Baudrate parameter init*  void MainWindow::fillPortsParameters(QComboBox \*box)  {  box->clear();  box->addItem(QStringLiteral("9600"), QSerialPort::Baud9600);  box->addItem(QStringLiteral("19200"), QSerialPort::Baud19200);  box->addItem(QStringLiteral("38400"), QSerialPort::Baud38400);  box->addItem(QStringLiteral("57600"), QSerialPort::Baud57600);  box->addItem(QStringLiteral("115200"), QSerialPort::Baud115200);  box->addItem(tr("Custom"));  }  */\*\**  *\* @brief MainWindow::on\_btn\_connect\_clicked*  *\* 连接串口*  *\*/*  void MainWindow::on\_btn\_connect\_clicked()  {  QString name = ui->serialNameBox->currentText();  QString baud = ui->baudRateBox->currentText().trimmed();  **if**(baud.isEmpty())  {  QMessageBox::critical(**this**, tr("Error"), "波特率输入错误！");  **return** ;  }  serialPort->setPortName(name);  serialPort->setBaudRate(baud.toInt(),QSerialPort::AllDirections);  **if** (serialPort->open(QIODevice::ReadWrite)) {  ui->btn\_connect->setEnabled(false);  ui->btn\_disconnect->setEnabled(true);  ui->btn\_refresh->setEnabled(false);  } **else** {  ui->btn\_connect->setEnabled(true);  ui->btn\_refresh->setEnabled(true);  ui->btn\_disconnect->setEnabled(false);  QMessageBox::warning(**this**,tr("提示"),tr("初始化%1失败！请检查串口是否已经被占用？").arg(name),QMessageBox::Yes);  }  }  */\*\**  *\* @brief MainWindow::on\_btn\_disconnect\_clicked*  *\* 断开连接*  *\*/*  void MainWindow::on\_btn\_disconnect\_clicked()  {  **if**(!serialPort->isOpen())  **return** ;  serialPort->close();  ui->btn\_connect->setEnabled(true);  ui->btn\_refresh->setEnabled(true);  ui->btn\_disconnect->setEnabled(false);  }  */\*\**  *\* @brief MainWindow::on\_btn\_refresh\_clicked*  *\* 刷新按钮点击事件*  *\*/*  void MainWindow::on\_btn\_refresh\_clicked()  {  QStringList list = getSerialName();  ui->serialNameBox->clear();  ui->serialNameBox->addItems(list);  }  */\*\**  *\* @brief MainWindow::checkCustomBaudRatePolicy*  *\* @param idx combox被选中的索引值*  *\* 设置自定义波特率*  *\*/*  void MainWindow::checkCustomBaudRatePolicy(int idx)  {  QComboBox \*box = **dynamic\_cast**<QComboBox\*>(QObject::sender());  bool isCustomBaudRate = !box->itemData(idx).isValid();  box->setEditable(isCustomBaudRate);  **if** (isCustomBaudRate) {  box->clearEditText();  box->setValidator(intValidator);  }  }  */\*\**  *\* @brief MainWindow::readData*  *\* 读取串口数据*  *\*/*  void MainWindow::readData()  {  **if**(serialPort->bytesAvailable() < 5)  **return**;  QByteArray data = serialPort->readAll();  **if**(m125dll->LF125K\_FrameAnalysis((uint8 \*)(data.data())) == 0)  {  QString tagId = CharStringtoHexString(tr(" "),data.data(),data.length());*//获取标签ID*  QString time = CurrentDateTime();*//获取时间*  int index = model->findRecord(tagId);*//查询此标签记录*  **if**(index >= 0 )  {  QString text = model->record(index).value(2).toString();  **if**(text == tr("进"))  model->updateRecord(index,tagId,time,tr("出"));  **else**  model->updateRecord(index,tagId,time,tr("进"));  }  **else** {  model->addRecord(tagId,time,tr("进"));  }  }  }  */\*\**  *\* @brief MainWindow::handleError*  *\* @param error SerialPortError枚举类,详细请看SerialPortError的定义*  *\* 处理错误信息*  *\*/*  void MainWindow::handleError(QSerialPort::SerialPortError error)  {  **if** (error == QSerialPort::ResourceError) {  QMessageBox::critical(**this**, tr("Critical Error"), serialPort->errorString());  **this**->on\_btn\_disconnect\_clicked();  }  } |