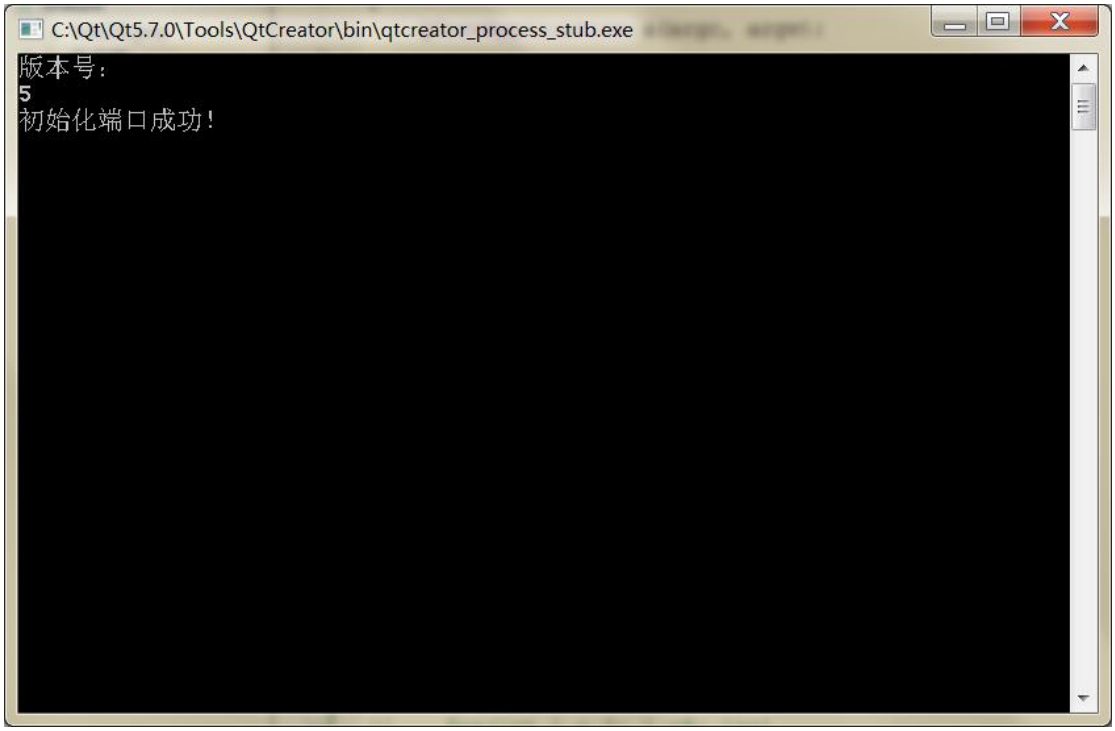
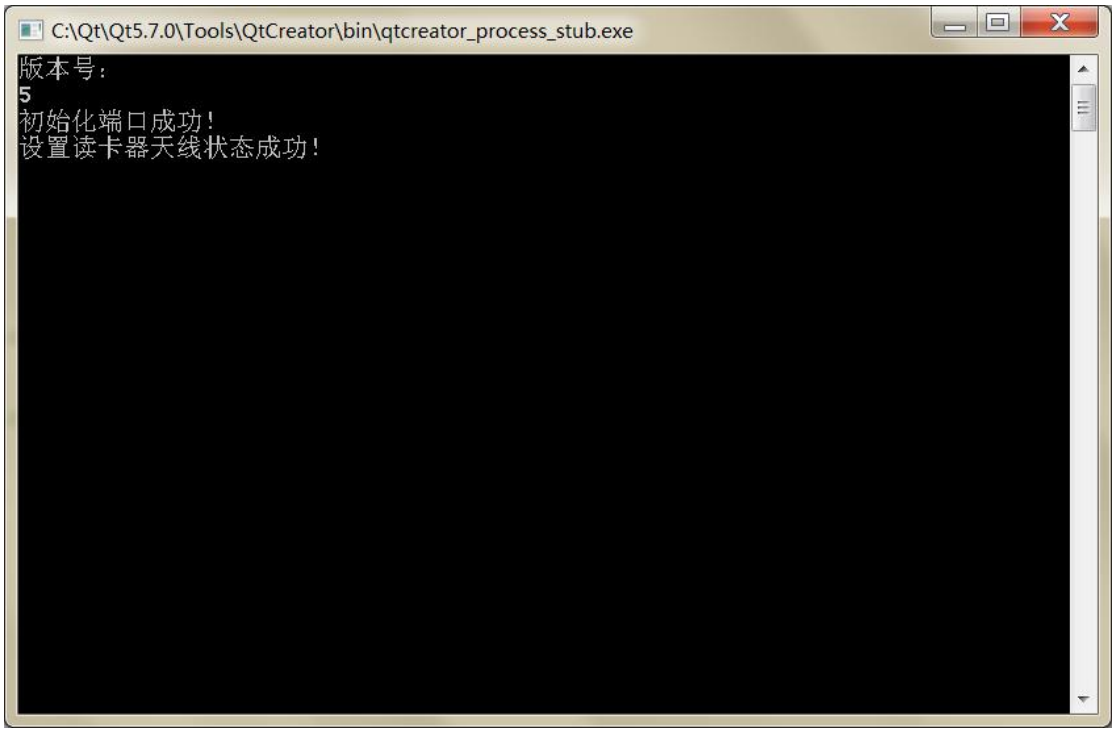
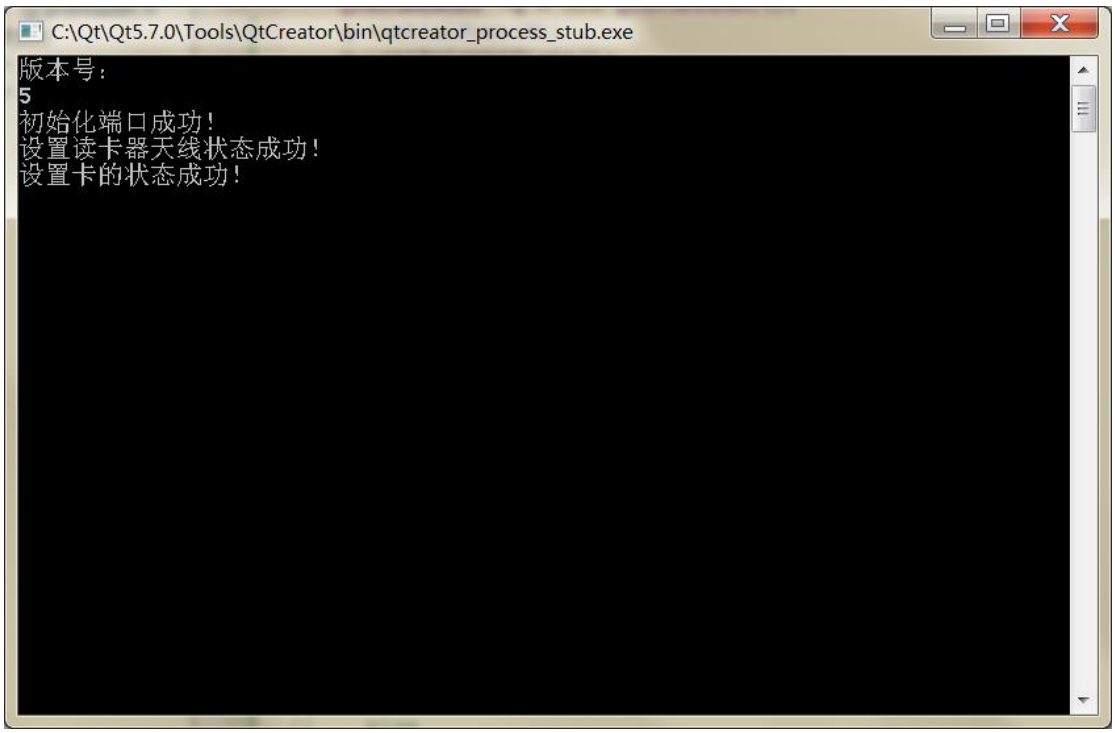
课程项目编程

已完成部分：

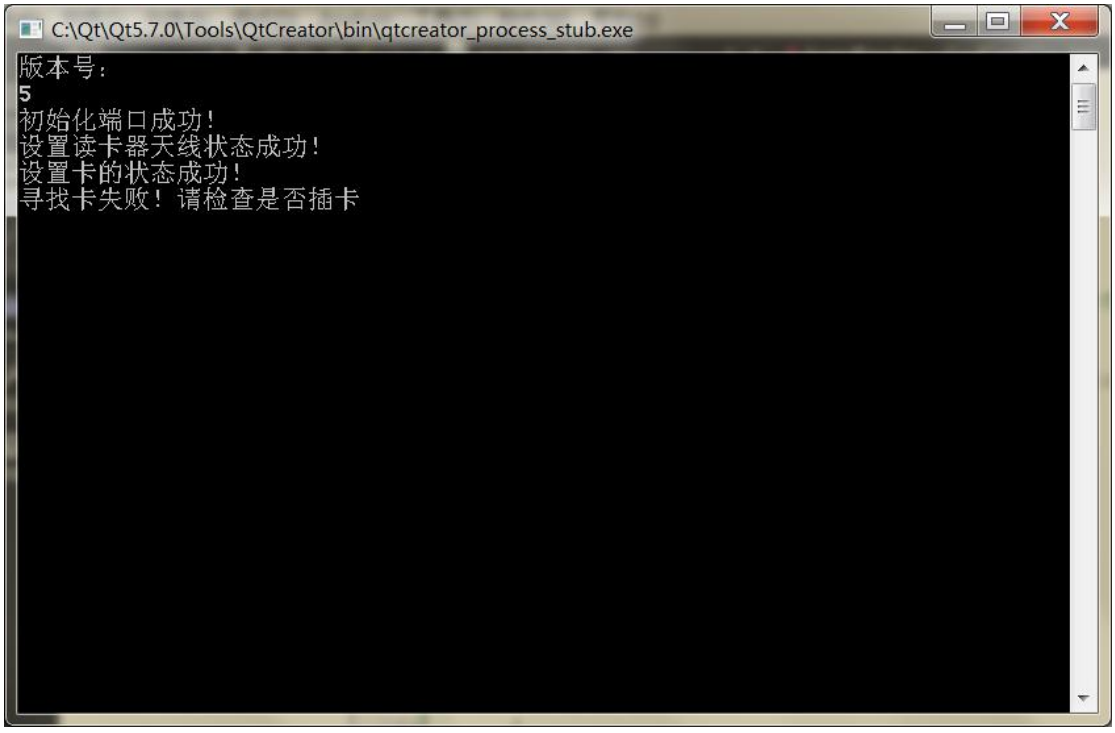
读卡器部分截图：



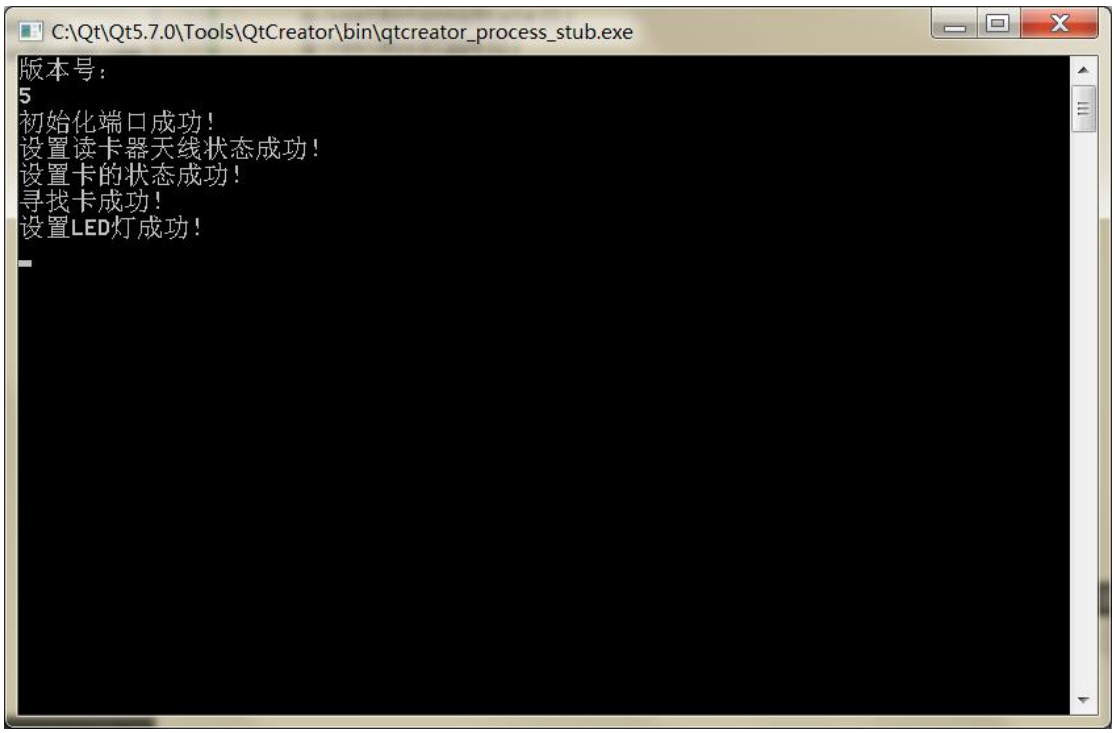


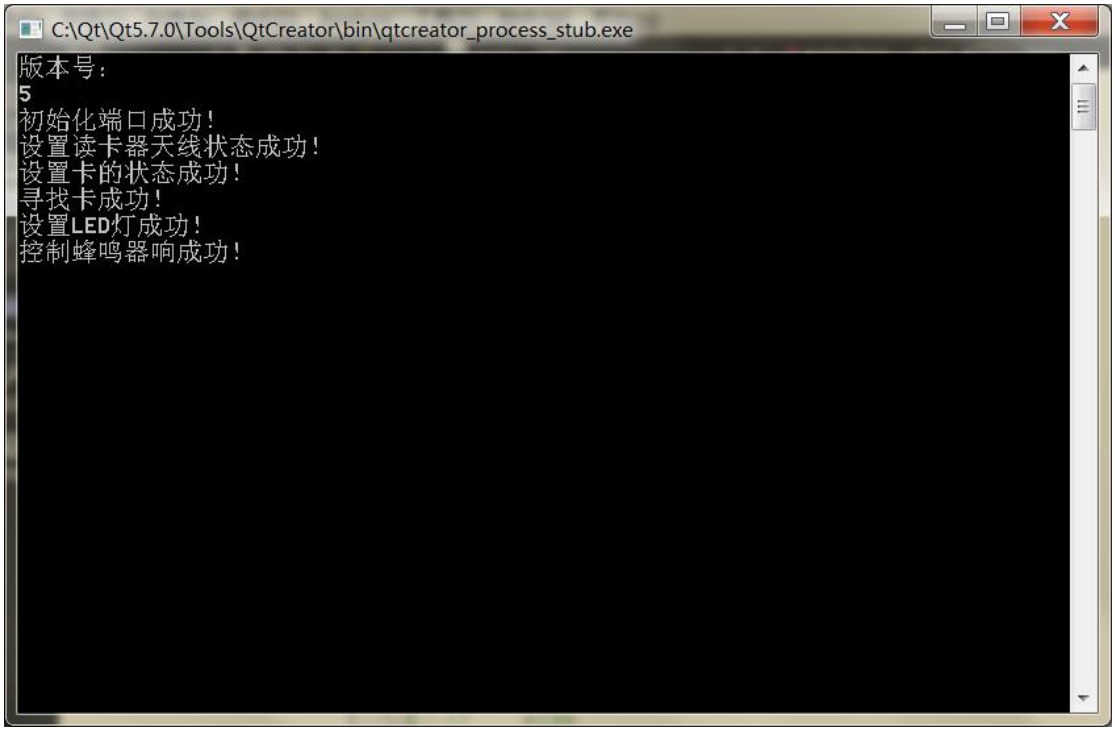

C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!

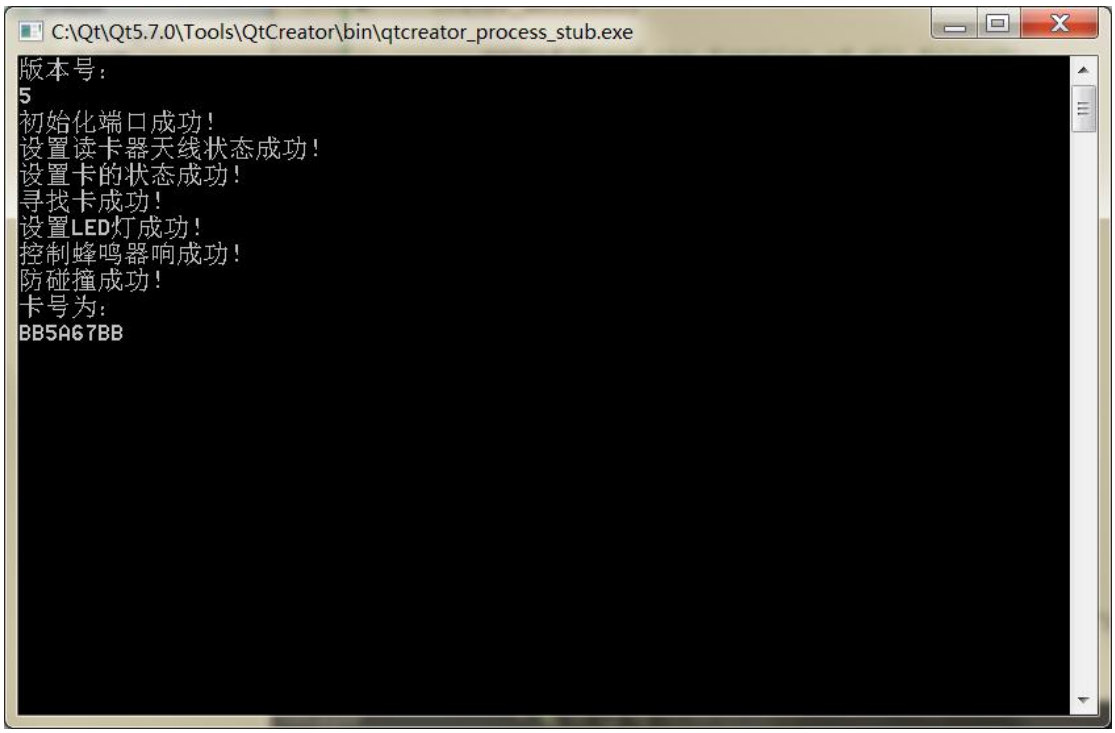
未 插 卡 时 :

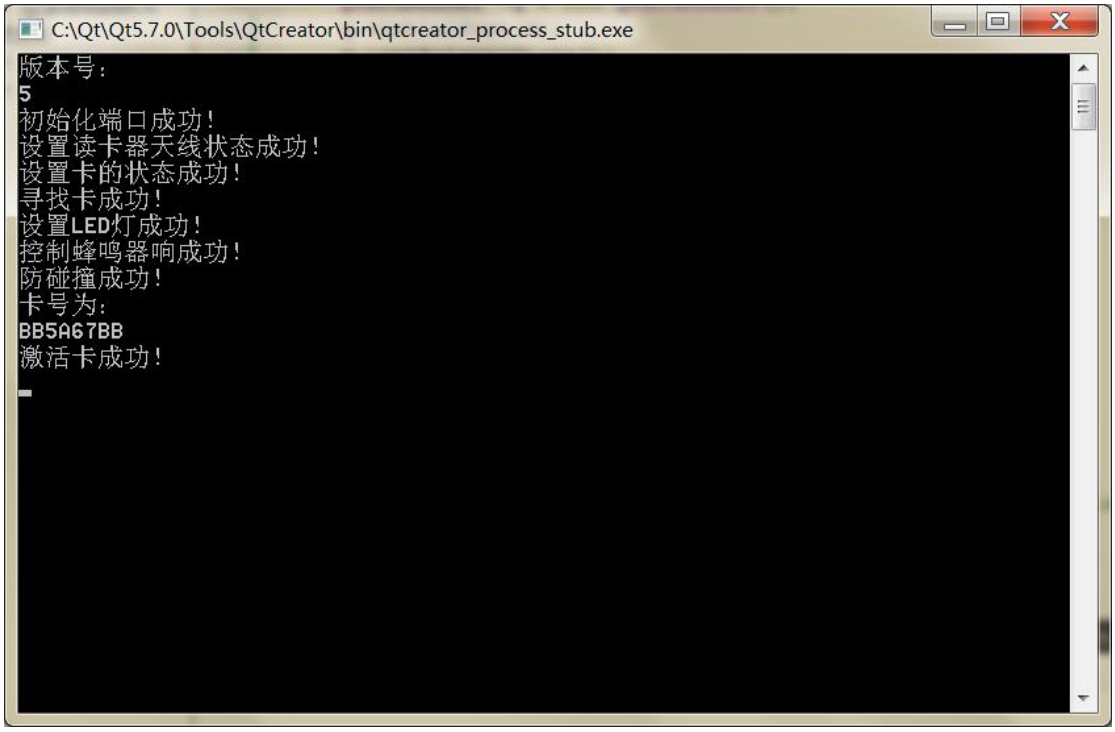

C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡失败! 请检查是否插卡

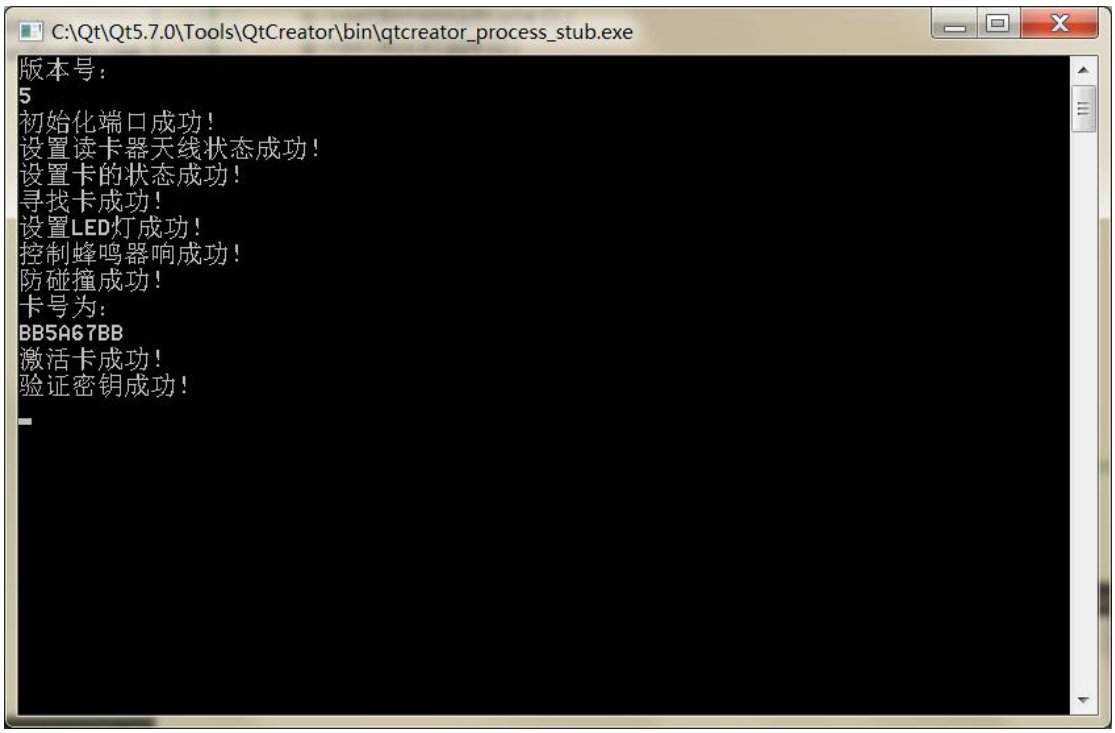
插 卡 后 :

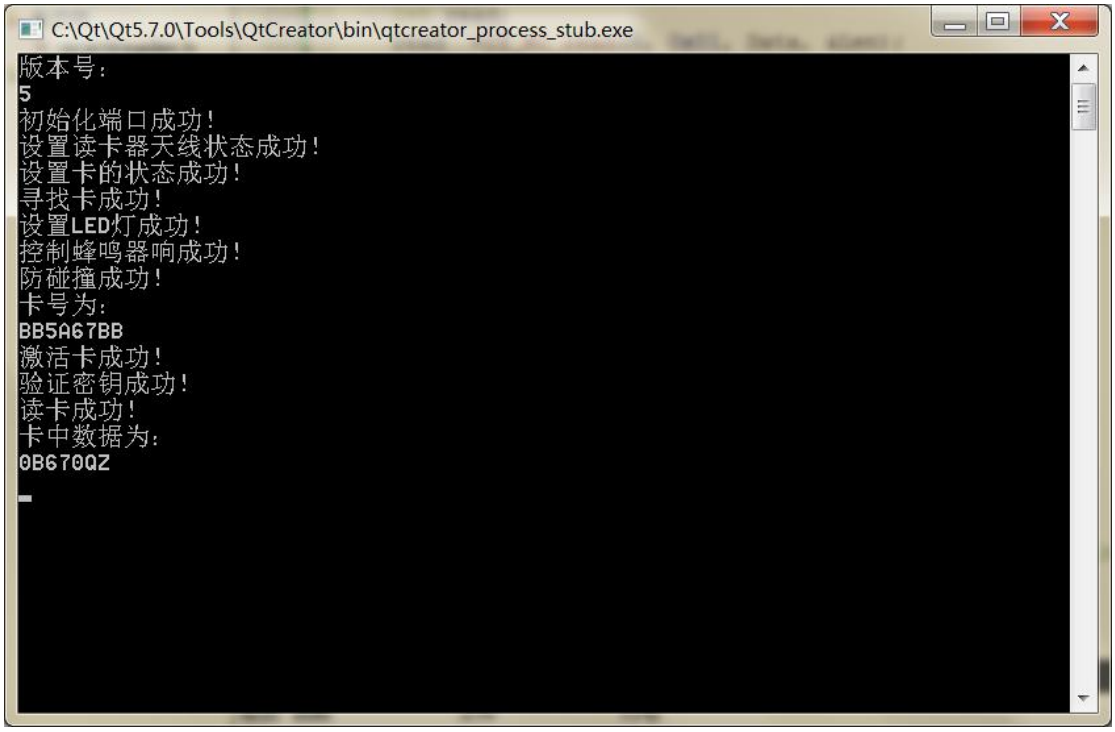

C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡成功!
设置LED灯成功!

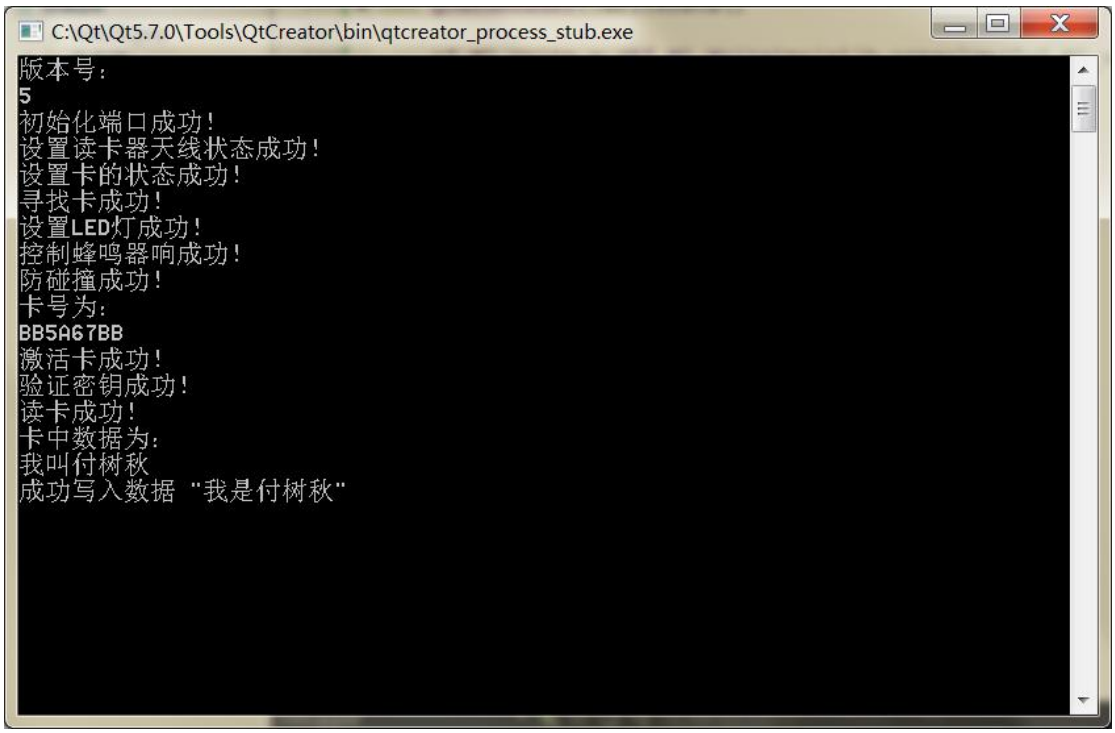

C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡成功!
设置LED灯成功!
控制蜂鸣器响成功!


C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡成功!
设置LED灯成功!
控制蜂鸣器响成功!
防碰撞成功!
卡号为:
BB5A67BB

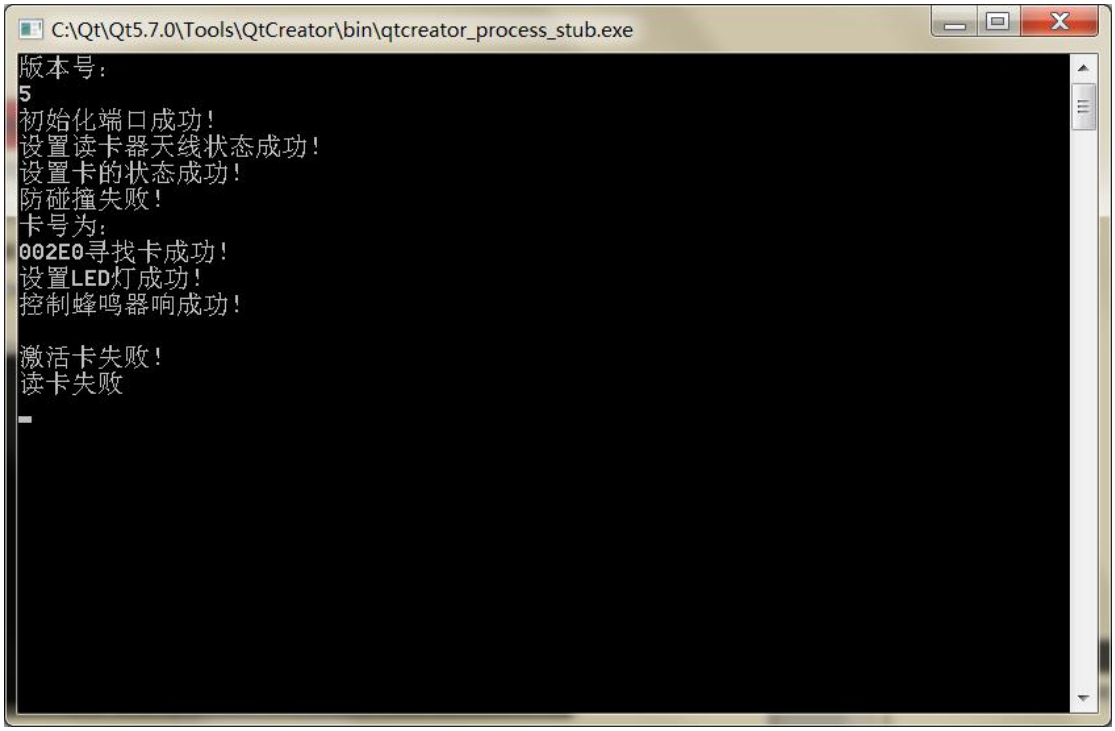

C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡成功!
设置LED灯成功!
控制蜂鸣器响成功!
防碰撞成功!
卡号为:
BB5A67BB
激活卡成功!


C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡成功!
设置LED灯成功!
控制蜂鸣器响成功!
防碰撞成功!
卡号为:
BB5A67BB
激活卡成功!
验证密钥成功!


C:\Qt\Qt5.7.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
版本号:
5
初始化端口成功!
设置读卡器天线状态成功!
设置卡的状态成功!
寻找卡成功!
设置LED灯成功!
控制蜂鸣器响成功!
防碰撞成功!
卡号为:
BB5A67BB
激活卡成功!
验证密钥成功!
读卡成功!
卡中数据为:
0B670QZ



若不按工作流程读卡：



自定义函数

```
public:
    //自定义函数
    //获取动态库的版本号
    int currentLibraryVersion();
    //初始化端口
    int intialPort(int port);
    //关闭串口
    int closePort();
    //设置读卡器天线状态
    int setAntennaState();
    //设置LED指示灯
    int setLED(unsigned char col);
    //读取读写卡器型号及产品型号
    int getModel();
    //控制蜂鸣器响
    int controlBeep(unsigned char time);
    //寻找卡
    int findCard();
    //防冲撞
    int anticoll(unsigned char pSnr[], unsigned char &pLen);
    //激活卡
    int selectCard();
    //设置工作状态
    int setWorkType();
    //验证密钥
    int authentication(unsigned char block);
    //读取数据
    int readData(unsigned char Data[], unsigned char &Len);
    //写入数据
    int writeData();
```

声明动态链接库中函数

```
private:
    //从动态链接库里面加载的函数
    //获取动态库的版本号
    typedef int(*Lib_Ver)(unsigned int *pVer);
    Lib_Ver lib_ver;
    //初始化端口
    typedef int(*Rf_Init_Com)(unsigned short icdev,int port,long baud);
    Rf_Init_Com rf_init_com;
    //关闭串口
    typedef int (*Rf_ClosePort)();
    Rf_ClosePort rf_ClosePort;
    //设置读卡器天线状态
    typedef int (*Rf_Antenna_Sta)(unsigned short icdev, unsigned char model);
    Rf_Antenna_Sta rf_antenna_sta;
    //设置读卡器非接触工作方式
    typedef int (*Rf_Init_Type) (unsigned short icdev, unsigned char type);
    Rf_Init_Type rf_init_type;
    //设置LED指示灯颜色
    typedef int (*Rf_Light) (unsigned short icdev, unsigned char color);
    Rf_Light rf_light;
    //控制蜂鸣器响
    typedef int (*Rf_Beep) (unsigned short icdev, unsigned char msec);
    Rf_Beep rf_beep;
    //寻TYPE_A卡
    typedef int (*Rf_Rrequest) (unsigned short icdev, unsigned char model,  unsigned short
                                *pTagType);
    Rf_Rrequest rf_request;
    //防冲撞
    typedef int (*Rf_Anticoll)(unsigned short icdev,unsigned char bcnt,
                               unsigned char *pSnr, unsigned char *pLen);
    Rf_Anticoll rf_anticoll;
    //激活卡
    typedef int (*Rf_Select) (unsigned short icdev, unsigned char *pSnr, unsigned char snrLen,unsigned char *pSize);
    Rf_Select rf_select;
```

```
    //验证密钥
    typedef int (*Rf_M1_Authentication2)(unsigned short icdev, unsigned char model,
                                         unsigned char block, unsigned char *pKey);
    Rf_M1_Authentication2 rf_M1_authentication2;
    //从卡片读取数据
    typedef int (*Rf_M1_Read) (unsigned short icdev, unsigned char block, unsigned char *pData,unsigned char *pLen);
    Rf_M1_Read rf_M1_read;
    //向卡片写入数据
    typedef int (*Rf_M1_Write) (unsigned short icdev, unsigned char block, unsigned char *pData);
    Rf_M1_Write rf_M1_write;
protected:
    //加载动态库
    void loadDLL();
private :
    QLibrary *mainLib;
public:

    unsigned char pSnr[10];//卡的序列号
    unsigned char pLen;//卡序列号长度
     char pData[20];//读取的数据
    unsigned char dataLen;//数据的长度

```

调用函数

```
125 //寻找卡
126 int QcardReader::findCard()
127 {
128     rf_request = (Rf_Rrequest)mainLib->resolve("rf_request");
129     if(!rf_request)
130     {
131         printf("load the function of dll falled");
132         exit(1);
133     }
134     int find;
135
136     unsigned short type;
137
138     find = rf_request(0, 0x52, &type);
139     if(find == 0)
140     {
141         qDebug()<<"寻找卡成功！";
142     }
143     else
144     {
145         qDebug()<<"寻找卡失败！请检查是否插卡";
146     }
147     //    printf("The type is:\n");
148     //    printf("%d\n",type);
149     return find;
150 }
```

```
152 //防冲撞
153 int QcardReader::anticoll(unsigned char pSnr[], unsigned char &pLen)
154 {
155     rf_anticoll = (Rf_Anticoll)mainLib->resolve("rf_anticoll");
156     if(!rf_anticoll)
157     {
158         printf("load the function of dll falled");
159         exit(1);
160     }
161     int ant = rf_anticoll(0, 4, pSnr, &pLen);
162     if(ant == 0)
163     {
164         qDebug()<<"防碰撞成功！";
165     }
166     else
167         qDebug()<<"防碰撞失败！";
168
169     qDebug()<<"卡号为：";
170     for(int i = 0; i <4; i++)
171         printf("%X", pSnr[i]);
172
173     return ant;
174 }
```

```cpp
//激活卡
int QcardReader::selectCard()
{
    rf_select = (Rf_Select)mainLib->resolve("rf_select");
    if(!rf_select)
    {
        printf("load the function of dll falled");
        exit(1);
    }
    unsigned char Size=0;//返回卡的容量
    int select;//测试是否激活成功
    select = rf_select(0, pSnr, pLen, &Size);
    if(select == 0)
        qDebug()<<"激活卡成功！";
    else
        qDebug()<<"激活卡失败！";
    //    printf("select successfully?");
    //    printf("%d\n", select);
    //    printf("The size is:");
    //    printf("%x\n", Size);
    return select;
}
```

```cpp
//验证密钥
int QcardReader::authentication(unsigned char block)
{
    Rf_M1_Authentication2 rf_M1_authentication2 = (Rf_M1_Authentication2)mainLib->resolve("rf_M1_authentication2");
    if(!rf_M1_authentication2)
    {
        printf("load the function of dll failed");
        exit(1);
    }
    int authenticate;

    unsigned char key[6];
    memset(key, 0xff, sizeof(key));
    authenticate = rf_M1_authentication2(0, 0x60, block, key);

    if(authenticate == 0)
        qDebug()<<"验证密钥成功！";
    return authenticate;
}
```

```cpp
//读取数据
int QcardReader::readData(unsigned char Data[], unsigned char &Len)
{
    rf_M1_read = (Rf_M1_Read)mainLib->resolve("rf_M1_read");
    if(!rf_M1_read)
    {
        printf("load the function of dll fail");
        exit(1);
    }
    int read;
    read  =rf_M1_read(0, 0x01, Data, &Len);
    if(read != 0)
    {
        qDebug()<<"读卡失败";
    }
    else
    {
        qDebug()<<"读卡成功！";
        qDebug()<<"卡中数据为：";
//        QString str = QString::fromLocal8Bit(pData);
        printf("%s\n",pData);
//        qDebug()<<str;
    }
    return read;
}
```

```
//写入数据
int QcardReader::writeData()
{
    rf_M1_write = (Rf_M1_Write)mainLib->resolve("rf_M1_write");
    if(!rf_M1_write)
    {
        printf("load the function of dll fail");
        exit(1);
    }
    int write;
    //    QString string = QStringLiteral("666777");
    QString string = "我是付树秋";
    QByteArray ba;//定义字节数组

    char *ch;
    ba = string.toLocal8Bit();
    ch = ba.data();
    write = rf_M1_write(0, 0x01, (unsigned char *)ch);
    if(write == 0)
        qDebug()<<"成功写入数据"<<string;
}
```

```
//加载动态库
void QcardReader::loadDLL()
{
    //    qDebug()<<QDir::currentPath();

    //加载读卡器动态链接库
    mainLib = new QLibrary("MasterRDnew.dll");
    if(!mainLib->load())
    {
        printf("load MasterRDnew.dll false");
        //没有动态链接库退出程序
        exit(1);
    }
    //验证动态链接库函数是否读取成功
    qDebug()<<"版本号：";
    printf("%d\n", currentLibraryVersion());
}
```

前端部分:

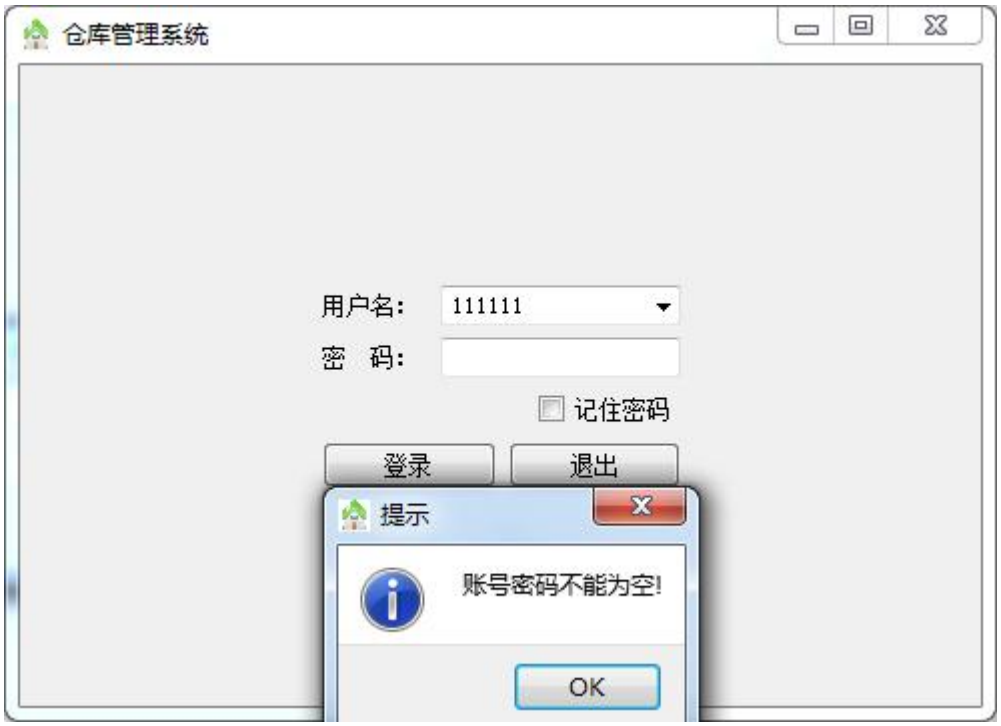
前端（部分界面、代码、输出）：

登录：


```
//验证账号
void Login::on_LoginPushButton_clicked()
{

    getUserNameAndPassFromWidget();

    if(checkInputIsEmpty())
    {
        errorMessage("账号密码不能为空!");
        return;
    }

    QJsonObject json;
    json.insert("MessageType", "Login");
    json.insert("UserNo", userName);
    json.insert("UserPass", userPass);

    QJsonDocument document;
    document.setObject(json);
    QByteArray byteArrayFromJson =
document.toJson(QJsonDocument::Compact);

    qDebug() << "登录";
    qDebug() << byteArrayFromJson;
```

```
    service->sendMessage(byteArrayFromJson);
}
```

验证账号密码的正确：




```
//检验账号密码正确
bool Login::checkInputIsEmpty()
{
    return userName.isEmpty() || userPass.isEmpty();
}
```

测试输出

登录

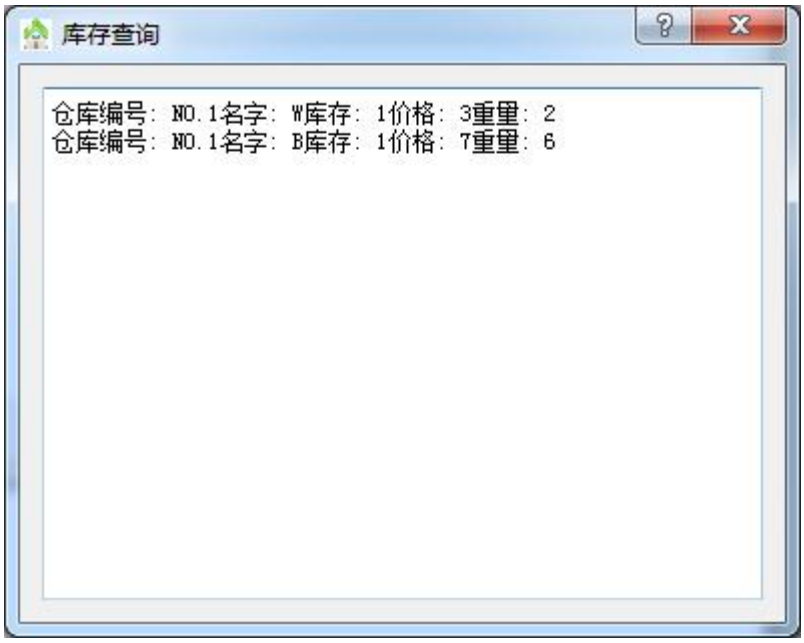
```
"{\"MessageType\":\"Login\",\"UserNo\":\"1111111\",\"UserPass\":\"0\"}
"
```

发送消息：

```
"{\"MessageType\":\"Login\",\"UserNo\":\"1111111\",\"UserPass\":\"0\"}
\r\n"
```

收消息：

```
"{\"Result\":\"false\",\"MessageType\":\"Login\"}\r\n"
```




```
//库存查询
void QueryResult::obligationResult()
{
    setWindowTitle("库存查询");
    if(json.contains("Quantity"))
    {
        int count = json["Quantity"].toInt();
        for(int i = 0; i < count; ++i)
        {
            QString str("");

            if(json.contains(QString("Commint" + QString::number(i))))
            {
                qDebug() << QString("Commint" + QString::number(i));

                QJsonObject obj = json[QString("Commint" +
QString::number(i))].toObject();
                if(obj.contains("depotNo"))
                    str += QString("仓库编号: ") +
obj["depotNo"].toString();

                if(obj.contains("goodsName"))
                    str += QString("名字: ") +
obj["goodsName"].toString();

                if(obj.contains("goodsSum"))
```

```
                    str += QString("库存: ") +
QString::number(obj["goodsSum"].toInt());

                if(obj.contains("goodsPrice"))
                    str += QString("价格: ") +
QString::number(obj["goodsPrice"].toInt());

                if(obj.contains("goodsWeight"))
                    str += QString("重量: ") +
QString::number(obj["goodsWeight"].toInt());

                ui->textEdit->append(str);
            }
        }
    }
}
```

测试输出：

发送消息：

"{\"MessageType\":\"Query\",\"QueryType\":\"Obligation\"}\r\n"

收消息：

"{\"QueryType\":\"Obligation\",\"Quantity\":2,\"Commint1\":{\"depotNo\":\"NO.1\",\"goodsSum\":1,\"goodsPrice\":7,\"goodsWeight\":6,\"goodsNo\":\"0\",\"goodsName\":\"B\"},\"MessageType\":\"Query\",\"Commint0\":{\"depotNo\":\"NO.1\",\"goodsSum\":1,\"goodsPrice\":3,\"goodsWeight\":2,\"goodsNo\":\"1\",\"goodsName\":\"W\"}}\r\n"

出库：


```
//出库
void RfidMainWindow::on_outPushButton_clicked(bool checked)
{
    if(checked)
    {
        currentWorkType = OUT_OF_The_LIBRARY;
        ui->outPushButton->setText("停止出库");
        ui->enterPushButton->setDisabled(true);
        ui->textEdit->setText(QString("开始出库时间: ") +
QTime::currentTime().toString());

        currentRecInfo.clear();

        jsonObject = new QJsonObject();

        jsonObject->insert("MessageType", "OutGoods");
        jsonObject->insert("StartTime",
QTime::currentTime().toString());
        jsonObject->insert("OperatorName", operatorName);
        //批次号
        jsonObject->insert("BatchNumber", getRandString());
    }
    else
```

```
    {
        currentWorkType = NO_WORK_TYPE;
        ui->outPushButton->setText("出  库");
        ui->enterPushButton->setDisabled(false);
        ui->textEdit->append(QString("停止出库时间: ") +
QTime::currentTime().toString());

        jsonObject->insert("EndTime",
QTime::currentTime().toString());

        toJson();

        QJsonDocument document;
        document.setObject(*jsonObject);
        QByteArray byteArrayFromJson =
document.toJson(QJsonDocument::Compact);

        qDebug() << byteArrayFromJson;

        emit sendMessage(byteArrayFromJson);

        qDebug() << jsonObject;
        delete jsonObject;
        jsonObject = 0;
    }

}
```

//测试输出

发送消息：

"{\"BatchNumber\":\"201612iwnCqNbK\",\"Commodity0\":{\"Count\":1,\"ID\":\"1\",\"Name\":\"W\",\"Price\":3,\"SupplierID\":\"1\",\"SupplierName\":\"KG\",\"Weiget\":2},\"EndTime\":\"21:36:53\",\"MessageType\":\"OutGoods\",\"OperatorName\":\"111111\",\"Quantity\":1,\"StartTime\":\"21:36:48\"}\r\n"

收消息：

"{\"Result\":\"true\",\"MessageType\":\"OutGoods\"}\r\n"

入库：


```
//入库
void RfidMainWindow::on_enterPushButton_clicked(bool checked)
{
    if(checked)
    {
        currentWorkType = IN_OF_The_LIBRARY;
        ui->enterPushButton->setText("停止入库");
        ui->outPushButton->setDisabled(true);
        ui->textEdit->clear();
        ui->textEdit->setText(QString("开始入库时间: ") +
QTime::currentTime().toString());

        currentRecInfo.clear();

        jsonObject = new QJsonObject();

        jsonObject->insert("MessageType", "InGoods");
        jsonObject->insert("StartTime",
QTime::currentTime().toString());
        jsonObject->insert("OperatorName", operatorName);
        //批次号
        jsonObject->insert("BatchNumber", getRandString());
    }
```

```
    else
    {
        currentWorkType = NO_WORK_TYPE;
        ui->enterPushButton->setText("入  库");
        ui->outPushButton->setDisabled(false);
        ui->textEdit->append(QString("停止入库时间: ") +
QTime::currentTime().toString());

        jsonObject->insert("EndTime",
QTime::currentTime().toString());

        toJson();

        QJsonDocument document;
        document.setObject(*jsonObject);
        QByteArray byteArrayFromJson =
document.toJson(QJsonDocument::Compact);

        emit sendMessage(byteArrayFromJson);

        delete jsonObject;
        jsonObject = 0;
    }
}
```

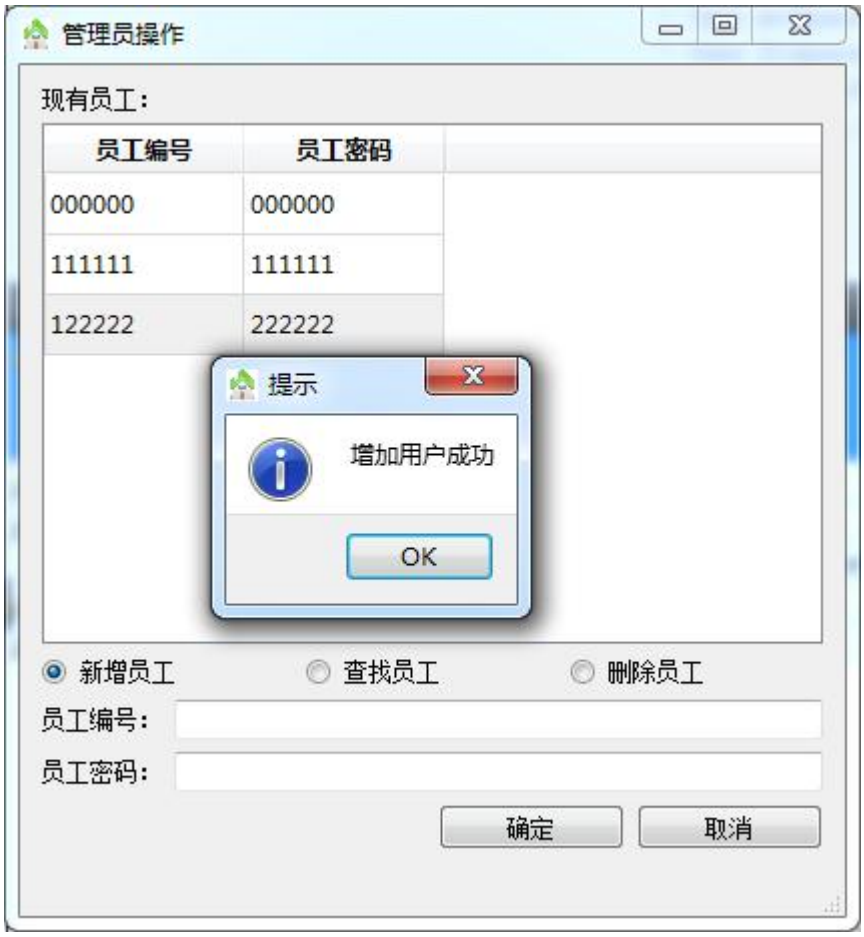
//测试输出

发送消息：

"{\"BatchNumber\":\"201612yzKTlyKr\",\"Commodity0\":{\"Count\":1,\"ID\":\"1\",\"Name\":\"W\",\"Price\":3,\"SupplierID\":\"1\",\"SupplierName\":\"KG\",\"Weiget\":2},\"EndTime\":\"21:37:24\",\"MessageType\":\"InGoods\",\"OperatorName\":\"111111\",\"Quantity\":1,\"StartTime\":\"21:37:15\"}\r\n"

收消息：

"{\"Result\":\"true\",\"MessageType\":\"InGoods\"}\r\n"



//增加员工

```
        int cols = ui->EmployeeInformation->columnCount();
        int rows = ui->EmployeeInformation->rowCount();
        qDebug()<<rows;
        ui->EmployeeInformation->insertRow(rows);
        for(int i = 0;i < cols;i++)
        {
            ui->EmployeeInformation->setItem(rows,0,new
QTableWidgetItem(ui->NumText->text()));
            ui->EmployeeInformation->setItem(rows,1,new
QTableWidgetItem(ui->PassText->text()));
        }
        ui->EmployeeInformation->selectRow(rows);
```

//删除用户

```
int rowIndex= ui->EmployeeInformation->currentRow();
QMessageBox msg;
msg.setText(QString::number(rowIndex));
msg.exec();
qDebug()<< rowIndex;
if(rowIndex != -1 )
    ui->EmployeeInformation->removeRow(rowIndex);
```

后台部分：

启动类：

```java
package cn.com;

import cn.com.Socket.Server;


public class Start {
    public static void main(String[] args) {

        Server s = new Server();
    }

}

```

数据库连接：

```java

import java.sql.Connection;

public abstract class DBUtil {
    private DBUtil() {

    }
    static {

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            System.out.println("数据库未连接");
        }
    }

    public static Connection getConn() {
        Connection conn = null;
        try {
            conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl","scott","tiger");
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            System.out.println("数据库未连接");
        }
        return conn;
    }

    public static void free(ResultSet rs,Statement pstm,Connection conn) {
        if(rs != null) {
            try {
                rs.close();
```

接收与发端口：

```java
package cn.com.Socket;

import java.io.BufferedReader;

public class Server {
    public Server(){
        init();
    }

    /**
     * @param args
     */
    public void init() {
        // TODO Auto-generated method stub
        ServerSocket ss = null;
        Socket s = null;
        BufferedReader br = null;
        PrintWriter pw = null;

        try {
            ss = new ServerSocket(50000);
            System.out.println("正在监听50000端口");
            s = ss.accept();
            String ip = s.getInetAddress().getHostAddress();
            System.out.println(ip + "已建立连接");
            br = new BufferedReader(new InputStreamReader(s.getInputStream(),"utf-8"));
            pw = new PrintWriter(s.getOutputStream(),true);

            while(true){
                String msgFromClient = br.readLine();
                System.out.println(msgFromClient);
                JSONObject  jsonObj  = JSONObject.fromObject(msgFromClient);
                String msgToClient = null;
```

库存查询:

```java
package cn.com.daos;

import java.sql.Connection;

public class GoodsInfoDAOImpl implements GoodsInfoDAOInf{

    @Override
    public List<GoodsInfoBean> getAllGoodsInfo() {
        // TODO Auto-generated method stub
        GoodsInfoBean g = null;
        List<GoodsInfoBean> list = new ArrayList<GoodsInfoBean>();
        Connection conn = DBUtil.getConn();
        PreparedStatement pstm = null;
        ResultSet rs = null;
        String sql = "select * from goods_inf";
        try {
            pstm = conn.prepareStatement(sql);
            rs = pstm.executeQuery();
            while(rs.next()) {
                g = new GoodsInfoBean();
                g.setGoodsNo(rs.getString("goods_no"));
                g.setGoodsName(rs.getString("goods_name"));
                g.setGoodsSum(rs.getInt("goods_sum"));
                g.setGoodsPrice(rs.getInt("goods_price"));
                g.setGoodsWeight(rs.getInt("goods_weight"));


                list.add(g);
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
```

数据添加：

```java
package cn.com.severs;

import java.util.ArrayList;


public class AddSever {
    public String tableAdd(JSONObject  jsonobjFromClient,JSONObject json){
        InOrOutInfoDAOImpl dao = new InOrOutInfoDAOImpl();
        GoodsInfoDAOImpl daog = new GoodsInfoDAOImpl();
        SupplierInfoDAOImpl daos = new SupplierInfoDAOImpl();
        String jsonString = null;
        if(dao.validateByInOrOutNo(jsonobjFromClient.getString("BatchNumber")) == false){
            InOrOutInfoBean iob = new InOrOutInfoBean();
            iob.setInOrOutNo(jsonobjFromClient.getString("BatchNumber"));
            iob.setInOrOutType(jsonobjFromClient.getString("MessageType"));
            iob.setUsername(jsonobjFromClient.getString("OperatorName"));
            List<InOrOutGoogsBean> list = new ArrayList<InOrOutGoogsBean>();
            for(int i = 0;i < jsonobjFromClient.getInt("Quantity");i++){
                JSONObject jsoncon = new JSONObject();
                jsoncon = jsonobjFromClient.getJSONObject("Commodity" + i);
                InOrOutGoogsBean iog = new InOrOutGoogsBean();
                SupplierInfoBean sib = new SupplierInfoBean();
                GoodsInfoBean gib = new GoodsInfoBean();
                iog.setGoodsNO(jsoncon.getString("ID"));
                iog.setGoodsName(jsoncon.getString("Name"));
                iog.setGoodsNum(jsoncon.getInt("Count"));
                iog.setGoodsPrice(jsoncon.getInt("Price"));
                iog.setSumPrice(jsoncon.getInt("Count")*jsoncon.getInt("Price"));
                iog.setSupplierNo(jsoncon.getString("SupplierID"));
                iog.setSupplierName(jsoncon.getString("SupplierName"));

                gib.setGoodsNo(jsoncon.getString("ID"));
```

登录验证：

```
package cn.com.servers;

import net.sf.json.JSONObject;

public class LoginServer {
    public String UserLogin(JSONObject  jsonobjFromClient,JSONObject json){
        UserInfoDAOImpl daou = new UserInfoDAOImpl();
        String jsonString = null;

        if(daou.validateByUserName(jsonobjFromClient.getString("UserNo"),jsonobjFromClient.getString("UserPass")) == true){
            jsonString = "true";
        }else{
            jsonString = "false";
        }


        json.put("Result", jsonString);
        String jstring = JSONUtils.valueToString(json);
        return jstring;
    }

}
```

查询类:

```
import java.util.List;

public class QueryServer {
    public String supplierQuery(JSONObject  json){
        SupplierInfoDAOImpl dao = new SupplierInfoDAOImpl();
        List<SupplierInfoBean> list = dao.getAllSupplierInfo();

        int i;
        for(i = 0;i < list.size();i++){
            json.put("Commint"  + i, list.get(i));
        }
        json.put("Quantity", i);
        String jsonString = JSONUtils.valueToString(json);
        return jsonString;
    }

    public String goodsQuery(JSONObject  json){
        GoodsInfoDAOImpl dao = new GoodsInfoDAOImpl();
        List<GoodsInfoBean> list = dao.getAllGoodsInfo();

        int i;
        for(i = 0;i < list.size();i++){
            json.put("Commint"  + i, list.get(i));
        }
        json.put("Quantity", i);
        String jsonString = JSONUtils.valueToString(json);
        return jsonString;
    }
```

删除类：

```java
import java.util.List;

public class QueryServer {
    public String supplierQuery(JSONObject  json){
        SupplierInfoDAOImpl dao = new SupplierInfoDAOImpl();
        List<SupplierInfoBean> list = dao.getAllSupplierInfo();

        int i;
        for(i = 0;i < list.size();i++){
            json.put("Commint"  + i, list.get(i));
        }
        json.put("Quantity", i);
        String jsonString = JSONUtils.valueToString(json);
        return jsonString;
    }

    public String goodsQuery(JSONObject  json){
        GoodsInfoDAOImpl dao = new GoodsInfoDAOImpl();
        List<GoodsInfoBean> list = dao.getAllGoodsInfo();

        int i;
        for(i = 0;i < list.size();i++){
            json.put("Commint"  + i, list.get(i));
        }
        json.put("Quantity", i);
        String jsonString = JSONUtils.valueToString(json);
        return jsonString;
    }
```