# 电子信息工程学院

实验报告书

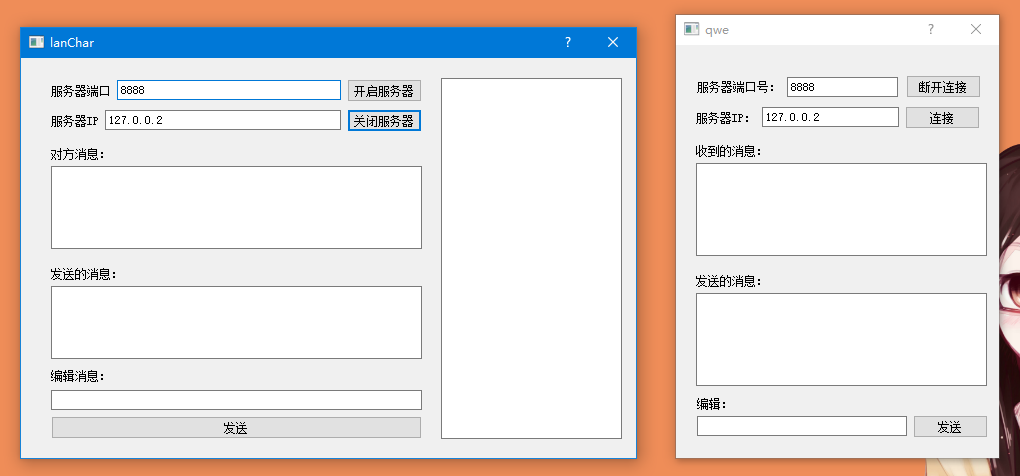
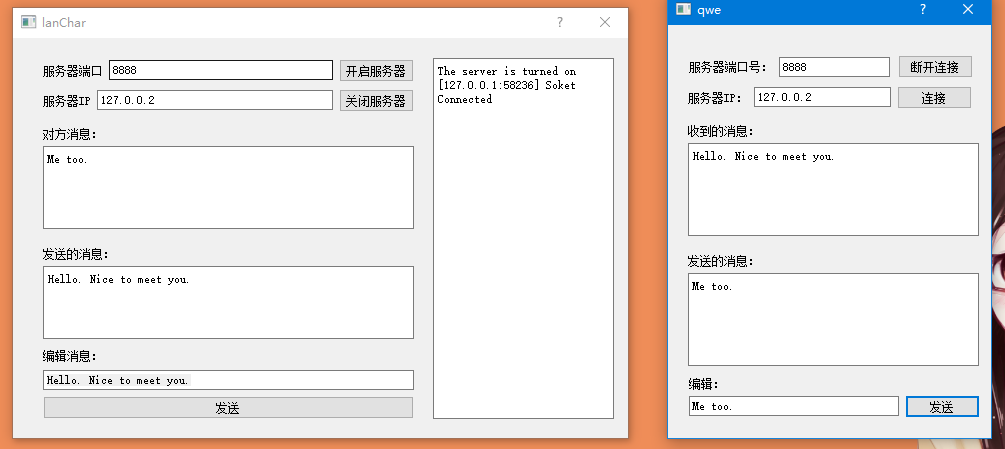
|  |  |
| --- | --- |
| 课程名称： | VC++ |
| 题 目： | VS2010 集成开发环境入门 |
| 实验类别： | vs利用qt插件实现局域网聊天 |
| 班 级： | 专升本1班 |
| 学 号： | 2022020321 |
| 姓 名： | 郭龙龙 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 评语： | | | | | | |
| 实验态度： | 认真 | （ ） | 一般 | （ ） | 差 | （ ） |
| 实验结果： | 正确 | （ ） | 部分正确 | （ ） | 错 | （ ） |
| 实验理论： | 熟悉 | （ ） | 了解 | （ ） | 不懂 | （ ） |
| 操作技能： | 掌握 | （ ） | 一般 | （ ） | 差 | （ ） |
| 实验报告： | 规范 | （ ） | 一般 | （ ） | 差 | （ ） |
| 成绩： |  | | 指导教师： |  | | |
| 批阅时间： |  | | |

1、概述

　　本程序利用了Visual Studio 2010中寄存了qt5.5插件，利用qt5.5插件实现了局域网的聊天的功能，其中程序分为服务器端和客户端，利用服务器端开启服务器，客户端进行连接服务器，实现客户端和服务器端消息传递，多个客户端实现局域网聊天的功能。

2、效果

3、实现步骤

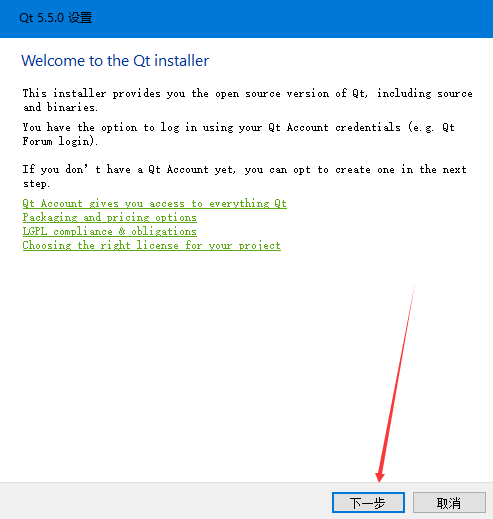
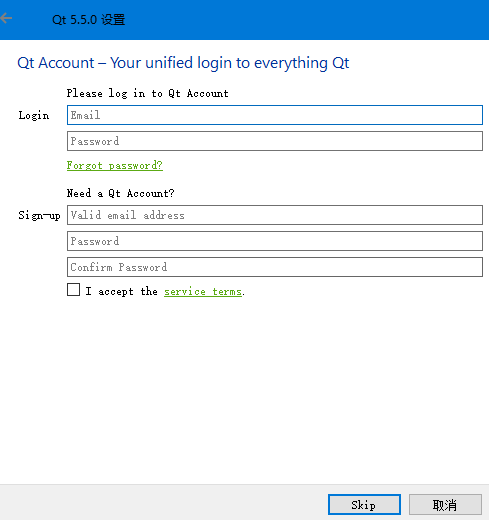
3.1、下载软件

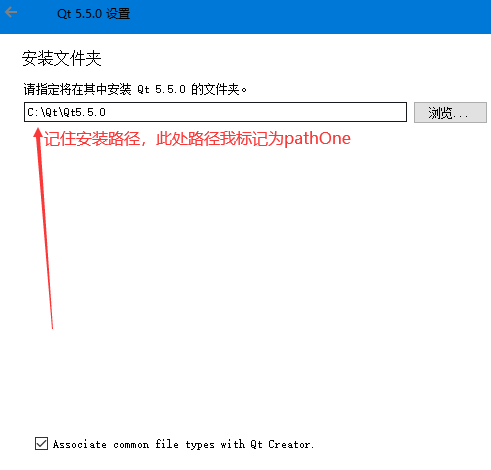
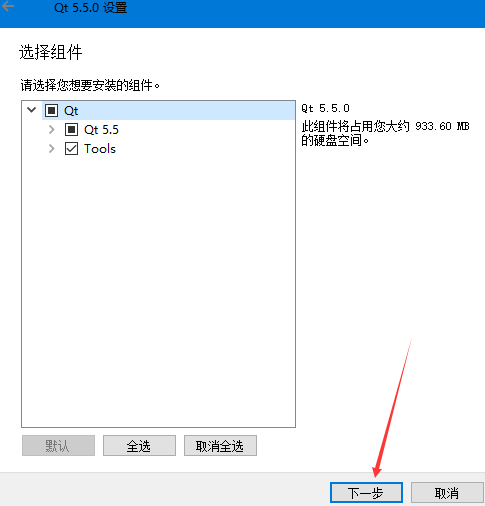
　　切记：下载qt插件软件必须和Visual Studio版本相匹配。

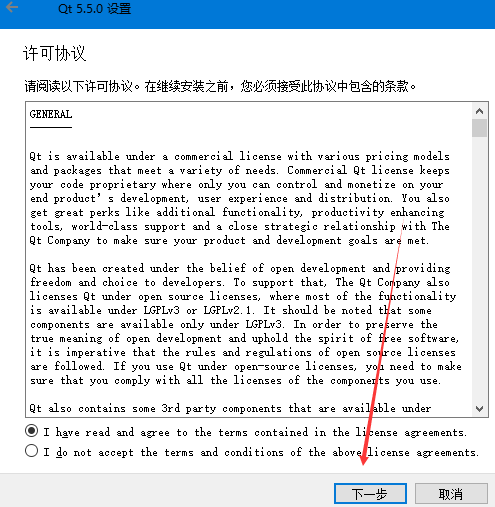
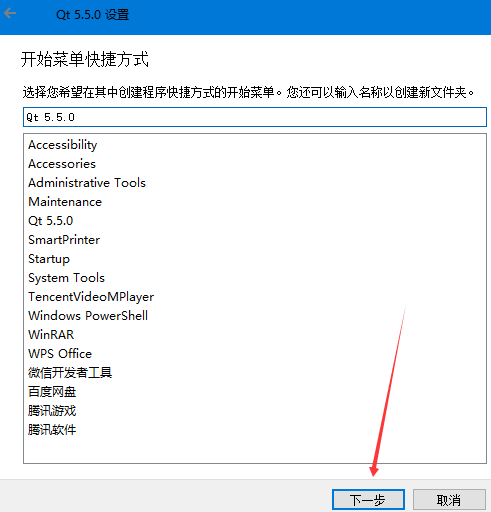
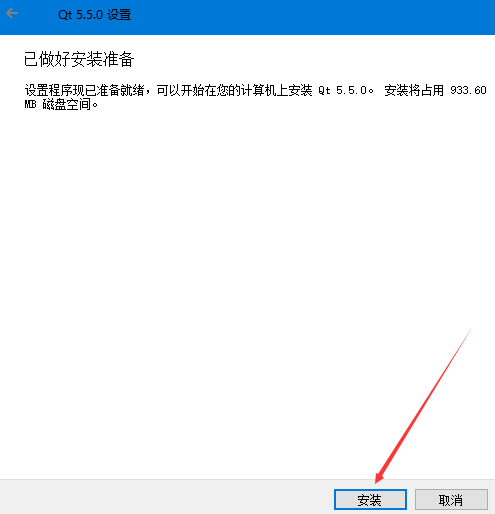
　　Visual Studio 2010：

3.2、安装软件

3.2.1、安装qt软件（qt-opensource-windows-x86-msvc2010-5.5.0）

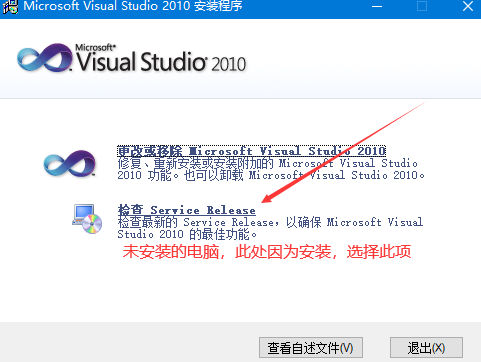
　　

 3.2.2、安装VS2010旗舰版.iso

 　　双击VS2010旗舰版.iso，　打开以下界面，选择autorun.exe



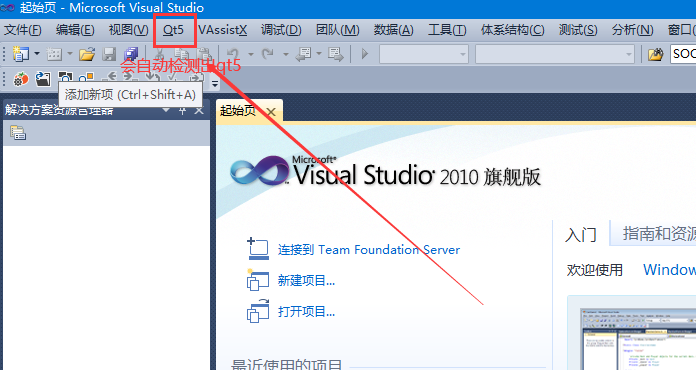
3.3.3、安装qt插件

　　双击qt插件（qt-vs-addin-1.2.4-opensource）

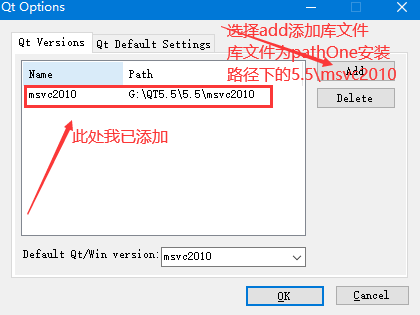
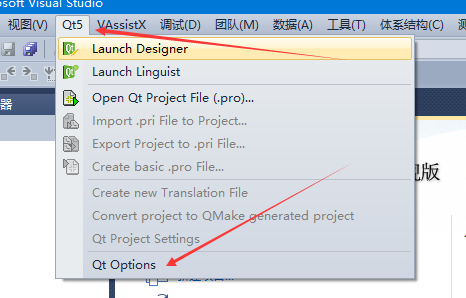
　　此处我已安装，系统会检测出已安装，所以也不作为截屏分享，按照安装流程，除安装路径可选择修改，其他均可以不做任何修改，点击下一步，安装即可。

4、配置环境

　　4.1、双击已安装的vs2010程序

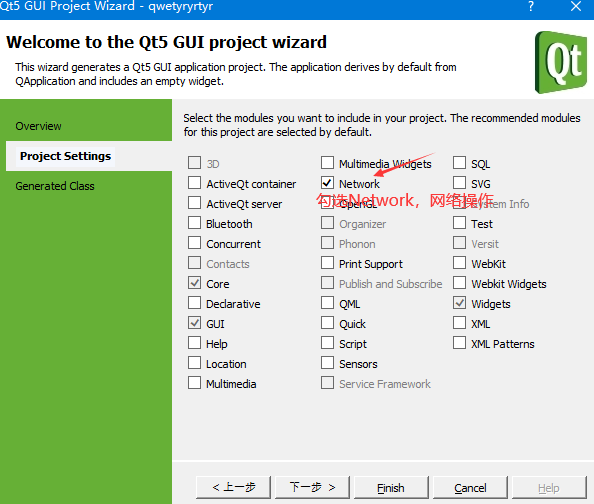
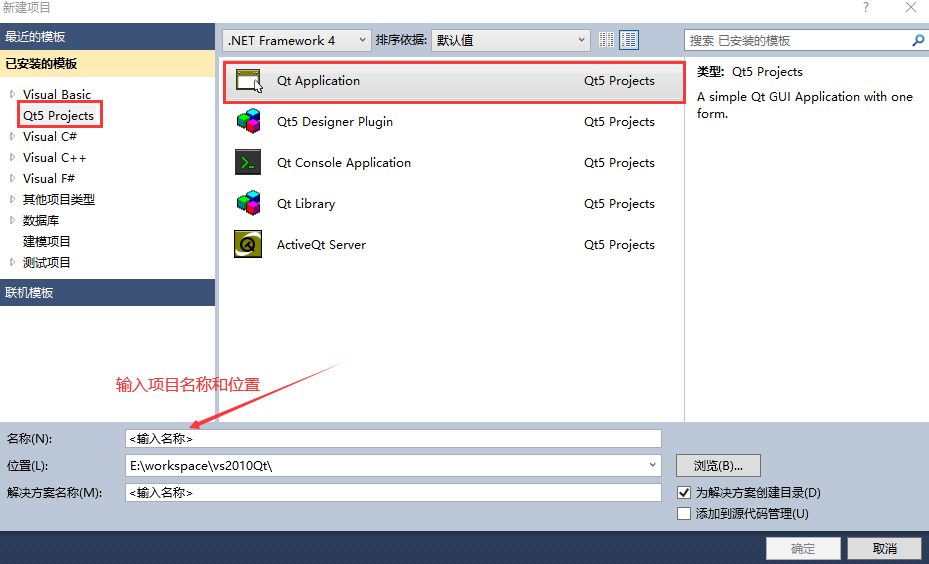
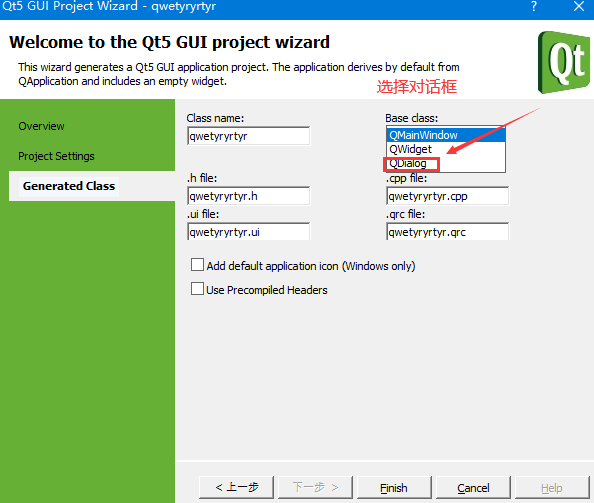


 　　4.3、选择qt库文件



5、新建项目

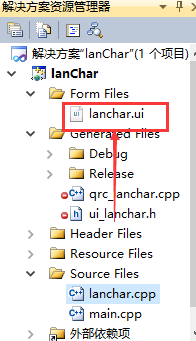
　　新建服务器项目

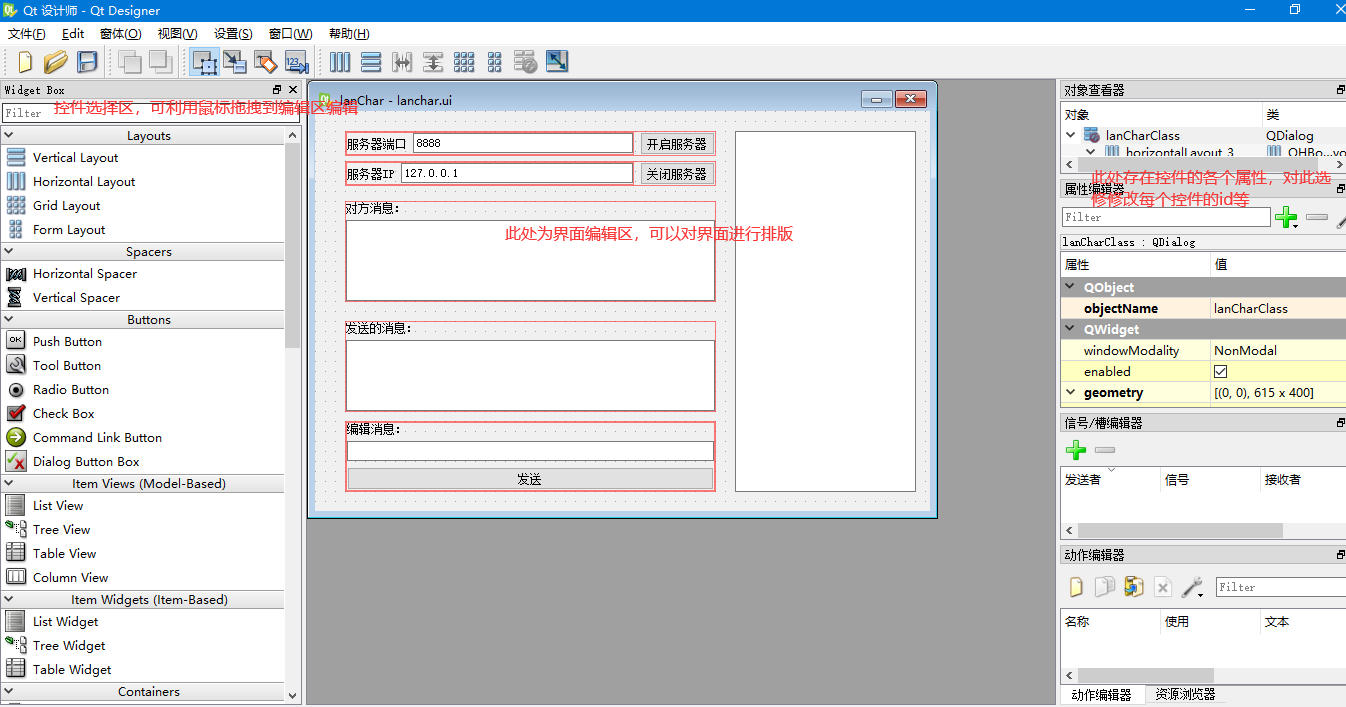
6、编辑页面、编写服务器代码

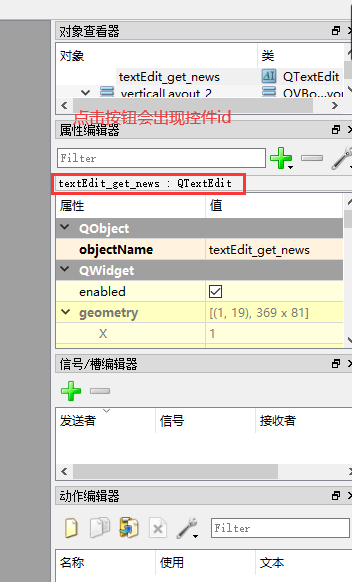
　　在编辑之前先编译，方便生成\*.h文件的同时，以防出错

6.1、双击如下图文件



出现以下页面，利用控件编辑出以下页面





6.2、代码编辑

　　6.2.1头文件代码

#ifndef LANCHAR\_H

#define LANCHAR\_H

#include <QtWidgets/QDialog>

#include "ui\_lanchar.h"

#include <QTcpServer>

#include <QTcpSocket>

#include <QHostAddress>

#include <QMessageBox>

class lanChar : public QDialog

{

Q\_OBJECT

public:

lanChar(QWidget \*parent = 0);

~lanChar();

private:

//初始化server操作

void initServer();

public slots:

void on\_button\_start\_server\_clicked();

void on\_button\_set\_news\_clicked();

void on\_button\_connect\_server\_clicked();

void sltSocketRead();

private:

//创建全局服务器对象

//存储已连接的socket对象

//存取已连接的用户

QList<QTcpSocket\*> clientList;

//创建服务器对象

QTcpServer \*server;

//socket对象

QTcpSocket \*client;

Ui::lanCharClass ui;

};

#endif // LANCHAR\_H

**6.2.3、cpp中代码**

#include "lanchar.h"  
#include <winsock.h>  
lanChar::lanChar(QWidget \*parent)  
: QDialog(parent)  
{  
ui.setupUi(this);  
  
client = NULL;  
}

lanChar::~lanChar()  
{

}  
void lanChar::initServer()  
{  
}  
//开启服务器按钮  
void lanChar::on\_button\_start\_server\_clicked(){  
//创建Server对象  
server = new QTcpServer(this);  
//获取ip地址  
const QString address\_text=ui.lineEdit\_address->text();  
const QHostAddress address=(address\_text=="Any")  
?QHostAddress::Any  
:QHostAddress(address\_text);  
//获取端口号  
const unsigned short port=ui.lineEdit->text().toUShort();  
//开始监听，并判断是否成功  
if(server->listen(address,port)){  
//连接成功就将输入框提示服务器已开启  
ui.textRecv->append("The server is turned on");  
}

connect(server, &QTcpServer::newConnection,[this](){  
//取出建立好连接的套接字  
client = server->nextPendingConnection();  
clientList.append(client);  
ui.textRecv->append(QString("[%1:%2] Soket Connected")  
.arg(client->peerAddress().toString())  
.arg(client->peerPort()));  
//QString li=client->peerAddress().toString();

QObject::connect(client, SIGNAL(readyRead()), this, SLOT(sltSocketRead()));  
});  
//QObject::connect(client, SIGNAL(readyRead()), this, SLOT(sltSocketRead()));  
}  
void lanChar::on\_button\_set\_news\_clicked(){  
const QString get\_data=ui.lineEdit\_3->text();  
//判断是否开启了server  
if(!server->isListening())  
return;  
if (get\_data==""){  
QMessageBox::about(NULL, "About","没有输入任何语句");  
return;  
}  
const QByteArray send\_data=get\_data.toUtf8();  
QTcpSocket \*socket=server->nextPendingConnection();  
ui.textEdit\_set\_news->append(get\_data);  
for(int i = 0; i < clientList.count(); i++)  
{  
socket = clientList.at(i);  
socket->write(send\_data);  
}  
socket->flush();  
}

void lanChar::on\_button\_connect\_server\_clicked(){  
server->close();  
ui.textRecv->append("The server is down！");  
for(int i = 0; i < clientList.count(); i++)  
{  
client->disconnectFromHost();  
if(client->state()!=QAbstractSocket::UnconnectedState){  
client->abort();  
}  
}  
}

void lanChar::sltSocketRead()  
{  
ui.textEdit\_get\_news->append(client->readAll());  
}

**此处服务器端代码编写完成**

**7、开始编辑客户端代码**

客户端和服务器端一样创建项目文件，编辑页面

7.1、以下为头文件代码

#ifndef QWE\_H  
#define QWE\_H

#include <QtWidgets/QDialog>  
#include "ui\_qwe.h"  
#include <QTcpServer>  
#include <QTcpSocket>  
#include <QHostAddress>  
#include <QMessageBox>

class qwe : public QDialog  
{  
Q\_OBJECT

public:  
qwe(QWidget \*parent = 0);  
~qwe();

public slots:  
void on\_pushButton\_clicked();  
void on\_pushButton\_2\_clicked();  
void on\_pushButton\_3\_clicked();  
private:  
QList<QTcpSocket\*> clientList;  
//socket对象  
QTcpSocket \*client;  
Ui::qweClass ui;

};

#endif // QWE\_H

**7.2、以下为cpp文件代码**

#include "qwe.h"

qwe::qwe(QWidget \*parent)  
: QDialog(parent)  
{  
ui.setupUi(this);  
}

qwe::~qwe()  
{

}

void qwe::on\_pushButton\_clicked(){

/\*连接服务器\*/  
//实例化QtcpSocket对象  
client = new QTcpSocket(this);  
//连接服务器的ip  
const QString address=ui.lineEdit\_2->text();  
//服务器的端口号  
const unsigned short port=ui.lineEdit->text().toUShort();  
//连接  
client->connectToHost(address,port);  
/\*if(!(client->state()==QAbstractSocket::ConnectedState))  
ui.lineEdit\_2->setText("The server is turned on");\*/  
/\*读取服务器消息\*/  
connect(client,&QTcpSocket::readyRead,[=]{  
//没有可读的数据就返回  
if(client->bytesAvailable()<=0)  
return;  
//注意收发两端文本要使用对应的编解码  
const QString recv\_text=QString::fromUtf8(client->readAll());  
ui.textEdit->append(recv\_text);  
});  
}  
void qwe::on\_pushButton\_2\_clicked(){  
const QString get\_data=ui.lineEdit\_3->text();  
ui.textEdit\_2->append(get\_data);  
const QByteArray send\_data=get\_data.toUtf8();  
client->write(send\_data.data());  
}  
void qwe::on\_pushButton\_3\_clicked(){  
client->disconnectFromHost();  
client->close();  
ui.textEdit->append("Server Disconnected");

}

**此处项目以全部完成，点击运行编译即可**

**总结：**

**在实验中，我使用 Visual Studio 借助 Qt 插件、Socket 编程实现了一个局域网聊天室和客户端。这让我对socket的通信机制有了更深的理解。对相关的代码有了更深的理解。**