南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

电子信息工程综合实验报告

作　　者：马子轩　　学　号：916104OG0826

学　　院：电子工程与光电技术学院

专业(方向)：电子信息工程

题　　目：物联网实验

指导者：钱玉文

评阅者：

2019 年　　10 月

目 次

1 实验一：触摸屏程序开发

1.1 触摸屏程序开发设备连接设置

1. 将 USB 线缆一端插入到触摸屏串口，另一端插入到 PC 机的 USB 接口，（操作之前，要先安装 ch340g usb 转串口驱动）,如图 1-1 所示

图 1-1USB 线缆接入设备触摸屏 USB 口

2. 将功能开关档位切换 2 档之后，可以将 PC 端开发的触摸屏程序下载至物联网设备的触摸屏中，如图 1-2 所示。

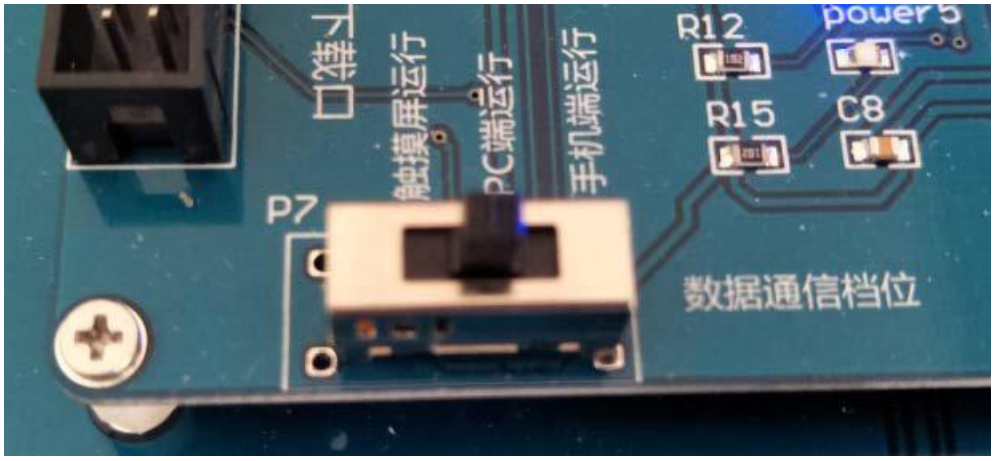

图 1-2 设备端与 PC 端通信档位

1.2 UI 设计与程序编写

1.2.1 新建工程

打开 USART HMI 开发平台，在起始页的项目窗体界面上，选择菜单中的文件->新建，进入新建项目对话框。在新建项目另存对话框中，按照项目路径保存新建项目名称，这里输入“HMI 风扇控制程序”，单击保存按钮。

当完成保存按钮之后，自动进入设置对话框，如图 1-3 所示，这里有设备类型选择和显示方向选择。根据实际的 HMI 串口屏的类型选择，这里选择 5 寸屏的 TJC8048T050_011 选项。

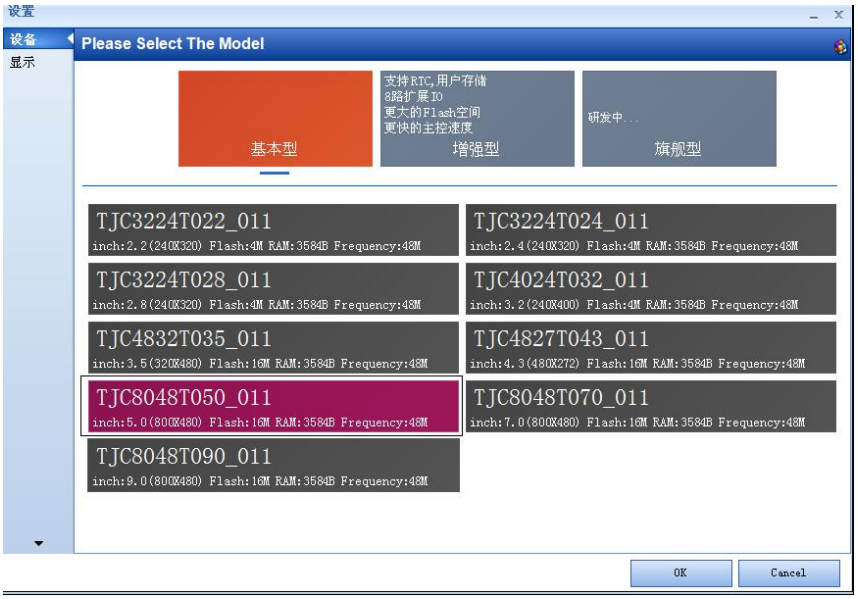

图 1-3 设置设备类型选项

显示方向选择根据实际需要进行选择，这里选择 0 度横屏显示，选择完成之后，单价 OK 按钮，完成项目构建。

1.2.2 项目图片添加

在 HMI 风扇控制程序的项目路径下，新建文件夹名为图片素材，然后将所需的图片拷贝进图片素材文件夹中。

在 HMI Uart 开发平台左下方的图片和字库的切换按钮中，选择图片选项，进行图片添加。

单击左边的“+”按钮，打开图片添加对话框，在 HMI 风扇空调控制程序的项目路径下，找到图片文件夹，打开文件夹，选择所需要的图片，选择完成之后，单击“打开”按钮。

选择完成之后，在图片栏中可以显示上一步所添加的各种图片，并自动完成图片的编号，这里的编号可以在后面的控件属性中进行设定。如图 1-4 所示.

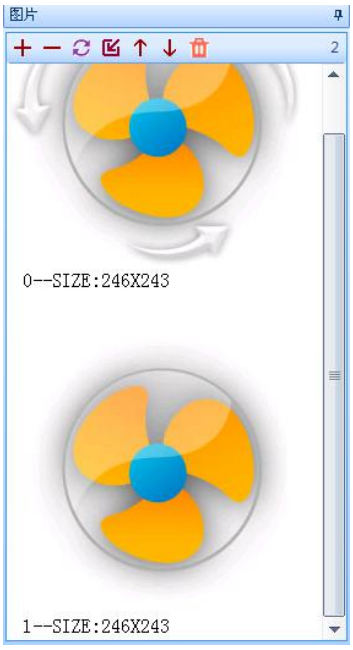

图 1-4 添加完成程序图片

1.2.3 项目字库制作与添加

在 HMI Uart 开发平台中，选择菜单中的工具->字库制作，进入字库制作对话框。

在字库制作工具对话框中，选择字高 40，字体加粗，汉字可以选择宋体，字母选择 Arial，字库名称输入 ziti，单击生成字库。如图 1-5 所示

图 1-5 设置字库制作相关属性

单击生成字库按钮，文件名输入 ziti，单击“保存”按钮。

完成之后,出现提示字库添加对话框，这里单击“是”按钮。

完成字库添加之后，在字库栏中显示上一步新建的字库内容。

1.2.4 项目界面设计

为了能在页面中显示背景颜色，这里选择页面 Page0，在属性栏中将 sta 属性设置为单色选项值，如图 1-6 所示。

图 1-6 页面背景 sta 属性设置

从 HMI Uart 开发平台工具箱中选择文本控件，然后在属性栏中将 txt-maxl 属性设置为 20，txt 属性设置为“风扇控制程序”，宽度 w 和高度 h 分别设置 300 和 50，如图 1-7 所示。

图 1-7 设置文本属性

文本控件属性设置成完成之后，出现如图 1-8 所示的页面效果.

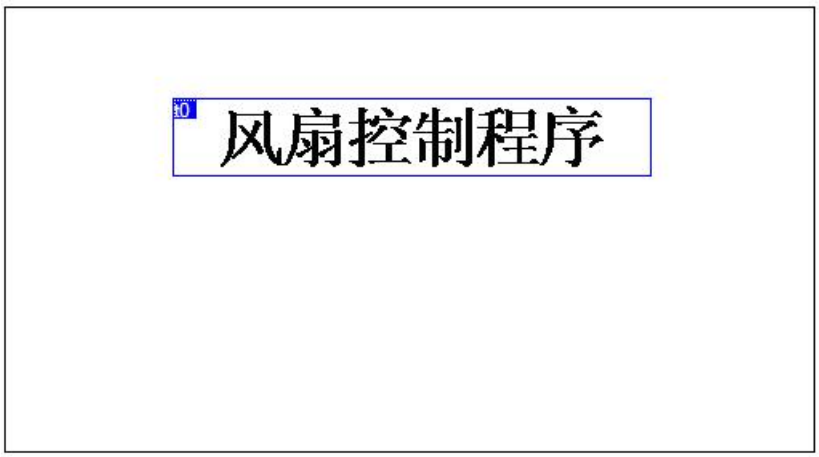

图 1-8 页面字高 40 的文本效果

按照前面添加字库的方法，在添加字高 32 的字库字体，从 HMI Uart 开发平台工具箱中选择一个文本控件，然后在属性栏中将 font 字体属性设置为 1，表示选择字高为 32 的字库字体，txt

属性设置为“风扇开”，文本控件属性设置成完成之后，出现如图 1-9 所示的页面效果.

图 1-9 设置字库和文本属性

从 HMI Uart 开发平台工具箱中单击一次双态按钮控件，添加一个双态按钮控件进入页面，然后将按钮 pic0 属性设置为显示的图片编号 1，按钮 pic1 属性设置为显示的图片编号 0 如图 1-10 所示.

图 1-10 双态按钮图片选择

上述操作步骤全部完成之后，如图 1-11 所示的页面效果。

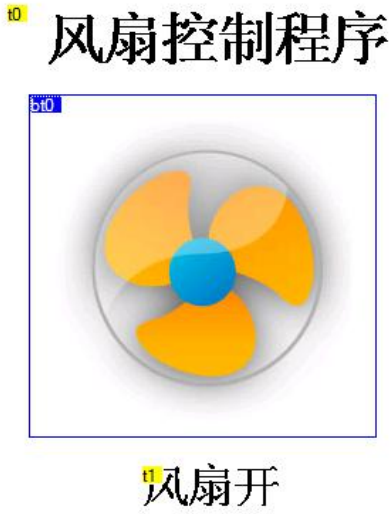

图 1-11 页面整体布局

1.2.5 项目功能实现

1. 风扇控制功能实现

选择双态按钮之后，在弹起事件栏中，填写如下代码：

```
if(bt0.val==1)
{
    print "268"
    t1.txt="风扇开"
}
else
{
    print "268"
    t1.txt="风扇关"
}
```

2. 代码编写完成之后，单击编译按钮，如果编译正确，显示如图 1-12 所示信息。

图 1-12 编译程序成功

单击调试按钮，这时在 PC 机端模拟触摸屏界面，单击风扇图片发出可以模拟风扇运行状态，如图 1-13 所示。

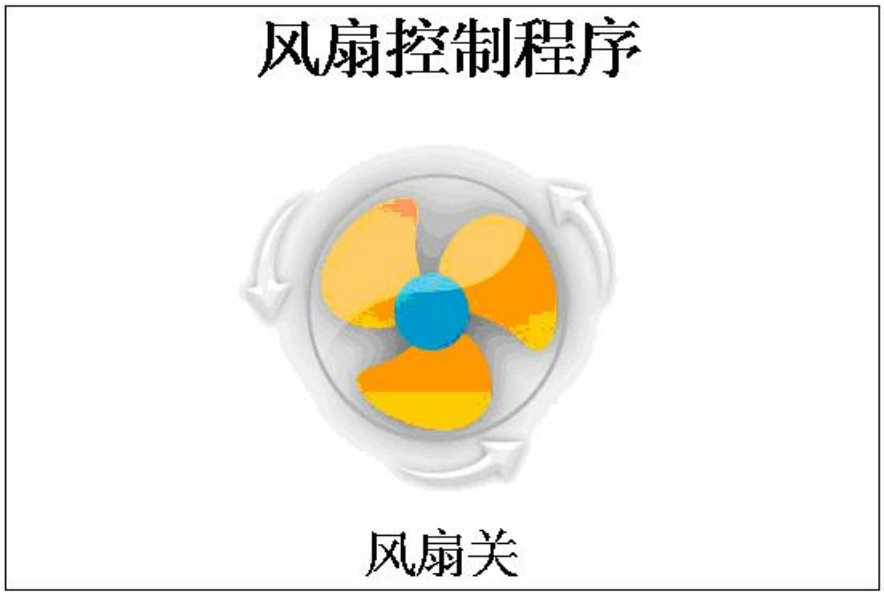

图 1-13 PC 端模拟程序运行

1.3 触摸屏程序下载至物联网云平台多功能教学设备

当单击 HMI 开发平台中的“下载”选项，能够将 PC 端触摸屏程序下载至物联网平台的触摸屏中。

单击“下载”选项之后，出现对话框信息，表示 PC 机和触摸屏连接成功，开始下载程序。

当如图 1-14 所示对话框信息，表示程序下载至触摸屏完毕。

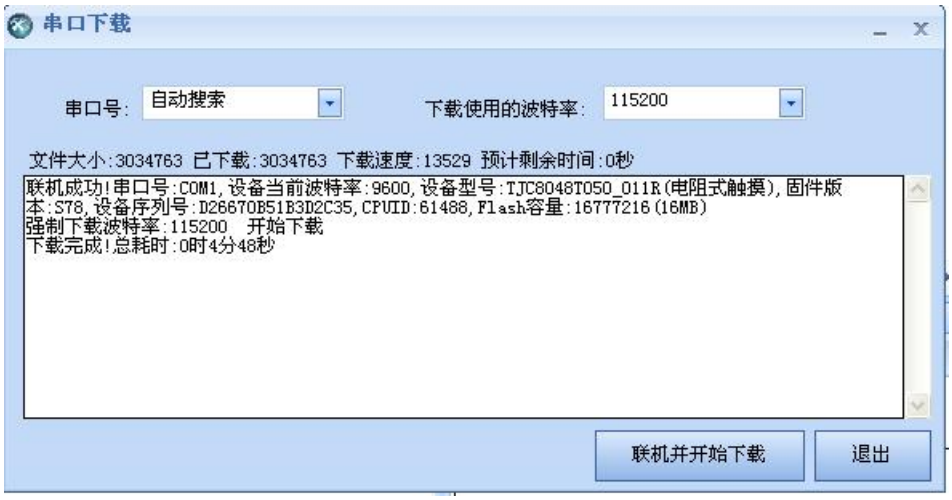

图 1-14 程序下载完成信息提示

将功能开关档位切换到 1 档之后，可以通过物联网设备中触摸屏控制风扇的转动和停止。如下图 1-15 所示。

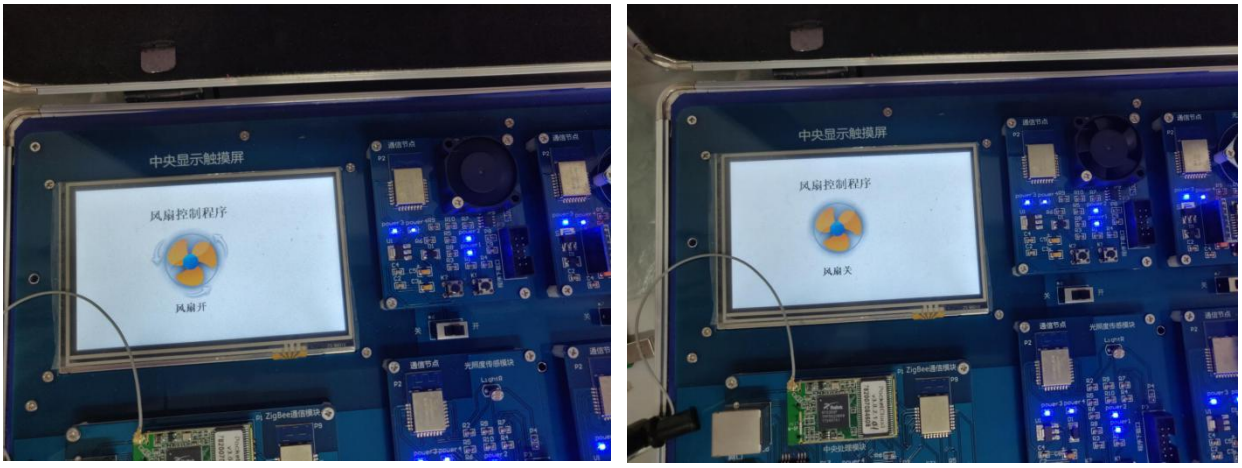

图 1-15 设备端触摸屏运行界面

2 基于 C#物联网开发

2.1 基于 C#物联网应用程序构建

在正确链接实验平台和正确安装软件之后，就可以新建项目。

2.1.1 创建温湿度采集控制系统工程项目

打开 VS.NET 开发环境，在起始页的项目窗体界面上，选择菜单中的文件->新建->项目选项，进入新建项目对话框，如图 2-1 所示，在左侧项目类型列表中选择 Windows 选项，在右侧的模板中选择 Windows 窗体应用程序选项，在下方的名称输入栏中输入将要开发的应用程序名“TempHumApp”，在位置栏选择应用程序所保存的路径位置，最后单击“确定”按钮。

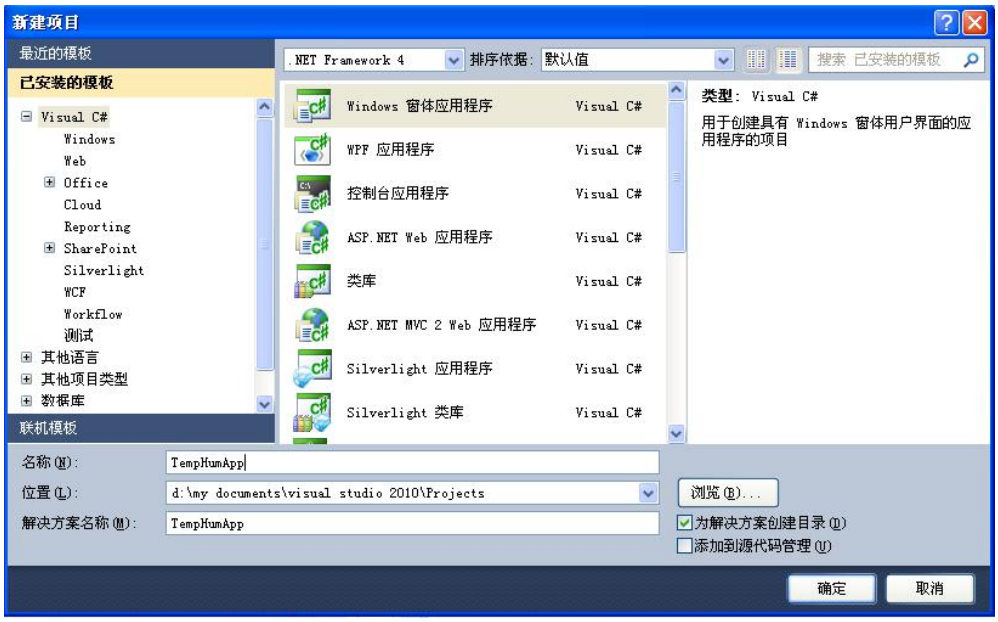

图 2-1 新建工程对话框

温湿度采集程序工程项目创建完成之后， 显示如图 2-2 所示的工程解决方案。

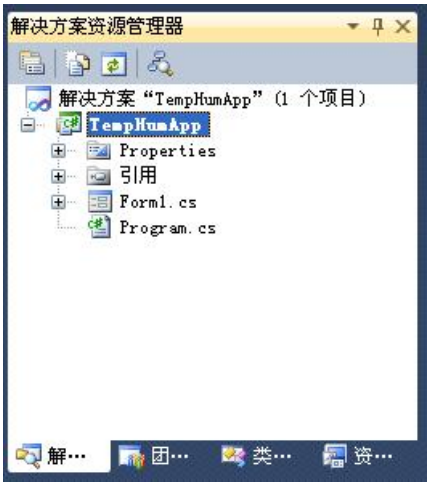

图 2-2 温湿度采集程序工程项目解决方案

2.1.2 窗体界面设计

选中整个 Form 窗体，然后在属性栏中的 Text 中输入温湿度采集程序文本值，如图 2-3 所示。

图 2-3 设置窗体名称文本信息

在界面设计中，添加两个 Label 控件，一个 GroupBox 控件，一个 ComboBox 控件,一个 Button 控件，完成程序标题的显示和界面串口参数的选择。在界面设计中，添加两个 Label 控件，一个 GroupBox 控件以及一个 ComboBox 控件，完成程序界面温湿度采集信息的实时显示，如图 2-4 所示.

图 2-4 温湿度界面设计

将上图中主要控件进行规范命名和设置初始值，如表 1 进行说明。

控件名称	命名	说明
ComboBox	comboPortName	设置串口名称，如 Com1、Com2、Com3
Button	buttonOpenCloseCom	打开或关闭串口按钮
TextBox	txtTemp	显示温度信息文本框

TextBox	txtHum	显示湿度信息文本框
GroupBox	gboxCom	串口操作组控件
GroupBox	gboxTemphum	温湿度操作组控件
Label	labeltitle	标题信息

表 1 程序各项主要控件说明

2.2 基于 C#温湿度采集程序功能实现

2.2.1 Form1 窗体代码文件（Form1.cs）结构

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
//以上几句是自动生成的
using System.IO.Ports;//我加的，包含串口相关的
namespace TempHumApp
{
    public partial class Form1 : Form
    {
        private SerialPort comm = new SerialPort();//我加的，新建一个串口变量
        string newstrdata = "";
        public Form1()
        {
            InitializeComponent();
        }
        private void buttonOpenCloseCom_Click(object sender, EventArgs e)
        {
```

```
        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }
        void comm_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {

        }
    }
}
```

2.2.2 方法说明

1. Form1_Load 方法。当窗体加载时，一方面执行串口类的 GetPortNames 方法，使之获得当前 PC 端可用的串口，并显示在下拉列表框中，另一方面添加事件处理函数 comm.DataReceived，使得当串口缓冲区有数据时，执行 comm_DataReceived 方法读取串口数据并处理。代码具体实现如下：

```
private void Form1_Load(object sender, EventArgs e)
{
    string[] ports = SerialPort.GetPortNames();
    Array.Sort(ports);
    comboPortName.Items.AddRange(ports);
    comboPortName.SelectedIndex = comboPortName.Items.Count > 0 ? 0 : -1;
    //初始化 SerialPort 对象
    comm.NewLine = "/r/n";
    comm.DataReceived += comm_DataReceived;
}
```

2. 打开或者关闭串口方法。单击打开串口按钮时，执行打开串口方法。首先 通过主界面窗体上的下拉列表框，选择可用的串口，如串口名称 Com1，设置波 特率为 9600，打开串口，再次单击关闭串口按钮时，执行关闭串口方法。在该 方法中将打开的串口对象进行关

闭操作代码具体实现如下：

```
private void buttonOpenCloseCom_Click(object sender, EventArgs e)
{
    //根据当前串口对象，来判断操作
    if (comm.IsOpen)
    {
        comm.Close();
        txtHum.Text = "";
        txtTemp.Text = "";
    }
    else
    {
        //关闭时点击，则设置好端口，波特率后打开
        comm.PortName = comboPortName.Text;
        comm.BaudRate = 9600;
        try
        {
            comm.Open();
        }
        catch (Exception ex)
        {
            //捕获到异常信息，创建一个新的 comm 对象，之前的不能用了。
            comm = new SerialPort();
            //现实异常信息给客户。
            MessageBox.Show(ex.Message);
        }
    }
    //设置按钮的状态
    buttonOpenCloseCom.Text = comm.IsOpen ? "关闭串口" : "打开串口";
}
```

3. 读串口数据方法。当串口缓冲区有数据时，执行 comm _DataReceived 方法 读串口数据。从串口读出数据之后，首先判断数据是否为空，当不为空时，在判 断字符串是否以“0101”开始，如果成立，则取 0101 的后面两位字符，它们是 温度数据。然后判断“0102”字符串是否存在，如果成立，则取 0102 的后面两 位字符，它们是湿度数据。代码具体实现如下：

```
void comm_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    this.BeginInvoke(new Action(() =>
    {
        string serialdata = comm.ReadExisting();
        newstrdata += serialdata;
        if (newstrdata.LastIndexOf("0101") >= 0)
        {
            int tempindex = newstrdata.LastIndexOf("0101");
            if (newstrdata.Substring(tempindex).Length >= 6)
            {
                txtTemp.Text = newstrdata.Substring(tempindex + 4, 2);
            }
        }
        if (newstrdata.LastIndexOf("0102") >= 0)
        {
            int humindex = newstrdata.LastIndexOf("0102");
            if (newstrdata.Substring(humindex).Length >= 6)
            {
                txtHum.Text = newstrdata.Substring(humindex + 4, 2);
            }
        }
    }), null);
}
```

2.3 实验结果

温湿度采集程序运行界面如图 2-5 所示。

图 2-5 温湿度采集程序运行界面

3 基于 QT 平台的物联网开发

首先先在 PC 上安装好 QT 平台，然后按正确的连接方式链接 PC 和试验箱。上述准备工作完成之后，即可开始项目构建

3.1 基于 QT 平台的物联网项目构建

1. 打开 QT Creator 开发环境，单击“文件”->“新建文件或工程…”，出现新建对话框如图 3-1 所示，单击选择“Qt GUI 应用”模版，单击“选择”按钮。

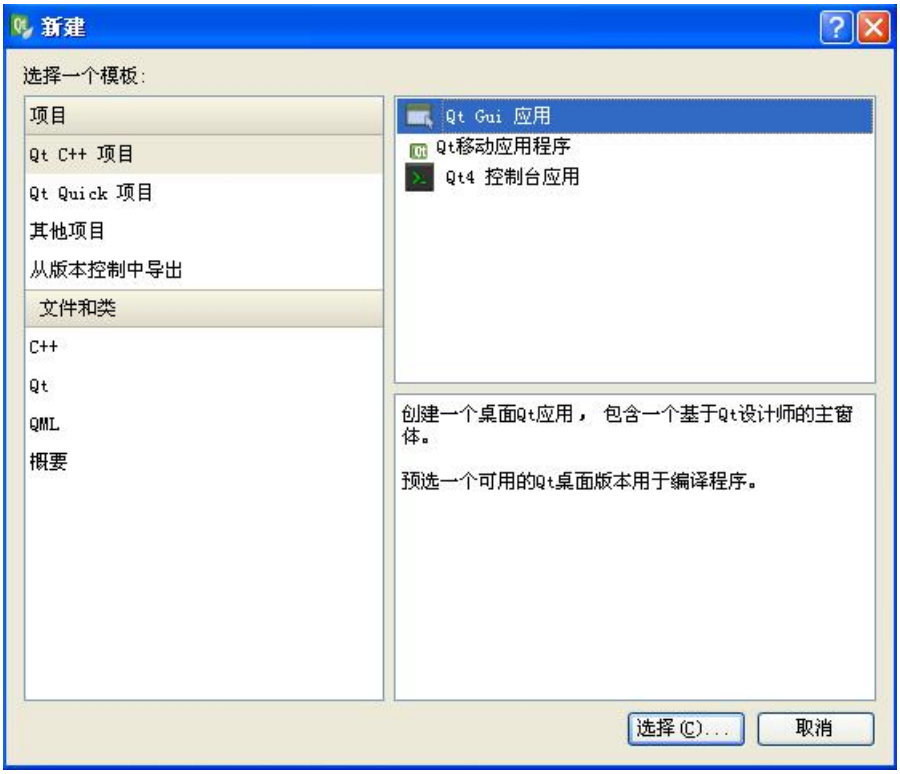

图 3-1 新建工程对话框

2. 名称输入：TempHumApp，单击“下一步”按钮。

3. 在 Qt 版本选择对话框，选择 QT4.8.5 版本，单击“下一步”按钮。

4. 在如图 3-2 所示的类信息对话框中，基类选择 QDialogt，类名为 myDialog，单击“下一步”按钮，工程构建完成。

图 3-2 选择 QDialog 基类

5. 添加第三方串口类文件

(1) 在 Window 系统下需要将 qextserialbase.cpp 和 qextserialbase.h 以及 win_qextserialport.cpp 和 win_qextserialport.h 这四个文件导入到 ZigbeeWenshiduApp 工程文件夹中。

6. 完成创建

温湿度采集系统工程项目创建完成之后，直接进入编辑模式，如图 3-3 所示，打开项目目录，可以看到 TempHumApp 文件夹，右键添加前面拷贝的四个文件吗，完成之后，在这个文件夹中包括了六个文件，各个文件功能说明如表 2 所示，

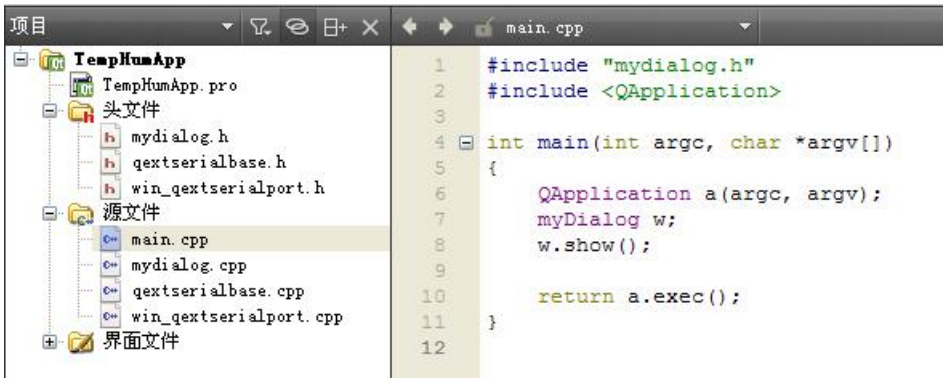

图 3-3 编辑模式

文件	功能说明
TempHumApp.pro	该文件是项目文件，其中包含了项目相关信息
TempHumApp.pro.user	该文件中包含了与用户有关的项目信息
mydialog.h	该文件是新建的 MyDialog 类的头文件

Mydialog.cpp	该文件是新建的 MyDialog 类的源文件
main.cpp	该文件中包含了 main()主函数
mydialog.ui	该文件是设计师设计的界面对应的界面文件

表 2 项目目录中各个文件功能说明

3.2 添加资源文件

在上面界面设计中，通过添加各种控件初步完成了整体界面设计，但为了美化程序的界面，QT 提供了一种资源文件的方法来美化程序界面，具体操作如下:

1. 单击“文件”->“新建文件或工程…”，出现新建对话框，在左侧的文件和类中选择 Qt，右侧选择 Qt 资源文件，单击“选择”按钮。

2. 将资源文件命名为 image，并将路径设置为 TempHumApp 工程项目所在的路径。

3. 完成上述操作之后，在项目中可以看到 TempHumApp 工程中增加了一个名为 image.qrc 资源文件。

4. 打开 image.qrc 资源文件，单击添加按钮，选择“添加前缀”，在前缀栏中 输入“/”,.添加了前缀之后，就可以往资源文件中添加资源文件了，依然选择单 击“添加” “添加文件”。这里选择工程项目中 images 文件夹下的四个图片文 件，添加完成之后，显示如图 3-4 所示的图片资源。

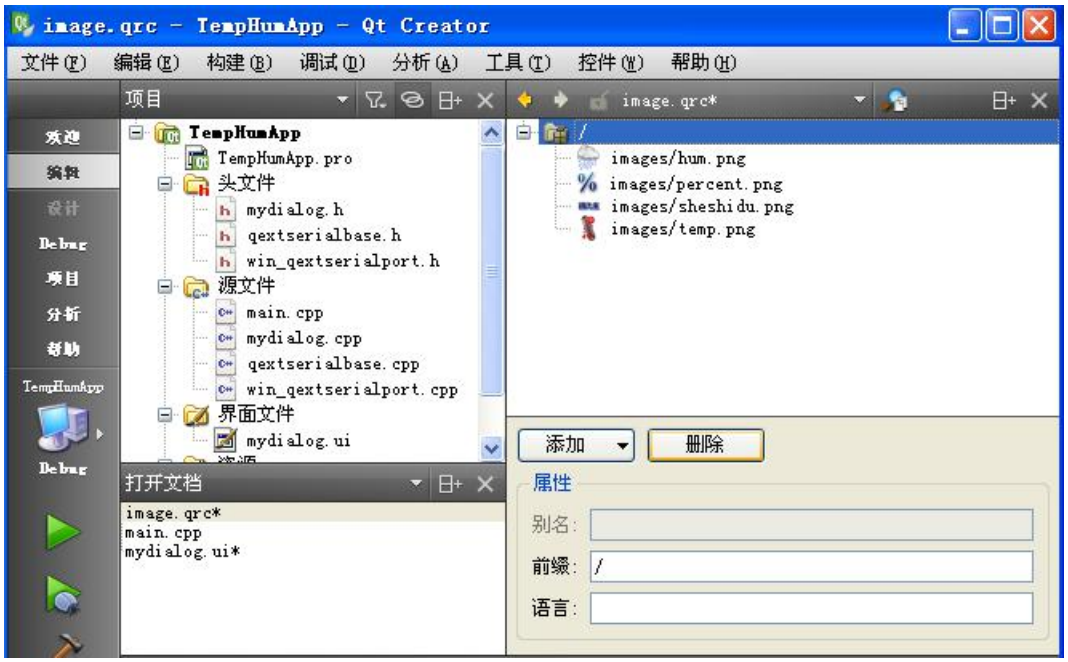

图 3-4 图片资源添加完成

3.3 窗体界面设计

1. 在界面设计中，添加一个 QComboBox 控件完成对串口通信参数的设置，二个

QPushButton 按钮，实现打开串口和关闭串口控制，一个 QGroupBox 控件，实现温湿度信息显示，操作完成之后，在主界面窗台上显示如图 3-5 所示的图片效果。

图 3-5 窗体设计界面

2. 将上图中主要控件进行规范命名和设置初始值，如表 3 进行说明

控件名称	命名	说明
ComboBox	portNameComboBox	设置串口名称，如 Com1、Com2、Com3
PushButton	openMyComBtn	打开串口按钮
PushButton	closeMyComBtn	关闭串口按钮
TextEdit	txtTemp	显示温度信息文本框
TextEdit	txtHum	显示湿度信息文本框

表 3 项目各项控件说明

3.4 基于 QT 平台的物联网项目功能实现

1. 定义和使用类对象 在 mywidget.h 文件中，添加 Window 平台下#include "win_qextserialport.h" 第三方头文件，并定义相应的方法和变量，具体定义如下：

```
#ifndef MYDIALOG_H
#define MYDIALOG_H
#include <QDialog>
#include "win_qextserialport.h"
#include <QTimer>
namespace Ui {
    class myDialog;
}
```

```
class myDialog : public QDialog
{
    Q_OBJECT
    public:
        explicit myDialog(QWidget *parent = 0);
        ~myDialog();
    private slots:
        void on_openMyComBtn_clicked();
        void on_closeMyComBtn_clicked();
        void readMyCom();
    private:
        Ui::myDialog *ui;
        Win_QextSerialPort *myCom;
        QString str;
        bool isOpen;
        QString wendu;
        QString shidu;
        QTimer *readTimer;
};
#endif
```

2. MYDIALOG_H2、mydialog.cpp 文件中方法说明如下：

（1）myDialog 构造方法。当实例化 myDialog 类对象的时候，执行 MyDialog 构造方法，在构造方法中，使关闭串口按钮不可用。代码具体实现 如下：

```
myDialog::myDialog(QWidget *parent) :
    QDialog(parent),
    ui(new Ui::myDialog)
    {
        ui->setupUi(this);
        ui->closeMyComBtn->setEnabled(false); //开始“关闭串口”按钮不可用
    }
```

（2）打开串口方法。单击打开串口按钮时，执行打开串口方法。首先通过主界面窗体上的下拉列表框，选择串口名称 Com3，构建串口对象，打开串口，设置波特率为 9600，设置无奇偶校验，设置数据位为 8 位，停止位为 1 位，最后通过 connect 函数建立信号和槽函数关联，使得当串口缓冲区有数据时，进行 readMyCom()读串口操作。代码具体实现如下：

```
void myDialog::on_openMyComBtn_clicked()
{
    QString portName = ui->portNameComboBox->currentText(); //获取串口名
    myCom = new Win_QextSerialPort(portName,QextSerialBase::Polling);
    //定义串口对象，并传递参数，在构造函数里对其进行初始化
    isOpen=myCom->open(QIODevice::ReadWrite); //打开串口
    myCom->setBaudRate(BAUD9600);
    myCom->setDataBits(DATA_8);
    myCom->setParity(PAR_NONE);
    myCom->setStopBits(STOP_1);
    myCom->setFlowControl(FLOW_OFF);
    myCom->setTimeout(500);
    readTimer = new QTimer(this);
    readTimer->start(100);
    //设置延时为 100ms
    connect(readTimer,SIGNAL(timeout()),this,SLOT(readMyCom()));
    //信号和槽函数关联，延时一段时间，进行读串口操作
    ui->openMyComBtn->setEnabled(false); //打开串口后“打开串口”按钮不可用
    ui->closeMyComBtn->setEnabled(true); //打开串口后“关闭串口”按钮可用
    ui->portNameComboBox->setEnabled(false);
}
```

（3）关闭串口方法。单击关闭串口按钮时，执行关闭串口方法。在该方法中首先将打开的串口对象进行关闭操作，然后将打开串口按钮变成可用状态，串口名称下拉列表框变成可用状态。代码具体实现如下：

```
void myDialog::on_closeMyComBtn_clicked()
{
```

```
    myCom->close();
    readTimer->stop();
    ui->openMyComBtn->setEnabled(true); //关闭串口后“打开串口”按钮可用
    ui->closeMyComBtn->setEnabled(false); //关闭串口后“关闭串口”按钮不可用
    ui->portNameComboBox->setEnabled(true);
}
```

（4）读串口数据方法。当定时器到，串口缓冲区有数据时，进行 readMyCom()读串口操作。从串口读出数据之后，首先判断数据是否为空，当不为空时，在判断字符串是否以“0101”开始，如果成立，则取 0101 的后面两位字符，它们是温度数据。然后判断“0102”字符串是否存在，如果成立，则取 0102 的后面两位字符，它们是湿度数据。代码具体实现如下：

```
void myDialog::readMyCom()
{
    QByteArray temp = myCom->readAll();//调用 readAll()函数，读取串口中所有数据，在上面可以看到其返回值是 QByteArray 类型。
    str=QString(temp);
    if(!str.isEmpty())
    {
        if(str.indexOf("0102")>=0)
        {
            QString shidu=str.mid((str.indexOf("0102")+4),2);
            if(shidu!=NULL)
            {
                ui->txtHum->setText(shidu);
            }
        }
        if(str.indexOf("0101")>=0)
        {
            QString wendu=str.mid((str.indexOf("0101")+4),2);
            if(wendu!=NULL)
            {
```

```
                ui->txtTemp->setText(wendu);
            }
        }
    }
}
```

3.5 实验结果

图 3-6 程序运行界面

4 基于 Andriod 平台的物联网开发

4.1 Andriod 手机开发项目构建

4.1.1 创建温湿度采集程序项目

打开 Eclipse 开发环境，单击“File”->New->Android Application Project”，在出现的对话框中，应用程序和项目名称输入：AndTempHumApp，SDK 选择 API9，单击“Next”按钮。在 Activity 对话框，Activity 窗体界面栏输入“MainActivity”，布局文件 XML 输入“main”，单击“Finish”按钮。

ZigBee 采集控制系统工程项目创建完成之后，如图 4-1 所示，打开项目目录，可以看到 AndTempHumApp 工程文件夹，在这个文件夹中包括了六个文件。

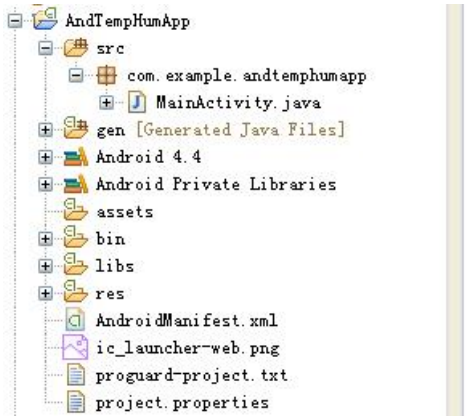

图 4-1 项目工程界面

4.1.2 添加图片资源

在 AndTempHumApp 工程所在的路径下，如 E:\AndroidSmart\AndTempHumApp \res\drawable-mdpi 中添加如图 4-2 所示的图片文件。

图 4-2 添加图片资源

4.1.3 窗体界面设计

首先打开项目 res\layout 目录，找到 activity_main.xml，双击文件，然后在选择 Graphical Layout 选项。

在右键界面出现下拉菜单选项，选择如图 4-3 所示的 Edit Background 项。

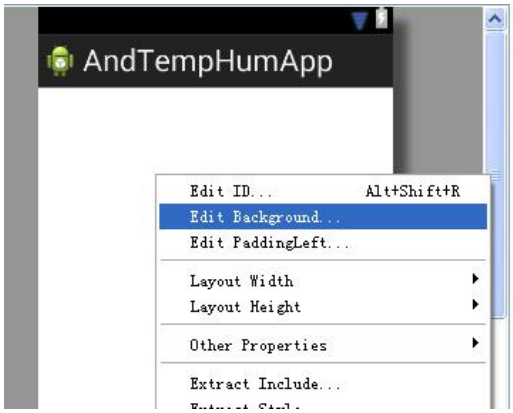

图 4-3 找到 Edit Background 项

出现的对话框中，选项 Drawable 目录下的 bk2 背景图片，单击 OK 按钮。选择完成之后，界面显示如图 4-4 所示的背景效果。

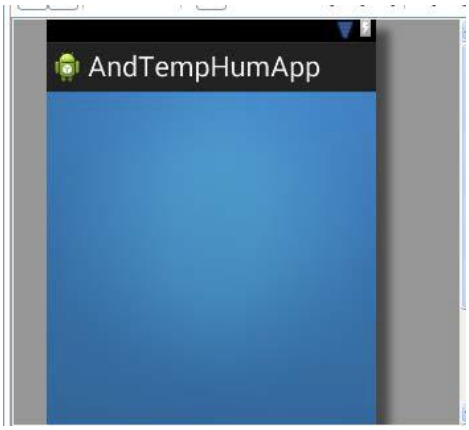

图 4-4 界面显示背景图片

打开项目 res\values 目录下的 strings.xml 文件，双击 strings.xml 文件，选择 app_name(String)，在 Value*栏中输入温湿度采集程序，如图 4-5 所示。

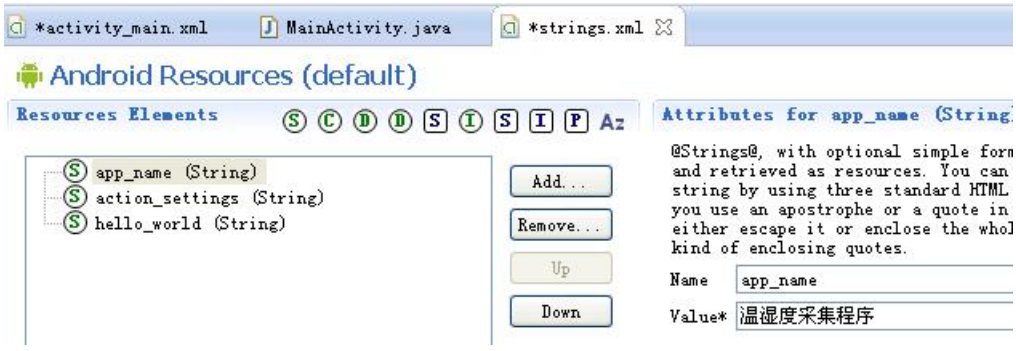

图 4-5 设置标题内容

输入完成之后，关闭并保存 activity_main.xml，然后在打开 activity_main.xml 文件，这就出现如图 4-6 所示的界面标题栏文字效果。

图 4-6 显示标题内容

添加三个 EditText 控件，分别代表网络 IP 地址文本框、温度文本框、湿度文本框，添加一个 Button 按钮控件，代表连接网络按钮，两个文本标签控件，显示温度和湿度文本，界面设计完成之后，如图 4-7 所示界面效果。

图 4-7 界面设计完成

4.1.4 编辑布局文件 main.xml

通过添加控件进行设计之后，形成以下 XML 界面布局文件。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/RelativeLayout1"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:background="@drawable/bk2"
android:orientation="vertical"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity" >
<Button
    android:id="@+id/button_connect_socketserver"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/editText_wendu"
    android:layout_toRightOf="@+id/editText_wendu"
    android:text="连接" />
<EditText
    android:id="@+id/editText_wendu"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:ems="10" />
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/textView1"
    android:layout_alignBottom="@+id/textView1"
    android:layout_toRightOf="@+id/textView2"
    android:ems="10" />
```

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/editText_wendu"
    android:layout_marginTop="21dp"
    android:layout_toLeftOf="@+id/editText1"
    android:text="温度："
    android:textAppearance="?android:attr/textAppearanceLarge" />
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/editText_shidu"
    android:layout_alignLeft="@+id/editText_wendu"
    android:text="湿度："
    android:textAppearance="?android:attr/textAppearanceLarge" />
<EditText
    android:id="@+id/editText_shidu"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/editText1"
    android:layout_below="@+id/editText1"
    android:ems="10" >
</EditText>
</RelativeLayout>
```

4.1.5 将上图中主要控件进行规范命名和设置初始值，如表 4 进行说明

控件名称	命名	说明
EditText	editText_wendu	显示温度信息文本框
EditText	editText_shidu	显示湿度信息文本框

EditText	editText_IP	输入 IP 地址信息
Button	button_connect_socketserver	打开网络连接

表 4 项目各项控件说明

4.2 Andriod 手机开发项目功能实现

4.2.1 MainActivity 代码编写

```
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.UnknownHostException;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
public class MainActivity extends Activity {
    private EditText ip,wenduEditText,shiduEditText;
    private Button connectButton;
    private boolean isConnect = false;
    private String ServerIP = null;
    private String recieve_wendu_String,recieve_shidu_String;
    private int ServerPort = 8002;
    private Socket socket = null;
    private byte[] recivebuffer = new byte[9];
    static final int RX_TEMPDATA_UPDATE_UI = 1;
    static final int RX_HUMDATA_UPDATE_UI = 2;
    private Handler mainHandler = null;
```

```
private ReceiveThread receiveThread = null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    initControl();
    initMainHandler();
    ip.setText("192.168.10.1");
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
class buttonClick implements OnClickListener{
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button_connect_socketserver:
                if (!isConnect){
                    new Thread(connectThread).start();
                }
                break;
            default:
                break;
        }
    }
}
Runnable connectThread = new Runnable() {
    @Override
```

```
        public void run() {
            try {
                ServerIP = ip.getText().toString();
                socket = new Socket(ServerIP, ServerPort);
                isConnect = true;
                //启动接收线程
                receiveThread = new ReceiveThread(socket);
                receiveThread.start();
            } catch (UnknownHostException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    };
    private class ReceiveThread extends Thread{
        private BufferedReader inStream = null;
        ReceiveThread(Socket socket){
            try {
                inStream= new BufferedReader(
                        new InputStreamReader(socket.getInputStream(), "UTF-8")
                );
            } catch (IOException e) {
                //TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        @Override
```

```
public void run(){
    while (socket.isConnected()) {
        try
        {
            /*定义接收数据的 buffer，并清零*/
            char[] buffer = new char[64];
            for (int i = 0; i < buffer.length; i++) {
                buffer[i] = '\0';
            }
            int len = inStream.read(buffer);
            if (len >0) {
                // 解析数据
                String strRecvMsg = String.copyValueOf(buffer);
                String Sub=null;
                int index;
                if(strRecvMsg.indexOf("0101")>-1)
                {
                    index=strRecvMsg.indexOf("0101");
                    Sub=strRecvMsg.substring(index+4,index+6);
                    Message msg = new Message();
                    msg.what = RX_TEMPDATA_UPDATE_UI ;
                    msg.obj=Sub;
                    mainHandler.sendMessage(msg);
                }
                if(strRecvMsg.indexOf("0102")>-1)
                {
                    index=strRecvMsg.indexOf("0102");
                    Sub=strRecvMsg.substring(index+4,index+6);
                    Message msg = new Message();
                    msg.what = RX_HUMDATA_UPDATE_UI ;
```

```
                                msg.obj=Sub;
                                mainHandler.sendMessage(msg);
                            }
                        }
                    }
                    catch (Exception e) {
                        // TODO: handle exception
                    }
                }
            }
        }
    }
    void initControl()
    {
        ip = (EditText) findViewById(R.id.editText_IP);
        wenduEditText = (EditText)findViewById(R.id.editText_wendu);
        shiduEditText = (EditText)findViewById(R.id.editText_shidu);
        connectButton = (Button) findViewById(R.id.button_connect_socketserver);
        connectButton.setOnClickListener(new buttonClick());
    //leddengButton.setEnabled(false);
    }
    @SuppressLint("HandlerLeak")
    void initMainHandler()
    {
        mainHandler = new Handler(){
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case RX_TEMPDATA_UPDATE_UI:
                    wenduEditText.setText(msg.obj.toString());
                    break;
                case RX_HUMDATA_UPDATE_UI:
```

```
                    shiduEditText.setText(msg.obj.toString());
                    break;
                }
            }
        };
    }
}
```

4.3 Android 温湿度采集程序下载至手机端运行

1. 当程序编译成功之后，可以通过 USB 线缆将手机和 PC 机进行连接，同时 PC 端安装手机助手，完成手机和 PC 机之间的同步，最后选择 Run As->Android Application 项，如图 4-8 所示，将温湿度采集程序下载至手机端。

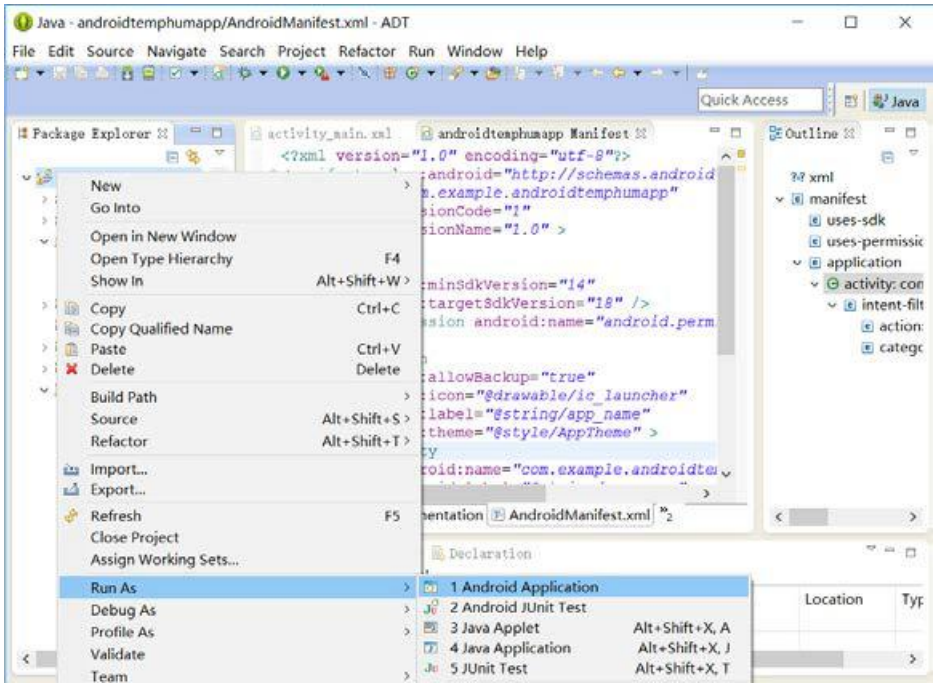

图 4-8 温湿度采集程序下载至手机端

2. 前面物联网开发平台中的智能网关的无线热点是不同的，每个箱子的热点名有编号，编号在实验平台的网关上有，这时在手机端打开无线网络，找到热点，并链接。值得注意的是，经反复测试，此热点只能一个设备链接。多个设备链接时，APP 会闪退。

4.4 实验结果

Andriod 手机端上的温湿度采集程序运行界面如图 4-9 所示

图 4-9 程序运行界面

5 实验总结

5.1 实验过程中遇到的问题和改进思考

5.1.1 问题一

使用触摸屏程序时，触摸屏虽可以实现功能，但屏幕显示的内容和实际情况不符，无论怎么点，都显示风扇开。

【解答】：程序的书写上出现了问题，两个 if else 语句都用的同样的表达条件。通过稍微修改程序，成功实现了正确的功能。

5.1.2 问题二

APP 链接时出现闪退等多种问题

【解答】：由于 APP 程序书写时，未加入访问许可的命令，导致手机链接 IP 时，链接的请求被拒绝。修改了相关程序，成功获取到物联网平台的信息。

5.2 实验的收获与感受

利用不到一周的时间，体验了物联网平台的基础知识和理论，增长了不少的技能。经过科研训练，我主要做的是网站的项目，在本次做实验的过程中，也体会到目前的科技越来越离不开网络。最后 APP 制作时，由于有了之前的 iOS APP 和 HTML5+ APP 的制作经验，对于 APP 请求数据的理论是大概清楚的。但是由于对 java 语言的不熟悉，加上对物联网网关的 request 方式不熟悉，一直没有做出来。最终，通过和同学以及老师交流之后，终于找到了问题所在，也成功做出了一个可以实现功能的 APP。

很感谢钱玉文老师的辛苦负责，也很感谢我的队友的积极配合。希望在以后的学习生活中，我们可以进一步合作，做出更好的项目。