**目录**

**摘要**..............................................................................................................................................I

**Abstract**.....................................................................................................................................IV

[1 绪论 3](#_Toc513209195)

[1.1 开发背景 3](#_Toc513209196)

[1.2 任务概述 3](#_Toc513209197)

[2 方案论证 3](#_Toc513209198)

[2.1 下位机控制芯片选择 3](#_Toc513209199)

[2.1.1 51单片机 3](#_Toc513209200)

[2.1.2 Stm32单片机 4](#_Toc513209201)

[2.2 上位机开发软件选择 4](#_Toc513209202)

[2.2.1 MFC 4](#_Toc513209203)

[2.2.2 Qt 4](#_Toc513209204)

[2.3 上下位机通信方式选择 4](#_Toc513209205)

[3 下位机系统设计 4](#_Toc513209206)

[3.1 系统总流程图 4](#_Toc513209207)

[3.2 电压参数获取 4](#_Toc513209208)

[3.3 电流参数获取 4](#_Toc513209209)

[3.4 温度参数获取 4](#_Toc513209210)

[3.5 时间参数获取 4](#_Toc513209211)

[3.6 注水控制 4](#_Toc513209212)

[3.7 串口通信 4](#_Toc513209213)

[4 上位机系统设计 5](#_Toc513209214)

[4.1 MVC设计模式 5](#_Toc513209215)

[4.2 Qt程序流程图 5](#_Toc513209216)

[4.3 信号与槽 5](#_Toc513209217)

[4.3.1 信号 5](#_Toc513209218)

[4.3.2 槽 5](#_Toc513209219)

[4.4 控件与布局管理 5](#_Toc513209220)

[4.4.1 Push Button 5](#_Toc513209221)

[4.4.2 Label 5](#_Toc513209222)

[4.4.3 Line Edit 5](#_Toc513209223)

[4.4.4 Combo Box 5](#_Toc513209224)

[4.5 QTimer类 5](#_Toc513209225)

[4.6 Qt第三方库之QcustomPlot 5](#_Toc513209226)

[4.7 串口通信 5](#_Toc513209227)

[4.8 鼠标坐标显示 5](#_Toc513209228)

[5 调试与分析 6](#_Toc513209229)

[5.1 下位机调试之J-Link 6](#_Toc513209230)

[5.2 上位机调试之QDebug 6](#_Toc513209231)

[5.3 结果分析 6](#_Toc513209232)

[5.4 误差分析 6](#_Toc513209233)

[6 总结与展望 6](#_Toc513209234)

[6.1 总结 6](#_Toc513209235)

[6.2 展望 6](#_Toc513209236)

谢辞............................................................................. ...........................................................5465

参考文献...................................................................................................................................121

基于单片机的热水器出厂性能指标自动检测装置设计（软件）

摘要

/\*此处省略一些字\*/

关键字： /\*此处省略一些字\*/

English Title

Abstract

/\*...\*/

Key words: /\*..\*/

1. 绪论
   1. 开发背景

/\*此处省略一些字\*/

* 1. 任务概述

/\*此处省略一些字\*/

1. 方案论证
   1. 下位机控制芯片选择

在本项目中，下位机的控制芯片主要需要完成一些GPIO口控制，串口通信，AD数据采集，数据处理等功能，而STM32芯片就能很好地实现上述功能。STM32xinpian 是由意法半导体(ST)开发的一款单片机，其主要特点是采用标准的ARM架构，内核为ARM公司为要求高性能，低成本，低功耗的嵌入式应用专门设计的Cortex-M内核。这使得STM32芯片拥有极高的性能，可以支持丰富的外设，功耗低，价格合理。而且STM32芯片也已经使用的非常广泛，开发者能够获得全面丰富的技术文档支持。STM32芯片为满足不同开发的需要，拥有着丰富的型号，开发者可以根据自己项目的大小需求，选择自己需要的性价比最高的型号，结合本项目的要求，我们选择了一款较为经济的STM3210f

* 1. 上位机开发软件选择
     1. MFC

MFC(Microsoft Foundation Classes)，是微软公司提供的库类，主要用于GUI(Graphical User Interface，图形用户界面)设计。它专用于Windows平台，或多或少使用了面向对象的方法包装了API(Application Programming Interface,应用程序编程接口)。但有时使用C，有时是用C++，甚至有时是两者的混合体。这就使得在使用起来存在内存消耗大的问题，例如有些API需要你在使用时提供一个包含10个成员的C语言的struct型参数，但其实可能只有一个是你想用到的参数，这与完全用C++编写的，拥有类，继承等方法的面向对象设计在内存占用方面存在着很大的差距。MFC采用的核心框架是事件驱动框架，要做出任何操作都必须对特定的消息做出相应，然而由于消息的数量庞大，导致我们很难在短时间内掌握并熟练应用这种框架。一个开发环境的帮助文档对开发者来说能起到很大的帮助作用，MFC拥有着内容丰富的帮助文档MSDN，但是需要单独付费购买，并且由于内容太过庞大（有10个CDROM光盘），导致刚入门的开发人员在使用文档方面会遇到很大的困难。

* + 1. Qt

Qt是一款跨平台的C++图形用户界面(GUI)应用程序设计的开发环境，它分为Qt4.x系列和Qt5.x系列。Qt4.x系类是由Nokia公司开发个较为经典的Qt版本，由于网上样例较多，适用于教学学习。Qt5.x系列是由Digia公司开发的较新的Qt版本，相比与Qt4.x版本添加了很多新内容，所以在开发时显得更加方便。Qt最主要的优点就在于它强大的跨平台能力，它同时支持Windows，Mac OS X，Unix，Linux等多个平台。这使得开发人员只需要在某个平台上进行开发，就能轻松地移植到各个其他的平台上去，减少了开发人员的工作量，也间接地提高了公司的效率，大大缩短了研发周期。这一点受到很多公司的赞许，所以Qt已经成为现在主流的图形用户界面开发环境。Qt采用面向对象的设计思想，提供给我们很多便于使用的API接口，使得我们在开发时更加简单方便。它的信号与槽的消息机制可以说是它的灵魂之一，利用SIGNAL()和SLOT()函数能够对各个种类的信号做出对应的槽相应，这使得它能绝大部分完成用户的开发要求。Qt的另一大亮点在于它详细而不冗杂的帮助文档，它覆盖了Qt的方方面面但却只有18M大小，这对于Qt开发的新手来说无异于是一种福音。帮助文档也是开发者们选着开发环境的一个重要指标。

通过2.2.1与2.2.2的比较，我们发现Qt在跨平台，内存占用，和易学性方面都优于MFC，而Qt5.x又相对于Qt4.x更便于开发，所以本次最终选择Qt5.x作为上位机的开发环境。

* 1. 上下位机通信方式选择

本次上下位机之间的通信方式采用的是串口通信，串口通信相对于通过互联网通信在硬件实现简单，成本低，容易实现，不存在设备间的相互干扰，通信速度快等方面表现出了一定的优越性。综合考虑到本项目的通信需求和开发成本，最终选择了串口通信作为上下位机的通信方式。

1. 下位机系统设计
   1. 系统总流程图

/\*此处省略一些字\*/

* 1. 电压参数获取

/\*此处省略一些字\*/

* 1. 电流参数获取

/\*此处省略一些字\*/

* 1. 温度参数获取

/\*此处省略一些字\*/

* 1. 时间参数获取

/\*此处省略一些字\*/

* 1. 注水控制

/\*此处省略一些字\*/

* 1. 串口通信

/\*此处省略一些字\*/

1. 上位机系统设计
   1. MVC设计模式

MVC是Model（模型）- View（视图）-Controller（控制器）的缩写，这是一种设计模式，主要目的是很好地将界面和业务逻辑分离。

Model是程序的主体部分，它表示业务数据和业务规则。Model拥有最多的处理任务，并且它返回的数据是中立的，也就是说它与数据格式无关，这个特点使得一个Model可以为多个View提供数据处理。由于一个Model代码能被多个View使用，这减少了代码的重复性，减轻了开发的任务量。

View是程序与用户的交互接口，用户可通过此层输入要处理的数据，并经过另外两部分的处理后，将最终的结果显示在View部分。这一部分让程序使用者拥有一个直观的图形用户界面，方便操作而不需要关心内部的代码执行过程。

Controller是负责转发请求，对请求进行处理的部分。它相当于一个中转站，接收View传入的数据，经过一定的处理（如数据格式处理等）后传给Model，而后Model处理完成的结果再通过Controller处理后传递给View，最终显示在图形用户界面上

总的来说，采用MVC模式开发存在着很多优点。首先它有利于开发中的分工，在大型开发项目中将Model，View，Controller分别交给不同的开发人员设计，最终再整合成整个项目。其次它有利于组件的重用，例如同一个Model就能被不同的项目所应用。最后，它能让三个层次各施其职互不干扰，当项目要改动Model，View，Contoller当中的某个部分时，并不需要改动其他部分，这大大有利于项目的开发。所以，在本次项目中也采用了MVC这个优秀的模型。

* 1. Qt程序流程图

/\*此处省略一些字\*/

* 1. 信号与槽

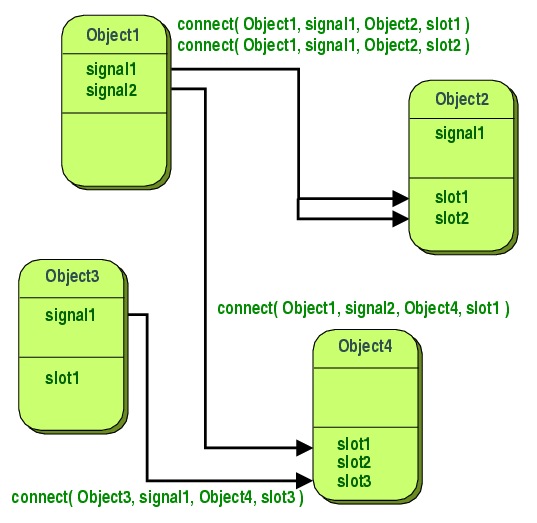
信号与槽用于两个对象之间的通信，信号与槽机制是Qt的核心特征，也是Qt不同于其他开发框架的最突出的特征。在GUI编程中，当改变了一个部件时，总希望其他部件也能了解到该变化。更一般的来说，我们希望任何对象都可以和其他对象进行一定的通信。例如，如果用户点击了关闭按键，则希望可以执行窗口的close()函数来关闭当前窗口。为了实现对象间的通信，一些工具包中使用了回调(callback)机制，而在Qt中使用了信号与槽来进行对象间的通信。当一个特殊的事件发生时便可以发射一个信号，比如按键被单击；而槽就是一个函数（为了与一般函数区分，故称为槽函数），它在信号发射后被调用来响应这个信号。Qt的部件类中已经定义了一些信号与槽，但是更多的做法是子类化这个部件，然后添加自己的信号与槽来实现开发者自己想要的功能。信号与槽的关联可以是一对一，也可以一个信号关联到多个槽上，多个信号也可以关联到同一个槽上，甚至，一个信号还可以关联到另一个信号上，信号与槽的关系如图4-1所示。如果存在多个槽与某个信号相关联，那么，当这个信号被发射是，这些槽将会一个接一个地执行，但是执行的顺序是随机的，无法指定它们的执行顺序。

图 1 信号与槽的链接

* 1. 控件与布局管理
     1. QPush Button

QPushButton控件提供了一个标准按键，这个标准按键是在所有GUI窗口中用的最多的控件，用户可以通过按下（点击）这个按键去让电脑执行一些列的操作或是回答一个问题。最典型的就是“确认”，“取消”等按键。QPushButton可以通过setText(const QString &text)来改变按键所显示的文字，并且当我们按下按键，我们能收到该控件发出的clicked信号，通过对开发者自定义的槽函数的绑定，我们就能实现很多想要的功能。

* + 1. QLabel

Qlabel控件提供一个文本或者图片的显示，我们可以通过很多方法来设置它的视觉效果，其主要功能是为其他控件添加一个说明。通常我们用setText(const QString &)函数来设置该控件文本的内容。

* + 1. QLineEdit

QLineEdit控件是一个一行的文本编辑器，它允许用户去输入或编辑一个单行的纯文本，这个纯文本可以被对应的处理函数所接收。当然，QLineEdit编辑器自带复制，粘贴，撤销等功能。通过调用它的echoMode()函数，我们可以将它改为只写模式，即输入的纯文本不会在编辑器上回显，这样我们就可以把它作为密码输入栏。QLineEdit拥有着各种像maxLength()，minLength()这样的函数来对它自身的属性进行设置，这样完全能够满足用户的个性化需要。在程序内部，我们也可以通过setText()函数来改变文本内容，而不是单纯的只有用户输入。

* + 1. QComboBox

QComboBox控件结合了按键和弹出列表功能，它提供一个可选择的列表选项。我们能够方便地通过insertItem()函数和setItemText()函数来添加自己想要的列表选项。当然，QComboBox还提供了像currentIndexChanged()，activated()，highlighted()，editTextChanged()等多种信号，以使得开发者能够清楚地感知到用户的各类操作，从而链接到各个自定义的槽函数当中，以实现各种各样的功能。

* + 1. 布局

上面我们讲述了一些基本的窗口部件，我们可以在图形化用户界面的任意位置放置这些部件，这并不会对部件的使用产生过多的影响，但是，对于一个完善的软件，布局管理却是必不可少的。无论是想要界面中部件有一个很整齐的排序，还是想要界面能适应窗口的大小变化，都要进行界面布局。

Qt的布局管理系统提供了简单而强大的机制来自动排列一个窗口中的部件，确保它们有效地使用空间。QLayout类及其子类便是Qt提供的布局管理器，它们可以实现常用的布局管理功能。QLayout类中几个总要的子类分别是QBoxLayout（本布局管理器），QgridLayout（栅格布局管理器），QFormLayout（表单布局管理器）。

基本布局管理器QboxLayout类可以使子部件子在水平方向或者垂直方向排成一列，将所有的空间分成一行盒子，然后将每个部件放入一个盒子中。它有两个子类QHBoxLayout水平布局管理器和QVBoxLayout垂直布局管理器，在编程中经常用到。

栅格布局管理器QGridLayout类使得部件在网格中布局，它将所有的空间分隔成一些行和列，行和列的交叉处就形成了单元格，然后将部件放入一个确定的单元格中。

表单布局管理器QFormLayout类用来管理表格的输入部件及其相关的标签，将它的子部件分为两列，左边是一些标签，右边是一些输入部件，比如行编辑器或者数字选择框等。

* 1. QTimer类

在单片机的程序编写中，定时器中断可以说是不可或缺的一部分，它能让我们完成很多与时间相关的任务。在Qt中也同样存在着用于完成类似任务的QTimer类，QTimer类提供了定时器信号和单触发定时信号。它的使用也非常简单，首先创建一个QTimer对象，再使用connect()函数来使它的timeout()信号链接到适当的槽函数，最后用start()函数开始这个定时器即可。这样，每当设置好的定时时间走完，定时器对象都会发出一个timeout()信号，之后再执行绑定好的槽函数，并重新开始新的定时。

实例：

QTimer \*timer = new QTimer(this);

connect(timer, SIGNAL(timeout()), this, SLOT(update()));

timer->start(1000);

* 1. Qt第三方库之QCustomPlot

Qt本身已经拥有十分强大的功能，但由于各个行业需求的不同，将所有功能都直接添加到Qt软件中去是十分不明智的选择，因为对于某个行业的开发者来说，其他行业需要的功能可能他几乎用不到。所以，Qt通过第三方库的概念，允许各个行业的开发者在Qt软件中添加各种自己需要的个性化库。

QCustomPlot即是Qt的一个常用的第三方库，它主要是用于绘制图表使得数据可视化，能够制作出漂亮的2D图，像曲线图，柱状图，饼状图等。它的存在让实时可视化应用程序有了高性能的服务，并且，我们还可以将QCustomPlot获得的坐标图以PDF或者是位图（如PNG，JPG，BMP）的方式保存下来，以方便再一次用作数据分析。

对于第三方库QCustomPlot的使用，首先得在工程项目中添加qcustomplot.h，qcustomplot.cpp两个文件，其次要在工程的.pro文件中加入QT += widgets printsupport语句，最后便可以将.ui文件中的Qwidget对象提升为QcustomPlot对象，在程序中便可对这个对象进行相应的操作。（上述流程如图x所示）

* 1. 串口通信

串口通信在Qt4.x版本需要通过添加第三方库的形式才能使用，而Qt5.x版本则直接为软件添加了这项功能，可见串口通信确实是比较常用的通信方式。

Qt5.x自带了QSerialPort类用于串口通信，使用时需要在工程的.pro文件中添加一行代码QT += widgets printsupport，然后直接引用头文件<QtSerialPort/QSerialPort>和<QtSerialPort/QserialPort>。其中前者是提供访问串口的功能，后者是提供系统中存在的串口信息。接下来就是创建一个QSerialPort的对象，对串口的名称、波特率、数据位、校验位、停止位等参数进行设置，然后才进行串口读写操作。（上述流程如图x所示）

* 1. 鼠标相对坐标显示

鼠标相对坐标显示功能就是将鼠标所在的相对于QCustomPlot对象坐标轴的坐标位置显示出来（如图x），这个功能的实现可以分为三个步骤。

步骤一，绑定mouseMove(QMouseEvent\*)信号与自定义的槽函数，在槽函数中调用QMouseEvent类的成员函数pos().x()和pos().y()以获得全局坐标。

步骤二，用pixelToCoord() 函数将全局坐标转化为对于QCustomPlot对象坐标轴的相对坐标。

步骤三，对数据做一定的处理（如格式，保留位数），然后打印在界面上。

代码实现如下

1. 调试与分析
   1. 下位机调试之J-Link

/\*此处省略一些字\*/

* 1. 上位机调试之QDebug

/\*此处省略一些字\*/

* 1. 结果分析

/\*此处省略一些字\*/

* 1. 误差分析

/\*此处省略一些字\*/

1. 总结与展望
   1. 总结

/\*此处省略一些字\*/

* 1. 展望

/\*此处省略一些字\*/

谢 辞

/\*此处省略一些字\*/

参考文献