基于超声波的空气污染物浓度测量系统研制

摘要

阅读并翻译相关文献，从理论上对使用超声波传感器进行空气污染物浓度测定的可行性进行分析。基于超声传播在不同浓度的混合气体中具有微小时差的特性，设计了空气污染物浓度检测系统。该系统选用瑞士盛世瑞恩(SENSIRION)公司的SHT10数字温湿度传感器测量空气的温度和湿度，使用超声传感器测量超声波在一段定长空气中传播的时间。同时该系统选用恩智浦(NXP)半导体公司的32位基于ARM Cortex-M0内核的LPC1114微控制器对传感器获得的温度，湿度和时间数据进行采集，并将这些数据以特定格式封装成帧并使用蓝牙芯片通过无线通信方式将数据帧发出。该系统使用自己编写的基于安卓的手机程序，通过调用手机自带的蓝牙模块对数据进行接收。手机程序对接收的数据帧进行解帧，并对解帧后的数据进行处理，将结果显示在手机界面上。整个项目使用Git以及Subversion作为版本控制工具。

关键词：空气污染物浓度，超声波，微控制器，无线通信，安卓，版本控制

The Development of Air Pollutants’ Concentration- Measuring System Based on Ultrasonic

ABSTRACT

Reading and translating relevant documents and materials, analyze the feasibility of applying ultrasonic sensor to measure the concentration of air pollutants in theory. Design the air pollutants’ concentration-detecting system based on the characteristic that there will be minor time difference when ultrasonic spreads in mixed gas of different concentration. This system makes use of SHT10 digital temperature and humidity sensor manufactured by SENSIRION Company in Switzerland to measure temperature as well as humidity of air and measures the time of ultrasonic spreading in the air of selected length by using ultrasonic sensor. At the same time, this research chooses NXP Company’s LPC1114 32-bit ARM Cortex-M0 microcontroller to collect temperature, humidity and time data gained from ultrasonic sensor, later packages these data into frames with certain format and eventually sends out those data frames through wireless communication by using a Bluetooth chip. The system takes advantage of self-written mobile-phone application based on Android OS to receive data by calling Bluetooth module that exists initially in phones. Mobile-phone applications then de-frame and process the received data frames, and display results in the interface of mobile-phones. The whole project uses Git and Subversion as the version controlling tools.

Key words: air pollutants’ concentration, ultrasonic, microcontroller, wireless communication, Android OS, version control

1 引 言

* 1. 研制空气污染物浓度检测系统的意义

随着社会经济的发展和人们生活水平的提高，城市环境空气质量也越来越成为人们普遍关注的热点话题，因而对城市大气环境质量做出客观、全面、实时的认识和评价是极其必要的。

空气污染指数(air pollution index, API)是反映大气环境质量水平的重要标准。根据环境空气质量标准和各项污染物对人体健康和生态环境的影响，将常规检测的集中空气污染物浓度简化为单一的概念性指数值形式，它将空气污染程度和空气质量状况分级表征，适合于表示城市的短期空气质量状况和变化。

* 1. 空气污染物浓度检测的常用方法

1.2.1 利用电阻式气敏元件测量气体浓度

电阻式气敏元件经过一段时间的预热后，其阻值与周围气体浓度有着确定的关系。因此，只需测出运行状态下气敏元件的电阻值，便可推算出当前运行状态下待测气体的浓度。这是气体浓度测量中最常用的传统方法。

利用气敏元件测量气体浓度原理简单，操作也比较方便，但存在以下不足：(a)由于在检测仪中一般将气敏元件与标准原件组成测量电桥电路，因此电桥电路的非线性会影响测量精度；(b)电桥供电电压的大小对测量精度有很大影响；(c)检测时还需考虑现场温度、空气扰动等因素，需要传接上补偿电路。所以这种方法主要用在测量精度不高的场合。

1.2.2 气相色谱法测量气体浓度

气相色谱法是基于不同的气体在通过色谱柱时速度不同的原理，主要适用于工厂生产现场的检测、设备检修过程中动力作业的安全检测。该方法对混合气体进行多次采样，注入色谱仪直至色谱峰值全部出现后再对其进行色谱分析。很多地方也称其为光谱测量法。不同浓度的混合气体的色谱存在明显差异，根据当前呈现的色谱便能分析出当前气体浓度的高低。这种检测方法测量方便、快捷、线性范围宽、灵敏度高、可靠性重现性号，在气体浓度检测中占有很重要的地位，但在测量中必须合理地选择色谱柱及载气流速等参数，只有进行多次重复实验后方能获得理想的测量效果。

1.2.3 载体催化燃烧法检测气体浓度

这种检测方法线性和稳定性较好。以爆炸下限百分体积浓度为单位的浓度标度方法能统一衡量各种可燃气体浓度所呈现出的爆炸危险度，而且量程符合工业要求，故常被用于爆炸危险场所可燃性气体的测爆。该检测方法以催化载体型气敏元件作为浓度的传感器，该元件由在铂丝上烧结一层陶瓷载体后，再涂覆催化活性物（Rh,Pd等）构成。当铂丝中通以工作电流使之达到临界反应温度（320~350℃）时，可燃气就在元件表面催化燃烧，铂丝电阻增加，在完全燃烧且热辐射可忽略时，电阻增量ΔR与可燃气体浓度C成正比，将ΔR转换成电信号，即可用于检测可燃气体的浓度。检测中一般将催化元件（检知元件）及与之配对的参比元件组成的电桥，作为浓度信号采样单元。由于催化元件的气敏特性除与可燃气体浓度有关外，还受到工作电流、环境的温度和湿度、气压的影响，所以在设计过程中应注意采用桥式单元或其他参比元件对其予以补偿。

1.2.4 光干涉法测量气体浓度

光干涉法是一种创新的检测方法，它基于同一光源发出的光波会被平面镜分为2束光，经不同路线后又汇集成一束光，发生光干涉，产生干涉条纹的物理现象。光干涉式气体浓度测量器目前被广泛用于煤矿井下空气中的甲烷、瓦斯等浓度，以及其他环境下各种气体浓度的测定；在实验室也往往使用这种测定仪器作为相对参考的仪器。用该方法测定气体浓度时，必须考虑到气体的温度、压力、含量、换算结果，以及光在该气体中的折射率等问题[1]。

* 1. 超声技术测量气体浓度

使用超声测量气体浓度是近10年来由于电子技术和测量技术的发展而出现的一种新技术。气体浓度的超声检测法为非接触式测量，具有测量范围宽、测量精度高等特点，而且测试设备体积小，无需维护，使用寿命长，在计量检测领域收到极大关注。近年来，超声技术作为一种新型的检测技术已广泛应用于气体浓度检测领域。国内外一些公司经过大量研究，已将超声技术成功地应用于气体浓度的高精度检测装置中，实现了气体浓度检测技术的重大突破[2]。

* 1. 本文所作的工作

本文首先从理论上分析利用超声波测量气体浓度的原理，通过数学推导验证超声法对气体污染物浓度进行监测的可行性。并从以下几个方面介绍毕业设计空气污染物浓度监测系统所做的工作:

(1) 对于LPC1114微控制器底层寄存器的配置，包括32位计时器的匹配输出与捕获输入，利用IO口模拟应用于温湿度传感器SHT10的两线串行接口(DATA总线与CLK总线)；

(2) 硬件电路的设计，包括原理图设计和印刷电路板(PCB)图设计；

(3) 基于安卓的手机上位机程序设计，包括使用设计模式中的依赖倒置原则；

(4) 版本控制，包括程序工程进行版本控制的工具(Git)和对电路工程进行版本控制的工具(Subversion)；

(5) 实验结果。

1. 理论部分

2.1 浓度声速理论

将空气简化为二元混合气体（普通空气与微量污染物气体），在常温常压实验条件下可利用理想气体模型对该气体进行分析。混合气体平均速度为[2]：



式中为混合气体的平均速度；为混合气体平均定压定容比热比；*P*是气体压力；是气体密度。

使用理想气体方程：



式中为气体体积；为通用气体常数，也称普适气体恒量，；T为气体的热力学温度；为气体的质量；为气体的平均相对分子质量。

将式(2.2)代入式(2.1)中可得：



对于二元混合气体，设其两种组分分别由下标组成，有：





式中为类气体的浓度，为类气体的定容热比，分别为两种气体的相对分子质量。

根据式(2.3)，令参数*Y*为：



由以上等式整理可得：



其中，系数均为参数*Y*的函数。；； 。

由于，所以方程(2.5)有单根，其解为：



通过测量出的声速和温度两个量，可得出参数的值，并由此计算出方程的3个系数，代入式(2.6)，求出浓度n值[3]。

2.2 嵌入式系统相关理论

本毕业设计使用的微控制器型号为恩智浦半导体公司基于ARM 架构Cortex-M0内核的32位低成本微控制器LPC1114。该款微控制器的外设组件包括：64KB片内FLASH程序存储器，8KB片内SRAM，一路I2C总线接口，一路RS-485/EIA-485 UART，两路SPI总线接口，四个通用定时/计数器以及多达42个通用I/O口[4]。

LPC1114 Cortex-M0微控制器具有以下特性：

1. Cortex-M0内核，工作频率高达50MHz；
2. 内置嵌套向量中断控制器（NVIC）；
3. 高达32KB片内Flash程序存储器；
4. 高达8KB片内SRAM存储器；
5. 在系统编程（ISP）和在应用编程（IAP）可通过片内引导装载程序软件实现；
6. 串行接口包括：

--可产生小数波特率、具有调制解调器、内部FIFO和支持RS-485/EIA-485标准的UART；

--SSP控制器，带FIFO和多协议功能（仅在LQFP48和PLCC44封装中有两路SSP）；

--I2C总线接口，完全支持I2C总线规范和快速模式，数据速率为1Mbit/s，具有多个地址识别功能和监控模式。

1. 其它外设：

--多达42个通用I/O（GPIO）引脚，带可配置的上拉/下拉电阻；

--P0.7引脚支持20mA的高驱动电流；

--I2C总线引脚在FM+模式下可支持20mA的灌电流；

--4个通用定时器/计数器，共有4路捕获输入和13路匹配输出；

--可编程的看门狗定时器（WDT）；

--系统节拍定时器。

1. 带有SWD调试功能；
2. 集成了电源管理单元（PMU），可在睡眠、深度睡眠和深度掉电模式中极大限度地减少功耗；
3. 具有三种低功耗模式：睡眠模式、深度睡眠模式和深度掉电模式；
4. 3.3V单电源供电（2.0V~3.6V）；
5. 10位模数转换ADC，在8个引脚中实现输入多路复用；
6. GPIO均可以配置为边沿或电平中断；
7. 带驱动的时钟输出功能可以反映主振荡器时钟、IRC时钟、CPU时钟和看门狗时钟；
8. 13个起始逻辑功能可以将CPU从深度睡眠模式中唤醒；
9. 掉电检测，具有4个独立的阀值，用于中断和强制的复位；
10. 上电复位（POR）；
11. 主振荡器工作范围：1Mhz~25MHz；
12. 12MHz内部RC振荡器可调节到1%的精度，可将其选择为系统时钟；
13. 锁相环PLL允许CPU在最大的CPU速率下操作，而无需高频晶振，可从主振荡器、内部RC振荡器或从看门狗振荡器中运行；
14. 可采用LQFP48、PLCC44或HVQFN33封装。

2.3 Android操作系统及JAVA编程相关知识

2.3.1 Android操作系统

Android，中文俗称安卓，是一个以Linux为基础的开放源代码移动设备操作系统，主要用于智能手机和平板电脑，由Google成立的Open Handset Alliance（OHA，开放手持设备联盟）持续领导和开发中。Android已发布的最新版本为Android 5.1.1（Lollipop）。

Android系统最初由安迪·鲁宾（Andy Rubin）等人开发制作，最初开发这个系统的目的是创建一个数码相机的先进操作系统；但是后来发现市场需求不够大，加上智能手机市场快速成长，于是Android被改造为一款面向智能手机的操作系统。于2005年8月被美国科技企业Google收购。2007年11月，Google与84家硬件制造商、软件开发商及电信营运商成立开放手持设备联盟来共同研发改良Android系统，随后，Google以Apache免费开放源代码许可证的授权方式，发布了Android的源代码，让生产商推出Android的智能手机，Android操作系统后来更逐渐拓展到平板电脑及其他领域上。

2010年末数据显示，仅正式推出两年的Android操作系统在市场占有率上已经超越称霸十年的诺基亚Symbian系统，成为全球第一大智能手机操作系统[5]。

2.3.2 JAVA程序设计理论

Java是一种计算机编程语言，拥有跨平台、面向对象、泛型编程的特性，广泛应用于企业级Web应用开发和移动应用开发。

Java编程语言的风格十分接近C++语言。继承了C++语言面向对象技术的核心，Java舍弃了C++语言中容易引起错误的指针，改以引用取代，同时移除原C++与原来运算符重载，也移除多重继承特性，改用接口取代，增加垃圾回收器功能。在Java SE 1.5版本中引入了泛型编程、类型安全的枚举、不定长参数和自动装/拆箱特性。太阳微系统对Java语言的解释是：“Java编程语言是个简单、面向对象、分布式、解释性、健壮、安全与系统无关、可移植、高性能、多线程和动态的语言”。由于JAVA是面向对象编程的语言，因此在编程中需要使用设计模式的理论作为指导进行编程。

2.3.3 设计模式

Christopher Alexander说过：“每一个模式描述了一个在我们周围不断重复发生的问题，以及该问题的解决方案的核心。这样，你就能一次又一次地使用该方案而不必做重复劳动。”尽管Alexander所指的是城市和建筑模式，但他的思想也同样适用于面向对象设计模式，只是在面向对象的解决方案里，我们用对象和接口代替了墙壁和门窗。两类模式的核心都在于提供了相关问题的结局方案[6]。

2.4 版本控制相关知识

本毕业设计仅对保存源代码的文本文件和PCB设计文件作版本控制管理，而实际上，你可以对任何类型的文件进行版本控制。

如果你是一位图形或网页设计师，可能会需要保存某一副图片或页面布局文件的所有修订版本[7]。采用版本控制系统（VCS）是个明智的选择。有了它你就可以将某个文件回溯到之前的状态，甚至将整个项目都回退到过去某个时间点的状态。你可以比较文件的变化细节，查出是谁最后修改了什么地方从而造成某些怪异问题，又是谁在何时报告了某个功能缺陷，等等。使用版本控制系统通常还意味着，就算你胡来搞砸了整个项目，把文件改的改，删的删，你也可以轻松恢复到原先的样子。而由此额外增加的工作量确微乎其微。版本控制系统分本地版本控制系统，集中化的版本控制系统和分布式的版本控制系统。

在本次毕业设计中，我使用了Git作为程序源代码工程的版本控制系统，又使用了Subversion作为基于Altium Designer软件设计的PCB电路工程的版本控制系统。

1. 需求分析与系统架构
   1. 需求分析

（1） 由于需要测量空气中的温度，湿度和超声波在声音中传播的速度，故需要设计硬件电路板，可以实现上述采集数据的功能；

（2） 需要设计基于LPC1114微控制器的程序，能控制数据采集的时序，并将采集回来的程序进行装帧并通过蓝牙串口发出；

（3） 在接收端需要设计基于Android OS的手机上位机，将单片机从蓝牙串口发出的数据进行接收并进行处理；

（4） 整体设计应该足够小，另外应该具有尽量低的功耗，这样可以满足采用电池供电的需求。

3.2 系统架构

3.2.1 硬件设计框架

对于温度和湿度的测量，该系统采用瑞士的盛世瑞恩公司的SHT10数字式温湿度传感器，该款传感器在环境温度25℃和供电电压3.3V的条件下测湿精度在±4.5%RH，测温精度在±0.5℃，基本可以满足系统对于温度和湿度测量的要求。SHT10传感器的接口为两线制串行接口，接口说明见下图。



图3.1 SHT10温湿度传感器典型电路

SHT10的串行接口，在传感器信号的读取及电源损耗方面，都做了优化处理；但与通用I2C接口不兼容[8]，因此，需要使用两个通用I/O口模拟串行时钟输入接口（SCK）和串行数据接口（DATA）。

对于超声波速度的测量，则需要使用单片机的计时功能。LPC1114提供了两种计时方式，一种是使用系统节拍定时器，最快可以每10毫秒产生一个中断，通过中断进行计时。另一种是使用微控制器内部的2个32位和2个16位的可编程定时器/计数器，最快可以以系统时钟频率（8MHz）的速度进行计时。

因为对于超声波在一小段距离内的计时要求非常精确，且激发超声波的信号频率在180kHz，故在这里采用硬件计时方式，即使用定时器/计数器进行计时。

超声接收部分的输出信号为180kHz的正弦波包络，为了计时，将正弦包络

由于最后单片机需要将数据发送到基于Android OS的手机终端，故还需使用微控制器的UART模块，通过UART将数据发送到蓝牙芯片上，再从蓝牙芯片发送到手机。

由以上条件可知，硬件电路的设计应基于LPC1114单片机，围绕单片机的GPIO，TIMER和UART通讯建立。



图3.2 硬件设计框架

3.2.2 软件设计架构

软件可以看两部分：前端测量模块，由LPC1114单片机和传感器电路组成，主要完成温湿度与超声波传播时间信号的采集和数据的无线传输；终端接收模块，由运行在Android OS手机上的上位机组成，主要负责完成蓝牙数据的接收，解帧和处理。

图3.3 软件设计框架

考虑到程序的可扩展性，在此使用分层的思想对数据帧进行设计。底层帧为帧头，帧长度，数据域和校验位；而数据域内又分命令字位和数据位。底层帧结构如下图所示：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 帧头1(0xAA) | 帧头2(0xBB) | 帧长度 | 数据域 | 校验字 |

图3.4 底层数据帧格式

数据域分命令字和数据位，命令字由一个字节构成，考虑到在监测过程中需要测量环境温度，环境湿度和超声波声时三个数据，故令该字节从低到高的三位为有效位，分别表示是否使能测量温度，湿度和声时。相应位置1说明使能了对应数据的测量。数据位则依次填入温度，湿度和声时的数据，每个数据值占两个字节，若未进行测量，则该数据省略，具体格式如下图：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 命令字(1 Byte) | 温度1 | 温度2 | 湿度1 | 湿度2 | 声时1 | 声时2 |

图3.5 数据域（上层数据帧）格式

* 1. 各模块设计思路

3.3.1 温湿度传感器模块设计思路

温湿度传感器SHT10的硬件设计比较简单，可以直接使用用户手册里的参考电路进行应用。同时，可在上拉电阻边串联一个led，通过led的亮灭状态判断数据总线上是否有数据通过。

由于SHT10为一款数字式传感器，采用串行接口进行通信，但SHT10的串行接口与I2C通用接口不兼容，因此需要使用通用I/O对串行接口进行模拟。具体原则为DATA总线在SCK时钟下降沿之后改变状态，并仅在SCK时钟上升沿有效。数据传输期间，在SCK时钟高电平时，DATA必须保持稳定。为避免信号冲突，微处理器应驱动DATA在低电平，即MCU不输出数据时DATA应为高电平。因此，需要一个外部的上拉电阻（例如：10kΩ）将信号提拉至高电平。

3.3.2 超声波传感器模块实现计时功能设计思路

超声波传感器是利用超声波的特性研制而成的传感器。在输入端，由换能晶片在180kHz脉冲输入电压的激励下发生震动产生超声波；在输出端，换能晶片再将接收到的超声波信号转换为频率为180kHz的正弦波包络。因此，该模块的硬件设计包括将单片机输出的脉冲信号的峰峰值放大到10V和输入端将1mV左右的信号放大并对放大之后的信号进行电压过零比较。

软件设计思路为输入端计数器采用匹配输出，经过实测可得当把计数器的匹配寄存器配置为132时输出的矩形脉冲为180kHz，在匹配输出的同时使能另一个计数器的捕获输入功能，当超声波传感器发出的正弦波经过放大和过零比较为矩形脉冲波输入到计数器引脚时，计数器产生中断将捕获寄存器中的数取出即为发送到接收经历的时间。

3.3.3 蓝牙接收端模块程序设计思路

通过蓝牙接收数据的程序是基于Android OS的手机开发的。使用主线程（UI线程）实现手机屏幕中控件的显示，并新建一个线程实现对数据的接收。在JAVA中实现多线程有两种途径：继承Thread类或者实现Runnable接口。但在程序开发中只要是多线程肯定永远以实现Runnable接口为主，因为实现Runnable接口相比继承Thread类有如下好处：

1. 避免点继承的局限，一个类可以继承多个接口；
2. 适用于资源的共享。

将接收的字节型数据进行处理即可得到目标数据。

1. 硬件设计与开发

4.1 温湿度模块的硬件设计与开发

由上文所述，本毕业设计使用的测量温湿度的传感器为瑞士的盛世瑞恩(SENSIRION)公司的SHT10数字温湿度传感器。其优点为精确度高（在25℃时测温精度为±0.5℃，测湿精度为±4.5%RH），集成性高（内部集成了测量模块，AD模块以及串行接口电路）。其外围电路原理图设计如下：

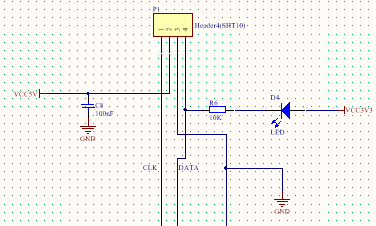


图4.1 空气温度湿度测量硬件电路

需要注意的是由于DATA（数据）总线需要微控制器与传感器交替控制，为避免信号冲突（例如前一个时刻MCU将总线拉低下一时刻传感器需要将总线拉高，由于负载能力不够造成输出的逻辑电压在位置状态），需要一个外部的上拉电阻。同时，为了在调试时方便观测DATA总线上是否有数据通过（即总线上是否有高低电平交错），在上拉电阻上可串联一个LED，当DATA总线上的逻辑电压为低电平时LED阳极与阴极存在正向压降LED就会发亮，当DATA总线上的逻辑电压为高电平时LED阳极与阴极不存在正向压降LED就会熄灭。

4.2 蓝牙模块的硬件设计与开发

本检测系统使用蓝牙传输方式与安卓上位机进行无线通信。其中，使用的通信模块为HC05蓝牙串口模块，这是一款主从一体式的蓝牙模块。其优点在于可以忽略蓝牙内部的通信协议，直接将蓝牙当作串口来使用。因此，只需将微控制器的发送端与蓝牙模块的接收端，微控制器的接收端与蓝牙模块的发送端相连接就可以完成硬件电路的开发。蓝牙模块的硬件电路原理图如下图：

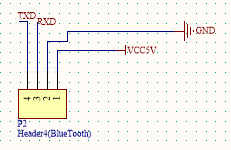


图4.2 蓝牙通信模块硬件电路

* 1. 超声波模块的声时测量硬件设计与开发

4.3.1 超声波模块发射端硬件电路设计

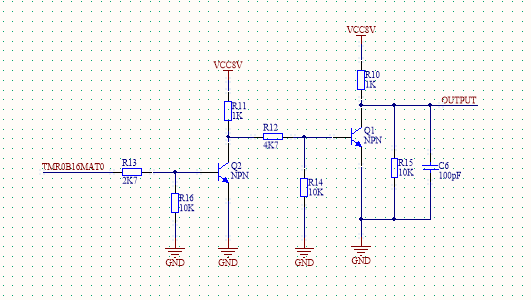
超声波模块发射端的主要作用就是将频率为180kHz的矩形脉冲波通过换能器转换为超声波发出。由于超声波模块没有外部供电，矩形脉冲波所需要提供的功率应尽可能大。通过信号发生器试验可得，当矩形脉冲的峰峰值高于8V时超声波模块的接收端可得到比较好的波形。故硬件电路的主要作用为将微控制器输出的频率180kHz，峰峰值为3.3V的脉冲信号转换为频率为180kHz，峰峰值为8V的脉冲信号。具体的实现主要利用了三极管处于截止区与饱和区的特性。具体原理图如下：

图4.3 超声波模块发射端硬件电路设计

矩形脉冲信号从三极管的基极输入，从三极管的集电极输出，若输入信号的逻辑电压为低电平，此时三极管处于截止区，集电极输出的逻辑电压为高电平；若输入信号的逻辑电压为高电平，此时三极管处于饱和区，集电极输出的逻辑电压为低电平。为了让输入与输出的电压同相，可使用两级三极管进行开关电路的设计。

4.3.2 超声波模块接收端硬件电路设计

当间隔一定的时间往超声波模块的输入端发出一组频率为180kHz的脉冲后，超声波的输出端就会产生如下所示的正弦包络。其中，包络内部的高频信号频率也为180kHz。高频信号的起始位置就是超声波从信号输入端到达输出端的时间。

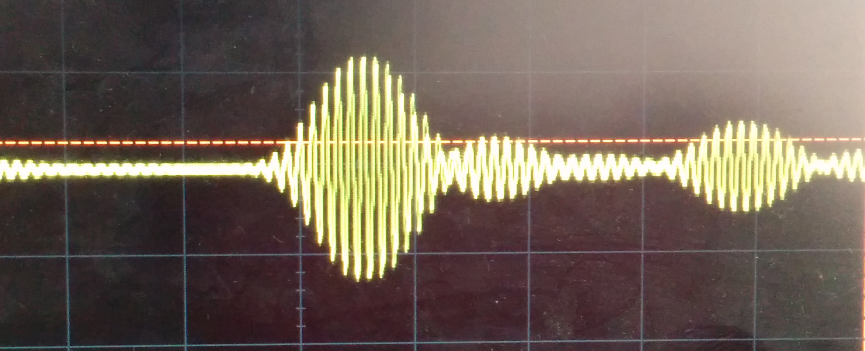


图4.4 超声波模块输出信号

因此超声波模块接收端硬件电路需要完成的任务为捕捉到输出信号的起始位置。由于输出信号的幅值较小，故首先需要将信号进行放大。由于输出信号具有频率高（180kHz），功率低（接收端仍然没有外部供电，依靠捕获超声波的能量输出信号）的特性，在这里我们选用精密运算放大器AD8032。这是一款精密运放同时具有80MHz的单位增益带宽[9]，应用在改电路是合适的。将信号放大后可利用运放在理论上开环增益无穷大的特性制作电压过零比较器。在实际测试中发现若单独使用电压比较器的话前几个小信号的输出不能到达饱和，因此在比较器的输出端再接一个电压比较器，即可达到微控制器捕获输入所需的电平需求，具体原理图如下：

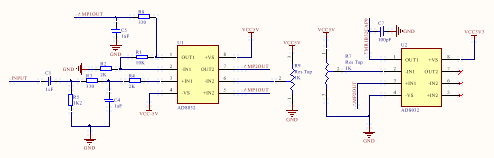


图4.5 超声波输出模块硬件电路图

经过该电路的输出波形如下：

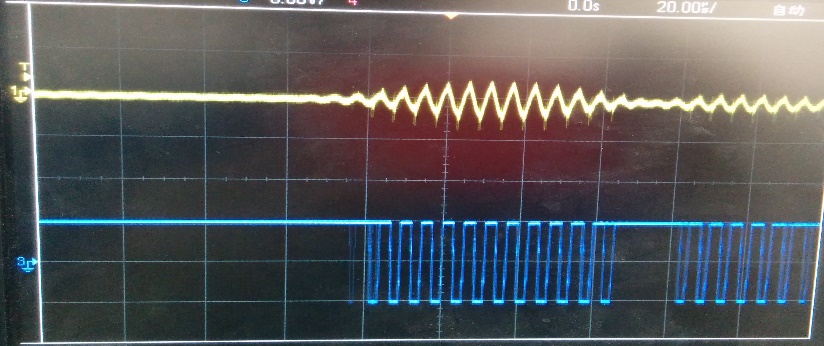


图4.6 超声波模块输出信号(黄)与经过处理后的输入微控制器的TTL信号(蓝)

4.4 印刷电路板(PCB)的设计与开发

根据 Altium Designer 软件自动生成的BOM(Bill of Material)表可得，用于本项目的硬件电路总计有50个电子元件。因此，使用双面板进行电子元件的摆放与布线。印刷电路板主要依照抗干扰设计原则进行设计，同时需要兼顾热设计原则，抗振设计原则和可测试性原则。抗干扰性原则主要包括以下原则：

1. 电源线的设计：选择合适的电源，尽量加宽电源线（增大负载电流），保证电源线，地线走向与数据传输方向一致，使用抗干扰元器件（如磁珠，电源滤波器等），电源入口添加去耦电容；
2. 电线的设计：模拟地与数字地分开，尽量采用单点接地，尽量加宽地线，将敏感电路连接到稳定的接地参考源，对PCB板进行分区设计，把高带宽的噪声电路与低频电路分开，尽量减少接地环路的面积；
3. 元器件的配置：不要有过长的平行信号线，保证PCB的时钟发生器、晶振和CPU的时钟输入端尽量靠近，同时远离其他低频器件，元器件应围绕核心器件进行配置，尽量减少引线长度，对PCB板进行分区布局，考虑PCB板在机箱中的位置和方向，缩短高频元器件之间的引线；
4. 去耦电容的配置：每10个集成电路要加一片充放电电容，引线式电容用于低频，贴片式电容用于高频，每个集成芯片要布置一个0.1uF的电容，对抗噪声能力弱、关断时电源变化大的器件要加高频去耦电容，电容之间不要公用过孔，去耦电容引线不能太长；
5. 降低噪声和电磁干扰的原则：尽量采用45°折线而不是90°折线，用串联电阻的方法来降低电路信号边沿的跳变速率，石英晶振的外壳要接地，闲置不用的门电路不要悬空，时钟垂直于IO线时干扰小，尽量让时钟线周围的电动势趋于零，IO驱动电路尽量靠近PCB的边缘，任何信号不要形成回路，对高频板，电容的分布电感不能忽略，电感的分布电容不能忽略，通常功率线、交流线尽量布置在和信号线不同的板子上；
6. 其他设计原则: CMOS的未使用引脚要通过电阻接地或接电源，用RC电路来吸收继电器等原件的放电电流，总线上加10K左右上拉电阻有助于抗干扰，采用全译码有更好的抗干扰性，元器件不用引脚通过10K电阻接电源，总线尽量短，尽量保持一样长度，两层之间的布线尽量垂直，发热元器件尽量避开敏感电容，除了地线，能用细线的不要用粗线。、

用于本项目的PCB设计图如下：

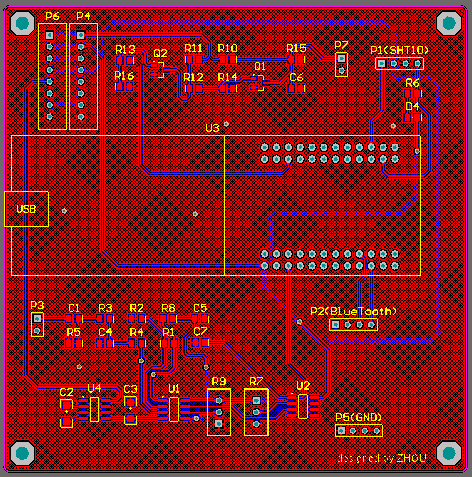


图4.7 使用Altium Designer开发的PCB图（2D视图）

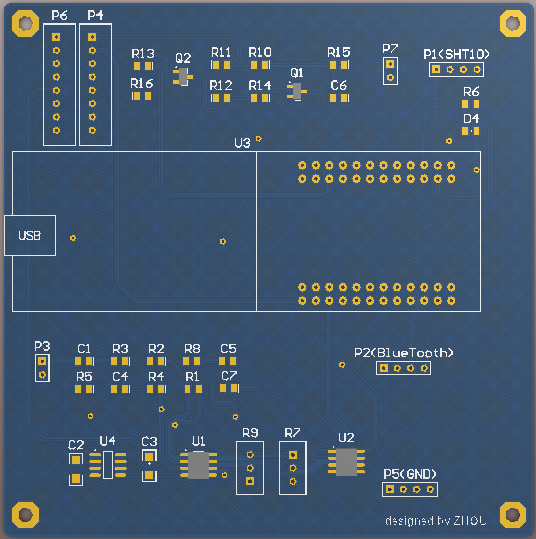


图4.8 使用Altium Designer开发的PCB图（3D视图）

5 软件设计及开发

本章主要介绍了本毕业设计的软件设计工作部分。其中，包括基于嵌入式系统的数据采集装置软件设计与开发，基于Android OS的数据接收与分析软件设计与开发和基于Matlab的数据滤波与数学模型的拟合。

5.1 基于嵌入式系统的数据采集装置软件设计与开发

本节主要介绍基于嵌入式系统的数据采集装置的软件设计与开发。主要包括了空气中温湿度数据采集的程序设计和超声波在空气中传播时间（声时）数据采集的程序设计。

5.1.1 空气中温湿度数据采集的程序设计

SHT10的串行接口，在传感器信号的读取及电源损耗方面，都做了优化处理；但与I2C接口不兼容。因此，需要使用微控制器的通用I/O口模拟出该串行接口的串行时钟输入(SCK)和串行数据(DATA)。SCK用于微处理器与SHT10之间的通讯同步。由于接口包含了完全静态逻辑，因而不存在最小SCK频率。串行数据DATA三态门用于数据的读取。DATA在SCK时钟下降沿之后改变状态，并仅在SCK时钟上升沿有效。数据传输期间，在SCK时钟高电平时，DATA必须保持稳定。为避免信号冲突，微处理器应驱动DATA在低电平。

为了避免可能在未来出现的复用引脚的I/O功能与其他外设功能的冲突，选择I/O引脚时应尽量使用没有复用功能的通用I/O引脚。基于该原则，我们在这里使用了LPC1114微控制器的P2.7作为DATA接口，将P2.8作为SCK接口。初始化I/O引脚时应将其定义为GPIO功能，并对内部上拉电阻使能，同时为了在初始化时不产生信号冲突，应将这些GPIO的方向寄存器置1（配置为输出），并配置为初始化输出高电平（SHT10在下降沿读取数据）。同时由于LPC1114微控制器不能配置为同时使能输入与输出模式，故在编写代码实现数据的读写时应时刻注意切换GPIO的输入输出模式。

1. 发送命令

用一组“启动传输”时序，来表示数据传输的初始化。它包括：当SCK时钟高电平时DATA翻转为低电平，紧接着SCK变为低电平，随后是在SCK时钟高电平时DATA翻转为低电平。这样做的好处在于这与串行数据读取的规则是冲突的，因此在传输数据时不会出现误传“启动传输”的命令。具体时序图如下：

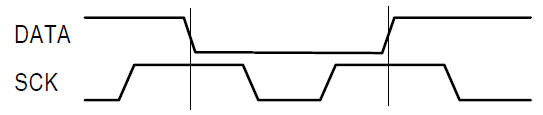


图5.1 “启动传输”时序

后续命令包含三个地址位（目前只支持“000”），和五个命令位。SHT10会以下述方式表示已正确地接受到指令：在第8个SCK时钟的下降沿之后，将DATA下拉为低电平（ACK位）。在第9个SCK时钟的下降沿之后，释放DATA（恢复高电平）。

表5.1 SHT10命令集

|  |  |
| --- | --- |
| 命令 | 代码 |
| 预留 | 0000x |
| 温度测量 | **00011** |
| 湿度测量 | **00101** |
| 读状态寄存器 | 00111 |
| 写状态寄存器 | 00110 |
| 预留 | 0101x-1110x |
| 软复位，复位接口、清空状态寄存器，即清空为默认值  下一次命令前等待至少11ms | **11110** |

1. 测量时序(RH和T)

发布一组测量命令（‘00000101’表示相对湿度RH，‘00000011’表示相对温度T）后，控制器要等待测量结束。这个过程需要大约20/80/320ms，分别对应8/12/14bit测量（在此处使用14bit测量）。确切的时间随内部晶振速度，最多可能有-30%的变化。SHT10通过下拉DATA至低电平并进入空闲模式，表示测量的结束。控制器在再次触发SCK时钟前，必须等待这个“数据备妥”信号来读出数据。检测数据可以先被存储，这样控制器可以继续执行其它任务在需要时再读出数据。

接着传输2个字节的测量数据。微控制器需要通过下拉DATA为低电平，以确认每个字节。所有的数据从最高有效位（MSB）开始，右值有效。（例如，对于14bit数据，从第3个SCK时钟起算作MSB；而对于8bit数据，首字节则无意义）。

在测量和通讯结束后，SHT10自动转入休眠模式。同时，为保证自身温升低于0.1℃，SHT10的激活时间不要超过10%。

具体测量时序举例如下图：

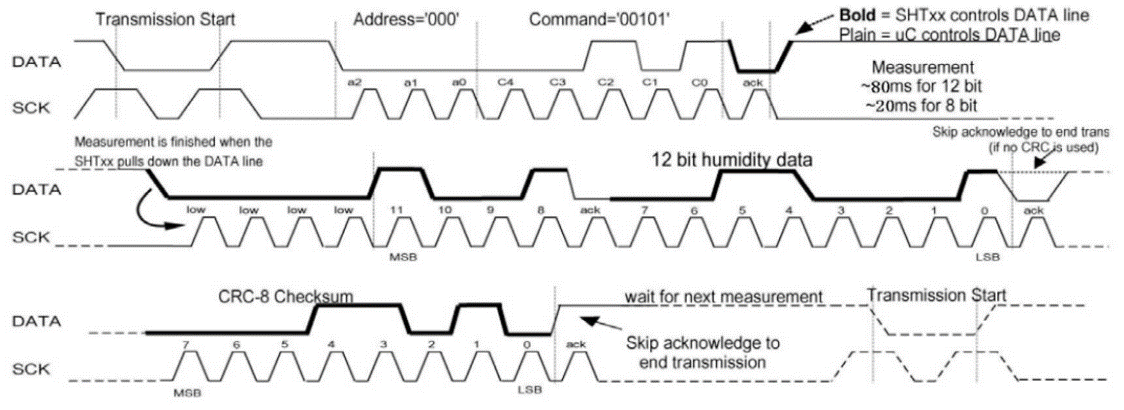


图5.2 RH测量时序举例:“0000’1001’0011’0001”=2353=75.79%RH（未包含温度补偿）

1. 通讯复位时序

如果与SHT10通讯中断，下列信号时序可以复位串口：

当DATA保持高电平时，触发SCK时钟9次或更多。在下一次指令前，发送一个“传输启动”时序。这些时序只复位串口，状态寄存器内容仍然保留。具体时序图如下：

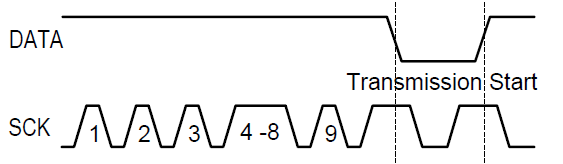


图5.3 通讯复位时序

5.1.2 超声波在空气中传播时间（声时）数据采集装置的软件设计与开发

声时测量模块的软件设计主要实现的功能包括向超声波传感器输入端发送180kHz的矩形脉冲波和从超声波输出端接收矩形脉冲波并捕获到第一个矩形脉冲。由于需要完成产生和捕获180kHz的矩形脉冲，嵌入式系统的节拍定时器产生的软件中断（周期为10ms）不满足需求。因此需要使用嵌入式系统的硬件定时器/计数器进行脉冲信号的发送与接收。

LPC1100系列Cortex-M0微控制器拥有2个32位和2个16位可编程定时器/计数器，均具有捕获比较功能。定时器用来对外设时钟(PLCK)进行计数，而计数器对外部脉冲信号进行计数，可选择在规定的时间内产生中断或执行其他操作。每个定时器/计数器还包含1个捕获输入，用来在输入信号变化时捕获定时器瞬时值和产生中断。

1. 微控制器输出矩形脉冲信号的程序设计与开发

本项目使用16位定时器0（CT16B0）进行输出信号的控制。将引脚P0.8配置为匹配输出功能，并打开16为定时器0的时钟模块，设置定时器的计数器（TC）与匹配寄存器0（MR0）的数匹配后复位TC并产生中断，同时将输出翻转，经过实验可知在MR0为132时输出脉冲的频率为180kHz。产生中断后在中断处理函数中将定义的计数的count对象加一，当count的值为10时将定时器关闭，此时即可产生5个180kHz的矩形脉冲信号。

1. 微控制器捕获经过放大和过零比较之后的超声波信号的程序设计与开发

在此处使用32位定时器1（CT32B1）进行输入信号的捕捉控制。具体初始化操作为将引脚P1.0配置为捕获输入功能，并打开32位定时器1的时钟模块，设置该定时器在信号下降沿（或上升沿，均可）捕获，捕获之后将定时器计数器的值载入捕获寄存器中并进入中断，最后在中断处理函数中将捕获寄存器中的数取出即可。需要注意的是在硬件电路设计与开发章节中可知经过硬件电路处理之后的超声波信号为一串矩形波，因此当定时器捕获到第一个矩形波之后需要将定时器关闭，否则捕获寄存器将在对应的上升沿或下降沿不停载入定时器计数器（TC）的值并不断进入中断。

5.1.3 蓝牙发送数据模块程序设计与开发

该项目使用的蓝牙模块为HC-05嵌入式蓝牙串口通讯模块，该模块具有两种工作模式：命令响应工作模式和自动连接工作模式[10]。在自动连接模式下模块又分为主(Master)、从(Slave)、和回环(Loopback)三种工作角色。蓝牙技术规定每一对设备之间进行蓝牙通讯时，必须一个为主角色，另一个为从角色，才能进行通信。通信时，必须由主端发起查找，发起配对，建链成功后，双方即可收发数据。由于需要实现安卓手机与蓝牙模块的通信，而当蓝牙模块配置为主模式时无法通过设置PIN码的方式与安卓手机配对，故只能将蓝牙模块配置为从模式。配置过程为使用蓝牙模块的相应工作模式，在该工作模式下时用户可以向模块发送各种AT指令，为模块设定控制参数或发布控制命令。具体操作如下：

1. 将蓝牙模块的TXD引脚与串口模块的RXD引脚，蓝牙模块的RXD引脚与串口模块的TXD引脚相连接（有的串口可能由于制作工艺的原因TXD与RXD标反，因此需要在实际测试后才能确定引脚连接）；
2. 在上电之前将蓝牙模块的KEY引脚与高电平的引脚相连（例如VCC引脚），再将VCC引脚和GND引脚与串口模块对应的VCC和GND引脚连接，上电之后若蓝牙模块的指示灯长亮长灭则说明进入命令相应工作模式（自动连接工作时蓝牙模块指示灯为快闪）；
3. 使用串口助手，设置串口波特率为38400，数据位8位，停止位1位，无校验位，无流控制；
4. 打开串口，用串口发送字符“AT\r\n”，若通信成功则蓝牙模块将向串口发送字符”OK\r\n”，如下图所示(图见下页)。

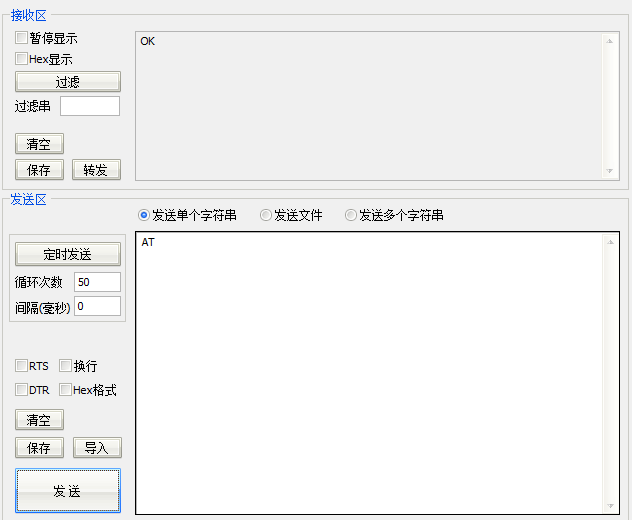


图5.4 使用串口发送AT指令配置蓝牙芯片

当蓝牙模块成功进入命令相应工作模式并能成功接收AT指令后，就可以使用AT指令对蓝牙芯片进行配置，具体指令见下表：

表5.2 常见的AT指令集

| 指令说明 | 指令 | 响应 | 参数 |
| --- | --- | --- | --- |
| 测试指令 | AT | OK | 无 |
| 模块复位 | AT+RESET | OK | 无 |
| 恢复默认状态 | AT+ORGL | OK | 无 |
| 获取模块蓝牙地址 | AT+ADDR? | +ADDR:<Param> OK | Param:模块蓝牙地址 |
| 设置设备名称 | AT+NAME=<Param> | OK | Param:蓝牙设备名称 默认名称：“HC-05” |
| 查询设备名称 | AT+NAME? | 1、 +NAME:<Param>  OK——成功 2、 FAIL——失败 | 同上 |
| 设置模块角色 | AT+ROLE=<Param> | OK | Param:参数取值如下： 0——从角色（Slave） 1——主角色（Master） 2——回环角色（Slave-Loop） 默认值：0 |

续表 5.2

|  |  |  |  |
| --- | --- | --- | --- |
| 查询模块角色 | AT+ROLE? | +ROLE:<Param> OK | 同上 |
| 设置配对码 | AT+PSWD=<Param> | OK | Param:配对码 默认名称：“1234” |
| 查询配对码 | AT+PSWD? | +PSWD:<Param> OK | 同上 |
| 设置串口参数 | AT+UART=<Param1>,<Param2>, <Param3> | OK | Param1：波特率（bit/s） 取值如下（十进制）： 4800，9600，19200，38400， 57600，115200，23400，460800，921600，1382400 Param2：停止位 0——1位 1——2位 Param3：校验位 0——None 1——Odd 2——Even 默认设置：9600，0，0 |
| 查询串口参数 | AT+UART? | +UART=<Param1>,<Param2>, <Param3> OK | 同上 |

使用AP指令配置好蓝牙模块后，即可按硬件电路章节中所述形式将蓝牙模块与微控制器的UART模块相连，再配置好UART的底层寄存器，就可以使用蓝牙收发数据了。

微控制器UART模块配置步骤为：将P1.6设置为串行输入管脚（RXD），P1.7设置为串行输出管脚（TXD）,打开UART功能部件时钟，设置与蓝牙模块相同的波特率，即可使用。

5.2 基于Android OS的手机上位机程序设计与开发

本节主要介绍了基于安卓操作系统的手机上位机接收程序的设计与开发，主要实现了调用手机的蓝牙模块接收数据，对声时数据进行初步的滤波和将温度，湿度和声时数据以折线图形式显示在手机屏幕上三个功能。原本期望能直接在手机上实现将声时数据与温度值进行分析拟合出数学模型，可是在实际操作中发现Android OS提供的API不足以实现该功能，若自己重写API工作量过于庞大，因此在此只将声时数据进行简单的整理和滤波，而拟合数学模型的工作则是Android上位机将处理好的数据写进手机存储卡的文件内再将文件导入电脑使用Matlab进行操作（具体操作见下节）。

5.2.1 使用手机自带的蓝牙模块接收数据的程序设计与开发

首先，要操作蓝牙，先要在Android工程文件的AndroidManifest.xml里加入权限:

<uses-permissionandroid:name="android.permission.BLUETOOTH\_ADMIN" />

<uses-permissionandroid:name="android.permission.BLUETOOTH" />

设备间的蓝牙连接可通过一下两种方法解决：

方法1： 先进行蓝牙自动配对，配对成功，通过UUID获得BluetoothSocket，然后执行connect()方法。

方法2：通过UUID获得BlueToothSocket，然后先根据mDevice.getBondState()进行判断是否需要配对，最后执行connect()方法。

由于在程序中进行蓝牙设备的搜索与配对需要占用手机的大量资源，而且在Android系统的设置中能非常方便的对设备进行配对操作，故本项目选用第一种方法进行蓝牙连接。

当手机与蓝牙模块连接之后即可通过蓝牙将需要采集的数据进行接收。蓝牙接收的数据格式为字节流格式，因此需要使用read()方法将流读出。需要注意的是，由于Android的内部机制不允许在主线程（即UI线程）里进行数据的接受和发送，故需要新建一个线程（Thread）对象再将接收的程序通过Handler使用sendMessage()方法将数据传到UI线程中再更新UI。

由于在发送时单片机将数据进行了装帧发送，因此在接受数据时Android程序需要将数据进行解帧。在编写改程序的时候需要遵从设计模式中的依赖倒置原则，即高层模块不应该依赖于底层模块，二者都应该依赖其抽象；抽象不应该依赖细节，细节应该依赖抽象。依赖倒置原则的核心就是需要我们面向接口编程，因此我创建了一个DataHandler接口，并新建了一个HandleDataFromBlueTooth的对象来实现该接口并实现接口中的HandleData()抽象方法。再将从流中读取的字节数据一个字节一个字节的传入该对象并进行解帧，解帧的方法如下：

1. 新建一个状态对象，初始化为在帧外（可使用枚举变量），读传入对象的字节，若字节为0xAA，则状态更新为帧头1；
2. 当状态处于帧头1后读入下一个字节，若下一个字节为0xBB，则状态更新为帧头2，表示读到帧头，否则状态更新为帧外，并新建一个字节型的校验位变量，赋初值为0x00；
3. 当状态处于帧头2后读入下一字节，该字节的数据为数据帧的长度，注意数据帧的长度为数据域的长度（详情见图3.4），同时将该字节数据与校验位变量进行异或运算，最后新建一个数据域的字节型数组对象存放数据域并将状态对象更新为数据帧；
4. 当状态处于数据帧后依次将读入的字节数据放入之前已建立的数据域数组对象，并将校验位对象与每个字节依次进行异或运算，指导该数组写满，此时将状态对象更新为校验位；
5. 当状态处于校验位后将计算之后的校验位字节与读入的字节进行比较，若一致则说明数据传输过程中没有出差错，将数据域数组通过Handler传入UI线程，否则丢弃该帧，最后将状态更新为帧外。

5.2.2 对声时数据进行简单滤波的程序设计与开发

将接收回来的数据进行分析并绘图后发现声时数据有比较大的偏差，具体图像如下：

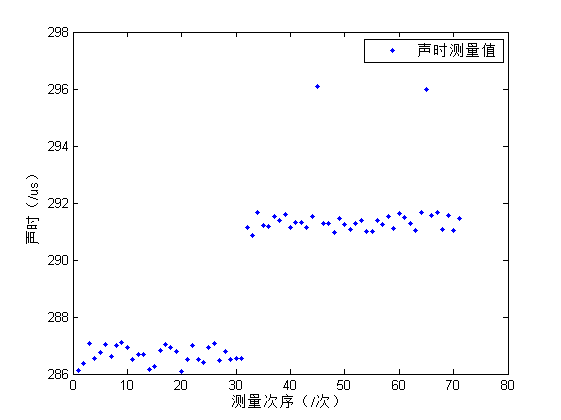


图5.5 未经手机算法处理的声时数据绘图

造成这种现象的原因是随着温度的变化，超声波模块输出的波形会受到温度和环境噪声的影响，因此有时过零比较器不能将包络中正弦信号的前几个低幅值的波捕捉到并输出为0-3.3V的逻辑电平，造成数据的偏差。不过由于每个正弦信号的频率都为180kHz，造成偏差值都是固定的，故可利用该特性对信号进行处理。处理算法思路为：从宏观上看假设前后两个声时数据的变化值非常小（小于一个180kHz的信号时长），故可将之后的一个数据与之前的数据进行比较，若偏差值大于66（单片机的计数器计132为180kHz），则将该数据相应的加或减132，再将该数据与之前一个数据进行比较，直到偏差值小于66，通过这种方法消除由于硬件造成的不可靠数据偏差，经过该算法处理之后的声时数据图如下：

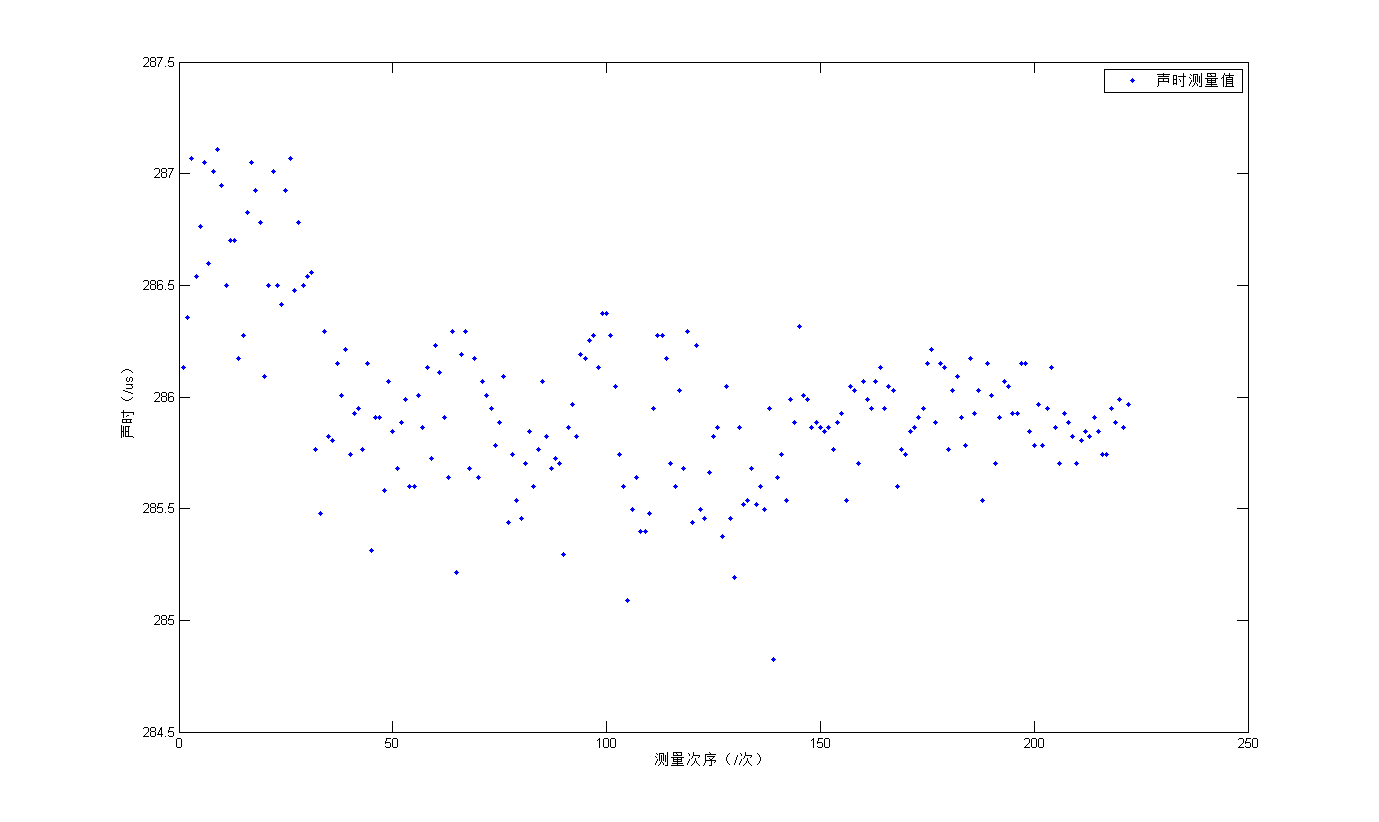


图5.6 经过手机算法处理后的声时图像

5.2.3 将处理过后的数据画图并显示在手机屏幕的程序设计与开发

目前获得的数据类型为字节型，通过阅读SHT10温湿度传感器的用户手册和使用示波器观察计数器计数值与时间的对应关系后，我们得到如下的数学关系表达式：







其中，,,分别为单片机通过蓝牙装填在数据帧中的两字节温度，湿度和声时数据，而,,分别为实际的温度，湿度和声时数据，这些数据都为双精度浮点型对象。

将字节型数据对象转化为双精度浮点型数据对象之后就可以进行画图操作了。Android SDK的API自带了Canvas类进行2D绘图，不过由于Canvas类使用过于复杂，且不能画动态的曲线，故本项目不使用Canvas类进行绘图，而选择了google的开源图标库Achartengine。该绘图库使用非常简单，在Android 4.4.2系统下可在项目下直接新建libs文件夹并把jar文件放入即可导入该库，而在具体的绘图过程中，调用ChartFactory的get\*\*\*()方法即可获得特定类型的图表。（例如，本项目需要使用折线图，调用ChartFactory的getLineChartView()方法就可以返回一个图标的Intent，至于图标的内容是什么，就需要在dataset和renderer中布置。）dataset里为基本统计数据，例如每种元素的名称(string)和数量(double)。renderer则指明了图的样式，例如每个元素的颜色，标题的大小，背景颜色等。

5.3 基于Matlab的卡尔曼滤波算法程序设计与开发

将Android手机接收的数据通过文件输出流写入文本文件，将文本文件导入到电脑里便可使用Matlab程序对数据进行处理。Matlab程序主要实现了通过卡尔曼滤波对数据进行处理的功能。算法主要的思想是不停的将最优值误差进行递归，估算出最优值，例如要对温度进行卡尔曼滤波时，首先查询温湿度传感器SHT10的技术手册得到测量误差为±0.5℃，再假定在宏观状态下每一测量时刻与下一测量时刻的温度差别在±0.1℃之内，使用两者的协方差来估算出该时刻的温度值，同时求出该时刻最优值的偏差，将该偏差代入到下一时刻对于温度的计算。具体效果如下图：

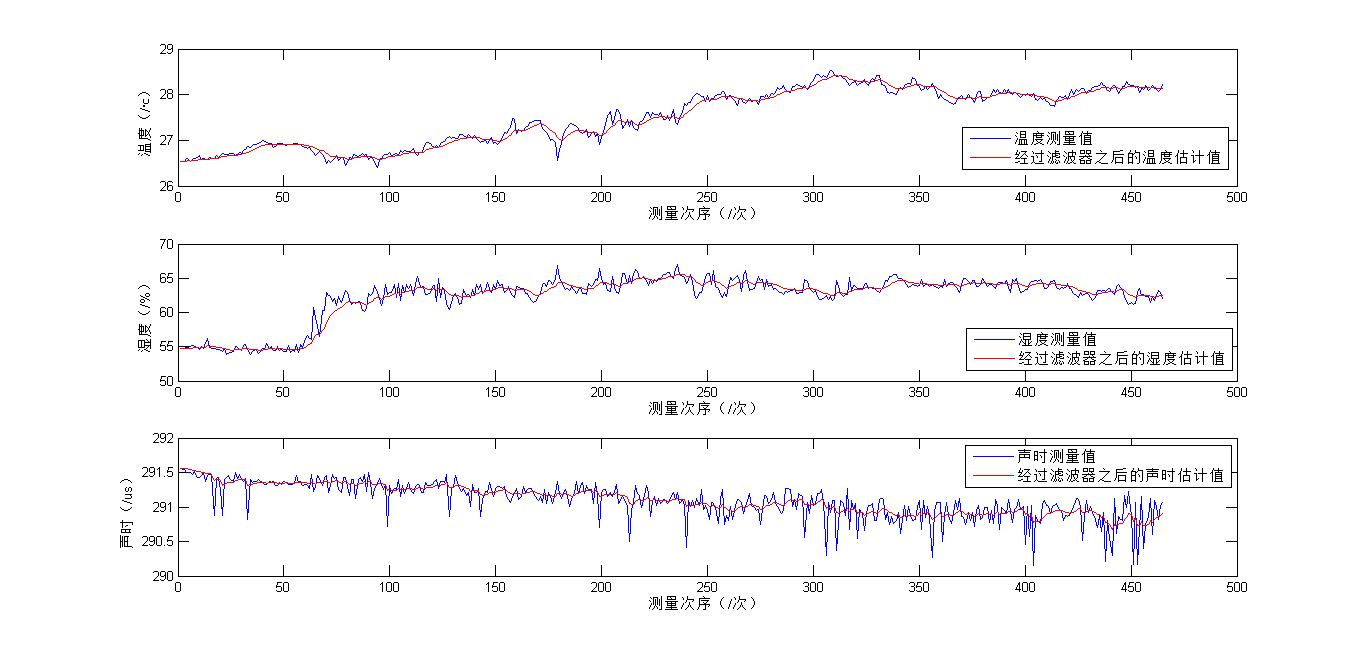


图5.7 经过卡尔曼滤波器之前与之后的图像

5.4 使用版本控制软件对代码工程进行管理

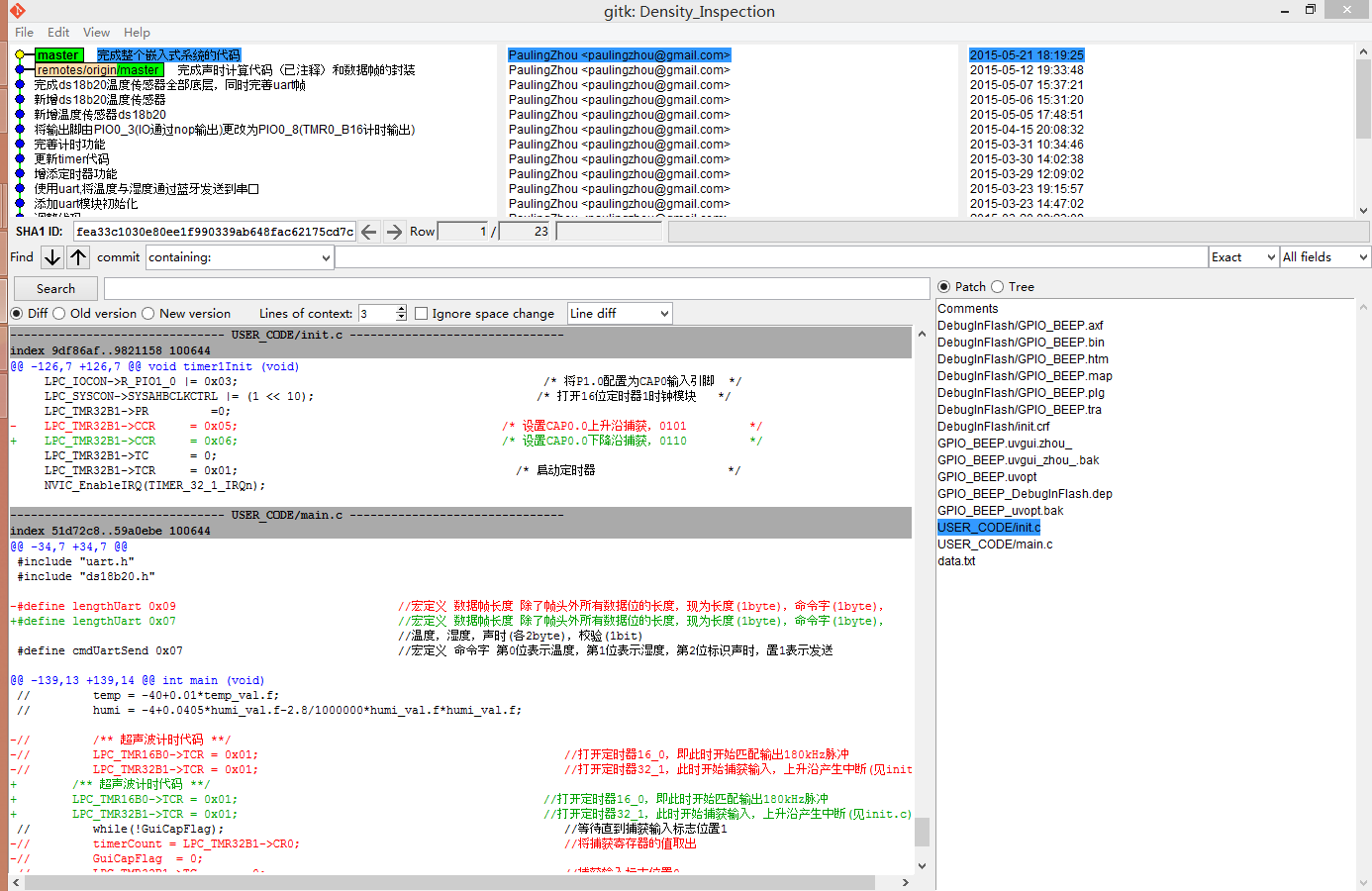
在此的版本控制软件特指GitHub，只需在每次实现一个小的功能后将使用commit命令整个工程文件提交到本地仓库，实现一个比较大的突破后使用push命令将工程文件提交到远端仓库。如果想回到之前的版本可以使用checkout命令就能非常简便的回退到之前的版本。同时使用gitk命令可以阅览每次提交的说明和具体修改代码的位置。下图为嵌入式系统代码的提交记录：

图5.8 嵌入式系统工程Git提交记录

6. 实验分析

6.1 试验装置的搭建

将超声波模块固定在试验架上保持输入与输出模块处于一条直线上。将超声波输入模块的接口与硬件电路板的输出接口相连，将超声波输出模块与硬件电路板的输入模块相连；使用杜邦线将温湿度传感器与硬件电路板给出的接口相连，同时将温湿度传感器固定在超声波模块旁边；将蓝牙模块配置为从模式并插在硬件电路板给出的蓝牙接口上即完成试验装置的搭建。搭建完毕的图如下：

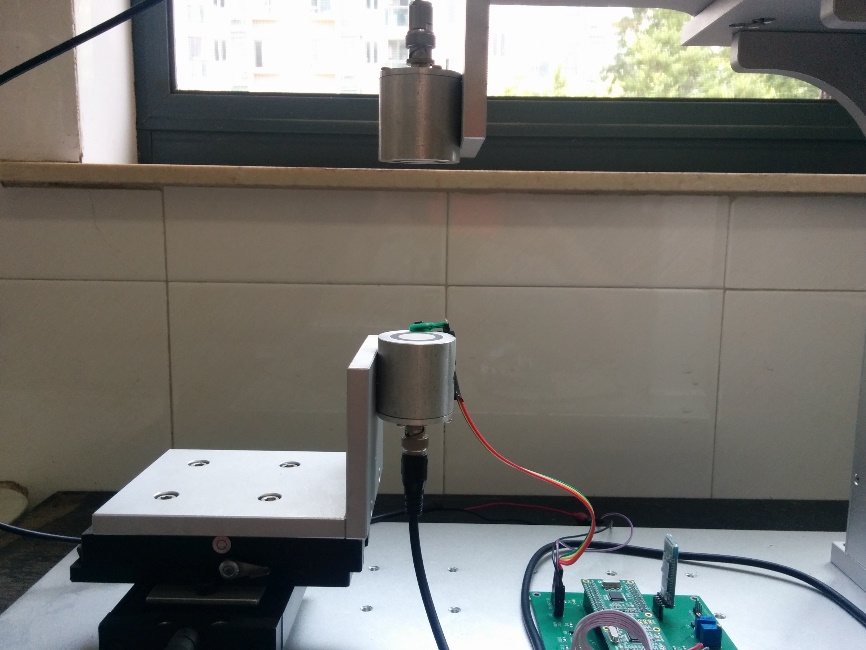


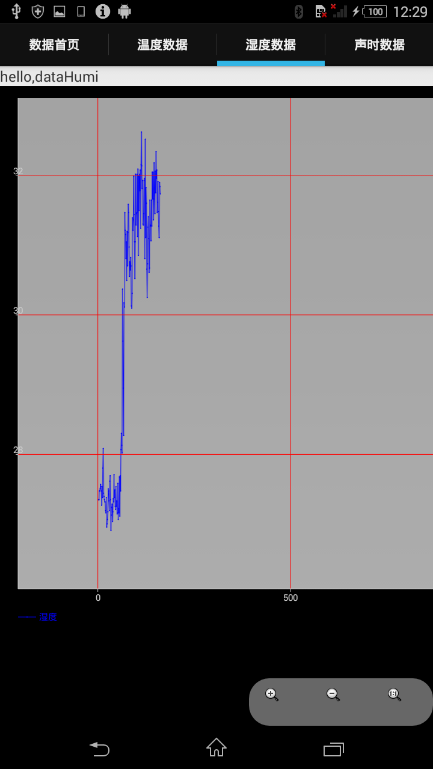
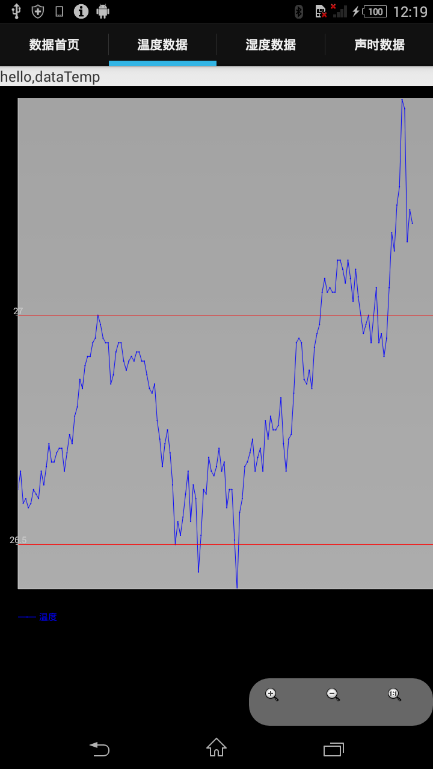
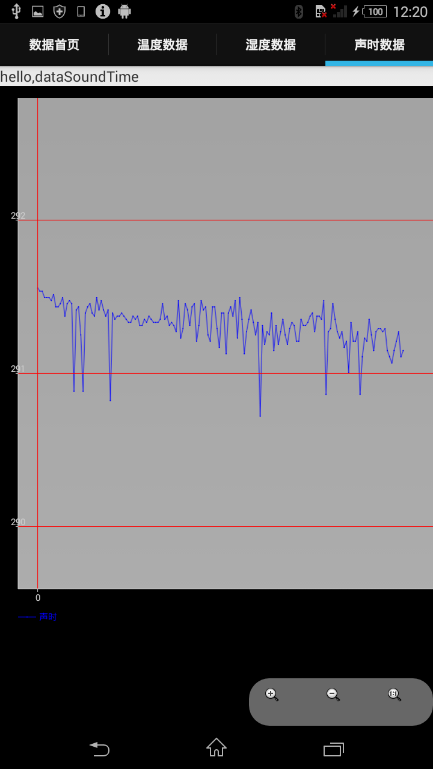
图6.1 实验装置图

6.2 实验数据的获取

将实验装置放在窗台边，上电，更改单片机代码使单片机每隔一分钟对环境温度湿度和超声波声时进行测量并装帧发送到手机上，手机的上位机程序对测量数据进行简单处理就将数据写入手机内存的文本文件，将文本文件取出即可使用Matlab对数据进行处理。具体流程图如下图：



图6.2 数据获取流程图

图6.3 使用Android上位机接收数据

6.3 实验数据与理论模型比较并分析

由第二章理论知识可得如下公式：



其中，为二元混合气体平均声速；为二元混合气体平均定压定容热容量比；为摩尔气体常数；为混合气体的摩尔质量；为热力学温度。

通过查阅文献可得空气的摩尔质量=28.96g/mol，定压定容热容量比=1.399[11]，代入式(1)可得空气的绝对温度为：



将式(6.2)改写为速度关于温度的函数：



其中t为摄氏温度。

由于条件限制，本实验验证的是超声波声速与温度之间的函数关系，将实验测得的数据与理论模型进行比较，已知超声波发送与接收探头之间的距离为10厘米，则实测超声波声速为：



其中为声音传播的时间。

在Matlab里将理论计算的数学模型与实际测得的数据进行绘图并比较，图像如下：

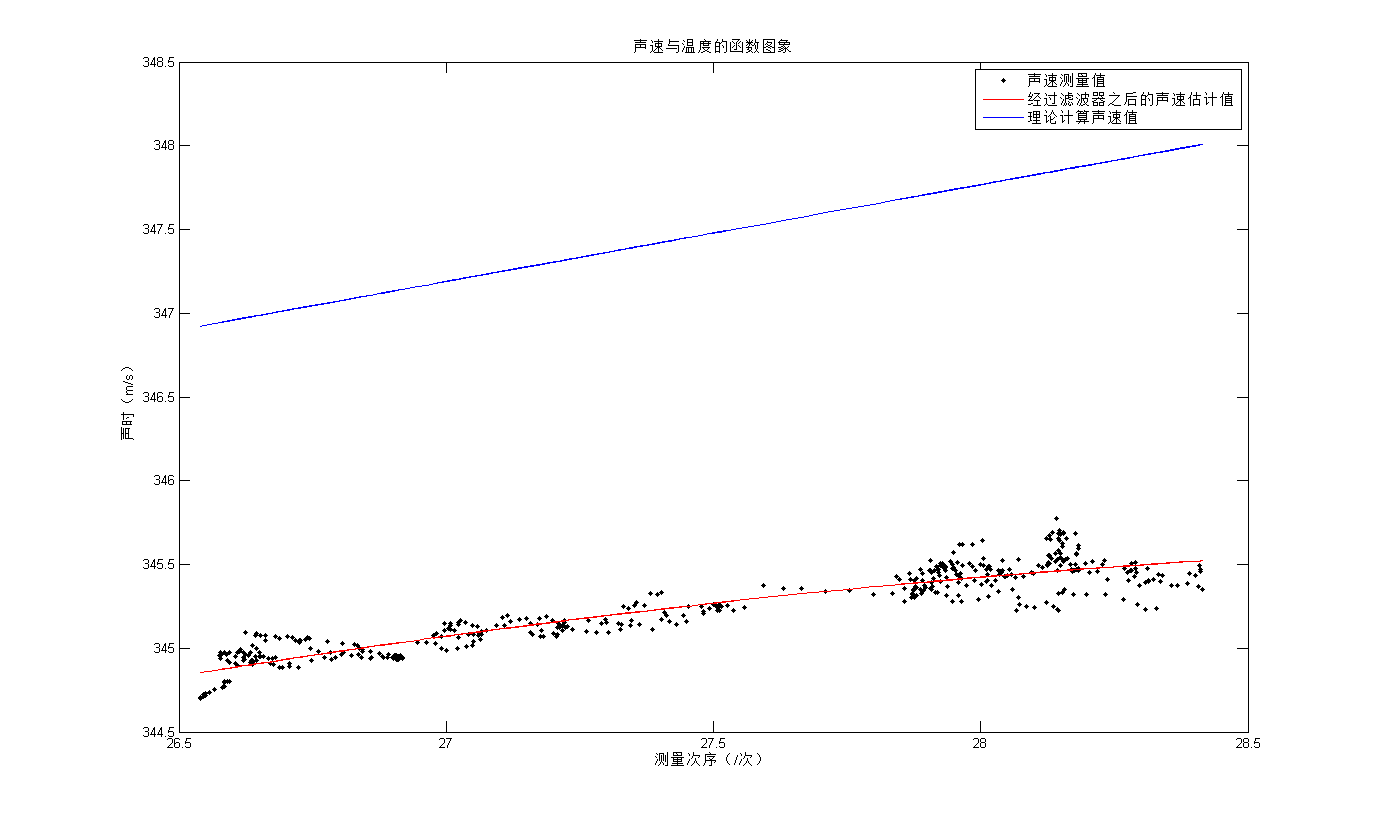


图6.4 理论计算与实际测量数据图像

有图像可得实测数据与理论计算比较有2m/s的速度差，折合为声时即有300ns的误差，分析原因为硬件电路中信号传输与换能晶片电信号与超声信号相互转换之间产生的时间延迟。取出这些影响该系统测量数据满足理论计算的气体模型，故该系统可用于检测空气浓度变化。