基于超声波的空气污染物浓度测量系统研制

装 订 线

摘要

随着现代社会工业水平的不断进步与人民生活水平的不断提高，环境保护，尤其是空气保护成为被热议的话题之一。因此，研制一款实时的且方便操作的空气污染物浓度测量系统势在必行。本项目基于这些原理，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。利用超声波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。本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。同时本系统选用恩智浦(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，再将这些数据以特定格式封装成帧并通过无线通信方式发出。本系统使用本人编写的基于安卓操作系统(Android OS)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并以图像形式显示在手机屏幕上。整个项目使用 Git 以及 Subversion 作为版本控制工具。通过查阅相关文献和分析实验图像可得开发出的检测系统实物可测出误差为±0.5℃的温度，误差为 3.5%RH 的湿度和误差为±50ns 的声时数据，同时通过实验测出空气中声时对应于温度的变化并与理论数学模型进行分析和比较。

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

With the development of modern society and the improvement of people's lives, the environmental protection, especially the air protection, has become one of the hot topics. Thus, developing a real time as well as easy operating system in using air pollutants' concentration measuring is imperative. This project is based on these principles. Through 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. 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. By referring to some relevant literatures and analyzing images of experiment we can conclude that the equipment which is developed can measure temperature with error in ±0.5℃, humidity with error in ±3.5RH and the time data of ultrasonic signal with error in ±50ns. Meanwhile, I measure the time data corresponding to changes in temperature through experiments and analyze these data with comparison of mathematical model in theory.

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

目 录

1 引 言 1
1.1 研制空气污染物浓度检测系统的意义 1
1.2 空气污染物浓度检测的常用方法 1
1.2.1 利用电阻式气敏元件测量气体浓度 1
1.2.2 气相色谱法测量气体浓度 1
1.2.3 载体催化燃烧法检测气体浓度 2
1.3 超声技术测量气体浓度的基本概况 2
1.4 本文所作的工作 2
2 理论部分 3
2.1 浓度声速理论 3
2.2 嵌入式系统相关理论 4
2.3 Android 操作系统及 JAVA 编程相关知识 5
2.3.1 Android 操作系统 5
2.3.2 JAVA 程序设计理论 6
2.3.3 设计模式 6
2.4 版本控制相关知识 6
3 需求分析与系统架构 8
3.1 需求分析 8
3.2 系统架构 8
3.2.1 硬件设计框架 8
3.2.2 软件设计架构 9
3.3 各模块设计思路 10
3.3.1 温湿度传感器模块设计思路 10
3.3.2 超声波传感器模块实现计时功能设计思路 10
4 硬件设计与开发 12
4.1 温湿度模块的硬件设计与开发 12
4.2 蓝牙模块的硬件设计与开发 12
4.3 超声波模块的声时测量硬件设计与开发 13
4.3.1 超声波模块发射端硬件电路设计 13
4.3.2 超声波模块接收端硬件电路设计 14
4.4 印刷电路板(PCB)的设计与开发 15
5 软件设计及开发 18
5.1 基于嵌入式系统的数据采集装置软件设计与开发 18
5.1.1 空气中温湿度数据采集的程序设计 18
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 20
5.1.3 蓝牙发送数据模块程序设计与开发 21
5.2 基于 Android OS 的手机上位机程序设计与开发 23
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 24
5.2.2 对声时数据进行简单滤波的程序设计与开发 25
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 26
5.3 基于 Matlab 的卡尔曼滤波算法程序设计与开发 26
5.4 使用版本控制软件对代码工程进行管理 27
6. 实验分析 29

装 订 线

6.1 试验装置的搭建 29
6.2 实验数据的获取 29
6.3 实验数据与理论模型比较并分析 30
7 总结与展望 33
7.1 总结 33
7.2 展望 33
参考文献 35
谢辞 36

装

订

线

1 引言

1.1 研制空气污染物浓度检测系统的意义

随着社会现代化水平的不断提高，人们已将注意力由温饱转向提高自己的生活质量。每个人都希望能看见蓝天白云，能在清新的空气下工作生活。因此，设计出一套实时性强且易于操作的空气浓度检测系统将会极大改善人们的生活水平。

如今人们一般都会使用公共网站所给出的空气污染指数（air pollution index, API）作为判断环境空气质量好坏的标准。空气污染指数是将各种复杂的空气污染物浓度集成为单一的数值形式的概念性对象，其结果简单明了，使用方便，适合于表征城市中短期内空气质量的状况和未来变化的趋势。不过，空气污染指数只能反映一片区域的平均空气质量，当对应到要测量某一特定区域的空气质量，例如某间教室或者某间实验室时，就不能使用网络提供的数据来判断空气质量的好坏，而需要使用传感器对污染物的浓度进行实时的测量。

同时，若能设计并制造出精度高且实时性强的空气污染物浓度检测仪，并将其放置于工厂的废气出口处，把采集的数据通过互联网传输到相关部门，就可以对工厂排放的废气浓度进行实时的监测，这样就能保障环境保护法中“企业事业单位和其他生产经营者应当防止、减小环境污染和生态破坏，对所造成的损害依法承担责任”[1]的条例能够顺利执行。

1.2 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，其阻值将随环境气体浓度的变化而变化。因此，只需测出某一时刻的元件阻值，便可利用公式或之前已通过实验标定拟合出的函数图像推算出实时的气体浓度。这也是气体浓度测量中最常用的传统方法。

利用气敏元件检测气体浓度只需测量电阻，因此操作比较简单，但存在以下不足：(a)由于检测仪一般使用电桥电路测量电阻，电桥输出值的非线性将会造成测量误差；(b)电桥电路的供电电压将极大影响测量精度；(c)由于气敏元件的电阻还与环境温度、空气扰动等一系列因素有关，所以在测量时还需要串接上补偿电路。因此，电阻式气敏元件只适用于允许测量误差较大的场合。

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所设计的测量气体浓度的方法，其主要被应用在工厂生产现场空气的检测。该方法对混合气体进行多次采样，将采样气体注入色谱仪后等待色谱峰值全部出现再进行色谱分析。由于混合气体的色谱随着气体浓度的变化存在比较明显的差异，故根据得到的色谱便可分析出混合气体中各气体组分的浓度值。这种检测方法操作方便、快捷，同时线性范围宽、灵敏度高、可靠性重现性好，因此在气体浓度的检测方法中占据很重要的地位。但在测量过程中必须合理地选择例如色谱柱或载气流速等参数，且只有进行多次重复性实验后才能获得比较理想的测量效果。[2]

1.2.3 载体催化燃烧法检测气体浓度

载体催化燃烧法使用载体型气敏元件作为测量可燃气体浓度的传感器，该元件由表面烧结一层陶瓷载体并涂覆催化活性物（例如铑元素或钯元素）的铂丝构成。当铂丝通入负载电流使之达到临界反应温度（320~350℃）时，可燃气体便会在元件表面催化燃烧，造成铂丝电阻增加，在可燃气完全燃烧且燃烧时产生的热辐射可被忽略时，铂丝电阻的增量ΔR将与可燃气体浓度C成正比关系。由此可知，若将电阻增量ΔR转换为电信号，便能用于检测可燃气体的浓度。在检测气体浓度的过程中，浓度信号采样单元一般由将催化元件及与之对应的参比元件组成的电桥电路组成。由于催化气敏元件的气敏特性除了与可燃气的浓度相关外，还会收到工作电流、环境温湿度及气压的影响，在具体的设计过程中应当注意采用桥式单元或者其他的参比元件对其进行补偿。

1.3 超声技术测量气体浓度的基本概况

利用超声波原理测量混合气体中各组分气体浓度是一种新型的检测方法，这种方法是由于近十年电子电路和测量技术的不断发展而产生的。使用超声波检测气体浓度的方法具有实时性强、测量范围广、精度高等优点。同时，由于超声波装置为非接触式测量气体浓度，故使用寿命长，因此在测量领域受到了极大关注。近年来，作为一种新型的检测技术，超声技术已被广泛应用于气体浓度的测量领域。国内外的一些公司经过了大量研究，已将超声技术成功地应用在高精度检测装置中，在气体检测领域取得了巨大突破。[3]

1.4 本文所作的工作

本文从理论上简单介绍使用超声模块对于气体浓度检测的原理，并介绍了完成该课题所做的工作。具体如下：

(1) 学习相关的理论知识，利用理想气体模型建立声速对于温度和空气平均分子质量的数学模型，并分析和验证使用超声波对空气浓度检测的可行性；

(2) 学习并掌握使用 Altium Designer 软件进行硬件电路的设计，包括原理图设计和印刷电路板(PCB)图设计，并利用 Subversion 进行硬件电路工程的版本控制；

(3) 学习并掌握 LPC1114 微控制器底层寄存器的配置，编写程序完成微控制器对温度、湿度和超声波声时数据的测量，学习并掌握面向对象编程的相关知识，包括依赖倒置原则，应用该原则编写程序完成基于安卓的手机终端接收程序，利用 Git 进行代码项目的版本控制；

(4) 完成实物的制作与开发，可成功利用传感器采集数据，对数据进行处理得到环境温度与超声波声时的关系，将得到的实验模型与理论的数学模型进行比较，并对误差进行分析。

2 理论部分

2.1 浓度声速理论

若按照化学性质分，空气污染物可被细分为许多种类。在这里使用空气污染指数的概念，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根据理想气体数学模型可得到混合气体的平均速度如下式[4]：

$$\overline{c}=\left(\frac{\overline{\gamma}P}{\rho}\right)^{0.5} \tag{2.1}$$

式中 $\overline{c}$ 为混合气体的平均速度；$\overline{\gamma}=\left(C_p/C_v\right)$ 为混合气体平均定压定容比热比；P 是气体压力；ρ 是气体密度。

使用理想气体方程：

$$PV=nRT=\frac{m}{\overline{M}}RT=\frac{\rho V}{\overline{M}}RT \tag{2.2}$$

式中 V 为气体体积；R 为通用气体常数，也称普适气体恒量，$R=8.314J/(mol\cdot k)$；T 为气体的热力学温度；m 为气体的质量；$\overline{M}$ 为气体的平均相对分子质量。

将式(2.2)代入式(2.1)中可得：

$$\overline{c}^2=\frac{\overline{\gamma}RT}{\overline{M}} \tag{2.3}$$

对于二元混合气体，设其两种组分分别由下标 a,b 组成，有：

$$\overline{\gamma}=\frac{nC_{pa}+(1-\mathrm{n})\mathrm{C}_{pb}}{nC_{va}+(1-\mathrm{n})\mathrm{C}_{vb}} \tag{2.4}$$

$$\overline{M}=nM_a+(1-\mathrm{n})\mathrm{M}_b \tag{2.5}$$

式中 n 为 a 类气体（普通空气）的浓度，$1-\mathrm{n}$ 为 b 类气体（污染物）的浓度，M_a,M_b 分别为 a,b 两种气体的相对分子质量。

根据式(2.3)，可令参数 Y 为：

$$Y=\frac{\overline{c}^2}{RT}=\frac{\overline{\gamma}}{\overline{M}} \tag{2.6}$$

将式(2.4)，式(2.5)代入式(2.6)后，整理可得：

$$An^2+Bn+K=0 \tag{2.7}$$

其中，系数 A,B,K 均为参数 Y 的函数。$A=(M_aC_{va}+M_bC_{vb}-M_aC_{vb}-M_bC_{va})Y$；$B=(M_aC_{vb}+M_bC_{va}-2C_{cb}M_b)Y-C_{pa}+C_{pb}$；$K=M_bC_{vb}Y-C_{pb}$。

由于 $0\le n\le 1$，所以方程(2.5)有单根，其解为：

$$n=\frac{-B+\sqrt{B^2-4AK}}{2A} \tag{2.8}$$

通过测量出的声速和温度两个量，可得出参数 Y 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[5]。

当微控制器的计时器以 50MHz 进行计数时，其对于测量的声时分辨率可达到 20ns，折合为测量超声波的声速精度为 0.03m/s（超声波发射与接收模块相距 0.1 米时），可测量气体平均分子质量的相对精度为±0.1(假设气体组分发生微小改变时定压定容比热比不变)，对于测量大分子量的污染物气体（例如二氧化硫或氮氧化合物）有着比较好的分辨率。

2.2 嵌入式系统相关理论

本设计选用的微控制器为恩智浦（NXP）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。本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 I^2C 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[6]。

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 片内静态随机存储器（SRAM）；

(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、内部先进先出数据缓存器（FIFO）和支持 RS-485/EIA-485 标准的通用异步收发传输器（UART）；

--同步通用串行接口控制器（SSP），带 FIFO 和多协议功能（仅在 LQFP48 和 PLCC44 封装中有两路 SSP）；

--I^2C（Inter-Integrated Circuit）总线接口，完全支持 I^2C 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。

(7) 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。

(8) 带有 SWD（Serial Wire Debug）调试功能；

(9) 集成了 PMU（电源管理单元），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；

(10)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；

(11)3.3V 单电源供电（2.0V~3.6V）；

(12)10 位模数转换器（ADC），在 8 个引脚中实现输入多路复用；

(13)GPIO 均可以配置为边沿或电平中断；

(14)带驱动的时钟输出功能可以反映主振荡器时钟、IRC（Internal Reference Clock）时钟、CPU 时钟和看门狗时钟；

(15)13 个起始逻辑功能可以将 CPU 从深度睡眠模式中唤醒；

(16)掉电检测，具有 4 个独立的阀值，用于中断和强制的复位；

(17)上电复位（POR）；

(18)主振荡器工作范围：1Mhz~25MHz；

(19)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；

(20)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；

(21)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[7]。

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[8]。

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。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。[9]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。

3 需求分析与系统架构

3.1 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

本项目主要架构如下图：

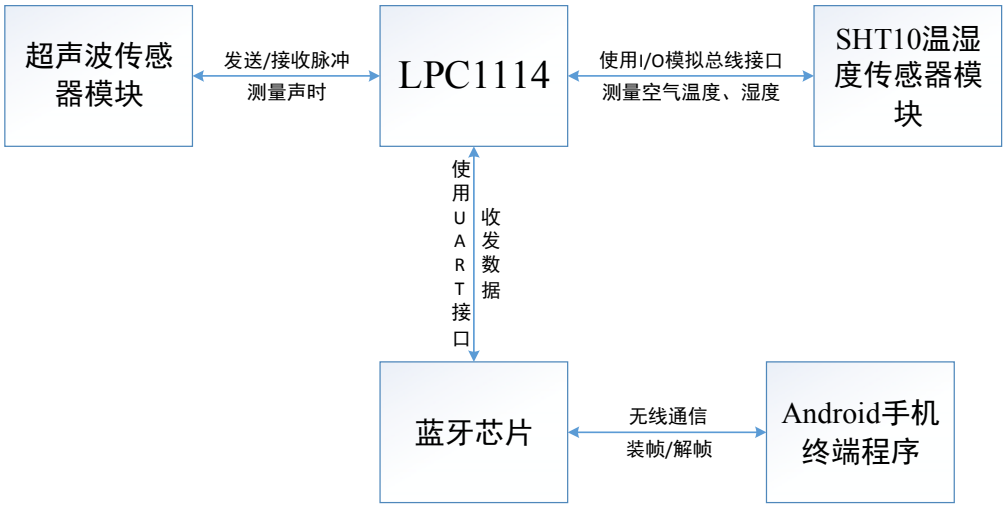

图 3.1 系统框架框图

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。

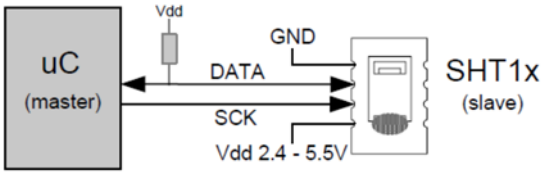

图 3.2 SHT10 温湿度传感器典型电路

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 I^2C 接口不兼容[10]，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（50MHz）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 3.3 硬件设计框架

3.2.2 软件设计架构

软件可以看做两部分：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。

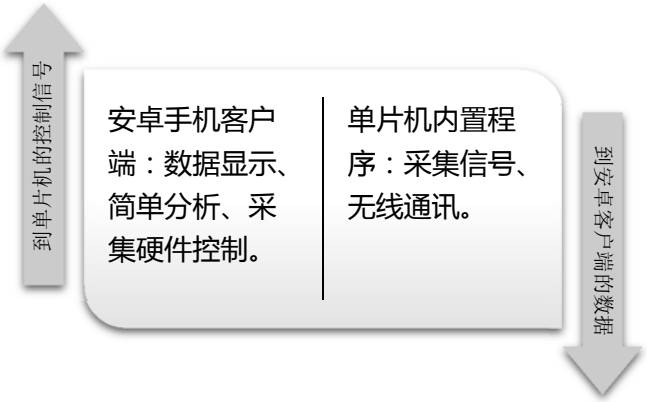

图 3.4 软件设计框架

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。底层帧的结构如下图所示：

帧头 1(0xAA)	帧头 2(0xBB)	帧长度	数据域	校验字

图 3.5 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 3.6 数据域（上层数据帧）格式

3.3 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 I^2C 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，经过实际测试可知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。

装

订

线

4 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：

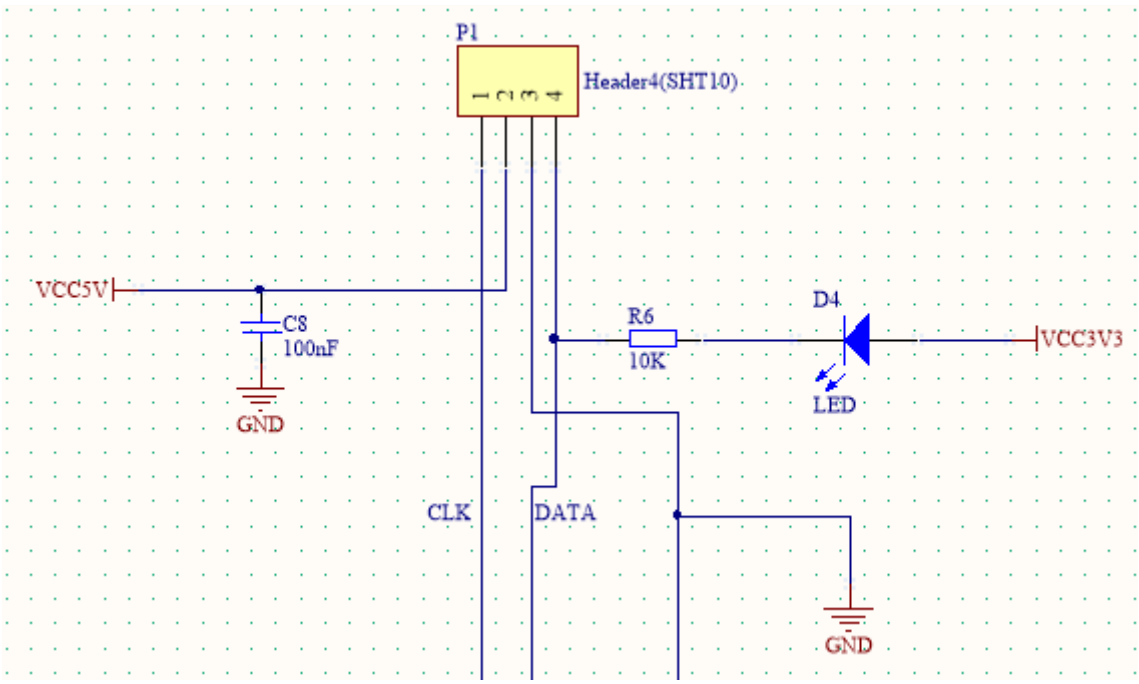

图 4.1 空气温度湿度测量硬件电路

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：

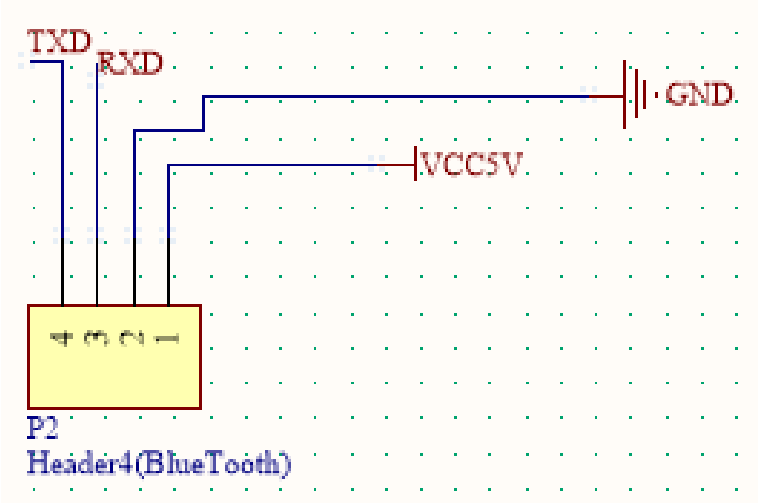

图 4.2 蓝牙通信模块硬件电路

4.3 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：

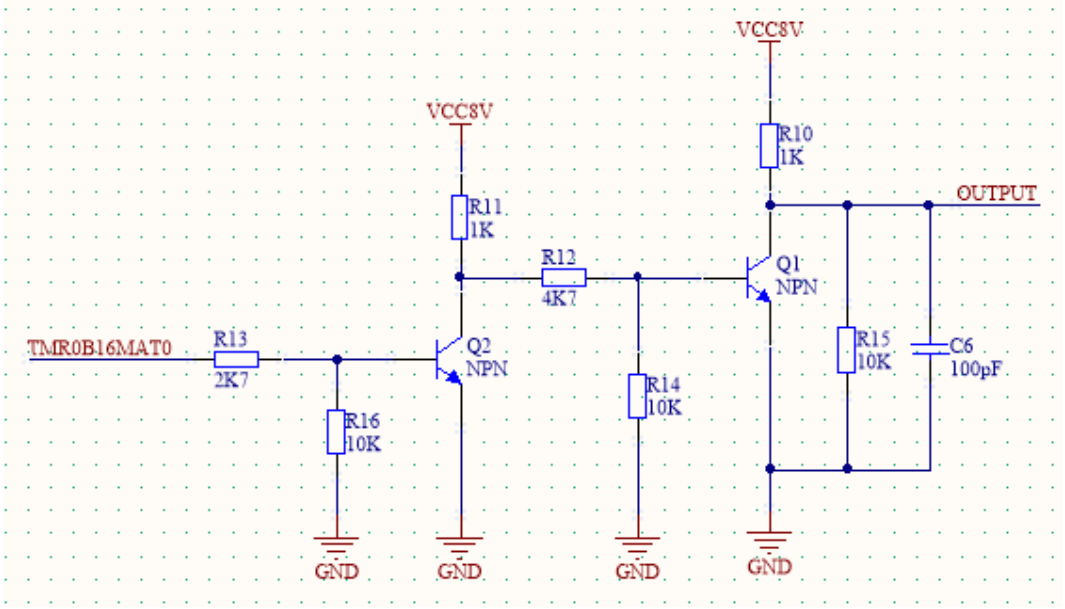

图 4.3 超声波模块发射端硬件电路设计

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。

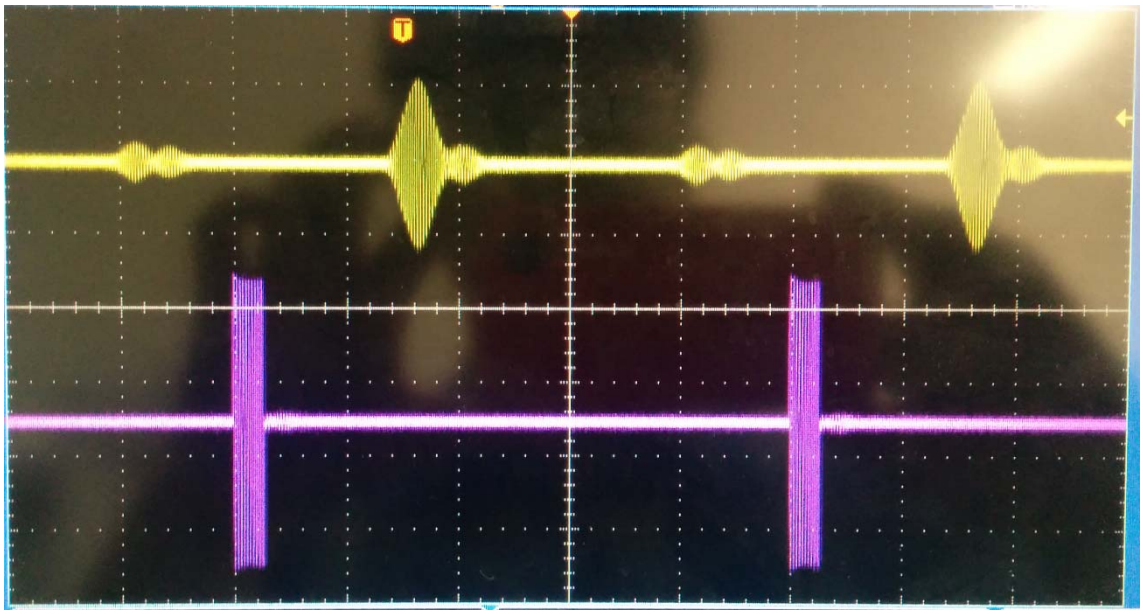

图 4.4 超声波模块输入（紫色）与输出（黄色）信号图

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。

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[11]，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：

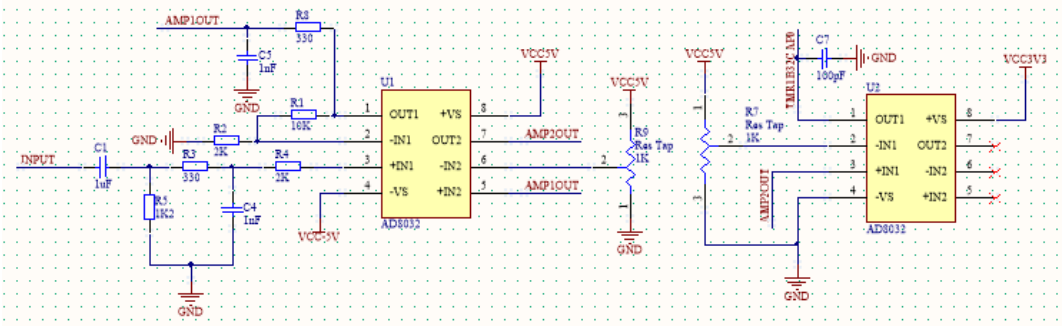

图 4.5 超声波输出模块硬件电路图

经过该电路的输出波形如下：

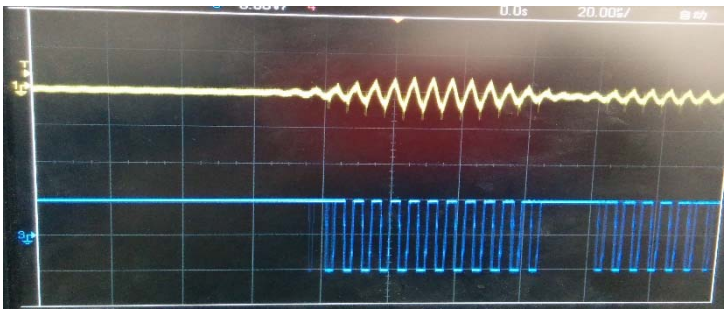

图 4.6 超声波模块输出信号(黄)与经过处理后的输入微控制器的 TTL 信号(蓝)

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：

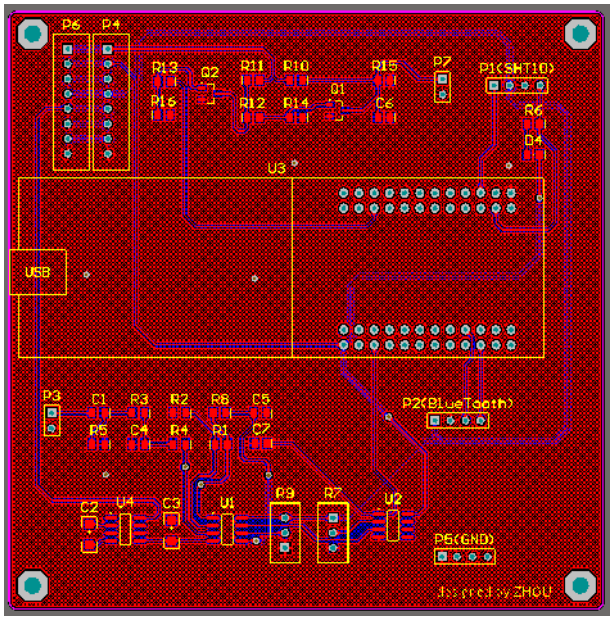

图 4.7 使用 Altium Designer 开发的 PCB 图（2D 视图）

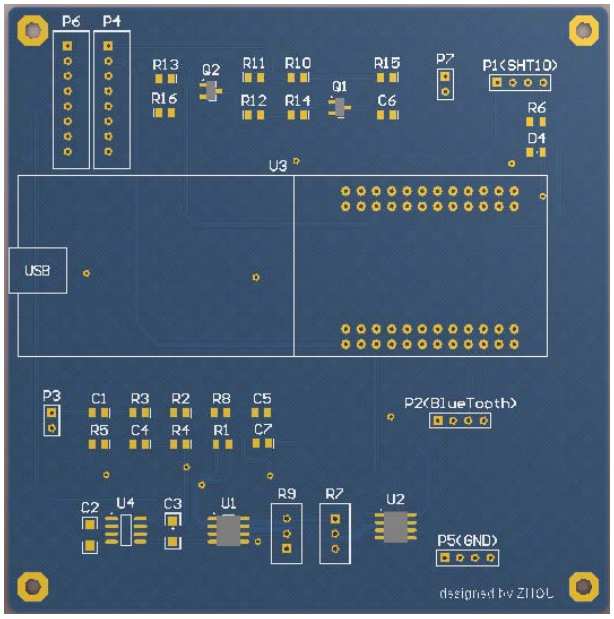

图 4.8 使用 Altium Designer 开发的 PCB 图（3D 视图）

装 订 线

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 I^2C 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。

A. 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：

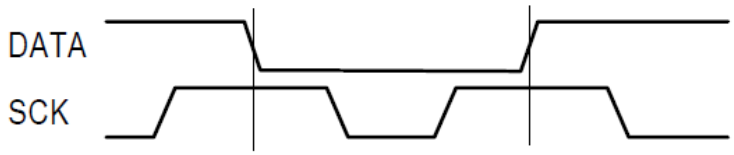

图 5.1 “启动传输”时序

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**

装

B. **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。

订

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）。

线

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：

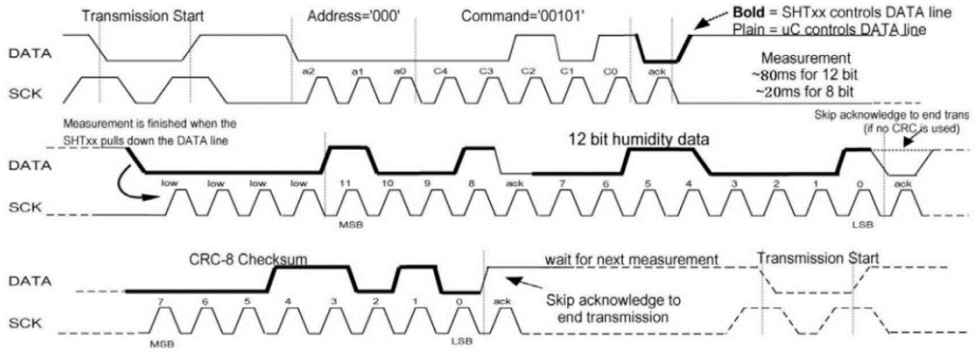

图 5.2 RH 测量时序举例:“0000’1001’0011’0001”=2353=75.79%RH（未包含温度补偿）

C. 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：

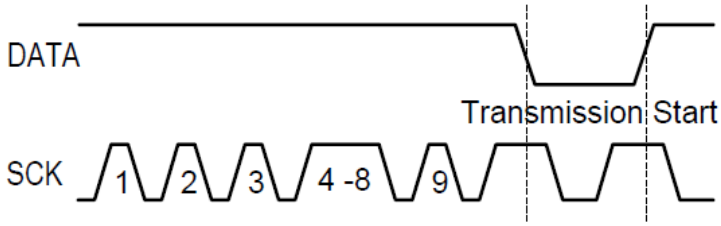

图 5.3 通讯复位时序

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。

A. 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。

B. 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。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：

A. 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）；

B. 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）；

C. 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；

D. 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)。

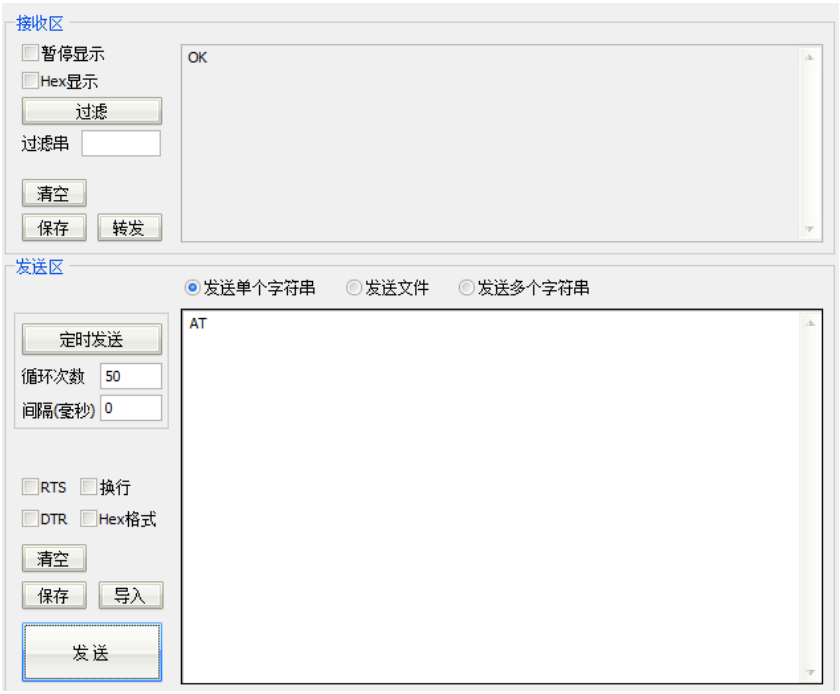

图 5.4 使用串口发送 AT 指令配置蓝牙芯片

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：

A. 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；

B. 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；

C. 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；

D. 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；

E. 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：

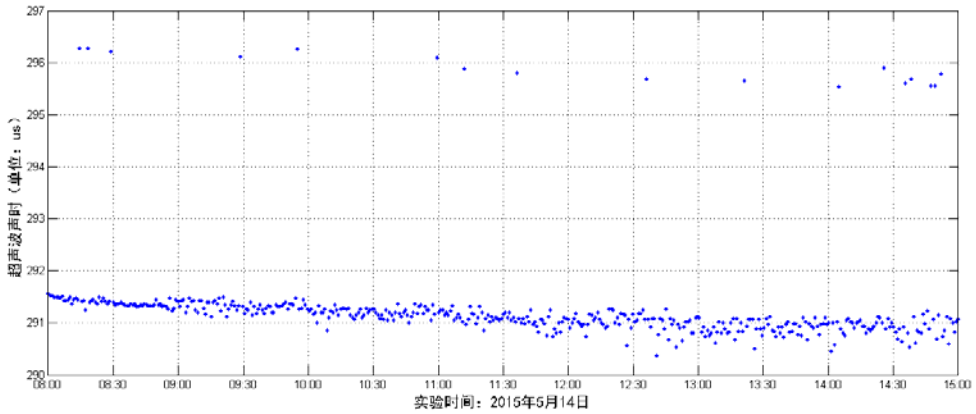

图 5.5 未经手机算法处理的声时数据绘图

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 132（单片机的计数器计 264 为 180kHz），则将该数据相应的加或减 264，再将该数据与之前一个数据进行比较，直到偏差值小于 132，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：

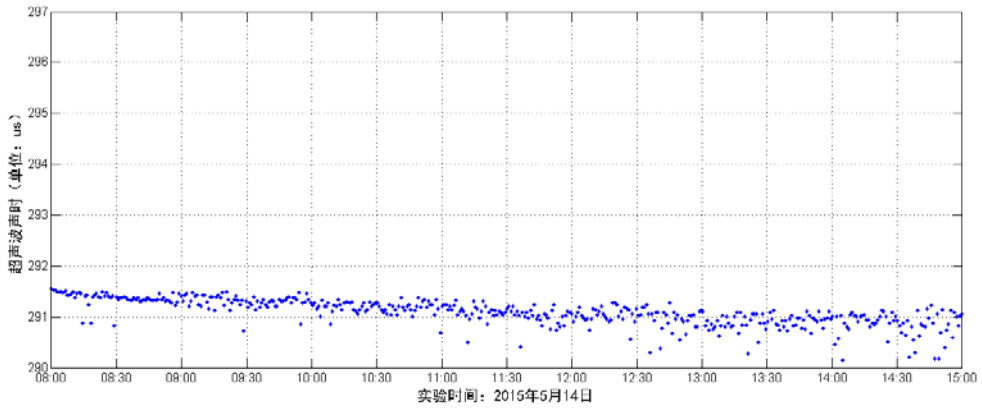

图 5.6 经过手机算法处理后的声时图像

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：

$$temp\_f = temp\_i \times 0.01 - 40 \tag{5.1}$$

$$humi\_f = humi\_i \times 0.0405 - humi\_i^2 \times 2.8 \times 10^{-6} - 4 \tag{5.2}$$

$$lengthTimer = lengthTimerCount \times 2.04 \times 10^{-2} \tag{5.3}$$

其中，$temp\_i$, $humi\_i$, $lengthTimerCount$ 分别为单片机通过蓝牙封装在数据帧中的两字节温度，湿度和声时数据，而 $temp\_f$, $humi\_f$, $lengthTimer$ 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：

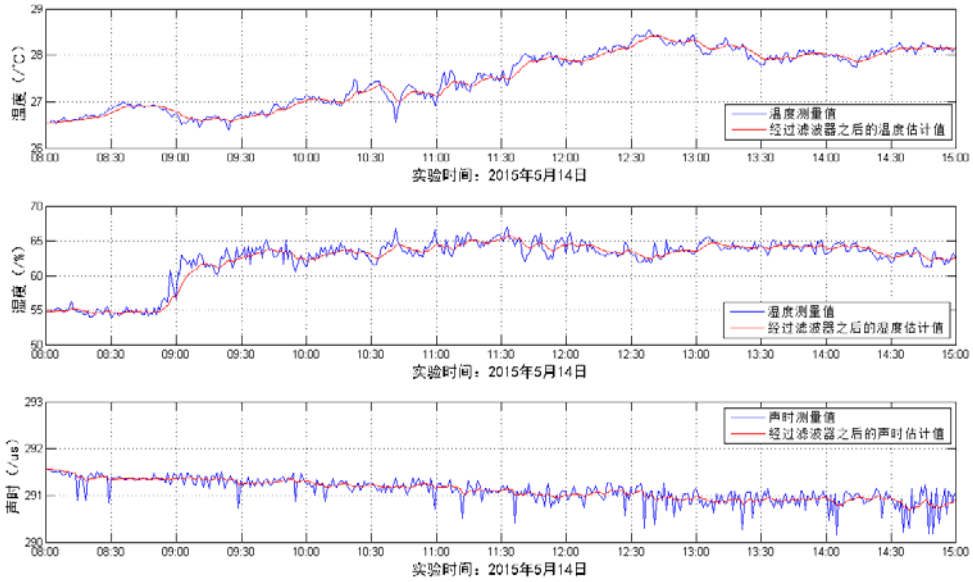

图 5.7 经过卡尔曼滤波器之前与之后的图像

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：

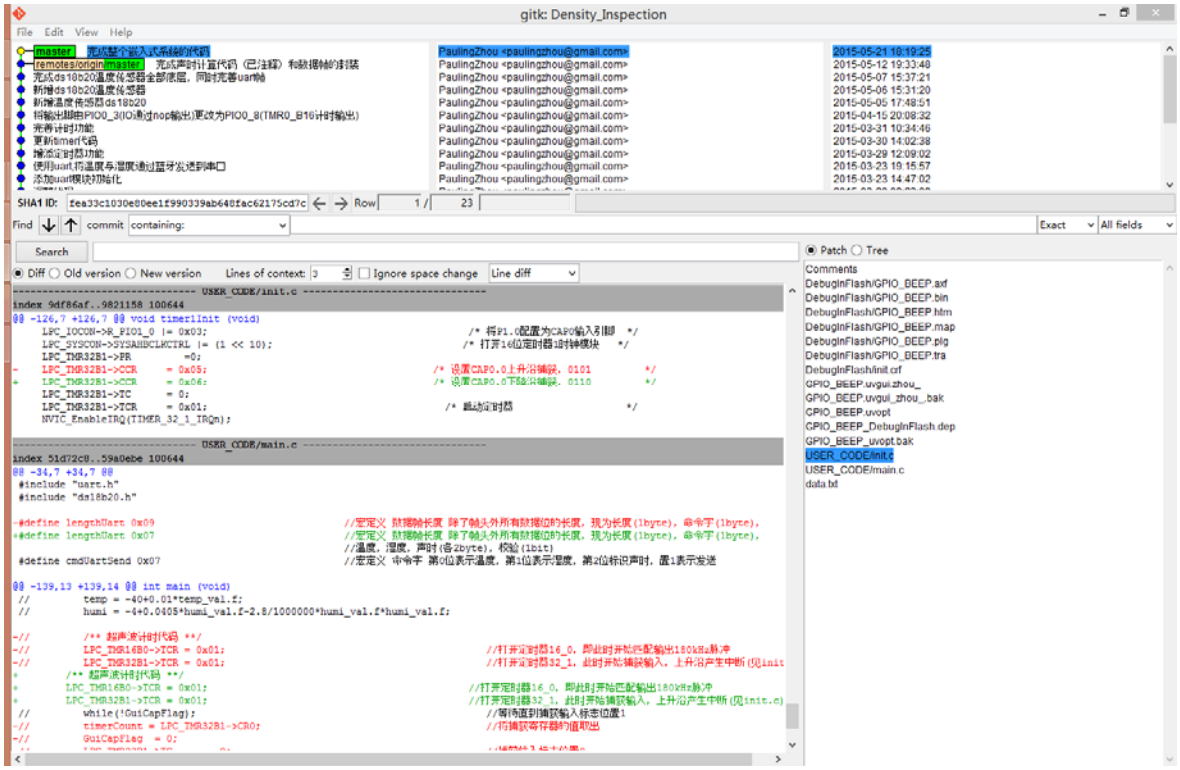

图 5.8 嵌入式系统工程 Git 提交记录

装 订 线

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：

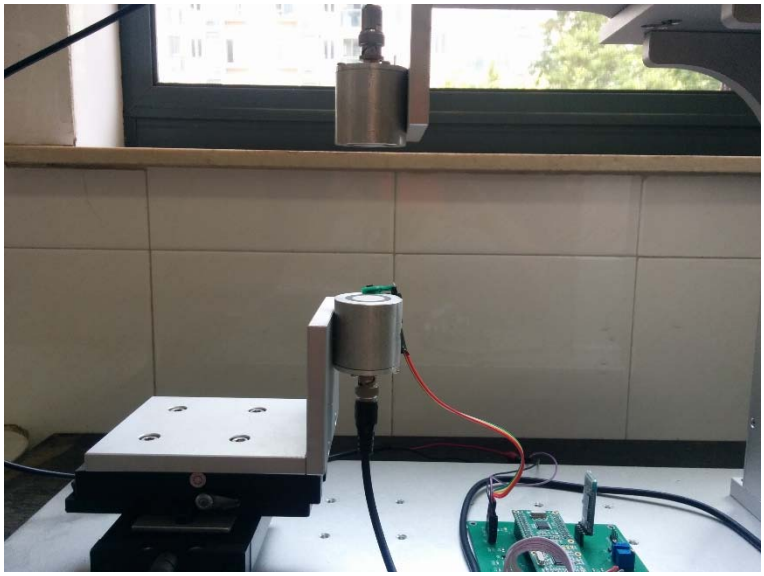

图 6.1 实验装置图

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：

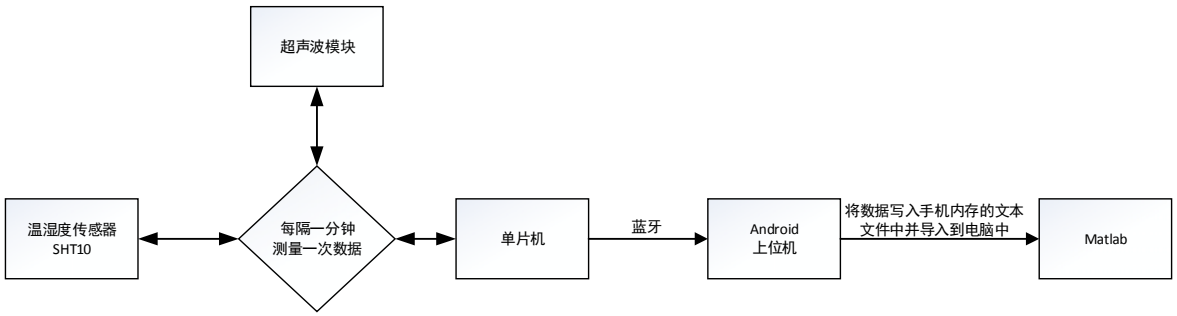

图 6.2 数据获取流程图

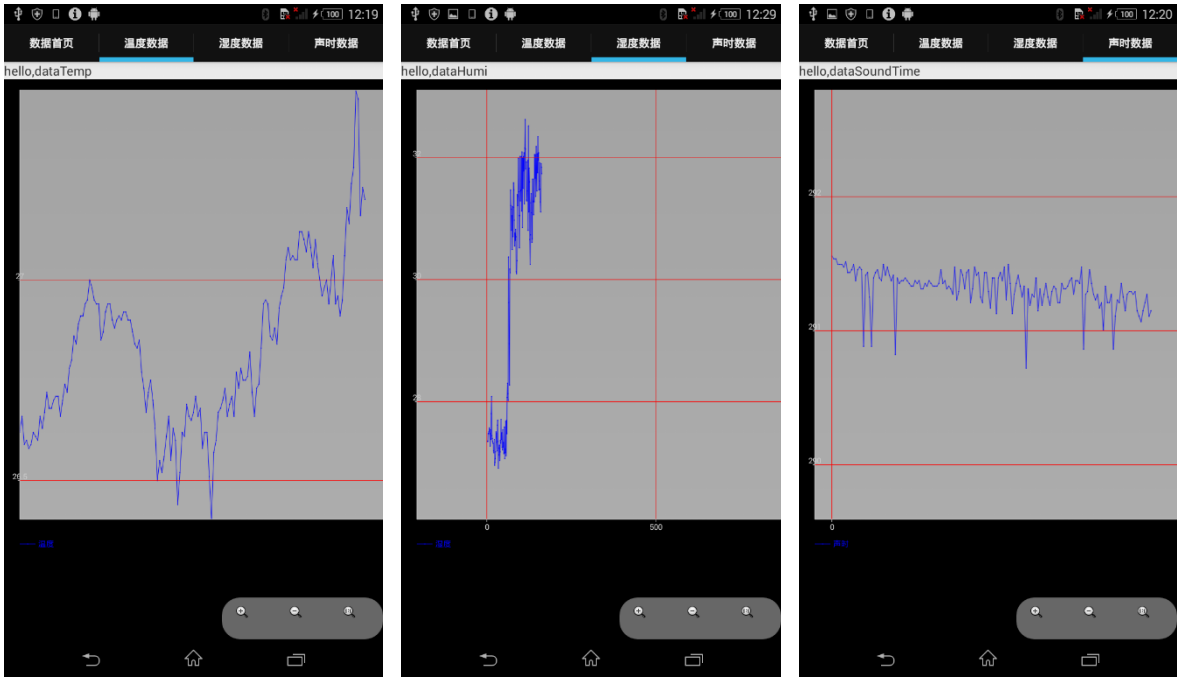

图 6.3 使用 Android 上位机接收数据

6.3 实验数据与理论模型比较并分析

首先通过调整超声波收发模块之间的距离来验证测量出的声速数据是否正确，具体方法是将接收端（或发射端）固定，在竖直方向调整调整发射端（或接收端）在实验台上位置，通过发射端与接收端之间的距离差与计数器测量的声时差来测量超声波的速度，这样的好处是可以排除硬件电路与单片机读写寄存器所带来的时间误差。注意由于在安卓客户端的程序中限制了声时变化的速率，因此，在调整高度时应缓慢操作或将限制声时变化速率的代码注释掉，在此选择注销代码。获取的数据如下表：

表 6.1 声速测量实验

发射模块刻度(cm)	声时数据（字节）	声时数据（us）	超声波声速（m/s）
30	37D4	291.5568	null
35	5372	435.7848	346.6733
40	6F04	579.7680	347.2627
45	8B9C	729.0960	334.8333
50	A735	873.2220	346.9186

这与声音传播速度为 340m/s 的常识基本吻合，其中存在 334m/s 的数据的原因是随着发送与接收模块距离超过一定范围之后，过零检测电路不能检测到包络信号（图 4.4）内部的第一个谐波信号而造成约 5us 的误差。接下来将对采集回的数据进行拟合并与理论模型进行对比。

首先查阅相关文献可得：在正常环境条件下声速与温度近似呈正线性关系，而受湿度的影响很小。[12]即在一定温度下，随着湿度的变化声速几乎为常数。因此本实验只探究温度与声时之间的关系。为了彻底消除湿度对于声速的影响，我们通过图 5.7 选择了从上午九点至下午三点间总计六小时的数据，此时空气湿度基本为定值。

由第二章理论知识可得如下公式：

$$\bar{c} = \sqrt{\frac{\bar{\gamma} R T}{\overline{M}}} \tag{6.1}$$

其中，$\bar{c}$ 为二元混合气体平均声速；$\bar{\gamma}$ 为二元混合气体平均定压定容热容量比；R 为摩尔气体常数；$\overline{M}$ 为混合气体的摩尔质量；T 为热力学温度。

通过查阅文献可得空气的摩尔质量 $\overline{M}$ =28.96g/mol，定压定容热容量比 $\bar{\gamma}$ =1.399[13]，代入式(1)可得空气的绝对温度为：

$$T = \frac{\bar{c}^2 \overline{M}}{\bar{\gamma} R} = 2.49 \times 10^{-3} \bar{c}^2 \tag{6.2}$$

将式(6.2)改写为速度 $\bar{c}$ 关于温度 T 的函数：

$$\bar{c} = 20.04 \times \sqrt{t + 273.15} \tag{6.3}$$

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：

$$\bar{c} = \frac{1}{dataUltraSonicTime} \times 10^5 \tag{6.4}$$

其中 *dataUltraSonicTime* 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：

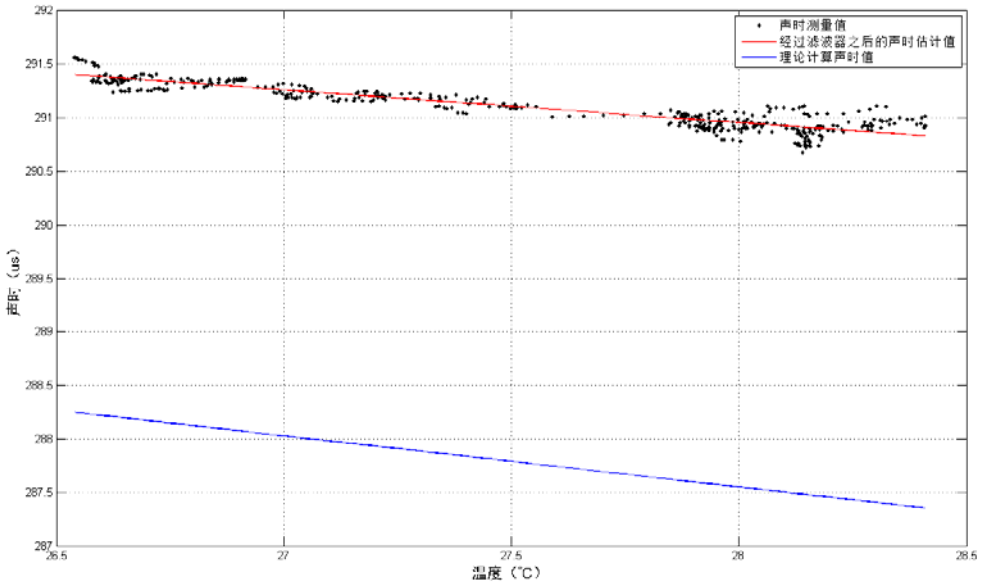

图 6.4 超声波声时理论数据与实测数据

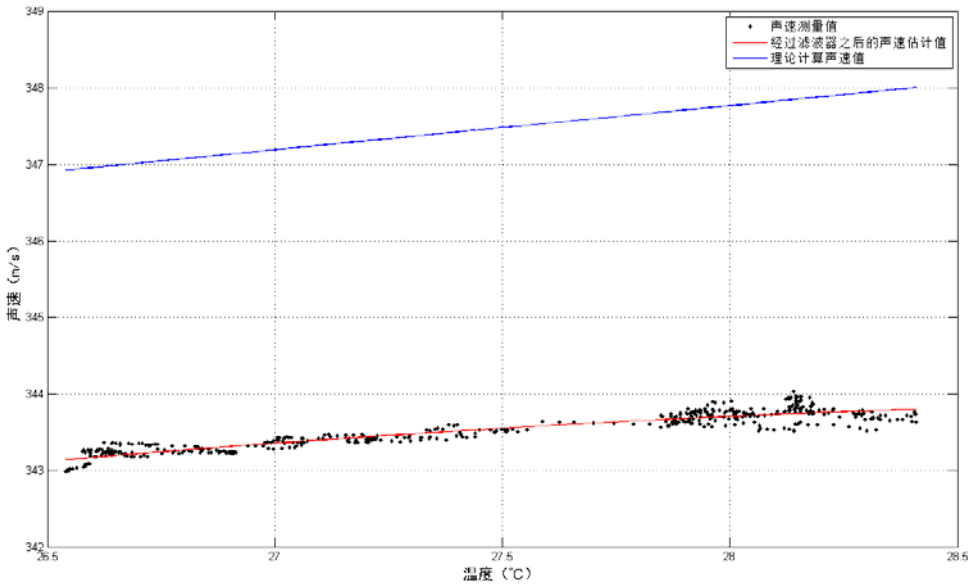

图 6.5 超声波声速理论数据与实际数据

首先进行图 5.7 中的使用卡尔曼滤波算法对温度、湿度以及空气声时最优估计之后的误差分析，发现温度，湿度的测量值与最优估计值之间的误差都控制在了用户手册规定的范围之内（温度为±0.5℃，湿度为±3.5%RH），而声时数据最高则会有近 50ns 的偏差（理论偏差为计数器的最小周期，即±21ns），说明硬件电路仍有会产生误差的地方，通过逐级排查发现可能的原因为由于超声波接收模块输出的电信号经过放大电路之后幅值还是过小，导致使用运放开环电路进行电压比较时不能达到理想情况下的只要高于参考电压就输出正供电电压，只要低于参考电压就输出负供电电压的特性，因此在信号变换的时候就可能产生误差。若想消除误差，可尝试提高第一级放大电路的放大倍数。同时，分析超声波声时和声速的理论与实测图像，我们发现实测的超声波声时的数据与理论数据都是随着温度的上升而下降，但两者相比有近 3us 的误差，相当于测出的声速比理论计算出的声速低了近 3m/s，分析原因猜测是硬件电路中的信号传输的时间滞后和超声波模块中换能晶片将电信号与超声信号相互转换产生的时间滞后。若使用测量声速实验的数据与理论数据进行比较则能发现，若去除这些影响，该系统测量的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。

7 总结与展望

7.1 总结

一个学期的时间很快就过去了，我的本科毕业设计也基本进入了收尾阶段。本次毕业设计是我第一次独立开发一整套包含软硬件的项目，并且通过实验验证了采集到的数据的可靠性。具体来说，本设计完成了如下方面的工作：

(1) 硬件电路的学习与开发：大学前几年我参与的都是一些偏向软件编程方面的项目开发，对硬件电路的开发比较陌生。因此在刚接触硬件电路开发的时候，自己感受到了很大的压力和挑战。通过这次毕业设计，我将大学低年级时学习的电路知识成功应用在电路原理图的设计上，并体会了理想元件与实际元件的差别，例如在使用运算放大器的时候需要考虑单位增益带宽和失调电压，开环接入电路的时候也不是理想情况下的放大倍数无穷大。并独立完成硬件电路项目开发的所有工作，包括原理图文件，原理图库文件，PCB 封装库文件和 PCB 文件的开发。同时与同学共同研究基本掌握了 Altium Designer 软件自带的版本控制功能。

(2) 软件编程的学习与开发：虽然之前有过一些软件开发的经验，但我对于编程的了解仅仅停留在能够正确使用程序语言完成预设的目标。为了进一步提高编程技巧，我在编写安卓客户端程序时学习了面向接口编程的基本准则，即“上层代码不能依赖下层代码，下层代码不能依赖上层代码，两者都应该依赖其抽象”的依赖倒置原则。在实际写代码的过程中，我成功应用该思想写出了接口对象并新建类通过实现该接口对象中的抽象方法完成数据处理方面的工作。同时，在嵌入式系统方面，我自学了 LPC1114 单片机中包括 TIMER、ADC、UART 在内多个外设模块的底层控制寄存器的配置，同时阅读相关技术手册实现了利用 I/O 口模拟串行总线接口，从而对 SHT10 传感器的环境温湿度测量进行控制。

(3) 在最后的实验阶段，我学习了在理想气体中超声波传播的声速方程，查阅相关文献获得参数计算出了超声波声速与环境温度的理论数学模型，并将实测的数据与理论模型进行对比和分析，验证了测量数据的准确性。

总体来说本设计集成了软件与硬件两部分的开发，做出的实物可成功测量环境的温度，湿度以及超声波在一段定长空气中传播的时间，同时使用了卡尔曼滤波算法对测量的数据进行优化得出最优估计值。其中，对于温度和湿度的测量的误差都满足传感器技术手册中给出的参考误差值，说明温湿度传感器的电路设计和程序编写没有问题。但声时数据的误差比理论值高出约 30ns，说明硬件电路仍有需要改进的地方。

7.2 展望

在获得知识的同时本项毕业设计还有很多的不足与可扩展的空间。具体如下：

(1) 硬件方面可以提高电路板的可靠性，例如可以提高超声波模块输出信号的第一级放大电路的放大倍数已达到消除超声波声时测量的误差，同时可以适当缩小电路板的尺寸以达到方便易携带的目的。

(2) 软件方面可以对代码进行优化，例如在安卓客户端的程序设计上可以整合基于 Matlab 编写的程序部分，能够直接使用手机进行数据接收，处理和显示。

(3) 实验方面由于实验次数较小，环境温度的变化范围太小（26℃-28℃），本次实验只大体验证出了声速与温度变化的趋势，但没有对测量产生的误差做进一步的分析和消除，因此需要长时间的进行该项实验以获得比较完整的声速与温度之间的关系曲线图，再将其与理论模型进行对比并修正误差。

(4) 若能得到一台可以精确测量空气污染物浓度的机器，便可以在之前的基础上进行空气污染物浓度与声速之间关系的标定，才能将该系统真正应用于实际。

参考文献

[1] 中华人民共和国环境保护法（自 2015 年 1 月 1 日起施行）

[2] 江福椿,朱昌平,林善明,徐霞,王森. 气体浓度检测技术的现状和应用[J]. 河海大学常州分校学报,2004,01:16-19.

[3] 江福椿,朱昌平,赵帅. 超声技术在气体浓度检测中的应用[J]. 河海大学常州分校学报,2005,02:1-4+34.

[4] J.C. Vyas,V.R. Katti,S.K. Gupta,J.V. Yakhmi. A non-invasive ultrasonic gas sensor for binary gas mixtures[J]. Sensors & Actuators: B. Chemical,2005,1151:.

[5] 阎玉舜,陈亦娟,汤建明. 超声分析二元混合气体浓度的理论及应用[J]. 声学技术,1995,03:105-108.

[6] NXP. LPC111x datasheet[EB/OL]. (2014-03-31)[2015-05-26]. http://www.nxp.com/technical-support-portal/#/tid=,sid=,bt=LPC1114FBD48/302,tab=,p=1,rpp=,sc=,so=,jump=.

[7] Palo Alto. Android takes almost 50% share of worldwide smart phone market[EB/OL]. (2011-08-01)[2015-05-26]. http://www.canalys.com/newsroom/android-takes-almost-50-share-worldwide-smart-phone-market.

[8] Erich Gamma. 设计模式：可复用面向对象软件的基础[M]. 北京：机械工业出版社，2007：21.

[9] Scott Chacon. Pro Git[EB/OL]. (2009-08-27)[2015-05-26]. http://git.oschina.net/progit/.

[10] Sensirion. Datasheet SHT1X[EB/OL]. (2011-12)[2015-05-26]. http://www.sensirion.com/en/products/humidity-temperature/humidity-temperature-sensor-sht1x/.

[11] Analog Devices. Datasheet AD8031/AD8032[EB/OL]. (2006)[2015-05-26]. http://www.analog.com/cn/products/amplifiers/operational-amplifiers/rail-to-rail-amplifiers/ad8032.html#product-evaluationkit

[12] 张燕,陈爱国,高荣贵. 声速的温湿度修正研究[J]. 压电与声光,2011,01:26-29.

[13] 蔡艺剧,黄勇,尹遴,方汉方. 一种新的微量气体浓度检测方法[J]. 化工自动化及仪表,2012,04:477-479+497.

装 订 线

谢辞

从刚开始接触毕业设计的课题到完成毕业设计论文的撰写，总共花费了我这学期所有的时间。虽说在如此短的时间接触了很多新的知识并应用在项目中的确不是一件很轻松的事，但看着自己的论文，这不仅仅是公式和文字的填充，时间和精力的沉淀，更凝结了老师的殷殷期待和谆谆教诲，此时此刻，我怀着感恩的心，想对指导我的师长们说声谢谢。

首先，我要感谢汪镭老师和潘永东老师在项目设计和论文编写方面的指导，汪老师和潘老师教会我很多东西，不仅仅是技术，还有做学问的道理和方向，这些课本上永远学不到的东西让我受益终生。

然后，我要感谢实验室和我一起工作的同学们，尤其是李树同学和王继朗同学在我刚开始接触硬件电路方面开发时对我的指导和帮助，他们给我的指导和鼓励是我最佳的动力。

最后，感谢同济大学 TU-SMART 智能车队给予我的开发支持和技术支持，特别感谢焦剑同学在版本控制方面对我的帮助。

装

订

线