西 南 交 通 大 学

本科毕业设计

基于 Raspberry Pi 的可微信控制的空气质量监控系统

年　级: 2011 级

学　号: 20111743

姓　名: 吕靖

专　业: 电子信息工程

指导老师: 晏寄夫

2015 年 6 月

院系 电气学院电子信息工程系 专 业 电子信息工程

年级 2011 级 姓 名 吕靖

题目 基于 Raspberry Pi 的可微信控制的空气质量监控系统

指导教师

评 语

指导教师 (签章)

评 阅 人

评 语

评 阅 人 (签章)

成 绩

答辩委员会主任 (签章)

年 月 日

毕业设计任务书

班　　级 2011 级电子 1 班 学生姓名 吕　靖 学　号 20111743

发题日期: 2015 年 3 月 9 日　　完成日期: 2015 年 6 月 19 日

题　　目: 基于 Raspberry Pi 的可微信控制的空气质量监控系统

1、本论文的目的、意义: Raspberry Pi，简称Rpi或者RasPi，中文名称为“树莓派”，是一款基于ARM的微型电脑主板，以SD卡为内存硬盘。主要用于教育用途，专为业余兴趣者和想学习编程的年轻人们设计的，提供一个具有最基本功能的，廉价的硬件平台。其尺寸只有信用卡大小，运行速度比台式机稍慢，但已具备了电脑的所有基本功能，只需接通电视机和键盘，就能执行如电子表格，文字处理，玩游戏，播放高清视频等诸多功能。Raspberry Pi的GPU运算能力达到1G pixel每秒，1.5Gtexel每秒或24 GFLOPs的普通运算，并且符合OpenGL 2.0标准，换而言之，它的图形能力跟初代Xbox的图形能力相约。Raspberry Pi有A，B两板。A板提供一个USB接口，配备256MB内存，功率2.5W，500mA； B板提供两个USB接口，HDMI输出和10/100自适应以太网端口，配备512MB内存，功率3.5W，700mA。因此开展“树莓派”的软、硬件开发及应用设计具有重要意义。

本设计利用“树莓派”微型电脑主板，加上一些可以检测空气质量的传感器，收集数据之后可以通过Raspberry Pi传输数据至手机App微信公众帐号，在微信上就可以查看家庭室内空间质量，也可以对Raspberry Pi进行一些简单控制。

2、学生应完成的任务:

① Raspberry Pi 的电路原理;

② 外围接口扩展方法及电路;

③ 操作系统的移植;

④ 应用软件的设计;

⑤ 系统调试及结果分析。

3、论文各部分内容及时间分配(共 16 周)

第一部分 资料收集、熟悉开发工具的使用 (2 周)

第二部分 原理结构框图，芯片选择 (3 周)

第三部分 “树莓派”外围接口电路的设计 (4 周)

第四部分 应用软件的设计 (3 周)

第五部分 系统调试及结果分析 (2 周)

评阅及答辩 论文整理及答辩 (2 周)

备 注: 应交出“树莓派”空气质量检测系统的原理图及硬件电路原理图；系统控制部分硬件原理电路图；注释清楚的源程序清单。

指导教师: 晏寄夫 2015 年 3 月 9 日

审 批 人: 2015 年 6 月 19 日

摘要

在国内空气质量日况愈下和智能家居兴起的背景之下，本论文基于开源硬件树莓派设计与实现了一种可微信远程操控的室内空气质量监控系统。随着物联网行业的愈演愈烈，人们与智能机器的距离也将越来越近，硬件加上智能软件所组成的物联网产品在很大程度上协助着人们追求和体验更好的生活方式。伴随着工业时代所带来的空气污染等环境问题是目前国内亟需解决的问题之一，本项目也就是在这样的背景之下，搭建了一套基于物联网的家庭空气检测和控制中心。整套系统的实现基于时下热门的开源硬件平台，即树莓派和 Arduino 单片机，不同空气传感器将其所采集的电压值输出到 Arduino 模块，进行一些基本的信号处理，如模拟信号转化为数字信号，最后将收集到气体物质含量信息发送至树莓派。搭载了 Linux 系统的树莓派可以直接运行相应的 Python 应用程序，将采集到的传感器数据进行处理和分析，并根据不同的分析结果给出不同的空气质量评定指数。在此基础之上，更是将整套监控系统接入了互联网，使得用户可以在微信公众平台上方便而又快捷得获取空气质量信息。并且实现了在家中发生起火或煤气泄漏等紧急情况下，监控系统能够直接在微信公众平台中向用户发出警报，提醒用户采取相应的措施。

关键词:树莓派； Arduino；智能家居； 开源硬件；空气质量；

Abstract

Under the background of the air quality in China getting worse and the rise of the smart home. In this paper, I design and implement this system which can use WeChat to remote control of indoor air quality monitoring system based on the open source hardware Raspberry Pi. With the networking industry intensified, the distance between people and machine intelligence will also be getting closer. To a large extent, the Internet of things composed of intelligent software in the hardware help people to pursuit and experience a better way of life. The industrial age has brought many environmental problems such as air pollution, which is one of the urgent problems in China. The project is also based on such a background, build the a IOT of family air detection and control center. The whole system realization of nowadays popular open source hardware platform based on, that Raspberry Pi and Arduino micro controller. Different air sensor will be acquired by the voltage output value to Arduino module, and Arduino will do some of the basic signal processing, such as conversion of an analog signal into a digital signal. Finally, these collected gas material content information is transmitted to the Raspberry Pi. Equipped with a Linux system of Raspberry Pi can be run corresponding Python applications directly. The collected sensor data for processing and analysis, and according to the different analysis results are given for different air quality evaluation index. On this basis, it is the whole system of monitoring that access to the Internet, so that users can easily and quickly get air quality information on the WeChat public platform. And if there are some emergency such as the fire or gas leak at home, the monitoring system can send an warning to the user directly in the WeChat public platform, to remind users to take the appropriate measures.

Keywords: Raspberry Pi; Arduino; Smart Home; Open-Source hardware; Air quality;

目录

第一章 绪论

1.1 研究概括

1.1.1 项目背景

在全球变暖和国内工业高速发展的背景下，伴随着人口增长和城镇化的进展，国内空气环境的质量可谓日况愈下。特别是京津冀地区，我国大气污染最严重的区域之一，雾霾状况频发且愈发严重。人类生活离不开空气，空气质量与人体健康的关系密切相关，在政府机构和绿色环保组织的积极宣传和倡导下，公众对国内空气状况的了解和重视程度也越来越深。因此，空气环境状况成为了人民生活的又一热门话题，如何改善空气质量是目前来说最为迫切的环境问题。

随着空气状况的日渐恶化，对空气质量的监控也就变得无比重要，对不同地区的空气进行质量监控，可以为日后的空气状况研究提供数据支持，以便于研究人员能够针对空气质量尽快提出有效的解决方案。而对于个人来说，对家庭居住和出行的周边环境中空气质量的了解和认识也非常重要，这为保护家庭成员的身体健康提供了数据前提和出行保障。而在另一方面，实时掌握家里的环境状态以及家庭险情的自动报警等功能，能够保护全家人的身体健康与财产安全。

1.1.2 项目目的

本项目就是基于日前的国内空气状况所提出的个人解决方案，旨在设计和研究一种可移动性的、快速搭建的、便捷的低成本空气监控设备。在空气监控设备这方面的尝试在国内也尚为少见，据笔者了了解，只有少部分的创业公司想抓住时代的机遇，正尝试于推出空气监控相关的产品。“空气果”就是一款墨迹天气推出的一款智能硬件产品，可以检测空气状况，并能够一键检测，智能体检，甚至于提出推荐解决方案。相当于给大自然戴上一个智能运动手环，凭借硬件来获得大自然的运动数据即空气质量因素。而与此同时，国内热门的互联网手机厂商也在近期推出了小米空气净化器，针对国内的空气污染问题，以满足于消费者所热切需要的室内空气净化功能。

本项目属于创新实践应用，基于国外比较热门的开源硬件 Raspberry Pi(中文名:树莓

派)进行相关空气传感器和互联网接入等相关扩展，从而有了一套从硬件到软件，线下到线上的物联网解决方案。在软件方面，也从空气传感器数据收集和处理到服务器的搭建、网络应用的开发等方面进行了软件开发研究和实践，完成了一个软硬件结合的综合性应用项目。

1.1.3 项目意义

在本项目中，整套监控系统将围绕着树莓派进行传感器和网络等相关扩展，其重点在于提出一种低成本的家居解决方案。这套解决方案首先可以对室内空气状况进行检测和实现对设备的相关控制，而作为应用创新实践项目，也旨在研究影响空气质量状况的空气因素和相关评定标准和实际项目设计与实践。除此之外，树莓派的扩展能力也远不止于此，还可以将树莓派直接作为网关实现网络服务器的搭建，从而使整套监控系统能够接入互联网，使用户能够直接远程查看该设备所检测到的空气状况和其本身的工作状态，进而操控设备能够在不同的空气状况下做出不同的反应。

并且在此次毕业设计中，笔者还将采用微信作为整套系统的控制终端，借助于微信公众账号打通用户和监控系统的交互路径，能够从微信中直接查看室内空气状况或者跳转到网页获得如数据图表等更为细致的信息。与此同时，探究了移动端产品设计和用户体验等方面人文心理类知识，将之运用到具体的产品开发中也让笔者对全面运用知识的能力得到很大的提升。

1.2 发展现状

1.2.1 智能家居

在经历计算机和互联网、移动互联网的兴起之后，物联网被称为信息时代的又一次信息产业技术性革命。随着物联网技术的发展，智能家居应用也将被推向新的发展高度，将智能家居相关设备与互联网连接，可以实现更加丰富的应用模式和场景。近几年物联网设备和应用都如雨后春笋一般层出不穷，国内外都在进行着非常踊跃的尝试与实践，一切的电子设备都开始拥有数据收集和处理能力，将相关数据加以利用可以对人们的生活习惯和生活环境有很大的改善作用。而智能家居作为最贴近于人们的生活起居的一个维度，也是最能够对人们生活产生最大影响的一个环境变量。从而对智能家居中空气环境状况的研究和针对于空气所做的智能监控设备，都是非常有意义和具有前景的方向。

1.2.2 互联网+

2015 年 3 月，中国政府在工作报告中首次提出“互联网+”的行动计划，意在推动移动互联网、云计算、大数据、物联网等与现代制造业结合，促进电子商务、工业互联网和互联网金融健康发展，引导互联网企业拓展国际市场。目前国家为了产业创新，设立了 400 亿元作为新兴产业创业投资引导基金。简单地说，“互联网+”就是以互联网平台为基础，利用互联网通信技术与各行业进行跨界融合。国务院总理李克强还称，“在互联网+的风口上顺势而为，会使中国经济飞起来。”

在这样的背景之下，互联网对国民的生活甚至于国家经济都已经产生了巨大的影响，国家政府也正式在政策上提出利用互联网拉动经济水平的行动计划。由此可见，智能家居等相关物联网产业必将在新的信息产业发展中前景良好，物联网作为互联网中新兴的一种延展形式，也将会拉动经济水平的进步和国民生活的改善。

1.2.3 移动浪潮

所谓移动互联网，就是将移动通信终端全面接入互联网。随着互联网技术的迅速发展，互联网服务早已延伸到人们的生活日常之中，移动互联网尤甚，可以说很多人在一天的生活中是离不开智能手机的。在移动互联网时代，首先需要考虑的就是，对用户来说，移动设备具有更加易用的可操作性和便捷性，用户也会更加习惯于使用手机等移动设备来接入互联网服务，而另一方面就是基于更多的用户需求，进而可以连接更多的产业领域，将更多的设备接入互联网就可以创造出更多的商业价值。

而在此项目中，笔者也将顺应信息时代的移动浪潮，将监控设备的可控制场景不局限于家庭室内，而是将其接入互联网服务。借助于微信公众账号当下这一热门领域，尽量将用户和产品的距离拉到最近，使之能够使用最便捷的方式来操作和查看整个监控系统。

1.3 研究内容

1.3.1 主要模块简介

本论文主要研究空气质量监控系统对空气中气体物质含量的实时监控。包括以下内容:

1. 空气传感器的研究与选定:传感器对整套空气质量监控系统的作用就像人的鼻子一样，灵敏度和稳定性是非常重要的。根据国家室内空气质量标准--GB/T18883-

2002 文件中所定义的对可能影响人体健康以及舒适程度的环境参数，以及结合相关传感器的硬件成本考虑，目前决定采集监控的气体物质包括一氧化碳、烟雾、粉尘和液化气，需要选定不同的传感器对不同的物质进行检测。

2. 树莓派电路板的使用与扩展:树莓派是整套系统的核心硬件，首先是作为控制中心，需要分别控制不同的传感器模块检测空气质量的相关数据，进而对采集到的数据进行具有实际意义的处理分析。再者，树莓派作为软件服务器对整套系统提供运行环境，同时还要处理微信公众号服务器与该系统的数据收发，以及搭建 Web 应用与网页进行功能性的交互。

3. Linux 系统的安装与使用:树莓派可以搭载开源系统 Linux，从而对于 Linux 的学习和使用就必不可少。如何利用 Linux 系统进行快速高效的开发是本部分内容的关键，主要包括两个方面:服务器的搭建和 Web 程序的部署和运行，前者将采用标准的 WSGI 接口来进行设计，后者将采用较为流行的相关技术框架来简化开发流程。

4. 服务器的环境搭建:基于树莓派的 Linux 系统可以直接运行整套监控系统的应用程序，服务器开发语言为 C 和 Python 混合编程，并且会运用 HTML+CSS+JavaScript 进行相关的 Web 页面开发，本部分内容就是对服务器的设计与实现，从而将整套系统的代码部署在该服务器上来与外部通信。

5. Web 应用开发与实现:监控系统的 Web 应用服务器直接运行于搭载在树莓派的 Linux 系统中，用户通过微信公众账号与服务器间接进行功能交互，此部分需要处理微信公众号服务器与本地服务器上所搭载的 Web 应用所进行的交互，还需要进行开发的开发就是用户在微信中所跳转到的网页功能，该部分的应用功能将利用 yeelink 平台进行开发实现。与此同时，还需要实现对家中险情的智能提醒功能。

1.3.2 章节安排

在本毕业设计论文中，将主要分为五个章节进行编写，第一部分就是绪论章节，该部分主要介绍本项目的研究概括，包括项目的背景、目的和意义，然后针对国内外的相关产业发展现状进行了分析，主要介绍了物联网背景下的智能家居，政府所提出的“互联网+”政策行动计划，以及在移动浪潮的影响之下本项目的设计趋向，最后对整套监控系统的主要模块进行了简要介绍。第二部分将详细介绍监控系统的总体设计，主要是从应用场

景和用户需求出发，注重于系统的的使用产品和相应的功能分析，在完成需求分析之后给出系统的具体实现思路和开发流程，在主控制中心和功能模块化的思想指导下，选择相应的传感器和硬件平台，从而在此基础之上开发软件应用。在第三章中，本论文将详细介绍整套系统的硬件架构，从介绍主控制中心即树莓派开始，分模块解释了传感器硬件的相关设计原理，进而介绍硬件的连接电路和每个模块的作用。主要包括树莓派的 GPIO 和串口与其他模块的数据传输，以及模数转换模块模拟信号转换为数字信号。第四部分会详细描述监控的本地软件设计，此部分主要关注于在树莓派上对传感器数据的采集和处理，包括操作系统的安装和使用，以及主控制流程的设计与实现，并且在空气数据采集之后进行对数据的实意化处理和有效数据的存储。而后进入最后一部分即第五章，将会详细论述整套系统如何接入互联网，并且与微信公众号和 yeelink 两个平台进行功能交互，其中包括如何搭建一个 WSGI 风格的服务器，并将服务器所接入的两个平台上的功能设计和开发都进行了详细介绍。

第二章 监控系统的总体设计

2.1 系统功能分析

2.1.1 应用场景

在我们的生活中的不同场景里，人们总是会有各种各样的需求出现，不管是基于人性本身的贪痴嗔，亦或是生理和心理上的自然诉求，由此所衍生出来的相应的商业消费品或者科技创新，都在满足着人们对更美好生活的向往，也在无形中影响着我们的世界。有了这样的思考，笔者不由得联想到智能家居在未来生活中所扮演的角色，比如说在回家的路上，拿出手机就可以提前关闭窗户，打开空调降低室内温度，以便于一回到家便可以享受清爽的环境。更可能的是，时刻查看室内空气质量指数，再由此决定是否开启空气净化器净化室内环境，室外的世界在日渐恶化的环境中充斥着雾霾，回到家难道不应该享受一下干净而又安逸的空气吗？而且在家居环境中，一切无形的隐患也在威胁着全家人的身体安全，比如说用户已经进入睡眠，家中却不幸地发生了煤气泄露，如果能够有相应的装置在一探测到液化气就自动发出报警，那可能就可以挽救全家人的性命。又或者出门在外，家里却不幸发生了火灾，如果能够提前发现火情的话，也可以及时得通知消防队前来灭火，保护财产安全。

2.1.2 用户需求

在上一节中笔者基于家居环境提出了几种假设的应用场景，对用户来说，身体健康和家庭财产安全是处于首要位置的，如何保证在最适当的时候提供最佳的解决方案也是用户最迫切的需求。最佳解决方案可以分为两种，一种属于用户的主动行为，包括查看信息和控制设备，当用户有了掌控家庭环境的需求的时候，用户可以通过手机终端来直接查看传感器所检测到的具体数据，以及监控系统根据相应的空气质量数据给出的空气状况评定级数。另一种情况属于设备的主动行为，用户被动接受信息。当然如果家中出现紧急情况，监控设备在用户没有进行操作的时候就主动向用户发出警报信息，从而提醒用户查看家中情况并采取相应的措施。除此之外，监控系统也应该具有一定的智能危机处理能力，比如该用户并没有及时看到系统所推送的报警信息，所以监控系统应该要在等待用户采取

操作的同时，设置相应的等待阀值，当用户在一定时间内没有响应操作的话，智能系统就应该要主动给出解决方案。系统根据相应的权限和设置的功能可以达到不同的自动化解决能力。比如探测到家中火情发生，系统可以在第一时间打开火灾防护系统，根据火灾发生地点自动选择水的喷洒处，又或者在于此同时，主动通知消防队并且给出详细的家庭位置信息。

2.2 系统开发流程

2.2.1 总体设计思路

从用户需求出发，整套系统需要完成向用户展示传感器的数据和相应的空气质量指数、树莓派工作状态的显示和控制以及空气状况异常时的智能提醒和报警等功能。根据国家室内空气质量标准，选取相应的气体物质传感器进行数据收集，并且对采集到的数据进行空气质量评定。本质量监控系统也将采用模块化的设计方式，将系统分为终端采集模块即传感器模块、数据处理模块以及控制中心模块。在软件上也需要实现功能的模块化，使不同的模块尽可能解耦，数据和功能处理分离。为了能够实现用户能够在移动端直接控制本空气监控系统，则需要将树莓派通过 WiFi 连接 Internet，分为两种方式实现了数据的展示和信息的推送。

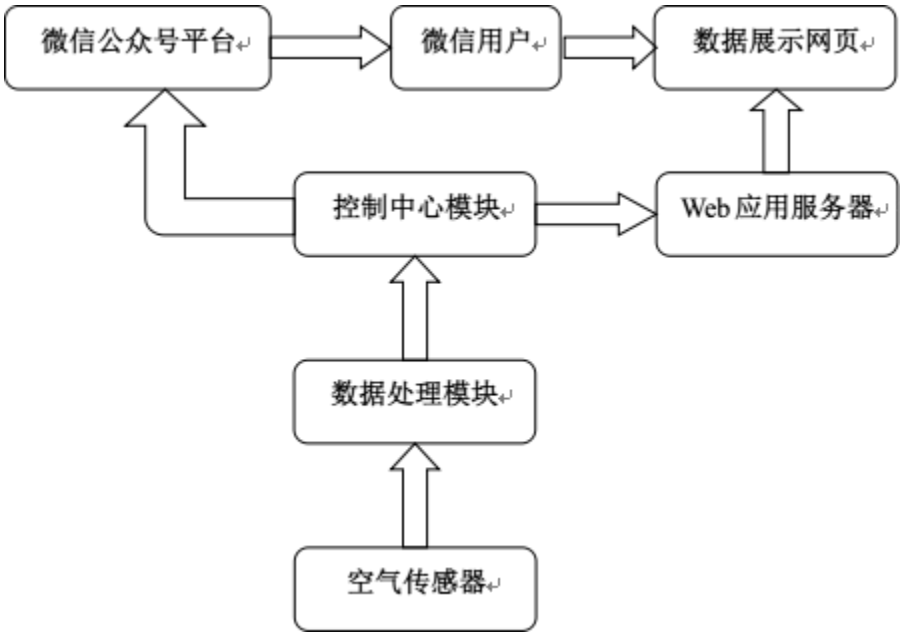

图 2-1 模块结构图

2.2.2 硬件模块设计

目前，国内外都对智能家居进行了广泛而深入的研究及建设，然而大多数开发组都是自己设计电路、焊接元件，这就将很多非电子专业的电脑爱好者限制在了门外。本文在比较了热门集中集成电路板之后选定了树莓派作为监控系统的硬件平台，并且介绍了以树莓派为中心的空气监控系统的设计与实现，树莓派的使用可以大大降低了普通爱好者及个人开发智能家居的难度。

在研究树莓派的硬件构造以及内部电路之后，便将不同的硬件功能模块与树莓派相对应的接口连接起来，主要分为传感器采集模块和模数转换等数据处理模块，以及树莓派自身工作所需的硬件接口设置，如无线网卡、USB 转串口、SD 卡读取、I^2C 总线等等。针对于每种传感器，也会有对应的硬件电路连接，而且在不同的电路处理之下有不同的效果，因此需要对传感器进行一定的实验，已测试传感器对其他物质的检测性能。其实在准备好硬件电路的同时便可以调试软件的效果，更甚至于根据不同的情况同时调整硬件和软件，两者相辅相成。

2.2.3 软件模块设计

软件的设计与实现将遵循两个原则，首先就是测试先行，先将不同的硬件模块调通，保证每个模块都能够正确运行；另一个就是功能解耦，确保数据处理和功能逻辑的分离，每一个模块都互不依赖，执行单一职责。首先需要在树莓派上安装移植 Linux 操作系统，从而才有了接下来进行软件开发的基础。本次软件开发将采用 Python 和 C 语言混合编程，主要是基于不同层次的功能开发框架有不同的优先选择，并且这两种语言在不同的平台上拥有不一样的性能表现。树莓派在获取传感器数据之后，需要对其进行数据的分析，根据不同气体物质的浓度共同判定当前的空气质量指数。采集空气中相关物质的数据之后，需要将其进行存储以便于后续的数据使用，目前只是将数据简单存储于文本文件中作为记录，而不是存储于数据库。与此同时，在接入互联网的 Web 应用开发分为两个部分，第一个部分是基于微信公众平台的服务器端开发，完成微信公众号的功能设计和思想；第二部分是将传感器数据上传至 yeeklink 平台进行图形化展示，以便于用户能够更方便更直观得查看数据。

第三章 监控系统的硬件架构

3.1 控制中心硬件设计

3.1.1 硬件平台的选择

随着集成电路技术的发展，计算机的体积也在不断得缩小，而各方面的性能也在飞速提高。物联网和集成电路的快速发展以及人们对居住环境舒适、安全、便捷等要求的不断提高，智能家居、可穿戴设备等的概念已经呼之欲出，互联网和智能设备已经逐渐走进人们生产生活的各个领域。而在这个过程当中，基于不同的硬件产品出现过不同硬件平台，以下将分别介绍单片机、Arduino、树莓派(Raspberry Pi)这几种硬件:

单片机(Microcontrollers)是一种集成电路芯片，直接将计算机的 CPU、RAM、ROM、定时数器和多种 I/O 接口集成在一片大规模集成电路上，形成小而完善的芯片级的微型计算机系统。单片机电路主要包括了具有数据处理能力的中央处理器 CPU、随机存储器 RAM、只读存储器 ROM、多种 I/O 口和中断系统、定时器/计数器等功能模块，当然还可能包括显示驱动电路、脉宽调制电路、模拟多路转换器、A/D 转换器等电路。从上世纪 80 年代，在经过不断的研究和发展，历经 4 位，8 位，到现在的 16 位及 32 位，甚至 64 位。产品的成熟度，以及 投入厂商之多，应用范围之广，真可谓之空前。单片机也由当时的 4 位、8 位单片机，发展到现在的 300M 的高速单片机。

Arduino 是一款便捷灵活、方便上手的开源电子原型平台，包含硬件和软件两部分，Arduino 封装了很多函数和大量的传感器函数库。降低了开发难度，即使不懂软件开发和电子的人也可以很快的使用 Arduino 设计电子产品。从本质上来说，Arduino 的内部结构主要就是以 AVR 单片机为核心控制器的单片机应用开发板，但是 Arduino 开发人员开发设计了很多简单的函数，还有许多具有相应功能的应用库都能够直接使用，而不用去直接操作单片机底层的寄存器。这使得没有很好的单片机基础的人员也可以使用 Arduino 做出自己想要的东西。Arduino 的开发人员还开发了一个非常简洁的具有图形界面的 IDE(集成开发环境)，也就是写代码，编译，调试，下载的上位机软件。

树莓派(英文名为“Raspberry Pi”)是一种基于 ARM 的只有信用卡大小的微型卡片

式电脑，初衷是为了学生计算机编程教育而设计。树莓派可以直接在 SD 卡中安装和运行 Linux 系统，SD 卡作为内存和硬盘使用。卡片主板周围有两个 USB 接口，可以直接连接键盘、鼠标，还有一个以太网接口可以直接插入网线连接局域网或互联网。树莓派同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口，这意味着可以直接连接电视或显示屏显示相应的图像界面，具备 PC 机的所有基本功能，如电子表格、文字处理、玩游戏、播放高清视频等等。值得一提的是树莓派使用的是一颗运行在 700MHZ 的 ARM11 芯片，可以直接运行完整的操作系统，如 Debian 等常见 Linux 发行版。这意味着开发人员可以使用自己最熟练的语言(如 Python、Java 等)和熟悉的库来进行开发，同时后台运行多个进程也毫无压力。

3.1.2 树莓派硬件结构

前面提到，随着近几年智能手机硬件的发展，廉价而性能足够强劲的移动处理器芯片为树莓派的问世铺平了道路。剑桥大学 Eben Upton 博士成立了英国树莓派基金会，宣布树莓派的目标为“造价 25 美元，运行 Linux，信用卡尺寸，可以连接电视机，有高清视频播放能力”。由于其强大的计算能力，丰富的外部扩展，传播广泛社区完善和较为低廉的价格等特点而被选作本毕业设计的硬件开发平台。

下面是树莓派宏观接口图:

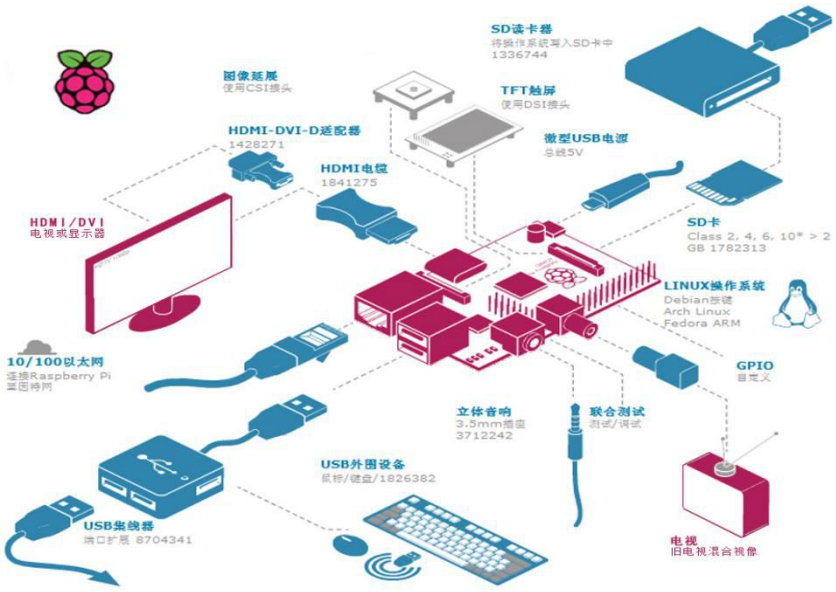

图 3-1 树莓派宏观接口图

从本质上来说，树莓派其实就相当于是一块电脑主板了，将计算机所需的相关接口都集成到了一块板子上。最核心的就是一块 BCM2835 片上系统，其中包括了一个 ARM11 处理器和一块 VideoCore IV GPU，还有 512M RAM（A 型板内存为 256M）。树莓派上没有内置的长期存储设备，可以将烧录好的的 SD 卡插入 SD 插槽作为内存和硬盘使用，SD 卡上可以直接运行相应的 Linux 操作系统。SD 插槽的旁边是一个 Micro-USB 电源插孔，可以由此引入 5V 的电源，当然比较极端的做法是通过 GPIO 接口的 2 号（VCC）和 3 号（GND）引脚直接提供 5V 电源。树莓派上有一排 26 针的扩展接口，也就是常说的 GPIO 口，可以作用通用输入输出和特殊功能处理。树莓派还提供了两种视频接口方式，即 HDMI 和 RCA 接口。比较明显可以看到的是两个叠在一起的 USB 接口，可以用来连接鼠标和键盘，通过一个有源的 USB-HUB 可以扩展多个 USB 接口。在 USB 接口旁边是一个 10/100 M 以太网接口，可以用来连接网线连入局域网或者互联网。另外有三个不常用的外设，即音响/测试接口、JTAG 接口和 TFT 触摸屏接口，可以直接接入液晶显示屏。

核心芯片 BCM2835：这是博通（Broadcom）公司为树莓派专供的核心芯片，具备了第四代 VideoCore 技术的高清嵌入式多媒体应用处理器 BCM2835 是一种低成本，全高清的多媒体应用处理器，适用于需要高品质多媒体性能的移动和嵌入式应用设备。

下面是该芯片的内部结构图：

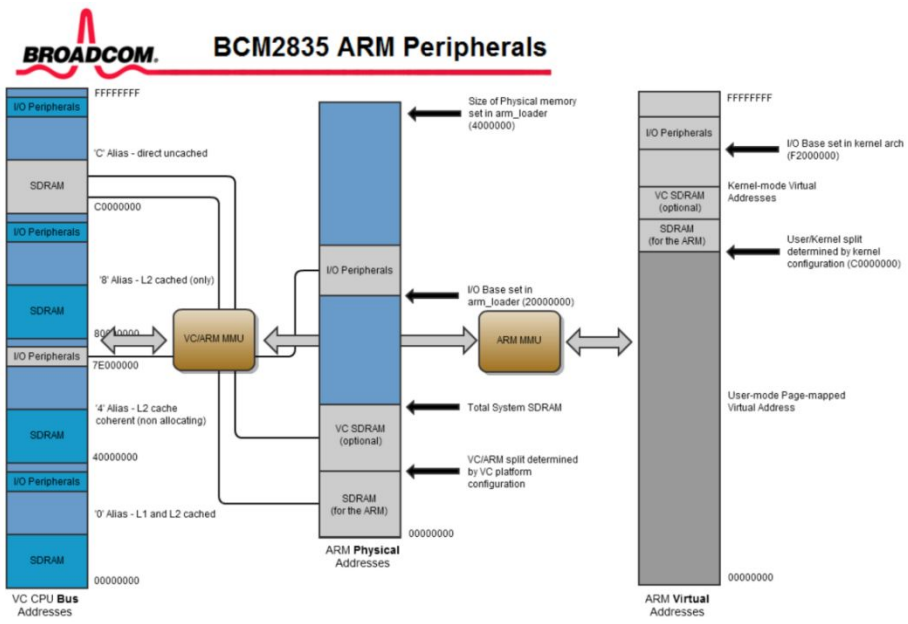

图 3-2 BCM2835 芯片内部结构图

- 低功耗 ARM1176JZ-F 应用处理器
- 双核第四代 VideoCore 多媒体协同处理器
- 1080p30 帧每秒全高清视编码/解码
- 高性能视频输出。1080p60 帧每秒持续高分辨率 LCD、hdmi 输出
- 低功耗，高性能 OpenGL-ES1.1/2.0VideoCoreGPU
- 先进的图像传感器流水线(ISP)长达 20 万像素的摄像头高达 220 万像素每秒

这里需要额外提到的是，树莓派在电源上串联了一个自恢复保险(PTC)进去，试图防止出现短路、过流等故障。PTC 本来是一种简单的过流保护器件，过流时发热造成内阻上升抵抗过流，异常解除后可以冷却让内阻下降，恢复设备工作。但 PTC 简单就有简单的问题:PTC 自身的电阻不低(实测很可能有 0.2Ω以上)，串接在电源上容易把电源电压拉下 0.3-0.5V 左右的程度。这对于 5V 这个低压来说，电源跌落这个程度就很成问题了，极易造成依赖 5V 的 USB 等外设欠压异常。事实上如果供电电压是 5.2-5.4 等比 5V 整数稍微高点的规格，那么树莓派得到的电压还可能在 5V 上。如果电源老老实实提供 5V，那树莓派就很可能无法正常供电，笔者最终也是采用了 5V、2A 的电源适配器为树莓派供电。

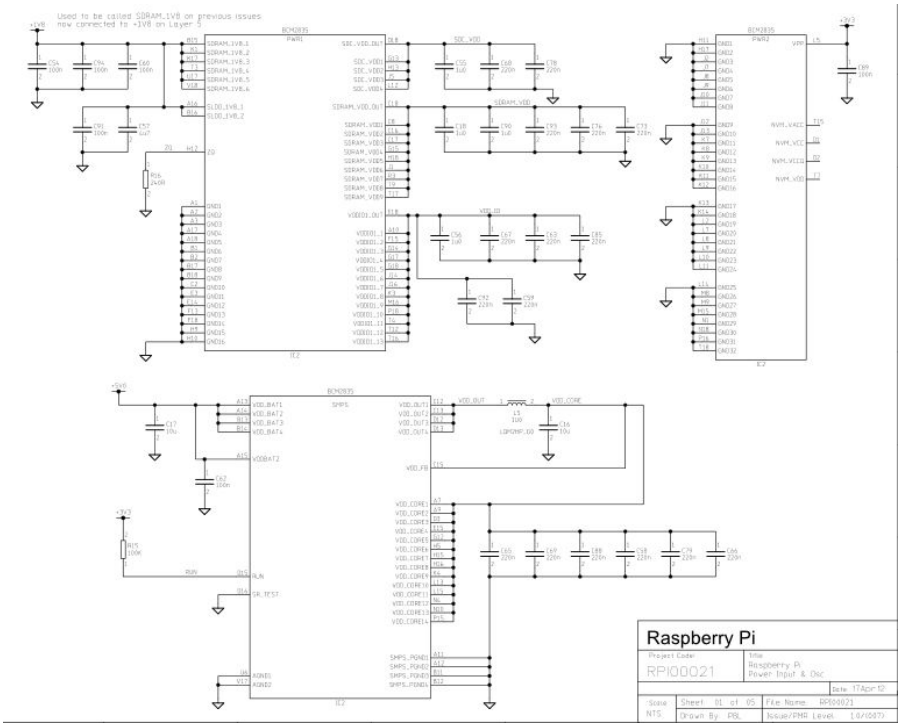

图 3-3 树莓派内部原理图

3.2 采集模块硬件设计

3.2.1 温湿度传感器

数字温湿度传感器 DHT11 可以直接信号输出具备已校准的数字信号，使用了专用的数字采集和温湿度传感技术。每个 DHT11 传感器都包括一个电阻式感湿元件和一个 NTC 测温元件。传感器在检测信号的处理过程还需要调用在 OTP 内存中存储的校准系数，该系数均已在极为精确的湿度校验室校准完毕。产品为 4 针单排引脚封装。连接方便，特殊封装形式可根据用户需求而提供。

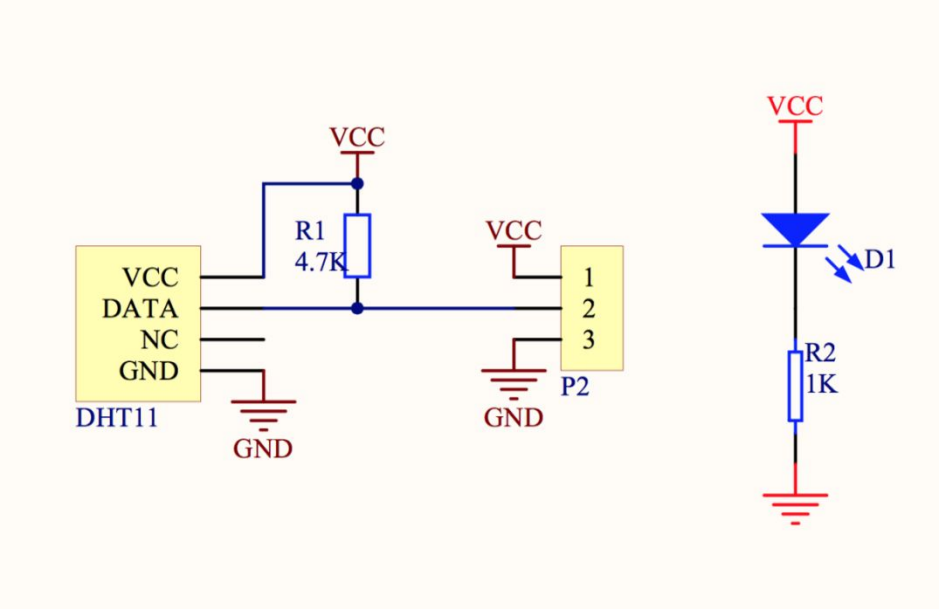

图 3-4 DHT11 模块原理图

DHT11 模块使用单线双向的串行接口与微处理器进行数据传输。一次完整的数据传输为 40 字节，高位先出。数据传送正确时，校验和为传感器采集到的温湿度数据的整数和小数部分总和的最后 8 位。

表格 3-1 DHT11 模块接线说明

VCC	外接 3.3V-5V
GND	外接 GND
DATA	传感器数字量输出接口

3.2.2 可燃性气体传感器

MQ 系列的气体传感器使用二氧化锡(SnO2)，这种气敏材料的电导率跟空气中可燃性气体浓度的大小密切相关。由于电导率的变化所输出相对应的电压信号可以反映出当前空气中的气体浓度。

其中 MQ-5 气体传感器对丁烷、丙烷、甲烷等可燃性气体的灵敏度较高，特别是天然气。只需要简单的驱动电路即可工作，具有长寿命、低成本等特点，主要应用于煤气泄漏报警器等便携式气体检测器。而 MQ-2 气体传感器对液化气、丙烷、氢气的灵敏度比较高，在本项目中主要针对于烟雾的检测。结合 MQ-2 和 MQ-5 这两种传感器就可以在一定程度上检测出火灾或者煤气泄漏等家庭险情。

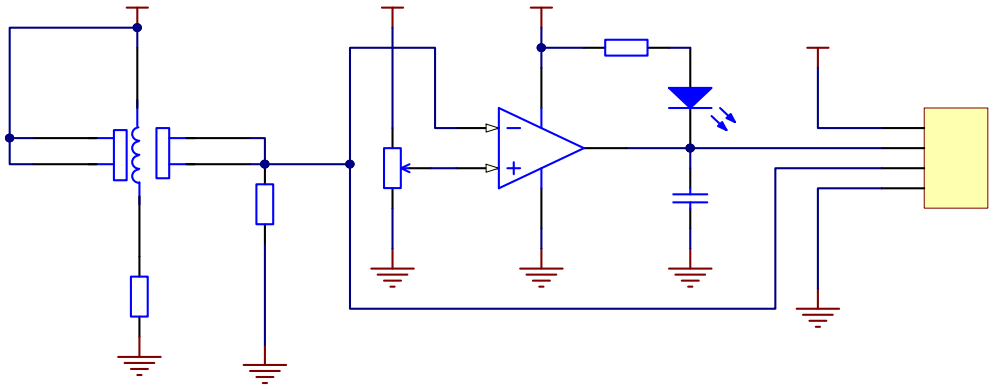

图 3-5 气体传感器原理图

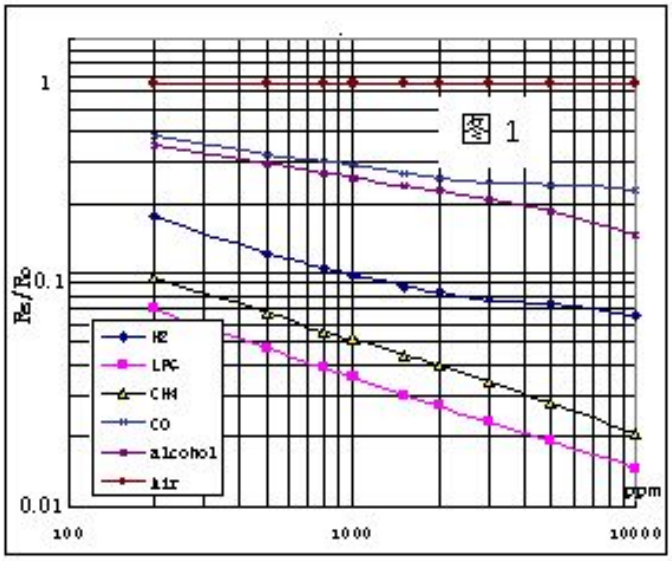

图 3-6 传感器典型的灵敏度特性曲线

图中纵坐标为传感器的电阻比(Rs/Ro)，横坐标为气体浓度。Rs 表示传感器在不同

浓度气体中的电阻值 Ro 表示传感器在洁净空气中的电阻值。图中所有测试都是在标准试验条件下完成的。

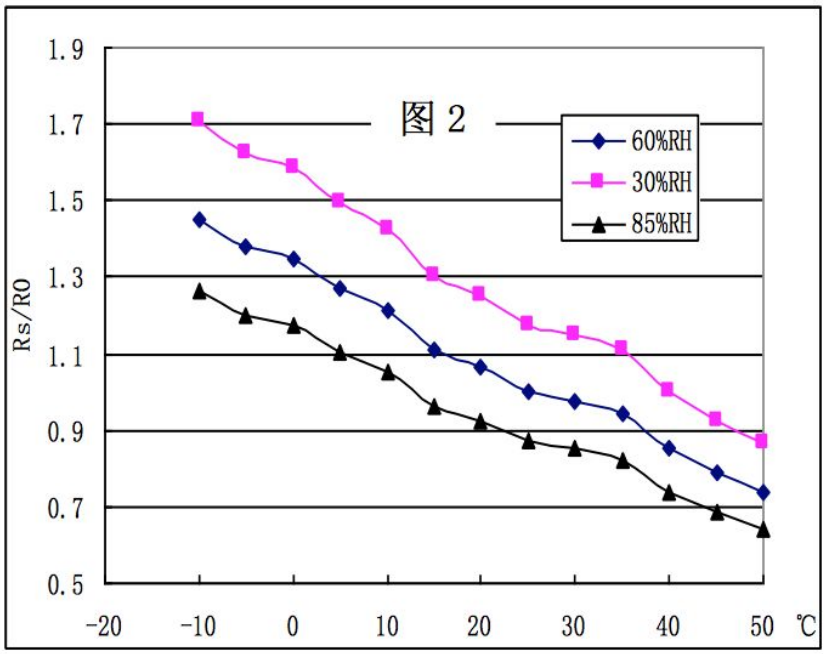

图 3-7 传感器典型的温度、湿度特性曲线图

图中纵坐标是传感器的电阻比(Rs/Ro)。Rs 表示在含 1000ppm 丙烷、不同温/湿度下传感器的电阻值。Ro 表示在含 1000ppm 丙烷、20°C/65%RH 环境条件下传感器的电阻值。

3.2.3 粉尘浓度传感器

GP2Y1010AU0F 是一种基于光学传感系统的粉尘浓度传感器，该传感器中心有个洞可以让空气自由流过，将一个红外发光二极管和光电晶体管对角放置，对其定向发射 LED 光，就可以检测到在空气中的灰尘反射光，从而判断灰尘的含量。GP2Y1010AU0F 粉尘传感器尤其能够能够有效地检测到非常细的颗粒，如香烟烟雾。此外，它还可以通过区分输出电压的脉冲模式用来判断室内灰尘烟雾。该传感器输出的模拟电压与粉尘浓度成正比关系，可以检测到的最小浓度为 0.5V/0.1mg/m^3。

关于检出方法:

GP2Y1010AU 通电后会在短时间内稳定工作，其输出电压的绝对值，并不能够判定出是否有无灰尘，而且可以根据输出电平大小和时间的变化判断出灰尘的种类。

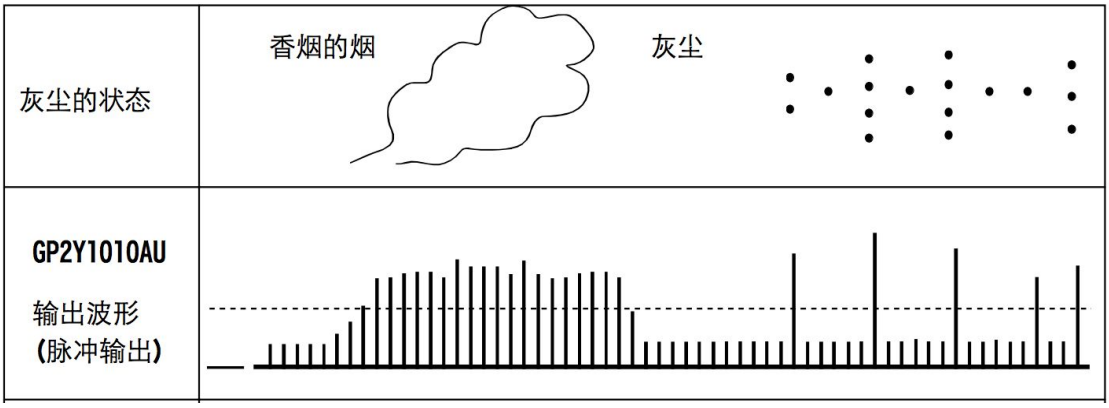

图 3-8 GP2Y1010AU 对不同灰尘的输出波形

GP2Y1010AU 的内部结构图如下:

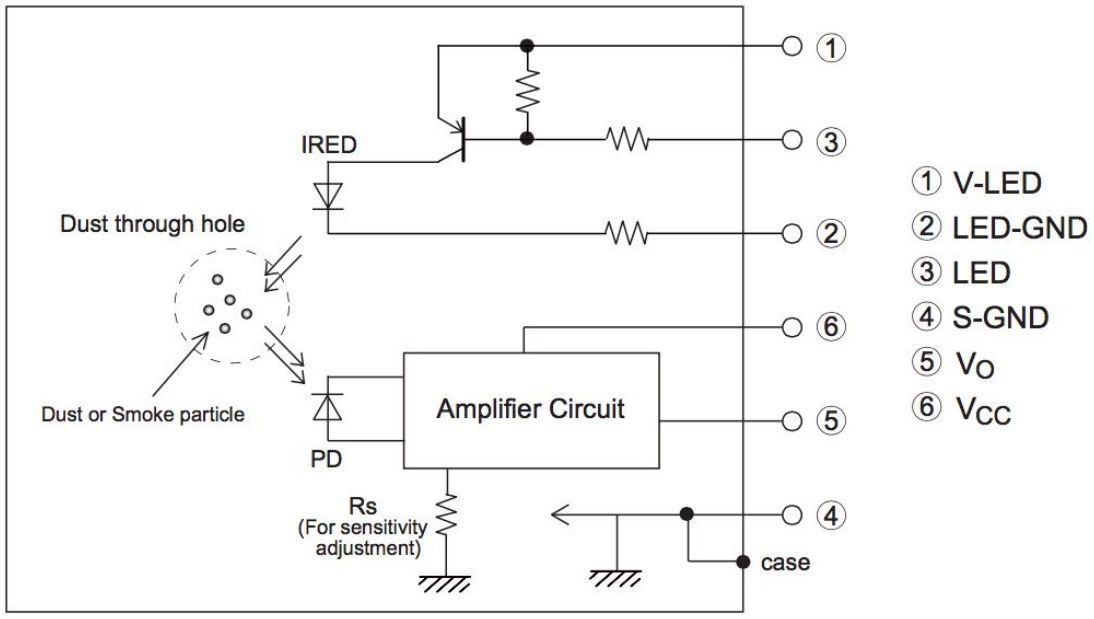

图 3-9 GP2Y1010AU 内部结构图

根据引脚说明图，需要在引脚 1 上串联 150 欧电阻和并联一个 220μF 电容，在发射端通过引脚 3 驱动传感器内部的红外发射管。引脚 5 即可输出模拟电压信号。

GP2Y1010AU 的引脚说明图如下:

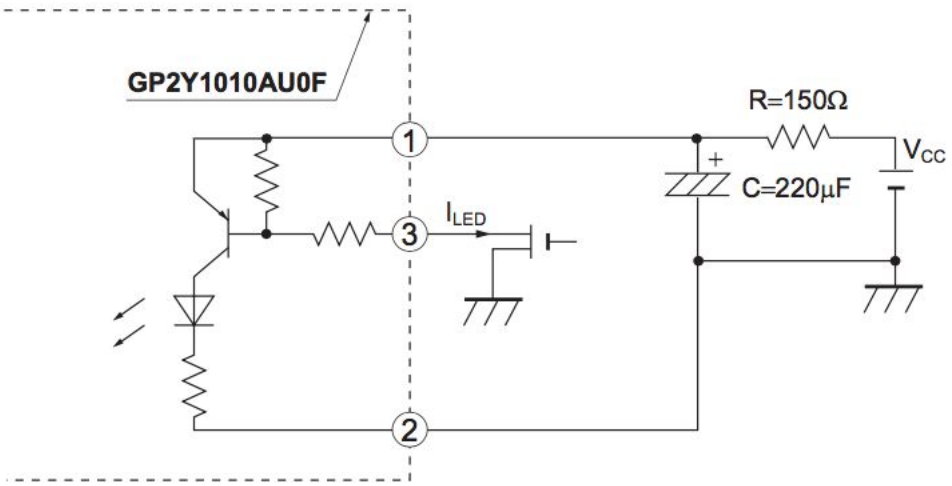

图 3-10 GP2Y1010AU 引脚说明图

3.3 模数转换模块

3.3.1 数字信号与模拟信号

我们周围的世界是模拟的。尽管我们可能听说这个世界正在“走向数字化”，但自然绝大部分能观察到的特性在本质上都是模拟的。这个世界可以承载无限多的可能状态，无论是阳光的颜色、海洋的温度还是空气中污染物的浓度。通过模数转换模块可以将这些无穷的可能性离散化为我们喜爱的数字值，这样它们就可以被树莓派这种微控制器系统所分析。如果想让设备与周围的世界交互，就不可避免要使用模拟数据。如果使用一个开关来控制 LED 的话，开关就是一种数字输入——它只有两种可能的状态:开或关、高或低、1 或 0，等等。数字化信息(计算机或者树莓派能够处理的信息)是一系列二进制(或数字化)数据。每一位都只能是的两个可能值中的一个。然而，我们身边的世界却很少只使用两种状态来表示信息。看一眼窗外，你看到了什么?如果现在是白天，那么你也许会看到阳光，看到树木在微风中摇摆，抑或是看到川流不息的车辆和人群。你看到的一切都不能轻易地用二进制数据表示。阳光不能以“开”或“关”表示，它的亮度在一天中不断发生变化。类似地，风也不是只有两种状态，其风力无时无刻不在改变。

计算机系统根本无法测量一个具有无穷多小数位的模拟量，这是因为计算机的存储和计算能力都是有限的。那么在这种情况下，怎样才能将树莓派与现实中的“真实世界”连接在一起呢？答案就是模数转换器(analog-to-digital converter，ADC)，它可以将模拟量

以有限的精度和速度表示为数字量。假设我们需要测定房间的亮度，一个良好的亮度传感器可以产生随房间亮度变化而变化的输出电压。全黑时，设备会输出 0V，而在日光下完全饱和时则输出 5V，在这之间的电压值的值则表示了各种不同的光照强度。看起来还不错，可是如何才能在树莓派上读取这些数值并计算出房间的亮度呢？这时可以使用模数转换器模块来将模拟电压值转换为数字表示，这样就可以在计算机上处理相应的数值了。

3.3.2 PCF8591 模块

PCF8591 是一个单片集成的低功耗模数转换模块，具有 4 个模拟输入和 1 个模拟输出，通过 I²C 总线串行接口与微处理器连接，通过 I²C 总线直接传输数据信号，以及相应的地址信号和控制信号。

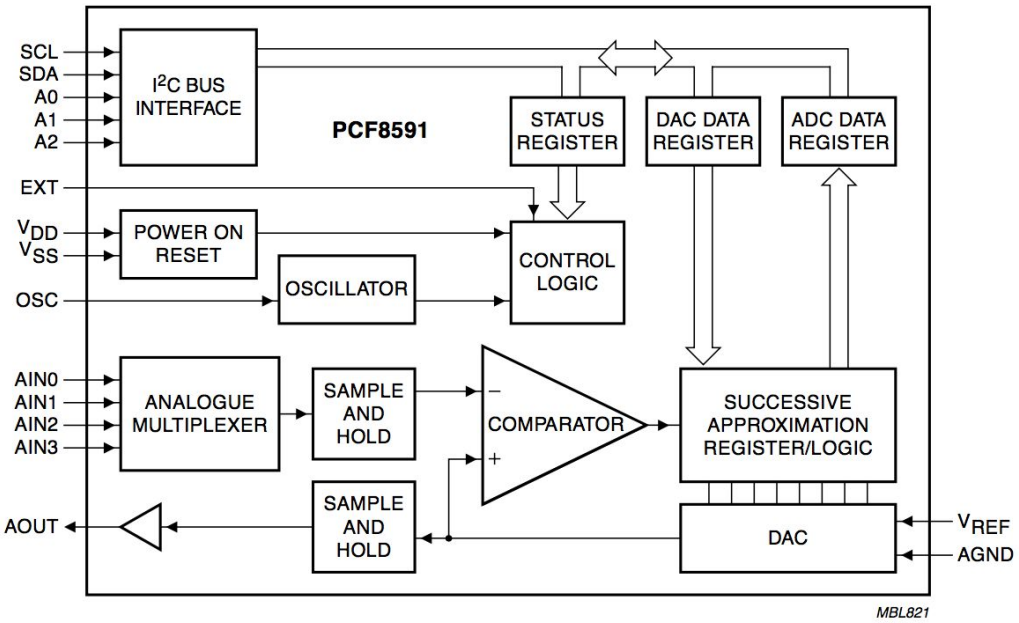

图 3-11 PCF8951 芯片原理图

该模块采用 PCF8951 芯片，支持外部 4 路电压输入采集(电压输入范围 0-5V)，模块集成光敏电阻和热敏电阻，可以直接通过 A/D 模块采集环境光强和温度精确数值，可以直接输入 0-5V 电压，通过蓝色电位器即可调节。与此同时，还具有电源指示灯和模数转换输出指示灯，当输出的数字电压达到一定值，就会点亮模数输出指示灯，指示灯亮度随着电压大小而变化。

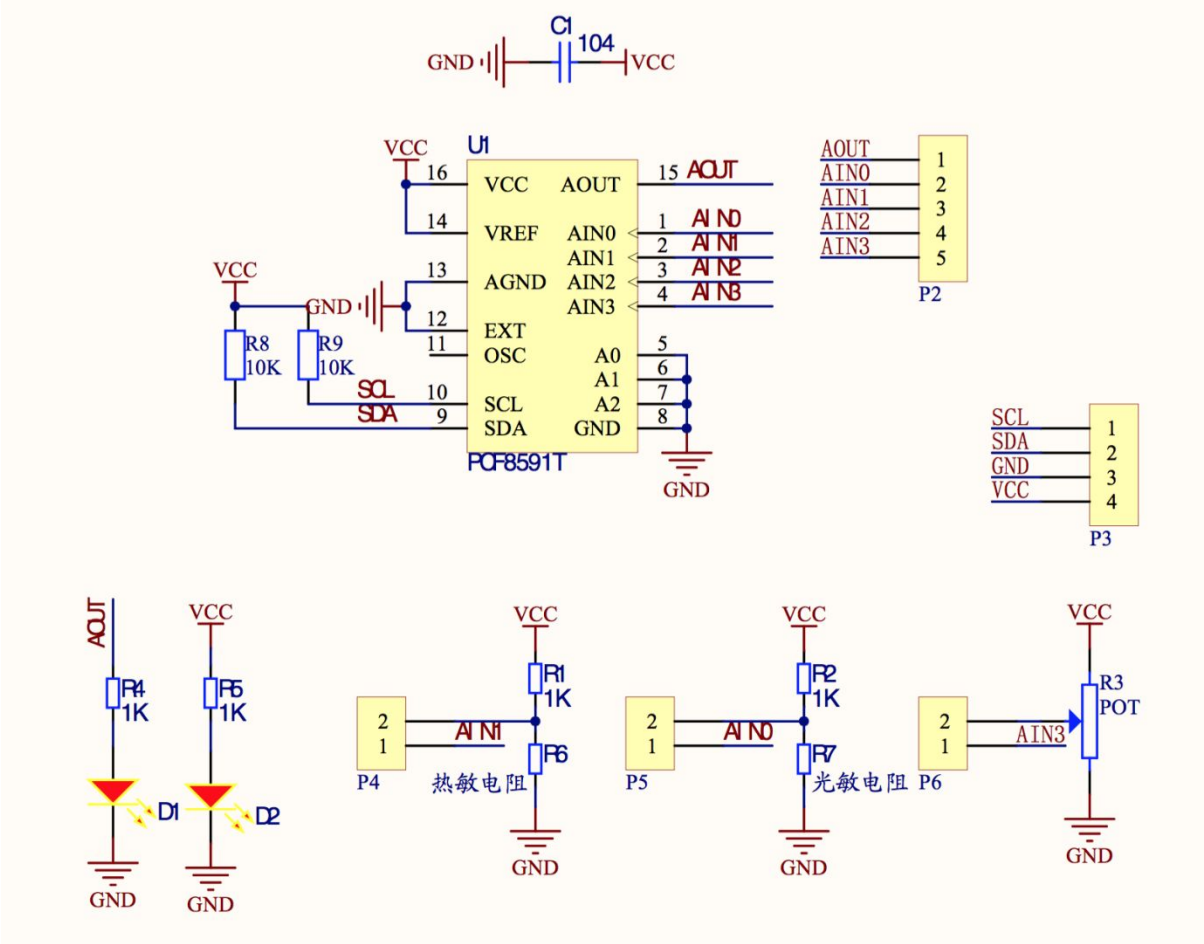

图 3-12 模数转换模块电路图

本模块左边和右边分别外扩 2 路排针接口，分别说明如下:

表格 3 - 2 PCF8591 模块引脚说明图

左边	右边
AOUT 芯片模数转换模块输出接口	SCL IIC 时钟接口
AINO 芯片模拟输入接口 0	SDA IIC 数字接口
AIN1 芯片模拟输入接口 1	GND 模块地 外接地
AIN2 芯片模拟输入接口 2	VCC 电源接口 外接 3.3V～5V
AIN3 芯片模拟输入接口 3	

3.3.3 Arduino 模块

Arduino 是一个微控制器开发平台，它配备了直观的编程语言，可以直接使用 Arduino 集成开发环境(IDE)来开发它。当然也可以为 Arduino 安装传感器、效应器、灯、扬声器、扩展板及其他集成电路，来将 Arduino 变成一个可编程的“大脑”，应用于几乎任何控制系统。Arduino Uno 是 Arduino 中的旗舰，在本次毕业设计中将使用这块板卡。它采用了一片 16U2 USB-串口转换器芯片，另有一片 ATMega 328p 作为主 MCU。如图所示，下面是一些需要关注的 Arduino Uno 关键器件:

- Atmel 微控制器；
- USB 编程/通信接口；
- 稳压器和电源连接；
- 引出的 I/O 引脚；
- 调试、供电和 RX/TX LED；
- 复位按键；
- 在线串行编程器(ICSP)连接器。

每片 Arduino 的核心都是一个 Atmel 微控制器单元(MCU)。绝大部分的 Arduino 板卡，包括 Arduino Uno，使用的是一片 AVR ATMega 328p 微控制器。Arduino 编程语言可以用于访问微控制器的外设，包括模数转换器 (ADC)、通用输入/输出(I/O)引脚、通信总线(包括 I^2C 和 SPI)及串口。所有这些有用的功能都从微控制器上那些微小的引脚引出到了 Arduino 上更容易使用的母头连接器上，可以将导线或者扩展板插入其中。一个 16MHz 的陶瓷谐振器连接到了 ATMega 的时钟引脚，作为所有程序指令执行的参考。当然，也可以使用复位按键来重启程序。绝大部分 Arduino 板卡带有一个已经连接到 13 号引脚的调试 LED，这使得能够在不连接额外的电路的情况下运行第一个程序，即让一个 LED 灯闪烁。

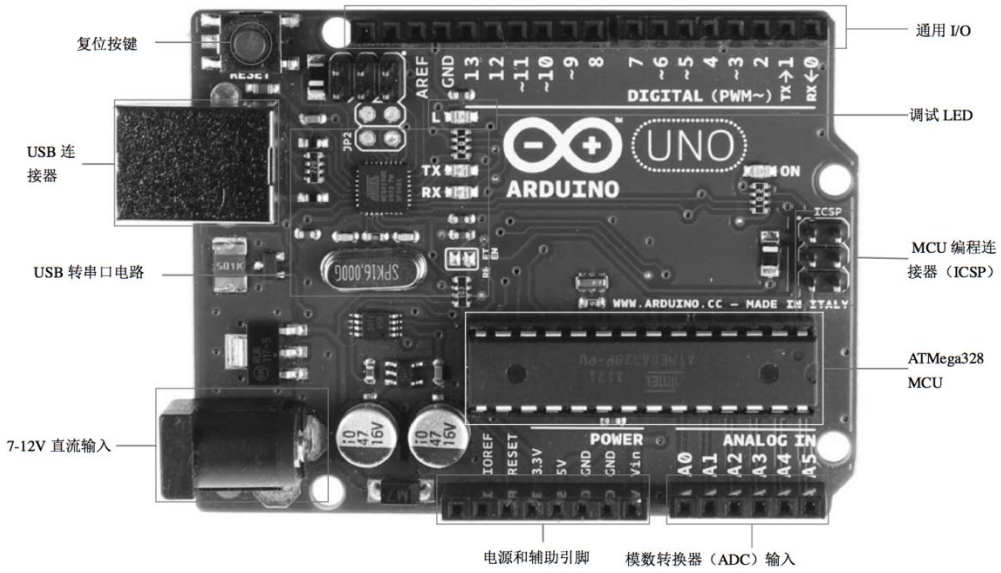

图 3-13 Arduino 硬件接口图

在本文中最关注 Arduino 上的部分就是通用 I/O 和模数转换器引脚。所有的这些引脚都可以通过编写的程序来被独立地寻址。它们都可以用作数字输入、输出。ADC 引脚还可以作为模拟输入端，用来测定 0～5V 的电压，由此就可以判定传感器的对空气质量的检测数值。当然在这些引脚中，有许多还可以被复用作其他功能，这些特殊的功能包括各种通信接口、串口、脉冲宽度调制输出及外部中断。

3.4 硬件的连接

3.4.1 树莓派 GPIO

树莓派和普通电脑不一样的地方在于它还带了 17 个可编程的 GPIO（General Purpose Input/Output 即通用输入输出接口）连接器，可以用来驱动如传感器，步进电机等其他的硬件设备。既然一个引脚可以用于输入、输出或其他特殊功能，那么一定有寄存器用来选择这些功能。对于输入，一定可以通过读取某个寄存器来确定引脚电平的高低；对于输出，一定可以通过写入某个寄存器来让这个引脚输出高电平或者低电平；对于其他特殊功能，则有另外的寄存器来控制它们。GPIO 连接器实际上是由许多不同类型的接口组成的：

- 真正的 GPIO（General Purpose Input Output，通用输入/输出）针脚，可以用来控制 LED 灯的开和关。

- I^2C（Inter－Integrated Circuit）接口针脚，能够仅使用 2 个控制针脚连接硬件模块。
- SPI（Serial Peripheral Interface，串行外设接口）接口用以连接 SPI 设备，概念上与 I^2C 接口类似，但是标准不同。
- Serial Rx 和 Tx 针脚用于和其他的串行外围设备通信。

另外，一些针脚可被用于 PWM（pulse Width Modulation， 脉冲宽度调制）进行电源控制，以及 PPM（Pulse Position Modulation，脉冲位置调制）控制马达。

GPIO#	2nd func	pin#		pin#	2nd func	GPIO#
N/A	+3V3	1		2	+5V	N/A
GPIO2	SDA1 (I2C)	3		4	+5V	N/A
GPIO3	SCL1 (I2C)	5		6	GND	N/A
GPIO4	GCLK	7		8	TXD0 (UART)	GPIO14
N/A	GND	9		10	RXD0 (UART)	GPIO15
GPIO17	GEN0	11		12	GEN1	GPIO18
GPIO27	GEN2	13		14	GND	N/A
GPIO22	GEN3	15		16	GEN4	GPIO23
N/A	+3V3	17		18	GEN5	GPIO24
GPIO10	MOSI (SPI)	19		20	GND	N/A
GPIO9	MISO (SPI)	21		22	GEN6	GPIO25
GPIO11	SCLK (SPI)	23		24	CE0_N (SPI)	GPIO8
N/A	GND	25		26	CE1_N (SPI)	GPIO7

图 3-14 树莓派 GPIO 引脚图

除了供电针脚（包括 GND，3.3V 和 5V），所有的 GPIO 针脚均可被用于数字输入或输出。作为数字输出，可以通过程序直接切换某些针脚的高低电平，如果置为 3.3V 就是高电平，0V 就是低电平。如果使用针脚作为数字输入，则可以把开关和简单的传感器连到一个针脚上然后检查它是否打开或关闭，也就是是否激活。当然，如果是传感器等模拟信号，则需要通过 ADC（即模数转换器）进行转化。标记为 SCL 和 SDA 的针脚可被用于 I^2C。标记为 MOSI，MISO 和 SCKL 的针脚可以用于连接高速 SPI 设备。需要注意的是，当从树莓派的 GPIO 接口上读取数据时，实际上是在检查这个 and 接口是被接在 3.3V 电

源(高电平)上还是被接地(低电平)了。必须牢记，输入接口必须连接一个确定的信号(3.3V 或接地)，如果尝试读取一个即未接电源又未接地的引脚，将会得到一个不稳定的信号。所有针脚有一个 3.3V 的逻辑电平但是并不是 5V 安全的，所以输出电平是 O-3.3V 并且输入电平也不应高于 3.3V。如果想要连接 5V 的输出电平作为树莓派的输入电平，这是就需要使用一个电平位移器。

3.4.2 I^2C 总线

I^2C(Inter－Integrated Circuit)一种广泛采用的两线式串行总线标准，用于连接微控制器及其外围设备，主要由双向串行时钟线 SCL 和双向串行数据线 SDA 两条线路组成。每个器件通过唯一的地址作为标示，器件的功能决定了是作为发送器还是接收器。器件在数据传输的时候也可以被看作是主机或从机，主机在数据传输之前进行初始化，并能够产生允许传输的时钟信号。此时，任何被寻址的器件都被认为是从机。

在本毕业设计论文中，将通过 I^2C 总线连接树莓派和 PCF8591 模数转换模块，PCF8591 具有 1 个串行 I^2C 总线接口，SCL 口接树莓派 P1 的 5 脚，对应树莓派的 I2C1_SCL 口；SDA 口接树莓派 P1 的 3 脚，对应树莓派的 I2C1_SDA 口。

3.4.3 UART 串口

通信双方约定好数据将按照逐位按顺序的约定进行传送，这就叫做串行通讯。串行接口可以将传输给 CPU 的并行数据字符和串行数据流进行相互转换。在本项目中，当树莓派通过串行端口发送数据时，字节数据要变成成串行位；接收数据时，串行位则转换成字节数据，所以必须给树莓派安装相应的驱动程序。

Arduino 与树莓派通过串口通信的方式实现通信，相互传输所需要的数据，树莓派将资源传于互联网上对应的接口，接口可以在互联网上被访问。Arduino 与 Raspberry 通过串口通信的方案一般有两种，通过树莓派 GPIO 串口直接通信，当然还有通过 Arduino 的 USB 转串口进行通信。树莓派的 GPIO 外部接口中含有一路 UART 串行接口，利用该接口可以实现树莓派与 Arduino、GPRS 模块、GPS 等其他外部系统的对接。但是由于树莓派与 Arduino 的串口电平不一致，即树莓派输入输出电压为 3.3v，而 Arduino IO 口电压为 5V，所以必须使用电平转换芯片，或串联电阻限流，否则有可能损坏树莓派 IO 口。显然，通过 USB 通信会比 GPIO 通信高效稳定得多，同时使用起来也比较方便。由于笔者

将使用 Arduino 自带的 USB 转串口，因此必须给树莓派安装其驱动，而驱动就在 arduino 提供的软件上，所以必须在树莓派上直接安装 Arduino 软件即集成开发环境。

3.4.4 硬件电路结构

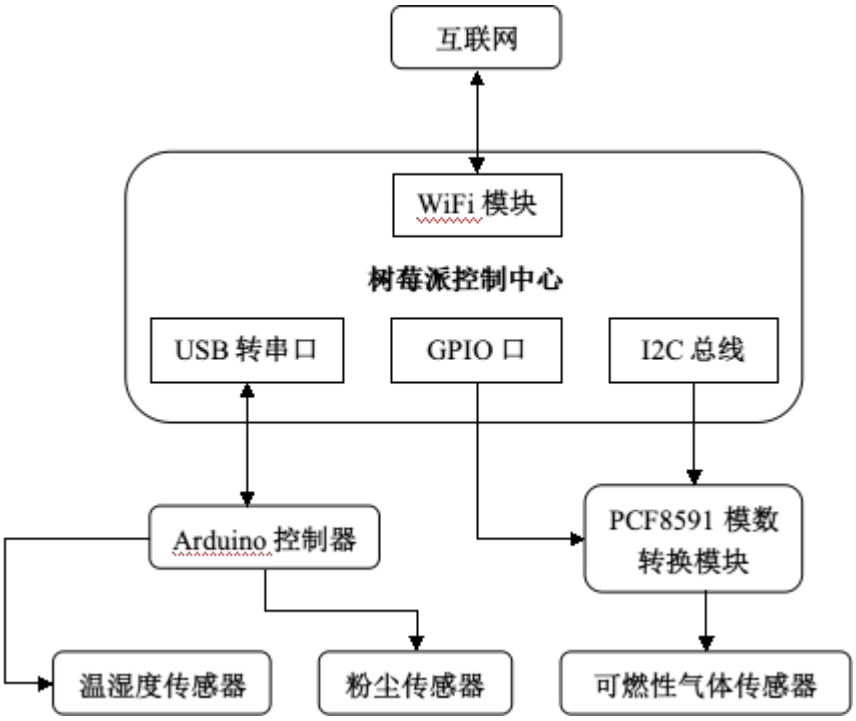

图 3-15 整体硬件电路结构图

第四章 监控系统的本地软件设计

4.1 操作系统的安装与使用

4.1.1 Linux 内核简介

Linux 得名于电脑业余爱好者 Linus Torvalds。Linux 就是一种类 UNIX 操作系统内核，具有多用户、多任务、支持多线程和多 CPU 等特点。它支持 32 位和 64 位硬件，继承了 Unix 以网络为核心的设计思想。作为领先的操作系统内核，它还是开源的，我们可以直接使用和修改 Linux 的所有底层源代码。通常情况下，开发人员可以针对不同的应用场景推出不同的发行版本如 Debian，最典型的就是它的衍生版本 Ubuntu，以及树莓派所使用的 Raspbian 嵌入式版本，还有比较著名的就是 Fedora 和 RedHat 等。

4.1.2 安装 Raspbian 系统

树莓派相比于一般的 ARM 开发板来说，由于其本身搭载着 Linux 操作系统，可以用诸如 Python、Ruby 或 Bash 来执行脚本，而不是通过编译程序来运行，因此具有更高的开发效率。

笔者为树莓派选用的是官方提供的 Raspbian 即官方推荐使用的 Raspberry Pi 专用 Debian 版本。到官方网站打开下载页后，选择 Raspbian “wheezy”下载最新版。将 zip 文件解压缩后会得到一个.img 镜像文件，将其写入 SD 卡即可作为树莓派的系统。首先下载安装 SD Formatter 4.0 for SD/SDHC/SDXC 和 win32DiskImager 这两款软件，前者用来格式化 SD 卡，后者加载镜像为 SD 卡烧录系统。需要注意的是一定要选择 SD 卡所在的盘符，避免不恰当的操作导致笔记本硬盘数据丢失。写入完毕之后，为了使用方便还需要修改 SD 卡的配置文件内容，首先要修改 SD 卡是 cmdline.txt 文件，在最开始加入语句 ip=192.168.31.183，就可以将树莓派设置成静态 IP 地址，到时候就可以很方便得通过这个地址连接到树莓派。

至此，树莓派的系统已经安装到 SD 卡，将其插入树莓派的 SD 卡插槽。但是在连接到树莓派需要说明的一点是，树莓派的供电电源需要使用 2A 的电源适配器，普通的智能手机充电器只有 1A，可能会导致电源供电出现问题。另外一个问题是，在连接外接显示

器的时候，由于树莓派只有 HDMI 而没有 VGA 接口，此时需要准备一个 HDMI-VGA 转接线，而且必须是有源的 HDMI 转接线，即需要有外接电源，否则有可能烧坏二极管。当然，可能是由于笔者使用的是 5V、2A 的电源，直接使用无源的 HDMI 转接线也是没有问题的。连接好鼠标键盘之后，启动电源，便可以进入到树莓派的开机界面。

开机之后需要对树莓派的进行第一次配置，树莓派内置一个名为“Raspi-Config”的配置工具，它会在第一次启动树莓派时自动运行。它会在桌面系统运行前启动，所以需要使用方向键和回车键来定位菜单系统。这有点像调整 PC 的 BIOS 设置。一旦配置正确，可能以后都不再需要配置第二次了。首先树莓派默认只使用操作系统需要的 SD 卡空间。这意味着即使有一个很大容量的 SD 卡，操作系统也不会使用那么多容量。由于笔者需要在 Raspbian 系统上安装软件和运行程序，则必须将 SD 卡的所有空间都得以使用，使用上/下方向键选择“expand_roofts”菜单项然后敲一下回车。之后可以通过修改“overscan”来设置全屏、修改“change_timezone”设置时区、修改“boot_behaviour”设置树莓派自动登录并且启动到桌面环境，另外需要修改的是“ssh”，这个选项将打开 ssh 使外部网络能够对树莓派进行远程操作，即从另一台计算机上控制树莓派。

设置完毕，点击“Finish”便可以进入到树莓派的桌面环境了，如果在之后的时间里需要改变某些配置，可以点击桌面上的“LXTerminal”图标打开一个新终端，在任何时候运行 sudo raspi-config 命令即可进入之前的配置页面。

4.1.3 网络配置流程

由于树莓派的性能不足以流畅运行桌面环境，而且鼠标、键盘和显示器等外接设备也占用了本来就少的硬件接口，所以本节将介绍如何配置树莓派接入网络，从而通过远程操作的方式来控制树莓派。

如果已经成功运行且进入了树莓派的桌面环境，则可以插入 USB 无线网卡，笔者使用的是 EDUP 厂商的 EP-N8508GS 型号。最新的 Raspbian 和 Occidentalis 已经原生支持这个芯片，无须安装驱动即可使用，并且它的速度也比较快。这时会有一个问题就是把无线网卡插到树莓派之后，那就只剩一个 USB 口可以用来接键盘或鼠标了，因此需要额外的 USB 扩展，但必须是一个有源的 USB Hub 才可以。由于笔者没有买有源 USB Hub，直接是采用了鼠标、键盘轮流插拔的方式来操作树莓派。

首先介绍如何通过图形界面进行无线网的配置，Raspbian 系统都带有一个 WiFi 配置工具，可以直接在桌面上找到一个“WiFi Config”的快捷方式，具体配置详情就不再赘述。当然，也可以使用命令行工具通过以太网登录，手动编辑/etc/network/interfaces 即可:sudo nano /etc/network/interfaces。

在将树莓派接入网络之后，由于笔者在安装完毕的 SD 卡 cmdline.txt 文件中已经设置了树莓派的静态 IP，所以直接使用该 IP 地址访问到树莓派。但与此同时，如果没有完成之前的设置可以通过以下几种方式找到树莓派的 IP 地址。第一种是直接使用命令行来查找，打开一个 LXTerminal 终端窗口并且键入下面的命令:sudo ifconfig，便可以在 wlan0 旁边看到树莓派的 IP 地址 192.168.31.183。需要注意的是，如果树莓派的 IP 地址以 192.168 或者 10.0 开头，那么这是一个内网地址，它意味着树莓派可以访问因特网但是内部网络以外的其他用户是无法通过这个 IP 直接访问到树莓派的。所以在接下的工作中，需要把树莓派作为一个能从外网访问的网页服务器。而把树莓派放在内网中再通过内网穿透的方式接入外网，那它就会更加安全一些而不会被轻易黑掉。

但是，如果没能进入树莓派的图形界面该如何走到树莓派的 IP 地址并且操作树莓派呢，首先比较简单的方式是将树莓派通过以太网接口使用网线连接至路由器，在路由器里可以为树莓派所对应的 Mac 地址保留一个固定的 IP 地址。由于开发人员对树莓派所接入的网络的路由器有完全的控制权，通过 DHCP(动态主机设置协议)为所绑定的设备分配静态 IP。当然，如果是通过无线网卡的方式连接到路由器的话也是一样的，可以直接在路由器后台为树莓派绑定一个静态 IP，以便于之后的操作。

通过一根网线也可以直接将树莓派和笔记本电脑进行连接，连接之后可以通过扫描软件 PortScan 扫描到树莓派的 IP 地址。如果笔记本已经通过 WIFI 连接到互联网，另一种方式就是将无线网卡的互联网资源共享给本地连接。然后运行 DOS 窗口，输入 arp -a，在接口下显示为动态类型的 IP 地址就是树莓派的地址。

4.1.4 远程操控树莓派

上一节中笔者已经将树莓派接入互联网且已知树莓派的动态 IP 地址或者设置好的静态 IP，下面将介绍两种无显示器远程操控树莓派的方式:SSH 和 VNC。前者是命令行，后者是图形界面。如果熟悉命令行下的 Linux 系统的话，SSH 已经足够使用了。SSH 指的

就是 Secure Shell，这是 Linux 系统的一个功能可以直接从主机电脑上快速打开一个树莓派的终端会话。如果使用的是 Mac 或者 Linux 计算机的话，可以直接在终端窗口输入下面的命令：ssh 192.168.31.183 -l pi，参数“-l pi”的意思是将使用“pi”作为用户名登录到树莓派上。第一次运行这个命令，将会得到一个安全警告关于不能确认该机器的身份，输入密码（默认是“raspberry”）进行树莓派的命令行操作页面。在 Windows 电脑中，需要下载一个叫做“putty”的免费软件，通过 putty 软件输入树莓派 ip，默认端口 22，点击 open 就会连接到树莓派，同样需要输入用户名和密码。为了使用更方便，可以通过命令启用 root 用户，命令为：sudo passwd root，这是设置 root 账号的密码，需要输入两次新密码，然后再启用 root 用户登录，命令为：sudo passwd –unlock root。执行完之后，使用 reboot 命令重启就可以使用 root 用户登录，不需要每次打开终端都要求输入密码。

然后我们需要通过 SSH 为树莓派安装 VNC 软件，VNC（Virtual Network Connection）是一种可视化网络连接的标准，通过 VNC 可以就可以远程访问和操作树莓派的图形界面。打开先前已经建立好的 SSH 连接就可以为树莓派安装 VNC server，并且设置好开机自启动，还调整 VNC 屏幕的尺寸。与此同时，需要在 Windows 下也安装一个 VNC 客户端，笔者使用的是“VNC Viewer”，输入树莓派的 IP 地址:1（意味着端口号）点击连接，输入树莓派上的 VNC server 的密码就会弹出 VNC 窗口。

有时候需要在树莓派和笔记本电脑直接互传文件，可以使用 SFTP 远程传输树莓派文件，下载安装 FileZilla，在“快速连接”中输入：主机 sftp://192.168.31.183 和用户名、密码，就能建立连接，无需设置直接就是 UTF-8 编码，中文名文件上传到树莓派就不会发生乱码的问题。当然，也可以使用 Sublime Text 编辑器的 SFTP 插件，直接在编写代码的同时传输代码文件，非常方便。

4.2 控制中心主程序设计

4.2.1 整体程序框图

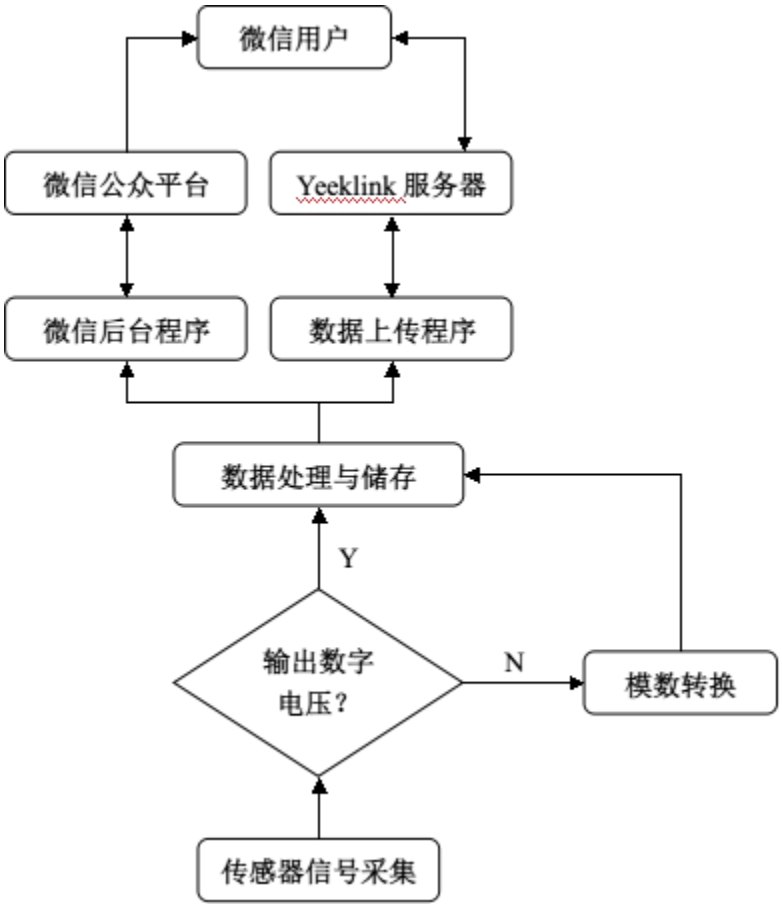

图 4-1 系统总体工作流程图

4.2.2 程序控制流程

从上一节中的总体工作流程图中可以看出，传感器采集数据之后会得到相应的电压数值，不同的传感器有不同的电压信号，如果是数字电压，例如数字温湿度传感器 DHT11，则可以将 DHT11 的数字量输出接口直接连接至树莓派的 GPIO 口，树莓派可以通过相应的代码库 Rpi.GPIO 读取到数字化处理后的数值。如果是模拟电压，则有两种方式获得传感器的有效数字量，第一种方式是将传感器连接到 PCF8591 模数转换模块的模拟输入口，经 PCF8591 模块进行模数转换之后，再通过 I^2C 总线将传感器值传输给树莓派。第二种方式是使用 Arduino 作为数据收集模块，Arduino 既能读取数字信号，又能读取模拟信号，且由于更为底层的原因，效果往往比树莓派直接读取要好得多。例如 MQ

系列的气体物质传感器就是将模拟输出口连接至 Arduino 的模拟输入口，然后 Arduino 再将读取到的模拟量经数字化处理后通过串口的方式发送给树莓派。

树莓派得到相应的数字化处理的模拟电压量之后，将其转化为相应的电压伏值，根据不同的传感器参数计算出传感器电压值所代表的空气质量情况，从而推算出当前室内空气质量指数。与此同时，在完成一次传感器数据的收集之后，需要将有效数据存储到相应的记录文件中，以便于后续的网络应用能够直接读取到有效的数据，避免程序出错。

4.3 传感器数据读取

4.3.1 数据采集程序

针对于不同的传感器，有不同的方式进行数据的采集，下面将对温湿度传感器 DHT11、MQ 系列气体传感器、GP2Y1010AU 粉尘传感器以及树莓派自带的工作状况分别进行介绍:

首先对于温湿度传感器 DHT11 来说，输出的是数字信号，可以直接通过 Python 模块 RPi.GPIO 直接读取树莓派的 GPIO 口从而得到数据。与此同时也可以使用 Python 代码调用子进程直接运行 C 语言模块，然后从 shell 中获取相应的输出结果，解析出所需要的温湿度数据。还有一种方法直接使用 setup.py 将 dhtreader.c 编译成一个 dhtreader.so 库，从而可以在 Python 代码中直接调用这个模块获得温湿度数据，这有点类似于 LAB View 中的 C 节点。

当然，相较于树莓派来说，Arduino 更适合作为传感器数据的采集和处理，根据硬件时序图可以使用 C 语言编写相应的采集代码，直接在 Arduino 中运行。程序如下:

```
#include "dht11.h"

// 此函数读取温度值，并且返回对应的读取状态:
// DHTLIB_OK
// DHTLIB_ERROR_CHECKSUM
// DHTLIB_ERROR_TIMEOUT
int dht11::read(int pin)
{
  uint8_t bits[5]; //存储读取的数据
  uint8_t cnt = 7; //计数器
```

```
uint8_t idx = 0;

//初始化数组为0
for (int i=0; i< 5; i++) bits[i] = 0;

//发送开始信号
pinMode(pin, OUTPUT);//设置为输出
digitalWrite(pin, LOW);//保持低电平至少 18ms
delay(18);
digitalWrite(pin, HIGH);//保持高电平 20-40us
delayMicroseconds(40);
//开始接受 dht 响应信号
pinMode(pin, INPUT);//设置为输入

unsigned int loopCnt = 10000;
while(digitalRead(pin) == LOW)
  if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

loopCnt = 10000;
while(digitalRead(pin) == HIGH)
  if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

// READ OUTPUT - 40 BITS => 5 BYTES or TIMEOUT
//开始等待温度传感器回应(根据数据手册温度传感器回应时会拉低数据线)
for (int i=0; i<40; i++)
{
  //如果此时的电平和当前电平相等说明电平没有发生变化，所以继续等下一次循环
  loopCnt = 10000;
  while(digitalRead(pin) == LOW)
    if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

  unsigned long t = micros();
```

```
    loopCnt = 10000;
    while(digitalRead(pin) == HIGH)
      if (loopCnt-- == 0) return DHTLIB_ERROR_TIMEOUT;

    if ((micros() - t) > 40) bits[idx] |= (1 << cnt);//左移一位并且将数据 1 写入(此时最后一位为 0)
    if (cnt == 0)   // 是否为下一个字节
    {
      cnt = 7;    // restart at MSB
      idx++;      // 下一个字节
    }
    else cnt--;
  }
  /*
  开始处理数据

  1.因为一次完整的数据输出为 40 位，所以 j 应大于 40
  2.dht11_val[4]为校验和，此处判断校验和是否正确
  3.只打印 dht11_val[0], dht11_val[2]是因为 dht11_val[1], dht11_val[3]是小数部分，根据数据手册小数始终为零
  */
  // 写入正确的数据
  // 由于 bits[1]和 bits[3]总是 0，所以就省略格式了
  humidity    = bits[0];
  temperature = bits[2];

  uint8_t sum = bits[0] + bits[2];
  //校验正确，返回正确的状态码
  if (bits[4] != sum)
      return DHTLIB_ERROR_CHECKSUM;
  return DHTLIB_OK;
}
```

对于 MQ 系列气体传感器和 GP2Y1010AU 粉尘传感器来说，是直接将模拟信号发送

到 Arduino 的模拟输入口。根据 GP2Y1010AU 粉尘传感器 DataSheet 中的脉冲周期图可知，需要在读取传感器输出值之前，开启内部的 Led 并等待 280μs（微秒）。由于整个脉冲持续时间为 320μs，因此还需要等待 40μs 才能将 Led 关闭。程序如下：

```
int measurePin = 0; // 连接模拟口 0
int ledPower = 2;   // 连接数字口 2

int samplingTime = 280;
int deltaTime = 40;
int sleepTime = 9680;
float voMeasured = 0;
float calcVoltage = 0;
float dustDensity = 0;

void setup(){
  Serial.begin(9600); //串口波特率设置成 9600
  pinMode(ledPower,OUTPUT); //设置数字口为输出
  delay(1000); //给串口一定的准备时间
}

void loop(){
  digitalWrite(ledPower,LOW);           //开启内部 LED
  delayMicroseconds(samplingTime);      // 开启 LED 后的 280us 的等待时间
  voMeasured = analogRead(measurePin);  // 读取模拟值
  delayMicroseconds(deltaTime);         // 40us 等待时间
  digitalWrite(ledPower,HIGH);          // 关闭 LED
  delayMicroseconds(sleepTime);

  int chk = DHT11.read(DHT11PIN);
  if (chk == DHTLIB_OK){
    Serial.print(voMeasured);//输出采集到的粉尘传感器电压
    Serial.print("-"); //设置分隔符，便于树莓派处理数据
    Serial.print(analogRead(1)); //输出 MQ-2 气体传感器的信号
```

```
    Serial.print("-");
    Serial.println(analogRead(2)); //输出 MQ-2 气体传感器的信号
  }
  delay(3000);
  Serial.flush();
}
```

与此同时，树莓派内置了一个传感器可以用来直接获取树莓派的 CPU 和 GPU 温度。这对于保护监控设备非常有用，当树莓派温度过高的时候可以设置关掉空气监控系统或者在温度过热的时候进行报警提醒。监控树莓派的 CPU 和 GPU 温度的 python 脚本如下:

```
import commands
import os

def get_CPU_temp():
    CPU_temp_file = open( "/sys/class/thermal/thermal_zone0/temp" )
    CPU_temp = CPU_temp_file.read()
    CPU_temp_file.close()
    return float(CPU_temp)/1000

def get_GPU_temp():
    GPU_temp = commands.getoutput( '/opt/vc/bin/vcgencmd measure_temp' ).replace( 'temp=',
'' ).replace( '\'C', '' )
    return float(GPU_temp)
```

4.3.2 模数转换程序

在本节中将介绍如何通过 Python 封装的 smbus 模块读取 I^2C 总线所传输的数据，这个 Python 模块可以使得 SMBus 能够直接访问 Linux 主机的 I^2C/dev 设备接口。SMBus 是基于 C-扩展 cffi 基础重新实现的 Python 版本，绑定到 SMBus 库的模块现在还是相当缓慢的加载模块。特别是对像树莓派这样的慢速设备，可能需要一点时间来加载 SMBus 模块。树莓派的 Raspbian 系统支持 I^2C 和 I^2C 设备接口，以及总线适配器驱动程序。但是在树莓派中，I^2C 被系统默认禁止，所以首先要解除 I^2C 设备的限制。修改系统中的 /etc/modprobe.d/raspi-blacklist.conf 文件，将 I^2C 和 SPI 驱动从黑名单中注释掉即可。与此同时打开/etc/modules 配置文件进行修改，添加 i2c-bcm2708 和 i2c-dev 设备，重启树莓派

就可以永久载入该设备。

然后通过 apt-get 的方式安装 Python 依赖库即 python-smbus，以及 i2c-tools 工具，有了 i2c-tools 就可以直接使用 i2cdetect 命令：sudo i2cdetect -y -a 查看总线上所挂载的所有设备。如图所示：UU 为系统占用地址；48 为 PCF8591 的地址，十六进制的，转换为二进制就是 1001 000（根据硬件电路不同，末三位可能为 000~111）。记下本步查询的结果 0x48，如果查询结果只有 UU，可能是硬件电路没有连接好。

```
#Read a value from analogue input channel
#in A/D in the PCF8591P @ address 0x48
from smbus import SMBus
import time
bus = SMBus(1)

def read_channel(sensor, channel):
   bus.write_byte(0x48, channel) # 设置寄存器读取通道
   reading = bus.read_byte(0x48) # 读取该通道的模拟量
   print "%s sensor | channel %s: %s"%(sensor, channel, reading)

while(1): # do forever
   read_channel("resistance", 0);
   read_channel("light", 1);
   read_channel("temprature", 2);
   read_channel("external", 3);
   print "*"*40
   time.sleep(2)
```

4.3.3 串口读取程序

作为一款优秀的开发平台，树莓派的确能够承担诸多外围设备，比如电机，LED 等驱动功能，但是与此同时，对项目进行功能模块化也是非常重要的。毕竟树莓派的 GPIO 数目是一定的，扩展能力有限。所以在本次系统设计中也将介绍 Arduino 与树莓派如何进行通信，这样就可以有效地增加设备的输入输出的数量，而且能够减轻 Raspberry Pi 的负荷。

关于树莓派与 Arduino Uno 如何对接，实现方式至少有三种：USB 方式对接、GPIO

方式对接、I²C 方式对接。本文将主要介绍如何使用 USB-串口方式进行连接。首先在树莓派上安装 Python 的 serial 库，用于串口通信及 USB 通信，还可以安装串口调试工具 minicom，可以非常方便得修改串口波特率等等。将 Arduion 通过 USB 转串口的线连接至树莓派的 USB 口，通过 ls /dev/tty*命令可以查找到挂载到树莓派上的 USB 设备端口，笔者使用的是/dev/ttyUSB0 端口，这与网络上普遍所说的 ttyACM0 端口有所出入。

```
import serial

port = '/dev/ttyUSB0' #设置串口的端口
arduino = serial.Serial(port,9600,timeout=1) #设置串口波特率
time.sleep(1.5) #给串口一定的准备时间
while (True):
   arduino.write("Everything is OK!")
   print ("Message from arduino: ")

   msg = arduino.readline().strip('\n\r') #读取 arduino 发送的数据
   print msg

   try:
      key = ['pm2.5', 'MQ-2', 'MQ-5', "hum", "temp", "PM2_5"]
      value = msg.split('-') #解析数据格式
      data = dict(zip(key,value)) #构造字典数据结构:{键:值}
      print data
   except:
      print 'ERROR!'

   #将传感器数据写入记录文件
   with open('../data/sensor.log', 'a') as f:
      if (len(data) == 6): #判断数据长度是否正确
         f.write(str(data) + '\n') #按行写入数据
         print "write success!"
      else:
         print "[ERROR:] write error!"
```

```
    print "*"*40
    time.sleep(3) #每 3 秒执行一次脚本
else:
    print "Exiting"
```

需要注意的是由于 Arduino 是单线程的，所以 Arduino 的串口通信是单信道。而且这里有 2 个时间延迟，第一个是做串口初始化等待，第二个是等待数据传输。简单地在树莓派和 Arduino 之间直接发送与接收来传递信息，存在信息丢包的现象。

4.4 空气质量数据处理

4.4.1 粉尘密度算法

GP2Y1010AU 粉尘浓度传感器对于相当于香烟的烟雾检出物，可以按如下公式表示：

可以检出的范围（输出电压可变范围 (V) ）= 输出电压范围：VoH(V) — 无尘时输出电压：Voc(V)

将此换算成粉尘浓度：

检出粉尘浓度范围(mg/m3) = 检出可能范围（输出电压可变范围(V)）÷ 检出感度：K(V/(0.1 mg/m3)

烟检出的情况下，其判定值如下：

判定值 = 检出浓度(mg/m3) ÷ 10 × K（V/(0.1 mg/m3)）+ 无尘时输出电压(V)

举个例子，当检出浓度为 0.2(mg/m3) 来判定时，已知检出感度 K 为 TYP 0.5 (V/(0.1 mg/m3)，无尘时输出电压为 0.9 (V) 的情况下，判定值 = (0.2×10)×0.5+0.9 = 1.9 (V)

灰尘的检出则是在规定时间内，在某一输出电压变化的标准以上，判定在某一时间的输出被记入什么，从而检出灰尘的有无。

数据处理和计算程序的代码如下：

```
def cal_vol(voMeasured):
    voMeasured = float(voMeasured)
    # 0 - 5V mapped to 0 - 1023 integer values
    # recover voltage
    cal_voltage = voMeasured * (5.0 / 1024.0)  #将模拟值转换为电压值
    return cal_voltage
```

```
def cal_den(cal_voltage):
    cal_voltage = float(cal_voltage)
    # linear eqaution taken from http:#www.howmuchsnow.com/arduino/airquality/
    # Chris Nafis (c) 2012
    dust_density = 0.17 * cal_voltage - 0.1      #将电压值转换为粉尘密度输出单位
    return float("%.6f"%(dust_density))  # 输出单位: 毫克/立方米

def cal_air_index(cal_voltage):
    #根据电压值计算颗粒数量
    air_index = (float(cal_voltage*5.0)-0.0356)*120000*0.035
    return air_index
```

4.4.2 空气质量评定

结合国外的一个对夏普 GP2Y1010AU0F 的烟雾实验，根据 DataSheet 图中所提供的颗粒浓度(毫克/平方米)与电压相对值的比较图，建立起输出电压与颗粒数量的对应关系:(V-0.0356)*120000

表格 4 - 1 粉尘浓度的评定

小颗粒读数(大于 0.5 微米)	评定指数
3000+	很差很差
1050-3000	差
300-1050	还行
150-300	好
75-150	非常好
0-75	棒极了

借鉴于此思路，笔者也将 MQ 系列的气体物质传感器和室内温湿度结合起来，共同判断室内空气的整体状况。为了达到检测的空气质量评定的综合性以及减少单一传感器的偶然性误差，在数据处理时需要进行多传感器的数据融合，共同判定空气质量指数。以国家室内空气质量标准 GB／TB8832002 作为参考对数据融合后的数据进行模糊化处理。为每个传感器参数在其取值范围上定义 3 个概念(低、一般、高)，选用梯形隶属度函数。PM2.5 粉尘和甲醛作为人体健康主要的影响因素，需要将其语言值进行综合评定求出室内

空气质量指数，相应的空气质量指数规则如表所示。

表格 4 -2 空气质量指数评定

影响因素	参数范围	影响权重
温度	~21、22~28、29~	2
湿度	~35、36~70、71~	1
PM2.5	~300、301~1050、1050~	3
烟雾	~50、51~120、121~	5
液化气	~30、31~100、101~	4

通过语义化处理可以为用户提供更加直观易懂的空气质量检测信息。根据不同的空气质量指数还可以为用户提供相应的提醒建议或者报警功能，当家居环境湿度过高时提醒用户打开窗户注意通风，并且能够直接提出相应的有效措施等建议，这样有助于人体呼吸健康；当出门在外的时候，家中如果发生煤气泄漏或者着火的情况，如当空气中危险气体浓度将要超过标准数值后立刻发出警报提醒主人采取相应的应对措施。

与此同时，本次毕设的一个需要改进的地方就在于居室内某一个参数需要通过分布在居室不同空间位置的多个传感器进行数据采集。根据各个传感器终端节点具体的分布情况，同样使用不同的比例权重对不同传感器所采集的数据进行分析，从而避免了主观因素的影响。

第五章 接入互联网：网络应用开发

5.1 服务器搭建

5.1.1 HTTP 协议

从一个整体上的角度来考虑现在的互联网通讯，计算机硬件资源就是生活的土地，驱动程序是地基，操作系统便是建筑物，应用程序就是生活在房子里的人们。与此同时，网络就可以类比成这样一个家庭和外界进行沟通的邮政系统，对亲朋好友的祝福与问候是书信的内容。而且，如果想要要合理地投送人们的真情惬意的话，就需要 HTTP、IP/TCP 或 UDP 等通信协议作为信封，将地址、发件人、收件人和时间等信息作为报头，而内容则还可以被封装起来。

因为所涉及的内容较多，这里仅仅就最常见的 HTTP 通信协议做简要介绍。HTTP 协议(HyperText Transfer Protocol)即超文本传输协议，广泛应用于 Web 应用中服务器和客户端之间所进行的通信会话。使用面向对象编程的思想来说的话就是:其会话的结构是一个简单的请求/响应序列，即浏览器发出请求，然后服务器做出响应。

TCP 协议(Transmission Control Protocol)即传输控制协议是一种传输层通信协议，面向连接的端到端。因此，建立 TCP 连接之后才可以使用 HTTP 协议进行数据传输，这也就是传说中的“三次握手”。IP(Internet Procotol)即网络通信协议，这是为互联网中的计算机之间进行通信所设计的网络协议。简单来说就是，信封上面的发件人地址和收件人地址，或者说本机和目标机器在英特网上面的位置坐标。

5.1.2 内网穿透原理

由于学校的网络连接处于校园网内部，也就是说树莓派处于局域网内，只能分配到一个内网 IP。而微信开发模式中服务器配置要求填写公网 IP，并且只允许 80 端口。这时需要 ngrok 进行内网穿透，使外网能够直接访问到树莓派。ngrok 是一个反向代理的隧道工具，通过在公共的端点和本地运行的 Web 服务器之间建立一个安全的通道。ngrok 可以捕获和分析所有通道上的流量，便于后期分析和重放。其原理是建立一个公网到 localhost 的基于 TCP 之上的端到端的隧道，让居于内网主机上的服务可以暴露给公网，俗称内网

穿透。并且支持对隧道中数据的 introspection（内省），支持可视化的观察隧道内数据，并 replay（重放）相关请求（诸如 HTTP 请求）。因此 ngrok 可以很便捷的协助进行服务端程序调试，尤其在进行一些 Web server 开发中。ngrok 更强大的一点是它支持 tcp 层之上的所有应用协议，也就是说与应用层协议是没有关系的。比如：你可以通过 ngrok 实现 ssh 登录到内网主机，也可以通过 ngrok 实现远程桌面(VNC)方式访问内网主机。

5.1.3 Web 服务器

使用 WSGI（Python Web Server Gateway Interface）能够确保 web 服务器兼容多个 web 框架的同时而又不用写代码来改变 web 服务器或者 web 框架。WSGI 允许开发者分别选择 web 框架和 web 服务器，因此混合使用匹配的 web 框架和服务器来满足你的需求。web 服务器必须实现 WSGI 的服务端接口，现在所有的 Python web 框架已经实现了 WSGI 的框架端接口，所以无需修改代码来适配一个特殊的 web 框架，服务端和框架开发者能够集中注意力于它们想关注的方面。下面是一个通过 Flask 框架搭建的最简单的 web 应用程序：

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello Raspberry Pi!'

if __name__ == '__main__':
    app.run()
```

5.2 微信公众号开发

5.2.1 微信公众号简介

微信已经成为了移动手机端的超级 App，拥有了海量用户群体，微信公众平台也成为近期国内最火热的开发平台之一，贴近用户、易于开发的优点也使得微信公众账号成为本次毕设监控系统的首选移动终端平台。微信公众号目前分为订阅号、服务号和企业号三种类型，均有两种模式以供使用，前者为编辑模式，可在微信公众号后台手动编辑文章推送文章给用户、设置关键字自动回复等，另一种就是开发者模式，可以将公众账号接入相应的 web 应用服务器，从而可以在服务器上编写程序实现很多自动化的功能。

5.2.2 接入微信服务器

微信用户向公众平台提交信息后，微信服务器将发送 GET 请求到填写的 URL 上，并且带上四个参数:

表格 5-1 微信服务器验证参数

参数	描述
signature	微信加密签名
timestamp	时间戳
nonce	随机数
echostr	随机字符串

然后开发者通过检验 signature 对本次请求进行校验。若确认此次 GET 请求来自微信服务器，将原样返回 echostr 参数内容，则接入生效，否则接入失败。signature 结合了开发者填写的 token 参数和请求中的 timestamp 参数、nonce 参数。

加密/校验流程如下:

1. 将 token、timestamp、nonce 三个参数进行字典序排序
2. 将三个参数字符串拼接成一个字符串进行 sha1 加密
3. 开发者获得加密后的字符串可与 signature 对比，标识该请求来源于微信

简要的检验 signature 程序:

```
def _check_hash(data):
    #sha1 加密算法
    signature=data.signature
    timestamp=data.timestamp
    nonce=data.nonce
    #在微信公众平台里输入的 token
    token="pi"
    #字典序排序
    list=[token,timestamp,nonce]
    list.sort()
```

```
    sha1=hashlib.sha1()
    map(sha1.update,list)
    hashcode=sha1.hexdigest()
    #如果是来自微信的请求，则回复 True
    if hashcode == signature:
        return True
    return False
```

5.2.3 微信号功能开发

本空气监控系统的微信号功能开发基于非常优秀的开源框架 WeRoBot，框架的开发者仍然是一名高中生，但不妨碍这是一个很好用的框架。WeRoBot 是一个微信机器人框架，主要实现了多种类消息的获取和推送，并且封装了官方的操作接口，例如创建菜单栏、微信支付等功能。笔者现主要实现了菜单栏的创建，主要分为三个模块，如下图所示，第一个为树莓派控制模块，能够在子菜单中查看树莓派的工作状态，并能够对树莓派采集和上传传感器数据的行为进行操控；第二个是树莓派的传感器数据采集模块，可以分别查看 PM2.5、液化气、烟雾、温度和湿度的具体详情，并给出一定的空气质量评定指数；第三个则是关于本人和本次毕业设计的相关信息，分别为毕设论文、简历和博客的菜单入口。

下面是其中一个实现查看室内温度的程序示例：

```
import json
import werobot
from werobot.client import Client #从 werobot/client.py 中导入 Client 类
#初始化 WeRoBot 机器人，开启 Session 功能
robot = werobot.WeRoBot(token='pi', enable_session=True)

@robot.text
def first(message, session):
    if 'last' not in session:
        session['last'] = message.content
        return "这是你第一次跟我说话"
    else:
        reply = session['last'] #回复上一次的对话内容
```

```
        session['last'] = message.content #保存本次对话内容
        return reply

@robot.key_click("TEMP") #点击“温度”菜单栏
def get_air_hum():
    sensor_data = get_sensor_data() #获得传感器信息
    if (sensor_data):
        return "室内温度:{0:.2f} °C".format(float(sensor_data['temp'])) #返回温度信息，保留小数点两位
    else:
        return err_msg

def get_sensor_data():
    #通过子进程调用 shell 脚本，获取最新的数据信息(即最后一行)
    output = subprocess.check_output("tail ./data/sensor.log -n 1", shell = True)
    sensor_data = eval(output) #从字符串中提取字典数据结构
    print "data:", sensor_data
    return sensor_data

with open('./data/menu.json', 'r') as f: #安全打开 json 文件
    menu_data = json.load(f) #加载字典

# 初始化微信 API 客户端，并创建菜单
client = Client("wx6eff06f20136ac85", "f2cafa0e5900a415e812ac2ef557d0f6")
client.create_menu(menu_data)

# 在 8888 端口运行微信程序
robot.run(host='0.0.0.0', port=8888)
```

图 5-1 微信公众号控制界面

5.3 Web 应用开发

5.3.1 数据上传与显示

随着物联网的迅速发展，在国内外基于树莓派或 Arduino 的 web 应用平台也层出不穷，国内正在做的主要有三家:Yeeklink 起步最早也最成熟，Weline 是专门针对于微信公众号所开发的硬件互联平台，而智城云则是中科院推出的物联网平台。Yeeklink 是国内第一个开放的公共物联网接入平台，本监控系统基于成熟互联网应用 yeelink 的页面展示及控制模块。此次页面展示及控制采用 Yeelink 网站提供的应用来完成，目的是使传感器数据的接入、存储和展现变得轻松简单，且支持多种方式显示(web、app 等)，可以大大减少重复工作量，界面也较为友好。后台程序在读取传感器信息之后，处理过后将其存储于记录文件中，再使用 yeelink 网站提供的 API 进行上传。只要并获取 U-ApiKey 以及设备和传感器的 URL。然后在监控系统控制程序中既可以通过相应 API 不断发送数据，在网站上还可以图形化显示传感器数据曲线，监测到实时的传感器数据并能够设置动作，如传感器数据异常则发送邮件通知用户等操作。

这里介绍温湿度数据的上传、显示和统计:

```
#coding=utf-8
import time
import json
import requests
import subprocess

#yeelink api 配置
api_url='http://api.yeelink.net/v1.1'
api_key='64a20e273bf186e50b2cd30b936743dd' #填入专属的 api key
api_headers={'U-ApiKey':api_key,'content-type': 'application/json'}
raspi_device_id = 20814
air_temp_sensor_id = 41044
air_hum_sensor_id = 41045

#上传到 yeelink
def upload_to_yeelink(sensor_value, sensor_id):
    url=r'%s/device/%s/sensor/%s/datapoints' % (api_url,raspi_device_id,sensor_id)
    strftime=time.strftime("%Y-%m-%dT%H:%M:%S")
    data={"timestamp":strftime , "value": sensor_value}

    res=requests.post(url,headers=api_headers,data=json.dumps(data))
    print "status_code:",res.status_code

def main():
    while True:
        # Reading data back
        output = subprocess.check_output("tail ../data/sensor.log -n 1", shell = True)
        sensor_data = eval(output)
        print "data:", sensor_data

        air_temp = sensor_data["temp"]
        print "air_temp:",air_temp
        upload_to_yeelink(air_temp, air_temp_sensor_id)
```

```
    air_hum = sensor_data["hum"]
    print "air_hum:",air_hum
    upload_to_yeelink(air_hum, air_hum_sensor_id)

    print "*"*40
    #休眠 10 秒
    time.sleep(10)

if __name__ == '__main__':
  main()
```

最终在网页上所显示的图形页面如下:

图 5-2 数据展示界面图

5.3.2 智能报警功能

传感器距离火源的远近，将会直接影响所检测到的烟雾浓度，所以必须要设置合适的电压阀值以确定初始烟雾浓度。MQ 烟雾传感器对可燃气体有着很高的灵敏度，根据这一特性非常适合布置与厨房浴室等可燃气体易溢出的地方。根据传感器所采集到的数据进行分析，监控系统在发现数据异常，如烟雾和液化气的浓度数据偏高，则是就需要主动提醒用户。智能报警功能的实现分为三种，一种是通过 yeeklink 平台的设备工作功能，设置数据阀值，从而一旦触发就自动采取行动，但是其自带的功能有限。另一种则是借助于第三方的手机通知类 APP，如 Instapush，通过设置好事件和通知内容的 API 之后，就可以编程实现对危险情况的通知了。最后一种则还是直接通过微信公众账号进行通知，由于微信 API 的限制，官方为了防止垃圾信息的推送，普通的订阅号没有主动向用户发送消息

的权限。但是在这种情况下，还是可以采用爬虫的方式模拟登陆微信公众号的后台，通过操作页面的方式发送信息，主动向用户通知家中的危险情况。

```
import json
from wechat_sdk import WechatExt
wechat = WechatExt(username='2205955115@qq.com', password='liqing123')

# 主动发送消息
def send_msg(msg="成功啦！"):
    user_info_json = wechat.get_top_message()
    user_info = json.loads(user_info_json)
    print "*"*20 + "报警啦！" + "*"*20
    try:
        wechat.send_message(user_info['msg_item'][i]['fakeid'], msg)
    except:
        print 'ERROR! No more user!'

@robot.key_click("PM2.5")
def warning():
    sensor_data = get_sensor_data()
    PM_data = float(sensor_data['PM2_5'])
    MQ_data = float(sensor_data['MQ-2']) + float(sensor_data['MQ-5'])
    if (PM_data > 3000 and MQ_data > 250):
        send_msg("家中火情！家中火情！")
        air_index = "很差"
    elif (PM_data > 1050 and MQ_data > 200):
        send_msg("空气质量太差，打开窗户！")
        air_index = "差"
    elif (PM_data > 300):
        air_index = "一般"
    else:
        air_index = "非常好"
    if (sensor_data):
```

```
        return '''
        粉尘浓度:{0} ug/m3
        空气指数:{1} {2}
        '''.format(float(sensor_data['pm2.5']), float(sensor_data['PM2_5']), air_index)
    else:
        return err_msg
```

以下为险情发生时向用户发送的报警信息：

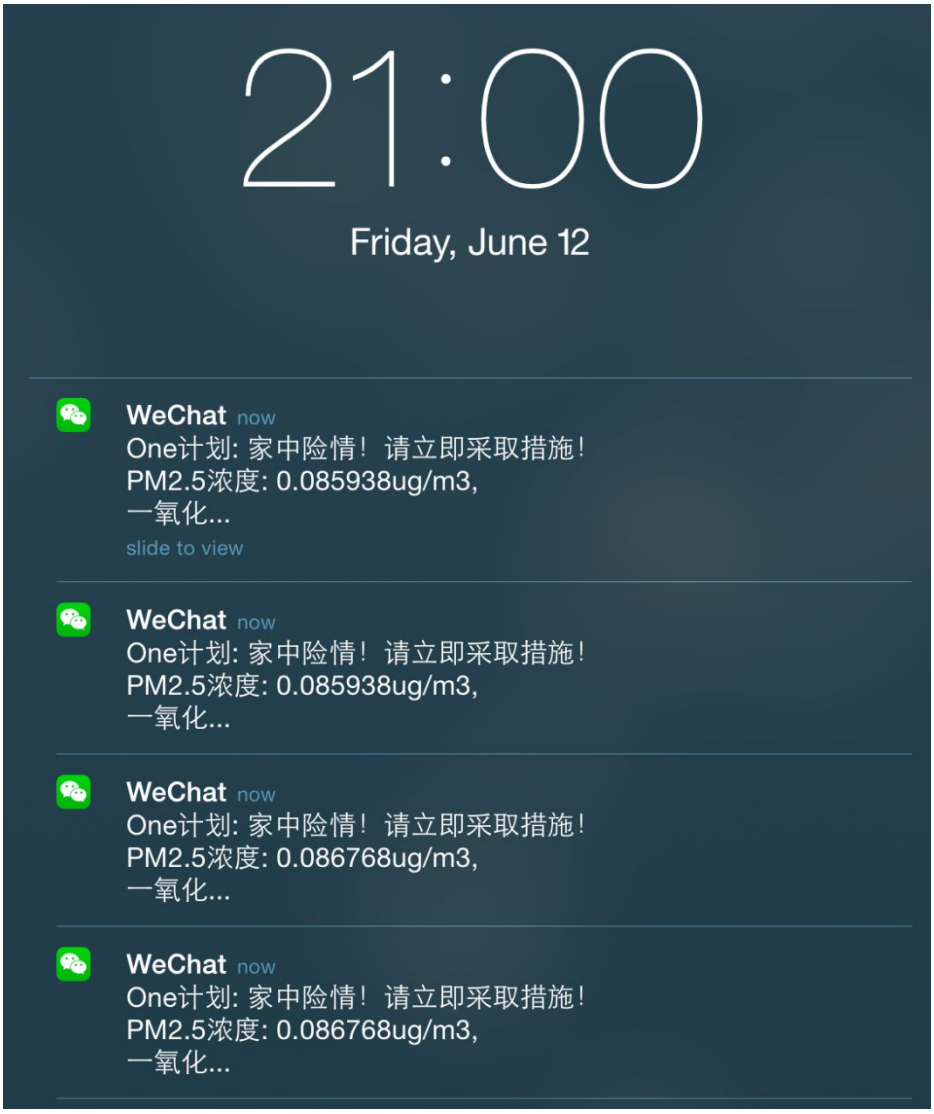

图 5-3 微信报警提示

总结与展望

在本次毕业设计中所完成的室内空气质量监控系统，是一种对智能家居方向的物联网产品的有趣尝试。在互联网迅速发展并逐渐改变着传统行业的同时，连接网络的硬件产品也不再局限于个人电脑和手机终端。伴随着工业时代所带来的空气污染等环境问题是目前国内亟需解决的问题之一，本项目也就是在这样的背景之下，搭建了一套基于物联网的家庭空气检测和控制中心。

本项目完成度较高，已实现了基本的空气质量数据的采集和分析，并且用户能够直接在移动端与监控系统进行相应的功能交互，具有一定的产品可用性。但是整套系统还存在一些问题，具有很大的改进空间。首先需要指出的是树莓派作为一款开源硬件，其实与真正的电子消费品还是存在产品形态上的巨大差距。树莓派成本较低和丰富可扩展等特性使之成为一款非常优秀的硬件“玩具”，非常适合极客们在产品初期快速实现自己的想法。但是真正做出一款消费级的家庭空气质量监控系统，仅仅靠一款开源硬件来进行扩展是远远不足以实现的，还需要很多商业上的考虑和实现。那如果说这是一套针对于极客的“个人”家庭物联网系统实现，这个系统的功能也有待完善，需要进一步改善的主要包括以下几个方面:首先是空气质量指数评定功能，其实在这方面的数据处理存在一定的片面性，由于不同物质对于空气质量的影响程度是不同的，但是目前来说只是做了简单的区间化处理，而且并没有做实验对其评定效果进行完整性测试。另一个方面则是项目对于其他第三方平台的依赖性，可以为数据可视化这个功能自主搭建 Web 平台，使用原生的数据处理库进行可定制化的数据显示效果。还需要完善的传感器数据的持久化存储，目前来说并没有使用数据库而是直接利用单一的文本文件存储相关数据，不利于数据的修改、删除和查询等操作。

致谢

在毕设项目进行之前，笔者在享受着互联网所带来的极大便利的同时也对互联网行业有所向往，非常关注开源硬件与智能家居的行业动向，也曾设想为未来的家居环境亲手搭建一个智能化的家庭物联网系统。此项目属于自主命题，意在将自己在大学里面所接触到的所有硬件和软件方面的知识都串连起来，并亲手做出一个完全由自己设计和实现的硬件产品。在此特别感谢晏寄夫老师能够接受我自己所准备的毕业设计题目，并且在整个毕设项目和论文撰写过程中对我所进行的非常细心、耐心和悉心的指导。正是在晏老师的指引下，我才能够顺利完成大学里最为重要的一个实践项目。与此同时，晏寄夫老师每个星期都不辞辛苦地从九里校区赶到犀浦校区对我进行毕设指导，这样的老师是非常值得我们尊敬和学习的，老师对学生热心体贴的关怀和对教学工作孜孜不倦的态度也将影响我的一生。

另外需要感谢的是互联网的开源社区，开源的力量使得每个人都能够发现、使用、和交流自己的开源项目。在整个项目开发过程中，我使用开源硬件树莓派和 Arduino，并参考了很多相关开源社区所贡献出的教程和代码库。个人也决定本次基于开源硬件所扩展开发的整个监控系统也将进行开源许可，在享受开源的同时也为开源做出自己的一份贡献。如果不是树莓派的强大，我也不可能在短时间内做出这样一个较为完善的系统。有多大的想象力，就可以做多大的事情。

最后，也要感谢学校对我们大学四年的悉心教导，学校所提供的教学资源对于大学生来说是一笔巨大的财富，我在大学过程中所学到的知识和所实践的项目都与母校密切相关。当然也要感谢在大学里所遇到的各位尊敬的老师和可爱的同学们，老师教导我们的所有知识和展现出来的职业素养都将影响我们终身，和我一起努力奋斗过的同学们也充满青春的活力与激情，愿今后真情常驻。总之，对所有在大学中帮助过我的人致以最真诚的感谢。

参考文献

[1]陈儒敏，侯思名，颜江等.基于开源软硬件的智能家居系统设计与实现[J].现代计算机.2013

[2]Abraham, K. Pandian, S.A Low-Cost Mobile Urban Environmental Monitoring System.Intelligent Systems Modelling & Simulation (ISMS).2013

[3]Mladen Milosevic, Armen Dzhagaryan, Emil Jovanov, Aleksandar Milenković.An Environment for Automated Power Measurements on Mobile Computing Platforms.2013

[4]Ivan Oliveira Nunes, Magnos Martinello, Antˆonio A. F. Loureiro.Designing a Low Cost Home WSN for Remote Energy Monitoring and Electronic Devices Control.2015

[5]Luiz H. Nunes , Luis H. V. Nakamura , Heitor de F. Vieira , Rafael M. de O. Libardi , Edvard M. de Oliveira , Julio C. Estrella , Stephan Reiff-Marganiec.Performance and Energy Evaluation of RESTful Web Services in Raspberry Pi.2014

[6]Sarthak Jain, Anant Vaibhav, Lovely Goyal.Raspberry Pi based Interactive Home Automation System through E-mail.2014

[7]S. Sathya Prabha, A. John Paul Antony, M. Janaki Meena, and S. R. Pandian.Smart Cloud Robot using Raspberry Pi.2014

[8]Fung Po Tso, David R. White, Simon Jouet, Jeremy Singer, Dimitrios P. Pezaros.The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures.2013

[9]Sheikh Ferdoush, Xinrong Li.Wireless Sensor Network System Design using Raspberry Pi and Arduino for Environmental Monitoring Applications.Procedia Computer Science 34 (2014) 103 – 110

[10]邢波.基于 RaspberryPi、 Zigbee 技术的无线智能家居系统设计.数字技术与应用.1007-9416(2014)06-0059-02

[11]郑世珏，徐 虹.基于 Raspberry Pi 的远程监测系统的设计与实现.微型机与应用.674-7720(2014)19-0105-03

[12] 李杨，郭培源，刘波，向玲孜.基于嵌入式技术的居室健康环境监测系统.《电子技术应用》2014 年 08 期

附录

项目结构图:

```
.
├── Adafruit_DHT
├── __init__.py
├── arduino_c
│   ├── arduino_c.ino
├── data
│   ├── menu.json
│   └── sensor.log
├── ngrok
├── sensor
│   ├── PCF.py
│   ├── __init__.py
│   ├── arduino.py
│   ├── pi_info.py
├── server
│   └── yeelink.py
├── test
│   ├── __init__.py
│   ├── dht11_simple.py
│   ├── dht11_subprocess.py
│   ├── dhtreader.so
│   ├── dhtreader_argv.py
│   ├── link_arduino.py
├── weixin.py
└── werobot_session
```

微信公众号功能代码:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import sys
reload(sys)
sys.setdefaultencoding('UTF-8')

import subprocess
import dht11
from sensor import pi_info

from smbus import SMBus
bus = SMBus(1)

import json
import requests
import werobot
```

```
from werobot.client import Client
robot = werobot.WeRoBot(token='pi', enable_session=True)

from wechat_sdk import WechatExt
wechat = WechatExt(username='2205955115@qq.com', password='liqing123')

err_msg = "出错啦，请重试！"

# 获取与最新一条消息用户的对话内容
def get_user_info():
    user_info_json = wechat.get_top_message()
    user_info = json.loads(user_info_json)
    print wechat.get_dialog_message(fakeid=user_info['msg_item'][0]['fakeid'])

# 主动发送消息
def send_msg(msg="成功啦！"):
    user_info_json = wechat.get_top_message()
    user_info = json.loads(user_info_json)
    print "*"*20 + "报警啦！" + "*"*20
    try:
        wechat.send_message(user_info['msg_item'][i]['fakeid'], msg)
    except:
        print 'ERROR! No more user!'

def simsimi(ask):
    baseurl = u'http://sandbox.api.simsimi.com/request.p?key=d31ca5f8-9bb2-4dab-b712-
39ea43416b95&lc=ch&ft=1.0&text='
    url = baseurl+ask
    r = requests.get(url)
    res = json.loads(r.text)
    return res['response']

def get_sensor_data():
    # Reading data back
    output = subprocess.check_output("tail ./data/sensor.log -n 1", shell = True)
    sensor_data = eval(output)
    print "data:", sensor_data
    return sensor_data

@robot.subscribe
def subscribe():
    return "Hello! 这是我的毕设:Pi 的呼吸之旅"

@robot.filter("报警")
def warning():
    send_msg()
    return "报警啦！"

@robot.text
def first(message, session):
```

```
    if 'last' not in session:
        session['last'] = message.content
        return "这是你第一次跟我说话"
    else:
        try:
            reply = simsimi(message.content)
        except:
            reply = session['last']
            session['last'] = message.content
        return reply

@robot.key_click("PI_INFO")
def get_Pi_info():
    CPU_temp = pi_info.get_CPU_temp()
    GPU_temp = pi_info.get_GPU_temp()
    CPU_usage = pi_info.get_CPU_use()
    RAM_usage = pi_info.get_RAM_use()
    DISK_percent = pi_info.get_Disk_info()[3]
    return '''
    CPU 温度:{0} °C
    GPU 温度:{1} °C
    CPU 使用率:{2} %
    RAM 使用率:{3} %
    硬盘使用率:{4}
    '''.format(CPU_temp, GPU_temp, CPU_usage, RAM_usage, DISK_percent)

@robot.key_click("TEMP")
def get_air_hum():
    sensor_data = get_sensor_data()
    if (sensor_data):
        return "室内温度:{0:.2f} °C".format(float(sensor_data['temp']))
    else:
        return err_msg

@robot.key_click("HUM")
def get_air_hum():
    sensor_data = get_sensor_data()
    if (sensor_data):
        return "室内湿度:{0:.2f} %".format(float(sensor_data['hum']))
    else:
        return err_msg

@robot.key_click("LIGHT")
def get_air_hum():
    #得到光照强度
    bus.write_byte(0x48, 1) # set control register to read channel 1
    light = bus.read_byte(0x48) # read A/D
    if (light):
        return "光照强度:{0:.2f} 勒克斯".format(1080 - float(light))
```

```
    else:
        return err_msg

@robot.key_click("SWITCH")
def switch():
    return "open/close"

@robot.key_click("PM2.5")
def switch():
    sensor_data = get_sensor_data()
    PM_data = float(sensor_data['PM2_5'])
    if (PM_data > 3000):
        air_index = "很差"
    elif (PM_data > 1050):
        air_index = "差"
    elif (PM_data > 300):
        air_index = "一般"
    elif (PM_data > 150):
        air_index = "好"
    elif (PM_data > 75):
        air_index = "很好"
    else:
        air_index = "非常好"
    if (sensor_data):
        return "粉尘浓度:{0} ug/m3\n 空气指数:{1} {2}".format(float(sensor_data['pm2.5']),
float(sensor_data['PM2_5']), air_index)
    else:
        return err_msg

@robot.key_click("CO_GAS")
def switch():
    sensor_data = get_sensor_data()
    if (sensor_data):
        return "液化气浓度:{0} ppm".format(float(sensor_data['MQ-2']))
    else:
        return err_msg

@robot.key_click("SMOKE")
def switch():
    sensor_data = get_sensor_data()
    if (sensor_data):
        return "烟雾浓度:{0} ppm".format(float(sensor_data['MQ-5']))
    else:
        return err_msg

@robot.click
def click_event():
    return "我还没有准备好！"
```

```
# Reading menu data back
with open('./data/menu.json', 'r') as f:
    menu_data = json.load(f)

client = Client("wx6eff06f20136ac85", "f2cafa0e5900a415e812ac2ef557d0f6")
client.create_menu(menu_data)

robot.run(host='0.0.0.0', port=8888)
```