智能室内湿度检测器设计与实现

摘要

本文介绍了一款以树莓派为核心，以DHT11温湿度传感器和红外模块为基础的智能湿度检测器的设计。主要内容包括对研究背景及意义的介绍、器件选型原则以及系统方案的确定并在此基础上完成了软硬件的设计及实物调试工作。其中对器件选型主要从应用领域以及成本出发，最终选择以树莓派作为系统核心，对于此次设计可以满足需求。同时从需求出发，选择了DHT11作为系统中的湿度传感器。二者结合很好的完成了此次设计。文中对包括硬件以及软件在内的设计思想以及具体流程均作了详细介绍。

关键词：树莓派 DHT11 红外通信

1 绪论

1.1 背景与意义

通信技术和传感器技术以及计算机技术并称为现代信息技术的支柱。随着科学技术的不断完善与发展，现代信息技术也取得了长足的进步。微电子理论的完善与发展促进了集成电路的研究与发展。得益于这一进步，计算机技术在过去的几十年间飞速发展，从最初的8086处理器到当今最先进的处理器，经历了跨越式的发展。目前，世界上主流的处理器以X86架构与ARM架构，二者分工明确，前者主要存在于PC机与服务器中，后者则由于其较好的能耗控制而广泛应用于手持设备中。随着近些年以来各行各业对于生产自动化的需求，传感器与通信技术也在不断的研究与发展之中。特别是进入新世纪以后，随着智能家居这一概念的提出，很多新式的传感器逐渐进入市场，使得智能家居的实现成为可能。在人们生活水平逐渐提高的背景下，家居的自动化与智能化成为提高生活品质的重要因素。其中，通过现代信息技术提供一个适合的家居环境，例如适宜的温度以及湿度等，并在无需人为干预的情况下自动调整，已经成为智能家居的最基本需求。因此，设计一款智能室内湿度检测器，对室内实时的湿度情况进行检测并加以控制十分必要。本文便是在此背景下，介绍了一款以ARM处理器为基础的智能室内湿度检测器的设计与实现方法。

1.2 研究目的和方法

此次设计以微处理器作为核心器件，配合湿度传感器模块以及红外发射接收模块来实现一款智能室内湿度检测器的设计。该智能湿度检测器主要用于家居环境中，通过传感器检测环境中的湿度值并通过信号线将湿度值传输到微处理器中，由微处理器判断当前的湿度是否处于让人体感觉舒适的湿度范围，当当前的湿度值过高时，则启动空调的除湿功能以降低环境中的湿度，当湿度值恢复正常后再关闭空调的除湿功能，以避免环境过于干燥，引起人体不适。此次设计以功能为导向，采用的研究方法如下：查阅相关文献资料，明确系统的设计目标，从功能出发确定系统的设计方案；分模块对系统的各部分元件进行选型并分析其合理性；完成系统的硬件设计并编写相应的软件；制作实物验证系统功能。

1.3 论文章节安排

本文介绍了一款智能室内湿度检测器的设计，主要内容及章节安排如下：

1. 绪论

介绍此次设计的背景以及意义，并介绍此次设计的方法。

1. 系统设计方案

介绍系统的功能、设计目标以及器件选型。

1. 系统硬件设计

介绍系统总体以及分模块的硬件设计原理。

1. 系统软件设计

介绍系统软件设计方法并给出流程图

1. 系统调试

介绍设计完成的实物并在此基础上介绍实物制作过程中所遇到的问题并给出解决办法。

2 系统设计方案

2.1 方案论证

2.1.1 系统结构

按照该系统的需求，从功能角度出发，有两种方案可供选择。第一种是使用单片机采集湿度传感器的信息，然后通过蓝牙串口的方式将采集到的湿度信息发送到手机并由手机APP接收，然后通过手机上的红外发射接口发射控制信号。这种方式下主要工作由单片机完成，手机调用红外接口即可完成控制，设计简单，但是也存在着一定的不足。在此方案下需要一部手机作为系统的一部分，当手机电量耗尽后将无法完成系统功能。另外，如果手机一直充电可以解决供电问题，但手机长期处于充电状态可能导致安全隐患。因此该方案不够完善，存在一定的缺陷。第二种方案是采用微处理器，配合湿度传感器以及红外模块，由微处理器读取湿度传感器的数据并进行判断，并由此决定是否通过红外接口发送控制信号。该方案下系统可以长期带电工作，不存在方案一中的隐患情况。唯一的不足在于红外发射部分的软件需要自己实现，给系统开发带来一定的困难。但是相较于可能存在的危险而言，这个困难可以克服。因此，选择方案二作为最终的系统结构。

2.1.2 器件选型

2.1.2.1 主控制器选型

器件的选择对于此次设计至关重要。选择使用简单，精确度高，可替换性好的器件能够在很大程度上简化此次设计。此次设计中按照既定的方案，主控制器主要由三种可选方案。

第一种是单片机，例如常用的STM32系列单片机，该系列单片机具有较为广泛的用途，常用于各类嵌入式系统的设计中，而且其外设丰富，因此广泛应用于智能家居中。单片机作为一类常用的控制器，具有价格低廉的优势，而且其所具有的外设能够满足此次设计需求，但是问题在于此次设计中需要使用红外接口，这部分的软件编写难度较大。

第二种方案是使用FPGA作为控制器。FPGA目前在各类系统中同样具有广泛应用，并且其具有使用灵活的特点，能够根据不同的需求设计出对应的功能，这一点对于此次设计十分必要。而且此次设计中与传感器和红外接口的通信均按照一定的时序进行，使用FPGA作为主控制器进行开发将具有很大的优势。但是FPGA价格较为昂贵，这在极大的增加系统成本。

第三种方案则是选择近年来较为流行的以ARM处理器作为核心的微型电脑，这类设备具有单片机的优点，即外设丰富。例如常见的通信接口（串口、以太网以及USB接口等），同时还具有很多引出的I/O引脚，能够方便的调用这些引脚完成设计。除此之外，此类设备一般都具有完善的文档资料以及外设驱动，能够快速的搭建系统。更为重要的是，使用此类设备对于硬件开发经验的要求较低，而且大多数均支持python语言，这对于不了解硬件底层的人员而言极大的降低了开发门槛。此类设备一般都运行各种Linux发行版，能够方便的使用网络上的开源资源，加快开发进度。

通过上述论证可以看出，使用第三种方案对于此次设计而言具有较大的优势，因此选择方案三作为此次控制器的方案。

2.1.2.2 湿度选型

常见的湿度传感器有模拟和数字两种类型。模拟型的湿度传感器测量精度更高，这对于需要精确控制的场合而言是极为重要的。但是精度高同时也意味着价格昂贵，这对于此次设计而言不具有优势。同时模拟传感器由于其固有性质，容易受到干扰，这对于系统布线而言造成一定困扰。数字型湿度传感器相对于模拟型传感器而言精度稍低，但是对于此次设计中的使用场合而言不会造成太大影响。而且数字模拟器抗干扰能优于模拟型传感器，这对于系统的准确控制而言极为重要。除此之外，数字型传感器价格一般较低，能够在一定程度上降低系统开发成本。因此此次设计选择数字型湿度传感器作为最终选择。

2.2 系统设计方法

采用上述论证后的方案，此次设计按照模块化设计的思想，将系统分解为主控制器模块、湿度采集模块、红外接收模块以及红外发射模块，系统的结构框图如下所示：



图2-1 系统结构框图

其中，主控制器需要完成实时湿度值的采集并于设定的湿度值相比较，除此之外，还要根据比较结果判断是否需要打开空调的除湿功能。湿度采集模块的功能则是实时的采集环境湿度。系统第一次使用时通过采集所要控制的空调的控制信号，即要对设备的遥控器按键功能进行学习，这便是红外接收模块的作用，这一环节必不可少。红外发射模块的功能不言而喻，即传输主控制器发出的控制指令，并遥控空调打开除湿功能。

按照上一节的论证，此次设计更进一步地，选择树莓派作为主控制器，选择DHT`11作为湿度传感器，红外接收选择HS0038B、红外发射选择TSAL6200。以下将对各主要器件进行介绍。

2.3 主要器件介绍

2.3.1 树莓派

树莓派自2012年诞生以来，在全世界范围内引起了巨大的反响。 自树莓派面世以来，各种类似的产品不断出现，但仅有极少数最终生存下来。这一方面是因为树莓派的先发优势，另一方面也是因为树莓派软硬件支持度较高，使用方便。其模仿者往往出于成本等考虑，无法长期支持某一款产品，造成用户流失。树莓派迄今为止已经迭代了三代，从最初的单核、256M内存的发展到现在的树莓派3B+，已经具备了四核、1G内存，树莓派四代也已经在筹划阶段，不久将面世。树莓派作为目前世界范围内使用最广泛的微型电脑之一，能够很好地使用现有的开源软件，同时有广大的贡献者不断为其添砖加瓦，这使得树莓派能够不断发展。目前，树莓派广泛应用于智能家居中，这得益于其丰富的接口以及其易用性。使用python能够方便的调用其硬件接口，这对于开发效率以及成本而言十分友好。树莓派板载资源十分丰富，具有USB接口、以太网接口、HDMI接口、CSI相机接口等常用接口，同时还具有嵌入式系统开发中常用的SPI接口、串口以及IIC接口，如下图所示为树莓派外观图以及引脚图。

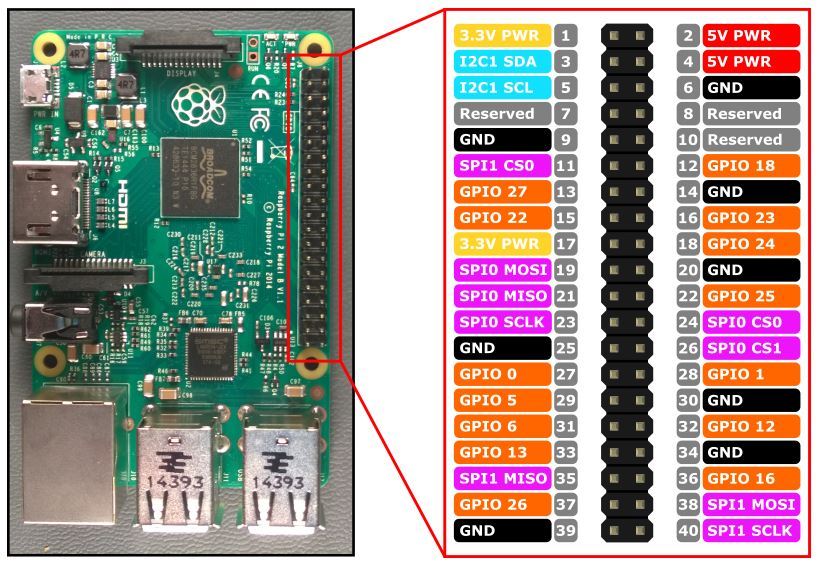


图2-2 树莓派外观及引脚图

2.3.2 DHT11

DHT11是一款温湿度传感器，其湿度测量精度为+-5%RH， 温度的测量精度为+-2℃，湿度的量程为20-90%RH， 温度的量程为0~50℃。其输出数据带有8bit的检验信息，可以用于验证数据传输的准确性。其内部集成了电阻式湿度传感器与NTC温度传感器以及一个八位的单片机，具有性能稳定、响应时间短、抗干扰以及性价比高的特点。在出厂时，每个DHT11都会经过校正，并且校正系数均以代码形式存储在其内部单片机的ROM中，内部单片机在读取数据时会读取这些校正系数对测量结果进行校正后再发送。DHT11使用单总线的形式传输数据，减少了I/O口的占用，便于日后对系统的扩展。该传感器与同类产品相比具有较小的体积，同时功耗也较低，因此在一些对系统体积以及功耗要求较高的场合其具有较大的优势。下图所示为DHT11的外观图。

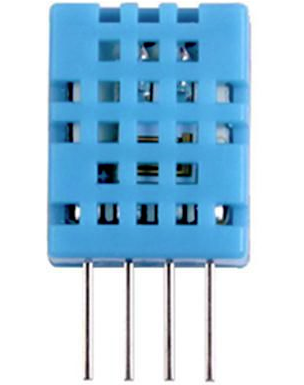


图2-3 DHT11外观图

2.3.3 HS0038B

HS0038B是一款集成化的红外通信接收模块，其内部电路集成了红外监测二极管，信号放大器，限幅器，滤波器，积分电路以及比较器等。HS0038B只需要提供电源输入便可以完成红外信号的接收。目前主流的红外信号传输协议有NEC协议，ITT协议，Nokia NRC 协议,Sharp协议等。可以看出这几种协议均是由业界领导企业制定的，其中以NEC协议使用最为广泛。HS0038B作为一款红外接收器，具有极强的对自然光的抗干扰能力，这对于红外通信系统的稳定性具有极为重要的作用，是通信正常进行的前提。除此之外其还具有极低的功耗，这对于低功耗的应用场合极为重要。HS0038B输出兼顾TTL和CMOS，因此能在不同类型的数字系统中使用。HS00338B有三个引脚，分别为电源引脚，地引脚以及信号引脚。如下图所示为其外观及引脚图。

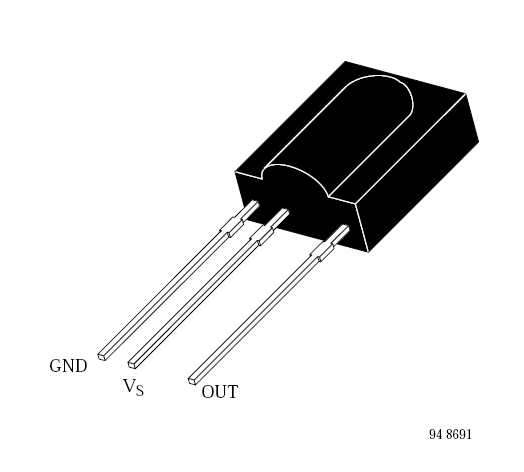


图2-4 HS0038B外观及引脚图

2.3.3 TSAL6200

TSAL6200是一款以砷化镓或砷镓铝为原料制成的红外发光二极管。因此，使用这款器件能够发射此次设计所需的红外通信信号。由于在原有技术上进行了较大改进，该器件能够提高发光强度达1倍以上。因此该器件适合作为标准红外发射器件的替代品。相较于原有器件，能够在相同功耗下发射出强度更大的信号，增加通信距离。目前，TSAL6200广泛应用于需要大功率红外通信发射源的场合，例如一些需要较远距离的通信。除此之外，其还可用于烟雾探测器中。此次设计中其主要用于发射红外通信的信号，控制空调工作与停止。TSAL6200一般采用蓝灰色的塑封，以增强抗干扰能力。如下图所示为其外观图。

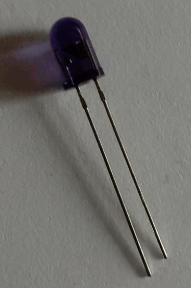


图2-4 TSAL6200外观图

3 系统设计

3.1 系统硬件设计

此次设计以树莓派为核心器件，以DHT11、 HS0038B以及TSAL6200作为外部设备。其中，外围设备通过杜邦线与树莓派相连。由于树莓派的供电接口数量有限，因此设计中还使用面包板对树莓派的接口进行扩展。通过树莓派的数据手册可知，树莓派的I/O引脚可以承受3.3V的电压，设计中所有外部设备的供电均采用3.3V，以免过高的电压损坏树莓派的引脚或者直接损坏处理器。下图所示分别为此次设计的红外遥控模块接线图以及湿度测量模块的接线图。

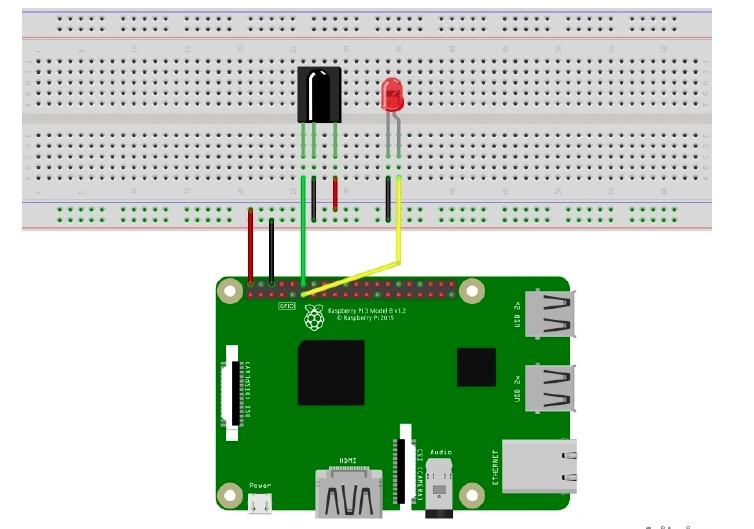


图3-1 红外发射与接收接线图

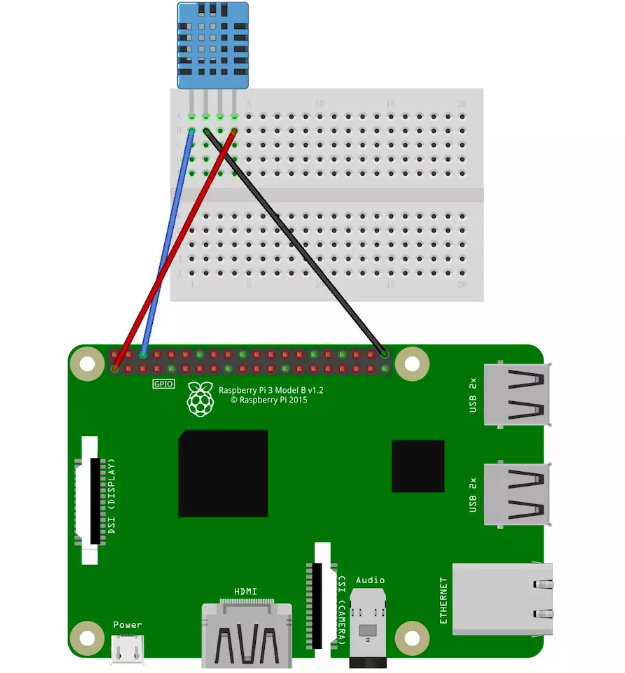


图3-2 湿度模块接线图

由图可以看出，红外接收模块的数据引脚与树莓派的第12号引脚相连，红外发送模块的数据引脚与树莓派的第11号引脚相连。湿度模块的数据引脚与树莓派的第6号引脚相连。在设计中，湿度模块使用制作完成的模块。该模块的外观如下图所示。

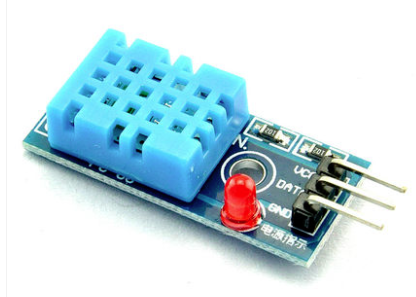


图3-3 湿度模块外观

上图所示的模块中，从左至右三个引脚分别为接地引脚，数据引脚以及电源引脚。该模块使用一个发光二极管作为电源指示灯，上电后该灯会亮起，以表明模块正常上电。该模块的原理图如下图所示。

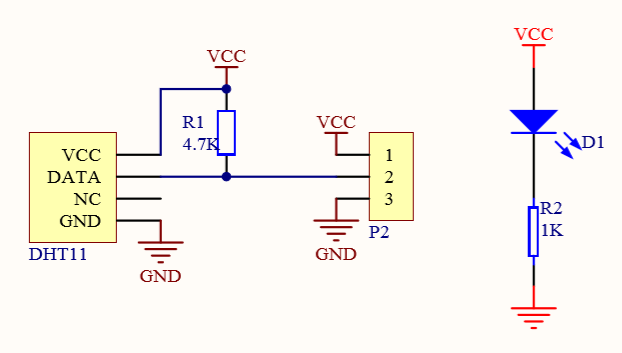


图3-4 湿度模块原理图

如上图所示为此次设计中使用的湿度模块的原理图，R1为上拉电阻，用于增强模块的驱动能力，以保证高低电平均能稳定，减少出错的可能性。D1为外观图中所示的发光二极管，R2为限流电阻，以保证发光二极管上的电流不会太大而导致发光二极管损坏。P2为外观图中所示的排针。按照DHT11数据手册的说明，在焊接该模块的时候应注意在260摄氏度的情况下持续焊接时间不可超过10秒，否则有可能造成器件永久性损坏或者灵敏度下降。

3.2 系统软件设计

3.2.1 系统软件流程



图3-5 系统软件流程图

如上图所示为此次设计中的程序流程图。在程序一开始需要对各个模块进行初始化设置，这其中包括对树莓派自身的初始化以及DHT11湿度模块的初始化。当初始化完成后，系统读取DHT11的湿度值，并与设定的舒适湿度相比，如果测得的湿度值较大，则通过红外遥控模块打开空调的除湿功能，否则不采取任何行动。当湿度值从较高恢复正常之后，关闭空调的除湿功能。以上便是系统整体的软件流程图。以下将分别对湿度模块以及红外遥控模块的驱动做一简单介绍。

3.2.2 湿度模块软件流程

此次设计中使用的湿度传感器DHT11为单总线设备，此类设备的特点是，数据通过一根数据线按照bit发送，直到所有的数据发送完成。从DHT11的数据手册可知，该器件一次发送流程中共会发送40bit的数据，分别为2个字节的湿度值，两个字节的温度值以及一个字节的校验值。发送过程中先发送湿度的整数部分，然后发送湿度的小数部分，之后发送的是温度的整数部分以及温度的小数部分，校验部分为前四个字节的数据相加后的结果的最后8位。因此通过读取所有数据并对前四字节的数据做加法后取最后8位与校验值相比较便可验证本次收发过程是否正确。DHT11发送数据的流程图如下所示。



图3-6 湿度模块软件流程图

DHT11在使用前需要经过唤醒，以便设备从低功耗模式进入高速模式。等待主机发送开始信号结束后，DHT11开始发送过程。主机的开始信号为一持续最少18毫秒的低电平信号，以保证DHT11准确的收到开始指令。如下所示为主机与DHT11之间唤醒过程的时序图。

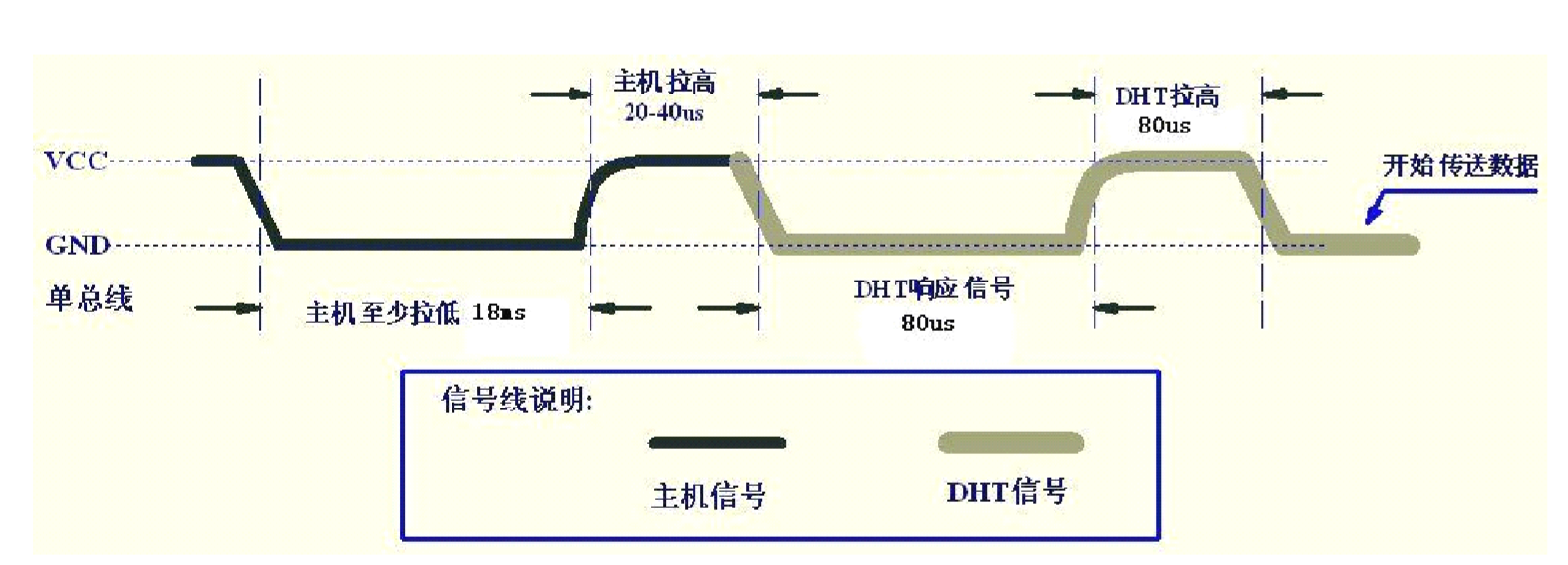


图3-7 DHT11唤醒过程时序图

从上图可以看出，主机在将总线拉低18毫秒后需要再将总线拉高，且该高电平应持续20~40毫秒，以保证从机能够正确响应。在此之后DHT11接管总线控制权并通过总线回复应答信号，将总线拉低，该过程按照图示应持续80微妙秒，随后DHT11再将总线拉高，该过程同样需要持续80微秒。在DHT11将总线拉高经过80微秒之后，数据开始经过单总线依次发送。发送过程的时序图如下图所示。

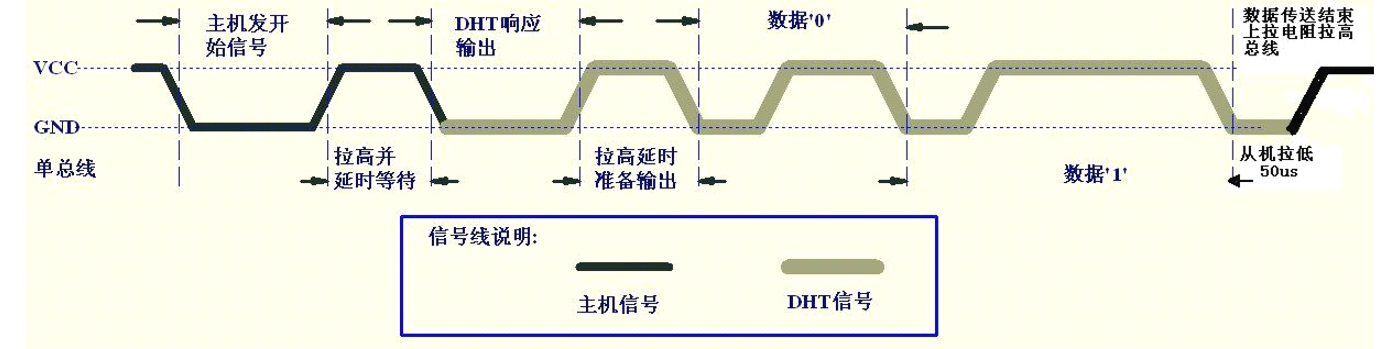


图3-8 DHT11数据发送时序图

可以看到上图的数据发送过程是在主机发送开始信号并收到从机应答之后开始的。从机发送的数据通过高电平持续时间的不同来定义此次发送的数据位是高电平或是低电平。查阅DHT11的数据手册可知，每一个数据位的发送过程都是以固定时长的低电平开始的。低电平结束后的高电平持续时间长的代表此次发送的是数据“1”，低电平结束后的高电平持续时间短的为“0”。下图所示为高电平持续时长的具体界定方法。

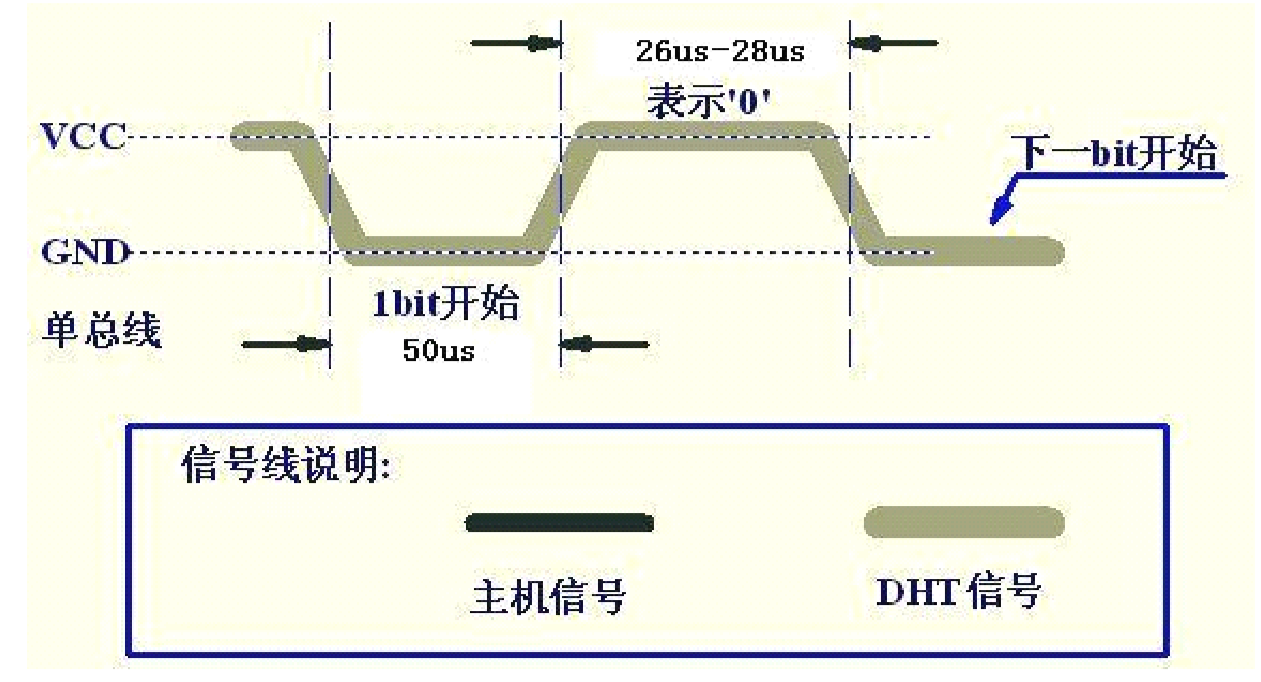


图3-9 DHT11低电平时序图

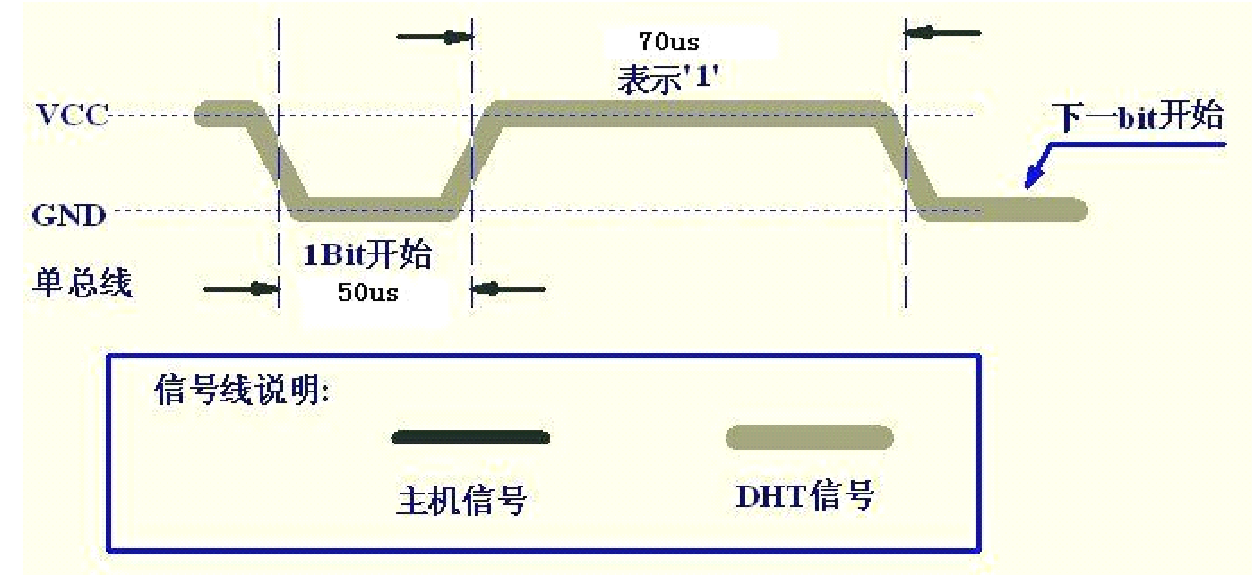


图3-10 DHT11高电平时序图

从上图可以看出，数据“0”和数据“1”在高电平段持续时间有着明确的定义。即26~28微秒的高电平代表数据“0”，而持续时间长达79微秒的高电平则代表数据“1”。通过上述时序图，能够明确发送过程的各个状态。

在使用DHT11的过程中需要注意的是，该器件分辨率不高，仅能达到整数级的分辨率，因此其32位数据位中仅有16位为有效数据，即第1个字节和第3个字节的数据。第2个字节和第4个字节的数据为无效数据。DHT11湿度模块的主要代码如下图所示。

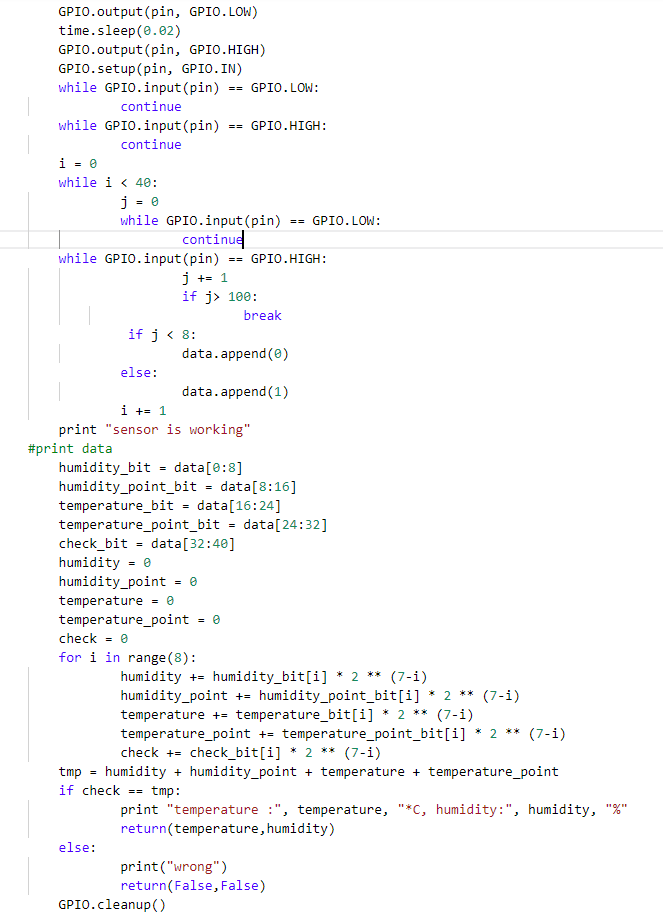


图3-11 DHT11模块部分代码

3.2.3 红外模块软件流程

红外模块软件包括控制指令的录制与发送，其中红外指令的录制过程是单独进行的。使用空调遥控器对准红外接收模块，按下相应功能键，系统即可记录下相应的数据并保存至文本文件。记录过程的软件流程图如下所示。



图3-12 记录过程的软件流程图

在上述的过程中，树莓派通过与红外接收模块相连的引脚，开启I/O口中断，记录下遥控器一次发送过程中的数据，然后将数据写入文本文件中。便完成了红外指令的录制过程。录制过程中记录下的是红外脉冲信号与空白信号持续的时间。这一文本文件在随后的控制过程中需要用到。下面大致说明遥控器的数据是如何发送以及接收的。

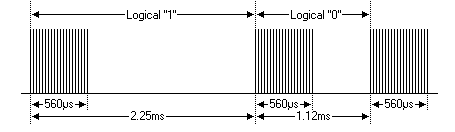


图3-13 NEC调制方式

NEC协议的红外数据发送过程与DHT11的数据发送过程类似。不同之处在于，NEC协议中的逻辑“1”是由脉冲和空白构成的，如上图所示的阴影部分实际上是一个个的脉冲波，从上图可以看出，逻辑“1”和逻辑“0”是根据二者空白部分的长短确定的。其中，逻辑“1”的总时长为2.25毫秒，而逻辑“0”的持续时长为1.12毫秒，约为逻辑“1”的时长的一半。可以看出，逻辑“1”和逻辑“0”均是以脉冲波作为起始信号的。

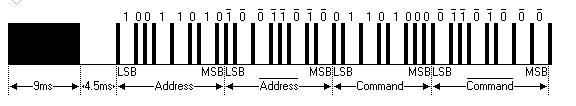


图3-14 NEC协议格式

NEC协议格式如上图所示，在一次完整的发送过程中，存在9毫秒的脉冲波与4，5毫秒的空白作为固定格式，随之而来的是地址数据以及指令数据。在这一过程中，根据NEC协议的调制方式，我们可以根据脉冲波持续的时间以及空白持续的时间作为依据，也就是说，记录下脉冲波与空白分别持续的时间便记录下了发送的数据内容。

由于红外遥控的协议不止NEC协议一种，因此如果将其翻译成逻辑“1”与逻辑“0”，在某种程度上来说是不够准确的，因此采用记录下一次发送过程中脉冲与空白持续的时间更具有实用价值。即上图中的9毫秒与4.5毫秒在文本文件中应为9000和4500，而紧接着的1应该被记录为560和1690，以此类推。这便是数据的记录过程。



图3-15 发送过程的软件流程图

数据发送过程是接收过程的逆过程，不同的命令需要读取不同的指令内容，并遵循一定的协议原则，按照顺序发送，便可以正确的发送控制指令，进而达到控制空调的目的。发送过程按照接收过程的介绍，应从文本文件中依次读取待发送的脉冲波持续时间以及空白持续时间并发送对应时长的脉冲波信号以及空白信号。即，若要发送图3-13所示的指令，应先发送一个持续9毫秒的脉冲波，随后发送一个持续4.5毫秒的空白信号，以此类推。

4 系统开发及测试

此次设计中以树莓派官方系统为载体，使用python作为开发语言，通过安装一些程序运行过程中必须的python库，最终完成了实物的调试工作。最终搭建完成的实物如下图所示。

图4-1 实物图

4.1 系统开发

此次系统开发的环境如下：

硬件环境：树莓派3B

操作系统：Raspbian Stretch Lite（以Debian Stretch为基础）

软件环境：Python3

依赖库：RPi.GPIO库（Python库）

开发环境：Vim

此次设计中系统安装在一张大小为32GB的存储卡中，安装系统的过程中首先需要到官方网站下载相应的镜像文件，然后使用Etcher软件将镜像文件烧写到存储卡中以启动树莓派。Etcher软件界面如下图所示。

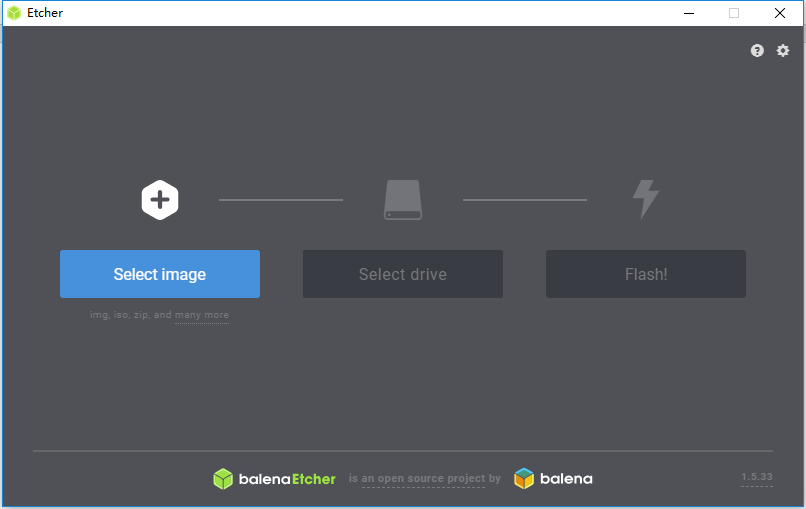


图4-2 Etcher软件界面

按照上述界面中的英文可以得出烧写的步骤，首先应先选中镜像文件（通常以.img后缀结尾），然后在select drive时选中存储卡，完成后点击Flash便可自动完成烧写过程。

系统烧写完成后，插入树莓派背面的存储卡插槽中，大约等待1分钟便可进入系统。第一次进入系统所需时间较长，后续会变快。在系统烧写完成后便可以按照第三章给出的接线图连接系统后便可以进行软件的开发。由于此次烧写的系统没有桌面，因此软件开发使用Vim为开发环境。Vim的界面如下图所示。

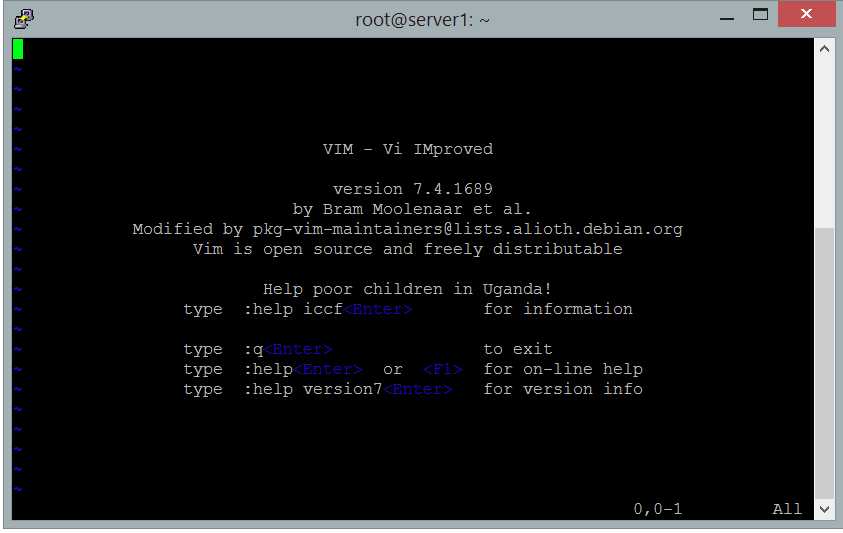


图4-3 Vim界面

4.2 系统测试

测试包括湿度数据的读取，红外数据的接收以及红外数据的发送三部分。其中湿度数据读取的结果如下图所示。

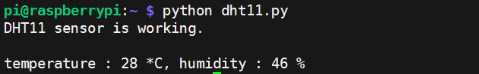


图4-4 湿度传感器测试结果

上图的显示结果表明，该系统能够正确的读取DHT11的数据。

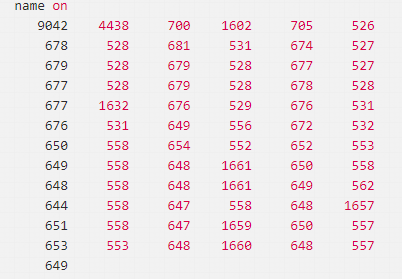


图4-4 红外数据录制测试

上图所示为红外接收模块录制数据的过程，上述结果表明，程序能够记录遥控器的键值并将结果记录于文件中。

对于红外发射模块的测试结果在此省略，因为这部分的测试需要空调作为测试的一部分，无法用图片的形式完整展示。因此，在此处不再赘述。

5 总结

此次在查阅多部文献的基础上完成了智能室内湿度检测器的设计，其可以以当前室内湿度为依据，通过判断决定是否开启空调进行除湿，且该设定值可在一定范围内进行调节。论文对设计过程及最终的效果进行了论述，此次设计所做的主要工作如下：

1. 对设计的背景及意义进行了简要的阐述；
2. 对系统的硬件选型思路以及设计方法进行了介绍；
3. 对系统的软件设计思路以及其中部分难点做了简要分析。

由于本人水平有限，同时时间也比较仓促，因此此次设计只完成了基本的功能。设计还存在较大的改进空间，例如可以加入温度控制的功能，这对于现实生活而言更加实用。除此之外，还可以利用树莓派丰富的通信接口，使得能够远程对空调进行操作。同时，树莓派还有很多的I/O引脚未使用，因此可以加入更多的家用电器，以实现更加丰富多样的控制过程。