编号：1602120130　　单位代码：14439

山东农业工程学院
SHANDONG AGRICULTURE AND ENGINEERING UNIVERSITY

本科毕业设计（论文）

题　　目：基于 STM32 单片机的物联网农业检测控制与控制系统的设计

学　　院：机械电子工程学院

专　　业：电气工程及其自动化

班　　级：1 班

学生姓名：赵国承

学生学号：1602120130

指导教师：潘莹月

完成日期：2020.05.12

山东农业工程学院制

毕业设计（论文）诚信声明书

本人提交的毕业设计（论文）《基于 STM32 单片机的物联网农业检测控制与控制系统的设计》是在指导教师指导下独立研究、写作的成果，论文中引用他人的研究成果，均已在论文中加以注释说明；指导教师和其他人员对本文提出，并被本文采纳的修改意见和建议，均已在我的谢辞中加以说明并深致谢意。

设计（论文）作者（签字）　　年　　月　　日

指导教师（签字）　　年　　月　　日

毕业设计（论文）版权使用授权书

本毕业设计（论文）《基于 STM32 单片机的物联网农业检测控制与控制系统的设计》是本人在校期间所完成学业的组成部分，是在山东农业工程学院教师的指导下完成的。本人特授权山东农业工程学院可将本毕业设计（论文）的全部或部分内容编入有关书籍、数据库保存，可采用复制、印刷、网页制作等方式将设计（论文）文本和经过编辑、批注等处理的设计（论文）文本提供给读者查阅、参考，可向有关学术部门和国家有关教育主管部门呈送复印件和电子文档。

设计（论文）作者（签字）　　年　　月　　日

指导教师（签字）　　年　　月　　日

目　录

基于 STM32 单片机的物联网农业检测控制与控制系统的设计

摘要：随着云平台、物联网以及智能手机的疾速发展，原有的农业作业方式已不能适应时代发展的需要，如何能够实现实时获取农作物生长环境参数是确保农作物产量增长的关键。

因此本文构建了一个远程农业检测控制系统，该系统借助于云平台完成了控制端对设备端的实时监测和远程控制。该系统主要由三个部分组成：手机 APP、云平台、STM32 单片机和 ESP8266 模块相结合的设备端。首先，手机 APP 作为控制终端，提供了形象的操作界面，不仅能够远程查看设备周围环境信息的动态变化，还可以控制 led 灯以及蜂鸣器的开关。其次，STM32F103C8T6 芯片作为设备端的控制终端，使用了 WIFI 通信模块进行配网，利用光强和温湿度传感器对周围环境进行探测。为了使环境信息更加方便显示，外加了 OLED 显示屏，能够在显示屏上显示温湿度以及光强等信息。

最后，本设计通过使用虚拟设备和真实设备对系统可靠性进行了测试验证，验证结果满足本次设计的目的。

关键词：物联网　云平台　农业检测　APP

Design of IoT agricultural detection control and control system based on STM32 microcontroller

Abstract: With the rapid development of cloud platform, Internet of things and smart phone, the original agricultural operation mode has not been able to meet the needs of the development of The Times. How to achieve real-time access to the environmental parameters of crop growth is the key to ensure the growth of crop output.

Therefore, a remote agricultural detection and control system is constructed in this paper, which realizes real-time monitoring and remote control from the control end to the equipment end by means of the cloud platform. The system consists of three parts: mobile APP, cloud platform, STM32 SCM and ESP8266 module. First of all, as the control terminal, mobile APP provides an image operating interface, which can not only remotely view the dynamic changes of the surrounding environment information of the device, but also control the switch of led lights and buzzer. Secondly, STM32F103C8T6 chip is used as a control terminal of the device, and WIFI communication module is used to distribute the network, and light intensity and temperature and humidity sensors are used to detect the surrounding environment. To make it easier to display environmental information, an OLED display is added to display temperature, humidity and light intensity.

Finally, the design uses virtual equipment and real equipment to test and verify the reliability of the system, and the verification results meet the purpose of the design.

Key words: Internet of things；cloud platform；Agriculture detection；APP

1.绪论

1.1 选题背景和意义

1.1.1 选题背景

近年来，由于国内外环境发生了很大变化以及我国农业经济的飞快发展，中央政府对“十二五”期间农业经济的发展做了更加高的要求。随着产业结构的不断调整以及城市规模的扩大和人口的不断增加，必然会促进农产品产量需求的急剧增长，这对农产品质量的安全提出了更加高的要求。所以要加快转换农业发展方式，加速农业现代化的步伐。

对于传统的农业监测，都是需要人工去察看农田信息。在此过程中所获的信息基本上都是靠农户以往的经验来确定农作物的实际情况，这种方法一是费时费力，需要人工去农田查看信息，占用农户完成其他农活的时间；二是得到的信息不够准确，只能根据以往的经验进行大概估测农田的情况[1]。所以通过物联网来远程进行农业监测与控制就显得很重要，它很好的解决了费时费力以及获得信息不准确的问题，它对于农作物质量的保证起到了十分关键的作用。

1.1.2 选题意义

利用现代化技术改造和装备传统农业对农业达到现代化水平起着重要作用。物联网技术是当代信息技术的新生力量，是推动农业现代化的重要切入点[2]。本设计将传统的农业监测方式进行了改变，运用了基于 STM32 单片机与物联网远程通信的方式，可对目标监测农田发挥精确感知、速度反馈、远程控制、提升决策水平的作用。对于实现农业监测与控制、提高农户收入具有不可或缺的重要意义。

1.2 农业物联网的国内外研究现状

1.2.1 农业物联网的国外研究现状

近些年来，许多发达国家在农业生产领域进行了一系列物联网技术的研究，取得了许多优秀成果，完成了物联网技术与农业生产、流通等领域的关联实践，促进了物联网以及其附属产业的发展。例如，许多美国农场主利用当下的物联网技术，用于预防和控制自然灾害和害虫，并取得了良好的效果。在以色列，气候炎热，土地资源稀缺，但其发达的物联网农业技术提高了生产效率。其中，完整的农业服务体系和先进的农业技术是以色列在物联网农业发展方面的无与伦比的优势[3]。

1.2.2 农业物联网的国内研究现状

在国内，中国大部分的农村地区仍然使用比较原始的体力劳动从事农业工作，生产方式十分落后，生产效率也很低。人们在农业生产当中获得有关农作物目前生长环境信息有限，主要获取的方法是利用人力活动。虽然近年来，我们国家通过在农业上运用物联网技术取得了相当不错的成果，但是与国外相比各方面还相对比较落后，总体还处于比较低的水平。虽然已经有能够完全实现农业监测的产品被研发成功，例如我国研制的地面监测台与遥感技术相结合的监测系统等，但其应用范围、控制精度较国际的先进水平还是有一定的差距。实现物联网农业系统在农业领域的大范围应用仍还需要继续努力[4]。

1.3 设计研究内容

根据当前我国农业自动化的程度和现实需求，本设计能够使农业监测控制水平实现高度的自动控制。即通过使用温湿度及光强传感器将数据反馈给 STM32，STM32 通过无线通信模块与云端进行通信，云端将数据下发到 APP，进而实现检测控制的目的[5]。本设计主要的研究内容如下：

（1）系统概念层次设计：进行功能分析，对检测控制系统的构成、预计要实现的功能进行简单的规划设计。

（2）光强及温湿度传感器的选择：对各种适合测量光强及温度和湿度传感器进行分析，最终选择各方面相对性能较好的 DHT11 和 GY-30 传感器。

（3）硬件方面设计：对系统主控制器的选择、元件的选择、在 Altium Designer 当中对其硬件电路进行设计。

（4）软件方面设计：利用 Keil uVision5 作为开发环境对系统设备端程序进行编写，利用 Android Studio 作为开发环境编写 APP 端软件程序。

（5）检测与调试：在条件允许的范围内对本系统进行调试与检测，对其传输的数据准确值进行评估，分析误差的来源，寻找能够优化测量数据的方案。

2. 总体设计

2.1 研究方法及技术路线

2.1.1 研究方法

（1）经验总结法：通过对农业监测控制系统中的实际问题进行分析与总结进而使之能够系统化。

（2）文献资料法：采用文献资料法分析农业监测控制系统的研究背景及意义、国内外应用研究现状。

（3）归纳总结法：使用归纳总结法对监测控制内容总结归纳，并及时改进设计的不足。

2.1.2 技术路线

本设计的技术路线如图 2-1 所示，通过技术路线图可以看出，通过前期的调研分析，确定了设计的总体方案，进而分析计算确定各类参数，最后确定了可执行的操作运行方案。

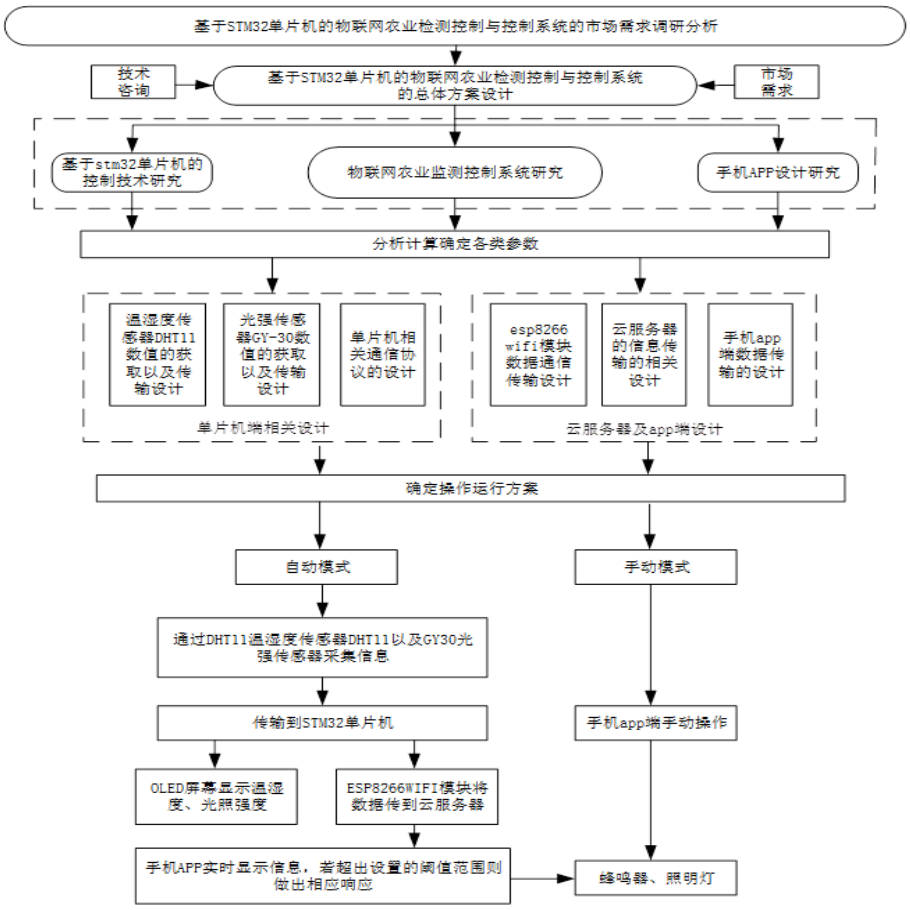


图 2-1 技术路线图

2.2 系统的组成

本设计主要由以下几个部分组成：数据采集端、单片机检测平台、显示部分、ESP8266 数据传输部分、云平台以及手机 APP 实时显示控制部分。具体如图 2-2 所示。

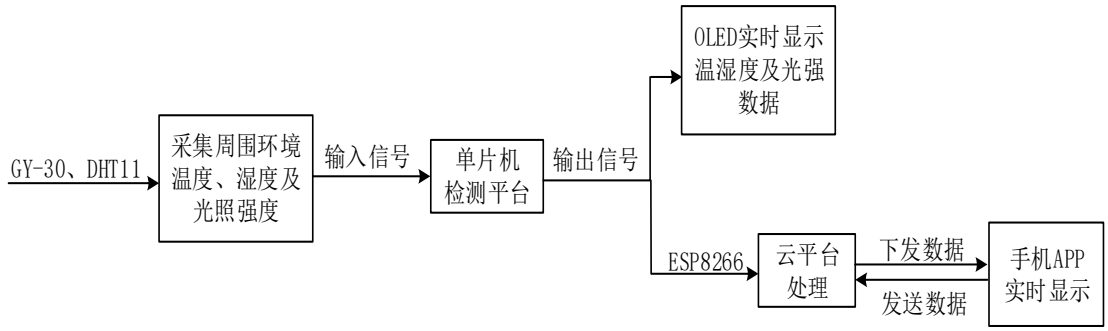


图 2-2 系统装置框架图

3.系统硬件的设计

本文设计的农业检测控制系统，将准确的采集农作物周围的环境信息作为主要目标，为科学研究方面提供一些参考，进而实现农业的精准操作[6]其硬件总体框图如图 3-1 所示。

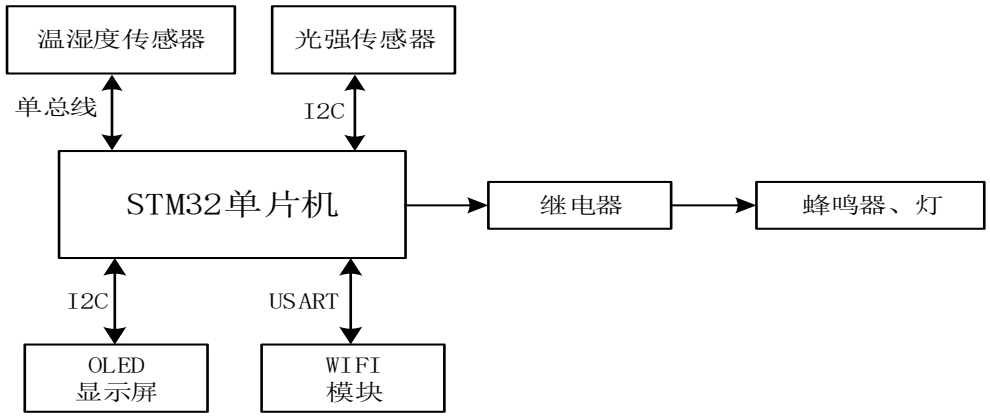


图 3-1 系统硬件总体框图

3.1 单片机

3.1.1 单片机选型

将运算器、存储器、控制器和各种 I/O 接口等集成在一块芯片上,就能得到一个微型计算机,其在功能和组成上已经具备了计算机系统的特点,因此被称为单片微型计算机简称单片机[7]。当前常用的有 PIC、51、STC、AVR、Freescale、MSP430、STM32 单片机被广泛使用。其比较信息见表 3-1。

表 3-1 主流单片机信息比较

名称	代表产品	特点	应用范围
PIC	PIC16F873	低工作电压、低功耗	工业场合应用
51	8051、80C51	操作简单、价格低	性能要求不高场合
STC	STC12C2052AD	高速、超强抗干扰	性能要求不高场合
AVR	ATUC64L3U	高性能、高速度	医疗设备、GPS 等
Freescale	MC9S12G 系列	低成本，高性能	主要应用汽车领域
MSP430	MSP430F 系列	低功耗、速度快	低功耗工业场合应用多
STM32	STM32F1 系列	低功耗、高性能、低成本、高性价比	较其他单片机而言，应用极广泛

从表 3-1 不难看出，七款主流的单片机不管是性能还是应用领域都有所或多或少的差异，但是 STM32 系列微控制器，具有很高的性价比和丰富的信息可以参考，大大节省了开发成本。因此选择 STM32 系列单片机。

3.1.2STM32F103C8T6 芯片简介

本次设计中选用的是 STM32F103C8T6 作为系统主控芯片，其引脚分布如图 3-2 所示。STM32F103C8T6 作为一款 32 位的单片机其内核是 ARM Cortex-M3，其程序内存容量为 64KB，故可以充分满足本设计的所有需求。具体的硬件设计图见附录 A。

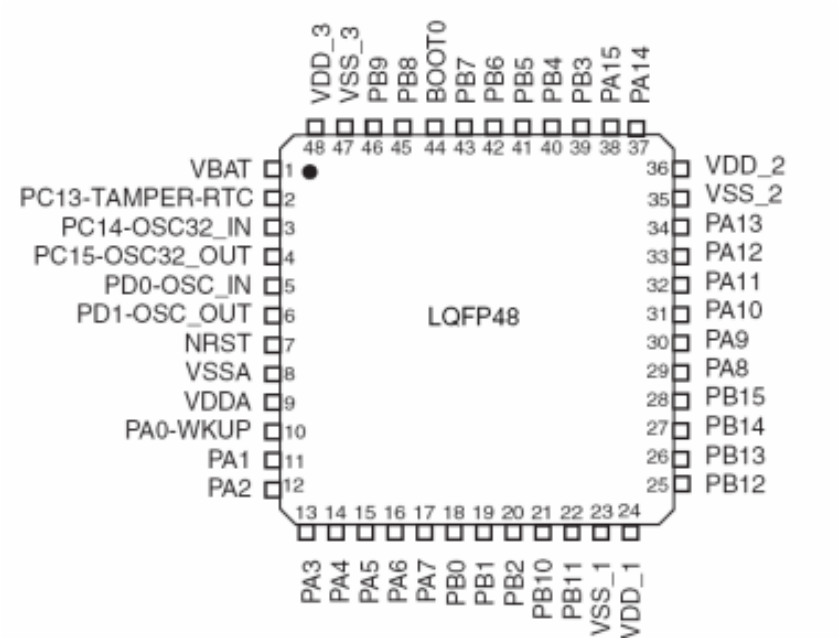


图 3-2 STM32F103C8T6 引脚分布

3.2 温度传感器

3.2.1DHT11 温湿度传感器简介

DHT11 作为复合的传感器能够将标定的数字信号进行输出，其稳定性和可靠性是非常高的，测量湿度范围为 20%～90%，测量温度范围为 0～50℃[8]。

其包括一个测温元件和电阻式感湿元件，同时与一个具有高性能的 MCU 连接。内部结构图如图 3-3 所示。其中 NTC 元件用于采集温度信息，另一个元件用于采集湿度信息。DHT11 具有一块可编程的存储器 OTP，用于存储校准后的系数。经过测温和感湿元件检测后，与 OTP 内含有校准系数比较，得到准确的检测值[9]。

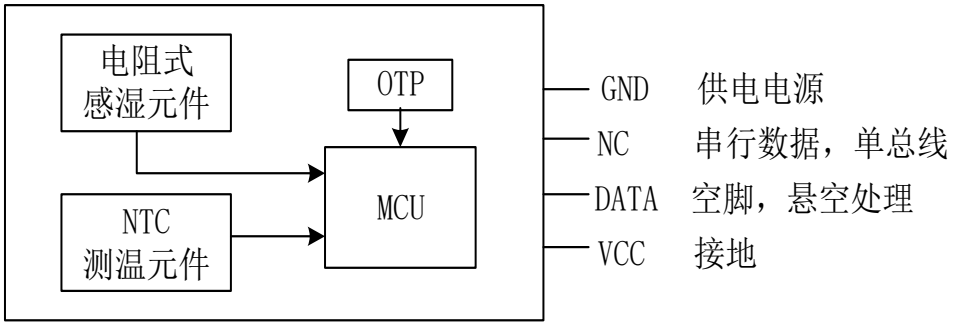


图 3-3 DHT11 内部结构图和引脚说明

3.2.2DHT11 温湿度传感器硬件电路设计

如图 3-4 所示，DHT11 外接 3.3V 电源，通过 PB9 引脚在 4.7k 上拉电阻的作用下与单片机相连。

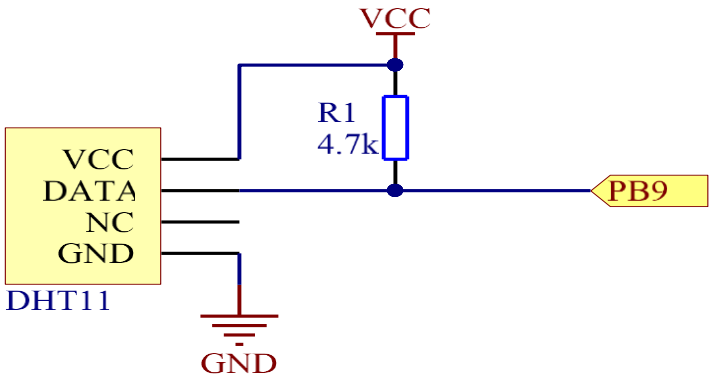


图 3-4 DHT11 硬件连接图

3.3 光强传感器

3.3.1 光强传感器简介

GY-30 内部采用的芯片是 BH1750FVI，其芯片的通信方式是两线 IIC 总线通信，一条对数据传输处理，另一条用于对模块的时钟控制，通过脉冲和时钟变化决定数据的接收和应答[10]。利用其高分辨率能够探测光强度在 1 到 655351x 范围之间，由于其检测范围较大，故可以满足本次设计的需要，其内部结构图如图 3-5 所示。

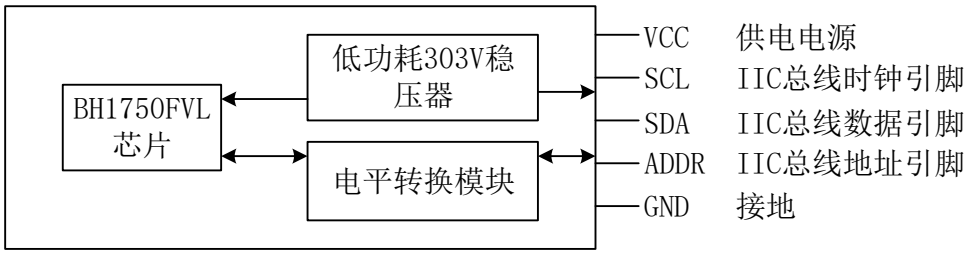


图 3-5 GY-30 内部结构图和引脚说明

3.3.2 光强传感器硬件电路设计

如图 3-6 所示，该传感器的外接 3.3V 电源，通过 PB10 和 PB11 管脚在 10k 上拉电阻的作用下与单片机相连。

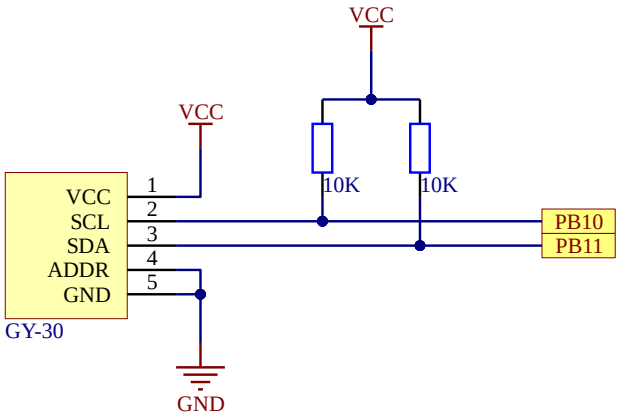


图 3-6 GY-30 硬件连接图

3.4 无线通信模块

3.4.1 无线通信模块选型

根据本次设计的要求，选择的通信模块必须具有以下两个基本功能：一个是实现接收 STM32 通过串口发送的数据，一个是实现通过路由器发送接收的数据到云平台，综合考虑以后选择的是当前比较流行的高性能 UART-WIFI 模块—ESP8266 模块[11]。

ESP8266 模块不仅是一个独立的模块,而且具有完善的 WIFI 网络处理方案，可以独立运行，也能作为从机运行。ESP8266 模块采用 MCU 和串口通信，并有内置的协议能够支持串口和 WIFI 之间进行转换[12]。

除此之外，ESP8266 模块价格便宜、集合程度高而且应用非常广泛，而且 ESP8266 有丰富的资料，可以加快研发周期，降低研发费用[13]。

3.4.2ESP8266 硬件电路设计

如图 3-7 所示，ESP8266 模块的 TXD 和 RXD 引脚分别和单片机的 PA10 和 PA9 引脚相连接。

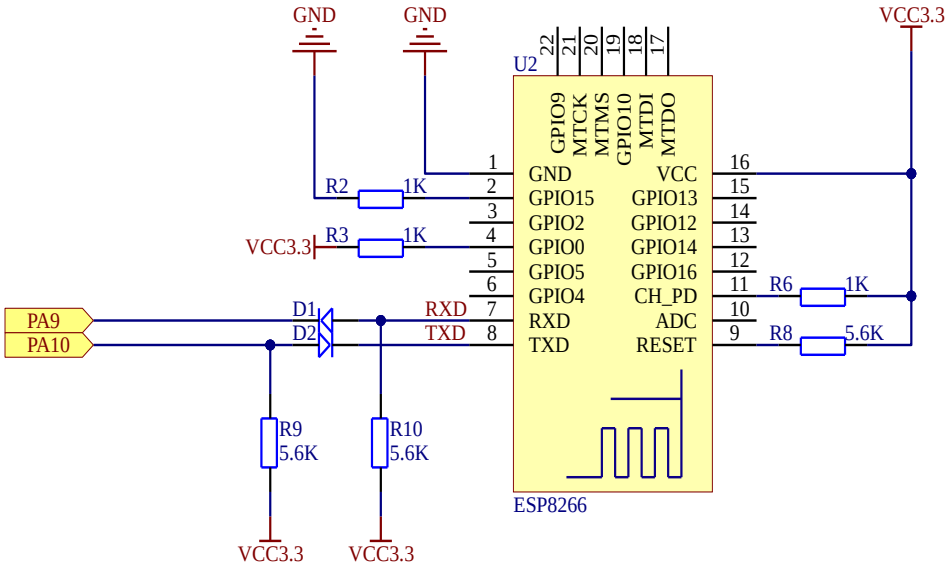


图 3-7 ESP8266 硬件连接图

3.5 显示屏

3.5.1 显示屏选型

目前比较常见的三种显示屏分别是液晶屏 LCD、OLED 显示屏以及点阵屏 LED，各自特点如下：

（1）LCD 液晶屏，由于像素高、亮度好、对比度高，且功耗低等特点，被广泛的使用到现在的电子设备中。

（2）LED 显示器是平板显示器，用于显示图像、文本、视频等各种信息。LED 灯体积较大，因此设计的显示屏也很大。

（3）OLED 屏反应快，不需背光源、具备自发光、较大温度使用范围等特性，被称作为下一代显示屏的主流技术。

综上可以看出，OLED 显示屏各项性能更为突出，故选择 OLED 显示屏。

3.5.20LED 硬件电路设计

如图 3-8 所示，OLED 具有四个引脚，分别是 GND、VCC、SDA、SCL，且 GCC 需要接地，VCC 接电源（3-5V），SCL 接 STM32 单片机的 PB6 引脚，SDA 接 STM32 单片机的 PB7 引脚。

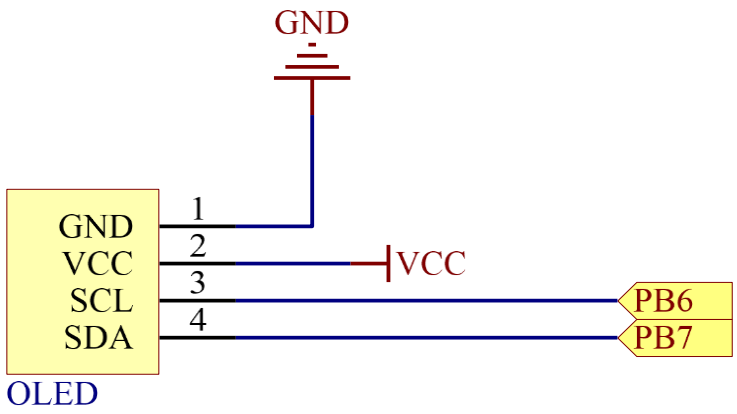


图 3-8 OLED 硬件电路

3.6 继电器

为了实现对本设计中控制蜂鸣器和灯开关的目的，故在设备端加入两个一路继电器设备。继电器能够实现电路开与关以及安全防护等作用[14]。其利用了电磁感应原理，通过通电给小电流回路，使内部带铁芯的导线圈能够产生磁场，进而磁场能够吸附继电器开关触点，实现对大电流回路控制，具体电路连接方法如下。

以控制灯为例，将继电器的 DC+与 5V 直流电源的正极相连，COM 接地，IN 与 STM32 的 PA2 引脚相连，将继电器的输出端 NC 与灯的负极相连，灯另一端接电源。由于本设计采用的是接常闭端的接法，且 NO 端为继电器的常开端口，故悬空处理。当单片机输入高电平时，继电器 NC 端与 COM 口短接，灯亮。继电器控制电路图如图 3-9 所示。

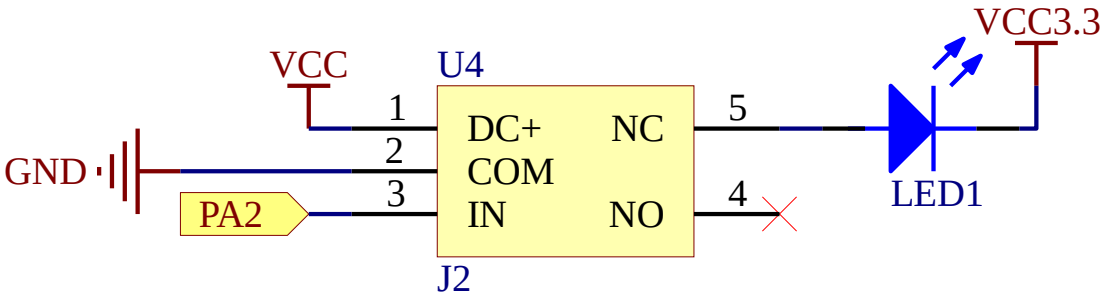


图 3-9 继电器控制电路

4.系统软件的设计

为了配合系统硬件电路的工作，本系统还需设计出相关的软件部分，程序见附录 B。软件设计分为 GY-30 光强的采集、DHT11 温湿度的采集、OLED 液晶显示、ESP8266 模块的通信、单片机控制系统。

本系统软件运行流程图如图 4-1 所示，首先进行系统的初始化等操作，然后光强传感器及温湿度传感器开始进行采集信息。采集到的信息会发送给单片机，其接收到数据后分别将数据发送给 OLED 和 ESP8266。而 ESP8266 连接路由器后进一步发送数据到云平台，云平台会将数据传输至远程控制端，控制端将收到的内容进行分析比较，超出阈值则设备端进行报警。

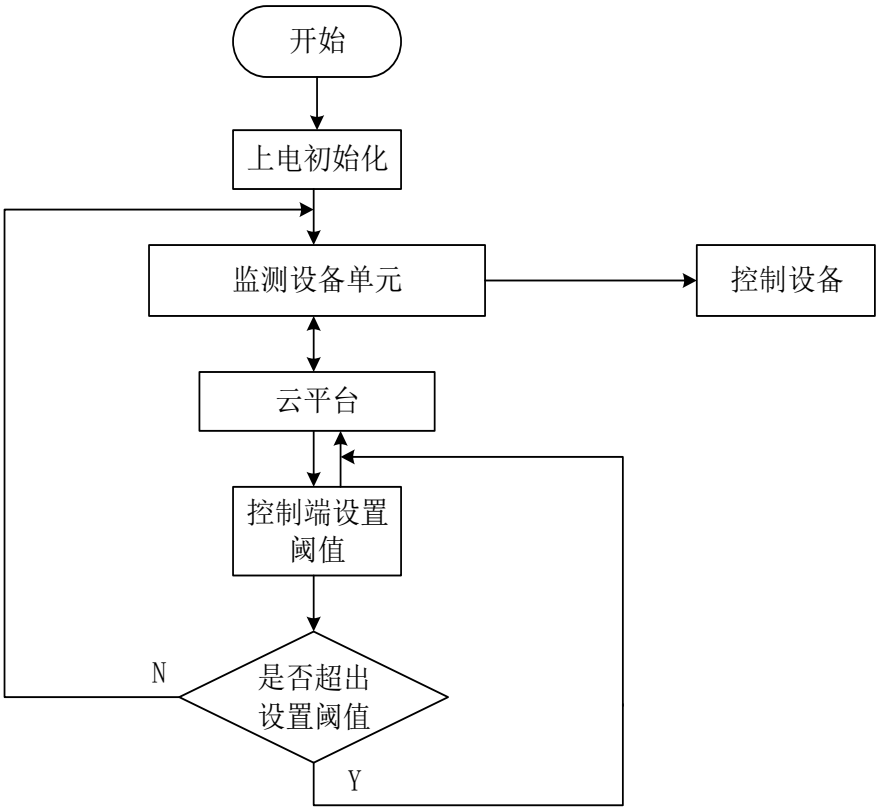


图 4-1 软件运行流程图

4.1 系统软件开发环境

4.1.1STM32 单片机开发环境

本检测控制系统中各传感器的开发工具使用的是 KEIL 软件。Keil uVision5 开发软件是目前 STM32 系列单片机开发的主流工具。该软件能够实现 C 语言的下载、仿真调试和编译等一系列的开发功能，是开发 STM32 系列单片机最实用的开

发工具[15]。

4.1.2APP 开发环境

系统控制端的开发环境使用的是 Android Studio,是一款非常综合的 JAVA 编程软件,被许多行业专家称为目前最好的开发环境之一,尤其在自动代码提示、代码智能助手、重构等功能是非常便利的[16]。

4.2 云平台的应用

云平台作为本文设计的物联网农业检测控制系统的重要枢纽,发挥了信息传输的作用，是本设计系统信息传输的关键点。由于自主完全设计云平台开发周期长、难度大，因此本设计选择了机智云云平台进行本系统的配置开发。

4.2.1 机智云平台简介

机智云作为一个云平台，不仅能够为使用者们提供十分便利的云服务，而且还能够在很大程度上减少物联网开发存在的一些技术门槛，进而降低研发费用。其主要包括云端、设备端和 SDK 三部分，如图 4-2 所示。

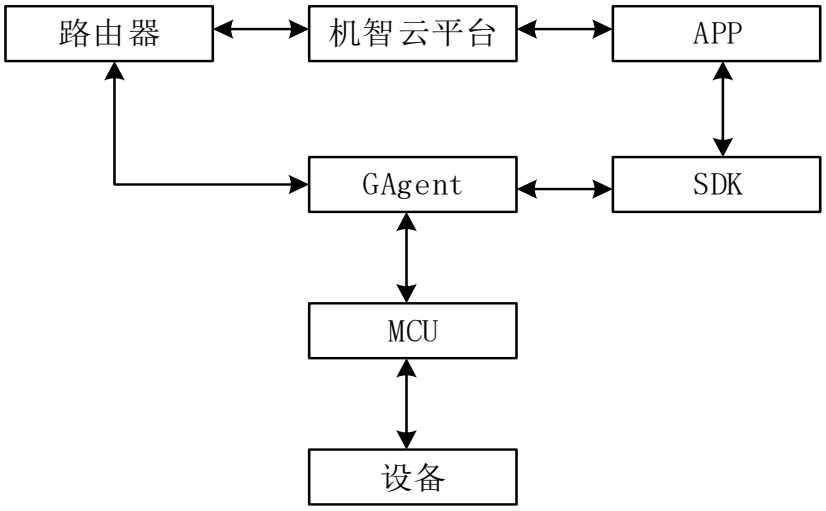


图 4-2 机智云平台的基本构造

4.2.2 设备接入云平台开发

物联网农业检测控制系统设备端与手机控制端通信的前提是能够接入到云平台，本文使用了 ESP8266 模块接入到云平台的方案，使用机智云平台提供的通信固件。同时只有将机智云的固件刷入到 WIFI 模块中，设备才能够正常接入机智云平台。使用机智云固件将农业检测控制与控制系统接入到机智云平台并与手机控制端建立通信，步骤如下：

（1）设备上电。

（2）配置农业检测控制系统入网。

（3）APP 绑定农业检测控制与控制系统。

在设备配网完成后，APP 端不管是在广域网，还是与设备在同一个局域网但是无法连接网络时，都可以通过云端来控制设备。

4.3 主要功能模块程序设计

4.3.1 温湿度传感器模块程序

DHT11 传感器在本农业检测控制与控制系统设计中起到采集温湿度信息的作用，采用的是单总线数据格式[17]，传输时序如图 4-3 所示。

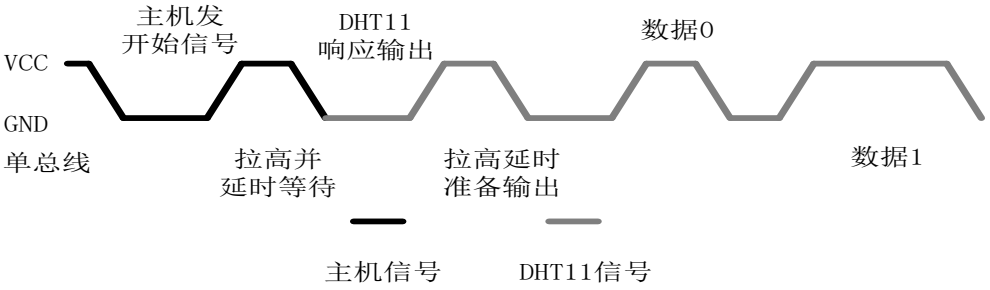


图 4-3 DHT11 数据时序图

MCU 发送起始信号时，MCU 至少需要拉低 18ms，之后拉高等待读 DHT11 发送的响应信号，若读取到总线上变为低电平，则 DHT11 开始传响应信号，发送完毕后，拉高总线[18]。其中部分代码示例如图 4-4 所示，完整代码见附录 B。

```
//在oled和app上显示温湿度值
DHT11_Read_Data(&temp,&hum);//读取 DHT11 传感器
OLED_ShowNum(48,0,temp/10%10,1,16);//x,y，第三位是数据位，长度，字体大小
OLED_ShowNum(56,0,temp%10,1,16);
OLED_ShowNum(48,2,hum/10%10,1,16);
OLED_ShowNum(56,2,hum%10,1,16);//x,y，第三位是数据位，长度，字体大小
//上传温湿度数据
currentDataPoint.valueTemp = temp ;
currentDataPoint.valueShidu = hum;
```

图 4-4 DHT11 代码示例

4.3.2 光强传感器模块程序

GY-30 光强传感器在本农业检测控制与控制系统设计中起到采集周围光照强度的作用，与 STM32 单片机之间以标准 IIC 总线协议进行连接通讯。如图 4-5 为 GY-30 的 IIC 时序图。

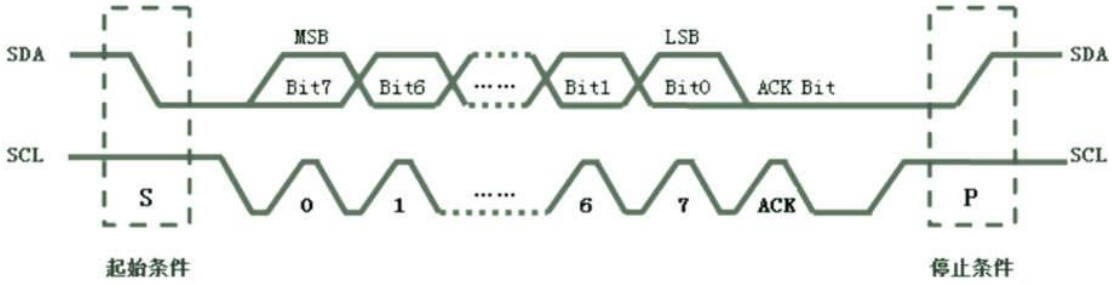


图 4-5 GY-30 的 IIC 时序图

当发起始信号时，数据传输开始，时钟线 SCL 上每发一个脉冲，都会在 SDA 上传一位数据，其数据从高往低传输。在 SCL 处于高电平的期间，SDA 传输数据有效。通常在 SCL 时钟线变低时，SDA 会进行电平切换。字节传送完成后，后面会有应答位，当不需要数据继续传输时，这时主机便产生终止信号。根据其工作原理便可以写出控制 GY-30 的程序，其中部分代码示例如图 4-6 所示，完整代码见附录 B。

```
GQ_Data = BH1750_ReadContinuous1(); //读取GY-30传感器
OLED_ShowNum(48,4,GQ_Data/10000%10,1,16);
OLED_ShowNum(56,4,GQ_Data/1000%10,1,16);
OLED_ShowNum(64,4,GQ_Data/100%10,1,16);
OLED_ShowNum(72,4,GQ_Data/10%10,1,16);
OLED_ShowNum(80,4,GQ_Data%10,1,16);
currentDataPoint.valueGQ = GQ_Data;//上传光强数值
```

图 4-6 GY-30 代码示例

4.3.30LED 显示屏模块程序

OLED 显示屏和 GY-30 传感器都是用的 IIC 通信协议，其具体通信原理就不在赘述。OLED 模块的控制器是 SSD1306，驱动 OLED 显示器件[19],要使 OLED 可以正常使用，可以根据下面这三步执行：先初始化 OLED 的相关 IO 引脚，然后将模块进行初始化，在向 SSD1306 写值，将内容在 OLED 屏幕上显示。其工作流程如图 4-7 所示。

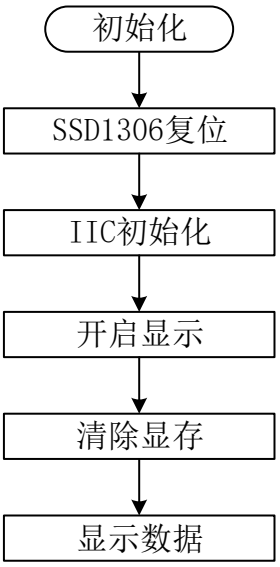


图 4-7 OLED 工作流程图

此外，因为 OLED 不能自动识别出的汉字，所以还需要借助取模软件取模，取出的字模可以在 OLED 显示屏上正常显示。其中部分代码示例如图 4-8 所示。

```
void OLED_Init(void)
{

    GPIO_InitTypeDef GPIO_InitStructer;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

    GPIO_InitStructer.GPIO_Pin=GPIO_Pin_6 | GPIO_Pin_7; //6--SCL   7--SDA
    GPIO_InitStructer.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_InitStructer.GPIO_Mode=GPIO_Mode_Out_PP;

    GPIO_Init(GPIOB, &GPIO_InitStructer);
```

图 4-8 OLED 代码示例

4.3.4WIFI 通信模块程序

ESP8266 WIFI 模块可以完成用户的串口数据与无线通信网络之间的交换[20]。因此如果只是想驱动 WIFI 模块工作，只需要完成对串口的初始化即可。串口通信初始化的步骤如下：

（1）对 GPIO、串口进行使能；

（2）对串口进行复位操作；

（3）对 GPIO 相关模式进行设置；

（4）对串口相关参数进行初始化操作；

（5）对 NVIC 初始化、启动中断；

（6）对中断相关的处理函数进行编写。

这样 WIFI 便能够正常的和 MCU 通信了，其中部分代码示例如图 4-9 所示，完整代码见附录 B。

```
for(i=0; i<info->num; i++)
{
    switch(info->event[i])
    {
    case EVENT_LED:
        currentDataPoint.valueLED = dataPointPtr->valueLED;
        GIZWITS_LOG("Evt: EVENT_LED %d \n", currentDataPoint.valueLED);
        if(0x01 == currentDataPoint.valueLED)
        {
            if(1 == currentDataPoint.valueLED1)
                LED0 = 0;
        }
        else
        {
            if(1 == currentDataPoint.valueLED1)
                LED0 = 1;
        }
        break;
    case EVENT_BEEP:
```

图 4-9 WIFI 模块代码示例

4.4 手机 APP 设计

为了实现 APP 快速接入到云平台，本设计采用机智云平台提供的 APP 开源框架。因此 APP 开发的主要内容是在其框架之上完成一些具有本设计特色的功能，进而实现 APP 快速开发[21]。总体工作流程如图 4-10 所示。

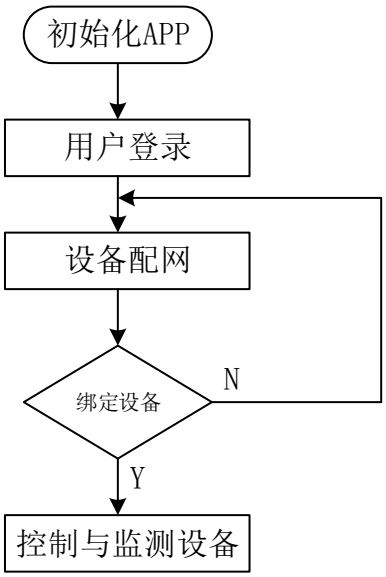


图 4-10 APP 总体工作流程图

对于本设计而言，此 APP 的主要功能需求是可以远程控制监视产品设备。当在手机中打开 APP 后，首先进入登录界面，如图 4-11 所示。这里可以选择注册或者直接跳过就可以。

图 4-11 用户登录界面

然后进入监测控制页面 UI 如图 4-12 所示，监测控制界面使用了 EditText 控件、ImageView 控件、Switch 控件、SeekBar 控件等控件。其在工程 XML 的文件里面可以对控件属性进行设置，比如控件的文字尺寸、颜色、宽度和高度以及控件间布局等。本文使用了两种较常用的相对布局和线性布局。

图 4-12 监测控制界面

本设计的监测控制界面的作用是清晰的展示给用户可以使用的主要功能。在该界面用户可以看到温湿度以及光照强度的数值，可以在范围内自由设置温湿度

以及光照强度的阈值，同时可以控制蜂鸣器和灯。

对于一个 APP 而言往往少不了全局配置文件，其主要代码如图 4-13 所示。

```
"gizwitsInfo": {
    "appId": "3e41dd99dd974df482bbcf6a736801ba",
    "appSecret": "ca0ab8605109446a82b3ba4f8cbef101"
},
"productInfo": [{
                "productKey": "50dfd048833c4283ba509979fa559a27",
                "productSecret": "7b45a7c0de2940598f8e74c2903e9f49"
                }],
```

图 4-13 全局配置文件

根据图 4-15 可以看到 appId 是应用的标识码，appSecert 是应用的密匙，与 appId 一起生成。productKey 是产品唯一标识码，其产品密码 productSecret 和 productKey 有同样功能，也起着产品身份识别作用[22]。

5. 系统测试与结果分析

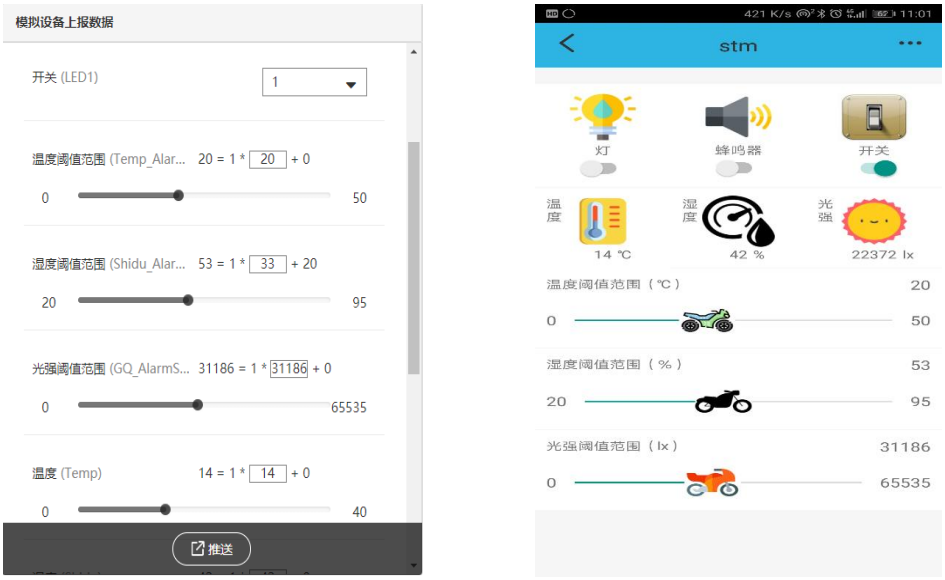
5.1 虚拟测试

如图 5-1 所示是机智云平台提供的在线调试界面，通过虚拟的设备对 APP 进行调试[23]。

首先进入机智云的开发者中心，启动相应的虚拟设备，扫描对应的二维码，扫描成功后，在其通信日志窗口中可以发现设备上线完成。

图 5-1 虚拟设备测试界面

然后开始测试 APP 端与虚拟设备之间的通信：在云平台虚拟设备界面中，填入要测试数据点的数据，之后使用“推送”向手机 APP 端下发数据。其中虚拟设备使用的各测试值如图 5-2（a）所示，APP 端接收到的数据值如图 5-2（b）所示。



(a)虚拟设备下发数据　　(b)APP 端接收数据

图 5-2 APP 与虚拟设备通信测试

然后在使用 APP 向虚拟设备上传数据，查看虚拟设备端的接收数值是否与 APP 端一致。经过大量的测试以后，发现手机 APP 接收到的数据和虚拟设备下发的数据完全一致，故符合本设计要求。

5.2 实物测试

虚拟设备测试后，能够确定云平台和手机 APP 端能够正常通信。在此基础上验证系统功能能否达到预期要求，进而保证农业检测控制系统的实用性和稳定性。测试步骤如下：

（1）在不同的网络下测试 APP 和农业检测控制系统设备端的通信。将手机分别的接入到移动网络和局域网下看是否能够和设备进行正常的通信。

（2）测试控制系统的通信成功率以及稳定性。包括设备端正确接收到 APP 端的数据成功率、手机 APP 端准确接收到来自设备端的数据成功率以及设备断网重新连接成功率。

（3）APP 的功能测试。包括测试检测控制单元的功能，以及能否实时的显示设备端发送的数据，能否控制继电器的动作，能否准确设置传感器的阈值等。

5.3 测试结果和分析

（1）如图 5-3 所示，当农业检测控制系统处于手动模式时，手机 APP 端能够控制继电器进而控制灯和蜂鸣器，且 OLED 上能够正常显示温湿度以及光照强度数据。在自动模式下，若数据超过设定阈值则会自动发出警报，OLED 屏幕上会

闪烁警告图标，设备蜂鸣器发出声音并且灯亮。

图 5-3 OLED 屏幕显示

（2）当 APP 和设备接入相同的局域网时，即使所连接的路由器没网也不影响 APP 对设备的控制；当设备和 APP 不属于一个局域网时，手机使用 WIFI 或者移动网络连接互联网，此时手机可以实现对设备的远程监视控制，但是若设备所连接的路由器没有网络则不可以进行控制[24]。

（3）本设计对农业检测控制系统进行了三个方面的通信测试，经过了许多次的测试和统计之后得到的数据可由表 5-1 内容所示，由此可以得出系统具有比较高的数据通信稳定性。

表 5-1 系统通信测试表

测试方法	测试数	成功率（%）
手机收到数据	200	99.5
设备断网自连	50	98.5
设备收到数据	150	90.8

（4）APP 控制界面。此时设备端已经将数据上传到 APP，并且不断实时变化。如图 5-4 所示，在 APP 界面显示了已绑定设备当前上报的温湿度以及光照强度数据，手机端可以在阈值范围内自由的设置阈值，并且若传感器探测数据超过设定阈值则会自动发出警报，此时继电器会控制灯闪烁和蜂鸣器响。


526 B/s
21:35
stm
灯
蜂鸣器
开关
温度
11 ℃
湿度
83 %
光强
14 lx
温度阈值范围（℃）
36
0
50
湿度阈值范围（%）
86
20
95
光强阈值范围（lx）
23995
0
65535

图 5-4 APP 监控界面

6. 总结与展望

6.1 主要设计总结

本文构造了一款基于 STM32 单片机的物联网农业检测控制系统。其中硬件平台是以 STM32F103C8T6 作为核心处理器，结合现代物联网技术以及传感器方面的技术，能够远程查看温湿度、光照强度的变化，又通过 ESP8266 WIFI 模块接入了云平台，实现了可在手机 APP 侧查看设备当前状态和远程控制设备的目的。主要设计成果包括：

（1）分析了物联网农业检测控制系统相关方面的近年来国内外的发展现状，给出了物联网农业检测控制系统的技术路线。

（2）通过分析结合本设计需要实现的功能，确定了物联网农业检测控制系统的基本结构、核心元器件的选型等。

（3）选用 DHT11 传感器和 GY-30 传感器能够很好的探测周围环境温湿度及光照强度的变化，并能够在 OLED 上显示出来，方便观察。

（4）实现了 STM32 单片机利用 WIFI 模块通过云平台与手机 APP 的通信，能够比较稳定的将数据进行传输。

（5）对设计系统进行了实验测试，证明此设计具有一定的实用性。

6.2 系统特点

（1）系统性价比高

在充分考虑能够完成设计任务要求的前提下，本设计尽可能选择性价比高的器件，比如本设计选择的主控器 STM32F103C8T6 就具有很高的性价比，且能够很好的完成设计需求。

（2）系统传输速度快

本设计的农业检测控制系统，能够将传感器检测到数据实时的在 OLED 和手机 APP 上显示，传输速度快。

（3）通信无距离限制

本设计在理论上说，只要手机和设备端能够正常连接网络，不论距离多远，手机端也能够实现对设备端的检测控制。

6.3 展望

物联网农业检测控制系统虽然已经设计完成，并经过一系列验证测试，证明

其实现了远程检测控制的能力，但仍然还存在一些问题有待更进一步的研究：

（1）本设计的功能还不够完善。对农业环境只是集中于光照强度、温度、湿度上的检测，这使设备获取的环境信息有限，从而不能够为使用者提供更加全面的参考信息。

（2）本设计虽然能够检测农业环境温湿度以及光照强度，但是并未提出相关的环境调节控制方法，未来研究需要在调控环境策略方面做更加深入的研究。

（3）由于时间的原因，只完成了对安卓操作系统的 APP 设计，在未来可以完成对 IOS 操作系统的 APP 以及微信端小程序的设计。

谢辞

寒来暑往，斗转星移，转眼间大学学习生活已经悄然结束。回首大学四年，从懵懵懂懂到变得逐渐成熟，期间经历了太多的事情，在这里我留下了自己最宝贵的青春年华，带走了不仅是美好的回忆还有各位老师对我的谆谆教诲。

首先，我要感谢我的指导老师潘莹月老师，在我完成毕业设计时遇见的问题，潘老师都能耐心及时的帮我解答，正因如此，我的毕业设计才能顺利的完成。

在此，我向尊敬的潘老师表示最衷心的感谢！

然后我要感谢大学四年的同学们，感谢你们在我学习生活中给我提供的许多帮助，尤其是我的室友们，感谢你们在宿舍生活中的包容和陪伴。

感谢我的父母，在我的求学生涯中，是你们给了我全力的支持，你们的殷殷期望是我进步的动力。

最后，对我论文中所引用的文章作者一并致以诚挚的感谢。

参考文献

[1]陈韵秋,李峥.基于 STM32 和 Android 系统的智能农业大棚设计[J].淮北师范大学学报(自然科学版),2019,40(01):43-48.

[2]李国刚,李旭文,温香彩.物联网技术发展与环境自动监控系统建设[J].中国环境监测,2011,27(1):5-10.

[3]豁保强. 智能大棚监测与控制关键技术的研究[D].天津科技大学,2014.

[4]Hongwei Zhu. Research on Agricultural Internet of Things Information Collection and Control System[P]. Proceedings of the 2018 3rd International Workshop on Materials Engineering and Computer Sciences (IWMECS 2018),2018.

[5]Zhuang Miao. Research on Intelligent Agriculture Monitoring System Based on Internet of Things[P]. Proceedings of the 2018 3rd International Workshop on Materials Engineering and Computer Sciences (IWMECS 2018),2018.

[6]王冬. 基于物联网的智能农业监测系统的设计与实现[D].大连理工大学,2013.

[7]韩毓. 基于单片机的蔬菜大棚温度控制系统[D].中国海洋大学,2010.

[8]廖建尚.基于物联网的温室大棚环境监控系统设计方法[J].农业工程学报,2016,32(11):233-243.

[9]陈建新.DHT11 数字温湿度传感器在温室控制系统中的应用[J].山东工业技术,2016(18):120.

[10]邹曙光. 基于 Android 的嵌入式农业环境采集系统设计与实现[D].江西农业大学,2016.

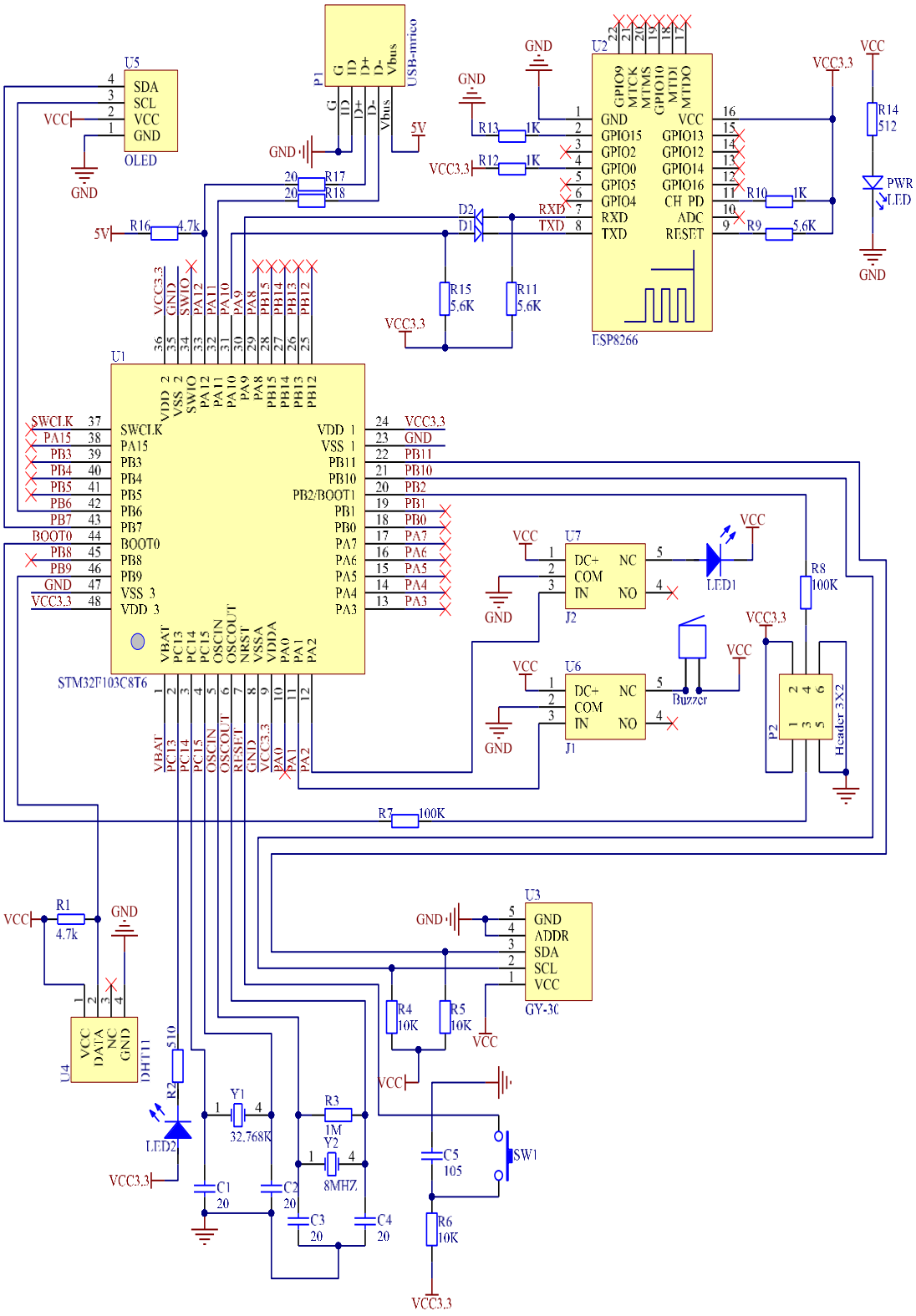
[11]范兴隆.ESP8266 在智能家居监控系统中的应用[J].单片机与嵌入式系统应用,2016,16(09):52-56.

[12]王小娟.基于 ESP8266 无线传输的温湿度检测仪设计[J].九江职业技术学院学报,2017(04):22-24+32.

[13]刘振. 基于 STM32 智能家居的无线网关设计与实现[D].浙江理工大学,2017.

[14]郑婷婷. 基于物联网的智慧农业控制管理系统[D].西安工程大学,2018.
[15]莫先. 基于STM32单片机家电控制及家居环境监测系统设计与实现[D].重庆理工大学,2016.
[16]尹孟征.基于 Android 的 APP 开发平台综述[J].通信电源技术,2016,33(04):154-155+213.
[17]韩丹翱,王菲.DHT11 数字式温湿度传感器的应用性研究[J].电子设计工程,2013,21(13):83-85+88.
[18]李升红. 基于STM32和WIFI技术的家居盆栽植物智能监控系统[D].武汉轻工大学,2018.
[19]赵晶. 单片机控制OLED显示系统研究[D].重庆大学,2006.
[20]薛翔,王琰.基于 ESP8266 的智能开关控制系统设计[J].电子世界,2018(21):147-148.
[21]程峥. 基于嵌入式和设备云平台的家庭植物工厂系统设计[D].天津工业大学,2019.
[22]高蒙. 基于机智云平台的远程监控系统开发关键技术研究[D].西安理工大学,2019.
[23]曾建清. 基于 STM32 的多功能空气质量监测系统设计[D].电子科技大学,2019.
[24]许锰. 基于云平台的家庭安防系统研究与设计[D].山东大学,2018.

附录 A



附录 A 图 系统硬件连接图

附录 B

```
/**********************************************************************
                        农业检测控制系统主程序
**********************************************************************/
#include "sys.h"
#include "delay.h"
#include "usart.h"
#include "led.h"
#include "dht11.h"
#include "timer.h"
#include "usart3.h"
#include "beep.h"
#include "oled.h"
#include "BH1750.h"
#include "gizwits_product.h"
#define TScope   currentDataPoint.valueTemp_AlarmScope
#define HScope   currentDataPoint.valueShidu_AlarmScope
#define GQScope  currentDataPoint.valueGQ_AlarmScope
/* 用户区当前设备状态结构体*/
dataPoint_t currentDataPoint;
//机智云协议初始化
void Gizwits_Init(void)
{
    TIM3_Int_Init(9,7199);//1MS 系统定时
    uart_init(9600);//这里的波特率只能是 9600
    memset((uint8_t*)&currentDataPoint, 0, sizeof(dataPoint_t));
    gizwitsInit();//缓冲区初始化
}
//数据采集、并在 OLED 显示数据、上报数据、判断是否触发警报
void userHandle(void)
{
    static u16 GQ_Data;
    static u8 temp, hum;
    OLED_ShowCHinese(0,0,0);  //x,y,第三位是汉字库的次序
    OLED_ShowCHinese(16,0,1);
    OLED_ShowCHinese(32,0,10);
    OLED_ShowString(64,0," ",16);//加空格是为了消除电压的影响
    OLED_ShowCHinese(88,0,18);
    OLED_ShowCHinese(88+16,0,18);
    OLED_ShowCHinese(72,0,8);
    OLED_ShowCHinese(0,2,2);
    OLED_ShowCHinese(16,2,3);
    OLED_ShowCHinese(32,2,10);
```

```
    OLED_ShowString(64,2," ",16);
    OLED_ShowCHinese(88,2,18);
    OLED_ShowCHinese(88+16,2,18);
    OLED_ShowCHinese(72,2,9);
    //光强
    OLED_ShowCHinese(0,4,4);
    OLED_ShowCHinese(16,4,6);
    OLED_ShowCHinese(32,4,10);
    OLED_ShowString(88,4," ",16);
    OLED_ShowCHinese(88+24,4,18);
    OLED_ShowString(96,4,"lx",16);//这里 lx 是光强的单位
    //“动模式”
    OLED_ShowCHinese(48+16,6,14);
    OLED_ShowCHinese(48+32,6,15);
    OLED_ShowCHinese(48+48,6,16);
    if(timeuser > 1000)//每 1 秒上报一次数据
    {
        timeuser=0;//在.h 文件里面已经声明过
        //在 oled 和 app 上显示温湿度值
        DHT11_Read_Data(&temp,&hum);//读取 DHT11 传感器
        OLED_ShowNum(48,0,temp/10%10,1,16);
        OLED_ShowNum(56,0,temp%10,1,16);
        OLED_ShowNum(48,2,hum/10%10,1,16);
        OLED_ShowNum(56,2,hum%10,1,16);
        //上传温湿度数据
        currentDataPoint.valueTemp = temp ;
        currentDataPoint.valueShidu = hum;
        //在 oled 和 app 上显示光强数值
        GQ_Data = BH1750_ReadContinuous1();  //读取 GY-30 传感器
        OLED_ShowNum(48,4,GQ_Data/10000%10,1,16);
        OLED_ShowNum(56,4,GQ_Data/1000%10,1,16);
        OLED_ShowNum(64,4,GQ_Data/100%10,1,16);
        OLED_ShowNum(72,4,GQ_Data/10%10,1,16);
        OLED_ShowNum(80,4,GQ_Data%10,1,16);
        currentDataPoint.valueGQ = GQ_Data;//上传光强数值
        //如果 led1 发送数据则响，开启警报，默认自动模式
        if(0 == currentDataPoint.valueLED1)
        {
            OLED_ShowCHinese(48,6,13);
            //判断是否超出给定值，若超出蜂鸣器响，led 闪烁
            if((temp>=TScope||hum>=HScope||GQ_Data>=GQScope)&&(TScope
!=0&&HScope!=0&&GQScope!=0))
            {
                OLED_ShowCHinese(0,6,17);//警告图标
```

```
                BEEP = 1;
                LED0 = 1;
                delay_ms(500);
                OLED_ShowCHinese(0,6,18);
                BEEP = 0;
                LED0 = 0;
            }
            else
            {
                BEEP = 0;
                LED0 = 0;
           currentDataPoint.valueLED = currentDataPoint.valueBEEP = 0;
             }
        }
       else
           OLED_ShowCHinese(48,6,11);
    }
}
//主函数
int main(void)
{
    delay_init();               //延时函数初始化
    //设置 NVIC 中断分组 2:2 位抢占优先级，2 位响应优先级
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
    LED_Init();                 //LED 端口初始化
    DHT11_Init();
    BEEP_Init();                //蜂鸣器初始化
    OLED_Init();                //初始化 OLED
    BH1750_Init();
    Gizwits_Init();             //协议初始化
    while(1)
    {
        userHandle();//用户采集函数
        gizwitsHandle((dataPoint_t *)&currentDataPoint);//协议处理
    }
}
/************************************************************************
                        DHT11 程序——dht11.c
************************************************************************/
#include "dht11.h"
#include "delay.h"
//复位 DHT11
void DHT11_Rst(void)
{
```

```
    DHT11_IO_OUT();  //SET OUTPUT
    DHT11_DQ_OUT=0;  //拉低 DQ
    delay_ms(20);        //拉低至少 18ms
    DHT11_DQ_OUT=1;  //DQ=1
    delay_us(30);        //主机拉高 20~40us
}
//等待 DHT11 的回应
u8 DHT11_Check(void)
{
    u8 reply=0;
    DHT11_IO_IN();
    while (DHT11_DQ_IN&&reply<100)//DHT11 会拉低 40~80us
    {
        reply++;
        delay_us(1);
    }
    if(reply>=100)return 1;
    else reply=0;
    while (!DHT11_DQ_IN&&reply<100)//DHT11 拉低后会再次拉高 40~80us
    {
        reply++;
        delay_us(1);
    }
    if(reply>=100)return 1;
    return 0;
}
//从 DHT11 读取一个位
u8 DHT11_Read_Bit(void)
{
    u8 reply=0;
    while(DHT11_DQ_IN&&reply<100)//等待变为低电平
    {
        reply++;
        delay_us(1);
    }
    reply=0;
    while(!DHT11_DQ_IN&&reply<100)//等待变高电平
    {
        reply++;
        delay_us(1);
    }
    delay_us(40);//等待 40us
    if(DHT11_DQ_IN)return 1;
    else return 0;
```

```
}
//从 DHT11 读取一个字节
u8 DHT11_Read_Byte(void)
{
    u8 i,dat;
    dat=0;
    for (i=0; i<8; i++)
    {
        dat<<=1;
        dat|=DHT11_Read_Bit();
    }
    return dat;
}
//从 DHT11 读取一次数据
u8 DHT11_Read_Data(u8 *temp,u8 *humi)
{
    u8 buf[5];
    u8 i;
    DHT11_Rst();
    if(DHT11_Check()==0)
    {
        for(i=0; i<5; i++) //读取 40 位数据
        {
            buf[i]=DHT11_Read_Byte();
        }
        if((buf[0]+buf[1]+buf[2]+buf[3])==buf[4])
        {
            *humi=buf[0];
            *temp=buf[2];
        }
    } else return 1;
    return 0;
}
//初始化 DHT11 的 IO 口 DQ 同时检测 DHT11 的存在
u8 DHT11_Init(void)
{
    GPIO_InitTypeDef  GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);
    DHT11_Rst();
    return DHT11_Check();
```

```
}
/**********************************************************************
                        DHT11 程序——dht11.h
**********************************************************************/
#ifndef __DHT11_H
#define __DHT11_H
#include "sys.h"
//IO 方向设置
//#define DHT11_IO_IN()  {GPIOA->CRH&=0XFFFF0FFF;GPIOA->CRH|=8<<12;}
//#define DHT11_IO_OUT() {GPIOA->CRH&=0XFFFF0FFF;GPIOA->CRH|=3<<12;}
//这里是高八位设置，若是低八位改成 L 即可
#define DHT11_IO_IN()  {GPIOB->CRH&=0XFFFFFF0F;GPIOB->CRH|=8<<4;}
#define DHT11_IO_OUT() {GPIOB->CRH&=0XFFFFFF0F;GPIOB->CRH|=3<<4;}
//IO 操作函数
#define    DHT11_DQ_OUT PBout(9)           //数据端口    PB9
#define    DHT11_DQ_IN  PBin(9)            //数据端口    PB9
u8 DHT11_Init(void);                       //初始化 DHT11
u8 DHT11_Read_Data(u8 *temp,u8 *humi);     //读取温湿度
u8 DHT11_Read_Byte(void);                  //读出一个字节
u8 DHT11_Read_Bit(void);                   //读出一个位
u8 DHT11_Check(void);                      //检测是否存在 DHT11
void DHT11_Rst(void);                      //复位 DHT11
#endif
/**********************************************************************
                        GY-30 程序——BH1750.c
**********************************************************************/
#include "BH1750.h"
#include "delay.h"
#include "sys.h"
#include "stm32f10x.h"
#include "stdint.h"
u8 BUF[8];
u16 temp2=0;
static u32 lux=0;
void Single_Write_BH1750(unsigned char Reg_Address)
{
    IIC_Start1();
    IIC_Send_Byte(0x46);  //发送器件地址 0100 0110 最后一位 0，表示写
    IIC_Send_Byte(Reg_Address);
    IIC_Stop1();
}
void BH1750_Init(void)
{
    IIC_Config();
```

```
    Single_Write_BH1750(0x01);
}
//从 BH1750 连续读 lx
u16  BH1750_ReadContinuous1(void)
{
    u16 temp=0,temp1=0;
    IIC_Start1();
    IIC_Send_Byte(0x46);  //发送器件地址 0100 0110  最后一位 0，表示写
    IIC_Wait_Ack1();
    IIC_Send_Byte(0x10);
    IIC_Wait_Ack1();
    IIC_Stop1();
    delay_ms(200);
    IIC_Start1();
    IIC_Send_Byte(0x47);
    IIC_Wait_Ack1();
    temp=IIC_Read_Byte(1);
    temp1=IIC_Read_Byte(0);
    IIC_Stop1();
    temp2=temp1+(temp<<8);
    lux=temp2;
    return lux;
}
void IIC_Config(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);
    GPIO_InitStructure.GPIO_Pin=GPIO_Pin_10|GPIO_Pin_11;
    GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode=GPIO_Mode_Out_PP;  //推挽输出
    GPIO_Init(GPIOB,&GPIO_InitStructure);
    GPIO_SetBits(GPIOB,GPIO_Pin_10|GPIO_Pin_11);
}
//总线初始化
void IIC_Start1(void)
{
    SDA_OUT()        //设置 SDA 线为输出
    IIC_SDA=1;    //发送起始条件的数据信号，释放总线
    delay_us(2);
    IIC_SCL=1;
    delay_us(5);   //Tsu;STA：起始条件的建立时间大于 4.7us。
    IIC_SDA=0;    //SDA 由高变为低表示开始信号
    delay_us(4);   //起始条件的保持时间大于 4us
    IIC_SCL=0;    //准备发送或者接收数据
```

```
    delay_us(2);
}
void IIC_Stop1(void)
{
    SDA_OUT()        //设置 SDA 线为输出
    IIC_SDA=0;   //发送停止信号的数据信号
    delay_us(2);
    IIC_SCL=1;   //发送停止信号的时钟信号
    delay_us(5);  //停止信号的建立时间大于 4us
    IIC_SDA=1;   //发送停止信号
}
u8 IIC_Wait_Ack1(void)
{
    u8 ucErrorTime=0;
    SDA_IN();
    IIC_SDA=1;
    IIC_SCL=1;
    while(READ_SDA)
    {
        ucErrorTime++;
        if(ucErrorTime>=250)
        {
            IIC_Stop1();
            return 1;
        }
    }
    IIC_SCL=0;
    return 0;
}
//发送一个字节
void IIC_Send_Byte(unsigned char c)  //要传送的数据长度为 8 位
{
    u8 i;
    SDA_OUT();
    for(i=0; i<8; i++)
    {
        if((c<<i)&0x80)  //判断发送位
        IIC_SDA=1;
        else IIC_SDA=0;
        IIC_SCL=1;     //拉高 SCL，通知被控器开始接收数据位
        IIC_SCL=0;    //拉低 SCL，允许 SDA 传输下一位数据。
    }
    IIC_SCL=0;      //拉低 SCL，为下次数据传输做好准备
    }
```

```
u8 IIC_Read_Byte(unsigned char ack)
{
    unsigned char i,receive=0;
    SDA_IN();
    for(i=0; i<8; i++)
    {
        IIC_SCL=0;    //置 SCL 为低，准备接收数据位
        delay_us(5);  //时钟低电平周期大于 4.7us
        IIC_SCL=1;   //置 SCL 为高，使 SDA 上数据有效
        receive=receive<<1;
       //读取 SDA，把接收的数据位放入 receve 中
        if(READ_SDA) receive=receive+1;
    }
    if(!ack) IIC_NAck();   //发送 NAck
    return receive;
}
/**************************************************************************
                        GY-30 程序——BH1750.h
**************************************************************************/
#ifndef __BH1750_H
#define __BH1750_H
#include "sys.h"
//IO 操作函数
#define IIC_SCL    PBout(10) //SCL
#define IIC_SDA    PBout(11) //SDA
#define READ_SDA   PBin(11)  //输入 SDA
//IO 方向设置，设置输入还是输出，半双工就这样
#define SDA_IN()  {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=8<<12;}
#define SDA_OUT() {GPIOB->CRH&=0XFFFF0FFF;GPIOB->CRH|=3<<12;}
void IIC_Config(void);
void IIC_Init(void) ;
void IIC_Start1(void);  //起始信号
void IIC_Stop1(void);  //停止信号
u8 IIC_Wait_Ack1(void);
void IIC_NAck(void);
void IIC_Send_Byte(unsigned char c);  //要传送的数据长度为 8 位
u8 IIC_Read_Byte(unsigned char ack);
void Single_Write_BH1750(unsigned char Reg_Address);
void BH1750_Init(void);
u16 BH1750_ReadContinuous1(void);
#endif
 /**************************************************************************
                    ESP8266 程序——gizwits_product.c
**************************************************************************/
```

```
#include "stdio.h"
#include "string.h"
#include "gizwits_product.h"
#include "common.h"
#include "usart3.h"
#include "led.h"
#include "beep.h"
#include "usart.h"
static uint32_t timerMsCount;
uint8_t aRxBuffer;
uint16_t  timeuser = 0;
extern dataPoint_t currentDataPoint;
//在这里控制蜂鸣器等
int8_t  gizwitsEventProcess(eventInfo_t  *info,  uint8_t  *gizdata,
uint32_t len)
{
    uint8_t i = 0;
    dataPoint_t *dataPointPtr = (dataPoint_t *)gizdata;
    moduleStatusInfo_t *wifiData = (moduleStatusInfo_t *)gizdata;
    protocolTime_t *ptime = (protocolTime_t *)gizdata;
#if MODULE_TYPE
    gprsInfo_t *gprsInfoData = (gprsInfo_t *)gizdata;
#else
    moduleInfo_t *ptModuleInfo = (moduleInfo_t *)gizdata;
#endif
    if((NULL == info) || (NULL == gizdata))
        return -1;
    for(i=0; i<info->num; i++)
    {
        switch(info->event[i])
        {
        case EVENT_LED:
            currentDataPoint.valueLED = dataPointPtr->valueLED;
            GIZWITS_LOG("Evt: LED%d\n", currentDataPoint.valueLED);
            if(0x01 == currentDataPoint.valueLED)
            {
                if(1 == currentDataPoint.valueLED1)
                    LED0 = 1;
            }
            else
            {
                if(1 == currentDataPoint.valueLED1)
                    LED0 = 0;
            }
```

```
            break;
        case EVENT_BEEP:
            currentDataPoint.valueBEEP = dataPointPtr->valueBEEP;
            GIZWITS_LOG("Evt: BEEP %d \n", currentDataPoint.valueBEEP);
            if(0x01 == currentDataPoint.valueBEEP)
            {
                if(1 == currentDataPoint.valueLED1)
                    BEEP = 1;
            }
            else
            {
                if(1 == currentDataPoint.valueLED1)
                    BEEP = 0;
            }
            break;
        case EVENT_LED1:
            currentDataPoint.valueLED1 = dataPointPtr->valueLED1;
            GIZWITS_LOG("Evt: LED1 %d \n", currentDataPoint.valueLED1);
            if(0x01 == currentDataPoint.valueLED1)
                LED2 = 0;
            else
                LED2 = 1;
            break;
        case EVENT_Temp_AlarmScope:
            currentDataPoint.valueTemp_AlarmScope                    =
dataPointPtr->valueTemp_AlarmScope;
            GIZWITS_LOG("%d\n",currentDataPoint.valueTemp_AlarmScope);
            break;
        case EVENT_Shidu_AlarmScope:
            currentDataPoint.valueShidu_AlarmScope                   =
dataPointPtr->valueShidu_AlarmScope;
           GIZWITS_LOG("%d\n",currentDataPoint.valueShidu_AlarmScope);
            break;
        case EVENT_GQ_AlarmScope:
            currentDataPoint.valueGQ_AlarmScope                      =
dataPointPtr->valueGQ_AlarmScope;
            GIZWITS_LOG("%d\n",currentDataPoint.valueGQ_AlarmScope);
            break;
        case WIFI_SOFTAP:
            break;
        case WIFI_AIRLINK:
            break;
        case WIFI_STATION:
            break;
```

```
        default:
            break;
        }
    }
    return 0;
}
void userInit(void)
{
    memset((uint8_t*)&currentDataPoint, 0, sizeof(dataPoint_t));
}
void gizTimerMs(void)
{
    timerMsCount++;
    timeuser++;
}
uint32_t gizGetTimerCount(void)
{
    return timerMsCount;
}
void mcuRestart(void)
{
    __set_FAULTMASK(1);
    NVIC_SystemReset();
}
//发送给云服务器
int32_t uartWrite(uint8_t *buf, uint32_t len)
{
    uint32_t i = 0;
    if(NULL == buf)
        return -1;
    for(i=0; i<len; i++)
    {
        USART_SendData(USART1,buf[i]);
       //循环发送,直到发送完毕
        while(USART_GetFlagStatus(USART1,USART_FLAG_TC)==RESET);
        if(i >=2 && buf[i] == 0xFF)
        {
            USART_SendData(USART1,0x55);
            //循环发送,直到发送完毕
            while(USART_GetFlagStatus(USART1,USART_FLAG_TC)==RESET);
        }
    }
#ifdef PROTOCOL_DEBUG
    GIZWITS_LOG("MCU2WiFi[%4d:%4d]: ", gizGetTimerCount(), len);
```

```
    for(i=0; i<len; i++)
    {
        GIZWITS_LOG("%02x ", buf[i]);
        if(i >=2 && buf[i] == 0xFF)
            GIZWITS_LOG("%02x ", 0x55);
    }
    GIZWITS_LOG("\n");
#endif
    return len;
}
/************************************************************************
                    ESP8266 程序——gizwits_product.h
************************************************************************/
#ifndef _GIZWITS_PRODUCT_H
#define _GIZWITS_PRODUCT_H
#ifdef __cplusplus
extern "C" {
#endif
#include <stdint.h>
#include <stm32f10x.h>
#include "gizwits_protocol.h"
extern uint16_t  timeuser;
#define SOFTWARE_VERSION "03030000"
#define HARDWARE_VERSION "03010100"
#define MODULE_TYPE 0 //0,WIFI ;1,GPRS
extern dataPoint_t currentDataPoint;
void gizTimerMs(void);
void timerInit(void);
void uartInit(void);
void userInit(void);
void userHandle(void);
void mcuRestart(void);
uint32_t gizGetTimerCount(void);
int32_t uartWrite(uint8_t *buf, uint32_t len);
int8_t gizwitsEventProcess(eventInfo_t *info, uint8_t *data, uint32_t
len);
#endif
```