|  |
| --- |
| **江 西 理 工 大 学**  **本 科 毕 业 设 计（论文）**  **题 目：智慧农业系统嵌入式开发**  **专题题目：**  **学 院： 信息工程学院**  **专 业： 电子信息工程**  **班 级： 电信152班**  **学 号： 1520153277**  **学 生： 陈嘉彬**  **指导教师： 黄友文 职称： 副教授**  **时间： 2019年 5 月 25 日** |

**江西理工大学 信息工程 学院 2019届**

**本 科 毕 业 设 计（论文）任 务 书**

**题 目：智慧农业系统嵌入式开发**

**专题题目**（若无专题则不填）：

**原始依据**(包括设计（论文）的工作基础、研究条件、应用环境、工作目的等)：

**工作基础**：在本科本专业教学体系中，开设了C++程序设计、微型计算机原理、单片机原理等课程，为本次课题奠定了扎实的理论基础。与此同时，在相关课程设计和生产实习等实践中，学生学习过ARM-CortexM系列MCU的使用，掌握了各类通信协议，为课题的研究开展和实施创造了条件。

**研究条件：**硬件条件——计算机、STM32开发板、各类传感器；软件条件——C语言、MDK开发环境，云服务器。

**应用环境：**农田信息采集，对农田的远程操控。

**工作目的：**

（1）使学生提高综合运用所学专业知识的能力；

（2）使学生掌握从事学术研究的基本方法；

（3）使学生掌握撰写学术论文的基本格式；

（4）使学生具备嵌入式系统开发的能力；

**主要内容和要求：**（包括设计（研究）内容、主要指标与技术参数，并根据课题性质对学生提出具体要求）：

设计实现一套智慧农业嵌入式系统，完成对农田各类信息的采集，并将数据上传至云端，同时可以从云端下发指令以实现对农田的控制功能。采用STM32开发板进行部署，在该平台上运行整个嵌入式系统，实现要求的功能。

要求完成以下工作：（1）掌握STM32在MDK环境下的开发流程（2）掌握C语言和各类通信协议的使用。（3）通过各类通信协议将STM32和各类传感器对接以完成信息的采集并将采集到的数据上传至云端服务器。（4）云端服务器通过网络协议下发指令，STM32接收指令后做出相应的控制动作（5）使用AltiumDesigner2014完成整个嵌入式系统的电路板设计。

**日程安排：**

第1～2周： 收集、整理课题资料，阅读相关文献（2周）

第3～4周： MDK和Arduino开发环境的搭建（2周）

第5～6周： 通信协议的使用（2周）

第7～8周： STM32和传感器之间的通信数据的上传和下发（2周）

第9～10周： 使用AltiumDesigner2014设计系统的电路板（2周）

第11～12周： 综合模块与系统功能测试，修复BUG（2周）

第13周： 撰写毕业设计论文（1周）

第14周： 做好答辩前的各项准备工作并参加答辩（1周）

**主要参考文献和书目：**

[1] 物联网在智慧农业系统中的应用[J]. 北京农业,2014(17):8-13.

[2] 智慧农业宏图[J]. 高科技与产业化,2015(05):30-31.

[3] 龚炳铮. 发展我国智慧农业的思考[J]. 办公自动化,2017,22(03):23-26.

[4] 谭浩强.c语言程序设计[M].北京：清华大学出版社，1991：1-100

[5]（英） JosephYiu.ARM Cortex-M3权威指南[M]. 北京：北京航空航天大学出版社.2009.07

[6] 刘一.基于STM32的嵌入式系统设计[M].北京：中国铁道出版社.2015.09

[7] 刘理云.嵌入式单片机开发与应用.北京：北京理工大学出版社.2016.01

[8] 齐美娟. 智慧农业瞄准成果转化[J]. 中国国情国力,2018(06):80.

[9] 张洁. 物联网智慧农业实训平台应用系统[J]. 物联网技术,2017,7(06):118-120.

[10]Journal of Integrative Agriculture Instruction to Authors[J]. Journal of Integrative Agriculture,2018,17(07):1696-1697.

[11]Xiaohang ZHANG,Shoufu CUI,Fuping LIU. General Situations of Development of Photovoltaic Agriculture[J]. Asian Agricultural Research,2015,7(10):13-16.

**指导教师**（签字）**：**

**年 月 日**

注：本表可自主延伸，各专业根据需要调整。

**江西理工大学 信息工程 学院 2019届**

**本 科 毕 业 设 计（论文）开 题 报 告**

**学号：1520153277 姓名：陈嘉彬 班级：电信152班**

**题 目：智慧农业系统嵌入式开发**

**专题题目**（若无专题则不填）**：**

**本课题来源及研究现状：**

**课题来源**

我国作为农业大国，农业直接影响到我国的基础经济，如何更好的发展好我国的农业，是一个非常值得研究的问题。

我国人口数量大，人均耕地面积小，提高农作物的产量是很有必要的，几十年前，袁隆平通过生物技术迅速的提高的农作物的产量，但是我们不可能通过生物技术无限的提高产量，况且农作物产量低下也并非完全是基因不良导致的，恰恰相反，农作物产量低下往往是因为管理不当造成的，而智慧农业则可以很好的解决这类问题，智慧农业将可以实时的检测农田的信息，并作出相应的调整。利用现代化做到精确掌握农田信息。二十一世纪以来我国着重科技强国，在传统的农业方面如何结合现代化科技一直以来都是热门话题，目前为止，现代化农业已经完成了雏形，只是还没有全面普及。如何迅速的全面普及智慧农业将会是二十一世纪新的研究方向，再者利用智慧农业的模型还可以迅速普及到林业，养殖业等其他行业，所以智慧农业的研究也是行业研究的热门课题。

**研究现状**

智慧农业的研究开始于本世纪初，经过十几年的研究和发展目前已经取得了较好的成果。不过，目前的研究成果主要是人工创造农作物适宜的环境，例如蔬菜大棚等。这类方法的缺陷是成本高、面积小、种植数量有限。因此我们还是要将农作物放置在最原始的自然环境中去培养以降低成本。

大面积培养种植的最大难度在于不方便管理。同时随着改革开放，大部分农民都进城工作，导致了真正在农村种田的农民大量减少，而耕地减少的幅度并不大。如何实现少数几个人种植、管理大量的农田将会是一大问题。智慧农业对这类问题做了良好的解决方案，首先智慧农业系统的传感器模块将采集农田的各类信息并将信息传送至云端服务器。农民只需访问服务器就可以全面了解农田的真实情况，同时智慧农业的控制模块可以实现远程操控，完成对农田环境的调整例如自动灌溉等。这样既可以远程获取农田的基本信息，又可以远程控制农田的状况。而无需农民亲自到农田去观察。可以节省大量的人力物力，以实现少数的农民种植大量的农作物。避免耕地荒废的同时还可以科学方便的管理。智慧农业实现了农业的现代化，是现代化科学技术逐渐走向实用化。

**课题研究目标、内容、方法和手段：**

**研究目标：**

完成基于ARM-CortexM嵌入式系统的开发、采用STM32作为MCU外加各类传感器实现对农田信息的采集，以及对农田的远程控制。

**研究内容：**

（1）STM32嵌入式系统开发方法。

（2）各类通信协议和传感器的使用。

（3）主控电路板的设计。

（4）网络通信的上传与下发接口和程序设计。

**研究方法和手段：**

本嵌入式系统分为软件部分和硬件部分。其中软件部分是利用AltiumDesigner2014绘制系统所需的硬件电路板，硬件部分是由MCU、传感器、驱动电路等组成。本系统是利用MDK开发环境完成对STM32的开发，通过各种通信协议完成和各类传感器的数据交换。并将交换来的数据通过ESP8266模块发送到服务器，同时服务器也可以下发各种指令，STM32也是通过ESP8266来接收，处理接收来的数据之后做出相应的控制动作。

**设计（论文）提纲及进度安排：**

第1～2周： 收集、整理课题资料，阅读相关文献（2周）

第3～4周： MDK和Arduino开发环境的搭建（2周）

第5～6周： 通信协议的使用（2周）

第7～8周： STM32和传感器之间的通信数据的上传和下发（2周）

第9～10周： 使用AltiumDesigner2014设计系统的电路板（2周）

第11～12周： 综合模块与系统功能测试，修复BUG（2周）

第13周： 撰写毕业设计论文（1周）

第14周： 做好答辩前的各项准备工作并参加答辩（1周）

**主要参考文献和书目：**

[1] 谢希仁.计算机网络（第七版）[M].北京：电子工业出版社.2017:1-400

[2] 谭浩强.c语言程序设计[M].北京：清华大学出版社，1991：1-100

[3]（英） JosephYiu.ARM Cortex-M3权威指南[M]. 北京：北京航空航天大学出版社.2009.07

[4] 刘一.基于STM32的嵌入式系统设计[M].北京：中国铁道出版社.2015.09

[5] 刘理云.嵌入式单片机开发与应用.北京：北京理工大学出版社.2016.01

[6] 李道亮. 物联网与智慧农业[J]. 农业工程,2012,2(01):1-7.

[7] 施连敏,陈志峰,盖之华. 物联网在智慧农业中的应用[J]. 农机化研究,2013,35(06):250-252.

[8] 路顺涛,林珂,蒋玲,徐啸峰,马奉先,马乐. 智慧农业发展趋势浅析[J]. 中国管理信息化,2016,19(05):170-171.

[9] 美国打造智慧农业的启示[J]. 北京农业,2014(17):24-27.

[10]HUANG Guo-qin,ZHAO Qi-guo,GONG Shao-lin,SHI Qing-hua. Overview of Ecological Agriculture with High Efficiency[J]. Asian Agricultural Research,2012,4(09):71-77.

[11]Manisha Basu,Manish Pande,P.B.S.Bhadoria,S.C.Mahapatra. Potential fly-ash utilization in agriculture:A global review[J]. Progress in Natural Science,2009,19(10):1173-1186.

**指导教师审核意见：**

**指导教师（**签字）**： 年 月 日**

注：本表可自主延伸

摘 要

智慧农业是传统农业和IOT结合的产物，利用物联网、嵌入式技术使得农业的生产、管理更加方便科学。智慧农业技术可以广泛的应用在农田信息的测量，农田器械的操作等方面。智慧农业技术还可以推广到我国的林业、养殖业、畜牧业等方面。因此智慧农业具有非常重要的价值，智慧农业也是当前研究的热门课题。

本论文采用STM32作为嵌入式系统的MCU，同时采用WiFi作为物联网的无线通信协议。DHT11、RFID、TSL2561等传感器外设采集数据。外加AltiumDesigner软件绘制的电路板共同构建了整套嵌入式系统的硬件部分。而软件部分则是由嵌入式固件以及网页、云服务器等组成。智慧农业嵌入式系统主要功能为采集农田的温湿度、光强等信息，并将这些信息通过WiFi传送至云端服务器。同时云端服务器也可以通过WiFi下发一个指令，智慧农业嵌入式系统接收到这个指令后可以做出一些相关的操作。例如远程控制浇水、开灯等。当然智慧农业还有一些无需联网的本地操作，例如通过RFID识别用户身份等。

**关键字**：智慧农业；IOT；嵌入式；STM32

**ABSTRACT**

Smart agriculture is the product of the combination of traditional agriculture and IOT, Using the Internet of Things and embedded technology to make agricultural production and management more convenient. Smart agricultural technology can be widely used in the measurement of farmland information, the operation of farmland equipment, etc. Smart agricultural technology can also be extended to China's forestry, aquaculture, animal husbandry and other aspects. Therefore, smart agriculture has very important value, and smart agriculture is also a hot topic in current research.

This thesis uses STM32 as the MCU of the embedded system, and uses WiFi as the wireless communication protocol of the Internet of Things. Sensor peripherals such as DHT11, RFID, and TSL2561 collect data. In addition, the boards drawn by AltiumDesigner software together build the hardware part of the entire embedded system. The software part is composed of embedded firmware and web pages, cloud servers and so on. The main function of the intelligent agriculture embedded system is to collect information such as temperature, humidity and light intensity of the farmland, and transmit the information to the cloud server via WiFi. At the same time, the cloud server can also issue an instruction through WiFi, and the intelligent agriculture embedded system can perform some related operations after receiving this instruction. For example, remote control of watering, turning on lights, etc. Of course, smart agriculture also has some local operations that do not require networking, such as identifying users by RFID.

**KeyWords:** Smart agriculture; IOT; embedded; STM32

**目 录**

[**第一章 绪论 10**](#_Toc1203933)

[1.1课题来源与背景 10](#_Toc1203934)

[1.2智慧农业发展现状 10](#_Toc1203935)

[1.3研究目的与意义 10](#_Toc1203936)

[1.4本章小结 10](#_Toc1203937)

[**第二章 智慧农业的总体结构 10**](#_Toc1203938)

[2.1 总体设计与原理分析 10](#_Toc1203939)

[2.2 云端控制系统 10](#_Toc1203940)

[2.3本地端控制系统 10](#_Toc1203941)

[2.4 本章小结 10](#_Toc1203942)

[**第三章 智慧农业嵌入式系统的硬件电路设计 10**](#_Toc1203943)

[3.1 STM32F103C8T6核心电路 10](#_Toc1203944)

[3.2 电源电路 10](#_Toc1203945)

[3.3 传感器模块电路 10](#_Toc1203946)

[3.3.1 10](#_Toc1203947)

[3.4控制模块电路 10](#_Toc1203948)

[3.4.1 11](#_Toc1203949)

[3.5 PCB封装与实现 11](#_Toc1203950)

[3.6 本章小结 11](#_Toc1203951)

[**第四章 智慧农业嵌入式系统的软件平台设计 11**](#_Toc1203952)

[4.1 单总线协议 11](#_Toc1203953)

[4.2 I2C协议 11](#_Toc1203954)

[4.3 ESP8266 11](#_Toc1203955)

[4.5 服务器 11](#_Toc1203956)

[4.6本章小结 11](#_Toc1203957)

[**第五章 总结 11**](#_Toc1203958)

1. 绪论

1.1课题来源与背景

自古以来我国都是农业大国，农业也一直作为我国的基础经济，可以说我国的经济发展离不开农业的发展。随着21世纪互联网、物联网时代的来临，我们的传统农业已经逐渐不适应现在的快时代。为此我们提出了利用现代科技与传统方法结合的模型——物联网+传统农业，这便是我们所说的智慧农业。

智慧农业是对传统农业的一个有效的改良。在传统农业模式下，我国现在依旧实行土地承包责任制。在此制度下虽然每家每户都有一亩三分地，然而实际上大部分的农村人来到城市工作，这就导致了我国有大部分的土地是荒芜的，在传统农业的耕种下让少数人种植大面积农田是不可能的。但我们智慧农业可以很好的解决这个问题。

大面积种植农田最大的困难就是农田的管理，智慧农业将采取传感器加物联网的方式帮助农民收集农田的一些相关的数据，例如温度、湿度等。然后将这些得到的数据通过WiFi的方式上传至网络云端，农民只需在电脑或者手机上就可观测农田的状况，同时还可以在电脑或者手机上向智慧农业发送指令，也是通过WiFi的方式传到农田里去，农田将会对接收到的指令做出相应的动作。

在智慧农业的帮助下，可以实现少数人种植大面积的农田，让我国不再有荒废的耕地，同时智慧农业还可以尽量的将环境调节为农作物理想的生长环境，又可以进一步的提高农作物的产量。

1.2智慧农业发展现状

智慧农业这个概念早在上世纪九十年代就已经提出，到如今已有几十年的发展。一些西方的发达国家，英国、美国等已经有了较为全面的技术和成套的设备。已经可以做到自动调节环境至农作物所需的最佳环境，且可以不受外界自然环境的干扰。还可以利用航天卫星对地面的资源进行检测，并将检测的数据发到云端分析，而我国发展智慧农业较晚，至今也才十几年的时间，相对一些西方的发达国家落后。但我国的智慧农业发展速度却较西方发达国家快。近年来，我国在智慧农业领域获得了很多成果，其中我国自主研发的“农用通”等已经相对成熟。该产品可以精确的获得监控数据，有良好的数据传输和人机交互。操作简单，其功能还在不断的完善，不过价格也相对偏高。

1.3研究目的与意义

目前现有的智慧农业系统，国外的虽然发展比较成熟，但是我们学不到其中的核心技术，而购买全套设备则价格非常昂贵。国内的又良莠不齐，性能好些的价格居高不下，相对便宜的又经常出故障。为此我们计划研究实现一套性能各方面相对优秀且价格也相对便宜的智慧农业系统。希望可以更好更快的推广我国的农业的发展。

为了降低智慧农业系统的成本，我们采用了目前市面上性能较好，价格又比较便宜的STM32作为系统的MCU，同时利用各类传感器来采集数据，并将所得到的数据发送至网关节点。同时将这些数据在云端进行可视化。我们也可以在云端下发一些指令，通过嵌入式端的解析可以做出相关的操作。而且我们已经在实验室完成了这部分的实验。从实验结果上分析，我们的智慧农业系统稳定性好，实用性强，成本较低。可以很好的进行推广。

1.4本章小结

本章主要介绍了本论文课题的来源与背景，从我国现状分析，我们需要一个能够帮助少数人种植大面积农田的智慧农业系统，同时我们还查阅了一些资料，观望和借鉴了他国的发展状况，以及我国现有的实际水平，虽然目前我国的智慧农业方面和一些西方发达国家还存在着不小的差距。由于我国的发展速度较西方国家快，所以这差距正在逐渐的缩短。我们研究的目的是为了更快更好的向全国推广智慧农业，同时也为本论文明确了研究方向。

第二章 智慧农业的总体结构

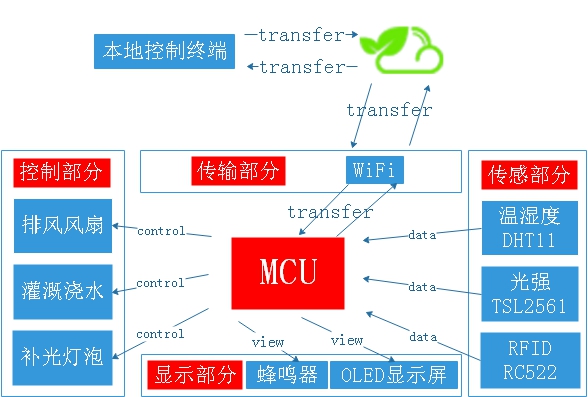
2.1 总体设计与原理分析

图2-1系统总体结构图

图2-1为智慧农业嵌入式系统的总体结构示意图，从图中可以看出智慧农业嵌入式系统主要由控制部分、显示部分、传感部分、和传输部分组成。各部分之间通过MCU连接起来，MCU协调各部分之间的工作。控制部分由排风风扇、灌溉喷头、补光灯泡组成。其中排风风扇为智慧农业中的温室大棚提供流动的空气，灌溉喷头为智慧农业提供水资源，是智能、远程喷灌的重要组成部分。补光灯泡主要是在阴雨天为农作物补充光照，为农作物的光合作用提供重要的条件。显示部分主要由蜂鸣器提供声音的显示，OLED显示屏提供文字内容的显示。传感部分有各类传感器组成，其中我们采用了DHT11作为温湿度传感器来监测农田的温湿度状况，采用TSL2561作为光照强度传感器，采用RC522利用RFID无线识别技术作为智慧农业温室大棚的门禁控制。传输部分目前采用WiFi传输，承担着整个智慧农业嵌入式系统的数据传输，数据传输分为两部分，一是数据的上传，也就是各类传感器所测得的数据，二是命令的下发，也就是远程操控下发的命令，嵌入式系统可以解析相应的命令并作出相关的动作。

2.2云端控制系统

云端服务器种类较多，目前也有一些做的比较好的开源项目，比如中国移动推出的中移物联，还有贝壳物联等等。而我们采用自主搭建的云服务器（以下简称云端），这样可以做到高度定制，去掉一些不必要的东西为我们的智慧农业量体裁衣。

2.2.1嵌入式端和云端通信的通信方式

我们的云端和嵌入式端的通信采用MQTT（消息队列遥测传输协议）传输协议。MQTT传输协议也是目前物联网流行协议之一。 是一种发布、订阅模式的协议，一种轻量、简单、开放、易于实现的协议。MQTT协议是基于TCP/IP协议上的，其最大的优点在于仅仅只用极少的代码和带宽就能获得和远程设备的连接，并且保证非常可靠的数据传输，同时他还提供了一种一对多的消息分发来实现应用程序的解耦，非常适合我们的嵌入式系统。

MQTT协议的实现它需要客户端和服务器端，同时还需要有三种身份的扮演者，他们分别是发布者、代理和订阅者，其中我们的嵌入式可以作为发布者或订阅者。云端的后台服务器作为代理，前台网页也可以作为订阅者或发布者。

智慧农业嵌入式系统选用了MQTT作为和远程通信的协议，而具体的数据流则采用了JSON格式来打包。其优点在于格式简单，方便读写，且格式是压缩过的，占用带宽小。虽然JSON也有它的缺点。比如：必须是Unicode字符集，否则很容易出现乱码现象。语法必须要非常的严谨，否则容易解析出错等等。但是对于我们智慧农业嵌入式系统来说，因为嵌入式端和云端都是我们自己设计的，所以我们可以统一所有的编码都采用Unicode。这样可以避开因跨平台带来的乱码困扰。而如果我们不小心在语法上出错，我们的云端将会自动丢弃不能解析的数据包。这样我们就可以解决因JSON缺点带来的麻烦。同时可以享受JSON的优点为我们带来的便利。JSON的格式要求非常严格，一个数据包的内容用花括号括起来。花括号里面有多个键值对，键值对之间用逗号隔开。每个键值对又分为前后两部分，分别用双引号引住，中间用冒号隔开（用数学公式形式表示如下：{"键值对1前":"键值对1后",…,"键值对n前":"键值对n后"}）。JSON只对数据包的格式做了严格的要求，而对数据包的大小却没有规定，这样当我们嵌入式系统想要向云端上传数据的时候便可选择性的上传，没有较大变化的数据便可不必上传，这样可以减少网络资源的使用。具体的体现无非就是JSON格式的数据包的长短不一样而已。而我们云端是怎么知道上传数据的意思呢？这个也是依赖于JSON格式。一般来说键值对的前半部分代表主题。比如究竟是温度还是湿度，后半部分则代表内容，比如是几度还是十几度。虽然JSON格式非常严格，但也正是这种严格的格式要求带来了数据的准确传输。利用JSON传输数据不仅在数据的传输错误率上得到了保障，同时也方便了我们云端的解析。

我们可以看出，不管是在传输协议上的MQTT还是在传输格式上的JSON。它们都有一个共同的优点，那就是在保证网络传输可靠性的前提下，尽可能的减少代码的复杂度和带宽的占用。这样可以减少因网络短暂的中断和网络的延迟而带来的数据部分的缺失，甚至是整个数据包的丢失。从而来减轻网络的负担。

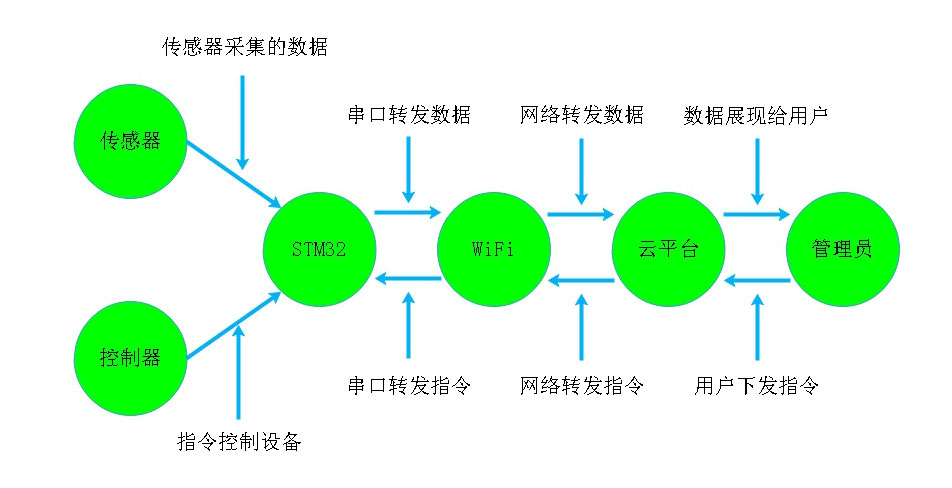
2.2.2数据流的传输过程

图2-2数据流向图

我们选定了一个非常适合智慧农业系统数据的传输方式，那么数据究竟是怎么传输的呢？智慧农业的数据传输如图2-2所示。图中分为上支路和下支路。其中上支路是先由传感器获取相关物理量的数据，然后通过某种通信协议（比如单总线）将数据传输至STM32，STM32将得到的数据打包成JSON格式通过串口发送给WiFi模块，WiFi模块再将数据包通过MQTT转发给云平台，云平台收到数据解析后以网页的形式图像化展现给管理员观看。至此一条数据完成了它的上传之路。而下支路和上支路非常类似。可以看成是上支路的逆过程。首先管理员想要下发某个命令，需要在网页上点击某个按钮，然后云平台通过MQTT将此条指令下发至WiFi模块，WiFi模块接收到数据包之后原封不动的将数据包通过串口发给STM32，STM32通过解析后以某种形式（比如GPIO引脚高低电平的变换）来操作控制器。至此一条指令完成了它的下发之路。

2.2.3云端界面

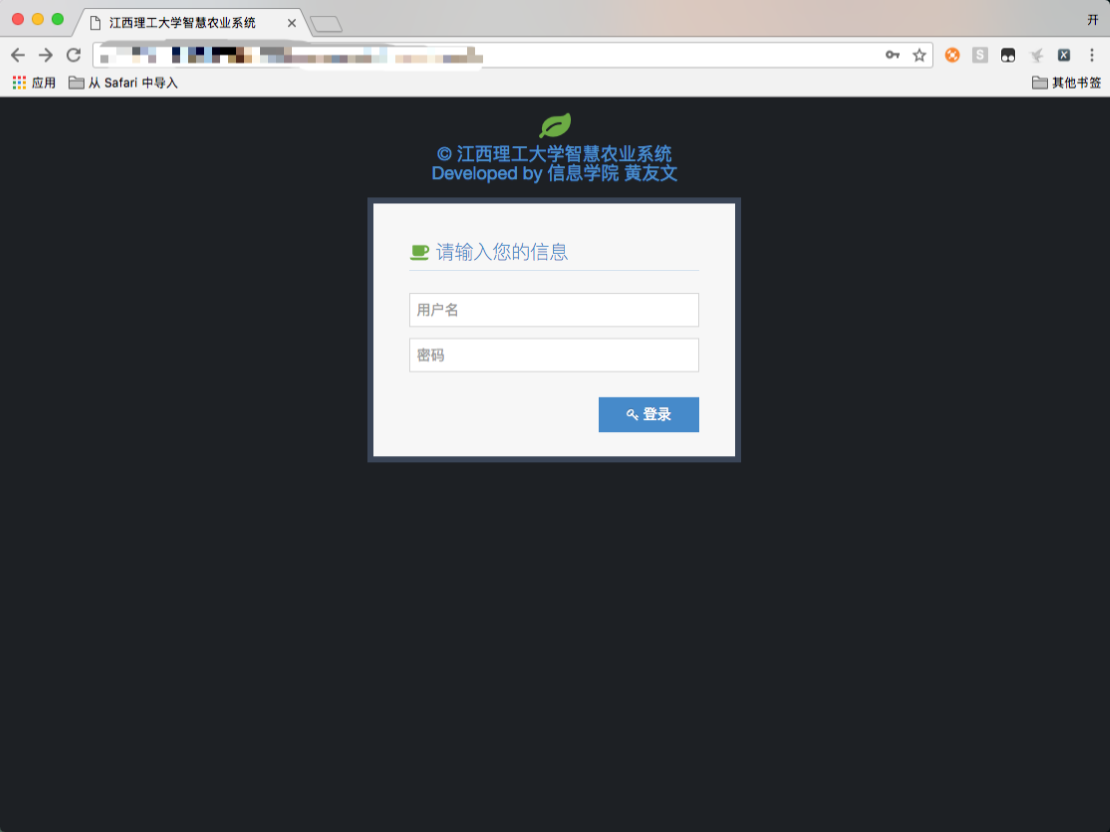


图2-3 云端登录界面

我们为云端的访问设置了一层屏障，可以防止陌生人刻意修改、下发指令导致嵌入式端做出不必要的动作，类似于所有的平台需要用户名密码进行登录操作，目前我们还没有设计网页版的注册方式。如有需要注册新的用户我们可以在后台的数据库新增一位用户。当输入用户名和密码登录后我们将可以看到如图2-4的选择界面。目前我们设计有如图2-5的历史数据，可以观测一周之内的数据，并且还将数据做成折线图，可以很清晰直观的看出数据变化的走向和趋势。当然数据也有如图2-6的列表模式。此模式下一般只能看到十几个数据，且每个数据之间的时间间隔非常短，一般为三秒钟。所以基本也就是一个实时数据。这种模式相比于折线图模式的好处在于，他可以看到精确的数据，比如现在的温度为33.7℃。无论是折线图模式还是列表模式。他们都有一个共同的缺点，那就是一次只能观看一个数据。所以我们还设计了一个仪表盘模式如图2-7所示。在仪表盘模式下，首先可以看到智慧农业上传数据的全部种类，还可以很清楚直观的通过每一个仪表的指针看到某类数据的一个大概的值，再看仪表的下方可以清楚的看到这类数据的瞬时精确值。

折线图、列表和仪表盘三种模式可以非常清楚直观的将智慧农业嵌入式端上传的数据多样化的展现在管理员的面前。当然管理员需要的不仅仅是远程观测若干数据，更多的是需要远程操控。这样以便尽可能的营造出适合农作物的

图2-4 云端选择界面

图2-5 历史数据

图2-6 数据列表模式

图2-6数据仪表盘模式

图2-7 控制管理

生长的最佳环境，这样自然就少不了控制部分。云端控制部分如图2-7所示，目前控制管理界面大致可分为数字控制和模拟控制两大类，图2-7显示了两个数字控制按钮和一个模拟控制滑块，数字控制按钮主要控制布尔型的量，比如水阀的开启关闭两种状态，而模拟控制滑块可以在一定的区间内调节物理量的多少，比如控制灯由暗逐渐变亮的过程。

2.3本地端控制系统

尽管我们有一套比较完善的云端管理控制系统。我们还是制作了一个本地的控制系统，如图2-8所示。我们制作本地控制系统的目的在于弥补云端的某些不足，因为我们不是很方便的在大棚内放置一台电脑来作为智慧农业的显示和控制终端，为此我们特意设计了一款小巧，便于携带的移动终端。此移动终端可以放置在大棚内的任意角落，也可悬挂在某个固定的地方。使用起来非常方便。其界面参考了部分云端的显示界面。主要由仪表盘显示部分和控制部分组成。移动终端由电阻触摸屏加WiFi模块组成。其工作原理是WiFi模块接收到了数据通过串口发给屏幕，屏幕将其数据显示出来，这个数据的来源是嵌入式端WiFi发出的。而控制部分是点击屏幕后，屏幕通过串口发送特定的指令给WiFi模块，WiFi模块将会模拟网页点击操作将此操作发送给服务器，通过服务器的转接将本次操作事件发送给嵌入式端完成操作。

图2-8 本地移动终端

2.4 本章小结

本章主要介绍了智慧农业嵌入式系统的整体框架。系统由嵌入式端和云端组成，嵌入式端由传感部分、控制部分、显示部分和传输部分组成。它们分别由STM32来协调分配管理。云端由数据由显示部分和控制部分组成。嵌入式端和云端采用MQTT协议外加JSON格式的数据流来进行通信，同时为了方便我们还设计了一个本地端的终端。通过确定整体框架，为我们后续的软硬件设计打下一个良好的基础，同时也为后面的软硬件设计指明一个方向。

第三章 智慧农业嵌入式系统的硬件电路设计

3.1 STM32F103C8T6最小系统

我们智慧农业嵌入式系统采用STM32F103C8T6作为主控芯片。我们都知道在嵌入式领域中任何代码的运行都需要硬件电路的支持。而在所有电路当中和MCU关系最紧密的也是最重要的当属我们的最小系统了。最小系统的意思是能让MCU正常工作的最小电路单元。不同MCU的最小系统也会略有不同，而C8T6这款芯片的最小系统由晶振电路、复位电路和启动电路几部分组成。

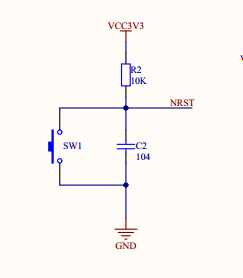
下面我们简单的介绍下这几个电路的组成和功能。晶振电路由晶体振荡器和起振电容组成。晶体振荡器可以提供一个非常稳定的震荡频率，因此非常适合做单片机的时钟供应来源。晶振有有源和无源之分。有源晶振比无源更稳定，同时成本也会高许多，其优点是不需要外围电路但需要单独供电。而无源和有源是互补的，我们从不需要单独供电、经济实惠等方面考虑选择无源晶振，付出的代价是增加一个起振电容的外围电路。起振电容的作用是使晶振两端的等效电容接近或等于负载电容和起到一定的滤波作用，可以过滤晶振中的一些高频杂波。一般选用8M无源晶振和20PF的电容就行。虽然晶振只提供8M频率的方波。但由于STM32内部可以倍频。所以STM32可以跑到72M的主频。复位电路的作用是不管程序运行到什么状态都可以快速恢复到单片机启动状态。复位电路图如图3-1所示。当按键按下时复位引脚和地导通，产生一个低电平，从而实现复位。

图3-1 复位电路

STM32启动方式由两个引脚boot0和boot1来决定。boot引脚需要接入一个100K的高阻来限制低电流以免电流过大烧坏单片机。启动方式如表3-1所示。

表3-1 boot启动方式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BOOT1 | BOOT0 | 启动方式 | 说明 | |
| X | 0 | 主闪存存储器 | 主闪存存储器被选为启动区域 |
| 0 | 1 | 系统存储器 | 系统存储器选为启动区域 |
| 1 | 1 | 内置SRAM | 内置SRAM被选为启动区域 |

当boot0接地时，无论boot1接VCC还是GND，STM32都是从主闪存启动的，主闪存是STM32内置的flash。一般我们用jlink等下载器下载程序就是下载在这个区域。重启也是从这里启动。当boot0接VCC，boot1接GND时，STM32从系统存储器启动。系统存储器就是一块ROM空间，只不过这个ROM空间的位置比较特别，一般是STM32可寻址空间的开头。内容也比较特殊，因为厂家会在这里放一段BootLoader程序，且这个程序是不可以被改变的。一般串口下载会选用这个启动方式。当boot0接VCC，boot1也接VCC时，STM32从内置SRAM中启动，也就是从内存中启动。此种启动方式有个缺点。因为内存是掉电丢失数据的，所以当系统掉电后是不能在重新启动的。此种方式一般只是用来调试代码时用的。

3.2 电源电路

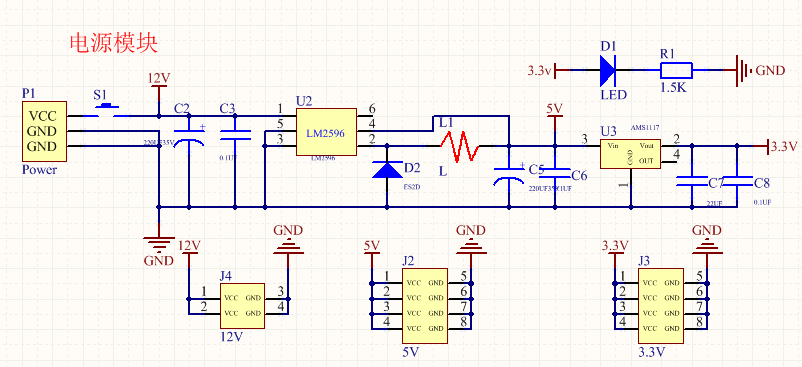
其实我们的STM32自带一个小的电源电路来给芯片供电，但是考虑到我们有部分控制电路需要12V电压。为了统一供电我们放弃STM32自带的供电系统。STM32的电源改为从GPIO引入。又因为大多数传感器为5V、3.3V供电所以我们的电源必须要同时有12V、5V、3.3V。如图3-2所示，为我们的电源电路，外界

图3-2 电源电路

供入12V的电，由LM2596电源转化芯片可得5V电压，为保证输入电压的稳定性我们加入了C2、C3滤波电容。同理为了保证输出电压的稳定性，我们加入了C5、C6两个电容。为了保证电源电路的健壮性，还加入了反向过压保护二极管D2。得到5V电压之后再通过AMS1117电源转化芯片就可以得到3.3V的稳定电压了。

3.3 蜂鸣器电路

蜂鸣器有有源和无源之分，在外观上来看，有源蜂鸣器比无源厚一些，有源蜂鸣器的末端用黑胶封住，而无源蜂鸣器则裸露了绿色的电路板。在使用上也是有区别的，有源蜂鸣器控制相对简单些，类似于发光二极管，给个高电平就可以发声，而无源蜂鸣器的两个引脚不分正负，需要提供一个PWM波才能控制发声。我们为了节约软件资源，同时也为了操作简单，选用了有源蜂鸣器。

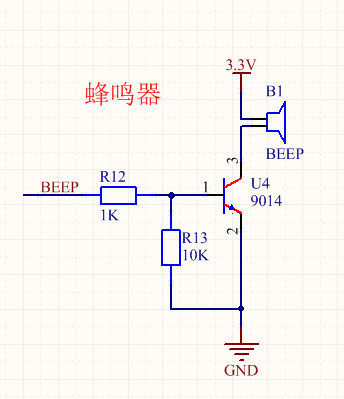
虽然有源蜂鸣器理论上提供一个高电平就可以发声，但实际上单片机IO引脚能提供的电流非常小，所以发出的声音也非常小几乎听不到的那种。为此我们需要增加一个驱动来提高电流。我们采用三极管放大的方法。如图3-3。三极管相当于一个开关电路的同时也起到了放大作用。三极管的基极接至单片机的IO引脚，若IO引脚输出高电平三极管导通有电流流过蜂鸣器。反之IO引脚输出低电平，三极管截止无电流流过蜂鸣器。如果IO引脚输出高电平，蜂鸣器位于三

图3-3 蜂鸣器电路

极管的集电极上，流经它的电流等于基极电流乘以三极管的放大倍数。这样便将电流放大了足以驱动蜂鸣器的正常发声。

另外本系统中部分控制器需要用到继电器，继电器的驱动电路也采用蜂鸣器的驱动电路。原因和原理都非常类似就不重复介绍。

3.4温湿度传感器电路

温湿度传感器采用了单总线协议传输数据，所以电路也非常简单。除了提供VCC、GND接口之外仅仅需要提供一根引线和单片机相连就行。因为这条是传输数据的

数据线，所以我们最好还是加上一个上拉电阻。如图3-4，常见的温湿度传感器

有DHT11、DS18B20等。简单介绍下两者的区别，DHT11有温湿度数据，而DS18B20

只有温度数据没有湿度数据。DHT11的量程为0~50℃而DS18B20的量程为-55℃～

+125℃且精度比DHT11要高，外观还要小巧。虽然DHT11是4引脚封装而DS18B20

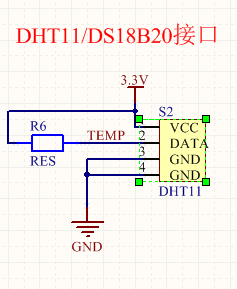
为3引脚封装，但它们都是以单总线协议和单片机通信。为了以后有更好的选择，我们在硬件电路上做了兼容处理，也就是说这个借口既可以插DS18B20也可以插DHT11。不过这次的嵌入式系统采用了DHT11传感器。

图3-4 温湿度传感器电路图

3.5 电机驱动电路

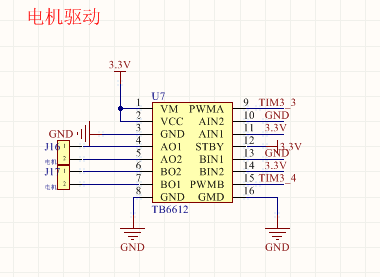
智慧农业嵌入式系统设有排风换气设备，涉及到风扇的工作。电机是不能直接接在单片机引脚上的。原因有二。1、单片机的引脚不能提供电机正常工作时的电流（这一点和蜂鸣器类似）。2、电机旋转时会产生一个反电动势，若电机

图3-5 电机驱动电路图

突然卡住，由反电动势产生的电流会倒灌入单片机，很容易将单片机烧毁。所以在单片机和电机之间加入一个驱动电路，如图3-5我们采用TB6612模块。它是一个能同时驱动A、B两路电机的模块。其中STBY为总使能引脚。我们将它固定接入高电平，也就是一直使能。IN1、IN2高低电平的不同搭配可以使电机正反转。但由于风扇反向旋转不能产生气体的流动所以就固定了正转模式。PWM引脚可以理解为单路使能引脚，一般接一个PWM波，控制PWM的占空比就控制了平均使能时间。也就间接控制了电机的转速。O1、O2为输出引脚直接接电机就行。需要注意的是。此模块有两个电源引脚VCC和VM。事实上VCC只是给模块本身供电而VM才是真正给电机供电的电源。由于我们的电机只是带动一个小风扇，可以接3.3V，具体接多少伏特的电压由电机的额定电压决定，最高不超过15V。

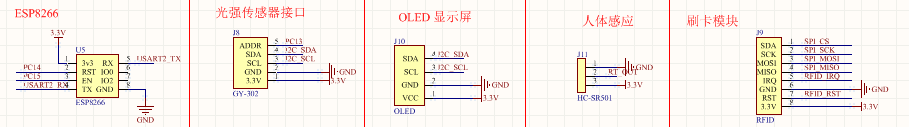
3.6 其他传感器电路

图3-6 其他传感器电路图

本嵌入式系统还有很多传感器，例如ESP8266WiFi、TSL2561光强传感器、OLED显示屏、RC522读卡设备。不过这些传感器需要的外围电路已经被模组厂商做成了模块，同时引出了通信接口。所以我们的电路板也只是引出了通信接口和模块对榫，就不多做介绍。

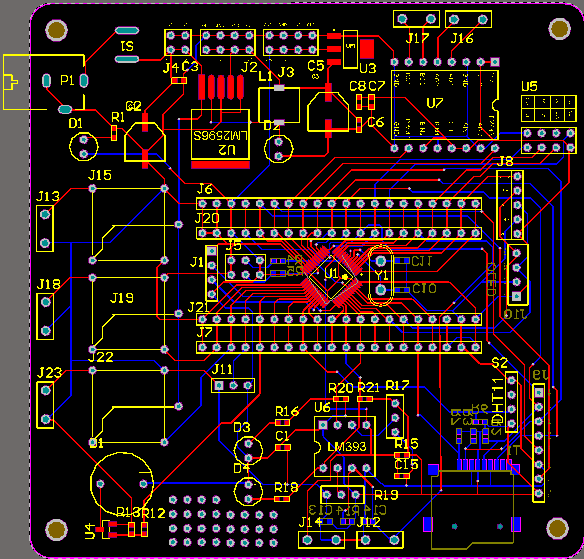
3.7 PCB封装

图3-7 PCB图

在设计完硬件原理图后就是PCB的封装了。PCB的封装主要目的是为了和

实物大小相对应，这样就可以将实物焊接在PCB板上。在PCB制板前还应做好布局布线，布局是为了确定某个元件放在板子上的某个相对位置。良好的布局还可以简化布线的过程。减少过孔的数量等。布线完成后就可以制板了。

3.7 本章小结

本章内容主要讲解了智慧农业嵌入式系统的硬件电路的设计。从最小系统到电源电路再到各类传感器电路。详细的分析介绍了为什么要设计外围电路以及外围电路是如何解决之前存在的一些问题。同时在硬件电路搭建好以后我们才可以书写软件代码。硬件是基础，软件是灵魂。关于软件部分我们在下章将详细介绍。

第四章 智慧农业嵌入式系统的软件平台设计

4.1 开发环境的搭建

软件部分的设计和使用首先要搭建好开发环境。而在搭建开发环境之前要对我们的需求做即我们需要用到些什么。回顾第二章的内容，本嵌入式系统主要是单片机的采集控制以及和云端的通信。那么我们的软件开发主要集中在单片机和WiFi模块上了。通过查阅资料后得知，WiFi模块可以通过AT指令和单片机通信，也可以自编程。如果选用AT指令的话，单片机需要准备好大量的格式正确的数据发给WiFi模块，这无疑增加了单片机的工作量和编程难度。如果选用自编程的话，我们有许多开源库可以使用，大大降低编程难度，且效率较高。数据处理更简单。缺点是单片机和WiFi模块都需要编程还需注意他们之间的衔接。最终我们选用WiFi模块自编程的方案且选用arduinoIDE来实现。

选定方案之后就需要选用操作系统了，目前市面上使用量比较多的操作系统有Windows、Linux、macOS。因为macOS只能安装在苹果电脑上。让很多人望而却步，故舍弃。而arduinoIDE在Windows和Linux下都有免费的软件可使用。对于WiFi模块来说选用什么操作系统并没有什么影响。而对于STM32来讲在Windows下官方推荐使用keil，当然我们也可使用IAR等其他软件。在Linux下编辑器可以选择sublime、eclipse、vi等。编译器只能安装交叉编译链arm-none-eabi-gcc，还需要安装jlink或其他的下载驱动。个人觉得两者并没有太大的本质区别。但为了其他人在后期维护的方便性。最终选用Windows系统加arduinoIDE、keilIDE来开发。

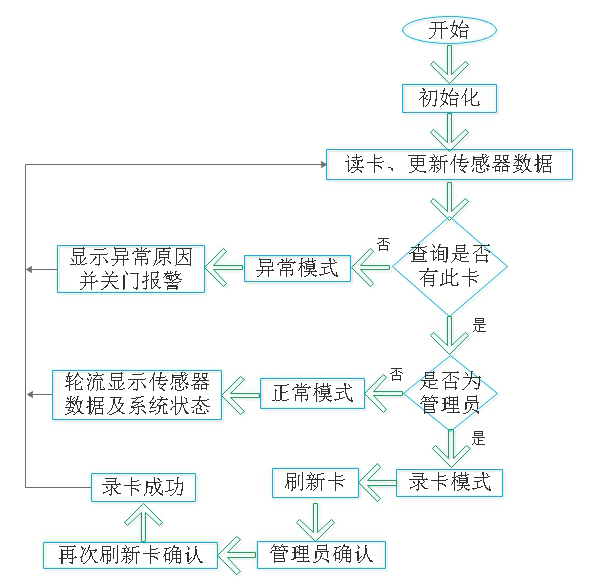
4.2 软件设计总体框架

图4-1 软件设计流程图

图4-1展示了智慧农业软件架构的总体流程（中断不在流程内）系统启动经过一系列初始化后就进入死循环。总体大概可分为三部分，第一部分为更新各类传感器的数据。第二部分为刷卡并根据得到的信息确定程序运行的模式。第三部分为不同模式下进行不同的控制反应。主流程就这三步，由图2-1可知数据从单片机传给WiFi模块。这一部分在工作在定时器中断中完成。每2.5秒传输一次。传输的数据在第一部分被更新。WiFi模块拿到新的数据后立马上传至云端。这部分后面会详细介绍。此处略过。

4.3 软件设计详解

上节给出了软件的总体框架为后续的代码立下一个方向标，本节将按照流程图详细介绍每个部分的具体实现过程。同时还会适当补充流程图中不曾给出的中断部分。

4.3.1 各类传感器的数据采集

在第一部分中传感器数据的采集包括温度、湿度、光强。对应的传感器分别是DHT11和TSL2561。当程序运行到此处时将会读取两个传感器三个物理量的值。

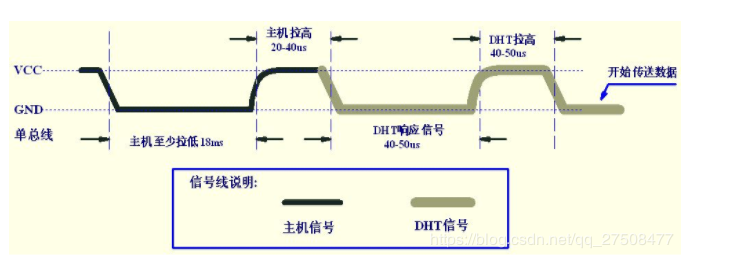
DHT11采用单总线协议和单片机通信。单总线协议可大致分为三个部分。初始化、写、读。但DHT11不涉及到写操作。初始化分为两个部分复位脉冲和从机应答信号。复位脉冲由主机拉低信号线至少18ms，然后再拉高20~40us。当DHT11检测到复位信号时，DHT11拉低信号线80us告诉主机准备就绪。然后再拉高80us。之后开始传输数据。

图4-2 单总线协议初始化

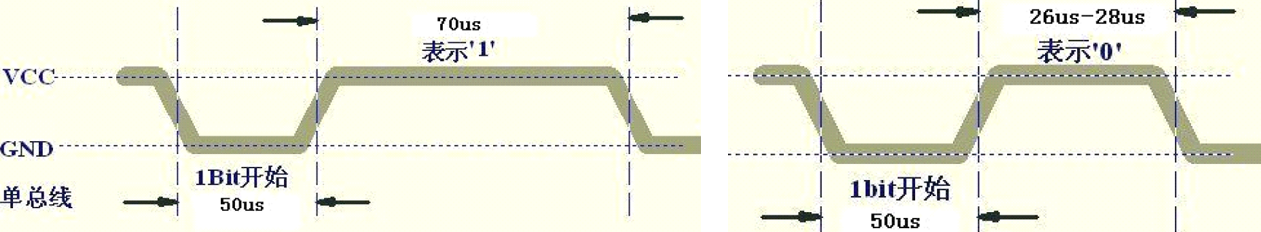
DHT11在拉高信号线80us后开始传输数据每传输1bit位需要有50us低电平的时间间隔，50us后拉高信号线。若高电平持续时间为70us则认为DHT11向主机传输逻辑1，若高电平持续时间为26~28us则认为DHT11向主机传输逻辑0。

图4-3单总线协议读操作

当主机需要读取温湿度时先通过信号线发出初始化信号。然后等待DHT11响应，之后就连续读取5byte数据，每个byte都是未经编码的二进制数据，第一位为湿度的整数值，第二位为湿度的小数值，第三位为温度的整数值，第四位为温度的小数值，第五位为校验和。当读取到5byte后如果校验没出错就将二进制的数据转化为十进制的数据临时保存在全局变量中，每隔2.5s取出通过串口发送给WiFi模块，WiFi模块会将此数据发送至云端。

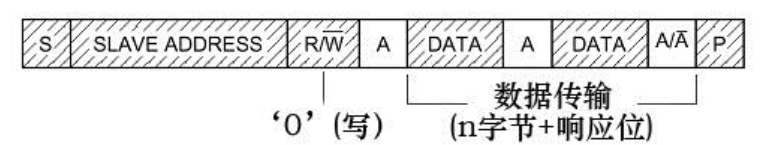
TSL2561采用I2C通信协议和单片机通信。I2C是一种半双工同步通信协议，物理层由SCL时钟线和SDA数据线。一组I2C信号线最多可同时挂载127个I2C从设备，本系统挂载了TSL2561和OLED显示屏两个从设备。和单总线一样I2C也有读写操作，不过没有初始化。但是多了起始信号。I2C写时序如图

图4-4 I2C写时序

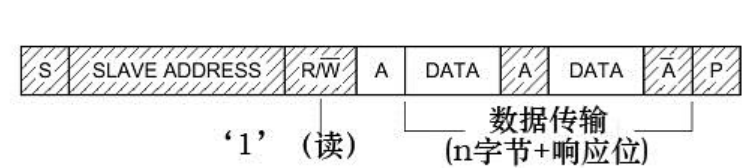
4-4所示。主机先发送起始信号，紧接着发送7位设备地址（这个地址是设备的唯一地址，用此 来区别是和那个设备通信）。再跟一位“读”、“写”位。0表示写，1表示读。如果有这个设备从机会产生一个应答信号。然后开始传输数据。每传输一个字节的数据从机都需要产生一个应答信号，以此来表示确认收到。最后主机发送停止信号来结束本次通信。I2C读时序和写时序非常类似，只是数据传输方为从机，应答方为主机。

图4-5 I2C读时序

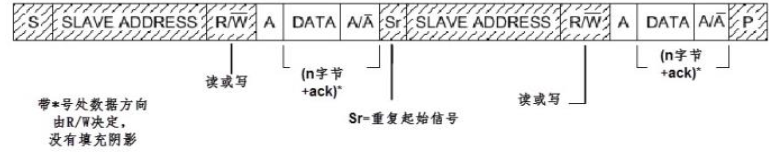
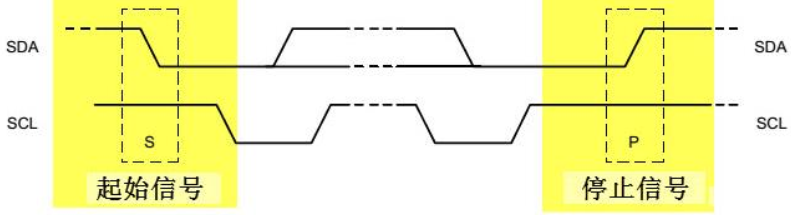
然而在实际应用当中我们单纯用读或写操作比较少，更多的是读写混合操作，如图4-6，仔细观察会发现其实就是单纯的读写时序拼接了一下，只不过

图4-6 I2C读写混合

在拼接的时候会把前面的停止信号给去掉了，其实也很好理解，因为停止信号代表本次通信的结束，当然只能有一个且在通信的末端。

上面简单的介绍了下I2C的协议层，那么协议层是怎样依赖物理层实现的呢？如图4-7所示。当时钟线SCL为高电平的时候，数据线SDA有一个下降沿则从机认为这是一个起始信号，同理在SCL高电平时，SDA有上升沿则认为是停止信号。I2C的数据传输如图4-8所示，在SCL为高电平的时候对SDA进行采样，若为高电平则认为此bit位为1，反之为0。在SCL为低电平的时候SDA

图4-7 起始停止信号

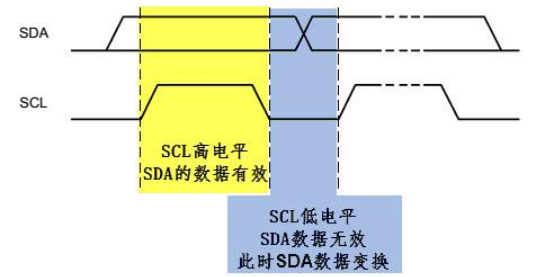
的电平状态无效，此时应该准备下一次希望传输的电平，也就是说在这个时候很有可能会产生电平的跳变，正因为SCL为低电平时不对SDA进行采样，此时的电平跳变不会影响传输的结果。应答信号就是在该应答的时候拉低数据线就可以了。

图4-8 I2C数据的采样

至此我们就可以通过I2C和光强传感器通信了。光强数据的计算比温湿度稍微复杂一点。首先我们要通过I2C读取TSL2561传感器的0x8C、0x8D、0x8E、0x8F四个8位寄存器的值，其中前两个值拼接形成一个16位的数据

ch0，后两位也拼接形成一个16位的数据ch1，根据表4-1查找出相应的公式

表4-1 光强计算公式

|  |  |
| --- | --- |
| ch1\*100/ch0 | 公式 |
| 0-50 | lx=0.0304\*ch0-0.062\*ch0\*((ch1/ch0)/4) |
| 50-61 | lx=0.0224\*ch0-0.031\*ch1 |
| 61-80 | lx=0.0128\*ch0-0.0153\*ch1 |
| 80-130 | lx=0.00146\*ch0-0.00112\*ch1 |
| >130 | lx=0 |

计算后得出光强值，单位为流明 。得到光强值和温湿度一样，暂时存在全局变量中等待传输时机。

4.3.2 刷卡确定工作模式

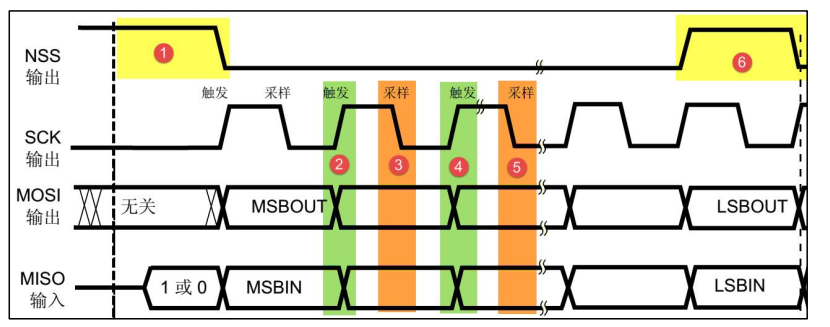
系统确定是否有非接触式IC卡靠近是利用RFID识别技术。我们具体用到的RC522这个传感器。RC522的工作比一般的传感器稍微复杂一点，不过大致可分为驱动层和应用层两部分。驱动层就是让这个模块能工作起来。应用层就是让这个模块怎么工作。先讲下驱动层吧，RC522和单片机的通信方式采用了SPI通信协议，它是一种全双工同步通信协议，和前面介绍的I2C类似但又有所不同，在物理层I2C有SCL、SDA两条线，而SPI有3+N条线，分别是SCK时钟线，MOSI数据输出线，MISO数据输入线。CS片选线。有N个设备就会有N条片选线。依赖于物理层的复杂，所以协议层相比I2C会简单点。如图4-9所

图4-9 SPI时序图

示。当单片机想要和SPI外设通信时，先拉低这个外设的片选线，只有在片选线为低电平的时候其他三条线上的电平变换才有意义。MOSI只负责数据从主机传到从机，MISO只负责数据从从机传到主机。他们互不干扰所以才可以做到全双工通信。和I2C有点区别的是SPI是在时钟线下降沿的时候对信号线进行电平的采样，在上升沿的时候信号线会选择跳变与否。但其实图中给出的只是SPI模式的一种，SPI共有4种模式，由时钟极性CPOL和时钟相位CPHA排列组合决定，具体见表4-2。图中的时序是模式2。

表4-2 SPI工作模式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SPI模式 | CPOL | CPHA | 空闲时SCK时钟 | 采样时刻 |
| 0 | 0 | 0 | 低电平 | 奇数边沿 |
| 1 | 0 | 1 | 低电平 | 偶数边沿 |
| 2 | 1 | 0 | 高电平 | 奇数边沿 |
| 3 | 1 | 1 | 高电平 | 偶数边沿 |

了解了SPI时序后就可以写出驱动层的代码了。然后就是RC522的应用层了。应用层简单点讲就是读出非接触式IC卡的唯一ID号，读ID分为两部分寻卡和防冲撞，寻卡就是查询RC522工作范围内是否有某类卡。具体操作就是往某些特定的寄存器写入某些特定的值，这点可以查阅官方datasheet也可以阅读代码。防冲撞就是防止卡号被重复读取，新老数据发生错位产生一个错误的ID号。具体操作也是往相关寄存器写入相应的值。

当系统没有检测到有卡靠近RC522的时候系统将直接跳过这一部分的工作，转而持续采集传感器的值，当有卡靠近时就会读取卡号。并将读到的新卡在预先存留的卡号库中查询。若没有查询到卡，这认为是误操作，或者非相关人员的恶意操作，简单讲就是拿错了钥匙。此时系统会被标记进入异常模式。如果查询到了卡号则认为用了正确的钥匙。系统会被标记进入正常模式，如果通过查询发现是管理员的卡号，则会进行判断，短刷就和正常卡没什么区别，系统将会进入正常模式，长刷3秒则会认为进入录卡模式。当执行完这三种模式后就会回到跟新传感器的状态。进行死循环。

进入异常模式会立刻关门，并且蜂鸣器会响30秒作为警报。然后回到正常模式。正常模式下将会开启恒温调控系统，就是预先设定一个范围。如果温度传感器测量到的温度值在这个范围内则不做任何动作，如果温度过低则开启供暖设备，反之如果温度过高则开启制冷设备。还需要在OLED屏幕上显示当前系统的运行状态和传感器的数据。OLED显示屏通过I2C和单片机通信。I2C在前面介绍过，就不在介绍。屏幕显示需要注意显示内容和位置。当识别到管理员的卡持续停留在RC522上超过3秒时就会进入录卡模式。进入录卡模式之后就会等待刷入新的卡，并且会在显示屏上显示新的卡号。此时再刷管理员卡。作为管理员授权。最后再刷入刚才的卡确认即可。这四个步骤一环扣一环，无论哪个环节出错都会使录卡失败。OLED屏幕上就会显示每一个过程作为提示，如果出错也会显示错误原因。整个过程系统预留了一分钟的时间。如果一分钟内没有操作完成的话也将录卡失败。录卡成功后新的卡将会被加入卡号库，新录取的卡号只有普通用户权限，管理员权限是在代码中写入的，一旦写入将不能更改，除非更换固件 。

本系统的主体大致就是上文介绍的，但是还有一个不可或缺的中断部分。本系统的中断大致可分为三种，ADC中断、串口中断、定时器中断。ADC中断是因为土壤湿度传感器需要采集电压，放在中断中能更好的提高传感器的采集效果，避免湿度值很长一段时间保持不变。串口中断主要是接收WiFi模块发送的数据，这个数据是由云端下发的，单片机接收到这个数据后就可分析以做出相应的动作，来实现云端下发的命令。还有就是定时器中断，系统中所有和时间有关的地方都需要定时器中断的参与，比如长刷管理员3秒能进入录卡模式，录卡模式一分钟内未完成就自动退出，每2.5秒就上传一次传感器的数据给云端等等。各种不同时间的定时全部依赖于一个定时器，这得益于各种flag变量的定义。需要上传的数据先封装成json格式然后再从串口传给WiFi模块就行了。

4.4 ESP8266 WiFi模块

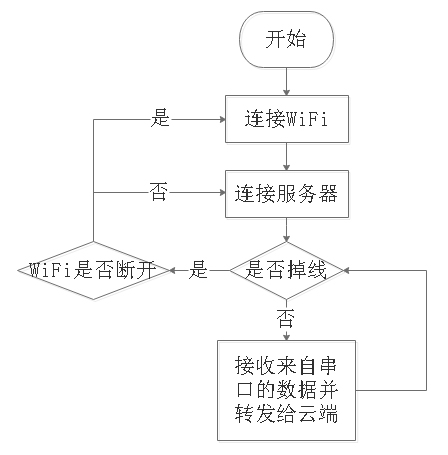
ESP8266模块是乐鑫公司推出的一种WiFi模块，可以连接WiFi后在2.4G信道上向云端服务器传输数据。本章第一节就简单的介绍了下ESP8266。并决定采用ESP8266自编程的模式，arduino是一款开源软件，可以很容易在网上获取安装包，安装好软件后需要导入三个库ESP8266WiFi、PubSubClient、ArduinoJson。前两个库可以让ESP8266连接网络，第三个库可以非常简单方便的解析json格式的字符串。

图4-10 ESP8266程序流程图

程序流程非常清晰，首先是连接好WiFi。这个用户名和密码是在程序中写死的，不能随意更改，当然也可改成NONE这样的话就会自动寻找没加密的WiFi进行连接。连接好WiFi后就可以访问互联网，这时我们就需要系统去访问服务器，能成功连上服务器之后就可以开始传数据了，由于网络中存在很多不确定因素，很有可能会发生连接断开的情况。所以必须在死循环中加入断开重连的机制，就是在循环体中不断检测是否掉线。如果是，则进一步怕判断是WiFi连接失败还是仅仅是服务器连接失败。不管是何种原因都要让系统能够重新连接上服务器，之后就是传输数据了，从前面的内容可知，单片机发送给WiFi模块的数据已经是一个符合要求json格式的字符串。那么WiFi模块就只需要简单的将这个字符串原封不动的通过网络转发给服务器就行。至于下发的命令，也非常简单，因为下发是不确定的，即谁也不知道什么时候会有什么命令下发，所以这个过程并不能在流程图中展示出来，就像单片机程序的中断一样，一旦WiFi模块接收到了来自服务器的下发命令就将数据提取出来转而通过串口发送给单片机。这样WiFi模块就成功的作为一个桥梁连接单片机和服务器。

第二章中介绍到了为了系统的操作方便我们还设计了一个移动控制终端，并且拥有电阻触摸屏作为输入，这个移动终端的实现其实运用了一个偷梁换柱的概念，就是屏幕会模拟网页发送一个信号给服务器，在服务器看来这就是网页发送来的消息，然后按照正常的流程实现下发命令。具体实现的方法是WiFi模块加电阻触摸屏，碰巧的是电阻触摸屏也是串口通信，所以当触摸之后通过串口发送命令给WiFi模块，WiFi模块将会模拟网页发送给服务器。当有数据上传的时候WiFi模块将会定时主动去读取服务器的值，并且通过串口更新屏幕上的显示。

4.5本章小结

本章主要介绍了智慧农业嵌入式系统的软件端的设计。宏观上详细介绍了软件总体框架、流程。微观上详细介绍了传感器是如何工作的，从驱动层的各种通信协议时序，到应用层各种数据的处理。再到所有传感器数据的统一处理调配。还详细介绍了WiFi模块的工作流程，数据具体是如何上传下发的，对移动终端也做了个简单的介绍。

第五章 总结

物联网的话题这几年是越来越热，物联网方面的发展也确实越来越好，日常生活中随处可见物联网的产品。而本文主要是将物联网在农业方面的一个应用。本文主要是利用STM32和若干传感器搭建的一个智慧农业系统。系统的主要功能是远程对农田实时状况的检测和控制。现在为实现这个功能所做的工作总结如下。

前期对市场分析，其中包括关于农业方面有哪些非常需要应用物联网技术的，哪些不是非常需要的。还有国内外目前在农业方面物联网技术发展状况，我国和西方国家到底还存在多大的差距。中期是对整体架构的设计，包括嵌入式端需具备哪些功能，需要解决哪些问题。云端该如何向用户展示怎么样一个友好的界面。以及解决嵌入式端和云端数据通信的问题等等。后期是最重要的部分了。前面所做的一切基本都是以理论为主。而后期的内容主要是根据前期的理论指导来做实际的内容，主要分为软硬件的设计与实现。在硬件方面，主要设计了STM32最小系统电路，各种传感器电路或传感器与单片机的接口，还有电源电路等。对每个电路的细节部分做了详细的解释，为什么要这样设计？这样设计的好处在哪里？都做了说明。软件方面是花较多时间编写和调试的一个步骤。首先要有一个良好的代码框架，这样不仅能使代码条理更加清晰，也便于代码的书写，还方便后期的维护。所以写代码之前先画出流程图，搭建好框架，剩下的就是反复的调试，不过本文不仅对实现功能的代码做出了解释说明，还对传感器功能的实现原理也做了部分介绍，其中包括各种通信协议的时序在代码中如何体现等等。

通过本文的介绍以及本文给出的示例，我相信以后物联网技术不仅在农业方面有所提升和发展，更是在生活中的方方面面都会有所涉及，因为这已经是一个趋势，就像前些年的互联网一样。是时代发展的必然。不过由于自身学术水平还不够高，眼界还不够宽阔，能力还不是很强。本文介绍的智慧农业嵌入式系统还是有一些不足之处。比如说在硬件设计方面PCB的布线还可以进行优化，甚至连传感器电路。传感器的选择都可以设计一个更优的出来。例如，本文利用WiFi模块传输数据，但是这样做的前提条件是附近必须拥有一个局域网，也就是说在农田安装一个路由器，这样整个系统使用起来并不是很方便。目前已有NB-IOT可以代替WiFi，好处是借助SIM卡流量来传输数据，无需局域网，不过这毕竟是一个新兴技术，还存在很多不稳定的条件，而且目前我国计划全面关闭2G网络，所以我们为保守起见还是选用WiFi。在软件方面，代码还可以优化还可以更加节省CPU资源，等等。

我相信随着物联网时代的到来，必能进一步的改善人民的生活。

附 录

WiFi模块向云端传输数据代码

#include <ESP8266WiFi.h>

#include <ArduinoJson.h>

#include <PubSubClient.h>

char ssid[] = "s008-flying";

char pass[] = "dianxin132";

#define greenhouseID "8e27efe0-eb81-11e7-8341-353f63eeab61"

#define greenhouseKEY "i8j8gFS00lwaNzSOxrEe"

char ServerAddr[] = "117.21.178.99"; //MQTT服务器地址

char StateTopicAddr[] = "v1/devices/me/telemetry"; //client.publish(StateTopicAddr, get\_gpio\_status().c\_str());上传地址

char ControllTopicAddr[] = "v1/devices/me/rpc/request/+"; //client.subscribe(ControllTopicAddr);连接下发地址

WiFiClient wifiClient;

PubSubClient client(wifiClient);

int status = WL\_IDLE\_STATUS;

void setup() {

// put your setup code here, to run once:

Serial.begin(115200);

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)//WiFi.status() ，这个函数是wifi连接状态，返回wifi链接状态

{

delay(500);

// Serial.print(".");

}

client.setServer( ServerAddr, 1883 );//链接服务器及端口

client.setCallback(on\_message);

}

void loop() {

// put your main code here, to run repeatedly:

if (!client.connected())//几个非连接的异常处理

{

reconnect();

}

while (Serial.available()) { // 来自stm32的数据

String s = Serial.readString();

client.publish(StateTopicAddr, s.c\_str());

delay(100);

}

client.loop();

}

void reconnect() {

while (!client.connected())

{

status = WiFi.status();

if ( status != WL\_CONNECTED)

{

// Serial.print("error wifi");

WiFi.begin(ssid, pass);

while (WiFi.status() != WL\_CONNECTED)

{

delay(500);

// Serial.print(".");

}

// Serial.println("Connected to AP");

}

else

{

client.setServer( ServerAddr, 1883 );

client.connect( greenhouseID, greenhouseKEY , NULL);

}

client.subscribe(ControllTopicAddr);//连接下发的地址

// Serial.print("begin"); // 回应stm32已经连接服务器

}

}

void on\_message(const char\* topic, byte\* payload, unsigned int length)

{

char json[length + 1];

strncpy (json, (char\*)payload, length);

json[length] = '\0';

StaticJsonBuffer<200> jsonBuffer;

JsonObject& data = jsonBuffer.parseObject((char\*)json);

if (!data.success())

{

// Serial.print("error json data");

return;

}

String methodName = String((const char\*)data["method"]);

String params = String((const char\*)data["params"]);

Serial.print(methodName+"\_"+params+"\_"+"\*");

delay(100);

}

# 参考文献

# 外文资料

# 中文译文

# 致 谢

时光飞逝，转眼间大学四年已过。这四年给了我太多的成长。从一个青涩的小男孩逐渐演变成年轻小伙子。在大学期间发生了太多太多的事，每一件都使我一步一步走向成熟。

随着毕业论文的结束，也就意味着大学生活走到尽头。在这里首先感谢我的母校。给我提供了一个良好的学习生活环境，其次要感谢我的知道老师黄友文博士。在项目上老师给出了很多建议，可以说有了老师的建议，项目才可以做到如此完善。在论文中，老师对论文的框架，重点进行详解，对论文的许多细节部分给了修改意见。没有老师的建议。就写不出现在这样的论文。

还应感谢我的家人，一直以来对我学业的支持与鼓励，感谢我的许多位同学、室友，在大学四年对我的关心照顾。我将不负大家对我的期望继续前行。

# 小论文

智慧农业系统嵌入式开发

陈嘉彬

（1.江西理工大学，信息工程学院，江西 赣州 341000；）