目录

摘　要

改革开放以来，我国的生活水平稳步提高，现在人们的生活品味与生活品质要求也日益增长，所以喜欢养花弄草的人越来越多。不过与社会发展一起而来的还有当今快节奏、高压力的社会生存环境，这就使得大家对于自己养护的花草经常心有余而力不足。很多植物对于空气湿度、空气温度、土壤湿度要求很高，疏于管理的花草很难拥有良好的生存环境，很容易发生枯萎，甚至死亡的情况。现在科技的进步发展愈来愈快，物联网传感技术的发展也愈加的成熟，人们对于生产生活中的各式物品进行监测和自动控制的想法也不再是天方夜谭，以往高大上的黑科技现在也可以很容易的走入寻常百姓家。除了面向个人用户，本设计还可以大规模应用于规模性的大棚农业种植中，能够大幅减轻种植户的人工成本。

本文设计了一款基于 STM32 单片机的空气温湿度、土壤湿度智能检测并进行自动调节的系统。本文首先阐述了生产生活中各式环境监测传感器的重要性，这些传感器在现代生产、生活中愈加不可替代[1]。紧接着本文针对上文提到的问题提出了系统的解决方案，此系统可以很好的完成植物生长环境中的空气温湿度、土壤湿度检测并对其进行控制。与同类产品相比，具有较多的优势。随后本文介绍了这个系统的硬件设计以及软件设计部分。利用传感器感知到的数据，STM32 芯片对于得到的数据进行分析和处理，并对继电器做出控制，以此来实现调节空气温湿度、土壤湿度的目的。最终实验表明，本系统能够正常工作，并且有很好通用性。

关键词：STM32；温湿度感知；智能测控系统

ABSTRACT

Since the reform and opening up, the living standard of our country has steadily improved, and now people's life taste and quality of life requirements are also growing, so more and more people like to grow flowers and grass. However, along with social development, there is today's fast-paced, high-stress social living environment, which makes everyone often have more than enough energy for their own conservation of flowers and plants. Many plants have high requirements for air humidity, air temperature, and soil moisture. It is difficult for a flower that is neglected to manage to have a good living environment, and it is prone to wilting or even death. Nowadays, the progress of science and technology is getting faster and faster, and the development of Internet of Things sensing technology is becoming more and more mature. The idea of monitoring and automatic control of various items in production and life is no longer a fantasy. The black technology can now easily enter the homes of ordinary people. In addition to targeting individual users, this design can also be applied to large-scale greenhouse farming in large scale, which can greatly reduce the labor costs of growers.

This paper designs a system based on STM32 MCU for intelligent detection of air temperature and humidity, soil moisture and automatic adjustment. This paper first describes the importance of various environmental monitoring sensors in production and life. These sensors are becoming more and more irreplaceable in modern production and life. Following this paper, a systematic solution to the above mentioned problems is proposed. This system can well control and control the air temperature and humidity, soil moisture in the plant growth environment. Compared with similar products, it has more advantages[1]. This article then introduces the hardware design and software design of this system. Using the data sensed by the sensor, the STM32 chip analyzes and processes the obtained data and controls the relay to achieve the purpose of adjusting the temperature and humidity of the air and the humidity of the soil. The final experiment shows that the system can work normally and has good versatility.

Key words：STM32; temperature and humidity sensing; intelligent measurement and control system

第一章 概述

1.1 文章结构

首先向大家介绍一下本文的文章脉络，设计把文章一共分为六个章节，第一个章节为概述部分，重点介绍了本设计的相关的产业现状以及与本设计相关的技术发展，还有本设计的意义。第二章主要介绍了本次设计的核心平台 STM32 以及与 Cortex-M3 架构，随后介绍了一些传感器相关的原理。第三章从宏观上主要从硬件方面对设计予以说明，让大家对于本设计有一个宏观上的认识和了解。第四章的内容作为具体的系统实现的讲解部分，其中也包括了本次设计中软件使用的开发环境。第五章为测试部分，向大家介绍了本设计在实际中的测试过程。第六章是对于本次设计的总结。

1.2 设计背景

1.2.1 产业现状

现代生活中，人们的生活水平日益改善，大家的生活品味、品质要求也日益增长，随之而来的喜欢养花弄草的人越来越多。凡事都有利有弊，伴随着人类社会高速发展的还有人们与日俱增的生活压力，这就使得大家对于自己养护的花草经常心有余而力不足。很多植物对于空气湿度、空气温度、土壤湿度要求很高[18]，疏于管理的花草很难拥有良好的生存环境，很容易发生枯萎，甚至死亡的情况。

现在我国的农业发展的很快，很多地方告别了以往的粗放式发展，开始发展经济效益更高的温室大棚等新型栽培技术，但是，在这些生产中主要还是依赖于人工。例如大棚灌溉、大棚的通风换气、大棚的温度控制等，都是依靠着人工操作来完成的[2]。

无论是个人的养植需求，还是快速扩张的农业发展，传统的植物种植培养的技术都无法满足需求，一种新型的植物种植技术已经迫在眉睫，以此可将人类从繁冗劳累的工作中解脱出来。

1.2.2 技术发展

现在科技的进步发展愈来愈快，单片机控制技术与物联网传感技术的发展也愈加的成熟，人们对于生产生活中的各式物品进行监测和自动控制的想法也不再是天方夜谭，以往只能出现在电视与报纸上的先进技术，现在也可以走进寻常百姓的家中。

近现代工业革命之前，制造业的动力主要依靠大型动物，奴隶，底层劳动力。随着蒸汽机的诞生，第一次工业革命开始。此后，人类的动力来源由动物和人力变成了蒸汽机和煤炭。底层社会的人们开始解放出来，由奴隶变成了自由人类。第二次工业工业革命主要是伴随着电力和内燃机的产生而兴起，就此人类迈进电器化。在最近的一次工业革命—第三次工业革命革命中，计算机诞生了，由此翻开了历史的新篇章。电脑实现了自动化运行，人们只需把要的事情编写好对应的程序，然后让程序开始运行，单片机的发明和发展对于第三次工业革命有着举足轻重的地位。从单片机的诞生到目前，已经过去了数十年之久。短短数十载，星云际会、斗转星移，单片机也从开始时期的 SCM（single-chip microcomputer）单芯片微型处理器发展成为 MCU（Microcontroller Unit）微控制单元，直至现在的 SOC（System on Chip）系统级芯片，使之成为很多行业的技术支撑。在发展的过程中，单片机也呈现出体积越来越小，容量越来越大，功耗越来越小，性能越来越高，而价格却越来越低的趋势。单片机技术就目前的现状而言，现在的单片机应用技术无论从技术层面，还是普及度来说都已经相当的成熟。

伴随着单片机技术一起发展的还有传感器感知技术，从开始到现在已经经过了很多年的发展。与单片机的发展类似，传感器技术从刚开始的结构型传感器（这类传感器属于第一代传感器，所以工作原理相对比较简单，主要是把结构参数的变化转化为可读的信号）发展到固体型传感器，目前在市面上流通最广，使用最多的也是这类传感器。比如热电偶传感器、霍尔传感器、光敏传感器等。传感器发展到第三步的时候就开始变的智能化了，传感器变得会“思考”。在未来，传感器还会继续大踏步的向前发展，总的趋势是体积微型化、功能多样化、还有最重要的，能够智能化处理。

无论是电力的发明，还是单片机技术的发展，甚至是传感技术的越来越成熟，都为现在本次的设计打下了坚实的基础。

1.3 设计意义

在植物的自然生长过程中，能够对其产生影响的要素有很多，其中具有代表性的比如空气湿度、环境温度、土壤湿度、土壤肥力、二氧化碳浓度等，都会对植物的生长产生很大的影响。对于人类而言，能够控制植物生长的要素越多，对于自然的依赖性就越少，甚至彻底摆脱靠天吃饭的历史。因为有了控制植物生长要素的技术，我们可以帮助植物发挥出最大的生长潜能，借此可以达到增产的目的。对于个人而言，平时因为快节奏的生活无法对自己种植的花草进行很好的照顾，时常发生植物因缺水枯萎的情形，本设计可以很好

的解决这个问题。自然界里很多植物对于自己的生长环境非常挑剔，稍微不如意就选择拒绝成长，更有甚者选择“死亡”。这类植物对于空气的湿度，环境温度的要求特别高，比如生活中常见的金针菇，本设计可以提供良好的解决方案。从社会的角度出发，我国作为传统的农业大国在这方便有着非常高的需求。本世纪以来，我国大棚种植也得到快速发展，所占农业比重越来越大。大棚种植的经济效益很高，但是需要的人力成本也很高，因为需要人来进行管理。本设计提供的解决方案对于此类状况同样适用。机器可以代替人们之前做的那些重复易出错的工作，极大解放提高了生产力。

1.4 本章小结

本章在 1.1 节介绍了整个文章的结构框架，简单介绍了每个章节的内容，在 1.2 节中介绍了本设计的应用背景和相关技术的发展，1.3 节中说明本设计的意义。这一章节让大家对本片文章和本设计有一个简单的了解，可以得知为什么要实现本设计，本设计的实现基础在哪里。下一章节中将着重介绍本设计设计的主要技术。

第二章 技术支撑

2.1 STM32 微控制器

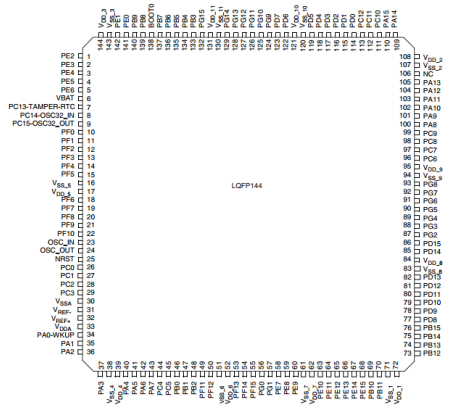

图 2.1 STM32F103ZET6 引脚图

本次设计主要是依托于意大利的 ST（意法半导体）公司的 STM32 平台[8]，处理器的具体的型号为 STM32103ZET6。在本次设计中，不需要处理器有特别高的性能，所以此款以 Cortex-M3 位内核的 F1 系列微控制器适用本次设计。现在很多的处理器价格都比较昂贵，设计中使用到的 Cortex-M3 内核的微处理器虽然是 32 位的处理器，但是它的价格却和 8 位机的价格差不多，低廉的价格、强劲的性能，使得它在同类产品的竞争中无往不利。本款处理器的接口丰富引脚有 144 个之多（如图 2.1 所示）。其它的资源还包括 64KB 的 SRAM、512KB 的 FLASH、SPI、IIC[3]、DMA、CAN[13]等。除了以上提到的这些，本款芯片具有性价比高，能耗低[20]，处理速度快等突出优势[11]，还有丰富的定时器资源并且支持 Flash 在线编程[14]，这些丰富的资源给本次设计提供了强大的保障。

2.2 ARM Cortex-M3 内核说明

STM32F103ZET6 采用的是 Cortex-M3 内核，通通俗点说，内核也就是 CPU。这款芯片的内核部分是由 ARM 公司设计的，其余部分称为片上外设由 ST 公司设计，如图 2.2 所示。Cortex-M3 相比于前代的产品而言有两个非常大的创新点。第一，芯片可以实现 CPU 进行数据访问的时候去取指令，两者可以同时进行；第二，芯片还新增了中断控制器，能够对中断进行更好的控制，大大延迟了中断的延迟。

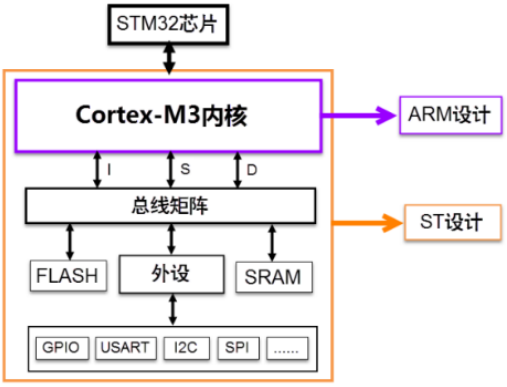

图 2.2 STM32 架构简图

如图 2.3 所示，Cortex-M3 的存储器映射从 0X0000 0000——0X0XFFFF FFFF，其中 0X0000 0000——0X1FFF FFFF 共计 512MB 为代码区，主要用于存储启动后缺省的中断向量表。0X2000 0000——0X3FFF FFFF 共计 512MB 为片上静态 RAM 区域。0X4000 0000——0X5FFF FFFF 共计 512MB 用于片上外设。0X6000 0000——0X9FFF FFFF 共计 1GB 的区域用于扩展外部存储器。0XA000 0000——0XDFFF FFFF 共计 1GB 的区域用于扩展片外的外设。0XE000 0000——0XFFFF FFFF 共计 512MB 的区域中包括了 NVIC 寄存器，MPU 寄存器以及片上调试系统。

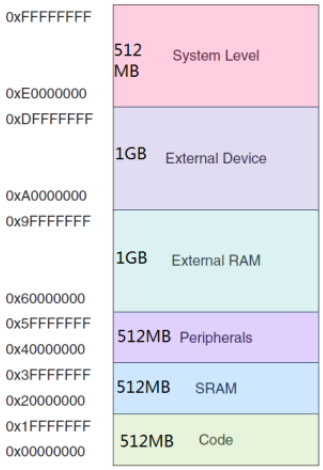

图 2.3 存储器映射

2.3 传感器技术

设计中主要感知的数据有温度和湿度，本小节简略的介绍一下这些数据的感知原理。DHT11 型传感器在测量湿度和温度的时候主要使用的是电阻式测湿元件和一个 NTC 测温元件。土壤湿度传感器使用的测量湿度的原理也是基于电阻式的，所以这里将它们放在一起来讲。在环境中，测湿元件在周围环境的湿度发生变化时，自身电阻阻值也会相应的发

生变化，我们就可以基于这一特性来对湿度进行测量。每种物质的感湿特性是不同的，根据物质材料的不同，大致可以分为高分子式、陶磁式以及电解质式，根据实际需求以及财务状况具体选用。在感知温度时主要使用的是 NTC 测温元件也就是热敏电阻，主要是由半导体材料来制成。使用起来方便，还具有低功耗的特性。但是这种元件的缺点也比较明显，在实际使用中会发现，它的温度和阻值不是呈线性相关的，所以元件的稳定性不太好。瑕不掩瑜，在本设计中它的功能完全满足使用，不失为一个很好的选择。

2.4 本章小结

在本章节中，介绍了本次设计的核心技术，STM32 微处理器以及使用到的传感器技术。单片机技术发展至今，资源越来越丰富，通用性越来越高，而价格越来越低廉。作为集大成者 STM32 单片机为本次设计提供了一个良好的支撑，2.1 节中简单介绍了 STM32 有哪些重要的资源，2.2 节重点介绍了一下 Cortex-M3 内核。内核是 STM32 的核心部分，所有的数据处理，都是基于它来完成，想要灵活的使用这款芯片必须要了解它的架构以及地址资源。所以，本章的重点放在了内核部分，需要着重了解。芯片的处理性能再好，没有传感器感知到的数据也是英雄无用武之地，在 2.3 节中介绍了本次设计中使用到的温湿度传感器的基本原理。前两章节的背景介绍完毕，下面正式进入设计部分。

第三章 总体方案设计

3.1 系统设计总体说明

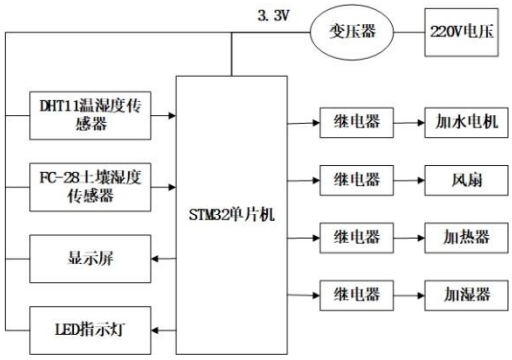

图 3.1 系统整体架构图

在本次设计中，大致可以划分为中心处理单元（STM32 单片机控制器）、感知单元（DHT11 型温湿度传感器、FC-28 土壤湿度检测器）、电源模块、显示模块以及基于继电器的加水电机、风扇、加热器、加湿器等具体执行模块，如图 3.1 所示，还包括 LED 指示灯[19]，设计的实物图如图 3.2 所示。

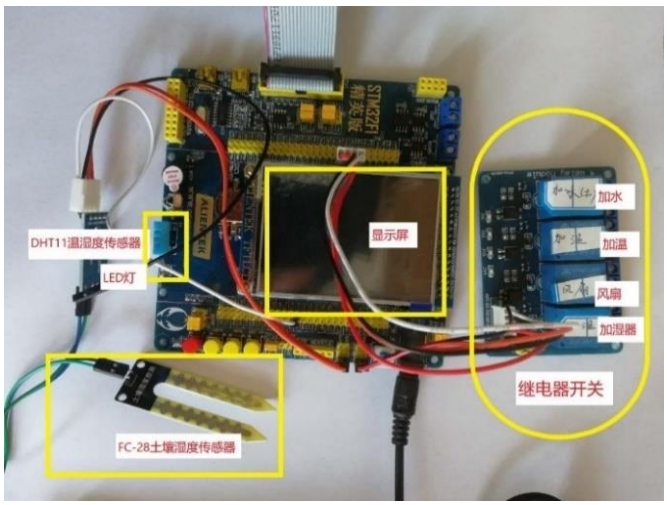

图 3.2 系统实物图

系统上电之后，整个系统开始初始化、运行。DHT11 型温湿度传感器负责检测植物周围的空气中的湿度、温度[10]，通过 GPIO 接口实时将数据传输到 STM32 主芯片内，并显示 LCD 屏幕上。与此工作道理相同的是 FC-28 型土壤湿度传感器，此模块的主要功能是实时检测植物生长环境中的土壤湿度[6]，随后将采集到的湿度信号通过 GPIO 接口将数据传输至 STM32 主芯片内，实时采集的土壤湿度显示在 LCD 屏幕上。LED 指示灯的功能是芯片

内程序在运行时 LED 指示灯不断闪烁，这是我们可以知道系统在正常工作。继电器组为主要执行模块，由继电器组控制的风扇、加热器、加湿器在通常情况下，它们的工作电压是220V。而本次设计中使用到的加水电机的工作电压就小巫见大巫了，为 12V。这是在个人用户的条件下，如果在面积更大的农业生产中，可以将此工作电压换成 220V，以此驱动更大功率的加水电机。此设计中的控制条件可以根据实际具体修改，对不同的环境、需求进行个性化的定制。本次设计仅仅针对于某些情况，不具有不可更改性。不同的植物、农作物的生长环境也各不相同，视实际情况灵活变通。

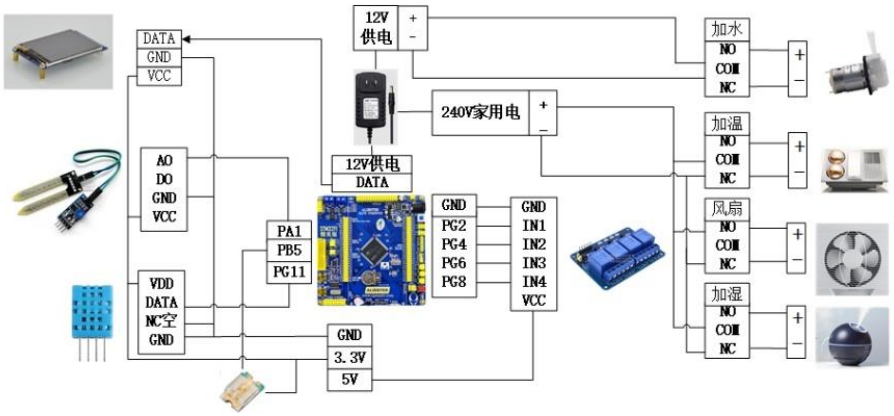

图 3.3 系统硬件接线图

系统的硬件连接如图 3.3 所示，针对于具体的模块单元接下来将会对其一一说明。

3.2 感知单元

3.2.1 DHT11 温湿度传感器模块

图 3.4 DHT11 型温湿度传感器

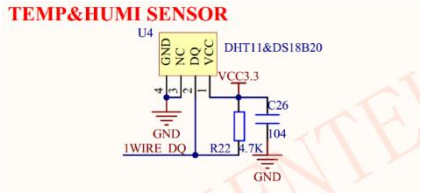

图 3.5 DHT11 型温湿度传感器接线图

本设计中选用的 DHT11 型传感器是一款应用范围广、技术应用成熟的传感器，具有性价比极高、抗干扰能力强、反应灵敏、性能有保障等优点[7]。DHT11 型传感器采集温度主要依赖于其中 NTC 测温元件[15]。

在上面两幅图中，左边的图片图 3.4 是温湿度传感器的实物图，右边的图片图 3.5 为

温湿度传感器的电路图。温湿度传感器有孔的一面将其当做是正面，也就是图片中显示的状态，在此情形下四个引脚可将其从左往右依次编号 1、2、3、4。在实际的电路连接中，1、2、3、4 号引脚在电路中的排列顺序与之刚好相反，应当从右往左排列，这点应当注意，接下来的借号主要以图 3.5 的顺序以及标号来介绍。1 号引脚为供电端，连接 3.3V 电源，为传感器供电。在电源引脚旁有一个电容，可以滤波去耦合，保护传感器信号的稳定传输。传感器的 2 号口为数据口，与处理器的 PG1 口连接，旁边的电阻为上拉电阻，设计中，电阻选用 5K 即可。最后，3 号口为 NC（Not Connect）口，设计中将其接与 4 号口 GND。

3.2.2 FC-28 土壤湿度传感器

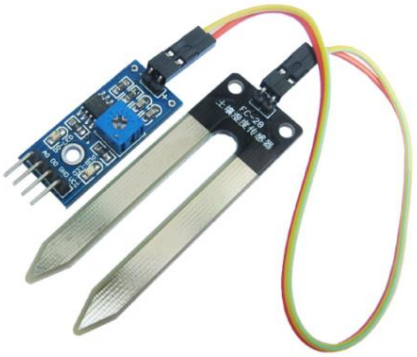

图 3.6　FC-28 实物图

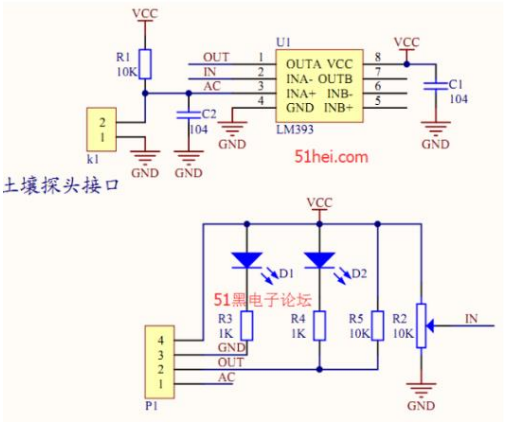

图 3.7　FC-28 电路图

FC-28 的实物图如图 3.6 所示，设计中选用的接口为 4 线制，如图 3.7 所示，实际使用中 VCC 连接在 3.3V 电源上，GND 接地，数据线选用 AO 口连接到处理器的 PA1 口，DO 接口悬空。DO 口输出的数字量，AO 口输出的是模拟量，本设计中选用 AO 口来采集模拟量。两个比较长的是插片是探头端，插在土壤里，随后根据事先的测出的土壤在不同湿度下的电阻值。传感器的主要原理相当于的是测量被测物的电阻，湿度越大电阻越小，反之也就不越大。然后通过实际使用中的比对测量值来根据电压测算出此时的湿度值。图中的 LED 灯 D1 是用来看电路是否正常工作的。

3.3 执行模块

本设计中选用的继电器为带有光耦的继电器，具有较好的抗干扰能力和稳定性。这次使用的继电器将四路继电器放在同一块板子上，共用控制端的 5V 电源，虽然如此，它的每一路可以独立工作互不干扰，继电器的工作电压为 5V。继电器的 IN1、IN2、IN3、IN4

分别于处理器的 PG2、PG4、PG6、PG8 端口连接，控制着加水电机[17]、加温装置、风扇、加温装置。继电器所连接的加水电机、加温装置[16]、风扇、加湿器的一端为实际工作端。这个端口一共有三个接口，从上到下是 normal open（常开）、common ground （公共端）、normal close（常闭）。常开端口 NO（normal open）和常闭端口 NC（normal close）连接工作装置的正极线路，工作装置的负极连接 COM 公共端。电路的连接如图 3.8 所示，设计中电机供电电压为 12V，其余电器供电电压为 220V。本次选用的继电器低电平有效，满足运行条件时 GPIO 口释放低电平信号，继电器导通，继电器导通之后 NO 和 COM 端之间就会导通，达到工作的目的。每一路的继电器都有一个 LED 指示灯作指示，继电器只要导通工作， LED 灯就会点亮，提示用户电源导通。

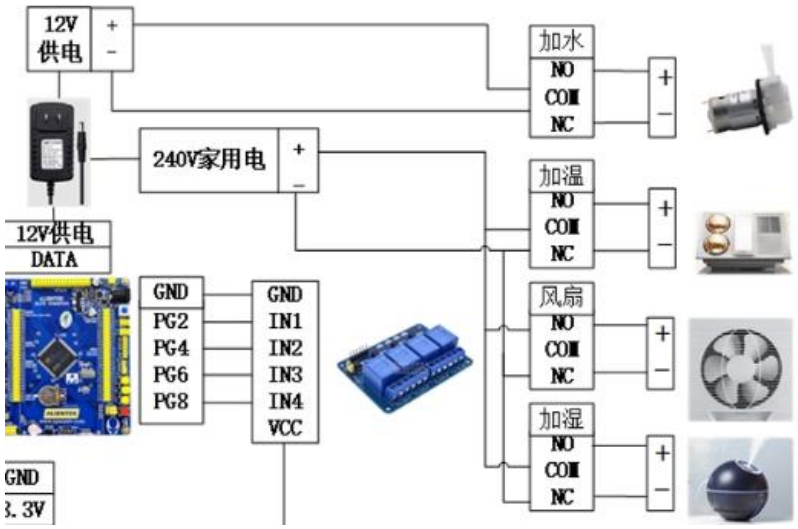

图 3.8　执行模块连接图

3.4 电源模块

在本设计中，电路板上的弱电部分使用到的工作电压主要有 3.3V 和 5V，这两个工作电压由电源模块转换而来。如图 3.9 所示，首先外部强电 220V 家用电使用变压器给整个电路从最左边 POWER 给电路提供 12V 的 DC 直流电源。接着电流来到 MP2359 直流降压转换器，经过降压处理获得 5V 工作电压。随后，继续讲 5V 电压输送到 AMS11117-3.3 稳压芯片，5V 的输入电压就可转换为 3.3V 的单片机供电电压。至此，设计中所需的 12V、5V、3.3V 电压都成功获得。

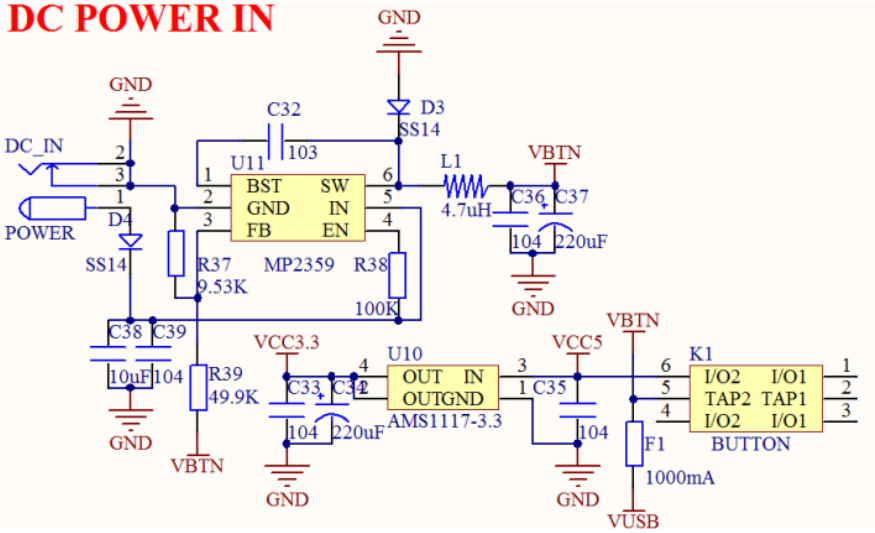

图 3.9 电源

3.5 显示模块

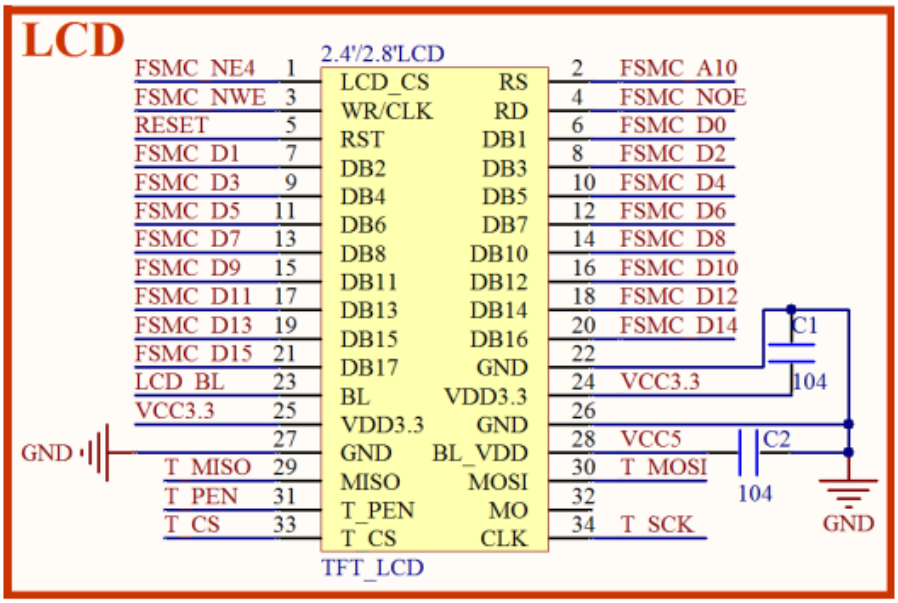

图 3.10 LCD 模块结口

LCD 显示屏的显示控制主要需要如下几个信号线（如图 3.10 所示）：

① 图中的 1 号接口，片选信号线 CS 连接在芯片端口 PG12。

② 图中的 3 号接口，用于数据写入的数据接口 WR 连接在芯片端口 PD5。

③ 用于复位的 5 号引脚，液晶显示屏工作的时候还需要复位，这个引脚与电路的整体复位公用同一个复位，在系统进行复位的时候 LCD 也会进行复位。

④ 命令和数据标志位选择接口 2 号接口，RS 信号线用来选择运行的时候读取数据还是读取指令，与 MCU 的 PG0 口相接。图中的 23 号引脚连接在 MCU 的 PB0 上，

这个引脚的功能是用来控制 LCD 的背光的。

3.6 其他

3.6.1 复位电路

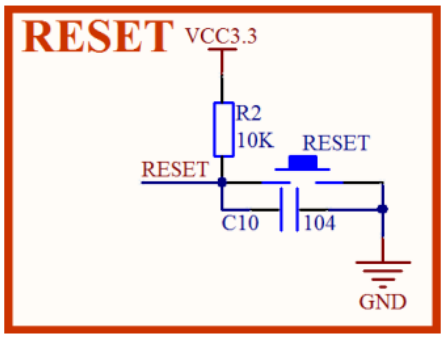

图 3.11　复位电路结口

STM32 是采用低电平复位的，设计中，我们把芯片的复位引脚中间间隔一个按键接地。如图 3.11 所示，R2 和 C10 构成的是上电复位电路，在上电的时候 MCU 会自动复位，按键可以进行手动复位。

3.6.2 LED 灯

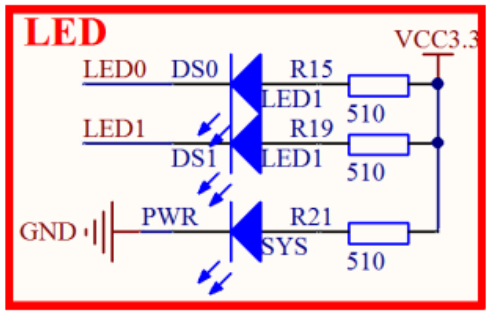

图 3.12 LED 线路

DS1 为绿色的 LED。本次设计中只使用到了 DS0[9]和 PWR 这两个 LED 灯，DS0LED 灯的颜色为红色，接在单片机的 PB5 引脚，主要在程序运行的时候闪烁，提醒用户程序运行正常。PWR 是最小系统的电源指示灯，指示灯亮的时候，表示电路电源正常[9]。

3.7 扩展补充

因为本次设计的经费和时间有限，原先计划中的二氧化碳浓度传感器和 PM2.5 监测传感器未能如预期装配，后期时机成熟再对本设计进行完善。在开始的规划中，CO_2 传感器可以实时检测植物生长环境的 CO_2 浓度情况。众所周知的，CO_2 含量过高和过低对植物的生长都不好。所以在植物生长环境中 CO_2 过高的时候，我们会打开排风扇通风。这就带来了另外一个问题，现在的雾霾特别严重，当我们为植物通风的时候，雾霾也会闯入，进而灰尘会落满植物影响植物生长。所以，在设计中加入了一个 PM2.5 检测装置，假如空气不好，为植物通风的操作将不能执行。下面对于这个补充的内容作一下说明。

3.7.1 CO_2 传感器

如图 3.13 所示为 MG811 型 CO_2 气体传感器，这款传感器受温湿度的影响较小。但是需要注意的是，在实际使用之中，这款传感器的需要至少两个小时的预热时间。这款传感器的工作电压为直流 DC 电源 6V，工作电流是 200mA，检测的范围最多至 10000ppm（parts per million）。

图 3.13 MG811 型 CO_2 气体传感器

如图 3.14 所示我们可以看出，浓度越大传感器的 EMF（输出电势）越小，根绝这个近似线性关系我们可以测量出 CO_2 的浓度。

MG811 传感器与主机相接的主要有三条线，如图 3.15 所示，2 号、3 号、5 号线。Dout 为电平信号输出口，Aout 为模拟输出口与 STM32 的 ADC 口相接。这里要特别说明的是 5 号 Tcm 口，在传感器工作的时候传感器内部温度非常高（500℃），温度过高的情况下会影响测量值，所以要加温度补偿。

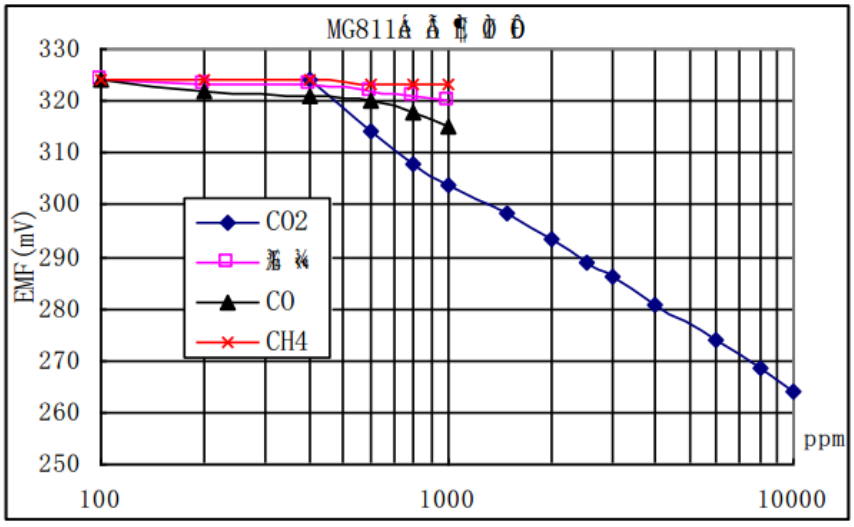

图 3.14 MG811 型 CO2 气体传感器灵敏特性

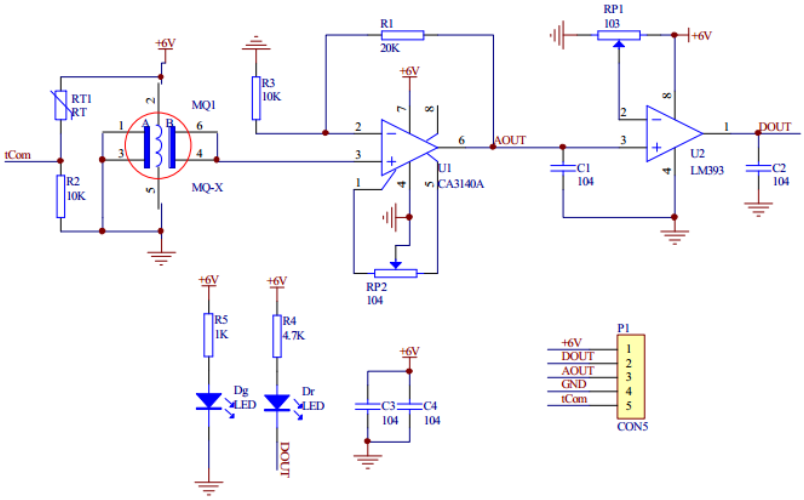

图 3.15 MG811 模块电路图

3.7.2 PM2.5 传感器

图 3.16 PM2.5 传感器

如图 3.16 所示即为灰尘传感器 GP2Y1010AU 的实物图，可以用来检测空气中的粉尘含量。如图 3.17 所示为 GP2Y1010AU 粉尘浓度特性例，根据此特性图，我们可以从输出电压中了解到现在粉尘浓度。

在实际检测中，还有一个比较重要的问题，就是我们应该如何区分形如香烟这种燃烧型的烟雾以及真正的灰尘，款传感器给我们提供了一种比较好的解决方案。如图 3.18 所示，我们知道，香烟的烟雾颗粒状态分布比较均匀且浓度较高，所以传感器会连续的输出特性电压；而灰尘就不同了，灰尘的颗粒比较分散，所以它的特性电压也是断断续续的，这样我们就可以对灰尘和烟做一个比较好的区分。

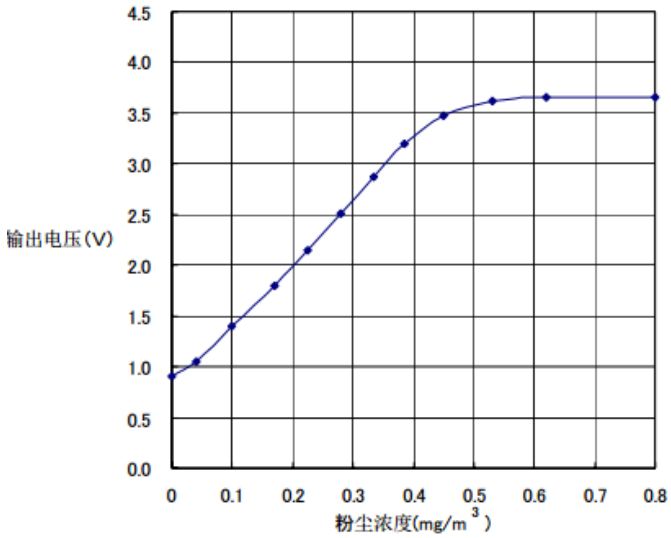

图 3.17　GP2Y1010AU 粉尘浓度特性例

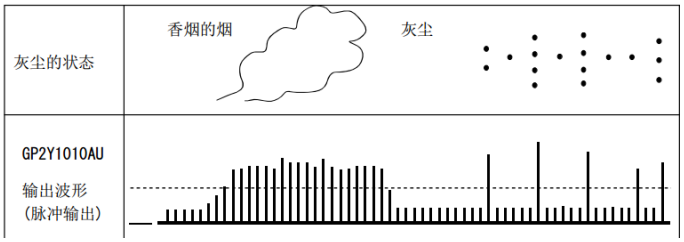

图 3.18　PM2.5 传感器反馈烟与灰尘输出波形

3.8 本章小结

本章是重点章节，花费了较长的篇幅来做说明，也花费了较多的时间和精力，这个章节主要是针对于硬件部分。本章大致可以分为两个部分，3.1 节至 3.6 节为一部分，这部分

是功能已经实现的部分，3.7 节中对于本次设计做了一点补充，因为时间和经费的原因，这部分没有实现，所以作为扩展补充来说明。

在已经实现的部分中，本章刚开始的 3.1 小节文章对设计作出了一个宏观整体的说明。介绍了本设计的工作方式和具体的硬件线路连接，在有了宏观了解之后，随后的几个小节将设计整体拆分成为感知单元、执行模块、电源模块、显示模块以及其他来详细说明。3.2 小节感知单元中包括 DHT11 温湿度传感器和 FC-28 土壤湿度传感器。3.3 小节执行模块重点说明了继电器的工作，以及通过继电器来控制的具体工作单位，因为本次设计中具体工作单位都是生活中随处可见的产品，文中不做赘述。3.4 小节电源模块解释了如何为电路提供 3.3V、5V、12V 的工作电压并对其电路进行展示。3.5 小节显示模块中介绍了本次用来进行实时显示数据的液晶屏，着重介绍了显示屏的几个重要接口。3.6 节中主要是复位电路与 LED 等，这部分功能比较原理比较简单不在赘述。以上就是本次设计中已经实现了的部分。

介绍完实现的部分，再简单说一说扩展补充的部分。这部分主要涉及 CO_2 传感器与 PM2.5 传感器。在 3.7 节开头部分，对这两个传感器的工作方式实现功能进行了说明。接着针对每一款具体的传感器进行了详细的介绍，在后续的功能拓展中提供一个比较好得参考。

这个章节中主要涉及的是硬件部分，硬件部分介绍完成，下面就该进行系统软件的介绍了，接下来的一个章节将对设计的软件部分进行说明。

第四章 系统实现

4.1 系统程序流程图与实现说明

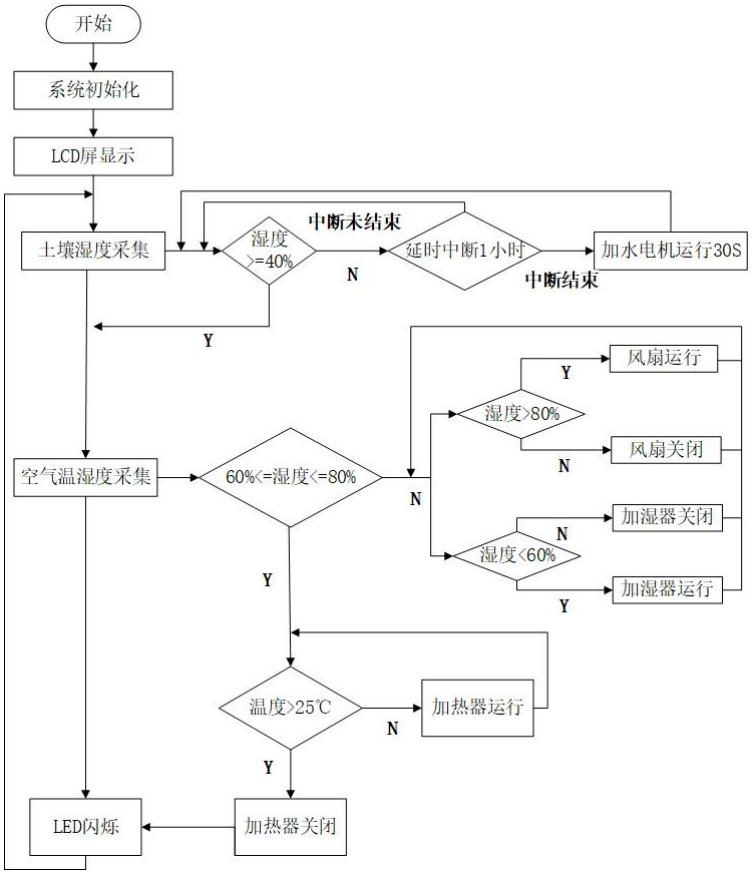

图 4.1 系统程序流程图

上一章节中，比较系统的介绍了本次设计的各个工作部分的相关硬件。在这一章节中，将系统地介绍上面的这些硬件是如何具体工作运行的，软件部分给硬件赋予了灵魂。系统上电之后，首先要做的就是进行系统初始化，点亮显示屏，紧接着 DHT11 温湿度传感器、土壤湿度传感器就要发挥其感官的功能，去感知周围环境中的变化，为 STM32 的工作提供情报讯息。STM32 读取到采集到的土壤湿度，实时显示在屏幕上，并且 STM32 芯片根据读取到的湿度值判断。当土壤湿度低于 40%的时候，打开控制加水电机的继电器，加水电机通电。通电后加水电机工作 30S，无论此时加了多少水，土壤湿度如何，与加水电机相连的继电器都要断开，停止继续加水。随后控制加水电机端的 GPIO 口停止工作 1 小时。在此期间，加水装置无论如何都不会工作。因为加水后土壤吸收水分需要充足的时间，这样的话土壤湿度传感器探测出来的值才比较准确。如果土壤湿度大于等于 40%，加水电机不工作。接下来程序开始采集空气温湿度，实时检测空气温湿度，检测到的数据实时显示

在屏幕，当空气湿度低于 60%时加湿器端的继电器闭合，加湿器通电开始启动，在加湿器工作的同事，湿度传感器也不会闲着，要不断的采集空气湿度讯息，如果湿度值高于或等于 60%加湿器停止工作。当空气湿度大于 80%时，风扇启动，对室内进行除湿操作，湿度低于或等于 80%时风扇不工作。室内温度正常需要维持在 25℃以上，当环境温度低于 25℃时，加热装置启动。温度达到 25℃或者以上时，传感器将讯息传送到处理器，处理器发出应对指令，加热器自动断电停止工作。在程序快要结束的时候，LED1 闪烁，当 LED 闪烁的时候说明程序运行正常。

4.2 开发环境的介绍

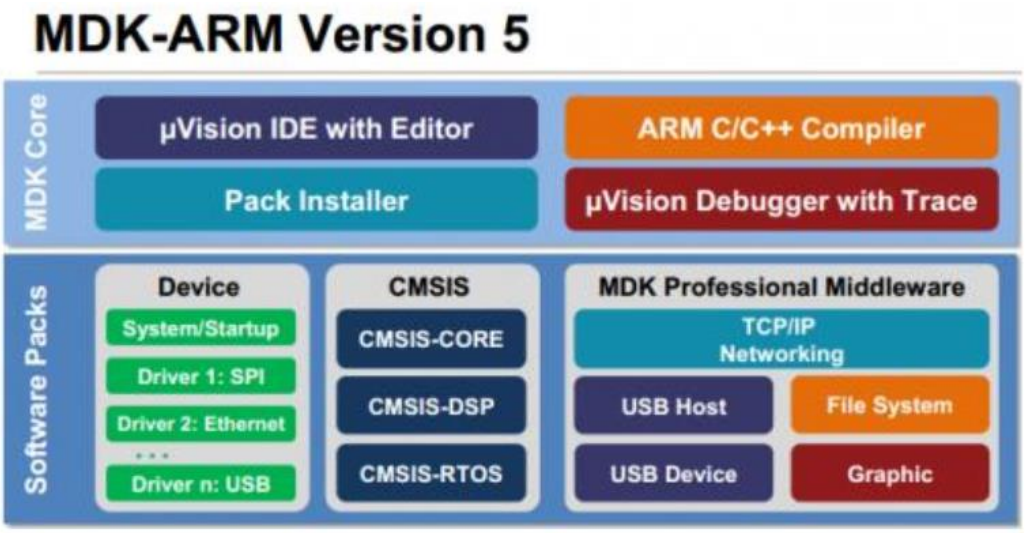

图 4.2 MDK5 组成

本次使用的开发软件是 Keil5，就目前来说，这款软件算是市场中应对 CM 系列芯片编程的最好的软件了。这款软件具有强大的兼容性，在以往的环境 Keil4 或者 keil3 上编写的程序在这款软件上仍然可以继续开发。除此以外，这款软件支持在线升级，这里所说的升级是指升级硬件资源库，这就为开发工程师节省了很多的时间，大大缩短了开发周期。

4.3 主程序

主程序分为两个部分，前半部分是变量声明，以及系统和各个模块单元的初始化。后半部分基于一个死循环，为主要的功能实现部分，系统使用传感器不断对空气温湿度以及土壤湿度信息进行采集，然后将采集到的数据传回 CPU，CPU 根据响应条件对执行模块进行调用。下面将对程序的函数与实现逻辑作出详细的说明。

```
#include "led.h"
#include "switch.h"
#include "delay.h"
#include "key.h"
#include "sys.h"
#include "lcd.h"
#include "usart.h"
#include "dht11.h"
#include "adc.h"
#include "timer.h"
int sum1 = 0;              //用于加水定时器延时的计数
int sum2 = 0;              //用于加水定时器加水时间的计数
int main(void)
{
  float temp;
  double result;
  u16 adcx;
  u8 t=0;
  u8 temperature;
  u8 humidity;
  delay_init();                    //系统延时函数的初始化
  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
  uart_init(115200);
  LED_Init();                      //LED 初始化
  LCD_Init();                      //显示屏初始化
  SWITCH_Init();
  Adc_Init();
  DHT11_Init();
  TIM3_Int_Init(4999,7199);        //延时定时器初始化，0.5 秒
  TIM4_Int_Init(4999,7199);        //加水时长定时器设置为 0.5 秒
  POINT_COLOR=BLUE;
```

```
LCD_ShowString(30,50,200,16,24,"sudawenzheng");
LCD_ShowString(30,80,200,16,24,"qiubo");
LCD_ShowString(30,110,200,16,24,"ADCVAL:          ");
LCD_ShowString(30,140,200,16,24,"SoilHumi:      %");
LCD_ShowString(30,170,200,16,24,"AirTemp:      C");
LCD_ShowString(30,200,200,16,24,"AirHumi:      %");
```

以上为前半部分，完成了初始化系统与硬件的初始化工作。

需要说明的是其中 TIM3_Int_Init(4999,7199)为定时器初始化，其基本原理将在下面定时中断程序中进行说明。屏幕显示函数 LCD_ShowString(30,50,200,16,24,"sudawenzheng")中，(30,50)为起点坐标，数字 30 表示是横坐标，纵坐标为数字 50。(200,16)确定了名目显示区域大小，200 是区域长度，16 是区域宽度。24 为字体大小，最后显示的是字符串内容。以下至结尾的程序为主程序的后半部分，也是主要的功能实现部分。

```
while(1)
{
    if(t%10==0)                                    //设置读取的周期频率为 100ms
    {
        DHT11_Read_Data(&temperature,&humidity);   //对空气的温湿度进行读取
        adcx=Get_Adc_Average(ADC_Channel_1,10);    //读取土壤湿度
        LCD_ShowxNum(110,110,adcx,4,24,0);
        temp=4096-adcx;
        result=(float)temp*(100.0/2076);
        LCD_ShowxNum(140,140,result,2,24,0);       //显示土壤湿度值
        LCD_ShowNum(135,170,temperature,2,24);     //显示空气温度
        LCD_ShowNum(135,200,humidity,2,24);        //显示空气湿度
    }
```

设计中将数据采集的时间间隔设置额为 100ms，也就是每 100ms 采集一次温室度数据。随后将采集到的数据显示在 LCD 显示屏上。在实际试验中对土壤湿度传感器进行 ADC 采集，发现 ADC 的值一直稳定在 2020 到 4096，且湿度越大值越小。所以根据测量到的值来度用公式 temp = 4096-adcx，result = (float)temp*(100.0/2076)。来计算土壤湿度的百分比。采集到土壤湿度值、空气温湿度值利用 LCD_ShowNum()函数显示在显示屏幕上。

```
    if(adcx>3265) //土壤湿度小于 40%
```

```
{
    if(sum2==60)        //30 秒加水
    {
        SWITCH0   = 0;
        sum2 = 0;
    }
    if(sum1==7260)        //每隔一小时检测一次
    {
        SWITCH0   = 1;  //加水装置关闭
        sum1=0;
    }
}else
{
    SWITCH0   = 1;       //加水装置关闭
    sum1   = 0 ;
    sum2   = 0 ;
}
```

由上文可知 adc 的采集值为 2020 到 4096，[4]所以根据这个值我们算出当湿度需要达到 40%的时候，此时 adcx 的值为 3265。湿度越大，adcx 越小，当 adcx>3265 的时候也就是土壤湿度小于 40%。满足触发条件后，加水电机启动，加水 30S。 sum2 用于加水时长的计时，加水完成后要进行清零。加水延时操作计数 sum1 的原理与此相同。

```
if(temperature<25)    //空气温度低于 25
{
        SWITCH1 = 0;   //加温装置打开
}
else{
        SWITCH1   = 1;   //加温装置关闭
}
```

本段为加温装置的控制段，触发条件时空气温度小于 25 摄氏度启动。SWITCH1 等于 0 的时候为开启状态，等于 1 的时候为关闭状态。

```
if(humidity>80)    //空气湿度高于 80
```

```
        {
                SWITCH2 = 0;    //风扇打开
        }else
        {
                SWITCH2 = 1; //风扇关闭
        }
```

本段为排湿装置风扇的控制段，触发条件是空气湿度大于 80%的时候，风扇会开启，进行排湿操作。

```
        if(humidity<60)        //空气湿度小于 60
        {
                SWITCH3 = 0; //加湿器打开
        }
        else
        {
                SWITCH3 = 1; //加湿器关闭
        }
```

本段为加湿装置加湿器的控制段，触发条件是空气湿度小于 60%的时候，加湿器会开启，进行加湿操作。

```
        delay_ms(10);
        t++;
        if(t==20)
        {
            t=0;
            LED0=!LED0;
        }
    }
}
```

上面最后一段是点亮程序运行指示灯 LED，指示灯被点亮闪烁，表示程序一直处于开启状态。

4.4 空气温湿度采集程序

空气温湿度采集主要依托于 DHT11 型传感器，此款传感器为单总线型，所以有一根数据线就可以，设计中数据线接在 PG11 口。

```
void DHT11_Rst(void)
{
        DHT11_IO_OUT();         //IO 口设置为输出
        DHT11_DQ_OUT=0;         //拉低输出口
        delay_ms(22);           //主机信号拉低至少时间为 22ms
        DHT11_DQ_OUT=1;         //输出口拉高
        delay_us(30);           //主机持续输出 30us 高电平信号
}
```

本段代码 DHT11_Rst 函数功能是 DHT11 温湿度传感器复位，在 DHT11 开始工作之前主机要发送开始信号，如图 4.3 所示。T1 与 T2 黑色部分是主机信号，先拉低数据线至少 18ms，程序中的设计为 22ms。随后要将主机信号拉高 20us—40us，程序中本部分设置为 30us。到这里，DHT11 复位结束。

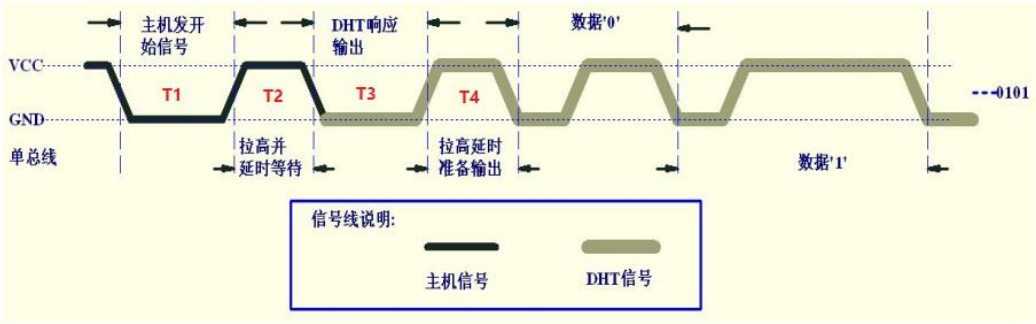

图 4.3 数据传输

```
u8 DHT11_Check(void)
{
  u8 retry=0;
  DHT11_IO_IN();                    //将 DHT11 设置为输入端口
  while (DHT11_DQ_IN&&retry<100)    //如果 DHT11 低电压 40us~80us
  {
        retry++;
```

```
        delay_us(1);
    };
    if(retry>=100)return 1;
    else retry=0;
    while (!DHT11_DQ_IN&&retry<100)  //DHT11 是否高电平 40us~80us
    {
        retry++;
        delay_us(1);
    };
    if(retry>=100)return 1;
    return 0;
}
```

在主机发出复位信号后主机要等待 DHT11 的反馈信号，本段代码 DHT11_Check 函数的功能主要是检测 DHT11 的反馈。如图 4.3 所示墨绿色部分为 DHT11 的响应信号，也就是图 4.3 中已经标注出来的 T3 时间段。通常情况下，传感器 DHT11 先拉低数据线告诉主机自己已经接收到了从主机发送来的开始工作讯息，这个将信号拉低的时间段应等于 40us-50us。紧接着，传感器告诉主机自己要开始向它递送“情报”讯息了，也就是图 4.3 中 DHT11 拉高数据线，如图 4.3 所示，图中的 T4 部分，保持 40us—50us，随后开始输出数据。while (DHT11_DQ_IN&&retry<100)用来检测 T3 部分是否满足条件，如果满足接着执行 while (!DHT11_DQ_IN&&retry<100)代码段，这段代码检测信号是否满足 T4 部分。如果以上回应条件都满足将 return 0，表示检测到 DHT11 的存在，否则 return 1，表示未检测到 DHT11 的存在。

```
u8 DHT11_Read_Bit(void)
{
    u8 retry=0;
    while(DHT11_DQ_IN&&retry<100)    //检测低电平
    {
        retry++;
        delay_us(1);
    }
    retry=0;
```

```
    while(!DHT11_DQ_IN&&retry<100)    //检测高电平
    {
        retry++;
        delay_us(1);
    }
    delay_us(40);                     //检测等待 40us
    if(DHT11_DQ_IN)return 1;
    else return 0;
}
```

经过复位与校验操作后，DHT11 前期的准备工作就完成了，然后开始向主机发送采集信息的信号。DHT11_Read_Bit 函数为读取 1bit 信号，每一 bit 用 0 和 1 表示。如图 4.4 所示 DHT11 先发送 12us—14us 的低电平信号，再发送 26us—28us 的高电平信号来表示数字‘0’，随后采集的信号为下一 bit。如图 4.5 所示 DHT11 先发送 12us—14us 的低电平信号，再发送 116us—118us 的高电平信号来表示数字‘1’，随后采集的信号为下一 bit。

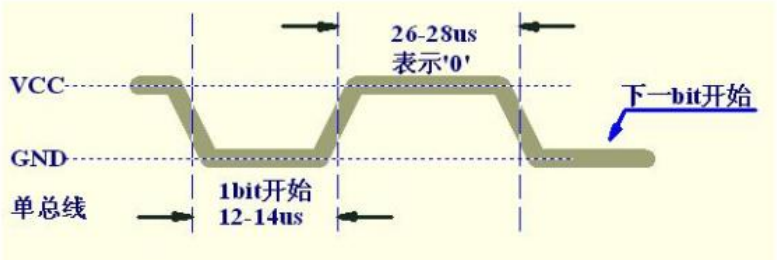

图 4.4 数字‘0’信号表示

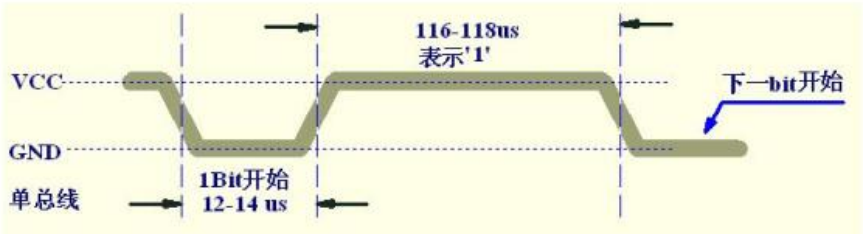

图 4.5 数字‘1’信号表示

while(DHT11_DQ_IN&&retry<100)函数先检测 DHT11 发出的是否为低电平信号，while(!DHT11_DQ_IN&&retry<100)函数检测 DHT11 发出的信号是否为高电平信号，检测正常说明现在 DHT11 开始发送采集温湿度的数据信号。接着 delay_(40us)等待 40us，随后如果检测到高电平信号说明传输的信号是数字‘1’，否则为数字‘0’。传感器不会人类的

语言，如何表达出 0 和 1 的讯息呢，传达讯息的时间太长太短都不合适。如图 4.4 和图 4.5 所示可以发现，人类想出了一个非常聪明的解决方式，根据高电平的持续时间来区分代表数字 0 和 1。如图 4.4 所示，数字‘0’为高电平持续 26us—28us，数字‘1’为 116us—118us，因此以等待 40us 来判断。

```
u8 DHT11_Read_Byte(void)
{
        u8 i,dat;
        dat=0;
        for (i=0;i<8;i++)
   {
                dat<<=1;
                dat|=DHT11_Read_Bit();
      }
      return dat;
}
```

8bit 为一字节，上面 DHT11_Read_Byte 函数代码功能是将位信号打包成字节输出。

```
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 5Byte 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;
}
```

DHT11_Read_Data 函数的功能是解析温度和湿度值，在第三章介绍 DHT11 温湿度传感器的时候我们知道 DHT11 电路是如何连接的，那么温湿度是如何传送温湿度信息的呢？温湿度传感器的讯息传送是以数据包的方式进行的，发送一次完整的数据包(包含温湿度信息)为 5Byte 也就是 40Bit。数据包中主要包含了三个信息：第一，温度值是多少；第二，湿度值是多少；第三，发送的数据是否准确。相对应的，数据包中的前 16 位（8 位表示整数、8 位表示小数）是温度信息，数据包的 17 位至 32 位（8 位整数、8 位小时）表示湿度信息，而最后的 8 位就是校验信息（前面 32 位相加）了。在温湿度传感器工作之前首先要进行复位，所以先使用 DHT11_Rst 函数复位温湿度传感器，然后 if(DHT11_Check()==0)判断等待响应。响应开始后读取 5Byte 数据，if((buf[0]+buf[1]+buf[2]+buf[3])==buf[4])函数用于数据校验。

```
u8 DHT11_Init(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOG, ENABLE);
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOG, &GPIO_InitStructure);                    //配置初始化 IO 口
  GPIO_SetBits(GPIOG,GPIO_Pin_11);
  DHT11_Rst();
  return DHT11_Check();                                     //等待 DHT11 的回应
}
```

DHT11_Init 函数用来进行 DHT11 的初始化，同时配置 GPIO 口 PG11 为 DATA 口，用于数据传输。

4.5 土壤湿度采集程序

设计中的温湿度采集主要基于 ADC 采集[5]，使用 ADC1 的通道 1 和 PA1 口进行数据

采集。首先，需要使用 Adc_Init 函数对 ADC 进行初始化操作，步骤流程如下：

① 开启时钟

② 复位并设置 ADC 工作频率

③ 设置 ADC 的工作模式

④ 使能 ADC 并校准[12]。

⑤ 配置规则通道参数。

⑥ 开启软件转换。

⑦ 等待转换完成，读取 ADC 值。

```
void    Adc_Init(void)
{
  ADC_InitTypeDef ADC_InitStructure;
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |RCC_APB2Periph_ADC1  , ENABLE );
```

RCC_APB2PeriphClockCmd 函数使能 PA1 端口与通道 ADC1 的时钟。

```
  RCC_ADCCLKConfig(RCC_PCLK2_Div6);
```

在 STM32F103 中，ADC 的正常工作频率是不能超过 14MHZ 的，而系统系统的时钟频率是 72MHZ，我们必须进行分频操作，程序中使用的是 6 分频，为 ADC 提供 12MHZ 的工作频率。

```
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
```

这三行代码主要将 PA1 设置为模拟输入引脚。

```
  ADC_DeInit(ADC1);
```

这里使用 ADC_DeInit 函数进行复位操作。

```
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;                  //是否开启扫描模式
  ADC_InitStructure.ADC_ContinuousConvMode = DISABLE; //是否开启连续转换模式
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;   //设置触发事件
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStructure);
```

在设计中 ADC 主要采用独立模式工作，扫描、轮转这些都不需要开启，触发使用软件触发。最后要对 ADC 数据对其方式进行配置，有左对齐还是右对齐两种，设计中使用的是右对齐方式，即 ADC_DataAlign_Right。

```
    ADC_Cmd(ADC1, ENABLE);              //使能 ADC
    ADC_ResetCalibration(ADC1);         //使能复位校准
    while(ADC_GetResetCalibrationStatus(ADC1));    //等待复位校准结束
    ADC_StartCalibration(ADC1);         //开启 AD 校准
    while(ADC_GetCalibrationStatus(ADC1));     //等待校准结束
  }
```

对于 ADC 初始化配置完成以后还要对 ADC 进行校准的工作。到这里 ADC 前期的预备工作就完成了，接下来要进行的工作就是 ADC 采样了，并将采集到的值转换为我们熟悉的面孔。

```
u16 Get_Adc(u8 ch)
{
  ADC_RegularChannelConfig(ADC1, ch, 1, ADC_SampleTime_239Cycles5 );
```

首先设置采样周期为 239.5。

```
   ADC_SoftwareStartConvCmd(ADC1, ENABLE);              //使能软件转换功能并启动
   while(!ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC ));      //等待转换结束
   return ADC_GetConversionValue(ADC1);                 //返回最近一次转换结果
}
```

下面的函数 Get_Adc_Average 用于多次获取 ADC 值，为了提高采集样本的精确精确度与可靠性，我们采用取平均值法进行计算。

```
u16 Get_Adc_Average(u8 ch,u8 times)
{
  u32 temp_val=0;
  u8 t;
  for(t=0;t<times;t++)
  {
      temp_val+=Get_Adc(ch);
      delay_ms(5);
  }
```

```
    return temp_val/times;
}
```

4.6 定时中断程序

设计中主程序使用到了两个定时中断，分别是 TIM3 和 TIM4。TIM3 主要是服务于60min 的延时操作，TIM4 服务于 30S 的加水操作。本次设计的时间控制没有选用 delay 延时函数，而是选用了定时中断，是因为如果选用 delay 延时，整个系统都会停止工作来等待 30S 加水或者 60min 延时进程的完成才能运行程序中的其他工作，这样是不符合实际使用情况的。使用定时中断，可以保证其它模块的同时运行。定时器中断实现步骤如下：

① 使能定时器时钟。

② 初始化定时器。

③ 开启定时器中断，与此同时配置 NVIC 进行中断优先级设置。

④ 使能定时器。

⑤ 编写中断服务函数。

```
void TIM3_Int_Init(u16 arr,u16 psc)
{
  TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
  NVIC_InitTypeDef NVIC_InitStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
```

对 TIM3 定时器进行使能操作。

```
  TIM_TimeBaseStructure.TIM_Period = arr;
  TIM_TimeBaseStructure.TIM_Prescaler =psc;
  TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
```

上面五行代码主要用于 TIM3 定时器的初始化，依次是设置定时周期，设置分频系数还有计数方式。

```
  TIM_ITConfig(TIM3,TIM_IT_Update,ENABLE );
```

使能定时器中断。

```
  NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;                  //TIM3 中断
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
```

```
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 3;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
```

上面的程序用来中断优先级的配置，设计中设置 TIM3 中断的优先级为 0，从优先级为 3，优先级越小，优先级别越高。

```
    TIM_Cmd(TIM3, ENABLE);
}
```

前面的程序中 TIM3 定时器已经配置完成，所有的准备工作已经就绪，此时还有一个工作，就是要打开定时器，TIM_Cmd 函数的作用就是定时器配置完成之后将定时器打开。

```
void TIM3_IRQHandler(void)     //TIM3 中断
{
  if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET)          //检查 TIM3 更新中断发生与否
        {
              TIM_ClearITPendingBit(TIM3, TIM_IT_Update    );   //清除 TIMx 更新中断标志
              sum1++;
        }
}
```

经过 TIM3_Int_Init 函数的初始化，定时器可以正常工作了，TIM3_IRQHandler 函数用来配置定时器的服务程序，程序中我们设置每中断一次 sum1 自加 1。也就是计时 0.5 秒一次。下面程序为 TIM4 定时器，与 TIM3 基本一致，这里不再详细说明。

```
void TIM4_Int_Init(u16 arr,u16 psc)
{
   TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;
   NVIC_InitTypeDef NVIC_InitStructure;
   RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
   TIM_TimeBaseStructure.TIM_Period = arr;
   TIM_TimeBaseStructure.TIM_Prescaler =psc;
   TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
   TIM_TimeBaseInit(TIM4, &TIM_TimeBaseStructure);
   TIM_ITConfig(TIM4,TIM_IT_Update,ENABLE );
```

```
    NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
    TIM_Cmd(TIM4, ENABLE);
}
void TIM4_IRQHandler(void)
{
    if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)
    {
TIM_ClearITPendingBit(TIM4, TIM_IT_Update    );
        sum2++;
    }
}
```

4.7 本章小结

本章节主要是系统软件部分，在 4.1 节中给出了设计的系统流程图，并将系统的运行机制作了详细的说明。4.2 小节简单的介绍了在程序设计中使用到的开发环境 MDK5，这部分不是本文想要重点说明的部分，稍作了解即可。在 4.3 节中给出了系统的主程序，并对其进行了解释说明，主程序主要是调用其他封装好的硬件库程序来运行，实现本次的系统功能，需要指出的是，继电器的工作状态为低电平有效，所以对于继电器的控制，大家完全可以把它当做是点亮 LED 灯，这个应该是最基本的知识了。在 4.3 节和 4.4 节中，对于设计的感知部分温湿度传感器和土壤湿度传感器的软件部分进行了说明，DHT11 温湿度传感器的操作比较复杂，需要了解这款传感器的运行机制，小节中已经作出了详细的说明。相对于温湿度传感器，土壤湿度传感器就要简单的许多了，只需了解普通的 ADC 就可以。在第 6 小节中介绍了定时中断程序，本次设计中为什么没有选择 delay 函数来做延时，反而选择了定时中断程序来实现，这里做一下说明。因为在实际调试中，在使用 delay 的时候，系统整体都会处于停滞的状态去等待延时的完成，在实际使用中这样的代价太大了，不可能说让其他所有的工作部分都停止工作来等待延时。所以选用定时中断来完成延时的操作，即使处于延时阶段除了抽水机的其他部分还能正常工作，以上就是总结部分。

第五章 系统测试

本次的设计就相当于是在做一款产品，一款新产品的诞生有不可或缺的一环，在产品研发中，测试有着举足轻重的地位。测试的方法有很多，针对不同的产品，不同的用户，不同的开发者都有不同的测试方法。在本次设计中将测试分为硬件测试与软件测试两个部分，下面进行详细的说明。

首先进行的是硬件测试，嵌入式开发中硬件是基础，硬件如果有问题而在前期又没有发现，在后面的程序调试中会埋下很大的隐患。最先开始测试的是电源模块，只有稳定合格的电源，电路才能源源不断的获得能量持续运行。这里主要用到的是万用表和电压源，先为电路提供 3.3V 的电源，因为 3.3V 是电路可承受的电压，不会损害任何元器件。测试完成后，电路所有器件导通正常，在外接高电压，这是还要测量，测量电压是否都完全正常。随后进行 MCU 的测试，MCU 的测试要在开展实验之前，在实验中我采用的方法是下载测试程序，如果程序可以下载正常，且能够实现测试实验中的相关功能，说明芯片正常。在测试显示屏的时候直接下载一个显示屏显示程序即可，能够正常显示就说明显示屏正常。DHT11 温湿度传感器与土壤湿度传感器的测试可以使用示波器检测传感器上电后的数据口是否能够正常传输数据。

硬件测试正常之后，即可灌输程序，进行下一步的软件测试。实际测试中，程序下载完成后启动电源，首先观察 DS0LED 灯是否闪烁，如果闪烁，说明程序能够运行，接着观察显示屏上的温湿度值的显示。目前（江南四月份）的气温正常维持在 20 摄氏度左右，此时控制加热器端的继电器上的 LED 点亮，表明加热器可以正常工作。在实际测试中，我们使用吹风机代替加热装置对 DHT11 型传感器进行加热，显示屏可以实时显示温度值，实验结果显示当温度值大于等于 25 摄氏度时继电器端的 LED 灯熄灭了，表明加热装置停止工作。同样的，我们可以使用吹风机对 DHT11 传感器除湿，将其湿度值降至 60%以下，此时我们观察加湿器端的继电器指示灯点亮，表明风扇工作正常。随后用一块湿润纸片对传感器加湿，湿度超过 60%后继电器上的 LED 灯熄灭，也就是说此时的加湿器已经停止工作不在运行了。随后我们继续将纸片覆盖在温湿度传感器上，一直到湿度值显示为 80%以上，此时我们会发现风扇端的继电器 LED 灯点亮，LED 灯被点亮表明风扇开始工作。因为现在正常湿度不高，所以过一会之后湿度值就会自动降至 80%以下。此时的继电器 LED 灯熄灭，表明风扇停止工作。最后一步是测试土壤湿度传感器，主程序启动之后，土壤湿度传感器可以测量到土壤湿度值，但是此时的加水电机不会工作了。因为在程序中我们设置了加水电机等待的定时中断，需要等待这个定时结束之后加水电机才会响应。延时等待结

束后，土壤湿度如果小于 40%加水电机就会启动 30S，30S 后自动停止，这期间会等待一个小时。在这一小时当中无论土壤湿度传感器如何变化，加水电机都不会影响。

经过测试之后确认系统工作顺利，功能正常。到这里测试还没有结束，对于一款好的嵌入式产品来说，运行要具有长期性，也就是可靠性。火星车机遇号在 2003 年发射升空，美国宇航局赋予它的使命是进行火星探测，因为当时科技的限制以及火星恶劣的环境，当局对于机遇号的工作周期设计是 90 天，也就是说 2004 年机遇号就可以退休了。可是令人没想到的是，机遇号一直到 2018 年 6 月 10 号才停止工作，原本计划 90 天的工作周期一直持续到了 5250 天。机遇号是可靠性设计的典范，不论是现在还是将来，我们做的产品都应该有这种可靠性的意识，这是应该是一名嵌入式工程师的基本素养。

第六章 总结与展望

通过本次的设计过程，使得自己对于专业知识有了更深层次的理解。不仅如此，设计中还涉及到其他专业领域的知识。作为一名嵌入式工程师，需要掌握的专业知识有很多，比如说硬件相关的硬件设计调试，物联网感知相关的传感器技术，还有与软件相关的嵌入式软件技术等。刚刚所提到的这些还仅仅是一名嵌入式工程师的基本条件，如果想要更进一步有较高的发展和作为，并能够成为一名优秀的嵌入式工程师，仅仅掌握这些是远远不够的，还应当拥有敏锐的产品嗅觉。通俗来讲就是能过做出大家喜欢、爱用、有用的产品，这就需要工程师的综合知识能力过硬。

本次设计虽然一波三折，困难重重，最终的结果还是比较喜人的，前期设计中规划的功能都已实现，系统能够实现对于空气的温湿度，土壤的湿度进行自动的控制。在后续的持续观测中，系统运行良好，有较好的可靠性。虽然以前没有接触过 STM32 芯片，但是在平时学习中打下了坚实的基础。在自学之中明显感受到以往课堂中学习的知识不够用，但是如果以前课堂中没有学过这些内容，自己不会接受的这么快。通过这次机会对芯片有了更加清晰，深刻的理解，对于日后的目标方向也有了比较好的把握。现在校园里的很多同学比较浮躁，整天抱怨学校学的东西没用或者是老师教的不好，殊不知只有沉下去才得一飞冲天。一屋不扫何以扫天下，太多的人自命不凡，不是什么惊天动地的大事情都懒得伸手，久而久之也就养成了眼高手低的坏习惯。作为一名学生，我们应当做好眼前该做的事情，“莫等闲，白了少年头，空悲切!”。在我看来，学校教授的课程就像少林寺的武僧每天站桩、扎马步练习基本功一样，基本功修炼不好，再高深的武功到头来也是花拳绣腿。

因为本次设计时间有些仓促以及经费不足，本次设计做出的功能还有许多地方需要改进完善，现在实现的功能相对来说比较简单。在以后后续的工作中计划将本设计接入云端，感知系统感知到的数据可以实时上传至网上，用户通过手机或者网页就可以实时了解植物的状态。本次设计中也没有做到控制参数的自由调节，虽然本次使用的屏幕是触摸屏，但是其性能并没有被充分的开发出来。后续工作中希望可以完成植物生长环境的定制化服务，简简单单的通过触摸屏就可以完成。这就需要 STM32 有一个自己的运行 OS，这些都是此次设计中没能实现的。除此以外因为本次设计经费缺乏，前期计划的 CO_2 传感器与粉尘传感器未能上马。如果以后经费充足，可以对本次设计继续改进。通过 CO_2 传感器检测植物生长环境中的 CO_2 浓度，给植物的光合作用创造良好的条件，存进植物的生长。我们大家都知道，现在的环境质量很差，PM2.5 的问题还是相当严重的。或许很多人不了解，PM2.5 除了对人有影响，对植物的损伤也很大，因为 PM2.5 中的颗粒物会阻塞植物的空气交换也

就是植物无法呼吸，进而影响植物生长。我们可以根据 PM2.5 的浓度，选择给植物通风换气的合适时机。促进植物的生长。

参考文献

[1] 韩允. MEMS 传感器的发展概况[J]. 电子产品世界, 2019(1):4-8.

[2] 邹丰谦, 邱成军. 植物生长环境测控系统设计[J]. 传感器与微系统, 2018(1):111-113,116.

[3] 朱斌, 张磊, 怯肇乾. STM32-MCU 片内 IIC 接口的驱动程序设计[J]. 电子世界, 2018, No.550(16):115-117.

[4] 崔锟, 孙如军, 郝萌, et al. 新型智能养花装置的研究[J]. 电子世界, 2018, 557(23):100+102.

[5] 刘洪涛, 邓二伟. 基于 STM32 的智能自动浇水花盆的设计[J]. 自动化与仪器仪表, 2016(8):232-233.

[6] 徐功林, 封蕾. 基于 5TM 传感器的土壤温湿度监测系统设计[J]. 山西电子技术, 2017(2):35-37.

[7] 倪天龙. 单总线传感器 DHT11 在温湿度测控中的应用[J]. 单片机与嵌入式系统应用, 2010(6):60-62.

[8] 意法半导体新系列 STM32 微控制器[J].电子产品世界,2019,26(1):88.

[9] 程文龙, 徐瑾, 孙智勇. 基于 STM32 呼吸灯的实现[J]. 电脑知识与技术：学术交流, 2018, 14(3): 198-199.

[10] 庞岳峰, 朱巍巍, 谢克佳, et al. 基于 SHT11 传感器的测控设备温湿度监测系统设计[J]. 无线电通信技术, 2018, v.44；No.265(05):106-110.

[11] 黄飞龙, 黄海莹, 何艳丽. 基于 STM32 的气象数据在线监测仪设计[J]. 计算机测量与控制, 2018, 26(11):294-298.

[12] 曹金华, 王宜怀. AD 转换的动态在线校正技术[J]. 实验室研究与探索, 2008, 27(4):44-47.

[13] 舒胜强, 王宜怀, 曹金华. 一种单芯片的 CAN-USB 桥协议及接口设计[J]. 微计算机信息, 2010, 26(35):62-64.

[14] 王林, 王宜怀. 基于 U-ISP 的 UF32 Flash 编程方法[J]. 计算机工程, 2009, 35(24):274-275.

[15] 张立良, 宁祎, 刘磊. 基于云服务的智能花盆系统设计[J]. 物联网技术, 2017(05):82-83+85.

[16] 刘绍丽,王献合.基于 STM32 单片机的智能温度控制系统的设计[J].电子测试,2018,(21):34-35,140.

[17]苑新宇, 马淑香, 艾志杰. 全自动智能花盆的设计与实现[J]. 信息技术与信息化, 2017(5):35-37.

[18]Boselin Prabhu S R , Dhasharathi C V , Prabhakaran R , et al. Environmental Monitoring and Greenhouse Control by Distributed Sensor Network[J].International Journal of Advanced Networking & Applications, 2014.

[19]Wang H , Su J . The Design and Research of Infrared Sensor Monitoring System Based on STM32[C]// Fourth International Conference on Computational & Information Sciences. IEEE Computer Society, 2012.

[20]Seelye, Gupta, Bailey, et al. Low cost colour sensors for monitoring plant growth in a laboratory[C]// Instrumentation & Measurement Technology Conference. IEEE, 2011.

致谢

本次毕业设计工作的结束，也意味着自己在文正的大学时光要结束了，现在回想甚是不舍。

首先感谢我的指导老师王林老师，感谢王林老师对于本次设计的耐心指导。除此以外，两年的大学时光还教会了很多，学习上生活上的都有，使我受益良多。学习上，因为王林老师的教诲，使我对于嵌入式这门学科有了更加深入的了解，并且明白自己现在的水平价值几何，使我更加坚定了在这个行业做下去的决心。生活中，王林老师教会我要如何去面对生活中的种种困难、挫折，使我的心智更加成熟坚定，老师的谆谆教导现在还在耳畔回响。

其次要感谢的是学校，感谢学校提供了这么好的平台。我是一名转本的学生，文正学院的基础设施比起以前的学校，可以说是非常差劲的，记得刚来的时候很多地方不习惯。基础设施差，规矩也多，尤其是图书馆，各种限制。即便如此，我也不后悔来到文正学院，因为这两年在文正我遇到了好的老师，使我不虚此行。学校的很多老师来自苏大，真心感谢各位老师能够有教无类，能够遇到各位老师真的是三生有幸。不见高山，不显平川；不见大海,不知溪流。各位老师于我而言就是高山、大海，因为你们我的视野变得开阔。

最后，感谢审阅设计的各位专家们，感谢抽出宝贵的时间审阅本设计。

附录：STM32F103ZET6 资源地址图