基于 stm32 的智能花盆控制系统

摘要：随着人们生活质量的上升，越来越多的人追求更智能、更便利的生活方式，这体现在生活的方方面面，包括花卉种植。很多人喜欢种植花草盆栽，这不仅是调剂生活的一项兴趣爱好，有时候也能给人们带来实际的益处，比如白领们喜欢在桌上放一小盆绿色植物，可以清新空气甚至防辐射。同时，种植盆栽又是一件需要耐心和考究的事情，如果照料不当，植物就会干瘪，发黄甚至枯萎，这恐怕是种植者最不愿看到的事情。

因此，本人设计了一款使用 STM32F103 单片机运行的智能花盆控制系统，利用温湿度传感器和光照传感器，可实时监控植物的环境状态等，再反馈给种植者，当周围环境和土壤状况未达到植物所需，就会及时提醒种植者。

关键词：智能花盆，STM32，温湿度传感器

Intelligent Flowerpot Control System Based on STM32

Abstract: With the rise of people's quality of life, more and more people pursue a more intelligent and convenient way of life, which is reflected in all aspects of life, including flower planting. Many people like to plant flowers and plants in potted plants, which is not only an interest in adjusting life, but also can bring practical benefits to people sometimes, such as white-collar workers like to put a small pot of green plants on the table, which can clean the air and even prevent radiation. At the same time, planting potted plants is a matter of patience and refinement, if not properly cared for, plants will dry, yellow or even withered, which is probably the last thing growers would like to see.

As a result, I designed an intelligent flowerpot control system running using STM32F103 single chip microcomputer. The planting of flowers can monitor the environmental state and soil state of plants in real time, and then feedback to the growers, when the surrounding environment and soil conditions do not meet the needs of plants, will remind the growers in time.

Keywords: Intelligent flower pot, STM32,Temperature and humidity sensor

目　录

1 绪 论

1.1 研究背景和意义

1.1.1 背景分析

随着物联网技术走进人们的视野并越来越受到重视，物理空间不再是人们对生存空间的唯一需求，人们还需要空间智能化[1]。在这个新时代，人们想了很多办法钻研如何让我们的生活方式更智能化，以智能去抵抗一些自然的不便，或者给生活增添便利和乐趣。

许多人喜欢种植花草盆栽怡情，能给枯燥的生活带来乐趣，还能让人们在快节奏的生活里慢下来看看身边的美好的小生物。人们的生活水平提高，有更多精力去培养像种植盆栽这样的兴趣爱好，不仅能调剂生活，有时候也能给人们带来实际的益处，比如可以清新空气，有的还可以防辐射。据介绍，休闲农业观光园是伴随着社会经济和城市化发展而兴起的一种高度复合型的乡村旅游形式,而主题花卉类休闲农业观光园是其中极具代表性和发展前景的一类[2]。主题花卉农园建设中,花卉的选择极其重要。此外，花卉旅游产业依托其农业种植资源以及深厚的花卉文化,在旅游市场上很受欢迎[3]。云南独特自然条件和气候条件使观赏植物资源非常丰富,是著名的花卉大省,素有“天然花园”之称。花卉产业不仅促进了云南经济发展,而且带动了花卉旅游升级转型[4]。

同时，种植盆栽又是一件需要耐心和考究的事情，对于花卉生长的土壤,最简单的要求就是要保证土壤肥沃,不积水,不板结,没有杂草[5]。如果不及时给予所需的足够的水分，光照，养分，就会影响植物的茁壮生长，且其衰败不可逆。随着人们生活质量的上升，人们不满足于原始的种植方式，越来越多的人追求更智能、更便利的生活方式，追求在天然的种植中用有更强的可控性。

1.1.2 本课题的研究意义

花卉种植有时候就像一门艺术，既可操控又不可捉摸，假如可以使花卉的“感受”和“心情”显现在我们眼前，则更为可操控性更为明显。花卉所处环境经常变化而我们不一定感受得到，但是通过数据实时检测则更为直观，这将大大便利了花卉种植者，且更精准。

对于不同的花种，我们需要设置各种花的成长环境。如百合花，此花的生产环境是高海拔产区为主。又如玫瑰花，此花的生存环境是高海拔产区为主，质量相对稳定，但因中低海拔地区因气候炎热和病害产品质量不佳，花期较短[6]。

除了为了更好地，更科学地栽培花外，精准浇水还能解决水资源的问题。实施节水技术以缓解水资源枯竭势在必行[7]。

本课题要设计一个基于 stm32 的智能花盆控制。于是系统的，可控的照料可减轻植物种植爱好者的照料难度，也增加把植物种植得更漂亮更茁壮的成功几率。人们可以享受到智能给种植带来的便利。系统运用于花卉的种植可实时监控植物的环境状态和土壤状态等，再反馈给种植者，当周围环境和土壤状况未达到植物所需，就会提醒种植者做出补充，使种植者和植物之间有更直观的“交流”，为花卉种植爱好者提供更智能可控的新时代种植方式。

1.2 课题研究方法和内容

1.2.1 研究方法

此篇论文在编写的过程中重点采取了以下的几种研究方法：

文献研究法：从自身的设计内容和特点出发，按需求和自己的实际情况有针对性地查询对自己有参考价值的、有辅助作用的书籍、资料，借鉴有相关经验人士的方法经验，从而能够达到更直接地构建自己的设计思路，以便于开始有自己特色的新的设计的钻研。

功能分析法：此方法可以通过设想和调查从大众及社会对这个设计的需求的角度分析这个设计存在的必要性，即设想它即将会给人们的生活带来哪些变化，哪些因素是我们可以设计改变的，以及调查大环境下的真实情况来相对准确地预判结果。

通过实验的方法：通过对已有问题的研究和分析，从已得出的结论和经验出发，编写各个模块的测试代码，再根据需求整合代码。从而实现智能花盆控制系统。

1.2.2 研究内容

本论文总共分为五个部分，每个部分内容安排如下：

第1章：绪论。说明课题研究的背景、课题意义和研究方法等。

第2章：系统的总体设计和硬件设计。主要介绍使用的模块线路连接，模块初始化方法并展示部分重要代码。

第3章：系统软件实现。介绍模块的基础理论及实现方法，展示部分重要代码。

第4章：系统调试。测试 DHT11 温湿度传感器等实际效果。

第5章：结论。主要分析系统在总体设计上待改进的一些地方，分析优点以及如何对缺点进行改进等。

最后是参考文献以及致谢内容。

本系统主要设计功能是系统先设计一些阈值，包含温度，湿度及光照。一旦不满足系统的阈值范围内，蜂鸣器就会发出警报。可通过按键停止警报。

2　系统总体设计及硬件设计

2.1　系统总体设计

本系统的总体设计主要分为硬件选择及设计、设计环境介绍、软件设计部分。硬件设计部分主要模块有 STM32F103 芯片开发板、LCD 屏、DHT11 温湿度模块、蜂鸣器模块、按键模块和光敏模块组成。

系统的总体设计软件部分则分为 STM32 的初始化；数据的采集；显示界面等几个部分组成。各部分通信如图 2-1 所示，其中传感器和 STM32 通过 GPIO 口通信，实现实时的监测当前环境中温度、湿度等环境因素，从而检测出花的环境。

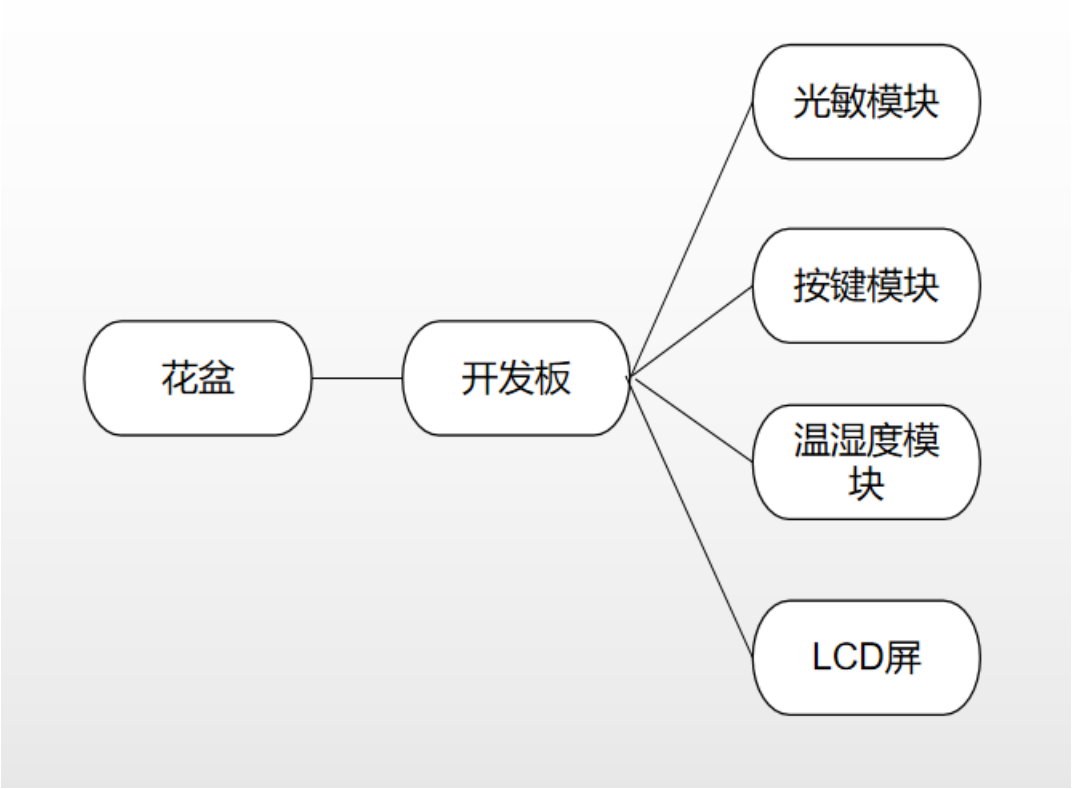

图 2-1 总体设计框图

硬件电路主要包括温湿度传感器、光敏模块、LCD 屏、蜂鸣器、和 STM32 单片机的连接，串口板和 STM32 单片机的连接。

2.2 系统硬件实现

2.2.1 STM32 处理器简介

目前 51 系列单片机实验箱众多,但其部分 I/O 口无输出能力、AD 需要靠扩展、运行速度较慢,而 STM32F103 具有 Cortex-M 内核,输出脚驱动能力强,系统时钟达 72MHz,同时具有 ADC、UART、SPI 等流的外设,本文提出开发基于 STM32F103 的实验板,由按键输入,蜂鸣器、步进电机、跑马灯、LCD 显示等输出。实践证明,STM32 实验板功耗低、集成度高、结构简单、开发工具易于使用,有助于提高学生的实践创新能力和电子设计竞赛能力,取得了良好效果[8]。

STM32 单片机是由意法半导体公司推出的一款芯片，相比于传统使用的 51 单片机，STM32 的拥有更强大的性能，并且在成本和消耗方面也比较理想，装有 ARM Cortex-M3 内核，最高可达到 72MHz 的频率。本课题使用到的具体型号是 STM32F103C8T6，与传统的 8 位或者 16 位嵌入式单片机相比，STM32 使用的 RISC 精简指令集拥有更高的代码运行效率和更大的 Flash，其 Flash 可以存储数据也可以储存程序，而且 STM32 拥有 68KB 的嵌入式 SRAM，可以同步 CPU 的时钟频率来进行读写数据，并且 STM32 可以兼容市面上所有的 ARM 工具和软件。

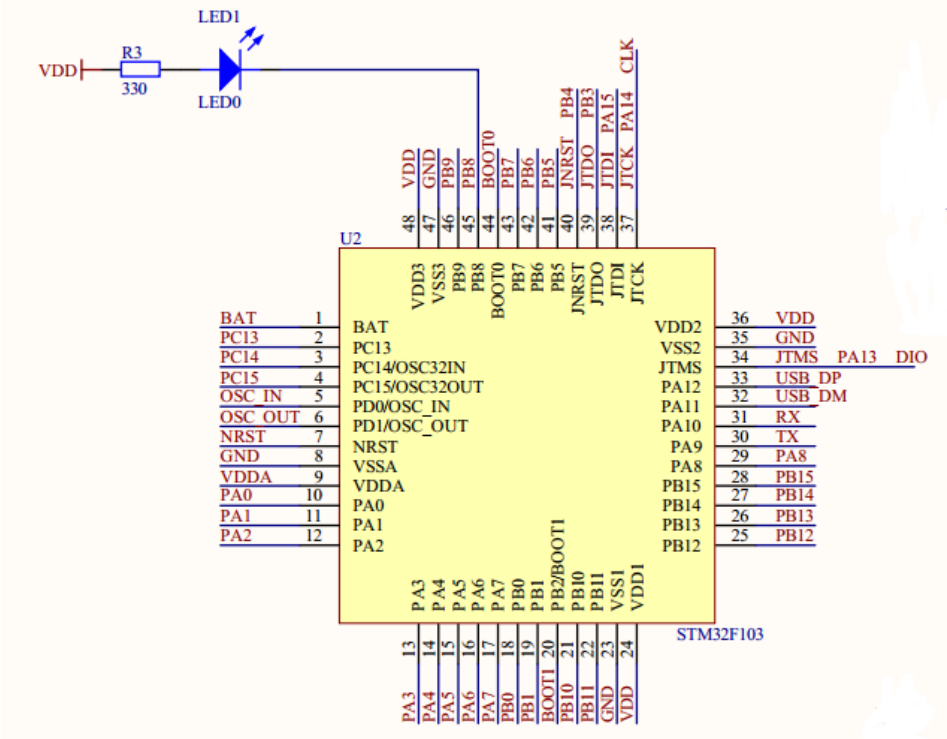

图 2-2 STM32 控制器引脚图

如图 2-2 所示，芯片共有 48 个引脚，本课题主要使用到了其中的 GPIO 引脚用于和各传感器进行数据的通信，VDD 引脚提供各传感器的供电。

2.2.2 温湿度传感器模块

DHT11 温度和湿度传感器是一种复合型的传感器，其输出为已校准的数字信号，其产品拥有卓越的长期稳定性和极高的可靠性是因为使用了专用的温湿度传感技术和数字模块技术，一般市面上的温湿度传感器都有一个测温元件、一个感湿元件和一个高性能单片机组成，与 STM32 的连接如图 2-3 所示：

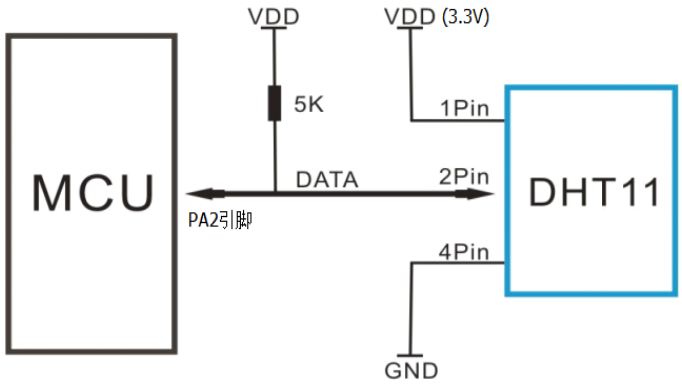

图 2-3 DHT11 温湿度传感器连接图示

通过查看 DHT11 的使用手册，得知 DHT11 的数据格式为单总线数据格式，平均每 4MS 完成一次通讯，数据分为小数部分和整数部分，数据格式为“8 位的湿度整数数据+8 位的湿度小数数据+8 位的温度整数数据+8 位的温度小数数据”，数据传送正确时校验和等于“8 位的湿度整数数据+8 位湿度小数数据+8 位的温度整数数据+8 位的温度小数数据”所得结果的后 8 位，数据通过读取 STM32 的 PA2 引脚的值获取。此外，还会取用多次采集数据，去掉最大最小值，再取平均值的方法，以实现数据的精确[9]。

2.2.3 光敏传感器模块

光敏传感器模块是个常用的模块之一。通常来说，光敏传感器是由光信号转换成电信号的传感器。它的敏感波长在可见光波长附近，包括红外线波长和紫外线波长。本模块采用的光敏二极管，即光敏电阻。其特性是具有单向导电性，因此工作时需加上反向电压。模块连接如下图所示：

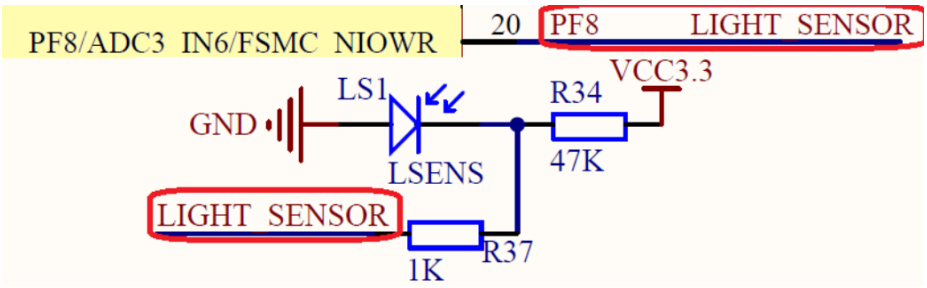

图 2-4 光敏模块连接图

2.2.4 蜂鸣器模块

蜂鸣器一般分为压电式和电磁式，本次课题使用的是电磁式有源蜂鸣器。

蜂鸣器模块一共有三个引脚：VCC 引脚，I/O 引脚，GND 引脚。当 I/O 引脚输入低电平时，蜂鸣器不响。当 I/O 引脚输入高电平时，蜂鸣器响。与 STM32 的连接方法如图 2-5 所示：

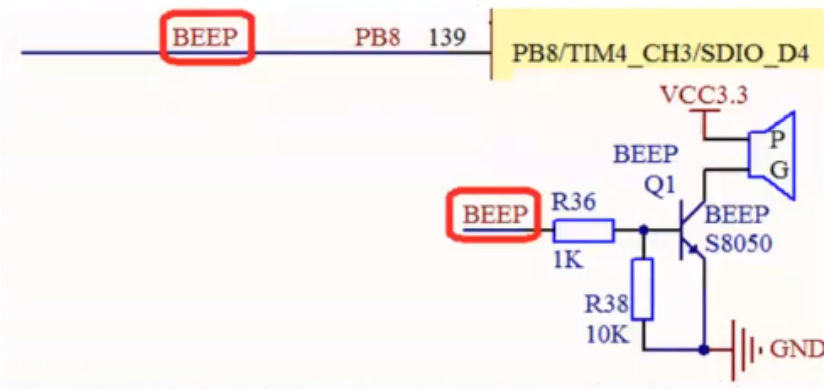

图 2-5 蜂鸣器连接图示

2.2.5 LCD 模块

LCD （ Liquid Crystal Display 的简称）液晶显示器。其特点是外观小巧精致、低压微功耗、颜色鲜艳、被动显示型不刺眼等优点。LCD 屏是由简单的三原色显示出图案，使用 16 位数据。使用 0 到 4 位代表蓝色，第 5 位到第 10 位为绿色，第 11 位到位 15 位为红色。数值越大，颜色越深。LCD 屏硬件连接图如下：

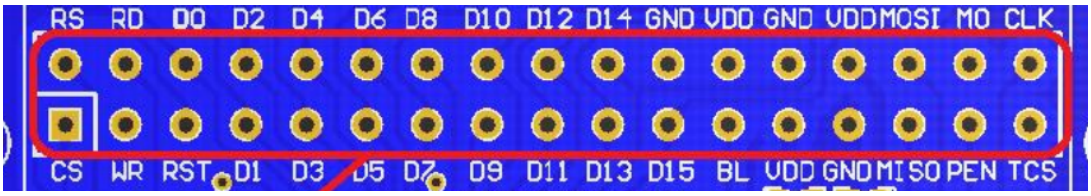

图 2-6 LCD 屏连接图

把 LCD 屏插入图片上主板的这个区域。

2.2.6 按键模块

按键是最常用的功能之一。本次选用的开发板有 4 个按键。分别是 KEY0、KEY1、KEY2、和 WK_UP。板上的按键 KEY0 连接在 PE4 上、KEY1 连接在 PE3 上、KEY2 连接在 PE2 上、WK_UP 连接 在 PA0 上。如图 2-7 所示：

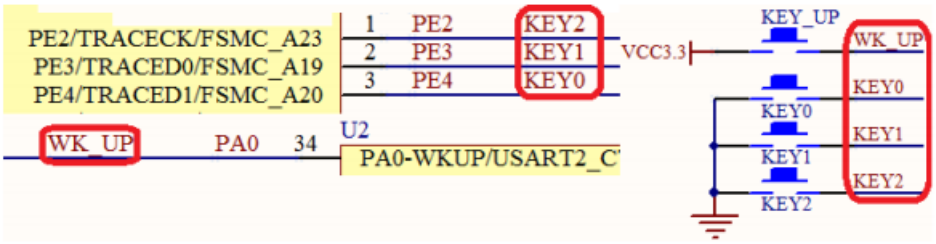

图 2-7 按键连接图示

系统软件实现

2.3 开发工具的介绍

2.3.1 Keil uVision5

Keil 由美国 Keil Software 公司开发出来的软件，兼容 C 语言软件系统。Keil 通过一个集成开发环境将 C 编译器、调试器、编辑器、宏汇编、链接器、库管理等组合在一起。本系统是使用该软件加入固件库进行代码编写，编译的。

2.3.2 下载工具

ST-Link 作为一种主流的仿真器，可以免费下载，代码大小和使用时间不受限制[10]。使用 ST-Link 前需要安装其驱动以及配置编译软件（MDK）。

2.4 软件逻辑框架

软件设计整体逻辑框架如图 3-1 所示:

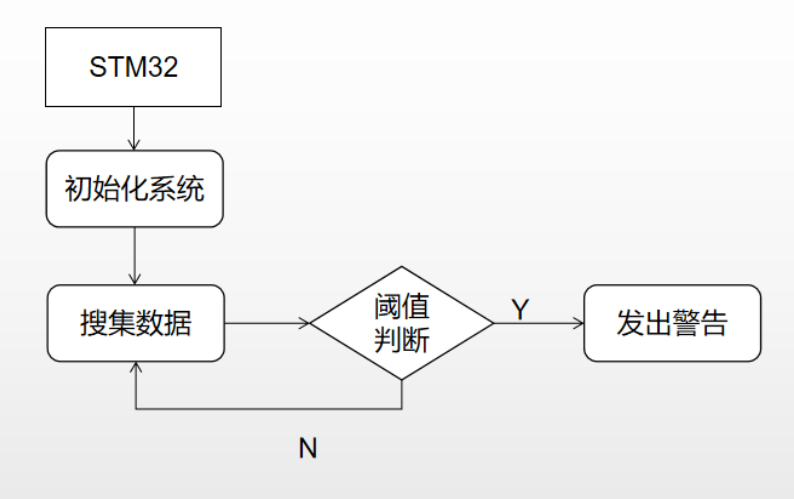

图 3-1 逻辑框架图示

2.5 模块控制代码实现

2.5.1 建立工程

在新建工程之前先新建一个文件夹备用，命名为Flowerpot，这个文件夹可以用来存放后面建的工程。然后再在MDK里新建立一个工程，工程的目录选择之前新建的文件夹Flowerpot，然后在这个目录下新建一个子文件夹USER用来存放工程文件（main.c文件也存放在此文件夹里）。做完前面的工作初步建立好工程文件夹之后会弹出一个选项框选择设备芯片，根据我自己所使用的开发板我选择的是STMicroelectronics→STM32F1 Series→STM32F103→STM32F103ZET6这个选项。

初建立了一个工程之后，还需要往工程里面添加一些启动代码和一些特定文件，在工程目录下面新建三个文件夹CORE, OBJ 和STM32_F1。CORE用来放核心文件和启动文件，OBJ用来存放编译中产生的中间文件和生成的hex文件，STM32_F1用来存放开发板的官方库函数源码文件。将Output路径设置为OBJ。将官方固件包中CoreSupport里的core_cm3.c和core_cm3.h文件以及startup\arm里的startup_stm32f10x_hd.s文件复制到CORE里。将官方固件库包中的STM32F10x_StdPeriph_Driver里的src和inc文件夹复制到STM32_F1里。再将STM32F10x里的stm32f10x.h，system_stm32f10x.c和system_stm32f10x.h文件以及STM32F10x_StdPeriph_Template里的main.c，stm32f10x_conf.h，stm32f10x_it.c，stm32f10x_it.h复制到USER里。

前面已经在文件夹里放好了相关文件，接下来在MDK工程里打开Manage Project Itmes，在Groups框里依次点击每个Groups然后点击Add Files,往Group里面添加对应文件夹里的文件，最后再检查是否全部加入进去。

在MDK里添加好头文件路径，然后还需要配置一个全局宏定义变量STM32F10X_HD,USE_STDPERIPH_DRIVER，把USER里main.c的代码复制到工程的main.c置换原有的。至此工程模板就建立好了。

2.5.2 Main主函数

main函数先调用各个模块的初始化函数，然后一直读取温湿度和光照数据，并显示出来，当超过或者低于阈值时，蜂鸣器就会响起。主要代码如下：

```
u8 t=0;
u8 temperature;
u8 humidity;
static u8 keytime = 0;
vu8 key=0;

u8 lcd_id[12];          //存放LCD ID字符串
delay_init();            //延时函数初始化
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
//设置NVIC中断分组2:2位抢占优先级，2位响应优先级
uart_init(115200);      //串口初始化为115200
LED_Init();              //LED端口初始化
LCD_Init();
POINT_COLOR=RED;        //设置字体为红色
delay_init();            //延时函数初始化

Lsens_Init();           //初始化光敏传感器

BEEP_Init();            //初始化蜂鸣器端口

KEY_Init();          //初始化与按键连接的硬件接口

BEEP=0;

while(DHT11_Init()) //DHT11初始化
{
    LCD_ShowString(30,130,200,16,16,"DHT11 Error");
    delay_ms(200);
    LCD_Fill(30,130,239,130+16,WHITE);
    delay_ms(200);
}
LCD_ShowString(30,150,200,16,16,"Temp:  C");
LCD_ShowString(30,170,200,16,16,"Humi:  %");
```

上面代码初始化 LCD 屏，延时函数初始化，中断初始化，光敏模块初始化，蜂鸣器初始化，按键初始化，以及温湿度模块初始化。

```
while(1)
{
    adcx=Lsens_Get_Val();
    LCD_ShowxNum(30+10*8,130,adcx,3,16,0);//显示ADC的值

    DHT11_Read_Data(&temperature,&humidity);    //读取温湿度值
    LCD_ShowNum(30+40,150,temperature,2,16);    //显示温度
    LCD_ShowNum(30+40,170,humidity,2,16);       //显示湿度

    if (adcx<30&&temperature>30&&humidity<20&&keytime == 0)
        //超过阈值响起蜂鸣器警告
    {
        BEEP=1;
        LCD_ShowString(30,200,200,16,16,"warming");
        //蜂鸣器响时，显示warming
    }
    else
    {
        BEEP=0;
    }
```

上面代码主要获取温湿度数值，光强数值。并一直检测是否不满足预设的值。

2.5.3 温湿度传感器模块

本小节主要介绍 DHT11 温湿度传感器模块的工作方法及代码的实现方法。

DHT11 温湿度传感器顾名思义是可以检测周围环境的温度和湿度的传感器口。其模块的示意图如下图所示：

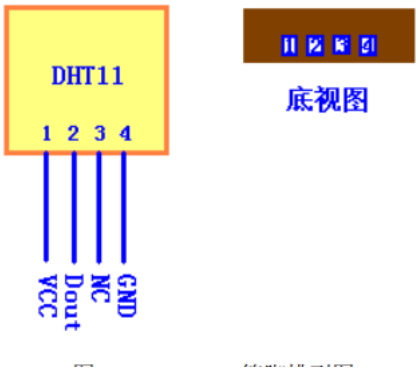

图 3-2 DHT11 模块示意图

从图可看该模块一共有 4 个引脚。包括 GND、NC、Dount、VCC 硬件。其中 Dount 引脚是用来传输数据的。DHT11 温湿度传感器单靠 Dount 引脚进行数据传输。即，该引脚完成输入输出功能。

接下来介绍 DHT11 的传输时序。DHT11 数据传输时序如下图所示：

图 3-3 DHT11 传输时序图

首先主机发送开始信号，即：拉低数据线，保持 t1（至少 18ms）时间，然后拉高数据线 t2（20~40us）时间，然后读取 DHT11 的响应，正常的话，DHT11 会拉低数据线，保持 t3（40~50us）时间，作为响应信号，然后 DHT11 拉高数据线，保持 t4（40~50us）时间后，开始输出数据。传输数字“0”和数字“1”时序如下图所示：

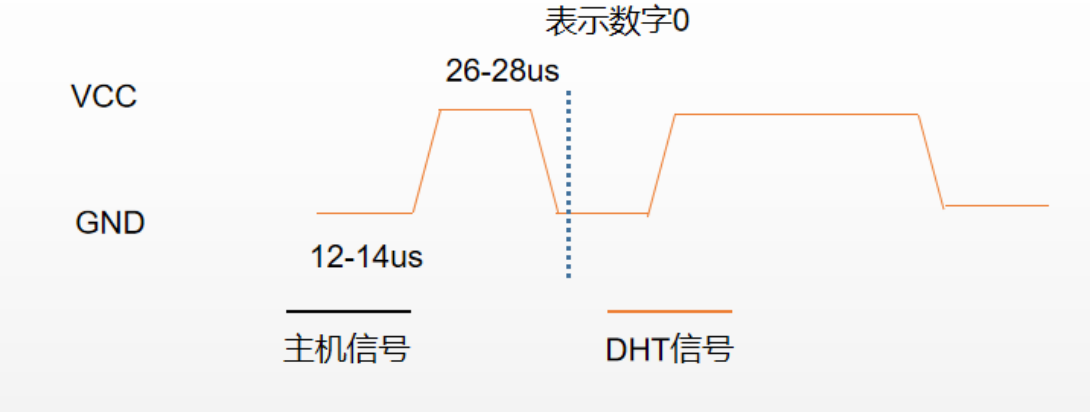

图 3-4 DHT11 传输数字“0”时序

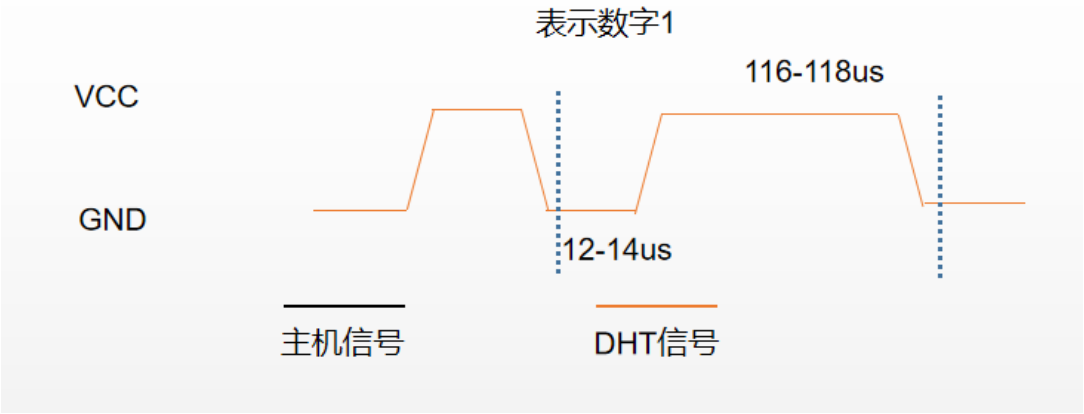

图 3-5 DHT11 传输数字“1”时序

当 DHT11 温湿度感应器的数据包是一次传输 5Byte，即 40bit。字节结构如下图所示：

byte4	byte3	byte2	byte1	byte0
00101101	00000000	00011100	00000000	01001001
整数	小数	整数	小数	校验和
湿度		温度		校验和

图 3-6 DHT11 数据包图解图

如图得出以下计算结果的公式：

湿度= byte4 . byte3=20.0 (%RH)

温度= byte2 . byte1=30.0 (℃)

结果校验：byte4+ byte3+ byte2+ byte1=50(结果正确)

接下来将介绍代码的实现方法。

（1）dht11.h 头文件

主要配置了传输数据的 GPIO 端口，初始化需要调用的函数。需要初始化 GPIOG11 引脚。先定义 IO 口的方向，分别是输入和输出的 GPIO 端口的配置。其次因为需要读取和输出数据，所以还要配置数据寄存器，初始化 GPIOG11 引脚读取和写入的模式。

```
#ifndef __DHT11_H
#define __DHT11_H
#include "sys.h"

//IO方向设置
#define DHT11_IO_IN()  {GPIOG->CRH&=0XFFFF0FFF;GPIOG->CRH|=8<<12;}
#define DHT11_IO_OUT() {GPIOG->CRH&=0XFFFF0FFF;GPIOG->CRH|=3<<12;}
////IO操作函数
#define DHT11_DQ_OUT PGout(11) //数据端口    PA0
#define DHT11_DQ_IN  PGin(11)  //数据端口    PA0


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
```

（2）DHT11.c 文件

里面主要实现了温湿度传感器所需要用到的函数，具体流程如下所示：

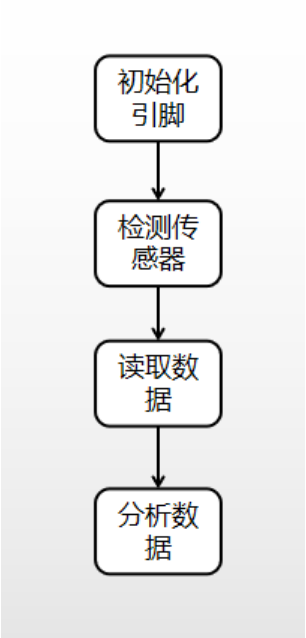

图 3-7 DHT11 传感器工作流程图

（1）调用 DHT11_Read_Data 函数开始读取数据；

（2）用 DHT11_Init 初始化引脚；

（3）用 DHT11_Check 函数检测传感器；

（4）用 DHT11_Read_Byte 函数读取 5 个字节。

（5）通过不同字节分析出湿度数据和温度数据。

各函数代码实现如下所示：

```
u8 DHT11_Init(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOG, ENABLE);  //使能PG端口

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;          //PG11端口配置
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;     //推挽输出
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOG, &GPIO_InitStructure);          //初始化IO口
  GPIO_SetBits(GPIOG,GPIO_Pin_11);              //PG11 输出高

  DHT11_Rst();  //复位DHT11
  return DHT11_Check();//等待DHT11的回应
}
```

上面代码主要初始化 GPIOG11 引脚。并等待模块回应。

```
u8 DHT11_Check(void)
{
   u8 retry=0;
   DHT11_IO_IN();//SET INPUT
     while (DHT11_DQ_IN&&retry<100)//DHT11会拉低40~80us
   {
     retry++;
     delay_us(1);
   };
   if(retry>=100)return 1;
   else retry=0;
     while (!DHT11_DQ_IN&&retry<100)//DHT11拉低后会再次拉高40~80us
   {
     retry++;
     delay_us(1);
   };
   if(retry>=100)return 1;
   return 0;
}
```