





题 目： 基于STM32的智能温控风扇设计与实现

学 院： 信息学院

专 业： 电子信息科学与技术（第二学士学位）

姓 名： 刘洵梓

指导教师： 王玉梅

完成日期： 2023年4月17日

**摘 要**

电风扇常用于对人体或者器件进行降温，但传统的电风扇多采用机械手段控制，存在功能单一、需要手动换挡等问题。随着科学技术的发展和人们生活水平的提高，家用电器产品趋于自动化、智能化、环保化和人性化，使智能电风扇逐渐进入人们的生活。智能温控风扇可根据环境温度自动调节风扇的启停和转速。在日常实际使用中，智能温控风扇不仅可以节省宝贵的电力资源，还给人们的生活提供便利。

为解决传统散热风扇的弊端，对风扇进行智能温控，本设计为一种可以检测周围环境温度并且有效通过控制空气的流动改变温度的智能温控风扇，具有灵敏的温度检测和显示功能。本设计主要基于STM32F103C8T6最小系统板和 DS18B20 温度传感器，通过温度传感器监测目标温度，将所监测的温度值返还给单片机，单片机对温度值进行分析和比较，进而通过 PWM脉冲控制风扇的转速对目标进行降温。该设计能有效地降低被监测对象的温度，避免因温度过高而带来的危害，同时解决了传统人工操作效率低下以及资源浪费的问题。

**关键词**： STM32F103C8T6；温度检测；DS18B20；PWM；智能温控风扇

**Abstract**

Electric fans are often used to cool the human body or devices, but the traditional electric fans are mostly controlled by mechanical means, and there are some problems such as single function and manual shift. With the development of science and technology and the improvement of people's living standards, household electrical products tend to be automatic, intelligent, environmental protection and humanized, so that intelligent electric fans gradually enter people's life. The intelligent temperature control fan automatically adjusts the start, stop, and speed of the fan based on the ambient temperature. In practical daily use, intelligent temperature fan can not only save valuable power resources, but also provide convenience for people's life.

In order to solve the disadvantages of the traditional cooling fan, intelligent temperature control is carried out on the fan. This design is an intelligent temperature control fan which can detect the ambient temperature and effectively change the temperature by controlling the flow of air. It has sensitive temperature detection and display functions. This design is mainly based on STM32F103C8T6 minimum system board and DS18B20 temperature sensor, through the temperature sensor to monitor the target temperature, the temperature value of the monitoring is returned to the single chip microcomputer, the single chip microcomputer to analyze and compare the temperature value, and then control the speed of the PWM fan to cool the target. The design can effectively reduce the temperature of the monitored object, avoid the harm caused by excessive temperature, and solve the problem of low efficiency of traditional manual operation and waste of resources.

**Keywords:** STM32F103C8T6；Temperature detection；DS18B20；Pulse Width Modulation；Intelligent temperature fan

目 录

[序 言 1](#_Toc132613098)

[第一章 系统的总体设计方案 2](#_Toc132613099)

[1.1系统的设计思路 2](#_Toc132613100)

[1.2系统的硬件描述 2](#_Toc132613101)

[1.3系统的软件描述 3](#_Toc132613102)

[第二章 系统的硬件部分 4](#_Toc132613103)

[2.1硬件设计总体说明 4](#_Toc132613104)

[2.2 系统主控单片机介绍 6](#_Toc132613105)

[2.2.1 主控芯片的选择 6](#_Toc132613106)

[2.2.1 STM32F103C8T6系统板简介 6](#_Toc132613107)

[2.2.2 STM32F103C8T6的优势 8](#_Toc132613108)

[2.3 各部分硬件模块介绍 9](#_Toc132613109)

[2.3.1 温度检测模块 9](#_Toc132613110)

[2.3.2 按键交互模块 10](#_Toc132613111)

[2.3.3 液晶显示模块 11](#_Toc132613112)

[2.3.4 电机驱动模块 11](#_Toc132613113)

[2.3.5 USB-TTL串口通信模块 13](#_Toc132613114)

[第三章 系统的软件部分 15](#_Toc132613115)

[3.1 软件设计总体说明 15](#_Toc132613116)

[3.2 各部分功能程序介绍 15](#_Toc132613117)

[3.2.1 初始化程序 15](#_Toc132613118)

[3.2.2 温度采集 16](#_Toc132613119)

[3.2.3 按键监测 17](#_Toc132613120)

[3.2.4 液晶显示 17](#_Toc132613121)

[3.2.5 PWM风扇调速 18](#_Toc132613122)

[3.2.6 串口通信 19](#_Toc132613123)

[第四章 系统的功能测试 20](#_Toc132613124)

[4.1 硬件和软件的调试 20](#_Toc132613125)

[4.2 系统实现的功能 21](#_Toc132613126)

[第五章 总结与展望 22](#_Toc132613127)

[5.1总结 22](#_Toc132613128)

[5.1展望 22](#_Toc132613129)

[参考文献 23](#_Toc132613130)

[附 录 24](#_Toc132613131)

[致 谢 35](#_Toc132613132)

# 序 言

散热风扇在生活中是不可或缺的，但是传统散热风扇只能依靠人工手动开启、关闭以及调整风扇转速，是比较繁琐的。若不进行人为的调整，传统风扇的转速是固定的，当被散热物体的温度下降或上升时，转速不会随之改变。若初始转速设置太小，有可能达不到理想的散热效果，转速设置过大，也会导致资源的浪费，而且较大的转速也会给周围环境带来较大的噪声。所以为解决传统散热风扇的弊端，对风扇进行智能温控很有必要。比如装电脑时，通常会在主板里给CPU风扇和机箱风扇设置一套智能方案：当电脑工作量大，CPU以及主板温度上升时，风扇会增大转速以保证电脑的散热，当电脑工作量少，CPU以及主板温度下降时，为避免产生多余的环境噪音，风扇的转速也会随之下降。因此本次设计采用STM32最小系统与各种外围电路构成单片机温度自动检测和控制系统，实现对温度的实时监测和控制。

本设计是基于STM32F103C8T6的智能温控风扇系统，使用温度传感器DS18B20来检测环境中的温度，将检测数据发送给单片机，单片机将温度信息及对应电机的工作模式输出到OLED显示屏中，使得显示屏显示当前环境的温度并且驱动直流电机工作。若温度低于最低阈值，风扇将不会转动，超过该值风扇则会以对应的转数转动，若温度高于最高阈值，风扇将会全速转动。并且在两阈值温度之间也设定了四个温度区间，在不同区间风扇也会以不同的转速运行，最高最低阈值温度也可通过键盘或者接收电脑发送的信号而改变。

本文第一章介绍系统的总体设计方案，第二章介绍系统的硬件部分，第三章介绍系统的软件部分，第四章讲述系统的功能测试，第五章为总结与展望。

# 第一章 系统的总体设计方案

## 1.1系统的设计思路

本设计的整体思路是利用温度传感器DS18B20完成对温度的精确检测，获取温度信号后给STM32单片机进行处理，并且通过键盘或者电脑通信来设置阈值温度，其中通过输入调整的阈值温度为整数形式，传感器检测的当前环境温度可精确到小数点后两位。当前温度处于不同的温度区间时，采用不同占空比的PWM脉冲来驱动直流风扇电机的转动，以达到有效降温的效果。

## 1.2系统的硬件描述

硬件部分包括STM32单片机模块、温度传感器模块、按键交互模块、OLED液晶显示模块、电机驱动模块和串口通信模块等。

智能温控风扇的硬件结构框图如图1.1所示：

图示

描述已自动生成

图1.1 硬件结构框图

本系统所需要的器件如表 1.1 所示：

表1.1 硬件清单

|  |  |
| --- | --- |
| 名称 | 数量（个） |
| STM32F03CT系统板 | 1 |
| 自设智能风扇拓展板 | 1 |
| DS18N20温度传感器 | 1 |
| 0.96寸OLED | 1 |
| 矩阵按键 | 1 |
| L298N电机驱动板 | 1 |
| 散热风扇 | 1 |
| USB-TTL串口通信模块 | 1 |
| 导线 | 若干 |

## 1.3系统的软件描述

本设计软件系统主要传感器数据的获取以及处理，单片机的控制。运用C语言编程，并采用Keil uVision5集成开发环境，程序搭建采用库函数版本。系统软件主要实现对温度传感器数据的获取、数据的处理、屏幕的显示和风扇电机的控制。整体流程框图如下图1.2所示：

图示

描述已自动生成

图1.2功能整体流程框图

# 第二章 系统的硬件部分

## 2.1硬件设计总体说明

本设计硬件系统主要分为单片机、温度检测、按键交互、液晶显示、串口通信和电机驱动模块五个模块。系统的原理图如图2.1：

图示, 示意图, 日历

描述已自动生成

图2.1系统原理图

系统的连接方式采用拓展板，以下为一块智能风扇系统拓展板，用于各个模块与主控模块的连接，还预设多个端口方便于后续系统功能拓展，拓展板原理图如图2.2：

图示, 示意图

描述已自动生成

图2.2拓展板原理图

拓展板PCB采用两层电路，在 PCB 板布局设计中，布线时首先选择自动布线，然后对布线不合理的地 方采用手动布线进行修改。尽可能的减少甚至消除元件之间引脚的交叉走线、 平行走线所带来的干扰。与此同时，需要把电源线和地线的走线宽度加大或者采取覆铜。如下图2.3和图2.4：

图示

描述已自动生成

图2.3 拓展板PCB（顶层）

电脑屏幕的照片上有字

中度可信度描述已自动生成

图2.4 拓展板PCB（底层）

## 2.2 系统主控单片机介绍

### 2.2.1 主控芯片的选择

主控芯片是智能风扇系统的核心，不仅需要采集、传输和处理数据，还需要控制各种外部设备。根据本设计的功能要求，在选择主控芯片时应考虑以下几个方面的影响:

(1)要考虑芯片的性能和价格是否满足要求，尽量追求低成本高性能;

(2)要考虑需要研究和学习的芯片的时间成本，根据时间成本最低的原则进行选择;

(3)考虑到芯片功耗，结合实际应用，在选择MCU时优先选择功耗低的芯片;

(4)考虑芯片是否包含USART串行异步通信模块、RTC实时时钟模块和定时器模块。

因此，根据本系统的功能需求，采取STM32F103C8T6为本设计的主控芯片十分合适，并且它具有高性能、高兼容、易开发、低功耗、低工作电压等特性。

### 2.2.1 STM32F103C8T6系统板简介

STM32F103C8T6 是基于 ARM Cortex-M3 内核 STM32 系列的 32 位的微控 制器，程序存储器容量是 64 KB[[10]](#_参考文献)，工作温度为-40~85℃，最高工作频率为 72 MHz。 该芯片采用 2 到 3.6 V 供电，I/O 引脚，上电复位、断电复位、可编程电压 监测器，4 到 16 MHz 晶体振荡器，内嵌经出厂调校的 8 MHz 的 RC 振荡器和带 校准的 40 KHz 的 RC 振荡器以及带校准功能的 32 KHz RTC 振荡器，并且可以 产生 CPU 时钟的 PLL。 如 图 3-4 所示为主控芯片 STM32F103C8T6 的最小系统原理图， STM32F103C8T6 芯片的供电电压正常为 3.3 V，需要通过电源滤波电路之后为 芯片供电。 芯片的时钟系统，分为两种： (1)低频时钟源为外接 32768 Hz 的时钟晶体； (2)高频时钟源为外接 8 MHz 的时钟晶体。 低频时钟源一般可以使主控芯片实现准确定时的功能，而高频时钟源能够 提供高效准确的工作时序为主控芯片服务。

STM32系统板电路原理图如下图2.5：

图示, 示意图

描述已自动生成

图2.5 STM32系统板电路原理图

系统板PCB如图2.6：

图形用户界面, 图示

描述已自动生成

图2.6 STM32系统板PCB图

系统板实物图如图2.7：

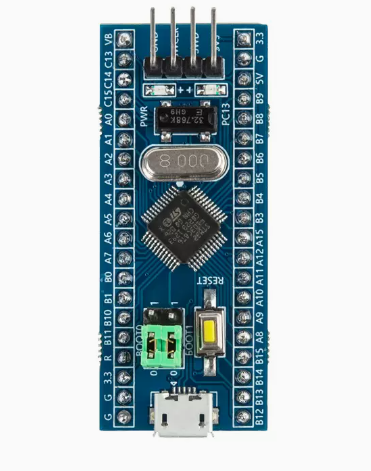


图2.7 STM32系统板

### 2.2.2 STM32F103C8T6的优势

最开始研制 STM32 系列微控制器的目标，是为了提高系统的性能指标，并且降低工作状态的功率损耗。在微控制器领域，STM32 的出现无疑是一个前所 未有的飞跃，与之前的微控制器相比，STM32 在以下多个方面具有明显的优势[[11]](#_参考文献)。

(1)精密性。STM32 是一种相对高端的微控制器，外设分布集中且完备，布 局精致巧妙，器件布置的紧密并且有特色。

(2)可靠性。STM32 的外设布置构局愈来愈精密，对于可靠性的要求却没有因为布局而得到有所降低。伴随着外挂的器件越来越多，STM32 要保持高的可靠性，就需要配备足够的硬件电路，主要有低电压检测器、看门狗和时钟管理器等几部分。

(3)安全性。保证信息的安全，实则就是在信息传递的过程中保证信息不被 泄露。STM32 通过采取锁定 Flash 引脚的措施，保证信息完整且不会被窃取或者 泄露。如果发现有获取芯片内部信息的行为举动，就会立刻拉高引脚状态，同 时会自动消除芯片内部信息，从而保证信息的安全性不被破坏。

(4)在线调试功能。STM32 支持 Tumb-2 指令，在 C 语言环境下能够实现软 件的编译、仿真以及调试。程序在软件平台上编写好后，下载到芯片内部，可 以实现在线调试功能，发现错误的同时进行改正，实时性好。

## 2.3 各部分硬件模块介绍

### 2.3.1 温度检测模块

温度传感器模块由DS18B20可编程数字温度传感器构成，DS18B20是一种很常见的数字温度传感器，采用美国DALLAS半导体公司生产的DS18B20可组网数字温度传感器芯片封装而成。它具有微型化、低功耗、高性能、抗干扰能力强、易配微处理器等优点，可直接将温度转化成串行数字信号供处理器处理，适用于各种狭小空间设备数字测温和控制领域。

DS18B20的主要特征：测量的结果直接以数字信号的形式输出，以“一线总线”方式串行传送给CPU，同时可传送CRC校验码，具有极强的抗干扰纠错能力；温度测量围在-55℃~+125℃之间，在-10℃~+85℃时精度为±0.5℃；可检测温度分辨率为9~12位，对应的可分辨温度分别为0.5℃，0.25℃，0.125℃和0.0625℃，可实现高精度测温；它单线接口的独特性，使它与微处理器连接时仅需一条端口线即可实现与微处理器的双向通信；支持多点组网功能，即多个DS18B20可以并联在唯一的三线上，实现组网多点测温的功能；工作电压围宽，其围在3.0~5.5V。

DS18B20结构主要有四部分：64位ROM、温度传感器、非挥发的温度报警触发器TH和TL、配置寄存器。其管脚有三个，其中DQ为数字信号端，GND为电源地，VDD为电源输入端（图2.8）。

图片包含 文本

描述已自动生成

图2.8 DS18B20引脚图

DS18B20数字温度传感器通过其部计数时钟周期来的作用，实现了特有的温度测量功能。低温系数振荡器输出的时钟信号通过由高温度系数振荡器产生的门周期而被计数，计数器预先置有与-55℃相对应的一个基权值。如果计数器计数到0时，高温度系数振荡周期还未结束，则表示测量的温度值高于-55℃，被预置在-55℃的温度寄存器中的值就增加1℃，然后这个过程不断重复，直到高温度系数振荡周期结束为止。此时温度寄存器中的值即为被测温度值，这个值以16位二进制形式存放在存储器中，通过主机发送存储器读命令可读出此温度值，读取时低位在前，高位在后，依次进行。由于温度振荡器的抛物线特性的影响，其用斜率累加器进行补偿。在本设计中将DS18B20接在PB15口实现温度的采集。其与单片机的连接电路如图2.9。

图示, 示意图

描述已自动生成

图2.9温度传感器模块电路

### 2.3.2 按键交互模块

按键交互模块使用的是4×4矩阵键盘，该按键是轻触式开关仅需很小的力即可轻松开闭，具有段落感好，声音清脆的特点，实物图如图2.10：

图片包含 游戏机, 电子, 电路

描述已自动生成

图2.10 4×4矩阵键盘

本设计选用其中四个按键用于系统控制，按键S1控制控制风扇的启停，按键S2选择所需调整的是最高阈值还是最低阈值，按键S3提高阈值温度，按键S4降低阈值温度。将C1~C4分别连接STM32单片机B11、B10、B1、B0引脚（如图2.11），R1接地，当按键按下时，端口读取低电平有效。系统上电后，进入键盘扫描子程序，以查询的方式确定各按键，完成温度初值的设定。

图示, 表格

描述已自动生成

图2.11矩阵键盘引脚

### 2.3.3 液晶显示模块

屏幕选用的是 4 针9.6寸OLED12864 屏幕（如图2.12），这是一种新型的屏幕技术，拥有单个像素自发光的特性，可视角度大、功耗低、对比度高。对比 LCD 屏幕，OLED 显示效果更好、功耗更低、适合小系统。并且OLED 屏幕支持 3.3V~5V 供电，所以正好可使用系统板上的接口供电。OLED 屏幕包含4个引脚分别是 GND、VCC、SCL和SDA。GND 和 VCC 接地和电源, SCL 连接 PB8, SDA 连接 PB9。OLED 屏幕可以根据需要显示出字符、数字、汉字等，使用起来简单方便。

图形用户界面, 应用程序

描述已自动生成

图2.12 0.96寸OLED\_4P

### 2.3.4 电机驱动模块

电机驱动模块采用的是L298N电机驱动模块，其核心芯片L298N是一种双[H桥](https://baike.baidu.com/item/H%E6%A1%A5/7876818?fromModule=lemma_inlink" \t "_blank)[电机驱动芯片](https://baike.baidu.com/item/%E7%94%B5%E6%9C%BA%E9%A9%B1%E5%8A%A8%E8%8A%AF%E7%89%87/14446800?fromModule=lemma_inlink)，其中每个H桥可以提供2 A的电流，功率部分的[供电电压](https://baike.baidu.com/item/%E4%BE%9B%E7%94%B5%E7%94%B5%E5%8E%8B/5149694?fromModule=lemma_inlink" \t "_blank)范围是2.5 ~ 48 V，逻辑部分5 V供电，接受5 V TTL电平。一般情况下，功率部分的电压应大于6 V否则芯片可能不能正常工作。

1、驱动芯片：L298N 双 H 桥直流电机驱动芯片

2、驱动部分端子供电范围 Vs：＋5 V～＋35 V ； 如需要板内取电，则供电范围

Vs：+7 V～+35 V

3.驱动部分峰值电流 Io：2 A

4.逻辑部分端子供电范围 Vss：＋5 V～＋7 V（可板内取电＋5 V）

5.逻辑部分工作电流范围:0～36 mA

6.控制信号输入电压范围：

低电平：－0.3 V≤Vin≤1.5 V

高电平：2.3 V≤Vin≤Vss

7.使能信号输入电压范围：

低电平：－0.3≤Vin≤1.5 V（控制信号无效）

高电平：2.3 V≤Vin≤Vss（控制信号有效）

8.最大功耗：20 W（温度 T＝75℃时）

9.存储温度：－25℃～＋130℃

10.驱动板尺寸: 48 mm\*43 mm\*33 mm(带固定铜柱和散热片高度)

11.驱动板重量：33 g

12.其他扩展：控制方向指示灯、逻辑部分板内取电接口。

L298N电机驱动模块实物和原理图，如图2.13和图2.14：

桌子上摆放着黑色的机器

低可信度描述已自动生成

图2.13 L298N电机驱动模块

图示, 示意图

描述已自动生成

图2.14 L298N电机驱动模块电路图

直流电机的驱动:

该驱动板可驱动 2 路直流电机，使能端 ENA、ENB 为高电平时有效，控制方式及直流电机状态表如表2.1所示：

表2.1直流电机控制状态表

|  |  |  |  |
| --- | --- | --- | --- |
| ENA|ENB | IN1 | IN2 | 直流电机状态 |
| 0 | X | X | 停止 |
| 1 | 0 | 0 | 停止 |
| 1 | 1 | 0 | 正转 |
| 1 | 0 | 1 | 反转 |
| 1 | 1 | 1 | 停止 |

一般对直流电机进行 PWM 调速，需设置 IN1 和 IN2，确定电机的转动方向，然后对使能端输出 PWM 脉冲，即可实现调速。注意当使能信号为 0 时，电机处于自由停止状态；当使能信号为 1，且 IN1 和 IN2 为 00 或 11 时，电机处于制动状态，阻止电机转动。

本设计采用一个电机风扇，并且该风扇为反转风扇，所以将ENA端口接置为1后，IN1即可控制风扇的启停，IN2接收PWM脉冲信号即可进行PWM调速。其中OUT1和OUT2连接风扇的两端，IN1和IN2连接STM32单片机的PA1和PA2端口。

### 2.3.5 USB-TTL串口通信模块

USB转TTL串口模块是一个非常实用的工具，可以测试模块的UART串口通信和通过单片机的UART接口给单片机等下载程序。能够在电脑上的串口助手软件非常直观的显示出串口设备返回的数据以及发送相应的控制数据给串口设备。常见的有CP2102、PL2303、FT232、CH340等串口芯片方案的USB转串口模块，本设计使用的串口通信模块的串口芯片是CH340，模块引脚如图2.15：

图示

描述已自动生成

图2.15 USB-TTL串口通信模块电路图

其中TXD引脚连接单片机A10接口，RXD引脚连接单片机A9接口， 最后一个引脚接地。

# 第三章 系统的软件部分

## 3.1 软件设计总体说明

本设计控制系统使用C语言编程，使用模块化编程的思想，除主程序外有多个功能子程序，分别控制温度采集、显示功能和电机驱动调速等。

程序总体运行流程如图3.1：

图示

描述已自动生成

图3.1 总体运行流程图

## 3.2 各部分功能程序介绍

### 3.2.1 初始化程序

其中包含OLED显示程序的初始化、温度采集程序的初始化、风扇调速驱动程序的初始化和按键程序的初始化，并且包含温度变量、转速变量、按键信号变量的初始化。

部分程序：

uint8\_t KeyNum; //定义按键信号变量

uint8\_t Speed; //定义风速变量

float temper; //定义温度获取变量

uint8\_t temper\_i, temper\_f; //定义温度显示变量

uint8\_t temper\_max, temper\_min; //定义温度阈值变量

OLED\_Init(); //OLED初始化函数

DS18B20\_Init(); //温度获取初始化函数

Fan\_Init(); //PWN调速风扇初始化函数

Key\_Init(); //按键检测初始化函数

### 3.2.2 温度采集

先对DS18B20初始化，再进行ROM操作命令，最后才能对存储器操作，数据操作。DS18B20每一步操作都要遵循严格的工作时序和通信协议。如主机控制DS18B20完成温度转换这一过程，根据DS18B20的通讯协议，须经三个步骤：每一次读写之前都要对DS18B20进行复位，复位成功后发送一条ROM指令，最后发送RAM指令，这样才能对DS18B20进行预定的操作。温度采集流程如图3.2

图示

描述已自动生成

图3.2温度采集流程

温度采集部分程序：

/\* DS18B20时钟端口、引脚定义 \*/

#define DS18B20\_PORT GPIOB

#define DS18B20\_PIN (GPIO\_Pin\_15)

#define DS18B20\_PORT\_RCC RCC\_APB2Periph\_GPIOB

////IO操作函数

#define DS18B20\_DQ\_OUT PBout(15) //数据端口

#define DS18B20\_DQ\_IN PBin(15) //数据端口

u8 DS18B20\_Init(void); //初始化DS18B20

float DS18B20\_GetTemperture(void); //获取温度

void DS18B20\_Start(void); //开始温度转换

void DS18B20\_Write\_Byte(u8 dat); //写入一个字节

u8 DS18B20\_Read\_Byte(void); //读出一个字节

u8 DS18B20\_Read\_Bit(void); //读出一个位

u8 DS18B20\_Check(void); //检测是否存在DS18B20

void DS18B20\_Reset(void); //复位DS18B20

### 3.2.3 按键监测

基于硬件上预设的4个按键，通过软件程序实现：S1为风扇的开关， S2为最高最低阈值设定的选择，按一次可在后续设定最高阈值，再按一次可在后续设定最低阈值，同时可循环轮转。当按下S3键，对应阈值加1，按下S4键，对应阈值减1。最低阈值温度和最低阈值温度的设置范围为0~100摄氏度，满足一般使用要求，并且最低阈值不可高于最高阈值。按键检测流程如图3.3：

图示

描述已自动生成

图3.3按键检测流程图

### 3.2.4 液晶显示

在完成初始化函数OLED\_Init()后，可调用各个定义好的函数实现液晶屏幕的显示，本系统用到的函数有：屏幕清除函数OLED\_Clear()，显示单个字符函数OLED\_ShowChar(uint8\_t Line, uint8\_t Column, char Char)，显示字符串函数OLED\_ShowString(uint8\_t Line, uint8\_t Column, char \*String)，显示数字函数OLED\_ShowNum(uint8\_t Line, uint8\_t Column, uint32\_t Number, uint8\_t Length)，其中 Line和Column参数设置在屏幕中显示行列，Char，String和Length参数为所要显示的信息。并且在系统测试阶段，液晶显示可用来输出所需调试的参数，有助于系统的完善。

### 3.2.5 PWM风扇调速

本设计采用单片机软件编程实现PWM（脉冲宽度调制）调速的方法。PWM是英文Pulse Width Modulation的缩写，它是按一定的规律改变脉冲序列的脉冲宽度，以调节输出量和波形的一种调节方式，在PWM驱动控制的调节系统中，最常用的是矩形波PWM信号，在控制时需要调节PWM波的占空比。占空比是指高电平持续时间在一个周期时间的百分比。在控制电机的转速时，占空比越大，转速就越快，若全为高电平，占空比为100%时，转速达到最大。若全为低电平，占空比为0%时，停止转动。

当温度小于最小阈值温度MIN时，风扇的转速设为0，当温度大于最大阈值温度MAX时，风扇的转速设为最大。将MIN到MAX区间分成四等分，等分点分别为T1、T2、T3，当温度在这四个区间时，给风扇设定不同的转速值。所以，程序提供总共六档风扇转速如表3.1：

表3.1 温度-挡位对应表

|  |  |  |
| --- | --- | --- |
| 温度区间 | 挡位 | 风扇转速（%） |
| 小于MIN | 0 | 0 |
| MIN ~ T1 | 1 | 20 |
| T1 ~ T2 | 2 | 40 |
| T2 ~ T3 | 3 | 60 |
| T3~MAX | 4 | 80 |
| 大于MAX | ALL | 100 |

调速程序首先确定当前温度所属的档位，调用Fan\_SetSpeed()函数设置对应的风扇速度，再调用PWM\_SetCompare3()函数设置输出脉冲的占空比，由输出脉冲驱动风扇电机使其达到该档位的转速。流程如图3.4：

图表

描述已自动生成

图3.4 风扇调速流程图

### 3.2.6 串口通信

本设计使用USB-TTL串口通信使单片机与电脑连接起来，在电脑上发送数据，可呈现与键盘输入同样的效果，亦可设置风扇的开关，最高最低阈值温度。

在串口初始化程序中定义变量Serial\_RxFlag，来判断是否有来自电脑的信息，变量Serial\_RxPacket接收电脑发送的数据，并用string.h库中的strcmp函数来判断是否接收到所需要的数据信号，是的话就执行相应的操作，否则在电脑端口发送报错。Serial\_SendString()函数可向电脑发送状态信号和报错信号，其流程如图3.5所示：

图示

描述已自动生成

图3.5串口通信流程图

# 第四章 系统的功能测试

## 4.1 硬件和软件的调试

系统调试中为验证DS18B20是否能在系统板上工作，将手心靠拢或者捏住芯片，即可发现OLED显示的温度也迅速升高，验证了DS18B20能在系统板上工作。在本系统中风扇电机的转速可实现6挡调速，通过温度传感器检测的温度位于的区间设定，其屏幕显示效果如图4.1~4.6：

图形用户界面, 文本

描述已自动生成

图4.1 位于0挡屏幕显示

文本

描述已自动生成

图4.1 位于1挡屏幕显示

手表

中度可信度描述已自动生成

图4.1 位于2挡屏幕显示

钟表的特写

描述已自动生成

图4.1 位于3挡屏幕显示

电视游戏的萤幕

描述已自动生成

图4.1 位于4挡屏幕显示

屏幕上写着字

描述已自动生成

图4.1 位于ALL挡屏幕显示

## 4.2 系统实现的功能

本系统能够实现单片机系统检测被测对象温度的变化，然后根据温度和设定的阈值来控制风扇直流电机输入占空比的变化，从而产生不同的转动速度，单片机键盘与电脑上的输入可用于调节最高最低阈值，也可用来开启和关闭风扇。

预设六个转速档位，当温度低于最低阈值时，转速档位为最低档0，风扇停止转动。当温度升高至最高最低阈值之间时，设有四个档位分别对应不同转速，并且其四个档位的分界温度随最高最低阈值温度的设定自动调整。当温度高于最高阈值温度时，调整至最高档位即风扇全速运行。系统还能动态的显示当前温度、最高最低阈值温度，当前的档位以及当前的转速。

# 第五章 总结与展望

## 5.1总结

本次毕业设计的课题为智能温控风扇，本次课题是专业知识与实际很好的一次结合，为了更好的完成这次毕业设计，我们查阅了大量的关于温度控制类书籍、学术论文等相关资料。我们从一开始的无从下手到后来的慢慢理解，通过这中间的经历的确让我们学习到了很多。

本次设计的系统以单片机为控制核心，以温度传感器DS18B20检测环境温度，实现了根据环境温度变化调节不同的风扇电机转速，OLED显示屏能连续稳定显示环境温度、阈值温度、风扇开关状态，风扇档位和风扇转速，并能通过独立按键调节阈值温度，根据当前温度所在的温度区间，控制输出脉冲的占空比调节风扇转速，实现了基于STM32的智能风扇的设计。

本系统可适用于多种散热系统中，如日常的室内散热，温室控温，某一器件的扇热等等，仅需将温度传感器安装到适当的位置，均可实现该位置的智能温控。亦可更换其他的传感器或者电机，实现不同模式的智能控制，比如用上湿度传感器即可智能控制湿度，换上速度传感器和小车电机即可智能控速。综上所述，该系统的设计和研究在社会生产和生活中具有重要地位。

## 5.1展望

本文初步完成了基于STM32的智能温控风扇的设计，经过对该系统的实验发现，还有许多需要改进和完善的地方，具体如下：

（1）硬件系统还可继续拓展，比如添加蓝牙和WIFI等通信模块加入智能家居，实现智能风扇与手机的通信，以达到远程的检测与控制。添加异物检测模块，当人或物体靠近风扇叶片时迅速降低风扇转速至零，以防止事故的发生。

（2）程序设计不够完善，代码还可进一步修改和添加性能更好的控制算法。

# 参考文献

[1]潘勇,孟庆斌.基于DS18B20的多点温度测量系统设计[J].电子测量技术，2008(9):108-112.

[2]薛智宏，赵金，解丽红.DS18B20的测量原理及提高分辨率的方法[J].河北工业科技，2002(6):4~7.

[3]何立民. 单片机应用系统设计[M].北京航天航空大学出版社，1995.

[4]郭炳坤.简单的恒温箱温控电路[J].仪器与未来，1991(7):22.

[5]张少康,尹睿,鲍琦,吴子鑫,高钊.基于单片机的智能温控风扇系统设计[j].电子测试,2019(01):19-20 32.

[6]唐晓珊，管琼，李运彪，黄城兮，王欢，田婷.电风扇自动温控装置设计[J].电子技术，2016，45(12):85-87.

[7]孟萧振，宁秋月，姜宁，裴若男，谢印庆.基于DS18B20的智能温度控制系统[J].电子世界，2021(03):178-179.

[8]基于 STM32 的大棚温湿度监测调节系统设计

[9]陈跃.DS18B20集成温度传感器原理与应用. 安徽: 安徽机电学院学报, 2012.

[10]张瀚中，曹江涛，邵鹏飞，等.基于 WSNs 的温室大棚智能灌溉控制系统设计[J].控制工 程，2019，26(1):108-113.

[11]王晓. 基于STM32F103C8T6电气化铁路供电实训系统研究与设计[D].石家庄铁道大学,2019.

# 附 录

**部分主要程序：**

**main.c:**

#include "stm32f10x.h"

#include "Delay.h"

#include "Key.h"

#include "Buzzer.h"

#include "OLED.h"

#include "Timer.h"

#include "L298N.h"

#include "ds18b20.h"

#include "Serial.h"

#include "string.h"

int main(void)

{

OLED\_Init();

DS18B20\_Init();

Fan\_Init();

Key\_Init();

Buzzer\_Init();

LightSensor\_Init();

uint8\_t KeyNum;

uint8\_t Speed = 0;

uint8\_t temper\_i, temper\_f;

uint8\_t flag = 1, set\_flag = 0;

float temper;

uint8\_t MIN = 20, MAX = 40;

uint8\_t T[4] = {0};

OLED\_ShowString(1, 1, "Temp: . C");

OLED\_ShowString(2, 1, "fan OFF,gear: ");

OLED\_ShowString(3, 1, "MIN C,MAX C");

OLED\_ShowString(4, 1, "Speed:");

// Serial

Serial\_Init();

uint8\_t MyArray[] = {0x42, 0x43, 0x44, 0x45};

Serial\_SendArray(MyArray, 4);

while (1)

{

temper = DS18B20\_GetTemperture();

temper\_i = temper;

temper\_f = (int)(temper \* 100) % 100;

KeyNum = Key\_GetNum();

OLED\_ShowNum(3, 5, MIN, 2);

OLED\_ShowNum(3, 13, MAX, 2);

T[0] = MIN + (MAX - MIN) / 4;

T[1] = MIN + (MAX - MIN) / 2;

T[2] = MIN + (MAX - MIN) \* 3 / 4;

if (Serial\_RxFlag == 1)

{

// OLED\_ShowString(4, 1, " ");

// OLED\_ShowString(4, 1, Serial\_RxPacket);

if (strcmp(Serial\_RxPacket, "fan\_turn") == 0)

{

flag = Fan\_Turn();

if (flag == 0)

{

play\_notice\_UP();

OLED\_ShowString(2, 5, "ON ");

Serial\_SendString("ON\r\n");

}

if (flag == 1)

{

play\_notice\_DOWN();

Serial\_SendString("OFF\r\n");

OLED\_ShowString(2, 5, "OFF");

OLED\_ShowString(4, 8, " 0");

OLED\_ShowString(2, 14, " ");

}

}

else if (strcmp(Serial\_RxPacket, "change") == 0)

{

Serial\_SendString("Change success!\r\n");

Buzzer\_Turn();

set\_flag = set\_flag + 1;

}

else if (strcmp(Serial\_RxPacket, "+") == 0)

{

Serial\_SendString("+\r\n");

Buzzer\_Turn();

if (set\_flag % 2 == 0)

{

if (MIN < MAX)

{

MIN = MIN + 1;

}

}

else

{

if (MAX < 100)

{

MAX = MAX + 1;

}

}

}

else if (strcmp(Serial\_RxPacket, "-") == 0)

{

Serial\_SendString("-\r\n");

Buzzer\_Turn();

if (set\_flag % 2 == 0)

{

if (MIN > 0)

{

MIN = MIN - 1;

}

}

else

{

if (MIN < MAX)

{

MAX = MAX - 1;

}

}

}

else

{

Serial\_SendString("ERROR\_COMMAND\r\n");

// OLED\_ShowString(2, 1, " ");

// OLED\_ShowString(2, 1, "ERROR\_COMMAND");

}

Serial\_RxFlag = 0;

}

if (KeyNum == 1)

{

flag = Fan\_Turn();

if (flag == 0)

{

play\_notice\_UP();

OLED\_ShowString(2, 5, "ON ");

}

if (flag == 1)

{

play\_notice\_DOWN();

OLED\_ShowString(2, 5, "OFF");

OLED\_ShowString(4, 8, " 0");

OLED\_ShowString(2, 14, " ");

}

}

if (KeyNum == 2)

{

Buzzer\_Turn();

set\_flag = set\_flag + 1;

}

if (flag == 0)

{

if (temper < MIN)

{

Speed = 0;

OLED\_ShowString(2, 14, "0 ");

OLED\_ShowString(4, 8, " 0");

}

if (temper > MIN && temper < T[0])

{

Speed = 20;

OLED\_ShowString(2, 14, "1 ");

OLED\_ShowString(4, 8, " 20");

}

if (temper > T[0] && temper < T[1])

{

Speed = 40;

OLED\_ShowString(2, 14, "2 ");

OLED\_ShowString(4, 8, " 40");

}

if (temper > T[1] && temper < T[2])

{

Speed = 60;

OLED\_ShowString(2, 14, "3 ");

OLED\_ShowString(4, 8, " 60");

}

if (temper > T[2] && temper < MAX)

{

Speed = 80;

OLED\_ShowString(2, 14, "4 ");

OLED\_ShowString(4, 8, " 80");

}

if (temper > MAX)

{

Speed = 100;

OLED\_ShowString(2, 14, "ALL");

OLED\_ShowString(4, 8, "100");

}

Fan\_SetSpeed(Speed);

}

if (KeyNum == 3)

{

Buzzer\_Turn();

if (set\_flag % 2 == 0)

{

if (MIN < MAX)

{

MIN = MIN + 1;

}

}

else

{

if (MAX < 100)

{

MAX = MAX + 1;

}

}

}

if (KeyNum == 4)

{

Buzzer\_Turn();

if (set\_flag % 2 == 0)

{

if (MIN > 0)

{

MIN = MIN - 1;

}

}

else

{

if (MIN < MAX)

{

MAX = MAX - 1;

}

}

}

OLED\_ShowNum(1, 7, temper\_i, 2);

OLED\_ShowNum(1, 10, temper\_f, 2);

if (set\_flag % 2 == 0)

{

OLED\_ShowString(3, 4, "-");

OLED\_ShowString(3, 12, " ");

}

else

{

OLED\_ShowString(3, 4, " ");

OLED\_ShowString(3, 12, "-");

}

}

}

**Key.h：**

#ifndef \_\_KEY\_H

#define \_\_KEY\_H

void Key\_Init(void);

uint8\_t Key\_GetNum(void);

#endif

**OLED.h：**

#ifndef \_\_OLED\_H

#define \_\_OLED\_H

void OLED\_Init(void);

void OLED\_Clear(void);

void OLED\_ShowChar(uint8\_t Line, uint8\_t Column, char Char);

void OLED\_ShowString(uint8\_t Line, uint8\_t Column, char \*String);

void OLED\_ShowNum(uint8\_t Line, uint8\_t Column, uint32\_t Number, uint8\_t Length);

void OLED\_ShowSignedNum(uint8\_t Line, uint8\_t Column, int32\_t Number, uint8\_t Length);

void OLED\_ShowHexNum(uint8\_t Line, uint8\_t Column, uint32\_t Number, uint8\_t Length);

void OLED\_ShowBinNum(uint8\_t Line, uint8\_t Column, uint32\_t Number, uint8\_t Length);

#endif

**L298N.h：**

#ifndef \_\_L298N\_H

#define \_\_L298N\_H

void Fan\_Init(void);

void Fan\_ON(void);

void Fan\_OFF(void);

uint8\_t Fan\_Turn(void);

void Fan\_SetSpeed(int8\_t Speed);

#endif

**ds18b20.h：**

#ifndef \_DS18B20\_H\_

#define \_DS18B20\_H\_

#include "stm32f10x.h"

#include "Delay.h"

#include "io\_bit.h"

/\* DS18B20时钟端口、引脚定义 \*/

#define DS18B20\_PORT GPIOB

#define DS18B20\_PIN (GPIO\_Pin\_15)

#define DS18B20\_PORT\_RCC RCC\_APB2Periph\_GPIOB

////IO操作函数

#define DS18B20\_DQ\_OUT PBout(15) //数据端口

#define DS18B20\_DQ\_IN PBin(15) //数据端口

u8 DS18B20\_Init(void); //初始化DS18B20

float DS18B20\_GetTemperture(void); //获取温度

void DS18B20\_Start(void); //开始温度转换

void DS18B20\_Write\_Byte(u8 dat);//写入一个字节

u8 DS18B20\_Read\_Byte(void); //读出一个字节

u8 DS18B20\_Read\_Bit(void); //读出一个位

u8 DS18B20\_Check(void); //检测是否存在DS18B20

void DS18B20\_Reset(void); //复位DS18B20

#endif

Serial.h：

#ifndef \_\_SERIAL\_H

#define \_\_SERIAL\_H

#include <stdio.h>

extern char Serial\_RxPacket[];

extern uint8\_t Serial\_RxFlag;

void Serial\_Init(void);

void Serial\_SendByte(uint8\_t Byte);

void Serial\_SendArray(uint8\_t \*Array, uint16\_t Length);

void Serial\_SendString(char \*String);

void Serial\_SendNumber(uint32\_t Number, uint8\_t Length);

void Serial\_Printf(char \*format, ...);

#endif

# 致 谢

两年短暂而又充实的时光悄悄溜走，入学那刻记忆犹新，而今却又即将告别。有幸在本科毕业之后获得第二学士学位的学习机会，从中南到东北，在不同的环境里延续着我的求学之路。

一朝沐杏雨，一生念师恩。感谢王玉梅老师对我毕业设计指导和帮助，感谢辽宁大学信息学院每一位教导过我的老师。

感谢我的同学们在学习和生活中给我提供的帮助和乐趣，风雨同舟，苦乐共济，感恩相遇，未来可期！

特别感谢我的父母，感谢他们数十年如一日对我不求任何回报的付出、无微不至的照顾。当年孩童的我已然成人，他们却被风霜刻下了皱纹。寸草春晖，山高海深，养育之恩，无以为报。

在接下来的新的征程，我会继续努力。

天辽地大，情怀永存。

山水一程，三生有幸。

刘洵梓

2023年4月 于沈阳