

**单片机应用技术综合组训**

红外温控风扇设计

|  |  |
| --- | --- |
| 部（中心）名称 | 工学部电子信息科教中心 |
| 专业名称 | 电子信息工程 |
| 学生姓名 | 王萌萌/彭硕伟/牛兵 |
| 学生学号 | 2303020052/51/54 |
| 指导教师 | 张具琴 |

**2024**年**5**月**25**日

红外温控风扇设计

摘 要

随着气温的逐渐上升，风扇的需求量也逐渐扩大。传统风扇不能根据外界温度的变化对风扇转速快慢进行调整，也不能对风扇的开关与否进行自动控制，这将会损耗大量的电力资源。针对这些问题，开发设计了红外温控风扇控制系统。

该系统以51单片机最小系统为核心，利用DS18B20温度采集模块，LCD1602显示屏、红外遥控器、按键组成遥控风扇控制系统。该系统设计了三种工作方式，第一种工作方式为按键控制—手动模式；第二种工作方式为自动模式，根据温度自动调节风速；第三种工作方式为红外遥控器控制，在遥控器上按下相应的功能按键，即可控制风扇，手动模式和自动模式切换，以及遥控改变风扇的转速等。

其中自动模式通过DS18B20温度采集模块采集到的温度与系统初始设置的阈值做比较，当温度高于所设置的温度，风扇将会自动吹风；当温度低于所设置的温度时风扇保持关闭状态。采集到的温度数据可以在LCD1602显示屏显示。

文中具体介绍了使用到的各种硬件的特性，风扇的各种功能与详细设计，软件的具体设计思路，各模块的详细介绍，部分模块的程序流程图，关键部分代码的详细讲解等。

关键词：STC89C52，红外遥控，DS18B20

# 目 录

[1 绪论 3](#_Toc25256)

[1.1红外温控风扇的发展与现状 3](#_Toc10125)

[2 系统方案设计 4](#_Toc19731)

[2.1设计任务分析 4](#_Toc13266)

[2.2设计方案 4](#_Toc25088)

[3 系统硬件的设计 7](#_Toc7207)

[3.1单片机最小系统部分 7](#_Toc7117)

[3.2 显示模块LCD1602 8](#_Toc24054)

[3.3 温度模块DS18B20 8](#_Toc12761)

[3.4 按键模块 9](#_Toc3593)

[3.5 红外遥控模块 10](#_Toc15647)

[3.6 蜂鸣器模块 10](#_Toc17246)

[3.7 电机模块 11](#_Toc6861)

[3.8 存储模块 12](#_Toc10220)

[4 系统软件的设计 12](#_Toc2178)

[4.1 软件开发工具 12](#_Toc9986)

[4.2 主程序设计 13](#_Toc9389)

[4.3 DS18B20子程序设计 14](#_Toc6486)

[4.4 按键子程序设计 15](#_Toc18125)

[4.5 红外子程序设计 16](#_Toc24260)

[4.6 电机子程序设计 16](#_Toc11705)

[5 仿真设计 17](#_Toc21633)

[6 系统运行效果 18](#_Toc23716)

[结论 22](#_Toc26745)

[参考文献 23](#_Toc19243)

[附录 24](#_Toc30646)

1 绪论

1.1红外温控风扇的发展与现状

随着科技的不断发展，生活中的传统家电也逐渐向着智能化变迁。电扇作为生活中必不可少的家电，人们也早已不满足于传统电扇温度调节的固定模式，逐渐趋向于能够通过温度变化监控，从而实现智能化温度调节控制的模式。传统电扇虽然也推出了睡眠模式等，但是睡眠模式的温度变化控制遵循的是固定温度控制曲线，并不是根据实际室温变化而进行温度控制。

对于生活生活水平、生活质量不断提高的用户来说，传统电扇根本无法满足人们对舒适感的最高要求。为适应人们的需求，智能控制电风扇也应运而生，我们已经对传统的家用电风扇系统做出了改进，大大改善了传统家用电风扇无法随着气温的变动而调整风速高低的问题，并增加远程遥控功能，让用户能够在不靠近风扇的情况下进行温度设置。该智能温控风扇具有环保、智能、安全高效、方便、性价比高等特点，将为人们的生活带来便利。

红外温控风扇的发展主要体现在智能化和节能方面。我们设计的红外温控风扇可以根据环境温度的变化自动调整风速，例如，当环境温度升高到一定程度时，风扇会自动启动，并随着环境温度的升高而自动加快转速。当环境温度过高，会停止旋转。这种智能调节不仅提供了舒适的使用体验还有助于节省能源。此外，这种风扇还配备了数码管或LED屏，能够显示当前的环境温度和工作状态，增强了产品的人性化特点。

在国际上虽然国外在电风扇方面的研究不如中国积极，但在智能化电器领域却更为成功。智能化电器包括智能化的电器元件，如智能化断路器，智能化接触器等。这些都是未来发展的趋势。新型的智能化电器元件采用微处理器和可编程元件，大量功能通过软件实现，并具备“现场”设计的能力。

总体来看红外遥控风扇的发展正朝着更加智能化，节能化的方向前进，旨在提供更好的用户体验和更高效的能源利用。

2 系统方案设计

2.1设计任务分析

本文对传统的风扇中存在着功能单一等问题，设计出一款操作简单，可遥控的温控风扇，本次设计的主要功能如下：

1. 采用PWM调速的原理来实现风扇速度的控制，当温度高于上限温度时蜂鸣器产生报警，当温度低于温度的下限值时，风扇不转，当温度高于下限温度时风扇转动，同时温度每升高1℃风扇档位自动调节占空比来提高风速。
2. 当室内温度超过设定的最低温度值时风扇开始转动。根据温度的改变来实时调节占空比来控制转速的改变。温度可设置上下限值：从左到右按键依次是：加、减、档位加、档位减、切换（手动模式和自动模式切换）。按下“加”或“减”键后对需要设置的温度进行设置，手动模式下按下档位加、档位减则对风扇档位进行加减（一共四个档位，0档则风扇关闭），按下模式切换则对风扇进行手动模式和自动模式进行切换。
3. 可通过红外遥控器实现按键的所有功能（温度设置，档位加减，手自动模式切换）。
4. LCD1602显示当前温度，温度上下限，以及当前风扇档位和模式。
5. 当室内温度超过设定的高温度值时蜂鸣器会发出报警声响来提示温度达到指定值。

该系统以STC89C52单片机为核心，通过温度采集模块，LCD1602显示屏、按键，红外遥控器控制模块构成。温度采集模块主要将采集到的温度值与系统设置的初始温度值进行比较。若采集到的温度值大于初始温度值，则风扇将会开启；若采集到的温度值小于初始温度值，风扇将一直保持关闭状态。LCD1602显示屏主要显示风扇档位，温度、模式等数据。自动模式下采集到的温度大于初始温度值，则风扇转动；反之，风扇关闭。手动模式下风扇转速和温度无关，需要通过按键调节风速转速。

系统的硬件方案如图1所示。

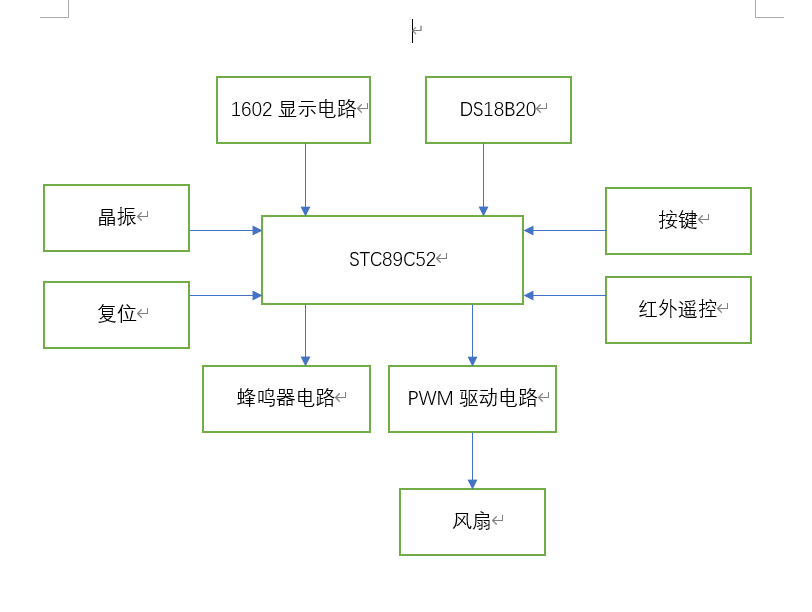


图1 整体框图

## 2.2设计方案

2.2.1 主控芯片的选择

本系统以STC89C52 芯片为核心控制器，该芯片是一款八位 CMOS 微处理器，性 能高、低损耗。内部存储器容量为 4k 字节，可进行快速擦除和写入程序，并且最多支 持 1000 次的重复写入或擦除，数据可长达十年保存。相比于 MCS-51 系列单片机， STC89C52 不仅能完全替代，还能为系统赋予更多功能。该芯片不易损坏，只需一个电 源即可运行，且兼具便于编程和硬加密的优势。同时，该芯片也支持外部存储器的读写 操作。因此，采用STC89C52芯片作为系统的核心控制器是成本、体积和可靠性等方面 的最佳选择。

2.2.2 显示模块的选择

LCD1602 具有高质量和始终恒定的发光特点，每个像素点在收到信号后拥有稳定的 物理性质，并且不会闪烁。此外，由于 LCD1602 采用数字化液晶显示技术，这类技术 使得其接口简单，操作简便。该液晶模块可通过控制电极状态来实现显示效果，相比传 统显示屏，其重量更轻，且能耗更低。综合考虑以上因素，本设计方案决定采用LCD1602 作为显示模块。

2.2.3 步进电机模块的选择

步进电机可以完成对位置非常准确的操作，它是一中非常特殊的电机。用步进电机 可以通过控制器发出信号，信号传到驱动器里，在驱动器的驱动下以一个特定的角度来 转动。一种方法是通过不断发送脉冲信号来令步进电机连续运转一定距离，同时，通过 改变脉冲频率来直接控制电机的转速。由于其具有原理简单易学、易于使用等的优点， 所以步进电机极其的适合用于单片机控制。步进电机已经广泛应用于各种控制设备中。

3 系统硬件的设计

3.1单片机最小系统部分

在单片机的选择上，选用STC89C52芯片作为主控制器。STC89C52是一种低功耗、高性能CMOS8位微控制器，具有8K在系统可编程Flash存储器。

STC89C52单片机最小系统电路由复位电路、时钟电路和电源电路。拥有这三部分电路后，单片机即可正常工作。

VCC和GND为单片机的电源引脚，为单片机提供电源。复位电路由按键S1、电解电容EC1和电阻R1组成。具有手动按键复位和上电自动复位功能。系统上电复位按键接口采集到两个高端信号后进行手动复位，就是非自动的按键复位；系统检测到的电压由低电平上升到高电平的一段时间后，在这段时间过后，系统通过电阻与接地之间形成一条通路，然后自动把高电平进行拉低，使得单片机从高电位变为低电位，从而就是给单片机自动进行复位即上电复位。时钟电路由晶振Y1、瓷片电容C1和C2组成。有控制芯片的数字电路正常工作是少不了TIME(时钟)电路的，我们需要时钟电路自动发出系统时间，让控制芯片正常工作。给控制芯片正常工作的时钟信号，一般把这种工作方式称为“拍”，以至于让整个控制系统能正常工作，由于要保证控制系统能正常工作，提高他的工作能力，我们经常用11.0592MHZ晶振和30PF的电容进行组合[2]，电容为了帮助晶振起振的，满足了数字控制器上电以后可以正常工作。JD1为单片机的下载接口。单片机最小系统部分如图2所示。



图2 单片机最小系统原理图

## 3.2 显示模块LCD1602

LCD1602是内部集成字库的字符型液晶显示模块，内含32字节显示区RAM，一个字节对应屏幕上的一个字符，如图3-2所示。LCD1602内的CGROM已经存储了160种字符的编码，可以直接查表获得。在操作时，需要先确定显示的地址，即光标地址，再送入需要显示的字符编码。LCD1602显示存储空间如图3所示。

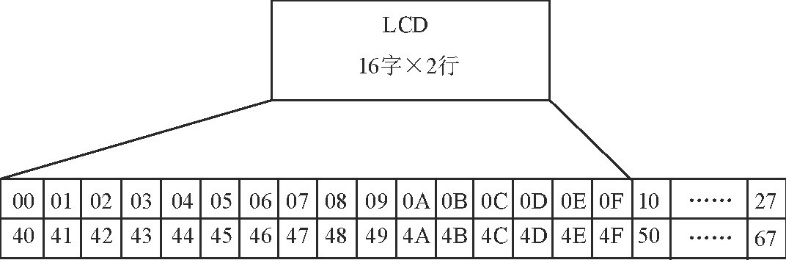


图3 LCD1602显示存储空间图

单片机与LCD1602的通信分为读状态、写指令、读数据、写数据四种操作模式，将EN、RW、RS引脚分别连接单片机的P3.4~P3.6，以控制操作模式。将P35引脚RS置低、P36引脚RW置高，为读状态模式，此时D0~D6脚输出当前地址指针值，D7脚输出忙信号。将RS、RW置低，为写指令模式，可以通过D0~D7脚写入指令。将RS、RW置高，为读数据模式，此时D0~D7脚输出当前指针所指向的RAM数据。将RS置高、RW置低，为写数据模式，通过D0~D7脚向当前指针所指向的RAM写入数据。RS、RW设置正确后，给使能P34引脚EN上升沿并保持，才能开始读、写操作，LCD1602原理图如图4所示。



图4 LCD1602操作时序图

## 3.3 温度模块DS18B20

DS18B20是常用的数字温度传感器，其输出的是数字信号，具有体积小，硬件开销低，抗干扰能力强，精度高具有以下特点：1.独特的单线接口方式，DS18B20在与微处理器连接时仅需要一条口线即可实现微处理器与DS18B20的双向通讯。2.测温范围 －55℃～+125℃，固有测温误差1℃。3.工作电源: 3.0~5.5V/DC （可以数据线寄生电源）4.支持多点组网功能，多个DS18B20可以并联在唯一的三线上，最多只能并联8个，实现多点测温，如果数量过多，会使供电电源电压过低，从而造成信号传输的不稳定。原理图如图5所示。



图5 DS18B20模块原理图

## 3.4 按键模块

与独立按键不同，4x4矩阵键盘是通过行扫描和列扫描的方式，用8个I/O口实现16个按键的检测。硬件原理图如图6所示，行线为P30~P33，列线为P34~P37。



图6 键盘原理图

51单片机的P3口不是标准意义的双向口，是准双向口。双向口与准双向口的区别主要是：准双向口I/O口操作时做数据输入时需要对其置1。准双向口只能有效的读取0，而对1则是采用读取非零的方式，就是读入的时候要先向接口上写1，然后再读。

具体原理为：向I/O口写1即为给锁存器写1，那么锁存器的反向端就输出0，和它相接的MOS管也就在截止状态，也就是呈高阻态，这样P0口上数据就会从读引脚的三态缓冲器上正确的输入。如果不向I/O口写1，那么锁存器上次锁存的可能为0，那么反向端有可能出现1，这样和反向端相接的MOS管导通，直接拉到地，那么不管你这个I/O口上输入什么信号都会被拉低，无法获得正确的输入结果。

## 3.5 红外遥控模块

红外线遥控器已被广泛使用在各种类型的家电产品上，一般由红外发射装置和红外接收设备两大部分组成。红外发射装置又可由键盘电路、红外编码芯片、电源和红外发射电路组成。红外接收设备可由红外接收电路 红外解码芯片、电源和应用电路组成。通常为了使信号能更好的被传输发送端将基带二进制信号调制为脉冲串信号，通过红外发射管发射。常用的有通过脉冲宽度来实现信号调制的脉宽调制（PWM）和通过脉冲串之间的时间间隔来实现信号调制的脉时调制（PPM）两种方法。

开发板所使用的TC9012红外遥控器，编码方式为PPM，发送的每一帧数据包含引导码、用户码、数据码。

TC9012晶振频率455KHz，经内部分频电路，将信号调制在37.91KHz通过红外发射，通过板载红外接收模块HS0038解调，由单片机软件解码。红外遥控模块原理图如图7所示。



图7 红外遥控模块原理图

## 3.6 蜂鸣器模块

本设计采用有源蜂鸣器，三极管Q3起开关的作用，它的基极低电平让三极管饱和导通，让蜂鸣器发出声响；还有基极高电平会让三极管停止工作，使蜂鸣器停止发出声响。当遥控器按键按下时，蜂鸣器报警一次。蜂鸣器硬件原理图如图8所示。



图8 蜂鸣器模块原理图

## 3.7 电机模块

ULN2003是大电流驱动阵列，具有电流增益高、工作电压高、温度范围宽、带负载能力强等特点，适应于各类要求高速大功率驱动的系统。ULN2003由七个硅NPN达林顿管组成。每一对达林顿都串联一个2.7K的基极电阻，在5V的工作电压下它能与TTL和CMOS电路直接相连，可以直接处理原先需要标准逻辑缓冲器来处理的数据。多用于单片机、智能仪表、PLC、数字量输出卡等控制电路中。可直接驱动继电器等负载。

本系统使用一片ULN2003来实现对有刷直流电机的驱动，内部构造图如图9所示。

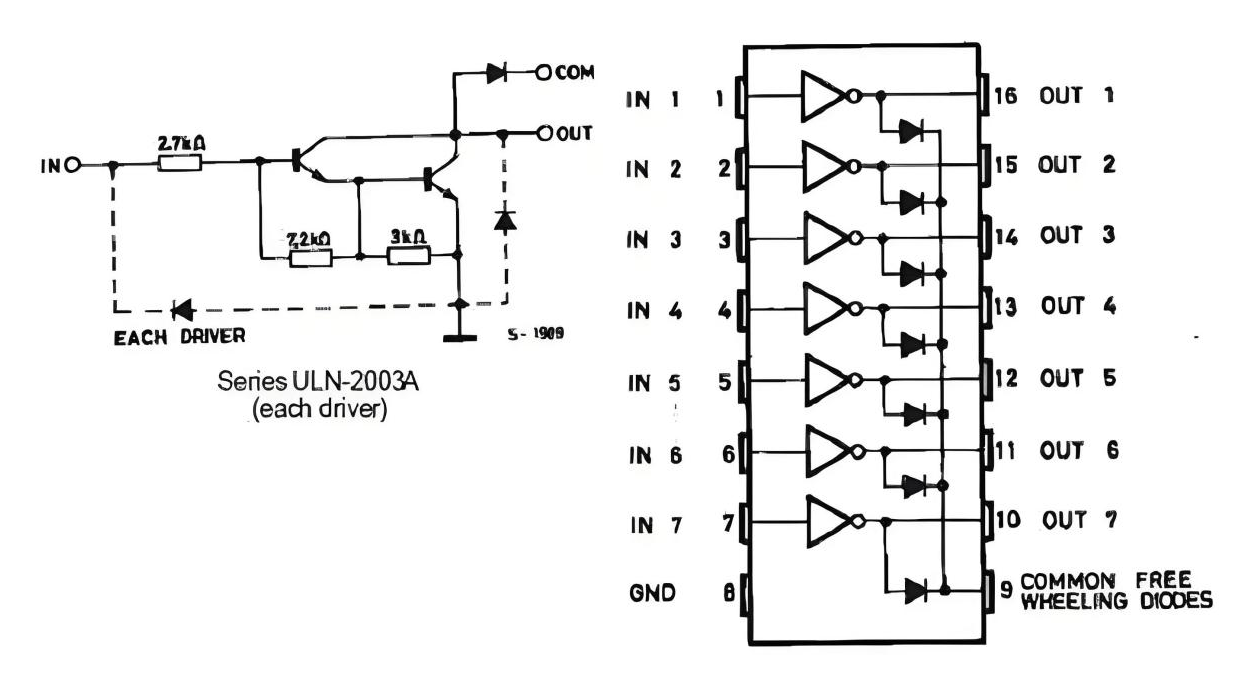


图9 ULN2003内部构造图

ULN2003硬件原理图如图10所示。



图10 电机模块原理图

## 3.8 存储模块

STC89C52运行中产生的数据存储在片内RAM中，这些数据在芯片掉电后会一并丢失。如果有些需要灵活读写且长期保存的数据，就需要将这些数据存储在EEPROM中。24C02是一个2K位串行CMOS EEPROM， 内部含有256个8位字节，一个8字节页写缓冲器。该器件通过IIC总线接口进行操作，对于STC89C52单片机，则需要通过软件IIC实现操作。硬件原理图如图11所示。



图11 存储原理图

4 系统软件的设计

## 4.1 软件开发工具

本系统的软件设计主要使用以下工具：

1. Keil5：用于编写和编译程序代码。
2. STC-ISP：用于将编译好的程序下载到单片机中。

这些工具的安装和配置步骤如下：

1. 安装Keil5：安装的是Keil5 C51版本，因为这是专门为51单片机设计的开发环境。
2. 安装STC-ISP：该软件用于将编译后的程序下载到单片机中。
3. 其他工具（可选）：如有道词典和福昕阅读器等辅助工具。

## 4.2 主程序设计

为了根据当前温度实时控制风扇的状况，需要检测当前温度值是否超过程序中设制的动作温度值。由于单片机的工作频率高达12MHz，程序运行时不断比较和检测当前温度和调整后的动作温度。当超过设定温度值范围时，转去执行温度处理子程序，并在温度处理下，实时控制风机停机、低风、强风状态。显示驱动程序检查七段代码，以获得显示屏中应显示的数字，并逐位扫描和显示。主程序流程图如图12所示。

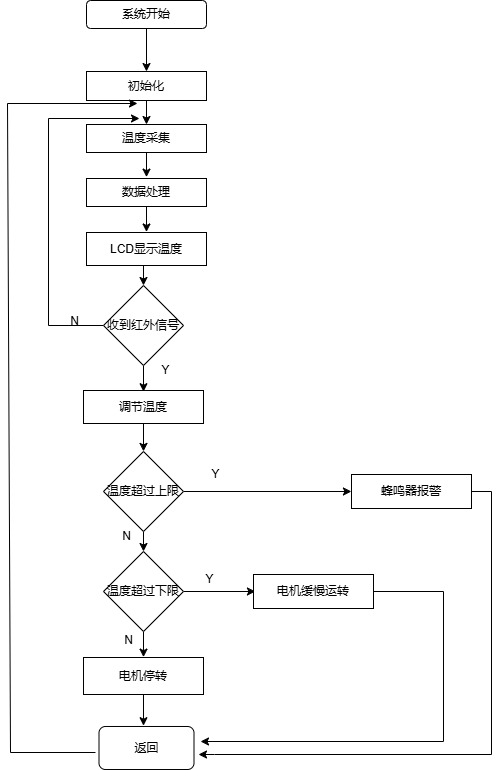


图12 主程序详细流程图

系统的软件流程可以分为以下几个步骤：

1. 系统初始化：上电后执行main函数，进行必要的初始化操作，包括I/O端口、定时器、中断和显示模块的初始化。
2. 主循环设计：主循环采用switch语句，根据不同的运行状态（如自动模式、手动模式、自然风模式）执行相应的功能模块。自动模式：根据温度传感器采集的温度值自动调节风扇转速。手动模式：通过按键或红外遥控手动调节风扇转速。自然风模式：模拟自然风的变化，使风扇转速随机变化。
3. 中断服务程序：使用定时器0、定时器1和外部中断0来处理不同的任务，如计时、红外信号捕捉和温度测量。系统的中断资源使用了定时器0中断、定时器1中断、外部中断0。定时器0中断。定时器0中断每20ms触发一次，根据当前温度，当计时位累计达到一定值时将更新一次电机占空比。外部中断0用于捕捉红外遥控信号的下降沿，并打开定时器1计算红外遥控信号的低电平持续时间，如果检测到有效的引导码，则开始记录接下来的低电平时长数据存储在irdata[]中。

## 4.3 DS18B20子程序设计

首先初始化DS18B20，然后运行ROM操作命令，最后操作内存和数据。操作DS18B20的每个步骤遵循通信协议和工作时序。如果控制DS18B20完成温度转换过程，则必须根据DS18B20通信协议进行以下三个步骤：在每次读写之前重新启动DS18B20。复位成功后，发送ROM指令，最后发送RAM指令只有这样才可能在DS18B20上执行预定操作。DS18B20程序流程图如图13所示。

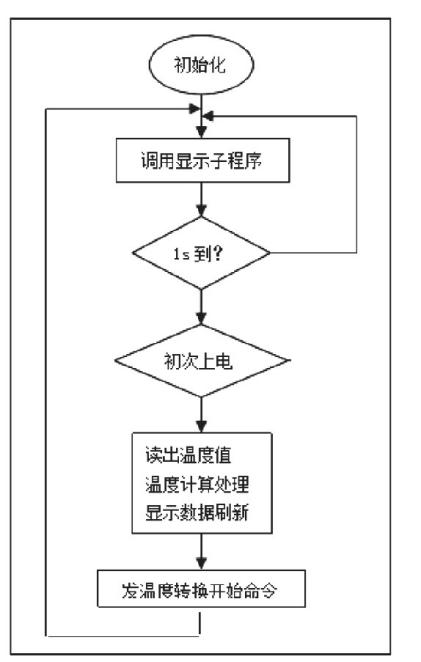


图13 DS18B20温度函数程序框图

## 4.4 按键子程序设计

硬件设计通过矩阵按键，扫描子程序提供软件支持。第一次按下K1键，进入温度控制模式设置温度上限值，按“加号”键K2加一，然后按“减号”键K3减1。再次按K1调整键，进入温度下限设定状态，按“加号”K4键加一，按“减号”K5键减一。最低作用温度值和最高作用温度值的调节范围为1-100度满足一般使用要求，再次按K1调整键退出上下限设置温度状态。按键程序流程图如图14所示。

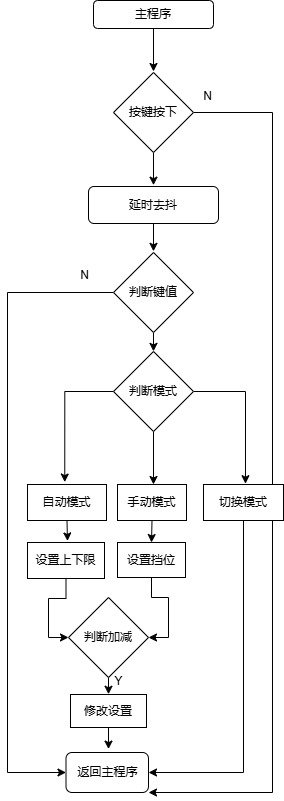


图14 按键扫描函数程序框图

## 4.5 红外子程序设计

通过红外信号和单片机通信, 通过串口中断来实现。在主程序中, 开始“flag=0”, 风扇工作在自动模式, 单片机每隔20ms访问一次串口中断程序, 判断模式切换键是否按下, 如果是则flag取反, 即“flag=1”;如果“flag=1”, 则风扇工作在手动模式, 温度不对风扇的工作方式起作用。在手动模式中, 可以选择风扇的工作方式, 也可以切换工作方式选择自动模式。红外接收函数编写，只需循环检测按键端口。如果相应的端口位于高电平时，说明端口被按下，函数返回对应的档位；如果端口皆为低电平，则输入0，此时无法实现遥控功能，红外子程序流程图如图15所示。

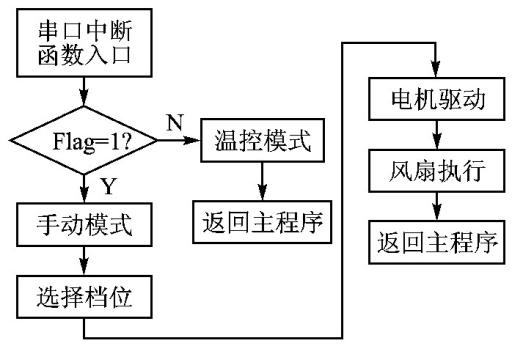


图15 红外扫描函数程序框图

## 4.6 电机子程序设计

风扇电机的速度是根据单片机P1.0口输输出的PWM脉冲控制的。利用定频调宽的方法改变PWM占空空比。PWM波的产生是依靠单片机的定时器0来发生的, 在主程序中, 单片机定时器0每隔20ms清零一次并重新计时。中断定时时间和预设的PWM变量值进行比较, 大于预设值输出0, 小于PWM值则输出1, 以此方式来输出高低脉冲, 产生PWM波。在主程序中, 温度的大小会改变PWM变量的值。因此变量PWM是一个可改变的量, 以此产生脉宽调制信号。图17为PWM波产生子程序流程图。每次产生一个中断时，中断控制量n就会做减1操作，当n为0 时，则将控制门关闭，从而过零脉冲也就无法通过。当n不等于0时，控制电平保持为“1”，脉冲信号由导通电压的T1秒和断开开电压的T2秒组成，占空比D=T1/T2，电机的平均电压为：：Ud=D\*Um。因此，定频调宽的原理就是保持T不变，单片机通过改变T1,T2来调节占空比，从而使风扇电机获得不同的平均电电压，达到调节速度的效果。电机驱动子程序流程图如图16所示。

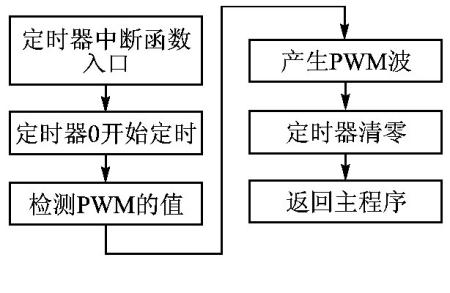


图16 电机驱动函数程序框图

5 仿真设计

根据设计需要，程序设计了系统初始化、温度采集与读取、LCD1602显示当前温度与风速档位、按键扫描、报警预处理、PWM电机控制等主要功能。本系统程序采用C语言编写，在美国Keil Software公司开发的Keil u Vision5软件环境下进行编译与调试。Keil C51拥有丰富的数据库函数，支持51系列单片机的开发与应用。仿真采用英国Labcenter electronics公司出版的Proteus软件。Proteus是目前世界上最先进，最完整的嵌入式系统设计与仿真平台。它是一种可视化的支持多种机型号单片机（如51，PIC,AVR,Motorolahcll等），并且支持与当前流行的单片机开发环境（Keil，MPLBA,IRB）连接调试的软硬件仿真系统，使用Proteus制作出原理图，选择程序后运行加载程序，得到的结果如图17所示。当前环境温度为28.12 ℃，处于设置的温度上限值和下限值之间，蜂鸣器停止，风扇低速转动，LED灯亮，通过加值键和减值键调整温度阈值来控制风扇转速。在仿真过程中添加一个上拉排阻，它的作用为：增强输出引脚的驱动能力，提高输出电平；提高芯片输入信号的噪声容限，增强抗干扰能力，提高输出的高电平值。

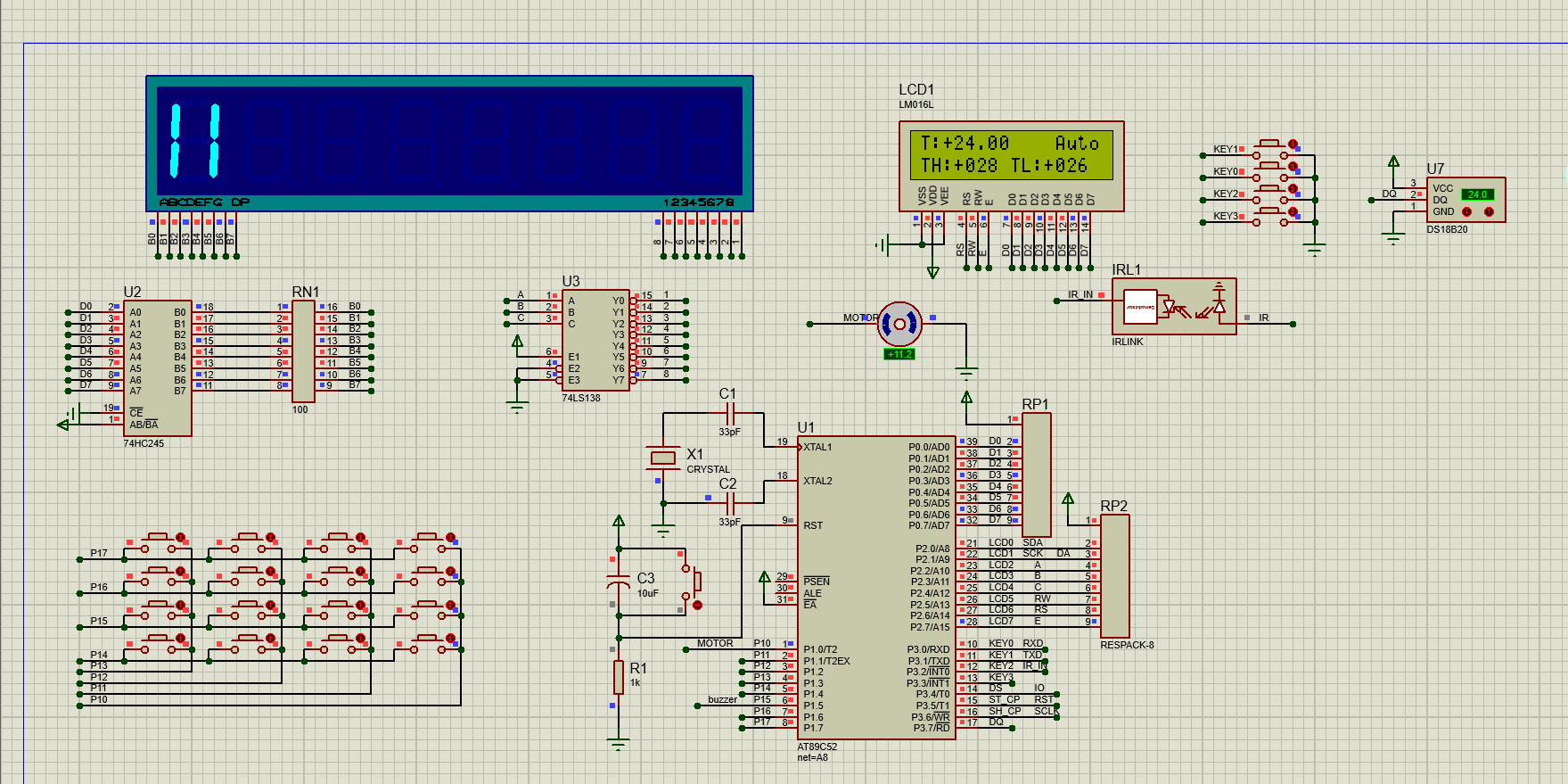


图17 Proteus虚拟仿真图

6 系统运行效果

系统的软件方面通过Keil软件进行编写，将编写好的程序生成.HEX文件后通过下载器下载到单片机中。通过观察整个系统运行的状态，然后对程序进行修改和调试，最终能够获到一个完整的程序。在实验中发现了LCD与矩阵键盘的总线冲突乱码问题，为了解决这个问题，在扫描LCD之前将单片机P0口置位0xff，以避免送入不确定的数据，同时将矩阵键盘扫描程序置于LCD刷新程序之前，这样即便导致了显示错误，也能在一次主循环中及时刷新掉。在调试过程中发现按键按下抖动会产生误差，致使实验数据产生偏差，所以加入延时函数判断按键按下与松开的状态，以此来消除因抖动带来的误差。为了使风扇的温度阈值能够在加减键之间进行切换，所以添加了一个设置切换按键。单片机系统整体模块展示，包含了STC89C52单片机主板，以及外设LCD1602显示屏、电机风扇、红外遥控。整体外观如图18所示。

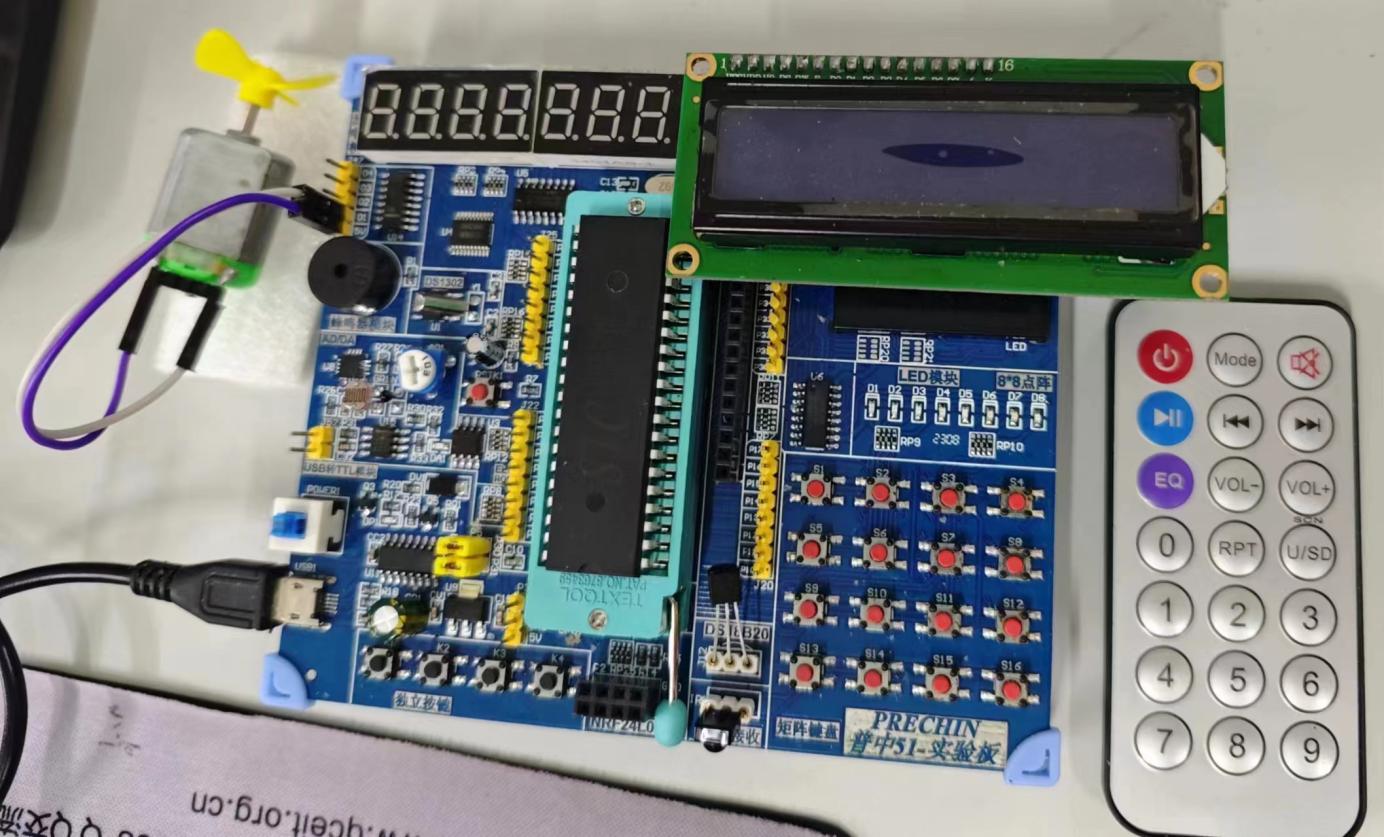


图18 整体界面

系统上电之后，进入自动模式界面，第一行左边显示“T:”后边的数字代表当前室内温度，第一行右边显示“温度和模式如Auto”，第二行显示“TH，TL”字样，表示设定的最高和最低温度可通过按键和红外遥控改变，如图19所示。

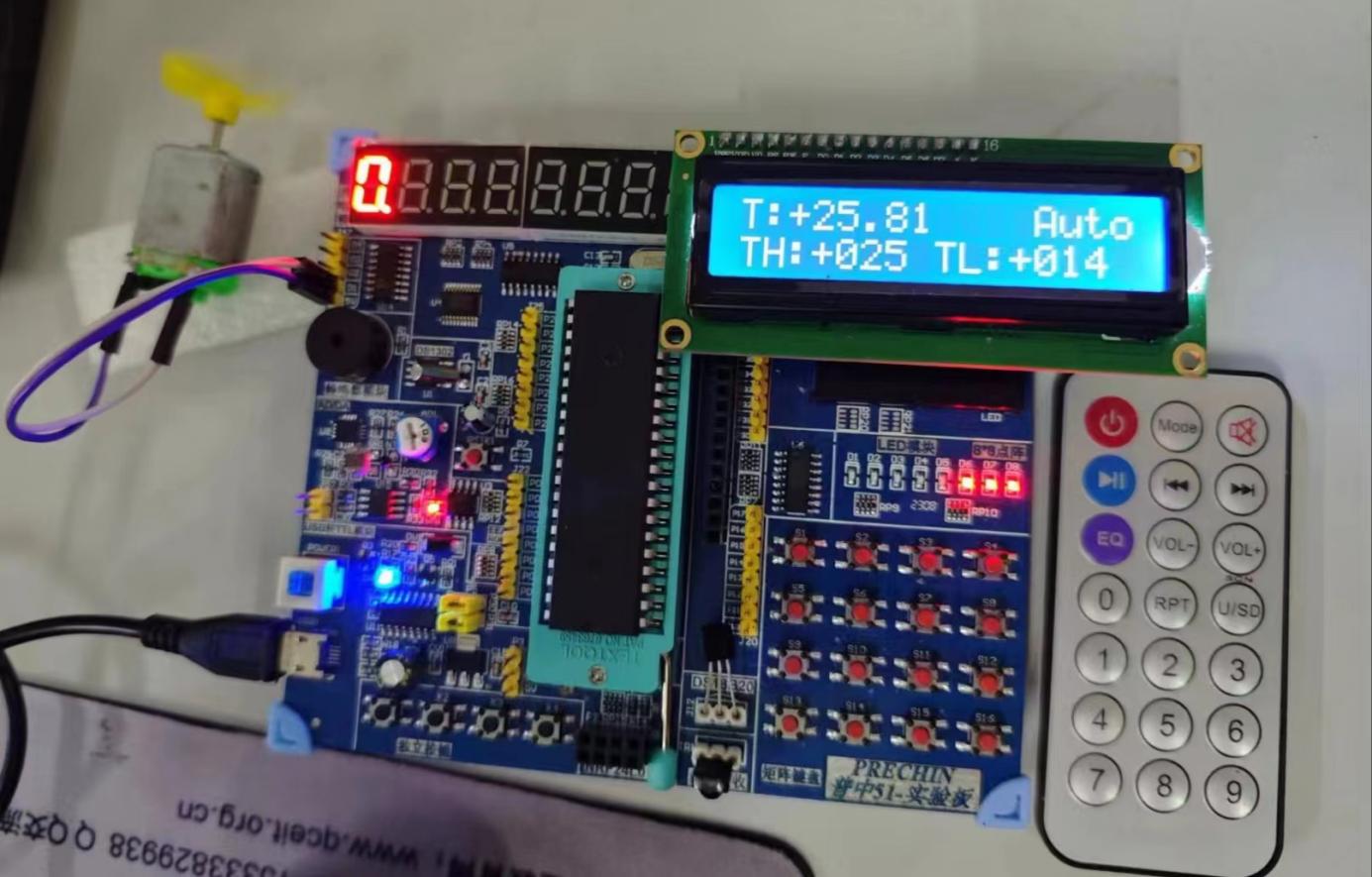


图19 自动界面

切换模式后，进入手动模式界面，第一行右边显示“Hand”表示当前为手动模式，第二行显示“Gear”字样，表示当前风扇转速的挡位，可通过按键和红外遥控改变，图中09表示矩阵键盘按键键码为9，如图20所示。

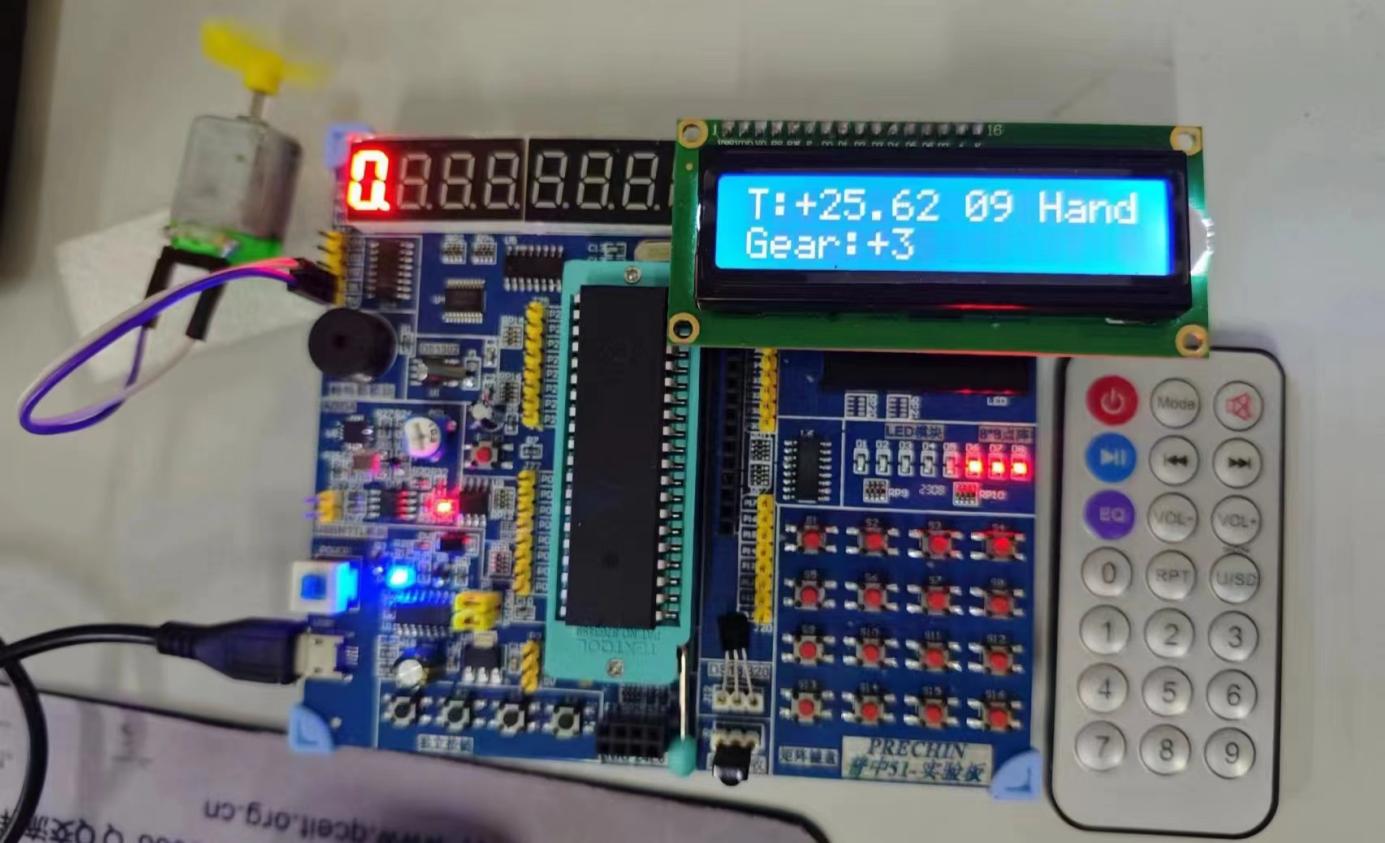


图20 手动界面

自然风模式界面，第一行右边显示“N W”表示当前为自然风模式，第二行显示“Speed”字样，表示当前风扇转速的大小，模拟自然风速转动，转动期间不可调节如图21所示。

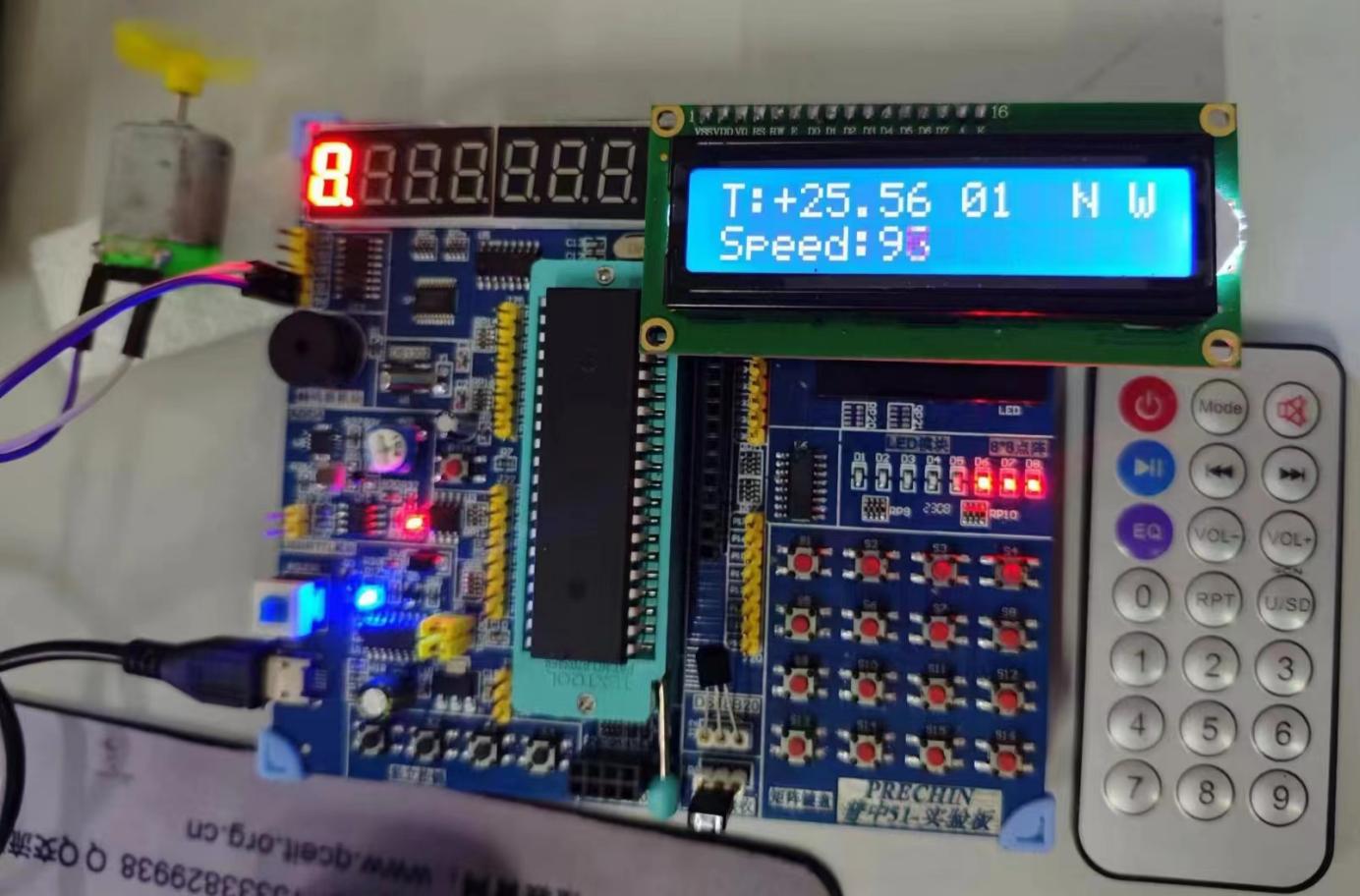


图21 自然风界面

结 论

为适应现代人在炎炎夏日中对舒适度的高需求，并根据现代家居生活需求更偏向于的方便、环保、安全等特点，本文作者设计出了一个能够随着周围温度变化而自动控制速度的风扇系统，将智能控制与环保节能相结合，更加符合现代科技发展的大方向。该设计以单片机控制器STC89C51为核心，使用了DS18B20和LCD1602的温度采集与显示方案，并同时应用了PWM电机调制原理和红外无线遥控技术，具有控制功能齐备、耗电少、造价低、自动化程度较高、原理和程序设计简单等的特点。实现了智能温控风扇系统，监测到的温度值会被返还给单片机；单片机通过分析和比较接收到的温度值来控制风扇的转速。该系统能够监测环境温度并针对温度变化做出相应的调整措施，解决了以前需要人工操作且效率低下的问题，为社会生产和生活提供了便利。

# 参考文献

[1]宋德杰.AT24C02在单片机中的应用[J].电子制作,2005,(04):17-19.[8]修成竹.光敏传感器的控制方法及电子设备.

[2]范兵.基于STC89C52单片机的学习开发板[P].四川省2023-08-01.

[3]李宗瑛.I2C在单片机中的应用[J].数字技术与应用,2018,36(02):17-18.

[4]段旭阳,韩延义,王晓丹,等.基于FPGA的LCD1602液晶显示屏系统控制设计[J].工业和信息化教育,2018,(03):56-60.

[5]王赛,张立新,王军昂,等.基于STC单片机的红外遥控设计[J].电子测试,2022,36(09):26-28.

[6]徐敏.基于STC89C52单片机的蜂鸣器的设计与实现[J].无线互联科技,2022,19(16):95-98.  
[7]李艳芳,and 李治慧.基于单片机的无刷直流电机控制系统设计.电子测试 .21(2021):

[8]魏雪义,陈明虎,唐雨潇.一种基于单片机的键盘控制系统及方法[P].江苏省:,2023-05-16.

[9]郑振杰,江衍煊,游德智.单片机结合ULN2003驱动步进电机[J].电机技术,2010,(06):44-46.

附录



图22 原理图

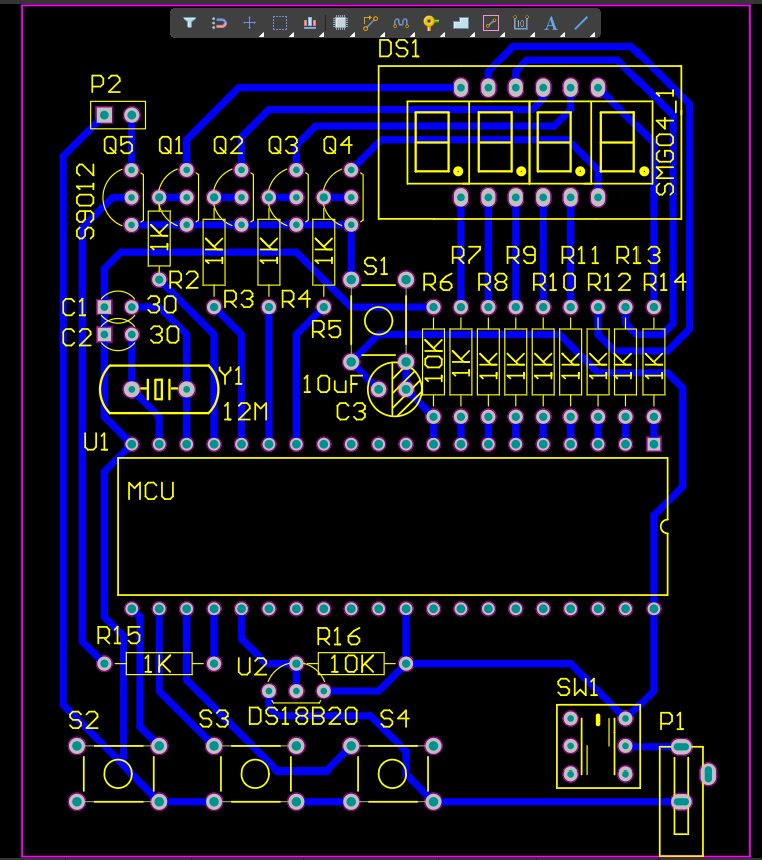


图23 PCB板图

全部程序

1、主程序（main）

#include <REGX52.H>

#include "Delay.h"

#include "LCD1602.h"

#include "DS18B20.h"

#include "AT24C02.h"

#include "Key.h"

#include "Nixie.h"

#include "Motor.h"

#include "IR.h"

#include "Buzzer.h"

#include "MatrixKey.h"

//温度显示存储

float T,TShow;

char TLow,THigh;

//红外电机

unsigned char Command,Speed;

//模式控制

unsigned char Mode;

unsigned char KeyNum;

void Thermosensation() {

//设定的最小温度大于室温则开启风扇

if(TShow>= TLow ) {

if(Speed == 0) {

Speed = 1;

}

// 无级变速逻辑

Motor\_SetSpeed(TShow); //改变占空比

//ShowCompare(); //显示占空比

}else{

Speed = 0;

Motor\_SetSpeed(0);

}

//高于设定温度则会进行报警

if(THigh >= TShow) {

Buzzer\_Time(100);

}

}

void main()

{

//温度模块

DS18B20\_ConvertT(); //上电先转换一次温度，防止第一次读数据错误

Delay(1000); //等待转换完成

THigh=AT24C02\_ReadByte(0); //读取温度阈值数据

TLow=AT24C02\_ReadByte(1);

if(THigh>125 || TLow<-55 || THigh<=TLow)

{

THigh=20; //如果阈值非法，则设为默认值

TLow=15;

}

Mode = AT24C02\_ReadByte(2); //读取模式设置

if(Mode>3 || Mode<=0 )

{

Mode = 1; //如果阈值非法，则设为默认值

}

LCD\_Init();

LCD\_ShowString(1,1,"T:");

/\*LCD\_ShowString(2,1,"TH:");

LCD\_ShowString(2,9,"TL:");

LCD\_ShowSignedNum(2,4,THigh,3);

LCD\_ShowSignedNum(2,12,TLow,3);

\*/

//红外模块

Motor\_Init();

IR\_Init();

while(1)

{

//温度模块

/\*温度读取及显示\*/

DS18B20\_ConvertT(); //转换温度

T=DS18B20\_ReadT(); //读取温度

if(T<0) //如果温度小于0

{

LCD\_ShowChar(1,3,'-'); //显示负号

TShow=-T; //将温度变为正数

}

else //如果温度大于等于0

{

LCD\_ShowChar(1,3,'+'); //显示正号

TShow=T;

}

LCD\_ShowNum(1,4,TShow,2); //显示温度整数部分

LCD\_ShowChar(1,6,'.'); //显示小数点

LCD\_ShowNum(1,7,(unsigned long)(TShow\*100)%100,2);//显示温度小数部分

//LCD\_ShowString(2,13,"S: "); //电机显示速度

//设定的最小温度大于室温则开启风扇

KeyNum = MatrixKey(); //获取按键数

if(IR\_GetDataFlag() || KeyNum /\*|| IR\_GetRepeatFlag()\*/) //如果收到数据帧//或者收到连发帧 || 按键

{

if(KeyNum) {

LCD\_ShowNum(1,10,KeyNum,2);

//获取遥控器命令码

}

Command=IR\_GetCommand(); //获取遥控器命令码

if(Command == IR\_MODE || KeyNum == 1) { //模式切换

//自动模式

if(Mode == 1) {

LCD\_ShowString(1,13,"Auto");

LCD\_ShowString(2,1," ");

}

//遥控模式

if(Mode == 2) {

LCD\_ShowString(1,13,"Hand");

LCD\_ShowString(2,1," ");

}

//自然风模式

if(Mode == 3) {

LCD\_ShowString(1,13," N W");

LCD\_ShowString(2,1," ");

}

Mode += 1;

if (Mode > 3 || Mode <= 0) // 循环切换--

{

Mode = 1;

//ModeDelta = -ModeDelta;

}

AT24C02\_WriteByte(2, Mode);

Delay(5);

//LCD\_ShowString(1,13,"auto");

}

/\*阈值判断及显示\*/

if(Command==IR\_NEXT || KeyNum == 2) //K2按键，THigh自增

{

THigh++;

if(THigh>125){THigh=125;}

}

if(Command==IR\_PREVIOUS || KeyNum == 3) //K3按键，THigh自减

{

THigh--;

//if(THigh<=TLow){THigh++;}

}

if(Command==IR\_VOL\_ADD || KeyNum == 4) //K4按键，TLow自增

{

TLow++;

//if(TLow>=THigh){TLow--;}

}

if(Command==IR\_VOL\_MINUS || KeyNum == 5) //K5按键，TLow自减

{

TLow--;

if(TLow<-55){TLow=-55;}

}

LCD\_ShowSignedNum(2,4,THigh,3); //显示阈值数据

LCD\_ShowSignedNum(2,12,TLow,3);

AT24C02\_WriteByte(0,THigh); //写入到At24C02中保存

Delay(5);

AT24C02\_WriteByte(1,TLow);

Delay(5);

if(Command==IR\_0 || Command==IR\_POWER || KeyNum == 6){Speed=0;} //根据遥控器命令码设置速度

if(Command==IR\_1 || KeyNum == 7){Speed=1;}

if(Command==IR\_2 || KeyNum == 8){Speed=2;}

if(Command==IR\_3 || KeyNum == 9){Speed=3;}

if(Speed==0){Motor\_SetSpeed(0);} //速度输出

if(Speed==1){Motor\_SetSpeed(25);}

if(Speed==2){Motor\_SetSpeed(75);}

if(Speed==3){Motor\_SetSpeed(100);}

}

switch (Mode){

case 1:

LCD\_ShowString(1, 13, "Auto");

LCD\_ShowString(2, 1, "TH:");

LCD\_ShowString(2, 9, "TL:");

LCD\_ShowSignedNum(2, 4, THigh, 3);

LCD\_ShowSignedNum(2, 12, TLow, 3);

Thermosensation();

break;

case 2:

LCD\_ShowString(1, 13, "Hand");

LCD\_ShowString(2,11," ");

LCD\_ShowString(2, 1, "Gear:");

LCD\_ShowSignedNum(2, 6, Speed, 1);

break;

case 3:

LCD\_ShowString(2,11," ");

LCD\_ShowString(1,13," N W");

MotorAddSpeed();

MotorReduceSpeed();

break;

}

//LCD\_ShowNum(1,15,Speed, 2); //LCD1602显示当前速度

//Nixie(1,Speed); //数码管显示速度\*/

}

}

2、存储模块（24C02）

#include <REGX52.H>

#include "I2C.h"

#define AT24C02\_ADDRESS 0xA0

/\*\*

\* @brief AT24C02写入一个字节

\* @param WordAddress 要写入字节的地址

\* @param Data 要写入的数据

\* @retval 无

\*/

void AT24C02\_WriteByte(unsigned char WordAddress,Data)

{

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS);

I2C\_ReceiveAck();

I2C\_SendByte(WordAddress);

I2C\_ReceiveAck();

I2C\_SendByte(Data);

I2C\_ReceiveAck();

I2C\_Stop();

}

/\*\*

\* @brief AT24C02读取一个字节

\* @param WordAddress 要读出字节的地址

\* @retval 读出的数据

\*/

unsigned char AT24C02\_ReadByte(unsigned char WordAddress)

{

unsigned char Data;

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS);

I2C\_ReceiveAck();

I2C\_SendByte(WordAddress);

I2C\_ReceiveAck();

I2C\_Start();

I2C\_SendByte(AT24C02\_ADDRESS|0x01);

I2C\_ReceiveAck();

Data=I2C\_ReceiveByte();

I2C\_SendAck(1);

I2C\_Stop();

return Data;

}

3、延时函数

void Delay(unsigned int xms)

{

unsigned char i, j;

while(xms--)

{

i = 2;

j = 239;

do

{

while (--j);

} while (--i);

}

}

4、红外遥控

#include <REGX52.H>

#include "Timer0.h"

#include "Int0.h"

unsigned int IR\_Time;

unsigned char IR\_State;

unsigned char IR\_Data[4];

unsigned char IR\_pData;

unsigned char IR\_DataFlag;

unsigned char IR\_RepeatFlag;

unsigned char IR\_Address;

unsigned char IR\_Command;

/\*\*

\* @brief 红外遥控初始化

\* @param 无

\* @retval 无

\*/

void IR\_Init(void)

{

Timer0\_Init();

Int0\_Init();

}

/\*\*

\* @brief 红外遥控获取收到数据帧标志位

\* @param 无

\* @retval 是否收到数据帧，1为收到，0为未收到

\*/

unsigned char IR\_GetDataFlag(void)

{

if(IR\_DataFlag)

{

IR\_DataFlag=0;

return 1;

}

return 0;

}

/\*\*

\* @brief 红外遥控获取收到连发帧标志位

\* @param 无

\* @retval 是否收到连发帧，1为收到，0为未收到

\*/

unsigned char IR\_GetRepeatFlag(void)

{

if(IR\_RepeatFlag)

{

IR\_RepeatFlag=0;

return 1;

}

return 0;

}

/\*\*

\* @brief 红外遥控获取收到的地址数据

\* @param 无

\* @retval 收到的地址数据

\*/

unsigned char IR\_GetAddress(void)

{

return IR\_Address;

}

/\*\*

\* @brief 红外遥控获取收到的命令数据

\* @param 无

\* @retval 收到的命令数据

\*/

unsigned char IR\_GetCommand(void)

{

return IR\_Command;

}

//外部中断0中断函数，下降沿触发执行

void Int0\_Routine(void) interrupt 0

{

if(IR\_State==0) //状态0，空闲状态

{

Timer0\_SetCounter(0); //定时计数器清0

Timer0\_Run(1); //定时器启动

IR\_State=1; //置状态为1

}

else if(IR\_State==1) //状态1，等待Start信号或Repeat信号

{

IR\_Time=Timer0\_GetCounter(); //获取上一次中断到此次中断的时间

Timer0\_SetCounter(0); //定时计数器清0

//如果计时为13.5ms，则接收到了Start信号（判定值在12MHz晶振下为13500，在11.0592MHz晶振下为12442）

if(IR\_Time>12442-500 && IR\_Time<12442+500)

{

IR\_State=2; //置状态为2

}

//如果计时为11.25ms，则接收到了Repeat信号（判定值在12MHz晶振下为11250，在11.0592MHz晶振下为10368）

else if(IR\_Time>10368-500 && IR\_Time<10368+500)

{

IR\_RepeatFlag=1; //置收到连发帧标志位为1

Timer0\_Run(0); //定时器停止

IR\_State=0; //置状态为0

}

else //接收出错

{

IR\_State=1; //置状态为1

}

}

else if(IR\_State==2) //状态2，接收数据

{

IR\_Time=Timer0\_GetCounter(); //获取上一次中断到此次中断的时间

Timer0\_SetCounter(0); //定时计数器清0

//如果计时为1120us，则接收到了数据0（判定值在12MHz晶振下为1120，在11.0592MHz晶振下为1032）

if(IR\_Time>1032-500 && IR\_Time<1032+500)

{

IR\_Data[IR\_pData/8]&=~(0x01<<(IR\_pData%8)); //数据对应位清0

IR\_pData++; //数据位置指针自增

}

//如果计时为2250us，则接收到了数据1（判定值在12MHz晶振下为2250，在11.0592MHz晶振下为2074）

else if(IR\_Time>2074-500 && IR\_Time<2074+500)

{

IR\_Data[IR\_pData/8]|=(0x01<<(IR\_pData%8)); //数据对应位置1

IR\_pData++; //数据位置指针自增

}

else //接收出错

{

IR\_pData=0; //数据位置指针清0

IR\_State=1; //置状态为1

}

if(IR\_pData>=32) //如果接收到了32位数据

{

IR\_pData=0; //数据位置指针清0

if((IR\_Data[0]==~IR\_Data[1]) && (IR\_Data[2]==~IR\_Data[3])) //数据验证

{

IR\_Address=IR\_Data[0]; //转存数据

IR\_Command=IR\_Data[2];

IR\_DataFlag=1; //置收到连发帧标志位为1

}

Timer0\_Run(0); //定时器停止

IR\_State=0; //置状态为0

}

}

}

5、键盘

#include <REGX52.H>

#include "Delay.h"

unsigned char MatrixKey\_KeyNumber;

/\*\*

\* @brief 获取按键键码

\* @param 无

\* @retval 按下按键的键码，范围：0,1~16,0表示无按键按下

\*/

unsigned char MatrixKey(void)

{

unsigned char Temp=0;

Temp=MatrixKey\_KeyNumber;

MatrixKey\_KeyNumber=0;

return Temp;

}

/\*\*

\* @brief 矩阵键盘读取按键键码

\* @param 无

\* @retval KeyNumber 按下按键的键码值

如果按键按下不放，程序会停留在此函数，松手的一瞬间，返回按键键码，没有按键按下时，返回0

\*/

unsigned char MatrixKey\_GetState()

{

unsigned char KeyNumber=0;

P1=0xFF;

P1\_3=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=1;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=5;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=9;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=13;}

P1=0xFF;

P1\_2=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=2;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=6;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=10;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=14;}

P1=0xFF;

P1\_1=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=3;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=7;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=11;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=15;}

P1=0xFF;

P1\_0=0;

if(P1\_7==0){Delay(20);while(P1\_7==0);Delay(20);KeyNumber=4;}

if(P1\_6==0){Delay(20);while(P1\_6==0);Delay(20);KeyNumber=8;}

if(P1\_5==0){Delay(20);while(P1\_5==0);Delay(20);KeyNumber=12;}

if(P1\_4==0){Delay(20);while(P1\_4==0);Delay(20);KeyNumber=16;}

return KeyNumber;

}

/\*\*

\* @brief 按键驱动函数，在中断中调用

\* @param 无

\* @retval 无

\*/

void MatrixKey\_Loop(void)

{

static unsigned char NowState,LastState;

LastState=NowState; //按键状态更新

NowState=MatrixKey\_GetState(); //获取当前按键状态

//如果上个时间点按键按下，这个时间点未按下，则是松手瞬间，以此避免消抖和松手检测

if(LastState==1 && NowState==0)

{

MatrixKey\_KeyNumber=1;

}

if(LastState==2 && NowState==0)

{

MatrixKey\_KeyNumber=2;

}

if(LastState==3 && NowState==0)

{

MatrixKey\_KeyNumber=3;

}

if(LastState==4 && NowState==0)

{

MatrixKey\_KeyNumber=4;

}

if(LastState==5 && NowState==0)

{

MatrixKey\_KeyNumber=5;

}

if(LastState==6 && NowState==0)

{

MatrixKey\_KeyNumber=6;

}

if(LastState==7 && NowState==0)

{

MatrixKey\_KeyNumber=7;

}

if(LastState==8 && NowState==0)

{

MatrixKey\_KeyNumber=8;

}

if(LastState==9 && NowState==0)

{

MatrixKey\_KeyNumber=9;

}

if(LastState==10 && NowState==0)

{

MatrixKey\_KeyNumber=10;

}

if(LastState==11 && NowState==0)

{

MatrixKey\_KeyNumber=11;

}

if(LastState==12 && NowState==0)

{

MatrixKey\_KeyNumber=12;

}

if(LastState==13 && NowState==0)

{

MatrixKey\_KeyNumber=13;

}

if(LastState==14 && NowState==0)

{

MatrixKey\_KeyNumber=14;

}

if(LastState==15 && NowState==0)

{

MatrixKey\_KeyNumber=15;

}

if(LastState==16 && NowState==0)

{

MatrixKey\_KeyNumber=16;

}

}

6、显示模块（LCD1602）

#include "LCD1602.h"

/\*=================================================

\*函数名称：LCD1602\_Read\_Busy

\*函数功能：判断1602液晶忙，并等待

=================================================\*/

void LCD1602\_Read\_Busy()

{

u8 busy;

LCD1602\_DB = 0xff;//复位数据总线

LCD1602\_RS = 0; //拉低RS

LCD1602\_RW = 1; //拉高RW读

do{

LCD1602\_EN = 1;//使能EN

busy = LCD1602\_DB;//读回数据

LCD1602\_EN = 0; //拉低使能以便于下一次产生上升沿}while(busy & 0x80); //判断状态字BIT7位是否为1，为1则表示忙，程序等待}

/\*=================================================

\*函数名称：LCD1602\_Write\_Cmd

\*函数功能：写LCD1602命令

\*调用：Read\_Busy();

\*输入：cmd:要写的命令

=================================================\*/

void LCD1602\_Write\_Cmd(u8 cmd)

{LCD1602\_Read\_Busy(); //判断忙，忙则等待

LCD1602\_RS = 0;

LCD1602\_RW = 0; //拉低RS、RW操作时序情况1602课件下中文使用说明基本操作时序章节

LCD1602\_DB = cmd;//写入命令

LCD1602\_EN = 1; //拉高使能端 数据被传输到LCD1602内

LCD1602\_EN = 0; //拉低使能以便于下一次产生上升沿

}

/\*=================================================

\*函数名称：LCD1602\_Write\_Dat

\*函数功能：写LCD1602数据

\*调用：Read\_Busy();

\*输入：dat：需要写入的数据

=================================================\*/

void LCD1602\_Write\_Dat(u8 dat)

{

LCD1602\_Read\_Busy();

LCD1602\_RS = 1;

LCD1602\_RW = 0;

LCD1602\_DB = dat;

LCD1602\_EN = 1;

LCD1602\_EN = 0;

}

/\*=================================================

\*函数名称：LCD1602\_Dis\_OneChar

\*函数功能：在指定位置显示一个字符

\*调用：LCD1602\_Write\_Cmd(); LCD1602\_Write\_Dat();

\*输入：x：要显示的横坐标取值0-40，y：要显示的行坐标取值0-1（0为第一行,1为第二行）

dat：需要显示的数据以ASCLL形式显示

=================================================\*/

void LCD1602\_Dis\_OneChar(u8 x, u8 y,u8 dat)

{

if(y) x |= 0x40;

x |= 0x80;

LCD1602\_Write\_Cmd(x);

LCD1602\_Write\_Dat(dat);

}

/\*=================================================

\*函数名称：LCD1602\_Dis\_Str

\*函数功能：在指定位置显示字符串

\*调用：LCD1602\_Write\_Cmd(); LCD1602\_Write\_Dat();

\*输入：x：要显示的横坐标取值0-40，y：要显示的行坐标取值0-1（0为第一行,1为第二行）

\*str：需要显示的字符串

=================================================\*/

void LCD1602\_Dis\_Str(u8 x, u8 y, u8 \*str)

{

if(y) x |= 0x40;

x |= 0x80;

LCD1602\_Write\_Cmd(x);

while(\*str != '\0')

{

LCD1602\_Write\_Dat(\*str++);

}

}

/\*=================================================

\*函数名称：LCD1602\_Init

\*函数功能：1602初始化

\*调用： LCD1602\_Write\_Cmd();

=================================================\*/

void LCD1602\_Init()

{

LCD1602\_Write\_Cmd(0x38); // 设置16\*2显示，5\*7点阵，8位数据接口

LCD1602\_Write\_Cmd(0x0c); //开显示

LCD1602\_Write\_Cmd(0x06); //读写一字节后地址指针加1

LCD1602\_Write\_Cmd(0x01); //清除显示

}

/\*=================================================

\*函数名称：LCD1602\_clean

\*函数功能：1602清屏

\*调用： LCD1602\_Write\_Cmd();

=================================================\*/

void LCD1602\_clean()

{

LCD1602\_Write\_Cmd(0x01);

}

/\*=================================================

\*函数名称：\*u8\_to\_str

\*函数功能：将一个字节的数据转换为字符串

=================================================\*/

u8 strHEX[3];

u8 \*u8\_to\_str(u8 num)

{u8 i = 0;

if(num/16!=0)

switch(num/16){

case 0: strHEX[i] = '0'; i++; break;

case 1: strHEX[i] = '1'; i++; break;

case 2: strHEX[i] = '2'; i++; break;

case 3: strHEX[i] = '3'; i++; break;

case 4: strHEX[i] = '4'; i++; break;

case 5: strHEX[i] = '5'; i++; break;

case 6: strHEX[i] = '6'; i++; break;

case 7: strHEX[i] = '7'; i++; break;

case 8: strHEX[i] = '8'; i++; break;

case 9: strHEX[i] = '9'; i++; break;

case 10: strHEX[i] = 'A'; i++; break;

case 11: strHEX[i] = 'B'; i++; break;

case 12: strHEX[i] = 'C'; i++; break;

case 13: strHEX[i] = 'D'; i++; break;

case 14: strHEX[i] = 'E'; i++; break;

case 15: strHEX[i] = 'F'; i++; break;}

switch(num%16){

case 0: strHEX[i] = '0'; i++; break;

case 1: strHEX[i] = '1'; i++; break;

case 2: strHEX[i] = '2'; i++; break;

case 3: strHEX[i] = '3'; i++; break;

case 4: strHEX[i] = '4'; i++; break;

case 5: strHEX[i] = '5'; i++; break;

case 6: strHEX[i] = '6'; i++; break;

case 7: strHEX[i] = '7'; i++; break;

case 8: strHEX[i] = '8'; i++; break;

case 9: strHEX[i] = '9'; i++; break;

case 10: strHEX[i] = 'A'; i++; break;

case 11: strHEX[i] = 'B'; i++; break;

case 12: strHEX[i] = 'C'; i++; break;

case 13: strHEX[i] = 'D'; i++; break;

case 14: strHEX[i] = 'E'; i++; break;

case 15: strHEX[i] = 'F'; i++; break;}

strHEX[i] = '\0';

return (strHEX);}

7、DS18B20温度模块

#include <REGX52.H>

#include "OneWire.h"

//DS18B20指令

#define DS18B20\_SKIP\_ROM 0xCC

#define DS18B20\_CONVERT\_T 0x44

#define DS18B20\_READ\_SCRATCHPAD 0xBE

/\*\*

\* @brief DS18B20开始温度变换

\* @param 无

\* @retval 无

\*/

void DS18B20\_ConvertT(void)

{

OneWire\_Init();

OneWire\_SendByte(DS18B20\_SKIP\_ROM);

OneWire\_SendByte(DS18B20\_CONVERT\_T);

}

/\*\*

\* @brief DS18B20读取温度

\* @param 无

\* @retval 温度数值

\*/

float DS18B20\_ReadT(void)

{

unsigned char TLSB,TMSB;

int Temp;

float T;

OneWire\_Init();

OneWire\_SendByte(DS18B20\_SKIP\_ROM);

OneWire\_SendByte(DS18B20\_READ\_SCRATCHPAD);

TLSB=OneWire\_ReceiveByte();

TMSB=OneWire\_ReceiveByte();

Temp=(TMSB<<8)|TLSB;

T=Temp/16.0;

return T;

}

9、电机模块

#include <REGX52.H>

#include "Timer1.h"

#include "Delay.h"

#include "LCD1602.h"

#include "MatrixKey.h"

//引脚定义,电机控制端口

sbit Motor=P1^0;

unsigned char Counter,Compare;

unsigned char SpeedDelta = 1; // 速度变化量，每次增加或减少的速度值

/\*\*

\* @brief 电机初始化

\* @param 无

\* @retval 无

\*/

void Motor\_Init(void)

{

Timer1\_Init(); // 定时器初始化函数，配置定时器以生成PWM信号

}

/\*\*

\* @brief 电机设置速度

\* @param Speed 要设置的速度，范围0~100

\* @retval 无

\*/

void Motor\_SetSpeed(unsigned char Speed)

{

Compare=Speed;

}

//定时器1中断函数

void Timer1\_Routine() interrupt 3

{

static unsigned int T1Count;

TL1 = 0x9C; //设置定时初值

TH1 = 0xFF; //设置定时初值 65436

Counter++; // 每次中断计数器加1

Counter%=100; //计数值变化范围限制在0~99

if(Counter<Compare) //计数值小于比较值,Compare是PWM占空比的值

{

Motor=1; //输出1

}

else //计数值大于比较值

{

Motor=0; //输出0

}

/\*

// 无级变速逻辑

//Compare += SpeedDelta; // 改变占空比

if (Compare >= 100 || Compare <= 0) // 如果达到速度极限，改变增加方向

{

//SpeedDelta = -SpeedDelta;

Compare = 30;

}

\*/

//按键扫描

T1Count++;

if(T1Count>=200)

{

T1Count=0;

MatrixKey\_Loop(); //20ms调用一次矩阵按键驱动函数

}

}

void ShowCompare() {

LCD\_ShowNum(1,10,Compare, 2);

}

void MotorAddSpeed() {

// 将占空比从0%逐渐增加到100%

unsigned char currentSpeed;

LCD\_ShowString(2,1,"Speed:");

for (currentSpeed = 0; currentSpeed < 100; currentSpeed++) {

Compare=currentSpeed;

// 在这里添加延时，以改变占空比变化的速率

LCD\_ShowNum(2,7,Compare, 2);

Delay(10); // 延时函数

}

}

void MotorReduceSpeed() {

// 从100%逐渐减少到0%

unsigned char currentSpeed;

LCD\_ShowString(2,1,"Speed:");

for (currentSpeed = 99; currentSpeed > 0; currentSpeed--) {

Compare=currentSpeed;

LCD\_ShowNum(2,7,Compare, 2);

Delay(10); // 延时函数

}

}

/\*放到定时器中断里

void MotorNaturalWind(unsigned char currentSpeed) {

Compare = currentSpeed;

Compare += SpeedDelta; // 改变占空比

if (Compare >= 99 || Compare <= 0) // 如果达到速度极限，改变增加方向

{

SpeedDelta = -SpeedDelta;

}

}

\*/

10、蜂鸣器模块

#include <REGX52.H>

#include <INTRINS.H>

//蜂鸣器端口：

sbit Buzzer=P2^5;

/\*\*

\* @brief 蜂鸣器私有延时函数，延时500us

\* @param 无

\* @retval 无

\*/

void Buzzer\_Delay500us() //@12.000MHz

{

unsigned char i;

\_nop\_();

i = 247;

while (--i);

}

/\*\*

\* @brief 蜂鸣器发声

\* @param ms 发声的时长，范围：0~32767

\* @retval 无

\*/

void Buzzer\_Time(unsigned int ms)

{

unsigned int i;

for(i=0;i<ms\*2;i++)

{

Buzzer=!Buzzer;

Buzzer\_Delay500us();

}

}