杭州科技职业技术学院

毕业设计

题　　目	基于漏磁法的直流电机测速装置
专　　业	机电一体化
班　　级	机电 1502
学　　号	2015112231
姓　　名	余蔚迪
指导教师	胡冬生

机电工程学院

2017 年 **11** 月 **25** 日

目录

基于漏磁法的直流电机测速装置

【摘要】在工程实践中,经常会遇到各种需要测量转速的场合，例如在发动机、电动机、卷扬机、机床主轴等旋转设备的试验、运转和控制中，常需要分时或连续测量和显示其转速及瞬时转速。要测速，首先要解决是采样问题。目前国内外测量电机转速的方法有很多，常用的转速测量方法有测速发电机测速法和光电码盘测速法等。比如，永磁直流测速发电机、无刷直流测速发电机、数字脉冲测速机、旋转编码器等，常用的就是直流电机轴端安装一个测速发电机。其优点是不需要电刷和换向器，结构简单、维护容易、惯量小、无滑动，缺点是存在剩余电压和相位误差且负载的大小会影响输出电压的幅值和相位，价格高。对于有刷直流电机测速来说，这些方法过于昂贵，不易安装。因此，本文设计了以单片机 STC12C5A60S2 为硬件的控制核心，以自制铁芯线圈为检测元件，并通过串口屏显示的直流电机的转速测量装置，解决了电机转轴不方便安装测速装置时的问题。

本文设计了一个直流电动机测速装置，该装置由漏磁式电机测速模块、单片机控制模块、电源模块、显示模块和语音模块组成。该装置的基本原理实现方式是电机转动使转子电流产生漏磁通，用自制铁芯电感检测电动机外壳电磁信号，经过检流芯片将电流值转化为电压值再经过放大器放大电压，通过 74HC04 构成施密特触发器将模拟信号转化为数字脉冲信号。由单片机处理算出其频率，从而求得电机转速并在 UsartGPU35D 串口屏上显示。经过测试，均能较好的测量电机转速的数值、精度等各项要求，还实现了曲线图形显示，转速记录、语音播报等创新功能。

【关键词】STC12C5A60S2 单片机 UsartGPU35D 串口屏 漏磁式电机测试速 74HC04 触发器

A dc motor measuring device based on the magnetic flux leakage method

【**Abstract**】 In engineering practice, various occasions that need to measure speed are often encountered. For example, when testing, running and controlling rotating equipments such as engines, motors, windlass, machine spindles, they often need to measure and display their rotational speed and instantaneous speed continuously or continuously. To measure speed, the first thing to solve is the sampling problem. There are many methods to measure the speed of motor at home and abroad. The common methods of measuring speed are speed measuring method of the speed measuring generator and the method of photoelectric encoder. For example, permanent magnet DC tachometer, brushless DC tachometer, digital pulse speed measuring machine, rotary encoder and so on are commonly used to install a speed generator on the shaft end of DC motor. The advantage is that it doesn't need brush and commutator, and has simple structure, easy maintenance, little inertia and no slip. The drawback is that there is residual voltage and phase error. The magnitude of load will affect the amplitude and phase of output voltage, and the price is high. For the speed measurement of brushed DC motor, these methods are too expensive and are not easy to install. Therefore, this paper designs a microcontroller STC12C5A60S2 as the core of the hardware control, a self-made core coil as the detection element, and a DC motor speed measuring device displayed by the serial port screen, which solves the problem that the motor shaft is not convenient to install the speed measuring device.

A DC motor speed measuring device is designed in this paper, which consists of a magnetic leakage motor speed module, a single-chip microcomputer control module, a power module, a display module and a voice module. The basic principle of the device is the way to achieve the rotation of the motor so that the rotor current leakage flux, using self-made iron inductance detecting motor shell electromagnetic signal, after detecting current chip will be current value into the voltage value through the amplifier voltage, composed by 74HC04 Schmidt trigger the analog signal into digital pulse signal. The frequency is calculated by a single chip microcomputer, and the speed of the motor is obtained and displayed on the UsartGPU35D serial screen. After testing, it can well measure all the requirements of motor speed, such as numerical value and accuracy, and also achieve the innovative functions of curve and graphics display, speed recording, voice broadcast and so on.

【**Keywords**】 STC12C5A60S2 microcontroller UsartGPU35D serial port flux leakage motor test speed 74HC04 trigger

第一章 绪论

1.1 研究背景

转速表作为机械行业必备的仪器之一 ，用来测定电机的转速 、线速度或频率 。在电机 、电扇、造纸、塑料、化纤、洗衣机、汽车、飞机、轮船等制造业中，转速表都得到了广泛的应用。转速表、转速测量在国民经济的各个领域，都是必不可少的。随着社会经济，科学技术不断发展，各个领域的机械设备对转速表要求也越来越高。而单片机具有体积小、成本低、功能强、智能化等优点 。将单片机嵌人到转速表内可以很大程度上改善转速表的稳定性、抗干扰能力、体积 、功能 、测量精度与范围等性能 。因此研究单片机多功能转速表的设计是非常有必要的。

随着现代社会的了发展，人们对转速表也提出了更高的要求，其中微处理技术和传感技术的巨大进步，使得转速表在这方面有革命性的进展。如今的转速表在功能、精度及自动化水平定方面发生了巨大变化，并相应的出现了各种各样的智能仪器控制系统，使得科学实验和应用工程的自动化程度有了显著提高。转速表产品的技术性能向高速率、高准确度、高稳定性、高可靠性方向发展等革命性功能，其以测量准确，测量速度快，易于实时测量和监控的巨大优点，逐渐取代了传统型的测速装置，从而成为测量领域的主流产品。

1.2 国内外研究现状

转速是能源设备与动力机械性能测试中的一个重要的特性参量，因为动力机械的许多特性参数是根据它们与转速的函数关系来确定的，例如压缩机的排气量、轴功率、内燃机的输出功率等等，而且动力机械的振动、管道气流脉动、各种工作零件的磨损状态等都与转速密切相关。转速测量的方法很多，测量仪表的型式也多种多样，其使用条件和测量精度也各不相同。根据转速测量的工作方式可分为两大类：接触式转速测量仪表与非接触式转速测量仪表。前者在使用时必须与被测转轴直接接触，如离心式转速表、磁性转速表与测速发电机等；后者在使用时不需要与被测转轴接触，如光电式转速表、电子数字式转速表、闪光测速仪等。测量发动机转速的传统方法是使用光电式转速表测量。用这种方法测量时，既要在发动机转动轴上粘贴光标纸，又要求测量人员把转速表与光标纸的距离控制在很近的范围，测量十分不方便。

随着科学技术的迅速发展，转速测量仪表已步入现代化、电子化的行列。过去曾经使用过的接触式测量仪表，如离心式转速表、磁性转速表、微型发电机转速表及钟表是定时转速表，均已先后受到冷落；而利用已知频率的闪光与被测轴转速同步的方法来测速的闪光测速仪，虽属非接触式仪表，目前仍有应用，但也退居次要地位。代之而起的是非接触式的电子与数字化的测速仪表。这类转速仪表大多具有体积小、重量轻、读数准确、使用方便等优点，容易实现电脑荧屏显示和打印输出，能够连续的反映转速变化，既能测定发动机稳定情况下的平均转速，也能够用来在足够小的时间间隔这一特定条件下测定发动机的瞬时转速。转速测量的应用系统在工业生产、科技教育、民用电器等各

领域的应用极为广泛，往往成为某一产品或控制系统的核心部分，其各种参数在不同的应用中有其侧重，但转速测量系统作为普遍的应用在国民经济发展中，有重要的意义。

1.3 本课题主要研究内容及方法

本文在分析直流电机特点的基础上，完成了一款结构简单、测量精度高的直流电动机测速装置设计，该装置由漏磁式电机测速模块、单片机控制模块、电源模块、显示模块和语音模块组成。其中，单片机为系统的控制核心，通过自制铁芯电感检测采集转子电流产生的漏磁通，得到微弱电流，经过检流芯片讲电流值转化为电压值在进过放大器放大电压，通过 74HC04 构成施密特触发器讲模拟信号转化为数字脉冲信号。并由单片机处理算出其频率，从而求得电机转速并在 UsartGPU35D 串口屏上显示。该装置具有测量精度高，安装方便，能很好的进行显示等特点。另外还实现了曲线图形显示，转速记录、语音播报等创新功能。

本设计主要完成以下工作:

1、基于 STC12C5A60S2 的测速装置设计方案。

2、漏磁式电机测速模块的硬件设计。

3、外部中断、串口屏与单片机的接口电路设计。

4、软件的编写与调试。

5、硬件 PCB 的制作与调试。

第二章 系统硬件设计

2.1 系统的整体功能概述

本系统主要是实现对直流电机的测速与显示。主要依据硬件与软件的相互协助工作来完成系统的功能。其中，硬件部分对传感器信号进行采集放大处理，并在串口屏中显示直流电机转速；软件部分支撑硬件，最采集脉冲进行处理及显示。本系统最终拟定以 STC12C5A60S2 作为核心控制芯片，并通过按键实现数据存储和语音播报等功能。系统有自制铁芯电感作为传感器进行电机转速检测，经过检流芯片将电流值转化为电压值再经过放大器放大电压，通过 74HC04 构成施密特触发器将模拟信号转化为数字脉冲信号，然后传送到单片机 STC12C5A60S2 芯片中进行数据处理。经过计算转换的数据再进一步传送到串口屏进行电机转速的显示。与此同时，将电机的历史转速通过曲线表格的形式发送到串口屏进行显示。在本系统中可以更直观的观察电机转速的变化。

系统框图如图 2-1 所示。

图 2-1　系统框架图

2.2 核心芯片选择

2.2.1 单片机

单片机是一个单芯片的微型计算机。单片机的优势在于实时控制能力强，并且可靠性高。本系统中采用的 STC12C5A60S2 单片机就是由 CPU 系统部分，存储器系统部分以及 I/O 口和其他功能单元部分组成的。STC12C5A60S2 是一种低功耗、高性能 CMOS 8 位微控制器。使用 Atmel 公司高密度非易失性存储器技术制造，指令代码完全兼容传统 8051, 但速度快 8-12 倍。片上 flash 允许程序存储器在线可编程，也适于常规编程器。在单芯片上，拥有灵巧的 8 位 CPU 和在系统上可编程闪烁存储单元，使得 STC12C5A60S2 为众多嵌入式控制应用系统提供灵活、有效的解决方案。

（1）STC12C5A60S2 的主要特性

1. 增强型 8051 CPU，1T，单时钟/机器周期，指令代码完全兼容传统 8051；

2. 工作电压：STC12C5A60S2 系列工作电压：5. 5V-3. 3V(5V 单片机)STC12LE5A60S2 系列工作电压：3. 6V-2. 2V(3V 单片机)；

3. 工作频率范围：0 － 35MHz，相当于普通 8051 的 0～420MHz；

4. 用户应用程序空间 8K /16K / 20K / 32K / 40K / 48K / 52K / 60K / 62K 字节；

5. 片上集成 1280 字节 RAM；

6. 通用 I/O 口(36/40/44 个)，复位后为：准双向口/弱上拉(普通 8051 传统 I/O 口)，可设置成四种模式：准双向口/弱上拉，推挽/强上拉，仅为输入/高阻，开漏，每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不要超过 120ma；

7. ISP(在系统可编程)/IAP(在应用可编程)，无需专用编程器，无需专用仿真器 可通过串口(P3. 0/P3. 1)直接下载用户程序，数秒即可完成一片；

8. 有 EEPROM 功能(STC12C5A62S2/AD/PWM 无内部 EEPROM)；

9. 内部集成 MAX810 专用复位电路(外部晶体 12M 以下时，复位脚可直接 1K 电阻到地)；

10. 外部掉电检测电路:在 P4. 6 口有一个低压门槛比较器，5V 单片机为 1. 32V，误差为+/-5%, 3. 3V 单片机为 1. 30V，误差为+/-3%；

11. 时钟源：外部高精度晶体/时钟，内部 R/C 振荡器(温漂为+/-5%到+/-10%以内) 1 用户在下载用户程序时,可选择是使用内部 R/C 振荡器还是外部晶体/时钟,常温下内部 R/C 振荡器频率为：5. 0V 单片机为：11MHz～15. 5MHz，3. 3V 单片机为：8MHz～12MHz，精度要求不高时，可选择使用内部时钟，但因为有制造误差和温漂，以实际测试为准；

12. 共 4 个 16 位定时器 两个与传统 8051 兼容的定时器/计数器, 16 位定时器 T0 和 T1，没有定时器 2，但有独立波特率发生器 做串行通讯的波特率发生器 再加上 2 路 PCA 模块可再实现 2 个 16 位定时器；

13. 2 个时钟输出口，可由 T0 的溢出在 P3. 4/T0 输出时钟，可由 T1 的溢出在 P3. 5/T1 输出时钟；

14. 外部中断 I/O 口 7 路, 传统的下降沿中断或低电平触发中断, 并新增支持上升沿中断的 PCA 模块，Power Down 模式可由外部中断唤醒
INT0/P3. 2, INT1/P3. 3, T0/P3. 4, T1/P3. 5, RxD/P3. 0, CCP0/P1. 3(也可通过寄存器设置到 P4. 2), CCP1/P1. 4 (也可通过寄存器设置到 P4. 3)；

15. PWM(2 路)/PCA(可编程计数器阵列, 2 路)：

——也可用来当 2 路 D/A 使用

——也可用来再实现 2 个定时器

——也可用来再实现 2 个外部中断(上升沿中断/下降沿中断均可分别或同时支持)；

16. A/D 转换，10 位精度 ADC，共 8 路，转换速度可达 250K/S(每秒钟 25 万次)，通用全双工异步串行口(UART)，由于 STC12 系列是高速的 8051，可再用定时器或 PCA 软件实现多串口；

17. STC12C5A60S2 系列有双串口，后缀有 S2 标志的才有双串口，RxD2/P1. 2(可通过寄存器设置到 P4. 2)，TxD2/P1. 3(可通过寄存器设置到 P4. 3)；

18. 工作温度范围：0 － 75℃(商业级)

19. 封装：PDIP-40, LQFP-44, LQFP-48 I/O 口不够时，可用 2 到 3 根普通 I/O 口线外接 74HC164/165/595(均可级联)来扩展 I/O 口，还可用 A/D 做按键扫描来节省 I/O 口，或用双 CPU，三线通信，还多了串口。

（2）STC12C5A60S2 引脚如图 2-2 所示。

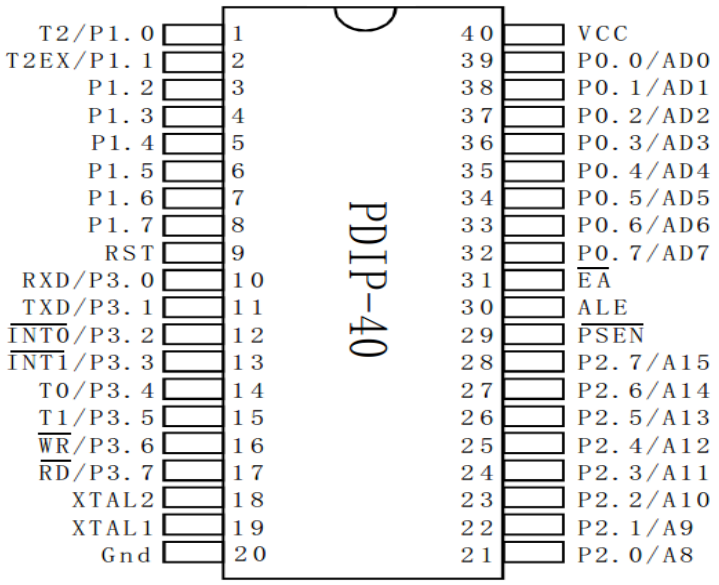


图 2-2　STC89C52RC 管脚图

①电源及时钟引脚

• VCC：电源接入引脚

• GND：接地引脚

• XTAL1：晶体振荡器接入引脚，在采用外部振荡器时，该引脚接地。

• XTAL2：与 XTAL2 同为晶体振荡器接入引脚，不同的是当采用外部振荡器时，该引脚作为外部振荡信号的接入引脚。

②控制线引脚

• RST：复位输入。当振荡器复位器件时，要保持 RST 脚两个机器周期的高平时间;

• ALE / PROG ：当访问外部存储器时，地址锁存允许的输出电平用于锁存地址的低位字节。在 FLASH 编程期间，此引脚用于输入编程脉冲。在平时，ALE 端以不变的频率周期输出正脉冲信号，此频率为振荡器频率的 1/6。因此它可用作对外部输出的脉冲或用于定时目的。然而要注意的是：每当用作外部数据存储器时，将跳过一个 ALE 脉冲。如想禁止 ALE 的输出可在 SFR8EH 地址上置 0。此时， ALE 只有在执行 MOVX，MOVC 指令时 ALE 才起作用。另外，该引脚被略微拉高。如果微处理器在外部执行状态 ALE 禁止，置位无效;

• PSEN：外部程序存储器的选通信号。在由外部程序存储器取址期间，每个机器周期 PSEN 两次有效。但在访问内部部数据存储器时，这两次有效的 PSEN 信号将不出现;

•EA/VPP：当 EA 保持低电平时，访问外部 ROM;注意加密方式 1 时，EA 将内部锁定为 RESET;当 EA 端保持高电平时，访问内部 ROM。在 FLASH 编程期间，此引脚也用于施加 12V 编程电源(VPP);

③并行 I/O 引脚

• P0 口：P0 口为一个 8 位漏级开路双向 I/O 口，每个管脚可吸收 8TTL 门电流。当 P0 口的管脚写“1”时，被定义为高阻输入。P0 能够用于外部程序数据存储器，它可以被定义为数据/地址的第八位。在 FLASH 编程时，P0 口作为原码输入口，当 FLASH 进行校验时，P0 输出原码，此时 P0 外部电位必须被拉高;

• P1 口：P1 口是一个内部提供上拉电阻的 8 位双向 I/O 口，P1 口缓冲器能接收输出 4TTL 门电流。P1 口管脚写入“1”后，电位被内部上拉为高，可用作输入，P1 口被外部下拉为低电平时，将输出电流，这是由于内部上拉的缘故。在 FLASH 编程和校验时，P1 口作为第八位地址接收;

• P2 口：P2 口为一个内部上拉电阻的 8 位双向 I/O 口，P2 口缓冲器可接收，输出 4 个 TTL 门电流，当 P2 口被写“1”时，其管脚电位被内部上拉电阻拉高，且作为输入。作为输入时，P2 口的管脚电位被外部拉低，将输出电流，这是由于内部上拉的缘故。P2 口当用于外部程序存储器或 16 位地址外部数据存储器进行存取时，P2 口输出地址的高八位。在给出地址“1”时，它利用内部上拉的优势，当对外部八位地址数据存储器进行读写时，P2 口输出其特殊功能寄存器的内容。P2 口在 FLASH 编程和校验时接收高八位地址信号和控制信号;

• P3 口：P3 口管脚是 8 个带内部上拉电阻的双向 I/O 口，可接收输出 4 个 TTL 门电流。当 P3 口写入“1”后，它们被内部上拉为高电平，并用作输入。作为输入时，由于外部下拉为低电平，P3 口将输出电流(ILL)，也是由于上拉的缘故。P3 口也可作为 AT89C51 的一些特殊功能口：

• P3. 0 RXD(串行输入口)

• P3. 1 TXD(串行输出口)

• P3. 2 INT0(外部中断 0)

• P3. 3 INT1(外部中断 1)

• P3. 4 T0(记时器 0 外部输入)

• P3. 5 T1(记时器 1 外部输入)

• P3. 6 WR（外部数据存储器写选通）

• P3. 7 RD（外部数据存储器读选通）

2.2.2 INA199A1 检流芯片

由自制铁芯电感检测产生一定规律的电流信号,将铁芯电感两端接入 INA199A1 芯片的 IN+, IN−，因为电机转动切割磁感线，由磁生电，电感两端产生电压，再放大 10 倍输出电压差。由于此时 INA199A1 只有半个电信号，VCC 的电频必须变为 1/2。当电频变为 1/2 时，放大倍数应比之前的大上两倍，输出信号经过带通滤波送到运算放大器，再由运算放大器放大 10 倍，获得一个有规律的信号，再经由 74HC04 构成的施密特触发器获得的数字脉冲信号，发送至单片机，计算得转速值。

2.2.3 74HC04 施密特触发器

74HC/HCT04 是高速的硅栅 CMOS 器件并兼容低功耗肖特基的 TTL（ LSTTL ）非门（逆变器）。74HC04 是内含 6 组相同的反相器。即 1A 输入高电平，1Y 输出低电平六反相器。74HC04 逻辑图如图 2-3 所示。

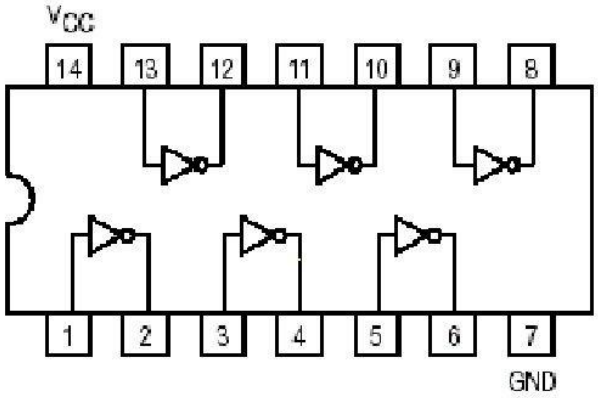


图 2-3 74HC04 逻辑图

根据设计要求，电机产生的漏磁通因外界干扰的原因，所产生的电压值不稳定。所以采用 74HC04 构成的施密特触发器。施密特触发器也有两个稳定状态，但与一般触发器不同的是，施密特触发器采用电位触发方式，其状态由输入信号电位维持；对于负向递减和正向递增两种不同变化方向的输入信号，施密特触发器有不同的阈值电压。施密特触发电路如图 2-4 所示。

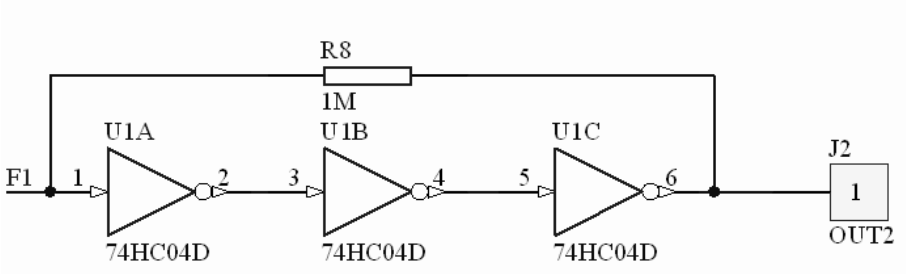


图 2-4 施密特触发电路

施密特触发器基本介绍：

门电路有一个阈值电压，当输入电压从低电平上升到阈值电压或从高电平下降到阈值电压时电路的状态将发生变化。施密特触发器是一种特殊的门电路，与普通的门电路不同，施密特触发器有两个阈值电压，分别称为正向阈值电压和负向阈值电压。在输入信号从低电平上升到高电平的过程中使电路状态发生变化的输入电压称为正向阈值电压，在输入信号从高电平下降到低电平的过程中使电路状态发生变化的输入电压称为负向阈值电压。正向阈值电压与负向阈值电压之差称为回差电压。

它是一种阈值开关电路，具有突变输入——输出特性的门电路。这种电路被设计成阻止输入电压出现微小变化（低于某一阈值）而引起的输出电压的改变。

利用施密特触发器状态转换过程中的正反馈作用，可以把边沿变化缓慢的周期性信号变换为边沿很陡的矩形脉冲信号。输入的信号只要幅度大于 vt+，即可在施密特触发器的输出端得到同等频率的矩形脉冲信号。

当输入电压由低向高增加，到达 V+时，输出电压发生突变，而输入电压 Vi 由高变低，到达 V−，输出电压发生突变，因而出现输出电压变化滞后的现象，可以看出对于要求一定延迟启动的电路，它是特别适用的.

从传感器得到的矩形脉冲经传输后往往发生波形畸变。当传输线上的电容较大时，波形的上升沿将明显变缓；当传输线较长，而且接受端的阻抗与传输线的阻抗不匹配时，在波形的上升沿和下降沿将产生振荡现象；当其他脉冲信号通过导线间的分布电容或公共电源线叠加到矩形脉冲信号时，信号上将出现附加的噪声。无论出现上述的那一种情况，都可以通过用施密特反相触发器整形而得到比较理想的矩形脉冲波形。只要施密特触发器的 vt+和 vt−设置得合适，均能收到满意的整形效果。根据上图 2−4 施密特触发电路调整电阻大小，控制阈值得到施密特波形图如图 2−4 所示。

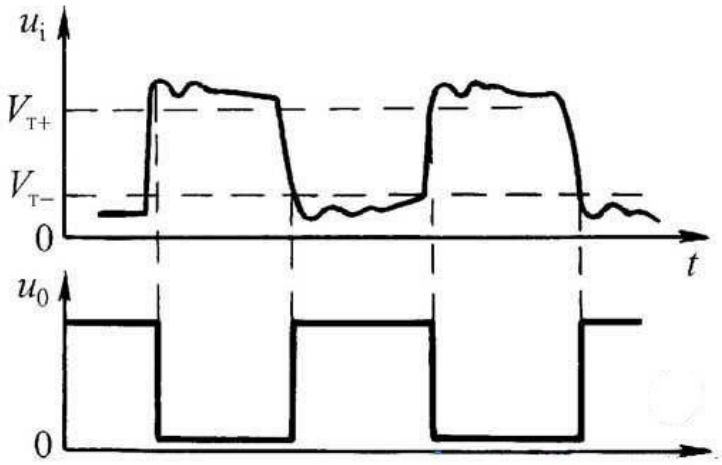


图 2−5 施密特波形图

2.2.4 串口屏显示

串口屏就是，带串口控制的液晶屏。广泛应用于工业自动化、电力、电信、环保、医疗、金融、石油、化工、交通、能源、地质、冶金、公共查询与监控等数十个行业和领域，其中，在某些军工及航天领域，凭借可靠、稳定的产品性能已被列入合格供应商名录。串口屏相比较普通的 LCD1602 串口屏接口简单，色彩显示丰富，字体大小可调整，可显示曲线图形等。串口屏实物图如图 2−6 所示。

图 2-6 串口屏实物图

2. 3 各模块电路

2.3.1 漏磁式电机测速电路

漏磁式电机测速模块电路图，如下图 2-7 所示。漏磁式电机测速是从输入交流信号，经过 INA199A1 芯片，由于此时 INA199A1 只有半个电信号，VCC 的电频必须变为 1/2。当电频变为 1/2 时，放大倍数应比之前的大上两倍，输出信号经过带通滤波送到运算放大器，再由运算放大器放大 100 倍，获得一个有规律的信号，再经由 74HC04 构成的施密特触发器获得的数字脉冲信号，发送至单片机，计算得转速值。INA199A1 和 74HC04 原理图如图 2-7 所示。

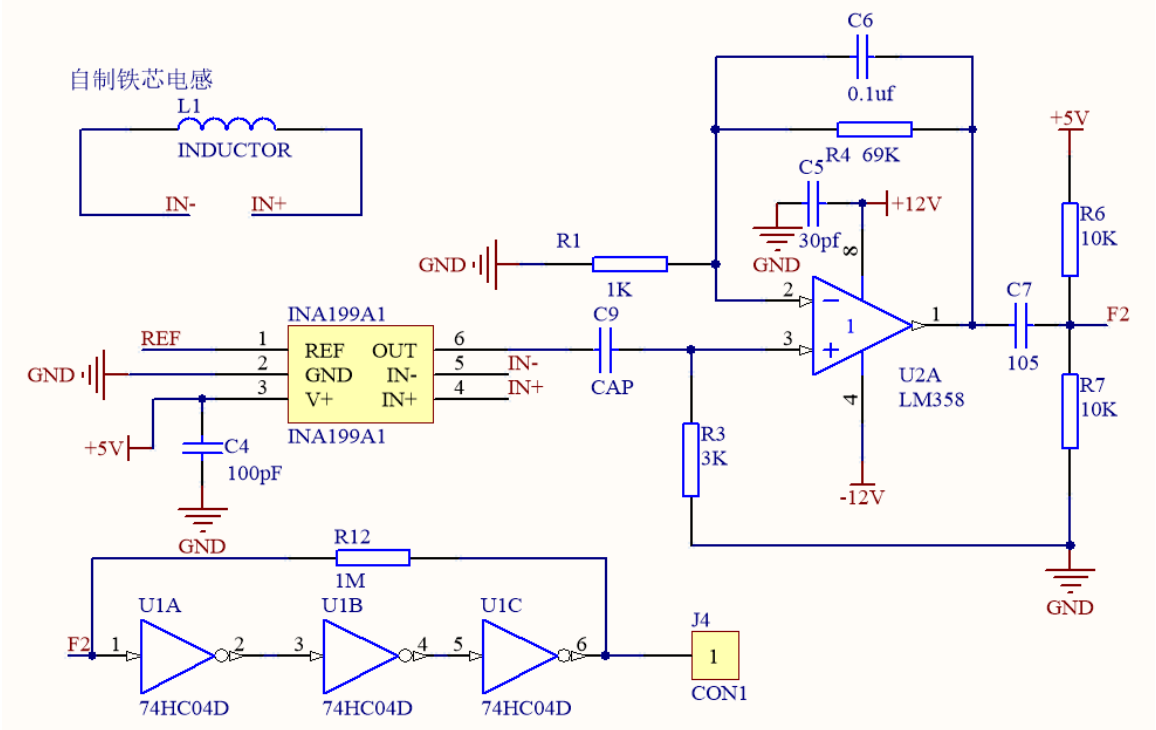


图 2-7 INA199A1 和 74HC04 原理图

2. 3. 2 按键电路

按键电路采用了 3 个按键，包括测速储存键、历史速度显示键、语音播报键，分别连接单片机的 P20、P21、P22 如图 2-8 所示。

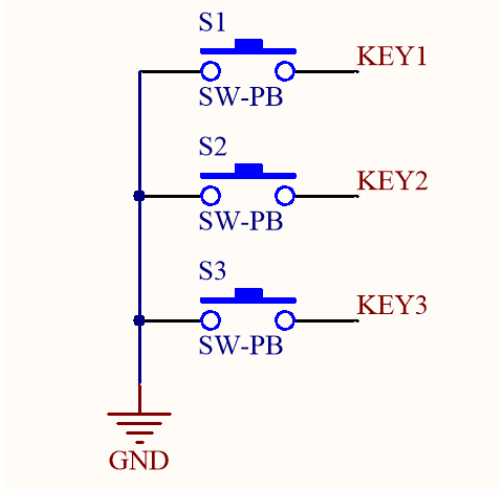


图 2-8　按键电路

2.3.3 串口屏显示电路

在本设计中 TXD，RXD 接在 STC12C5A60S2 单片机的 P30，P31 口，因为这两个 IO 口是单片机的串口一。因为串口屏背光的原因，需要的电流为 200mA，而单片机电流在 100mA 所以串口屏需要单独供电，GND 与单片机共地。液晶显示电路如图 2-9 所示

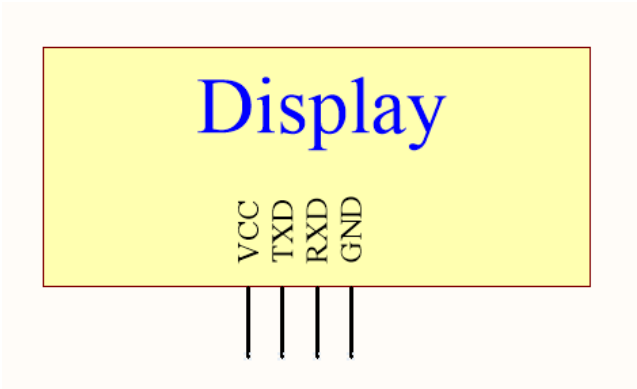


图 2-9 串口屏示电路

串口屏 GPU35B 是为+5V 的电压 200mA 的电流驱动，分辨率为 480X320，就是表明横向有 480 个点，纵向有 320 个点，也就是说显示 16 点阵的汉字横向可以显示 30 个汉字，纵向可以显示 15 行。确定了分辨率，我们就可以根据分辨率制定了一套坐标体系，左上角的坐标是（0, 0）；向右 X 方向增加，最大 319；向下 y 方向增加，最大 239；有了坐标，我们就可以在屏幕指定位置，做显示汉字，显示图片等操作。本串口屏的所有指令都是和坐标有关的，靠坐标确定相应的位置。串口屏

支持 64K 真彩，所谓的 64K，实际是 65536 种颜色；我们知道，颜色是由 RGB（红，绿，蓝）三基色组成，每种颜色的不同灰度叠加到一起，组成所有五颜六色的颜色。65536 在数学书，是 16 位二进制的表现范围，因此 RGB 在其中，瓜分了 16 位，呈现了 RRRRRGGG GGGBBBBB ，这样的 16 位二进制，就是说，红色 R 占据前 5 位，绿色 G 占据中间 6 位；蓝色占据后 5 位，这就是俗称 RGB565 模式的颜色值；在这种颜色状态线，红色可以表示 32 级灰度，绿色 64 级，蓝色 32 级；可以组合 32*64*32=65536 种颜色。

在串口屏中有一个用于记录历史数据曲线的控件，比如：像这种温度曲线，PM2. 5 曲线，电压电流曲线之类就特别适合曲线控件来使用；制作的时候，我们假设 AD 采样值是 0~1023；如果是 1023 对应曲线的最高处，可以采用公式：255-AD 值/4 得到曲线值，用 Sn 或 Qn 送入串口屏既可以显示出来。

在 DEMO 系统中都有相应的示例，即：先用： DQX（20, 30, 12, 12, 20, 16）初始化一个，240*192 的曲线，并通过：S128；S136；S145；S153；S162；S170；S177；S185；S192；S198；S204；. . . . . 等一系列数值画出曲线，实际单片机可以完全根据 AD 值送入返些数据，即可完成曲线显示。

GPU35B 串口屏的指令列表（部分）如表 1 所示。

表 1 GPU35B 串口屏的指令列表（部分）

类别	指令	说明	示例
基本绘图指令	CLS(c)	用 c 颜色清屏	CLS(0);
	CIRF(x1,y1,r,c);	在(x1,y1)处用颜色 c 画一个半径 r 的实心园	CTRF(100,100,50,1);
	CBOX(x1,y1,x2,y2,r,c);	用颜色 c 画一个圆角方框，左上角(x1,y1),右下角(x2,y2)；圆角半径 r;	CBOX(10,10,100,100,5,1);
	CBOF(x1,y1,x2,y2,r,c);	用颜色 c 画一个实心圆角方框 ,左上角(x1,y1),右下角(x2,y2)；圆角半径 r;	CBOF(10,10,100,100,5,1);
	W8BF(x1,y1,x2,y2,cn)	Win8 风格的方框	
汉字显示指令	DS12(x1,y1,'显示内容字符串',c，limitX);	在(x,y)处用颜色 c 显示一行 12 点阵字；limitX 可选，本参数 V3.0 才支持，下同；	DS12(0,0,'显示字符串',1);
	DS16(x1,y1,'显示内容字符串',c，limitX);	在(x,y)处用颜色 c 显示一行 16 点阵字	DS16(0,0,'显示字符串',1);
	DS24(x1,y1,'显示内容字符串',c，limitX);	在(x,y)处用颜色 c 显示一行 24 点阵字	DS24(0,0,'显示字符串',1);
	DS32(x1,y1,'显示内容字符串',c，limitX);	在(x,y)处用颜色 c 显示一行 32 点阵字	DS32(0,0,'显示字符串',1);
	DS48(x1,y1,'显示内容字符串',c，limitX);	在(x,y)处用颜色 c 显示一行 48 点阵字	DS48(0,0,'显示字符串',1);

	DS64(x1,y1,'显示内容字符串',c，limitX);	在(x,y)处用颜色 c 显示一行 64 点阵字	DS64(0,0,'显示字符串',1);
	BS12(x1,y1,x2,lw,'显示内容',c);	在(x1,y1)处，显示 12 点阵字符串，在 x2 处自动折行，行间距 lw,颜色 c;	BS12(0,0,219,4,'显示内容...很多字',c);
	BS16(x1,y1,x2,lw,'显示内容',c);	在(x1,y1)处，显示 16 点阵字符串，在 x2 处自动折行，行间距 lw,颜色 c;	BS16(0,0,219,4,'显示内容...很多字',c);
曲线相关指令	DQX(x,y,xp,yn,xn,yp)	设置并在(x,y)为左上角的地方画一个曲线，曲线横向有 xn 个格子，纵向有 yn 个格子；横向每个格子有 xp 点；纵向每个格子有 yp 点；(*)注：曲线格子总长和总宽不得超过 255；	DQX(0,0,10,10,10,10);
	RQX	刷新曲线(*)	RQX;
	Qn	送入一个数据点，为 0~255 的数值，128 表示曲线中间点；(*)	Q128;Q129;Q10;Q0;
	Sn	同 Qn,但是每送 1 点，曲线就会刷新，用于连续采样显示送的数据(*)	S128;S129;

2. 3. 4 单片机的晶振电路

单片机的时钟电路有内部自震荡电路和外部时钟电路两种。内部的自振荡电路是根据 STC12C5A60S2 单片机内部反相放大器与石英晶体和两个电容构成的，XTAL1 和 XTAL2 分别作为放大器的输入端与输出端，电容起到了快速起振和稳定频率的作用。STC12C5A60S2 系统中晶振频率一般在 1. 2~12MHz 选择。外接电容 C2、C3 的大小会影响振荡器频率的高低、振荡频率的稳定度、起振时间及温度稳定性。在设计电路板时，晶振和电容应靠近单片机，以便减少寄生电容，保证振荡器稳定可靠工作。在本设计中，采用外接石英晶体的方法，所用晶振频率为 12MHz，所选择电容为 30pF。如图 2-10 所示。

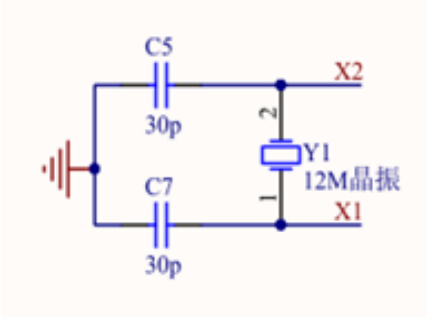


图 2-10　晶振电路

2.4 电源电路

本设计一个二角插头进行供电，电源经过变压器，全桥整流，稳压后输出。再经过 LM7812、LM7805、LM7905 得到 12V、5V、−5V。图 2−11 为电源电路。

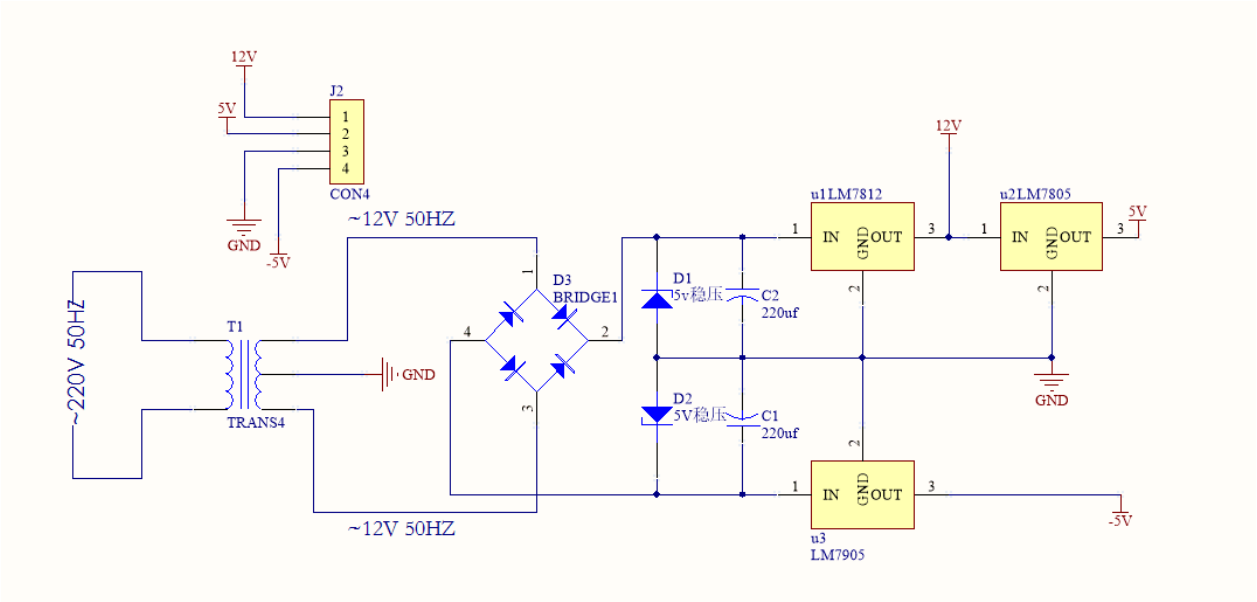


图 2−11 电源电路

2.5 语音播报电路

当测得转速时，按下语音播报键，会播报实时数据。XFS5152CE 是一款高集成度的语音合成芯片，可实现中文、英文语音合成；并集成了语音编码、解码功能，可支持用户进行录音和播放；除此之外，还创新性地集成了轻量级的语音识别功能，支持 30 个命令词的识别，并且支持用户的命令词定制需求。支持 UART、I2C、SPI 三种通讯方式，本设计采用的是串口通讯。对语音模块发送字符串，语音模块就会发出合成人声播报。串口屏的 TXD，RXD 和单片机的 RXD，TXD 相连。图 2−12 是语音模块电路。

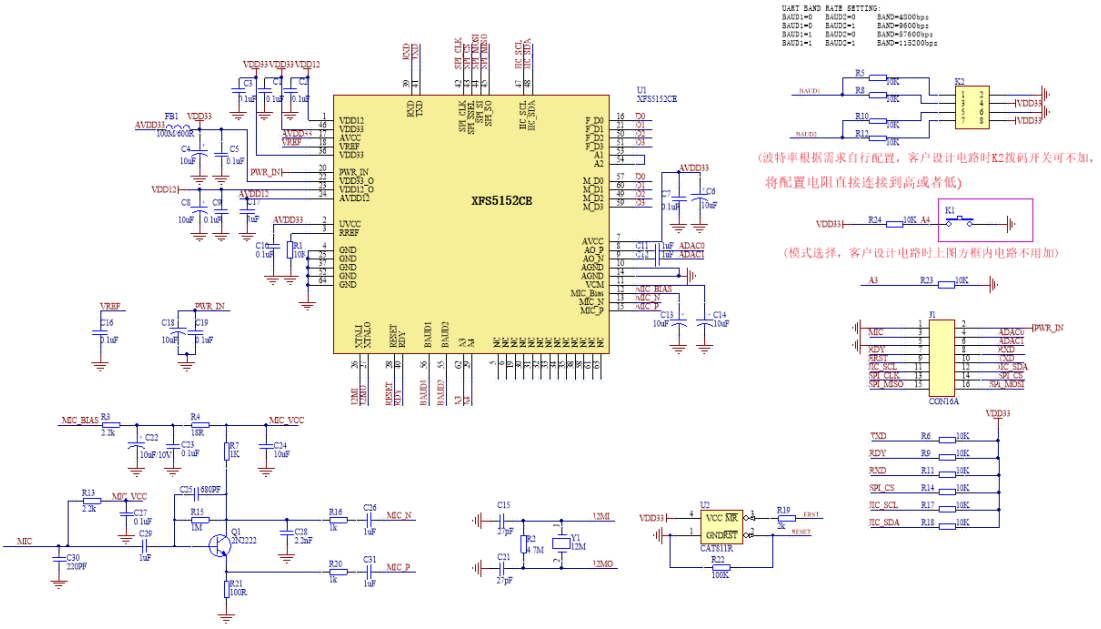
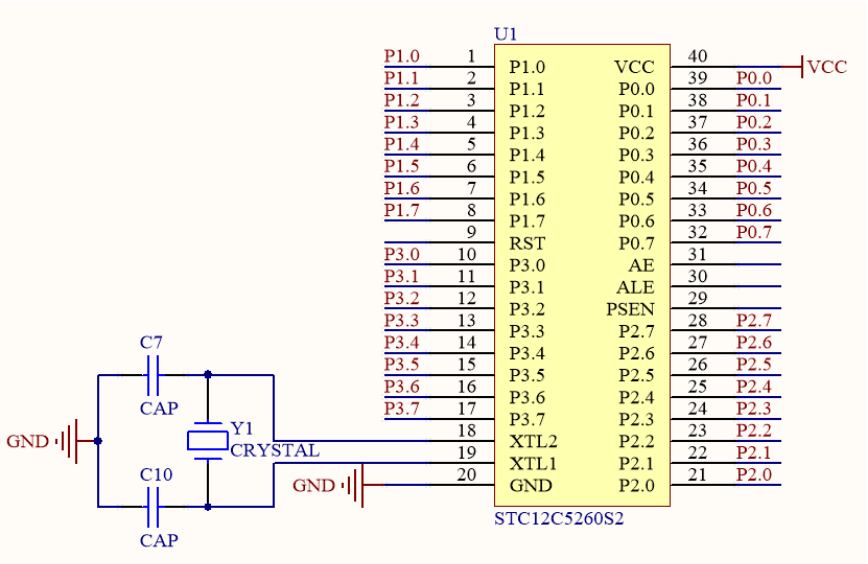


图 2-12 语音模块电路

2.6 最小单片机电路

将单片机所有引脚单独引到排针，方便自己设计开发。



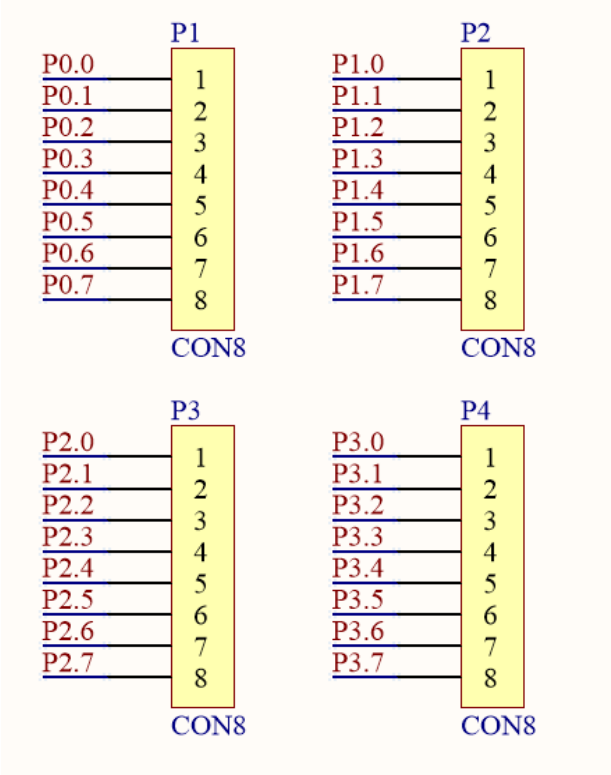


图 2-13 最小单片机电路图

2.7 原理图

先画完子模块的原理图，再把各模块的网路线连接，整理。

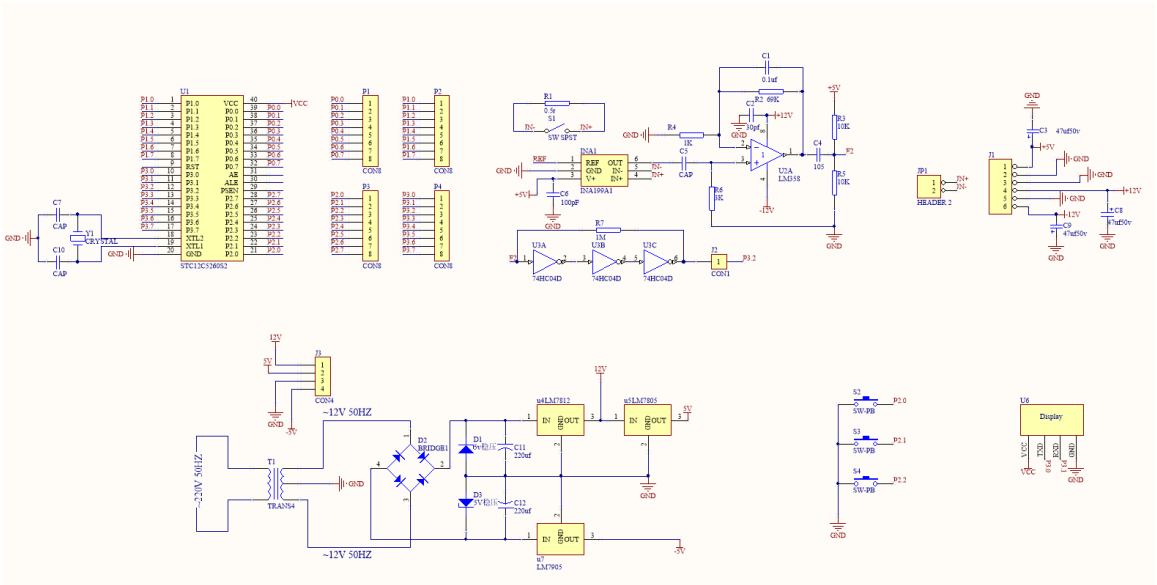


图 2-14 原理图

第三章 系统软件设计

在单片机应用系统的开发中，软件的设计是最复杂和困难的，大部分情况下工作量都较大，特别是对那些控制系统比较复杂的情况，往往需要同时考虑单片机的软硬件资源分配。本系统的软件设计主要分为系统初始化、按键、显示处理及信号频率输入处理。

程序的设计是一件复杂的工作，为了把复杂的工作简单化，就要有相应的步骤和方法。其步骤可概括为以下三点：

⑴ 分析系统控制要求，确定算法：对复杂的问题进行具体的分析，找出合理的计算方法及适当的数据结构，从而确定编写程序的步骤。这是能否编制出高质量程序的关键。

⑵ 根据算法画流程图：画程序框图可以把算法和解题步骤逐步具体化，以减少出错的可能性。

⑶编写程序：根据程序框图所表示的算法和步骤，选用适当的指令排列起来，构成一个有机的整体，即程序。

程序数据的一种理想方法是结构化程序设计方法。结构化程序设计是对利用到的控制结构类程序做适当的限制，特别是限制转向语句(或指令)的使用，从而控制了程序的复杂性，力求程序的上、下文顺序与执行流程保持一致性，使程序易读易理解，减少逻辑错误和易于修改、调试。根据系统的控制任务，本系统的软件设计主要由主程序、初始化程序、显示子程序、数据采集子程序和延时程序等组成。

3.1 软件功能概述

在完成了对系统硬件的设计后，就要求有一个功能完善并且稳定的软件来指导协调硬件的内部操作，对硬件起到指导作用，使硬件系统的作用发挥到最大化，同时还要考虑到未来硬件设计的更新升级等。系统软件有监控和执行的功能。在本设计中，系统软件设计主要分为主程序设计、延时程序设计、外部中断程序设计和显示程序设计等。主程序完成各器件的初始化，并且调用各个相应的子程序模块设计；延时程序使得单片机在处理显示程序时有一定间隔；显示程序即为把采集程序模块存储的数据送到串口屏进行显示。

3.2 主程序设计

主程序流程图如下，主程序是以检测转速为主要目的，通过按键控制题目的切换，按键一是测量转速部分，通过外部中断累加和定时器定时 1s，求出转速。按键二是控制语音部分，触发语音播报。按键三是信息储存，记录当时的转速。按键四是显示信息，即显示按键三所记录的转速。

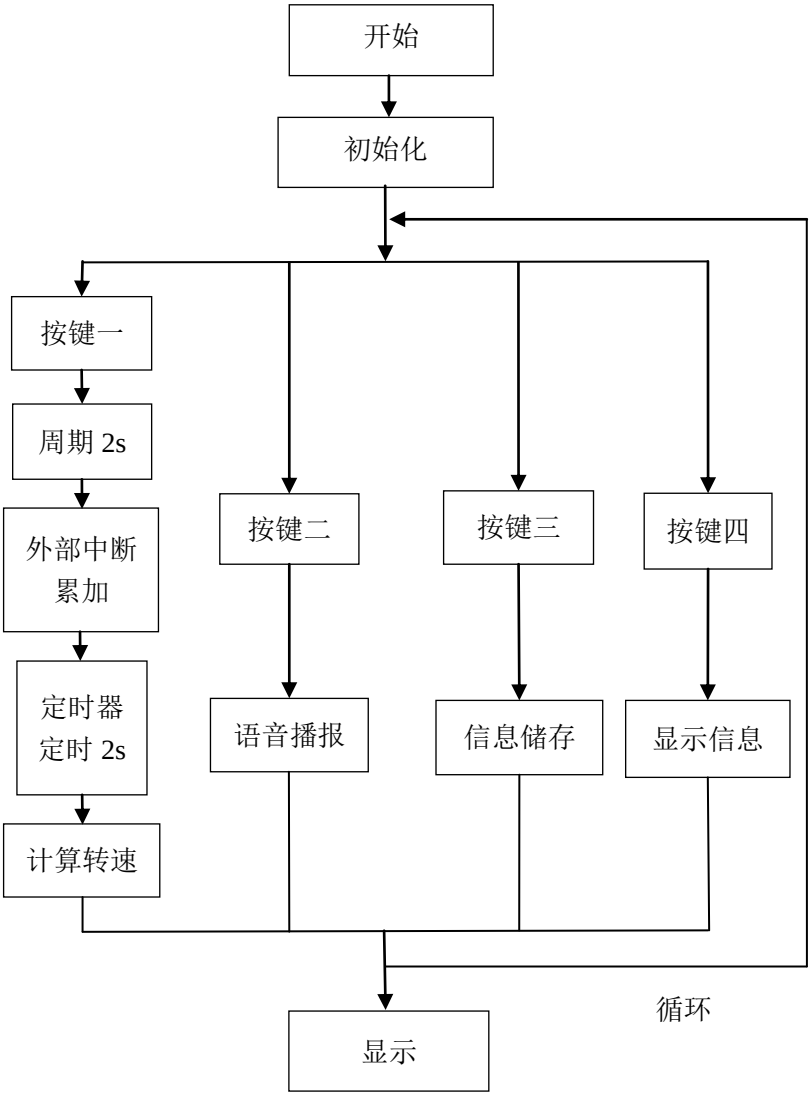


图 3-1 主程序流程图

3.3 子程序设计

3. 3. 1 外部中断数据采集程序

12 单片机的外部中断源有两个，分别为 INT0 和 INT1，分别由引脚 INT0（P3. 2）和 INT1（P3. 3）输入，中断源请求方式有两种：低电平触发和下降沿触发。外部中断触发方式选择位：ITx。ITx=0 时，外部中断 x 触发方式为低电平触发，在这种方式下，CPU 在每个机器周期检测外部中断请求输入引脚电平，若检测为低电平，则硬件将置位相应中断标志位 IEx，响应请求后通过硬件清零 IEx。ITx=1 时，外部中断 x 触发方式为下降沿触发，在这种方式。下，CPU 在连续的两个机器周期先后检测到先高电平后低电平，则将置位相应中断标志位 IEx，响应请求后通过硬件清零 IEx。其流程图如图 3-2 所示。

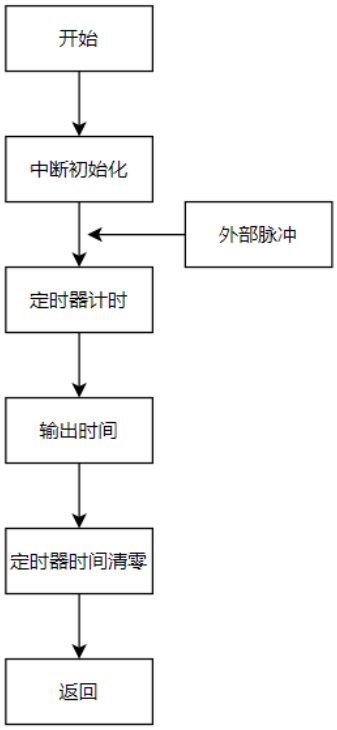


图 3-2 外部中断采集流程图

程序方面，需要对定时器及中断寄存器就行初始化，过程如下：

1. 设置 TMOD，以确定 Tx 的工作方式；
2. 计算初值，并将初值写入 TH 和 TL；
3. 允许中断，如果使用中断方式，需要对寄存器 IE 赋 1；
4. 置位 TRx，启动定时器。

初始化完成后编写定时器服务函数，把一些需要进行精准计时的变量写入服务函数。

3. 3. 2 按键程序

键盘电路采用了 4 个按键，包括按键一是测量转速部分，通过外部中断累加和定时器定时 1s，求出转速。按键二是控制语音部分，触发语音播报。按键三是信息储存，记录当时的转速。按键四是显示信息，即显示按键三所记录的转速。

设计流程图如图 3-4 所示。

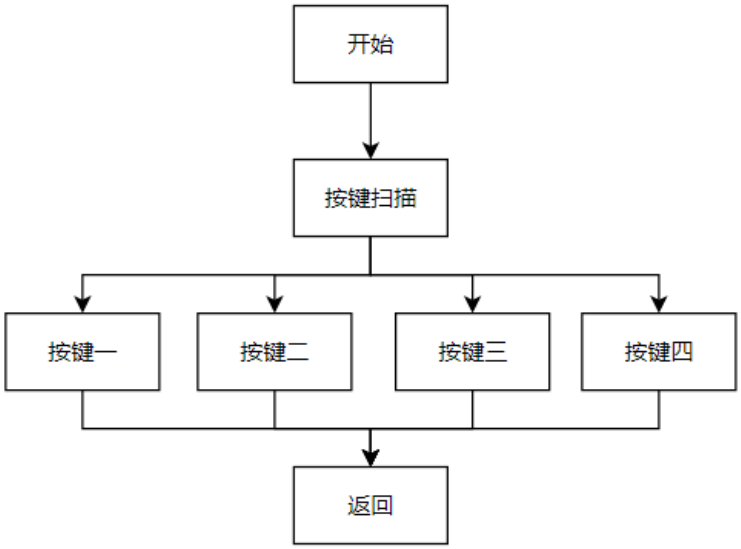


图 3-4 按键程序流程图

程序方面，实际过程中按键的弹片不是一接触就是紧紧的闭合，它还存在一定的抖动，尽管这个时间非常短暂，但是对于我们执行时间以 us 为计算的单位来说，它太漫长了。因此要对按键程序进行软件消抖处理，当按键按下后由定时器产生一个 20ms 的时间，如果 20ms 后按键仍然是按下的，那么就确定这次按键被按下了，将按下的按键的值传给 key_num 变量，以方便主程序的调用。

3.3.3 串口屏显示程序

串口屏显示程序主要是来显示的数据内容，按键功能，图表绘制等，也是十分重要的程序之一。设计流程图如图 3-3 所示。

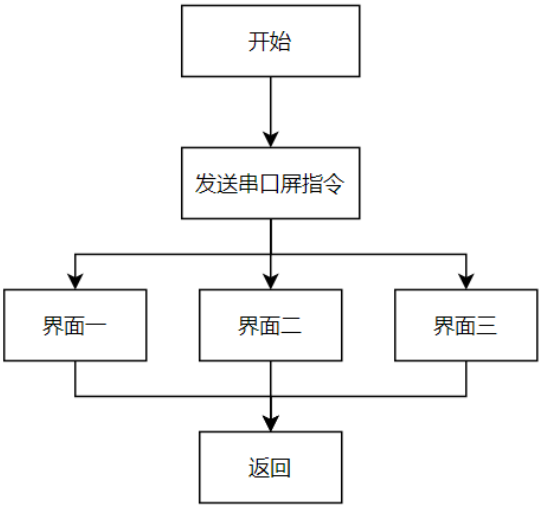


图 3-3 LCD 显示程序流程图

程序方面，首先要初始化串口，对串口通讯进行设置，步骤如下：

1. 配置 SCON：设定串口方式（ SM0 及 SM1 位配置 4 种方式）；串口是否要接收数据（REN 位）

2. 设定 PCON 寄存器的 SMOD 位（若非串口方式 0）

3. 若串口为方式 1 和方式 3，则需要配置 TMOD:选择定时器 1 并配置其初值装载方式，并设置 TH1,TL0 的初值以确定通信的波特率(由波特率的计算公式),然后配置 TCON:开启定时器 1(TR1)。

4. 允许使用中断(ES)，开启 UART 中断(EA)

配置完成之后将做需要显示的汉字和变量通过串口的形式传到串口上，例如：

sprintf(buf_display, ″DS48(80, 100, ' 转速：%3. 1f　rpm　', 1, 0); \r\n″, PL);

PrintString(buf_display);

把″DS48(80, 100, ' 转速:%3. 1f　rpm　', 1, 0); \r\n″这一串字符保存至 buf_display 这个数组中，再把 buf_display 数组通过 PrintString 函数发送至串口屏。至此完成一条串口屏的显示，再将其他的 buf_display 逐次往串口屏进行发送，就完成了人机界面。

第四章 制作与调试

4. 1 硬件电路的制作

4. 1. 1 PCB 板的设计

本次设计使用的是 Altium Designer（Altium Designer 是原 Protel 软件开发商 Altium 公司推出的一体化的电子产品开发系统，主要运行在 Windows 操作系统。这套软件通过把原理图设计、电路仿真、PCB 绘制编辑、拓扑逻辑自动布线、信号完整性分析和设计输出等技术的完美融合，为设计者提供了全新的设计解决方案，使设计者可以轻松进行设计，熟练使用这一软件必将使电路设计的质量和效率大大提高。），来绘制 PCB。使用该软件极大的方便了电路的布线，使硬件电路制作更加方便，性能更加稳定。布线效果如图 4-1 所示。

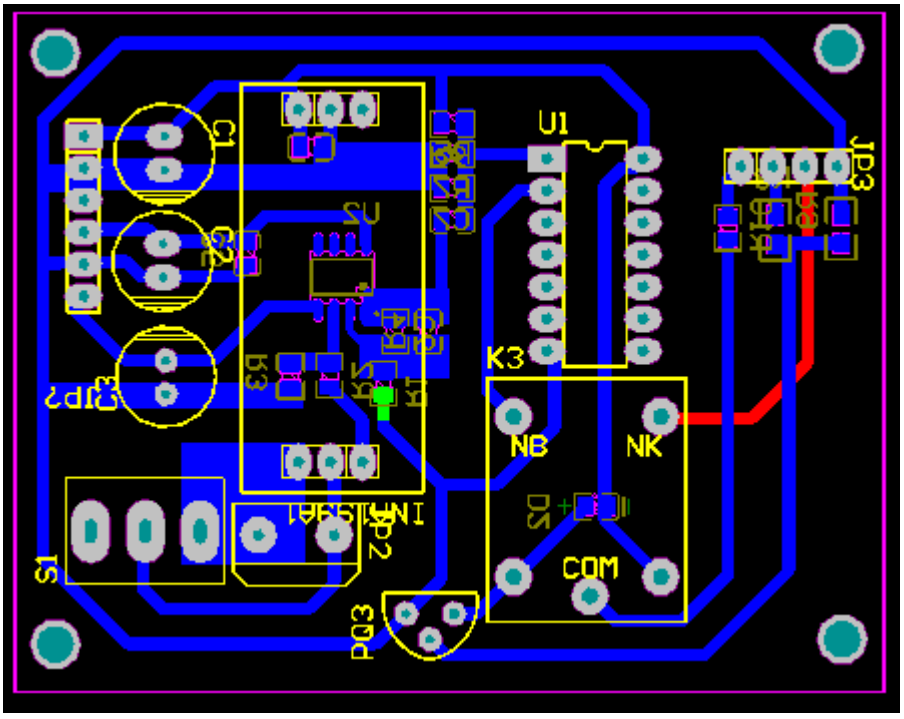


图 4-1 PCB 原理图

4. 1. 2 PCB 的制作

（1）生成布线报告，检查布线合理。

（2）将 PCB 图打印到油性黄纸上。

（3）用砂纸将覆铜板上面的氧化铜打磨干净。

（4）贴合好后的铜板放在制版机内高温来回压，将转印纸上面的图案转移到铜板上。

（5）将铜板上面不够黑的地方用马克笔涂黑，保证所有不该漏铜的地方都不漏铜。

（6）放到硫酸铜腐蚀液中并加热形成一个完成的电路板。

（7）完成后用清水冲洗一下。洗净擦干后用砂纸打磨掉碳粉。

（8）按照原理图进行钻孔，孔的大小不得超过画的板子上孔的大小。

（9）检查形成效果，焊接元器件。

4.1.3 元器件的焊接

焊接前应熟悉各芯片的引脚，焊接时参照电路图，仔细地连接引脚。按照以下原则进行焊接：

（1）先焊接各芯片的电源线和地线，这样确保各芯片有正确的工作电压。

（2）同类的芯片应顺序焊接，在一片焊接并检查好之后，其他的同类芯片便可以参照第一片进行焊接。这样便可大大节省时间，也可降低出错率。

4. 2 电路的调试

至此，电路已经焊好，剩下的工作就是进行系统调试。PCB 实物（反面）如图 4-2 所示。

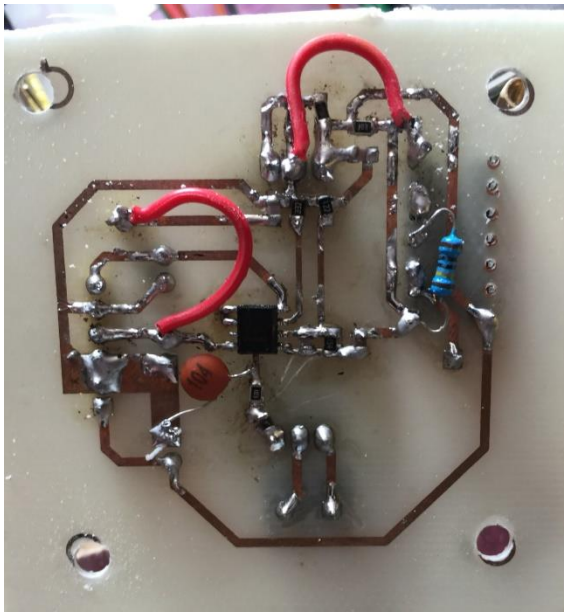


图 4-2 PCB 实物（反面）

4. 2. 1 硬件调试

系统硬件调试比较简单，按照设计电路接线图检查安装电路，在安装好的电路中按电路图一一对照检查连线，在检查中要对已经检查过的连线做标记，使用万用表对检查连线很有帮助。

直观检查电源，地线，信号线，元器件接线端之间有无短路，联线处有无接触不良，有极性元器件引线短有无接错，反接等，集成块是否插对。通电观察。把经过准确测量的电源电压加入电路，但暂不接入信号源信号。电源接通后，首先观察有无异常现象，包括有无冒烟，异常气味。触摸元

件是否发烫，电源是否短路等。如果出现异常，应立即切断电源，排除故障后，才可重新通电。至此硬件调试完成。PCB 实物（正面）如图 4-3 所示。整体实物图如图 4-4。

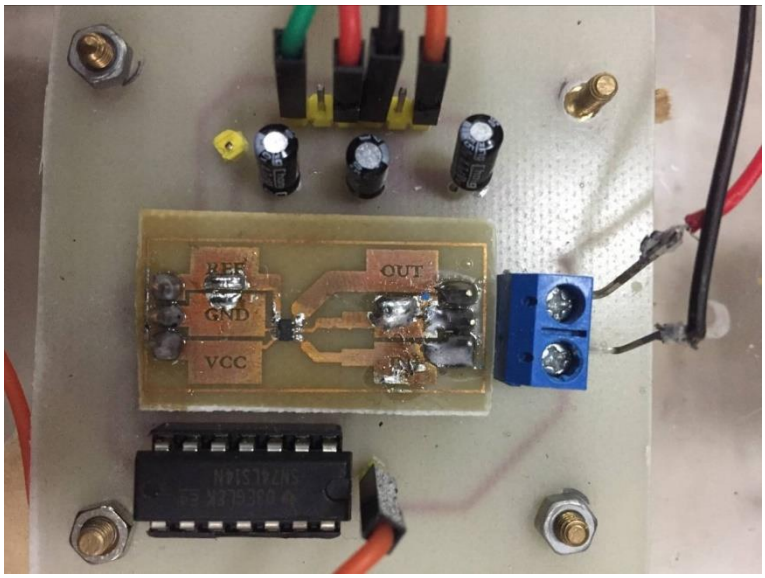


图 4-3 PCB 实物（正面）

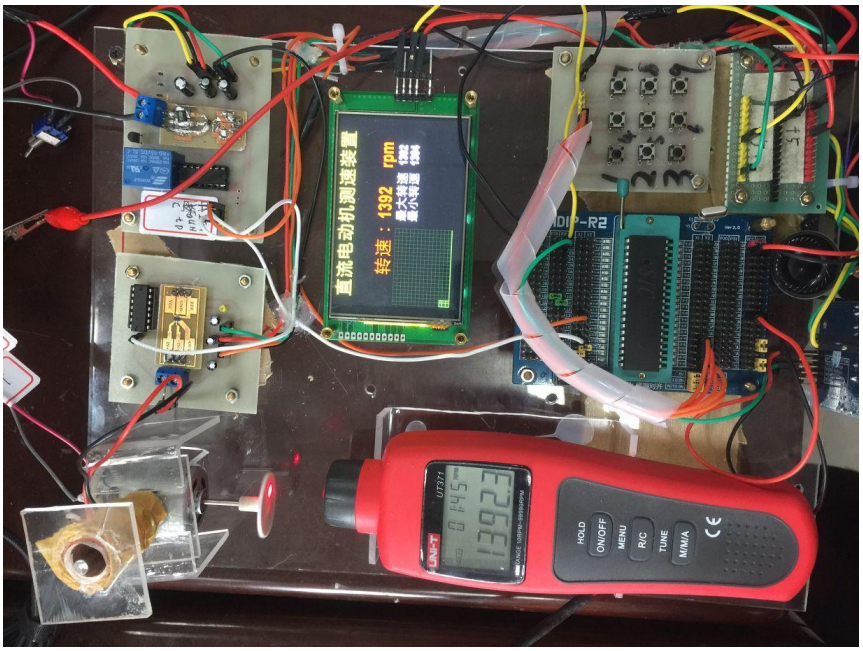


图 4-4 整体实物图

4. 2. 2 软件调试

将电路程序通过仿真软件 Keil 下载到单片机中，先进行部分调试，打开电源运行程序，检查显示电路部分是否可正常运行，调试结果不正确时要修改程序，然后再下载到单片机中进行再次调试，直到显示器有和要求的显示结果一样为止。再下载程序到按键测试通过、显示器显示正常为止。

在系统软件调试上主要遇到以下几个问题：

(1)串口屏显示乱码。

解决方法：在本设计中用到了串口屏显示，刚开始以为串口屏只要一条条给它发指令就能显示出需要的字符，结果并不是这样，有时候会出现乱码现象。通过查串口屏手册发现，串口屏的刷新频率有所限制不能在很短的时间完成多条指令的更新，因此将所有指令写成一个块函数，并加上一定的延迟，重新下载程序后就没出现乱码的现象。

(2)外部中断时间不准确。

解决方法：在本设计中电机的转速通过外部中断计算脉冲每一周期的时间，而经过试验发现与实际周期有所不同，通过检查外部中断服务函数发现，外部中断函数含有大量的计算过程。因此将服务函数中的数据用另一个变量记录下来不做计算，将计算部分放入主函数，不占用外部中断时间，重新下载程序后与实际周期相符。软件调试结果如图 4-5

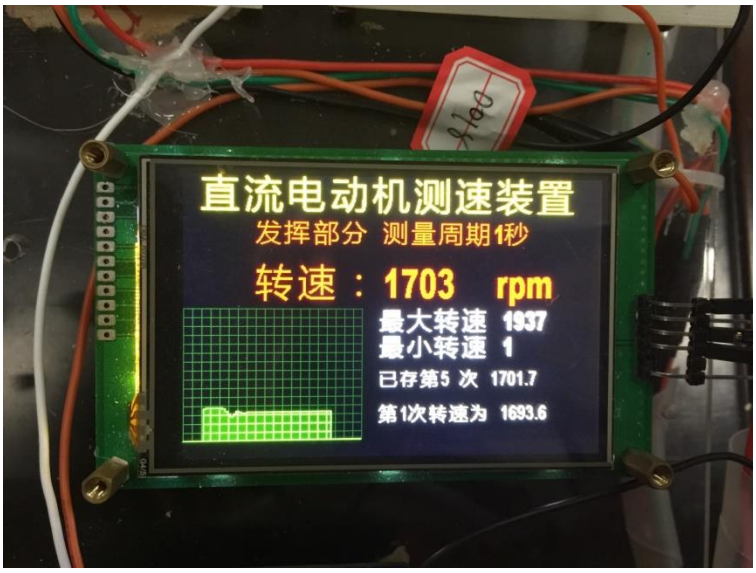


图 4-5 软件调试结果

4.3 漏磁法测速分析计算

首在漏磁中除了以转子电流引起的磁通外，还有占很大部分的电源工频信号引起的磁通，所以在检测转子电流频率过程中，必须依靠滤波器抑制电源工频信号。电机转子电流频率 f2，定子电流频率为 f1，转差率 S 与其关系为：

$$F2=Sf1 \quad n=n0（1-S） \tag{1}$$

$$n=（60/P）x（f1-f2） \tag{2}$$

$$n=（60/P）x（1/T1-1/T2） \tag{3}$$

公式中 P 是磁极对数，n 是电机转速，n0 是电机同步转速，T1 定子电流变化周期，T2 转子电流变化周期。

4. 4 测试方案与测试结果

4.4.1 测试条件与仪器

测试条件：检查多次，硬件电路必须与系统原理图完全相同，并且检查无误，硬件电路保证无虚焊。

测试仪器：

（1）直流稳压电源:QJ−3005S

（2）示波器：RIGOL DS1102C

（3）万用表：VICTOR VC890D

（4）函数信号发生器：JC5620P

（5）非接触式转速计：UT370。

4. 4. 2 测试结果及分析

以检测电动机壳外电磁信号的方式所测量的转速结果如表 2 所示

表 2 漏磁转速测量结果

序 号 测试项目	1	2	3	4	5	6	7	8	9	10
实际转速/rpm	596.3	929.7	1505	2015	2597	3124	3443	4047	4541	5003
单片机转速/rpm	596. 8	929. 6	1503	2014	2596	3124	3445	4051	4539	4998
转速误差%	0. 084	−0. 01	−0. 13	−0. 05	−0. 04	0	0. 058	0. 099	−0. 04	−0. 10
是否符合要求	符合	符合	符合	符合	符合	符合	符合	符合	符合	符合

以上数据结果符合要求。漏磁转速测量结果簇状图如图 4−6 所示。

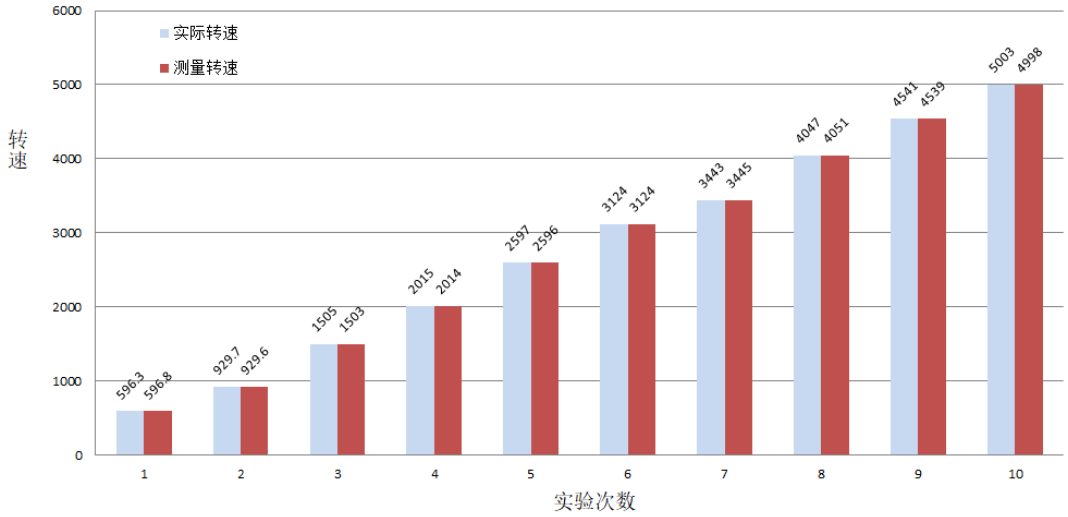


图 4-6 漏磁转速测量结果簇状图

根据上述测试数据，由此可以得出以下结论：

漏磁式电机测速法的精度更高，测量的转速与实际相比较，误差值只有 0. 1%左右，精度更高，误差更小。

结　论

随着集成电路和计算机技术的迅速发展，使电子仪器的整体水平发生巨大变化，传统的仪器逐步的被智能仪器所取代。智能仪器的核心部件是单片机，因其极高的性价比得到广泛的应用与发展，从而加快了智能仪器的发展。而传感器作为测控系统中对象信息的入口，越来越受到人们的关注。传感器好比人体“五官”的工程模拟物，它是一种能将特定的被测量信息（物理量、化学量、生物量等）按一定规律转换成某种可用信号输出的器件或装置本次设计中的全桥电子称就是在以上仪器的基础上设计而成的。因此，只有充分了解有关智能仪器、单片机、传感器以及各部分之间的关系才能达到要求。

本设计完成了基于漏磁式电机测速系统。本装置的硬件电路是由信号采集、数据处理、数据显示、语音播报几个部分组成的。

一、 主要工作及结论

1、熟悉 STC12C5A60S2 单片机功能及工作特性，掌握其接口扩展方法。

2、通过对数据采集的分析，了解了各种传感器、放大器及外部中断有了更深的认识。

3、采用面向对象的思想，分层次、分模块构建设计的总体框架。

二、 存在的问题

1、系统设计不够优化，有待改善。对于自制电感线圈的摆放位置会影响测量数据的准确性。

2、没有扩展更多电路，如通讯接口电路等。通讯接口电路可以与上位机（PC 机）进行通讯，从而将大量的数据存于上位机，然后通过串口或并口通讯与测速装置相连，达到远距离控制的目的。

3、对各种实用芯片价格了解不够，选择上任有欠缺。

由于水平有限，设计的比较粗糙，硬件设计和软件设计方面都还有待于完善。却也为我今后的学习和工作留下了积极的影响。

致　谢

大学生涯即将结束，临近毕业，每天忙忙碌碌，经过几个月的努力，终于按照毕业设计进度要求如期完成了漏磁式测速装置系统的硬件设计任务。在做毕业设计的过程中，虽然碰到了不少的困难，但是在老师的指导以及自己的努力下，终于取得了一定成果。在这段经历中中，带给我的是学生生涯里前所未有的收获。虽然在实物制作与论文写作过程中遇到了无数的困难和障碍，都在同学和老师的帮助下克服并顺利解决了。我查阅了许多有关检流芯片、外部中断和 C 程序设计方面的文献资料，使我对检流芯片的设计和使用有了更深的认识。从论文选题到完成论文，我要强烈感谢我的论文指导老师胡冬生老师，他渊博的知识、严谨的治学态度，求实的精神，时刻感染着我，没有他对我进行了不厌其烦的指导和帮助，无私的为我进行论文的修改和改进。同时，我也要感谢本论文所引用的各位学者的专著，如果没有这些学者的研究成果的启发和帮助，我将无法完成本篇论文的最终写作。我也要感谢周围不断鼓励、支持、帮助我的同学们，他们在我写论文的过程中给予我了很多有用的素材，也在论文的排版和撰写过程中提供热情的帮助，是你们的友情、关心、爱心让我倍感生活的精彩和温暖。最后，我要特别感谢我的家人，他们辛勤的付出，无私的关爱让我体会到亲情的伟大。

金无足赤，人无完人。由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和同学批评和指正！

参考文献

[1]胡仁杰,堵国樑. 全国大学生电子设计竞赛优秀作品设计报告选编[M]. 南京: 东南大学出版社,2014.

[2]谭浩强. C 语言程序设计[M]. 2 版,北京: 清华大学出版社,2008.

[3]堵国樑,吴建辉,樊兆雯等. 模拟电子电路基础[M]. 北京:机械工业出版社,2014.

[4] 郭敬枢,庄继东,孔峰. 微机控制技术[M]，重庆大学出版社,1994.

[5] 刘国荣. 单片微型计算机技术[M]，机械工业出版社,1996.

[6]王益权，电动机原理与实用技术，北京；科学出版社，2005. 7

[7]周美娟等. 单片机技术及系统设计. 北京：清华大学出版社，2007

[8] 张鑫. 单片机原理及应用[M]. 北京：电子工业出版社,2010. 10

[9] 郭天祥. 新概念单片机郭天祥 51 单片机 c 语言教程[M]. 电子工业出版社，2009. 1

[10] 王文成，李健. 基于单片机的电机转速测量系统的设计[J]，2011.

[11] 朱嵘涛，武洪涛. 基于增量式 PID 算法的直流电机调速系统[J]，2017.

[12] 陈博炜. 基于单片机的转速测量系统设计[J]，2012. 9

[13] 陆丽婷，项 岩. 基于单片机的电机测速系统设计[J]，2017. 9

[14] 郑隆举，李慧芳，王志全，石 蕊. 基于单片机的电机测速系统设计[J]，2015. 3

[15] 谭浩强. C 语言程序设计[M]. 2 版,北京：清华大学出版社，2008

[16] 堵国樑，吴建辉，樊兆雯等. 模拟电子电路基础[M] . 北京：机械工业出版社,，2014

[17] Yeager Brent.How to troubleshoot your electronic scale[J]..Powder and Bulk Engineering.1995

[18] Meehan Joanne,Muir Lindsey.SCM in Merseyside SMEs:Benefits and barriers[J]..TQM Journal.2008

附录一 程序源码

```
//////////////////////////////////////////////////////////////////////////////////
//管脚定义在 config. h 文件中
//////////////////////////////////////////////////////////////////////////////////
#include <MAIN. H>
u8 buf_display[150];
u8 buf_YUYING[150];
float save_Speed[10];
extern uint second, num_buf, TH0_buf, TL0_buf, down_times_buf;
long time = 0;
extern u8 flag_high, flag_keynum;
extern unsigned long down_times;
float PL = 0, PL_buf = 0;
bit flag = 0, flag_display = 0;
extern uint clean;
u8 display_1_or_2 = 0, memory_num = 0, tt = 0, save_num = 0;
u16 set_num = 300;
int PL_MAX = 0, PL_MIN = 32760;
bit flag_min, flag_excel, flag_display_4, flag_display_5;
void display();
void yuying();
void display_excel();

/*************************主函数*************************/
void main()
{
    time_init();
    pwm_init();        //两路 pwm 调节 P1. 3 P1. 4
    uart1_init();
    PID_init();
    delayms(5);
    sprintf(buf_display, "CLS(0);\r\n");
    PrintString(buf_display);
    delayms(5);
    sprintf(buf_display, "DS48(20, 0, '直流电动机测速装置', 32, 0);\r\n");
```

```
PrintString(buf_display);
delayms(10);
sprintf(buf_display,"DS48(20, 0,'直流电动机测速装置', 32, 0);\r\n");
PrintString(buf_display);
delayms(10);
sprintf(buf_display,"DQX(0, 150, 10, 15, 20, 10);\r\n");
PrintString(buf_display);
delayms(10);
sprintf(buf_YUYING,"当前系统工作电压 4.95 伏 初始化成功  ", PL);YING(&buf_YUYING);

delayms(8000);
PL_MIN = 32750;
flag_min=0;
P3M1=0X00;
P3M0=0X80;
jdq=0;
while(1)
{

    if(clean >=300)
    {PL = 0;}
    if(flag == 1 )
    {
        PL = (down_times_buf*11058200.0)/time;
        flag = 0;
        PL=PL*6;          //每分钟转速 乘以 60 除以磁极对数
        if(PL>0)flag_min=1;
    }
    if(flag_display == 1){
        flag_display = 0;
        display();
    }
    if(flag_keynum==1)    //按键一 周期 2 秒
    {
        display_1_or_2 = 1;
        flag_keynum = 0;
```

```
            set_num =  300; //1.8 秒
            jdq = 0;
        }
        if(flag_keynum==2)    //按键二 周期 1 秒
        {
            display_1_or_2 = 2;
            flag_keynum = 0;
            set_num = 150;   //0.9 秒
            jdq = 1;
        }
        if(flag_keynum==3)    //按键三 语音播报
        {
            flag_keynum = 0;
            yuying();
        }
        if(flag_keynum==4)    //按键四 数据存储
        {
            flag_keynum = 0;
            flag_display_4= 1;
            tt++;
            save_Speed[tt] = PL;
        }
        if(flag_keynum==5)    ////按键五 存储显示
        {
            flag_keynum = 0;
            save_num++;
            if(save_num>tt){save_num = 1;}
            flag_display_5 = 1;
        }
    }

}

/*****************************显示函数*****************************/

void  display()
```

```
{
    if(PL<1000){
        sprintf(buf_display,"DS48(80,100,'转速：%3.1f   rpm   ',1,0);\r\n",PL);  //四位数换挡 小于1000 显示3位整数一位小数
        PrintString(buf_display);
    }
    else  if(PL>=1000){
        sprintf(buf_display,"DS48(80,100,'转速：%4.0f   rpm   ',1,0);\r\n",PL);  //四位数换挡 大于等于1000 显示4位整数
        PrintString(buf_display);
    }
        delayms(10);
    if(display_1_or_2==1) //按键一 周期2秒
        {
            sprintf(buf_display,"DS32(80,50,'  基础题 测量周期2秒   ',1,0);\r\n");
            PrintString(buf_display);
            delayms(10);
        }
    if(display_1_or_2==2) //按键一 周期2秒
        {
            sprintf(buf_display,"DS32(80,50,'发挥部分 测量周期1秒 ',1,0);\r\n");
            PrintString(buf_display);
            delayms(10);
        }
    if(PL>PL_MAX)
    {
        PL_MAX = PL;
        sprintf(buf_display,"DS32(220,150,'最大转速 %d   ',16,0);\r\n",(int)PL_MAX+1);
        PrintString(buf_display);
        delayms(3);
    }
    if(PL<PL_MIN)
    {
        if(flag_min==1)PL_MIN = PL;
        sprintf(buf_display,"DS32(220,180,'最小转速 %d   ',16,0);\r\n",(int)PL_MIN+1);
        PrintString(buf_display);
```

```
            delayms(3);
        }
        if(flag_display_4==1)
        {
            flag_display_4 = 0;
            sprintf(buf_display,"DS24(220,220,'已存第%d 次 %4.1f
',16,0);\r\n",(int)tt,(float)save_Speed[tt]);
            PrintString(buf_display);
        }
        if(flag_display_5==1)
        {
            flag_display_4 = 0;
            sprintf(buf_display,"DS24(220,260,'第%d 次转速为 %4.1f
',16,0);\r\n",(int)save_num,save_Speed[save_num]);
            PrintString(buf_display);
        }

        sprintf(buf_display,"S%d;\r\n",(int)(PL/30));
        PrintString(buf_display);
}


void yuying()                    //语音模块
{
        if(PL<1000)              {PL = (int)(PL*10)/10.0;}
        else if(PL>=1000){PL = (int)(PL);}
        if(PL != PL_buf){
            PL_buf = PL;
            if(PL<1000) {
                sprintf(buf_YUYING,"当前转速为 %3.2f 转
",PL);YING(&buf_YUYING);delayms(2000);
            }
            else if(PL>=1000){
                sprintf(buf_YUYING,"当前转速为 %4.1f 转
",PL);YING(&buf_YUYING);delayms(2000);
            }
```

```
    }
}

    #include "delay.h"

void delayms(unsigned int x)    //11.0592M
{
    unsigned int a,b;
    for(a=x;a>0;a--)
        for(b=847;b>0;b--);

}
//void delayms(unsigned int x)    //12M
//{
//    unsigned int a,b;
//    for(b=x;b>0;b--)
//        for(a=920;a>0;a--);
//}
#include "key.h"

 uchar KEY_NUM=0;
/////////////////////////////////////////////////////////////////////////////////
//本程序应用于 STC12C5A60S2
//独立键盘
/////////////////////////////////////////////////////////////////////////////////
//#include "KEY.h"

bit keyS1_OLD=1;
bit keyS2_OLD=1;
bit keyS3_OLD=1;
bit keyS4_OLD=1;
bit keyS5_OLD=1;
bit keyS6_OLD=1;
bit keyS7_OLD=1;
```

```
bit  keyS1_NOW=1;
bit  keyS2_NOW=1;
bit  keyS3_NOW=1;
bit  keyS4_NOW=1;
bit  keyS5_NOW=1;
bit  keyS6_NOW=1;
bit  keyS7_NOW=1;



uchar  keyscan()
{
u8  i=0;
       keyS1_NOW=keyS1;
       keyS2_NOW=keyS2;
       keyS3_NOW=keyS3;
       keyS4_NOW=keyS4;
       keyS5_NOW=keyS5;

    if(keyS1_NOW  &&keyS7_NOW)
       {

            i=0;
       }
      if(keyS1_NOW  ==0)
      {
           if(keyS1_OLD==1)  i=  1  ;
      }
      if(keyS2  ==0)
      {
           if(keyS2_OLD==1)i=  2  ;
      }
      if(keyS3    ==0)
      {
           if(keyS3_OLD==1)i=  3;
      }
```

```
    if(keyS4  ==0)
    {
        if(keyS4_OLD==1)i= 4 ;
    }
    if(keyS5  ==0)
    {
        if(keyS5_OLD==1)i= 5 ;
    }


    keyS1_OLD= keyS1_NOW;
    keyS2_OLD= keyS2_NOW;
    keyS3_OLD= keyS3_NOW;
    keyS4_OLD= keyS4_NOW;
    keyS5_OLD= keyS5_NOW;


    return i ;
}
#include "time.h"
#include "key.h"
#include "pid.h"
uint num = 0, second = 0, num_buf = 0, TH0_buf = 0, TL0_buf = 0, down_times_buf = 0, clean = 0;
u8 keynum=0, flag_keynum=0;
u8 flag_high = 0;
uint t = 0, key_delay = 0;
extern long time;
unsigned long down_times = 0;
extern bit flag, flag_display;
extern u16 set_num;
sbit LED = P0^0;
//*****************************************************
//中断初始化
//*****************************************************
void time_init()
```

```
{
    AUXR |= 0x80;       //定时器时钟 1T 模式
    TMOD &= 0xF0;       //设置定时器模式
    TMOD |= 0x01;       //设置定时器模式
    TL0 = 0;            //设置定时初值
    TH0 = 0;            //设置定时初值
    TF0 = 0;            //清除 TF0 标志
    TR0 = 1;            //定时器 0 开始计时

    IT1=1;
    EX1=1;

    IPH=0X04;
    IP=0X04;
    EA=1;
    ET0=1;
}
//*************************************************
//定时器中断
//*************************************************
void timer0() interrupt 1
{
    num++;
    t++;
    key_delay++;
    clean++;
    if(t>100)
    {
        t = 0;
        flag_display = 1;
    }
    if(key_delay>4)
    {
        key_delay = 0;
        keynum=keyscan();
        if(keynum==1){flag_keynum=1;}
```

```
            if(keynum==2){flag_keynum=2;}
            if(keynum==3){flag_keynum=3;}
            if(keynum==4){flag_keynum=4;}
            if(keynum==5){flag_keynum=5;}
        }
}


//*****************************************************
//外部中断
//*****************************************************
void time1() interrupt 2
{

    down_times++;
    if(num>=set_num)
    {
    down_times_buf = down_times;
    num_buf = num;
    TH0_buf=TH0;
    TL0_buf=TL0;
    down_times = 0;
    TH0=0;
    TL0=0;
    time = num_buf*65535+  TH0_buf*256 + TL0_buf;
    num_buf=0;
    num = 0;
    }
    clean = 0;
    flag = 1;
}
#include "uart.h"
#include <string.h>
char buf[100];
/*************** ????? ****************************/
```

```
#define  Baudrate1     115200     //define the baudrate,  ????BRT???????, ????????2??
                                  //12T mode: 600~115200 for 22.1184MHZ,  300~57600 for
11.0592MHZ
#define  Baudrate2     19200      //define the baudrate2,
                                  //12T mode: 600~115200 for 22.1184MHZ,  300~57600 for
11.0592MHZ

#define       BUF_LENTH 128      //??????????

/*************************************************************/


unsigned char     uart1_wr;          //???
unsigned char     uart1_rd;     //???
unsigned char     xdata RX1_Buffer[BUF_LENTH];
bit       B_TI;

unsigned char     uart2_wr;          //???
unsigned char     uart2_rd;     //???
unsigned char     xdata RX2_Buffer[BUF_LENTH];
bit       B_TI2;


/***************** ???????, ?????? ***********************************/

#define T1_TimerReload    (256 - MAIN_Fosc / 192 / Baudrate1)        //Calculate the timer1
reload value  at 12T mode
#define BRT_Reload        (256 - MAIN_Fosc / 12 / 16 / Baudrate2)    //Calculate BRT
reload value

#define   TimeOut1        (28800 / (unsigned long)Baudrate1 + 2)
#define   TimeOut2        (28800 / (unsigned long)Baudrate2 + 2)

#define   TI2             (S2CON & 0x02) != 0
#define   RI2             (S2CON & 0x01) != 0
```

```
#define   CLR_TI2()        S2CON &= ~0x02
#define   CLR_RI2()        S2CON &= ~0x01

/*****************************************************/

/******************** ?????? ***************/
void uart1_init(void);
void uart2_init(void);
void UART1_TxByte(unsigned char dat);
void UART2_TxByte(unsigned char dat);
void PrintString1(char *puts);

char putchar(char c);


void UART1_TxByte(unsigned char dat)
{
    B_TI = 0;
    SBUF = dat;
    while(!B_TI);
    B_TI = 0;
}

//void   UART2_TxByte(unsigned char dat)
//{
//  B_TI2 = 0;
//  S2BUF = dat;
//  while(!B_TI2);
//  B_TI2 = 0;
//}

void PrintString(char *puts)          //???????
{
    for (; *puts != 0;    puts++)   UART1_TxByte(*puts);    //?????0??
}
```

```
char  putchar(char  c)
{
    UART1_TxByte(c);
    return  c;
}

void uart1_init(void)    //11.0592M  115200 波特率
{
    PCON  &= 0x7F;        //波特率不倍速
    SCON  = 0x50;         //8 位数据,可变波特率
    AUXR  |= 0x04;        //独立波特率发生器时钟为 Fosc,即 1T
    BRT  = 0xFD;          //设定独立波特率发生器重装值
    AUXR  |= 0x01;        //串口 1 选择独立波特率发生器为波特率发生器
    AUXR  |= 0x10;        //启动独立波特率发生器
    ES   = 1;
    EA  = 1;
}

/*********************************************/
void  UART0_RCV  (void)  interrupt  4
{
    if(RI)
    {
        RI  = 0;
        RX1_Buffer[uart1_wr]  =  SBUF;
        if(++uart1_wr  >=  BUF_LENTH)uart1_wr  =  0;
    }

    if(TI)
    {
        TI  = 0;
        B_TI  = 1;
    }
}
```

```
/*********************************************/

/*********************************************/
void UART2_RCV (void) interrupt 8
{
    if(RI2)
    {
        CLR_RI2();
        RX2_Buffer[uart2_wr] = S2BUF;
        if(++uart2_wr >= BUF_LENTH)uart2_wr = 0;
    }

    if(TI2)
    {
        CLR_TI2();
        B_TI2 = 1;
    }
}

void PrintString2(char *puts, unsigned char len)        //发送一串字符串
    {
        unsigned char i = 0;
        for (i=0; i < len+5;  i++)
        UART1_TxByte(*(puts+i));      //遇到停止符 0 结束
    }

void YING(unsigned char *szText)
{
    unsigned char nLength=1;
    unsigned char i=1;
    nLength = strlen(szText);  //strlen 所作的仅仅是一个计数器的工作，它从内存的某个位置（可以是字符串开头，中间某个位置，甚至是某个不确定的内存区域）开始扫描，直到碰到第一个字符串结束符'\0'为止，然后返回计数器值(长度不包含'\0')。
    buf[0] = 0XFD;
```

```
    buf[1] = 0X00;
    buf[2] = nLength+2;
    buf[3] = 0x01;
    buf[4] = 0x00;
    for( i=0; i<nLength; i++){      buf[5+i] = szText[i];}
    PrintString2(&buf, nLength);
    /*发送过程 */
}
#ifndef _config_H__
#define _config_H__
#define  u8 unsigned char
#define u16 unsigned int
#define  uchar unsigned char
#define uint unsigned int
#define MAIN_Fosc          11059200L
#include〈STC12C5A60S2. H〉


sbit keyS1=P2^0; //按键
sbit keyS2=P2^1;
sbit keyS3=P2^2;
sbit keyS4=P2^3;
sbit keyS5=P2^4;

sbit ain1=P3^4;          //AD6612
sbit ain2=P3^5;
sbit bin1=P3^6;
sbit bin2=P3^7;
sbit find=P0^2;
sbit stop_signal=P0^3;
sbit beep = P0^4;
sbit jdq= P3^7;          //继电器
extern int adc1, adc2, adc3, adc4, adc5, adc6;
#endif
```