中文题目:基于51单片机的多功能小车设计与实现

外文题目:Design and Realization of multi function car based on 51 single chip microcomputer

毕业设计（论文）共 73 页（其中：外文文献及译文 13 页）图纸共 3 张

完成日期 2015 年 6 月　　答辩日期 2015 年 6 月

摘要

本毕设是基于 51 单片机为核心的智能小车，这个设计是为了奠定自己实际工作的基础，该设计中小车以 STC89C52 单片机为主控芯片控制系统，小车的驱动由 L293D 驱动电路完成，速度由单片机输出的 PWM 波进行控制，使用红外对管模块、红外接收模块、蓝牙无线遥控模块、超声波发射与接收模块、舵机模块和电路驱动模块等构成外围扩展电路。通过将小车的控制电路、四轮小车机械结构和控制程序以及手机蓝牙操控软件几者相结合，在对小车进行实际环境的测试，最后根据得出的测试结果对程序进行调试，最终使智能小车可以完美实现蓝牙遥控、红外遥控、红外避障、自动循迹、超声波避障功能。

关键词：智能小车；STC89C52单片机；L293D；蓝牙遥控

Abstract

The graduation project is a Smart car, designed to lay the groundwork to achieve their future practical work, the design of the car as the main chip microcontroller with STC89C52 control system, the car driven by the completion L293D driver circuit speed by the microcontroller output PWM wave control, infrared tube module, infrared receiver module, Bluetooth wireless remote control module, an ultrasonic transmitter and receiver modules, modules, and circuits servo drive module and the like peripheral expansion circuit. By car control circuit, four-wheel car mechanical structure and control program, and Bluetooth phone control software several are combined in the actual environment of the car test, the final test results obtained in accordance with the program debugging, and finally to the smart car You can achieve the perfect Bluetooth remote control, infrared remote control, infrared obstacle avoidance, automatic tracking, ultrasonic obstacle avoidance function.

Keywords： Smart car；STC89C52 MCU；L293D；Bluetooth telecontrol

目录

引言

最近几年智能化这个概念已经深刻的影响到我们的生活，在我们的世界中充满了各种智能化产品，例如国家重工的航天航空，还有最近流行的智能家居，或者是谷歌的 AlphaGo，以及至少人手一个的智能手机，可见智能化对我们的的生活造成的影响是不可想象的，它是未来长期的主要发展概念。本次设计也是基于这个概念，设计一个多功能的智能小车，为自己以后的实际生产奠定基础。本设计的小车主要功能是循迹，扩展功能有超声波避障与红外避障、红外遥控与蓝牙遥控。但是由于各方面的影响，主要是硬件问题，导致一些功能冲突，因此本小车不得不去掉一些功能，这是本次设计最可惜之处，但也为自己以后的研究积累了非常好的经验。

1 项目的概述

1.1 选题背景

智能化作为近代社会非常主流的一种新概念，也是未来的主流发展方向，它可以按照预先设定的模式在一个特定的环境里自动的运作，无需人为管理，便可以完成预期所要达到的或是更高的目标。在这种发展趋势下，决定选择智能小车这个课题，该课题设计的智能小车可以通过计算机编程来实现其对行驶方向、启停以及速度的控制，无需人工干预，是一个集环境感知、规划决策，自动行驶等功能于一体的综合系统，它集中地运用了计算机、传感、信息、通信、导航、人工智能及自动控制等技术，是一个技术综合体。

1.2 目的及意义

选题的目的在于利用所学单片机知识设计一个控制系统，独立设计并制作一辆具有智能化的简易小车，将自己所学的理论知识，通过自己亲手的实践转换成经验。

选题的意义在于培养自己的设计开发能力，还有加深课堂上的学习。由于在校事单片机教学例子有限，不能很好的将自己的理论知识转化成有用的经验，而亲自动手的单片机智能车能很好的综合课堂上的知识，使自己能更好的了解单片机的相关知识。通过此次的智能小车制作，还可以使自己从理论到实践，初步体会单片机项目的设计、制作、调试和成功完成项目的过程及困难，以此学会用理论联系实际，通过对实践中出现的不足与学习来补充教学上的盲点。

1.3 国内外研究现状与发展趋势

随着航天工业与汽车工业的迅速发展，关于智能车的研究也就越来越受人关注。集多功能于一身的智能探测遥控小车和高度智能的无人驾驶车，各个国家都在大力研究，尤其是在工业大国，由于政府的资助，并且起步较早，而且高端工业发展完善，技术上占据着明显优势。而我国的研究工作始于 20 世纪中后期，目前在国家的重点支持下，全国的一些高科技资金雄厚的公司正在大力的研究，同时全国电子大赛和省内电子大赛几乎每次都有这方面的题目，全国各高校也很重视该题目的研究，通过这些年的不断研究，智能小车的功能不断地扩展与完善，包括原有循迹、避障与无线遥控等功能的实现技术也是更加先进，但是与国际还存在一定的差距，这就是目前的现状和以后的发展趋势。

2 相关技术

2.1 单片机

单片机的基本结构：运算器，控制器，主要寄存器。

2.1.1 运算器

运算器由运算部件——算术逻辑单元（Arithmetic & Logical Unit，简称 ALU）、累加器和寄存器等几部分组成。ALU 的作用是把传来的数据进行算术或逻辑运算，输入来源为两个 8 位数据，分别来自累加器和数据寄存器。ALU 能完成对这两个数据进行加、减、与、或、比较大小等操作，最后将结果存入累加器。运算器有两个功能：

（1）执行各种算术运算。

（2）执行各种逻辑运算，并进行逻辑测试，如零值测试或两个值的比较。

运算器所执行全部操作都是由控制器发出的控制信号来指挥的，并且一个算术操作产生一个运算结果，一个逻辑操作产生一个判决。[1]

2.1.2 控制器

控制器由程序计数器、指令寄存器、指令译码器、时序发生器和操作控制器等组成，是发布命令的“决策机构”，即协调和指挥整个微机系统的操作。其主要功能有：

（1）从内存中取出一条指令，并指出下一条指令在内存中的位置。

（2）对指令进行译码和测试，并产生相应的操作控制信号，以便于执行规定的动作。

（3）指挥并控制 CPU、内存和输入输出设备之间数据流动的方向。

微处理器内通过内部总线把 ALU、计数器、寄存器和控制部分互联，并通过外部总线与外部的存储器、输入输出接口电路联接。外部总线又称为系统总线，分为数据总线 DB、地址总线 AB 和控制总线 CB。通过输入输出接口电路，实现与各种外围设备连接。[2]

2.1.3 主要寄存器

（1）累加器 A

累加器 A 是微处理器中使用最频繁的寄存器。在算术和逻辑运算时它有双功能：运算前，用于保存一个操作数；运算后，用于保存所得的和、差或逻辑运算结果。

（2）数据寄存器 DR

数据寄存器通过数据总线向存储器和输入/输出设备送（写）或取（读）数据的暂存单元。它可以保存一条正在译码的指令，也可以保存正在送往存储器中存储的一个数据字

节等等。

（3）指令寄存器 IR 和指令译码器 ID

指令包括操作码和操作数。

指令寄存器是用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存中取到数据寄存器中，然后再传送到指令寄存器。当系统执行给定的指令时，必须对操作码进行译码，以确定所要求的操作，指令译码器就是负责这项工作的。其中，指令寄存器中操作码字段的输出就是指令译码器的输入。

（4）程序计数器 PC

PC 用于确定下一条指令的地址，以保证程序能够连续地执行下去，因此通常又被称为指令地址计数器。在程序开始执行前必须将程序的第一条指令的内存单元地址（即程序的首地址）送入 PC，使它总是指向下一条要执行指令的地址。

（5）地址寄存器 AR

地址寄存器用于保存当前 CPU 所要访问的内存单元或 I/O 设备的地址。由于内存与 CPU 之间存在着速度上的差异，所以必须使用地址寄存器来保持地址信息，直到内存读/写操作完成为止。当 CPU 向存储器存数据、CPU 从内存取数据和 CPU 从内存读出指令时，都要用到地址寄存器和数据寄存器。[3]

2.2 传感器

2.2.1 红外传感器

红外对管是红外线发射管与光敏接收管，或者红外线接收管，或者红外线接收头配合在一起使用时候的总称。

红外发射管是由红外发光二极管矩组成发光体，用红外辐射效率高的材料（常用砷化镓）制成 PN 结，正向偏压向 PN 结注入电流激发红外光，其光谱功率分布为中心波长 830～950nm。

光敏接收管是一个具有光敏特征的 PN 结，属于光敏二极管，具有单向导电性，因此工作时需加上反向电压。无光照时，有很小的饱和反向漏电流（暗电流）。此时光敏管不导通。当光照时，饱和反向漏电流马上增加，形成光电流，在一定的范围内它随入射光强度的变化而增大。

红外线接收管功能与光敏接收管相似只是不受可见光的干扰，感光面积大，灵敏度高，属于光敏二极管，一般只对红外线有反应。[4]

2.2.2 蓝牙传感器

蓝牙无线传感器主要包括两大模块:传感器模块(Sensor Module)和蓝牙无线模块(Bluetooth Module):前者主要用于进行现场信号的数据采集,将现场信号的模拟量转化为数字量,并完成数字量的变换和存储。后者运行蓝牙无线通信协议,使得传感器设备满足蓝牙无线通信协议规范,并将现场数据通过无线的方式传送到其它蓝牙设备当中。两模块之间的任务调度、相互通信,以及同上位机通信的流程由控制程序控制完成。控制程序包含一种调度机制,并通过消息传递的方式完成模块间的数据传递以及同其它蓝牙设备的通信,从而完成整个蓝牙无线系统的功能。[5]

2.2.3 超声波传感器

超声波传感器是利用超声波的特性研制而成的传感器。超声波是一种振动频率高于声波的机械波，由换能晶片在电压的激励下发生振动产生的，它具有频率高、波长短、绕射现象小，特别是方向性好、能够成为射线而定向传播等特点。超声波对液体、固体的穿透本领很大，尤其是在阳光不透明的固体中，它可穿透几十米的深度。超声波碰到杂质或分界面会产生显著反射形成反射成回波，碰到活动物体能产生多普勒效应。基于超声波特性研制的传感器称为“超声波传感器”，广泛应用在工业、国防、生物医学等方面。

超声波传感器主要由如下四个部分构成：

发送器：通过振子（一般为陶瓷制品，直径约为 15mm）振动产生超声波并向空中幅射。

接收器：振子接收到超声波时，根据超声波发生相应的机械振动，并将其转换为电能量，作为接收器的输出。

控制部分：通过用集成电路控制发送器的超声波发送，并判断接收器是否接收到信号（超声波），以及已接收信号的大小。

电源部分：超声波传感器通常采用电压为 DC12V ± 10 % 或 24V ± 10 %外部直流电源供电，经内部稳压电路供给传感器工作。[6]

3 总体设计

3.1 设计方案

根据题目要求，确定如下设计方案：在一个扩展性很大的小车底盘上加装主控系统与各个模块，包括超声波模块、舵机模块、蓝牙模块、单片机控制模块、动力和转向模块、以及三个红外模块等，通过将这些模块结合到一起，并烧入程序，从而完成智能小车。智能小车通过红外模块、超声波模块、蓝牙模块收集信息，然后对信息进行处理，并将处理后的信息传输给控制器，控制器作出相应的处理，在对智能小车进行控制，智能车进驶过程中，采用双极式 H 型 PWM 脉宽调制技术实现快速、平稳地的调速。这个设计方案能实现智能小车运动状态的实时控制，而且控制灵活、可靠，有较高的精度，可满足各种要求，而且还有很大的扩展空间，方便以后的研究。

3.2 系统硬件总体设计

3.2.1 系统硬件结构

智能小车采用四轮驱动，前后左右各有一个电机驱动，前进与后退所有的轮子必须保证速度一致才能直线前行，当小车需要转向时，小车一侧轮子向前转，一侧向后转，就实现这个功能了。将循迹光电对管装在车体头部底盘的下面，左右各一个当车身底盘下左边的传感器检测到黑线时，主控芯片控制左侧轮电机停止，车向左修正，当车身底盘下右边传感器检测到黑线时，主控芯片控制右侧轮电机停止，车向右修正。避障采用红外避障和超声波避障，红外避障原理和循线一样，在车身底盘前端上面装一个光电对管，当其检测到障碍物时，主控芯片控制车子转向，从而避开障碍物。超声波避障原理是通过超声波发射模块发射超声波信号，接收模块接收超声波信号，根据探测到的距离，如果是非安全距离，主芯片控制小车转向，进而避开障碍物。小车的遥控模块采用蓝牙模块与红外模块，两者原理都类似，蓝牙操控端与红外发射端负责发射信号，而小车只负责接收信号，然后主芯片处理信号，从而控制小车转向。

3.2.2 系统硬件方案设计

系统的硬件设计主要由五部分组成，也是有几大传感器模块组成，分别为主控系统、电机驱动部分、循迹部分、避障部分、遥控部分，主控系统使用单片机，电机驱动部分使用 H 型桥式电路和 PWM 调速方案，循迹部分使用两组红外对管进行自动循迹；避障部分采用两种方式综合进行避障，主要使用红外避障，并以超声波避障为辅，从而提高小车的

避障精度；遥控部分也采用两种方式综合遥控，其中以红外遥控为主，蓝牙遥控为辅，两种遥控方式都只接收信号，然后控制小车的前进方式，两种传感器都不需要发射信号。以上就是智能小车的主要硬件组成部分。[7]

3.3 系统软件总体设计

系统硬件的设计主要由五部分组成，每个部分还有自己的子部分，系统软件的设计就要包括所有硬件的组成部分，还要体现出主部分包含的子部分，软件部分包含主控部分、电机驱动模块、红外循迹模块、超声波模块、舵机模块、红外避障模块、蓝牙遥控模块、红外模块部分，这八个主要部分。

3.3.1 主控部分

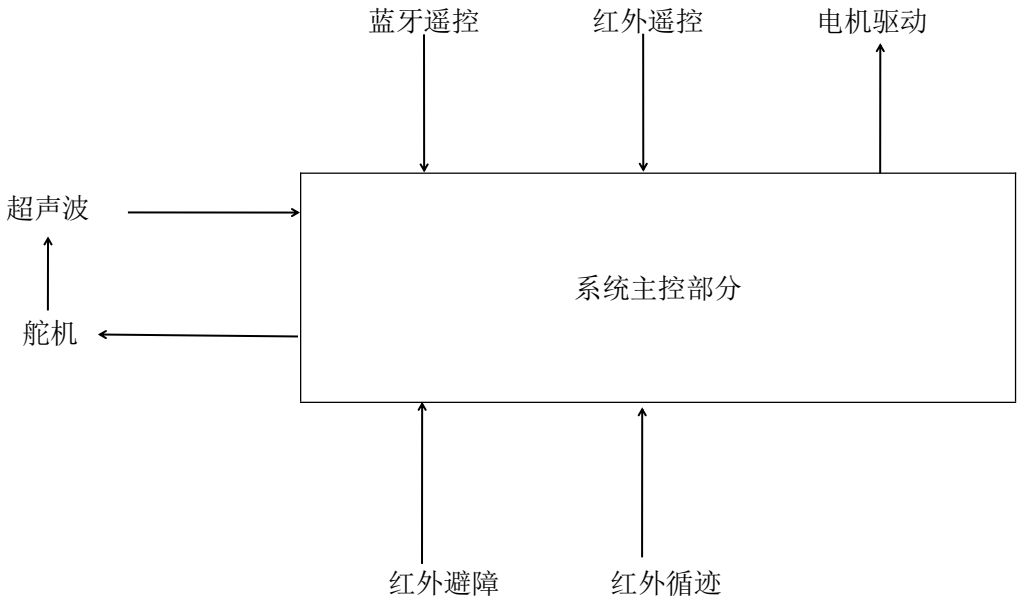


图 3-1 系统框架图

Fig.3-1 The diagram of system framework

系统的主控部分的系统框架图如图 3-1 所示。主控部分由红外循迹模块、红外避障模块、舵机模块、超声波模块、蓝牙模块、红外遥控模块组成，每个模块都收集外界信号，并将这些信号发给系统主控部分，然后系统的主控部分通过处理各个传感器传输回来的信号，根据处理的结果，判断是否要改变小车的行进状态，智能小车的实时控制就是通过这样的方式控制小车。[8]

3.3.2 遥控模块

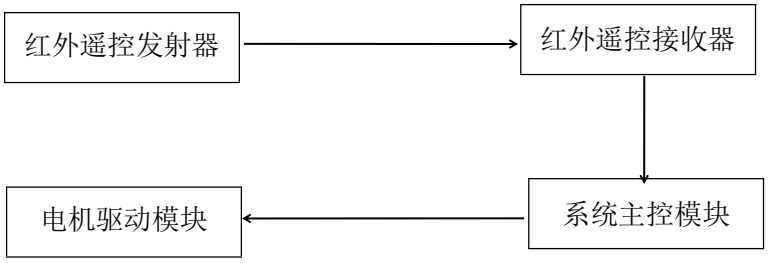


图 3-2 红外遥控模块图

Fig.3-2 The diagram of infrared remote control module

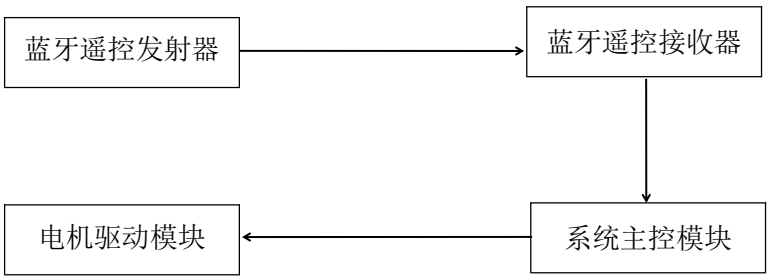


图 3-3 蓝牙遥控模块图

Fig.3-3 The diagram of bluetooth remote control module

智能小车的遥控模块由两部分组成，其中红外遥控模块图如图 3-2 所示，蓝牙遥控模块图如图 3-3 所示。小车的的遥控主要以红外遥控为主，红外遥控可以中断蓝牙遥控。这两个模块主要硬件是红外接收传感器与蓝牙接收传感器，两个传感器只负责接收信号，并传输给主控系统处理，主控系统处里信号，调用电机驱动模块，控制小车的运动状态。[9]

3.3.3 避障模块

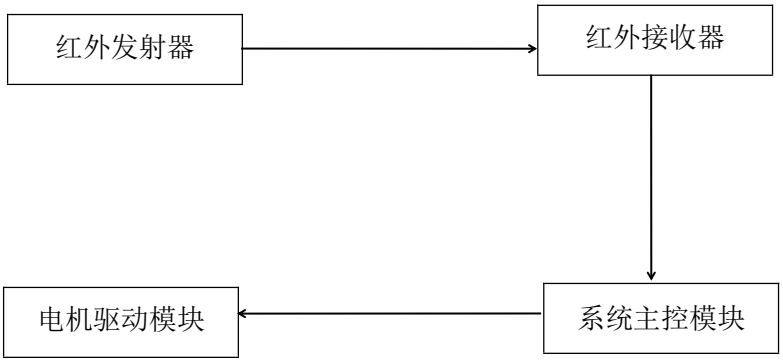


图 3-4 红外避障模块图

Fig.3-4 The diagram of infrared obstacle avoidance module

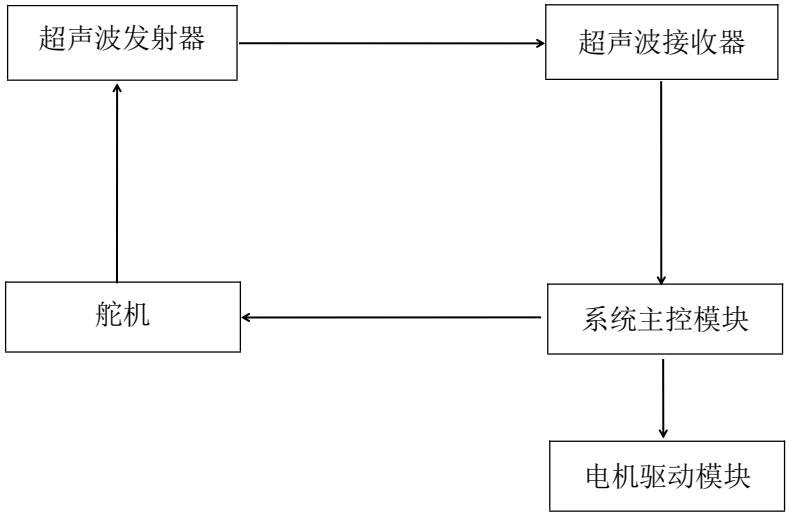


图 3-5 超声波避障模块图

Fig.3-5 The diagram of ultrasonic obstacle avoidance module

智能小车的避障模块由两部分组成，红外避障模块图如图 3-4 所示与超声波避障模块图如图 3-5 所示。两个模块的硬件主要为红外对管与超声波传感器，红外对管由红外发射器与红外接收器组成，通过讲接收回来的信号传输给主控系统，主控系统处理数据，然后调用电机驱动模块，实时控制小车的运动状态。超声波传感器由超声波发射器与超声波接收器组成，同样将接收回来的信号传给主控系统，主控系统处理完数据，如果是安全距离，则小车按照原来的行进方式继续前行，如果是非安全距离，小车停止前进，调用舵机模块，从而转动超声波传感器，并实时的把超声波接收的信号，传送给主控系统，直到为安全距离时，调用电机驱动模块，控制小车的前进的状态。

3.3.4 循迹模块

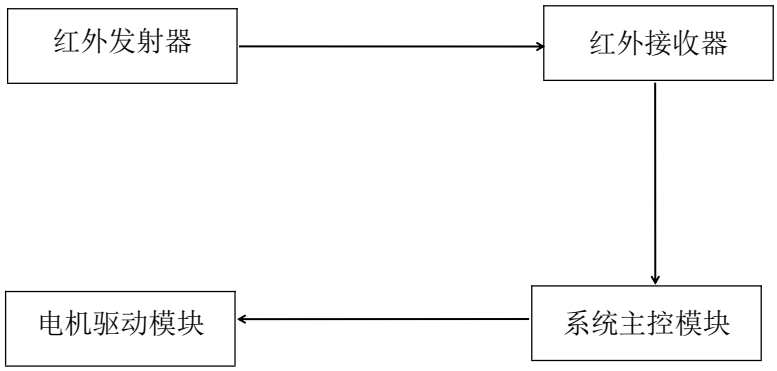


图 3-6 红外循迹模块图

Fig.3-6 The diagram of Infrared tracking module

智能小车的循迹模块图如图 3-6 所示。循迹模块由两对红外对管组成，红外对管由红外发射器与红外接收器组成，红外对管中红外接收器接收信号，并传输给主控部分进行处理，根据处理的结果判断是否调整小车的轨迹。

4 系统硬件设计

4.1 系统硬件设计原则

系统硬件的选择从底盘、控制系统、电池、传感器到电机的都要满足很多条件，底盘要有足够的扩展空间，可以满足后期的扩展和研究的要求。传感器要选择精度高的，而且抗干扰性要强，例如在温度、光照、湿度等等反差很大的情况，测量的数据和返回给控制器的结果不能有稍大偏差。电机要选择功率足够大，而且要非常稳定的。电池最好使用蓄电池，这样电流比较稳定，不会烧坏电机与单片机，而且可重复使用。控制系统要选择能满足所有需求，尤其是具有多开关程序处理的处理器，而且控制简单、方便快捷、廉价。以上就是本次系统硬件要满足的基本原则。

4.2 核心部件选型

4.2.1 STC89C52 单片机

STC89C52 是一种低功耗、高性能 CMOS8 位微控制器，具有 8K 字节系统可编程 Flash 存储器。STC89C52 使用经典的 MCS-51 内核，但是做了很多的改进，使得芯片具有了许多传统 51 单片机不具备的功能。在单芯片上，拥有灵巧的 8 位 CPU 和在系统可编程 Flash，使得 STC89C52 可以为嵌入式控制应用系统提供高灵活、超有效的解决方案。

STC89C52 具有以下标准功能：8k 字节 Flash，512 字节 RAM，32 位 I/O 口线，看门狗定时器，内置 4KB EEPROM，MAX810 复位电路，3 个 16 位定时器/计数器，4 个外部中断，一个 7 向量 4 级中断结构（兼容传统 51 的 5 向量 2 级中断结构），全双工串行口。另外 STC89C52 可降至 0Hz 静态逻辑操作，支持 2 种软件可选择节电模式。空闲模式下，CPU 停止工作，允许 RAM、定时器/计数器、串口、中断继续工作。掉电保护方式下，RAM 内容被保存，振荡器被冻结，单片机一切工作停止，直到下一个中断或硬 件复位为止。最高运作频率 35MHz，6T/12T 可选。

STC89C52 的主要参数：

1. 增强型 8051 单片机，6 时钟/机器周期和 12 时钟/机器周期可以任意选择，指令代码完全兼容传统 8051。

2. 工作电压：5.5V～3.3V（5V 单片机）/3.8V～2.0V（3V 单片机）。

3. 工作频率范围：0～40MHz，相当于普通 8051 的 0～80MHz，实际工作频率可达 48MHz。

4. 用户应用程序空间为 8K 字节。

5. 片上集成 512 字节 RAM。

6. 通用 I/O 口（32 个），复位后为：P0/P1/P2/P3 是准双向口/弱上拉，P0 口是漏极开路输出，作为总线扩展用时，不用加上拉电阻，作为 I/O 口用时，需加上拉电阻。

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（RxD/P3.0,TxD/P3.1）直接下载用户程序，数秒即可完成一片。

8. 具有 EEPROM 功能。

9. 共 3 个 16 位定时器/计数器。即定时器 T0、T1、T2。

10. 外部中断 4 路，下降沿中断或低电平触发电路，Power Down 模式可由外部中断低电平触发中断方式唤醒。

11. 通用异步串行口（UART），还可用定时器软件实现多个 UART。

12. 工作温度范围：-40～+85℃（工业级）/0～75℃（商业级）。

13. PDIP 封装。

本设计采用 STC89C52 单片机作为整个系统的核心的原因是，STC89C52 单片机具有功能强大的位操作指令，I/O 口均可按位寻址，程序空间多达 8K，对于本设计也绰绰有余，而且 STC89C52 单片机价格非常低廉。智能小车的主控系统，其关键在于实现小车的自动控制，而在这一点上，STC89C52 单片机就显现出来它的优势——控制简单、方便、快捷。

而且针对本设计特点——多开关量输入的复杂程序控制系统，需要擅长处理多开关量的标准单片机，而不能用精简 I/O 口和程序存储器的小体积单片机，D/A、A/D 功能也不必选用。根据这些分析,在综合考虑了传感器、四部电机的驱动等诸多因素后，决定使用 STC89C52 单片机。

4.2.2 H6-0C 蓝牙接收器

蓝牙模块，是一种集成蓝牙功能的 PCBA 板，用于短距离无线通讯，按功能分为蓝牙数据模块和蓝牙语音模块，本设计使用蓝牙数据模块。

本设计的模块分主机和从机，主机能和从机配对通信，从机与从机之间或主机与主机之间不能通信，从机能和电脑、手机等的蓝牙配对通信。在做智能小车控制时，蓝牙模块主要是接收从手机端发送过来的指令，所以实际上只需要从机模块就足够满足需求了。蓝牙串口在模块功能上，偶数命名的互相兼容，从机命名的也互相兼容，HC-04 与 HC-06，HC-03 与 HC-05 在功能上是兼容的。HC-04 与 HC-06 是比较早的版本，不可以自己切换主

机或者从机，AT 指令集很少，包括修改蓝牙名，修改密码（模块在出厂时的默认配对密码是 1234），修改波特率，询问版本号等几个基本功能。在本次设计中只需实现简单的通信，因此选用 HC-06 模块就足以满足需求。HC-06 模块只记忆最后一次配对过的从机，并只与该从机配对，直到 KEY（26 脚）高电平触发时放弃记忆，26 脚默认应该为低电平。

HC-06 模块的主要参数：

1、采用 CSR 主流蓝牙芯片，蓝牙 V2.0 协议标准。

2、模块供电电压：3.3V~3.6V。

3、默认参数：波特率 9600、配对码 1234、工作模式 从机。

4、核心模块尺寸大小为：27mm x 13 mm x 2mm。

5、工作电流：不大于 50MA（以实测为准）。

6、通讯距离：空旷条件下 10M，正常使用环境 8M 左右。

7、用于 GPS 导航系统，水电煤气抄表系统，工业现场采控系统，可以与蓝牙笔记本电脑、电脑加蓝牙适配器、PDA 等设备进行无缝连接。

8 可以对 STC 单片机无线升级和下载程序。

根据实际需求，并且蓝牙遥控为辅助功能，所以选择 HC-06 模块足以满足需求。[10]

4.2.3 超声波传感器

超声波传感器是利用超声波的特性研制而成的传感器。超声波是一种振动频率高于声波的机械波，由换能晶片在电压的激励下发生振动产生的，它具有频率高、波长短、绕射现象小，特别是方向性好、能够成为射线而定向传播等特点。超声波对液体、固体的穿透本领很大，尤其是在阳光不透明的固体中，它可穿透几十米的深度。超声波碰到杂质或分界面会产生显著反射形成反射成回波，碰到活动物体能产生多普勒效应。

探测障碍的最简单的方法是使用超声波传感器，它是利用向目标发射超声波脉冲，计算其往返时间来判定距离的。算法简单，价格合理。

本设计使用 HC-SR04 超声波模块，该模块性能稳定，测度距离精确，模块高精度，盲区小。

HC-SR04 超声波模块的主要技术参数：

1、使用电压：DC---5V 。

2、静态电流：小于 2mA 。

3、电平输出：高 5V。

4、电平输出：底 0V。

5、感应角度：不大于 15 度。

6、探测距离：2cm-450cm。

7、高精度 可达 0.2cm。

本模块的基本工作原理：

（1）采用 IO 口 TRIG 触发测距，给至少 10us 的高电平信号。

（2）模块自动发送 8 个 40khz 的方波，自动检测是否有信号返回。

（3）有信号返回，通过 IO 口 ECHO 输出一个高电平，高电平持续的时间就是超声波从发射到返回的时间。测试距离=(高电平时间*声速(340M/S)) /2。

这个模块使用方法简单，一个控制口发一个 10US 以上的高电平，就可以在接收口等待高电平输出。一有输出就可以开定时器计时，当此口变为低电平读定时器的值，此时就为测距的时间，然后算出距离。由于本设计以红外避障为主，所以这个模块足以满足需求。

4.2.4 红外遥控接收器

红外遥控是利用红外线进行传递信息的一种控制方式，红外遥控具有抗干扰，电路简单，容易编码和解码，功耗小，成本低的优点。

本设计模块使用红外线遥控接收头 HS0038B，这个接收头有三个脚，第 1 脚为信号输出，第 2 脚为电源地，第 3 脚为电源正。HS0038B 是红外接收电路一体化的红外接收装置，这个装置将遥控信号的接收、放大、检波、整形集于一身，并且输出可以让单片机识别的 TTL 信号，这样大大简化了接收电路的复杂程度和电路的设计工作，方便使用。

红外线遥控接收头接收电路工作原理为：

当接收到载波频率为 38KHz 的脉冲调制信号时，HS0038B 内的红外敏感元件将脉冲调制红外光信号转换成电信号，再由前置放大器和自动增益控制电路进行放大处理，然后通过带通滤波器进行滤波，滤波后的信号由解调电路进行解调，最后由输出电路进行反向放大并输出低电平；未接收到载波信号时，电路则输出高电平。

4.2.5 红外对管

红外对管是红外线发射管与光敏接收管，或者红外线接收管，或者红外线接收头配合在一起使用时候的总称。红外对管参数：直径为 3mm，波长为 940nm，工作电压为 1. 2V，工作电流为 20mA，测量距离小于 20cm。波段为红外光，受可见光干扰小。

红外发射管是由红外发光二极管矩组成发光体，用红外辐射效率高的材料（常用砷化

镓）制成 PN 结，正向偏压向 PN 结注入电流激发红外光，其光谱功率分布为中心波长 830～950nm。LED 是英文 Light Emitting Diode 的简称，表现是正温度系数，电流越大温度越高，温度越高电流越大，LED 红外灯的功率和电流大小有关，但正向电流超过最大额定值时，红外灯发射功率反而下降。

红外线接收管具有单向导电性，因此工作时需加上反向电压。无光照时，有很小的饱和反向漏电流（暗电流）。此时光敏管不导通。当光照时，饱和反向漏电流马上增加，形成光电流，在一定的范围内它随入射光强度的变化而增大。并且不受可见光的干扰，感光面积大，灵敏度高，属于光敏二极管，一般只对红外线有反应。

4.3 硬件电路设计

一个单片机应用系统的硬件电路设计包含有两部分内容：一是系统扩展，即单片机内部的功能单元，如 ROM、RAM、I/O 口、定时/记数器、中断系统等不能满足应用系统的要求时，必须在片外进行扩展，选择适当的芯片，设计相应的电路。二是系统的配置，既按照系统功能要求配置外围设备，在本设计中包括电机驱动模块、红外传感器模块、蓝牙模块、超声波模块等，还要设计合适的接口电路。[11]

4.3.1 主控电路

主控电路主要是对各个传感器传输回来的数据进行处理，主控系统将数据处理后，根据数据的情况，决定是否再次调用传感器，或者是直接调用电机驱动模块，通过这些处理实时的控制小车的速度，还有小车的启停状态。小车的主控系统图如图 4-1 所示。[12]

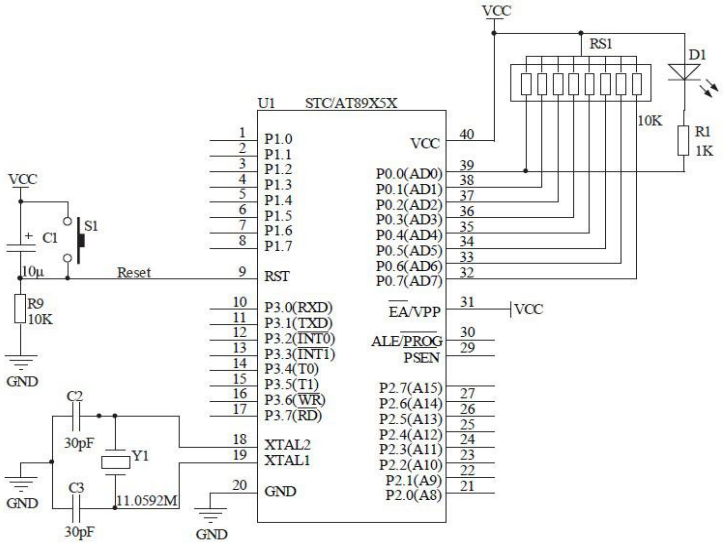


图 4-1 主控系统图

Fig.4-1 The diagram of master Control System

4.3.2 蓝牙遥控电路

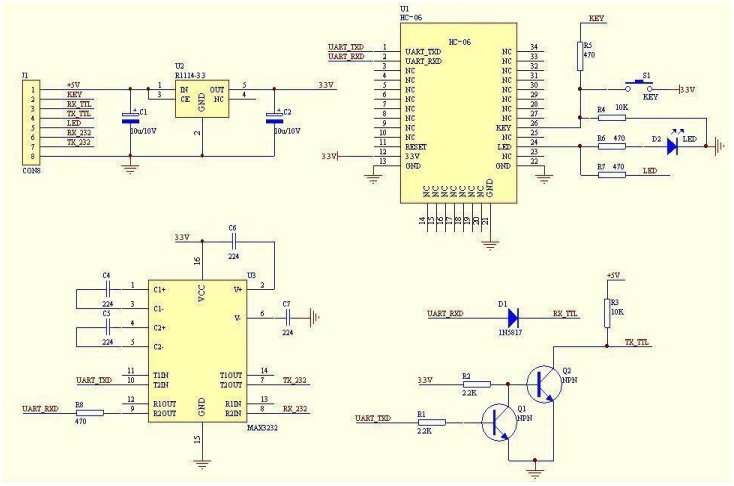


图 4-2 蓝牙控制电路图

Fig.4-2 The diagram of Bluetooth control circuit

蓝牙模块选用 HC-06,蓝牙控制电路图如图 4-2 所示。蓝牙协议定义了 HCIUART、HCIRS232 和 HCIUSB 三种主控制器传输层，故单片机可以通过相应的接口与蓝牙模块相连接。在设计中单片机通过 UART 接口与蓝牙模块连接单片机的 RXD 和 TXD 端分别接蓝牙模块的 TXD 和 RXD 端。至于 CTS 和 RTS 信号端，一般来说由于单片机与蓝牙模块做在同一块 PCB 板上，相距甚短，不接 CTS 和 RTS 信号进行流量控制也可保证数据的可靠传输。若要用则可用单片机的其它 I/O 口来模拟流量控制，主机通过 HCI 传输层将控制命令和数据发送给蓝牙模块，蓝牙模块也通过 HCI 传输层将其状态信息及数据以 HCI 事件的形式发返给主机。

一对简单的蓝牙遥控系统，需要 2 个 BF10 模块，一个模块工作在主模式下，一个模块工作在从模式下。当两模块设置为相同的波特率，相同的通道（不能为通道 64）。上电之后，主从模块则自动连接形成串口透明。此时的数据传输则是全双工的。

（1）设置主模块的 PIO0 为高或悬空，从模块的 PIO0 为低。

（2）设置两个模块的 PIO2、PIO3、PIO4、PIO5 高低到对应的波特率，具体参考设置串口通信波特率。

（3）设置两个模块的 PIO6、PIO7、PIO8、PIO9、PIO10、PIO11 相同的通道，不能为通道 64（即全高电平）。具体参考设置模块通道。

（4）模块上电，主模块则自动去查找该通道的从模块，此时主模块和从模块的 PIO1 脚都是输出为高低脉冲。可以连接一个 LED 进行显示状态。未连接时，LED 灯一直闪烁，

若连接成功之后，LED 灯则一直闪亮，主从模块的 PIO1 管脚输出为高电平。连接成功之后，两个模块两端就能进行串口数据全双工通信了。[13]

4.3.3 电机驱动电路

从单片机输出的信号功率很弱，即使在没有其他外在负载的情况下依旧无法带动电机，所以在实际电路中我们就要加入一块电机驱动芯片来提高输入电机信号的功率，这样也能根据实际的需要来控制电机的转动。电机驱动电路图如图 4-3 所示。

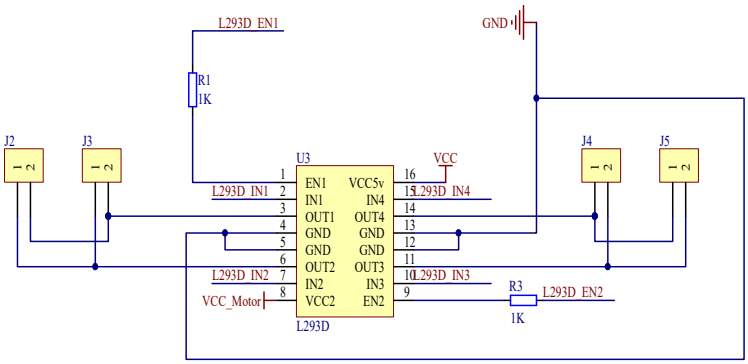


图 4-3 电机驱动电路图

Fig.4-3 The diagram of motor drive circuit

本设计采用 L293D 芯片，L293D 是 ST 公司生产的一种高电压、小电流电机驱动芯片。该芯片采用 16 脚封装，内部是由双极性管组成的 H 桥电路。其输出电流为 100mA，最高电流 2A，最高工作电压 36V，可以驱动感性负载，可以控制电机的正反转，且很容易被单片机控制。用单片机控制晶体管使之工作在占空比可调的开关状态，精确调整电机转速。这种电路由于工作在管子的饱和截止模式下，效率非常高；H 桥电路保证了可以简单地实现转速和方向的控制；电子开关的速度很快，稳定性也很高，是一种广泛采用的调速技术。

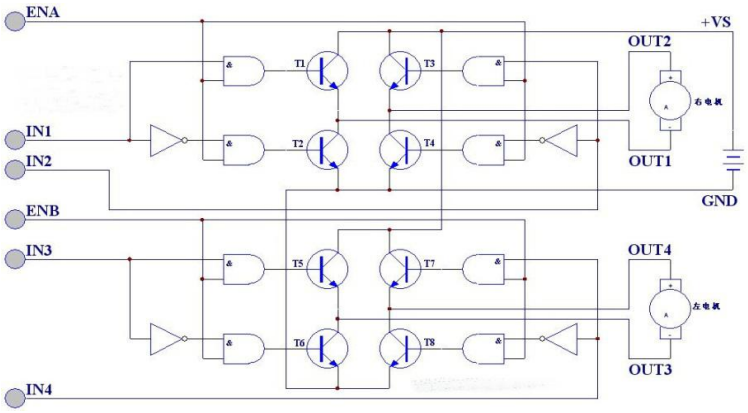


图 4-4 L293D 内部结构图

Fig.4-4 The diagram of L293D internal structure

L293D 的内部结构图如图 4-4 所示，L293D 内置了与门、非门、三级管组成的两组电

路，因为其排列形状像‘H’子母，所以称其为 H 桥路。通过控制三极管的通断就可以是电机旋转起来，而通过控制不同三极管的导通，电流的流向就会发生改变，电机的转向也就会发生变化。在图 4.3 中，使 ENA 与 ENB 两个使能端始终为 1，通过控制 IN1 ~IN4 输入端的状态来改变电机的转向。

表 4-1 L293D 真值表

IN1	IN2	IN3	IN4	右电机	左电机
1	0	1	0	正	正
1	0	0	1	正	反
0	1	1	0	反	正
0	1	0	1	反	反
0	0	0	0	×	×

L293D 真值表如表 4-1 所示。L293D 当 IN1、IN2、IN3 和 IN4 分别为 1010 时，T1、T4、T5 和 T8 导通，左电机和右电机正转；

示例程序 1：void Forward(unsigned char Speed_Right,unsigned char Speed_Left)//前进

```
{
    L293D_IN1=1;
    L293D_IN2=0;
    L293D_IN3=1;
    L293D_IN4=0;
    PWM_Set(255-Speed_Right,255-Speed_Left);
}
```

脉宽调制器本身是一个由运算放大器和几个输入信号组成的电压比较器。运算放大器工作在开换状态，稍微有一点输入信号就可使其输出电压达到饱和值，当输入电压极性改变时，输出电压就在正、负饱和值之间变化，这样就完成了把连续电压变成脉冲电压的转换作用。加在运算放大器反相输入端上的有三个输入信号。一个输入信号是锯齿波调制信号，另一个是控制电压，其极性大小可随时改变，与锯齿波调制信号相减，从而在运算放大器的输出端得到周期不变、脉宽可变的调制输出电压。只要改变控制电压的极性,也就改变了 PWM 变换器输出平均电压的极性,因而改变了电动机的转向.改变控制电压的大小,则调节了输出脉冲电压的宽度,从而调节电动机的转速.只要锯齿波的线性度足够好,输出脉冲的宽度是和控制电压的大小成正比的。

示例程序 2 为脉宽调制函数，其中 PWM_Set 为 PWM 调速函数，本设计采用的是软件调速，Speed_Right、Speed_Left 为用户给定的初值速度，速度值范围：0 ~255,数值越大，速度越快。在通常的程序中多是采用定时器为波特率发生器。而本次设计中采用的是 MCU 自带的 PWM 脉冲发生器，stc12c5a60s2 有内置的一个计数器和比较寄存器 CCAPnL 和 CL，CCAPnL 用来存放一个 0−255 之间的一个数据，CL 是一个计数器，当 CL 的值小于 CCAPnL 时，PWM 引脚输出低电平脉冲，当 CL 的值大于 CCAPnL 时，PWM 引脚输出高电平脉冲

示例程序 2：void PWM_Set(unsigned char PWM0_DATA,unsigned char PWM1_DATA)

```
{
    CCAP0L=PWM0_DATA;//装入比较初值
    CCAP0H=PWM0_DATA;
    CCAP1L=PWM1_DATA; //装入比较初值
    CCAP1H=PWM1_DATA;
}
```

4.3.4 超声波避障电路

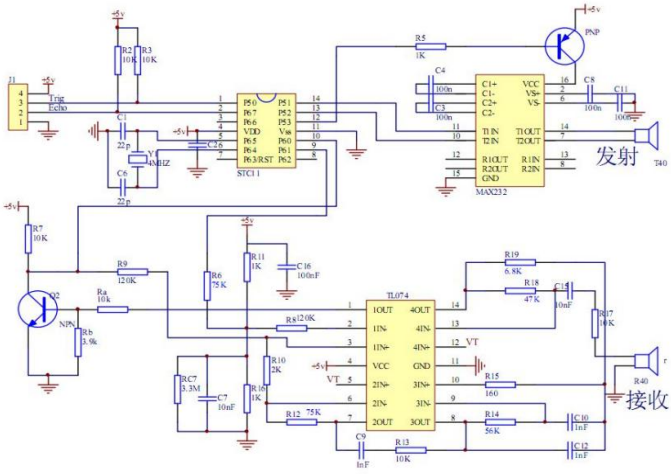


图 4-5 超声波避障电路图

Fig.4-5 The diagram of ultrasonic obstacle avoidance circuit

超声波避障电路图如图 4-5 所示。超声波避障使用 HC-SR04 超声波测距模块，这个模块可提供 2cm-400cm 的非接触式距离感测功能，测距精度可达高到 3mm；模块包括超声波发射器、接收器与控制电路。根据模块测出的距离，判定小车的行进状态。

基本工作原理：

（1）采用 IO 口 TRIG 触发测距，给最少 10us 的高电平信呈。

（2）模块自动发送 8 个 40khz 的方波，自动检测是否有信号返回；

（3）有信号返回，通过 IO 口 ECHO 输出一个高电平，高电平持续的时间就是超声波从发射到返回的时间。测试距离=(高电平时间*声速(340M/S))/2；

4.3.5 红外遥控电路

红外遥控电路主要有两部分，一部分是红外发射器，也就是红外遥控器，另一部分是小车上的红外接收器。根据红外传送协议，只需对操作码进行编码操作，这样按下不同的键位，就会产生不同的操作码，接收端根据接收到不同的操作码进行不同的操作。

4.3.6 红外避障电路

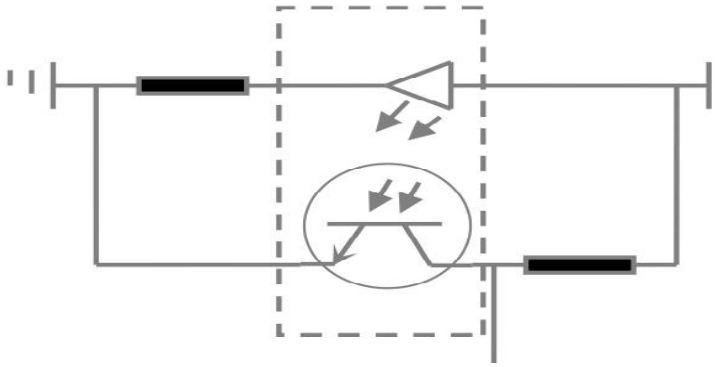


图 4-6 红外避障电路图

Fig.4-6 The diagram of infrared obstacle avoidance circuit

红外避障电路图如图 4-6 所示。红外避障模块由红外对管构成，小车根据处理红外对管传输回来的数据，对小车的行进状态进行控制。同时辅以超声波避障功能，进一步优化小车的避障功能。

4.3.7 红外循迹电路

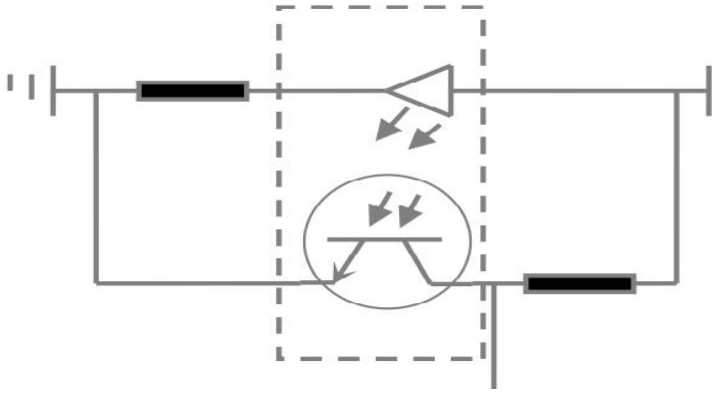


图 4-7 红外循迹电路图

Fig.4-7 The diagram of infrared tracking circuit

红外循迹电路图如图 4-7 所示。红外循迹模块由红外对管构成，红外接收管接收的红外信号强弱是影响小车行进状态的关键，红外接收器接收信号后，将信号传输给控制系统，控制系统根据处理红外对管传输回来数据的结果，对小车的行进状态进行实时控制。

5 软件设计

在进行微机控制系统设计时，除了系统硬件设计外，大量的工作就是如何根据每个生产对象的实际需要设计应用程序。所以软件设计在微机控制系统设计中占重要地位。对于本系统，软件更为重要。在单片机控制系统中，大体上可分为数据处理、过程控制两个基本类型。数据处理包括：数据的采集、数字滤波、标度变换等。过程控制程序主要是使单片机按一定的方法进行计算，然后再输出，以便控制生产。在进行软件设计时，通常把整个过程分成若干个部分，每一部分叫做一个模块。所谓“模块”，实质上就是所完成一定功能，相对独立的程序段，这种程序设计方法叫模块程序设计法。

模块程序设计法的主要优点是：

1.单个模块比起一个完整的程序易编写及调试；

2.模块可以共存，一个模块可以被多个任务在不同条件下调用；

3.模块程序允许设计者分割任务和利用已有程序，为设计者提供方便。

5.1 软件设计原则

软件的设计要满足是个基本要求：

1. 可靠性。用软件系统规模越做越大越复杂，其可靠性越来越难保证。应用本身对系统运行的可靠性要求越来越高，软件系统的可靠性也直接关系到设计自身的声誉和生存发展竞争能力。软件可靠性意 味着该软件在测试运行过程中避免可能发生故障的能力，且一旦发生故障后，具有解脱和排除故障的能力。软件可靠性和硬件可靠性本质区别在于：后者为物理机理的衰变和老化所致，而前者是由于设计和实现的错误所致。故软件的可靠性必须在设计阶段就确定，在生产和测试阶段再考虑就困难了。

2. 健壮性。健壮性又称鲁棒性，是指软件对于规范要求以外的输入能够判断出这个输入不符合规范要求，并能有合理的处理方式。软件健壮性是比较模糊的概念，但是却是非常重要的软件外部量度标准。软件设计的健壮与否直接反应分析设计和编码人员的水平。

3. 可修改性。要求以科学的方法设计软件，使之有良好的结构和完备的文档，系统性能易于调整。

4. 容易理解，程序简便。软件的可理解性是其可靠性和可修改性的前提。它并不仅仅是文档清晰可读的问题，更要求软件本身具有简单明了的结构。这在很大程度上取决于设计者的洞察力和创造性，以及对设计对象掌握的程度。

6. 可测试性。可测试性就是设计一个适当的数据集合，用来测试所建立的系统。

7. 效率性。软件的效率性一般用程序的执行时间和所占用的内存容量来度量。在达到原理要求功能指标的前提下，程序运行所需时间愈短和占用存储容量愈小，则效率愈高。

8. 标准化原则。在结构上开放，基于业界开放式标准，符合国家和信息产业部规范。

9. 先进性。满足客户需求，系统性能可靠，易于维护。

10. 可扩展性。软件设计完要留有升级接口和升级空间。对扩展开放，对修改关闭。

5.2 主程序设计

主程序是整个程序设计的主体，也是整个系统中最重要的环节，它负责各个子程序模块的执行顺序、时序以及它们之间的关系。[14]主程序通过系统的自检以及调用各种子程序模块，从而实现系统的初始化，进行数据显示、数据处理、按键处理、参数传递、产生控制信号等功能。主程序流程图如图 5-1 所示。[15][16]

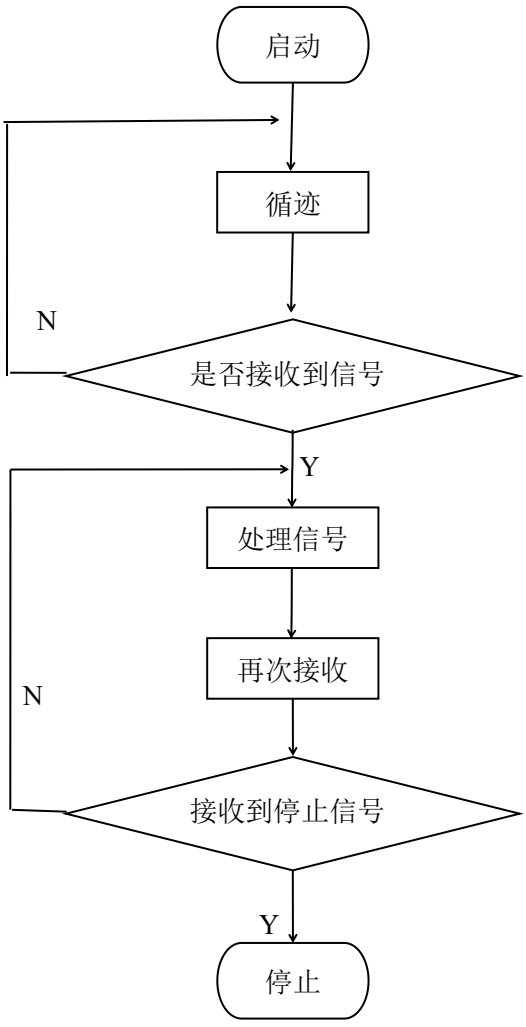


图 5-1 主程序流程图

Fig.5-1 The flow chart of main program

6 系统测试与运行

6.1 测试方案

测试硬件是否满足需求，是否摸个功能或者模块有问题，针对性的对相应的功能点进行测试。

6.1.1 循迹测试方案

针对小车的智能循迹功能，设计一个黑色的环形跑道，跑道可以由简单向复杂过渡，通过不停地增加难度，测试小车是否可以满足需求并达到预期效果。

图 6-1 简单测试场景图

Fig.6-1 The diagram of simple test

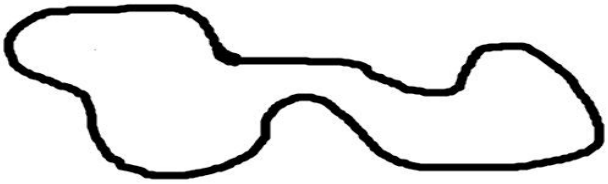


图 6-2 正常测试场景图

Fig.6-2 The diagram of normal test

图 6-3 复杂测试场景图

Fig.6-3 The diagram of complex test

三个测试场景图如图 6-1、图 6-2、图 6-3 所示。对小车的循迹功能进行测试，在测试过程中，采取顺时针和逆时针两个方向进行测试，以此来检测智能小车的左转和右转，在两种情况下，小车的测试结果是否有差异。每个测试场景经过顺时针、逆时针各 2 次，如果是封闭图进行 2 圈的实际测试，记录测试数据结果。

6.1.2 红外避障测试方案

针对小车的红外避障功能，设计相应测试场景，检验在不同情况下，小车的红外避障功能是否有问题，检验红外避障功能是否能满足设计的需求，能否达到设计的预期效果，记录测试的数据。

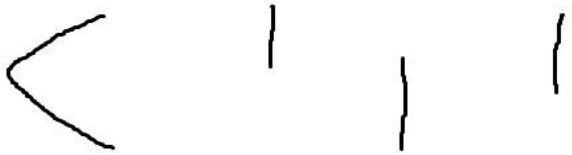


图 6-4 红外避障测试场景

Fig.6-4 The diagram of Infrared obstacle avoidance test

测试场景图如图 6-4 所示。对小车的红外避障功能进行测试，在这个场景进行五次测试，记录测试数据结果。

6.1.3 超声波避障测试方案

针对小车的超声波避障功能，由于与红外避障的功能相差不是很大，可以使用类似的场景进行测试，由于超声波传感器的功能更强大，可以对超声波模块进行更多和更复杂的测试。

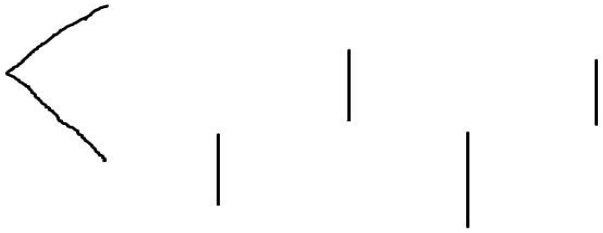


图 6-5 超声波避障测试场景

Fig.6-5 The diagram of Ultrasonic obstacle avoidance test

测试场景图如图 6-5 所示。对小车的超声波避障功能进行测试，在这个场景进行五次测试，记录测试数据结果。

6.1.4 红外遥控测试方案

对于智能小车的红外遥控测试，在任何场景下都可以进行，不需要特殊设定，但需要在距离上与角度对小车进行多次测试。检验在不同的情况下，红外遥控是否能满足智能小车设计需求。

6.1.5 蓝牙遥控测试方案

对于智能小车的蓝牙遥控测试，同在样任何场景下都可以进行，也不需要特殊设定，但需要在距离上与角度对小车也需要进行多次测试。检验蓝牙遥控是否可以满足智能小车设计需求。

6.2 测试结果与分析

6.2.1 循迹测试结果与分析

按照预定的测试方案，每次小车运行结束后，记录小车每次测试的结果，还有每次测试小车运行的时间，通过这些条件对比，观察小车的循迹是否有问题，将 6 次测试的结果制表。

表 6-1 循迹测试结果表

Table.6-1 The table of tracking test results

测试编号	运动方向	完成效果	执行时间	测试场景
1	顺时针	完成 2 圈	1 分 48 秒	简单
2	逆时针	完成 2 圈	2 分 01 秒	简单
3	顺时针	完成 2 圈	2 分 33 秒	正常
4	逆时针	完成 2 圈	2 分 51 秒	正常
5	顺时针	完成 2 圈	3 分 03 秒	复杂
6	逆时针	完成 2 圈	3 分 33 秒	复杂

智能小车循迹测试结果表如表 6-1 所示。通过表可以发现小车在简单场景顺时针运动时，主要执行右转和直行这两种运动控制；在逆时针运动时，主要执行左转和直行这两种运动控制。在这组测试过程中，逆时针的运动时间相对顺时针的运动时间较长，则可大致得出结论：左转控制的响应时间慢。

其他两种情况下的测试结果，由于场景比较复杂设计行进状态改变情况较多，测试时间意义不大，只要小车正常循迹完成运行，可以确认小车循迹模块功能正常，每次测试时由于电池电压不同，重复这个测试方案的结果或许会有不同，每轮测试的时间会有差异。

6.2.2 红外避障测试结果与分析

按照预定的测试方案，每次小车运行结束后，记录小车每次测试的结果，通过这些条件对比，观察小车的红外避障是否有问题，将 4 次测试的结果制表。

表 6-2 红外避障测试结果表

Table.6-2 The table of infrared obstacle avoidance test results

测试编号	完成效果	执行时间
1	顺利	1 分 48 秒
2	不顺利	
3	顺利	1 分 51 秒
4	不顺利	

智能小车红外避障测试结果表如表 6-2 所示。通过表可以发现，小车在这个场景运动时，通过处理红外接收回来的信号，来改变小车的运动轨迹，主要执行右转和直行这两种运动控制，但是实际测试中，小车的只要转向的角度大一点，就达不到预期的效果，小车的轨迹就变得不可预测，而且由于传感器的误差问题，对智能小车的影响也非常大，由于每次转向后，再次处理红外接收器接收回来的信号，响应时间也很长。对于转角问题只能尽量修改，经过反复试验后，决定将其变为辅助的避障功能，以后在做扩展研究。

6.2.3 超声波避障测试结果与分析

按照预定的测试方案，每次小车运行结束后，记录小车每次测试的结果，通过这些条件对比，观察小车的超声波避障是否有问题，将 5 次测试的结果制表。

表 6-3 超声波避障测试结果表

Table.6-3 The table of ultrasonic obstacle avoidance test results

测试编号	完成效果	执行时间
1	顺利	1 分 48 秒
2	顺利	1 分 42 秒
3	不顺利	
4	顺利	1 分 51 秒
5	顺利	1 分 54 秒

智能小车超声波避障测试结果表如表 6-3 所示。通过表可以发现小车在这个场景运动时，通过处理超声波接收器接收回来的信号，改变小车的运动轨迹，主要执行右转和直行这两种运动控制，在实际测试中，小车的转向的角度对小车的运动轨迹有着绝对的影响，只要稍有偏差，小车的轨迹就不可预测了，由于超声波传感器的优势，相对于红外避障的预期效果要好很多，而且改进空间很大，可以做更多的优化。

6.2.4 红外遥控测试结果与分析

对于智能小车的红外遥控测试结果，只要可以看到小车的红外接收器，就可以智能的遥控小车，控制小车的行进，但是超过一定范围时，就不可以遥控，或是遥控器的发射受到阻挡，同样不可遥控，根据红外接收传感器的功能上分析，由于传感器接收的信号是红外线，某些原因导致无法接收信号，例如遥控角度，是否有障碍物阻挡等等，但这些是可接收的。所以通过测试的结果来看，智能小车的红外遥控功能，可以达到了设计的要求。

6.2.5 蓝牙遥控测试结果与分析

对于智能小车的了蓝牙遥控测试结果，只要是十米范围内，就可以随意的遥控小车，控制小车的行进，但是超过这个范围时，就不可以遥控，其中即使有障碍物阻挡一样可以遥控小车，根据蓝牙接收模块的功能上分析，由于传输的方式和协议不同，蓝牙传输唯一的影响就是范围，其他的情况对它的干扰基本没有太大影响。所以通过测试的结果来看，智能小车的红外遥控功能，可以达到了设计的要求。

结论

本次毕业设计，对于 STC89C52 单片机，电路的连接，软硬件的结合，以及功能模块的集合都有了很多认识，本次毕设从设计开始到调试完毕，遇到了很多问题，有些问题经过老师指导，或者是自己查询资料得以解决，但有些由于硬件的问题和自己知识有限，导致无法解决，只能将其变为辅助功能，或者直接去掉该功能，例如同时使用两种避障功能，或者同时使用两种遥控功能，在软件上需要使用很多次中断嵌套，对于这些功能只能修改为辅助。设计开始时只想着添加功能，让小车成为一个拥有多功能的智能小车，对于后期的模块结合没有仔细考虑，导致后期模块结合时功能冲突，并且无法解决有效问题。小车零件采集的时候，由于想到一个功能就买一部分，导致小车成本飙升。小车硬件的焊接由于技术不纯熟，导致第一次整个板子都被废掉了，还有对于硬件电路的不了解，有些电路被焊接反了，只能重新采购，重新进行焊接。但是经过多次的失败和努力终于做成本次毕设多功能的智能小车。通过本次设计，对于一个产品从设计、开发、测试，到最终成型的过程有了重新的认识，对于整体的考虑也更加明确，而不是把所有的问题都等到后期才解决。本次毕设对于我由非常大的提升，是四年大学所学习知识的总结，是将理论转化成实践最好的历练，是自己以后实际生产中最重要的经验。

致谢

经过大半年的毕业设计，真是收获了很多，对嵌入式的硬件也深入的了解一些，本次毕业设计从开始到结束，我的指导老师、公司的同事，以及身边的同学都无时无刻给予我极大的帮助，本次毕设单单靠我一个人是无法完成的，所以我要感谢很多人，我的母校、我的学院、我的指导老师、我的父母、我的同事和我的同学。最先感谢我的母校“辽宁工程技术大学”，感谢母校给了我这个机会，让我可以收获知识，还有机会好好锻炼和展现自己，再感谢的是我的学院“软件学院”，感谢学院的老师教授我知识，可以让我谋生的知识，真是要由衷的感谢这些老师，然后是我的指导老师刘万军老师，感谢刘老师给我的指导，让我顺利解决一些开发中的问题，帮助我顺利完成毕业设计，还有我的父母，感谢他们养育了我，感谢他们给了我受教育的机会，最后要感谢的是在毕设过程中帮助我的师父，大连华信公司指导我的师父，还有我身边的同学，没有他们给我的帮助，就无法完成我的毕设，在此，再次感谢这些帮助过我的老师、父母、师父、同学，还有给我提供平台的母校。

参考文献

[1] 靳桅. 单片机原理及应用[M]. 西南交通大学出版社，2012.

[2] 李建中. 单片机原理及应用[M]西安电子科技大学出版社，2013.

[3] 谢维成，杨加国. 单片机原理与应用及 C51 程序设计[M]. 北京：清华大学出版社，2012.

[4] 杨帮文. 新编传感器实用宝典[M]. 机械工业出版社，2013.

[5] 金发庆. 传感器技术与应用[M]. 北京:机械工业出版社，2014.

[6] 肖景和. 红外线热释电与超声波遥控电路[M]. 人民邮电出版社，2011.

[7] 刘玉宾，朱焕立. 单片机与接口技术[M]. 北京：机械工业出版社，2010.

[8] 范晶彦. 传感器与检测技术应用[M]. 北京:机械工业出版社，2009.

[9] YD. Tulone. Is it possible to ensure strong data guarantees in highly mobile

[10] 王俊峰，孟令启. 现代传感器应用技术[M]. 北京:机械工业出版社，2013.

[11] 谢自美. 电子线路设计. 试验. 测试[M]. 华中科技大学出版社，2009.

[12] 沙占友，王彦朋，孟志永. 单片机外围电路设计[M].北京：电子工业出版社，2011.

[13] 朱清慧，张凤蕊，翟天嵩，王志奎. Proteus 教程——电子线路设计、制作与仿真[M]. 北京：清华大学出版社，2008.

[14] 周坚编. 单片机 C 语言轻松入门[M]. 北京航空航天大学出版社，2010.

[15] 周航慈. 单片机应用程序设计技术[M]. 北京：北京航空航大大学出版社，2008.

[16] 夏继强. 单片机实验与实践教程[M]. 北京：北京航空航天大学出版社，2009.

[16] 张睿. 关于单片机的直流电机控制探讨[M]. 河南：河海大学文天学院 ，2016.

[17] 周杨. 基于单片机 2.4GHz 遥控车设计[M]. 河南：电脑知识与技术:学术交流，2016.

附录 A 中文译文

单片机

单片机也被称为微控制器（Microcontroller Unit），常用英文字母的缩写 MCU 表示单片机，它最早是被用在工业控制领域。单片机由芯片内仅有 CPU 的专用处理器发展而来。最早的设计理念是通过将大量外围设备和 CPU 集成在一个芯片中，使计算机系统更小，更容易集成进复杂的而对体积要求严格的控制设备当中。INTEL 的 Z80 是最早按照这种思想设计出的处理器，从此以后，单片机和专用处理器的发展便分道扬镳。

早期的单片机都是 8 位或 4 位的。其中最成功的是 INTEL 的 8031，因为简单可靠而性能不错获得了很大的好评。此后在 8031 上发展出了 MCS51 系列单片机系统。基于这一系统的单片机系统直到现在还在广泛使用。随着工业控制领域要求的提高，开始出现了 16 位单片机，但因为性价比不理想并未得到很广泛的应用。90 年代后随着消费电子产品大发展，单片机技术得到了巨大提高。随着 INTEL i960 系列特别是后来的 ARM 系列的广泛应用，32 位单片机迅速取代 16 位单片机的高端地位，并且进入主流市场。而传统的 8 位单片机的性能也得到了飞速提高，处理能力比起 80 年代提高了数百倍。目前，高端的 32 位单片机主频已经超过 300MHz，性能直追 90 年代中期的专用处理器，而普通的型号出厂价格跌落至 1 美元，最高端[1]的型号也只有 10 美元。当代单片机系统已经不再只在裸机环境下开发和使用，大量专用的嵌入式操作系统被广泛应用在全系列的单片机上。而在作为掌上电脑和手机核心处理的高端单片机甚至可以直接使用专用的 Windows 和 Linux 操作系统。

单片机比专用处理器更适合应用于嵌入式系统，因此它得到了最多的应用。事实上单片机是世界上数量最多的计算机。现代人类生活中所用的几乎每件电子和机械产品中都会集成有单片机。手机、电话、计算器、家用电器、电子玩具、掌上电脑以及鼠标等电脑配件中都配有 1-2 部单片机。而个人电脑中也会有为数不少的单片机在工作。汽车上一般配备 40 多部单片机，复杂的工业控制系统上甚至可能有数百台单片机在同时工作！单片机的数量不仅远超过 PC 机和其他计算的总和，甚至比人类的数量还要多。

单片机又称单片微控制器,它不是完成某一个逻辑功能的芯片,而是把一个计算机系统集成到一个芯片上。相当于一个微型的计算机，和计算机相比，单片机只缺少了 I/O 设备。概括的讲：一块芯片就成了一台计算机。它的体积小、质量轻、价格便宜、为学习、应用和开发提供了便利条件。同时，学习使用单片机是了解计算机原理与结构的最佳选择。

单片机内部也用和电脑功能类似的模块，比如 CPU，内存，并行总线，还有和硬盘作用相同的存储器件，不同的是它的这些部件性能都相对我们的家用电脑弱很多，不过价钱也是低的，一般不超过 10 元即可……用它来做一些控制电器一类不是很复杂的工作足矣了。我们现在用的全自动滚筒洗衣机、排烟罩、VCD 等等的家电里面都可以看到它的身影！……它主要是作为控制部分的核心部件。

它是一种在线式实时控制计算机，在线式就是现场控制，需要的是有较强的抗干扰能力，较低的成本，这也是和离线式计算机的（比如家用 PC）的主要区别。

单片机芯片

单片机是靠程序运行的，并且可以修改。通过不同的程序实现不同的功能，尤其是特殊的独特的一些功能，这是别的器件需要费很大力气才能做到的，有些则是花大力气也很难做到的。一个不是很复杂的功能要是用美国 50 年代开发的 74 系列，或者 60 年代的 CD4000 系列这些纯硬件来搞定的话，电路一定是一块大 PCB 板！但是如果要是用美国 70 年代成功投放市场的系列单片机，结果就会有天壤之别！只因为单片机的通过你编写的程序可以实现高智能，高效率，以及高可靠性！

由于单片机对成本是敏感的，所以目前占统治地位的软件还是最低级汇编语言，它是除了二进制机器码以上最低级的语言了，既然这么低级为什么还要用呢？很多高级的语言已经达到了可视化编程的水平为什么不用呢？原因很简单，就是单片机没有家用计算机那样的 CPU，也没有像硬盘那样的海量存储设备。一个可视化高级语言编写的小程序里面即使只有一个按钮，也会达到几十 K 的尺寸！对于家用 PC 的硬盘来讲没什么，可是对于单片机来讲是不能接受的。 单片机在硬件资源方面的利用率必须很高才行，所以汇编虽然原始却还是在大量使用。一样的道理，如果把巨型计算机上的操作系统和应用软件拿到家用 PC 上来运行，家用 PC 的也是承受不了的。

可以说，二十世纪跨越了三个“电”的时代，即电气时代、电子时代和现已进入的电脑时代。不过，这种电脑，通常是指个人计算机，简称 PC 机。它由主机、键盘、显示器等组成。还有一类计算机，大多数人却不怎么熟悉。这种计算机就是把智能赋予各种机械的单片机（亦称微控制器）。顾名思义，这种计算机的最小系统只用了一片集成电路，即可进行简单运算和控制。因为它体积小，通常都藏在被控机械的“肚子”里。它在整个装置中，起着有如人类头脑的作用，它出了毛病，整个装置就瘫痪了。现在，这种单片机的使用领域已十分广泛，如智能仪表、实时工控、通讯设备、导航系统、家用电器等。各种产品一

旦用上了单片机，就能起到使产品升级换代的功效，常在产品名称前冠以形容词——“智能型”，如智能型洗衣机等。现在有些工厂的技术人员或其它业余电子开发者搞出来的某些产品，不是电路太复杂，就是功能太简单且极易被仿制。究其原因，可能就卡在产品未使用单片机或其它可编程逻辑器件上。

单片机历史

单片机诞生于 20 世纪 70 年代末，经历了 SCM、MCU、SoC 三大阶段。

起初模型

1.SCM 即单片微型计算机（Single Chip Microcomputer）阶段，主要是寻求最佳的单片形态嵌入式系统的最佳体系结构。“创新模式”获得成功，奠定了 SCM 与通用计算机完全不同的发展道路。在开创嵌入式系统独立发展道路上，Intel 公司功不可没。

2.MCU 即微控制器（Micro Controller Unit）阶段，主要的技术发展方向是：不断扩展满足嵌入式应用时，对象系统要求的各种外围电路与接口电路，突显其对象的智能化控制能力。它所涉及的领域都与对象系统相关，因此，发展 MCU 的重任不可避免地落在电气、电子技术厂家。从这一角度来看，Intel 逐渐淡出 MCU 的发展也有其客观因素。在发展 MCU 方面，最著名的厂家当数 Philips 公司。

Philips 公司以其在嵌入式应用方面的巨大优势，将 MCS-51 从单片微型计算机迅速发展到微控制器。因此，当我们回顾嵌入式系统发展道路时，不要忘记 Intel 和 Philips 的历史功绩。

嵌入式系统

单片机是嵌入式系统的独立发展之路，向 MCU 阶段发展的重要因素，就是寻求应用系统在芯片上的最大化解决；因此，专用单片机的发展自然形成了 SOC 化趋势。随着微电子技术、IC 设计、EDA 工具的发展，基于 SOC 的单片机应用系统设计会有较大的发展。因此，对单片机的理解可以从单片微型计算机、单片微控制器延伸到单片应用系统。

单片机的应用领域

目前单片机渗透到我们生活的各个领域，几乎很难找到哪个领域没有单片机的踪迹。导弹的导航装置，飞机上各种仪表的控制，计算机的网络通讯与数据传输，工业自动化过程的实时控制和数据处理，广泛使用的各种智能 IC 卡，民用豪华轿车的安全保障系统，录像机、摄像机、全自动洗衣机的控制，以及程控玩具、电子宠物等等，这些都离不开单

片机。更不用说自动控制领域的机器人、智能仪表、医疗器械了。因此，单片机的学习、开发与应用将造就一批计算机应用与智能化控制的科学家、工程师。

单片机广泛应用于仪器仪表、家用电器、医用设备、航空航天、专用设备的智能化管理及过程控制等领域，大致可分如下几个范畴：

1.在智能仪器仪表上的应用

单片机具有体积小、功耗低、控制功能强、扩展灵活、微型化和使用方便等优点，广泛应用于仪器仪表中，结合不同类型的传感器，可实现诸如电压、功率、频率、湿度、温度、流量、速度、厚度、角度、长度、硬度、元素、压力等物理量的测量。采用单片机控制使得仪器仪表数字化、智能化、微型化，且功能比起采用电子或数字电路更加强大。例如精密的测量设备（功率计，示波器，各种分析仪）。

2.在工业控制中的应用

用单片机可以构成形式多样的控制系统、数据采集系统。例如工厂流水线的智能化管

3.在家用电器中的应用

可以这样说，现在的家用电器基本上都采用了单片机控制，从电饭褒、洗衣机、电冰箱、空调机、彩电、其他音响视频器材、再到电子秤量设备，五花八门，无所不在。

4.在计算机网络和通信领域中的应用

现代的单片机普遍具备通信接口，可以很方便地与计算机进行数据通信，为在计算机网络和通信设备间的应用提供了极好的物质条件，现在的通信设备基本上都实现了单片机智能控制，从手机，电话机、小型程控交换机、楼宇自动通信呼叫系统、列车无线通信、再到日常工作中随处可见的移动电话，集群移动通信，无线电对讲机等。

5.单片机在医用设备领域中的应用

单片机在医用设备中的用途亦相当广泛，例如医用呼吸机，各种分析仪，监护仪，超声诊断设备及病床呼叫系统等等。

6.在各种大型电器中的模块化应用

某些专用单片机设计用于实现特定功能，从而在各种电路中进行模块化应用，而不要求使用人员了解其内部结构。如音乐集成单片机，看似简单的功能，微缩在纯电子芯片中（有别于磁带机的原理），就需要复杂的类似于计算机的原理。如：音乐信号以数字的形式存于存储器中（类似于 ROM），由微控制器读出，转化为模拟音乐电信号（类似于声卡）。

在大型电路中，这种模块化应用极大地缩小了体积，简化了电路，降低了损坏、错误率，也方便于更换。

7.单片机在汽车设备领域中的应用

单片机在汽车电子中的应用非常广泛，例如汽车中的发动机控制器，基于 CAN 总线的汽车发动机智能电子控制器，GPS 导航系统，abs 防抱死系统，制动系统等等。

此外，单片机在工商，金融，科研、教育，国防航空航天等领域都有着十分广泛的用途。

学习应用六大重要部分

单片机学习应用的六大重要部分

一、总线：

我们知道，一个电路总是由元器件通过电线连接而成的，在模拟电路中，连线并不成为一个问题，因为各器件间一般是串行关系，各器件之间的连线并不很多，但计算机电路却不一样，它是以微处理器为核心，各器件都要与微处理器相连，各器件之间的工作必须相互协调，所以需要的连线就很多了，如果仍如同模拟电路一样，在各微处理器和各器件间单独连线，则线的数量将多得惊人，所以在微处理机中引入了总线的概念，各个器件共同享用连线，所有器件的 8 根数据线全部接到 8 根公用的线上，即相当于各个器件并联起来，但仅这样还不行，如果有两个器件同时送出数据，一个为 0，一个为 1，那么，接收方接收到的究竟是什么呢？这种情况是不允许的，所以要通过控制线进行控制，使器件分时工作，任何时候只能有一个器件发送数据（可以有多个器件同时接收）。器件的数据线也就被称为数据总线，器件所有的控制线被称为控制总线。在单片机内部或者外部存储器及其它器件中有存储单元，这些存储单元要被分配地址，才能使用，分配地址当然也是以电信号的形式给出的，由于存储单元比较多，所以，用于地址分配的线也较多，这些线被称为地址总线。

二、数据、地址、指令：

之所以将这三者放在一起，是因为这三者的本质都是一样的——数字，或者说都是一串‘0’和‘1’组成的序列。换言之，地址、指令也都是数据。指令：由单片机芯片的设计者规定的一种数字，它与我们常用的指令助记符有着严格的一一对应关系，不可以由单片机的开发者更改。地址：是寻找单片机内部、外部的存储单元、输入输出口的依据，内部单元

的地址值已由芯片设计者规定好，不可更改，外部的单元可以由单片机开发者自行决定，但有一些地址单元是一定要有的（详见程序的执行过程）。

三、P0 口、P2 口和 P3 的第二功能用法：

初学时往往对 P0 口、P2 口和 P3 口的第二功能用法迷惑不解，认为第二功能和原功能之间要有一个切换的过程，或者说要有一条指令，事实上，各端口的第二功能完全是自动的，不需要用指令来转换。如 P3.6、P3.7 分别是 WR、RD 信号，当微片理机外接 RAM 或有外部 I/O 口时，它们被用作第二功能，不能作为通用 I/O 口使用，只要一微处理机一执行到 MOVX 指令，就会有相应的信号从 P3.6 或 P3.7 送出，不需要事先用指令说明。事实上‘不能作为通用 I/O 口使用’也并不是‘不能’而是（使用者）‘不会’将其作为通用 I/O 口使用。你完全可以在指令中按排一条 SETB P3.7 的指令，并且当单片机执行到这条指令时，也会使 P3.7 变为高电平，但使用者不会这么去做，因为这通常会导致系统的崩溃。

四、程序的执行过程：

单片机在通电复位后 8051 内的程序计数器（PC）中的值为‘0000’，所以程序总是从‘0000’单元开始执行，也就是说：在系统的 ROM 中一定要存在‘0000’这个单元，并且在‘0000’单元中存放的一定是一条指令。

五、堆栈：

堆栈是一个区域，是用来存放数据的，这个区域本身没有任何特殊之处，就是内部 RAM 的一部份，特殊的是它存放和取用数据的方式，即所谓的‘先进后出，后进先出’，并且堆栈有特殊的数据传输指令，即‘PUSH’和‘POP’，有一个特殊的专为其服务的单元，即堆栈指针 SP，每当执一次 PUSH 指令时，SP 就（在原来值的基础上）自动加 1，每当执行一次 POP 指令，SP 就（在原来值的基础上）自动减 1。由于 SP 中的值可以用指令加以改变，所以只要在程序开始阶段更改了 SP 的值，就可以把堆栈设置在规定的内存单元中，如在程序开始时，用一条 MOV SP，#5FH 指令，就时把堆栈设置在从内存单元 60H 开始的单元中。一般程序的开头总有这么一条设置堆栈指针的指令，因为开机时，SP 的初始值为 07H，这样就使堆栈从 08H 单元开始往后，而 08H 到 1FH 这个区域正是 8031 的第二、三、四工作寄存器区，经常要被使用，这会造成数据的混乱。不同作者编写程序时，初始化堆栈指令也不完全相同，这是作者的习惯问题。当设置好堆栈区后，并不意味着该区域成为一种专用内存，它还是可以象普通内存区域一样使用，只是一般情况下编程者不会把它当成普通内存用了。

附录 B 英文原文

Single-chip

SCM is also known as micro-controller (Microcontroller Unit), commonly used letters of the acronym MCU that it was first used in industrial control.

Only a single chip by the CPU chip developed from a dedicated processor. The first design is by a large number of peripherals and CPU on a chip in the computer system, smaller, more easily integrated into a complex and demanding on the volume control device which. INTEL'sZ80 is the first designed in accordance with this idea processor, then on the development of microcontroller and dedicated processors have parted ways.

Are 8-bit microcontroller early or4 bits. One of the most successful is the INTEL 8031, for a simple, reliable and good performance was a lot of praise. Then developed in 8031 out of MCS51 MCU Systems. SCM systems based on this system until now is still widely used. With the increased requirements of industrial control field, began a 16-bit microcontroller, because the cost is not satisfactory but have not been very widely used. After 90 years with the great development of consumer electronics, microcontroller technology has been a huge increase. With INTEL i960 series, especially the later series of widely used ARM, 32-bit microcontroller quickly replace high-end 16-bit MCU status and enter the mainstream market. The traditional 8-bit microcontroller performance have been the rapid increase capacity increase compared to 80 the number of times. Currently, high-end 32-bit microcontroller clocked over 300MHz, the performance catching the mid-90's dedicated processor, while the average model prices fall to one U.S. dollars, the most high-end [1] model only 10 dollars.Modern SCM systems are no longer only in the development and use of bare metal environment, a large number of proprietary embedded operating system is widely used in the full range of SCM. The handheld computers and cell phones as the core processing of high-end microcontroller can even use a dedicated Windows and Linux operating systems.

SCM is more suitable than the specific processor used in embedded systems, so it was up to the application. In fact the number of SCM is the world's largest computer. Modern human life used in almost every piece of electronic and mechanical products will be integrated single chip.Phone, telephone, calculator, home appliances, electronic toys, handheld computers and computer accessories such as a mouse with a 1-2 in both the Department of SCM. Personal computer will have a large number of SCM in the work. General car with more than 40 SCM, complex industrial control systems may even have hundreds of SCM in the same time work! SCM is not only far exceeds the number of PC and other computing the sum, or even more than the number of human beings

Single chip, also known as single-chip microcontroller, it is not complete a certain logic

chips, but to a computer system integrated into a chip.Equivalent to a micro-computer, and computer than just the lack of a microcontroller I / O devices. General talk: a chip becomes a computer. Its small size, light weight, cheap, for the study, application and development of facilities provided. At the same time, learning to use the MCU is to understand the principle and structure of the computer the best choice.

SCM and the computer functions internally with similar modules, such as CPU, memory, parallel bus, the same effect as well, and hard disk memory devices, and different is its performance of these components were relatively weak many of our home computer, but the price is low , usually not more than 10 yuan you can do with it ..... some control for a class is not very complicated electrical work is enough of. We are using automatic drum washing machine, smoke hood, VCD and so on appliances which could see its shadow! ..... It is primarily as a control section of the core components

It is an online real-time control computer, control-line is that the scene is needed is a stronger anti-jamming ability, low cost, and this is, and off-line computer (such as home PC), the main difference.

Single chip

MCU is through running, and can be modified. Through different procedures to achieve different functions, in particular special unique features, this is another device much effort needs to be done, some great efforts are very difficult to do. A not very complex functions if the 50's with the United States developed 74 series, or the 60's CD4000 series of these pure hardware buttoned, then the circuit must be a large PCB board! But if the United States if the 70's with a series of successful SCM market, the result will be a drastic change! Just because you are prepared by microcomputer programs can achieve high intelligence, high efficiency and high reliability!

As the microcontroller on the cost-sensitive, so now the dominant software or the lowest level assembly language, which is the lowest level in addition to more than binary machine code language, and as so low why is the use? Many high-level language has reached the level of visual programming Why is not it? The reason is simply that there is no home computer as a single chip CPU, not as hard as a mass storage device.A visualization of small high-level language program which even if only one button, will reach tens of K of size! For the home PC's hard drive in terms of nothing, but in terms of the MCU is not acceptable. SCM in the utilization of hardware resources to be very high for the job so although the original is still in the compilation of a lot of use. The same token, if the giant computer operating system and applications run up to get home PC, home PC, also can not afford to.

Can be said that the twentieth century across the three "power" era, that is, the age of electricity, the electronic age and has entered into the computer age. However, this computer, usually refers to the personal computer, referred to as PC. It consists of the host, keyboard,

monitor and other components. Another type of computer, most people do not know how. This computer is to give all kinds of intelligent machines single chip (also known as micro-controller). As the name suggests, this computer system took only a minimal integrated circuit, can be a simple operation and control. Because it is small, usually hidden in the charged mechanical "stomach" in. It is in the device, like the human brain plays a role, it goes wrong, the whole plant was paralyzed. Now, this microcontroller has a very broad field of use, such as smart meters, real-time industrial control, communications equipment, navigation systems, and household appliances. Once all kinds of products were using SCM, can serve to upgrade the effectiveness of products, often in the product name preceded by the adjective - "intelligent," such as intelligent washing machines. Now some technical personnel of factories or other amateur electronics developers to engage in out of certain products, not the circuit is too complicated, that function is too simple and can easily be copied. The reason may be stuck in the product did not use a microcontroller or other programmable logic device.

SCM history

SCM was born in the late 20th century, 70, experienced SCM, MCU, SOC three stages.

First model

1.SCM the single chip microcomputer (Single Chip Microcomputer) stage, mainly seeking the best of the best single form of embedded systems architecture. "Innovation model" success, laying the SCM and general computer completely different path of development. In the open road of independent development of embedded systems, Intel Corporation contributed.

2.MCU the micro-controller (Micro Controller Unit) stage, the main direction of technology development: expanding to meet the embedded applications, the target system requirements for the various peripheral circuits and interface circuits, highlight the object of intelligent control.It involves the areas associated with the object system, therefore, the development of MCU's responsibility inevitably falls on electrical, electronics manufacturers. From this point of view, Intel faded MCU development has its objective factors. In the development of MCU, the most famous manufacturers as the number of Philips Corporation. Philips company in embedded applications, its great advantage, the MCS-51 single-chip micro-computer from the rapid development of the micro-controller. Therefore, when we look back at the path of development of embedded systems, do not forget Intel and Philips in History.

Embedded Systems

Embedded system microcontroller is an independent development path, the MCU important factor in the development stage, is seeking applications to maximize the solution on the chip; Therefore, the development of dedicated single chip SOC trend of the natural form. As the microelectronics, IC design, EDA tools development, application system based on MCU SOC design have greater development. Therefore, the understanding of the microcontroller chip

microcomputer can be, extended to the single-chip micro-controller applications.

MCU applications

SCM now permeate all areas of our lives, which is almost difficult to find traces of the field without SCM. Missile navigation equipment, aircraft, all types of instrument control, computer network communications and data transmission, industrial automation, real-time process control and data processing, extensive use of various smart IC card, civilian luxury car security system, video recorder, camera, fully automatic washing machine control, and program-controlled toys, electronic pet, etc., which are inseparable from the microcontroller. Not to mention the area of robot control, intelligent instruments, medical equipment was. Therefore, the MCU learning, development and application of the large number of computer applications and intelligent control of the scientists, engineers.

SCM is widely used in instruments and meters, household appliances, medical equipment, aerospace, specialized equipment, intelligent management and process control fields, roughly divided into the following several areas:

1. In the application of Intelligent Instruments

SCM has a small size, low power consumption, controlling function, expansion flexibility, the advantages of miniaturization and ease of use, widely used instrument, combining different types of sensors can be realized Zhuru voltage, power, frequency, humidity, temperature, flow, speed, thickness, angle, length, hardness, elemental, physical pressure measurement. SCM makes use of digital instruments, intelligence, miniaturization, and functionality than electronic or digital circuits more powerful. Such as precision measuring equipment (power meter, oscilloscope, various analytical instrument).

2. In the industrial control application

With the MCU can constitute a variety of control systems, data acquisition system. Such as factory assembly line of intelligent control

3. In Household Appliances

can be said that the appliances are basically using SCM, praise from the electric rice, washing machines, refrigerators, air conditioners, color TV, and other audio video equipment, to the electronic weighing equipment, varied, and omnipresent.

4. In the field of computer networks and communications applications

MCU general with modern communication interface, can be easy with the computer data communication, networking and communications in computer applications between devices had excellent material conditions, are basically all communication equipment to achieve a controlled by MCU from mobile phone, telephone, mini-program-controlled switchboards, building automated communications call system, train radio communication, to the daily work can be seen everywhere in the mobile phones, trunked mobile radio, walkie-talkies, etc..

5. Microcomputer in the field of medical device applications

SCM in the use of medical devices is also quite extensive, such as medical respirator, the various analyzers, monitors, ultrasound diagnostic equipment and hospital beds, etc. call system.

6. In a variety of major appliances in the modular applications

Designed to achieve some special single specific function to be modular in a variety of circuit applications, without requiring the use of personnel to understand its internal structure. If music integrated single chip, seemingly simple function, miniature electronic chip in the net (the principle is different from the tape machine), you need a computer similar to the principle of the complex. Such as: music signal to digital form stored in memory (like ROM), read by the microcontroller, analog music into electrical signals (similar to the sound card).

In large circuits, modular applications that greatly reduce the volume, simplifies the circuit and reduce the damage, error rate, but also easy to replace.

7. Microcontroller in the application field of automotive equipment

SCM in automotive electronics is widely used, such as a vehicle engine controller, CAN bus-based Intelligent Electronic Control Engine, GPS navigation system, abs anti-lock braking system, brake system, etc..

In addition, the MCU in business, finance, research, education, national defense, aerospace and other fields has a very wide range of applications.

Application of six important part of learning

MCU learning an important part of the six applications

1, Bus:

We know that a circuit is always made by the devices connected by wires, in analog circuits, the connection does not become a problem because the device is a serial relationship between the general, the device is not much connection between the , but the computer is not the same circuit, it is a microprocessor core, the device must be connected with the microprocessor, the device must be coordination between, so they need to connect on a lot, as if still analog circuit like the microprocessor and devices in the connection between the individual, the number of lines will be a little more surprising, therefore the introduction of the microprocessor bus Zhong Each device Gongtong access connections, all devices 8 Shuju line all received eight public online, that is the equivalent of all devices together in parallel, but only this does not work, if there are two devices send data at the same time, a 0, a 1, then, whether the receiver received what is it? This situation is not allowed, so to be controlled by controlling the line, time-sharing the device to work at any time only one device to send data (which can have multiple devices to receive both). Device's data connection is known as the data bus, the device is called line of control all the control bus. Internal or external memory in the microcontroller and other devices have memory cells, the memory cell to be assigned addresses, you can use, distribution, of course, to address given in

the form of electrical signals, and as more memory cells, so, for the address allocation The line is also more of these lines is called the address bus.

Second, data, address, command

The reason why these three together because of the nature of these three are the same - the number, or are a string of '0 'and '1' form the sequence. In other words, addresses, instructions are also data. Instruction: from single chip designer provides a number of commonly used instructions with mnemonic we have a strict correspondence between the developer can not be changed by the MCU. Address: the search for MCU internal, external storage units, input and output port based on the address of the internal unit value provided by the chip designer is good, can not be changed, the external unit can be single chip developers to decide, but there are a number of address units is a must (see procedures for the implementation of the process).

Third, P0 port, P2 and P3 of the second function I use:

Beginners often on the P0 port, P2 and P3 port I use the second function puzzled that the second function and have a switch between the original function of the process, or have a directive, in fact, the port The second feature is automatic, do not need instructions to convert. Such as P3.6, P3.7 respectively WR, RD signal, when the microchip processing machines external RAM or external I / O port, they are used as a second function, not as a general-purpose I / O port used, so long as a A microprocessor implementation of the MOVX instruction, there will be a corresponding signal sent from the P3.6 or P3.7, no prior use of commands. In fact 'not as a general-purpose I / O port use' is also not a 'no' but (user) 'not' as a general-purpose I / O port to use. You can arrange the order of a SETB P3.7's instructions, and when the MCU execution to the instruction, the also make P3.7 into a high, but users will not do so because this is usually will cause the system to collapse.

Fourth, the program's implementation:

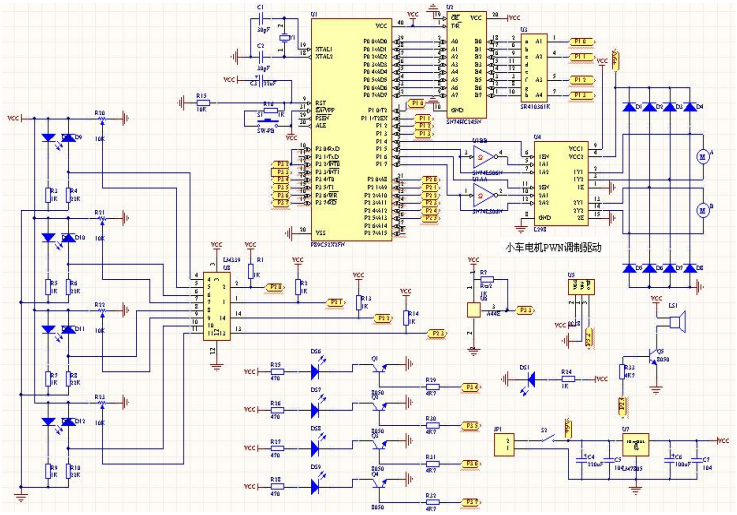
Reduction in power after the 8051 microcontroller within the program counter (PC) in the value of 0000 ', the process is always from the 0000' units started, that is: the system must exist in ROM 0000 'this unit , and in 0000 'unit must be stored in a single instruction.

5, the stack:

Stack is a region, is used to store data, there is no special about the region itself is a part of internal RAM, special access to its data storage and the way that the so-called 'advanced post out backward first out ', and the stack has a special data transmission instructions that' PUSH 'and' POP ', has a special expertise in its services unit, that is, the stack pointer SP, whenever a PUSH instruction execution, SP on (in the Based on the original value) automatically add 1, whenever the implementation of a POP instruction, SP will (on the basis of the original value) automatically by 1. As the SP values can be changed with the instructions, so long as the beginning of the process to change the value of the SP, you can set the stack memory unit

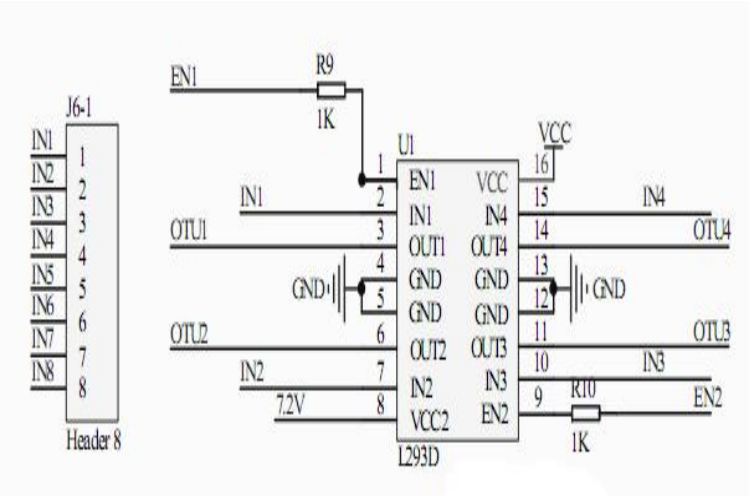
required, such as the program begins, with an MOV SP, # 5FH instructions When set on the stack starting from the memory unit 60H unit. There is always the beginning of the general procedure with such a directive to set the stack pointer, because boot, SP initial value of 07H, 08H This unit from the beginning to stack next, and 08H to 1FH 8031 is the second in the region, three or four working register area, often used, this will lead to confusion of data. Different authors when writing programs, initialize the stack is not exactly the same directive, which is the author's habit. When set up the stack zone, does not mean that the region become a special memory, it can still use the same memory region as normal, but generally the programmer does not regard it as an ordinary memory used.

附录 C 系统连接原理图



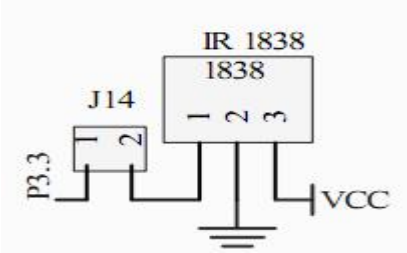
附 1 系统连接图

系统连接图如附 1 所示。P1.0 接口到 P1.7 接口连接驱动电路模块，P3.4 接口到 P3.7 接口连接红外避障和红外循迹模块，其中 P3.4 与 P3.7 为红外避障模块。超声波模块连接 P2.4 与 P2.5 接口。红外遥控模块连接 P3.3 接口。蓝牙遥控模块接 P3.0 与 P3.1 接口。



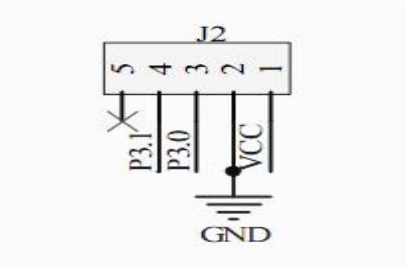
附 2 电机驱动连接图

电机驱动电路连接图如图附 2 所示，有两个 L293D 驱动芯片，每个芯片控制两个电机，每个芯片连接连接四个接口，在整个系统中连接 P1.0 到 P1.7 接口。



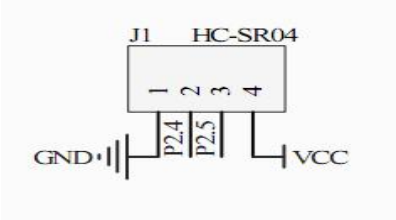
附 3 红外遥控连接图

红外遥控连接图如图附 3 所示，只有一条数据传输线，在整个系统中只连接 P3.3 接口。



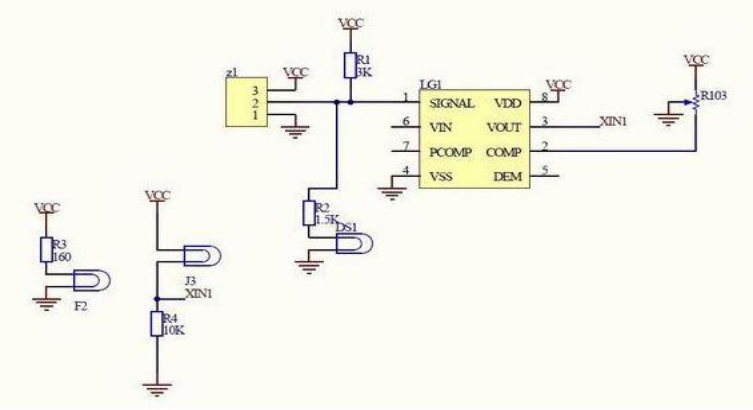
附 4 蓝牙遥控连接图

蓝牙遥控连接图如图附 4 所示，需要连接两个单片机接口，在整个系统中连接 P3.0 与 P3.1 接口。



附 5 超声波连接图

超声波连接图如图附 5 所示，同样需要连接两个单片机接口，在整个系统中连接 P2.4 与 P2.5 接口。



附 6 红外循迹与避障连接图

红外循迹避障两个模块连接图相似，只是接口不同。连接图如图附 6 所示。每个模块需要连接两个单片机接口，循迹模块为 P3.5 与 P3.6 接口，避障模块为 P3.4 与 P3.7 接口。

附录 D 源程序代码

系统主程序

```
#include<AT89X52.H>
#define Left_moto_go       {P1_4=1,P1_5=0,P1_6=1,P1_7=0;}      //左电机向前走
#define Left_moto_back     {P1_4=0,P1_5=1,P1_6=0,P1_7=1;}     //左电机向后转
#define Left_moto_Stop     {P1_4=0,P1_5=0,P1_6=0,P1_7=0;}     //左电机停转
#define Right_moto_go      {P1_0=1,P1_1=0,P1_2=1,P1_3=0;}     //右电机向前走
#define Right_moto_back    {P1_0=0,P1_1=1,P1_2=0,P1_3=1;}     //右电机向后走
#define Right_moto_Stop    {P1_0=0,P1_1=0,P1_2=0,P1_3=0;}    //右电机停转
#define Imax 14000       //此处为晶振为 11.0592 时的取值,
#define Imin 8000        //如用其它频率的晶振时,
#define Inum1 1450       //要改变相应的取值。
#define Inum2 700
#define Inum3 3000
      unsigned char f=0;
      unsigned char Im[4]={0x00,0x00,0x00,0x00};
      unsigned char show[2]={0,0};
      unsigned long m,Tc;
      unsigned char IrOK;
/*****************************************************************/
//延时函数
void delay(unsigned int k)
{
      unsigned int x,y;
        for(x=0;x<k;x++)
          for(y=0;y<2000;y++);
}
/*****************************************************************//
/外部中断解码程序
```

```
void intersvr1(void) interrupt 2 using 1
{
    Tc=TH0*256+TL0;                 //提取中断时间间隔时长
    TH0=0;
    TL0=0;                  //定时中断重新置零
    if((Tc>Imin)&&(Tc<Imax))
    {
        m=0;
        f=1;
        return;
    }               //找到启始码
    if(f==1)
    {
        if(Tc>Inum1&&Tc<Inum3)
        {
            Im[m/8]=Im[m/8]>>1|0x80; m++;
        }
        if(Tc>Inum2&&Tc<Inum1)
        {
            Im[m/8]=Im[m/8]>>1; m++;                //取码
        }
        if(m==32)
        {
            m=0;
            f=0;
            if(Im[2]==~Im[3])
            {
                IrOK=1;
            }
            else IrOK=0;                //取码完成后判断读码是否正确
```

```
            }
        }
}
/*****************************************************************/
//前速前进
void    run(void)
{
    Left_moto_go ;              //左电机往前走
    Right_moto_go ;             //右电机往前走
}
//前速后退
void    backrun(void)
{
    Left_moto_back ;            //左电机往前走
    Right_moto_back ;           //右电机往前走
}
//左转
void    leftrun(void)
{
    Left_moto_back ;            //左电机往前走
    Right_moto_go ;             //右电机往前走
}
//右转
void    rightrun(void)
{
    Left_moto_go ;              //左电机往前走
    Right_moto_back ;           //右电机往前走
}
//STOP
void    stoprun(void)
```

```
{
    Left_moto_Stop ;              //左电机往前走
    Right_moto_Stop ;             //右电机往前走
}
/*********************************************************************/
/*--主函数--*/
void main(void)
{
    m=0;
    f=0;
    IT1=1;
    EX1=1;
    TMOD=0x11;
    TH0=0;
    TL0=0;
    TR0=1;
    EA=1;
    delay(100);

    while(1)                  /*无限循环*/
    {
        if(IrOK==1)
        {
            switch(Im[2])
            {
                case 0x18:  run();            //前进
                break;
                case 0x52:  backrun();            //后退
                break;
                case 0x5a:  leftrun();            //左转
```

```
                    break;
                    case 0x08:    rightrun();              //右转
                    break;
                    case 0x1C:    stoprun();               //停止
                    break;
                    case 0x0C:    ultrasonic_wave();              //超声波避障
                    break;
                    case 0x5E:    infrared_avoidance();              //红外避障
                    break;
                    case 0x42:    infrared_tracking();              //红外循迹
                    break;
                    case 0x4A:    bluetooth_remote();              //蓝牙遥控
                    break;
                    default:    break;
                }
            IrOK=0;
            }
        }
}
```

超声波避障程序

```
#include<AT89x51.H>
#include <intrins.h>
#define Sevro_moto_pwm        P2_7      //接舵机信号端输入 PWM 信号调节速度
#define    ECHO    P2_4                 //超声波接口定义
#define    TRIG    P2_5                 //超声波接口定义
//HJ-4WD 小车驱动接线定义
#define Left_moto_go        {P1_4=1,P1_5=0,P1_6=1,P1_7=0;}      //左电机向前走
#define Left_moto_back      {P1_4=0,P1_5=1,P1_6=0,P1_7=1;}      //左电机向后转
#define Left_moto_Stop      {P1_4=0,P1_5=0,P1_6=0,P1_7=0;}      //左电机停转
```

```
#define Right_moto_go       {P1_0=1,P1_1=0,P1_2=1,P1_3=0;}      //右电机向前走
#define Right_moto_back     {P1_0=0,P1_1=1,P1_2=0,P1_3=1;}      //右电机向后走
#define Right_moto_Stop     {P1_0=0,P1_1=0,P1_2=0,P1_3=0;}      //右电机停转
    Unsigned char const discode[] =
{ 0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0xBF,0xff/*-*/};
    unsigned char const positon[3]={ 0xfe,0xfd,0xfb};
    unsigned char disbuff[4] ={ 0,0,0,0,};
    unsigned char posit=0;
    unsigned char pwm_val_left    = 0;                  //变量定义
    unsigned char push_val_left =14;
    unsigned long S=0;
    unsigned long S1=0;
    unsigned long S2=0;
    unsigned long S3=0;
    unsigned long S4=0;
    unsigned int    time=0;                  //时间变量
    unsigned int    timer=0;                 //延时基准变量
    unsigned char timer1=0;                  //扫描时间变量
/*********************************************************************/
void delay(unsigned int k)                  //延时函数
{
    unsigned int x,y;
     for(x=0;x<k;x++)
       for(y=0;y<2000;y++);
}
/*********************************************************************/
void Display(void)                  //扫描数码管
{
    if(posit==0)
    {
```

```
        P0=(discode[disbuff[posit]])&0x7f;
    }
    else
    {
        P0=discode[disbuff[posit]];
    }
    if(posit==0)
    {
        P2_1=0;P2_2=1;P2_3=1;
    }
    if(posit==1)
    {
        P2_1=1;P2_2=0;P2_3=1;
    }
    if(posit==2)
    {
        P2_1=1;P2_2=1;P2_3=0;
    }
    if(++posit>=3)
        posit=0;
}
/************************************************************************/
void   StartModule()                //启动测距信号
{
    TRIG=1;
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
```

```
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    _nop_();
    TRIG=0;
}
/*****************************************************************/
void Conut(void)                //计算距离
{
    while(!ECHO);                //当 RX 为零时等待
    TR0=1;                       //开启计数
    while(ECHO);                 //当 RX 为 1 计数并等待
    TR0=0;                       //关闭计数
    time=TH0*256+TL0;            //读取脉宽长度
    TH0=0;
    TL0=0;
    S=(time*1.7)/100;            //算出来是 CM
    disbuff[0]=S%1000/100;       //更新显示
```

```
    disbuff[1]=S%1000%100/10;
    disbuff[2]=S%1000%10 %10;
}
/*********************************************************************/
//前速前进
void   run(void)
{
    Left_moto_go ;              //左电机往前走
    Right_moto_go ;             //右电机往前走
}
/*********************************************************************/
//前速后退
void   backrun(void)
{
    Left_moto_back ;            //左电机往前走
    Right_moto_back ;           //右电机往前走
}
/*********************************************************************/
//左转
void   leftrun(void)
{
    Left_moto_back ;            //左电机往前走
    Right_moto_go ;             //右电机往前走
}
/*********************************************************************/
//右转
void   rightrun(void)
{
    Left_moto_go ;              //左电机往前走
    Right_moto_back ;           //右电机往前走
```

```
}
/*****************************************************************/
//STOP
void   stoprun(void)
{
    Left_moto_Stop ;              //左电机停走
    Right_moto_Stop ;              //右电机停走
}
/*****************************************************************/
void   COMM( void )
{
    push_val_left=5;              //舵机向左转 90 度
    timer=0;
    while(timer<=4000);               //延时 400MS 让舵机转到其位置
    StartModule();                //启动超声波测距
    Conut();              //计算距离
    S2=S;
    push_val_left=23;              //舵机向右转 90 度
    timer=0;
    while(timer<=4000);               //延时 400MS 让舵机转到其位置
    StartModule();                //启动超声波测距
    Conut();              //计算距离
    S4=S;
    push_val_left=14;              //舵机归中
    timer=0;
    while(timer<=4000);               //延时 400MS 让舵机转到其位置
    StartModule();                //启动超声波测距
    Conut();              //计算距离
    S1=S;
    if((S2<20)||(S4<20))              //只要左右距离小于 20CM 小车后退
```

```
    {
        backrun();                  //后退
        timer=0;
        while(timer<=4000);
    }
    if(S2>S4)
    {
    rightrun();                     //车的左边比车的右边距离小，右转
    timer=0;
    while(timer<=4000);
    }
    else
    {
    leftrun();                  //车的左边比车的右边距离大，左转
    timer=0;
    while(timer<=4000);
    }
}
/*********************************************************************/
/*PWM 调制电机转速*/
/*********************************************************************/
/*左电机调速*/
/*调节 push_val_left 的值改变电机转速,占空比*/
void pwm_Servomoto(void)
{
    if(pwm_val_left<=push_val_left)
        Sevro_moto_pwm=1;
    else
        Sevro_moto_pwm=0;
    if(pwm_val_left>=200)
```

```
    pwm_val_left=0;
}
/*********************************************************************/
/*TIMER1 中断服务子函数产生 PWM 信号*/
void time1()interrupt 3      using 2
{
    TH1=(65536-100)/256;                //100US 定时
    TL1=(65536-100)%256;
    timer++;                  //定时器 100US 为准。在这个基础上延时
    pwm_val_left++;
    pwm_Servomoto();
    timer1++;                  //2MS 扫一次数码管
    if(timer1>=20)
    {
    timer1=0;
    Display();
    }
 }
/*********************************************************************/
/*TIMER0 中断服务子函数产生 PWM 信号*/
void timer0()interrupt 1      using 0
{

}
/*********************************************************************/
void main(void)
{
    TMOD=0X11;
    TH1=(65536-100)/256;                //100US 定时
    TL1=(65536-100)%256;
```

```
    TH0=0;
    TL0=0;
    TR1= 1;
    ET1= 1;
    ET0= 1;
    EA = 1;
    delay(100);
    push_val_left=14;              //舵机归中
    while(1)                    /*无限循环*/
    {
        if(timer>=1000)             //100MS 检测启动检测一次
        {
            timer=0;
            StartModule();              //启动检测
            Conut();              //计算距离
            if(S<20)                //距离小于 20CM
            {
                stoprun();              //小车停止
                COMM();              //方向函数
            }
            else if(S>30)             //距离大于，30CM 往前走
            run();
        }
    }
}
```

红外避障程序

```
#include<AT89X52.H>
//HJ-4WD 小车驱动接线定义
#define Left_moto_go        {P1_4=1,P1_5=0,P1_6=1,P1_7=0;}      //左电机向前走
```

```
#define Left_moto_back     {P1_4=0,P1_5=1,P1_6=0,P1_7=1;}      //左电机向后转
#define Left_moto_Stop     {P1_4=0,P1_5=0,P1_6=0,P1_7=0;}      //左电机停转
#define Right_moto_go      {P1_0=1,P1_1=0,P1_2=1,P1_3=0;}      //右电机向前走
#define Right_moto_back    {P1_0=0,P1_1=1,P1_2=0,P1_3=1;}      //右电机向后走
#define Right_moto_Stop    {P1_0=0,P1_1=0,P1_2=0,P1_3=0;}      //右电机停转
//传感器定义 R 是右(right),L 是左(left)      小车对着自己看时分的左右
//避障传感器 B  左 P3.4    右 P3.7
#define Left_B_led            P3_4  //P3_4 接左边  红外避障传感器  接第 2 路输出信号即中控板上面标记为 OUT1
#define Right_B_led           P3_7  //P3_7 接左边  红外避障传感器  接第 2 路输出信号即中控板上面标记为 OUT4
/*****************************************************************/// 延 时 函数
void delay(unsigned int k)
{
      unsigned int x,y;
       for(x=0;x<k;x++)
         for(y=0;y<2000;y++);
}
/*****************************************************************/
//前速前进
void    run(void)
{
      Left_moto_go ;                //左电机往前走
      Right_moto_go ;               //右电机往前走
}
/*****************************************************************/
//前速后退
void    backrun(void)
{
```

```
    Left_moto_back ;              //左电机往前走
    Right_moto_back ;             //右电机往前走
}
/*****************************************************************/
//左转
void   leftrun(void)
{
    Left_moto_back ;              //左电机往前走
    Right_moto_go ;               //右电机往前走
}
/*****************************************************************/
//右转
void   rightrun(void)
{
    Left_moto_go ;                //左电机往前走
    Right_moto_back ;             //右电机往前走
}
/*****************************************************************/
//STOP
void   stoprun(void)
{
    Left_moto_Stop ;              //左电机停走
    Right_moto_Stop ;             //右电机停走
}
//主函数
void main(void)
{
    unsigned char i;
    P1=0X00;                      //小车停止
    TMOD=0X01;
```

```
TH0= 0XFc;                //1ms 定时
TL0= 0X18;
TR0= 1;
ET0= 1;
EA = 1;
while(1)                  //无限循环
{
    //有信号为 0   没有信号为 1
    //传感器定义  R 是右(right),L 是左(left)    小车对着自己看时  分的左右
    if(Left_B_led==1&&Right_B_led==1)              //前面无物体时
        run();
    else
    {
        //左边检测到红外     小车需要向右转
        if(Left_B_led==0&&Right_B_led==1)
        {
            rightrun();              //调用右转函数
        }
        //右边检测到红外     小车需要向左转
        if(Right_B_led==0&&Left_B_led==1)
        {
            leftrun();               //调用左转函数
        }
        //两边检测到红外  小车后退
        if(Right_B_led==0&&Left_B_led==0)
        {
            backrun();               //调用电机向后转函数
        }
    }
}
```

```
}
```

红外循迹程序

```
#include<AT89X52.H>            //包含 51 单片机头文件，内部有各种寄存器定义
//HJ-4WD 小车驱动接线定义
#define Left_moto_go        {P1_4=1,P1_5=0,P1_6=1,P1_7=0;}      //左电机向前走
#define Left_moto_back      {P1_4=0,P1_5=1,P1_6=0,P1_7=1;}      //左电机向后转
#define Left_moto_Stop      {P1_4=0,P1_5=0,P1_6=0,P1_7=0;}      //左电机停转
#define Right_moto_go       {P1_0=1,P1_1=0,P1_2=1,P1_3=0;}      //右电机向前走
#define Right_moto_back     {P1_0=0,P1_1=1,P1_2=0,P1_3=1;}      //右电机向后走
#define Right_moto_Stop     {P1_0=0,P1_1=0,P1_2=0,P1_3=0;}      //右电机停转
//传感器定义 R 是右(right),L 是左(left)      小车对着自己看时 分的左右
//循迹传感器 X 左 P3.5   右 P3.6
#define Left_X_led            P3_5  //P3_5 接左边 红外循迹传感器 接第 2 路输出信号即中控板上面标记为 OUT2
#define Right_X_led           P3_6  //P3_6 接右边 红外循迹传感器 接第 3 路输出信号即中控板上面标记为 OUT3
/*****************************************************************/
//延时函数
void delay(unsigned int k)
{
      unsigned int x,y;
       for(x=0;x<k;x++)
         for(y=0;y<2000;y++);
}
/*****************************************************************/
//前速前进
void    run(void)
{
      Left_moto_go ;                //左电机往前走
```

```
    Right_moto_go ;              //右电机往前走
}
/*********************************************************************/
//前速后退
void   backrun(void)
{
    Left_moto_back ;             //左电机往前走
    Right_moto_back ;            //右电机往前走
}
/*********************************************************************/
//左转
void   leftrun(void)
{
    Left_moto_back ;             //左电机往前走
    Right_moto_go ;              //右电机往前走
}
/*********************************************************************/
//右转
void   rightrun(void)
{
    Left_moto_go ;               //左电机往前走
    Right_moto_back ;            //右电机往前走
}
/*********************************************************************/
//STOP
void   stoprun(void)
{
    Left_moto_Stop ;             //左电机停走
    Right_moto_Stop ;            //右电机停走
}
```

```
//主函数
void main(void)
{
    unsigned char i;
    P1=0X00;              //小车停止
    TMOD=0X01;
    TH0= 0XFc;            //1ms 定时
    TL0= 0X18;
    TR0= 1;
    ET0= 1;
    EA = 1;
    while(1)    //无限循环
    {
        //有信号为 0   没有信号为 1
        if(Left_X_led==0&&Right_X_led==0)     //白线
            run();
        else
        {
            if(Left_X_led==0&&Right_X_led==1)        //左边检测到红外
            {
                leftrun();                           //左边两个电机正转
            }
            if(Right_X_led==0&&Left_X_led==1)        //右边检测到红外
            {
                rightrun();                          //右边两个电机正转
            }
        }
    }
}
```

蓝牙遥控

```
#include<AT89x51.H>
//HJ-4WD 小车驱动接线定义
#define Left_moto_go        {P1_4=1,P1_5=0,P1_6=1,P1_7=0;}     //左边两个电机向前走
#define Left_moto_back      {P1_4=0,P1_5=1,P1_6=0,P1_7=1;}     //左边两个电机向后转
#define Left_moto_Stop      {P1_4=0,P1_5=0,P1_6=0,P1_7=0;}     //左边两个电机停转
#define Right_moto_go       {P1_0=1,P1_1=0,P1_2=1,P1_3=0;}    //右边两个电机向前走
#define Right_moto_back     {P1_0=0,P1_1=1,P1_2=0,P1_3=1;}    //右边两个电机向后走
#define Right_moto_Stop     {P1_0=0,P1_1=0,P1_2=0,P1_3=0;}    //右边两个电机停转

#define left          'C'
#define right         'D'
#define up            'A'
#define down          'B'
#define stop          'F'
char code str[] =    "收到指令，向前!\n";
char code str1[] = "收到指令，向后!\n";
char code str2[] = "收到指令，向左!\n";
char code str3[] = "收到指令，向右!\n";
char code str4[] = "收到指令，停止!\n";

bit    flag_REC=0;
bit    flag       =0;
unsigned char    i=0;
unsigned char    dat=0;
unsigned char    buff[5]=0; //接收缓冲字节
/*****************************************************************/// 延 时 函
数
void delay(unsigned int k)
```

```
{
    unsigned int x,y;
     for(x=0;x<k;x++)
       for(y=0;y<2000;y++);
}
/*********************************************************************/
//字符串发送函数
void send_str( )
// 传送字串
{
    unsigned char i = 0;
    while(str[i] != '\0')
    {
        SBUF = str[i];
        while(!TI);              // 等特数据传送
        TI = 0;                  // 清除数据传送标志
        i++;                     // 下一个字符
    }
}
void send_str1( )
// 传送字串
{
    unsigned char i = 0;
    while(str1[i] != '\0')
    {
        SBUF = str1[i];
        while(!TI);              // 等特数据传送
        TI = 0;                  // 清除数据传送标志
        i++;                     // 下一个字符
    }
```

```
}
void send_str2( )
// 传送字串
{
    unsigned char i = 0;
    while(str2[i] != '\0')
    {
        SBUF = str2[i];
        while(!TI);              // 等特数据传送
        TI = 0;                  // 清除数据传送标志
        i++;                     // 下一个字符
    }
}
void send_str3()
// 传送字串
{
    unsigned char i = 0;
    while(str3[i] != '\0')
    {
        SBUF = str3[i];
        while(!TI);              // 等特数据传送
        TI = 0;                  // 清除数据传送标志
        i++;                     // 下一个字符
    }
}
void send_str4()
// 传送字串
{
    unsigned char i = 0;
    while(str4[i] != '\0')
```

```
    {
        SBUF = str4[i];
        while(!TI);             // 等特数据传送
        TI = 0;                 // 清除数据传送标志
        i++;                    // 下一个字符
    }
}
/*********************************************************************/
//前速前进
void  run(void)
{
    Left_moto_go ;    //左电机往前走
    Right_moto_go ;   //右电机往前走
}
//前速后退
void  backrun(void)
{
    Left_moto_back ;    //左电机往前走
    Right_moto_back ;   //右电机往前走
}
//左转
void  leftrun(void)
{
    Left_moto_back ;    //左电机往前走
    Right_moto_go ;   //右电机往前走
}
//右转
void  rightrun(void)
{
    Left_moto_go ;    //左电机往前走
```

```
    Right_moto_back ;   //右电机往前走
}
//STOP
void   stoprun(void)
{
    Left_moto_Stop ;    //左电机往前走
    Right_moto_Stop ;   //右电机往前走
}
/*****************************************************************/
void sint() interrupt 4    //中断接收 3 个字节
{
    if(RI)                          //是否接收中断
    {
        RI=0;
        dat=SBUF;
        if(dat=='O'&&(i==0)) //接收数据第一帧
        {
            buff[i]=dat;
            flag=1;             //开始接收数据
        }
        else if(flag==1)
        {
            i++;
            buff[i]=dat;
            if(i>=2)
            {
                i=0;flag=0;flag_REC=1 ;
            }  // 停止接收
        }
    }
```

```
}
/*****************************************************************//*-- 主 函数--*/
void main(void)
{
    TMOD=0x20;
    TH1=0xFd;              //11.0592M 晶振，9600 波特率
    TL1=0xFd;
    SCON=0x50;
    PCON=0x00;
    TR1=1;
    ES=1;
    EA=1;
    while(1)                          /*无限循环*/
    {
        if(flag_REC==1)                    //
        {
            flag_REC=0;
            if(buff[0]=='O'&&buff[1]=='N') //第一个字节为 O，第二个字节为 N，第三个字节为控制码
            switch(buff[2])
            {
                case up :                  // 前进
                send_str( );
                run();
                break;
                case down:                 // 后退
                send_str1( );
                backrun();
                break;
```

```
                case left:                  // 左转
                send_str3( );
                leftrun();
                break;
                case right:                 // 右转
                send_str2( );
                rightrun();
                break;
                case stop:                  // 停止
                send_str4( );
                stoprun();
                break;
            }
        }
    }
}
```