基于STC89C52RC设计智能循迹小车

SUAT24000148蔡成易 SUAT24000164 刘耀昊

算力微电子学院

* 1 概述

本文基于STC89C52RC单片机设计了一种智能小车，该智能小车分为智能小车主控板，智能小车电源板和智能小车底板。该智能小车由STC89C52RC单片机、功能按键、状态显示LED、红外循迹模块、电机驱动、供电电源等组成。不仅如此，智能小车还可以根据实际需要添加相关功能模块，进行功能的拓展。

* 2 硬件设计

2.1智能小车主控板设计

2.1.1 单片机最小系统板设计

我们设计的C51单片机的最小系统板，其具备以下模块：

1. 处理器：型号：STC89C52RC。
2. 震荡电路：STC89C52RC单片机是一种时序电路，必须提供脉冲信号才能正常工作。在这里STC89C52RC单片机外接24MHz的晶振X1来提供脉冲信号。
3. 复位电路：复位电路包括10uF电容C5、10kΩ电阻R1和按键SW3。当按下按键时，处理器会进行复位。
4. 图示, 示意图

   AI 生成的内容可能不正确。供电指示：加入led灯来指示是否能供上电给主控板。

图1 单片机最小系统板原理图

2.1.2 程序下载电路设计

之前出现过这样的问题：当闭合开关后，供电指示灯亮，但断开开关后，供电指示灯还是亮着的。通过排查故障，发现问题产生原因是CH340N充上电空闲时，RXD和TXD引脚都维持着5V电压，然后连接到MCU就会给芯片供电，让芯片的VCC引脚产生了3.3V电压，即产生电流倒灌现象。

为了解决这样的问题，参考关于CH340DS1的手册设计的电路，得知所用的程序下载芯片CH340N需要防止CH340 有电但MCU无电时的外灌问题。于是根据手册，我们考虑加入场效应管（2SK3018)，肖基特二极管（B0520W）防止双电源方式下 CH340N 通过 MCU 的 RXD 或 TXD 内部二极管向失电MCU产生电流倒灌的问题。（注：本文的CODE\_VCC指的是USB\_VCC）

图图示

AI 生成的内容可能不正确。2 程序下载模块原理图

2.1.3 Type-C接口电路设计

参考相关数据手册，下面介绍Type-C接口电路设计。

1. 引脚A1,A12,B1,B12：接地；
2. 引脚A4,A9,B4,B9：VBUS相当于电源正极；
3. 引脚A5,B5：两个CC引脚，是在Type-C协议中用来协商充电协议的，还可以协商设备身份（判断主从设备，靠CC引脚上的电阻来判断）；
4. 引脚A6,A7,B6,B7：USB2.0数据传输；
5. 引脚A8,B8：用来传输视频或者音频信号（这里不需要）；
6. 外壳引脚1，2，3，4：接地，为了防止手上的静电影响USB的数据传输，损坏后面元件；
7. 所使用的Type-C接口型号：C456018，CODE\_VCC仍指的是USB\_VCC)。

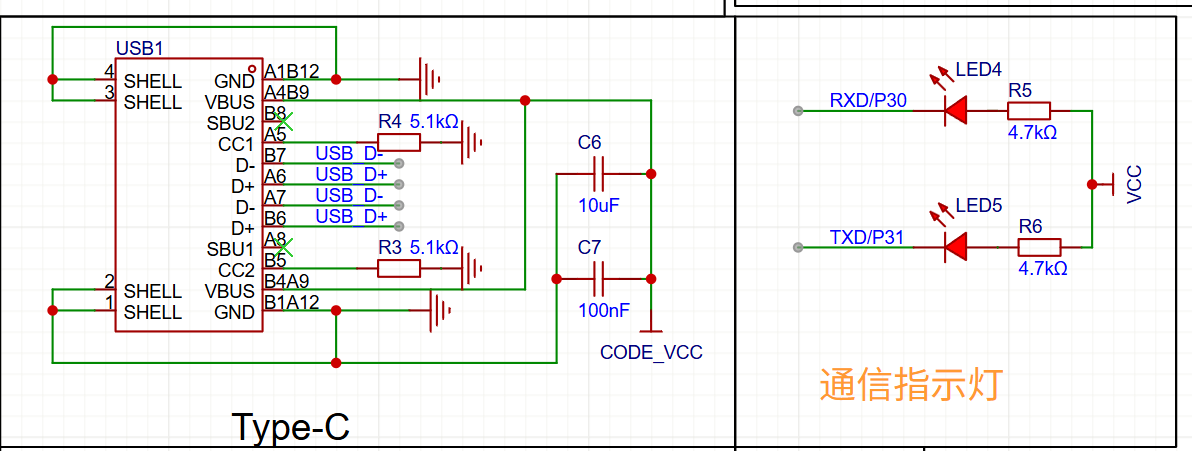


图3 Type-C接口通信原理图

2.1.4 排针电路设计

由于STC89C52RC型号的P0端口在作为通用I/O时具备开漏特性，无法通过引脚输出高电平（3.3V / 5V），**​**​添加10kΩ上拉电阻连接高电平，使得引脚能够输出高电平。其中，所用的上拉电阻型号：C326649。

图示意图

AI 生成的内容可能不正确。4 排针设计原理图

2.1.5 主控板PCB设计经验教训

1. 晶振模块应该紧贴单片机，减少寄生电容干扰。其中PCB走线过长会产生​​寄生电容（5pF~10pF/cm）​​，与晶振负载电容（CL）形成并联：

*C*实际​=*C*负载​+*C*寄生​

（1）

导致振荡频率偏移，起振困难。

1. 原理图标号将CH340N的CODE\_VCC错打成CODE-VCC，导致type-C接口的通信线路没有连接上CH340N，使得通信失败。尝试过飞线处理，但没有解决根本问题，后续是重新做新的板卡。
2. 主控板上的所有排针，led灯等器件都应该标好丝印，方便软件设计上定义引脚。
3. 其中上拉电阻的布线可以通过两个过孔穿过其他线的方式来处理。

2.2 智能小车电源板设计

2.2.1 已知的供电需求

1. C51单片机需要5V供电；
2. 传感器、外设等模块需要5V供电；
3. L298N电机模块需要12V供电；

所以需要一个DC降压模块（用于将12V的电压转换为5V电压，该模块有个旋钮可以调整降压所需要的电压）。一方面向L298N电机模块提供12V电压，另一方面向主控板和其他外设提供5V电压；

2.2.2 电源板所需器件型号

1. XT60接口（C98732）：用于连接大电流电池；
2. DCDC降压模块：用于将12V的电压转换为5V电压；
3. WJ500V接口（C8465）：用于连接L298N与电源管理模块；
4. 船型开关（C309067）：用于控制电源板卡的电路通断；
5. 贴片电容，电阻，led灯等；

2.2.3 电源板原理图和PCB设计经验

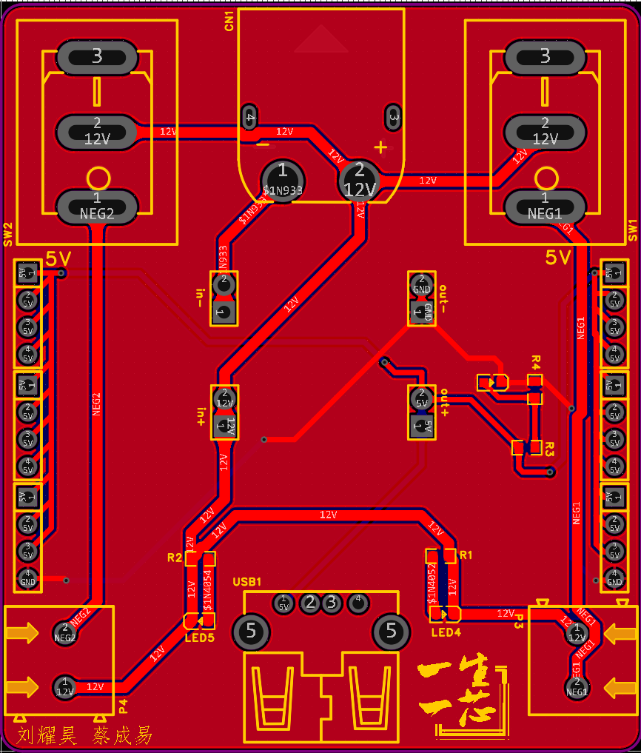
1. 所有接口都应该尽量贴近板框边缘，方便接入。
2. 可以用3D视角来检查USB接口或者Type-C接口有没有接反（血的教训）。
3. 大电压的导线可以选用更粗更宽的，降低导线的电阻，减少电压损失，同时也提高载流能力。宽走线转角要钝角处理，防止蚀刻液会在转角处过度腐蚀铜箔。
4. 可以在设计过程中加入泪滴，有利于增强机械连接强度，也能提高信号传输完整度。
5. 设计过程中要注意DC降压模块的尺寸大小，并且标好其输入端和输出端的正负极的丝印，避免接反，否则会发生通电后降压模块直接被击穿的现象。
6. 假如设计的电源板没有足够的GND时，可以考虑用刀片等锋利物品刮断板卡上的导线，使得VCC引脚电路发生断路，然后从针脚处焊接到GND，实现将VCC改成GND的目的。但是这个方法比较冒险，慎用。

图5 电源板PCB设计图（有错误，仅供参考）

* 3 软件设计
* 3.1概述

代码主要实现循迹（xunji）功能。系统通过两个传感器（left1和right1）检测黑线路径，控制四个电机（通过H桥信号a1、a2、b1、b2、c1、c2、d1、d2以及PWM使能信号ENa、ENb、ENc、ENd）实现前进、左转、右转或停止。定时器中断用于生成PWM信号以控制电机速度。

* 3.2模块介绍

1. main.c 主程序入口，包含主循环

delay\_50ms()：为24MHz晶振设计的50ms延时函数，使用嵌套循环。(测试用)

main()：调用init()初始化系统（来自turn.c），然后进入无限循环，持续调用xunji()执行循迹逻辑。

void main()

{

     init();

    while(1)

    {

    //      delay\_50ms(); // 延时50ms

            xunji();

    }

}

1. motor.c 定义电机控制接口和实现

定义引脚：a1、a2、b1、b2、c1、c2、d1、d2用于控制四个电机的正反转；ENa、ENb、ENc、ENd用于PWM使能。

函数（如IN1(i)、IN2(i)、IN3(i)、IN4(i)）：设置电机方向（i=1正转，i=0反转）。

函数ENA(i)和ENB(i)：控制PWM使能信号，分别对应左右电机组。

void IN1(int i){

    if (i){

    a1 = 1;

    a2 = 0;

    }

    else {

    a1 = 0;

    a2 = 1;

    }

}

1. xunji.c 实现循迹逻辑

定义引脚：a1、a2、b1、b2、c1、c2、d1、d2用于控制四个电机的正反转；ENa、ENb、ENc、ENd用于PWM使能。

函数（如IN1(i)、IN2(i)、IN3(i)、IN4(i)）：设置电机方向（i=1正转，i=0反转）。

函数ENA(i)和ENB(i)：控制PWM使能信号，分别对应左右电机组。

void xunji()

{

    unsigned char flag = 0;

    if((right1 == 0)&&(left1 == 0)&&(right2 == 0)&&(left2 == 0))

    {

        flag = 0;

    }

    if((right1 == 0)&&(left1 == 1)&&(right2 == 0)&&(left2 == 0))   //left1

    {

        flag = 1;

    }

    if((right1 == 1)&&(left1 == 0)&&(right2 == 0)&&(left2 == 0))   //right1

    {

        flag = 2;

    }

    if((right1 == 1)&&(left1 == 1))                                //stop

    {

        flag = 3;

    }

    if((right2 == 1)&&(left2 == 0)&&(right1 == 0)&&(left1 == 0)) //right2

    {

        flag = 4;

    }

    if((right2 == 0)&&(left2 == 1)&&(right1 == 0)&&(left1 == 0)) //left2

    {

        flag = 5;

    }

switchswitch(flag)

    {

        case 0: qianjin(); break;

        case 1: turn\_left1(); break;

        case 2: turn\_right1(); break;

        case 3: stop(); break;

        case 4: // 右急转，减速

            zkb1 = 0; zkb2 = 0; // 低速

            turn\_right2();

            while(1) {

                // 实时检测flag变化

                if(!((right2 == 1)&&(left2 == 0)&&(right1 == 0)&&(left1 == 0))) {

                    break;

                }

            }

            zkb1 = 7; zkb2 = 7; // 恢复正常速度

            break;

        case 5: // 左急转，减速

            zkb1 = 2; zkb2 = 2; // 低速

            turn\_left2();

            while(1) {

                if(!((right2 == 0)&&(left2 == 1)&&(right1 == 0)&&(left1 == 0))) {

                    break;

                }

            }

            zkb1 = 7; zkb2 = 7; // 恢复正常速度

            break;

        default: qianjin(); break;

    }

}

4, turn.c 定义运动控制函数和定时器中断逻辑

全局变量：zkb1和zkb2控制左右电机的PWM占空比，t为定时器计数。

 init()：初始化定时器T0（模式1，100us中断一次），启用全局中断和定时器中断。

 timer0()中断函数：每100us触发，更新PWM信号：

* 若 t < zkb1，则ENA(1)（左电机组开启）；否则ENA(0)。
* 若 t < zkb2，则ENB(1)（右电机组开启）；否则ENB(0)。
* t自增，50次后清零，生成PWM周期。

 运动控制函数：

* qianjin()：所有电机正转（IN1(1)等），zkb1=8、zkb2=8（低速前进）。
* stop()：所有电机停止（IN1(0)等），zkb1=0、zkb2=0。
* turn\_left1()：左电机正转，右电机停止，zkb1=13、zkb2=13（左转）。
* turn\_right1()：右电机正转，左电机停止，zkb1=13、zkb2=13（右转）。
* void init()

    if(t < zkb1)

    {ENA(1);}

    else

    {ENA(0);}

    if(t < zkb2)

    {ENB(1);}

    else {ENB(0);}

    ++t;

    if(t >= 50)

    {t = 0;}

* {
* EA = 1;
* TMOD = (TMOD & 0xF0) |  0x01;
* TH0 =(65536 - 100) / 256;
* TL0 = (65536 - 100)% 256;
* ET0 = 1;
* TR0 = 1;
* }
* 3.3 工作流程
  1. 系统启动，调用init()初始化定时器和中断。
  2. 主循环调用xunji()，读取传感器状态。 根据传感器输入，设置flag并调用相应的运动函数（前进、左转、右转、停止）。
  3. 定时器中断每100us更新PWM信号，根据zkb1和zkb2控制电机速度。
  4. 小车通过传感器检测黑线，动态调整运动方向以实现循迹。
  5. 具体流程如下图6所示

图示

AI 生成的内容可能不正确。

图6 小车工作程序流程图

* 3.4困难部分

1. 小车面对直角/大角度转弯时，由于我们将红外传感器放置在小车正车头处，导致红外传感器无法及时识别路径的变化，小车就直接走出路径。

尝试1：希望小车运动速度尽可能慢些，让红外传感器能够识别到路径。

发现PWM不能太小，否则小车无法克服自身的最大静摩擦力，无法启动。经过多次测试之后，我们设定当小车前进时，PWM为8最为合适。但仍然不能解决问题。

我们猜想：就算小车行进速度足够慢，由于红外传感器接受来自路径上的变化的信号是统一的（要么是1，要么是0），它没有能力去判断什么情况下应该要转多大的角度才能循到黑线。

尝试2：我们想从硬件层面上解决这个问题，需要增加两个红外传感器（right2,left2）置于小车中间部分两侧，xunji.c中对于方向的判断加入right2和left2，以增加侧边识别黑线时大角度转向的控制。

* 4 心得总结

一 项目收获​​

1. ​​硬件设计能力的提升​​
   * ​​最小系统搭建​​：通过设计STC89C52RC最小系统（晶振电路、复位电路、供电指示），比较深刻理解了单片机工作原理。
   * ​​电流倒灌问题的解决​​：在CH340N下载电路中，通过添加场效应管和肖特基二极管，成功解决了USB断电后MCU被反向供电的隐患。这一过程让我们体会到阅读数据手册的重要性，以及硬件细节对系统稳定性的关键影响。
   * ​​PCB设计经验​​：
     + 晶振布线需紧贴MCU，避免寄生电容干扰（教训：首次布线过长导致频率偏移）；
     + 接口丝印标注不全引发调试混乱（如未标注LED引脚功能）；
     + 电源模块正负极反接导致击穿（深刻理解“防呆设计”的必要性）。
2. ​​软件调试与算法优化​​
   * ​​PWM电机控制​​：通过定时器中断（100μs周期）动态调节占空比（zkb1/zkb2），实现电机调速。调试中发现占空比低于阈值（如zkb=8）时电机无法克服静摩擦力，最终通过实验确定了合理区间。
   * ​​循迹逻辑的局限性​​：基础传感器布局（车头两个红外传感器）无法处理直角转弯，尝试降低速度仍失败。这促使我们提出改进方案：​​增加侧向传感器（right2/left2）​​，通过多传感器状态组合提高转向灵活性。
3. ​​团队协作与问题解决​​
   * 硬件（电源板、主控板）与软件（电机控制、循迹算法）分工协作，但调试阶段频繁交叉定位问题（如电机不动时需排查供电、PWM信号、驱动电路三层链路）。
   * 在电源板VCC与GND设计失误时，尝试“刮线改GND”的应急方案，虽险中求成，但也警示了设计阶段充分验证的必要性。

二 遇到的挑战与反思​​

1. ​​硬件设计教训​​
   * ​​接口反接​​：Type-C接口在PCB中反向焊接，因未用3D视图检查封装方向。后续设计中所有接口必须核对3D模型。
   * DCDC降压模块尺寸设计出错：因没认真看明白手册
2. ​**​**软件调试痛点​​
   * ​​直角转弯失效​​：车头传感器无法检测侧向路径变化，暴露了单点检测的局限性。改进方向：
     + 增加传感器数量，构建“田”字形布局；
     + 引入转弯角度分级控制（如小角度微调、大角度急转）。
   * ​​PWM参数调试耗时​​：占空比与电机速度的非线性关系需反复测试，之后有时间可加入编码器测速反馈实现PWM调速。

**​​**三 项目意义与未来展望​​

本次项目不仅巩固了单片机开发、电路设计、PCB绘制等核心技能，更让我们体验到​**​**从理论到落地的完整工程闭环​​：

* ​**​**硬件上​​：理解相关电路原理设计和工程细节，和学会如何利用调试设备发现问题；
* ​​软件上​​：体会到算法必须结合机械特性（如静摩擦力、惯性）；
* ​​工程上​​：全方面提高工程思维和能力，丰富工程经验；

四 致谢

非常感谢学院老师和“一生一芯”项目组李好老师提供的锻炼机会，也非常感谢工作室所有同学和李好老师的耐心解答和帮助。（每次遇到困难，两个李总都能神通广大地帮助我们解决问题，在此特别鸣谢）