重庆邮电大学

学生实验实习报告册

课程名称：自动化与电气工程类基础实验

学　　院：自动化学院/工业互联网学院

专业班级：08052102

姓　　名：王忠全

学　　号：2021212981

指导教师：陈晓雷 林峰

成　　绩：

学年学期：2022 -2023 学年 ☑春□秋学期

重庆邮电大学教务处制

实验项目名称	循迹小车设计实验(1)：基本原理与硬件平台		
实验地点	综合实验楼 C601/602	实验时间	第 4 周周一第 9-12 节
实验指导教师	陈晓雷 林峰	成绩	

一、实验目的

1.理解 51 单片机最小系统原理、结构及应用方法，实现 Keil 环境下单片机程序的编译、下载和运行。

2.自行搭建循迹小车，掌握最小系统板、红外传感器、电机及驱动模块接线方法。

二、实验原理（或设计方案）

1 设计要求

熟悉循迹小车所需器件，了解其原理、结构及功能。实现程序的编译和下载，检验并修正小车控制效果。

2 设计方案

2.1 硬件结构图：

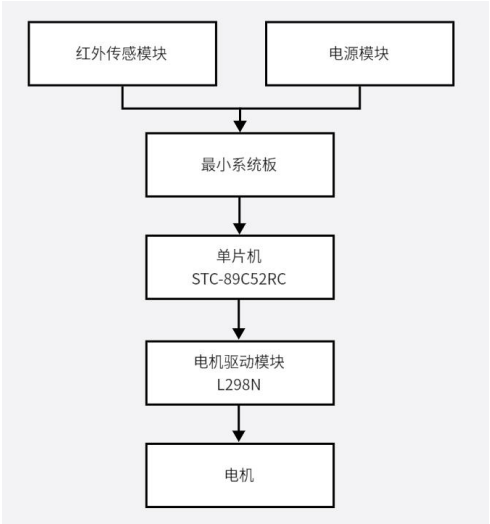


图 1 硬件结构图

采用以单片机 STC89C52RC 为核心，最小系统板，通过电源模块输入驱动电流，搭载红外传感模块、L298N 电机驱动模块，实现红外传感模块监测黑色轨迹，将信号反馈到单片机，通过 CH340G 芯片烧录进入单片机的程序控制电机驱动模块使能电机状态，从而实现循迹功能。

2.2 硬件选型：

循迹小车底盘：带有适配的孔洞透明亚克力板；

杜邦线若干：作为导线以及信号传递；

红外传感器：两个实现能够检测黑色轨迹；

STC-89C52 单片机：控制电机驱动模块以及红外传感检测、电源驱动处理

L298N 电机驱动模块：两个控制一左一右两个驱动轮的转动
CH340G USB 转 TTL 烧录芯片模块：将编译的程序烧录进单片机
最小系统板：为单片机提供最基本的电路支持
万向轮+驱动轮：一个万向轮在前，两个驱动轮一左一右

三、实验仪器设备、材料

3.1. 硬件及材料

图 2 硬件以及材料图

1. **循迹小车底盘**：小车主体，用于承载开发板等，包括转向轮，后轮，电机，载重板等。
2. **杜邦线**：电子行业中，杜邦线可用于实验板的引脚扩展，增加实验项目等，能够非常牢靠地和插针连接，无需焊接，便于快速进行电路试验。
3. **红外传感器**：是一种能够探测并测量红外辐射的电子设备，其作用在于将红外辐射转换为电信号，从而实现对物体的非接触式测量和控制。红外传感器广泛应用于多种领域，如安防、医疗、环保、工业自动化等。此次实验中用于感光以控制小车转向。

图 3 红外传感器实物图

① 当模块检测到前方障碍物信号时，电路板上绿色指示灯点亮电平，同时 OUT 端口持续输出低电平信号,该模块检测距离 2～30cm，检测角度 35°，检测距离可以通过电位器进行调节，顺时针调电位器，检测距离增加；逆时针调电位器，检测距离减少。
② 传感器主动红外线反射探测,因此目标的反射率和形状是探测距离的关键。其中黑色探测距离最小,白色最大;小面积物体距离小,大面积距离大。
③ 传感器模块输出端口 OUT 可直接与单片机 IO 口连接即可，也可以直接驱动一个 5V 继电器；连接方式：VCC-VCC;GND-GND;OUT-IO
④ 可采用 3-5V 直流电源对模块进行供电。当电源接通时，红色电源指示灯点亮；
⑤ 电路板尺寸：3.2CM*1.4CM

4. **STC89C52 单片机**：STC89C52 是一种低功耗、高性能 CMOS8 位微控制器，具有 8K

在系统可编程 Flash 存储器。在单芯片上，拥有灵巧的 8 位 CPU 和在系统可编程 Flash，使得 STC89C52 为众多嵌入式控制应用系统提供高灵活、超有效的解决方案。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 4KBEEPROM，MAX810 复位电路，三个 16 位定时器/计数器，一个 6 向量 2 级中断结构，全双工串行口。

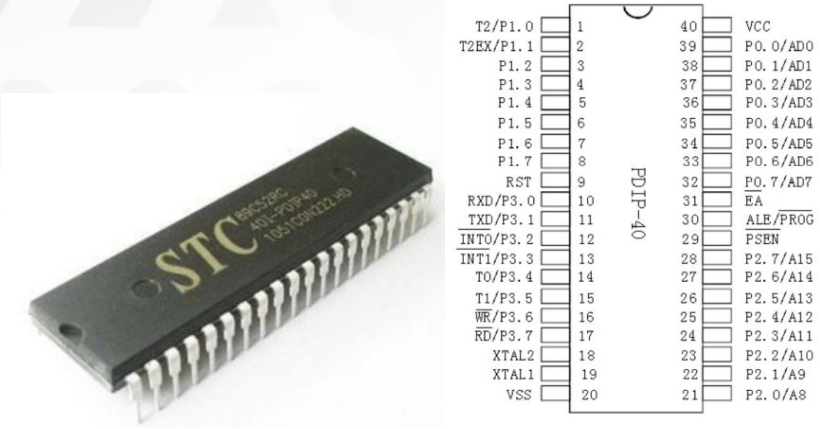
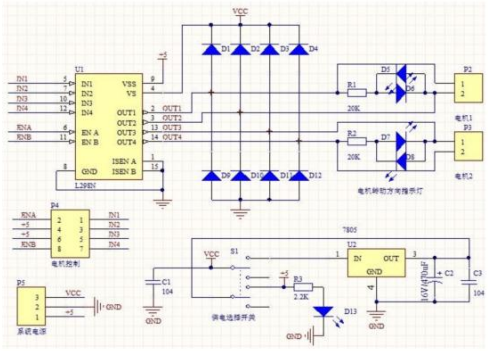


图 4 STC89C52 单片机实物图(左) 引脚以及功能图(右)

5. **L298N 电机驱动模块**：HW-095A L298N 模块是一种电机驱动模块，其作用在于通过控制电流方向和大小，实现对直流电机或步进电机的驱动和控制。该模块采用了双 H 桥电路设计，可以实现正反转和调速等功能，具有高效、稳定、可靠的特点。在机器人、智能小车、航模等领域，HW-095A L298N 模块可以用于控制电机的转向和转速，从而实现运动控制和位置控制。在工业自动化领域，该模块可以用于控制生产线上的电机和传送带等设备。



ENA	IN1	IN2	直流电机状态
0	X	X	停止
1	0	0	制动
1	0	1	正转
1	1	0	反转
1	1	1	制动

图 5 L298N 电机驱动模块电路图(左) 驱动状态表(右)

此双 H 桥直流电机驱动板采用 ST 公司的 L298N 典型双 H 桥，直流电机驱动芯片，可用于驱动直流电机和双极性步进电机，此驱动板体积小，重量轻，具有强大的驱动能力:2A 的峰值电流和 46V 的峰值电压;外加续流二极管可防止电机线圈在断电时的方向电动势损坏芯片;虽然芯片过热时具有自动关断功能，但安装散热片使芯片温度降低，让驱动性能更加稳定;板子设有 2 个电流反馈检测接口、内逻辑取电选择端、4 个上拉电阻选择端、2 路直流电机接口和四线两相步进电机接口、控制电机方向指示灯、4 个标准固定安装孔。此驱动板适用于智能程控小车、轮式机器人等，可配合各种控制器使用。

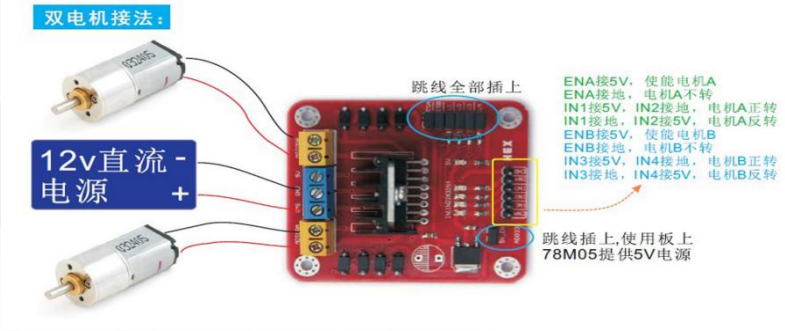


图 6 L298N 电机驱动双电机接法

6. **CH340G USB 转 TTL 烧录芯片模块：**

其中 CH340G 是目前最常用的转换芯片，它不仅能在 PC 系统上面使用，也能使用在嵌入式 linux 系统里面，在 linux 内核版本中已有相应的驱动源码，很容易进行移植开发。

CH340G 支持 5V 电源电压和 3.3V 电源电压甚至 3V 电源电压。

CH340G 芯片内置了 USB 上拉电阻，D+和 D-引脚应该直接连接到 USB 总线上，USB 工作在 USB2.0 全速模式。

CH340G 芯片内置了电源上电复位电路。CH340B 芯片还提供了低电平有效的外部复位输入引脚。

CH340G 芯片正常工作时需要外部向 XI 引脚提供 12MHz 的时钟信号。

硬件全双工串口，内置收发缓冲区，支持通讯波特率 50bps～2Mbps。

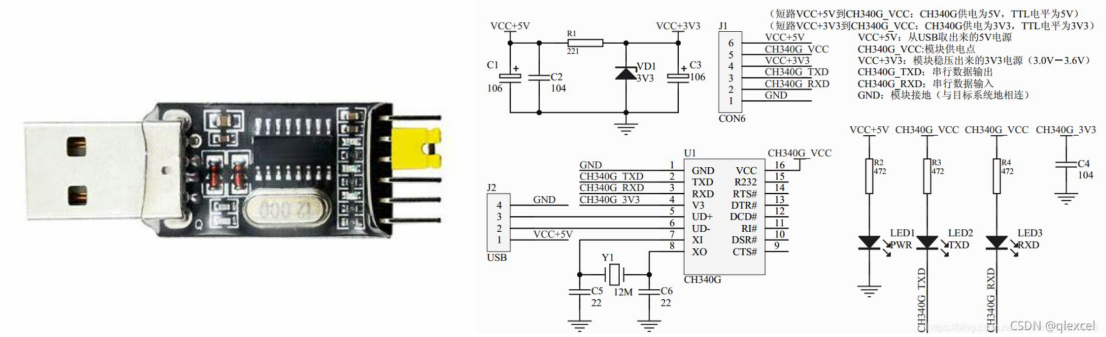


图 7 CH340G USB 转 TTL 烧录芯片模块(左) 原理图(右)

7. **最小系统板：**为单片机提供最基本的电路支持。

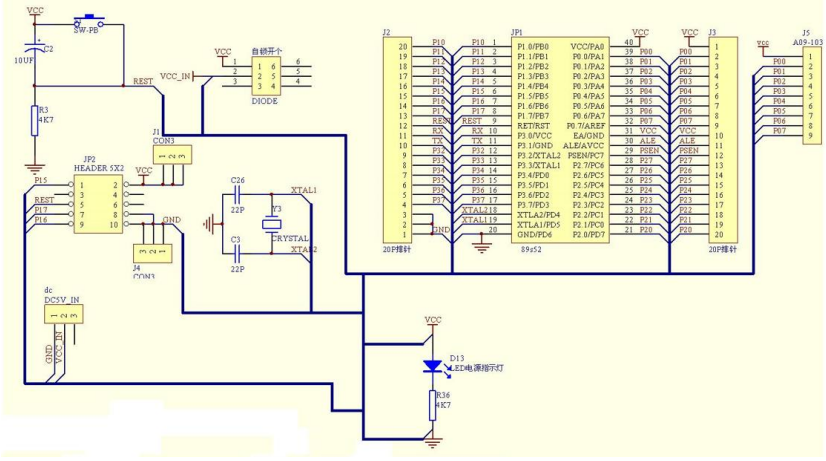


图 8 带锁紧座 51 最小系统板电路图

2. 软件：

1. Keil 4.02

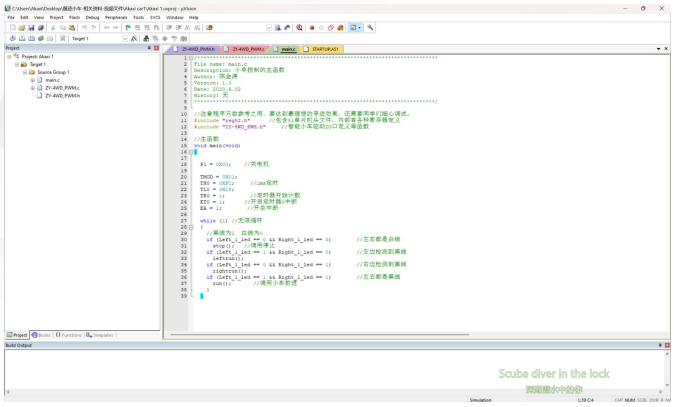


图 9 Keil 程序编译软件

2. STC-ISP 6.68L

图 10 STC-ISP 程序烧录软件

四、实验步骤（或设计过程）

1.车模搭建

按视频逐级搭建小车底板，最小系统板和计算机通信模块，最后将其用杜邦线连接。下面为电机驱动模块 L298N 连接步骤。

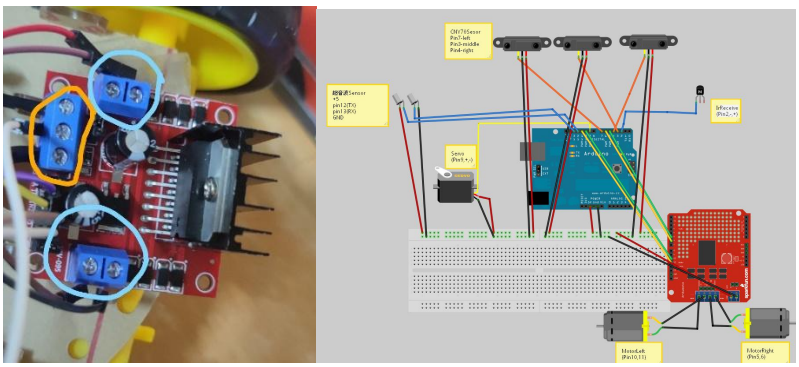


图 11 车模搭建参考接线图

橙色圈出的部分由上至下分别系统板地，系统板 Vcc 和单片机 Vcc。
蓝色圈出的部分连接电机，上方连接左轮，下方连接右轮。
连接方式采用螺丝紧固，将螺丝拧松后插入杜邦线再拧紧螺丝。

1) 小车电机安装；
2) 尾部万向轮固定；
3) 固定电池盒；
4) 编码器和车轮安装；
5) 核心板供电；
6) 驱动与电机和电池盒之间连接；
7) 红外传感器、电机驱动与核心板连接；
8) 单片机程序烧录；
9) 红外传感器固定；

2.成本分析（主要部件成本）

元器件	数量	参考价格
杜邦线	80 根	8.27 元
STC-89C52RC	1 个	3.95 元
红外传感器	2 个	5 元
循迹小车底盘	1 个	6 元
CH340G	1 个	5.26 元
L298N	1 个	2.9 元
最小系统板	1 个	4.9 元
合计	87	36.28 元

五、实验过程原始记录

1.循迹小车实物图

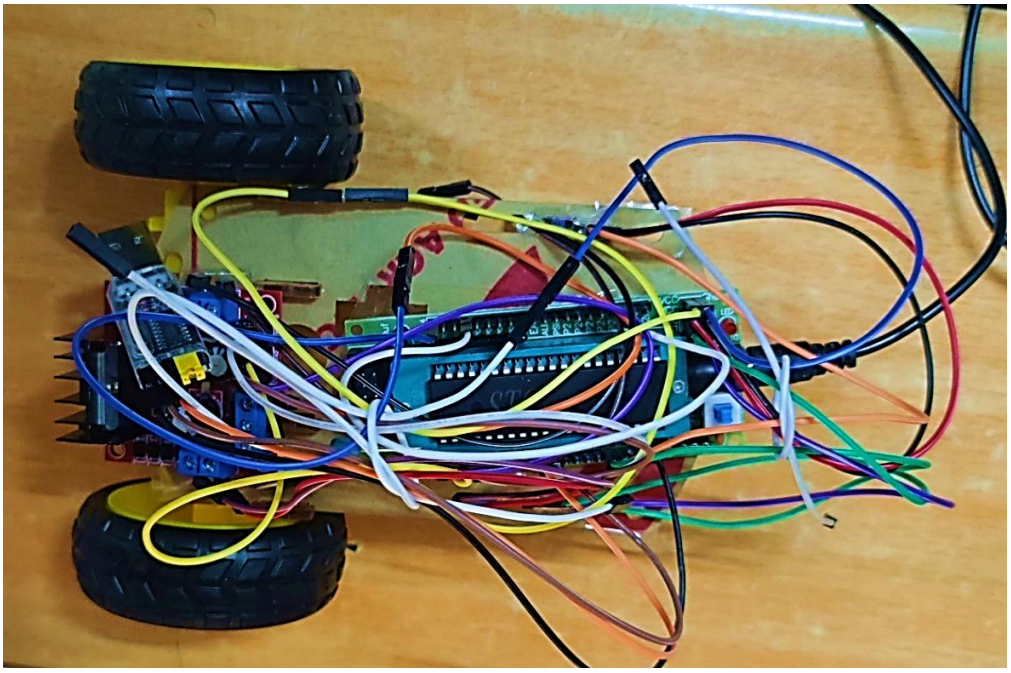


图 12 循迹小车实物图

2.其它具有一定原创性的实验内容

前进转向指示灯

1.我们模拟汽车的转向灯，力图使循迹小车在转弯时，能自动实现转弯灯闪烁。左侧车轮前进时，有 LED 灯 1 亮起；当右侧车轮前进时，有 LED 灯 2 亮起。具体实现方法为从 Vcc 引出两根导线，串联 2K 欧姆电阻和 LED 灯，并连接到开发板引脚上，通过程序控制引脚的高低电平即可实现。

2.同样的方法，使用更多的 LED 灯实现了流水灯。

图 13 模拟汽车转弯灯(左) LED 灯(右)

按钮调速

1.我们力图通过外部可调控手段，避免通过重复调试代码再烧录单片机，采用按钮来控制小车的速度。通过按钮可以调节不同车速，使小车能更加灵活可控。将 P30 至 P32 端口连接按钮后接地实现按钮，再通过软件实现具体功能。

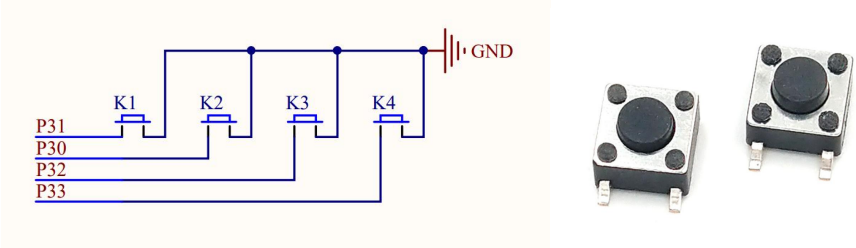


图 14 按钮调速设计图(左) 按钮实物图(右)

六、实验结果及分析（或设计总结）

1.本实验中，如何判断电机是否故障？

将电机正负极直接接入电源，观察电机是否按预期转动，

2.如何判断单片机芯片功能良好或已损坏？

编写简单的测试程序，比如使某引脚输出高电平，改引脚连接二极管后，通过判断二极管是否按预期点亮，判断单片机是否正常。

或将单片机移植到其他可以正常工作的电路中，烧录相同程序，观察电路是否有相同工作状态。

3.如何判断 L298N 是否故障？

在保证单片机芯片和电机芯片正常后判断较为简单，按正常接线图连接以后，确保杜邦线连接牢固。开启电源后检查电机是否正常工作。

4.如何判断单片机最小系统板是否故障？

使用万用表，示波器对电源，各引脚，晶振电路，复位电路进行测试。在确保以上正常运行后，可以通过电脑软件检查通信是否正常。

5.简述 USB-TTL 与单片机最小系统板连接时的接线方法。

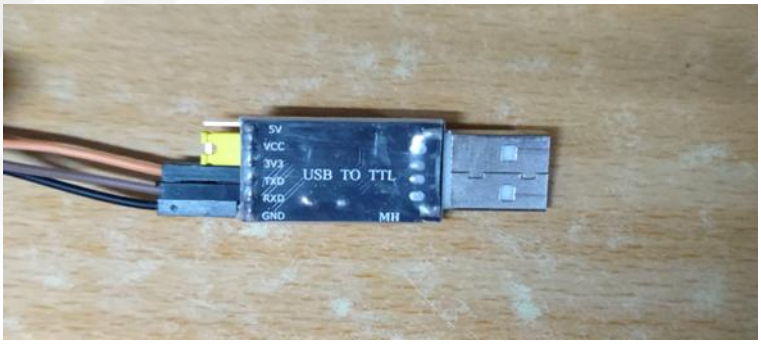


图 15 USB-TTL 图

1. 用跳线帽短接 VCC 和 3V3。
2. 连接 GND。
3. TXD 连接到系统板上的 RXD，RXD 连接到系统板上的 TXD。
4. 在供电能力足够的情况下可以接入 5V，该实验中由于供电能力不足，可以不接入。

6.如何判断红外传感器是否故障？

1. 直接连接电源。
2. 不同距离观察输出指示灯变化。
3. 相同距离调整灵敏度，观察输出指示灯变化。
4. 通过单片机确认输出信号是否正常。

7.问题分析

1. 测试时小车无法正常运行，最小系统板指示灯亮：考虑未给单片机供电，供电后循迹小车正常运行。

2. 调试时，小车在地面上停止，拿起后车轮转动：考虑到问题与距离相关，多次测试红外传感器后，调整了红外传感器的灵敏度，小车正常运行。

3. 测试时小车无法按照程序设想运行：重新烧录时发现并没有烧录最新修改的程序，重新烧录后解决问题。

4. 小车不小心掉落后无法正常运行：检查后发现晶振和多处杜邦线松动，重新紧固后小车正常运行。

项目名称	循迹小车设计实验(2)：程序设计与系统调试		
实验地点	综合实验楼 C601/602	实验时间	第 4 周周一第 9-12 节
实验指导教师	陈晓雷 林峰	成绩	

一、实验目的

1.掌握电机开环调速原理及方法。

2.掌握红外传感器使用方法。

3.完成循迹小车程序编写及整车调试。

二、实验原理（或设计方案）

1.设计要求

实现程序的编译和下载，检验并修正小车控制效果。

2.设计方案

2.1 基本功能描述

红外传感器对黑线和白线进行检测并输出信号，单片机检测输出信号后判断电机对应转动方向速度并输出至 L298N，最后传递给电机实现功能。

当两侧传感器均检测到黑线时，小车直行，两侧车轮转动方向一致。

当两侧传感器均检测到白线时，小车停止。

当左侧传感器检测到黑线右侧检测到白线时，小车左转，两侧车轮转动方向相反，右侧车轮向前。

当右侧传感器检测到黑线右侧检测到白线时，小车左转，两侧车轮转动方向相反，左侧车轮向前。

2.2 程序流程图

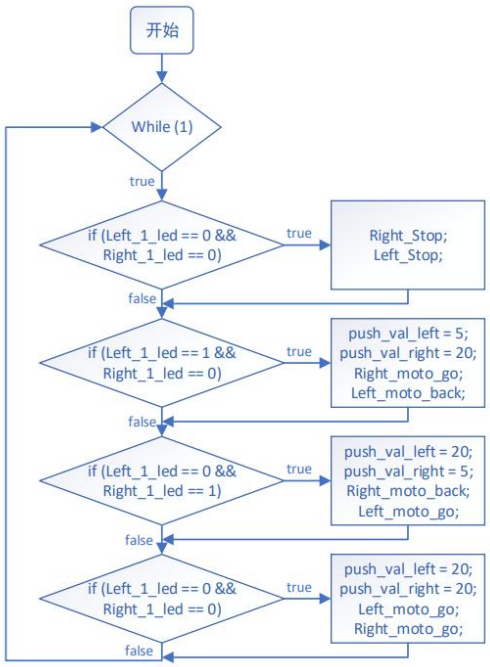


图 1 循迹小车程序流程图

Push_val_left 和 Push_val_right 为变量，控制电机占空比，数值为 N/20。

Left_moto_go 和 Right_moto_go 为宏定义，控制两个车轮前进后退，通过控制两个引脚高低电平实现。

Right_stop 和 Left_Stop 为宏定义，控制车轮是否旋转，通过控制两个引脚高低电平实现。

三、实验仪器设备、材料

1. 硬件及材料

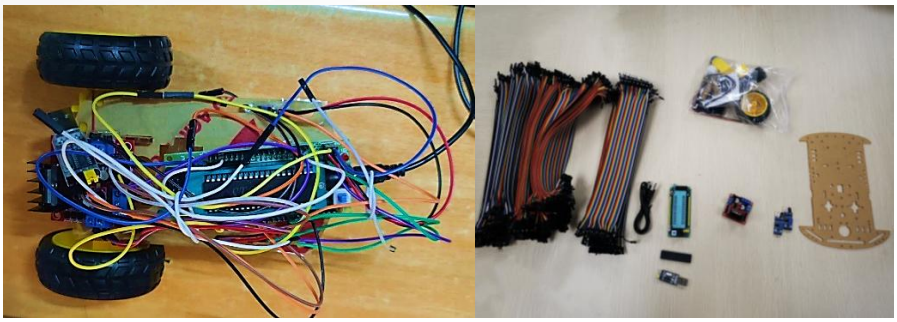


图 2 循迹小车实物图(左) 硬件以及材料(右)

2.软件平台：

1. Keil 4.02

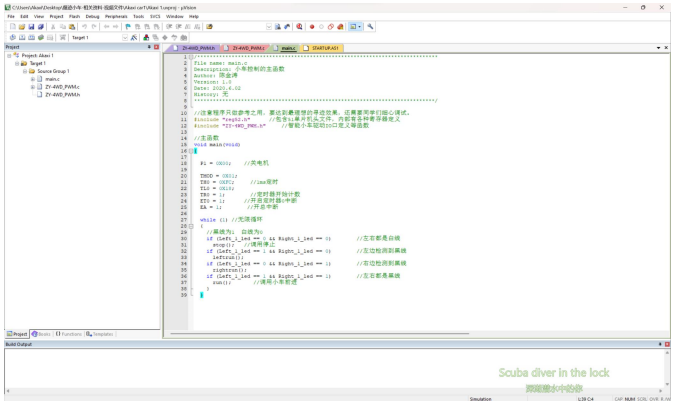


图 3 Keil 程序编译软件

2. STC-ISP 6.68L

图 4 STC-ISP 程序烧录软件

四、实验步骤（或设计过程）

1.程序设计

电机驱动模块：整个模块包括 pwm_out_left_moto()和 pwm_out_right_moto()函数。它们分别负责控制左右电机的转速和方向。函数中，通过比较计时器计数值和预先设定的占空比值 push_val_left 和 push_val_right 来决定电机的转速，同时根据变量 Left_moto_stop 和 Right_moto_stop 来控制电机的停止和启动。

运动控制模块：整个模块包括 run()、leftrun()、rightrun()和 stop()函数，用于控制小车的运动。通过检测两侧的光电传感器输出信号来判断小车的运动方向，并调用相应的电机驱动函数来控制电机的转速和方向。在设计过程中，首先定义了控制左右电机转速和方向的函数 pwm_out_left_moto()和 pwm_out_right_moto()。

通过改变变量 push_val_left 和 push_val_right 来控制电机的转速和方向。同时，变量 Left_moto_stop 和 Right_moto_stop 可以控制电机的停止和启动。

接着，通过定时器中断模块中的 timer0()函数来产生 PWM 信号，控制电机的转速和方向。

最后，通过小车运动控制模块中的 run()、leftrun()、rightrun()和 stop()函数来控制小车的运动方向，根据传感器检测到的黑白线情况，调用相应的电机驱动函数来控制电机的转速和方向，通过这些模块的协同工作，完成了小车的运动控制。

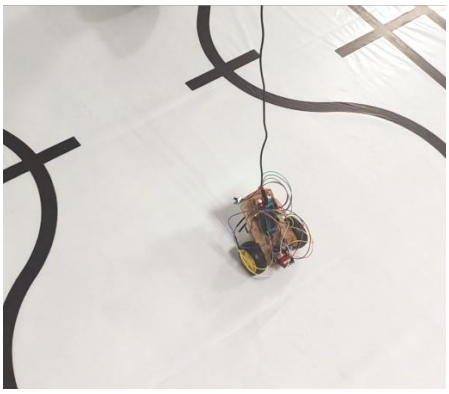


图 5 循迹小车实现功能图

2.系统调试

确认程序基础功能是否正常：在开始调试差速控制程序之前，需要确保程序的基础功能已经正常。例如，程序能够正确识别黑线和白线，能够根据检测到的黑线实现左转、右转、前进和停止等基本动作。

确认控制策略的正确性：差速控制的实现需要一定的控制策略，例如根据检测到的黑线位置计算出左右轮速度的差异等。在调试差速控制程序之前，需要先确认控制策略的正确性，可以通过在程序中增加一些调试输出语句或者使用调试器来检查控制策略是否正确。

增加差速控制代码：在确认程序基础功能和控制策略正确的基础上，可以开始增加差速控制代码。通常情况下，差速控制代码需要在定时器中断服务函数中实现，根据黑线位置计算出左右轮速度的差异，并将差异值作为参数传递给 PWM 输出函数，实现控制左右轮速度的差异。

调试 PWM 输出函数：在增加差速控制代码之后，需要调试 PWM 输出函数，确保左右轮速度的差异能够正确地传递给 PWM 输出函数，并且 PWM 输出函数能够正确地控制左右轮的速度。

五、实验过程原始记录

1.编译程序的 Keil 截图

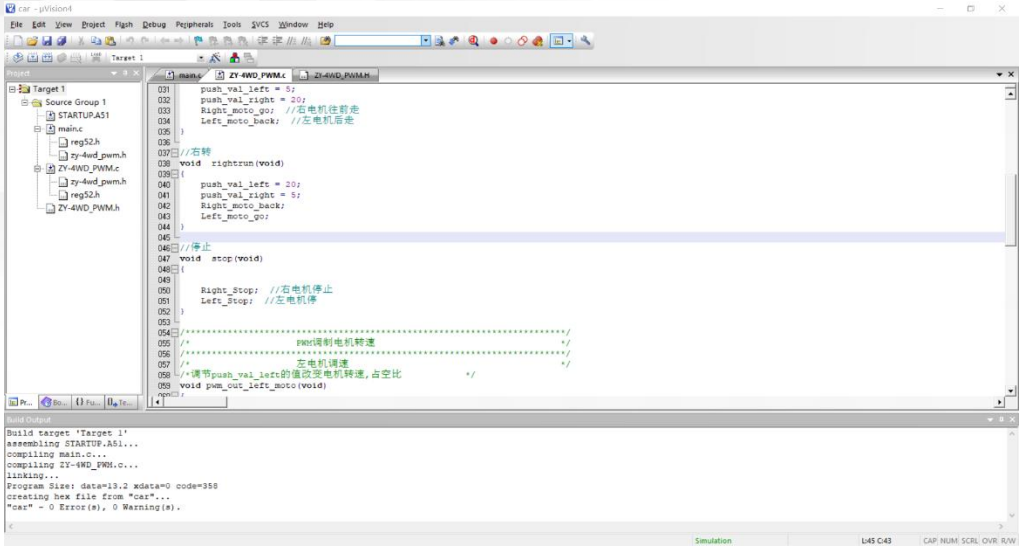


图 6 编译程序的 keil 截图

2.具有一定原创性的实验内容

1. 前进转弯指示灯

车轮是否向前运动，关键部分代码如下。

```
48  sbit LED_left=P2^0;
49  sbit LED_rigt=P2^1;
50 while (1){
51     if (Left_1_led == 0 && Right_1_led == 0){
52         stop();
53         LED_left = 1;
54         LED_rigt = 1;
55     }
56     if (Left_1_led == 0 && Right_1_led == 1){
57         leftrun();
58         LED_left = 1;
59         LED_rigt = 0;
60     }
61     if (Left_1_led == 1 && Right_1_led == 0){
62         rightrun();
63         LED_left = 0;
64         LED_rigt = 1;
65     }
66     if (Left_1_led == 1 && Right_1_led == 1){
67         run();
68         LED_left = 0;
69         LED_rigt = 0;
70     }
71 }
```

图 7 编译程序的 keil 截图

LED_left 和 LED_right 为定义的两个引脚，实际电路中为 P2-0 和 P2-1。小车进行循环检测的时候，决定是否点亮。

2. 流水灯实现

通过编写程序实现流水功能

```
sbit LED_1=P2^0;
sbit LED_2=P2^1;
sbit LED_3=P2^2;
int i=0,j=0;
while (1){
    i ++;
    if (i == 50000){
        i = 0;
        j ++;
    }
    if (j == 0){
        LED_1 = 0;
        LED_2 = 1;
        LED_3 = 1;
    }
    else if (j == 1){
        LED_1 = 1;
        LED_2 = 0;
        LED_3 = 1;
    }
    else if (j == 2){
        LED_1 = 1;
        LED_2 = 1;
        LED_3 = 0;
    }
    else
        j = 0;

    if (Left_1_led == 0 && Right_1_led == 0){
        stop();
    }
}
```

图 8 流水灯程序设计

未使用定时器，点亮时间大概为 450ms 左右。

3. 按钮控制

展示部分代码。

```
sbit KEY1=P3^1;
sbit KEY2=P3^2;
int key_scan(int mode)
{
    static int key=1;

    if(mode)key=1;
    if(key==1 && (KEY1==0&&KEY2==1){
        sleepy(1000);
        key=0;
        if(KEY1==0)
            return 1;
        if(KEY2==0)
            return 2;
    }
    else if(KEY1==1&&KEY2==1){
        key=1;
    }
    return 0;
}
```

图 9 按钮控制速度程序设计 1

按钮按下检测代码，检测是否有按钮被按下，并且考虑了抖动问题。

sleepy 函数为 while 循环，不多赘诉。

```
void main(void)
{
    int key=0;
    float spee=1;
    while(1){
        key=key_scan(0);
        if(key==1)
            break
    }
    while(1){
        key=key_scan(0);
        if(key==1)
            break;
        if (key==2){
            spee = 0.5;
            break;
        }
    }

    while (1){
        if (Left_1_led == 0 && Right_1_led == 0)
            stop();
    }
}
```

图 10 按钮控制速度程序设计 2

主函数前添加两个 while 循环，只有按 1 并且选择速度后，小车才会开始检测。

```
void  leftrun(float timy)
{
    push_val_left = int(5*timy);
    push_val_right = int(20*timy);
    Right_moto_go;
    Left_moto_back;
}
```

图 11 按钮控制速度程序设计 3

为函数添加参数，确保可以调整速度。

六、实验结果及分析（或设计总结）

1.直流电机的开环调速方法总结

电机驱动电路的选型。选择合适的驱动电路对于电机运行稳定、可靠性以及效率至关重要，常见的电机驱动器包括 H-Bridge、MOSFET 或者三极管等，不同的驱动电路注意事项各有所不同。

固定 PWM 输出频率。为了减少噪声干扰以及保证电机的转速稳定，应当将 PWM 输出的频率和占空比写死在代码中，避免在运行过程中改变这些参数。

实时监测电机状态并进行错误处理。软件实施控制，会存在跑飞和误差等问题，应该编写代码监测电机的状态并进行错误处理。

2.如何控制电机正反转？

通过以下函数控制电机正反转：

Left_moto_go()：左电机正转；

Left_moto_back()：左电机反转；

Right_moto_go()：右电机正转；

Right_moto_back()：右电机反转；

Left_Stop()：停止左电机；

Right_Stop()：停止右电机。

3.如何通过单片机实现对电机转速的调节？已掌握了哪些方法？

通过单片机可以使用 PWM（脉冲宽度调制）信号来实现对电机转速的调节。

具体实现方法如下：

将单片机的定时器功能与 PWM 合成，使定时器每隔一段时间产生一个定时脉冲，然后根据 PWM 信号的高低电平来调节电机转速的快慢，达到调节电机转速的目的。

将 PWM 信号直接作为电机驱动信号，通过改变 PWM 信号的占空比来控制电机转速。

已掌握的方法有：**使用定时器产生 PWM 信号，通过改变 PWM 信号的占空比来控制电机转速。**

4.问题分析

1. 测试时小车无法正常运行，最小系统板指示灯亮：考虑未给单片机供电，供电后循迹小车正常运行。

2. 调试时，小车在地面上停止，拿起后车轮转动：考虑到问题与距离相关，多次测试红外传感器后，调整了红外传感器的灵敏度，小车正常运行。

3. 测试时小车无法按照程序设想运行：重新烧录时发现并没有烧录最新修改的程序，重新烧录后解决问题。

4. 小车不小心掉落后无法正常运行：检查后发现晶振和多处杜邦线松动，重新紧固后小车正常运行。

5.具有原创性的实验设计内容

1.指示车轮是否向前或转弯的 LED 指示灯。

2. LED 流水灯。

3.按钮控制小车开启和调整速度。

图 12 原创性实验设计

参考文献

[1] 基于 51 单片机的循迹小车（初学者必备!!!）_51 循迹小车_·Cx330·的博客-CSDN 博客

[2] 2 个红外传感器循迹原理_智能循迹小车_weixin_39682673 的博客-CSDN 博客

[3] 【mcuclub】STC89C52 单片机最小系统讲解_单片机俱乐部--官方的博客-CSDN 博客

[4] L298N 电机驱动模块详解_l298n 电机驱动模块介绍_楸壳的博客-CSDN 博客

[5] 简单的使用 CH340g 来下载烧录或调试的具体操作步骤_ch340g 烧录_淡水咸鱼 00 的博客-CSDN 博客

[6] Keil 使用教程（详解）_莫余的博客-CSDN 博客

【附件】设计程序

1. 循迹小车主函数 main.c

```
/****************************************
**********************************
File name: main.c
Description: 小车控制的主函数
Author: 胡润笙、王忠全
Version: 1.0
Date: 2023.05.01
History: 无
****************************************
**********************************/

#include "reg52.h"            //包含 51 单片机头文件，内部有各种寄存器定义
#include "ZY-4WD_PWM.h"       //智能小车驱动 IO 口定义等函数

int key_scan(int mode)
{
    static int key=1;

    if(mode)key=1;
    if(key==1 && (KEY1==0&&KEY2==1){
        sleepy(1000);
        key=0;
        if(KEY1==0)
            return 1;
        if(KEY2==0)
            return 2;
    }
    else if(KEY1==1&&KEY2==1){
        key=1;
    }
    return 0;
}

void main(void)
{
    int key=0 , i=0 , j=0;
    float speed=1;
    while(1){
        key=key_scan(0);
        if(key==1)
            break
    }
    while(1){
        key=key_scan(0);
        if(key==1)
            break;
        if (key==2){
            speed = 0.5;
            break;
        }
    }

    while (1){
        i++;
        if (i == 50000){
            j++;
            i = 0;
        }
        if (j == 0){
            LED_1 = 0;
            LED_2 = 1;
            LED_3 = 1;
        }
        else if(j == 1){
            LED_1 = 1;
            LED_2 = 0;
            LED_3 = 1;
        }
        else if(j == 2){
            LED_1 = 1;
            LED_2 = 1;
            LED_3 = 0;
        }
        else
            j = 0;

        if (Left_1_led == 0 && Right_1_led == 0){
            stop();
            LED_left = 1;
            LED_right = 1;
        }
        if (Left_1_led == 1 && Right_1_led == 0){
            lefturn(speed);
```

```
            LED_left = 1;
            LED_right = 0;
        }
        if (Left_1_led == 0 && Right_1_led ==
1){
            righturn(speed);
            LED_left = 0;
            LED_right = 1;
        }
        if (Left_1_led == 1 && Right_1_led ==
1){
            run(speed);
            LED_left = 0;
            LED_right = 0;
        }
    }
}
```

2. 电机驱动函数 ZY-4WD_PWM.c

```
/**************************************
**********************************
File name: ZY-4WD_PWM.c
Description: 用于电机驱动，接线见
ZY-4WD_PWM.h
Author: 胡润笙、王忠全
Version: 1.0
Date: 2023.05.01
History: 无
**************************************
**********************************/
#include "ZY-4WD_PWM.h"

unsigned char pwm_val_left = 0;        //变量定义
（定时器中计数）
unsigned char push_val_left = 0;  // 左电机占空
比 N/20
unsigned char pwm_val_right = 0; //变量定义（定
时器中计数）
unsigned char push_val_right = 0; // 右电机占空
比 N/20
bit Right_moto_stop = 1;                    //
停下置 0
bit Left_moto_stop = 1;

/**************************************
**********************************/
//全速前进
void   run(float timy)
{
    push_val_left = 20;  //速度调节变量 0-20。。。
0 最小，20 最大
    push_val_right = 20;
    Left_moto_go;    //左电机往前走
    Right_moto_go;   //右电机往前走
}

void   leftrun(float timy)
{
    push_val_left = int(5*timy);
    push_val_right = int(20*timy);
    Right_moto_go;
    Left_moto_back;
}

//右转
void   rightrun(float timy)
{
    push_val_left = int(20*timy);
    push_val_right = int(5*timy);
    Right_moto_back;
    Left_moto_go;
}

//停止
void   stop(void)
{

    Right_Stop;   //右电机停止
    Left_Stop;   //左电机停止
}

/**************************************
**********************************/
/*                          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_out_left_moto(void)
{
    if (Left_moto_stop)
    {
        if (pwm_val_left <= push_val_left)
        {
            Left_moto_go;
        }
        else
        {
            Left_Stop;
        }
        if (pwm_val_left >= 20)
            pwm_val_left = 0;
    }
    else
    {
            Left_Stop;
    }
}
/****************************************
*************************/
/*                                  右电机调速
*/
void pwm_out_right_moto(void)
{
    if (Right_moto_stop)
    {
        if (pwm_val_right <= push_val_right)
        {
            Right_moto_go;
        }
        else
        {
            Right_Stop;
        }
        if (pwm_val_right >= 20)
            pwm_val_right = 0;
    }
    else
    {
            Right_Stop;
    }
}

/****************************************
***********/
///*TIMER0 中断服务子函数产生 PWM 信号*/
void timer0()interrupt 1    using 2
{
    TH0 = 0XFc;     //1Ms 定时
    TL0 = 0X18;
    pwm_val_left++;
    pwm_val_right++;
    pwm_out_left_moto();
    pwm_out_right_moto();
}
```

3. 电机驱动文件 ZY-4WD_PWM.h

```
/****************************************
*********************************
File name: ZY-4WD_PWM.c
Description: 声明一些接口，声明一些函数，简单
的宏定义控制电机转向
Author: 胡润笙、王忠全
Version: 1.0
Date: 2023.05.01
History: 无
L298N:
IN1--接到--实验板上的 P1.0
IN2--接到--实验板上的 P1.1
IN3--接到--实验板上的 P1.2
IN4--接到--实验板上的 P1.3
传感器:
左 OUT1--接到--实验板上的 P3.2
OUT3--接到--实验板上的 P3.4
右 OUT2--接到--实验板上的 P3.3
OUT4--接到--实验板上的 P3.5
```

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

#ifndef _ZY_4WD_PWM_H_
#define _ZY_4WD_PWM_H_
#include "reg52.h"

#define u8    unsigned char
#define u16 unsigned int

//定义小车驱动模块输入 IO 口

sbit L298N_IN1 = P1 ^ 0;
sbit L298N_IN2 = P1 ^ 1;
sbit L298N_IN3 = P1 ^ 2;
sbit L298N_IN4 = P1 ^ 3;

// 按键
sbit KEY1=P3^1;
sbit KEY2=P3^2;

// 灯
sbit LED_left = P2^6;
sbit LED_right = P2^7;
// 流水灯
sbit LED_1 = P2^5;
sbit LED_2 = P2^4;
sbit LED_3 = P2^3;


sbit Left_1_led = P3^7;   //左传感器

sbit Right_1_led = P3^6; //右传感器



#define Right_moto_go
    L298N_IN1=1,L298N_IN2=0;   //左电机向前走
#define                          Right_moto_back
    L298N_IN1=0,L298N_IN2=1;    //左边电机向后转
#define Right_Stop
    L298N_IN1=0,L298N_IN2=0;   //左边电机停转
#define Left_moto_go
    L298N_IN3=0,L298N_IN4=1;    //右边电机向前走
#define Left_moto_back
    L298N_IN3=1,L298N_IN4=0;    //右边电机向后走
#define Left_Stop
    L298N_IN3=0,L298N_IN4=0;   //右边电机停转


void   run(void);                    //向前跑
void   leftrun(void);          //左转
void   rightrun(void);         //右转
void   stop(void);                   //停下
void pwm_out_left_moto(void);        //控制左电机 PWM
void pwm_out_right_moto(void); // 控制右电机 PWM


#endif
```