智能小车自动避障测控系统设计与制作

摘 要

高移动性，高负载性和已于控制是现代智能移动机器人的主要优势，这些新兴科技能够在很多场合帮助人类进行赈灾、探索、救援等任务，他们的应用场合也很多样化，像室内服务、空间勘探、军事、娱乐、以及医疗服务等等。本次设计根据 STM32 微型处理器为核心，以电机驱动模块、红外避障模块、红外循迹模块红外遥控模块等设计制作了一台智能小车。使用双桥步进电机作为驱动，在红外避障模块感知到前方有障碍物时，可以对小车发出转向的命令，做到自动避开障碍物的功能；同时，使用红外遥控器还可实现小车不同模式下的切换，并可控制小车的前进、后退、左右转向的基本功能。

关键词：STM32 微型处理器；红外遥控；步进电机；自动避障

1 前言

随着经济的快速发展，人们生活水平不断提高，人们对生活质量的要求也越来越高，随之提高的，还有人们对智能化产品的接受程度。从工业革命开始，移动工具就逐渐开始给人类在各方面带来帮助，在生产方面以及生产过程的自动化和现代化也在不断提高。随着要求的提高，人们不仅仅需要能够自由操控的机器，还希望它们能够自行运作。随之产生的就是智能的移动机器人。　　对于这些技术，主要由：信息技术，传感器技术，机器人技术等的综合运用。新技术的应用又促进了其他技术的发展，像是，生产速度以及精确性和良品率等等。在移动工具中汽车工业是最早发展的工业之一，在应用在不同领域时逐渐发展。随着传感器技术的发展，逐渐使得智能移动工具具备视野并能够执行一些工作，为了检测身边环境和周边物体，在机器人身上安装了光学传感器、触觉传感器、红外传感器和超声波传感器等，使得机器人的工作灵敏度被极其大地改善了，即便是复杂的工作也能被更快速充分地完成。伴随传感器技术的更深层次的发展，机器人的应用范围也将变得更加广泛。在现在，高移动高负载和易于控制是现在移动机器人的主要优势，在现在的研究中注重于提高这些机器人的性能以及可靠性并使其能在环境更恶劣的地方工作，如赈灾救援，矿井勘探，深海科考等等。

1.1 本设计的目的、意义及应达到的技术要求

随着科技和信息技术的不断进步，各种机械设备也在不断地发展，已经呈现出了智能化地发展形势，随着控制技术、动力技术、传感器技术的发展促进了智能移动机器人的飞速发展，使得智能移动机器人变得更加稳定和可靠。人们逐渐认识到智能移动机器人带来的价值，无论是救援行动、搜查工作还是在民用领域均可发挥重要的作用，可突破人类视野的局限性，代替人类在各种各样或极限或危险的环境工作，在抗震救灾时人员或普通交通工具无法抵达时，在外太空人类未知的领域时，都可以采集数据、执行巡逻、运输物资等。目前，各国都在研发更好，更先进的自主移动机器人。本次课题通过 STM32 为处理核心研究自动避障小车如何实现室内自动识别障碍并进行回避动作，通过改变各种

参数，实现不同距离或者速度并根据 stm32 单片机以及红外自动避障系统实现车子向前，向后以及提高速度等动作。

1.2 本设计在国内外的发展概况及存在的问题

机器人和无人机的技术进步对所有人都是显而易见的，各种在工场流水线作业的机器或是集成了世界最尖端技术智能机器人的相关文章都出现在报纸的头条新闻上。当然，在有关“智能”这一领域，传感器有关技术的使用就是不可避开的。以目前情况来看，在未来，智能机器人领域发展的潜力巨大，相对的，这方面的技术要求会越来越高，根据预测 2019-2024 年，全球的服务机器人销售额将会更上一层楼，预计 2024 年全球服务机器人销售额将会达到 170 亿美元。在当前技术飞速发展的智能机器人领域中，相关军事和民用领域对其性能和各种指标的要求将会越发严苛，这就要求现在的智能机器人有着很高的性能。

目前在国内有二零一零年国家的“863”计划联合了以浙江大学为主体自研发的‘高性能四足仿生机器人’项目，还有国内宇通工作室研发的‘绝影’系列智能机器人，动作自由顺滑，反应也非常精准快速，面对复杂的地形和环境可以表现出很强的适应力。

此外，在智能机器人领域中的翘楚波士顿动力公司也是移动机器人研发中的领军人物，这家公司怀着一腔热情攻克了很多严苛的理论难关，在结构、液压驱动以及机电一体化等方面地出色设计，给予了智能机器很大地操作空间，超快地响应速度及驱动力，是世界独一档的，这家公司在复杂的机械构造中佐以最先进的液压系统，再加上自平衡的全身控制，将搭载最尖端的智能传感器还有作用于各种不同环境情况下的算法的高智能机器人和次世代软件融合组装在一起，制造出令人惊艳的智能机器人。

当然，现在国内还面临着许多技术问题，像液压控制，路线生成控制等问题还需要继续研究，如何才能让机器人像动物一样灵活的运动，这需要的不仅仅是协调性和适应性，还需要的是极其强大的驱动能力，因为电机驱动的问题主要是功率体积比太小，需要的功耗太大，这样就会使得机器人的体积太过于庞大。此外，抛开控制技术方面不谈，在基础的底层技术上还包含机器人的整体架构问题，就像机械部件构造方面的设计、液压器以及液压油流向的管理、传感器以及相关物体或环境的识别配置方面、能源要以何种形式何种介质来提供、散热及散热系统的构造。复杂精密程度来看，一个智能移动机器人的相关构造和原理就是一项系统工学，任何微小的设计失误或者不合理之处都有可能无限放大从而造成系统的失败或故障。

1.3 本设计应解决的主要问题

在小车行驶过程中，不进行人工干预，小车通过各种内置传感器的作用下获取各种现场的信息，车子的控制系统根据这些信号发出控制指令并使小车做出回避动作。

（1）利用小车的红外传感器进行近距离障碍物的识别，并调整电阻的灵敏度使系统稳定运行并控制方向的变化。

（2）通过红外遥控操控系统实现小车的远程操纵

（3）使直流电机的每个相绕组遵从适当的顺序使其分别通电，并且对电机驱动模块的参数分别调节，区分正转，反转，制动的状态。

（4）在软硬件层面调试即程序代码的运行和烧录以及硬件的组装使各个模块正确运行并能互相协调工作。

2 本设计

（1） 在了解 stm32 单片机的原理，通过一些编程语言，将避障，遥控，循迹功能结合并通过硬件的适配完成一套智能小车的全部系统。

（2） 本设计由 stm32 主板，扩展板，电机驱动模块和电机为主要部件实现小车的基本驱动功能。其余拓展部件为循迹模块，红外避障模块，红外遥控电路等组成。

2.1 设计原理

本设计以 stm32 单片机器为主体，加上带有电源电路，红外遥控电路的扩展板以及 电机驱动模块组成。电机驱动通过连接到拓展板的信号线和连接到电机的电机线驱动小车的左转，右转，前进和后退。小车也可通过遥控器发射红外信号，由红外遥控电路接收，可分别选择 3 种模式：避障模式，循迹模式和遥控模式。

在避障模式下由红外避障模块的 OUT 输出引脚探测前方有无障碍物，如果有，则红外避障模块上的蓝色指示灯亮起，小车自动转向。

在循迹模式下，有三个循迹模块组成的传感器安装在小车前端，由于颜色较深的地方对红外光的反射率比较小，所以当地面的颜色不是深色而是相对较浅的时候，模块发出的红外光多数被反弹回传感器，于是传感器就可以检测到并输出信号，小车就可以自动沿着轨迹走。

2.2 方案选择

2.2.1 核心控制器

在核心控制器方面，我选择了 STM32 作为控制处理核心，这款处理芯片具有诸多优点，能够很好地根据功能添加设计外围系统框架并且整合成为一个综合系统；STM32 处理器功能极其强大，处理器中央处理器基于 ARM 架构，ARM 体系结构内核为特别要求高处理能力、低消耗的嵌入式应用为基础进行研发设计，于此同时处理器中还集成了一系列周边设备：1 微米的双 12 位数字转化器，4Mbps 的异步收发传输器，18Mbps 的串行外设接口等，这使得这颗处理器在能耗控制和集成度方面以及运算速度均有着不错的表现。此外，在其芯片上还集成了 32 至 512KB 的闪存存储器。6 至 64KB 的静态随机存取存储器当中包含有多个接口，为小车的功能打下基础。

2.2.2 电机驱动模块

首先，小车的动力核心源于由电机驱动模块驱动的两对电机，本台小车选用直流电机，直流电机重量轻，体积小，动力强大，使用和装配起来方便简单；该驱动芯片为 L298H 双 H 桥直流电机电机驱动芯片在温度 T=75° C 时功耗为

20W，该驱动板可驱动 2 路直流电机，使能端 ENA，ENB 为高电平时有效，控制方式及直流电机状态图如表 2.1 所示：表 2.1 直流电机状态图

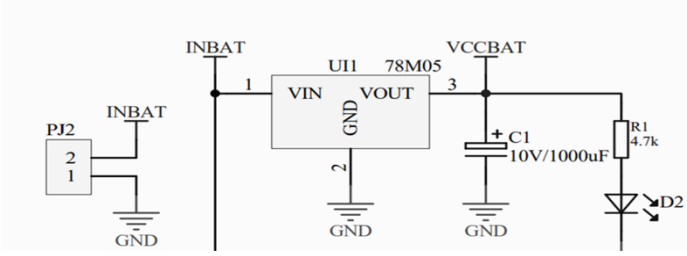
ENA	IN1	IN2	直流电机状态
0	×	×	停止
1	0	0	制动
1	0	1	正转
1	1	0	反转
1	1	1	制动

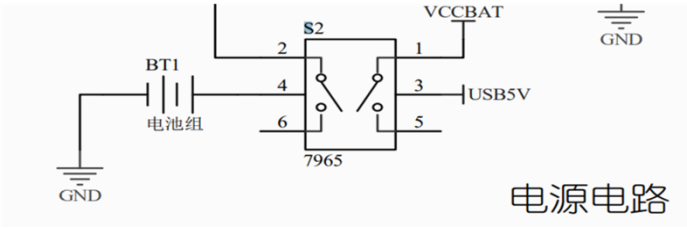
电机转动的本质是对 PMW（Pulse width modulation wave）波的控制和输出。PWM 调速是目前比较主流的电机调速方式。电机脉冲的信号有最大值、最小值和频率。我要使用的信号的周期为 20ms。电机在脉冲宽度为 0.5ms 时处于中间位置即到最大转动角度与最小转动角度的量相等。当改变脉冲的长短就会改变电机的转动角度例如：当 t=1.0ms 时电机转动 45°，t=1.5ms 时电机转动 90°。因此，如果要进行直流电机的基于脉冲宽度调制的速度，则首先要设置 IN1 和 IN2，明确电动机的旋转方向，然后将脉冲波输出到使能端上，并更改脉冲波的占空比，即调节在在使能端 EN1 和 EN2 上的脉冲波的输入。本质上就是改变施加在电动机两测的电压以实现速度的控制。

2.2.3 电源模块

电源电路由图 2.2 所示；由两节 18650 锂电池供电，电池每节电压为 3.7V，总电压为 7.4V。其中 S2 为电源开关， PJ2 为电机驱动模块供电，78M05 为稳压芯片，可以借助在引脚上的电位器，利用串联分压的原理改变输出电压的大小，在之后的原部件中还会看到。

图 2.2

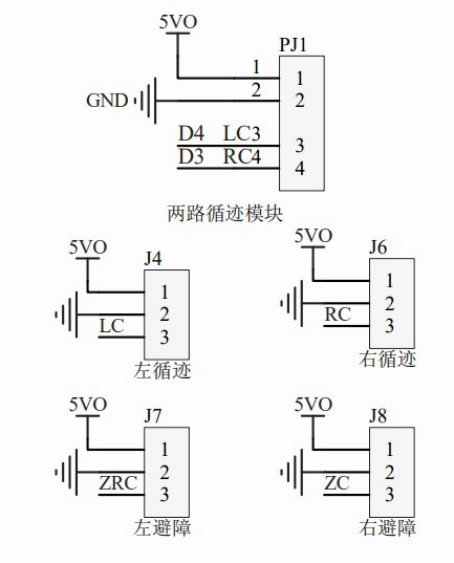




2.2.4 红外循迹与避障模块

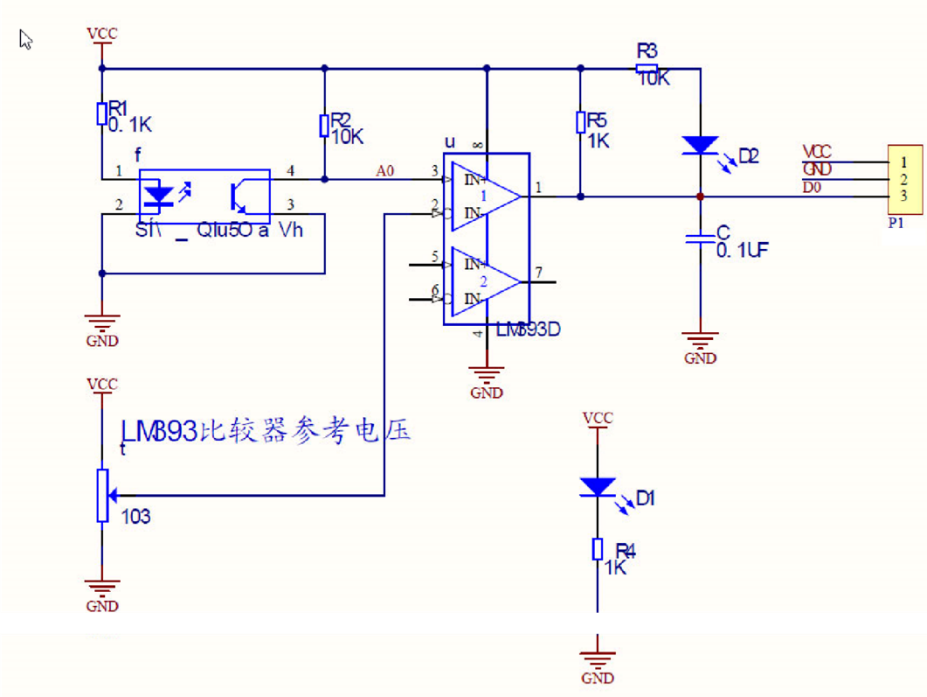
循迹模块是这俩小车的一个重要组件，本系统使用两个装在小车两侧的具有发射和接收红外光束的传感器，由于地上的黑线会吸收大部分的红外光，当发出的红外光没有照射到黑线时，模块内接收管导通，输出低电平，当模块发出的红外光照射到黑线时，由于反射回来的光线减少，接收管不会导通，于是输出高电平，处理器通过判断接收到的信号来控制小车方向。系统原理图中 LC 为左循迹模块，接在 ARDUUINO 转接板的 D4 接口上；RC 为右循迹模块，接在 D3 端口；ZRC 为左避障，接在 D5 端口上；ZC 为右避障，接在 D6 端口上。因该注意 VCC 应该接在 5V 的接口，GND 对应
GND 接口。如图 2.3 所示。

图 2.3 循迹与避障模块电路



由图 2.4 原理图；下面的 R103 为可变电阻，调节可变电阻可改变输出到 LM693D 比较器的电压；探头根据接收到红外光的强弱而输出不同的电压到比较器；比较器根据两边电压的大小判断输出哪边的电压。

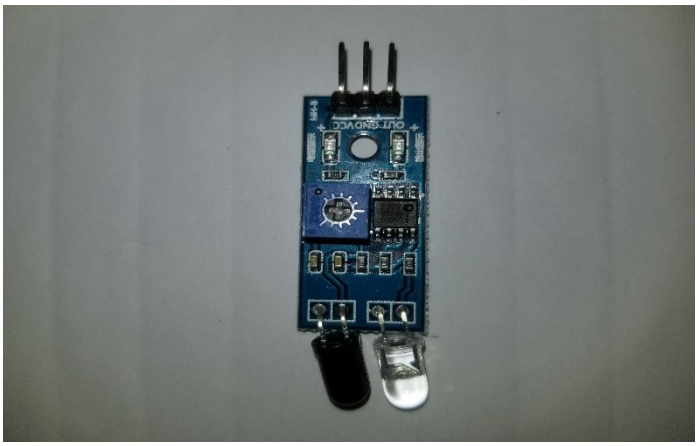
图 2.4 避障模块原理图



如图 2.5 红外避障模块所示，左上方为电源指示灯，当接通点源时绿色灯光会常亮，右上方为输出指示灯，当接收到低电平即感知到障碍物时会亮起，中间方形物体为可调电阻，如果灵敏度越高则可感应距离越远；最远为 5cm。

此外，红外线避障是短距离感应，如果想要长距离感应则可以采用超声波感应。

图 2.5 红外避障模块

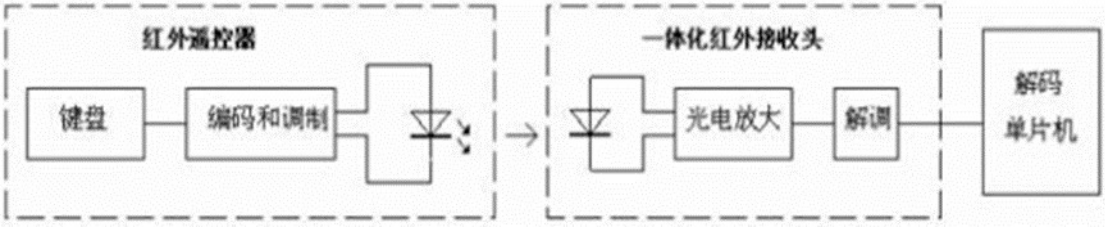


2.2.5 红外遥控模块

目前被应用最为广泛的一种信息传输或者控制系统就是红外遥控系统了，红外遥控装置具有非常多优势，像是装置小巧，能源消耗低，使用方便，易于制造等，在多种场合应用非常广泛。红外发光管和红外接收头两个组件构成了红外遥控系统；红外发光管能产生 940 纳米波长的近红外线。红外接收头的工作电压在 4.8-5.3V；工作电流为

1.7-2.7mA。通常来说，红外发射装置的功耗都比较低，这意味着当没有遥控按钮按下时，它们几乎处于不消耗任何电能的低功耗状态，而当遥控器的按钮按下时，透明就会立刻唤醒并发射相应红外命令。就本系统而言简单来说，利用红外进行数据的传输的实质其实就是调制与解调二进制信号；这个系统利用脉冲调制（PPM）的原理进行信号的调制，二进制的信号在发送端被转换成特定频率的脉冲组合，再让红外发射管以发射红外光的形式将信息发出，高频滤波电路被集成在接收端中，这可以启到对信号进行整流和滤波的作用，可以用来过滤信号中的无用频率，接着将红外发射管发射出的红外光信号处理成电信号，并经过放大，滤波等步骤后，接收端就会发出经转换过变成二进制的接收到的信号，再通过解码程序将信号进行处理，单片机就能够知晓操控者按的是遥控器上的哪一个位置的按钮，并发出相应的操控命令。

图 2.6 红外遥控系统框图



2.2.6 超声波避障模块

此种系统具有操作容易，成本低廉，精度较高等优点。此超声波模块采用带有舵机摇头的安装方案；超声波具有比较好的方向性，此种超声波避障方法是利用超声波在碰到障碍物会反射回来的性质，勘测其在空气中传播后碰到障碍物并反射回接收头所花费的时间，根据超声波的已知速度并测算发射和接收之间所需要的时间，以此来计算

出发射位置以及碰到反射物体所在位置的实际间隔，此种原理和雷达测距的原理比较相似。测量距离的公式为

L=C×T，H=L*cos θ，　其中 L 为测距得出的超声波的传播距离；C 为超声波在空气中传播速度（C=344m/s，空气温度为 20°）；T 为发射到接收到反射回来的超声波的时间差的一半，θ 为接收到声波的入射角，H 为传感器到被测物体之间的距离，此外，由于超声波在空气中的传播速度于空气的温度和密度有着紧密联系，超声波的速度越快，即代表周围空气的温度越高，亦或是周围空气的密度越大，所以当要求测量精度较高时，需要把测量时的空气温度也考

虑进去。此模块工作启动时采用 IO 触发来进行测距，在其工作时需要至少 10 微秒的高电平信号，当工作时，系统会发射 8 个 40 千赫兹的脉冲波并自发探测是否有信号返还，一旦探测到有信号返回时，回响信号就会通过 IO 口输出，此回响信号的脉冲宽度和所要测得的距离成正比，且超声波信号从发射到返回所检测到的时间即为这个高电平信号延续的时间，计算公式为 L=（高电平持续时间*344）/2。当然，系统测出的距离可能会有误差，这与当时周围的环境例如灰尘、空气湿度等有一定影响，所以使用时应该尽可能保证超声波模块于测距物体尽量保持在同一水平线上

2.3 程序设计 2.3.1 电机程序设计

首先，小车的电机驱动主程序中包含一个延时函数和一个引擎函数分别命名为 delay.h 和 motor.h。

int main(void) { delay_init();

TIM4_PWM_Init(7199,0); //初始化 PWM while(1) {

ZYSTM32_back(70,1000);//速度为 70，后退 1s

ZYSTM32_brake(500);//停止 0.5S

ZYSTM32_run(70,1000);//速度为 70，前进 1S

ZYSTM32_brake(500);//停止 0.5S

//

ZYSTM32_Left(70,1000);//速度为 70，左转 1S

ZYSTM32_Right(70,1000);//速度为 70，右转 1S

////

ZYSTM32_Spin_Right(70,1000);//速度为 70，向右旋转 2S

ZYSTM32_Spin_Left(70,1000);//速度为 70，向左旋转 2S ZYSTM32_brake(500);//停止 0.5S

图 2.7 电机程序示例

```
int main(void)
{
  delay_init();
  TIM4_PWM_Init(7199,0);
   while(1)
   {
      ZYSTM32_back(70,1000);
      ZYSTM32_brake(500);
      ZYSTM32_run(70,3000);
      ZYSTM32_brake(500);
//
        ZYSTM32_Left(70,1000);
        ZYSTM32_Right(70,1000);
////
        ZYSTM32_Spin_Right(70,1000);
        ZYSTM32_Spin_Left(70,1000);
        ZYSTM32_brake(500);
   }
}
```

#define LEFT_MOTOR_GO GPIO_Pin_7 //定义端口

#define LEFT_MOTOR_GO_GPIO GPIOB

#defineLEFT_MOTOR_GO_SET GPIO_SetBits(LEFT_MOTOR_GO_GPIO,LEFT_MOTOR_GO) //置高电平置 0 #defineLEFT_MOTOR_GO_RESET GPIO_ResetBits(LEFT_MOTOR_GO_GPIO,LEFT_MOTOR_GO) //置低电平置 1 图 2.8 电机引脚程序示例

```
#define LEFT_MOTOR_GO              GPIO_Pin_7
#define LEFT_MOTOR_GO_GPIO         GPIOB
#define LEFT_MOTOR_GO_SET          GPIO_SetBits(LEFT_MOTOR_GO_GPIO,LEFT_MOTOR_GO)
#define LEFT_MOTOR_GO_RESET        GPIO_ResetBits(LEFT_MOTOR_GO_GPIO,LEFT_MOTOR_GO)

#define LEFT_MOTOR_PWM             GPIO_Pin_8
#define LEFT_MOTOR_PWM_GPIO        GPIOB
#define LEFT_MOTOR_PWM_SET         GPIO_SetBits(LEFT_MOTOR_PWM_GPIO,LEFT_MOTOR_PWM)
#define LEFT_MOTOR_PWM_RESET       GPIO_ResetBits(LEFT_MOTOR_PWM_GPIO,LEFT_MOTOR_PWM)
```

```
51  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);
52  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOB , ENABLE);
53
54  GPIO_InitStructure.GPIO_Pin = LEFT_MOTOR_GO;
55  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
56  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
57  GPIO_Init(LEFT_MOTOR_GO_GPIO, &GPIO_InitStructure);
```

RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);// 对 APB1 的定时器 4 设置时钟使能

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOB , ENABLE); //使能 GPIO 外设时钟使能

GPIO_InitStructure.GPIO_Pin = LEFT_MOTOR_GO; // 左电机方向控制 PB7

GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //电机复用 50MHz

GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

GPIO_Init(LEFT_MOTOR_GO_GPIO, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Pin = LEFT_MOTOR_PWM; //左电机 PWM 控制 PB8

GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; //复用推挽输出

GPIO_Init(LEFT_MOTOR_PWM_GPIO, &GPIO_InitStructure);

2.3.2 循迹程序设计

对于循迹模块，要设定好当左右循迹分别探测到黑线时小车的行动逻辑；比如，当小车右侧探测到黑线而左侧没有探测到时，小车将右转。图 2.9 循迹程序示例

```
void SearchRun(void)
{
  if(SEARCH_L_IO == WHITE_AREA && SEARCH_R_IO == WHITE_AREA)
    ctrl_comm = COMM_UP;
  else if (SEARCH_L_IO == BLACK_AREA && SEARCH_R_IO == WHITE_AREA)
    ctrl_comm = COMM_RIGHT;
  else if (SEARCH_R_IO == BLACK_AREA & SEARCH_L_IO == WHITE_AREA)
    ctrl_comm = COMM_LEFT;
  else ctrl_comm = COMM_STOP;

  if(ctrl_comm_last != ctrl_comm)
  {
      ctrl_comm_last = ctrl_comm;
      switch(ctrl_comm)
        {
          case COMM_UP:    ZYSTM32_run(60,1);break;
          case COMM_DOWN:  ZYSTM32_back(70,1);break;
          case COMM_LEFT:  ZYSTM32_Spin_Right(100,1);break;
          case COMM_RIGHT: ZYSTM32_Spin_Left(100,1);break;
          case COMM_STOP:  ZYSTM32_brake(10);break;
          default : break;
        }
```

void SearchRun(void) {

if(SEARCH_L_IO == WHITE_AREA && SEARCH_R_IO == WHITE_AREA) ctrl_comm = COMM_UP; //小车左右均探测为白色时前进

else if (SEARCH_L_IO == BLACK_AREA && SEARCH_R_IO == WHITE_AREA) ctrl_comm = COMM_RIGHT;//小车左侧探测为黑色，右侧探测为白色，小车右转

else if (SEARCH_R_IO == BLACK_AREA & SEARCH_L_IO == WHITE_AREA) ctrl_comm = COMM_LEFT;// 小车右侧探测为黑色，左侧探测为白色，小车左转 else ctrl_comm = COMM_STOP;//如果上述情况均不符合，小车停止

if(ctrl_comm_last != ctrl_comm) { ctrl_comm_last = ctrl_comm; switch(ctrl_comm) {

case COMM_UP: ZYSTM32_run(60,1);break;//当小车需要前进时，调用前进函数

case COMM_DOWN: ZYSTM32_back(70,1);break;//小车需要后退时，调用后退函数

case COMM_LEFT: ZYSTM32_Spin_Right(100,1);break; //小车左转时，调用左转函数

case COMM_RIGHT: ZYSTM32_Spin_Left(100,1);break;//小车右转时，条用右转函数 case COMM_STOP: ZYSTM32_brake(10);break;//小车刹车时，调用刹车函数 default : break; }

}

}

2.3.3 红外避障程序设计

对于红外避障模块，要判断两侧避障模块有无检测到障碍，当没有检测到障碍物时传感器置 1，检测到时置 0，先判断条件然后做相应函数的调用。

void AVoidRun(void) {

```
SR_2 = AVOID_RIGHT_IO;//获取右侧传感器状态
SL_2 = AVOID_LEFT_IO;// 获取左侧传感器状态
if(SL_2 == 1 && SR_2 == 1) {
ZYSTM32_run(60,10);//如果两侧均为 1，代表均无障碍物，小车前进
BEEP_RESET;
LED_D3_RESET; } else if (SL_2 == 1 && SR_2 == 0) {
    ZYSTM32_Right(70,300);// 如果左侧置 1，右侧置 0，代表左侧无障碍物，小车左转
} else if(SR_2 == 1 && SL_2 == 0) {
    ZYSTM32_Left(70,300);// 如果右侧置 1，左侧置 0，代表右侧无障碍物，小车右转
} else {
BEEP_SET;
LED_D3_SET;
ZYSTM32_brake(300);//如果条件不是上述情况，小车刹车
ZYSTM32_back(70,1000);//小车后退 ZYSTM32_Spin_Left(100,500);//小车右转
```

图 2.10 红外避障程序示例

```
void AVoidRun(void)
{
  SR_2 = AVOID_RIGHT_IO;
  SL_2 = AVOID_LEFT_IO;
  if(SL_2 == 1 && SR_2 == 1)
  {
     ZYSTM32_run(60,10);
     BEEP_RESET;
     LED_D3_RESET;
  }
  else if (SL_2 == 1 && SR_2 == 0)
  {
      ZYSTM32_Right(70,300);
  }
  else if(SR_2 == 1 && SL_2 == 0)
  {
      ZYSTM32_Left(70,300);
  }
  else
  {
     BEEP_SET;
     LED_D3_SET;
     ZYSTM32_brake(300);
     ZYSTM32_back(70,1000);

     ZYSTM32_Spin_Left(100,500);


  }
```

2.3.4 超声波避障程序设计对于超声波避障，当左右探测到障碍物距离小于一定值时，舵机转向，小车做出相应后退和转向动作。图 2.11 超声波避障程序示例

```
if(Q_temp<60 && Q_temp>0)
{
  ZYSTM32_brake(500);
  ZYSTM32_back(60,500);
  ZYSTM32_brake(1000);

  L_temp=left_detection();
  delay_ms(500);
  R_temp=right_detection();
  delay_ms(500);

  if((L_temp < 60 ) &&( R_temp < 60 ))
  {
    ZYSTM32_Spin_Left(60,1000);
  }
  else if(L_temp > R_temp)
  {
    ZYSTM32_Left(60,1000);
    ZYSTM32_brake(500);
  }
  else
  {
    ZYSTM32_Right(60,1000);
    ZYSTM32_brake(500);
  }
}
else
{
  ZYSTM32_run(60,10);
}
```

if(Q_temp<60 && Q_temp>0) //测量到前方有障碍物距离小于 6cm

{

ZYSTM32_brake(500);//小车刹车

ZYSTM32_back(60,500);//小车后退

ZYSTM32_brake(1000);

L_temp=left_detection();//测量左边障碍物的距离值 delay_ms(500);

R_temp=right_detection();//测量右边障碍物的距离值 delay_ms(500);

if((L_temp < 60) &&(R_temp < 60))//当左右两侧均有障碍物靠的比较近

{

ZYSTM32_Spin_Left(60,1000);//小车右转

}

else if(L_temp > R_temp)//如果左侧距离大于右侧

{

ZYSTM32_Left(60,1000);//小车右转

ZYSTM32_brake(500);

}

else //如果右侧距离大于左侧

{

ZYSTM32_Right(60,1000);//小车左转

ZYSTM32_brake(500);

```
} } else
{
ZYSTM32_run(60,10);//如果两侧均无障碍物靠的较近。小车前进
}
}
```

3 测试过程

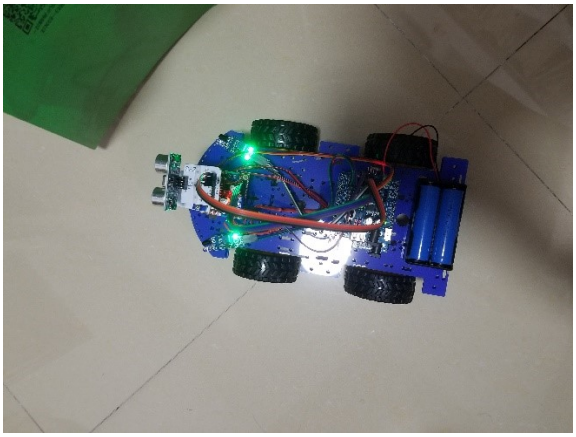
在开始测试前，最好能将 2 节电池给小车先进行上电操作，因为如果不先用电池上电只连接 usb 进行测试，容易造成小车冷启动，造成功能无法正常开启，可能会给接下来的测试造成误判。在打开开关后，如果扩展板和驱动板分别亮起绿色和红色的灯光代表各部件焊接正常，通过 kevil 程序编写并测试代码并将其编译并烧录到单片机中就可测试程序和方案的完整性。

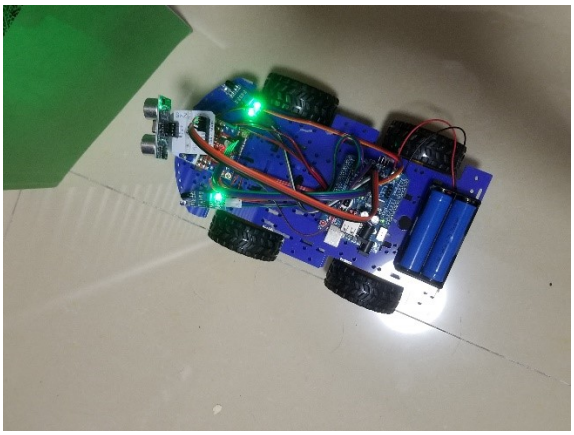
3.1 红外避障的测试

对于避障模块，要确保小车可以确实避开前方障碍物并继续行驶，在接线的时候要注意其有三个引脚，其中 VCC 接 5V 电压，GND 接地即为 0V 电压，OUT 为输出端，前端透明的部件是红外发射头，黑色部件为红外接收头。利用障碍物会反射红外光的性质是红外避障系统的基本原理，当小车探测到前方出现不透明物体时，红外发射头发出的大多数红外光都会被反射回到传感器前方，红外接收器接收到，系统中的 OUT 端就会输出 0V 即低电平，如果没有障碍物

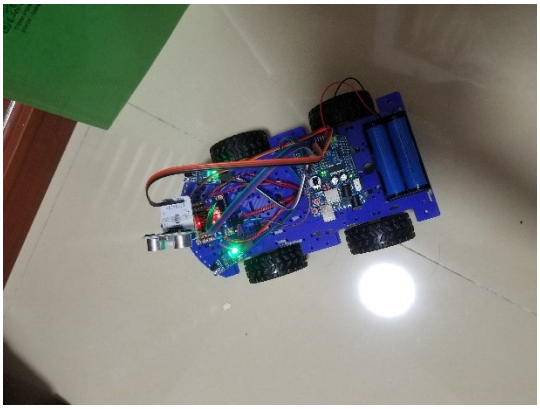
OUT 端就会输出 5V 高电平。在测试时，先将 usb 线连接到电脑并烧录进主板内，如果主板亮起绿色的灯，并且红外避障模块上的探头亮起绿色灯光就代表烧写成功，用手尝试遮挡探头前方位置避障的输出指示灯会亮起就代表模块可以正常运转。但是在调试过程中，我发现小车在行驶过程中并没有表现出红外避障的功能，推测红外避障模块出现问题，我仔细检查了排线，我发现左探头的 vcc 接口对应的 5V 母线在插入时松动，接触不良导致的。另外，在测试时应当注意的一点是电阻不可调节的过于灵敏，否则会造成指示灯频繁跳动，容易引起电压不稳定。

图 2.12 (abc)所示：小车右侧检测到障碍物，右侧传感器指示灯亮起，小车左转





(a) (b)



(c)

3.2 循迹的测试

首先测试循迹探头能否正常运行，先察看安装在小车底盘上的探头有无发出白色光亮，如果正常显示说明模块在正常运作，接着将小车放在黑色线上查看小车能否沿着轨迹行走。经测试，我发现小车放在地上不能运行，我猜测是因为循迹探头不能识别到地面导致小车无法循迹。可能有以下几个原因

(1)循迹发射和接收探头可能没有焊接好，或者接线的排针没有插好或接触不良。

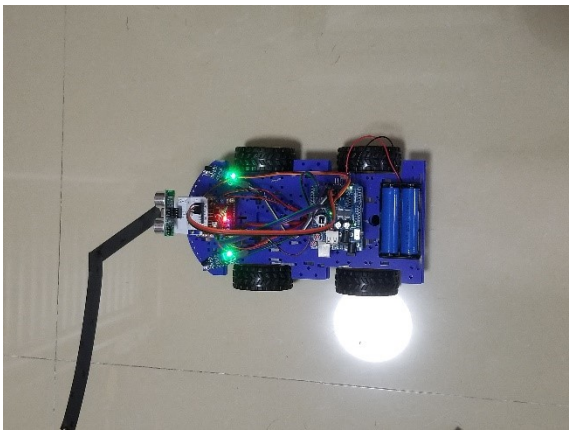
(2)循迹探头灵敏度过低，我可能需要旋转模块上的电位器调节灵敏度，使得模块能够探测到地面并传输信号。

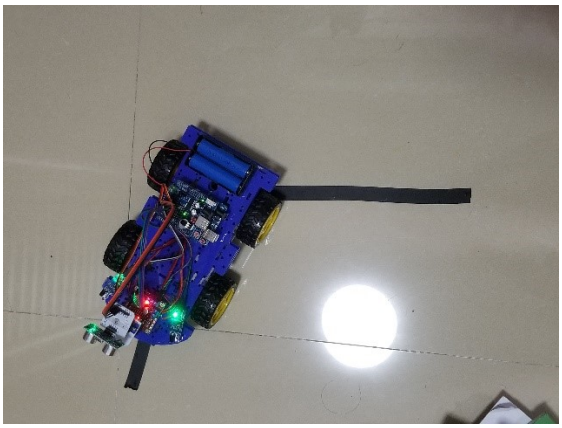
在经过上述排查之后，小车能够识别到地面并能运行，但是在循迹过程中常常偏离轨道，就像是没有安装循迹模块一般，发生这种情况可能是两侧探头灵敏度不一致，导致循迹模块经常错误识别地面有黑线，要解决这个问题，要先判断小车是往哪个方向偏离，若

(1)在小车运行过程中经常性往右侧偏转，这代表右侧探头灵敏度过高，我要往顺时针旋转电位器降低灵敏度。

(2)在小车运行过程中经常性往左侧偏转，这说明左侧探头灵敏度过高，我要往顺时针旋转电位器降低灵敏度。

图 2.13(ab)所示：小车按照黑线行驶



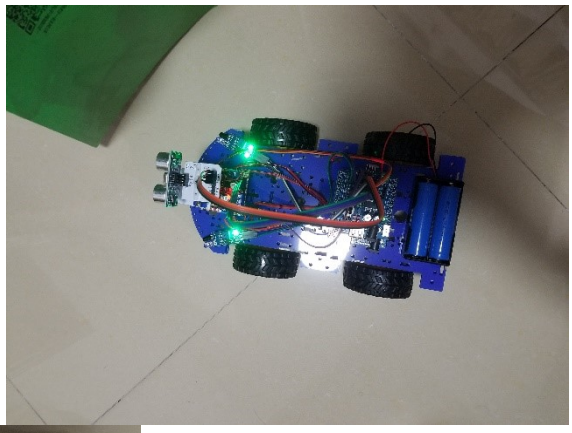


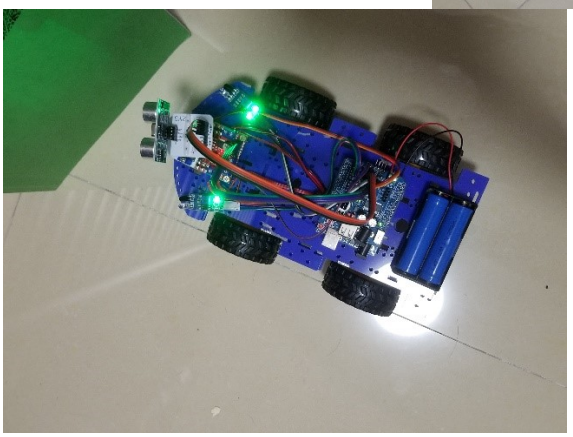
(a) (b)

3.3 超声波避障的测试

本次课题我采用的是 HC-SR04 超声波模块，此模块的工作电压为 5V，工作电流为 15mA，最远射程为 4m，最近射程为 2m，是有别于红外避障的远距离避障方案。装上超声波模块，再将程序烧录进去，按下扩展版上的复位按钮，在听到蜂鸣器的声音之后代表系统已经在开始运行了，正常来说，在装上舵机后的超声波模块应该以左右各 90°做往复探测，但是我看到舵机的初始位置不在正中间，往返运动偏离了 90°，变成在正前方和正后方做来回运动，小车不需要探测后方的障碍物，我的初始位置没有设定好，导致了这种错误，先按下重置按钮，再将舵机拆下，把初始位置调正就不会偏离了。在随后的测试中，我看到小车的行驶速度有点太快，有时超声波模块虽然识别到前方有障碍物，但小车由于速度过快，还是有可能刮蹭到物体，考虑到小车的亚克力底板较为脆弱，我担心过多的碰撞会导致小车底板或安装在前方的超声波舵机模块出现损坏，便将小车函数中的移动速度参数从 100 改为 70.

图 2.14(abc)所示：小车超声波舵机检测到前方有障碍物，小车和舵机均向左转





(a) (b)

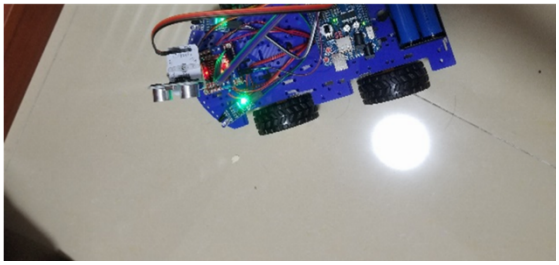
(c)

3.4 遥控的测试

在测试时发现，按下遥控器小车并没有按照遥控器的指示进行行动，有可能是我在焊接时不注意，造成车子的红外接收器虚焊，或者是遥控器部件损坏，如果是这样，我就要重新找到新的替代原件进行替换。在排除完故障后，首先，通过遥控器按下模式一即循迹模块，并将小车循迹模块放在黑线上，可以看到小车轮子开始自动转动。再按下模式二就进入避障模式；分别在左右两侧的红外避障传感器设置障碍物并测试输出指示灯能否亮起，亮起的同时应该通过电机驱动模块控制车子的左转或右转（车子的左转或右转可根据程序中改写 left 或 right 即可改变）模式三为遥控模式，即可用遥控器控制小车的前进后退及左右转向的操控。

4 结论

在现代，智能化设备大规模运用的环境下，智能机器人的应用将会更加广泛，随着智能设备的逐渐私人化、定制化，繁杂庞大的数据对设备的耐用性造成了不小的考验，这更加要求我们要有扎实的



专业基础和实践能力，通过这次毕业设计的实践，我感觉我对智能化机器人的理解更加深刻，在遇到问题时查询资料或者咨询同学、老师，更是加强了我的自我学习能力，还有通过自己对零件的装配、焊接，程序的编写等等的，小车整体的组装，从零到有把小车一步步组装起来给了我比较大的满足感与成就感。经测试，本智能小车具有较好的稳定性，良好的可拓展性，更因为其采用了 STM32 单片机，更是具备了稳定性、易于控制、拓展性高等等优点，本次设计，使得小车具有向前后、左右自由移动的能力，还具有主动避障功能和遥控功能。

参考文献

[1] 张于慧振，徐磊．基于 STM32 的智能移动小车[J]．电子世界，2020，(10):139-140.

[2] 蔡光昭，洪远泉，周永明．基于 STM32 的超声波测速测距系统设计[J]．现代电子技术，2014，第 37 卷

(24):87-89.

[3] 郑润芳，张海．STM32 的小车自主定位与控制系统设计[J]．单片机与嵌入式系统应用，2013，(9):46-49.

[4] 杨德坤．单片机技术在传感器设计中的应用[J]．电子测试，2018，(18):127，140.

[5] 朱向庆，何昌毅，朱万鸿等．基于STM32单片机的通信技术实验系统设计[J]．实验技术与管理，2019，第

36 卷(8):81-84.

[6] 盛珣华．单片机原理及应用［M］.6 版.武汉：华中科技大学出版社，2014.03

[7] 陈朝大，韩剑．单片机原理与应用 实验实训和课程设计[M]． 武汉：华中科技大学出版社，2014.06.

[8] 刘一．基于 STM32 的嵌入式系统设计[M]．北京：中国铁道出版社，2015.09.

[9] 阮毅，杨影，陈伯时．电力拖动自动控制系统：运动控制系统［M］.5 版.北京：机械工业出版社，2016.08.

[10] 董涛，刘进英，蒋苏．基于单片机的智能小车的设计与制作[J]．计算机测量与控制，2009(02):380-382.

[11] Juang, Shih-Yao; Juang, Jih-Gau. 2016. "Remote Control of a Mobile Robot for Indoor Patrol" Appl. Sci. 6, no. 3: 82.

[12] Arboleda, E., Alegre, M., & Idica, K. (2015). Development of a Low-Cost Electronic Wheelchair with Obstacle Avoidance Feature. Journal of Mechatronics, Electrical Power, and Vehicular
Technology, 6(2), 89-96.

附 录附录 1 程序源代码

避障函数 Avoid.c

```
void AVoidRun(void) {
SR_2 = AVOID_RIGHT_IO; SL_2 = AVOID_LEFT_IO; if(SL_2 == 1 && SR_2 == 1) {
ZYSTM32_run(40,1);
BEEP_RESET;
LED_D3_RESET; } else if (SL_2 == 1 && SR_2 == 0) {
//ZYSTM32_back(70,200);
//ZYSTM32_brake(30);//停止 30MS
ZYSTM32_Spin_Left(70,300); } else if(SR_2 == 1 && SL_2 == 0) {
//ZYSTM32_back(70,200);
//ZYSTM32_brake(30);//停止 30MS
ZYSTM32_Spin_Right(70,300); } else {
BEEP_SET;
LED_D3_SET;
ZYSTM32_brake(300);//停止 300MS
ZYSTM32_back(70,1000);//后退 400MS
ZYSTM32_Spin_Left(100,500);//左转 500MS
}
}
```

电机引擎函数 Motor.c

```
    void TIM4_PWM_Init(unsigned short arr,unsigned short psc)
{
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
TIM_OCInitTypeDef TIM_OCInitStructure;
```

GPIO_InitTypeDef GPIO_InitStructure;