**摘 要**

自动避障小车是智能小车中的一类智能化的机械产物，它的本身自带一个红外传感器，用来感知周围环境的事物以及小车本身的状况，在未知的环境中探索人类所不知道的东西，照亮人们前进的道路。然而，智能小车的提出对于智能汽车产业将会带来机遇，同时也会带来挑战，让汽车智能化在未来一定具有广阔的前景，因此，在这样的条件下，开展对智能小车的研究已经成为一种必然。智能小车涉及到单片机原理、自动控制原理、传感器技术、人工智能技术和焊接技术，它是将多项技术综合在一起的技术。

本论文主要是以STM32单片机为基础，结合各种技术来以单片机的微控制器为基本控制单元的控制设计技术。自动避障小车主要是以三个轮子为基本结构作为研究平台，我们选择STM32F103V8T6微控制器作为主控芯片，这种芯片在STM32单片机中的32位的FLASH MCU中性能是最好的，在我们的实际应用中，它是非常合适的，因为它具有很强的连通性和控制性。我们采用L298N芯片作为小车的驱动芯片。

该论文首先介绍了与智能小车有关的机器人的发展史和背景、机器人的研究的价值，以及与该设计相关芯片的介绍；然后介绍了避障小车的硬件电路的设计构思以及软件设计的方案。论文最后对该设计过程进行了总结并附带了该设计各个模块的电路原理图和整个设计的实物图，以及完整的源程序。

**【关键字】**STM32 L298N 智能小车 自主避障

**ABSTRACT**

Automatic obstacle avoidance car is belong to the smart car which is a intelligent mechanical product, It comes with an infrared sensor for itself ,which it is used to perceive things around and itself condition, exploration of things what human don’t know in the unknown environment, illuminate the road for people. However, intelligent car mode for intelligent car’s industry will opportunities but also challenges, let intelligent vehicles will have broad prospects in the future. So, in this conditions, develop the intelligent vehicles research has become a necessity. The smart include SCM theory、automatic control theory、sensor technology、artificial intelligence technology and welding technology, which is integrated with a number of technical techniques.

This thesis is based on STM32 microcontrollers, combining a variety of techniques to control single-chip microcontroller for the basic design of the control unit. Automatic obstacle avoidance car use three wheels as the basic structure as a research platform, We choose STM32F103V8T6 as the main chip microcontroller , The chip’s performance is the best in the FLASH MCU of 32-bit STM32 microcontroller. In our practice, it is very appropriate, because it has a strong connectivity and control.

This paper introduces the development process and background of the robot, value of Research for the robot and introduce the chip for this design. Then introduced the obstacle avoidance car of hardware circuit design concept and software design solutions. Finally, the paper summarizes the design process and comes with a circuit diagram of the design of each module and the overall design of the physical map, and complete source code.

**【Key words】** STM32 L298N Intelligent car Autonomous obstacle avoidance

目 录

[前 言 1](#_Toc452133086)

[第一章 机器人概述 2](#_Toc452133087)

[第一节 课题选题背景 2](#_Toc452133088)

[第二节 机器人的发展史及前景 3](#_Toc452133089)

[第三节 机器人的研究价值 5](#_Toc452133090)

[第四节 本章小结 5](#_Toc452133091)

[第二章 避障小车控制系统设计 6](#_Toc452133092)

[第一节 总体控制方案 6](#_Toc452133093)

[一、 总体控制方案框图 6](#_Toc452133094)

[第二节 避障机器人结构方案 6](#_Toc452133095)

[一、避障机器人结构的选择 6](#_Toc452133096)

[第三节 主控制器选择方案 8](#_Toc452133097)

[一、主控芯片的选择 8](#_Toc452133098)

[第四节 电机驱动的选择方案 11](#_Toc452133099)

[一、驱动电机的选择 11](#_Toc452133100)

[二、电机驱动芯片的选择 12](#_Toc452133101)

[第五节 传感器选择方案 15](#_Toc452133102)

[第六节 本章小结 16](#_Toc452133103)

[第三章 硬件设计 17](#_Toc452133104)

[第一节 传感器模块电路设计 17](#_Toc452133105)

[一、红外传感器障原理 17](#_Toc452133106)

[二、避障模块电路设计 18](#_Toc452133107)

[三、红外避障模块参数说明 19](#_Toc452133108)

[第二节 主控制器STM32的最小系统 19](#_Toc452133109)

[第三节 电机驱动模块设计 21](#_Toc452133110)

[第四节 电量检测模块设计 22](#_Toc452133111)

[第五节 实物组装与调试 24](#_Toc452133112)

[第六节 本章小结 24](#_Toc452133113)

[第四章 系统软件设计 25](#_Toc452133114)

[第一节 软件开发平台简介 25](#_Toc452133115)

[第二节 系统程序设计总流程 26](#_Toc452133116)

[第三节 避障模块软件设计 27](#_Toc452133117)

[第四节 电机模块的软件设计 27](#_Toc452133118)

[第五节 本章小结 28](#_Toc452133119)

[结 论 29](#_Toc452133120)

[致 谢 30](#_Toc452133121)

[参考文献 31](#_Toc452133122)

[附 录 32](#_Toc452133123)

[一、英文原文 32](#_Toc452133124)

[二、英文翻译 47](#_Toc452133125)

[三、源程序 60](#_Toc452133126)

# 前 言

自从第一台工业机器人的诞生开始，机器人技术的研究经历了一个漫长的发展过程。然而随着科学技术的快速发展，智能化早已经成为电子行业研究人员研究的热点课题之一，关于人工智能的研究也越来越备受人类的关注。智能小车对于现在我们所研究的很多研究领域来说，它都是有很大的应用前景，智能小车涉及到单片机原理、自动控制原理、传感器技术、人工智能技术和焊接技术，它将多项技术综合在一起。这项技术对于我们人类来说，它的研究无论是对于我们人类还是其它的事物都是具有重要的意义。并且，机器人已经遍及了电子、冶金、机械、交通、国防、宇航等领域。智能车可以代替人类在工作条件比较恶劣的环境下工作，如货物搬运、设备检测、未知地方的探索等。对于智能小车自身而言，行走在没有障碍的路线是最好的，但是，在探索未知的环境当中，要想没有障碍那是不可能的，所以它按照我们为它所设计的路线上行走并成功避开前方的障碍物，这对于完成我们给它的任务是很重要的。所以，在智能小车避障这一模块上的研究是非常重要的。

智能小车是一类智能化的机械产物，它的本身自带一个红外传感器，用来感知周围环境的事物以及小车本身的状况，在未知的环境中探索人类所不知道的东西，达到人类所要的目的，因此我们用它来探知复杂的环境，让我们更加地了解未知领域的东西。智能化技术覆盖了电子技术、控制、传感器技术、机械等多科学的设计。本次所设计的东西主要是采用意法半导体公司的STM32单片机，选用这一块芯片，主要是因为这块芯片的很多特点都比较符合我们所要设计的避障小车功能，如很好的功耗控制、高程度的集成整合，并且它还易于开发，还具有先进的内核结构，所以我们说这块芯片特别适合作为自动避障小车的主控制芯片[1]。

本次设计的作品是一个自动避障小车，信号发射、信号接收、信号处理、作出反应等性能，它采用了四个模块(电量检测、电机驱动、避障模块、传感器模块)，每个模块相对独立，各自执行各自的功能，但是相互之间又是可以调用他人函数，使得整个系统具有很高的稳定性。避障小车可用于那些给人们带来生命危险的工作，而且通过小车不仅可以降低作业的危险程度，还可以减少时间和资金的投入。自动避障小车不仅能给一个企业带来利益，甚至是一个社会都会受益无穷，但是它本身昂贵的造价只能是让我们望而却步，所以我们应设计一种造价低但是功能性又强大的自动避障小车已经是我们刻不容缓的一个重要任务。

1. 机器人概述

## 第一节 课题选题背景

由美国NASA喷气推进实验室研制的“勇气号”火星车于2003年6月在美国的佛罗里达州被送进太空，同年7月”机遇号“火星车也被送入太空，之后他们分别于2004年1月的3号和24号在火星的不同地方落脚，设计这两辆火星车的最初的目的是为了证明火星上有水，为了这一个任务，它们必须在火星上工作至少92个地球日的时间，这是研究人员事先规定的考核准则，同时还包括他们在火星上行走的距离至少600米，考察的地点在8个以上，而且还要把探测过程中的环境的照片拍下来作为证据。此外这两辆火星车必须有至少60个地球日的时间是同时处于工作状态。在2004年4月5号“勇气号“通过考核，同年的4月26号“机遇号”也完成了考核标准，所以，NASA的孪生火星车探索火星的计划宣告完成。“勇气号”在着陆的地方发现了碳酸矿物质的存在，这说明在火星上曾经有水的村在，这是因为这种矿物质只有在有水的环境下才能够形成，种种证据显示在火星上很有可能有人类生存所必须的物质（水），这为人类对火星的研究更进一步，同时也说明人类在智能小车方向的研究有巨大的进步，火星车在探索的过程中，若是发现有重要的东西值得探测的时候，它就会带动6个轮子向目的地行驶，在探测到前方有障碍物时，火星车自己就会寻找可行的路径进行行驶[2]。

然而在现代社会，已经研制出许多的机器人，例如Sinora齿雕机器人, Sinora齿雕机器人，它突破传统的修复方法，具有数字化的网络平台，同时也是经过3D智能数字化技术系统直接设计的，它是由德国的西诺德公司自主研发和生产的，而且这种机器人不会出现与我们设计之初规定的混合物和设定的时间有不同的地方，对于产品来说，当我们想要拥有一副干净、整齐的牙齿是，我们整个制作过程，从诊断、拍摄、设计、制作和试戴这一整个过程，我们只需要一个小时就可以搞定。所以可以看得出来，现代社会的发展是越来越快，机器人在我们的生活中占的位子越来越高。如2008年世界首例机器人切除脑瘤手术成功。也许很多人都想不到，做这一起手术的居然是卡尔加里大学医学院设计和研发的“神经臂”。相对与经验老道的外科医生的双手来说，它的成功率更高，不是因为他有更高的经验，而是因为它能够在更小的范围内更加稳定地移动，使整个手术不会出现错误。所以我们对智能小车的研究是非常有必要的。

STM32系列微控制器是意法半导体ST公司最先推出的基于Cortex-M3内核的处理器，这种处理器按照性能可以分为两种，STM32F103V8T6是增强型这一系列。STM32的工作频率是72MHz，这个工作频率能够使整个系统在运行的状态下实现高速度的运算，当它与其他的外部设备连接在一起时，它具有很好的控制。它还具有先进的内核控制、良好的功耗控制、高程度的集成整合，并且提供丰富的数据库，特别适合作为小车的主控芯片。

## 第二节 机器人的发展史及前景

据史料记载，美国是机器人的发源地，早在1962年研发出世界上第一台工业机器人，比起所谓的“机器人王国”的日本开始至少要提前五到六年[3]。在20世纪80年代之后，美国人觉得情况不好，政府和企业界才开始重新对机器人的发展状况进行详细的，还体现在政策，不仅是鼓励当地的公民对工业机器人的开发和应用，还加大对机器人研究费用的投入，加快机器人的研究进度，美国能够快速的发展，不仅是在军事上，商业上，主要的还是它对技术研究的重视，认为技术研究的快慢代表着这个国家发展的速度，所以当时很多人把机器人的发展看成是美国工业化的象征，同时美国从那以后，机器人的研究得到快速的发展。说机器人是美国再一次工业化的象征，那是因为在20世纪60年代到70年代，美国政府并没有专注与机器人的研究，他们只是在一些企业或者是高校开启部分的工作。在20世纪80年代，因为各大厂商对所拥有的机器人的使用技术日益成熟，然而对于一个商人来说，他们把利益摆在第一位，而且也因为用户对产品的要求也是越来越高，这就使得厂家必须得更新产品，这就使得厂家必须得去更新设备（机器人），所以研究机构就开始生产带有眼睛和可以对力量控制的机器人，对于很多人来说，这就是2代机器人，而且很快它在美国市场的占有率就达到了60%以上。美国人对时代技术的革新有着重大的作用。

早在20世纪50年代早期，由于对未知领域的不熟悉，同时也由于他们对机器人的定义并没有一个准确的定位，所以他们在20世纪50年代的后半期才开始着手在机器人技术的理论与实践上，但是很快他们便开展对机器人实体的研究，而且还在20世纪6年代的时候研发出了前苏联史上第一台机器人，它是一台用于深水作业的机器人，这在前苏联当时是引起了巨大的轰动，因为这标志着苏联人对机器人的研究开始步上了正轨[4]。有了前期的研究开展工作，而且也因为第一台机器人所引起的响应，同时也是因为苏联政府意识到机器人的重要性，所以政府将机器人的研究列入了国家科学技术发展的纲领中，苏联当时的研究速度也是非常的快，在20世纪70年代中期就已经研发出30多种类型的机器人，也许对于他人来说，这并不是很重要的，但是这对于苏联的经济起到巨大的作用，同时机器人技术的发展也在慢慢推动着苏联社会的发展，不管在哪个方面都占着重要的位置。所以他们对机器人的研究也是越发的重视[5]。

对于日本来说，他们在60年代末正处于经济高度发展时期，年增长率达11％。第二次世界大战后，日本的劳动力急剧缩减，而高速发展的经济更加剧了劳动力的严重不足。为了解决人口的不足，同时国家的占地面积不是很大，人口不能无限的增长，所以当时日本政府做了一个明智的选择，那就是用机器人来代替人工，这既可以解决劳动里力的不足，同时也可以降低成本，因此，当时日本的川崎重工业公司将技术比较成熟的美国的Unimation公司的机器人和技术一起引进来，并且还建立了专门的生产车间。他们将技术引进来之后立即投入到研究开发之中，于是他们在20世纪60年代末期在川崎重工业公司的生产车间试制出了第一台机器人，他们将这个具有历史意义的机器人命名为“尤尼曼特”机器人。但是另一方面，由于对机器人的研发需要大量的资金，这只能在一些大型的企业才能有足够的资金去研究，对于一些中、小型企业来说，只能是望尘莫及，由于这个反响，日本政府立即做出了决策，那就是由政府银行提供低息贷款，而且用户所租赁的机器人都是只需要支付很低的租金就可以了，所以日本政府对机器人这一块的投入还是非常大的，主要是机器人受到了很多政府以及研究机构的重视[6]。

然而在我们国家，机器人的研究也是被列入了国家重点科学研究之中，就目前我们国家的研究而言，已经建立了多个机器人研究示范工程，并且对机器人的理论与实践的研究也是颇有成效，还将这些研究付诸于实践之中，例如我们生产除了搬运机器人，喷漆机器人等等，就喷漆机器人而言，它对于我们工业的生产是非常重要的，因为我们人类在喷漆的过程中，由于油漆中含有有毒的物质，如果长期在这种环境下工作，它对于我们的生命是具有很大的危害的，而且使用机器人来完成这些工作，效率不仅能够提高，还可以节省很多的人力资源，做成流水线，这样我们的产品成本就会降低很多。根据我们国家目前的情况来说，我们的很多工厂需要的是那些成本较低，但是又实用的机器人。我们不应该盲目地区开发，我们得跟着时代的脚步前进，这样国家的整体水平才会得到提高。所以机器人的生产对我们国家的社会的推动是非常重要的[7]。

## 第三节 机器人的研究价值

社会的发展，技术的进步都离不开人类的努力，同时也不得不说，科学研究的推动起着至关重要的作用，这其中就有机器人的研究，在社会进步如此之快的今天，我们好像已经离不科技，就好比我们已经习惯了使用电灯，当某一天突然电灯全部消失了，我们重新点上了蜡烛，我们会感觉这一切都是黑暗的。同样机器人已经在我们的生活中扮演着重要的角色，这大大的提高了我们的生活水平[8]。

其实很多人说他们没有见过机器人，很多机器人要么是模型，要么就是在电影科幻片中见过，从来没有在现实中见过。其实机器人在我们的生活中是处处可见的，例如那些拍摄原始森林里面的动物的活动情况的时候，他们是用的是摄像机，可是你不要忘了，很多人是不能靠近这些生物的，他们只有通过遥控机器人来携带摄像机去拍摄那些小动物或者是对人类有伤害的动物；如一些考古学家去探测一些墓穴或者是一些山洞的时候，由于他们对这些山洞都是未知的，他们也只有通过机器人先进去做一个大概的了解，然后才能进去。而且在很多的工厂中都是在使用机器人，因为机器人可以无限制地工作，只要你定期去维护他们就可以了。在科研上，机器人还可以在空中，海底或者离我们更远的太空中工作，这为我们提供了很多的便利。就如NASA的火星车“勇气号”和“机遇号”来说，其发射到火星勘测的最主要的作业就是确认火星表面是否水存在的证据。而且它探测到的数据对于我们了解火星是非常重要的。

不仅如此，智能机器人在车辆的研究方面也是有着重要的作用，例如，我们可以利用智能小车扩展到智能车辆的研究，这样我们可以实现车子的无人驾驶化，这对于我们人类来说，又是在机器人领域的一大改革，假如我们将无人驾驶车辆研制成功，它不仅能为人类提供更为良好的交通网络以及降低车辆的燃油功耗，更好的是它至少可以多一份对我们生命安全的保障。所以智能小车的研究无论是在理论研究还是在现实应用中都具备重要的价值，因此我们对它的研究应该更加快脚步，让我们的生活得到更大的改变。

## 第四节 本章小结

本章介绍了选题的背景、机器人的发展史及前景和机器人的研究价值，从各个方面介绍了机器人无论实在我们的生活中还是科学研究上都有着重要的价值。并且还从我国国情上分析了智能机器人在我国的发展以及生产的情况。

# 第二章 避障小车控制系统设计

## 第一节 总体控制方案

1. 总体控制方案框图

对于控制系统每个部分的模块所做的选择得到的整体的结构方案设计框图如下：

电源模块

红外模块

STM32 主控模块

L298N 电机驱动模块

      图2.1 系统控制框图

控制系统整个模块主要实现的功能有：  
      （1）避障功能：小车在未知环境中自主运动。  
      （2）感应功能：小车在运行过程中，光电红外传感器感知距离范围内的事物。  
      （3）运动功能：前进、后退、左转、右转等动作。

## 第二节 避障机器人结构方案

### 一、避障机器人结构的选择

机器人按照行走的结构来分的话，主要是有轮式机器人、足步式机器人和履带式机器人三种。对于履带式机器人来说，它的着地面积大，特别适合野外工作，特别是一些道路不好的地方，例如我们常见的挖掘机就是其中的一种，还有我们军用的坦克，他们由于体积大，重量较大，所以他们的稳定性特别的好，缺点是控制算法不够完善[9]。

而足步式机器人无论是机械结构设计还是控制算法都相当的复杂，主要应用于道路崎岖的地方，但是它具有比较强的机动性。对于轮式机器人来说，它是目前人们使用最多的机器人结构方式，主要特点是运动速度和方向比较傲容易控制，并且操作简单，还具有比较强的活动能力和良好的稳定性；对于本次的设计而言，选择轮式结构的车体结构比较适合。

轮式机器人结构按照车轮的来分的话，它有一个轮子的、双轮的、三轮的、四轮的、或者是更多轮的，还有就是有的轮式机器人的轮子是可以360度移动的。然而我们的生活中主要常见的还是三轮和四轮的。一般多轮的都是比较大型的车辆才会配置或者是用于外星探测的小车。

轮式机器人根据驱动方式来分，可以分为两种，分别是；

导向驱动方式：就是机器人的运动速度和方向由不同的轮子和驱动器来控制。

差动驱动方式：当机器人在走动的时候，它的走动方向和行走速度是由相同的轮子和驱动结构来决定的走动方向的改变经过有规律掌控每个轮子的转速来操控的。

对于这几种轮式结构的机器人来说，三个轮子所形成的结构的方式比较多，而且它的稳定比较高，这种结构在我们研究所使用的结构和我们生活当中都是比较常见的，并且它还是轮式结构式的机器人的最基础的结构。对于我们这次的设计来说，我们所使用的是两个驱动轮和一个360度随动轮组成，这整个模块中，前轮是由两个高精度驱动轮组成，后轮是一个无驱动的360度随动轮来支撑。小车的行进速度是由前轮的速度来控制，也就是我们小车的驱动方式是前驱，同时小车实现转弯也是通过控制电机的开关来控制的。，所以这种结构属于三轮式 的差分驱动方式，这种结构的特点是结构组成比较简单，运动灵活。

差动驱动式三轮结构图如下；

左轮

左电机

主控制器

右电机

右轮

随动轮

移动机器人

      图2.2 差动式三轮结构图

## 第三节 主控制器选择方案

### 一、主控芯片的选择

在我们所熟知的单片机当中，能够是我们所需要的产品达到低功耗、低成本、高性能的单片机都很少，但是本次设计所选用的STM32系列的单片机却能够满足这些要求，它的内核是ARM Cort-0（(STM's product portfolio contains a comprehensive range of microcontrollers, from robust, low-cost 8-bit MCUs up to 32-bit ARM-based Cortex®-M0 and M0+, Cortex®-M3, Cortex®-M4 Flash microcontrollers with a great choice of peripherals. ST has also extended this range to include an ultra-low-power MCU platform) 。根据内核架构可以分为不同的产品，如下:  
 STM32F1系列产品的有:  
      ①STM32F103"增强型"系列  
      ②STM32F101"基本型"系列  
      ③STM32F105和STM32F107"互联型"系列

STM32F103类型的芯片是增强型系列的额，它的时钟频率是72MHz,而因为它的时钟频率比较高，所以在它达到 最高时钟频率时，对与闪存它所执行的程序，总的功耗才36mA，在32位单片机的市场是性价比比较高的产品[10]。

对于STM32型号的单片机的一些命名说明，有相当多的类型，这里我们以STM32F103V8T6来进行说明，规则如下；

表2.1 STM32F103V8T6参数

|  |  |  |
| --- | --- | --- |
| 1 | STM32 | STM32代表ARM Cortex-M内核的32位微控制器。 |
| 2 | F | F代表芯片子系列。 |
| 3 | 103 | 103代表增强型系列。 |
| 4 | V | v这一项代表引脚数，其中T=36脚，C=48脚，R=64脚，V=100脚，Z=144脚，I=176脚。 |
| 5 | 8 | 8这一项代表内嵌Flash容量，其中6代表32K字节Flash，8代表64K字节Flash，B代表128K字节Flash，C代表256K字节Flash，D代表384K字节Flash，E代表512K字节Flash，G代表1M字节Flash。 |
| 6 | T | T这一项代表封装，其中H=BGA封装，T=LQFP封装，U=VFQFPN封装。 |
| 7 | 6 | 6这一项代表工作温度范围，其中6代表-40--85℃，7代表-40--105℃。 |

101型号的性能；36MHz CPU 多达16K字节SRAM 1x12位ADC温度传感器。  
 103型号的性能如下；  
      ⑴特点

内核:ARM32位Cortex-M3 CPU，最高工作频72MHz，1.25DMIPS/MHz。单周期乘法和硬件除法。

存储器:片上集成32-512KB的Flash存储器。6-64KB的SRAM存储器。

时钟、电源管理和复位：电源供电我们使用的是3.6V的电压，接口用的是I/O接口电压。用于CPU时钟的PLL。带校准用于RTC的32kHz的晶振。

低功耗：当我们的模块不在工作状态的时候，这是电路可能出于以下这三种模式；待机、停止、休眠。当然我们可以随时进入工作状态，只要我们启动工作开关。就相当于是它随时处于待定状态，但是又是在低功耗模式下的。

调试模式:串行调试(SWD)和JTAG接口。

DMA:12通道DMA控制器。支持的外设:定时器，ADC，DAC，SPI，IIC和UART。

在STM32单片机中，它有很多个定时器，包括2个16位的基本定时器，主要是用于驱动DAC,2个看门狗定时器，主要是用于对信息用户的初始看门和窗口页面看门。还有2个16位的具有6个通道的控制定时器，主要是用于控制PWM的输出。4个16位的定时器，这些定时器中的每一个都具备4个PWM、OC、IC、脉冲计数器。还有一个24位的倒计时器（Systick定时器）。

最多多达13个通信接口:2个IIC接口(SMBus/PMBus)。5个USART接口(ISO7816接口，LIN，IrDA兼容，调试控制)。3个SPI接口(18 Mbit/s)，两个和IIS复用。CAN接口(2.0B)。USB 2.0全速接口。SDIO接口。  
 ECOPACK封装:STM32F103xx系列微控制器采用ECOPACK封装形式。

1. 架构优势

和以前的芯片相比，除了我们新加进来的功能经过强化后的外设接口，它的本身还具有同位处理器所拥有的标准接口，对于这个加强后的芯片而言，它无疑就是将自己定位在了一个比较高的阶段。这对于我们的开发人员来说，它是非常有利的，因为我们在设计中可以重复使用同一个程序接入口。这不仅为我们带来了方便，这也为电子产业带来了方便。

这类芯片除了在接口上有一定的强化外，同时在电压和节能上也同样保留了下来，所以低电压和节能依然是这种新系列微控制器的优点，它不同于其他产品的就是，其他相同类的产品在增强某一些功能之后，有一些功能就需要摒弃，也就是说以前的哪些功能都不能在这块芯片，其中它还具备一些常规的技术，如在2.0-3.6V这个范围内工作时，工作电压兼容主流的额电池技术，也就是说我们不用再去考虑电池这一块对我们的影响。

（3）低功耗性能

对于我们所说的这一块芯片，它的低功耗模式总共只有四种，当我们的工作时钟频率在72MHz时，就是在这一个时钟频率在闪存中执行代码的运算，消耗的电压仅仅只有27mA，而其他的芯片在这一块上消耗的电流可能时它的两倍以上，而且它还可以将电流消耗降低至2微安。当控制器出于停止模式的时候，如果我们需要将它转换到工作模式，我们只需要6微秒就可以完成这个工作。这为我们在使用的时候节省了很多时间，同时还提高了机械的工作效率，STM32芯片除了节能这一个特点之外，它还具有很多的功能，当然，这还需要我们去开发，但是它的6个超低功耗模式是最为显著的；

• 10.4μA低功耗运行模式，32kHz运行频率  
• 6.1 μA低功耗睡眠模式，一个计时器工作  
• 1.3 μA 停机模式:实时时钟(RTC)运行，保存上下文，保留RAM内容  
• 0.5 μA 停机模式:无实时时钟运行，保存上下文，保留RAM内容  
• 1.0μA待机模式:实时时钟运行，保存后备寄存器  
• 270nA待机模式:无实时时钟运行，保存后备寄存器

1. 系统作用

主要的结构有，集成嵌入式Flash和SRAM存储器的ARM Cortex-M3内核。和8/16位设备相比，ARM Cortex-M3 32位RISC处理器提供了更高的代码效率。因此我们在实现代码的运算的时候，它的高代码效率会为我们节约很多时间。它的FLASH存储器和ARM存储器具有很大的内存，所以我们在储存代码的时候，不同再担心内存不够，我们还必须要再外加一个新的存储器来使用，这样不仅增加整个系统的复杂性，还为我们的运行增加了困难。系统在启动的时候还必须要对系统时钟进行选择，这样我们在使用的时候就不会觉得内部会产生冲突，因为在复位的时候内部有一个8MHz的晶振被用来作为CPU的时钟。对于Boot模式的选择，我们会利用Boot导入程序进入系统存储器，用于通过USART1重新对Flash存储器编程。这样就不会产生系统冲突而导致机子死机的状态。我们在设计STM32的时候，根据普遍化设计，它的电压范围时2.0V-3.6V，电源从外部通过VDD引脚引入，主要提供的方向时I/O和内部调压器。

我们选用STM32主要还是电路在设计中我们是以低电平实现高性能。所以优化各种方案，ATM32对于我们来时，它的价值我们不言而喻。

本文采用的STM32单片机的封装是LQFP,它的实际物品外观图为；



      图2.3 STM32单片机外观图

## 第四节 电机驱动的选择方案

      由机器人结构方式的选择得意得知，本次设计的智能小车是采用左右两轮独立的驱动，利用差速转向结构，不同的车轮由不同直流电机进行控制，电机模块的选择如下；

### 一、驱动电机的选择

在避障机器人的驱动电机的选择上，有很多的电机可以让我们挑选，但是针对我们本次的设计而言，我们最后的选择时直流电机，它具有的优点如下；

①有比较大的转矩，因此可以消除传动装置产生的摩擦转矩和负载转矩。

②具有快速响应的能力，可以快速适应控制信号的变换和速度的变化；

③电机具有较大的负载能力，很硬的负载特性，保证在运行过程中速度步受负载的影响；

④直流电机具有很大的空载力矩，可以在控制系统发出停转的同时可以立即响应，而且还可以产生很大的力矩阻止小车运动所产生的惯性；

⑤直流电机相对别的电机来说运行时更加平稳。

### 二、电机驱动芯片的选择

智能小车在驱动这一块使用的电机时直流电机，而在工作的过程中，我们需要对小车的前进和后退进行控制，也就是我们需要去控制电机转动的方向，在避开障碍物时又需要通过调整直流电机转速来控制速度以达到转向的功能，直流电机在这种工作状态下消耗都是非常大的，电机的转动方向是需要我们在设计的时候将定时器对小车的启动于关闭以及小车在后退的时候我们需要改变电压的极性来控制电机，在我们的设计中，我们采用电机驱动芯片来实现。

这次我们设计所使用的芯片L298N，它是SGS公司的产品，它可以同时驱动两个直流电机，实现我们的双电机控制。

L298N可以同时控制两个二相的电机，也可以直接控制一个四相的电机。我们用它所设计的电路比较的简单，我们不需要去设计复杂的电路，因为这些功能已经达到了我们的要求，L298N芯片是可以直接对我们所要输出的电压进行调节，不仅如此，我们还可以使用单片机的I/O提供信号。

L298N可接受标准TTL逻辑电平信号 , 可接4.5～7V电压。4脚 接电源电压, 电压范围 为＋2.5～4.6V。当芯片的输出电流达到2A时，整个电路的电流大达到最大值，而此时芯片可以驱动电感性的负载，驱动芯片的15个引脚中第一个引脚和最后一个引脚下的三极管的发射极独立引出，这样是为了我们方便接入直流电流的采样电阻，这样我们在收集信号的时候，才能形成传感电流信号。

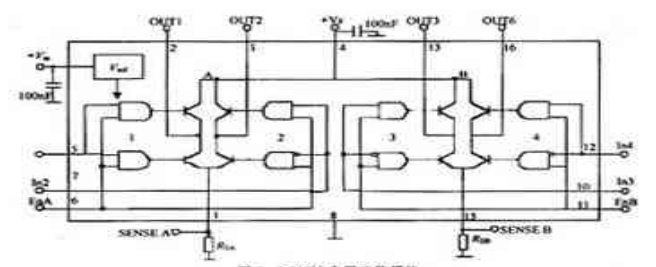


图2.4 L298N内部功能模块

表2.2 L298N功能逻辑图

|  |  |  |  |
| --- | --- | --- | --- |
| EnA | In1 | In2 | 运转状态 |
| 0 | × | × | 停止 |
| 1 | 1 | 0 | 正转 |
| 1 | 0 | 1 | 反转 |
| 1 | 1 | 1 | 刹停 |
| 1 | 0 | 0 | 停止 |

In3和In4的逻辑功能图和In1和In2的逻辑功能图是一样的，由上表可知，当EnA为高电平时，而In1和In2的输入电平为低电平时，电机处于停止状态；n1和In2的输入电平是一个高电平，一个低电平时，电机处于正转或者是反转；n1和In2的输入电平都为高电平时，电机会立马停止，这对于我们在实现小车的转弯时是非常重要的，因为小车在感应到障碍物的时候会立即停止并反向运动，这就时刹停。当EnA为低电平时，输入电平对电机都是起控制作用的，此时无论我们的In1和In2输入的是什么样的电平，我们的电机依然在运转。

L298N控制器原理图如下；

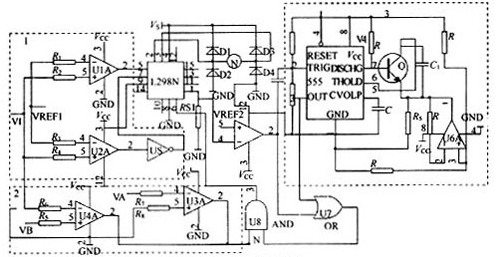


图2.5 控制器原理图

在图2.5中，我们可以看到有3个虚线框图，在框图1中，其中U1A和U2A时两个比较器，他们的任务就是用来比较内部压强传感器所传来的电压的大小，我们就是利用压强传感器的电压大小来控制电机的正向转动和反向转动，同时我们也可以通过电机的正转和反转来控制炉体的压强，。其实这就是一个循环的过程[11]。

框图2中U3A和U4A也是两个比较器，但是他们是联合在一起共同组合成一个双限比较器，这两个比较器的作用就是控制炉体的压强的稳定性，它可以实时地报告炉体的情况，以便我们对系统作出相应的措施。

框图3是一个长延时电路，这个模块的设计是为了让内部的比较器U5A和采样电阻Rs1,来控制输入电平的高低。

由上面可知，框图1控制电机正反转，框图2控制炉体压强的纹波大小。当炉体压强太小或太大时电动机转到两端固定位置停止，根据直流电机稳态运行方程；  
U=N+

其中： 为电机每极磁通量；

为电动势常数；

N为电机转数；

Ia为电枢电流；

Ra电枢回路电阻。

由公式可知，当电机的转数为0时，这时通过电机的电流就会加大，如果我们长时间让电机处于这种状态，那么电机很容易就会被损坏 。当然还有一种情况也会使电机的电流变化速度也很快，而且是呈现上升的速度，就是在电机启动的时候，这种情况的变化速度是很快的，它不会影响电机。这两种情况对于我们框图3来说，它的内部因为有长延时电路，它会自动区分，并且在遇到电机转数为0的情况的时候，长延时电路就会自动启动，并控制电路，而在遇到电机启动是电流过大的情况的时候，他会自动吸收电机启动过程中产生的多余的电流，使电机能够正常地启动。

驱动芯片L298N的引脚图如下；

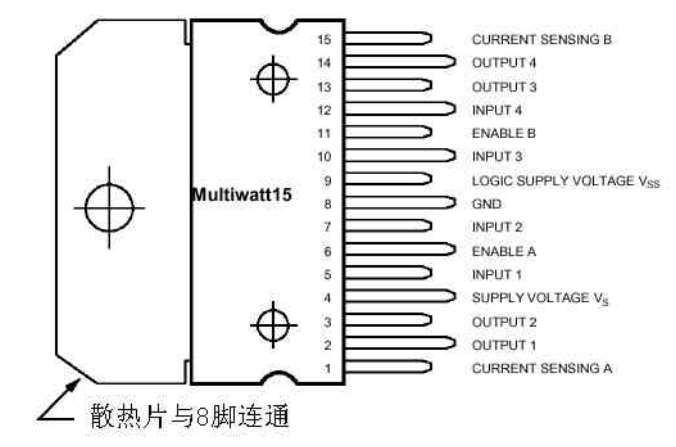


图2.6 L298N引脚图

## 第五节 传感器选择方案

对于自主避障小车来说，对障碍物的检测是整个系统中非常重要的一环，小车在未知环境的运动中，很容易会遇到各种各样的障碍，他不像我们人类一样可以依靠视觉来感知外界的事物，而避障小车唯一的感觉器官就只有我们为它安装的传感器[12]。对于我们所了解到的传感器有红外光电传感器和超声波传感器，而激光测距仪是这里面最好的，但是它的成本比较高，对于制造本身来说，不是特别的划算。超声波传感器是利用列大的原理来作用的。对于红外光电传感器而言，它具有探测视角小，且具有很强的方向性，就是无法确定与障碍物的距离。但是对于我们本次所设计的东西而言，红外光电传感器的作用已经达到我们的目的了。这种传感器具备很强的环境适应能力，它的原理就是利用反射的原理来做成的，就是当红外传感器向前方发射一定频率的红外线，遇到障碍无后会反射回来，此时就需要接收管来接收反射回来的红外线，然后在进行信息的处理，最后作出反应。红外光电传感器的感应强度是可以通过电位器来调节的，它的优点就是对其他模块的干扰比较小，而且容易装配和使用[13]。

## 第六节 本章小结

本章对整个小车的设计所需要的主体部分进行详细的描述，以及对使用各个部件的选择方案的对比，如驱动芯片、主控芯片、电机驱动、电源设计等都进行了对比。在此之上，我们对小车所要实现的功能做出了总体上的规划。并且在选择主控芯片时，对主控芯片进行了详细的描述。

1. 硬件设计

## 第一节 传感器模块电路设计

### 一、红外传感器障原理

红外光电传感器的工作原理是运用红外光电传感器的反射的特点来实现的，也就是利用镜面反射的原理来感应，传感器的的主要成分是红外发光二极管发射出的能够被光敏接收管接收的光线，光敏接受管的作用就是专门用来接收障碍物反射回来的光，然后再由光敏接收管将信号传递给系统。而且根据接收管接收到的信号可以判断前方是否有障碍物的存在，还可以判断障碍物距离红外传感器发射管的距离，根据这个原理来判断小车该什么时候作出反应[14]。光电红外传感器反射红外线以及光明接收管接收信号的原理如下图；

红外发射管

光敏接收管

距离X

图3.1红外避障原理图

红外光电传感器的的实物图如下；

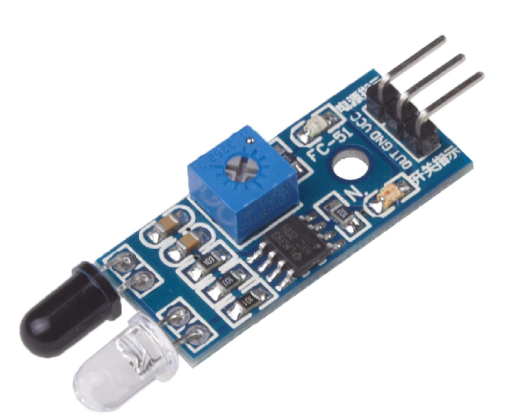
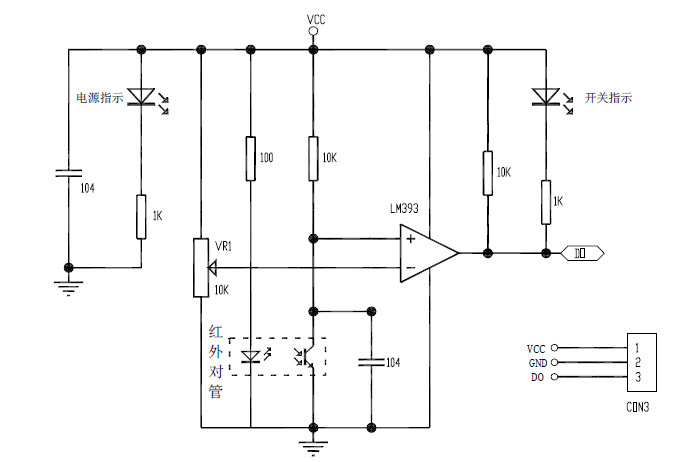


      图3.2红外光电传感器实物图

### 二、避障模块电路设计

避障小车的红外模块电路设计图如下；

  
      图3. 3红外避障模块电路设计图

### 三、红外避障模块参数说明

1.当红外避障模块的传感器的光敏接收管没有接收到前方或者是左右两边的反射信号的时候，在我们焊接的电路板上的指示灯亮，并呈现高电平，这个时候在模块电路的OUT段输出的电平是低电平，这时驱动芯片的In2呈现低电平，也就是说电机时正转，继续向前行进，而红外传感器的能够检测到的范围是2-30cm，传感器能够检测到的范围是发射管的35度范围内，我们检测的距离是可以调节的，但是距离越大，检测的结果就越不明显，通过调节电位器来调节检测到的最大距离。

2.光电红外传感器的感应管有两个，一个白色的和一个黑色的，其中白色检测到的距离最大，黑色检测到的距离最小，这是因为在小车运动的过程中，会遇到面积不同，形状大小不同的障碍物，这时就需要不同的探测器来检测障碍物是否存在。传感器检测电路图如下；

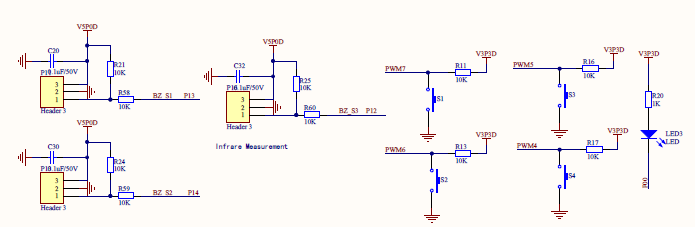


图3.4传感器避障检测图

[3.传感器模块输出端口OUT可以直接与单片机IO口连接，也可以直接驱动一个5V继电器；连接方式：VCC-VCC;GND-GND;OUT-IO[2].](javascript:;)比较器采用LM393，工作比较稳定。  
 4.可以采用3-5V直流电源对模块进行供电。当电源连接通时，红色电源指示灯点亮。

## 第二节 主控制器STM32的最小系统

STM32的最小系统电路图如下;

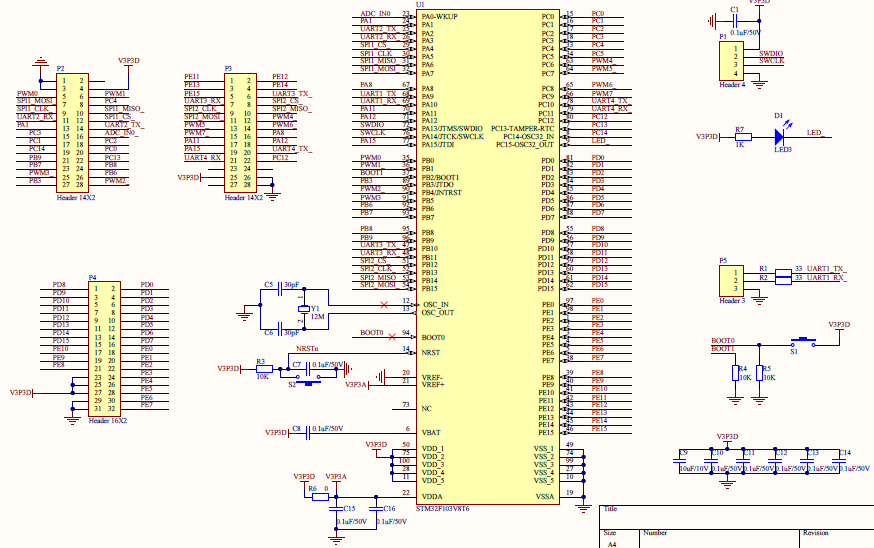


图3. 5STM32的最小系统电路图

在STM32F103系列单片机的内部的高速RC振荡器（HSI）是由VDDA、VSSA供电，因此我们在使用单片机的时候即使不用单片机自带的A/D转换器，那么也必须保证VDDA、VSSA的供电，否则STM32F103就不能按照正常情况启动。

在STM32芯片中，它有一个芯片引脚就是备用电源引脚，VBAT引脚的作用是，当VDD关闭或者是整个电路断电的时候，这时实时时钟与备用寄存器就可以通过VBAT引脚来获得动力。

而我们在对STM32启动时，我们可以选择Boot0或者是Boot1引脚的启动,如下表；

表3.1STM32的选择启动模式

|  |  |  |  |
| --- | --- | --- | --- |
| BOOT选择模式引脚 | | 启动模式 | 别名 |
| BOOT1 | BOOT2 |
| x | 0 | 用户闪存 | 用户闪存被选作启动区 |
| 0 | 1 | 系统内存 | 洗统内存被选作启动区 |
| 1 | 1 | 嵌入式SRAM | 嵌入式SRAM |

STM32 的配置启动模式图如下；

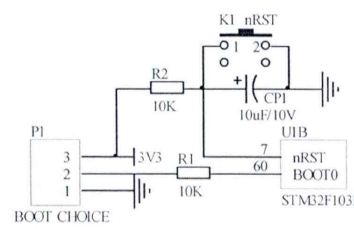
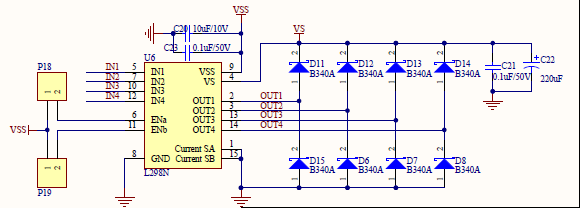
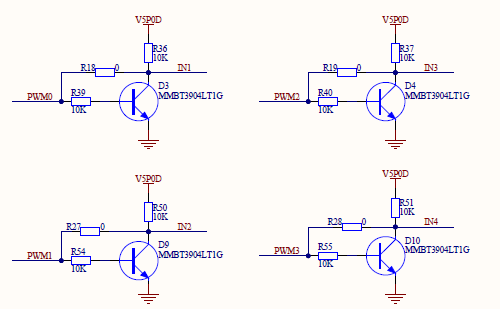


      图 3.6STM32启动模式图

## 第三节 电机驱动模块设计

避障小车采用的两轮独立的驱动方式，也就是左边的轮子和右边的轮子由各自的电机驱动，相互之间的运行情况不受彼此的影响，相较于其他的电机，我们选用直流电机，因为直流电机的驱动功率比其他电机的功率大，对于我们所选用的驱动芯片L298N而言，除了它本身具有比较稳定的动态外，还有就是它的里面嵌入了H桥的高压大电流双全桥式驱动器，所以我们选用它作为我们这次设计的驱动芯片，原理图如下；





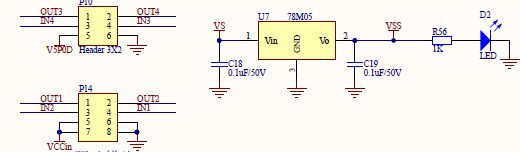


图 3.7L298N电机驱动电路图  
      在电路图中，芯片L298N的IN1、IN2、IN3、IN4引脚是芯片控制信号的输入端口，不同的逻辑电平组合控制着电机的转向，ENa、ENb是驱动芯片内集成的2个H桥使能端口，高电平使能。电机运动的使能状况如下表；

表3.2电机运动状况控制

|  |  |  |  |
| --- | --- | --- | --- |
| 控制信号 | | | 电机状态 |
| ENa | IN1 | IN2 |
| 1 | 1 | 0 | 正转 |
| 1 | 0 | 1 | 反转 |
| 1 | 1 | 1 | 刹车 |
| 0 | X | X | 停止 |

## 第四节 电量检测模块设计

在一块电路的设计之中，电源对它们来说都是很必要的，电源就好像是我们的打火机，当我们的面前有一大堆的好烟，但是你却没有火机，此时你的这些烟对你来说就没有什么意义了，如果一个电路设计从硬件到软件再到焊接以致完成所有，而只剩下的就是上电，但是此时你没有电源，你的这个设计就更本无法运行，此时他就只有仅供欣赏，没有其他用了。同时对电源电量的检测也是非常重要的，因为对于一个系统来说，电量的充足供应是非常重要的,在这一块中，我们使用AMS1117CD-3.3作为主芯片，因为AMS1117CD-3.3三端都可以调节，或者固定电压3.3V输出电流为1A的线路调整率：0.2%（最大）负载调整率：0.4%（最大），而且连接的有LM2904DR放大电路，它具有内部频率补偿，大直流电压增益，极地的电源电流/OP(500霢)基本上是独立的电源电压。并且还使用MP1584这个内部集成高侧高压功率MOSFET的高频降压型的开关稳压器，使系统在工作的时候不会出现电压不稳定的状况，这样就能保证系统要么一直在工作，要么因为断电停止工作，这样系统就不会出现电压不稳

定而导致系统部件被损毁[15]，电量检测的模块电路图如下；

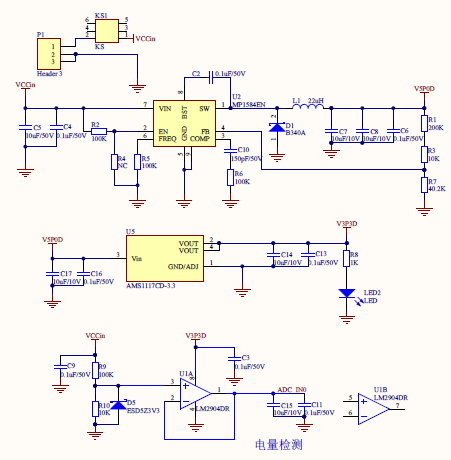


图3.8电量检测模块电路图

## 第五节 实物组装与调试

避障小车的控制系统的硬件电路时利用模块设计来实现的，在这一个大的模块中，各个小模块是相互独立的，整个大模块是以STM32芯片为主要的控制器（核心），模块与模块之间的运行是独立的，系统尽量去除由于模块化而对整个系统带来的影响，同时也有模块与模块之间产生的干扰，完成后的这个那个避障小车的实物图如下所示；



图3.8避障小车实物图

## 第六节 本章小结

本章介绍了避障小车从传感器模块设计到电量检测模块设计，总共四个主要的模块，对于每一个模块都有完整的介绍，并且每一个模块里的一些比较小的芯片都有介绍和他们在这一个模块中的构成情况，最后总结了我们在硬件设计中需要知道的东西。

1. 系统软件设计

软件设计是避障小车控制系统设计的重要组成成分，只有通过软件开发平台对硬件进行合理有效的编程，才能达到我们想要的小车所能实现的功能，软件设计是我们赋予给硬件的思想，那么它所代表的就是整个系统既看不见，但是它又是存在的东西，本次软件设计采用模块化的编程设计，各个部分相互独立，但是又可通过各模块之间的函数接口，使每一个模块都连接到一起，组成完整的控制系统。

## 第一节 软件开发平台简介

主控制器STM32F103V8T6是基于ARM Cortex-M3内核的单片机，所以只要是基于ARM的开发环境都可以，而ARM开发工具有ADS1.1、ARM REALVIEW DEVELOPER SUITE、IAR EWARM、KEIL ARM-MDKARM、WINARM(GCCARM).而这次我们使用的KEIL ARM-MDKARM作为本次所使用的开发工具，KEIL ARM-MDKARM是ARM公司推出的一款开发工具，它将Keil公司的IDE环境的μVision和ARM公司的RealView编译工具RVCT3.1这两者的优点集于一体。然而KEIL ARM-MDKARM作为ARM公司的主流开发工具，也是到目前为止为数不多的完全支持Cortex-M3控制器的开发工具[16]，开发环境如下图所示；

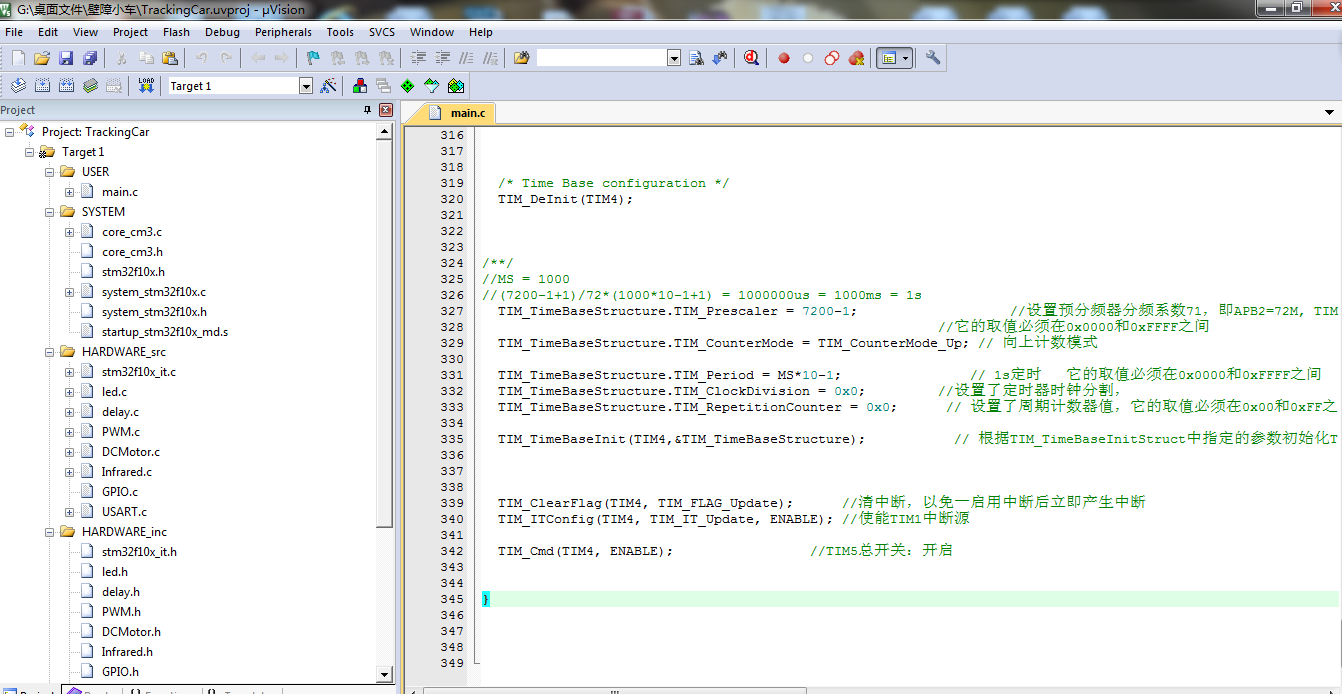


图 4.1 keil开发环境图

在STM32单片机方面的开发，我们使用最多的是Keil ARM软件，因为这个软件生成目标代码的速度非常的快，由C语言转换成汇编语言的时候，虽然看似非常的拥挤，但是当你慢慢观察的时候，你会发现，如果我们开发大型软件的时候，你就会知道Keil ARM 的优点体现在什么地方了。对于这一款软件而言，我们在使用的时候若是使用C语言，它的作用要比汇编语言要占一定的优势，因为C语言的可读性、可维护性、结构性这一款软件中体现出来的东西要比汇编语言的对多一些。所以，这款软件对于很多人来说，是比较好的选择，它的易学易用让我们减少了很多的麻烦。

## 第二节 系统程序设计总流程

避障小车的系统的程序设计的最终的目的就是为了让小车能够跑动，并完成我们给它定下的任务。小车总的设计就是，当我们启动电源开关灯的时候，在没有障碍的环境中，小车向前跑动，这个过程中小车身上的传感器一直处于待命状态，当前方有障碍物的时候，传感器会自动感应，并将信号传递给下一个功能模块，最后小车的反应是向后倒退一定的时间，然后进行转弯，一定时间后小车又恢复直线向前前进，在小车的完成这些动作的过程中，小车检测到左边有障碍物和右边检测到有障碍物作用的方法是一样的。在整个过程中，重要的点是红外传感器的检测，电机驱动芯片对电机的驱动以及主控芯片对整个系统的控制。系统程序控制主流程框图如下；

开关启动开始

向前移动

红外传感器是否检测到障碍物

左边传感器是否检测到障碍物

右边或中间的传感器是否检测到障碍物

后退一定的时间后右转

后退一定的时间后左转

否

是

是

是

否

否

图4.2 系统程序总流程框

## 第三节 避障模块软件设计

在小车运动的过程中，我们对小车在避障处理这一块的算法有三种；

（1）小车前方没有障碍，左右电机同时启动，向前直行。

（2）小车的中间传感器和右边传感器检测到有障碍物时，小车作出的反应是向左转；

（3）小车左边的传感器检测到有障碍物时，小车向右转；

在设计中，我们考虑到了小车可能遇到的一些情况，也根据传感器能检测到的范围，我们使用了三个传感器，分别安装在小车车头正前方，右上方和左上方，这样更能感应到更大的范围。当然，由于我们在本次设计中使用的是光电红外传感器，所以当我们在使用小车的时候，有时候由于光线的问题，使传感器接收到信号，此时，小车会自动转弯。

## 第四节 电机模块的软件设计

对于电机模块部分的设计，我们在设计中采用的是两个电机来配合使用，在整个模块中，就是利用电机来控制方向，在设计中，我们采用PWM调速算法来控制，再利用传感器和电机配合设计，如，当左边传感器感应到障碍物时，小车会由于电机的反转而是小车向后倒退，而此时调用主控芯片的定时器，对小车的两个电机的反转时间进行定时，当定时停止时，小车恢复正常转动，而由于左边红外传感器的作用，也就是调用了传感器的[函数部分，](javascript:;)使右边的电机停止转动，只有左边的电机在运行，这个时候主控芯片的定时时钟开始计时，达到设定的时间后，计时完成，右边的电机启动，开始运转，此时两边的电机都在运转，小车向前直行。

## 第五节 本章小结

本章一开始就对总系统软件模块化的设计的重要性进行了介绍，仅接着我们介绍了与本次软件设计有关的软件开发平台Keil ARM以及描述了系统总的程序设计的流程框图。还对电机驱动模块和避障模块的程序设计进行了说明。

# 结 论

避障小车是一门具有广泛应用前景的技术，不管是在工业应用领域，还是在科学探测领域。不论是在我们的生活中还是在那未知的星球上，避障小车的出现无疑给我们带来了许多的便利，它为我们省下了很多的物资，它为我们的生命作出了巨大的贡献，无论怎样，它都扮演着重要的角色在我们的生命中。本次设计了一个以STM32F103V8T6为主控制器的避障小车。总结的工作如下；

1.基于这次我们设计要使用的单片机使STM32以及我们查阅了大量的资料后，我们将这个系统分成四个模块来设计，分别是；电量检测模块设计、电机驱动模块设计、传感器模块设计和主控制器模块设计。

2.在每一个模块的设计上，我们为了既能使设计达到最优化，又能使小车达到我们所设定的功能，我们对每一个模块的每一个元器件进行了对比、选择。

3.在软件设计中，我们使用了ARM公司的keil开发工具作为软件开发平台。

当然，在整个设计的过程中，又或者是已经设计出来的小车还存在很多的缺陷，在很多的细节上面都没有考虑完全，例如；若是我们能做成用蓝牙、语言、又或者是用我们的肢体语言来控制小车。本次的设计使用的是模块化，若是我们有更好的方案，可以将设计成整体化，这样就可以消除模块化设计带来的彼此之间的干扰。在之后的设计之中，我们可以将它与视频连接，这样我们既可以控制它的方向，又可以了解小车的实时情况。

# 致 谢

在论文即将完成之际，首先我要感谢我的指导老师，无论是在实物的硬件设计方面还是论文的撰写，他都给了我很大的帮助，。在我完成设计的过程当中，老师对我的每一步都是严格要求，在每一个学习阶段的进度报告中都是给予悉心的指导。虽然我们相处的时间只有那么几个月，但是在他身上所学到的东西却是无比的多，他教给我的不仅仅是学习上的东西，还有为人处事的态度。在此我对我的指导老师致以深深的谢意，同时祝您工作顺利，身体健康。

[最后感谢我的父母，谢谢他们一直以来对我的生活的无微不至的照顾，给予我精神上的鼓励，学习上的支持以及为人处事的教导，这些都是我生命中重要的财富，也是我前进的动力](javascript:;)。

# 

# 参考文献

[1] 刘崇翔.基于ARM的智能机器人设计与研究[D].无锡：江南大学，2012.

[2] 新华网.机遇号、勇气号都达标美火星探测计划成功[EB/OL].

[3] 徐国华，谭明.移动机器人的发展现状及其趋势[J].机器人技术与应用，2001（3）：7-14

[4] 徐德，皱伟.室内移动式服务机器人的感知、定位与控制[M].北京：科学出版社，2008.

[5] 张过良.自主移动机器人的设计与制作[M].西安.西安交通大学，2008.

[6] 袁源，张永汀.工业机器人时代新生农民工的生存抉择[A].西安.2014.

[7] 长枫.中国机器人之梦兑现[A].2015.

[8] 福建省电子协会,福建省机器人学科发展研究报告[A].福建.2013.

[9] 史科科.基于虚拟样机技术的嗅觉机器人机械系统的研究.[M].哈尔滨.哈尔滨工业大学.2009.

[10] 孙白艳，李众立.基于电能专用计量芯片ATT7022A的低压无功补偿控制器设计.[M].四川.西南科技大学.2015.

[11] 孙中华，孙南海.基于L298N芯片对离子渗氮中压强的控制.[M].湖南.中南民族大学.2003.

[12] 周柱.基于STM32的智能小车研究.[D].四川.西南交通大学.2011.

[13] 郑佩，带博文.基于51单片机智能小车的设计与实现.[M].宁波.公安海警学院.2014.

[14] 王志.车载轨道水平检测装置.[M].黑龙江.黑龙江大学.2010.

[15]董雷刚.智能循迹电动小车的设计.[M].黑龙江.大庆师范学院.2010.

[16] 程冲，陈华琳.基于PROTEUS与KEIL软件仿真的频率测量实现.[M].湖北.武汉铁路职业技术学院.2008.

# 附 录

### 一、英文原文

**Proceedings of the 18th International Conference on Automation & Computing, Loughborough University, Leicestershire, UK, 8 September 2012**

**Development of an Expert Control Strategy for Controlling Obstacle Crossing of a High-voltage Transmission Line Inspection Robot**

**Jidai Wang**， **Aiqin Sun**， **Wei Su**

School of Mechanical and Electronic Engineering

Shandong University of Science and Technology

Qing Dao 266590, P.R. China

Djdwang8911@sina.com

**Jihong Wang**，**Hao Liu**

School of Engineering

University of Warwick

Coventry CV4 7AL, UK

jihong.Wang@warwick.ac.uk

**Abstract**

A rule based expert control system is designed for controlling a crawling robot to cross obstacles along highvoltage transmission lines. The objective of this study is to achieve automatic inspection of the line faults by using a remotely controlled inspection robot. The expert control strategy presented in the paper adopts the principles of artificial intelligence and embeds intelligent algorithms and strategies in the controller for the robot. The paper discusses the structure of the expert system based control strategy and the design methodology to derive the robot knowledge base and inference engine. VC++ and CLIPS are

used as the programming languages for software design and implementation. The feasibility and practical applicability of the expert system are tested and verified through experimental study. The experimental results show that the control strategy works well with the satisfactory performance and has a great potential to be applied in practice.

【**Keywords**】robot control artificial intelligence expert system knowledge base inference engine

1. **INTRODUCTION**

In recent years, the theoretical research and practical applications of expert system based control methods have made remarkable progresses. Introducing expert system is a turning point in artificial intelligence (AI) application history that led the AI from the exploration of general thinking to the use of specialized knowledge and moved AI from the theoretic research to the design of actual systems [1]. An expert system is a typical "knowledge engineering" system, which is the comprehensive application object of the knowledge expression, storage, reasoning, acquisition, and management. In addition, it is

a technology tool for research and development of knowledge based control and decision systems. It has lead to the realization of the development of the computer software, hardware and system from data information processing to knowledge information processing. The expert system has been traditionally used in the field of fault diagnosis, fault-tolerant control, interpretation and analysis systems [2]. There are fewer application cases in direct robot motion control apart from robot path findings and decision making.

Through our research, a high-voltage transmission line inspection robot is designed as shown in Fig.1 ([3], [4]). However, it is an extremely challenging task for the robot to work in a complex scenario, such as, obstacle crossing, inspection while moving, operation in different weather conditions, etc. To address those challenges, this paper reports our research in adopting expert system based

control methods in designing an intelligent rule based control for driving a robot crossing obstacles while it is crawling along high-voltage transmission lines. The simulation study has shown that this method is an effective approach for the robot to accomplish the obstacle crossing task in an intelligent and smart.



Figure 1. Picture of the transmission line inspection robo

**II. THE STRUCTURE OF THE EXPERT SYSTEM BASED CONTROL STRATEGY**

1. **Components of the expert system**

This subsection introduces the components consisting of the expert system for robot control. The expert system based control is developed as a rule based control system, which allows the control to be more flexible in structure (as the new rules can be added in at any design stages) and in-cooperate more human intelligence in its decision making mechanism [5]. The knowledge can be expressed by using IF {…} THEN logics and this expression is similar to the way used by humans thinking. The logics are used to derive the action and decision and the knowledge obtained from human experiences (expert) are stored in a knowledge base then the knowledge is extracted from the base through an inference engine.Expressed in a simple way, the expert system could be represented by:

Expert system = Knowledge + Inference [6]

In order to refine the design of the expert system, the composition of the system is expanded to have the structure as shown in Fig. 2. In addition to the knowledge base and inference engine, the system consists of a static database, dynamic database, explaining mechanism and man-machine interface.

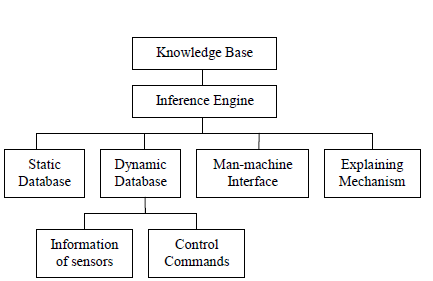


Figure 2. Structure of the expert system based control

**B. Functions of the system components**

To assign or describe the functions of each component in the system, the first step is to number the test line and then extract the features of obstacles associated with it.The standard state of motion is listed with the sensor information by tabulation according to experts’ experience. All of the information above is static knowledge. This information then needs to be translated into machine language and then are stored in the static database. With these facts, the efficient and concise rules are established. These rules are stored in the knowledge base. The next step is the comparison of sensor information with the information in knowledge base. If the sensor information corresponds to the rules stored in the knowledge base, the inference engine gives the matched rule priority, and the robot acts according to the highest priority rule provided. At the same time the new information that the inference engine produced will be stored in the dynamic database. The dynamic database also stores sensor messages that are received in real-time and the control commands generated. To link the above together, there is an explanation mechanism and manmachine interface in the expert system. The man-machine

interface is helpful for user to communicate with the robot while the explaining mechanism is used to explain the motion state of robot and the error in the system. It will help the users to test, update and maintain the expert system.

1. KNOWLEDGE BASE
2. **The realization of the rule base**

The expert system works based on rules pre-stored in the database and knowledge base and the rules are used in inference. While inspecting of the transmission line, the robot attitude should be adjusted according to the surroundings. Therefore, the rules for the system are the bases of changing the action and attitude. As shown in Fig. 3, it is designed for the three-layer structured rule base of operation mode along the line, work category and action rule. The robot operation mode and work category are distinguished by the infrared sensor and visual processing. The action rules are established by imitating the artificial crossing obstacle and dividing the process into some steps. And in order to inspect the environment change and determine the attitude adjustment,we placed different sensors at the respective part of the robot.

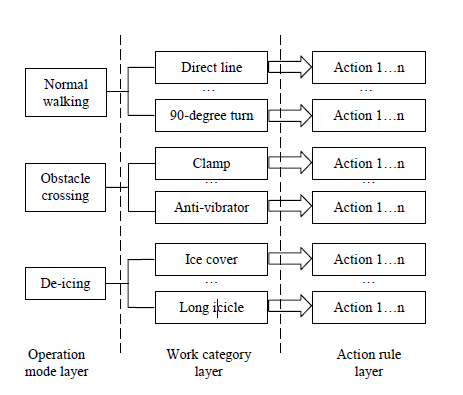


Figure 3. Sturcture of rule base

The rules are expressed in IF {…} THEN format [7]by CLIPS (The C Language Integrated Production System). They have the following descriptions and expressions:

(defrule <rule name > [<comment>]

<patterns>\*;// triggered instructions of the rule

=>

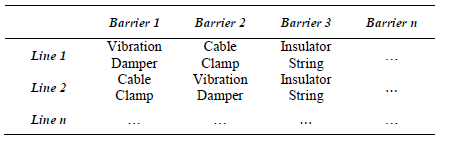
<actions>\*); // alterative action of the rule

**B. The composition and realization of fact base**

**i) The composition of fact base**

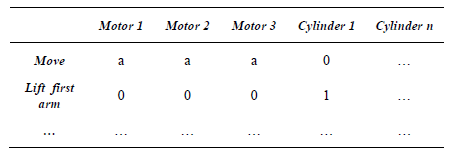
The “fact” base is made up of lines library and action library. The numbered lines and the barriers within the line segment are stored in the lines library while the action library stores the motion commands of executive components in different states of motion.

TABLE I. THE LINES LIBRARY



In Table I, abscissa represents different barriers and ordinate represents the lines number.

TABLE II. THE ACTION LIBRARY

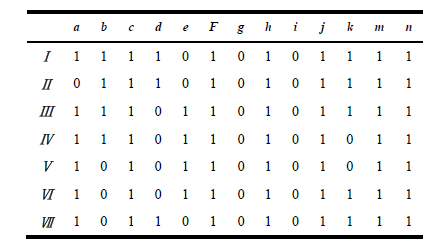


In Table II , abscissa represents all executive components including three motors and nine cylinders. The ordinate represents the standard state of motion.

**ii) The realization of fact base**

The “fact” base is realized by tabulation method.According to the analysis of the state of the motion and repetitive tests, the sensor signal level in different states of the motion is listed by tabulation method. For example,when the robot comes across a suspension clamp, the fact base is as shown in TableIII:

TABLE III. THE STANDARD STATE OF MOTION



In Table III, abscissa represents the detection information of the robot sensor system (low level is

effective.). The ordinate represents the standard state in combination of different status messages.

The facts stored in the tabulation are translated into information that the inference engine can recognize. This information is stored in the facts list. We can use “assert” to add new facts [8].

(assert<<<fact>>>)

**C. The Realization of Inference Engine**

The inference engine is the organizational control organ of the expert system and has general control knowledge of problem solving. It is established in a forward reasoning way. It uses the rules from the standard state of motion in different sensors information. The basic idea is:

• Put the standard state of motion in the general database.

• According to the knowledge that is already stored in the general database, search for the usable knowledge in knowledge database.

• Group them as available knowledge sets.

• Choose some knowledge for reasoning in knowledge sets according to conflict resolution strategies.

• Then add the new facts that are reasoned out in general databases for future inference. Repeat thisprocess until the needed result is achieved or there is no usableknowledge in knowledge base.The reasoning flow chart is shown in Fig. 4:

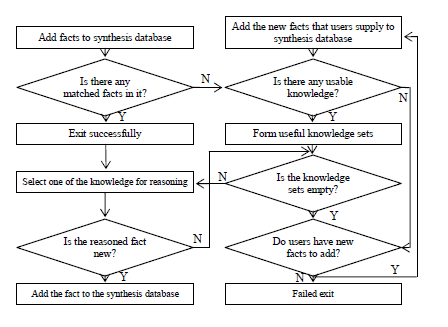


Figure 4. The flow chart of inference

Though CLIPS has the great advantage in formulating knowledge base and fact base, it is not as efficient as VC++ in data acquisition and processing and humancomputer interface design. Therefore, the expert system design adopts a mixed-languages programming strategy.

CLIPS can be fitted into VC++ in two ways [9]:either it can be embedded directly or indirectly into VC++. Here we use indirect embedding method to design the interface. Implementation is as follows:

• First, download software development environment and source code, DLL (clips.dll), which is provided by CLIPS develope􀁕􀀃 􀀾􀀔􀀓􀁀. This will be added into newly-built project documentation.

• Second, set newly-built project, and connect clips.lib to the project. Finally, initialize the class file

CCLIPSWrap. Then the core functions such as RUN, LOAD and RESET can be called directly in VC++. The initialize program is as follows:

CString sDisplay;

m\_pClips=new CCLIPSWrap();

CStringArray m\_ddisplay;

if(m\_pClips->CLIPSInit())

sDisplay="CLIPS load sucessfully！";

else

sDisplay="CLIPS failed load ";

……

m\_pClips->CLIPSExit();

return FALSE;

m\_display+="\r\n it can run！";

m\_Display.SetWindowText(m\_display);

UpdateData(FALSE);

• Save the rule and the facts which are programmed by CLIPS in “rule.txt” and “fact.txt”. Copy the two test files to the directory of a new project that is built by VC++.

• Edit codes in VC++:

CString m\_eFile="rule.txt";

m\_pClips->CLIPSClear();

iErrCode=m\_pClips->CLIPSLoad(m\_eFile);

…

m\_pClips->CLIPSExit();

return FALSE;

• When the rules are loaded successfully, we should load the facts for reasoning. The reasoning result is saved in “result.txt”. The codes are as follows:

CString facts="fact.txt", strOutFile="result.txt";

m\_pClips->CLIPSLoadFacts(facts);

m\_pClips->CLIPSDribble(strOutFile,TRUE);

m\_pClips->CLIPSRun();

m\_pClips->CLIPSDribble(strOutFile,FALSE);

FILE \*p;

p=fopen("result.txt","r");

• The robot moves independently according to thereasoning result.

IV. THE EXPERIMENTAL TEST AND RESULTS

**A. The working mode of the robot**

According to the human experience, we plan the working procedure for the robot. Take the first arm as an example, its working procedure is shown in Fig. 5. If sensor a has signal, the inference engine is started and collects the signal of sensor a to sensor m to stored in the fact base. Compare these facts with rules, and trigger the matched rules. Then store the result in result.txt file. At last the robot acts as the result of the judgment subroutine.

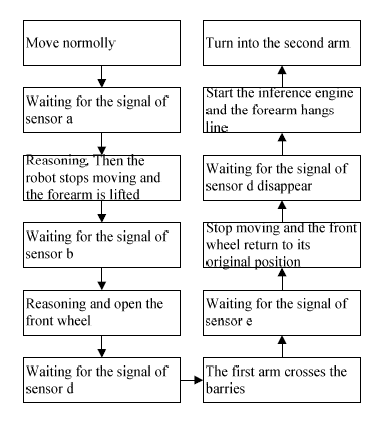


Figure 5. The operation flow chart of robot

**B. The analysis of experimental results**

We checked the research method in laboratory simulation lines. The result is shown in Fig. 6 which shows the process of crossing Cable Clamp of the first arm. Fig. 6 (a) is the static state of robot. This moment the infrared sensor in front wheel detects obstacle, so the signal turns to 0 from 1. Then one of the facts in the facts list is（0,1,1,1,0,1,0,1,0,1,1,1,1）. At the same time the contents saved in “fact.txt” also changed. The inference engine compares this fact with rules and triggers the rule- 2 that is “raise the first arm” as Fig. 6 (b) shows. Then the reasoning result is saved in “result.txt”. In the state of Fig. 6 (b) the two limit switches in the first arm turn 1 to 0 and 0 to 1. The limit switch in first wheel turns from 1 to 0. So, the fact is (1,1,1,0,1,1,0,1,0,1,0,1,1). Then the rule of

“the robot is moving forward” is matched. So the state of motion is shown in Fig. 6 (c). This moment the infrared sensor in second wheel detects obstacle and the robot stops to move waiting for the first arm to hang line as Fig.6 (d) shows. The principle of Fig. 6 (c) and (d) is the same as (a) and (b).





Figure 6. The first arm comes across the barrier

V. CONCLUSIONS

An expert system based control strategy has been proposed in the paper. Through experimental study, the expert system based control has a reasonable simple structure for implementation and can realize intelligent control for the robot motion. It provides an efficient way to guide, control and drives the robot to crossing obstacles smoothly along the tested high-voltage transmission lines.

REFERENCES

[1] Tu X.Y., Artificial intelligence and its application. Beijing: PHEI,1994

[2] Durkin, J., Expert System Design and Development, New York:Macmillan Publishing Company, 1994.

[3] Wang, J.D., Sun, A., Zheng, C. and Wang, J., Research on a newcrawler type inspection robot for power transmission lines, *the* *Proceedings of IEEE International Conference on the Applied Robotics for the Power Industry*, Montreal, Canada, October, 5-7,2010.

[4] Zhou, F.Y., Wang, J.D., Li, Y.B. and Wang, J., Control of an Inspection Robot for 110KV Power Transmission Lines Based on Expert System Design Methods, *IEEE Conference on Control* *Applications*, Canada, August, 2005.

[5] Li W.Z., Design and the study of the control system of a power transmission line inspection robot. Beijing: Manufacturing

automation, 2008.

[6] Giarratano, J. C., The principle and programming of Expert system. Beijing: China Machine Press, 2000.

[7] Villa, A., Expert control system: A rationale to handle system uncertainty and control structure complexity, IEEE Trans SMC,

1987, 17-28.

[8] Cai Z.X., Fu K S., Robot planning expert systems. Proc of IEEE Int Conf on Robotics and Automation, Vol. 3: 1973-1978, 7-10.

[9] Sardis, G N. Intelligent robotic control, IEEE Trans AC, 1983, 28(5): 547-557.

[10] Sawada, J., Kusumoto, K., Munakata, T., A mobile robot for inspection of power transmission lines. IEEE Translation on

Power Delivery, 1991, 6(1): 309-315.

### 二、英文翻译

**第18届国际会议论文集自动化与计算**

**拉夫堡大学，莱斯特郡，英国，2012年9月8日**

**关于控制高压输电线路巡线机器人跨越障碍的发展略**

王吉岱 孙爱琴 苏伟

山东科技大学机电工程学院

中国，青岛，266590 ，Djdwang8911@sina.com

王吉鸿 刘浩

英国华威大学工程学院

英国 ， 诺思 ，CV4 7AL, [jihong.Wang@warwick.ac.uk](mailto:jihong.Wang@warwick.ac.uk)

**摘 要**

一个专家以控制系统为基础设计了控制一个爬行机器人可以跨越障碍以及高电压输电线路，本研究的目的是通过使用远程控制检查机器人来实现对线路故障的自动检查，专家们提出了很多的控制策略而本文采用了人工智能恶化嵌入式的算法在机器人的控制上面。专家们对机器人的架构进行探讨推导出了机器人的设计方法知识库和推理引擎，VC++和CLIPS被用作程序语言设计对于软件设计和安装。对专家系统的可行性和实用性进行了测试，实验研究还通过了验证。实验结果表明，该控制策略的效果很好，性能让人满意，并且在应用实践中有巨大潜力。

【**关键字**】控制机器人 人工智能 专家系统 知识库 推理机构

**I介绍**

近些年来，在理论研究和实践中基于专家系统控制方法的应用取得显着进展，专家系统的引入是人工智能在应用程序中的一个转折点，历史导致AI运用专业的知识进行探索，从理论研究到系统的实际设计。专家系统是一个典型知识引擎系统，这是知识对表达对象进行全面的存储，推理，采集和管理。事实上，这是一个科技工具对于搜索和发展知识库控制和更新系统。它让我们意识到一个现实就是计算机软件，硬件和从数据信息系统处理知识信息的发展。专家系统以前被用作故障诊断，容错控制，翻译和分析系统领域。有较少的应用案例出现在机器人的运动控制除了机器人路径发现和决策。

尽管我们发现，一个高压输电线路的检查机器人被设计如下图，然而，这是一个极大的挑战任务对于机器人工作在一个复杂的环境中，例如，越过障碍，移动检查，在不同的天气换镜下操作等等。为了克服这些挑战，这份报告显示，我们在利用专家系统研究的同时要设计一个智能的规则控制方法用于控制驱动机器人穿越障碍物，同时它是高压输电线路爬行。该模拟研究表明，这种方法是机器人完成障碍穿越任务的有效的方法以一种智能的方式。

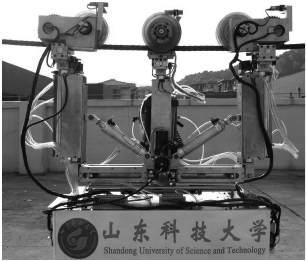


图1 输电线路巡检机器人图

**II专家系统的结构基于控制策略**

**A.专家系统的组件**

本节介绍专家系统的组件，这个专家系统是在控制的基础上发展的医德具有规则的控制系统，它允许更灵活的控制结构（例如心得规则能允许增加任何设计方案）和在人工智能中决策机制中合作。知识能被如果｛…..｝然后标语所表达，这些是被人类的思维用简单地方式表达出来。这些逻辑被用于推导行为、决定和知识是从人类的经验中获得的，通过推理机制的基础得到，用一种简单的额方式表达，专家系统表示如下；

专家系统=知识+推理

为了提炼专家系统的设计，该系统被扩展的结构图如图2，除了知识库和推理机制，该系统由一个静态的数据库、动态的数据库、解释机制和人际界面组成。

信息传感器

控制命令

人工界面

解释机制

推理机制

知识库

静态数据库

动态数据库

图2 基础专家系统控制框图

**B.系统组件的功能**

为了分配和描述系统每个组件的功能，第一步是对测试线路编号，然后提取与障碍相关的东西。动态标准是例举信息表通过专家的经验，上述的知识都是静态的知识，这些信息都需要去翻译成机器语言而且被存储在静态数据库里，然而事实上，它是高效、简洁的规则建立，这些规则被存储在基础知识当中，下一步是传感器的比较信息也录进基础信息里。如果比较信息对应与相应的规则信息在知识库中，推理机制将给予匹配规则的优先级，机器人将根据规则最高优先级表示。同时新的信息也就是所产生的推理机制将存储在动态数据库中，动态数据库存储传感器信息，而且将接收在对的时间里和所控制的命令，为了将它们连在一起，就是解释机制和人机界面，人机界面将帮助人们与机器沟通也就是解释机器人的动作状态系统的错误，它将帮助使用去测试、更新和保持专家系统。

**III .知识库**

**A.规则库的实现**

专家系统是工作在数据库中，知识库和规则被用在推理中。机器人的 动态应该是适应根据环境来检查传输线路，当然，规则系统的建立是为了改变机器人的动作和姿势。

如图3所示，它是专为三层操作模式的结构规则库二设计的，是工作类别和操作规则。机器人操作模型和工作策略被红外线传感器和可视处理所区分，这些动作是在模仿人工在越障和处理的过程。为了应对环境的变化和对动作姿势的适应。因此我们在机器人的相应部分放了不同的传感器。

直行

夹紧

动作1…..n

动作1…..n

长冰柱

冰盖

除冰

动作1…..n

防震

动作1…..n

穿越障碍

动作1…..n

90度转弯

动作1…..n

正常行走

操作模型层 工作策略层 动作规则层

图3 规则库架构

这个规则被用 IF{……}THEN 格式通过CLIPS(C语言综合生成系统)。它们有下面的描述和表达：

（(defrule <rule name > [<comment>]

<patterns>\*;// triggered instructions of the rule

=>

<actions>\*); // alterative action of the rule

**B.组合和事实实现库**

**i)全自动校准技术组合库**

全自动校准技术库由线路图和动作图组成。编号线和内部障碍线段存储在线库中，而动作库存储的执行的动作命令的组件使用不同的动作状态。

表1 路线库

屏障1 屏障2 屏障3 屏障n

线路1 振动 电缆 绝缘体 …………

甲板 钳子 绳子 …………

线路2 电缆 振动 绝缘体 …………

钳子 甲板 绳子 …………

线路n …… …… …… …………

在表1中，横坐标代表不同的障碍，纵坐标代表不同的线路标号。

表2 动作库

电机1 电机2 电机3 柱面1 柱面n

移动 a a a 0 …

首臂 0 0 0 1 …

… … … … … …

在表2中，横坐标代表所有执行组件包括3个电机和9个柱面。纵坐标代表标准的运动状态。

**ii)自动校准技术的实现基础**

自动校准技术库实现了制表方法，根据动作的分析标准和重复任务。传感器信号水平在不同标准状态中被例为制表方法。例如，当机器人穿越悬垂线夹时，情况如表3；

表3 标准的运动状态

a b c d e f g h I j k m n

I 1 1 1 1 0 1 0 1 0 1 1 1 1

II 0 1 1 1 0 1 0 1 0 1 1 1 1

III 1 1 1 0 1 1 0 1 0 1 1 1 1

IV 1 1 1 0 1 1 0 1 0 1 0 1 1

V 1 0 1 0 1 1 0 1 0 1 0 1 1

VI 1 0 1 0 1 1 0 1 0 1 1 1 1

VII 1 0 1 1 0 1 0 1 0 1 1 1 1

在表3中，横坐标代表机器人传感器系统信息（低电平有效），纵坐标代表标准状态时的不同信息状态的组合。

存储在表中的信息被翻译成可以被推理机制识别的信息，这些信息被存储在表中，我们可以使用“assert”去增加新的事物。

（assert<<<fact>>>）

**C.推理机制的实现**

推理机制是组织控制专家系统的器官，简单地控制了知识的解决方案。它是建立在一个正向的推理方式，它利用规则建立标准状态在不同的传感器信息，基本点是；

1.把动态动态的标准放在一般的数据库中。

2.根据知识也就是已经存储在数据库中的有用的知识在知识数据库中。

3.选择一些有关推理的知识，根据冲突解决策略来设置。

4.把新增加的项目也就是被推理用一般的数据库作为以后的推理，重复这个过程，直到所需的结果达到或没有可用的知识在知识库中。

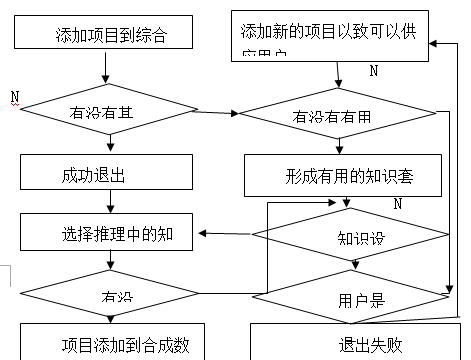


图4.推断的流程图

虽然CLIPS的制定有很大的优势在知识库和项目库， 它不像VC++那样有效在数据采集处理和人机接口设计。因此，专家系统的设计采用混合语言编程的策略。

CLIPS能装配到VC++中用两种方式：或者它可以直接或者间接进入VC++。这里我们使用间接埋线法来设计接口。实现如下；

• 首先，下载软件开发环境和源代码，DLL（clips,dll），它被CLIPS开发者提供。这将被添加到新的项目文档。

• 第二步，设置新建文档，并链接到clips.lib到新项目。最后，初始化类文件CCLIPSWrap。接着就是核心功能，例如运行、下载和重置 可以直接在 VC++中调用，初始化程序如下；

CString sDisplay;

m\_pClips=new CCLIPSWrap();

CStringArray m\_ddisplay;

if(m\_pClips->CLIPSInit())

sDisplay="CLIPS load sucessfully􀊽";

else

sDisplay="CLIPS failed load ";

……

m\_pClips->CLIPSExit();

return FALSE;

m\_display+="\r\n it can run􀊽";

m\_Display.SetWindowText(m\_display);

UpdateData(FALSE);

• 保存规则和 项目被CLIPS编入“rule.txt” 和 “fact.txt”. 复制两个测试文件也就是建立在一个新的项目的目录VC++。

• 编辑代码在VC++中:

CString m\_eFile="rule.txt";

m\_pClips->CLIPSClear();

iErrCode=m\_pClips->CLIPSLoad(m\_eFile);

…

m\_pClips->CLIPSExit();

return FALSE;

• 当规则被下载成功，我们应该下载推理项目推理结构保存在 “result.txt” ，代码如下；

CString facts="fact.txt", strOutFile="result.txt";

m\_pClips->CLIPSLoadFacts(facts);

m\_pClips->CLIPSDribble(strOutFile,TRUE);

m\_pClips->CLIPSRun();

m\_pClips->CLIPSDribble(strOutFile,FALSE);

FILE \*p;

p=fopen("result.txt","r");

• 机器人独立移动根据推理结果。

**IV .实验测试和结果**

**A.机器人的工作原理**

据人类的实验测试，我们对机器人制定了工作程序，就拿第一臂作为例子，例如它的工作过程展示如图5 。如果传感器具有信号 ，推理机制启动并且收集传感器的信号存储在项目库中，比较这些项目与规则，并触发匹配的规则,然后将文件存储在文件‘txt.file’中，在最据人类的实验测试，我们对机器人制定了工作程序，就拿第一臂作为例子，例如它的工作过程展示如图5 。如果传感器具有信号 ，推理机制启动并且收集传感器的信号存储在项目库中，比较这些项目与规则，并触发匹配的规则后机器人充当判定子程序的结果;

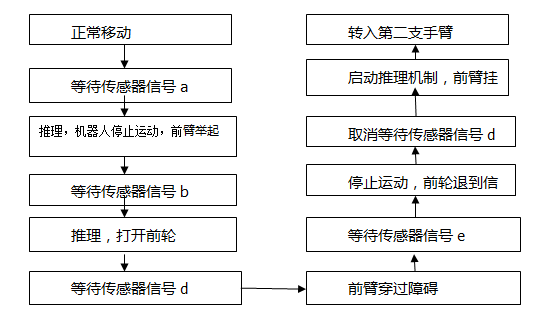


图5.机器人的操作流程图

**B.实验结果分析**

我们检查推理方法在实验室模拟线上，结果如图6展示首臂穿越电缆线夹的过程，图6（a）是机器人的静止状态，这时前轮的红外传感器检测到障碍物，所以信号从1变为0，所以这个项目在项目表是（0,1,1,1，0，1,0,1,1,1）.同时目录被存储在“fact.txt”中，也发生了变化，推理机制这一项目与规则和触发器比较，规则2就是举起第一臂在图6（b）显示，推理结果被存储在“result.txt”.在图6（b）状态中两个限制变化在第一臂中从1到0.所以数据是（1,1,1,0,1,1,0,1，0,1,0,1,1）。规则是“机器人向前移动”被匹配，所以运动状态如图6（c）所示，这一刻红外传感器在第二轮检测到障碍物，机器人停止移动等待第一臂挂线如图6（d）所示。图6（c）（d）的原理与图6（a）（b）相似。





图6.首臂穿越障碍

**V.结 论**

以专家系统为基础的控制战略已经在文中提出，通过实验得知，以该专家系统为基础的控制有一个简单合理结构的实施，可以实现智能化控制机器人的运动，它提供了一种有效的方式去引导、控制和驱动机器人来穿越障碍，以顺利的沿着测试高压输出电路。

**参考文献**

[1] Tu X.Y., 人工智能及其应用.北京：PHEI,1994.

[2] Durkin, J.,专家控制系统的设计与发展, New York:Macmillan Publishing Company, 1994.

[3] Wang, J.D., Sun, A., Zheng, C. and Wang, J., 研究新履带式巡检机器人的电力传输线路, *the Proceedings of IEEE International Conference on the AppliedRobotics for the Power Industry*, Montreal, Canada, October, 5-7,2010.

[4] Zhou, F.Y., Wang, J.D., Li, Y.B. and Wang, J., Control of an

Inspection Robot for 110KV Power Transmission Lines Based on Expert System Design Methods, *IEEE Conference on Control* *Applications*, Canada, August, 2005.

[5] Li W.Z., 输电线路巡检机器人的控制系统的研究和设计. 北京: Manufacturing automation, 2008.

[6] Giarratano, J. C., 专家系统的原理及其编程 . 北京: 中国机械工业出版社, 2000.

[7] Villa, A., 专家控制系统: 合理地处理系统的不确定性和控制结构的复杂性, IEEE Trans SMC,1987, 17-28.

[8] Cai Z.X., Fu K S., 专家系统对机器人的规划. Proc of IEEE Int Conf on Robotics and Automation, Vol. 3: 1973-1978, 7-10.

[9] Sardis, G N. 智能机器人控制, IEEE Trans AC, 1983,28(5): 547-557.

[10] Sawada, J., Kusumoto, K., Munakata, T., 移动机器人对输电线路的检查 . IEEE Translation on Power Delivery, 1991, 6(1): 309-315.

### 三、源程序

1.主程序

#include "stm32f10x.h"

#include "led.h"

#include "PWM.h"

#include "DCMotor.h"

#include "global.h"

#include "VoiceControl.h"

#include "Infrared.h"

#include "USART.h"

//---------------------------------------------------------------//

//- 函数名称： STM32\_hardInit

void STM32\_hardInit(void);

//---------------------------------------------------------------//

//- 函数名称： STM32\_softInit

void STM32\_softInit(void);

//---------------------------------------------------------------//

//- 函数名称： mainLoop

void mainLoop(void);

void KeyInit(void);

void ModeChangeByKey(void);

void TIM4\_Configuration(uint16\_t MS);

//----------------------------------------------------------//

//main entrance

int main(void)

{

//----------------------------------------------------------//

//variable define

//----------------------------------------------------------//

//STM32 hard init

STM32\_hardInit();

//----------------------------------------------------------//

//STM32 soft init

STM32\_softInit();

//----------------------------------------------------------//

//main loop

mainLoop();

}

//---------------------------------------------------------------//

//- 函数名称： STM32\_hardInit

//---------------------------------------------------------------//

//- 函数功能： 硬件初始化

//---------------------------------------------------------------//

//- 函数参数：

//---------------------------------------------------------------//

//- 返回值 ： 无

void STM32\_hardInit(void)

{

//系统初始化

SystemInit();

//避障传感器

Infrared\_Config(InfraredLeft\_RCC, InfraredLeft\_GPIO, InfraredLeft\_Pin, GPIO\_Mode\_IN\_FLOATING);

Infrared\_Config(InfraredMiddle\_RCC, InfraredMiddle\_GPIO,InfraredMiddle\_Pin, GPIO\_Mode\_IN\_FLOATING);

Infrared\_Config(InfraredRight\_RCC, InfraredRight\_GPIO, InfraredRight\_Pin, GPIO\_Mode\_IN\_FLOATING);

//循迹传感器

Infrared\_Config(LookingForTrack\_1\_RCC, LookingForTrack\_1\_GPIO, LookingForTrack\_1\_Pin, GPIO\_Mode\_IN\_FLOATING);

Infrared\_Config(LookingForTrack\_2\_RCC, LookingForTrack\_2\_GPIO, LookingForTrack\_2\_Pin, GPIO\_Mode\_IN\_FLOATING);

Infrared\_Config(LookingForTrack\_3\_RCC, LookingForTrack\_3\_GPIO, LookingForTrack\_3\_Pin, GPIO\_Mode\_IN\_FLOATING);

Infrared\_Config(LookingForTrack\_4\_RCC, LookingForTrack\_4\_GPIO, LookingForTrack\_4\_Pin, GPIO\_Mode\_IN\_FLOATING);

Infrared\_Config(LookingForTrack\_5\_RCC, LookingForTrack\_5\_GPIO, LookingForTrack\_5\_Pin, GPIO\_Mode\_IN\_FLOATING);

//LED灯

LED\_Config(RCC\_APB2Periph\_GPIOC, LED\_3\_GPIO, LED\_3\_Pin);

//电机pwm控制

PWM\_Config();

//

// KeyInit();

TIM4\_Configuration(100);

}

//---------------------------------------------------------------//

//- 函数名称： STM32\_softInit

//---------------------------------------------------------------//

//- 函数功能： 软件数据初始化

//---------------------------------------------------------------//

//- 函数参数：

//---------------------------------------------------------------//

//- 返回值 ： 无

void STM32\_softInit(void)

{

//

// //初始化为声控模式

// show(2,voiceControlModeStr);

//

DCMotor\_stop();

}

//---------------------------------------------------------------//

//- 函数名称： mainLoop

//---------------------------------------------------------------//

//- 函数功能： 主循环

//---------------------------------------------------------------//

//- 函数参数：

//---------------------------------------------------------------//

//- 返回值 ： 无

void mainLoop(void)

{

int32\_t mainCounter = 0;

// DCMotor\_Forward(0, 60000);

// while(1);

//----------------------------------------------------------//

//main loop

while(1)

{

//主循环计数累加

mainCounter++;

//主循环计数处理

if(mainCounter >= 100000)

{

//主循环计数清零

mainCounter = 0;

LED\_toggle(LED\_3\_GPIO, LED\_3\_Pin);

}

//ModeChangeByKey();

avoidMode();

}

}

void KeyInit(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOE, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_12;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_ResetBits(GPIOB,GPIO\_Pin\_12);

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_15;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_ResetBits(GPIOE,GPIO\_Pin\_15);

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_ResetBits(GPIOB,GPIO\_Pin\_10);

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_13;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_ResetBits(GPIOE,GPIO\_Pin\_13);

GPIO\_Init(GPIOE, &GPIO\_InitStructure);

// GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3;

// GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPD;

// GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

// GPIO\_ResetBits(GPIOC,GPIO\_Pin\_3);

// GPIO\_Init(GPIOC, &GPIO\_InitStructure);

}

void ModeChangeByKey(void)

{

// if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_12) == Bit\_SET)

// {

// delay\_ms(50);

// if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_12) == Bit\_SET)

// {

// if(mode == MODE\_VOICE\_CONTROL)

// {

// voiceCmd = VOICE\_CMD\_LOOKING\_FOR\_TRACK;

// newVoiceCmdFlag = 1;

// }

// else if(mode == MODE\_LOOKING\_FOR\_TRACK)

// {

// voiceCmd = VOICE\_CMD\_VOICE\_CONTROL;

// newVoiceCmdFlag = 1;

// }

// else

// {}

// }

// while(GPIO\_ReadInputDataBit(GPIOA,GPIO\_Pin\_11) == Bit\_SET);

// }

if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_12) == Bit\_SET)

{

delay\_ms(50);

if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_12) == Bit\_SET)

{

//A

newVoiceCmdFlag = 1;

voiceCmd = VOICE\_CMD\_BACK;

}

while(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_12) == Bit\_SET);

}

if(GPIO\_ReadInputDataBit(GPIOE,GPIO\_Pin\_15) == Bit\_SET)

{

delay\_ms(50);

if(GPIO\_ReadInputDataBit(GPIOE,GPIO\_Pin\_15) == Bit\_SET)

{

//B

newVoiceCmdFlag = 1;

voiceCmd = VOICE\_CMD\_TURNRIGHT;

}

while(GPIO\_ReadInputDataBit(GPIOE,GPIO\_Pin\_15) == Bit\_SET);

}

if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_10) == Bit\_SET)

{

delay\_ms(50);

if(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_10) == Bit\_SET)

{

//C

newVoiceCmdFlag = 1;

voiceCmd = VOICE\_CMD\_FORWARD;

}

while(GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_10) == Bit\_SET);

}

if(GPIO\_ReadInputDataBit(GPIOE,GPIO\_Pin\_13) == Bit\_SET)

{

delay\_ms(50);

if(GPIO\_ReadInputDataBit(GPIOE,GPIO\_Pin\_13) == Bit\_SET)

{

//D

newVoiceCmdFlag = 1;

voiceCmd = VOICE\_CMD\_TURNLEFT;

}

while(GPIO\_ReadInputDataBit(GPIOE,GPIO\_Pin\_13) == Bit\_SET);

}

}

//

void TIM4\_Configuration(uint16\_t MS)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM4, ENABLE);

NVIC\_InitStructure.NVIC\_IRQChannel = TIM4\_IRQn; //

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1; //抢占优先级1

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0; //响应优先级0

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

/\* Time Base configuration \*/

TIM\_DeInit(TIM4);

/\*\*/

//MS = 1000

//(7200-1+1)/72\*(1000\*10-1+1) = 1000000us = 1000ms = 1s

TIM\_TimeBaseStructure.TIM\_Prescaler = 7200-1; //设置预分频器分频系数71，即APB2=72M, TIM1\_CLK=72/72=1MHz ，

//它的取值必须在0x0000和0xFFFF之间

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up; // 向上计数模式

TIM\_TimeBaseStructure.TIM\_Period = MS\*10-1; // 1s定时 它的取值必须在0x0000和0xFFFF之间

TIM\_TimeBaseStructure.TIM\_ClockDivision = 0x0; //设置了定时器时钟分割，

TIM\_TimeBaseStructure.TIM\_RepetitionCounter = 0x0; // 设置了周期计数器值，它的取值必须在0x00和0xFF之间。

TIM\_TimeBaseInit(TIM4,&TIM\_TimeBaseStructure); // 根据TIM\_TimeBaseInitStruct中指定的参数初始化TIMx的时间基数单位

TIM\_ClearFlag(TIM4, TIM\_FLAG\_Update); //清中断，以免一启用中断后立即产生中断

TIM\_ITConfig(TIM4, TIM\_IT\_Update, ENABLE); //使能TIM1中断源

TIM\_Cmd(TIM4, ENABLE); //TIM5总开关：开启

}

2.STM程序

/\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @file Project/STM32F10x\_StdPeriph\_Template/stm32f10x\_it.c

\* @author MCD Application Team

\* @version V3.5.0

\* @date 08-April-2011

\* @brief Main Interrupt Service Routines.

\* This file provides template for all exceptions handler and

\* peripherals interrupt service routine.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* @attention

\*

\* THE PRESENT FIRMWARE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS

\* WITH CODING INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE

\* TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY

\* DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING

\* FROM THE CONTENT OF SUCH FIRMWARE AND/OR THE USE MADE BY CUSTOMERS OF THE

\* CODING INFORMATION CONTAINED HEREIN IN CONNECTION WITH THEIR PRODUCTS.

\*

\* <h2><center>&copy; COPYRIGHT 2011 STMicroelectronics</center></h2>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*/

/\* Includes ------------------------------------------------------------------\*/

#include "stm32f10x\_it.h"

#include "global.h"

#include "led.h"

/\*\* @addtogroup STM32F10x\_StdPeriph\_Template

\* @{

\*/

/\* Private typedef -----------------------------------------------------------\*/

/\* Private define ------------------------------------------------------------\*/

/\* Private macro -------------------------------------------------------------\*/

/\* Private variables ---------------------------------------------------------\*/

/\* Private function prototypes -----------------------------------------------\*/

/\* Private functions ---------------------------------------------------------\*/

\_\_IO uint32\_t Index;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Cortex-M3 Processor Exceptions Handlers \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief This function handles NMI exception.

\* @param None

\* @retval None

\*/

void NMI\_Handler(void)

{

}

/\*\*

\* @brief This function handles Hard Fault exception.

\* @param None

\* @retval None

\*/

void HardFault\_Handler(void)

{

/\* Go to infinite loop when Hard Fault exception occurs \*/

while (1)

{

}

}

/\*\*

\* @brief This function handles Memory Manage exception.

\* @param None

\* @retval None

\*/

void MemManage\_Handler(void)

{

/\* Go to infinite loop when Memory Manage exception occurs \*/

while (1)

{

}

}

/\*\*

\* @brief This function handles Bus Fault exception.

\* @param None

\* @retval None

\*/

void BusFault\_Handler(void)

{

/\* Go to infinite loop when Bus Fault exception occurs \*/

while (1)

{

}

}

/\*\*

\* @brief This function handles Usage Fault exception.

\* @param None

\* @retval None

\*/

void UsageFault\_Handler(void)

{

/\* Go to infinite loop when Usage Fault exception occurs \*/

while (1)

{

}

}

/\*\*

\* @brief This function handles SVCall exception.

\* @param None

\* @retval None

\*/

void SVC\_Handler(void)

{

}

/\*\*

\* @brief This function handles Debug Monitor exception.

\* @param None

\* @retval None

\*/

void DebugMon\_Handler(void)

{

}

/\*\*

\* @brief This function handles PendSVC exception.

\* @param None

\* @retval None

\*/

void PendSV\_Handler(void)

{

}

/\*\*

\* @brief This function handles SysTick Handler.

\* @param None

\* @retval None

\*/

void SysTick\_Handler(void)

{

}

void DMA1\_Channel1\_IRQHandler(void)

{

}

void TIM4\_IRQHandler(void)

{

// static uint16\_t i = 0;

if(TIM\_GetFlagStatus(TIM4,TIM\_FLAG\_Update))//

{

TIM\_ClearITPendingBit(TIM4, TIM\_IT\_Update); //清中断

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* STM32F10x Peripherals Interrupt Handlers \*/

/\* Add here the Interrupt Handler for the used peripheral(s) (PPP), for the \*/

/\* available peripheral interrupt handler's name please refer to the startup \*/

/\* file (startup\_stm32f10x\_xx.s). \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* @brief This function handles PPP interrupt request.

\* @param None

\* @retval None

\*/

/\*void PPP\_IRQHandler(void)

{

}\*/

/\*\*

\* @}

\*/

3.PWM程序

#include "PWM.h"

void PWM\_Config(void)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;

TIM\_OCInitTypeDef TIM\_OCInitStructure;

GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1|GPIO\_Pin\_4|GPIO\_Pin\_5;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //????

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //50M????

GPIO\_Init(GPIOB, &GPIO\_InitStructure);

TIM\_TimeBaseStructure.TIM\_Period = 65535;

TIM\_TimeBaseStructure.TIM\_Prescaler = 256;

TIM\_TimeBaseStructure.TIM\_ClockDivision = 2;

TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;

TIM\_TimeBaseInit(TIM3, &TIM\_TimeBaseStructure);/\* Channel 1 Configuration in PWM mode \*/

TIM\_OCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1; //PWM mode 2

TIM\_OCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable; //

TIM\_OCInitStructure.TIM\_OutputNState = TIM\_OutputNState\_Enable;//

TIM\_OCInitStructure.TIM\_Pulse = 0; //

TIM\_OCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High; //????

TIM\_OCInitStructure.TIM\_OCNPolarity = TIM\_OCNPolarity\_Low; //??????

TIM\_OCInitStructure.TIM\_OCIdleState = TIM\_OCIdleState\_Set;

TIM\_OCInitStructure.TIM\_OCNIdleState = TIM\_OCIdleState\_Reset;

TIM\_OC1Init(TIM3,&TIM\_OCInitStructure); //??1

TIM\_OC1PreloadConfig(TIM3, TIM\_OCPreload\_Enable);

/\* TIM2 counter enable \*/

TIM\_Cmd(TIM3,ENABLE);

TIM\_ITConfig(TIM3, TIM\_IT\_CC1 | TIM\_IT\_CC2 | TIM\_IT\_CC3 | TIM\_IT\_CC4, ENABLE);

}

4.直流电机程序

#include "DCMotor.h"

void DCMotor\_LeftWheelForward(uint16\_t speed)

{

TIM\_SetCompare3(TIM3,0);

TIM\_SetCompare4(TIM3,speed);

TIM\_SetCompare3(TIM3,speed);

TIM\_SetCompare4(TIM3,0);

}

void DCMotor\_LeftWheelBack(uint16\_t speed)

{

TIM\_SetCompare1(TIM3,0);

TIM\_SetCompare2(TIM3,speed);

TIM\_SetCompare1(TIM3,speed);

TIM\_SetCompare2(TIM3,0);

}

void DCMotor\_RightWheelForward(uint16\_t speed)

{

TIM\_SetCompare3(TIM3,0);

TIM\_SetCompare4(TIM3,speed);

TIM\_SetCompare3(TIM3,speed);

TIM\_SetCompare4(TIM3,0);

}

void DCMotor\_RightWheelBack(uint16\_t speed)

{

TIM\_SetCompare1(TIM3,0);

TIM\_SetCompare2(TIM3,speed);

TIM\_SetCompare1(TIM3,speed);

TIM\_SetCompare2(TIM3,0);

}

void DCMotor\_Forward(uint16\_t levelLeft,uint16\_t levelRight)

{

DCMotor\_LeftWheelForward(levelLeft);

DCMotor\_RightWheelForward(levelRight);

}

void DCMotor\_Back(uint16\_t levelLeft,uint16\_t levelRight)

{

DCMotor\_LeftWheelBack(levelLeft);

DCMotor\_RightWheelBack(levelRight);

}

void DCMotor\_TurnLeft(uint16\_t levelLeft,uint16\_t levelRight)

{

DCMotor\_LeftWheelBack(levelLeft);

DCMotor\_RightWheelForward(levelRight);

}

void DCMotor\_TurnRight(uint16\_t levelLeft,uint16\_t levelRight)

{

DCMotor\_LeftWheelForward(levelLeft);

DCMotor\_RightWheelBack(levelRight);

}

void DCMotor\_stop(void)

{

TIM\_SetCompare1(TIM3,0);

TIM\_SetCompare2(TIM3,0);

TIM\_SetCompare3(TIM3,0);

TIM\_SetCompare4(TIM3,0);

}

5.字符串定义程序

#include "global.h"

uint8\_t carState = CAR\_STOP;

//模式标记，初始化为语音模式

uint8\_t mode = MODE\_LOOKING\_FOR\_TRACK;

//接收到语音命令标记，初始化第一条语音命令为停止

uint8\_t newVoiceCmdFlag = 0;

//语音命令，初始化第一条语音命令为停止

uint8\_t voiceCmd = VOICE\_CMD\_STOP;

//循迹传感器标志

uint8\_t lookingForTrackSensor[6] = {0};

//字符串定义

uint8\_t lookingForTrackCarStr[] = {" 循迹小车 "};

uint8\_t lookingForTrackModeStr[] = {" 循迹模式 "};

uint8\_t voiceControlModeStr[] = {" 避障模式 "};

uint8\_t lookingForTrackDetectFlagStr[] = "十";

uint8\_t lookingForTrackNoDetectFlagStr[] = " ";

uint8\_t \_100ms\_count = 0;

6.检测程序

#include "VoiceControl.h"

#define BACK\_MS\_WITH\_OBSTACLE 600

#define TURN\_MS\_WITH\_OBSTACLE 600

#define LEFT\_HIGH\_SPEED 64535

#define LEFT\_MEDIUM\_SPEED 34767

#define LEFT\_LOW\_SPEED 14383

#define RIGHT\_HIGH\_SPEED 64535

#define RIGHT\_MEDIUM\_SPEED 34767

#define RIGHT\_LOW\_SPEED 14383

void avoidMode(void)

{

uint16\_t backSecCount = 0;

uint16\_t turnSecCount = 0;

INFRARED\_STATE infraredLeft = Infrared\_isDetect(InfraredLeft\_GPIO,InfraredLeft\_Pin);

INFRARED\_STATE infraredMiddle = Infrared\_isDetect(InfraredMiddle\_GPIO,InfraredMiddle\_Pin);

INFRARED\_STATE infraredRight = Infrared\_isDetect(InfraredRight\_GPIO,InfraredRight\_Pin);

//如果红外检测到物体

if((infraredLeft ==INFRARED\_STATE\_DETECT ||

infraredMiddle ==INFRARED\_STATE\_DETECT ||

infraredRight ==INFRARED\_STATE\_DETECT)&&

(carState!=CAR\_STOP))

{

//先后退

DCMotor\_Back(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

for(backSecCount=0; backSecCount<BACK\_MS\_WITH\_OBSTACLE; backSecCount++)

{

delay\_ms(1);

}

//如果是左边检测到，则向右转

if(infraredLeft ==INFRARED\_STATE\_DETECT)

{

//向右转

DCMotor\_TurnRight(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

//延时一段时间

for(turnSecCount=0; turnSecCount<TURN\_MS\_WITH\_OBSTACLE; turnSecCount++)

{

delay\_ms(1);

}

}

//如果是右边检测到，则向左转

else

{

//向左转

DCMotor\_TurnLeft(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

//延时一段时间

for(turnSecCount=0; turnSecCount<TURN\_MS\_WITH\_OBSTACLE; turnSecCount++)

{

delay\_ms(1);

}

}

DCMotor\_Forward(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

}

if(newVoiceCmdFlag == 1)

{

newVoiceCmdFlag = 0;

switch(voiceCmd)

{

//前进

case VOICE\_CMD\_FORWARD:

DCMotor\_Forward(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

break;

//后退

case VOICE\_CMD\_BACK:

DCMotor\_Back(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

break;

//左转

case VOICE\_CMD\_TURNLEFT:

DCMotor\_TurnLeft(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

break;

//右转

case VOICE\_CMD\_TURNRIGHT:

DCMotor\_TurnRight(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

break;

//停止

case VOICE\_CMD\_STOP:

DCMotor\_stop();

break;

//制动

case VOICE\_CMD\_BRAKING:

DCMotor\_stop();

break;

//声控

case VOICE\_CMD\_VOICE\_CONTROL:

//show(2,voiceControlModeStr);

mode = MODE\_VOICE\_CONTROL;

break;

//循迹

case VOICE\_CMD\_LOOKING\_FOR\_TRACK:

//show(2,lookingForTrackModeStr);

mode = MODE\_LOOKING\_FOR\_TRACK;

break;

default:

break;

}

}

}

7.检测程序

#include "VoiceControl.h"

#define BACK\_MS\_WITH\_OBSTACLE 600

#define TURN\_MS\_WITH\_OBSTACLE 600

#define LEFT\_HIGH\_SPEED 64535

#define LEFT\_MEDIUM\_SPEED 34767

#define LEFT\_LOW\_SPEED 14383

#define RIGHT\_HIGH\_SPEED 64535

#define RIGHT\_MEDIUM\_SPEED 34767

#define RIGHT\_LOW\_SPEED 14383

void avoidMode(void)

{

uint16\_t backSecCount = 0;

uint16\_t turnSecCount = 0;

INFRARED\_STATE infraredLeft = Infrared\_isDetect(InfraredLeft\_GPIO,InfraredLeft\_Pin);

INFRARED\_STATE infraredMiddle = Infrared\_isDetect(InfraredMiddle\_GPIO,InfraredMiddle\_Pin);

INFRARED\_STATE infraredRight = Infrared\_isDetect(InfraredRight\_GPIO,InfraredRight\_Pin);

//如果红外检测到物体

if((infraredLeft ==INFRARED\_STATE\_DETECT ||

infraredMiddle ==INFRARED\_STATE\_DETECT ||

infraredRight ==INFRARED\_STATE\_DETECT)&&

(carState!=CAR\_STOP))

{

//先后退

DCMotor\_Back(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

for(backSecCount=0; backSecCount<BACK\_MS\_WITH\_OBSTACLE; backSecCount++)

{

delay\_ms(1);

}

//如果是左边检测到，则向右转

if(infraredLeft ==INFRARED\_STATE\_DETECT)

{

//向右转

DCMotor\_TurnRight(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

//延时一段时间

for(turnSecCount=0; turnSecCount<TURN\_MS\_WITH\_OBSTACLE; turnSecCount++)

{

delay\_ms(1);

}

}

//如果是右边检测到，则向左转

else

{

//向左转

DCMotor\_TurnLeft(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

//延时一段时间

for(turnSecCount=0; turnSecCount<TURN\_MS\_WITH\_OBSTACLE; turnSecCount++)

{

delay\_ms(1);

}

}

DCMotor\_Forward(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

}

if(newVoiceCmdFlag == 1)

{

newVoiceCmdFlag = 0;

switch(voiceCmd)

{

//前进

case VOICE\_CMD\_FORWARD:

DCMotor\_Forward(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

break;

//后退

case VOICE\_CMD\_BACK:

DCMotor\_Back(LEFT\_MEDIUM\_SPEED,RIGHT\_MEDIUM\_SPEED);

break;

//左转

case VOICE\_CMD\_TURNLEFT:

DCMotor\_TurnLeft(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

break;

//右转

case VOICE\_CMD\_TURNRIGHT:

DCMotor\_TurnRight(LEFT\_LOW\_SPEED,RIGHT\_LOW\_SPEED);

break;

//停止

case VOICE\_CMD\_STOP:

DCMotor\_stop();

break;

//制动

case VOICE\_CMD\_BRAKING:

DCMotor\_stop();

break;

//声控

case VOICE\_CMD\_VOICE\_CONTROL:

//show(2,voiceControlModeStr);

mode = MODE\_VOICE\_CONTROL;

break;

//循迹

case VOICE\_CMD\_LOOKING\_FOR\_TRACK:

//show(2,lookingForTrackModeStr);

mode = MODE\_LOOKING\_FOR\_TRACK;

break;

default:

break;

}

}

}

8.串口程序

#include "USART.h"

void USARTx\_Configuration(USART\_TypeDef\* USARTx,uint32\_t baud) //串口初始化函数

{

USART\_InitTypeDef USART\_InitStructure;

GPIO\_InitTypeDef GPIO\_InitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

//引脚复用使能

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE);

//中断优先级分组

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

//串口2相关初始化

if(USARTx == USART2)

{

//串口2时钟使能

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2, ENABLE);

//串口2引脚使能

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);

//串口2中断优先级设置

NVIC\_InitStructure.NVIC\_IRQChannel = USART2\_IRQn; //通道

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 0;//占先级

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 0; //响应

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //启动

NVIC\_Init(&NVIC\_InitStructure); //初始化

// GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1; //

// GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING; //

// GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

// GPIO\_Init(GPIOA, &GPIO\_InitStructure); //

//串口2的管脚初始化

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_2; //USART2 TX

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用输出

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStructure); //

//串口2的管脚初始化

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_3; //USART2 RX

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING; //复用浮空输入

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitStructure); //

}

USART\_DeInit(USARTx);

//初始化参数设置

USART\_InitStructure.USART\_BaudRate = baud; //波特率9600

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b; //字长8位

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1; //1位停止字节

USART\_InitStructure.USART\_Parity = USART\_Parity\_No; //无奇偶校验

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;//无流控制

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;//打开Rx接收和Tx发送功能

USART\_Init(USARTx, &USART\_InitStructure); //初始化

//TXE发送中断，TC传输完成中断，RXNE接收中断，PE奇偶错误中断，可以是多个。

USART\_ITConfig(USARTx, USART\_IT\_RXNE, ENABLE);

//USART\_ITConfig(USARTx, USART\_IT\_TXE, ENABLE);

USART\_GetITStatus(USARTx, USART\_IT\_TXE);

USART\_Cmd(USARTx, ENABLE);

}

void USARTx\_disable(USART\_TypeDef\* USARTx)

{

USART\_ITConfig(USARTx, USART\_IT\_RXNE, DISABLE);

USART\_Cmd(USARTx, DISABLE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

整形数据转字符串函数

char \*itoa(int value, char \*string, int radix)

radix=10 标示是10进制 非十进制，转换结果为0;

例：d=-379;

执行 itoa(d, buf, 10); 后

buf="-379"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char \*itoa(int value, char \*string, int radix)

{

int i, d;

int flag = 0;

char \*ptr = string;

/\* This implementation only works for decimal numbers. \*/

if (radix != 10)

{

\*ptr = 0;

return string;

}

if (!value)

{

\*ptr++ = 0x30;

\*ptr = 0;

return string;

}

/\* if this is a negative value insert the minus sign. \*/

if (value < 0)

{

\*ptr++ = '-';

/\* Make the value positive. \*/

value \*= -1;

}

for (i = 10000; i > 0; i /= 10)

{

d = value / i;

if (d || flag)

{

\*ptr++ = (char)(d + 0x30);

value -= (d \* i);

flag = 1;

}

}

/\* Null terminate the string. \*/

\*ptr = 0;

return string;

} /\* NCL\_Itoa \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 名 称：void USART\_OUT(USART\_TypeDef\* USARTx, uint8\_t \*Data,...)

\* 功 能：格式化串口输出函数

\* 入口参数：USARTx: 指定串口

Data： 发送数组

...: 不定参数

\* 出口参数：无

\* 说 明：格式化串口输出函数

"\r" 回车符 USART\_OUT(USART1, "abcdefg\r")

"\n" 换行符 USART\_OUT(USART1, "abcdefg\r\n")

"%s" 字符串 USART\_OUT(USART1, "字符串是：%s","abcdefg")

"%d" 十进制 USART\_OUT(USART1, "a=%d",10)

\* 调用方法：无

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void USART\_OUT(USART\_TypeDef\* USARTx, uint8\_t \*Data,...)

{

const char \*s;

int d;

char buf[16];

va\_list ap;

va\_start(ap, Data);

while(\*Data!=0)

{

//判断是否到达字符串结束符

if(\*Data==0x5c)

{ //'\'

switch (\*++Data)

{

case 'r': //回车符

USART\_SendData(USARTx, 0x0d);

Data++;

break;

case 'n': //换行符

USART\_SendData(USARTx, 0x0a);

Data++;

break;

default:

Data++;

break;

}

}

else if(\*Data=='%')

{ //

switch (\*++Data)

{

case 's': //字符串

s = va\_arg(ap, const char \*);

for ( ; \*s; s++)

{

USART\_SendData(USARTx,\*s);

while(USART\_GetFlagStatus(USARTx, USART\_FLAG\_TC)==RESET);

}

Data++;

break;

case 'd': //十进制

d = va\_arg(ap, int);

itoa(d, buf, 10);

for (s = buf; \*s; s++)

{

USART\_SendData(USARTx,\*s);

while(USART\_GetFlagStatus(USARTx, USART\_FLAG\_TC)==RESET);

}

Data++;

break;

default:

Data++;

break;

}

}

else USART\_SendData(USARTx, \*Data++);

while(USART\_GetFlagStatus(USARTx, USART\_FLAG\_TC)==RESET);

}

}

void USART\_transmitBytes(USART\_TypeDef\* USARTx, uint8\_t\* bytes, int32\_t len)

{

int32\_t i = 0;

for(i=0; i<len; i++)

{

USART\_SendData(USARTx,bytes[i]);

while(USART\_GetFlagStatus(USARTx, USART\_FLAG\_TC)==RESET);

}

}

/\*/////////////////////////////////////////////////////////////////

//加入以下代码,支持printf函数,而不需要选择use MicroLIB

#if 1

#pragma import(\_\_use\_no\_semihosting)

//标准库需要的支持函数

struct \_\_FILE

{

int handle;

// Whatever you require here. If the only file you are using is

// standard output using printf() for debugging, no file handling

// is required.

};

// FILE is typedef’ d in stdio.h.

FILE \_\_stdout;

//定义\_sys\_exit()以避免使用半主机模式

\_sys\_exit(int x)

{

x = x;

}

//重定义fputc函数

int fputc(int ch, FILE \*f)

{

if('\n' == ch)

{

putchar('\r');

}

while((USART1->SR&0X40)==0);//循环发送,直到发送完毕

USART1->DR = (u8) ch;

return ch;

}

#endif

//end

\*/