

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR



论文题目 多功能模块化移动机器人的控制系统设计

学生姓名 崔运凯

学生学号 5090209365

指导教师 费燕琼副教授

专业 机械电子

学院(系) 机械与动力工程学院

Submitted in total fulfilment of the requirements for the degree of
Bachelor
in Mechanical Engineering

Design of a Multifunctional Mobile Modular Robot System

YUNKAI CUI

Supervisor
Prof. FEIYAN QIONG

DEPARTMENT OF MECHANICAL ENGINEERING AND AUTOMATION, SCHOOL OF
MECHANICAL ENGINEERING
SHANGHAI JIAO TONG UNIVERSITY
SHANGHAI, P.R.CHINA

Jun. 4th, 2010

上海交通大学

毕业设计（论文）学术诚信声明

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：_____

日期：_____年____月____日

上海交通大学

毕业设计（论文）版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保 密 ，在 ____ 年解密后适用本授权书。

本学位论文属于

不保密 。

(请在以上方框内打“√”)

作者签名：_____

指导教师签名：_____

日期：____年____月____日

日期：____年____月____日

多功能模块化移动机器人的控制系统设计

摘要

本文从模块化机器人与移动机器人的发展现状出发，对于新式的机器人的需求做出了合理的分析，并确定了一种将两种机器人模式进行整合的方案。其兼具有移动机器人的移动能力，同时又可根据用户对于机器人的需求而组合不同的模块以实现需求的功能的灵活性。这样一款智能多模块移动机器人符合现代工业界与特殊应用领域对于机器人的需求。

在确定这款多功能模块化移动机器人的结构后，本文对于控制这一机器人的系统的设计进行了详细的分析与介绍。作为一个完整的系统，其应该包括三个层面上的设计。第一为外在的与环境交互的机器人机械结构，第二为为控制外部结构提供可能的机器人系统电路硬件，第三为对机器人与外界交互制定规则的控制外部结构的软件。本文从每个部分所需要的功能出发，对每一个部分的内容都进行了详细的介绍。

机械结构部分，提出了基于以前机器人模块设计方案的诸多改进，其中包括对于通用模块的重设计，对于传感器支架重设计，对于标准通讯接口的重设计，以及对新增加的用于辅助定位的轮式编码器的设计。在设计的说明中，本文给出了最后实现的效果图。

电路硬件部分，进行了多块 PCB 印刷板的设计。其中硬件电路主要分为两个部分，第一部分为每个模块都要使用的基本功能控制电路。其包括单片机，电源管理单元，模块间通讯接口。其他未使用的接口全部引出，以供外接模块使用。第二部分为外设辅助电路。其中包括实现各个具体功能的电路，如

用于定位的 IMU 模块，用于测位移和轮速的编码器模块，用于驱动直流电机的电机驱动模块，用于驱动传感器的传感器接口模块和用于实际模块间通讯实现的模块间通讯接口模块。

控制软件部分，对控制系统的基本软件单元进行了合理的解释与介绍。这些基本单元包括电机驱动程序，车体定位算法，未知环境避障算法和地图绘制与最优路径生成算法。本论文提出了改进的虚拟势场算法，并在动力学仿真环境 GAZEBO 中实现了模块化移动机器人的运动仿真。

关键词：模块化机器人，移动机器人，模块化硬件电路，定位方案，虚拟势场，GAZEBO

Design of a Multifunctional Mobile Modular Robot System

ABSTRACT

From the state of the art of development of mobile and modular robots, this paper talked about the potential applications and requirements of the new type of robots, and attempt to combine these two concepts together to design a new kind of robot. A robot with moving ability and can be customized by combining different modules will be more attractive and useful.

After figuring out the structure of this new kind of robot, this paper talked a little bit more about the design process. An independent system should contain three crucial parts. The first of all is the mechanism of the robot which will define how the robot interact with environment. The second one is the circuit that can help the software to control the robot. The last is the software, which is the soul of a system and can be regarded as a set of rules that define the behaviour of the robot. In this paper, it will provide enough details about these three parts.

For mechanism, the paper shows a lot of re-design of the old modular robot model. For instance, it contains a redesign of module body, a re-design of the sensor stand, a re-design of communication port. There are also some newly designed parts, like an encoder wheel. After each explanation, there will be a rendered image about the real look of those parts.

For circuit, the paper provided many PCB design documents, which illustrate the final appearance of a PCB. In this part, there will be two kind of circuit being introduced. The first kind of circuit is a basic one, which contains MCU and communication unit, the other part of circuit are peripherals. The peripherals will help specific module to achieve specific functions.

For control software. Most of the software in this paper are written in C, a lot of details in math will be provided. In this part, the paper also propose a new kind of path planing method which is modified virtual potential field method.

In the last part of this paper, it shows the simulation of the robot moving in a test environment. The simulation software we used is GAZEBO. The simulation succeed and proofed that this kind of algorithm is applicable and valid.

KEY WORDS: modular robot, Control, localization, potential field, GAZEBO

目 录

第一章 绪论	1
1.1 模块化机器人的研究背景	1
1.2 避障式移动机器人及路径规划研究现状	2
1.3 课题的研究内容和意义	3
第二章 模块化结构的设计与改良	6
2.1 先前设计方案的回顾	6
2.1.1 传感器模块结构	7
2.1.2 控制模块结构	7
2.1.3 移动模块结构	9
2.1.4 标准化机电接口	9
2.2 通用可扩展模块的补充设计	9
2.3 模块间通用接口的设计	11
2.4 超声波传感器支架的设计	12
2.5 底盘编码轮的设计	13
2.6 本章小结	14
第三章 通用控制电路设计	16
3.1 通用模块控制电路	17
3.1.1 最小系统	17

3.1.2 功能接口板	19
3.2 功能型外设及接口设计	22
3.2.1 IMU 模块	23
3.2.2 驱动模块	25
3.2.3 编码器模块	25
3.2.4 传感器模块	28
3.2.5 模块间通讯接口	29
3.3 本章小结	30
第四章 控制与智能避障算法	33
4.1 底盘驱动算法	33
4.2 车体定位方案	37
4.2.1 拖地轮及方位角传感器定位	37
4.2.2 IMU 定位	40
4.3 未知环境避障算法	42
4.4 地图生成和最优路径生成	44
4.5 本章小结	45
第五章 避障算法的仿真	46
5.1 仿真软件环境介绍	46
5.2 移动机器人模型的建立	46
5.3 机器人所处的仿真世界的建立	52
5.4 机器人的扩展组件撰写	52
5.5 仿真过程与算法验证	57
5.6 本章小结	58

全文总结	59
附录 A 仿真中所使用的机器人所处世界的代码	60
参考文献	88
致谢	91

插图索引

2-1 原设计方案模型	7
2-2 原设计方案传感器支架	8
2-3 原设计方案控制模块	8
2-4 原设计方案移动模块	9
2-5 原设计方案标准化接口	10
2-6 新设计的通用模块三维立体图	11
2-7 新设计的通用模块的连接机构	12
2-8 新设计的 CAN 总线接口	12
2-9 新设计的传感器支架三维立体图	13
2-10 新设计的底盘编码轮三维立体图与工程图	15
3-1 最小系统实物图	17
3-2 最小系统电路图	18
3-3 功能接口板电路图	20
3-4 电源管理电路原理图	21
3-5 CAN 电路原理图	21
3-6 功能接口板的设计 PCB 样图	22
3-7 功能接口板的虚拟实物图	22
3-8 功能接口板的虚拟实物图 3D	23
3-9 IMU 电路原理图	24

3-10 IMU 的设计 PCB 样图	25
3-11 IMU 的虚拟实物图	25
3-12 驱动电路电路图	26
3-13 驱动电路的设计 PCB 样图	27
3-14 编码器用外部计数器接线图	27
3-15 超声波传感器驱动板原理图	28
3-16 超声波传感器	29
3-17 模块间通讯接口板电路原理图	31
3-18 模块间通讯接口板设计 PCB 样图	32
3-19 模块间通讯接口板虚拟实物图	32
4-1 底盘上坐标系的定义	38
4-2 机器人运动示意	39
4-3 虚拟势能避障算法示意图	44
5-1 GAZEBO 系统结构图	47
5-2 仿真所建立的障碍物世界	52
5-3 仿真进行示意图	58

第一章 绪论

1.1 模块化机器人的研究背景

随着控制理论，传感器，计算机科学和人工智能等技术的发展，机器人的研究越来越受到关注。从上世纪 90 年代至今，机器人技术得到了空前的发展，由单一化，大型化和功能固定化转向小型化，廉价化和模块化^[1]。与此同时，机器人技术正在被应用在越来越多的领域，从工业生产，到未知环境探测，从航天工程，到服务餐饮。而在绝大多数的应用中，或多或少的对机器人的移动性能有要求，如生产线中的搬运机器人，应用在航天探测中的火星车月球车，应用在军事领域的拆弹机器人等等。所以机器人的移动性一直是机器人研究领域的热点。如 W. Grey Walter 等人在 1948 年设计并演示的移动机器人 Elmer 和 Elsie^[2]。这两个机器人可以说是自主式移动机器人的“祖先”了。时至今日，只具备简单移动能力的机器人显然无法满足现代社会对于机器人功能的需求。因此移动机器人被加装各种各样的机构而成为更为复杂的功能型机器人。

随着人们对机器人功能需求的不断增加，机器人越做越复杂。但是功能强大的机器人却未必是完成单一工作的最优选择，因为对于机器人在某一阶段所从事的工作来说，大多数其他功能并没有被使用到。在这样的背景下，模块化正是一个很好的解决方案。模块化是根据功能将机器人进行拆分，并通过用户对于具体功能的需求进行组合的过程。用户不再为自己不需要的功能而付出任何金钱上的代价，正相反，每一个模块化机器人都像是面向用户定制的产品，而这一产品对于用户永远是最优的。同时，一个具有移动能力的模块化机器人相较于其他机器人有更多的应用价值和能节省更多的成本。

模块化机器人的研究，始于上个世纪 80 年代中期^[3]。而对于模块化机器

人，也有了越来越准确的定义：它由多个功能模块及标准接口装配而成，各种功能模块可批量生产、独立维修、独立扩展，快速组装成不同性能的移动机器人；模块化移动机器人之间可局部通信、相互合作，完成全局任务。它有下面显著特征：更广泛的任务领域、更高的效率、改良的系统性能、容错性、鲁棒性、更低的经济成本、容易开发、分布式的感知与作用以及内在的并行性等。

目前对于模块化机器人的研究主要集中在两个方向上，一为静态可重构机器人，另一个为自重构机器人。静态可重构机器人是指，可以借助外力进行重构的，在工作状态是其结构为静态结构的机器人^[4]。简单地说，是有人为选择模块并可利用这些模块的工作的机器人。而自重构机器人这是可以动态的改变自身结构的机器人。

在静态可重构机器人领域，已有很多人做过相应的研究。如 Benhabib-B,DaiM 等人^[5] 设计了一个基于遥驱动技术的模块机器人单元，驱动方式类似于传统的工业机器人。Paredis C J J 等人^[6] 设计了 RMMS 模块化机器人系统。它有一个模块库，可根据需要来搭建模块，实现不同的功能。Fujita^[7] 在 Sony 公司的 OPEN-R 标准之下，开发了一套可重构模块化机器人系统。可以通过众多模块组成不同的机械结构。

在自重构机器人的领域，也有很多人进行了相关研究。如 M Yim^[8] 设计的 Polypod 和 PolyBot 系统。还有 Murata^[8] 的机器人系统 Fracta，是通过仿生学的细胞概念而提出的一种三维的自重构机器人系统，等等。

1.2 避障式移动机器人及路径规划研究现状

移动机器人的研究最早可以追溯到上个世纪 60 年代^[9]。斯坦福研究院的 Nils Nilssen 和 Charles Rosen 等人在那一时期设计出了一款叫做 shakey 的自动避障移动机器人^[10]。随着计算机的应用和传感技术的发展，有越来越多的公司和研究机构参与到移动机器人的研究中来，从而大大促进了移动机器人技术

的发展。

对于移动化智能机器人的研究，两大关键研究领域是定位和环境探知（传感器）。现在的机器人导航和定位有多种方法，如基于环境信息的地图模型匹配导航，基于各种导航信号的陆标导航，视觉导航和触觉导航等^[9]。环境地图模型匹配导航机器人可以通过多种传感手段对于地图与采集信息信息进行匹配，从而得到自身所处位置信息，最后通过规划算法来行进至目的地。这是一套基于已知环境的移动方案。陆标导航则是在机器人活动区域设置可以探知的信号源，通过对信号源的感知来获得自身相对位置，并在路标的指引下，向终点前进的方法。视觉和触觉导航，这是通过视觉和触觉传感器，对行进环境进行识别，并作出行进决策的导航手段。这是一种动态的可适用于未知环境的机器人移动方案。移动机器人传感技术主要是对机器人自身内部的位置和方向信息以及外部环境信息的检测和处理，采用的传感器分为内部传感器和外部传感器，其中内部传感器有：编码器，线加速度计，陀螺仪，磁罗盘，激光全局定位传感器，激光雷达等^[9]。

而基于两大模块，避障算法的研究也有了长足的发展。避障算法经历了人为设计环境避障，已知环境避障，半已知环境避障和完全未知环境避障等几个发展阶段。Ting-Kai Wang 等人^[11]提出了基于障碍物边界和模糊逻辑的未知环境避障算法。Torvald Ersson 等人^[12]提出了基于网络化简的针对于短程传感器的未知环境探测算法。Shuichi Utsugi^[13]提出了一种基于视觉的连续关注点捕捉技术的智能避障算法，等等。

1.3 课题的研究内容和意义

移动机器人的功能化带来了其结构过于复杂化的问题。而对于大多数应用领域来说，如不能使用机器人的全部功能，将造成巨大的资源浪费。通过之前的讨论可以得知，模块化是解决功能复杂化的最好的方法。人们对于机器人的

选用，也会由机器人能提供什么，到希望机器人能做什么的方向转变。只选用自己有用的模块而使机器人的使用和维护成本达到最小。与此同时，模块化机器人的实现，也为系统的升级减小了开支。设想人们可以对机器人进行简单的升级，就像升级软件一样，这将是具有革命意义的。因此，我对于复杂机器人的模块化研究，将主要集中在如何将功能进行拆分和如何对众多模块进行控制上。

正如先前所讨论的，定位对于移动机器人控制来说是一个非常重要的环节。而如何对移动的机器人进行定位，这是我的设计所需要面临的另一个问题。目前对于定位的研究，非常依赖于 GPS 模块，因为其相对简单，信息容易获取。但存在的缺点是，只能在开阔的户外环境中所使用。而对于需要在楼内或障碍物内移动的机器人来说，这一方法是极其不可靠的。所以我需要在前人的设计基础上进行一定的改进，引入一种可以获得坐标信息的机械结构到模块化底盘上。通过对虚拟坐标的计算来获得车体的位置信息，从而完成定位。

关于避障算法的研究，已经进行了多年，多种方案都已经被提出。但这些方案往往根据需要而去选择传感器，却不会根据传感器限制，去使用有限的感知条件去适应和感知未知环境。机器人只有对于不同场合选用不同的传感手段，这样才能做到最优化，避免了不必要的资源浪费。但现在的智能移动机器人为了兼容尽可能多的环境，而装备了多种传感器，应用了多种传感手段，这对于日常应用，是一种不必要的浪费。模块化的拆分思想这有助于建立解决这一问题的思路。根据这一思想，我对于传感器和避障算法的研究将主要集中在如何拆分多种传感手段，如何使传感信息和主控制系统沟通，如何利用这一信息而完成避障和驶向目标点。而这套系统模型的建立，传输机制和协议的制定，将会提供一整套嵌入式模块移动避障机器人的解决方案。

综上所述，我的主要设计任务是设计多功能模块化移动机器人的控制系统，根据单个机器人的功能特征及多个模块间的协作行为，确定该控制系统的

结构；这将包括硬件及软件层面两个部分。硬件层面，实现多机互通接口的设计，模块核心控制单元的设计。并成功实现对底盘模块的驱动，和传感器模块对外部环境的感知；软件层面，实现多模块事件响应模型的设计，标准化传输协议的制定，底盘实际驾驶控制、定位和追踪，传感器模块信号处理分析及核心控制单元功能的实现。开发仿真平台，实现单个多功能模块化机器人在未知环境下的避障移动，应用或改善现有路径规划算法。同时可以简单模拟多模块通讯协议机制。

第二章 模块化结构的设计与改良

对于控制系统的工作，往往也会涉及到机械结构的配套设计。一个控制系统对于机器人控制的实现，是要通过执行器来实现的，因此控制器往往会对执行器的外观有一定的要求；同时控制器决策的确定，是需要传感器辅助来实现的，因此传感器对于所在的机器人本体也会有一定的要求。综上所述，机械结构的设计是控制系统设计的前提。在本章中，会给出和控制系统相关的机械结构的设计细节，以期起到对于后面的控制系统设计的辅助说明。

2.1 先前设计方案的回顾

<<<< HEAD 本文所设计的控制系统是以所在实验室先前设计的太阳能驱动模块化机器人作为控制对象原形来进行设计的。所以在对本文所涉及的对于机械结构的改进与重设计之前，对前人的设计方案进行回顾是有必要的。原模块化机器人拼接成的太阳能机器人的 3D 模型和实物图如图2-1 所示。

===== 本文所设计的控制系统是以所在实验室先前设计的太阳能驱动模块化机器人作为控制对象原形来进行设计的。原模块化机器人拼接成的太阳能机器人的 3D 模型和实物图如图2-1 所示。

>>>> 344532f7f0d9415fa34ed575a4c3b19c275e0e14 原机器人的坐标系规定如下：

原点 O 取为移动模块两驱动轮中心连线的中点，Z 轴垂直机器人底面向上，Y 轴沿机器人前进方向，X 轴由右手定则确定。太阳能驱动模块化机器人结构与坐标方向示意如图3.19(a) 所示。

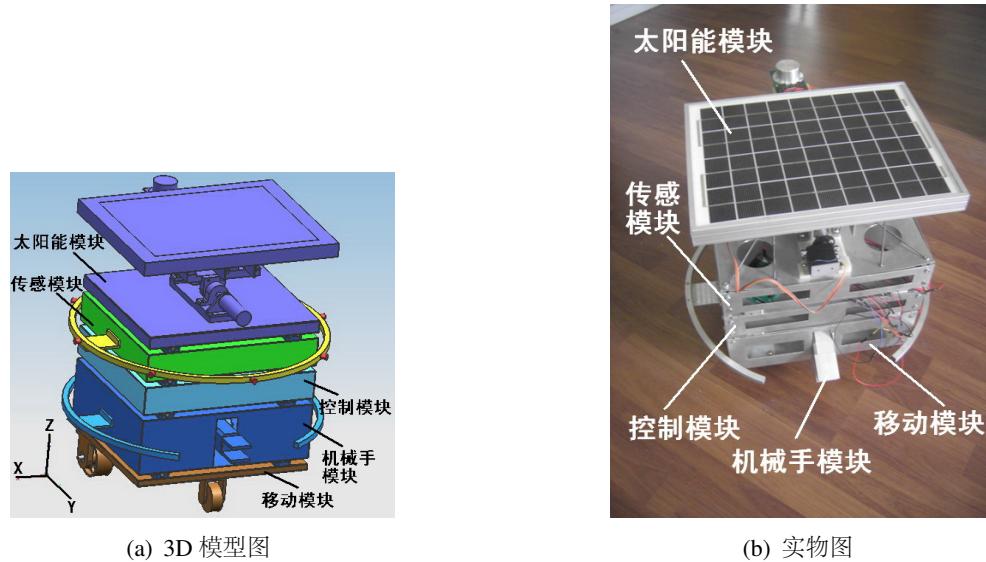


图 2-1 原设计方案模型

Fig 2-1 The Model of the Original Design

2.1.1 传感器模块结构

传感模块主体为立方壳体，模块上、下表面分别安装上、下接口。两侧分别固定一个支架，再在支架上固定一个圆环，在圆环的前、后、左、右及左前、右前、左后、右后的八等分圆处分别安装一个超声波传感器。主要用于实现对八个方位的障碍物探测与测距功能。如图2-2 所示。

2.1.2 控制模块结构

控制模块主体也为立方壳体，模块上、下表面也分别安装上、下接口。其内安装主控制系统电路板和双轴陀螺仪、电子罗盘等，实现机器人自身的倾角和方位测量等功能，接收传感模块输入的障碍物探测信息，并输出对移动模块和机械手模块的控制信号。如图2-3 所示。

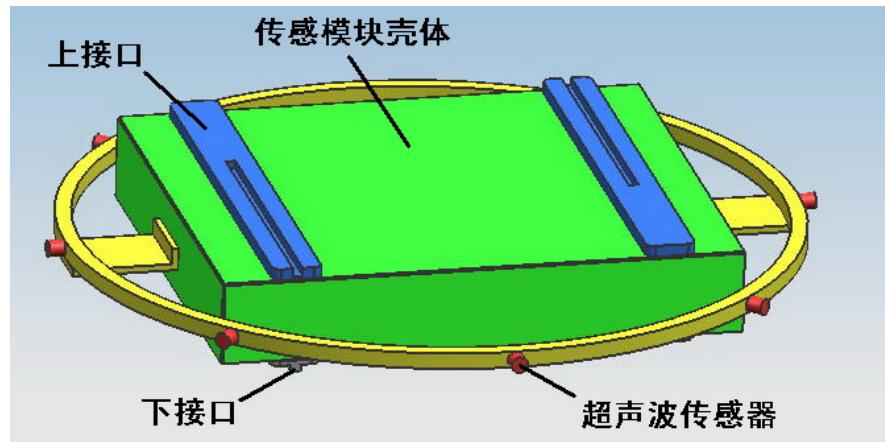


图 2-2 原设计方案传感器支架
Fig 2-2 The Sensor Stand of Original Design

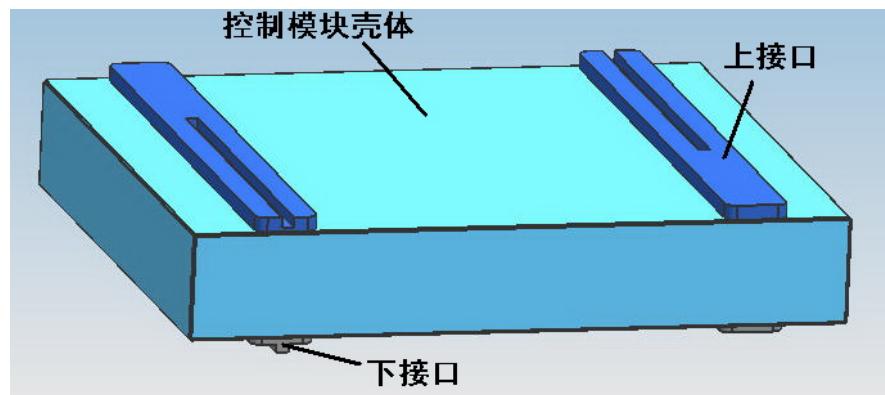


图 2-3 原设计方案控制模块
Fig 2-3 The Control Module of Original Design

2.1.3 移动模块结构

移动模块主体为较薄的立方壳体，模块上表面安装上接口，下表面安装两个驱动轮和一个从动脚轮。两个驱动轮分别由一个直流电机带动，利用两驱动轮差速运动实现转向运动；从动脚轮具有两个自由度，在摩擦作用下可以自动适应转动方向；磁珠装于轮轴上，通过安装于底盘上的霍尔传感器输出转速信息给控制模块。如图2-4 所示。

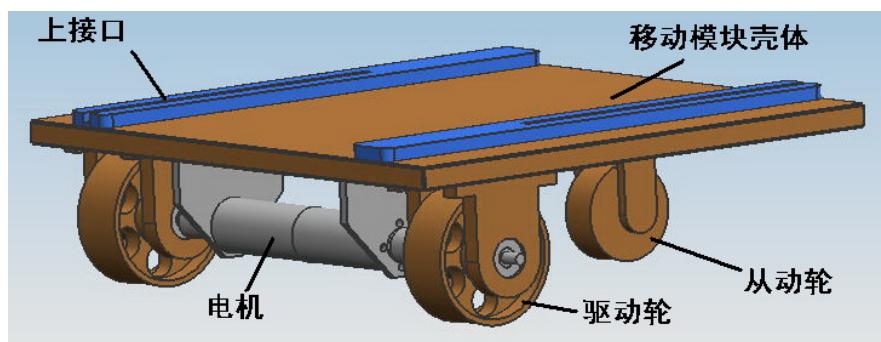


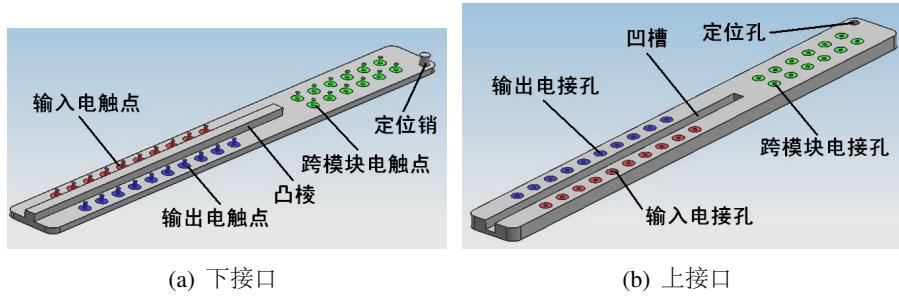
图 2-4 原设计方案移动模块
Fig 2-4 The Mobile Module of Original Design

2.1.4 标准化机电接口

原方案设计的接口如图2-5 所示。对接时，下接口面上的凸棱与上接口面上的凹槽配合，可限制对接的模块在 X 方向上的滑动自由度，再加上定位销与定位孔的配合，可限制模块沿着凹槽在 Y 方向上的滑动自由度，而上接口与下接口面配合，可限制垂直模块底面的 Z 方向上的平移自由度。如此，可方便地进行模块的机械对接。

2.2 通用可扩展模块的补充设计

在新的机器人系统中，将不再采用以前机器人的控制结构。在上面的介绍中，可以获知以前机器人拥有单一控制模块，所有其他模块都由这单一的模块



(a) 下接口

(b) 上接口

图 2-5 原设计方案标准化接口
Fig 2-5 The Communication Port Original Design

进行控制。这就要求控制系统在新加入模块时要对总控制模块再编程。而且对于这种设计，为了实现其他模块的功能，不得不将大量的控制接口引出，集成在机械结构上，如上一节标准化机电接口设计所述。这种做法存在众多缺点。首先，接出来的众多接口提高了模块系统的复杂性，这些结构必须以有一定的形式存在于整合机器人个体的各个模块上，但是提供接口的上层模块却未必会用到接口功能，这就产生了设计资源与加工资源的浪费；其次这种设计缺少灵活性，在新加入功能模块或是主控芯片变动时，需要对接口进行再设计与加工，不利于机器人模块的升级使用；第三机械触点式接口易于磨损，长期使用的可靠性极差。综上所述，本设计提供了一种新的思路去解决上面所述的种种问题。

将对于模块的控制电路集成在各个模块上，即抛弃主控模块的概念，而在各个模块上设置控制电路，是每个模块可以脱离主体进行单独工作。这样只需要一种协调各模块的机制就可以了。现存的多种基于嵌入式系统的通讯机制可以满足这一要求。不过在众多系统中 CAN 总线系统及协议，是相对最好的解决方案，也是本设计所采用的方案。其有如下优点，第一，CAN 总线接线简单，总线中只有 CANH 和 CANL 两条线；第二，协议成熟，这一协议的 2.0 版本是由博世公司于 1996 年提出的。至今经过了 10 余年的使用足以证明这个协议的可靠性；三，可靠性高，因为依赖的连线比较少，所以容错率会比 SCI，

SPI, I^2C 等其他协议要高；四，可适用于大规模集群网络，就是指可以使众多模块在系统中协同工作。

根据这一思路，对系统进行了设计。关于电路硬件部分的详细说明请见下一章相应部分内容。在本节和下一节将给出根据此思想所设计的通用机械模块和模块间通讯接口的机械设计。

新的模块设计三维立体图如图2-6 所示，用于连接上下两个模块的槽型和凸起机构如图2-7 所示。其中模块结构包括上下两个交叉的加强筋和一个电路板固定区，以及可以扩展的侧方空间。

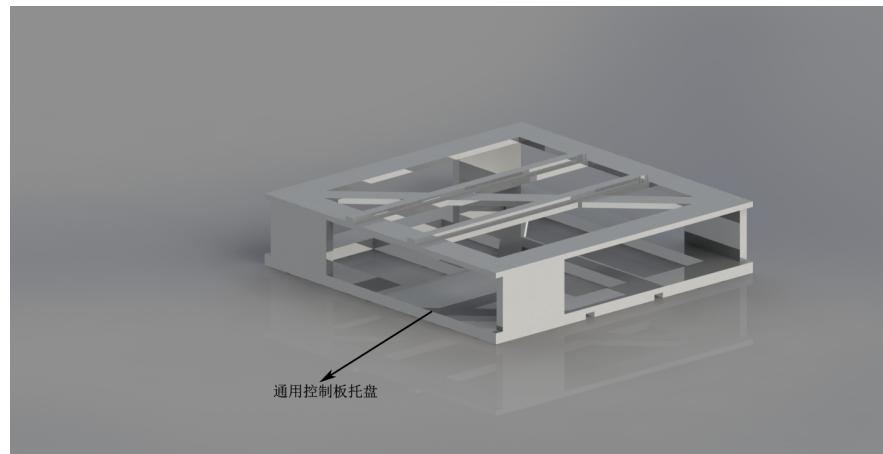


图 2-6 新设计的通用模块三维立体图
Fig 2-6 The General Module of New Design

2.3 模块间通用接口的设计

模块间通用接口既是固定在通用模块体上的，实现模块间通讯的接口电路板的机械固定装置。正如前文说明的那样，为了提高稳定性和简化模块结构，本设计决定使用 CAN 总线作为传输途径。但是又如前文分析的那样，如果采用接触式方案，将极易磨损而使传输变得不可靠。所以这里采用了光学手段，将总线中的高低电平变成红外 LED 的亮灭，而另一模块则通过一个光敏三极



(a) 处于模块上端面的凸起

(b) 处于模块下端面的凹槽

图 2-7 新设计的通用模块的连接机构

Fig 2-7 The Connection Structure of the New Design



(a) 处于凸起上的接口

(b) 处于凹槽上的接口

图 2-8 新设计的 CAN 总线接口

Fig 2-8 The Communication Port of the New Design

管接收传过来的光信号，并变回电信号传回另一模块的总线中去。因为光敏三极管是一个极其敏感的元件，为了避免不同光源之间的相互干扰，需要使对应的红外 LED 和光敏三极管处于同一个密封环境，这就需要在设计机械结构上得以实现。关于转换电路的内容会在第三章相应部分进行详细介绍。而接口的机械接口如图2-8所示，其位于槽和凸起的末端。

2.4 超声波传感器支架的设计

因为定位的原因，本设计对传感器支架进行了重新设计。虽然圆形的传感器排布可以实现对环境的较好的检测，但是对于计算车体相对于障碍物的相对位置却变得非常复杂。为了解决这一问题，本设计重新对传感器的位置进行了安排。先前的设计文献中只把传感器作为圆柱处理，这显然是不现实的，在这

里本设计对传感器尺寸进行了仔细测量，并按实际尺寸画出了传感器对于传感器支架的安装位置。其设计图如图所2-10示。图中为传感器模块一侧传感器支架的样子，在模块的四个边上各有一个支架，每个支架上有三个超声波传感器。



图 2-9 新设计的传感器支架三维立体图
Fig 2-9 The General Module of Sensor Stand

2.5 底盘编码轮的设计

最直观的的对于二维坐标的测量莫过于直接测量车体在二维坐标中的的位移。在传统的方法中，我们一般会在电机驱动轴上加装编码器，这样通过测量轮子转过的圈数，在已知轮直径的基础上可以算出每一个轮子走过的路程，再通过车体尺寸等其他参数，可以计算出车体的轨迹，从而得到车体的定位信息。但实际上，这种方法存在着一些问题。首先驱动轮是非常容易打滑的，所以将编码器装到驱动轮上，经过积分后，计算值与实际值之间的误差会发散，得不到准确的坐标值；其次，编码器放在轮子处在好多时候都会增加对于车体轨迹的计算的复杂性。综上所述，如果想使用这种直接的方法对车体进行定位，必须要克服上述诸多的问题。

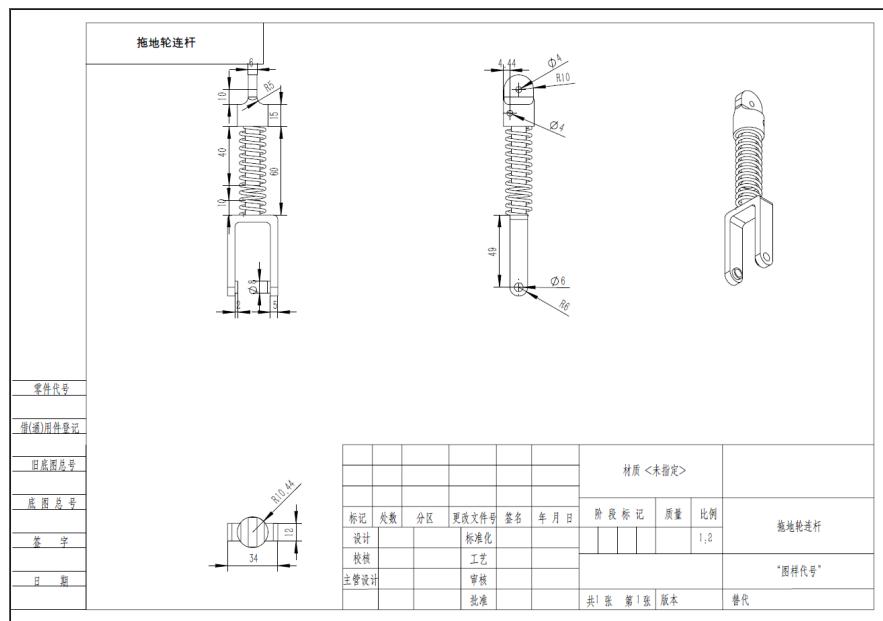
在本设计中，将编码器独立出来，放置到可以自由安放在底盘某一部位上的从动轮机构上，以实现编码器与主动轮的脱离。因为编码轮可以放置在机器人底盘的任意位置上，所以可以很容易解决计算复杂性的问题。接下来需要解决的就是打滑问题。对于主动轮，打滑的原因是因为电机输出扭矩与地面提供摩擦力不匹配造成的。而因为从动轮没有扭矩输出，所以只需要使地面提供用于克服转轴处摩擦转矩的力就可以了。对摩擦力的要求相对较小。而为了提供足够的摩擦力，只需要施加适当的正压力就可以了。所以这一独立的轮式编码单元必须是弹性结构，可以适应高低起伏的地形，同时可以提供对地面足够的正压力。基于这一准则，本文设计了一个基于四连杆机构的弹性轮式独立编码系统，如图2-10所示。其中蓝色的轮子为万向轮，用来减少对于车体控制的影响。因为难以绘制，所以在这里以普通轮子表示。一段比较长的轮轴上固定光学编码器。

2.6 本章小结

本章由已有的机器人设计出发，分析了以前设计的特点，并对设计中的不妥提出了疑问。再根据这些疑问出发，着手改进原来的设计方案。在本章中，详细的介绍了对于通用模块，传感器支架，模块连接机构，模块间传输单元机构和独立轮式编码单元的新的设计方案。通过新旧方案的对比，可以较为容易的发现，新的设计方案克服了以前设计方案所存在的问题。



(a) 三维立体图



(b) 部分工程图

图 2-10 新设计的底盘编码轮三维立体图与工程图
Fig 2-10 The General Module of Independent Passive Wheel Encoder

第三章 通用控制电路设计

模块化机器人的各个模块都是可以独立工作的个体。从上一章的设计可知，每个模块都有一个核心板载系统来对本模块的基本功能进行控制。这一设计思路在其它模块化机器人上也有所体现。例如美国加州大学戴维斯分校的 Graham G. Ryland 等人^[14] 设计的模块化机器人系统，每个模块是包含四个自由度的可以独立控制并运动的个体。而个体之间通过通信和协议来完成作为组合的总体所实现的功能。在本设计中，因为每个模块都有不同的功能，进行拼装后的实体的机器人通过协同来实现全部功能。为了使设计的硬件能在最大程度上实行公用，本设计将机器人电路进行拆分，并对电路实现了模块化。

每个机器人模块都将实现作为一个模块的基本功能，如实现与其他模块的通信；单独的程序上传与下载、与在线调试；对于自身模块的电源管理；与外部其他系统的通讯；简单的片上调试按钮和指示用数码管和 LED。据此本方案设计了每个模块都要使用的通用模块控制电路。而对于不同模块的不同功能需求，本方案设计了大量的外设电路，每一个模块将会在功能性外设及接口设计部分进行讨论与说明。这样的好处是使每一个模块都拥有其工作所用的资源而不会有资源的浪费。例如，用于定位的 IMU 模块只有地盘模块需要使用。而上层实现其他功能的模块，如机械手并不需要使用，这里没有把其集成在通用板上，是极大的避免了资源的浪费。为了能够使用这些外设电路，本设计已经将片上所有资源系数引出。这样做同时还可供未来通用板的扩展。

以下各小节将对整个机器人系统的电路系统进行详细阐述。

3.1 通用模块控制电路

通用电路板本着简单，稳定，可以快速开发和方便部署的原则进行了设计。控制芯片选用了引脚众多，功能强大和文档完善的 Freescale 公司的 MC9S12XS128 单片机。本设计使用的是拥有 112 个引脚的版本。这款芯片的特点是低功耗、高集成、易于扩展，自带看门狗计数器、PWM 输出、增强型捕捉定时器^[15]。在本设计中，将通用控制电路板分为两个部分，一个部分是包含 BWM 程序烧写在线调试器，复位按键和单片机芯片的最小系统；另一部分是提供通用电路功能的接口板。这样设计而不是直接将单片机焊接在接口板上的原因是，一旦单片机烧毁，可以同过更换最小系统来使电路板恢复使用，而不用丢弃整个电路板基板上昂贵芯片。

下面的部分将对这两个模块进行详细的介绍。

3.1.1 最小系统

最小系统的电路图如图3–2 所示。最小系统实物图如图3–1 所示。

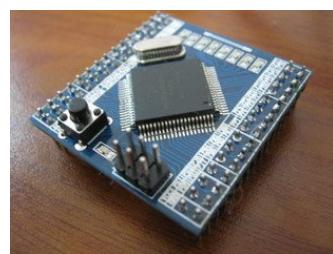


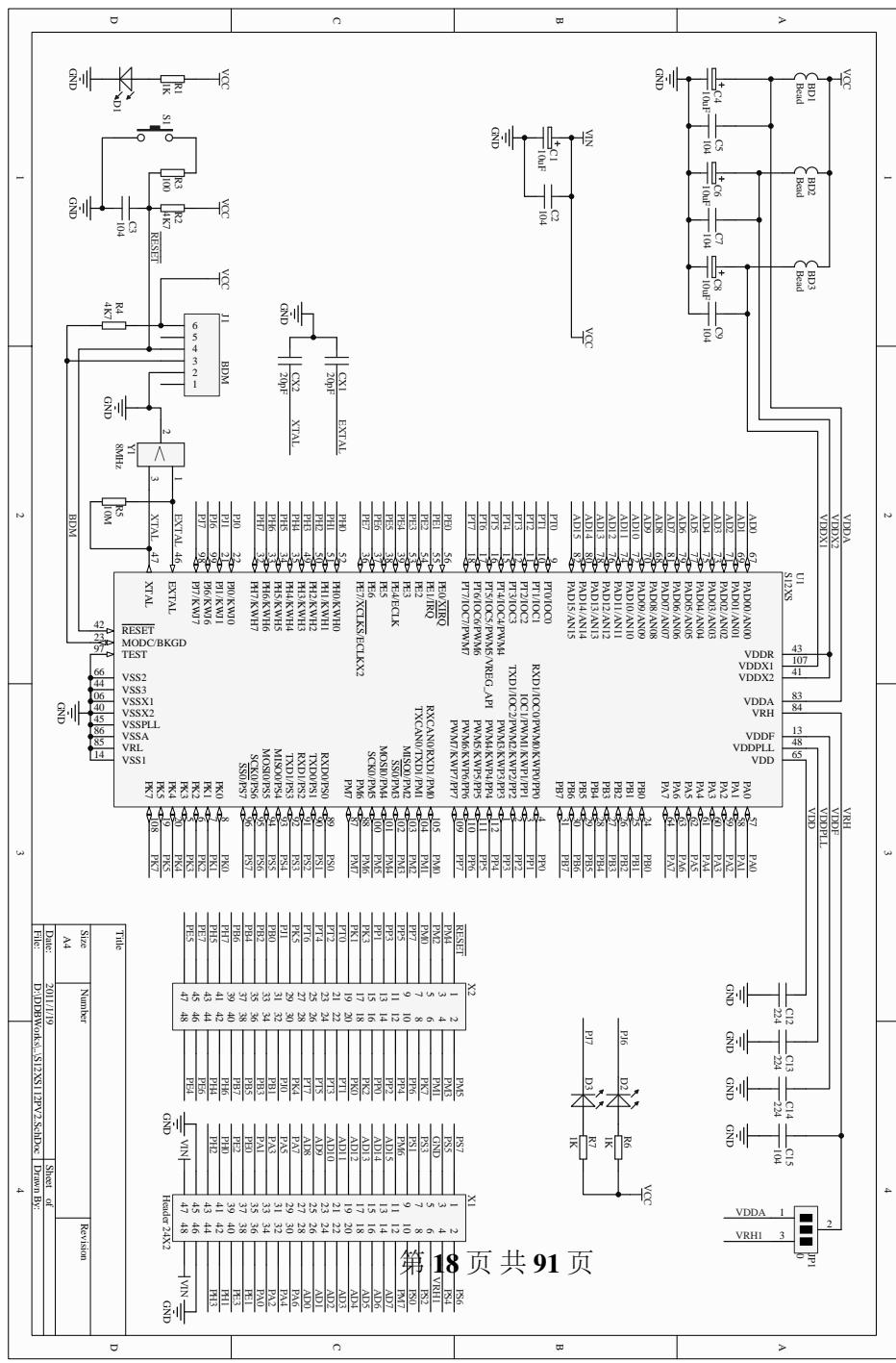
图 3–1 最小系统实物图
Fig 3–1 the Minimal Core System

这块最小系统板上有一个 16MHz 的晶振连接至芯片。芯片正常工作时的频率是由锁相环 (PLL) 超频到 40MHz 的。这块最小系统还有一个复位键，引入芯片 RESET 引脚，将检测电路的下降沿进行复位。同时最小系统上还提供



图 3-2 最小系统电路图

Fig 3-2 Schematic of the Minimal Core System



一个 BWM 程序上传和下载及在线调试模块。片上提供 128k 片上 RAM，可以用来储存芯片执行用程序。还有一个独立的电源管理模块，以提供稳定的 5v 电压给单片机工作。之所以称为最小系统，就是因为以上是这块芯片的全部功能元件。如原理图所示，这块最小系统的其他功能就是简单的将单片机所有引脚引出。

3.1.2 功能接口板

功能接口板是通用模块控制电路的核心部分。它具有承上启下的作用，得到最小系统引出的单片机全部空余引脚，并对其功能进行设计。其电路图如图3-3 所示。其共分为电源管理，CAN 总线，单片机接口，电压测试单元，串口通讯，数码管显示单元，LED 及按键阵列和引出接口几部分。

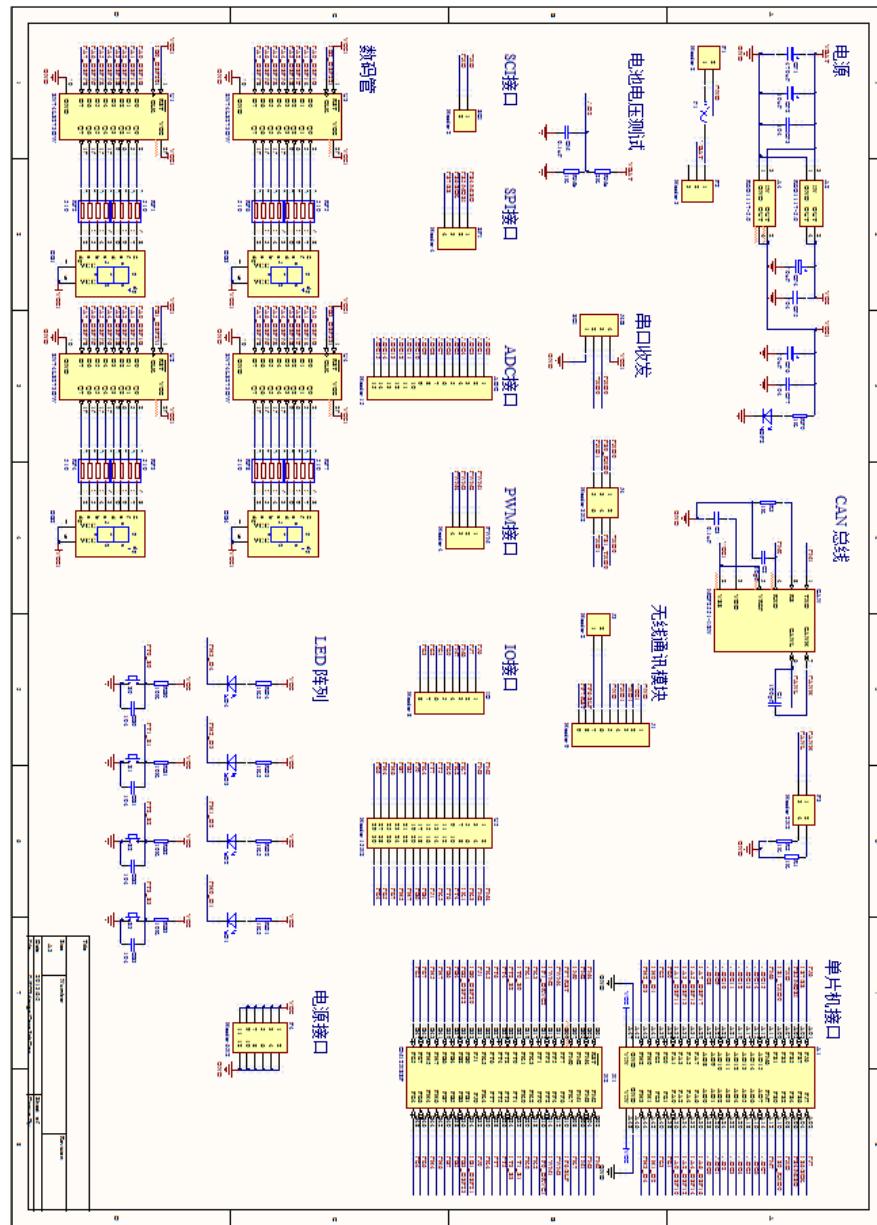
其中电源管理部分的电路如图3-4。使用两片 1117 输出两个 5v 电压，为了使电流相差较大模块之间不会相互干扰。其中一个给单片机及通讯模块供压，电流小，对电流稳定性要求高；而另一路主要给外设电路供压。

CAN 总线电路图如图所示3-5。用一块 Microchip 公司生产的 MCP2511 作为单片机内置 CAN 模块的 Transceiver，将 CAN 端口数据打包编码后传输到总线中去。前面已经介绍过了这款机器人将打破传统的接触式传输连线，而是使用光学方法进行数据传输，其中这里 CAN 总线的输出口，就是要连接到光学传输接口板上。关于光学传输接口板的具体信息将在下面的模块间通讯接口小节进行详细的介绍。

CAN 总线 2.0 协议^[16] 是由博世公司提出被提供支持的。因为其两线制的机制，和可以同时连接非常多个模块，并能组织各个节点模块共同工作的特点而大量应用在汽车领域。其传输速度快，稳定性强的特点也为其实现轻松移植到机器人系统中来提供了可行性。相对较少的接线同时也意味着极大的容错率，这就是本设计选择 CAN 总线为模块间传输工具的原因。这一设计的灵感来源



图 3-3 功能接口板电路图
Fig 3-3 Schematic of Base Board



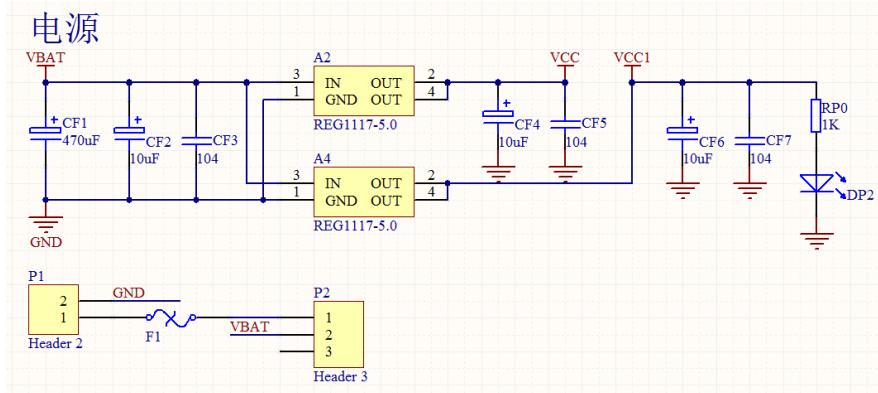


图 3-4 电源管理电路原理图
Fig 3-4 Schematic of Power Source Management Unit

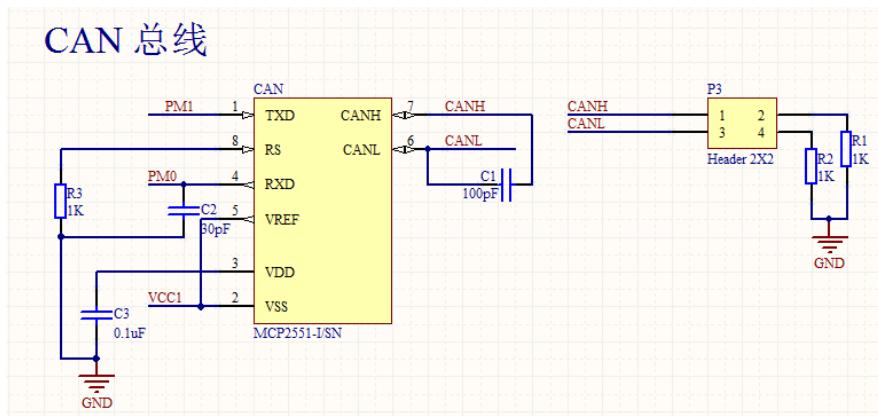
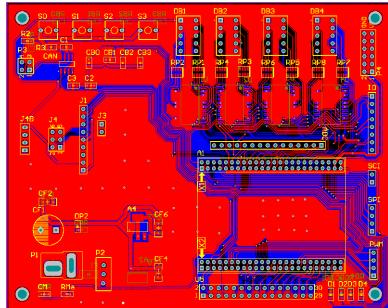
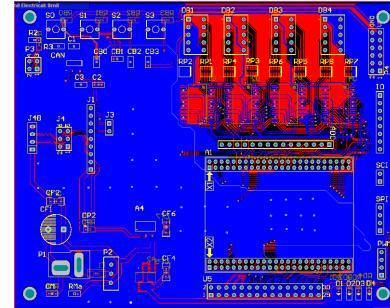


图 3-5 CAN 电路原理图
Fig 3-5 Schematic of CAN Bus Unit

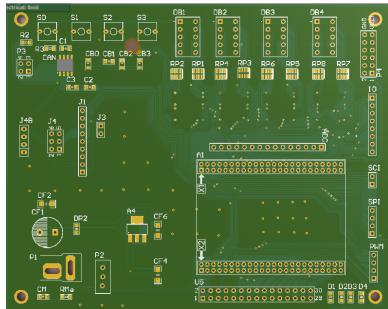


(a) PCB 正面

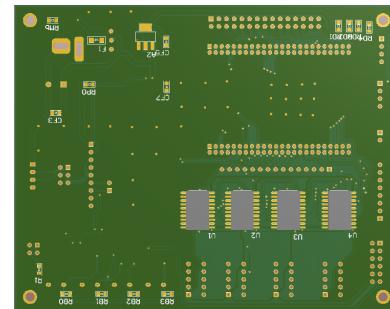


(b) PCB 背面

图 3-6 功能接口板的设计 PCB 样图
 Fig 3-6 The PCB View of Main Base Board



(a) PCB 正面



(b) PCB 背面

图 3-7 功能接口板的虚拟实物图
 Fig 3-7 The Virtual View of Main Base Board

于 Ying Zhang 等人^[17]的研究。

功能接口板的 PCB 设计图如图3-6 所示。其虚拟实物图如图3-7 所示，3D 立体仿真图如图3-8 所示。

3.2 功能型外设及接口设计

功能性外设主要提供某个模块所需要的独特功能。这些模块可能包括机器人应用中的方方面面。如无线通讯模块：用来使每个机器人模块与外部设备，如远程计算通讯，从而实现信息的沟通或是机器人与远程系统的交互；IMU 和 GPS 模块：用来实现定位，机器人姿态探测等功能；电机驱动模块，用来驱动

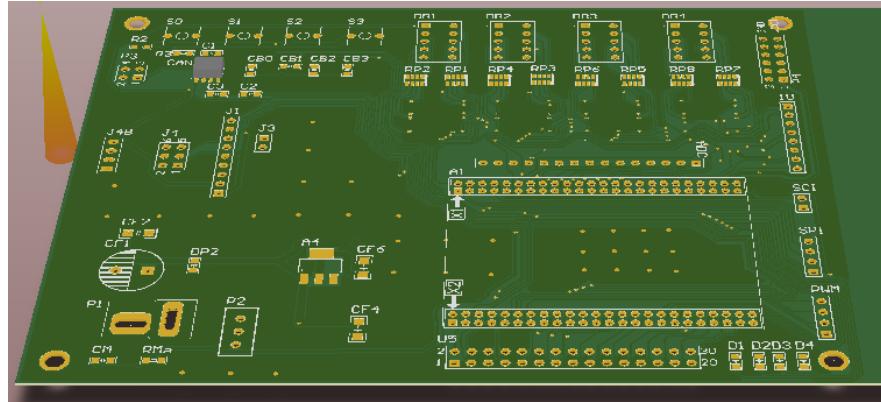


图 3-8 功能接口板的虚拟实物图 3D
Fig 3-8 The 3D Virtual View of Main Base Board

电机，可以用在轮式底盘或是履带式底盘的驱动；编码器模块：可以用来对固定于某一电机或伺服电机舵机上的编码器进行驱动与计数；传感器模块：用来驱动传感器来对外部信息进行采集和处理；以及其他应用领域的模块等等。本设计因为只对移动平台进行了设计及研究，所以只具体设计了 IMU 模块，电机驱动模块，编码器模块，传感器模块和光电 CAN 传输模块。对于每一个模块的详细设计如下。

3.2.1 IMU 模块

IMU 是 Inertial measurement unit (惯性测量单元) 的缩写。本方案所设计的 IMU 包括传统的加速度计和陀螺仪，同时参考了 Minor 等人^[18] 的设计，在模块中引入了磁倾角传感器（电子罗盘）。在这一模块的设计中，使用了一片 SCA100T 两轴角速度传感器，用来采集平行于地面的平面内的加速度；一个 ZCC212N 电子罗盘封装单元，这一单元将测得的与地磁场倾角数据通过 SCI 协议传输回单片机进行处理；以及一个 LYPR540 三轴陀螺仪，用以测量角加速度。电路设计原理图如图3-9 所示，PCB 设计样图如图3-10 所示，虚拟实物图如图3-11 所示。

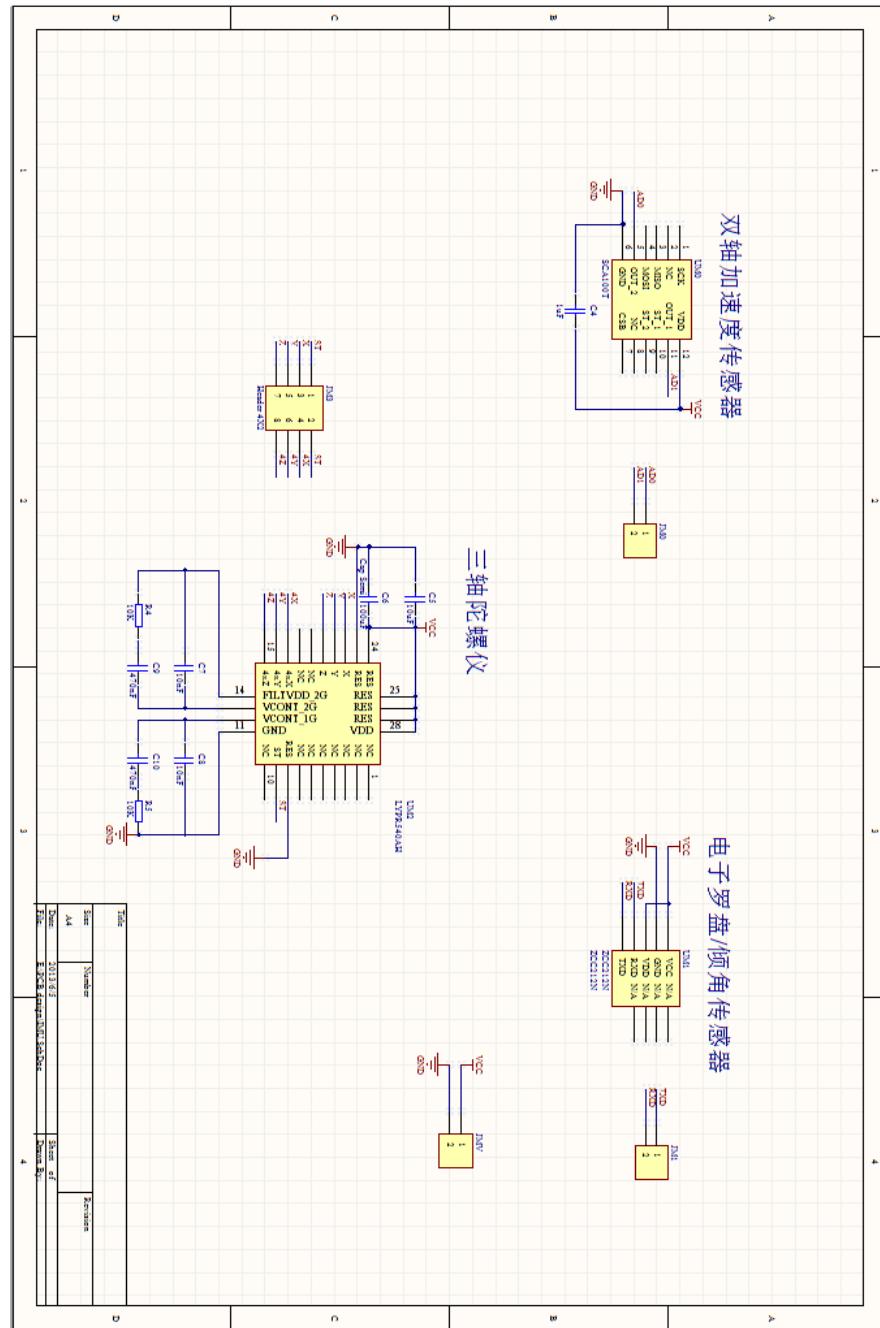
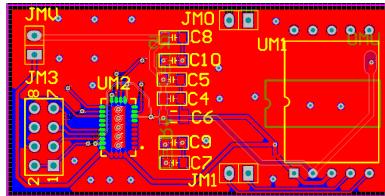
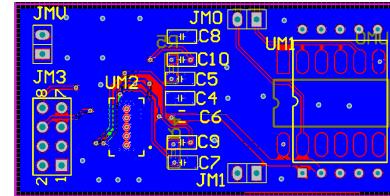


图 3-9 IMU 电路原理图
Fig 3-9 Schematic of IMU Unit



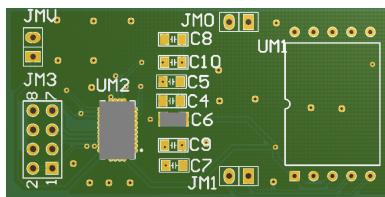
(a) PCB 正面



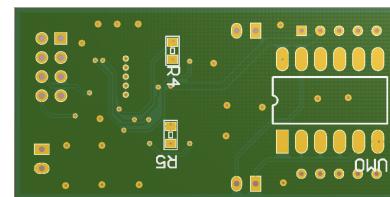
(b) PCB 背面

图 3–10 IMU 的设计 PCB 样图

Fig 3–10 The PCB View of IMU



(a) PCB 正面



(b) PCB 背面

图 3–11 IMU 的虚拟实物图

Fig 3–11 The Virtual View of IMU

3.2.2 驱动模块

本方案的驱动电路是直接基于 BTS7960 式 H 桥芯片设计的。同时电机需要使用 PWM 脉冲调制器产生方波来进行控速，通过改变方波的占空比来改变作用在电机上的等效电压来实现调速。其原理图如图3–12 所示,PCB 设计样图如图3–13 所示。

3.2.3 编码器模块

编码器模块实际上是比较简单的模块，其主要功能是为编码器提供片外计数器，并把结果反馈给主控板。在本设计中，选用的是 CD4520 芯片。这是一个最高计数值较小的高速用计数器。每个芯片的接线方法的电路原理图如图[?]。

图 3-12 驱动电路电路图

Fig 3-12 Schematic of Driver Board

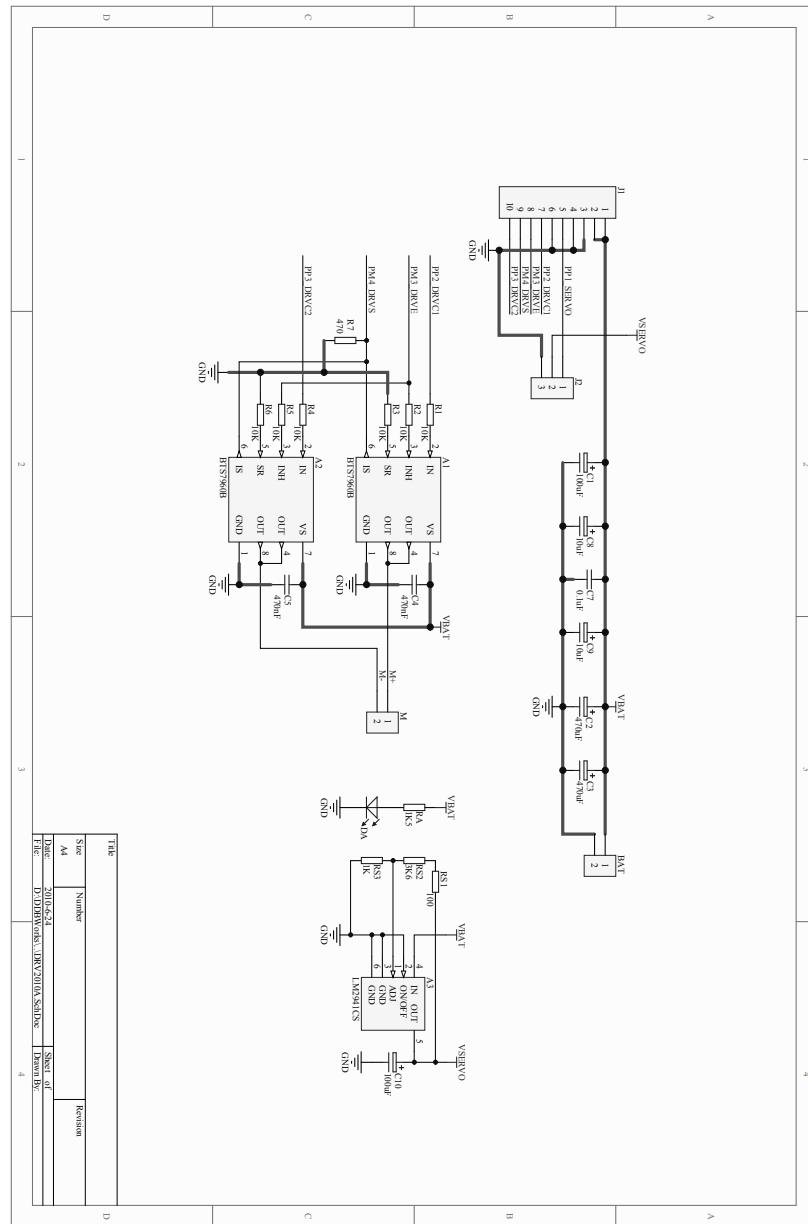




图 3-13 驱动电路的设计 PCB 样图
Fig 3-13 The PCB View of Driver Board

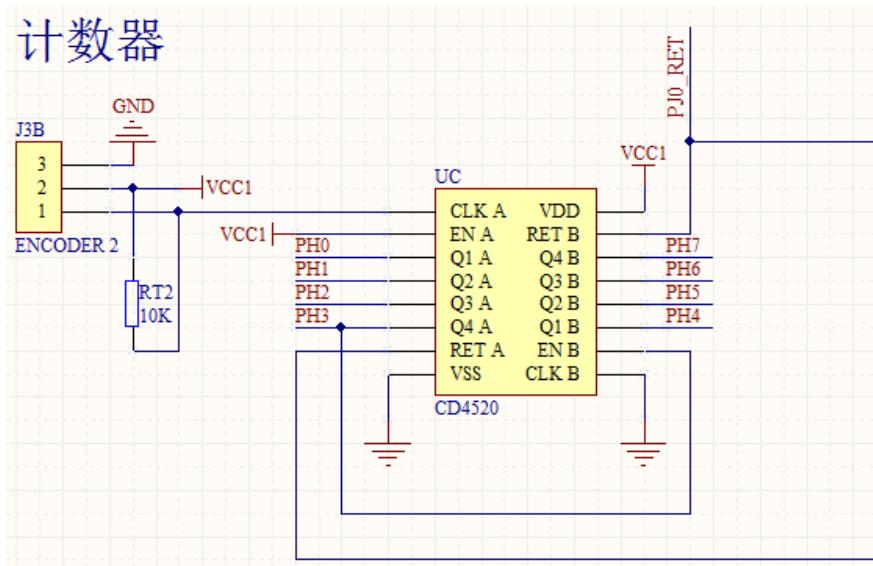


图 3-14 编码器用外部计数器接线图
Fig 3-14 Schematic of counters for encoder

3.2.4 传感器模块

传感器模块需要因传感器种类的不同而不同。在这里只对本设计需要用的超声波传感器的驱动电路进行讨论。因为传感器众多，为了节省片上资源，需要新的方法去驱动这么多传感器。其中一种方案就如笔者的设计，轮流激活传感器网络中的各个个体，这样可以复用同一套数据读取口，只需要不同的选通IO接口就可以了。这种方法充分的考虑到了片上资源不足的情况，极大的减少所要使用IO口的数量。传感器驱动网络的原理图如图3-15所示。从图中可以看出PK7口是复用的，是有上升沿和下降沿检测中断的端口。如果使用译码器驱动选通的话，可以以供使用4个IO口驱动8个传感器。

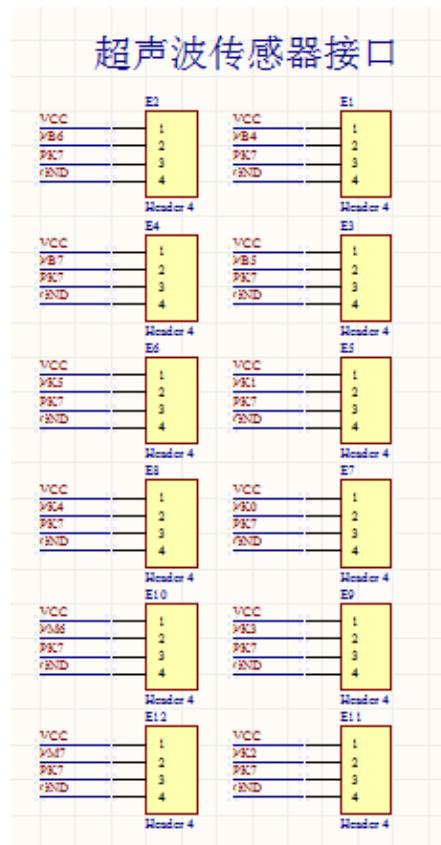


图 3-15 超声波传感器驱动板原理图
 Fig 3-15 Schematic of Ultrasonics Driver Unit

在本设计中选用的传感器是比较常见的 HC-SR04 模块。其有效检测距离是 4cm 到 8m。超声波传感器不同于红外传感器，抗光线和环境干扰能力比较强，相对可靠。同时因为模块的常见性，又使其非常廉价。这款模块的实物图如图3-16 所示。

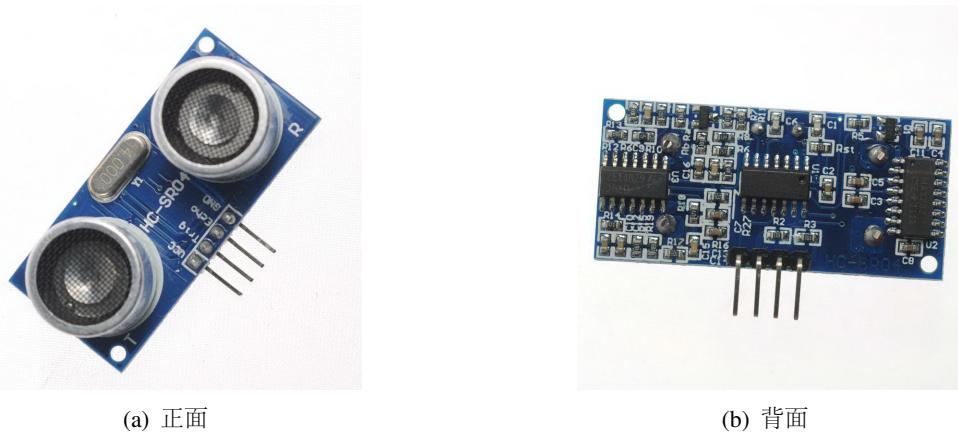


图 3-16 超声波传感器
Fig 3-16 Ultrasonics Sensor Unit

3.2.5 模块间通讯接口

模块间通讯接口是本设计中很重要的一部分，也是本设计的一个创新点。在 Ying Zhang^[17] 的关于模块化机器人的大规模控制网络系统设计中，提供了使用 CAN 总线的模块化机器人网络进行控制的很好的解决方案。这套方案给出了一个在软件层面上可行的方法，受此启发，本设计将整个模块化机器人的相互通信架构在 CAN 总线上。在前面的机械设计部分，对于为何使用光学式通道连接代替机械式触点连接给出了充分的解释。而在本章这是在电路硬件层面上将其实现。

对于 CAN 总线的每一条输出线都采用一对红外线光电对管来发送和接受相应线中的内容。CANH 中是以 2.5V 为基电位的脉冲信号，最大幅值为 5v；

而 CANL 中是以 0V 为基电位的脉冲信号，最大幅值为 2.5V。根据其这一特点，分别设计了 CANH 和 CANL 的发送单元。对于 CANH 是先将信号通过减法器，减去一个 2.5V 的参考电压，再输出到一个 MOSFET 的 G 极，控制红外 LED 的开关，来达到把信号发出的目的。CANL 则可以直接将引脚接入 MOSFET 的 G 极来控制红外 LED 的亮灭。CANH 和 CANL 的接收单元也不相同。因为接收到的光敏二极管的输出电压为一模拟值，所以要通过一定地方法把其变为数字值。对于 CANH，输出电压先经过一个比较器，变为基值是 0v，幅值是 5V 的脉冲信号，再通过加法器加上一个 2.5V 的参考电压，来变成基值为 2.5V，最大幅值为 5V 的脉冲信号，并输出回 CANH 线。对于 CANL 线，情况相似，也是要先要经过一个比较器，在对脉冲信号进行调幅。这一模块的原理图如图所示，PCB 设计样图如图所示，虚拟实物图如图所示。

3.3 本章小结

本章对多功能模块化机器人的控制系统的电路系统部分的设计进行了详尽的说明。对于每个模块化机器人来说，其电路系统分为两个部分，一个是通用模块控制电路，一个是外设功能电路。通用控制电路实现模块化机器人每个模块的基本功能，并提供足够的扩展接口；外设功能电路是帮助模块使某一功能具体实现的电路。不同的模块可以选择需要的外设功能，以在实现期望功能的前提下，减少资源的浪费。

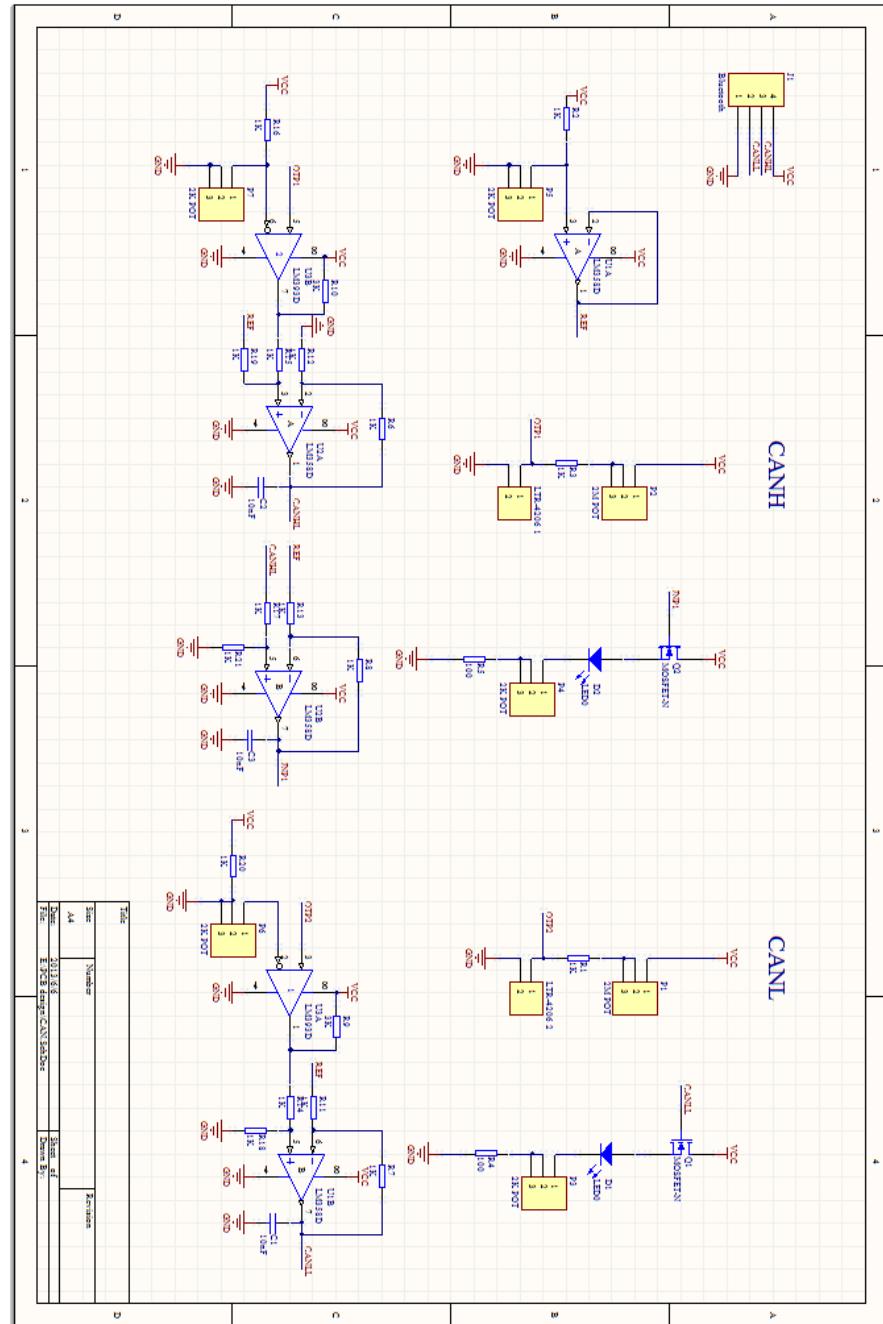
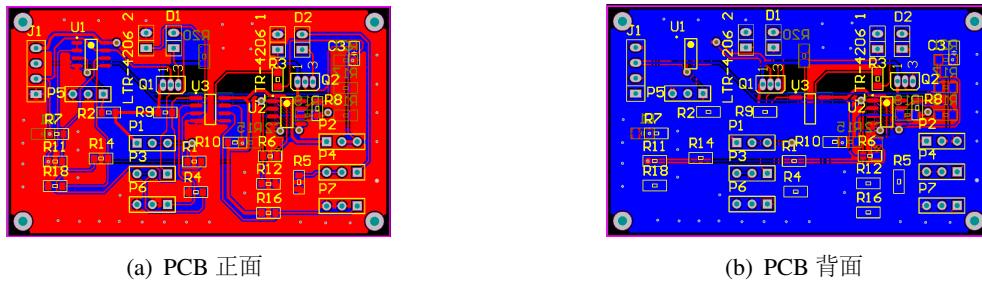


图 3-17 模块间通讯接口板电路原理图
Fig 3-17 Schematic of Modular Level Communication Unit

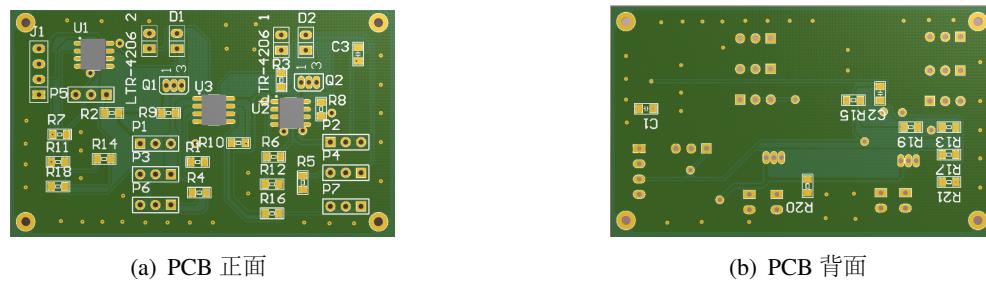


(a) PCB 正面

(b) PCB 背面

图 3-18 模块间通讯接口板设计 PCB 样图

Fig 3-18 The PCB View of Modular Level Communication Unit



(a) PCB 正面

(b) PCB 背面

图 3-19 模块间通讯接口板虚拟实物图

Fig 3-19 The Virtual View of Modular Level Communication Unit

第四章 控制与智能避障算法

在本章将主要介绍模块化移动机器人的软件系统。由于本移动机器人具有自主移动，自主定位，自动避障，地图生成和最优路径计算等功能，所以需要将其软件系统分为底盘电机驱动，车体定位，未知环境避障和探测地图绘制等部分。其中只有电机驱动部分是与选用的控制芯片有关系的，而其他诸部分皆是上层算法，所以处理驱动算法的程序不可移植外，其他部分皆可移植到其他系统中去。在下面的介绍中，除了会给出具体算法程序外，还会对每一部分算法的数学原理进行简单的介绍。

4.1 底盘驱动算法

底盘驱动采用了经典的 H 桥方案，通过单一 IO 口的高低电平来对方向进行控制，而在使能端接入 PWM 方波，通过控制占空比来控制电机的转速。控制占空比来控制转速的具体原理是，占空比的改变可以改变方波的等效电压。设一列方波的占空比是 a ，则在一个方波周期内等效电压与占空比的关系为

$$V_{org}aT = V_{equ}T \quad (4-1)$$

其中 V_{org} 是方波最大幅值， T 为方波周期， V_{equ} 为等效电压。通过公式(4-1)，可以得到

$$V_{equ} = aV_{org} \quad (4-2)$$

等效电压与占空比呈简单线性关系。而通过电机的特性公式(4-3)

$$V_{supply} = K_e \dot{\theta} \quad (4-3)$$

可知电机转速和输入电压 V_{supply} 成正比，这就意味轮子的转速与方波占空比成正比。所以控制轮子的转速只需要改变输入方波的占空比就可以了。但是占空比只给出了一个速度的相对关系，并没有与具体速度值的对应，所以控制的时候还要通过传感器的数据，对速度控制进行闭环。

电机系统所需输出口初始化，和一般电机控制程序如下。此段程序，是参考罗翔的程序所撰写。

代码 4.1 针对 BTS7960 的单片机初始化程序

```

1 #include "BTS7960B.h"
2 #include "SystemConfig.h"
3
4 /* Crystal's frequency is 40MHz
5   USE PPL change the bus frequency*/
6 //——宏参数定义——
7 #define HIGH 1
8 #define LOW 0
9 #define OUTPUT 1
10 #define INPUT 0
11
12 #define SETPWM23 0x0C          //置1 PWM23 选择项
13 #define CLRPWM23 (~SETPWM23)    //清零 PWM23 选择项
14 #define CLKB1D08 0x30          //ClockB 运行方式
15
16 #if SYS_BUS_FREQUENCY == SYSTEM_BUS_64M
17 //64M-预分频: fA=2M, fB=4M(使用 fB)
18 #define _PWMMAX 199           //占空比最大值
19 #define _PWMMIN 0              //占空比最小值

```

```
20 #define _PWMUNT 2
21 #else
22 //其他分频: fA = 2.5MHz, fB = 5MHz, 请在此基础上设定频率
23 #define _PWMMAX 249          //占空比最大值
24 #define _PWMMIN 0           //占空比最小值
25 #define _PWMUNT 5/2
26 #endif
27
28 //——操作函数定义——
29 void BTS7960BRestart(void)
30 //重启函数: 对需使用 I/O 口初始化
31 {
32     DDRM_DDRM5 = OUTPUT;
33     _BTSDRVE = LOW ;
34     DDRM_DDRM4 = INPUT ;
35     _BTSDRVS = LOW ;//Initial BITS OF PTM PORT
36
37     PWME &= CLRPWM23;//关 PWM23
38
39     //PWMPRCLK |= CLKB1D08;//ClockB frequency = 40MHz/8=5MHz
40     //PWMSCLB = 5 ;
41
42     PWMPER2 = _PWMMAX ;//PWM2 frequency = 10kHz
43     PWMDTY2 = _PWMMAX ;//占空比 0
44     PWMPER3 = _PWMMAX ;//PWM3 frequency = 10kHz
45     PWMDTY3 = _PWMMAX ;//占空比 0
46
47     PWME |= SETPWM23;
48
49     BTS7960BEnable(HIGH) ;
50 }
51
52 unsigned char BTS7960BControl(unsigned char ForwardPWM, unsigned char
```

```

      BackwardPWM)

53 //控制函数: 正反转控制。输入 PWM 占空比, 输出正反转状态(输入均为百分之)
54 {
55   unsigned char PStatus = BTS7960STOP;
56   //PWM 状态赋值
57   PStatus = (ForwardPWM > BackwardPWM)?BTS7960FORWARD:
58   BTS7960BACKWARD;//电机运行状态判
59   断
60   PStatus = (ForwardPWM == BackwardPWM)?BTS7960STOP :PStatus ;
61
62   ForwardPWM = (ForwardPWM > 100)?100:ForwardPWM ;
63   BackwardPWM = (BackwardPWM > 100)?100:BackwardPWM;
64   PWMDTY2 = (byte)((_PWMMAX < ForwardPWM * _PWMUNT)?_PWMMIN:(
65   _PWMMAX - ForwardPWM *_PWMUNT));//计算 PWM 占空
66   比
67   PWMDTY3 = (byte)((_PWMMAX < BackwardPWM * _PWMUNT)?_PWMMIN:(
68   _PWMMAX - BackwardPWM*_PWMUNT));
69   return PStatus ;
70 }

```

在底盘模块正常情况下是沿着算法所确定的方向匀速行进，速度只需规定一个为常数的占空比值就可以了。但是因为电机特性不一致的原因，即便规定同一 PWM 值，车体行进方向仍然可能偏离既定方向，所以要使底盘保持行进方向，就要对其行进角度进行闭环。这一功能的可以通过电子罗盘返回的数据来设计实现。这里将采用传统的 PID 算法来进行闭环程序的设计。PID 系统的传递方程的拉普拉斯形式可以写为

$$G(s) = K_p + \frac{K_i}{s} + K_d s \quad (4-4)$$

其中 K_p, K_i 和 K_d 为 PID 方法的参数，对于系统方程，输入量是实际角度与计算角度的差值，输出量是驱动左轮与驱动右轮 PWM 占空比的差值。因为系统

实际是离散的，所以需要将系统方程从 S 域映射到离散系统的 Z 域，关于这个控制器的 C 程序就是根据这一思想设计的，其伪代码可以表示如下。

代码 4.2 PID 控制器伪代码

```
1 double KP = 一个合理的值;
2 double KI = 一个合理的值;
3 double KD = 一个合理的值;
4 double Error = 0;           // 实际角度与计算角度的偏差
5 static double ErrorHis = 0; // 对于角度偏差的记录
6 static double ErrorAcu = 0; // 偏差积分
7
8 Error = AngleActual - AngleCalculated;
9 ErrorAcu += Error ;
10 DutyCycleDiff = KP*Error + KD*(Error-ErrorHis) + Ki*ErrorAcu ;
11 ErrorHis = Error;
```

关于底盘模块行进方向角度的计算将在未知环境避障算法部分介绍，关于遇到避障时转向的控制也将在那部分章节介绍。

4.2 车体定位方案

车体定位方案将直接决定了底盘模块的行进轨迹，对于是否完成目标的判断和地图的绘制等功能的实现。为了获得机器人准确的位置，在本设计中，使底盘集成了多种传感器模块。这其中包括轮式编码器，俗称拖地轮；IMU 模块等单元，如果需要，则可以通过串口扩展，很容易的链接 GPS 模块。在下文中将详细介绍各个方案定位的数学原理，及坐标值获得的方法。

4.2.1 拖地轮及方位角传感器定位

拖地轮的机械结构设计已经在第二章的相应部分进行详细的叙述，在本节，将主要介绍基于拖地轮的定位算法。拖地轮的设计目标之一就要简化计

算，所以将拖地轮固定在底盘模块相邻的两条边中央的位置上。底盘模块的坐标系，与轮子的位置如图4-1所示。

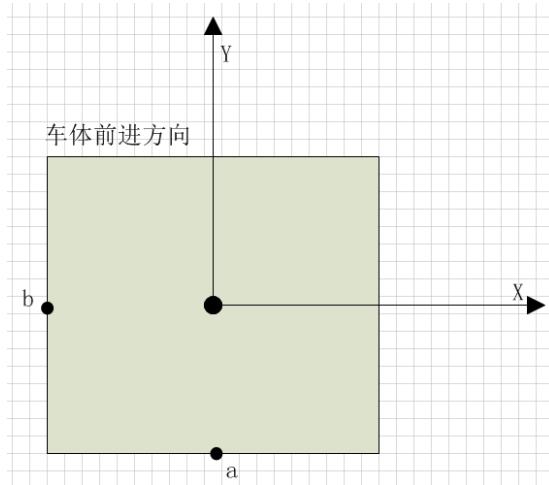


图 4-1 底盘上坐标系的定义
Fig 4-1 The Definition of the Coordinates On the Chassis

其中拖地轮 a 只能采集到与底盘模块 y 坐标轴平行的转动，同理拖地轮 b 只能采集到与底盘模块 x 坐标轴平行的转动。底盘的控制芯片将会以一定的周期进行对传感器信息的采集和对底盘模块体的控制。这一周期在本设计中为 10 毫秒，所以控制周期是 100HZ。当每一个周期结束后，如果只有 y 方向的轮子采集到数据，则机器人的世界坐标改变量为

$$\begin{cases} \Delta x_{world} = \Delta y \cdot \sin \theta \\ \Delta y_{world} = \Delta y \cdot \cos \theta \end{cases} \quad (4-5)$$

其中 x_{world} , y_{world} 为机器人的世界坐标系中的坐标， θ 为机器人前进方向与世界坐标 y 轴所成的夹角。

同理当只有 x 方向的轮子采集到位移数据时，则机器人的世界坐标为

$$\begin{cases} \Delta x_{world} = \Delta x \cdot \cos \theta \\ \Delta y_{world} = \Delta x \cdot \sin \theta \end{cases} \quad (4-6)$$

而在同时采集到 x 与 y 方向的位移信息时，就需要用到一定的数学技巧来获得世界坐标的地位改变量。如图4-2所示，因为控制周期比较短，可以用直角模型来计算机人人体的位移，并假设车体的运动方向是初始时的运动方向保持不变，这样就可以通过机器人前进方向与世界坐标 y 轴的夹角和采集到的传感器信息来计算车体的实际位移了。

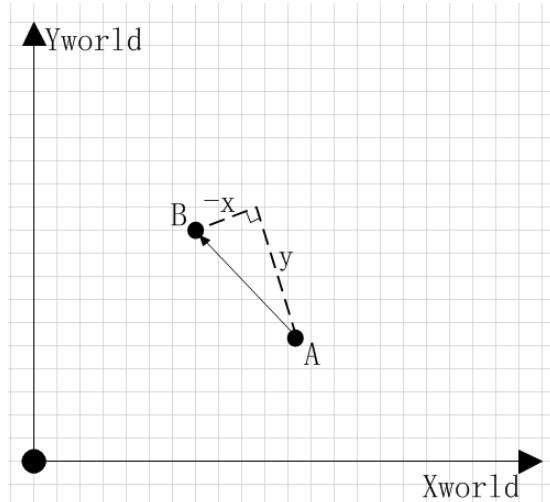


图 4-2 机器人运动示意
 Fig 4-2 The Motion of the Robot in World Coordinates

这时世界坐标该变量的计算公式为

$$\begin{cases} \Delta x_{world} = \Delta y \cdot \sin \theta - \Delta x \cdot \cos \theta \\ \Delta y_{world} = \Delta y \cdot \cos \theta + \Delta x \cdot \sin \theta \end{cases} \quad (4-7)$$

4.2.2 IMU 定位

通过 IMU 上所拥有的传感器元件可知，其对于位置的确定是依赖于积分的。加速度传感器的输出量是加速度，陀螺仪的输出量是角加速度，电子罗盘的输出量是角度。表征速度，位移和加速度关系的公式是

$$D = \int \dot{D} = \int_T \int_T \ddot{D} \quad (4-8)$$

所以只要对加速度进行二次积分就可以算出在某一轴上的位移分量。因为控制周期比较短，所以可以认为在一个周期 T 内，机器人是在做匀加速度，匀速度运动。如果以 \ddot{x} 和 \ddot{y} 作为在底盘模块参考系中的加速度传感器所能测出的加速度量，则坐标的数学表达式可以写作

$$\begin{cases} \Delta x_{world} = \ddot{y} \cdot T^2 \cdot \sin \theta - \ddot{x} \cdot T^2 \cdot \cos \theta \\ \Delta y_{world} = \ddot{y} \cdot T^2 \cdot \cos \theta + \ddot{x} \cdot T^2 \cdot \sin \theta \end{cases} \quad (4-9)$$

但是因为加速度传感器的噪声带会对积分误差的收敛造成影响，所以光靠加速度传感器的值做计算是不够的。还需要加上陀螺仪的输出值作参考。加速度值与陀螺仪测得的关于垂直于地面平面的转轴的角加速度值的关系是

$$\arctan\left(\frac{\ddot{x} \cdot T^2}{\ddot{y} \cdot T^2}\right) = \int_T \int_T \ddot{\phi} \quad (4-10)$$

其中 ϕ 是角加速度。

但是仅仅是使用这个比较依然是不可靠的，还要对采集到的数据进行滤波。本设计根据 Houtekamer 等人^[19] 的文章，设计了对信号处理的卡尔曼滤波器。其 C 程序实现如下。

代码 4.3 Kalman 滤波的 C 程序代码

```
1 float angle, angle_dot; // 外部需要引用的变量
```

```
2 float Q_angle = 0.05, Q_gyro = 0.01, R_angle = 5, dt = 0.005;
3 float P[2][2] = {{ 1, 0 },{ 0, 1 }} ;
4 float Pdot[4] ={0,0,0,0} ;
5 const char C_0 = 1;
6 float q_bias, angle_err, PCt_0, PCt_1, E, K_0, K_1, t_0, t_1;
7 //_____
8 void Kalman_Filter( float angle_m, float gyro_m)
9 {
10     angle+=(gyro_m-q_bias) * dt;           //先验估计
11
12     Pdot[0]=Q_angle - P[0][1] - P[1][0];    //先验估计误差协方差的微分
13     Pdot[1]=- P[1][1];
14     Pdot[2]=- P[1][1];
15     Pdot[3]=Q_gyro;
16
17     P[0][0] += Pdot[0] * dt;                 // 先验估计误差协方差
18     P[0][1] += Pdot[1] * dt;
19     P[1][0] += Pdot[2] * dt;
20     P[1][1] += Pdot[3] * dt;
21
22
23     angle_err = angle_m - angle;
24
25     PCt_0 = C_0 * P [0][0];
26     PCt_1 = C_0 * P [1][0];
27
28     E = R_angle + C_0 * PCt_0;
29
30     K_0 = PCt_0 / E; //Kk
31     K_1 = PCt_1 / E;
32
33     t_0 = PCt_0;
34     t_1 = C_0 * P [0][1];
```

```
35
36     P[0][0] -= K_0 * t_0;           //后验估计误差协方差
37     P[0][1] -= K_0 * t_1;
38     P[1][0] -= K_1 * t_0;
39     P[1][1] -= K_1 * t_1;
40
41     angle += K_0 * angle_err;      //后验估计
42     q_bias += K_1 * angle_err;     //后验估计
43     angle_dot = gyro_m - q_bias;  //输出值（后验估计）的微分 = 角速度
44 }
```

4.3 未知环境避障算法

在本设计中采用了虚拟势能场的算法来实现路径计算和自动避障的。这一算法最先由 Khatib^[20] 提出，并应用于机械臂行进路线的规划。后来虚拟势场的概念经由 Yoram Koren 等人^[21] 的发展，而可以应用到平面移动物体的二维运动规划中。

基本的虚拟势场思想主要是引入了两个虚拟力。一个为虚拟引力，一个为虚拟斥力。虚拟引力是指运动体与目标点之间的力，虚拟斥力是指运动体与障碍物之间的力。目标点在全局坐标系中的位置是已知的，所以在初始状态下就存在着引力场的作用，是机器人向目标点运动。因为势能的等势线为圆形，所以只有运动物体向着目标点前进，才是势能变化梯度最大方向。因此在某种意义上讲，由虚拟引力所计算出的运动方向是最优的。在本设计中，设计的机器人所处的环境是未知的，就是说初始条件下并无障碍物的存在，而在运动过程中会遇到障碍物，并在障碍物的斥力影响下而改变运动方向。这说明斥力是一种短程力，它的作用范围是有限的。

在本设计中接纳了耿兆丰等人^[22] 的方案，将目标点产生的势场 P 定义为

$$P = r^2/2 = [(x - x_t)^2 + (y - y_t)^2]/2 \quad (4-11)$$

其中 x_t, y_t 为目标点坐标。则此时机器人所处位置的势场梯度可以表示为

$$\nabla P = \frac{\partial P}{\partial x} i + \frac{\partial P}{\partial y} j = (x - x_t) i + (y - y_t) j \quad (4-12)$$

因此引力可以表示为

$$F_{att} = K(|\nabla P|_{r=d}) = Kd \quad (4-13)$$

其中 K 为引力常数。

对于斥力，正如上文所说，为短程力，所以其应该具有随着距离的变长而减弱的特点，如果斥力能与距离成反比的话，将符合这一要求。所以斥力的形式应为

$$F_{rep} = \frac{K_r}{(x - x_o)^2 + (y - y_o)^2} \quad (4-14)$$

其中 K_r 为斥力常数。与引力不同，斥力是一个三维力，计算斥力时不仅相对于质心坐标，而是相对于传感器所在的点，所以 x_o 和 y_o 是每个传感器点的坐标。在此处分母使用距离的平方是因为，第一可以简化计算，提高计算速度；第二可以加快斥力的衰减和增加。斥力的作用方向即为力场梯度方向。

$$\nabla F_{rep} = -2 \frac{K_r(x - x_o)}{((x - x_o)^2 + (y - y_o)^2)^2} i - 2 \frac{K_r(y - y_o)}{((x - x_o)^2 + (y - y_o)^2)^2} j \quad (4-15)$$

其方向矢量可以化简的表示为 $[-(x - x_o) \quad -(y - y_o)]$ 。因为斥力是个短程力，所以要设定一个阀值，使斥力小于这一阀值时为 0。

在有了引力与斥力的公式后，就可以对每个点对车体的作用积分而算出合

力，再通过合力的方向来计算机器人的行动方向。本机器人的控制系统所有的方向都表示为机器人前进方向与世界坐标系的y轴的夹角。算法的示意图如图^[?]所示。

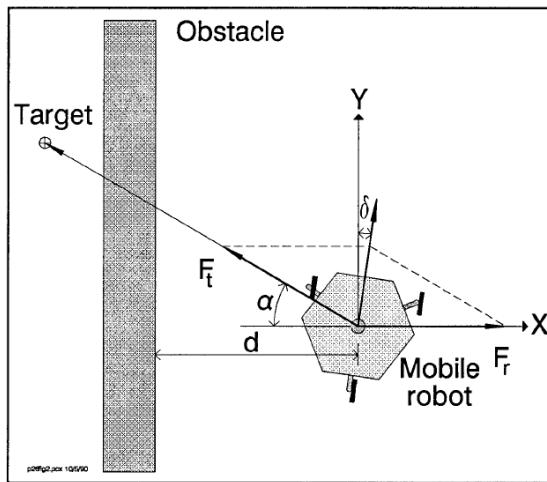


图 4-3 虚拟势能避障算法示意图^[21]
 Fig 4-3 Motion of a mobile robot applying a potential field algorithm

但是这种算法存在的问题就是，如果障碍物足够大，可能存在着受力平衡点，使机器人永远无法越过障碍物。为了改变这一现象，不得不对机器人引入惯性。但是光引入惯性还是不够的，同时还要引入运动方向锁，来使最后的搜索过程可以得到收敛解。

4.4 地图生成和最优路径生成

关于地图的绘制，是受到了 David Waller 等人^[23]研究的启发，同时参考了周斌等人^[24]的研究。底盘模块进行这样设计的原因是如果需要多次重复的使机器人从某一出发点到达某一目标点，在经过第一次机器人的探索后，可以记住环境信息，并在下次进行相同任务时，可以直接沿着最优路径前进而不需要重新探测。则可以大大加快机器人在同一环境长时间重复执行同一任务时的效率。

与周斌给出的方法不同，在本设计中，地图信息并不是保存在车体的控制器里，而是底盘模块会将数据发回给上位机，并由上位机处理并绘制地图，再通过上位机来计算最优路径，返回给片上系统。地图绘制的过程主要还是靠传感器模块标出障碍物的坐标，并根据这一坐标和机器人本身坐标尺寸来绘制障碍物和机器人可通过通道。

而最优路径生成这是通过现将绘制的地图网格化，然后从目标点出发向起始点回溯的去评价每一个点。评价公式就是与目标点的距离。在评价过程中只关心处在网格节点上的点，只要网格足够小，最后所得到的结果是可信的。当所有点评价完毕后，出发点会有多种评价值。找出最小的评价值，就可以找出最优的路径。然后连接这条线路上的每一个点，对这条路径的关键点信息进行计算并返回给底盘模块进行存储。这样就完成了最优路径生成的过程。

4.5 本章小结

本章主要介绍了所设计的移动模块化机器人系统的软件部分。这部分内容主要是针对轮式底盘和超声波传感器部分所设计的。本章节只涉及到了本移动式模块化平台的关键算法。关键算法包括 4 部分，第一为底盘的驱动算法，这是其他对车体控制算法的基础；其次是车体定位算法，这些算法是决定避障是否可行的关键；接下来是避障算法，详细介绍了机器人是如何能绕过障碍物而自主向目标点行进的；最后是地图的绘制和最优路径的生成，其中地图绘制是最优路径生成的基础。

第五章 避障算法的仿真

关于移动化机器人底盘系统的设计说明已经在前三章完成，在本章将对于算法的仿真做以详细的介绍。因为机器人的控制算法相对复杂，同时又涉及到在三维空间内的避障，所以一定要先进性动力学仿真，来验证算法的可行性。

5.1 仿真软件环境介绍

本设计将会使用 GAZEBO 动力学仿真环境来对在设计中出现的算法进行仿真。Gazebo 是一个多机器人仿真环境，目前软件有能力在三维环境中仿真已有的多种机器人，传感器和其他物体。在仿真环境里，可以模拟生成传感器返回量和根据物理引擎来模拟物体之间的相互作用，其尤为擅长对于刚体物理特性的仿真。GAZEBO 具有一种分布式的结构，它的功能被封装在不同的库中，如物理仿真引擎，渲染，UI，通讯和传感器生成。在仿真中将涉及三个不同的部分：物理仿真，传感器生成，GUI 界面，最后再加上一个全局坐标管理其。

5.2 移动机器人模型的建立

在 GAZEBO 中，所有机器人模型都是通过通用 SDF 文件来描述的，SDF 文件是一个符合标准 XML 协议的文件。对于一个模型的描述包括四个部分：Link，包含着一个物理模型的物理性质信息；collision，一个 collision 包含着一个需要检查碰撞的空间，同时其还有创建可视图形，定义惯性和粘连传感器；joint，定义了可以相对移动的不同部分的连接部分；组件，是一个公用库，定义了如何控制模型的方法。

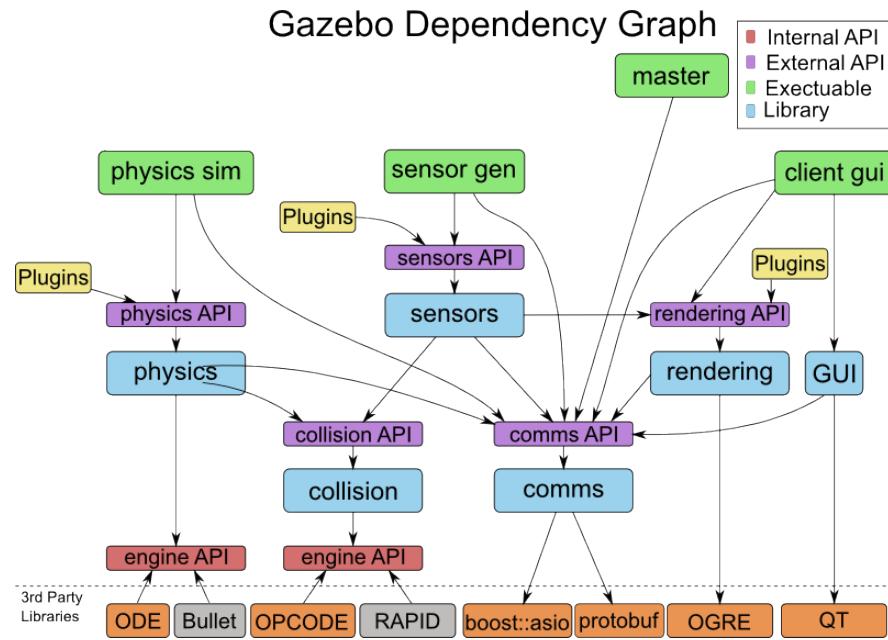


图 5-1 GAZEBO 系统结构图
Fig 5-1 GAZEBO system diagram

对于本设计的机器人，因为要进行三维的碰撞检查，所以需要定义一个需要检查的空间；同时因为还有避障的需求，所以要在机器体上粘接 12 个超声波传感器；机器人是靠两个轮子进行驱动，因此还要加入两个驱动轮和一个万向轮。依据这些要求，说建立的机器人模型为

代码 5.1 机器人模型描述文件

```

1 <?xml version='1.0'?>
2 <sdf version='1.3'>
3   <model name="my_robot">
4     <link name='chassis'>
5       <pose>0 0 0 0 0</pose>
6       <collision name='collision'>
7         <geometry>
8           <box>
9             <size>.4 .2 .1 </size>

```

```
10      </box>
11      </geometry>
12  </ collision >
13
14  <visual name='visual'>
15      <geometry>
16          <box>
17              <size>.4 .2 .1</ size>
18          </box>
19      </geometry>
20  </ visual >
21
22  < collision name='caster_collision'>
23      <pose>-0.15 0 -0.05 0 0 0</pose>
24      <geometry>
25          <sphere>
26              <radius>.05</radius>
27          </sphere>
28      </geometry>
29
30      <surface>
31          < friction >
32              <ode>
33                  <mu>0</mu>
34                  <mu2>0</mu2>
35                  <slip1>1.0</slip1>
36                  <slip2>1.0</slip2>
37              </ode>
38          </ friction >
39      </surface>
40  </ collision >
41
42  <visual name='caster_visual'>
```

```
43 <pose>-0.15 0 -0.05 0 0 0</pose>
44 <geometry>
45   <sphere>
46     <radius>.05</radius>
47   </sphere>
48 </geometry>
49 </ visual >
50 </link >
51 <link name="left_wheel">
52   <pose>0.1 0.13 0.1 0 1.5707 1.5707</pose>
53   < collision name="collision">
54     <geometry>
55       <cylinder>
56         <radius>.1</radius>
57         <length>.05</length>
58       </cylinder>
59     </geometry>
60   </ collision >
61   <visual name="visual">
62     <geometry>
63       <cylinder>
64         <radius>.1</radius>
65         <length>.05</length>
66       </cylinder>
67     </geometry>
68   </visual >
69 </link >
70
71 <link name="right_wheel">
72   <pose>0.1 -0.13 0.1 0 1.5707 1.5707</pose>
73   < collision name="collision">
74     <geometry>
75       <cylinder>
```

```
76      <radius>.1</radius>
77      <length>.05</length>
78      </cylinder>
79      </geometry>
80      </collision>
81      <visual name="visual">
82          <geometry>
83              <cylinder>
84                  <radius>.1</radius>
85                  <length>.05</length>
86              </cylinder>
87          </geometry>
88      </visual>
89  </link>
90  <joint type="revolute" name="left_wheel_hinge">
91      <pose>0 0 -0.03 0 0 0</pose>
92      <child>left_wheel</child>
93      <parent>chassis</parent>
94      <axis>
95          <xyz>0 1 0</xyz>
96      </axis>
97  </joint>
98
99  <joint type="revolute" name="right_wheel_hinge">
100     <pose>0 0 0.03 0 0 0</pose>
101     <child>right_wheel</child>
102     <parent>chassis</parent>
103     <axis>
104         <xyz>0 1 0</xyz>
105     </axis>
106  </joint>
107  </model>
108
```

```
109 <include>
110   <uri>model:// ultrasonic </uri>
111   <pose>0.2 0 0.2 0 0 0</pose>
112 </include>
113 <joint name="ultrasonic_joint" type="revolute">
114   <child>hokuyo::link</child>
115   <parent>chassis</parent>
116   <axis>
117     <xyz>0 0 1</xyz>
118     <limit>
119       <upper>0</upper>
120       <lower>0</lower>
121     </ limit >
122   </axis>
123 </ joint >
124 </sdf>
```

配置文件为

代码 5.2 机器人模型配置文件

```
1 <?xml version="1.0"?>
2 <model>
3   <name>My Robot</name>
4   <version>1.0</version>
5   <sdf version='1.3'>model.sdf</sdf>
6
7   <author>
8     <name>Cui Yunkai</name>
9     <email>tccyk@sjtu.edu.cn</email>
10    </author>
11
12   < description >
```

```
13   My awesome robot.  
14   </ description >  
15   </model>
```

5.3 机器人所处的仿真世界的建立

对于算法的验证，我们只需要随机建立一个充满障碍物的世界，唯一要保证的就是从起始点到终点是有解的。本设计所建立的世界如图5-2 所示，相关代码请见附录A。

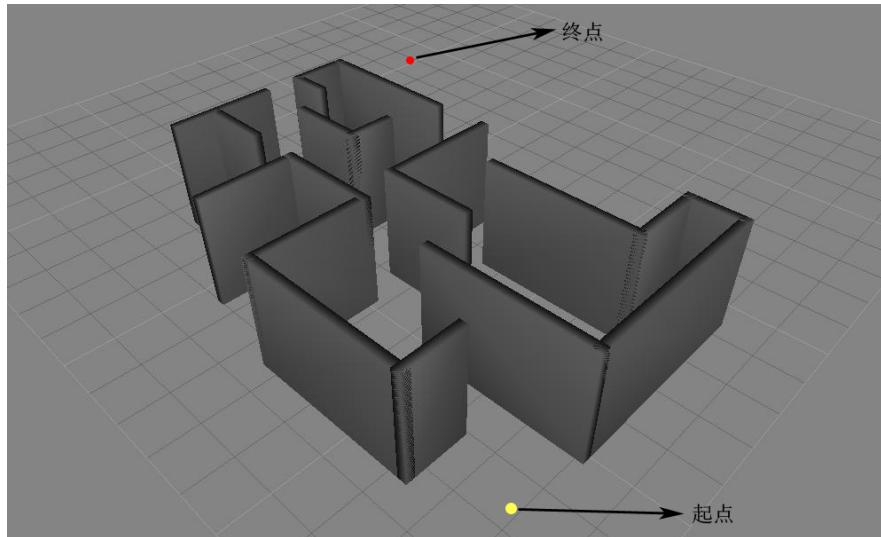


图 5-2 仿真所建立的障碍物世界

Fig 5-2 World in simulation

5.4 机器人的扩展组件撰写

机器人的扩展组件是指机器人需要实现某一功能而需要拥有的方程文件，GAZEBO 支持 C++ 进行组件编写。在本设计中需要通过模型控制文件来实现避障算法。

代码 5.3 机器人控制扩展组件

```
1 #include <boost/bind.hpp>
2 #include <gazebo.hh>
3 #include <physics/physics.hpp>
4 #include <sensors/sensors.hpp>
5 #include <common/common.hpp>
6 #include <stdio.h>
7
8 namespace gazebo
9 {
10    class MobileBasePlugin : public ModelPlugin
11    {
12        public: void Load(physics::ModelPtr _parent, sdf::ElementPtr _sdf)
13        {
14
15            // Store the pointer to the model
16            this->model = _parent;
17
18            // Load parameters for this plugin
19            if (this->LoadParams(_sdf))
20            {
21                // testing to see if race condition exists
22                gzerr << this->leftWheelJoint->GetAngle(0) << "\n";
23                gzerr << this->rightWheelJoint->GetAngle(0) << "\n";
24                // Listen to the update event. This event is broadcast every
25                // simulation iteration.
26                this->updateConnection = event::Events::ConnectWorldUpdateStart(
27                    boost::bind(&MobileBasePlugin::OnUpdate, this));
28            }
29        }
30
31        public: bool LoadParams(sdf::ElementPtr _sdf)
32        {
```

```
33
34     // Find controller gain
35     if (_sdf->HasElement("gain"))
36     {
37         gzerr << "param [gain] not found\n";
38         return false ;
39     }
40     else
41     {
42         // Get sensor name
43         this ->gain =
44             _sdf->GetElement("gain")->GetValueDouble();
45     }
46
47     // Find sensor name from plugin param
48     if (_sdf->HasElement("ray_sensor"))
49     {
50         gzerr << "param [ray_sensor] not found\n";
51         return false ;
52     }
53     else
54     {
55         // Get sensor name
56         std :: string sensorName =
57             _sdf->GetElement("ray_sensor")->GetValueString();
58
59         // Get pointer to sensor using the SensorManager
60         sensors :: SensorPtr sensor =
61             sensors :: SensorManager::Instance ()->GetSensor(sensorName);
62
63         if (!sensor)
64         {
65             gzerr << "sensor by name ["
```

```
66             << sensorName
67             << "] not found in model\n";
68         return false ;
69     }
70
71     this ->laser = boost :: shared_dynamic_cast<sensors :: RaySensor>
72     (sensor);
73     if (! this ->laser)
74     {
75         gzerr << "laser by name ["
76         << sensorName
77         << "] not found in model\n";
78         return false ;
79     }
80 }
81
82 // Load joints from plugin param
83 if (! this ->FindJointByParam(_sdf, this ->leftWheelJoint,
84                               "left_wheel_hinge") ||
85     ! this ->FindJointByParam(_sdf, this ->rightWheelJoint,
86                               "right_wheel_hinge"))
87     return false ;
88
89 // success
90 return true ;
91 }
92
93 public : bool FindJointByParam(sdf :: ElementPtr _sdf,
94                                 physics :: JointPtr &_joint,
95                                 std :: string _param)
96 {
97     if (!_sdf->HasElement(_param))
98     {
```

```
99     gzerr << "param [" << _param << "] not found\n";
100    return false ;
101 }
102 else
103 {
104     _joint = this->model->GetJoint(
105         _sdf->GetElement(_param)->GetValueString());
106
107     if (! _joint)
108     {
109         gzerr << "joint by name ["
110             << _sdf->GetElement(_param)->GetValueString()
111             << "] not found in model\n";
112     return false ;
113     }
114 }
115 return true ;
116 }
117
118 // Called by the world update start event
119 public : void OnUpdate()
120 {
121     unsigned int n = this->laser->GetRangeCount();
122     double min_dist = 1e6;
123     for (unsigned int i = 0; i < n; ++i)
124     {
125         if (this->laser->GetRange(i) < min_dist)
126             min_dist = this->laser->GetRange(i);
127     }
128
129     double target_dist = 2.0;
130
131     if (min_dist < this->laser->GetRangeMax())
```

```
132  {
133      double torque = this->gain*( min_dist - target_dist );
134      this->leftWheelJoint->SetForce(0, torque);
135      this->rightWheelJoint->SetForce(0, torque);
136  }
137 }
138
139 // Pointer to the model
140 private : physics :: ModelPtr model;
141
142 // Pointer to the update event connection
143 private : event :: ConnectionPtr updateConnection;
144
145 private : physics :: JointPtr leftWheelJoint ;
146 private : physics :: JointPtr rightWheelJoint ;
147 private : sensors :: RaySensorPtr laser ;
148 private : double gain;
149 };
150
151 // Register this plugin with the simulator
152 GZ_REGISTER_MODEL_PLUGIN(MobileBasePlugin)
153 }
```

5.5 仿真过程与算法验证

在不断对代码修改后，最终仿真得以实现。仿真截图如图所示。其中蓝色部分为传感器前侧所能感知到的区域，图中车体正在向障碍物间空隙前进。因为在仿真中所计算的数据量非常大，所以仿真速度相对较慢，走出迷宫大概花了 25 小时 46 分钟。不过最后小车基本顺利到达指定位置，经测试说明在上章中提出的虚拟势场算法基本可行。

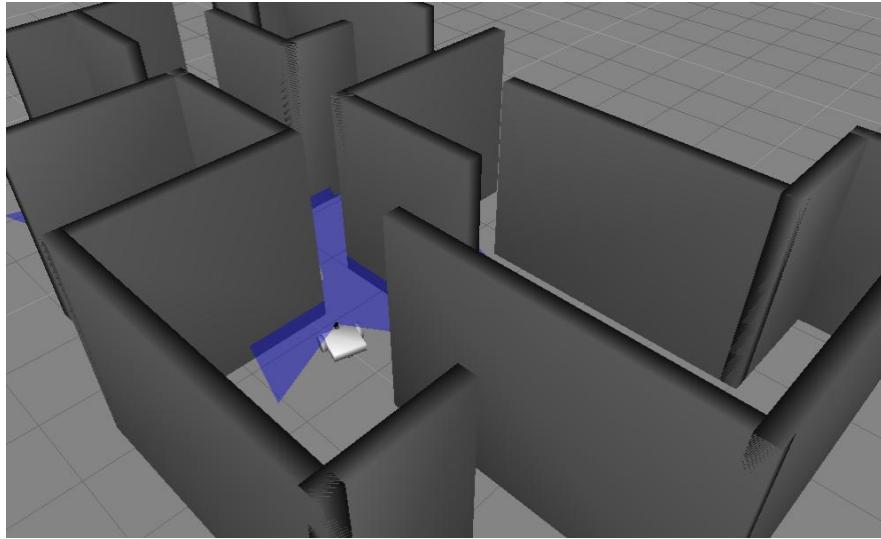


图 5-3 仿真进行示意图
Fig 5-3 The simulation is on going

仿真中存在着的问题也比较多，比如算法对于环境还是有一定的依赖性，并不能完全脱离某一种特定的对环境的要求。所以算法还可以据此进行一些改进。但因为在本设计中使用的环境也即为复杂，而实际中的环境并没有这么复杂，所以说，算法在一般的环境中还是可以正常工作的。所以仿真基本是成功的。而对于这个机器人控制系统的设计也由此基本完成。

5.6 本章小结

本章中详细的介绍了仿真环境的搭建，其中包括模型的创建及生成，传感器的定义与模型体的连接，传感器的信息收集程序，模型的控制程序等。在本节中所验证的算法为上一章所介绍的改良虚拟势场方法。虚拟势场方法在二维环境中，已被多次验证，但在三维环境下被模拟比较少见，而本设计则展示了这一方法的有效性。

全文总结

本文详细的介绍了多功能模块化机器人控制系统的设计方案，整套系统共分为 3 部分。第一部分为在实际环境中进行任务执行的机械结构，其中大部分的设计继承了实验室前辈的设计方案，但对关键部位进行了改良。其中包括对每个子模块的机械结构进行了重设计，如重新设计了传感器支架和增加了轮式编码器。第二部分为对执行机构动力元件进行驱动的电路。电路按功能可以分为两类，第一类为所有模块都要使用的通用控制接口板；第二类为根据每个模块不同功能而选用的外设电路。其中通用接口板具有着控制的核心单元——单片机，和模块间通讯的关键接口——CAN 总线；而外设电路则都是根据具体功能而设计的，包括了定位用 IMU、编码器模块；对电机驱动的驱动模块；对传感器进行驱动的传感器接口模块以及要放置在连接处的模块间通讯接口实现模块。第三部分为控制算法部分，控制算法主要涉及四点，对底盘的驱动，对车体的定位，对障碍物的躲避，对所经历环境地图的绘制与最优路线的计算。这部分的每一个内容都是承上启下的。最后是对仿真过程进行了详细的介绍。

经过对系统三部分的研究设计出的控制系统整体，很好的满足了工业与探索应用领域对于模块化机器人的要求。本设计提供的控制方案是按功能进行划分的，只要合理规划，本方案中涉及的方法是很容易移植到其他的机器人及应用中去的。最后经过仿真的验证，证明了本设计提出的避障算法是实际可行的。

附录 A 仿真中所使用的机器人所处世界的代码

```
1 <sdf version='1.4'>
2   <world name='default'>
3     <light name='sun' type='directional'>
4       <cast_shadows>1</cast_shadows>
5       <pose>0.000000 0.000000 10.000000 0.000000 0.000000 0.000000</pose>
6       <diffuse>0.800000 0.800000 0.800000 1.000000</diffuse>
7       <specular>0.100000 0.100000 0.100000 1.000000</specular>
8       <attenuation>
9         <range>1000.000000</range>
10        <constant>0.900000</constant>
11        <linear>0.010000</linear>
12        <quadratic>0.001000</quadratic>
13     </attenuation>
14     <direction>-0.500000 0.500000 -1.000000</direction>
15   </light>
16   <model name='ground_plane'>
17     <static>1</static>
18     <link name='link'>
19       <collision name='collision'>
20         <geometry>
21           <plane>
22             <normal>0.000000 0.000000 1.000000</normal>
23             <size>100.000000 100.000000</size>
24           </plane>
25         </geometry>
26       <surface>
27         <friction>
28           <ode>
```

```
29      <mu>100.000000</mu>
30      <mu2>50.000000</mu2>
31      </ode>
32      </ friction >
33      <bounce/>
34      <contact>
35          <ode/>
36      </ contact >
37      </ surface >
38      <max_contacts>10</max_contacts>
39      </ collision >
40      <visual name='visual'>
41          <cast_shadows>0</cast_shadows>
42          <geometry>
43              <plane>
44                  <normal>0.000000 0.000000 1.000000</normal>
45                  <size>100.000000 100.000000</size>
46              </plane>
47          </geometry>
48          <material>
49              <script>
50                  <uri> file :// media/ materials / scripts /gazebo. material </uri>
51                  <name>Gazebo/Grey</name>
52              </ script >
53          </ material >
54      </ visual >
55      <velocity_decay>
56          <linear>0.000000</linear>
57          <angular>0.000000</angular>
58      </ velocity_decay >
59      <self_collide >0</ self_collide >
60      <kinematic>0</kinematic>
61      <gravity>1</gravity>
```

```
62      </link>
63    </model>
64
65      <physics type='ode'>
66        <max_step_size>0.001000</max_step_size>
67        <real_time_factor>1.000000</real_time_factor>
68        <real_time_update_rate>1000.000000</real_time_update_rate>
69        <gravity>0.000000 0.000000 -9.800000</gravity>
70        <max_contacts>20</max_contacts>
71
72      </physics>
73
74      <scene>
75        <ambient>0.200000 0.200000 0.200000 1.000000</ambient>
76        <background>0.700000 0.700000 0.700000 1.000000</background>
77        <shadows>1</shadows>
78
79      </scene>
80
81      <model name='Maze4robot'>
82        <link name='Wall_0'>
83          <collision name='Wall_0_Collision'>
84            <geometry>
85              <box>
86                <size>3.512000 0.200000 2.500000</size>
87              </box>
88            </geometry>
89            <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
90            <max_contacts>10</max_contacts>
91            <surface>
92              <bounce/>
93              <friction>
94                <ode/>
95              </friction>
96              <contact>
97                <ode/>
98              </contact>
99            </surface>
```

```
95      </ collision >
96      <visual name='Wall_0_Visual'>
97          <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
98          <geometry>
99              <box>
100                  <size>3.512000 0.200000 2.500000</size>
101          </box>
102      </geometry>
103      <material>
104          < script >
105              <uri> file :// media/ materials / scripts /gazebo. material </uri>
106              <name>Gazebo/Grey</name>
107          </ script >
108      </material>
109      </ visual >
110      <velocity_decay>
111          <linear>0.000000</linear>
112          <angular>0.000000</angular>
113      </velocity_decay>
114      <pose>-0.551000 -1.302500 0.000000 0.000000 0.000000 3.141590</pose>
115      < self_collide >0</ self_collide >
116      <kinematic>0</kinematic>
117      <gravity>1</gravity>
118  </link>
119  <link name='Wall_1'>
120      < collision name='Wall_1_Collision'>
121          <geometry>
122              <box>
123                  <size>1.453500 0.200000 2.500000</size>
124          </box>
125      </geometry>
126      <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
127      <max_contacts>10</max_contacts>
```

```
128      <surface>
129          <bounce/>
130          < friction >
131              <ode/>
132          </ friction >
133          <contact>
134              <ode/>
135          </ contact >
136      </ surface >
137  </ collision >
138  <visual name='Wall_1_Visual'>
139      <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
140      <geometry>
141          <box>
142              <size>1.453500 0.200000 2.500000</size>
143          </box>
144      </geometry>
145      <material>
146          < script >
147              <uri> file :// media/ materials / scripts /gazebo.material </uri>
148              <name>Gazebo/Grey</name>
149          </ script >
150      </ material >
151  </ visual >
152  <velocity_decay>
153      < linear >0.000000</linear>
154      < angular>0.000000</angular>
155  </ velocity_decay >
156  <pose>-2.207000 -1.929250 0.000000 0.000000 0.000000 -1.570800</pose>
157  < self_collide >0</ self_collide >
158  <kinematic>0</kinematic>
159  < gravity >1</ gravity >
160  </link>
```

```
161 <link name='Wall_10'>
162   < collision name='Wall_10_Collision'>
163     <geometry>
164       <box>
165         <size>4.572520 0.200000 2.500000</size>
166       </box>
167     </geometry>
168     <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
169     <max_contacts>10</max_contacts>
170     <surface>
171       <bounce/>
172       < friction >
173         <ode/>
174       </ friction >
175     <contact>
176       <ode/>
177     </ contact >
178   </ surface >
179 </ collision >
180 <visual name='Wall_10_Visual'>
181   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
182   <geometry>
183     <box>
184       <size>4.572520 0.200000 2.500000</size>
185     </box>
186   </geometry>
187   <material>
188     < script >
189       <uri> file :// media/ materials / scripts /gazebo.material </uri>
190       <name>Gazebo/Grey</name>
191     </ script >
192   </ material >
193 </ visual >
```

```
194 <velocity_decay>
195   <linear>0.000000</linear>
196   <angular>0.000000</angular>
197 </velocity_decay>
198 <pose>8.066370 -2.897990 0.000000 0.000000 0.000000 -1.570800</pose>
199 <self_collide>0</self_collide>
200 <kinematic>0</kinematic>
201 <gravity>1</gravity>
202 </link>
203 <link name='Wall_11'>
204   <collision name='Wall_11_Collision'>
205     <geometry>
206       <box>
207         <size>3.680520 0.200000 2.500000</size>
208       </box>
209     </geometry>
210   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
211   <max_contacts>10</max_contacts>
212   <surface>
213     <bounce/>
214     <friction>
215       <ode/>
216     </friction>
217     <contact>
218       <ode/>
219     </contact>
220   </surface>
221 </collision>
222 <visual name='Wall_11_Visual'>
223   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
224   <geometry>
225     <box>
226       <size>3.680520 0.200000 2.500000</size>
```

```
227      </box>
228      </geometry>
229      < material >
230          < script >
231              <uri> file :// media/ materials / scripts /gazebo.material </uri>
232              <name>Gazebo/Grey</name>
233          </ script >
234      </ material >
235      </ visual >
236      <velocity_decay>
237          < linear >0.000000</linear>
238          <angular>0.000000</angular>
239      </ velocity_decay >
240      <pose>6.326110 -5.084250 0.000000 0.000000 0.000000 3.141590</pose>
241      < self_collide >0</ self_collide >
242      <kinematic>0</kinematic>
243      <gravity>1</gravity>
244      </link >
245      <link name='Wall_13'>
246          < collision name='Wall_13_Collision'>
247              <geometry>
248                  <box>
249                      <size>1.818220 0.200000 2.500000</size>
250                  </box>
251              </geometry>
252              <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
253              <max_contacts>10</max_contacts>
254              <surface>
255                  <bounce/>
256                  < friction >
257                      <ode/>
258                  </ friction >
259              <contact>
```

```
260      <ode>
261      </contact>
262      </ surface >
263      </ collision >
264      <visual name='Wall_13_Visual'>
265          <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
266          <geometry>
267              <box>
268                  <size>1.818220 0.200000 2.500000</size>
269              </box>
270          </geometry>
271          <material>
272              < script >
273                  <uri> file :// media/ materials / scripts /gazebo. material </uri>
274                  <name>Gazebo/Grey</name>
275              </ script >
276          </material>
277      </ visual >
278      <velocity_decay>
279          < linear>0.000000</linear>
280          <angular>0.000000</angular>
281      </ velocity_decay >
282      <pose>-0.037374 -3.463730 0.000000 0.000000 0.000000 0.000000</pose>
283      < self_collide >0</ self_collide >
284      <kinematic>0</kinematic>
285      <gravity>1</gravity>
286      </link>
287      <link name='Wall_14'>
288          < collision name='Wall_14_Collision'>
289              <geometry>
290                  <box>
291                      <size>1.149830 0.200000 2.500000</size>
292              </box>
```

```
293      </geometry>
294      <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
295      <max_contacts>10</max_contacts>
296      <surface>
297          <bounce/>
298          < friction >
299              <oode/>
300          </ friction >
301      <contact>
302          <oode/>
303      </contact>
304      </ surface >
305      </ collision >
306      <visual name='Wall_14_Visual'>
307          <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
308          <geometry>
309              <box>
310                  <size>1.149830 0.200000 2.500000</size>
311          </box>
312      </geometry>
313      <material>
314          <script>
315              <uri> file :// media/ materials / scripts /gazebo.material </uri>
316              <name>Gazebo/Grey</name>
317          </ script >
318      </material>
319      </ visual >
320      <velocity_decay>
321          <linear>0.000000</linear>
322          <angular>0.000000</angular>
323      </ velocity_decay >
324      <pose>0.771737 -2.988820 0.000000 0.000000 0.000000 1.570800</pose>
325      < self_collide >0</ self_collide >
```

```
326 <kinematic>0</kinematic>
327 <gravity>1</ gravity >
328 </link >
329 <link name='Wall_16'>
330   < collision name='Wall_16_Collision'>
331     <geometry>
332       <box>
333         <size>2.416260 0.200000 2.500000</size>
334       </box>
335     </geometry>
336     <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
337     <max_contacts>10</max_contacts>
338     <surface>
339       <bounce/>
340       < friction >
341         <ode/>
342       </ friction >
343       <contact>
344         <ode/>
345       </ contact >
346     </ surface >
347   </ collision >
348   <visual name='Wall_16_Visual'>
349     <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
350     <geometry>
351       <box>
352         <size>2.416260 0.200000 2.500000</size>
353       </box>
354     </geometry>
355     <material>
356       < script >
357         <uri> file :// media/ materials / scripts /gazebo.material </uri>
358       <name>Gazebo/Grey</name>
```

```
359      </ script >
360      </ material >
361      </ visual >
362      < velocity_decay >
363          < linear >0.000000</ linear >
364          < angular>0.000000</ angular>
365      </ velocity_decay >
366      < pose>-2.183280 -4.642220 0.000000 0.000000 0.000000 -1.570800</ pose>
367      < self_collide >0</ self_collide >
368      < kinematic>0</ kinematic>
369      < gravity>1</ gravity >
370      </ link >
371      < link name='Wall_18'>
372          < collision name='Wall_18_Collision'>
373              < geometry>
374                  < box>
375                      < size>1.607150 0.200000 2.500000</ size>
376                  </ box>
377              </ geometry>
378              < pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</ pose>
379              < max_contacts>10</ max_contacts>
380              < surface>
381                  < bounce/>
382                  < friction >
383                      < code/>
384                  </ friction >
385                  < contact>
386                      < ode/>
387                  </ contact >
388              </ surface >
389          </ collision >
390          < visual name='Wall_18_Visual'>
391              < pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</ pose>
```

```
392 <geometry>
393   <box>
394     <size>1.607150 0.200000 2.500000</size>
395   </box>
396 </geometry>
397 < material >
398   < script >
399     <uri> file :// media/ materials / scripts /gazebo.material </uri>
400     <name>Gazebo/Grey</name>
401   </ script >
402 </ material >
403 </ visual >
404 <velocity_decay>
405   < linear>0.000000</linear>
406   <angular>0.000000</angular>
407 </ velocity_decay >
408 <pose>-1.409350 -4.835700 0.000000 0.000000 0.000000 0.000000</pose>
409 < self_collide >0</ self_collide >
410 <kinematic>0</kinematic>
411 <gravity>1</gravity>
412 </link>
413 <link name='Wall_2'>
414   < collision name='Wall_2_Collision'>
415     <geometry>
416       <box>
417         <size>1.005000 0.200000 2.500000</size>
418       </box>
419     </geometry>
420   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
421   <max_contacts>10</max_contacts>
422   <surface>
423     <bounce/>
424     < friction >
```

```
425      <ode>
426      </ friction >
427      <contact>
428          <ode>
429      </ contact >
430      </ surface >
431      </ collision >
432      <visual name='Wall_2_Visual'>
433          <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
434          <geometry>
435              <box>
436                  <size>1.005000 0.200000 2.500000</size>
437              </box>
438          </geometry>
439          <material>
440              < script >
441                  <uri> file :// media/ materials / scripts /gazebo. material </uri>
442                  <name>Gazebo/Grey</name>
443              </ script >
444          </material>
445      </ visual >
446      <velocity_decay>
447          <linear>0.000000</linear>
448          <angular>0.000000</angular>
449      </ velocity_decay >
450      <pose>-1.804500 -2.556000 0.000000 0.000000 0.000000 0.000000</pose>
451      < self_collide >0</ self_collide >
452      <kinematic>0</kinematic>
453      <gravity>1</gravity>
454      </link>
455      <link name='Wall_20'>
456          < collision name='Wall_20_Collision'>
457              <geometry>
```

```
458      <box>
459          <size>2.240370 0.200000 2.500000</size>
460      </box>
461  </geometry>
462  <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
463  <max_contacts>10</max_contacts>
464  <surface>
465      <bounce/>
466      < friction >
467          <ode/>
468      </ friction >
469      <contact>
470          <ode/>
471      </ contact >
472      </ surface >
473  </ collision >
474  <visual name='Wall_20_Visual'>
475      <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
476  <geometry>
477      <box>
478          <size>2.240370 0.200000 2.500000</size>
479      </box>
480  </geometry>
481  <material>
482      <script>
483          <uri> file :// media/ materials / scripts /gazebo.material </uri>
484          <name>Gazebo/Grey</name>
485      </ script >
486  </ material >
487  </ visual >
488  <velocity_decay>
489      < linear >0.000000</linear>
490      < angular>0.000000</angular>
```

```
491      </ velocity_decay >
492      <pose>0.560665 -5.891060 0.000000 0.000000 0.000000 1.570800</pose>
493      < self_collide >0</ self_collide >
494      <kinematic>0</kinematic>
495      <gravity>1</gravity>
496      </link>
497      <link name='Wall_21'>
498          < collision name='Wall_21_Collision'>
499              <geometry>
500                  <box>
501                      <size>2.345910 0.200000 2.500000</size>
502                  </box>
503              </geometry>
504              <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
505              <max_contacts>10</max_contacts>
506              <surface>
507                  <bounce/>
508                  < friction >
509                      <ode/>
510                  </ friction >
511                  <contact>
512                      <ode/>
513                  </contact>
514              </surface>
515          </ collision >
516          <visual name='Wall_21_Visual'>
517              <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
518              <geometry>
519                  <box>
520                      <size>2.345910 0.200000 2.500000</size>
521                  </box>
522              </geometry>
523          <material>
```

```
524 < script >
525   <uri> file :// media/ materials / scripts /gazebo. material </uri>
526   <name>Gazebo/Grey</name>
527   </ script >
528   </ material >
529   </ visual >
530   <velocity_decay>
531     <linear>0.000000</linear>
532     <angular>0.000000</angular>
533   </ velocity_decay >
534   <pose>1.633620 -4.870880 0.000000 0.000000 0.000000 0.000000</pose>
535   < self_collide >0</ self_collide >
536   <kinematic>0</kinematic>
537   <gravity>1</gravity>
538   </link >
539   <link name='Wall_22'>
540     < collision name='Wall_22_Collision'>
541       <geometry>
542         <box>
543           <size>2.662510 0.200000 2.500000</size>
544         </box>
545       </geometry>
546       <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
547       <max_contacts>10</max_contacts>
548       <surface>
549         <bounce/>
550         < friction >
551           <ode/>
552         </ friction >
553         <contact>
554           <ode/>
555         </ contact >
556       </ surface >
```

```
557     </ collision >
558
559     <visual name='Wall_22_Visual'>
560         <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
561         <geometry>
562             <box>
563                 <size>2.662510 0.200000 2.500000</size>
564             </box>
565         </geometry>
566         <material>
567             <script>
568                 <uri> file :// media/ materials / scripts /gazebo. material </uri>
569                 <name>Gazebo/Grey</name>
570             </script>
571         </material>
572     </visual >
573     <velocity_decay>
574         <linear>0.000000</linear>
575         <angular>0.000000</angular>
576     </velocity_decay >
577     <pose>2.706570 -6.102140 0.000000 0.000000 0.000000 -1.570800</pose>
578     <self_collide >0</self_collide >
579     <kinematic>0</kinematic>
580     <gravity>1</gravity >
581     </link >
582     <link name='Wall_23'>
583         <collision name='Wall_23_Collision'>
584             <geometry>
585                 <box>
586                     <size>3.788230 0.200000 2.500000</size>
587                 </box>
588             </geometry>
589             <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
590             <max_contacts>10</max_contacts>
```

```
590      <surface>
591          <bounce/>
592          < friction >
593              <oode/>
594          </ friction >
595          <contact>
596              <oode/>
597          </ contact >
598      </ surface >
599  </ collision >
600  <visual name='Wall_23_Visual'>
601      <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
602      <geometry>
603          <box>
604              <size>3.788230 0.200000 2.500000</size>
605          </box>
606      </geometry>
607      <material>
608          < script >
609              <uri> file :// media/ materials / scripts /gazebo. material </uri >
610              <name>Gazebo/Grey</name>
611          </ script >
612      </ material >
613  </ visual >
614  <velocity_decay>
615      < linear >0.000000</linear >
616      < angular >0.000000</angular>
617  </ velocity_decay >
618  <pose>4.500690 -7.333390 0.000000 0.000000 0.000000 0.000000</pose>
619  < self_collide >0</ self_collide >
620  <kinematic>0</kinematic>
621  < gravity >1</ gravity >
622  </link >
```

```
623 <link name='Wall_24'>
624   < collision name='Wall_24_Collision'>
625     <geometry>
626       <box>
627         <size>1.396080 0.200000 2.500000</size>
628       </box>
629     </geometry>
630     <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
631     <max_contacts>10</max_contacts>
632     <surface>
633       <bounce/>
634       < friction >
635         <ode/>
636       </ friction >
637     <contact>
638       <ode/>
639     </ contact >
640   </ surface >
641 </ collision >
642 <visual name='Wall_24_Visual'>
643   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
644   <geometry>
645     <box>
646       <size>1.396080 0.200000 2.500000</size>
647     </box>
648   </geometry>
649   <material>
650     < script >
651       <uri> file :// media/ materials / scripts /gazebo.material </uri>
652       <name>Gazebo/Grey</name>
653     </ script >
654   </ material >
655 </ visual >
```

```
656 <velocity_decay>
657   <linear>0.000000</linear>
658   <angular>0.000000</angular>
659 </velocity_decay>
660 <pose>6.294800 -6.735350 0.000000 0.000000 0.000000 1.570800</pose>
661 <self_collide>0</self_collide>
662 <kinematic>0</kinematic>
663 <gravity>1</gravity>
664 </link>
665 <link name='Wall_4'>
666   <collision name='Wall_4_Collision'>
667     <geometry>
668       <box>
669         <size>2.788530 0.200000 2.500000</size>
670       </box>
671     </geometry>
672     <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
673     <max_contacts>10</max_contacts>
674     <surface>
675       <bounce/>
676       <friction>
677         <ode/>
678       </friction>
679       <contact>
680         <ode/>
681       </contact>
682     </surface>
683   </collision>
684   <visual name='Wall_4_Visual'>
685     <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
686     <geometry>
687       <box>
688         <size>2.788530 0.200000 2.500000</size>
```

```
689      </box>
690      </geometry>
691      <material>
692          <script>
693              <uri> file :// media/ materials / scripts /gazebo.material </uri>
694              <name>Gazebo/Grey</name>
695          </script>
696      </material>
697      </visual>
698      <velocity_decay>
699          <linear>0.000000</linear>
700          <angular>0.000000</angular>
701      </velocity_decay>
702      <pose>2.417080 -2.600660 0.000000 0.000000 0.000000 -1.570800</pose>
703      <self_collide>0</self_collide>
704      <kinematic>0</kinematic>
705      <gravity>1</gravity>
706      </link>
707      <link name='Wall_5'>
708          <collision name='Wall_5_Collision'>
709              <geometry>
710                  <box>
711                      <size>2.106420 0.200000 2.500000</size>
712                  </box>
713              </geometry>
714              <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
715              <max_contacts>10</max_contacts>
716              <surface>
717                  <bounce/>
718                  <friction>
719                      <ode/>
720                  </friction>
721              <contact>
```

```
722      <ode>
723          </contact>
724          </surface>
725      </ collision >
726      <visual name='Wall_5_Visual'>
727          <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
728          <geometry>
729              <box>
730                  <size>2.106420 0.200000 2.500000</size>
731          </box>
732      </geometry>
733      <material>
734          < script >
735              <uri> file :// media/ materials / scripts /gazebo. material </uri>
736              <name>Gazebo/Grey</name>
737          </ script >
738      </material>
739      </ visual >
740      <velocity_decay>
741          <linear>0.000000</linear>
742          <angular>0.000000</angular>
743      </velocity_decay>
744      <pose>3.370280 -3.894930 0.000000 0.000000 0.000000 0.000000</pose>
745      < self_collide >0</ self_collide >
746      <kinematic>0</kinematic>
747      <gravity>1</gravity>
748      </link>
749      <link name='Wall_7'>
750          < collision name='Wall_7_Collision'>
751              <geometry>
752                  <box>
753                      <size>3.558090 0.200000 2.500000</size>
754          </box>
```

```
755 </geometry>
756 <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
757 <max_contacts>10</max_contacts>
758 <surface>
759   <bounce/>
760   <friction>
761     <oode/>
762   </friction>
763   <contact>
764     <oode/>
765   </contact>
766   </surface>
767 </collision>
768 <visual name='Wall_7_Visual'>
769   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
770   <geometry>
771     <box>
772       <size>3.558090 0.200000 2.500000</size>
773     </box>
774   </geometry>
775   <material>
776     <script>
777       <uri> file :// media/ materials / scripts /gazebo.material </uri>
778       <name>Gazebo/Grey</name>
779     </script>
780   </material>
781 </visual>
782 <velocity_decay>
783   <linear>0.000000</linear>
784   <angular>0.000000</angular>
785 </velocity_decay>
786 <pose>5.285450 -2.303330 0.000000 0.000000 0.000000 0.000000</pose>
787 <self_collide>0</self_collide>
```

```
788 <kinematic>0</kinematic>
789 <gravity>1</ gravity>
790 </link>
791 <link name='Wall_8'>
792   < collision name='Wall_8_Collision'>
793     <geometry>
794       <box>
795         <size>1.791600 0.200000 2.500000</size>
796       </box>
797     </geometry>
798   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
799   <max_contacts>10</max_contacts>
800   <surface>
801     <bounce/>
802     < friction >
803       <ode/>
804     </ friction >
805     <contact>
806       <ode/>
807     </ contact >
808   </ surface >
809 </ collision >
810 <visual name='Wall_8_Visual'>
811   <pose>0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</pose>
812   <geometry>
813     <box>
814       <size>1.791600 0.200000 2.500000</size>
815     </box>
816   </geometry>
817   <material>
818     < script >
819       <uri> file :// media/ materials / scripts /gazebo.material </uri>
820     <name>Gazebo/Grey</name>
```

```
821      </ script >
822      </ material >
823      </ visual >
824      < velocity_decay >
825          < linear >0.000000</ linear >
826          < angular>0.000000</ angular>
827      </ velocity_decay >
828      < pose>6.964490 -1.507540 0.000000 0.000000 0.000000 1.570800</ pose>
829      < self_collide >0</ self_collide >
830      < kinematic>0</kinematic>
831      < gravity>1</ gravity >
832      </ link >
833      < link name='Wall_9' >
834          < collision name='Wall_9_Collision' >
835              < geometry >
836                  < box >
837                      < size >1.301870 0.200000 2.500000</ size >
838                  </ box >
839              </ geometry >
840              < pose >0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</ pose >
841              < max_contacts >10</ max_contacts >
842              < surface >
843                  < bounce />
844                  < friction >
845                      < code />
846                  </ friction >
847                  < contact >
848                      < ode />
849                  </ contact >
850              </ surface >
851          </ collision >
852          < visual name='Wall_9_Visual' >
853              < pose >0.000000 0.000000 1.250000 0.000000 0.000000 0.000000</ pose >
```

```
854     <geometry>
855         <box>
856             <size>1.301870 0.200000 2.500000</size>
857         </box>
858     </geometry>
859     < material >
860         < script >
861             <uri> file :// media/ materials / scripts /gazebo.material </uri>
862             <name>Gazebo/Grey</name>
863         </ script >
864     </ material >
865     </ visual >
866     <velocity_decay>
867         < linear>0.000000</linear>
868         <angular>0.000000</angular>
869     </ velocity_decay >
870     <pose>7.515430 -0.711737 0.000000 0.000000 0.000000 0.000000</pose>
871     < self_collide >0</ self_collide >
872     <kinematic>0</kinematic>
873     <gravity>1</gravity>
874     </link>
875     < static >1</ static >
876   </model>
877   < state  world_name='default'>
878       <sim_time>0 0</sim_time>
879       <real_time>0 71587</real_time>
880       <wall_time>1370763498 50562385</wall_time>
881   </ state >
882   <gui  fullscreen ='0'>
883     <camera name='user_camera'>
884         <pose>11.832000 -12.190100 10.122000 0.000000 0.679643 2.336200</pose>
885         < view_controller > orbit </ view_controller >
886     </camera>
```

887 </gui>

888 </world>

889 </sdf>

参考文献

- [1] DUDEK G, JENKIN M. Computational principles of mobile robotics[M]. Cambridge: Cambridge university press, 2010.
- [2] HOLLAND O. Grey Walter: the pioneer of real artificial life[C]//Proceedings of the 5th international workshop on artificial life. .[S.l.]: [s.n.] , 1997:34–44.
- [3] HARRISON R, WESTERN R, MOORE P, et al. A study of application areas for modular robots[J]. Robotica, 1987, 5(03):217–221.
- [4] 王鹏飞, 秦小云. 可重构模块化机器人的研究 [J]. 中国水运, 2007, 7(7):164–165.
- [5] BENHABIB B, DAI M. Mechanical design of a modular robot for industrial applications[J]. Journal of Manufacturing Systems, 1991, 10(4):297–306.
- [6] PAREDIS C J, KHOSLA P K. Kinematic design of serial link manipulators from task specifications[J]. The International Journal of Robotics Research, 1993, 12(3):274–287.
- [7] FUJITA M, KITANO H, KAGEYAMA K. A reconfigurable robot platform[J]. Robotics and autonomous Systems, 1999, 29(2):119–132.
- [8] YIM M. New locomotion gaits[C]//Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on. .[S.l.]: [s.n.] , 1994:2508–2514.
- [9] 李磊, 叶涛, 谭民, et al. 移动机器人技术研究现状与未来 Ξ[J]. 机器人, 2002, 24(9):475–480.

- [10] NILSSON N. A Mobile Automation: An Application of Artificial Intelligence Techniques[J]. Autonomous Mobile Robots: Control, Planning, and Architecture, 1969, 2:233–244.
- [11] WANG T K, DANG Q, PAN P Y. Path planning approach in unknown environment[J]. International Journal of Automation and Computing, 2010, 7(3):310–316.
- [12] ERSSON T, HU X. Path planning and navigation of mobile robots in unknown environments[C]//Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on. .[S.l.]: [s.n.] , 2001 , 2:858–864.
- [13] UTSUGI S, SUZUKI H. Path planning in an unknown environment on the basis of observations of occluded areas[C]//Cybernetics and Intelligent Systems, 2008 IEEE Conference on. Tokoyo: [s.n.] , 2008:248–253.
- [14] RYLAND G G, CHENG H H. Design of iMobot, an intelligent reconfigurable mobile robot with novel locomotion[C]//Robotics and Automation (ICRA), 2010 IEEE International Conference on. .[S.l.]: [s.n.] , 2010:60–65.
- [15] 谢宁, 周翔, 刘露露, 等. 基于 XS128 单片机的智能循迹车硬件系统设计 [J]. 国外电子测量技术, 2012, 31(12):63–66.
- [16] BOSCH R. CAN specification version 2.0[J]. Rober Bousch GmbH, Postfach, 1991, 300240.
- [17] ZHANG Y, ROUFAS K, YIM M, et al. Massively Distributed Control Nets for Modular Reconfigurable Robots[C]//2002 AAAI Spring Symposium on Intelligent Distributed and Embedded Systems. .[S.l.]: [s.n.] , 2002.

- [18] MINOR R R, ROWE D W. Utilization of a magnetic sensor to compensate a MEMS-IMU/GPS and de-spin strapdown on rolling missiles[J]. 2001. US Patent 6,208,936.
- [19] HOUTEKAMER P L, MITCHELL H L. Data assimilation using an ensemble Kalman filter technique[J]. Monthly Weather Review, 1998, 126(3):796–811.
- [20] KHATIB O. Real-time obstacle avoidance for manipulators and mobile robots[J]. The international journal of robotics research, 1986, 5(1):90–98.
- [21] KOREN Y, BORENSTEIN J. Potential field methods and their inherent limitations for mobile robot navigation[C]//Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on. .[S.l.]: [s.n.] , 1991:1398–1404.
- [22] 耿兆丰, 吴永敢. 基于势场的运动路径规划 [J]. 机器人, 1992, 14(5):38–43.
- [23] WALLER D, HUNT E, KNAPP D. The transfer of spatial knowledge in virtual environment training[J]. Presence, 1998, 7(2):129–143.
- [24] 周斌, 刘旺, 林辛凡, et al. 智能车赛道记忆算法的研究 [J]. 电子产品世界, 2006(08S):160–161.

致 谢

本设计论文的完成离不开费燕琼老师的帮助与指导，特此表示感谢。在电路的设计上，电院自动化系王冰老师的设计给了我许多启发，同时也得到过工训中心宋立博老师的意见，在此一并感谢。关于超声波传感器的选用来自于几次与陆正辰学长的谈话，对其能给我提供这么好的建议，同样表示感谢。在实验室进行项目期间，得到了吕海洋学长的大力支持与帮助，对此我表示由衷的谢意。关于 GAZEBO 的使用要感谢我在外国实验室的 PhD 学生 Nick，是他将这么好的动力学仿真软件介绍给我的。还要感谢国外的 Finie 老师，使我第一次接触到了 IMU 模块，以及我的国外的导师 Mark，是他在 CAN 总线通讯上给了我启发。最后，感谢所有在设计当中给我提供意见与帮助的老师同学！

DESIGN OF A MULTIFUNCTIONAL MOBILE MODULAR ROBOT SYSTEM

Robot system has drawn more attentions since the 1960s, with the developing of control theory, sensor manufacturing and computer capability. However, applications of robot have evolved, so robots get more and more complex to maintain the necessary functionality. The complexity of robots makes it more difficult to upgrade, and for some specific application, it may not need all the functions of a robot, then there will be a great waste of resource. Therefore, the concept of modular robot has been introduced, to divide the functions into different independent parts which can be easily combined and upgraded.

Modularization is not the only requirement of robot developing, to make a robot move is another crucial research area of the study of robots. A mobile robot not only can do a task but can also do that task in multiple positions, which can save a lot of money on buying new machines. And in some exploring applications, a mobile robot can replace human beings to get into unknown and dangerous area to help people finish the exploring tasks.

Since modularization and mobility are two important parts of robot developing, then the combination of these parts will result a birth of a new kind of very useful robots. In this paper, the author will illustrate a design of the control system of this new kind of robots.

According to the understanding of the author, this system should contain three

crucial parts. The first is the mechanism of the robot body, which is the part interacting with external environment and containing task actuators. The second part is the circuits which will be used to drive actuators. The last part is the software which provide functions and rules to control the behaviour of the robots. The control system need all these three parts to work together to achieve the functions of this newly designed robot. Details of the design of these three parts provided in the paper, and followings are some short reviews.

For the mechanism of the robot system, the paper provides several new concepts and design of the robot body based on an old design from the author's lab. The old design contains a central control unit to control all the modules of the robot. It has been proved not efficient because with the increasing of modules, the extended ports have to be redesign and the central control unit has to be reprogrammed. However, if each module is self-controllable, then we will get rid of the trouble of reprogramme. The connection port we will use is also fixed, which would only be used for communication between different modules. Based on this kind of idea, in this paper, it provided a new design of of module body, sensor stand and communication connection port. A new design of wheel encoder will also be provide in this section.

For the circuit part, this paper introduce a new type of structure of circuit combination. Because all the modules are self-controllable, each module has to have a central control unit. However, different modules have different functions, then we have to operate functional circuit from the main control board. In the paper, the author propose a main control board plus peripherals structure to form the control circuit of each module. The main control board only provide the power management unit and module level communication circuit. All the special functions will be provided by the peripherals. The peripherals designed in this paper includes IMU unit, Motor driver circuit,

encoder support circuit, sensor driver circuit and a communication port circuit for the connection part.

For the software part, the paper only discuss the software which is need for self-navigation in unknown environment. The low level programme is the driver programme for the chassis. In this part, it contains the initialization of the related register on MCU, the PID controller that make the moving direction of the robot along with the direction that has been calculated and the free turn methods. The mid-level programme is the programme for localization. In the paper, it provides three ways to localize the robot position. The high level programme is the programme that controls the robot moving to the target point avoiding all the obstacles on the path. In addition, this paper also talked about a way to record the detected environment and calculate the optimal path.

After design of all those three parts, the author provide a simulation of the algorithm in a simulation environment called GAZEBO. In the paper, it provides enough details about the simulation process and get the desired result which can proofed the validity of the algorithm proposed in previous section.