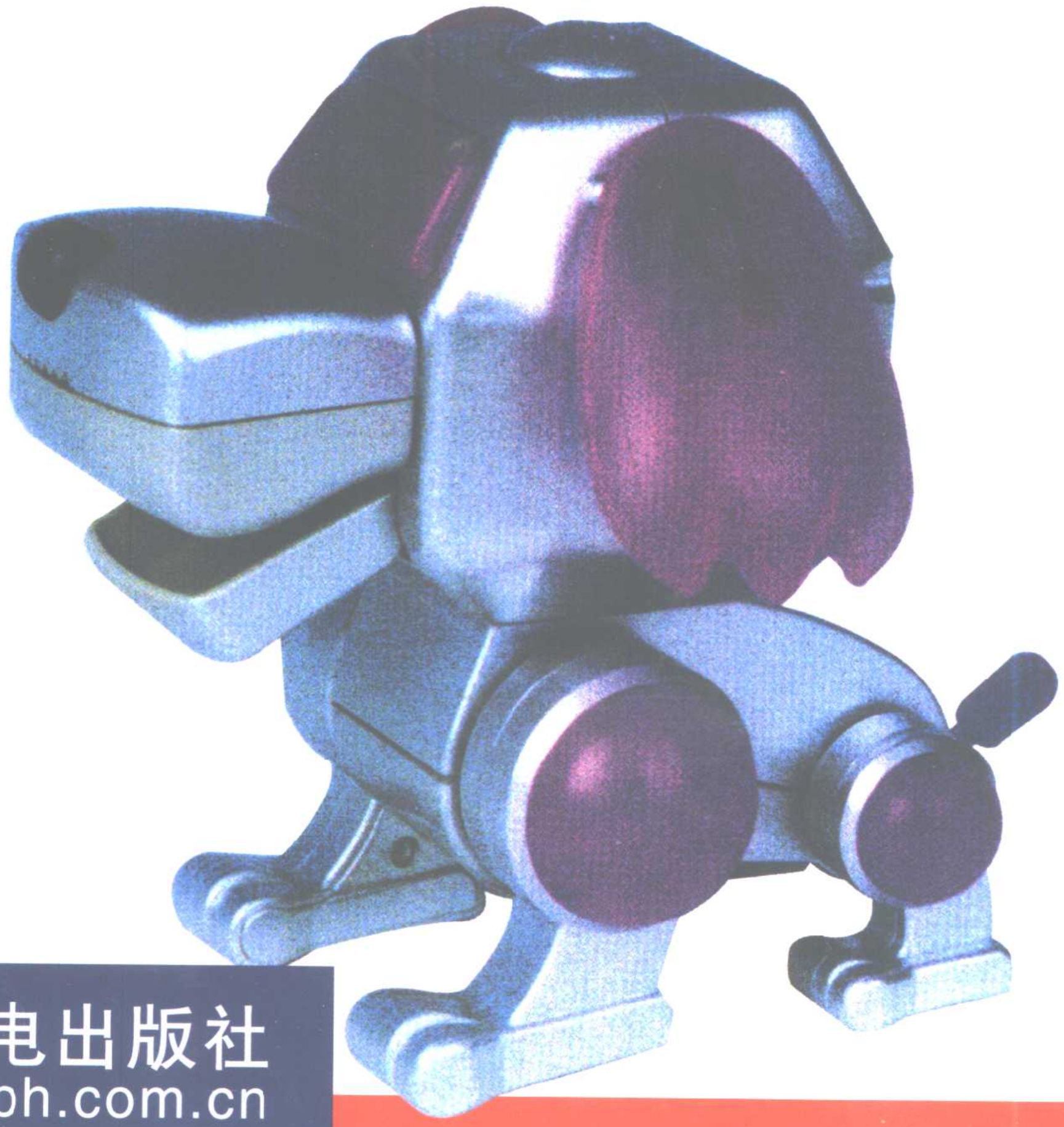
全国中小学
电脑制作活动指导丛书

# 智能机器人制作入门

教育部基础教育课程教材发展中心　组织编写
徐爱平　沙有威　主编

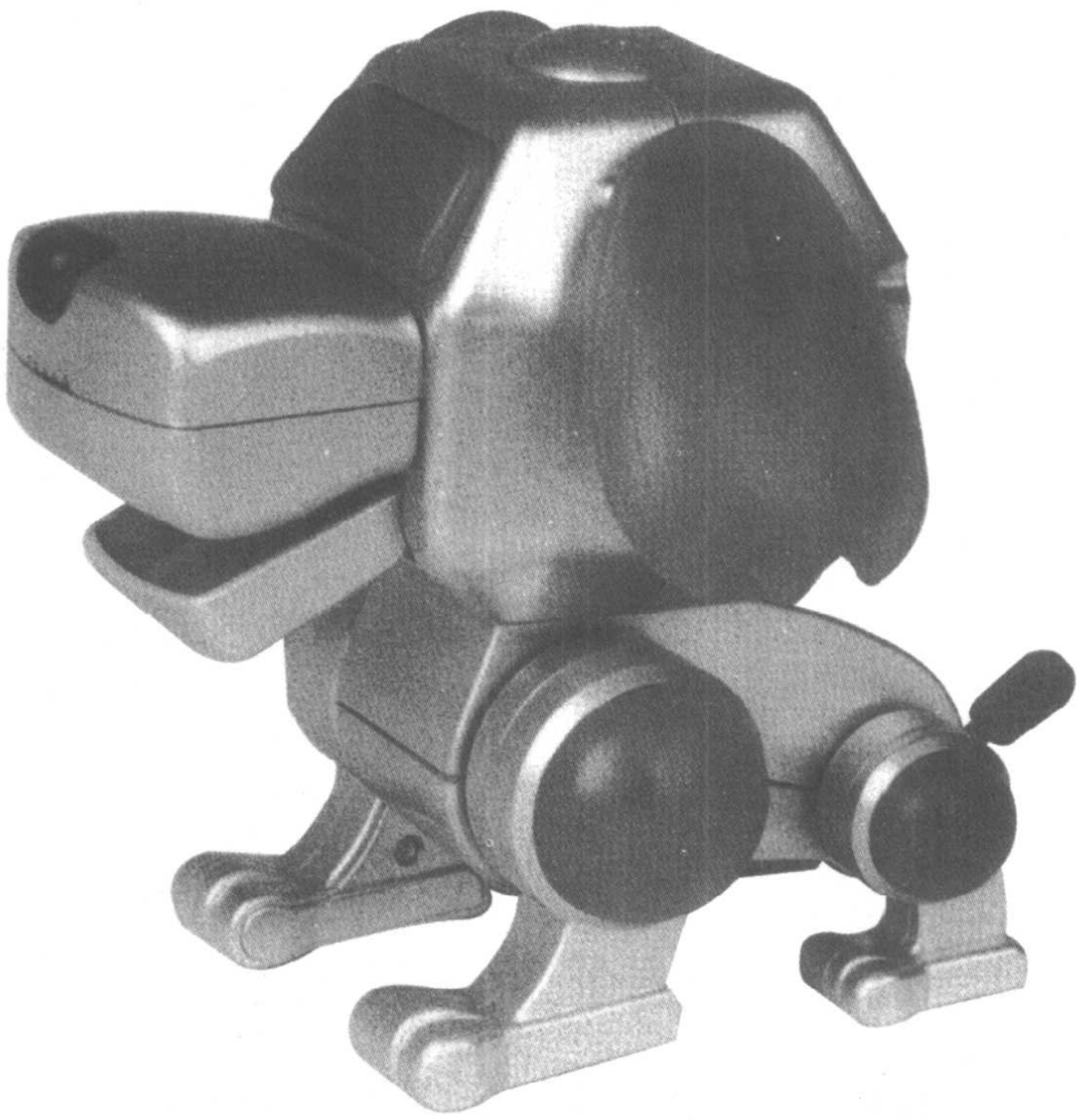


人民邮电出版社
www.pptph.com.cn

全国中小学
电脑制作活动指导丛书

# 智能机器人制作入门

教育部基础教育课程教材发展中心　组织编写
徐爱平　沙有威　主编



人民邮电出版社

**图书在版编目(CIP)数据**

智能机器人制作入门/徐爱平,沙有威主编.—北京:人民邮电出版社,2001.7

(全国中小学电脑制作活动指导丛书)

ISBN 7-115-09365-2

Ⅰ.智… Ⅱ.①徐…②沙… Ⅲ.智能机器人-青少年读物 Ⅳ.TP242.6-49

中国版本图书馆 CIP 数据核字(2001)第 044124 号

全国中小学电脑制作活动指导丛书

**智能机器人制作入门**

◆ 教育部基础教育课程教材发展中心 组织编写

主　　编　徐爱平　沙有威

责任编辑　苏　欣

执行编辑　贾鸿飞

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号

邮编　100061　　电子函件　315@ pptph.com.cn

网址　http://www.pptph.com.cn

读者热线:010-67129212　010-67129211(传真)

北京汉魂图文设计有限公司制作

北京顺义向阳胶印厂印刷

新华书店总店北京发行所经销

◆ 开本:720×980　1/16

印张:10.25

字数:192 千字　　2001 年 7 月第 1 版

印数:1-5 000 册　　2001 年 7 月北京第 1 次印刷

ISBN 7-115-09365-2/TP·2257

定价:24.00 元

# 内容提要

本书介绍了有关智能机器人的知识和制作技术，并结合能力风暴个人机器人的操作方法，对智能机器人的基本结构作了介绍。全书共分 4 章：第 1 章介绍了能力风暴个人机器人的基本结构和操作方法；第 2 章逐一讲述了能力风暴个人机器人的传感器和执行器，并介绍了其控制命令的使用方法；第 3 章总结了 JC 及其程序的基本结构，结合 7 项任务提高机器人的智能程度；第 4 章结合机器人比赛和项目提出一些思路，让学习者通过实践去完成。

本书属于科普型读物，突出教与学互动的特点，适合广大青少年及对机器人制作有兴趣的读者阅读。也适于开展校外活动时使用。

内容提要

# 《全国中小学电脑制作活动指导丛书》
# 编 委 会

# 序

《全国中小学电脑制作活动指导丛书》与全国广大中小学生和教师见面了。首先要对参与这套丛书编写工作的专家学者表示感谢。

2000年，教育部组织了“首届全国中小学电脑制作与设计作品制作活动”并评选出现优秀的电脑作品。这是一次“普及中小学信息技术教育、激发中小学生创新精神”的活动。在全国各省、自治区、直辖市教育部门大力支持下，得到了全国各地中小学生的积极响应，首届全国中小学电脑作品制作活动及优秀作品评选取得了圆满成功。

“全国中小学电脑制作与设计作品活动”的内容和形式，比较符合广大中小学生热爱科学、追求新异、喜欢探索、崇尚个性的心理特点，也深受广大中小学生的欢迎和喜爱。2001年，这一“活动”更名为“全国中小学生电脑制作活动”，并被列入“全国青少年科技活动周”的主要活动内容之一。参与这一活动的中小学生可以运用所掌握的计算机知识和实践能力，将自己对学习的探讨、对社会生活的感受、对祖国以及家乡的热爱、对社会时尚的关注，融进亲手制作的电脑作品并展现给社会。这是一件值得中小学生引以为荣的活动。

目前，计算机技术的成熟、国际互联网的出现，将人类（包括古今中外）发展过程中的智慧，汇聚到一个能够覆盖全球的计算机网络系统中。这不仅延伸了人类个体的大脑和思维活动，而且创造了一个外化的、每时每刻都在急剧发展的全人类“大脑”。崭新的信息化和数字化的新环境给人类的生存、生活、生产、学习搭建了一个与我们的习惯完全不同的虚拟空间，构成了一个更加开放、平等、自由的人类及社会发展的大平台。人们现在普遍认为，人类已经进入计算机和网络时

代。虽然把计算机和网络技术赋予“时代”的概念，但其极为丰富和深奥的内涵还需要人类进一步去理解和认识。

在跨入21世纪的时候，计算机和网络的确对人类的生活产生了巨大的影响，这一影响还将随着科学技术的迅猛发展而更加明显。

“教育要面向现代化、面向世界、面向未来”，邓小平同志的精辟论断为中小学信息技术教育指明了方向。“用信息化推进教育现代化”，教育部长陈至立同志的重要指示为中小学信息技术教育确立了目标。为了适应21世纪人类社会的竞争和挑战，使我们中华民族有足够的能力屹立于世界，一项重要的基础性工作，就是使全国亿万中小学生更加普遍和深入地接受信息技术教育：“计算机的普及要从娃娃做起”。

为了进一步在中小学普及信息技术教育，丰富中小学生的学习生活，教育部将继续组织“全国中小学生电脑制作活动”，其主要目的仍然是激发学生的创新精神，培养实践能力，在中小学全面推进素质教育。我们期望，本套丛书的出版能为全国各地中小学组织开展学生电脑制作活动提供指导，为各地中小学开设信息技术课程提供一个实践园地。我们期待，“全国中小学生电脑制作活动”更加丰富、更加成功。

王晓芜

# 前言

机器人技术是20世纪人类最伟大的发明之一。“机器人学的进步和应用是20世纪自动控制最有说服力的成就，是当代最高意义的自动化。”这是宋健院士对机器人技术所取得的成就的精辟概括，他预计21世纪“自动化技术仍将是高技术前沿，继续是推进新技术革命的核心力量。制造业和服务业仍然是它取得辉煌成就的主要领域。”可以说机器人技术的诞生和发展是社会和经济发展的必然结果，是高新技术发展的必然结果。

机器人技术的发展与信息技术的发展有着十分重要的关系。信息技术是当前高技术发展中的主流技术，它的发展对其他技术将产生极大的影响，特别是计算机技术的发展，会使许多系统和设备具有某种“智能”作用和功能，而机器人就是这些“智能”系统和设备的代表。同时机器人技术的提高也会对信息技术的发展产生推动作用。当前网络机器人、软件机器人等交互技术和新概念的产生，使机器人技术与信息技术的结合成为发展的必然趋势。

机器人技术涉及了多门学科，是一个国家科技发展水平和国民经济现代化、信息化的重要标志。因此，机器人技术是世界强国重点发展的高技术之一。为了在广大青少年当中开展智能机器人知识和技术的普及教育，促进机器人技术的发展和人才的培养；为了大力推进素质教育，探索教育的新途径、新方法；也为了培养青少年的创新精神和实践能力，展示青少年的聪明才智，我们以能力风暴个人机器人为平台，编写了这本面向广大青少年的科普读物——《智能机器人制作入门》。

能力风暴个人机器人是为培养广大青少年的创造能力和协作能力而专门推出的开放式平台，它融合了现代工业设计、机械、电子、传感器、计算机和人工智能等诸多领域的先进技术，功能强大，应用面广。广大青少年通过使用能力风暴个人机器人，可以接触到多方面的知识和技术，并在自主活动中学会钻研问题，解决问题。智能机器人独特的吸引力，容易使学生产生浓厚的兴趣，并通过自己动手装配、实验、编程和实施机器人项目、参加机器人比赛，直至设计出自己独特的机器人伙伴。能力风暴个人机器人将引导广大青少年进入激动人心的科技前沿领域。

人类正在步入一个以“智力资源的占有、配置，知识的生产、分配、使用（消费）为最重要因素的经济时代”。站在新世纪的起点，我们要在探索与建构自主性创新教育和教学体系的过程中，加快培养具有创新精神和创新能力的高素质人才，为我国机器人技术的发展培养后备力量，为我国高新技术的发展培养后备力量。

本书在编写过程中得到国家高新技术智能机器人专家组、《机器人技术与应用》杂志社和上海广茂达电子信息有限公司的技术支持提供资料。参与本书编写的还有《机器人技术与应用》杂志社主编刘进长和编辑丁俊武，在此一并表示诚挚的感谢。

由于作者水平有限，编写时间仓促，书中难免有不妥之处，望广大读者批评指正。

编　　者

# 目 录



目 录

# 第1章 了解能力风暴个人机器人

能力风暴个人机器人是具有感知能力、决策能力和动作能力的智能型机器人，这个智能机器人是我国自行设计、开发的，是为培养广大青少年动手能力、创造能力和协作能力而专门推出的开放式智能机器人教学平台。在这个直径为 20cm 的小平台上，读者可以学习和接触机械、电子、电机、传感器、计算机软硬件、机器人学和人工智能等多方面的知识，并在参加比赛和完成各种应用项目的过程中提高自己的综合能力。通过本章内容的学习，我们将对能力风暴个人机器人有一个基本的了解，掌握一些基本的操作方法。

## 1.1 认识能力风暴个人机器人

现在我们就一同走进智能机器人世界，看看能力风暴个人机器人将带给我们哪些意外和惊喜。

在你的想象中，机器人和人类有着相似的外形，身上装着笨重的机械装置，1m 多高的大个子，拖着一条长长的粗电线，全身由“金属”武装。但是当你见到能力风暴个人机器人时，你会为它的小巧和运动的灵活感到意外，同时你会产生疑问：这么一个小东西就是机器人吗？在进一步了解能力风暴个人机器人之后，你一定会为它的交互能力和“生命”的活力感到惊奇和不可思议，并且会觉得这个“小不点儿”简直就是一个“小精灵”。

现在我们一起来看看能力风暴个人机器人的各项性能，了解它的特点以及在教学中的作用。能力风暴个人机器人外观如图 1.1 所示。

### 1.1.1 性能指标

能力风暴个人机器人有着特殊的“体形”，“说”着独特的语言，下面我们通过对各项性能指标的介绍，使你对能力风暴个人机器人有一个初步的印象。

1. 外形：能力风暴个人机器人的外形被设计为宇宙飞船的形状，它的重量是 1kg。

2. 机电系统：能力风暴个人机器人的底盘由高强度的 ABS 做成，有两只高性能

直流电机，通过锌合金减速器，带动两只驱动轮和两只带缓冲的导向轮，使机器人可以原地转向。

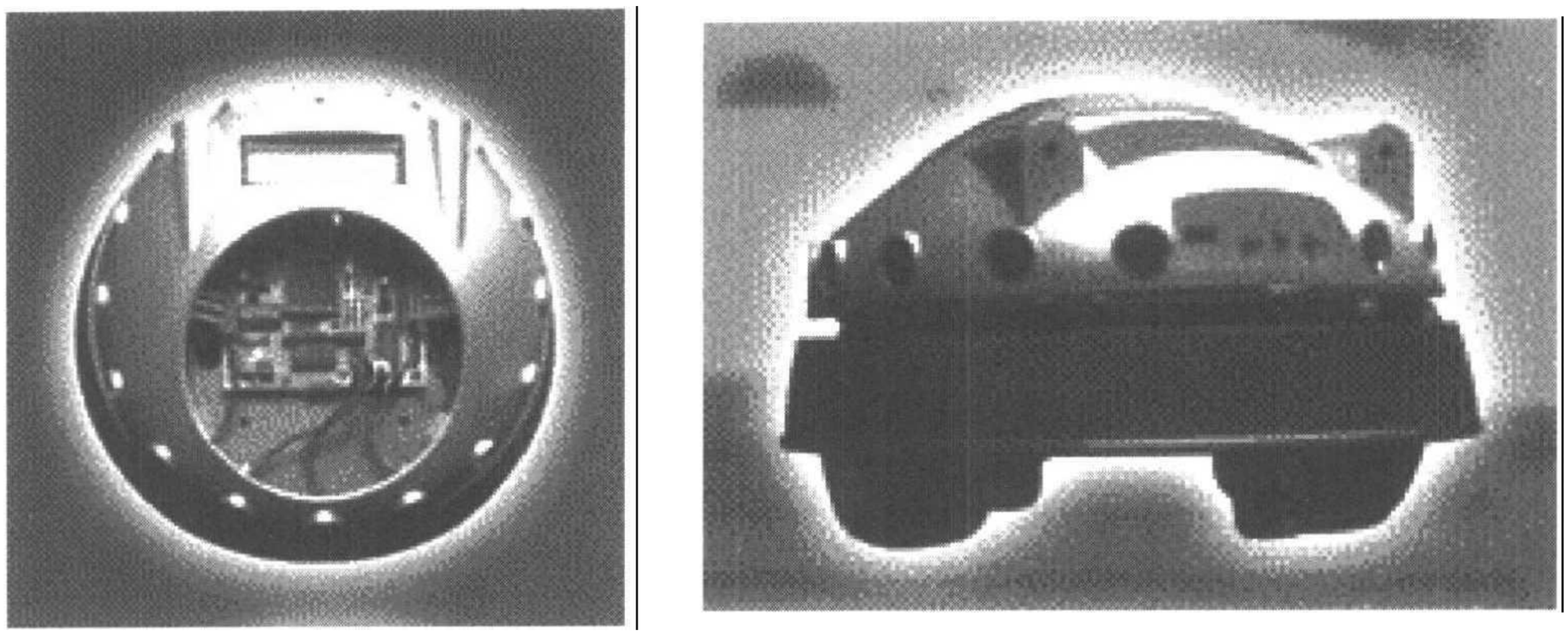


图 1.1　能力风暴个人机器人

3．电池：能力风暴个人机器人配有高能量镍氢电池和智能充电器，使机器人可以连续运行 3h 左右。

4．传感器：能力风暴个人机器人配备 5 种传感器，它们是：光电编码器、光敏传感器、红外接近传感器、麦克风和碰撞传感器等。

5．计算机硬件：能力风暴个人机器人采用 68HC11A1 单片机，有 32KB 内存，并有较强的计算能力，通过 LCD 液晶显示屏可以显示 16×2 个字符。

6．操作系统：能力风暴个人机器人装有多任务操作系统 ASOS。

### 1.1.2　特点和用途

能力风暴个人机器人融合了造型设计、机械、电子、电机、计算机软件、计算机接口、传感器、人工智能等众多先进技术，因而它的功能十分强大；能力风暴个人机器人拥有开放式的开发工具平台，使它的应用面十分广泛。作为教学用智能机器人，它的特点和用途如下：

1．特点：

（1）模块化结构：你可以轻松自如地拆卸和组装能力风暴个人机器人，在拆卸、组装的过程中，你将对能力风暴个人机器人的结构有深入的了解，并初步掌握机器人的工作原理。

（2）交互式 C 语言：具有标准 C 语言的特性，出色的交互式功能，使语言学习变得直观、轻松并富有乐趣。

（3）开放式接口：能力风暴个人机器人具有开放性的接口，你可以根据不同任务

的需要，增加相应的功能。

（4）扩展模块：是为增加机器人的特殊功能而设置的。如 ASDIY 实验卡、ASIO 输入输出扩展卡、ASIRanger 红外测距卡、ASSonar 超声测距卡、ASSpeach 可录放声卡、ASHand 手爪装置、ASArms 手臂装置等等，你可以根据需要进行选择。

2．用途：

（1）学习平台:能力风暴个人机器人为学习和掌握智能机器人的有关知识及技能，提供了一个直观的平台。特别是在 68HC11 上开发的专用系统——交互式 C 语言（简称 JC 语言），能交互运行一条语句或一个函数，使计算机语言编程变得直观具体，有利于计算机语言的学习，同时为语言编程和程序调试提供了极大的方便。在这个学习平台上，你很快就能学会并掌握 JC 语言，为你的机器人编写程序。在这一学习过程中你不会感到乏味，学习速度之快、掌握内容之多、学习兴趣之浓、学习收获之大是你难以想象的。可以说这是具有革命性和时代性的学习平台。

（2）实验平台：在这个平台上你可以进行电子、单片机、传感器和机械等设计，进行人工智能、数字控制、电机控制、数字信息处理等实验。这些听起来非常专业又非常高深的实验，在这个平台上你会不知不觉轻松自如地去完成。

（3）研究开发平台：在机器人 ASBUS 总线的支持下，你可以扩展多个直流电机或伺服电机，还可以扩展超声、颜色、红外等多种传感器，使你的机器人智能化程度更高。只要你有一个明确的任务，在 ASBUS 的支持下，运用 JC 语言一定能实现你的设想。

（4）能力培养平台：能力风暴个人机器人是拥有多项专利的高科技产品。你通过亲手装配、实验、编程和实施机器人项目、参加机器人比赛，直至设计出你自己独特的机器人伙伴，能力风暴个人机器人将引导你进入激动人心的前沿领域，你将发现自己的潜力无穷、自己的智慧无限，自己的信心在增强、综合能力在提高。

### 1.1.3 基本结构

能力风暴个人机器人具有“积木式”可拆装结构，你可以采用从上到下，从外到里的顺序，把能力风暴个人机器人全部拆散为零件，然后你再按照由里到外，由下到上的顺序把机器人重新组装起来。这对于喜欢动手的青少年朋友们来说是一个非常好的学习机会。

现在我们就通过能力风暴个人机器人（以下简称机器人）的分解图，一一介绍机器人的各部分名称和作用。

1．透明上盖

透明上盖安装在机器人外壳的上部。这个上盖可以随时拆下，以便在机器人的底盘上加装和扩充机械、电子装置。透明上盖如图 1.2 所示。

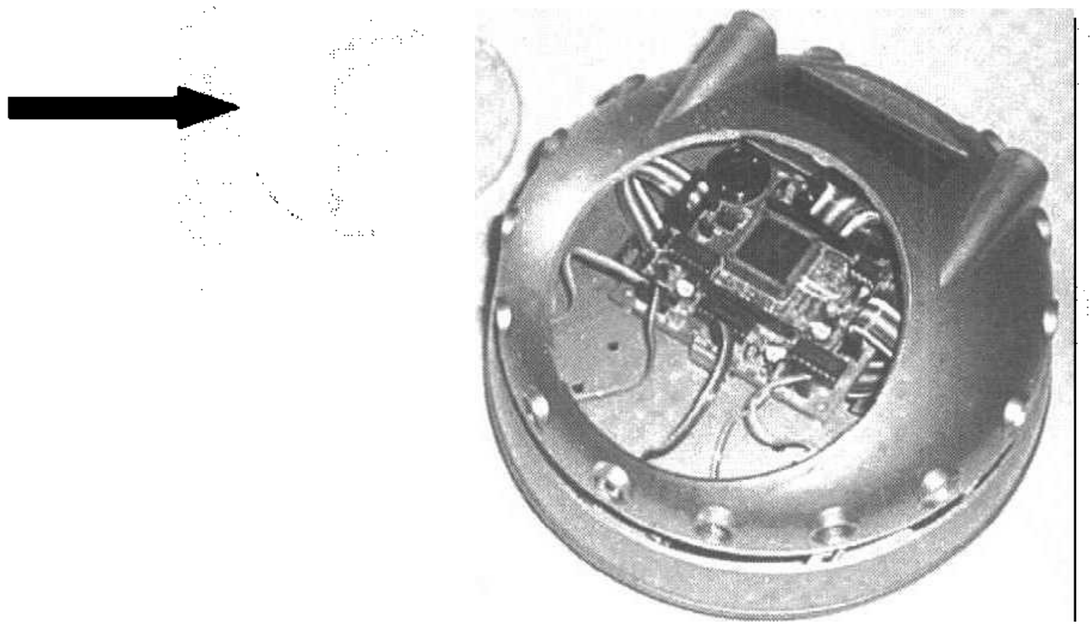


图 1.2　透明上盖

2．外壳

机器人的外壳直接卡在机器人的底盘上。在机器人的外壳上装有一块 LCD 液晶显示屏，以及红外接近传感器和光敏传感器等。

你看到在机器人外壳的圆周上有一圈“射灯孔”吗？这些“射灯孔”的主要作用是为了增加或改变传感器的位置而做的预留孔。有了这些预留孔，你便可以根据不同的任务和需要去重新组合、安装相关的传感器，以提高机器人的“感知”能力。机器人外壳如图 1.3 所示。

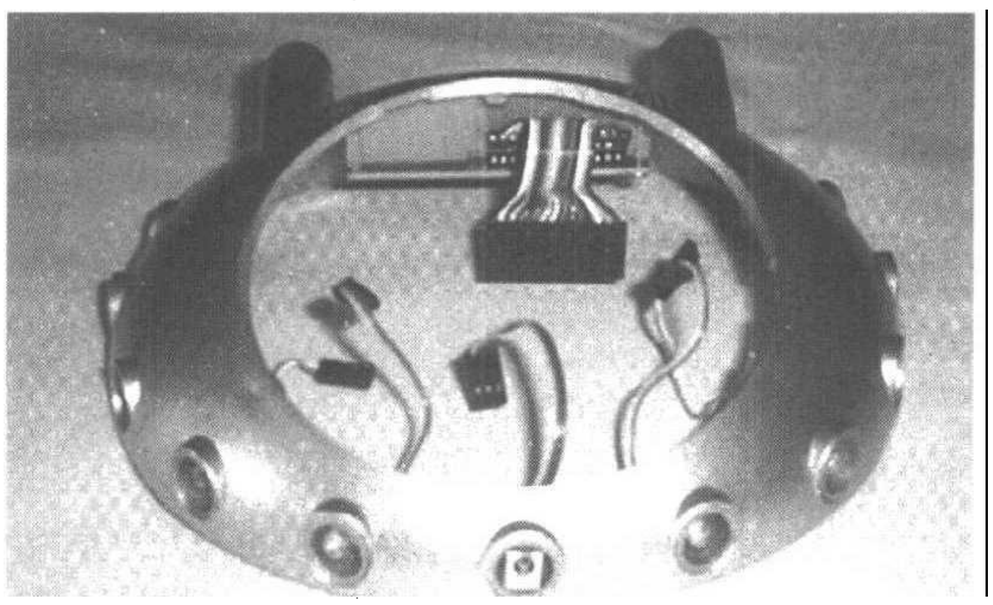


图 1.3　外壳

3．碰撞环

机器人的碰撞环位于底盘的下部。碰撞环通过螺钉和弹簧与底盘相连，使碰撞环能灵敏地接收碰撞信息。在底盘下部装有四个分别成 90° 夹角的碰撞开关，使机器人能感知来自不同方向的碰撞力。机器人的碰撞环如图 1.4 所示。

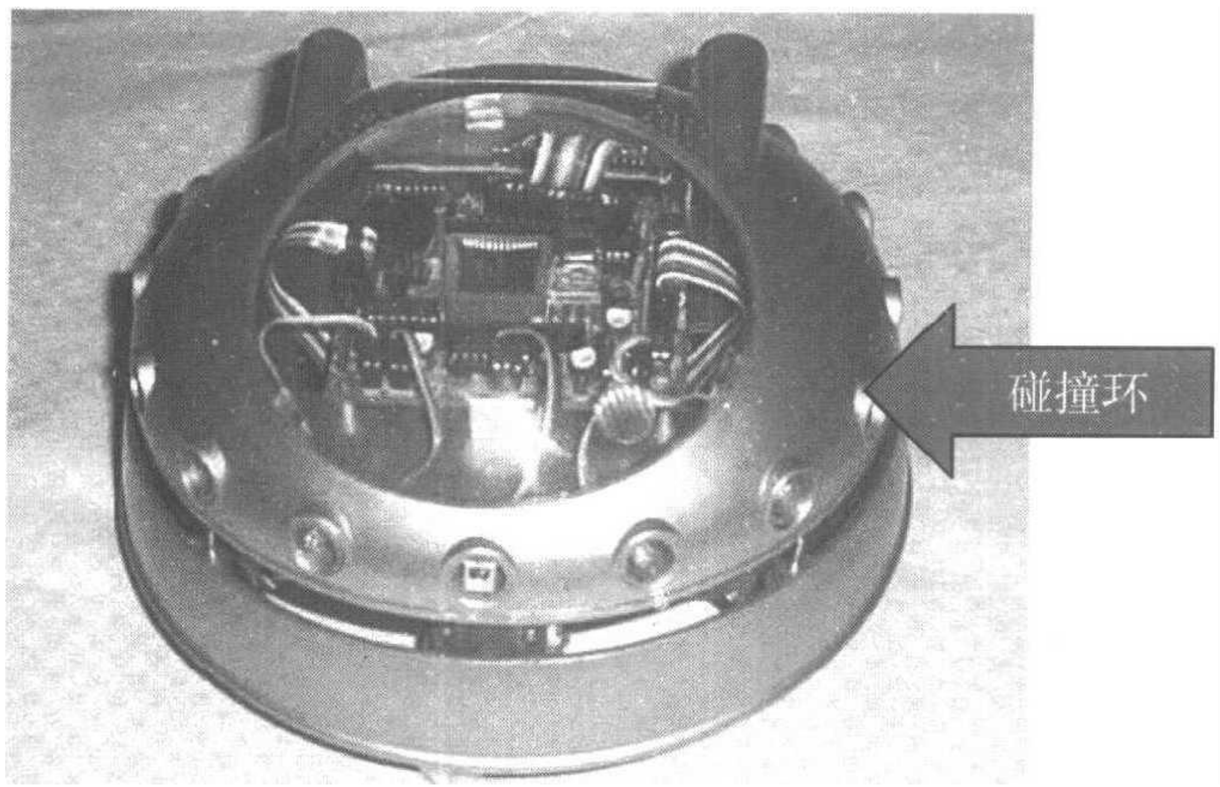



图 1.4　碰撞环

4．底盘上部

在机器人的底盘上部安装着机器人的主线路板，如图 1.5 所示，这是机器人的控制板。在控制板上的这部分计算机硬件是智能机器人的“大脑”，也可以说是智能机器人的“中枢”。机器人通过“大脑”识别来自外部的信息，并根据不同的感知出相应的“决策”，以控制机器人的“动作”。

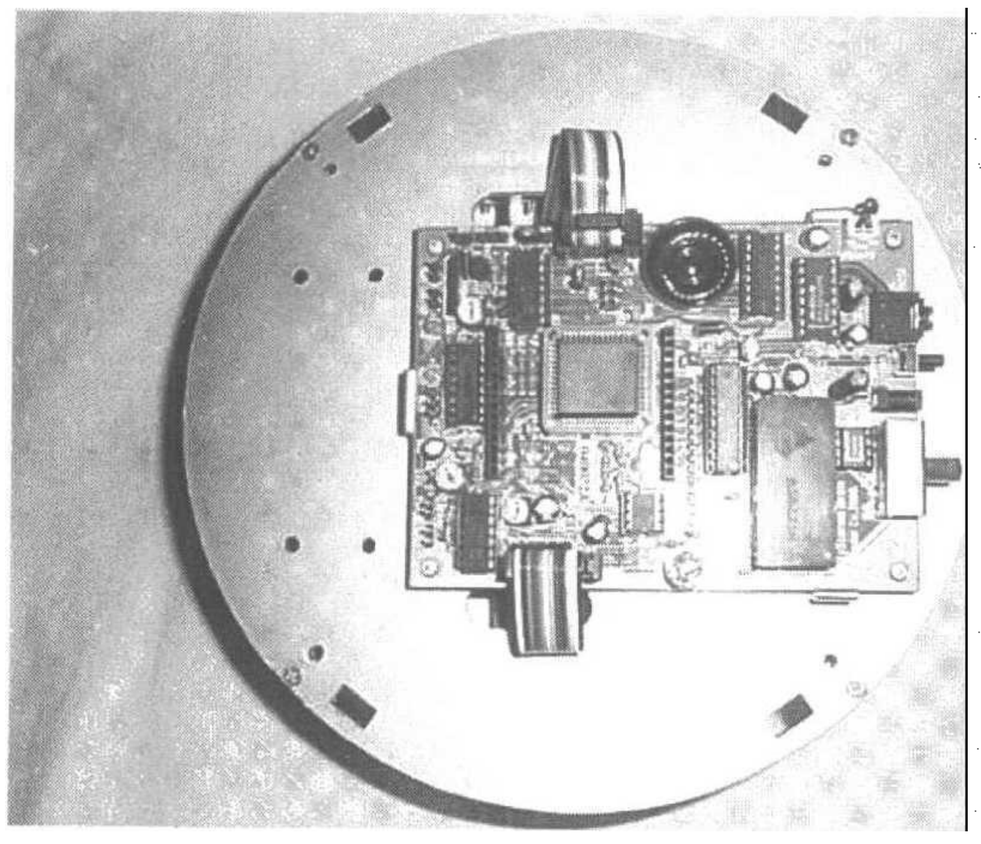


图 1.5　底盘上部

5．底盘下部

在机器人的底盘下部是机器人的驱动部分，如图 1.6 所示。下面我们逐一介绍。

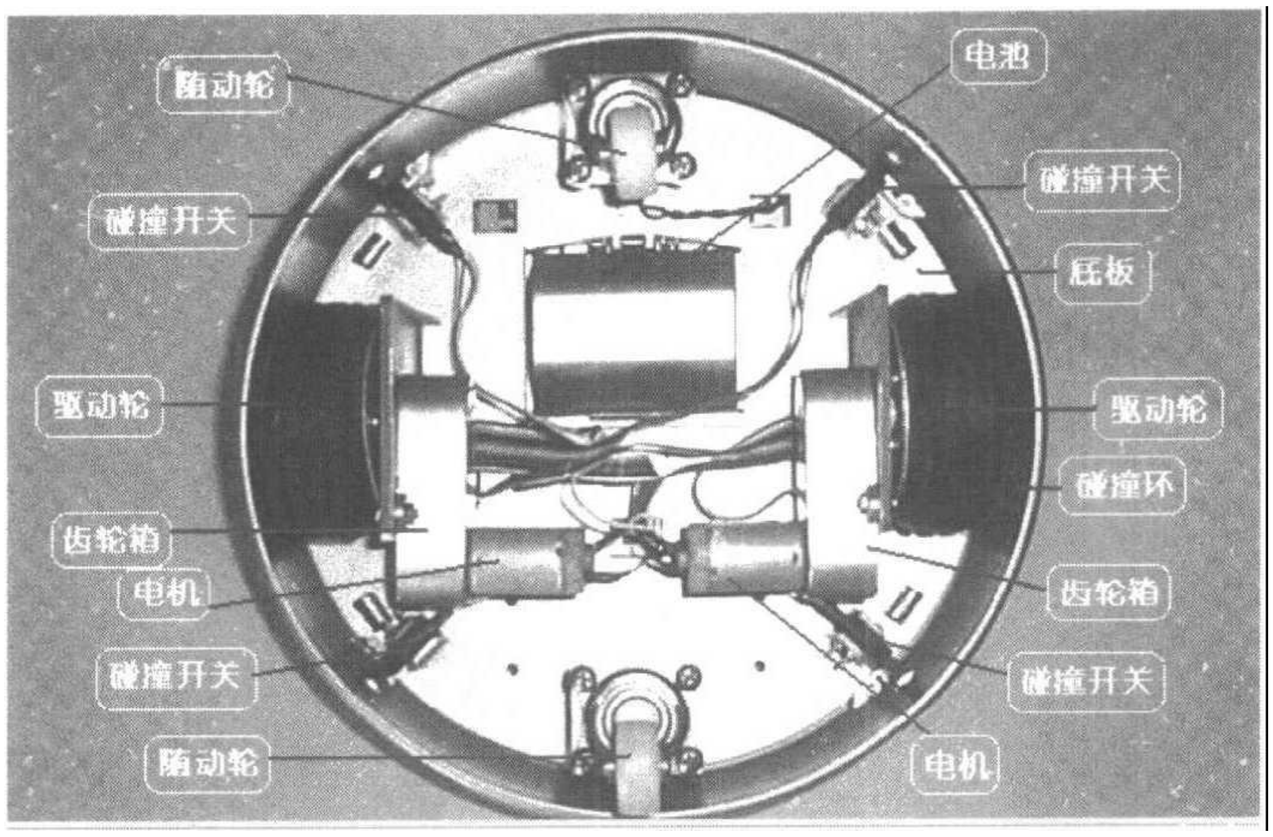



图 1.6　底盘下部

（1）电机：这是机器人的能量转化装置，通过它就可以把电能转化为动能，使机器人实现转动或移动。

（2）减速器：这是将高速运动转变成低速运动的一种机械装置。它通过三级齿轮减速，把电机的转动传给驱动轮，以改变机器人的运动速度和转矩。

（3）驱动轮：我们也称为主动轮，这是驱使机器人行走的主要装置。驱动轮通过减速器与电机相连，当电机转动后直接带动驱动轮的转动，使机器人可以直行前进、后退，左、右转弯，原地旋转等。

（4）随动轮：我们又称它为从动轮或导向轮，这也是驱使机器人行走的装置之一。随动轮有非常灵活的方向轴和缓冲器，使机器人可以原地旋转。随动轮在驱动轮的带动下既可引导机器人的方向又可与驱动轮一起使机器人保持平衡。

（5）电池：机器人使用的是 6V、1.2A 的镍氢电池，它是机器人的能源，可以供给机器人所需的电能。

（6）碰撞传感器：由 4 只碰撞开关和一个碰撞环组成。4 只碰撞开关分别安装在机器人的左前、右前、左后、右后 4 个位置上，加上设计巧妙的碰撞环，使机器人有了全身碰撞感知的能力。

6．光电编码器

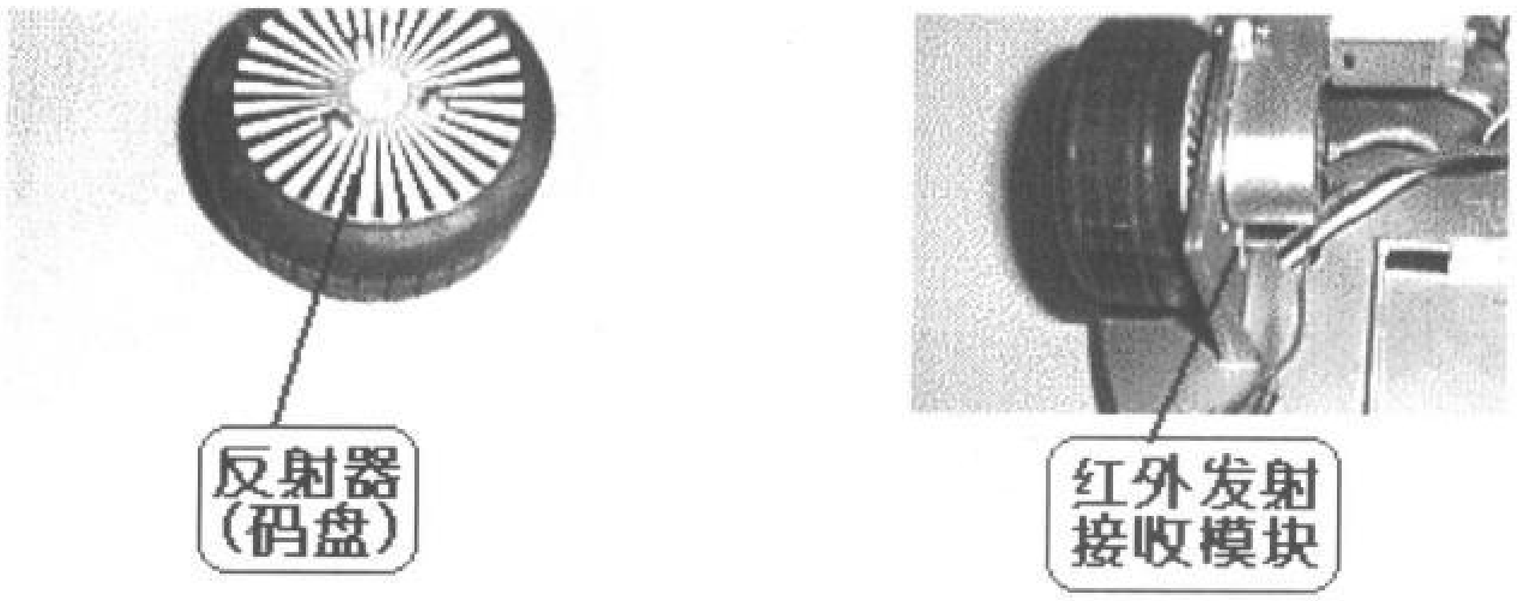
机器人的光电编码器是由红外发射接收模块和反射器两部分组成的。

你在两个驱动轮的内侧，可以看到各安装了一块黑白相间的铝合金圆片，这就是光电编码器的反射器，我们又称它为码盘。这个码盘被分成了黑白相间的 66 等份，如图 1.7（a）所示。

你在减速箱下方可以看到一块大约 2.5cm×1cm 的长方形线路板，这就是光电编

码器所用的红外发射接收模块线路板。这个线路板是目前世界上最小的红外发射接收模块，如图 1.7（b）所示。

光电编码器是一种能够传递位置信息的传感器，通过光电编码器我们可以比较准确地知道机器人行走的距离和转弯角度。




（a）　　(b)

图 1.7　光电编码器

7．功能结构图

机器人的功能结构如图 1.8 所示。现在我们按照结构图中从上到下的顺序进行介绍。

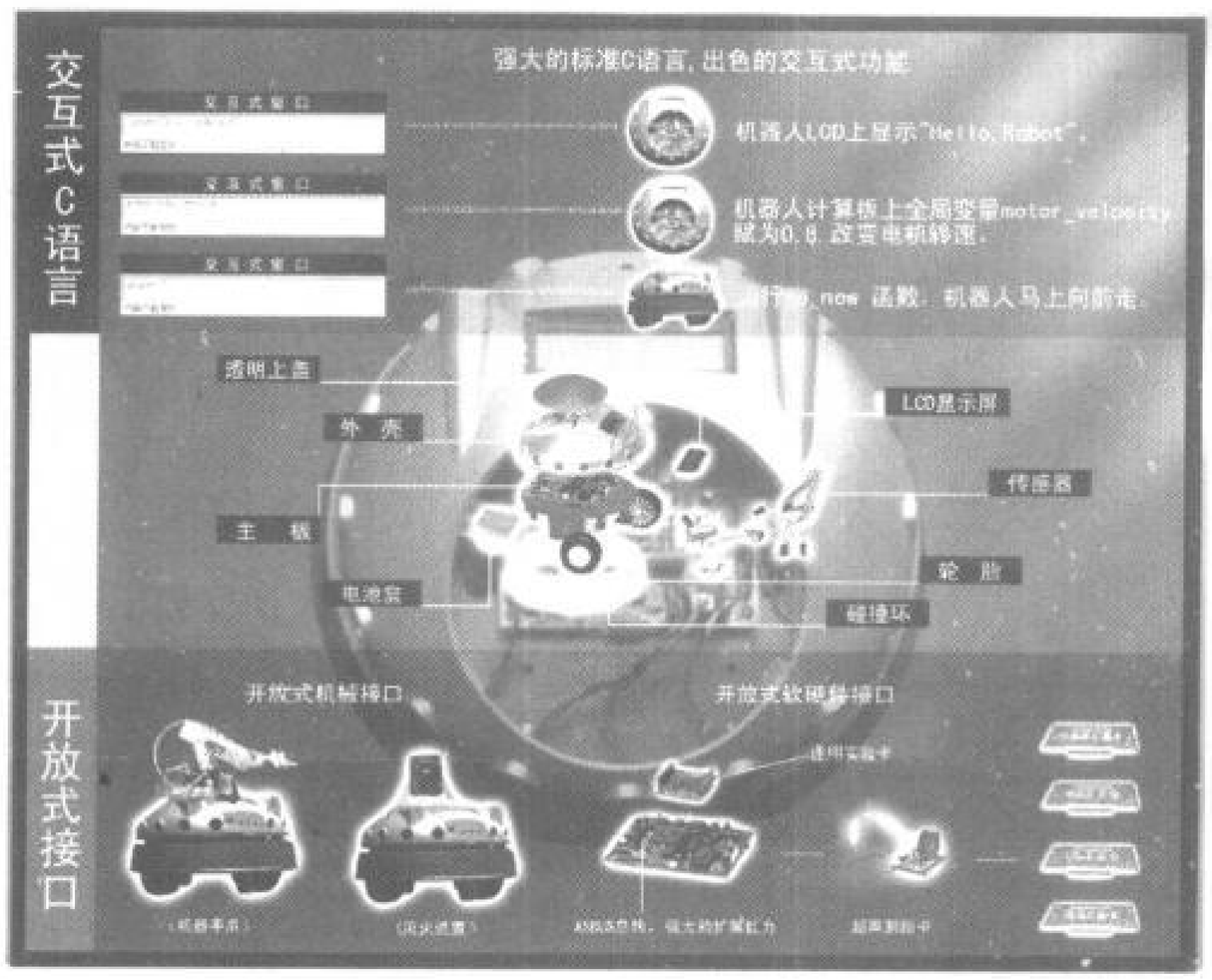



图 1.8　能力风暴个人机器人功能结构图

（1）交互式C语言

交互式C语言（以下简称JC语言）是用于机器人的专用开发系统，它是由编译环境和操作系统两部分组成的：

● 编译环境

JC的编译环境包括交互式命令行和调试功能。

在交互式命令行，C的表达式、函数的调用和JC命令等都可以直接输入。输入后JC先在计算机上编译所输入的内容，然后通过串口传给机器人操作系统。经过机器人操作系统计算后，将结果返回，显示在计算机的JC窗口里或机器人的LCD液晶显示屏上。

当你编写的JC语言程序出现语法错误时，编译环境则不能编译，程序就不能下载。这时调试功能将会一步一步提示引导你修改程序，直到程序正确并下载成功为止。

● 操作系统

机器人的操作系统是机器人的软件资源和硬件资源的“管理者”，通过操作系统，可以有效地管理好机器人的一切资源。

（2）模块化结构

模块化结构包括机械零部件和电子元器件。如上盖、外壳、底盘、碰撞环、齿轮箱、轮子和电池、电机、控制板、液晶显示器以及各种传感器等。由于各个模块采用交互式语言来驱动，所以各部分可以分解组合。

（3）开放式接口

开放式接口是机器人的重要特点之一，它包括机械接口和电子接口。我们可以在原有控制板的基础上扩展机械接口，如图1.8所示的“机械手爪”和“灭火装置”，就是利用机械接口扩展的两种使用装置。

如果我们要想使机器人的功能更强大，可能现有的控制板就不够用了，这时我们只要通过ASBUS总线扩展卡，就可以非常方便地扩展控制板的功能。如增加超声测距卡、传感器扩展卡、电机扩展卡、I/O扩展卡及通用扩展卡等，使你的机器人在原有基础上，智能化程度更高，功能更强大。

你已经认识了能力风暴个人机器人，了解了它的基本结构，并一定渴望马上掌握机器人的操作方法，实现你的梦想。

## 1.2 机器人的基本操作方法

我们在了解机器人的基本结构之后，已经知道通过JC语言可以与机器人进行交流，并通过JC语言为机器人编程，使你的机器人能按照程序要求完成任务。

现在的关键是，要为你的机器人编写程序，必须具备两个主要因素：一是硬件设备，二是软件工具。如果把硬件看成是“机器人”的躯壳，那么软件就是“机器人”的灵魂或大脑。现在我们就分别介绍硬件设备和软件工具。

### 1.2.1　硬件设备

机器人与计算机的连接如图 1.9 所示。

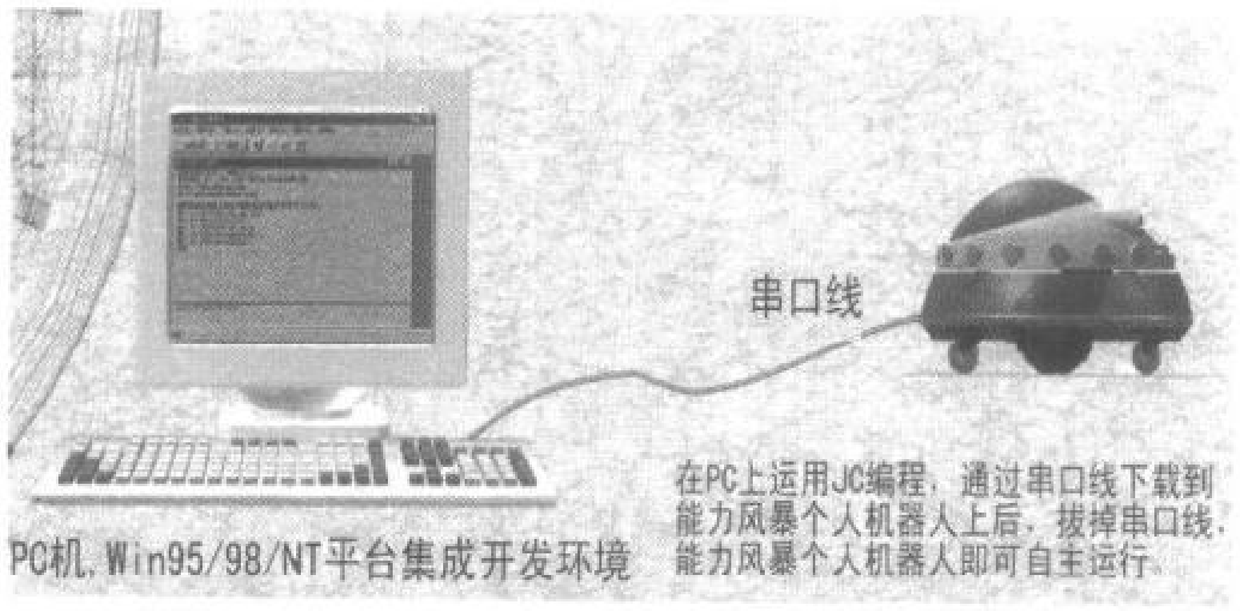



图 1.9　计算机与机器人连接图

1．首先要有一台计算机，操作系统为 Windows 95/98 或 Windows NT 4.0 以上平台，计算机有一个 9 针的空余串口。

2．要有一台机器人。

3．有一根可以连接计算机和机器人的串口线。

4．有一个为机器人专配的智能充电器。

### 1.2.2　软件工具

你的机器人能够识别的是 JC 语言。你要用这种语言与你的机器人进行交流，为它编写程序，教它完成各项任务。因此你必须要有一张 JC1.0 的系统盘，这是你为机器人编写程序的软件工具。

1．安装 JC 语言系统

JC 语言是机器人的专用语言，一般的计算机中没有安装这种语言，所以在我们使用之前必须安装 JC1.0 系统。

现在请你和我一起按下面的步骤在计算机中安装 JC1.0 系统。

（1）在计算机的光盘驱动器中放入机器人配套光盘。

（2）Windows 将自动安装 JC1.0 系统，并把系统安装在 C 盘“JC1.0”目录下，同时在桌面上放置了快捷方式；也可以直接运行 Setup.exe，选择 JC 的安装目录，装载语言系统。

（3）当 JC 语言系统安装完毕后，在 Windows 桌面上，会出现如图 1.10 所示的交互式 C 语言图标。

到此，我们的第一步——安装 JC 系统就完成了。


图 1.10　交互式 C 语言(JC)图标

2. 为机器人下载操作系统

机器人的操作系统像机器人的管家一样管理着机器人的软件资源和硬件资源。在操作系统的支持下，你的机器人才能识别程序，并按程序的步骤完成指定的任务，因此第二步我们要为机器人下载操作系统。

在为机器人下载操作系统之前，我们先来认识一下机器人的操作面板，如图 1.11 所示。

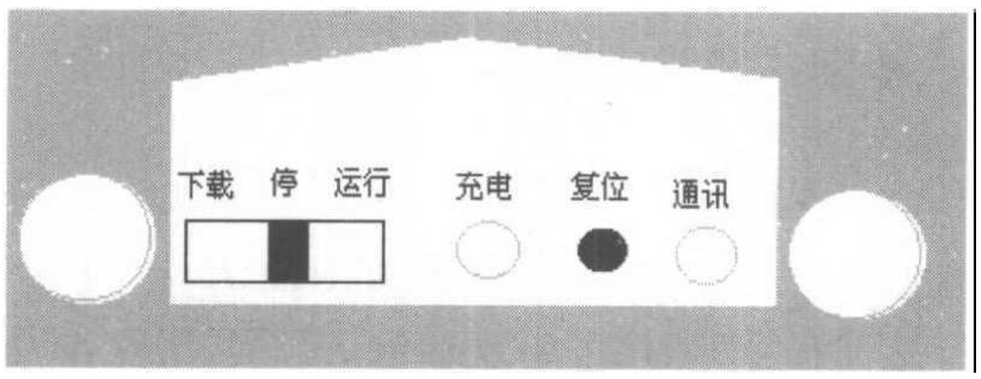



图 1.11　机器人的操作面板

机器人的操作面板在机器人外壳的后侧，上面有电源开关、充电插孔、复位键和串行通信接口。

当你把机器人的电源开关拨到“运行”位置时，如果听到“嘟”一声响，机器人的 LCD 交互窗口显示“JC V1.0 Grandar Ability Storms”和一个跳动的太极图，表示机器人的操作系统已经可以正常运行了，这时为机器人下载操作系统这一步可以省略。

如果开关拨到“运行”位置，机器人没有声响，或机器人的 LCD 窗口也没有显示“JC V1.0 Grandar Ability Storms”和跳动的太极图，那你就必须重新为你的机器人

下载操作系统，否则你将无法与机器人进行交流。

如果需要为机器人下载操作系统，请跟我按下面的步骤来进行：

（1）将机器人与计算机用配套的信号线连接，计算机接 9 针串口，机器人接串行通信接口。注意接在计算机 9 针串口的连接线不用拔下来。

（2）在桌面上双击如图 1.9 所示的 JC 语言图标，计算机显示屏会出现如图 1.12 所示的“JC”对话框。

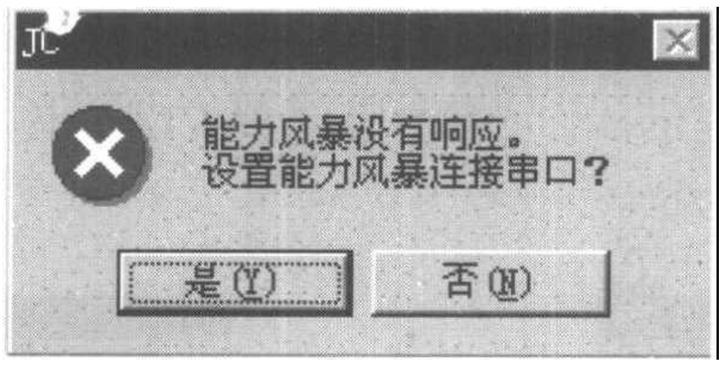



图 1.12 “JC”对话框

（3）把机器人的开关拨到“运行”位置上，此时机器人主控制板上绿色发光二极管亮为正常。如果绿色发光二极管不亮，其故障排除方法将在 1.5 节介绍。绿色发光二极管位置如图 1.13 所示。

（4）单击图 1.12 所示对话框的“是(Y)”按钮，弹出“能力风暴控制板设置”对话框，如图 1.14 所示。

（5）现在在“能力风暴控制板设置”对话框中选择一个 COM 串口，然后单击“检测（T）”按钮。

如果出现如图 1.15 所示的提示，说明机器人已连接在相应的计算机串口上。

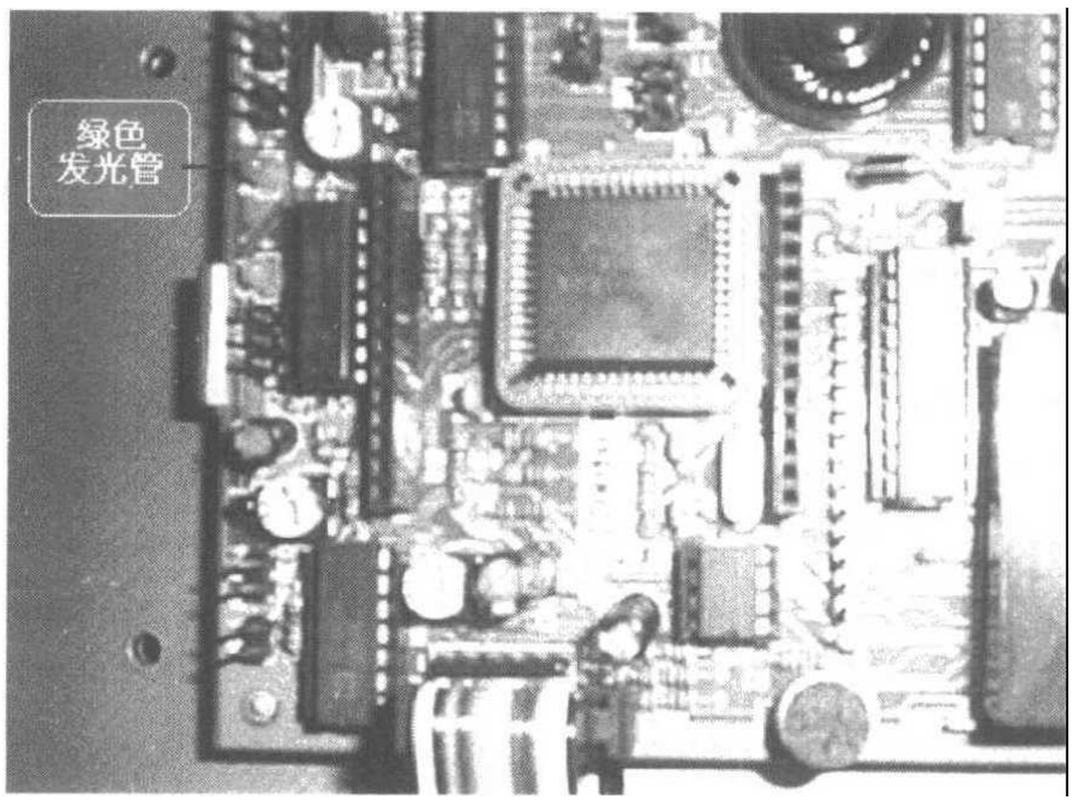



图 1.13 绿色发光管位置图

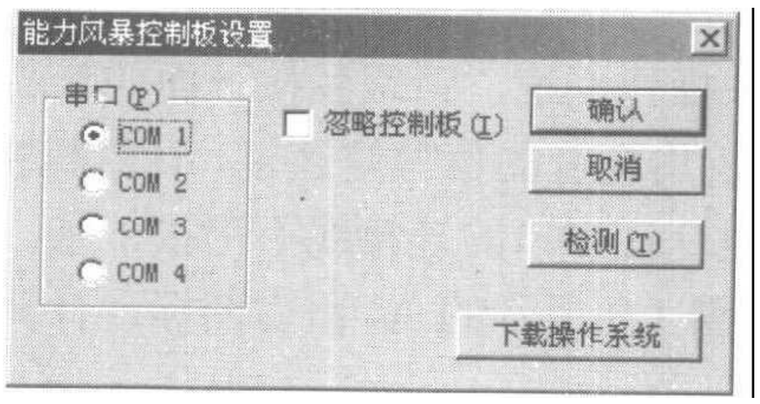



图 1.14 “能力风暴控制板设置”对话框

如果出现如图 1.16 的提示，你必须重新选择 COM 串口，直到找到对应的串口，出现如图 1.15 所示的提示为止。


图 1.15 对连接成功进行提示

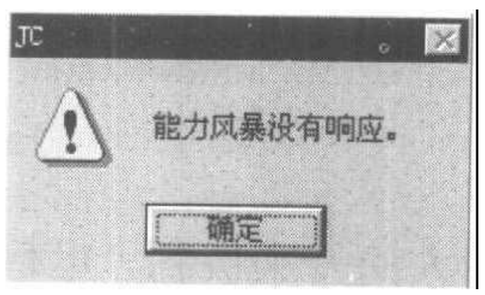



图 1.16 对连接失败进行提示

（6）在确认找到 COM 串口之后，单击图 1.14 所示的“能力风暴控制板设置”对话框中“下载操作系统”按钮，会出现如图 1.17 所示的“打开”对话框。

（7）你在“打开”对话框中选择“Ability_Storms.jcd”系统库文件，然后打开此文件，就会出现如图 1.18 所示的“下载操作系统”对话框。

（8）现在请你按照如图 1.18 所示的“下载操作系统”对话框提示进行操作。

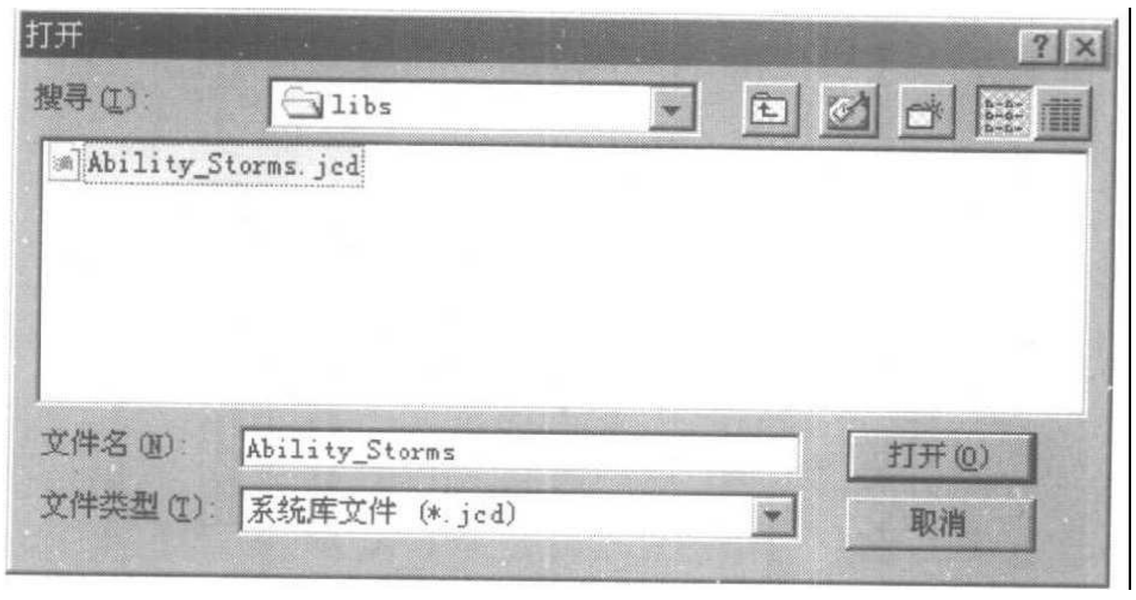



图 1.17 “打开”对话框

注意：在下载操作系统过程中，JC（能力风暴版）窗口会给出“正在下载操作系统到能力风暴...请稍等”的提示，如图 1.19 所示。

此时机器人主控制板上的黄色发光二极管闪动，表示数据正在传送。黄色发光二

极管的位置如图 1.20 所示。

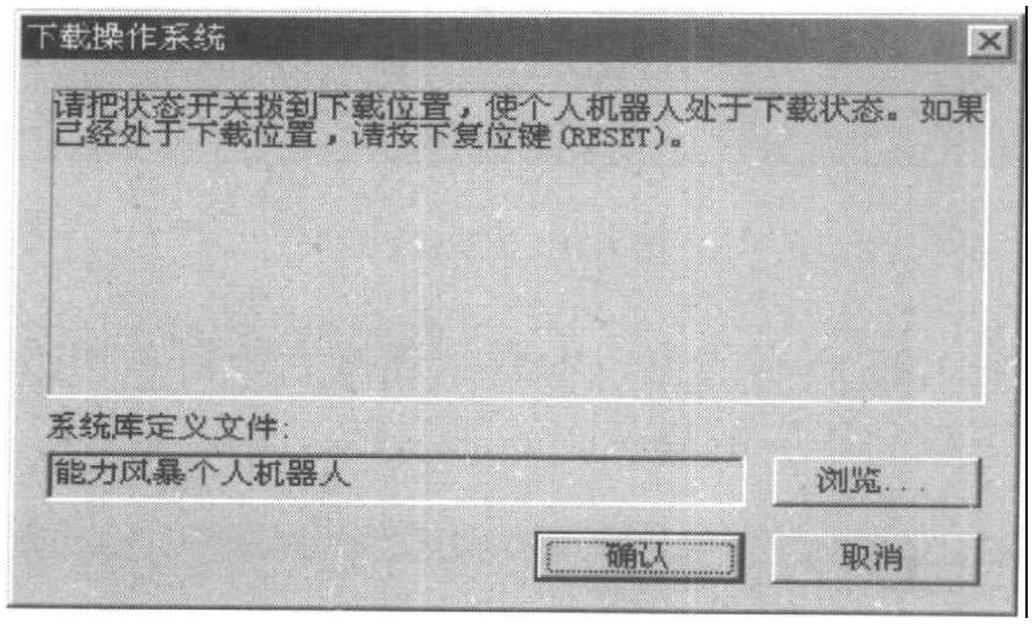



图 1.18　“下载操作系统”对话框

（9）当操作系统下载完成后，就会出现如图 1.21 所示的“下载成功”对话框。请按提示将机器人的开关拨至“运行”位置后，按“确定”按钮。

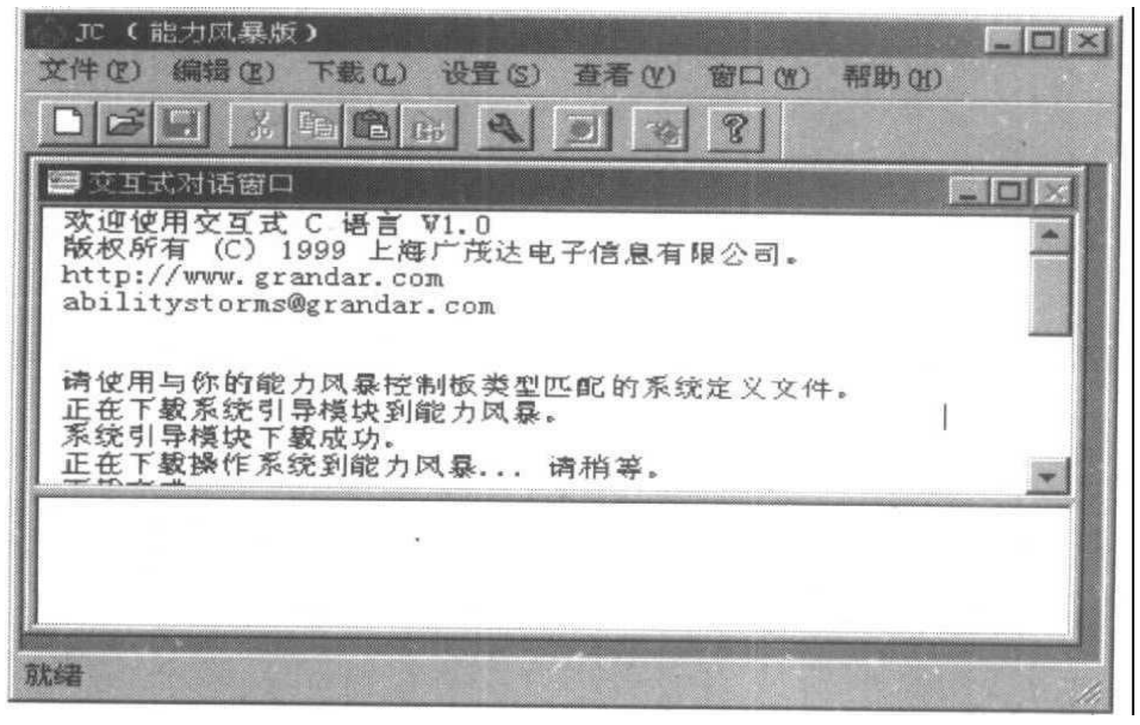



图 1.19　下载操作系统提示

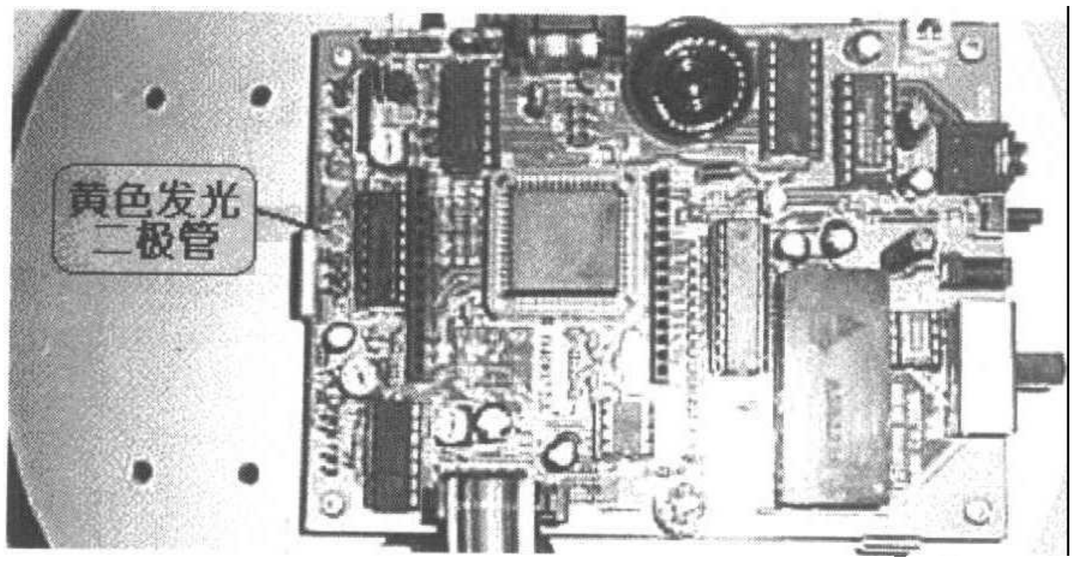



图 1.20　黄色发光二极管位置图


图 1.21　下载成功

随后，你会在如图 1.22 所示的窗口中看到连接、刷新和初始化机器人的过程。过程结束后出现“代码下载完毕”的提示。

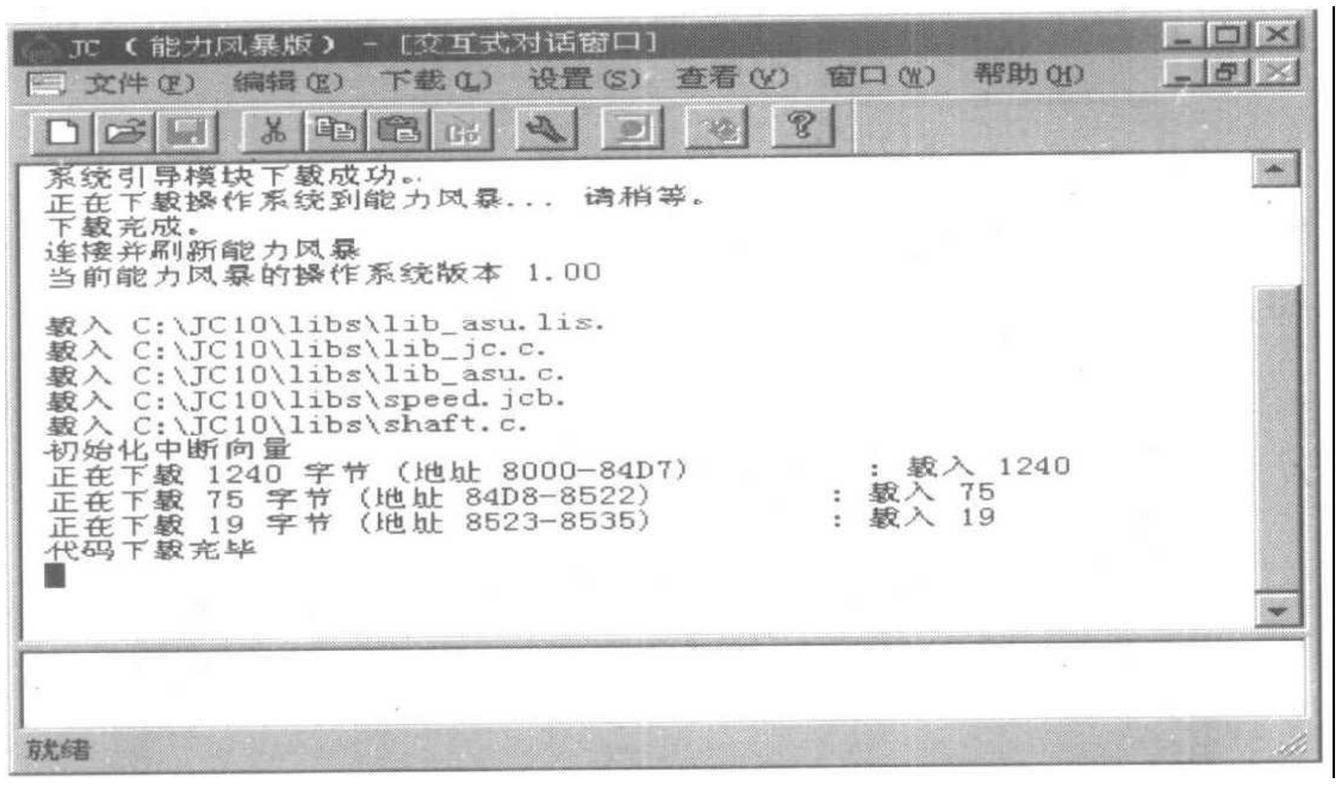



图 1.22　连接、刷新和初始化过程

现在你已经成功地为机器人下载了操作系统。JC 在你和你的机器人之间架起了一座“桥梁”，机器人可以任你指挥了。

3．用 JC 与机器人交流

运用 JC 与机器人进行交流，是一件非常容易、非常有趣的事。我们一起试试看。

【实例】请在“JC（能力风暴版）”对话窗口中，直接键入“2+2”，如图 1.23 所示。

然后按一下回车键，在 JC 窗口中得到的响应是“<int> 4”，如图 1.24 所示。

在图 1.23 和 1.24 中，我们看到 JC 窗口的下方，有一个条形窗口。这个窗口是一个带历史记录的命令行编辑器，它的作用是可以编辑和重用输入过的语句和命令，通过键盘上的“↑”、“↓”键即可浏览。现在你按一下“↑”键，刚才输入的内容又显示在命令行编辑窗口中，如图 1.25 所示。

在这个命令行编辑器中，Windows 的编辑命令也可以使用。

我们通过实例 1 说明你和机器人利用 JC 进行交流的过程。

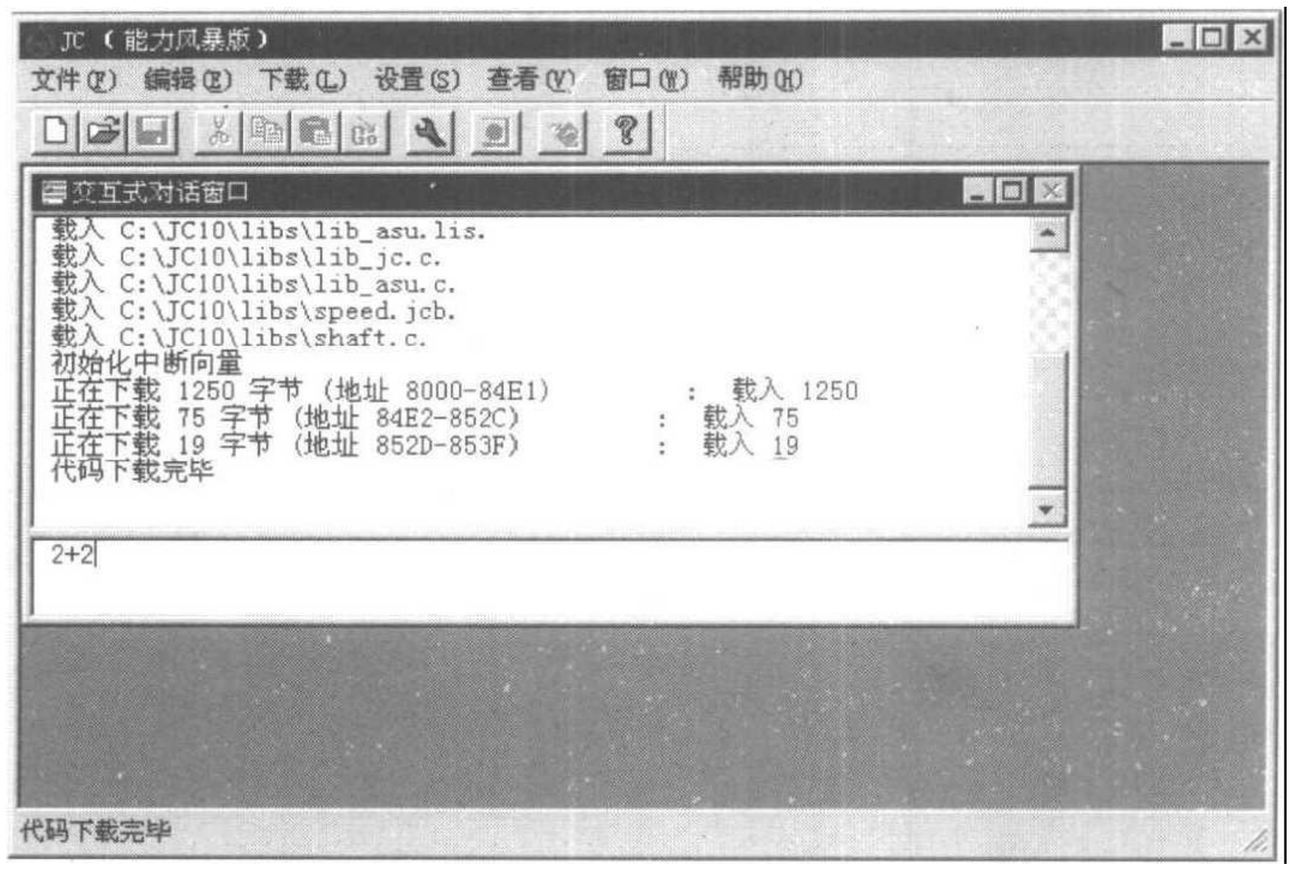



图 1.23　JC 窗口

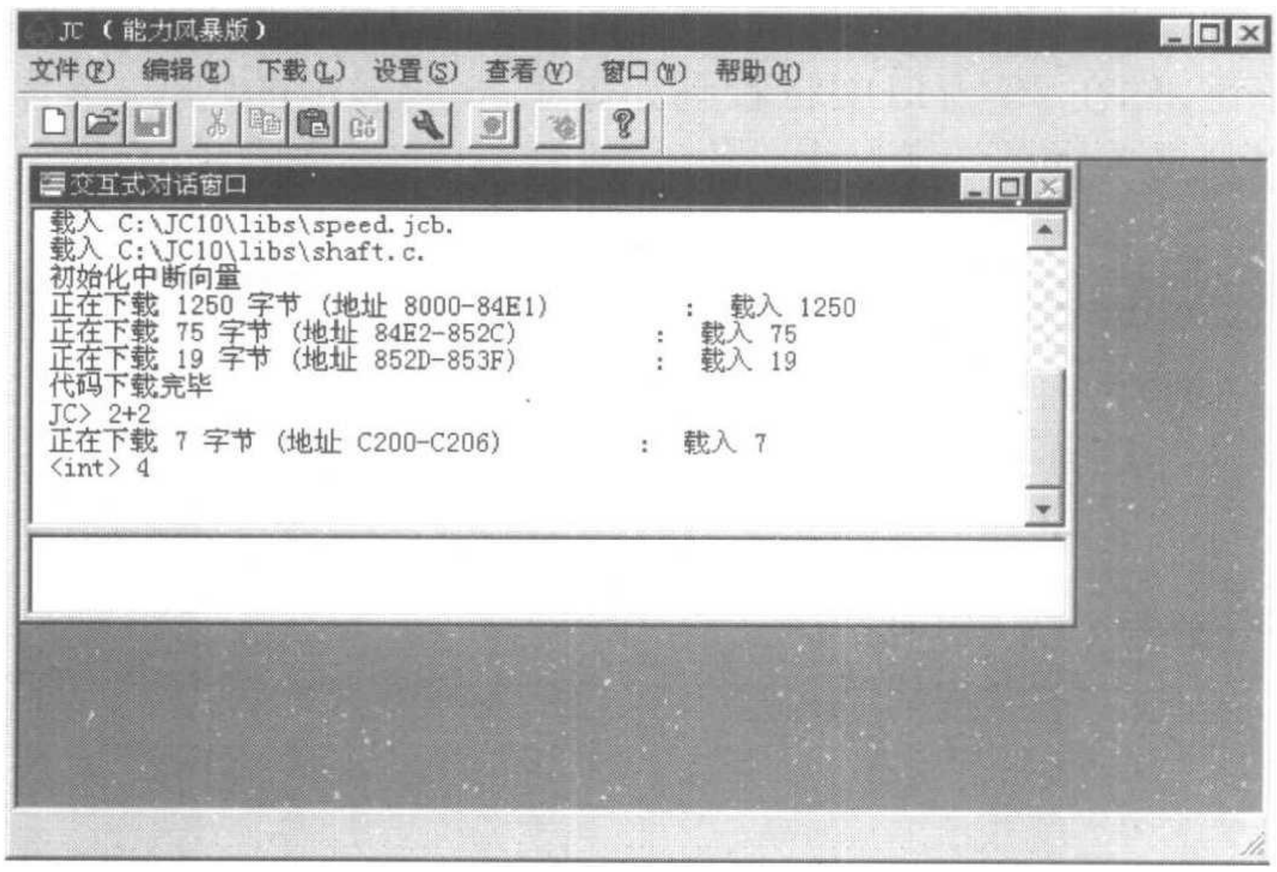



图 1.24　JC 对话窗口

当你键入表达式“2+2”之后，JC 先在计算机上编译此表达式，然后通过串口将此表达式传给机器人操作系统进行计算。在表达式经过机器人计算编译后，返回结果“4”，并将结果显示在计算机的 JC 窗口中。到此你和机器人的交流过程就告一段落。

通过对这个实例的说明，你是不是已经感到 JC 的重要了？下面我们再通过几个练习体会一下。

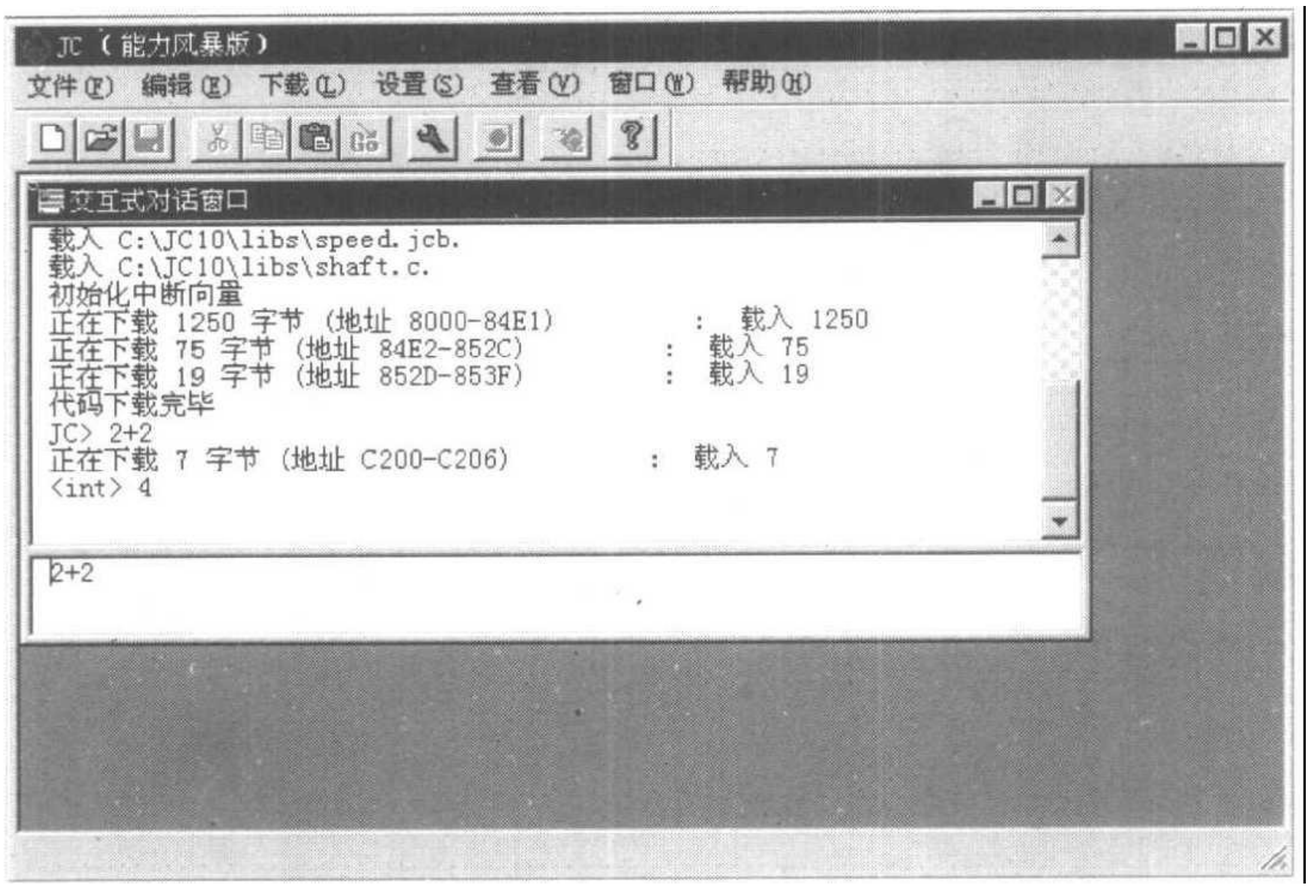



图 1.25　JC 窗口和命令行窗口

【练习 1】请在 JC 命令行编辑器窗口中，键入内容：printf("Hello，world!\n")。然后按一下回车键，你在 JC 窗口中将看到如图 1.26 所显示的内容，请你把机器人 LCD 窗口显示的内容记录下来_____。

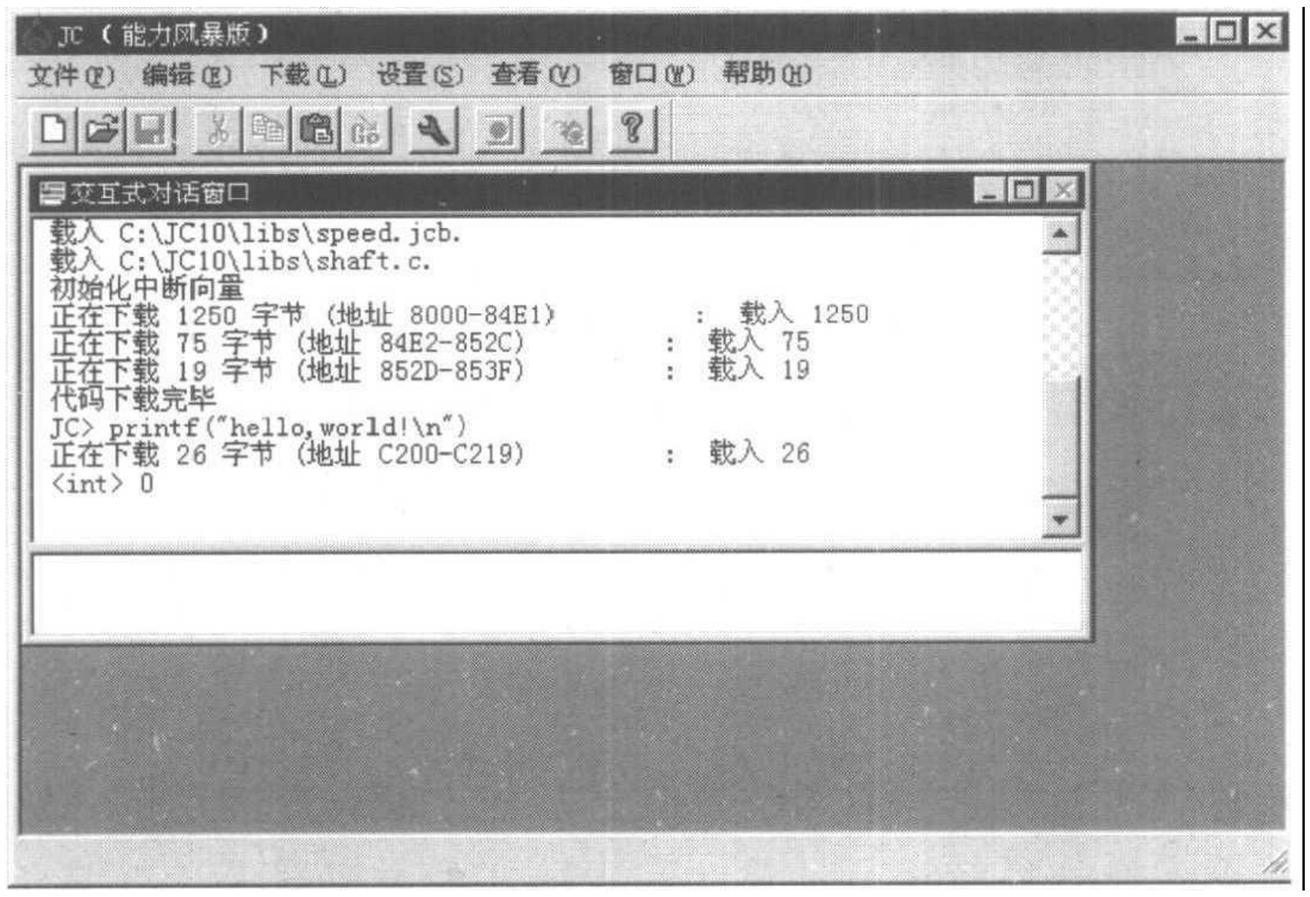



图 1.26　执行“printf”提示

【练习 2】想一想怎样让机器人 LCD 窗口显示“Welcome my friend!”。仿照练习 1 的方法试一下。

【练习 3】在 JC 命令行键入：beep()。

4. 建立文件夹

通过命令行编辑器给机器人的指令是有限的，要想让机器人按照我们的要求去完成任务，光靠命令行编辑器是远远不够的，因此我们要为机器人编写程序。为了让机器人完成不同的任务，就要给它编写不同的程序。所以为了对你编写的程序进行有效的管理，我们先在计算机中为以后要编写的程序建立一个文件夹，你可以把这个文件夹建在如图 1.27 所示的 C 盘下的“Jc10” 文件夹中。

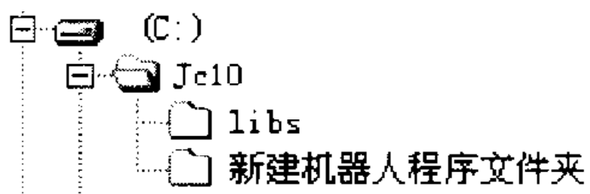



图 1.27 建立机器人程序文件夹

好了，到现在为止，我们的各项准备工作已经完成，可以为机器人输入程序了。

## 1.3 让你的机器人动起来

你一定非常渴望让你的机器人动起来，但是你现在还不会为你的机器人编写程序，不要着急。在我们还没有学会为机器人编写程序之前，你可以先掌握为机器人输入程序的方法。

### 1.3.1 为机器人输入程序

为机器人输入程序也是我们本章要掌握的操作之一，现在请你和我一起按照下面的方法为机器人输入程序。步骤如下：

1. 新建文档

方法：在 JC 窗口菜单栏中单击“文件”菜单，然后单击“新建”选项或直接单击图标建立空白文档，新建的空白文档如图 1.28 所示。

如果在 JC 窗口中直接输入程序，JC 窗口就会出现错误信息提示，如图 1.29 所示。所以必须建立一个空白文档，然后在空白文档中输入程序。

2. 输入程序

现在你可以在如图 1.28 所示的空白文档中输入下列实例中的程序。

注意:只输入左侧的程序部分即可，右侧的“/*...*/”是对程序命令或语句的注释，程序并不执行它，如图 1.30 所示。


图 1.28　新建空白文档

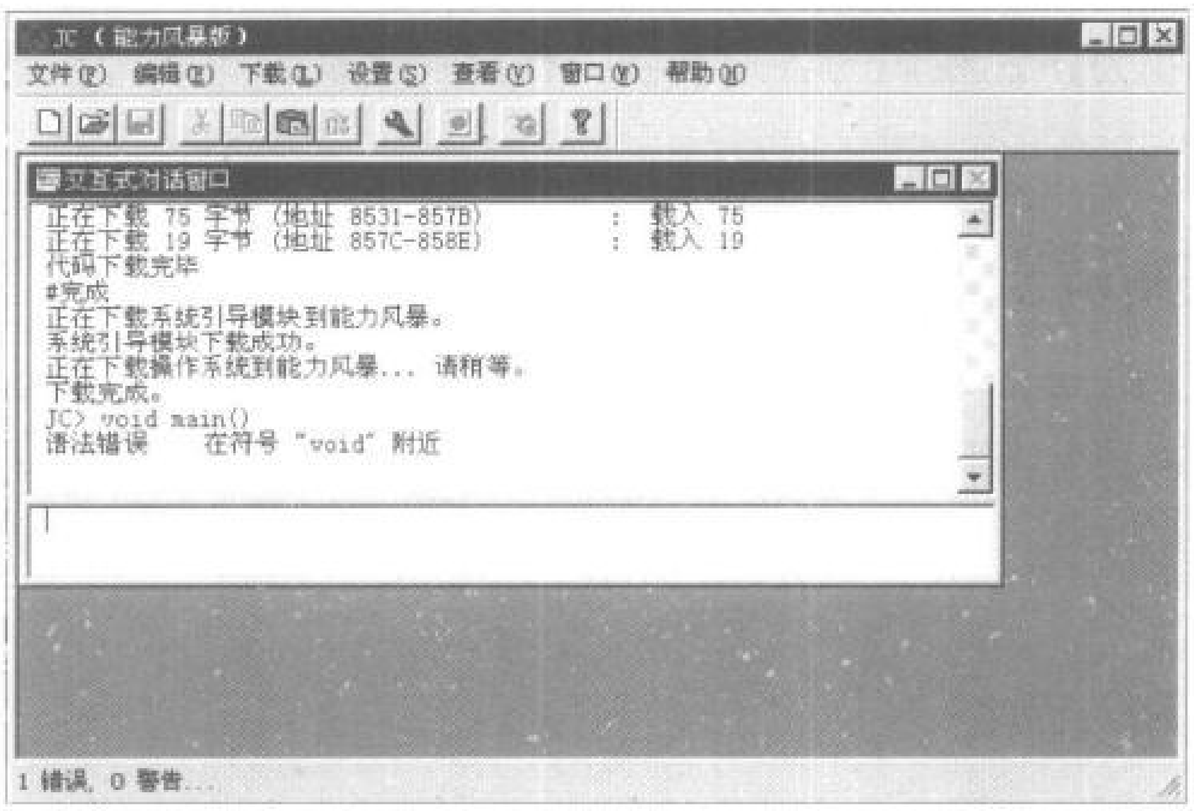



图 1.29　错误信息提示

【实例】

```
void main()              /*主程序*/
{                        /*程序起始标志*/
drive(60,0);             /*机器人往前走*/
sleep(5.0);              /*延时 5 秒钟*/
drive(-60,0);            /*机器人往后走*/
sleep(5.0);              /*等待延时 5 秒钟*/
stop();                  /*暂停运行*/
}                        /*程序结束标志*/
```

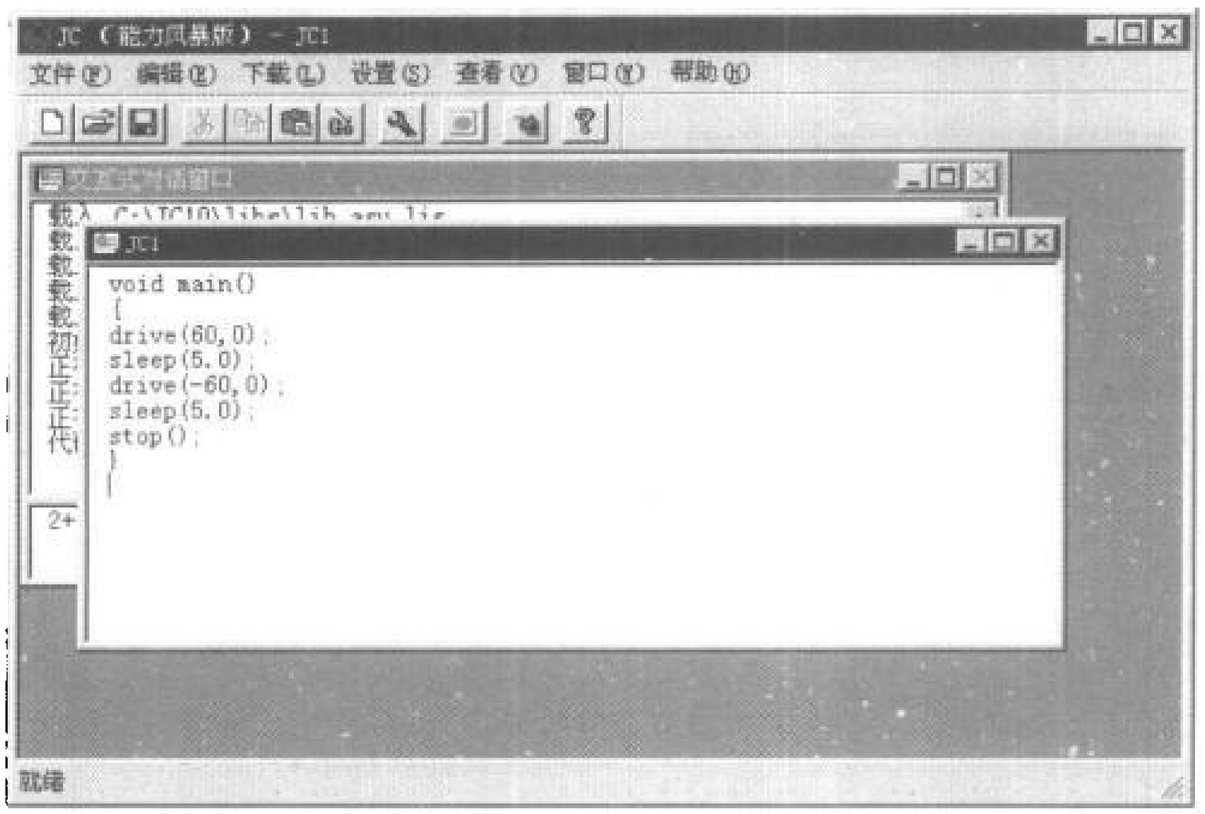



图 1.30　实例 2 程序输入窗口

程序输入完以后，你先检查一下，看看是否有输入错误的地方，如果没有就可以保存程序了。

3．保存程序

程序输入完以后，需要把它保存起来，方法如下：

在 JC 窗口中单击“文件”菜单，单击“保存”选项或直接单击图标，我们把程序保存在“新建机器人程序文件夹”中，我们给这个实例的程序起个名叫“进进退退”。

如果你没有保存程序，而直接进行其他操作，JC 窗口会提醒你保存程序，然后才能往下进行。由此可以看出，JC 窗口给我们的帮助还是非常大的。

我们已经把程序保存在计算机当中了，现在我们要让机器人按照输入的程序去工作，那么怎样把程序传给机器人呢？

## 1.3.2　为机器人下载程序

在 1.3.1 节中，为机器人输入程序，实际上是把程序输入到计算机，并保存在计算机中，需要时可以随时调用。现在我们要通过计算机把程序输入给机器人，因此我们把通过计算机将程序传给机器人的过程，叫做为机器人下载程序。

注意：在为机器人下载程序之前，你必须先把信号连接线插在机器人的通信串口上，然后将机器人开关拨到“运行”位置。在做好这两项工作之后，你才能为机器人下载程序。

现在我们为机器人下载实例 2 中的程序，方法如下：

1．下载程序

如果程序已经显示在 JC 窗口中，如图 1.31 所示，我们就可以开始下载了。

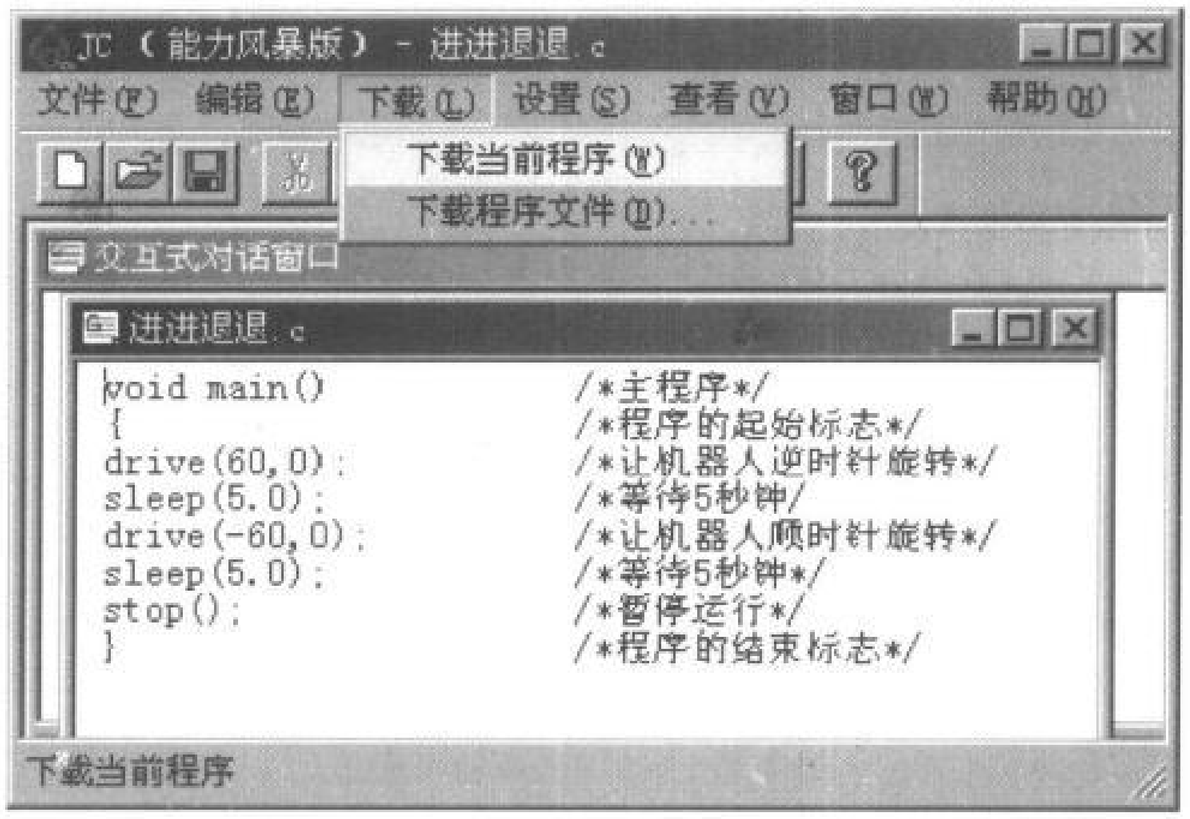



图 1.31　进进退退程序窗口

请你在 JC 窗口菜单栏中单击“下载”菜单，然后单击“下载当前程序”选项或直接单击[redacted]图标，此时程序就会被下载到机器人当中。在下载过程中你可以看到机器人控制板上的黄色发光二极管在闪动，表示正在传递信息。

如果下载程序成功，JC 窗口会出现如图 1.32 所示的提示。这时你可以继续往下操作。

如果下载程序失败，JC 窗口就会出现如图 1.33 所示的提示，告诉你问题出在什么地方，你可以在修改程序之后，再继续下载，直到出现如图 1.32 所示的提示为止。

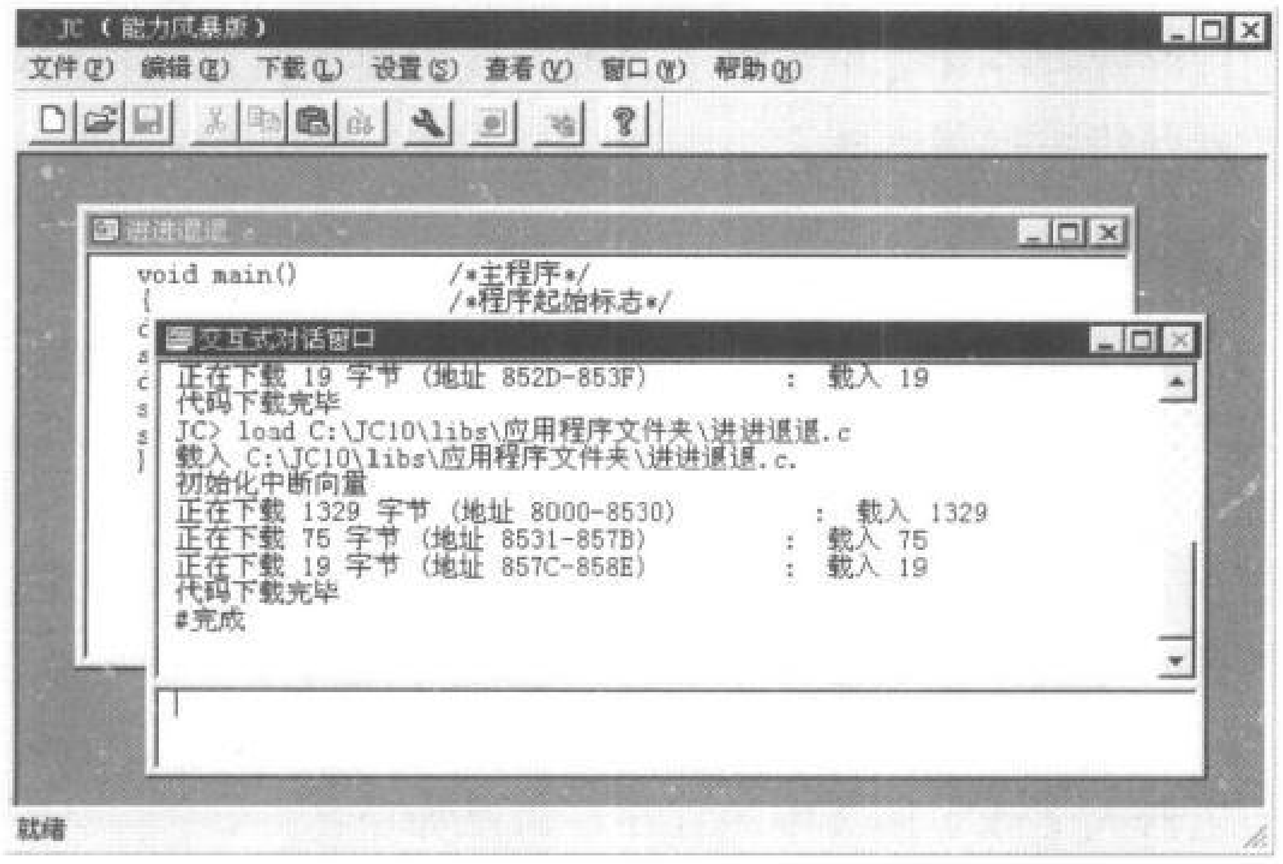



图 1.32　下载程序成功提示

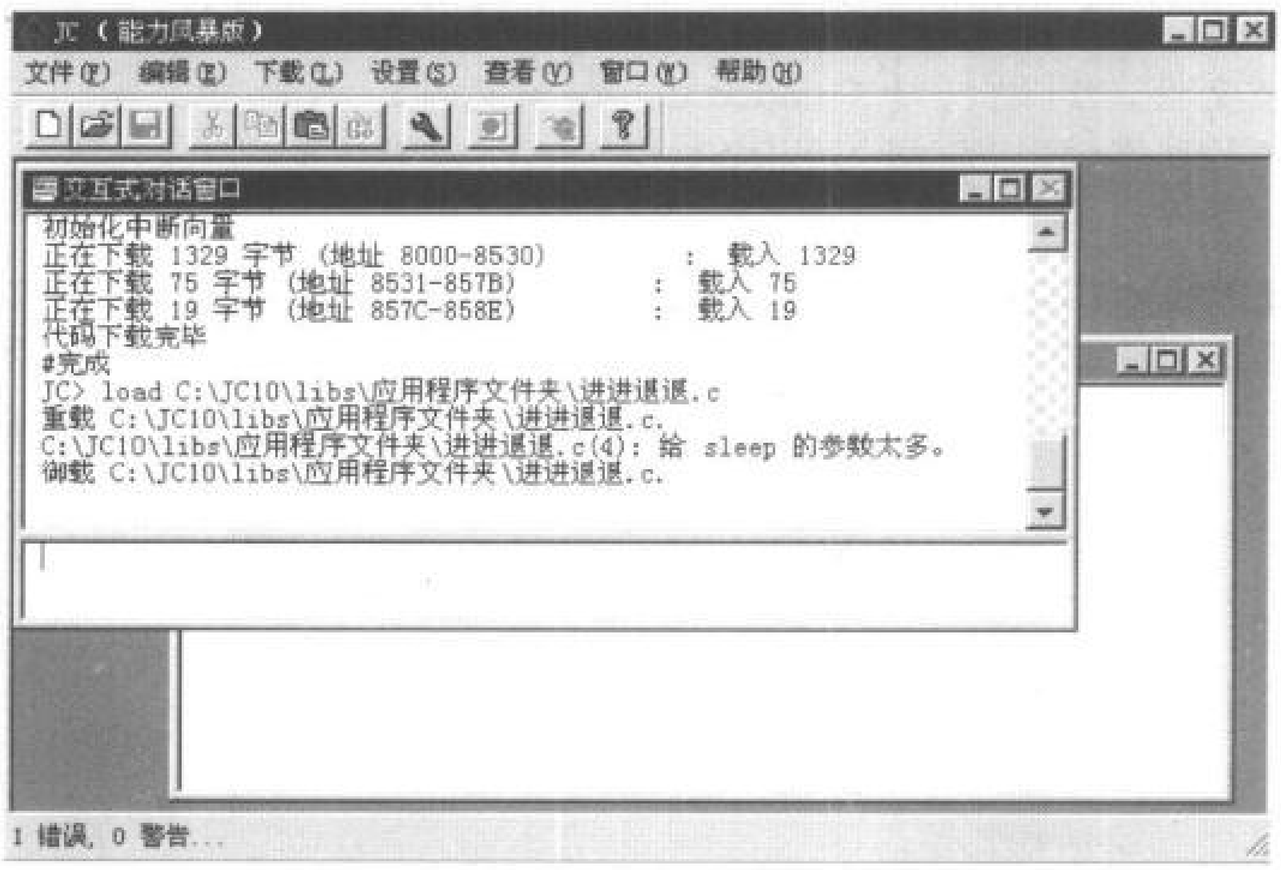



图 1.33　下载程序失败提示

2．运行程序

运行程序也是检验程序、调试程序的过程。

现在从机器人的通信串口拔下串口线，把机器人放在地上，然后按一下机器人操作面板上的“复位”键，机器人就开始按程序步骤执行动作了。

请注意观察机器人是怎样运动的。

如果你还想再观察机器人的动作，只要再按一下机器人操作板上的“复位”键，机器人就会再次执行程序。因为程序装载后，可以一直保留在机器人的内存中，直到你通过 JC 窗口卸载该程序或为机器人下载新程序，此程序才不再保留。

程序运行后机器人在做什么动作？请你记录观察结果。

通过观察记录你判断一下，程序中的哪个命令对应机器人的哪个动作？请你把思考答案写出来。

【练习 1】　修改实例 2 中的程序，将 drive(60,0)中的 60 改为 90 后，重新下载程序，在执行程序时观察机器人的动作将发生什么变化，并记录变化情况。

【练习 2】　修改实例 2 中的程序，将 sleep(5.0)中的 5.0 改为 2.0 后，重新下载程序，在执行时观察机器人的动作的变化情况，并记录变化情况。

3．进一步掌握为机器人下载程序的方法

前两节我们是先为机器人输入程序，保存程序，然后才下载程序。这一节我们将通过文件夹，选择要下载的程序。请你跟我一起完成下面的任务。

（1）打开文件夹：在 JC 窗口文件菜单栏中选择“文件”菜单，点击“打开”命令或单击 图标。

（2）选择文件：在打开的 JC 对话框中，选择“Jc10”文件夹→“libs”文件夹→“dance.c”程序文件，然后打开“dance.c”程序文件，于是 JC 窗口中出现了一个名为“dance.c”的文件窗口，如图 1.34 所示。

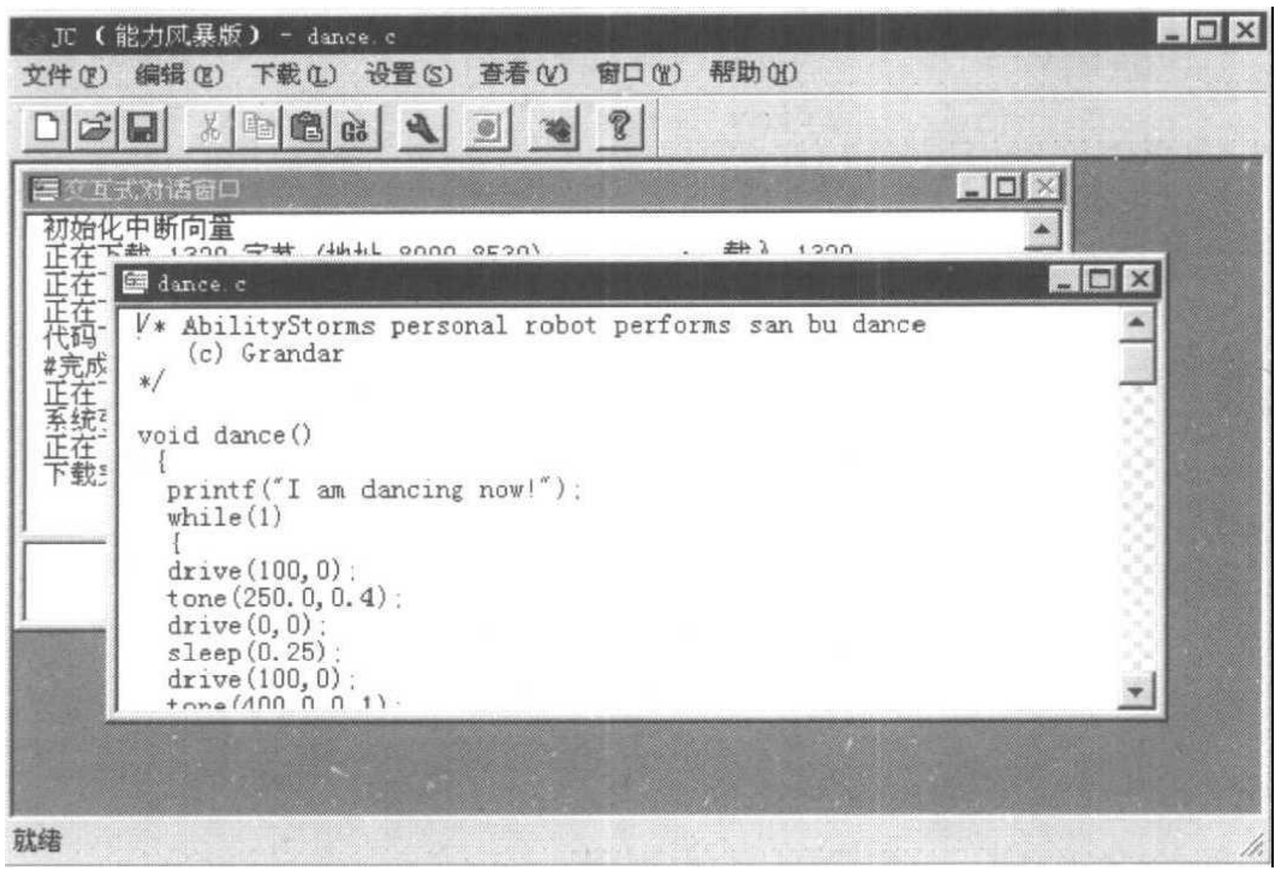



图 1.34　程序文件窗口

（3）下载程序：先将串口线插在机器人的通信串口上，把机器人的电源开关拨到“运行”位置，然后单击图标，“dance.c”程序就会被下载到机器人当中。

如果出现如图 1.35 所示的情况，说明机器人已被下载程序占用，请你按以下 3 种不同的方法卸载机器人当中的文件。

- 直接在交互式对话窗口的命令编辑窗口中输入语句“unload <程序名>.C”。
- 用重载操作系统来卸载机器人当中的程序。
- 重新启动 JC1.0，刷新机器人内存以卸载机器人的程序。

以上 3 种方法，你都可以试一下。

有关卸载方法的详细说明将在本章 1.5 节中介绍。

（4）执行程序：从机器人操作面板上拔下串口线，按一下机器人的“复位”键，机器人就开始动作了。现在你的机器人正在给你跳舞，是吗？

至此我们完成了为机器人下载程序的全过程。

通过这一节，我们学会了为机器人下载程序的两种方法：

- 先输入程序，然后保存和下载程序，最后执行程序。
- 在文件夹中选择程序并打开，下载程序，执行程序。

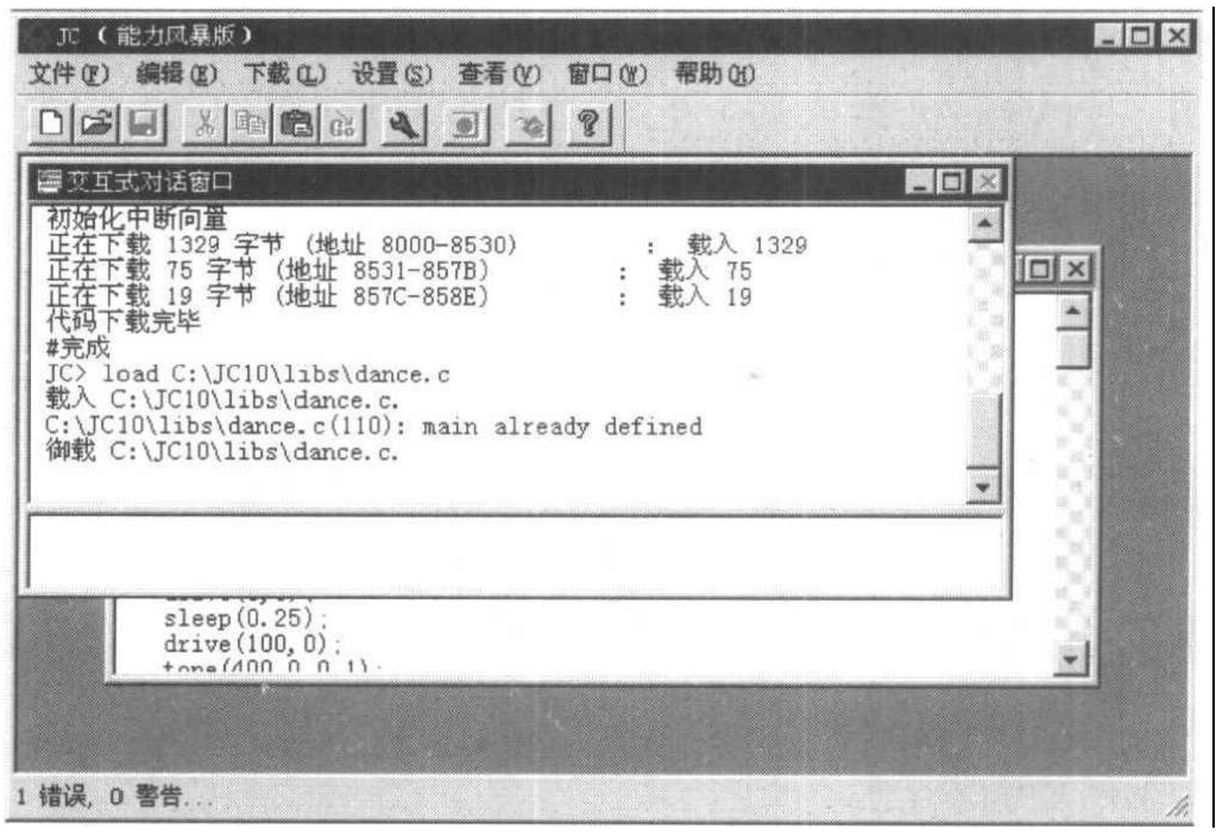



图 1.35　下载 dance.c 错误提示

### 1.3.3　习题

现在我们按照前面的步骤和方法试着做一做下面的事情。

1．寻找文件或程序：在 JC 窗口打开文件，点击“libs”文件夹，找到“dance.c”文件，并打开“dance.c”文件。

2．下载程序：将串口线插在机器人的通信接口上，机器人的开关拨到“运行”位置，在“下载”菜单中点击“下载当前程序”命令或直接点击图标，这时已经编写好的程序就被下载到机器人中了。下载时你可以看到机器人主板上的黄色发光二极管在闪动，说明机器人正在接收信息。

3．执行程序：拔下机器人通信接口的连线，按一下机器人操作面板上的“复位”键，机器人便开始动了。

请你仔细观察机器人的动作，并把机器人的动作状态一一记录下来。

请你想一想以下两个问题：

（1）机器人为什么可以原地旋转？

（2）怎样控制机器人的行进距离？

## 1.4　为你的机器人“体检”

当你得到一个机器人的时候，它像一个刚出生的孩子，需要教它识别光线、物体，教它走路，教它躲避障碍等等，你不仅要告诉你的机器人去做什么事情，还要告诉它

怎样做，这一切都要通过一种语言来实现，这种语言就是 JC。当你用 JC 为机器人编写好任务程序，并“告诉”机器人后，它就会按照程序规定的步骤去工作了。

为了使你的机器人伙伴更快、更好地成长，我们先来给机器人伙伴做一次全面的“体检”。

### 1.4.1 准备工作

现在让我们按照下载程序的 3 个步骤进行：

1．打开 JC 文件窗口，选择“libs”文件夹，点击“ASU_selftese.lis”程序；

2．在机器人的通信端口插上串口线，把其开关拨到“运行”位置，开始下载“ASU_selftese.lis”程序；

3．拔下信号线，运行程序。

在运行程序之前，LCD 显示屏如图 1.36 所示。


图 1.36　LCD 示意图

现在，你可以开始为机器人“体检”了。

### 1.4.2 LCD 显示屏

我们前面介绍了 LCD 液晶显示屏，它装在机器人的外壳上，作用是显示各种信息，便于你了解程序执行中的情况。这个显示屏可以显示 2×16 个字符，机器人与你的交流可以通过这个显示屏反映。LCD 显示屏如图 1.37 所示。

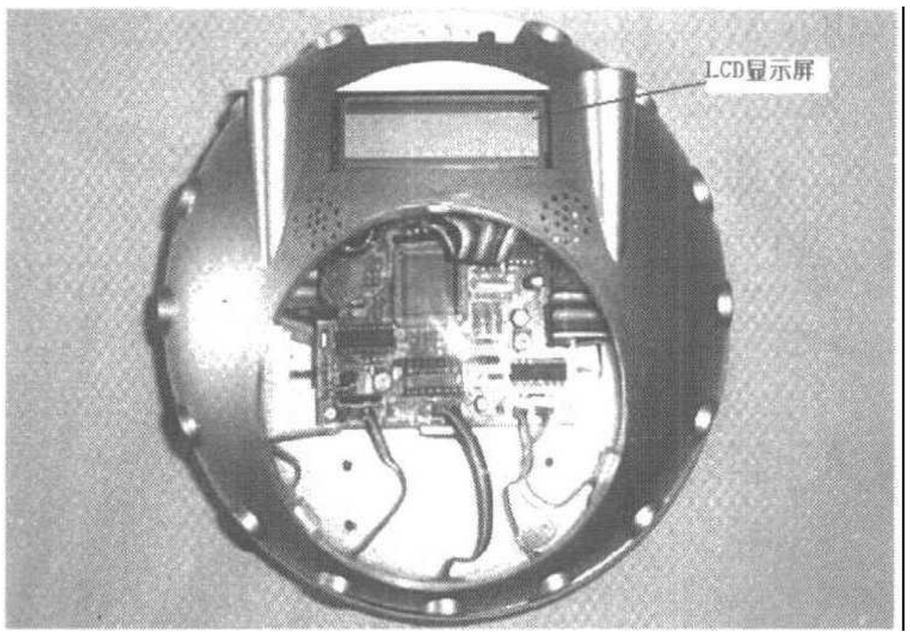



图 1.37　LCD 显示屏

现在请你按一下机器人控制板上的“复位”键，观察 LCD 显示屏。显示屏上的内容如图 1.38 所示。

LCD 显示屏出现“LCD Test”提示，然后就开始显示字符，如数字、符号、字母等。

观察机器人的 LCD 显示屏有没有出现黑屏或无字符等现象。如果没有，说明机器人的 LCD 显示屏是正常的。

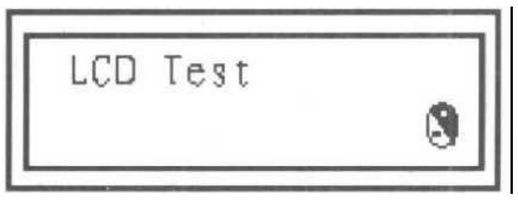



图 1.38 LCD Test 示意图

### 1.4.3 “发声”检测

能力风暴个人机器人用喇叭发声，可以说喇叭就是机器人的“嘴巴”，它装在主板上。在你需要机器人用“嘴巴”与你交流时，机器人会按照你的要求去做。机器人的“嘴巴”在如图 1.39 所示的位置。

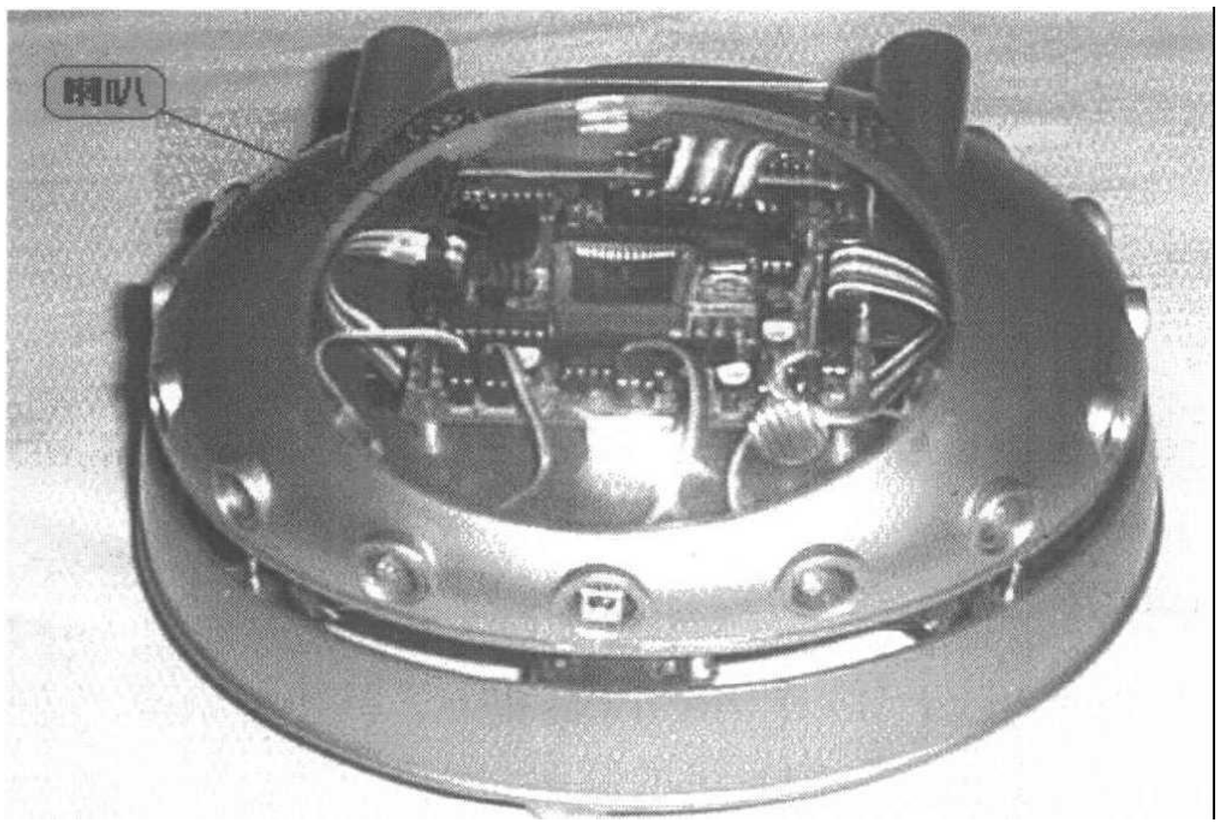



图 1.39 喇叭位置图

现在按一下“复位”键，机器人就开始“唱歌”了。注意 LCD 显示屏出现了变化，如图 1.40 所示。

在图 1.40 中，括号内的数字随着机器人的声音在不断变化，显示的数字是机器人发出的声音频率。

请注意听机器人是否在给你“唱歌”，声音是否清晰响亮。

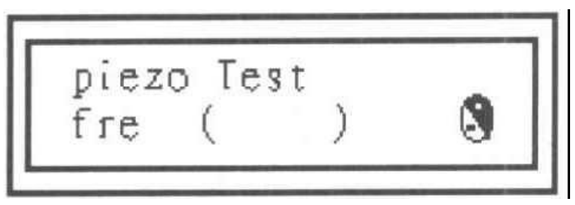



图 1.40　Piezo Test 示意图

### 1.4.4　“视力”检测

机器人的视觉系统是由光敏传感器和红外传感器两部分组成的，相当于人的眼睛。现在我们一起来检测一下你的机器人的“眼睛”。

1．光敏检测

机器人的光敏传感器安装在外壳上，如图 1.41 所示。两只光敏传感器一左一右，它的作用是识别来自外界的光线以及光线的强弱。

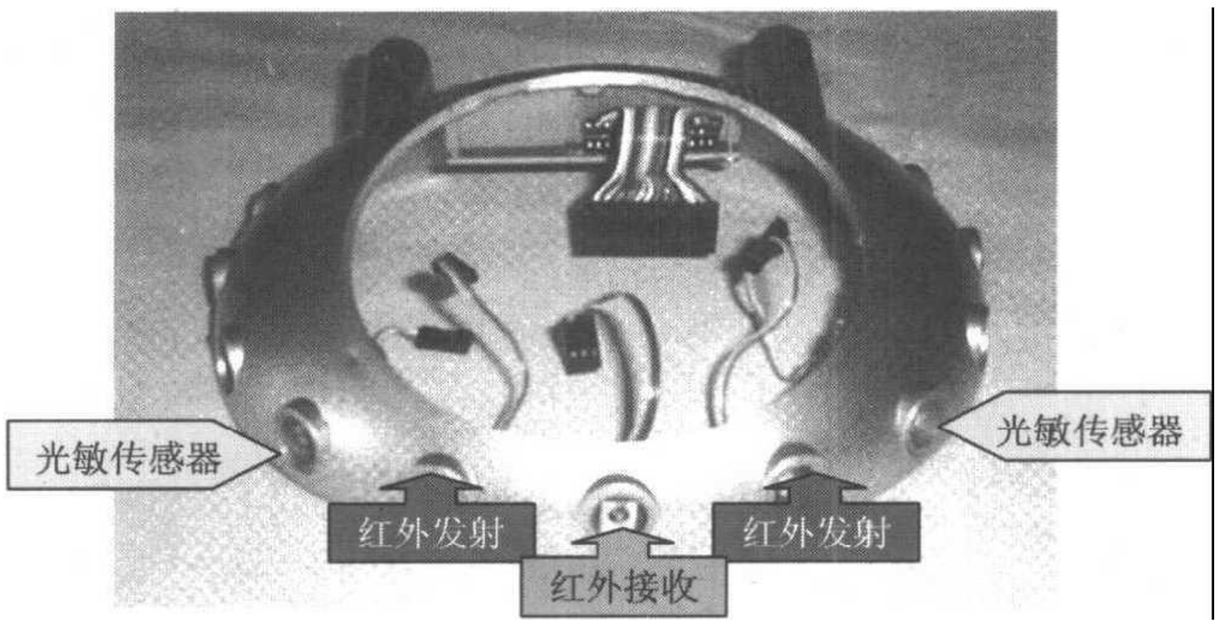



图 1.41　“视觉”传感器

现在请再按一下“复位”键，仔细观察 LCD 显示屏，如图 1.42 所示。

LCD 显示屏出现“Photo　L($x$)　R($y$)”的提示，其中 L 表示“左眼”，R 表示“右眼”，$x$ 和 $y$ 的值表示照射机器人的光线的强弱。

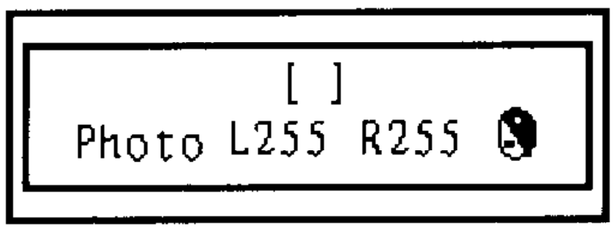
经过观察，我们发现：

（1）当同时遮住左右光敏传感器时，LCD 显示屏的状况如图 1.42(a) 所示。此时 L 和 R 的值都为 255。这说明如果左右光强一样时 L 和 R 的值相等。

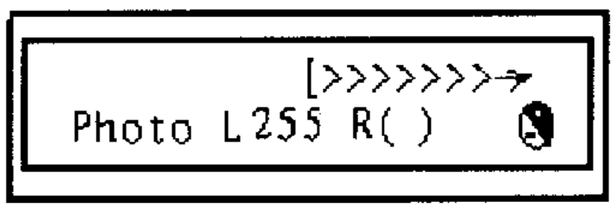
（2）当用手遮住左侧的光敏传感器时，LCD 显示屏的状况如图 1.42(b) 所示。L 的值为 255，R 的值随着光线的强弱不断变化。

（3）当用手遮住右侧的光敏传感器时，LCD 显示屏的状况如图 1.42(c) 所示。这

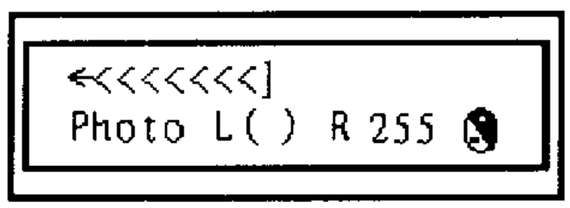
时 R 的值为 255，L 的值在不断变化。




(a)




(b)




(c)

图 1.42　光敏检测示意图

【练习】

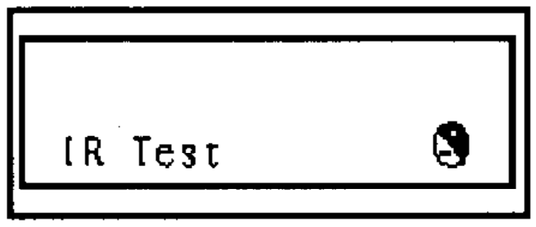
（1）光线越强，L 和 R 的数值越_____；光线越弱，L 和 R 的数值越_____。

（2）当 L 值大于 R 值时，你能判断哪一侧的光强？左侧还是右侧？反之呢？

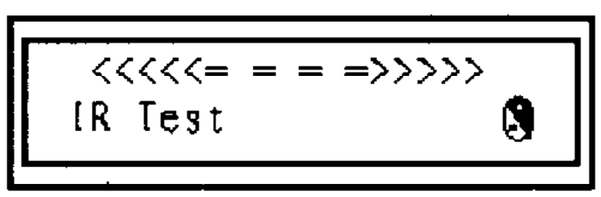
（3）当 L 值等于 R 值时，说明什么？

2．红外检测

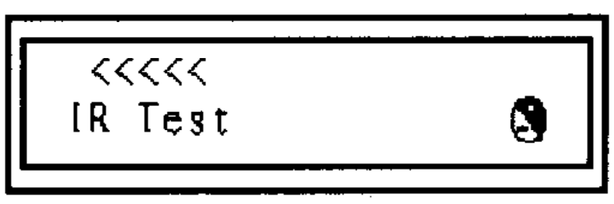
红外传感系统也安装在机器人的外壳上，如图 1.41 所示，左右两侧一边一只红外发射管，中间一只红外接收模块，红外传感器的作用是检测机器人前方、左前方、右前方是否有障碍物。好了，你再按一下“复位”键，看 LCD 显示屏已经出现了“IR Test”的提示，如图 1.43(a)所示。



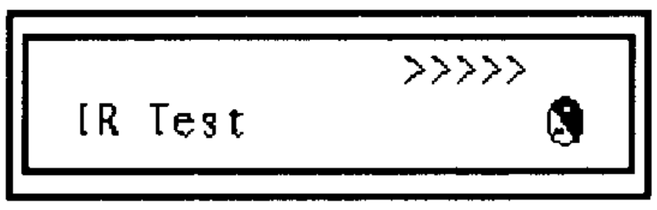

(a)




(b)




(c)




(d)

图 1.43　红外检测示意图

现在你可以试试了。注意检测距离在 30cm 左右，被检测物体不要太矮，因为机

器人的“视角”没有那么广：

（1）LCD 显示屏如图 1.43(a) 所示时，表示前方没有障碍物。

（2）当物体出现在机器人的正前方时，LCD 显示屏如图 1.43(b) 所示，表示前方有障碍。

（3）当你将物体放在机器人的左侧时，LCD 显示屏如图 1.43(c) 所示，表示左侧有障碍。

（4）当你将物体放在机器人的右侧时，LCD 显示屏如图 1.43(d) 所示，表示右侧有障碍。

现在你的机器人既能识别光又能检测前方是否有障碍物，可见你的机器人“视力”是正常的。

### 1.4.5 “听力”检测

能力风暴个人机器人可以通过一只麦克风识别简单的语音命令，这只麦克风就相当于人的耳朵，如图 1.44 所示。

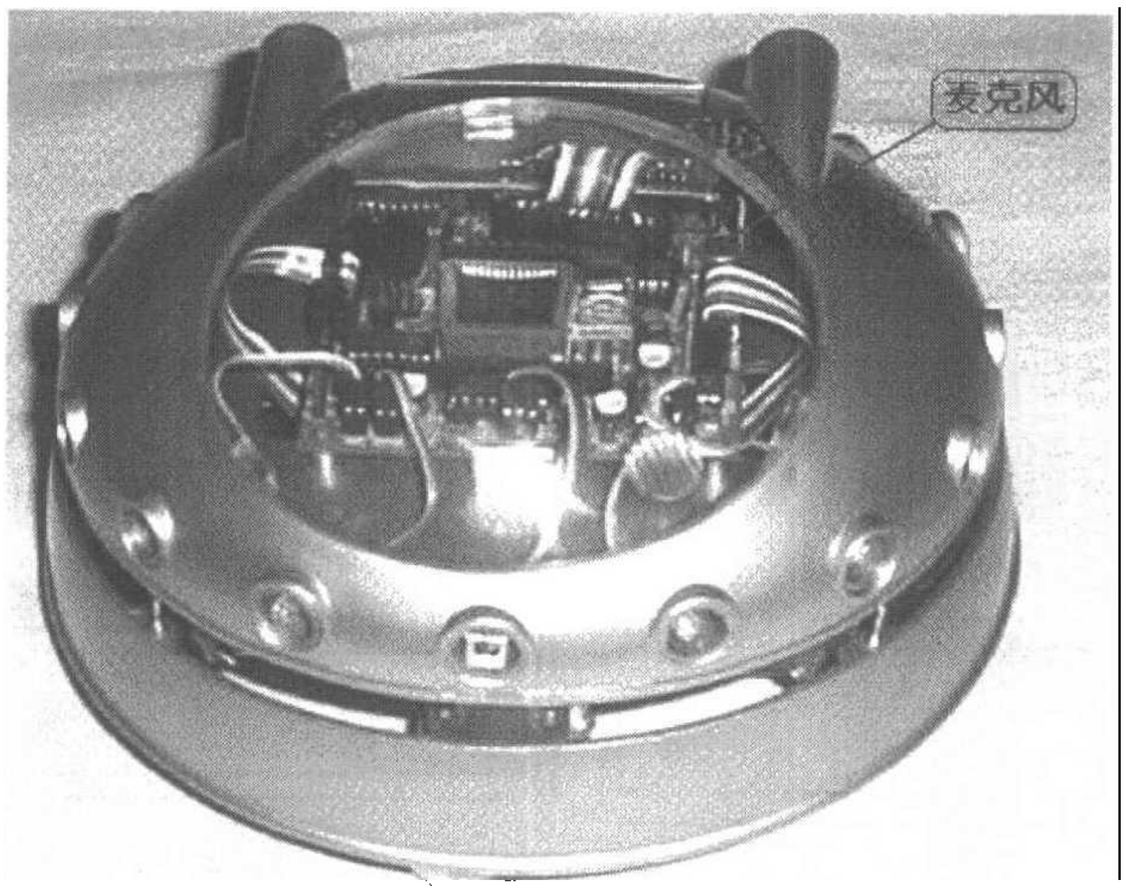



图 1.44 麦克风位置

按一下“复位”键，LCD 显示屏就会出现如图 1.45 所示的提示。你对着麦克风说话、拍手、吹气，就会看到机器人“听到”声音的反应是 LCD 上显示的“>”符号在增加。

你的机器人“听”到了吗？

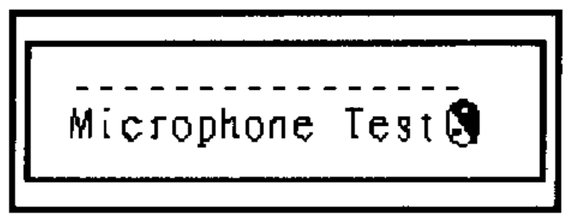



图 1.45 Microphone Test 示意图

### 1.4.6 “触觉”检测

能力风暴个人机器人的碰撞传感器，就相当于我们人类感知外部信息的触觉。它虽然只用了四只碰撞传感器，然而设计巧妙的碰撞环，使机器人具备了感知全身碰撞的能力。现在看看你的机器人能否感知 8 个方向的碰撞。

请你按一下“复位”键，LCD 显示屏出现如图 1.46 所示的字符，“none”表示没有碰撞。

这时你可以用手从不同方向触动碰撞环，LCD 显示屏就会相应地显示出机器人受到碰撞的方向：

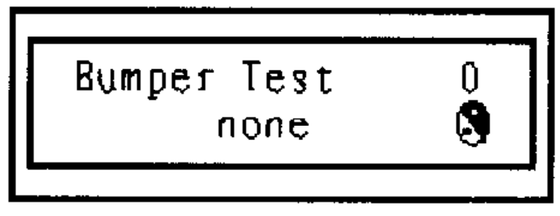



图 1.46 Bumper Test 示意图

1. 用手从正前方压碰撞环，LCD 显示屏上显示“Front”；
2. 用手从正后方压碰撞环，LCD 显示屏上显示“Back”；
3. 用手从左侧压碰撞环，LCD 显示屏上显示“Left”；
4. 用手从右侧压碰撞环，LCD 显示屏上显示“Right”。

### 1.4.7 运动检测

机器人“走路”是靠电机带动轮子实现的。我们先检测光电编码器，再检测电机。

1. 按一下“复位”键，机器人的 LCD 显示屏出现“Encoders Test”提示如图 1.47 所示。

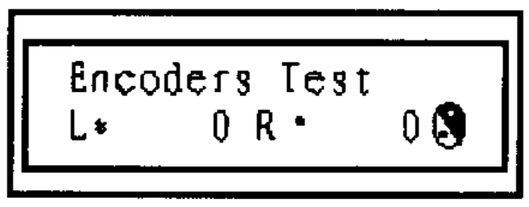



图 1.47 Encoders Test 示意图

你再拿起机器人，在其中一个轮子上做个标记，然后用手转动轮子，正好一圈时，你看 LCD 显示屏上 L 或 R 的值是多少。我们前面介绍过，轮子转一圈是 33 个脉冲，当 LCD 显示值为 33 时，说明轮子转了一周。

请你分别检测左右轮子的光电编码器。

2．现在我们来做最后一项检测，按一下“复位”键，机器人 LCD 显示屏出现的提示如图 1.48 所示。

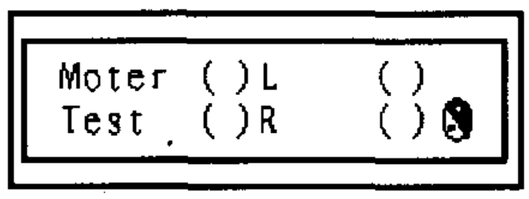



图 1.48　Moter Test 示意图

请观察并记录机器人的运动状况和 L、R 的数值变化情况。

你可以先让机器人在地上跑，观察记录机器人的运动状况后，再拿起机器人观察 LCD 显示屏的数值变化。

如果通过了以上 6 个系统和 8 个项目的检测，则表示能力风暴个人机器人的“身体”是健康的，现在你可以教它本领了。

## 1.5　排除故障

初次使用和接触机器人，由于对操作方法不熟练以及迫切的心情，在操纵中可能会遇到各种问题，请你不要着急，按照操作方法和步骤再认真做一次。如果问题依然出现，请参照以下方法解决。

### 1.5.1　电源打开时机器人无反应

当把机器人的电源开关拨到“运行”位置时，应听到“嘟”一声响，同时 LCD 显示屏将出现如图 1.49 所示“JC V1.0 Grandar Ability Storms”和一个跳动的太极图，此时表示操作系统已经可以正常运行。


图 1.49　开机示意图

开关拨到“运行”位置后，如果没有前面提示的内容，则请你按以下方法检查并解决。

1．检查控制板上绿色发光二极管和红色发光二极管亮不亮，如果不亮，请检查电源。把电池拿下来，重新安装。发光二极管仍然不亮，说明电池没电，请拿出智能充电器为机器人充电。一般 4 小时充满。

2．如果控制板上绿色发光二极管和红色发光二极管都亮，表示电压不足，可以充电；只有红色发光二极管亮，表示机器人需要充电。

3．如果绿色发光二极管亮，LCD 没有显示，可能是操作系统丢失，请按照 1.2.2 软件准备一节中讲述的方法，为机器人下载操作系统。

发光管位置如图 1.50 所示。

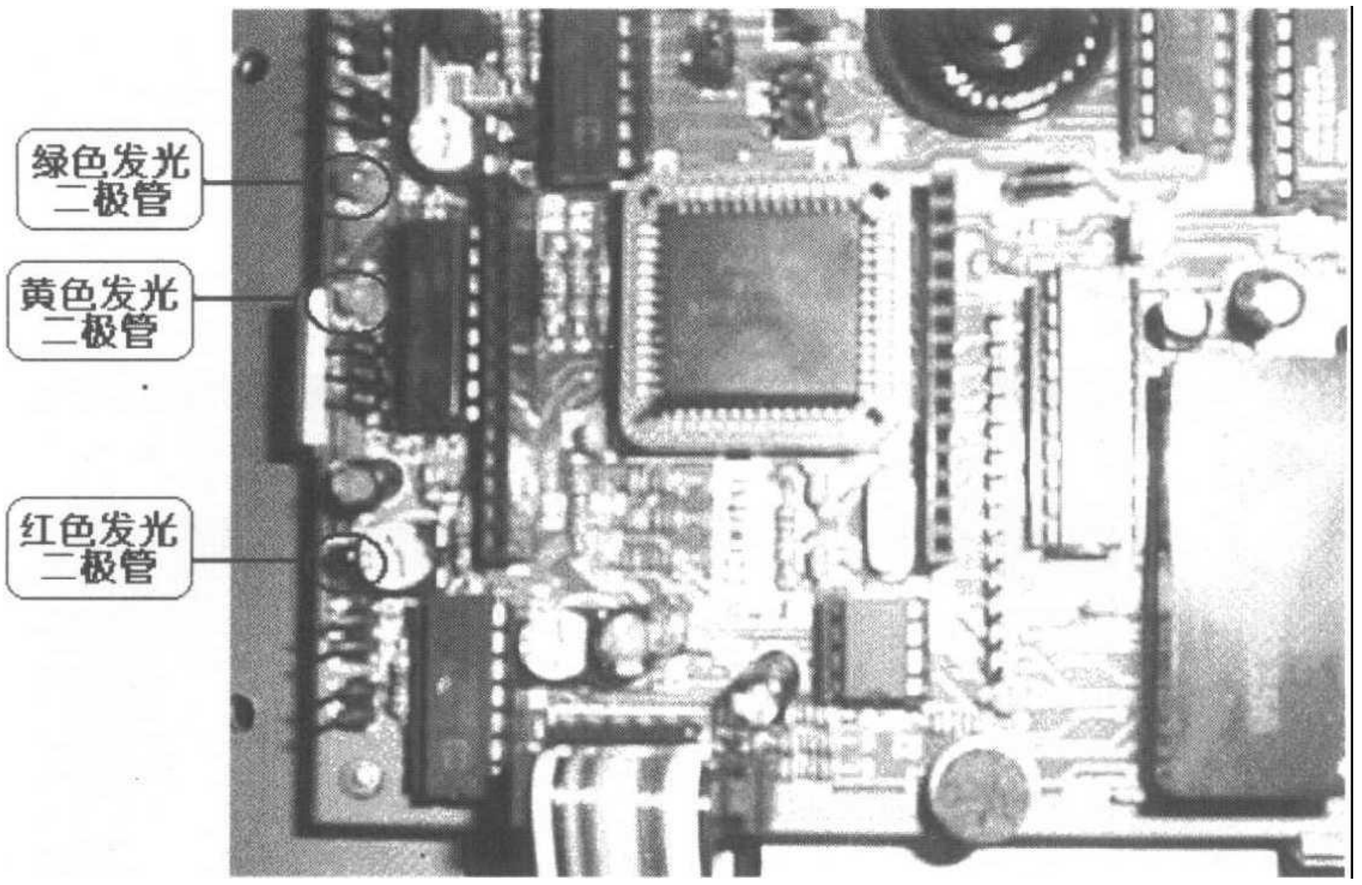



图 1.50　发光管位置图

### 1.5.2　卸载程序的方法

在为机器人下载程序之前，如果机器人已有程序，必须先卸载。为机器人卸载程序一共有 3 种方法：

1．单击 JC 窗口，然后在命令行编辑器中输入“unload <程序名>.c”，如图 1.51 所示。按回车键之后，你可以看到 JC 窗口出现卸载程序的提示，如图 1.52 所示。

2．如果你不知道机器人已装载的程序名，可以在 JC 窗口中单击“设置”菜单，然后单击“重载操作系统”选项，或直接单击图标，在随后出现的机器人控制板设置窗口，单击“下载操作系统”按钮，这样也可以卸载机器人当中的程序。

3．如果前两种方法都不能卸载机器人当中现有的程序，你就必须重新进入 JC1.0

系统给机器人进行初始化，最终达到卸载程序的目的。

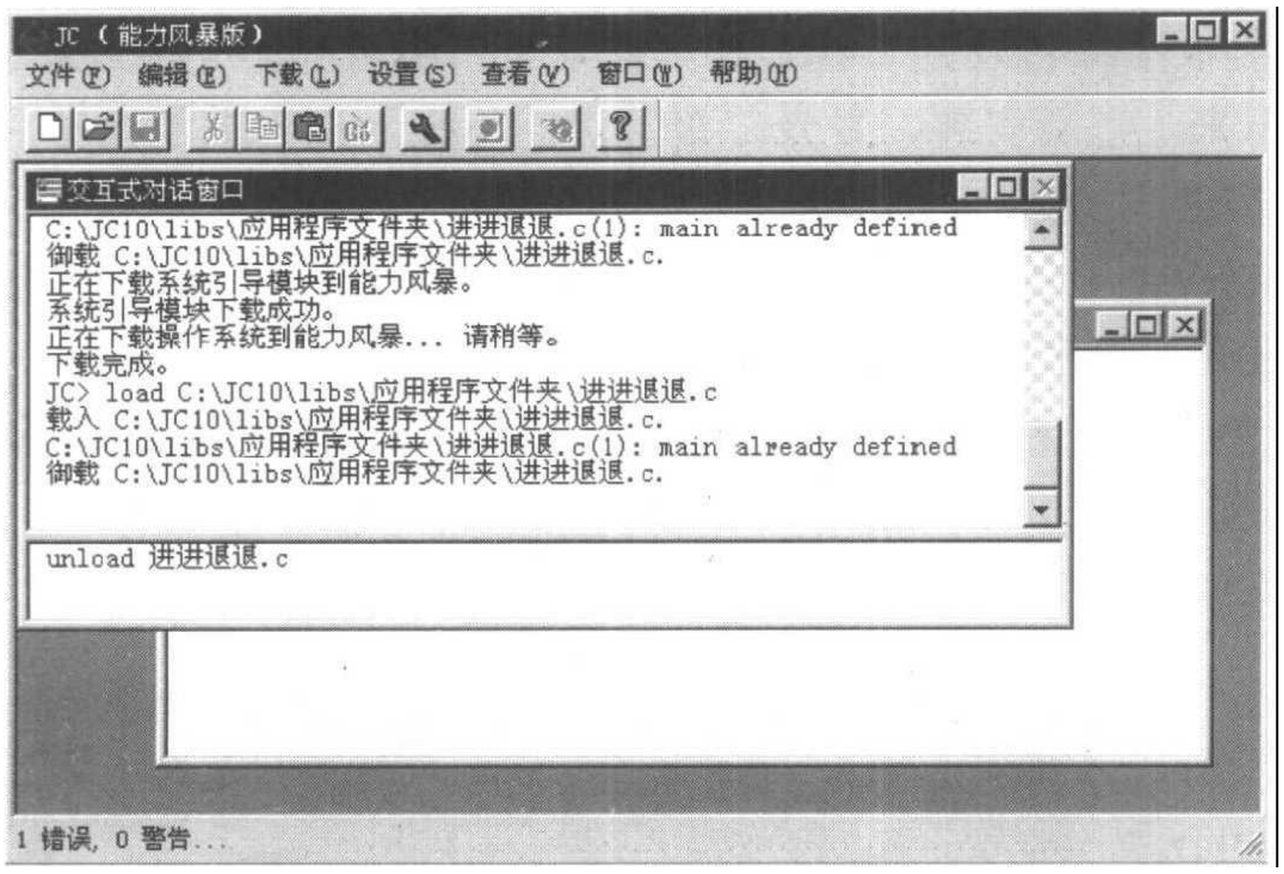



图 1.51　在命令行输入卸载命令

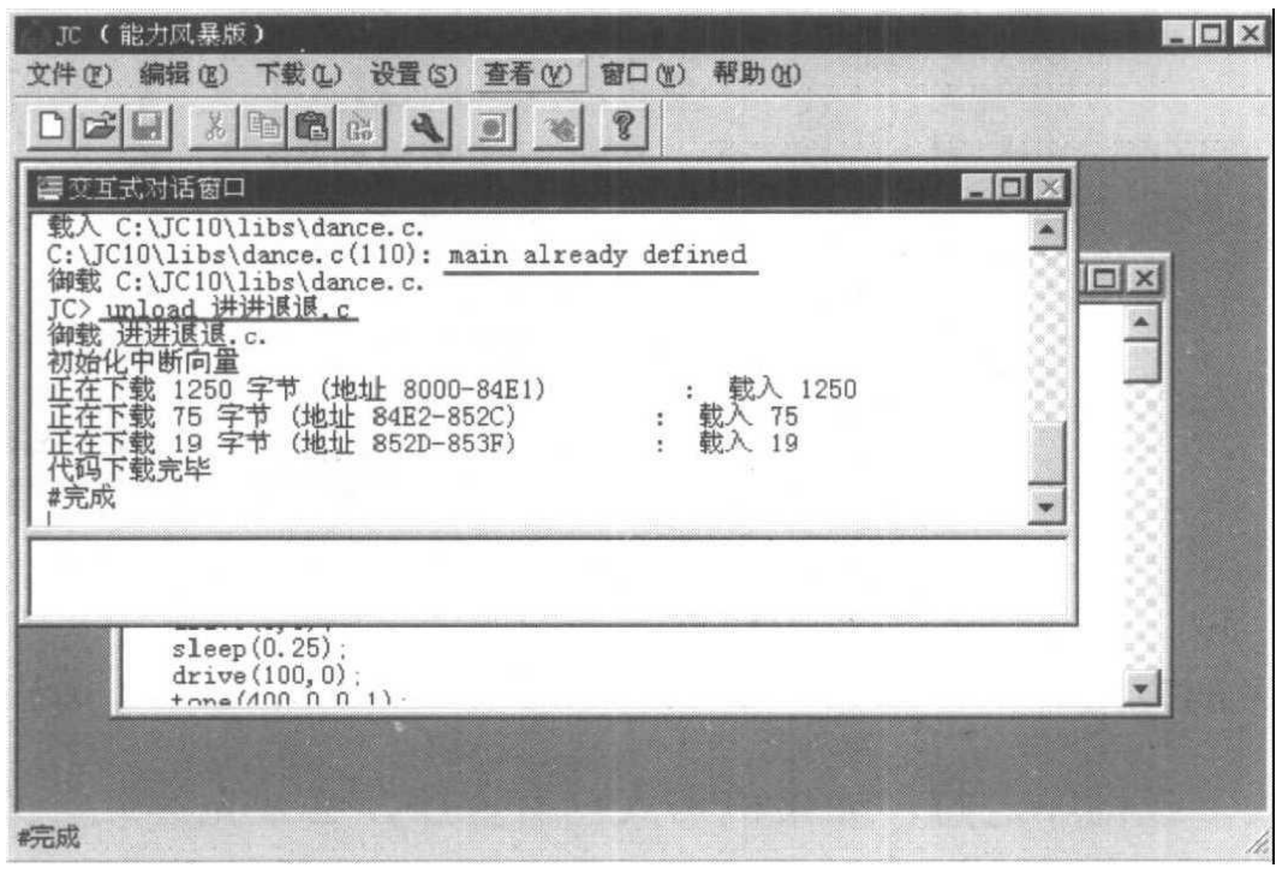



图 1.52　执行卸载命令

对于以上 3 种方法，你可以根据不同的情况选择使用。

### 1.5.3　其他常见问题

1. LCD 显示屏出现缺行、缺字、字符混乱的现象，通常是 LCD 显示板接触不良，将 LCD 显示板的插针拔下来重新插好。

2．当机器人的电池不足、电压过低时，就会出现以下几种现象：

（1）控制板前方红灯闪烁；

（2）能力风暴个人机器人程序意外丢失；

（3）机器人走走停停；

（4）机器人无故原地转圈 。

3．当机器人失去控制，LCD 又显示“run time error XX”时，可能是用户程序有错误，你可以用 JC 帮助里的“运行错误”一项进行检查，并按照提示修改程序。

在不知不觉中，我们已经全面了解了机器人的基本结构，以及机器人的各部分名称和作用，学会了为机器人输入程序和下载程序的基本操作方法。你是不是对机器人产生了兴趣，并希望能对它有进一步的了解。

现在我们开始教你的机器人学习本领。

# 第2章　教教你的机器人

你已经为你的机器人伙伴做了全面的“体格检查”。如果它“身体”健康，那么你现在就可以教它更多的本领，赋予它智慧，使它更加“聪明”。

本章我们在教你的机器人“学习”本领的同时，要掌握“培养”机器人能力的方法，并了解为机器人编写程序用到的语句和函数。在学习中我们要勇于实践、注意观察、勤于思考。

## 2.1　教你的机器人“走路”

能力风暴个人机器人是可移动的智能机器人，它的移动就相当于我们人的行走。你可能会问：机器人没有“脚”靠什么走路？这一节我们要熟悉机器人用于走路的“脚”，要教你的机器人学会走路，同时你要掌握控制机器人走路的基本方法。

### 2.1.1　机器人为什么会“走”

人的行走是靠脚来实现的，而机器人的“行走”是靠由电机、齿轮箱和轮子构成的行走装置来完成的。因此，要想让机器人移动，就要控制电机的转动。控制机器人“行走”的基本指令是 motor($x$,$y$)函数和 drive($x$,$y$)函数。

1．关于机器人的“脚”

“脚”是走路的关键。用于机器人走路的“脚”，是安装在机器人底盘下部的两只驱动轮和两只随动轮，两只驱动轮的转动是由电机带动的。电机、驱动轮、齿轮等构成了机器人的行走装置。电机、驱动轮、齿轮等部件的位置如图 2.1 所示。

2．控制机器人走路

机器人的“腿脚”已经有了，显然，要让机器人的“腿脚”动起来，我们只要通过控制机器人的行走装置就可以实现了。机器人的行走装置需要它的“大脑”来支配，就如同人用大脑去支配腿脚运动一样。我们要控制机器人的行走，就要通过 JC 程序来实现，即通过控制电机的转动，使机器人产生移动。

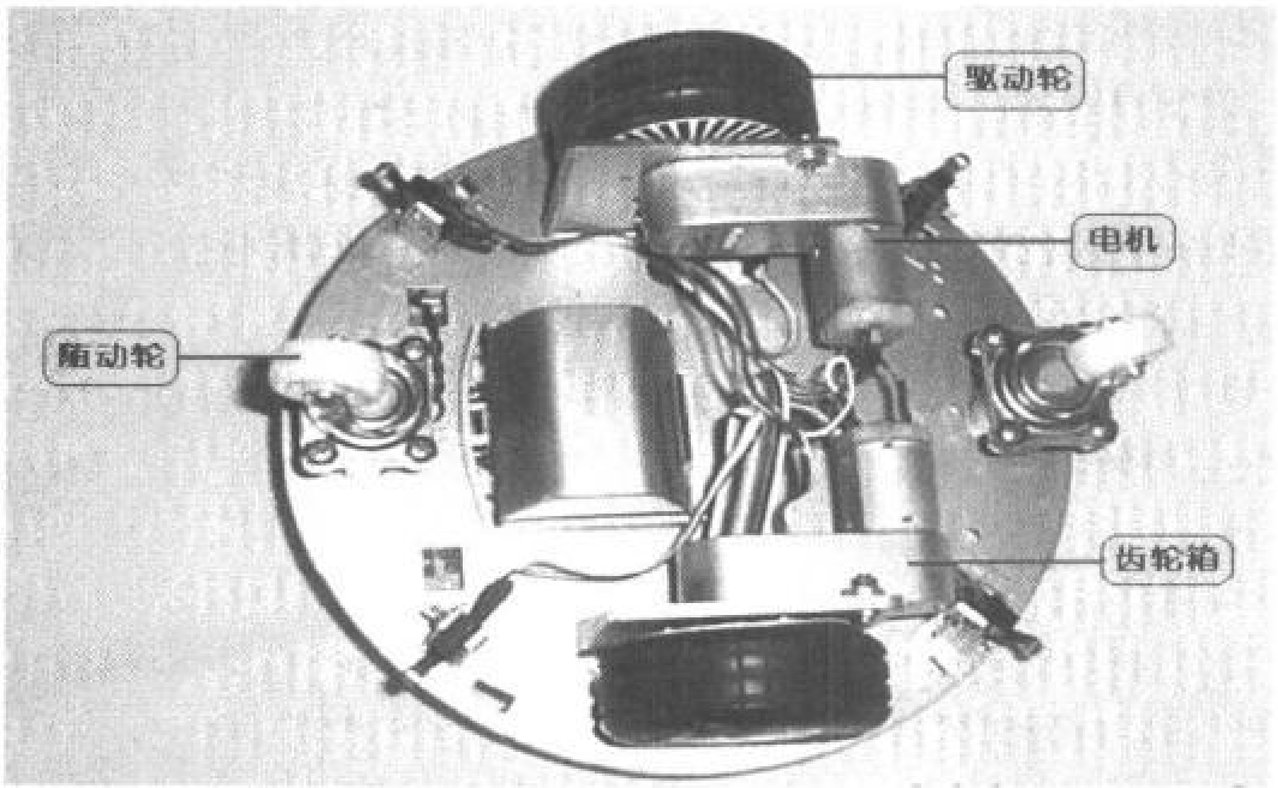



图 2.1　机器人的“脚”

### 2.1.2　驱动电机的函数

我们已经知道电机是机器人行走的关键，只要电机转动，就可以使机器人产生移动。我们通过 JC 程序控制电机转动，使机器人行走的指令有两个，它们是 motor(*x*,*y*)函数和 drive(*x*,*y*)函数，下面我们分别介绍。

1．motor(*x*,*y*)函数

现在请你接好串口线，把机器人开关拨到“运行”位置，然后运行 JC1.0，将出现如图 2.2 所示的界面。

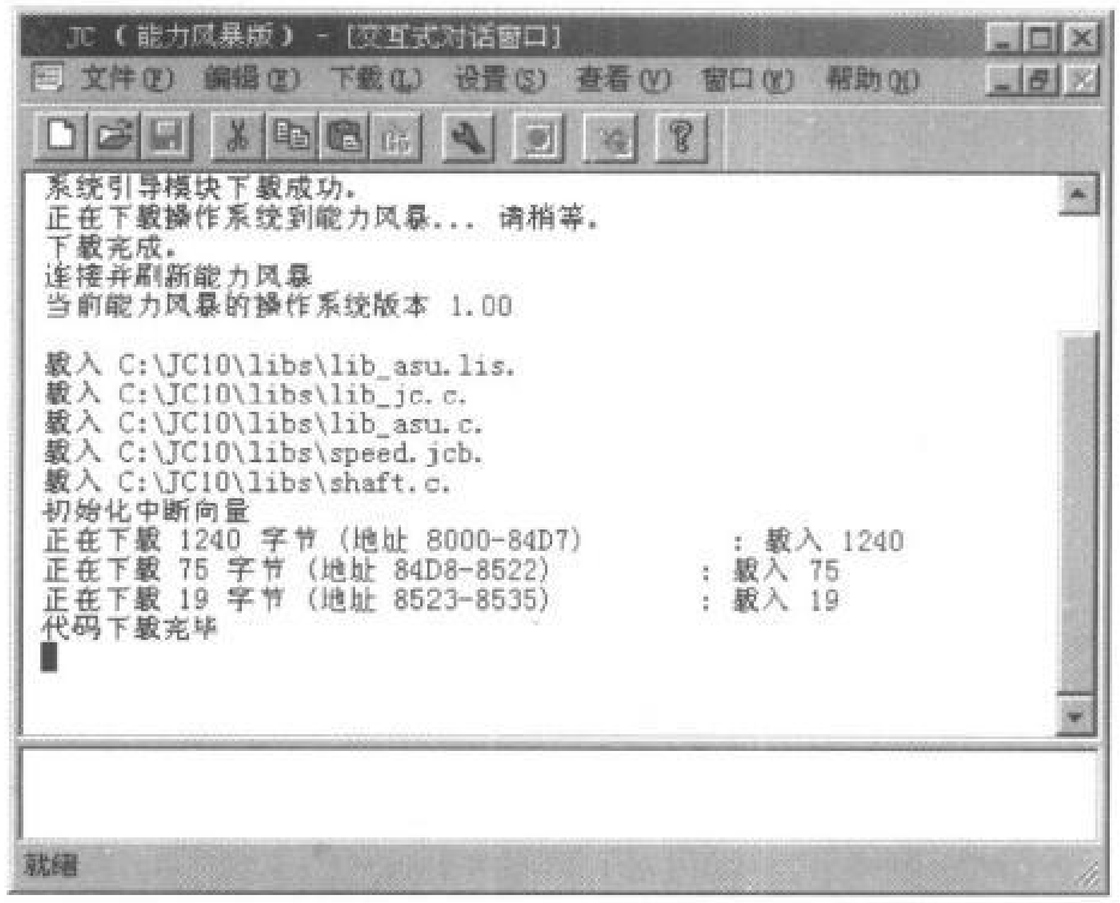



图 2.2　JC 交互式对话窗口

【实例 1】请你在窗口命令行编辑器中直接输入下面的语句。

{motor(0,40);sleep(5.0);stop();}

请你观察机器人的运动状况，并记录下来。如果你没看清楚，想再观察一次，就按“↑”键，命令行就会出现刚刚输入过的语句，再按回车键，把该语句再执行一次即可。

【实例 2】现在用同样的方法再输入语句“{motor(1,40);sleep(5.0);stop();}”。

请你记录机器人的运动状况。

请你比较：

- 实例 1 语句和实例 2 语句有什么不同？
- 实例 1 和实例 2 使机器人产生的运动有什么不同？

【实例 3】现在请你将实例 1 或实例 2 中的 40 换成 80，观察机器人的运动发生什么变化？

【实例 4】如果将实例 1 或实例 2 中的 40 换成－40，机器人将怎样运动？请注意观察机器人的运动与前面的运动有什么不同？

通过对前面 4 个实例的学习，我们可以看出：

- 当 $x$ 为 0 时，机器人的左侧电机转动，机器人往右转圈；当 $x$ 为 1 时，机器人的右侧电机转动，机器人往左转圈；可见 $x$ 是控制机器人左右两只电机工作的，就相当于指挥它的左右“脚”；
- $y$ 既可以控制机器人电机的转动速度，也可以控制机器人电机的转动方向。当 $y>0$ 时，机器人“前进”，数值越大，机器人“前进”速度越快；当 $y<0$ 时，机器人“后退”。请你记住 $y$ 的取值是从 100（正向最大）到－100（反向最大）。

【练习】新建一个空白文档，然后输入电机驱动程序。

```
void main()
{
motor(0,80);          /*左侧电机以 80 的速度启动*/
motor(1,80);          /*右侧电机以 80 的速度启动*/
sleep(2.0);           /*延时 2 秒钟*/
motor(0,0);           /*左侧电机速度为 0*/
motor(1,80);          /*右侧电机仍以 80 的速度转动*/
sleep(1.8);           /*延时 1.8 秒钟*/
motor(0,80);          /*左侧电机以 80 的速度转动*/
motor(1,80);          /*右侧电机继续以 80 的速度转动*/
sleep(1.0);           /*延时 1 秒钟*/
```

```
stop();
}
```

程序输入完以后，先保存到文件夹中，然后再按照为机器人下载程序的步骤进行操作。

请你根据前面总结的 motor(*x*,*y*)函数的规律来判断一下：执行练习中的程序，机器人将进行什么样的运动？请把你的判断填入下面的空格。

首先机器人_____，之后机器人_____，然后机器人_____，最后机器人_____。

现在你为机器人下载并运行练习中的程序，观察机器人的运动与你想象的是否一样。

2．drive(*x*,*y*)函数

【实例】请在“新建空白文档”输入下面的电机驱动程序。

```
void main()
{
drive(60,0);
sleep(2.0);
stop();
}
```

需要注意的是，如果这个程序与机器人中当前的程序相同，则可以直接为机器人下载程序；如果程序名不同，必须按照第 1 章讲的方法，先卸载机器人当中已有的程序，然后再为机器人下载要执行的程序。

请注意观察，运行程序后，机器人是怎样运动的？它的运动与运行 motor(*x*,*y*)函数产生的运动有哪些相似？

通过运行这个程序，我们从机器人的运动状况可以看出：drive(*x*,*y*)函数包含了驱动机器人左右两只电机的 motor(*x*,*y*)函数。

在 drive(*x*,*y*)函数中，*x* 表示机器人电机运动的基准速度，*y* 表示机器人左右电机与基准速度的差。机器人左侧电机的速度为 *x*—*y*，机器人右侧电机速度为 *x*+*y*。

通过这个实例，我们可以看出 drive(*x*,*y*)函数的作用。

（1）在实例中，drive(60,0)表示机器人运动的基准速度为 60，而机器人左右两侧电机的速度与基准速度的差为 0，所以机器人左右两侧电机的速度是相等的。因此，机器人往前走。

（2）如果将函数 drive(60,0)改为 drive(60,20)，我们看到机器人的基准速度保持不变，而机器人左侧电机的速度就要比基准速度小 20，即 40；机器人右侧电机的速度要

比基准速度大 20，即 80。

（3）当函数 drive(60,20)确定以后，机器人右侧电机的速度比左侧电机的速度快一倍，所以机器人就往左侧转弯。因此我们用一个 drive(*x*,*y*)函数就可以让机器人实现不同的运动方式了。

【练习】现在我们修改 drive(*x*,*y*)函数的参数值，再观察机器人的运动状况。

- 当函数表达式为 drive(60,0)时，机器人_____。
- 当函数表达式为 drive(60,20)时，机器人_____。
- 当函数表达式为 drive(60,－20)时，机器人_____。
- 当函数表达式为 drive(－60,0)时，机器人_____。
- 当函数表达式为 drive(－60,20)时，机器人_____。
- 当函数表达式为 drive(－60,－20)时，机器人_____。
- 当函数表达式为 drive(0,20)时，机器人_____。
- 当函数表达式为 drive(0,－20)时，机器人_____。

通过这个练习，我们可以看出 drive(*x*,*y*)函数的使用与机器人的运动有着十分密切的关系，即 *x* 和 *y* 取不同的值，将使机器人产生不同的运动。我们总结这个练习，发现以下规律：

- 当 *y*=0 时，若 *x*>0，机器人往前直行；若 *x*<0，机器人往后直行。
- 当 *x*>0 时，若 *y*>0 且 *y*<*x* 时，机器人往左前方走；若 *y*<0，机器人往右前方走。
- 当 *x*<0 时，若 *y*>0，机器人往右后方走；若 *y*<0 且 *y*>*x* 时，机器人往左后方走。
- 当 *x*=0 时，若 *y*>0，机器人原地逆时针旋转；若 *y*<0，机器人原地顺时针旋转。

请你记住这一组规律，以后会经常用到。

现在我们已经学会控制机器人走路的基本方法，下面开始教机器人如何完成任务。

### 2.1.3 任务 1

机器人必须在运动中，才能完成我们交给它的有关任务。现在我们通过不同的任务教机器人学会走路。

1．motor(*x*,*y*)函数控制机器人往右前方行走

要让机器人转弯，机器人的左右两只轮子的速度就不能相等。任务是右转弯，那么机器人右侧电机的速度应该慢，左侧电机速度应该快，就能让机器人实现往右前方行走了。现在我们模仿前面讲过的实例程序来编写。

【实例 1】让机器人往右前方行走。

```
void main()                          /*主程序说明*/
{                                    /*程序开始标志*/
motor(0,80);                         /*左电机以 80 的速度前进*/
motor(1,60);                         /*右电机以 60 的速度前进*/
sleep(10.0);                         /*行进 10 秒钟*/
stop();                              /*停止执行程序*/
}                                    /*程序结束标志*/
```

请观察机器人是否按照任务要求行走？如果是，说明你的思路是对的，程序是正确的；如果不是，则可能你的思路或程序有问题，需要调整程序，以达到任务要求。

【实例 2】让机器人往左前方行走。

使左侧电机的速度慢，右侧电机的速度快，即可让机器人往左前方行走。

```
void main()                          /*主程序说明*/
{                                    /*程序开始标志*/
motor(0,60);                         /*左电机以 60 的速度前进*/
motor(1,80);                         /*右电机以 80 的速度前进*/
sleep(10.0);                         /*行进 10 秒钟*/
stop() ;                             /*停止执行程序*/
}                                    /*程序结束标志*/
```

运行这个程序，看看机器人是否正在往左前方走。

2．用 drive(*x*,*y*)函数让机器人走一个“∞”字形

让机器人从中间的交点出发，先“画”一个右圆，到中点后再“画”一个左圆。这个程序的关键是要调整好延时，以保证行走路线的准确。

【实例】让机器人走一个“∞”字形。

```
void main()                          /*主程序说明*/
{                                    /*程序开始标志*/
drive(60,－20);                      /*左电机以 80 的速度前进，右电机以 40 速度前进*/
sleep(5.0);                          /*此处的延时你要根据机器人走一圈的时间做调整*/
stop();
sleep(5.0);
drive(60,20);                        /*左电机以 40 的速度前进，右电机以 80 速度前进*/
sleep(5.0);                          /*此处的延时你要根据机器人走一圈的时间做调整*/
stop();                              /*停止执行程序*/
```

```
}                                    /*程序结束标志*/
```

程序输入完后，要先保存，再下载，后执行。

请注意观察机器人是否按照任务要求去走。

如果机器人在“画”右圆或左圆时，转了一圈多，你应该减少 sleep 的延时；如果机器人走了不到一圈，你应该增加 sleep 的延时。

你必须通过反复试验，修改程序，才能使机器人走出一个漂亮的“∞”字形来。

现在我们知道了 motor(*x*,*y*)函数和 drive(*x*,*y*)函数是控制机器人“行走”的两个基本命令，当指令中的数据不同时，机器人的运动状况就不一样。

motor(*x*,*y*)函数比较直观，一看就能判断出机器人的运动状况；drive(*x*,*y*)函数当中，前一个数 *x* 为机器人运动的基准速度，后一个数 *y* 使机器人产生转动；机器人左侧电机的速度为 $x-y$，机器人右侧电机的速度为 $x+y$。

机器人的运动是完成任务的关键，要使机器人的运动速度和运动方向符合任务要求，你只要在实践中不断摸索、不断实验，就一定会很快掌握机器人运动的规律。

## 2.2 教你的机器人“看世界”

能力风暴个人机器人是具有“视觉”的智能机器人，它的“视觉”就相当于我们人的眼睛一样，可以识别外界的物体和光线。这一节我们将在熟悉机器人“视觉”系统的同时，教机器人去“看世界”，同时掌握控制机器人“视觉”系统的基本方法，并根据任务确定机器人的动作。

### 2.2.1 机器人的“视觉”系统

机器人靠它的“视觉”系统来识别外界的物体和光线，而它的“视觉”系统是由红外传感器和光敏传感器组成的。因此，要让机器人“睁眼”看世界，就要控制机器人“视觉”系统，使机器人的“视觉”系统像人一样。控制机器人“视觉”系统的基本指令是红外传感器的 ir_detect()函数和光敏传感器的 analog(*x*)函数。

1．机器人的“眼睛”

机器人看世界的关键是“眼睛”，如图 2.3 所示，机器人的“眼睛”装在机器人外壳的前面，要让机器人“睁眼看见”外界的物体和光线，就要控制红外传感器和光敏传感器。

2．机器人“睁眼”的控制

机器人的“眼睛”有了，怎样让机器人用“眼睛”去观察世界，显然我们通过控制红外传感器和光敏传感器就能作到。要控制机器人的“视觉”系统，也需要机器人

“大脑”的支配，相当于我们人用视觉神经系统去控制眼睛的活动一样。我们指挥机器人的“大脑”，控制机器人的“视觉”系统，也要通过JC程序实现，即通过基本指令ir_detect()函数和analog(*x*)函数，使机器人的“视觉”系统正常工作。

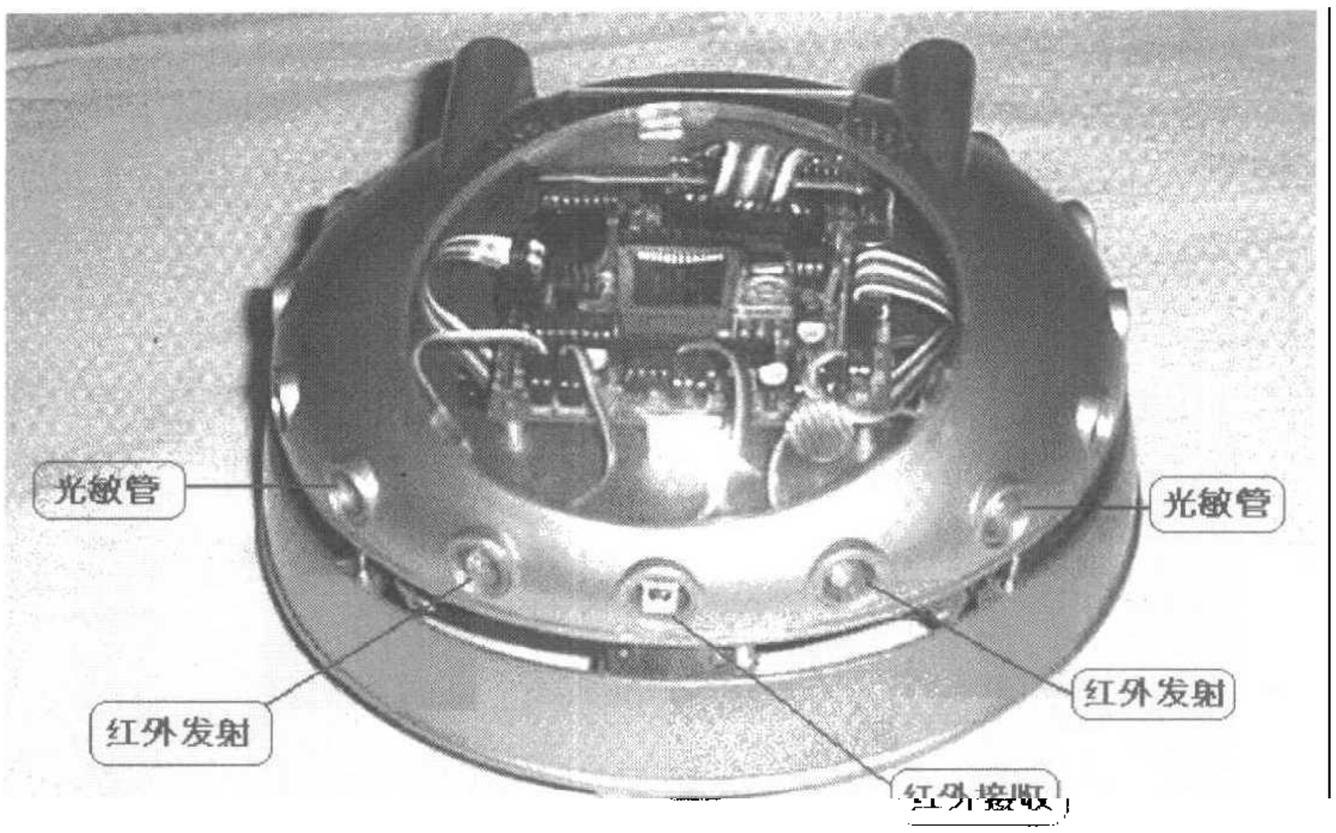



图2.3 机器人的“眼睛”

### 2.2.2 控制机器人“睁眼”的函数

我们已经知道红外传感器和光敏传感器是机器人“睁眼”的关键，只要控制好红外传感器和光敏传感器的“工作”，就可以使机器人的“视觉”系统工作正常。下面我们分别介绍ir_detect()函数和analog(*x*)函数。

1．ir_detect()函数

通过ir_detect()函数的使用，可以使机器人识别“眼前”的物体，为了保证实验效果，物体和传感器之间的距离最好不要超过40cm。

现在我们通过实例看ir_detect()函数使用的效果。

【实例】红外传感器程序。

```
void ir_test()
{
int irvalue;                          /*定义整型变量 irvalue*/
while(1)                              /*控制循环的语句*/
   {
   irvalue=ir_detect();               /*将接收到的红外信号存放于变量 irvalue 中*/
   printf("rvalue=%d\n",irvalue );    /*以十进制格式输出 irvalue 的值*/
   sleep(0.5);
```

}

}

保存、下载这个程序，在执行过程中注意观察 LCD 窗口显示的内容。

当你把手挡在机器人的“左眼”（左侧红外发射器）前，机器人的 LCD 显示什么内容？你再把手挡在机器人的“右眼”（右侧红外发射器）前，机器人的 LCD 显示什么内容？如果你把手放在机器人的正前方，距离 20cm 左右，机器人的 LCD 显示什么内容？

【练习 1】修改这个实例程序。

把程序中的 printf 语句改为：printf(“irvalue=%b\n”,irvalue)。观察机器人的 LCD 显示内容有什么不同？请你记录下来：机器人“眼”前没有物体，显示_____；机器人右侧有物体，显示_____；机器人左侧有物体，显示_____；机器人前方有物体，显示_____。

程序修改后你会发现机器人 LCD 显示的内容与原程序显示的内容完全不一样，原因是：一个是二进制输出，一个是十进制输出。%d 使结果以十进制形式输出，%b 使结果以二进制形式输出。

【练习 2】修改这个实例程序，让 LCD 同时显示二进制和十进制。

```
printf(“ir=%b,ir=%d\n”,irvalue,irvalue);
```

请注意以下几点：

- 当机器人前方没有物体时，LCD 显示的是“0”或“0b00”；
- 当机器人右前方有物体时，LCD 显示的是“1”或“0b01”；
- 当机器人左前方有物体时，LCD 显示的是“2”或“0b10”；
- 当机器人正前方有物体时，LCD 显示的是“3”或“0b11”。

现在你的机器人已经可以识别前方、左前方和右前方有没有物体了，以后我们可以根据不同的任务需要来设置。

2．analog(*x*)函数

analog(*x*)函数的使用，让机器人可以“感觉”到周围光线的强弱。请你输入实例 9 程序并观察 analog(*x*)函数的效果。

【实例 1】检测机器人的左光敏传感器程序。

```
void leftphotic_test()
{
int L;                                  /*定义整型变量*/
while(1)                                /*用于控制循环的语句*/
  {
```

```
    L=analog(1);                    /*将左光敏传感器的检测值存放到变量 L 中*/
    printf("L=%d\n",L);             /*以十进制方式输出光的检测值*/
    sleep(0.5);                     /*让检测暂停 0.5 秒*/
    }
}
```

请你把实例 1 程序下载到机器人中，按“复位”键运行该程序，在机器人的 LCD 上将出现不断变化的数值，显示光线的强弱。

在做这个实例时，请你作以下记录：

你把机器人面向光，用手挡住机器人左侧的光敏传感器，看 LCD 的数值变化。光线强时机器人 LCD 显示的数值是_____；光线弱时机器人 LCD 显示的数值是_____。

【实例 2】检测机器人的右光敏传感器程序。

```
void rightphotic()
{
int R;                              /*定义整型变量*/
while(1)                            /*用于控制循环的语句*/
{
   R=analog(0);                     /*将右光敏传感器的检测值存放到变量 R 中*/
   printf("R=%d\n",R);              /*以十进制方式输出光的检测值*/
   sleep(0.5);                      /*让检测暂停 0.5 秒*/
   }
}
```

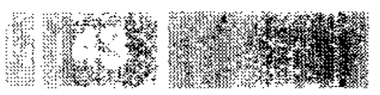
你再试一下右侧光敏传感器，看数值变化的情况。

请你比较以下几个方面：

- 实例 1 和实例 2 两个程序中的语句有什么区别？
- 控制左右光敏传感器的 analog(*x*)函数中的 *x* 值有什么不同？
- 机器人对光的反应如何？

【练习】将实例 1 和实例 2 两个程序合并为一个程序，让机器人能同时“感觉”来自左右两侧的光线，并在 LCD 上同时显示左右光敏传感器检测到的值。请你完成下面的程序。

```
void photic_test()
{
int ___,___;                        /*定义两个整型变量*/
```



```
while(1)
   {
   R=analog(__);                          /*右光敏传感器检测值存放到变量 R 中*/
   L=analog(__);                          /*左光敏传感器检测值存放到变量 L 中*/
   printf("L=%d\n,R=%d\n",_____,_____);   /*以十进制方式输出左右光的检测*/
   sleep(0.5);
   }
}
```

运行程序，现在你的机器人是不是已经可以同时“感觉”来自左右两侧的光线强弱了。

通过观察和实验，我们发现：

- 当 analog(*x*)函数中，*x*=0 时，机器人右侧光敏传感器检测光的强弱；
- 当 analog(*x*)函数中，*x*=1 时，机器人左侧光敏传感器检测光的强弱；
- 通过机器人 LCD 显示结果表明：光线越强，数值越小；光线越弱，数值越大。

### 2.2.3 任务 2

我们通过 ir_detect()函数和 analog(*x*)函数来控制机器人的红外传感器和光敏传感器，使机器人的“视觉”系统功能正常。经过前面的试验，机器人已经会“睁开眼睛”“观察”周围的物体和识别不同的光线了。现在我们要教你的机器人在运动中，利用“视觉”系统完成指定的任务。

1．用 ir_detect()函数控制机器人躲开前面的障碍物

红外传感器安装在机器人的外壳前面，使机器人可以识别在“视线”范围内前方、左前方、右前方的物体。根据任务需要机器人要边走边“观察”，当遇到前方有物体时避开行走。现在我们一起完成下面的程序。

【练习】让机器人躲开前面的障碍物程序。

```
void ir_front()
{                                         /*主程序开始*/
int ir_f;                                 /*定义整型变量 ir_f*/
while(1)                                  /*控制循环*/
   {                                      /*循环开始*/
   ir_f=ir_detect();                      /*把检测到的红外信号保存到 ir_f 中*/
   sleep(0.5);                            /*延时 0.5 秒*/
   if (ir_f==0b11)                        /*条件判断，表示前方有障碍*/
```

```
    {
    printf("ir_front=Y\n");                    /*显示有障碍 ir_front=Y*/
    drive(__,__);                              /*左转或右转*/
    sleep(____);                               /*延时控制转弯方向*/
    }
else printf("ir_front=N\n");                   /*否则显示没障碍 ir_front=N*/
drive(___,___);                                /*前进*/
sleep(___);                                    /*延时*/
}
```

}

你的机器人是否能按照任务要求，遇到前方有障碍物就躲开了？注意左右转的速度和时间要控制好；最后一个延时是控制机器人前进的，时间不要设置太长。

2、用 analog(*x*)函数控制机器人往光线强的方向走

机器人左右两侧各有一个光敏传感器，当左侧光强时，机器人往左侧走；当右侧光强时，机器人往右侧走；在左右光线强度一样时，机器人往前走。

【练习】让机器人往光线强的方向走。

```
void photic_test
{
int ___,___;                                   /*定义两个整型变量 L 和 R*/
while(1)                                       /*控制循环的语句*/
  {
  L=_____;                                     /*将左侧光敏传感器的检测值存到变量 L 中*/
  R=_____;                                     /*将右侧光敏传感器的检测值放到变量 R 中*/
sleep(0.5);
if (L<R )                                      /*条件判断，表示左侧光比右侧光强*/
   {
   drive(___,___);                             /*机器人原地左转*/
   sleep(_____);                               /*控制左转时间*/
   drive(___,___);                             /*机器人直行*/
   }
else if (R<L)
   {
```

```
        drive(___,___);                    /*机器人原地右转*/
        sleep(____);                       /*右转时间*/
        drive(___,___);                    /*机器人直行*/
        }
        else drive(___,___);               /*机器人向前直行*/
    sleep (___);                           /*控制机器人前进时间*/
    }
}
```

请注意观察机器人是否按照要求往光线强的方向走。如果效果不明显，你可以用手电筒在前面为机器人“引路”。

你已经掌握了控制机器人“视觉”系统的方法，你要在实践中反复摸索、实践，以加强机器人的“视觉”功能。

## 2.3 教你的机器人躲避撞到的物体

能力风暴个人机器人是带有“触觉”功能的智能机器人，它的“触觉”就像我们人利用四肢去获取外界环境信息一样，这是能力风暴个人机器人感知外界环境信息的一个重要方面。在这里，我们要教机器人利用碰撞环感知来自 8 个方向的碰撞信息，同时你要学会控制机器人“触觉”的基本方法。

### 2.3.1 机器人的“触觉”系统

机器人的“触觉”系统，我们称它为碰撞传感器。这个碰撞传感器是由 4 只碰撞开关和 1 个碰撞环组成的。因此，要想让机器人通过“触觉”系统获取外界环境信息，就要控制能力风暴个人机器人的碰撞传感器。控制机器人碰撞传感器的基本指令是 bumper()函数。

1．机器人的“触觉”

机器人的碰撞传感器如图 2.4 所示。在机器人的下方有一个圆形环，我们叫它碰撞环，在碰撞环内侧装有 4 只碰撞开关。每当与外界物体碰撞一次，相应的开关就闭合一次。

2．机器人的“触觉”控制

机器人可以通过碰撞环感知外界的环境信息，并通过自己的“大脑”来识别外界环境信息的“特征”。因此我们使用 bumper()函数，实现对机器人“触觉”系统的控制，

使能力风暴个人机器人可以感知来自8个方向的碰撞信息。

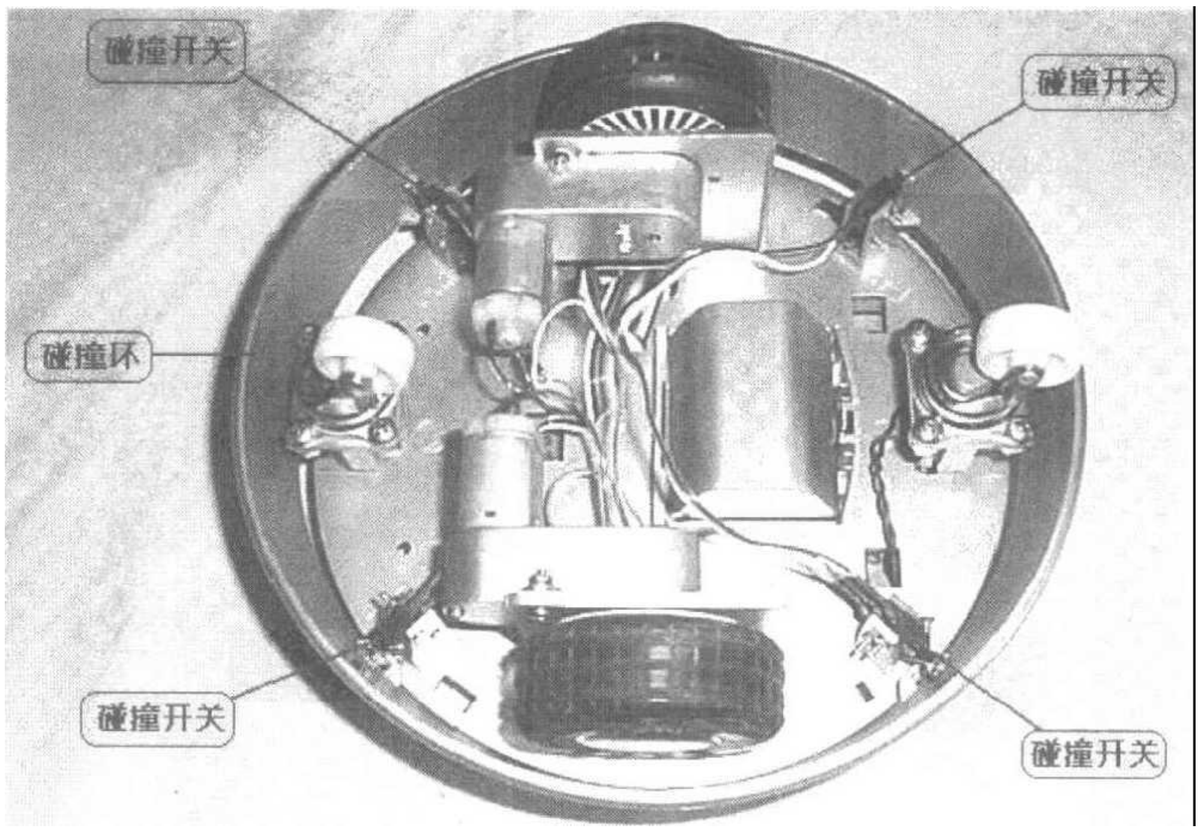



图2.4 机器人的“触觉器官”

### 2.3.2 bumper()函数

控制机器人的“触觉”系统，是机器人感知碰撞信息的关键。我们通过bumper()函数的使用，使机器人在运动中撞到物体时，能“感知”被撞物体位于机器人的哪个方位上。

现在请你输入并运行下面这个程序，观察机器人的碰撞反应。

【实例】碰撞传感器程序。

```
void bumper_test()
{
int bumpvalue;                          /*定义整型变量*/
while(1)                                /*循环控制*/
   {
   bumpvalue=bumper();                  /*调用碰撞传感器库函数并把检测值存放在变量
                                          bumpvalue中*/
   printf("bum=%b\n",bumpvalue);        /*在LCD上用二进制方式显示碰撞检测值*/
   sleep(0.5);
   }
}
```

为机器人下载完这个程序之后，按“复位”键，然后用手从不同的方向按压机器

人的碰撞环，你会发现机器人 LCD 显示的值在不断变化。请你记录下这些检测值。

- 从前方按压机器人的碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从后方按压机器人的碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从左侧按压机器人的碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从右侧按压机器人的碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从左前方按机器人的压碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从右前方按机器人的压碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从左后方按机器人的压碰撞环，记录机器人 LCD 显示的检测值为_____；
- 从右后方按机器人的压碰撞环，记录机器人 LCD 显示的检测值为_____；
- 当机器人没有感受到碰撞时，机器人的 LCD 显示内容为_____。

根据上面的记录，我们一起完成下面这个练习。

【练习】显示前、后、左、右 4 个方向的碰撞信息。

```
void bum1_tast()
{
int bump;
while(1)
 {
 bump=bumper();                    /*在变量 bump 中保存碰撞检测值*/
 if (bump==_____)                  /*条件判断当 bump 为什么值时*/
 printf("Front\n");                /*LCD 显示“前” */
 else if (bump== _____)            /*否则，当 bump 为什么值时*/
       printf("Back\n");           /*LCD 显示“后” */
       else if (bump== _____)      /*否则，当 bump 为什么值时*/
            printf("Left\n");      /*LCD 显示“左” */
            else if (bump== _____) /*否则，当 bump 为什么值时*/
                 printf("Right\n"); /*LCD 显示“右” */
 sleep(0.5);
 }
}
```

需要注意的是程序中需要填写的空格内容为前、后、左、右的二进制值。

由于机器人的碰撞开关分别安装在机器人的左前、右前、左后、右后 4 个位置上，我们总结这个实例和练习，可以得出以下规律：

- 当机器人遇到碰撞时开关值为“1”，无碰撞时开关值为“0”；

● 机器人遇到碰撞后，LCD 显示出一组 6 位数，其中前两位的“0b”是 C 语言中说明二进制常数的前缀，后 4 位则显示碰撞的返回值。

● 后 4 位二进制数分别表示 4 个碰撞开关的不同状态，即不同的数值分别表示左前、右前、左后、右后 4 个位置。

● 四位二进制可以表示出 16 种状态，而在能力风暴个人机器人的碰撞系统中，有 9 种状态是有效的，如前面的观察记录所示。

### 2.3.3 任务 3

机器人的避碰行为是以感知碰撞为前提的。我们的能力风暴个人机器人有 4 只碰撞开关，可以感知来自 8 个方位的碰撞。

现在用 bumper()函数来编写程序，让机器人能感知并显示 8 个方位的碰撞信息。

我们利用条件语句判断机器人的某个方向是否发生碰撞，如果有碰撞，就显示碰撞的方向；如果没有碰撞，就判断下一个方向。前面我们已经完成了 4 个方位碰撞信息的显示，现在把另外 4 个方位发生碰撞的信息显示补齐。

【练习】让机器人能感知并显示 8 个方位的碰撞信息。

```
void bum1_tast()
{
int bump;
while(1)
  {
  bump=bumper();                    /*在变量 bump 中保存碰撞检测值*/
  if (bump== _______)               /*条件判断当 bump 为什么值时*/
  printf("No bumper\n");             /*LCD 显示“无碰撞” */
  else if (bump== _______)            /*否则，当 bump 为什么值时*/
printf("Front\n");                  /*LCD 显示“前” */
else if (bump== _______)             /*否则，当 bump 为什么值时*/
    printf("Back\n");                /*LCD 显示“后” */
    else if (bump== _______)          /*否则，当 bump 为什么值时*/
        printf("Left\n");              /*LCD 显示“左” */
        else if (bump== _______)       /*否则，当 bump 为什么值时*/
            printf("Right\n");           /*LCD 显示“右” */
            else if (bump== _______)      /*否则，当 bump 为什么值时*/
                printf("Front Right\n");    /*LCD 显示“右前” */
```

```
        else if (bump== _______)              /*否则，当 bump 为什么值时*/
            printf("Front Left \n");          /*LCD 显示“左前”*/
            else if (bump== _______)          /*否则，当 bump 为什么值时*/
                printf("Back Right\n");       /*LCD 显示“右后”*/
                else if (bump== _______)      /*否则，当 bump 为什么值*/
                    printf("Back Left \n");   /*LCD 显示“左后”*/
sleep(0.5);
}
}
```

看看机器人是不是已经可以感知并显示来自 8 个方向的碰撞信息了。

你又教你的机器人学会了一种本领，在今后的实践中，你要充分发挥机器人的“触觉”功能，提高机器人的“感知”能力。

## 2.4 教你的机器人“发声”

能力风暴个人机器人可以通过喇叭发声，或者说喇叭就是机器人的“嘴巴”。在为机器人体检的时候，你一定听到了机器人的歌声，它不仅可以“唱”歌，还可以报警、“呼叫”同伴。在这一节里，我们要教你的机器人学会“发声”，并掌握控制机器人“发声”的基本方法。

### 2.4.1 机器人的“发声”系统

机器人的“发声”系统是由喇叭和控制喇叭发声的指令 tone()函数或 beep()函数组成的。机器人用喇叭“说话”，但不是想“说”就“说”，控制机器人“发声”必须通过 tone()函数或 beep()函数才能实现。

1．关于机器人的“嘴巴”

你知道机器人的“嘴巴”在哪吗？在机器人主板上有一个黑色圆形薄薄的动圈式喇叭，这就是机器人的“嘴巴”，如图 2.5 所示。通过它机器人可以发出声音了。

2．机器人“发声”的控制

机器人的“嘴巴”有了，我们要想让机器人 “说话”或者“唱歌”，只要控制机器人主板上的喇叭“发声”就行了。我们通过 JC 程序控制机器人的喇叭，以实现对机器人“发声”的控制。

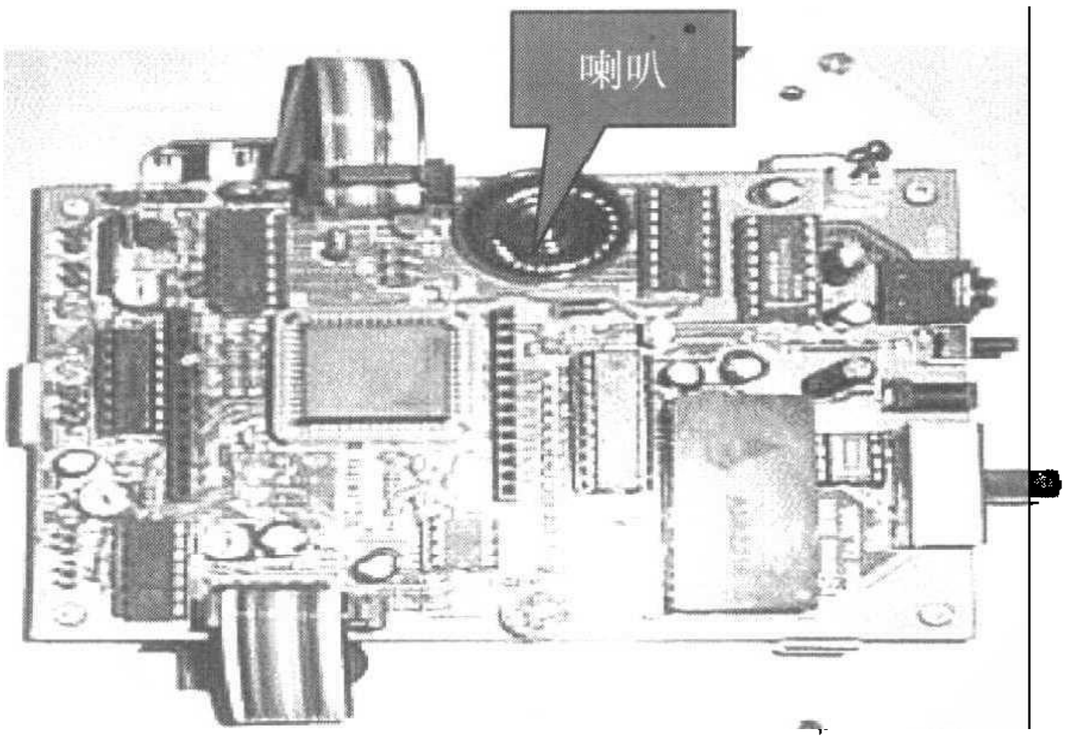



图 2.5　机器人喇叭位置图

### 2.4.2　驱动机器人发声的函数

我们已经知道喇叭是机器人“发声”的关键，只要驱动喇叭，就可以使机器人发出声音。我们用 JC 程序控制喇叭声音的指令有 tone()函数和 beep()函数。下面我们分别介绍这两个函数的使用。

1．tone()函数

使用 tone()函数，可以让你的机器人发出各种不同频率的声音，声音的长短可以根据不同的需要来设置。

现在我们从命令行编辑器直接输入实例 1 的命令。

【实例 1】让你的机器人发出频率为 400Hz 的声音，时间为 3 秒钟。

输入：

```
tone（400.0,3.0）;
```

按一下回车键，机器人就会发出一声“长鸣”。

【实例 2】用 tone()函数控制机器人发出 3 种不同频率的短音。

```
void music1()
{
tone(262.0,1.0);
tone(330.0,1.0);
tone(262.0,1.0);
}
```

【实例 3】让机器人发出 7 个音阶的声音。

```
void music_teat()
{
tone(262.0,1.0);
tone(294.0,1.0);
tone(330.0,1.0);
tone(350.0,1.0);
tone(393.0,1.0);
tone(441.0,1.0);
tone(495.0,1.0);
}
```

运行程序实例 3，可以听到 7 种不同的声音，如果 262.0 相当于简谱的音阶“1”，请你记录一下其它发音的音阶：

（1）tone(262.0,1.0)相当于音阶“1”；

（2）tone(294.0,1.0)相当于音阶“____”；

（3）tone(330.0,1.0)相当于音阶“____”；

（4）tone(350.0,1.0)相当于音阶“____”；

（5）tone(393.0,1.0)相当于音阶“____”；

（6）tone(441.0,1.0)相当于音阶“____”；

（7）tone(495.0,1.0)相当于音阶“____”。

修改实例 3 程序，让机器人发出的声音提高一个八度，节拍延长一倍。

【练习】music_teat1 程序。

```
void music_teat1()
{
tone(_____,_____);
tone(_____,_____);
tone(_____,_____);
tone(_____,_____);
tone(_____,_____);
tone(_____,_____);
tone(_____,_____);
}
```

其实，音阶提高一个八度，频率即增加 1 倍；节拍延长 1 倍，时间即延长 1 倍。

现在你可以完成这个练习的程序了吧。

运行一下这个练习的程序，听一听机器人的发音是不是提高了，每个音节发音是不是长了。

通过这几个实例和练习，我们发现了 tone(*x*,*y*)函数的以下规律：

● *x* 表示机器人发音的频率，*x* 值大，机器人发出的声音频率就高；*x* 值小，机器人发出的声音频率就低。

● *y* 表示机器人发出声音的长短，*y* 值大，机器人发出的声音长；*y* 值小，机器人发出的声音短。

● 选择适当的 *y* 值可以控制乐曲的节奏。假如把 *y*=1.0 作为 1 拍，那么 *y*=0.5 时就是半拍，即 1/2 拍。

● 函数 tone(*x*,*y*)中的两个数均为浮点数。

2．beep()函数

使用 beep()函数，机器人会发出一个 0.3 秒 500 Hz 的声音。

如果要想用异步控制喇叭发声，可以使用函数 set beeper pitch()，beeper on()，beeper off()。

set beeper pitch(*f*)函数的功能：是用于启动喇叭发声的，发声频率为 *f*。*f* 为浮点数，在使用中你可以自己根据不同的要求或任务选择发声频率。

beeper on()函数的作用：用于 set beeper pitch(*f*)函数选定的频率 *f* 后，启动喇叭发声，发声的长短，由何时执行 beeper off()函数决定。

beeper off()函数的作用：关闭喇叭。

### 2.4.3 任务 4

你已经教会你的机器人通过 tone(*x*,*y*)函数或 beep()函数驱动喇叭发声了，现在我们通过以下的任务，教你的机器人成为“歌唱家”。

根据《小星星》的乐谱，先填写音阶和节拍。我们定 *y*=0.5 为一拍。

【实例 1】让机器人“演唱”歌曲《小星星》。

```
void music_star()
{
while(1)                /*循环控制*/
   {
   tone(262.0, 0.5);
   tone(262.0, 0.5);
   tone(393.0, 0.5);
```

```
tone(393.0, 0.5);
tone(441.0, 0.5);
tone(441.0, 0.5);
tone(393.0,1.0);
tone(350.0, 0.5);
tone(350.0, 0.5);
tone(330.0, 0.5);
tone(330.0, 0.5);
tone(294.0, 0.5);
tone(294.0, 0.5);
tone(262.0,1.0);
for (i=1; i <2; i++)          /* 循环两次下面的程序*/
   {
   tone(393.0, 0.5);
   tone(393.0, 0.5);
   tone(350.0, 0.5);
   tone(350.0, 0.5);
   tone(330.0, 0.5);
   tone(330.0, 0.5);
   tone(294.0,1.0);
   }
}
```

下载并运行“【实例 1】”程序，机器人就开始为你“演唱”《小星星》了。听听机器人是不是按照你的要求进行“演唱”的。

你可能已经发现机器人“唱”起来就不住口了。怎么办？比如，如果让机器人“演唱”4 遍《小星星》就停下来，我们只需要在程序实例 1 中做一点修改就能实现。

【实例 2】修改前面的实例 1，让机器人“演唱”4 遍《小星星》。

```
void music_star()
{
for (i=1;i< 4; i++)      /* 控制循环 4 次*/
   {
```

```
.
.
.
    {
.
.
.
.
    }
}
```

现在祝贺你的机器人成为天才的“小歌唱家”。

## 2.5 教你的机器人“听”指挥

机器人可以通过麦克风获取外部的声音信息，这是它能“听”指挥的关键。但是由于语音系统是一个高难度的技术，在能力风暴个人机器人中还没有得到应用，所以你的机器人能够识别的声音信号只是其声强的信号。在这一节里我们不光要认识机器人的“耳朵”，还要学会掌握控制机器人“听力”的基本方法。

### 2.5.1 机器人的“听觉”系统

控制机器人的“听觉”系统是机器人“听”指挥的关键，就相当于我们人的耳朵。因此，要使机器人能够“听”指挥，就要使机器人的“听觉”系统功能正常。能力风暴个人机器人的“听觉”系统是由麦克风和 analog($x$)函数组成的。控制 analog(2)函数就可以使机器人接收到外界的声音信息。

1．关于机器人的“耳朵”

机器人的“耳朵”是用于接收外界声音信息的。它是一个小麦克风，安装在机器人的主板上，如图 2.6 所示。你找到了吗？这就是机器人的“耳朵”，机器人通过它可以“听”你的指挥。

2．机器人的“听力”控制

机器人的“耳朵”有了，我们只要控制了麦克风就实现了对机器人“听力”的控制，使机器人可以“听”到你发出的简单命令，并做出相应的反应。在这里我们通过 JC 程序指挥机器人“大脑”，控制机器人的“听力”，使你的机器人能“听”、会“听”。

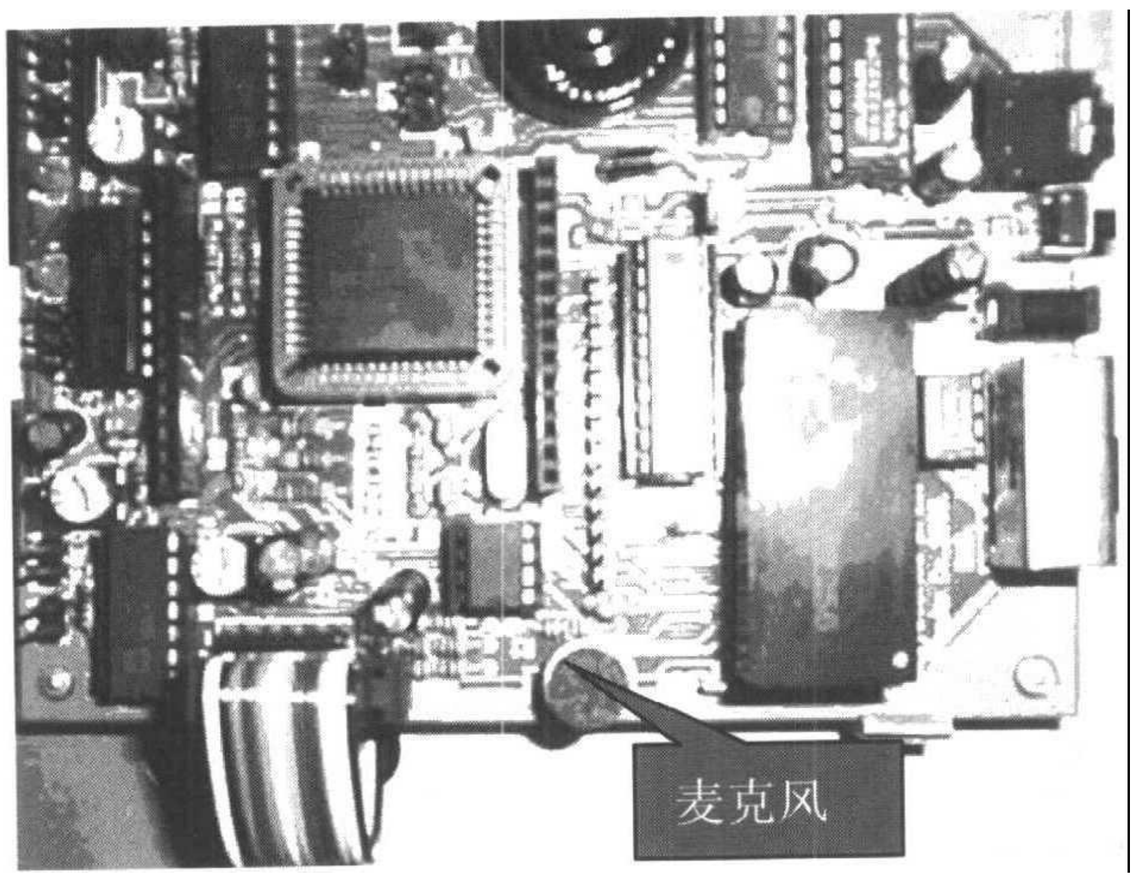



图 2.6　机器人麦克风位置图

### 2.5.2　analog(2)函数

analog(*x*)函数我们已经见过了，在机器人识别外界光线时已经用到过。当时在analog(*x*)函数中，*x* 的值为 0 或 1，分别表示左光敏传感器和右光敏传感器。现在 *x* 取值为 2，表示声音传感器。使用 analog(2)函数，就可以使你的机器人具有识别声音信号的能力。

现在我们输入下面的程序，测试机器人的“听力”。

【练习 1】检测“听力”。

```
void mic_test
{
int micvalue;
while(1)
  {
  micvalue=analog(2);                    /*检测声音信号*/
  printf("mic=%\n",micvalue);      /*在 LCD 上显示检测到的声音信号值*/
  sleep(0.5);
  }
}
```

运行这个程序后，你可以通过拍手、命令或其它发声方式，观察 LCD 上声音信号值的变化。你可以发现外界的声音越大，机器人 LCD 显示的数值就越_____；外界的

声音越小，机器人 LCD 显示的数值就越_____。

在这个练习程序的基础上，编写下面这个程序，让机器人“听”到命令就“回答”。

【练习 2】“有叫必应”。

```
void mic_test
{
int micvalue;
while(1)
  {
  micvalue=analog(2);                    /*检测声音信号*/
  if (micvalue>_____)                    /*条件判断当收到的声音信号大于某个值*/
  tone(400.0,1.0);                       /*机器人“回答”一声*/
  else printf(“mic=%\n”,micvalue);       /*否则在 LCD 上显示检测到的声音信号值*/
  sleep(0.5);
  }
}
```

声音信号值的选择，可以参考运行练习 1 的程序测到的某个值来确定。

另外，你要注意两点：

● analog(2)函数只能检测声音的大小，即声音信号的音量，而不能检测语声音信号的频率；

● 测到的声音信号越强，数值越大；信号越弱，数值越小。

可见机器人有“听力”，但它的“耳朵”并不是很灵敏。

### 2.5.3 任务 5

现在你的机器人已经可以通过 analog(2)函数驱动麦克风接收声音信号了，我们充分发挥机器人“听觉”功能的作用，来完成相应的任务。

【练习】让机器人“听”到“开始”的命令就向前“跑”。

机器人在起跑线待令，当它“听”到“开始”的命令就向前“跑”。在程序中用条件语句判断机器人是否“听”到“开始”的命令。

```
void leg
{
int micvalue;
while(1)
  {
```

```
    micvalue=analog(2);                    /*检测声音信号*/
    if (micvalue>_____)          /*条件判断当收到的声音信号大于某个值*/
      motor(0,100);                        /*机器人左电机启动*/
      motor(1,100);                        /*机器人右电机启动*/
    else printf("mic=%\n",micvalue); /*否则在 LCD 上显示检测到的声音信号值*/
    sleep(0.5);
    }
}
```

运行这个程序后注意观察机器人的反应是否灵敏，你可能要反复调试声音信号值，使你的机器人一“听”到发令就能“起跑”。

显然，如果把机器人判断声音信号的值设置得比较低，机器人对声音反应的灵敏度就会高，而此时也会使机器人产生误动作；如果将机器人判断声音信号的值设置过高，就会降低机器人的灵敏度，使机器人的“听力”下降。所以要在实际中反复测试机器人的“听力”，以确定取值范围，使机器人既反应灵敏又不产生误动作。

现在你已经掌握了控制机器人的“听觉”和提高机器人“听力”的办法。你的机器人是不是很“听话”？如果不“听话”，则可能是你没有教好。你要根据不同的任务和要求，反复调试、测试它的“听力”，使你的机器人“听力”达到任务需要的标准。

## 2.6 让你的机器人走得更好

机器人的“行走”是我们完成各项任务的关键，我们只有控制好机器人的两只“脚”，才能让机器人按照要求走好，“走出”规定的路线。

### 2.6.1 矫正机器人的电机

你在教你的机器人走路时，可能发现机器人的两只“脚”有点“毛病”，你让它往前“走”，它可能斜着往左或者往右“走”，这是由于左右两只电机之间的差异造成的。在这种情况下，我们首先要矫正机器人的左右两只电机，使两只电机转动达到最佳效果，为机器人“走”得直、“走”得正奠定基础。

1．输入或调用矫正电机程序。

【实例】 电机矫正程序。

```
int drive_bias=0;
void driveb(x,y)                       /*矫正电机*/
```

```
{
int y1=(drive_bias*x)/100;
motor(0,x－(y+y1));
motor(1,x+(y+y1));
}
```

也可以在 JC 窗口打开“libs”文件夹，找到“common.c”程序，如图 2.7 所示。


图 2.7 打开“libs”文件夹窗口

打开此文件，可以看到在“common.c”程序的最后是矫正电机的函数，如图 2.8 所示。

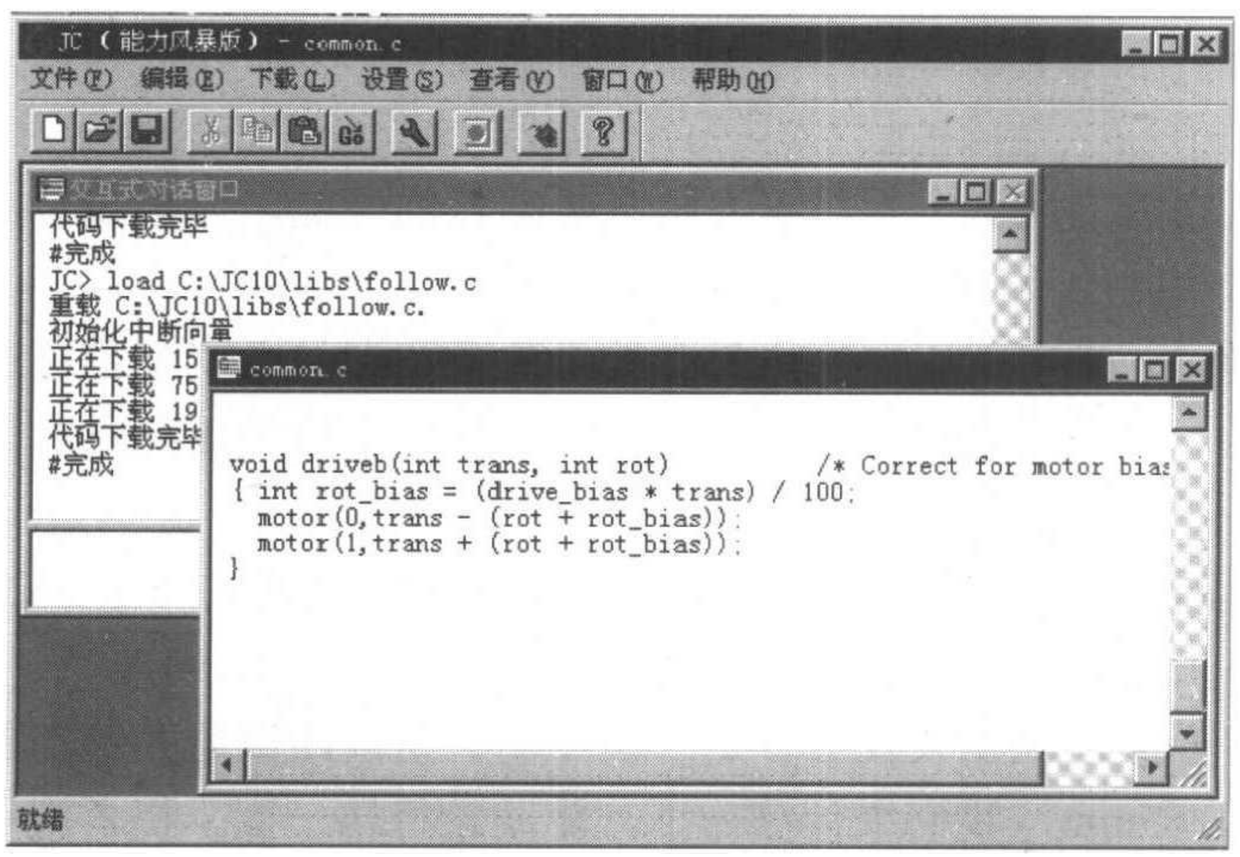



图 2.8 “common.c”程序窗口

2．试验、调试电机并认真观察。

（1）先取偏置量 drive_bias=0，在 driveb(*x*,*y*)函数中，取 *x*=80，*y*=80，然后观察机器人的行走路线。如果机器人走的是直线，那么机器人的电机就不需要矫正；否则就要用改变偏置量 drive_bias 函数的值，来矫正机器人的两只电机，以使机器人能走出一条直线。

（2）如果需要矫正电机，取偏置量 drive_bias=5，*x* 和 *y* 的值不变，观察机器人的行走路线偏向哪一侧？

（3）再取 drive_bias=－5，*x* 和 *y* 的值仍然不变，观察机器人的行走路线偏向哪一侧？

经过反复进行试验、调试，直到机器人基本能走直线或矫正值 drive_bias 已无法再小为止。

### 2.6.2 关于光电编码器

能力风暴个人机器人还有两套特殊功能的传感器，这就是光电编码器，它是能够传递位置信息的传感器，如图 2.9 所示。

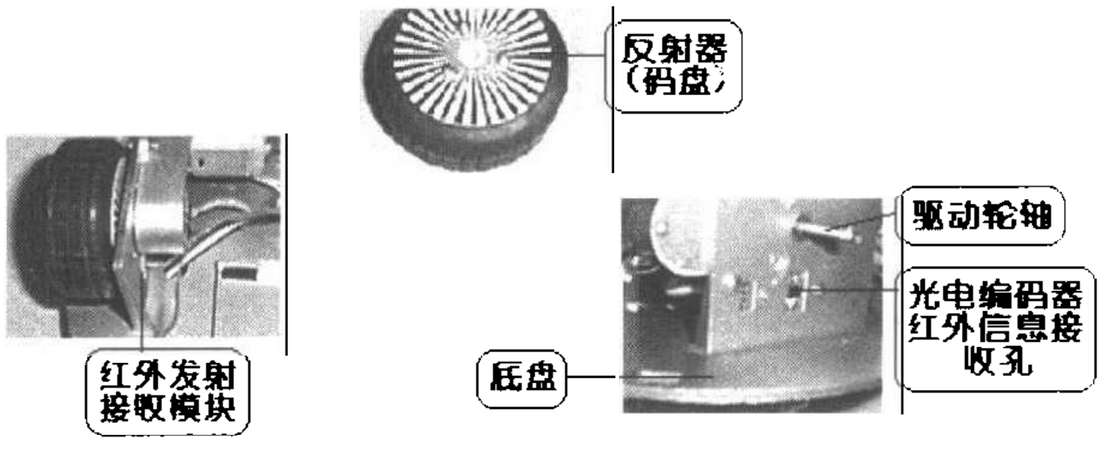



图 2.9　光电编码器

我们看到，在能力风暴个人机器人两只驱动轮的内侧，分别装有由微型红外发射接收模块和黑白相间的反射器组成的光电编码器。

驱动轮转动时，光电编码器的码盘上的黑色条和白色条就交替经过光电编码器的红外信息接收孔。当黑色条经过时，红外光无反射；当白色条经过时，红外光就会产生反射，这样一黑一白就构成一个脉冲。

360° 的码盘被分成 66 等份，因此驱动轮转一圈，就可以产生 33 个脉冲。我们计算一下，每产生一个脉冲，驱动轮转的角度是：360 / 33≈10.91° 。

驱动轮每转一圈，机器人就会行走一段距离，这段距离就是驱动轮的一个周长，驱动轮的直径是 65mm，我们来计算一下，驱动轮每转一圈，机器人行走的距离是 2πR，即：2×3.14×(65 / 2)≈204mm；每一个脉冲机器人行走的距离应该是：204 / 33

≈6.18mm。

由此我们看出，光电编码器的使用，可以比较准确地计算出机器人行走的距离。在此之前，我们通过延时函数 sleep(*t*)控制机器人的行走距离和转弯角度，但是调整起来比较麻烦，并且受电源能量和电机转速的影响，稳定性也比较差。因此，要想提高机器人行走距离和转弯角度的准确性和稳定性，就要通过光电编码器来实现了。

1．初始化光电编码器函数。

（1）格式：init_velocity()

（2）功能：初始化编码器

（3）说明：在使用编码器计数前必须对光电编码器进行初始化。

2．光电编码器状态函数。

（1）格式 1：left_shaft()

（2）功能：返回左编码器的当前状态。

（3）说明：返回的值为 0，表示脉冲为低电平；返回的值为，1 表示脉冲为高电平（分别对应码盘上的白格和黑格）。

（4）格式 2：right_shaft()

（5）功能：返回右编码器的当前状态。

（6）说明：返回的值为 0，表示脉冲为低电平；返回的值为 1，表示脉冲为高电平（分别对应码盘上的白格和黑格）。

3．光电编码器计数读取函数。

（1）格式 1：get_left_clicks()

功能：取左编码器的计数值，并将计数重新设置为零。

（2）格式 2：get_right_clicks()

功能：取右编码器的计数值，并将计数重新设置为零。

4．应用有关函数检测光电编码器。

【实例 1】检测光电编码器。

```
void main()
{
 int lsh;
 while()
  {
  lsh=left_shaft();                    /*左编码器的状态*/
  printf("lsh=%d\n",lsh);
  sleep(0.5);
```

```
    }
}
```

请注意观察：

（1）转动机器人的左轮子，观察 LCD 显示左编码器的当前状态。

（2）当反射器上的白色条对准红外接收芯片时，lsh 的值为_____。

（3）当反射器上的黑色条对准红外接收芯片时，lsh 的值为_____。

【实例 2】用光电编码器测量行走距离。

```
#define LEFT 1
#define RIGHT 2
{
int read_encodar(int leftorright)      /*定义一个函数，有一个整形输入参数*/
if (leftorright==LEFT)
return get_left_clicks();              /*返回整数值*/
if (leftorright==RIGHT)
return get_right_clicks();
}                                      /*每次用完后，记数器将清零*/
void main()
{
int leftclick=0;
init_velocity();          /*初始化编码器，计数前必须要初始化*/
while()
   {
    leftclick=leftclick+read_encodar(LEFT);
    printf("leftclick=%d/n",leftclick);
    }
}
```

请注意观察一下两点：

（1）旋转机器人的左轮胎一周，观察 LCD 显示的计数值是多少？

（2）修改实例 2 的主程序，旋转机器人右轮胎一周，观察 LCD 显示的计数值是多少？

5．光电编码器的应用。

【练习】让你的机器人走 2m。

请思考一下，驱动轮转一圈机器人行走的直线距离是_____cm；如果让机器人走 2m 远的距离，机器人的轮子大约转_____圈；机器人走 2m 时光电编码器记录的数值应该为_____。

让机器人走 2m 程序的部分流程图，如图 2.10 所示。

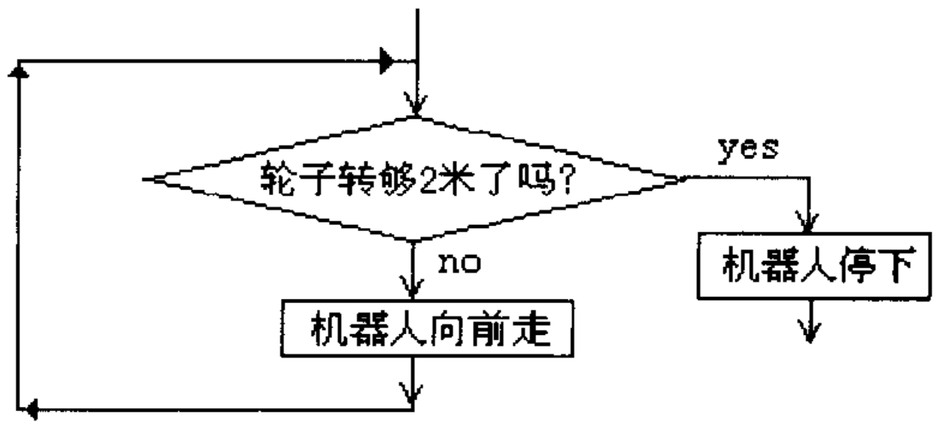



图 2.10　让机器人走 2m 的程序流程图

让机器人走 2m 的程序如下，请把它补充完整。

```
void main()
{
int s,lsh;
init_velocity();                          /*编码器初始化*/
while (s<____)
   {
   drive(100,0);
   sleep(____);
   lsh=get_left_clicks();                 /*左轮编码器的计数*/
   s=s+1sh;                               /*累加计数值*/
   }
stop();
}
```

## 2.7　习题

1．机器人的电机驱动。

（1）机器人的电机带动轮子相当于人的什么动作?

（2）机器人的电机有几种驱动方式？它们有什么区别？

（3）机器人可以有几种不同的运动方式？

（4）任务：让机器人往左后方行走。

（5）任务：让机器人往右后方行走。

（6）任务：让机器人走“∞”字形，要求机器人先“画”左圆，再“画”右圆。

（7）任务：让机器人走大“∞”字形，然后修改程序，让机器人走小“∞”字形。

（8）任务：让机器人走正方形。

（9）任务：让机器人走三角形。

（10）任务：让机器人走“M”形。

2．机器人的“视觉”系统。

（1）机器人的“视觉”系统由哪几个部分组成？

（2）机器人通过什么识别光线的强弱？

（3）机器人通过什么判断前方有没有障碍物？

（4）控制机器人“视觉”系统的指令有几个？作用分别是什么？

（5）任务：让机器人在前进中遇到障碍物时，“后退”一步停下来。

（6）任务：让你的机器人能避开左前方或右前方的障碍物。

（7）任务：让机器人往光线弱的方向走。

（8）任务：让你的机器人在找到指定光源后立刻停下来。

（9）任务：让机器人在寻找光源的过程中能避开障碍物。

3．机器人的“触觉”系统。

（1）机器人的“触觉”系统相当于人体的哪一部分？

（2）机器人怎样通过“触觉”去“感知”信息的？

（3）控制机器人“触觉”系统的指令是什么？

（4）任务：使你的机器人在运动中躲避碰撞的物体。

（5）任务：让你的机器人在运动中碰到物体时停下来。

（6）任务：让你的机器人在运动中在某个方向上碰到物体时，就往相反的方向运动。

4．机器人的“发声”系统

（1）机器人发声的喇叭相当于人体的什么器官？

（2）怎样控制机器人的喇叭发声？

（3）怎样让机器人加快或放慢“演唱”速度？

（4）任务：让机器人在前进中遇到障碍物时，“后退”一步停下来，并放歌曲加以提示。

（5）任务：让机器人找到光源时，停下来并发出报警声。

（6）任务：让机器人在运动中碰到物体时就发出一声短音。

（7）任务：自己编一个程序 music_birthday，在祝贺朋友生日的时候，让你的机器人为朋友“演唱”这首《祝你生日快乐》，别有一番情趣。

（8）任务：让你的机器人在天亮时，“演唱”一首歌伴你起床。

5．机器人的“听觉”系统

（1）能力风暴个人机器人的“听力”与人耳的听力有什么共同之处和不同之处？

（2）控制机器人“听力”的指令是什么？

（3）任务：让你的机器人“听”到命令，回答一声就“跑”出去或“跑”回来。

（4）任务：编个程序让机器人“听”到第一声发令就“跑”出去，听到第二声发令就停下来。

6．让机器人走得更好

（1）如果机器人走的不是直线你该怎么办？

（2）怎样让机器人走出规定的路线？

（3）任务：编程测试机器人在行走时轮子转过的角度。

（4）任务：计算机器人原地转动的角度与编码器值的关系。编一个程序，让机器人转过 90°，运行后测量一下精度。

（5）任务：编程让机器人进行 1~3 米的“短跑”比赛。

（6）任务：用光电编码器编程，让机器人走“之” 字型。

通过这一章的学习，我们知道了：

● 机器人要想“走”，离不开电机，而电机的转动要靠 motor(*x*,*y*)函数和 drive(*x*,*y*)函数驱动，否则机器人将“寸步难行”。

● 机器人要想“看”，离不开红外传感器和光敏传感器，控制这两个传感器的指令分别是 ir_detect()函数和 analog(*x*)函数，否则机器人将成为“睁眼瞎”。

● 机器人要想“摸”，离不开碰撞传感器和控制指令 bumper()函数，否则机器人将没有“感知”能力。

● 机器人要想“说”，离不开喇叭及驱动喇叭“发声”的 tone(*x*,*y*)函数和 beep()函数，否则机器人将成为“哑巴”。

● 机器人要想“听”，离不开麦克风和控制“听力”的函数 analog(*x*)，否则机器人的“耳朵”将成为摆设。

现在你的机器人已经学会了怎样“看”、怎样“听”、怎样“说”、怎样“摸”和怎样“走”，但是你这个“小老师”还没有完成任务。你还要继续教你的机器人，让它能综合运用已经学到的本领去处理问题，提高它的智能化程度，从而更好地完成你交给它的任务。

# 第 3 章　你的机器人最聪明

在第 2 章里你教机器人学会了一些本领，这一章我们通过了解机器人的“大脑”，介绍控制机器人的 JC 及其程序的基本结构，掌握为机器人编程的方法，从而使机器人能综合运用已有本领，提高处理信息、完成任务的综合能力。

## 3.1　关于机器人的“大脑”

我们人对周围环境做出反应的过程，主要是“感觉→大脑思考→做出反应”，机器人对信息的处理过程也是如此，它通过各种传感器获取内部信息和外界环境信息，然后经过处理，再按照你的要求去进行工作。因此，机器人“大脑”的作用是为了“思考”，而它“思考”的过程就是对各种信息的加工、处理的过程。所以机器人的“大脑”是机器人全部智慧的关键。利用 JC 这一优秀的软件开发工具，可以使机器人更“聪明”。

### 3.1.1　智能机器人的三大要素

机器人是一种具备一些与人相似的智能的工具。机器人的智能体现在获取信息、加工信息、处理信息以及做出相应的反应等方面。也就是说，我们把能“感觉”、会“思考”、能“决策”、会“动作”的系统或者工具称为智能机器人。能力风暴个人机器人是智能机器人的一种，它可以通过 5 种 11 个传感器获取不同的外界环境信息，然后通过微控制器 68HC11 芯片“思考”、“决策”，最后通过驱动直流电机、麦克风或 LCD 做出相应的反应。

1．信息的获取

一个正常人，通过眼、耳、鼻、舌、身，就可以获取各种不同的外界环境信息，这是轻而易举的事。但是对于一台机器来说，要让它能“看”、能“摸”、会“听”、会“说”就不那么容易了。

我们知道机器人获取外界的环境信息，靠的是各种不同的传感器，如红外传感器、光敏传感器、碰撞传感器、超声传感器、温度传感器等等，这些传感器可以根据机器

人的任务和需要来设置。通过不同的传感器，机器人能获得不同的外界环境信息。

2．信息的加工和处理

机器人把各种传感器获得的外界环境信息，输入自己的“大脑”进行“思考”。它能根据不同的需要，判断外界的环境信息，并按照不同的任务和要求作出不同的反应。机器人的“大脑”实际上就是一个微控制器，人通过某种计算机语言就可以与机器人进行交流。我们与能力风暴个人机器人是通过 JC 进行交流的。

3．作出反应

机器人在获取外部环境信息并对信息进行加工和处理之后，就要作出相应的反应。比如，当红外传感器探测到前方有障碍物时，此信息经过“大脑”处理后，机器人作出绕开障碍物的“决定”，并改变电机的转速，从而改变前进方向绕开障碍物；我们决定让机器人“冲上去”，这时就要让机器人加速前进。总之反应的方式是根据任务的要求制定的。不同的信息，要求机器人作出的反应是不一样的。

### 3.1.2 能力风暴个人机器人与智能机器人

机器人学按机器人的智能程度把机器人分为两大类：第一类被称为一般机器人，这类机器人不具有智能，只具有一般的可编程功能和操作功能；第二类被称为智能机器人，在这类机器人当中，又根据机器人不同的智能程度分为以下 3 种：

- 传感型机器人。它的特点是对接收到的传感信息，如视觉、听觉、触觉、接近觉、力觉和红外、超声以及激光等进行处理，实现控制与操作的功能。
- 交互型机器人。这种机器人让我们通过与计算机系统进行“人-机”对话，实现对它的控制与操作。
- 自主型机器人。这类机器人在设计制作完成之后，就不再需要人的干预，它能够在各种环境下自动完成各项拟人的任务。

能力风暴个人机器人既有利用传感器传递信息，实现控制与操作的能力，又有“人-机”对话，实现对机器人的控制与操作能力，还有自动完成各项拟人任务的能力。所以我们说能力风暴个人机器人是智能机器人。

1．获取外界环境信息的传感器

机器人感知环境的能力，是机器人产生智能行为的前提。通过第 1 章和第 2 章我们已经知道，能力风暴个人机器人本身配有 5 种 11 只传感器，因此它对环境的感知能力非常强，可以产生许多智能行为。另外由于能力风暴个人机器人的可扩展性，它还能根据不同的需要扩展其他功能的传感器，增强感知能力，提高它的智能程度。

能力风暴个人机器人本身配的传感器有红外传感器、光敏传感器、碰撞传感器、光电编码器、麦克风等 5 种，如图 3.1 所示。

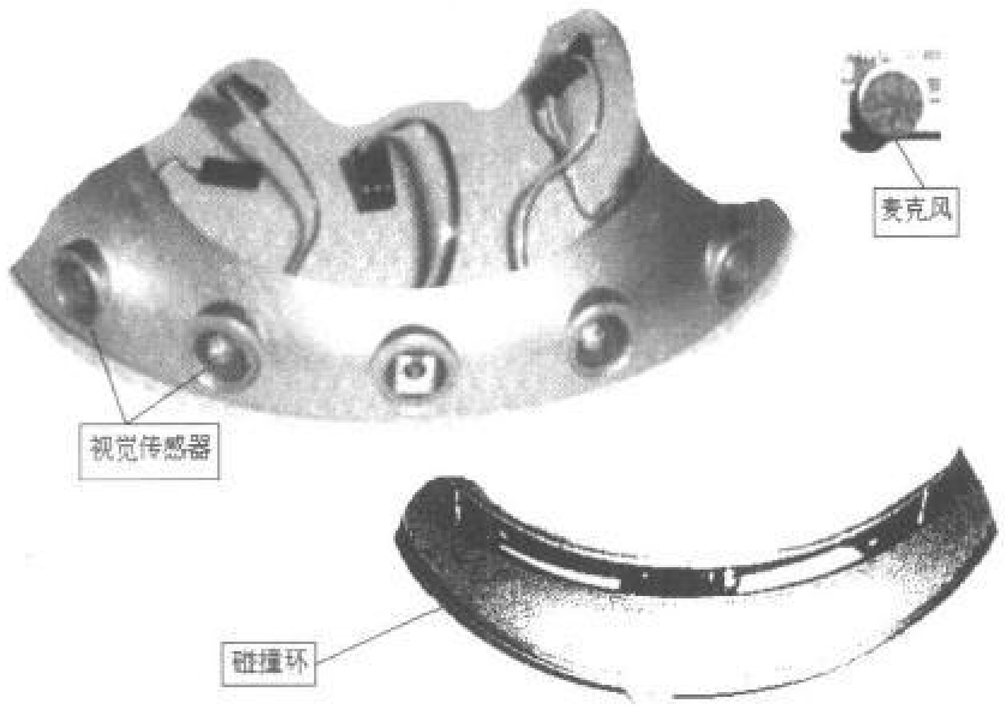



图 3.1　部分传感器

2．加工处理信息的“大脑”

能力风暴个人机器人的微控制器是 68HC11，它包括 CPU、片内存储器、定时器系统、串行口、A/D 转换器、并行 I/O 口、中断和复位系统。68HC11 的自下载功能，为我们提供了纯软件开发调试的优秀工具——JC。JC 的特点是：既可用于高层应用软件的开发，又可便于低层驱动的开发，还能交互调试。

能力风暴个人机器人的“大脑”结构如图 3.2 所示。

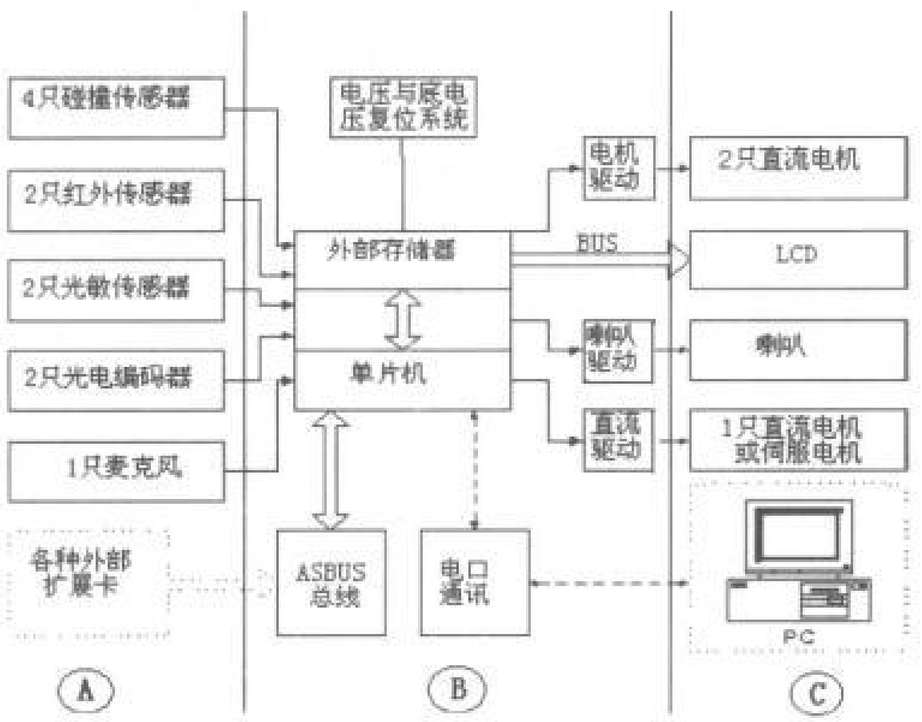



图 3.2　能力风暴个人机器人的“大脑”结构图

由图 3.2 我们看到,机器人的“大脑”结构分为 3 个部分,下面我们按照从左到右的顺序来看一下这张“大脑”结构图。

首先我们看一下如图 3.2 所示的左侧部分Ⓐ。

（1）信息获取部分。在如图 3.2 所示的左侧部分Ⓐ，看到了我们已经比较熟悉的

5种11只传感器，它们是：碰撞传感器4只、红外传感器2只、光电传感器2只、光电编码器2只、麦克风1只。这11只传感器，都是能力风暴个人机器人已经具有的。机器人通过这11只传感器对周围的环境进行搜索和检测，以获得各种相应的信息或信号，然后把这些信息或信号送给机器人的微控制器去进行处理。如果我们要想提高机器人对外部环境的感知能力，就可以通过增加传感器来实现。在11种传感器的下方有一个虚线框，标着“各种外部扩展卡”，这是一个开放的电子接口，通过这个接口，我们就可以非常方便地为机器人增加相应的传感器，提高机器人对外界环境的感知能力。

（2）信息的加工与处理部分。在如图3.2所示的中间部分Ⓑ，就是能力风暴个人机器人的微控制器，它是机器人“思考”问题的“大脑”。机器人通过微控制器对获取的信息进行加工、处理，并作出决策。机器人的决策能力是我们通过JC与机器人进行交流的结果。当我们为提高机器人的感知能力而增加传感器时，你可能会担心机器人控制板的功能是否够用。请你放心，能力风暴个人机器人有一个功能非常强大的硬件扩展总线ASBUS，在如图3.2所示中间部分Ⓑ的下面。通过这个硬件扩展总线我们可以非常方便地扩展控制板的功能，满足机器人的任务需要。

（3）动作反应部分。 在如图3.2所示的右侧部分Ⓒ,有两只直流电机,一块LCD显示屏和一只喇叭,这些是能力风暴个人机器人的执行器,它是机器人做出反映的执行机构。机器人的“大脑”根据任务需要，作出决策，驱动执行机构完成任务。

如果现有的执行机构不能满足任务的需要，我们可以通过开放的机械接口，增加执行机构，使机器人能够达到任务的要求。

3．执行器

能力风暴个人机器人在获取外界环境信息后，经过“大脑”的计算处理，然后再作用于环境。能力风暴个人机器人作用于环境的驱动器有3个：直流电机、喇叭、LCD，如图3.3所示。

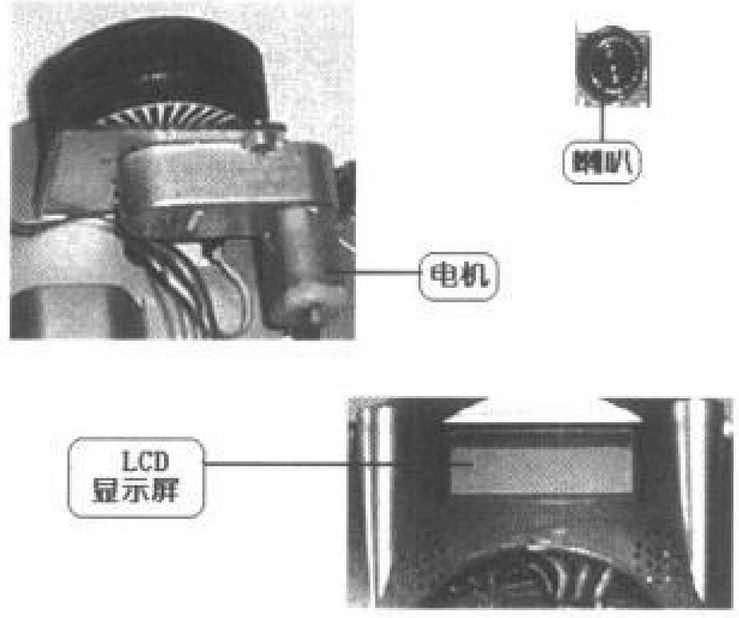



图3.3 能力风暴个人机器人的执行器

其中两只直流电机构成了能力风暴个人机器人的行走装置；喇叭就是机器人的“嘴”，可以“唱歌”、发出警告或“呼叫”同伴；LCD 用于显示机器人实时运行的信息，便于检测状态和诊断诊断。

## 3.2 关于程序

我们这里所说的程序，是指用 JC 为能力风暴个人机器人编排的工作顺序，我们把这个工作顺序称为程序。

我们人在做事情或处理问题的时候，是按照事情或问题的性质、程度分轻重缓急的去落实、办理。因此，我们让机器人去完成任务的时候，也要告诉它先做什么，后做什么，遇到特殊情况怎么办等等。机器人一定会按照我们给它编排的程序去进行工作，完成我们交给它的任务。

### 3.2.1 程序的基本结构

在我们还没有学习为机器人编写程序之前，你已经可以用 JC 与你的机器人进行交流了。但是要想让你的机器人伙伴做好一件事情，既省时又省力，高智慧、高水平地去完成任务，你就必须根据任务的需要为你的机器人编写相应的程序。现在我们就开始学习为机器人编写程序的方法。

在为机器人编写程序之前，我们首先要了解程序的基本结构。在第 1 章和第 2 章中，我们为机器人输入程序、下载程序、执行程序时，看到一般程序的运行，是按照程序的基本结构，自上而下顺序执行的。所以程序的基本结构可以分为 3 种：

- 顺序结构
- 循环结构
- 选择结构

1．顺序结构程序

顺序结构的程序比较简单，但是非常可靠。为了使机器人不受外界的任何干扰，认认真真地完成你交给它的任务，我们采取顺序结构的方式为机器人编程。下面通过实例了解顺序结构程序。

【实例】先进后退。

```
void main()                          /*主程序*/
{                                    /*程序起始标志*/
drive(60,0);                         /*机器人往前走*/
sleep(5.0);                          /*延时 5 秒钟*/
```

```
drive(-60,0);                    /*机器人往后走*/
sleep(5.0);                      /*延时 5 秒钟*/
stop();                          /*暂停运行*/
}                                /*程序结束标志*/
```

这就是一个顺序结构的程序。在运行时，机器人先前进再后退。顺序结构的程序用流程图来表示，如图 3.4 所示。

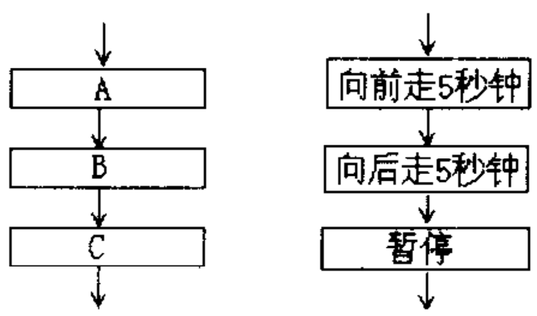



图 3.4　顺序结构程序的流程图

流程图中的 A、B、C 代表的是能完成某一个“动作”的 JC 语句，流程线的走向是程序的执行顺序。图 3.4 右边所示的流程图是“先进后退”程序的执行步骤。

注意观察机器人是否是按照我们编排的顺序执行的。

【练习】如果让机器人“先退后进”，应该怎样画出相应的流程图，并修改程序。请你试着做一下。

在顺序结构程序中，机器人完全按照你的指令一步一步地执行，有时你会觉得它有点“傻”。在机器人本身正常的情况下，当你发现机器人的“工作”与你的要求不相符时，肯定是你给它的指令有问题，你要根据出现的问题去修改程序。

2．循环结构程序

我们采用循环结构为机器人编写程序，既能使程序简洁明了，又能让机器人反复做某个动作或事情。JC 程序的循环结构我们在第二章已经见过很多了。下面我们通过几个实例对循环结构的程序加以说明。

【实例 1】无限循环程序。

我们在实例 1 程序基础上加以修改。

```
void main()                      /*主程序*/
{                                /*程序起始标志*/
while(1)                         /*控制循环*/
{                                /*循环开始标志*/
drive(60,0);                     /*机器人往前走*/
sleep(5.0);                      /*延时 5 秒钟*/
```

```
drive(-60,0);                    /*机器人往后走*/
sleep(5.0);                      /*延时 5 秒钟*/
}                                /*循环终止标志*/
}                                /*程序结束标志*/
```

运行这个程序，你会看到机器人“来来回回”走个不停。能不能让它停下来歇一歇再走呢？

【练习 1】现在我们让机器人每走一个来回，就停下来休息 2 秒钟。流程图如图 3.5 所示，现在请你按照如图 3.5 所示的流程，修改、完善实例 1 程序。

【练习 2】让机器人每做一个动作就停一下。流程图如图 3.6 所示，根据 3.6 所示流程图修改实例 1 程序。

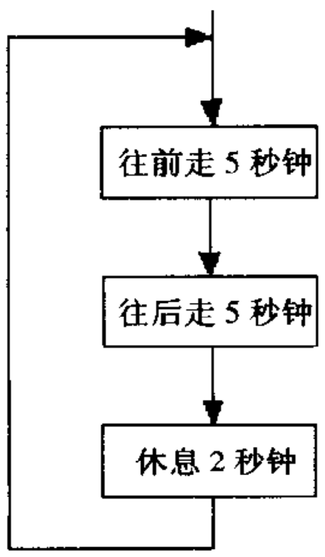



图 3.5　无限循环流程图

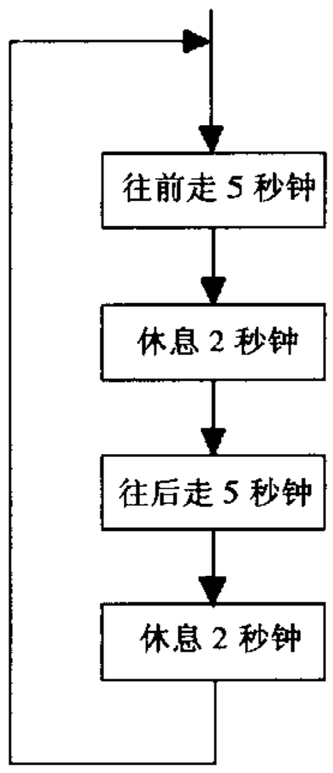



图 3.6　流程图

【实例 2】有限循环程序。让机器人走一个正方形。

让机器人把“前进，右转弯 90° ”的动作执行 4 次，就能实现。流程图如图 3.7 所示。

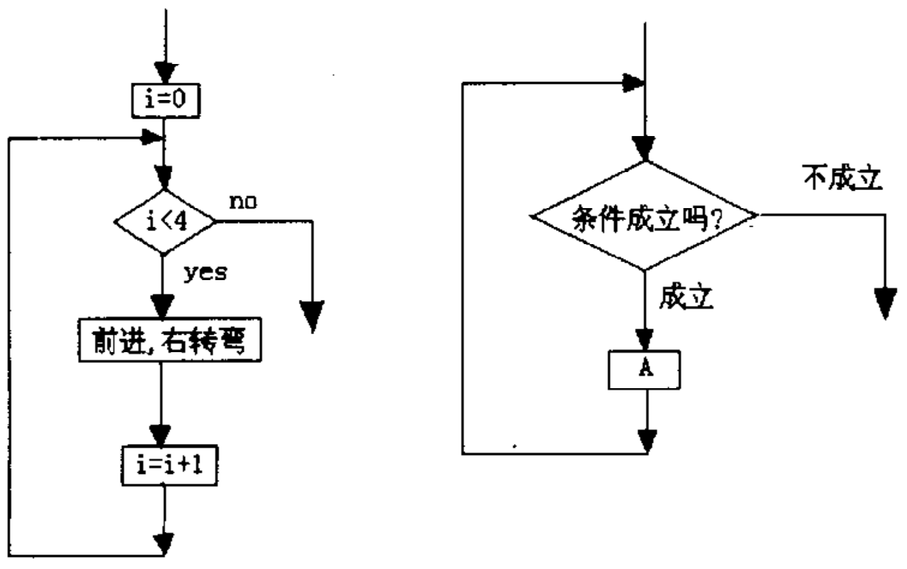



图 3.7　有限循环流程图

流程图菱形框中的内容是一个条件表达式，菱形框表示对这个条件成立与否的判断。若条件式成立，程序执行让机器人前进、右转弯的动作，否则退出循环体，转而执行循环结构后面紧接的程序内容。

图 3.7 中右边的流程图说明了一般程序的循环过程，流程图中矩形框的 A 在这里表示被重复执行的程序段，我们称这个程序段为循环体。

【实例 3】让机器人走一个正方形。

```
void main()
{
int i=0;                                    /*对整型变量说明并赋初值*/
while (i<4)                                 /*控制循环执行次数*/
{                                           /*循环体起始标志*/
drive(60,0);                                /*机器人前进*/
sleep(4.0);                                 /*延时 4 秒钟*/
drive(60,-60);                              /*原地右转*/
sleep(2.0);                                 /*延时 2.0 秒钟*/
i=i+1;                                      /*改变整型变量的值*/
}                                           /*循环体结束标志*/
stop();
}
```

运行实例 2 程序，适当调整每一次转弯的时间，使机器人每次右转约 90° 。

从实例 1 和实例 2 程序中我们发现，while(*x*)语句在执行时首先判断 *x* 的值。若为真，则执行循环体；若为假，则跳出循环体。

（1）当 while(*x*)语句中，*x* 的值为 1（即 *x* 为真）时，while 就控制程序反复执行循环体的内容。

（2）当 while(*x*)语句中，*x* 为条件式，且条件式的值为真时，执行循环体；当条件式的值为假时，程序跳出循环体。while 控制程序执行循环体的次数，由 *x* 条件式确定。

（3）无限循环多用于各种传感器对外界环境信息的处理上，如红外、光敏和碰撞等，机器人会反复检测信息、接收信息、判断信息、处理信息。

（4）有限循环多用于各种执行器上，如电机、麦克风等的控制，机器人按照指令的要求执行若干次。

3．选择结构程序

我们采用选择结构，是为了让机器人可以根据不同的外界环境条件和要求去做事

情，提高它的智能化程度。请看下面的实例。

【实例】前方有碰撞就后退。

```
void main()
{
int bump;
drive(60,0);
while(1)                                    /*用无限循环反复检测碰撞信息*/
  {
  bump=bumper();                            /*检测、保存碰撞信息*/
  if (bump==0b0011)                         /*判断是否前方有碰撞*/
    {
    drive(-60,0);                           /*如果前方有碰撞，让机器人后退*/
    sleep(2.0);                             /*延时 2 秒钟*/
    stop();                                 /*停止*/
    }
  }
}
```

这个程序的流程图如图 3.8 所示。

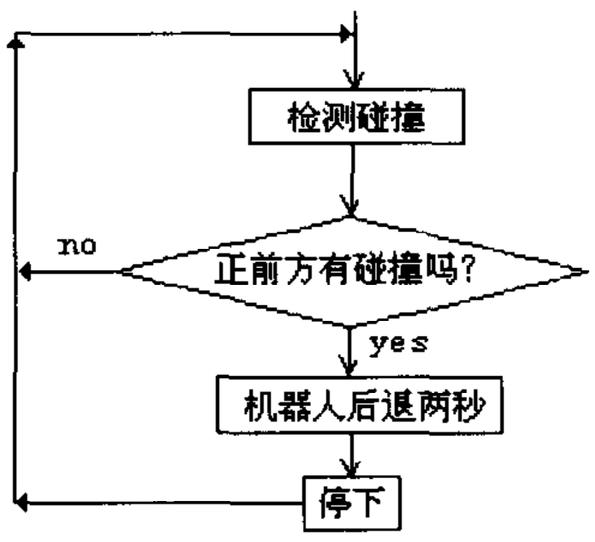



图 3.8　选择结构流程图

请注意观察机器人的运动是否达到了预期的效果。

通过这个程序，你会发现你的机器人已经具有了一定的判断能力，并且能够根据不同的外界环境信息，去做不同的事情。瞧，它是不是比以前“聪明”了？

### 3.2.2 编写程序的原则和步骤

1．编程原则

（1）简洁明了

编写的程序应该具有一定的可读性，程序的表述要简洁明了。

（2）省时高效

按照解决问题的要求，程序运行时应该尽量做到省时高效。

（3）结构清楚

程序要尽量符合结构化程序设计的基本原则。

（4）逻辑合理

程序的逻辑关系要清楚合理。

2．编程步骤

（1）明确任务

明确任务要求，是正确编写程序的前提。所以在编写程序之前，首先应明确要解决的问题和要求。

（2）画流程图

在明确任务后，最好将解决问题的步骤用流程图的方式表示出来，以便直观反映程序步骤。

（3）编写程序

按照结构化程序设计的原则，根据流程图编写程序。

（4）试验调试

在下载程序和运行程序的过程中，修改程序中的逻辑错误和语法错误。使编写的程序符和任务要求。

## 3.3 关于 JC

JC 是用于能力风暴个人机器人的专用开发系统，它是由编译系统和能力风暴操作系统两部分组成，是建立在 ANSI C 标准上的。

由于 JC 是先编译生成基于堆栈虚拟机的伪代码，然后再由能力风暴操作系统解释执行，它的优点在于：

- 允许检查运行错误。例如：JC 在运行时要检查数组下标，以防止程序错误，因此 JC 比标准 C 更“安全”。
- JC 伪代码与机器硬件无关，所以设计和移植都很方便。

- 伪代码比机器代码更简短，所以 JC 比标准 C 更简单。
- 任务切换由操作系统处理，因此可以方便地实现多任务。

在为机器人编写程序的时候，要按照 JC 的语法规则来书写程序，所以在学习程序设计的时候要注重语法规则的掌握，要注重语句和函数的书写格式，要知道使用的语句和函数在执行时会让你的机器人产生什么动作，要知道语句的功能。

下面我们介绍一些 C 语言的基本规则以及前面程序中用到的语句、函数的格式和功能。

### 3.3.1 主程序

1．格式：void main()

{

<程序内容>

}

2．功能：建立一个无返回值的程序。其中 void 表示后面的 main()程序执行后无返回值。在一个程序中主程序的标志 main()是唯一的，后面花括号中括着的是主程序的程序内容。

在对语句格式的叙述过程中千万别忘了用“<>”把被说明的内容括起来，用以说明程序的格式或内容。

### 3.3.2 整型变量类型的说明语句

1．格式：int <变量 1>,<变量 2>,<变量 3>,……,<变量 n>;

（其中 n 为自然数）

2．功能：“int”表示后面被说明的变量是整型变量。

3．说明：

（1）变量可以看作是存放数据的地方。

（2）变量的名称应由字母、数字和“_”下划线字符组成（首字符必须为字母）。

（3）变量只有被说明类型后才能使用。

（4）被 int 说明的变量可以是一个，也可以是多个，但是多个变量之间要用逗号分隔。

（5）给变量起名字的时候不能用 C 语言规定的专用词。

（6）变量被说明为整型后，变量中存放的数据只能是－32767 到 32768 范围内的整数。

（7）在为变量进行说明时也可以为变量提供数据。

（8）变量的类型有多种，我们将在后面介绍。

【实例 1】

● 对整型变量 $i$、$j$、$k$ 的说明

```
int i,j,k;
```

● 对整型变量 max、min 的说明

```
int max ,min;
```

【实例 2】在对整型变量 max、min 说明的同时为它们分别提供数据 100 和 0。

```
int max=100,min=0;
```

### 3.3.3 延时函数

1. 格式：sleep($t$);
2. 功能：等待 $t$ 秒钟后再执行后面的语句。
3. 说明：参数 $t$ 为实型数。

### 3.3.4 暂停函数

1. 格式：stop();
2. 功能：暂停程序的执行。

### 3.3.5 程序的注释

1. 格式：/*<注释内容>*/
2. 功能：对程序内容进行注释。
3. 说明：在程序执行的过程中注释内容不使机器人产生任何动作。

### 3.3.6 条件成立执行循环的 while 语句

1. 格式：while(<条件式>)

```
{
<循环体>
}
```

2. 功能：当条件成立（即条件式的值不为 0 时）执行循环体，否则（即条件式的值为 0 时）退出循环继续执行紧接此循环体的程序内容。

3. 说明：下面是机器人执行 while 语句的步骤：

（1）判断条件式的值是否为 0，若不为 0 执行步骤(2)，若为 0 执行步骤(5)。

（2）执行循环体。

（3）改变条件式的值。

（4）返回步骤(1)。

（5）执行循环体后面紧接的程序内容。

需要注意的是，如果条件式的值是个不为 0 的常量，并且没有步骤(3)，程序就会无限循环。

4．用文字描述下面这个实例中 while 语句的循环步骤。

【实例】“转圈”程序。

```
void main()
{
int i=0;                              /*对整型变量说明并赋予初值*/
while（i<4）                          /*控制循环执行{……}中的内容*/
  {                                   /*循环执行内容的起始标志*/
  drive(60,－50);                     /*左、右电机的功率级别分别为 110 和 10*/
  sleep(2.0);                         /*延时两秒钟*/
  i=i+1;                              /*改变整型变量的值*/
  }                                   /*循环执行内容的结束标志*/
stop();
}
```

（1）程序中用 while 语句来控制循环。

```
    .
    .
    .
int   i=0;
while   (i<4)                         /*控制循环的语句*/
  {                                   /*循环执行内容的起始标志*/
<让机器人转一圈的程序段>
i=i+1;                                /*改变控制循环的变量*/
}                                     /*循环执行内容的结束标志*/
  .
  .
  .
```

（2）用文字描述循环步骤。

a 用 *i*=0 给变量 *i* 赋初值；

b 判断条件式 *i*<4 是否成立,若成立执行步骤 c,若不成立执行步骤 f；

c 执行循环体，即让机器人转一圈的程序段；

d 执行 $i=i+1$，即把变量 $i$ 的值加 1 后再送到变量 $i$ 中；

e 返回步骤 b。

f 执行循环体后面的语句。

如图 3.9 所示的流程图，体现了“转圈”程序 while 语句控制循环的过程。

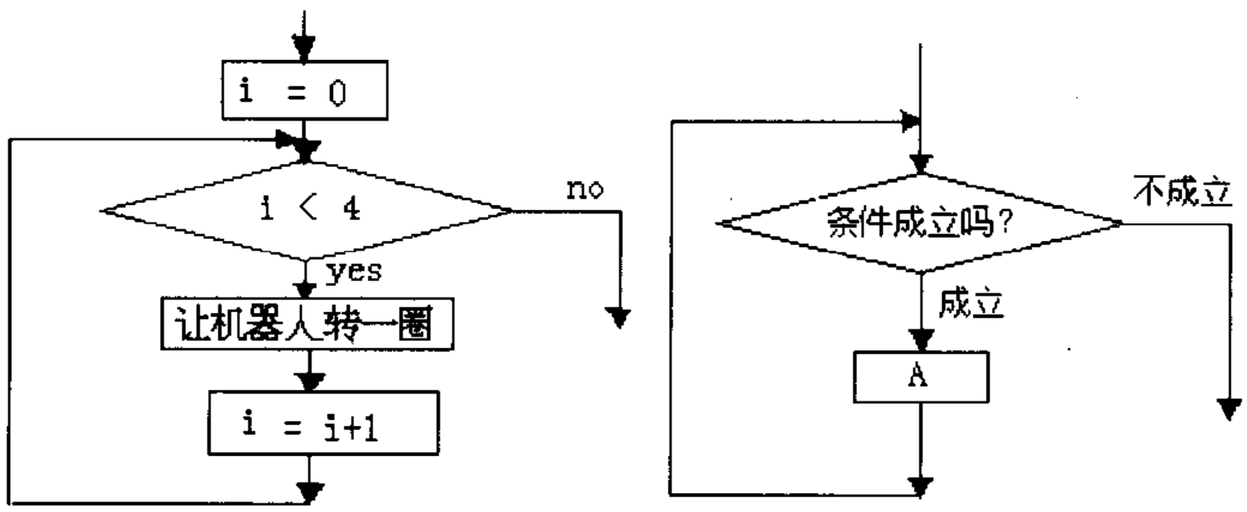



图 3.9　程序循环结构的流程图

图 3.9 中左边的流程图是对“转圈”程序循环结构的描述。

### 3.3.7　赋值语句

1．格式：<变量>=<表达式>

2．功能：计算赋值号“=”右边表达式的值，并将计算的结果赋值给赋值号左边的变量。

3．说明：

（1）表达式指的是用运算符号将函数、变量、常量连接起来的式子。

（2）一般算术表达式的运算顺序如表 3.1 所示。

**表 3.1　　运算顺序表**

| 运算顺序 | 运算 | 运算符号 |
|---|---|---|
| 1 | 括号 | （） |
| 2 | 函数 | 函数名 |
| 3 | 乘法、除法 | *、/ |
| 4 | 加法、减法 | +、− |

（3）赋值语句中的“=”是赋值号不是等号，它与相等的含义不同。例如，在数

学中 i=i+1 是不成立的，但在程序设计中是允许的。

（4）在有乘法运算的表达式中乘号“*”必须要写。

（5）表达式中的括号只能用小括号。

### 3.3.8 具有选择结构的 if 语句

1．格式 1：if (<条件式>)

{<程序段 1>}

else

{<程序段 2>}

<程序段 3>

2．功能：当条件成立时执行程序段 1，然后执行程序段 3，否则执行程序段 2，然后执行程序段 3。

3．格式 2：if (<条件式>)

{<程序段 A>}

<程序段 B>

4．功能：当条件成立时执行程序段 A，然后执行程序段 B，否则跳过程序段 A 执行程序段 B。

5．说明：

格式 2 也可以写成以下格式：

if (<条件式>) {<程序段 A>}

{<程序段 B>}

当程序段中的语句不止一条时要把该程序段用花括号“{}”括起来。

### 3.3.9 关系表达式和逻辑表达式

if 语句和 while 语句中的条件式具有布尔运算结果，即取值为 0 或 1 的表达式，关系表达式和逻辑表达式都是这种结果为 0 或 1 的表达式。

1．用关系运算符连接算术表达式构成的式子是关系表达式。关系运算符如表 3.2 所示。

**表 3.2　　关系运算符表**

| 运算 | 大于 | 小于 | 大于等于 | 小于等于 | 等于 | 不等于 |
|---|---|---|---|---|---|---|
| 运算符 | > | < | >= | <= | == | != |

2．用逻辑运算符连接关系表达式构成的式子是逻辑表达式。逻辑运算符如表 3.3

所示。

**表 3.3　　逻辑运算符表**

| 运算 | 非 | 与 | 或 |
|---|---|---|---|
| 运算符 | ! | && | \|\| |
| 运算的优先级别 | 高 | 中 | 低 |

逻辑运算的真值表，如表 3.4 所示。

**表 3.4　　逻辑运算真值表**

| 关系式 a | 关系式 b | a!（非） | a&&b（与） | a\|\|b（或） |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

真值表中的 1 表示真，即条件成立；0 表示假，即条件不成立。

### 3.3.10　输出语句

1．格式：printf(“<输出格式>”,<变量 1 或表达式 1>,<变量 2 或表达式 2>,……);

2．功能：按照输出格式的规定输出变量或表达式的值。

3．说明：

（1）语句格式中的变量或表达式是可选项，如

printf(“Hello!”);

输出结果：　Hello!

（2）在语句的输出格式中可以加入换行标志，如

- printf(“Hello! Welcome you!”);

  输出结果：Hello! Welcome you!

- Printf(“Hello! \n Welcome you!”);

  输出结果：Hello!

  　　Welcome you!

比较上面两个 printf 语句的输出结果，你能体会到“\n”是语句的换行标志。

（3）在语句的输出格式中，应该规定变量或表达式的值以什么数据格式输出。

● int R=8,L=13;

printf(“R=%d　L=%d\n”,R,L)

输出结果：R=8　L=13

观察上面 printf 语句的输出结果可以看出输出格式中前后两个“%d”分别规定了变量 R 和 L 以十进制整数的格式输出，同时还指定了输出的顺序和位置。

● 将 printf 语句修改为：

printf(“R=%b　L=%b\n”,R,L)

输出结果：R=0100　L=1101

观察修改后的 printf 语句，输出格式中的格式符“%d”被改成了格式符“%b”，从而使得变量 R 和 L 的值转换成二进制的格式输出。

（4）数据转换格式符的规定如表 3.5 所示。

**表 3.5　　数据转换格式表**

| 转换格式符 | 转换后的格式 |
|---|---|
| %d | 十进制整数 |
| %b | 二进制整数 |
| %x | 十六进制整数 |
| %c | ASCII 字符 |
| %f | 浮点数 |

## 3.4　充分发挥机器人的本领

你的机器人已经学会了一些基本的本领,现在我们要教它综合运用这些本领去完成任务。

### 3.4.1　任务：“走向光明”

编一个程序让机器人向光线强的方向走，我们给这个程序起个名字叫“走向光明.c”(C 语言编写的程序文件的扩展名为“.c”)。

1．思路：当程序检测到光敏传感器的信息后，判断左、右方向哪边光线强，若左面光线强向左面走，右面光线强向右面走。部分程序结构流程图如图 3.10 所示。

在如图 3.10 所示的流程图菱形框中，如果条件成立，执行矩形框中程序段 A，否则执行矩形框中的程序段 B。程序根据条件成立与否执行了不同的内容后继续执行下

面的程序内容。

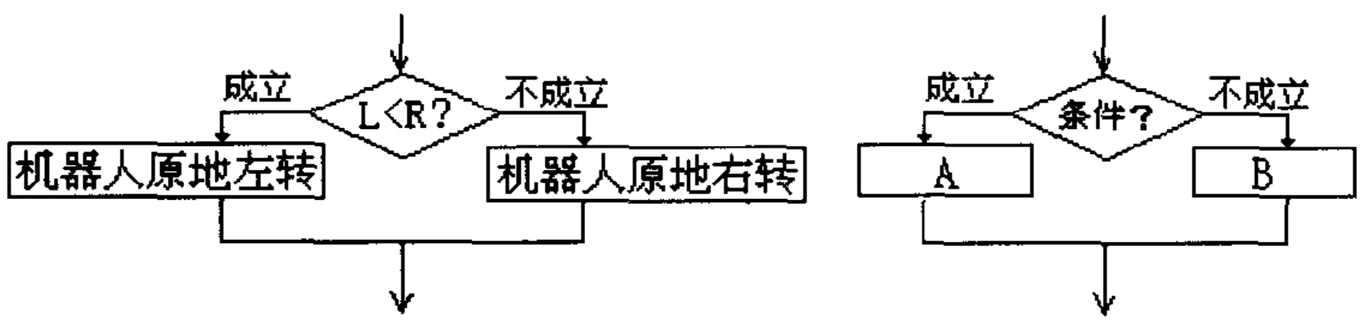



图 3.10 “走向光明”选择结构流程图

2.“走向光明”程序用到的有关语句：

```
.
.
.
if (L<R )
   {
   drive(0,80);                          /*机器人原地左转*/
   sleep(____);                          /*机器人左转时间*/
   }
else
   {
   drive(0，-80);                        /*机器人原地右转*/
   sleep(____);                          /*机器人右转时间*/
   }
.
.
.
```

在执行程序的过程中，机器人根据条件判断的结果，去分别执行不同的程序段。这种程序结构我们前面介绍过叫选择结构，它是 3 种结构化程序设计之一。

3．试试看：请你给“走向光明”的程序填空。

【实例】

```
void main()
{
int _______;                             /*对整型变量 R、L 说明*/
```

```
while(1)                                    /*用于控制循环的语句*/
 {
  R=_____;                                  /*将右光敏传感器的检测值存放到变量 R 中*/
  L=_____;
  printf("R=%d     L=%d\n",R,L);            /*输出检测值*/
  if (L<R)
     {
     drive(0,80);                           /*机器人原地左转*/
     sleep(_____);                          /*机器人左转时间*/
      }
    else
     {
     drive(0,-80);                          /*机器人原地右转*/
     sleep(_____);                          /*机器人右转时间*/
      }
    drive(80,0);                            /*机器人向前直行*/
    sleep(_____);                           /*机器人直行时间*/
    stop();
    }
}
```

4．观察、调试

（1）下载程序、执行程序并观察机器人的行走过程思考程序的执行过程。

（2）调整程序中的参数使机器人的行走更符合“走向光明”的要求。

（3）修改程序使机器人朝光线弱的地方走。

5．试试看：

任务：编写程序“找光源”（程序名为“找光源.c”）

要求：在一个光线基本相同的空间地面上放一盏小灯，让机器人找到小灯后，在离小灯一定的距离停下。

步骤：

（1）在没有开灯的情况下，在多个位置用机器人测试一下左右环境光线的平均值。把多点测试的平均值记录下来备用。

（2）打开小灯，在以小灯为圆心，半径为 R 的圆上均匀选几个点，让机器人面

向小灯，测试左右环境光线的平均值。把几个测试点的平均值记录下来备用。

（3）“找光源”程序的部分流程图，如图 3.11 所示。

（4）按照任务、要求编写“找光源”的程序。

（5）调试程序，修正参数使程序的运行趋于合理。

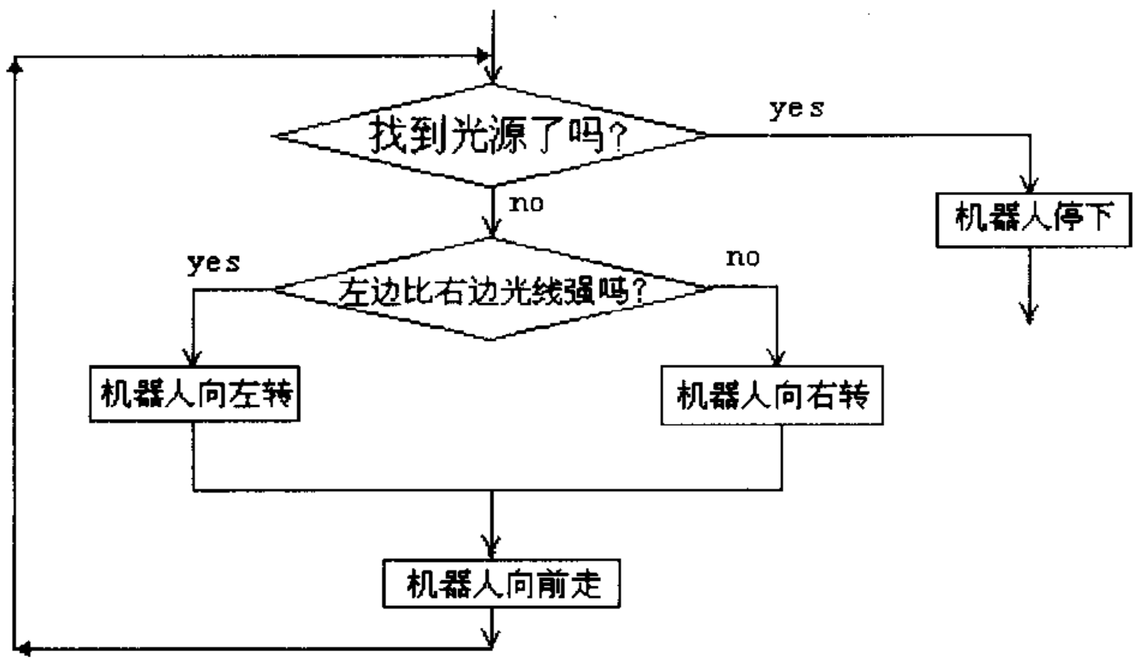



图 3.11　找光源流程图

### 3.4.2　任务：“躲避障碍”

编一个程序让机器人在前进时能避开遇到的障碍物。

1．思路：通过红外传感器检测前方、左前、右前是否有障碍物，没有障碍物继续前进，有障碍物绕开再前进。

2．步骤：

（1）看懂机器人避开右前方障碍物程序的部分流程图，如图 3.12 所示。

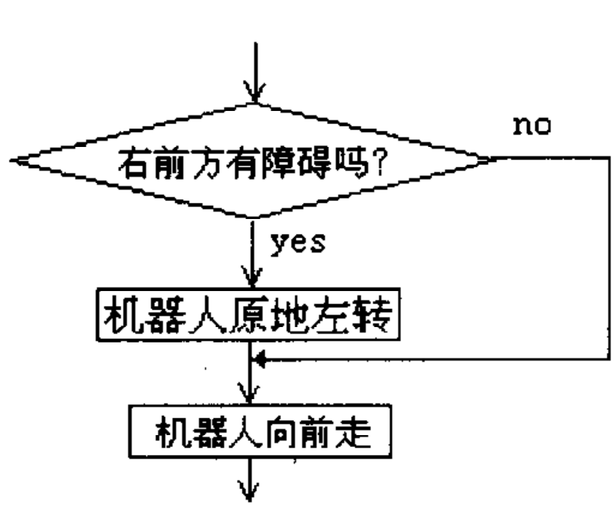



图 3.12　右前方避障流程图

（2）给右前方避障程序填空，程序名为“避右障.c”。

【实例】

```
void main ( )
{
int irv;
while (1)
   {
   irv=ir_detest();
   printf("irv=%b\n",irv);
   if (irv==0b01)                          /*右前方有障碍*/
      {
      drive(0,80);                         /*向左转*/
      sleep(____);                         /*左转时间*/
      }
   drive(_____);                           /*向前走*/
   sleep(_____);                           /*向前走的时间*/
   stop();
   }
}
```

（3）修改上面给出的“避右障”程序，使机器人在行进中可以避开左前方、右前方和正前方的障碍。修改后的程序名为“避.c”。

3．试试看:

（1）看懂机器人避障程序的部分流程图，如图 3.13 所示。编写避障程序。

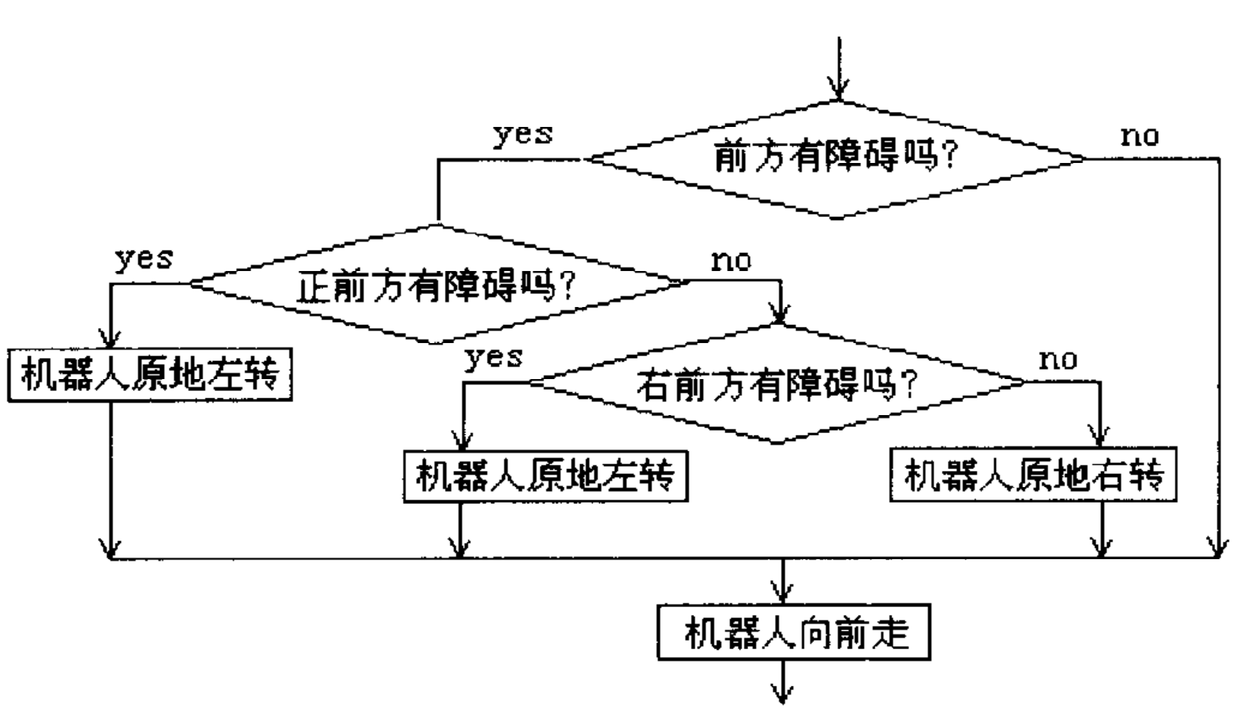



图 3.13　避障程序部分流程图

（2）编写程序如下：

【实例】避障程序

```
void main()
{
int irv;
while(1)
   {
   irv=ir_detect();
   printf("irv=%b\n",irv);
if (irv!=0b00)
   {
   if (irv==0b11)                    /*正前方有障碍*/
      {
      drive(0,80);                   /*向左转 90° */
      sleep(___);                    /*时间*/
      }
      else if (irv=_____)            /*右前方有障碍*/
            {
            drive(0,80);             /*向左转 45° */
            sleep(____);             /*左转时间*/
            }
         else if (irv=_____)         /*左前方有障碍*/
               {
               drive(0,-80);         /*向右转 45° */
               sleep(____);          /*右转时间*/
               }
      }
      drive(80,0);                   /*向前走*/
      sleep(____);                   /*向前走时间*/
      stop();
   }
}
```

（3）试验、调试避障程序。

### 3.4.3 任务：避障寻光

完善“找光源”的程序，使机器人在寻找光源的过程中能够避开障碍。程序名为“避障寻光.c”。

1．思路：机器人在检测光源的同时还要避开障碍，这就要求机器人要同时做几件事情。

2．程序中用到的语句和函数

要实现能力风暴的多任务，只要调用 start_process 函数就能实现。

（1）函数格式：start_process(<函数标识>,t)

（2）功能：为标识的函数建立一个执行时间为 t 毫秒的进程。

（3）说明：进程函数在程序执行的过程中，将时间分成非常小的时间段，每段时间转去执行标识的函数，规定时间到时后记录该函数的当前位置等，以便下次这个函数执行时从该记录位置继续执行。反复地按顺序执行每个时间段规定的函数，我们称这种程序的执行方式为进程。在进程函数中参数 t 是可选项，当不选 t 时,进程默认执行时间为 5 毫秒。

3．步骤：

（1）把程序“找光源”和“避障”以函数的方式放在“避障寻光”中，并准备在主程序中调用它们。

【实例】

```
void light()                          /*找光源函数*/
{
int R,L;                              /*对整型变量 R、L 说明*/
while(1)                              /*用于控制循环的语句*/
{
R=analog(0);                          /*将右光敏传感器的检测值存放到变量 R 中*/
L=analog(1);
if (L<190 || R<190)
stop();
else
   {
      if (L<R)
         {
```

```
    drive(0,80);                    /*机器人原地左转*/
    sleep(0.3);                     /*机器人左转时间*/
    }
else
    {
    drive(0,-80);                   /*机器人原地右转*/
    sleep(0.3);                     /*机器人右转时间*/
    }
drive(80,0);                        /*机器人向前直行*/
        sleep(1.0);                 /*机器人直行时间*/
        stop();
      }
   }
}

void irtest()  .                    /*避障函数*/
{
int irv;
while(1)
   {
   irv=ir_detect();
   printf("irv=%b\n",irv);
if (irv!=0b00)
   {
   if (irv==0b11)                   /*正前方有障碍*/
      {
      drive(0,80);                  /*向左转 90° */
      sleep(0.6);                   /*时间*/
      }
   else if (irv==0b01)              /*右前方有障碍*/
         {
         drive(0,80);               /*向左转 45° */
```

```
        sleep(0.4);                    /*左转时间*/
        }
      else if (irv==0b10)              /*左前方有障碍*/
          {
          drive(0,-80);                /*向右转 45° */
          sleep(0.4);                  /*右转时间*/
          }
   }
drive(80,0);                           /*向前走*/
   sleep(1.0);                         /*向前走时间*/
   stop();
   }
}

void main()                            /*主程序*/
{
start_process(light());                /*调用找光源函数的进程*/
start_process(irtest());               /*调用避障函数的进程*/
while(1)
   {
   drive(80,0);
   sleep(2.0);
   }
}
```

在此程序中，我们是以建立进程函数的方式来编写“避障寻光”程序的，主程序中两个进程的建立构成了一个多任务的进程表，主程序执行时会按照进程表规定的顺序和每个进程规定的执行时间（本程序使用的是默认时间 5ms），反复执行进程表中各个程序。

（2）请你下载运行“避障寻光”程序。

（3）调试程序使其既能寻找光源又能避开障碍。

### 3.4.4 任务：碰撞躲避

编写避开来自正前方、左前方和右前方碰撞的程序，程序名为“避碰.c”。

1. 思路：

（1）在机器人行进的过程中遇到来自前方碰撞的避碰方法：

- 检测前方的碰撞；
- 向后退一段距离；
- 向左转 90°；
- 继续前进。

（2）在机器人行进的过程中遇到来自左前方碰撞的避碰方法：

- 检测左前方的碰撞；
- 向后退一段距离；
- 向右转 45°；
- 继续前进。

（3）在机器人行进的过程中遇到来自右前方碰撞的避碰方法：

- 检测右前方的碰撞；
- 向后退一段距离；
- 向_____转 45°；
- 继续前进。

以上避碰策略仅供参考，你也可以自己设计避碰策略。

2. 步骤：

（1）看懂机器人避开正前方碰撞程序的部分流程图，如图 3.14 所示。

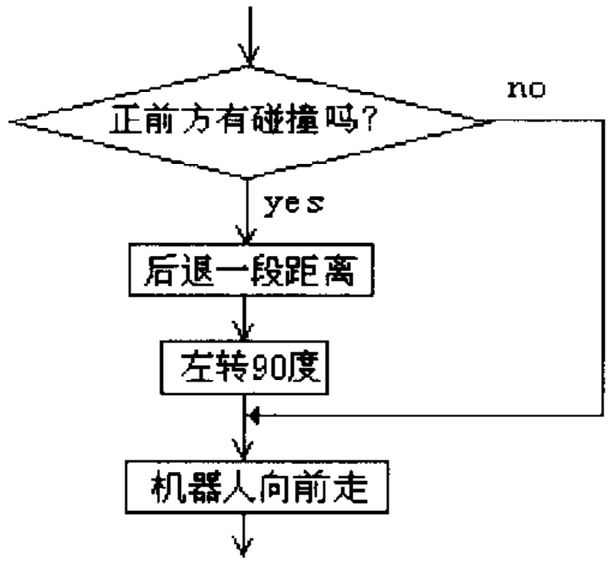



图 3.14 正前方避碰程序流程图

（2）编写避开正前方碰撞的程序，程序名为“正前方避碰.c”。

【实例 1】

```
void main()
{
int bumpv;
while(1)
   {
   bumpv=bumper();                          /*将碰撞的检测值存放到变量 bumpv 中*/
   printf("bump=%d    %b\n",bumpv,bumpv);
   if (bumpv==0b0011)
      {
      drive(-80,0);
      sleep(_____);                         /*后退一段距离所用的时间*/
      drive(0,80);                          /*左转*/
      sleep(_____);                         /*左转 90° 所用的时间*/
      }
   drive(80,0);                             /*前进*/
   sleep(0.5);                              /*前进时间*/
   stop();
   }
}
```

（3）看懂机器人避开来自正前方、左前方和右前方的碰撞程序的部分流程图，如图 3.15 所示。

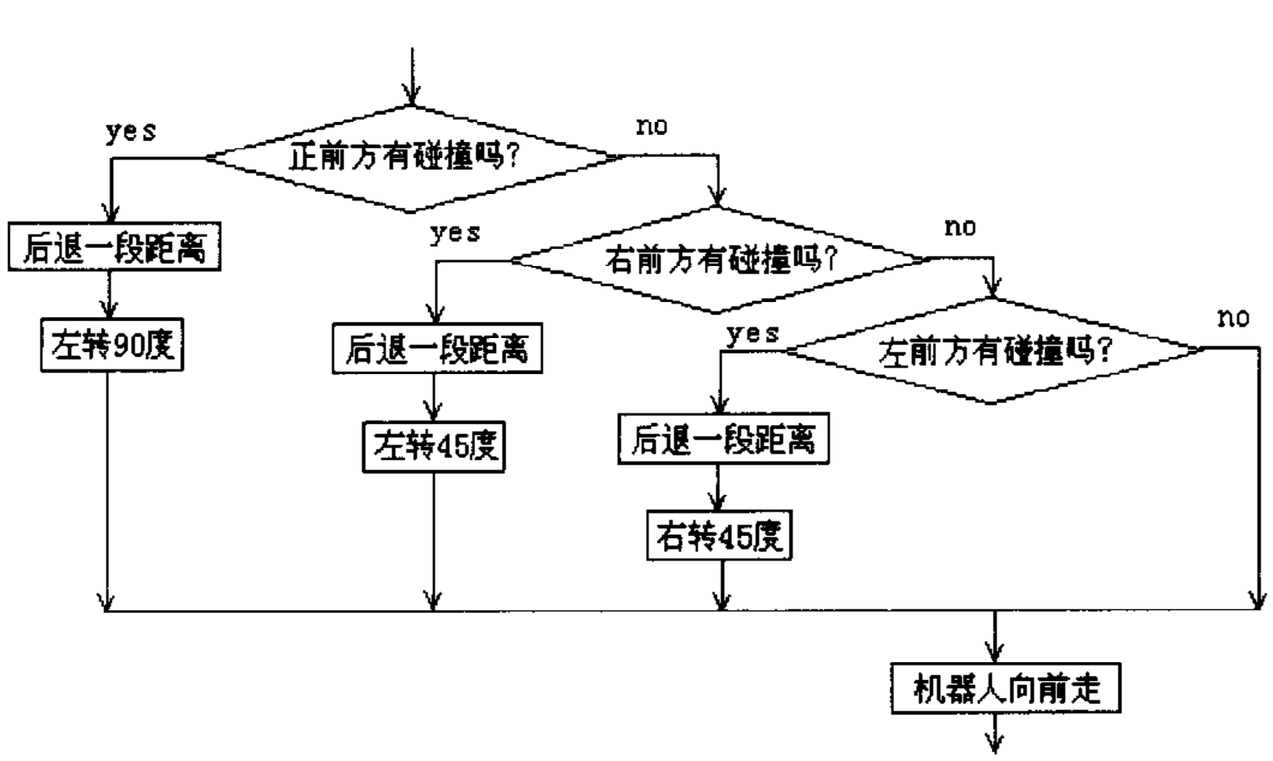



图 3.15　前方避碰程序流程图

（4）修改上面的程序使其能够避开来自正前方和左、右前方的碰撞。修改后的程序名为“前方避碰.c”。

【实例2】

```
void main()
{
int bump;
while(1)
   {
   bump = bumper();                    /*检测碰撞*/
if (bump==0b0000)                      /*没有碰撞*/
   {
   drive(100,0);                       /*向前行进*/
   sleep(0.5);                         /*行进时间*/
   }
else
   {
   drive(-80,0);                       /*后退一段距离*/
   sleep(0.2);
   if (bump==0b0011)                   /*正前方有碰撞吗？ */
      {
      drive(0,80);                     /*左转 90° */
      sleep(0.4);
      }
else if (bump==0b0001)                 /*右前方有碰撞吗？ */
      {
      drive(0,80);                     /*左转 45° */
      sleep(0.15);
      }
   else if (bump==0b0010)              /*左前方有碰撞吗？ */
         {
         drive(0,-80);                /*右转 45° */
         sleep(0.15);
```

```
            }
        }
      stop();
      }
    }
```

### 3.4.5 任务："避碰避障寻光"

修改前面编写过的"避障寻光"程序使其在寻光的过程中不仅可以避障而且可以避开碰撞。

1. 思路：将"避碰"的程序作为函数进程放在"避障寻光"程序中。

2. 避碰避障寻光程序（供参考）。

【实例】

```
void bumptest()                          /*避碰函数*/
{
int bump;
   while(1)
      {
      bump=bumper();                     /*检测碰撞*/
      if (bump==0b0000)                  /*没有碰撞*/
         {
         drive(100,0);
         sleep(0.5);
         }
else
    {
   drive(-80,0);                         /*后退一段距离*/
   sleep(0.2);
   if (bump==0b0011)                     /*正前方有碰撞吗？ */
      {
      drive(0,80);                       /*左转 90° */
      sleep(0.4);
      }
```

```
else if (bump==0b0001)              /*右前方有碰撞吗？ */
    {
    drive(0,80);                    /*左转 45° */
    sleep(0.15);
    }
    else if (bump==0b0010)          /*左前方有碰撞吗？ */
        {
        drive(0,-80);               /*右转 45° */
        sleep(0.15);
        }
    }
    stop();
  }
}

void light()                                /*找光源函数*/
{
int R,L;                                    /*对整型变量 R、L 说明*/
while(1)                                    /*用于控制循环的语句*/
  {
  R=analog(0);                              /*将右光敏传感器的检测值存放到变量 R 中*/
  L=analog(1);
if (L<190 || R<190)
  stop();
else
  {
  if (L<R)
    {
    drive(0,80);                            /*机器人原地左转*/
    sleep(0.3);                             /*机器人左转时间*/
    }
  else
```

```
        {
        drive(0,-80);                    /*机器人原地右转*/
        sleep(0.3);                      /*机器人右转时间*/
        }
    drive(80,0);                         /*机器人向前直行*/
    sleep(1.0);                          /*机器人直行时间*/
    stop();
    }
}
}

void irtest()                            /*避障函数*/
{
int irv;
while(1)
   {
   irv=ir_detect();
   printf("irv=%b\n",irv);
   if (irv!=0b00)
      {
      if (irv==0b11)                     /*正前方有障碍*/
         {
         drive(0,80);                    /*向左转 90° */
         sleep(0.6);                     /*时间*/
         }
   else if (irv==0b01)                   /*右前方有障碍*/
         {
         drive(0,80);                    /*向左转 45° */
         sleep(0.4);                     /*左转时间*/
         }
      else if (irv==0b10)                /*左前方有障碍*/
            {
```

```
                drive(0,-80);                    /*向右转 45° */
                sleep(0.4);                      /*右转时间*/
                }
        }
        drive(80,0);                             /*向前走*/
        sleep(1.0);                              /*向前走时间*/
        stop();
    }
}

void main()                                      /*主程序*/
{
start_process(bumptest());                       /*调用避碰函数的进程*/
start_process(light());                          /*调用找光源函数的进程*/
start_process(irtest());                         /*调用避障函数的进程*/
while(1)
   {
   drive(80,0);
   sleep(2.0);
   }
}
```

3．将修改后的程序以“避碰避障寻光.c”为文件名存盘。

4．注意：“避碰避障寻光”程序是一个多进程的程序，因为进程在程序执行时是并行的所以在编写程序的时候要注意不要在多个进程中直接使用同一个设备，例如电机、LCD、麦克风等，即一个设备不要同时被两个或两个以上的进程访问，例如两个进程都通过 drive 函数驱动电机的转速，那么在某一个时刻一个进程让它正转，而另一个进程让它反转，那结果可能会是一团糟。

5．建立一个函数，完成电机驱动的过程，在“避碰避障寻光”程序的 3 个进程中通过调用该函数的方法来实现电机的驱动。

【实例】

```
int trans=100,rot=0;
int bumpv;
```

```
int irv;
int l,r;
void set_drive(int a,int b)                    /*新建立的电机驱动函数*/
{
trans=a;rot=b;
}
void bumptest()                                /*避碰函数*/
{
   while(1)
{
bumpv=bumper() ;                               /*将碰撞的检测值存放到变量 bumpv 中*/
if (bumpv==0b0011)                             /*正前方有碰撞*/
   {
   set_drive(－80,0);                          /*调用新建的电机驱动函数*/
   sleep(0.2);                                 /*后退一段距离所用的时间*/
   set_drive(0,80);                            /*左转*/
   sleep(0.35);                                /*左转 90° 所用的时间*/
   }
else if (bumpv==0b0001)                        /*右前方有碰撞*/
        {
        set_drive(－80,0);
        sleep(0.2);
   set_drive(0,80);                            /*左转*/
   sleep(0.18);                                /*左转 45° 所用的时间*/
   }
else if (bumpv==0b0010)                        /*左前方有碰撞*/
        {
        set_drive(－80,0);
        sleep(0.2);
        set_drive(0,－80);                     /*右转*/
        sleep(0.18);                           /*右转 45° 所用的时间*/
        }
```

```
        set_drive(100,0);                    /*前进*/
        sleep(0.5);
        stop();
        }
}

void light()                                 /*找光源函数*/
{
.
.
}

void irtest()                                /*避障函数*/
{
.
.
.
}

void main()                                  /*主程序*/
{
 start_process(bumptest());                  /*调用避碰函数的进程*/
 start_process(light());                     /*调用找光源函数的进程*/
 start_process(irtest());                    /*调用避障函数的进程*/
 while(1)
    {
    drive(trasn,rot);
    sleep(0.2);
    }
}
```

请你继续完成程序的修改。

### 3.4.6 任务：机器人电子琴

编写一个机器人电子琴程序，使机器人在碰到障碍时能发出简谱的 8 个音，如图 3.16 所示，8 个方向对应的 8 个音阶。

1．思路：机器人可以感知 8 个方向的碰撞，检测碰撞的方向，发出不同的声音。

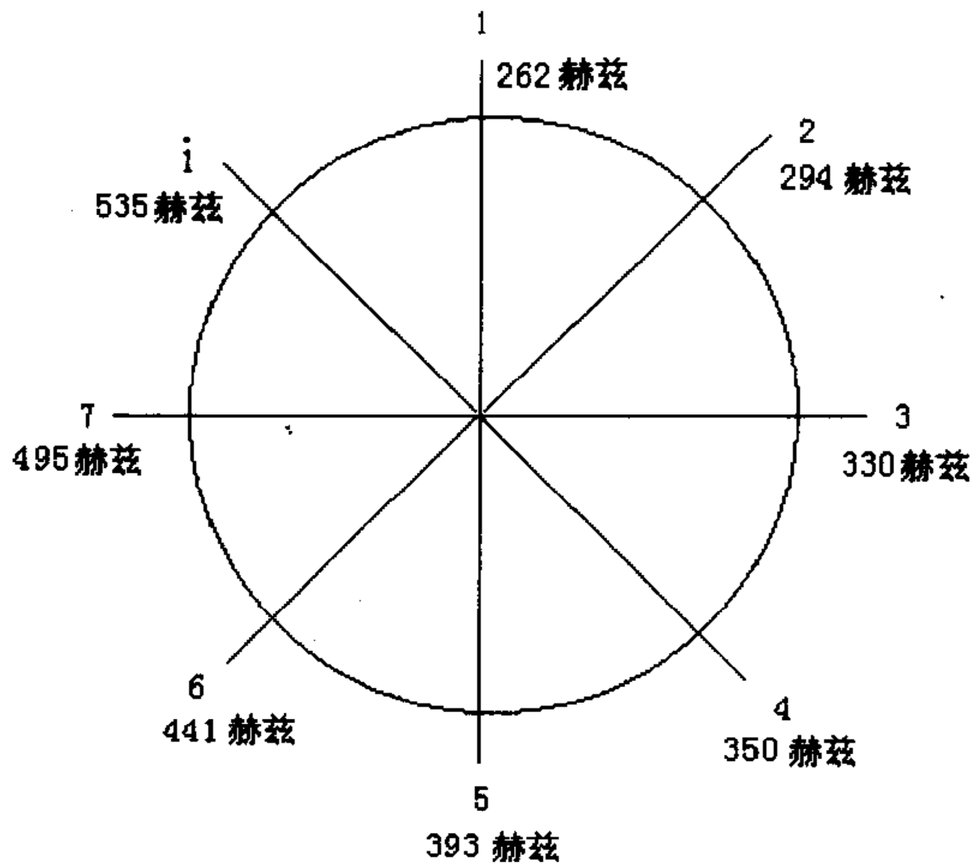



图 3.16　机器人电子琴示意图

2．输入音阶程序。

【实例】音阶程序（程序名为“音阶.c”）

```
void main()
{
int i;
for (i=0;i<4;i++)                    /*循环 4 次下面的程序段*/
   {
   tone(262.0,1.0);                  /*简谱 dou 的音*/
   tone(294.0,1.0);                  /*简谱 rai 的音*/
   tone(330.0,1.0);                  /*简谱 mi 的音*/
   tone(350.0,1.0);                  /*简谱 fa 的音*/
   tone(393.0,1.0);                  /*简谱 sou 的音*/
   tone(441.0,1.0);                  /*简谱 la 的音*/
   tone(495.0,1.0);                  /*简谱 xi 的音*/
```

```
    tone(535.0,1.0);                    /*简谱 dou 的高音*/
    }
}
```

运行这个程序，听一听这段音乐反复了几次。

3．程序中用到的语句和函数

在“音阶”的程序中，我们是用 for 语句来控制循环过程的。

（1）格式：for (<表达式 1>;<条件式>;<表达式 2>)

```
        {
        <循环体>
        }
```

（2）功能：当条件式成立的时候执行循环,条件不成立的时候退出循环。

（3）说明：语句中的<表达式 1>是用来给控制循环的变量提供初值的赋值表达式，<表达式 2>是用来改变循环变量的值的表达式。for 语句控制循环的流程图如图 3.17 所示。

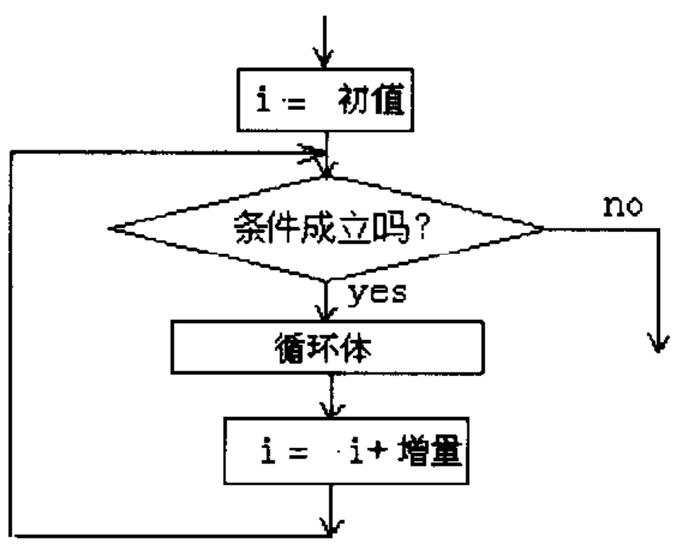



图 3.17 for 语句的循环控制流程图

4．参照例 12 和例 16 完成本程序。

### 3.4.7 任务：“听令行事”

让机器人“听”到发令后再开始“执行任务”。

1．思路：在“避碰避障寻光”程序前加上一个声音检测程序，如图 3.18 所示。

2．修改、完善“避碰避障寻光”程序。

3．试验、调试“避碰避障寻光”程序，达到任务要求。

你的机器人伙伴是不是非常棒？它现在已经学得很“聪明”，会利用各种“感觉”系统执行任务、完成工作了。

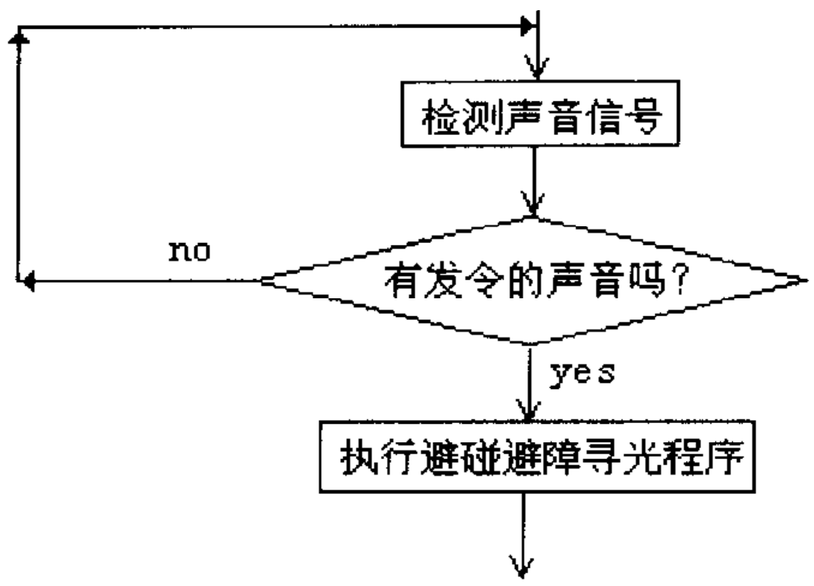



图 3.18　声音检测流程图

## 3.5　习题

1．让你的机器人先原地右转，再原地左转。

要求：根据任务先画出流程图，然后再编写程序。

2．让你的机器人前进 3 秒钟，原地右转，再后退 3 秒钟，原地左转。

要求先完成如图 3.19 所示的流程图，再根据流程图编写程序。

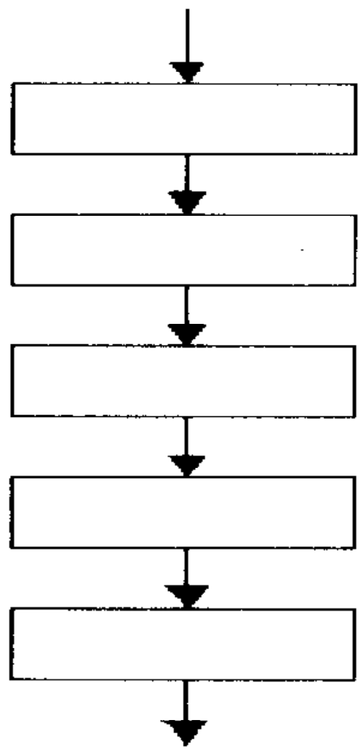


图 3.19　框图

3．现在请你编一个程序，让机器人每走一个来回停下来并“通知”你一声。要求先画出流程图，再编写程序。

4．让机器人先右转弯走个正方形，再左转弯走个正方形。流程图如图 3.20 所示。

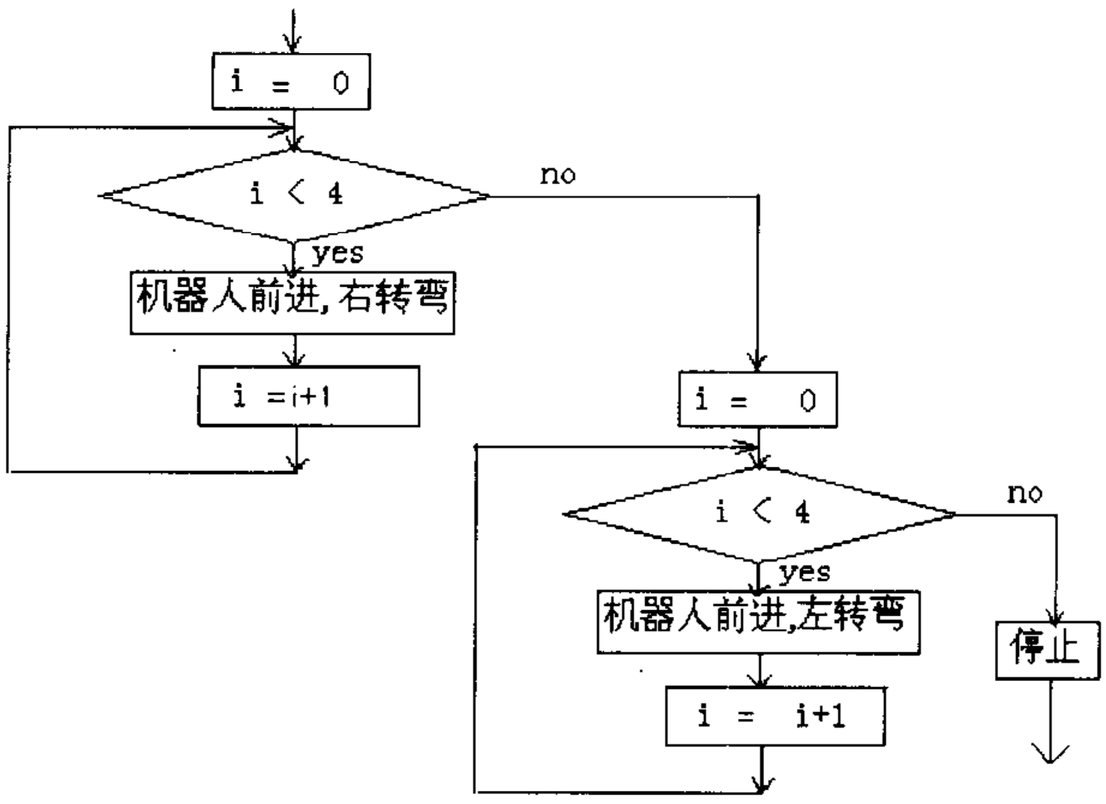



图 3.20 右正方形、左正方形流程图

请你根据流程图完成程序，并通过机器人来验证。

5. 用循环语句编程，让机器人走出 6 级台阶形。先画流程图，再编写程序。

6. 编程当机器人前进时，前方遇到碰撞，让机器人先后退一步，再右转或左转 90° 继续前进。流程图如图 3.21 所示。

要求:参照流程图完成程序。

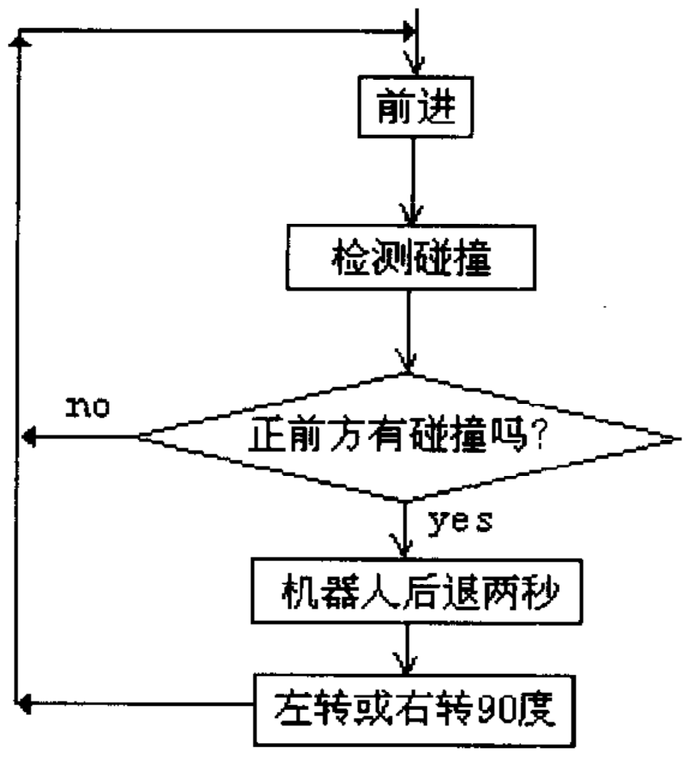



图 3.21 前方碰撞转弯再走

7. 要求参照流程图 3.22 所示编写程序，让机器人在前、左前、右前分别遇到碰撞时，躲开碰撞继续前进。

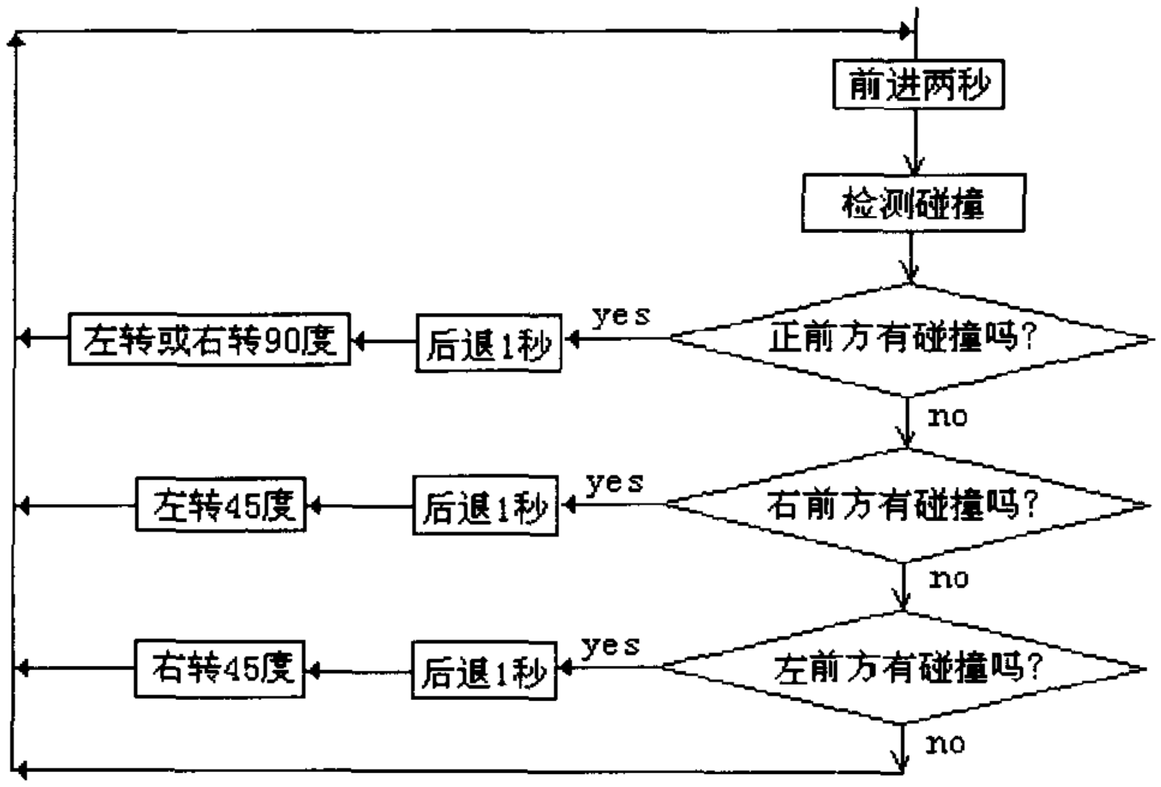



图 3.22　躲开碰撞流程图

8．编程让机器人能够避开来自 8 个方位的碰撞，程序名为“八方避碰.c”。

9．利用机器人的碰撞开关，编一个机器人八音盒的程序。

要求：8 个不同方向“感觉”到碰撞，发出 8 种不同的声音。

10．让机器人在背后遇到碰撞时，发出报警声。

11．修改“避碰避障寻光”程序，使机器人在发现光源时能奏出一段音乐，以示寻到光源。

到目前为止，你已经学会了为机器人编程，掌握了与机器人进行交流的方法，你的机器人伙伴也比以前更“聪明”了。

现在你可以带你的机器人去参加比赛了，通过参赛你将展示你的聪明才智与你的创造精神，与机器人一起成长。

# 第4章 看谁的机器人本领大

经过前3章对能力风暴个人机器人的初步探索，你已经熟悉和掌握了它的基本操作方法以及为能力风暴个人机器人编写程序的方法。现在我们要和你的机器人伙伴一起去实施迷人的机器人项目，参加精彩的机器人比赛。

## 4.1 让机器人成为你的好伙伴

现在，你有了一个可以朝夕相处、既“听话”又“懂事”的好朋友——机器人，让它伴随你成长，让它成为你的好伙伴。

现在，我们再来看一些有趣的机器人应用程序，希望能对你有所启发。

### 4.1.1 把机器人“领”回家

假如你有一个小弟弟（小妹妹），你可以带他（她）出去玩，或者手拉手把他（她）领回家。你的机器人伙伴最“听”你的话，现在就请你把它“领”回家。

1．编程思路：利用机器人的“视觉”系统，让它在一定范围内，“盯住”目标并跟着目标前进；当机器人找不到目标或者碰到障碍时，它就停下来“等待”。

2．提示：利用红外传感器检测目标。当红外传感器检测到目标时，机器人就跟着目标“走”；当碰到障碍或检测不到“目标”时，机器人就停下来。

3．输入程序。

【实例】跟随目标前进。

```
void follow()
{
int ir=0;                                   /*定义红外检测变量*/
int bmp=0;                                  /*定义碰撞检测变量*/
int old_bmp=0;                              /*定义前一次碰撞检测结果*/
int fol_trans_def=80;                       /*预设前进速度*/
```

```
int fol_rot_def=45;                          /*预设转弯速度*/
printf("Follow\n");                          /*在 LCD 显示 "Follow" */
while(1)
  {
  ir=ir_detect();                            /*取红外系统检测结果*/
  bmp=bumper()&0b0011;                       /*检测前方的碰撞*/
  if (old_bmp&&(!bmp))                       /*如果连续两次碰撞*/
     sleep(0.5);                             /*等一会儿*/
  else if (bmp)                              /*如果前方有碰撞*/
     stop();                                 /*停止前进*/
    else if (ir==0)                          /*如果前方没有物体*/
       stop();                               /*停止前进*/
      else if (ir==0b11)                     /*如果前方有物体*/
         drive(fol_trans_def,0);             /*跟随物体前进*/
        else if (ir==0b01)                   /*物体在右侧*/
           drive(fol_trans_def,(- fol_rot_def));  /*向右转*/
          else if (ir==0b10)                 /*物体在左侧*/
             drive(fol_trans_def,fol_rot_def);   /*向左转*/
  sleep(0.1);                                /*让运动持续一会儿*/
  old_bmp=bmp;
  }
}
```

```
void main()
{
follow();
}
```

4．运行这个程序。你在机器人前面走，它会在你后面紧跟，并且死死地“咬住”你。你也可以用一个物体在机器人前面引导，机器人就会跟着“走”。你领它到哪儿，它就跟到哪儿,绝对服从，没有“怨言”。

5．增加功能设想

（1）运行这个程序你会发现一个问题，当你在前面走时，若机器人跟不上，它就

会停下来。如果你不知道机器人没跟上，并且还在往前走，机器人就会离你越来越远。如果在机器人没跟上你时，让它“喊一声”，你就会知道，可以回去找它。你还可以让它在碰到物体时发出两个短音，找不到你时发出一个长音，这样你在前面走不用回头就能知道你的伙伴为什么没有跟上来了。

（2）让你的机器人碰到障碍物时，可以绕开“走”。

请你自己把新功能补上，看看谁的机器人最棒。

### 4.1.2 机器人歌唱家

在你和同学、朋友聚会的时候，大家会高兴地“唱”上一曲，以活跃气氛。如果你把你的机器人伙伴也带去，请它出来给大家“表演”：你拍它一下，它就为大家“演唱”一首歌曲。这一定会为你们的聚会增加乐趣。

1．编程思路：利用碰撞开关让机器人演唱。你先把歌曲按不同的曲目输入，编成曲目 1、曲目 2 等 8 支曲目，当你从不同的方向拍一下机器人时，机器人就演唱不同的歌曲。

2．提示：先把 8 支不同的曲目做成 8 个曲目函数，然后在主程序中调用。

3．程序的部分流程图，如图 4.1 所示。

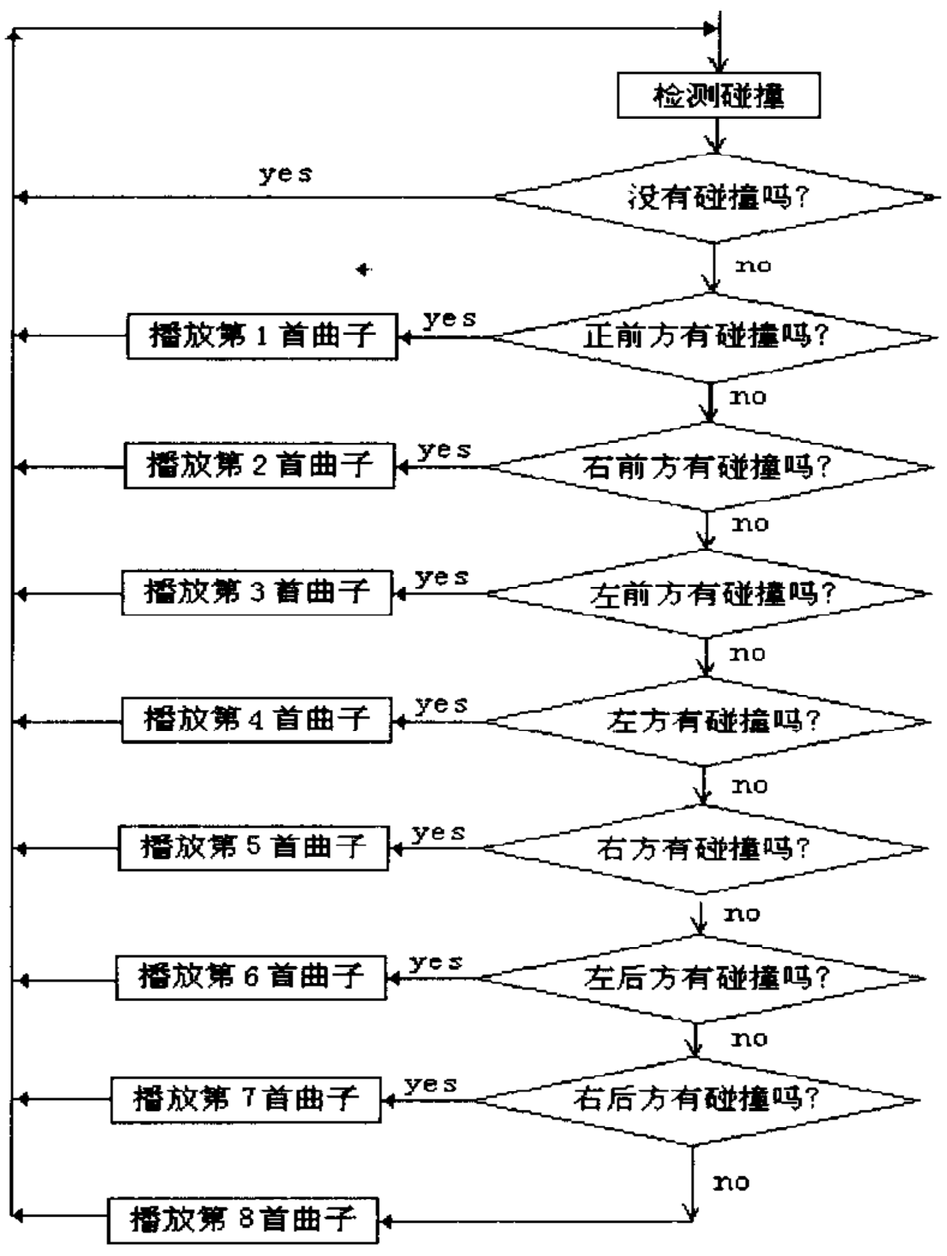



图 4.1 机器人歌唱家部分流程图

机器人“歌唱家”的基本程序结构如下：

```
void main()
{
int bump;
while(1)
   {
   bump=bumper();                          /*检测碰撞*/
   if (bump==0)
      printf(“wait for\n”);                /*没有碰撞*/
else if (bump==0b0011)                     /*正前方有碰撞吗？ */
      {
      printf(“descant-1\n”);
      descant-1();                         /*调用第 1 首曲子*/
      }
    else if (bump==0b0001)                 /*右前方有碰撞吗？ */
          {
          printf(“descant-2\n”);
          descant-2();                     /*调用第 2 首曲子*/
          }
        else if (bump==0b0010)             /*左前方有碰撞吗？ */
              {
              printf(“descant-3\n”);
              descant-3();                 /*调用第 3 首曲子*/
              }
            else if (bump==0b1010)         /*左方有碰撞吗？ */
                  {
                  .
                  .
                  .
                  }
   }
}
```

请你自己选择8首歌曲，编成8个曲目函数，然后完成机器人“歌唱家”的主程序。

### 4.1.3 机器人“足球”

你喜欢足球运动吗？一听说足球是不是特兴奋？我们这里不是让机器人去踢足球，而是把你的机器人当作一只足球来踢。一只机器人“足球”在我们脚下提溜提溜，转来转去，也是非常有趣的。

1. 编程思路： 如果把机器人当作“足球”踢，我们还是要利用机器人的碰撞传感器，让机器人在感知到来自前、后、左、右等方向的碰撞后，能相应地改变机器人的运动方向，就像一只足球，你踢它一下，它就往一个方向“跑”，所以我们称它为机器人“足球”。下面我们一起看一下机器人“足球”的程序。

【实例】机器人“足球”程序。

```
void billiards()
{
int bill_trans=0;
int bill_rot=0;
int bmpr=0;
while(1)                                    /*无限循环检测*/
   {
   bmpr=bumper();                           /*检测碰撞传感器*/
   if (bmpr!=0)
      {
      if (bmpr==0b0011)                     /*正前方发生碰撞*/
   {
   bill_trans=-80;                          /*后退*/
   bill_rot=0;
   }
else if (bmpr==0b1100)                      /*正后方发生碰撞*/
      {
      bill_trans=80;                        /*前进*/
      bill_rot=0;
      }
   else if (bmpr&0b1010)                    /*左侧发生碰撞*/
```

```
    {
    bill_trans=0;
    bill_rot=-80;
    drive(bill_trans,bill_rot);
    sleep(0.5);                    /*顺时针转一个角度*/
    bill_trans=80;                 /*前进*/
    bill_rot=0;
    }
  else if (bmpr&0b0101)            /*右侧发生碰撞*/
                {
                bill_trans=0;
                bill_rot=80;
                drive(bill_trans,bill_rot);
                sleep(0.5);                /*逆时针转一个角度*/
                bill_trans=80;             /*前进*/
                bill_rot=0;
                }
      drive(bill_trans,bill_rot);          /*驱动电机*/
      }
   }
}
void main()
{
billiards ();
}
```

2．机器人“足球”程序说明。

（1）“while(1) { 检测碰撞传感器，做出响应 }”，这个常用结构实现了对传感器的循环检测，使机器人能对周围环境的变化及时做出响应。

（2）程序开始运行时，机器人静止不动，只要你先踢它一下，它就开始运动起来，遇到碰撞就改变运动方向。

运行这个程序，你会看到机器人就像一只足球在地上“滚来滚去”。

3．多进程程序。

如果我们把机器人“足球”程序改为多进程程序，看看机器人的运动有什么不同。

【实例】

```
int bill_trans=0;
int bill_rot=0;
int bmpr=0;
int running=0;                          /*能力风暴初始值处于静止状态*/

void billiards()
{
while(1)                                /*无限循环检测*/
   {
   bmpr=bumper();                       /*检测碰撞传感器*/
   if (bmpr!=0)
      {
      if (bmpr==0b0011)                 /*正前方发生碰撞*/
         {
         bill_trans=-80;                /*后退*/
         bill_rot=0;
         }
      else if (bmpr==0b1100)            /*正后方发生碰撞*/
            {
            bill_trans=80;              /*前进*/
   bill_rot=0;
   }
else if (bmpr&0b1010)                   /*左侧发生碰撞*/
      {
      bill_trans=0;
      bill_rot=-80;
      leep(0.5);                        /*顺时针转一个角度*/
      bill_trans=80;                    /*前进*/
      bill_rot=0;
```

```
    }
else if (bmpr&0b0101)              /*右侧发生碰撞*/
    {
    bill_trans=0;
    bill_rot=80;
    sleep(0.5);                    /*逆时针转一个角度*/
    bill_trans=80;                 /*前进*/
    bill_rot=0;
    }
  }
}

void billiards_drive()
{
while(1)
  {
  running = bill_trans;                    /*能力风暴正在运动*/
  drive(bill_trans,bill_rot);              /*驱动电机*/
  }
}

void main()
{
start_process(billiards_drive());          /*创建电机驱动进程*/
start_process(billiards());                /*创建碰撞处理进程*/
}
```

4．多进程机器人“足球”程序的说明。

（1）这个程序在主程序当中创建了两个进程，一个是电机驱动进程 billiards_drive()，它是专门设置电机速度的；另一个是碰撞处理进程 billiards()，用于判断碰撞并改变电机速度的。

（2）主程序的两个进程，通过全局变量 bill_trans 和 bill_rot 进行数据传输。

（3）能力风暴的操作系统可以自动调用两个进程，给它们分配时间。

（4）从执行效果看，这两个程序没有什么区别，相当于两个进程并列运行。

（5）这种程序结构的优点是：便于增加新的进程，同时处理更多的外部信息。

比较这两个程序，看看两个程序有什么不同之处。

5．使机器人增加功能的设想。

让机器人遇到碰撞时发出“叫声”，以表示踢了它一下。

（1）我们先在机器人“足球”程序后面加入以下程序内容：

```
float octave=440.0;
float c_note=octave*(2.0^3.0/12.0);
float f_note=octave*(2.0^8.0/12.0);
float a_note=octave*(2.0^12.0/12.0);
float c1_note=octave*(2.0^15.0/12.0);
float f1_note=octave*(2.0^20.0/12.0);
float a1_note=octave*(2.0^24.0/12.0);
float c2_note=octave*(2.0^27.0/12.0);
float f2_note=octave*(2.0^31.0/12.0);
float a2_note=octave*(2.0^35.0/12.0);
float c3_note=octave*(2.0^39.0/12.0);

void select_bumper()
{
int bpr=bumper();                              /* 取碰撞值*/
if (bpr==0b1000)
  set_beeper_pitch(c_note);
  else if (bpr==0b1010)
        set_beeper_pitch(f_note);
      else if (bpr==0b0010)
            set_beeper_pitch(a_note);
          else if (bpr==0b0110)
                set_beeper_pitch(c1_note);
              else if (bpr==0b0100)
                    set_beeper_pitch(f1_note);
```

```
else if (bpr==0b1100)
        set_beeper_pitch(a1_note);
    else if (bpr==0b1000)
          set_beeper_pitch(c2_note);
      else if (bpr==0b1001)
            set_beeper_pitch(f2_note);
        else if (bpr==0b1010)
              set_beeper_pitch(a2_note);
          else if (bpr==0b1100)
                set_beeper_pitch(c3_note);
if (bpr!=0)                          /* Turn beeper on if any switch is pressed */
   beeper_on();
else                                 /* Turn beeper off otherwise */
   beeper_off();
}
```

（2）在程序中调用 select_bumper()函数

【实例】

```
int bill_trans=0;
int bill_rot=0;
int bmpr=0;
int running=0;                       /*能力风暴初始值处于静止状态*/

void billiards()
{
while(1)                             /*无限循环检测*/
   {
   bmpr=bumper();                    /*检测碰撞传感器*/
   select_bumper();
   if (bmpr!=0)
      {
      if (bmpr==0b0011)              /*正前方发生碰撞*/
         {
```

```
    bill_trans=-80;                    /*后退*/
    bill_rot=0;
    }
else if (bmpr==0b1100)                 /*正后方发生碰撞*/
        {
        bill_trans=80;                 /*前进*/
        bill_rot=0;
        }
     else if (bmpr&0b1010)             /*左侧发生碰撞*/
    {
    bill_trans=0;
    bill_rot=-80;
    sleep(0.5);                        /*顺时针转一个角度*/
    bill_trans=80;                     /*前进*/
    bill_rot=0;
    }
else if (bmpr&0b0101)                  /*右侧发生碰撞*/
       {
       bill_trans=0;
       bill_rot=80;
       sleep(0.5);                     /*逆时针转一个角度*/
       bill_trans=80;                  /*前进*/
                       bill_rot=0;
                       }
        }
    }
}

void billiards_drive()
{
while(1)
   {
```

```
    running = bill_trans;                    /*能力风暴正在运动*/
    drive(bill_trans,bill_rot);              /*驱动电机*/
    }
}

void main()
{
start_process(billiards_drive());            /*创建电机驱动进程*/
start_process(billiards());                  /*创建碰撞处理进程*/
}
```

现在运行这个程序，你轻轻地踢它一下，它就会“叫一声”。

6．再添加一个新进程。

【实例】我们再给机器人“足球”程序增加一个红外避障进程，看看机器人怎样运动。

我们在前一个程序中加入一个红外避障程序，全部程序如下：

```
int bill_trans=0;
int bill_rot=0;
int bmpr=0;
int forward=0;
int running=0;                               /*能力风暴初始值处于静止状态*/

void billiards()
{
while(1)                                     /*无限循环检测*/
   {
   bmpr=bumper();                            /*检测碰撞传感器*/
   select_bumper();
   if (bmpr!=0)
      {
      if (bmpr==0b0011)                      /*正前方发生碰撞*/
         {
         forward=0;
```

```
    bill_trans=-80;                    /*后退*/
    bill_rot=0;
    }
else if (bmpr==0b1100)                 /*正后方发生碰撞*/
        {
        forward=1;
        bill_trans=80;                 /*前进*/
        bill_rot=0;
        }
    else if (bmpr&0b1010)              /*左侧发生碰撞*/
        {
        bill_trans=0;
        bill_rot=-80;
        sleep(0.5);                    /*顺时针转一个角度*/
        forward=1;
        bill_trans=80;                 /*前进*/
        bill_rot=0;
        }
      else if (bmpr&0b0101)            /*右侧发生碰撞*/
          {
          bill_trans=0;
          bill_rot=80;
          sleep(0.5);                  /*逆时针转一个角度*/
          forward=1;
          bill_trans=80;               /*前进*/
          bill_rot=0;
          }
      }
   }
}

void billiards_ir()
```

```
{
int ir;
while(1)
  {
  if (running)                          /*没开始运动,不检测障碍*/
    {
    ir=ir_detect();                     /*检测红外传感器*/
    if (bmpr==0&&forward)               /*后退或发生碰撞时,不避障*/
      {
      if (ir==0b01)                     /*右侧有障碍,向左绕*/
        {
        bill_trans=20;
        bill_rot=80;                    /*逆时针转*/
        }
      else if (ir==0b10)                /*左侧有障碍,向右绕*/
        {
        bill_trans=20;
        bill_rot=-80;                   /*顺时针转*/
        }
      else if (ir==0b00)                /*前方无障碍,继续直行*/
        {
        bill_trans=80;
        bill_rot=0;
        }
      }
    sleep(0.1);
    }
  }
}

void billiards_drive()
{
```

```
while(1)
   {
   running=bill_trans;                    /*能力风暴正在运动*/
   drive(bill_trans,bill_rot);            /*驱动电机*/
   }
}

void main()
{
start_process(billiards_drive());         /*创建电机驱动进程*/
start_process(billiards_ir());            /*创建避障进程*/
start_process(billiards());               /*创建碰撞处理进程*/
}

float octave = 440.0;
float c_note=octave*(2.0^3.0/12.0);
float f_note=octave*(2.0^8.0/12.0);
float a_note=octave*(2.0^12.0/ 2.0);
float c1_note=octave*(2.0^15.0/12.0);
float f1_note=octave*(2.0^20.0/12.0);
float a1_note=octave*(2.0^24.0/12.0);
float c2_note=octave*(2.0^27.0/12.0);
float f2_note=octave*(2.0^31.0/12.0);
float a2_note=octave*(2.0^35.0/12.0);
float c3_note=octave*(2.0^39.0/12.0);

void select_bumper()
{
int bpr=bumper();                         /*取碰撞值*/
if (bpr==0b100)
   set_beeper_pitch(c_note);
else if (bpr==0b101)
```

```
    set_beeper_pitch(f_note);
else if (bpr==0b001)
        set_beeper_pitch(a_note);
    else if (bpr==0b011)
            set_beeper_pitch(c1_note);
        else if (bpr==0b010)
                set_beeper_pitch(f1_note);
            else if (bpr==0b110)
                    set_beeper_pitch(a1_note);
                else if (bpr==0b1000)
                        set_beeper_pitch(c2_note);
                    else if (bpr==0b1001)
                            set_beeper_pitch(f2_note);
                        else if (bpr==0b1010)
                                set_beeper_pitch(a2_note);
                            else if (bpr==0b1100)
                                    set_beeper_pitch(c3_note);
if (bpr!=0)                                 /* Turn beeper on if any switch is pressed */
   beeper_on();
else                                        /* Turn beeper off otherwise */
beeper_off();
}
```

程序说明：

（1）机器人增加了避开障碍物的功能。

（2）程序增加了红外避障进程，进程之间的参数传递和同步是多进程编程的难点。

（3）把 bmpr 设为全局变量，并通过全局变量 bmpr 来划分进程生效的时间。

（4）当发生碰撞时，只有碰撞处理进程可以修改电机速度。

（5）在其他时间，碰撞处理进程不断检测碰撞传感器，此时只有红外避障进程才能修改电机速度。

（6）碰撞处理进程的优先级高于红外避障进程。

（7）新增加的两个全局变量 forward 和 running 是红外避障行为逻辑的需要。

（8）running 是机器人开始运动的标志，红外避障进程要等这一事件发生之后才

起作用。

（9）forward 表示机器人当前运动的方向，避免红外避障进程在后退时处理前方障碍。

（10）程序当中的黑体部分为新增加的红外避障功能。

你会看到机器人与前面的运动有所不同。它可以在运动中躲闪要撞到的物体，当它离物体有一定的距离时，机器人一扭头就“跑”开了，你看它在地上来回“跑”着、“躲闪”着，就像一个顽皮的小孩儿在和你玩捉迷藏。

通过比较，我们看到使用多进程程序的好处在于：

- 这样的程序结构便于给机器人增加新的功能；
- 这样的程序结构可以同时处理多个外部信息，以提高机器人的智能程度。

通过多进程程序的应用，你再看你的机器人，它已经变得“聪明、伶俐、足智多谋”了。现在就带你的机器人去参加比赛吧。

## 4.2　让你的机器人去参加比赛

比赛是具有真正意义的竞争，通过比赛不仅可以考察你对机器人知识和技术的掌握、运用，也可以展示你的才华和智慧，更是发挥你创造能力的好机会。赶快行动吧，胜利永远属于敢于实践的人。

### 4.2.1　机器人绕标比赛

按照规定的路线运动,看谁的速度快。

1．如图 4.2 所示的是绕标比赛的场地图。

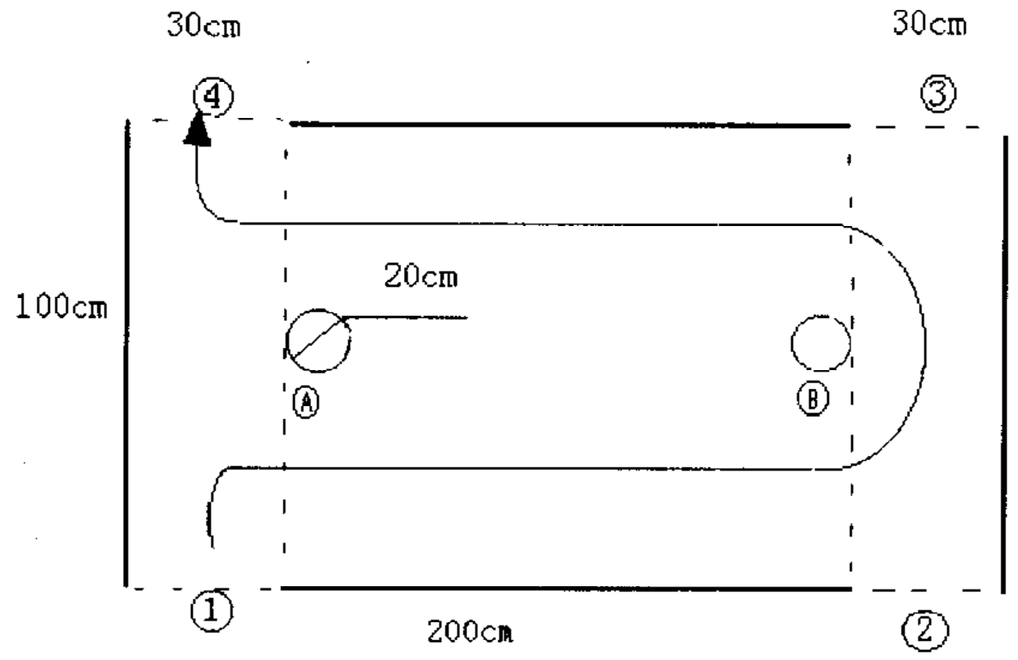



图 4.2　绕标比赛场地图

2．比赛：完成两条规定路线:从①门进，经 A 标后绕过 B 标，由④门出；从②门进，经 B 标后绕过 A 标，由③门出。

3．记分：两条路线成绩之和为总成绩。

4．参考程序如下：

【实例】

```
void main()
{
drive(100,0);                          /*从入口出发*/
sleep(0.5);
drive(20,-80);                         /*原地右转弯*/
sleep(0.3);
drive(100,0);                          /*直线前进*/
sleep(2.0);
drive(20,80);                          /*左转弯*/
sleep(0.4);
drive(100,0);                          /*前进*/
sleep(1.0);
drive(30,80);                          /*左转弯*/
sleep(0.4);
drive(100,0);                          /*前进*/
sleep(2.0);
drive(20,-80);                         /*右转弯*/
sleep(0.4);
drive(100,0);                          /*前进从 4 出口*/
sleep(1.0);
stop();
tone(3000.0,2.0);                      /*奏乐*/
tone(2000.0,1.0);
tone(1000.0,0.5);
tone(500.0,0.25);
tone(250.0,0.25);
}
```

以上是一种绕标的源程序，我们按照比赛路线为机器人编写了比赛程序。在程序中我们利用时间 sleep()控制机器人的“行走”，在机器人结束任务后会放一段音乐。

程序中每一个 sleep()的值，你必须根据场地的实际情况来调整参数。

你想一想还有没有其他方法来完成任务，比如利用光电编码器、红外传感器等。

### 4.2.2 救援行动

让机器人代替人类去执行一些危险任务，也是一件有实际意义的事。

1．比赛要求：机器人在指定地点待令，听到警笛响，机器人立即出发，在自己的辖区内进行搜索，找到被救物，将其救起后，送到有灯光标志的红十字急救中心放下，然后原地待令。

比赛可以有两种方式：一种是看谁在最短的时间内完成任务，另一种是在指定的时间内，看谁完成的最好（救的“人员”多）。

2．提示： 让机器人搜索、寻找目标，对于它来讲不应该有什么问题，难点在于怎样“救起”被救物，然后到指定地点还要放下被救物。是让机器人把被救物“提起”，还是“铲起”，或者“抱起”。请你们自己去创造发挥，看谁的构思更新颖，设计更巧妙。

请你参考前面的灭火程序，完成本程序。

### 4.2.3 机器人灭火比赛

机器人走进家庭为人类服务是我们的梦想，现在可以清晰地预见：家用机器人将是下一次 IT 产业的革命。美国三一学院的 Jake 门德尔森创立了机器人灭火比赛，目前已成为美国最大众化的、规模最大的机器人比赛，并吸引了世界众多国家的参与，已成为世界级的比赛。

1．比赛目的与意义

（1）开发有实际用途的家用机器人。

（2）面对实际的问题和需求，刺激机器人新技术的发展。

（3）使各界能了解家用机器人的实际应用和前景，鼓励支持机器人产业的发展。

（4）通过比赛促使青少年亲手制作机器人，使他们从机械、电子、计算机、人工智能和工程组织等方面获得极好的锻炼，激励青少年投身机器人事业。

（5）通过比赛提高青少年的动手能力、创造能力，培养探索精神和协作精神。

2．比赛规则

（1）场地：比赛场地如图 4.3 所示。单位：cm。

比赛场地的墙壁为白色高 33cm，地面为黑色；走廊宽 46cm；每个房间的门是一个 46cm 的开口，有一条 2.5cm 宽的白线表示房间入口。在走廊的中心，有一个标有

字母“H”的直径为 30cm 的白色圆圈代表家。

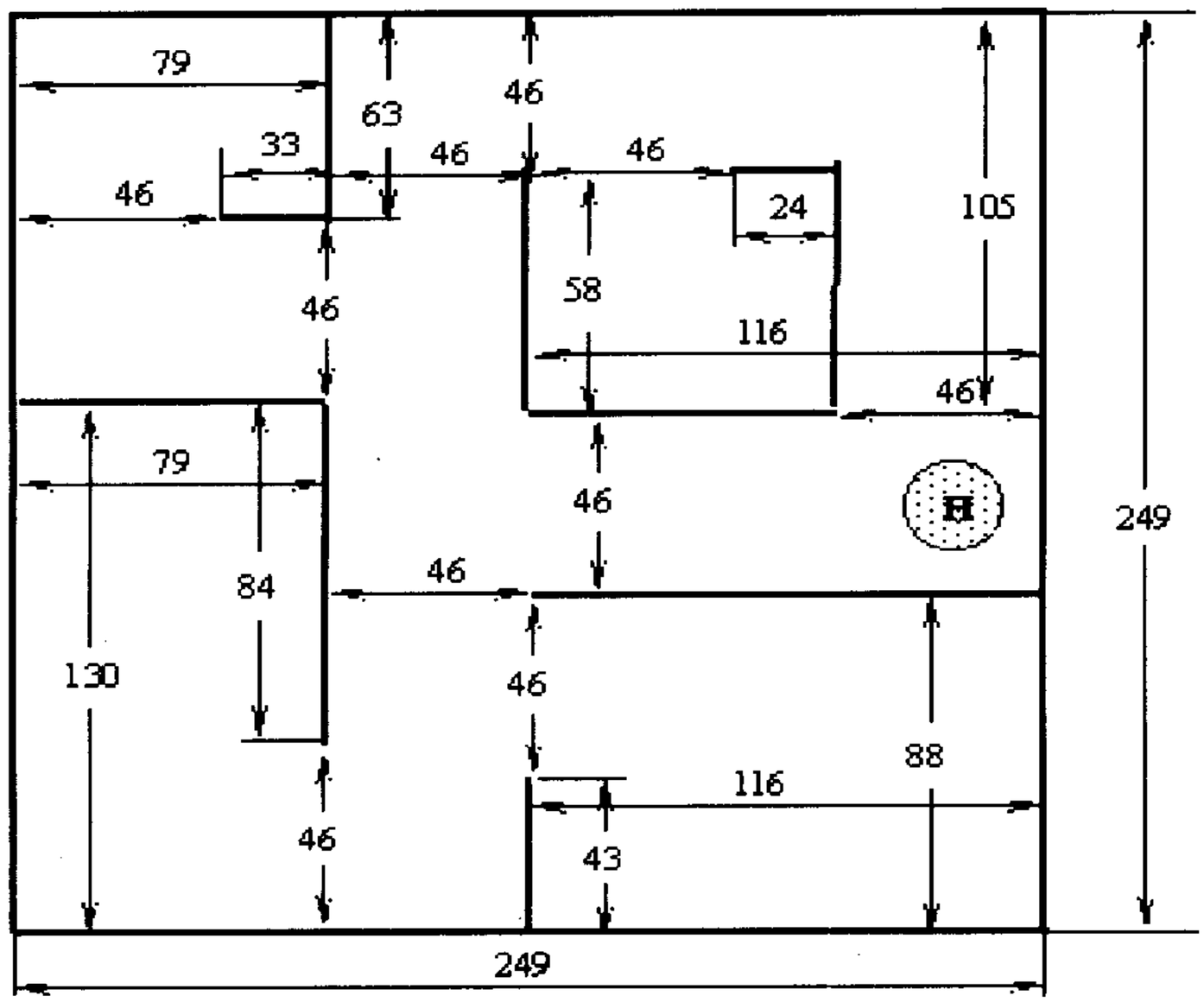



图 4.3　灭火比赛场地示意图

（2）火源：用蜡烛代表火源。约 2.5cm 直径的蜡烛安装在 7.5cm*7.5cm*3.5cm 的基座上，火焰底部离地面 15cm~20cm 高。蜡烛必须放在房间里，在距离火源 30cm 的圆上有一条 2.5cm 宽的白线。

（3）机器人：机器人的最大尺寸不能超过 31*31*31cm。机器人的重量和材料没有限制。

（4）其他：禁止在墙上或地上放置任何标记，帮助机器人导航。在不违反规则和规范的情况下对传感器型号没有限制。

（5）灭火：机器人不能运用任何带有破坏性或危险性的方法灭火。比如：机器人不能通过燃放爆竹产生冲击熄灭蜡烛，不能通过碰倒蜡烛灭火。可以运用类似水、空气、CO2、Halon 等方法灭火。

（6）比赛：机器人必须在标有“H”的白色圆圈中启动，然后穿越走廊，逐个房间巡视。发现火源之后，机器人要在 30cm 的圆内（部分即可）将火扑灭。整个灭火过程要求机器人自主完成，不许人为干预。整个过程不能超过 6 分钟，所用时间越少

成绩越好，在寻找火源的过程中还有一些违规的加分规则，最后所得分数越少成绩越好。

3．编程思路

根据比赛的有关规定，列出一些必要的因素供参考。

（1）机器人可以“听令”启动，也可以通过“复位”键启动。

（2）因为碰壁一次要被加分，所以机器人要有避障功能。

（3）可以增加识别火焰的传感器。安装时要考虑方向和火焰的高度，通过调用analog(5)和 analog(6)即可检测是否有火焰。

（4）灭火装置要用电机启动。

（5）寻找到火源将火焰扑灭后，要返回起点。

4．灭火程序

【实例】灭火程序。

```
int trans=100,rot=0;
int ir,bump;
int b_mark=0,fire_mark=0;

void set_speed(int a,int b)
{
trans=a;
rot=b;
}

void bumptest()
{
while(1)
   {
   if (bump)
      b_mark=1;
   if (bump==0b0011)
      set_speed(0,-70);
   else if (bump==0b1100)
            set_speed(70,0);
         else if (bump&0b1010)
```

```
    {
    set_speed(-60,0);
    sleep(0.2);
    set_speed(60,-40);
    }
else if (bump&0b0101)
        {
        set_speed(-60,0);
        sleep(0.2);
        set_speed(60,40);
        }
   sleep(0.05);
   b_mark=0;
   }
}

void irtest()
{
while(1)
   {
   ir=ir_detect();
   if (b_mark==0&&fire_mark==0)
{
if (ir==0b11)
   {
   set_speed(0,-40);
   sleep(0.5);
   set_speed(100,0);
   }
else if (ir==0b10)
        {
        set_speed(0,-40);
```

```
    sleep(0.5);
    set_speed(100,0);
    }
else if (ir==0b01)
        {
        set_speed(0,40);
        sleep(0.5);
        set_speed(100,0);
        }
                else
                   {
                   set_speed(100,0);
                   sleep(0.1);
                   }
        }
    }
}

void light()
{
int left;
int right;
int diff;
while(1)
   {
  bump=bumper();
   left=(analog(1)+analog(1))/2;
   right=(analog(0)+analog(0))/2;
   printf("L=%d     R=%d\n",left,right);
   diff=right-left;
   if (left<30&&right<30)
      {
```

```
fire_mark=1;
set_speed(0,0);
dcmotor3(1);                    /*启动电机*/
sleep(5.0);
tone(1000.0,0.5);
tone(2000.0,0.5);
dcmotor3(0);                    /*关闭电机*/
fire_mark=0;
set_speed(60,0);
   }
else if (left<30||diff>100)
        {
        fire_mark=1;
        set_speed(0,20);
        sleep(0.3);
        set_speed(0,0);
        dcmotor3(1);
        sleep(5.0);
        tone(1000.0,0.5);
        tone(2000.0,0.5);
        dcmotor3(0);
        fire_mark=0;
    set_speed(60,0);
    }
else if (right<30||diff<-100)
        {
        fire_mark=1;
        set_speed(0,-20);
        sleep(0.3);
        set_speed(0,0);
        dcmotor3(1);
        sleep(5.0);
```

```
            tone(1000.0,0.5);
            tone(2000.0,0.5);
            dcmotor3(0);
            fire_mark=0;
            set_speed(60,0);
            }
        else sleep(0.05);
    }
)

void main ()
{
start_process(light());
start_process(irtest());
start_process(bumotest());
while(1)
   {
   drive(trans,rot);
   sleep(0.1);
   }
}
```

程序说明：

● 此程序是通过启动电机带动风扇完成灭火任务的。

● 通过扩展口增加电机，用 dcmotor3(1)函数启动电机，用 dcmotor3(0)函数关闭电机。

● 扩展电机的连接如图 4.4 所示。将电机接在 DC 电机插针的位置，然后把跳线接在直流和伺服选择上。

你应该根据比赛的规则和要求来编写程序，上面介绍的程序仅供参考。

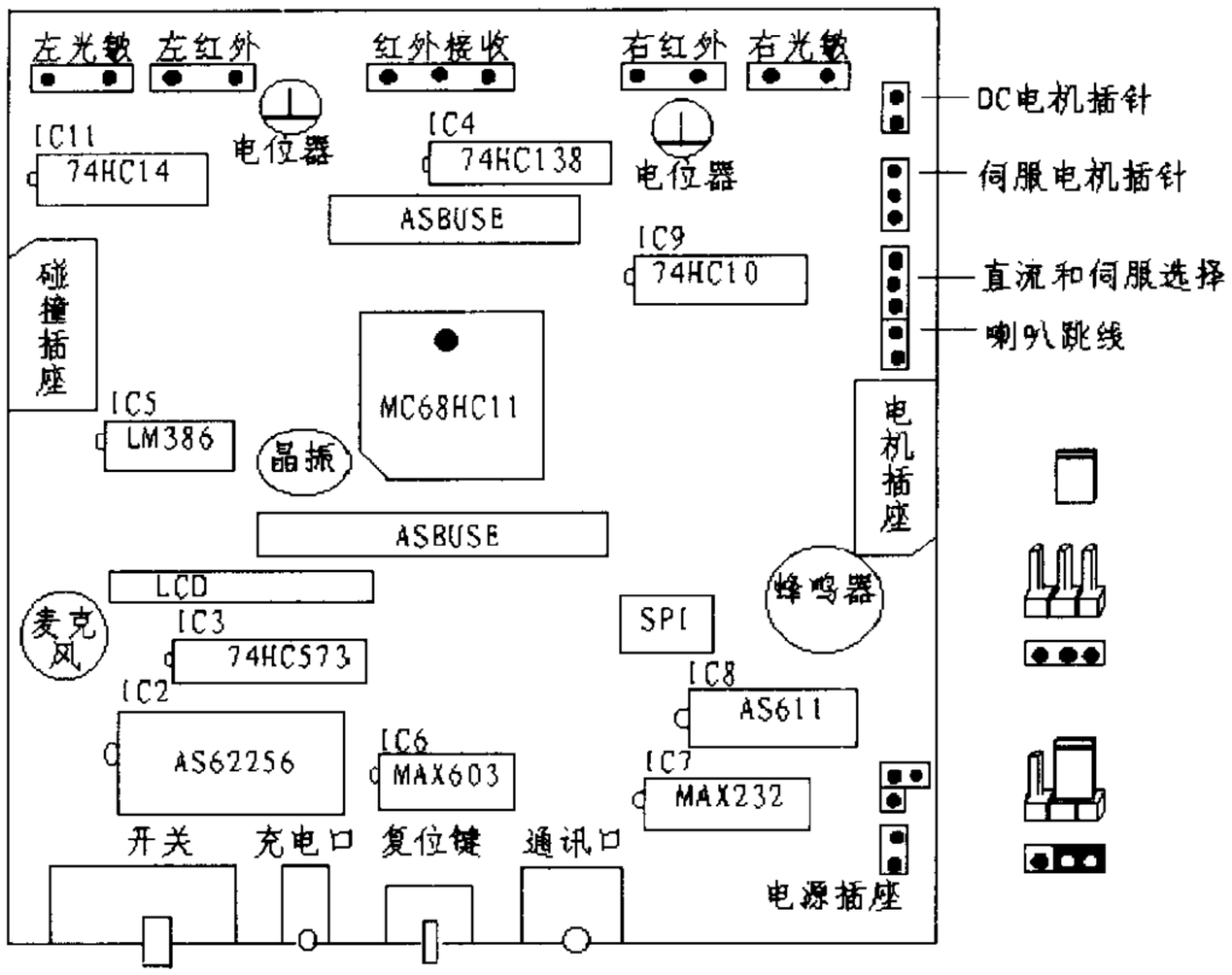



图 4.4　控制板布局图

## 4.3　迷人的机器人项目

下面列举一些机器人的比赛和游戏项目，供你在学习、试验中巩固和提高。

### 4.3.1　机器人体育比赛

把机器人与竞技体育相结合,设计机器人的体育比赛。

1．机器人短跑比赛

场地：可以根据环境大小设置距离，1 米、3 米、5 米均可，只要是直线就行。

比赛：机器人在起跑线待令，听到发令，立即全速前进直奔终点。

提示：机器人对发令反映要快，“跑”的路线要尽量直。其他传感器可不使用。

2．机器人长跑比赛

场地：跑道设置成椭圆形，10 米一圈（也可根据环境自行确定）。

比赛：机器人听到发令后“起跑”，看谁先“跑”完 50 米。如果场地没有隔板，“跑”出场地就退出赛场；如果场地有隔板，碰撞一次加 1 分。

增加难度：“跑”到终点自动停下来。

提示：因为有弯道，所以要用红外避障。

3．机器人“跨栏”赛

场地：大小不限，只是在场地中要设若干个障碍，让机器人“跨过”障碍。

比赛：比赛中机器人必须绕过障碍，碰到障碍就加 1 分。

4．机器人接力赛

场地：最好设立一个直线场地，距离长短自定。

比赛：要求每组 4 个机器人，第一棒，听发令，后 3 棒靠碰撞接力。

提示：当后面有碰撞时，后 3 棒就开始“起跑”；每个机器人在前面有碰撞时，就停下来，表示已完成接力。

当然，你还可以自己设计出机器人马拉松赛、机器人越野赛等项目。

### 4.3.2 机器人实验

1．机器人找家。如图 4.5 所示的是机器人找家的示意图。


图 4.5 机器人找家示意图

我们在小圆板上放一只蜡烛或灯泡，把机器人放在 1.5 米外，光敏传感器背对着蜡烛，看谁的机器人先找到以蜡烛为标志的家，然后停下来。机器人能找到家这一功能有非常重要的作用，这意味着未来的机器人一定要有能够自己回家去充电的“本领”。

2．载人飞船，如图 4.6 所示的是载人飞船示意图。

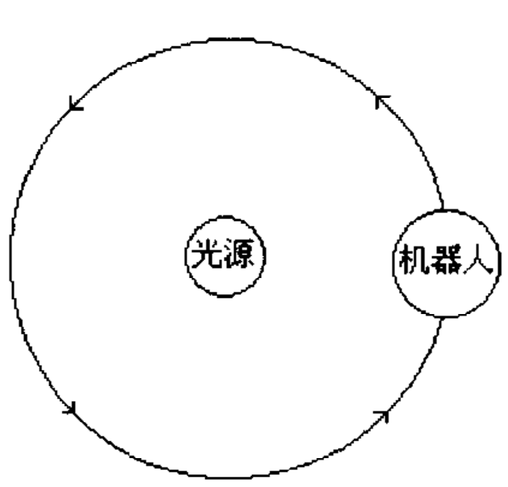



图 4.6 载人飞船示意图

点燃一只蜡烛或灯泡，使机器人绕着光源转，就像载人飞船绕着地球转一样。

3．机器人弹性球

用机器人验证中学课本中的惯性定理，如图 4.7 所示。

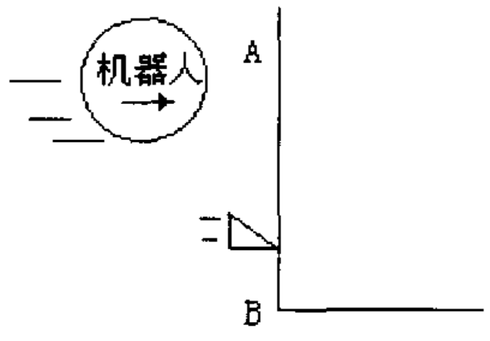



图 4.7　弹性球实验示意图

机器人以全速直线前进，在 LCD 上显示前进速度，碰撞开关碰上 A 点后机器人马上停止运行，使机器人按惯性前行，读一下弹簧表上的读数，算一算，是不是和惯性定律相吻合。

4．机器人秒表

编一个机器人秒表程序，使机器人听到发令后开始计时，当遇到碰撞时停止计时，显示计时结果。

### 4.3.3　其他项目

1．自由战士

让你的机器人在房间里漫游，什么障碍也挡不住，有时停下来“唱”首歌。注意在“唱”歌时，如果有人或物体碰到它，它会避开。

2．舞蹈家

首先让机器人能做到自己表演舞蹈；然后增加难度，它他边“唱”边“舞”；再后它他一边“跳舞”还要避障、避碰。

3．书法家

让机器人“写”一个“之”字，再让他连续写出十个数字“0、1、2、3、4、5、6、7、8、9”，最后试试它能不能写出 26 个英文字母。

经过这 4 章的学习和实践，相信你对机器人已经产生了兴趣，对自己有了信心，希望你和你的机器人伙伴共同完成精彩的项目，在未来的比赛中能有精彩的表演。

现在你可以设计一些有意义的项目，让你的机器人伙伴和你一起来实现吧。

# 附录 1　JC 语言部分语句、函数一览

## 1. 传感器函数

● 电机初始化

格式：int init_motors()

功能：初始化电机，必须在使用电机强调用。

● 关闭电机

格式：stop()

功能：关闭两个电机。

● 设定电机功率级别

格式：motor(int m,int p)

功能：以功率级别 p 启动电机 m，其中 m 为 0 时表示左电机，m 为 1 时表示右电机；功率级别 p 为-100~100 的整数，p 值为正数时电机正向转动，p 值为负数时电机反向转动。

● 同时设定两个电机速度

格式：drive(int trans,int rot)

功能：设定两个电机的转动速度。左电机速度为 trans-rot，右电机速度为 trans+rot。

● 单向电机驱动

格式：dcmotor3(int flag)

功能：flag=1 时启动电机，flag=0 时关闭电机。

● 伺服电机的驱动

格式：int servo_a3_init(int enable)

功能：enable=0 时关闭伺服电机电源，enable=1 时表示允许伺服电机工作。

● 数字口传感器的值

格式：int digital(int p)

功能：返回数字口 p 上传感器的值（1 或 0）。

● 光敏传感器

格式：analog(int p)

功能：检测光敏传感器测试值。当 p 值为 0 表示右光敏传感器，p 值为 1 时表示左光敏传感器。函数的检测值为 0 到 255 之间的整数，数值越大说明光线越暗，数值越小说明光线越亮。

● 声音传感器

格式：analog(2)

功能：检测声音传感器（麦克风）的测试值。函数的检测值为 0 到 255 之间的整数，数值越大说明声音越大，数值越小说明声音越小。

● 红外传感器

格式：ir_detect()

功能：检测红外线传感器接收到的检测值。

红外传感器收到的检测值与外界障碍位置的关系见附表 1。

**附表 1　　红外传感器的检测值**

| 红外信号接收状态 | 二进制数 | 十进制数 | 前方障碍状况 |
|---|---|---|---|
| 没有接收到红外信号 | 0b00 | 0 | 无障碍 |
| 右边接收到红外信号 | 0b01 | 1 | 右前方障碍 |
| 左边接收到红外信号 | 0b10 | 2 | 左前方障碍 |
| 两边接收到红外信号 | 0b11 | 3 | 正前方障碍 |

● 碰撞传感器

格式：bumper()

功能：检测 4 个碰撞传感器，来自 8 个方位的碰撞测试值。

碰撞传感器接收到的检测值与外界碰撞的方向的关系见附表 2。

**附表 2　　碰撞传感器的检测值**

| 碰撞传感器方向 | 十进制检测值 | 二进制检测值 |
|---|---|---|
| 右前方 | 1 | 0b0001 |
| 左前方 | 2 | 0b0010 |
| 右后方 | 4 | 0b0100 |
| 左后方 | 8 | 0b1000 |
| 前方 | 3 | 0b0011 |

续表

| 碰撞传感器方向 | 十进制检测值 | 二进制检测值 |
| --- | --- | --- |
| 左方 | 10 | 0b1010 |
| 右方 | 5 | 0b0101 |
| 后方 | 12 | 0b1100 |

● 初始化编码器

格式：init_velocity()

功能：初始化编码器。在编码器开始计数时必须先初始化。

● 左轮编码器的计数值

格式：get_left_clicks()

功能：取左轮编码器的计数值，并将编码器的计数重置为零。

● 右轮编码器的计数值

格式：get_right_clicks()

功能：取右轮编码器的计数值，并将编码器的计数重置为零。

● 左编码器的状态

格式：left_shaft()

功能：返回左编码器的当前状态。0 为脉冲的低电平（码盘上的白格），1 为脉冲的高电平（码盘上的黑格）。

● 右编码器的状态

格式：right_shaft()

功能：返回右编码器的当前状态。0 为脉冲的低电平（码盘上的白格），1 为脉冲的高电平（码盘上的黑格）。

● 产生音频信号

格式：beep()

功能：产生一段持续时间为 0.3s，频率为 500Hz 的音频信号。

● 产生规定的音频信号

格式：tone(float x,float y)

功能：产生一个持续时间为 xs，频率（音调）为 yHz 的音频信号，m 和 h 为实型数。

## 2. 控制语句

● 条件语句

格式：if <条件>

{<程序段 1>}

else

{<程序段 2>}

<程序段 3>

功能：当条件成立时执行<程序段 1>，然后执行<程序段 3>，否则执行<程序段 2>后执行<程序段 3>。

● 控制循环的 while 语句

格式：while <条件>

{<循环体>}

功能：当条件成立实执行<循环体>，否则执行<循环体>后面紧接的程序内容。

● 控制循环的 for 语句

格式：for (<赋值表达式>,<条件>,<增量表达式>)

{<循环体>}

功能：当 for 语句执行时，<赋值表达式>为循环变量提供初值，然后判断关于循环变量的条件是否成立，若条件成立执行<循环体>，并通过增量表达式改变循环变量的值；再次重复条件判断的过程，当条件不成立时退出循环控制，执行<循环体>后面紧接的程序内容。

● 退出循环控制的语句

格式：break

功能：退出循环。

## 3. 其它语句或函数

● 输出语句

格式：printf("<输出格式>",<变量 1 或表达式 1>,<变量 2 或表达式 2>, ……)

功能：按照设定的输出格式输出变量或表达式的值。

● 赋值函数

格式：<变量>=<表达式>

功能：计算赋值号“=”右边表达式的值，并将计算的结果赋值给赋值号左边的变量。

● 延时函数（单位：s）

格式：sleep(float t)

功能：等待 t 秒执行后面紧接的语句，t 为实型数。

● 延时函数（单位：ms）

格式：msleep(fioat t)

功能：等待 t 毫秒后执行后面紧接的语句，t 为实型数。

● 时间复位

格式：reset_system_time()

功能：将系统时间复位清零。

● 系统运行时间（单位：ms）

格式：get_time()

功能：以毫秒形式返回系统运行时间的计数。函数值为长整型数。

● 系统运行时间（单位：s）

格式：seconds()

功能：以秒形式返回系统运行时间的计数，函数值为实型数。

● 创建进程

格式：start_process(<进程标识名称>)

功能：创建一个所标识名称的进程，运行时间默认为 5 毫秒。

● 撤销进程

格式：int kill_process(int pid)

功能：撤消一个所标识名称的进程。

## 4. 表达式

● 算术表达式

表达式指的是用运算符号将函数、变量、常量连接起来的式子。

一般情况下，算术表达式中运算符号的运算顺序见附表 3。

**附表 3　　算术表达式中的运算符号**

| 运算顺序 | 运算 | 运算符号 |
|---|---|---|
| 1 | 括号 | ( ) |
| 2 | 函数 | 函数名 |
| 3 | 乘方 | ∧ |
| 4 | 乘法、除法 | *、/ |
| 5 | 加法、减法 | +、− |

● 关系表达式

用关系运算符连接算术表达式或变量构成的式子是关系表达式。关系运算符见附表 4。

**附表 4　　关系表达式中的运算符号**

| 运算 | 大于 | 小于 | 大于等于 | 小于等于 | 等于 | 不等于 |
|---|---|---|---|---|---|---|
| 运算符 | > | < | >= | <= | == | != |

● 逻辑表达式

用逻辑运算符连接关系表达式或变量构成的式子是逻辑表达式。逻辑运算符见附表 5。

**附表 5　　逻辑表达式中的运算符号**

| 运算 | 非 | 与 | 或 |
|---|---|---|---|
| 运算符 | ! | && | \|\| |
| 运算的优先级别 | 高 | 中 | 低 |

● 逻辑运算的真值表见附表 6。

**附表 6　　逻辑运算的真值表**

| 关系式 a | 关系式 b | a! （非） | a&&b（与） | a\|\|b（或） |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

# 附录 2　机器人基本知识简介

机器人在小说和影视中一露面，就给人留下了深刻的印象和无限的遐想。曾几何时，机器人走出了书本、影视，来到了现实生活中，走进了工厂、矿山、农场等。在新世纪，随着技术的发展，机器人将会像个人电脑一样，走进家庭，走进人们的生活，成为人类社会必不可少的生活用品或工具，这是世界机器人发展的趋势。

## 1. 机器人的发展历史

机器人技术是 20 世纪人类最伟大的发明，机器人以它特有的魅力，从一面世就受到全社会的关注，并吸引着广大青少年的目光。

（1）机器人的由来

“机器人”3 个字体现了长期以来人类的一种愿望，即创造出一种机器，能够代替人去进行各种工作。

机器人的概念在人类的想象中已经存 3000 多年了。早在我国西周时代，就流传有关巧匠偃师献给周穆王一个歌舞机器人的故事。公元前 3 世纪，古希腊发明家戴达罗斯用青铜为克里特岛国王迈诺斯塑造了一个守卫宝岛的卫士塔罗斯。在公元前 2 世纪出现的书籍中，描写过一个机械化剧院，里面的这些类似机器人的角色能够在宫廷仪式上进行舞蹈和列队表演。我国东汉时期，张衡发明的指南车是世界最早的机器人雏形。

人类历史进入近代之后，出现了第一次工业和科学革命。随着各种自动机器和动力系统的问世，机器人开始由幻想时期转入自动机械时期，许多机械式控制的机器人，主要是各种精巧的机器人玩具和工业品，应运而生。

公元 1768~1774 年间，瑞士钟表匠德罗斯父子 3 人，设计制造出 3 个像真人一样大小的机器人——写字偶人、绘图偶人和弹风琴偶人。它们是由凸轮控制和弹簧驱动的自动机器，至今还作为国宝保存在瑞士纳切特尔市艺术和历史博物馆内。同时，还有德国梅林制造的巨型泥塑偶人“巨龙戈雷姆”、日本物理学家细川半藏设计的各种自动机械图形，法国杰夸特设计的机械式可编程织造机等，都在机器人发展史上留下了

光辉的一页。1893 年，加拿大人摩尔设计了能行走的以蒸汽为动力的机器人“安德罗丁”。

这些机器人工艺珍品，标志着人类在机器人从梦想到现实这一漫长道路上，前进了一大步。

进入 20 世纪之后，机器人已躁动于人类社会和经济的母胎之中，人们含有几分不安地期待着它的诞生。他们不知道即将问世的机器人是个宠儿，也是个怪物。1920 年捷克作家卡雷尔·卡佩克发表了幻想情节剧《罗萨姆的万能机器人》，第一次提出了“机器人”这个名词。剧中的罗萨姆公司把机器人作为人类生产的工业产品推向市场，让它去充当劳动力，以呆板的方式从事繁重的体力劳动。后来，罗萨姆公司使机器人具有了感情，在工厂和家务劳动中，机器人成了必不可少的成员。该剧预示了机器人的发展对人类的影响。在剧本中，卡佩克把捷克语“Robota”（农奴）写成了“Robot”（机器人）。这也是人类社会首次使用“机器人”这一概念。

1954 年，美国人德沃尔设计了第一台可编程的工业机器人，并于 1961 年申请了专利。1962 年，美国万能自动化（Unimation）公司的第一台机器人 Unimate 在美国通用汽车公司（GM）投入使用，这标志着第一代机器人的诞生。从此，机器人成为现实，人类继续以自己的智慧和劳动，不断谱写机器人历史的新篇章。

（2）机器人的定义

在技术领域中，几乎每一术语都有明确的定义，但关于机器人的定义却和盲人摸象一样，仁者见仁，智者见智，至今尚无统一的意见。究竟人意味着什么？自古以来就是一个哲学问题。因此，在对机器人进行定义之前，应首先从词源方面加以考察。机器人是 20 世纪出现的新名词。在卡佩克的剧本《罗萨姆的万能机器人》中，罗萨姆在捷克语中是理性（reason）的意思，它由 rozum 转用而来，在古代斯拉夫语中，robota 意味着强制劳动，卡佩克根据这一词造出具有“奴隶机器”含义的新词 robot（机器人）。

日本的人工手研究会（现在改为仿生机构学会）和视听觉信息研究会，曾于 1967 年联合召开了日本第一届机器人学术会议。当时有两个代表性的机器人定义。一是森政弘与合田周平提出的：“机器人是一种具有移动性、个体性、智能性、通用性、半机械半人性、自动性、奴隶性等 7 个特性的柔性机器。”从这一定义出发，森政弘又提出了用自动性、智能性、个体性、半机械半人性、作业性、通用性、信息性、柔性、有限性、移动性等 10 个特性来表示机器人的形象。另一个定义是加藤一郎提出的，即把具有以下 3 个条件的机器称为机器人：

- 具有脑、手、脚等 3 个要素的个体；
- 具有非接触传感器（用眼、耳接收远方信息）和接触传感器；
- 具有平衡觉和固有觉的传感器。

这个定义强调了机器人应当仿人的含义，即它靠手进行作业，靠脚实现移动，由

脑统一指挥。非接触传感器和接触传感器相当于人的五官，使机器人能识别外界环境，而平衡觉和固有觉则是机器人感知自身状态不可缺少的传感器。这里描述的不是工业机器人而是自主机器人。不言而喻，正如人由于某种原因而缺少身体的某一部分或丧失某一器官功能，仍然还是人一样，机器人也并不一定要具有上面所述的所有构成要素。机器人的定义是多种多样的，其原因是它具有一定的模糊性，动物一般也具有上述这些要素，所以在把机器人理解为仿人机器的同时，也可以广义地把机器人理解为仿动物的机器。

因此，要给机器人下个合适的和为人们普遍接受的定义就比较困难。下面列举几个有代表性的机器人定义：

美国机器人协会的定义。机器人是“一种用于移动各种材料、零件、工具或专用装置的，通过可编程动作来执行种种任务的，并具有编程能力的多功能机械手”。这里指的主要是工业机器人。

日本工业机器人协会的定义。工业机器人是“一种装备有记忆装置和末端执行器的、能够转动并通过自动完成各种移动来代替人类劳动的通用机器”。

美国国家标准局的定义。机器人是“一种能够进行编程并在自动控制下执行某些操作和移动作业任务的机械装置”。这也是一种广义的工业机器人定义。

国际标准化组织的定义。“机器人是一种自动的、位置可控的、具有编程能力的多功能机械手，这种机械手具有若干个轴，能够借助于可编程操作来处理各种材料、零件、工具和专用装置，以执行种种任务”。

我国的“机器人之父”蒋新松院士曾建议把机器人定义为“一种拟人功能的机械电子装置”。

上述的各种定义有共同之处，即认为机器人像人，并能模仿人的动作；具有智力或感觉与识别能力；是人造的机器或机械电子装置。随着机器人技术的发展，这些定义都有可能得到修改，机器人的范畴将不断扩大，甚至需要对机器人重新定义。

（3）机器人学的发展

自动化技术的发展，特别是计算机的诞生，推动了现代机器人的发展。

20 世纪 60 年代，随着传感技术和工业自动化的发展，工业机器人进入成长期，机器人开始向实用化发展，并被用于焊接和喷涂作业中。

70 年代随着计算机和人工智能的发展，机器人进入实用化时代。日本虽起步较晚，但有关部门结合国情，面向中小企业，采取了一系列鼓励使用机器人的措施，使其机器人拥有量很快超过了美国，一举成为“机器人王国”。

80 年代，机器人发展成为具有多种移动机构、通过传感器控制的机器。工业机器人进入普及时代，开始在汽车、电子等行业得到大量使用，从而推动了机器人产业的发展。为满足人们个性化的要求，工业机器人的生产趋于小批量、多品种。

90 年代初期，工业机器人的生产和需求进入高潮期。1990 年世界上新装备机器人 81，000 台，1991 年新装备 76，000 台，到 1991 年底世界上已有 53 万台工业机器人工作在各条战线上。随后由于受到日本等国经济危机的影响，机器人产业也一度跌入低谷。近几年随着世界经济的复苏，机器人产业又出现了一片生机。90 年代还出现了具有感知、决策、动作能力的智能机器人，产生了智能机器和机器人化机器。随着信息技术的发展，机器人的概念和应用领域也在不断扩大。

现阶段，除了制造环境下的工业机器人之外，非制造环境下的特种机器人也获得了长足发展。如人型机器人、军用机器人、水下机器人、空间机器人、服务机器人、微型机器人和足球机器人以及各种各样的娱乐机器人等。

## 2. 机器人的分类

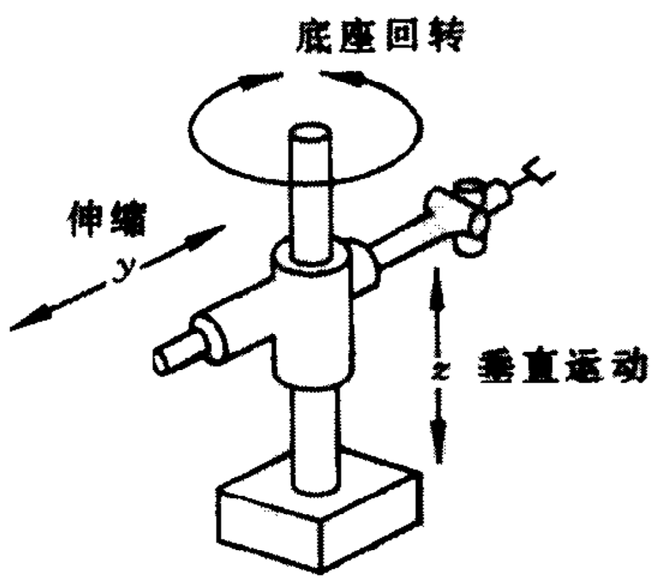
机器人的分类方式很多，这里主要介绍几种常用的分类方法。

（1）按机械手的结构来分

机器人机械手的机械配置形式多种多样，最常见的结构形式是用其坐标特性来描述的。这里简单介绍柱面坐标结构、球面坐标结构和关节式球面坐标结构 3 种最常见的机器人。

● 柱面坐标机器人

柱面坐标机器人主要由垂直柱子、水平手臂（或机械手）和底座构成。水平机械手装在垂直柱子上，能自由伸缩，并可沿垂直柱子上下移动。垂直柱子安装在底座上，并能与水平机械手一起（作为一个部件）在底座上移动。这样，这种机器人的工作空间就形成一段圆柱面，因此这种机器人叫做柱面坐标机器人。如附图 1 所示。



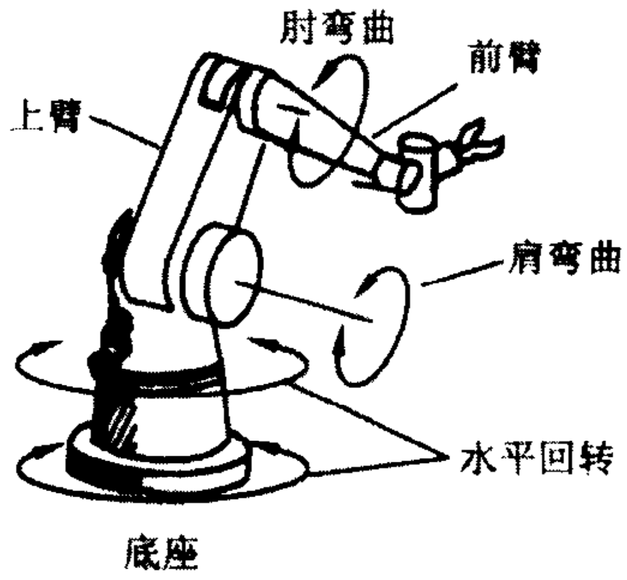

附图 1　柱面坐标机器人示意图

● 球面坐标机器人

这种机器人像坦克的炮塔一样，其机械手能够里外伸缩移动，还可以在垂直平面上摆动，以及绕底座在水平面上转动，它的运动空间是球面的一部分，因此被称为球面坐标机器人。

● 关节式球面坐标机器人

这种机器人主要由底座（或躯干）、上臂和前臂构成。上臂和前臂可以在通过底座的垂直平面上运动，在前臂和上臂之间，机械手有个肘关节，在上臂和底座间，机械手还有个肩关节。在水平面上的旋转运动，既可由肩关节进行，也可以通过绕底座旋转来实现。这种机器人的工作空间形成球面的大部分，因此称为关节式球面机器人。如附图 2 所示。




附图 2　球面坐标机器人示意图

（2）按机器人的控制方式来分

按照控制方式可把机器人分为非伺服机器人和伺服机器人两种。

● 非伺服机器人

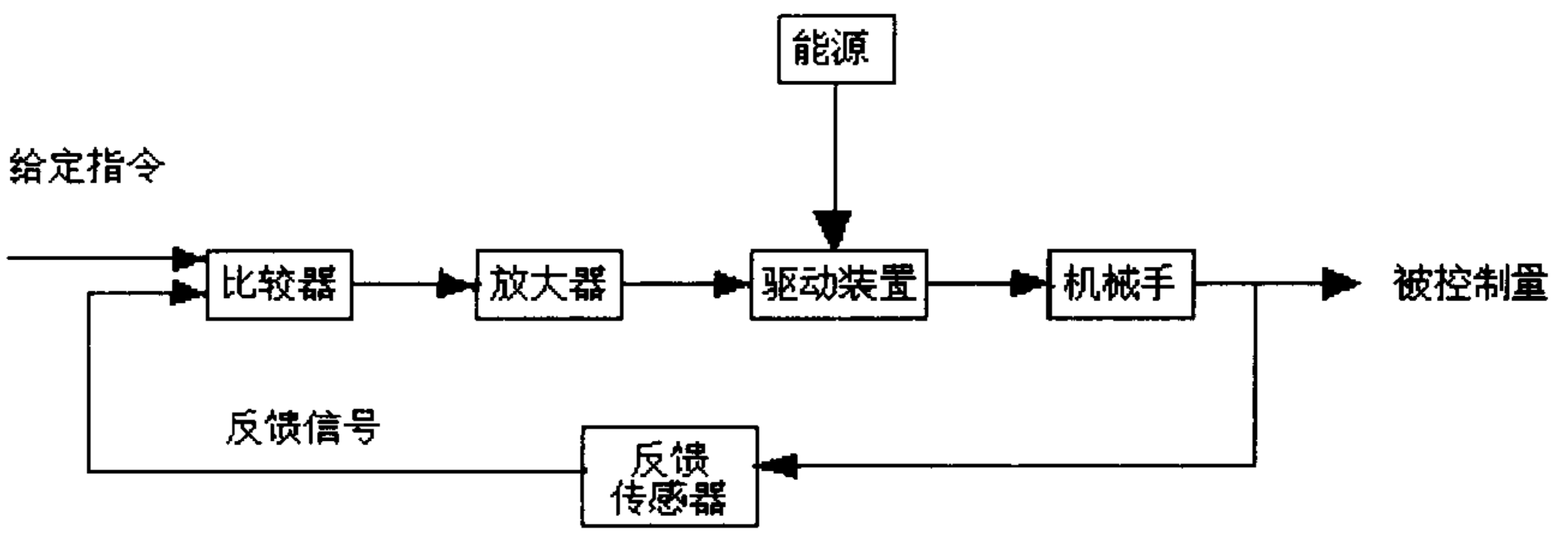
非伺服机器人工作能力比较有限，它们往往涉及那些叫做“终点”、“抓放”或“开关”式机器人，尤其是“有限顺序”机器人。这种机器人按照预先编好的程序顺序进行工作，使用终端限位开关、制动器、插销板和定序器来控制机器人机械手的运动。

● 伺服机器人

伺服机器人比非伺服机器人有更强的工作能力，结构更复杂，价格也更昂贵，但在某些情况下可靠性下降。如附图 3 所示。伺服系统的被控制量可为机器人端部执行装置的位置、速度、加速度和力等。通过反馈传感器取得的反馈信号与来自给定装置的综合信号，经过比较器加以比较后，得到误差信号，经过放大后用以激发机器人的驱动装置，进而带动末端执行装置以一定规律运动，到达预期的位置或速度等，这就

是一个反馈控制系统。

伺服控制机器人又可分为点位伺服控制和连续路径伺服控制两种，这里就不详细介绍了。




附图 3　伺服机器人的控制示意图

伺服控制机器人又可分为点位伺服控制和连续路径伺服控制两种，这里就不详细介绍了。

（3）按机器人的智能程度来分

● 一般机器人

一般机器人不具有智能，只具有一般编程能力和操作功能。

● 智能机器人

智能机器人具有不同程度的智能，我们根据它的智能程度又把它分为三类：

① 传感型机器人。具有对收到的传感信息（包括视觉、听觉、触觉、超声及激光等）进行处理，实现控制与操作的功能。

② 交互式机器人。这种机器人让我们通过与计算机系统进行人机对话，实现对它的控制与操作。

③ 自主型机器人。这种机器人无需人的干预，能够在各种环境下自动完成各项拟人任务。

现今的大多数工业机器人智能化程度不是很高，属于传感型机器人。而一部分特种机器人和与人类生活关系密切的服务机器人则须具有较高的智能化，所以自主型智能机器人将是机器人未来的发展方向。

（4）按机器人的用途来分

● 工业机器人

工业机器人主要应用在工农业生产中的制造环境下，如喷漆、焊接、装配、搬运、检验等作业。

● 特种机器人

特种机器人主要是指非制造业环境下的机器人，如水下机器人、爬壁机器人、擦窗机器人、星球探索机器人等。

● 军用机器人

军用机器人主要用于军事方面，如探雷机器人、扫雷机器人、侦察机器人等。

## 3. 机器人学与人工智能

机器人学与人工智能有着十分密切的关系。

（1）人工智能简介

人工智能是英语 artificial intelligence 的译名，现在关于人工智能的概念有多种说法。“人工智能”一词是 1956 年夏，在美国达特默斯学院召开的有关人工智能的会议上，第一次作为学术用语使用的。从 50 年代后期起，除了计算机领域，其他领域的也经常使用人工智能这个词。人们开始认识到计算机不仅能够进行数值计算，而且有解决各个学科的各种问题的可能性。于是，人们开始探索利用计算机来实现人类所有的智能问题。简单的说，这就是人工智能的原始含义。如果对“人工智能”这个术语进行广义解释的话，可以认为它是指用计算机实现人类在生活各个方面的种种行为。

随着人工智能研究的进展，各种问题逐渐明朗。人们逐渐认识到，计算机并不能简单地代替人的智能，要实现人所有的智能是不可能的，同样，机器人也不可能完全代替人。

可以说，人工智能是一门不断发展、不断完善的科学，它的定义、研究方法、研究对象将会随着时间的推移而产生一些变化，有一点可以肯定：人工智能的发展必将对我们的生活产生重大的影响。

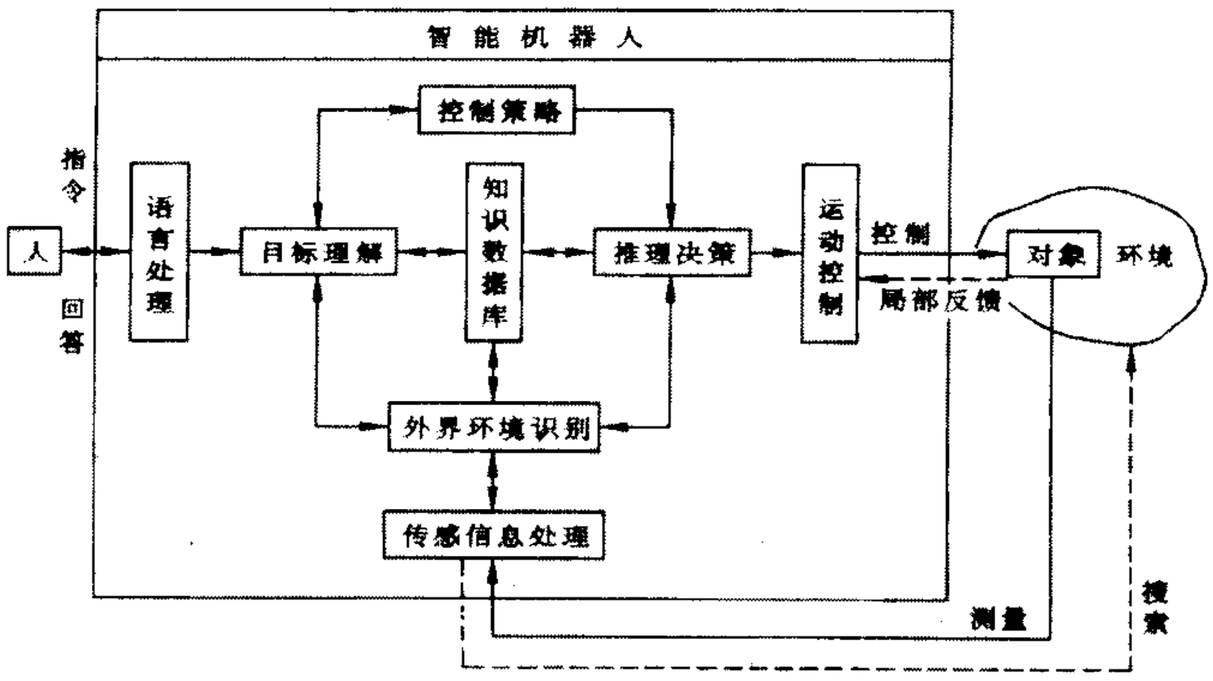
（2）人工智能与机器人学的关系

20 世纪 60 年代中期，机器人学曾经作为人工智能研究的一个课题开始起步，现在虽然与人工智能研究仍保持着密切的联系，但早已形成了一个独立的研究领域。目前，学术界普遍认为，真正意义上的智能机器人必须具有人工智能领域中所研究的各种智能。从这个意义上讲，人工智能又包括在机器人学研究之中。

一方面，机器人学的进一步发展需要人工智能基本原理的指导，并采用各种人工智能的技术；另一方面，机器人学的发展又为人工智能的发展带来了新的生机，产生了新的推动力，并提供一个很好的试验和应用场所。也就是说，人工智能会在机器人学中得到实际应用，并使问题求解、搜索规划、知识表达和智能系统等基本理论得到进一步发展。因此，可以说人工智能和机器人学始终有着密不可分的关系。

（3）智能机器人

智能机器人是机器人技术的发展方向，智能机器人应具有解决问题和理解知觉信息的能力，能适应外界的条件和环境，能根据人的指示进行作业。如附图 4 所示。




附图 4　典型的智能机器人系统的方框图

智能机器人是具有感知、思维和行动功能的机器，是机构学、自动控制、计算机、人工智能、微电子学、光学、通信技术、传感技术、仿生学等多种学科和技术的综合成果。比起一般的工业机器人来，智能机器人具有更大的灵活性、机动性和更广泛的应用领域。作为新一代生产和服务工具，智能机器人在制造领域和非制造领域具有更广泛、更重要的作用，如在核工业、农业、工程机械、建筑、医用、救灾、排险、军事、服务、娱乐等方面，可以代替人去完成各种危险的工作。

同时，智能机器人作为自动化、信息化的装置和设备，完全可以进入网络世界，发挥更大的作用。这对人类开辟新的产业，提高生活水平和生产水平具有十分现实的意义。

目前来看，要实现真正意义上的智能机器人还需要在模式识别技术、问题求解技术、自然语言的处理技术和人机接口技术等方面取得实质性突破。

## 4. 机器人的未来

机器人是 20 世纪人类最伟大的发明之一。从某种意义上讲，一个国家机器人技术水平的高低反映了这个国家综合技术实力的高低。目前机器人技术已在工业领域得到广泛的应用，而且其应用正以惊人的速度不断向非工业领域扩展。

（1）机器人技术发展展望

21 世纪将是机器人技术大发展的世纪，据专家预测，未来的机器人技术发展将有以下一些特点：

● 网络机器人技术

通信网络技术的发展完全能够将各种机器人连接到计算机网络上，并通过网络对机器人进行有效的控制。这种技术包括网络操作控制技术、众多信息组的压缩与扩展技术及信息传输技术等。

● 虚拟机器人技术

许多特种机器人在使用时，遥控是一种主要手段。基于多传感、多媒体和虚拟现实、增强现实（或临场感）的虚拟遥控操作和人机交互将成为一项需要共同发展的技术。

● 多智能体协调控制技术

用于实现决策和操作自治性的、由多智能体组成的群体行为控制是一项具有挑战性的关键技术，包括任务的解释和表达、学习、实时推理和广义反应能力、监控，还包括异况处理、多智能体协调等。

● 微型和微小型技术

有人称微型机器和微型机器人为 21 世纪的尖端技术之一。小型化是机器人发展的一个趋势，其移动灵活方便，速度快，精度高，适于进入大中型工件进行直接作业。未来的微型机器人将采用纳米技术，用于医疗和军事侦察等。

● 采用模块化设计技术

智能机器人和高级工业机器人的结构要力求简单紧凑，其高性能部件甚至全部部件的设计已向模块化方向发展。

● 软机器人技术

许多特种机器人，特别是用于医疗和护理、休闲和娱乐等场合时，经常处于与人共存的环境中，这就要求这种机器人要使人有安全感和亲近感。软机器人技术要求机器人的结构和相应的控制以及所用的传感器是“软”的，能够避免伤害人类。

● 应用领域不断扩展

机器人技术向非制造业和服务业扩展也是一个重要方向。开发适用于非结构环境下工作的机器人将是机器人发展的一个长远方向。这些非制造业包括航天、海洋、军事、建筑、医疗护理、服务、农林、采矿、电力、煤气、供水、下水道工程、建筑物维护、社会福利、家庭自动化、办公自动化和灾害救护等。

（2）机器人将与人类和谐共处

在讨论机器人未来的应用问题时，许多人提出诸如机器人智能是否会超过人类智能，机器人的发展会不会威胁到人类生存等问题，下面我们就来分析一下这些问题。

● 机器人引起社会结构变化

人们希望机器人能够代替人类从事各种劳动，为人类服务，但又担心机器人的发展将引起新的社会问题，人们在期待中含有几分不安。

确实，在过去的近 40 年中，机器人的发展极为迅速，机器人使社会结构正在静悄悄地变化。估计过去人与机器的社会结构，终将为“人—机器人—机器”的社会结构所代替，人们将不得不学会与机器人相处。由于与机器人打交道毕竟不同于与人打交道，所以人们必须改变自己的一些传统观念和思维方式。

但机器人对所有的服务对象都是平等的，现今的机器人还不具备抽象思维能力，也不可能会具有人的感情，所以人们并不需要过分担心。

● 机器人不会威胁人类

人类出现的另一个担忧是：会不会有一天，机器人征服了人类以后，把人关进笼子里，让机器人们来参观，就像今天人类去动物园参观猴子一样，并在笼子门口写着“看，这就是我们的祖先”。

这种担心随着科幻小说和电影、电视、网络的传播，已经十分普遍了。造成这种担心主要有两个原因：一是人们对机器人还不够了解；二是现代社会矛盾在人们心理上的反应。比如，西方社会在使用机器人后，给工人带来失业的恐惧。但实际上，机器人的智能是不会超过人类的。

首先，机器人的发展与人类的进化在本质上是完全不同的，至少在可见的未来是不同的。机器人要由人去设计制造，它们既不是生物，也不是生物结构，不是由生命物质构成的，而仅仅是一种电子机械装置。即使有智能的机器人，它们的智能也不是生命现象，而是非生命的模仿。再先进的机器人技术也是由人类创造的，并由人掌握与控制。人是一切技术的主人，而不是技术的奴隶。

其次，任何先进技术都是一把双刃剑，正确使用将为人类造福，反之则带来灾难，如原子弹。机器人技术也不例外。

未来的某些高智能机器人的某些功能，可能超过人类。但从总体上来看，机器人的智能是不会超过人类的，这一点毋庸置疑。

● 失业问题

机器人能够代替人类进行各种体力劳动和脑力劳动，被称为“钢领”工人。因此，有一部分工人和技术人员暂时失去自己的工作岗位，产生失业现象，这也是不可回避的事实。

对这个问题应该有个正确的认识。任何先进的机器设备，都会提高劳动生产率和产品质量，创造出更多的社会财富，也就必然会提供更多的就业机会，这已被人类的生产发展史所证明。不妨回顾一下，火车、汽车、轮船和飞机的发明，的确曾夺去了不少挑夫、车夫、船夫的饭碗，但是这些运输工具产业的形成，不仅创造出了更多的

就业岗位，还大大促进了运输业和旅游业等相关产业的发展，从而使得更多的人得以就业，这也是有目共睹的史实。计算机的发明也是一个很好的例子。

机器人技术的发展具有同样的道理。对于机器人造成的工人再就业问题，我们要有预见性，并及时采取措施，这是社会工作者和企业管理者应着力解决的问题。英国的一位著名政治家曾经说过，“日本机器人的数量居于世界各国之首，而失业人口最少；英国机器人数量在发达国家中最少，而失业人口却居高不下。”这其中的道理，对中国也是适用的。

所以，机器人的发展和进步只会使我们的生活更加美好，使我们的社会更加文明。