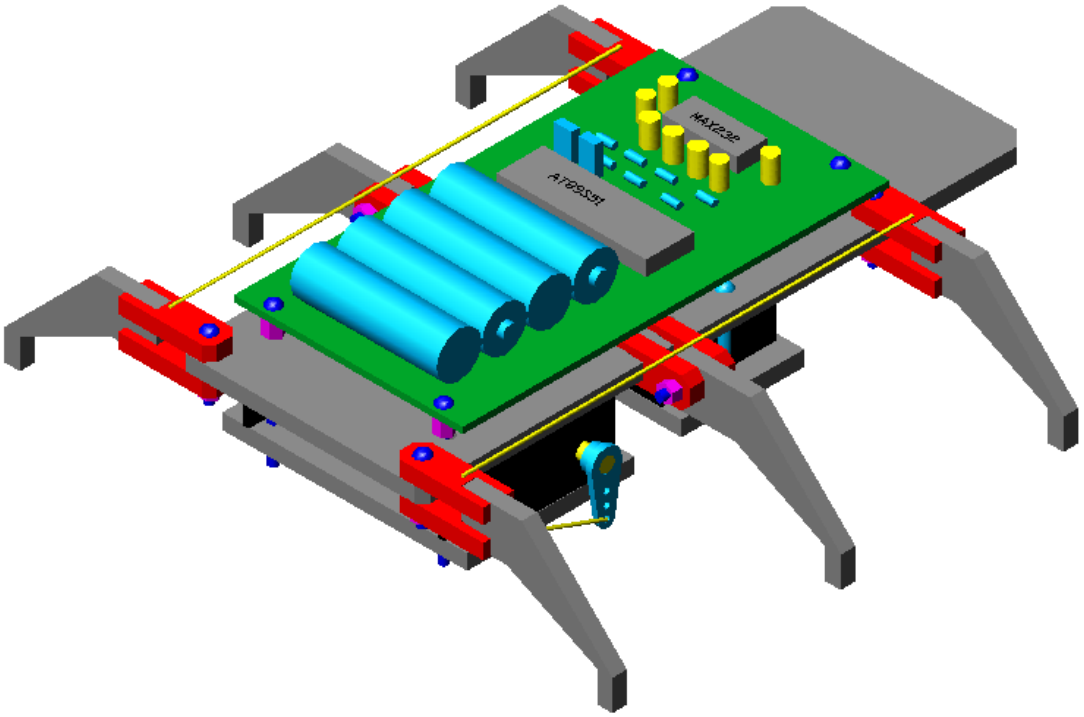
六足爬虫机器人设计



设计人：李海鹰

日期：2004 年 9 月 30 日

目　　录

前言

今年年初，学校为参加中央电视台举办的第三届全国大学生机器人电视大赛，组建了机器人制作小组。我积极参加，有幸成为了其中的一员。因为我们以前没有参加过类似的比赛，也没有制作机器人的经验。可以说我们什么都是从零开始，边学习边制作。通过这半年多的制作过程，我从中学到了很多书本上学不到的东西，也得到了很好的学习与锻炼的机会。

最初，我们组建了机器人制作实验室。到五金机电市场购买了必要的工具和一些制作材料。然后开始制作实验机器人的身体——框架。

实验机器人的框架我们是使用轻型万能角钢制作的，这种角钢的两侧都有间隔均匀的孔槽，可以很方便的用螺栓进行连接。用不同长度的角钢组合后，就可以得到不同大小的立方体和长方体及多边形。机器人身体的框架就搭建好了。在它的上面将装上：机器人的大脑——可编程控制器、机器人的眼睛耳朵——传感器、机器人的腿——驱动轮、机器人的手臂——机械传动专制、机器人的心脏——电池……之所以使用轻型万能角钢，主要是因为是在制作试验机型，而轻型万能角钢安装拆卸方便和便于修改长度，调整设计。

实验机器人定型后，就照其尺寸用不锈钢方管焊接制作机器人的身体。再在上面进行打孔等工作，后就可以将机器人的其它部分安装上去。这样一个机器人就制作好了。

下面我介绍一下机器人的基本组成部分：

（一）、机器人的大脑

它可以有很多叫法，可以叫做：可编程控制器、微控制器，微处理器，处理器或者计算器等，不过这都不要紧，通常微处理器是指一块芯片，而其它的是一整套控制器，包括微处理器和一些别的元件。任何一个机器人大脑就必须要有这块芯片，不然就称不上机器人了。在选择微控制器的时候，主要要考虑：处理器的速度，要实现的功能，ROM 和 RAM 的大小，I/O 端口类型和数量，编程语言以及功耗等。

其主要类型有：单片机、PLC、工控机、PC 机等。

单有这些硬件是不够的，机器人的大脑还无法运行。只有在程序的控制下，它才能按我们的要求去工作。可以说程序就是机器人的灵魂了。而程序是由编程语言所编写的。

编程语言是一个控制器能够接受的语言类型，一般有 C 语言，汇编语言或者 basic 语言等，这些通常能被高级一点的控制器直接执行，因为在高级控制器里面内置了编译器能够直接把一些高级语言翻译成机器码。微处理器将执行这些机器码，并对机器人进行控制。

（二）、机器人的眼睛耳朵

传感器，是机器人的感觉器官，是机器人和现实世界之间的纽带，使机器人

能感知周围的环境情况。其主要有：光电传感器、红外传感器、力传感器、超声波传感器、位置和姿态传感器等等。下面我将就几种常用传感器进行介绍：

1、光电传感器：光电传感器的原理是光电效应。其主要用途是颜色识别（机器人就可以沿着地上的线条行进了）和光电编码等。

2、红外传感器：红外传感器是用来测量距离和感知周围情况的。因为发射出去的红外信号在一定距离内遇到物体就会反射回来。通过发送红外线信号，并接收反射回来的信号，机器人就可以感知前方或身体周围的情况，做出相应的调整（如：倒退或绕行等）。

3、力传感器：力传感器是用来检测碰撞或者接触信号的，比如机械手的应用，当你放一个东西到机械手的时候，机械手自动抓住它，它就需要力传感器检测东西抓的紧不紧。典型的力传感器是微动开关和压敏传感器。微动开关其实就是一个小开关，通过调节开关上的杠杆长短，能够调节触动开关的力的大小。用来做碰撞检测这是最好不过了。但是这种传感器必须事先确定好力的阀值，也就是说只能实现硬件控制（开还控制）。而压敏传感器是能根据受力大小，自动调节输出电压或者电流，从而可以实现软件控制（闭环控制）。

4、超声波传感器：超声波传感器是从蝙蝠那里学来的，通过把发射出的信号与接收到的信号进行对比，就可以测定周围是否有障碍物，及障碍物的距离，也属于距离探测传感器，能提供交远的探测范围，而且还能提供在一个范围内的探测而不是一条线的探测。

5、位置和姿态传感器：机器人在移动或者动作的时候必须时时刻刻知道自己的姿态动作，否则就会产生控制中的一个开环问题，没有反馈，无法获知运动是否正确。 位置传感器和姿态传感器就是用来解决这个问题的。常用的有光电编码器，由于机器人的执行机构一般是电机驱动，通过计算电机转的圈数，可以得出电机带动部件的大致位置，编码器就是这样一种传感器，它一般和电机轴或者转动部件直接连接，电机或者转动部件转了多少圈或者角度能够通过编码器读出，控制软件再根据读出数据进行位置估计计算。还有一种是陀螺仪，这是利用陀螺原理制作的传感器，主要可以测得移动机器人的移动加速度，转过的角度等信息。

（三）、机器人的腿——驱动器与驱动轮

驱动器就是驱动机器人的动的部件。最常用的是电机了。当然还有液压，气动等别的驱动方式。一个机器人最主要的控制量就是控制机器人的移动，无论是自身的移动还是手臂等关节的移动，所以机器人驱动器中最根本和本质的问题就是控制电机，控制电机转的圈数，就可以控制机器人移动的距离和方向，机械手臂的弯曲的程度或者移动的距离等。所以，第一个要解决的问题就是如何让电机能根据自己的意图转动。一般来说，有专门的控制卡和控制芯片来进行控制的。有了这些控制卡和芯片，我们所要做的就是把微控制器和这些连接起来，然后就可以用程序来控制电机了。第二个问题是控制电机的速度，在机器人上的实际表现就是机器人或者手臂的实际运动速度了，机器人走的快慢全靠电机的转速，这样，我们就要求控制卡对电机有速度控制。电机目前常用的有两种，步进电机和直流电机。下面我将就这两种电机进行介绍：

1、直流电机：这是最最普通的电机了。直流电机最大的问题是你没法精确控制电机转的圈数，也就前面所说的位置控制。你必须加上一个编码盘，来进行反馈，来获得实际转的圈数。但是直流电机的速度控制相对就比较简单，用一种叫 PWM（脉宽调速）的调速方法可以很轻松的调节电机速度。现在也有很多控制芯片带调速功能的。选购时要考虑的参数是电机的输出力矩，电机的功率，电机的最高转速。

2、步进电机：看名字就知道了，它是一步一步前进的。也就是说，它可以一个角度一个角度旋转，不象直流电机，你可以很轻松的调节步进电机的转角位置，如果你发一个转 10 圈的指令，步进电机就不会转 11 圈，但是如果是直流电机，由于惯性作用，它可能转 11 圈半。步进电机的调速是通过控制电机的频率来获得的。一般控制信号频率越高，电机转的越快，频率越低，转的越慢。选购时要考虑的参数是电机的输出力矩，电机的功率，每个脉冲电机的最小转角。

还有就是关于输出的动力，要说明一下：一般情况下，电机都没法直接带动轮子或者手臂，因为速度过高力矩不够大，所以我们需要加上一个减速箱来增加电机的输出力矩，但是代价是电机速度的减小，比如一个 1：250 的齿轮箱，会让你电机的输出力矩增大 250 倍，但是速度只有原来的 1/250 了。首先计算出机器人所需要的速度与力矩大小，然后根据速度与力矩去选择电机与减速器。

（四）、机器人的手臂——机械传动专制

机械传动专制就是，由电机驱动的一些杆件和机构（如：凸轮机构、螺杆机构等），用以实现机械手臂的上升、下降、伸缩、弯曲等动作。通常运用的机构有四杆机构、凸轮机构、螺杆机构、摇臂等。

（五）、机器人的心脏——电池

电池为机器人的控制系统与驱动系统提供能源供应。主要有：电瓶及可充电电池、电池。

前面介绍了机器人的一些基本知识，但这是远远不够的。机器人学科，是在多学科基础上发展起来的综合性技术。机器人技术涉及机械、电子、计算机、语言学和人工智能等许多学科。现在机器人已经应用在人类社会生活的各个领域，发挥着越来越重要的影响。

我利用暑假的时间设计了一个六足爬虫机器人，用日立（HITACHI）的录像机遥控器来对它进行控制。基本原理是：遥控器发出红外学号，机器人通过红外接收器接收倒红外信号后，对信号进行解码，并以存储的代码进行比较，确定指令的含义，后可以实现前进、后退、左转、右转及发声等功能。控制系统我使用的是 AT89S51 单片机，编程语言使用的是汇编语言，动力系统使用的是微型伺服马达，能源系统使用的是 9V 电池。下面我将就具体设计进行介绍。

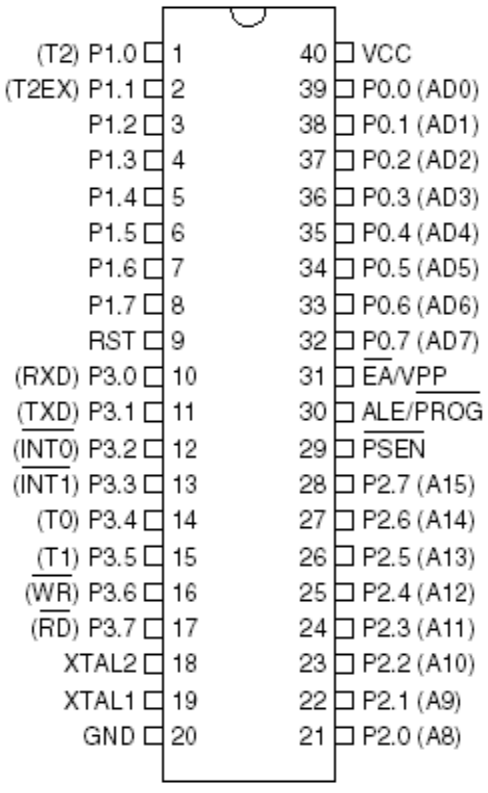
一、AT89S51 单片机简介

AT89S51 为 ATMEL 所生产的可电气烧录清洗的 8051 相容单芯片，其内部程序代码容量为 4KB

（一）、AT89S51 主要功能列举如下：

1、为一般控制应用的 8 位单芯片
2、晶片内部具时钟振荡器（传统最高工作频率可至 12MHz）
3、内部程式存储器（ROM）为 4KB
4、内部数据存储器（RAM）为 128B
5、外部程序存储器可扩充至 64KB
6、外部数据存储器可扩充至 64KB
7、32 条双向输入输出线，且每条均可以单独做 I/O 的控制
8、5 个中断向量源
9、2 组独立的 16 位定时器
10、1 个全多工串行通信端口
11、8751 及 8752 单芯片具有数据保密的功能
12、单芯片提供位逻辑运算指令

（二）、AT89S51 各引脚功能介绍：



AT89S51

VCC：

AT89S51 电源正端输入，接+5V。

VSS：

电源地端。

XTAL1：

单芯片系统时钟的反相放大器输入端。

XTAL2：

系统时钟的反相放大器输出端，一般在设计上只要在 XTAL1 和 XTAL2 上接上一只石英振荡晶体系统就可以动作了，此外可以在两引脚与地之间加入一 20PF 的小电容，可以使系统更稳定，避免噪声干扰而死机。

RESET：

AT89S51 的重置引脚，高电平动作，当要对晶片重置时，只要对此引脚电平提升至高电平并保持两个机器周期以上的时间，AT89S51 便能完成系统重置的各项动作，使得内部特殊功能寄存器之内容均被设成已知状态，并且至地址

0000H 处开始读入程序代码而执行程序。

EA/Vpp：

"EA"为英文"External Access"的缩写，表示存取外部程序代码之意，低电平动作，也就是说当此引脚接低电平后，系统会取用外部的程序代码（存于外部 EPROM 中）来执行程序。因此在 8031 及 8032 中，EA 引脚必须接低电平，因为其内部无程序存储器空间。如果是使用 8751 内部程序空间时，此引脚要接成高电平。此外，在将程序代码烧录至 8751 内部 EPROM 时，可以利用此引脚来输入 21V 的烧录高压（Vpp）。

ALE/PROG：

ALE 是英文"Address Latch Enable"的缩写，表示地址锁存器启用信号。AT89S51 可以利用这支引脚来触发外部的 8 位锁存器（如 74LS373），将端口 0 的地址总线（A0～A7）锁进锁存器中，因为 AT89S51 是以多工的方式送出地址及数据。平时在程序执行时 ALE 引脚的输出频率约是系统工作频率的 1/6，因此可以用来驱动其他周边晶片的时基输入。此外在烧录 8751 程序代码时，此引脚会被当成程序规划的特殊功能来使用。

PSEN：

此为"Program Store Enable"的缩写，其意为程序储存启用，当 8051 被设成为读取外部程序代码工作模式时（EA=0），会送出此信号以便取得程序代码，通常这支脚是接到 EPROM 的 OE 脚。AT89S51 可以利用 PSEN 及 RD 引脚分别启用存在外部的 RAM 与 EPROM，使得数据存储器与程序存储器可以合并在一起而共用 64K 的定址范围。

PORT0（P0.0～P0.7）：

端口 0 是一个 8 位宽的开路汲极（Open Drain）双向输出入端口，共有 8 个位，P0.0 表示位 0，P0.1 表示位 1，依此类推。其他三个 I/O 端口（P1、P2、P3）则不具有此电路组态，而是内部有一提升电路，P0 在当做 I/O 用时可以推动 8 个 LS 的 TTL 负载。如果当 EA 引脚为低电平时（即取用外部程序代码或数据存储器），P0 就以多工方式提供地址总线（A0～A7）及数据总线（D0～D7）。设计者必须外加一锁存器将端口 0 送出的地址栓锁住成为 A0～A7，再配合端口 2 所送出的 A8～A15 合成一完整的 16 位地址总线，而定址到 64K 的外部存储器空间。

PORT2（P2.0～P2.7）：

端口 2 是具有内部提升电路的双向 I/O 端口，每一个引脚可以推动 4 个 LS 的 TTL 负载，若将端口 2 的输出设为高电平时，此端口便能当成输入端口来使用。P2 除了当做一般 I/O 端口使用外，若是在 AT89S51 扩充外接程序存储器或数据存储器时，也提供地址总线的高字节 A8～A15，这个时候 P2 便不能当做 I/O 来使用了。

PORT1（P1.0～P1.7）：

端口 1 也是具有内部提升电路的双向 I/O 端口，其输出缓冲器可以推动 4 个 LS TTL 负载，同样地若将端口 1 的输出设为高电平，便是由此端口来输入数据。如果是使用 8052 或是 8032 的话，P1.0 又当做定时器 2 的外部脉冲输入脚，而 P1.1 可以有 T2EX 功能，可以做外部中断输入的触发脚位。

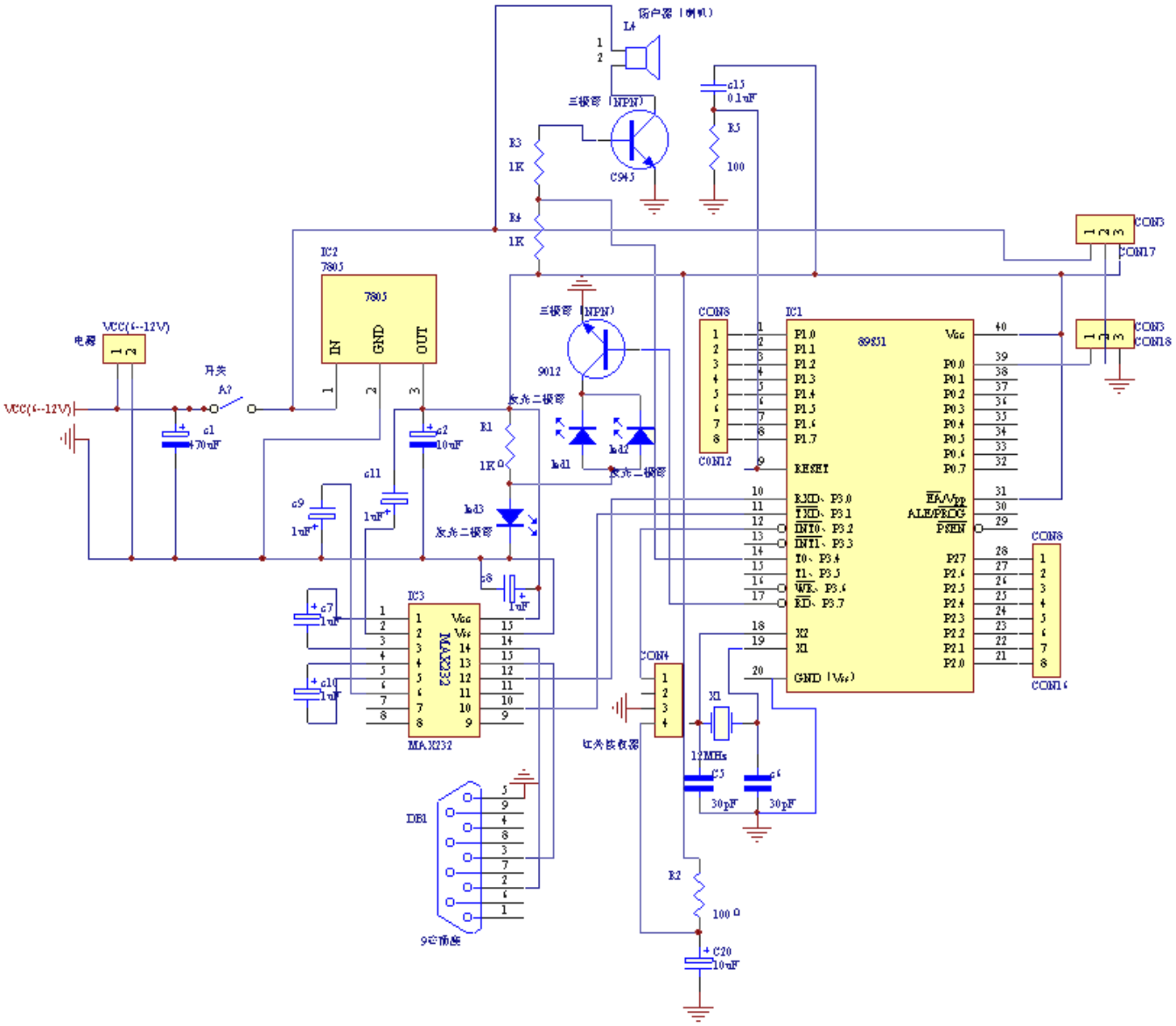
PORT3（P3.0～P3.7）：

端口 3 也具有内部提升电路的双向 I/O 端口，其输出缓冲器可以推动 4 个 TTL 负载，同时还多工具有其他的额外特殊功能，包括串行通信、外部中断控制、

计时计数控制及外部数据存储器内容的读取或写入控制等功能。
其引脚分配如下：

P3.0：RXD，串行通信输入。
P3.1：TXD，串行通信输出。
P3.2：INT0，外部中断 0 输入。
P3.3：INT1，外部中断 1 输入。
P3.4：T0，计时计数器 0 输入。
P3.5：T1，计时计数器 1 输入。
P3.6：WR：外部数据存储器的写入信号。
P3.7：RD，外部数据存储器的读取信号。

二、控制系统电路图

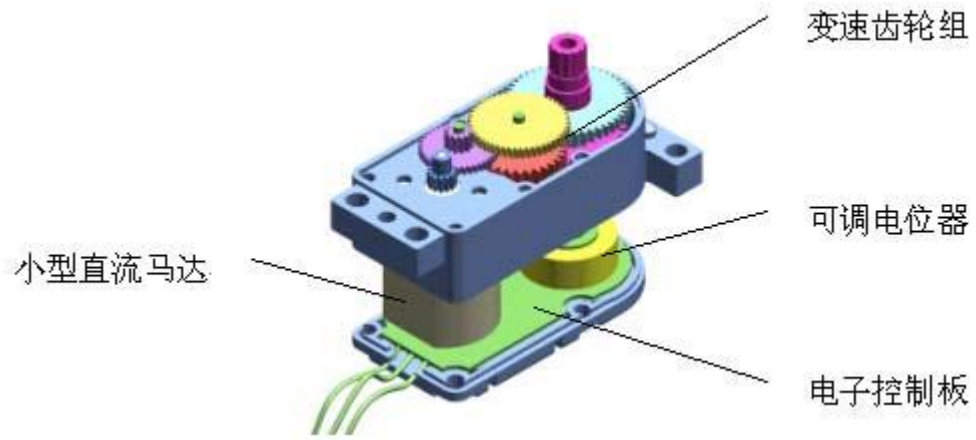


控制系统电路图

三、微型伺服马达原理与控制

（一）、微型伺服马达内部结构

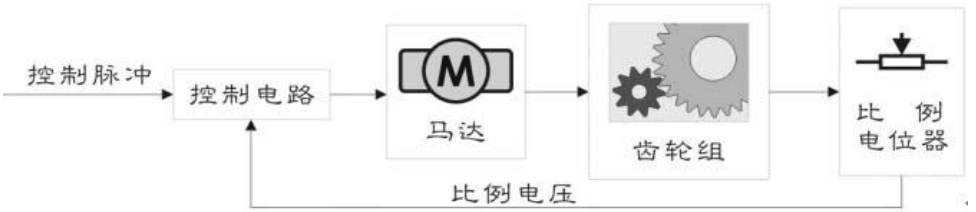
一个微型伺服马达内部包括了一个小型直流马达；一组变速齿轮组；一个反馈可调电位器；及一块电子控制板。其中，高速转动的直流马达提供了原始动力，带动变速（减速）齿轮组，使之产生高扭力的输出，齿轮组的变速比愈大，伺服马达的输出扭力也愈大，也就是说越能承受更大的重量，但转动的速度也愈低。



微型伺服马达内部结构图

（二）、微行伺服马达的工作原理

一个微型伺服马达是一个典型闭环反馈系统，其原理可由下图表示：



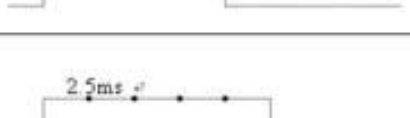
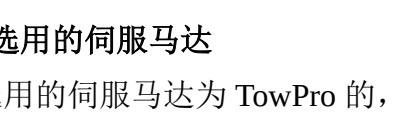
微行伺服马达工作原理图

减速齿轮组由马达驱动，其终端（输出端）带动一个线性的比例电位器作位置检测，该电位器把转角坐标转换为一比例电压反馈给控制线路板，控制线路板将其与输入的控制脉冲信号比较，产生纠正脉冲，并驱动马达正向或反向地转动，使齿轮组的输出位置与期望值相符，令纠正脉冲趋于为 0，从而达到使伺服马达精确定位的目的。

（三）、伺服马达的控制

标准的微型伺服马达有三条控制线，分别为：电源、地及控制。电源线与地线用于提供内部的直流马达及控制线路所需的能源，电压通常介于 4V—6V 之间，该电源应尽可能与处理系统的电源隔离（因为伺服马达会产生噪音）。甚至小伺服马达在重负载时也会拉低放大器的电压，所以整个系统的电源供应的比例必须合理。

输入一个周期性的正向脉冲信号，这个周期性脉冲信号的高电平时间通常在 1ms—2ms 之间，而低电平时间应在 5ms 到 20ms 之间，并不很严格，下表表示出一个典型的 20ms 周期性脉冲的正脉冲宽度与微型伺服马达的输出臂位置的关系：

输入正脉冲宽度（周期为 20ms）	伺服马达输出臂位置
	 ≈ -90°
	 ≈ -45°
	 ≈ 0°
	 ≈ 45°
	 ≈ 90°

（四）、选用的伺服马达

我选用的伺服马达为 TowPro 的，型号为 SG303。其主要技术参数如下：

- 转速：0.23 秒 / 60 度。
- 力矩：3.2kg • cm。
- 尺寸：40.4mm×19.8mm×36mm。
- 重量：37.2g。
- 5V 电源供电。

控制周期脉冲宽度为 20ms。送出不同的正脉冲宽度是，就可以得到不同的控制效果。控制正脉冲宽度如下：

- 正脉冲宽度为 0.3ms 时，伺服马达反转。
- 正脉冲宽度为 2.5ms 时，伺服马达正转。
- 正脉冲宽度为 1.4ms 时，伺服马达回到中点。

四、红外遥控

家中许多的电器产品都有遥控的功能，例如电视机、录像机、VCD、空调等家电产品，它们都是以红外遥控的方式进行遥控。

（一）、 红外遥控系统

通用红外遥控系统由发射和接收两大部分组成，应用编/解码专用集成电路芯片来进行控制操作，如图 1 所示。发射部分包括键盘矩阵、编码调制、LED 红外发送器；接收部分包括光、电转换放大器、解调、解码电路。

（二）、 遥控发射器及其编码

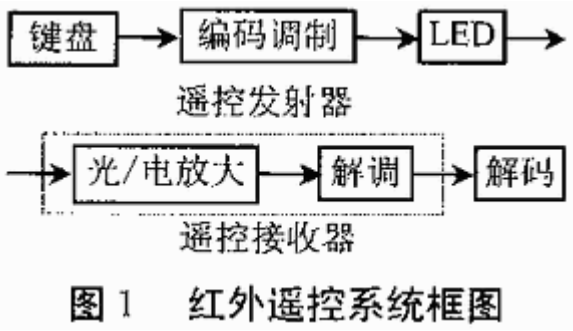


图 1　红外遥控系统框图

遥控发射器专用芯片很多，根据编码格式可以分成两大类，这里我们以运用比较广泛，解码比较容易的一类来加以说明，现以日本 NEC 的 uPD6121G 组成发射电路为例说明编码原理。当发射器按键按下后，即有遥控码发出，所按的键不同遥控编码也不同。这种遥控码具有以下特征：

采用脉宽调制的串行码，以脉宽为 0.565ms、间隔 0.56ms、周期为 1.125ms 的组合表示二进制的“0”；以脉宽为 0.565ms、间隔 1.685ms、周期为 2.25ms 的组合表示二进制的“1”，其波形如图 2 所示。

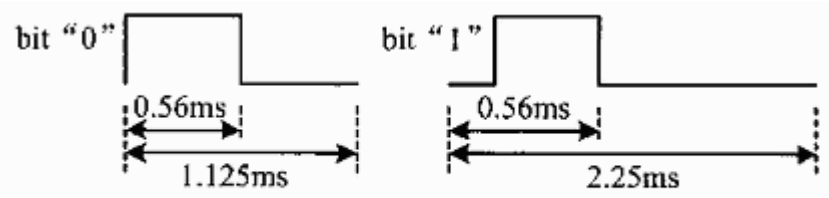


图 2　遥控码的“0”和“1”

上述“0”和“1”组成的 32 位二进制码经 38kHz 的载频进行二次调制以提高发射效率，达到降低电源功耗的目的。然后再通过红外发射二极管产生红外线向空间发射。

遥控编码是连续的 32 位二进制码组，其中前 16 位为用户识别码，能区别不同的电器设备，防止不同机种遥控码互相干扰。该芯片的用户识别码固定为十六进制 01H；后 16 位为 8 位操作码（功能码）及其反码。UPD6121G 最多额 128 种不同组合的编码，如图 3 所示。

遥控器在按键按下后，周期性地发出同一种 32 位二进制码，周期约为 108ms。一组码本身的持续时间随它包含的二进制“0”和“1”的个数不同而不同，大约在 45～63ms 之间，图 4 为发射波形图。

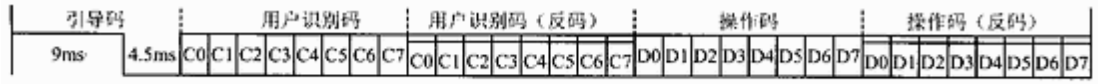


图 3　遥控信号编码波形图

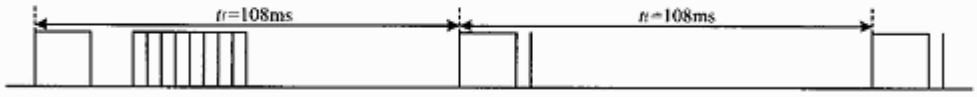


图 4　遥控信号的周期性波形

（三）、红外接收模块

红外接收模块

左图为一常用的红外接收模块。其内部含有高频的滤波电路，专门用来滤除红外线合成信号的载波信号（38KH），并送出接收到的信号。当红外线合成信号进入红外接收模块，在其输出端便可以得到原先发射器发出的数字编码，只要经过单片机解码程序进行解码，便可以得知按下了哪一个按键，而做出相应的控制处理，完成红外遥控的动作。

（四）、红外解码程序设计

红外解码程序主要工作为等待红外线信号出现，并跳过引导信号，开始收集连续 32 位的表面数据，存入内存的连续空间。位信号解码的原则是：以判断各个位的波宽信号来决定高低信号。位解码原理如下：

- 解码为 0：低电平的宽度 0.56ms+高电平的宽度 0.56ms。
- 解码为 1：低电平的宽度 1.68ms+高电平的宽度 0.56ms。

程序中必须设计一精确的 0.1ms 延时时间作为基础时间，以计数实际的波形宽度，若读值为 5 表示波形宽度为 0.5ms，若读值为 16 表示波形宽度为 1.6ms，以此类推。高电平的宽度 1.12ms 为固定，因此可以直接判断低电平的宽度的计数值 5 或时 16，来确定编码为 0 或是 1。程序中可以减法指令 SUBB 来完成判断，指令“SUBB　A，R2”中若 R2 为计数值，A 寄存器设为 8，就可如下：

- 当“8－R2”有产生借位，借位标志 C=1，表示编码为 1。
- 当“8－R2”无产生借位，借位标志 C=0，表示编码为 0。

将借位标志 C 经过右移指令“RRC　A”转入 A 寄存器中，再经由 R0 寄存器间接寻址存入内存中。

详细解码程序请参看“红外遥控爬虫机器人 ASM 程序”中的“红外解码子程序”。

五、控制程序

```
; 红外遥控爬虫机器人 ASM 程序
;-------------------------------------------------------------------------------------------------------------------------
HOME     EQU   14                ; 伺服马达回到中点时间常数
BACK     EQU   3                 ; 伺服马达反转时间常数
FOR      EQU   25                ; 伺服马达正转时间常数
; -----------------------------------------------------------------------------------------------------------------------
; 遥控器按键 1～6 比较码
CODE_K1  EQU   19H               ; 机器人前进比较码
CODE_K2  EQU   18H               ; 机器人后退比较码
CODE_K3  EQU   0AH               ; 机器人左转比较码
CODE_K4  EQU   09H               ; 机器人右转比较码
CODE_K5  EQU   0BH               ; 机器人回到中点比较码
CODE_K6  EQU   14H               ; 机器人行走启动进比较码
; -----------------------------------------------------------------------------------------------------------------------
IRCOM    EQU   30H               ; 红外线信号解码数据放置变量起始地址
COM      EQU   32H               ; 比较第 3 字节变量
; -----------------------------------------------------------------------------------------------------------------------
IRIN     EQU   P3.2              ; 红外线 IR 信号输入位引脚定义
WLED     EQU   P3.7              ; 发光二极管引脚定义
SPK      EQU   P3.4              ; 压电喇叭引脚定义
DJZ      EQU   P1.0              ; 中间伺服马达引脚定义
DJL      EQU   P1.1              ; 左侧伺服马达引脚定义
DJR      EQU   P1.2              ; 右侧伺服马达引脚定义
; -----------------------------------------------------------------------------------------------------------------------
     ORG   0H                    ; 程序代码由地址 0 开始执行
     JMP   BEGIN                 ; 进入主程序
; -----------------------------------------------------------------------------------------------------------------------
BEGIN:
     CLR   DJZ                   ; 关闭中间伺服马达
     CLR   DJL                   ; 关闭左侧伺服马达
     CLR   DJR                   ; 关闭右侧伺服马达
     CLR   SPK                   ; 关闭压电喇叭
     CALL  LED_BL                ; 发光二极管闪烁，表示程序开始执行
     CALL  BZ                    ; 压电喇叭发出嘀的一声
     CALL  GO_HOME               ; 全部伺服马达回到中点
     CALL  LED_BL                ; 发光二极管闪烁，表示机器人准备完毕
     CALL  BZ                    ; 压电喇叭发出嘀的一声
     CALL  QD                    ; 运行行走启动子程序，摆好行走姿态
     SETB  IRIN                  ; 红外线信号 IR 输入位设为高电平，准备接收红外信号
LOOP:
     MOV   R0,#IRCOM             ; 设置 IR 解码起始地址
     CALL  IR_IN                 ; 进行 IR 解码
     CALL  OP                    ; 进行解码比较，并控制机器人动作
     JMP LOOP                    ; 继续循环执行
; -----------------------------------------------------------------------------------------------------------------------
DELAY:   MOV    R6,#50           ; 10ms 延时子程序
D1:      MOV    R7,#99
         DJNZ   R7,$
         DJNZ   R6,D1
         DJNZ   R5,DELAY
         RET
; -----------------------------------------------------------------------------------------------------------------------
LED_BL:  MOV R1,#4               ; 发光二极管闪烁子程序
LE1:     CPL    WLED             ; 发光二极管反向
         MOV    R5,#10
         CALL   DELAY            ; 进行 100ms 延时
         DJNZ   R1,LE1
         RET
; -----------------------------------------------------------------------------------------------------------------------
BZ:      MOV    R6,#0            ; 压电喇叭发声子程序
```

```
B1:     SETB   SPK                    ；压电喇叭得电，开始发声
        DJNZ   R6,B1
        MOV    R5,#5
        CALL   DELAY                  ；进行 50ms 延时
        CLR    SPK                    ；关闭压电喇叭
        RET
; ------------------------------------------------------------------------------------------------------------------------
DEL:                                  ；0.1ms 延时子程序
          MOV    R5,#1
DELAY1:
          MOV    R6,#2
E1:       MOV    R7,#22
E2:       DJNZ   R7,E2
          DJNZ   R6,E1
          DJNZ   R5,DELAY1
          RET
; ------------------------------------------------------------------------------------------------------------------------
IR_IN:                                ；红外解码子程序
I1:   JNB    IRIN,I2                  ；等待红外 IR 信号出现
       JMP    I1
I2:   MOV    R4,#20                   ；发现红外 IR 信号，延时一下
I20:  CALL   DEL
      DJNZ   R4,I20
      JB     IRIN,I1                  ；确认红外 IR 信号出现
I21:  JB     IRIN,I3                  ；等待 IR 变为高电平
      CALL   DEL
      JMP    I21
I3:   MOV    R3,#0                    ；8 位数清 0
LL:   JNB    IRIN,I4                  ；等待 IR 变为低电平
      CALL   DEL
      JMP    LL
I4:   JB     IRIN,I5                  ；等待 IR 变为高电平
      CALL   DEL
      JMP    I4
I5:   MOV    R2,#0                    ；0.1ms 计数
L1:   CALL   DEL
      JB     IRIN,N1                  ；等待 IR 变为高电平
      MOV    A,#8                     ；设置减数为 8
      CLR    C                        ；清除借位标志 C
      SUBB   A,R2                     ；判断高低位
      MOV    A,@R0                    ；取出内存中原先数据
      RRC    A                        ；右移指令，将借位标志 C 右移进入 A 寄存器中
      MOV    @R0, A                   ；将数据写入内存中
      INC    R3                       ；处理完成一位，R3+1（R3 计数）
      CJNE   R3,#8, LL                ；循环处理 8 位
      MOV    R3,#0                    ；R3 清 0
      INC    R0                       ；处理完成 1 个字节，R0+1（R0 计数）
      CJNE   R0,#34H, LL              ；循环收集到 4 个字节
      JMP    OK                       ；至完成返回
N1:   INC    R2                       ；R2+1（R2 计数）
      CJNE   R2,#30, L1               ；0.1ms 计数过长，时间到自动离开
OK:   RET                             ；完成返回
; ------------------------------------------------------------------------------------------------------------------------
OP:                                          执行解码动作子程序
      MOV    A,COM
      CJNE   A,#CODE_K5, A1          ；对解码进行比较，看是否是回到中点指令，否就转至下一项比较
     CALL   LED_BL                   ；发光二极管闪烁
     CALL   BZ                       ；压电喇叭发出嘀的一声
     CALL   GO_HOME                  ；执行回到中点
     CALL   LED_BL                   ；发光二极管闪烁
     CALL   BZ                       ；压电喇叭发出嘀的一声
     RET
A1:
```

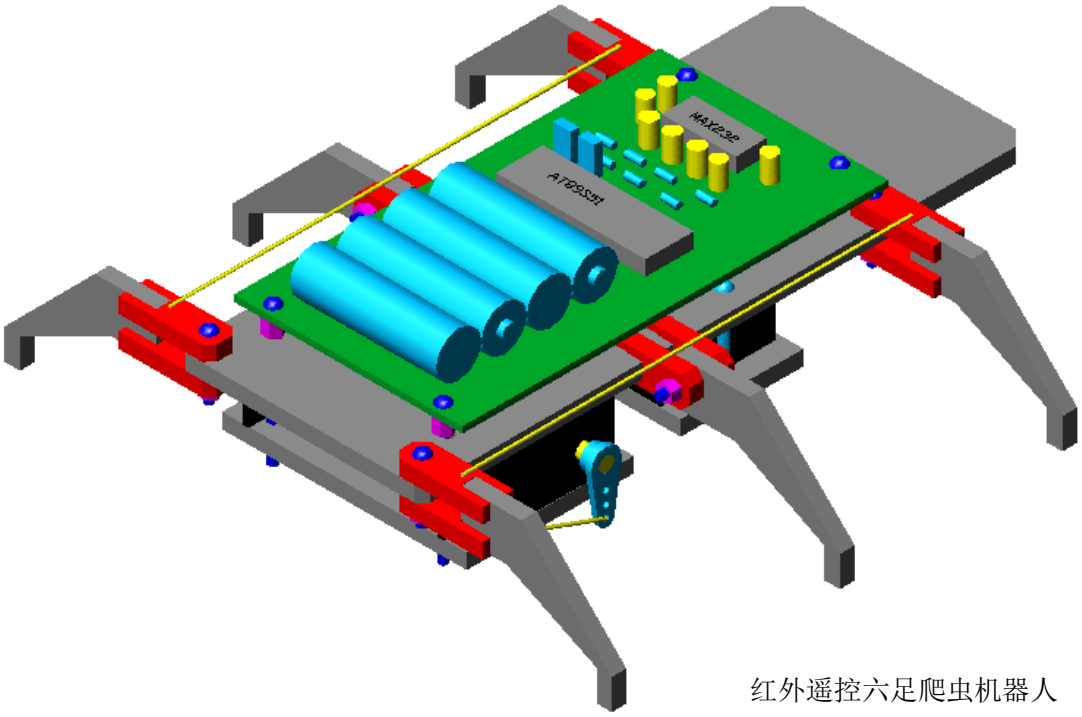
```
      MOV    A,COM
      CJNE   A,#CODE_K1, A2              ；对解码进行比较，看是否是前进指令，否就转至下一项比较
      CALL   BZ                          ；压电喇叭发出嘀的一声
      CALL   GO_FOR                      ；执行前进
       RET
A2:
      MOV    A,COM
      CJNE   A,#CODE_K2, A3              ；对解码进行比较，看是否是后退指令，否就转至下一项比较
      CALL   BZ                          ；压电喇叭发出嘀的一声
       CALL   GO_BACK                    ；执行后退
       RET
A3:        ;L
      MOV    A,COM
      CJNE   A,#CODE_K3, A4              ；对解码进行比较，看是否是左转指令，否就转至下一项比较
      CALL   BZ                          ；压电喇叭发出嘀的一声
       CALL   GO_L                       ；执行左转
       RET
A4:        ;R
       MOV    A,COM
      CJNE   A,#CODE_K4, A5              ；对解码进行比较，看是否是右转指令，否就转至下一项比较
      CALL   BZ                          ；压电喇叭发出嘀的一声
       CALL   GO_R                       ；执行右转
      RET
A5:
       MOV    A,COM
       CJNE   A,#CODE_K6, A6             ；对解码进行比较，看是否是行走启动指令，否就转至下一项
       CALL   LED_BL                     ；发光二极管闪烁
       CALL   BZ                         ；压电喇叭发出嘀的一声
       CALL   QD                         ；执行行走启动
      CALL   LED_BL                      ；发光二极管闪烁
     CALL   BZ                          ；压电喇叭发出嘀的一声
      RET
A6:
    RET                                 ；返回
; ---------------------------------------------------------------------------------------------------------------------------
HOME1:      SETB   DJZ                  ；各伺服电机回中点控制子程序
      SETB   DJL
      SETB   DJR
      MOV    R4,#HOME
G1:   CALL   DEL
      DJNZ   R4,G1
      CLR    DJZ
        CLR    DJL
      CLR    DJR
      MOV    R4,#(200-HOME)
G2:   CALL   DEL
      DJNZ   R4,G2
      RET
; ---------------------------------------------------------------------------------------------------------------------------
GO_HOME:     MOV    R3,#15              ；机器人回中点子程序
H1:    CALL   HOME1
       DJNZ   R3,H1
        RET
; ---------------------------------------------------------------------------------------------------------------------------
DJZ_FOR:        SETB   DJZ              ；中间电机正转子程序
       MOV    R4,#FOR
FZ1:   CALL   DEL
       DJNZ   R4,FZ1
       CLR    DJZ
       MOV    R4,#(200-FOR)
FZ2:   CALL   DEL
       DJNZ   R4,FZ2
       RET
```

```
; ---------------------------------------------------------------------------------------------------------------------------
DJL_FOR:        SETB    DJL                 ；左侧电机正转子程序
        MOV     R4,#FOR
FL1:    CALL    DEL
        DJNZ    R4,FL1
        CLR     DJL
        MOV     R4,#(200-FOR)
FL2:    CALL    DEL
        DJNZ    R4,FL2
        RET
; ---------------------------------------------------------------------------------------------------------------------------
DJR_FOR:        SETB    DJR                 ；右侧电机正转子程序
        MOV     R4,#FOR
FR1:    CALL    DEL
        DJNZ    R4,FR1
        CLR     DJR
        MOV     R4,#(200-FOR)
FR2:    CALL    DEL
        DJNZ    R4,FR2
        RET
; ---------------------------------------------------------------------------------------------------------------------------
DJZ_BACK:       SETB    DJZ                 ；中间电机反转子程序
            MOV     R4,#BACK
DJZBA1:     CALL    DEL
            DJNZ    R4,DJZBA1
            CLR     DJZ
            MOV     R4,#(200-BACK)
DJZB2:      CALL    DEL
            DJNZ    R4,DJZB2
            RET
; ---------------------------------------------------------------------------------------------------------------------------
DJL_BACK:       SETB    DJL                 ；左侧电机反转子程序
            MOV     R4,#BACK
DJLBA1:     CALL    DEL
            DJNZ    R4,DJLBA1
            CLR     DJL
            MOV     R4,#(200-BACK)
DJLB2:      CALL    DEL
            DJNZ    R4,DJLB2
            RET
; ---------------------------------------------------------------------------------------------------------------------------
DJR_BACK:       SETB    DJR                 ；右侧电机反转子程序
            MOV     R4,#BACK
DJRBA1:     CALL    DEL
            DJNZ    R4,DJRBA1
            CLR     DJR
            MOV     R4,#(200-BACK)
DJRB2:      CALL    DEL
            DJNZ    R4,DJRB2
            RET
; ---------------------------------------------------------------------------------------------------------------------------
GO_FOR:     MOV     R3,#5                   ；机器人向前行走子程序
F1:         CALL    DJZ_FOR
            DJNZ    R3,F1
            MOV     R3,#10
F2:         CALL    DJR_BACK
            DJNZ    R3,F2
            MOV     R3,#10
F3:         CALL    DJL_BACK
            DJNZ    R3,F3
            MOV     R3,#5
F4:         CALL    DJZ_BACK
            DJNZ    R3,F4
```

```
            MOV    R3,#10
F5:         CALL   DJL_FOR
            DJNZ   R3,F5
            MOV    R3,#10
F6:         CALL   DJR_FOR
           DJNZ   R3,F6
           RET
; ------------------------------------------------------------------------------------------------------------------------
GO_BACK:       MOV    R3,#10                    ；机器人向后行走子程序
BA1:        CALL   DJL_BACK
           DJNZ   R3,BA1
            MOV    R3,#10
BA2:        CALL   DJR_BACK
           DJNZ   R3,BA2
           MOV    R3,#5
BA4:        CALL   DJZ_FOR
           DJNZ   R3,BA4
            MOV    R3,#10
BA5:        CALL   DJR_FOR
           DJNZ   R3,BA5
            MOV    R3,#10
BA6:        CALL   DJL_FOR
           DJNZ   R3,BA6
            MOV    R3,#5
BA7:        CALL   DJZ_BACK
             DJNZ   R3,BA7
             RET
; ------------------------------------------------------------------------------------------------------------------------
GO_L:       MOV    R3,#10                     ；机器人左转行走子程序
GL1:         CALL   DJL_BACK
            DJNZ   R3,GL1
             MOV    R3,#5
GL2:         CALL   DJZ_FOR
            DJNZ   R3,GL2
             MOV    R3,#10
GL3:         CALL   DJR_BACK
            DJNZ   R3,GL3
            MOV    R3,#10
GL7:         CALL   DJL_FOR
            DJNZ   R3,GL7
            MOV    R3,#5
GL4:         CALL   DJZ_BACK
            DJNZ   R3,GL4
             MOV    R3,#10
GL5:         CALL   DJR_FOR
            DJNZ   R3,GL5
            RET
; ------------------------------------------------------------------------------------------------------------------------
GO_R:       MOV    R3,#10                     ；机器人右转行走子程序
GR1:         CALL   DJR_BACK
            DJNZ   R3,GR1
             MOV    R3,#5
GR2:         CALL   DJZ_FOR
            DJNZ   R3,GR2
             MOV    R3,#10
GR3:         CALL   DJL_BACK
              DJNZ   R3,GR3
              MOV    R3,#10
GR4:         CALL   DJR_FOR
              DJNZ   R3,GR4
             MOV    R3,#5
GR5:         CALL   DJZ_BACK
              DJNZ   R3,GR5
```

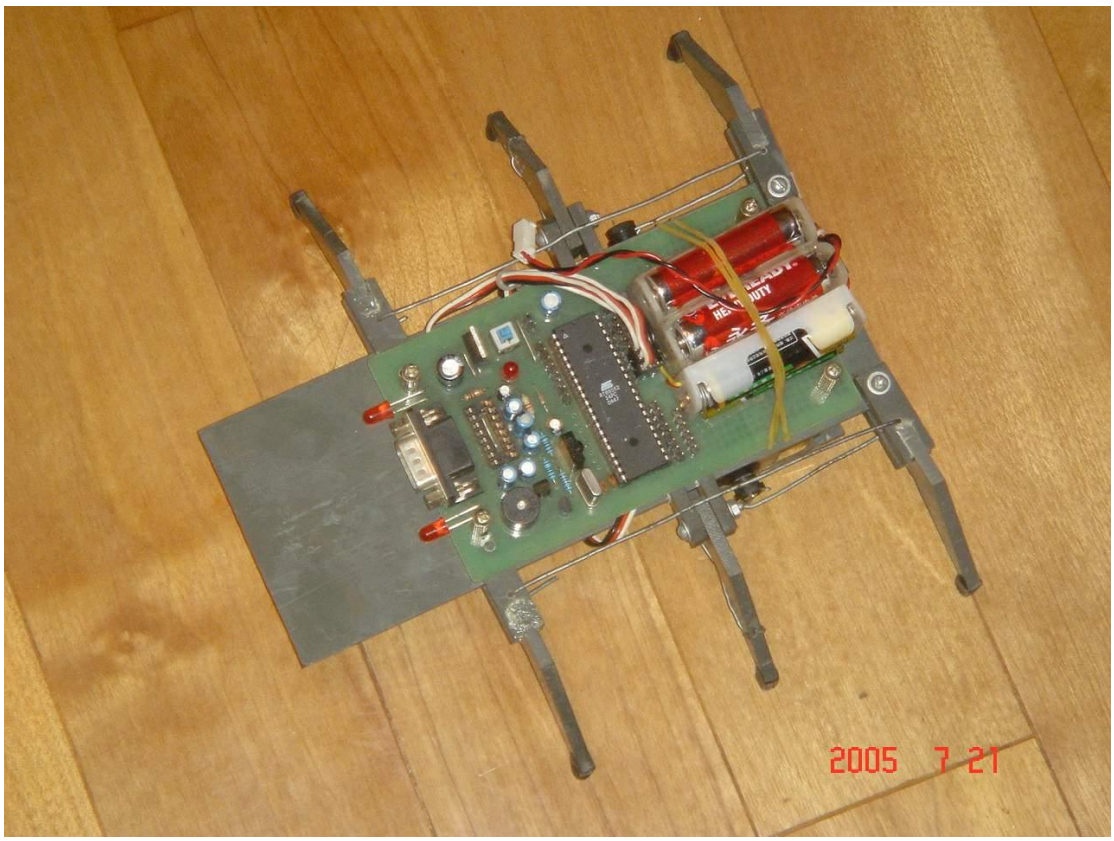
```
                MOV    R3,#10
GR6:            CALL   DJL_FOR
                DJNZ   R3,GR6
                RET
; ------------------------------------------------------------------------------------------------------------------------------------------
QD:   MOV    R3,#5                                          ；机器人行走启动子程序
Q1:   CALL   DJZ_BACK
      DJNZ   R3,Q1
      MOV    R3,#5
Q2:   CALL   DJR_FOR
      DJNZ   R3,Q2
      MOV    R3,#5
Q3:   CALL   DJL_FOR
      DJNZ   R3,Q3
      RET
; ------------------------------------------------------------------------------------------------------------------------------------------
        END
```

六、六足爬虫机器人结构设计图



红外遥控六足爬虫机器人

七、制作完成后照片



NPN 和 **PNP** 主要就是电流方向和电压正负不同，说得“专业”一点，就是“极性”问题。

NPN 是用 B→E 的电流（IB）控制 C→E 的电流（IC），E 极电位最低，且正常放大时通常 C 极电位最高，即 VC > VB > VE

PNP 是用 E→B 的电流（IB）控制 E→C 的电流（IC），E 极电位最高，且正常放大时通常 C 极电位最低，即 VC < VB < VE

总之 VB 一般都是在中间，VC 和 VE 在两边，这跟通常的 BJT 符号中的位置是一致的，你可以利用这个帮助你的形象思维和记忆。而且 BJT 的各极之间虽然不是纯电阻，但电压方向和电流方向同样是一致的，不会出现电流从低电位处流行高电位的情况。

如今流行的电路图画法，通常习惯“男上女下”，哦不对，“阳上阴下”，也就是“正电源在上负电源在下”。那 **NPN** 电路中，E 最终都是接到地板（直接或间接），C 最终都是接到天花板（直接或间接）。**PNP** 电路则相反，C 最终都是接到地板（直接或间接），E 最终都是接到天花板（直接或间接）。这也是为了满足上面的 VC 和 VE 的关系。一般的电路中，有了 **NPN** 的，你就可以按“上下对称交换”的方法得到 **PNP** 的版本。无论何时，只要满足上面的 6 个“极性”关系（4 个电流方向和 2 个电压不等式），BJT 电路就可能正常工作。当然，要保证正常工作，还必须保证这些电压、电流满足一些进一步的定量条件，即所谓“工作点”条件。

对于 **NPN** 电路：

对于共射组态，可以粗略理解为把 VE 当作“固定”参考点，通过控制 VB 来控制 VBE（VBE=VB-VE），从而控制 IB，并进一步控制 IC（从电位更高的地方流进 C 极，你也可以把 C 极看作朝上的进水的漏斗）。

对于共基组态，可以理解为把 VB 当作固定参考点，通过控制 VE 来控制 VBE（VBE=VB-VE），从而控制 IB，并进一步控制 IC。

如果所需的输出信号不是电流形式，而是电压形式，这时就在 C 极加一个电阻 RC，把 IC 变成电压 IC*RC。但为满足 VC>VE， RC 另一端不接地，而接正电源。

而且纯粹从 BJT 本身角度，而不考虑输入信号从哪里来，共射组态和共基组态其实很相似，反正都是控制 VBE，只不过一个“固定” VE，改变 VB，一个固定 VB，改变 VE。

对于共射组态，没有“固定参考点”了，可以理解为利用 VBE 随 IC 或 IE 变化较小的特性，使得不论输出电流 IE 怎么变化（当然也有个限度），VE 基本上始终跟随 VB 变化（VE=VB-VBE），VB 升高，VE 也升高，VB 降低，VE 也降低，这就是电压跟随器的名称的由来。

PNP 电路跟 **NPN** 是对称的，例如：

对于共射组态，可以粗略理解为把 VE 当作“固定”参考点，通过控制 VB 来控制 VEB（VEB=VE-VB），从而控制 IB，并进一步控制 IC（从 C 极流向电位更低的地方，你也可以把 C 极看作朝下的出水管）。

对于共基组态，可以理解为把 VB 当作固定参考点，通过控制 VE 来控制 VEB（VEB=VE-VB），从而控制 IB，并进一步控制 IC。

……

上面所有的 VE 的“固定”二字都加了引号。因为 E 点有时是串联负反馈的引入点，这时 VE 也是变化的，但这个变化是反馈信号，即由 VB 变化这个因造成的果。