第 3 讲　机器人的组装和测试

学习情境

前两讲已经学习了机器人的大脑和执行机构的使用方法，现在是时候将它们集成到一起组装成机器人了。之所以叫作基础机器人，是因为它非常简单，组装和编程都很容易。简单到何种程度，通过本讲的学习就会有直接的体会了。这里的基础机器人是一款两轮驱动的小型自主移动机器人，通过组装和测试，可以为后续几讲讲解基础机器人的导航运动和基于传感器的运动控制搭建一个实验和实践的平台。

本讲需要完成以下任务。

（1）组装机器人。

（2）测试伺服电机是否连接正确。

（3）连接并测试蜂鸣器，蜂鸣器能让你知道什么时候电池电压偏低。

（4）用调试终端控制并测试伺服电机的速度。

任务 1：组装机器人

如图 3.1 所示是基础机器人车体，接下来我们一起来看一下怎么制作一台轮式机器人。

需要的零件

① L 型 2×2 连接件 6 个
② 2×10 板件 2 个
③ 2×11 板件 2 个
④ 开槽杆件 5 个
⑤ 钢珠 1 个
⑥ 万向轮 2 个
⑦ 不锈钢螺钉 2 个
⑧ 不锈钢六角螺母 2 个
⑨ 垫圈筒 2 个
⑩ 铆钉 18 个
⑪ 十字沉头 M3×10 螺钉（黑色尼龙）2 个
⑫ 十字沉头 M3×6 螺钉（黑色尼龙）38 个
⑬ 车轮固定螺钉 2 个
⑭ M3 六角螺母（透明）24 个

⑮ 单通 M3×10+6 六角螺柱（黑色尼龙）2 个
⑯ 单通 M3×12+6 六角螺柱（黑色尼龙）2 个
⑰ 单通 M3×15+6 六角螺柱（黑色尼龙）10 个
⑱ 双通 M3×30 六角螺柱（黑色尼龙）4 个
⑲ 双通 M3×35 六角螺柱（黑色尼龙）2 个
⑳ 电池盒 1 个
㉑ 拓展盖板 1 个
㉒ 车轮 2 个
㉓ BasicDuino 微控制器 1 个
㉔ QTI 传感器（绿色）4 个
㉕ 面包板 1 个
㉖ 伺服舵机 2 个
㉗ 扎带 1 条
㉘ 杜邦线 4 条
㉙ 椭圆形金色 QC 标贴 1 个

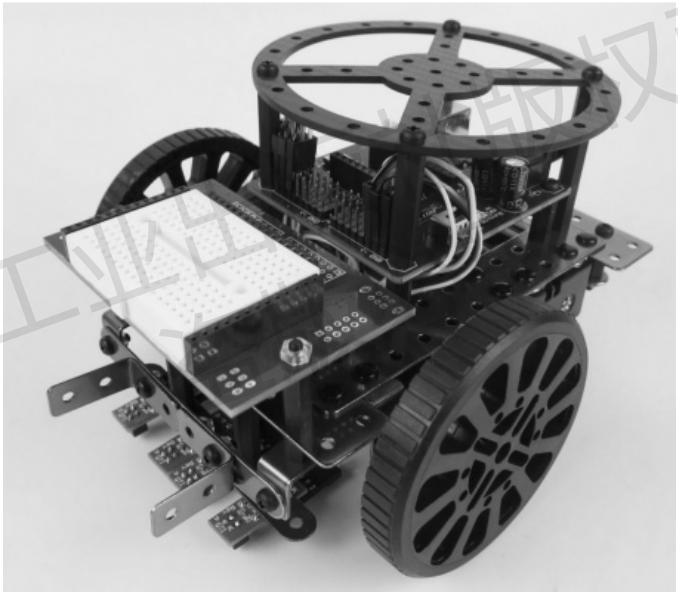


图 3.1　基础机器人车体

需要的工具

① 螺丝刀 1 把　　② 尖嘴钳 1 把

组装步骤

第一步：把铆钉安装在机器人板件上，如图 3.2 所示。

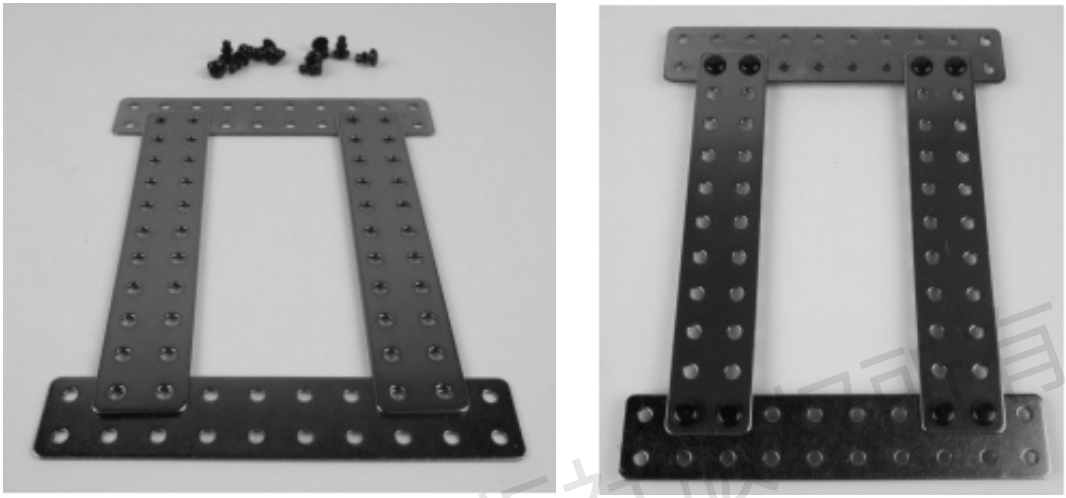


图 3.2　在板件上安装铆钉

注意：8 个铆钉要拧紧，操作时应注意双手的配合！

第二步：安装开槽杆件，如图 3.3 所示。

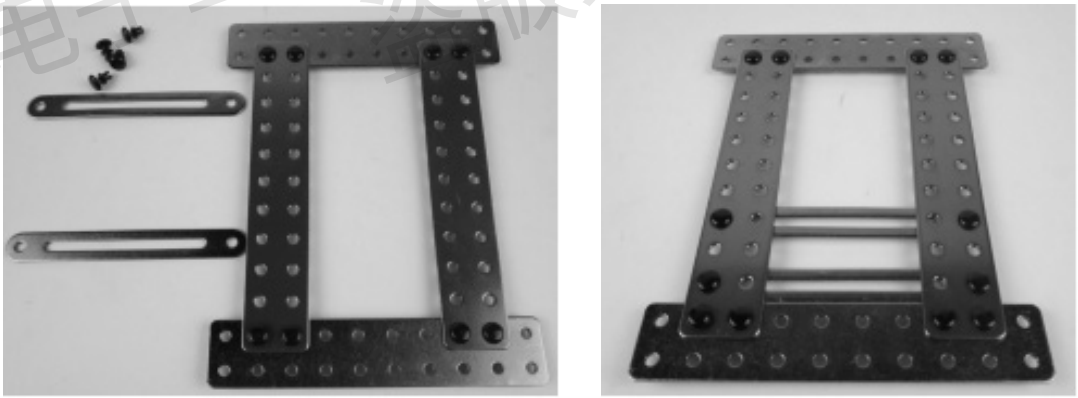


图 3.3　安装开槽杆件

注意：4 个铆钉的安装位置一定要准确！

第三步：安装螺柱，如图 3.4 所示。

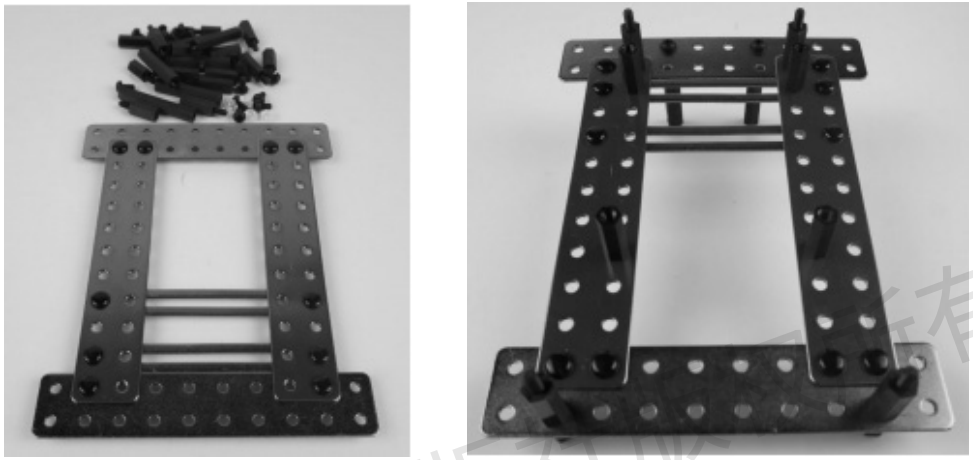


图 3.4　安装螺柱

注意：螺柱的大小和型号一定要正确！

第四步：安装伺服舵机，如图 3.5 所示。

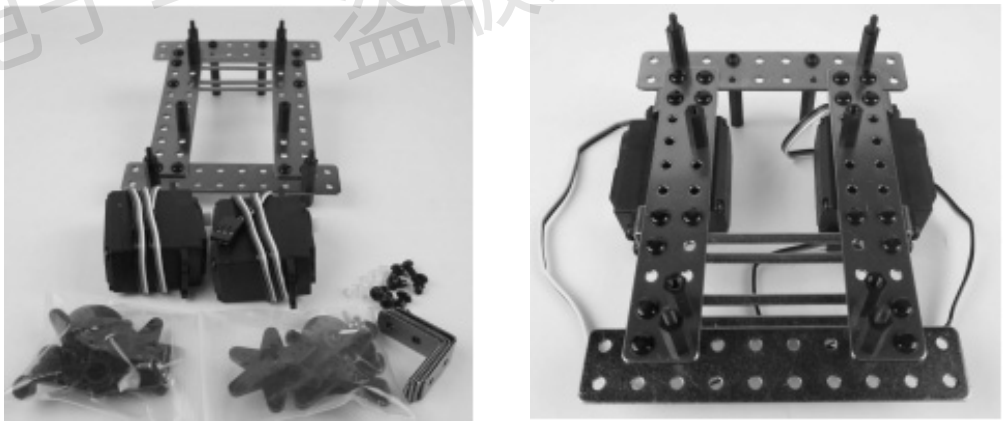


图 3.5　安装伺服舵机

注意：伺服舵机的安装位置要用杆件固定好，铆钉的位置要正确！

第五步：安装伺服电池盒，如图 3.6 所示。

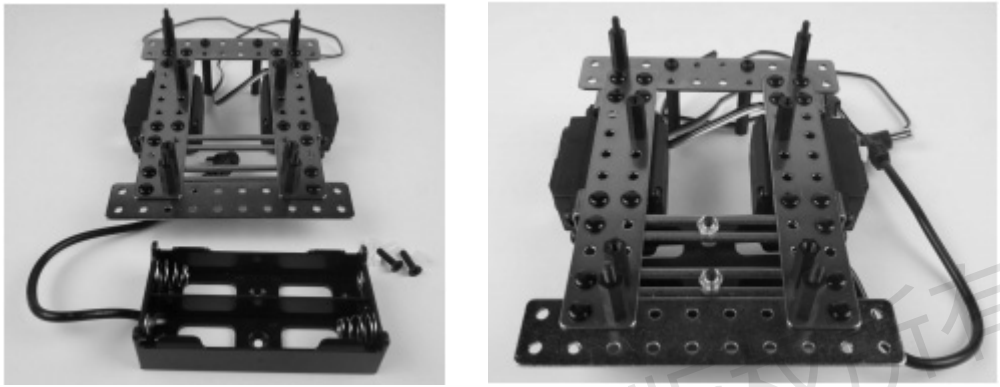


图 3.6　安装伺服电池盒

注意：电池盒要固定安装到开槽杆件中间！

第六步：安装车轮和万向轮，如图 3.7 所示。

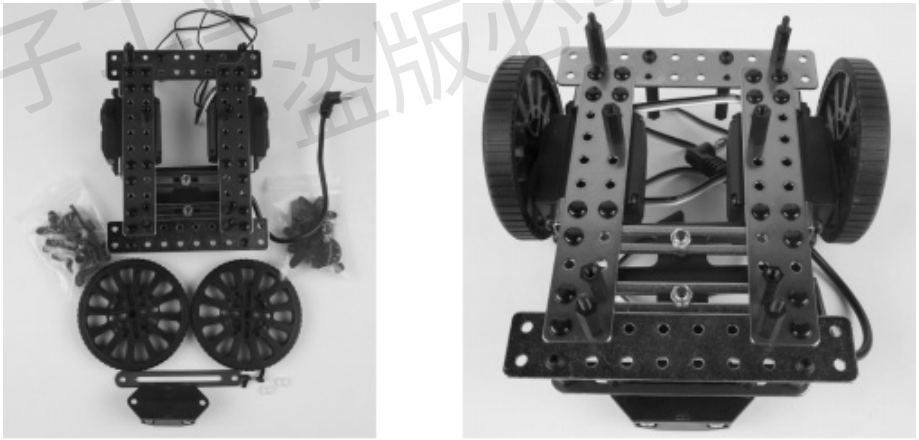


图 3.7　安装车轮和万向轮

注意：安装车轮时，将防滑带套到轮子上需要一些技巧和力量！如果防滑带太紧，可以先使劲拉伸几次，让它松软一些，然后再套在轮子上！

第七步：安装 BasicDuino 微控制器和面包板，如图 3.8 所示。

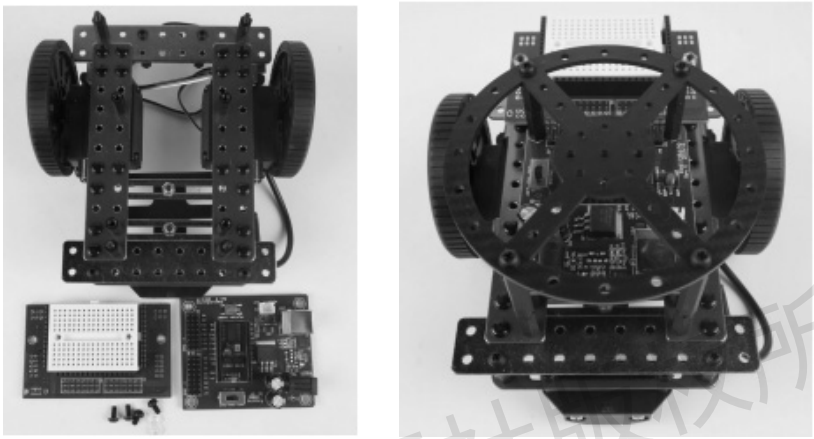


图 3.8　安装 BasicDuino 微控制器和面包板

注意：螺柱的大小和型号要准确，微控制器和面包板的位置要正确！

第八步：安装 QTI 传感器并接线，如图 3.9 所示。

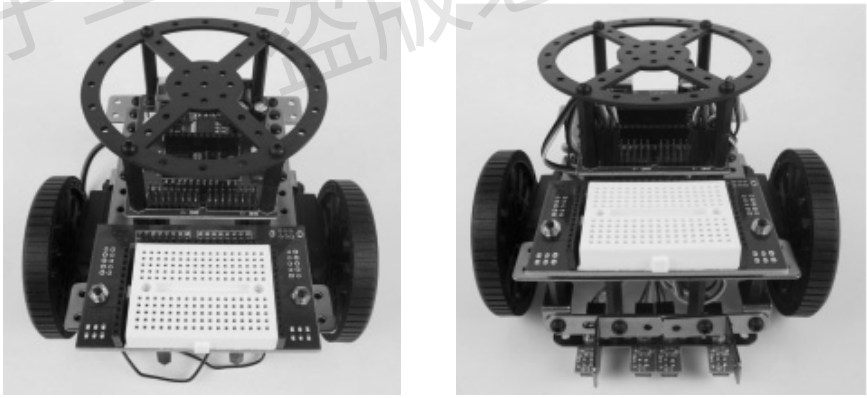


图 3.9　安装 QTI 传感器并接线

注意：QTI 传感器和固定杆件的安装位置要准确，QTI 传感器和伺服舵机的接线不能接错！

任务 2：重新测试伺服电机

机器人组装好后必须重新进行测试，以确保 BasicDuino 微控制器和伺服电机之间的电气连接正确。

测试右轮

下面的例程用来测试连接右轮的伺服电机，程序将使右轮先顺时针旋转 3s，再停止 1s，最后逆时针旋转 3s。

例程：RightServoTest.bs2

- 将机器人架起来，使右轮悬空。
- 把电池装到电池盒中。
- 将 BasicDuino 微控制器上的三挡电源开关拨到“2”位。
- 输入、保存并运行程序 RightServoTest.bs2。
- 验证右轮是否先顺时针旋转 3s，再停止 1s，最后逆时针旋转 3s。
- 如果右轮和伺服电机的运动与预期不同，则参考本例程后面的伺服电机故障排除部分。
- 如果结果正确，跳到“该你了”部分，用同样的方法测试左轮。

```
' RightServoTest.bs2
' Right servo turns clockwise three seconds, stops 1 second, then
' counterclockwise three seconds.

' {$STAMP BS2}
' {$PBASIC 2.5}

DEBUG "Program Running!"
counter VAR Word
```

```
FOR counter = 1 TO 122 ' Clockwise just under 3 seconds.
    PULSOUT 12, 650
    PAUSE 20
NEXT

FOR counter = 1 TO 40 ' Stop one second.
    PULSOUT 12, 750
    PAUSE 20
NEXT

FOR counter = 1 TO 122 ' Counterclockwise three seconds.
    PULSOUT 12, 850
    PAUSE 20
NEXT

END
```


伺服电机故障排除——一些常见的故障现象和维修方法

（1）伺服电机根本不转。

- 确定 BasicDuino 微控制器上的三挡电源开关拨到了“2”位，然后按下并释放复位键，重新运行程序。
- 参照图 2.3 仔细检查伺服电机的接线。
- 检查程序输入是否正确。

（2）右边的伺服电机不转，但是左边的伺服电机旋转。

这意味着两个伺服电机接反了，连接到 P12 端口的伺服电机应该连接到 P13 端口，而连接到 P13 端口的伺服电机应该连接到 P12 端口。

- 断开电源，拔下伺服电机插头。

- 把原来连接到 P12 端口的伺服电机连接到 P13 端口，把原来连接到 P13 端口的伺服电机连接到 P12 端口。
- 打开电源，重新运行程序 RightServoTest.bs2。

（3）轮子不能完全停下来，而是缓慢地旋转。

这意味着伺服电机可能没有正确调零。可以调节程序让伺服电机停止，即通过修改 PULSEOUT 12,750 语句的参数 750 让伺服电机停止。

- 如果轮子缓慢地逆时针旋转，则换一个比 750 小一点的数。
- 如果轮子缓慢地顺时针旋转，则换一个比 750 大一点的数。
- 如果在 740～760 中找到了一个数能让伺服电机完全停止，则用这个数代替程序中所有 PULSEOUT 12,750 语句中的 750。

（4）轮子在顺时针旋转和逆时针旋转之间不停止。

车轮可能快速地朝一个方向旋转 3s，然后向另一个方向旋转 4s；可能快速地旋转 3s，然后慢速地旋转 1s，之后又快速地旋转 3s；还可能快速地朝一个方向旋转 7s。不管怎样，都说明伺服电机的电位器失调。

- 拆除轮子，取下伺服电机。
- 重新执行第 2 讲中的任务 2，完成伺服电机调零。

该你了——测试左轮

现在在左轮上做同样的测试，更改程序 RightServoTest.bs2，发送 PULSOUT 指令到连接 P13 端口的伺服电机，而不是连接 P12 端口的伺服电机，即将 3 条语句中的“PULSOUT 12”修改为“PULSOUT 13”。

- 将程序 RightServoTest.bs2 另存为 LeftServoTest.bs2。
- 更改 3 条语句中的“PULSOUT 12”为“PULSOUT 13”。
- 保存并运行程序。
- 验证左轮是否先顺时针旋转 3s，再停止 1s，最后逆时针旋转 3s。
- 如果左轮和伺服电机的运动与预期不同，则参考前面的伺服电机故障排除部分。
- 如果左轮和伺服电机的运动与预期相同，则说明机器人已调试好，可以准备执行下面的任务了。

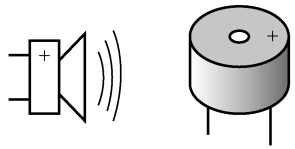
任务3：开始/复位指示电路和编程

当电源电压低于设备正常工作所需的电压时，叫作欠压。发生欠压时，BasicDuino微控制器可以使其处理器和内存芯片处于休眠状态来进行自我保护，直到电源电压恢复到正常水平为止。当BasicDuino微控制器的电源电压Vin低于5.2V时，BasicDuino微控制器内部的电压整流输出将低于4.3V，BasicDuino微控制器上的欠压检测电路就会检测到欠压已经发生，马上使处理器和程序内存芯片进入休眠状态。当电源电压回升到5.2V以上时，BasicDuino微控制器又开始运行，但不是从程序中断的地方运行，而是从头开始重新运行，就像拔掉电源插头又插上或按下BasicDuino微控制器上的复位键后又释放一样。

当机器人的电池电压过低时，欠压会使程序重启，这将导致机器人行为混乱。一种可能的情况是：机器人正在按程序规定的路线行走，突然它好像迷路了一样不再按照原来规定的路线行走。另外一种可能的情况是：机器人程序不断地反复重启，机器人不再行走。

为了防止这种现象的发生，可以为机器人编写一个重新开始指示程序作为机器人的诊断工具。一种指示程序重启的方法是在所有机器人程序的开始处包含一个不会错过的信号指示，这个信号在每次打开电源或每次电压过低导致程序复位时都会产生。一种有效表明程序重启的方法是扬声器，它在程序每次从头开始运行或重启时均会发出声音。

其实，所有的自动化设备都有这个功能，就像人们每天使用的台式计算机，在每次开启或复位时都会听到“滴”的一声。本任务就是学习如何为机器人设计和实现这个功能。首先介绍压电扬声器，它可以产生音调。该扬声器能从BasicDuino微控制器中接收不同频率的高低信号，然后产生不同的音调。如图3.10所示是压电扬声器的电气符号和零件图。下面通过编程实现当BasicDuino微控制器重启时扬声器可以发出声音。



（a）电气符号　　（b）零件图

图3.10　压电扬声器

部件清单

- 已经过测试的机器人。
- 扬声器 1 个。
- 连接线若干。

搭建开始/复位指示电路

如图 3.11 所示是扬声器报警指示电路，而图 3.12 是其实际接线电路。

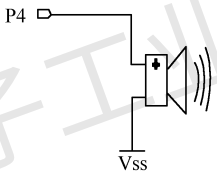


图 3.11　扬声器报警指示电路

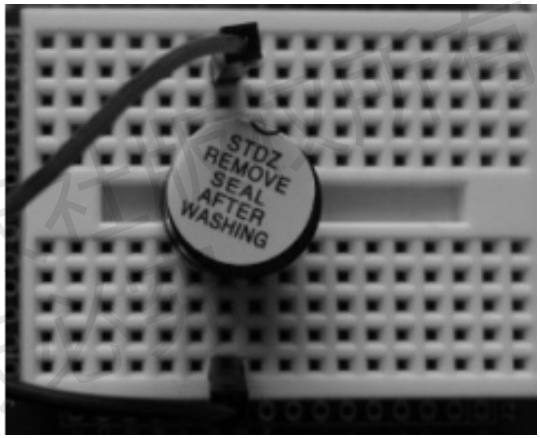


图 3.12　扬声器报警指示实际接线电路

按照图 3.11 和图 3.12 在机器人面包板上搭建开始/复位指示电路。

对开始/复位指示电路编程

下面的例程用于测试扬声器。程序采用 FREQOUT 指令给扬声器发送精确定时的高低电平信号。FREQOUT 指令的语法格式如下：

```
FREQOUT Pin, Duration, Freq1 {,Freq2}
```

下面的 FREQOUT 指令将用于下面的例程。

```
FREQOUT 4, 2000, 3000
```

参数 Pin 的值是 4，这意味着高低电平信号将被送至 P4 端口；参数 Duration 的值为 2000，代表着高低电平信号持续的时间是 2000，即 2000ms（2s）；参数 Freq1 代表信号的频率，Freq1 的值为 3000，代表着将产生一个 3000Hz 频率的音调。

例程：StartResetIndicator.bs2

该例程在开始执行时让扬声器发出声音，然后每半秒发一个 DEBUG 信息。因为信息位于 DO…LOOP 循环之间，所以这些信息将一直持续显示。如果程序运行到 DO…LOOP 循环之间时电源中断，则程序将从头开始执行。当程序重新开始时，将再一次发出声音。你可以模仿电源欠压时的情景：按下并释放 BasicDuino 微控制器上的复位键或断开再接通 BasicDuino 微控制器上的电源插头。

- 重新接通 BasicDuino 微控制器上的电源。
- 输入、保存并运行程序 StartResetIndicator.bs2。
- 验证在“Waiting for reset…”信息显示在调试终端之前，扬声器是否发出清晰且持续 2s 的响声。
- 如果没有听到声音，则检查接线和程序代码，直到听到扬声器发出清晰的声音为止。
- 如果听到声音，则按下并释放 BasicDuino 微控制器上的复位键，模拟电源欠压的情况，验证扬声器是否在每一次复位之后都能发出清晰的声音。
- 断开再接通电源，再次验证扬声器是否能发出声音。

```
' StartResetIndicator.bs2
' Test the piezospeaker circuit.
' {$STAMP BS2}                    ' Stamp directive.
' {$PBASIC 2.5}                   ' PBASIC directive.

DEBUG CLS, "Beep!!!"              ' Display while speaker beeps.
FREQOUT   4, 2000, 3000           ' Signal program start/reset.
```

```
DO                                  ' DO...LOOP
    DEBUG CR, "Waiting for reset ..."   ' Display a message
    PAUSE 500                       ' every 0.5 seconds
LOOP                                ' until hardware reset.
```

程序 StartResetIndicator.bs2 的工作原理

程序 StartResetIndicator.bs2 一开始显示信息“Beep!!!!”，在信息显示完后，FREQOUT 指令立刻使扬声器发出 3000Hz 的声音并持续 2s。由于程序执行得很快，以至于信息显示和扬声器发出声音好像同时发生一样。

扬声器响过后，程序进入 DO…LOOP 循环，一遍遍地显示相同的信息“Waiting for reset…”。每次 BasicDuino 微控制器上的复位键被按下再释放或电源被断开再接通时，程序均重新开始显示“Beep!!!!”信息并发出 3000Hz 的声音。

该你了——将程序 StartResetIndicator.bs2 加到另一个程序中

上述指示程序中的两行代码可以加入到本讲之前的每个例程的开始处。你可以把它当作机器人程序的“初始化过程”或“引导过程”的一部分。

所谓初始化过程是指在一个设备或程序启动时所必须执行的所有指令，通常包括一些变量的赋值和发出报警的声音等，对于复杂的设备，则需要自检和标定。

任务 4：用调试终端测试速度

本任务要制作一个速度和脉宽的关系曲线图，如图 3.13 所示的调试终端窗口能帮助你加快作图过程。可以通过该窗口的传送窗格（Transmit Windowpane）给 BasicDuino 微控制器发送信息，通过发送信息告诉 BasicDuino 微控制器发送给伺服电机的脉宽是多少，然后测量在不同脉宽下伺服电机的运行速度。

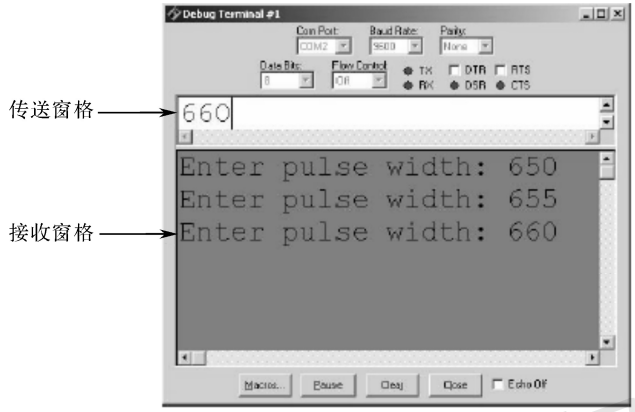


图 3.13　调试终端窗口

使用 DEBUGIN 指令

到目前为止，你应该已经熟悉了 DEBUG 指令，并且知道了它是如何让 BasicDuino 微控制器发送信息给调试终端窗口的。显示信息的地方叫作接收窗格，用于显示从 BasicDuino 微控制器接收到的信息。调试终端还有一个传送窗格，它允许在程序运行时发送信息给 BasicDuino 微控制器。可以利用 DEBUGIN 指令使 BasicDuino 微控制器接收输入到传送窗格中的值，并将其存储到一个或几个变量中。

DEBUGIN 指令将输入到传送窗格中的值存入一个变量中。在下面的例程中，字变量 pulseWidth 将被用来存储 DEBUGIN 指令接收到的值。

```
pulseWidth VAR Word
```

现在用 DEBUGIN 指令来获取输入到传送窗格中的十进制数值，并将其存储到变量 pulseWidth 中。

```
DEBUGIN DEC pulseWidth
```

后面的程序可以使用此值，在本任务中该值用在 PULSOUT 指令的参数 Duration 中。

```
PULSOUT 12, pulseWidth
```

例程: TestServoSpeed.bs2

```
' TestServoSpeed.bs2
' Enter pulse width, then count revolutions of the wheel.
' The wheel will run for 6 seconds
' Multiply by 10 to get revolutions per minute (r/min).
'{$STAMP BS2}
'{$PBASIC 2.5}

counter VAR Word
pulseWidth VAR Word
pulseWidthComp VAR Word

FREQOUT 4, 2000, 3000 ' Signal program start/reset.

DO
    DEBUG "Enter pulse width: "
    DEBUGIN DEC pulseWidth
    pulseWidthComp = 1500-pulseWidth
    FOR counter = 1 TO 244
        PULSOUT 12, pulseWidth
        PULSOUT 13, pulseWidthComp
        PAUSE 20
    NEXT
LOOP
```

该程序允许你通过调试终端的传送窗格给 PULSOUT 指令的参数 Duration 赋值。

- 将机器人架起来，使其轮子不能着地。
- 输入、保存并运行程序 TestServoSpeed.bs2。
- 用鼠标指向调试终端窗口的传送窗格，激活光标以便输入。
- 输入 650，然后按回车键。
- 验证伺服电机是否全速顺时针旋转 6s。

当伺服电机完成旋转后，将提示你输入另一个值。

- 输入 850，然后按回车键。
- 验证伺服电机是否全速逆时针旋转 6s。

试着测试轮子在脉宽为 650～850 时的旋转速度，以 r/min（每分钟转动的转数）为单位。下面给出具体的测试过程。

- 在轮子上贴一个标记，这样就可以知道它在 6s 内旋转了几转。
- 使用调试终端测量轮子在下述脉宽参数下转过的转数：650，660，670，680，690，700，710，720，730，740，750，760，770，780，790，800，810，820，830，840，850。
- 对于每一个脉宽，转动的转数乘以 10 即得转速（单位是 r/min）。例如，如果轮子转了 3.65 转，那么转速为 36.5r/min。
- 请分析说明如何通过脉宽来控制伺服电机的旋转速度。

程序 TestServoSpeed.bs2 是如何工作的

首先声明 3 个变量，分别是 FOR…NEXT 循环的 counter 变量、DEBUGIN 指令和第一个 PULSOUT 指令的 pulseWidth 变量以及第二个 PULSOUT 指令的 pulseWidthComp 变量。

```
counter VAR Word
pulseWidth VAR Word
pulseWidthComp VAR Word
```

FREQOUT 指令用来表示程序已经开始执行。

```
FREQOUT    4,2000,3000
```

程序的剩余代码都在 DO…LOOP 循环中，因此它会一遍又一遍地执行下去。每次在调试终端操作者（就是你）输入脉宽后，DEBUGIN 指令都会将此值存储在变量 pulseWidth 中。

```
DEBUG "Enter pulse width: "
DEBUGIN DEC pulseWidth
```

要使测量更精确，必须使用两个 PULSOUT 指令，一个脉宽参数小于 750，另一个脉宽参数大于 750，两个脉宽参数的和是 1500，这就确保在每次循环执行时两个 PULSOUT 指令所用的时间之和是相同的，即无论 PULSOUT 指令的脉宽参数是多少，FOR…NEXT 循环都要花费同样的时间去执行，这可以使后面的转速测量更加准确。

下面的指令是根据所输入的脉宽参数计算另一个脉宽参数，使两个脉宽参数的和为 1500。如果输入的值是 650，则 pulseWidthComp 的值是 850；如果输入的值是 850，则 pulseWidthComp 的值是 650；如果输入的值是 700，则 pulseWidthComp 的值是 800。总之，它们加起来的和一定是 1500。

```
pulseWidthComp = 1500 - pulseWidth
```

一个运行时间为 6s 的 FOR…NEXT 循环首先发送 pulseWidth 的值给右轮的伺服电机（P12），然后发送 pulseWidthComp 的值给左轮的伺服电机（P13），使两轮的旋转方向相反。

```
FOR counter = 1 TO 244
    PULSOUT 12, pulseWidth
    PULSOUT 13, pulseWidthComp
    PAUSE 20
NEXT
```

该你了——脉宽与转速关系曲线图

如图 3.14 所示是一个连续旋转伺服电机控制脉宽和转速的关系曲线。横坐标代表脉宽，

单位是 ms；纵坐标代表伺服电机的旋转速度，单位是 r/min（rpm）。图中，顺时针旋转是负值，逆时针旋转是正值。这个特定伺服电机的转速范围为-48～48r/min，对应的脉宽范围为 1.3～1.7ms。

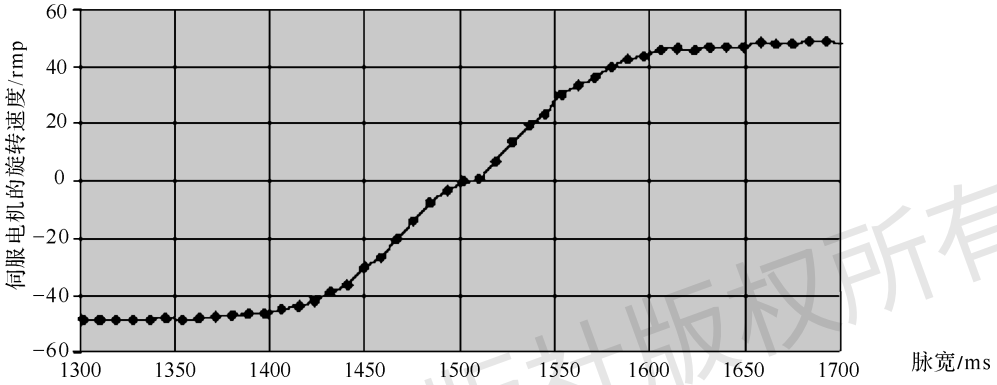


图 3.14　连续旋转伺服电机控制脉宽和转速的关系曲线

可以用表 3-1 来记录关系曲线的数据。注意：在例程中是用输入值来直接控制右轮转速的，而左轮转速则是用计算值来控制且左轮旋转的方向与右轮相反。

表 3-1　脉宽与转速的关系

脉宽/ms	转速/rmp	脉宽/ms	转速/rmp	脉宽/ms	转速/rmp	脉宽/ms	转速/rmp
1.300		1.400		1.500		1.600	
1.310		1.410		1.510		1.610	
1.320		1.420		1.520		1.620	
1.330		1.430		1.530		1.630	
1.340		1.440		1.540		1.640	
1.350		1.450		1.550		1.650	
1.360		1.460		1.560		1.660	
1.370		1.470		1.570		1.670	

续表

脉宽/ms	转速/rmp	脉宽/ms	转速/rmp	脉宽/ms	转速/rmp	脉宽/ms	转速/rmp
1.380		1.480		1.580		1.680	
1.390		1.490		1.590		1.690	
						1.700	

注意：PULSOUT 指令的参数 Duration 是以 2μs 为单位的。PULSOUT 12, 650 表示发送脉宽为 1.3ms 的脉冲给 P12 端口；PULSOUT 12, 655 表示发送脉宽为 1.31ms 的脉冲给 P12 端口；PULSOUT 12, 660 表示发送脉宽为 1.32ms 的脉冲给 P12 端口，依此类推。

Duration=650×2μs=650×0.000002s=0.00130s=1.3ms

Duration=655×2μs=655×0.000002s=0.00131s=1.31ms

Duration=660×2μs=660×0.000002s=0.00132s=1.32ms

- 给右轮做标记使你有一个参考点。
- 运行程序 TestServoSpeed.bs2。
- 单击调试终端窗口的传送窗格。
- 输入 650，然后按回车键。
- 测量轮子转动的转数。

因为伺服电机的转动时间为 6s，故可以用转数乘以 10 得到转速（单位是 r/min）。

- 把转数乘以 10 的值记录在表 3-1 的 1.300ms 之后。
- 输入 655，然后按回车键。
- 测量轮子转动的转数。
- 把转数乘以 10 的值记录在表 3-1 的 1.310ms 之后。
- 增加 durations 的值，直到 850。
- 用电子表格、计算机或图表来描绘上述关系曲线。
- 对另一个伺服电机重复上述过程。

可以用左轮重复上述测量过程。更改 PULSOUT 指令，将 pulseWidth 的值发送给 P13，将 pulseWidthComp 的值发送给 P12。

工程素质和技能归纳

- 能够完成机器人的机械组装。
- 能够完成伺服电机的重新测试和子系统测试。
- 掌握开始/复位指示电路工作原理和 FREQOUT 指令的使用方法。
- 掌握运行时向 BasicDuino 微控制器发送数据的方法及 DEBUGIN 指令的使用方法。
- 会测试伺服电机控制脉宽和转速的关系曲线。
- 能够完成脉宽与转速关系曲线的作图。