SRV-1 机器人使用手册

—适用于 SRV-1/SRV-A/UCR-3 系列机器人产品

版本： 2013-03-18

内容目录

一、SRV-1 机器人介绍

SRV-1 是一款多功能履带式机器人，基于开源软件平台，可通过 WIFI 进行无线操控，并可实时采集现场视频，同时还支持自主运行以及集群管理模式。其设计可广泛应用于各个相关行业的科研、教学以及新产品原型开发等领域当中。

SRV-1 无线操控机器人由 SRV-1 Blackfin 摄像头主板（使用 Analog Devices 公司的 500MHz Blackfin BF537 处理器），CMOS 摄像头（分辨率由 160x280 到 1280x1024 像素），激光照射点以及 Lantronix Matchport 802.11b/g 无线通讯模块组成，然后整合到一个双履带可移动的机体之上。

作为一个可以远程控制的网络摄像头或者一个具有自主导航能力的机器人，SRV-1 可以在 Blackfin 主板的固件之中运行 C 语言解释器解释执行 C 程序，或者在 Windows，MacOS/X 和 Linux 等系统上，运行基于 Python 或 Java 语言编写的控制台软件，进行远程管理。使用 Java 控制台的软件，借助 SRV-1 内建的网络服务器，可以实现通过网络浏览器，在世界上任何地方监控并控制 SRV-1，并可以根据需求保存视频。SRV-1 内置的固件还可以支持 RoboRealm 之类的第三方软件。

硬件参数：

- 处理器：500MHz 的 ADI Blackfin BF537(1000 mips), 32MB SDRAM, 4MB SPI Flash, JTAG
- 摄像头：Omnivision OV7725, 30 万像素(最大分辨率 640x480) ，图像速度每秒最多可达 60 帧，视角范围为 90 度
- 串口无线通讯模块：Lantronix Matchport 802.11b/g WiFi，支持外接天线加强无线信号
- 传感器：2 个激光点
- 驱动：双路直流齿轮电机（四个电机，100：1 齿轮减速比）驱动双履带
- 速度：20~40 厘米每秒（大约 1 英尺每秒或 0.5 公里每小时）
- 框架：铝制机械
- 规格：12cm 长 x 10cm 宽 x 8cm 高
- 重量：约 380 克
- 电源：7.2V 2Ah 锂电池 使用时间大约 4 小时
- 充电器：100~240 交流电 50/60Hz

产品特点：

- 遵循 GPL 开放全部软件源码及硬件设计
- 机器人具有执行自主运行程序的能力
- 第三方软件的支持，如 RoboRealm
- 可以通过软件控制台或网络浏览器实现对机器人的远程遥控，具备一定的监控能力
- 系统内建网络服务器并支持视频存档
- 机器人可以解释执行储存在 FLASH 中或内存中的 C 程序
- 无线遥控范围：室内 20 米室外 200 米
- 机器人可以通过终端控制台进行快捷的控制与调试
- 支持二次开发，适合计算机，视觉图像处理，嵌入式，自动化控制等高校专业做科研教学或课题产品的原型设计及开发
- 支持 GNU/Linux 操作系统也同时支持直接在 DSP 上通过 GNU bfin-elf-工具链开发 bare-metal 应用

软件功能：

- 基本的图像处理能力：histogram, pixel sampling, frame difference, blob, scan, count, find
- 动态 JPEG 图像压缩处理能力
- 可支持的摄像头分辨率：1280x1024 (OV9655 适用), 640x480, 320x256, 160x128
- 提供了马达控制接口：PWM (H-bridge) and PPM (servo)
- 内置的类 c 解释器，支持用户编写脚本让其自主运行
- RTC 时钟精确到毫秒，内部定时器精确到 10 纳秒
- 支持扩展外接具有 I2C 或 SPI 通讯能力的设备
- 支持 XMODEM 文件传输协议
- 已支持的一些外部传感器： Maxbotics 超声波测距或者 Sharp 红外测距传感器；Locosys 或者 uBlox5 GPS 定位传感器； Honeywell HMC6352 或者 HMC5843 罗盘，ST LIS3LV02DQ 倾斜传感器, Analog Devices AD7998 8-channel A/D（多为第三方支持）
- 支持 GNU/Linux 操作系统（包括 u-boot/uClinux）

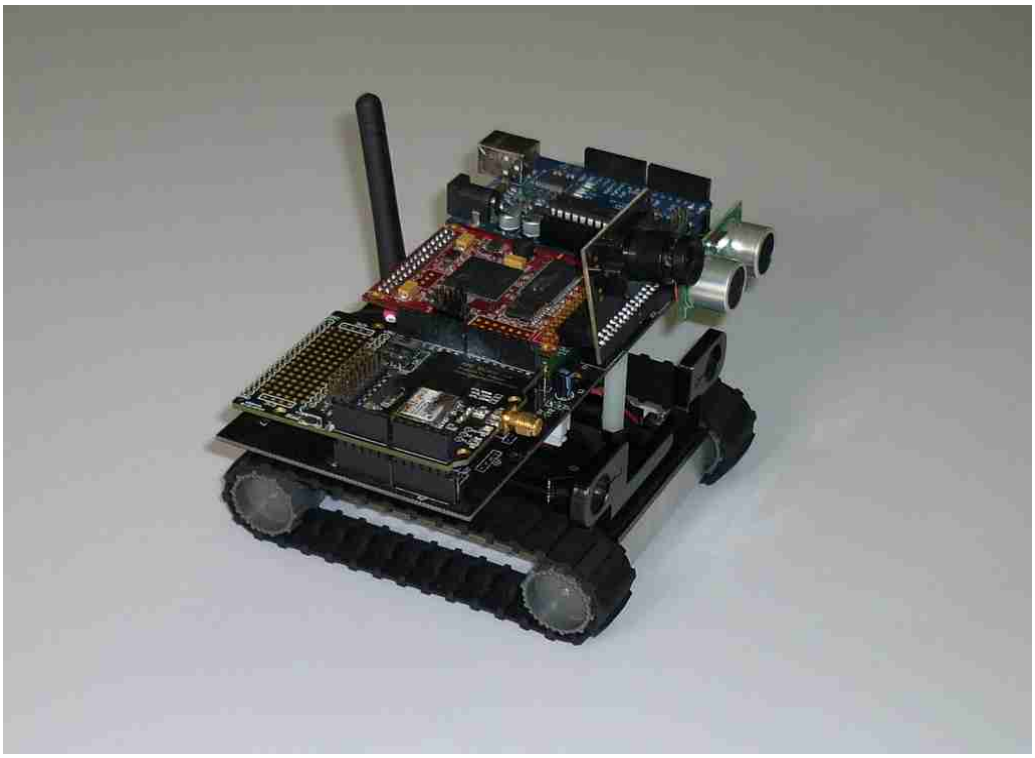
软件特点：

- 机器人固件更新方式简单，可支持通过 web 页面直接更新，当然也支持命令行方式更新（如故障恢复）
- 丰富的客户端控制程序，有基于 Java, Python, Delphi 等各种版本提供选择
- 丰富的控制协议支持，详见后面的机器人控制协议
- 通过内置的微型 HTTP 服务模块实现了对机器人操控的平台无关性
- 通过内嵌的 C 解释器程序实现机器人的自主运行能力
- 提供 Android / iOS / NDSL 等流行的移动设备操作系统客户端软件支持
- 强劲的邮件列表技术探讨支持

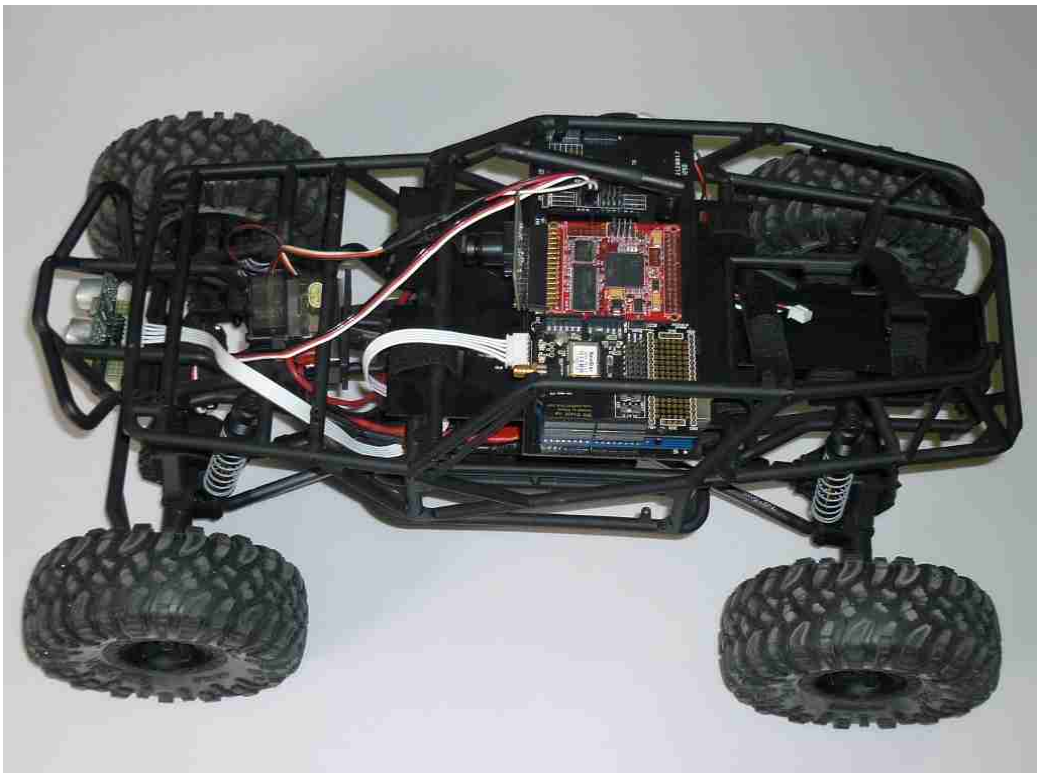
二、基于 SRV-1 机器人衍生版 SRV-A 和 UCR-3 介绍

SRV-A 是一款基于 SRV-1 机器人的扩展版，除了继承 SRV-1 的全部优点外，还集成了 Arduino 的便捷开发优势，可以帮助用户很方便的添加各种外部传感器。默认搭配的传感器如下：

- GPS 传感器，用为定位
- 陀螺仪及加速度传感器，用于姿态反馈
- 超声波传感器，用于测距及避障，精确测试范围 2cm ~ 450cm



UCR-3 是一款越野攀爬机器人，是针对 SRV-1 行驶局限性所做的一个改进版，同时搭载了与 SRV-A 一样的各种传感器。



硬件特点：

- 框架：美国进口底盘
- 驱动：大功率直流电机，带转向舵机
- 规格：45cm 长 x 28cm 宽 x 24cm 高
- 重量：约 2.5 千克
- 电源：7.4V 5Ah 10C 锂电池（10C，即最高可达普通电池的 10 倍放电量） 使用时间约 2.5 小时
- 充电器：100-240 交流电 50/60Hz 高倍率平衡充电器

进口底盘特点：

- 大功率电调
- 液压避震
- 独立式悬挂
- 超强越野攀爬能力
- 独立舵机转向

三、开始之前

对于新开箱的产品，固件与 Matchport 无线模块都是已经配置好的，所以可以直接通过客户端连接操控。

Matchport 无线模块的默认配置是 Adhoc 无线网络，SSID 为"SRV-1"，默认 IP 是 169.254.0.10。给 SRV-1 机器人上电后，通过 PC Host 端的无线网卡连接"SRV-1"那个 AP。

"SRV-1"本身是具有 DHCP 功能的，但效果不怎么理想，尤其是第一次连接 DHCP 过程会非常缓慢，所以还是推荐用户为 PC Host 配置一个静态 IP，如：

- IP 地址 -> 169.254.0.88
- 网关 -> 169.254.0.1
- 掩码 -> 255.255.0.0
- DNS -> 169.254.0.1

连接成功后应该就可以 ping 通 169.254.0.10 了。

Matchport 无线模块提供两种无线通讯模式：Adhoc 模式和 Infrastructure 模式。简单理解，Adhoc 模式不依赖公共网络，可以随时随地自由组建；Infrastructure 模式可以理解为像其它终端一样接入公共网络(如公司,家庭的 WIFI)。

注意下小车的开关按钮，有三个档位：开(on)/关(of)/充电(charge)，所以如果需要充电，需要将开关调换到 charge 档位。

四、移动客户端连接

SRV-1 支持的移动客户端有：

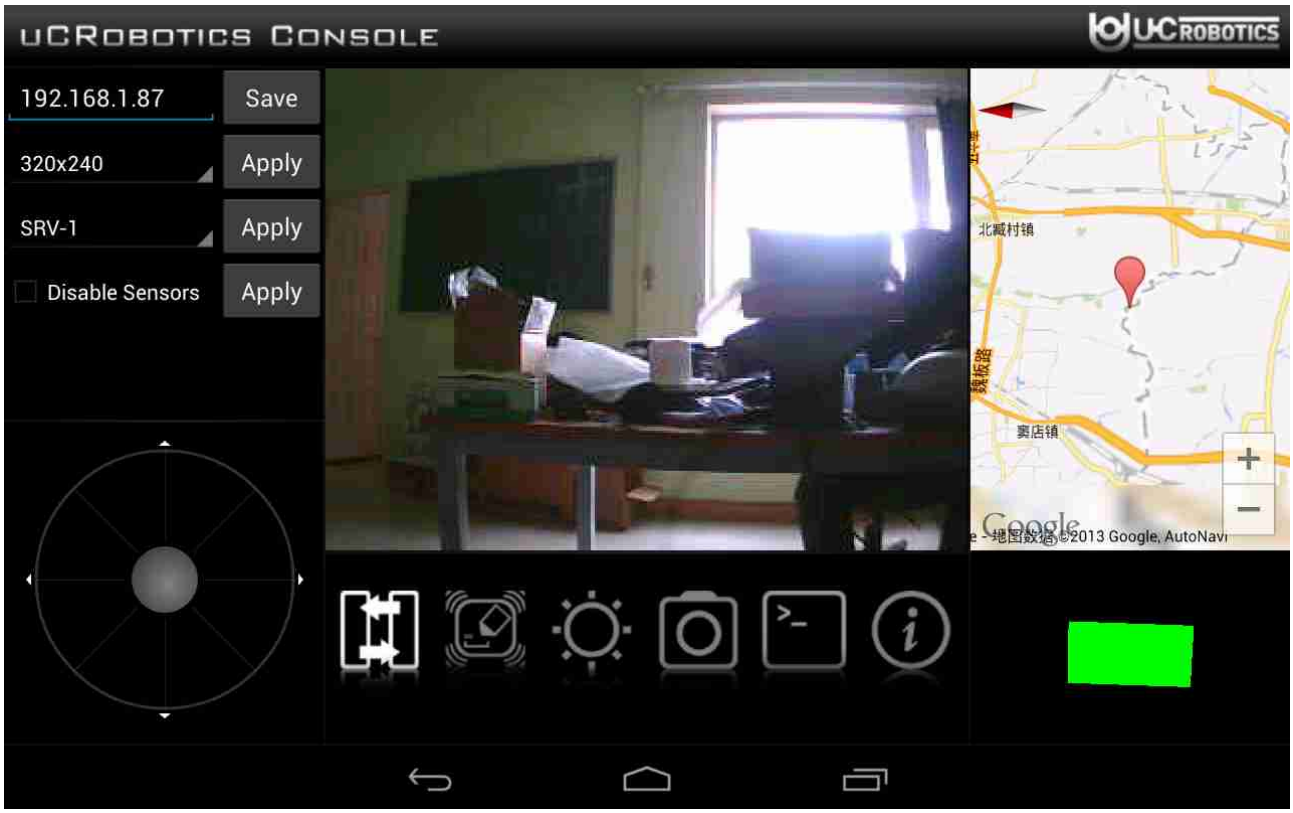
- Android 设备
- iOS 设备
- NDSL 设备

因为 Android 客户端是功能最全的客户端，所以优先推荐使用。

尤其是 Android 客户端，可以完美支持 SRV-1 / SRV-A / UCR-3 这几种机器人，除了在控制体验方便做了很大的

提升，同时对搭载的各种传感器进行了功能展示。

因为要展示的内容较多，所以该客户端是以 Nexus 7 为蓝本进行开发的。因此运行该客户端所需要的 Android 设备分辨率最好能达到 1280x800。



如图所示：

1. 左边第一个输入框为机器人 IP 地址设置选项
2. 左边第二个下拉框为分辨率选项，默认 320x240，当然也可以选项 640x480，就是延迟感比较强
3. 左边第三个下拉框为机器人类型选项，可支持 SRV-1 和 UCR-3；（注意：这里的 SRV-1 默认状态下其实是指 SRV-A，如果是经典版 SRV-1 控制，需要选中第四个单选框，即禁止掉各个传感器的请求）
4. 左边第四个单选框，选中状态表示禁止掉各个传感器的数据请求，在控制经典版 SRV-1 时必需选中
5. 下方的大圆为模拟摇杆，用于控制机器人的行动。（推动中心的半透明圆盘即可）
6. 中间为视频显示区域
7. 右边为 Google Map 展示区，通过 GPS 传感器获取当前所在位置的地理坐标，然后标示在地图上
8. 右下方为姿态反馈区，通过解析陀螺仪和加速度传感器所采集的数据将机器人的姿态通过 3D 绘图技术实时绘制出来
9. 中间底部几个功能按钮分别为：
    - 连接/断开控制按钮
    - 测距按钮

- 激光控制按钮（仅针对经典版 SRV-1 有效，SRV-A 和 UCR-3 对应是 LED）
- 视频图像保存开始/停止按钮（会将视频的每一帧图像保存到/sdcard/ucrobotics/xxxx/，其中 xxxx 为当前日期+时间的组合，如 20130318185048）
- 简单终端按钮，由于 SRV-1 固件功能众多，无法通过应用一一展示，所以也可以通过终端发送指令到机器人，如查看当前固件版本，发送'V'即可。其它更多指令，可参阅后面的附录：SRV-1 机器人控制协议

保存视频图像结束后，可将图像目录（如 20130318185048）拷贝到 PC 上，然后通过如下命令转换成视频文件：

```
    $ cd 20110729185048/
    $ mencoder mf://*.jpg -mf w=320:h=240:fps=24:type=jpg -ovc lavc -lavcopts vcodec=mpeg4:mbd=2:trell
-oac copy -o output.avi
```

关于上述命令参数的说明：

- mencoder 一行表示将当前目录下的所有.jpg 图片转换成一个.avi 视频
- w=320:h=240 - 图像的尺寸是 320x240，根据实际情况可更改
- fps=24:type=jpg - 每秒 24 帧，文件的类型是 jpg，根据实际情况可更改
- -ovc - 指定视频编码
- -oac - 指定音频编码，此处 copy，表示不做任何更改，这里也就是没有音频了

Android 客户端可通过 GooglePlay 进行下载。

关于 iOS 设备的客户端是一个简单版本，可通过 AppStore 进行下载。

关于 NDSL 客户端，由于 NDSL 硬件约束，体验不是很好，有需要者可与 uCRobotics 联系。

下面是 NDSL 连接控制时的照片：

注意：每次只能有一个终端连接到机器人，如果连接的终端没有断开，其它中断是连接不了的！

五、PC 客户端连接

SRV-1 的客户端有很多，在源码包里有一个 Console 子目录，是一个基于 Java 开发的控制端软件。所以运行之前需要确保已经安装了 Java 环境。

无论是 Windows 还是 Linux 环境，打开（虚拟）终端，输入”java”并回车，要是看到一大串输出信息表明已经安装了，如果没有，可参考以下步骤进行安装：

对于 Windows 用户：

1. 下载 Java sdk 1.6

关于 Java sdk 与 Java jre 区别： Java sdk 是 Java 程序开发环境，包含有 jre，可编译可运行 Java 程序；Java jre 是 Java 程序运行环境，不包含编译工具，如 javac 命令。

可以从 http://www.ucrobotics.com/index.php/resource 这里下载：
jdk-6u10-rc2-bin-b32-windows-i586-p-12_sep_2008.exe（或者从这里下载：aben328.iteye.com/blog/341499 ）

2. 一路默认选项的安装下去即可

3. 环境变量设置

右键“我的电脑” -> “属性” -> “高级” -> “环境变量（系统变量）”
选中“PATH”，然后点击“编辑”，在最后添加：

```
;C:\Program Files\Java\jdk1.6.0_10\bin;C:\Program Files\Java\jdk1.6.0_10\jre\bin
```

然后点击“新建”，“变量名”输入“CLASSPATH”，“变量值”输入：

```
.;C:\Program Files\Java\jdk1.6.0_10\lib;C:\Program Files\Java\jdk1.6.0_10\lib\tools.jar
```

注意前面的".;"不能丢掉。然后保存退出。接着在“运行” -> “cmd”，分别输入 java 和 javac 命令回车，如果都有一长串东西输出，表明安装完成。

对于 Linux 用户：Java 环境应该都默认安装了。

Windows 用户可通过双击 SRV1Console.exe 来启动 SRV1Console 客户端。Linux 用户可通过执行 srv1 脚本启动 SRV1Console 客户端。

注意：连接的机器人 IP 地址是在 srv.conf 里面配置的，如果默认地址更改了，需要修改 srv.conf 里面设置。

几个常用客户端介绍：

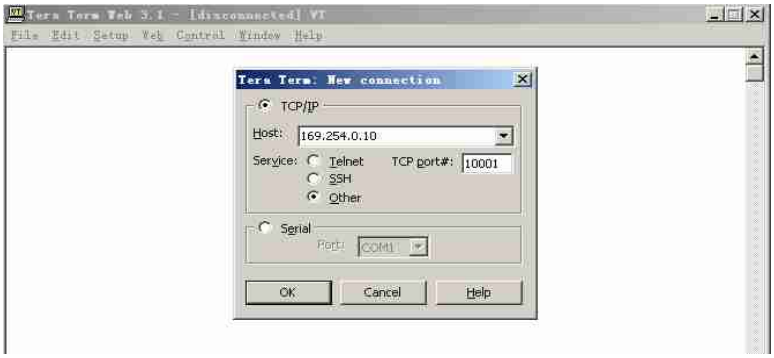
1、终端程序(Terminal)

终端程序是最简单最直接与机器人进行交互的方式。用户可以通过 TCP 或 Telnet 方式连接到机器人。

Linux 下可通过'netcat'命令进行连接：'nc robot-ip 10001'，如图所示，可以发送一个指令'V'常看当前固件的版本信息。 如下内容为测距传感器（指令'D'）与 GPS（指令'Ux'）操作记录：

```
zwang@wzc-x200:~$ nc 169.254.0.11 10001

#?V
##Version - SRV-1 Blackfin 17:15:54 - Aug 16 2012
##Version - SRV-1 Arduino - 2012-08-02

#?D
##Distance: 32 cm

#?U
====== Usage: ======
== U0: get $GPRMC ==
== U1: get $GPVTG ==
== U2: get $GPGGA ==
== U3: get $GPGSA ==
== U4: get $GPGSV ==
== U5: get $GPGLL ==
====================

#?U0
$GPRMC,094027.00,A,3958.39821,N,11620.21349,E,0.226,31.69,160812,,,A*5E

#?U1
$GPVTG,23.68,T,,M,0.483,N,0.895,K,A*09

#?U2
$GPGGA,094031.00,3958.39713,N,11620.21411,E,1,06,1.87,117.1,M,-8.7,M,,*44

#?U3
$GPGSA,A,3,20,01,32,04,17,28,,,,,,,3.07,1.87,2.44*00

#?U4
$GPGSV,3,1,09,01,18,056,36,02,10,246,21,04,48,254,27,17,73,001,27*74
$GPGSV,3,2,09,20,47,070,36,23,04,109,17,27,13,286,,28,54,181,27*74
$GPGSV,3,3,09,32,23,047,23*42

#?U5
$GPGLL,3958.39710,N,11620.21364,E,094042.00,A,A*6F
```

Windows 下使用的是 TTPRO（下载：www.ucrobotics.com/index.php/resource）程序进行连接。下载解压并进入 TTPRO 程序目录，双击 ttermpro.exe 会看到如下画面。



选中“other”选项， “TCP port#”设置为 10001，然后点击 OK 进行连接。

优点：方便直接，用于开发、调试和设置等。

缺点：只能显示字符，无法传递图像。(如发送一个指令"I"抓取一帧图像，这返回到终端的将会是一堆看似乱码的图像二进制字符。)

2、SRV1Console

SRV1Console 是一款 Java 语言开发的机器人动作客户端程序。可以通过按钮操控机器人的行为。

优点：简单易用，且有图像实时回传。

缺点：只有有限的几个控制按钮，无法完成与机器人更多的交互操作。

这里给出它的编译命令及参数说明。

编译命令：

```
javac -classpath .:ImageButton:RXTXcomm.jar:wstreamd_embed.jar: SRV1Console.java
```

执行命令：

```
$ cat SRV1Console/srv1
java -cp .:ImageButton:RXTXcomm.jar:wstreamd_embed.jar: SRV1Console -c
```

除了"-c"参数之外，还有一个"-p"参数：

- -c 表示客户端启动后自动链接机器人，不加该参数则需要点击客户端上的"connect"进行人工连接。
- -p xxxx 可以通过该参数更改 WebcamSat 连接的默认端口号，默认是 8888。

3、SRV1Test

SRV1Test 是一款 Java 语言开发的机器人控制客户端程序。同样具有图像实时回传功能，且可以实现对机器人的任何操控。该客户端程序与 SRV1Console 处在同一目录下。

优点：可以实现与机器人的任何交互，且有图像实时回传。
缺点：用户需要熟悉机器人的相关控制协议指令方能进行操作。

编译命令及参数说明。

编译命令：

```
javac SRV1Test.java
```

执行命令：

```
java -cp SRV1Test.jar SRV1Test -remote_addr 169.254.0.10 -remote_port 10001 -local_port 10001 -protocol TCP
```

相关参数：

- -remote_addr (SRV-1 机器人的 IP 地址 - 默认是 169.254.0.10)
- -remote_port (SRV-1 机器人的端口号 - 默认是 10001)
- -local_port (主机端口号 - 默认是 10001)
- -protocol (使用的协议 TCP 或 UDP - 默认是 TCP)
- -archive (采集所有的图像帧到指定目录)

重要说明：

1. 关于使用的协议，经测试 UDP 好像不能正常工作，所以最好使用默认的 TCP

2. 采集参数后面必须指定目录，如"-archive ."表示保持到当前目录。保存的是每一帧图片，而非一个连续的视频文件。（可同样参照“移动客户端连接”一节提供的方法转换成 avi 视频格式）

4、Python Console

SRV1SDLConsole.py（下载：www.ucrobotics.com/index.php/resource）是一款通过 Python 实现的基于 SDL 库开发的客户端程序。可以通过键盘按键进行操控。

优点：键盘直接操控，方便直观。
缺点：动作不连续执行，需要频繁按键操作。

使用方法：安装 python 环境、pygame 模块以及 SDL 库。下载后先编辑里面的"robotip"指向你的实际机器人 IP 地址，然后执行"python SRV1SDLConsole.py"。

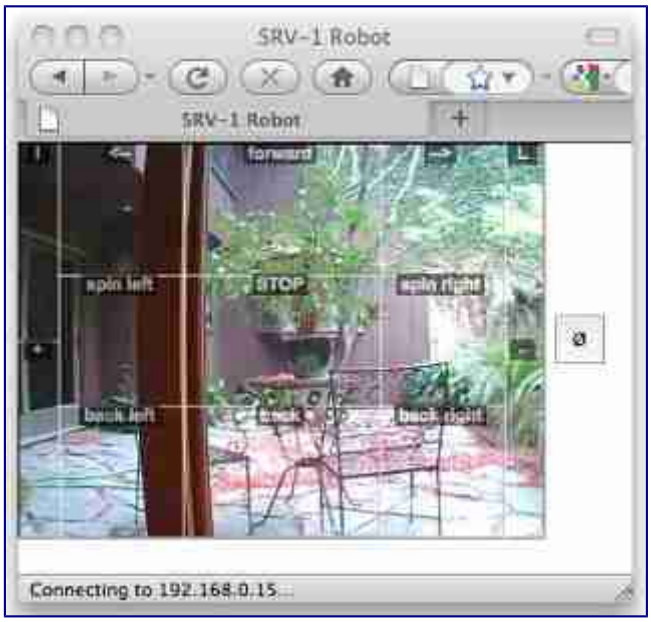
如果连接正常，将会看到一个如上图所示的 800x600 图像窗口，窗口中的红色字体表示当前的电机(servo)功率/速度，左右电机的默认速度都是 063。支持如下控制按键：

- Space - 空格键，实时采集并返回一帧图像
- ESC - 退出该客户端程序
- UP - 上方向键，控制机器人前进(每次按键触发，动作只执行 160ms，可连续触发，下同)
- DOWN - 下方向键，控制机器人后退
- LEFT - 左方向键，控制机器人左后退(即右转弯)
- RIGHT - 右方向键，控制机器人右后退(即左转弯)
- PERIOD - 句号键，增加电机功率(默认以 10 为单位递增，配合 Shift 键则以 1 往上递增；测试发现与 Shift 键配合好像不工作。)
- COMMA - 逗号键，降低电机功率(默认以 10 为单位递减，配合 Shift 键则以 1 往下递减；测试发现与 Shift 键配合好像不工作。)
- a - 设置采集分辨率为 160x128(注意：图像显示区域大小固定，分辨率只影响清晰度；默认分辨率是 320x240)
- b - 设置采集分辨率为 320x256
- c - 设置采集分辨率为 640x512

- A - 设置采集分辨率为 1280x1024(切记，分辨率越大，图像尺寸越大，回传越慢，慎用)
- p - 保存当前帧图像到该客户端程序所在的目录
- r - 查找红色色块(好像工作不稳定，慎用)
- v - 切换图像(好像会将图像切分成 5 个区域)
- l - 开启激光探测点
- L - 关闭激光探测点
- f - 向前行驶(好像工作不稳定，慎用)

5、Web Console

可以通过固件内置的 Web 控制页面对机器人进行操控。控制的原理其实就是通过 Ajax 请求对 Web 上图片进行实时回传更新。将浏览器的 URL 指向：http://169.254.0.10:10001/ 即可访问。



通过截图可以看出，控制区域与图像是重叠的，所以可以充分使用屏幕资源。右边的按钮用于开启/隐藏控制区域。

优点：平台无关性，适合屏幕资源紧缺的设备，如手机，PDA 等手持设备。
缺点：有限的交互操作，且性能还有待改进。

6、其它更多的客户端软件及界面库

- LabVIEW 2009 Console "labview-srv" - code.google.com/p/labview-srv/
- Roborealm - www.roborealm.com/help/Surveyor_SRV1.php
- DelphiSRV Console - Delphi-based console for SRV and SVS - latest version needs srv-blackfin-081809 or later firmware
- C# framework - AForge.NET with SRV-1 and SVS support - www.aforgenet.com/framework/

- Nintendo DS Console for SRV1 - roboticjourney.blogspot.com/
- SDL-based C/C++ console - client-srv1
- Matlab Control Scripts- Drexel University MEM456/MEM800: RoboticsII/Advanced Robotics
- Microsoft Robotics Studio - www.surveyor.com/MSRS.html
- SRV1 CSharp Console - www.ofitselfso.com/SRV1CSharpConsole.php
- RoboBrain Console for OS/X - www.voxatec.com/
- Transterpreter / Occam-Pi - www.transterpreter.org
- C++ Console - agents.sci.brooklyn.cuny.edu/robotics.edu/bcsoftware.php
- Webots - www.cyberbotics.com/

六、内置 SRV_Httpd 服务介绍

SRV_httpd 是 SRV-1 机器人固件程序中内置的一个微型 HTTP 服务模块，该模块可以实现通过编辑浏览器的 URL 对 SRV-1 机器人进行各种操作。既简化了机器人的操作步骤，也实现了对机器人操控的平台无关性。该模块定义的接口与 "SRV-1 机器人控制协议" 的定义是一致的，所以对于描述信息不是很丰富的项目，请参阅控制协议相关部分。

注意：有些基于之前老版本机器人(如四驱版 SRV-4WD，导航版 SRV-NAV)的指令是不可用的！

http://169.254.0.10:10001/index.html - 加载存储在 Flash 扇区 10-11 中的机器人控制脚本(即 Web 客户端界面)
http://169.254.0.10:10001/admin - 管理界面，用于更新固件和脚本等程序到机器人的 Flash(大大简化固件更新操作)
http://169.254.0.10:10001/robot.jpg - 从摄像头实时采集一帧图像并传回到浏览器(webcam 界面)

robot-ip:10001/robot.cgi? - 机器人控制功能入口：

/robot.cgi?$a_ - 从四驱 SRV 机器人中读取模拟通道，'_'的取值范围如下：(所谓四驱 SRV 是之前一款老版本的机器人，所以可以忽略)

* channel 0 = battery (通道 0 表示电池)
* channel 1 = 5V gyro (通道 1 表示 5V 陀螺仪)
* channel 2 = 3.3V gyro (通道 2 表示 3.3V 陀螺仪)
* channel 3 = IR 1 (红外传感器 1)
* channel 4 = IR 2 (红外传感器 2)
* channel 6 = IR 3 (红外传感器 3)
* channel 7 = IR 4 (红外传感器 4)

(SRV-NAV 也是之前一款老版本的机器人，所以也可以忽略)
/robot.cgi?$e_ - 读取指定的轮编码器(wheel encoder)，'_'的取值范围如是：1, 2, 3 or 4
/robot.cgi?$c - 读取 SRV-NAV(Surveyor Navigation Board, 即 GPS 模块)上的罗盘(compass)
/robot.cgi?$g - 读取导航 SRV 机器人(SRV-NAV)上的 gps
/robot.cgi?$T_ - 读取导航 SRV 机器人(SRV-NAV)上的倾斜传感器在指定坐标轴上的值，'_'的取值范围是 1,2,3, 分别代表 x,y,z 坐标轴

/robot.cgi?V - 返回固件的版本信息

/robot.cgi?t - 返回机器人从重置到现在的时间，毫秒显示
/robot.cgi?$! - 重置 Blackfin 控制板

/robot.cgi?m____ - PWM 电机(motor)控制，如 m5050 表示停止两个电机(取值范围是 01~99, 50=stop, 01=最大反向值(max rev), 99=最大正向值(max fwd))
/robot.cgi?x____ - 对四驱 SRV 机器人电机(motor)控制，如 x5050 表示停止两个电机
/robot.cgi?S____ - 设置 2/3 通道上的伺服装置(servos, 即舵机)，如 S5050 表示将这两个伺服装置调整到居中位置
/robot.cgi?s____ - 设置 6/7 通道上的伺服装置(servos, 即舵机)，如 s5050 表示将这两个伺服装置调整到居中位置

/robot.cgi?$A__ - RCM(SRV-1 扩展模块) analog read channel 01-08, 11-18, 21-28
/robot.cgi?l1 - 开启激光束
/robot.cgi?l0 - 关闭激光束

/robot.cgi?a - 设置采集分辨率为 160x120
/robot.cgi?b - 设置采集分辨率为 320x240
/robot.cgi?c - 设置采集分辨率为 640x480(OV7725 最大可支持到该分辨率)
/robot.cgi?d - 设置采集分辨率为 1280x1024(OV9655 最大可支持到该分辨率；注意：分辨率设置的越大，图像尺寸越大，回传越慢)
/robot.cgi?qx - 设置 JPEG 图像的质量，x 取值范围是 1~8(1 表示最高质量，默认值是 4)

/robot.cgi?iraa - I2c 命令部分，详细描述参见控制协议文档，下同
/robot.cgi?iRaa
/robot.cgi?iwaabb
/robot.cgi?iWaabbcccc
/robot.cgi?idaabbccddee

/robot.cgi?g0 - 设置差分帧(set frame differencing)
/robot.cgi?g1 - 开启颜色分割显示(enable color segmentation display)
/robot.cgi?g2 - 开启边缘检测显示(enable edge detect display)
/robot.cgi?g3 - 开启地平线检测显示(enable horizon detect display)
/robot.cgi?g4 - 开启障碍物检测显示(enable obstacle detect display)
/robot.cgi?g5 - 开启立体视觉差异显示(enable stereo disparity display) (仅适用于 SVS 双摄像头系统机器人)
/robot.cgi?g6 – 开启粒子(blob,图像处理软件词汇)检测显示
/robot.cgi?gx - 关闭以上的所有特殊显示

/robot.cgi?va - 开启/关闭 AGC/AWB/AEC 等摄像头控制功能
/robot.cgi?vb - 查找匹配 color bin #x 的粒子
/robot.cgi?vc - 设置 color bin 的颜色信息
/robot.cgi?vm - 计算 YUV 颜色的平均值
/robot.cgi?vp - 提取单个像素
/robot.cgi?vr - 检索 color bin #c 中存储的颜色信息
/robot.cgi?vs - 边缘扫描
/robot.cgi?vt - 为边缘检测设置全局变量"edge_thresh"(范围是 0000-9999，默认是 3200)。
/robot.cgi?vz - 将所有的颜色区间校正归零

七、内置的 C 解释器介绍

在 SRV-1 机器人的产品特点描述中明确表述了其自主运行的能力，所谓自主运行就是指脱离人为的控制和干预，以

程序预选设计的方式完成特定的任务，这也是智能机器人的重要特征之一。而通过 SRV-1 中内嵌的 C 解释器程序，用户可以很容易做到这一点。

内置的 C 解释器程序来自于开源项目 picoC：http://code.google.com/p/picoc/ 。通过该 C 解释器，用户可以使用类似 C 语言的语法规则编写脚本程序(无需编译，直接解释执行)，以便让机器人能够自主完成某些特定的任务。下面是一些示例代码以及函数简介。

/* 注释信息 */

```
printf("Hello\n"); /* 这是一条注释信息(跟 C 的一样) */
printf("Hello\n"); // 这也是一条注释(跟 C 的一样)
```

/* printf 语句 */

```
int Count;
for (Count = -5; Count <= 5; Count++)
        printf("Count = %d\n", Count);
printf("String 'hello', 'there' is '%s', '%s'\n", "hello", "there");
printf("Character 'A' is '%c'\n", 65);
printf("Character 'a' is '%c'\n", 'a');
```

/* 结构体定义及使用 */

```
struct fred
{
        int boris;
        int natasha;
};
struct fred bloggs;
bloggs.boris = 12;
bloggs.natasha = 34;
printf("%d\n", bloggs.boris);
printf("%d\n", bloggs.natasha);
```

/* 数组定义及使用 */

```
int Count;
int Array[10];
for (Count = 1; Count <= 10; Count++)
        Array[Count-1] = Count * Count;
for (Count = 0; Count < 10; Count++)
        printf("%d\n", Array[Count]);
```

/* switch 关键字 */

```
int Count;
for (Count = 0; Count < 4; Count++)
{
        printf("%d\n", Count);
        switch (Count)
        {
        case 1:
                printf("%d\n", 1);
                break;
        case 2:
                printf("%d\n", 2);
                break;
        default:
                printf("%d\n", 0);
                break;
        }
}
```

/* while 和 do while */

```
int a;
```

```
int p;
int t;
a = 1;
p = 0;
t = 0;
while (a < 100)
{
	printf("%d\n", a);
	t = a;
	a = t + p;
	p = t;
}
do
{
	printf("%d\n", a);
	t = a;
	a = t + p;
	p = t;
} while (a < 100);
```

/* 指针调用 */

```
int a;
int *b;
int c;
a = 42;
b = &a;
printf("a = %d\n", *b);
struct ziggy
{
	int a;
	int b;
	int c;
} bolshevic;
bolshevic.a = 12;
bolshevic.b = 34;
bolshevic.c = 56;
printf("bolshevic.a = %d\n", bolshevic.a);
printf("bolshevic.b = %d\n", bolshevic.b);
printf("bolshevic.c = %d\n", bolshevic.c);
struct ziggy *tsar = &bolshevic;
printf("tsar->a = %d\n", tsar->a);
printf("tsar->b = %d\n", tsar->b);
printf("tsar->c = %d\n", tsar->c);
```

/* 宏定义 */

```
#define FRED 12
#define BLOGGS(x) (12*(x))
printf("%d\n", FRED);
```

/* int 整型操作 */

```
int a = 24680;
int b = 01234567;
int c = 0x2468ac;
int d = 0x2468AC;
int e = 0b010101010101;
printf("%d\n", a);
printf("%d\n", b);
printf("%d\n", c);
printf("%d\n", d);
```

/* if else 语句 */

```
int a = 1;
if (a)
	printf("a is true\n");
else
	printf("a is false\n");
int b = 0;
```

```
if (b)
	printf("b is true\n");
else
	printf("b is false\n");
```

/* 递归调用 */

```
int factorial(int i)
{
	if (i < 2)
	return i;
	else
	return (i * factorial(i - 1));
}
int Count;
for (Count = 1; Count <= 10; Count++)
	printf("%d\n", factorial(Count));
```

/* 嵌套语句 */

```
int x, y, z;
for (x = 0; x < 2; x++)
{
	for (y = 0; y < 3; y++)
	{
		for (z = 0; z < 3; z++)
		{
			printf("%d %d %d\n", x, y, z);
		}
	}
}
```

/* 相关机器人功能函数：*/

```
* void autorun(int seconds)
```

如果使用该函数，则只能在脚本的第一行调用。例如，autorun(20)表示在机器人开机 20 秒后将自动执行存储在 Flash 缓冲区中的 C 语言脚本。当然了，如果在这 20 秒之内，机器人的 uart 串口接收到了用户发送的 ESC，则放弃自启动(注意：如果添加 autorun 语句，记得时间要设置长一些，如 30 秒，否则发送 ESC 命令终止时会比较仓储)。

```
* int abs(int data)
```

获取整数 data 的绝对值

```
* int acos(int adjacent, int hypotenuse)
```

arccos(adjacent, hypotenuse) ——反余弦函数

```
* int analog(int channel)
```

读取 AD7998 八通道 12 位 A/D 转换(read AD7998 8-channel 12-bit A/D)

- channel 1-8 相当于 i2c 设备中的 0x20
- channel 11-18 相当于 i2c 设备中的 0x23
- channel 21-28 相当于 i2c 设备中的 0x24

```
* int analogx(int channel)
```

从四驱 SRV 机器人中读取模拟通道(analog channel)

- channel 0 = battery (通道 0 表示电池)
- channel 1 = 5V gyro (通道 1 表示 5V 陀螺仪)
- channel 2 = 3.3V gyro (通道 2 表示 3.3V 陀螺仪)
- channel 3 = IR 1 (红外传感器 1)
- channel 4 = IR 2 (红外传感器 2)

- channel 6 = IR 3 (红外传感器 3)
- channel 7 = IR 4 (红外传感器 4)

* int asin(int opposite, int hypotenuse)

arcsin(opposite, hypotenuse) ——反正弦函数

* int atan(int opposite, int adjacent)

arctan(opposite, adjacent) ——反正切函数

* int battery()

电池状态 1=okay, 0=low battery

* int compass()

读取 HMC6352 罗盘(read HMC6352 compass)

* int compassx()

读取导航 SRV 机器人(SRV-NAV)上的 HMC5843 罗盘(NAV - Surveyor Navigation Board, 即 GPS 模块)
注意，可以通过 cxmin, cxmax, cymin, cymax 来访问最小/最大校准数据(calibration data)。

* void compassxcal(xmin, xmax, ymin, ymax, continuous_calibration)

设置 HMC5843 的校准数据(calibration data)

- 使用控制台命令$c 来收集数据
- 通过"continuous_calibration"标记来决定 compassx()函数是否继续收集校准数据(calibration data)。"continuous_calibration"标记: off = 0, on = 1
- 该函数对于编写自动校准程序非常有帮助，如固件里面的测试程序 test4wd.c

* int cos(int angle)

cos(angle) * 1000 ——余弦函数

* void delay(int milliseconds)

延时 xxx 毫秒

* void encoders()

通过轮编码器计算脉冲/秒(compute pulses/second from wheel encoders)
通过全局变量 lcount 和 rcount 返回数据

* int encoderx(channel)

从四驱 SRV 机器人(SRV-4WD)中的特殊电机编码器(motor encoder)1-4 中读取累计的脉冲数

- 在 65535 次脉冲之后计算转数(count cycles after 65535 pulses)
- 取决于轮子的尺寸，通常 4.5 寸(4.5")轮子上的 1000 次脉冲大约等于一步的距离

* void exit()

一般用于脚本结束时调用，如果是在终端则会退出 picoC 交互模式回到机器人等待接受处理指令的状态。

* void gps()

从 gps 中解析字符串$GPGGA

- 通过全局变量 gpslat, gpslon, gpsalt, gpsfix, gpssat 和 gpsutc 返回数据
- 36.5 deg = 36500000
- 100.5W deg = -100500000

* int gps_dist(int lat1, int lon1, int lat2, int lon2)

以米为单位计算两个 gps 坐标之间的距离

- 坐标的格式为：deg*1000000
- 36.5 deg = 36500000
- 100.5W deg = -100500000

* int gps_head(int lat1, int lon1, int lat2, int lon2)

以度为单位计算两个 gps 坐标之间的方向

- N == 0-deg
- 36.5 deg = 36500000
- 100.5W deg = -100500000

* void init_uart1(int baudrate)

初始化 Blackfin 上的第二个 UART

* int input()

return single character from read of serial channel(urat0，该函数功能等同于 getch())

* int input1()

return single character from read of uart1

* void iodir(int iopins)

设置 GPIO-H15/14/13/12/11/10 为输入或输出引脚

- 0 = input, 1 = output
- iodir(0x31) == H15-out H14-out H13-in H12-in H11-in H10-out
- iodir(0b110001) == H15-out H14-out H13-in H12-in H11-in H10-out
- iodir(0x03) == H15-in H14-in H13-in H12-in H11-out H10-out

* int ioread()

读取 GPIO H15/14/13/12/11/10 引脚的状态值。例如：H15=1 H14=1 H13=0 H12=0 H11=0 H10=0, 那么 ioread()的返回值为 48 == 0x30

* void iowrite(int iopins)

设置 GPIO H15/14/13/12/11/10 引脚的状态值。例如：iowrite(0x31)或 iowrite(0b110001) 将会使 H15=1 H14=1 H13=0 H12=0 H11=0 H10=1

* void laser(int which_laser)

控制两个激光的开和关。0=off, 1=left, 2=right, 3=both

* void motors(int left, int right)

设置左右 PWM 电机(motor)的功率，取值范围是：-100～100

* void motors2(int left, int right)

设置左右 PWM2 电机(tmr6/7)的功率，取值范围是：-100～100

* void motorx(int left, int right)

设置四驱 SRV(SRV-4WD)机器人的电机功率。取值范围是：-100～100

* void nninit()

神经网络初始化

* void nnlearnblob(int pattern_number)

将粒子(blob,图像处理软件词汇)缩放并保存为 8x8 像素的模式

* int nnmatchblob(int blob_number)
* void nnset(int first8bits, int second8bits, int ..., int, int, int, int, int, int)

设置 nn 模式

* void nnshow(int which_pattern)

显示 nn 模式

* int nntest(int first8bits, int second8bits, int ..., int, int, int, int, int)
* void nntrain()

训练神经网络(train neural net)

* void output(int):

output a single character to serial channel (uart0)

* void output1(int)

output a single character to uart1

* int peek(int address, int wordsize)

int x = peek(addr, size) where size = 1, 2, 4 bytes
byte/short/word alignment is forced(强制对齐)

* void poke(int address, int wordsize, int value)

poke(addr, size, val) where size = 1, 2, 4 bytes
byte/short/word alignment is forced(强制对齐)

* int rand(int number_range)

生成 0～xxx 之间的随机数

* int range()

使用激光点估距

* int read_int()

从控制台读入一个整数，终止字符可以是除'-'和'0'～'9'之外的任意字符。

* int read_str(char *)

从控制台读入一个字符串并存入字符数组中，返回值是字符串中的字符数。终止字符是 0x00 或者 0x01, 或者字符数超过 1023 时自动终止。

* int readi2c(int device, int register)

从指定的 i2c 设备地址读一字节

* int readi2c2(int device, int register)

从指定的 i2c 设备地址读两字节

* void servos(int timer2, int timer3)

set pin 7/8 (tmr 2/3) PPM levels 0 to 100

* void servos2(int timer6, int timer7)

set pin 5/6 (tmr 6/7) PPM levels 0 to 100

* int signal()

对串口通道(uart0)的输入进行非阻塞检测 - 如果返回非零则表明可以正常输入(non-zero return indicates an input)

* int signal1()

对第二个串口通道(uart1)的输入进行非阻塞检测 - 如果返回非零则表明可以正常输入(non-zero return indicates an input)

* int sin(int angle)

sin(angle) * 1000 ——正弦函数

* int sonar(int which_channel)

ping modules 1, 2, 3 or 4

* int sqrt(int value)

计算整数平方根

* int tan(int angle)

tan(angle) * 1000 ——正切函数

* int tilt(int axis)

返回倾斜传感器从指定坐标轴中读取的坐标值。取值范围是 1,2,3, 分别代表 x,y,z 坐标轴。该函数的使用方法如下：
int x = tilt(1); int y = tilt(2); int z = tilt(3);

* int time()

返回开机到现在为止的时间值，毫秒为单位。

* int vblob(int color_bin, int which_blob)

搜索匹配 color bin #x 的粒子(blob,图像处理软件词汇)，返回匹配的粒子数目。

- 'int'整型的返回值显示了有多少个匹配粒子被发现
- 第二个参数决定使用哪个粒子进行匹配(2nd value determines which blob (largest to smallest))
- 通过全局变量 blobcnt, blobx1, blobx2, bloby1, bloby2 返回数据

* void vcam(int settings):

开启/关闭自动增益，白平衡以及曝光相机等功能(默认 x=7)

- vcam(4) -> AGC enable
- vcam(2) -> AWB enable
- vcam(1) -> AEC enable
- vcam(7) -> AGC+AWB+AEC on
- vcam(0) -> AGC+AWB+AEC off

* void vcap()

采集一帧图像。

* void vcolor(int color_bin, int ymin, int ymax, int umin, int umax, int vmin, int vmax)

设置 color bin #x 的值为 ymin, ymax, umin, umax, vmin, vmax

* void vdiff(int flag)

开启/关闭 vcap()差分(enable/disable differencing with vcap())

- vdiff(1)表示开启
- vdiff(0)表示关闭

* int vfind(int color, int x1, int x2, int y1, int y2)

计算 x1->x2, y1->y2 区域内 color bin #x 中的像素数。

* int vjpeg(int quality)

压缩通过 vcap()采集的图像。 使用 vsend()函数传送：

```
int size = vjpeg(int quality); vsend(size);
```

返回值是JPEG 图像的大小。输入的参数指定JPEG 图像的质量(范围是 1~8, 1 代表最高质量)

```
* void vmean()
```

获取所有图像帧的 YUV 平均值
通过全局变量 y1, u1, v1 返回数据

```
* void vpix(int x, int y)
```

获取图像中（x,y）坐标那一像素点的 YUV 值

- 通过全局变量 y1, u1, v1 返回数据
- 例如 vpix(0,0)是指读取图像左上角第一个像素点的 YUV 值

```
* void vrcap()
```

为差分采集参考帧(capture reference frame for differencing)

```
* int vscan(int columns, int threshold)
```

边缘检测函数

- 计算边缘像素数并将图像分成多列(counts edge pixels and divides image into columns)
- 行范围是 1~9，阈值范围是 0001～9999 (4000 是一个很好的起点)
- 返回每一列中从图像底部到第一个边缘像素的距离。
- 如将结果存储在数组 scanvect[] 之中：

```
int ii;
vcap();
vscan(3, 4000); // search 3 columns, set threshold to 4000
for (ii=0; ii<3; ii++)
    printf("column %d distance %d\r\n", ii, scanvect[ii]);
```

```
* void vsend(int size)
```

传送通过 vcap()和 vjpeg()采集和压缩过的 JPEG 图像

```
* void writei2c(int device, int register, int value)
```

向指定的 i2c 设备地址写一字节

八、C 解释器使用简介及 C 脚本编程入门

因为有了 C 解释器的支持，SRV-1 机器人才具备了自主执行程序的能力，这也是其智能特性的重要展现方式之一。可以通过两种方式来执行 C 脚本程序：

1. 直接编辑 Flash 缓冲区

先通过终端程序以 TCP 或 Telnet 方式连接到机器人，如使用'netcat'命令进行连接：'nc 169.254.0.10 10001'，然后使用指令'E'开启 Flash 缓冲区的的行编辑器即可进行编辑。

```
$ nc 169.254.0.10 10001
V
##Version - SRV-1 Blackfin w/picoC 1.0 11:59:13 - Nov 2 2010
#?E
(T)op (B)ottom (P)revious line (N)ext line (L)ist (I)nsert until ESC (D)elete line (H)elp (X)exit
*
* i  ------>进入插入模式
INSERT
printf("hello\n");  ------> 输入内容
```

```
    printf("hello\n");  ------> 回显输入的内容
    ^[   ------> 发送 ESC 退出插入模式
    *
    * t  ------> 指向 Flash 缓冲区的顶端
    TOP
    *
    * l  ------> 输出 Flash 缓冲区里面的前 20 行内容
    printf("hello\n");
    *
    * x  ------> 退出行编辑器
    leaving editor
    #?
    #?Q  ------> 执行 Flash 缓冲区里面的程序(因为程序的最后没有调用 exit()函数，所以程序执行结束后还会再回到 picoC 的交互模式，需要发送 ESC 退出该交互模式)
    hello starting
    picoc >
    -
    -
    - ^[ ------> 发送 ESC 退出 picoC 交互模式
    leaving picoC
    #?
    #?
```

关于指令'E'支持的几个编辑选项：

- t - (T)op 指向 Flash 缓冲区的顶部
- b - (B)ottom 指向 Flash 缓冲区的底部
- p - (P)revious line 指向前一行
- n - (N)ext line 指向后一行
- l - (L)ist 输出从当前位置开始的前 20 行数据
- i - (I)nsert until ESC 进入插入模式，通过 ESC 退出该
- d - (D)elete line 删除行(测试发现好像每次仅删一个字符)
- h - (H)elp 显示帮助信息
- x - (X)exit 退出行编辑器

注意：通过行编辑器写入的数据只是存储在缓冲区中的，一旦机器人重启或断电数据就会丢失。所以要是想长期保存，可以在编辑完成使用'zw'将 Flash 缓冲区的数据写入到 Flash 扇区(磁盘介质)。

关于 Flash 缓冲区操作的几个命令小结：

- zc – 清空 Flash 缓冲区(如果不想重启机器人，该指令可以清空前面写入的数据)
- zC - 计算 Flash 缓冲区的 crc16_ccitt(一种校验算法)
- zd – 将 flash 缓冲区的内容输出到机器人终端(一般在发送指令'Q'执行程序之前，可以通过'zd'查看前面的输入内容是否正确)
- zr – 将 flash 扇区(#04)的内容取回到 Flash 缓冲区(因为程序只能在 Flash 缓冲区中才能执行，所以如果要执行存储在扇区的程序，必须先执行指令'zr')
- Rxx – 取回指定扇区的内容到 Flash 缓冲区(范围是 02~63，因为 00 和 01 禁止写入)
- zw – 将 flash 缓冲区的数据写入到 Flash 扇区(磁盘介质，#04)
- zWxx – 将 flash 缓冲区的数据写入到指定的 Flash 扇区(范围是 02~63，因为 00 和 01 禁止写入)
- E – 开启对 Flash 缓冲区直接操作的行编辑器
- Q - 执行 Flash 缓冲区中的 C 脚本程序

关于这些指令的更多详细信息请参阅附录：SRV-1 机器人控制协议

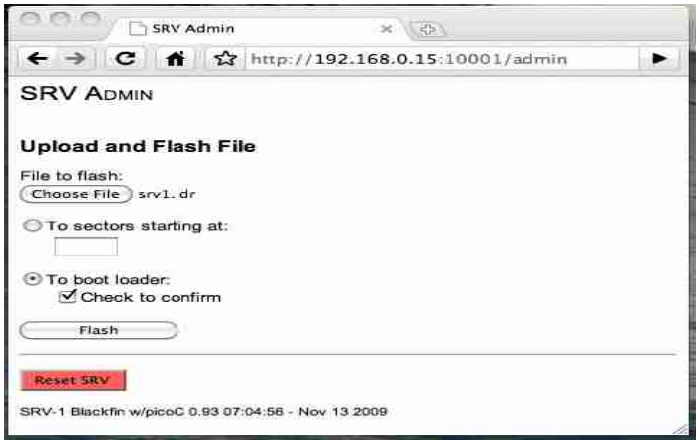
2. 上传 C 脚本到 Flash 扇区

如果需要进行大量的编程，当然还是应该先在本地编辑，然后上传到 Flash 扇区了。

注意：picoC 中是没有 main()函数的，所以 C 解释器文档中的任何一段示例代码都可以作为独立程序运行。
这里有一个示例程序：test.c，其功能是通过对随机取值的判断进行左漂移或是右漂移运行。

```
//autorun(30); /* 必须放到第一行 */

/*
  如果放开上面对 autorun(x)函数的注释，则会在机器人启动后 x 秒后自动执行该脚本；而本测试程序是一个无限循环，所以一定要将自启动时间设置大一些，如 30 秒或更多，否则就没有足够的时间来终止自启动程序的执行了；
  要终止自启动，需要在定义的时间内(如 30s)，通过 nc 命令连接到机器人，然后发送一个 ESC 即可。所以不建议将 autorun(x)与无限循环一起测试。
*/

int x;
char ch;

ch = 0;
while (ch == 0) {
    x = range();   /* user laser pointer ranging */
    printf("range = %d\n", x);
    if (x < 30) {
        motors(10, 50);  /* drift right */
    } else {
        motors(50, 10);  /* drift left */
    }
    delay(3000); /* delay 3s, also means run 3s every time */
    //ch = input();  /* continue until any console input */
}
motors(0, 0);
exit();
```

下载后通过浏览器访问：http://169.254.0.10:10001/admin, 然后选中“To sectors starting at: ”选项，下方输入框输入 04，然后确认上传。上传成功后顶栏会有绿色消息提示，然后重启机器人。（注：下图是烧写固件时选项的截图）



如果放开了脚本第一行的 autorun(x)函数注释，则重启后等待 30 秒会自动自行脚本程序的；
如果没有，则需要手动执行该脚本：先连接到机器人终端（如 nc 169.254.0.10 10001），然后执行指令'zr'将扇区中的程序（就是之前上传保存的脚本程序）加载到缓冲区，然后发送指令'Q'执行即可。

附录：SRV-1 机器人控制协议

从控制终端发送到 SRV-1 机器人的所有指令都是由 ASCII 字符，或带 8 位二进制的 ASCII 字符，或十进制 ASCII 字符构成。所有发送的指令都会从机器人端收到一个返回，返回字符串为：以'#'为前缀的原指令字符串或以'##'为前缀的可变长度指令字符串。可变长度指令不会指定返回值的大小，只是给返回字符串追加一个回车符（CR）和一个换行符（'\r\n'）。

注意：所有的指令都能通过终端程序以 TCP 或 telnet 方式连接到机器人进行执行。例如：用户可以使用'netcat'命令进行连接'nc robot-ip 10001'

当机器人第一次上电启动的时候，会自动输出版本指令'V'的执行结果，形如字符串"##Version ...\r\n"，由此便可以获知当前运行固件的版本信息（一般是指更新了固件之后，第一次上电通过 nc 连接会自动输出版本信息，以后再次上电连接则需要 人工发送指令'V'来查看版本信息）。机器人启动后大约还需要 2 秒的延迟才能接收处理指令。因为这期间它需要进行一些初始化工作，如初始化摄像头和其它传感器等。初始化过程处理器会重启，同时那个黄色的指示灯会闪烁。启动完成之后，可以测试以下指令：

- 发送指令'V'来获取固件版本信息。
- 发送指令'I'来获取一帧图像，当然了，在终端上显示的将会是一堆乱码了，而且会充满你的屏幕。

在固件里面内嵌了一个叫做"picoC"的 C 解释器（interpreter）。如果在 Flash 缓冲区已经有了 C 程序，可以通过指令'Q'来运行该 C 程序；也可以通过指令'!'来开启一个 C 交互程序界面；使用 ESC 键退出该交互程序。

可以通过指令'zr'或'X'来操作闪存缓冲区：指令'zr'可以将用户 Flash 段（flash segment）的内容传输到闪存缓冲区；指令'X'可以将主机上的文件通过 XMODEM 协议传输到闪存缓冲区。

在执行程序之前，可以使用指令'zd'对闪存缓冲区的内容进行检查。假如有一个 C 程序不是无线循环的，那么在其执行完毕之后会将控制权返还给 C 解释器（即 返回到 C 交互程序界面）。但如果该 C 程序调用了'exit()'函数，其结果就像点击 ESC 键一样，会退出 C 交互程序界面，返回到 SRV-1 的命理处理终端。假如在 C 程序的第一行调用了'autorun(x)'函数，且该 C 程序已经被存储到机器人的第 4 个 Flash 扇区（flash sector #4），则机器人将会在启动'x'秒之后自动执行该 C 程序。如果要取消该自启动行为，则必须在自启动之前发送 ESC 中断。

指令	返回	描述
新增加的传感器指令： 适用于 SRV-A 和 UCR-3		
'H'	'$GPRMC,...'	获取 GPS 返回值的$GPRMC 字段，同'U0'返回一样，但返回效率更高
'K'	'##accelgyro: ...'	获取陀螺仪读数
'Ux'	'$GPXXX,...'	获取 GPS 返回值的各个字段。x 取值范围是 0~5；当然也可以直接发送 U，返回的使用帮助信息。 U0 - 返回 $GPRMC 字段 U1 - 返回 $GPVTG 字段 U2 - 返回 $GPGGA 字段 U3 - 返回 $GPGSA 字段 U4 - 返回 $GPGSV 字段（可能有多条信息返回） U5 - 返回 $GPGLL 字段
'W'	'##Distance: xx cm'	获取测距传感器读数
机器人核心指令： 除了指令'q'之外，所有参数都是以 8 位二进制字符发送的(0x00-0xFF)		

指令	返回	描述
'7'	'#7'	左漂移(左方向转大圈) **注意：**机器人在收到一个形如'Mxxx'电机控制指令之前是不会将'1'-'9'这些指令转化成实际动作的！关于指令'M'请参考后面，如'M505000'。
'8'	'#8'	向前
'9'	'#9'	右漂移(右方向转大圈)
'4'	'#4'	左转(左方向转小圈)
'5'	'#5'	停止
'6'	'#6'	右转(右方向转小圈)
'1'	'#1'	后退左转
'2'	'#2'	后退
'3'	'#3'	后退右转
'0'	'#0'	左旋 20 度
'.'	'#.'	右旋 20 度
'+'	'#+'	提升电机转速
'-'	'#-'	降低电机转速
'<'	'#<'	电机平衡向左调整
'>'	'#>'	电机平衡向右调整
'a'	'#a'	设置采集分辨率为 160x120
'b'	'#b'	设置采集分辨率为 320x240(默认分辨率)
'c'	'#c'	设置采集分辨率为 640x480(ov7725 型号摄像头最高分辨率支持到此)
'd'/'A'	'#A'	设置采集分辨率为 1280x1024(ov9655 型号摄像头最高分辨率支持到此)
'D'	'##D'	检测 MN13812 型号(SRV-1 所用)电池的电量，返回字符串为以下二者之一： ##D - battery voltage okay(表示电量充足) ##D - low battery voltage detected(表示电量所剩不多)
'E'	-	开启 Flash 缓冲区的的行编辑器，编辑命令列表：(T)op (B)ottom (P)reviouse (N)ext line (L)ist (I)nsert until ESC (D)elete (H)elp (X)exit
'Fab'	'#F'	对电机控制启用故障保护模式。 参数'ab'以 8 位二进制发送：a = 左电机/伺服故障安全等级，b = 右电机/伺服故障安全等级 如果在 2 秒之内没有通过无线链路收到任何命令，则表示为'M'和'S'指令设置的电机/伺服等级。
'f'	'#f'	关闭故障保护模式
'G'	HTML stream	HTTP GET 命令，如解析 HTTP 的请求。 GET /index.html HTTP/1.1 - 确认文件/00.html .. /09.html, 分别存储在 Flash 扇区 10/11, 12/13 .. 28/29 之中

指令	返回	描述
		- /index.html == /00.html - 将使用现场采集的JPEG（base64 live captured JPEG）替换"$$camera$$" - 扇区对(sector pair)使用'zBxx'进行存储，如'zb10'将128K内容从Flash缓冲区存储到10/11扇区。
'I'	'##IMJxs0s1s2s3....'	获取JPEG视频压缩帧 x = 帧尺寸的像素：1 = 80x64, 3 = 160x120, 5 = 320x240, 7 = 640x480, 9 = 1280x1024 s0s1s2s3=帧尺寸的字节： (s0 * 2560 + s1 * 2561 + s2 * 2562 + s3 * 2563) .... = 全帧的JPEG(full JPEG frame) **注意：**如果机器人摄像头繁忙，有时发送的指令'I'会没有返回，所以需要尽可能多的发送指令'I'直到有图像返回。
'irab'	'##iraa cc'	I2C寄存器读取(参数'ab'以8位二进制发送) a是设备id，b是寄存器，cc是一个以十进制值显示的8位寄存器返回值
'iRab'	'##iRaa cc'	I2C寄存器读取(参数'ab'以8位二进制发送) a是设备id，b是寄存器，cc是一个以十进制值显示的16位寄存器返回值
'iMabc'	'##iMaa xx xx ... xx'	I2C多寄存器读取(参数'abc'以8位二进制发送) a是设备id，b是寄存器，c是寄存器的个数，xx是一个以十进制值显示的8位寄存器返回值
'iwabc'	'##iwaa'	I2C寄存器写入(参数'abc'以8位二进制发送) a是设备id，b是寄存器，c是写入到寄存器的值
'iWabcd'	'##iWaa'	I2C寄存器多字节写入(参数'abcd'以8位二进制发送) a是设备id，b是第一个字节，c是第二个字节，d是第三个字节
'idabcef'	'##idaa'	I2C双寄存器多字节写入(参数'abcde'以8位二进制发送) a是设备id，b是寄存器1，c是写入到寄存器1的值，d是寄存器2，e是写入到寄存器2的值
'l'	'#l'	打开激光器
'L'	'#L'	关闭激光器
'Mabc'	'#M'	直接电机控制 参数'abc'以8位二进制发送 a = 左速度，b = 右速度，c = 持续时间*10milliseconds(毫秒) 速 度是8位二进制2的补码值(2's complement 8-bit binary values)，如0x00到0x7F是向前，0xFF到0x81是倒退。与4字节序列等价的十进制：0x4D 0x32 0xCE 0x14 = 'M' 50 -50 20 (表示右旋转200ms) 持续时间设为00表示没有限制，如4字节序列：0x4D 0x32 0x32 0x00 = M 50 50 00 以50%的速度向前无限行驶(drive forward at 50% indefinitely)
'mabc'	'#m'	对第二组定时器(TMR6 & TMR7)采用直接PWM电机控制，与指令'M'的格式相同。

指令	返回	描述
'o'	'#o'	开启图像叠加
'O'	'#O'	关闭图像叠加
'p'	'##ping xxxx xxxx xxxx xxxx\r\n'	ping 引脚 27，28，29，30 与引脚 18 上的开关相连接的超声波测距模块，已与 Maxbotics EZ0 和 EZ1 模块进行过测试。xxxx 的返回值范围是：inches * 100 (2500=25 英寸)
'qx'	'##quality x\r\n'	设置 JPEG 的质量，范围是 1~8('x'是一十进制 ASCII 字符)，1 的图片质量最高，8 最低。一般系统默认为 4。
'R'	'##Range(cm) = xxx'	使用激光点测量最近障碍物的距离
'r'	'##Range(cm)'	同上面的'R'指令一样，都是通过激光测距。但会有更多的诊断信息输出。
'Sab'	'#S'	直流电机(servo)控制(TMR2 & TMR3) 参数'ab'以 8 位二进制发送：a = 左伺服设置(0x00-0x64)，b = 右伺服设置(0x00-0x64) 伺服设置的值都是 8 位二进制，表示定时脉冲的宽度范围是 1 到 2 毫秒之间。 0x00 相当于 1 毫秒脉冲，0x64 相当于 2 毫秒脉冲，0x32 当然就是 1.5 毫秒脉冲了。
'sab'	'#s'	第二组直流电机控制(TMR6 & TMR7) 参数'ab'以 8 位二进制发送：a = 左伺服设置(0x00-0x64)，b = 右伺服设置(0x00-0x64) 伺服设置的值都是 8 位二进制，表示定时脉冲的宽度范围是 1 到 2 毫秒之间。 0x00 相当于 1 毫秒脉冲，0x64 相当于 2 毫秒脉冲，0x32 当然就是 1.5 毫秒脉冲了。
't'	'##time - millisecs: xxxx\r\n'	输出机器人重置后到现在的运行时间，毫秒表示
'Tx'	'#T'	设置'g2'边缘检测的阈值，范围是'T1'-'T9'，默认是'T4'。
'V'	'##Version ...\r\n'	读取固件的版本信息，遇到换行符则终止响应
'X'	'#Xmodem transfer count: bytes'	XModem 协议：1K 的文件传输，通过 XModem 协议接收文件，并存储在 Flash 缓冲区。
'y'	'#y'	翻转视频采集(一般配合倒置相机使用)
'Y'	'#Y'	恢复视频采集到正确方向
'zAxx'	'##zA - read 131072 bytes\r\n'	大块读取 Flash 内存：从指定 Flash 扇区对(sector pair) xx 02/03 ... 62/63 中读取 128kb 到 Flash 缓冲区(如在执行 C 解释器之前需要先从 Flash 扇区读取 C 程序)。扇区 00 和 01 是禁止使用的。
'zBxx'	'##zB - wrote 131072 bytes\r\n'	大块写入 Flash 内存：将 128kb 从 Flash 缓冲区写入到指定 Flash 扇区的缓冲区对 02/03 ... 62/63(扇区 00 和 01 是禁止使用的)
'zc'	'##zclear\r\n'	清空 Flash 缓冲区内容
'zC'	'##zCRC xxxx\r\n'	计算 Flash 缓冲区的 crc16_ccitt(一种校验算法)
'zd'	'##zd...\r\n'	转储(dump)Flash 缓冲区：将 Flash 内存缓冲区的内容转储到控制台

指令	返回	描述
'zr'	'##zr\r\n'	读取 Flash 内存：从用户 Flash 扇区读取 65kb 到 Flash 缓冲区(如在执行 C 解释器之前需要先从 Flash 扇区读取 C 程序)
'zRxx'	'##zRead\r\n'	读取 Flash 内存：从指定 Flash 扇区 xx(02 - 63)读取 65kb 到 Flash 缓冲区(如在执行 C 解释器之前需要先从 Flash 扇区读取 C 程序)。扇区 00 和 01 是禁止使用的。
'zw'	'##zw\r\n'	写入 Flash 内存：将 65kb 从 Flash 缓冲区写入到用户 Flash 扇区
'zWxx'	'##zWxx\r\n'	写入 Flash 内存：将 65kb 从 Flash 缓冲区写入到指定 Flash 扇区 02-63(扇区 00 和 01 是禁止使用的)
'zZ'	'##zZ\r\n'	更新 Flash 内存的引导扇区：将 Flash 缓冲区的内容写入到 Flash 内存的引导扇区，一般用于更新 u-boot.ldr 或 srv1.ldr，会首先检查一个有效的 LDR 的格式图像在 Flash 缓冲区
特殊指令：		
'$!'	-	重置 Blackfin
'$E'	-	读取 GPIO-H14 和 H15 中的可选轮编码器(optional wheel encoders)
'$ex'	-	x = 电机#(1-4)；从四驱 SRV 机器人的轮编码器(wheel encoders)中读取累计的脉冲数
'$g'	-	解析 GPS 输入
'$R'	-	SVS(立体视觉机器人，有两个摄像头系统搭配两个处理器，一个是主处理器，另一个是从处理器)控制指令：配置 Blackfin 从处理器接收通过 SPI 传输到 Flash 缓冲区的内容(to receive SPI transfer to flash buffer)
'$X'	-	SVS 控制指令：配置 Blackfin 主处理器通过 SPI 传输 Flash 缓冲区的内容
'$Axx'	'##$Axx yyyy\r\n'	读取 AD7998 的模数转换通道(A/D channel) xx(01-08, 11-18 or 21-28)
'$C'	'##$C xxx\r\n'	读取 HMC6352 罗盘(compass)
'$c'	'##c heading=344 x=-505 y=-110 z=447 xmin=-1032 xmax=-228 ymin=-970 ymax=-89\r\n'	读取 HMC5843 罗盘(compass)
'$Ta'	'##$Tx yyyy\r\n'	读取 LIS3LV02DQ 倾斜传感器 a 从指定坐标轴中读取的坐标值。取值范围是 1,2,3, 分别代表 x,y,z 坐标轴。
视觉指令： 所有参数都是以十进制 ASCII 字符发送的('0'-'9')		
'g0'	'##g0'	获取参考帧并启用差分帧(grab reference frame and enable frame differencing)
'g1'	'##g1'	启用颜色分割(color segmentation)
'g2'	'##g2'	启用边缘检测(edge detection，通过指令'T'更改阈值)
'g3'	'##g3'	启用地平线检测(horizon detection，通过指令'T'更改阈值)
'g4'	'##g4'	启用障碍物检测(obstacle detection，通过指令'T'更改阈值)

指令	返回	描述
'g5'	'##g5'	启用立体视觉处理(仅仅工作在通过 GPIO-H8 连接两个处理器的 SVS 系统上)

'g6x'	'##g6 bin# x'	将匹配 color bin #x 的粒子(blob,图像处理软件词汇)搜索结果在图像上突显出来。可以理解为在图像上搜索指定颜色的粒子，并动态突出显示。color bin #x 的是通过指定 YUV(一种颜色编码方法)的值进行定义的，如红色可以定义为：
		ymin[ix]=075;
		ymax[ix]=225;
		umin[ix]=100;
		umax[ix]=150;
		vmin[ix]=175;
		vmax[ix]=250;
		然后就可以通过 SRV1Test 客户端测试：首先通过 SRV1Test 客户端连接机器人，接着发送指令'vc3075225100150175250'，指令'vc'介绍见后面，'3'表示定义 color bin #3，'075225100150175250'就是上面关于红色的定义了；然后再发送指令'g63'，就会看到图像上红色区域被突显出来了。

'g_'	'#g_'	关闭差分帧，颜色分割和边缘检测。'_'可以是除了 0，1，2，3，4 之外的人意字符
'vax'	'##vax\r\n'	指令'va'用来开启/关闭自动增益，白平衡以及曝光相机等功能(默认 x=7) x=4 -> AGC enable x=2 -> AWB enable x=1 -> AEC enable x=7 -> AGC+AWB+AEC on x=0 -> AGC+AWB+AEC off
'vbc'	'##vbc dd\r\n ssss x1 x2 y1 y2\r\n ....'	指令'vb'用来搜索图像中与 color bin #c 相匹配的粒子(blob,图像处理软件词汇；但并不突显)，dd 是指相匹配的粒子数，同时返回每个粒子中匹配的像素量，以及包含该粒子的矩形区域坐标 x1 x2 y1 y2。最多可以支持返回达 16 个粒子的信息，各个粒子的信息按照像素量从大到小依次输出。当然了，对于像素量小于 MIN_BLOB_SIZE(当前定义为 5)的粒子是不显示的。
'vccy1y2u1u2v1v2'	'##vcc\r\n'	指令'vc'用来直接设置 color bin #c 的内容(即颜色定义)。该指令的返回字符串是'vc'跟上 color bin 号码(即#c)。 例如，我们可以对 color bin #c 存储一组颜色(在其它时间测量得到的对应数值)，就拿前面所提到的橙色高尔夫球的颜色，可以表示为'vc3127176086111154200'。然后我们就可以通过发送指令'vr3'对 color bin #3 的内容进行检索以证实色彩已被妥善的保存。关于指令'vr'请参考后面。
'vfxxx1xxx2yyy1yyy2'	'##vf xxxx\r\n'	指令'vf'用来统计一个矩形区域(坐标为 x1, x2, y1, y2)内匹配 color bin #x 的像素数。 例如，'vf10100020001500220'表示搜索 x 轴范围为 100-200，y 轴范围为 150-220 这个区域内匹配 color bin #1 的像素数。指令中的数字可以分成这样几组： 1，100，200，150，220，分别指向#1，x1, x2, y1, y2，这几

指令	返回	描述
		组数据都是通过'0'分隔。
'vh'	'##vhist y u v\r\n'	计算并列出整个图像区域里的Y，U和V像素分布，被分成0-3, 4-7, 8-11, ... 248-251, 252-255这些区间。
'vm'	'##vmean yy uu vv\r\n'	计算整个图像区域里的Y，U和V的平均值。
'vpxxxxyyyy'	'##vp yyy uuu vvv\r\n'	指令 'vp'用来提取一个由坐标xxxx(根据分辨率xxxx的取值范围有：0000-0159, 0000-0319, 0000-0639, 0000-1279)和yyyy(yyyy的取值范围有：0000-0127, 0000-0255, 0000-0511, 0000-1023, 其中0000指图像的顶端)定义的单像素。例如，'vp01600128'将提取一幅分辨率为320x256图像正中间的一个像素。 'vp01600000'将提取该图片第一行正中的一个像素。
'vrc'	'##vrc y1 y2 u1 u2 v1 v2\r\n'	指令 'vr'用来检索color bin #c中存储的颜色信息。该指令的返回字符串是'vr'跟上color bin号码(即#c)，再跟上y1=Ymin, y2=Ymax, u1=Umin, u2=Umax, v1=Vmin, v2=Vmax. 可以参考上面关于橙色高尔夫球的颜色示例。
'vsx'	'##vscan = pix xxxx xxxx xxxx xxxx ... xxxx\r\n'	指令'vs'使用由指令'T'或'vt'设置的"edge_thresh"对图像的x(1-9)列进行边缘像素扫描。返回总的边缘像素数目，以及每一列中从底部到第一个边缘像素的距离。
'vtxxx'	'##vthresh xxxx\r\n'	为边缘检测设置全局变量"edge_thresh"(范围是0000-9999，默认是3200)，等价于'T'的控制台功能，但要更精确。
'vzx'	'##vzero\r\n'	指令'vz0'将所有的颜色区间校正归零。vz1 / vz2 / vz3 / vz4 会将颜色分割成各种不同色彩空间，它们可以用来开启指令'g1'的颜色分割功能。(指令'G'关闭该功能))
神经网络指令： 所有参数都是以十六进制ASCII字符发送的('0'-'f')		
'np'	-	存储一个新模式(pattern)
'nd'	-	显示一个已存的模式
'ni'	-	使用随机权重对网络进行初始化
'nt'	-	通过已存的模式对网络进行训练
'nx'	-	使用示例模式对网络进行测试
'ng'	-	使用通过指令'vb'搜索到的粒子(blob,图像处理软件词汇)来获取一个模式(grab a pattern using blob located by "vb")
'nb'	-	针对通过指令'vb'搜索到的特定粒子进行模式匹配
C解释器：		
'Q'	执行C程序	执行存储在Flash缓冲区中的C程序。可以通过以下方式将 C程序传输到Flash缓冲区：通过指令'E'开启行编辑器直接编辑；通过指令'zr'或'zRxx'将程序从Flash扇区读取到Flash缓冲区；通 过指令'X'使用Xmodem协议将C程序文件传输到Flash缓冲区
'!'	启动C交互程序	通过ESC退出交互程序(关于更多交互命令，请参考picoC文档)