

SRV-1 机器人开发手册

—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

SRV-1 机器人开发手册.....	1
一、适用于 SRV-1/SRV-A/UCR-3 系列机器人产品.....	1
开始之前.....	3
一、安装交叉编译工具链.....	3
二、编译固件源代码.....	4
三、更新固件到 SRV-1 机器人.....	4
四、添加并测速新的协议指令.....	8
五、追球程序开发.....	9
六、添加新的传感器支持.....	13
七、客户端改进升级.....	14
附录一：Matchport 无线模块完整设置 Log.....	14
附录二：从 Adhoc 模式更改成为 Infrastructure 模式.....	16
附录三：集群模式设置.....	18
附录四：如何修改默认采集图像的分辨率(出厂默认是 320x240).....	20
附录五：忘记了 SRV-1 机器人的 IP 地址怎么办？.....	21
附录六：如何通过 webcamsat 进行网络控制.....	21
附录七：Windows 上编译固件时提示：bfin-elf-ld：xxx No such file Or file not found.....	22

开始之前

1. SRV-1 能做什么？

- 图像采集
- 安全监控
- 野外勘探
- 产品原型
- 人工智能
- 激光测距
- 立体成像
- 模块扩展

2. 用户能够学到什么？

- 完善并加强自己专业知识的实践能力
- 了解开源软件开发流程
- 熟悉 Linux 开发环境（支持 M\$ 但力荐 Linux）
- 了解 Blackfin 架构体系
- 提升编程等实际动手能力

一、安装交叉编译工具链

交叉编译工具链下载（包括 Linux 版和 Windows 版，以及 64 位版本）：www.ucrobotics.com/index.php/resource

Linux 环境交叉编译工具链包括三个 blackfin-toolchain-*前缀的 tar 包，下载后安装步骤如下：

1. 解压工具包：

需要使用 root 权限解压：

```
sudo tar xjf blackfin-toolchain-*.tar.bz2 -C /
```

解压完成之后，你会发现在/opt 目录下多了个 uClinux 目录，这就是 toolchain 的安装文件。

注意：如果上面通配符的方式解压出错，那就一个一个解压好了，就是指定完整的包名。然后检查下/opt 及 /opt/uClinux 两目录的权限。如果不是 root 的话需要改成 root，并设置为 755 权限：

```
sudo chown -R root.root opt
sudo chmod -R 755 /opt/uClinux
```

2. 设置\$PATH 环境变量：

编辑你的 ~/.bashrc 文件，在文件的最后添加：

```
export PATH=$PATH:/opt/uClinux/bfin-elf/bin:/opt/uClinux/bfin-uclinux/bin
```

然后执行：

```
$ source ~/.bashrc
```

即可。

3. 验证安装是否正确

在终端上输入 "bfin-"，然后点击 Tab 键进行补全，如果出现很多 bfin-elf*和 bfin-uclinux*命令，则表明你可以正常使用

用 toolchain 进行编译了。

Windows 环境交叉编译工具链就是一个 exe 文件，点击双击安装：

如果安装过程中看到如下提示信息：

```
-----  
ADI Blackfin Toolchain 2009R1.1 Setup  
-----  
Your system does not appear to have LibUsb-Win32 installed.  
You need to have this installed if you wish to use USB based JTAG tools.  
Do you wish to install LibUsb-Win32?  
-----  
Yes No  
-----
```

一定要记得选择 "No"。因为 LibUsb-Win32 这个库并不是必须的，安装后反而会对你系统的 USB 造成损害！

二、编译固件源代码

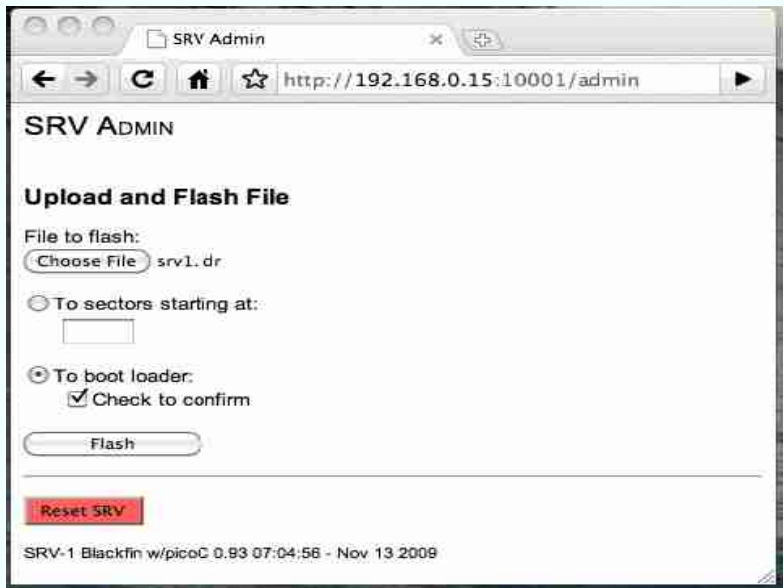
进入源代码目录后执行编译命令：

```
$ make clean  
$ make
```

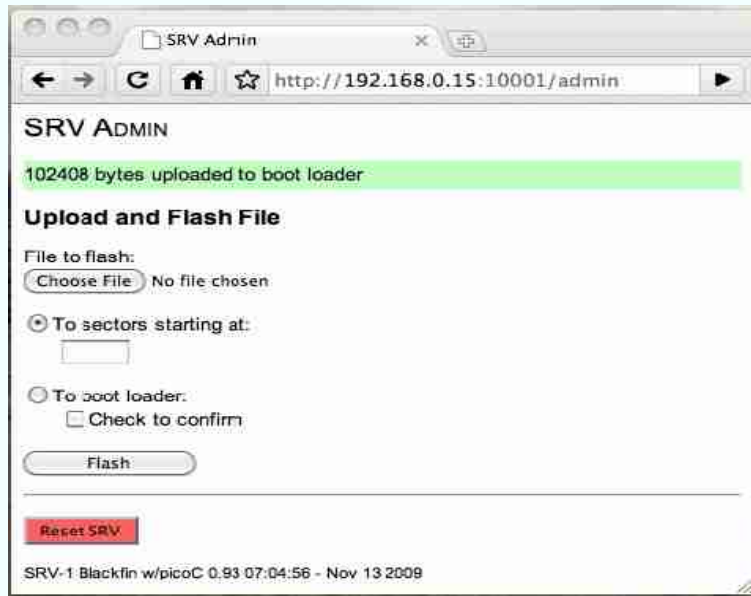
最后生成的目标文件是 srv1.ldr，其中还有一个 srv1.ldr.recovery 恢复文件记得不要删除了，在恢复烧写的时候需要用到。

三、更新固件到 SRV-1 机器人

可以通过 Web 管理界面进行升级，操作步骤：先启动机器人，然后确保 PC 能 ping 通机器人，然后通过浏览器访问：<http://169.254.0.10:10001/admin>，选择最新编译的 srv1.ldr 固件程序，并选中页面上的 "To boot loader:" 及其子选项 "Check to confirm"，如下图所示：



然后点击 "Flash" 按键上传即可。如果看到类似 "xxx bytes uploaded to boot loader" 的提示信息，则表示 Firmware 更新成功，如下图所示：



如果更改后的固件运行出现异常，会导致 `http://169.254.0.10:10001/admin` 这个页面无法访问，这时只能通过命令行进行故障恢复更新了。

故障恢复烧写步骤：

1. 移除 J1 上 7, 8 引脚的跳线(板子右侧倒数第四个)，这样就把 blackfin 主板跳线成“UART”启动模式(正常启动模式为 SPI Flash 启动模式)。
2. 设置 Matchport 无线模块的波特率

先通过终端 telnet 到 169.254.0.10 的 9999 端口 (Windows 下通过“cmd”运行)：

```
telnet 169.254.0.10 9999
```

会看到 0~9 共十个配置项，选择配置项 1，将波特率(Baudrate)更改成：115200，流控(flow control)设置为：0

```
Change Setup:
0 Server
1 Channel 1
2 Channel 2
3 E-mail
4 WLAN
5 Expert
6 Security
7 Defaults
8 Exit without save
9 Save and exit Your choice ? 1

Channel 1 Serial (1) -
for Baudrate, enter 115200
for flow control, enter 0 (this disables hardware flow control)
skip the rest of the options
Save and Exit (9)
```

然后选择配置项 9 保存退出。

注意：下面的 3、4 步会区分 Linux 及 Windows 环境下的操作

3. 烧写 `srv1.ldr.recovery` 恢复固件文件：

首先应进入 srv 目录，然后执行：

Linux 系统：

```
$ bfin-uclinux-ldr -l -v srv1.ldr.recovery 169.254.0.10:10001
```

Windows 系统：

```
> ldr -l -v srv1.ldr.recovery 169.254.0.10:10001
```

烧写的 Log 看起来如下：

```
> Loading LDR srv1.ldr.recovery ... auto detected LDR as 'BF537'  
> OK!  
> Connecting to remote target '169.254.0.x' on port '10001' ... OK!  
> Trying to send autobaud ... OK!  
> Trying to read autobaud ... OK!  
> Checking autobaud ... OK!  
> Autobaud result: 50bps 0.19mhz (header:0xBF DLL:0x18 DLH:0x00 fin:0x00)  
> Sending blocks of DXE 1 ... [10:6208 bytes] [10:7400 bytes] [10:19448> bytes] [3/OK!  
> You may want to run minicom or kermit now  
> Quick tip: run 'ldr <ldr> <tty> && minicom'
```

表明烧写完成！

4) 烧写正式的 srv1.ldr 固件文件：

Linux 系统下：

可以通过执行 nc 命令使用 V 查看版本信息，此时的恢复固件只是运行在内存中，你还需要再次烧写正式固件并存储到 SPI Flash 介质中。

首先通过 X 指令设置 XMODEM 传输协议：

```
$ nc 169.254.0.10 10001  
V  
##Version - SRV-1 Blackfin 17:46:45 - Jun 16 2010  
X  
CCCC
```

看到几个“CCCC”字符出现之后通过 Ctrl+C 退出 nc 命令即可。

然后执行烧写命令：

```
sz -Xbkv --tcp-client 169.254.0.10:10001 srv1.ldr
```

注意：有的 Linux 发行版使用的是“lsz”命令！

然后通过执行“zZ”命令将固件从内存写入到 SPI Flash：

```
$ nc 169.254.0.x 10001  
zZ
```

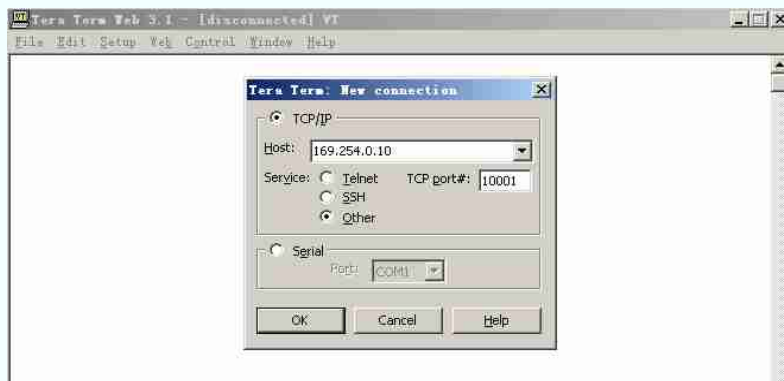
如果看到字样：

```
"##zZ boot image write count: 131072"
```

表明烧写完成。

Windows 系统下：

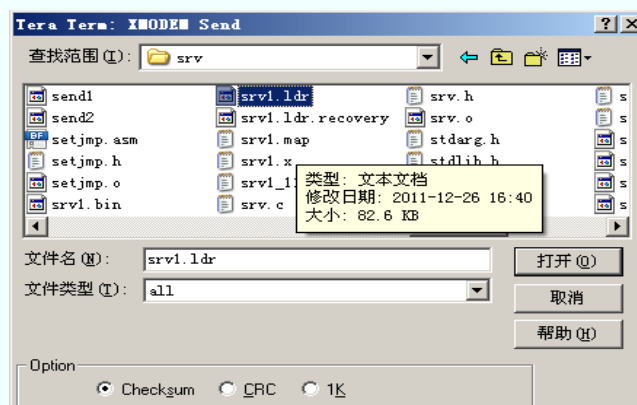
解压并进入 TTPRO 程序目录，双击 ttermpro.exe



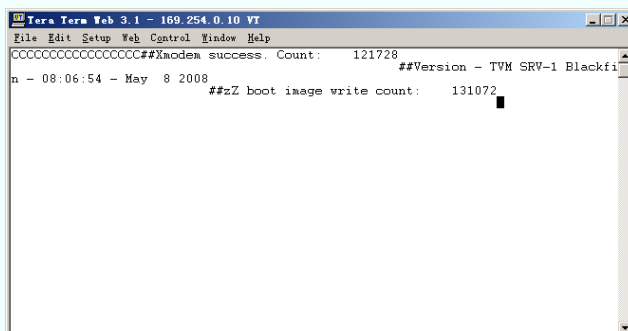
选中“other”选项，“TCP port#”设置为 10001，然后单击 OK



写入大写的 X，然后单击菜单 file→ transfer xmodem send，选中 srv1.ldr 文件开始烧写。



然后执行命令 zZ 将烧写到内存中的固件写入到 SPI Flash :



5) 更改 Matchport 无线模块的波特率到 2.5M, 同样通过 telnet 到 9999 端口的方式进行更改(波特率设置成: -1, 流控选项设置成: 2, 然后选择 9 保存退出):

```
Change Setup:
0 Server
1 Channel 1
2 Channel 2
3 E-mail
4 WLAN
5 Expert
6 Security
7 Defaults
8 Exit without save
9 Save and exit Your choice ? 1

Channel 1 Serial (1) -
for Baudrate, enter -1
for flow control, enter 2
skip the rest of the options

Save and exit (9)
```

注意: 此处的波特率 -1 即表示设置成 2.5M。

6) 重新接上 7, 8 引脚跳线, 把启动模式从 “UART” 换成 “SPI”, 然后重启机器人。至此故障恢复烧写完成。

四、添加并测速新的协议指令

通过前面的介绍, 应该已经对 SRV-1 的使用有了初步的认识。如果要做更深入的了解或是二次开发, 则需要熟悉其软件的结构设计了。

入口函数为 main.c 里面的 main() 函数, 里面执行一个 while() 大循环语句, 不断的监听客户端发来的请求。

每一个 case 语句代表一条控制指令, 所以要想了解 SRV-1 的某项功能, 只需先找到该功能的触发指令, 然后再追踪到对应的实现函数中去研究即可。

稍带提一下 config.h 文件, 里面有关于串口波特率的定义, 如:

```
#define UART0_BAUDRATE 2304000
```

这与前面固件烧写过程中最后配置的 2.5M 速率是对应的。

还有一个 myfunc.c 文件，这是一个留给用户来开发自己代码的文件。可以看一下 main.c 里面语句：

Case '%':

这个就是用来启动用户自己协议指令的入口。所以如果用户在 myfunc.c 文件里面添加了自己的协议指令，需要通过发送组合字符串方式执行：%x。

下面来简要介绍一下如何添加自己的控制指令：

打开 myfunc.c 文件，在里面添加如下内容(其实固件里面已经默认添加了这段内容)：

```
#include "myfunc.h"
#include "print.h"

void myfunc() {
    unsigned char ch;
    ch = getch();
    switch (ch) {
        case 'h': // test myfunc - string is "%h"
            printf("myfunc: hello !\r\n");
            break;
    }
    return;
}
```

可以看到，上面的语句实际是添加了一个新的'h'指令。交叉编译并更新到 SRV-1 机器人后，可以通过终端发送指令 '%h' 来验证该指令的输出：

```
$ nc 169.254.0.10 10001
#?V
##Version - SRV-1 Blackfin w/PicoC v2.1 built:16:25:40 - Feb 22 2013
##Version - uCRobotics Control System - 2012-11-18
#?
#?h
#?%h
myfunc: hello !
#?
```

说明一下：由于 SRV-1 固件支持的协议指令比较多，单独一个字符的协议指令在 main() 函数里面基本用完了。所以推荐用户开发自己的指令时使用组合的方式，如上面的 '%h' 就是一个二位组合，当然还可以三位、四位（参见'M'指令）或是更复杂的组合。

五、追球程序开发

下面将展示一个 SRV-1 小车追球程序的开发过程。

其大致思路是：每秒采集 10 帧图片，然后根据定义的色彩空间（YUV min 和 YUV max）进行判断，如果包含色彩空间定义的颜色，则表明发现目标。

由于只需 10 帧的速率，所以 matchport 无线网卡的波特率设置成 921600 即可。

然后依据以下规则控制机器人运动：

- 如果目标在机器人视野（视觉）的左侧，则左转；
- 在右侧则右转；
- 如果目标在视野的中间地带，则追逐小球而去……

因为默认采集的图像尺寸是 320x240，所有这里规定：

- 0 ~ 120 像素范围为视野左侧
- 120 ~ 200 像素范围为视野中间地带

- 200 ~ 320 像素范围为视野右侧

当然了，如果没发现目标，则只能原地左转不断找寻了。

关于 YUV 色彩的定义的说明：YUV 是西方常用的一种颜色编码方法，类似于 RGB 表示，但 YUV 定义的是一个色彩区间，即只要在定义的区间内，都可以认为是预定义的颜色。如红色可以定义如下：

```
ymin[ix]=075;
ymax[ix]=225;
umin[ix]=100;
umax[ix]=150;
vmin[ix]=175;
vmax[ix]=250;
```

如果不知道如何定义所需的颜色，可访问 YUV 与 RGB 互相转换链接：

http://www.ucrobotics.com/downloads/english_docs/YUV_RGB/YUV_RGB.html

具体更改如下：

1. 编辑 config.h 文件，将串口 0 的波特率改成 921600：
2. 在 srv.h 里面定义找球函数
3. 在 srv.c 里面实现找球函数
4. 在 main.c 里面调用找球功能，支持开机即启动找球程序

详细代码补丁如下：

```
diff -Nur srv1_bf53x/config.h srv1_bf53x_ball/config.h
--- srv1_bf53x/config.h 2012-11-21 11:35:49.562132777 +0800
+++ srv1_bf53x_ball/config.h 2012-11-28 17:08:03.489234610 +0800
@@ -26,13 +26,15 @@
 #define CORE_CLOCK (MASTER_CLOCK * VCO_MULTIPLIER / CCLK_DIVIDER)
 #define PERIPHERAL_CLOCK (CORE_CLOCK / SCLK_DIVIDER)
+
+/*
 // UART config
 #if defined(_SRV_UART0_BAUDRATE_115200)
 #define UART0_BAUDRATE 115200
 #else
 #define UART0_BAUDRATE 2304000
 #endif
+*/
+#define UART0_BAUDRATE 921600
 #define UART1_BAUDRATE 115200

 // must be power of 2!
diff -Nur srv1_bf53x/main.c srv1_bf53x_ball/main.c
--- srv1_bf53x/main.c 2012-11-21 11:35:49.546132698 +0800
+++ srv1_bf53x_ball/main.c 2012-11-29 16:44:35.469507953 +0800
@@ -56,10 +56,21 @@
 serial_out_version();
 check_for_autorun();
 #endif /* STEREO */
+
+ /**
+ * 追球函数，开机即会运行，如果要中止该函数，需要连接到 SRV-1 串口（如'nc robot-ip 10001'），
+ * 然后随便回车一下或是发送个字符即可中止
+ */
+ track_ball();
 while (1) {
 if (getchar(&ch)) {
 switch (ch) {
 case 'B':
```

```

+         track_ball();
+         case 'C':
+             //setPWM(20, 50);
+             follow_ball(-70, 70, 20);
+             case 'T':
+                 grab_frame();
+                 send_frame();
diff -Nur srv1_bf53x/srv.c srv1_bf53x_ball/srv.c
--- srv1_bf53x/srv.c 2012-11-21 11:35:49.542132685 +0800
+++ srv1_bf53x_ball/srv.c 2012-11-29 16:34:31.198511535 +0800
@@ -2370,3 +2370,97 @@
     return (ch1*1000 + ch2*100 + ch3*10 + ch4);
}

+/**=====track ball function =====
+ * track_ball(): 这里的循环条件是串口没有字符输入;
+ * 所以如果要中止机器人寻球程序, 只需给串口随意发送一个字符即可。机器人
+ * 先朝前看, 没有发现目标, 则左转, 如果依然没有发现目标,
+ * 则继续左转, 循环下去; 一旦发现目标, 调用 find_ball()函数
+ */
+int x_position=0, y_position=0;
+void track_ball(){
+    unsigned char ch;
+    delayUS(12000000/UART0_BAUDRATE); // flush recv buffer
+    while (uart0GetChar(&ch))
+        delayUS(12000000/UART0_BAUDRATE); // allow enough time for characters to flow in
+    while(!uart0GetChar(&ch)){
+        find_ball();
+    }
+}
+
+/**
+ * 控制小车运动函数, 三个参数表示: 左马达速度, 右马达速度, 延续时间
+ */
+void follow_ball(int ls, int rs, int ms) { //left motor speed, right motor speed, delay mseconds
+    if (!pwm1_init) {
+        initPWM();
+        pwm1_init = 1;
+        pwm1_mode = PWM_PWM;
+        base_speed = 70;
+        lspeed = rspeed = 0;
+    }
+    setPWM(ls, rs);
+    if (ms) {
+        delayMS(ms * 10);
+        setPWM(0, 0);
+    }
+}
+
+/**
+ * findout_ball(): 找球的核心函数, 并执行相关动作
+ *
+ * 它先用 YUV 值来定义一种目标颜色 (范围), 这里定义了红色。它会拍 10 帧图像,
+ * 只要其中有一帧发现了红色目标, 则表明找到, 并记录下该目标的中心点坐标
+ *
+ * 如果发现目标, 则判断目标的位置: 如果偏左, 则左转; 偏右则右转; 如果在中间地带, 则行驶向小车
+ * 默认采集的图像是 320x240, 如果目标的中心点 X 坐标小于 120, 则认为在左边; 大于 200 在右边。
+ *
+ * 如果没有发现目标, 则一直左转下去寻找
+ */
+void find_ball(){
+    int frame_count = 10;
+    int found_count = 0;
+    unsigned int s=0; // pixels counts
+    char ball_color;
+    ball_color = 'R';
+    int ix=0;

```

```
+ if (ball_color == 'B') { //define black color
+   ymin[ix]=55;
+   ymax[ix]=65;
+   umin[ix]=175;
+   umax[ix]=185;
+   vmin[ix]=95;
+   vmax[ix]=105;
+ }
+ else if (ball_color == 'R') { //define red color
+   ymin[ix]=75;
+   ymax[ix]=225;
+   umin[ix]=85;
+   umax[ix]=150;
+   vmin[ix]=175;
+   vmax[ix]=250;
+ }
+ for (i=0; i<frame_count; i++ )
+ {
+   grab_frame();
+   vblob((unsigned char *)FRAME_BUF, (unsigned char *)FRAME_BUF3, ix);
+   if (blobcnt[0] > 0){
+     found_count++;
+     x_position = (blobx1[0] + blobx2[0]) / 2;
+     y_position = (bloby1[0] + bloby2[0]) / 2;
+     s = blobcnt[0];
+   }
+ }
+
+ if (s >= 1){ //find
+   if(x_position < 120) follow_ball(-70, 70, 8); //球在图像左边，左转
+   else if(x_position > 200) follow_ball(70, -70, 8); //球在图像右边，右转
+   else follow_ball(70, 70, 50); //追球而去
+   //return 1;
+ }
+ else{ //not find
+   follow_ball(-70, 70, 8); //左转
+   //return 0;
+ }
+}
+
diff -Nur srv1_bf53x/srv.h srv1_bf53x_ball/srv.h
--- srv1_bf53x/srv.h 2012-11-21 11:35:49.846134185 +0800
+++ srv1_bf53x_ball/srv.h 2012-11-29 11:19:20.951951862 +0800
@@ -239,5 +239,10 @@
extern int svb_disp_left, svb_disp_right, svb_steer;
extern unsigned char version_string[];

+/*kick ball */
+void track_ball();
+void find_ball();
+void follow_ball(int ls, int rs, int ms);
+
#endif
```

将上面的补丁添加进代码之后编译并上传到 SRV-1 机器人。

然后将 Matchport 无线网卡波特率设置成 921600，即与 config.h 里面定义的一致：

```
Change Setup:
0 Server
1 Channel 1
2 Channel 2
3 E-mail
4 WLAN
5 Expert
6 Security
7 Defaults
8 Exit without save
```

```
9 Save and exit Your choice ? 1
```

```
Channel 1 Serial (1) -  
for Baudrate, enter 921600  
for flow control, enter 2  
skip the rest of the options
```

```
Save and exit (9)
```

注意：如果在设置 921600 波特率时提示无法修改，则需要先进入到【5 Expert】高级设置里面将“CPU Performance”设置成高（选项值设置成 2），然后再回来设置波特率。

然后重启 SRV-1 机器人就开始执行追球行动了。

追球实录视频展示：<http://i.youku.com/ucrobotics>

六、添加新的传感器支持

该章节主要是针对 SRV-A 或 UCR-3 进行开发。其它在 SRV-1 的基础上添加各种传感器的支持也是可以的，只不过门槛比较高，所以这里引入了 Arduino 模块，可以很方便的帮助用户去开拓其自己的领域。

Blackfin 核心控制板与 Arduino 之间是通过串口 1 进行通讯的（Matchport 无线模块对应的配置项是【2 Channel 2】），波特率为 9600。所有的核心控制部分依然还是由 Blackfin 来完成，Arduino 仅用于采集各种传感器的数据。

下面将介绍如何添加一个红外避障检测传感器的支持：

1. 修改 config.h 文件，将串口 1 的波特率设置为 9600

```
#define UART1_BAUDRATE 9600
```

2. 修改 main.c 文件

在 main() 函数里面的 while 大循环中，第一段代码” if(uart1GetChar(&ch)){...}”用于处理 Arduino 端的数据返回（该段代码在经典版 SRV-1 固件中是没有的），第二段代码” if (getchar(&ch)) {...}”负责客户端的指令监听。

先在第二段代码中找个合适的位置添加如下代码：

```
case 'Z': //ir  
    uart1SendChar('Z');  
    delayMS(10);  
    break;
```

上面代码表示 Blackfin 核心板接收到指令后不做任何事情，直接转发给 Arduino 进行处理。

这里定义通过'Z'指令来请求红外避障检测传感器的状态（当然也可以在 myfunc.c 里面添加自己的指令，但因为需要与 Arduino 进行通讯，协议指令的字符太多会增加处理的复杂度，影响响应时间，所以这里直接在 while 里面操作了）。

然后在第一段代码中找个合适的位置添加返回数据处理代码：

```
case 'Z': //ir  
    uart1GetString();  
    printf("%s", str);  
    printf("\r\n");  
    break;
```

3. 修改 Arduino 控制文件 ucrduino.ino

在 setup() 函数之前定义接口：

```
int irPin = 12;
```

说明：该红外传感器就三个针脚：信号，正极，负极；这里信号接入点定义为 Arduino 的数字口针脚

在 loop() 函数中的 switch 语句中添加：

```
case 'Z': //ir
    view_irir();
```

然后在该文件的后面添加 view_irir() 函数的实现代码：

```
void view_irir(){
    String vir = "Z##IRIR";
    vir += String(digitalRead(irPin));
    vir += '@';
    delay(10); //very important
    Serial.print(vir);
    Serial.flush();
}
```

简要解释一下上面这段程序：

vir 定义的是返回字符串的前缀：必需以请求的指令 ('Z') 开头，然后返回字符串的最后必需以 '@' 结束。这些都是与 Blackfin 固件约定好的规则，不可更改。

还有一点需要注意：在返回数据之前，必需至少 delay 10 毫秒，否则结果无法正确返回。

由于机器人默认搭配的陀螺仪传感器模块的编译需要额外库支持，所以编译前需先下载：
<http://www.ucrobotics.com/index.php/resource>

然后将解压的两个目录 I2Cdev/ 和 MPU6050/ 拷贝到 Arduino IDE 工具包的 libraries/ 目录下 (Linux 系统下默认目录：/usr/share/arduino/libraries/)。这两个目录看起来如下：

```
$ ls /usr/share/arduino/libraries/I2Cdev /
I2Cdev.cpp I2Cdev.h keywords.txt
$ ls /usr/share/arduino/libraries/MPU6050 /
Examples helper_3dmath.h MPU6050.cpp MPU6050.h
```

通过 Arduino IDE 进行编译烧写 Arduino 程序时必需将机器人断电；当然如果不嫌麻烦也可以将 Arduino 板子拔下来烧写，但拔之前也必需先关闭机器人，非常不建议在带电状态下进行任何热拔插操作。

将更改过 Blackfin 固件和 ucrduino.ino 控制程序分别更新到板子后，通过控制台发送 'Z' 指令就可以获得返回了，返回字符串形如：##IRIRO 或者 Z##IRIR1（最后一位才是关键：'0' 表示前方有障碍物，'1' 表示没有）

七、客户端改进升级

我们已经提供了针对 Android，iOS，NDSL 等移动终端的控制客户端软件，另外还有第三方提供的各种 PC 版客户端。PC 版客户端实现的语言有 Java，Python，Delphi 等等。

所以这里是留给用户的发挥空间，用户可以使用自己熟悉的语言创建一个自己的客户端软件，或是改进别人的客户端。

这可是提升编程能力的大好机会，因为你是在对着一个实体在编程。

附录一：Matchport 无线模块完整设置 Log

涉及用户配置的选项包括：【0 Server】，【1 Channel 1】，【2 Channel 2】，【4 WLAN】，【5 Expert】，下面就将这个配置项的标准配置完整的列出来：

```
Change Setup:
0 Server
1 Channel 1
2 Channel 2
```

```
3 E-mail
4 WLAN
5 Expert
6 Security
7 Defaults
8 Exit without save
9 Save and exit

Your choice ? 0

Network mode: 0=Wired Only, 1=Wireless Only, 2=Bridging(One Host) (1) ?

IP Address : (192) .(168) .(001) .(089)
Set Gateway IP Address (Y) ?
Gateway IP addr (192) .(168) .(001) .(001)
Netmask: Number of Bits for Host Part (0=default) (8)
Set DNS Server (N) ?
Change telnet config password (N) ?

Your choice ? 1
Baudrate (divisor 2) ?
I/F Mode (4C) ?
Flow (02) ?
Port No (10001) ?
ConnectMode (C0) ?
Send '+++ ' in Modem Mode (Y) ?
Show IP addr after 'RING' (Y) ?
Auto increment source port (N) ?
Remote IP Address : (000) .(000) .(000) .(000)
Remote Port (0) ?
DisConnMode (00) ?
FlushMode (80) ?
Pack Cntrl (C0) ?
InterCh Time (3) ?
DisConnTime (00:00) ?:
SendChar 1 (00) ?
SendChar 2 (00) ?

Your choice ? 2
Baudrate (9600) ?
I/F Mode (4C) ?
Flow (00) ?
Port No (10002) ?
ConnectMode (C0) ?
Send '+++ ' in Modem Mode (Y) ?
Show IP addr after 'RING' (Y) ?
Auto increment source port (N) ?
Remote IP Address : (000) .(000) .(000) .(000)
Remote Port (0) ?
DisConnMode (00) ?
FlushMode (00) ?
DisConnTime (00:00) ?:
SendChar 1 (00) ?
SendChar 2 (00) ?

Your choice ? 4
Topology: 0=Infrastructure, 1=Ad-Hoc (0) ?
Network name (SSID) (matchport) ?
Security suite: 0=none, 1=WEP, 2=WPA, 3=WPA2/802.11i (0) ?
TX Data rate: 0=fixed, 1=auto fallback (1) ?
TX Data rate: 0=1, 1=2, 2=5.5, 3=11, 4=18, 5=24, 6=36, 7=54 Mbps (7) ?
Enable power management (Y) ?

Your choice ? 5
TCP Keepalive time in s (1s - 65s; 0s=disable): (45) ?
ARP Cache timeout in s (1s - 600s) : (600) ?
CPU performance (0=Regular, 1=Low, 2=High): (00) ?
Disable Monitor Mode @ bootup (N) ?
HTTP Port Number : (80) ?
SMTP Port Number : (25) ?
MTU Size (512 - 1400): (1024) ?
```

Enable alternate MAC (N) ?
Ethernet connection type: (0) ?

选项 7：恢复到出厂默认设置

选项 8：不保存退出

选项 9：保存退出

附录二：从 Adhoc 模式更改成为 Infrastructure 模式

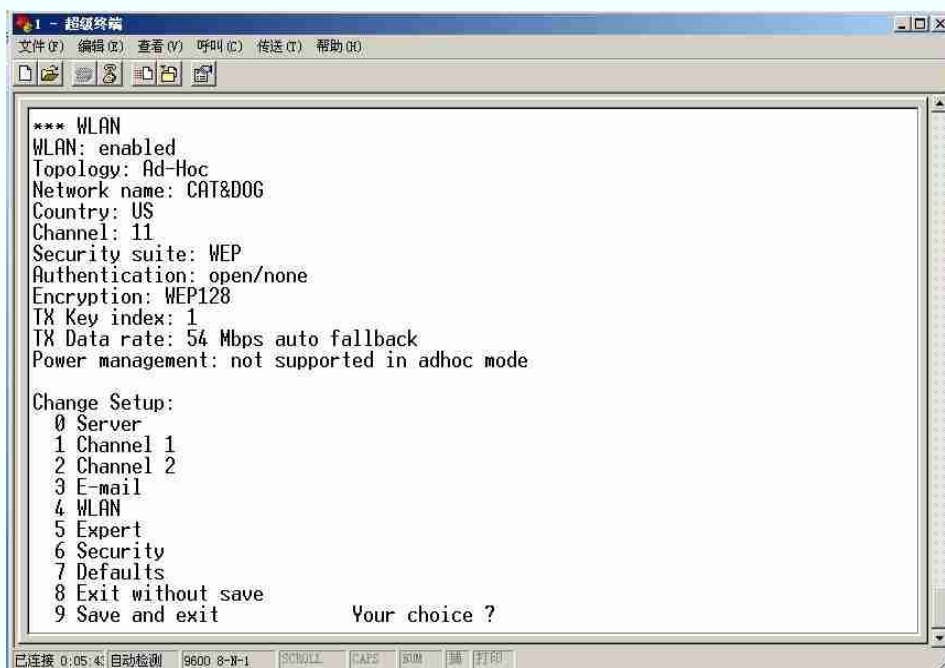
关于 Adhoc 和 Infrastructure 模式的含义，简单的来说，Adhoc 模式不依赖公共网络，可以随时随地自由组建；Infrastructure 模式可以理解为像其它终端一样接入公共网络（如公司，家庭的 WIFI）

SRV-1 产品的默认出厂设置一般都是 Adhoc 模式，IP 为 169.254.0.10；有的客户喜欢直接连接到自己的本地网络，这时就需要配置 matchport 无线网卡为 Infrastructure 模式。

首先通过 telnet 命令进入到 matchport 配置界面：

```
telnet 169.254.0.10 9999
```

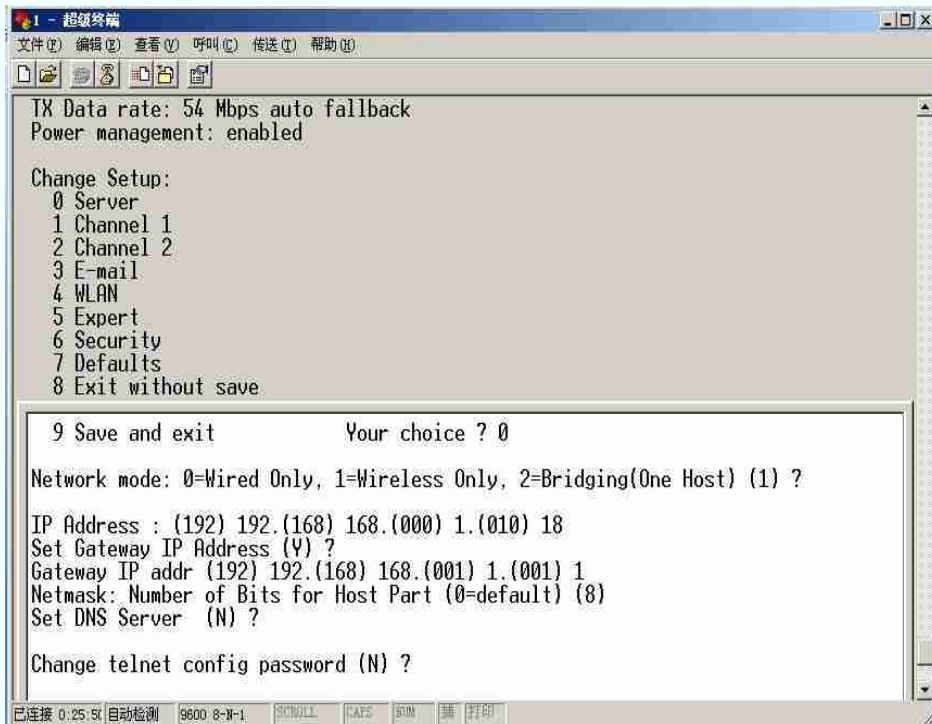
进入后会看到如下画面：



```
*** WLAN
WLAN: enabled
Topology: Ad-Hoc
Network name: CAT&DOG
Country: US
Channel: 11
Security suite: WEP
Authentication: open/none
Encryption: WEP128
TX Key index: 1
TX Data rate: 54 Mbps auto fallback
Power management: not supported in adhoc mode

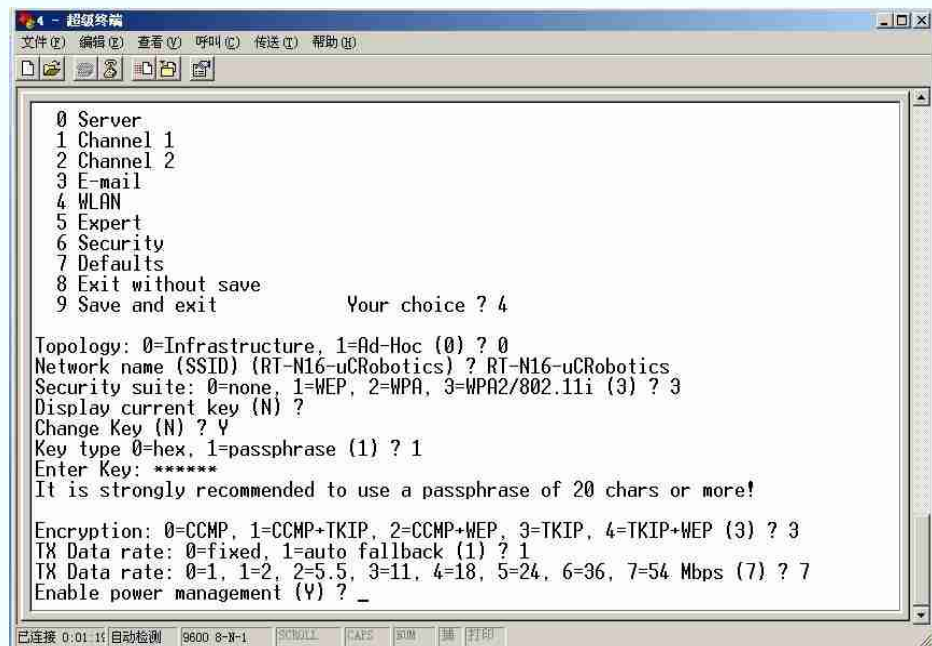
Change Setup:
0 Server
1 Channel 1
2 Channel 2
3 E-mail
4 WLAN
5 Expert
6 Security
7 Defaults
8 Exit without save
9 Save and exit          Your choice ?
```

然后选择选项【0】- Server，配置如下：



- IP Address：配置一静态 IP，一般本地使用的都是 192.168.xx.xx (以自己的实际网段为准，选择一未被使用的 IP 地址进行配置)
- Set Gateway IP Address (Y)：是否设置网关
- Gateway IP Address：配置一个网关，一般都是 192.168.xx.1

然后选择选项 [4] - WLAN，配置如下：



1. Topology：选择网络模式为 0（Infrastructure 模式）
2. Network Name(SSID)：输入要连接的网络 SSID(以自己的实际 SSID 为准)
3. Security suite：选择加密协议（以自己的实际路由设置为准，不清楚的可以通过 192.168.xx.1 登录到路由进行查看）【注：如果这里选择 0，即无加密，后面的几个选项就不会显示了】
4. Change Key (N)：是否要修改密码
5. Key Type: 密码字符类型（1 表示词组模式，易读）
6. Enter Key: ***（连接密码）
7. Encryption：加密方式（不清楚的可以通过 192.168.xx.1 登录到自己路由进行查看）

最后选择选项【9】保存退出，然后重启 SRV-1 即可。

附录三：集群模式设置

注意：集群模式设置仅对经典版 SRV-1 机器人有效，因为它需要用第二个串口（uart1）进行指令广播，而 SRV-A 与 UCR-3 上的 uart1 已经被用于与 Arduino 通讯了。

1. 功能描述

准备两台 SRV-1 机器人，对应的 IP 分别为 169.254.0.10（主机器人/master robot）和 169.254.0.11（从机器人/slave robot），通过客户端（或 nc 命令）给主机器人发送开激光指令（即指令 'I'），主机器人会同时控制从机器人点亮激光；同理发送关激光指令（即指令 'L'），从机器人也会跟着关闭。

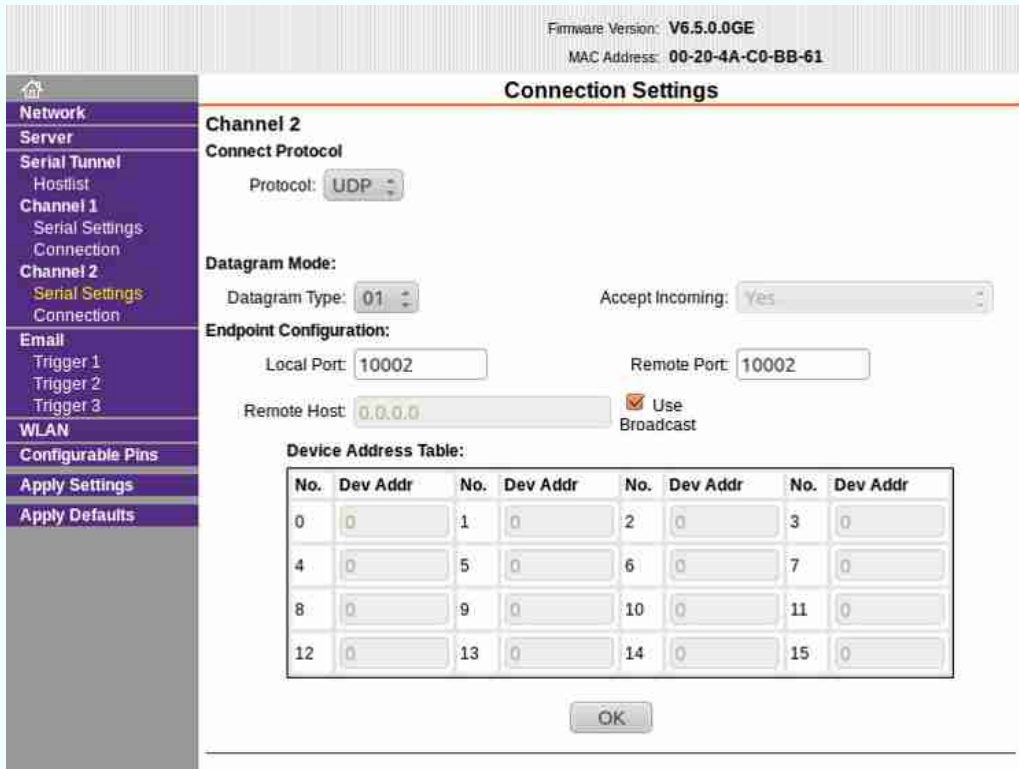
2. Matchport 设置

注意：通常情况下都推荐使用 telnet 方式进行 matchport 设置更改，但也可以通过 matchport 的配置页面进行设置：<http://169.254.0.10>

主要是对 channel 2 进行设置（主要是更改波特率和流控方式），两个 SRV-1 机器人可设置一样。关于 Channel 2 的“Serial Settings”设置如图所示：

The screenshot displays the 'Serial Settings' configuration page for 'Channel 2'. At the top, it shows the firmware version 'V6.5.0.0GE' and the MAC address '00-20-4A-C0-BB-61'. The left sidebar contains a navigation menu with options like Network, Server, Serial Tunnel, Hostlist, Channel 1, Channel 2, Email, WLAN, Configurable Pins, Apply Settings, and Apply Defaults. The main content area is titled 'Serial Settings' and includes a 'Channel 2' section with a 'Disable Serial Port' checkbox. Below this are several configuration sections: 'Port Settings' with dropdowns for Protocol (RS232), Flow Control (None), and input fields for Baud Rate (115200), Data Bits (8), Parity (None), and Stop Bits (1); 'Pack Control' with a checked 'Enable Packing' checkbox and an 'Idle Gap Time' of 12 msec; 'Match 2 Byte Sequence' with radio buttons for Yes/No and a 'Match Bytes' field set to 0x00; and 'Flush Mode' with sub-sections for 'Flush Input Buffer' and 'Flush Output Buffer', each containing 'With Active Connect', 'With Passive Connect', and 'At Time of Disconnect' options, all currently set to 'No'. An 'OK' button is located at the bottom of the configuration area.

注意：上面设置的波特率与固件源码里面的定义是一致的（config.h 中的“UART1_BAUDRATE”），然后设置传输模式（UDP 传输，数据报为 01，俩端口号都是 10002，以及使用广播），“Connection”设置如图所示：



如果机器人数量比较少，也可以不使用广播，而是指定特定的 IP 地址。

3. 程序代码

在当前固件源码基础上更改即可，主要就是更改 main.c 文件。

找到“init_uart0(UART0_BAUDRATE);”然后添加 uart1 口的初始化，如下注释部分：

```
init_uart0(UART0_BAUDRATE);
init_uart1(UART1_BAUDRATE); //初始化uart1
```

然后找到 while(1) 循环，添加一段监听 uart1 端口的函数，如下注释部分：

```
while (1) {
    if (getchar(&ch)) {
        // This function does all firmware command processing
        ProcessCommands(ch);
        reset_failsafe_clock();
        delayUS(12000000/UART0_BAUDRATE); // flush recv buffer
        while (getchar(&ch))
            delayUS(12000000/UART0_BAUDRATE); // allow enough time for characters to flow in
    }

    //监听uart1的数据
    if (uart1GetChar(&ch)) {
        // This function does all firmware command processing
        _ProcessCommands(ch);
        reset_failsafe_clock();
        delayUS(12000000/UART0_BAUDRATE); // flush recv buffer
        while (uart1GetChar(&ch))
            delayUS(12000000/UART0_BAUDRATE); // allow enough time for characters to flow in
    }
    ... ..
}
```

然后添加 `_ProcessCommands()` 函数的定义（定义在 `main()` 函数之前即可）：

```
void _ProcessCommands(unsigned char CharIn)
{
    switch (CharIn) {
        case 'l': // lasers on
            lasers_on();
            break;
        case 'L': // lasers off
            lasers_off();
            break;
        default:
            printf("#?"); // unknown command
    }
}
```

找到 “`ProcessCommands()`” 函数然后添加转发语句，如下注释部分：

```
void ProcessCommands(unsigned char CharIn)
{
    unsigned char TestChar;
    unsigned char *cp;
    int ix;
    unsigned int t0, loop;
    unsigned short sx;

    //向 uart1 口广播数据，表明从客户端接收到的任何东西都广播到 uart1 口（即 10002 端口）
    //不过当前 _ProcessCommands() 里面只定义了对 "l/L" 两个命令的操作，所以对于广播出去的其它命令不会有响应
    uart1SendChar(CharIn);

    switch (CharIn) {
        ...
    }
}
```

4. 编译测试

```
$ make clean
$ make
```

然后分别烧写到两个机器人（浏览器方式烧写即可）

然后通过客户端连接到主机器人，发送开激光指令，看看两个机器人的激光是不是都已经打开了，如果是表明测试成功！

当然也可以通过 `nc` 命令连接到主机器人：

```
nc 169.254.0.10 10001
```

然后发送 'l' 和 'L' 进行测试。

再附上通过 `nc` 连接 10002 UDP 端口的命令，可用于测试机器人自身 `uart1` 口通讯情况：

```
nc -u 169.254.0.10 10002
```

附录四：如何修改默认采集图像的分辨率(出厂默认是 320x240)

一、如果是使用客户端采集图像，可直接修改客户端即可(参看控制协议指令文档)！

二、如果是通过 `Picoc` 函数采集，则需要修改 `firmware` 源代码，方法如下：

修改函数 `firmware_trunk/blackfin/srv/srv.c/camera_setup()` 里面的相关内容。

I) 默认 320x240 所涉及的代码：

```
imgWidth = 320;
imgHeight = 240;
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x21, ov7725_qvga, sizeof(ov7725_qvga)>>1, SCCB_ON);
}
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x30, ov9655_qvga, sizeof(ov9655_qvga)>>1, SCCB_ON);
}
```

II) 如要改成 160x120：

```
imgWidth = 160;
imgHeight = 120;
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x21, ov7725_qqvga, sizeof(ov7725_qqvga)>>1, SCCB_ON);
}
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x30, ov9655_qqvga, sizeof(ov9655_qqvga)>>1, SCCB_ON);
}
```

III) 如要改成 640x480：

```
imgWidth = 640;
imgHeight = 480;
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x21, ov7725_vga, sizeof(ov7725_vga)>>1, SCCB_ON);
}
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x30, ov9655_vga, sizeof(ov9655_vga)>>1, SCCB_ON);
}
```

IV) 如要改成 1280x1024 (仅支持 ov9655 型号摄像头)：

```
imgWidth = 1280;
imgHeight = 1024;
}
for (ix=0; ix<3; ix++) {
    delayMS(100);
    i2cwrite(0x30, ov9655_sxga, sizeof(ov9655_sxga)>>1, SCCB_ON);
}
```

附录五：忘记了 SRV-1 机器人的 IP 地址怎么办？

这里提供一个 Python 小工具 [ping.py](#)，通过多线程能非常快速的向一个网段的所有 IP 发送 ping 包，有回应的则为活跃 IP，然后通过浏览器访问那些活跃 IP 就会很容易找到机器人的 IP 地址了。

执行该脚本的系统需要 Python 环境，执行命令：

```
$ python ping.py
```

当前默认发送的网段是 169.254.0.x，如果需要 ping 其它网段，自己更改一下脚本即可。

附录六：如何通过 webcamsat 进行网络控制

测试 webcamsat 步骤如下：

1. 准备一个 Windows 环境
2. 正确安装 java runtime 等 Java 环境，以及 DirectX 插件
3. 确保 JavaConsole 客户端能在 Windows 环境下正常工作（图像显示，操控等）
4. 执行 JavaConsole.exe 确认连接到 SRV-1 机器人后，就可以通过 <http://localhost:8888/view> 查看 Web 版的控制界面了，Web 上的动态图像跟控制按钮都是可以正常工作的。

在启动 JavaConsole 的时候 webcamsat 就已经启动了，所以你不必再需要额外安装其它程序。

注意一定得在 windows 环境下测试 <http://localhost:8888/view>，Linux 下会因为提示缺少插件而无法显示图像，但控制按钮功能都还是正常的。

附录七：Windows 上编译固件时提示：**bfin-elf-ld : xxx No such file Or file not found**

这需要修改固件源码目录里的 Makefile 文件，

把

```
LD=bfin-elf-ld
```

改为

```
LD=bfin-elf-ld.real
```



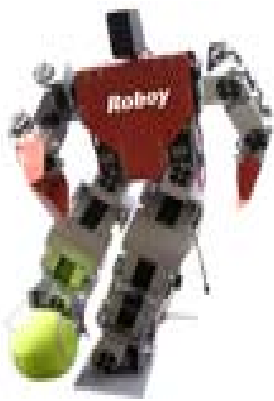
亿学通[®]

纳克斯科技公司，成立于 2006 年底（亿学通是公司产品商标），专精于高校电子竞技创新实验产品和设备的研发、生产及销售。客户遍布全国 1400 余所高校，累计数十万人次在亿学通平台上受益。



公司致力于中国教育产品领域的创新与扩展，公司倡导“创新实践、学以致用”的教育理念，以市场需求为导向、以科技创新为依托，为客户研发最有针对性的产品和设备，为客户量身定制有针对性的创新模式和解决方案。

公司具有专业的研发团队和第一流市场调研人员的。可以保证我们能够为客户提供最合适的产品。



现在公司产品共有“电动/气动机械手、履带/轮式/人形机器人、基于 DSP 的图像识别/监控设备、传感器/电子模组/机器人配件”等四大类两百多种产品。其中气动机械手、人形机器人、asuro 智能车、RP6 智能坦克车、野外侦查机器人、单片机焊接套件、电子元器件创新礼包、传感器礼包（加强版/完整版）等产品在高校得到广泛应用。

公司为高校“电子竞技创新中心”“工程训练中心”“电工电子中心”“开放实验室”“机器人创新基地”量身定制各种创新方案，满足不同类型院校的创新需求。欢迎和更多客户开展合作。

企业定位

- 亿学通从事开发、制造及销售技术新潮、形式新颖、可靠易用的高校教学产品。
- 我们的成功源自于不懈地帮助高校用户提高学生动手实践能力，提升就业竞争力。

合作院校

清华大学	北京理工大学	北京邮电大学	北京工业大学
北京化工大学	北京吉利大学	北京联合大学	北京石油化工学院
中国石油大学(北京)	中国地质大学(北京)	中国传媒大学	北京电子科技学院
南开大学	天津大学	河北工业大学	天津工业大学
天津理工大学	东北大学秦皇岛校区	防灾科技学院	河北机电职业技术学院
邯郸学院	河北建筑工程学院	河北科技大学	衡水学院
唐山学院	燕山大学	山西大学	太原理工大学
中北大学	运城学院	忻州师范学院	太原工业学院
太原科技大学	大连理工大学	东北大学	大连理工大学城市学院
辽宁工程技术大学	沈阳航空工业学院	沈阳工业大学	沈阳理工大学
长春大学	北华大学	长春工业大学	长春理工大学
东北电力大学	吉林师范大学	哈尔滨商业大学	吉林工程技术师范学院
黑龙江大学	哈尔滨工业大学	上海大学	黑龙江农业工程职业学院
南京理工大学	苏州科技大学	徐州师范大学	浙江大学
湖州职业技术学院	浙江大学城市学院	厦门理工学院	集美大学
厦门大学嘉庚学院	江西理工大学	九江职业技术学院	江西农业大学
山东大学	中国海洋大学	济南大学	山东建筑大学
山东师范大学	山东经济学院	滨州学院	德州科技职业学院
德州学院	东营职业学院	海军航空工程学院	哈尔滨工业大学(威海)
菏泽学院	山东职业学院	莱芜职业技术学院	鲁东大学
青岛大学	青岛科大大学	青岛理工大学	曲阜师范大学
山东大学威海分校	山东科技大学	山东理工大学	山东电力高等专科学校
山东农业大学	潍坊教育学院	潍坊学院	烟台大学
烟台南山学院	烟台职业学院	枣庄学院	中国石油大学(华东)
合肥学院	皖西学院	滁州学院	中国石油大学胜利学院
河南工业大学	郑州轻工	郑州航空工业	河南机电高等专科学校
河南理工大学	鹤壁职业技术学院	南阳理工大学	解放军信息工程学院
三门峡职业技术学院	焦作师范高专	黄冈职业技术学院	郑州大学西亚斯国际学院
长江大学	沙市大学	湖北汽车工业学院	十堰职业技术学院
华中师范大学	武汉大学	中南民族大学	武汉电力职业技术学院
襄樊学院	长沙大学	长沙理工大学	吉首大学
湖南工程学院	湘潭大学	湖南科技大学	永州职业技术学院
广东工业大学	佛山科学技术学院	广东技术师范学院	惠州学院
韶关学院	汕头大学	广西大学	深圳信息职业技术学院
广西工学院	广西师范学院	河池学院	广西机电职业技术学院
梧州学院	海南大学	海南大学三亚学院	广西交通职业技术学院
海南职业技术学院	重庆科技学院	四川大学	西南交大大学
成都理工大学	成都大学	内江职业技术学院	四川理工大学
西南科技大学	西南民族大学	成都信息工程学院	四川信息职业技术学院
西南石油大学	西华大学	西南林学院	四川托普职业技术学院
重庆邮电大学	贵州大学	贵阳学院	昆明师范高等专科学校
昆明理工大学	云南大学	云南大学滇池学院	昆明冶金高等专科学校
西藏大学	陕西理工学院	西安理工学院	西安航空技术高等专科学校
西北工业大学	兰州大学	西北民族大学	西北师范大学
兰州交通大学	兰州理工大学	新疆大学	兰州工业高等专科学校

(以上排名不分先后)