

智龙 V2.0 使用手册



硬件篇

1.1 龙芯 1C 芯片介绍

龙芯 1C 芯片是基于 LS232 处理器核的高性价比单芯片系统,可应用于指纹生物识别、物联传感等领域。1C 包含浮点处理单元,可以有效增强系统浮点数据处理能力。1C 的内存接口,支持多种类型的内存,允许灵活的系统设计。支持 8-bit SLC NAND 和 MLC NAND FLASH, 提供高容量的存储扩展接口。1C 为开发者提供了丰富的外设接口及片上模块,包括 Camera 控制器, USB OTG 2.0 及 USB HOST 2.0 接口, AC97/I2S 控制器, LCD 控制器, 高速 SPI 接口, 全功能 UART 接口等, 提供足够的计算能力和多应用的连接能力。



图 1-1 龙芯 1C

1.2 智龙开发板介绍

开源龙芯创客主板-“智龙”是由龙芯爱好者社区开发的一款基于国产龙芯以全开源方式推广的嵌入式最小系统主板。具有完全开源、可手工焊接、接口丰富、本土化服务等特点。适合物联网、智能硬件、机器人等应用和创客开发。智龙创客主板上集成了龙芯 1C SOC、网口、USB 口、电源，SD 卡插槽和 RTC 时钟等主要部件，并提供排针接口，可通过扩展板实现更多的功能。智龙创客主板可以运行嵌入式 Linux 系统和 RT-Thread 实时操作系统，方便用户开发，实现各种创意。

智龙创客主板是首个基于龙芯的创客开发硬件平台，与目前已有的创客开发板 Arduino 相比，具有性能高、网络支持好，接口丰富、可运行 Linux 操作系统等优势。智龙创客主板应用领域为物联网控制、智能硬件、机器人、龙芯嵌入式教学开发，也可作为为 Arduino 主板的升级产品，

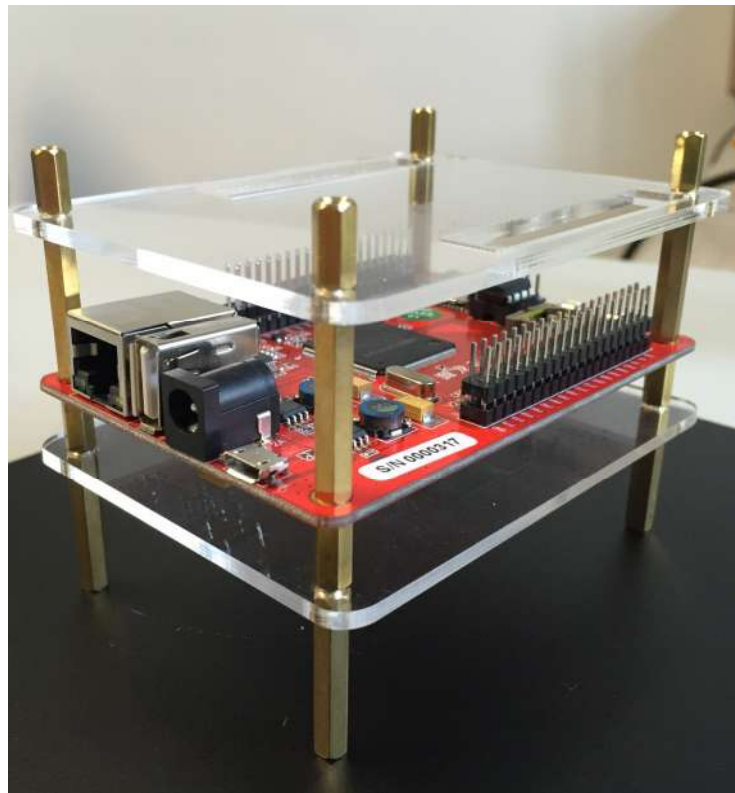


图 1-2 智龙 V2.0



图 1-3 开发板正面

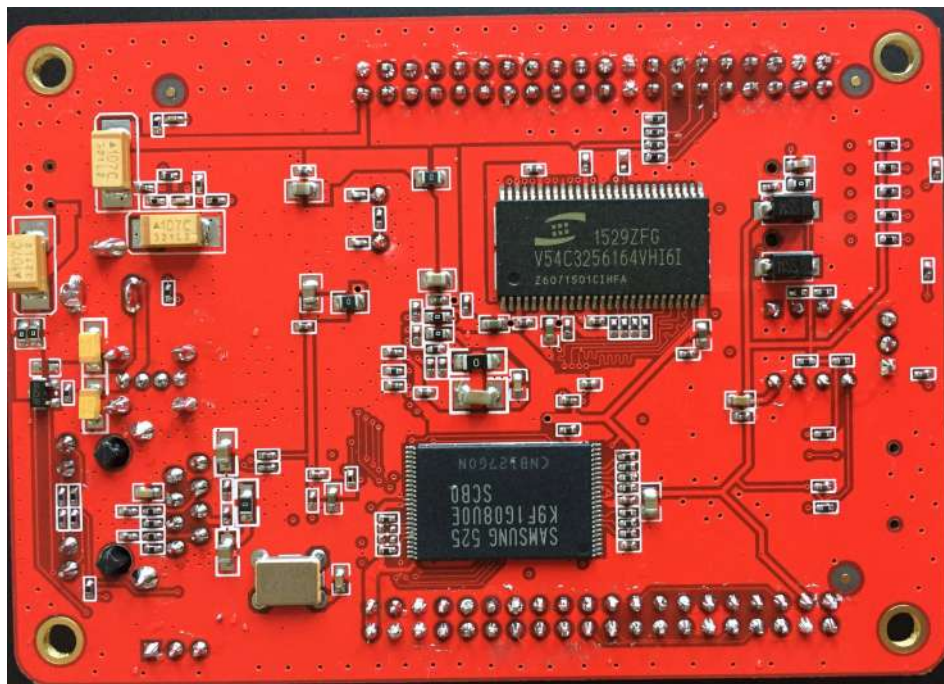


图 1-4 开发板背面

1.3 硬件接口

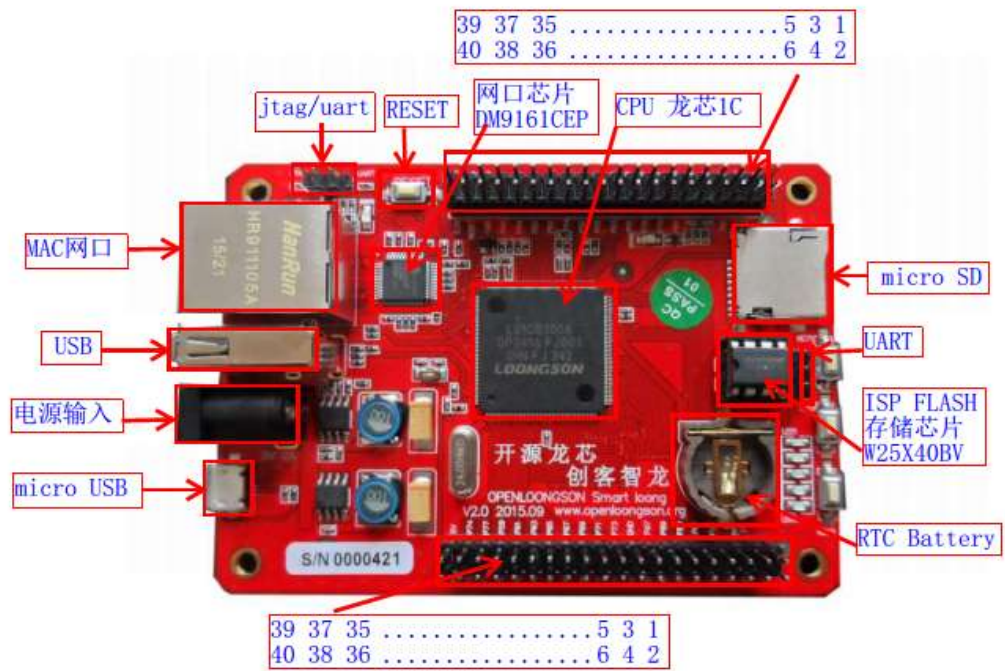


图 1-5 开发板标识

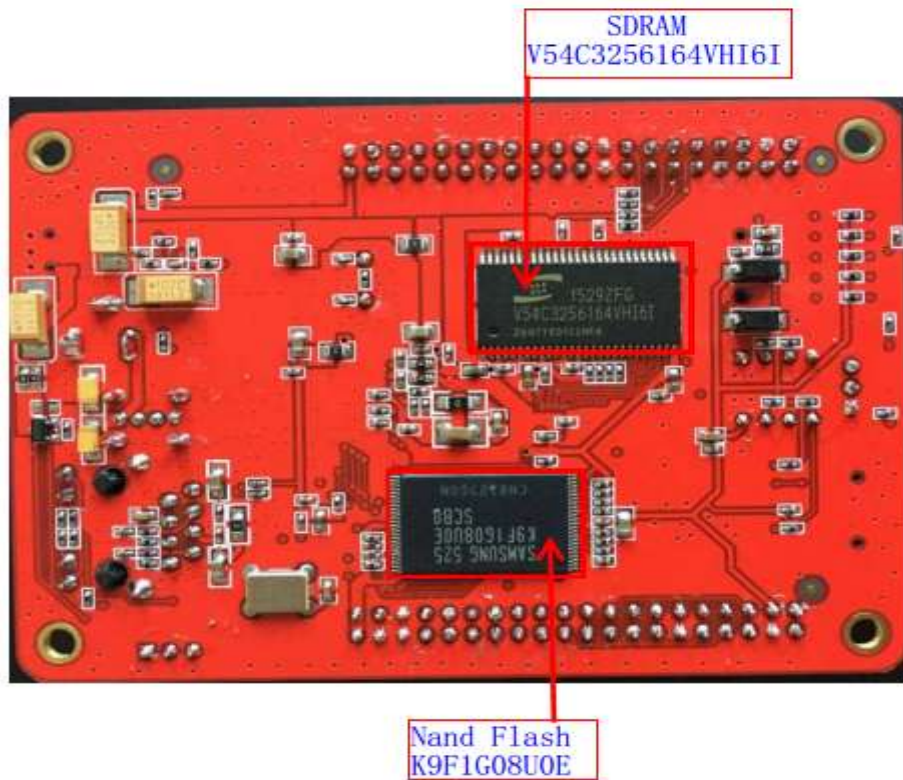


图 1-6 开发板标识

1.4 串口调试连接

1.3.1 设置串口终端软件

为了在开发板上进行相关的命令操作，需要使用交叉串口线连接开发板和主机，同时还需要在主机上使用一个串口终端软件。

如果主机的系统是 Windows 操作系统，可以使用 SecureCRT 或者超级终端。

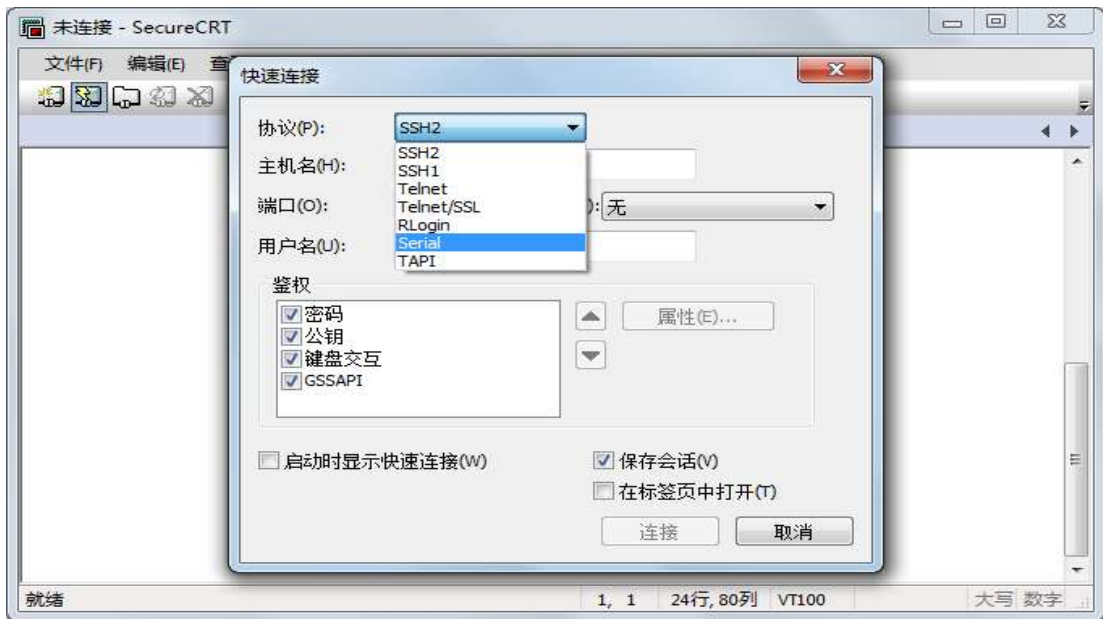
如果主机的系统是 Linux 操作系统，可以使用 minicom 串口终端软件。

下面以 Windows 操作系统上的 SecureCRT 串口终端软件为例。

(1) 安装并打开 SecureCRT，如下图：

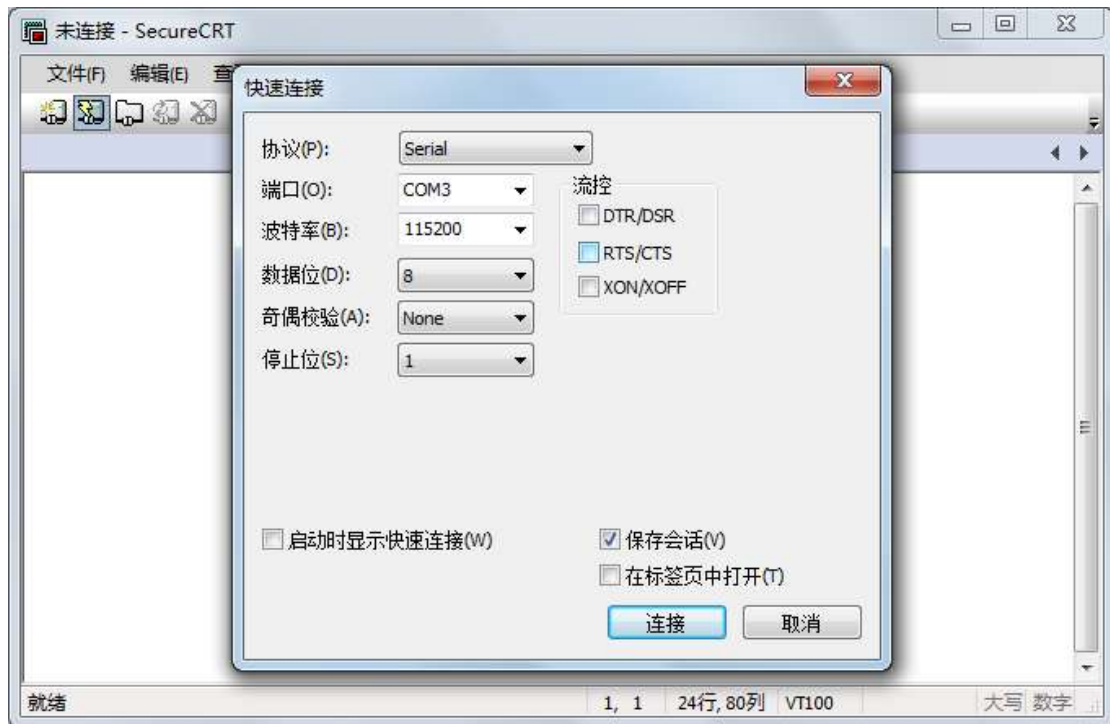


(2) 在快速连接对话框中，协议 (P) 下拉菜单中选择串口 Serial 协议选项，如下图：

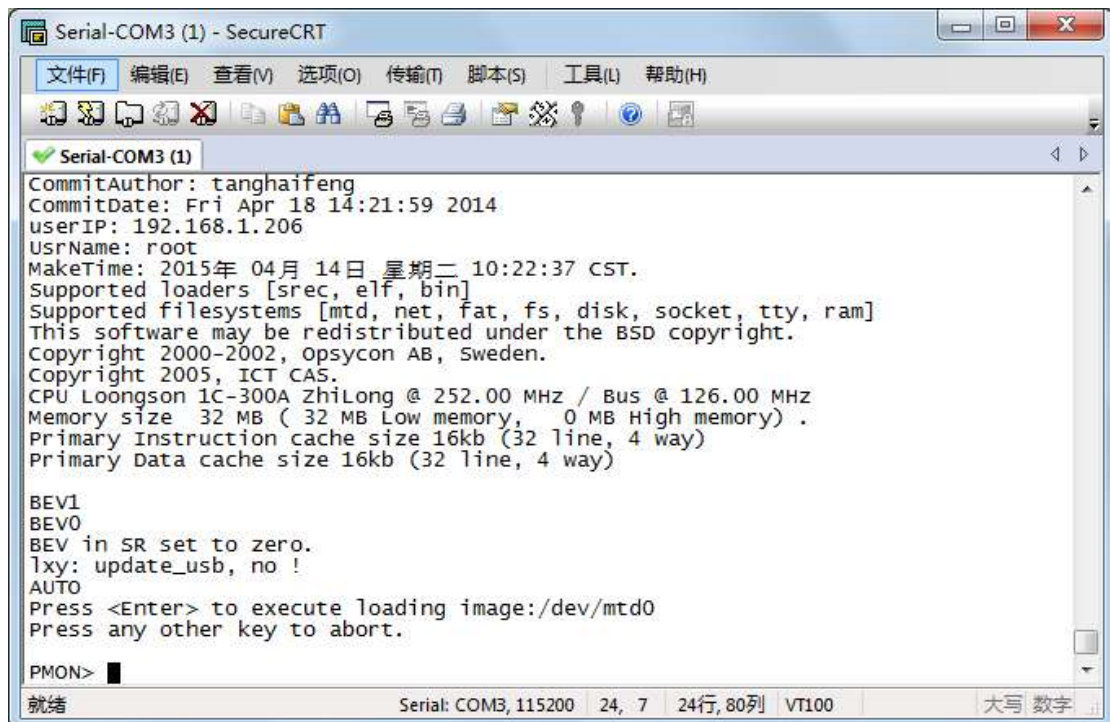


(3) 在快速链接对话框中，设置端口 Port 为 COM3，设置波特率为 115200，8 位数据位，无奇偶检验，1 位停止位，去掉 Flow Control 中所有的勾选，最后连接，如下图：**【实际串口信息在“设备管理器”中可以看到串口的端口信息。注意：应根据具体信息设置正确的端口】**





(4) 开发板上电，按任意键进入 PMON(不进行任何操作，则初始化完成后系统开始运行)，如下图：



1.4 Flash 烧写 linux 系统

1.4.1 windows 下使用 TFTP 服务器

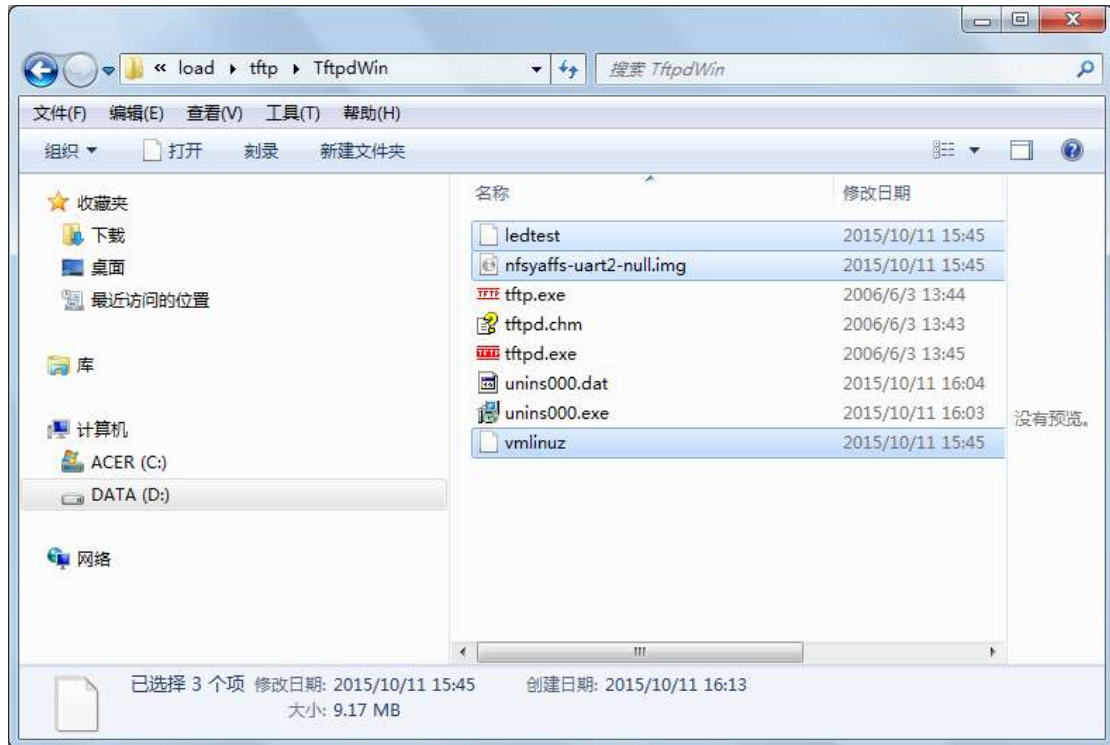
由于智龙 V2.0 开发板的系统固件的恢复与更新通常是通过网络 TFTP 协议来传输的，所以需要使用 TFTP 服务器。因为在开发板的 PMON 上已经内置了 TFTP 的客户端，所以只需在主机上安装 TFTP 的服务端。

这里安装与设置的 TFTP 服务器是 Windows 上的。

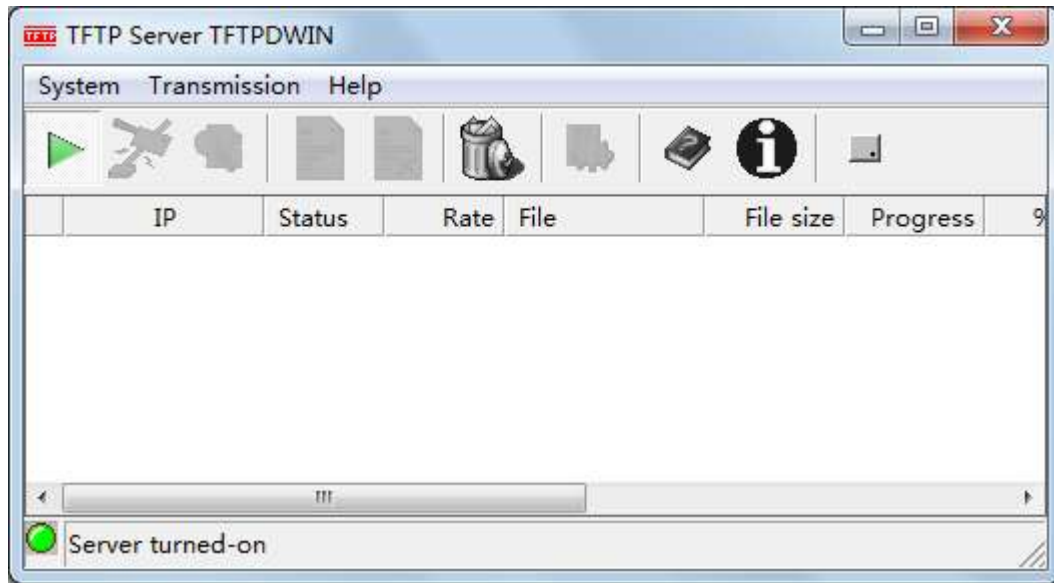
(1) 安装 TFTP



(2) 把要编译好的文件放置于 TFTP 所在目录

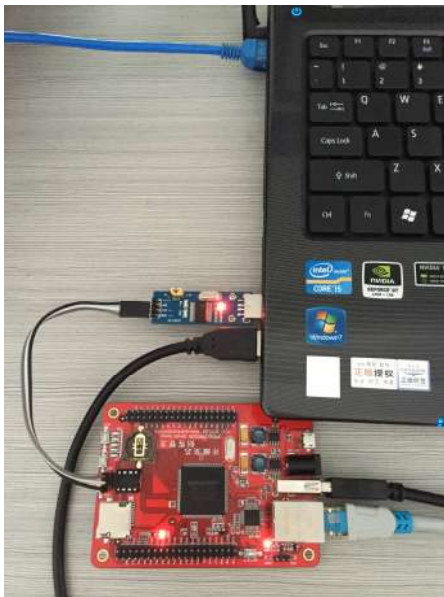


(3) 打开 TFTP，即可运行。

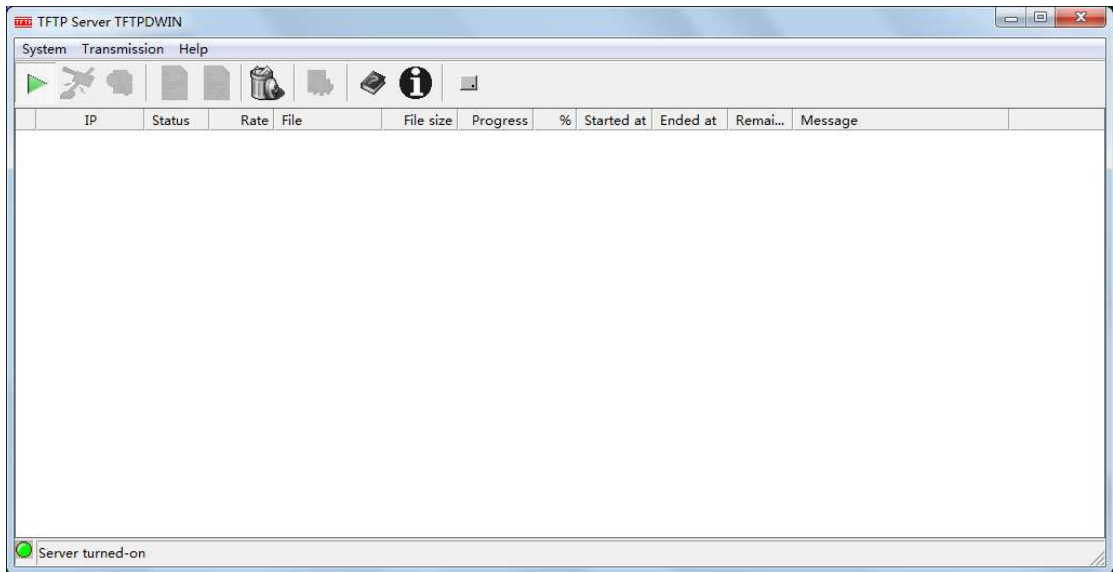
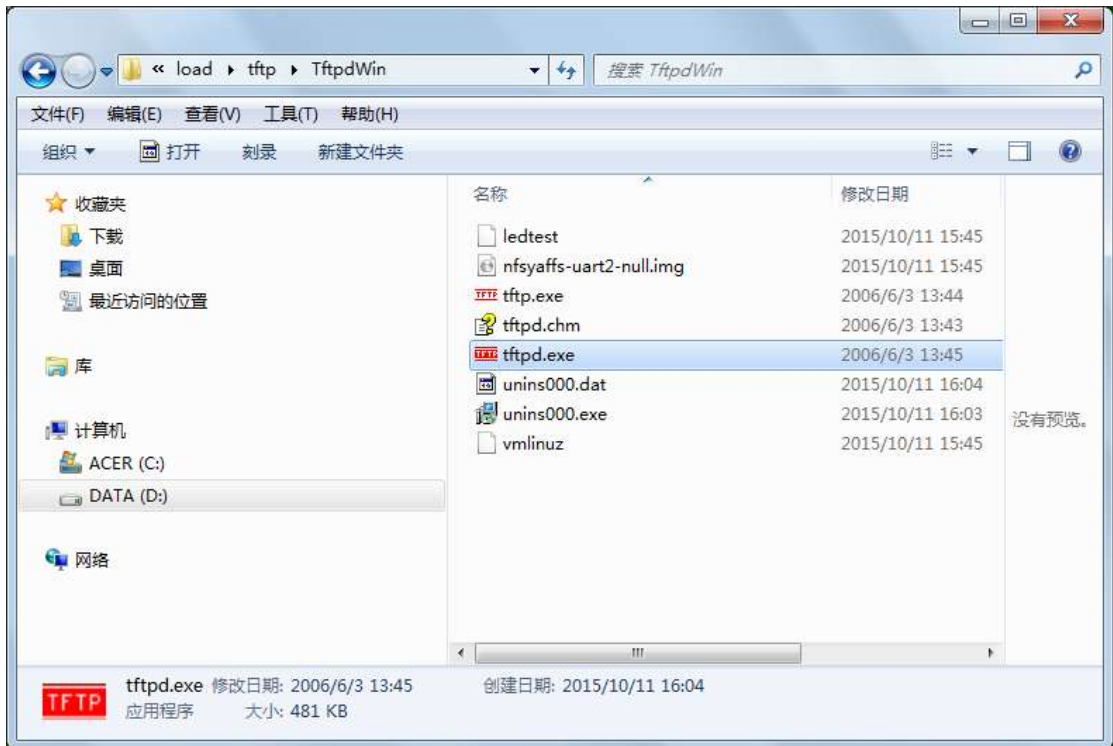


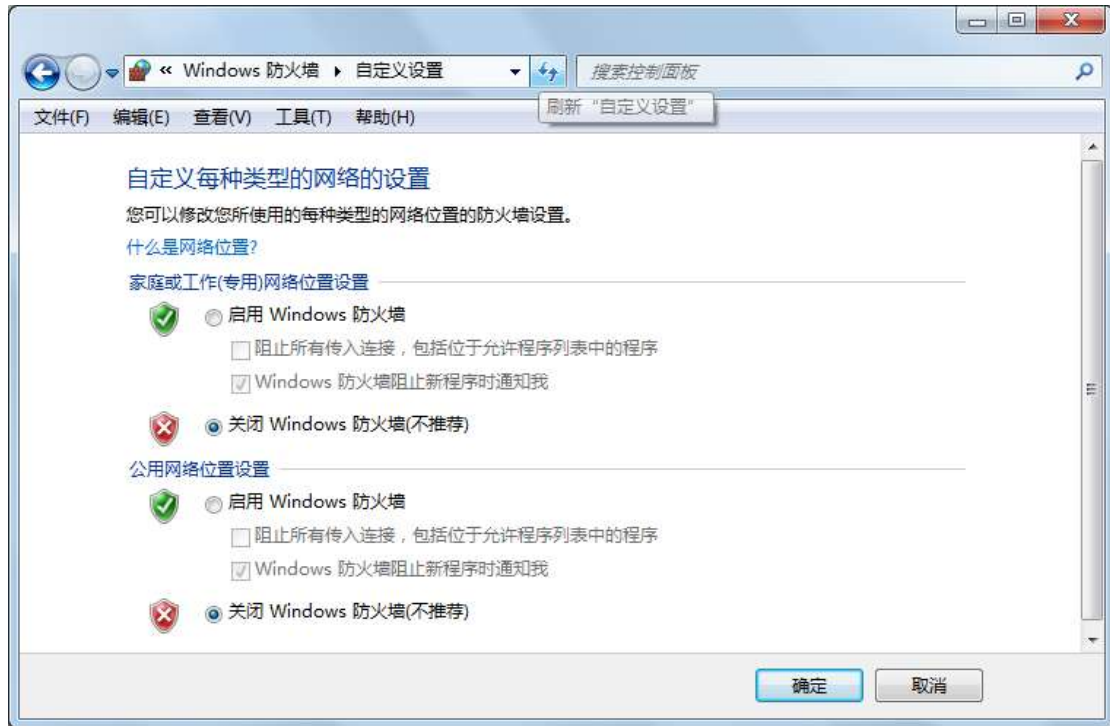
1.4.2 烧写过程

(1) 开发板连接：串口与 PC 连接；采用 USB 给开发板供电；网线 1 连接开发板与交换机，网线 2 连接 PC 与交换机（手册采用直通网线，若用户使用交叉网线，则直接用交叉网线连接开发板与 PC）如下图：



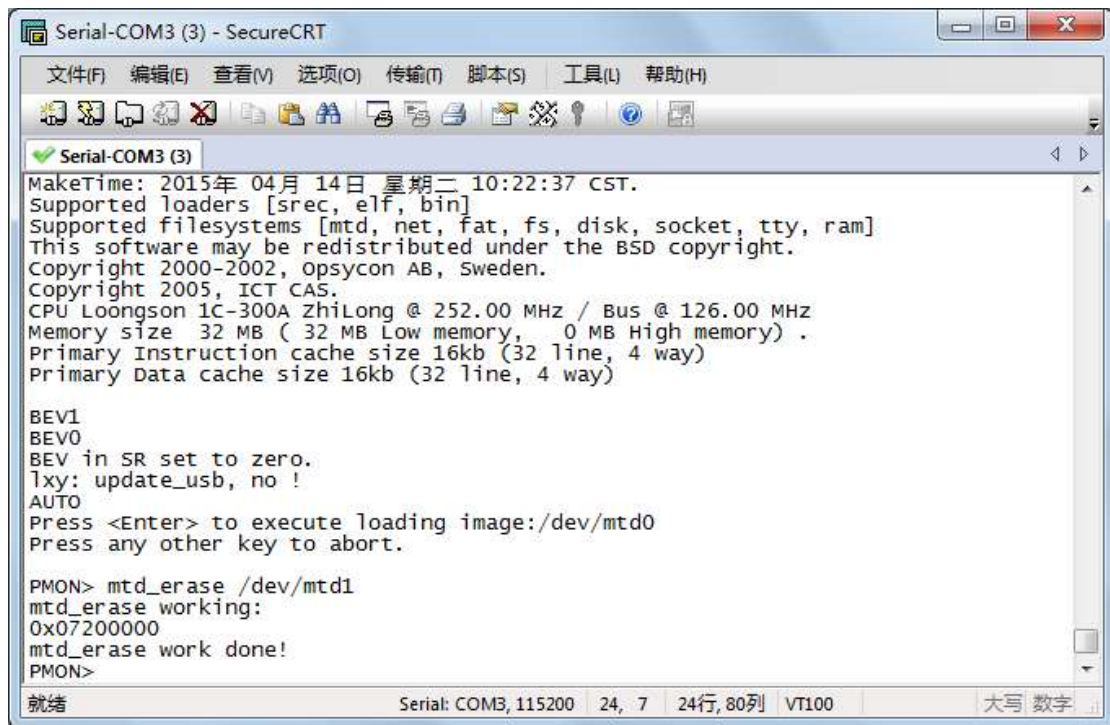
(2) 打开 TFTP，并把编译好的内核、文件系统放置于 TFTP 的安装目录下，**关闭 PC 防火墙**，如下图：



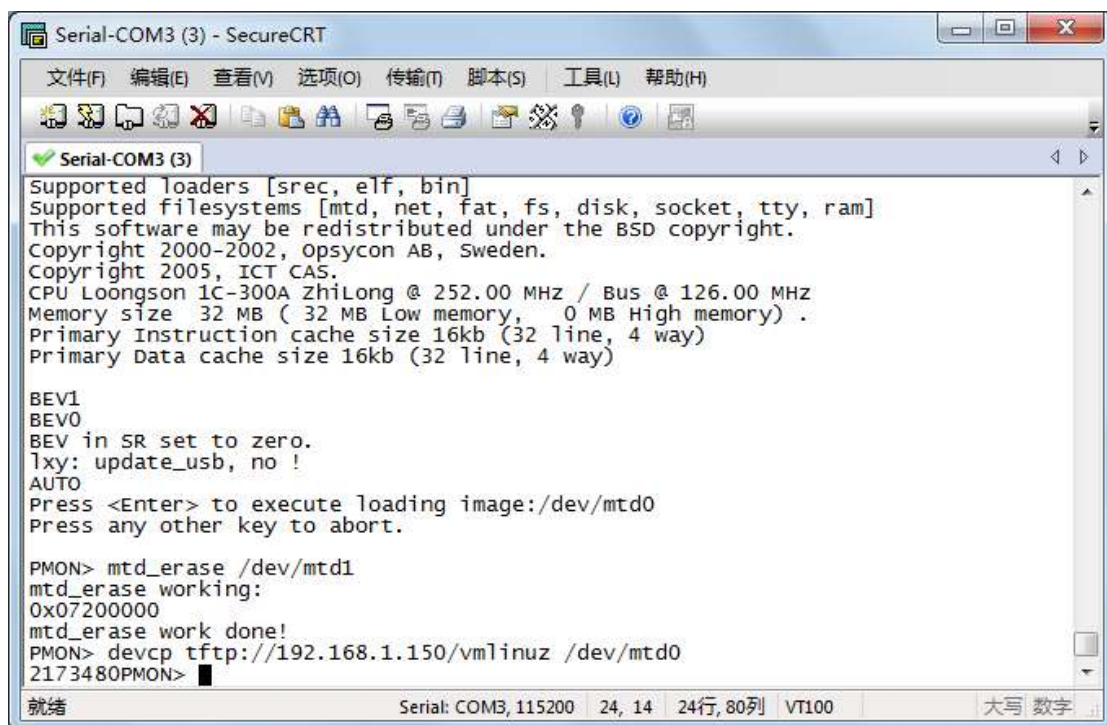
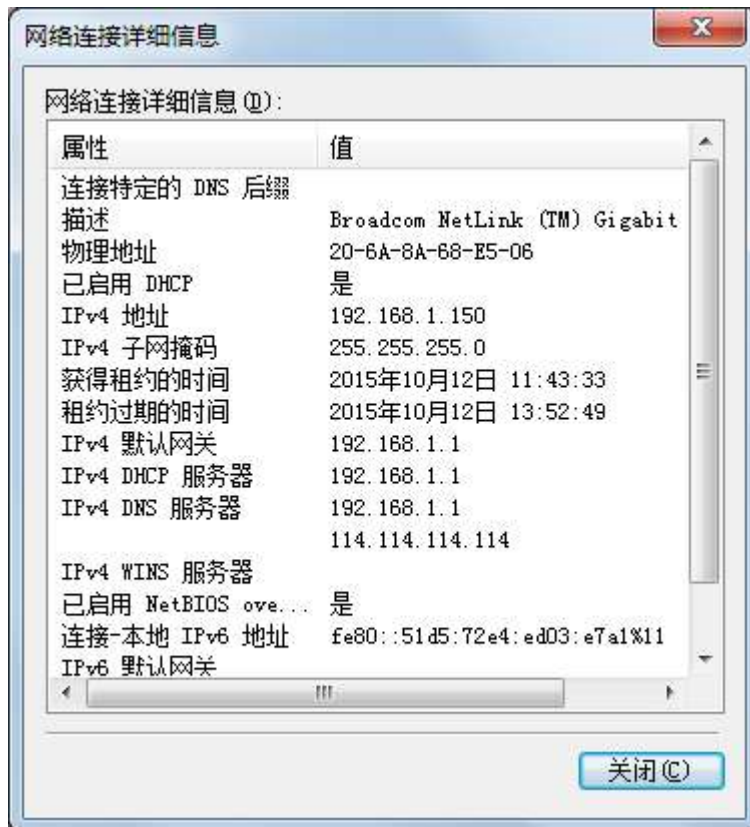


(3) 参考 1.3.1 进入 PMON

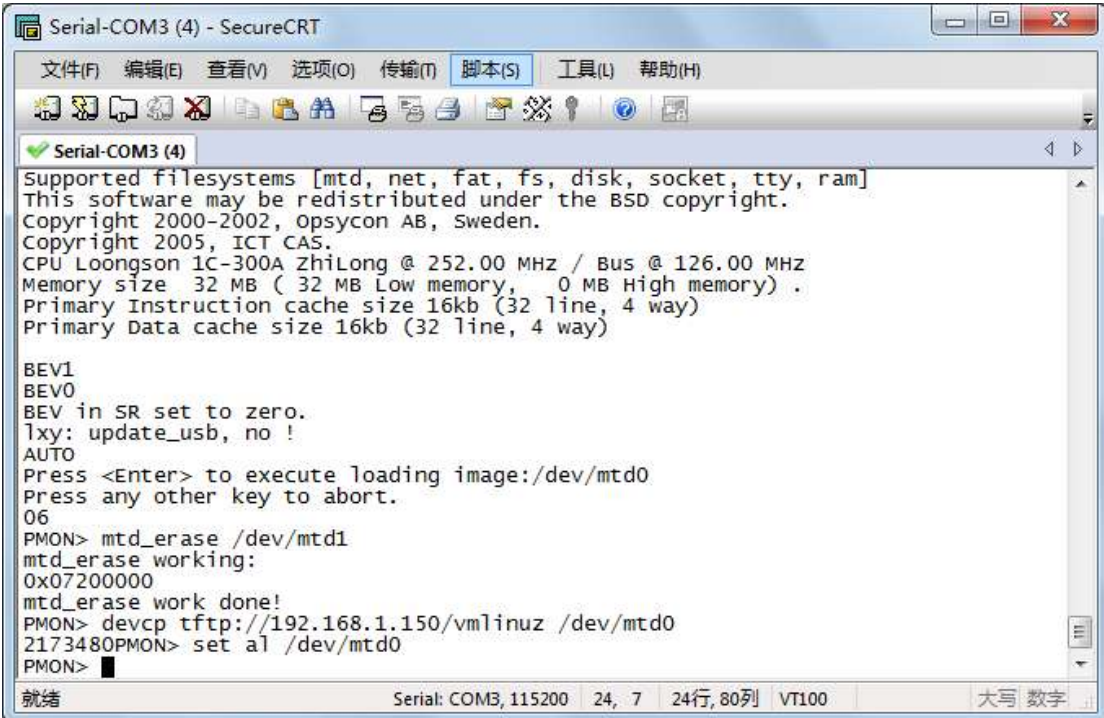
(4) 在 PMON 中输入命令: `mtd_erase /dev/mtd1` //擦除数据, 然后回车, 如下图



(5) 在 PMON 中输入命令: `devcp tftp://192.168.1.150/vmlinuz /dev/mtd0` //下载内核,然后回车 (注意命令中应写 PC 的 IP), 如下图:



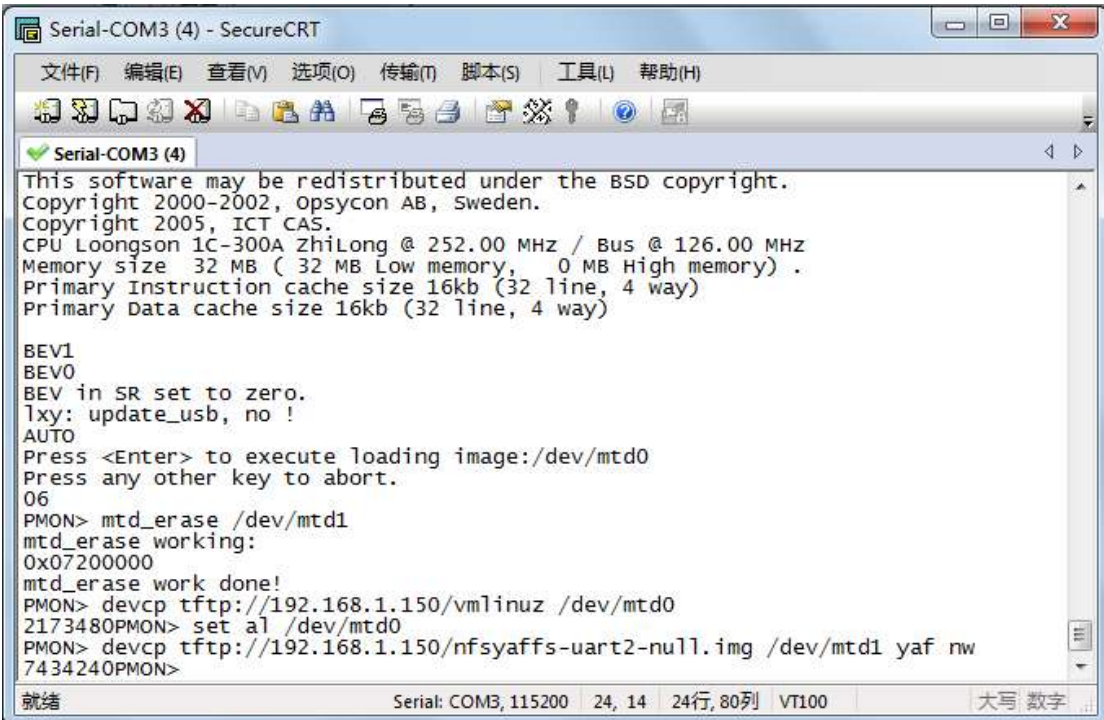
(6) 在 PMON 中输入命令: set al /dev/mtd0 //设置启动参数, 然后回车, 如下图:



```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
Supported filesystems [mtd, net, fat, fs, disk, socket, tty, ram]
This software may be redistributed under the BSD copyright.
Copyright 2000-2002, Opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU Loongson 1C-300A ZhiLong @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory,  0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV1
BEV0
BEV in SR set to zero.
lxy: update_usb, no !
AUTO
Press <Enter> to execute loading image:/dev/mtd0
Press any other key to abort.
06
PMON> mtd_erase /dev/mtd1
mtd_erase working:
0x07200000
mtd_erase work done!
PMON> devcp tftp://192.168.1.150/vmlinuz /dev/mtd0
2173480PMON> set al /dev/mtd0
PMON> █
就绪 Serial: COM3, 115200 24, 7 24行, 80列 VT100 大写 数字
```

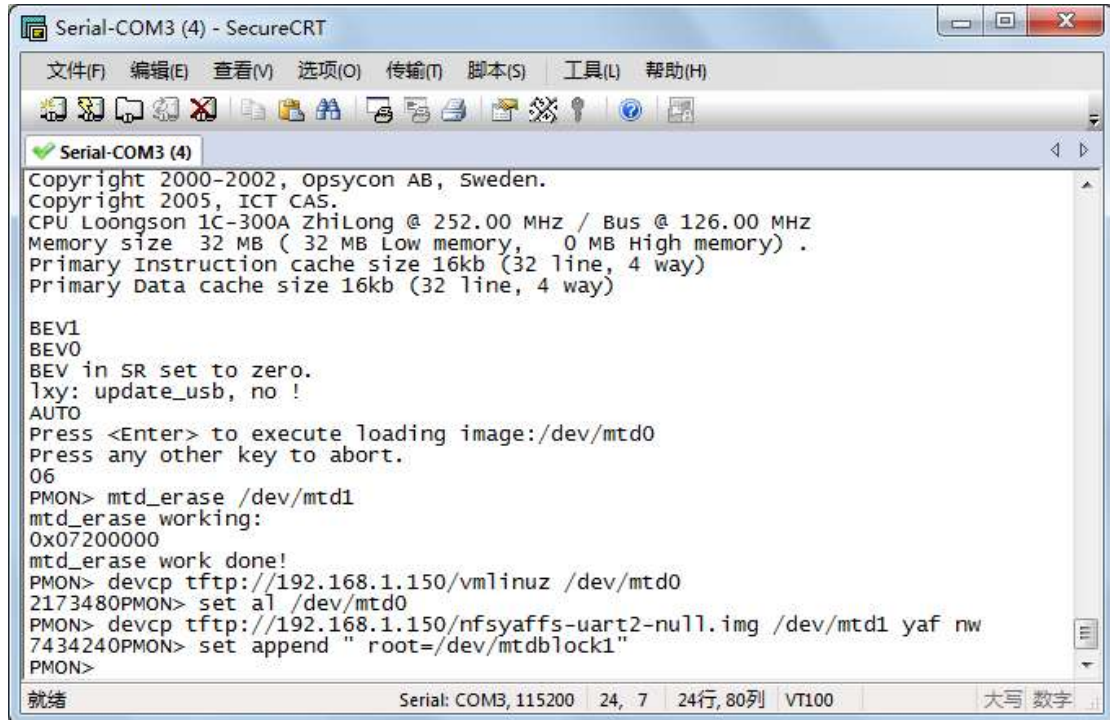
(7) 在 PMON 中输入命令: devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf nw //烧写文件系统, 然后回车, 如下图:



```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
This software may be redistributed under the BSD copyright.
Copyright 2000-2002, Opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU Loongson 1C-300A ZhiLong @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory,  0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV1
BEV0
BEV in SR set to zero.
lxy: update_usb, no !
AUTO
Press <Enter> to execute loading image:/dev/mtd0
Press any other key to abort.
06
PMON> mtd_erase /dev/mtd1
mtd_erase working:
0x07200000
mtd_erase work done!
PMON> devcp tftp://192.168.1.150/vmlinuz /dev/mtd0
2173480PMON> set al /dev/mtd0
PMON> devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf nw
7434240PMON>
就绪 Serial: COM3, 115200 24, 14 24行, 80列 VT100 大写 数字
```

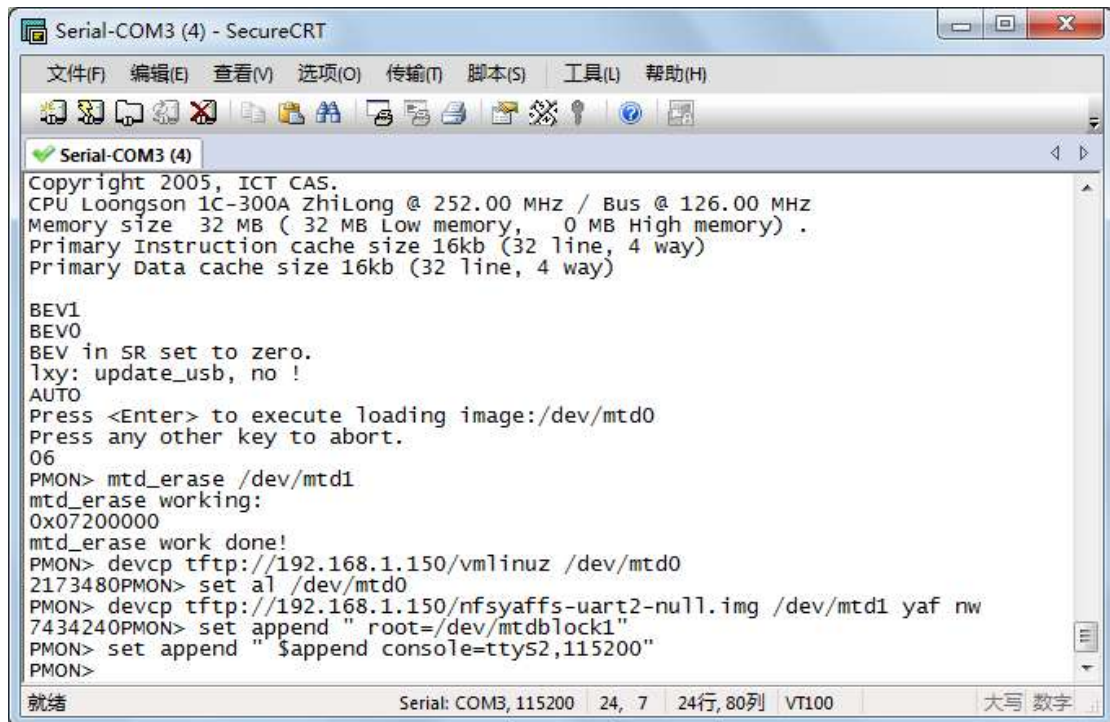
(8) 在 PMON 中输入命令: `set append " root=/dev/mtdblock1" //根目录位置, 块设备, 然后回车, 如下图:`



```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
Copyright 2000-2002, opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU Loongson 1C-300A ZhiLong @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory,  0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV1
BEV0
BEV in SR set to zero.
lxy: update_usb, no !
AUTO
Press <Enter> to execute loading image:/dev/mtd0
Press any other key to abort.
06
PMON> mtd_erase /dev/mtd1
mtd_erase working:
0x07200000
mtd_erase work done!
PMON> devcp tftp://192.168.1.150/vmlinuz /dev/mtd0
2173480PMON> set al /dev/mtd0
PMON> devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf nw
7434240PMON> set append " root=/dev/mtdblock1"
PMON>
就绪 Serial: COM3, 115200 24, 7 24行, 80列 VT100 大写 数字
```

(9) 在 PMON 中输入命令: `set append "$append console=ttyS2,115200" //设置串口 3, 115200 波特率, 然后回车, 如下图:`



```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
Copyright 2005, ICT CAS.
CPU Loongson 1C-300A ZhiLong @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory,  0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV1
BEV0
BEV in SR set to zero.
lxy: update_usb, no !
AUTO
Press <Enter> to execute loading image:/dev/mtd0
Press any other key to abort.
06
PMON> mtd_erase /dev/mtd1
mtd_erase working:
0x07200000
mtd_erase work done!
PMON> devcp tftp://192.168.1.150/vmlinuz /dev/mtd0
2173480PMON> set al /dev/mtd0
PMON> devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf nw
7434240PMON> set append " root=/dev/mtdblock1"
PMON> set append "$append console=ttyS2,115200"
PMON>
就绪 Serial: COM3, 115200 24, 7 24行, 80列 VT100 大写 数字
```

(10) 在 PMON 中输入命令: `set append " $append noinitrd init=/linuxrc rw rootfstype=yaffs2" //noinitrd 代表没有使用 ramdisk; init=/linuxrc 是指内核启动起来后进入系统中运行的第一个脚本, 挂载之后文件系统是只读的, 所以就加了个 rw; rootfstype=yaffs2 指明文件系统类型为 yaffs2 不然没法挂载根分区, 然后回车, 如下图:`

```

Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
CPU Loongson 1C-300A ZhiLong @ 252.00 MHz / Bus @ 126.00 MHz
Memory size 32 MB ( 32 MB Low memory,  0 MB High memory) .
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV1
BEV0
BEV in SR set to zero.
lxy: update_usb, no !
AUTO
Press <Enter> to execute loading image:/dev/mtd0
Press any other key to abort.
06
PMON> mtd_erase /dev/mtd1
mtd_erase working:
0x07200000
mtd_erase work done!
PMON> devcp tftp://192.168.1.150/vmlinuz /dev/mtd0
2173480PMON> set al /dev/mtd0
PMON> devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf rw
7434240PMON> set append " root=/dev/mtdblock1"
PMON> set append " $append console=ttys2,115200"
PMON> set append " $append noinitrd init=/linuxrc rw rootfstype=yaffs2"
PMON>
就绪 Serial: COM3, 115200 24, 7 24行, 80列 VT100 大写 数字

```

(11) 在 PMON 中输入命令:
`set append " $append video=ls1bfb:480x272-16@60 fbcon=rotate:1 consoleblank=0"//fbcon=rotate:1 标示屏幕可旋转; consoleblank=0 禁用屏幕白色待机, 然后回车, 如下图:`

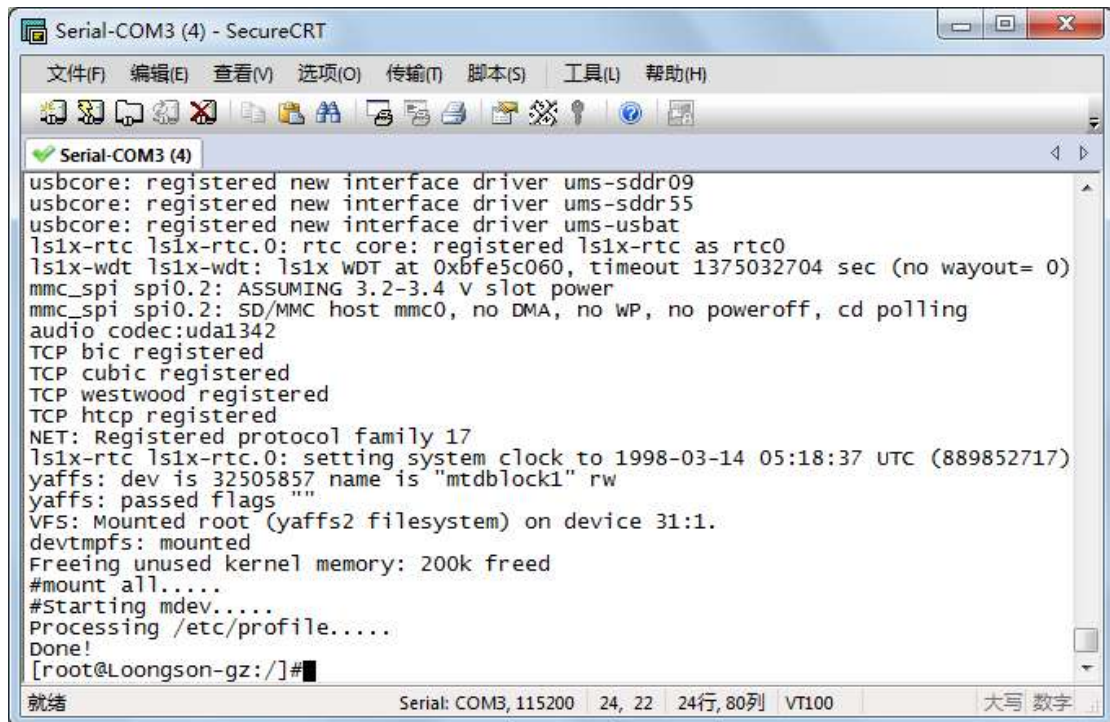
```

Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
Primary Instruction cache size 16kb (32 line, 4 way)
Primary Data cache size 16kb (32 line, 4 way)

BEV1
BEV0
BEV in SR set to zero.
lxy: update_usb, no !
AUTO
Press <Enter> to execute loading image:/dev/mtd0
Press any other key to abort.
06
PMON> mtd_erase /dev/mtd1
mtd_erase working:
0x07200000
mtd_erase work done!
PMON> devcp tftp://192.168.1.150/vmlinuz /dev/mtd0
2173480PMON> set al /dev/mtd0
PMON> devcp tftp://192.168.1.150/nfsyaffs-uart2-null.img /dev/mtd1 yaf rw
7434240PMON> set append " root=/dev/mtdblock1"
PMON> set append " $append console=ttys2,115200"
PMON> set append " $append noinitrd init=/linuxrc rw rootfstype=yaffs2"
PMON> set append " $append video=ls1bfb:480x272-16@60 fbcon=rotate:1 consoleblank=0"
PMON>
就绪 Serial: COM3, 115200 24, 7 24行, 80列 VT100 大写 数字

```


(12) 在 PMON 中输入命令: reboot //重启, 然后回车, 如下图:



```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
usbcore: registered new interface driver ums-sddr09
usbcore: registered new interface driver ums-sddr55
usbcore: registered new interface driver ums-usbat
lsix-rtc lsix-rtc.0: rtc core: registered lsix-rtc as rtc0
lsix-wdt lsix-wdt: lsix WDT at 0xbfe5c060, timeout 1375032704 sec (no wayout= 0)
mmc_spi spi0.2: ASSUMING 3.2-3.4 V slot power
mmc_spi spi0.2: SD/MMC host mmc0, no DMA, no WP, no poweroff, cd polling
audio codec:uda1342
TCP bic registered
TCP cubic registered
TCP westwood registered
TCP htcp registered
NET: Registered protocol family 17
lsix-rtc lsix-rtc.0: setting system clock to 1998-03-14 05:18:37 UTC (889852717)
yaffs: dev is 32505857 name is "mtdblock1" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs2 filesystem) on device 31:1.
devtmpfs: mounted
Freeing unused kernel memory: 200k freed
#mount all.....
#starting mdev.....
Processing /etc/profile.....
Done!
[root@Loongson-gz:~]#
```

就绪 Serial: COM3, 115200 24, 22 24行, 80列 VT100 大写 数字

至此, 烧写 linux 系统结束。

软件篇

2.1 linux 内核配置

以 Linux-3.0.82 内核为例，描述 Linux 系统的配置编译过程：

配置内核：

(1) 新建一个放置内核源码的文件夹：

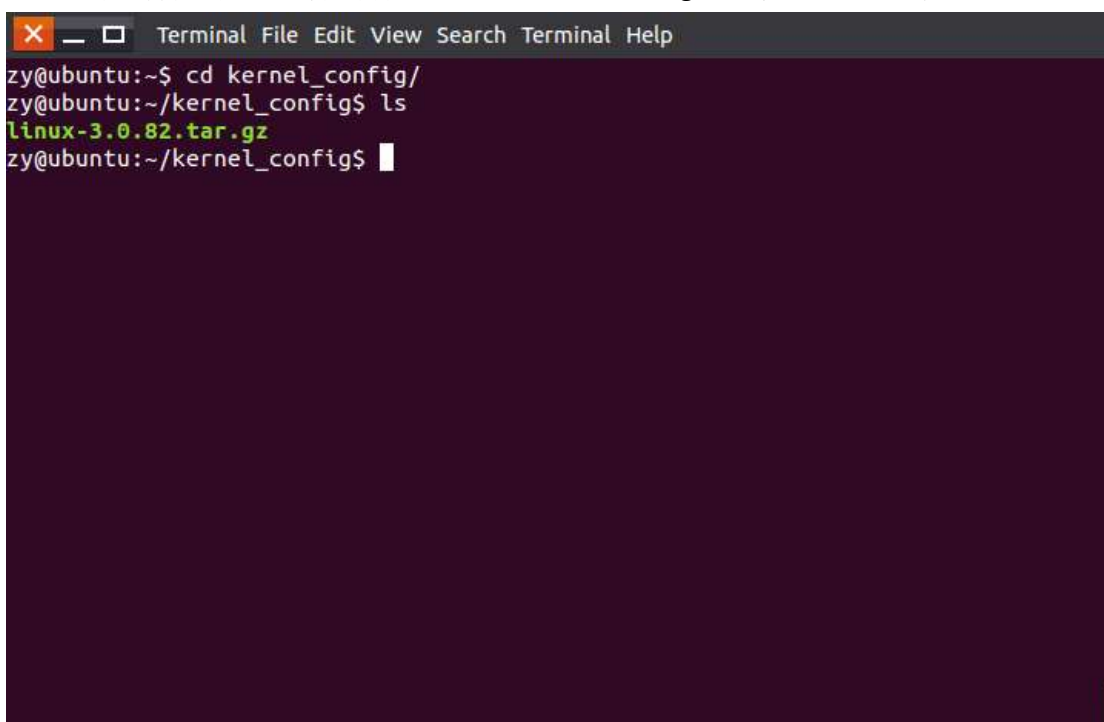
示例文件夹暂时命名为 kernel_config，用户可自行创建不同名称的文件夹。

(2) 将内核源码的压缩包放入新建的文件夹中，并解压缩：

```
#cd kernel_config
```

```
#ls
```

进入到文件夹中可以看到有 linux-3.0.82.tar.gz 压缩包（绿色字体）

A terminal window screenshot with a dark background. The title bar shows 'Terminal File Edit View Search Terminal Help'. The terminal content shows the following commands and output:

```
zy@ubuntu:~$ cd kernel_config/  
zy@ubuntu:~/kernel_config$ ls  
linux-3.0.82.tar.gz  
zy@ubuntu:~/kernel_config$
```

The file 'linux-3.0.82.tar.gz' is highlighted in green text.

```
#tar -zxvf linux-3.0.82.tar.gz
```



```
#ls
```

解压缩后可以看到出现了 linux-3.0.82 文件夹（蓝色字体）

```
zy@ubuntu:~/kernel_config$ ls
linux-3.0.82-1c/crypto/Makefile
linux-3.0.82-1c/crypto/cryptd.c
linux-3.0.82-1c/crypto/cbc.c
linux-3.0.82-1c/crypto/compress.o
linux-3.0.82-1c/crypto/shash.o
linux-3.0.82-1c/crypto/.algapi.o.cmd
linux-3.0.82-1c/crypto/rng.c
linux-3.0.82-1c/crypto/algapi.c
linux-3.0.82-1c/crypto/blkcipher.c
linux-3.0.82-1c/crypto/deflate.c
linux-3.0.82-1c/crypto/.sha256_generic.o.cmd
linux-3.0.82-1c/crypto/crypto_wq.c
linux-3.0.82-1c/crypto/blowfish.c
zy@ubuntu:~/kernel_config$ ls
linux-3.0.82  linux-3.0.82.tar.gz
zy@ubuntu:~/kernel_config$
```

```
#cd linux-3.0.82
```

```
#ls
```

可以看到各种文件夹和文件

```
zy@ubuntu:~/kernel_config$ cd linux-3.0.82
zy@ubuntu:~/kernel_config/linux-3.0.82$ ls
\          CREDITS          include  lib          qctransfer.sh  sound      vmlinux
arch       crypto              init     MAINTAINERS  README         System.map  vmlinux.o
block     Documentation      ipc      Makefile     REPORTING-BUGS tags
cmd.sh    drivers            Kbuild  mm           samples        tools
config.lsi1c  firmware          Kconfig Module.symvers  scripts        usr
COPYING   fs                 kernel  net          security       virt
zy@ubuntu:~/kernel_config/linux-3.0.82$
```

(3) 拷贝龙芯 1c 的默认配置到 linux 源代码树根目录:

```
#cd arch/mips/configs
```

```
#ls
```

可以看到许多不同 mips 架构芯片的默认配置

```
zy@ubuntu:~/kernel_config/linux-3.0.82$ cd arch/mips/configs
zy@ubuntu:~/kernel_config/linux-3.0.82/arch/mips/configs$ ls
ar7_defconfig          fuloong2e_defconfig  ls1b_defconfig       pnx8550-jbs_defconfig
bcm47xx_defconfig     gpr_defconfig        ls1c_defconfig       pnx8550-stb810_defconfig
bcm63xx_defconfig     ip22_defconfig       malta_defconfig      powertv_defconfig
bigsur_defconfig      ip27_defconfig       markeins_defconfig   rb532_defconfig
capcella_defconfig    ip28_defconfig       mipssim_defconfig    rbtx49xx_defconfig
cavium-octeon_defconfig ip32_defconfig       mpc30x_defconfig     rm200_defconfig
cobalt_defconfig      jazz_defconfig       msp71xx_defconfig    sb1250-swarm_defconfig
db1000_defconfig      jmr3927_defconfig    mtx1_defconfig       tb0219_defconfig
db1100_defconfig      lasat_defconfig      nlm_xlr_defconfig    tb0226_defconfig
db1200_defconfig      lemote2f_defconfig   pb1100_defconfig     tb0287_defconfig
db1500_defconfig      ls1a_cloud_config    pb1200_defconfig     workpad_defconfig
db1550_defconfig      ls1a_defconfig       pb1500_defconfig     wrppmc_defconfig
decstation_defconfig  ls1b_cloud_defconfig pb1550_defconfig     yosemite_defconfig
e55_defconfig         ls1b_core_defconfig  pnx8335-stb225_defconfig
zy@ubuntu:~/kernel_config/linux-3.0.82/arch/mips/configs$
```

```
#cp ls1c_defconfig ../../..
#cd ../../..
#mv ls1c_defconfig .config
#ls -a
```

将 ls1c 的默认配置 ls1c_defconfig 拷贝到 linux 源代码树根目录中，并重命名为 .config

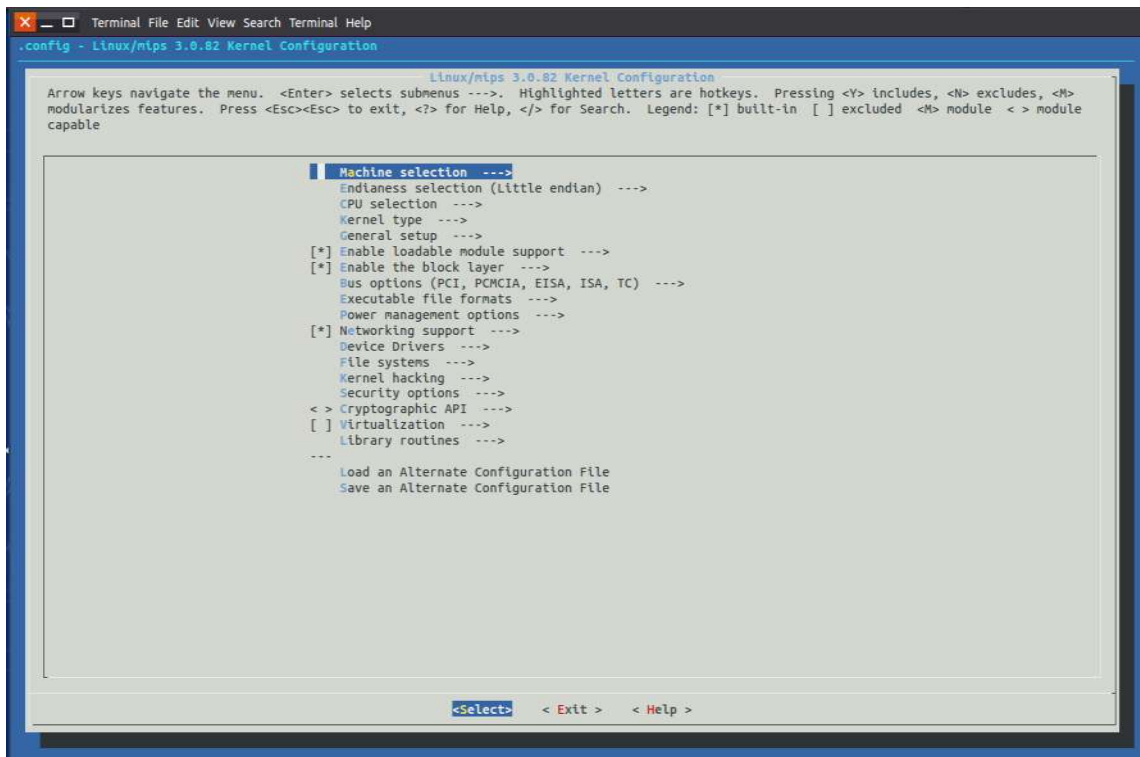
```
zy@ubuntu:~/kernel_config/linux-3.0.82/arch/mips/configs$ cp ls1c_defconfig ../../..
zy@ubuntu:~/kernel_config/linux-3.0.82/arch/mips/configs$ cd ../../..
zy@ubuntu:~/kernel_config/linux-3.0.82$ mv ls1c_defconfig .config
zy@ubuntu:~/kernel_config/linux-3.0.82$ ls -a
.          drivers      MAINTAINERS          System.map          ..tmp_vmlinux2.cmd
..         firmware    Makefile              tags                tools
\          fs           .missing-syscalls.d .tmp_kallsyms1.o   usr
arch       .git         mm                    ..tmp_kallsyms1.o.ccmd .version
block     .gitignore  Module.symvers       .tmp_kallsyms1.S   virt
cmd.sh     include     net                   .tmp_kallsyms2.o   vmlinux
.config    init        qctransfer.sh        ..tmp_kallsyms2.o.ccmd .vmlinux.cmd
config.ls1c ipc          README                .tmp_kallsyms2.S   vmlinux.o
.config.old Kbuild      REPORTING-BUGS       .tmp_System.map    .vmlinux.o.ccmd
COPYING    Kconfig     samples               .tmp_versions
CREDITS    kernel      scripts               .tmp_vmlinux1
crypto     lib         security              ..tmp_vmlinux1.ccmd
Documentation .mailmap    sound                 .tmp_vmlinux2
zy@ubuntu:~/kernel_config/linux-3.0.82$
```

注意：若采用默认配置，可以跳过第（4）步。

(4) 对内核进行图形化配置:

```
#make menuconfig
```

进入到内核配置菜单，配置完成后保存退出即可



(5) 编译 linux 内核

```
#make
```

```
#ls
```

可以看到最终生成的内核映像文件 vmlinux 与 vmlinuz 等就在内核源代码的根目录下。

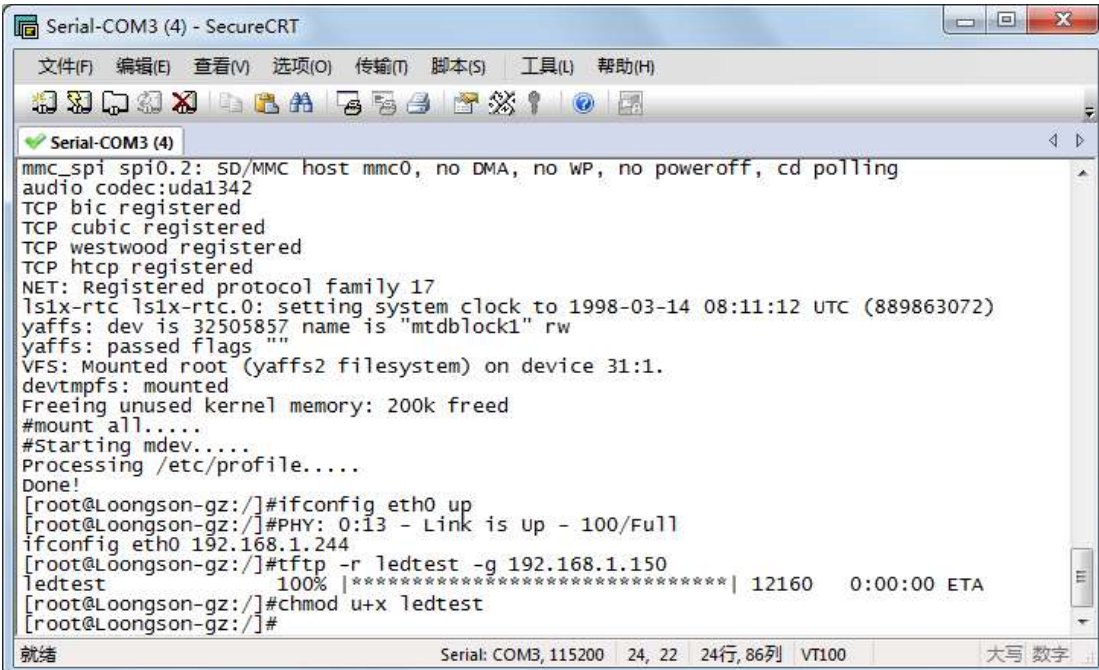
```
Terminal File Edit View Search Terminal Help
AS arch/mips/lib/strlen_user.o
AS arch/mips/lib/strncpy_user.o
AS arch/mips/lib/strlen_user.o
CC arch/mips/lib/uncached.o
AR arch/mips/lib/lib.a
LD vmlinux.o
MODPOST vmlinux.o
GEN .version
CHK include/generated/compile.h
UPD include/generated/compile.h
CC init/version.o
LD init/built-in.o
LD .tmp_vmlinux1
KSYM .tmp_kallsyms1.5
AS .tmp_kallsyms1.o
LD .tmp_vmlinux2
KSYM .tmp_kallsyms2.5
AS .tmp_kallsyms2.o
LD vmlinux
SYSMAP system.map
SYSMAP .tmp_System.map
AS arch/mips/boot/compressed/head.o
CC arch/mips/boot/compressed/decompress.o
CC arch/mips/boot/compressed/dbg.o
CC arch/mips/boot/compressed/dummy.o
OBJCOPY arch/mips/boot/compressed/vmlinux.bin
GZIP arch/mips/boot/compressed/vmlinux.bin.z
OBJCOPY arch/mips/boot/compressed/piggy.o
HOSTCC arch/mips/boot/compressed/calc_vmlinux_load_addr
LD vmlinuz
STRIP vmlinuz
Building modules, stage 2.
MODPOST 1 modules
CC drivers/scsi/scsi_wait_scan.mod.o
LD [M] drivers/scsi/scsi_wait_scan.ko
zy@ubuntu:~/kernel_config/linux-3.0.82$ ls
\  config.lsic Documentation include Kconfig Makefile Module.symvers REPORTING-BUGS sound usr vmlinux
arch COPYING drivers init kernel mm net samples System.map virt
block CREDITS firmware ipc lib modules.builtin qctransfer.sh scripts tags vmlinux
cmd.sh crypto fs Kbuild MAINTAINERS modules.order README security tools vmlinux.o
zy@ubuntu:~/kernel_config/linux-3.0.82$
```

2.2 基于 linux 的基础实验

2.2.1 LED 跑马灯

- (1) 把编译好的 LED 文件放置于 TFTP 安装目录下
- (2) 连接 PC 与开发板
- (3) 打开 SecureCRT 并与开发板连接，打开 TFTP，关闭 PC 防火墙
- (4) 在 CRT 中输入一下命令：

```
ifconfig eth0 up //在文件系统里启动网口 ， 然后回车。  
ifconfig eth0 192.168.1.244 //文件系统里配置网络 ip(ip 不要和主机  
一样) ， 然后回车。  
tftp -r ledtest -g 192.168.1.150 //下载编译好的 LED 文件， 回车。  
chmod u+x ledtest //给予权限， 回车。
```

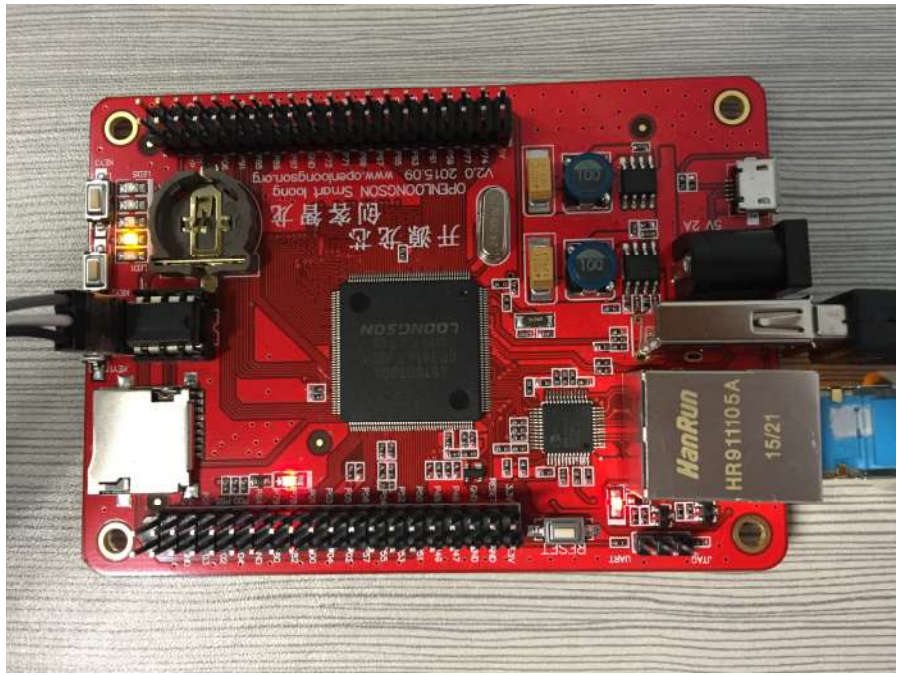


```
Serial-COM3 (4) - SecureCRT  
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)  
Serial-COM3 (4)  
mmc_spi spi0.2: SD/MMC host mmc0, no DMA, no WP, no poweroff, cd polling  
audio codec:uda1342  
TCP bic registered  
TCP cubic registered  
TCP westwood registered  
TCP htcp registered  
NET: Registered protocol family 17  
ls1x-rtc ls1x-rtc.0: setting system clock to 1998-03-14 08:11:12 UTC (889863072)  
yaffs: dev is 32505857 name is "mtdblock1" rw  
yaffs: passed flags ""  
VFS: Mounted root (yaffs2 filesystem) on device 31:1.  
devtmpfs: mounted  
Freeing unused kernel memory: 200k freed  
#mount all.....  
#Starting mdev.....  
Processing /etc/profile.....  
Done!  
[root@Loongson-gz:~]#ifconfig eth0 up  
[root@Loongson-gz:~]#PHY: 0:i3 - Link is up - 100/Full  
ifconfig eth0 192.168.1.244  
[root@Loongson-gz:~]#tftp -r ledtest -g 192.168.1.150  
ledtest 100% |*****| 12160 0:00:00 ETA  
[root@Loongson-gz:~]#chmod u+x ledtest  
[root@Loongson-gz:~]#  
就绪 Serial: COM3, 115200 24, 22 24行, 86列 VT100 大写 数字
```

- (5) 在 SecureCRT 中输入命令：
./ledtest //运行 LED 文件 ， 回车
或者输入：
nohup /ledtest & //在后台运行 LED 文件， 回车
(在 CRT 中使用 ctrl+c 可以暂停运行)


```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
NET: Registered protocol family 17
lsix-rtc lsix-rtc.0: setting system clock to 1998-03-14 08:11:12 UTC (889863072)
yaofs: dev is 32505857 name is "mtdblock1" rw
yaofs: passed flags ""
VFS: Mounted root (yaofs2 filesystem) on device 31:1.
devtmpfs: mounted
Freeing unused kernel memory: 200k freed
#mount all.....
#Starting mdev.....
Processing /etc/profile.....
Done!
[root@Loongson-gz:~]#ifconfig eth0 up
[root@Loongson-gz:~]#PHY: 0:13 - Link is up - 100/Fu11
ifconfig eth0 192.168.1.244
[root@Loongson-gz:~]#tftp -r ledtest -g 192.168.1.150
ledtest          100% |*****| 12160   0:00:00 ETA
[root@Loongson-gz:~]#chmod u+x ledtest
[root@Loongson-gz:~]#./ledtest
Led initial success...
Led initial success...
Led initial success...
Led initial success...
Led initial success...
就绪                               Serial: COM3, 115200  24, 1  24行, 86列  VT100  大写 数字
```

```
Serial-COM3 (4) - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM3 (4)
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
Led off success...
Led on success...
AC
[root@Loongson-gz:~]#nohup ./ledtest &
[root@Loongson-gz:~]#nohup: appending output to nohup.out
就绪                               Serial: COM3, 115200  24, 1  24行, 86列  VT100  大写 数字
```



应用篇

龙芯创客在英国和他的智龙摩斯电码播放器

我其实很早就知道我国有龙芯这么一款国产处理器，但对其的印象一直停留在“安全应用”、“试验品”等字眼上，认为一个普通人并不会接触到龙芯系列的处理器；尤其是在得知龙芯所使用的 MIPS 架构无法兼容 Windows 之后，便更加认为龙芯不会在桌面系统上有所成就——直到今年三月份在某新闻客户端上看到了一则新闻，内容大约是“中国的树莓派 开源龙芯主板开始接受团购”，发现其价格为 199 元并不贵，同时由于对国产 CPU 的好奇，便拍下了这款产品。在经过了漫长的等待——五月份发货，以及我八月份才回国过暑假——之后，终于拿到了这块板子。当然由于这块板子，我也通过社区了解到了龙芯产品的最新动向，比如自主指令集，以及 3A2000 和 3B2000 CPU。这些新产品受到了媒体的广泛关注，龙芯中科也因此被人民日报头版所报道，在此便不再累述。

走出实验室 迈步闯市场

龙芯中科：让“中国芯”奔腾起来

本报北京 8 月 30 日电（记者赵永新）从追求高水平论文到注重满足用户需求，从追求单一性能指标到搭建产业生态系统，龙芯中科技术有限公司（以下简称龙芯中科）面向市场搞研发、加快成果产业化，基于龙芯 CPU（中央处理器）的信息产业生态系统已初步形成。

CPU 是信息产业的基础部件，如果一味依赖进口，信息产业就会受制于人。2001 年，32 岁的中科院计算所研究员胡伟武带领团队研制自主可控的“中国芯”，先后研制出龙芯 1 号、2 号、3 号等系列高性能 CPU。令人遗憾的是，龙芯 CPU 一直未能实现大规模推

广应用。2010 年，龙芯总设计师胡伟武率领团队骨干脱离事业编制，离开中科院计算所创办龙芯中科，开始从学院派转向市场派，着力推进龙芯的产业化。

“与国外芯片相比，龙芯最大的差距不在于技术指标，而是未能与产业链对接，建立与之相匹配的生态系统。”胡伟武告诉记者，龙芯中科成立后，瞄准用户需求搞研发，与曙光、浪潮、锐捷网络、东软集团等国内中下游软硬件企业紧密合作，自主开发出高性能计算机、高端服务器、网络交换机、千兆防火墙等终端产品。目前，基于龙芯 CPU 进行下游解决方案开发的合作伙伴已有

数百家，基于龙芯 CPU 的研发人员已经达到上万人规模，围绕龙芯的自主可控的信息产业生态圈已初步形成。“近年来龙芯中科在 PC 和服务器、智能移动终端、嵌入式应用等三大板块持续发力，2013 年我们的芯片销量为 1.8 万片，去年卖了 35 万片，今年估计要比去年翻一番。”胡伟武说。

（相关报道见第二版）

行进中国·改革故事

人民日报 8 月 31 日所报道的龙芯中科

事实上，我此前并没有任何嵌入式开发的经验。在最开始拿到板子的时候，甚至不知道要通过串口线连接主板。但是通过开源社区中提供的教程，我很快便掌握了对主板的简单操作。在这里要感谢教程的作者 xieyug2012 以及 shigeng。

小作品——摩斯电码播放器

摩斯码使用不同长短的连续波（嘀和嗒）和间隙来表示 26 个英文字母，10 个数字以及一些标点符号，由于其编码简单、可靠性高，被广泛应用于无线电通讯领域。其简便性使我萌生了利用龙芯主板来将英文文本翻译为摩斯码，并通过扬声器以正弦波播放出来的想法。由于我并没有无线电执照及相应的设备，所以此作品仅限以声音形式播放而不能通过无线电发报，但进行改动，龙芯主板是完全有能力成为全自动发报机的。

硬件部分

此作品电路板的基本系统结构为：使用运算放大器芯片构造一个正弦波发生器，其频率可由一个双联电阻器在约 500Hz 至无限大（实际极限数值由芯片决定，但肯定超出人耳听力范围）之间调整，此正弦波将作为扬声器的声源。扬声器与正弦波之间有一个由开源龙芯主板 GPIO 接口所控制的继电器，用来控制线路的断连，从而达到播放摩斯码的目的。

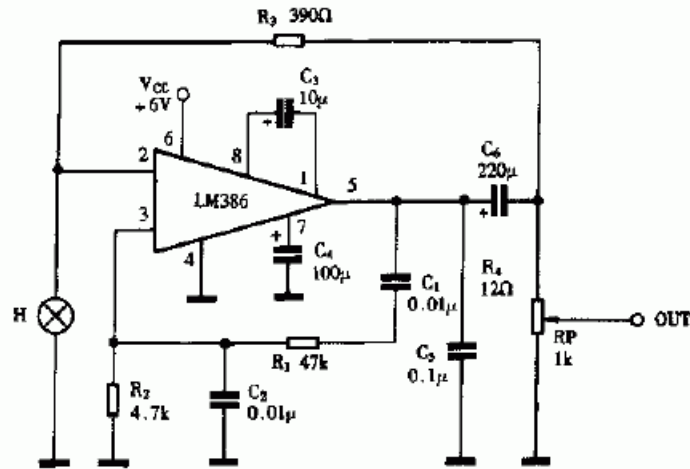
确定了系统结构之后，下一步就要制作电路原理图并确定元件参数。毫无电路设计基础的我在网络上找到了这张原理图作为参考。

A	·—	V	···—
B	—···	W	·— —
C	—·—·	X	—··—
D	—··	Y	—·— —
E	·	Z	— —··
F	··—·	.	···—·—
G	— —·	/	— —·— —
H	····	?	··— —··
I	··	/	— ·— ·
J	·— — —	@	···— — —
K	—·—	1	·— — — —
L	·—··	2	··— — —
M	— —	3	···— —
N	—·	4	····—
O	— — —	5	·····
P	·— —·	6	—····
Q	— —·—	7	— —···
R	·—·	8	— — —··
S	···	9	— — — —·
T	—	0	— — — — —
U	··—		

摩斯电码表

巧用 LM386 作正弦波振荡器

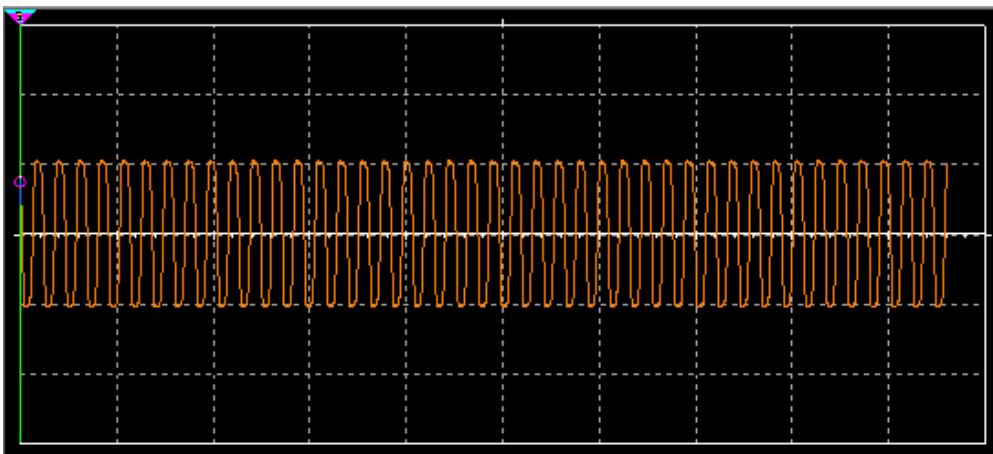
图 是一个利用 LM386 制作而成的正弦波振荡器,电路采用文氏电桥振荡方式,输出信号的失真系数极低。小电珠 H 与电阻 R_3 组成负反馈电路,它使振荡器输出信号的幅值保



持稳定,而且具有较低的失真。当电容 C_1 、 C_2 取值相同时,电路振荡频率可由公式 $f = 1/2\pi C_1 \sqrt{R_1 R_2}$ 求得,采用图示数据时正弦波频率为 1kHz。在实际制作时,H 可选用 3V、15mA 的小电珠。

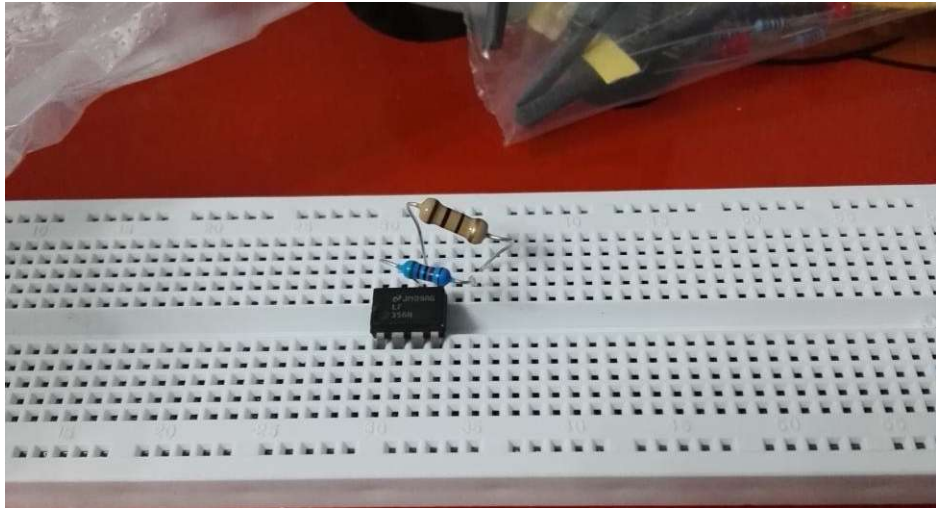
最初所找到的电路图

然而此电路在模拟器中的模拟并不成功。在 QQ 群中 electron 群友的帮助下,将运算放大器改为 LF356N,并对电路做出了大幅度简化及调整,确定了电路构造以及元件参数,并在软件中进行了电路模拟,成功输出了预定频率的正弦波。



电路模拟所得波形:每列表示 10 毫秒,每列有约 5 个波,可见其频率为 500Hz

我住在天津,作为一个大城市购买电子元件并不困难,我很快便购齐了所需的元件及面包板,开始真正的制作。



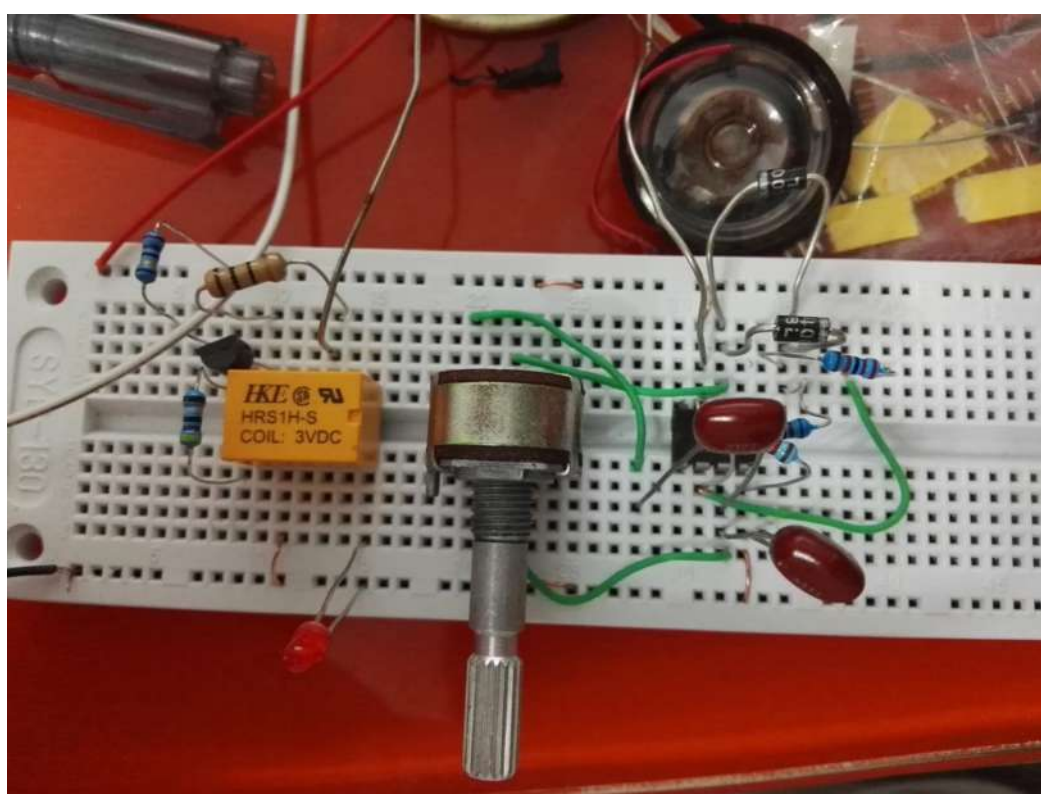
最初的三个元件



杂乱的环境

在实际制作中，我碰到了许多在模拟中没有预料到的问题。首先碰到的问题是 GPIO 无法直接触发继电器。通过查阅所使用的 HRS1H-S 型继电器的数据表，发现其线圈需要约 300mW 的功率来触发，然而龙芯 GPIO 的最大输出功率仅为约 50mW。在 QQ 群中群友的建议下，我使用了一颗 8050 三极管来提升功率，成功触发了继电器。

第二个问题则是吃了不认真和不懂电路原理的亏。当我把一切元件及线路按照电脑自动生成的鼠线连接之后，扬声器没有任何反应。这个问题困扰了我好几天，我一点一点地将面包板上的接线与电脑给出的鼠线比对，希望能发现什么接线错误，然而并没有。而这时我也该回到英国了，于是只能把所有的东西装到盒子里，到了英国再处理。

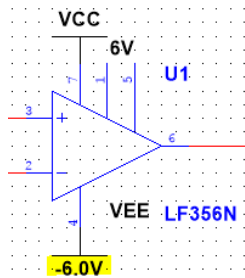


临走之前的情况

到了英国，我就有万用表能用了。通过测试发现，输出端是有直流电压输出的，而且与预期的结果一样，这说明电路并没有断路；但当测试交流电压时，万用表毫无反应，这说明输出的波形是一条直线，而不是正弦波。

既然电路板上没问题，那会不会是电路本身的设计问题？但如果电路设计有问题的话，仿真软件又是如何输出预期的波形的呢？我开始仔细检查起仿真软件

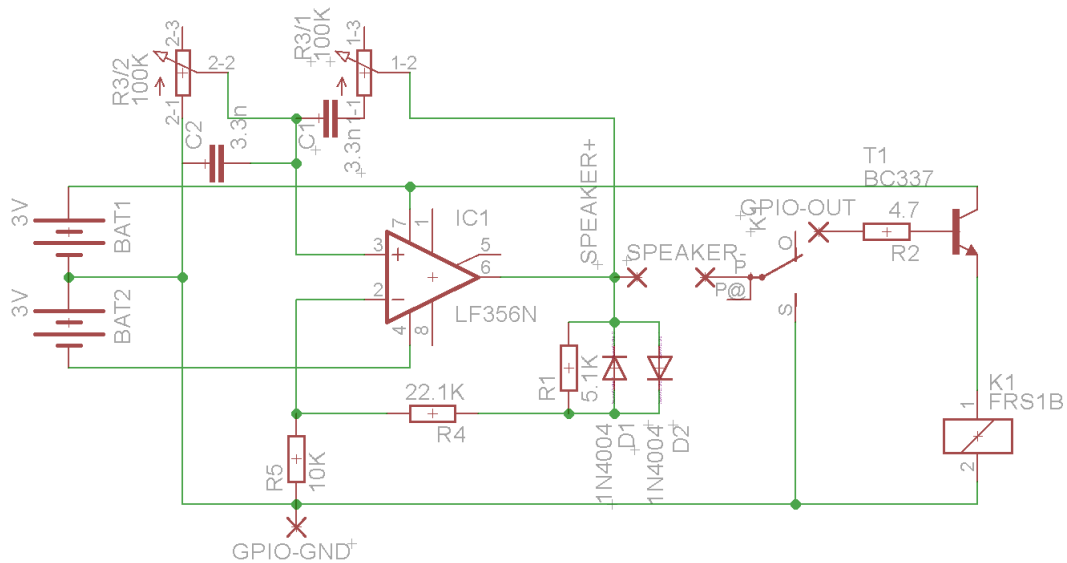
中的电路，最终发现了问题：



对于我这种不懂电路的人来说，这是个很不起眼的细节

问题就出在这个-6.0V 身上。此前我一直是把 4 号针脚接地的，并没有意识到这个负电压的问题。当我在仿真软件中也把 4 号针脚接地后，其输出的波形也变成了一条直线。

依然是在群友的帮助下，我对线路做了一点小改动，将 4*1.5V AA 电池盒的负极接到 4 号针脚而不是地，并把第二节电池的负极当成了地。由此便创造出了一个±3V 的直流电源，但这也导致了输出电压成了原来的一半，因此不再需要起初串联在扬声器上的限流电阻。



最终电路

以上即为最终的电路图。可见由 2 颗 3.3nF 电容和一个最大电阻为 100k Ω 的双联电位器所组成的反馈回路。运放输出端直接连接扬声器正极，扬声器的回路则由继电器所控制。继电器的线圈一边接地，一边与一颗三极管的发射级连接，此三极管的集电极由电池驱动，基极则连接 GPIO 出口和一个限流电阻。5.1k Ω 电阻+2 个 1N4004 二极管并联的设计来自群友 electron，用来限制峰值电压。

最终，扬声器里成功输出了干净的正弦声波，通过调整双联电位器可将其频率调高至人耳听力以外。至于波形实际的噪音水平为多少，最高可达到的频率为多少，由于我并没有示波器，因此无从得知。



面包板上的成品

• 软件部分

以上内容仅是成功的一半而已。有电路只能让喇叭发出正弦波，要想发送摩斯码，还要进行编程。我按照论坛上的教程在 Ubuntu 上搭建了交叉编译环境，所使用的语言是 C。我的 C 语言水平也是半斤八两，许多地方都靠现学现卖。

程序结构如下：主函数将输入的文本存入字符数组并传递给 `morslize` 函数，此函数将输入的字符数组遍历，并将每个字符翻译为一个代表此字符摩斯码的整形数组。比如字母 A 的摩斯码是 嘀 嗒，那么它的数组就是 $\{0, 1\}$ ，字母 C 的摩斯码是 嗒 嘀 嗒 嘀，那么它的数组就是 $\{1, 0, 1, 0\}$ 。总之，用 1 表示嗒，0 表示嘀。此数组再传递给 `play` 函数，用来控制 GPIO 口的电平高低及电平持续时间。控制 GPIO 口的函数及其头文件来自论坛网友 shigeng。他/她并未写明此头文件的版权协议，在此我将其默认为与龙芯开源主板使用相同的版权协议。如有侵权请联系我。

```

□/*****
  > File Name: gpio.h
  > Author: shigeng
  > Mail: shigeng_bj@sohu.com
  > Created Time: 2015/6/27 18:46:40
  *****/

□#ifndef _MY_GPIO_H_
#define _MY_GPIO_H_
extern int gpio_export(int pin);
extern int gpio_unexport(int pin);
extern int gpio_direction(int pin, char *dir);
extern int gpio_direction_get(int pin); //获取当前方向
extern int gpio_direction_in(int pin); //设置当前方向为输入
extern int gpio_direction_out(int pin); //设置当前方向为输出
extern int gpio_value_get(int pin); //获取当前值
extern int gpio_value_set(int pin, int value); //value 可选值: 0, 1
#endif

```

从论坛上获得的控制 GPIO 口的头文件

摩斯码很重要的一点就是嘀、嗒的持续时间以及嘀嗒之间、字母之间以及单词之间间隔的时间。依照标准，一个嘀的长度为一个单位时间，一个嗒的长度为三个单位时间，嘀嗒之间的间隔为一个单位，两个字母之间为三个单位，两个单词之间为七个单位。一个单位时间是由码速所决定的，码速则由 WPM 为单位，意为单词每分钟。这里的单词以“PARIS”一词为标准；因此，码速的完整意思是“在一分钟之内可以使用摩斯码发送多少个‘PARIS’”。由此我们可以得到公式：单位时间(秒)=1.2/WPM。但由于 C 语言中 usleep 函数是以微秒计的，因此在源码中应该用 1200000/WPM 来计算单位时间。

```

printf("请输入wpm数\n");
scanf("%d", &wpm);
dit=1200000/wpm;
dah=3600000/wpm;
wrp=8400000/wpm;

```

时间的计算

程序中涉及到了两次遍历数组，第一次是 morssize 函数遍历所输入的字符。此数组长度未知，因为输入的内容未知。然而通过循环遍历数组是需要知道数组长度的。由于在 C 中数组参数的本质是指针，我们不能在 morssize 函数中简单地使用 sizeof(数组) 来获得长度，因为这里的数组本质是指针。对此我调用了 string.h 头函数中的 strlen 函数来获得长度。

第一个遍历尚好解决，第二个就比较棘手：对于 play 最终的函数来讲，其参数（也就是上文提到的 0 1 数组）长度未知，因为每个字母摩斯码嘀 嗒的数量都不一样。而其数据类型为 int，并非可以使用 strlen 函数的字符数组。对此问题我的解决方法是把嘀嗒数量当成数组的第一个元素传递出去，而在 play 函数遍历的时候跳过第一个元素。

```
int morslize(char o[]) {
    int i;
    for(i=0;i<strlen(o);i++){
        if(o[i]==65|o[i]==97){ //a
            int m[3]={2,0,1};
            play(m);
            usleep(iidit);
        }
        else if(o[i]==66|o[i]==98){ //b
            int m[5]={4,1,0,0,0};
            play(m);
            usleep(iidit);
        }
        else if(o[i]==67|o[i]==99){ //c
            int m[5]={4,1,0,1,0};
            play(m);
            usleep(iidit);
        }
        else if(o[i]==68|o[i]==100){ //d
            int m[4]={3,1,0,0};

```

Morslize 函数的一部分

在 play 函数的具体机理是：如果遍历到的元素是 1，则把 GPIO 口设置为高电平，等待 3 个单位时间（即一个嘀的时间），然后把 GPIO 设置为低电平；如果元素是 0，则同理，不过等待的时间只有一个单位时间。

```
int play(int a[]) {
    int i;
    for(i=1;i<a[0]+1;i++){
        if(a[i]==1){
            gpio_value_set(47,1);
            usleep(dah);
            gpio_value_set(47,0);
        }
        else{
            gpio_value_set(47,1);
            usleep(dit);
            gpio_value_set(47,0);
        }
        usleep(dit);
    }
    return 0;
}
```

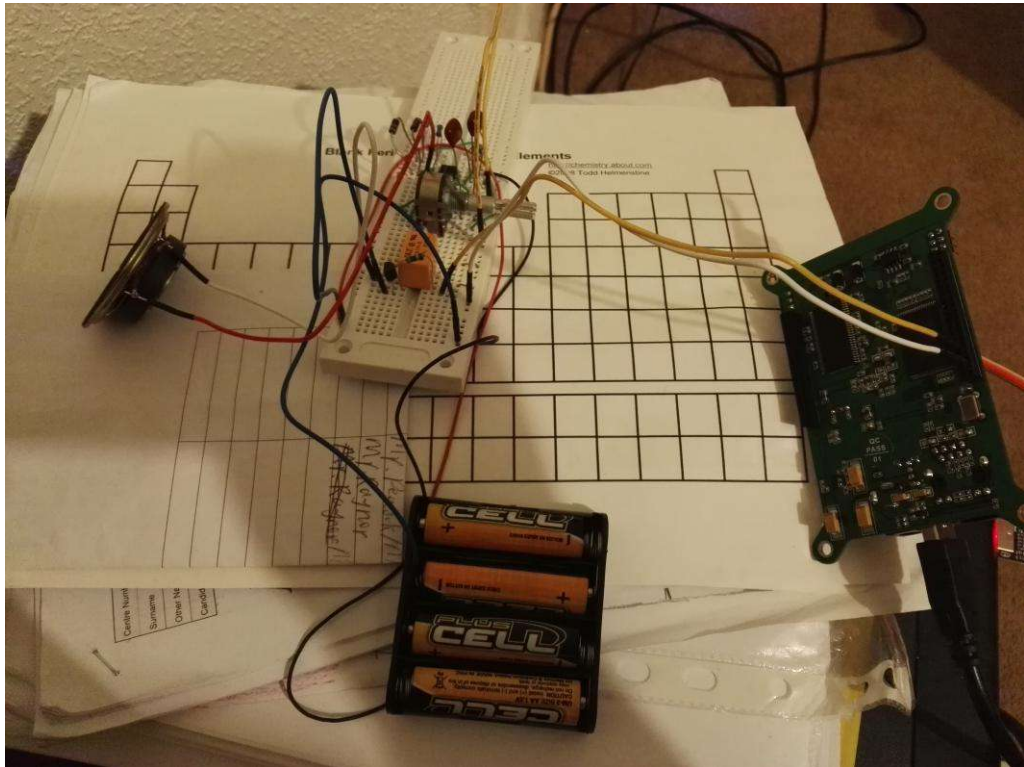
play 函数

程序码好之后,修改一下 shigeng 所提供的 makefile 文件,将源码以及 GPIO 头文件及函数利用 mipsel-linux-gcc 编译为一个独立的、不需要外部依赖的 mipsel 可执行文件,然后通过 ncat 发送到龙芯主板上。

将 GPIO47 口及 GND 与电路连接,执行 morse 程序,输入 WPM 值及要播放的文本,即可听到继电器开闭以及扬声器播放摩斯码的声音。


```
inners@inners-All-Series: ~
lsix-rtc lsix-rtc.0: setting system clock to 2069-03-23 12:21:59 UTC (3131266919)
)
yaffs: dev is 32505857 name is "mtdblock1" rw
yaffs: passed flags ""
VFS: Mounted root (yaffs2 filesystem) on device 31:1.
devtmpfs: mounted
Freeing unused kernel memory: 200k freed
#mount all.....
#Starting mdev.....
Processing /etc/profile.....
Done!
[root@Loongson-gz:~]#ls
bin          helloLS.o   linuxrc     man          root        sys          var
dev          home        lost+found  mnt         sbin        tmp
etc          lib         lynx.tar    proc        share       usr
[root@Loongson-gz:~]#cd home
[root@Loongson-gz:/home]#ls
2048        listprime  morse
[root@Loongson-gz:/home]#./morse
请输入wpm数
30
请输入要翻译的文本
The quick brown fox jumps over the lazy dog
```

运行时界面



全家福。由于我没有笔记本电脑，这些东西只能放在主机上面，空间略显窘迫

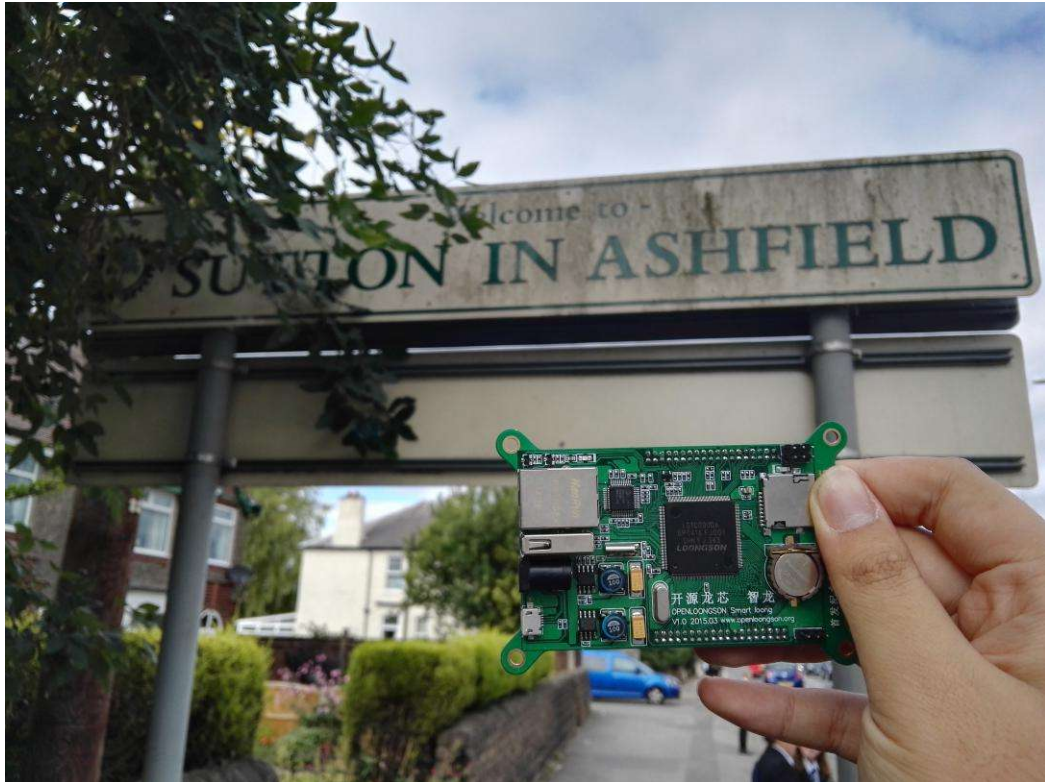
至此，这个在中国开始，在英国完成的项目正式完工了。从构想到完成总共花了约一周多时间。其实这个小项目很简单，对于有电路设计经验的人来说或许三四天就能完成。下一步我可以把这些元件焊到实验板上，甚至设计出 PCB 让厂

家来生产——不过设计 PCB 也不是件简单事，如果我有时间的话可能会研究下。

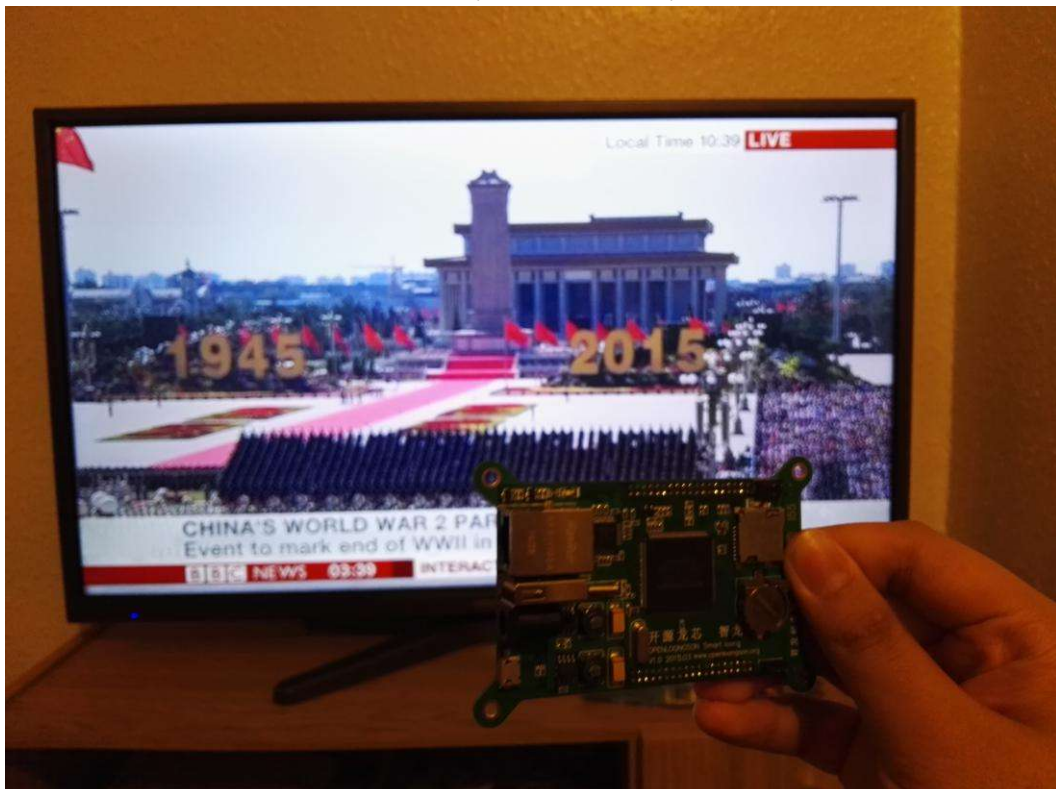
此外，我也应该是第一个将开源龙芯主板带出国门的人。可惜我所在的小镇离著名的地标性建筑很远，因此就让龙芯主板与我小镇上的一些标志合影吧。



英国的红绿灯



我所在的小镇(Sutton in Ashfield)的路牌



BBC 直播的阅兵



消防队



我的学校 我在上 11 年级，相当于国内高一



社区

原文地址：<http://www.loongsonclub.com/bbs/portal.php?mod=view&aid=55>