

第一章 RFID 基础实验

1.1 RFID 模块测试实验

1.1.1 实验目的

- 1、熟悉单片机和 RC522RFID 模块；
- 2、学习射频卡的工作原理；
- 3、学习 RC522 的工作原理。

1.1.2 实验设备

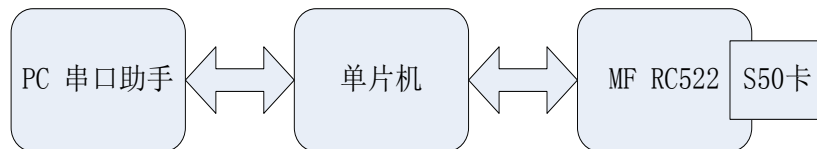
- 1、单片机模块一个
- 2、标准串口线一根
- 3、RFID 电源供电 USB 线一根
- 4、S50 卡一张

1.1.3 准备知识

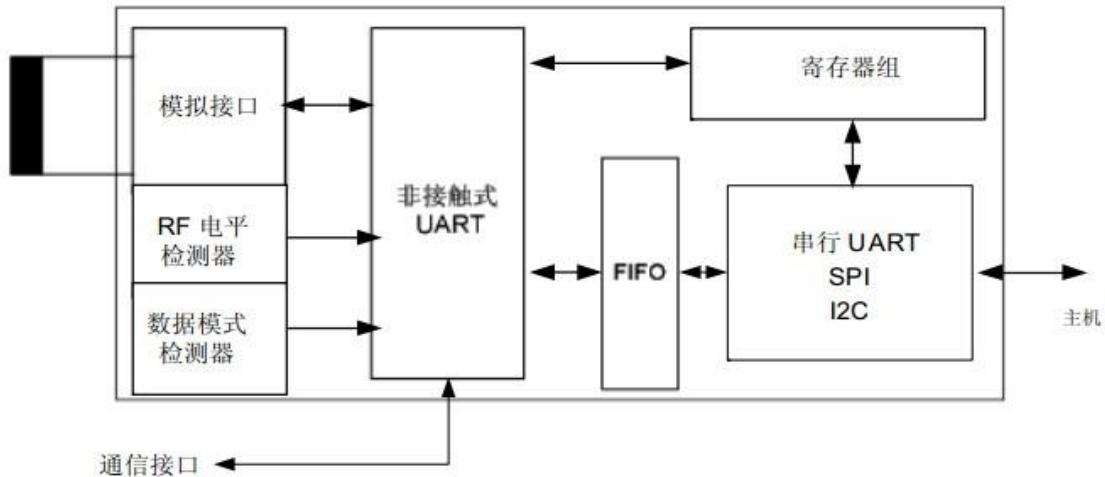
阅读 MF RC522 PDF 资料，重点学习 RC522 的主要寄存器和命令集，并了解兼容的 SPI 接口和定时器等其他部分的使用。

1.1.4 实验原理

通过串口助手发送命令，控制 RFID 模块执行相应的读写修改操作，基本框架如图如下：

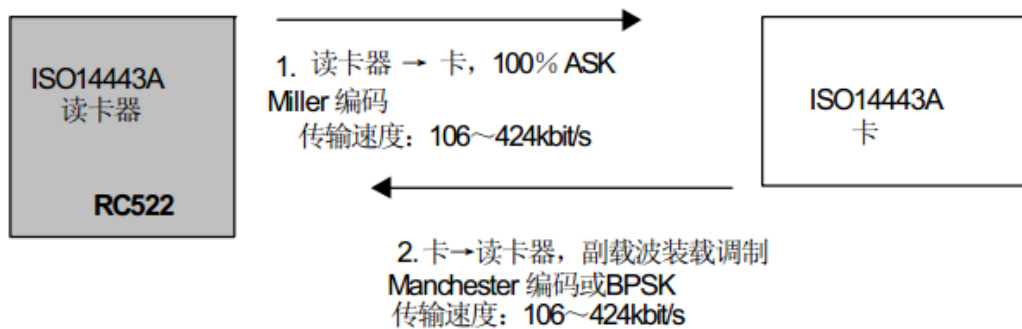


MFRC522 是高度集成的非接触式（13.56MHz）读写卡芯片，支持 ISO 14443A/MIFARE，支持 SPI 接口、串行 UART 和 I2C，本模块使用的是 SPI 接口，简化 MFRC522 框图如下：



寄存器的详细配置请参照 MF RC522 PDF 资料，这里就不详细列出。

ISO 14443A/MIFARE 读写器功能的通信图：



MFRC522 的操作由可执行一系列命令的内部状态机来决定，通过寄存器写入相应的命令代码来启动命令。执行一个命令所需要的参数和数据是通过 FIFO 缓冲区来交换的。

命令概述如下：

命令	命令代码	动作
Idle	0000	无动作；取消当前命令的执行。
CalcCRC	0011	激活 CRC 协处理器或执行自测试。
Transmit	0100	发送 FIFO 缓冲区的命令。
NoCmd Change	0111	无命令改变。该命令用来修改命令寄存器的不同位，但又不触及其它命令，如掉电。
Receive	1000	激活接收器电路。
Transceive	1100	如果寄存器 ControlReg 的 Initiator 位被设为 1： 将 FIFO 缓冲区的数据发送到天线并在发送完成后自动激活接收器。 如果寄存器 ControlReg 的 Initiator 位被设为 0： 接收天线的数据并自动激活发送器。
MFAuthent	1110	执行读卡器的 MIFARE 标准认证。
Soft Reset	1111	复位 MFRC522。

S50 卡总共对应应有 16 个扇区，03、07、11、15、19、23、27、31、35、39、43、57、51、55、59、63 分别对应 1-16 扇区的密钥存储块，密钥存放格式如下：

字节	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	密码 A						权限位				密码 B					
	位: 7 6 5 4 3 2 1 0															
6	$\overline{C2}_3$		$\overline{C2}$		$\overline{C2}_1$		$\overline{C2}_0$		$\overline{C1}_3$		$\overline{C1}_2$		$\overline{C1}_1$		$\overline{C1}_0$	
7	$C1_3$		$C1_2$		$C1_1$		$C1_0$		$\overline{C3}_3$		$\overline{C3}_2$		$\overline{C3}_1$		$\overline{C3}_0$	
8	$C3_3$		$C3_2$		$C3_1$		$C3_0$		$C2_3$		$C2_2$		$C2_1$		$C2_0$	
9																

密钥块访问：

权限代码			访问权限						说明
			密码 A		权限字节		密码 B		
C1 ₃	C2 ₃	C3 ₃	读	写	读	写	读	写	—
0	0	0	N	A	A	N	A	A	密码 B 可读
0	1	0	N	N	A	N	A	N	密码 B 可读
1	0	0	N	B	A/B	N	N	B	—
1	1	0	N	N	A/B	N	N	N	—
0	0	1	N	A	A	A	A	A	密码 B 可读
0	1	1	N	B	A/B	B	N	B	—
1	0	1	N	N	A/B	B	N	N	—
1	1	1	N	N	A/B	N	N	N	—

数据块访问 ($i = 0,1,2$):

权限代码			访问权限				应用
$C1_i$	$C2_i$	$C3_i$	读	写	增值	减值	
0	0	0	A/B	A/B	A/B	A/B	空卡默认状态
0	1	0	A/B	N	N	N	读写块
1	0	0	A/B	B	N	N	读写块
1	1	0	A/B	B	B	A/B	数值块
0	0	1	A/B	N	N	A/B	数值块
0	1	1	B	B	N	N	读写块
1	0	1	B	N	N	N	读写块
1	1	1	N	N	N	N	读写块

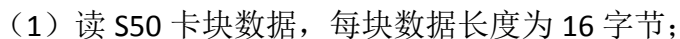
1.1.5 实验步骤

- 1、标准串口线一根将单片机模块与 PC 串口模块连接，单片机和 MF RC522 模块连接，PC 通过 USB 方口线给单片机供电；
- 2、打开串口助手 CommAssistant.exe；

读 S50 卡序列号。

在发送框输入：02 A0;

其中 02 表示发送数据长度为 2 个字节，A0（A 零）表示读序列号功能。



在发送框输入：09 A1 FF FF FF FF FF FF 02

其中 09 表示发送数据长度 9 个字节, A1 表示读数据块功能, FF FF FF FF FF FF 表示六字节密钥(初始默认密钥为 FF FF FF FF FF FF), 如果密钥被修改, 请将 FF FF FF FF FF FF 替换为新密钥, 02 表示读第三(从 00 开始计, 02 表示第三块)块数据(共 64 块, 0-63)。

返回 16 字节数据块和 4 字节序列号（初始值可能会存在不同）。



(2) 写 S50 卡块数据，每块数据长度为 16 字节；

在发送框输入：19 A2 FF FF FF FF FF FF 02 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16；

其中 19 (16 进制) 表示 25 字节，A2 表示写数据块，FF FF FF FF FF FF 表示六字节密钥 (初始默认密钥为 FF FF FF FF FF FF)，如果密钥被修改，请将 FF FF FF FF FF FF 替换为新密钥。02 表示读第三 (从 00 开始计，02 表示第三块) 块数据，**请不要随意写(04*n-01，即 03, 07, 11, 15, 19, 23, 27, 31, 35, 39, 43, 57, 51, 55, 59, 63 因为这些块存放的是密钥，如果误操作，如 19 A2 FF FF FF FF FF FF 03 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16**

将会导致第一个扇区 (00-03 块) 不可使用，S50 卡一共有 16 个扇区，其余 15 个扇区仍然可用。后续还会进行二次开发，通过判断来避免误操作，此处只是初始演示学习。

返回 16 字节数据块和 4 字节序列号。



(3) 修改 S50 密钥第一个扇区密钥：

在发送框输入：0F A3 FF FF FF FF FF FF 03 00 00 00 00 00 00

其中 0F 表示发送数据长度 15 个字节，A3 表示修改密钥，FF FF FF FF FF FF 表示六字节密钥 (初始默认密钥为 FF FF FF FF FF FF)，如果密钥被修改，请将 FF FF FF FF FF FF 替换为新密钥，03 块存放的是第一块扇区的密钥，00 00 00 00 00 00 表示修改后的新密钥。请务必记住新密钥，如果密钥忘记，那对应的扇区将被锁死。

(4) 在进行写数据、修改密码操作后，在发送框输入：09 A1 00 00 00 00 00 00 02，此时便可以读取到写入的数据。如图所示：

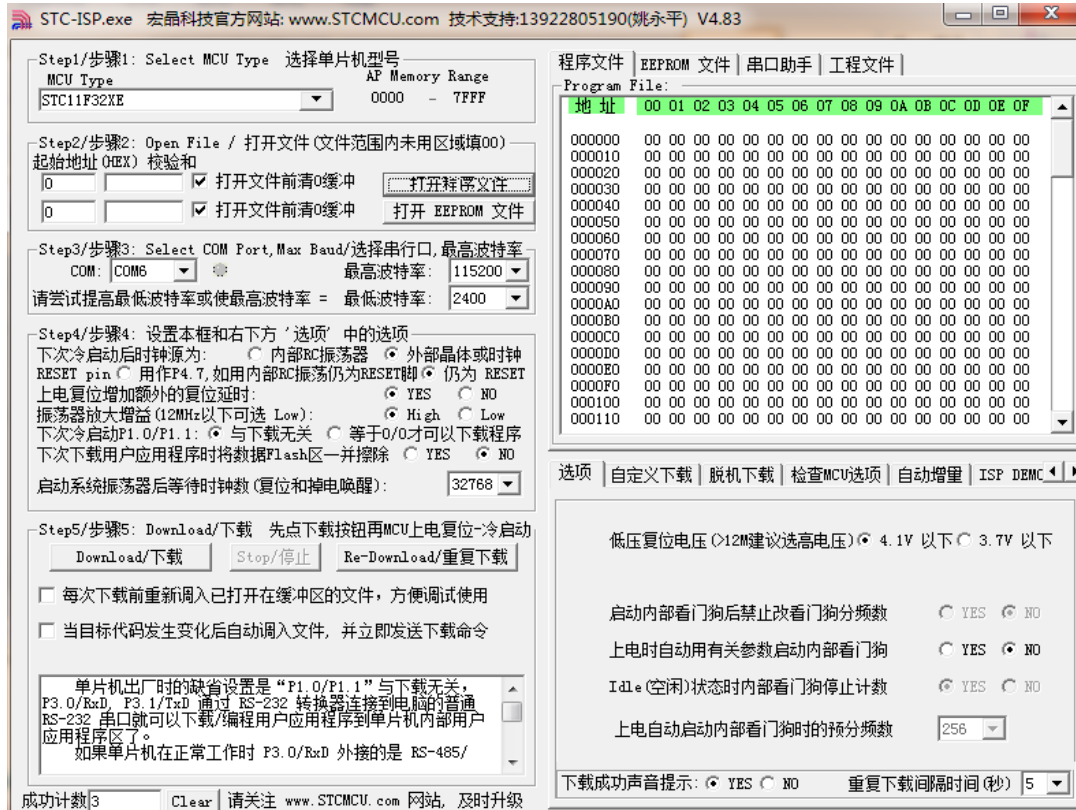


1.1.6 注意事项

1、RFID 模块默认情况下，已经烧录好程序。**实验前，先确认程序烧录正确。**若实验过程中，读写数据不正常，芯片可能是未烧录程序。此时，先将 RFID 模块通过串口与 PC 相连，通过方口 USB 供电，打开光盘目录“辅助工具/STC 烧录”下 STC_ISP_V483 软件，将“辅助工具/STC 烧录”目录下的 MFRC522_STC_NO12864.hex 文件烧录进单片机。

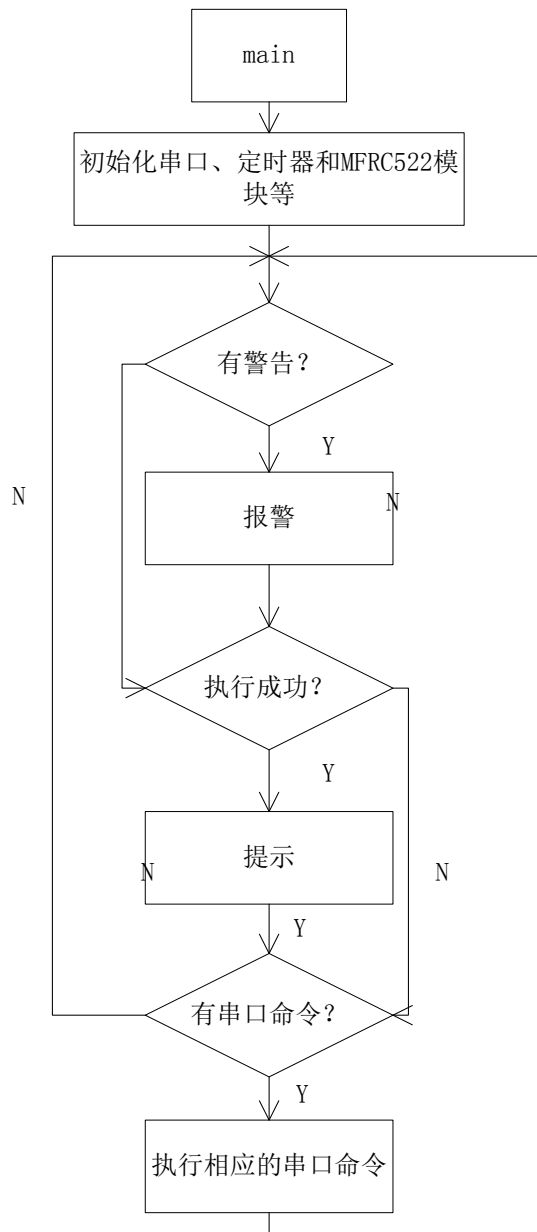
程序烧录下载过程中需要注意：

- (1) 选择 MCU Type 为 STC11F32XE
- (2) 打开 hex 文件
- (3) 选择正确的 COM 口（可在“我的电脑/属性/设备管理器”中查看到）
- (4) 其他全部默认
- (5) 板子断电
- (6) 点击 Download/下载
- (7) 板子上电
- (8) 等待下载完成



2、请勿随意修改密钥，如果修改密钥，请务必记住，如果密钥忘记，那么该密钥对应的扇区将被锁死。S50 卡总共对应 16 个扇区，03、07、11、15、19、23、27、31、35、39、43、57、51、55、59、63 分别对应 1-16 扇区的密钥存储块。

1.1.7 程序框图



1.1.8 模块附带代码解析

单片机通过 SPI 读写 MFRC522 模块：

```

/*SPI 读*/
unsigned char SPIReadByte(void)
{
    unsigned char data SPICount;
    unsigned char data SPIData;
}
    
```



```
SPIData = 0;
for (SPICount = 0; SPICount < 8; SPICount++)
{
    SPIData <<=1;
    CLR_SPI_CK; nop();nop();
    if(STU_SPI_MISO)
    {
        SPIData |=0x01;
    }
    SET_SPI_CK; nop();nop();
}
return (SPIData);
}

/*SPI 写*/
void SPIWriteByte(unsigned char data SPIData)
{
    unsigned char data SPICount;
    for (SPICount = 0; SPICount < 8; SPICount++)
    {
        if (SPIData & 0x80)
        {
            SET_SPI_MOSI;
        }
        else
        {
            CLR_SPI_MOSI;
        }
        nop();nop();
        CLR_SPI_CK;nop();nop();
        SET_SPI_CK;nop();nop();
        SPIData <<= 1;
    }
}
```

通过 RC522 和 ISO14443 卡通讯

```
////////////////////////////////////
//功    能：通过 RC522 和 ISO14443 卡通讯
//参数说明：Command[IN]:RC522 命令字
//           pInData[IN]:通过 RC522 发送到卡片的数据
//           InLenByte[IN]:发送数据的字节长度
//           pOutData[OUT]:接收到的卡片返回数据
//           *pOutLenBit[OUT]:返回数据的位长度
////////////////////////////////////
char PcdComMF522(unsigned char data Command,
                 unsigned char *pInData,
```

```

        unsigned char InLenByte,
        unsigned char *pOutData,
        unsigned int *pOutLenBit)
{
    char data status = MI_ERR;
    unsigned char data irqEn    = 0x00;
    unsigned char data waitFor = 0x00;
    unsigned char data lastBits;
    unsigned char data n;
    unsigned int data i;
    switch (Command)
    {
        case PCD_AUTHENT:
            irqEn    = 0x12;    /*允许空闲中断请求，允许错误中断请求*/
            waitFor = 0x10;    /* 等待命令自行结束*/
            break;
        case PCD_TRANSCEIVE:
            irqEn    = 0x77;    /*允许发送器、接收器、空闲、报警、错误和定时器中断请求*/
            waitFor = 0x30;    /* 等待接收一个有效数据流或命令自行结束*/
            break;
        default:
            break;
    }

    WriteRawRC(ComIrqReg,irqEn|0x80);    /*开启中断，IRQ 上的信号与 Status1Reg 的 IRq 位的值相反*/
    ClearBitMask(ComIrqReg,0x80);    /*开启 IRQ 中断*/
    WriteRawRC(CommandReg,PCD_IDLE);    /*使 MFRC522 处于空闲模式，终止正在执行的命令*/
    SetBitMask(FIFOLevelReg,0x80);    /*清空 FIFO 缓存*/

    for (i=0; i<InLenByte; i++)
    {
        WriteRawRC(FIFODataReg, pInData[i]);    }    /*将发送数据写入 FIFO*/
    WriteRawRC(CommandReg, Command);    /*启动命令*/

    if (Command == PCD_TRANSCEIVE)
    {
        SetBitMask(BitFramingReg,0x80);    }    /*启动数据的发送*/

    //i = 600;//根据时钟频率调整，操作 M1 卡最大等待时间 25ms
    i = 2000;
    do
    {

```

```

        n = ReadRawRC(ComIrqReg);
        i--;
    }
    while ((i!=0) && !(n&0x01) && !(n&waitFor));
    ClearBitMask(BitFramingReg,0x80);

    if (i!=0)
    {
        if(!(ReadRawRC(ErrorReg)&0x1B))    /*检测是否 FIFO 溢出,是否冲突,
        是否奇偶校验错误, 是否协议错误*/
        {
            status = MI_OK;
            if (n & irqEn & 0x01)    /*检测定时器是否超时*/
            {
                status = MI_NOTAGERR;
            }
            if (Command == PCD_TRANSCEIVE)
            {
                n = ReadRawRC(FIFOLevelReg);
                lastBits = ReadRawRC(ControlReg) & 0x07;
                if (lastBits)
                {
                    *pOutLenBit = (n-1)*8 + lastBits;
                }
                else
                {
                    *pOutLenBit = n*8;
                }
                if (n == 0)
                {
                    n = 1;
                }
                if (n > MAXRLEN)
                {
                    n = MAXRLEN;
                }
                for (i=0; i<n; i++)
                {
                    pOutData[i] = ReadRawRC(FIFODataReg);
                }
            }
        }
        else
        {
            status = MI_ERR;
        }
    }

    SetBitMask(ControlReg,0x80);    // stop timer now
    WriteRawRC(CommandReg,PCD_IDLE);    /*使 MFRC522 处于空闲模式,终止
    正在执行的命令*/
    return status;
}

```

MFRC522 操作函数

```

char PcdReset(void);    /*复位 RC522*/
char PcdRequest(unsigned char req_code,unsigned char *pTagType);    /*寻卡函

```

```

数*/
void PcdAntennaOn(void);      /*开启天线*/
void PcdAntennaOff(void);    /*关闭天线*/
char M500PcdConfigISOType(unsigned char type); /*设置工作方式*/
char PcdAnticoll(unsigned char *pSnr); /*防冲撞*/
char PcdSelect(unsigned char *pSnr); /*选定卡片*/
char PcdAuthState(unsigned char auth_mode,unsigned char addr,unsigned char
*pKey,unsigned char *pSnr); /*验证卡片密码*/
char PcdWrite(unsigned char addr,unsigned char *pData); /*写数据数据到 M1
卡*/
char PcdRead(unsigned char addr,unsigned char *pData); /*读 M1 卡数据*/
char PcdHalt(void); /*命令卡片进入休眠*/

```

串口接收数据并转换为相应的命令:

```

串口接收数据
void uart_interrupt_receive(void) interrupt 4
{
    uchar data R_Char;
    if(RI==1);
    {
        RI=0;
        WaitTimes=0;
        R_Char=SBUF;
        RevBuffer[uart_count]=R_Char;
        uart_count++;
        if(uart_count==RevBuffer[0])
        {
            uart_comp=1;
        }
    }
}

```

将串口接收的数据转换为相应的命令

```

void ctrl_uart(void)
{
    uchar ii;
    switch(RevBuffer[1])
    {
        case 0xa0:
            oprationcard=SENDID;
            break;
        case 0xa1://读数据
            oprationcard=READCARD;
            for(ii=0;ii<6;ii++)
            {

```

```
    PassWd[ii]=RevBuffer[ii+2];
}
    KuaiN=RevBuffer[8];
    break;
case 0xa2://写数据
    oprationcard=WRITECARD;
    for(ii=0;ii<6;ii++)
    {
        PassWd[ii]=RevBuffer[ii+2];
    }
    KuaiN=RevBuffer[8];
    for(ii=0;ii<16;ii++)
    {
        WriteData[ii]=RevBuffer[ii+9];
    }
    break;
case 0xa3://修改密码
    oprationcard=KEYCARD;
    for(ii=0;ii<6;ii++)
    {
        PassWd[ii]=RevBuffer[ii+2];
    }
    KuaiN=RevBuffer[8];
    for(ii=0;ii<6;ii++)
    {
        NewKey[ii]=RevBuffer[ii+9];
        NewKey[ii+10]=RevBuffer[ii+9];
    }
    break;
case 0xa4://消费
    oprationcard=CONSUME;
    for(ii=0;ii<6;ii++)
    {
        PassWd[ii]=RevBuffer[ii+2];
    }
    KuaiN=RevBuffer[8];
    for(ii=0;ii<16;ii++)
    {
        WriteData[ii]=RevBuffer[ii+9];
    }
    break;
case 0xa5://充值
    oprationcard=ADDMONEY;
    for(ii=0;ii<6;ii++)
```

```
{
    PassWd[ii]=RevBuffer[ii+2];
}
KuaiN=RevBuffer[8];
for(ii=0;ii<16;ii++)
{
    WriteData[ii]=RevBuffer[ii+9];
}
break;
default:
    break;
}
uart_comp=0;
uart_count=0;
}
```

主函数:

```
void main(void)
{
    InitAll(); /*初始化端口, RC522,定时器和串口*/
    while(1)
    {
        /*获取按键*/
        /*KeyNum=GetKey();
        if(KeyNum==N_1)
        {
            KeyTime=15;
            sendchar1(0x11);
            oprationcard=SENDID;
        }*/

        if(bWarn)
        {
            bWarn=0;
            Warn();
        }
        if(bPass)
        {
            bPass=0;
            Pass();
        }
        if(uart_comp)
        {
            ctrl_uart();
            ctrlprocess();
        }
    }
}
```

```
}

/*执行系统任务*/
/*if(SysTime>=2)
{
    SysTime=0;
    ctrlprocess();
}*/
}
}
```

1.1.9 实验总结

完成本实验，能够比较完整的学习如何控制使用 MFRC522 芯片，能够对 S50 卡读写，并修改 S50 密码，掌握本套 RFID 模块的使用，为后续二次开发打下基础。

1.1.10 实验思考

本次实验，只是涉及到基本的卡片读写操作，并未进行完善保护机制，请大家思考，如思考如何完善本实验。在下一章实验室，将在本实验的基础之上进行二次开发，使操作更加简单。

1.2 上位机读写 RFID 模块实验

1.2.1 实验目的

- 1、熟悉 window 串口编程；
- 3、熟悉 linux 串口编程；
- 3、通过串口读写 RFID 模块。

1.2.2 实验设备

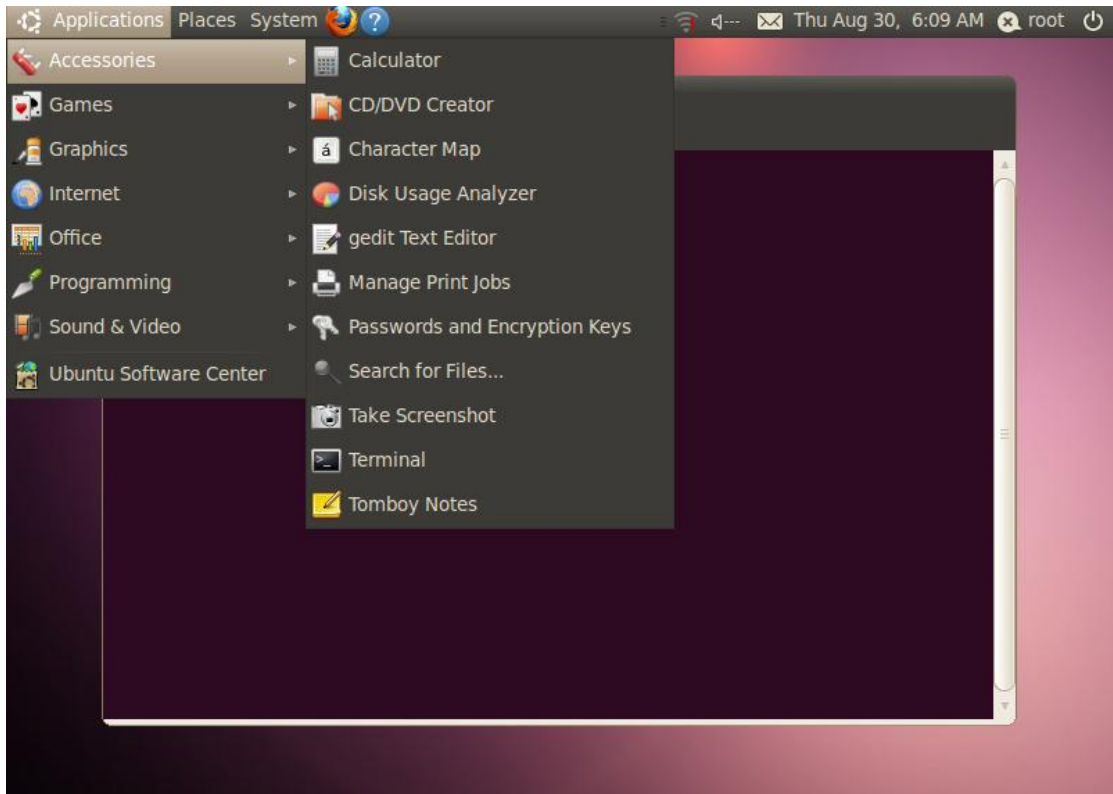
- 1、单片机模块一个
- 2、标准串口线一根
- 3、RFID 电源供电 USB 线一根
- 4、S50 卡一张
- 5、ARM 开发板
- 6、ARM 开发板配套交叉串口线一根

- 7、SD 卡一张
- 8、读卡器一个

1.2.3 准备开发环境

安装 VS2010 开发环境, 安装 VMware 虚拟机, 在虚拟机下安装 Ubuntu11.10, 在 Ubuntu11.10 下安装交叉编译环境。

安装好 Ubuntu11.10, 打开终端。



键入命令 `sudo passwd root` 可以启动 root 登录, 注销后, 使用 root 根用户登录。(注: 此处可参照 ATOS 使用手册第四章 4.2)

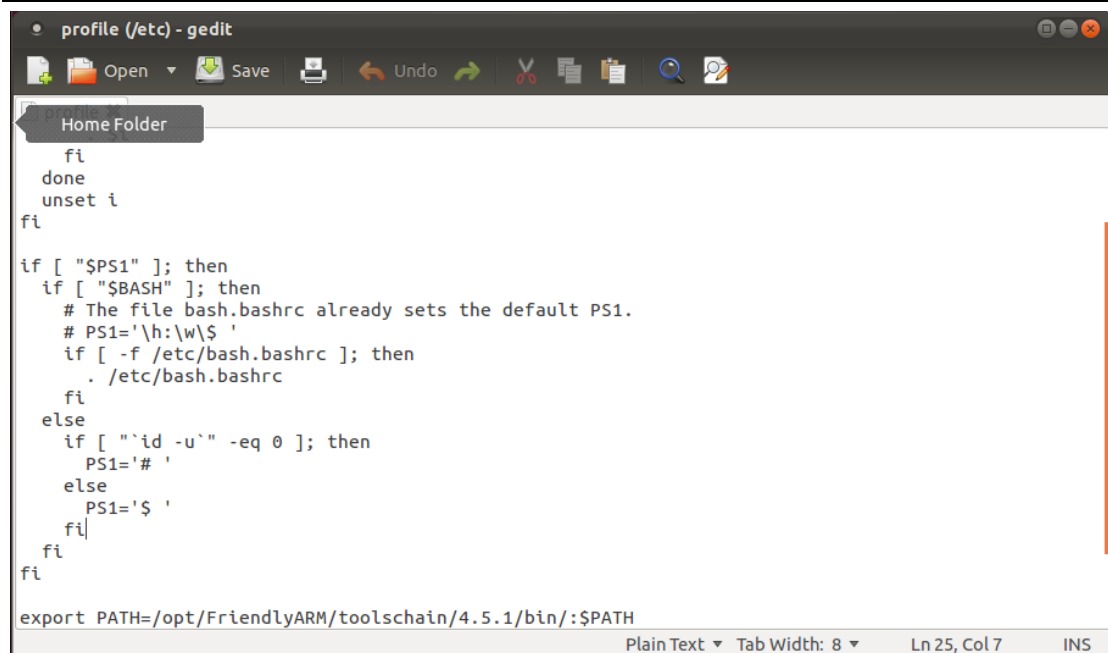
安装适合 ARM 开发板的交叉编译环境, 将光盘“Linux 平台/开发工具”文件夹下的 `arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz` 复制到 ubuntu 某个目录下如 `tmp/`, 然后进入到该目录, 执行解压命令:

```
#cd /tmp
```

```
#tar xvzf arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz -C /
```

执行该命令, 将把 `arm-linux-gcc` 安装到 `/opt/FriendlyARM/toolschain/4.5.1` 目录。

执行 `cd /etc`, 再执行 `gedit profile`, 在 `profile` 文件中, 添加一条 `export PATH=/opt/FriendlyARM/toolschain/4.5.1/bin/:$PATH`。



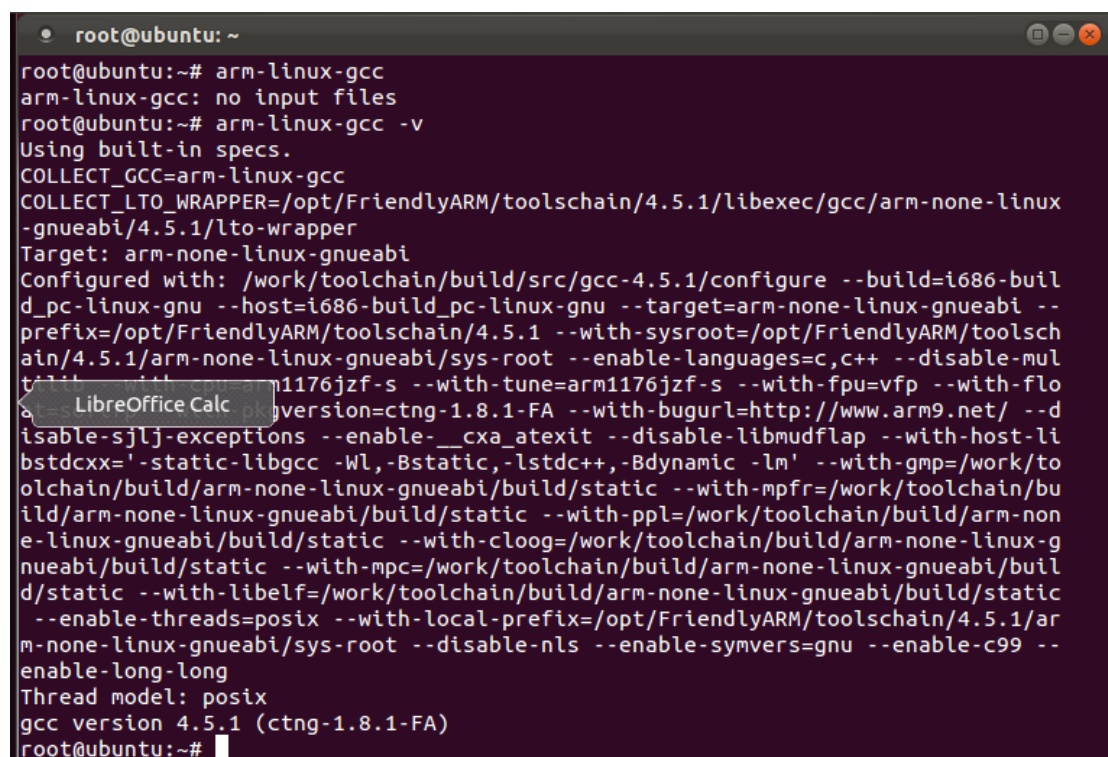
```

profile (/etc) - gedit
Home Folder
fi
done
unset i
fi
if [ "$PS1" ]; then
  if [ "$BASH" ]; then
    # The file bash.bashrc already sets the default PS1.
    # PS1='\h:\w\$ '
    if [ -f /etc/bash.bashrc ]; then
      . /etc/bash.bashrc
    fi
  else
    if [ "`id -u`" -eq 0 ]; then
      PS1='# '
    else
      PS1='$ '
    fi
  fi
fi
export PATH=/opt/FriendlyARM/toolschain/4.5.1/bin/:$PATH

```

最后，直接执行命令 `source /etc/profile`（或者注销重新登录），就可以使用 `arm-linux-gcc` 命令了。

执行命令：`arm-linux-gcc -v`



```

root@ubuntu: ~
root@ubuntu:~# arm-linux-gcc
arm-linux-gcc: no input files
root@ubuntu:~# arm-linux-gcc -v
Using built-in specs.
COLLECT_GCC=arm-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolschain/4.5.1/libexec/gcc/arm-none-linux-gnueabi/4.5.1/lto-wrapper
Target: arm-none-linux-gnueabi
Configured with: /work/toolchain/build/src/gcc-4.5.1/configure --build=i686-build_pc-linux-gnu --host=i686-build_pc-linux-gnu --target=arm-none-linux-gnueabi --prefix=/opt/FriendlyARM/toolschain/4.5.1/arm-none-linux-gnueabi/sys-root --enable-languages=c,c++ --disable-multilib --with-tune=arm1176jzf-s --with-fpu=vfp --with-flo
LibreOffice Calc gcc version=ctng-1.8.1-FA --with-bugurl=http://www.arm9.net/ --d
isable-sjlj-exceptions --enable-__cxa_atexit --disable-libmudflap --with-host-libstdcxx=-static-libgcc -Wl,-Bstatic,-lstdc++,-Bdynamic -lm' --with-gmp=/work/to
olchain/build/arm-none-linux-gnueabi/build/static --with-mpfr=/work/toolchain/bu
ild/arm-none-linux-gnueabi/build/static --with-ppl=/work/toolchain/build/arm-non
e-linux-gnueabi/build/static --with-cloog=/work/toolchain/build/arm-none-linux-g
nueabi/build/static --with-mpc=/work/toolchain/build/arm-none-linux-gnueabi/bui
ld/static --with-libelf=/work/toolchain/build/arm-none-linux-gnueabi/build/static
--enable-threads=posix --with-local-prefix=/opt/FriendlyARM/toolschain/4.5.1/ar
m-none-linux-gnueabi/sys-root --disable-nls --enable-symvers=gnu --enable-c99 --
enable-long-long
Thread model: posix
gcc version 4.5.1 (ctng-1.8.1-FA)
root@ubuntu:~#

```

交叉编译环境编译安装完成。

1.2.4 实验原理

本实验主要是编程设置串口，通过串口控制发送命令，使得 RFID 模块按照

程序设定工作。Window 和 linux 串口的设置有比较大的差异，下面会根据程序具体解析。

1.2.5 实验步骤

1、Window 环境下，打开 RFID 基础实验\上位机读写 RFID 模块实验\RfidSerialDemo 文件夹，打开 RfidSerialDemo.sln。

2、执行“开始执行”（不调试）。注意：如果串口没有连接好，程序会自动退出，本实验默认的串口是串口 4，如果不是请自行在 main.cpp 文件下修改。

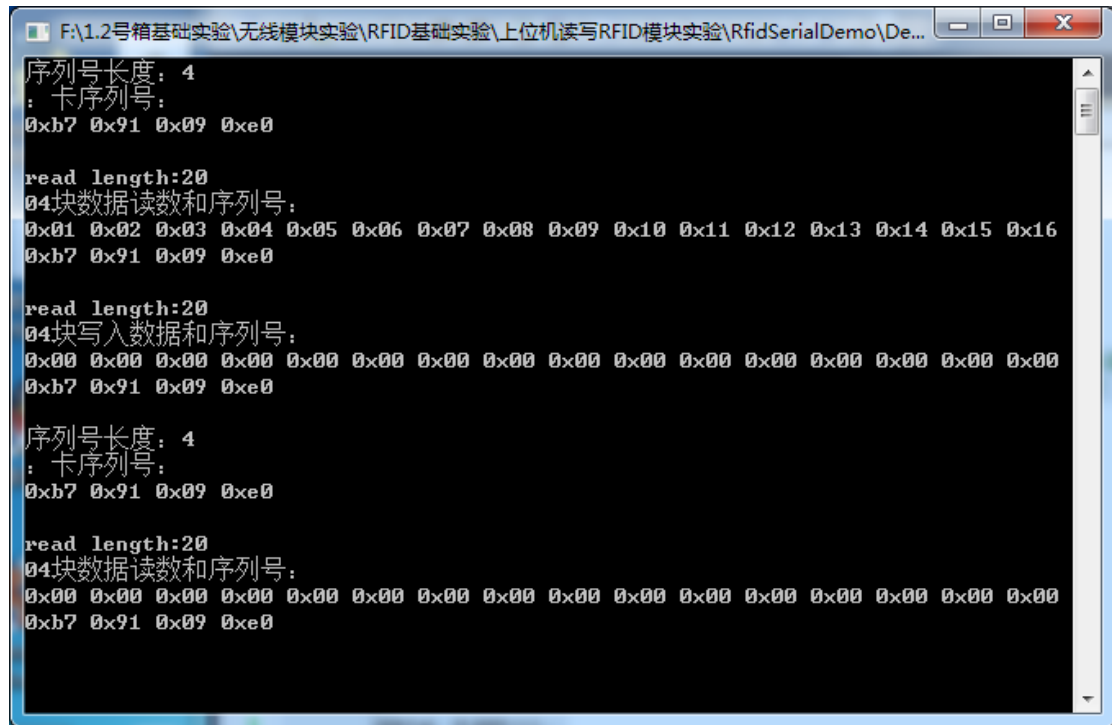
```
using namespace std;

#ifdef _WIN32    /*在windows环境下*/
    #include "Windows.h"    /*windows的头文件*/
    #define SERIAL_SLEEP(n)    Sleep(n*1000)    /*休眠n秒*/
    #define WSN_APP_PORT    "\\.\COM4"    /*windows下的串口4*/
#else    /*在linux环境下*/
    #include <unistd.h>    /*linux下的头文件*/
    #define SERIAL_SLEEP(n)    usleep(n*1000000)    /*休眠n秒*/
    #define WSN_APP_PORT    "/dev/ttySAC1"    /*linux下的串口1*/
#endif

#define WSN_APP_BAUD    9600    /*设置串口波特率*/
```



3、读取卡上数据。



```

序列号长度: 4
: 卡序列号:
0xb7 0x91 0x09 0xe0

read length:20
04块数据读数和序列号:
0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x10 0x11 0x12 0x13 0x14 0x15 0x16
0xb7 0x91 0x09 0xe0

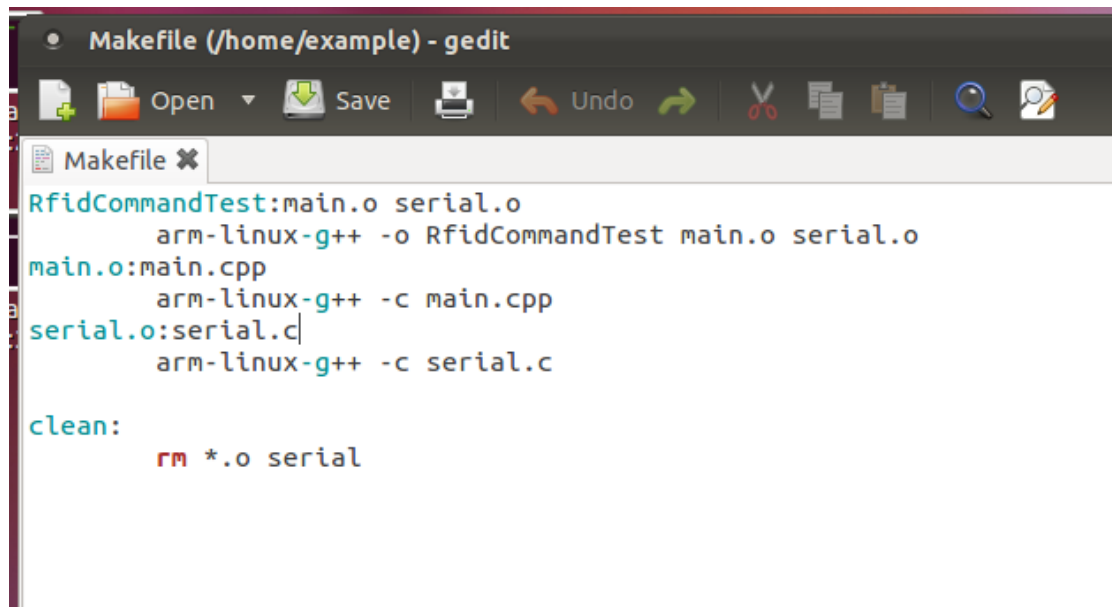
read length:20
04块写入数据和序列号:
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0xb7 0x91 0x09 0xe0

序列号长度: 4
: 卡序列号:
0xb7 0x91 0x09 0xe0

read length:20
04块数据读数和序列号:
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0xb7 0x91 0x09 0xe0
  
```

4、Linux 环境下，将 RFID 基础实验\上位机读写 RFID 模块实验\RfidSerialDemo\RfidSerialDemo 文件夹下的文件 main.cpp、serial.c、serial.h、serial_linux.c.h 和 serial_win32.c.h 复制到 linux 下的文件夹，如/home/example。在该文件夹下编写 Makefile 文件，并保存。

Makefile 文件如下：



```

Makefile (/home/example) - gedit

RfidCommandTest:main.o serial.o
    arm-linux-g++ -o RfidCommandTest main.o serial.o
main.o:main.cpp
    arm-linux-g++ -c main.cpp
serial.o:serial.c
    arm-linux-g++ -c serial.c

clean:
    rm *.o serial
  
```

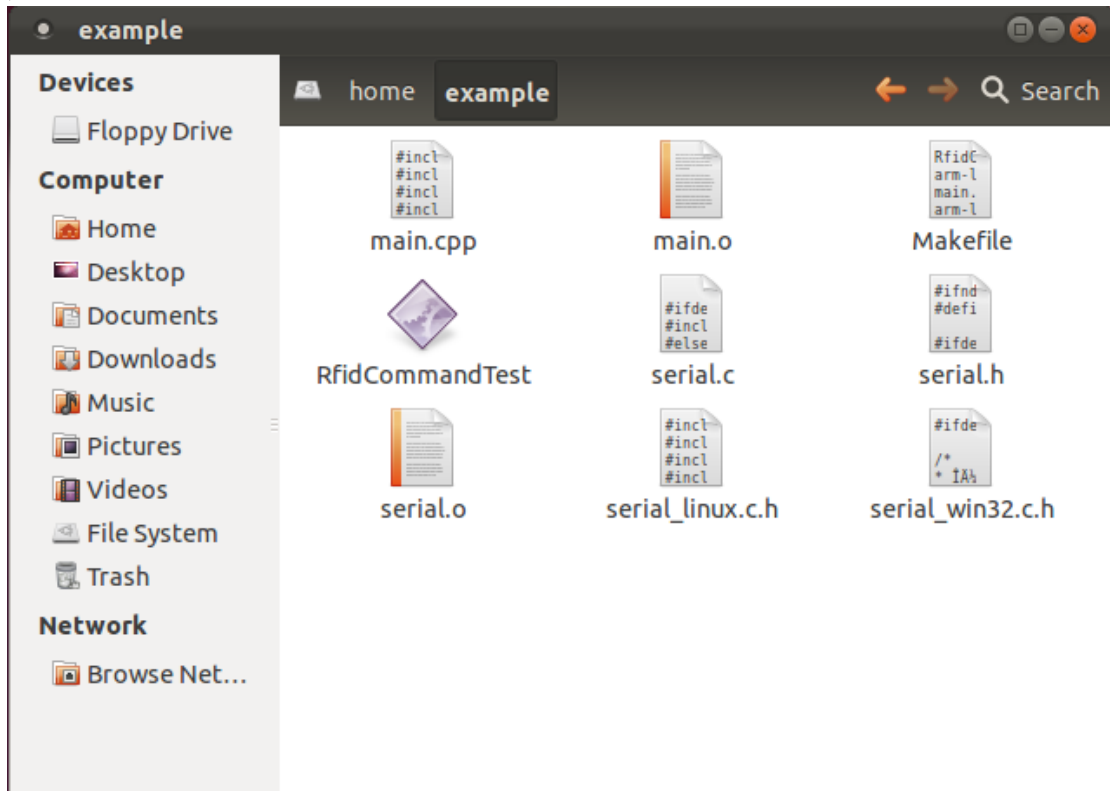
5、打开 terminal 终端，在文件所处目录下执行 make 命令：

```

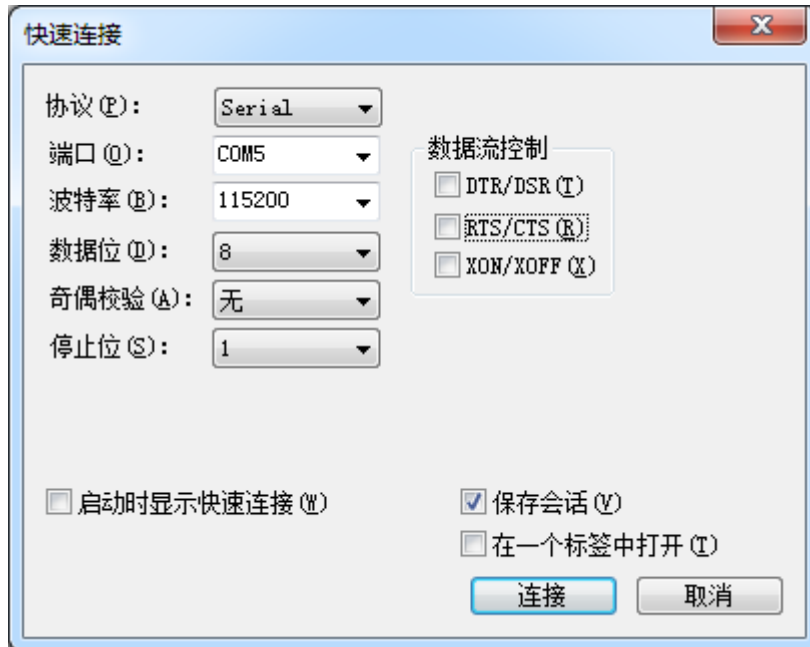
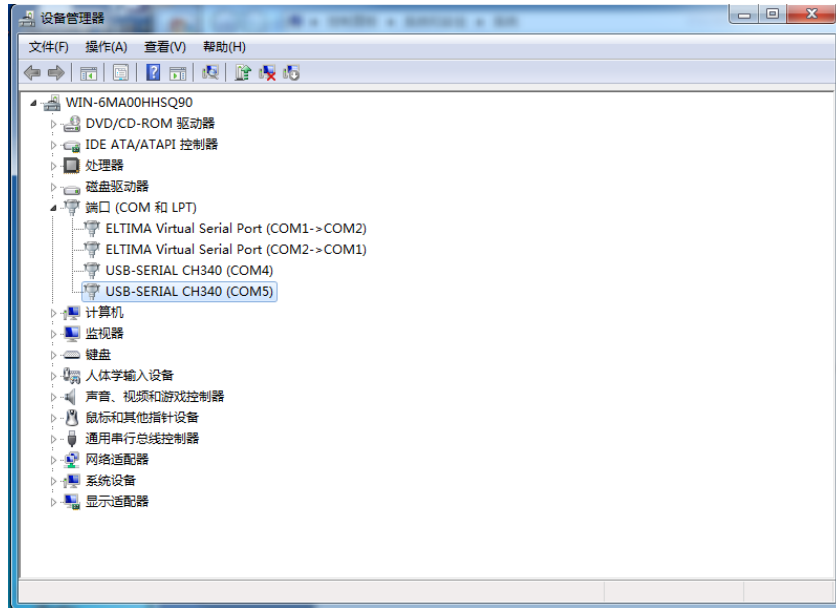
root@ubuntu: /home/example
root@ubuntu: /# cd /home/example
root@ubuntu: /home/example# ls
main.cpp  Makefile  serial.c  serial_linux.c.h  serial_win32.c.h
main.cpp~ Makefile~ serial.h  serial_linux.c.h~
root@ubuntu: /home/example# make
arm-linux-g++ -c main.cpp
arm-linux-g++ -c serial.c
arm-linux-g++ -o RfidCommandTest main.o serial.o
root@ubuntu: /home/example# ls
main.cpp  Makefile  serial.c  serial_linux.c.h~
main.cpp~ Makefile~ serial.h  serial.o
main.o  RfidCommandTest  serial_linux.c.h  serial_win32.c.h
root@ubuntu: /home/example#

```

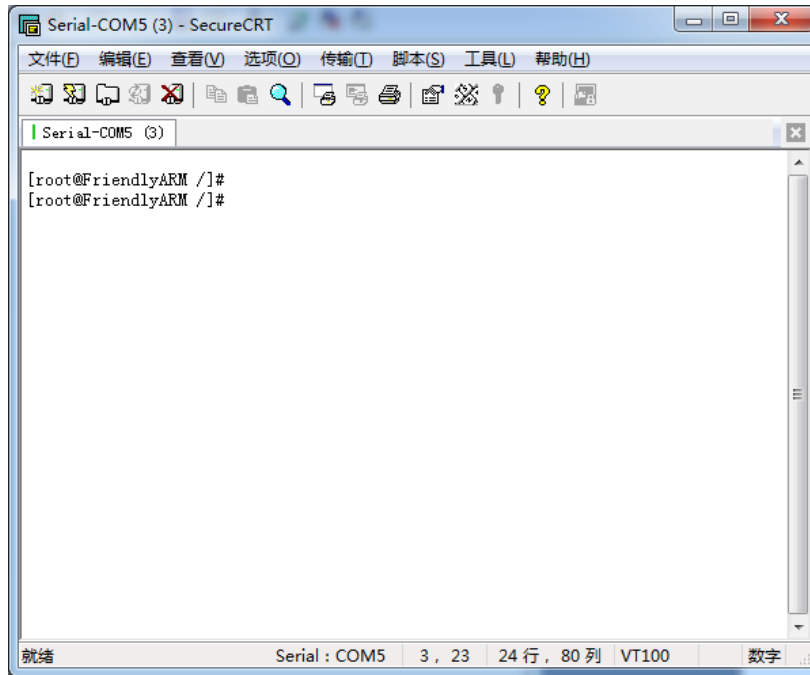
将编译生成的文件 RfidCommandTest 拷贝到 SD 卡上，然后复制到 ARM 开发板上，如果没有 SD 卡，也可以通过 FTP 或者 NFS 将文件拷贝到 ARM 开发板上。



6、确保 ARM 已经打开，并且已经通过串口连接到 PC 机，查看设备管理器，查看连接的串口，演示实验对应的串口为 COM5，打开 SecureCRT 5.1，选择 文件->快速连接，按照如图下图配置连接。



回车，得到如下图，则 PC 与 ARM 连接好了，如果没有连接好，请检查 ARM 是否上电，串口是否连接，PC 机上的串口是否与设备管理器上面的对应。



7、将 RFID 模块通过串口与 ARM 连接好。

转到根目录：cd /

新建文件夹：mkdir example

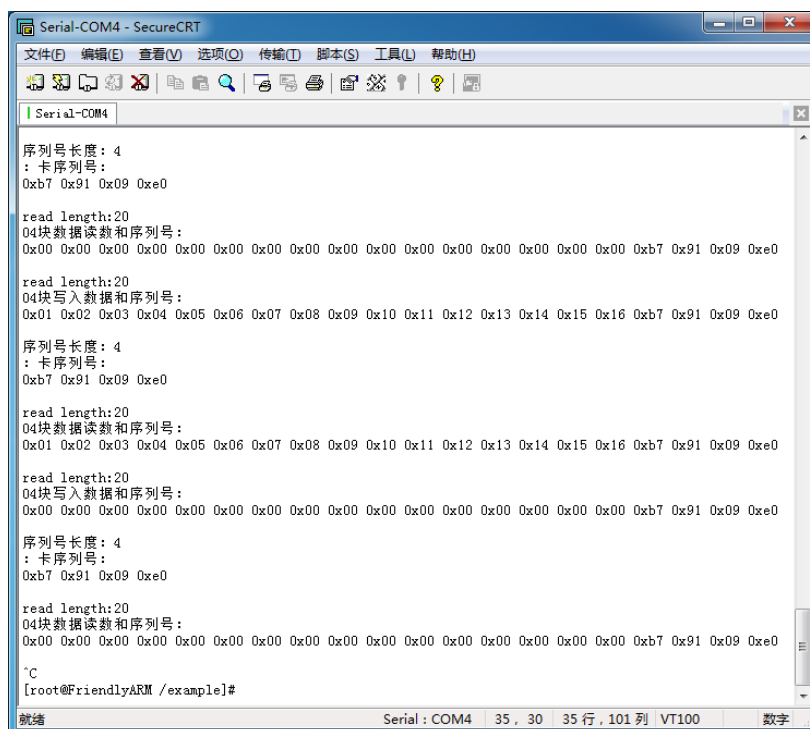
打开文件夹：cd example

将 SD 卡上面的可执行文件 serial 拷贝到 example 文件夹：cp /sdcard/linshi/

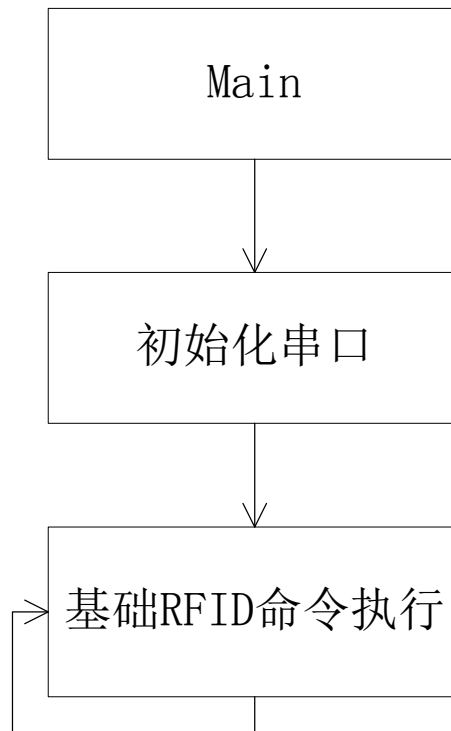
RfidCommandTest .

其中 linshi 是 sdcard 下的一个临时文件夹。

然后执行： ./RfidCommandTest



1.2.6 程序框图



1.2.7 代码解析

主函数 main.cpp:

```
#include <iostream>
#include <string.h>
#include <stdio.h>
#include "serial.h"

using namespace std;

#ifdef _WIN32    /*在windows环境下*/
    #include "Windows.h"    /*windows的头文件*/
    #define SERIAL_SLEEP(n)    Sleep(n*1000)    /*休眠n秒*/
    #define WSN_APP_PORT "\\\\.\\COM4"    /*windows下的串口4*/
#else    /*在linux环境下*/
    #include <unistd.h>    /*linux下的头文件*/
    #define SERIAL_SLEEP(n)    usleep(n*1000000)    /*休眠n秒*/
    #define WSN_APP_PORT "/dev/ttySAC3"    /*linux下的串口1*/
#endif
#endif
```

```
#define WSN_APP_BAUD 9600    /*设置串口波特率*/

int main()
{
    unsigned char readCardNumber[2] = {0x02, 0xA0};    /*读卡序列号命令字符
数组*/
    unsigned char readFourthSector[9] = {0x09, 0xA1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0x04};    /*读卡命令字符数组*/
    unsigned char writeFourthSectorOne[25] = {0x19, 0xA2, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x04,
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12,
0x13, 0x14, 0x15, 0x16};    /*写卡命令字符数组1*/
    unsigned char writeFourthSectorTwo[25] = {0x19, 0xA2, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x04,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};    /*写卡命令字符数组2*/
    unsigned char receivedbuf[16] = {0};

    int _fd = serial_open(WSN_APP_PORT, WSN_APP_BAUD);    /*打开串口*/
    if (_fd < 0)
    {
        printf("Open serial failed\n");
        return -1;
    }

    bool selectWriteBuffer = true;
    int len;
    while (1)
    {
        /*读序列号*/
        serial_write(_fd, readCardNumber, 2);    /*发送读序列号命令*/
        len = serial_read(_fd, receivedbuf, 4);    /*接收序列号*/
        printf("序列号长度: %d\n: ", len);
        printf("卡序列号: \n");
        for(int i = 0; i < len; i++)
        {
            printf("0x%02x ", receivedbuf[i]);
        }
        printf("\n\n");
        SERIAL_SLEEP(1);

        /*读块数据*/
        serial_write(_fd, readFourthSector, 9);    /*发送读块命令*/
```



```
len = serial_read(_fd, receivedbuf, 20);    /*接收块数据*/
printf("read length:%d\n", len);
if(len == 20)
{
    printf("04块数据读数和序列号: \n");
    for(int i = 0; i < len; i++)
    {
        printf("0x%02x ", receivedbuf[i]);
    }
}
printf("\n\n");
SERIAL_SLEEP(1);

/*写块数据*/
if(selectWriteBuffer == false)
{
    serial_write(_fd, writeFourthSectorOne, 25);    /*发送写块命名*/
    len = serial_read(_fd, receivedbuf, 20);    /*写入数据返回*/
    printf("read length:%d\n", len);
    if(len == 20)
    {
        printf("04块写入数据和序列号: \n");
        for(int i = 0; i < len; i++)
        {
            printf("0x%02x ", receivedbuf[i]);
        }
    }
    printf("\n\n");
    selectWriteBuffer = true;
}
else
{
    serial_write(_fd, writeFourthSectorTwo, 25);    /*发送写块命名*/
    len = serial_read(_fd, receivedbuf, 20);    /*写入数据返回*/
    printf("read length:%d\n", len);
    if(len == 20)
    {
        printf("04块写入数据和序列号: \n");
        for(int i = 0; i < len; i++)
        {
            printf("0x%02x ", receivedbuf[i]);
        }
    }
    printf("\n\n");
}
```

```
selectWriteBuffer = false;
    }
    SERIAL_SLEEP(1);
}
return 0;
}
```

接口函数 serial.h 和 serial.c:

```
#ifndef __SERIAL_H__
#define __SERIAL_H__

#ifdef __cplusplus
extern "C" {
#endif

int serial_open(const char *dev, unsigned int speed);    /*打开串口*/
void serial_close(int fd);    /*关闭串口*/

int serial_read (int fd, void *buf, size_t count);    /*读串口*/
int serial_write(int fd, const void *buf, size_t count);    /*写串口*/

#ifdef __cplusplus
};
#endif

#endif /* __SERIAL_H__ */
```

```
#ifdef _WIN32
#include "serial_win32.c.h"
#else
#include "serial_linux.c.h"
#endif
```

Windows 和 Linux 下接口实现，请参照 serial_win32.c.h 和 serial_linux.c.h。

1.2.8 实验总结

本次实验主要是讲解在 window 下或这 linux 环境下编程来控制 RFID 模块数据，为以后电子钱包的实现奠定了基础，后续将在本实验下继续深入，做一个简单的 RFID 应用。

1.2.9 实验思考

如何将本实验所学的，做成一个简单的应用？

1.3 RFID 图形化开发之环境搭建

1.3.1 实验目的

学习 QT4.7.0 安装。

1.3.2 实验设备

- 1、ARM 开发板
- 2、ARM 开发板配套交叉串口线一根
- 3、SD 卡一张
- 4、读卡器一个

1.3.3 准备知识

网上查阅 QT 相关环境搭建教程。准备安装文件：X16-60997VS2010UltimTrial CHS.iso(vs2010 安装软件)，VS10sp1-KB983509.exe (vs2010 补丁)，qt-vs-addin-1.1.7.exe (vs2010 qt 插件)，qt-win-opensource-4.7.0-vs2008.exe (windows 环境下的 QT 安装软件)，qt-everywhere-opensource-src-4.7.0.tar.gz(linux 环境下的 QT 包，本光盘“Linux 平台/开发工具”文件夹下已附有该文件)，arm-linux-gcc-4.5.1-v6-vfp-20101103.tgz (linux 下的交叉编译包)。

1.3.4 环境搭建

Linux 环境，本机在虚拟机下安装的是 Ubuntu 11.10 版本，键入命令 `sudo passwd root` 可以启动 root 登录，注销后，使用 root 根用户登录。安装适合 ARM 的交叉编译环境（在上个实验已经安装好了交叉编译环境，请确保 arm-linux-gcc 可用）：

- 1、`apt-get install build-essential;`
- 2、`apt-get install libx11-dev libfreetype6-dev libavahi-gobject-dev libSM-dev libXrender-dev libfontconfig-dev libXext-dev;`
- 3、将 Qt4.7.0 源代码的原始包 qt-everywhere-opensource-src-4.7.0.tar.gz 拷贝到/opt 下，然后 `cd /opt/`，`tar xvzf /opt/qt-everywhere-opensource-src-4.7.0.tar.gz;`
- 4、`cd /opt/qt-everywhere-opensource-src-4.7.0`，执行 `./configure -prefix /opt`

```
/Qt4.7 -opensource -embedded arm -xplatform qws/linux-arm-g++ -no-webkit -qt-libtiff -qt-libmng -qt-mouse-tslib -qt-mouse-pc -no-mouse-linuxfp -no-neon;
```

5、make && make install。第 5 步持续时间比较长，编译完成后，Qt4.7 被安装在/opt/Qt4.7 目录下。

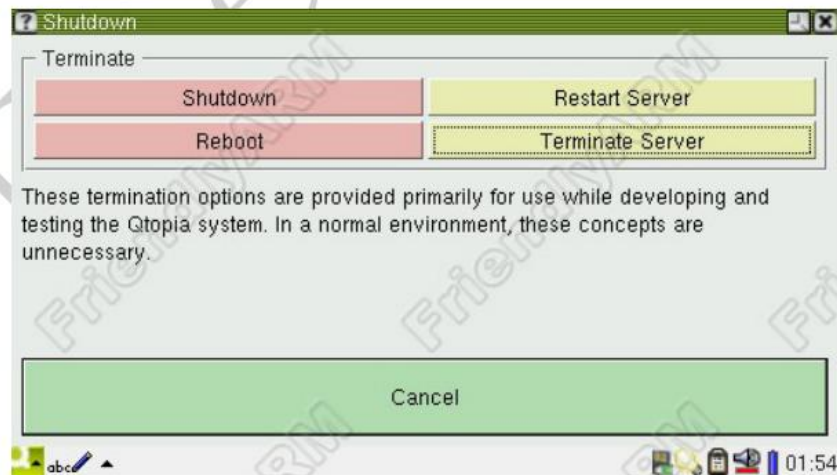
在 PC 上执行如下命令将 Qt4.7 打包：

```
#cd /opt
#tar cvzf qt4.7.tgz Qt4.7
```

打包完成后，将 qt4.7.tgz 拷贝到 SD 卡，然后将 SD 卡插入 ARM 开发板，执行一下命令将 qt4.7.tgz 解压到开发板上的/opt 目录下：

```
@#rm /usr/local/Trolltech/QtEmbedded-4.7.0-arm/ -rf
@#cd /opt
@#tar xvf /sdcard/qt4.7.tgz
```

运行 Qt4.7 程序前，需要退出 Qtopia2.2.0（默认是没有启动的，不需要退出，其他图形化界面程序可以直接通过点击右上角的 X 按钮）。



在 ARM 开发板上执行

```
@#cp /bin/qt4 /bin/setqt4env
```

然后按照如下代码简单修改 setqt4env

```
@#vi /bin/setqt4env
```

Setqt4env 代码

```
#!/bin/sh

if [ -e /etc/friendlyarm-ts-input.conf ] ; then
    . /etc/friendlyarm-ts-input.conf
fi
true${TSLIB_TSDEVICE:=/dev/touchscreen}

TSLIB_CONFFILE=/etc/ts.conf
```

```
export TSLIB_TSDEVICE
export TSLIB_CONFFILE

export TSLIB_PLUGINDIR=/usr/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal

export QWS_DISPLAY=:1
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin

if [ -c ${TSLIB_TSDEVICE} ]; then
    export QWS_MOUSE_PROTO="Tslib MouseMan:/dev/input/mice"
    if [ ! -s /etc/pointercal ] ; then
        rm /etc/pointercal
        /usr/bin/ts_calibrate
    fi
else
    export QWS_MOUSE_PROTO="MouseMan:/dev/input/mice"
fi

export QWS_KEYBOARD=TTY:/dev/tty1

export HOME=/root

cd /opt/Qt4.7/demos/embedded/fluidlauncher/
./fluidlauncher -qws
```

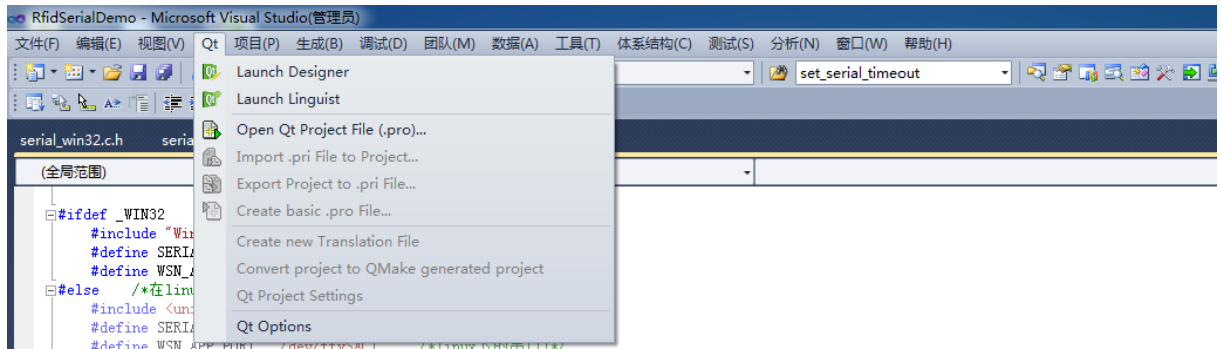
运行示例程序：

```
. /bin/setqt4env
```

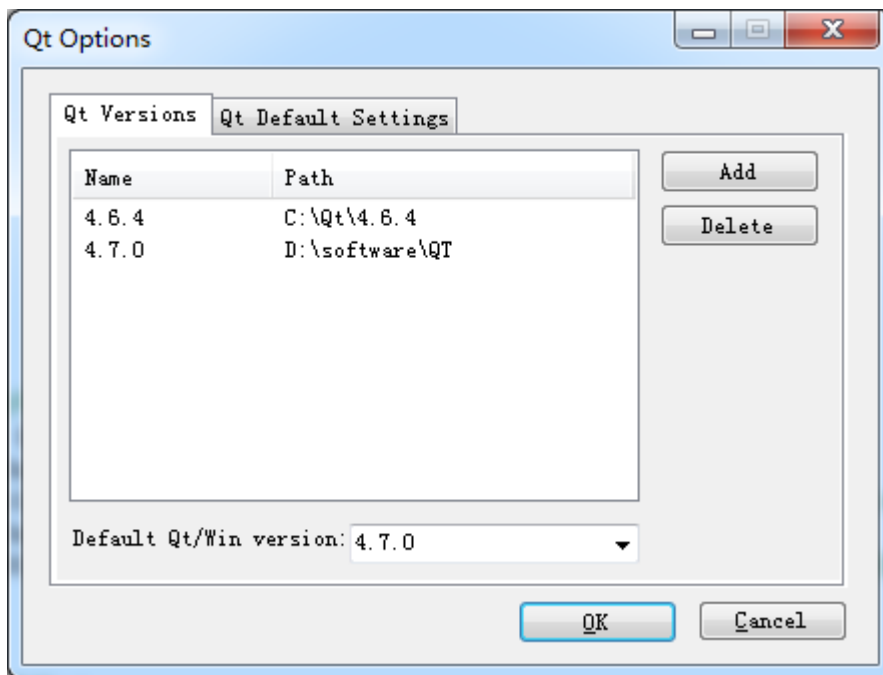
示例程序的运行结果如下：



Windows 环境，首先安装 vs2010 和 vs2010sp1，然后安装 qt-win-opensource-4.7.0-vs2008.exe，再安装 QT 插件 qt-vs-addin-1.1.7.exe。安装完成后，可以打开 VS2010 可以看到有一个 Qt 选项。打开如下图所示的 Qt-Options。



然后选择 Qt4.7.0 为默认选项即可。



1.3.5 实验总结

本实验已将 Windows 下面的开发环境和 ARM 下的 QT 交叉编译环境都已经搭建好，为下一节实验 QT 图形化小软件开发打下了基础。

1.4 简单电子钱包实验

1.4.1 实验目的

简单 RFID 应用。

1.4.2 实验设备

- 1、单片机模块一个
- 2、标准串口线一根
- 3、RFID 电源供电 USB 线一根
- 4、S50 卡一张
- 5、ARM 开发板
- 6、ARM 开发板配套交叉串口线一根
- 7、SD 卡一张
- 8、读卡器一个

1.4.3 准备知识

认真学习实验一、二，本实验是建立在前两个实验之上的，确保开发环境已经搭建好。

1.4.4 实验原理

本实验主要是开发一个简易的图形化 RFID 的小应用，主要是学习如何开发一些简单的图形化小软件。在 vs2010 开发环境下面编辑代码并调试，将调试好的代码放在 ubuntu 下交叉编译生成可以在 ARM 下面运行的二进制文件。

1.4.5 实验步骤

因为 QT 编译不支持中文路径，请将“无线模块实验\RFID 基础实验\简单电子钱包应用”目录下面的 BaseRfidDemo 复制到 D 盘根目录下面，然后打开该目录下面的 BaseRfidDemo.sln，本实验默认的串口是串口 4，如果不是，请自行在 wsnrfid.cpp 文件下修改。

```
using namespace std;

#ifdef _WIN32    /*在windows环境下*/
#include "Windows.h"    /*windows的头文件*/
#define SERIAL_SLEEP(n)    usleep(n*1000000)    /*休眠n秒*/
#define WSN_APP_PORT "\\\\.\\COM4"    /*windows下的串口4*/
#else    /*在linux环境下*/
#include <unistd.h>    /*linux下的头文件*/
#define SERIAL_SLEEP(n)    usleep(n*1000000)    /*休眠n秒*/
#define WSN_APP_PORT "/dev/ttySAC1"    /*linux下的串口1*/
#endif

#define WSN_APP_BAUD    9600    /*设置串口波特率*/
```

选择 调试->开始执行(不调试)



Windows 平台下，运行结果如下，初始化值为 118.9 元，充值 1.3 元，余额值为 120.2 元。



将软件移植到 ARM 下面：

将 BaseRfidDemo 文件夹拷贝到 ubuntu /home 目录下面。

在 ubuntu 系统下面打开 /home/BaseRfidDemo/BaseRfidDemo

执行如下命令：


```

root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo
root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo# ls
BaseRfidDemo.pro          homewidget.ui            qmake.exe
BaseRfidDemo.vcproj       main.cpp                 Qt4VSPROPERTYSheet.props
BaseRfidDemo.vcxproj      mainform.cpp            release
BaseRfidDemo.vcxproj.filters  mainform.h              Resources
BaseRfidDemo.vcxproj.user  mainform.qrc            rfid.png
Debug                     mainform.ui             run.png
GeneratedFiles            myinputpanelcontext.cpp serial
global.cpp               myinputpanelcontext.h  stop.png
global.h                 myinputpanel.cpp       wsnrfid.cpp
homewidget.cpp           myinputpanelform.ui    wsnrfid.h
homewidget.h             myinputpanel.h         wsnrfid.ui
root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo# qmake -project
root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo# qmake
root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo# make
/opt/Qt4.7/bin/uic homewidget.ui -o ui_homewidget.h
/opt/Qt4.7/bin/uic mainform.ui -o ui_mainform.h
/opt/Qt4.7/bin/uic myinputpanelform.ui -o ui_myinputpanelform.h
/opt/Qt4.7/bin/uic wsnrfid.ui -o ui_wsnrfid.h
arm-linux-g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_NETWORK_LIB -DQT_CORE_LIB -DQT_SHARED -I/opt/Qt4.7/mkspecs/qws/linux-arm-g++ -I. -I/opt/Qt4.7/include/QtCore -I/opt/Qt4.7/include/QtNetwork -I/opt/Qt4.7/include/QtGui -I/opt/Qt4.7/include -I. -Iserial -I. -I. -o global.o global.cpp

```

将生成的 BaseRfidDemo 文件通过 SD 卡拷贝到 ARM 板上面的 /example 目录下：

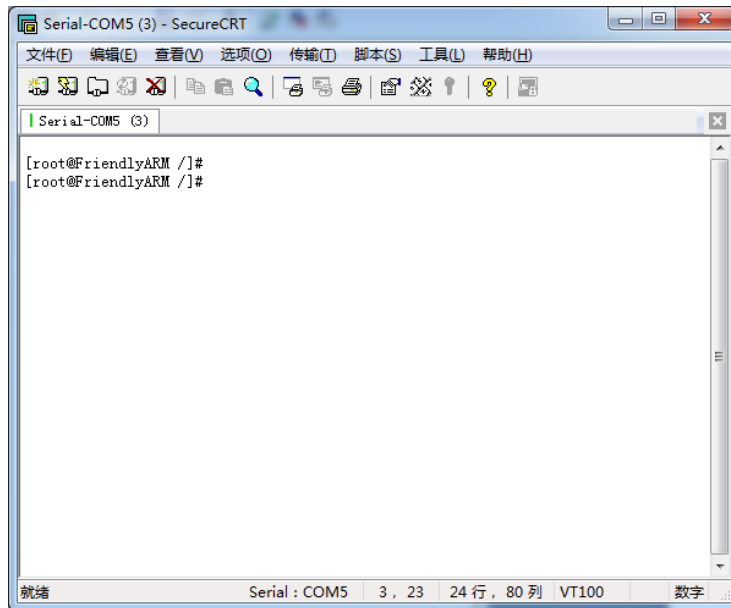
```

root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo
root@ubuntu: /home/BaseRfidDemo/BaseRfidDemo# ls
BaseRfidDemo          moc_myinputpanel.o
BaseRfidDemo.pro      moc_wsnrfid.cpp
BaseRfidDemo.vcproj   moc_wsnrfid.o
BaseRfidDemo.vcxproj  myinputpanelcontext.cpp
BaseRfidDemo.vcxproj.filters  myinputpanelcontext.h
BaseRfidDemo.vcxproj.user  myinputpanelcontext.o
Debug                 myinputpanel.cpp
GeneratedFiles        myinputpanelform.ui
global.cpp            myinputpanel.h
global.h              myinputpanel.o
global.o              qmake.exe
homewidget.cpp        qrc_mainform.cpp
homewidget.h          qrc_mainform.o
homewidget.o          Qt4VSPROPERTYSheet.props
homewidget.ui         release
main.cpp              Resources
mainform.cpp          rfid.png
mainform.h            run.png
mainform.o            serial.o
mainform.qrc          stop.png
mainform.ui           ui_homewidget.h
main.o                ui_mainform.h
Makefile

```

注：文件目录“无线模块实验\RFID 基础实验\简单电子钱包应用\BaseRfidDemo\ARM 环境可执行文件”下已保存有 BaseRfidDemo 和 BaseRfidDemoStart 两个编译好的可执行文件。

同实验二，打开 SecureCRT 5.1：



将 RFID 模块通过串口与 ARM 连接好

转到根目录: `cd /`

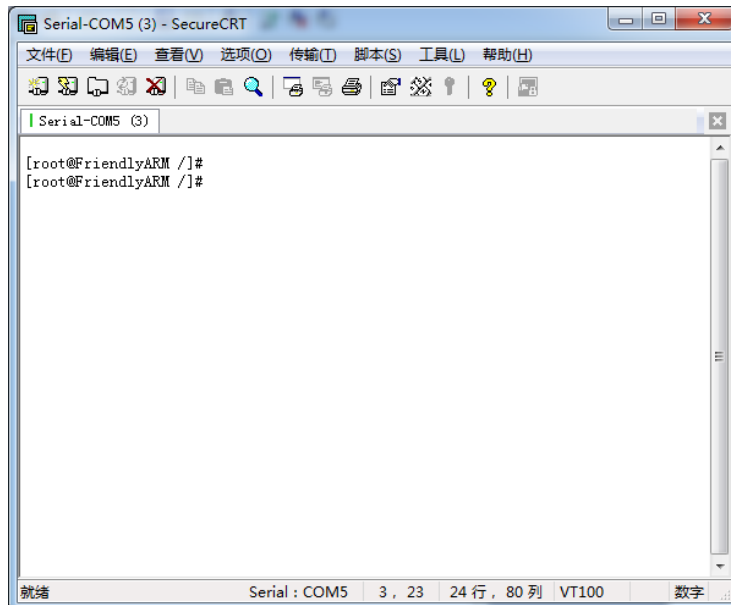
新建文件夹: `mkdir example`

打开文件夹: `cd example`

将 SD 卡上面的可执行文件 BaseRfidDemo 拷贝到 example 文件夹: `cp /sdcard/linshi/BaseRfidDemo`

其中 linshi 是 sdcard 下的一个临时文件夹。

然后执行: `./BaseRfidDemo`



将 RFID 模块通过串口与 ARM 连接好

转到根目录: `cd /`

新建文件夹: `mkdir example`

打开文件夹: `cd example`

将 SD 卡上面的可执行文件 BaseRfidDemoStart 拷贝到 example 文件夹：cp /sdcard/linshi/BaseRfidDemoStart .

其中 linshi 是 sdcard 下的一个临时文件夹。

然后执行：./ BaseRfidDemoStart

编辑 sh 脚本文件 BaseRfidDemoStart

```
#!/bin/sh

if [ -e /etc/friendlyarm-ts-input.conf ] ; then
    . /etc/friendlyarm-ts-input.conf
fi

true${TSLIB_TSDEVICE:=/dev/touchscreen}

TSLIB_CONFFILE=/etc/ts.conf

export TSLIB_TSDEVICE
export TSLIB_CONFFILE

export TSLIB_PLUGINDIR=/usr/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal

export QWS_DISPLAY=:1
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin

if [ -c ${TSLIB_TSDEVICE} ]; then
    export QWS_MOUSE_PROTO="Tslib MouseMan:/dev/input/mice"
    if [ ! -s /etc/pointercal ] ; then
        rm /etc/pointercal
        /usr/bin/ts_calibrate
    fi
else
    export QWS_MOUSE_PROTO="MouseMan:/dev/input/mice"
fi

export QWS_KEYBOARD=TTY:/dev/tty1

export HOME=/root

./BaseRfidDemo -qws -font wenquanyi
```

执行如下命令：

```
[root@FriendlyARM /]# cd example/  
[root@FriendlyARM /example]# ls  
BaseRfidDemo      BaseRfidDemoStart  RfidCommandTest  
[root@FriendlyARM /example]# ./BaseRfidDemo  
./BaseRfidDemo      ./BaseRfidDemoStart  
[root@FriendlyARM /example]# ./BaseRfidDemoStart  
onChangeApp:mainStackedWidget current index = 0
```

就绪

Serial : COM4

35 , 1

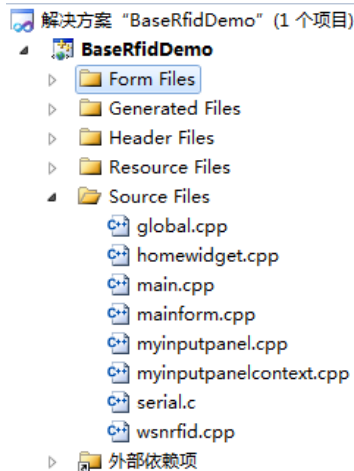
35 行, 101 列

VT100

大写 数字

1.4.6 实验代码解析

工程文件目录



其中涉及到 RFID 操作的主要文件是：wsnrfid.cpp 和 wsnrfid.h。对于串口操作的文件同实验上位机读写 RFID 模块实验。

```
#include "wsnrfid.h"  
#include <QWidget>  
#include <QSlider>  
#include <QVBoxLayout>  
#include <QToolBar>  
#include <QPushButton>  
#include <QLabel>  
#include <QAction>  
#include <QGroupBox>  
#include <QButtonGroup>  
#include <QLineEdit>  
#include <QList>  
#include <QPixmap>  
#include <QPalette>  
#include <QTextEdit>  
#include <QMessageBox>  
  
#ifdef WIN32      /*在windows环境下*/  
    #include "Windows.h"    /*windows的头文件*/
```

```

#define SERIAL_SLEEP(n)    Sleep(n)    /*休眠n耗秒*/
#define WSN_APP_PORT "\\\\.\\COM4"    /*windows下的串口4*/
#else    /*在linux环境下*/
#include <unistd.h>    /*linux下的头文件*/
#define SERIAL_SLEEP(n)    usleep(n*1000)    /*休眠n耗秒*/
#define WSN_APP_PORT    "/dev/ttySAC3"    /*linux下的串口1*/
#endif

#define RFID_BAUD 9600    /*设置串口波特率*/

WSNRfid::WSNRfid(const QString & typeName, const QString & selfName, const
QString & iconPath, QWidget *parent)
    : QWidget(parent), WSNPlugin(typeName,selfName,iconPath){//,
QThread(parent)
{
    //ui.setupUi(this);
    QVBoxLayout * mainlayout = new QVBoxLayout;
    QToolBar * mainToolBar = new QToolBar;
    QVBoxLayout * showWidgetLayout = new QVBoxLayout;

    m_showWidget = new QWidget;
    m_open_serialport = new QAction(QIcon(":/ico/run.png"), tr("打开通讯端口
"),NULL);
    m_close_serialport = new QAction(QIcon(":/ico/stop.png"),tr("关闭通讯端口
"),NULL);

    m_serialNumber = new QLabel("卡序号: ");
    m_password = new QLabel("密码 : ");
    m_initMoney = new QLabel("初始化: ");
    m_readMoney = new QLabel("余额值: ");
    m_addMoney = new QLabel("充 钱: ");
    m_deductMoney = new QLabel("扣 除: ");

    m_serialNumberValue = new QLineEdit("null");
    m_serialNumberValue->setReadOnly(true);
    m_passwordValue = new QLabel("FFFFFFFFFFFF");
    //m_passwordValue->setReadOnly(true);
    m_initMoneyValue = new QLineEdit("1000");
    m_readMoneyValue = new QLineEdit("null");
    m_readMoneyValue->setReadOnly(true);
    m_addMoneyValue = new QLineEdit("1");
    m_deductMoneyValue = new QLineEdit("1");

    m_readCardBtn = new QPushButton("读卡");

```

```
// m_changePasswordBtn = new QPushButton("改密码");
m_initMoneyBtn = new QPushButton("初始化");
m_addMoneyBtn = new QPushButton("充钱");
m_deductMoneyBtn = new QPushButton("扣钱");

QGridLayout * baseInformationLayout = new QGridLayout();
QHBoxLayout * operateBtnLayout = new QHBoxLayout();

baseInformationLayout->addWidget(m_serialNumber, 0, 0);
baseInformationLayout->addWidget(m_serialNumberValue, 0, 1);
baseInformationLayout->addWidget(m_initMoney, 0, 2);
baseInformationLayout->addWidget(m_initMoneyValue, 0, 3);
baseInformationLayout->addWidget(m_readMoney, 1, 0);
baseInformationLayout->addWidget(m_readMoneyValue, 1, 1);
baseInformationLayout->addWidget(m_addMoney, 1, 2);
baseInformationLayout->addWidget(m_addMoneyValue, 1, 3);
baseInformationLayout->addWidget(m_password, 2, 0);
baseInformationLayout->addWidget(m_passwordValue, 2, 1);
baseInformationLayout->addWidget(m_deductMoney, 2, 2);
baseInformationLayout->addWidget(m_deductMoneyValue, 2, 3);

operateBtnLayout->addWidget(m_readCardBtn);
operateBtnLayout->addWidget(m_initMoneyBtn);
operateBtnLayout->addWidget(m_addMoneyBtn);
operateBtnLayout->addWidget(m_deductMoneyBtn);

showWidgetLayout->addLayout(baseInformationLayout);
showWidgetLayout->addLayout(operateBtnLayout);

showWidgetLayout->setAlignment(Qt::AlignHCenter);

m_showWidget->setLayout(showWidgetLayout);

mainlayout->addWidget(mainToolbar);
mainlayout->addWidget(m_showWidget);

setLayout(mainlayout);

mainToolbar->addAction(m_open_serialport);
mainToolbar->addAction(m_close_serialport);

connect(m_open_serialport, SIGNAL(triggered()), this,
SLOT(onOpenSerialPort()));
connect(m_close_serialport, SIGNAL(triggered()), this,
```

```
SLOT(onCloseSerialPort()));
connect(m_readCardBtn, SIGNAL(clicked()), this, SLOT(onReadCardBtn()));
connect(m_initMoneyBtn, SIGNAL(clicked()), this, SLOT(onInitMoneyBtn()));
connect(m_addMoneyBtn, SIGNAL(clicked()), this, SLOT(onAddMoneyBtn()));
connect(m_deductMoneyBtn, SIGNAL(clicked()), this,
SLOT(onDeductMoneyBtn()));

m_close_serialport->setDisabled(true);

m_showWidget->setDisabled(true);
}

WSNRfid::~WSNRfid()
{
}

void WSNRfid :: onOpenSerialPort()    /*关闭串口*/
{
    _fd = serial_open(WSN_APP_PORT, RFID_BAUD);
    if(_fd >= 0)
    {
        m_open_serialport->setDisabled(true);
        m_close_serialport->setEnabled(true);
        m_showWidget->setEnabled(true);
    }
    else
    {
        QMessageBox::information(this, tr("err"), tr("串口打开失败"));
    }
}

void WSNRfid :: onReadCardBtn()    /*读取钱包值*/
{
    FloatToArray convertData;
    unsigned char readFourthSector[9] = {0x09, 0xA1, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0x04};
    serial_write(_fd, readFourthSector, 9);
    int len = serial_read(_fd, receivedbuf, 20);
    if(len ==20)
    {
        convertData.dataArray[0] = receivedbuf[0];
    }
}
```

```

        convertData.dataArray[1] = receivedbuf[1];
        convertData.dataArray[2] = receivedbuf[2];
        convertData.dataArray[3] = receivedbuf[3];
        QString pos = QString::number(convertData.dataFloat);
        m_readMoneyValue->setText(pos);
        pos = QString::number(receivedbuf[16], 16);
        pos += QString::number(receivedbuf[17], 16);
        pos += QString::number(receivedbuf[18], 16);
        pos += QString::number(receivedbuf[19], 16);
        m_serialNumberValue->setText(pos);
    }
    else
    {
        m_serialNumberValue->setText("null");
        m_readMoneyValue->setText("null");
    }
}

void WSNRfid :: onInitMoneyBtn()    /*初始化钱包值*/
{
    bool OKFlag;
    FloatToArray convertData;
    convertData.dataFloat = m_initMoneyValue->text().toFloat(&OKFlag);
    if(OKFlag == false)
    {
        QMessageBox::information(this, tr("err"), tr("初始值输入有误!"));
    }
    else
    {
        {
            if(convertData.dataFloat < 0)
            {
                QMessageBox::information(this, tr("err"), tr("不允许输入负数!"));
                return;
            }
            unsigned char writeFourthSector[25] = {0x19, 0xA2, 0xFF, 0xFF, 0xFF, 0xFF,
            0xFF, 0xFF, 0x04,
                0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12,
            0x13, 0x14, 0x15, 0x16};
            writeFourthSector[9] = convertData.dataArray[0];
            writeFourthSector[10] = convertData.dataArray[1];
            writeFourthSector[11] = convertData.dataArray[2];
            writeFourthSector[12] = convertData.dataArray[3];

```



```

serial_write(_fd, writeFourthSector, 25);
int len = serial_read(_fd, receivedbuf, 20);
if(len == 20)
{
    convertData.dataArray[0] = receivedbuf[0];
    convertData.dataArray[1] = receivedbuf[1];
    convertData.dataArray[2] = receivedbuf[2];
    convertData.dataArray[3] = receivedbuf[3];
    QString pos = QString::number(convertData.dataFloat);
    m_readMoneyValue->setText(pos);
    pos = QString::number(receivedbuf[16], 16);
    pos += QString::number(receivedbuf[17], 16);
    pos += QString::number(receivedbuf[18], 16);
    pos += QString::number(receivedbuf[19], 16);
    m_serialNumberValue->setText(pos);
    SERIAL_SLEEP(400);
}
else
{
    m_serialNumberValue->setText("null");
    m_readMoneyValue->setText("null");
}
}

void WSNRfid :: onAddMoneyBtn()    /*充值*/
{
    bool OKFlag;
    FloatToArray convertData;
    convertData.dataFloat = m_addMoneyValue->text().toFloat(&OKFlag);
    if(OKFlag == false)
    {
        QMessageBox::information(this, tr("err"), tr("初始值输入有误!"));
    }
    else
    {
        if(convertData.dataFloat < 0)
        {
            QMessageBox::information(this, tr("err"), tr("不允许输入负数!"));
            return;
        }
        unsigned char writeFourthSector[25] = {0x19, 0xA5, 0xFF, 0xFF, 0xFF, 0xFF,
        0xFF, 0xFF, 0x04,
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12,

```

```

0x13, 0x14, 0x15, 0x16};
    writeFourthSector[9] = convertData.dataArray[0];
    writeFourthSector[10] = convertData.dataArray[1];
    writeFourthSector[11] = convertData.dataArray[2];
    writeFourthSector[12] = convertData.dataArray[3];
    serial_write(_fd, writeFourthSector, 25);
    int len = serial_read(_fd, receivedbuf, 20);
    if(len == 20)
    {
        convertData.dataArray[0] = receivedbuf[0];
        convertData.dataArray[1] = receivedbuf[1];
        convertData.dataArray[2] = receivedbuf[2];
        convertData.dataArray[3] = receivedbuf[3];
        QString pos = QString::number(convertData.dataFloat);
        m_readMoneyValue->setText(pos);
        pos = QString::number(receivedbuf[16], 16);
        pos += QString::number(receivedbuf[17], 16);
        pos += QString::number(receivedbuf[18], 16);
        pos += QString::number(receivedbuf[19], 16);
        m_serialNumberValue->setText(pos);
        SERIAL_SLEEP(400);
    }
    else
    {
        m_serialNumberValue->setText("null");
        m_readMoneyValue->setText("null");
    }
}

void WSNRfid :: onDeductMoneyBtn()    /*消费*/
{
    bool OKFlag;
    FloatToArray convertData;
    convertData.dataFloat = m_deductMoneyValue->text().toFloat(&OKFlag);
    if(OKFlag == false)
    {
        QMessageBox::information(this, tr("err"), tr("初始值输入有误!"));
    }
    else
    {
        if(convertData.dataFloat < 0)
        {
            QMessageBox::information(this, tr("err"), tr("不允许输入负数!"));
        }
    }
}

```

```
        return;
    }
    unsigned char writeFourthSector[25] = {0x19, 0xA4, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0x04,
        0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x10, 0x11, 0x12,
0x13, 0x14, 0x15, 0x16};
    writeFourthSector[9] = convertData.dataArray[0];
    writeFourthSector[10] = convertData.dataArray[1];
    writeFourthSector[11] = convertData.dataArray[2];
    writeFourthSector[12] = convertData.dataArray[3];
    serial_write(_fd, writeFourthSector, 25);
    int len = serial_read(_fd, receivedbuf, 20);
    if(len == 20)
    {
        convertData.dataArray[0] = receivedbuf[0];
        convertData.dataArray[1] = receivedbuf[1];
        convertData.dataArray[2] = receivedbuf[2];
        convertData.dataArray[3] = receivedbuf[3];
        QString pos = QString::number(convertData.dataFloat);
        m_readMoneyValue->setText(pos);
        pos = QString::number(receivedbuf[16], 16);
        pos += QString::number(receivedbuf[17], 16);
        pos += QString::number(receivedbuf[18], 16);
        pos += QString::number(receivedbuf[19], 16);
        m_serialNumberValue->setText(pos);
        SERIAL_SLEEP(400);
    }
    else
    {
        m_serialNumberValue->setText("null");
        m_readMoneyValue->setText("null");
    }
}

void WSNRfid :: onCloseSerialPort()    /*关闭串口*/
{
    m_close_serialport->setDisabled(true);
    m_open_serialport->setEnabled(true);
    m_showWidget->setDisabled(true);
    serial_close(_fd);
}

QWidget * WSNRfid :: getUI() /*获取UI界面*/
```

```
{
    return this;
}

int WSNRfid :: load() /*载入widget*/
{
    return 1;
}

int WSNRfid :: unLoad()    /*卸载widget*/
{
    onCloseSerialPort();
    return 1;
}
```

1.4.7 实验总结

本实验主要学习了 QT 简单开发和嵌入式开发的一些基本技能，基本掌握如何去开发自己的小软件，本实验的小软件主要实现了充值消费和初始化这些基本功能，能够实现数据的存取功能。

1.4.8 实验思考

本实验主要是针对学生介绍一些开发的基本技巧，电子钱包软件的开发，还可以进一步完善，如注册机制，修改密码机制，加上数据库，可以做成一个成型的产品。有兴趣的同学可以自己完善本开发。

第二章 WiFi 基础实验

2.1 ARM 平台上 WiFi 无线模块的使用

2.1.1 实验目的

学习在 ARM 平台上使用 USB 无线网卡

2.1.2 实验设备

1、ARM 开发板

- 2、USB 无线网卡 SAGEM XG-760
- 3、交叉串口线一根

2.1.3 准备知识

嵌入式 Linux 平台下使用了 USB WiFi 和 SD WiFi 等无线网卡，本嵌入式平台上面有一套命令行的 USB WiFi kits 工具程序，该工具集可以支持上千种型号的 USB 无线网卡（大部分的 USB 无线网卡使用的内部芯片是相同的）。

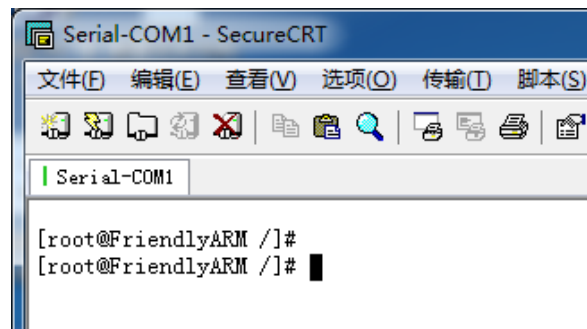
该工具集包含了无线网卡驱动程序，和下面将要使用的三个实用命令程序：

- 1、scan-wifi 用来扫描附近的无线网络
 - 2、start-wifi 用来开启连接无线网络
 - 3、stop-wifi 停止使用无线网络
- 这三个程序被安装在开发板的 usr/sbin 目录下

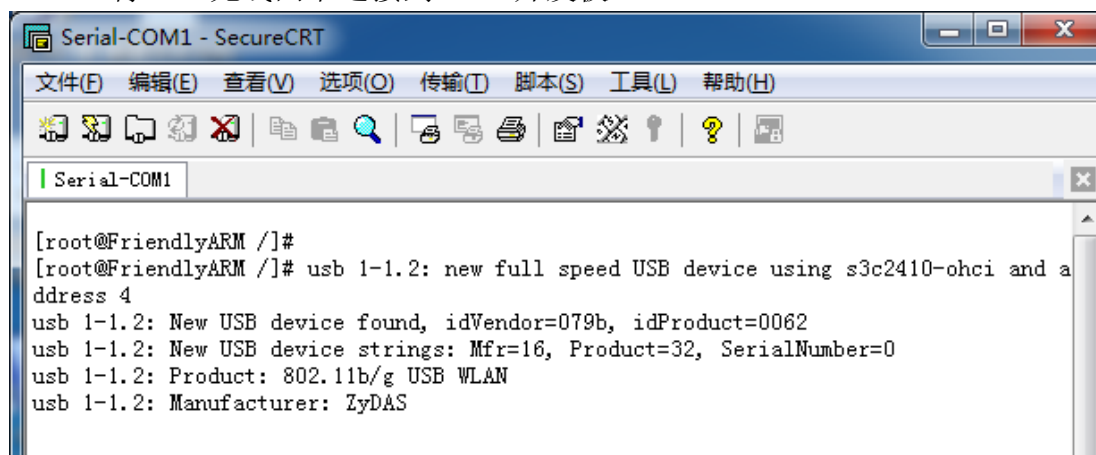
2.1.4 实验步骤

实验前请确保实验室内有 WiFi 接入点。

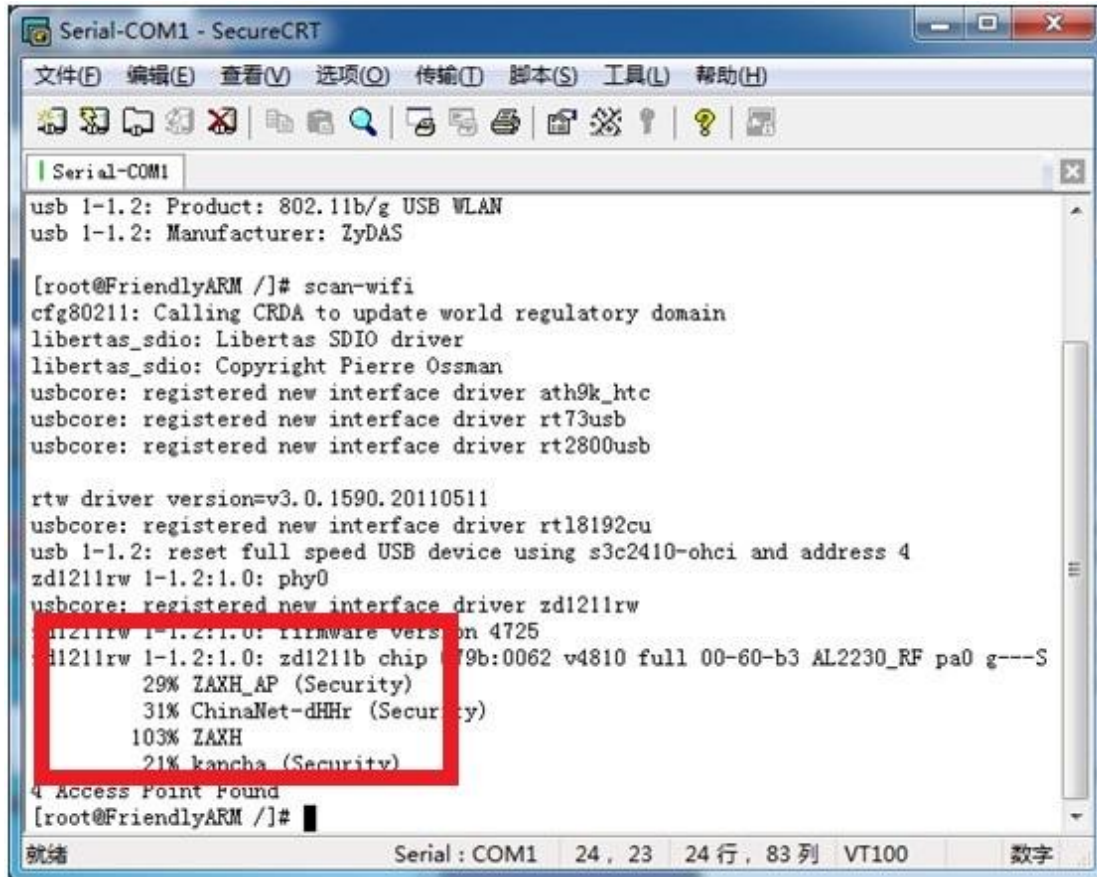
- 1、ARM 开启，连接好串口线，开启 SecureCRT 5.1 同上实验。



- 2、将 USB 无线网卡连接到 ARM 开发板上。



- 3、执行 scan-wifi 命令，如下图，扫描到 wifi 接入点，其中 ZAXH 无密码 wifi 接入点。

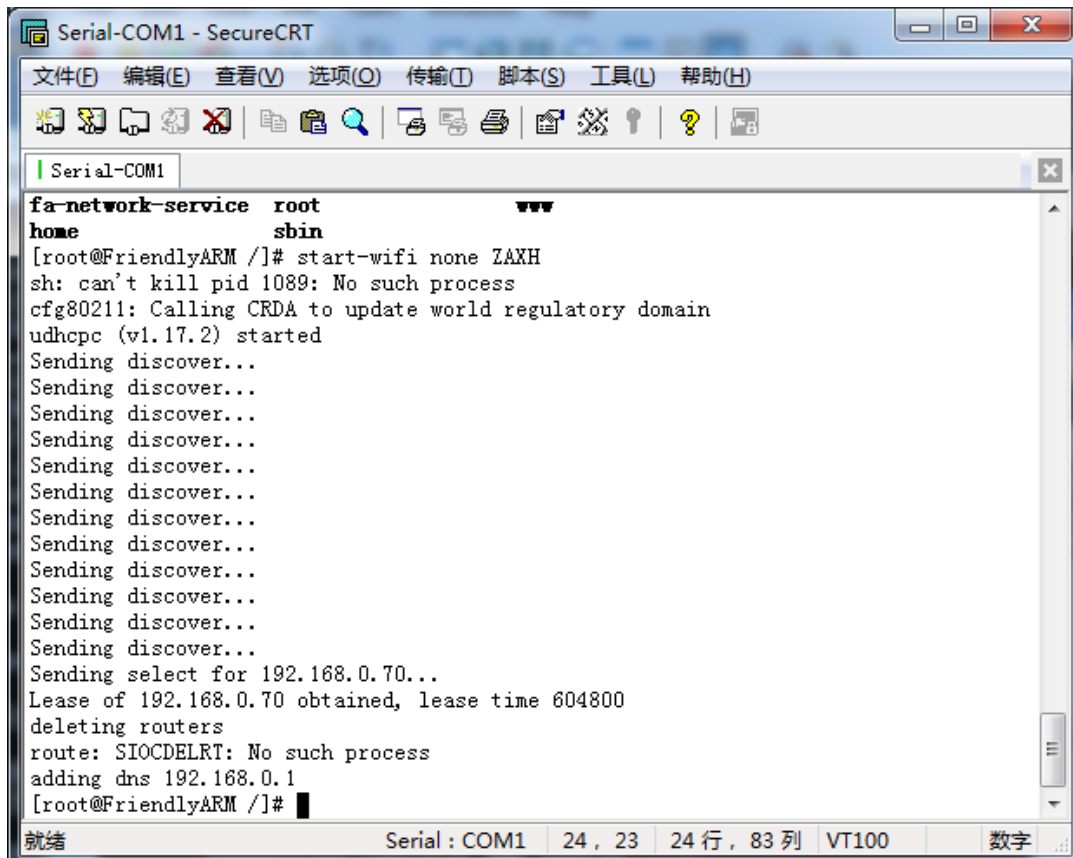


```
Serial-COM1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM1
usb 1-1.2: Product: 802.11b/g USB WLAN
usb 1-1.2: Manufacturer: ZyDAS

[root@FriendlyARM /]# scan-wifi
cfg80211: Calling CRDA to update world regulatory domain
libertas_sdio: Libertas SDIO driver
libertas_sdio: Copyright Pierre Ossman
usbcore: registered new interface driver ath9k_htc
usbcore: registered new interface driver rt73usb
usbcore: registered new interface driver rt2800usb

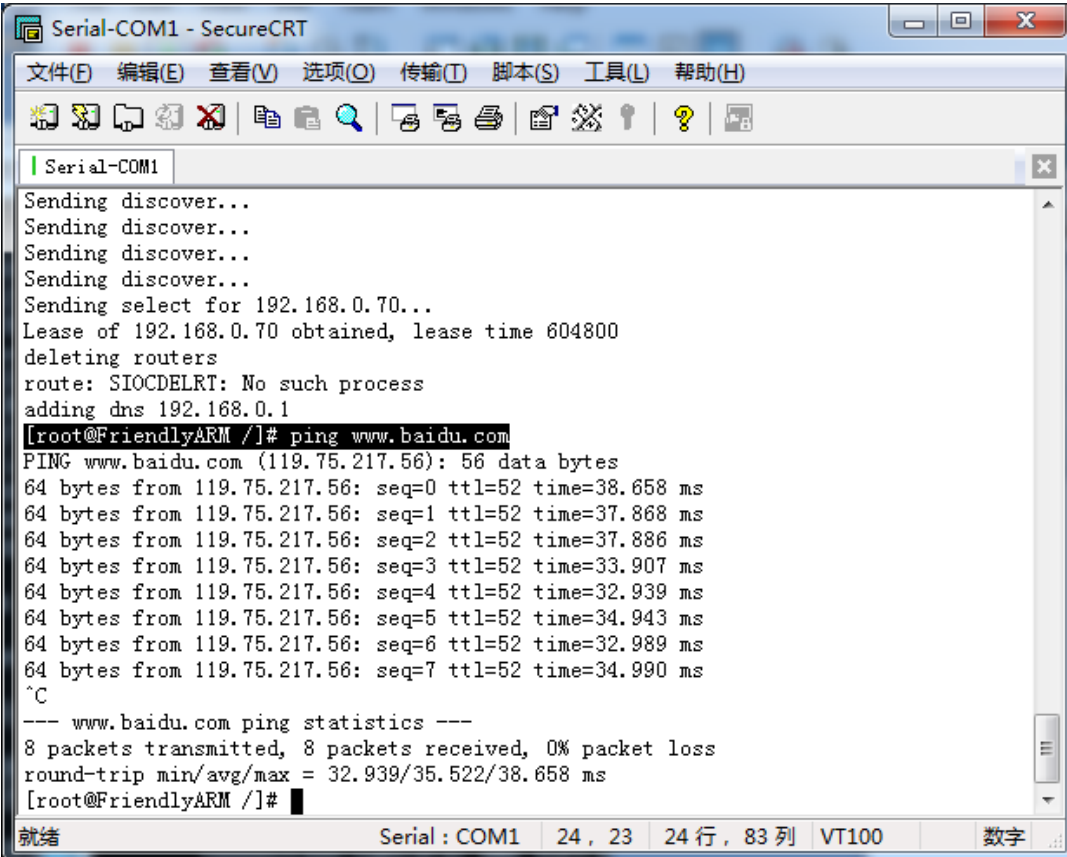
rtw driver version=v3.0.1590.20110511
usbcore: registered new interface driver rtl8192cu
usb 1-1.2: reset full speed USB device using s3c2410-ohci and address 4
zd1211rw 1-1.2:1.0: phy0
usbcore: registered new interface driver zd1211rw
zd1211rw 1-1.2:1.0: firmware version 4725
zd1211rw 1-1.2:1.0: zd1211b chip 079b:0062 v4810 full 00-60-b3 AL2230_RF pa0 g---S
    29% ZAXH_AP (Security)
    31% ChinaNet-dHHR (Security)
    103% ZAXH
    21% kancha (Security)
4 Access Point Found
[root@FriendlyARM /]#
```

4、 连接无密码 wifi 接入点，start-wifi none ZAXH。



```
Serial-COM1 - 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-COM1
fa-network-service root www
home sbin
[root@FriendlyARM /]# start-wifi none ZAXH
sh: can't kill pid 1089: No such process
cfg80211: Calling CRDA to update world regulatory domain
udhcpd (v1.17.2) started
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.0.70...
Lease of 192.168.0.70 obtained, lease time 604800
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.0.1
[root@FriendlyARM /]#
```

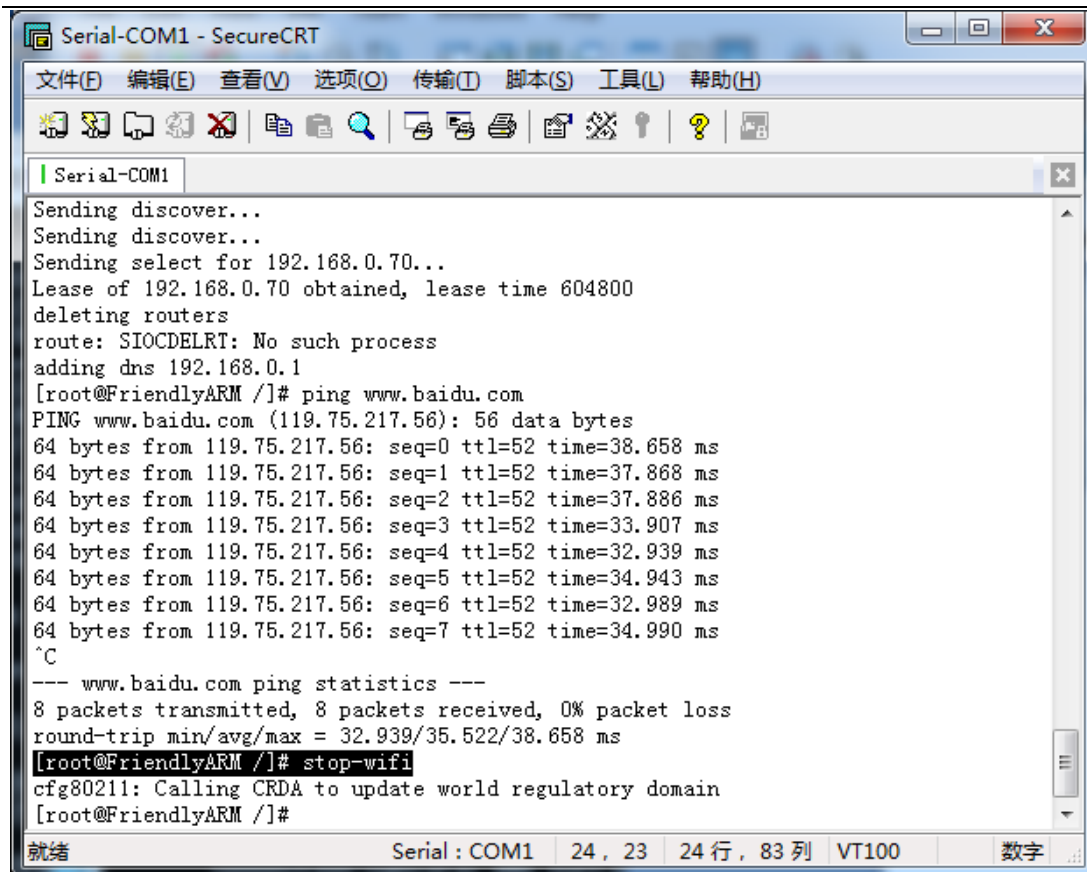
5、测试网络是否连接: ping www.baidu.com, 如下图, 数据发送成功。



```
Serial-COM1 - 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-COM1
Sending discover...
Sending discover...
Sending discover...
Sending discover...
Sending select for 192.168.0.70...
Lease of 192.168.0.70 obtained, lease time 604800
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.0.1
[root@FriendlyARM /]# ping www.baidu.com
PING www.baidu.com (119.75.217.56): 56 data bytes
64 bytes from 119.75.217.56: seq=0 ttl=52 time=38.658 ms
64 bytes from 119.75.217.56: seq=1 ttl=52 time=37.868 ms
64 bytes from 119.75.217.56: seq=2 ttl=52 time=37.886 ms
64 bytes from 119.75.217.56: seq=3 ttl=52 time=33.907 ms
64 bytes from 119.75.217.56: seq=4 ttl=52 time=32.939 ms
64 bytes from 119.75.217.56: seq=5 ttl=52 time=34.943 ms
64 bytes from 119.75.217.56: seq=6 ttl=52 time=32.989 ms
64 bytes from 119.75.217.56: seq=7 ttl=52 time=34.990 ms
^C
--- www.baidu.com ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 32.939/35.522/38.658 ms
[root@FriendlyARM /]#
```

就绪 Serial : COM1 24, 23 24 行, 83 列 VT100 数字

6、停止 wifi: stop-wifi。

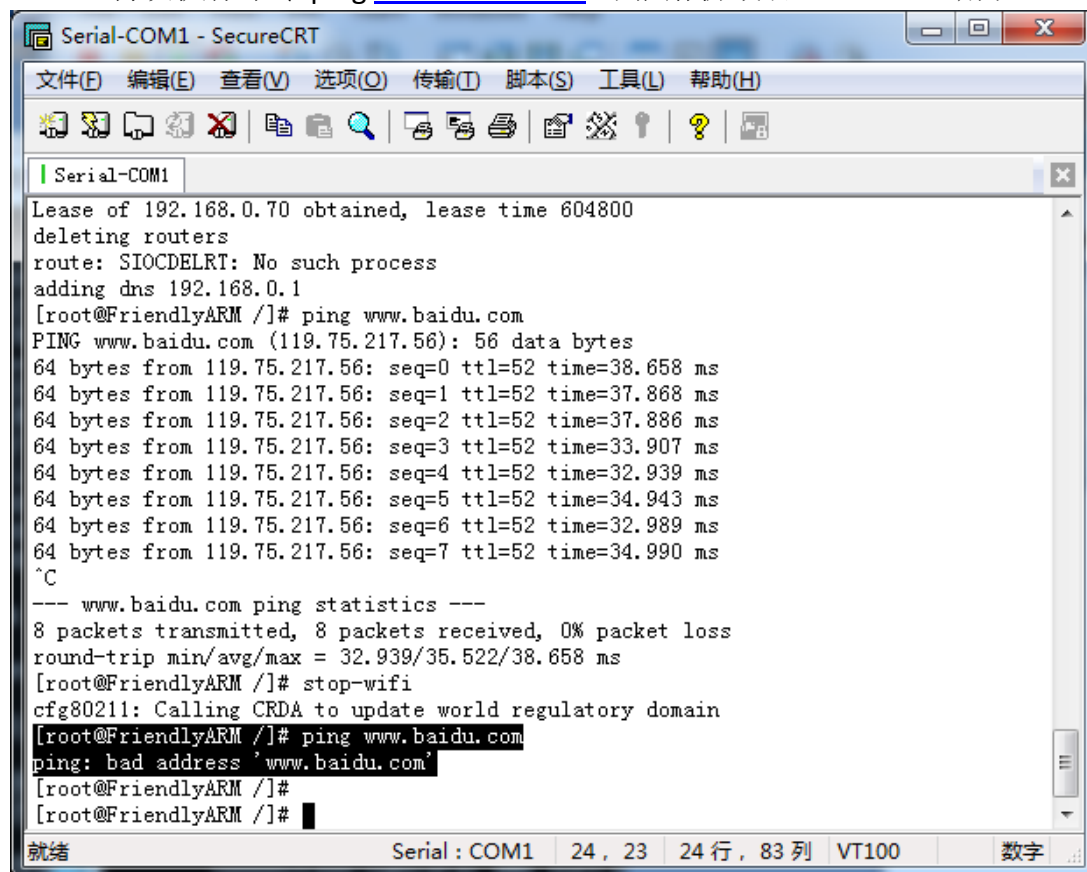


```

Serial-COM1 - 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-COM1
Sending discover...
Sending discover...
Sending select for 192.168.0.70...
Lease of 192.168.0.70 obtained, lease time 604800
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.0.1
[root@FriendlyARM /]# ping www.baidu.com
PING www.baidu.com (119.75.217.56): 56 data bytes
64 bytes from 119.75.217.56: seq=0 ttl=52 time=38.658 ms
64 bytes from 119.75.217.56: seq=1 ttl=52 time=37.868 ms
64 bytes from 119.75.217.56: seq=2 ttl=52 time=37.886 ms
64 bytes from 119.75.217.56: seq=3 ttl=52 time=33.907 ms
64 bytes from 119.75.217.56: seq=4 ttl=52 time=32.939 ms
64 bytes from 119.75.217.56: seq=5 ttl=52 time=34.943 ms
64 bytes from 119.75.217.56: seq=6 ttl=52 time=32.989 ms
64 bytes from 119.75.217.56: seq=7 ttl=52 time=34.990 ms
^C
--- www.baidu.com ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 32.939/35.522/38.658 ms
[root@FriendlyARM /]# stop-wifi
cfg80211: Calling CRDA to update world regulatory domain
[root@FriendlyARM /]#
就绪 Serial: COM1 24, 23 24行, 83列 VT100 数字

```

7、再次执行命令 `ping www.baidu.com` 不能解析域名，wifi 已经断开。

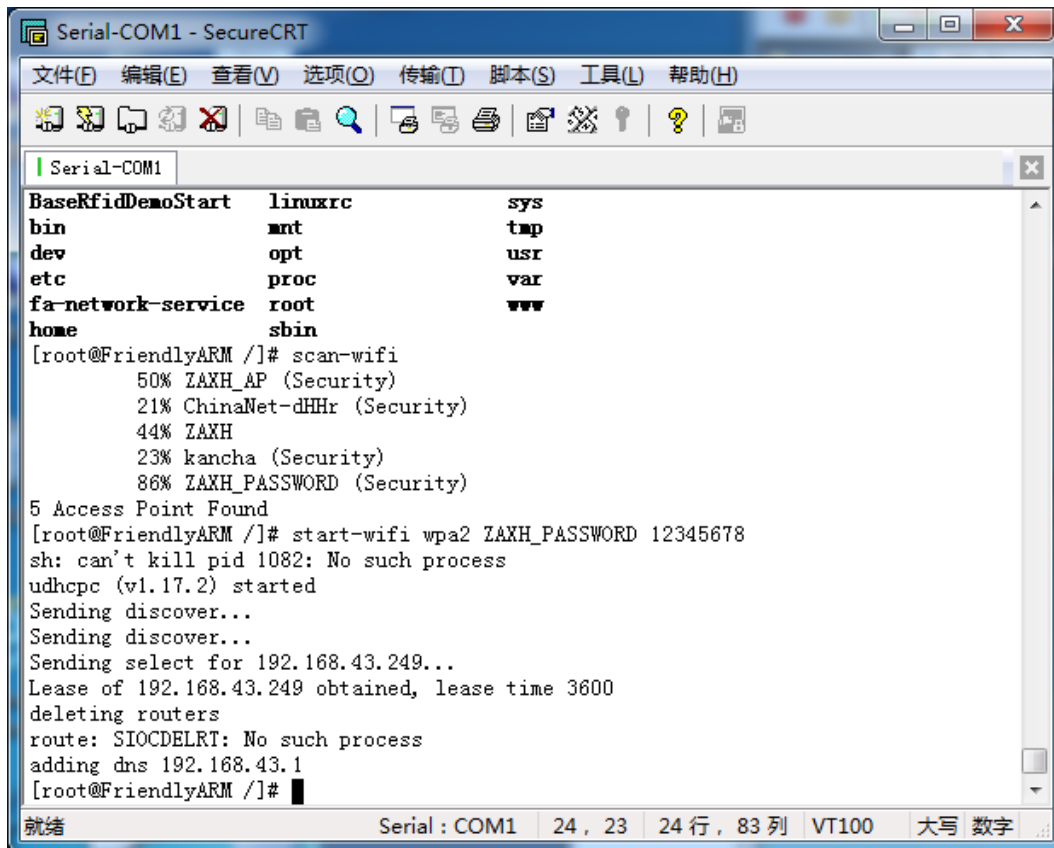


```

Serial-COM1 - 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-COM1
Lease of 192.168.0.70 obtained, lease time 604800
deleting routers
route: SIOCDELRT: No such process
adding dns 192.168.0.1
[root@FriendlyARM /]# ping www.baidu.com
PING www.baidu.com (119.75.217.56): 56 data bytes
64 bytes from 119.75.217.56: seq=0 ttl=52 time=38.658 ms
64 bytes from 119.75.217.56: seq=1 ttl=52 time=37.868 ms
64 bytes from 119.75.217.56: seq=2 ttl=52 time=37.886 ms
64 bytes from 119.75.217.56: seq=3 ttl=52 time=33.907 ms
64 bytes from 119.75.217.56: seq=4 ttl=52 time=32.939 ms
64 bytes from 119.75.217.56: seq=5 ttl=52 time=34.943 ms
64 bytes from 119.75.217.56: seq=6 ttl=52 time=32.989 ms
64 bytes from 119.75.217.56: seq=7 ttl=52 time=34.990 ms
^C
--- www.baidu.com ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 32.939/35.522/38.658 ms
[root@FriendlyARM /]# stop-wifi
cfg80211: Calling CRDA to update world regulatory domain
[root@FriendlyARM /]# ping www.baidu.com
ping: bad address 'www.baidu.com'
[root@FriendlyARM /]#
[root@FriendlyARM /]#
就绪 Serial: COM1 24, 23 24行, 83列 VT100 数字

```

8、开启 wpa2 加密 WiFi 接入点。

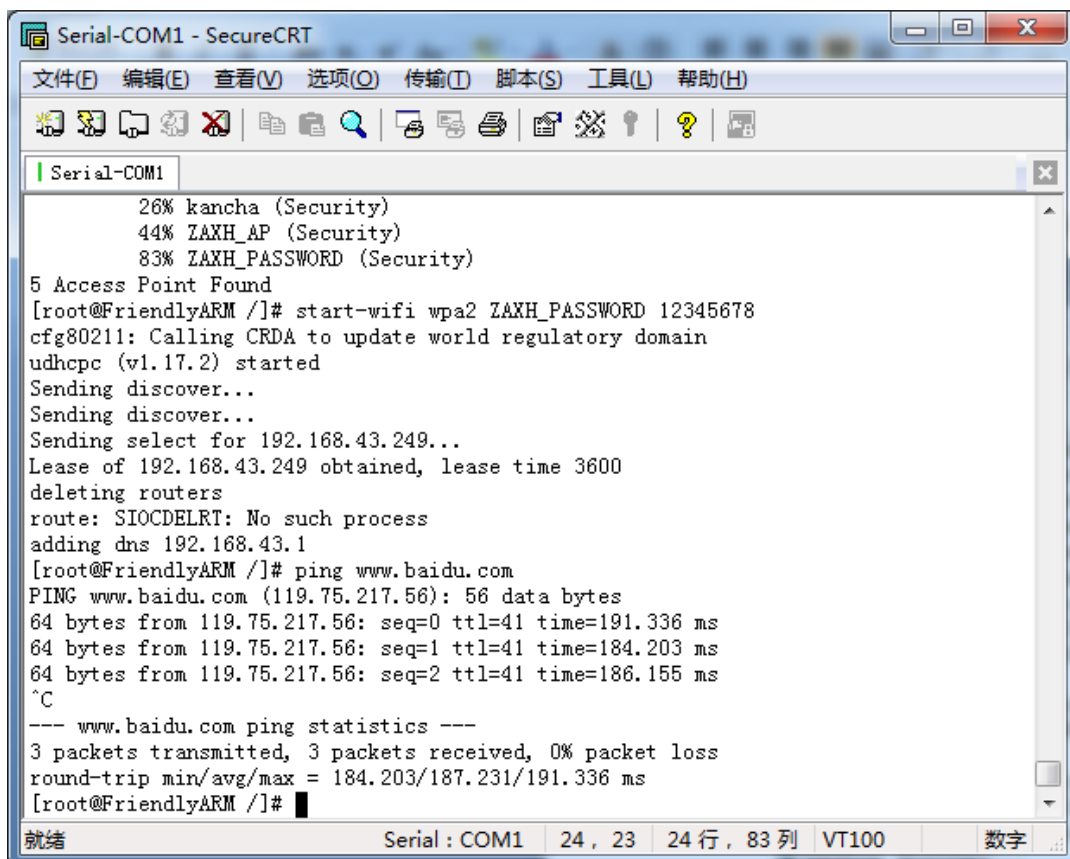


```
Serial-COM1
BaseRfidDemoStart  linuxrc      sys
bin                mnt          tap
dev                opt          usr
etc                proc         var
fa-network-service root        www
home               sbin

[root@FriendlyARM /]# scan-wifi
50% ZAXH_AP (Security)
21% ChinaNet-dHHR (Security)
44% ZAXH
23% kancha (Security)
86% ZAXH_PASSWORD (Security)
5 Access Point Found
[root@FriendlyARM /]# start-wifi wpa2 ZAXH_PASSWORD 12345678
sh: can't kill pid 1082: No such process
udhcpd (v1.17.2) started
Sending discover...
Sending discover...
Sending select for 192.168.43.249...
Lease of 192.168.43.249 obtained, lease time 3600
deleting routers
route: SIODELRT: No such process
adding dns 192.168.43.1
[root@FriendlyARM /]#
```

就绪 Serial: COM1 24, 23 24行, 83列 VT100 大写 数字

9、再次 ping www.baidu.com。



```
Serial-COM1
26% kancha (Security)
44% ZAXH_AP (Security)
83% ZAXH_PASSWORD (Security)
5 Access Point Found
[root@FriendlyARM /]# start-wifi wpa2 ZAXH_PASSWORD 12345678
cfg80211: Calling CRDA to update world regulatory domain
udhcpd (v1.17.2) started
Sending discover...
Sending discover...
Sending select for 192.168.43.249...
Lease of 192.168.43.249 obtained, lease time 3600
deleting routers
route: SIODELRT: No such process
adding dns 192.168.43.1
[root@FriendlyARM /]# ping www.baidu.com
PING www.baidu.com (119.75.217.56): 56 data bytes
64 bytes from 119.75.217.56: seq=0 ttl=41 time=191.336 ms
64 bytes from 119.75.217.56: seq=1 ttl=41 time=184.203 ms
64 bytes from 119.75.217.56: seq=2 ttl=41 time=186.155 ms
^C
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 184.203/187.231/191.336 ms
[root@FriendlyARM /]#
```

就绪 Serial: COM1 24, 23 24行, 83列 VT100 数字

2.1.4 实验说明

scan-wifi 执行扫描命令，以搜索附近的无线网络，被搜索到的带密码的网络将被标识 Security。

scan-wifi mode ssid password

start-wifi 连接到指定的无线网络接入点，根据不同的无线网路特性，会有不同的参数，网络类型 mode 可以有“wpa”，“wpa2”，“wep”或“none”，“none”表示没有不需要密码的无线网络。

ssid 表示要连接的无线网络名称。

Password 表示加密的无线网络需要的密码，none 模式不需要此参数。

stop-wifi 断开 ARM 开发板上面的 USB WiFi 连接。

2.1.5 实验总结

通过本实验，学习了在 ARM 平台上面使用 WiFi 模块连接到网络，从而可以在 ARM 开发板上面使用无线网络上网，在本实验的基础之上可以进一步开发，进行 SOCKET 编程，在 ARM 平台上，开发自己的游览器，或者搭建自己的服务器，这些将留给学生自己去学习。

2.1.6 实验思考

如何搭建自己的 WiFi 热点？

2.2 搭建 WiFi Access Point

2.2.1 实验目的

学习搭建 WiFi 接入点。

2.2.2 实验设备

- 1、安装有 window XP 的电脑
- 2、USB 无线网卡 SAGEM XG-760

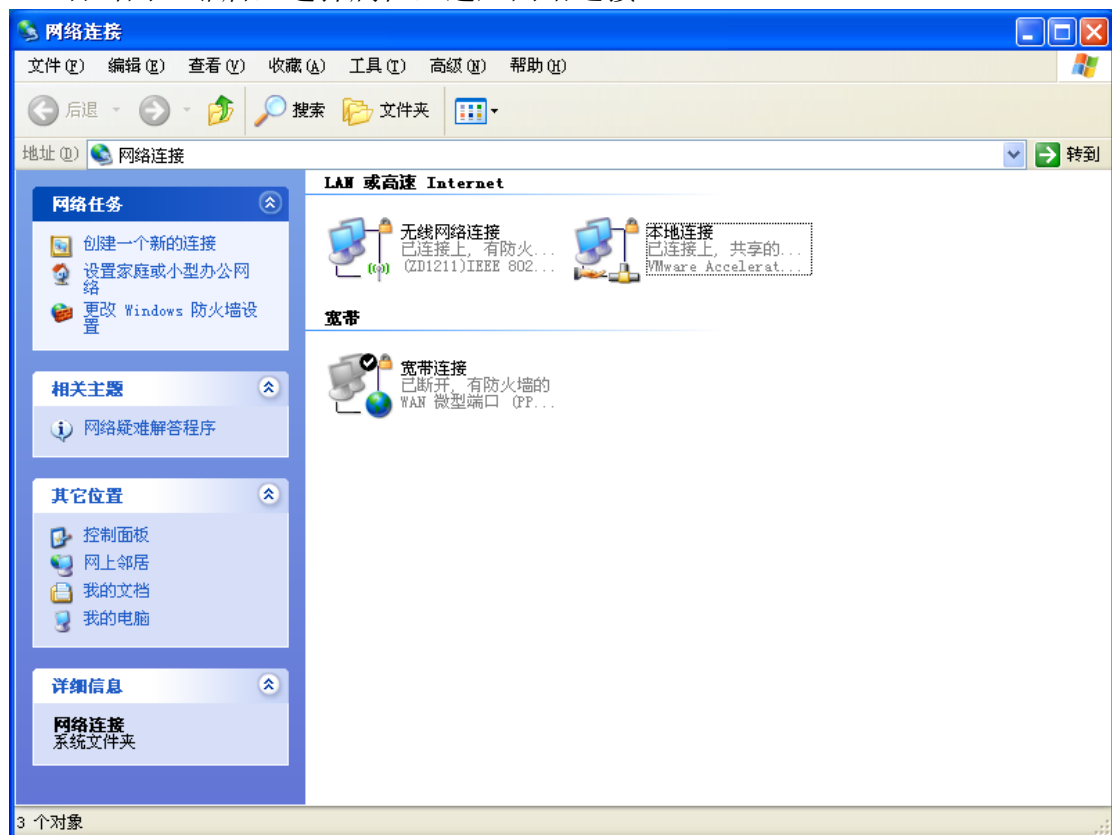
2.2.3 准备实验环境

本实验主要是使用 SAGEMXG-760 USB 无线网卡制作 WiFi 接入点（也可以利

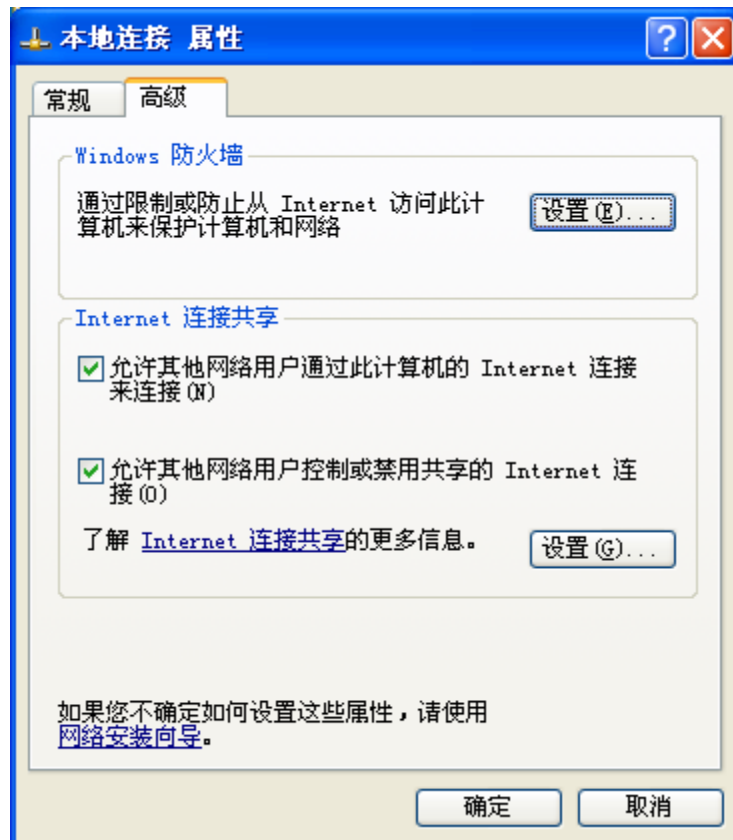
用笔记本和手机上面的 WiFi 制作接入点，本实验不讲解笔记本和手机制作 WiFi 接入点)，在 window XP 系统的台式电脑（笔记本）上面安装 XG_760A_ZD1211 USB_Install_4_13_0_0_ALL.exe，该软件默认安装即可。连接 USB 无线网卡 SAGE M XG-760，安装驱动，驱动路径默认为 C:\Program Files\ZyDAS Technology Corporation\ZyDAS_802.11g_Utility\InsDrvTemp_USB。安装完成后重启一下。

2.2.4 实验步骤

- 1、确保你电脑已经连接网络。
 - 2、将先在连接的网络设置为共享。
- 右击网上邻居，选择属性，进入网络连接。

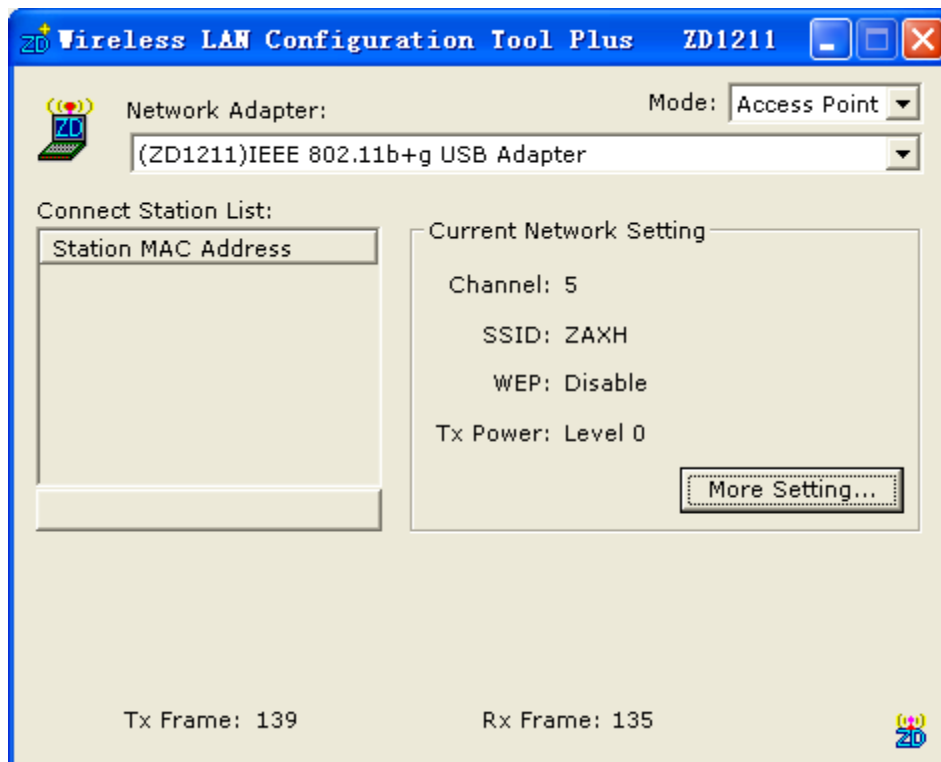


右击本地连接（因为本机是通过是有线上网的），选择属性，然后选择高级，



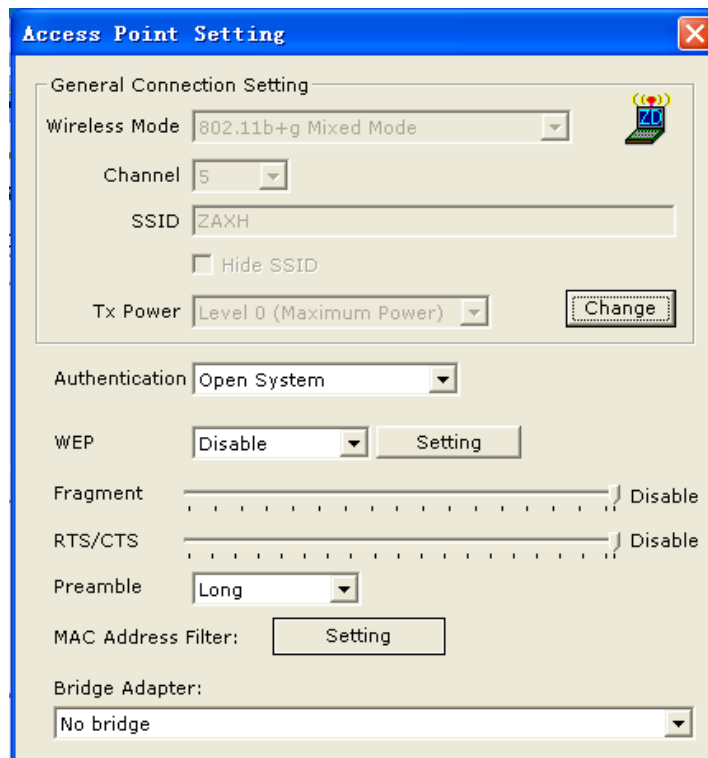
勾选“允许其他网络用户通过此计算机的 Internet 连接来连接”，然后确定。

3、启动 C:\Program Files\ZyDAS Technology Corporation\ZyDAS_802.11g_Utility\ZDWlan.exe。



5、选择模式为 Access Point，即 WiFi 的一个接入点。

6、 点击 Mode Setting...



7、 设置通用连接参数。

点击 Change,

Wireless Mode 选择为 802.11b+ Mixed Mode,

Channel 信道可以从 1-13 选择,

SSID 接入点名称可自行设置, 本实验设置名称为 ZAXH,

Tx Power 设置 WiFi 信号强度 (0-3 种级别, 0 级别信号强度最大),

设置完成后, 点击 Apply,

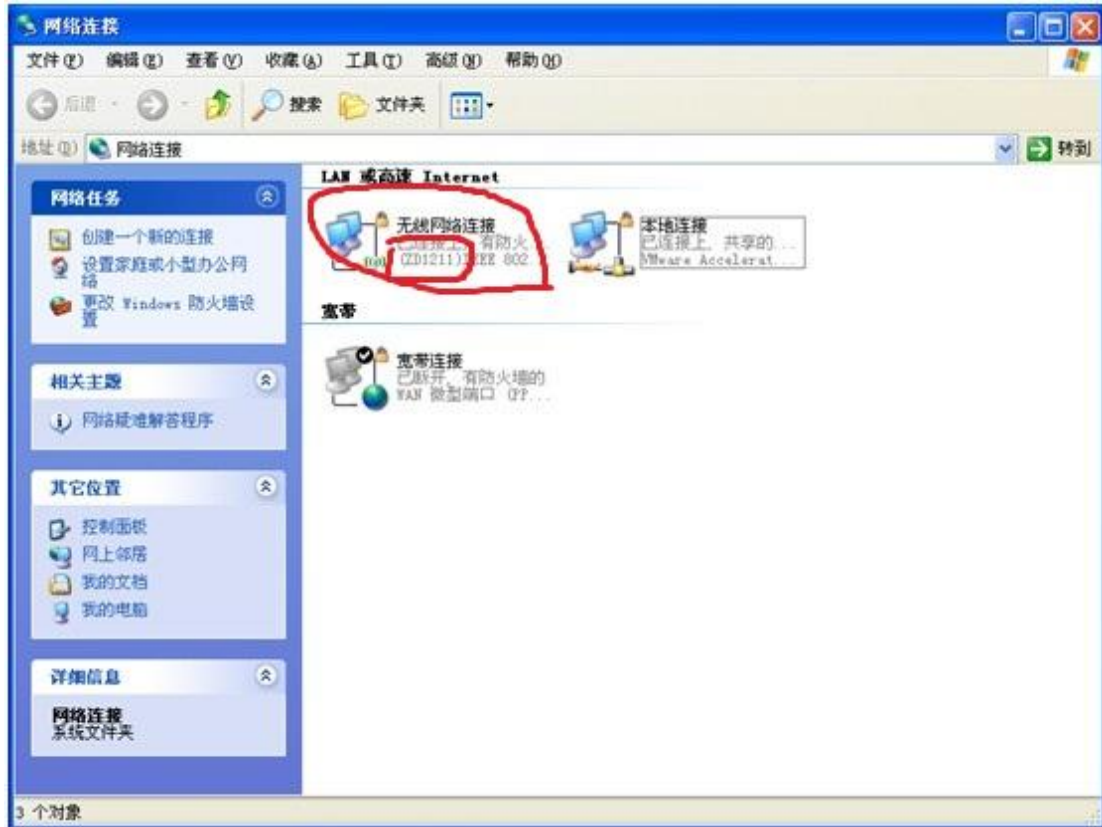
Authentication (认证) 选择 Open System,

WEP 选择 Disable,

其他默认即可。

8、 设置无线网络连接接入点 IP 和 DNS 等。

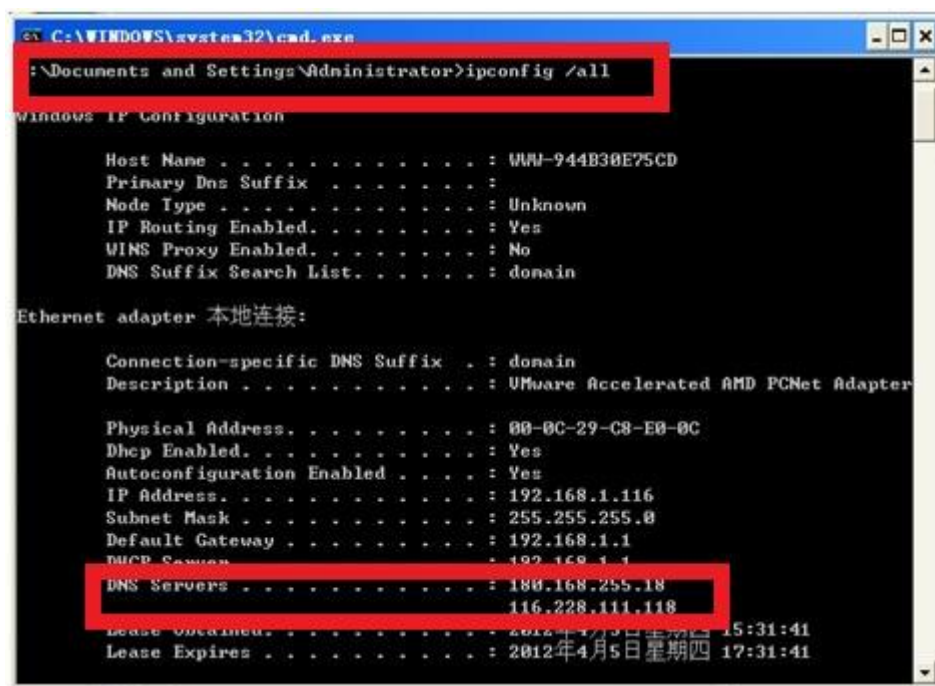
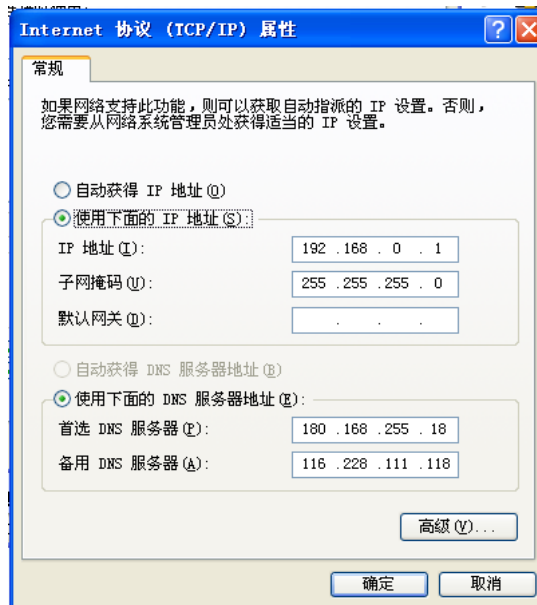
右击网上邻居, 选择属性, 进入网络连接。



然后右击带有 ZD1211 的无线网络连接，选择属性，然后选择 Internet 协议 (TCP/IP)，再点击属性。



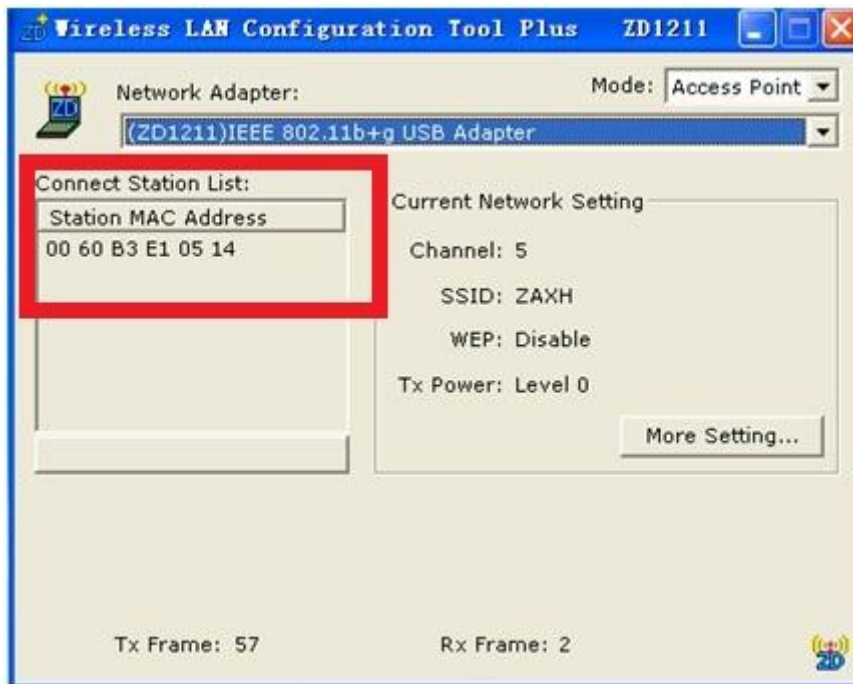
设置 IP 等参数如下, DNS 请根据本机电脑设置,可以在命令行输入 ipconfig /all 查询。



点击确定, 所有配置完成。

9、现在可以根据上一个实验来测试新建立的无密码的 WiFi 接入点。

当有设备接入到该无线网络, 该设备的 MAC 地址, 将显示在 Static MAC Address 栏。



2.2.5 实验说明

XG_760A_ZD1211USB_Install_4_13_0_0_ALL.exe 所带的驱动不支持 Win7，不能利用该应用软件在 Win7 下面组建 Access Point。但是 Win7 是可以通过该 USB 无线网卡 SAGEM XG-760 上无线 WiFi 网。

2.2.6 实验总结

本实验主要学习了如何搭建自己无线局域网，通过 SAGEM 760A 无线模块新建了一个 WiFi 热点，使嵌入式和手机等模块可以通过该 WiFi 热点上网。

2.2.7 实验思考

什么是虚拟 WiFi？什么是 SoftAP？怎么用 win7 系统笔记本做 wifi 热点？

第三章 蓝牙基础实验

3.1 蓝牙模块测试实验

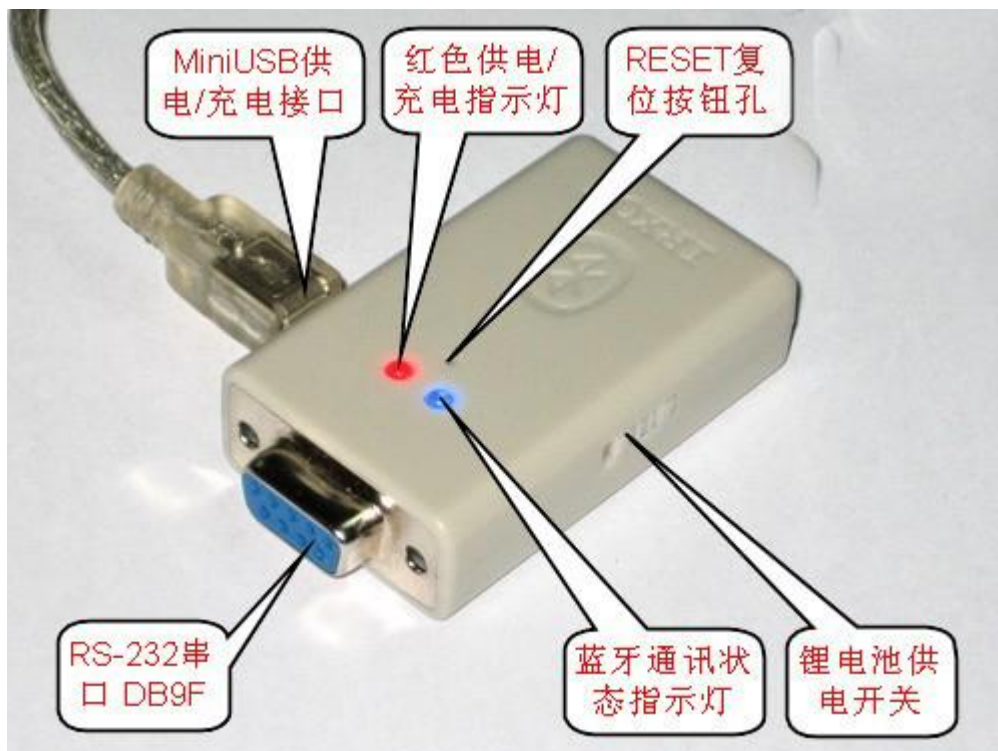
3.1.1 实验目的

熟悉蓝牙模块。

3.1.2 实验设备

- 1、主机蓝牙模块一个（可用带蓝牙 android 手机代替）
- 2、从机蓝牙模块一个
- 3、平行串口线两根

3.1.3 准备知识



- 1、主机和从机成对使用：

主机和从机分别连接串口或 UART，主机会记忆与它完成配对的从机蓝牙地址，主机上电后就会找它所记忆地址的从机，此时主机的蓝牙状态指示灯闪烁地较慢，大概每秒钟闪一次。

要让主机和新的从机配对，则按一下 RESET 复位按钮，复位后主机的蓝牙状

态指示灯开始快速闪烁，主机放弃原来配对的从机地址，重新寻找新的从机。

从机上电之后，在完成配对之前，蓝牙状态指示灯会一直快速地闪烁。在主机找到新的从机并与之完成配对并成功连接后，主机和从机的蓝牙状态指示灯不再闪烁，变为常亮状态。

主机和从机连接，即使两者的波特率不同，也可以互传数据。主从成对使用无需设置，两个模块都上电后就可以进行串口通讯。

如果主机和从机距离太远导致连接中断，主机和从机的蓝牙状态指示灯会恢复到未连接之前的闪烁状态，当它们再次靠近后，会自动恢复到连接状态。

2、从机可与任何支持蓝牙 SPP 串口服务(Series Port Profile)的设备连接使用。如蓝牙笔记本电脑、蓝牙适配器、蓝牙手机、蓝牙 PDA 等等。从机只能等待蓝牙主设备来寻找，当主设备找到从机后，从主设备发出配对请求，配对密码固定为“0000”或“1234”。

3.1.4 实验步骤

1、准备工作

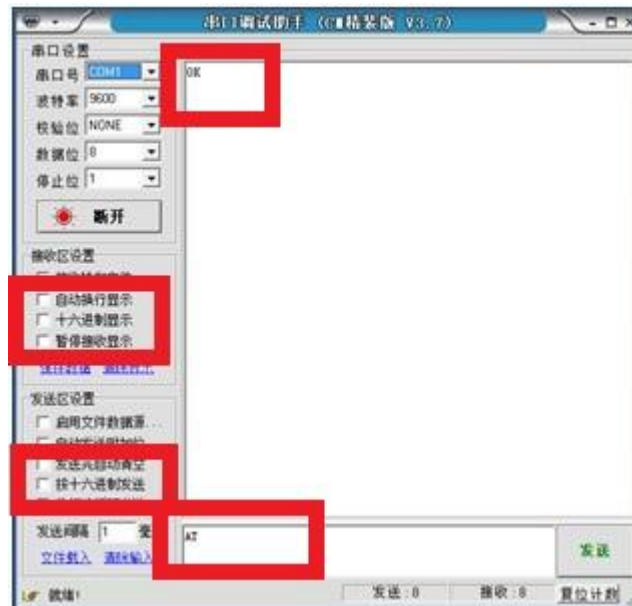
首先将主或从机连接至电脑的 RS-232 串口（电脑没有串口，可以使用 USB 转串口线替代），并确认串口号，这里假设适配器是连接在 COM1 上。

AT 命令既适用于主机，也适用于从机。但在发出 AT 命令前，请确保主机或从机处于指示灯闪烁的非连接状态。不要同时开启主机和从机，否则，AT 命令会被当作通讯数据发到通讯对象的一边。

默认通讯速率是 9600bps，必需将 COM1 的通讯速率也设置成与适配器一致的速率，即 9600，才能正常通讯。

2、打开串口助手

打开串口助手，输入大写“AT”，点击发送，模块将返回”OK”，说明 AT 通讯测试成功。



3、修改适配器的通讯速率

适配器的出厂默认通讯速率是 9600bps，要使适配器在更高的速率下通讯，

可以通过发送 AT 命令对适配器的速率进行修改。修改速率的 AT 命令格式是“AT+BAUDn”，其中的 n 是个变量，代表不同的速率，如下表所示。比如向适配器发送“AT+BAUD7”，7 对应下表的 57600bps 速率，如果返回“OK57600”，就说明速率修改成功。

n	对应速率bps
1	1200
2	2400
3	4800
4	9600
5	19200
6	38400
7	57600
8	115200



如上图所示，适配器的速率已成功修改为 57600，但电脑串口的速率还是原来的 9600，电脑串口速率与适配器速率已不再相同，这将会使以后的 AT 命令操作无法继续，所以应该及时调整电脑串口的速率，使其与适配器的速率一致。一下实验都是在 9600 波特率下面实验，请自行设置为波特率为 9600。



4、修改适配器的蓝牙

名称本产品的默认蓝牙名称是 irxon, 用户可以通过发送 AT 命令为适配器取一个更形象化的名字。

AT 命令的格式为: AT+NAMEname, 返回: OKsetname 即为修改成功。

参数 name 为所要设置的适配器蓝牙名称, 也就是被其它蓝牙设备搜索到时显示出的名称, 不能超过 20 个字符。

如发送 “AT+NAMEirxon”, 返回 “OKsetname”, 这时适配器的蓝牙名称即被改成了 “irxon”。

5、修改适配器的配对密码

用户可以通过发送 AT 命令修改适配器 的蓝牙配对码。

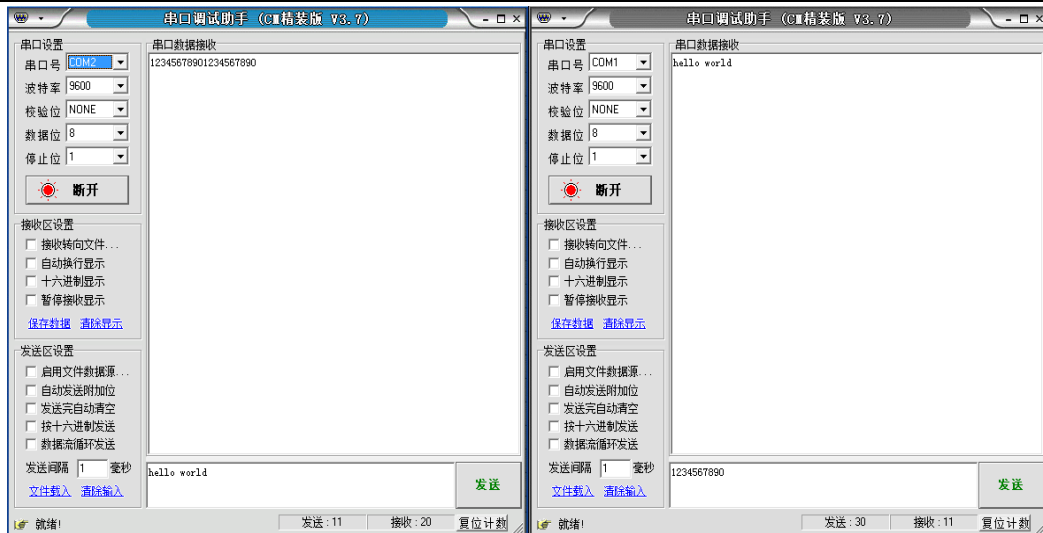
AT 命令的格式为: AT+PINpassword, 返回: OKsetpassword 即为修改成功。参数 password 为所要设置的蓝牙配对密码, 必须是 4-8 位的数字。

如发送 “AT+PIN0000”, 返回 “OKsetpassword”, 这时适配器的蓝牙配对密码即被改为 “0000”。

6、主从机通讯

同时开启主机和从机, 并都通过串口与电脑相连接, 主从机传送数据。

主机按一下 RESET 复位按钮, 等到主从机蓝灯都不闪烁, 说明主从机已经配对连接。



7、手机与从机通讯

只开启从机模块，带蓝牙的 android 手机安装“蓝牙串口助手 PRO.apk”软件，并打开该软件。

手机软件操作如图所示：

点击“连接设备”，

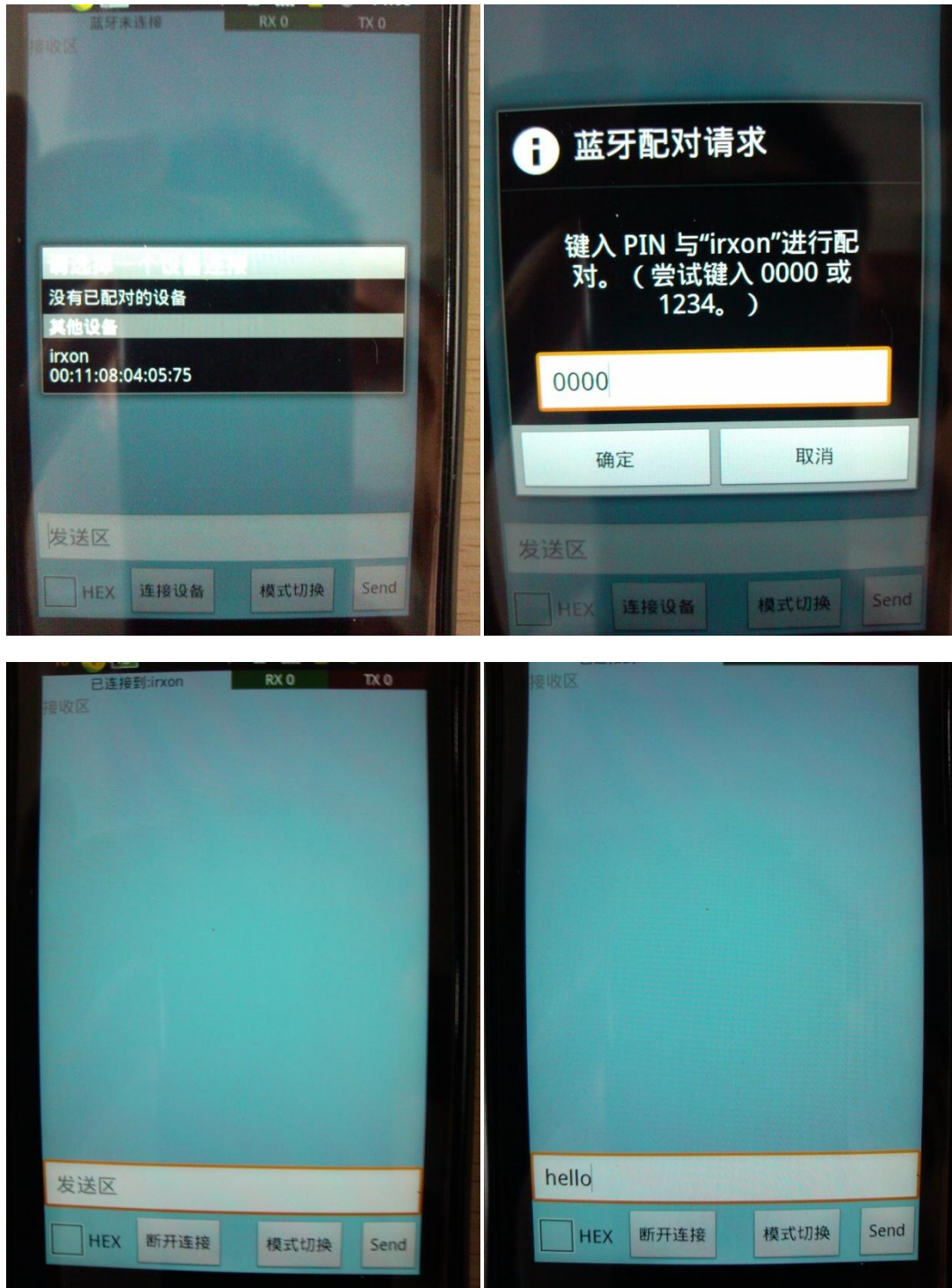
点击“扫描新设备”，

选择扫描到的设备，

输入配对密码，连接到蓝牙设备。

手机发送数据给从机设备，从机设备可以发送数据给手机。







3.1.5 实验总结

本实验主要讲解了如何使用串口蓝牙模块，通过一些基本的 AT 指令能够对蓝牙模块做一些简单的设置，主从机配对可以实现主机和从机的数据交互，手机和从机配对，可以实现手机和从机的数据交互。

3.1.6 实验思考

本实验实现了手机与串口蓝牙从机模块的数据交互，如果将从机模块连接到单片机上面，即可通过手机发送命令给单片机，控制单片机执行相应的操作，请同学们思考如何去利用此模块实现一个简单的小应用。

3.2 简易 ARM 版蓝牙助手

3.2.1 实验目的

在 ARM 平台上使用蓝牙模块
简单 QT 小软件学习

3.2.2 实验设备

- 1、ARM 开发板
- 2、从机蓝牙模块一个
- 3、android 手机（可用主机蓝牙模块代替）

3.2.3 准备知识

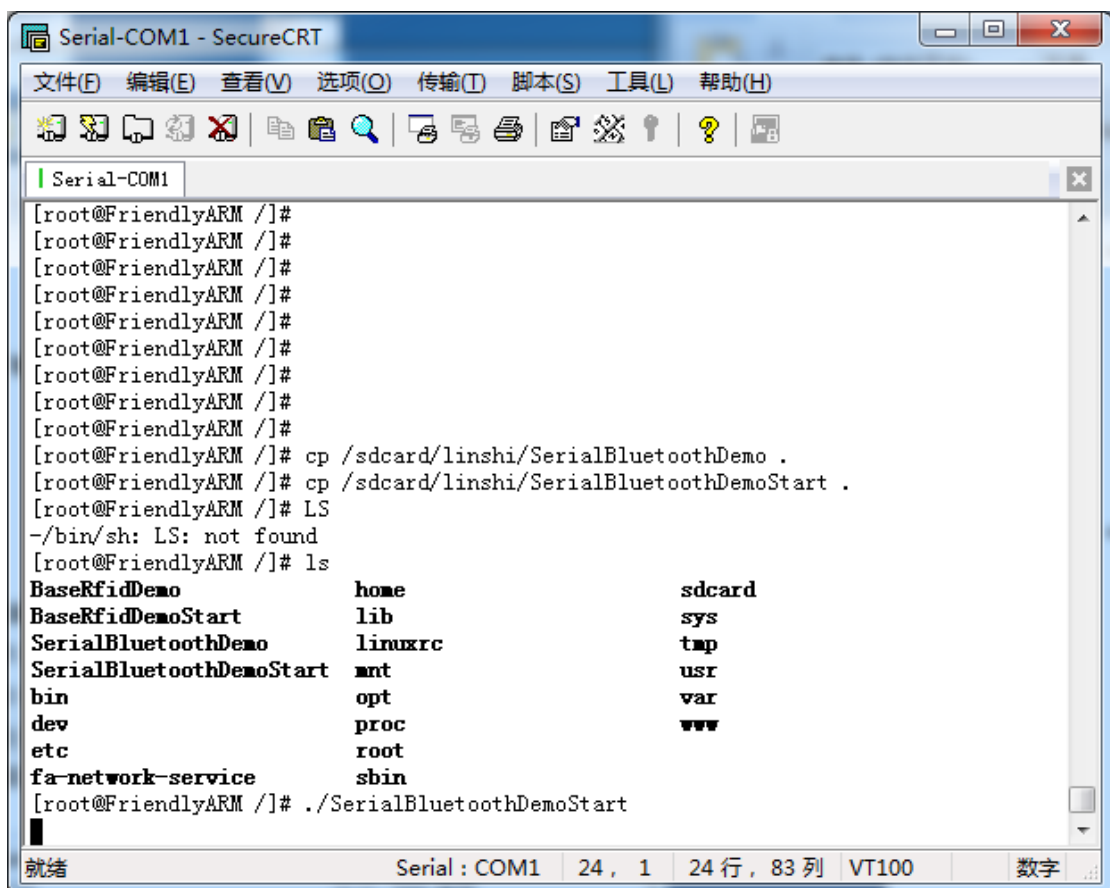
复习 RFID 图形化开发之环境搭建 实验

3.2.4 实验原理

- 1、主机和从机数据通讯
- 2、手机与从机数据通信
- 3、ARM 平台下简单串口助手的开发

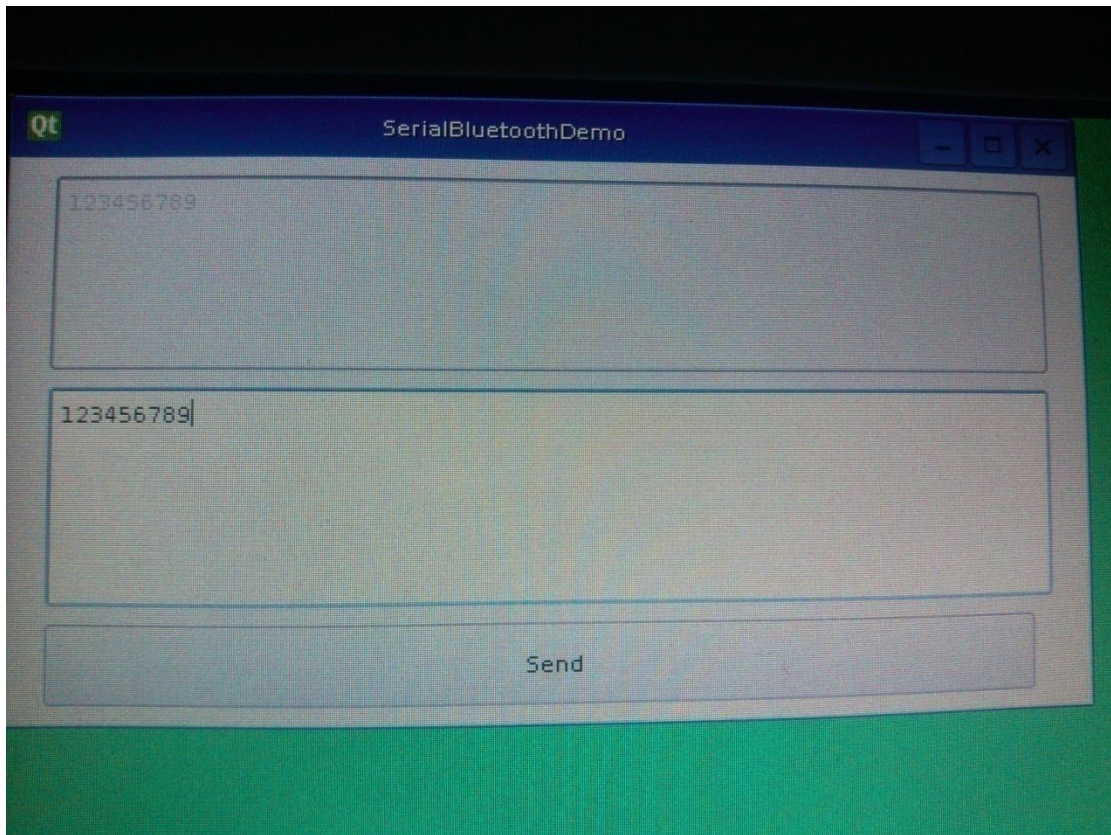
3.2.5 实验步骤

将“无线模块实验\蓝牙基础实验\简易 ARM 版蓝牙助手\可执行文件”下的两个文件 SerialBluetoothDemo 和 SerialBluetoothDemoStart 通过 SD 复制到 ARM 开发板上，并执行 SerialBluetoothDemoStart。



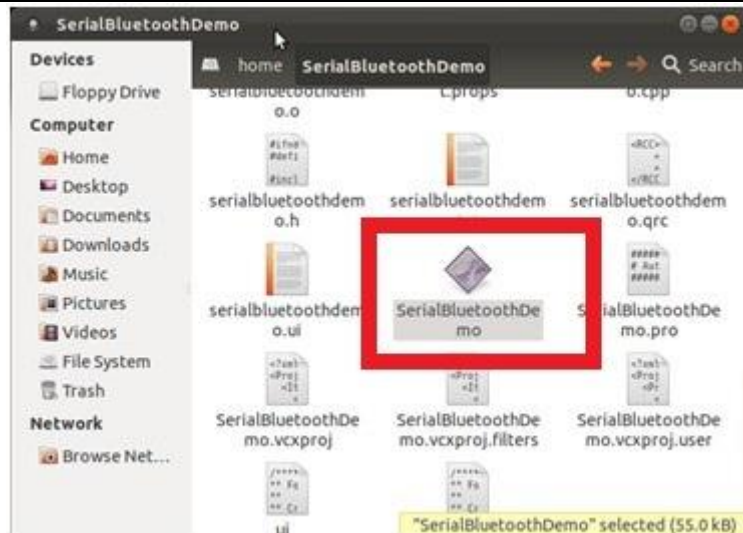
```
Serial-COM1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
Serial-COM1
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]#
[root@FriendlyARM /]# cp /sdcard/linshi/SerialBluetoothDemo .
[root@FriendlyARM /]# cp /sdcard/linshi/SerialBluetoothDemoStart .
[root@FriendlyARM /]# LS
~/bin/sh: LS: not found
[root@FriendlyARM /]# ls
BaseRfidDemo          home                  sdcard
BaseRfidDemoStart     lib                   sys
SerialBluetoothDemo   linuxrc               tmp
SerialBluetoothDemoStart  mnt                   usr
bin                   opt                   var
dev                   proc                  www
etc                   root
fa-network-service   /sbin
[root@FriendlyARM /]# ./SerialBluetoothDemoStart
就绪      Serial : COM1  24, 1  24 行, 83 列  VT100  数字
```

ARM 开发板上将启动简易串口蓝牙助手，默认波特率是 9600。软件启动后，将从机蓝牙模块插在 ARM 板的串口 3 上面，手机启动蓝牙助手，扫描蓝牙设备，连接蓝牙设备，手机就可以和 ARM 通过蓝牙通信了。



实验源代码位于 无线模块实验\蓝牙基础实验\简易 ARM 版蓝牙助手\SerialBluetoothDemo\SerialBluetoothDemo 文件夹，将该文件夹下面的代码拷贝到已经安装好图形化开发环境的 Ubuntu(已经安装好交叉编译环境和 QT4.7)，然后在该文件夹下面执行如下命令。

```
root@ubuntu: /home/SerialBluetoothDemo
root@ubuntu:/home/SerialBluetoothDemo# pwd
/home/SerialBluetoothDemo
root@ubuntu:/home/SerialBluetoothDemo# qmake -project
root@ubuntu:/home/SerialBluetoothDemo# qmake
RCC: Warning: No resources in 'serialbluetoothdemo.qrc'.
root@ubuntu:/home/SerialBluetoothDemo# make
/opt/Qt4.7/bin/uic myinputpanelform.ui -o ui_myinputpanelform.h
/opt/Qt4.7/bin/uic serialbluetoothdemo.ui -o ui_serialbluetoothdemo.h
arm-linux-g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_
_NETWORK_LIB -DQT_CORE_LIB -DQT_SHARED -I/opt/Qt4.7/mkspecs/qws/linux-arm-g++ -I
-I/opt/Qt4.7/include/QtCore -I/opt/Qt4.7/include/QtNetwork -I/opt/Qt4.7/includ
e/QtGui -I/opt/Qt4.7/include -I. -I. -I. -o main.o main.cpp
arm-linux-g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_
_NETWORK_LIB -DQT_CORE_LIB -DQT_SHARED -I/opt/Qt4.7/mkspecs/qws/linux-arm-g++ -I
-I/opt/Qt4.7/include/QtCore -I/opt/Qt4.7/include/QtNetwork -I/opt/Qt4.7/includ
e/QtGui -I/opt/Qt4.7/include -I. -I. -I. -o myinputpanel.o myinputpanel.cpp
arm-linux-g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_
_NETWORK_LIB -DQT_CORE_LIB -DQT_SHARED -I/opt/Qt4.7/mkspecs/qws/linux-arm-g++ -I
-I/opt/Qt4.7/include/QtCore -I/opt/Qt4.7/include/QtNetwork -I/opt/Qt4.7/includ
e/QtGui -I/opt/Qt4.7/include -I. -I. -I. -o myinputpanelcontext.o myinputpanelco
ntext.cpp
arm-linux-g++ -c -pipe -O2 -Wall -W -D_REENTRANT -DQT_NO_DEBUG -DQT_GUI_LIB -DQT_
_NETWORK_LIB -DQT_CORE_LIB -DQT_SHARED -I/opt/Qt4.7/mkspecs/qws/linux-arm-g++ -I
-I/opt/Qt4.7/include/QtCore -I/opt/Qt4.7/include/QtNetwork -I/opt/Qt4.7/includ
```



生成可执行文件 SerialBluetoothDemo。

然后编辑可以 sh 文件 SerialBluetoothDemoStart

```
#!/bin/sh

if [ -e /etc/friendlyarm-ts-input.conf ] ; then
    . /etc/friendlyarm-ts-input.conf
fi
true${TSLIB_TSDEVICE:=/dev/touchscreen}

TSLIB_CONFFILE=/etc/ts.conf

export TSLIB_TSDEVICE
export TSLIB_CONFFILE

export TSLIB_PLUGINDIR=/usr/lib/ts
export TSLIB_CALIBFILE=/etc/pointercal

export QWS_DISPLAY=:1
export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
export PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin

if [ -c ${TSLIB_TSDEVICE} ] ; then
    export QWS_MOUSE_PROTO="Tslib MouseMan:/dev/input/mice"
    if [ ! -s /etc/pointercal ] ; then
        rm /etc/pointercal
        /usr/bin/ts_calibrate
    fi
else
    export QWS_MOUSE_PROTO="MouseMan:/dev/input/mice"
fi
```

```
export QWS_KEYBOARD=TTY:/dev/tty1

export HOME=/root

cd /
./SerialBluetoothDemo -qws -font wenquanyi
hotplug
```

然后按照前面的步骤将两个文件 `SerialBluetoothDemo` 和 `SerialBluetoothDemoStart` 通过 SD 复制到 ARM 开发板上面，并执行 `SerialBluetoothDemoStart`。

3.2.6 代码解析

Serialbluetoothdemo.cpp

```
#include "serialbluetoothdemo.h"
#include <qlineedit.h>
#include <qprogressbar.h>
#include <qtimer.h>
#include <qapplication.h>
#include <qmessagebox.h>
#include <qstringlist.h>

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <linux/fs.h>
#include <errno.h>
#include <string.h>
#include <termio.h>

SerialBluetoothDemo::SerialBluetoothDemo(QWidget *parent, Qt::WFlags flags)
    : QWidget(parent, flags)
{
    ui.setupUi(this);
    m_fd = openSerialPort();    /*打开串口*/
    if (m_fd < 0) {
        QMessageBox::warning(this, tr("Error"), tr("Fail to open serial port!"));
        return ;
    }
}
```

```
}
m_notifier = new QSocketNotifier(m_fd, QSocketNotifier::Read, this);    /* 监控是否有数据可读*/
connect (m_notifier, SIGNAL(activated(int)), this, SLOT(remoteDataIncoming()));
connect (ui.m_sendButton, SIGNAL(clicked()), this, SLOT(sendButtonClicked()));
}

SerialBluetoothDemo::~SerialBluetoothDemo()
{
    if (m_notifier) {
        delete m_notifier;
        m_notifier = 0;
    }

    if (m_fd >= 0) {
        ::close(m_fd);
        m_fd = -1;
    }
}

void SerialBluetoothDemo::sendButtonClicked()    /*启动发送*/
{
    QString text = ui.m_sendEdit->toPlainText();
    if (text.isEmpty()) {
        return ;
    }
    ::write(m_fd, text.toLatin1(), text.length());
    ui.m_sendEdit->setText("");
}

int SerialBluetoothDemo::openSerialPort()    /*打开串口*/
{
    int fd = -1;

    const char *devName = "/dev/ttySAC3";    /*默认选择串口2*/
    fd = ::open(devName, O_RDWR|O_NONBLOCK);
    if (fd < 0) {
        return -1;
    }

    termios serialAttr;
    memset(&serialAttr, 0, sizeof serialAttr);
    serialAttr.c_iflag = IGNPAR;
    serialAttr.c_cflag = B9600 | HUPCL | CS8 | CREAD | CLOCAL;    /*默认波特率
```

```
设置为9600*/
serialAttr.c_cc[VMIN] = 1;
if (tcsetattr(fd, TCSANOW, &serialAttr) != 0) {
    return -1;
}
return fd;
}

void SerialBluetoothDemo::remoteDataIncoming()    /*读取接收数据*/
{
    char c;
    if (read(m_fd, &c, sizeof c) != 1) {
        QMessageBox::warning(this, tr("Error"), tr("Receive error!"));
        return;
    }
    ui.m_receiveEdit->insertPlainText(QString(QChar(c)));
}
```

全部源代码位于 无线模块实验\蓝牙基础实验\简易 ARM 版蓝牙助手\SerialBluetoothDemo\SerialBluetoothDemo 请自行查看。

注：本实验代码只能在 linux 环境下面编译，不能在 windows 环境下面编译。

3.2.7 实验总结

本次实验主要是做了一个简易的串口蓝牙数据接收发送端，主要是使用串口蓝牙模块，通过简单的编程可以将蓝牙模块编写到我们的软件中实现我们需要的无线控制，这些还需要同学们自己去完善，本实验只是提供了一个简单的小例子。

3.2.8 实验思考

完善本实例，做一个通过串口蓝牙控制的小软件。