实验二 节点-PC 串口通信实验

实验目的

本实验的目的是实现节点和 PC 间的串口双向通讯，通过串口连接，PC 可以从网络收集其他节点的数据，也可以发送数据或者命令到节点，因此，串口通信编程是无线传感器网络中的重要内容。

实验要求

根据例子提供的例子程序，详细了解程序结构，并尝试进行程序的修改运行；具体实验要求如下：

1.学会使用串口通信相关的接口函数，实现串口通信。

2.修改 BlinkToRadio 程序，添加串口收发，实现一个简单的基站功能。

3.了解串口双向通信的方法，学会使用 mig 工具以及 SerialForwarder

实验内容

1、TestSerial 例子程序

节点与 PC 之间的通信在 TinyOS 中被抽象为数据包源（packet source）。一个数据包源就是一种与节点双向通信的介质，可以是串口，也可以是 TCP socket，或是 SerialForwarder 工具（该工具后面介绍）。

先看以下示例程序，将一个节点连接到 PC，进入 TestSerial 例子程序目录，编译并烧录程序。

```
$ make telosb install
```

运行 TestSerial 程序

```
$java TestSerial -comm serial@/dev/ttyUSB0:telos
```

输出类似以下内容：

```
Sending packet 1
Received packet sequence number 4
Sending packet 2
Received packet sequence number 5
Sending packet 3
Received packet sequence number 6
Sending packet 4
Received packet sequence number 7
Received packet sequence number 8
Sending packet 5
```

```
Received packet sequence number 9
Sending packet 6
```

这时节点的 LED 灯会闪烁。此时表明节点与串口双向通信正常。

为了之后的实验方便，我们可以设置串口参数的环境变量：

```
export MOTECOM=serial@/dev/ttyUSB0:telos
```

2、基站程序示例

基站节点是无线传感器网络的重要组成部分，它负责与后台服务器进行串口通信以及与网络中的其他节点进行无线通信，是一个桥梁的作用。

取两个节点，一个节点烧录 BlinkToRadio 程序，一个烧录 BaseStation 程序，将两个节点都通电。可以看到 BaseStation 的 LED1 等闪烁，按住 BlinkToRadio 节点的 RESET，LED1 不闪烁。

BaseStation 节点的 LED0 闪烁表示它收到了网络包，LED1 闪烁表示将网络包发送到串口，LED2 闪烁表示网络包被丢弃，丢弃的原因可能是串口的带宽小于节点的无线带宽。

再将 BaseStation 节点连接到 PC，使用 Listen 命令读取串口的内容。

```
$java net.tinyos.tools.Listen -comm serial@/dev/ttyUSB0:telos
```

Listen 命令的功能是创建数据包源，然后打印出每一个监听到的包。输出的内容类似以下：

```
00 FF FF 00 00 04 22 06 00 02 00 01
00 FF FF 00 00 04 22 06 00 02 00 02
00 FF FF 00 00 04 22 06 00 02 00 03
00 FF FF 00 00 04 22 06 00 02 00 04
00 FF FF 00 00 04 22 06 00 02 00 05
00 FF FF 00 00 04 22 06 00 02 00 06
00 FF FF 00 00 04 22 06 00 02 00 07
00 FF FF 00 00 04 22 06 00 02 00 08
00 FF FF 00 00 04 22 06 00 02 00 09
00 FF FF 00 00 04 22 06 00 02 00 0A
00 FF FF 00 00 04 22 06 00 02 00 0B
```

BlinkToRadioC 应用的消息格式如下（忽略开始的 00 字节，表示 AM 数据包）：

- 目标地址 Destination address（2 bytes）
- 连接源地址 Link source address（2 bytes）
- 消息长度 Message length（1 bytes）
- 组号 Group ID（1 byte）
- Active Message handler 类型 Active Message handler type（1 byte）
- Payload（最大 28 bytes）

  源节点 ID Source mote ID（2 bytes）
  示例计数值 Sample counter（2 bytes）

3、MIG 及数据包对象

Listen 程序是与节点通讯最基础的方式。但是它只打印了二进制的包。显然这种方式不是十分易读。在实际中，往往需要读取这些二进制数据后，再根据二进制的内容分析其字段。TinyOS 提供了一种方便解析二进制数据包内容的工具-MIG（Message InterFace Generator）。MIG 可以为用户指定的消息结构建立一个 Java，Python 或 C 的接口，免除了解析二进制的麻烦。

进入到 TestSerial 目录下，输入 make clean 清除上次的编译结果。然后输入 make telosb，可以看到类似以下：

```
rm -rf build *.class TestSerialMsg.java
rm -rf _TOSSIMmodule.so TOSSIM.pyc TOSSIM.py
mkdir -p build/telosb
mig java -target=telosb -I%T/lib/oski
-java-classname=TestSerialMsg TestSerial.h TestSerialMsg
-o TestSerialMsg.java
javac *.java
    compiling TestSerialAppC to a telosb binary
ncc -o build/telosb/main.exe -Os -O -mdisable-hwmul -Wall -Wshadow -
DDEF_TOS_AM_GROUP=0x66
-Wnesc-all -DCC2420_DEF_CHANNEL=19 -target=telosb -fnesc-
cfile=build/telosb/app.c
-board= -I%T/lib/oski  TestSerialAppC.nc -lm
    compiled TestSerialAppC to build/telosb/main.exe
            6300 bytes in ROM
             281 bytes in RAM
msp430-objcoy --output-target=ihex build/telosb/main.exe
build/telosb/main.ihex
    writing TOS image
```

这个输出表示在编译 TinyOS 应用程序之前，Makefile 中要求先生成 TestSerialMsg.java 文件，然后将 TestSerialMsg.java 和 TestSerial.java 进行编译，最后再编译 TinyOS 应用程序。打开 TestSerial 目录下的 Makefile 文件，如下：

```
COMPONENT=TestSerialAppC
BUILD_EXTRA_DEPS += TestSerial.class
CLEAN_EXTRA = *.class TestSerialMsg.java

TestSerial.class: $(wildcard *.java) TestSerialMsg.java
	javac *.java

TestSerialMsg.java:
	mig java -target=null -java-classname=TestSerialMsg TestSerial.h
test_serial_msg -o $@
```

BUILD_EXTRA_DEPS 这行说明 TinyOS 应用生成以前要先编译 TestSerial.class。下面这行

```
TestSerial.class: $(wildcard *.java) TestSerialMsg.java
	javac *.java
```

说明 TestSerialMsg.java 是 TestSerial.class 的依赖文件，在编译生成 TestSerial.class 之前需要生成 TestSerialMsg.java。

TestSerialMsg.java 的生成是利用 mig 工具。

```
TestSerialMsg.java:
	mig java -target=null -java-classname=TestSerialMsg TestSerial.h
test_serial_msg -o $@
```

这行命令的相关参数的含义如下：

mig	调用 mig 命令
java	生成 class 文件
-target=null	无特定平台
-java-classname=TestSerialMsg	生成的 java 类的名字
TestSerial.h	数据包的定义头文件
test_serial_msg	头文件的中的结构名称
-o $@	输出

4、SerialForwarder 和其他数据包源

直接监听串口的问题就是只能与一个程序进行读取，因为这是一个硬件资源。并且需要监听程序物理连接到串口才可以进行读取。为解决以上限制，TinyOS 中的 SerialForwarder 工具提供了一个解决方案。

一般来说，SerialForwarder 工具连接一个真实的物理串口，然后在此之上创建一个虚拟数据包源，然后允许同时多个程序通过 TCO/IP 连接来访问这个数据包源。可以看出，SerialForwarder 相当于物理串口的读写包的代理。

一个 SerialForwarder 数据包源的语法如下：

```
sf@HOST:PORT
```

HOST 和 PORT 是可选的。默认状态是 localhost 和 9002 端口的。

运行 SerialForwarder 命令

```
java net.tinyos.sf.SerialForwarder
```

会弹出一个窗口。

这表示已经成功创建了一个数据包源 sf，它的地址是本地的 9001 端口。

还可以在这个 sf 上再次创建 sf 数据包源，如下：

```
java net.tinyos.sf.SerialForwarder -port 9003 -comm sf@localhost:9001
```

这条命令创建了第 2 个 sf，它的源是工作在 9001 端口上的第 1 个 sf，第一个 sf 的引用数会增加 1。

这里仅仅是演示需要，将第 2 个 sf（9003）关闭。

使用 MsgReader 工具来读取第一个 sf 的内容：

```
java net.tinyos.tools.MsgReader -comm sf@localhost:9002
BlinkToRadioMsg
```

可以观察到第一个 sf 的引用数加 1，并且使用 MsgReader 开始打印内容，与上一节结果类似。

除了串口和 SerialForwarder 以外，TinyOS 还支持 Crossbow MIB 600 类型的以太网卡，缺省端口是 10002。总结一下数据包源的语法格式如下：

语法	源
serial@PORT:SPEED	串口
sf@HOST:PORT	SerialForwarder
network@HOST:PORT	MIB 600

5、mote 节点向串口发送数据包

在 TinyOS 中向串口发送一个 AM 类型的数据包与通过 radio 发送是非常相似的。可以使用 AMSend 接口，调用 AMSend.send 发送数据包，然后处理 AMSend.sendDone 事件，不用设置 AM 地址字段。以下是一个对比图：

Serial	Radio
SerialActiveMessageC	ActiveMessageC

SerialAMSenderC	AMSenderC
SerialAMReceiverC	AMReceiverC

其中 SerialActiveMessageC 用来打开和关闭栈，其余两个分别为收发组件。组件接口上的相似，也带来了实现上的相似。如下所示：

Radio	Serial

components ActiveMessageC;
  components new AMSenderC(AM_BLINKTORADIOMSG);
 	
  BlinkToRadioC.Packet -> AMSenderC;
  BlinkToRadioC.AMPacket -> AMSenderC;
  BlinkToRadioC.AMSend ->
  AMSenderC;
  BlinkToRadioC.AMControl -> ActiveMessageC;
	components SerialActiveMessageC;
	  components new SerialAMSenderC(AM_BLINKTORADIOMSG);
	
	  BlinkToRadioC.Packet -> SerialAMSenderC;
	  BlinkToRadioC.AMPacket -> SerialAMSenderC;
	  BlinkToRadioC.AMSend ->
	  SerialAMSenderC;
	  BlinkToRadioC.AMControl -> SerialActiveMessageC;

6．PC 端写串口

在 PC 进行串口写时，可以使用 net.tinyos.tools.Send 函数，但是只能发送二进制的数据，使用该函数时需要设置 MOTECOM 环境变量，如下所示，其中指定了目标节点为 1，计数值为 1：

```
java net.tinyos.tools.Send 00 FF FF 00 00 04 00 06 00 01 00 01
```

除了使用自带的写串口函数，还可以编写 BlinkToRadio.java 类控制从 PC 端发送数据，将数据发送的指定的目标节点，从而更好的控制数据的转发和路由过程。

通过使用包 net.tinyos.message.* 中的 MoteIF 类,实现 PC 端自定义的数据收发，需要实现 MessageListener 接口：

```
public class BlinkToRadio implements MessageListener {
  private MoteIF moteIF;

  public BlinkToRadio(MoteIF moteIF) {
    this.moteIF = moteIF;
    this.moteIF.registerListener(new BlinkToRadioMsg(), this);
```

```
  }
```

其中使用方法 registerListener()方法注册自动生成的类 BlinkToRadioMsg()；

需要实现 messageReceived()函数，

```
  public void messageReceived(int to, Message message) {
    BlinkToRadioMsg msg = (BlinkToRadioMsg)message;
    System.out.println("\nReceived packet to: " + to + " nodeid: "+
msg.get_nodeid() +" counter: " + msg.get_counter());
    System.out.print("Input the nodeid and counter(like 1  2): ");
  }
```

其中参数 to 是消息的目的节点编号（单播或广播地址），默认会接收所有能监听到的数据包；

```
    BlinkToRadioMsg payload = new BlinkToRadioMsg();
    payload.set_nodeid(nodeid);
    payload.set_counter(counter);
    moteIF.send(2 , payload);
```

Payload 可以使用 get 和 set 方法获取或使用其中的数据，注意和 BlinkToRadio.h 头文件的定义的一致，send() 函数第一个参数为发送的目节点（可以指定单播地址）；

完成 BlinkToRadio.java 类之后还需要修改 Makefile 文件，添加编译依赖:

```
    BUILD_EXTRA_DEPS += BlinkToRadio.class
    BlinkToRadio.class: $(wildcard *.java) BlinkToRadioMsg.java
    javac  *.java
```

之后运行 make clean 、make telosb 重新编译，即可生成 BlinkToRadio.class 文件，可以使用这个类来进行串口的读写操作，命令如下：

```
    Java BlinkToRadio -comm serial@/dev/ttyUSB0:telos
```

获取用户输入和格式化输出的逻辑需要在 BlinkToRadio.java 中实现，参考 TestSerial.java 即可；

实验要求说明

1．修改 BlinkToRadio 程序实现简单的基站程序 RadioAndSerial．使得当 RadioAndSerial 接收到无线数据包时 Led2 闪烁，并将数据包转发到串口；当串口接收到数据包时，Led0 闪烁，并将数据包转发到无线模块；

2．使用 mig 创建 BlinkToRadioMsg 的 java 对象，BlinkToRadio 发送的消息由 RadioAndSerial 接收并转发到串口，然后使用 MsgReader 读取 BlinkToRadioMsg 对象，展示接收到的数据。

提示步骤：

1.mote 节点的串口读写与无线模块的读写基本一致，可以参考 testSerial 源码

2.本次实验中不需要使用定时器，当收到数据包时即进行转发，不用考虑数据包队列，延迟等.

3.按照 TestSerial 例子修改 makefile。注意，javac 后面应该接一个 tab 字符而不是空格。

4．设置环境变量 MOTECOM 可以减少参数的输入

使用 Mig 工具时如果遇见以下类似信息

```
warning: Cannot determine AM type for BlinkToRadioMsg
 (Looking for definition of AM_BLINKTORADIOMSG)
```

就需要将 AM 类型从 AM_BLINKTORADIO 改为 AM_BLINKTORADIOMSG，这是 mig 命名规则的需要。然后再次编译烧录。

3.BaseStation 节点与修改后的 BlinkToRadio 节点都通电。然后在 BlinkToRadio 目录输入：

```
java net.tinyos.tools.MsgReader -comm serial@/dev/ttyUSB0:telos
BlinkToRadioMsg
```

结果应该类似以下：

```
1152232617609: Message
  [nodeid=0x2]
  [counter=0x1049]

1152232617609: Message
  [nodeid=0x2]
  [counter=0x104a]

1152232617609: Message
  [nodeid=0x2]
  [counter=0x104b]
BlinkToRadio.class: $(wildcard *.java) BlinkToRadioMsg.java
	javac  *.java
```

```
1152232617621: Message
  [nodeid=0x2]
  [counter=0x104c]
```

4．（扩展部分）生成 BlinkToRadio.class 文件，PC 端通过该类收发串口的数据．除了正常收到串口的数据外，还可以从 PC 主动发送数据到基站，由基站转发到节点，可以观察到基站有发送数据的显示，节点收到数据后会亮灯；
类似如下：

```
serial@/dev/ttyUSB2:115200: resynchronising
Input the nodeid and counter(like 1  2): 3 1
Sending to 3 number is : 1
Received packet to: 3 nodeid: 3 counter: 1
Input the nodeid and counter(like 1  2): 2 1
Sending to 2 number is : 1
Received packet to: 1 nodeid: 1 counter: 3
Input the nodeid and counter(like 1  2): 4 1
Sending to 4 number is : 1
Received packet to: 3 nodeid: 4 counter: 1
```

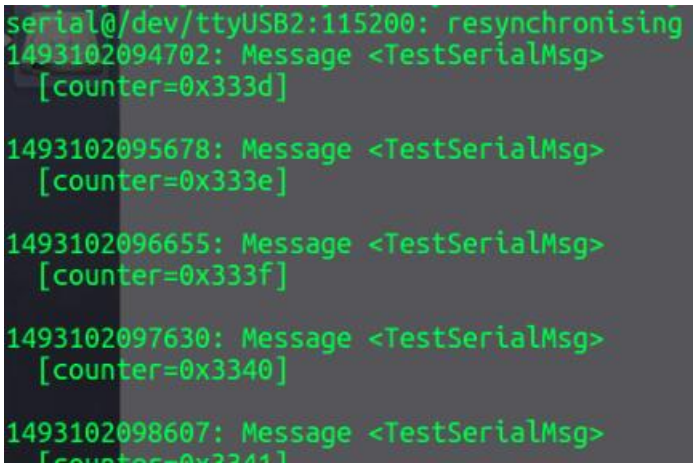

实验过程说明

首先需要修改节点通信的相关信道，选择 26 信道，在 Makefile 文件中进行修改，PFLAGS+=-DCC2420_DEF_CHANNEL=26。对 BlinkToRadio 程序中的 Makefile 文件进行修改，使用 mig 创建 BlinkToRadioMsg 的 java 对象 ：

```
    BlinkToRadioMsg.class: BlinkToRadioMsg.java
    javac BlinkToRadioMsg.java
    BlinkToRadioMsg.java:
     mig java -target=null $(CFLAGS) -java-classname=BlinkToRadioMsg
BlinkToRadio.h BlinkToRadioMsg -o $@
```

申请节点，自动分配为 3 号和 4 号节点，将 3 号节点作为普通节点，烧录 BlinkToRadio 程序，4 号节点作为基站节点，烧录 RadioAndSerial 程序，可以看到 4 号节点的 LED1 灯在不停的闪烁，说明基站节点不停的收到网络包并将其发送至串口。

使用 MsgReader 命令，结果如下图所示：



图中每两行代表基站节点收到的一个网络包，后面显示的是计数值。