



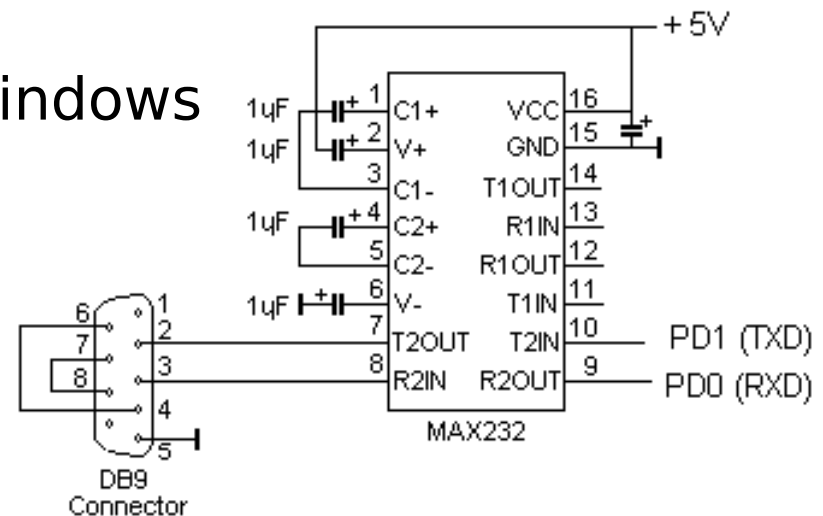
JavaComm: controllare dispositivi seriali in Java

Stefano Sanna
<http://www.gerdavax.it>



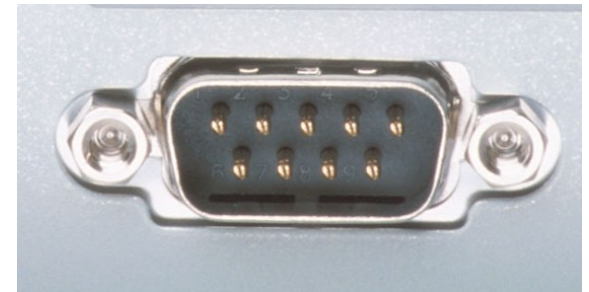
Parliamo di...

- Porte seriali: tante applicazioni interessanti!
- L'API JavaComm
 - La libreria
 - Uso in ambiente Linux e Windows
 - Applicazioni
- Trucchetti... :-)
- Conclusioni



Porte seriali

- L'interfaccia seriale RS-232 è uno standard molto diffuso. Alcune delle applicazioni più conosciute:
 - Connessione remota attraverso modem wired e wireless
 - GPS, Loran, bussole
 - Programmazione microcontrolli
 - Controllo motori passo-passo
 - Strumenti di misura (multimetri, sensori...)
 - Registratori di cassa, POS, controllo accessi (RFID)
 - Lettura dati stazioni meteorologiche
 - Interfaccia *economica* per giocattoli *costosi* (LEGO Mindstorm RIS 1.0-2.0)





Ancora porte seriali! :-)

- L'adozione di nuovi standard di connettività wired (USB) e wireless (IrDA, Bluetooth) non ha reso obsoleta la porta seriale!
- Appositi layer di emulazione permettono di *riciclare* protocolli, driver e applicazioni sui nuovi sistemi di trasporto:
 - USBSerial: emulazione seriale su bus USB
 - IrCOMM: emulazione seriale su link infrarosso
 - RFCOMM: emulazione seriale su link Bluetooth



JavaComm

- E' una libreria opzionale che fornisce una API per l'accesso completo alle porte seriali (RS-232) e parziale alle parallele (IEEE-1284)
- Principali caratteristiche:
 - Astrazione rispetto all'API del sistema operativo
 - Discovery delle porte disponibili
 - Modello ad eventi per dati disponibili e ownership
 - Accesso a basso livello:
 - Segnali EIA232 standard DTR, CD, CTS, RTS, DSR
 - Opzioni hardware e software flow-control

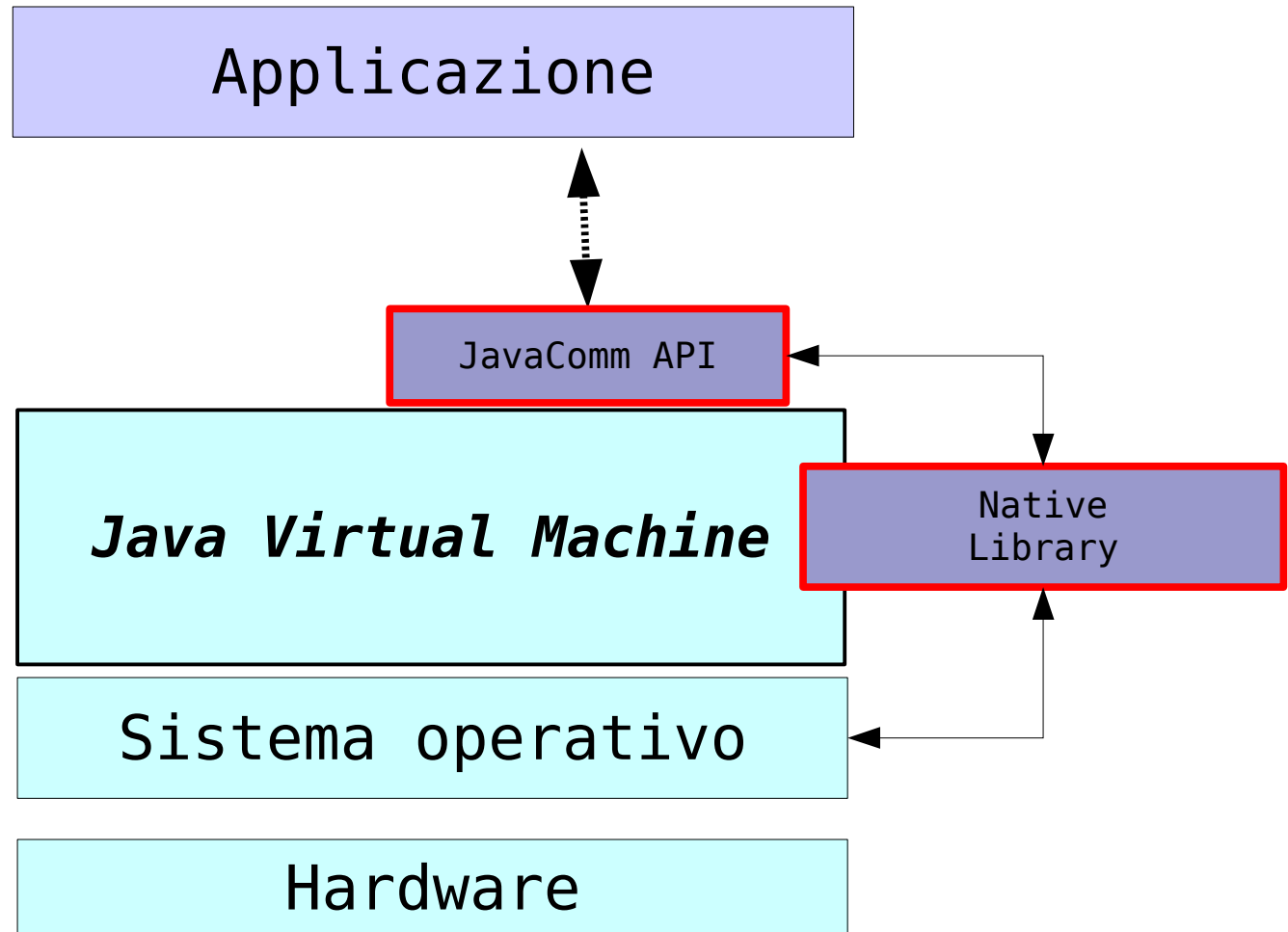


Piattaforme supportate

- Implementazione ufficiale Sun:
 - 2.X: Solaris e Windows *(e Linux???)*
 - 3.0: Solaris e Linux *(e Windows???)*
- Implementazioni di terze parti per Windows, Linux, PocketPC (Windows CE), Mac OS, SCO Open Server, BSD, HP-UX, Unixware, Digital/UNIX:
 - RXTX:
 - <http://www.rxtx.org> (open source, multipiattaforma)
 - SerialIO:
 - <http://www.serialio.com> (26 piattaforme!)



Architettura JavaComm





Installazione di JavaComm

- Una generica distribuzione è costituita da tre elementi principali:
 - Uno o più file jar contenenti il bytecode dell'API Java
 - Una o più librerie contenenti il codice nativo utilizzato dalla VM per accedere alle porte seriali **rese disponibili** dal sistema operativo
 - Un file di property contenente i riferimenti al driver da utilizzare, alla libreria nativa nonché altri parametri utili al funzionamento dell'interfaccia



Installazione su Linux

- Il pacchetto è costituito da tre file principali:
 - **comm.jar**: contiene le classi Java. Deve essere **incluso del CLASSPATH** dell'applicazione
 - **libLinuxSerialParallel.so**: è il file della libreria nativa. Deve essere copiato nella directory **/usr/lib/** oppure caricato dalla VM attraverso il parametro **-Djava.library.path** (percorso ove si trova il file)
 - **javax.comm.properties**: è il file contenente le property. Deve essere copiato nella directory lib/ del JRE o nella stessa directory contenente il file comm.jar



Installazione su Windows

- Il pacchetto (versione 2.X) è costituito da tre file:
 - **comm.jar**: contiene le classi Java. Deve essere **copiato nella directory lib\ext** del JRE
 - **comm.dll**: è la libreria a caricamento dinamico nativa. Deve essere **copiata nella directory bin** del JRE
 - **javax.comm.properties**: è il file contenente le property. Deve essere **copiato nella directory lib** del JRE



Seriali in ambiente Java ME

- E' possibile accedere alle porte seriali anche in ambiente Java ME. In particolare:
 - CDC/PersonalProfile (+ PersonalJava):
 - grazie a **JNI** è possibile utilizzare librerie di terze parti (ad esempio, Telio/SerialCE per PocketPC) con modalità analoghe alla versione standard
 - CLDC/MIDP:
 - il **Generic Connection Framework** fornisce lo schema “comm:com0;baudrate=19200...” e l'interfaccia CommConnection
 - non tutti i dispositivi supportano l'accesso alle porte seriali! Esistono estensioni per PocketPC e PalmOS



Panoramica sulla libreria

- **javax.comm**
 - **CommPortIdentifier**:
 - fornisce funzionalità ad alto livello per l'enumerazione delle porte disponibili
 - CommPort, **SerialPort**, ParallelPort:
 - forniscono una astrazione e le relative specializzazioni per l'accesso alle porte
 - CommPortOwnershipListener, **SerialPortEventListener**, ParallelPortEventListener
 - forniscono un meccanismo asincrono per la gestione dello stato delle porte e data buffering



Gestione eventi


- **CommPortOwnershipListener**
 - riceve eventi sull'assegnazione esclusiva di una porta di comunicazione.
- **SerialPortEventListener, ParallelPortEventListener**
 - ricevono eventi (SerialPortEvent, ParallelPortEvent) sui dati in arrivo nelle rispettive porte: cambio stato delle linee, presenza di dati nel buffer...), compresa l'informazione sulle transizioni di stato:

```
SerialPortEvent(SerialPort src,  
                int eventtype, boolean oldvalue,  
                boolean newvalue)
```



Tipo eventi (SerialPortEvent)

- **BI**: Break interrupt
- **CD**: Carrier detect
- **CTS**: Clear to send
- **DATA_AVAILABLE**: Data available at the serial port
- **DSR**: Data set ready
- **FE**: Framing error
- **OE**: Overrun error
- **OUTPUT_BUFFER_EMPTY**: Output buffer is empty
- **PE**: Parity error
- **RI**: Ring indicator



Permette di scrivere un driver
asincrono per la ricezione dei dati



Elencare le porte disponibili

- La classe **CommPortIdentifier** fornisce una enumerazione delle porte disponibili nel sistema:

```
Enumeration en = CommPortIdentifier.getPortIdentifiers();  
  
while (en.hasMoreElements()) {  
    CommPortIdentifier id = (CommPortIdentifier)  
                                en.nextElement();  
    System.out.println("Port found: " + id.getName());  
}
```



Aprire una porta seriale...

- La classe **SerialPort** modella una porta seriale, della quale è possibile avere controllo pressoché completo:

```
CommPortIdentifier portID =  
    CommPortIdentifier.getPortIdentifier("dev/ttyS0");
```

```
SerialPort port =  
    (SerialPort) portID.open("MyApplication", 5000);
```

Nome convenzionale della
applicazione che chiede la porta

Tempo massimo di attesa
per avere ownership della porta



Impostare i parametri di porta

- Attraverso il metodo **setSerialParams()** è possibile impostare la velocità e il formato dei dati: bit di dati, bit di stop, bit di parità
- Le velocità valide sono quelle standard: 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- Gli altri parametri sono identificati da costanti:
 - DATABITS_5, DATABITS_6, DATABITS_7, **DATABITS_8**
 - PARITY_EVEN, PARITY_MARK, **PARITY_NONE**, PARITY_ODD, PARITY_SPACE
 - **STOPBITS_1**, STOPBITS_1_5, STOPBITS_2



Applicazioni

- JavaComm permette di implementare alcune interessanti funzionalità nelle applicazioni Java:
 - Lettura posizione da GPS
 - Invio/ricezione di SMS attraverso cellulare
 - Accesso alla rubrica del cellulare
 - Interfacciamento a display LCD
 - Interfaccia multifunzione I/O



Lettura posizione da GPS

- I ricevitori GPS dotati di interfaccia RS-232, USB o Bluetooth inviano stringhe di testo contenenti le informazioni di localizzazione
- Lo standard NMEA 0183 stabilisce il formato delle *sentence* inviate al calcolatore:
 - GPRMC
 - Recommended Minimum Specific GPS/TRANSIT Data
 - GPGGA
 - Global Positioning System Fix Data
 - GPGSV
 - GPS Satellites in View





Stampa delle sentence NMEA

```
port.setSerialPortParams(4800, SerialPort.DATABITS_8,  
    SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
```

```
BufferedReader buffer = new BufferedReader(  
    new InputStreamReader(port.getInputStream()));
```

```
String sentence;
```

```
while((sentence = buffer.readLine()) != null) {  
    System.out.println(sentence);  
}
```



Un questione di parsing...

\$GPGGA,164922.982,**3859.4108,N**,**00856.1785,E**,1,**04**,3.4,**127.9**,M
,47.6,M,0.0,0000*7A

Latitudine

Longitudine

Satelliti

Altitudine

\$GPGSA,A,3,20,23,11,24,,,,,,,,,4.4,3.4,2.8*3A

\$GPGSV,3,1,10,11,83,014,42,20,59,273,44,01,52,071,47,19,36,
172,45*71

\$GPGSV,3,2,10,23,24,192,35,14,20,041,40,17,19,316,41,24,17,
299,35*7F

\$GPGSV,3,3,10,25,14,103,35,28,10,272,00*72

\$GPRMC,164922.982,A,3859.4108,N,00856.1785,E,0.068674,,2701
06,,*17



Trucchetti... [per Linux] :-)

- JavaComm è efficace per il controllo di porte seriali standard. La libreria risulta insufficiente per:
 - controllare periferiche connesse via USB e Bluetooth
 - accedere alle porte seriali su piattaforme per le quali non sia stato fatto il porting della libreria nativa
- **Come fare™?**
- Con alcuni accorgimenti è possibile avere:
 - Accesso a dispositivi Bluetooth senza JSR 82
 - Accesso a dispositivi USB senza Java-USB
 - ... accesso a seriali senza JavaComm! :-)



javax.comm.properties

- Il file di property contiene le informazioni sulle porte seriali alle quali la VM ha accesso:

...

```
# Implementation specific driver  
driver=com.sun.comm.LinuxDriver
```

```
# Paths to server-side serial port devices  
serpath0 = /dev/ttyS0  
serpath1 = /dev/ttyS1
```

...

- Se il sistema operativo esporta altre interfacce seriali... è sufficiente **aggiungerle** nella lista!



javax.comm.properties

- Aggiungiamo due porte rfcomm e una usbserial:

```
...  
# Implementation specific driver  
driver=com.sun.comm.LinuxDriver  
  
# Paths to server-side serial port devices  
serpath0 = /dev/ttyS0  
serpath1 = /dev/ttyS1  
serpath2 = /dev/rfcomm0  
serpath3 = /dev/rfcomm1  
serpath4 = /dev/ttyUSB0  
...
```



RFCOMM senza JSR 82

- Utilizzando Bluez, lo stack Bluetooth ufficiale in ambiente Linux, è possibile avere il binding di porte seriali rfcmm su specifici servizi SPP:

```
rfcomm bind 0 00:11:22:AA:BB:CC 1
```

- Sul file /etc/bluetooth/rfcomm.conf si può avere il binding automatico:

```
rfcomm0 {  
    bind yes;  
    device 00:11:22:AA:BB:CC;  
    channel 1;  
}
```



Seriali senza JavaComm (Linux)

- Nel caso non sia possibile installare JavaComm (o non si voglia installare), è possibile utilizzare le utility native del sistema operativo e sfruttare il filesystem di Unix:
 - creare il device opportuno (ttySx, rfcomm, usbserial)
 - impostare i parametri con setserial
 - accedere al device seriale come ad un normale file
- Da verificare per device non standard...



Conclusioni

- JavaComm permette di scrivere applicazioni Java che si interfacciano ad hardware specializzato: GPS, modem, controller, sensori...
- Con qualche accorgimento è possibile accedere a porte USB, IrDA e Bluetooth: in ambiente Linux, con un po' di tuning, è possibile avere accesso in modo semplice a porte USBSerial e RFCOMM
- L'altalena del supporto ai diversi sistemi operativi non facilita l'attività di porting...



Bibliografia

- JavaComm API
 - Sun Microsystems
<http://java.sun.com/products/javacomm>
- Java Cookbook
 - Ian F. Darwin
O'Reilly (ISBN: 0-596-00701-9)
- JavaComm: introduzione alla libreria; interfacciare un display LCD e un telefono cellulare
 - Stefano Sanna
DEV n. 113 e 114, Gruppo Editoriale Infomedia



JavaComm

controllare dispositivi seriali in Java

(Versione 1.1)

(C) 2006 Stefano Sanna (gerdavax@tiscali.it)

è garantito il permesso di copiare, distribuire e/o modificare questo documento seguendo i termini della Licenza per Documentazione Libera GNU, Versione 1.1 o ogni versione successiva pubblicata dalla Free Software Foundation. Una copia della licenza in lingua italiana è disponibile presso: <http://www.softwarelibero.it/gnudoc/fdl.it.html>

Realizzato in ambiente Linux con OpenOffice 2.0

Tutti i marchi commerciali sono di proprietà dei rispettivi titolari e sono stati citati in questa presentazione a solo scopo illustrativo.