Bluetooth and leJOS

Versión 0.1

Juan Antonio Breña Moral

7-jul-08

Index

Revision History

Name	Date	Reason For Changes	Version
Juan Antonio Breña Moral	05/07/2008		0.1

1.- Introduction

1.1.- Goals

Many developers around the world choose leJOS, Java for Lego Mindstorm, as the main platform to develop robots with NXT Lego Mindstorm. I consider that this eBook will help leJOS community, Lego Mindstorm community, Robot's developers and Java fans to develop better software.

Robotics will be very important for the humanity in the next 10 years and this eBook is an effort to help in this way.

Many people spend several hours in their robotics projects with problems with wires & electronics, protocols and problems with programming languages, Lego Mindstorm is easy and Java/leJOS is an excellent platform to demonstrate your software engineering skills to develop better robots. NXT Brick is the easiest way to enter in the robotics world and leJOS, the best platform in the moment to use software engineering ideas.

Enjoy, Learn, Contact with me to improve the eBook and share your ideas.

Juan Antonio Breña Moral.
www.juanantonio.info

1.1.1.- About this document

Lego Mindstorm incorporated a Bluetooth module in the new Lego Mindstorm brick, NXT Brick.

Bluetooth is a smart way to communicate with others NXT Bricks or a PC/Mac. Bluetooth is a short-range radio link intended to replace wires to connect with portable and/or fixed electronic devices. Key features are robustness, low complexity, low power and low cost.

This document explains Bluetooth technology in general and how to use this technology with your NXT Brick using leJOS.

1.2.- LeJOS Project

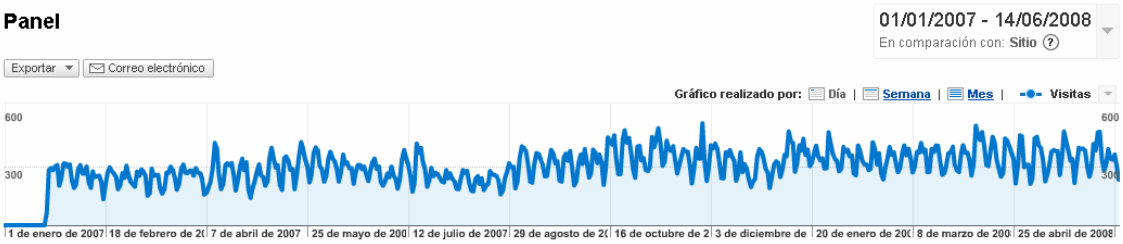
LeJOS is Sourceforge project created to develop a technological infrastructure to develop software into Lego Mindstorm Products using Java technology.

Currently leJOS has opened the following research lines:

1. NXT Technology
    a. NXJ
    b. LeJOS PC API
    c. iCommand
2. RCX Technology

a. leJOS for RCX

LeJOS project's audience has increased. Currently more than 500 people visit the website every day.



This eBook will focus in NXT technology with NXJ using a Windows Environment to develop software.

1.3.- NXT Brick

The NXT is the brain of a MINDSTORMS robot. It's an intelligent, computer-controlled LEGO brick that lets a MINDSTORMS robot come alive and perform different operations.

Motor ports
The NXT has three output ports for attaching motors - Ports A, B and C

Sensor ports
The NXT has four input ports for attaching sensors - Ports 1, 2, 3 and 4.

USB port
Connect a USB cable to the USB port and download programs from your computer to the NXT (or upload data from the robot to your computer). You can also use the wireless Bluetooth connection for uploading and downloading.

Loudspeaker
Make a program with real sounds and listen to them when you run the program

NXT Buttons
Orange button: On/Enter /Run
Light grey arrows: Used for moving left and right in the NXT menu
Dark grey button: Clear/Go back

NXT Display
Your NXT comes with many display features - see the MINDSTORMS NXT Users Guide that comes with your NXT kit for specific information on display icons and options

Technical specifications

- 32-bit ARM7 microcontroller
- 256 Kbytes FLASH, 64 Kbytes RAM
- 8-bit AVR microcontroller
- 4 Kbytes FLASH, 512 Byte RAM
- Bluetooth wireless communication (Bluetooth Class II V2.0 compliant)
- USB full speed port
- 4 input ports, 6-wire cable digital platform (One port includes a IEC 61158 Type 4/EN 50 170 compliant expansion port for future use)
- 3 output ports, 6-wire cable digital platform
- 100 x 64 pixel LCD graphical display
- Loudspeaker - 8 kHz sound quality. Sound channel with 8-bit resolution and 2-16 KHz sample rate.
- Power source: 6 AA batteries

1.3.1.- NXT Sensors used in the eBook

NXT Sensors used in the document are the following:

- NXT Motor
- Ultrasonic Sensor
- Compass Sensor
- NXTCam
- Tilt Sensor
- NXTCam
- NXTe

NXT Motor

Ultrasonic Sensor

Compass Sensor

Tilt Sensor

NXTCam

NXTe

1.4.- About the author

Juan Antonio Breña Moral has collaborated in leJOS Research team since 2006. He works in Europe leading Marketing, Engineering and IT projects for middle and large customers in several markets as Defence, Telecommunications, Pharmaceutics, Energy, Automobile, Construction, Insurance and Internet.

Further information:
www.juanantonio.info
www.esmeta.es

2.- Bluetooth Technology

2.1.- Introduction

Bluetooth technology has achieved global acceptance such that any Bluetooth enabled device, almost everywhere in the world, can connect to other Bluetooth enabled devices in proximity. Bluetooth enabled electronic devices connect and communicate wirelessly through short-range, ad hoc networks known as piconets. Each device can simultaneously communicate with up to seven other devices within a single piconet. Each device can also belong to several piconets simultaneously. Piconets are established dynamically and automatically as Bluetooth enabled devices enter and leave radio proximity.

Bluetooth technology operates in the unlicensed industrial, scientific and medical (ISM) band at 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec. The 2.4 GHz ISM band is available and unlicensed in most countries.

The operating range depends on the device class:

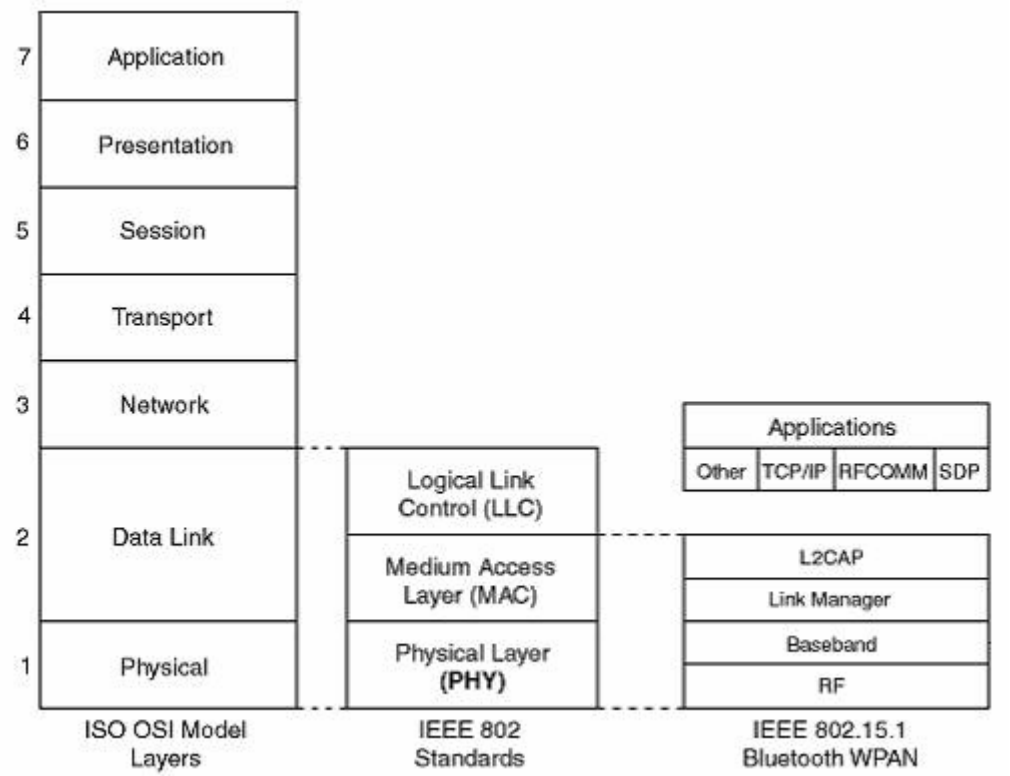
- Class 3 radios – have a range of up to 1 meter or 3 feets
- Class 2 radios – most commonly found in mobile devices – have a range of 10 meters or 33 feets
- Class 1 radios – used primarily in industrial use cases – have a range of 100 meters or 300 feets

2.2.- History

Originally invented in Scandinavia, the Bluetooth technology was named after the Danish Viking king Harold Bluetooth. However, when the technology was launched in 1998, it was very much an international initiative. A handful of leading companies within the computer and telecommunications industry formed the Bluetooth Special Interest Group (SIG). The goal was for devices from different manufacturers to be able to communicate with each other. Today, a great number of companies have joined the SIG as adopters of the Bluetooth technology, and the number is increasing all the time. The magnitude of industry involvement should ensure that Bluetooth becomes a widely adopted technology.

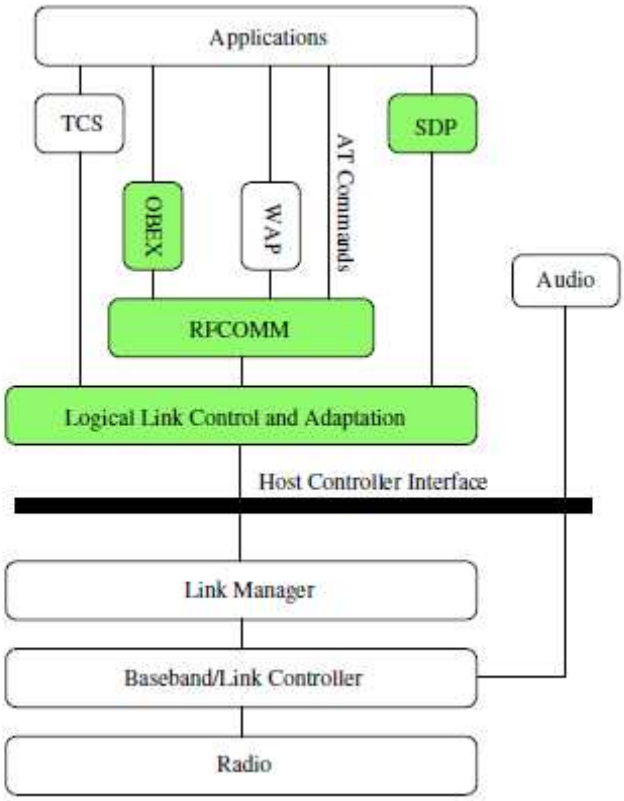
2.3.- Bluetooth Architecture

The Bluetooth architecture and its mapping to OSI model is shown below:



2.4.- Bluetooth Protocols

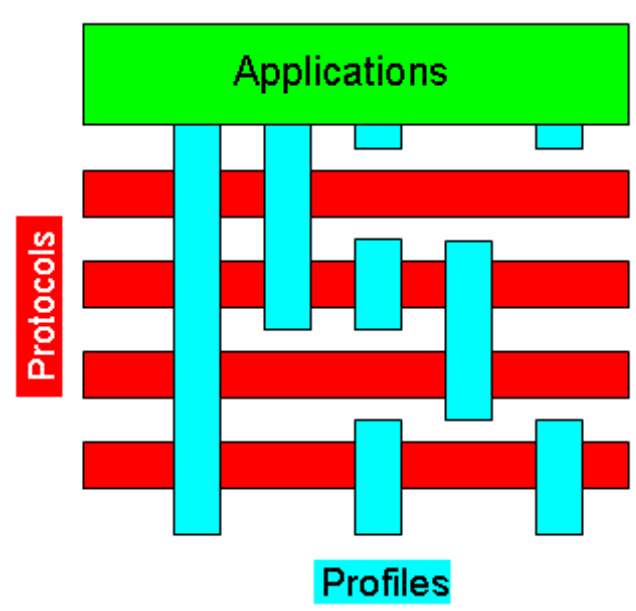
The Bluetooth protocols contain the standard procedures for connections and data exchange between Bluetooth devices.



Radio: Modulates and demodulates data for transmission and reception on air
Baseband/Link Controller: Controls the physical links via the radio, assembling packets and controlling frequency hopping
Link Manager: Responsible for security and link set-up between Bluetooth devices
Host Controller Interface (HCI): Handles communication between a separate host and a Bluetooth module (also referred to as the Bluetooth controller)
Logical Link Control and Adaptation (L2CAP): Multiplexes data from higher layers, converts between different packet sizes
RFCOMM: Emulates an RS-232 like serial interface
WAP and OBEX Adopted protocols
SDP (Service Discovery Protocol): Lets Bluetooth devices discover what services other Bluetooth devices support
OBEX: The Object Exchange Protocol (OBEX) is a specification for object data exchange.
TCS (Telephony Control Protocol Specification): Provides telephony services. It defines call control signaling for establishing speech and data calls between Bluetooth devices, providing them with telephony services.

2.5.- Bluetooth profiles

In order to use Bluetooth wireless technology, a device must be able to interpret certain Bluetooth profiles. The profiles define the possible applications. Bluetooth profiles are general behaviors through which Bluetooth enabled devices communicate with other devices. Bluetooth technology defines a wide range of profiles that describe many different types of use cases.

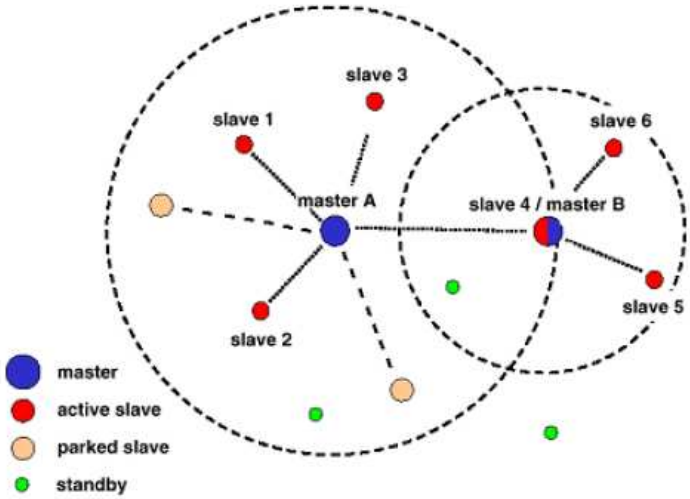


Devices do not need to implement all the profiles. A device only needs to implement the ones needed to support its applications. An exception however, is the Generic Access Profile, which is required in all devices.

2.6.- Bluetooth Networks

2.6.1.- Piconet

A piconet network is a group of Bluetooth devices joined together into a short range network by Bluetooth links. The group is synchronized to the timing and hopping sequence of the Master. Piconet is the heart of the Bluetooth technology. When a Bluetooth device has established a link to one or more other devices, a piconet has been formed. The device that initiates a connection acts as the master. The other devices are slaves. The master controls all traffic in the piconet. Communication between slaves can only take place via the master.



In any piconet network there can only be one master. Furthermore, up to seven slaves can be active. However, there can be additional slaves which are not active but remain synchronized to the piconet. Such slaves are referred to as parked. A parked device can very quickly become active and begin communicating in the piconet. By swapping active and parked slaves, you can increase the number of slaves virtually connected to the piconet from seven to 255 devices.

The master is the central entity that decides transmit/receive resource allocation to different slaves and thus controls bandwidth usage among slaves

The Slaves in a piconet only have links to the Master; there are no direct links between Slaves in a piconet:

- Master may become bottleneck for data transfer
- One piconet can be split into two piconets by one Slave becoming

A Slave have three power saving modes for idle devices. In order of power savings these modes are: PARK, HOLD, SNIFF.

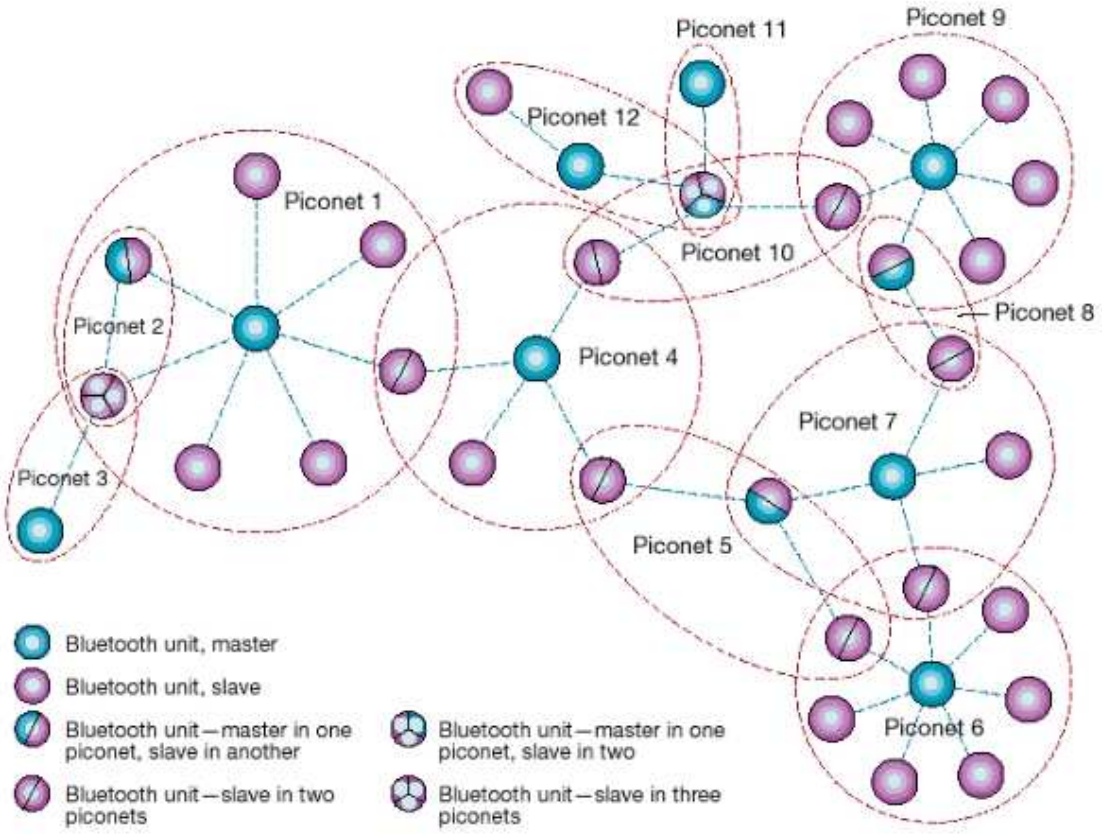
The master can command slaves to go quite and then wake them up. In park these devices need only about 2 mW to stay operational. This makes them well suited for battery operations.

A Sniff mode, the slave listens within its Piconet at a reduced scan rate (programmable).

A Park mode, the slave is still synchronized within the network, but doesn't participate in data transmission. Devices in this mode only occasionally listen to maintain synchronization and for a wake up call. This is the maximum power saving mode.
A Hold mode, only an internal timer circuit keeps on working. Slaves can request the master to allow them to go into hold mode.

2.6.2.- Scatternet

A Scatternet is a group of Bluetooth piconets joined together by devices that are in more than one piconet.



Scatternet network consist on two piconets networks with a different power class devices.

2.7.- Bluetooth Connections

Two types of connections are possible: Synchronous Connection Orientated (SCO), or Asynchronous Connection Less (ACL).

The SCO (Synchronous Connection Oriented) channels are used for voice transfer by reserving slots, they are symmetric between the master and a slave. There is a maximum of three in a piconet, a slave being able to control two originating from different masters. One can use the voice packets or the mixed voice/data packets.

An ACL (Asynchronous Connection Less) channel supplies an asynchronous access between master and slave (a single channel per couple), with the slot as base. The data packets only are used. In addition, a slave can only transmit after having received a packet from the master (the following slot). For this purpose, the master can send polling packets to the slaves (when there is nothing more asynchronous to transmit).

3.- How to use Bluetooth with leJOS

3.1.- Introduction

Once you have understood Bluetooth technology, this document explains how to use LeJOS technology to manage Bluetooth connections and exchange data with others Bluetooth Device.

LeJOS packages which you have to use when you have to manage Bluetooth communications are:

- java.io.*
- Javax.bluetooth.*
- lejos.nxt.comm.*
- lejos.nxt.remote.*
- lejos.devices.*

The steps to connect and manage a Bluetooth device are the following:

1. Discovery Bluetooth Device
2. Connect with a Bluetooth device
3. Exchange Data between a NXT Brick with another one or a Bluetooth device
4. Listen a Bluetooth connection

3.1.1.- Discovery Bluetooth Device

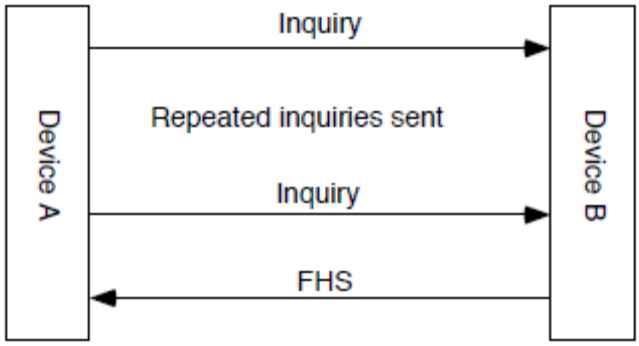
If you want to connect with another NXT brick or another Bluetooth Device, it is necessary to know 2 things:

1. What device do you want to connect?
2. Is available that device to connect?

NXT Brick has a memory where store information about Bluetooth devices. You can use the static method: **Bluetooth.getKnownDevicesList**(); to extract values from NXT brick.

```
      //Extract data from NXT brick
      Vector curList = Bluetooth.getKnownDevicesList();
      if (curList == null)
      {
            RConsole.println("getKnownDeviceList returns null\n");
            return false;
      }else{
            //Show remote devices stored on NXT brick
            for(int i = 0; i < curList.size(); i++){
                  RConsole.println("Current      device      "      +
((RemoteDevice)curList.elementAt(i)).getFriendlyName(false) + "\n");
            }
      }
```

In case of your NXT brick doesn't have any data about your environment, it is necessary to make an inquire using **Bluetooth.inquire()**



This piece of code, explains the concept:

```
byte[] cod = {0,0,0,0}; // Any
RConsole.println("Searching...\n");
Vector devList = Bluetooth.inquire(5, 10,cod);
if (devList == null)
{
      RConsole.println("Inquire returns null\n");
      return false;
}
if (devList.size() > 0) {
      String[] names = new String[devList.size()];
      for (int i = 0; i < devList.size(); i++) {
            RemoteDevice       btrd       =       ((RemoteDevice)
devList.elementAt(i));
            names[i] = btrd.getFriendlyName(false);
            RConsole.println("Got device " + names[i] + "\n");
      }
}
```

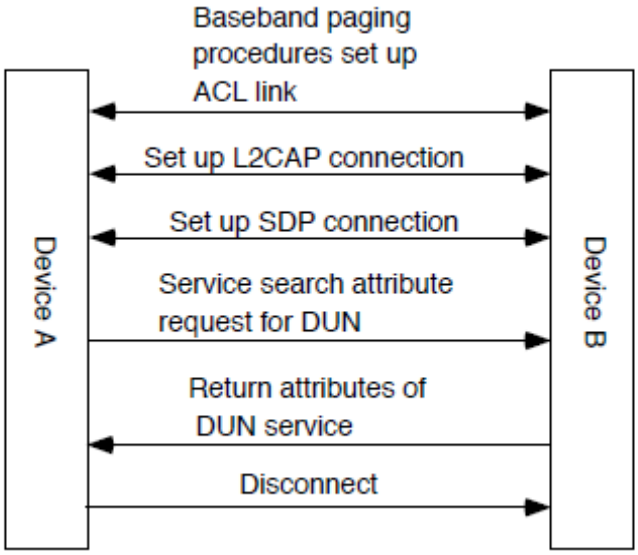
Once you discover others Bluetooth devices, it is necessary to add into NXT brick:

```
for(int i = 0; i < devList.size(); i++){
      RemoteDevice btrd = ((RemoteDevice) devList.elementAt(i));
      Bluetooth.addDevice(btrd);
}
```

Conclusions
See your list about known devices then if you don't have any device then search your environment to detect new devices. Using inquire methods. Remember that if you want to connect with them, then you need to add into your list.

3.1.2.- Connect with a Bluetooth device

Once NXT brick has added NXT brick or Bluetooth device, the connection it is very easy.



The first step is create a **RemoteDevice** object using the name of the target:

```
RemoteDevice btrd = Bluetooth.getKnownDevice("NXT");
if (btrd == null) {
      return false;
}
```

Once you have a **RemoteDevice** Object then connect with it:

```
BTConnection btc = Bluetooth.connect(btrd);
if (btc == null) {
      return false;
}
byte [] status = Bluetooth.getConnectionStatus();
if (status == null)
{
      btc.close();
      return false;
}
```

If the Bluetooth device has a pin then when have to connect, indicate it:

```
      final byte[] pin = {(byte) '0', (byte) '0', (byte) '0', (byte)
'0'};
      BTConnection btGPS = null;
      btGPS = Bluetooth.connect(btrd.getDeviceAddr(),
NXTConnection.RAW, pin);
```

3.1.3.- Exchange Data between a NXT Brick with another one or a Bluetooth device

To exchage data between a NXT Device with another, exist 2 ways:

- Using setIOMode
- Using DataInputStream

The first way **setIOMode** to **NXTConnection.RAW**. This piece of code explain the idea:

```
btc.setIOMode(NXTConnection.RAW);
for(int i = 0; i < 100; i++)
{
      byte [] b = strToByte("Hello world " + i + "\r\n");
      if (btc != null) btc.write(b, b.length);
      try{Thread.sleep(delay);}catch(Exception e){}
}
if (btc != null) btc.close();
```

Another example using this way:

```
btc.setIOMode(NXTConnection.RAW);
byte []inBuf = new byte[255];
while (true)
{
      int cnt;
      if (userp)
            cnt = bt.readPacket(inBuf, inBuf.length);
      else
            cnt = bt.read(inBuf, inBuf.length);
      if (cnt == 0) continue;
      if (cnt == -2)
      {
            RConsole.println("Lost data, resync\n");
            bt.read(null, 256);
            continue;
      }
      int val = ((int)inBuf[2] & 0xff) + (((int)inBuf[3] & 0xff)
<< 8);
      RConsole.println("Read len " + cnt  +" val " + val + "\n");
      if (val == 0xffff) break;
      byte [] b = strToByte("Hello from bt " + val + "\r\n");
      if (btc != null) btc.write(b, b.length);
}
if (btc != null) btc.close();
```

The second way to manage a Bluetooth connection, is the following:

Understand this piece of code:

```
InputStream in = null;
in = btGPS.openInputStream();
try {
      in.read(segment);
} catch (IOException e) {
      // How to handle error?
}
```

This second example is used for bidirectional communications:

```
DataInputStream dis = btc.openDataInputStream();
DataOutputStream dos = btc.openDataOutputStream();

for(int i=0;i<100;i++) {
      try {
            LCD.drawInt(i*30000, 8, 0, 2);
            LCD.refresh();
```

```
                dos.writeInt(i*30000);
                dos.flush();
        } catch (IOException ioe) {
                LCD.drawString("Write Exception", 0, 0);
                LCD.refresh();
        }

        try {
                LCD.drawInt(dis.readInt(),8, 0,3);
                LCD.refresh();
        } catch (IOException ioe) {
                LCD.drawString("Read Exception ", 0, 0);
                LCD.refresh();
        }
}

try {
        LCD.drawString("Closing...    ", 0, 0);
        LCD.refresh();
        dis.close();
        dos.close();
        btc.close();
} catch (IOException ioe) {
        LCD.drawString("Close Exception", 0, 0);
        LCD.refresh();
}
```

3.1.4.- Listen a Bluetooth connection

If your goal is connect with another NXT brick then the second NXT brick should listen a Bluetooth connection to exchange data or receive commands.

To listen a Bluetooth connection use: **Bluetooth.waitForConnection()**

```
BTConnection btc = Bluetooth.waitForConnection();
```

Once you have connected with a NXT brick sender then exchange data using this way:

```
DataInputStream dis = btc.openDataInputStream();
DataOutputStream dos = btc.openDataOutputStream();

for(int i=0;i<100;i++) {
        int n = dis.readInt();
        LCD.drawInt(n,7,0,1);
        LCD.refresh();
        dos.writeInt(-n);
        dos.flush();
}

dis.close();
dos.close();
Thread.sleep(100); // wait for data to drain
LCD.clear();
LCD.drawString(closing,0,0);
LCD.refresh();
btc.close();
```

4.- LeJOS examples using Bluetooth

4.1.- Introduction

The following examples have been extracted from current leJOS NXJ release.

- Bluetooth Concepts
  - BTConnectTest.java
  - BTReceive.java
  - Bttest.java
  - Bluestats.java
  - SignalTest.java
  - StartUpText.java
- Connection with a GPS
  - BTGPS.java
  - GPS.java
- Connection with a BT Keyboard
  - KeyboardTest.java
  - Keyboard.java
  - KeyListener.java
  - KeyEvent.java

This source code is very useful to understand the concepts about Bluetooth technology and leJOS.

4.2.- Bluetooth concepts

4.2.1.- BTConnectTest.java, Connect with another NXT Brick

This example shows how to connect with another NXT brick. The Bluetooth communication is bidirectional.

```
import java.io.IOException;
import lejos.nxt.*;
import lejos.nxt.comm.*;
import java.io.*;
import javax.bluetooth.*;

/**
 *
 * Test of NXT to NXT Bluetooth comms.
 *
 * Connects to another NXT, sends 100 ints, and receives the
 * replies. Then closes the connection and shuts down.
 *
 * Works with the BTReceive sample running on the slave NXT.
 *
 * Change the name string to the name of your slave NXT, and make sure
 * it is in the known devices list of the master NXT. To do this, turn
 * on the slave NXT and make sure Bluetooth is on and the device
 * is visible. Use the Bluetooth menu on the slave for this. Then,
 * on the master, select the Bluetooth menu and then select Search.
 * The name of the slave NXT should appear. Select Add to add
```

```
 * it to the known devices of the master. You can check this has
 * been done by selecting Devices from the Bluetooth menu on the
 * master.
 *
 * @author Lawrie Griffiths
 *
 */
public class BTConnectTest {
      public static void main(String[] args) throws Exception {
            String name = "NXT";

            LCD.drawString("Connecting...", 0, 0);
            LCD.refresh();

            RemoteDevice btrd = Bluetooth.getKnownDevice(name);

            if (btrd == null) {
                  LCD.clear();
                  LCD.drawString("No such device", 0, 0);
                  LCD.refresh();
                  Thread.sleep(2000);
                  System.exit(1);
            }

            BTConnection btc = Bluetooth.connect(btrd);

            if (btc == null) {
                  LCD.clear();
                  LCD.drawString("Connect fail", 0, 0);
                  LCD.refresh();
                  Thread.sleep(2000);
                  System.exit(1);
            }

            LCD.clear();
            LCD.drawString("Connected", 0, 0);
            LCD.refresh();

            DataInputStream dis = btc.openDataInputStream();
            DataOutputStream dos = btc.openDataOutputStream();

            for(int i=0;i<100;i++) {
                  try {
                        LCD.drawInt(i*30000, 8, 0, 2);
                        LCD.refresh();
                        dos.writeInt(i*30000);
                        dos.flush();
                  } catch (IOException ioe) {
                        LCD.drawString("Write Exception", 0, 0);
                        LCD.refresh();
                  }

                  try {
                        LCD.drawInt(dis.readInt(),8, 0,3);
                        LCD.refresh();
                  } catch (IOException ioe) {
                        LCD.drawString("Read Exception ", 0, 0);
                        LCD.refresh();
                  }
            }
```

```
            try {
                  LCD.drawString("Closing...    ", 0, 0);
                  LCD.refresh();
                  dis.close();
                  dos.close();
                  btc.close();
            } catch (IOException ioe) {
                  LCD.drawString("Close Exception", 0, 0);
                  LCD.refresh();
            }

            LCD.clear();
            LCD.drawString("Finished",3, 4);
            LCD.refresh();
            Thread.sleep(2000);
      }
}
```

4.2.2.- BTReceive.java, Receive a Bluetooth connection from another NXT Brick

This example shows how to receive a Bluetooth connection.

```
import lejos.nxt.*;
import lejos.nxt.comm.*;
import java.io.*;

/**
 * Receive data from another NXT, a PC, a phone,
 * or another bluetooth device.
 *
 * Waits for a connection, receives an int and returns
 * its negative as a reply, 100 times, and then closes
 * the connection, and waits for a new one.
 *
 * @author Lawrie Griffiths
 *
 */
public class BTReceive {

      public static void main(String [] args)  throws Exception
      {
            String connected = "Connected";
        String waiting = "Waiting...";
        String closing = "Closing...";

            while (true)
            {
                  LCD.drawString(waiting,0,0);
                  LCD.refresh();

              BTConnection btc = Bluetooth.waitForConnection();

                  LCD.clear();
                  LCD.drawString(connected,0,0);
                  LCD.refresh();

                  DataInputStream dis = btc.openDataInputStream();
                  DataOutputStream dos = btc.openDataOutputStream();
```

```
                for(int i=0;i<100;i++) {
                      int n = dis.readInt();
                      LCD.drawInt(n,7,0,1);
                      LCD.refresh();
                      dos.writeInt(-n);
                      dos.flush();
                }

                dis.close();
                dos.close();
                Thread.sleep(100); // wait for data to drain
                LCD.clear();
                LCD.drawString(closing,0,0);
                LCD.refresh();
                btc.close();
                LCD.clear();
          }
     }
}
```

4.2.3.- Bttest.java, Program to test Bluetooth technology with leJOS

This example shows many Bluetooth features with leJOS.

```
import java.io.*;
import java.util.Vector;
import lejos.nxt.comm.*;
import lejos.nxt.*;
import javax.bluetooth.*;

/*
 * Bluetooth test code.
 * Tests developed alongside the new Bluetooth implementation.
Included as an
 * indication how these classes may be tested. Note that some test
cases
 * use the names of target Bluetooth devices and may need changing.
Also
 * some tests require the use of a PC based test program (to be be
found in
 * BTTestPC).
 */

public class bttest {
     static byte [] strToByte(String s)
     {
          byte [] b = new byte[s.length()];
          for(int i = 0; i < s.length(); i++)
          {
               char c = s.charAt(i);
               //dOut.writeByte((byte)c);
               b[i] = (byte)c;
          }
          return b;
     }
```

```
	public static boolean badSwitchTest() throws Exception
	{
		// Demonstrates the data loss that switching out of stream
mode and into
		// command mode when input is being sent from a remote
device. The test
		// program should be used to send data in fast mode to show
this problem.
		byte []outBuf = new byte[255];
		byte []inBuf = new byte[255];
		int buflen;
		RConsole.println("Bad switch test, expect lost data\n");
		RConsole.println("Wait for connection...");
		BTConnection bt = Bluetooth.waitForConnection();
		if (bt == null)
		{
			RConsole.println("Did not get connection\n");
			return false;
		}
		RConsole.println("After connect\n");
		while (bt.readPacket(inBuf, inBuf.length) == 0) ;
		outBuf[0] = 4;
		outBuf[1] = (byte)131;
		outBuf[2] = 4;

		RConsole.println("signal strength " +
bt.getSignalStrength() + "\n");

		while (true)
		{
			//int cnt = bt.readPacket(inBuf, inBuf.length);
			int cnt = bt.read(inBuf, inBuf.length);
			if (cnt == 0) continue;
			if (cnt == -2)
			{
				RConsole.println("Lost data, resync\n");
				bt.read(null, 256);
				continue;
			}
			int val = ((int)inBuf[2] & 0xff) + (((int)inBuf[3] &
0xff) << 8);
			RConsole.println("Read len " + cnt  +" val " + val +
"\n");
			if (val == 0xffff) break;
			if (val == 27) RConsole.println("signal strength " +
bt.getSignalStrength() + "\n");
		}
		if (bt != null) bt.close();
		return true;
	}

	public static boolean ioTest() throws Exception
	{
		// Test various types of I/O with the remote PC program.
Run this test
		// and then...
		// 1. Connect using the test program. The test will send
data to the PC.
		// 2. Use the fast or slow send buttons to send data to the
NXT for
		//    testing the input streams.
```

```
            byte []outBuf = new byte[255];
            byte []inBuf = new byte[255];
            int buflen;
            RConsole.println("ioTest\n");
            RConsole.println("Wait for connection...\n");
            BTConnection bt = Bluetooth.waitForConnection();
            if (bt == null)
            {
                  RConsole.println("Did not get a connection\n");
                  return false;
            }
            RConsole.println("After connect\n");
            int start = (int)System.currentTimeMillis();

            // First test low level writes
            outBuf[0] = 4;
            outBuf[1] = (byte)131;
            outBuf[2] = 4;

            RConsole.println("signal strength " +
bt.getSignalStrength() + "\n");

            RConsole.println("Testing write\n");
            for(int i=0; i < 1000; i++)
            {
                  outBuf[3] = (byte)(i & 0xff);
                  outBuf[4] = (byte) ((i >> 8) & 0xff);
                  bt.write(outBuf, 133);
            }
            RConsole.println("write complete time " + ((int)
System.currentTimeMillis() - start) + "\n");
            RConsole.println("Testing output stream");
            // Now do the same thing using a higher level stream
            OutputStream os = bt.openOutputStream();
            for(int i=0; i < 100; i++)
            {
                  os.write(4);
                  os.write(3);
                  os.write(4);
                  os.write(i & 0xff);
                  os.write((i>>8) & 0xff);
                  os.flush();
            }
            RConsole.println("signal strength " +
bt.getSignalStrength() + "\n");
            RConsole.println("Testing read\n");
            while (true)
            {
                  //int cnt = bt.readPacket(inBuf, inBuf.length);
                  int cnt = bt.read(inBuf, inBuf.length);
                  if (cnt == 0) continue;
                  if (cnt == -2)
                  {
                        RConsole.println("Lost data, resync\n");
                        bt.read(null, 256);
                        continue;
                  }
                  int val = ((int)inBuf[2] & 0xff) + (((int)inBuf[3] &
0xff) << 8);
                              //RConsole.println("Memory " +
(int)Runtime.getRuntime().freeMemory() + "\n");
```

```
                  RConsole.println("Read len " + cnt  +" val " + val +
"\n");
                  if (val == 0xffff) break;
            }
            RConsole.println("Testing input stream\n");
            // Now test input
            InputStream is = bt.openInputStream();
            while (true)
            {
                  //int cnt = bt.readPacket(inBuf, inBuf.length);
                  int cnt = is.read(inBuf, 0, 18);
                  if (cnt == 0) continue;
                  int val = ((int)inBuf[2] & 0xff) + (((int)inBuf[3] &
0xff) << 8);
                  RConsole.println("Read len " + cnt  +" val " + val +
"\n");
                  if (val == 0xffff) break;
            }
            if (bt != null) bt.close();
            return true;
      }


      public static boolean powerOffTest(int delay) throws Exception
      {
            // Test that the NXT Bluetooth module can be powered off.
Also check
            // that the LCD continues to operate.
            RConsole.println("Power off test\n");
            Bluetooth.setPower(false);
            try{Thread.sleep(1000);}catch(Exception e){}
            LCD.drawString("BT Off...", 0, 0);
            LCD.refresh();
            RConsole.println("Power now off\n");
            try{Thread.sleep(delay);}catch(Exception e){}
            LCD.drawString("BT still Off...", 0, 0);
            LCD.refresh();
            Bluetooth.setPower(true);
            LCD.drawString("BT now on...", 0, 0);
            LCD.refresh();
            RConsole.println("Power on\n");
            try{Thread.sleep(5000);}catch(Exception e){}
            byte [] ver = Bluetooth.getVersion();
            if (ver == null)
            {
                  RConsole.println("Failed to get version\n");
                  return false;
            }
            RConsole.println("Version major " + ver[0] + " minor " +
ver[1] + "\n");
            return true;
      }

      public static boolean singleConnectTest(String name, int delay)
throws Exception
      {
            // Create a single outbound connection to a device using
the serial
            // profile. I tested this by simply running hyper-terminal
using one
```

```
            // of the widcomm com ports. Test can be run with by slow
and fast data
            // rates.
            RConsole.println("Single connect test delay " + delay +
"\n");
            RemoteDevice btrd = Bluetooth.getKnownDevice(name);
            if (btrd == null) {
                  RConsole.println("No such device " + name + "\n");
                  return false;
            }

            BTConnection btc = Bluetooth.connect(btrd);
            RConsole.println("After connect\n");
            if (btc == null) {
                  RConsole.println("Connect failed\n");
                  return false;
            }
            byte [] status = Bluetooth.getConnectionStatus();
            if (status == null)
            {
                  RConsole.println("Failed to get connection
status\n");
                  btc.close();
                  return false;
            }
            for(int i = 0; i < status.length; i++)
                  RConsole.println("Handle " + i + " status " +
status[i] + "\n");
            btc.setIOMode(NXTConnection.RAW);
            for(int i = 0; i < 100; i++)
            {
                  byte [] b = strToByte("Hello world " + i + "\r\n");
                  if (btc != null) btc.write(b, b.length);
                  try{Thread.sleep(delay);}catch(Exception e){}
            }

            if (btc != null) btc.close();
            RConsole.println("All closed\n");
            return true;
      }


      public static boolean multiConnectTest(String name, String
name2, boolean userp) throws Exception
      {
            // Test multiple connections...
            // Wait for a packet based connect
            // open stream connections to two systems
            // Read packets from first connect
            // Send text messages to the other two connections

            try{Thread.sleep(1000);}catch(Exception e){}
            LCD.drawString("Running...", 0, 0);
            LCD.refresh();

            RConsole.println("Multi connect test readPacket " + userp +
"\n");
            RConsole.println("Wait for connection...");
            BTConnection bt = Bluetooth.waitForConnection();
            if (bt == null)
            {
```

```
                RConsole.println("Wait for connection failed\n");
                return false;
            }
            RConsole.println("Connected\n");

            RemoteDevice btrd = Bluetooth.getKnownDevice(name);
            if (btrd == null) {
                RConsole.println("No such device " + name + "\n");
                bt.close();
                return false;
            }

            BTConnection btc = Bluetooth.connect(btrd);
            RConsole.println("After connect\n");
            if (btc == null) {
                RConsole.println("Connect failed\n");
                bt.close();
                return false;
            }

            btrd = Bluetooth.getKnownDevice(name2);
            if (btrd == null) {
                RConsole.println("No such device " + name2 +"\n");
                bt.close();
                btc.close();
                return false;
            }

            BTConnection btc2 = Bluetooth.connect(btrd);

            if (btc2 == null) {
                RConsole.println("Connect fail\n");
                bt.close();
                btc.close();
                return false;
            }

            btc.setIOMode(NXTConnection.RAW);
            btc2.setIOMode(NXTConnection.RAW);
            byte []inBuf = new byte[255];
            while (true)
            {
                int cnt;
                if (userp)
                    cnt = bt.readPacket(inBuf, inBuf.length);
                else
                    cnt = bt.read(inBuf, inBuf.length);
                if (cnt == 0) continue;
                if (cnt == -2)
                {
                    RConsole.println("Lost data, resync\n");
                    bt.read(null, 256);
                    continue;
                }
                int val = ((int)inBuf[2] & 0xff) + (((int)inBuf[3] &
0xff) << 8);
                RConsole.println("Read len " + cnt  +" val " + val +
"\n");
                if (val == 0xffff) break;
                byte [] b = strToByte("Hello from bt " + val +
"\r\n");
```

```
                if (btc != null) btc.write(b, b.length);
                if (btc2 != null) btc2.write(b, b.length);
            }
            if (btc != null) btc.close();
            if (btc2 != null) btc2.close();
            if (bt != null) bt.close();
            RConsole.println("All closed\n");
            return true;
      }

      public static boolean listenTest() throws Exception
      {
            // Test the various states of listening for a connection.
            byte []outBuf = new byte[255];
            byte []inBuf = new byte[255];
            int buflen;
            RConsole.println("listenTest\n");
            Bluetooth.getVersion();
            Bluetooth.closePort();
            RConsole.println("Connecting now should fail\n");
            try{Thread.sleep(20000);}catch(Exception e) {}
            RConsole.println("Wait for connection...\n");
            BTConnection bt = Bluetooth.waitForConnection();
            if (bt == null)
            {
                  RConsole.println("Did not get a connection\n");
                  return false;
            }
            RConsole.println("After connect\n");
            if (bt != null) bt.close();
            return true;
      }

      public static boolean miscTest() throws Exception
      {
            // Test various Bluetooth functions.
            byte [] ver = Bluetooth.getVersion();
            if (ver == null)
            {
                  RConsole.println("Failed to get version\n");
                  return false;
            }
            RConsole.println("Version major " + ver[0] + " minor " +
ver[1] + "\n");
            int vis = Bluetooth.getVisibility();
            if (vis < 0)
            {
                  RConsole.println("Failed to get visibility\n");
                  return false;
            }
            RConsole.println("Current visibility is " + vis + "\n");
            Bluetooth.setVisibility((byte)0);
            if (Bluetooth.getVisibility() != 0)
            {
                  RConsole.println("Failed to turn off visibility\n");
                  return false;
            }
            RConsole.println("NXT now not visibile\n");
            try{Thread.sleep(20000);}catch(Exception e) {}
            Bluetooth.setVisibility((byte)1);
            if (Bluetooth.getVisibility() != 1)
```

```
            {
                  RConsole.println("Failed to turn off visibility\n");
                  return false;
            }
            RConsole.println("NXT now visibile\n");
            try{Thread.sleep(20000);}catch(Exception e) {}
            Bluetooth.setVisibility((byte)vis);
            int stat = Bluetooth.getStatus();
            if (stat < 0)
            {
                  RConsole.println("Failed to read status\n");
                  return false;
            }
            RConsole.println("Current status is " + stat + "\n");
            Bluetooth.setStatus(2742);
            int val = Bluetooth.getStatus();
            if (val != 2742)
            {
                  RConsole.println("Failed to set/get status val is " +
val + "\n");
                  return false;
            }
            Bluetooth.setStatus(stat);
            RConsole.println("getStatus OK\n");
            int port = Bluetooth.getPortOpen();
            if (port < 0)
            {
                  RConsole.println("Failed to read port state\n");
                  return false;
            }
            Bluetooth.openPort();
            if (Bluetooth.getPortOpen() != 1)
            {
                  RConsole.println("getPortOpen failed\n");
                  return false;
            }
            Bluetooth.closePort();
            if (Bluetooth.getPortOpen() != 0)
            {
                  RConsole.println("getPortOpen failed\n");
                  return false;
            }
            if (port == 1) Bluetooth.openPort();
            RConsole.println("getPortOpen OK\n");
            int mode=Bluetooth.getOperatingMode();
            if (mode < 0)
            {
                  RConsole.println("Failed to read operating mode\n");
                  return false;
            }
            RConsole.println("Current mode is " + mode + "\n");
            Bluetooth.setOperatingMode((byte)0);
            if (Bluetooth.getOperatingMode() != 0)
            {
                  RConsole.println("Failed to set/get mode\n");
                  return false;
            }
            Bluetooth.setOperatingMode((byte)1);
            if (Bluetooth.getOperatingMode() != 1)
            {
                  RConsole.println("Failed to set/get mode\n");
```

```
                return false;
            }
            Bluetooth.setOperatingMode((byte)mode);
            RConsole.println("Get/Set operating mode OK\n");
            return true;
      }

      public static boolean deviceTest() throws Exception
      {
            // Test the device discovery, addition and removal
functions.
            RConsole.println("deviceTest\n");
            Vector curList = Bluetooth.getKnownDevicesList();
            if (curList == null)
            {
                  RConsole.println("getKnownDeviceList returns
null\n");
                  return false;
            }
            for(int i = 0; i < curList.size(); i++)
                  RConsole.println("Current device " +
((RemoteDevice)curList.elementAt(i)).getFriendlyName(false) + "\n");
            RConsole.println("Delete all\n");
            for(int i = 0; i < curList.size(); i++)

      Bluetooth.removeDevice((RemoteDevice)curList.elementAt(i));
            Vector newList = Bluetooth.getKnownDevicesList();
            if (newList == null)
            {
                  RConsole.println("getKnownDeviceList returns
null\n");
                  return false;
            }
            for(int i = 0; i < newList.size(); i++)
                  RConsole.println("Current device " +
((RemoteDevice)newList.elementAt(i)).getFriendlyName(false) + "\n");

            byte[] cod = {0,0,0,0}; // Any
            RConsole.println("Searching...\n");
            Vector devList = Bluetooth.inquire(5, 10,cod);
            if (devList == null)
            {
                  RConsole.println("Inquire returns null\n");
                  return false;
            }
            if (devList.size() > 0) {
                  String[] names = new String[devList.size()];
                  for (int i = 0; i < devList.size(); i++) {
                        RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
                        names[i] = btrd.getFriendlyName(false);
                        RConsole.println("Got device " + names[i] +
"\n");
                  }
            }
            RConsole.println("Add all\n");
            for(int i = 0; i < devList.size(); i++)
                  if
(!Bluetooth.addDevice((RemoteDevice)devList.elementAt(i)))
                  {
```

```
                        RConsole.println("Failed to add device " +
((RemoteDevice)devList.elementAt(i)).getFriendlyName(false) + "\n");
                        return false;
                  }
            newList = Bluetooth.getKnownDevicesList();
            if (newList == null)
            {
                  RConsole.println("getKnownDeviceList returns
null\n");
                  return false;
            }
            for(int i = 0; i < newList.size(); i++)
                  RConsole.println("Current device " +
((RemoteDevice)newList.elementAt(i)).getFriendlyName(false) + "\n");
            byte[] name = Bluetooth.getFriendlyName();
            byte[] saved = name;
            char [] cName = new char[name.length];
            int cNameLen = 0;
            for(int i = 0; i < name.length && name[i] != 0; i++)
                        cName[cNameLen++] = (char)name[i];
            String sName = new String(cName, 0, cNameLen);
            RConsole.println("Friendly name is " + sName + "\n");
            byte [] newName = {(byte)'T', (byte)'E', (byte)'S',
(byte)'T'};
            byte [] newNamePad = new byte[16];
            System.arraycopy(newName, 0, newNamePad, 0,
newName.length);
            Bluetooth.setFriendlyName(newNamePad);
            name = Bluetooth.getFriendlyName();
            cNameLen = 0;
            for(int i = 0; i < name.length && name[i] != 0; i++)
                        cName[cNameLen++] = (char)name[i];
            sName = new String(cName, 0, cNameLen);
            RConsole.println("New friendly name is " + sName + "\n");

            Bluetooth.setFriendlyName(saved);
            name = Bluetooth.getFriendlyName();
            cNameLen = 0;
            for(int i = 0; i < name.length && name[i] != 0; i++)
                        cName[cNameLen++] = (char)name[i];
            sName = new String(cName, 0, cNameLen);
            RConsole.println("Reset friendly name is " + sName + "\n");
            byte [] addr = Bluetooth.getLocalAddress();
            RConsole.println("Local address is");
            for(int i = 0; i < addr.length; i++)
                  RConsole.println(" " + addr[i]);
            RConsole.println("\n");
            return true;
      }


      public static void main(String[] args) throws Exception
      {
            int testCnt = 0;
            int passCnt = 0;
            RConsole.open();
            RConsole.println("Hello from bt test\n");
            testCnt++; if (powerOffTest(10000)) passCnt++;
            testCnt++; if (deviceTest()) passCnt++;
            testCnt++; if (miscTest()) passCnt++;
            testCnt++; if (listenTest()) passCnt++;
```

```
            testCnt++; if (ioTest()) passCnt++;
            testCnt++; if (singleConnectTest("EEYOREII", 1000))
passCnt++;
            testCnt++; if (singleConnectTest("EEYOREII", 1)) passCnt++;
            testCnt++; if (multiConnectTest("EEYORE", "EEYOREII",
false)) passCnt++;
            testCnt++; if (multiConnectTest("EEYORE", "EEYOREII",
true)) passCnt++;
            testCnt++; if (badSwitchTest()) passCnt++;
            RConsole.println("Tests complete. Tested " + testCnt + "
passed " + passCnt + "\n");
            try{Thread.sleep(5000);} catch(Exception e){}
            RConsole.close();
      }
}
```

4.2.4.- Bluestats.java, Another example to test others Bluetooth features.

This example tests others Bluetooth features.

```
import lejos.nxt.*;
import lejos.nxt.comm.*;

/**
 * BlueStats: display local device Bluetooth information.
 *
 * @author Lawrie Griffiths
 *
 */
public class BlueStats {
      public static void main(String[] args) {
            String versionString = "BC4 version ";
            String nameString = "Name";
            String visString = "Visibility";
            String statusString = "Status";
            String portString = "Port Open";
            String opString = "Op Mode";
            String addrString = "Adr";
            String connsString = "Conns";

            while(!Button.ESCAPE.isPressed()) {
                  byte[] name = Bluetooth.getFriendlyName();
                  byte[] version = Bluetooth.getVersion();
                  String fn = byteArrayToString(name);
                  byte [] connections =
Bluetooth.getConnectionStatus();
                  String addr =
getAddressString(Bluetooth.getLocalAddress());

                  // Friendly name of local device
                  LCD.drawString(nameString,0,0);
                  LCD.drawString(fn, 5, 0);

                  // Version of BlueCore software
                  LCD.drawString(versionString + version[0] + "." +
version[1],0,1);

                  // Local address
```

```
            LCD.drawString(addrString,0,2);
                  LCD.drawString(addr, 4, 2);

                  // Visibility
                  LCD.drawString(visString, 0, 3);
                  LCD.drawInt(Bluetooth.getVisibility(), 11, 3);

                  // Status byte
                  LCD.drawString(statusString, 0, 4);
                  LCD.drawInt(Bluetooth.getStatus() & 0xFF, 7, 4);

                  // Connections
                  LCD.drawString(connsString, 0, 5);
                  for(int i=0;i<4;i++) LCD.drawInt(connections[i], 2, 5
+ i*3, 5);

                  // Port open
                  LCD.drawString(portString, 0, 6);
                  LCD.drawInt(Bluetooth.getPortOpen(), 10, 6);

                  // Operating mode
                  LCD.drawString(opString, 0, 7);
                  LCD.drawInt(Bluetooth.getOperatingMode(), 8, 7);

                  LCD.refresh();
            }
      }

        private static String byteArrayToString(byte [] ba) {
              StringBuffer sb = new StringBuffer(ba.length);
              for(int i=0;i<ba.length;i++) {
                    sb.append((char) ba[i]);
              }
              return sb.toString();
        }

        private static final char[] hexChars =
{'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

        private static String getAddressString(byte [] addr) {
              char[] caddr = new char[12];

              for(int i=0; i<6; i++) {
                    int nr = addr[i] & 0xFF;
                    caddr[i*2] = hexChars[nr & 0x0F];
                    caddr[i*2+1] = hexChars[nr >> 4];
              }
              return new String(caddr, 0, 12);
        }
}
```

4.2.5.- SignalTest.java, An example to show how to use Bluetooth signal Strength.

This example shows how to use the feature to know the Bluetooth signal strength.

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStream;
```

```
import java.io.OutputStream;
import lejos.nxt.*;
import lejos.nxt.comm.*;

/**
 * Open a connection to the NXT using the BTSend sample
 * and then walk round the house measuring the signal
 * strength.
 *
 * @author Lawrie Griffiths
 *
 */
public class SignalTest {

      public static void main(String [] args)  throws Exception
      {
            String connected = "Connected";
        String waiting = "Waiting";
        String strength = "Signal: ";

            LCD.drawString(waiting,0,0);
            LCD.refresh();

        BTConnection btc = Bluetooth.waitForConnection();

            LCD.clear();
            LCD.drawString(connected,0,0);
            LCD.refresh();

            while(!Button.ESCAPE.isPressed()) {
                  LCD.drawString(strength, 0, 3);
                  LCD.drawInt(btc.getSignalStrength(), 3, 9 ,3);
                  LCD.refresh();
                  Thread.sleep(1000);
            }

            btc.close();
      }
}
```

4.2.6.- StartUpTest.java, A complex example with a thread which manages Bluetooth.

This is a complex example which shows an example with a Thread. That thread manages the Bluetooth connection

```
import java.io.*;
import java.util.Vector;
import javax.microedition.lcdui.*;
import lejos.nxt.comm.*;
import lejos.nxt.*;
import javax.bluetooth.*;

public class StartUpText
{
   static Graphics g = new Graphics();
   static boolean btPowerOn = false;
   static final String blank = "                ";
```

```
   static final String defaultProgramProperty =
"lejos.default_program";
   static final String defaultProgramAutoRunProperty =
"lejos.default_autoRun";
   static final String sleepTimeProperty = "lejos.sleep_time";
   static final int defaultSleepTime = 2;
   static final int maxSleepTime = 10;
   static final String revision = "$Revision: 1666 $";
   static final int MAJOR_VERSION=0;
   static final int MINOR_VERSION=6;

   public static void playTune()
   {
      int [] freq = {523,784, 659};
      for(int i = 0; i<3; i++)
      {
         Sound.playNote(Sound.XYLOPHONE, freq[i], (i==3 ? 500 : 300));
      }
   }
   static void drawTopRow()
   {
      LCD.drawString(blank,0,0);
      g.drawRect(0,1, 13,5);  // battery icon
      g.drawRect(14,3,1,1);
      // 2.5 v shows as empty, 9 as full;
      int b = -5+Battery.getVoltageMilliVolt()/500 ;
      g.fillRect(0,2, b,4);
      byte [] nam = Bluetooth.getFriendlyName();
      for(int i=0; i<nam.length; i++)
      {
         if(nam[i] <32)break;
         g.drawChar((char)nam[i],(4+i)*6,0,false);
      }
      g.drawString(" BT",82, 0,!btPowerOn);  // invert when power is
off
      g.refresh();
   }

      private static boolean setBluetoothPower(boolean powerOn,
boolean needReset)
      {
            // Set the state of the Bluetooth power to be powerOn. Also
record the
            // current state of this in the BT status bytes.
            // If power is not on we need it on to check things
            if (!Bluetooth.getPower())
            {
                  Bluetooth.setPower(true);
            }
            // First update the status bytes if needed
            int status = Bluetooth.getStatus();
            if (powerOn != (status == 0))
                  Bluetooth.setStatus((powerOn ? 0 : 1));
            Bluetooth.setPower(powerOn);
            return powerOn;
      }

//    private static void drawGauge(int x, int y, int w, int h, int
max, int cur)
//    {
//          int segWidth = (w/max);
```

```
//          for(int i = 0; i < cur; i++)
//                g.fillRect(x + i*segWidth, y+1, segWidth-1, h-1);
//          for(int i = cur; i < max; i++)
//                g.drawRect(x + i*segWidth, y+1, segWidth-1, h-1);
//    }

      private static String formatVol(int vol)
      {
            if (vol == 0) return "mute";
            if (vol == 10) return "10";
            return " " + vol;
      }

      private static String getExtension(String fileName) {
            int dot = fileName.lastIndexOf(".");
            if (dot < 0) return "";
            else return fileName.substring(dot+1, fileName.length());
      }

      private static String getBaseName(String fileName) {
            int dot = fileName.lastIndexOf(".");
            if (dot < 0) return fileName;
            else return fileName.substring(0, dot);
      }

      private static void runDefaultProgram()
      {
       String defaultProgram =
Settings.getProperty(defaultProgramProperty, "");
       if (defaultProgram != null && defaultProgram.length() > 0)
       {
           String progName = defaultProgram + ".nxj";
           File f = new File(progName);
           if (f.exists()) f.exec();
           else Settings.setProperty(defaultProgramProperty, "") ;
       }
      }

      public static void main(String[] args) throws Exception {
            Indicators ind = new Indicators();
            USBRespond usb = new USBRespond();
            BTRespond bt = new BTRespond();
            String devices = "Devices";
            String found = "Found";
            String status = "Status ";
            String on = "on ";
            String off = "off";
            String visible = "vis  ";
            String invisible = "invis";
            String bluetooth = "Bluetooth";
            String system = "System";
        String sound = "Sound";
            String freeFlash = "Free flash";
            String freeMem = "Free ram";
            String battery = "Battery ";
            TextMenu filesMenu = new TextMenu(null,1);
            String[] topMenuData = {"Run Default", "Files",
"Bluetooth", "Sound", "System", "Version"};
            TextMenu topMenu = new TextMenu(topMenuData,1);
            String[] fileMenuData = {"Delete file"};
            TextMenu fileMenu = new TextMenu(fileMenuData,2);
```

```
            String[] programMenuData = {"Execute program", "Set as
Default", "Delete file"};
            TextMenu programMenu = new TextMenu(programMenuData,2);
            String[] wavMenuData = {"Play sample", "Delete file"};
            TextMenu wavMenu = new TextMenu(wavMenuData,2);
            String[] fileNames = new String[File.MAX_FILES];
            TextMenu menu = topMenu;
            String[] blueMenuData = {"Power off", "Search",
"Devices","Visibility"};
            String[] blueOffMenuData = {"Power on"};
            TextMenu blueMenu = new TextMenu(blueMenuData,3);
            TextMenu blueOffMenu = new TextMenu(blueOffMenuData,3);
            String[] soundMenuData = {"Volume:    ", "Key click: "};
            String[] soundMenuData2 = new String[2];
            TextMenu soundMenu = new TextMenu(soundMenuData, 2);
            int [][] Volumes = {{Sound.getVolume()/10, 784, 250, 0},
{Button.getKeyClickVolume()/10, Button.getKeyClickTone(1),
Button.getKeyClickLength(), 0}};
        int enterTone = Button.getKeyClickTone(1);
            int curItem = 0;
            String sleepTime = "Sleep Time: ";
            String[] systemMenuData = {"Format",sleepTime, "Auto Run"};
            String dot = ".";
        String[] yes_no = {"No","Yes"};
        TextMenu yes_noMenu = new TextMenu(yes_no,6);
        TextMenu systemMenu = new TextMenu(systemMenuData,5);
        int timeoutMinutes =
SystemSettings.getIntSetting(sleepTimeProperty, defaultSleepTime);
        int timeout = timeoutMinutes * 60000;
        String firmwareVersionString = "Firmware Version";
        String firmwareVersion = System.getFirmwareMajorVersion() +
"." +
                                       System.getFirmwareMinorVersion() +
"(rev. " +
                                       System.getFirmwareRevision() +
")";
        String menuVersion = "Menu Version";

        playTune();

 // Run default program if required
        if( System.getProgramExecutionsCount() == 1 &&
            (Button.readButtons() & 2) != 2 &&  //left button down?
            Settings.getProperty(defaultProgramAutoRunProperty,
"").equals("ON")
          )
           runDefaultProgram();

            File[] files = null;
            boolean quit = false;
            int visibility = 0;
            btPowerOn = setBluetoothPower(Bluetooth.getStatus() == 0,
false);
            ind.setDaemon(true);
            ind.start();
            usb.setDaemon(true);
            usb.setIndicator(ind);
            usb.start();
            bt.setDaemon(true);
            bt.setIndicator(ind);
            bt.start();
```

```
            // Defrag the file system
            files = File.listFiles();
            try {
                  File.defrag();
            }
            catch (IOException ioe) {
                  File.reset();
            }

        // Make a note of starting volumes so we know if it changes
        for(int i = 0; i < Volumes.length; i++)
            Volumes[i][3] = Volumes[i][0];

            while (!quit)
            {
               LCD.clear();
               drawTopRow();
            usb.setMenu(menu);
            bt.setMenu(menu);
                  if (menu == filesMenu) {
                        files = File.listFiles();
                        int len = 0;
                        for(int i=0;i<files.length && files[i] !=
null;i++) len++;
                        for(int i=0;i<len;i++) fileNames[i] =
files[i].getName();
                        for(int i = len; fileNames[i] != null &&
i<files.length;i++)fileNames[i] = null;
                        filesMenu.setItems(fileNames);
                  } else if (menu == blueMenu) {
                        LCD.drawString(bluetooth, 3, 1);
                        LCD.drawString(status,0,2);
                        visibility = Bluetooth.getVisibility();
                        LCD.drawString(btPowerOn ? on : off, 7, 2);
                        LCD.drawString(visibility == 1 ? visible :
invisible, 11, 2);
                        LCD.refresh();
                  } else if (menu == blueOffMenu) {
                        LCD.drawString(bluetooth, 3, 1);
                        LCD.drawString(status,0,2);
                        LCD.drawString(btPowerOn ? on : off, 7, 2);
                        LCD.refresh();
                  } else if (menu == systemMenu) {
                        LCD.drawString(system, 4, 1);
                        LCD.drawString(freeFlash, 0, 2);
                        int free = File.freeMemory();
                        int size = 5;
                        int pos = 11;
                        if (free >= 100000) {
                              size = 6;
                              pos = 10;
                        }
                        LCD.drawInt(free,size, pos, 2);
                        LCD.drawString(battery, 0,3);
                  int  millis = Battery.getVoltageMilliVolt() + 50;
                  LCD.drawInt((millis - millis%1000)/1000,11,3);
                  LCD.drawString(dot, 12, 3);
                  LCD.drawInt((millis% 1000)/100,13,3);
                  LCD.drawString(freeMem,0,4);

LCD.drawInt((int)(Runtime.getRuntime().freeMemory()),11,4);
```

```
                systemMenuData[1] = sleepTime + timeoutMinutes;
                  } else if (menu == soundMenu)
                  {
                        LCD.drawString(sound, 5, 1);
                        for(int i = 0; i < Volumes.length; i++)
                              soundMenuData2[i] = soundMenuData[i] +
formatVol(Volumes[i][0]);
                        soundMenu.setItems(soundMenuData2);
                  }
                  int selection = menu.select(curItem, timeout);
                  if (selection == -3) System.shutDown();
                if (menu == topMenu) {
                   if (selection == 0)
                   {
                      runDefaultProgram();
                   } else if (selection == 1) {
                         menu = filesMenu;
                   } else if (selection ==2) {
                         menu = (btPowerOn ? blueMenu : blueOffMenu);
                   } else if (selection == 3) {
                               menu = soundMenu;
                               // Turn of key click when in the sound
menu so it does
                               // not screw with the feedback sounds
                               Button.setKeyClickTone(1, 0);
                         } else if (selection == 4) {
                         menu = systemMenu;
                   } else if (selection == 5) {
                         LCD.clear();
                         drawTopRow();
                         LCD.drawString(firmwareVersionString, 0, 2);
                         LCD.drawString(firmwareVersion,1,3);
                         LCD.drawString(menuVersion,0,4);
                         LCD.drawString(MAJOR_VERSION + "." +
                                        MINOR_VERSION + "(rev." +
                                        revision.substring(10,
revision.length()-2) + ")", 1, 5);
                         Button.waitForPress();
                   } else if (selection == -1) {
                         quit = true;
                   }
                } else if (menu == filesMenu) {
                      if (selection >= 0 && files[selection] != null) {
                        String fileName = files[selection].getName();
                        String ext = getExtension(fileName);
                          LCD.clear();
                          drawTopRow();
                          TextMenu subMenu = fileMenu;
                          if (ext.equals("nxj") || ext.equals("bin"))
subMenu = programMenu;
                          if (ext.equals("wav")) subMenu = wavMenu;
                              subMenu.setTitle(fileNames[selection]);
                        int subSelection = subMenu.select(0,timeout);
                        if (subSelection == -3) System.shutDown();
                     if ((subMenu == fileMenu && subSelection == 0) ||
                          (subMenu == wavMenu && subSelection == 1)
                          || subSelection == 2)
                        {
                              files[selection].delete();
                                    try {
                                          File.defrag();
```

```
                                    } catch (IOException ioe) {
                                          File.reset();
                                    }
                              LCD.clear();
                        } else if (subMenu == programMenu &&
subSelection == 0)
                        {
                              files[selection].exec();
                        } else if (subMenu == programMenu &&
subSelection == 1)
                        {

      Settings.setProperty(defaultProgramProperty,
getBaseName(fileName));
                          } else if (subMenu == wavMenu && subSelection
== 0) {
                              Sound.playSample(files[selection]);
                          }
                      } else if (selection == -1) {
                        menu = topMenu;
                      }
                } else if (menu == blueOffMenu) {
                        if (selection == 0)
                        {
                              LCD.clear();
                              LCD.drawString("   Power on...", 0, 0);
                             btPowerOn = setBluetoothPower(true, true);
                              menu = blueMenu;
                        }
                        else
                              menu = topMenu;
                  } else if (menu == blueMenu) {
                  if (selection == 2) { //Devices
                  Vector devList = Bluetooth.getKnownDevicesList();
                        if (devList.size() > 0) {
                              String[] names = new
String[devList.size()];
                              for (int i = 0; i < devList.size(); i++)
{
                                    RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
                                    names[i] =
btrd.getFriendlyName(false);
                              }

                              TextMenu deviceMenu = new
TextMenu(names,1);
                              String[] subItems = {"Remove"};
                              TextMenu subMenu = new
TextMenu(subItems,5);

                              int selected;
                              do {
                                    LCD.clear();
                                    LCD.drawString(devices,5,0);
                                    selected =
deviceMenu.select(0,timeout);
                                    if (selected == -3)
System.shutDown();
                                    if (selected >=0) {
```

```
                                        RemoteDevice btrd =
((RemoteDevice) devList.elementAt(selected));
                                        LCD.clear();
                                        LCD.drawString(devices,5,0);

      LCD.drawString(names[selected],0,1);

      LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
                                        for(int i=0;i<4;i++)
LCD.drawInt(btrd.getDeviceClass()[i], 3, i*4, 3);
                                        int subSelection =
subMenu.select(0, timeout);
                                        if (subSelection == -3)
System.shutDown();
                                        if (subSelection == 0) {

      Bluetooth.removeDevice(btrd);
                                              selected = -1;
                                        }
                                  }
                            } while (selected >= 0);

                      } else {
                            LCD.clear();
                            LCD.drawString("no known devices", 0, 0);
                            LCD.refresh();
                            try {
                                  Thread.sleep(2000);
                            } catch (InterruptedException e) {}
                      }
                } else if (selection == 1) { // Search
                      //byte[] cod = {0,0,8,4}; // Toy, Robot
                            byte[] cod = {0,0,0,0}; // All
                      LCD.clear();
                      LCD.drawString("Searching ...", 0, 0);
                      Vector devList = Bluetooth.inquire(5, 10,cod);

                      if (devList.size() > 0) {
                            String[] names = new
String[devList.size()];
                            for (int i = 0; i < devList.size(); i++)
{
                                  RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
                                  names[i] =
btrd.getFriendlyName(false);
                            }

                            TextMenu searchMenu = new
TextMenu(names,1);
                            String[] subItems = {"Add"};
                            TextMenu subMenu = new
TextMenu(subItems,4);

                            int selected;
                            do {
                                  LCD.clear();
                                        LCD.drawString(found,6,0);
                                  selected =
searchMenu.select(0,timeout);
```

```
                                    if (selected == -3)
System.shutDown();
                                    if (selected >=0) {
                                          RemoteDevice btrd =
((RemoteDevice) devList.elementAt(selected));
                                          LCD.clear();
                                          LCD.drawString(found,6,0);

      LCD.drawString(names[selected],0,1);

      LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
                                          int subSelection =
subMenu.select(0, timeout);
                                          if (subSelection == -3)
System.shutDown();
                                          if (subSelection == 0)
Bluetooth.addDevice(btrd);
                                    }
                              } while (selected >= 0);

                        } else {
                              LCD.clear();
                              LCD.drawString("no devices", 0, 0);
                              try {
                                    Thread.sleep(2000);
                              } catch (InterruptedException e) {}
                        }
                     } else if (selection == 0) // On/Off
                     {
                              LCD.clear();
                              LCD.drawString("   Power off...", 0, 0);
                              LCD.refresh();
                              btPowerOn = setBluetoothPower(false,
true);
                              menu = blueOffMenu;
                     }else if (selection == 3) // Visibility
                     {
                         Bluetooth.setVisibility((byte) (visibility == 1
? 0 : 1));
                     } else if (selection == -1) {
                         menu = topMenu;
                   }

                   } else if (menu == systemMenu) {
                if (selection == 0)
                {
                   yes_noMenu.setTitle("Delete all files?");
                   int subSelection = yes_noMenu.select(0,timeout);
                   if (subSelection == -3) System.shutDown();
                   if(subSelection== 1)File.format();
                } else if (selection == 1) // Sleep time
                {
                      timeoutMinutes++;
                      if (timeoutMinutes > maxSleepTime) timeoutMinutes
= 0;
                      timeout = timeoutMinutes * 60000;
                      Settings.setProperty(sleepTimeProperty, "" +
timeoutMinutes);
                      curItem = selection;
                } else if (selection == 2)
                {
```

```
                  String defaultPrgm =
Settings.getProperty(defaultProgramProperty, "");
                  if(defaultPrgm != null && defaultPrgm.length() >0)
                  {
                     LCD.drawString("Auto Run:
"+Settings.getProperty(defaultProgramAutoRunProperty, "")+blank,0,2);
                     LCD.drawString("Default Program:    ",0,3);
                     LCD.drawString(" " + defaultPrgm + blank,0 , 4);
                     yes_noMenu.setTitle("Run at power up?");
                     int subSelection =  yes_noMenu.select(0, timeout)
;
                     if (subSelection == -3) System.shutDown();
                     if(subSelection == 0)

Settings.setProperty(defaultProgramAutoRunProperty, "OFF");
                     else if(subSelection == 1)

Settings.setProperty(defaultProgramAutoRunProperty, "ON");
                  }
               }
               if (selection != 1) { // Return to top menu for all but
sleep time
                     curItem = 0;
                   menu = topMenu;
               }
            } else if (menu == soundMenu) {
                        if (selection >= 0)
                        {
                              Volumes[selection][0]++;
                              Volumes[selection][0] %= 11;
                              for(int i = 0; i < Volumes.length; i++)
                                    soundMenuData2[i] =
soundMenuData[i] + formatVol(Volumes[i][0]);
                              soundMenu.setItems(soundMenuData2);
                              if (selection == 0)
                              {
                                    Sound.setVolume(Volumes[0][0]*10);
                                    Sound.playNote(Sound.XYLOPHONE,
Volumes[selection][1], Volumes[selection][2]);
                              }
                              else

      Sound.playTone(Volumes[selection][1], Volumes[selection][2], -
Volumes[selection][0]*10);
                              curItem = selection;
                        }
                        else
                        {
                              // Make sure key click is back on and has
new volume

      Button.setKeyClickVolume(Volumes[1][0]*10);
                     Button.setKeyClickTone(1, enterTone);
                     // Wait for any sound to complete, writing to
flash distorts
                     // any playing sound...
                     Sound.pause(Sound.getTime()+250);
                     // Save in settings
                     if (Volumes[0][0] != Volumes[0][3])
                         Settings.setProperty(Sound.VOL_SETTING,
Integer.toString(Volumes[0][0]*10));
```

```
                    if (Volumes[1][0] != Volumes[1][3])
                        Settings.setProperty(Button.VOL_SETTING,
Integer.toString(Volumes[1][0]*10));
                    // Make a note of new volumes so we know if it
changes
                    for(int i = 0; i < Volumes.length; i++)
                        Volumes[i][3] = Volumes[i][0];
                              menu = topMenu;
                              curItem = 0;
                        }
                  }
            }
            System.shutDown();
      }
}

class Indicators extends Thread
{
      private boolean io = false;

      public void ioActive()
      {
            io = true;
      }

      public void run()
      {
            String [] ioProgress = {".   ", " .  ", "  . "};
            int ioIndex = 0;
            boolean rewrite = false;
            while(true)
            {
                  try
                  {
                    if (io)
                    {
                       StartUpText.g.drawString("     ", 76, 0);
                       ioIndex = (ioIndex + 1) % ioProgress.length;
                          StartUpText.g.drawString(ioProgress[ioIndex],
78, 0);
                          io = false;
                          rewrite = true;
                    }
                    else if(rewrite)
                    {
                       LCD.drawString("   ",13,0);
                        StartUpText.g.drawString(" BT",82,
0,!StartUpText.btPowerOn);  // invert when power is off
                        StartUpText.g.refresh();
                        rewrite = false;
                    }
                    Thread.sleep(1000);
                  } catch (InterruptedException ie) {}
            }
      }
}

class USBRespond extends Thread
{
      TextMenu menu;
      Indicators ind;
```

```
      public void setMenu(TextMenu menu) {
            this.menu = menu;
      }

      public void setIndicator(Indicators ind) {
            this.ind = ind;
      }

    private void setAddress()
    {
        // Ensure the USB address property is set correctly. We use
the
        // Bluetooth address as our serial number.
        String SerialNo =
Bluetooth.addressToString(Bluetooth.getLocalAddress());
        if (!SerialNo.equals(USB.getSerialNo()))
        {
            Settings.setProperty(USB.SERIAL_NO, SerialNo);
            USB.setSerialNo(SerialNo);
        }
        byte[] fName = Bluetooth.getFriendlyName();
        char []cName = new char[fName.length];
        int cNameLen = 0;
        while (cNameLen < fName.length && fName[cNameLen] != 0)
        {
            cName[cNameLen] = (char)fName[cNameLen];
            cNameLen++;
        }
        String name = new String(cName, 0, cNameLen);
        if (!name.equals(USB.getName()))
        {
            Settings.setProperty(USB.NAME, name);
            USB.setName(name);
        }
    }

      public void run() {
            byte[] inMsg = new byte[64];
            byte [] reply = new byte[64];
            boolean cmdMode = true;
        USBConnection conn = null;
            int len;

        setAddress();

            while (true)
            {
            conn = USB.waitForConnection(0, NXTConnection.LCP);
            if (conn == null) {
                continue;
            }
            cmdMode = false;
                  while(!cmdMode)
                  {
                        len = conn.read(inMsg,64);

                        if (len > 0)
                        {
                              ind.ioActive();
                              menu.resetTimeout();
```

```
                              int replyLen =
LCP.emulateCommand(inMsg,len, reply);
                              if ((inMsg[0] & 0x80) == 0)
conn.write(reply, replyLen);
                              if (inMsg[1] == LCP.CLOSE|| inMsg[1] ==
LCP.DELETE) {
                                    if (inMsg[1] == LCP.DELETE) {
                                          try {
                                                File.defrag();
                                          } catch (IOException ioe) {
                                                File.reset();
                                          }
                                    }
                                    Sound.beepSequenceUp();
                                    menu.quit();
                              }
                        if (inMsg[1] == LCP.BOOT)
                        {
                            // Reboot into firmware update mode. Only
supported
                            // for USB connections.
                            System.boot();
                        }
                              if (inMsg[1] == LCP.NXJ_DISCONNECT) {
                                    conn.close();
                                    cmdMode = true;
                              }
                           }
                           else if (len < 0)
                           {
                                 conn.close();
                                 cmdMode = true;
                           }
                           Thread.yield();
                     }
               }
         }
}

class BTRespond  extends Thread {
      TextMenu menu;
      Indicators ind;

      public void setMenu(TextMenu menu) {
            this.menu = menu;
      }

      public void setIndicator(Indicators ind) {
            this.ind = ind;
      }

      public void run()
      {
            byte[] inMsg = new byte[64];
            byte [] reply = new byte[64];
            boolean cmdMode = true;
            BTConnection conn = null;
            int len;

            while (true)
            {
```

```
                // Wait for power on
                while (!Bluetooth.getPower())
                {
                      try {
                            Thread.sleep(500);
                      } catch (InterruptedException e) {}

                }
            conn = Bluetooth.waitForConnection(0, NXTConnection.LCP,
null);
            if (conn == null) {
                continue;
            }
                  cmdMode = false;

                  while(!cmdMode)
                  {
                        len = conn.read(inMsg,64);

                        if (len > 0)
                        {
                              ind.ioActive();
                              menu.resetTimeout();
                              int replyLen =
LCP.emulateCommand(inMsg,len, reply);
                              if ((inMsg[0] & 0x80) == 0)
conn.write(reply, replyLen);
                              if (inMsg[1] == LCP.CLOSE|| inMsg[1] ==
LCP.DELETE) {
                                    if (inMsg[1] == LCP.DELETE) {
                                          try {
                                                File.defrag();
                                          } catch (IOException ioe) {
                                                File.reset();
                                          }
                                    }
                                    Sound.beepSequenceUp();
                                    menu.quit();
                              }
                              if (inMsg[1] == LCP.NXJ_DISCONNECT) {
                                    conn.close();
                                    cmdMode = true;
                              }
                        }
                        else if (len < 0)
                        {
                              conn.close();
                              cmdMode = true;
                        }
                        Thread.yield();
                  }
            }
      }
}
```

4.3.- Connection with a GPS

4.3.1.- BTGPS.java, Example to test the way to connect with a GPS receiver.

This example shows how to separate Bluetooth connection and Data management.

```
import lejos.nxt.*;
import lejos.nxt.comm.*;
import java.util.*;
import java.io.*;
import javax.bluetooth.*;
import lejos.devices.*;

/**
 * This sample allows you to connect to a Bluetooth GPS and
 * read latitude, longitude, and altitude on the NXT LCD.
 * 1. Turn on your Bluetooth GPS
 * 2. Turn on the NXT and run this program
 * 3. It will search out Bluetooth devices. Select your GPS
 * 4. After it connects it will output the data to the LCD.
 * @author BB
 *
 */
public class BTGPS {

      static String found = "Found";

      public static void main(String[] args) {

            byte[] cod = {0,0,0,0}; // 0,0,0,0 picks up every Bluetooth
device regardless of Class of Device (cod).

            final byte[] pin = {(byte) '0', (byte) '0', (byte) '0',
(byte) '0'};

            int sentenceCount = 0; // DELETE ME

            InputStream in = null;

            LCD.clear();
            LCD.drawString("Searching ...", 0, 0);
            LCD.refresh();
            Vector devList = Bluetooth.inquire(5, 10,cod);

            if (devList.size() > 0) {
                  String[] names = new String[devList.size()];
                  for (int i = 0; i < devList.size(); i++) {
                        RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
                        names[i] = btrd.getFriendlyName(true);
                  }

                  TextMenu searchMenu = new TextMenu(names,1);
                  String[] subItems = {"Connect"};
                  TextMenu subMenu = new TextMenu(subItems,4);

                  int selected;
```

```
                do {
                LCD.clear();
                      LCD.drawString(found,6,0);
                      LCD.refresh();
                      selected = searchMenu.select();
                      if (selected >=0) {
                            RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(selected));
                            LCD.clear();
                            LCD.drawString(found,6,0);
                            LCD.drawString(names[selected],0,1);

      LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
                            int subSelection = subMenu.select();
                            if (subSelection == 0)
Bluetooth.addDevice(btrd);

                            LCD.clear();

                            BTConnection btGPS = null;
                            btGPS =
Bluetooth.connect(btrd.getDeviceAddr(), pin);

                            if(btGPS == null)
                                  LCD.drawString("No Connection", 0,
1);
                            else
                                  LCD.drawString("Connected!", 0, 1);
                            LCD.refresh();

                            GPS gps = null;

                            btGPS.setIOMode(0); // Stream mode

                            try {
                                  in = btGPS.openInputStream();
                                  gps = new GPS(in);
                                  LCD.drawString("GPS Online", 0, 6);
                                  LCD.refresh();
                            } catch(Exception e) {
                                  LCD.drawString("Something bad", 0,
6);
                                  LCD.refresh();
                            }

                            while(true) {
                                  LCD.drawInt(++sentenceCount, 0, 0);
// DELETE
                                  LCD.drawString("Lat " +
gps.getLatitude(), 0, 1);
                                  LCD.drawString("Long " +
gps.getLongitude(), 0, 2);
                                  LCD.drawString("Alt " +
gps.getAltitude(), 0, 3);
                                  LCD.refresh();
                                  try {Thread.sleep(500);} catch
(Exception e) {}
                            }
                      }
                } while (selected >= 0);
```

```
            } else {
                  LCD.clear();
                  LCD.drawString("no devices", 0, 0);
                  LCD.refresh();
                  try {
                        Thread.sleep(2000);
                  } catch (InterruptedException e) {}
            }
      }
}
```

4.3.2.- GPS.java, This class manage an InputStream to process NMEA Sentences.

This class manages an InputStream to manage data.

```
package lejos.devices;

import java.io.*;
import lejos.nxt.comm.RConsole;

/**
 * Class to pull data from a GPS receiver
 * @author BB
 *
 */

/* DEVELOPER NOTES:
 * In general this could be improved by parsing print the appropriate
 * fields only at the time they are called, rather than parsing them
 * all at once as can be seen in the run() method.
 * This would save some CPU cycles since it would work to extract
 * data that the user wants, rather than parsing data that ends
 * up going unused.
 */
public class GPS extends Thread {

      /**
       * BUFF is the amount of bytes to read from the stream at once.
       * It should not be longer than the shortest NMEA sentence
otherwise
       * it might cause a bug.
       */
      private final int BUFF = 20;
      private byte [] segment = new byte[BUFF];
      private StringBuffer currentSentence = new StringBuffer();

      private String START_CHAR = "$";
      private String GGA_STR = "GGA";

      private InputStream in;

      private String BLANK = ""; // DELETE ME

      private int time = 0;
      private float latitude = 0;
      private float longitude = 0;
      private float altitude = 0;
      private byte satellites_tracked = 0;
```

```
	/**
	 *
	 * @param in An input stream from the GPS receiver
	 */
	public GPS(InputStream in) {
		this.in = in;
		this.setDaemon(true); // Must be set before thread starts
		this.start();
	}

	public float getLatitude() {
		return latitude;
	}

	public float getLongitude() {
		return longitude;
	}

	/**
	 * The altitude above mean sea level
	 * @return Meters above sea level e.g. 545.4
	 */
	public float getAltitude() {
		return altitude;
	}

	/**
	 * Returns the last time stamp retrieved from a satellite
	 *
	 * @return The time as a UTC integer. 123519 = 12:35:19 UTC
	 */
	public int getTime() {
		return time;
	}

	/**
	 * Returns the number of satellites being tracked to
	 * determine the coordinates.
	 * @return Number of satellites e.g. 8
	 */
	public byte getSatellitesTracked() {
		return satellites_tracked;
	}

	/**
	 * Placeholder idea: returns heading (from North) based on
	 * previous lat/long reading.
	 * @return heading
	 */
	public int getHeading() {
		return -1;
	}

	/**
	 * Placeholder idea: returns speed based on previous lat/long
reading.
	 * @return speed
	 */
	public int getSpeed() {
		return -1;
```

```
	}

	public void addGPSListener() {
		/* Placeholder method */
	}

	/**
	 * Placeholder Idea: Set a latitude/longitude as origin, then
	 * it will return x, y coordinates (in CM or Inches).
	 * By default, uses first reading as origin.
	 * Need to make methods for getX() and getY(), setUnits()
	 * @param longitude
	 * @param latitude
	 */
	public void setOriginPoint(String longitude, String latitude) {

	}

	/**
	 * Keeps reading sentences from GPS receiver stream and
extracting data.
	 * This is a daemon thread so when program ends it won't keep
running.
	 */
	public void run() {
		/* Code holder for parsing values */

		while(true) {
			String s = getNextString();

			// Check if sentence is valid:
			if(s.indexOf('*') < 0) {
				RConsole.print("Error no * caught!\n");
				RConsole.print("String: " + s + "\n");
				continue;
			}
			if(s.indexOf('$') < 0) {
				RConsole.print("Error no $ caught!\n");
				RConsole.print("String: " + s + "\n");
				continue;
			}

			// Make NMEASentence
			NMEASentence sen = new NMEASentence(s);
			RConsole.print("String: " + s + "\n");

			// Check if valid (discard if it is invalid)
			if(sen.isValid()) {

				// Check if contains lat/long data
				if(sen.getDataType().equals(GGA_STR)) {

					// Update all global vars
					//time =
Integer.parseInt((String)sen.getDataFields().elementAt(0)); // Convert
string to int

					latitude =
Float.parseFloat((String)sen.getDataFields().elementAt(1));
					longitude =
Float.parseFloat((String)sen.getDataFields().elementAt(3));
```

```
                              altitude =
Float.parseFloat((String)sen.getDataFields().elementAt(7));

                              //satellites_tracked  =
(byte)Integer.parseInt((String)sen.getDataFields().elementAt(9));
                        }
                  // Notify appropriate listeners if data changed
                  }
            }
      }

      /**
       * Pulls the next NMEA sentence as a string
       * @return NMEA string, including $ and end checksum
       */
      private String getNextString() {
            boolean done = false;
            do {
                  // Read in buf length of sentence
                  try {
                        in.read(segment);
                  } catch (IOException e) {
                        // How to handle error?
                  }
                  // Append char[] data into currentSentence
                  for(int i=0;i<BUFF;i++)
                        currentSentence.append((char)segment[i]);

                  // Search for $ symbol (indicates start of new
sentence)
                  if(currentSentence.indexOf(START_CHAR, 1) >= 0) {
                        done = true;
                  }

            } while(!done);

            int endIndex = currentSentence.indexOf(START_CHAR, 1);
            String sentence = currentSentence.substring(0, endIndex);

            // Crop print current sentence
            currentSentence.delete(0, endIndex);

            return sentence;
      }
}
```

4.4.- Connection with a BT Keyboard

4.4.1.- KeyboardTest.java, This example teach how to use Java Events and Bluetooth connection together.

This example tests others Bluetooth features.

```
import lejos.nxt.*;
import lejos.nxt.comm.Bluetooth;
import lejos.nxt.comm.RConsole;
import lejos.nxt.comm.NXTConnection;
```

```
import javax.bluetooth.RemoteDevice;
import lejos.nxt.comm.BTConnection;
import lejos.devices.*;

import java.util.Vector;
import java.io.*;

/**
 * This is some sample code to demonstrate the Keyboard class. It
 * allows you to connect and display typing on the NXT LCD.
 * Only works with SPP Bluetooth keyboards (very rare). Will not work
with
 * HID BT keyboards. See Keyboard Javadocs for more information.
 * @author BB
 */
public class KeyboardTest implements KeyListener {

      static boolean cont = true;

      public KeyboardTest() {
            Keyboard k = connectKeyboard();
            k.addKeyListener(this);
      }

      public static void main(String [] args) {
            //RConsole.openUSB(0);
            //RConsole.openBluetooth(0);
            //System.setErr(new
PrintStream(RConsole.openOutputStream()));
            //System.err.println("Starting...");

            KeyboardTest kt = new KeyboardTest();
            LCD.clear();
            LCD.refresh();
            while(cont) {Thread.yield();}
            System.out.println("Quitting...");
            //RConsole.close();
      }

      public void keyPressed(KeyEvent e) {
            System.out.print("" + e.getKeyChar());
            if(e.getKeyChar() == 'q') {
                  cont = false; // or System.exit(0);
            } else if(e.getKeyChar() == 'z') {
                  System.out.println("");
            }
      }

      public void keyReleased(KeyEvent e) {
            System.out.println("Key Released: " + e.getKeyChar());
            if(e.getKeyChar() == 'q') {
                  cont = false;
            }
      }

      public void keyTyped(KeyEvent e) {
            System.out.println("Key Typed: " + e.getKeyChar());
            if(e.getKeyChar() == 'q') {
                  cont = false;
            }
```

```
	}

	public Keyboard connectKeyboard() {

		Keyboard k = null;

		byte[] cod = {0,0,0,0}; // 0,0,0,0 picks up every Bluetooth
device regardless of Class of Device (cod).

		final byte[] pin = {(byte) '0', (byte) '0', (byte) '0',
(byte) '0'};

		InputStream in = null;
		OutputStream out = null;

		System.out.println("Searching ...");
		Vector devList = Bluetooth.inquire(5, 10,cod);

		if (devList.size() > 0) {
			String[] names = new String[devList.size()];
			for (int i = 0; i < devList.size(); i++) {
				RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(i));
				names[i] = btrd.getFriendlyName(false);
			}

			TextMenu searchMenu = new TextMenu(names,1);
			String[] subItems = {"Connect"};
			TextMenu subMenu = new TextMenu(subItems,4);

			int selected;
			do {
			LCD.clear();
				LCD.drawString("Found",6,0);
				LCD.refresh();
				selected = searchMenu.select();
				if (selected >=0) {
					RemoteDevice btrd = ((RemoteDevice)
devList.elementAt(selected));
					LCD.drawString(names[selected],0,1);

	LCD.drawString(btrd.getBluetoothAddress(), 0, 2);
					int subSelection = subMenu.select();
					if (subSelection == 0)
Bluetooth.addDevice(btrd);

					//LCD.clear();

					// BELOW: Once paired, no need to use
pin? Might
					// use alternate method.
					BTConnection btSPPDevice = null;
					//System.err.println("About to connect
w/pin " + (char)pin[0] + (char)pin[1] + (char)pin[2] + (char)pin[3] +
"\n");
				// Open connection in raw/stream mode
				btSPPDevice =
Bluetooth.connect(btrd.getDeviceAddr(), NXTConnection.RAW, pin);
					//if(btSPPDevice == null)
					//    System.err.println("Connect
failed.\n");
```

```
                              //else
                              //    System.err.println("Connect
worked.\n");

                              //System.err.println("Setting to stream
mode.\n");
                              //btSPPDevice.setIOMode(0); // 0 = Stream
mode?

                              try {
                                    //System.err.println("About to open
BTConnection.openInputStream()"  + "\n");
                                    in = btSPPDevice.openInputStream();
                                    out =
btSPPDevice.openOutputStream();
                                    //System.err.println("Streams
captured.\n");
                                    k = new Keyboard(in, out);
                                    selected = -1; // to make it stop
looping?

                              } catch(Exception e) {
                                    //System.err.println("InputStream
NOT captured." + "\n");
                              }
                        }
                  } while (selected >= 0);

            } else {
                  LCD.clear();
                  LCD.drawString("no devices", 0, 0);
                  LCD.refresh();
                  try {
                        Thread.sleep(2000);
                  } catch (InterruptedException e) {}
            }
            return k;
      }
}
```

4.4.2.- Keyboard.java, Another example to test others Bluetooth features.

This example shows another time how to separate a Bluetooth connection and data management. Keyboard.java manages an InputStream and OutputStream and define events. Very interesting example.

```
package lejos.devices;

import java.io.*;

/*
 * Developer Notes:
 * The following document explains keyboard control:
 * http://www.computer-engineering.org/ps2keyboard/
 */

/**
```

```
 * This class will only work with SPP keyboards, not standard HID
 * keyboards. If it doesn't say it supports Bluetooth SPP then it
 * will not work. There are only two known SPP models (also available
 * on eBay or Amazon):
 * Freedom Universal Bluetooth keyboard
 * http://www.freedominput.com
 * iTech Virtual Keyboard (SPP only)
 * http://www.virtual-laser-keyboard.com/
 *
 * Note: This class is currently only tested with Freedom Universal
 *
 */
// TODO: Create a method that returns a stream of ASCII from keyboard.
Thread handles keep alive, auto connect
// and reconnecting if disconnected.
// TODO: Currently only handles make code. Need break code (key
release).
// TODO: Repeat when key held down. (See above)


// Typematic Key Repeat:
// Typematic delay - 0.25 seconds to 1.00 second (500ms default)
// Typematic rate - 2.0 cps (characters per second) to 30.0 cps (10.9
default)
// "Set Typematic Rate/Delay" (0xF3) command
// Typematic data is not buffered within the keyboard.  In the case
// where more than one key is held down, only the last key pressed
// becomes typematic.  Typematic repeat then stops when that key is
// released, even though other keys may be held down.

public class Keyboard extends Thread {

      /**
       * Command used by keyboards. Full list of commands can be found
here:
       * http://www.computer-engineering.org/ps2keyboard/
       */
      private static int ECHO = 0xEE;

      /**
       * Time between echo command sent to keyboard to prevent it
       * from going to sleep. Universal Keyboard sleeps after 5000 ms
       */
      private static int KEEP_ALIVE_DELAY = 4000; // milliseconds

      private static int MAX_LISTENERS = 3;

      private InputStream in = null;
      private OutputStream out = null;

      // Use Vector? Resizable array?
      private KeyListener [] listeners = new
KeyListener[MAX_LISTENERS];

      /**
       * The index of this array is the Scan Code (e.g. 0x1C) and the
       * value at that index is the appropriate ASCII key (without
shift key).
       * NOTE: There is no ASCII Caps Lock for 0x58. There is shift-in
and shift-out, but that
```

```
       * seems to have a different function than Caps (i.e. Caps only
applies to A-Z, but shift
       * applies to number keys displaying symbols).
       * Note: Scan code 0x12 is left-shift, however ASCII has shift-
in (0x0F) and shift-out (0x0E) so
       * I'm uncertain how to handle this as a stream. Will have to do
some special case statements. This
       * might not really be a factor anyway since shifts probably
irrelevant to ASCII stream.
       * Note: ASCII has "Device Control" 1-4 I'm assuming ALT and
CTRL fit in here. Maybe Fn. Maybe Windows key.
       * Again, these might be irrelevant to ASCII stream.
       * NOTE: Enter - equivalent to line feed? (ASCII 0x0A) Should
stream
       * send two characters: line feed and carriage return?
       * NOTE: There are no ASCII characters for arrow keys.
       */
      // 64 unique keys on Freedom Universal, array is 113, many
blanks
      private static byte [] scanCodes = {
            0,0,0,0,0,0,0,0x14,0,0,0,0,0,0x09,0x60,0, // 0x00

      0,0x12,0x0F,0x12,0x11,0x71,0x31,0,0,0,0x7A,0x73,0x61,0x77,0x32,0
, // 0x10

      0,0x63,0x78,0x64,0x65,0x34,0x33,0,0,0x20,0x76,0x66,0x74,0x72,0x3
5,0, // 0x20

      0,0x6E,0x62,0x68,0x67,0x79,0x36,0,0,0,0x6D,0x6A,0x75,0x37,0x38,0
, // 0x30

      0,0,0x6B,0x69,0x6F,0x30,0x39,0,0,0x2E,0x2F,0x6C,0x3B,0x70,0x2D,0
, // 0x40
            0,0,0x27,0,0x5B,0x3D,0,0,0x00,0x0F,0x0A,0x5D,0x20,0x5C,0,0,
// 0x50
            0,0,0,0,0,0,0x7F,0,0,0,0,0,0,0,0,0, // 0x60
            0,0x08 // 0x70
      };

      public Keyboard(InputStream in, OutputStream out) {
            this.in = in;
            this.out = out;
            this.setDaemon(true);
            this.start();
      }

      // TODO: Make InputStream that implements KeyListener, outputs
ASCII
      public void addKeyListener(KeyListener l) {
            // !! Test code until I figure out storage object
            listeners[0] = l;
      }

      public void removeKeylistener(KeyListener l) {
            // !! Test code until I figure out storage object
            listeners[0] = null;
      }

      private void notifyListeners(KeyEvent e) {
            // !! Test code until I figure out storage object
            if(e.getID() == KeyEvent.KEY_PRESSED)
```

```
                listeners[0].keyPressed(e);
          else if(e.getID() == KeyEvent.KEY_RELEASED)
                listeners[0].keyReleased(e);
          else if(e.getID() == KeyEvent.KEY_TYPED)
                listeners[0].keyTyped(e);
    }

    /**
     * Converts raw keyboard scan code into ASCII.
     * Works with lower case only.
     * @param scanCode
     * @return the ascii character
     */
    // TODO: Make private
    public static byte getASCII(byte scanCode) {
          if(scanCode < 0) // bit 8 makes value 0
                return 0; // !! Need to handle break code here
          return scanCodes[scanCode]; // !! handles make codes only
    }

    public void run() {
          //Debug.out("Keyboard thread started.\n");
          int previousEcho = (int)System.currentTimeMillis();
          while(true) {

                // Keep-alive code:
                int now = (int)System.currentTimeMillis();
                if(now - previousEcho >= KEEP_ALIVE_DELAY) {
                      try {
                            //Debug.out("Echo Sent\n");
                            out.write(ECHO);
                            out.flush();
                      } catch(IOException e) {/*Debug.out("COMMAND
EXCEPTION");*/}
                      previousEcho = now;
                }

                // Notifier code:
                try {
                      if(in.available() > 0) { // Check if byte
available.
                            int bval = in.read();
                            KeyEvent e = getKeyEvent(bval, now);
                            if(e != null) notifyListeners(e); //
ignore non-chars
                      }
                } catch(IOException e) {/*Debug.out("EXCEPTION");*/}
                Thread.yield();
          }
    }

    /**
     * Helper method to generate keyEvent from scan code from
keyboard
     * @return
     */
    private KeyEvent getKeyEvent(int scanCode, int timeStamp) {
          // TEST CODE:
          char curChar = (char)Keyboard.getASCII((byte)scanCode);
          if((byte)scanCode < 0) return null; // if 8th bit on (i.e.
key release)
```

```
            KeyEvent e = new KeyEvent(this, KeyEvent.KEY_PRESSED,
timeStamp, 0, 0, curChar);
            //Debug.out(++sentenceCount + ": " + bval + " = " +
(char)Keyboard.getASCII((byte)bval) + "\n");
            //System.out.println("" +
(char)Keyboard.getASCII((byte)scanCode));
            return e;
      }
}
```

4.4.3.- KeyListener.java, This class is a nice example to show how to create listener for events.

Previous example showed how to define events. When you define events, you have to define a listener.

```
package lejos.devices;
// package lejos.devices; // UNCOMMENT

/**
 * This interface is for classes that wish to receive keyboard events.
 *
 * @author BB
 * @see Keyboard
 * @see KeyEvent
 * @since 0.6
 */
public interface KeyListener {
  /**
   * This method is called when a key is typed.  A key is considered
typed
   * when it and all modifiers have been pressed and released, mapping
to
   * a single virtual key.
   *
   * @param event the <code>KeyEvent</code> indicating that a key was
typed
   */
  void keyTyped(KeyEvent event);

  /**
   * This method is called when a key is pressed.
   *
   * @param event the <code>KeyEvent</code> indicating the key press
   */
  void keyPressed(KeyEvent event);

  /**
   * This method is called when a key is released.
   *
   * @param event the <code>KeyEvent</code> indicating the key release
   */
  void keyReleased(KeyEvent event);
}
```

4.4.4.- KeyEvent.java, This example shows how to define the events.

When you define an event then you define the listener and the events.

```
package lejos.devices;

public class KeyEvent {

	public static final int SHIFT_MASK = 1;
	public static final int CTRL_MASK = 2;
	public static final int META_MASK = 4;
	public static final int ALT_MASK = 8;
	public static final int ALT_GRAPH_MASK = 32;

	public static final int SHIFT_DOWN_MASK = 64;
	public static final int CTRL_DOWN_MASK = 128;
	public static final int META_DOWN_MASK = 256;
	public static final int ALT_DOWN_MASK = 512;
	public static final int ALT_GRAPH_DOWN_MASK = 8192;

	/**
	 * Keycode constant for the down arrow key
	 */
	public static final int VK_DOWN = 40;

	/**
	 * Keycode constant for the right arrow key
	 */
	public static final int VK_RIGHT = 39;

	/**
	 * Keycode constant for the up arrow key
	 */
	public static final int VK_UP = 38;

	/**
	 * Keycode constant for the left arrow key
	 */
	public static final int VK_LEFT = 37;

	public static short KEY_TYPED = 400;
	public static short KEY_PRESSED = 401;
	public static short KEY_RELEASED = 402;

	private Object source;
	private int when;
	private char keyChar;
	private short id;
	private int modifiers;
	private int keyCode;

	public KeyEvent(Object source, short id, int when, int
modifiers, int keyCode, char keyChar) {
		this.when = when;
		this.keyChar = keyChar;
		this.id = id;
		this.source = source;
		this.modifiers = modifiers;
		this.keyCode = keyCode;
```

```
}

public char getKeyChar() {
      return this.keyChar;
}

/**
 * Will actually only return key codes that don't have
 * ASCII/Unicode equivalents, such as arrow keys (e.g. VK_UP).
 * Otherwise it just returns the ASCII value.
 * @return the key code
 */
public int getKeyCode() {
      return this.keyCode;
}

/**
 * Not Implemented
 * @return 0
 */
public int getKeyLocation() {
      return 0;
}

public int getID() {
      return this.id;
}

public boolean isShiftDown() {
      return false;
}

public boolean isControlDown() {
      return false;
}

public boolean isMetaDown() {
      return false;
}

public boolean isAltDown() {
      return false;
}

public boolean isAltGraphDown() {
      return false;
}

public int getWhen() {
      return this.when;
}

public int getModifiers() {
      return this.modifiers;
}

public void consume() {

}

public boolean isConsumed() {
```

```
            return false;
      }

      public Object getSource() {
            return this.source;
      }
}
```