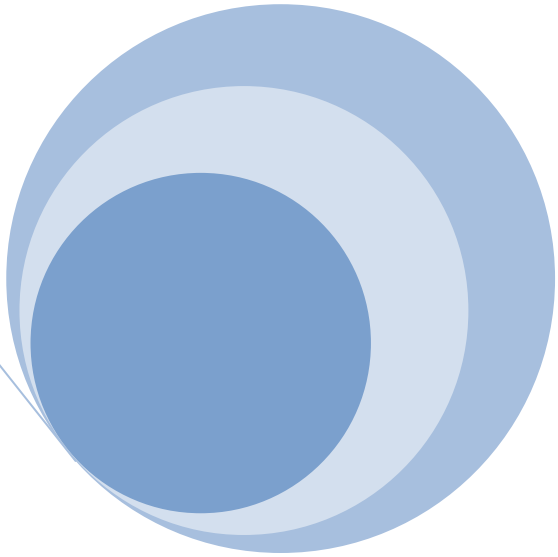
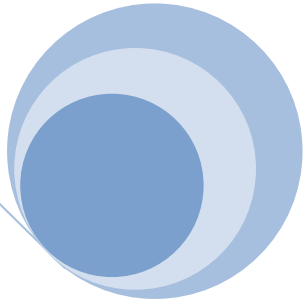
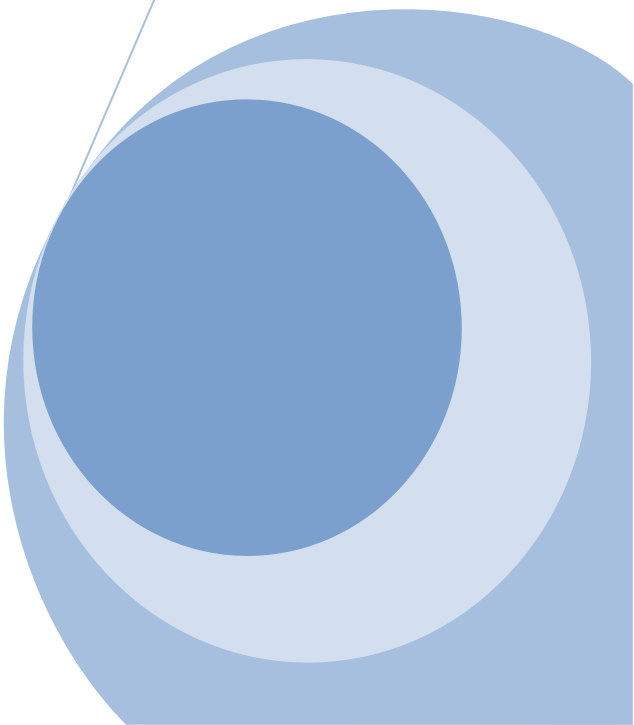




LiteOS Application Note AN-101: Mote-PC communication and data display

Last updated: October 9 2007



This application note describes how to stream sensor data to PC and display them using various tools.

Download location: www.liteos.net

Copyright ©2007 LiteOS developers, all rights reserved.

Purpose

This document shows how to stream data from the mote to the PC and display and analyze the data using provided tools. The environment LiteOS provides is based on Java. Most tools in the Java infrastructure is ported from TinyOS 1.1, such as SerialForwarder, Listen, Oscilloscope. This document also presents generic guidelines so that you can port other TinyOS java tools to LiteOS and use them to analyze data.

This document is related to, and based on two TinyOS tutorial lessons on the similar topic:

The Lesson 6 of TinyOS 1.x tutorial:

http://www.tinyos.net/tinyos-1.x/doc/tutorial/lesson6.html

The Lesson 4 of TinyOS 2.0 tutorial:

http://www.tinyos.net/tinyos-2.x/doc/html/tutorial/lesson4.html

It is recommended that you refer to these documents as well as this document to better understand and use the Java tools.

Introduction to LiteOS java environment

LiteOS 0.2 provides a directory called JavaTools, where all tools related to Java are organized. We have introduced and demonstrated the installer tool as well as the terminal in the user's guide. This documents focuses on the following tools:

- The Listen program for showing data messages
- The Oscilloscope program for showing data readings
- The SerialForwarder program for translating serial port into sockets so that multiple Java programs can access the same port

The SerialForwarder Program

The most basic way for Java programs to listen to incoming packets is that they directly open a serial port and dumps the packets in the screen. The problem with this approach lies in that multiple Java programs could not share the same serial port. Therefore, we need a better way such that the serial port can be multiplexed among multiple applications.

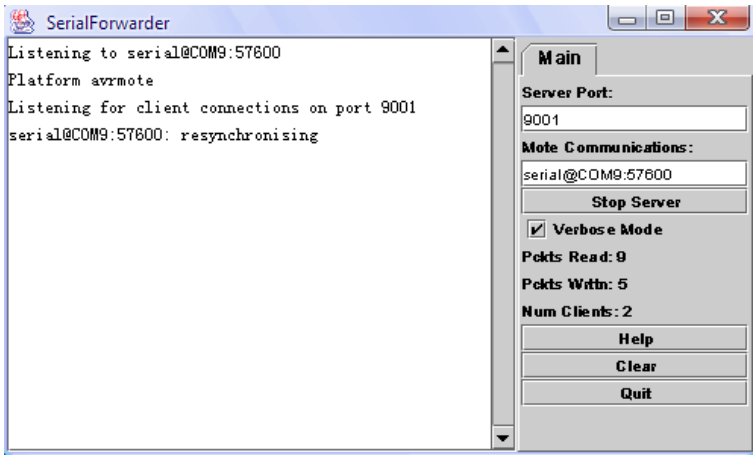
The SerialForwarder program is used to read data packets from a serial port, and forwards them to a socket. Other programs could then read and write to the serial forwarder using this socket, even across the Internet. The SerialForwarder in LiteOS is ported from TinyOS 1.1 with minor revisions.

To run the serial forwarder, cd to the javatools/classes directory, and type:

```
java tools.sf.SerialForwarder -comm serial@COM9:57600
```

Here, 57600 is the default port rate for MicaZ. You may use a different port if your programming board is connected to another serial port. Open the device manager to find the correct COM number for your environment.

The above command will open up a GUI window that looks like the following:



SerialForwarder does not display any data packets, but rather updates the packet counters in the lower-right hand corner of the window. Once SerialForwarder is started, it will listen for network client connections on a given TCP port

(9001 is the default), and simply forwards packets from the serial port to the network client connection, and vice versa. Note that, the key advantage of using SerialForwarder is that multiple applications can connect to the serial forwarder at once, and all of them will receive a copy of the messages from the sensor network.

Use the Listen Program to display incoming packets

After you have started SerialForwarder, you may use the Listen program to display incoming data packets. To do this, type the following after you cd into the javatools/classes directory.

```
java tools.tools.listen
```

The listen tool will directly connect to the 9001 port of the localhost. If it successfully connects to the SerialForwarder, you will see data displays something like this. (In this example, we are using the terminal to talk to the sensor network. The data flow hence is specific to the data formats defined in LiteOS commands)

```
FF FF EF EF 20 07 66 05 6E 6F 64 65 6E 6B 32 6C 6C 00 00 00 00 00 02 40 05 72 91
 84 07 13 92 00 00 00 00 00 00
FF FF EF EF 20 07 66 05 6E 6F 64 65 6E 6B 32 6C 6C 00 00 00 00 00 02 40 05 72 91
 84 07 13 92 00 00 00 00 00 00
FF FF EF EF 20 20 68 05 01 00 64 65 76 00 00 00 00 00 00 00 00 00 01 00 00 72 91
 84 07 13 92 00 00 00 00 00 00
FF FF EF EF 20 20 68 05 08 01 62 6C 69 6E 6B 00 00 00 00 00 00 00 02 10 05 72 91
 84 07 13 92 00 00 00 00 00 00
FF FF EF EF 20 20 68 05 09 02 69 6D 61 67 65 00 00 00 00 00 00 00 02 B2 01 72 91
 84 07 13 92 00 00 00 00 00 00
FF FF EF EF 20 20 68 05 0A 03 62 6C 69 6E 6B 32 00 00 00 00 00 00 02 40 05 72 91
 84 07 13 92 00 00 00 00 00 00
```

The Listen program simply prints out the raw data of each packet received from the serial port.

Parsing the data packets

The Listen application simply prints out the packets that it receives from the serial forwarder. It does not, however, tell you what each field means. Following is the definition of the generic packet header and content format for LiteOS packets received from the serial forwarder.

- **Destination Address** (2 bytes) (0xFFFF as the broadcast address.)
- **Port Number** (2 bytes) (Currently only the first byte of port number is meaningful. The 2 bytes length is kept for future update use.)
- **Message length** (1 byte) (Up to 256 bytes)
- **Payload** (up to 100 bytes) (Actual packets are usually shorter than 50 bytes to avoid corruptions.)

For the previous packets, the destination address is 0xFFFF, while the port number is 0xEFEF. Currently, only the first byte of this port number is used to differentiate different packets. The two-byte length is designed for future system updates.

In the previous example, the message length is 32 bytes. The payload of the message follows the definitions of the LiteOS command interface, which allows the LiteOS shell to communicate with sensor network using Unix-like commands. Describing them is beyond the scope of this document.

Starting the Oscilloscope to visualize readings

You may notice, as you use Listen, that it displays all packets in all port number and destination addresses. This is rarely the case when you develop your own applications. One such application, Oscilloscope, is an example. For the Oscillocope application, the sensor nodes is programmed with the SenseAndSendOscilloscope application, and sends out the data readings. It uses the port number of 10 when sending packets. On the PC side, the Oscilloscope java application listens to such incoming packets and visualizes them.

The detailed structure of the OscopeMsg format is as follows: (defined in Apps\SenseAndSendOscilloscope\Sense.c)

```
enum {
  BUFFER_SIZE = 10
};

struct OscopeMsg
```

```
{
   uint16_t sourceMoteID;
   uint16_t lastSampleNumber;
   uint16_t channel;
   uint16_t data[BUFFER_SIZE];
};
```

First, enter the LiteOS shell and start the Sense application as follows:

```
$cp /c/Temp2/sense sense
The reply has 1 packets.
Now trying to send 0 27
Now trying to send sync
Now reply is good on sync
cp succeed
Copy finished
Time elapes 2844

$ls -l
The returned has 2 packets.
Name    Type     Size      Protection
dev  directory  --    rwxrwxrwx
sense  file  1312    rwxrwxrwx
Time elapes 500

$exec sense
Process execution succeeds.
Time elapes 812
$Time elapes 813

$ls
The returned has 2 packets.
dev
sense
Time elapes 515
```

Then you start the SerialForwarder application if you have not done so:

```
java tools.sf.SerialForwarder -comm serial@COM9:57600
```

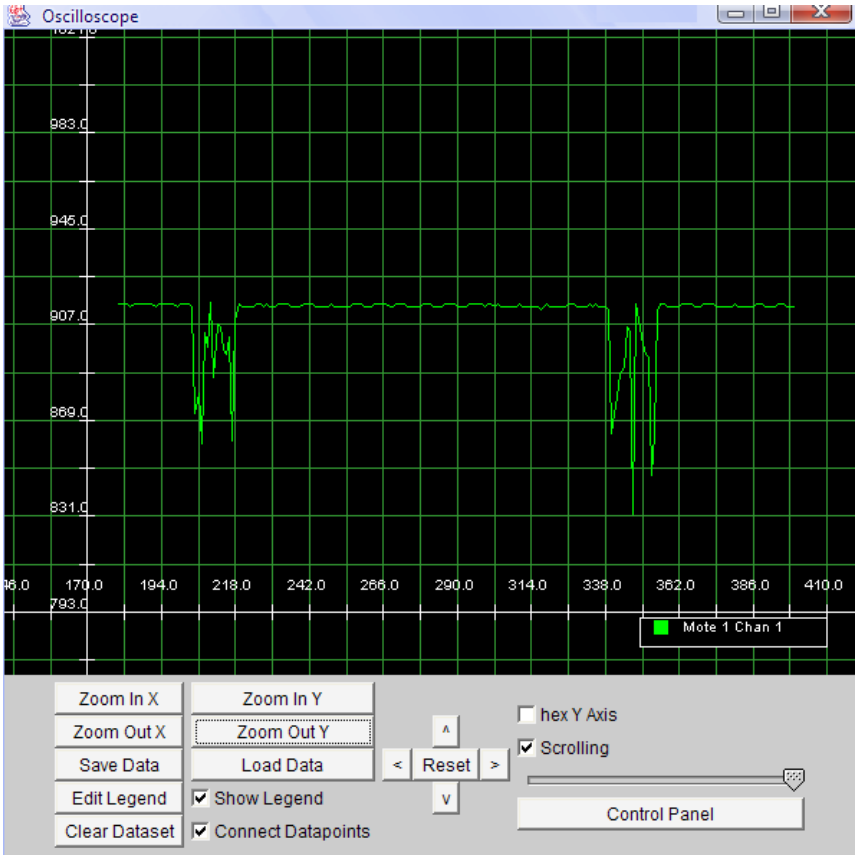
Next start the Listen application:

```
java tools.tools.Listen
```

You should be able to see that the data packets are coming in through the Listen program.

Next start the Oscillosopce application:

And you will see something like this:



Congratulations! You have now successfully streamed data into your PC from motes that are running LiteOS!