# Bluetooth Communication Sample Code

Here is some code to illustrate how to communicate between two NXT bricks via the build-in Bluetooth. The code follows the following protocol between a ‘sender’ and ‘receiver’ NXT:

1. **Receiver**: waits for connection
2. **Sender**: connect to receiver
3. **Receiver**: waits for data
4. **Sender**: sends an integer
5. **Receiver**: waits for data
6. **Sender**: sends an integer
7. **Sender**: waits for data
8. **Receiver**: sends an integer
9. **Sender & Receiver**: close connection

In an actually useful program you would put the code into methods like ‘send’ and ‘receive’ and store relevant data in state variables for further processing instead of just displaying it on the screen. To try the code ‘as is’, create two programs. Upload one to the ‘sender’, the other to the ‘receiver’, then follow the instructions.

/\*

\* Test of NXT to NXT Bluetooth communication

\* ------------------------------------------

\* One NXT (the 'sender') connects to another NXT, sends 2 integers, and

\* waits for a single integer as reply. Then it closes the connection.

\*

\* The other NXT (the 'receiver') waits for an NXT to establish a connection.

\* If it receives a connection, it waits for two integers. After receiving

\* two integers it sends a single integer back and closes the connection. For

\* this to work:

\*

\* => each NXT must have a unique name

\* => the 'receiver' must be a 'known device' for the 'sender' (see below)

\* => the receiver must run BTReceive first and wait

\* => the sender runs BTSend to get the chain of events going.

\*

\* To setup the 'sender' (before running the program):

\*

\* 1. Change the name string in BTSender.java to the name of your receiver NXT

\* 2. Make sure the receiver is in the known devices list of the sender:

\*

\* a) turn on the receiver NXT

\* b) make sure Bluetooth is on and the device is visible

\* c) on the sender, select the Bluetooth menu and select Search

\* d) the name of the receiver NXT should appear

\* e) select Add to add it to the known devices of the sender

\*

\* You can check this has been done by selecting 'Devices' from the Bluetooth

\* menu on the sender.

\*

\* 3. Compile, upload, and run 'BTReceiver' on the receiver NXT

\* 4. Specify correct name in sender code, then compile, upload, and run the

\* 'BTSender' program on the sender NXT

\*/

# BTReceive Program

See above comments that describe the two classes working in conjunction with each other (perhaps copy the comments into these classes). Note that you must create two separate classes, one for program. Also, the last line to sleep for 4 seconds is only there so you can read the screen!

**import** java.io.\*;

**import** lejos.nxt.LCD;

**import** lejos.nxt.comm.BTConnection;

**import** lejos.nxt.comm.Bluetooth;

**public** **class** BTReceive

{

**public** **static** **void** main(String[] args) **throws** Exception

{

LCD.*clear*();

LCD.*drawString*("Receiver wait...", 0, 0);

LCD.*refresh*();

**try**

{

BTConnection connection = Bluetooth.*waitForConnection*();

**if** (connection == **null**)

**throw** **new** IOException("Connect fail");

LCD.*drawString*("Connected.", 1, 0);

DataInputStream input = connection.openDataInputStream();

DataOutputStream output = connection.openDataOutputStream();

**int** answer1 = input.readInt();

LCD.*drawString*("1st = " + answer1, 2, 0);

**int** answer2 = input.readInt();

LCD.*drawString*("2nd = " + answer2, 3, 0);

output.writeInt(0);

output.flush();

LCD.*drawString*("Sent data", 4, 0);

input.close();

output.close();

connection.close();

LCD.*drawString*("Bye ...", 5, 0);

}

**catch**(Exception ioe)

{

LCD.*clear*();

LCD.*drawString*("ERROR", 0, 0);

LCD.*drawString*(ioe.getMessage(), 2, 0);

LCD.*refresh*();

}

Thread.*sleep*(4000);

}

}

# BTSend Program

**import** java.io.\*;

**import** javax.bluetooth.RemoteDevice;

**import** lejos.nxt.LCD;

**import** lejos.nxt.comm.BTConnection;

**import** lejos.nxt.comm.Bluetooth;

**public** **class** BTSend

{

**public** **static** **void** main(String[] args) **throws** Exception

{

// Change this to the name of your receiver

String name = "MyNXT";

LCD.*clear*();

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

LCD.*refresh*();

**try**

{

RemoteDevice receiver = Bluetooth.*getKnownDevice*(name);

**if** (receiver != **null**)

**throw** **new** IOException("no such device");

BTConnection connection = Bluetooth.*connect*(receiver);

**if** (connection == **null**)

**throw** **new** IOException("Connect fail");

LCD.*drawString*("connected.", 1, 0);

DataInputStream input = connection.openDataInputStream();

DataOutputStream output = connection.openDataOutputStream();

output.writeInt(42);

output.writeInt(-42);

output.flush();

LCD.*drawString*("Sent data", 2, 0);

LCD.*drawString*("Waiting ...", 3, 0);

**int** answer = input.readInt();

LCD.*drawString*("# = " + answer, 4, 0);

input.close();

output.close();

connection.close();

LCD.*drawString*("Bye ...", 5, 0);

}

**catch**(Exception ioe)

{

LCD.*clear*();

LCD.*drawString*("ERROR", 0, 0);

LCD.*drawString*(ioe.getMessage(), 2, 0);

LCD.*refresh*();

}

Thread.*sleep*(4000);

}

}