Wireless Connectivity Kit Getting Started Guide

Getting Started Guide

Wireless Connectivity Kit Getting Started Guide (90001456-13C)

Revision	Date	Description
A	April 2015	Initial release.
B	July 2015	Added a feedback form to the Wireless Connectivity Kit. Imported common documentation from the RF common space.
C	April 2016	Updated the documentation based on XKB2-A2T-WWC - Kit, Wireless Connectivity S2C 802.15.4.

Trademarks and copyright

Disclaimers

Information in this document is subject to change without notice and does not represent a commitment on the part of Digi International. Digi provides this document “as is,” without warranty of any kind, expressed or implied, including, but not limited to, the implied warranties of fitness or merchantability for a particular purpose. Digi may make improvements and/or changes in this manual or in the product(s) and/or the program(s) described in this manual at any time.

Warranty

To view product warranty information, go to the following website:

www.digi.com/howtobuy/terms

Customer support

Gather support information: Before contacting Digi technical support for help, gather the following information:

- Product name and model
- Product serial number (s)
- Firmware version
- Operating system/browser (if applicable)
- Logs (from time of reported issue)
- Trace (if possible)
- Description of issue
- Steps to reproduce

Contact Digi technical support: Digi offers multiple technical support plans and service packages. Contact us at +1 952.912.3444 or visit us at www.digi.com/support.

Feedback

To provide feedback on this document, email your comments to

techcomm@digi.com

Include the document title and part number (Wireless Connectivity Kit Getting Started Guide, 90001456-13 C) in the subject line of your email.

Contents

How XBee devices work

XBee transparent mode

Security and encryption

Low power and battery life

XBee API mode

Communication models

Inputs and outputs

Signal strength and radio frequency range

Additional resources

Wireless Connectivity Kit

Digi's Wireless Connectivity Kit is a great way to learn how to use XBee® RF devices for device connectivity and sensor networking. Starting with very simple examples, we provide step-by-step guidance as you assemble the kit components to create reliable device communications, working control systems, and sensing networks with incredible battery life and robust security.

The kit is designed for anyone getting started in the world of XBees. Professional makers, embedded engineers, educators, engineering students, and even young inventors should be able to quickly create wireless communications that work flawlessly.

Each point of this guide explains a basic topic related to XBees through a short theoretical introduction and examples that put into practice the concepts you have learned. The topics are arranged according to their complexity, from the most basic to the more powerful features. We recommend that new users work through them in the order that they appear.

All examples are explained step by step , so you should always feel well supported. Certain examples use the Java programming language. These examples are designed to be easy for anyone to use, and those with some programming background should find it simple to extend them.

Ready to create device systems that are sensitive, active, reactive, and communicative?

Let's get started!


Wireless Connectivity Kit
With XBee® Modules
GET STARTED!
Digi
XBee
www.digi.com

Kit contents

Once you've verified that the kit contains the following components, get started by learning about the XBee modules:

Qty.	Part	Picture of the part
2	XBee Grove Development Boards	
2	XBee 802.15.4 Modules	
2	Micro USB cables	

Qty.	Part	Picture of the part
2	XBee stickers	

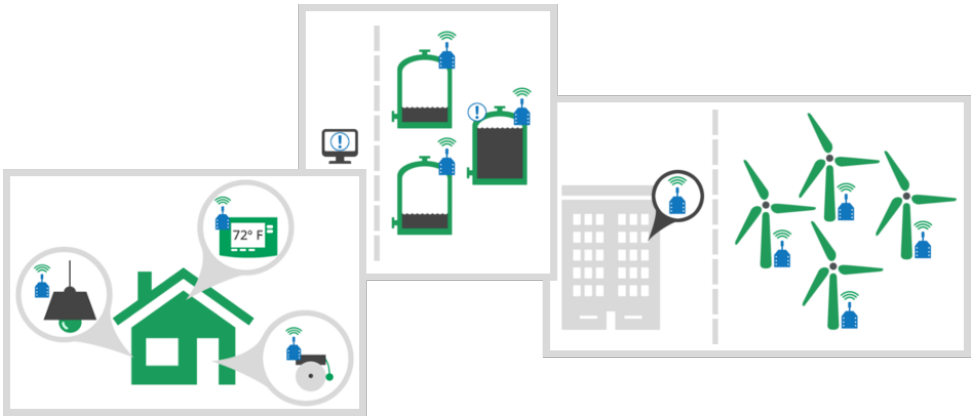
Introduction to XBee devices

XBee modules are small radio frequency (RF) devices that transmit and receive data over the air using radio signals. Wireless capability is essential whenever you want to place sensors where no cables can be installed, or where such tethering is undesirable.

XBee devices are highly configurable and support multiple protocols, which lets you choose the right technology for your application—whether you want to set up a pair of radios to swap data or design a large mesh network with multiple devices.

Here are some of the ways you can use XBee devices:

- Controlling a robot remotely or creating wearable electronics for people, pets, or wildlife, without hindering movement.
- Making a building smarter and more responsive to human interaction.
- Using XBee technology in industrial solutions. For example, XBee devices are used as sensors to monitor industrial tanks for liquid levels, temperature, and pressure, and to monitor and control complex machines such as wind turbines.



Download and install XCTU

XBee Configuration and Test Utility (XCTU) is a multi-platform program that enables users to interact with Digi radio frequency (RF) devices through a graphical interface. The application includes built-in tools that make it easy to set up, configure, and test Digi RF devices.

For instructions on downloading and using XCTU, see the *XCTU User Guide*.

Assemble the hardware

This guide walks you through the steps required to assemble and disassemble the hardware components of your kit.

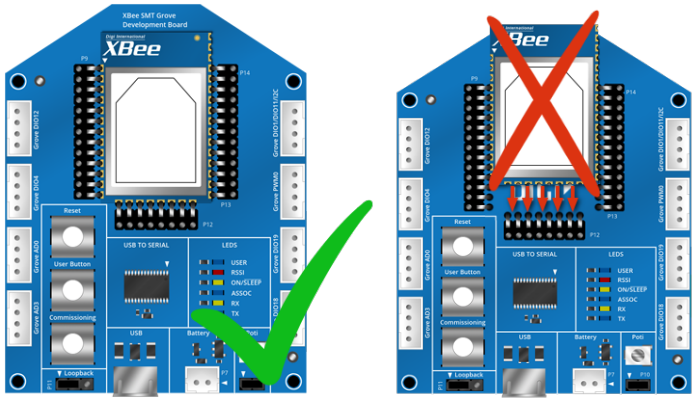
Plug in the XBee module

This kit includes two XBee Grove Development Boards. For more information about this hardware, visit the XBee Grove Development Board documentation.

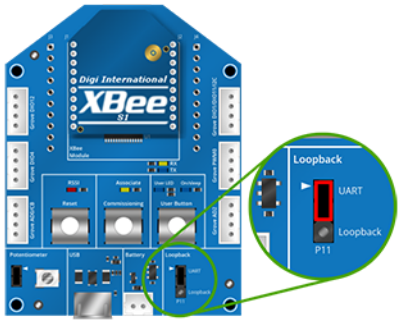
Follow these steps to connect the XBee devices to the boards included in the kit:

1. Plug one XBee 802.15.4 RF Module module into the XBee Grove Development Board.

Make sure the board is NOT powered (either by the micro USB or a battery) when you plug in the XBee module.



2. Once the XBee module is plugged into the board (and not before), connect the board to your computer using the micro USB cables provided.
3. Ensure the loopback jumper is in the UART position.



4. Connect an antenna (if applicable).

How to unplug an XBee device

To disconnect your XBee device from the XBee Grove Development board:

1. Disconnect the micro USB cable (or the battery) from the board so it is not powered.
2. Remove the XBee device from the board socket, taking care not to bend any of the pins.

CAUTION! Make sure the board is **not** powered when you remove the XBee device.

Basic chat

The goal of this first example is to transmit real-time text messages over the air via XBee modules—a wireless chat session.

You will assemble the hardware and connect it to your computer, configure the XBees for wireless communication, and then start chatting. The text you type to one XBee will be wirelessly transmitted to the other XBee, and vice versa.

Follow the steps below to set up the XBees and open a wireless chat session using XCTU.

Note If you get stuck, go to Troubleshooting.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee

module.

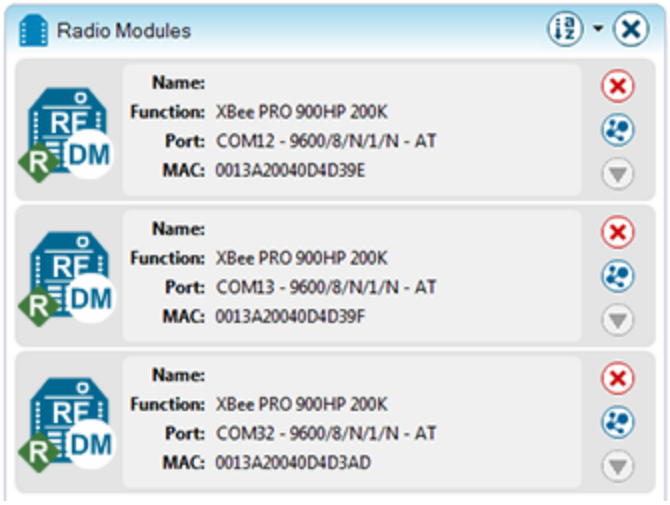
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Add the XBee modules to XCTU

Use XCTU to find your XBee modules and add them to the tool.

1. Click **Discover radio modules** from the toolbar.

2. In the **Discover radio modules** dialog, select the serial port(s) in which you want to look for radio modules. If you do not know the serial ports where your modules are attached, select all ports. Click **Next**.
3. In the **Set port parameters** window, maintain the default values and click **Finish**.
4. As XCTU locates radio modules, they appear in the **Discovering radio modules**… dialog box. Once the discovery process has finished, click **Add selected devices**.
5. At this point, assuming you have two modules connected to your computer, you should see something like this in the **Radio Modules** section on the left:



Note The function, port number and the MAC address displayed for your modules need not match those shown in the picture.

Configure the XBee modules

In order to transmit data wirelessly between your XBees, set the **DH** and **DL** parameters on each module to match the **SH** and **SL** (**SH** + **SL** = MAC address) of the other module. For example, if the MAC

of XBee A is 0013A20012345678, then the **DH** of XBee B should be 0013A200 and the DL 12345678.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

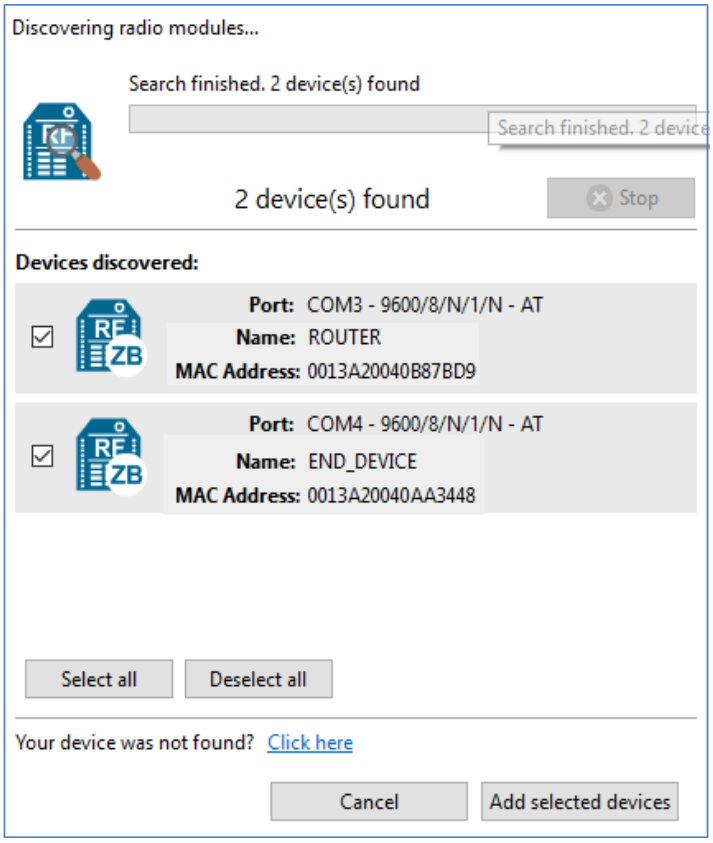
Parameter	XBee A	XBee B	Effect	Parameter
CH	B	B	Defines the frequency to use to communicate. This must be the same for all radios on your network.	CH
ID	2015	2015	Defines the network that a radio will attach to. This must be the same for all radios on your network.	ID
DH	0013A200	0013A200	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number High (SH) of the other module.	DH
DL	SL of XBee B	SL of XBee A	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the other module.	DL
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.	NI

3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

Check the network

Once you have configured your XBee modules, use XCTU to verify that they are in the same network and can see each other.

1. Click the **Discover radio nodes in the same network** button of the first radio module. The device searches for radio modules in the same network.



When the discovery process is finished, XCTU lists discovered devices found within the network in the **Discovering remote devices** dialog. You do not need to add the remote device that has been discovered.

2. Close the dialog by clicking **Cancel**.

Chat!

In order to chat with the other module, use the XCTU console or any serial port terminal application such as CoolTerm or TeraTerm (for Windows only). In this case, we will use the XCTU console for chatting.

Note If you have two computers, connect a module to each computer. You will need XCTU (or any other serial terminal) installed on both computers in order to chat.

To chat through XCTU:

1. If XCTU is not already running, open it on each computer. If you are using only one computer, one instance of XCTU is enough.
2. Switch to the **Consoles** working mode.
   This working mode of XCTU allows you to communicate with the radio modules in the devices list. XCTU loads a list of consoles in the working area—one for each module of the devices list,

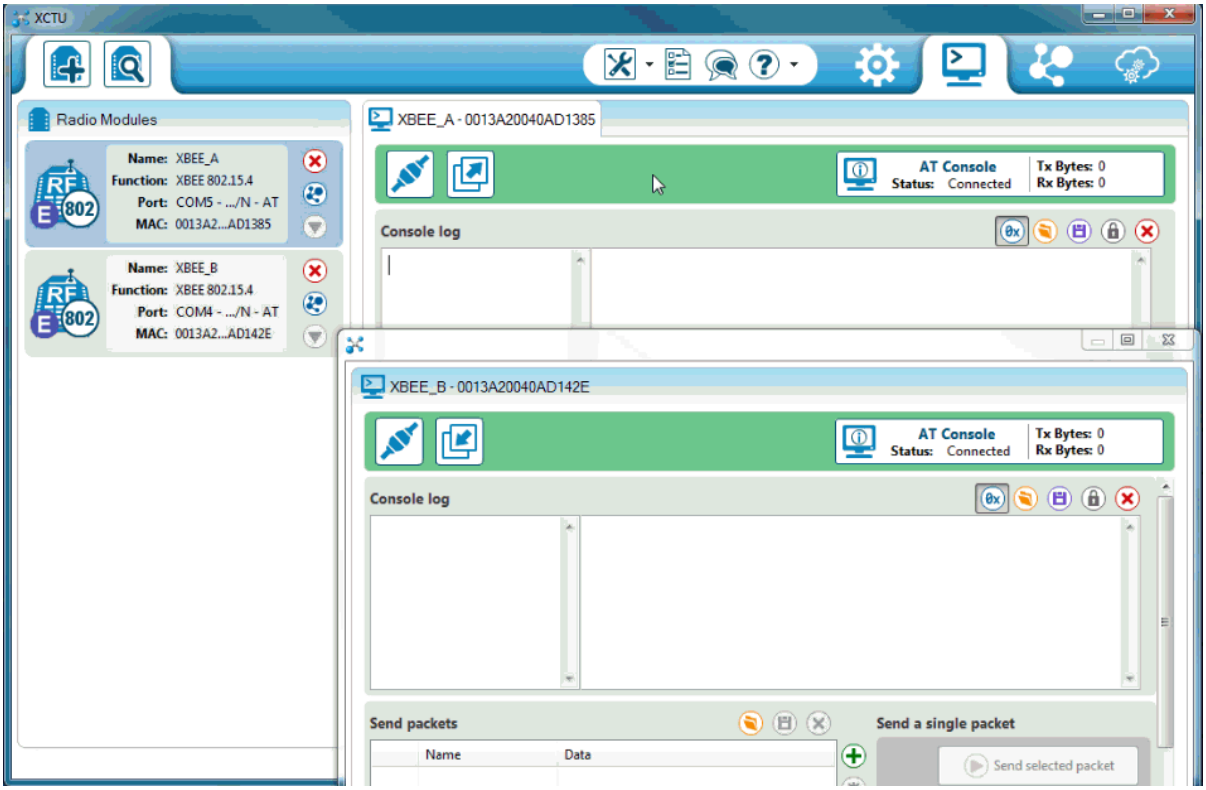
sorted in a tabbed format.

3. If it is not already there, add an XBee to XCTU so it is listed in the Radio Modules list. If you are using two computers, add the module attached to one of the PCs to the corresponding instance of XCTU.
4. Open the serial connection of the radio module: select the XBee in the **Radio Modules** section, and click the **Open serial connection** button.

The background changes to green to indicate that the connection is open.

5. Repeat steps 3 and 4 for the other XBee. If you are using two computers, repeat them in the other PC.
6. If you are using only one computer, you will see the consoles in two tabs. Click the **Detach view** button to see both at the same time.

7. Use the Console log section to type messages. Type **Hello, XBee!** in one of the consoles. The XBee sends every character to the other module and those characters are displayed in the other console.



8. To disconnect, click the Close serial connection button for each console.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

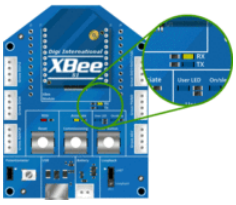
1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as

administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).

5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

> **Tip** To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
        at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.

  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
        at java.lang.Runtime.loadLibrary0(Runtime.java:849)
        at java.lang.System.loadLibrary(System.java:1087)
        at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

- "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
       at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
       at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
       ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar
- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

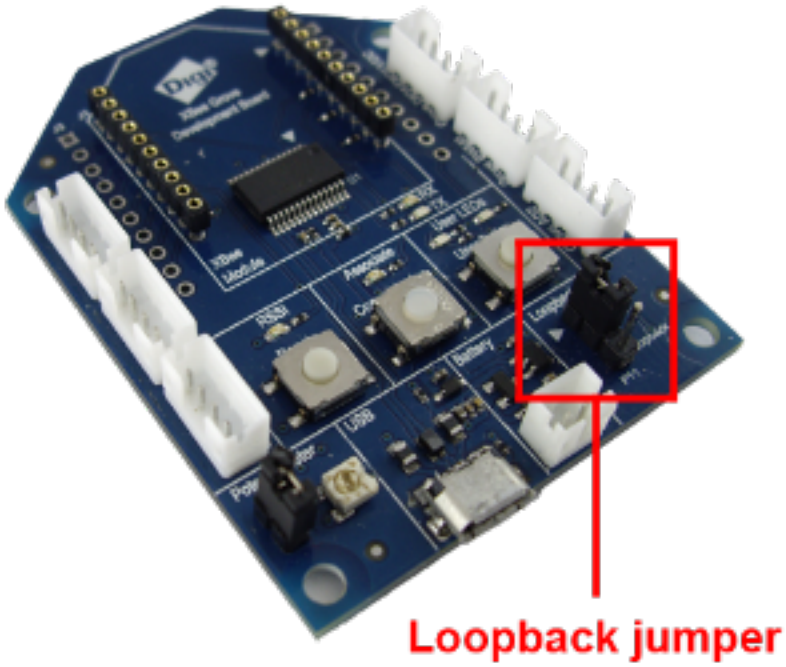
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

How XBee devices work

This section describes how XBee devices communicate, and introduces two communication methods - wireless and serial communication. Both communication types are important in the function of XBee devices.

How XBee devices communicate

XBee devices communicate with each other over the air, sending and receiving wireless messages. The devices only transfer those wireless messages; they cannot manage the received or sent data. However, they can communicate with intelligent devices via the serial interface.

XBee devices transmit data coming from the serial input over the air, and they send anything received wirelessly to the serial output. Whether for communication purposes or simply for configuring the device, a combination of both processes makes XBee communication possible. In this way, intelligent devices such as microcontrollers or PCs can control what the XBee device sends and manage incoming wireless messages.

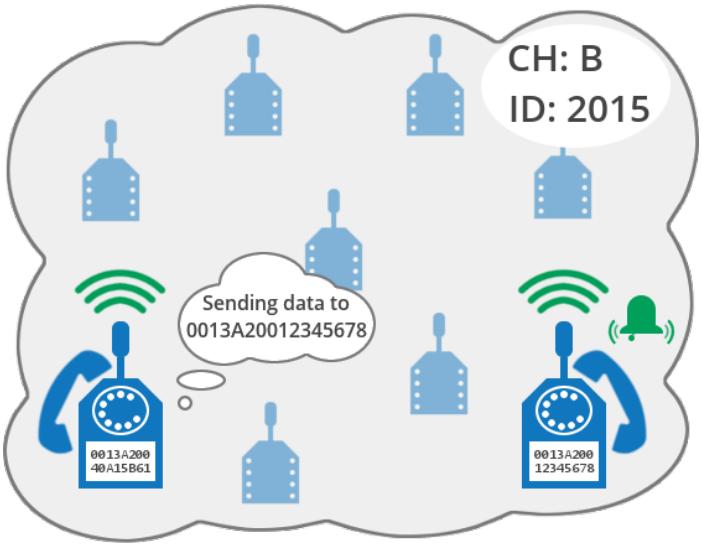
With this information, you can identify the two types of wireless data transmission in an XBee communication process:

1. Wireless communication: This communication takes place between XBee modules. Modules that are supposed to work together need to be part of the same network and they must use the same radio frequency. All modules that meet these requirements can communicate wirelessly with each other.
2. Serial communication: This communication takes place between the XBee module and the intelligent device connected to it through the serial interface.

Wireless communication

Compare XBee wireless communication to a telephone call. You pick up the phone and call one of your friends; here's the analogy as seen in the world of XBees:

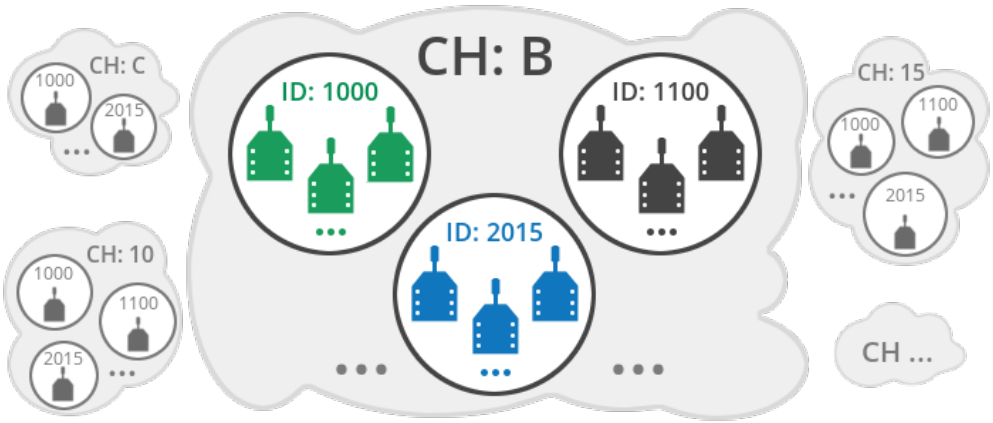
- The caller (you) is the XBee that is sending data.
- The person receiving the call (your friend) is the destination module. The phone line is the wireless link established between the two XBees.
- And the "conversation" between you and your friend is the transmitted data.



The "phone line" in XBee wireless communication is established over the air. The modules transmit and receive information via modulation of waves in the electromagnetic spectrum. In order to transmit data from one XBee to

other, both modules must be on the same frequency and in the same network. This is determined by the value of two parameters:

- Channel (**CH**) tells the XBee the frequency to use to communicate; that is, the channel inside their network. Like when you tune a radio to look for your favorite station, XBees in a network must use the same channel to "listen" to the rest.
- Personal Area Network identifier (**ID**) directs the XBees to talk to each other by establishing, via this unique identifier, that they are all part of the same network.



An XBee module will:

- Only receive data from other XBees within the same network (same ID value) and using the same channel (same CH value).
- Only be able to transmit data to other XBees within the same network (same ID value) and using the same channel (same CH value).

Note In this documentation, "modules in the same network" refer to the fact that they have the same value for their corresponding ID parameter (for example, 2015) and for their CH parameter (for example, B).

Addressing

XBee device addresses are similar to postal and email addresses for people. Some addresses are unique, like an email address, but others are not. For example, several people can live at the same postal address.

Each XBee device is known by several different addresses, each of which serves a purpose.

Type	Example	Unique
64-bit	0013A20012345678	Always
16-bit	1234	Yes, but only within a network
Node identifier	Bob's module	Uniqueness not guaranteed

64-bit address

Every XBee device has a 64-bit address to distinguish it from others and prevent duplicate information. That address (also called MAC) is assigned to Digi by the IEEE and is guaranteed to be **unique**, so two devices cannot have the same address.

You can determine the value of the 64-bit address by reading the Serial Number High (**SH**) and Serial Number Low (**SL**) parameters on any device. It is also printed on the back of the device.

Note The concatenation of **SH** + **SL** forms the 64-bit or MAC address of the device. It is stored in the device's memory as two 32-bit values: the high part, **SH**, and the low part, **SL**. The high part is usually the same for all XBee devices (0013A200), as this is the prefix that identifies Digi devices. The low part is different for every device.

The 64-bit address of **000000000000FFFF** is reserved for sending a broadcast message.

16-bit address

Unlike 64-bit addressing, with 16-bit addressing, each XBee device can be assigned a 16-bit address. The addresses are shorter but uniqueness is not guaranteed because more than one module can have the same 16-bit address.

The value of the 16-bit address can be read or set through the 16-bit Source Address (**MY**) parameter. If the value of this parameter is FFFF, the XBee device disables the reception of packets with 16-bit addresses.

Node identifier

The node identifier is a short string of text that allows users to address the module with a more human-friendly name. In this case, uniqueness is not guaranteed because you can assign the same node identifier to several modules.

You can read or set the value of the node identifier through the Node Identifier (**NI**) parameter.

Serial communication

An XBee module can operate as a stand-alone device or it can be attached to an intelligent device. For example, you can place several battery-powered XBee modules in remote locations to gather data such as temperature, humidity, light, or liquid level.

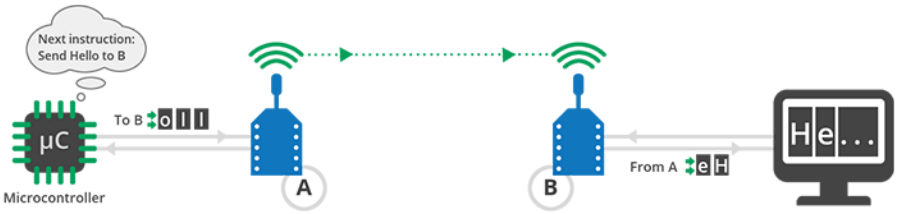
- When operating as a stand-alone device, an XBee module simply sends sensor data to a central node.
- When an XBee module is connected to an intelligent device (such as a computer, Arduino, or Raspberry Pi), it uses serial communication:

- The intelligent device sends data through the serial interface to the XBee module to be transmitted to other devices over the air.
- The XBee module receives wireless data from other devices, and then sends the data through the serial interface to the intelligent device.

The XBee modules interface to a host device such as a microcontroller or computer through a logic-level asynchronous serial port. They use a UART for serial communication with those devices.

For additional information about serial communication, go to the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide or the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

Microcontrollers attached to an XBee module can process the information received by the module and thus monitor or even control remote devices by sending messages through their local XBee module. For prototyping, you can use external microcontrollers such as Arduino or Raspberry Pi, sockets, and breadboards.



The boards included in this kit allow you to use the XBee modules in either mode:

- If you plug the modules into the boards and connect them to a computer or microcontroller using the micro USB cables, you can configure the XBee modules, test the connection, and send/receive data to/from other modules.
- If you plug the modules into the boards and connect them to a battery, the XBee modules work autonomously. For example, they can gather data from a sensor and send it to a central node.

Operating modes

XBee devices can use their local serial connection in very different ways. The "operating mode" establishes the way the host device communicates with an XBee module through the serial interface.

XBee modules support two different operating modes:

- Application Transparent ("transparent mode")
- Application Programming Interface ("API mode")

Application Transparent operating mode

This mode is called "transparent" because the radio passes information along exactly as it receives it. All serial data received by the radio module is sent wirelessly to a remote destination XBee module. When the other module receives the data, it is sent out through the serial port exactly as it was received. Transparent mode has limited functionality but is an easy way to get started with XBee devices.

To learn more about transparent mode, see XBee transparent mode.

API operating mode

Application Programming Interface (API) operating mode is an alternative to transparent mode. In API mode, a protocol determines the way information is exchanged. Data is communicated in packets (commonly called API frames). This mode allows you to form larger networks and is more appropriate for creating sensor networks to perform tasks such as collecting data from multiple locations, controlling devices remotely, or automating your home.

To learn more about API mode, see XBee API mode.

Comparison of transparent and API modes

XBee devices can use transparent or API operating mode to transmit data over the serial interface. You can use a mixture of devices running API mode and transparent mode in a network. The following table provides a comparison of the two modes.

Transparent operating mode	API operating mode
When to use: ■ Conditions for using API mode do not apply.	**When to use:** ■ Sends wireless data to multiple destinations. ■ Configures remote XBee devices in the network. ■ Receives wireless data packets from multiple XBee devices, and the application needs to identify which devices send each packet. ■ Receives I/O samples from remote XBee devices. ■ Must support multiple endpoints, clusters, and/or profiles (for Zigbee modules). ■ Uses Zigbee Device Object (ZDO) services (for Zigbee modules).
Advantages: ■ Provides a simple interface that makes it easy to get started with XBee devices. ■ Easy for an application to support; what you send is exactly what other modules get, and vice versa. ■ Works very well for two-way communication between XBee devices.	**Advantages:** ■ Can set or read the configuration of remote XBee devices in the network. ■ Can transmit data to one or multiple destinations; this is much faster than transparent mode where the configuration must be updated to establish a new destination. ■ Received data includes the sender's address. ■ Received data includes transmission details and reasons for success or failure. ■ Several advanced features, such as advanced networking diagnostics, and firmware upgrades.
Disadvantages: ■ Cannot set or read the configuration of remote XBee devices in the network. ■ Must first update the configuration to establish a new destination and transmit data. ■ Cannot identify the source of received data, as it does not include the sender's address. ■ Received data does not include transmission details or the reasons for success or failure. ■ Does not offer the advanced features of API mode, including advanced networking diagnostics, and firmware upgrades.	**Disadvantages:** ■ Interface is more complex; data is structured in packets with a specific format. ■ More difficult to support; transmissions are structured in packets that need to be parsed (to get data) or created (to transmit data). ■ Sent data and received data are not identical; received packets include some control data and extra information.

XBee transparent mode

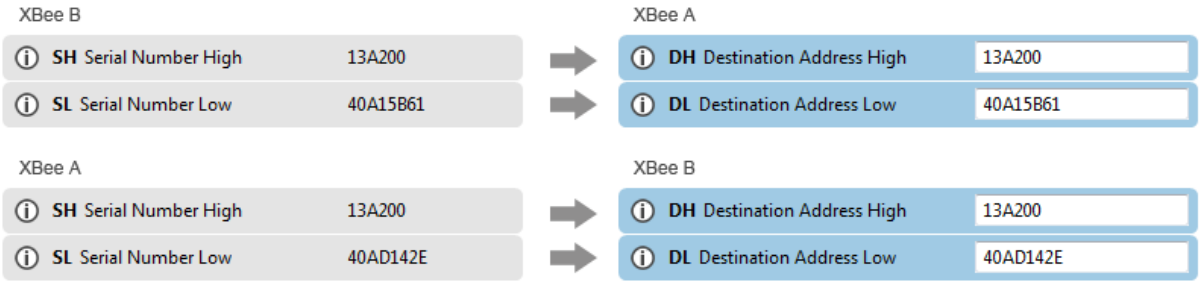
This section provides additional detail about XBee transparent mode. For a comparison of transparent and API modes, see Serial communication.

XBee transparent mode in detail

When operating in transparent mode, an XBee module acts as a serial line replacement. All data received through the serial input is immediately transmitted over the air. When the XBee module receives wireless data, it is sent out through the serial interface exactly at it is received. In fact, communication in transparent mode yields the same result as if the two modules were connected by a wire, but wireless communication makes that physical wire unnecessary.

For two XBee modules to communicate, the sending module needs the address of the recipient. When working in transparent mode, you must configure this address in the module that is communicating. XBee modules can store the complete 64-bit address of the destination module. This address must be programmed in two parameters: Destination Address High (**DH**) and Destination Address Low (**DL**).

If you want modules A and B to communicate, configure the destination address (**DH** + **DL**) of XBee A as the MAC address (**SH** + **SL**) of XBee B, and vice versa.



Transparent mode has some limitations. For example, when you are working with several modules, you must configure the destination before sending each message. However, transparent mode provides an easy way to get started with XBee devices for the following reasons:

- Operation is very simple.
- What you send is exactly what the other modules get.
- Compatible with any device that can communicate over a serial interface.
- Works very well when facilitating communication between two XBee modules.

What have you learned?

- An XBee communicates remotely with other XBees via wireless and locally with the intelligent device (microcontroller, computer) connected to it via the serial interface.
- To communicate wirelessly, your modules must be part of the same network, so the **ID** and **CH** parameters must have identical values for all XBees in the network.
- Every XBee module has a unique 64-bit address called MAC that distinguishes it from the rest of the devices. This address is formed by the concatenation of the parameters **SH** (Serial Number High) and **SL** (Serial Number Low).
- The operating mode of an XBee establishes the way to communicate with the module through the serial interface.

- There are two different operating modes: Transparent and API (Application Programming Interface).
- Transparent mode can be used as a serial cable replacement. What is sent through an XBee serial input is wirelessly received by the destination module and then sent out to its serial output exactly as it was transmitted from the first XBee (and vice versa).
- In order to communicate using transparent mode, you must pre-configure each of your devices by setting the parameters **DH** and **DL** on the first module with the **SH** and **SL** values of the other one respectively, and vice versa.

Extend the basic chat example

If you're ready to move beyond this exercise and extend the example, try the following:

- Use Arduino or Raspberry Pi instead of a computer to transmit data wirelessly.
- Use your XBees as a cable replacement for your serially communicating applications. For example, if you have an Arduino application that controls room lighting via the serial port, replace the serial cable with XBees and move your Arduino around the house.
- Try using XBees to send wireless messages to your friends and neighbors.

Command mode

An XBee device in Transparent mode simply passes information along exactly as it receives it. So, what you send is what other devices get. But sometimes you want to talk directly to the local device without sending data. For example, you may need to modify its configuration or alter the way it behaves. In that case, the XBee device needs to know that this communication should not be transmitted wirelessly.

Command mode is a state in which incoming characters are interpreted as commands. To get a device to switch into this mode, you must issue a unique string of text in a special way: **+++**. When the device sees a full second of silence in the data stream followed by the string **+++** (without Enter or Return) and another full second of silence, it knows to stop sending data through and start accepting commands locally.

Guard time silence	Command sequence	Guard time silence
One second before	+++	One second after

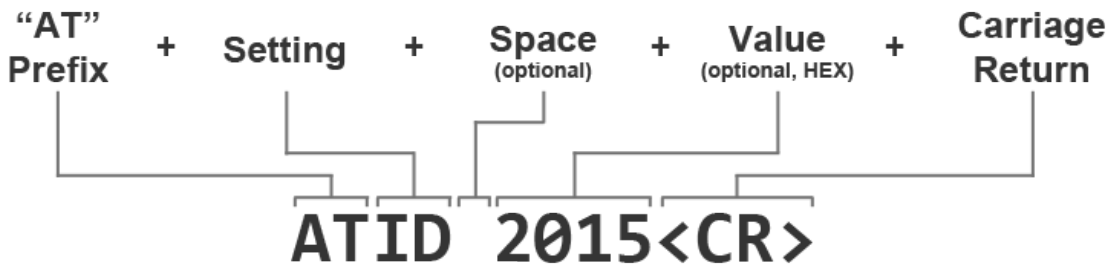
Do not press Return or Enter after typing the **+++** because it will interrupt the guard time silence and prevent the module from entering Command mode.

Once the device is in Command mode, it listens for user input for a while. If 10 seconds go by without any user input, the device automatically drops out of Command mode and returns to Transparent mode.

AT commands

The purpose of command mode is to read or change the configuration of the local XBee device. Every module has a number of settings, like channel or network ID, that define its behavior. These settings are identified by two characters, for example, **CH** for channel, and **ID** for network ID.

When you want to read or set any setting of the XBee **module**, you must send it an AT command. Every AT command starts with the letters "AT" followed by the two characters that identify the command being issued and then by some optional configuration values.



For example, to read and set the network ID setting:

```
// Enter command mode
+++OK

// Read the ID setting
ATID <Enter>
0

// Change the ID setting
ATID 2015 <Enter>
OK
```

Basic AT commands

- **AT**

  This command checks the connection with the module. This is like asking "Are you there?" and the device replying "Yes." When you send this command, the module simply replies OK. If you don't see an OK in response, you have probably timed out of command mode. Type the **+++** to go back into it.

- **ATCN**

  This command explicitly exits the module from command mode. Remember that if you don't type anything for 10 seconds, the device automatically drops out of Command mode.

- **ATWR**

  This command writes the current configuration to non-volatile memory so that it persists the next time the device powers up. Otherwise, parameters are restored to previously saved values after the device is reset.

For additional information about serial communication, go to the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide or the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

Use AT commands

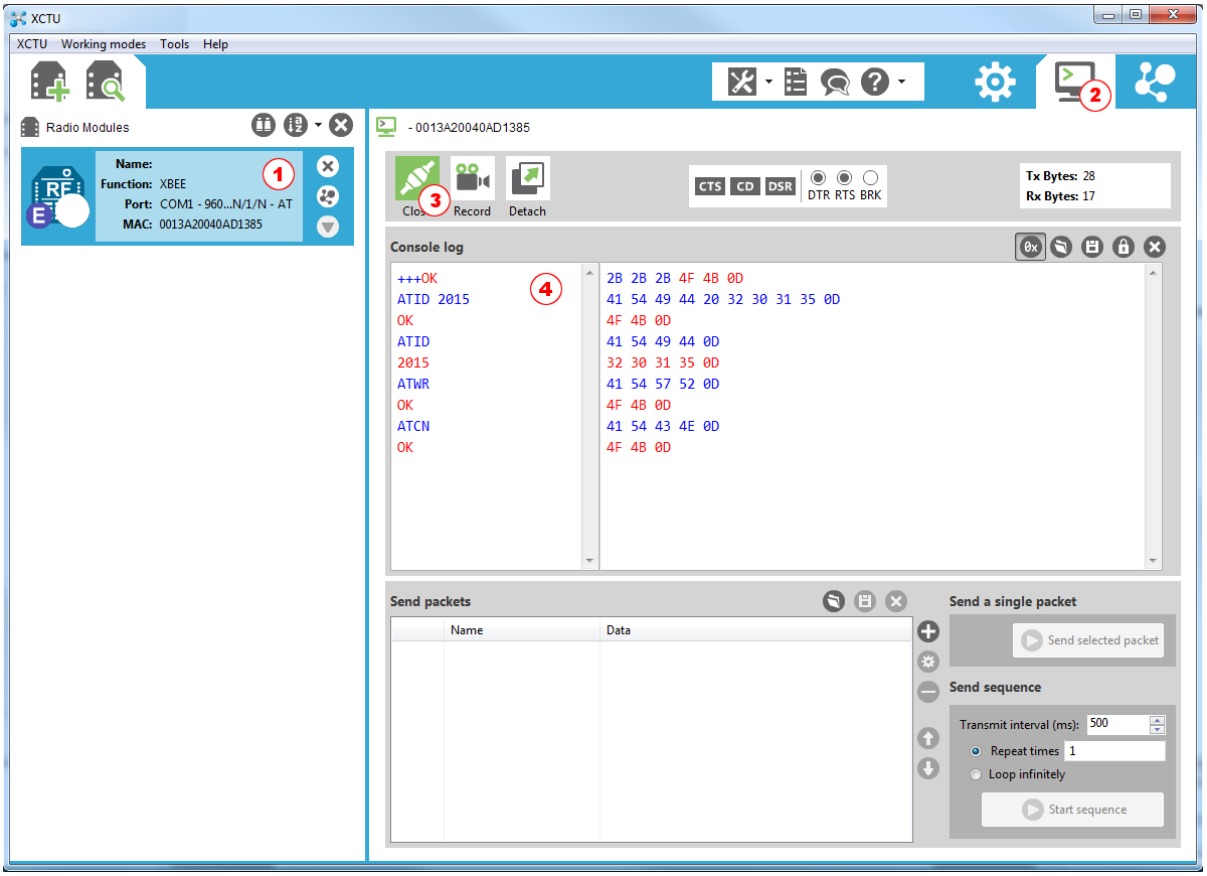
In the first example in this kit, you used XCTU to configure some settings of each of your modules, such as the network ID. XCTU uses AT commands in the background to read and set the settings. For example, when you changed the value of that parameter and clicked the Write button, XCTU went into command mode using **+++,** changed the value of the setting with the **ATID** command, wrote the setting with the **ATWR**command, and finally exited command mode with the `ATCN` command.

XCTU simplifies the configuration of the XBee modules so you don't have to use command mode or AT commands to configure them. However, you can always configure an XBee module through any serial port terminal application or the XCTU console.

The following example demonstrates how you can perform some of the configuration steps outlined in the first lab but via command mode and using AT commands:

1. In the Consoles working mode of XCTU, click the **Open the serial connection with the radio module** button.
2. Use **+++** to enter into command mode and wait for an OK response.
3. To set a register, type an AT command followed by the value you want to set; for example, **ATID 2015**; followed by a Return.
4. To read a register, type an AT command; for example, **ATID**; followed by a Return.
5. Use the **ATWR** command to write the new configuration to the module's memory.
6. Exit command mode with the **ATCN** command.

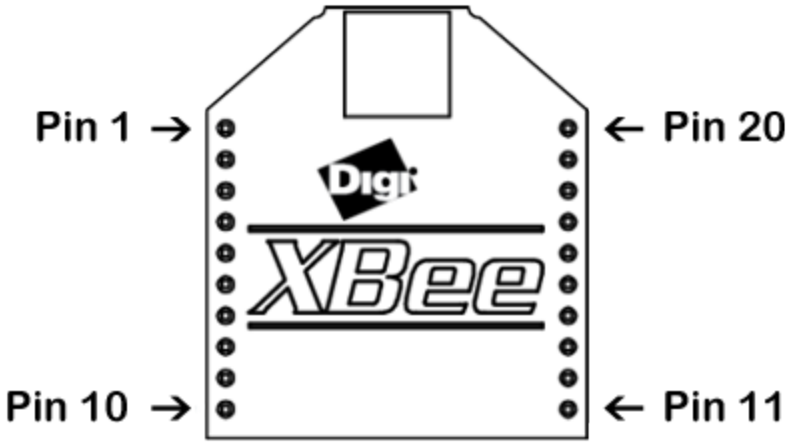
Note You should get an **OK** response after issuing each command to set parameters, write the changes, or exit from command mode. If not, you most likely took more than 10 seconds to issue the command and you have dropped out of command mode.



Pin pairing

All XBee modules have a set of pins which can be used to connect sensors or actuators and configure them for specific behavior. Each XBee radio has the capability to directly gather sensor data and transmit it without the use of an external microcontroller. The Legacy 802.15.4 modules provided in this kit have nine input/output pins, six of which can read analog values, the S2C 802.15.4 modules have four pins that can read analog values.

With these pins you can, for example, turn on a light by sending information to an XBee connected to an actuator, or measure the outside temperature by obtaining data from a temperature sensor attached to your XBee.



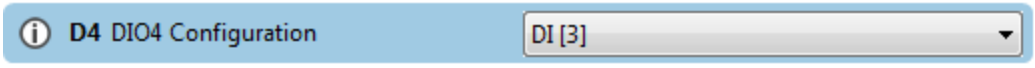
The term **pin pairing** (also known as **I/O line passing**) refers to direct input/output communication between two XBees. Pin pairing virtually links the pins of one XBee directly to the pins of another XBee. Imagine a wireless doorbell: press the button (input) and the doorbell rings (output).

The main advantage of pin pairing is that—because output is directly linked to input—you can change the value of the output without using a computer or microcontroller.

To make a pin pairing, set a pin on one XBee as input and set the matching pin on the other XBee as output. For example, to make a pin pairing with pin 11 (DIO4/AD4), perform these steps:

- In 'XBee A' (input):

1. Set the DIO4 to digital input by changing the value of the D4 parameter to DI [3].



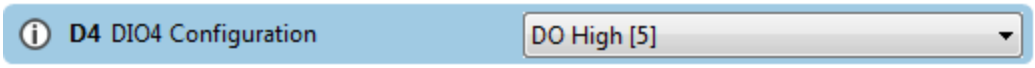
2. Configure the module to send automatic notifications when the state of the DIO4 changes. To do so, change the value of the DIO Change Detect (IC) parameter to 10 (see How XBee devices

get sensor data).

3. Set the destination address (DH + DL) to match the MAC address (SH + SL) of XBee B.

- In 'XBee B' (output):

1. Set the same DIO as digital output. To do so, change the value of the D4 parameter to DO Low [4] or DO High [5].



2. Set the address of the transmitting module. The I/O Input Address (IA) parameter tells which XBee to listen to, this effectively binds the outputs to a particular module's inputs. Assuming that the MAC address (SH + SL) of 'XBee A' is 0013A20040D2A82E, set the IA parameter as follows to connect XBee B outputs to XBee A inputs:

Congratulations! You have just completed the section, so you are ready to do the Example: Pin pairing.

Example: Pin pairing

In this example you will use the knowledge acquired in the previous section to create your first stand-alone wireless system. The project consists of two parts: a push button connected to one XBee, and an LED connected to the other. When you push the button of one XBee, the LED of the other XBee will light.

Tip If you get stuck, go to Troubleshooting.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables

Software

- XCTU 6.1.3 or later

For further information about XCTU, go to:
http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee modules

To make a pin pairing, you will configure the pin of the XBee where the button is connected as digital input, and configure the pin of the XBee where the LED is connected as digital output. You will also need to configure the first XBee to send a notification to the other XBee when the button changes state.

When you press the button of the first XBee, the LED of the other will light.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

Parameter	XBee A	XBee B	Effect
CH	B	B	Defines the frequency to use to communicate. This must be the same for all radios on your network.
ID	2015	2015	Defines the network that a radio will attach to. This must be the same for all radios on your network.
DH	0013A200	—	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number High (SH) of the other module.
DL	SL of XBee B	—	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the other module.
MY	FFFF	FFFF	Enables the reception of packets with 64-bit addresses.

Parameter	XBee A	XBee B	Effect
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.
D4	DI [3]	DO High [5]	Configures the pins: Digital Input in XBee A (button) and Digital Output in XBee B (LED).
IC	10	—	Configures the XBee A to transmit an Input/Output (I/O) packet when pin DIO4 changes.
IA	—	SH + SL of XBee A	Defines the address of the transmitting module.

Note The dash (**—**) in the table means to keep the default value. Do not change the default value.

3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

Try it

Press the button where XBee A is connected. If everything is set up properly, the LED of the XBee B board will light.

What have you learned?

In this section, you have learned that:

- You can create direct input/output communication between two XBees as you did in this example. In order to do this, you must set a pin as input on one module and designate the same pin as output on the other device.
- If you set the DIO Change Detect (**IC**) parameter on the module that has the Digital IO configured as digital input, it will send a wireless package to the other device every time this input state changes (for example, when a connected button is pressed or released). This package switches the status of the corresponding pin configured as digital output, for example to turn a connected LED on or off.
- The I/O Input Address (**IA**) parameter binds an XBee output to a specific source address. This means that the digital outputs (for example, an LED) of the XBee configured with the **IA** parameter will only change their status if the module receives a change IO data packet from the **IA** configured address' module (when the button is pressed).

Extend the pin pairing example

If you're ready to move beyond this exercise and extend the example, try the following:

- Create a real doorbell. Instead of the LED in XBee B, connect a buzzer or bell. When someone presses the button, the bell will ring.
- Create a basic home alarm system. Connect a motion sensor to XBee A and place it near your main door. Connect a buzzer to XBee B and place it in another room. When movement is detected, the buzzer will alert you.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

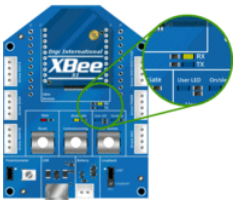
1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.

  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

- "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
       at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
```

```
       at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
       ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).

3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

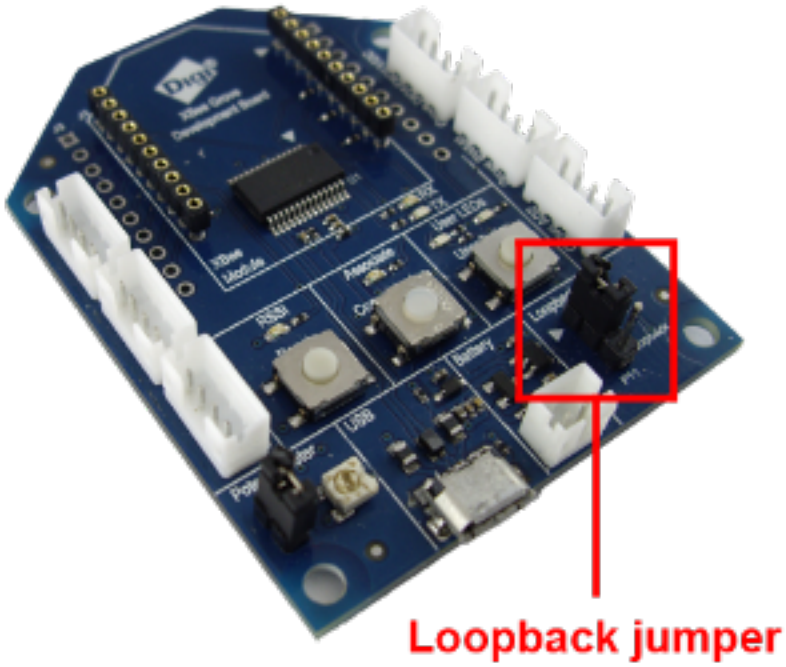
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

Security and encryption

Sometimes the information transmitted in an XBee network needs to be protected. For example, an XBee network transferring financial information must be carefully protected against external agents. XBee modules can be configured for secure communication via encryption keys.

Note When you enable security on a device, the information you transmit is encrypted before it is sent. Likewise, information you receive must first be decrypted in order to be readable. Consider your project requirements before enabling security on an XBee network, because this encryption/decryption process can result in a slight increase in both latency and packet size.

How to enable network security

To enable secure communication, configure the following parameters with the same value in all the devices of the network.

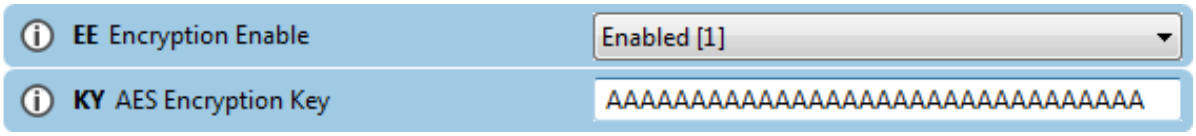
1. Set the AES Encryption Enable (**EE**) parameter to 1.
2. Set the AES Encryption Key (**KY**) parameter to any 32 hexadecimal character string. Setting this parameter enables the encryption/decryption handshake. Once this parameter has been set, it is impossible to retrieve the actual value.

Example: Basic (but secure) communication

In this example, you will add a security level in the basic chat by encrypting communication between the two XBee modules. Note that this feature is applicable for both AT and API operating modes.

If you get stuck, go to Troubleshooting.

First, follow the steps explained in each example. When you have added the modules to XCTU and changed the value of the corresponding settings, enable security on each module. To do so, set the EE and KY parameters as follows and write the new values:



Once you have done this, you can chat securely.

Test your encryption

Once you have completed the previous steps, try these methods to see how it's working:

1. Send a message from SENDER to the other two XBee devices.

   You will see that the message was received correctly, but because of the encryption/decryption process, you won't notice any difference in the way the XBee modules display the information.
2. Now try enabling the encryption on one device but not on the others. Or, enable encryption on the three devices but with a different key in one of them.

   In both of these instances, the receiver modules will not receive the data.

Additional recommendations

Many external agents can compromise the integrity of your security configuration. To maintain the integrity of your secure network, remember the following:

- Safeguard the value of your **KY** parameter. Do it virtually, but physically as well. Most communication security holes are due to inadequate information protection.
- The key you choose should not be easy to guess. While an encryption key can be any 32-bit hexadecimal value, you can create a more complex key by combining hexadecimal characters.
- If you are developing an application that performs radio configuration, do not set the value of the encryption key (**KY** parameter) programmatically. Instead, use XCTU to set the value directly on the module. Otherwise, the key could be obtained by anyone reading the data stream on the serial port.

Low power and battery life

This section introduces the key concepts you need to know to take advantage of the power saving capabilities of XBee devices. It also provides a lab that lets you put the concepts to work and see the results.

Low power devices and battery life

Wireless RF devices are typically battery-powered, which can pose a challenge: depending on the location of the device, it may be difficult or expensive to replace the battery.

XBee modules are low-power devices. They can put themselves into a temporary sleep state in which they consume virtually no current. During sleep, the device is almost completely turned off and is sometimes incapable of sending or receiving data until it wakes up.

A real world scenario

Extending battery life is important in many real world scenarios. For example, if you had several greenhouses, each with a temperature sensor connected to an XBee module, battery life would be critical. Fully charged batteries would only power the modules for one day.

There are several ways to maximize battery life. For example:

- Putting the modules into a cycle where they sleep for one second and then wake for one second before sleeping again could double the battery life to two days.
- Cyclically sleep for 59 seconds and then waking for one second might keep the same batteries going for 60 days. Taking this further, you could potentially extend the battery life for years.

Design considerations for applications using sleep mode

Before using sleep mode you must take into consideration the structure of your project and your XBee network. Some applications, like the greenhouse example, are particularly suited to sleep mode. In that scenario, the modules only send data periodically and are not expected to receive data. The modules can therefore be sleeping most of the time and wake up only to send the temperature value.

By contrast, there are applications where devices send and receive data frequently. These applications need to use sleep mode differently—for example, by triggering sleeping and waking with events. Continue learning about sleep mode so you can optimize its capabilities for your needs.

Configure sleep

For a basic sleep configuration you only need to set a few parameters::

- **SM** (Sleep Mode)
- **ST** (Time before sleep)
- **SP** (Cyclic sleep period)

Sleep Mode (SM)

The 802.15.4 protocol contains four basic sleep behaviors which can be divided into two categories: pin-controlled sleep modes and cyclic sleep modes. By default, sleep mode is disabled (**SM** = 0) so the radio will always be awake.

Tip Once you enable sleep mode, XCTU may display a warning message if you try to communicate with the module while it is sleeping (for example, to change the value of a setting). Press the reset button to wake it up.

Pin-controlled sleep mode

The Pin Hibernate (**SM** = 1) mode is controlled by the Sleep_RQ pin. In this mode, the XBee module will sleep when the Sleep_RQ (pin 9) is asserted or pulled high by connecting it to 3.3 volts. For the typical power-down current and wake-up times, see the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide or the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

Cyclic sleep modes

XBees have the ability to sleep and wake on a fixed schedule. This is the most common method of conserving power when a radio is being used as a simple sensor node. Within the cyclic sleep category, XBees offer two modes:

- Cyclic Sleep Remote (**SM** = 4) mode allows modules to periodically check for RF data.
- Cyclic Sleep Remote with Pin Wake-up (**SM** = 5) is the same as Cyclic Sleep Remote but with the option of also waking the module using physical pin 9.

In these two modes, you must configure the Time Before Sleep (**ST**) and Cyclic Sleep Period (**SP**) settings (described below). For the typical power-down current and wake-up times, see the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide or the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

Time before sleep (ST)

Defines the period of inactivity of the module (during which no data is sent or received) before returning to cyclic sleep. If the XBee is transmitting or receiving a message, it will not go to sleep. This setting is only applicable if the sleep mode is 4 or 5.

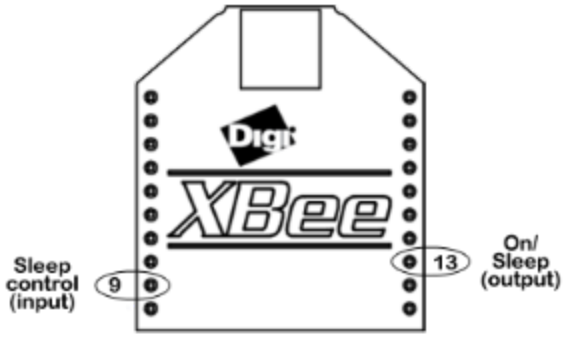
Cyclic sleep period (SP)

Defines the length of time an XBee remains asleep. This setting is only applicable if the sleep mode is 4 or 5.

For bidirectional communication with sleep modes, other protocols such as Zigbee or DigiMesh may provide more appropriate functionality.

Sleep-related pins

The XBee modules have two special pins related to the sleep mode: pin 9 and pin 13.



The Sleep control or sleep request pin, physical pin 9 on the XBee, is used in pin-controlled sleep modes to put the module to sleep when it is asserted or pulled high by connecting it to 3.3 volts.

The On/Sleep pin, physical pin 13, goes high when the module is awake and low when it is sleeping. Generally, this pin is connected to an LED as a visual indication of the module's current sleep state or to a microcontroller input to wake up the control unit and/or provide a simple way for the control unit to know if the radio is awake or sleeping.

Congratulations! You have just completed the section, so you are ready to do the Example: enable sleep mode.

Example: enable sleep mode

In this example, you will learn how to use sleep mode to extend the battery life of an XBee module in the context of the greenhouse example explained earlier. Note that since the kit does not include a temperature sensor, you will use the potentiometer that is already connected to the board. One of the modules will sleep for nine seconds, wake for one second to obtain and send the temperature value, and repeat that pattern. The cycle will continue until sleep mode is deactivated.

Note If you get stuck, go to Troubleshooting.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee modules

XBee B will be the remote module connected to the potentiometer. This XBee will sleep for nine seconds and then wake up to send the value of the potentiometer to XBee A (receiver).

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.

Parameter	XBee A	XBee B	Effect
CH	B	B	Defines the frequency used to communicate. This parameter must be the same for all the radios on your network.
ID	2015	2015	Defines the network that a radio will attach to. This parameter must be the same for all the radios on your network.
DH	—	0013A200	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number High (SH) of the other module.
DL	—	SL of XBee A	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the other module.
MY	FFFF	FFFF	Enables the reception of packets with 64-bit addresses.
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.
SM	—	Cyclic sleep remote [4]	Defines the sleep mode of the module. If you set the module to No Sleep (**SM**=0) it does not sleep. If you set the module to Cyclic Sleep Remote (**SM**=4), it sleeps and wakes on a fixed schedule as defined by the **ST** and **SP** commands.
ST	—	3E8	Defines the period of inactivity before going to sleep. 3E8 (hexadecimal) = 1000 (decimal) x 1 = 1 second.

Parameter	XBee A	XBee B	Effect
SP	—	384	Defines the duration of time spent sleeping. 384 (hexadecimal) = 900 (decimal) x 10 = 9 seconds.
AP	API enabled [1]	API enabled [1]	Enables the API operating mode.
D3	—	ADC [2]	Sets the DIO3/AD3 pin as ADC in the XBee B.

2. Use XCTU to configure the following parameters:
3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

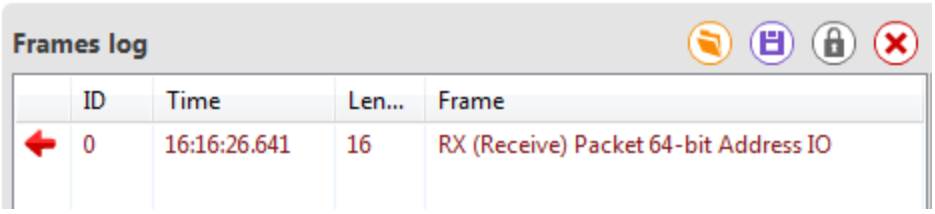
Sleep

With this configuration, XBee B sends the value of the potentiometer to XBee A every time it wakes up. To verify, perform the following steps in XCTU:

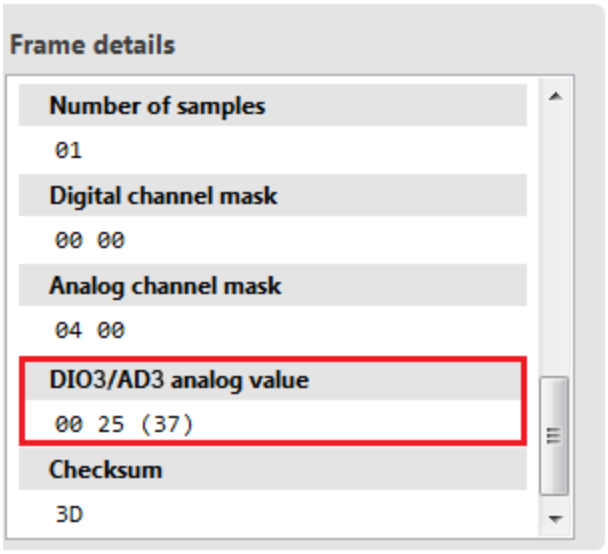
1. Select XBee A (receiver).
2. Switch to the Consoles working mode.

3. Open the serial connection with the module.

4. Check that the module receives an RX (Receive) Packet 64-bit Address IO frame every 10 seconds.



5. Select one frame and check its details in the right panel. You will see the value of the potentiometer (DIO3_AD3) and other details related to the frame.



6. Rotate the potentiometer of the board where XBee B is attached and check that the new packets that arrive contain different values for the DIO3_AD3 pin.

Note Don't forget to close the serial connection with the module when you finish the example.

What have you learned?

In this section, you have learned that:

- Modules with 802.15.4 protocol, as well as others, can go into a temporary sleep state in which they consume virtually no current. To configure your module to go to sleep, you must configure the following parameters:
  - Sleep Mode (**SM**):
    - Pin-controlled sleep mode (SM = 1) pull high pin 9 by connecting it to 3.3 volts to put the module to sleep.
    - Cyclic sleep modes (SM = 4 and SM = 5) enable the module to sleep and wake up on a fixed schedule. These modes need the **ST** and **SP** parameters to be configured.
  - Time before Sleep (**ST**) is the period of time during which no data is sent or received (while the module is awake) before returning to cyclic sleep. This parameter is only applicable for Cyclic sleep modes.
  - Cyclic Sleep Period (**SP**) is the length of time an XBee remains asleep. This parameter is only applicable for Cyclic sleep modes.
- While the XBee is in sleep mode, there is no data transmission or reception. So, if you try to communicate with the module when it is asleep using XCTU, you will receive a warning message saying that the module must be reset in order to wake up.
- Pins 9 and 13 are related to the sleep modes: pin 9 is used to put the module to sleep, and pin 13 is used to determine the sleep state of the device.

Step 6: Extend the example

If you are ready to move beyond this exercise and extend the example, try the following:

- Combine this feature with a real sensor to create a low-power sensor network.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

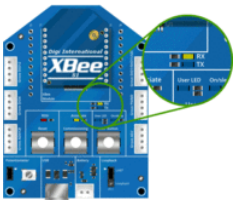
1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.
Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.

  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

- "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
       at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
```

```
       at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
       ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).

3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

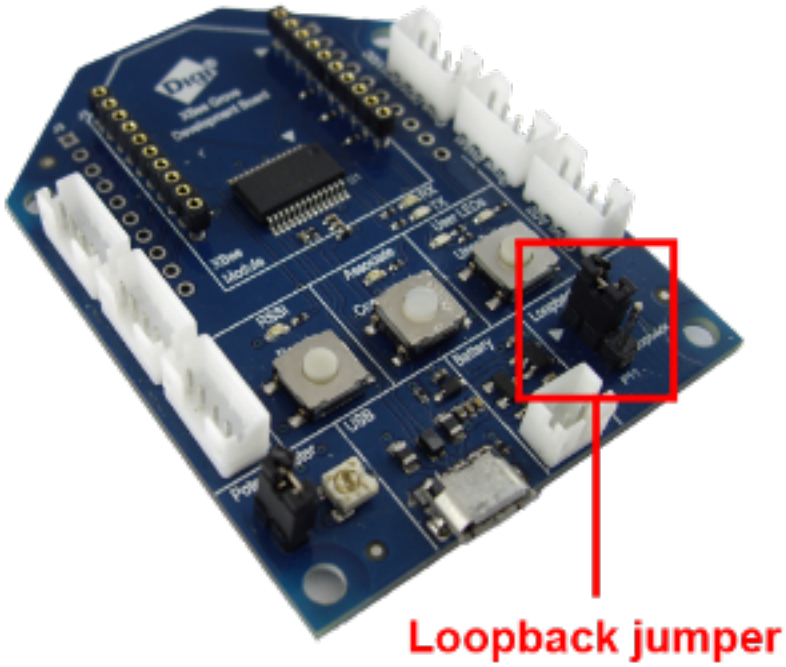
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

XBee API mode

This section provides additional detail about API mode and lets you put your knowledge into practice. For a comparison of transparent and API modes, see Serial communication.

API mode in detail

API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between devices without having to define your own protocol.

By default, XBee devices are configured to work in transparent mode: all data received through the serial input is queued up for radio transmission and data received wirelessly is sent to the serial output exactly as it is received, with no additional information.

Because of this behavior, devices working in Transparent mode have some limitations:

1. To read or write the configuration of an device in Transparent mode, you must first transition the device into Command mode.
2. If a device needs to transmit messages to different devices, you must update its configuration to establish a new destination. The device must enter Command mode to set up the destination.
3. A device operating in Transparent mode cannot identify the source of a wireless message it receives. If it needs to distinguish between data coming from different devices, the sending devices must include extra information known by all the devices so it can be extracted later. To do this, you must define a robust protocol that includes all the information you think you need in your transmissions.

To minimize the limitations of the transparent mode, devices provide an alternative mode called Application Programming Interface (API). API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between modules without having to define your own protocol.

API mode provides a much easier way to perform the actions listed above:

1. Since there are different frames for different purposes (such as configuration and communication), you can configure a device without entering Command mode.
2. Since the data destination is included as part of the API frame structure, you can use API mode to transmit messages to multiple devices.
3. The API frame includes the source of the message so it is easy to identify where data is coming from.

Advantages of API mode

- Configure local and remote XBee devices in the network.
- Manage wireless data transmission to one or multiple destinations.
- Identify the source address of each received packet.
- Receive success/failure status of each transmitted packet.

- Obtain the signal strength of any received packet.
- Perform advanced network management and diagnosis.
- Perform advanced functions such as remote firmware update, ZDO, ZCL and so on.

API frame structure

The structured data packets in API mode are called frames. They are sent and received through the serial interface of the device and contain the wireless message itself as well as some extra information such as the destination/source of the data or the signal quality.

When a device is in API mode, all data entering and leaving the module through the serial interface is contained in frames that define operations or events within the device.

An API frame has the following structure:

Start delimiter	Length		Frame data								Checksum
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API-specific structure								Single byte

Note MSB represents the most significant byte, and LSB represents the least significant byte.

Any data received through the serial interface prior to the start delimiter is silently discarded by the XBee. If the frame is not received correctly, or if the checksum fails, the data is also discarded and the module indicates the nature of the failure by replying with another frame.

Start delimiter

The start delimiter is the first byte of a frame consisting of a special sequence of bits that indicate the beginning of a data frame. Its value is always 0x7E. This allows for easy detection of a new incoming frame.

Length

The length field specifies the total number of bytes included in the frame data field. Its two-byte value excludes the start delimiter, the length, and the checksum.

Frame data

This field contains the information received or to be transmitted. Frame data is structured based on the purpose of the API frame:

Start delimiter	Length		Frame data								Checksum
			Frame type	Data							
1	2	3	4	5	6	7	8	9	...	n	n+1

Start delimiter	Length		Frame data		Checksum
			Frame type	Data	
0x7E	MSB	LSB	API frame type	Frame-type-specific data	Single byte

Note MSB represents the most significant byte, and LSB represents the least significant byte.

- **Frame type** is the API frame type identifier. It determines the type of API frame and indicates how the information is organized in the Data field.
- **Data** contains the data itself. The information included here and its order depends on the type of frame defined in the Frame type field.

Checksum

Checksum is the last byte of the frame and helps test data integrity. It is calculated by taking the hash sum of all the API frame bytes that came before it, excluding the first three bytes (start delimiter and length).

Note Frames sent through the serial interface with incorrect checksums will never be processed by the module and the data will be ignored.

Calculate the checksum of an API frame

1. Add all bytes of the packet, excluding the start delimiter 0x7E and the length (the second and third bytes).
2. From the result, keep only the lowest 8 bits.
3. Subtract this quantity from 0xFF.

Example: Checksum calculation

To calculate the checksum for the given frame:

Start Delimiter	Length		Frame Data															Checksum
			Frame type	Data														
7E	00	0F	17	01	00	13	A2	00	40	AD	14	2E	FF	FE	02	44	42	-

1. Add all bytes excluding the start delimiter and the length: 17 + 01 + 00 + 13 + A2 + 00 + 40 + AD + 14 + 2E + FF + FE+ 02 + 44 + 42 = 481
2. From the result, keep only the lowest 8 bits: 81.
3. Subtract that result from 0xFF: FF - 81 = 7E

In this example, 0x7E is the checksum of the frame.

Verify the checksum of a given API frame

1. Add all bytes including the checksum (do not include the delimiter and length).
2. If the checksum is correct, the last two digits on the far right of the sum will equal FF.

Example: Checksum verification

In our example above, we want to verify the checksum is 7E.

Start Delimiter	Length		Frame Data															Checksum
			Frame type	Data														
7E	00	0F	17	01	00	13	A2	00	40	AD	14	2E	FF	FE	02	44	42	7E

1. Add all data bytes and the checksum: 17 + 01 + 00 + 13 + A2 + 00 + 40 + AD + 14 + 2E + FF + FE + 02 + 44 + 42 + 7E = 4FF
2. Since the last two far right digits of 4FF are FF, the checksum is correct.

Supported frames

Support for API frame types depends on the type of XBee you are using. For the complete list of frames that the XBee 802.15.4 RF Modules support, see the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide and the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

Operating mode configuration

The API Enable (**AP**) parameter configures the XBee module to operate using a frame-based API instead of the default Transparent mode. It allows you to select between the two supported API modes and the default transparent operation.

Mode	AP value	Description
Transparent	0	API modes are disabled and the module operates in transparent mode
API 1	1	API mode without escaped characters
API 2	2	API mode with escaped characters

The only difference between API 1 and API 2 is that API 2 operating mode requires that frames use escape characters (bytes).

Configuration of the serial XBee communication—whether it is transparent, API non-escaped (API 1), or API escaped (API 2)—does not prevent wireless communication between XBee modules. Since only the payload portion of the API frame is transmitted over the air, the receiving XBee modules will alter the packet information based on their **AP** setting, allowing an API non-escaped module to successfully communicate with others working in API escaped or Transparent mode.

Note Devices in the same network do not need to have the same AP mode set to communicate.

API escaped operating mode (API 2)

API non-escaped (API 1) operation relies solely on the start delimiter and length bytes to differentiate API frames. If bytes in a packet are lost, the length count will be off, and the next API frame (packet) will also be lost. API escaped (API 2) operation involves escaping character sequences in an API frame in order to improve reliability, especially in noisy RF environments.

The basic frame structure of both API modes is the same, but in API escaped (API 2) mode, all bytes except for the start delimiter must be escaped if needed. The following data bytes must be escaped in API 2 mode:

- 0x7E: Start delimiter
- 0x7D: Escape character
- 0x11: XON
- 0x13: XOFF

API 2 mode guarantees all the 0x7E bytes received are start delimiters: this character cannot be part of any of the other frame fields (length, data, or checksum) since it must be escaped.

To escape a character:

1. Insert 0x7D, the escape character.
2. Append it with the byte to be escaped, XORed with 0x20.

In API 2 mode, the length field does not include any escape character in the frame and the checksum is calculated with non-escaped data.

Example: Escape an API frame

To express the following API non-escaped frame in API 2 mode:

Start Delimiter	Length		Frame Data															Checksum
			Frame type	Data														
7E	00	0F	17	01	00	13	A2	00	40	AD	14	2E	FF	FE	02	4E	49	6D

The 0x13 byte must be escaped:

1. Insert a 0x7D.
2. XOR the byte 0x13 with 0x20: 13 ⊕20 = 33.

This is the resulting frame. Note that the length and checksum are the same as the non-escaped frame.

Start Delimiter	Length		Frame Data															Checksum
			Frame type	Data														
7E	00	0F	17	01	00	7D	33	A2	00	40	AD	14	2E	FF	FE	02	4E	49	6D

XBee frame exchange

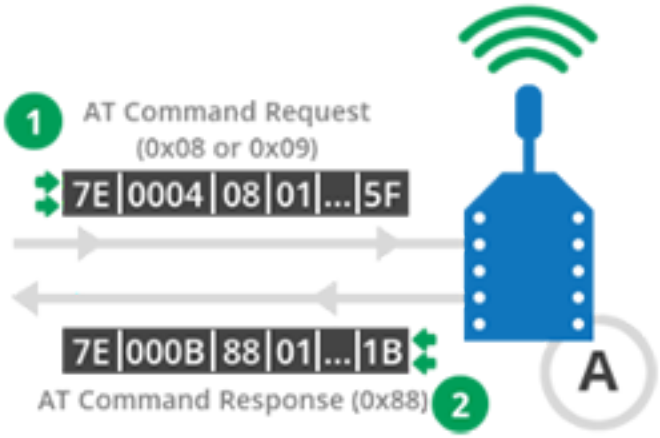
Now that you understand how API mode works and how API frames are structured, the next step is to learn about how frames are exchanged when you perform certain common operations such as configuring an XBee module or transmitting wireless data.

The following section provides examples using XCTU. You can use the Frames interpreter tool of the XCTU console to view detailed API frame structure.

AT Command: configure a local XBee device

To query or set the value of the local XBee—that is, the device directly connected to an intelligent device such as a microcontroller or PC via the serial interface—you must use AT parameters and commands. These are the same AT parameters and commands that are available in Transparent/Command mode, but included in an AT Command (0x08) frame. The response containing the result of the operation is sent back in an AT Command Response (0x88) frame.

The following image shows the API frame exchange that takes place at the serial interface when sending either an AT Command (0x08) or an AT Command Queue Parameter Value (0x09) request.



During an API frame exchange, the following process occurs:

1. An AT Command (0x08) frame is sent to the device through the serial input. This frame contains configuration instructions or queries parameters on the local XBee device.
2. The XBee device processes the command and returns an AT Command Response (0x88) through its serial output. If the frame ID of the AT Command frame is **0**, this response is not sent.

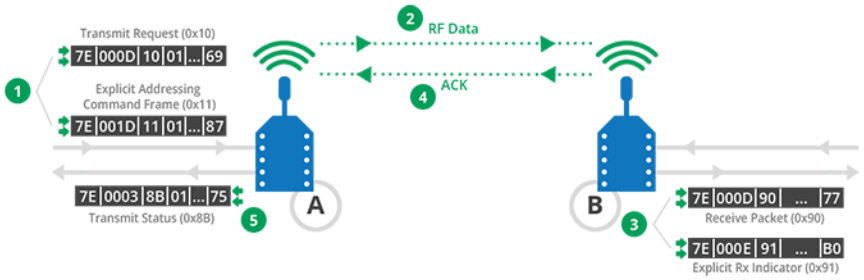
Transmit Request/Receive Packet: Transmit and receive wireless data

A Transmit Request frame encapsulates data with its remote destination and some transmission options. The wireless data received by an XBee module is included in a Receive Packet frame along with the remote transmitter and options for receipt.

Two more frames use Explicit addressing. They require that you specify application layer addressing fields (endpoints, cluster ID, profile ID).

For more information about Explicit addressing, see the Zigbee communication in depth chapter.

The following image shows the API exchanges that take place at the serial interface when transmitting wireless data to another XBee module.



1. The intelligent device (host) sends a Transmit Request (0x10) or an Explicit Addressing Command Frame (0x11) to XBee A through the serial input to transmit data to XBee B.
2. XBee A wirelessly transmits the data in the frame to the module configured as destination in the same frame; in this case, the destination is XBee B.
3. The remote XBee B module receives the wireless data and sends out through the serial output a Receive Packet (0x90) or an Explicit Rx Indicator (0x91), depending on the value of API Options (**AO**) setting. These frames contain the data received over the air and the source address of the XBee module that transmitted it, in this case XBee A.
4. The remote XBee B module transmits a wireless acknowledge packet with the status to the sender, XBee A.
5. The sender XBee A module sends out a Transmit Status (0x8B) through its serial output with the status of the transmission to XBee B.

The Transmit Status (0x8B) frame is always sent at the end of a wireless data transmission unless the frame ID is set to '0' in the transmit request. If the packet cannot be delivered to the destination, the transmit status frame will indicate the cause of failure.

To send data using an explicit frame:

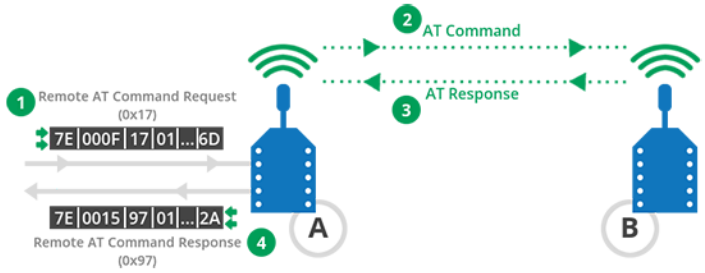
- The source and destination endpoints must be E8.
- The cluster ID must be 0011.
- The profile ID must be C105.

To receive an explicit frame, the API Options (**AO**) parameter must be configured to API Explicit Rx Indicator - 0x91 [1]. If this setting is API Rx Indicator - 0x90 [0], a Receive Packet (0x90) will be received instead of an Explicit Rx Indicator (0x91).

Remote AT Command: Remotely configure an XBee module

Working in API mode also allows you to configure remote XBee modules wirelessly. Any AT command or parameter that can be issued locally can also be sent wirelessly for execution on a remote XBee module.

The following image shows the API frame exchanges that take place at the serial interface when sending a Remote AT Command Request (0x17) to remotely read or set an XBee parameter.



1. The intelligent device (host) sends a Remote AT Command Request (0x17) to XBee A through the serial input to configure the remote XBee B.
2. XBee A wirelessly transmits the AT Command in the frame to the module configured as destination in the same frame; in this case, the destination is XBee B.
3. XBee B receives the AT command and processes the command to wirelessly return the result to the sender, XBee A.
4. XBee A sends out a Remote AT Command Response (0x97) through its serial output with the result of the AT command processed by XBee B. If the frame ID of the Remote AT Command frame is '0', this response is not sent.

Lab: Configure your local XBee module

This section demonstrates how to read the Node Identifier (NI) of your local XBee module configured in API mode. To do this, you create an AT command frame to read the NI parameter, send it to the XBee module, and analyze the response.

Note If you get stuck, go to Troubleshooting.

If you get stuck, see Troubleshooting.

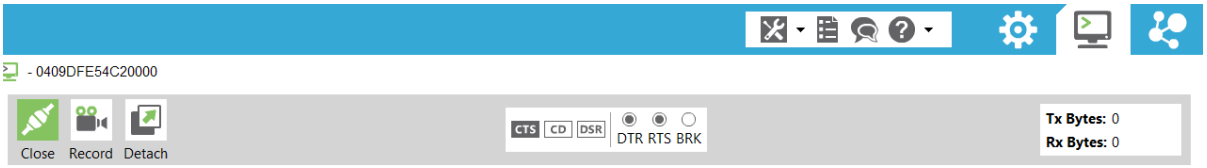
Configure the XBee module

Before creating and sending the frame, configure the XBee module as follows:

Param	Value	Effect
NI	XBEE_A	Defines the node identifier, a human-friendly name for the module. The default **NI** value is a blank space. Make sure to delete the space when you change the value.
AP	AP Enabled [1]	Enables API mode.

Open the XCTU console

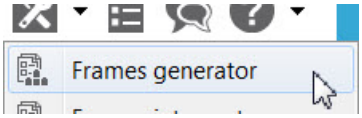
1. Switch to the **Consoles working mode** .
2. Open the serial connection with the radio module .



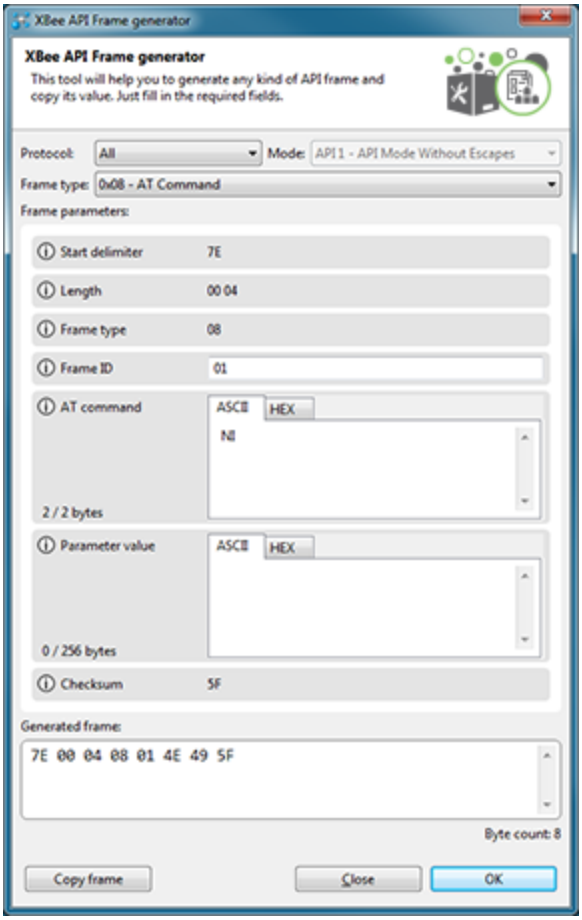
Generate the AT command frame

These instructions describe how to generate an AT command frame using the XCTU Frame Generator tool.

1. Click **Add new frame to the list** .
2. Open the **Frames Generator** tool.



3. In the **Frame type** section, select **0x08 - AT Command**.
4. In the **AT command** section, select the **ASCII** tab and type **NI**.
5. Click **OK**.
6. Click **Add frame**.

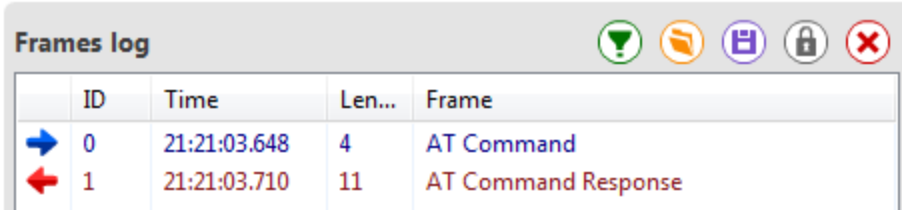


Send the AT command frame

After you have created an AT command frame, you must send it to the local XBee module to receive a response containing the configured **NI** value.

1. Select the frame in the XCTU **Send frames** section.
2. Click **Send selected packet**.

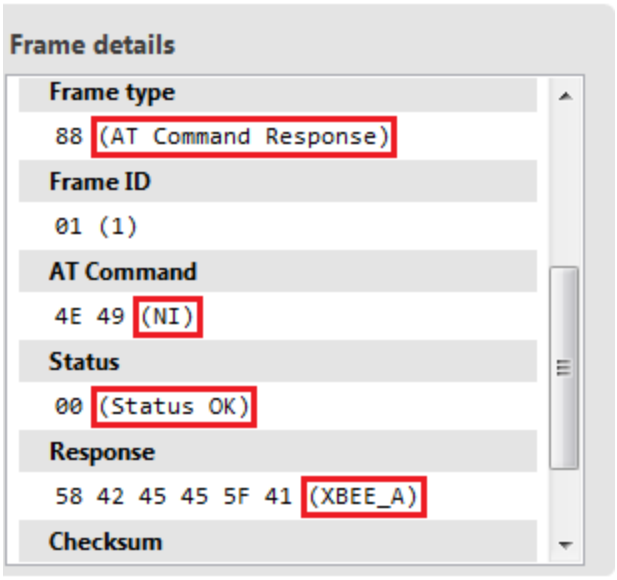
The **Frames log** indicates that one frame has been sent (blue) and another has been received (red).



Analyze the response

Once you have sent the frame, you can analyze the responses on the receiving end.

1. Select the frame received (AT Command Response) to see its details in the **Frame details** section.
2. Analyze its details and verify that it contains the **NI** value of your module.
   - **Frame type**: The received frame is an AT Command Response.
   - **Frame ID**: This AT Command Response frame is the answer to the sent AT Command request because both have the same value (1).
   - **Status**: The value was successfully read because the status is OK.
   - **Response**: This received frame contains the value of the NI parameter previously requested in the AT Command frame, XBEE_A



3. Disconnect the console by clicking **Close the serial connection** .

Example: Transmit and receive data

This section describes how to transmit data to another XBee module using the XCTU console. The steps include creating a Transmit Request frame with the message you want to transmit to the other module and sending the frame serially to the local XBee module. You can then analyze the responses, both in the local and the remote module.

Note If you get stuck, go to Troubleshooting.

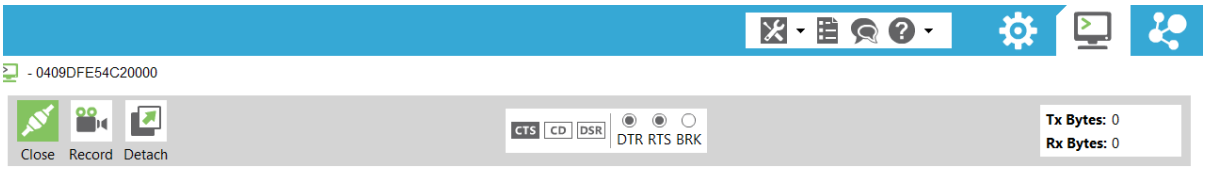
If you get stuck, see Troubleshooting for XBee ZigBee Mesh Kit.

Configure the XBee modules

Before creating and sending the frame, configure the XBee modules as follows:

Param	XBee A	XBee B	Effect
ID	2015	2015	Defines the network that a radio will attach to. This must be the same for all radios on your network.
NI	SENDER	RECEIVER	Defines the node identifier, a human-friendly name for the module. The default **NI** value is a blank space. Make sure to delete the space when you change the value.
AP	API Enabled [1]	API Enabled [1]	Enables API mode.

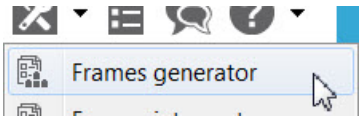
Open the XCTU console

1. Switch to the **Consoles working mode** .
2. Open the serial connection with the radio module .
3. Change to the console of the other XBee module.
4. Open the serial connection with the radio module .
5. 

Generate the Transmit Request frame

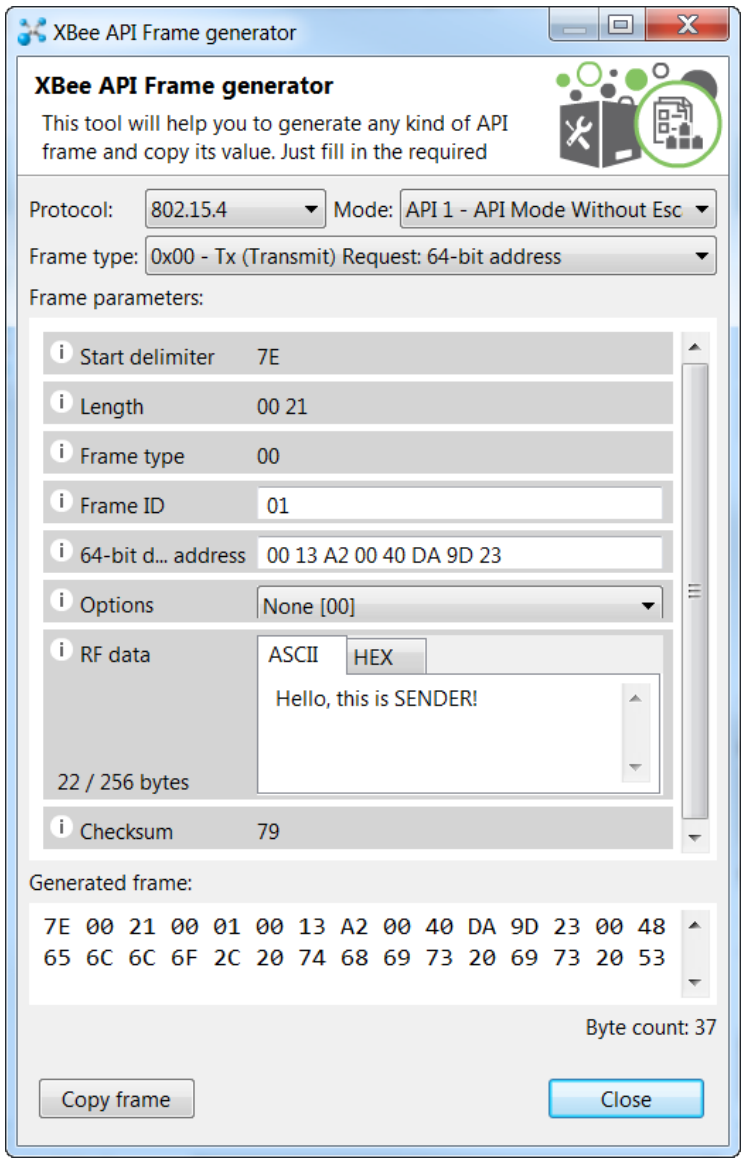
This topic describes how to generate a Transmit Request frame using the XCTU SENDER console.

1. Go to the **SENDER** console and detach it Detach to see two consoles at the same time.
2. In the SENDER console, click **Add new packet to the list** .
3. Open the **Frames Generator** tool.



4. In the **Protocol control**, select **802.15.4**.
5. In the **Frame type** control, select **0x00 – Tx (Transmit) Request: 64-bit address**.
6. In the **64-bit dest. address** box, type the 64-bit address of the RECEIVER module.
7. In the **RF data** box, click the **ASCII** tab and type the message "Hello, this is SENDER!"
8. Click **OK**.

9. Click **Add frame**.

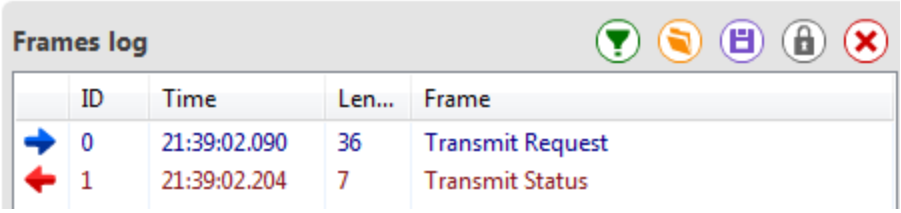


Send the Transmit Request frame

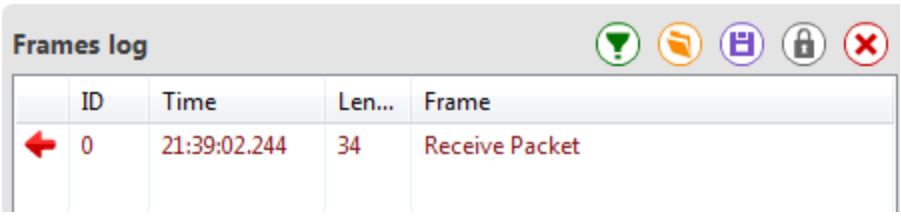
After you have created a Transmit Request frame, you must send it.

1. Select the frame in the XCTU **Send frames** section.
2. Click **Send selected packet**.

The **Frames log** indicates that one frame has been sent (blue) and another has been received (red).



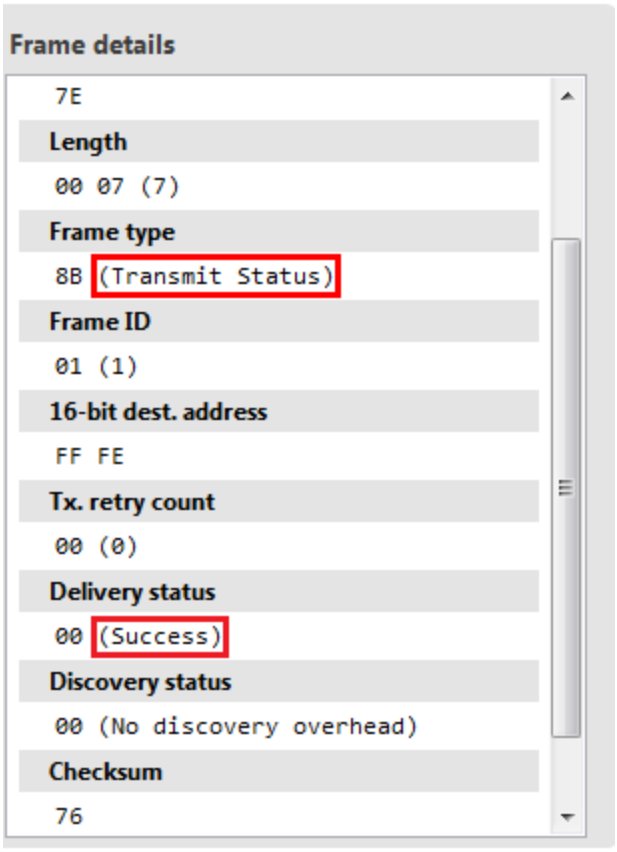
Additionally, the RECEIVER console indicates that another packet has been received.



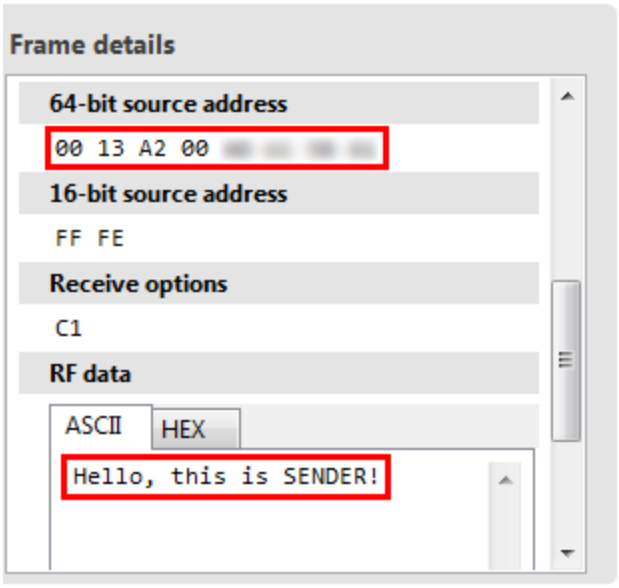
Analyze the responses

Once you have sent the frames, you can analyze the responses on the receiving end.

1. Select the received frame (Transmit Status) in the SENDER console to view the frame details on the right panel. Verify that the message was sent successfully.
   - **Frame type**: The received frame is a Transmit Status.
   - **Frame ID**: Since both frames have the same Frame ID, this is the response for the Transmit Request frame.
   - **Status**: The Success status indicates that the message was sent successfully.



2. Analyze the details of the Receive Packet for RECEIVER. Verify that the message is the one you typed and the sender's address belongs to SENDER.
   - **Frame type**: The received frame is a Receive Packet
   - **64-bit source address**: This field displays the 64-bit address of the sender module, SENDER.
   - **Receive options**:
     - It is a DigiMesh message (**0xC1** = **1100 0001**).
     - It is a unicast transmission (**0xC1** = **1100 0001**).
     - The packet was acknowledge (**0xC1** = **1100 0001**).
   - **RF data**: The message of the packet is "Hello, this is SENDER!".



3. Disconnect both consoles by clicking **Close the serial connection** .

Do more with API mode: XBee libraries

Let's say you want to write an application to enable an intelligent device to monitor and manage an XBee network. You can write your own code to work with API mode, and you can also take advantage of existing software libraries that already parse the API frames. Depending on your preferred programming language and the intelligent device connected to the serial interface of the XBee module, you can choose from a variety of available libraries.

Here are some of the libraries available:

- **XBee mbed Library** is a ready-to-import mbed extension to develop XBee projects on the mbed platforms. For details, go to https://developer.mbed.org/teams/Digi-International-Inc/code/XBeeLib/.
- **Digi XBee Ansi C Library** is a collection of portable ANSI C code for communicating with XBee modules in API mode. For details, go to https://github.com/digidotcom/xbee_ansic_library/.
- **XBee-arduino** is an Arduino library for communicating with XBees in API mode. To learn more, go to https://code.google.com/p/xbee-arduino/.
- **XBee Java Library** is an easy-to-use library developed in Java that allows you to interact with XBee modules working in API mode. For details, go to XBee Java Library documentation.

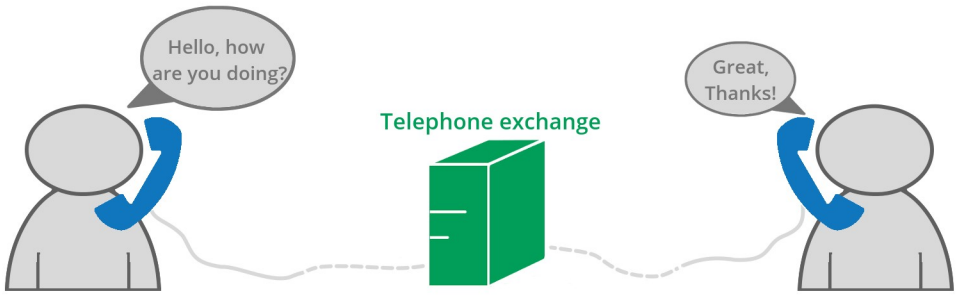
In this kit, you use the **XBee Java Library** to learn about the XBee features and capabilities offered in API operating mode. You create several Java applications to control and monitor XBee modules connected to your computer via the XBee Grove Development Board.

Communication models

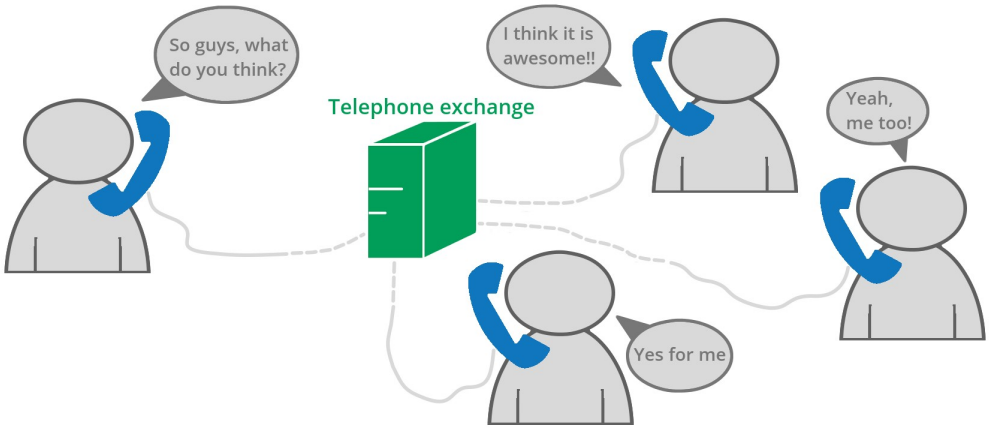
All types of communication require a sender, a message, and at least one receiver. Communication with XBee devices is no different: you need a device to send the message, the message itself, and a device (or devices) to receive that message. The channel in the device communication is the air because the message is transmitted wirelessly.

Considering the sender and the receiver, we can distinguish two types of communication:

Point-to-point communication involves just one sender and one receiver, so the message transmitted by the sender can only be heard by the receiver. Think of a simple telephone call with your friend.



Point-to-multipoint communication involves one sender and multiple receivers. The sender can transmit a message to one receiver or to all (broadcast message). Think of a conference call with all your friends.



In both cases, the communication is bi-directional. This means the sender can also be the receiver, and vice versa.

Point-to-point communication

The simplest way to communicate using XBee modules is point to point, which refers to a communication between two nodes. An example of this type of communication in the everyday world is a telephone call, where the two phones are connected and what is said by the caller can only be heard by the receiver.

In XBees, data received by one module from a computer or microcontroller is transmitted wirelessly to the other and vice versa. The point-to-point communication is a wireless link between two XBee modules.

To phone a friend, your telephone and your friend's must be connected to the telephone line; for XBees, the network is the equivalent to the line. This means the modules must be in the same network, so their Channel (CH) and Network ID (ID) must coincide, as explained in Wireless communication.

But to start the call you also have to dial your friend's number. In the same way, an XBee module needs to know the "telephone number" of the destination module where it is going to transmit the data. For an XBee, its 64-bit address or MAC is equivalent to the phone number. The way an XBee "dials" this number depends on how it is configured—transparent or API mode:

- Basic chat is an example of point-to-point communication in transparent mode.
- The same chat but in API mode is explained in Example: Chat.

Example: Chat

Now that you know how to build a Basic chat between two XBee modules in transparent mode, let's go one step further. As you learned, an XBee module in transparent mode simply passes information along exactly as it receives it. API mode allows for more flexibility and, in many cases, increased reliability.

In API mode, you still send the message to the module. But, you also send other necessary information, such as the destination address or checksum value, all wrapped in a packet with a defined structure called an API frame. This means that in API mode you don't need to set the

destination address (DH + DL) in the module. Similarly, the receiver module receives more information than the message itself, such as the source address, signal strength, or checksum value.

Tip Detailed information about API operation can be found in the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide and the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

In this example you will repeat the previous chat but in API mode. To that end, you will create an application in Java programming language. To simplify that application, you will use the XBee Java Library, an easy-to-use API that allows you to interact with the XBee modules.

Note If you get stuck, go to Troubleshooting.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-1.1.1.zip or later release file)
- Java Virtual Machine 6 or later
- A Java IDE (Eclipse, NetBeans, etc)

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee modules

To communicate, both modules have to be in the same network, so their Channel (**CH**) and Network ID (**ID**) must be the same.

Additionally, you must configure the modules to work in API mode by changing the API Enable (**AP**) parameter.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

Parameter	XBee A	XBee B	Effect	Parameter
CH	B	B	Defines the frequency to use to communicate. This parameter must be the same for all radios on your network.	CH
ID	2015	2015	Defines the network that a radio will attach to. This parameter must be the same for all radios on your network.	ID
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.	NI
AP	API enabled [1]	API enabled [1]	Enables API mode.	AP

3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

Chat!

In order to chat with the other module, use the following steps in order.

Create an empty Java project named XBeeAPIChatA

To create the project, choose one of these development options and follow the steps:

Option 1 - Eclipse

To create a new Java project in Eclipse, follow these steps:

1. Navigate to the **File** menu, select **New**, and click **Java Project**.
2. You are prompted with a **New Java Project** window. Enter the Project name from the title of this step.
3. Click **Next** and continue to Link all Java and native libraries to the project.

or

Option 2 - NetBeans

To create a new Java project in NetBeans, follow these steps:

1. Navigate to the **File** menu and select **New project...**.
2. You will be prompted with a New Project window. In the Categories frame, select **Java > Java Application** fro m the panel on the right. Click **Next**.
3. Enter the Project Name from the title of this step and the Project Location. De-select the option called Create Main Class; it will be created later.
4. Click **Finish** to create the project. The window closes and the project is listed in the Projects view on the left side of the IDE.

Link all Java and native libraries to the project

Link the XBee Java Library, the RXTX library (including the native one), and the logger library to the project.

Ensure that you have downloaded and unzipped the XBJL_X.Y.Z.zip library.

Option 1 - Eclipse

1. Go to the **Libraries** tab of the New Java Project window.
2. Click **Add External JARs...**.
3. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbjli b-X.Y.Z.jar** file.
4. Click **Add External JARs...** again.
5. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api-x.y.z.jar**
   - **slf4j-nop-x.y.z.jar**
6. Expand the **rxtx-2.2.jar** file of the Libraries tab list, select **Native library location**, and click **Edit...**.
7. Click **External folder**... to navigate to the extra-libs\native\Windows\win32 folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Windows\win32 must be replaced by the right directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits).

Tip In this last step, select the folder that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute java -version.

---

8. Click **OK** to add the path to the native libraries.
9. Click **Finish**.

or

Option 2 - NetBeans

1. From **Projects** view, right-click your project and go to **Properties**.
2. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.

3. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the **xb jlib-X.Y.X.jar** file.
4. Click **Add JAR/Folder** again.
5. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api- x.y.z .jar**
   - **slf4j-nop- x.y.z .jar**
6. Select **Run** in the left tree of the Properties dialog.
7. In the **VM Options** field, add the following option:
   -Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
   where,
   - <path_where_the_XBee_Java_Library_is_unzipped> must be replaced by the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Windows\win32 must be replaced by the right directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits).
8. Click **OK**.

Open the following source code, select all, and copy it to the clipboard:

```
package com.digi.wck.chat;

import java.util.Scanner;

import com.digi.xbee.api.RemoteXBeeDevice;
import com.digi.xbee.api.XBeeDevice;
import com.digi.xbee.api.XBeeNetwork;
import com.digi.xbee.api.exceptions.XBeeException;
import com.digi.xbee.api.listeners.IDataReceiveListener;
import com.digi.xbee.api.models.XBeeMessage;

/**
 * Wireless Connectivity Kit Chat Sample application.
 *
 * <p>This sample Java application shows how to send and receive data to/from
 * another XBee device on the same network using the XBee Java Library.</p>
 *
 * <p>In this example, every text line typed in the standard input is
wirelessly
 * sent to the specified remote device.</p>
 */
public class MainApp {

	/* Constants */

	// TODO Replace with the port where your module is connected to.
	private static final String PORT = "COM1";
	// TODO Replace with the baud rate of your module.
	private static final int BAUD_RATE = 9600;
	// TODO Replace with the node identifier (NI) of the other module.
	private static final String REMOTE_NODE_ID = "XBEE_B";
```

```
	private static final Scanner s = new Scanner(System.in);

	private static DataReceiveListener listener = new DataReceiveListener();

	/**
	 * Application main method.
	 *
	 * @param args Command line arguments.
	 */
	public static void main(String[] args) {
		System.out.println("+--------------------------------------------+");
		System.out.println("|               API Chat Sample              |");
		System.out.println("+--------------------------------------------+\n");

		XBeeDevice myDevice = new XBeeDevice(PORT, BAUD_RATE);

		try {
			myDevice.open();

			myDevice.addDataListener(listener);

			System.out.println("\nLocal XBee: " + myDevice.getNodeID());

			if (myDevice.getNodeID().equals(REMOTE_NODE_ID)) {
				System.err.println("Error: the value of the REMOTE_NODE_ID con
"
					+ "the Node Identifier (NI) of the OTHER modul
			} else {
				System.out.println("\nEstablishing connection with " + REMOTE_
"...");

				XBeeNetwork network = myDevice.getNetwork();
				RemoteXBeeDevice remote = network.discoverDevice(REMOTE_NODE_I
				if (remote != null) {
					System.out.println("Connection established. Type your
					while (true) {
						String message = s.nextLine();
						try {
							myDevice.sendData(remote, message.getB
						} catch (XBeeException e) {
							System.err.println("Error transmitting
						}
					}
				} else {
					System.err.println("Could not find the module " + REMO
the network.");
				}
			}

		} catch (XBeeException e) {
			e.printStackTrace();
			myDevice.close();
			System.exit(1);
		} finally {
			myDevice.close();
			if (s != null)
				s.close();
		}
	}
```

```
        /**
         * Class to manage the received data.
         */
        private static class DataReceiveListener implements IDataReceiveListener {
                @Override
                public void dataReceived(XBeeMessage xbeeMessage) {
                        System.out.println(xbeeMessage.getDevice().getNodeID() + ": " +
                                        new String(xbeeMessage.getData()));
                }
        }

}
```

Add the source code to the project

Once you have created your project, the next step is to add the Java source file.

Option 1 - Eclipse

1. In the **Package Explorer** view, select the project and right-click.
2. From the context menu, select **New > Class**. The **New Java Class** wizard opens.
3. Type the **Name** of the class: **MainApp**.
4. Click **Finish**.
5. The MainApp.java file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
6. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

or

Option 2 - NetBeans

1. In the **Projects** view, select the project and right-click.
2. From the context menu, select **New > Java Class...** The New Java Class wizard will open.
3. Modify the **Class Name** to be **MainApp**.
4. Click **Finish**.
5. The MainApp.java file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
6. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

Final steps

Change the port name in the Java source code to match the port that the module is connected to and the node identifier of the other module

```
// TODO Replace with the port where your module is connected.
private static final String PORT = "COM1";
// TODO Replace with the baud rate of your module.
private static final int BAUD_RATE = 9600;
```

```
// TODO Replace with the node identifier (NI) of the other module.
private static final String REMOTE_NODE_ID = "XBEE_B";
```

CAUTION! Make sure the REMOTE_NODE_ID constant is the Node Identifier (NI) of the OTHER XBee and NOT the NI of the XBee connected to the port specified in the PORT constant.

Duplicate the Java project for the other XBee and rename it to **XBeeAPIChatB**.

In the Java source code of the second project, change the port name to match the port that the module is connected to and the node identifier of the other module.

Launch the application in both computers. If you are using only one computer, launch two applications—one for each port.

Type *Hi XBee!* in one console and press enter. The XBee wirelessly sends the message to the other module. That module, in turn, displays the message in the associated console.

What have you learned?

In this section, you have learned that:

- Every XBee module has a unique 64-bit address called MAC that distinguishes it from the rest of devices. This address is formed by the concatenation of the parameters **SH** (Serial Number High) and **SL** (Serial Number Low).
- To communicate, your devices must be part of the same network. To do so, all modules must have the same values for the parameters **CH** and **ID**.
- Point-to-point communication involves one sender and one receiver, so the message transmitted by the sender can only be obtained by the receiver. This type of communication is bi-directional.
- Using API mode, the sender not only transmits a raw message but also some extra information, like the address of the source XBee, packaged in what is called an API frame .
- It is not necessary to set the **DH** and **DL** parameters of the receiver device because the address of the destination XBee is included in the API frame.
- You can use the XBee Java Library to simplify and improve the use of the API operating mode.

Extend the example

If you are ready to move beyond this exercise and extend the example, try the following:

- Protect the wireless messages and communicate in a secure way by enabling encryption in your XBee modules. Use Arduino or Raspberry Pi instead of a computer to transmit data wirelessly.
- Use your XBees as a serial cable replacement for your serially communicating applications. For example, if you have an Arduino application that controls your room lights via the serial port, replace the serial cable with XBees and move your Arduino around the house.
- Add more modules to the network and, modifying the code provided in this example, create an application that allows a module to communicate with any module on the same network.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

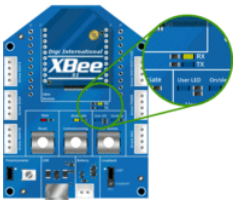
1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.
- Invalid operating mode exception message

If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

- Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

In this case, the library cannot access the module. Check that the PORT constant is correct.

- Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

  - "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
```

```
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
        at java.lang.Runtime.loadLibrary0(Runtime.java:849)
        at java.lang.System.loadLibrary(System.java:1087)
        at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
        at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
        at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
        ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
        at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
        at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
        at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
        at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
        at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

- xbee-java-library-X.Y.Z.jar
- rxtx-2.2.jar
- slf4j-api-x.y.z.jar
- slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

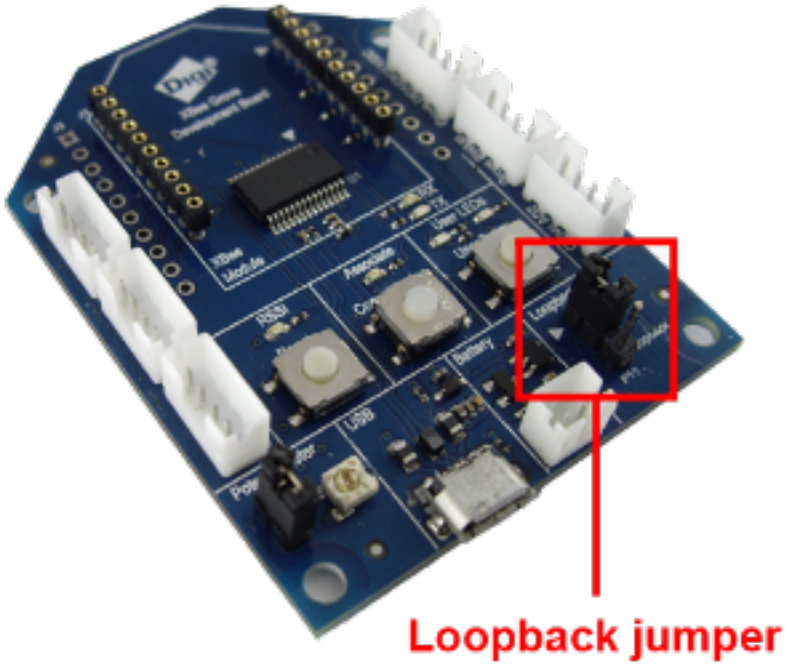
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

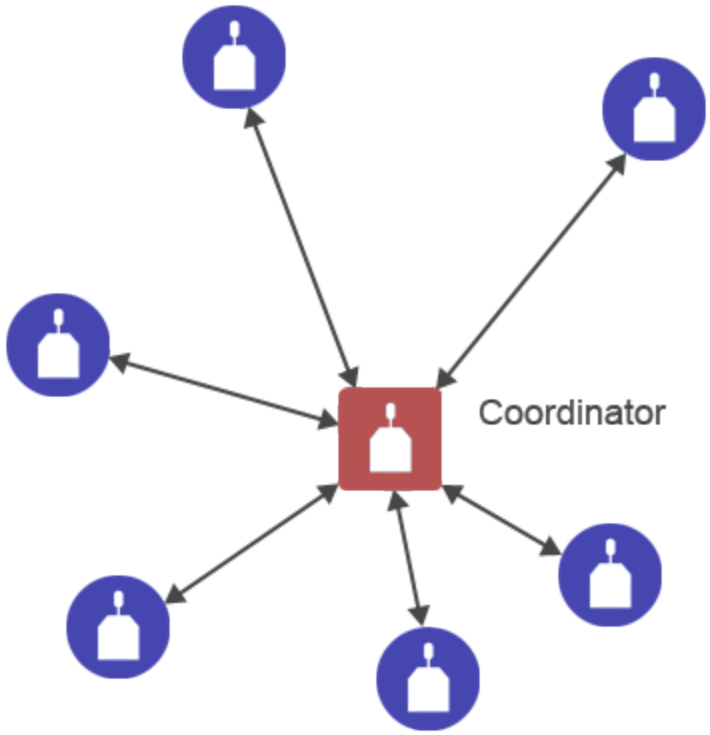
The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

Point-to-multipoint communication

Point-to-point communication works well if you have two XBee modules. When the number of devices increases, however, this model falls short. Using point-to-multipoint communication, a module can communicate with one or multiple devices on the network. This type of communication generally involves a central coordinator with multiple remote nodes (end devices) connecting back to the central host. This network topology is called star.



A home security system can illustrate the concept: proximity sensors connected to each end device send a notification to the central coordinator when movement is detected. The coordinator can also send data to a specific end device or to all of them.

Roles

In the 802.15.4 protocol, the XBee modules can have two different roles: coordinator or end device.

- The **coordinator** is the central node of the network. It starts the network, allows other devices to join it, can select the frequency channel to use, and provides network synchronization by polling nodes. To set an XBee module as coordinator, you have to change the Coordinator Enable (**CE**) parameter to Coordinator [1].

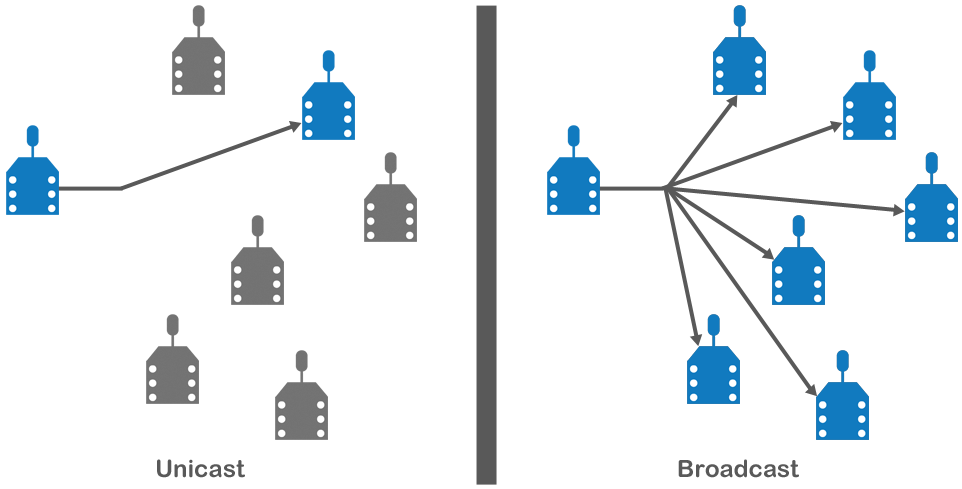
Note Note that in star topologies there can be only one coordinator in the network.

- An **end device** is a remote node of the network. It can communicate with the coordinator but also with other end devices. It can be put into states of sleep for low-power applications, as you will learn in the Extending the battery life topic. By default, all XBee modules in the 802.15.4 protocol are end devices. If you want to set a coordinator as end device, you must change the CE parameter to End Device [0].

Unicast and broadcast

A unicast transmission consists of sending messages to a single node on the network that is identified by a unique address. An example of unicast communication is a telephone call between two people. To send a unicast transmission, the XBee module needs to know the address of the destination module.

Broadcast, unlike unicast, means transmitting the same data to all possible nodes on the network. An example of broadcast communication is a radio station. To send a broadcast transmission, the destination address must be 000000000000FFFF.



Congratulations! You have just completed the section, so you are ready to do the Example: Advanced chat.

Example: Advanced chat

As you have just learned, point-to-point communication falls short when more than two XBee modules are used. In this example, you will extend the point-to-point chat example to use point-to-multipoint communication instead. Since this kit only contains two XBee modules, one module will act as coordinator and the other as end device.

Once you have everything set up, send a message to a specific device (unicast) or to all devices of the network (broadcast).

Note If you get stuck, go to Troubleshooting.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-1.1.1.zip or later release file)
- Java Virtual Machine 6 or later
- A Java IDE (Eclipse, NetBeans, etc)

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee modules

To communicate, both modules have to be in the same network, so their Channel (**CH**) and Network ID (**ID**) must be the same.

Additionally, you must configure the modules to work in API mode by changing the API Enable (**AP**) parameter.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

Parameter	XBee A	XBee B	Effect	Parameter
CH	B	B	Defines the frequency of communication. This parameter must be the same for all radios on your network.	CH

Parameter	XBee A	XBee B	Effect	Parameter
ID	2015	2015	Defines the network a radio will connect to. This parameter must be the same for all radios on your network.	ID
CE	Coordinator [1]	End Device [0]	Defines the role of the XBee module.	CE
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.	NI
AP	API enabled [1]	API enabled [1]	Enables API mode.	AP

3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

Chat!

In order to chat with other modules, use the following steps in order.

Create an empty Java project named XBeeAdvancedChatA

To create the project, choose one of these development options and follow the steps:

Option 1 - Eclipse

To create a new Java project in Eclipse, follow these steps:

1. Navigate to the **File** menu, select **New**, and click **Java Project**.
2. You are prompted with a **New Java Project** window. Enter the Project name from the title of this step.
3. Click **Next** and continue to Link all Java and native libraries to the project.

or

Option 2 - NetBeans

To create a new Java project in NetBeans, follow these steps:

1. Navigate to the **File** menu and select **New project...**.
2. You will be prompted with a New Project window. In the Categories frame, select **Java > Java Application** fro m the panel on the right. Click **Next**.
3. Enter the Project Name from the title of this step and the Project Location. De-select the option called Create Main Class; it will be created later.
4. Click **Finish** to create the project. The window closes and the project is listed in the Projects view on the left side of the IDE.

Link all Java and native libraries to the project

Link the XBee Java Library, the RXTX library (including the native one), and the logger library to the project.

Ensure that you have downloaded and unzipped the XBJL_X.Y.Z.zip library.

Option 1 - Eclipse

1. Go to the **Libraries** tab of the New Java Project window.
2. Click **Add External JARs...**.
3. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbjli b-X.Y.Z.jar** file.
4. Click **Add External JARs...** again.
5. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api-x.y.z.jar**
   - **slf4j-nop-x.y.z.jar**
6. Expand the **rxtx-2.2.jar** file of the Libraries tab list, select **Native library location**, and click **Edit...**.
7. Click **External folder**... to navigate to the extra-libs\native\Windows\win32 folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Windows\win32 must be replaced by the right directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits).

Tip In this last step, select the folder that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute java -version.

8. Click **OK** to add the path to the native libraries.
9. Click **Finish**.

or

Option 2 - NetBeans

1. From **Projects** view, right-click your project and go to **Properties**.
2. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.
3. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the **xb jlib-X.Y.X.jar** file.
4. Click **Add JAR/Folder** again.
5. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api- x.y.z .jar**
   - **slf4j-nop- x.y.z .jar**
6. Select **Run** in the left tree of the Properties dialog.
7. In the **VM Options** field, add the following option:
   -Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\nat

ive\Windows\win32
where,

- <path_where_the_XBee_Java_Library_is_unzipped> must be replaced by the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
- Windows\win32 must be replaced by the right directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits).

8. Click **OK**.

Open the following source code, select all, and copy it to the clipboard:

Delete this text and replace it with your own content.

```
package com.digi.wck.advancedchat;

import java.util.Scanner;

import com.digi.xbee.api.RemoteXBeeDevice;
import com.digi.xbee.api.XBeeDevice;
import com.digi.xbee.api.XBeeNetwork;
import com.digi.xbee.api.exceptions.XBeeException;
import com.digi.xbee.api.listeners.IDataReceiveListener;
import com.digi.xbee.api.models.XBeeMessage;

/**
 * Wireless Connectivity Kit Advanced Chat Sample application.
 *
 * <p>This sample Java application shows how to send and receive data to/from
 * another XBee devices on the same network using the XBee Java Library.</p>
 *
 * <p>In this example you can send messages to a specific XBee (unicast) or to
 * all (broadcast), following the next pattern:
 *   - Unicast: NODE_IDENTIFIER: message
 *   - Broadcast: ALL: message
 * </p>
 */
public class MainApp {

	/* Constants */

	// TODO Replace with the port where your module is connected to.
	private static final String PORT = "COM1";
	// TODO Replace with the baud rate of your module.
	private static final int BAUD_RATE = 9600;

	private static final Scanner s = new Scanner(System.in);

	private static DataReceiveListener listener = new DataReceiveListener();

	/**
	 * Application main method.
	 *
	 * @param args Command line arguments.
	 */
	public static void main(String[] args) {
		System.out.println("+--------------------------------------------+");
		System.out.println("|             Advanced Chat Sample           |");
		System.out.println("+--------------------------------------------+\n");
```

```
		XBeeDevice myDevice = new XBeeDevice(PORT, BAUD_RATE);

		try {
			myDevice.open();

			XBeeNetwork network = myDevice.getNetwork();
			network.startDiscoveryProcess();

			System.out.println("\nLocal XBee: " + myDevice.getNodeID());

			System.out.println("\nScanning the network, please wait...");
			while (network.isDiscoveryRunning()) {
				sleep(100);
			}

			System.out.println("Devices found:");
			for (RemoteXBeeDevice remote : network.getDevices()) {
				System.out.println(" - " + remote.getNodeID());
			}

			System.out.println("\nType your messages here:\n");

			myDevice.addDataListener(listener);

			while (true) {
				try {
					String[] data = parseData(s.nextLine());

					if (data[0].toLowerCase().equals("all")) {
						myDevice.sendBroadcastData(data[1].getBytes())
					} else {
						RemoteXBeeDevice remote = network.getDevice(da
						if (remote != null) {
							myDevice.sendData(remote, data[1].getB
						} else {
							System.err.println("Could not find the
network.");

						}
					}
				} catch (IndexOutOfBoundsException e) {
					System.err.println("Error parsing the text. Follow the
IDENTIFIER|ALL: message>");
				} catch (XBeeException e) {
					System.err.println("Error transmitting message: " + e.
				}
			}

		} catch (XBeeException e) {
			e.printStackTrace();
			myDevice.close();
			System.exit(1);
		} finally {
			myDevice.close();
			System.exit(0);
		}
	}

	/**
```

```
         * Parses the given text to obtain the destination node identifier and the
         * message.
         *
         * @param text Text in the format <NODE_IDENTIFIER|ALL: message>
         * @return String array that contains the node identifier of the remote
         *         device and the message to send.
         */
        private static String[] parseData(String text) throws
IndexOutOfBoundsException {
                String[] s = new String[2];
                s[0] = text.substring(0, text.indexOf(":"));
                s[1] = text.substring(text.indexOf(":") + 2);
                return s;
        }

        /**
         * Sleeps the thread for the given milliseconds.
         *
         * @param millis Time to sleep the thread in milliseconds.
         */
        private static void sleep(long millis) {
                try {
                        Thread.sleep(millis);
                } catch (InterruptedException e) {}
        }

        /**
         * Class to manage the received data.
         */
        private static class DataReceiveListener implements IDataReceiveListener {
                @Override
                public void dataReceived(XBeeMessage xbeeMessage) {
                        System.out.println("-----------------------------------------------");
                        System.out.println("> " + xbeeMessage.getDevice().getNodeID() +
                                        (xbeeMessage.isBroadcast() ? " (broadcast)" : "") +
                                        ": " + new String(xbeeMessage.getData()));
                        System.out.println("-----------------------------------------------");
                }
        }

}
```

Add the source code to the project

Once you have created your project, the next step is to add the Java source file.

Option 1 - Eclipse

1. In the **Package Explorer** view, select the project and right-click.
2. From the context menu, select **New > Class**. The **New Java Class** wizard opens.
3. Type the **Name** of the class: **MainApp**.
4. Click **Finish**.
5. The MainApp.java file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
6. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

or

Option 2 - NetBeans

1. In the **Projects** view, select the project and right-click.
2. From the context menu, select **New > Java Class...** The New Java Class wizard will open.
3. Modify the **Class Name** to be **MainApp**.
4. Click **Finish**.
5. The MainApp.java file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
6. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

Final steps

Change the port name in the Java source code to match the port the module is connected to.

```
// TODO Replace with the port where your module is connected.
private static final String PORT = "COM1";
// TODO Replace with the baudrate of your module.
private static final int BAUD_RATE = 9600;
```

Duplicate the Java project for the other XBee and rename it to **XBeeAdvancedChatB**.

Change the port name in the second project's source code to match the port the module is connected to.

Launch the application in both computers. If you are using only one computer, launch two applications.

Send messages to a specific XBee (unicast) or to all (broadcast) using the following pattern:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

What have you learned?

In this section, you have learned that:

- Point-to-multipoint communication involves one sender and multiple receivers, so the message can be transmitted to one receiver or to all of them. This type of communication is bi-directional.
- Also, point-to-multipoint communication in transparent mode requires you to configure the sender by setting the destination address (DH and DL parameters) before transmitting a message. This means that the module will enter command mode every time the destination changes. This is one of the advantages API mode has over transparent mode.

- The 802.15.4 protocol can have two roles:
  - The coordinator, or central node of the network, determines the frequency channel and starts the network allowing more devices to join it.
  - An end device, or remote node of the network, can communicate with the rest of the nodes and go into states of sleep to save energy.
- The topology of this kind of network—where there is a coordinator and multiple end devices connected to it—is called a star topology. Depending on the number of devices that are going to receive the message, there are two types of transmissions:
  - Unicast sends a message to one node identified by a unique address
  - Broadcast sends the same message to all possible nodes on the network.

Extend the example

If you are ready to move beyond this exercise and extend the example, try the following:

- Protect the wireless messages and communicate in a secure way by enabling encryption in your XBee modules.
- Extend the network by adding more XBee modules so you can chat with other end devices.
- Change the association settings (A2 in the coordinator to 100b and A1 in the end devices to 0111b) so that the end devices, without knowing the Channel (CH) and Network ID (ID) of the coordinator, associate themselves to the network started by the coordinator. This is useful when you want to add more modules to an already-established network.

Tip Association establishes membership between end devices and a coordinator. Membership is useful in scenarios that require a central unit (coordinator) to relay messages to or gather data from several remote units (end devices), assign channels, or assign PAN IDs.
An end device can associate with a coordinator without knowing the address, PAN ID, or channel of the coordinator. The End Device Association (A1) parameter bit fields determine the flexibility of an end device during association.
For more information on this topic, For additional information about serial communication, go to the legacy XBee/XBee-PRO 802.15.4 RF Module User Guide or the XBee/XBee-PRO S2C 802.15.4 RF Module User Guide.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

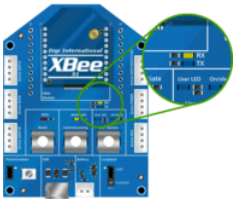
You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.
Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.
Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.
Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.
- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.
  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

In this case, the library cannot access the module. Check that the PORT constant is correct.

- Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

  - "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
       at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
       at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
       ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

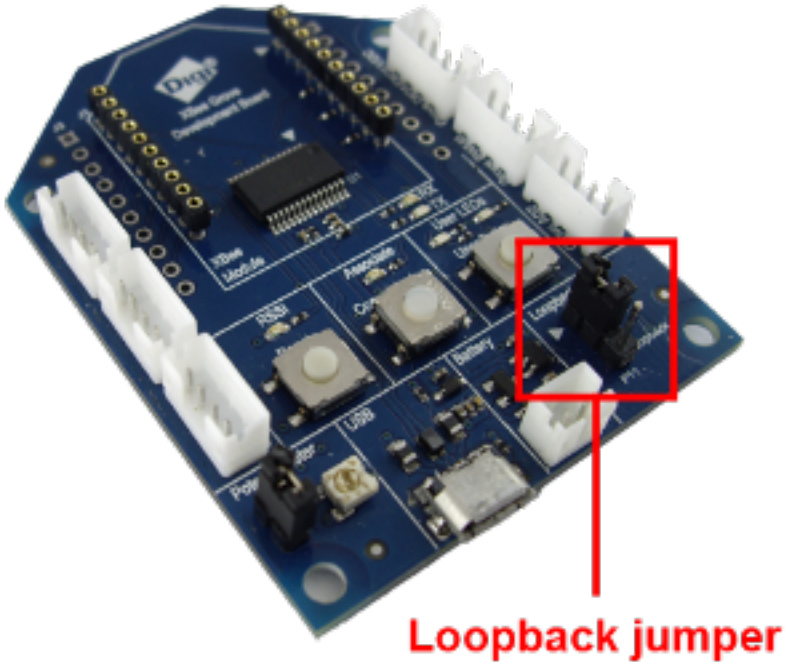
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

Inputs and outputs

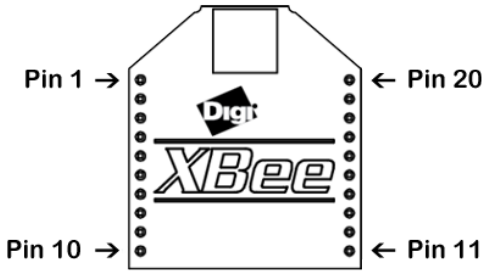
All XBee modules have a set of pins that can be used to connect sensors or actuators and configure them for specific behavior. Each XBee radio has the capability to directly gather sensor data and transmit it without the use of an external microcontroller.

With these pins you can, for example, turn on a light by sending information to an XBee module connected to an actuator, or measure the outside temperature by obtaining data from a temperature sensor attached to your XBee module.

Learn about I/O pins, sensors, actuators in this section, then put your knowledge to work by using sensors.

XBee I/O pins

The following table shows the I/O pins of XBee-PRO 802.15.4 modules:



Pin name	Physical pin #	Parameter
DIO0, AD0	20	D0
DIO1, AD1	19	D1
DIO2, AD2	18	D2
DIO3, AD3	17	D3
DIO4, AD4	11	D4
DIO5, AD5	15	D5
DIO6	16	D6
DIO7	12	D7
DI8	9	D8
PWM0	6	P0
PWM1	7	P1

(D = digital; I = input; O = output; AD = analog input; PWM = pulse-width modulation)

Note The number and type of IOs available can vary between different module variants.

How XBee devices get sensor data

XBee devices are often used to form **sensor networks**. In a sensor network, the main device—also called the local XBee device—receives data from the sensors attached to the remote XBee devices.

To receive that data, you must configure the remote XBee devices to "listen" on the particular pin where the sensor is connected and to send the data to the main XBee device.

Sensors

A **sensor** is a device that detects events or changes and provides a corresponding output, generally as an electrical signal.

There are two types of sensors: digital and analog. A motion sensor is a digital sensor because it can return two discrete values: movement detected or movement not detected. Other digital sensors might provide a binary value. A digital compass, for example, may provide your current heading by sending a 9-bit value with a range from 0 to 359. On the other hand, a thermometer is an analog sensor because the voltage output changes gradually as the temperature changes.

Setting pins for digital and analog sensors

Configure the pin of your XBee module according to the sensor that is connected to it:

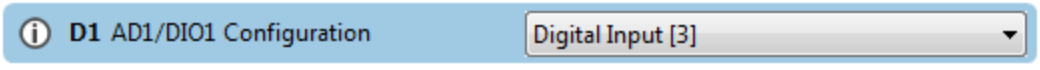
- If you connect a digital sensor, configure the pin as Digital Input.
- If you connect an analog sensor, configure the pin as Analog to Digital Converter (ADC).

Note For more information about sensors, see the How XBee devices get sensor data section.

How to configure a pin as an input

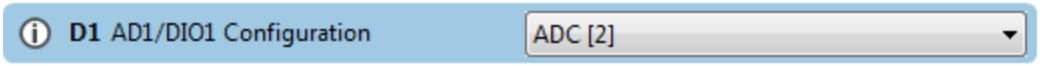
Configure a pin for digital input

You can configure a pin through XCTU. If your sensor reads digital values (like a doorbell) and is connected to the DIO1/AD1 pin, configure the D1 parameter as Digital Input [3]:



Configure a pin for analog input

If your sensor reads analog values (like a temperature sensor) and is connected to the DIO1/AD1 pin, configure the D1 parameter as ADC [2]:



How to obtain data from a sensor

There are two ways to obtain sensor information:

- Queried sampling to immediately read all enabled digital and analog input pins.
- Automatic sampling to transmit the sensor data periodically or whenever a digital pin changes.

In both cases, the information is sent to the other module is called **IO sample**. It contains which inputs (DIO lines or ADC channels) have sampling enabled and the value of all the enabled digital and analog inputs.

Queried sampling (IS)

The Force Sample (**IS**) command forces a read of all enabled digital and analog input pins. You can send it locally or to a remote device.

Use the XCTU console or any serial port terminal application to send this command.

When the module sends the **IS** command, the receiving device reads all enabled digital IO and analog input channels and returns their value. If the module transmits the **IS** command locally, it sends the IO data out the serial interface. If the module transmits the **IS** command to a remote XBee module, it sends the remote IO data over the air to the requester module.

Automatic sampling

Once you have set up the pin, the remote module must be configured to automatically transmit the sensor information to the main XBee module. The remote XBee module needs to know:

1. Where to transmit the sensor data: define this information for the module receiving this information by the destination address (**DH** + **DL**) parameters.
2. When to transmit the sensor data:
   - Periodically: The XBee can send the information read from the sensor at a specified interval.
   - By change detection: When a pin or several pins change status.

   Configure parameters IO Sampling Rate (**IR**) and Digital IO Change Detection (**IC**) to automatically transmit the sensor data.

Note These two features can work in combination with each other, depending on your requirements. For example, you could choose to receive an IO sample every minute (**IR**) but also when a certain pin changes state (**IC**).

IO Sampling Rate (IR)

The **IR** parameter sets the I/O sample rate: that is, how frequently to report the current pin state and transmit it to the destination address. The rate is set in milliseconds using hexadecimal notation. The value 0 disables the feature.

For example, if you want to transmit the sensor info every minute, set this parameter to EA60 (1 minute = 60 seconds = 60000 ms = EA60 hex).

Use XCTU to configure the sample rate interval.

Note Sleeping devices, configured to send samples periodically, transmit the first sample immediately after waking up, and then continue sending periodic IO samples at the **IR** rate, until the Time Before Sleep (**ST**) timer expires and the device can resume sleeping.

Digital IO Change Detection (IC)

The **IC** parameter allows you to set which pins to monitor for change detection. When the state of the monitored pin(s) changes, a sample is immediately sent to the destination address.

Use XCTU to set the value of **IC** parameter.

To select which pins monitor, assign a binary value to **IC** parameter based on the following pattern:

DIO12	DIO11	DIO10	DIO9	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
0	0	0	0	0	0	0	0	0	0	0	0	0

For example, if you want to monitor DIO1, the value would be **0000000000010**, which is 2 in hexadecimal notation. If you want to monitor DIO12, DIO8, DIO3 and DIO1, the value would be **1000100001010** (binary) = **110A** (hexadecimal). The value **0** disables the feature.

The Digital IO Change Detection (**IC**) feature only works for digital pins, so you will not receive anything if the value of an analog pin changes.

Example: receive digital data

This section teaches you how to create an XBee digital sensor network. Since the kit does not contain a real sensor, you simulate a digital sensor with the user button of the XBee Grove Development Board.

Note If you have a Grove sensor, you can connect it to the board for a more realistic example of a sensor network.

XBee B, where the sensor is connected, will be the sender. This XBee will transmit the value of the button to XBee A, the receiver, every time it is pressed or released.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-1.1.1.zip or later release file)
- Java Virtual Machine 6 or later
- A Java IDE (Eclipse, NetBeans, etc)

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee devices

XBee B will send the digital value from the button to XBee A every time the value changes (that is, when the button is pressed or released).

Set the destination address (**DH** + **DL**) of the sender (XBee B) to the MAC address (**SH** + **SL**) of the receiver (XBee A). Additionally, configure the pin where the button is connected (DIO4/AD4) as a digital input, and set the DIO change detect (**IC**) to monitor the same pin.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

Param	XBee A	XBee B	Effect
CH	B	B	Defines the frequency of communication. This parameter must be the same for all radios on your network.

Param	XBee A	XBee B	Effect
ID	2015	2015	Defines the network a radio will connect to. This parameter must be the same for all radios on your network.
DH	—	0013A200	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number High (SH) of the other module.
DL	—	SL of XBee A	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the other module.
MY	FFFF	FFFF	Enables the reception of packets with 64-bit addresses.
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.
AP	API enabled [1]	API enabled [1]	Enables API mode.
D4	—	DI [3]	Sets the DIO4/AD4 pin as digital input in XBee B. This pin is connected to a button.
IC	—	10	Configures XBee B to transmit an IO sample when pin DIO4 (where the button is connected) changes. 00010000 (binary) = 10 (hexadecimal) For further information on how to configure this parameter to monitor the pins, see Ob tain data from sensor topic.

3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

Receive data

In order to receive digital data from the other module, use the following steps in order.

Step 4: Create a Java project

Create an empty Java project named using Eclipse or NetBeans, with the following project name: **ReceiveDigitalData**.

Option 1: Eclipse

a. Select **File > New**, and click the **Java Project**.
b. The **New Java Project** window appears. Enter the Project name.
c. Click **Next**.

or

Option 2: NetBeans

a. Select **File > New project....**
b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.
c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.
d. Click **Finish** to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

Step 5: Link libraries to the project

This topic describes how to link the **XBee Java Library**, the **RXTX library** (including the native one), and the **logger library** to the project.

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

Option 1: Eclipse

a. Go to the **Libraries** tab of the New Java Project window.
b. Click **Add External JARs....**
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbee-java-library-X.Y.Z.jar** file.
d. Click **Add External JARs...** again.
e. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api-x.y.z.jar**
   - **slf4j-nop-x.y.z.jar**
f. Expand the **rxtx-2.2.jar** file of the Libraries tab list, select **Native library location**, and click **Edit....**
g. Click **External folder...** to navigate to the **extra-libs\native\Windows\win32** folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Replace **Windows\win32** with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

     ```
     java -version
     ```

h. Click **OK** to add the path to the native libraries.
i. Click **Finish**.

or

Option 2: NetBeans

a. From **Projects** view, right-click your project and go to **Properties**.
b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.
c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the **xbjlib-X.Y.X.ja**r file.
d. Click **Add JAR/Folder** again.
e. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api- x.y.z .jar**
   - **slf4j-nop- x.y.z .jar**
f. Select **Run** in the left tree of the **Properties** dialog.
g. In the **VM Options** field, add the following option:

```
-Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
```

where:

- **<path_where_the_XBee_Java_Library_is_unzipped>** is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
- **Windows\win32** is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```

h. Click **OK**.

Step 6: Add the source code to the project

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: MainApp.java.
2. Add the Java source file with Eclipse or NetBeans.

Option 1: Eclipse

a. In the **Package Explorer** view, select the project and right-click.
b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.
c. Type the **Name** of the class: **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

Option 2: NetBeans

a. In the **Projects** view, select the project and right-click.
b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.
c. Modify the **Class Name** to be **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

Step 7: Set the port name and launch the application

For this step, set the port name and launch the application.

1. Change the port name in the Java source code to match the port that the RECEIVER module (XBee A) is connected to.

```
// TODO: Replace with the port where your receiver module is connected
private static final String PORT = "COM1";
// TODO: Replace with the baud rate of your receiver module.
private static final int BAUD_RATE = 9600
```

2. Launch the application on your computer. Notice that every time you press or release the SENDER_1 (XBee B) or SENDER_2 (XBee C) user button, RECEIVER (XBee A) receives its status.
3. Press the button and check the received status. The output of the application should be similar to the following:
4.

```
+--------------------------------------+
|    Receive Digital Data Sample       |
+--------------------------------------+

WARNING: RXTX Version mismatch
       Jar version = RXTX-2.2pre1
       native lib Version = RXTX-2.2pre2

Listening for IO samples... Press the user button of any remote device.


Digital data from '0013A20012345678': Low (button pressed)
Digital data from '0013A20012345678': High (button released)
Digital data from '0013A20012345678': Low (button pressed)
Digital data from '0013A20012345678': High (button released)
```

What have you learned?

In this section, you have learned that:

- All XBee modules have a set of pins you can use to connect and configure sensors or actuators.
- A sensor is a device that provides a corresponding output as a response to events or changes in quantities. There are two types:

  - Digital sensors return discrete values such as on/off.
  - Analog sensors can return a wide variety of values such as the temperature of a room.
- A sensor network consists of a central module that reads data from remote nodes connected to sensors. Note that these types of networks are based on Point-to-multipoint communication.
- If, as in this example, you want to read data from a digital sensor, you must configure the selected IO as Digital Input.
- You can obtain digital data from a sensor by configuring the remote XBee to transmit the IO data:
  - To the local device by setting the **DH** and **DL** parameters to the MAC of the receiver module.
  - When a digital pin changes (using the **IC** or Digital IO Change Detection parameter) as you did in this example, or periodically (using the **IR** or IO Sampling Rate parameter).
- The data sent from one module to the other is called IO Sample. It contains the inputs (DIO lines or ADC channels) for which sampling has been enabled. It also contains the value of all enabled digital and analog inputs.

Extend the example

If you're ready to work more extensively with receiving digital data, try the following:

- Modify the example and add sleep support to the module connected to the button (XBee B). It can sleep for three seconds, then wake for one second and send the button status.
- Instead of using the button on the board, connect a digital Grove sensor (for example, a motion sensor) to the board.
- Form a larger sensor network by adding more XBee modules and configuring them to send digital data to XBee A.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

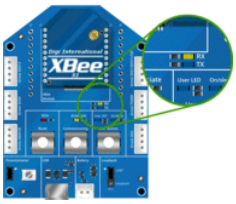
Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial

port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.

  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.

When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.

  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

  - "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
       at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
       at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
       ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example
When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example
When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example
The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

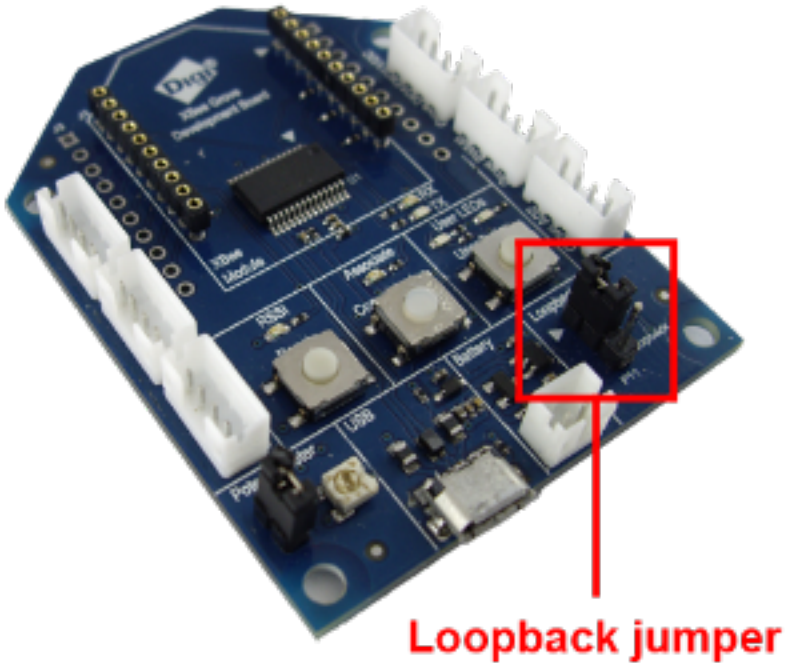
Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of

packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

Example: Receive analog data

This section demonstrates how to create an XBee sensor network. Since the kit does not contain a real sensor, you will simulate an analog sensor with the potentiometer of the XBee Grove Development Board.

Note If you have a Grove sensor, you can connect it to the board for a more realistic example of a sensor network.

XBee B, where the sensor is connected, will be the sender. This XBee will transmit the value of the potentiometer to XBee A, the receiver, every five seconds.

Note If you get stuck, go to Troubleshooting.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-1.1.1.zip or later release file)
- Java Virtual Machine 6 or later
- A Java IDE (Eclipse, NetBeans, etc)

Connect the components

If you have a Grove sensor (not included in the kit), plug it into the Grove AD2 connector of the XBee B or XBee C (senders) board. Follow the steps to connect your modules:

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. You can find more specific steps in Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee devices

XBee B will send the analog values read from the potentiometer to XBee A every five seconds.

Set the destination address (**DH** + **DL**) of the sender (XBee B) to the MAC address (**SH** + **SL**) of the receiver (XBee A). Additionally, configure the pin where the potentiometer is connected (DIO3/AD3) as an analog input, and set the sample rate parameter (**IR**) to five seconds.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.

2. Use XCTU to configure the following parameters:

Parameter	XBee A	XBee B	Effect
CH	B	B	Defines the frequency of communication. This parameter must be the same for all radios on your network.
ID	2015	2015	Defines the network a radio will connect to. This parameter must be the same for all radios on your network.
DH	—	0013A200	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number High (SH) of the other module.
DL	—	SL of XBee A	Defines the destination address to transmit the data to. The value of this setting should be the Serial Number Low (SL) of the other module.
MY	FFFF	FFFF	Enables the reception of packets with 64-bit addresses.
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the device.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.
AP	API enabled [1]	API enabled [1]	Enables API mode.
D3	-	ADC [2]	Sets the DIO3/AD3 pin as ADC in XBee B. This pin is connected to a potentiometer.
IR	-	1388	Configures XBee B to send IO samples every five seconds (5000 ms = 1388 in hexadecimal.)

Note The dash (**—**) in the table means to keep the default value. Do not change the default value.

Note If you are using a Grove sensor and have connected it to the Grove AD2 connector, you must configure the D2 parameter as ADC [2] instead of the D3.

3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

Create a Java project

Create an empty Java project named using Eclipse or NetBeans, with the following project name: **ReceiveAnalogData**.

Option 1: Eclipse

a. Select **File > New**, and click the **Java Project**.
b. The **New Java Project** window appears. Enter the Project name.
c. Click **Next**.

or

Option 2: NetBeans

a. Select **File > New project....**
b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.
c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.
d. Click **Finish** to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

Link libraries to the project

This topic describes how to link the **XBee Java Library**, the **RXTX library** (including the native one), and the **logger library** to the project.

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

Option 1: Eclipse

a. Go to the **Libraries** tab of the New Java Project window.
b. Click **Add External JARs....**
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbee-java-library-X.Y.Z.jar** file.
d. Click **Add External JARs...** again.
e. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api-x.y.z.jar**
   - **slf4j-nop-x.y.z.jar**
f. Expand the **rxtx-2.2.jar** file of the Libraries tab list, select **Native library location**, and click **Edit....**
g. Click **External folder...** to navigate to the **extra-libs\native\Windows\win32** folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
h. Replace **Windows\win32** with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```

i. Click **OK** to add the path to the native libraries.

j. Click **Finish**.

or

Option 2: NetBeans

a. From **Projects** view, right-click your project and go to **Properties**.

b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.

c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the **xbjlib-X.Y.X.ja**r file.

d. Click **Add JAR/Folder** again.

e. Go to the **extra-libs** folder and select the following files:

- **rxtx-2.2.jar**
- **slf4j-api- x.y.z .jar**
- **slf4j-nop- x.y.z .jar**

f. Select **Run** in the left tree of the **Properties** dialog.

g. In the **VM Options** field, add the following option:

```
-Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
```

where:

- **<path_where_the_XBee_Java_Library_is_unzipped>** is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
- **Windows\win32** is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```

h. Click **OK**.

Add the source code to the project

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: MainApp.java.
2. Add the Java source file with Eclipse or NetBeans.

Option 1: Eclipse

a. In the **Package Explorer** view, select the project and right-click.

b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.

c. Type the **Name** of the class: **MainApp**.

d. Click **Finish**.

e. The **MainApp.java** file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.

f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

Option 2: NetBeans

a. In the **Projects** view, select the project and right-click.
b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.
c. Modify the **Class Name** to be **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

Set the port name and launch the application

For this step, set the port name and launch the application.

1. Change the port name in the Java source code to match the port that the receiver (XBee A) is connected to.

```
// TODO: Replace with the port where your receiver module is connected
private static final String PORT = "COM1";
// TODO: Replace with the baud rate of your receiver module.
private static final int BAUD_RATE = 9600
```

Note If you are using a Grove sensor connected to AD2, make this additional code change to select the correct IO Line:

```
// Analog line to monitor.
private static final IOLine LINE = IOLine.DIO2_AD2;
```

2. Launch the application on your computer. Every ten seconds, you will receive the value of the potentiometer connected to SENDER_1 and SENDER_2. Rotate them and see that the values change.
3. The output of the application should be similar to the following:
4.

```
+--------------------------------------+
|    Receive Analog Data Sample        |
+--------------------------------------+

WARNING: RXTX Version mismatch
       Jar version = RXTX-2.2pre1
       native lib Version = RXTX-2.2pre2

Analog data from '0013A20011111111': 227
Analog data from '0013A20022222222': 0
Analog data from '0013A20011111111': 113
Analog data from '0013A20022222222': 1023
```

What have you learned?

In this section, you have learned that:

- All XBee modules have a set of pins that you can use to connect and configure sensors or actuators.
- A sensor is a device that provides a corresponding output as a response to events or changes in quantities. There are two types:
  - Digital sensors return discrete values such as on/off.
  - Analog sensors can return a wide variety of values such as the temperature of a room.
- A sensor network consists of a central module that reads data from remote nodes connected to sensors. Note that these types of networks are based on Point-to-multipoint communication.
- If, as in this example, you want to read data from an analog sensor, you must configure the selected IO as Analog to Digital Converter (ADC).
- You can obtain analog data from a sensor by configuring the remote XBee module to transmit the IO data:
  - To the local device, by setting the **DH** and **DL** parameters to the MAC of the receiver module.
  - Periodically, as you did in this example (using the **IR** or IO Sampling Rate parameter).

Note Remember that the Digital IO Change Detection (**IC**) feature only works for digital pins, so in this case you would not receive any data.

- In this case, the data sent from one module to the other is called IO Sample. It contains the inputs (DIO lines or ADC channels) for which sampling has been enabled. It also contains the value of all enabled digital and analog inputs.

Extend the example

If you are ready to work more extensively with receiving analog data, try the following:

- Modify the example and add sleep support to the module connected to the potentiometer (XBee B). It can sleep for four seconds and then wake for one second and send the potentiometer value.
- Instead of using the potentiometer on the board, connect an analog Grove sensor to the board to create a home automation system. You can monitor a number of factors, such as:
  - Temperature
  - Humidity
  - Luminance
  - CO2
  - UV
  - Gas

  You can find more analog sensors at SeeedStudio.
- Form a larger sensor network by adding more XBee modules and configuring them to send analog data to XBee A.

- Form a larger sensor network by adding more XBee modules and configuring them to send analog data to RECEIVER (XBee A).

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

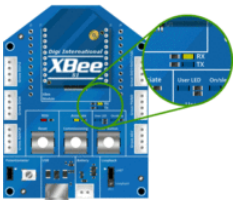
1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as

administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).

3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.

  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
```

```
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

- "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
        Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
        at java.lang.ClassLoader$NativeLibrary.load(Native Method)
        at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
        at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
        at java.lang.Runtime.loadLibrary0(Runtime.java:849)
        at java.lang.System.loadLibrary(System.java:1087)
        at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
        at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
        at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
        ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the**

project section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

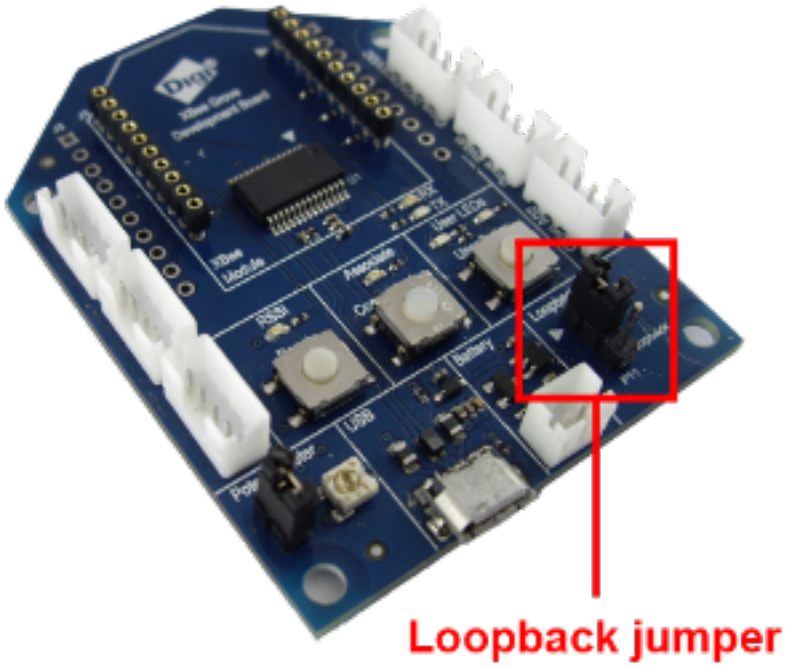
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

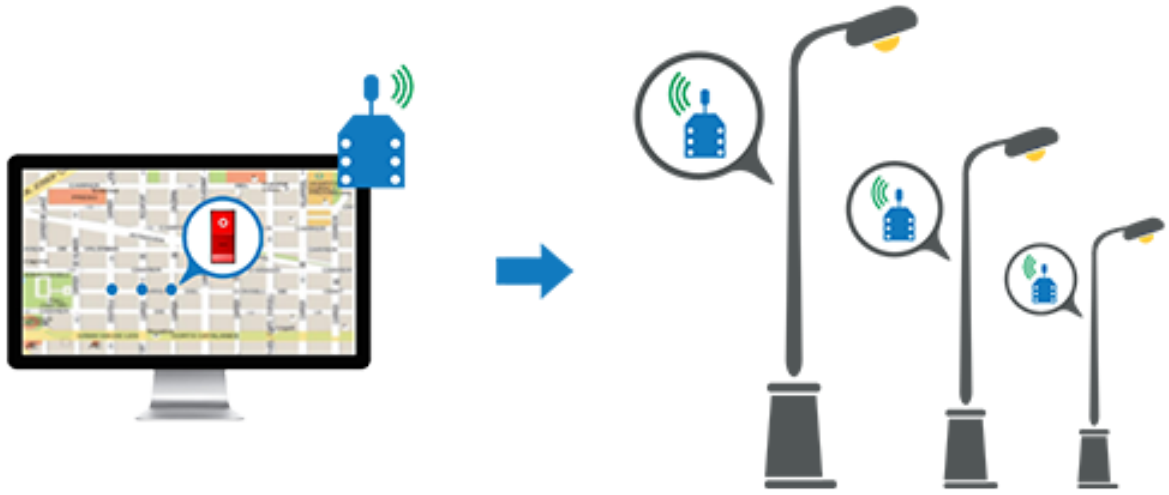
To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

How XBee modules control devices

There are many reasons to create a sensor network—that is, to collect data from multiple nodes and bring it to a central location. There are also many reasons you may want to take remote commands from a central location and create real events in multiple physical locations.

An XBee module is capable of receiving commands that set its digital and analog output pins to trigger real-world events without the use of an external microcontroller. By itself, an XBee device can power an LED, sound a small buzzer, or even operate a tiny motor. If you use a relay, you can operate many more devices directly from the XBee module.



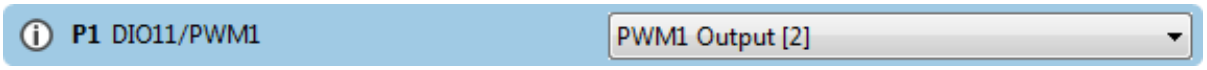
How to configure a pin as an output

Configure a pin for digital output

If you want to control a digital device, for example to switch a street lamp on or off, connect the device to a pin that supports digital output. XBee 802.15.4 devices have eight digital outputs (from **D0** to **D7**). In addition, configure that pin as Digital Output Low (Digital Out, Low [4]) or Digital Output High (Digital Out, High [5]), depending on the desired default state.

Configure a pin for analog output

If you want to control an analog device—for example, to change the brightness of an LED—connect it to a pin that supports analog output (PWM). XBee 802.15.4 RF Modules have two PWM outputs (**P0** and **P1**). Configure that pin as PWM Output [2].



Note PWM stands for pulse-width modulation, which is a way for digital devices to simulate an analog signal using a digital-to-analog converter (DAC). They do this by switching the pin between high and low with different duty cycles.

A full 3.3 volts from the XBee device has a 100% duty cycle. That is, the PWM output pin will always be high. If you want something halfway between off and on (1.65 V) the DAC keeps the pin high for 50% of the time and low for the remaining 50%. This happens very fast—about 16,000 times per second.

How to send actuations

Once the pin of a remote device is properly configured, you can send any actuation to that XBee device. You can use XCTU or the XBee Java Library.

To send an actuation in XCTU, create a Remote AT Command frame in the API console, configuring the following parameters:

- 64-bit destination address: the remote device's MAC address.
- AT command (ASCII): the AT command corresponding to the IO line you want to change.
  - For digital output lines, **D0-D7**. For example, if you want to send an actuation to DIO3, type **D3**.
  - For analog output lines (PWM), **M0-M1**. For example, if you want to send an actuation to PWM1, type **M1**.
- Parameter value (HEX): the new value for the selected IO line.
  - For digital output lines: **04** for off, **05** for on.
  - For analog output lines: a hexadecimal value between **00** and **03FF**.

Example: Send digital actuations

In this example, you will create a Java application that will blink the LED of a remote XBee module. XBee A will be the sender and will transmit the digital actuation to XBee B, which is connected to the LED.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-1.1.1.zip or later release file)
- Java Virtual Machine 6 or later
- A Java IDE (Eclipse, NetBeans, etc)

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.

3. Make sure you are in **Configuration working mode**.

Configure the XBee devices

Follow these steps to configure XBee A to send digital actuations to XBee B every second. The LED of the receiver dims with each signal received.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:
3.

Param	XBee A	XBee B	Effect
CH	B	B	Defines the frequency of communication. This parameter must be the same for all radios on your network.
ID	2015	2015	Defines the network that a radio will attach to. This must be the same for all radios in your network.
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module.  The default **NI** value is a blank space. Make sure to delete the space when you change the value.
AP	API enabled [1]	API enabled [1]	Enables API operating mode.
D4	—	Digital Out, High [5]	Sets the DIO4/AD4 pin as Digial Output High in XBee B. The LED is connected to this pin.

Note The dash (**—**) in the table means to keep the default value. Do not change the default value.

4. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

The following video demonstrates how to configure the modules with XCTU:

Create a Java project

Create an empty Java project named using Eclipse or NetBeans with the following project name: **SendDigitalActuations**.

Option 1: Eclipse

a. Select **File > New**, and click the **Java Project**.
b. The **New Java Project** window appears. Enter the Project name.
c. Click **Next**.

or

Option 2: NetBeans

a. Select **File > New project....**
b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.
c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.
d. Click **Finish** to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

Link libraries to the project

This topic describes how to link the **XBee Java Library**, the **RXTX library** (including the native one), and the **logger library** to the project.

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

Option 1: Eclipse

a. Go to the **Libraries** tab of the New Java Project window.
b. Click **Add External JARs....**
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbee-java-library-X.Y.Z.jar** file.
d. Click **Add External JARs...** again.
e. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api-x.y.z.jar**
   - **slf4j-nop-x.y.z.jar**
f. Expand the **rxtx-2.2.jar** file of the Libraries tab list, select **Native library location**, and click **Edit....**
g. Click **External folder...** to navigate to the **extra-libs\native\Windows\win32** folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
   - Replace **Windows\win32** with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

     ```
     java -version
     ```

h. Click **OK** to add the path to the native libraries.
i. Click **Finish**.

or

Option 2: NetBeans

a. From **Projects** view, right-click your project and go to **Properties**.
b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.
c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the **xbjlib-X.Y.X.ja**r file.
d. Click **Add JAR/Folder** again.
e. Go to the **extra-libs** folder and select the following files:
   - **rxtx-2.2.jar**
   - **slf4j-api- x.y.z .jar**
   - **slf4j-nop- x.y.z .jar**
f. Select **Run** in the left tree of the **Properties** dialog.
g. In the **VM Options** field, add the following option:

```
-Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-libs\native\Windows\win32
```

   where:
   - **<path_where_the_XBee_Java_Library_is_unzipped>** is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
   - **Windows\win32** is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

     ```
     java -version
     ```

h. Click **OK**.

Add the source code to the project

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: MainApp.java
2. Add the Java source file with Eclipse or NetBeans.

Option 1: Eclipse

a. In the **Package Explorer** view, select the project and right-click.
b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.
c. Type the **Name** of the class: **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

Option 2: NetBeans

a. In the **Projects** view, select the project and right-click.
b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.
c. Modify the **Class Name** to be **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

Set the port name and launch the application

For this step, set the port name and launch the application.

1. Change the port name in the Java source code to match the port that the module is connected to and the node identifier of the remote module.

```
// TODO Replace with the port where your module is connected to.
private static final String PORT = "COM1";
// TODO Replace with the baud rate of your module.
private static final int BAUD_RATE = 9600;
// TODO Replace with the node identifier (NI) of the other module.
private static final String REMOTE_NODE_ID = "XBEE_B";
```

2. Launch the application on your computer. Notice that every second, the LED of the RECEIVER module (XBee B) changes state.

What have you learned?

In this section, you have learned that:

- All XBee modules have a set of pins you can use to connect and configure sensors or actuators.
- An actuator is a device that controls a mechanism or system. For instance, you can use an XBee module connected to an actuator to send digital information to another XBee module so that it raises or lowers your window blinds.
- You can create a network with a central node that sends orders to remote nodes. This network allows you to trigger real-world events wirelessly, such as switching on all the lights at home, via actuators connected to remote node(s).
- The default state of a pin that supports digital output depends on the values of the DIO setting:
  - Digital Output, Low [4]: the output is set by default to low.
  - Digital Output, High [5]: the output is set by default to high.

Step 9: Do more with sending digital actuations

Extend the example

If you're ready to work more extensively with actuations, try the following:

- Add sensors to your network, as it is explained in the Example: receive digital data and Example: Receive analog data examples, and control your actuators depending on the value returned. For example, switch the XBee B LED on when a button connected to XBee A is pressed, or when the value of the XBee A potentiometer exceeds a defined threshold.
- Instead of controlling an LED, connect a relay to one of the digital output pins to create a home automation system. You can:
  - Switch lights on/off.
  - Switch the irrigation system of your garden on/off.
  - Raise/lower the blinds.
  - Control your garage door.
- Extend the network by adding more XBee modules connected to different devices.
- Control all your devices remotely with a smartphone application connected to an XBee Gateway.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

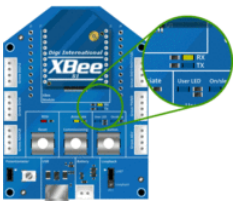
1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.

Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other

applications using the port.
Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.
Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.

- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.

  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
```

```
        at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

- "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
        Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
        at java.lang.ClassLoader$NativeLibrary.load(Native Method)
        at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
        at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
        at java.lang.Runtime.loadLibrary0(Runtime.java:849)
        at java.lang.System.loadLibrary(System.java:1087)
        at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
        at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
        at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
        ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

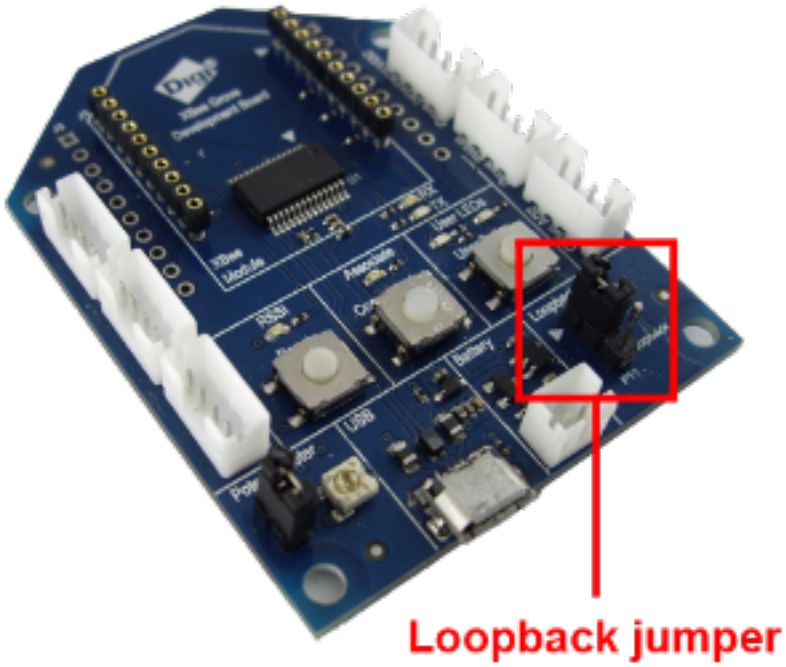
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

Lab: Send analog actuations

This section describes how to create a Java application that will dim the LED of a remote XBee module. XBee A, the sender, transmits the analog actuation to XBee B, which is connected to the LED.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards

- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later
- XBee Java Library (XBJL-1.1.1.zip or later release file)
- Java Virtual Machine 6 or later
- A Java IDE (Eclipse, NetBeans, etc)

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Configure the XBee devices

Follow these steps to configure XBee A to send analog actuations to XBee B every second. The LED of the receiver blinks with each signal received.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.
2. Use XCTU to configure the following parameters:

Param	XBee A	XBee B	Effect
CH	B	B	Defines the frequency of communication. This parameter must be the same for all radios on your network.
ID	2015	2015	Defines the network that a radio will attach to. This must be the same for all radios in your network.
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module. The default **NI** value is a blank space. Make sure to delete the space when you change the value.

Param	XBee A	XBee B	Effect
AP	API enabled [1]	API enabled [1]	Enables API operating mode.
P0	—	PWM Output [2]	Sets the PWM0 pin as PWM Output in XBee B. The LED is connected to this pin.

Note The dash (**—**) in the table means to keep the default value. Do not change the default value.

3. Write the settings of all XBee modules with the **Write radio settings** button  at the top of the Radio Configuration section.

The following video demonstrates how to configure the modules with XCTU:

Create a Java project

Create an empty Java project named using Eclipse or NetBeans, with the following project name: **SendAnalogActuations**.

Option 1: Eclipse

a. Select **File > New**, and click the **Java Project**.
b. The **New Java Project** window appears. Enter the Project name.
c. Click **Next**.

or

Option 2: NetBeans

a. Select **File > New project....**
b. The New Project window appears. In the Categories frame, select **Java > Java Application** from the panel on the right, and click **Next**.
c. Enter the Project name and the Project Location. Clear the Create Main Class option; you will create this later.
d. Click **Finish** to create the project. The window closes and the project appears in the Projects view list on the left side of the IDE.

Link libraries to the project

This topic describes how to link the **XBee Java Library**, the **RXTX library** (including the native one), and the **logger library** to the project.

1. Download the XBJL_X.Y.Z.zip library.
2. Unzip the XBJL_X.Y.Z.zip library.
3. Link the libraries using Eclipse or NetBeans:

Option 1: Eclipse

a. Go to the **Libraries** tab of the New Java Project window.
b. Click **Add External JARs....**
c. In the JAR Selection window, search the folder where you unzipped the XBee Java Library and open the **xbee-java-library-X.Y.Z.jar** file.
d. Click **Add External JARs...** again.
e. Go to the **extra-libs** folder and select the following files:
    - **rxtx-2.2.jar**
    - **slf4j-api-x.y.z.jar**
    - **slf4j-nop-x.y.z.jar**
f. Expand the **rxtx-2.2.jar** file of the Libraries tab list, select **Native library location**, and click **Edit....**
g. Click **External folder...** to navigate to the **extra-libs\native\Windows\win32** folder of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip).
h. Replace **Windows\win32** with the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is installed in your computer, open a terminal or command prompt and execute:

```
java -version
```

i. Click **OK** to add the path to the native libraries.
j. Click **Finish**.

or

Option 2: NetBeans

a. From **Projects** view, right-click your project and go to **Properties**.
b. In the categories list on the left, go to **Libraries** and click **Add JAR/Folder**.
c. In the **Add JAR/Folder** window, search the folder where you unzipped the XBee Java Library and open the **xbjlib-X.Y.X.ja**r file.
d. Click **Add JAR/Folder** again.
e. Go to the **extra-libs** folder and select the following files:
    - **rxtx-2.2.jar**
    - **slf4j-api- x.y.z .jar**
    - **slf4j-nop- x.y.z .jar**
f. Select **Run** in the left tree of the **Properties** dialog.
g. In the **VM Options** field, add the following option:

```
-Djava.library.path=<path_where_the_XBee_Java_Library_is_unzipped>\extra-
libs\native\Windows\win32
```

where:

- **<path_where_the_XBee_Java_Library_is_unzipped>** is the absolute path of the directory where you unzipped the XBee Java Library file (XBJL_X.Y.Z.zip)
- **Windows\win32** is the directory that matches your operating system and the Java Virtual Machine installed (32 or 64 bits). If you don't know which Java Virtual Machine is

installed in your computer, open a terminal or command prompt and execute:

```
java -version
```

h. Click **OK**.

Add the source code to the project

Follow these steps to add the source to the project.

1. Open the following source code, select all, and copy it to the clipboard: MainApp.java.
2. Add the Java source file with Eclipse or NetBeans.

Option 1: Eclipse

a. In the **Package Explorer** view, select the project and right-click.
b. From the context menu, select **New > Class**. The **New Java Class** wizard opens.
c. Type the **Name** of the class: **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file is automatically opened in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on that line; a pop-up appears. Select the first option (**Move 'MainApp.java' to package '...'**) to resolve the error.

Option 2: NetBeans

a. In the **Projects** view, select the project and right-click.
b. From the context menu, select **New > Java Class...** The New Java Class wizard opens.
c. Modify the **Class Name** to be **MainApp**.
d. Click **Finish**.
e. The **MainApp.java** file automatically opens in the editor. Replace its contents with the source code you copied in the previous step.
f. A line at the top of the pasted code is underlined in red. Click on the light bulb next to that line; a pop-up appears. Select the first option (**Move class to correct folder**) to resolve the error.

Set the port name and launch the application

For this step, set the port name and launch the application.

1. Change the port name in the Java source code to match the port that the module is connected to and the node identifier of the remote module.

```
// TODO Replace with the port where your module is connected to.
private static final String PORT = "COM1";
// TODO Replace with the baud rate of your module.
private static final int BAUD_RATE = 9600;
// TODO Replace with the node identifier (NI) of the other module.
private static final String REMOTE_NODE_ID = "XBEE_B";
```

> **Tip** Make sure the REMOTE_NODE_ID constant is the Node Identifier (NI) of the OTHER XBee and NOT the NI of the XBee connected to the port specified in the PORT constant.

2. Launch the application on your computer. Notice that the LED of the RECEIVER module (XBee B) dims.

What have you learned?

In this section, you have learned that:

- All XBee modules have a set of pins that can be used to connect and configure sensors or actuators.
- An actuator is a device that controls a mechanism or system. For instance, you can use an XBee module connected to an actuator to send digit information to another XBee module so that it raises or lowers your window blinds.
- You can create a network with a central node that sends orders to remote nodes. This network will allow you to trigger real-world events wirelessly, such as switching on all the lights at home via actuators connected to remote node(s).
- You can configure the selected pin as Analog Output (PWM Output) in order to send analog orders to dim an LED of a remote device, as in this example.
- The XBee 802.15.4 RF Modules have two PWM outputs: **P0** and **P1**.
- PWM stands for pulse-width modulation, a way for digital devices to simulate outputting analog signals using a digital-to-analog converter (DAC).

Extend the example

If you're ready to work more extensively with analog actuations, try the following:

Add sensors to your network, as it is explained in Example: receive digital data and Example: Receive analog data. Then control your actuators depending on the value returned. For example, dim the RECEIVER's (XBee B) LED based on the value read from the potentiometer connected to SENDER (XBee A).

- Continue the home automation exercise explained in the Example: Send digital actuations section by adding some of the following features:
  - Regulate the light intensity of a room.
  - Manage the temperature of your heating system.
  - Manipulate the speed the blinds go up and down.
  - Regulate the pressure of the irrigation system in your garden.
- Extend the network by adding more XBee modules connected to different devices.
- Control all your devices remotely with a smartphone application connected to an XBee Gateway. See Related products.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

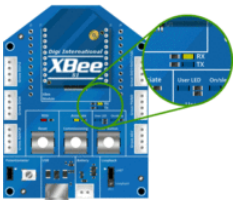
You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.
Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.
Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.
Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the

association LED on the adapter will be lit.

- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:
  - Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
  - Remove and re-insert the board into another USB port.
  - (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
  - You can download drivers for all major operating systems from FTDI for manual installation.
- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.
- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.

- Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

In this case, the library cannot access the module. Check that the PORT constant is correct.

- Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

  - "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
```

```
        at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

  If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
        at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
        at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
        at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
        ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
        at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
        at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
        at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
        at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
        at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
        at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
        at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added

to your project:

- xbee-java-library-X.Y.Z.jar
- rxtx-2.2.jar
- slf4j-api-x.y.z.jar
- slf4j-nop-x.y.z.jar

■ XBee Java Library and transparent mode

The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

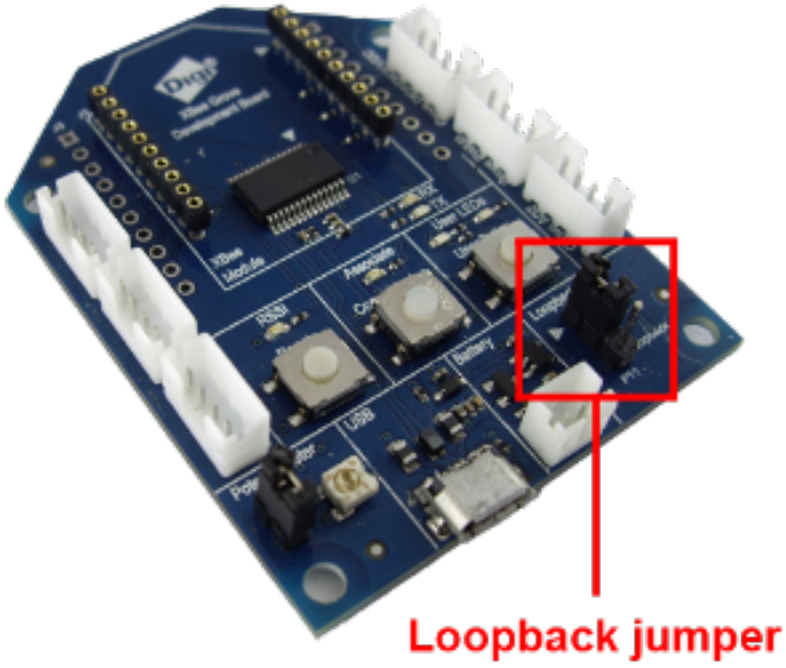
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

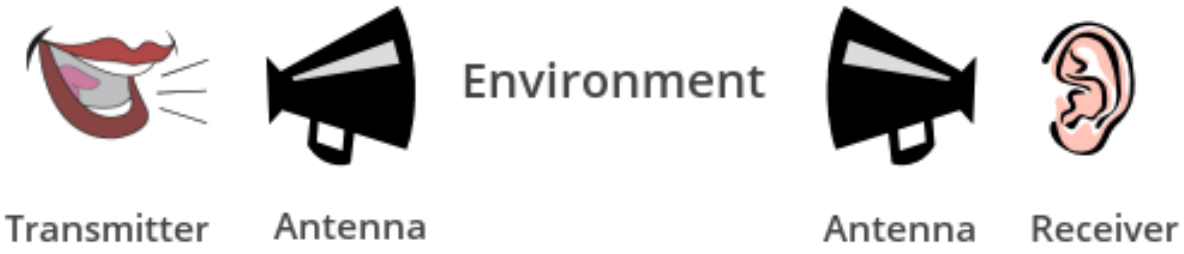
Signal strength and radio frequency range

This section describes how obstacles and other factors can impact how well the devices in your network communicate. Once you learn about the factors that can impact your signal and wireless communications, you can try performing a range test.

Distance and obstacles

Basic communication systems involve the following components:

- Transmitting element
- Receiving device
- Environment through which communication is occurring
- Antennas or other focusing elements



RF communication can be compared to simple audio communication: our vocal cords transmit sound waves that may be received by someone's eardrum. We can use a megaphone to focus and direct the sound waves in order to make the communication more efficient.

The **transmitter**'s role in wireless communication is to feed a signal to an antenna for transmission. A radio transmitter encodes data in RF waves with a certain signal strength (power output) to project the signal to a receiver.

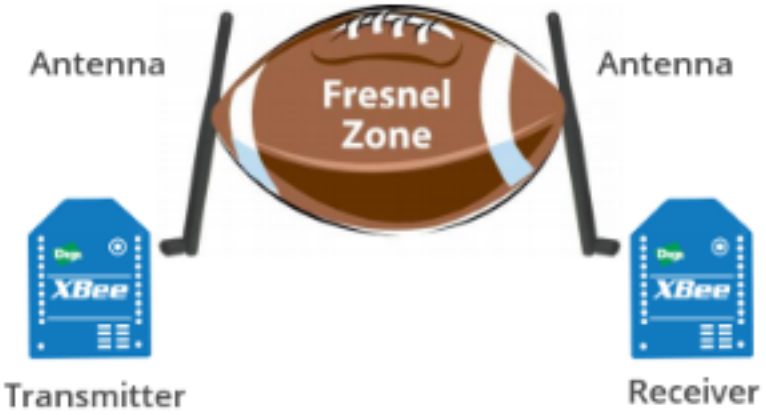
The **receiver** gets and decodes data that comes through the receiving antenna. The receiver performs the task of accepting and decoding designated RF signals while rejecting unwanted ones.

Antennas are devices that focus energy in a particular direction, similar to the way the megaphone focuses voice energy. Antennas can provide different radiation patterns depending on the design and application. How much the energy is focused in a given direction is referred to as antenna gain.

The space between the transmitter and receiver is the system's **environment**. Attaining RF line-of-sight (LOS) between sending and receiving antennas is essential to achieving long range in wireless communication. There are two types of LOS that are generally used to describe an environment:

- **Visual LOS** is the ability to see from one site to the other. It requires only a straight linear path between two points.
- **RF LOS** requires not only visual LOS, but also a football-shaped path called a **Fresnel zone** that is free of obstacles so data can travel optimally from one point to another. The Fresnel

zone can be thought of as a tunnel between two sites that provides a path for RF signals.

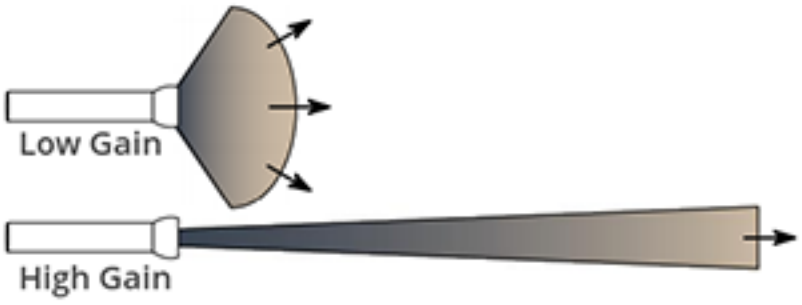


Factors affecting wireless communication

Although the communication distance specified for some XBee devices can be up to 25 miles or more, this value may be affected by factors that may decrease the quality of the signal:

- **Some materials can reflect the radio frequency waves, causing interference** with other waves and loss of signal strength. In particular, metallic or conductive materials are great reflectors, although almost any surface can reflect the waves and interfere with other radio frequency waves.
- **Radio waves can be absorbed by objects in their path, causing loss of power** and limiting transmission distance.
- **Antennas can be adjusted to increase the distance that data can travel in a wireless communication system**. The more focus the antenna can apply, the more range the system will yield. High-gain antennas can achieve greater range than low-gain antennas, although they cover less area.

  A flashlight can help illustrate the principle. Some flashlights allow the user to adjust the beam of light by twisting the lens to focus or spread the beam of light. When the lens spreads—or diffuses—the beam of light, that beam of light travels a shorter distance than when the lens is twisted to focus the beam of light.



- **Line-of-sight can help increase reliability of the signal**.

  To achieve the greatest range, the football-shaped path in which radio waves travel (Fresnel zone) must be free of obstructions. Buildings, trees, or any other obstacles in the path will decrease the communication range. If the antennas are mounted just off the ground, over half

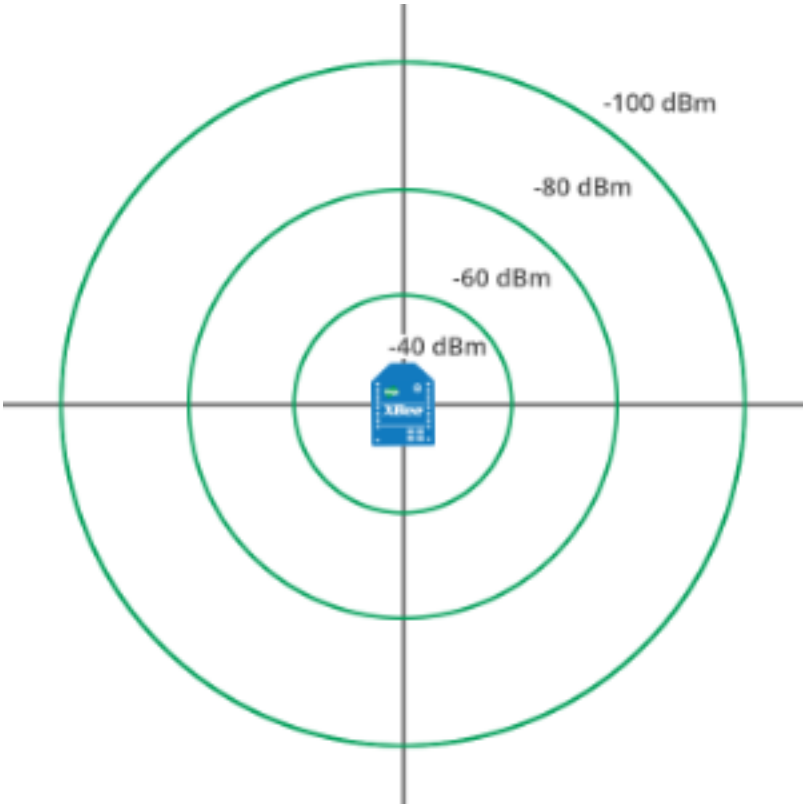
of the Fresnel zone ends up being obstructed by the curvature of the earth, resulting in significant reduction in range. To avoid this problem, mount the antennas high enough off the ground that the earth does not interfere with the central diameter of the Fresnel zone.

Signal strength and the RSSI pin

The Received Signal Strength Indicator (RSSI) measures the amount of power present in a radio signal. It is an approximate value for signal strength received on an antenna.

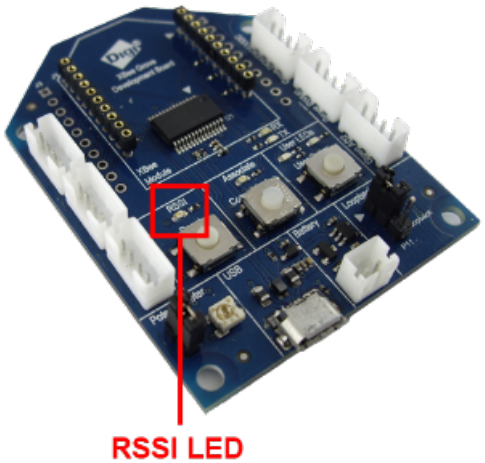
Measuring the signal strength at the receiving antenna is one way to determine the quality of a communication link. If a distant transmitter is moved closer to a receiver, the strength of the transmitted signal at the receiving antenna increases. Likewise, if a transmitter is moved farther away, signal strength at the receiving antenna decreases.

The RSSI is measured in dBm. A greater negative value (in dBm) indicates a weaker signal. Therefore, -50 dBm is better than -60 dBm.



XBee module's pin 6 can be configured as an RSSI pin that outputs a PWM (pulse-width modulation) signal representing this value. To do so, configure **P0** as RSSI [1]:

The XBee Grove Development Board includes an LED connected to the XBee module's pin 6. When this pin is configured as the RSSI pin, the LED lights every time the connected XBee module receives data. Its intensity represents the RSSI value of the last-received data: a brighter light means a higher RSSI value and better signal quality.



Configure the amount of time the RSSI pin is active, and therefore the amount of time the LED will remain lit, by modifying the RSSI PWM Timer (RP) setting:

RP value is expressed in hexadecimal notation. For example, a configured value of 0x1E is equivalent to 30 in decimal and means that the pin will be active for three seconds (30*100=3000ms.) So the LED will light for a total of three seconds, representing the last RSSI value.

After the **RP** time has elapsed and no data has been received, the pin will be set to low and the LED will not light until more data is received. The pin will also be set to low at power-up until the first data packet is received. A value of 0xFF permanently enables the pin; when configured in this way, it will always reflect the RSSI value of the last-received data packet.

Although the luminosity variations of the RSSI LED may be difficult to distinguish, the LED can be used to verify successful receipt of data packets. Each time the XBee module receives data, the LED is solid during the configured time.

Note Received Signal Strength (DB) parameter

The RSSI value can also be obtained by reading the XBee **DB** parameter value. It represents the RSSI absolute value of the last received data packet expressed in hexadecimal notation.

Is RSSI the best indication of link quality?

One thing to keep in mind is that the RSSI is only an indication of the RF energy detected at the antenna port. The power level reported could be artificially high because it may include energy from background noise and interference as well as the energy from the desired signal. This situation is worse in an interference-prone environment where it is possible to get consistently high RSSI readings, yet still have communication errors.

If the application is attempting to measure "link reliability" and not just "signal strength," it may be helpful to factor in "% packets received" or similar data.

Tip A range test is always a good idea, as it allows you to measure link performance in terms of signal strength and packet success rate. This will help you determine the reliability of your RF system. For more information, see perform a range test.

Range test

Since the communication between XBee modules takes place over the air, the quality of the wireless signal can be affected by many factors: absorption, reflection of waves, line-of-sight issues, antenna style and location, etc.

A range test demonstrates the real-world RF range and link quality between two XBee modules in the same network. Performing a range test will give an initial indication of the expected communication performance of the kit components.

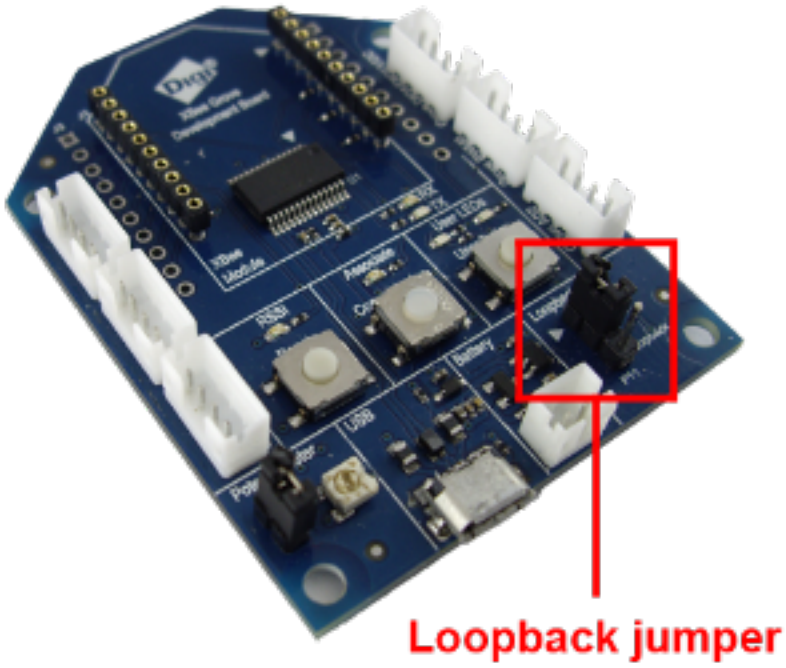
When deploying an actual network, multiple range tests are recommended to analyze varying conditions in your application.

XCTU allows you to perform a range test with at least one XBee module connected to your computer (local) and another remote XBee module, both in the same network. The range test involves sending data packets from the local XBee module to the remote and waiting for the echo to be sent from the remote to the local. During this process, XCTU counts the number of packets sent and received by the local module and measures the signal strength of both sides (RSSI):

- RSSI is the Received Signal Strength Indicator value.
- Every sent packet from the local XBee module should be received again as an echo by the same local XBee module.

There are two types of range tests:

- **Loopback cluster** (0x12): The range test is performed using explicit addressing frames/packets directed to the Cluster ID 0x12 on the data endpoint (0xE8) which returns the received data to the sender. Not all XBee variants support the Loopback Cluster. XCTU Range Test tool displays an error when this method is selected and the XBee module does not support it.
- **Hardware loopback**: The range test is performed using the serial port/USB hardware loopback capabilities. To use this type, the remote module must be configured to work in transparent mode and the loopback jumper must be closed before starting. This causes any received data to be transmitted back to the sender.

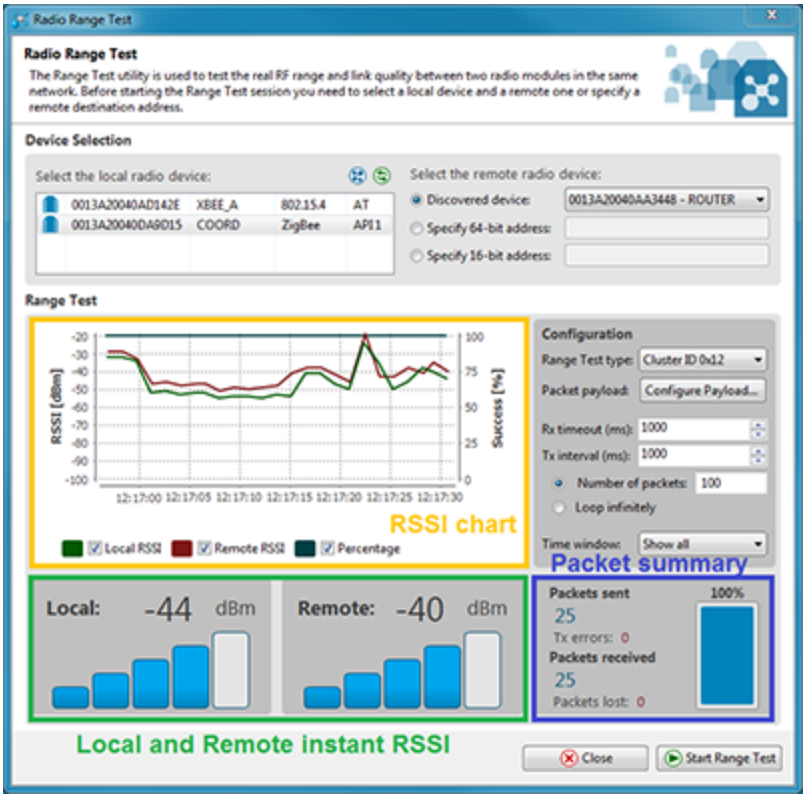


Note The local XBee module (the one attached to your computer) can be configured to use API or transparent mode. The RSSI value of the remote device can only be read when the local XBee module is working in API mode.

Once the range test process has started, XCTU represents the retrieved data in three ways:

- **RSSI Chart** represents the RSSI values of the local and remote devices during the range test session. The chart also contains the percentage of success for the total packets sent.
- **Local and Remote instant RSSI value** display the instant RSSI value of the local and remote devices. This value is retrieved for the last packet sent/received.
- **Packet summary** displays the total number of packets sent, packets received, transmission errors, and packets lost. It also displays the percentage of success sending and receiving

packets during the range test session.



Note For more information about the range test tool, read the XCTU documentation.

perform a range test

Follow the steps in this section to perform a range test with XCTU using the loopback cluster of your XBee modules.

The steps show you how to review the RSSI of both local and remote modules and the number of packets successfully sent and received by the local module during the range test session.

To perform a range test with 802.15.4 modules:

- The remote XBee must be configured to work in transparent mode.
- The remote XBee loopback jumper must be closed before starting. This will cause any received data to be transmitted back to the local device.
- The local XBee (the one attached to your computer) can be configured to use API or transparent mode. The RSSI value of the remote device can only be read when the local XBee is working in API mode.

Tip Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode. Also, note that you will have to connect the loopback jumper in the remote device before starting the range test.

Requirements

Hardware

- Two XBee 802.15.4 radios
- Two XBee Grove Development Boards
- Two micro USB cables
- One computer, although you may also use two

Software

- XCTU 6.3.1 or later

Connect the components

To get started, connect the components and start XCTU.

1. Plug the XBee modules into the XBee Grove Development Boards and connect them to your computer using the micro USB cables provided. For more information, see Plug in the XBee module.
2. After connecting the modules to your computer, open XCTU.
3. Make sure you are in **Configuration working mode**.

Step 3: Configure the XBee Zigbee modules

Both modules must be in the same network in order to communicate. This means their Channel (**CH**) and Network ID (**ID**) must be the same.

Configure the Node Identifier (**NI**) of the XBees to easily identify each module as XBEE_A and XBEE_B. XBEE_A will be the local device and XBEE_B the remote.

In order to get all possible data including the RSSI value of the remote XBee, the local device (XBEE_A) will work in API mode.

Note The local module (XBEE_A) can also work in transparent mode, but the value of the remote RSSI will not be read.

1. Restore the default settings of all XBee modules with the **Load default firmware settings** button at the top of the Radio Configuration section.

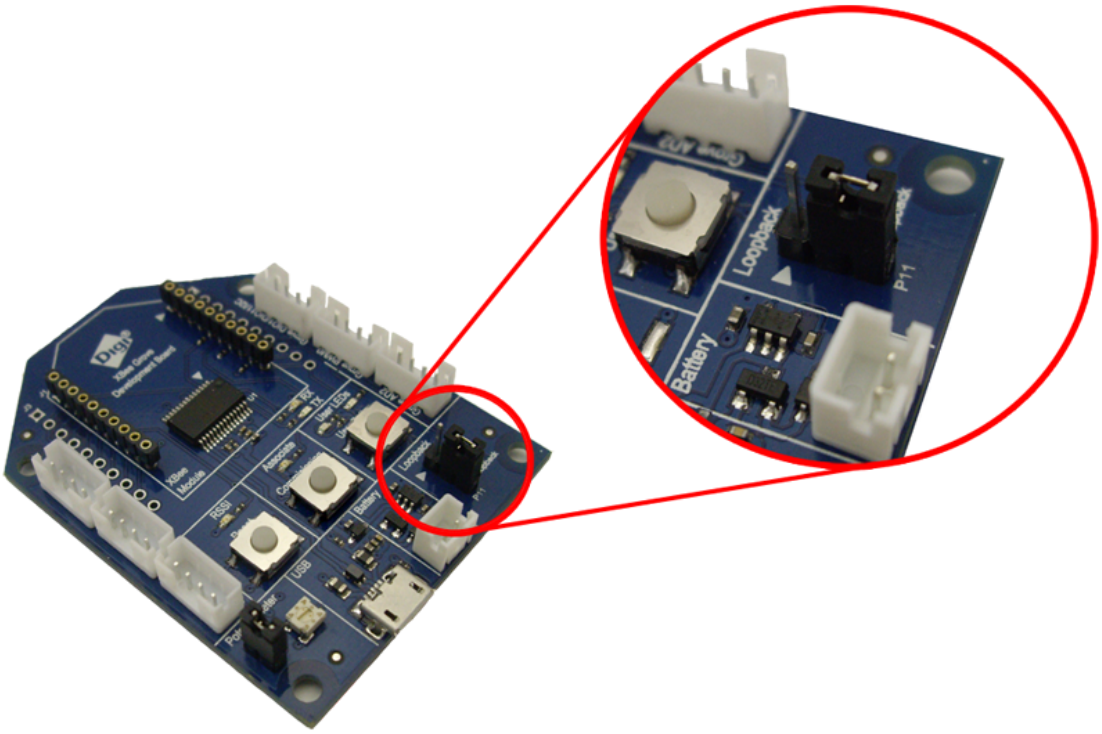
2. Use XCTU to configure the following parameters:

Param	XBee A (local)	XBee B (remote)	Effect
CH	B	B	Defines the frequency of communication. This parameter must be the same for all radios on your network.
ID	2015	2015	Defines the network a radio will connect to. This parameter must be the same for all radios on your network.
NI	XBEE_A	XBEE_B	Defines the node identifier, a human-friendly name for the module. ⚠ The default NI value is a blank space. Make sure to delete the space when you change the value.
AP	API enabled [1]	API enabled [1]	Enables/disables API mode.

3. Write the settings of all XBee modules with the **Write radio settings** button at the top of the Radio Configuration section.

Connect the loopback jumper

You will use the XBEE_A module as the local device. If you have two computers, connect XBEE_B to the other computer. If you have only one computer, keep both modules connected to it.



You also need to connect the loopback jumper in the remote device (XBEE_B) before starting the range test.

Perform a range test

Follow these steps to perform the range test.

1. Open the **Tools** menu within XCTU and select the **Range Test** option.
2. Your local devices are listed on the left side of the Devices Selection section. Select the XBEE_A module. Discovery of remote devices starts..
3. When the discovery process finishes, the other device (XBEE_B) is displayed in the Discovering remote devices... dialog. Click **Add selected devices**.
4. Select the XBEE_B module from the Discovered device list located on the right panel inside.
5. Choose Loopback in the **Range Test type** drop down.
6. Click **Start Range Test**.
7. A notification dialog will ask you to close the loopback jumper in the remote device, which is XBEE_B in this example. You already closed it in the previous step, so click OK to begin.
8. Range test data is represented in the chart. By default, 100 packets are being sent for the test. The instant local and remote RSSI are also displayed in two separate controls, as well as the number of packets sent and received.
9. Test the signal attenuation by doing one of the following: place your hands over one of the modules, block line-of-sight with your body, place a metal box over an XBee module, or move the remote XBee module to a further location like a different room or floor of the building. The RSSI value will decrease and some packets may even be lost.
10. Click **Stop Range Test** to stop the process at any time.

Tip Do not forget to restore the loopback jumper in the remote device, which is XBEE_B in this example. You already closed it in the previous step, so click **OK** to begin.

The following video shows you how to perform a range test with XCTU:

What have you learned?

In this section, you have learned the following:

- There are two types of line-of-site (LOS) that describe an environment:
  - Visual LOS describes the ability to see from one place to the other. It requires only a straight linear path.
  - RF LOS requires visual LOS and a Fresnel zone free of obstacles for data to travel optimally.
- Almost any surface, especially metallic or conductive materials, can cause interference and a resulting loss of quality in transmitted data.
- Adjust antennas to increase the distance data can travel. Place them high up off the ground to help increase their range.
- The RSSI or Received Signal Strength Indicator measures the amount of power present in a radio signal. It is measured in dBm, and its depends on the protocol.
- XBee pin 6 (**P0**) can be configured as RSSI. The Grove board includes an LED connected to it which visually represents the RSSI value of the last-received data by varying its light intensity.
- The RSSI PWM Timer (**RP**) parameter helps you configure the amount of time the RSSI pin is active.
- The Received Signal Strength (**DB**) parameter represents the absolute RSSI value, in hexadecimal notation, of the last data packet received.
- A range test determines the real-world RF range and link quality between two XBee modules in the same network by sending data packets from the local device to the remote device and waiting for an echo.

Troubleshooting

If you are encountering problems, these suggestions may help:

General

Can't find module's serial port

You can remove the XBee Grove Development Board from the USB port and see which port name disappears from your port list. The name that disappears is your XBee board.

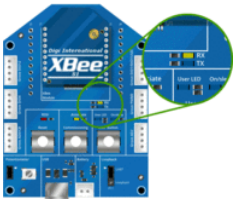
You may be able to figure out which port is right via trial and error, but you can also use XCTU to find it:

1. Open XCTU and discover the radio modules attached to your computer by clicking on the top-left corner.
2. Select all ports to be scanned.
3. Click **Next** and then **Finish**.

Once the discovery process has finished, a new window notifies you of the discovered devices and their details. The serial port and the baud rate are shown in the Port label.
Can't identify XBee modules

Once you have added the modules to XCTU, a simple way to identify them is to read the radio settings of each one and check the Rx and Tx LEDs of the XBee Grove Development Boards. These LEDs indicate that the XBee module is receiving (Rx) or transmitting (Tx) information through the serial port.

When you read or write the settings of a module, its Rx and Tx LEDs blink, so you can identify which module is connected to each serial port.



Error: Port is already in use

The serial port where the local XBee module is connected can only be in use by one application. Check that the connection with the module in the XCTU console is closed and there are no other applications using the port.

Error: Device driver was not successfully installed

Sometimes when you connect the XBee Grove Development Board into you computer, the operating system cannot install the driver automatically. If you get that error, try to remove and re-insert the board into your computer. If the OS is still unable to install the driver, remove and re-insert the board into another USB port.

As a last resort, see Manually install USB drivers.

XCTU

Error upon installation of XCTU

XCTU requires Administrator permissions. Check that you have Administrator access on the machine where you are installing XCTU. You may need to request permission to install or run applications as administrator from your network manager.

On Windows systems, a User Account Control dialog may appear when you install XCTU or try to run the XCTU program. You must answer **yes** when prompted to allow the program to make changes to your computer, or XCTU will not work correctly.

Discovery process either does not find devices, or XCTU does not list serial ports

There are several troubleshooting methods to try under these circumstances:

- **Check cables**: Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.
- **Check that the XBee is fully seated in the XBee Grove Development Board**: When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.
- **Check the XBee orientation**: The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.
- **Check driver installation**: Drivers are installed the first time the XBee Grove Development Board is plugged in. If this process is not complete or has failed, try the following steps:

- Remove and re-insert the board into your computer. This may cause driver installation to re-occur.
- Remove and re-insert the board into another USB port.
- (Windows) Open Computer management, find the failing device in the Device Manager section and remove it.
- You can download drivers for all major operating systems from FTDI for manual installation.

- **Check whether the modules are sleeping**: The On/Sleep LED of the XBee Grove Development Board indicates if the module is awake (LED on) or asleep (LED off). When a module is sleeping, it cannot be discovered in XCTU; press the **Commissioning** button to wake a module for 30 seconds.

XCTU reports errors for KY and DD settings after resetting to factory defaults

This is a known issue with XCTU version 6.1.2 and earlier. When the Invalid settings dialog appears, it is safe to continue to write settings.

- AES Encryption Key (KY) is a setting that must be set by the user when encryption is used and does not apply with factory settings.
- Device Type Identifier (DD) is a diagnostic parameter which is not used in the operation of the radio and can safely be set to any value.

Chat example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Verify the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be the Node Identifier (NI) of the other module.
When I launch the example, the message "Error: the value of the REMOTE_NODE_ID constant must be the node identifier of the OTHER module" appears.

This message indicates that the value of the REMOTE_NODE_ID constant that appears in the Java source code is not correct. This value must be the Node Identifier (NI) of the other module.

Advanced Chat example

When I launch the example, the message "Error parsing the text..." appears.

This message indicates that you typed the message incorrectly. Remember that you have to follow this pattern when sending a message:

- Unicast: NODE_IDENTIFIER: message
  For example, to send the message "Hi XBee" to XBEE_B:

```
XBEE_B: Hi XBee
```

- Broadcast: ALL: message
  For example, to send the message "Hi XBees" to all nodes of the network:

```
ALL: Hi XBees
```

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not send the message to the device whose node identifier is <XXXX>.

Ensure that you typed the node identifier correctly and that it is in the same network as your local device (same CH and ID).

Receive digital data example

The example does not show any digital data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D4 is DI [3].
6. The value of IC is 10 to monitor changes in DIO4 pin (00010000 binary = 10 hexadecimal).

Tip To learn how to configure IC parameter to monitor the pins, see How to obtain data from a sensor.

XBee Java library

- Warning message: RXTX version mismatch

  If you launch an application and see the message "WARNING: RXTX version mismatch", this indicates that the versions of the JAR file and the native library are not the same. You can safely ignore this message.
- Invalid operating mode exception message

  If you launch an application and you see the "com.digi.xbee.api.exceptions.InvalidOperatingMode" exception, review the following solutions.
  - Could not determine operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Could not
determine operating mode.
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:211)
       at com.digi.xbee.sendreceivedatasample,MainApp.main(MainApp.java:43)
```

  In this case, the library cannot access the module. Check that the PORT constant is correct.
  - Unsupported operating mode:

```
com.digi.xbee.api.exceptions.InvalidOperatingModeException: Unsupported
operating mode: AT mode
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:214)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

This indicates the module is in transparent mode. Change the AP parameter through XCTU to be API enabled [1].

- Java lang unsatisfied exception message

  If you experience the "java.lang.UnsatisfiedLinkError" exception, there are several possibilities.

  - "no rxtxSerial in java.library.path thrown while loading gnu.io.RXTXCommDriver":

```
java.lang.UnsatisfiedLinkError: no rxtxSerial in java.library.path thrown
while loading gnu.io.RXTXCommDriver
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1878)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library is not linked to the rxtx-2.2.jar file. See the second step of the Link the libraries to the project section.

  - "Can't load AMD 64-bit .dll on a IA 32-bit platform thrown while loading gnu.io.RXTXCommDriver" (or similar message):

```
java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform thrown
while loading gnu.io.RXTXCommDriver
       Exception in thread "main" java.lang.UnsatisfiedLinkError:
C:\Users\user\workspace\SendReceiveDataSample\libs\native\Windows\win64\r
xtxSerial.dll: Can't load AMD 64-bit .dll on a IA 32-bit platform
       at java.lang.ClassLoader$NativeLibrary.load(Native Method)
       at java.lang.ClassLoader.loadLibrary1(ClassLoader.java:1957)
       at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1882)
       at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1872)
       at java.lang.Runtime.loadLibrary0(Runtime.java:849)
       at java.lang.System.loadLibrary(System.java:1087)
       at gnu.io.CommPortIdentifier.<clinit>(CommPortIdentifier.java:123)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:161)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
```

  This exception indicates that the RXTX native library you have linked is not the correct one. Check to be sure you aren't using a 32-bit JVM linked the 64-bit library, or vice versa.

- Interface in use exception message

If you experience the "com.digi.xbee.api.exceptions.InterfaceInUseException" exception, it indicates that the port you are trying to open is already in use. Ensure that you don't have any applications running and that the XCTU console of that port is not connected.

```
com.digi.xbee.api.exceptions.InterfaceInUseException: Port COM5 is
already in use by other application(s)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:189)
       at com.digi.xbee.api.XBeeDevice.open(XBeeDevice.java:189)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:43)
Caused by: gnu.io.PortInUseException: Unknown Application
       at at gnu.io.CommPortIdentifier.open(CommPortIdentifier.java:467)
       at at com.digi.xbee.api.connection.serial.SerialPortRxTx.open
(SerialPortRxTx.java:167)
       ... 2 more
Error 0x5 at ..\src\termios.c(892): Access is denied.
```

- Java lang no class definition exception message

  If you experience the "java.lang.NoClassDefFoundError" exception, it indicates that the logger library (**slf4j-api-1.7.7.jar**) is not linked to the project. See the **Link the libraries to the project** section to know which libraries you have to link.

```
Exception in thread "main" java.lang.NoClassDefFoundError:
org/slf4j/LoggerFactory
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:170)
       at com.digi.xbee.api.connection.serial.AbstractSerialPort.<init>
(AbstractSerialPort.java:136)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:149)
       at com.digi.xbee.api.connection.serial.SerialPortRxTx.<init>
(SerialPortRxTx.java:124)
       at com.digi.xbee.api.XBee.createConnectiontionInterface(XBee.java:38)
       at com.digi.xbee.api.AbstractXBeeDevice.<init>
(AbstractXBeeDevice.java:164)
       at com.digi.xbee.api.XBeeDevice.<init>(XBeeDevice.java:90)
       at com.digi.xbee.sendreceivedatasample.MainApp.main(MainApp.java:40)
```

- SLF4J class path contains multiple bindings message

  If you receive the "SLF4J: Class path contains multiple SLF4J bindings" message, it indicates that you linked several logger libraries. Ensure that only the following four libraries are added to your project:

  - xbee-java-library-X.Y.Z.jar
  - rxtx-2.2.jar
  - slf4j-api-x.y.z.jar
  - slf4j-nop-x.y.z.jar

- XBee Java Library and transparent mode

  The XBee Java Library only supports API and API escaped operating modes. You cannot use it with modules in transparent mode.

Receive analog data example

The example does not show any analog data.

Check the following in XBee A (receiver):

1. The value of CH is B (the same as the CH value of XBee B).
2. The value of ID is 2015 (the same as the ID value of XBee B).

Check the following in XBee B (sender):

1. The value of CH is B (the same as the CH value of XBee A).
2. The value of ID is 2015 (the same as the ID value of XBee A).
3. The value of DH is 0013A200.
4. The value of DL is SL of XBee A (receiver).
5. The value of D3 is ADC [2].
6. The value of IR is is 1388 (5 seconds).

Tip To learn how to configure IR parameter to monitor the pins, see How to obtain data from a sensor.

Send digital actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not blink.

Check the following:

1. In the XBee B (receiver), the value of the D4 setting is DO High [5].
2. The LINE constant that appears on the Java source code is IOLine.DIO4_AD4.

Send analog actuations example

When I launch the example, the message "Could not find the module <XXXX> in the network" appears.

This message indicates that the module attached to your computer could not establish the connection with the device whose node identifier is REMOTE_NODE_ID. Check the following:

1. Both modules are in the same Channel (CH).
2. Both modules are in the same PAN ID (ID).
3. The Node Identifier (NI) of the XBee A (sender) is XBEE_A and the XBee B (receiver) is XBEE_B.

Tip The default NI value is a blank space. Make sure to delete the space when you change the value.

In addition, the value of the REMOTE_NODE_ID constant that appears in the Java source code should be XBEE_B.

The LED does not dim.

Check the following:

1. In the XBee B (receiver), the value of the P0 setting is PWM Output [2].
2. The LINE constant that appears on the Java source code is IOLine.DIO10_PWM0.

Range test example

The error 'In 802.15.4 protocol the destination device must be running in AT (Transparent) mode' is displayed and I cannot continue.

Range test using 802.15.4 XBees is only supported if the remote device is working in transparent mode.

You must reconfigure the remote device to work in transparent mode:

Parameter	Value	Effect
AP	API disabled [0]	Disables API mode to work in transparent mode.

There are no remote devices to select.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check that the XBees are in the same network

Check that the Channel (CH) value is the same for both XBees. Check that the PAN ID (ID) has the same value for both XBees

Restore default settings

If the XBees are properly connected and in the same network, restore default settings and configure them again.

The local RSSI and the number of packets received are always 0.

Check cables

Double check all cables. The USB cable should be firmly and fully attached to both the computer and the XBee Grove Development Board. When attached correctly, the association LED on the adapter will be lit.

Check that the XBee is fully seated in the XBee Grove Development Board

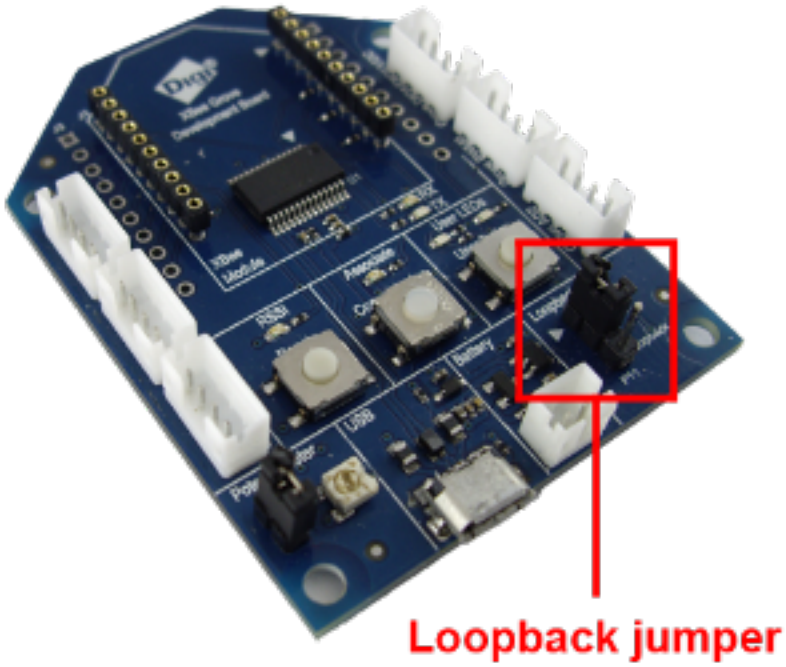
When the XBee is correctly installed, it should be pushed fully into the board and no air or metal should be visible between the plastic of the adapter socket and the XBee headers. Also, double check that all ten pins on each side of the XBee made it into a matching hole in the socket.

Check the XBee orientation

The angled "nose" of the XBee should match the lines on the silk screening of the board and point away from the USB socket on the XBee Grove Development Board.

Check the loopback jumper

The remote device loopback jumper must be closed before starting the range test. Otherwise, the packets sent from the local device will not be echoed by the remote device. This means the number of packets received will be always 0 and the RSSI therefore cannot be measured because no packets are being received.



Remote RSSI is not included in the chart and the Remote RSSI control is disabled.

The local device (the one attached to your computer) can be configured to use API or transparent mode. The remote device RSSI value can only be read when the local XBee is working in API mode.

To display the remote RSSI value, reconfigure the local module to work in API mode.

Parameter	Value	Effect
AP	API enabled [1]	Enables API mode.

Additional resources

Wireless connectivity offers almost unlimited options for making our surroundings smarter, more efficient, and more connected. Now that you have completed the activities in this kit, here are some additional resources to help you explore XBee modules.

Buying considerations

You have become familiar with the XBee modules included in the kit, but Digi makes a large variety of modules with different features and for different functions. So, which module is best suited for your applications? Why are there different types of antennas? Should you use a "PRO" version, or is a regular XBee module enough? In this guide, we look over the different XBee options to help you answer these questions.

Note that the XBee module you select affect the parameters of your application:

- The location of your application affects the operating frequency of the XBee modules.
- To get greater range, you may select an external antenna, a different operating frequency, or even an XBee-PRO.
- Power consumption is an important factor to consider.
- The required network topology also impacts the type of module you need.

To help you select an XBee module based on your requirements, see the XBee Buying Guide.

The following sections review the different options available for XBee radios and how they affect wireless communication.

Note Not all options are available for every XBee device.

Where to buy XBee devices

We are committed to providing our customers with local sales and support across the globe. With our reach extending to more than 70 countries worldwide, we work with a number of channel partners in local countries to provide you with excellent sales and support.

Advantages of working with our network of international channel partners include:

- Local language contacts within the organization.
- In-country and local language technical support and customer service.
- In-country RMA support.
- In-country product delivery.
- Understanding of local businesses and industry segments.

Contact us online or by phone at 1-952-912-3444 for more information about which channel partner is best suited to your individual needs.

Find products from Digi and Digi distributors

Digi products are available from many sources worldwide.

- You can find Digi products through distributors. Find a distributor now.
- You can also find Digi products in our official Digi online store:
  - Americas online store (U.S., Canada, and Latin America)
  - EMEA online store (Europe, Middle East, and Africa)
  - Japan online store
  - U.S. Government Sales

Find Digi products through resellers

You can purchase XBees and other Digi networking products from online retailers:

- Adafruit
- Fry's
- Maker Shed
- Microcenter
- Parallax
- RobotShop
- Seeed Studio
- Solarbotics
- Sparkfun
- TrossenRobotics

Real projects with XBee modules

Explore the links below for real-world projects made with XBee technology.

Community

XBee Projects

The largest collection of XBee projects on the web.

Digi XBee examples and guides

Learn more about wirelessly connecting XBees to sensors, outputs, motors, lights, and the Internet.

Industrial solutions

Wireless Tank Monitoring with 1844myfuels
Wireless ultrasonic sensors connected to XBee modules enable up-to-the-minute monitoring of farm silo levels.

Devergy Expands Solar Power Possibilities in Africa
Devergy uses XBee technology for the communication network, where hundreds of nodes are connected with XBee modules—making the solar micro-grids smart, cost effective and manageable.

Tracking Hand Washing Decreases the Spread of Infection at Hospitals
Hand washing is one of the most important daily routines to avoid the spreading of bacteria and disease, especially in hospitals and healthcare centers.

XBee helps Libelium monitor harsh environments
Embedded XBee and XBee-PRO modules enable low-cost, low-power remote monitoring of isolated and difficult-to-access sensors.

For more industrial solutions, visit www.digi.com/industries.

Related products

See the following products from Digi International.

XBee Gateway

The low-cost XBee-to-IP solution enables remote connectivity, configuration, and management of XBee networks with Digi Remote Manager. All XBee data sent to the gateway is automatically available to online applications via Digi Remote Manager. Additionally, this gateway can run custom Python applications that communicate and manage your XBee network.

XBee RF Modems

XBee RF Modems are small, low-power devices using XBee RF modules to communicate with systems using RS-232, RS-485, and USB interfaces. You can easily make existing wired systems wireless with this out-of-box solution. XBee RF modems are ideal for extended-range applications with a high data throughput.