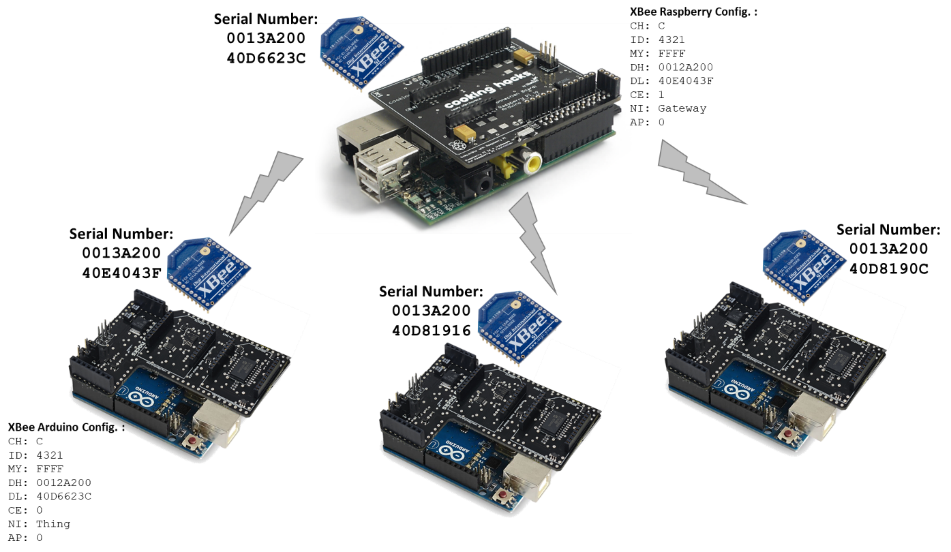
XBee 802.15.4 Tutorial

# Overview

In this tutorial, we will interconnect Arduino boards with a Raspberry Pi one, throw XBee 802.15.4 modules. Here is an illustration of the infrastructure we will implement.



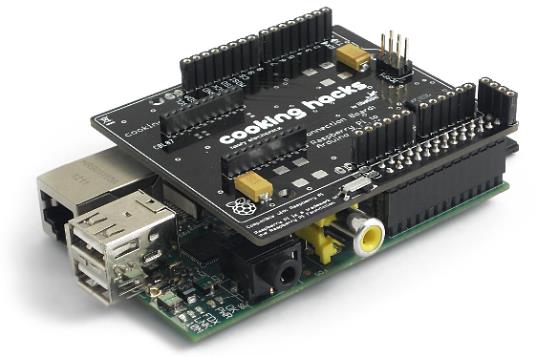
In the first step of this tutorial, we will configure a Raspberry Pi as a gateway (Coordinator) to receive XBee 802.15.4 message from multiple things (Endpoint). To simplify this integration example, we will use Node-Red.

# Raspberry Pi as a gateway

## Hardware

To configure “[Raspberry Pi to Arduino Shields Connection Bridge](https://www.cooking-hacks.com/raspberry-pi-to-arduino-shield-connection-bridge)” you should read the “Raspberry Configuration Tutorial” which is common to all communication modules.

In this specific tutorial, we will use the “[XBee Serie 1 modules](https://www.gotronic.fr/art-module-xbee-serie-1-xb24-api-001-23216.htm)” (which are a little bit easier to configure event if they don’t provide as functionalities as Serie 2). You can find a [tutorial made by Cooking Hacks](https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge/) who sells the Shield bridge module. Normally, you won’t need to apply all the Cooking Hacks tutorial, as we won’t write our own programs in C++, but use the Node-Red configuration.

## Software configuration of the Shield

Read and apply all the necessary commands descripted in the “Raspberry Configuration Tutorial: Software configuration of the Shield”.

Then you are ready to communicate over the serial port with the XBee module and test it.

## Test communication with XBee 802.15.4 module

To test if the XBee module is functional, you can use the minicom program to dialog with the XBee serial interface.

sudo apt-get install minicom

You can create a minicom profile with the right parameters to dialog the XBee module

sudo nano /etc/minicom/minirc.xbee

Add the following lines to this file:

pu port /dev/ttyAMA0

pu baudrate 9600

pu bits 8

pu parity N

pu stopbits 1

pu rtscts No

pu localecho Yes

To communication with the XBee module, type the following command:

minicom -o xbee

You can now use the AT commands to dialog and configure the XBee 802.15.4 module. Use +++ to enter this mode (with return line at the end).

Then you can configure the Channel, ID, Destination Address, … Use the following commands and type then in the minicom interface. After each command, you must receive a OK response, if all is correct.

+++

ATCH C // Channel C is the default one

ATID 4321 // Set an identifier, must be the same to all modules

ATMY FFFF // Don’t use the 16bit address so FFFF

ATDH 0013A200 // Define the Destination High address (set to Serail High)

ATDL 40E4043F // Define the Destination Low address (set to Serial Low)

ATCE 1 // Set to Coordinator

ATNI Gateway // Set the device name identifier (not necessary)

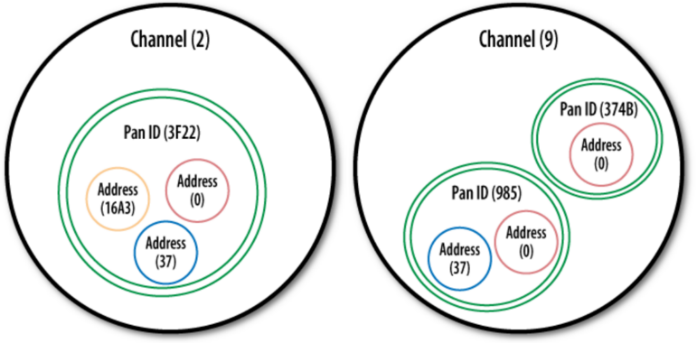
ATAP 0 // Set to API mode 0

ATWR // Write this configuration to be permanent one

ATCN // End communication with module

My choice was to use the Serial Number (High and Low) as addresses for the modules. The Destination address for the Raspberry Pi is not necessary as it will receive messages from Arduino modules. But for test purposes, I configured the one of the module used on the Arduino. In all case, Channel and ID must be the same for the modules to communicate. And I decided to use the 64bit address instead of the 16bit one (so MY is set to FFFF). As the Raspberry is a Coordinator, CE is set to 1 and we give a Name Identifier (but not necessary for the communication to work).

To clearly understand the XBee 802.15.4 parameters Channel, Pan ID and Address, here is a figure illustrating the principle.

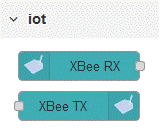


Note that it’s possible, if there is too much time between two instructions, the module automatically leave the command mode. So, to get back in the command mode to send AT commands, you must type +++ another time (still without any return). You read the [datasheet of these XBee Serie 1 modules](http://www.cooking-hacks.com/skin/frontend/default/cooking/pdf/90000982_A.pdf) if you want more information about the available AT command and much more.

## Node-RED installation and configuration

For the Node-Red installation on Raspberry Pi, you must read and apply commands descripted in the “Raspberry Configuration Tutorial / Node-RED installation”.

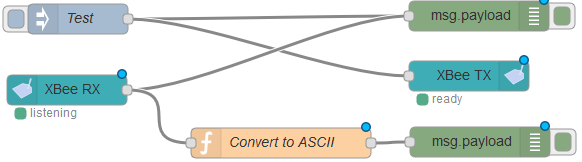
node-red-contrib-xbee version 1.0.0, which is the actual available release in Node-RED packages, depends on seriaport v4 which is not compiling with Node-RED 10.3. I ported this package to use serialport v7.

So, you need to install, at the minimum, release 1.1.0 node-red-contrib-xbee.

In Node-RED, you must install the XBee module to have the corresponding nodes. Go to the main menu (On the upper right of your Node-RED page), then choose the “Manage palette” entry. Go to the “Install” tab and type the module name: [node-red-contrib-xbee](https://www.npmjs.com/package/node-red-contrib-xbee) ([xbee-api documentation](https://github.com/jankolkmeier/xbee-api)). This will add 2 nodes types: the XBee RX and XBee TX to respectively “Receive” and “Transmit” data throw the XBee modules (and the underlying serial socket).

You must then configure the XBee serial connexion. You must select the following parameters: API Mode: API2 (escaped characters), Convert ADC, Serial Port: /dev/ttyAMA0, Baud Rate: 9600, Data Bits: 8, Stop Bits: 1, Parity: None. Create this serial configuration. Then you can interconnect these nodes to create your communication.

Here is a basic example and the corresponding flow. Don’t forget to modify the Test node with the right destination address.



Here is the code of this flow. Don’t forget to modify the address to send message to.

[{"id":"13da4faf.a2ee2","type":"tab","label":"Flow 1","disabled":false,"info":""},{"id":"898265ae.111a8","type":"debug","z":"13da4faf.a2ee2","name":"","active":false,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":532,"y":40,"wires":[]},{"id":"6fb4bc6e.001e0c","type":"xbee-tx","z":"13da4faf.a2ee2","name":"XBee TX","xBee":"a70592e5.e1efb8","x":447,"y":94,"wires":[]},{"id":"50320ad0.640964","type":"inject","z":"13da4faf.a2ee2","name":"Test","topic":"","payload":"{\"type\":0,\"id\":0,\"destination64\":\"0013A20040E4043F\",\"destination16\":\"FFFF\",\"broadcastRadius\":0,\"options\":0,\"data\":\"Hello from Node-RED! It's Raspberry Pi...\"}","payloadType":"json","repeat":"","crontab":"","once":false,"onceDelay":0.1,"x":113,"y":42,"wires":[["898265ae.111a8","6fb4bc6e.001e0c"]]},{"id":"279e13a1.9ec374","type":"xbee-rx","z":"13da4faf.a2ee2","name":"XBee RX","xBee":"a70592e5.e1efb8","x":99,"y":155,"wires":[["898265ae.111a8","5d10842d.d0bd34"]]},{"id":"5d10842d.d0bd34","type":"function","z":"13da4faf.a2ee2","name":"Convert to ASCII","func":"msg.payload = msg.payload.data.toString('ascii');\nreturn msg;","outputs":1,"noerr":0,"x":334,"y":155,"wires":[["24ee9cfa.8c4a0c"]]},{"id":"24ee9cfa.8c4a0c","type":"debug","z":"13da4faf.a2ee2","name":"","active":true,"tosidebar":true,"console":false,"tostatus":false,"complete":"false","x":538,"y":155,"wires":[]},{"id":"a70592e5.e1efb8","type":"xbee-config","z":"","apiMode":"2","rawFrames":false,"convertAdc":true,"vrefAdc":"1200","serialPort":"/dev/ttyAMA0","lock":true,"baudRate":"9600","dataBits":"8","stopBits":"1","parity":"none","bufferSize":"65536","rtscts":false,"xon":false,"xoff":false,"xany":false}]

# Configure and Test with a XBee 802.15.4 module connected to your PC

## Hardware

Before configuring the Arduino board with its own XBee socket, you can perform tests with a XBee 802.15.4 module connected to your PC with a XBee USB Board. It could be the following platforms.



## Software

Install on your PC the [XCTU program](https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu). A [tutorial](https://learn.sparkfun.com/tutorials/exploring-xbees-and-xctu/all) is available to use XCTU. You can set the default parameter, generate frame with your messages to examine the used protocol over the serial communication and of course send and receive message to and from other modules.

Here is information about frame you could have with such a tool:

Start delimiter: 7E

Length: 00 2F (47)

Frame type: 80 (RX (Receive) Packet 64-bit Address)

64-bit source address: 00 13 A2 00 40 D6 62 3C

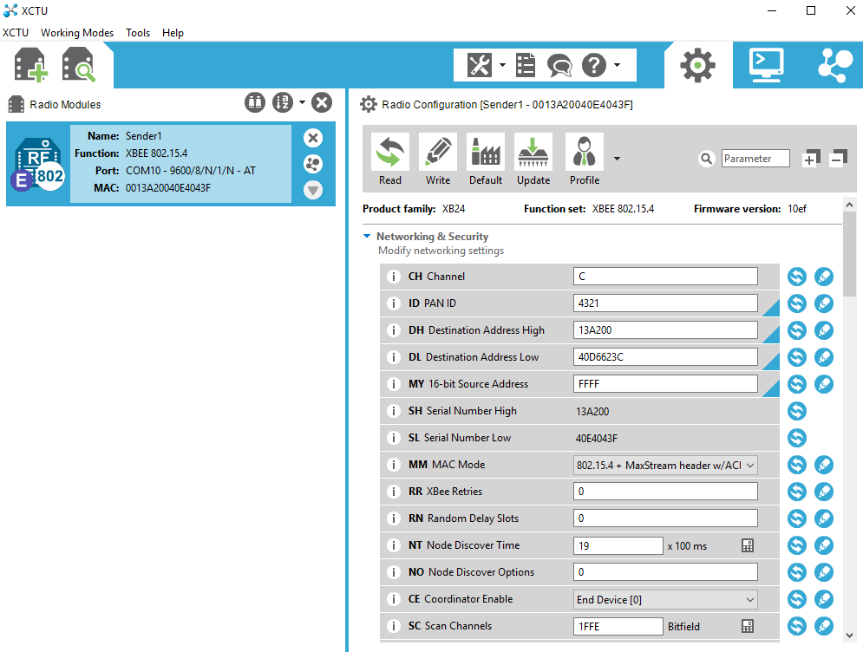
RSSI: 40

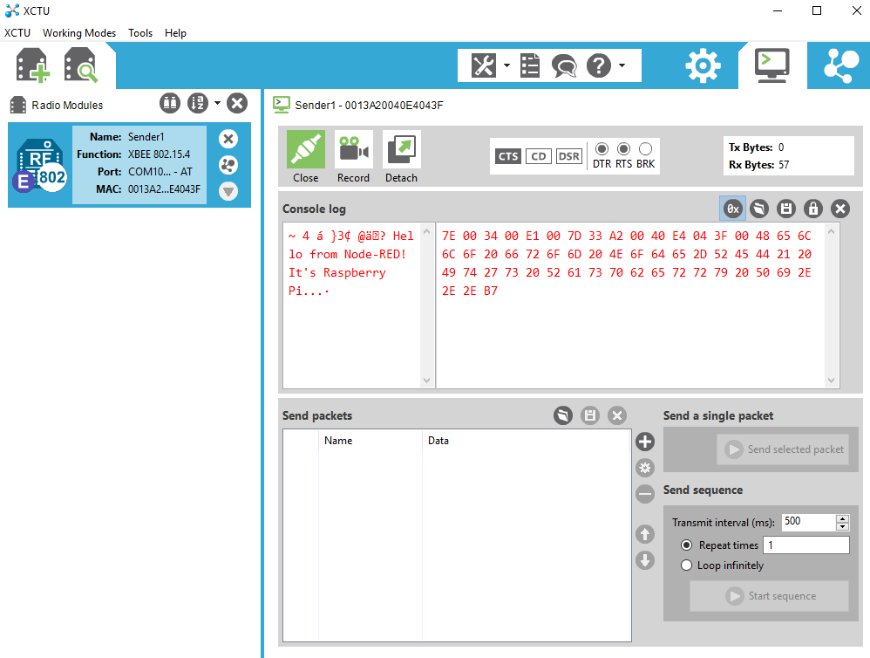
Options: 00

RF data: 48 65 6C 6C 6F 20 66 72 6F 6D 20 4E 6F 64 65 2D 52 45 44 21 20 49 74 27 73 20 41 72 64 75 69 6E 6F 2E 2E 2E

Checksum: 4C

Configure in the same way as we did for the Raspberry one. Set the same Cannel and Pan ID and set the destination Low and High address to the one of the XBee 802.15.4 module connected to your Raspberry Pi. As suggested, you should specify here the serial number of the XBee 802.15.4 module connected to the Raspberry Pi.





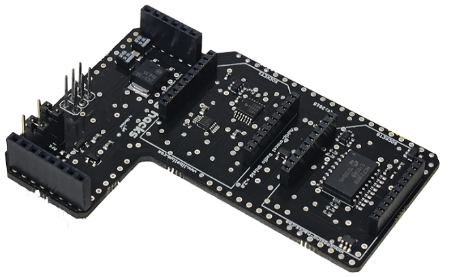
# Arduino as a Thing

## Hardware

Now, we will configure an Arduino to communicate throw an XBee 802.15.4 module. We use the following hardware: “[Multiprotocol Radio Shield for Arduino](https://www.cooking-hacks.com/multiprotocol-radio-shield-board-for-arduino-rpi-intel-galileo)” and of course a XBee Serie 1 module (Serie 1 and Serie 2 are not compatible together). You can find a [tutorial made by Cooking Hacks](https://www.cooking-hacks.com/documentation/tutorials/multiprotocol-shield-connect-two-xbee-connector-arduino-raspberry-pi-galileo/) who sells the multiprotocol shield for Arduino.

Connect the XBee 802.15.4 module to **Socket 0** of the multiprotocol radio shield for Arduino.





## Software configurations

First, you need to download the Arduino IDE for your operating system. We won’t detail this in this tutorial. Then you must [download the library to manage the multiprotocol shield](http://www.cooking-hacks.com/media/cooking/images/documentation/multiprotocol_radio_shield/multiprotocol_shield_library.zip). Copy the two folders MCP23008 and Multiprotocol\_Shield in your Arduino libraries folder (on Windows, it’s in Documents\Arduino\libraries).

Then, within the Arduino IDE, select “Sketch / Library / Manage Libraries”. In the filter section, type xbee and install the [XBee-Arduino](https://github.com/andrewrapp/xbee-arduino) library. This will allow to add an implementation of the XBee API protocol for Arduino.

Your Arduino IDE is no ready to compile a program and send it to the Arduino board.

## Program

Here is a simple example to communicate data to your Raspberry Pi.

#include <multiprotocolShield.h>

#include <MCP23008.h>

#include <Wire.h>

#include <XBee.h>

XBee xbee = XBee();

// Create an array for holding the data you want to send.

uint8\_t payload[] = { 'H', 'i', '!', ' ', 'I', '\'', 'm', ' ', 'A', 'r', 'd', 'u', 'i', 'n', 'o', '.' };

// Specify the address of the remote XBee (this is the SH + SL)

XBeeAddress64 addr64 = XBeeAddress64(0x0013a200, 0x40d6623c);

void setup() {

socket0.ON();

delay(100);

socket0.setMUX();

Serial.begin(9600);

delay(1000);

xbee.setSerial(Serial);

}

void loop() {

ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload));

xbee.send(zbTx);

delay(2000);

}

It will send the message “Hi! I’m Arduino.” to the XBee 802.15.4 module with address 0013a200 40d6623c which is the one plugged to the Raspberry Pi.

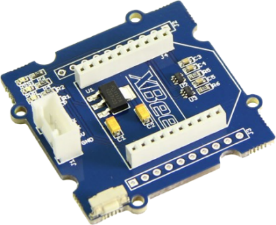
In your Node-RED debug console, you should read this message sent every 2s.

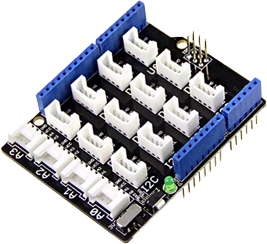
You can also use the XCTU program to send another message from another XBee 802.15.4 module connected to your PC. You should see the sent message interleaved with the one sent by your Arduino.

# Another Arduino as a Thing

## Hardware

As a second solution (and probably a cheaper one), you can you a [Grove Base Shield for Arduino](https://www.seeedstudio.com/Base-Shield-V2-p-1378.html) and a [Grove Bee Socket](https://www.seeedstudio.com/Grove-Bee-Socket-p-1449.html) to plug your XBee 802.15.4 module.





## Software configurations

The same configuration as the previous one for Arduino is needed: you must install the XBee-Arduino library.

## Program

The program to test your communication if quite the same as the previous one for Arduino. You just must suppress the code needed for the multiprotocol shield used previously. So, you must suppress the two first includes in your code (#include <multiprotocolShield.h> and #include <MCP23008.h>) and also remove the 3 first lines in the setup function (socket0.ON(); delay(100); socket0.setMUX();). The rest of the program don’t change.

# Conclusion

With this tutorial, you learned how to make XBee 802.15.4 communications from Arduino (with your own program) to a Raspberry Pi (using node-RED). We hope you enjoyed!