

Labor zur Vorlesung Kommunikationstechnik

IEEE 802.15.4 mit XBee3 Devices

Team-Member 1:

Team-Member 2:

1. Getting Started with XCTU

The purpose of this section is to get started with the XBee3 devices in particular with the software XCTU (<https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu>) from Digi for configuring the devices and first (manual) communication.

1.1 Configuration with XCTU

Tipp: Das Öffnen einer XCTU Instanz pro Device kann das Konfigurieren einfacher machen.



1. Install newest 802.15.4 Firmware



2. Reset all parameters to default parameters and write parameters
3. Set the following parameters to the same values for all devices:
 - CH (Channel): choose one
 - ID (Network PAN ID): choose one
 - MM (MAC Mode): Strict 802.15.4 with ACKs
 - AP (API Enable): Transparent Mode
 - BD (UART Baud Rate): 115200 (important for PyCharm PlugIn)
4. Individual configuration:
 - configure one device as PAN coordinator (CE=Device Role) and set an identifier for the device (NI=Node Identifier)
 - leave the other devices as End Devices and name them (NI=Node Identifier)
 - choose a 16-bit MAC address for the end devices (MY=16-bit Source Address), leave the 16-bit MAC address of the coordinator as 0

1.2 Communication via XCTU Console in transparent mode

The transparent mode is similar to a serial interface i.e. it is used to transmit a byte stream. Consequently, the destination address is not defined per frame but configured in the DH and DL parameters. We use 16-Bit short addresses so setting the 16 lower bits of the DL parameter is sufficient. In order to write or read in the transparent mode you need switch to the terminal mode and close the serial connection with the XBee device. The XCTU terminal is connected via a serial interface over USB to the XBee device and the XBee device in transparent mode transmit all bytes received on the serial interface to the destination address. When you write something into the console you should see it on the consoles of the receiving devices. Note that detaching the terminal window may be useful when handling multiple devices in one XCTU instance. Alternatively, you may open one XCTU instance per device.

Even if the data is transferred as a byte stream, the transmission over the 802.15.4 radio interface is done frame by frame with the bits taken from a buffer the bytes from the serial interface are written to.

1. Broadcasting in transparent mode

A broadcast frame is received by all nodes in the PAN if in radio range. Note that per se every packet is received physically by every device in radio range but a packet is only forwarded to the

higher layer if the MAC destination address of the received frame is either the address of the device or the broadcast address 0x000000000000FFFF.

Configure the broadcast address in one of the devices and verify that the characters you type into the console of the device are displayed in the consoles of the other two devices.

2. Sending unicast packets in transparent mode

A unicast frame is sent to exactly one device indicated by the unicast destination address.

Set the destination addresses of the remaining two devices to the source addresses of the broadcasting device and check that the messages are received by the broadcasting device only.

1.3 Configure your device in transparent mode

The configuration in transparent mode is done by sending AT commands over the serial interface.

To distinguish AT commands from the normal byte stream the sequence “+++” (without return) followed by a pause is used to switch from data mode to command mode. Returning from command mode to data mode happens automatically after a 10s interval of inactivity or after sending the command ATCN. You can configure the duration of this inactivity period by parameter CT (AT Command Mode Timeout).

In command mode the receiver interprets incoming data as AT commands. AT commands are in the format AT<Command>[<Value>]<CR>. A value is used only when the command requires an argument. A list of AT commands for the XBee devices can be found in Digi XBee® 3 Zigbee® RF Module. For accessing parameters the command is AT<2-Char-Parameter-Name>[<Value>] and here a value is given for setting a parameter while the command without a value reads the parameter.

1. Read the channel via AT commands.
2. Set the destination address via AT commands. Validate the change by sending data. Return to the configuration mode and reload (read) the destination address. Verify that it was actually changed.

1.4 Working with Devices in API mode


In the API mode the XBee device communicates via USB in terms of frames of different types. Telling the XBee device to transmit a frame over the radio interface is done by Tx (Transmit) Request frames using 64 bit addresses (type 0x00) or 16 bit addresses (type 0x01). The configuration of the device in API mode also works via AT commands but now sent in AT Command frames (type 0x08).

The XBee device transfers received 802.15.4 radio frames via the Rx Packet frame using either 64-bit addresses (0x80) or 64-bit addresses (0x81) and transfers responds to AT commands in AT command response frames (0x88). You can find an overview of all frame types in the Digi XBee3 802.15.4 RF Module User Guide p. 190ff.

1.4.1 Configure API mode

In the API configuration of your devices set API Enable to API Mode With Escapes.

1.4.2 Communication via XCTU Console in API mode

XCTU provides a frame generator tool for sending frames in API mode. When in terminal mode click on the  icon to generate frames. You can either send a single frame or the sequence of specified frames repeatedly. Transmitted and received messages are shown in the Frames Log. You can also try a mix of devices in transparent mode and API mode.

1. Use a 16-Bit Tx Request frame to broadcast a message
2. Use 16-Bit Tx Request frames to send a unicast message to both other devices.

1.4.3 Check and configure your device in API mode

Generate and execute AT command frames to

1. Perform an energy detect scan by sending AT command ED.
2. Change the name of the device by sending AT command NI.

1.5 A First Range Test

XCTU provides a Range Test that shows the RSSI of the received packets. The principle of the Range Test is that the local XBee module connected to XCTU transmits packets to a remote XBee module that loops back the packets.

1. Set the MAC Mode (MM) to “802.15.4 + Digi Header w/ACK[0]”. The range test does not work with regular 802.15.4 header. Check that the devices use the same channel and have the same PAN ID. If the range test does not work, reset the parameter to default.
2. Configure the local device to API mode and the remote devices to AT mode.
3. Start the Range Test (under Tools)
 - select local device
 - detect and select remote device
 - set range test type to “Loopback”
 - set loopback jumper to the right (the loopback jumper is to the right of the USB connector)
 - configure the range test
 - start the range test and check how the RSSI changes when
 - moving the remote device
 - putting an obstacle e.g. a hand between local and remote devices
 - changing the orientation of the antennas (parallel vs. peak-to-peak)
 - don't forget:
 - reset the loopback jumper
 - check that all devices have the same MAC mode

2 MicroPython

The XBee3 devices are able to run Micropython code. XCTU provides a simple Micropython Terminal where you can make first steps and try sending frames via Micropython commands. Actual programming is best done with the PyCharm XBee PlugIn.

2.1 *XCTU Terminal*

Use the XCTU MicroPython-Terminal to switch the on-board user LED (Pin D4) on and off. Note that the value "0" means ON. Take care to set the API Enable (AP) parameter to "MicroPython REPL" before activating the MicroPython Terminal.

2.1.1 Communication

Stay in the MicroPython Terminal and communicate (receive, transmit) with another device in API mode. If that works, try to communicate with both devices in MicroPython mode. Use the functions `xbee.transmit()` and `xbee.receive()`.

2.1.2 Configuration

You can send AT commands in MicroPython using the function `xbee.atcmd()`. Change the channel of all devices and communicate again.

2.2 *PyCharm PlugIn*

The "Xbee MicroPython" PlugIn allows you to compile, upload, and run code on the xbee device.

Take care that you set the baud rate to 115200. We often experienced problems with lower baud rates when flashing MicroPython scripts to the XBee3 devices.

Take care that by default the code runs again after rebooting the device. If you want to keep the device silent best upload an empty Python script. Alternatively, you may deactivate running the file automatically at start-up. You can do that under "Edit configurations ..." at the project selector box.

If the code does not upload with the message that the current program cannot be stopped also after rebooting, go to XCTU and deactivate the MicroPython AutoStart (ATPS).

2.2.1 User LED

The device should continuously receive frames, read the first byte as `n` from the payload and let the used LED (Pin D4) blink `n` times. You may decide on your own how to send the packets. Best check again all alternatives.

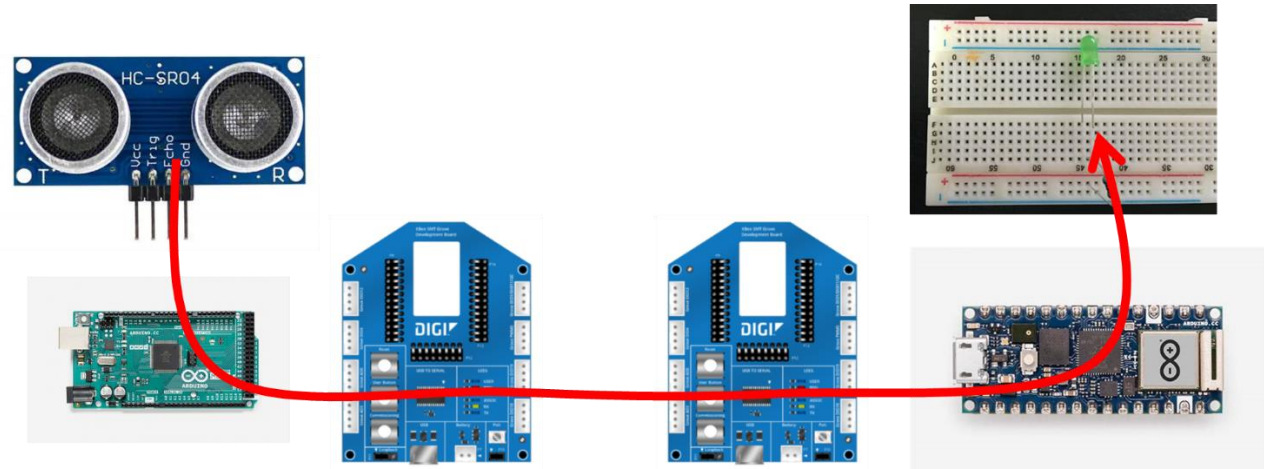
2.2.2 Range Test

Write your own multi-device range test. The local device should send a unicast message to both remote devices. After receiving the message the remote devices check the RSSI and respond with a message containing the RSSI as payload. After reception of the response the local device checks the RSSI and reads the Remote-RSSI from the message. The local device should wait for all responses and then transmit the next unicast message. If not all responses are received the

local device after a timeout, the next round starts. The range test should run k times and present as a result per remote device the percentage of successful responses, the minimum, the mean, and the maximum of the RSSI and the remote RSSI.

3 Control LED via Ultrasonic Sensor

The purpose of this lab is that you implement a simple project including communication via IEEE 802.15.4 and I2C. The idea is that you have two XBee devices and two Arduinos (a Nano and a Mega). An ultrasonic sensor is attached to the Mega that steers the brightness of an LED attached to the Nano. The Mega regularly measures the distance with the ultra-sonic sensor and transfers the distance to the XBee3 device A via I2C. The XBee3 device A sends the distance to XBee3 device B, and XBee3 device B sends the distance to the Nano again via I2C. The Nano then sets the brightness according to the distance.



3.1 Hints:

1. Usage of the ultrasonic sensor: <https://projecthub.arduino.cc/Isaac100/getting-started-with-the-hc-sr04-ultrasonic-sensor-7cabe1>
2. Brightness of the LED on the Arduino: Examples/Basics/Fade
3. Implementation of I2C on the Arduinos: Wire Library
4. Implementation of I2C on the XBee3 modules: machine.I2C
5. Upload auf den Nano: wenn der Nano beim Hochladen eines Skripts aus der Arduino IDE hängenbleibt: zweimal Reset-Knopf drücken
6. I2C Pins

