



**Übung zur Vorlesung**

**Vernetzung autonomer Systeme**

**(VASY)**

**Übung 1**

Die Abgabe erfolgt durch Durchführung bzw. Vorführung des Experiments in der Laborübung.

**Bearbeitung in Dreier-Teams**

**Team-Mitglied 1:**

**Team-Mitglied 2:**

**Team-Mitglied 3:**

# Experiments

## Getting Started

1. Get 2 Arduino Megas, 2 XBee Shields, 2 XBee 802.15.4 radio modules, 2 USB cables, 1 external power supply, 4 jumpers
2. Check that the following software is installed on your lab PC / private laptop:
   * Arduino: <https://www.arduino.cc/en/Main/Software>
   * XBee library for Arduino: <https://code.google.com/p/xbee-arduino/>
   * XCTU: <http://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu>
3. Add the XBee Shield and the XBee Radio module to your Arduino Mega and connect them via USB to your Lab-PC/Desktop. They can be connected to a single machine or two different machines.
4. Start the Arduino and execute the Blink and Fade examples. Be sure to use Pin 13 to control the onboard LED of the Mega. Afterwards load the BareMinimum sketch. You can find the sketches under Datei/Beispiele. Here you should also find example sketches for XBee if you have installed the XBee library properly.

## Communication via XCTU Console

1. Start the XCTU-Software and detect the two XBees. Take care that

* jumpers are set to USB
* no jumpers are set to connect DCIN and DCOUT
* if a device is not detected on COM Port, try to disconnect one or all devices from USB and reconnect again.

1. Check the configuration of your XBee module. What are the channel, PAN ID, destination address and source address?
2. Use the XCTU console to exchange messages between the two XBee modules. Anything that you type into the console of an XBee module should be received by all other modules including those from other teams. Check that.  
   Take care that
   * all XBee modules are configured with equal settings for channel, source and destination addresses and PAN Ids.
   * that the API mode is disabled in both modules (see XBee Guideline)
3. Switch the API mode to API disabled. Now you can generate and observe the frames sent over the radio interface.

## First Configuration of your Module

1. Configure your module in such a way that your messages are only received by the other device of your team. Try to achieve this effect by modifying PAN Id, Channel, and source/destination address. Also use the destination address 0xFFFF. What is the effect?

* There are two ways to configure your XBee module: via the XCTU configuration interface or directly via AT commands.

1. Use the XCTU interface to configure your XBee module. You can observe the respective AT commands on your console while API mode is disabled. What AT commands are used to change the configuration? You can also find out the meaning via the list of AT commands (see XBee Guideline).
2. Configure your device also by typing AT commands into your console.
3. Coordinate with the other teams such that every team has its own channel and PAN-Id. Configure these settings.

## 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. Detect the remote devices in XCTU by clicking the discovery icon next to one of the XBee modules. If the remote devices are not found, first enter the network discovery mode where the remote device is found and then detect the remote devices again. In order to do the network discovery, however, you need to set the discovering module to API mode.
2. After the remote module is detected, set it to transparent mode and connect the DIN and DOUT pins with a jumper. The local module needs to stay in API mode.
3. Perform the range test. Carry the remote device around, observe and note down the effects. If you have a laptop, connect your local device to your network such that you can perform the range tests in different environments. Also check different heights. Is there an impact? Can you confirm some of the propagation effects/models described in the lecture?
4. Now all teams should set their modules to the same channel and PAN-Ids. Can you detect all devices? Choose again your second device for performing the range test. Can you observe any difference?
5. Remove the DIN/DOUT jumper

## Serial Communication via Arduino

So far the Arduinos have only been pipes to connect the XCTU software on the PC with the XBee modules. Next you are going to program the Arduinos to transmit and receive data. In the first step the communication runs via the Serial interface so the Arduinos need to be set to the transparent mode (disable API).

In order to upload your program (sketch) to the Arduino you need to remove the two jumpers for the three pins on the XBee shield or set them to USB. In order to use the XBee communication you need to set the two jumpers to XBee.

1. Develop two scripts where one Arduino device steers the LED of the other Arduino device. The communication works via the Serial interface. Use on of the examples Communication/Dimmer or Communication/PhysicalPixel as starting point for the receiving side. Write a script for the transmitting side.
2. In order to work more comfortable with the Arduinos remove the jumpers and connect the DIN pin on the XBee Shield with TX1 on the Mega and DOUT with RX1, respectively. Now, the Mega communicates with the PC via USB over the Serial interface (upload sketches, serial monitor) and the Mega communicates via the Serial1 interface with the XBee Shield.
3. Check that you can now upload sketches without resetting jumpers.
4. Check your previous LED script. You will notice that it does not work anymore. Modify the script to use the two different serial interfaces. Additionally copy all inputs and outputs via the Serial1 interface to the Serial interface such that the communication is also shown in the serial monitor provided by Arduino. You may also use e.g. Putty to read the communication.

## API Communication via Arduino

For the API Communication with the Arduino use the XBee library and the Series1\_TX and Series1\_RX examples as a basis.

1. Develop a script that makes the LEDs of the two Arduinos fade and blink synchronously. The transmitting Arduino should generate a random fade pattern consisting of two alternating phases. Both phases are characterized by brightness duration. The Master-Arduino should transmit the setting to the Slave-Arduino and they should initiate the setting as synchronously as possible. What is the transmission delay for a message and how constant is it?
2. Make the LEDs of all Megas in the room blink synchronous.
3. Check the reliability (number of successful packets) and capacity (number of packets per second) of the 802.15.4 connection for different settings

* separate team networks: every team uses a different channel and a different PAN ID
  + generate unidirectional and bidirectional traffic
* all teams use the same channel but different PAN IDs
  + generate unidirectional and bidirectional traffic
* all teams use the same channel and the same PAN IDs but different source and destination addresses
  + generate unidirectional and bidirectional traffic between your two nodes
  + let one node per team generate broadcast messages

## Range Test

Perform a bi-directional range test with two movable Arduinos, one connected to the laptop and one connected to a battery. The range test should be implemented both with Arduinos in API mode and in transparent mode for configuring the XBee modules and retrieving RSSI information. The bidirectional range test should yield

* packet error rate
* RSSI
* round trip time (time for a packet to travel to the remote device and back)

Store the values on your remote device (and maybe also the laptop/PC device) to read it to your PC later. Use the EEPROM example as a starting point for storing the values. Design duration and sampling rate of your measurement campaign in a way such that you can store all measurements in EEPROM.