XBee

XBees[1] sind Funkmodule, die wie z.B. Bluetooth im 2.4 GHz Frequenzbereich arbeiten. Sie implementieren den IEEE 802.15.4 Standard[2], der dafür ausgelegt ist, um kleine Datenmengen möglichst stromsparend zu übertragen (z.B. Fernbedienungen, Steuerungen). Der Datentransfer ist dabei zuverlässig (reliable data transfer).
Mit XBee und XBee Pro kann nur Point-to-Point und Point-to-Multipoint Kommunikation implementiert werden. Das heisst Daten können nur an einen direkten Nachbarn versendet werden. Mit XBee ZNet (Series 2)[3] beispielsweise können auch Multihop-Netzwerke installiert werden.

Abbildung 1: XBee Funkmodul: Chipantenne (hellblau), 20 Pins für Stromversorgung und Kommunikation (Unterseite)

Einsatzzweck

XBees können eingesetzt werden, um eine direkte Kabelverbindung für serielle Kommunikation durch eine drahtlose Lösung zu ersetzen.
Ebenfalls kann ein XBee eingesetzt werden, um Daten von einem Sensor drahtlos an eine zentrale Stelle zu schicken. Ein solcher Sensor kann die Daten als seriellen Datenstrom (UART), oder als analoges oder digitales Signal bereitstellen.
Tabelle 1 zeigt den Einsatz von 802.15.4 (von XBee implementiertes Protokoll) im Vergleich zu Bluetooth:

---

[1] http://www.digi.com/products/wireless/point-multipoint/xbee-series1-module.jsp

[2] http://en.wikipedia.org/wiki/IEEE_802.15.4

[3] http://www.digi.com/products/wireless/zigbee-mesh/xbee-zb-module.jsp

Tabelle 1: Vergleich Einsatzzweck 802.15.4 zu Bluetooth-Protokoll

802.15.4	Bluetooth
kleinste Datenmengen in grossen Netzen	grössere Datenmenge in kleinen Netzen
vorwiegend statische Netzwerke	Ad-hoc Netzwerke
Steuerung, Automation, Sensoren	Dateiübertragung, Audio

Funktionen

Ein XBee hat einen UART-kompatiblen seriellen Eingang (Pin 3, Rx). Daten, die an diese Stelle gesendet werden, werden paketisiert und an den Empfänger geschickt, für den der XBee programmiert ist. Ein XBee kann seine Daten auch an alle erreichbaren XBees senden (Broadcast). Die empfangenen Daten werden am Pin 2 (Tx) wieder bereitgestellt.
Ein XBee hat zudem mehrere analoge und digitale Eingänge. An diese können direkt Sensoren angeschlossen werden, die ein entsprechendes Signal bereitstellen. Die Daten werden digital übertragen und an einem der digitalen Ausgänge bereitgestellt. Analoge Ausgänge gibt es keine. Mehrere Pins können sowohl als analoge Eingänge sowie als digitale Ein-/Ausgänge benutzt werden. Die Belegung der Pins ist im XBee Product Manual beschrieben[4].
Pin 6 liefert ein RSSI-Signal (RSSI: Rx Signal Strength Indicator). Dieses Signal zeigt an, ob im Moment im entsprechenden Frequenzbereich Daten übertragen werden.
Um einen XBee zu konfigurieren (z.B. um die Destination Address zu setzen), muss das Modul in den Command-Mode versetzt werden. Dies geschieht über die serielle Schnittstelle (Pins 2 und 3), indem man die Sequenz „+++“ sendet. Danach interpretiert der XBee ankommende Zeichen als Konfigurationskommandos. In diesem Modus können nun mit AT-Commands alle Parameter des Moduls gesetzt werden. Dazu gehören neben den Destination- und MY-Adressen auch der Sleep-Modus oder die Belegung der Analog/Digital-Pins. Die AT-Commands sind im XBee Product Manual[3] (Kapitel 3.2, s.26ff) beschrieben. In Anhang B ist beispielhaft die Programmierung der Adressen beschrieben.
Eine einfachere Möglichkeit, einen XBee zu konfigurieren, bietet die X-CTU-Software von Digi[5].
Um Strom zu sparen und damit die Lebensdauer der Batterie zu erhöhen, kann ein XBee in den sleep-mode versetzt werden. Mittels AT-Commands kann aus drei verschiedenen sleep-modes gewählt werden (Hibernate, Doze, Cyclic Sleep). Der XBee kann dann mit einem Signal am Pin 9 (Sleep_RQ) in den Ruhezustand versetzt respektive wieder geweckt werden. Die sleep-modes sind im XBee Product Manual[3] beschrieben.
Ein XBee benötigt eine 3.3 Volt Stromquelle (mindestens 2.8, höchstens 3.4 Volt). Je nach sleep mode verbraucht das Modul 3 – 50 mikroampere.

Kommunikation

Der von XBee und XBee Pro implementierte IEEE 802.15.4 Standard beschreibt die Kommunikation zweier Module auf der Sicherungsebene des OSI-Modells. Dabei können Daten direkt an ein Nachbarmodul geschickt werden, dessen Hardware-Adresse bekannt ist. Es können jedoch keine Multihop-Netzwerke aufgebaut werden. Ein XBee kann Daten also nur an direkte Nachbarn schicken. Um Multihop-Netzwerke zu implementieren, können XBee

---

[4] http://www.makingthings.com/resources/datasheets/manual_xb_oem-rf-modules_802.15.4.pdf

[5] Die X-CTU-Software befindet sich auf der im XBee-Starter-Package von MaxStream mitgelieferten CD

ZigBee-Module verwendet werden. Diese implementieren das auf IEEE 802.15.4 aufbauende ZigBee-Protokoll[6].
Die Datenrate beträgt maximal 250 kbit/s. Die maximale Übertragungsdistanz zwischen zwei Modulen in Gebäuden liegt für XBees bei 30 Metern (XBee Pro ca. 100 Meter).
XBees haben neben der festen 64 bit langen Hardware-Adresse eine 16-bit lange MY-Adresse, die frei gesetzt werden kann. Beim Versenden von Daten kann entweder die feste 64-bit Adresse des Empfängers als Destination-Adresse beim Sender gesetzt werden, oder die frei gewählte 16-bit MY-Adresse. Mit diesen Einstellungen kann Point-to-Point Kommunikation realisiert werden. Die Adresse kann entweder mit der X-CTU-Software oder mit den AT-Commands im Command-Mode gesetzt werden.
Point-to-Multipoint Kommunikation (Broadcast) kann realisiert werden, indem beim Sender die Broadcast-Adresse als Destination gewählt wird (DL = 0x0000FFFF, DH = 0x00000000). Empfänger, die solche Datenpakete erhalten, senden keine ACKs (acknowledgement). Bei dieser Kommunikation ist also nicht bekannt, welche Empfänger die Daten erhalten haben.
Die Kommunikation über die serielle Schnittstelle mit einem XBee kann entweder im transparent mode oder im API mode stattfinden. Im transparent mode werden ankommende Daten vom XBee einfach weiterversendet (Ausnahme: mit „+++“ eingeleiteter Command-Mode). Im API mode ist es möglich, die ganzen Datenpakete inklusive Absender- und Empfängeradresse (und weitere Daten aus dem Header) auszulesen. Dazu eignet sich zum Beispiel die Processing Library für XBees[7].

X-CTU

Mit der X-CTU Software von MaxStream können XBees sehr einfach parametrisiert werden. Das Hauptfenster besteht aus 4 Tabs:
Tab „PC Settings“: Hier kann unter anderem der richtige Port ausgewählt werden. Ebenfalls muss auf die richtige Baud-Rate geachtet werden. Danach kann die Verbindung zum XBee getestet werden.
Tab „Range Test“: Hier kann ein Loopback-Test durchgeführt werden. Dazu kann zum Beispiel die in Kapitel „Beispielapplikation 1: Loopback“ gezeigte Schaltung verwendet werden.
Tab „Terminal“: Das Fenster zeigt Daten an, die an diesem Port (und somit vom XBee) empfangen werden. Es können ebenfalls Daten and den XBee gesendet werden („Assemble Packet“). Mit dieser Funktion können auch AT Commands getestet werden.
Tab „Modem Configuration“: Hier können alle Parameter des XBees ausgelesen und gesetzt werden. Diese Funktionalität ist ein einfacher Ersatz für alle AT-Commands (Abbildung 2).

---

[6] http://en.wikipedia.org/wiki/ZigBee

[7] www.faludi.com/code/xbee-api-library-for-processing

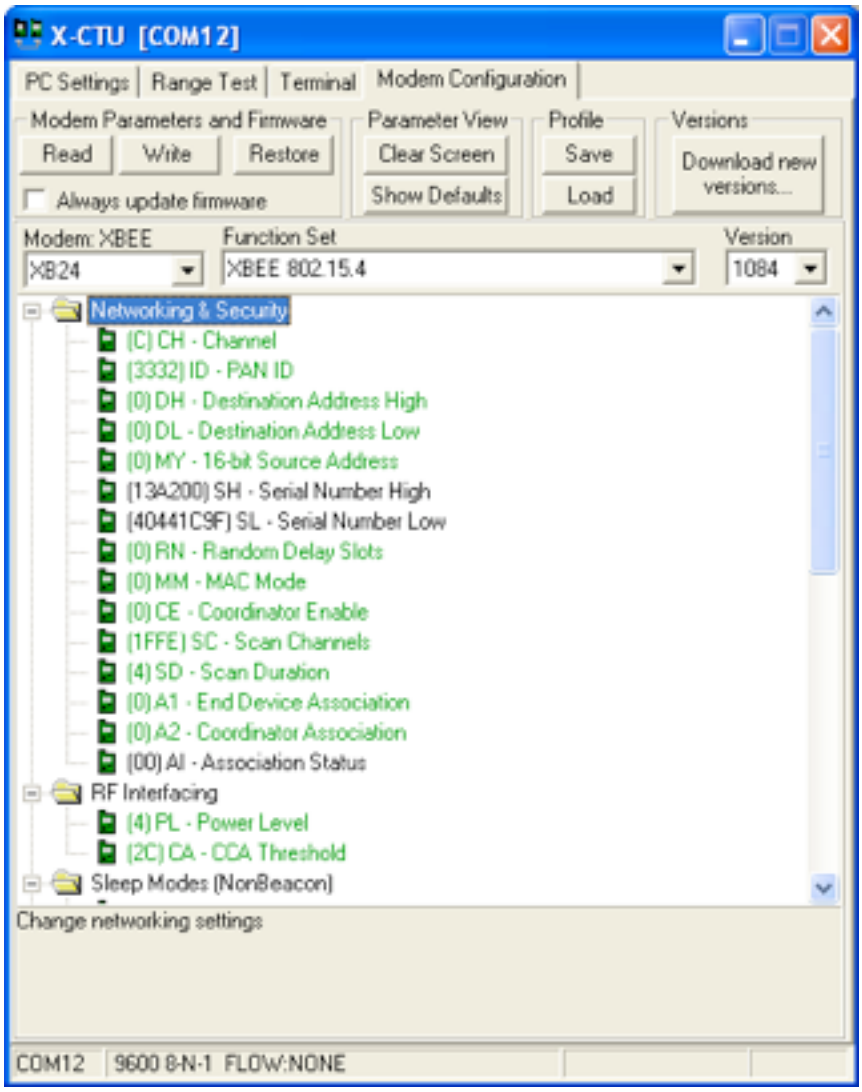

Abbildung 2: X-CTU Modem Configuration Tab

Verbindung zu anderen Modulen

Eine serielle Kabelverbindung zwischen zwei Modulen (z.B. zwischen einem Sensor und einem PC oder zwischen zwei PCs) kann durch zwei XBees ersetzt werden. Dazu müssen die entsprechenden Tx- und Rx-Pins des XBees mit denen des Sensors respektive des Rechners verbunden werden. Die XBees müssen zudem mit Strom versorgt werden.

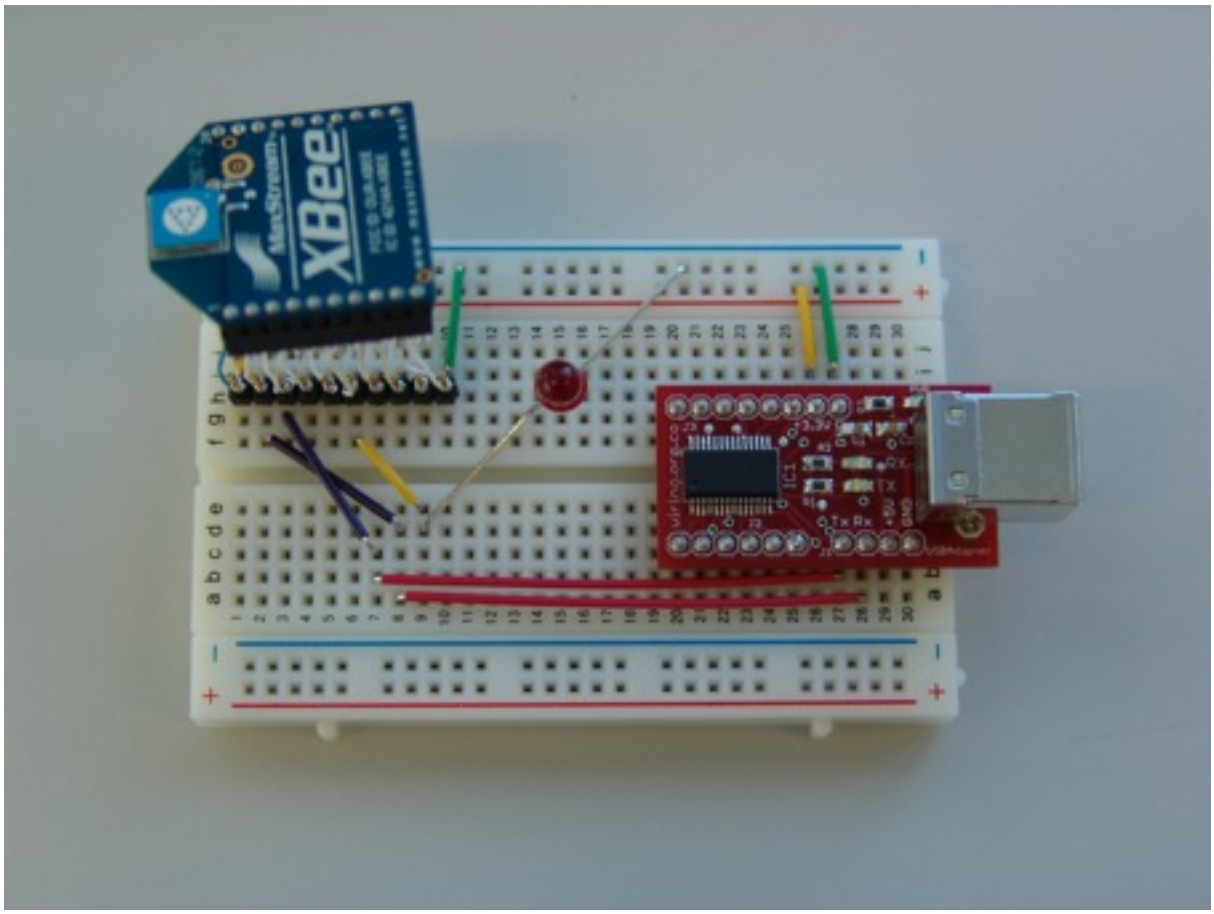

Abbildung 3: XBee verbunden dem USB-Modul eines mini-Wiring-Boards

Abbildung 3 zeigt, wie ein XBee an ein USB-Modul angeschlossen werden kann. Dank diesem Modul kann eine serielle Verbindung mit einem PC hergestellt werden, der über einen USB-Anschluss verfügt, aber nicht über eine serielle Schnittstelle. Die Stromversorgung geschieht über die 3,3-Volt-Pins des USB-Moduls (gelb und grün), welche mit Pin 1 (VCC) und Pin 10 (GND) des XBees verbunden sind. Die Rx- und Tx-Pins des XBees werden (gekreuzt) mit denjenigen des USB-Moduls verbunden (violett und rot). Die rote LED ist mit Pin 6 des XBees (RSSI) verbunden und leuchtet somit, sobald der XBee ein ausreichend starkes Signal erkennt.
Wenn ein USB Modul an den PC angeschlossen wird, erkennt der PC dieses als neue Hardware. Der „Found new Hardware“-Wizard (Windows) wird gestartet. Diesem muss nun angegeben werden, wo sich ein Treiber befindet, welcher das Modul als einen seriellen (COM-)Port erscheinen lässt. Ein solcher Treiber befindet sich z.B. auf der im XBee-Starter-Package mitgelieferten CD. Ausserdem kann er von der digi-Webseite heruntergeladen werden[8].

Beispielapplikation 1: Loopback

Eine einfache, wenn auch nur zum Testen der Module sinnvolle Anwendung besteht darin, ein Loopback-Modul einzurichten. Dies ist eine Schaltung mit einem XBee, die Daten, die von einem anderen XBee empfangen werden, wieder zurückschickt (dabei auf die richtige Adressierung der beiden XBees achten).

---

[8] http://www.digi.com/support/productdetl.jsp?pid=3249&osvid=0&s=315&tp=1

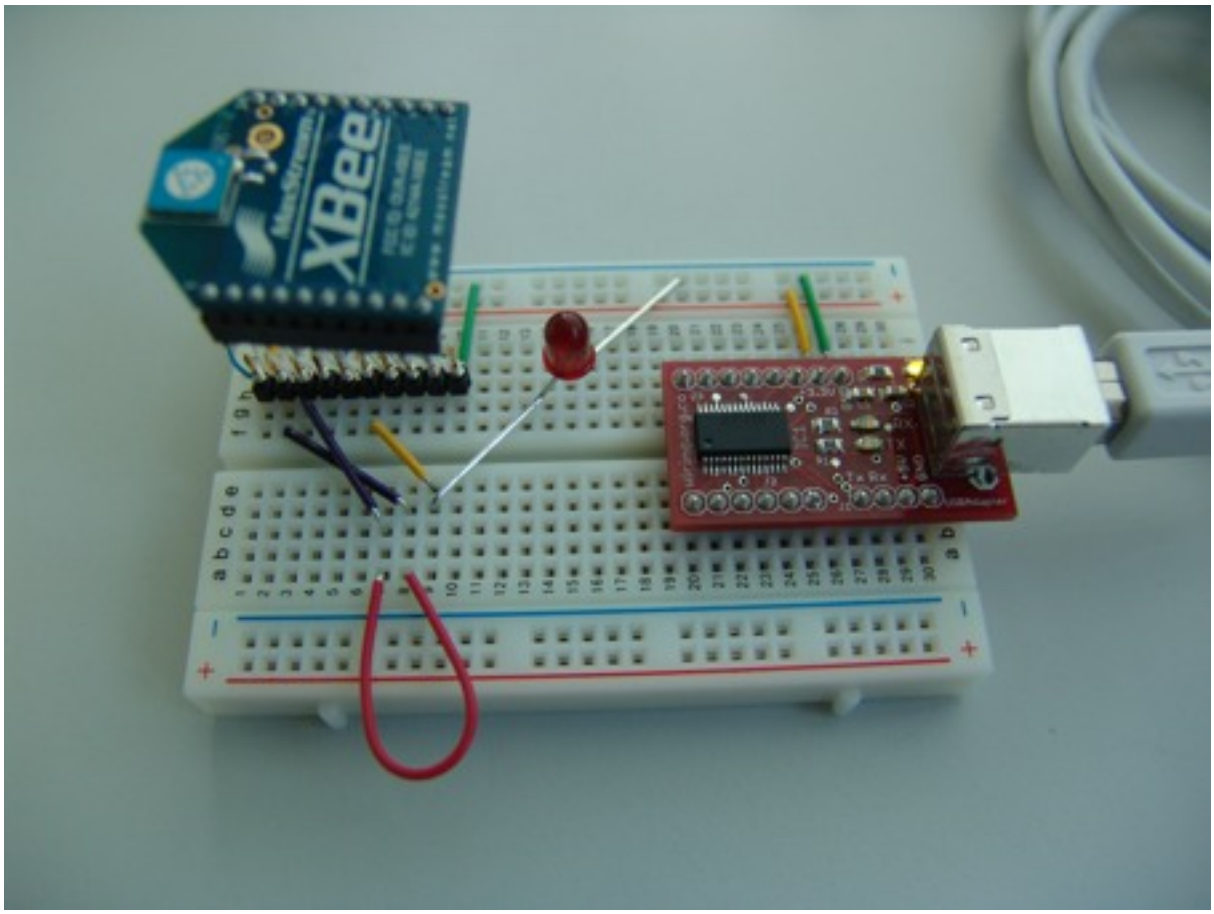

Abbildung 4: Loopback-Schaltung mit XBee

Abbildung 4 zeigt eine Loopback-Schaltung. Das USB-Modul dient hier einzig als Stromquelle für den XBee. Rx (Pin 3) und Tx (Pin 2) des XBees sind miteinander verbunden (rot). Ankommende Daten, die der XBee an Pin 2 zur Verfügung stellt, werden somit gleich wieder an Pin 3 empfangen und weiterverschickt.
Abbildung 5 zeigt den ganzen Testaufbau: Mit dem USB-Modul von MaxStream (im Starterkit mitgliefert) werden Daten über einen ersten XBee an denjenigen der Loopback-Schaltung geschickt. Dieser sendet die Daten zurück. Die DL-Adresse des einen XBee muss dabei der MY-Adresse des anderen entsprechen und umgekehrt.
Um Daten über den seriellen Anschluss eines PCs zu senden respektive ankommende Daten anzuzeigen, kann z.B. das Terminal-Programm „Realterm“[9] verwendet werden. Dabei muss neben dem richtigen Port auch die Baud-Rate ausgewählt werden. Im Fall vom mini-Wiring USB-Board ist die Baud-Rate 9600.

[9] realterm.sourceforge.net

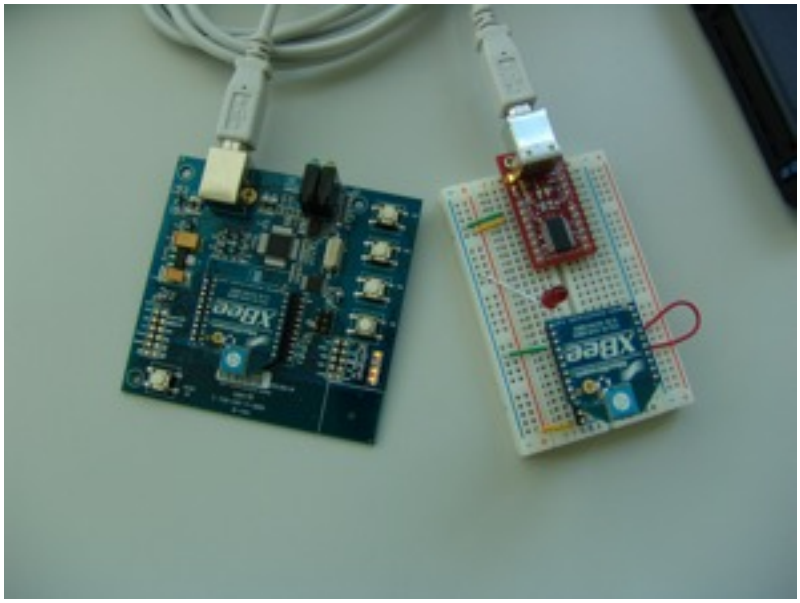

Abbildung 5: Testaufbau für Loopback-Test

Beispielapplikation 2: Steuerung einer LED (Wiring-Board)

Mit dem Loopback-Test können beliebige Daten zwischen zwei XBees versendet werden. In diesem Beispiel sollen diese Daten nun dazu dienen, ein an einen XBee angeschlossenes Wiring-Board[10] zu steuern.

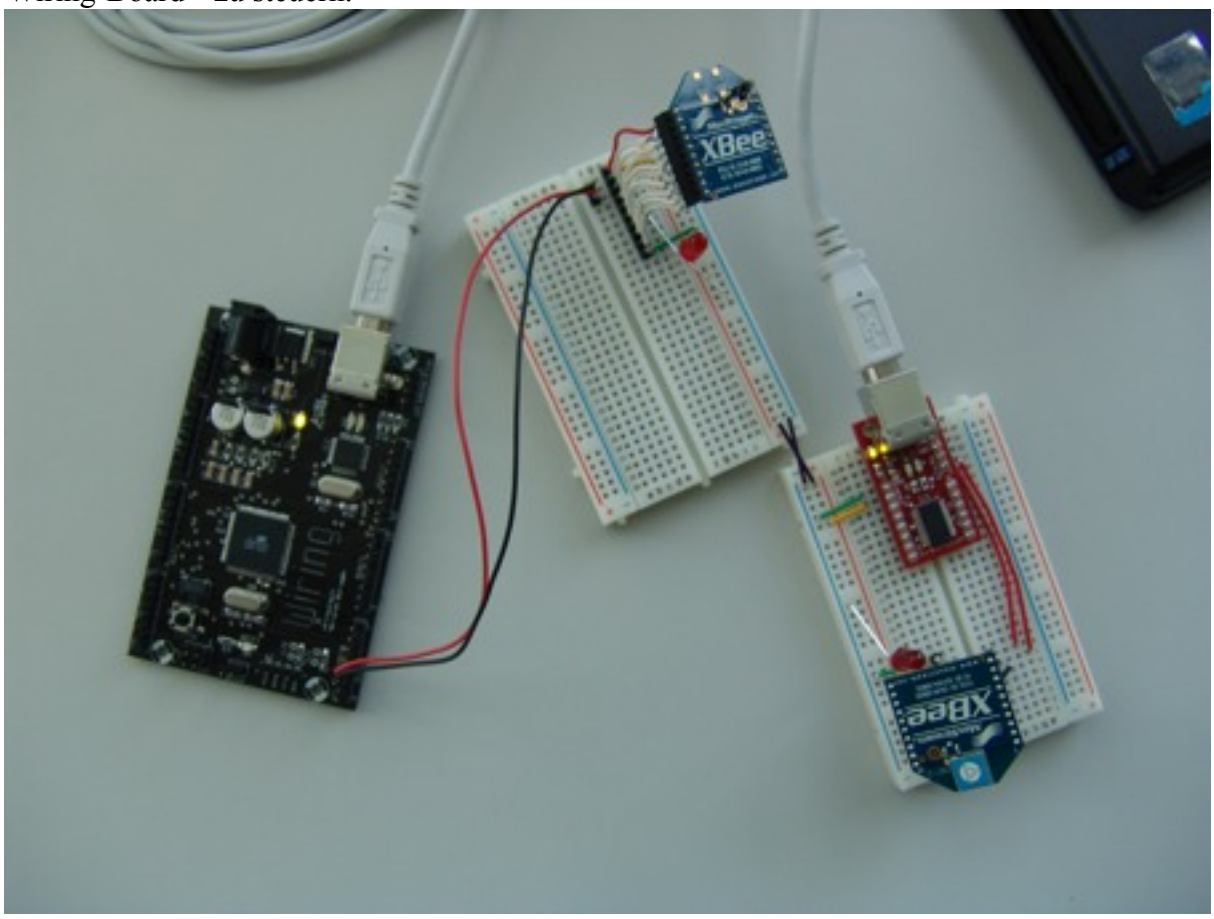

Abbildung 6: Wiring-Board erhält Befehle über XBee-Verbindung

---

[10] http://wiring.org.co/hardware/

Abbildung 6 zeigt den Versuchsaufbau: Beide XBees werden von der 3.3 Volt Stromquelle des USB-Moduls gespiesen. Der untere XBee ist über die serielle Schnittstelle mit dem USB-Modul verbunden (zwei rote Kabel) und kann damit direkt Daten an den PC senden und von diesem erhalten. Die Rx- und Tx-Pins des zweiten XBees sind mit denjenigen des Wiring-Boards verbunden (rot und schwarz). Das USB-Kabel am Wiring-Board dient nur dessen Stromversorgung und wird nicht für Datenverkehr benutzt.
Das Wiring-Board enthält ein Programm, das ankommende Zeichen interpretiert. Wird ein ‚E' empfangen, schaltet die on-board LED ein, ein ‚A' schaltet diese wieder aus. Abbildung 7 zeigt einen Ausschnitt aus dem entsprechenden Programm (Serial1 ist dabei die serielle Schnittstelle). Der gesamte Code kann aus Anhang A/Wiring Code kopiert werden.

```
if (Serial1.available()) {
  rx = Serial1.read();
  if (rx == 'E') {digitalWrite(ledpin, HIGH);}
  if (rx == 'A') {digitalWrite(ledpin, LOW);}
}
```

Abbildung 7: Code für Wiring-Board

Achtung: Unter gewissen Umständen empfängt das Wiring-Board die Daten des XBees über die serielle Schnittstelle (Rx – Tx) nicht. In diesem Fall muss der XBee mit 3.7 Volt anstatt den vorgeschriebenen 3.3 Volt betrieben werden.

Beispielapplikation 3: Drahtlose serielle Verbindung

Eine einfach zu implementierende und gleichzeitig nützliche Anwendung besteht darin, ein serielles Kabel durch zwei XBees zu ersetzen. Die Anwendung soll dabei vollkommen transparent sein, das heisst das Endprodukt soll genau gleich verwendet werden können wie ein einfaches serielles Kabel.
In diesem Beispiel entspricht die eine Seite des „Kabels" dem RS-232 Standard[11], während die andere Seite an einen PC angeschlossen werden soll, der nicht über eine RS-232 Schnittstelle verfügt, diese aber mittels USB-Anschluss und entsprechenden Treibern simuliert (dies entspricht den Voraussetzungen aus Kapitel „Verbindung zu anderen Modulen").
Der Aufbau für die „USB-Seite" entspricht exakt demjenigen in Abbildung 3. Mit einem USB-Kabel kann die Schaltung an einen PC angeschlossen werden und mittels einem Terminal-Programm können Daten gesendet und empfangene Daten angezeigt werden.
Die „RS-232-Seite" ist in Abbildung 8 dargestellt. Die serielle RS-232-Schnittstelle kann dabei mangels einer solchen am PC mit einem entsprechnden Konverter simuliert werden. Die Rx- und Tx-Pins des XBee sind dabei mit den Pins 2 und 3 des 9 poligen RS232 DB9-Steckers verbunden. Die Stromversorgung geschieht über ein Netzgerät, da die serielle Schnittstelle im Gegensatz zur USB-Schnittstelle keine Stromversorgung anbietet.

[11] http://de.wikipedia.org/wiki/EIA-232

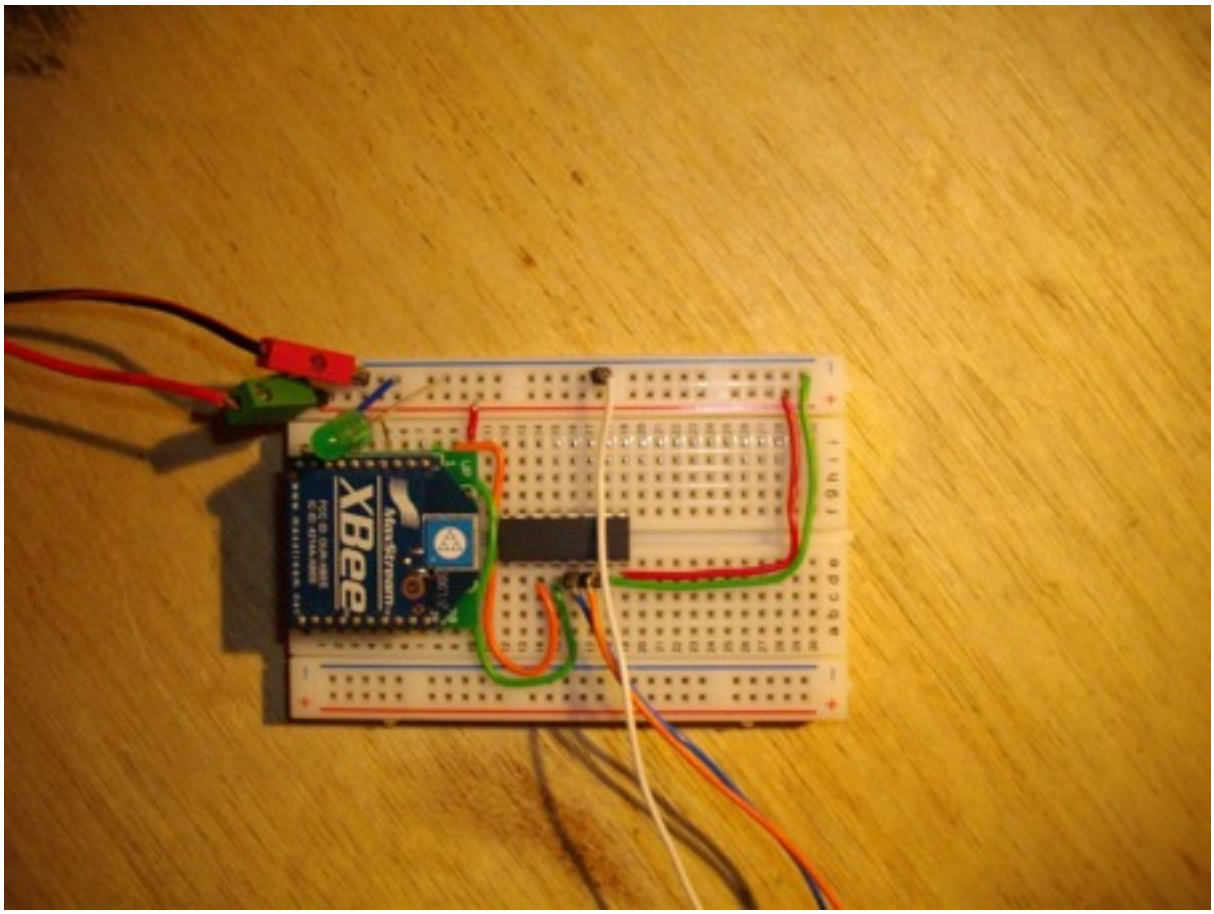

Abbildung 8: XBee mit serieller Schnittstelle verbinden

Tabelle 2 zeigt, wie die Pins eines XBees mit denjenigen eines 9-poligen seriellen Steckers verbunden werden sollen.

Tabelle 2: Verbindung XBee mit serieller Buchse/Stecker

Pin Name RS232	Pin Nr. RS232	Pin Nr. XBee	Pin Name XBee	Beschreibung
DCD	1			Data Carrier Detect
RX	2	2		Empfang
TX	3	3		Senden
DTR	4			Data Terminal Ready
GND	5	10		Ground; Signalmasse
DSR	6			Dataset Ready; Gerät einsatzbereit (0V)

RTS	7	12	Clear To Send	Request To Send; Gerät kann senden (0V) Nicht getestet, kann weggelassen werden
CTS	8	16	Request To Send	Clear To Send; Gerät empfangsbereit (0V) Nicht getestet, kann weggelassen werden
RI	9			Ring Indicator

Konvertieren von RxTx-Signal

Das 3.3 V RxTx-Signal des XBees muss an das 5 V RxTx-Signal des seriellen Ports eines üblichen PCs angepasst werden. Dazu wird ein MAX232A kompatibler Transceiver gemäss Schema in Abbildung 9 verbaut. Die ganze Schaltung kann dann mit 3 – 3.3 Volt betrieben werden.

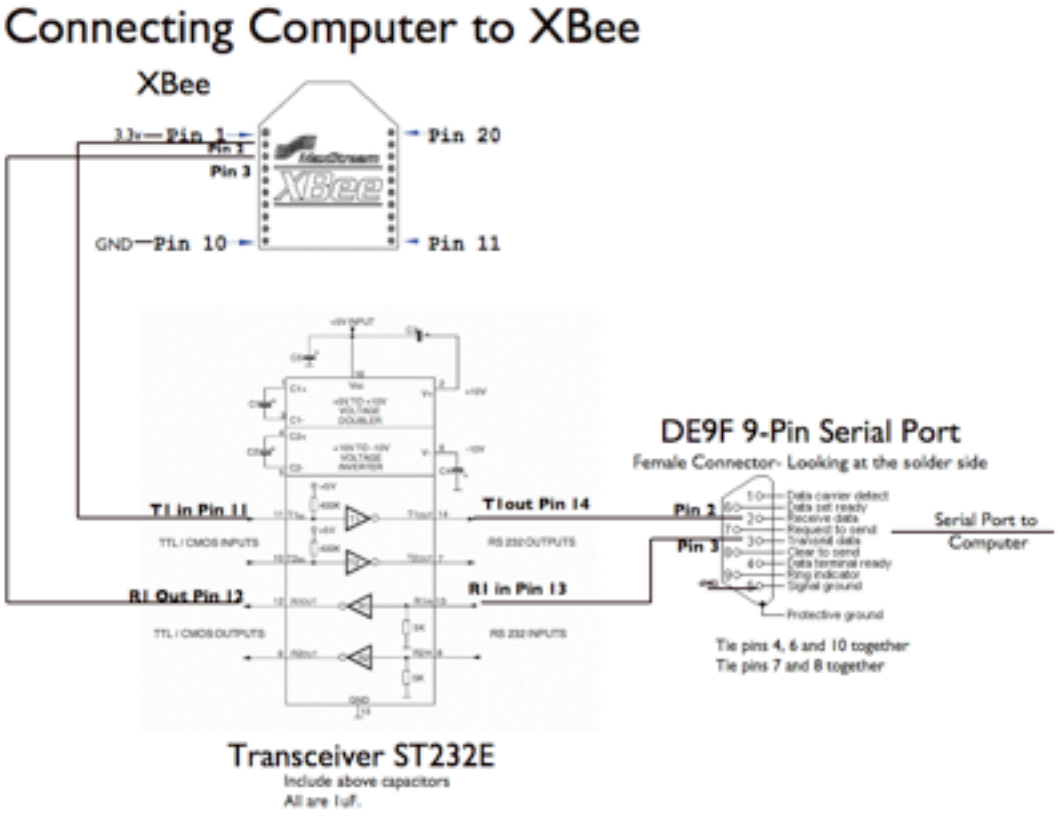

Abbildung 9: Schaltung für Einsatz von Transceiver

Beispielapplikation 4: Analoger Sensor

kommt noch....

Beispielapplikation 5: Digitaler Sensor

kommt noch....

Anhang A

Wiring Code

Code zu Beispielapplikation 2: In der setup()-Methode wird die on-board-LED getestet. Der code in der loop()-Methode sorgt neben der Steuerung der LED ausserdem dazu, dass jedes an

```
int ledpin = 48; //on board LED
char rx;

void setup()
{
  pinMode(ledpin, OUTPUT);
  Serial1.begin(9600);
  Serial.begin(9600);
  digitalWrite(ledpin, HIGH);
  delay(1000);
  digitalWrite(ledpin, LOW);
  delay(500);
  digitalWrite(ledpin, HIGH);
  delay(1000);
  digitalWrite(ledpin, LOW);
}

void loop()
{
  if (Serial.available()) {
    rx = Serial.read();
    Serial1.print(rx);
    if (rx == 'E') {digitalWrite(ledpin, HIGH);}
    if (rx == 'A') {digitalWrite(ledpin, LOW);}
  }
  if (Serial1.available()) {
    rx = Serial1.read();
    Serial.print(rx);
    if (rx == 'E') {digitalWrite(ledpin, HIGH);}
    if (rx == 'A') {digitalWrite(ledpin, LOW);}
```

Serial ankommende Zeichen an Serial1 ausgegeben wird und umgekehrt.

Anhang B

Destination Address setzen (Destination Low)

Zu sendender Befehl	**XBee Antwort**
+++	OK <CR>
ATDL <ENTER>	{Wert} <CR>
ATDL1A0D <ENTER>	OK <CR>
ATWR <ENTER>	OK <CR>
ATCN <ENTER>	OK <CR>

MY-Address setzen

Zu sendender Befehl	**XBee Antwort**
+++	OK <CR>
ATMY <ENTER>	{Wert} <CR>
ATMY1A0D <ENTER>	OK <CR>
ATWR <ENTER>	OK <CR>
ATCN <ENTER>	OK <CR>

Baud-Rate setzen

Zu sendender Befehl	**XBee Antwort**
+++	OK <CR>
ATBD <ENTER>	{Wert} <CR>
ATDB{Wert} <ENTER>	OK <CR>
ATWR <ENTER>	OK <CR>
ATCN <ENTER>	OK <CR>

Mögliche Werte:

0 1200 bps
1 2400 bps
2 4800 bps
3 9600 bps
4 19200 bps
5 38400 bps
6 57600 bps
7 115200 bps