

# VapsBee - ZigBee Lokalisierung

*Simon Kranzer, Michael Lippautz, Thomas Pfeiffenberger*

Februar 2010

EMBEDDED SOFTWARE ENGINEERING

Univ.-Prof. Dr. Christoph Kirsch

VP im Wintersemester 2009 · Institut für Computerwissenschaften · Universität Salzburg

Echtzeiterfassung und -verarbeitung von Sensordaten in einem ZigBee-Netzwerk zur Lokalisierung eines „Single Board Computers“ mittels Signalstärkemessungen.

# Inhaltsverzeichnis

<b>1</b>	<b>VapsBee</b>	<b>3</b>
1.1	Team . . . . .	4
<b>2</b>	<b>Hardware</b>	<b>4</b>
2.1	Beagleboard . . . . .	5
2.2	Igep V2 . . . . .	6
2.3	ZigBit . . . . .	6
<b>3</b>	<b>Toolchains</b>	<b>7</b>
3.1	Boards . . . . .	7
3.2	ZigBit . . . . .	8
<b>4</b>	<b>Software</b>	<b>9</b>
4.1	Sender . . . . .	11
4.2	Empfänger . . . . .	11
4.3	Parser . . . . .	12
<b>5</b>	<b>Lokalisierung</b>	<b>13</b>
5.1	Trilateration bei VapsBee . . . . .	14
<b>6</b>	<b>Fazit</b>	<b>14</b>
<b>7</b>	<b>Anhang</b>	<b>16</b>

# 1 VapsBee

Beim Projekt VapsBee werden Sensordaten mittels Zigbee<sup>1</sup> von verteilten Sensoren an einen Empfänger gesendet der, über eine serielle Schnittstelle, an ein eingebettetes System angeschlossen ist. Als Übertragungsprotokoll auf der Luftschnittstelle kommt WPAN (IEEE 802.15.4)<sup>2</sup> zum Einsatz. Sender und Empfänger sind durch Meshnetics ZigBit Module<sup>3</sup> mit dem OpenMac-Stack<sup>4</sup> realisiert. Als eingebettete Hardware wurden ein BeagleBoard<sup>5</sup> und ein Igep V2 Board<sup>6</sup> verwendet. Beide Boards besitzen einen OMAP3530 Prozessor<sup>7</sup> und diverse Schnittstellen. Als Betriebssystem wurde Angström Linux<sup>8</sup> mit einem Real-Time-Patch verwendet, wodurch alle beteiligten Komponenten ein Echtzeitverhalten unterstützen.

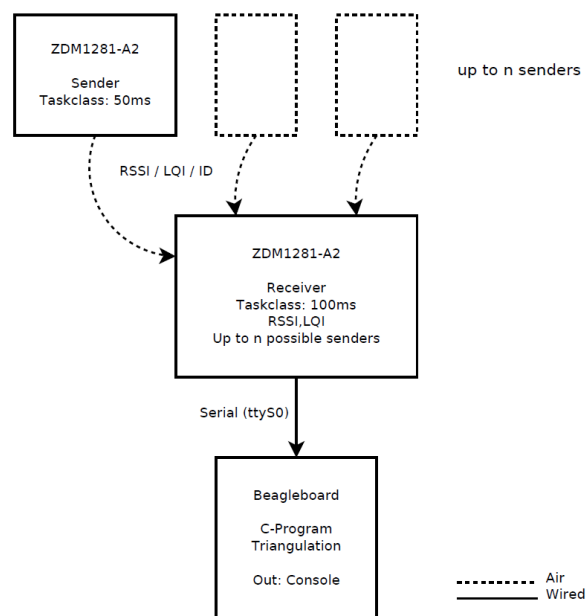


Abbildung 1: VapsBee Komponenten

Als Anwendungsfall für den beschriebenen Aufbau (Abbildung 1) wurde die Lokalisierung des eingebetteten Systems innerhalb eines definierten Bereiches gewählt. Die Sensoreinheiten senden in festen Abständen eine eindeutige Identifikationsnummer über die Luftschnittstelle zum Empfänger.

<sup>1</sup><http://www.zigbee.org/>

<sup>2</sup><http://www.ieee802.org/15/>

<sup>3</sup><http://www.meshnetics.com/zigbee-modules/>

<sup>4</sup><http://sourceforge.net/projects/openmac/>

<sup>5</sup><http://beagleboard.org/>

<sup>6</sup><http://www.igep-platform.com>

<sup>7</sup><http://focus.ti.com/docs/prod/folders/print/omap3530.html>

<sup>8</sup><http://www.angstrom-distribution.org/>

Dieser misst pro Empfänger die Stärke des Empfangenen Signals und gibt diese Informationen an das Board weiter. Anhand der Sendernummern und der Signalstärken wird dort die eigenen Position ermittelt.

## 1.1 Team

- Michael Lippautz
  - Beaglebord & Igep V2 - Setup und Programmierung
  - Präsentation, Dokumentation
- Simon Kranzer
  - ZigBit (Sender & Empfänger) - Setup und Programmierung
  - Positionierung mit Zigbee - Recherche
  - Präsentation, Dokumentation
- Thomas Pfeiffenberger
  - Positionsalgorithmus - Recherche und Programmierung
  - Präsentation, Dokumentation

## 2 Hardware

In den folgenden Abschnitten wird die für VapsBee verwendete Hardware genauer beschrieben. Abbildung 2 zeigt den gesamten Aufbau aller Komponenten.



Abbildung 2: Hardwareaufbau

Die Sendeeinheiten mit den eingebauten Sensoren befinden sich in den blauen Plastikboxen, der Empfänger ist mittels USB2Serial-Conector über einen aktiven USB-Hub am Board (Igep V2) angeschlossen. Hub und Conector ersetzen im Versuchsaufbau den direkten Anschluss des Empfängers an

eine serielle Schnittstelle des Boards. Wie in Abschnitt 2.1 näher ausgeführt wird, wurde das ursprünglich verwendete Beagleboard durch einen Kontaktfehler an der seriellen Schnittstelle zerstört.

## 2.1 Beagleboard

Das Beagleboard (Abbildung 3) ist ein „Single Board Computer“ das ursprünglich von Texas Instruments<sup>9</sup> und DigiKey<sup>10</sup> rund um den OMAP3530 Prozessor designed wurde.

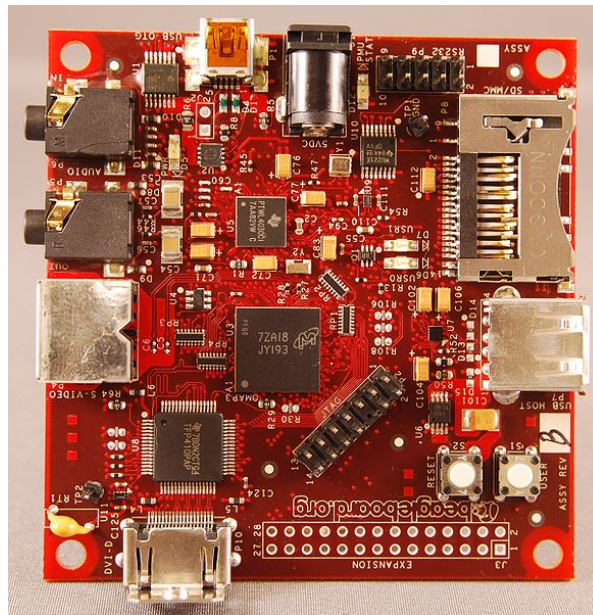


Abbildung 3: Beagleboard

### Die wichtigsten Leistungsmerkmale des Beagleboards:

- TI OMAP3530 Processor - 720MHz ARM Cortex-A8
- TMS320C64x+ DSP (520MHz bis zu 720p @30fps)
- PowerVR SGX 2D/3D Grafik
- 256MB LPDDR RAM
- 256MB NAND Flash

Das Board besitzt unter anderem eine RS232 Schnittstelle zu der im Versuchsaufbau eine Konsolenverbindung aufgebaut wurde um die Funktionalität des Programmes zu überwachen. Der Empfänger wurde mittel Kernelmux direkt an der seriellen Schnittstelle des OMAP-Chips angeschlossen. Bei

---

<sup>9</sup><http://www.ti.com/>

<sup>10</sup><http://www.digikey.com/>

abschließenden Tests kam es dabei zu einem Kurzschluss der den Chip und damit das Board unbrauchbar machte. Für die abschließende Präsentation von VapsBee wurde daher das Igep V2 Board als Alternativlösung verwendet. Dieses wird im nachfolgenden Abschnitt 2.2 kurz vorgestellt.

## 2.2 Igep V2

Der spanische Hersteller ISEE<sup>11</sup> entwickelte das Igep V2 ebenfalls rund um den OMAP3530 von Texas Instruments. Das Board (Abbildung 4) bietet zusätzlich zu den vom Beagleboard angebotenen Merkmalen insbesondere die Möglichkeit Geräte auch per USB, Ethernet und WLAN anzubinden.

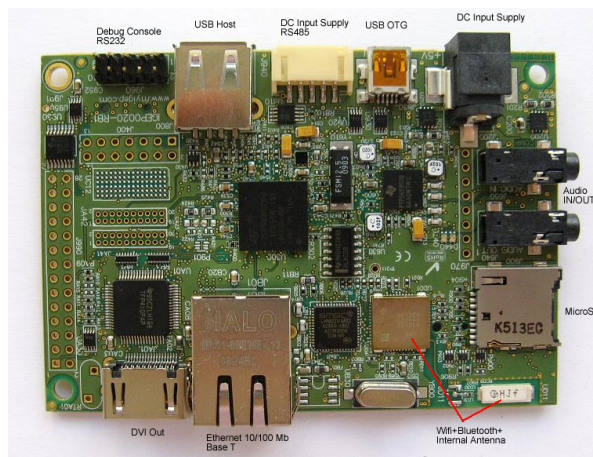


Abbildung 4: Igep V2

### Leistungsmerkmale:

- TI OMAP3530 Processor - 720MHz ARM Cortex-A8
- TMS320C64x+ DSP (520MHz bis zu 720p @30fps)
- PPOWERVR SGX 530 Grafik
- 4GB LPDDR RAM
- 4GB NAND Flash

## 2.3 ZigBit

Das Meshnetics ZDM1281-A2 ZigBit-Modul operiert im 2,4 Ghz Frequenzband und verwendet den IEEE 802.15.4 Standard. Als Microcontroller kommt ein Atmel ATmega1281<sup>12</sup> zum Einsatz. Dieser besitzt 128Kb Flashspeicher

<sup>11</sup><http://www.igep-platform.com/>

<sup>12</sup><http://www.atmel.com>

und arbeitet mit einer Taktfrequenz von 16MHz. Als Sende- und Empfangseinheit ist ein AT86RF230 ZigBee-Modul verbaut.

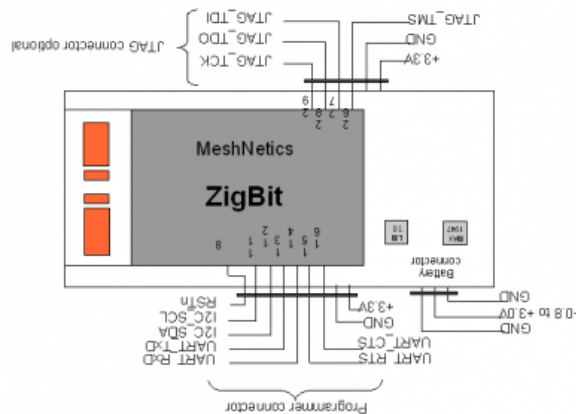


Abbildung 5: ZigBit Schnittstellen

**Das Zigbit bietet standardmäßig folgende Sensoren an:**

- Temperatur
- Feuchte
- Licht

Zusätzlich dazu ist auf den verwendeten Modulen ein Beschleunigungssensor verbaut. Die genaue Spezifikation dieses Sensors ist den Autoren nicht bekannt, er wurde über die GPIOs (Abbildung 5) des ZigBit angeschlossen und liefert relative Werte für x,y und z-Achse.

### 3 Toolchains

Nach Beschreibung der verwendeten Hardware geben die nächsten beiden Abschnitte einen Überblick über die im Projekt VapsBee verwendeten Softwaretools die zur Programmierung der Komponenten notwendig sind.

#### 3.1 Boards

Sowohl das Beagleboard als auch das Igep V2 wurden mit Hilfe des OpenEmbedded-Frameworks<sup>13</sup> mit einer reduzierten Version der Angsröm Distribution von Embedded Linux aufgesetzt. Dabei werden der Kernel und das Root-Filesystem

<sup>13</sup><http://wiki.openembedded.net/>

unter Angabe der Zielhardware erstellt und können dann direkt auf die Speicherkarten der Boards transferiert werden. Das dabei entstandene Standard-Linux-System bietet keine besondere Echtzeitfunktionalität. Userspace Programme können zwar priorisiert werden, der Kernel kann jedoch nicht unterbrochen werden wenn eine der folgenden Situationen auftritt:

- Die CPU ist im Usermode
- Der Kernel gibt Werte an den Userspace zurück (Systembefehle)
- Der Kernel ist durch eine Mutex gesperrt

Aus den genannten Gründen wurde zusätzlich der offizielle Linux-Echtzeitpatch<sup>14</sup> auf den Kernel angewendet. Ein derart veränderter Kernel kann zwar keine harten Echtzeitgarantien geben, es ist jedoch möglich Programmen eine Priorität zuzuweisen, die höher als die des Kernels ist. Dadurch ergeben sich folgende Möglichkeiten:

- Der Kernel kann unterbrochen werden
  - Mutex
  - Spinlocks
  - Interrupts

Wenn nun ein einzelner Task mit einer höheren Priorität als der Kernel läuft kann dieser als Echtzeittask geplant werden.

## 3.2 ZigBit

Die ZigBit-Module wurden über einen USB2Serial Converter am PC angeschlossen. Mit Hilfe der von Meshnetics zur Verfügung gestellten Bootloader-Applikation können Stack und Programme in den Flash transferiert werden. Als Stack wurde der offene OpenMac (Abbildung 6) verwendet, der auf TinyOs<sup>15</sup> basiert. TinyOs ist eventbasiert und wurde eigens für Sensornetzwerke entwickelt.

---

<sup>14</sup><http://rt.wiki.kernel.org>

<sup>15</sup><http://www.tinyos.net/>



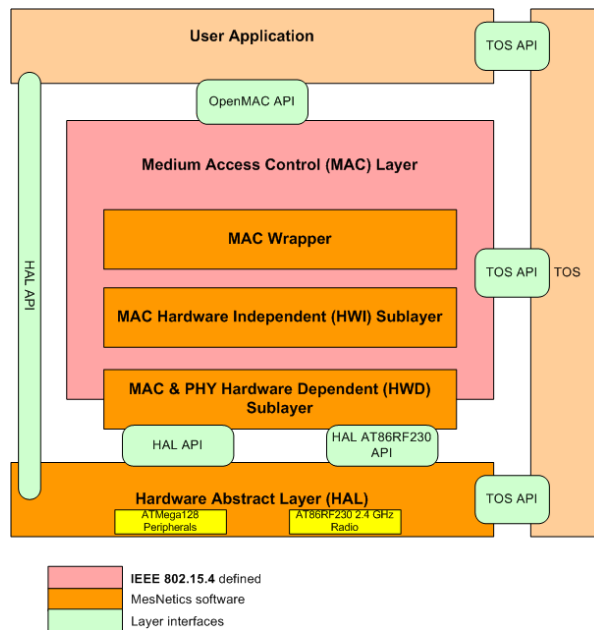


Abbildung 6: OpenMac Architektur

Als Programmierumgebung wurde der VIM unter CentOS 5<sup>16</sup> verwendet. Der Compiler stammt von Luxoft Labs und übersetzt Programmcode der in MeshC, einer Erweiterung von NesC für TinyOS.

## 4 Software

Aufbauend auf die beschriebene Hardware- und Softwareumgebung wurden drei Programmkomponenten entwickelt. Das Senderprogramm läuft auf den einzelnen ZigBit-Sendern und sendet neben den Sensordaten eine eindeutige Identifikationsnummer über das ZiogBee-Netzwerk. Das Empfängerprogramm empfängt diese Daten am ZigBit-Empfänger und verarbeitet sie. Zusätzlich führt er Messungen der Signalstärke und der Verbindungsqualität durch und sendet die ermittelten und teils errechneten Daten via serieller Schnittstelle weiter an das Board. Dort werden die Informationen vom Parser interpretiert und es wird die aktuelle relative Position errechnet.

Abbildung 7 visualisiert nochmals alle beteiligten Teile dieses verteilten Programmes.

<sup>16</sup><http://centos.org/>

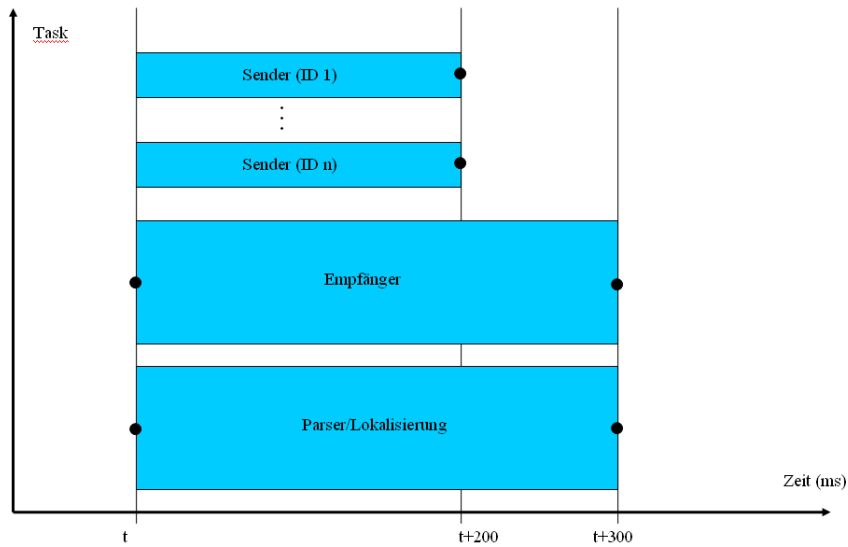


Abbildung 7: Das VapsBee Programm

**Timing** Die VapsBee Komponenten besitzen folgendes Zeitverhalten:

- Sender:
  - Sendraten ab 10ms.
  - 200ms (Demoprogramm)
- Empfänger:
  - Empfang und Verarbeitung ab 50ms.
  - 300ms (Demoprogramm)
- Parser, Lokalisierung, Ausgabe
  - Empfang und Verarbeitung ab 100ms.
  - 300ms (Demoprogramm)

Daraus ergibt sich für das Demoprogramm ein Ablauf wie er in Abbildung 8 dargestellt ist.

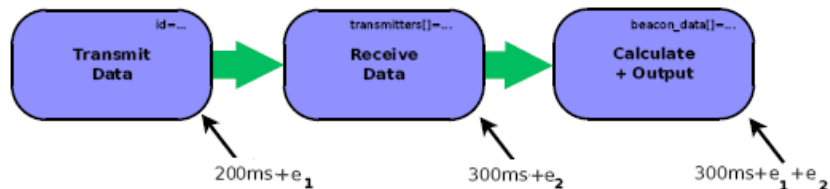
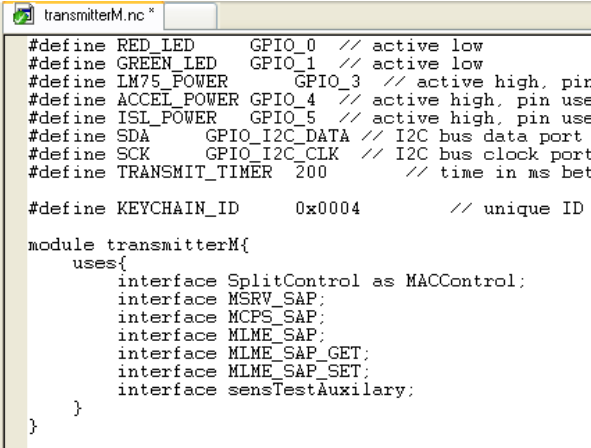


Abbildung 8: VapsBee Timing

Nachfolgend werden nun Ausschnitte aus den einzelnen Implementierungen gezeigt. Der vollständige Sourcecode findet sich im Anhang 7 dieses Dokumentes.

## 4.1 Sender

Die Sender sind als periodische Tasks implementiert. Der auslösende Event „TRANSMIT\_TIMER\_EVENT“ (Abbildung 9) wird alle „TRANSMIT\_TIMER“ Millisekunden ausgelöst. Werden alle Sensorendaten erfasst und versendet ergibt sich für den „TRANSMIT\_TIMER“ ein minimaler Wert von 10 Millisekunden. Dieser Wert wurde in Versuchen (Trial and Error) ermittelt.



```
transmitterM.nc
#define RED_LED    GPIO_0 // active low
#define GREEN_LED  GPIO_1 // active low
#define LM75_POWER GPIO_3 // active high, pin
#define ACCEL_POWER GPIO_4 // active high, pin use
#define ISL_POWER  GPIO_5 // active high, pin use
#define SDA        GPIO_I2C_DATA // I2C bus data port
#define SCK        GPIO_I2C_CLK  // I2C bus clock port
#define TRANSMIT_TIMER 200 // time in ms bet

#define KEYCHAIN_ID 0x0004 // unique ID

module transmitterM{
  uses{
    interface SplitControl as MACControl;
    interface MSRV_SAP;
    interface MCPS_SAP;
    interface MLME_SAP;
    interface MLME_SAP_GET;
    interface MLME_SAP_SET;
    interface sensTestAuxiliary;
  }
}
```

Abbildung 9: Senderprogramm (Auszug)

Innerhalb eines Zyklus werden dabei folgende Aufgaben erledigt:

1. Erfassung der Daten von den Sensoren (Liefert ein Sensor innerhalb des Zeitfensters keinen Wert wird der zuletzt gültige Wert gehalten.)
2. Verpacken der gesehene Werte in einem statischen Buffer
3. Versenden der Daten aus dem Buffer an die serielle Schnittstelle des Senders inkl. ID
4. Versenden der Daten aus dem Buffer über die Luftschnittstelle

## 4.2 Empfänger

Der Empfänger ist, wie der Sender, als periodischer Task implementiert. Die minimale Taskzeit wurde mit 50 Millisekunden ermittelt. Der dazu verwendete Event ist „RECEIVE\_PACKET“. Innerhalb dieser Zeit werden folgende Schritte durchgeführt:

1. Empfangen von Paketen auf der Luftschnittstelle (Zeitgleich) - Es können Pakete von mehreren Sendern innerhalb dieser Zeit empfangen werden. Bei Versuchen wurden bis zu fünf aktive Sender verwendet. Die empfangenen Werte werden in den Empfangsbuffer des Moduls zwischengespeichert.
2. Iteration über den Empfangsbuffer
  - ID und Daten aus den Paketen auspacken
  - Messung von Signalstärke und Verbindungsqualität
  - Berechnung von Mittelwerten
  - Aufbereitung der Daten in einem String
  - Versenden des Strings an die serielle Schnittstelle

Abbildung 10 zeigt die vom Empfänger aufbereiteten Daten, wie diese an der seriellen Schnittstelle am Board ankommen. Für den Versuchsaufbau wurden nur die jeweilige ID, die Verbindungsqualität (LQI<sup>17</sup>) und die mittlere und die aktuelle Signalstärke (RSSI<sup>18</sup>) übertragen.

```

SMT ; LQI : 255 ; RSSI : 67 (08) ; ID : 02
SMT ; LQI : 255 ; RSSI : 64 (09) ; ID : 03
SMT ; LQI : 255 ; RSSI : 70 (07) ; ID : 01
SMT ; LQI : 255 ; RSSI : 76 (05) ; ID : 02
SMT ; LQI : 255 ; RSSI : 64 (09) ; ID : 03
SMT ; LQI : 255 ; RSSI : 67 (08) ; ID : 02
SMT ; LQI : 255 ; RSSI : 73 (06) ; ID : 01
SMT ; LQI : 255 ; RSSI : 76 (05) ; ID : 02
SMT ; SMT ; LQI : 255 ; RSSI : 73 (06) ; ID : 01
SMT ; LQI : 255 ; RSSI : 70 (07) ; ID : 01
SMT ; LQI : 255 ; RSSI : 76 (05) ; ID : 02
SMT ; LQI : 255 ; RSSI : 64 (09) ; ID : 03
SMT ; LQI : 255 ; RSSI : 64 (09) ; ID : 02
SMT ; LQI : 255 ; RSSI : 61 (10) ; ID : 02
SMT ; LQI : 255 ; RSSI : 64 (09) ; ID : 03
SMT ; LQI : 255 ; RSSI : 55 (12) ; ID : 02
SMT ; LQI : 255 ; RSSI : 64 (09) ; ID : 03
SMT ; LQI : 255 ; RSSI : 73 (06) ; ID : 01
SMT ; LQI : 255 ; RSSI : 52 (13) ; ID : 02
SMT ; LQI : 255 ; RSSI : 70 (07) ; SMT ; LQI : 255 ; RSSI : 67 (08) ; ID : 03
SMT ; LQI : 255 ; RSSI : 49 (14) ; ID : 02

```

Abbildung 10: Screenshot der Minicom am IgepV2

### 4.3 Parser

Diese Komponente der Software hat die Aufgabe aus den am Board ankommenden Daten die aktuelle relative Position zu bestimmen.

Es sind pro Zyklus folgende Schritte notwendig:

1. Zeilenweise lesen der Daten aus dem Kernelbuffer der seriellen Schnittstelle
2. Zwischenspeichern der erfassten Werte
3. Übergabe der Werte an die Positionsberchnung

<sup>17</sup>Link Quality Indicator

<sup>18</sup>Received Signal Strength Indicator

4. Sleep 300 ms.

Durch eine Priorität höher als die des Kernels und durch ein zeitgesteuertes pausieren des Tasks wurde ein periodisches Zeitverhalten realisiert (siehe nachfolgendes Listing).

```
set scheduling priority // 1 above kernel
/* open & configure serial port (beagle) // data source */
open & configure usb2serial port (igep)
acquire memory // use to store node data
lock heap // no memory alloc after this point

while (TRUE)
{
sleep [xxx us - xx ms]
<<threaded>>
read data into memory
calc position
output
}
```

## 5 Lokalisierung

Dieser Abschnitt gibt zuerst einen kurzen Überblick zu Lokalisierung mit ZigBee. Danach wird der beim Demoprogramm verwendete Algorithmus (Trilateration) näher diskutiert.

**Lokalisierung mit ZigBee** Es existieren mehrere Ansätze mittels RSSI und LQI eine Lokalisierung in ZigBee-Netzwerken zu realisieren. Dabie können zwei grundsätzliche Herangehensweisen unterschieden werden:

**Triangulierung:** Bei der Triangulierung werden die Winkel zu den Referenzpunkten zur Berechnung der eigenen Position herangezogen.

**Trilateration** Bei der Trilateration werden die Entfernungen zu den Referenzpunkten zur Positionierung verwendet. Meist kennt der Empfänger die Position der Referenzpunkte oder diese senden ihre Position periodisch aus. Im Zusammenhang mit ZigBee wurde insbesondere die Methode des WCL<sup>19</sup> verwendet [RG07]. Dabei wird aus RSSI und LQI die Entfernung zu den Referenzpunkten ermittelt, wobei die Position von nähergelegenen Sendern stärker in die aktuelle Berechnung eingebunden wird.

---

<sup>19</sup>Weighted Centroid Localization

## 5.1 Trilateration bei VapsBee

Für den Versuchsaufbau wurde die Trilateration (Abbildung 11) verwendet, wobei alle Referenzpunkte zur Zeit gleichgewichtet werden und der Empfänger die Position der Sender nicht kennt. Dies ermöglicht nur die Berechnung einer relativen Position und erfordert zumindest drei aktive Sender. Eine Erweiterung durch den evaluierten Algorithmus (WCL) ist jedoch jederzeit möglich, es müsste jeder Sender zusätzlich seine aktuelle Position senden, oder es müssten fixe Positionen pro Sender-ID am Empfänger hinterlegt werden.

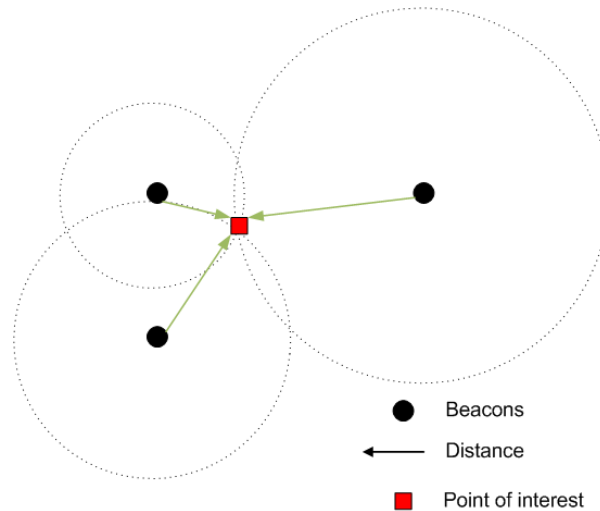


Abbildung 11: Trilateration

Für das weiters notwendige Mapping der Signalstärkewerte auf die Distanz und die Abstandskalkulation wurden folgende Formeln herangezogen [SX05].

$$RSSI(d) = RSSI(d_0) + 10 * n * \log\left(\frac{d}{d_0}\right)$$

$$dist_A^2 = (x_0 - x_A)^2 + (y_0 - y_A)^2$$

## 6 Fazit

Abschließend ist festzustellen, dass die größten Probleme bei der Entwicklung von VapsBee im Anwendungsfall Lokalisierung lagen. Um das grundsätzliche Verhalten der verschiedenen Komponenten, insbesondere deren Zeitverhalten, zu zeigen wäre es leichter gewesen andere Sensorwerte, wie etwa die Temperatur oder die Beschleunigung zu erfassen. Eine Positionierung innerhalb eines Funknetzwerkes wie Zigbee weist zu hohe Lokalisierungsfehler auf und kann nur in einem Bereich von mehreren 10 Metern genau angegeben werden (auch [RG07]).

## Literatur

- [RG07] Ralf Grossmann et. al.. *Localization in Zigbee-based Sensor Networks*. WISP 2007, Madrid, Spanien, 2007.
- [SX05] Xingfa Shen et. al.. *Connectivity and RSSI Based Localization Scheme for Wireless Sensor Networks*. Lecture Notes in Computer Science, Springer, Berlin, 2005.

## **7 Anhang**

**Sender und Empfänger** file:transiver.rar

**Parser und Lokalisierung** file:chiroptera.rar

**Präsentation** file:Praesentation\_ESE\_VapsBee.pdf