Besucherinteraktion auf Veranstaltungen mit der OpenBeacon-Technologie

Abschlussarbeit

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

an der

Hochschule für Technik und Wirtschaft Berlin
Fachbereich Wirtschaftswissenschaften II
Studiengang Angewandte Informatik

1. Prüfer: Prof. Dr. Jürgen Sieck
2. Prüfer: Dr.-Ing. Michael A. Herzog

eingereicht von: Stephan Bergemann
Matrikelnummer: XXXXXX

Datum: 13. Dezember 2010

Danksagung

Hier kommen die ganzen Personen und Sätze zu ihnen rein, denen man danken möchte ...

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation 1

1.2 Zielsetzung 1

1.3 Aufbau der Arbeit 1

2 Grundlagen **2**

2.0.1 Umgebungen 3

3 Systementwurf **7**

4 Implementierung **8**

5 Test und Auswertung **10**

6 Zusammenfassung und Ausblick **11**

Literatur **14**

Literaturverzeichnis 14

weitere Internetquellen 14

Bildquellen 15

Abbildungsverzeichnis **16**

Tabellenverzeichnis **17**

Listings **18**

1 Einleitung

Dieses Dokument dient als Beispiel einer Abschlussarbeit, wie sie unter anderem auch von mir an der HTW Berlin im Sommer 2010 eingereicht wurde und mir zum erfolgreichen Abschluss meines Bachelorstudiums verholfen hat.

1.1 Motivation

Dieses Dokument entstand, um es ihnen zu erleichtern, ihre Abschlussarbeit mit LaTeX sauber zu setzen und somit nicht mehr allzu sehr auf kommerzielle Programme angewiesen zu sein.

1.2 Zielsetzung

Es soll vermittelt werden, wie so ein Dokument entstehen kann.

1.3 Aufbau der Arbeit

wird erzeugt durch

```
1 pdflatex bachelorarbeit.tex && pdflatex bachelorarbeit.tex && bibtex www ↩
    && bibtex bilder && bibtex bachelorarbeit && pdflatex bachelorarbeit. ↩
    tex
```

2 Grundlagen

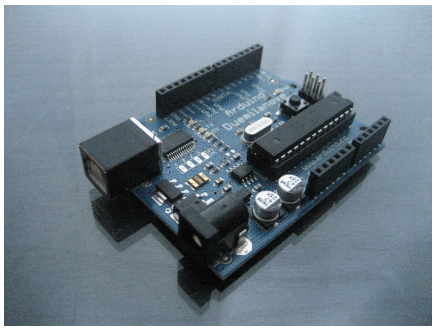
Das Verwenden von LaTeX für wissenschaftliche Arbeiten und Publikationen ist international sehr weit verbreitet. Daher verwundert es auch nicht, dass nahezu für jedes schritfsatztechnische Problem eine Lösung für LaTeX entwickelt und zur freien Verfügung gestellt wurde. So kann man mit LaTeX nicht nur Briefe und Bücher schreiben, sondern auch mathematische Formeln, Notensätze, Nassi-Shneiderman-Diagramme und vieles mehr[1] .

Dabei muss man sich zwar zunächst in die Syntax dieser Markup-Sprache einarbeiten, schreibt aber dann ziemlich schnell sehr gut aussehenede Dokumente zusammen. Und schließlich muss man sich in die Arbeitsweise mit anderen (beispielsweise Office-) Programmen auch erst einarbeiten.

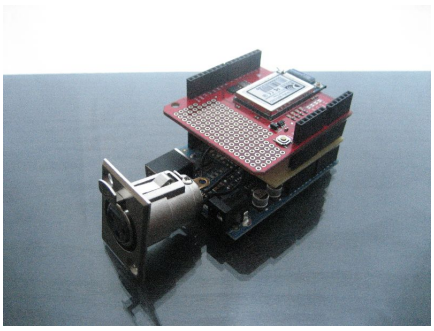
Schnell stößt man mit LaTeX oft an einen Punkt, wo man sich denkt, dass man eigentlich die Schriftgröße an dieser oder jener Stelle kleiner oder größer haben möchte, oder gar den gesamten Satzspiegel verändern möchte. Dies ist allerdings nicht ratsam, da die Vorlagen für LaTeX-Dokumente sich normalerweise ziemlich exakt an Normen und den goldenen Schnittalten. Wer davon noch nie etwas gehört hat, dem sei gesagt, dass die meisten Dokumente anderer Programme schon allein deswegen oft unakzeptabel aussehen, weil sie eben keinen ordentlichen Satzspiegel haben. Wer davon schonmal was gehört hat, wird selbst gar nicht auf den Gedanken kommen, etwas an den Vorgaben ändern zu wollen ;) .

Hier nun also eine Lose Sammlung von ein paar Dingen, wie man sie mit LaTeX umsetzen kann.

[1] Natürlich funktionieren auch Fußnoten mit LaTeX



(a) das Arduino Entwicklungsboard



(b) gestapelte Shields auf dem Arduinoboard

Abbildung 2.1: Arduino Hardware

2.0.1 Umgebungen

In verschiedenen Umgebungen kann man verschiedene Dinge adäquat darstellen. Der Inhalt der Umgebungen wurde hier aus Faulheit einfach aus meiner Bachelorarbeit übernommen :) .

Eine sogenannte Description:

Passive Transponder beziehen die Energie zum Auslesen des gespeicherten Wertes und entsprechender Modulation des Feldes des Lesegeräts aus dem Feld des Lesegeräts selbst - entziehen diesem Feld also Energie.

Aktive Transponder besitzen eine eigene Energieversorgung zum Auslesen des Speichers und zur Modulation des Feldes.

Mit dem Paket `subfig` kann man Subfiguren erstellen.
Auf die verschiedenen Unterbilder kann man natürlich auch einzeln verlinken: 2.1a
Anführungszeichen werden links mit `\glqq` („) und den rechten mit `\grqq~` (“) setzen, wobei die Tilde an den rechten Anführungszeichen nur ein geschütztes Leerzeichen

ist - notwendig, wenn danach noch Worte folgen sollen.

Zitate sind in LaTeX komfortabel möglich: [l_WB09, S. 253 ff.] oder auch [w_Wik10]

Mit `\enlargethispage{\baselineskip}` erreicht man die Verlängerung einer Seite um eine Zeile (um Hurenkinder und Schusterjungen zu umgehen).

Grafiken werden mit dem eigens dafür implementieren Befehl `\grafik` eingebunden:

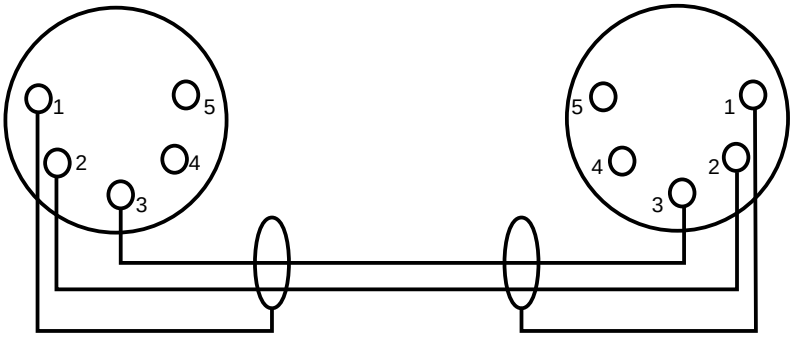


Abbildung 2.2: Verbindungsschema zweier DMX-Komponenten nach: [b_ard]

mit `\ScaleIfNeeded` erreicht man, dass die Grafik auf die Seitenbreite runterskaliert wird, wenn nötig:

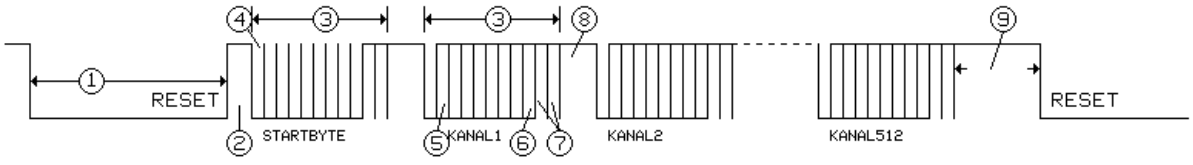


Abbildung 2.3: Das DMX-512 Protokoll [b_sou]

Und so könnte eine absolut positionierte Tabelle aussehen:

Produktqualität	**sehr gut**	**gut**	**normal**	**nicht relevant**
Funktionalität				
Angemessenheit		x		
Richtigkeit			x	
Interoperabilität		x		
Ordnungsmäßigkeit			x	
Sicherheit		x		
Zuverlässigkeit				
Reife		x		
Fehlertoleranz		x		
Wiederherstellbarkeit		x		
Benutzbarkeit				
Verständlichkeit		x		
Erlernbarkeit			x	
Bedienbarkeit		x		
Effizienz				
Zeitverhalten		x		
Verbrauchsverhalten		x		
Änderbarkeit				
Analysierbarkeit		x		
Modifizierbarkeit		x		
Erweiterbarkeit	x			
Stabilität	x			
Prüfbarkeit			x	
Übertragbarkeit				
Anpassbarkeit		x		
Installierbarkeit		x	x	
Konformität			x	
Austauschbarkeit		x		

Tabelle 2.1: Qualitätskriterien des zu entwickelnden Systems

auf diese Tabelle kann man natürlich auch verlinken: Tabelle 2.1.

3 Systementwurf

4 Implementierung

Ein bisschen mit Quelltexten hantieren kann man mit LaTeX natürlich auch wunderbar. Hier sind noch große Potenziale, was beispielsweise Syntaxhighlighting und ähnliche Spielereien angeht. Ich beschränkte mich (auch wegen der daraus resultierenden erhöhten Anzahl in Farbe auszudruckender Seiten auf Grautöne.

```
1  void ProximityDataReceiver::createTagToTagContactNewsPackage(PTNP::oid
       oid1, PTNP::oid oid2, PTNP::duration duration, PTNP::timestamp end) {
2      // Datensatz vorbereiten und einen Stream dafür
3      QByteArray *packageData = new QByteArray();
4      QDataStream packageDataStream(packageData, QIODevice::WriteOnly);
5
6      // alle Daten in den Datensatz schreiben
7      packageDataStream<<oid1;
8      packageDataStream<<oid2;
9      packageDataStream<<duration;
10     packageDataStream<<end;
11     _packageQueueMutex.lock();
12     // das Paket mit dem Datensatz an die Warteschlange anhängen
13     _queue.append(new PTNP::Package(PTNP::TAG_TO_TAG_CONTACT_NEWS,
           packageData));
14     _packageQueueMutex.unlock();
15     // das Signal emitieren, dass neue Daten vorhanden sind
16     emit receivedData();
17 }
```

Listing 4.1: Routine zur Erzeugung eines Paketes über einen laufenden Kontakt

Natürlich kann auch für jedes neue Listing alles neu gesetzt werden - hier nur die Sprache von C++ auf C.

```
1  for (;;)
2    {
3      // wenn Daten vorhanden sind
4      if (nRFCMD_WaitRx (10))
5        {
6        vLedSetRed (1); // Rote LED anschalten
7        do
8          {
9            // empfangene Daten vom Chip auslesen
10           nRFCMD_RegReadBuf (RD_RX_PLOAD, g_Beacon.byte,
11            sizeof (g_Beacon));
12           vLedSetGreen (0); // Grüne LED ausschalten
13           u_int8_t i=0;
14       // Absender schicken
15           vUSBSendByte(0x00);
16           vUSBSendByte(0xff);
17           vUSBSendByte(0x00);
18           vUSBSendByte(0xff);
19       // alle empfangenen Daten schicken
20           for(i=0; i<sizeof(g_Beacon); i++)
21           {
22             vUSBSendByte(g_Beacon.byte[i]);
23           }
24           vLedSetGreen (1); // Grüne LED anschalten
25         }
26       // solange die Warteschlange nicht leer ist
27       while ((nRFAPI_GetFifoStatus () & FIFO_RX_EMPTY) == 0);
28       vLedSetRed (0); // Rote LED ausschalten
29       }
30     nRFAPI_ClearIRQ (MASK_IRQ_FLAGS);
31   }
```

Listing 4.2: Routine zur Datenweiterleitung des USB-Nodes

5 Test und Auswertung

6 Zusammenfassung und Ausblick

Anhänge

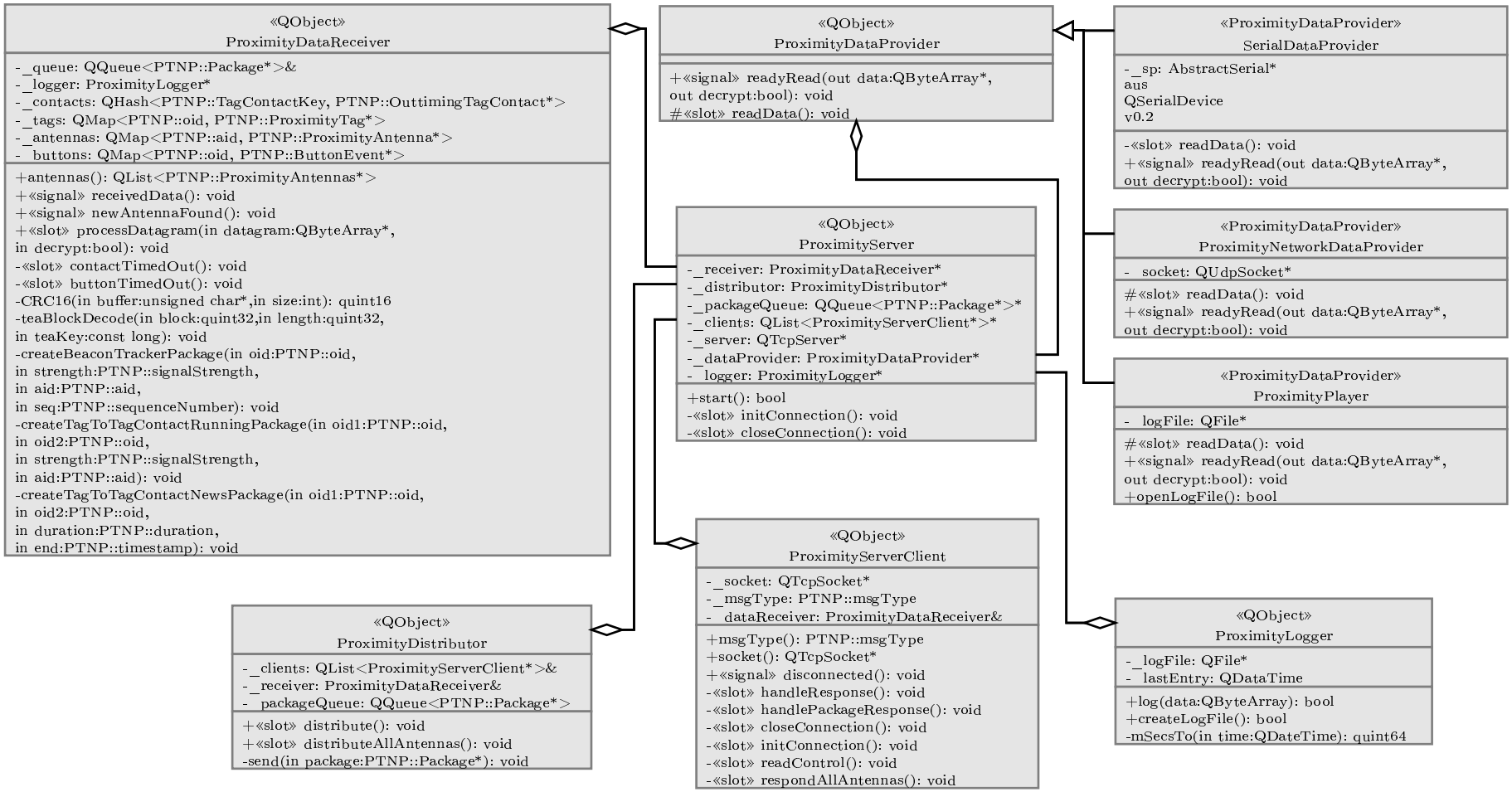


Abbildung .1: Klassendiagramm der RFID-Server-Komponente

Literaturverzeichnis

[l_WB09] Wörn, Heinz ; Brinkschulte, Uwe: *Echtzeitsysteme: Grundlagen, Funktionsweisen, Anwendungen*. Erste Aufl. Springer, Berlin, 2009

weitere Internetquellen

[w_Wik10] Wikipedia: *QR-Code*. http://de.wikipedia.org/wiki/QR-Code. Version: 2010. – [Online; abgerufen am 14. Mai 2010]

Bildquellen

[b_ard] ARDUINO.CC: *5-polige DMX-Verbindung.* `http://www.arduino.cc/playground/uploads/DMX/5_pol_xlr.jpg`. – [Online; abgerufen am 05. Juni 2010]

[b_sou] SOUNDLIGHT: *Das DMX-512 Protokoll.* `http://www.soundlight.de/techtips/dmx512/vpdmxti.gif`. – [Online; abgerufen am 18. Mai 2010]

Abbildungsverzeichnis

2.1 Arduino Hardware 3
2.2 Verbindungsschema zweier DMX-Komponenten 4
2.3 Das DMX-512 Protokoll 4

.1 Klassendiagramm der RFID-Server-Komponente 13

Tabellenverzeichnis

2.1 Qualitätskriterien des zu entwickelnden Systems 5

Listings

4.1 Routine zur Erzeugung eines Paketes über einen laufenden Kontakt . . . 8
4.2 Routine zur Datenweiterleitung des USB-Nodes 9

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Ort, Datum	Unterschrift