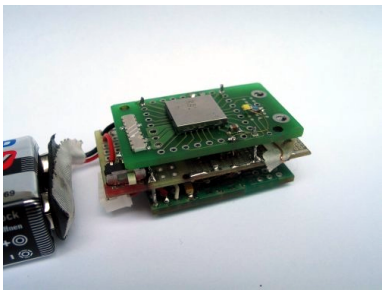
Projektdokumentation
Haptical Interfaces II

Bauhaus-Universität Weimar



Robert Scheibe, Robert Lenhardt,
Ferry Häntsch, Hendrik Schneider

Inhaltsverzeichnis

I. Einleitung 5

1. Ziele 5

II. Hardware 6

2. Aktoren 6
   2.1. rheologische Flüssigkeiten 6

3. Sensoren 9
   3.1. Accelerometer 9

III. Software 10

4. Implementation der seriellen Kommunikation 10

5. Einlesen der Messwerte 10
   5.1. digital Capturing 11
   5.2. analog Capturing 12

6. VRPN Treiber 13

7. Übertragungstechnik 15
   7.1. Bluetooth und Linux 15
   7.2. Beispielsession Aufbau einer Bluetoothverbindung 16
   7.3. Pythonframework 16
      7.3.1. Funktionsweise 17
      7.3.2. Beispiel: Kugellabyrinth 17
      7.3.3. Beispiel: 3d Desktop Umschalter 18

IV. InertiaMouse 20

8. Zusammenfassung **24**

V. Unterstützung des Projekts „Bücher“ 25

9. Analog Scrolling **25**

10. ReadMe **26**
10.1. Konzept 26
10.2. Umsetzung 28

11. X-BOOK **31**
11.1. Idee und Konzept 31
11.2. Positionen und Funktionen 32
11.3. Funktionsprototyp und Umsetzung 34
11.4. Einlesen der Potentiometerwerte 36
11.5. Unterstützung 36

12. AccelVis – JavaTool **38**
12.1. Implementation 38
12.1.1. JfreeChart 39
12.1.2. Java Communications API (javax.comm) 39
12.1.3. Ansprechen der seriellen Schnittstelle (innerhalb der Klasse AccelVis) 39
12.1.4. Die Protokoll-Implementation (innerhalb der Klasse comm-Listener) 40
12.2. Resümee 41

A. Literatur **43**

Abbildungsverzeichnis

1. Flüssigkeit in entspanntem Zustand... 7
2. Flüssigkeit in gespanntem Zustand... 8
3. Versuche mit Rheologischen Flüssigkeiten 9
4. Erste Versuche mit dem Accelerometer ADXL202 10
5. Zeitlicher Verlauf der Sensorsignale *Sn* und des Interruptsignals *INT* 12
6. Rohe Beschleunigungswerte in einer Bewegungsachse 14
7. Bluetooth Kommunikationsschema 15
8. Das reale Kugellabyrinth 17
9. Das virtuelle Kugellabyrinth 18
10. Der Desktopswitcher 19
11. Schaltung der InertiaMouse 22
12. Das InertiaMouse Gehäuse mit Innenleben 23
13. Der Elektronische Aufbau der InertiaMouse 23
14. Analog Scrolling 26
15. Schema der verwendeten Kugelschalter zur Lagebestimmung . . . 27
16. Vier eindeutige Zustände der Lageschalter 28
17. Platine mit Mikrocontroller und Bluetoothmodul, sowie dem Lesezeichen 29
18. Schema des Python Frameworks 30
19. Drei Achsen-Gelenkkonstruktion 31
20. Aufbau einer Socket-Verbindung in Macromedia Flash 35

Teil I.
Einleitung

Dies ist die Dokumentation des Projekts „Haptical Interfaces II“, welches im Sommersemester 2004 an der Bauhaus Universität Weimar durchgeführt wurde. Teilnehmer waren Robert Lenhardt (Mediensysteme), Ferry Häntsch (Freie Kunst), Hendrik Schneider (Mediensysteme) und Robert Scheibe (Mediensysteme).
Wir gehen speziell auf die Entwicklung der InertiaMouse und die Unterstützung von drei Projekten an der Fakultät Gestaltung ein.

1. Ziele

Die Ziele des Projekts waren breit gefächert. Das Hauptaugenmerk lag wie bereits erwähnt in der Entwicklung eines handlichen, kabellosen Geräts zur Eingabe in virtuellen Umgebungen – die Inertia Mouse. Dabei stützen wir uns auf Erfahrungen aus dem vorangegangenen Projekt Haptische Interfaces I, dem alle Beteiligten angehörten. Hier wurden bereits erste Erfahrungen in der Programmierung von Microcontrollern und der Entwicklung von Eingabegeräten gesammelt und im März 2005 auf der IEEE VR 2005 präsentiert.
Ein weiterer wichtiger Aspekt war die Zusammenarbeit mit der Fakultät Gestaltung. Hier unterstützen wir das Projekt “Bücher” geleitet von Prof. Wolfgang Sattler. Dabei entstanden die Arbeiten “XBook”, “ReadMe” und “AnalogScroll”.
Die bei diesem Projekt entwickelte InertiaMouse wurde außerdem im Rahmen des 3. Innovations- und Gründertags der Bauhaus-Universität Weimar 2004 präsentiert.

Teil II.
Hardware

2. Aktoren

Zu Beginn des Projekts beschäftigten wir und mit den Proceedings des „Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems“[2]. Experimente haben wir mit rheologischen Flüssigkeiten [10] und Accelerometern ausgeführt.

2.1. rheologische Flüssigkeiten

Rheologische Flüssigkeiten (RF) sind Dispersionen, d.h. Flüssigkeiten, die aus zwei sich nicht mischenden Komponenten bestehen. Bei unseren Experimenten verwendeten wir Leinöl und Maisstärke.
Die Besonderheit der RF besteht darin, daß sich die Viskosität des Materials beim Anlegen eines starken elektrischen Felds verändert. Die folgenden Abbildungen verdeutlichen unsere Experimente. Abbildung 1 auf der nächsten Seite zeigt die Maisstärkelösung in entspanntem Zustand. Sämtliche Maiskügelchen können sich frei in der Flüssigkeit bewegen. In Abbildung 2 auf Seite 8 liegen mehrere 1000 V Spannung an den Kondensatorplatten an. Hierbei bilden die bipolaren Maisstärkekörnchen Kettenformationen aus, welche die Flüssigkeit zähflüssig werden läßt. Der gesamte Vorgang ist sehr schnell. Innerhalb von Millisekunden sind die Teilchen polarisiert und bilden Ketten aus.
In Versuchsaufbau betrug der Abstand zwischen den Kondensatorplatten 1 cm, bei einer Spannung von 5 kV.

Es kam zu einer signifikaten Verfestigung der Flüssigkeit. Allerdings reichte der Effekt nicht aus, um einen Ansatz für ein taktiles Display zu entwickeln. Ausserdem bedeuten die sehr hohen Spannungen ein veritables Sicherheitsproblem [7]. Ausserdem gibt es Probleme mit der Langzeitstabilität der Flüssigkeiten. Schreitet die Sedimentation zu weit voran geht der Effekt verloren [7]. Die Beherrschung des Effekts ist für ein Handheldgerät, wie es in diesem Projekt erstellt werden sollte nicht realisierbar.

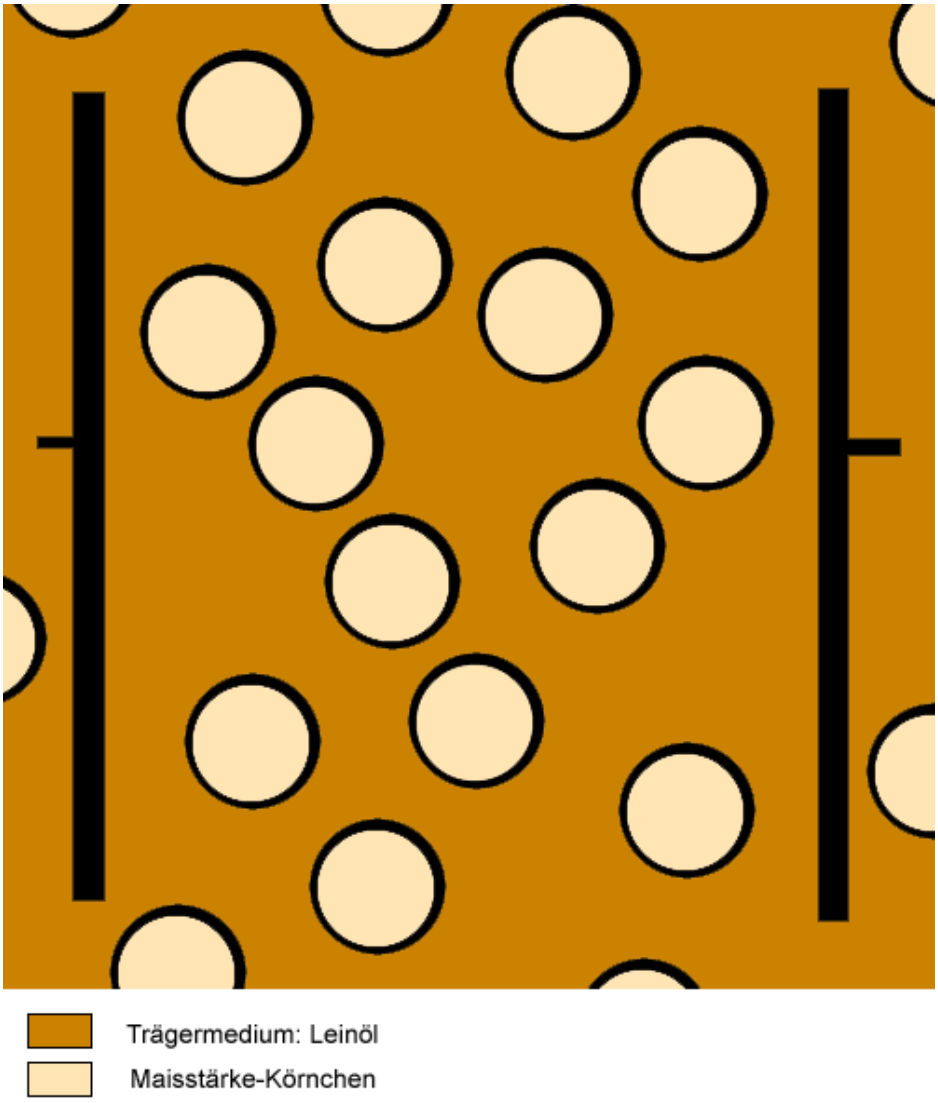


Abbildung 1: Flüssigkeit in entspanntem Zustand…

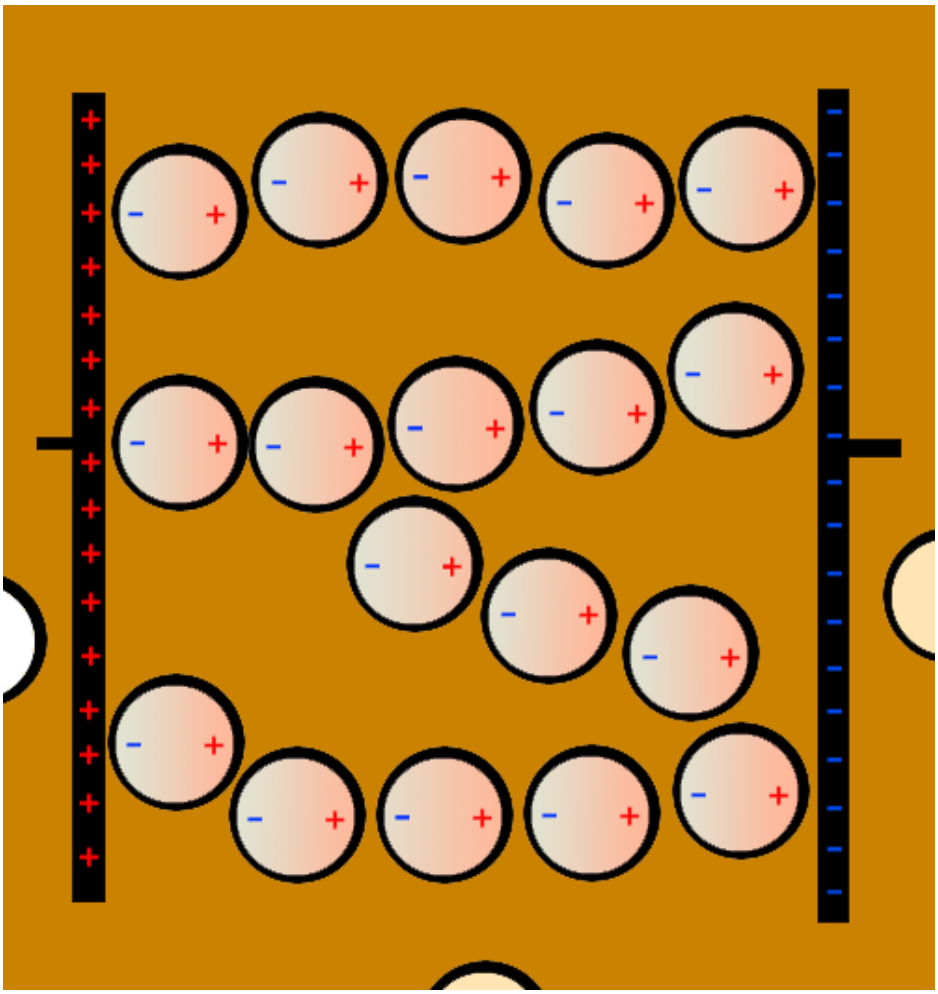


Abbildung 2: Flüssigkeit in gespanntem Zustand…

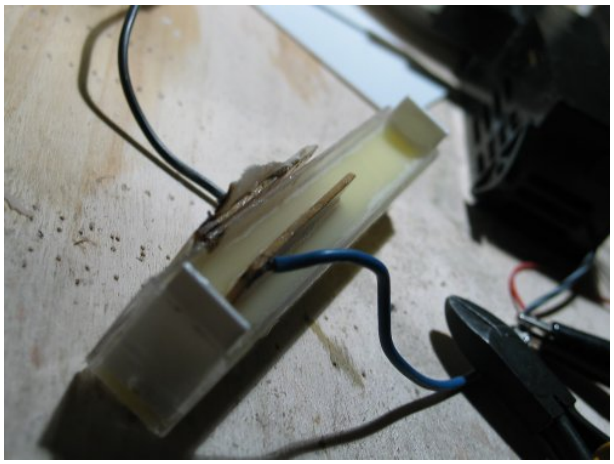


Abbildung 3: Versuche mit Rheologischen Flüssigkeiten

3. Sensoren

3.1. Accelerometer

Der ADXL202 [4] von Analog Devices ist ein 2-Achsen Beschleunigungssensor (Accelerometer) der Beschleunigungskräfte im Bereich von +/-2g misst und die Werte zum einen analog, in Form einer der Kraft äquivalenten Spannung, zum anderen digital, als pulsweitenmoduliertes Rechtecksignal (PWM) ausgibt. Die Frequenz dieser PWM kann mit der Wahl des Widerstandes an Pin “T2” variiert werden. Der Betriebsspannungsbereich ist mit 3-5,25V angegeben, bei einem Strom von nur 0,6-1mA, gute Voraussetzungen für den Einsatz in batteriebetriebenen Geräten.
Der eigentliche Sensor innerhalb des Schaltkreises ist ein Kondensator mit einer starren und einer federnd gelagerten Platte. Durch Beschleunigung bewegt sich letztere, wodurch sich der Abstand zwischen den Kondensatorplatten und damit seine Kapazität ändert. Diese Kapazitätsänderung moduliert dann die Amplitude einer Gleichspannung und diese wiederum den Duty Cycle (das Tastverhältnis) der Rechteckspannung für den PWM-Ausgang.

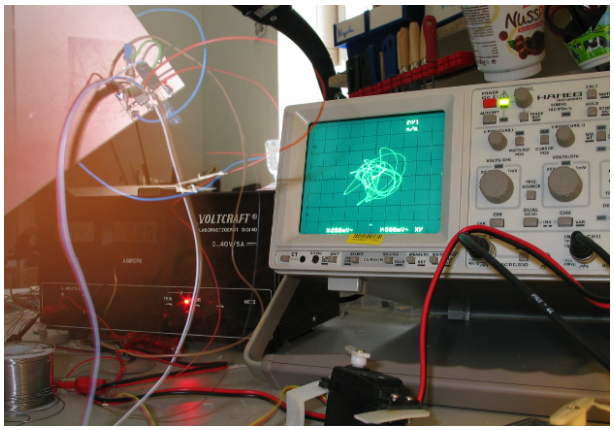


Abbildung 4: Erste Versuche mit dem Accelerometer ADXL202

Teil III.
Software

4. Implementation der seriellen Kommunikation

Um mit externen Rechnern zu kommunizieren, wurde eine Bibliothek zum Versenden und Empfangen von Daten über das UART des Atmel in C++ implementiert. Hierbei wurden Ideen, die in [1] beschrieben sind, umgesetzt, z.B. eine Abstraktion von Maschinenregistern, die es ermöglicht, schnell auf andere Architekturen zu portieren.

5. Einlesen der Messwerte

Der ADXL202 besitzt zwei Ausgangsarten – die Beschleunigungswerte werden digital als pulsweitenmodulierte Signale oder als analoge Spannung proportional zur Beschleunigung ausgegeben. Auf der Mikroprozessorseite kann man die Timer benutzen, um das digitale Signal einzulesen, oder die D/A Wandler, um die Spannungswerte zu digitalisieren. Beide Methoden haben Vor- und Nachteile im

Hinblick auf die Genauigkeit der Werte und die (Hardware- und Software) Komplexität des Einlesevorgangs. Darauf soll nun näher eingegangen werden.

5.1. digital Capturing

Der Atmel verfügt über zwei Timer/Counter mit 8bit Breite, und einen Timer mit 16bit Breite. Diese Timer können externe Ereignisse (Signaländerungen, Flanken) zum Prozessortakt in Beziehung setzen. Es handelt sich bei ihnen im Prinzip um ein Register, das automatisch bei jedem Taktzyklus (oder Vielfachen davon) inkrementiert wird. Bei Überlauf oder bei externen Ereignissen kann ein Interrupt ausgelöst werden, und benutzerdefinierter Code ausgeführt werden, um z.B. den Wert des Registers zu speichern, o.ä.. Um ein pulsweitenmoduliertes Signal (PWM-Signal) einzulesen, kann man das Signal an einen Interrupt-Pin anschließen, bei Änderung des Signals auf logisch 1 den Wert eines Timers in T_{Rise} speichern, bei den nächsten beiden Änderungen den Wert des Timers in T_{Fall} und T_{End} speichern, die Pulsweite des PWM-Signals PW_S ergibt sich dann aus

$$PW_S = T_{Fall} - T_{Rise} / T_{End} - T_{Rise}$$

Mittels diesem Verfahren kann man PWM-Signale mit einer Periode, die so groß ist wie Überlaufzeit des Timers verlustlos einlesen; die Genauigkeit hängt von der Länge der Periode ab, und wird am größten, wenn die Periode gleich der Überlaufzeit des Timers ist.

Die Periode $T_{End} - T_{Rise}$ der Signale der beiden Kanäle eines ADXL202 sind gleich lang, und ändert sich nur langsam aufgrund von Temperaturschwankungen, daher muss man die Messung dieser Werte nicht in jeder Periode des Signals vornehmen.

Das Einlesen wird schwieriger, wenn man zwei ADXL202 mit insgesamt vier Kanälen benutzen möchte, da nur zwei Interrupt-Pins zum Einlesen externer Werte verfügbar sind. Durch ein Multiplexen der Signale der vier Kanäle auf einen Interrupt-Pin, wobei jede Signaländerung auf einem der Kanäle einen Deltapuls am Interrupt-Pin auslöst, ist es dennoch möglich, die Werte einzulesen. Die Abbildung 5 auf der nächsten Seite veranschaulicht die jeweiligen Signalformen der insgesamt 4 Kanäle der Sensoren und des Interruptsignals. Bei einem Interrupt muss man nun prüfen, welches Signal seinen Wert geändert hat, und den entsprechenden Wert für den Kanal speichern.

Da das Interruptsignal mit externer Hardware (XOR Schaltung) aus den Kanälen der Sensoren erzeugt wird, kann es passieren, das Signaländerungen, die genau

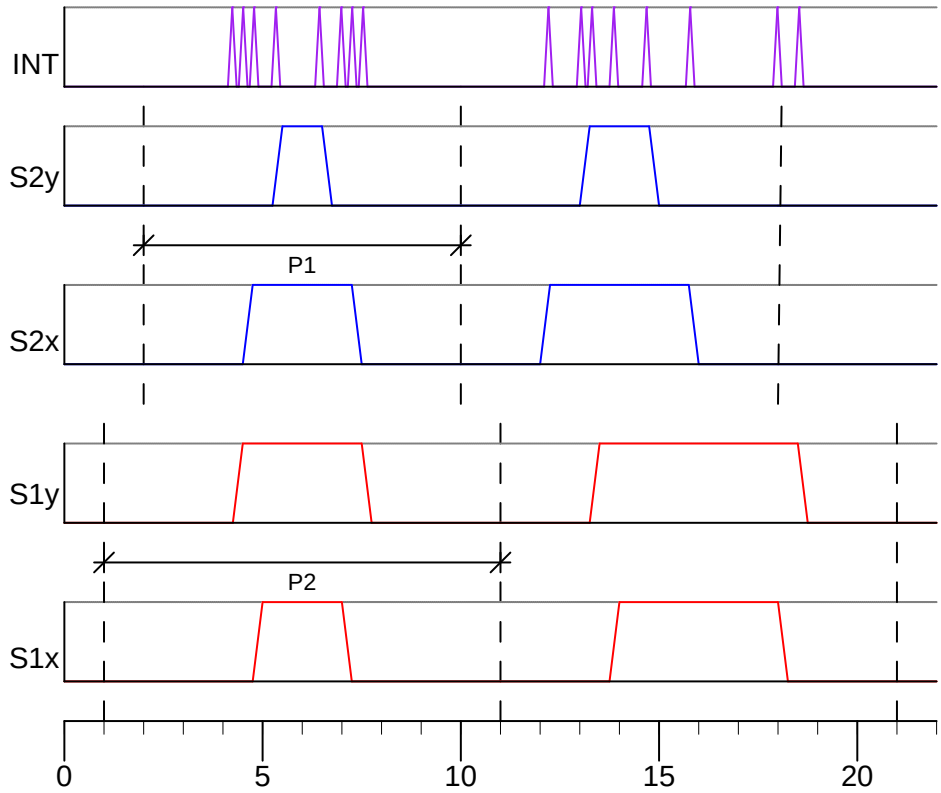


Abbildung 5: Zeitlicher Verlauf der Sensorsignale *Sn* und des Interruptsignals *INT*

gleichzeitig passieren, keinen Interrupt auslösen. Dies kann vom Einlesealgorithmus nicht erkannt werden. Daher müssen die Werte gemittelt werden. Dies senkt die Genauigkeit der Werte und die Latenzzeit steigt.

Da das Signal auf dem ganzen Signalweg digital vorliegt, ist es störunanfälliger. Die Signalverarbeitung wie Interpolation oder das Herausfiltern vom Gleichanteil des Erdschwerefelds muss aber auf dem Atmel in Software umgesetzt werden, der durch die gleichzeitige Kommunikation und die Einlesevorgänge belastet ist.

5.2. analog Capturing

Das Einlesen der Analogwerte über A/D Wandler wird vom Atmel durch 8 A/D Kanäle unterstützt, die seriell genutzt werden können, d.h. man kann ein Signal wandeln, und dann auf ein Anderes umschalten und Dieses wandlen. Das auf-

wendige Multiplexen wie beim Einlesen der digitalen Werte kann entfallen, da die Analogwerte zu jeder Zeit gesamplet werden können. Ebenso kann die Filterung durch Hardware erfolgen.

Die Genauigkeit wird durch die Auflösung der A/D Wandler bestimmt. Da bei der Filterung des Erdschwerefelds das Signal auch negative Spannungswerte aufweisen kann, die A/D Wandler aber nur positive Werte wandeln können, muss dem Signal durch Hardware ein Gleichspannungsanteil zugefügt werden. Dieser Gleichspannungsanteil schwankt, und kann daher nicht ohne weiteres aus dem Signal herausgerechnet werden. Softwareseitige Filterung beeinträchtigt das Signal weiter. Durch Wandlung des Gleichanteils könnte man den Wert direkt messen, was aber zusätzliche A/D Kanäle fordern würde.

Wir haben das analoge Einlesen umgesetzt, was noch durch obige Anmerkungen verbessert werden könnte.

6. VRPN Treiber

Das VRPN (Virtual Reality Peripheral Network) Framework ist ein geräteunabhängiges und netzwerktransparentes Interface für periphäre VR Ein- und Ausgabegeräte. Für nähere Informationen siehe [11]. Um unser Gerät mit diesem Framework nutzbar zu machen, wurde ein Modul implementiert, was auf der VRPN-Implementierung des Treibers für die Spacemouse aufsetzt. Es enthält Funktionalität, um die Kommunikation über die serielle Schnittstelle und die Transformation der Beschleunigungswerte in Geschwindigkeits- und Positionsdaten zu ermöglichen. Die Berechnung von Geschwindigkeit und Position aus der Beschleunigung erfolgt aufgrund der Proportionalität von Beschleunigung und der Ableitung von Geschwindigkeit bzw. der zweifachen Ableitung des zurückgelegten Weges. Das heißt, man muß das Beschleunigungssignal integrieren (einmal für die Geschwindigkeit, zweimal für den Weg). Sei a_i die Beschleunigung zur Zeit t_i und a_i konstant zwischen $\Delta t = t_{i+1} - t$. Dann ergibt sich die Geschwindigkeit v_i aus

$$v_i = v_{i-1} + a_i \cdot \Delta t$$

und der zurückgelegte Weg s_i aus

$$s_i = s_{i-1} + a_i \cdot \Delta t^2$$

Der Wert für die Geschwindigkeit und die Wegänderung v_i und s_i wird gespeichert und für die Berechnung des nächsten Schrittes verwendet. Diese Berechnung lässt

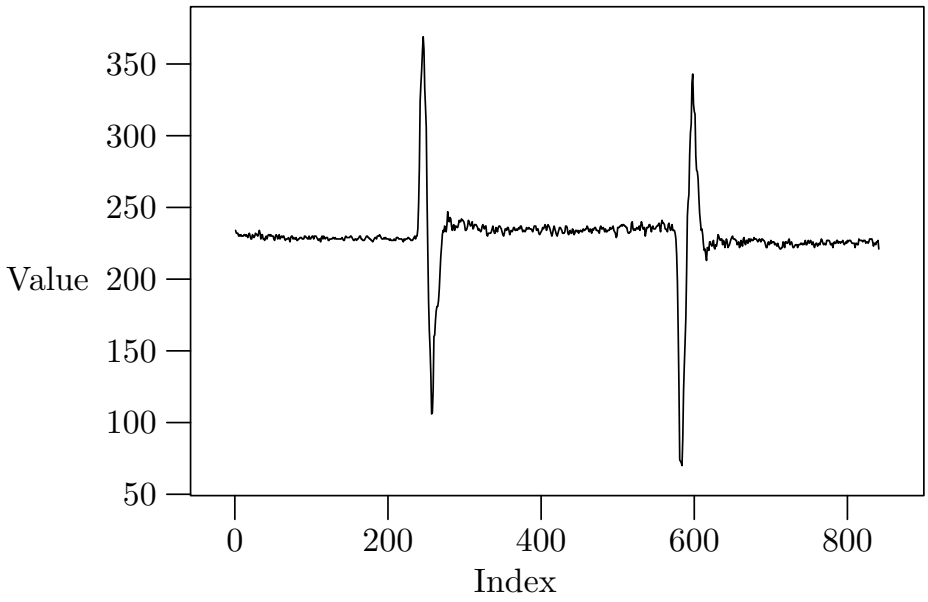


Abbildung 6: Rohe Beschleunigungswerte in einer Bewegungsachse

sich auch als digitaler IIR (Infinite Impulse Response) Filter auffassen, der auf das Beschleunigungssignal angewendet wird[6]. Die Differenzengleichung eines kausalen digitalen Filters lautet:

$$y(n) = \sum_{i=0}^{M} b_i \cdot x(n-i) - \sum_{j=1}^{N} a_j \cdot y(n-j)$$

Der Filter, der unseren Berechnungen entspricht, hat die Koeffizienten $a_1 = -1$, $b_0 = \Delta t$. Es handelt sich um einen rekursiven Filter 1. Ordnung. Da der Term a_1 nicht im Intervall $(-1,1)$ liegt, handelt es sich um einen Filter, der sich aufschaukelt, und beliebig große Werte annehmen kann. Daher wirken sich schon leichte Fehler im Signal ggf. stark aus.

Zwei Fehlerquellen treten im Signal (typischerwiese wie in Abbildung 6) auf: Es gibt einen Gleichanteil und zwei entgegengesetzte Beschleunigungen führen nicht zu einem Wegfall der Geschwindigkeit. Der Gleichanteil kann durch Addierung einer Konstante und anschließende Hochpassfilterung entfernt werden. Dies wirkt sich aber auf die weitere Signalverarbeitung aus. Um sinnvolle Geschwindigkeitswerte zu erhalten, müssen wir bei niedrigen Signalwerten stark

hochpassfiltern, bei hohen Werten weniger. Die genaue Kennlinie ist für jedes Signal anders, und nur bei Beschleunigungen mit einer bestimmten Stärke anzuwenden. Die Ergebnisse, die wir erzielt haben, zeigen, das weitere Arbeit nötig ist, um auch bei geringen Beschleunigungen sinnvolle Werte zu bekommen.

7. Übertragungstechnik

7.1. Bluetooth und Linux

Um eine drahtlose Kommunikation zwischen dem Gerät und der VR Anwendung zu ermöglichen, haben wir einen Bluetoothchip von LinTech [1] genutzt.
Dieses Gerät ermöglicht die serielle UART Datenkommunikation wodurch es direkt mit den Microcontroller verbunden werden kann.
Rechnerseitig wurden verschiedene USB Bluetooth-Dongle eingesetzt. Wir verwenden debian sid mit den bluez-utils 2.9-4. Das Packet bluez-hcidump ist ebenfalls von großem Nutzen, um Kommunikationsfehler zu erkennen.

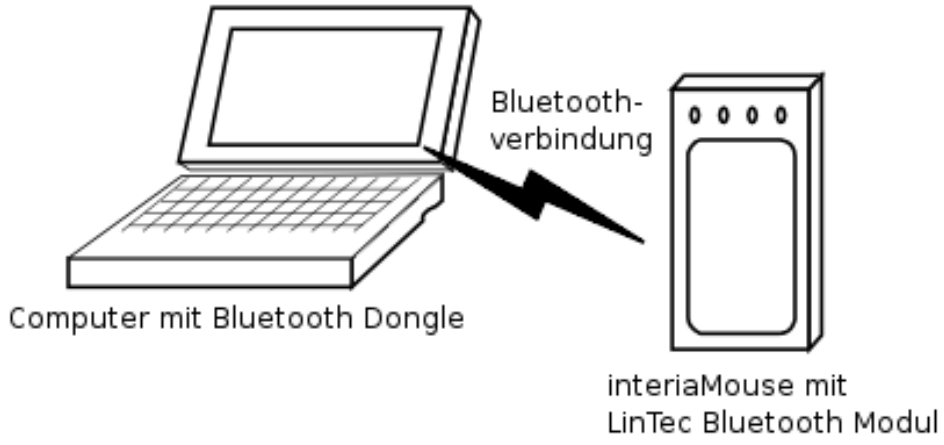


Abbildung 7: Bluetooth Kommunikationsschema

[1] http://www.lintech.de/ [zuletzt besucht: März 2006]

7.2. Beispielsession Aufbau einer Bluetoothverbindung

Beispielsession zur Konnektierung zweier Bluetoothadapter

```
#> rfcomm scan
scanning...
  FE:F9:ED:00:86:E0 LinTec Device
#> rfcomm bind 0 FE:F9:ED:00:86:E0
```

Mit diesen Kommandos wird zunächst nach sämtlichen Bluetoothgeräten in der Umgebung gesucht (je nach Bluetooth-Spezifikation bis zu 100m, i.A. aber nur bis ca. 10m Entfernung). Danach wird das gefundene Bluetoothgerät anhand seiner Mac-Adresse an das Device */dev/rfcomm0* gebunden.

Bevor jedoch auf das Gerät zugegriffen werden kann, muss eine PIN eingegeben werden. Diese PIN verhindert unerlaubten Zugriff auf Bluetoothgeräte. Dazu öffnet sich ein Fenster, welches zur Eingabe der PIN auffordert. Erst danach steht das Gerät zur Verfügung. Die PIN ist beim verwendeten LinTec Chip standartmäßig auf 1234 gesetzt, kann aber mit weiteren Grundeinstellungen mittels einer Terminalverbindung auf Kanal 2 verändert werden. Dazu konnektiert man Kanal 2 auf /dev/rfcomm0.

```
#> rfcomm bind 0 FE:F9:ED:00:86:E0 2
```

Danach kann man beispielsweise mit gtkterm auf /dev/rfcomm0 zugreifen und im Terminal Änderungen vornehmen.

7.3. Pythonframework

Um verschiedene Beispielanwendungen erstellen zu können, wurde ein Pythonframework entwickelt, welches sich einfach an die verschiedenen Anforderungen anpassen läßt.
Das Framework hat folgende Features:

- Konfiguration und Auslesen der seriellen Schnittstelle, welche vom BlueZ [2], dem offiziellen Linux Bluetoothstack, bereitgestellt wird
- Kommunikation mittels seriellen Protokolls
- Visualisierung

[2] http://www.bluez.org/ [zuletzt besucht: März 2006]

- Kommunikation mit anderen Programmen über *xte*, bzw. Erzeugen von X Kommandos

Durch die Nutzung von Python ist eine sehr kompakte Implementierung aller Teile möglich.

7.3.1. Funktionsweise

Das Pythonframework besteht aus folgenden Einzelteilen:

- pyODE [8], cgkit [3]
- pySerial [9]
- python-pygame, python-opengl (Debian Packete)

7.3.2. Beispiel: Kugellabyrinth

Dies ist eine Beispielanwendung, welche sowohl eine reale Anwendung ist, als auch eine virtuelle Entprechung hat. Es ist ein zweiachsig drehbar gelagertes Labyrinth.

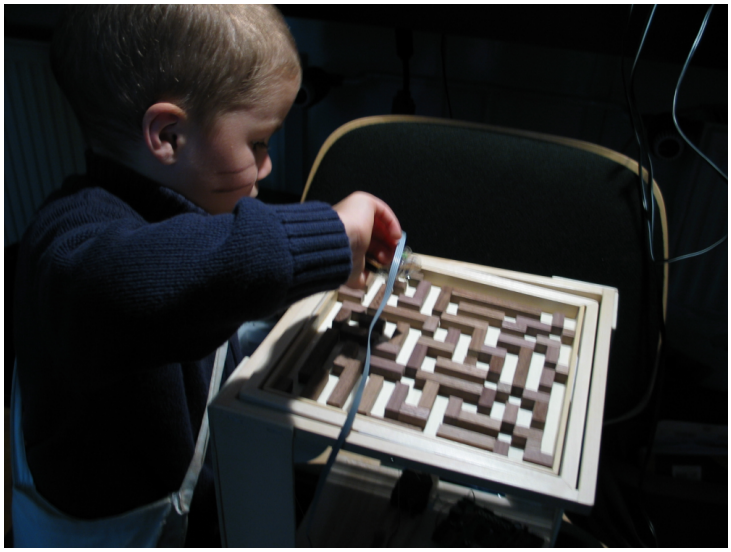


Abbildung 8: Das reale Kugellabyrinth

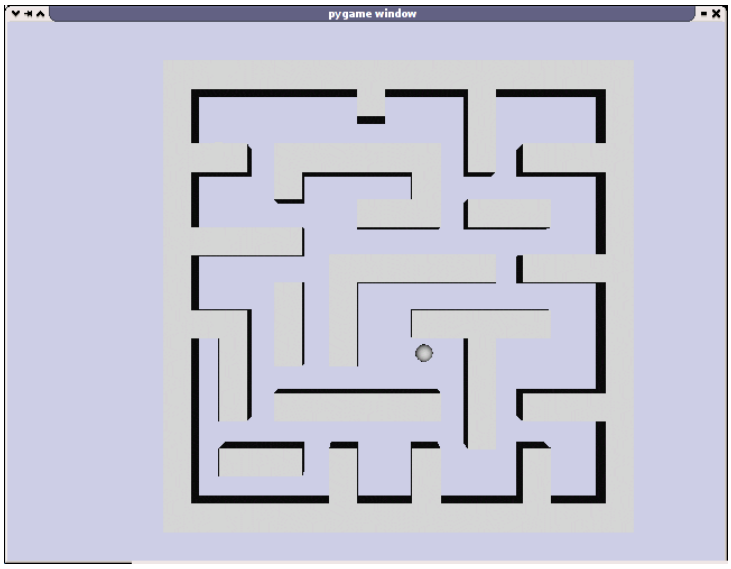


Abbildung 9: Das virtuelle Kugellabyrinth

Ziel des Spiels ist es, eine kleine Kugel quer durch das Labyrinth auf die andere Seite zu bewegen.
Das reale Labyrinth wird über eine Kabelfernsteuerung bedient. Um die Servomotoren, welche das Labyrinth bewegen, anzusteuern wurde wie bei den anderen Anwendungen ein 2-achsiger Accelerometer vom Typ ADXL202 [4] verwendet. Der PWM-Output der Beschleunigungswerte aus dem Accelerometer wird dazu benutzt die PWM Ansteuerung der Servomotoren zu realisieren. Dabei ist der Sensor so präzise, das sogar das Handzittern auf das Labyrinth übertragen wird.
Die digitale Entsprechung des Spiels wird über die InertiaMouse gesteuert. Durch kippen des Geräts in die entsprechende Richtung wird die Kugel durchs Labyrinth gesteuert.

7.3.3. Beispiel: 3d Desktop Umschalter

3ddesk [3] ist ein OpenGL Programm zum Umschalten der virtuellen Desktops bei Linux Desktop Umgebungen wie KDE oder Gnome. Es wird benutzt, um

[3] http://desk3d.sourceforge.net/

Abbildung 10: Der Desktopswitcher

ein dreidimensionales Wechseln von virtuellen Desktops zu ermöglichen. Dabei werden Screenshots der virtuellen Desktops wie auf 10 angeordnet.
Durch Kippen der InertiaMouse wird das Rotieren durch die virtuellen Desktops gesteuert. Auch hier wurde wieder das Pythonframework eingesetzt.

Die Kommandos für den Desktopswitcher lauten:

```
--schnipp--
if (channel[0] < 200 and channel[4]!=0):
  print "links", channel[4]
  a = os.system("xte 'keydown Control_L' 'keydown Alt_L' 'key s'
       'keyup Control_L' 'keyup Alt_L'")
  a = os.system("xte 'key Left'")

if (channel[0] > 300 and channel[4]!=0):
  print "rechts", channel[4]
  a = os.system("xte 'keydown Control_L' 'keydown Alt_L' 'key s'
       'keyup Control_L' 'keyup Alt_L'")
  a = os.system("xte 'key Right' ")
--schnapp--
```

Wobei im channel array die Meßwerte des Accelerometers gespeichert sind. Das Wechseln der Desktops gelingt sehr gut und intuitiv. Das Gerät ist nach kurzer Eingewöhnungszeit leicht zu handhaben.

Teil IV.
InertiaMouse

Als Microcontroller (μC) wählten wir einen Atmel ATega32, welcher uns bereits aus vorigen Projekten bekannt war. Die Anbindung des Accelerometers kann hier auf analoge oder digitale weise erfolgen. Beide Varianten haben sowohl Vor- , als auch Nachteile.

Die digitalen PWM-Signale können zwar mehr oder weniger direkt in den μC geführt werden und sind gegenüber analog-Signalen wesetlich unanfälliger für Störungen und Temperaturänderungen, aber die eigentlichen Messwerte müssen erst im μC mit Hilfe von Zählern ermittelt werden. Dafür wird zudem eine recht aufwändige Interrupt-Generierung benötigt. Bei unserer ersten Testschaltung orientierten wir uns am “BlueWand” [5], einem 2D-Eigabegerät-Prototypen, der auch mit dem ADXL202 arbeitet.

Eine Logikschaltung erzeugt Nadelimpulse bei jeder Zustandsänderung der beiden PWM-Signale des ADXL. Diese werden dann als Interrupt genutzt um den Zeitpunkt zu definieren, an dem der μC die PWM-Signale (x,y) einlesen soll und nun im Vergleich mit dem aktuellen Wert eines intern laufenden Zählers den Wert des Dutycycles zu ermitteln. Mit einem ADXL funktioniert das ganz gut, da die beiden PWM-Ausgänge synchron und phasengleich sind.

Beim Einbinden eines zweiten Accelerometers kommt es zu Problemen, weil dessen Ausgänge sich nicht mit denen des Ersten synchronisieren lassen, was zur Folge hat, daß die Interrupt-Impulse beider ADXLs sich unregelmäßig mischen, was eine Zuordnung des Zählerstandes auf die Zustandsänderungen und die damit verbundene Festlegung von Start- und Endpunkten des Dutycycles der einzelnen Kanäle behindert und damit zu Fehlern führt.

Daraufhin entschlossen wir uns, es auf analogem Weg zu probieren. Der Vorteil einer analogen Messwerterfassung liegt darin, dass die Messwerte nicht erst durch eine Zählung im μC ermittelt werden müssen, sondern direkt aus den Registern des Analog/Digital-Wandlers (ADC) des ATmega32 ausgelesen werden können. Eine Interrupt-Steuerung entfällt, da die analoge Spannung ja ständig anliegt und jederzeit abgefragt werden kann. Der Aufwand verschiebt sich in diesem Fall natürlich nur, denn nun ist eine Signalaufbereitung (Verstärkung auf ADC-Pegel) mit Operationsverstärkern (OPV) nötig.

Dadurch ergibt sich jedoch weiterer Vorteil: Durch Hochpässe können die durch

Neigung verursachten Spannungsänderungen herausgefiltert werden, so daß hauptsächlich durch Translation verursachte Beschleunigungen im Signal enthalten sind.

Der Schaltplan 11 zeigt den prinzipiellen Aufbau dieser Variante. X-, Y- und Z- Kanal werden jeweils zwei OPVs zugeführt, einmal mit Hochpass-Filterung, einmal ungefiltert. Die Ausgänge sind mit sechs ADC-Eingängen des ATmega verbunden, wobei die drei gefilterten Signale für Translation und die ungefilterten für Rotation verwendet werden, letzteres allerdings nur in zwei Achsen, da eine Rotation um die vertikale Achse mit einem Accelerometer natürlich nicht zu erfassen ist.

Die Ergebnisse, die wir mit der analogen Variante erzielten, sind jedoch nicht in dem Maße zufriedenstellend, wie wir uns erhofft hatten, was vermutlich auf die nicht allzu ausgereifte Signalaufbereitung zurückzuführen ist. Der Aufbau in SMD-Technik auf einer Rasterplatine machte die Schaltung nicht besoners sicher gegen Störungen und die Wahl der Operationsverstärker war für ein Batteriebetriebenes Gerät auch nicht die allerbeste. Energiesparende “Rail to Rail” OPVs wären den Standard TL081 sicher vorzuziehen gewesen.

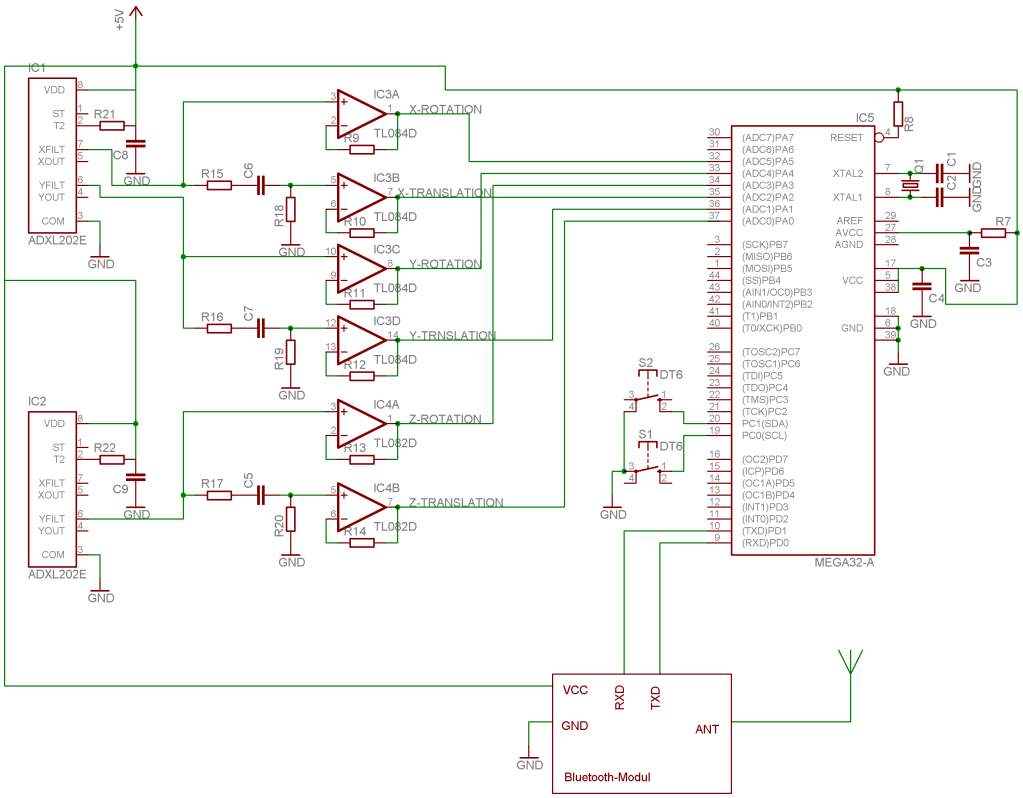


Abbildung 11: Schaltung der InertiaMouse

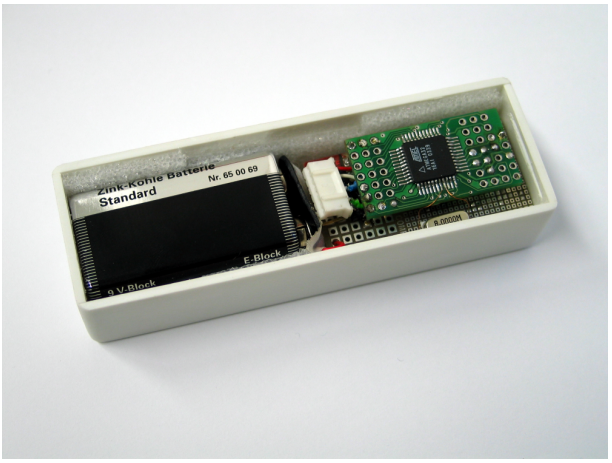


Abbildung 12: Das InertiaMouse Gehäuse mit Innenleben

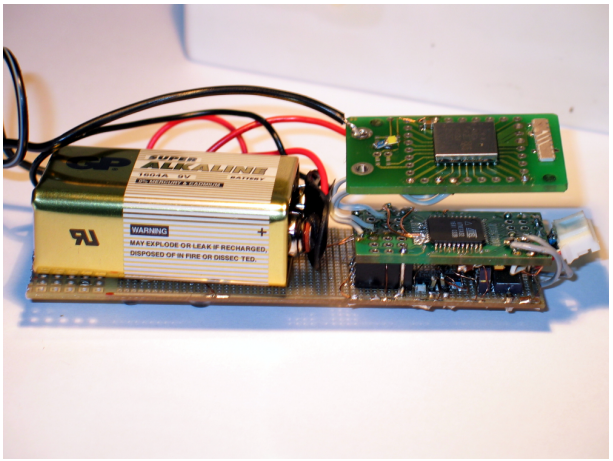


Abbildung 13: Der Elektronische Aufbau der InertiaMouse

8. Zusammenfassung

Wir haben einen Entwurf eines kabellosen Eingabegeräts entwickelt, welches auf Accelerometern als Sensoren aufbaut und zwei Rotations- und drei Translationsfreiheitgrade besitzt. Die Translation wird absolut ausgegeben und beruht auf einer Integration der Beschleunigungswerte der Accelerometer, um die Positionswerte zu erhalten.

Der entstandene Prototyp zeigt eine gute Auflösung der Rotationen, aber die Translationen sind meßfehlerbehaftet und unser Algorithmus zur Integration kann mit diesen Werten keine zufriedenstellende Ergebnisse liefern. Außerdem muss der Stromverbrauch des Geräts weiter gesenkt werden, um wirklich als kabelloses Gerät eingesetzt zu werden.

Teil V.
Unterstützung des Projekts „Bücher“

Aus dem Vorlesungsverzeichnis: „Der Inhalt eines Mediums ist stets ein anderes Medium (H. M. Mc L.). Tablet-PCs und E-Books sind »tangible books« und inzwischen weithin verfügbar. Das Lesen eines Buches oder auch eines umfangreichen Dokuments auf einem solchen Rechner hat bezüglich des Interfaces bisher wenig Ähnlichkeit mit dem Lesen eines realen Buches. Bei einem realen Buch kann jederzeit eine beliebige Seite aufschlagen, vorwärts und rückwärts blättern. Diese Operationen werden für ein virtuelles Buch meist nur durch einen »scrollbar« realisiert. Dieser erlaubt weder einen direkten Zugriff noch komfortables Blättern. In diesem Projekt sollen neue Eingabegeräte entstehen. Ziel ist die Entwicklung von sinnlichen Benutzungsebenen für digitale Bücher und deren gestalterische Umsetzung.
Mit den Mediensystemern geht es um die Programmierung von Mikrocontrollern zur Ansteuerung der Eingabegeräte. Dieses Projekt findet gemeinsam mit Studenten der Fakultät Medien statt.“
Im Verlaufe des Semesters haben wir 3 Projekte aktiv unterstützt. Dabei entwickelte sich eine enge und produktive Zusammenarbeit mit den Studenten der Fakultät Gestaltung.

9. Analog Scrolling

Michael / Ferry Häntsch

Im Rahmen des Projektes “Bücher?” der Fakultät Gestaltung baute ich eine Schrittmotorsteuerung für den Entwurf von Michael Grunwald. Dabei handelte es sich um ein “Gerät zur Anzeige langer Texte ohne Seitenstruktur”. Ein Bedien· inteface erlaubt das scrollen in verschiedenen Geschwindigkeiten und einen Zeilen- und Dreizeilen-, sowie Seitenweisen Vorschub.

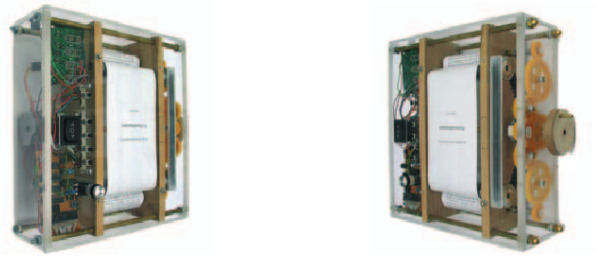


Abbildung 14: Analog Scrolling
Bild von Michael Grunwald

10. ReadMe

Jessica Matussek / Ferry Häntsch / Robert Scheibe

10.1. Konzept

ReadMe ist ein Navigationskonzept zum Lesen an Geräten mit kleinem Monitor. Der Entwurf stammt von Jessica Matussek.
Im Rahmen der Zusammenarbeit wurde über neue Funktionen für das vom „analogen“ Buch bekannte Lesezeichen gesucht. Durch die Nutzung einer bereits vorhanden Metapher, und deren Überladung mit neuen Interaktionsmöglichkeiten erübrigt sich die Installation neuer, unbekannter Steuerelemente, die den Nutzer zunächst vielleicht abschrecken könnten, das Buch zu nutzen.
Der Mehrwert eines elektronischen Buches besteht eben darin, bei gleicher Größe mehr als nur ein Buch zugänglich zu machen. Durch die Anordnung der beiden Bildschirme lässt sich z.B. auch in zwei Büchern gleichzeitig lesen.

Angedachte Funktionen für das Lesezeichen waren anfänglich:

- blättern
- scrollen
- kapitelweises Blättern

Außerdem wollten wir noch eine Lagesensorik einbauen, um beim Drehen des Buches auch den Text entsprechend zu rotieren. Dazu wurden Kugelschalter ver-

wendet. Auf der Abbildung 15 sieht man schematisch, wie das funktioniert. Wenn die Kugel, schwerkraftbedingt auf dem Boden liegt, wird der Kontakt geschlossen. Ansonsten ist er geöffnet.

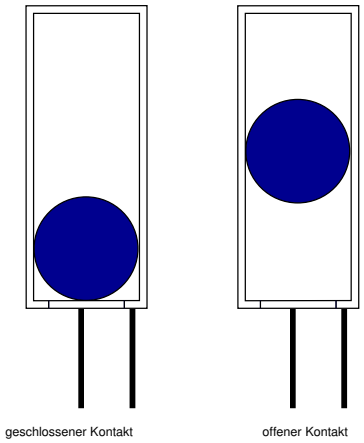


Abbildung 15: Schema der verwendeten Kugelschalter zur Lagebestimmung

Um eine Lagebestimmung zu erreichen werden zwei der Schalter benötigt, die dann folgende Konstellation haben:

Der Winkel zwischen den Schaltern beträgt 45°. Somit lassen sich vier eindeutige Zustände definieren, die dazu genutzt werden können das darzustellende Dokument in 90° Schritten zu rotieren.

Letztlich haben wir uns jedoch darauf beschränkt ein seitenweises Blättern zu implementieren. (Die Rotation wurde nicht fertiggestellt.) Dazu wurde das Lesezeichenband innerhalb des Buches an zwei Tastern vorbeigeführt, die dann beim Auslenken des Bändchens in die entsprechende Richtung ausgelöst wurden. Nach links ziehen bedeutet dann rückwärtsblättern, nach rechts ziehen, vorwärtsblättern.

Da wir im Gegensatz zu Abschnitt 11 auf Seite 31 von Anfang an nicht vorhatten Displays in den Prototypen einzubauen, konnten wir bei der Frage der Größe der Sensorik, Stromversorgung usw. etwas großzügiger sein. Nach Absprache mit Jessica bauten wir eine Platine mit allen Komponenten auf, sodaß sie einen Prototyp mit den gewünschten Größenverhältnissen bauen konnte.

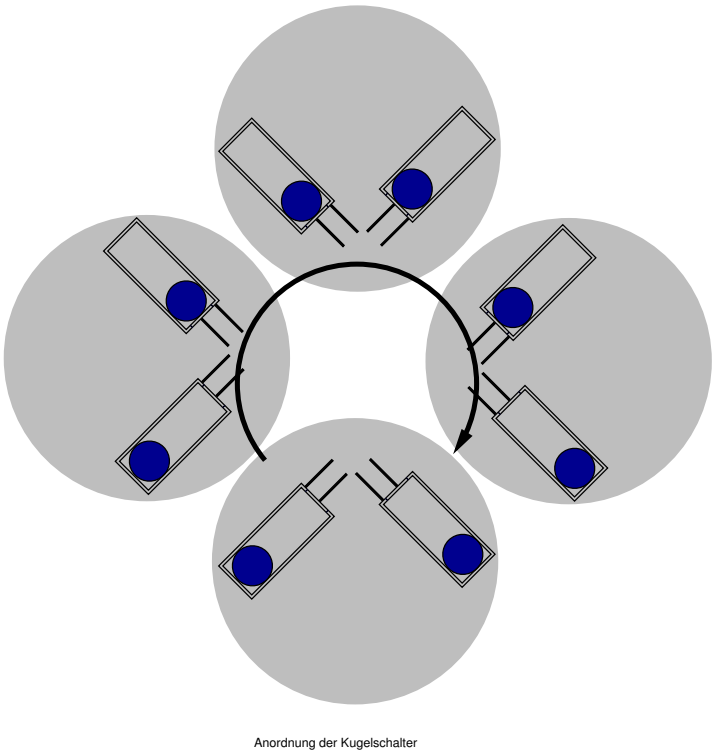


Abbildung 16: Vier eindeutige Zustände der Lageschalter

10.2. Umsetzung

Die Abbildung 17 auf der nächsten Seite zeigt die Platine, welche sich im Inneren des Buches befindet. Zur Umsetzung der Idee wurde das vorhandene Pythonframework (siehe Abbildung 18 auf Seite 30) erweitert, um den Acrobat Reader auf einem Linux Client fernzusteuern.

Das Python Script benutzt *pySerial*, um die serielle Schnittstelle auszulesen. Diese Routine läuft in einem eigenen Thread.

```
1 class MyThread(threading.Thread):
2  def run(self):
3    global erg
4    ser = serial.Serial(port='/dev/rfcomm0', baudrate=9600,
5  stopbits=serial.STOPBITS_ONE,parity=serial.PARITY_NONE,
6  bytesize=serial.EIGHTBITS,xonxoff=0, rtscts=0)
```

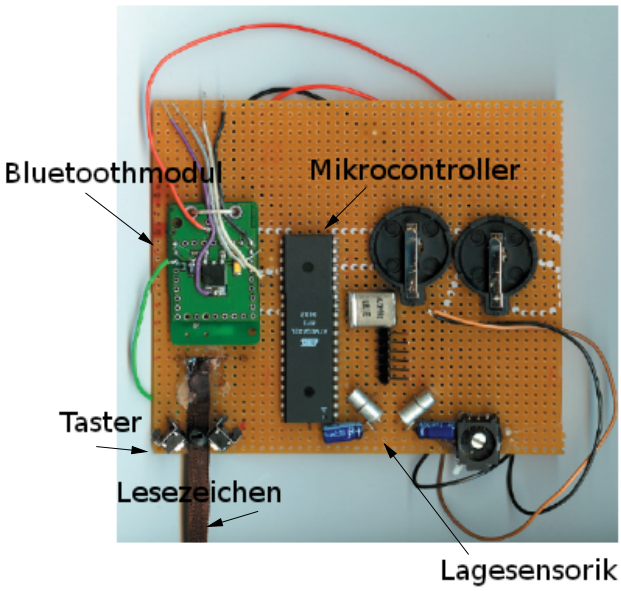


Abbildung 17: Platine mit Mikrocontroller und Bluetoothmodul, sowie dem Lesezeichen

```
7     while running:
8        packet =  SerialProtocol.SerialProtocol(ser.readline())
9         for i in range(1,3):
10         erg[i-1]=packet.getChannel(i)
```

Jedes empfangene Bluetooth- Packet ist nach dem Schema D R L \n zusammengesetzt, wobei D für Databegin, R für rechts/links, L für Lagesensorik, und \n für ein Newline steht, um die Packete zeilenweise von der seriellen Schnittstelle einzulesen. Das Protokoll besteht somit aus 3 Bytes. Wenn R = 1 ist, wird vorgeblättert, ist R = 2 zurück.
Das Blättern im Dokument übernimmt dann das Tool xte [4], mit dem sich beliebige X Events erzeugen lassen. Vorblättern entspricht dem XEvent key „Page_Up“,

[4] debian Packet xautomation

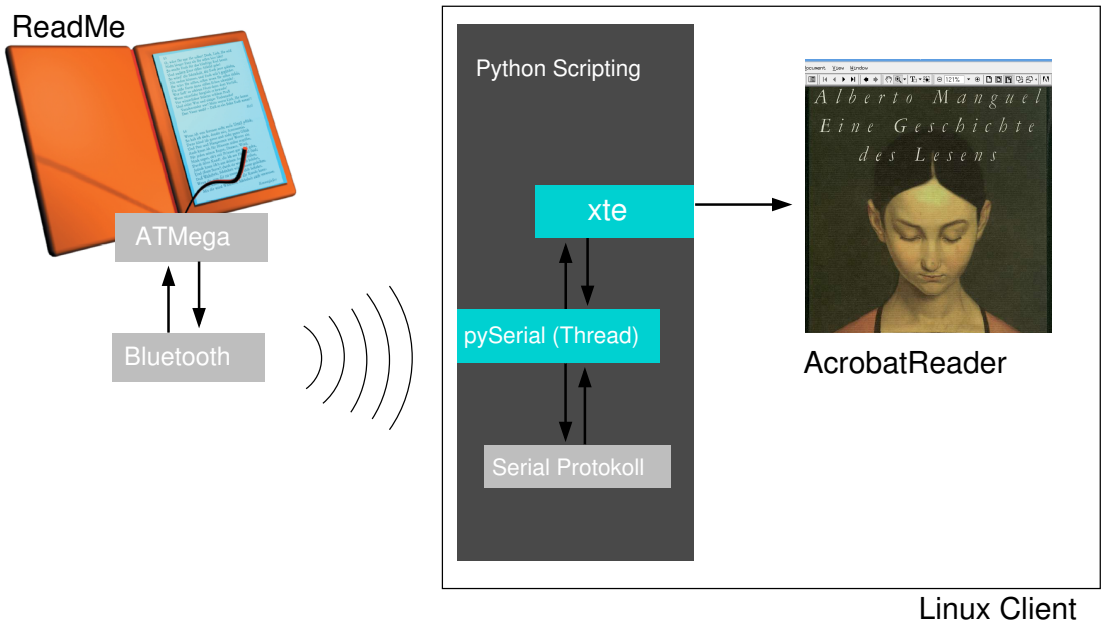


Abbildung 18: Schema des Python Frameworks

zurückblättern „key Page_Down“. Xte wird über einen System-Call aufgerufen.

11. X-BOOK

11.1. Idee und Konzept

Eine Arbeit aus dem Produktdesign-Projekt “Bücher” ist das Konzept X-BOOK von Cheng Liang. Das Ziel hierbei ist es, ein universell einsetzbares Buch zu entwerfen, welches alle Buchformen, wie z.B. Lesebuch, Zeitung, Bildband, Notizbuch, Skizzenbuch unterstützt und diese Anwendungsmöglichkeiten durch moderne Medien wie Fernsehen oder Computer erweitert.

Bei dem Entwurf handelt es sich um einen tragbaren, mobilen Computer, welcher mit zwei Touch Screen Monitoren und einer drei Achsen-Gelenkkonstruktion ausgestattet ist. Dadurch kann man in mehreren Positionen und Anwendungsbereichen arbeiten. Das Konzept sieht vor sämtliche Hardware in der Säule zu integrieren. Zudem soll es ein externes Modul geben, welches als Ladestation fungiert und den Anschluss weiterer Peripherie, z.B. optische Laufwerke, ermöglicht.

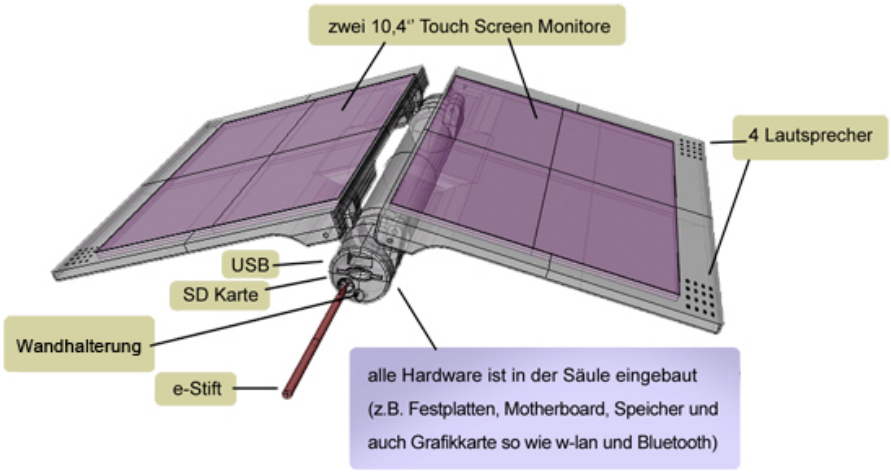
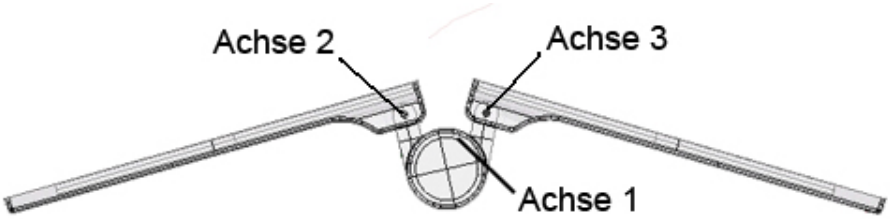


Abbildung 19: Drei Achsen-Gelenkkonstruktion

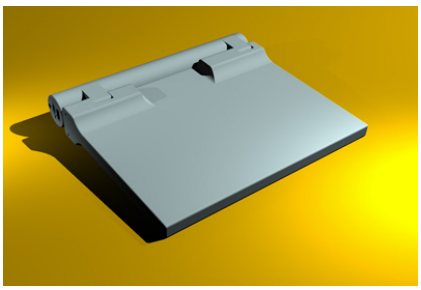
Achse Eins kann in einem Winkel bis zu 300° und Achse Zwei und Drei können in einem Winkel bis zu 70° geöffnet werden. Dadurch hat das X-BOOK mehrere Arbeitspositionen und somit verschiedene Anwendungsbereiche. An jeder Achse ist ein Potentiometer eingebaut, um den jeweiligen Dreh- bzw. Öffnungswinkel an der Achse zu bestimmen und somit die gewünschte Funktion abzuleiten. Die

Steuerungssoftware des X-BOOK erkennt automatisch, welche Arbeitsposition eingestellt ist und schaltet auf Wunsch zum jeweiligen Anwendungsmodus.



11.2. Positionen und Funktionen

1. Schlaf-Modus
Wenn beide Seiten zugeklappt werden, schaltet das X-BOOK in den Schlaf-Modus.



2. Buch-Modus
Wenn nur Achse Eins von 30° bis 150° geöffnet ist, dann schaltet das X-BOOK in den Buch-Modus. Der Nutzer kann das X-BOOK nun wie ein herkömmliches Buch lesen.

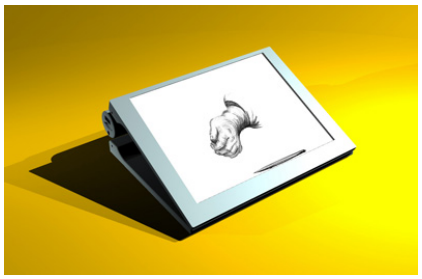
3. Grossbildschirm-Modus
Wenn das X-BOOK geöffnet ist und beide Monitore zu einer großen Fläche zusammen geschoben werden, so schaltet das X-BOOK die beiden Monitore zu einem großen Monitor zusammen. Diese Funktion ermöglicht den Einsatz als Fernseher, das Surfen im Internet oder auch andere Funktionen, wo ein großer Monitor von Vorteil ist.

4. Notebook-Modus
Sind beide Monitore analog eines herkömmlichen Notebooks zueinander positioniert, dann schaltet das X-BOOK in den Notebook-Modus. Der untere Monitor dient nun zur Dateneingabe über eine Touchscreen-Tastatur. Dies ermögliche ein Arbeiten wie mit einem Standard-Notebook. Der untere Monitor bietet zudem die Möglichkeit einige Informationen von dem oberen Monitor, wie z.B. die Webadresse oder die Funktionsknöpfe vom Webbrowser zu übernehmen.

5. Mehrbenutzer-Modus
Entspricht die Bildschirmpositionierung der gezeigten Abbildung, so schaltet das X-BOOK in den Mehrbenutzer-Modus. Hierbei kann das X-BOOK von beiden Seiten aus durch zwei Benutzer bedient werden. Beide Nutzer können gleichzeitig mit unterschiedlichen Anwendungen arbeiten. Die Dateneingabe erfolgt über die beiden Touch Screen Monitore. Der Mehrbenutzer-Modus eignet sich besonders gut für Anwendungen die wenig Tastatureingabe erfordern, wie zum Beispiel das Surfen im Internet, das Lesen unterschiedlicher Online-Zeitungen bzw. E-Papers und für Multi-Player Spiele.

6. Präsentations-Modus
Sind beide Monitore mit Rückseite zueinander angeordnet, so schaltet das X-BOOK in den Präsentationsmodus. Dieser Modus ermöglicht Präsentationen ohne Projektor und eignet sich besonders gut für Vorträge mit wenigen Zuschauern. Hierbei sieht der Vortragende seine Slides auf dem ihm zugewandten Monitor, währenddessen das Publikum das gleiche Bild auf dem gegenüberliegenden Monitor betrachten kann.

7. Skizzen-Modus



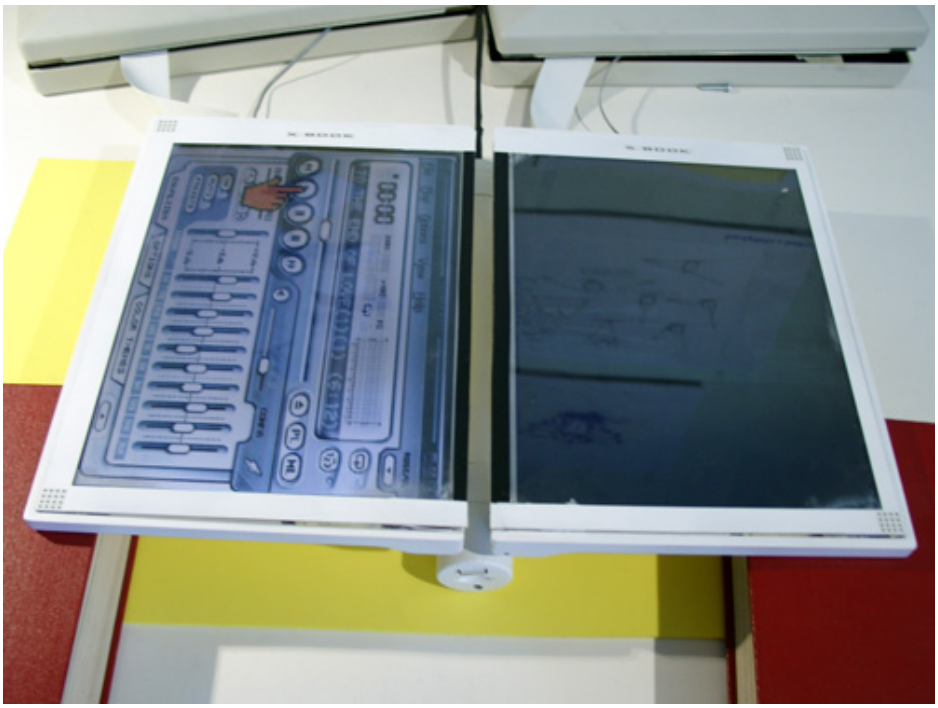
Ist das X-BOOK im maximalen Öffnungswinkel geöffnet, so dass sich die Rücken beider Hälften berühren, dann schaltet das X-BOOK in den Skizzen-Modus. Der Nutzer kann nun mit dem beiliegenden Stift zeichnen oder Notizen machen. Sobald der Stift herausgenommen wird, reagieren die Touch Screen Monitore nur noch auf den Stift. Dies ermöglicht das Auflegen der Hand beim Zeichnen. Im Skizzen-Modus ist allerdings nur einer der beiden Monitore aktiv um Strom zu sparen.

11.3. Funktionsprototyp und Umsetzung

Da sich ein im Detail funktionierender X-BOOK Prototyp mit den zur Verfügung stehenden Mitteln und Zeitaufwand nicht umsetzen ließ, sollte lediglich ein Funktionsprototyp in der von Cheng Liang geplanten optischen Erscheinung entwickelt werden.

Dieser Funktionsprototyp dient lediglich dazu, die bereits beschriebenen Funktionsmodi durch das Abspielen von Flash-Filmen zu simulieren. In dem abgebildeten Aufbau sind zwei 10¨ TFT-Monitore verbaut, die durch einen externen Rechner über eine Grafikkarte mit zwei VGA-Ausgängen angesteuert werden. Desweiteren sind an jeder der drei Achsen jeweils ein Potentiometer verbaut, um je nach Dreh- und Öffnungswinkel, die entsprechende Funktion bzw. den entsprechenden Funktionsmodus abzuleiten. Die drei Potentiometer sind über eine Vrib-Box, welche die analogen Potentiometerdaten in 10-bit Werte digitalisiert, an den PC angeschlossen.

Wie bereits beschrieben, soll der Funktionsprototyp die einzelnen Funktionsmodi durch das Abspielen von Flash-Filmen je nach Modus simulieren. Da Macromedia Flash keine Möglichkeit vorsieht, Daten über eine serielle Schnittstelle bzw. über einen USB-Virtual-ComPort einzulesen, mussten wir die Potentiometerwerte über ein Server-Applikation einlesen und danach an Flash senden. Flash unterstützt zwar keine serielle Kommunikation, wohl aber die Kommunikation über TCP/IP. Die Server-Applikation liest die Potentiometerwerte über das USB-Virtual-ComPort ein und sendet sie über TCP/IP an die Flash-Applikation. In unserem Fall läuft die Server-Applikation auf dem gleichen Rechner wie die Flash-Applikation. Deswegen verwendet die Flash-Applikation die IP-Adresse “local-



host” und das Port 4999 um die Verbindung mit der Server-Applikation herzustellen. Es ist natürlich auch möglich, Server-Applikation und Flash-Applikation auf getrennten Rechnern auszuführen.

```
Flash_Socket = new XMLSocket()

if ( Flash_Socket.connect( "localhost" , 4999 ) == 0 )
{
  trace("Fehler bei Connect.")
  Status_Verbindung="Fehler bei Connect."
}

Flash_Socket.onXML = Nachricht_Empfangen
```

Abbildung 20: Aufbau einer Socket-Verbindung in Macromedia Flash

11.4. Einlesen der Potentiometerwerte

Die Werte der 3 Potentiometer werden in den Variablen “Value0”, “Value1”, “Value2” gespeichert.

```
function Nachricht_Empfangen( doc )
{
  if (doc.firstchild != null)
  {
    var Daten = doc.firstchild;
    Vrib_Typ = Daten.attributes.typ ;
    Vrib_Id = Daten.attributes.id ;
    Vrib_Value = Daten.attributes.value ;

    if (Vrib_Typ == 3 ) Typ = "analog";
    if (Vrib_Typ == 1 )Typ = "digital";

    if (Vrib_Id == 0 ) ID0 = Vrib_Id;
    if (Vrib_Id == 1 ) ID1 = Vrib_Id;
    if (Vrib_Id == 3 ) ID3 = Vrib_Id;

    if (Vrib_Id == 0 ) Value0 = Vrib_Value;
    if (Vrib_Id == 1 ) Value1 = Vrib_Value;
    if (Vrib_Id == 3 ) Value2 = Vrib_Value;

    trace("typ=" + Vrib_Typ + "   id=" + Vrib_Id + "   value=" +
         Vrib_Value);
  }
}
```

Nun kann man über einfache if-Bedingungen die Potentiometerwerte(bereiche) analysieren und den entsprechenden Flash-Film je nach Funktionsmodus über die 10¨ Displays ausgeben.

11.5. Unterstützung

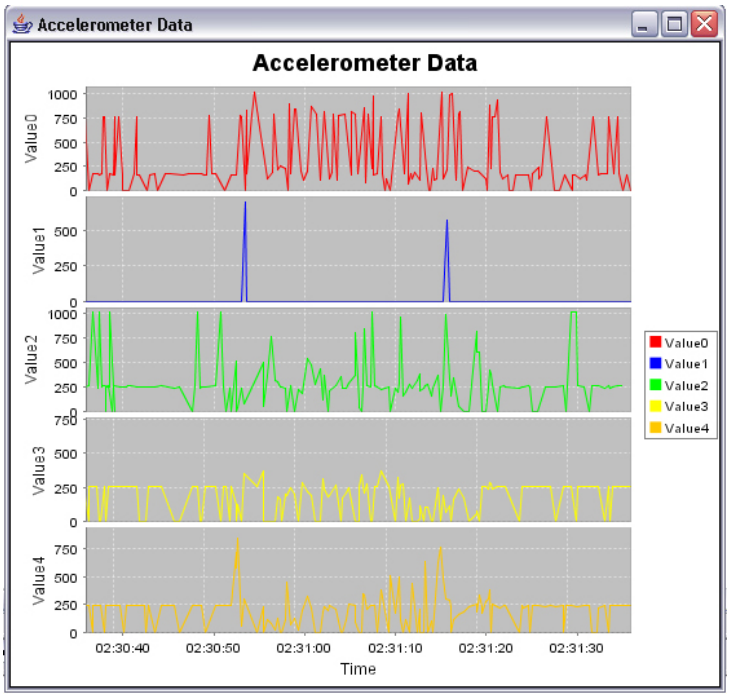
Die Umsetzung des Prototyps für das Konzept X-Book von Cheng Liang wurde durch unser Projekt in folgenden Bereichen unterstützt.:

- Integration der 10¨ TFT-Displays in das X-BOOK-Gehäuse

- Anschluss der Potentiometer an die USB-Vrib-Box
- Implementation der Flash-Applikation zum Abspielen der Flash-Filme entsprechend der Funktionsmodi
- (Verwendung der Server-Applikation von Christian Lessig) Vielen Dank an dieser Stelle an Christian Lessig

12. AccelVis – JavaTool

Dieses Werkzeug wurde mit dem Ziel entwickelt die Accelerometerdaten, welche das Eingabegerät übermittelt, auszuwerten und zu interpretieren. Da wir zunächst die Idee hatten, daß Eingabegerät auch für Gestikenerkennung zu evaluieren, brauchten wir zudem eine Anwendung, die vollzogene Bewegungsmuster anhand der numerischen Accelerometerdaten in Echtzeit graphisch darstellt. Diese Anforderung kann man mit Hilfe von Echtzeit-Linien-Diagrammen (engl. line charts) erfüllen, da hier numerische Daten über die Zeit dargestellt werden müssen.



12.1. Implementation

Die Anwendung wurde in Java unter Verwendung der Zusatzpakete javax.comm (Java Communications API) und JfreeChart implementiert. Beide Pakete sind nicht Bestandteil einer Java SDK Installation.

12.1.1. JfreeChart

Eine recht komfortable Lösung für die genannten Anforderungen bietet JfreeChart, weswegen wir uns auch für eine Java-Implementation entschieden haben. Hierbei handelt es sich um ein freie Java Klassenbibliothek, mit welcher man verschiedenartige Charts aus eigenen Quelldaten generieren kann. Für unser konkretes Szenario verwendeten wir sogenannte “Time Series Charts”. Hier werden die Daten der einzelnen Accelerometer-Achsen jeweils auf den Y-Achsen über die Zeit auf der X-Achse dargestellt.

12.1.2. Java Communications API (javax.comm)

Da es zum Implementationszeitpunkt vom Java-Kernbereich aus keine Möglichkeit gab, die seriellen Schnittstellen (RS-232) anzusprechen und Daten auszulesen bzw. zu schreiben, mußten wir auf eine Lösung des externen Projektes javax.comm zurückgreifen. Dieses Erweiterungspaket war zu diesem Zeitpunkt auch nur für die Plattformen Solaris und Microsoft Windows verfügbar und befand sich noch im Entwicklungsstadium, was sich bei der Umsetzung von AccelVis bestätigte. Mittlerweile ist javax.comm als Java Communications API 3.0 verfügbar und wurde in den Java-Kernbereich integriert. Die Unterstützung der Plattformen Solaris SPARC, Solaris x86 und Linux x86 ist nun ebenfalls gewährleistet.

12.1.3. Ansprechen der seriellen Schnittstelle (innerhalb der Klasse AccelVis)

Erzeugung des CommPortIdentifier-Objekts “portID”:

```
1  portID = CommPortIdentifier.getPortIdentifier("COM4");
```

Wandlung des Objekts “portID” in das Objekt “serss”. Dieses Objekt dient zum lesen oder gegebenenfalls zum schreiben von Daten. Desweiteren wird ein Timeout von 2000 ms angeben.

```
1  serss = (SerialPort) portID.open("AccelVis", 2000);
```

Erzeugung eines Dateneingabe-Streams:

```
1  ins = serss.getInputStream();
```

Erzeugung und Aktivierung eines Eventlisteners für die zu empfangenen Daten:

```
serss.addEventListener(new commListener());
serss.notifyOnDataAvailable(true);
```

Initialisierung der seriellen Schnittstelle:

```
serss.setSerialPortParams(57600,
                          SerialPort.DATABITS_8,
                          SerialPort.STOPBITS_1,
                          SerialPort.PARITY_NONE);
```

12.1.4. Die Protokoll-Implementation (innerhalb der Klasse commListener)

Der folgende Quelltext veranschaulicht die Implementation des Kommunikationsprotokolls mit dem Mikrocontroller.

```
    if (event.getEventType() == SerialPortEvent.DATA_AVAILABLE) {
       byte[] readBuffer = new byte[25];

       try {

         while (ins.available() > 0) {
           ins.read(readBuffer);
         }

         String nachricht = new String(readBuffer);

         int val, i, channel, acc_value;
         int[] channels = new int[7];

         if (nachricht.charAt(0) == 'D' && nachricht.charAt(24) == '\n') {
           for (i = 0; i < 6; i++) {
             //val ist 16 bit Channel Wert
             val = ( (int) (nachricht.charAt(i * 4 + 1)) & 0xf0) << 8;
             val += ( (int) (nachricht.charAt(i * 4 + 2)) & 0xf0) << 4;
             val += ( (int) (nachricht.charAt(i * 4 + 3)) & 0xf0);
             val += ( (int) (nachricht.charAt(i * 4 + 4)) & 0xf0) >> 4;

             //channel wird errechnet
```

```
24              channel = (val >> 10) & 7;
25              //accerlerometerwert
26              acc_value = val & 0x3ff; // 10 bit maske!
27
28              if (channel < 7 && channel > -1) {
29                channels[channel] = acc_value;
30              }
31
32              System.out.println(channels[0] + ", " + channels
                    [1] + ", " +
33                                  channels[2] + ", " + channels
                                        [3] + ", " +
34                                  channels[4] + ", " + channels
                                        [5]);
35
36            }
37
38            Millisecond now = new Millisecond();
39        // Erzeugung der 5 Chart-Datasets fuer die Subplots
40            if (chart != null && channels != null) {
41              if (chart.datasets != null) {
42                chart.datasets[0].getSeries(0).add(now, channels
                      [1]);
43                chart.datasets[1].getSeries(0).add(now, channels
                      [2]);
44                chart.datasets[2].getSeries(0).add(now, channels
                      [3]);
45                chart.datasets[3].getSeries(0).add(now, channels
                      [4]);
46                chart.datasets[4].getSeries(0).add(now, channels
                      [5]);
47              }
48            }
49          }
50         }
```

12.2. Resümee

Die Implementation hat gezeigt, dass die Anwendung grundsätzlich funktioniert. Es zeigte sich jedoch, dass Probleme in punkto Programmstabilität und Übertragungsfehler bei der seriellen Kommunikation gab. Aufgrund der Übertragungsfehler wurden teilweise die die Accelerometerwerte verfälscht oder gar nicht übert-

ragen, obwohl die Implementation des Protokolls bei Tests fehlerfrei funktionierte. Leider war es uns nicht möglich diese Fehler zu beheben. Nach einer Recherche im Internet stellte sich heraus, dass Fehler dieser Art auf die frühe Version der Java Communications API (javax.comm) zurückzuführen waren, welche zu diesem Zeitpunkt, laut den Berichten anderer Nutzer, auf manchen Systemen noch Probleme bereitet hat. Aufgrund der Unzuverlässigkeit des Systems, war die Anwendung für weitere Tests nicht zu gebrauchen. Deswegen wurde die Entwicklung eingestellt.

A. Literatur

[1] Technical report on c++ performance. 10

[2] *11th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS 2003), 22-23 March 2003, Los Angeles, CA, USA, Proceedings*. IEEE Computer Society, 2003. 6, 43

[3] cgkit. The python computer graphics kit. http://cgkit.sourceforge.net/ [Last visited March 2006]. 17

[4] Analog Devices. Adxl202 +/-2 g dual axis accelerometer. http://www.analog.com/en/prod/0,2877,ADXL202,00.html [Last visited March 2006]. 9, 18

[5] Thomas Fuhrmann, Markus Klein, and Manuel Odendahl. Bluewand-A versatile remote control and pointing device. In *KiVS Kurzbeiträge*, pages 81–88, 2003. 20

[6] Julius O. Smith III. Introduction to digital filters, December 26 2002. 14

[7] Andrea Mazzone, Rui Zhang, and Andreas M. Kunz. Novel actuators for haptic displays based on electroactive polymers. In *VRST*, pages 196–204, 2003. 6

[8] pyODE. pyode bindings. http://pyode.sourceforge.net/ [Last visited March 2006]. 17

[9] pySerial. pyserial bindings. http://pyserial.sourceforge.net/ [Last visited March 2006]. 17

[10] Enzo Pasquale Scilingo, Nicola Sgambelluri, Danilo De Rossi, and Antonio Bicchi. Haptic displays based on magnetorheological fluids: Design, realization and psychophysical validation. In *HAPTICS* [2], pages 10–15. 6

[11] Russell M. Taylor, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T. Helser. VRPN: a device-independent, network-transparent VR peripheral system. In *VRST*, pages 55–61, 2001. 13