Universität für Musik und darstellende Kunst, Graz
Institut für elektronische Musik und Akustik

Schlagzeug als Hyperinstrument mit kabellosen Beschleunigungssensoren

SS 2009

Projektarbeit

unter Betreuung von
Ao.Univ.-Prof. DI Winfried RITSCH

Florian KREBS
MatNr: 0231030
Datum: 26.03.2009

Contents

1 Einleitung

In seiner Werkserie TOYSRUS beschäftigt sich der Schlagzeuger und Klangkünstler Josef Klammer mit der Weiterentwicklung des natürlichen Drumsounds durch Live-Elektronik. Im Rahmen eines Toningenieursprojekts am IEM Graz soll in Zusammenarbeit mit Josef Klammer ein Hyperinstrument entstehen, das anschließend in einem Konzert verwendet werden soll. Die natürlichen Klänge des Schlagzeugs sollen hierfür mit Mikrofonen aufgenommen und in Echtzeit moduliert werden. Modulationsparameter werden mit Sensoren gemessen und an den Rechner geschickt.

Josef Klammer Der 1958 in Lienz (Osttirol) geborene Musiker und Medienkünstler Josef Klammer studierte Schlagzeug an der Universität für Musik in Graz und ist seit vielen Jahren als freier Musiker tätig. Neben vielen internationalen Konzerten ist Josef Klammer seit 1999 an der Organisation des V:NM Festivals (V:NM = Verein zur Förderung von Neuer Musik) in Graz beteiligt. 1994 wurde ihm eine Auszeichnung für Computermusik des Kulturministeriums, 2003 der Electronic Award des Musikforums Viktring für das *Klammer&Gründler Duo* verliehen.

Figure 1: Josef Klammer TOYSRUS

In seiner Werkserie **TOYSRUS** verwendet er das "'Schlagzeugspiel zur Echtzeitmodulation des Schlagzeugspiels und des Schlagzeugklangs. Die Daten dafür extrahiert Klammer aus seinem Gestus, seiner Dynamik und der Diadochokinese der vier Extremitäten. Damit werden Tonhöhen, Delayzeiten und Filterungen sowie die Aufteilung der Klänge auf das 8-kanalige Lautsprechersystem gesteuert."'

2 State of the art

Wenige Projekte wurden gefunden, bei denen versucht wurde das Spiel des Schlagzeugers durch eine Analyse der ausgeführten Bewegungen zu bereichern, beschränken sich doch viele Arbeiten darauf, einen Anschlag am Fell zu detektieren und damit das Abspielen von Samples zu steuern. Einige Beispiele möchte ich dennoch hier nennen :

The Radio-Baton entwickelt 1997 von Max Matthews. Eines der ersten Systeme, die in der Lage sind, der aktuellen Position der Sticks im dreidimensionalen Raum zu folgen. Er verwendet kleine, in den Stick integrierte Antennen und misst die Veränderung der Signalstärke bei der Übertragung zu einer fix montierten Antenne. Das System funktioniert allerdings nur auf einem eng begrenztem Bereich.

La percussion virtuelle du Scrime entwickelt 2004, bietet ein virtuelles Schlagzeug basierend auf Gestenerkennung/Positionsbestimmung in der Luft. Als Tracking Methode wird hier das "Flock of Birds" System verwendet, das mit Magnetsensoren arbeitet. Nachteilig sind hier die sehr hohen Kosten des "Flock of Birds" Systems.

Das **GMEM Marseille** hat ein System entwickelt, das Bewegungen per Kamera trackt und so verschiedene Samples steuert. Für unsere Anwendung kommt ein optisches Trackingsystem jedoch nicht in Frage, da die Hände des Schlagzeugers oft verdeckt sind, bzw. das Tracking für schnelle Bewegungen zu langsam wäre.

Charles Verron entwickelt 2003/04 bei **La Kitchen Paris** ein erweitertes Drumset, bei dem ein Magnetsensor zur Winkelmessung, sowie ein Beschleunigungssensor, der die tangentiale Beschleunigung misst, in die Sticks integriert werden. Die erhaltenen Parameter werden verwendet um den Schlagzeugsound zu ergänzen. Die Entwicklung ist allerdings sehr aufwendig, da mehrere Sensoren in den Stick integriert werden müssen. [6]

STEIM Labs (Studio for electro-instrumental music) beschäftigen sich seit vielen Jahren mit Motion Tracking und dessen Anwendung in musikalischen Aufführungen. Die Anwendung *junXion* kann bis zu 8 Inputsensoren verwalten, deren Daten verarbeiten und an verschiedene Audio-Programme über MIDI/OSC weiterleiten. Derzeit ist noch keine spezielle Anwendung für Schlagzeug bekannt.

3 Grundlagen

3.1 Beschleunigungssensoren

3.1.1 Prinzip

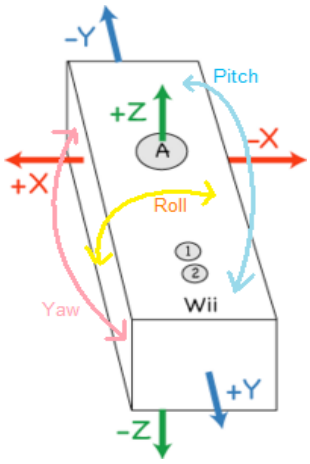
Ein Beschleunigungssensor erfasst die auftretenden Beschleunigungen entlang seiner Messachse. Zur Berechnung wird das zweite Newton'sche Gesetz angewandt, das besagt, dass die auftretende Kraft F abhängig von der Masse m und der Beschleunigung a ist:

$F = m \times a$

Die auftretenden Beschleunigungen werden nach dem Masse-Feder System ermittelt: Die dabei auftretende Kraft F lenkt die Feder aus. Über die Größe der Auslenkung x kann F ermittelt und die Beschleunigung errechnet werden, wenn die Masse bekannt ist. Die Beschleunigungssensoren unterscheiden sich dadurch wie sie die Auslenkung messen.

3.2 Bewegungsgleichungen

3.2.1 Berechnung der Rotationswinkel aus Beschleunigungsdaten



Ein starrer Körper im 3D Raum besitzt 6 Freiheitsgrade: 3 lineare Bewegungsachsen (X,Y,Z), sowie 3 Rotationswinkel (Pitch, Roll, Yaw). Mit einem dreiachsigem Beschleunigungssensor wie dem ADXL330 können allerdings nur drei davon gemessen werden (X,Y,Z), es ist also nicht möglich die genaue Position und Lage zu bestimmen. Auch die Gravitationsbeschleunigung beeinflusst eine Messung: Ist der Körper in Ruhe, d.h. die Vektorsumme der gemessenen Beschleunigungen entspricht exakt der Gravitation, so kann man die Rotationswinkel Roll und Pitch errechnen. Die Rotationsachse des Winkels liegt in der Standardposition des Sensors parallel zur Achse z. Somit kann die Gravitationskraft keine Arbeit in der Yaw-Rotationsachse verrichten, weshalb wir diesen Winkel nicht berechnen können. Wir beschränken uns hiermit auf Roll und Pitch. Unter der Vorraussetzung, dass der Sensor in Ruhe ist, gilt bei Verwendung eines rechthändigen Koordinatensystems: [2]

$$Roll = \begin{cases} arctan(x/z), & \text{für } z < 0 \\ arctan(x/z) + \pi, & \text{sonst} \end{cases}$$

$$Pitch = \begin{cases} arctan(-y/z), & \text{für } z < 0 \\ arctan(-y/z) + \pi, & \text{sonst} \end{cases}$$

3.2.2 Die Rotationsmatrix

Um Bewegungen analysieren zu können, interessiert uns die Richtung einer Bewegung gemessen in einem von außen fixen Koordinatensystem unabhängig von der jeweiligen Rotation des Körpers und von der Gravitationsbeschleunigung. Ist der Sensor nicht in Normalposition, wirkt die Gravitation je nach Rotation auf mehrere Achsen und kann nicht mehr so einfach abgezogen werden. Wir müssen nun erst den gemessenen Beschleunigungsvektor in die Ruhelage (=Normalposition) rücktransformieren, um dann die Gravitation abziehen zu können und so die Bewegungsbeschleunigung zu erhalten. Diese Rücktransformation erfolgt mit Hilfe der Rotationsmatrix, die wir aus Pitch und Roll ermitteln können. Ich möchte daran erinnern, dass der Winkel Yaw hier außer Acht gelassen wurde, da er mit einem dreiachsigem Beschleunigungssensors nicht gemessen werden kann. Da wir Pitch und Roll aber nur korrekt ermitteln können, wenn der Sensor in Ruhe ist, kann hier nur angenähert werden: Die Rotationsmatrix wird jedesmal upgedatet, wenn der Sensor in Ruhe ist. Zwischendurch wird angenommen, dass sich die Rotation des Sensors nicht ändert. Die Beschleunigung a_{mov} (Beschleunigung die abzüglich der Gravitationsbeschleunigung g auf den Körper wirkt) lässt sich darstellen als:

$$a_{mov} = R^{-1} * a_{measured} - g$$

R entspricht der Rotationsmatrix (Yaw vernachlässigend), die sich aus [2] ergibt zu :

$$\begin{pmatrix} cos(Roll) & sin(Roll) * sin(Pitch) & sin(Roll) * cos(Pitch) \\ 0 & cos(Pitch) & -sin(Pitch) \\ -sin(Roll) & cos(Roll) * sin(Pitch) & cos(Roll) * cos(Pitch) \end{pmatrix}$$

Als Pitch und Roll werden wie oben angesprochen die Werte der letzten Sensorruheposition (Vektorsumme aller drei Achsen = 1 g) verwendet.

Angemerkt sei, dass die Matrix R nicht immer regulär und somit nicht immer invertierbar ist. Dadurch können Mehrdeutigkeiten für Pitch und Roll entstehen. Um diesem Problem zu entgehen, interessiert uns in dieser Anwendung in erster Linie der Roll. Wir verwenden deshalb aus unserer ersten Berechnung nur den Roll, rotieren dann den Körper um den Winkel Roll zurück in die "'Roll-Ruheposition"' und errechnen jetzt den Pitch erneut. Verwendet wird dann der Roll der ersten Berechnung sowie der Pitch der zweiten Berechnung.

3.3 Datenübertragung

3.3.1 Bluetooth

Bluetooth ist ein in den 90er Jahren entwickelter Standart für kabellose Kommunikation innerhalb kurzer Distanzen. Gesendet und empfangen wird im unlizensierten Frequenzband um 2.4GHz. Die Datenraten können bei Bluetooth derzeit bis 3 MBit/s betragen. Bluetooth-Geräte können nach ihrer Leistung bzw. Reichweite in drei Klassen eingeteilt werden, wobei die größte Reichweite (bis 100m) von einem Klasse-1-Gerät erreicht wird.

Die Kommunikation eines Remote Device (Peripheriegerät) mit dem Local Device (Bluetooth Antenne des Computers) benötigt einen Bluetooth Stack, der die Implementierung des Bluetooth Protokolls [5] darstellt. Dieser Stack muss vom Local Device unterstützt werden. Das Remote Device hat seinen Stack meist intern implementiert. Die bekanntesten Bluetooth Stacks für die verschiedenen Plattformen sind:

- Windows: Widcomm, Bluesoleil
- Linux: Bluez
- Mac OS: Apple Bluetooth Stack

Im Nintendo Wiimote-Controller wird beispielsweise der Broadcom Bluetooth-Chip BCM2042 verwendet, der sowohl den Bluetooth Stack als auch das HID (Human Interface Device [4]) Protokoll auf einem Chip implementiert hat.

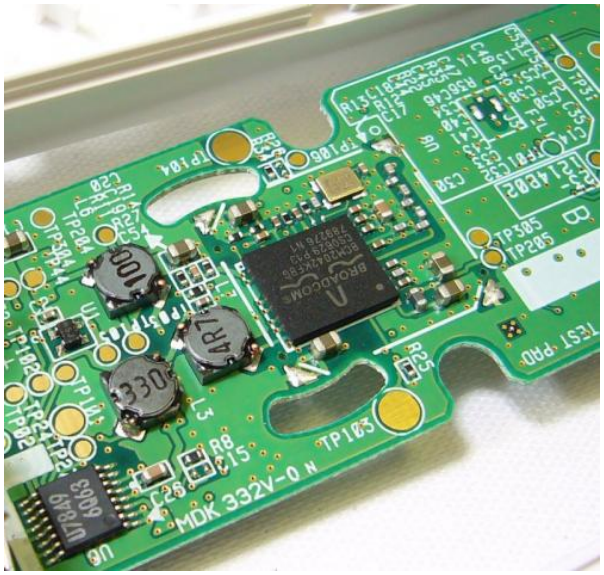


Figure 2: Beispiel eines Bluetooth Chips (quadratischer Broadcom-Chip) im Wiimote-Controller

3.3.2 Das OSC Protokoll

OSC (Open Sound Control) ist ein Protokoll zur Kommunikation zwischen Computern, Sound Synthesizern und anderen Multimedia Geräten [8]. In den späten 90er Jahren an der University of California in Berkeley entwickelt, soll es unter anderem das 1983 standartisierte MIDI Format ablösen. OSC ist unabhängig vom Transportlayer, wobei meist jedoch Ethernetverbindungen mit dem UDP Protokoll verwendet werden. Jeder moderne Computer verfügt über eine Netzwerk Schnittstelle mit Internet Protokol (IP) was die Anschaffung eines externen Gerätes, wie bei MIDI, unnötig macht. Das einfache aber dennoch mächtige Protokoll eignet sich hervorragend für Multimedia-Echtzeitanwendungen. Es bietet folgende Features:

- Offenes, dynamisches, URL-ähnliches symbolisches Benennungsschema
- Symbolische and hochauflösende numerische Datenargumente
- Mustererkennungssprache um mehrfache Empfänger einer Nachricht zu spezifizieren
- High resolution time tags
- Nachrichtenpakete für Daten die gleichzeitig in Effekt treten sollen
- Anfragensystem um dynamisch Informationen über die Eigenschaften eines OSC Servers zu beziehen

Figure 3: Pure Data: OSC senden

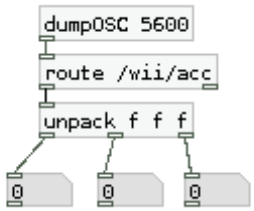


Figure 4: Pure Data: OSC empfangen

Zahlreiche Computermusik Programme unterstützen heutzutage das OSC-Protokoll, so z.B. Pure Data, Reaktor, MaxMSP, ... Die Anwendung ist sehr einfach: Zum Senden wird lediglich eine Portnummer für UDP oder TCP/IP gewählt, die Daten werden dann in Verzeichnisse gepackt (z.B. /pd/Roll) und verschickt (siehe Fig. 3).

3.4 Pure Data

Pure Data (Pd) ist eine freie grafische Entwicklungsumgebung für Audio und Videobearbeitung in Echtzeit. Pd wurde ursprünglich unter Leitung von Miller Puckette am IRCAM Institut entwickelt und von zahlreichen Programmierern weiterentwickelt. Es gehört zur Familie der Patch-Programmiersprachen wie auch z.B. Max/MSP. Pure Data unterstützt das OSC-Protokoll mit einer eigenen Library.

4 Realisierung

4.1 Anforderungen des Künstlers

Folgende Anforderungen an den Sensor wurden vom Künstler definiert:

- Positionsmessung als Endziel
- Kabellose Verbindung
- Kostengünstig, um weitere Exemplare bauen zu können
- Einfache Anbindung
- Sensor auf Handschuh montierbar
- Nicht zu schwer, darf nicht beim Spielen behindern
- Niedriger Energieverbrauch, Batterie aufladbar
- Stabil, robust

4.2 Wahl des Sensors

Aufgrund der begrenzten Entwicklungzeit und des engen finanziellen Rahmens wurde auf schon fertige kostengünstige Produkte gesetzt und von einer Eigenentwicklung abgesehen. Einzelbauteile sind teurer als das gesamte fertige Modul, so dass wir uns für eine Modifizierung eines fertigen Produkts entschieden haben. Im folgenden möchte ich die Modelle, die zur Auswahl standen, kurz vorstellen.

4.2.1 Sony Sixaxis

Der Sixaxis-Controller wurde für die Playstation 3 entwickelt und kam 2006 auf den Markt. Die Verbindung zur Konsole erfolgt drahtlos über Bluetooth. Er verfügt über einen dreiachsigen Beschleunigungssensor sowie einen Gyrometer, der in der Lage ist die Winkelbeschleunigung zu messen. Da hier vier von sechs Freiheitsgraden gemessen werden, kann eine Positionsbestimmung genauer erfolgen als ohne Gyrometer. Der Controller ist leicht erhältlich, und im unteren Preissegment anzusiedeln (ca. 35). Von der Größe her ist der Sensor eher an der oberen Grenze.

4.2.2 Nintendo Wiimote

Der Wii-Controler (auch Wiimote genannt) kommuniziert über Bluetooth mit der Konsole und verfügt über eine Vielzahl von Features. Es erfreut sich seit seiner Erscheinung auf dem Markt Ende 2006 immer größerer Beliebtheit bei Hackern und Bastlern. Das relativ kostengünstige Gerät (ca 40 Euro zur Zeit) bietet u.a. den dreiachsigen Akzelerationssensor ADXL330, eine Bluetooth Schnittstelle, einen Lautsprecher, einen kleinen Vibrationsmotor sowie eine Infrarotkamera. Mittels Infrarotkamera kann z.B. eine zusätzlich erhältliche Infrarotleiste angepeilt werden und das Wiimote so als Maus benutzt werden. Für eine Gestenerkennung für einige Spiele werden ausschließlich die Akzelerationssensoren benutzt, die hierzu nötige Software wurde im Auftrag von Nintendo von der Firma AiLive entwickelt. Für verschiedenste Betriebssysteme gibt es außerdem eine Reihe von Libraries, die die Verwendung eines Wiimotes unterstützen.

4.2.3 Vergleich

Beide Sensoren verfügen über dreiachsige Beschleunigungssensoren und lassen sich über Bluetooth an den Rechner anbinden. Der Sixaxis besitzt einen zusätzlichen Gyrometer, der für eine genauere Positionsbestimmung interessant wäre. Allerdings ist der Wiimote bislang besser untersucht, es gibt zahlreiche Programmbibliotheken und somit viele wertvolle Erfahrungswerte. Bei meinen Tests mit dem Sixaxis war es mir auch nicht sofort möglich, das Pairing über Bluetooth durchzuführen, bzw. die Beschleunigungsdaten auszulesen. Die Platine des Wiimote erfüllt auch eher meine Anforderungen an einen

Sensor, so dass ich mich schlussendlich für die Verwendung des Wiimote-Controllers entschied.

4.2.4 ADXL330

Einer der gängigsten Beschleunigungssensoren ist der ADXL330. Der kleine, dünne dreiachsige Beschleunigungssensor hat eine niedrige Stromaufnahme (320μA bei 1,8 V) und liefert ein signalabhängiges analoges Spannungssignal zwischen 0 und 3 V. Er misst Beschleunigungen im Bereich von $\pm$ 3 g und wird auch z.B. in Nintendos Wiimote verwendet.

Figure 5: Der ADXL330 im Nintendo Wiimote (quadratischer IC unten)

Der ADXL330 ist ein kapazitiver Sensor. Er besteht aus drei Kondensatorplatten. Die beiden äußeren Platten sind hierbei mit dem Sensorgehäuse fest verankert, während die mittlere Platte an der Masse befestigt ist. Die Masse ist über zwei Federn beweglich mit dem Sensorgehäuse verbunden. Wird die Masse beschleunigt, so kommt es zu einer Änderung der Kapazität, und damit zu einer der Beschleunigung proportionalen Änderung der anliegenden Spannung. Benötigt wird eine Betriebsspannung zwischen 2 und 3,6 V. Diese wird dem Wiimote durch zwei AA-Batterien in Serie zugeführt.

4.3 Adaption des Wiimotes

4.3.1 Testentwurf

Für einen Testentwurf soll der Sensor ADXL330 auf dem Handrücken, das restliche Wiimote mit der Stromversorgung und dem Bluetoothsender/Empfänger jedoch getrennt am Oberarm befestigt werden. Somit soll das Gewicht des Gerätes auf der Hand möglichst gering gehalten werden. Um jedoch den Sensor auf dem Handrücken befestigen zu können, muss der Sensor ausgelötet und durch ein Kabel verlängert werden. Dies ist äußerst schwierig, da der Sensor sehr klein ist (3x3mm), als SMD-Bauteil vorliegt und mit über 16 Lötstellen mit der Platine verlötet ist. Der Sensor wird nun auf eine Platine gelötet und die einzelnen Pins mit einem fünfadrigem Kabel verbunden (Vcc, Vx, Vy, Vz, Masse). Das Wiimote kann nun am Oberarm befestigt werden.

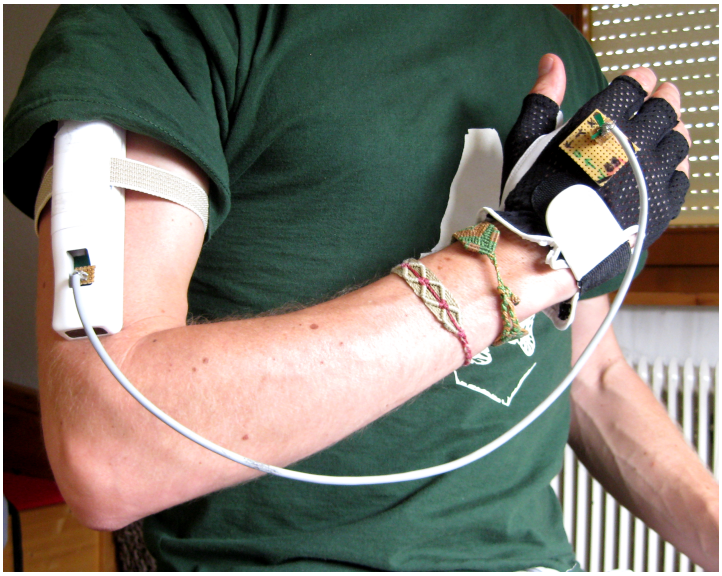


Figure 6: Testentwurf

Nachdem erste Tests durchgeführt waren, stellte sich heraus, dass sowohl die Befestigung des Wiimotes am Oberarm als auch das Kabel am Unterarm die Bewegungsfreiheit einschränkten.

4.3.2 Performanceversion

In einer zweiten Version sollte nun das Gewicht des Wiimotes reduziert werden, um es als Ganzes auf dem Handrücken zu befestigen. Hierzu wurden folgende Adaptionen vorgenommen:

Gehäuse Das Gehäuse wurde komplett entfernt. Nur die Leiterplatine wurde verwendet und zum Schutz in Epoxidharz gegossen. Bei einem Versuch vorne und hinten die Platine zu verkürzen, stellte sich heraus dass sich die Bluetooth-Reichweite extrem verkürzte, so dass ich schließlich die gesamte Platine verwendete. Eventuell wurde die Antenne beschnitten oder eine wichtige Komponente des Senders/Empfängers durchtrennt.

Stromversorgung Die Stromversorgung des Wiimotes mit zwei AA-Batterien ist recht großzügig ausgelegt. Ein erster Versuch mit Knopfzellen (SONY CR2032 Lithium Knopfzelle, 3V, 220mAh) schlug fehl, da die Kapazität wohl zu gering war. Bei längerem Gebrauch des Wiimotes wurde die Verbindung ständig unterbrochen. Knopfzellen mit größerer Kapazität hätten den Preis extrem in die Höhe getrieben, deshalb verwendete ich schließlich zwei AAA-Mikrozellen (mit je ca. 700mAh).

Infrarotkamera, Buttons und Extension-Buchse Außerdem wurden die Infrarotkamera, die Buttons sowie die Nunchuck-Extension-Buchse entfernt, da sie nur zusätzliches Gewicht darstellten und für diese Anwendung nicht benutzt werden.

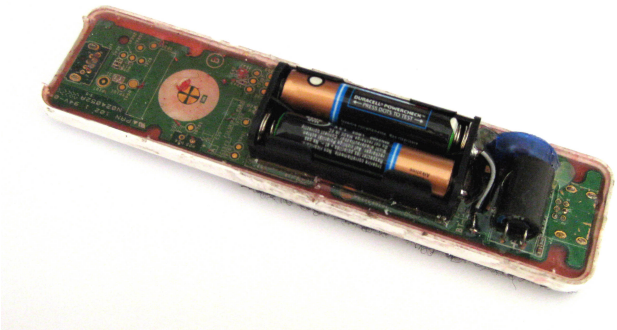


Figure 7: Performanceversion

4.4 Software

4.4.1 Verbindung des Wiimotes unter Pure Data

Zuerst muss eine Verbindung zwischen dem Wiimote (*remote device*) und der lokalen Bluetooth-Instanz (*local device*, z.B. Bluetooth-Dongle) hergestellt werden. Hier ist es wichtig den zum lokalen Bluetoothgerät passenden Bluetooth Stack zu verwenden.

Die weitere Verarbeitung der Daten wurde zwecks der guten Echtzeiteigenschaften in *Pure Data* vorgenommen. Um auf die Daten in Pd zugreifen zu können, benötigt man entweder ein eigens dafür geschriebene Library, das die Kommunikation mit dem Betriebssystem übernimmt, oder man schickt die Daten über das OSC-Protokoll an Pure Data. Da die jeweilige Anbindung eines Bluetoothgeräts plattformspezifisch ist (unterschiedliche Bluetooth-Stacks), muss hier unterschieden werden:

Windows Eine Verbindung unter Windows hat sich am problematischsten erwiesen. Unter Windows gibt es eine Vielzahl von Bluetooth-Stack Treibern. Die bekanntesten Treiber sind *Widcomm* und *Bluesoleil*. Nicht jeder Bluetooth-Dongle funktioniert aber mit jedem Bluetooth-Stack, sodass hier die richtige Kombination gefunden werden müsste. Um die Daten in Pure Data einzulesen, existiert für Windows die Bibliothek *WiiSense*.

Linux Unter Linux wird der *bluez*-Stack verwendet, der problemlos funktionierte. Das Pure Data-External *Wiimote* von Mike Wozniewski [7], basierend auf der cWiid-Library von L. Donnie Smith, wurde von mir auf die neueste Version upgedatet und funktioniert bisher ohne Fehler.

Mac OS Mit dem *Apple Bluetooth-Stack* hatte ich ebenfalls keine Probleme. Ein Pure Data External existierte zur Zeit meines Projektes für Mac OS noch nicht, so dass ich die freie Software *darwiinremoteOSC* verwendete, um Daten vom Wiimote einzulesen und über OSC an Pure Data zu senden.

4.4.2 Die Software Hyperdrums

Die von mir entwickelte Pd-Patch *Hyperdrums* soll folgende Funktionen ermöglichen:

1. Einlesen der Beschleunigungsdaten
2. Kalibrierung und Offset Abgleich
3. Berechnung der Rotationswinkel Roll und Pitch
4. Eliminierung der Gravitationsbeschleunigung

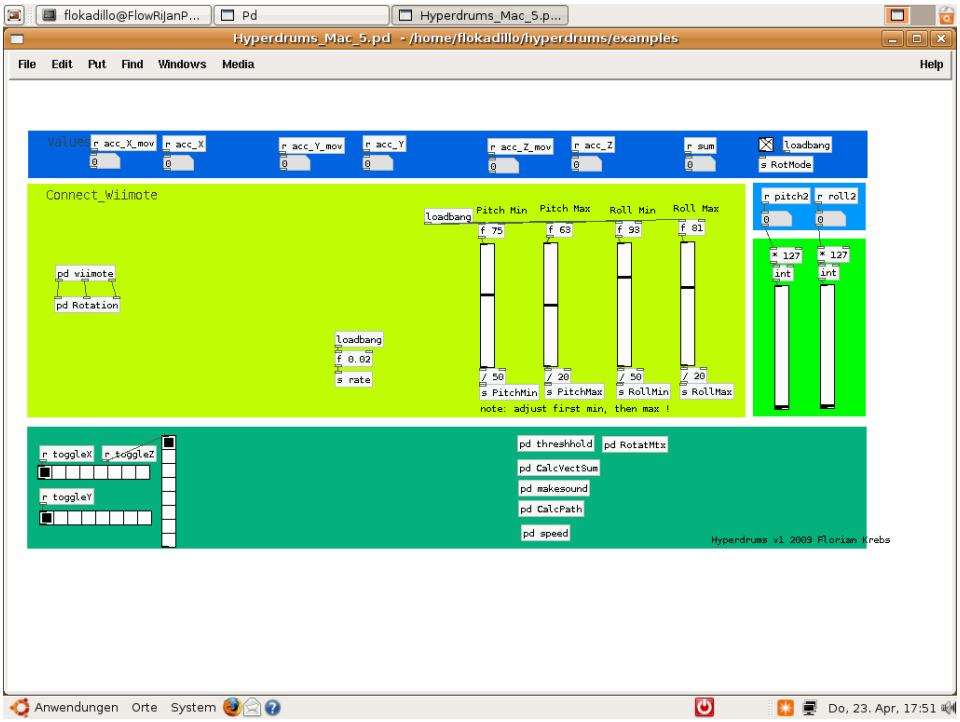


Figure 8: Hyperdrums

Einlesen der Beschleunigungsdaten Wie schon erwähnt existieren für die Betriebssysteme Linux und Windows fertige Libraries für Pure Data, die eine leichte Verwendung des Wiimotes ermöglichen. Unter Mac OS schicke ich die Wiimote-Daten von darwiinremoteOSC über OSC an Pure Data.

Kalibrierung und Offset Abgleich Um zu gewährleisten, dass die gemessenen Werte symmetrisch um den Nullpunkt liegen, also keine Spannungsoffsets durch Signalstörungen (Kabel, Lötstellen,...) auftreten, müssen die Sensoren kalibriert werden. Dazu positioniere ich das Wiimote auf eine ebene Fläche. Für alle sechs Positionen (entsprechend $+/-1\ g$ auf jeder Messachse) wird die auftretende Gravitationsbeschleunigung gemessen. Der Wert sollte auf allen Positionen vom Betrag her gleich $1\ g$ sein. Aus der Differenz zwischen gemessenen Werten und Sollwert wird für jede Achse ein Wert zum Offsetabgleich berechnet.

Berechnung der Rotationswinkel Roll und Pitch Siehe auch S. 5

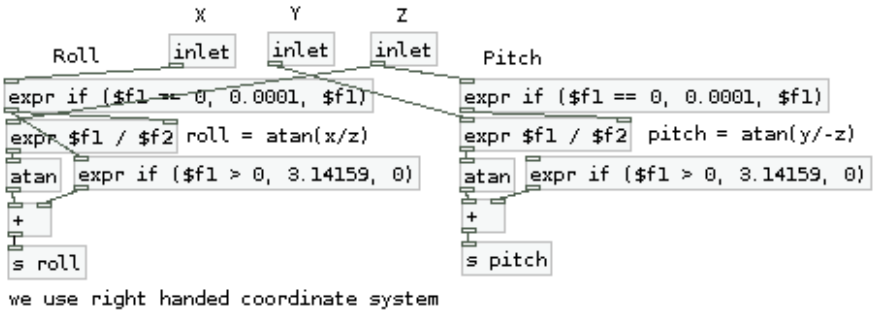


Figure 9: Berechnung von Pitch und Roll in Pure Data

Eliminierung der Gravitationsbeschleunigung Hier werden die Rohbeschleunigungsdaten mit der Rotationsmatrix in die Ruheposition zurücktransformiert, um die Gravitationsbeschleunigung abzuziehen. Das Ergebnis kann nun zur Aufintegration für die Ermittelung des zurückgelegten Weges verwendet werden.

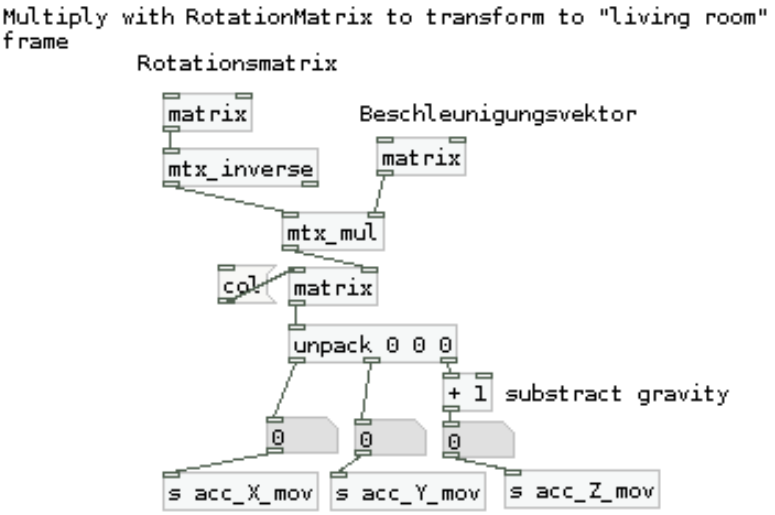


Figure 10: Anwendung der Rotationsmatrix in Pure Data

4.5 Messungen

4.5.1 Genauigkeit der Sensoren

Um zu kontrollieren, dass sich bei einer längeren Messung keine Offsets einschleichen, wurde folgender Versuch unternommen: Der Sensor wurde in der Z-Ebene (auf dem Tisch liegend) auf der X-Achse für 15 s lang hin- und herbewegt, sodass der Endpunkt am Schluss wieder dem Anfangspunkt entspricht. Wenn man nun die Beschleunigungswerte zweimal integriert, müsste als zurückgelegter Weg Null herauskommen, da der Startpunkt dem Endpunkt entspricht. Aufgenommen wurden 500 Samples bei einer Samplingrate von 33,3 Hz, wobei die Beschleunigung zwischen zwei Samples als konstant angenommen wurde. Da die Bewegung mit freier Hand (und deshalb nicht genau entlang der X-Achse) ausgeführt wurde, ergibt sich auch eine Komponente in Y-Richtung.

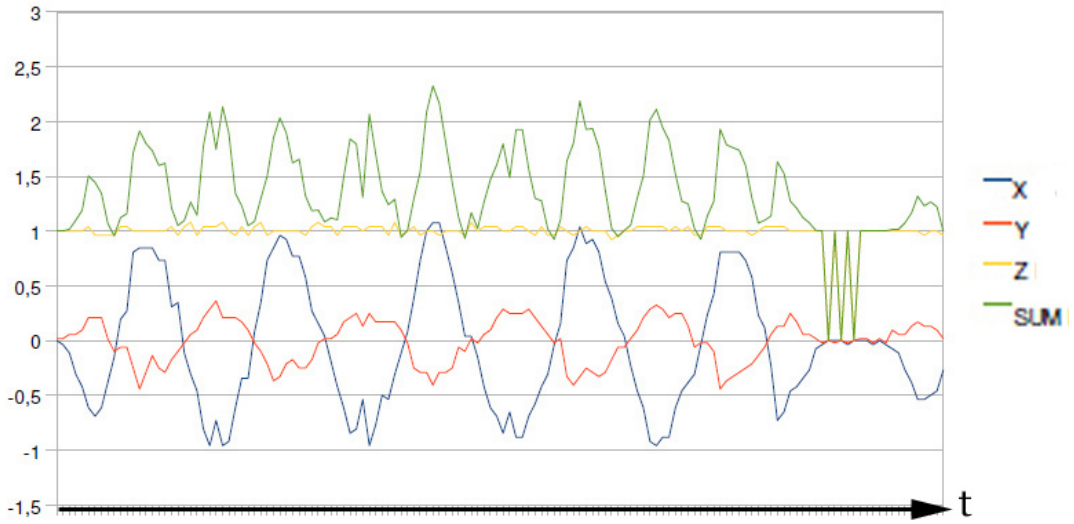


Figure 11: Beschleunigung

Einmal integriert ergibt sich der Verlauf der Geschwindigkeit:

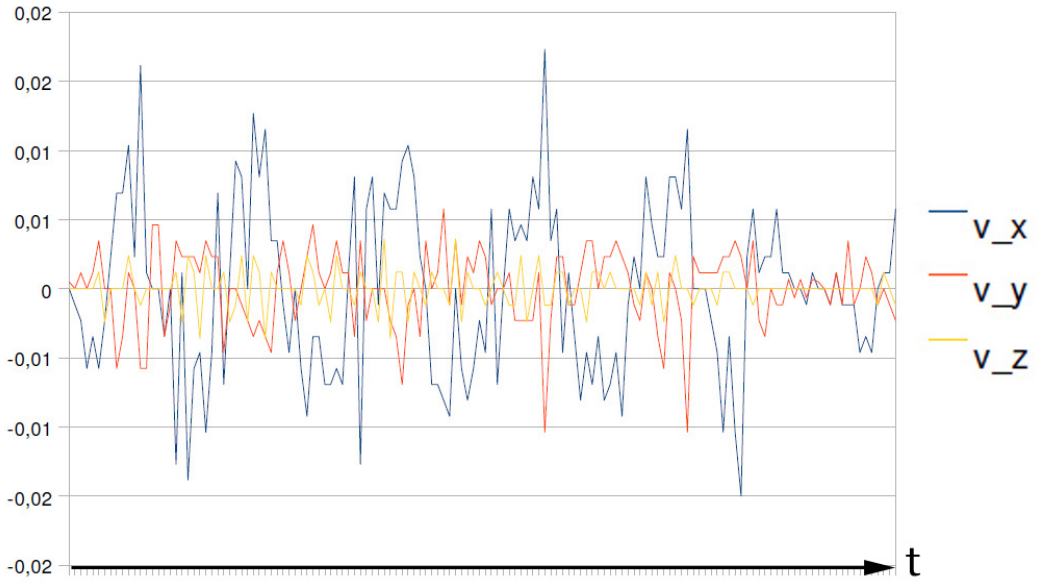


Figure 12: Geschwindigkeit

Zweimal integriert der zurückgelegte Weg:

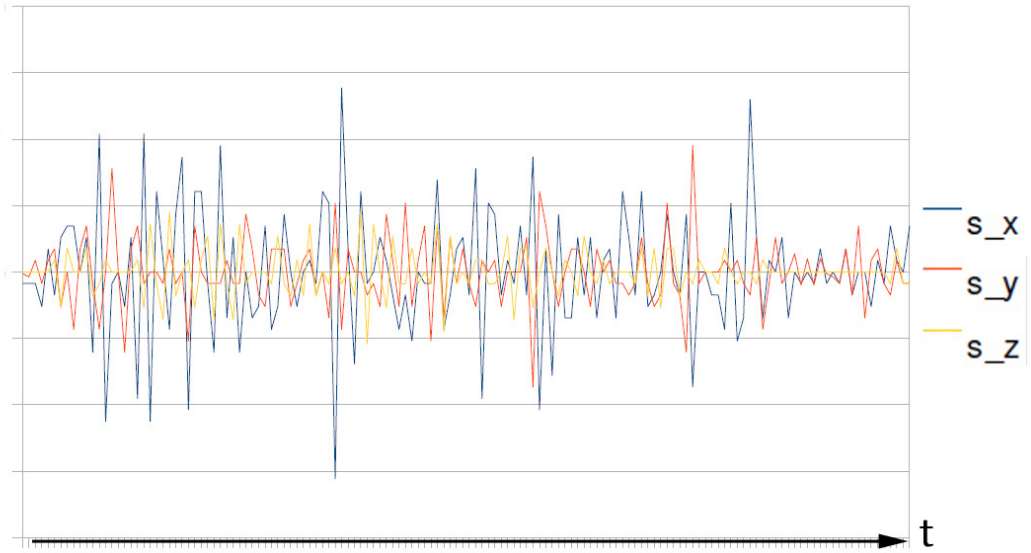


Figure 13: Weg

Alle Werte einer Bewegungsrichtung aufsummiert müsste Null ergeben, da der Anfangs- und Endpunkt meiner Bewegung gleich sind. Für die Summe aller Wegstücke erhalten wir:
x-Sensor: 3,46 mm
y-Sensor: 0 mm
z-Sensor: 0 mm
Die Abweichung für eine 15 s lange Bewegung ist sehr klein. Die Sensoren geben sehr gut wieder was in der Realität passiert. Dennoch haben wir uns hier auf eine Messachse beschränkt, die Messung im freien Raum (sechs Freiheitsgrade) ist um einiges komplexer.

4.5.2 Positionsbestimmung

Dass eine genaue Positionsbestimmung mit einem dreiachsigem Beschleunigungssensor nicht möglich ist, wurde weiter oben dargelegt. Folgende Vereinfachungen werden jedoch angenommen um eine Näherung zu erhalten:

- Roll und Pitch zwischen Ruhenlagen konstant (siehe S. 6)
- Beschleunigung zwischen zwei Messungzeitpunkten konstant

Zur Positionsbestimmung soll zuerst der in jede Messachsenrichtung zurückgelegte Weg bestimmt werden. Der Weg wird bestimmt durch zweifache Integration der Beschleunigung über ein Messintervall. Um ein Wegdriften der Ergebnisse besser beurteilen zu können, wurde der Sensor für ca. 10 s im Raum frei hin- und herbewegt, so dass jedoch die Anfangsposition der Endposition entspricht, der Gesamtweg also Null sein sollte. Die Messung wurde mit unterschiedlichen Threshholds durchgeführt, d.h. ein Beschleunigungswert wird erst verwendet wenn er einen bestimmten Schwellenwert überschreitet.

Es ergibt sich folgende Messreihe für die Y-Richtung:

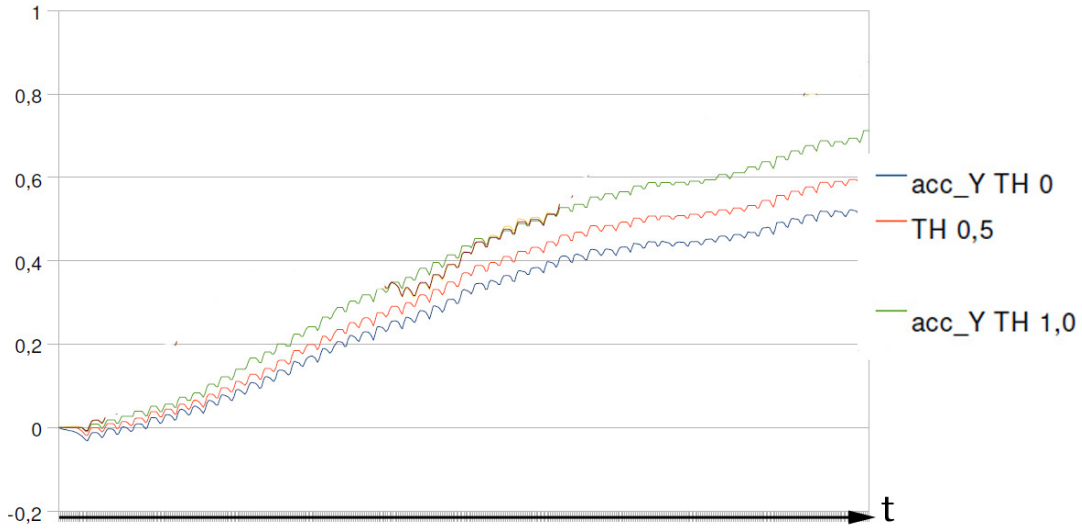


Figure 14: Zurückgelegter Weg in Y-Richtung

Deutlich erkennbar ist hier ein genereller Anstieg des Weges, der nicht durch die Bewegung des Sensors verursacht wurde. Ein höherer Threshhold bringt generell schlechtere Werte. Die Welligkeit wurde von der Sensorbewegung erzeugt.

Setzt man aber bei jedem Wegminimum den zurückgelegten Weg auf Null zurück, so kann dieses Abdriften verhindert werden:

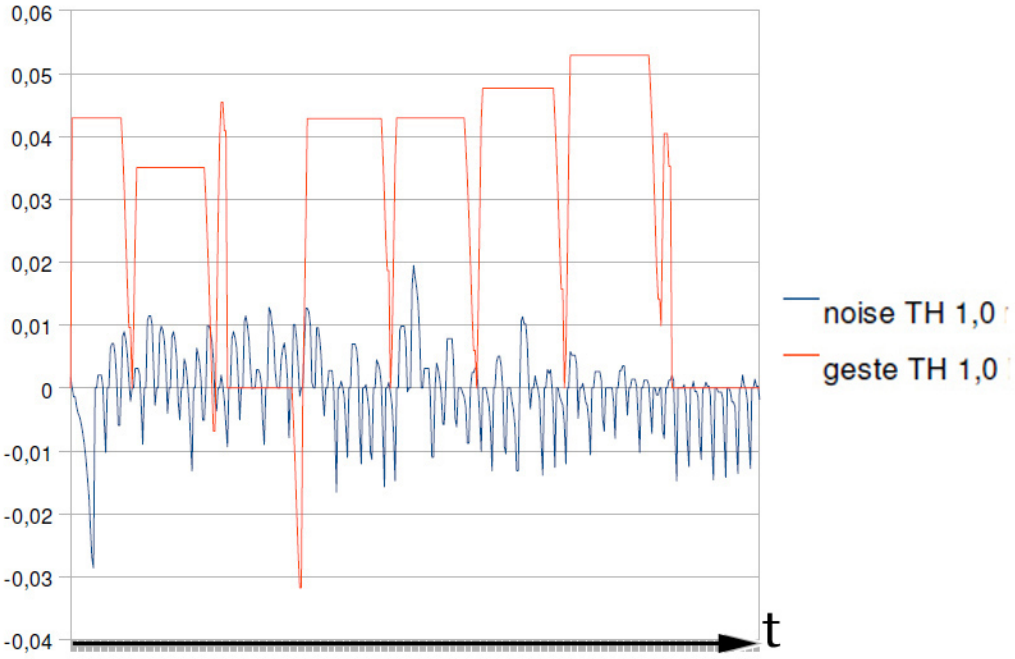


Figure 15: zurückgelegter Weg mit Reset

Die rote Linie entspricht hier einer geradlinigen Bewegung in Richtung der Y-Achse. Die Detektion eines Minimum erfolgt, wenn die Geschwindigkeit von zwei aufeinander folgenden Werten ihr Vorzeichen von negativ auf positiv ändert. Außerdem wird geprüft, ob die Beschleunigung am gleichen Punkt positiv ist. Eine Drift der Sensoren konnte hierdurch verhindert werden, dennoch ist die fortlaufenden Messung einer Position im Raum nicht möglich. Für eine Messung der Position müsste man hier mit zusätzlichen Sensoren (z.B. Gyrometern) arbeiten.

4.6 Parameter zur Klangerzeugung

Folgende Parameter wurden schließlich zur Klangerzwugung verwendet:

- Beschleunigungsdaten (mit/ohne Gravitation)
- Lagewinkel Pitch/Roll
- Gesamtbeschleunigung

5 Evaluierung

Um die Fragestellung nach einer praktischen Verwendbarkeit des entwickelten Moduls zu testen, wurde in einem Konzert eine Testumgebung geschaffen. Der Musiker Josef Klammer entwickelte eigens für das System Hyperdrums eine Komposition, die am 05.05.2009 im CUBE des Instituts für elektronische Musik aufgeführt wurde. Folgendes Setup wurde hierbei verwendet:

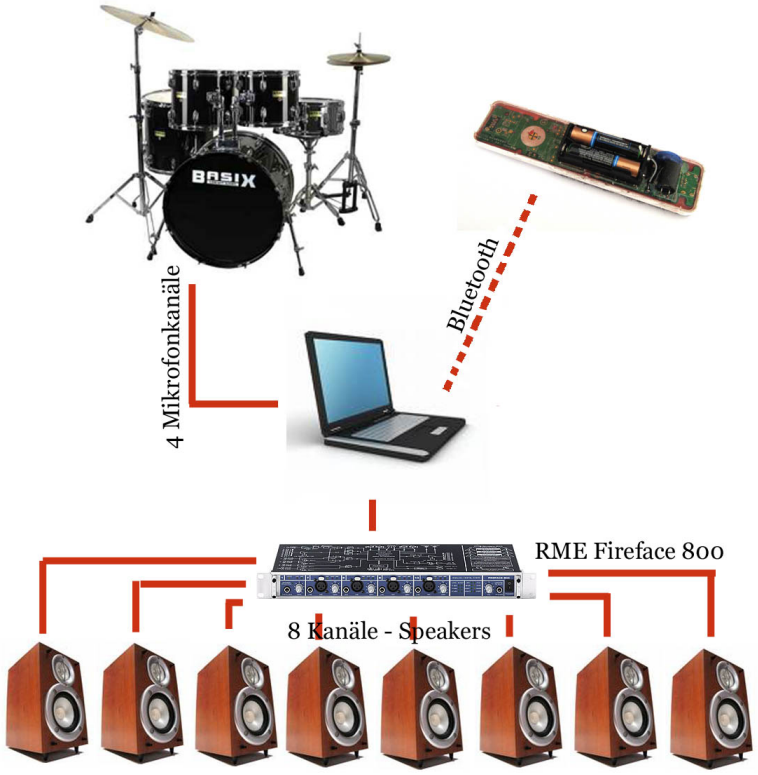


Figure 16: Gesamtsystem

Die Organisation im Rechner sah wie folgendermaßen aus:

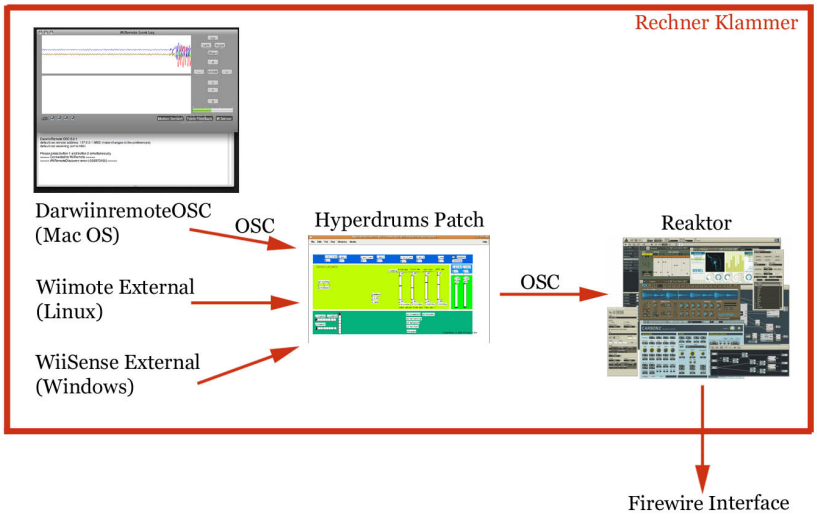


Figure 17: Signalverlauf

Die Verwendung des Sensors um Parameter wie Delayzeiten, Panning und Pitch von Oszillatoren zu kontRollieren, hat sich als sehr wirksam erwiesen.

Probleme:

- Pitch und Roll nicht immer eindeutig (siehe S. 6)
- Wii-Controller für Bedienung eines Charakters auf dem Bildschirm geschaffen. Kontrolle der realen Welt hat mehrere Freiheitsgrade.
- Für schnelle Bewegungen sind die Sensoren zu ungenau, bzw. die Datenrate zu gering. Für langsamere Bewegungen hat sich das System als sehr wirkungsvoll erwiesen.

6 Fazit

Mit dem Wiimote gibt es eine sehr einfache und günstige Möglichkeit verschiedenste Sensordaten kabellos an den Computer zu übermitteln. Eine annähernde Positionsbestimmung basierend nur auf den Beschleunigungssensoren ist allerdings nicht befriedigend, hierzu müsste man auf zusätzliche Sensoren zurückgreifen. Allein mit Verwendung der rohen Beschleunigungsdaten, der Rotationswinkel sowie der gefilterten Beschleunigungsdaten können jedoch eine Menge von Parameter für die Erweiterung des Schlagzeuges gewonnen werden. Interessant wäre in Zukunft noch eine Beschäftigung mit Gestenerkennung, was dem Schlagzeuger noch eine große Anzahl von Möglichkeiten bieten würde. In diesem Zuge möchte ich die *Wiigee* library für Java [3] erwähnen, die dafür sicherlich gut geeignet wäre.

Zu guter letzt möchte ich mich bei Herrn Prof. Ritsch und Herrn Josef Klammer für die gute Zusammenarbeit bei diesem interessanten Projekt bedanken.

Figure 18: Josef Klammer beim Konzert am IEM

References

[1] Yekaterina KHARITONOVA. Wii-trieve: Retrieving motion sequences using acceleration data of the wii controllers. Diplomarbeit, University of Hawaii, 2007.

[2] Steven M. LAVALLE. Planning algorithms. http://planning.cs.uiuc.edu/node102.html, Cambridge University, 2006.

[3] Benjamin POPPINGA. Beschleunigungsbasierte 3d-gestenerkennung mit dem wii-controller. Diplomarbeit, Universität Oldenburg, 2007.

[4] Craig Ranta. Human interface device (hid) profile version 1.0 adopted. `http://german.bluetooth.com/NR/rdonlyres/FEF51323-1EEC-49D6-A78F-ABF2277C202C/980/$HID_SPEC_V10$.pdf`, Mai 2009. 26. Mai 2009 um 13:34.

[5] Bluetooth Special Interest Group (SIG). Bluetooth core specifications v3.0 + hs. `http://www.bluetooth.com/Bluetooth/Technology/Building/Specifications/`, Mai 2009. 26. Mai 2009 um 12:57.

[6] Charles VERRON. Captation gestuelle pour une percussion augmentée. Paper, La kitchen, 2004.

[7] Mike WOZNIEWSKY, 2007. http://mikewoz.com/index.php?page=pd-stuff.

[8] Matt Wright. Opensound control specification. `http://archive.cnmat.berkeley.edu/OpenSoundControl/OSC-spec.html`, Mai 2009. 26. Mai 2009 um 13:49.