



Projektstudium, SS09

## Protokoll: OSC

Andreas Böhler

Update: 22. Juli 2009

FH-Linz  
Studiengang Medizintechnik

# Inhaltsverzeichnis

<b>1</b>	<b>Allgemeines</b>	<b>3</b>
<b>2</b>	<b>Aufbau der Nachrichten</b>	<b>4</b>
2.0.1	Mögliche „Endpfade“ WiiMote . . . . .	4
2.1	Nunchuk . . . . .	4
2.1.1	Mögliche „Endpfade“ Nunchuk . . . . .	4
2.2	BalanceBoard . . . . .	4
2.3	Statusmeldungen . . . . .	5
<b>3</b>	<b>Kommunikation mit dem Frontend</b>	<b>6</b>
<b>4</b>	<b>WiiMaze</b>	<b>7</b>
<b>5</b>	<b>ScrewDriver</b>	<b>8</b>

# 1 Allgemeines

Eigentlich kommt OSC (OpenSoundControl) aus dem Soundbereich. Es wurde als Protokoll zur Steuerung von Audiogeräten entworfen und ist als solches auf hohe Geschwindigkeit ausgelegt. OSC-Übertragungen bestehen aus Nachrichten, die von einem Sender an einen bestimmten Zielport gesendet werden können. Eine Nachricht besteht wiederum aus einem Pfad (z.B. der Pfad zu einer Steuereinheit) und daran angefügten Daten (z.B. die Einstellungen der Steuereinheit). Wir benutzen das OSC Protokoll wie folgt:

Server-Port MATLAB: 5600

Server-Port Python: 5601

Server-Port Flash: 3001

Server-Port WiiMaze: 5602

Die Messdaten der WiiMotes werden von einem Python-Script in OSC-Messages verpackt an MATLAB versendet. Die gleichzeitige Verwendung mehrerer WiiMotes und Nunchuks ist möglich, daher wurde das Protokoll entsprechend ausgelegt.

Das Python-Skript kann beim Start durch Kommandozeilen-Parameter konfiguriert werden

Parameter	Beschreibung
-i, -ip	Die IP-Adresse, auf der der OSC-Server laufen soll
-p, -port	Der Port, auf dem der OSC-Server laufen soll

## 2 Aufbau der Nachrichten

Der Pfad jeder Nachricht beginnt mit „/wii“ um zu signalisieren, dass Daten der WiiMote kommen. Anschließend wird die Nummer der WiiMote an den Pfad angehängt und abschließend die Steuergröße: „/wii/0/acc“ gibt den Pfad zu den Beschleunigungssensoren der ersten WiiMote an. „/wii/1/button/A“ ist nur vorhanden, wenn der A-Knopf der zweiten WiiMote gedrückt ist.

### 2.0.1 Mögliche „Endpfade“ WiiMote

Pfad	Beschreibung
acc	liefert im Datenbereich ein Tuple mit Sensordaten: [gx gy gz] im Bereich [0-255]. Diese Daten sind nicht kalibriert (Rohdaten!)
button/A, button/B, ...	liegen nur an, wenn der entsprechende Knopf gedrückt ist. Dann sind die Daten auf [1] gesetzt.
irdata	liefert die Koordinaten von bis zu vier getrackten IR-Punkten zurück. [ir1x, ir1y, ir2x, ir2y, ir3x, ir3y, ir4x, ir4y] in den Bereichen [(0-1023,0-767)]. Wird kein Punkt gefunden, liegt die y-Komponente auf 1023 (also außerhalb des eigentlichen Wertebereiches).

## 2.1 Nunchuk

Der Nunchuk ist mit dem Pfad der entsprechenden WiiMote verknüpft und setzt sich daher wie folgt zusammen:

„/wii/0/nunchuk/button/C“ ist nur vorhanden, wenn der C-Knopf des ersten Nunchuk gedrückt wird. „/wii/1/nunchuk/acc“ liefert die Beschleunigungsdaten des zweiten Nunchuks.

### 2.1.1 Mögliche „Endpfade“ Nunchuk

„button/C“, „button/Z“ sind nur vorhanden, wenn die entsprechenden Knöpfe gedrückt sind und enthalten den Datenwert [1]. „acc“ liefert als Tuple die Rohdaten der Beschleunigungssensoren im Bereich [0-255]: [gx, gy, gz]. „stick“ liefert die Rohdaten der Analog-Sticks im Bereich [0-255] als Tuple: [sx, sy].

## 2.2 BalanceBoard

Das BalanceBoard wird ähnlich einem Nunchuk unterstützt, da das BalanceBoard eine WiiMote mit Extensions-Controller emuliert. Die Pfade sehen wie folgt aus:

Pfad	Beschreibung
/wii/0/button/A	ist der einzige Knopf des BalanceBoards
/wii/0/bb/force	liefert ein Tuple der aktuellen Messdaten im Format [tr, br, tl, bl] als 2-Byte unsigned integer
/wii/0/bb/calib	liefert ein Tuple mit Kalibrationsdaten im Format [tr0,br0,tl0,bl0,tr17,br17,tl17,bl17,tr34,br34,tl34,bl34]

## 2.3 Statusmeldungen

Das Python-Skript sendet diverse Statusmeldungen per OSC.

Pfad	Beschreibung
/wii/devices	Sendet ein Tuple im Format ["Adresse", "Name"] über gefundene BT-Geräte
/wii/ping	Sendet ["OK"], falls das Frontend noch läuft
/wii/0/batterylevel	Sendet den Batteristand der gewählten WiiMote
/wii/0/status	Sendet den Status der gewählten WiiMote als Tuple, falls das entsprechende Feature aktiviert ist: ["Bat", "Ext", "Spk", "IR", "L1", "L2", "L3", "L4"]

### 3 Kommunikation mit dem Frontend

Das Python-Frontend arbeitet auf Port 5601 und erwartet folgende Nachrichten: „/wiipy“ ist die Adresse des Frontends mit folgenden Parametern:

Tuple	Beschreibung
["setup", 0, "debug", 1]	Setzt die Parameter für WiiMote der gegebenen Nummer
["connect", 0, "00:4f:4e:13:56:40"]	Baut eine Verbindung zur Adresse mit der gegebenen Nummer auf
["disconnect", 0]	Trennt WiiMote der gegebenen Nummer
["discover"]	Sucht nach Bluetooth-Devices und sendet das Ergebnis per OSC
["ping"]	Sendet nur ein simples Reply um zu testen, ob das Frontend noch läuft
["status"]	Liefert als Tuple [1, 1, 1, 1] zurück, wenn alle WiiMotes verbunden sind
["config", "ip", "127.0.0.1", "port", 5600]	Konfiguriert IP und Port, an die die OSC-Daten gesendet werden sollen

## 4 WiiMaze

Das Spiel WiiMaze kann ebenfalls per OSC gesteuert werden. Hierfür sind folgende Befehle an Port 5602 mit Pfad “/wiimaze” möglich:

Tuple	Beschreibung
["start"]	Startet ein neues Spiel
["position", xx, yy]	Bewegt den Cursor an Position xx/yy in Absolutwerten [0;1]
["level", ll]	Setzt das Level auf den gewählten Wert "ll"
["quit"]	Beendet das Spiel, lässt aber den OSC-Server laufen
["shutdown"]	Beendet Spiel und OSC-Server

## 5 ScrewDriver

Die Kommunikation mit dem ScrewDriver kann per Java-Kommunikator auf Port 3001 mit dem Pfad “/wiigame” erfolgen.

Mögliche Befehle:

Tuple	Beschreibung
["x", xx, "y", yy]	Bewegt den Cursor an Absolut-Positionen xx/yy [0;1]
["r", rr]	Rotiert die Schraube um den Winkel rr (in Grad [0;360])