Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Joscha Machatzke

Entwicklung eines kabellosen Sensor/Aktor Netzwerks
zur Steuerung einer verteilten Klanginstallation

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Joscha Machatzke

Entwicklung eines kabellosen Sensor/Aktor Netzwerks zur Steuerung einer verteilten Klanginstallation

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Birgit Wendholt
Zweitgutachter: Prof. Franziska Hübler

Abgegeben am 22.09.2014

Joscha Machatzke

Thema der Bachelorarbeit

Entwicklung eines kabellosen Sensor/Aktor Netzwerks zur Steuerung einer verteilten Klanginstallation

Stichworte

WSAN, kostengünstig, Arduino, nRF24l01+, interaktive Klanginstallation

Kurzzusammenfassung

Aus der Idee einer verteilten Klanginstallation, bei welcher der Besucher durch eine räumliche Bewegung Einfluss auf Klänge und musikalische Muster nehmen kann, wird eine kostengünstige Hardware-Software-Lösung entwickelt, die die Installation umsetzen lässt. Die Hardware-Software-Lösung umfasst ein kabelloses Sensor/Aktor Netzwerk, welches eine flexible Verteilung von Sensoren und Aktoren im Raum ermöglicht und zugleich den Aufbau auch in unterschiedlichen Umgebungen erleichtert. Zudem wird ein Software-Konzept entworfen, welches sowohl die Konfiguration von Sensoren und Aktoren als auch Klangsteuerung realisiert.

Title of the paper

Development of a Wireless Sensor and Actuator Network for controlling a distributed sound installation

Keywords

WSAN, low-cost, Arduino, nRF24l01+, interactive sound installation

Abstract

Based on the idea of an interactive distributed sound installation, in which the visitor's movement inside the room is taking influence on sounds and musical patterns, this work is concentrating on developing a supporting low-cost hardware-software solution, which is containing a Wireless Sensor and Actuator Network, which makes an easy setup of the installation and an adjustable arrangement of sensors and actuators in variable environments possible. Furthermore a software concept for configuration of sensors and actuators as well as sound controlling will be designed,

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Für diese Arbeit wurde eine begehbare auditiv interaktive Rauminstallation konzipiert, in welcher der Besucher sich auf die Suche nach Entstehung, Entwicklung und Zusammenhängen von Klängen und musikalischen Mustern machen kann und dabei maßgebend durch seine Bewegung im Raum Einfluss auf diese Klangwelt nimmt.

1.1 Motivation

Als Musiker und Klangliebhaber empfand ich es schon immer als faszinierend, wie man durch Auswahl und Ordnung von akustischen Signalen eine bestimmte Stimmung erzeugen kann. So hat mich bei der Produktion von Musik die Technik des Samplings inspiriert, bei welcher unterschiedliche ausgewählte Tonaufnahmen auseinander geschnitten und neu zusammengefügt werden, sodass eine komplett eigene Klangwelt entsteht.
Diese Arbeitsweise nutzend habe ich in der Vergangenheit mehrere Musikstücke unter anderem auch für Projekte im Bereich der „Digitalen Kunst“ komponiert, an welchen ich als Klangkünstler sowie Programmierer beteiligt war. Hauptauseinandersetzungspunkt dieser Arbeiten war die „Generative Kunst“, deren Hauptmerkmal die Verwendung eines Systems beliebiger Natur ausmacht, welches zu einem gewissen Grad selbstständig abläuft und zur Entstehung des Kunstwerks beiträgt, wie z.B. die Regeln und Prozesse eines Computer-Programms oder aber auch einer verbal verbreiteten Aufgabe an menschliche Teilnehmer (vgl. Galanter 2003, S. 4).
Beeindruckend an dem Thema finde ich, dass durch festgelegte Regeln und Ordnungssysteme aus einem Grundzustand heraus Muster entstehen, welche sich gefühlt unvorhersehbar entwickeln, sowie überraschend komplexe Ausmaße annehmen können und dadurch auf mich geradezu lebendig wirken.
Daraufhin beschloss ich, diese sonst eher im visuellen Bereich eingesetzten Prozessabläufe in den auditiven Kontext zu übertragen und die Resultate dessen für meine Mitmenschen durch eine interaktive Installation erfahrbar zu machen. Dabei stelle ich mir mehrere in einem Raum verteilte Schallquellen vor, die zusammen ein Klangbild erzeugen, welches der Besucher durch seine Bewegung im Raum interaktiv beeinflussen und erforschen kann, wobei der technische Hintergrund vor ihm verborgen bleiben sollte.
In Hinblick darauf weckte die Idee eines kabellosen Sensor/Aktor Netzwerks (WSAN; Wireless Sensor and Actuator Network) großes Interesse bei mir, da ein solches versteckt eingesetzt werden kann.
Es beschreibt ein Netzwerk von Knoten, welche eine Umgebung mittels Sensoren überwachen (z.B. das Lokalisieren von Menschen) und mittels Aktoren beeinflussen (z.B. Licht, bewegende Objekte oder Klang) und daher eine Interaktion zwischen Menschen und einer Umgebung möglich machen (vgl. Verdone 2008, S. 6). Der typische Aufbau eines Sensor-Knotens besteht neben den Sensoren aus einem Funkmodul, einem Mikroprozessor für Daten- und Nachrichtenverarbeitung sowie einer Energiequelle zum autonomen Betrieb (vgl. Matischek 2012, S. 1). Kabellose Sensor/Aktor Netzwerke finden vor allem im Bereich der Umweltüberwachung oder Industrie Anwendung, aber auch werden sie in der „Darstellenden Kunst“ eingesetzt. Dies zeigten beispielsweise die zwei Tanz-Performances *Schwelle* und *Chronotopia*, bei welchen mehrere am Körper angebrachte Sensoren die Bewegungen des Tänzers erfassten oder eine in der Bühnenumgebung installierte Lichtmatrix kabellos angesteuert wurde (vgl. Baalman 2010, S. 8-9).
Dass die fortschreitende Miniaturisierung von eingebetteten Computersystemen eine Vielzahl neuer Anwendungsbereiche ermöglicht, typisiert den Ausdruck „Pervasive Computing“ (vgl. Saha

2003, S. 25-31). Gerade diese Entwicklung brachte Plattformen hervor, die dem Entwickler nicht nur umfangreiche Dienste anbietet, sondern auch die Realisierung von günstigen und zugleich effektiven Systemen erleichtert (vgl. Denegri 2007, S. 1):

> *„Sensing devices provide pervasive systems with information such as the locations of people [...]. The system can use this information to interact more naturally with users“ (Saha 2003, S. 25-26)*

Ein kabelloses Sensor/Aktor Netzwerk kann dabei ein solches System bilden.

1.2 Zielsetzung

Zur Umsetzung der Klanginstallation soll ein kabelloses Sensor/Aktor Netzwerk entwickelt werden, welches eine im Raum verteilte Anordnung von variierbar vielen Klangkörpern erlaubt und gleichzeitig mittels Sensoren eine Interaktion mit der Installation ermöglicht. Netzwerkknoten sollen deshalb sowohl mit Sensoren als auch Aktoren ausgestattet werden können und in ihrer Anzahl anpassungsfähig sein. Die zu entwickelnde Lösung soll dabei nicht für fertig konfigurierte Knoten entworfen werden, sondern muss flexibel für verschiedenste Sensorik und Aktorik verwendet werden können.
Da zu Beginn der Installation noch viele Frage offen stehen, wie z.B. „Welche Reaktion soll auf welche Bewegung des Besuchers folgen?“, „Welche Sensoren erfassen dieses Verhalten am Besten?“, „Welchen Einfluss hat dabei die Anordnung der Klangquellen auf den Besucher?“, soll in dieser Arbeit eine Plattform entwickelt werden, auf welcher die Installation aufbauen und praktisch entwickelt werden kann. Darüber hinaus soll diese Plattform so anpassungsfähig sein, dass sie auch künftig in anderen Projekten einsetzbar ist.
Für die Errichtung eines solchen kabellosen Sensor/Aktor Netzwerks gilt es daher eine Hardware zu suchen und zu entwickeln. Diese Hardware muss frei programmierbar sein und sollte, da sie in einer Vielzahl vorkommen kann, möglichst kostengünstig sein. Sie sollte ebenfalls autonom durch eine eigene Stromversorgung betrieben werden können, um einen leichten, räumlich uneingeschränkten Aufbau der Installation zu begünstigen. Auch soll eine Firmware entwickelt werden, welche die Hardware sowohl netzwerkfähig als auch erweiterbar macht und eine Ansteuerung dieser von außerhalb des Netzwerks erlaubt.
Für die Steuerung und Konfiguration der Hardware sowie für die Klangsteuerung soll eine Software oder ein geeignetes Werkzeug entwickelt werden. Die Klangsteuerung soll konfigurierbar sein, sodass über Sensordaten und Regeln Klangmuster erzeugt werden können.

1.3 Gliederung

Die Arbeit gibt im nachfolgenden Kapitel zunächst eine genauere Beschreibung der Klanginstallation. Darauf aufbauend ergeben sich in Kapitel 3 die Anforderungen, welche an die technische Seite der Installation gestellt werden. In Kapitel 4 wird ein vergleichbares Projekt, welches ein kabelloses Sensor/Aktor Netzwerk als Hardware-Software-Lösung für Live Performances und interaktive Umgebungen bietet, vorgestellt und mit dem im Rahmen dieser Arbeit entstanden Lösungsentwurf verglichen. Kapitel 5 Lösungsentwurf beschreibt die Software-Architektur, erläutert das Nachrichtenprotokoll, welches die Architekturkomponenten zur Kommunikation untereinander nutzen, und gibt eine Hardware und Software Lösung zur

Realisierung der Installation an. In Kapitel 6 werden die wesentlichen Aspekte dieser Arbeit zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen dieser Arbeit gegeben.

2 Die Klanginstallation

Die Klanginstallation besteht aus mehreren in einem leeren, abgedunkelten, großen Raum verteilten Klangkörpern, durch und in deren Anordnung man sich bewegen und sie so zum Klingen erwecken kann. Dabei soll die Erfahrung weniger an das Spielen eines Instruments erinnern, sondern sich vielmehr auf das Wahrnehmen und Erforschen bestimmter Klänge und Klangmuster konzentrieren. Unterstützend für dieses auditive Erlebnis wird auch eine entsprechende visuelle und interaktive Umwelt aufgebaut.
In den folgenden Kapiteln werden die drei Einflussgrößen Akustik, Optik und Interaktion beschrieben, um die Idee der Klanginstallation genauer vorzustellen.

2.1 Akustik

Die Wahrnehmung von Schall durch das Ohr wird von vielen Faktoren beeinflusst und bietet die Grundlage der Psychoakustik. So verursachen beispielsweise Klänge, welche sich außerhalb der Hörfläche befinden, ein unbehagliches Gefühl. Infraschall zum Beispiel kann bei Menschen neben psychischen Auswirkungen wie Nervosität und Angst auch zu körperlichen Beschwerden wie Schwindel und Übelkeit führen, wie bei einem Experiment in London im Jahre 2003 herausgefunden wurde (vgl. Angliss 2003). Aber auch bestimmte Tonfolgen und Tempi können die Wahrnehmung auf unterschiedlichste Art beeinflussen: Schnelle unvorhersehbare Tonfolgen können als als hektisch empfunden werden und die Konzentration des Zuhörers auf sich lenken, wohingegen langsame, gleichmäßige und vorhersehbare Tonfolgen (wie bei Einschlafliedern) zu einer beruhigenden Wirkung tendieren.
In der geplanten Klanginstallation soll sich der Besucher in einer möglichst von der Außenwelt abgeschnittenen Atmosphäre wiederfinden, in welcher er ohne allzu große Ablenkung in eine Welt der Klänge eintauchen kann. Alle visuellen Reize werden dort also reduziert.
Die Stimmung wird durch eine bestimmte Art von Klängen und eine genau zugeschnittene Komposition gelenkt.

2.1.1 Klang

Für die Installation möchte ich eine geheimnisvolle Atmosphäre kreieren, in welcher der Besucher wachsam umherwandern kann. Deshalb habe ich mich auf der auditiven Ebene für reduzierte Klänge entschieden, die sich nach und nach aufbauen, sodass der Besucher mit jedem Schritt in der Tiefe der Klanginstallation Neues entdecken kann. Dabei fängt die Reise des Besuchers mit einem kaum hörbaren, leisen, dumpfen, an ein Uhrwerk erinnernden Klacken an und entwickelt sich über vereinzelnd wahrzunehmende intensive, klare Klänge bis hin zu einem sich über dem Raum ausbreitenden Klangteppich.
Zur Erzeugung dieser Klänge verwende ich zum Einen Klangstäbe (auch Chimes genannt), welche bspw. für meditative Zwecke eingesetzt werden können, um den Anfang und das Ende einer Meditationsphase kenntlich zu machen.[1] Sie geben einen hohen, klaren und langanhaltenden Signalton wieder, welcher den Hörer zur Aufmerksamkeit auffordert. Diese Klangeigenschaft möchte ich mir in der Installation zu Nutze machen, um die Konzentration des Besuchers verstärkt auf die Klänge zu leiten.

1 vgl. http://www.chimes.com/p-375-zenergy-meditation-chime.aspx

Zum Anderen werden, um diese Klangstäbe zum Klingen zu bringen, mechanische Anschläge verwendet (im Detail siehe 5.4.1.1.2). Diese entlocken nicht nur den Klangstäben ihren klaren Klang, sondern tragen durch ihre kaum hörbaren, leisen, dumpfen, mechanischen Geräusche selbst zum Klangbild der Installation bei und spannen so eine geheimnisvolle zweite Klangebene auf.
Bei der Installation habe ich mich für eine reale Klangerzeugung entschieden. Eine Erzeugung von Klängen durch viele im Raum verteilte Lautsprecher kommt für mich nicht in Frage. Mir ist es wichtig, dass die Entstehung des Klangs möglichst nah beim Besucher stattfindet und vor Allem, dass der Besucher dies bemerkt, um sich selbst als Teil dieser Klangumgebung identifizieren zu können.
Da Lautsprecher lediglich eine bestimmte Tonaufnahme oder ein (künstlich) erzeugtes Signal wiedergeben, dessen Klangqualität zuletzt nicht nur von dem Signal selbst, sondern auch von der physikalischen Beschaffenheit der Lautsprecher abhängt, ließe sich möglicherweise darauf schließen, dass erstens der Ursprung des erzeugten Signals in einem anderen Ort als dem der Installation liegt und zweitens eine Erzeugung des Signals bereits stattfand.
Eine reale Klangerzeugung hingegen findet tatsächlich in der Gegenwart und in unmittelbarer Nähe des Besuchers statt. Sie schafft eine besondere, nur für ihn erzeugte Stimmung, die es dem Besucher ermöglichen soll in die Klangwelt einzutauchen.

2.1.2 Komposition

In der Musik legt die Komposition den musikalischen Ablauf eines Stücks fest, bestimmt etwa Tonhöhen, -längen, Tempo und Rhythmen und weist diese als Noten gegebenenfalls mehreren Stimmen zu.
Im Gegensatz zur musikalischen Komposition, wo jede Note ihren genauen Platz findet, soll die Installation einer eher variablen Komposition folgen, dessen Ablauf und Notenbild von Besuchern beeinflusst werden kann. Dabei soll jedoch nicht wie bei Musikinstrumenten üblich eine direkte Übertragung zwischen Spielen des Instruments und der Erzeugung eines Klangs durch das Instrument erkennbar sein (wie z.B. der Zuordnung einer Klaviertaste mit einem bestimmten Ton). Vielmehr stelle ich mir ein Klangbild vor, das zwar durch Interaktion mit der Installation erzeugt und beeinflusst wird, dessen Entstehungsweise dem Besucher jedoch zunächst nicht eindeutig ist. Erst durch Erforschen und Erfahren der Installation sollen gewisse Abhängigkeiten und Regeln deutlich werden.
Inspiriert von der *fraktalen Geometrie* (vgl. Mandelbrot 1991), bei welcher durch Gesetzmäßigkeiten und das Anwenden dieser auf sich selbst geometrische Formen erzeugt werden können, die auf Grund ihrer Selbstähnlichkeit an jeder Stelle und jeder Tiefe/Vergrößerung gleiche (oder zumindest sehr ähnliche) Strukturen aufweisen, stelle ich mir eine Komposition vor, welche durch solche Regeln beschrieben werden kann.
Ein visuelles Beispiel fraktaler Formen aus dem Bereich „Generative Kunst" ist das Werk *EDF0* von *Raven Kwok* (vgl. http://ravenkwok.com/edf0/). Abbildung 1 zeigt einen Ausschnitt daraus[2]:

2 Bei Abb.1 handelt es sich um einen Screenshot aus der Processing Anwendung; vgl. http://www.openprocessing.org/sketch/116516. Ein detailreicheres Video ist hier zu finden: http://vimeo.com/43752422

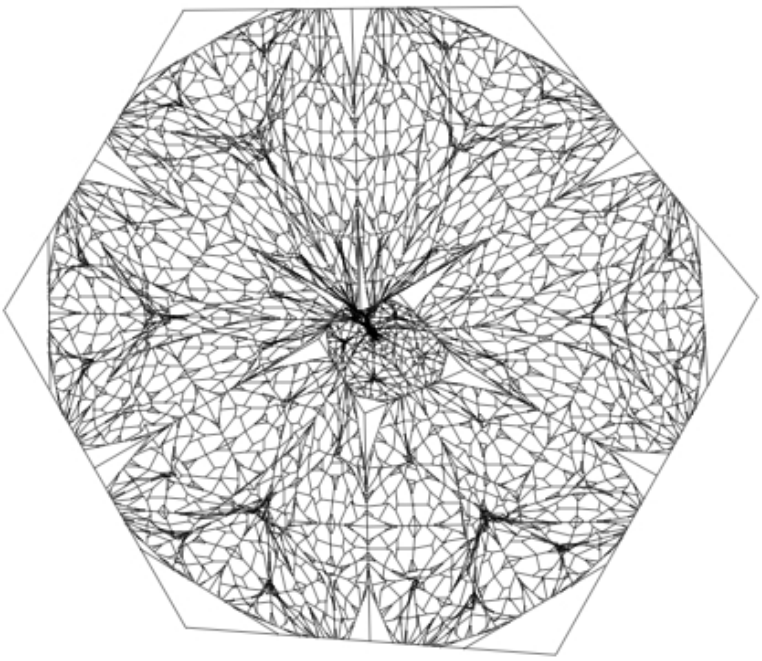

Abb. 1: Ausschnitt aus Raven Kwok's EDF0

Was hier als Standbild abgebildet wird, ist in Wirklichkeit ein sich ständig wandelnder Organismus, dessen Form sich über die Zeit verändert. Zu erkennen ist die Selbstähnlichkeit, die in der gesamten Struktur wiederzufinden ist. So kann diese in sechs Bereiche aufgeteilt werden, deren Muster starke Ähnlichkeiten sowohl untereinander als auch innerhalb eines einzelnen Bereichs aufweisen.
Beeindruckend an dieser Arbeit finde ich nicht nur die komplexe Form, die dieses System annimmt, sondern vor Allem den Wandlungsprozess, der über die Zeit stattfindet und an einen lebenden Organismus erinnert. Dies auf Musik übertragen könnte z.B. bedeuten, dass ein bestimmtes Melodiemuster ständig wiederholt, aber in leichter Abwandlung vorkommt. Eine Veränderung nach jedem Durchlauf wäre möglicherweise kaum bemerkbar, doch über einen längeren Zeitraum betrachtet könnte so ein aus sich selbst wachsendes Klangbild erzeugt werden.
Die im Zuge dieser Arbeit entstehende Installation stellt ein Experiment dar, solche durch Regeln erzeugten Muster in Klang zu übertragen und für Besucher hörbar zu machen. Der Besucher ist dabei Teil dieses Systems und kann durch seine Bewegung im Raum Klangmuster beeinflussen.

2.2 Interaktion

Auf seiner Entdeckungsreise durch die Installation wird der Besucher durch seine Bewegung im Raum zu einem aktiven Part für die Klangentfaltung. In der Ausgangsphase, wenn kein Besucher die Installation betreten hat, herrscht zunächst Stille. Beim ersten vorsichtigen Betreten erklingen dann vereinzelt reduzierte Klänge, die eine eigene geheimnisvolle Atmosphäre erzeugen. Je weiter der Besucher in die Installation tritt, desto mehr taucht er in die dort auf ihn wartenden Klangebenen ein.

2.3 Optik

Um die geheimnisvolle Atmosphäre zu unterstreichen, wird die Installation in einem nur schwach beleuchteten leeren Raum aufgebaut, in welcher sich der Besucher zwar angstfrei fortbewegen kann, sich aber in einer Welt fern ab von visuellen Reizen wiederfindet, sodass es ihm besser möglich ist, sich auf die auditiven Signale zu konzentrieren.
Die einzigen nur andeutungsweise zu erkennenden Objekte sind die Klangkörper, die sich von der Decke hängend mit einem Abstand von ca. 50cm über dem Besucher befinden. Diese bestehen aus einem Holz-Korpus und Klangstab aus Aluminium, haben ungefähr die Maße 15cm x 5cm x 10cm (Länge-Breite-Höhe) und werden je nach Größe und Beschaffenheit des Raums zu einer Anzahl von mindestens 20 Stück aufgehangen. Die Abstände der Klangkörper zueinander betragen ca. einen Meter.

2.4 Fazit

Konkrete Entscheidungen, die zum Einen vorgeben, was die Klanginstallation ausgibt (auditiv und visuell) und zum Anderen was und wie sie Änderung der Umgebung aufnimmt (interaktiv), sollen an dieser Stelle noch nicht festgelegt werden. Denn theoretisch lassen sich schwer Vermutungen über das Zusammenwirken zwischen Klangkomposition und Interaktion oder gar über ein Stimmungsbild der Installation ausmachen. Dies sollte praktisch erforscht werden und Anlass zum Experimentieren geben. Nur durch das gemeinsame Erfahren aller Einflüsse, die die Installation bestimmen, und deren Wechselwirkung zu einander, kann die Installation, wie sie der Besucher zur Ausstellung erfährt, am Besten gestaltet werden. Nur ein voriges Erforschen der Installation, kann auch dem Besucher ein interessantes Erforschen von Klängen und Klangmustern ermöglichen.

3 Anforderungsspezifikation

Die Klanginstallation soll so umgesetzt werden, dass sie den in Kapitel 2 Die Klanginstallation beschriebenen Vorstellungen entspricht und gleichzeitig genügend Freiraum bietet, um Änderungsmöglichkeiten flexibel und auch zu einem späteren Zeitpunkt zu erlauben. Dazu werden im Folgenden Anforderungen an die technische Seite der Umsetzung aufgestellt.
Kapitel 3.1 stellt die Anforderungen an das Netzwerk und 3.2 die Anforderungen an die Klangsteuerung. In Kapitel 3.3 werden nicht funktionale Anforderungen beschrieben und in 3.4 die Ergebnisse zusammengefasst.

3.1 Netzwerk

Die Installation besteht aus mehreren verteilten Klangkörpern, die ein vom Besucher beeinflussbares Klangbild erzeugen. Daher muss ein System entwickelt werden, welches es erlaubt die Umgebung zu überwachen, um Bewegungen des Besuchers aufzunehmen, und gleichzeitig durch Ausgabe von Klängen zu beeinflussen. Es soll dabei eine verteilte und möglichst flexible Anordnung realisiert werden können, die beliebig viele Klangkörper erlaubt. Deshalb scheint es sinnvoll ein kabelloses Netzwerk zu entwickeln, welches die Möglichkeit bietet Netzwerkknoten mit beliebiger Sensorik und Aktorik auszustatten. Es ist nicht denkbar, dass Knoten selbstständig, unkoordiniert laufen; sie müssen als Einheit funktionieren und können nur als Netzwerk verteilter Knoten die Umsetzung der Installation ermöglichen. Ein Netzwerk ist aus mehreren Gründen notwendig:
Die Qualität der Interaktion wird beispielsweise dadurch bestimmt, wie gut, schnell und zuverlässig die Sensoren funktionieren; also die Schnittstelle zwischen Benutzer und Installation. Bei der Installation handelt es sich um eine Anhäufung von mehreren Sensoren, die Bewegungen des Besuchers ermitteln. Einige Sensoren funktionieren in bestimmten Anordnungen nicht ohne Koordination, so z.B. wenn mehrere zur Distanzmessung verwendeten Ultraschallsensoren dicht beieinander platziert sind. Diese senden nämlich ein Pulssignal im Ultraschallbereich (typ. 42 kHz) aus und nutzen die Zeit vom Senden des Signals bis zum Empfang des durch Reflexion entstanden Echosignals um die Entfernung zum nächst gelegenen Objekts zu ermitteln. Bei vielen Ultraschallsensoren auf engem Raum kann das ausgesandte Schallsignal auch von anderen Sensoren empfangen werden und als das eigens ausgesandte Signal gedeutet werden. Diese Fehlinterpretation führe kontinuierlich zu falschen Messergebnissen. Deshalb wäre hier eine Koordination dieser Sensoren erforderlich. Sie könnte zum Beispiel in Gruppierung von Sensoren erfolgen, die sich beim Messvorgang nicht gegenseitig stören können. So wären Messungen innerhalb einer Gruppe unbedenklich. Gruppen könnten dann in regelmäßigen Abständen nacheinander messen.
Auch wäre die Umsetzung einer Klangkomposition ohne Netzwerk nicht möglich. Sicherlich könnte ein einzelner Knoten auf seine Sensorwerte reagieren und daraufhin bestimmte Klänge erzeugen. Dies böte sicherlich ein interessantes Schauspiel bei mehreren Knoten, allerdings wäre eine spätere Änderungen und Implementierung von Regeln (zur Änderung der Klangkomposition) sehr aufwändig, da sie stets auf jeden Knoten einzeln angewandt werden müsste. Auch ist nicht zwangsläufig gesagt, dass sich Sensoren und Aktoren am selben Ort befinden und von einem Knoten gesteuert werden können. Um der Flexibilität der Installation gerecht zu werden, sollten daher Sensorik und Aktorik von unterschiedlichen Knoten betrieben werden können. Dies wiederum erfordert eine Kommunikation mit Knoten und macht ein Netzwerk notwendig.

Des Weiteren soll die Installation eine gemeinsame Komposition aus unterschiedlichen Klangkörper erlauben. Das bedeutet, dass sich mehrere Klangkörper eine Melodie „teilen“ und erfordert einen gemeinsamen Takt oder irgendeine Art der zeitlichen Synchronisierung. Selbst wenn einzelne Knoten mit Melodien oder Klangregeln programmiert wären, die dem selben Tempo folgen, würden diese Knoten im Laufe der Zeit auseinander driften, da der Prozessortakt einzelner Knoten nie exakt der Selbe wäre und auch Verzögerungszyklen (delay) oder unerwartete Interruptroutinen den „normalen“ Ablauf verschieben könnten. Schon alleine ein gemeinsamer Taktbeginn wäre ohne zeitliche Abstimmung undenkbar. Deshalb soll die Klangkomposition nicht innerhalb eines Knotens geschehen, sondern über ein Netzwerk, welches mittels Anfragen den Zugriff auf Sensordaten und das Auslösen von Klängen ermöglicht, umgesetzt werden können.

3.1.1 Kabellos

Damit Einschränkungen in der visuellen Erscheinung so gering wie möglich gehalten werden, sollte das Netzwerk kabellos betrieben werden. Dies ermöglicht eine anpassungsfähigere Anordnung der Klangkörper und Sensorknoten und erleichtert eine Installation auch in unterschiedlichsten Umgebungen. Es erfordert jedoch auch eine unabhängige Spannungsversorgung und eine kabellose Kommunikationseinheit.

3.1.2 Knoten

Ein Knoten soll die Möglichkeit bieten die Umgebung mittels Sensor(en) zu erfassen oder mittels Aktor(en) zu beeinflussen. Eine Änderung der Sensorwerte soll eine Änderung der Klangkomposition hervorrufen und letztlich einen oder mehrere Klangstäbe in ausgewählten Knoten anschlagen lassen. Knoten sind daher mit Sensoren oder Klangstäben bestückt und können deshalb die Rolle eines Sensor- oder Aktorknotens einnehmen. Sie sollten um ihre Fähigkeit(en) wissen (z.B. wieviele Klangstäbe gespielt werden können) und sollten Dienste anbieten um diese zu bedienen. Zur Vereinfachung können alle Sensor- bzw. Aktorknoten mit der selben Anzahl an Sensoren bzw. Klangstäben ausgestattet werden. Sensortypen bzw. Tonhöhen der Klangstäbe sollen je nach Knoten variieren können.
Da sie kabellos ins Netzwerk integriert werden, sollte eine Spannungsversorgung mittels Batterie/Akku und ein Datenaustausch via Funk erfolgen.
Knoten kommen in einer Vielzahl im Netzwerk vor und sollten deshalb kostengünstig und, da sie kabellos sind, sowohl energieeffizient arbeiten, als auch energieeffiziente Hardware nutzen. Aus diesem Grund sollten sie so einfach wie möglich aufgebaut sein, wenig Strom verbrauchen und so wenig Arbeit wie nötig verrichten. Es scheint so nicht sinnvoll sie selbst mit der Aufgabe der Koordination von Sensormessungen oder etwa dem Auffordern anderer Knoten zur Klangauslösung aufgrund von eigenen oder gar fremden Sensordaten zu überlassen, sondern mittels einer übergeordneter Kontrolleinheit zu koordinieren.
Nach dem Einschalten eines Knoten soll dieser sich im Netzwerk anmelden, d.h. er teilt mit, dass er aktiv ist und welche Dienste er anbietet. Da jeder Knoten eine beabsichtigte Position in der Installation einnehmen könnte, sollte er sich mit einer (vorweg festgelegten) eindeutigen Nummer anmelden können. Diese Nummer soll einen Knoten identifizieren lassen und eine spätere Zuordnung erleichtern (z.B. beim Abbilden von Klangstäben auf Noten). Ist ein Knoten angemeldet, sollte er anschließend lediglich auf Anfrage bestimmte Dienste ausführen: Er wartet

in einem energiesparenden Zustand auf neue Tasks, führt diese aus und wechselt dann in den energiesparenden Zustand zurück.
Knoten sollen eine Schnittstelle für Dienste bereitstellen, die folgende Funktionalitäten implementieren lassen:

- Bekanntgabe der Schnittstelle für Knotendienste
- Zurücksetzen eines Knotens (Neustart des Knotens, Setzen auf Standardwerte und Auslösen einer Neuanmeldung im Netzwerk)
- Sensorik
  - Messen unabhängig vom Sensortyp
  - Konfigurieren der Sensoren
  - Einstellen von Messparametern (z.B. automatisches oder manuelles Messen, periodisches Mitteilen von Messwerten)
- Aktorik
  - Auslösen von Klängen/Anschlagen von Klangstäben
  - Konfigurieren von Aktoren
  - evtl. Ansteuerung visueller Effekte

3.1.3 Selbstorganisation des Netzwerks

Zur einfacheren Handhabung und Minimierung des Installationsaufwandes sollte das Netzwerk fehlertolerant sein, d.h. es sollte in bestimmten Ausnahmesituation funktionsfähig bleiben und sich mittels automatisierter Abläufe selbst organisieren um ein aufwendiges, manuelles Eingreifen zu ersetzen. Die Selbstorganisation dieses Netzwerks kann für zwei Ausnahmefälle reduziert werden: ein neuer Knoten kommt hinzu oder ein bereits bestehender Knoten fällt weg.
Der erste Fall tritt ein, wenn ein Knoten eingeschaltet oder zurückgesetzt wird. Dieser sollte sich dann automatisch im Netzwerk anmelden und in der Netzwerksicht sichtbar werden. Der Knoten wird dann *aktiv*, wenn er in der Netzwerksicht integriert wurde, und soll dann erst für weitere Zwecke verwendet werden können.
Fällt ein Knoten weg (bspw. im Falle einer leeren Batterie), muss der „tote“ Knoten in der Netzwerksicht als *inaktiv* markiert und auf das Problem aufmerksam gemacht werden. Die Tonausgabe kann aus diesem Knoten nicht mehr stattfinden, die Wiedergabe der restlichen Klangkörper allerdings schon: Dazu muss zum Einen in der Notenzuweisung verhindert werden, dass Noten nicht an inaktive Knoten weitergeleitet werden. Zum Anderen sollte für alle von diesem Knoten abhängigen Eingabedaten (z.B. Sensordaten von Knoten x) ein Standard-Wert greifen, z.B. der letzte Sensorwert oder (in mathematischen Funktionen oftmals am wenigsten dramatisch) die „1“. Der Knoten sollte dann, da er ja zu einem bestimmten Klangszenario gehört, manuell neugestartet werden und sich automatisch reintegrieren/aktivieren. Andernfalls müssen, wenn nicht mehr erwünscht, alle Abhängigkeiten manuell aus dem Klangszenario entfernt werden. Ein automatisches Löschen vorhandener Abhängigkeiten ist auch denkbar, wenn die Eingabewerte, die durch diesen Knoten erhalten wurden, durch Konstanten ersetzt werden.

3.1.4 Benutzerschnittstelle

Für das Netzwerk sollte eine Benutzerschnittstelle bereitgestellt werden, die zur Übersicht, Konfiguration und Steuerung des Netzwerks dient.

Dabei sollte eine Art Übersichtskarte einen schnellen Überblick über das Netzwerk geben und Knoten (ihrer realen Position nachempfunden) positioniert werden können. Denkbar wäre, dass jeder Knoten einen Standarddienst bereitstellt, mit dem er visuell oder auditiv auf sich aufmerksam machen kann; dies würde die reale Zuordnung eines Knotens erleichtern. Die Übersichtskarte sollte zudem den aktuellen Zustand eines Knotens anzeigen, d.h. sichtbar machen, ob ein Knoten „neu“, „aktiv“ oder „inaktiv“ ist.
Die Benutzerschnittstelle sollte also auch anzeigen, wenn neue Knoten hinzugekommen sind, aber auch darauf aufmerksam machen, wenn ein Knoten ausgefallen ist. Außerdem soll sie Informationen über einzelne Knoten anzeigen (z.B. welche Dienste der Knoten anbietet) und Dienste der Knoten testweise anfordern können.
D i e Benutzerschnittstelle sollte zudem Einstellmöglichkeiten für Sensoren und Aktoren bereitstellen. Dies sollte für Sensoren z.B. das Ändern der Periodendauer bei automatischen Messzyklen, aber auch - im Falle von Sensoren, die sich gegenseitig stören können - eine geeignete Koordination und Gruppierung von Sensoren erlauben. Für Aktoren sollte die Anschlagsdauer eines bestimmten Klangstabs einstellbar sein.

3.2 Klangsteuerung

Das Netzwerk stellt das Grundgerüst dar, in dessen Rahmen die beschriebenen Vorstellungen der Klanginstallation umgesetzt werden können: Messdaten werden von Sensorknoten erfasst, weitergeleitet und durch Regeln und Abläufe als Klangmuster von bestimmten Aktorknoten umgesetzt. Dieser Ablauf muss in einer Steuereinheit stattfinden, die die Brücke zwischen Sensoren und Aktoren bildet. Sie kann als Blackbox gesehen werden, dessen Funktionsweise der Besucher erforschen und erkunden kann (s. Abb. 2):

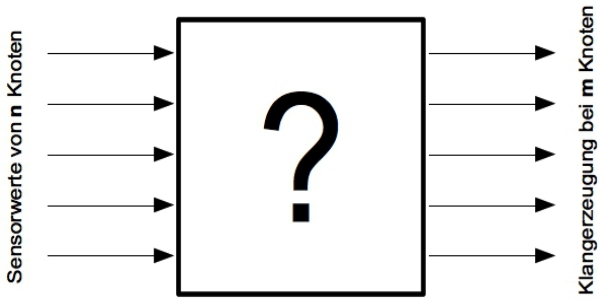

Abb. 2: Steuereinheit als Blackbox

Diese Steuereinheit übernimmt dabei zwei Aufgaben: Einerseits entscheidet sie auf Grund von Sensordaten und Regeln, welcher Klangstab wann gespielt wird. Andererseits muss sie dafür Sorge tragen, dass die jeweiligen Aktorknoten die beabsichtigten Klangstäbe zur beabsichtigten Zeit anschlagen. Der Weg zwischen Eingabe und Ausgabe findet demnach in diesen Schritten statt: der Kompostion und der Klangauslösung.
Typischerweise gibt eine Komposition durch Festlegung von Noten den Ablauf eines Stückes vor; eine Note repräsentiert dabei eine bestimmte Tonhöhe (z.B. a' bzw. A4 mit 440,0Hz). Entgegen der konventionellen Bedeutung repräsentiert im Folgenden der Begriff *Note* einen ihr zugewiesenen Klangstab. Eine Note gibt somit nicht nur eine Tonhöhe, sondern (vor Allem) den Ort des Klangs vor. Dies bedeutet allerdings auch, dass beispielsweise zwei Klangstäbe, die auf die selbe Tonhöhe gestimmt sind, von zwei unterschiedlichen Noten beschrieben werden.

Da die Installation über eine feste Anzahl an Klangstäben verfügt, kann von der selben Anzahl an Noten ausgegangen werden. Dabei sollte in der Klangsteuerung festgelegt werden können, welche Note welchen Klangstab repräsentiert; oder genauer gesagt welcher Knoten, welche Noten spielt. Gleiches gilt für die Sensoren: Es kann von einer festen Menge an Sensoren ausgegangen werden, die die Komposition zum Erzeugen der Noten nutzt. In der Klangsteuerung soll daher auch angegeben werden, welche Sensoren (von welchen Knoten) für die Komposition genutzt werden können.
Bei Hinzukommen bzw. Wegfallen von Knoten sollten die zugehörigen Sensoren und Aktoren aktiviert bzw. deaktiviert werden. Die Klangsteuerung sollte nur Dienste für Sensoren oder Aktoren anfordern/nutzen, dessen Knoten aktiv sind. Fällt ein Knoten während des Betriebs aus, sollte die Klangsteuerung außerdem Maßnahmen treffen, die die Installation (wenn auch eingeschränkt) weiter laufen lässt, so dass dem Besucher der Ausfall des Knotens weitestgehend verborgen bleibt.
Zusammenfassend vereint die Klangsteuerung die Schritte Komposition und Klangauslösung, und realisiert somit die endlose Schleife zwischen Besucherinteraktion und Klangerzeugung. Die Klangsteuerung soll zudem auf die beschriebenen Ausnahmefälle geeignet reagieren (vgl. Selbstorganisation des Netzwerks).

3.2.1 Komposition

Das Erzeugen von Noten findet durch die Komposition statt. Dabei soll es sich um eine Art variable, beeinflussbare Komposition handeln, die zwar gewissen Regeln folgt, sich aber stets (durch Interaktion) neu zusammenfügen kann (vgl. Kapitel 2.1.2).
Die Komposition geschieht dabei auf Grund von Sensordaten, aus denen regelhaft Notenmuster erzeugt werden. Diese Regeln müssen sich auf ein Regelwerk beziehen, welches sowohl die Eingabe- (Sensordaten) als auch die Ausgabeseite (Notendaten) bedienen kann. Da diese Größen typischerweise durch Zahlen beschrieben werden, sollen die Regeln mit Hilfe mathematischer Rechenoperatoren und Operatoren imperativer Programmierung definiert werden können. Dazu zählen:

- Arithmetische Operatoren (Addition, Subtraktion, Multiplikation, Division, Modulo)
- weitere Rechenarten (Potenzieren, Logarithmieren, etc.)
- elementare mathematische Funktionen (z.B. Sinus)
- weitere Funktionen (Rausch- und Zufallsfunktion)
- Kontrollstrukturen (Verzweigung und Schleifen)
- Vergleichsoperationen
- Bool'sche und binäre Operatoren

Mit diesen Regeln soll die Komposition in Abhängigkeit zu den Eingabegrößen einen zeitlichen Ablauf von Noten definieren können. Außerdem soll in der Komposition, da es sich um ein beabsichtigtes Muster (z.B. fraktale Strukturen) und nicht eine rein zufällige Reihenfolge von Klängen handelt, die Erzeugung von zeit-/tempo- und taktabhängigen Strukturen möglich sein.
Da die Komposition ein Klangbild und sogleich ein bestimmtes Stimmungsbild vorgibt, welches möglicherweise gar nicht zum Charakter des Klangs (oder dem visuellen Bild der Installation, etc.) passt, sollte es auch möglich sein die Komposition und/oder gewisse Notenfolgen/-muster an der laufenden Installation ausprobieren zu können, bevor der Besucher die vollendete Installation betritt - eine Art „live“ Komponieren. Die Komposition darf dabei nur die in der Klangsteuerung angegebenen Sensoren und Noten verwenden.

Da die Installation bei Ausfall von Knoten weiterlaufen soll, muss auch die Komposition im Ausnahmefall weiterhin funktionieren. Es kann davon ausgegangen, dass die Komposition nur von Sensoren nicht aber von Aktoren abhängt. Zwar erzeugt sie Noten, die bei Ausfall eines Aktorknotens nicht mehr verfügbar wären, jedoch findet das Auslösen der ihnen zugewiesenen Klangstäbe an anderer Stelle statt. Ein Ersatz der ausgefallenen Noten durch verfügbare Noten steht außer Frage, da diese ein völlig anderes Klangbild abgäben. Die Komposition muss deshalb nur auf den Ausfall und das Hinzukommen von Sensorknoten reagieren.

3.2.2 Klangauslösung

In der Komposition ist festgelegt, welche Noten wann gespielt werden. Damit diese in Klang umgesetzt werden, muss die Klangauslösung dafür sorgen, dass die richtigen Knoten zur richtigen Zeit benachrichtigt werden. Das heißt, sie muss die Komposition verstehen und umsetzen können. Da bereits vorgegeben ist, welche Note welchen Klangstab repräsentiert, ist eindeutig, welcher Knoten benachrichtigt werden muss.
Wichtig ist aber, dass ausschließlich solche Noten „gespielt“ werden, dessen Knoten aktiv sind. Ist im Ausnahmefall ein Knoten ausgefallen, dürfen ihm keine Nachrichten geschickt werden, so dass das Netzwerk nicht unnötig belastet wird.

3.3 Nichtfunktionale Anforderungen

Neben der in Kapitel 3.1.3 erwähnten Fehlertoleranz gibt es weitere nichtfunktionale Anforderungen, die im Folgenden kurz erläutert werden:
Die Installation sollte kostengünstig sein, da möglicherweise viele Knoten zum Einsatz kommen und je Knoten zusätzliche Hardware benötigt wird (z.B. Aktoren sowie notwendige elektrische Bauteile zur Ansteuerung der Aktoren)
Zudem muss die Installation besucherfreundlich „bedienbar“ sein. Das heißt, dass das Zusammenspiel zwischen Interaktion und Klang für den Besucher einerseits leicht verständlich sein soll und andererseits so interessant sein muss, dass ihn dieses nicht langweilt. Die Installation sollte weder überfordern noch sollte sie zu einem spielbaren Objekt werden, dessen Faszination nur kurzzeitig anhält. Sie soll zum Erforschen und Entdecken auffordern.
In diesem Sinne sollten die auditiven, visuellen und interaktiven Einflüsse der Installation flexibel gehalten und an der laufenden Installation praktisch erforscht werden können. Die Hardware- und Softwarelösung müssen deshalb so flexibel und erweiterbar sein, dass eine Änderung der Installation (bspw. Testen unterschiedlicher Sensortypen an der laufenden Installation) leicht und ohne großen Entwicklungsaufwand umsetzbar wäre.

3.4 Zusammenfassung

Zusammenfassend ist für die Realisierung der Klanginstallation ein kabelloses Sensor/Aktor Netzwerk erforderlich, welches zum Einen mittels Sensoren die Möglichkeit der Interaktion bietet und zum Anderen mittels Aktoren die Ausgabe und Erzeugung von Klang ermöglicht. Für dieses Netzwerk soll sowohl eine Hardware- als auch Software-Lösung entwickelt werden, welche eine möglichst flexible und kostengünstige Umsetzung der Installation erlauben. Des Weiteren sollen eine Benutzerschnittstelle für die Konfiguration des Netzwerks sowie eine Klangsteuerung zur Umsetzung einer von Sensordaten abhängigen Klangkomposition bereitgestellt werden.

4 Vergleichbare Arbeiten

Kabellose Sensor/Aktor Netzwerk finden heutzutage in den verschiedensten Gebieten Anwendung. Die häufigsten Beispiele stellen die Bereiche Umweltüberwachung, medizinische Vorsorge, Unterhaltung, Logistik oder Industrie dar (vgl. Verdone 2008, S. 16). Beispiele aus dem künstlerischem Bereich sind dagegen schon schwerer zu finden, was an einer fehlenden Hardware und Software Infrastruktur liegen mag. Dies hat sich das Projekt *Sense/Stage*[3] zu Nutze gemacht und eine kabellose Sensor Netzwerk Infrastruktur für Echtzeit-Anwendungen wie Live Performances oder interaktive Umgebungen geschaffen.
In den folgenden Kapiteln wird das Sense/Stage Projekt näher vorgestellt und eine Beispiel Anwendung beschrieben, bei welcher Sense/Stage zum Einsatz kommt. Anschließend wird das Sense/Stage Projekt mit dieser Arbeit verglichen um sowohl Gemeinsamkeiten als auch Unterschiede aufzuzeigen.

4.1 Sense/Stage

Sense/Stage ist ein Projekt, welches sich zum Ziel gemacht hat, eine kleine, kostengünstige und zugleich energieeffiziente kabellose Sensor Hardware sowie eine Software Infrastruktur zu entwickeln, welche speziell für Live Performances (der Bereiche Theater, Tanz und Musik) sowie interaktiven Echtzeit Umgebungen genutzt werden können. Sense/Stage setzt sich aus drei Komponenten zusammen (vgl. Baalmann 2010, S. 1):

- kleine, batteriebetriebene kabellose Mikrocontrollerboards für Sensoren und Aktoren
- Software zur Echtzeit-Verteilung von Daten
- Software-Module zur Analyse von Datenströmen

Die erste Komponente beschreibt die Hardwaremodule, welche Sense/Stage anbietet um ein kabelloses Sensor Netzwerk aufbauen zu können – sogenannte *MiniBees*[4]. Mit dem *SenseWorld DataNetwork*[5] bietet Sense/Stage zudem eine Softwarelösung an, welche Sensordaten der MiniBees in Echt-Zeit verfügbar macht. Die dritte Komponente sind Software-Module, die es ermöglichen Sensordaten auszuwerten und eine Grundlage bieten um komplexe, dynamische (Sensorik-)Bausteine für die Ausgabeseite erstellen zu können.
Nachfolgend stellen 4.1.1 das MiniBee Hardwaremodul und 4.1.2 das SenseWorld DataNetwork vor.[6] 4.1.3 zeigt ein Anwendungsbeispiel der Sense/Stage Plattform.

4.1.1 Sense/Stage MiniBee

Sense/Stage hat sich folgende Anforderungen an das Hardware-Design gestellt (vgl. Baalman 2010, S.2):

3 http://sensestage.hexagram.ca
4 http://sensestage.hexagram.ca/about/hardware/minibee
5 http://sensestage.hexagram.ca/about/software
6 Die Software-Module zur Analyse der Datenströme scheinen laut Webseite in Entwicklung zu sein. Weitere Veröffentlichungen (vgl. Baalman 2010, S. 6) schienen diese nur anzudeuten.

- geringe Kosten
- kleiner Formfaktor
- flexible Sensor Konfiguration
- benutzbar um Motoren, LEDs und andere Aktoren anzusteuern
- betriebsfähig in großen Gruppen (mindestens 10 Netzwerkknoten)
- anwenderfreundliche Benutzung
- programmierbar

Aus diesen Anforderung heraus entstand der Sense/Stage MiniBee - ein 34mm x 27mm kleines Mikrocontrollerboard, welches auf einem Arduino Pro Mini[7] basiert und kabellos über ein XBee Modul[8] kommunizieren kann. Der MiniBee läuft seit Revision B auf einem ATmega328p Mikrocontroller[9] bei einer Taktfrequenz von 12MHz. Neben den arduinotypischen Eigenschaften (wie analoge Eingänge, digitale Ein- und Ausgänge, Serial I/O, etc.) stellt der MiniBee einen integrierten 3-Achsen Beschleunigungssensor sowie den Anschluss für ein XBee Radiomodul bereit[10]. Die Spannungsversorgung des MiniBee's erfolgt typischerweise über eine Knopfzelle, kann jedoch auch individuell gestaltet werden.
Ein MiniBee lässt sich über die Arduino IDE[11] programmieren. Sense/Stage bietet dazu eine Firmware, die nicht nur die kabellose Kommunikation und Übertragung von Daten ermöglicht, sondern auch Funktionen bereitstellt, die bereits bestimmte Sensoren (wie Druck-, Temperatur- oder Beschleunigungssensoren) und Aktoren (wie PWM Output zum Dimmen von LEDs) unterstützt. Darüber hinaus kann der MiniBee aber auch um nicht unterstützte Sensoren/Aktoren erweitert werden (im Detail siehe https://github.com/sensestage/ssdn_minibee).
Die Firmware kann kabellos über einen Host Computer konfiguriert werden. Ein ständiges physikalisches Reprogrammieren ist deshalb nicht notwendig (es sei denn neue, nicht von der Firmware unterstützte Sensoren/Aktoren kommen zum Einsatz). Die Kommunikation zwischen Host und MiniBees geschieht über einen coordinator node[12], welcher über ein Kabel mit dem Host verbunden ist. Eine Schnittstelle zur Konfiguration der MiniBees bietet das SenseWorld DataNetwork an, welches im Folgenden vorgestellt wird.

4.1.2 SenseWorld DataNetwork

Das SenseWorld DataNetwork dient als Schnittstelle zum MiniBee Netzwerk. Es stellt bspw. Sensordaten bereit, erlaubt das Ansteuern der digitalen Ausgänge (etwa zum Ansteuern von Motoren) oder ermöglicht das Konfigurieren des Netzwerks. Es ist jedoch nicht nur auf die Integration des MiniBee Netzwerks beschränkt, sondern dient im Allgemeinen zur Verteilung von Daten zwischen allen Netzwerkteilnehmern. Im Folgenden wird die Struktur des SenseWorld DataNetworks anhand von Abbildung 3 erläutert:

7 http://arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardProMini
8 http://www.digi.com/products/wireless/point-multipoint/xbee-series1-module.jsp#overview
9 http://www.atmel.com/devices/atmega328p.aspx
10 Der MiniBee wird in der Standardversion ohne Xbee Modul für ca. 36€ angeboten; vgl. https://shop.sensestage.eu/nl/5-minibee
11 http://arduino.cc/en/main/software
12 Typischerweise Xbee Modul + XBee Explorer USB Board; vgl. https://shop.sensestage.eu/nl/xbee/43-coordinator-node.html

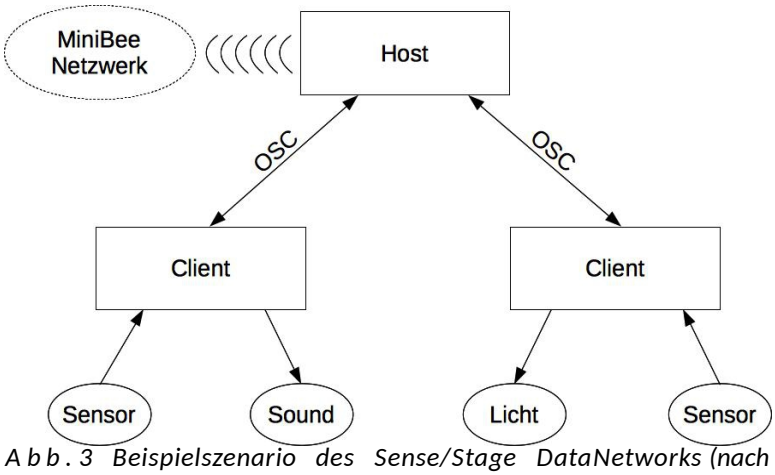

Abb. 3 Beispielszenario des Sense/Stage DataNetworks (nach Baalman 2010, S. 5)

Ein zentraler *Host* empfängt alle Daten und verwaltet die Verbindungen mit *Clients*. Ein Client kann einen oder mehrere *data nodes* (von z.B. einem MiniBee) abonnieren, von welchen er Daten empfangen möchte. Darüber hinaus kann ein Client selbst einen data node erzeugen und Daten im Netzwerk bereitstellen. Ein data node ist eine Sammlung von mehreren Daten (z.B. alle Sensordaten eines Geräts). Jedes Datum, welches in einem data node bereitgestellt wird, wird durch einen *data slot* repräsentiert. So hat bspw. ein 3-Achsen Beschleunigungssensor drei slots. Ein Client kann einzelne slots abonnieren (er muss also nicht alle drei slots des vorigen Beispiels verwenden).
Das MiniBee Netzwerk ist bereits im SenseWorld DataNetwork integriert, sodass der Host alle eingehenden Daten von und alle ausgehenden Daten an die MiniBees weiterleitet. Die Daten eines einzelnen MiniBees erscheinen dabei als ein data node, wobei jeder Sensor (mindestens) einen data slot repräsentiert. Ein Client kann nun den data node (oder slots) eines bestimmten MiniBees abonnieren. Um die Aktoren der MiniBees zu bedienen, kann ein Client einen selbst erstellten data node auf die digitalen Ausgänge der MiniBees abbilden. (vgl. Baalman 2010, S. 4-5)
Das SenseWorld DataNetwork ist so konzipiert, dass es nicht zwingend auf einem MiniBee Netzwerk aufbauen muss. Es funktioniert auch eigenständig, sodass auch eine Verteilung von Datenströmen nur zwischen Clients möglich ist.
Sense/Stage bietet eine Implementierung des Hosts für SuperCollider sowie eine Standalone Anwendung (für OSX) an. Des Weiteren werden Implementierungen der Client Anwendung für bspw. SuperCollider, PureData, Max/MSP, Processing, Java und C++ angeboten. Da die Kommunikation zwischen Host und Client auf Basis des OSC Protokolls[13] stattfindet, können auch Clients in anderen Software Umgebungen implementiert werden.

4.1.3 Beispiel Anwendung

Eine Installation, die die Sense/Stage Technologie nutzt ist *Rosa Dei Venti* von *Alberto Novello*[14]. Die Idee dieser Installation ist es, natürliche Elemente durch simple Gesten zu kontrollieren. Dazu

13 OSC im Detail in Kapitel 5.2
14 Im Detail siehe http://jestern.com/

wurde in einem dunklen, langen Korridor einer Burg die Geräusche von Wind und Sturm erzeugt um eine beängstigende Atmosphere zu schaffen. Durch das kabelloses Sensor System konnte der Besucher die Intensität des Windes mit den Händen steuern. Das Geräusch des Windes wurde von außerhalb aufgenommen und im Innern durch viele verteilte Lautsprecher (kontrolliert) abgespielt. Da die Lautsprecher jedoch nur den hohen Frequenzbereich des Windes abdeckten, wurden zusätzlich mit Hilfe von Resonatoren Wände, Fenster und Metall Platten zum Schwingen gebracht um einen größeren, realistischen und tieferen Klang zu schaffen. (vgl. Novello 2013)
Auch wenn die Installation dieser Arbeit (vgl. 2) ein anderes Klang- und Stimmungsbild erzeugen möchte, möchte sie jedoch wie bei der Installation von *Alberto Novello* einen möglichst realen, überzeugenden Klang kreieren. Um beim Besucher die beabsichtigte Stimmung zu erzeugen, findet deshalb eine reale, mechanische Klangerzeugung statt, die auf Grund verteilter Sensoren beeinflusst werden kann.

4.2 Sense/Stage im Vergleich mit dieser Arbeiten

Nachfolgend wird beschrieben, worin sich Sense/Stage aus Hardware- und Softwaresicht von dieser Arbeit unterscheidet und worin die Gemeinsamkeiten liegen:

4.2.1 Hardware

Sense/Stage's kabellose Netzwerk Hardware - der MiniBee - stellt ein fertiges Produkt dar, welches out-of-the-box benutzbar ist und ein schnelles Aufbauen eines kabellosen Sensor/Aktor Netzwerks erlaubt. Das Hardware-Design des MiniBees basiert auf einem Arduino Pro Mini und verwendet trotz geringerer Taktrate (12MHz zu 16MHz) eine zum Netzwerkknoten dieser Arbeit (siehe 5.4.1.1) vergleichbare Mikrocontrollereinheit. Der Hauptunterschied liegt jedoch im Funkmodul. Während der MiniBee ein XBee Modul zur kabellosen Kommunikation verwendet, baut der Netzwerkknoten dieser Arbeit auf einem nRF24l01+ Funkmodul auf (zum Vergleich beider Funkmodule siehe 5.4.1).
Auch wurde im Rahmen dieser Arbeit kein „fertig zu benutzendes“ Hardwareboard entworfen, jedoch aber eine Hardwarelösung bereitgestellt, die auf Basis von eines Arduino Boards und einem nRF24l01+ Radiomodul einen kabellosen Netzwerkknoten erstellen lässt. Im Gegensatz zum MiniBee kann das hier verwendete Arduino Board nach Bedarf und Anforderung an das Projekt durch ein anderes kompatibles[15] ausgetauscht werden.
Einen deutlichen Unterschied machen die Stückkosten beider Netzwerkknoten. Laut Angaben des offiziellen Shops liegt der Stückpreis eines MiniBees (inkl. XBee Modul) bei ca. 70 €. Die Kosten aller Einzelteile der hier verwendeten Grundkonfiguration (vgl. 5.4.1.1) liegen bei unter 15€.
Ein weiterer Unterschied liegt in der Firmware beider Hardwareknoten: Der MiniBee wird mit einer auf ihn zugeschnittenen Firmware programmiert, welche erweiterbar ist und bestimmte Programmabläufe vorgibt (z.B. dass aktivierte Sensoren regelmäßig messen und ihre Messergebnisse automatisch mitteilen). Der in dieser Arbeit entstandene Netzwerkknoten stellt vielmehr das Grundgerüst eines kabellosen Netzwerkknotens dar, welcher von sich aus keine Dienste anbietet, jedoch flexibel um solche erweitert werden kann (Code Beispiel; siehe 5.4.1.1). Des Weiteren werden jeweils das Beispiel eines Sensor und Aktorknotens gegeben, welche eine mögliche Implementierung bereitstellen. Im Vergleich zum MiniBee ist also zusätzlicher Programmieraufwand notwendig um einen Sensor- oder Aktorknoten zu erzeugen. Dies bietet

15Kompatibel sind solche, dessen Mikrocontroller vom Treiber des nRF24l01+ unterstützt werden. Unterstützte Boards sind bspw. Uno, Nano, Mega, Arduino Due, etc.; im Detail siehe https://github.com/TMRh20/RF24.

jedoch auch die Möglichkeit einen solchen Netzwerkknoten individuell (z.B. abhängig von der verwendeten Hardware) zu gestalten.

4.2.2 Software

Der Zugriff auf die Sensoren und Aktoren des MiniBee Netzwerk findet über eine zentrale Software (den Host des SenseWorld DataNetworks) statt. Das SenseWorld DataNetwork bietet dazu die Möglichkeit, dass Clients sich für bestimmte Sensoren des MiniBee Netzwerks anmelden und regelmäßig die neusten Messergebnisse der abonnierten Sensoren erhalten können. Die Kommunikation zwischen Client(s) und Host findet auf Basis von OSC statt, sodass nicht nur die für verschiedene Entwicklungsumgebungen bereitgestellten Clients zur Bedienung des MiniBee Netzwerks verwendet werden können, sondern auch solche, die die OSC Schnittstelle des Hosts bedienen.
Der Zugriff auf Knoten des in dieser Arbeit entwickelten kabellosen Sensor/Aktor Netzwerks findet über den *network controller* statt. Dies geschieht ebenfalls durch Senden von OSC Nachrichten. Jedoch werden die OSC Nachrichten (so wie sie sind) direkt an die an sie adressierten Knoten weitergeleitet um die Dienste der Knoten anzusprechen. Im Gegensatz zum SenseWorld DataNetwork findet also keine Verwaltung der Netzwerkknoten durch eine zentrale Instanz sondern vielmehr eine Vermittlung statt.

4.3 Fazit

Sense/Stage stellt eine kompakte, energieeffiziente und robuste Hardware-Software-Lösung dar, welche ein kabelloses Sensor/Aktor Netzwerk leicht und schnell umsetzen lässt. Für die Umsetzung dieser Installation kommt Sense/Stage dennoch nicht in Frage, da die MiniBees bei der benötigten Anzahl (10+) zu kostenintensiv wären. Sense/Stage wurde für Anwendungen wie Live Performances konzipiert, bei welchen kein zusätzlicher Entwicklungsaufwand für eine Hardware-Software-Infrastruktur gewünscht ist oder gar hinderlich für den kreativen Schaffensprozess der Künstlerin oder des Künstlers wäre.
Im Gegensatz zu Sense/Stage ist die nachfolgend vorgestellte Hardware-Software-Lösung an Entwickler und Hobby Bastler gerichtet, welche zur Errichtung eines kabellosen Sensor/Aktor Netzwerks auf einer kostengünstigen Plattform aufbauen möchten.

5 Lösungsentwurf

Im folgenden Kapitel wird anhand der Anforderungsspezifikation ein Lösungsentwurf zur Realisierung der Installation entwickelt. Dazu wird zunächst in Kapitel 5.1 eine mögliche Architektur, ihre Komponenten sowie die Kommunikationswege zwischen den Komponenten vorgestellt. Im Anschluss beschreibt Kapitel 5.2 das Nachrichtenprotokoll, welches die Komponenten nutzen um (auf Applikationsebene) miteinander zu kommunizieren. Kapitel 5.3 spezifiziert die Komponenten und zeigt auf, was zur Realisierung dieser nötig ist. Zuletzt stellt Kapitel 5.4 eine darauf aufbauende Hardware- und Softwarelösung vor.

5.1 Architektur der Klanginstallation

Typischerweise bestehen kabellose Sensor und Aktor Netzwerke (WSAN) aus *nodes* (entweder Sensoren oder Aktoren), *sinks* und *gateways*. Die Sensorknoten erfassen dabei die Umgebung und leiten die Information mittels kabelloser Verbindung an *sinks* (auch Controller oder Monitor) weiter. Diese können die Informationen lokal verarbeiten oder via *gateway* in anderen Netzwerken (z.B. Internet) bereitstellen (vgl. Verdone 2008, S. 1/5).
Zur Umsetzung der Klanginstallation wird eine Architektur vorgestellt, die sich sowohl aus einem solchen WSAN aufbaut als auch aus Programmen, die Dienste dieses Netzwerks in Anspruch nehmen. Die folgende Abbildung zeigt die Architektur der Installation, wie sie umgesetzt werden könnte:

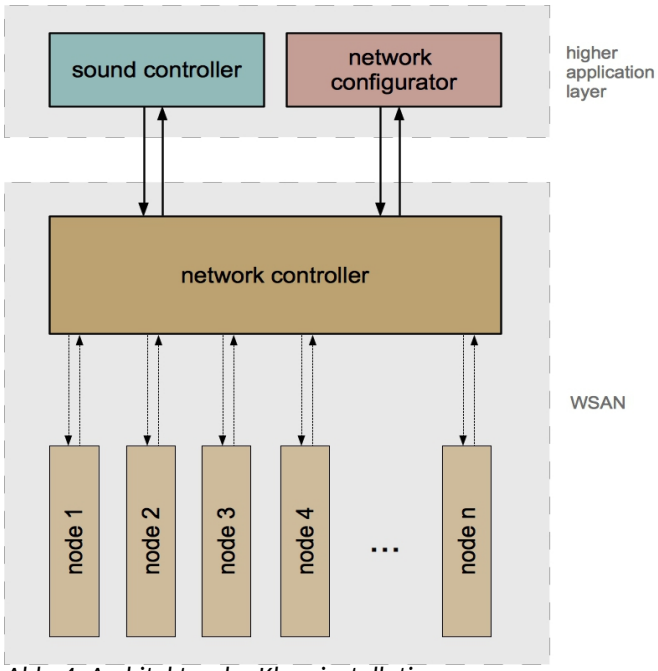

Abb. 4: Architektur der Klanginstallation

Das WSAN lässt sich hierbei auf eine simple Anordnung von *nodes* und *network controller* reduzieren. *Nodes* stellen die Netzwerkknoten dar, die mit Sensoren oder Aktoren ausgestattet sind und bestimmte Dienste bereitstellen. Der *network controller* nimmt zwei Rollen ein: Zum Einen ist er als Koordinator für den Aufbau (Anmelden von *nodes*) und die Instandhaltung (Wegfallen von *nodes*) des Netzwerks zuständig. Zum Anderen dient er als *gateway* und stellt die Kommunikation mit *nodes* von außerhalb des WSAN zur Verfügung. So leitet er Anfragen der *higher application layer* (z.B. „node X, spiele Klangstab Y“) an die jeweiligen Knoten weiter und schickt Nachrichten (z.B. Sensormessungen) von den *nodes* an diese nach draußen. Die *higher application layer* setzt sich zusammen aus dem *sound controller* und dem *network configurator*, die beide die Dienste der *nodes* in Anspruch nehmen.
Die Kommunikation zwischen allen beteiligten Komponenten findet bidirektional statt. Der Nachrichtenaustausch und das Nachrichtenformat basieren auf dem Open Sound Control (OSC) Protokoll, welches im folgenden Kapitel näher beschrieben wird.

5.2 Nachrichtenprotokoll

Die Kommunikation der einzelnen Komponenten findet, wie bereits erwähnt, bidirektional zwischen *higher application layer* (außerhalb des WSAN) und *network controller* sowie zwischen *network controller* und *nodes* statt. Da die Installation flexibel änderbar sein soll (was zum Beispiel bedeuten könnte, dass Knoten freiprogrammierbar sein können), muss ein einheitliches Nachrichtenprotokoll gewählt werden, welches auf Änderungen in der Installation reagieren kann und keine Neuentwicklung oder Umgestaltung des Hardware-Software-Designs erforderlich macht.
Open Sound Control (OSC) stellt ein solches universell einsetzbares Protokoll dar. Es ist transport-unabhängig, nachrichtenbasiert und wird typischerweise für die Kommunikation zwischen Computern, Sound Synthesizern und anderen Multimedia Geräten eingesetzt (vgl. Wright 2003).

Ein Anwendungsbereich des OSC Protokolls, welcher der Klanginstallation sehr ähnlich ist, ist das Gesten-Basierte Elektronische Musik Instrument, wie folgendes Zitat zeigt:

> *„A human musician interacts with sensor(s) that detect physical activity such as motion, acceleration, pressure, displacement, flexion, keypresses, switch closures, etc. The data from the sensor(s) are processed in real time and mapped to control of electronic sound synthesis and processing.“ (vgl. Wright 2010)*

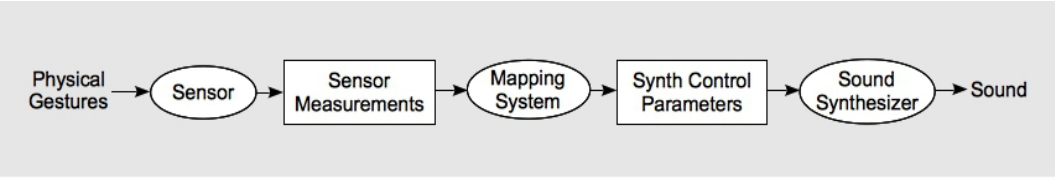

Abb. 5: Prozess- (Ovale) und Datenfluss (Rechtecke) in einem Sensor-basierten Musik Instrument (nach Wright 2010)

Abbildung 5 zeigt die oben beschriebene Funktionsweise des Gesten-basierten Elektronischen Musik Instruments. Die Installation findet auf ganz ähnliche Weise statt: Das Bewegen von Besuchern wird mittels Sensoren erfasst. Diese Sensordaten werden bestimmten Regeln zugewiesen, die eine Komposition von Noten erzeugen. Anschließend werden diese Noten in

Form von Nachrichten an die ihnen zugewiesenen Knoten geschickt und lösen einen realen, mechanisch erzeugten Klang aus.
Es findet also ein steter Datenfluss zwischen Prozessen statt, die ggf. auf unterschiedlichen Systemen laufen. OSC bietet hier den Vorteil, dass es transport-unabhängig, für moderne Netzwerk Technologien optimiert ist und bereits für viele Systeme implementiert wurde. So stellt beispielsweise *OSCuino* im Bereich Embedded Systems eine OSC Bibliothek für Arduino, Teensy und ähnliche Embedded Prozessor Plattformen[16] zur Verfügung. Zudem bringt OSC Features wie eine Pattern Matching Sprache, die es erlaubt mittels regulärer Ausdrücke ein flexibles Addressierungsschema zu gestalten (z.B. Ansprechen mehrerer Empfänger mit nur einer Nachricht).
Die Grundeinheit einer Übertragung sind sogenannte *OSC Packets* (Paket). Diese können zwei Typen beinhalten: *OSC Messages* (Nachrichten) und *OSC Bundles* (Bündel). Nachrichten repräsentieren ein individuelles Kommando und setzen sich zusammen aus dem *OSC Address Pattern* (hierarchisch; ähnlich dem Unix Dateisystemformat) gefolgt von einem *OSC Type Tag String* gefolgt von möglichen *OSC Arguments* (z.B. mögliche Datentypen wie int32, float32, OSC-string). Bündel sind eine Sammlung von Nachrichten (und/oder weiteren Bündeln) plus einem 64-bit timetag (im NTP-Format; Network Time Protocol), der die beabsichtige Zeit zur Ausführung der Kommandos vorgibt. OSC Packets sind ein zusammenhängender Block an Bytes, dessen Größe immer ein Vielfaches von vier bildet. [17]
Im Folgenden wird beispielhaft eine OSC Nachricht gezeigt, die das Anschlagen eines Klangstabs in einem Knoten auslösen könnte. Die linke Spalte stellt die Nachricht als ASCII Zeichen dar und die rechte Spalte zeigt die Darstellung der Bytes als Hexadezimalziffern:

ASCII Format				**Hex Format**			
/	n	o	d	2f	6e	6f	64
e	/	2	/	65	2f	32	2f
c	h	i	m	63	68	69	6d
e				65	0	0	0
,	i			2c	69	0	0
0	0	0	5	0	0	0	5

Das *OSC Address Pattern* lautet „`/node/2/chime`“. Der *OSC Type Tag String* „`,i`“ und das beinhaltete Datum 5 - die Nachricht lautet: „Knoten 2, schlage Klangstab 5 an!“.
Jedes *OSC Address Pattern* beginnt mit einem '`/`' gefolgt von der Adresse, wobei der letzte Teil der Name des auszuführenden Kommandos (*OSC Method*) ist. In diesem Beispiel ist „`/node/2`“ die Adresse des zweiten Knotens und „`/chime`“ (für Anschlagen) der Name des Kommandos.
Der *OSC Type Tag String* beginnend mit einem '`,`' enthält der Reihenfolge nach die Datentypen der beinhalteten Daten einer Nachricht ('`i`' für int32, '`f`' für float32, '`s`' für *OSC-string*, etc.). Anschließend folgen in selber Reihenfolge die Daten. Diese stellen die Argumente dar, die der *OSC Method* übergeben werden - in unserem Beispiel die `5`, die vorgibt, dass Klangstab 5 angeschlagen werden soll.
Auffällig in diesem Beispiel sind die jeweils folgenden Nullen (siehe Tabelle - Spalte „Hex Format“) am Ende des *OSC Address Pattern* und dem *OSC Type Tag String*. Damit Nachrichten (sowie

16 http://cnmat.berkeley.edu/oscuino
17 im Detail siehe http://opensoundcontrol.org/spec-1_0

bundles) ein Vielfaches von vier ergeben, ist vorgegeben, dass auch *OSC Address Pattern*, *OSC Type Tag String* und alle Daten jeweils ein Vielfaches von vier bilden müssen. Aus diesem Grund muss ggf. mit Nullen aufgefüllt werden.
Das Integer-Datum aus diesem Beispiel wurde nicht mit Nullen aufgefüllt, da es sich um eine 32-Bit Integer Zahl handelt und daher bereits ein Vielfaches von vier ist. Würde die Nachricht aber zusätzlich das *OSC-string* Datum „`play`“ enthalten, müsste, da ein String auf Null enden muss, mit drei weiteren Nullen aufgefüllt werden - also insgesamt vier Nullen: „`play\0\0\0\0`“.
Zusätzlich erlaubt die bereits angedeutete Pattern Matching Sprache die Verwendung regulärer Ausdrücke im Adress Bereich. Diese macht es möglich sowohl eine eindeutige Adresse als auch ein Adressmuster anzugeben um bspw. Gruppen von Knoten anzusprechen. Statt nur einen Knoten anzusprechen könnte mit folgendem *OSC Address Pattern* an alle Knoten adressiert werden: „`/node/*`“. Ein etwas umfangreicheres Beispiel ist aber auch möglich: „`/*/[0-9]/{chime,sensor}`“. Dies würde soviel bedeuten wie „Führe Methode `chime` und/oder `sensor` in einem beliebigen Gerät mit einer beliebigen (einstelligen) Nummer aus!“. Das Pattern Matching funktioniert so, dass ein OSC Server (in diesem Falle die Netzwerkknoten) eine Sammlung von *OSC Methods* (die Dienste) bereitstellt, die ein potentielles Ziel einer OSC Nachricht sein können. Trifft das Adressmuster auf die Adresse der Methode zu, wird diese ausgeführt. Dies ist vergleichbar mit einem Funktionsaufruf bei dem Argumente übergeben werden können.
Um den Dienst eines Netzwerkknotens in Anspruch zu nehmen, muss eine *OSC Nachricht* an den *network controller* geschickt werden, welche dieser an den adressierten Knoten weiterleitet. Das *OSC Address Pattern* dieser Nachricht muss, wie obiges Beispiel demonstriert, folgendes Schema aufweisen:

„`/node/`X[/...]/SERVICE“

„`/node/`X“ ist die OSC Adresse des Knotens, wobei X die Knotennummer des Knotens angibt. SERVICE gibt den Namen des Dienstes an, welcher ausgeführt werden soll. Dienste können wie bei der Verzeichnisstruktur eines Dateisystems hierarchisch gegliedert werden. So könnte beispielsweise „`/node/2/sensor/3/threshold/set`“ den Grenzwert von Sensor drei des zweiten Knotens ändern. Die in der Nachricht enthaltenen Daten stellen die Parameter dar, welche dem Dienst bei Aufruf übergeben werden.
Das heißt, dass alle Nachrichten, die an einen Knoten gerichtet sind, an diesen übermittelt werden – auch wenn diese Knoten das auszuführende Kommando gar nicht unterstützen und lediglich ignorieren. Dies hängt mit der in Kapitel 3.3 geforderten Erweiterbarkeit zusammen: Ein Knoten soll flexibel gestaltet und um beliebige Dienste erweitert werden können. Da diese über die mitgesandten Kommandos ausgeführt werden, soll der Knoten selbst entscheiden können, welche Nachrichten bearbeitet werden. Theoretisch könnte bereits der *network controller* entscheiden, welche Kommandos an welche Knoten weitergeleitet werden (diese könnten beim Anmelden eines Knotens übermittelt werden). Allerdings müsste der *network controller* alle Kommandos aller Knoten kennen. Dies stellt auf Grund von Speicherknappheit für viele Embedded Systeme ein Problem dar. Der *network controller* prüft daher lediglich, ob die Nachricht an einen angemeldeten Knoten gerichtet ist oder nicht, und leitet diese gegebenenfalls weiter.

5.3 Komponenten

Nachfolgend werden alle Architektur Komponenten im Einzelnen beschrieben und aufgezeigt, was notwendig ist, um diese zu realisieren.

5.3.1 WSAN

Das WSAN besteht aus einer zentralen Steuereinheit - dem *network controller* – und mehreren an ihn angeschlossene Knoten (*nodes*), die die Endpunkte des Netzwerks bilden. Das Netzwerk ist (physikalisch und logisch) in Stern Topologie ausgelegt, d.h. die Kommunikation findet bidirektional zwischen *network controller* und *nodes* statt. Eine Sterntopologie ist deshalb gewählt, da sie in den Fällen zu bevorzugen ist, in denen die durch das Netzwerk abgedeckte Fläche klein ist und es geringer Latenz bedarf (vgl. Verdone 2008, S. 130). Beide Fälle sind in der Installation gegeben, da sie einerseits in einem geschlossenen/begrenzten Raum stattfindet und andererseits der Klang im Vordergrund der Installation steht. Da sich dieser nicht nur aus der Klangerzeugung sondern auch der permanenten Interaktion ergibt, sollten die Ein- und Ausgabe Wege so kurz wie möglich gehalten werden.
Die Anordnung des Netzwerks in Sterntopologie bietet außerdem noch einen weiteren Vorteil: Das Hinzukommen sowie Wegfallen von Knoten betrifft nur den *network controller*, die Knoten bleiben jedoch unbetroffen. Ein neuer Knoten ist deshalb leicht in das Netzwerk zu integrieren, da dies nur über den *network controller* abläuft und nur er prüfen müsste, ob ein neuer Knoten akzeptiert werden kann oder nicht. Auch das Wegfallen eines Knotens findet nur über den *network controller* statt, weitere Knoten wären vom Ausfall nicht betroffen. Dies ermöglicht eine leichtere Fehlersuche und hätte möglicherweise eine nicht allzu große Auswirkung auf die laufende Installation. Diese Selbstorganisation des Netzwerks wäre in anderen Netzwerk-Topologien schon aufwändiger: Das Hinzukommen eines Knotens in einem Netzwerk mit Baumstruktur würde das Absuchen des Baumes nach einem freien Platz erfordern. Dramatischer wäre aber das Wegfallen eines Knotens. Hätte dieser als Wurzelknoten weitere von ihm ausgehende Unterknoten, wären diese nicht mehr erreichbar. Dies hätte zugleich eine aufwendige Fehlerbehebung zur Folge.
Zu erwähnen sind aber auch die Nachteile der Sterntopologie, denn einen großen Nachteil hat diese Topologie. Da die gesamte Kommunikation nur über einen zentralen Verteiler läuft, wäre bei Ausfall das gesamte Netzwerk unerreichbar (*single point of failure*). Auch die Anzahl an Netzwerkknoten kann nicht beliebig groß gewählt werden, denn je mehr Knoten der *network controller* bedienen muss, desto geringer wird (in Abhängigkeit von der Auslastung) die Übertragungsrate. Die hier vorgestellte Installation benötigt allerdings keine übermäßig große Anzahl an Knoten (schätzungsweise 10-20 Stück); dies wäre daher weniger von Nachteil. Der Ausfall des *network controllers* hat jedoch schon gravierendere Auswirkungen auf die Installation. Daher muss das Netzwerk so ausgelegt sein, dass der *network controller* leicht auswechselbar ist - sowohl aus Hardware- als auch aus Softwaresicht: Er muss leicht zu ersetzen sein und alle erforderliche Maßnahmen zur Instandsetzung des Systems treffen (etwa Neuanmeldung der Knoten auslösen, berichten höherer Applikationsebenen, etc.).
Im WSAN findet die Kommunikation zwischen *network controller* und *nodes* auf mehreren Ebenen statt. So muss sowohl das grundlegende kabellose Übertragen von Bitströmen über Radiomodule, als auch das Anmelden und Wegfallen von Knoten auf Netzwerkebene, sowie die Übermittlung von OSC für die Applikationsebene gewährleistet sein. Die Kommunikation kann deshalb an das OSI-Referenzmodell[18] angelehnt in unterschiedliche, logisch aufeinander

18 vgl.http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=2820

aufbauende Schichten gegliedert werden und sollte zur Erleichterung der (Weiter-)Entwicklung nach diesem implementiert werden.
Doch zunächst werden die beiden Komponenten *node* und *network controller* und ihre Rolle im Netzwerk vorgestellt:

5.3.1.1 Node

Ein Knoten (*node*) muss die Möglichkeit bieten die Umgebung mittels Sensor(en) zu erfassen oder mittels Aktor(en) zu beeinflussen. Er soll gleichzeitig kabellos und möglichst kostengünstig sein (vgl. 3.1.2).
Das bedeutet, dass ein Knoten autonom agieren und je nach Einsatz unterschiedliche Dienste bereitstellen können muss. So kann er als Sensorknoten fungieren oder als Aktorknoten die Klangerzeugung umsetzen. Ein Knoten muss daher vielseitig einsetzbar und freiprogrammierbar sein (auch unterschiedliche Hardwarekomponenten bedienen können!) und gleichzeitig die Rolle des Netzwerkknotens einnehmen (Verarbeitung der kabellosen Kommunikation, Anmelden beim *network controller*, etc.).
Zur Realisierung eines Knotens bedarf es daher zweierlei: Einer Grundkonfiguration an Hardwarekomponenten, die den Netzwerkknoten als solchen (kabellos) netzwerkfähig macht, und einer Softwarelösung, die sowohl die Netzwerkkommunikation abhandelt, als auch ermöglicht den Knoten um die Dienste, wie sie in Kapitel 3.1.2 beschrieben wurden, zu erweitern. Diese Dienste erfordern ggf. zusätzliche Hardware.
Doch bevor Dienste eines Knotens überhaupt genutzt werden können, muss dieser dem *network controller* bekannt und im Netzwerk integriert sein. Nach dem Einschalten beginnt ein Knoten deshalb in einer Initialisierungsphase. Erst wenn diese Initialisierungsphase abgeschlossen, wechselt der Knoten in einen empfangsbereiten Zustand, in dem er Anfragen entgegen nehmen kann. Der Programmablauf eines Knotens kann daher in zwei Zustände eingeteilt werden: *Initialisierung* und *Betrieb*.

5.3.1.1.1 Initialisierung

Zu Beginn der *Initialisierung* muss der Knoten zunächst das Funkmodul (im Detail siehe 5.4.1) konfigurieren. So wird bspw. jeder Knoten mit der selben (Hardware) Adresse gestartet; diese ist dem *network controller* bekannt und kann als Broadcast Adresse genutzt werden.[19]
Nach Einrichtung des Funkmoduls handelt der Knoten mit dem *network controller* eine eindeutige Knotennummer aus. Dazu meldet sich der Knoten nun solange beim *network controller* an, bis die Anmeldung erfolgreich ist. Der Anmeldevorgang eines Knotens beim *network controller* erfolgt auf Basis eines 3-Wege-Handshakes und ist in Abb. 6 dargestellt:

19 Alternativ hätte jeder Knoten mit einer zufällig generierten Adresse starten können, die dem *network controller* bei der Anmeldung mitgeteilt werden müsste. Dieser Ansatz schien jedoch aufgrund der beschränkten Adressierungsmöglichkeit durch die Hardware zu aufwändig als dass sie effektiv genug wäre. Außerdem findet eine Broadcast Adresse im späteren *Betrieb* sowieso Verwendung.

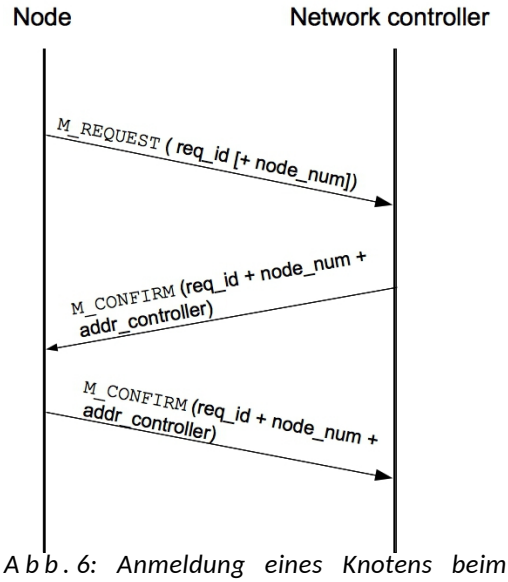

Abb. 6: Anmeldung eines Knotens beim network controller

Der Knoten schickt dem *network controller* zuerst eine *request* Nachricht. Diese enthält eine request-ID, die für jeden *request* neu generiert wird, und kann optional eine Knotennummer enthalten.
Die request-ID macht einen *request* eindeutig und ist für den *network controller* notwendig um einen bestimmten *request* einem bestimmten Knoten zuzuweisen.
Die Knotennummer ist diejenige Nummer, die einen Knoten eindeutig identifizieren lässt (vgl. 3.1.2) und mit der ein Knoten wie im Beispiel der OSC Nachricht aus Kapitel 5.2 angesprochen werden kann. Ein Knoten kann sich diese Knotennummer vom *network controller* geben lassen oder versuchen sich unter einer Knotennummer anzumelden, welche bei Programmierung des Knotens angegeben wurde. Diese wird optional der *request* Nachricht beigefügt. Theoretisch können nun zwar zwei unterschiedliche Knoten versuchen sich mit der selben Knotennummer anzumelden, akzeptiert wird jedoch nur einer. Dieser Fall ist praktisch gesehen auszuschließen, da in der Installationen jeder Knoten eine vorgesehene, eindeutige Knotennummer erhält. Da im Fehlerfall aber eine doppelte Knotennummer vergeben werden könnte, muss dieses auch berücksichtigt werden.
Hat der *network controller* eine *request* Nachricht erhalten, prüft er zunächst, ob eine Knotennummer mitgeschickt wurde. Ist dies nicht der Fall, wird dem Knoten die nächste freie Knotennummer zugeteilt. Wurde allerdings eine Knotennummer mitgeteilt, muss geprüft werden, ob bereits ein Knoten unter der mitgesandten Knotennummer angemeldet ist. Trifft dies zu, muss der *network controller* zusätzlich prüfen, ob dieser Knoten noch aktiv ist. Es kann nämlich vorkommen, dass ein Knoten ausfällt, jedoch beim *network controller* noch als aktiv gekennzeichnet ist. Damit der Knoten bei Neuanmeldung nicht durch seine eigene Knotennummer blockiert wird, muss der *network controller* diese zusätzliche Prüfung durchführen. Ist eine Knotennummer frei (geworden), schickt der *network controller* eine *confirm* Nachricht. Diese enthält als Bestätigung die vorgesehene Knotennummer, die request-ID sowie die Empfangsadresse des *network controllers*, an welche der Netzwerkknoten künftig im Zustand *Betrieb* seine Nachrichten schicken soll.[20]

20 Diese Adresse dient dazu, dass Knoten im Zustand *Betrieb* ihre Nachrichten nicht nur an eine Adresse des *network controllers* (die Default Adresse) schicken, sondern dass der Nachrichtenverkehr auf die Hardware-internen Empfangsbuffer des Funkmoduls (im Detail siehe 5.4.1) verteilt werden können. Das verwendete Funkmodul erlaubt bis

Kann einem Knoten keine Knotennummer zugeteilt werden, wird der *request* ignoriert. Bleibt eine *confirm* Nachricht des *network controllers* aus[21], muss der Knoten (nach Ablauf eines Timeouts) die Anmeldung erneut starten. Erreicht den Knoten jedoch eine Bestätigung, sendet dieser ebenfalls eine Bestätigung zurück. Diese enthält ebenfalls request-ID, Knotennummer sowie die Empfangsadresse des *network controllers*.
Erhält der *network controller* die Bestätigung des Knotens, kann er anhand der request-ID (nach Prüfung der Knotennummer und der Empfangsadresse) den jeweiligen Knoten für die jeweilige Knotennummer aktivieren. Erreicht allerdings die *confirm* Nachricht den *network controller* nicht, obwohl sie losgeschickt wurde, ist der Knoten zwar bereits im Zustand *Betrieb*, jedoch beim *network controller* nicht vollständig angemeldet. Da allerdings auch die Benachrichtigung für die *higher application layer* ausbleibt, kann von einem Fehler ausgegangen werden; der Knoten muss manuell neugestartet werden.[22]
Der Anmeldevorgang wird so oft wiederholt, bis der Knoten erfolgreich beim *network controller* angemeldet ist. Erst dann ist die Initialisierungsphase abgeschlossen, der Knoten erweitert sich um eine für die Knotennummer eindeutige Hardware Adresse (die Default Adresse bleibt für Broadcast Nachrichten erhalten; vgl. 5.3.1.1.3), teilt seine Dienste mit und wechselt dann in den Zustand *Betrieb*. Der Knoten kann nun auch von außerhalb des WSANs kontaktiert werden.
Das Mitteilen der Dienste geschieht auf Basis von OSC Nachrichten, die als Template Nachrichten an die *higher application layer* geschickt werden. Dabei geben jeweils das *OSC Address Pattern* die Adresse und der *OSC Type Tag String* die Argumente eines Dienstes vor. Die enthaltenen Daten der Nachricht sind entweder Dummy Bytes oder die letzte Konfiguration, die an dem Knoten zugeschickt wurde. Somit wird zugleich die Schnittstelle bekannt gegeben, die es erlaubt Knotendienste anzusprechen, und die letzte Konfiguration mitgeteilt.

5.3.1.1.2 Betrieb

Im Zustand *Betrieb* wartet der Knoten lediglich auf Nachrichten und muss, weil er batteriebetrieben sein kann, so energiesparend wie möglich arbeiten (vgl. 3.1.2).
Da das Funkmodul einen hohen Energiebedarf hat, muss die Kommunikation ressourcenschonend stattfinden. Aus diesem Grund kann der Knoten nicht auf ein permanentes (aktives) Empfangen[23] geschaltet sein, sondern wird in einem energiesparenden Standby-Modus betrieben. Dieser erlaubt ein Hardware internes Empfangen, welches bei neuer Nachricht einen Interrupt auslöst. Der Knoten kann sich also schlafen legen, so lange bis ein Interrupt auftritt. Ist dies der Fall, erwacht der Knoten, liest und verarbeitet alle neu verfügbaren Nachrichten und legt sich erneut schlafen.
Sensorknoten müssen zudem neben dem normalen Empfangen von Nachrichten zusätzlich Messungen vornehmen und Sensordaten verarbeiten. Dies kann auf Anfrage oder periodisch geschehen; die Mitteilung des Messergebnisses erfolgt dann auch je nach Fall als Antwort auf die Anfrage oder periodisch nach jeder neuen Messung. Da beim periodischen Messen auch dann Messergebnisse mitgeteilt würden, wenn überhaupt keine Interaktion stattfände (der Sensorwert also besonders klein ausfiele), kann für jeden Sensor eines Knotens ein Grenzwert (*threshold*) angegeben werden. Der Sensorwert wird erst dann (periodisch) mitgeteilt, wenn der Sensorwert über dem Grenzwert liegt.

---

zu sechs Empfangsadressen. Dies kann durch Benutzung mehrerer Funkmodule erweitert werden.

21 z.B. weil der Knoten abgelehnt wurde aber auch weil der *request* nicht zugestellt wurde

22 Zur Verbesserung dieses Problems stellt die verwendete Hardwarelösung (siehe 5.4.1) einen *auto acknowledge* Mechanismus bereit, der erkennen lässt, ob eine Nachricht zugestellt wurde oder nicht. Diese lässt den Knoten selbstständig entscheiden, ob eine Neuanmeldung erforderlich ist oder nicht.

23 Permanentes Anfragen beim Funkmodul, ob Nachrichten empfangen wurden

Des Weiteren sollen Knoten einen Dienst anbieten, der es erlaubt den Knoten zurückzusetzen (vgl. Kapitel 3.1.2). Dieser wird durch Senden einer *reset* Nachricht ausgelöst. Erhält ein Knoten eine solche Nachricht, wechselt er in den Zustand *Initialisierung*. Die Initialisierungsphase startet erneut, alle Konfigurationen werden zurückgesetzt und der Knoten muss sich nochmals anmelden um in den Zustand *Betrieb* wechseln zu können.

5.3.1.1.3 Ausnahmefall

Im Folgenden wird der Ausfall des *network controllers* behandelt[24]:

Befindet sich der Knoten bei Ausfall des *network controllers* im Zustand *Initialisierung*, wird die Anmeldung beim *network controller* fehlschlagen. Der Knoten wird versuchen sich solange anzumelden bis der *network controller* wieder aktiv ist.
Befindet sich der Knoten bei Ausnahmefall im Zustand *Betrieb*, wird er den Ausfall nicht mitbekommen, da er lediglich auf Empfangen eingestellt ist (mit Ausnahme der Sensorknoten, die auf periodisches Messen eingestellt sind). Dies ist für den Knoten nicht weiter schlimm, da er ohne *network controller* sowieso unerreichbar ist und sich in einem energiesparenden Zustand befindet; lediglich periodisch eingestellte Sensorknoten würden unnötige Messungen vornehmen. Erst wenn der *network controller* neugestartet wurde, kann dieser den Knoten den Ausfall berichten. Da der neugestartete *network controller* nicht wissen kann, welche Knoten aktiv waren, schickt er an alle eine *alive* Nachricht[25] um zu prüfen, ob bereits Knoten mit der jeweiligen Nummer existieren. Der *network controller* wartet nun für einen bestimmten Zeitraum auf die Bestätigung von aktiven Knoten. Dies geschieht durch Zurücksenden der *alive* Nachricht. Alle Knoten die in dem genannten Zeitraum geantwortet haben, können vom *network controller* erneut als aktiv markiert werden. Ein zuvor angemeldeter Knoten ist nun wieder erreichbar; alle seine zuvor gespeicherten Sensor-/Aktorkonfigurationen bleiben erhalten.

5.3.1.2 Network controller

Der *network controller* übernimmt zwei Rollen: Als zentraler Teil des Netzwerks ist er *Koordinator*, welcher das Netzwerk aufbaut und im Ausnahmefall eine geeignete Fehlerreaktion ausführt (vgl. 3.1.3 - Selbstorganisation des Netzwerks). Er kommuniziert einerseits mit Netzwerkknoten und andererseits mit der *higher application layer*, die die Dienste der Knoten in Anspruch nimmt. Er ist somit zugleich *gateway* und stellt eine Verbindung zwischen dem WSAN und einem äußeren Netzwerk her.
Zur Realisierung des *network controllers* bedarf es daher einer Entwicklungsplattform, die wie die Knoten auch die selbe Hardware-Software-Konfiguration zur kabellosen Datenübertragung nutzt, und zusätzlich die Kommunikation mit der *higher application layer* ermöglicht. Diese kann durchaus kabelgebunden stattfinden.
Der Ablauf des *network controllers* kann wie bei Knoten auch durch die Zustände *Initialisierung* und *Betrieb* beschrieben werden.

---

24 Der Ausfall von Komponenten der *higher application layer* ist irrelevant, da Knoten nur auf Anfragen warten. Lediglich Sensorknoten, die auf periodisches Messen eingestellt sind, verschicken regelmäßig Messergebnisse. Dies wäre dann unnötig, wenn alle Komponenten der *higher application layer* ausgefallen wären. Dieser Fall wird jedoch nicht behandelt.

25 Dies geschieht durch Senden einer Broadcast Nachricht. Diese erreicht den Knoten über die Default Adresse, die jeder Knoten bei der *Initialisierung* eingerichtet hat.

5.3.1.2.1 Initialisierung

Im Zustand *Initialisierung* wird zunächst das Funkmodul konfiguriert und eine Default Adresse, an welche die Knoten ihren *request* schicken, eingerichtet. Anschließend prüft der *network controller*, wie im vorigen Kapitel erläutert, durch Senden einer *alive* Nachricht, welche Knoten aktiv sind und merkt sich diese.
Anschließend meldet er sich beim Netzwerk der *higher application layer* an. Dazu wird im lokalen Netzwerk auf der IP-Adresse „127.0.0.1“ (localhost) jeweils ein Port zum Empfangen und ein Port zum Senden von Nachrichten eingerichtet. Auf Grund geringerer Latenz (die für Audioapplikationen von hoher Bedeutung sind) findet die Kommunikation zwischen *network controller* und *higher application layer* verbindungslos (auf Basis von UDP) statt.
Zuletzt benachrichtigt er die *higher application layer*, dass er nun betriebsbereit ist und wechselt dann in den Zustand *Betrieb*.

5.3.1.2.2 Betrieb

In diesem Zustand nimmt der *network controller* sowohl Nachrichten von Netzwerkknoten sowie von außerhalb des WSANs entgegen.
Nachrichten von Netzwerkknoten an den *network controller* sind solche, die entweder an ihn gerichtet sind (z.B. ein *request* bei der Anmeldung) oder an die *higher application layer* weitergeleitet werden sollen. Letztere sind alle OSC Nachrichten, die der *network controller* von einem Knoten erhält. Das heißt, dass alle OSC Nachrichten der Knoten ungefiltert an die *higher application layer* weitergeleitet werden. Nachrichten, die das WSAN betreffen (wie *request*, *confirm* oder *alive)*, werden nicht als OSC Nachrichten gesendet, sondern erhalten jeweils einen eigenen Nachrichten Typ. Dies bringt den Vorteil, dass einem Knoten (dessen Dienste über OSC Nachrichten angesprochen werden) der volle Namensraum zur Benennung seiner Dienste zur Verfügung steht.
Um zu erkennen um welche Nachricht es sich handelt, gibt ein *message type* den Typ einer Nachricht vor. So gibt bspw. `M_OSC_MESSAGE` an, dass es sich um eine OSC Nachricht handelt, und `M_REQUEST`, dass es eine *request* Nachricht ist.
Nachrichten von außerhalb können an den *network controller* gerichtet sein (z.B. Erfragen, welche Knoten aktiv sind) oder werden an die adressierten Knoten weitergeleitet um Dienste der Knoten in Anspruch zu nehmen. An welchen bzw. (durch Verwendung des Adresspatterns) welche Knoten eine Nachricht gerichtet ist, erkennt der *network controller* anhand der Knotennummer bzw. des Adresspatterns. Der *network controller* prüft nun zunächst, ob bereits ein Knoten unter der jeweiligen Knotennummer angemeldet ist. Ist dies der Fall, wird die Nachricht an den jeweiligen Knoten weitergeleitet. Ist kein Knoten unter der Knotennummer bekannt oder konnte die Nachricht nicht zugestellt werden, weil der Knoten ausgefallen ist, wird dies der *higher application layer* mitgeteilt.
Neben dem Verarbeiten und Weiterleiten von Nachrichten muss der *network controller* der *higher application layer* außerdem regelmäßig mitteilen, dass er noch verfügbar ist (im Detail vgl. 5.3.2).

5.3.1.2.3 Ausnahmefall

Im Folgenden wird der Ausfall von Netzwerkknoten behandelt. Der Ausfall von Komponenten der *higher application layer* bleibt wie in 5.3.1.1.3 unbeachtet.
Der *network* controller erkennt den Ausfall eines Knotens immer dann, wenn eine Nachricht an einen Knoten nicht zugestellt werden konnte. Jede Nachricht, die ein Knoten erhält, muss von

diesem daher bestätigt werden; dies geschieht hardwareintern durch das *auto acknowledge*. Das bedeutet, dass weder der *network controller* regelmäßig überprüfen muss, ob ein Knoten noch existiert, noch muss ein Knoten regelmäßig bekannt geben, dass er noch existiert. Das Erkennen eines ausgefallenen Knotens geschieht daher automatisch, wenn eine Nachricht an ihn nicht zugestellt werden konnte.
Ist ein Knoten ausgefallen und wird neugestartet, bevor der *network controller* den Ausfall mitbekommen hat, versucht er sich an einer noch als aktiv markierten Knotennummer anzumelden. Der *network controller* überprüft daher, ob der als aktiv markierte Knoten überhaupt noch existiert. Da diese Knotennummer von dem neugestarteten Knoten besetzt war, erhält der *network controller* keine Bestätigung des „aktiven" Knotens. Die Knotennummer ist freigeworden und der Knoten kann sich unter seiner alten Knotennummer anmelden.
Wurde ein neuer Knoten erfolgreich angemeldet, wird dies der *higher application layer* vom *network controller* mitgeteilt. Die Benachrichtigung über das Hinzukommen oder Wegfallen eines Knotens geschieht beides vom *network controller*. Dazu schickt der *network controller* eine *new* bzw. *dead* Nachricht sowie die jeweilige Knotennummer nach außerhalb.

5.3.2 Higher application layer

Die *higher application layer* kommuniziert mit dem *network controller* um Knotendienste in Anspruch zu nehmen. Sie setzt sich zusammen aus einem *network configurator*, der als Benutzerschnittstelle eine Übersicht über den aktuellen Zustand des Netzwerks gibt und Einstellmöglichkeiten an den Netzwerkknoten erlaubt, sowie einem *sound controller*, der die Interaktion zwischen Besucher und Installation realisiert.
Beide Komponenten nehmen die Dienste von sowohl Sensor- als auch Aktorknoten in Anspruch. Das bedeutet, dass beide alle Knoten kennen und vor Allem je nach Knotentyp wissen müssen, wie sie die unterschiedlichen Knotendienste nutzen können. Die Bekanntgabe der zugehörigen Schnittstellen der Knotendienste findet statisch während der Programmierung statt. Die Schnittstellen werden dann zur Laufzeit bei Neuanmeldung eines Knotens, welcher zu Beginn seine Dienste mitteilt, einem Knotentyp zugeordnet. Dies geschieht auf Grundlage von *OSC Address Patterns*, welche einen Knoten nach Typ unterscheiden lassen (bspw. alle Knoten die einen Knotendienst haben, dessen *OSC Type Tag String* auf das Adressmuster „`/node/[0-9]/sensor`" zutrifft, sind Sensorknoten).[26] So wissen *sound controller* und *network configurator* vorab anhand des *OSC Address Patterns*, wie die verschiedenen Knotentypen bedient werden können. Um eine (logische) Zuordnung bestimmter Klangstäbe und Sensoren mit bestimmten Knoten zu ermöglichen, erhält der *sound controller* zusätzlich die jeweilige Knotennummer sowie die Nummer des Klangstabs bzw. Sensors.
Das Hinzugekommen oder Weggefallen eines Knotens erfahren *network configurator* und *sound controller* von dem *network controller*. Dies geschieht durch Empfangen einer *new* bzw. *dead* Nachricht. Ist allerdings der *network controller* ausgefallen, müssen *network configurator* und *sound controller* dies selbstständig erkennen. Bei einer verbindungsorientierten Kommunikation wäre dies nicht weiter problematisch, jedoch findet eine verbindungslose Kommunikation statt.
Die Erkennung, ob der *network controller* ausgefallen ist, geschieht daher auf Basis eines *Watchdog Timers*. Diesen muss der *network controller* durch das Senden einer *alive* Nachricht in

26 Behandelt wird also lediglich der Spezialfall dieser Installation. Alternativ hätte - als allgemeine Lösung - der Knoten zur Laufzeit seinen Typ (Sensor-, Aktor- oder Hybridknoten) bekannt geben können und sich bspw. über eine vom Knotentyp abhängige Schnittstelle bedienen lassen. Ein Knoten müsste dann seine typabhängige Schnittstelle implementieren, die unabhängig von der Art der Sensoren bzw. Aktoren genutzt werden könnte. Dieser Ansatz schien jedoch der Flexibilität und der Möglichkeit zur individuellen Benutzung, die das OSC Protokoll bietet, entgegen zu wirken.

regelmäßigen Abständen zurücksetzen um den Ablauf des Timers zu verhindern. Läuft der Timer zu Ende, weil keine *alive* Nachricht empfangen wurde, scheint der *network controller* ausgefallen zu sein. Da eine Nachricht möglicherweise gesendet, aber nicht empfangen wurde, wird sicherheitshalber noch eine *alive* Anfrage an den *network controller* geschickt, die er mit einer *alive* Nachricht bestätigen muss. Bleibt auch diese Bestätigung aus, kann vom Ausfall des *network controllers* ausgegangen werden.
Wie die Komponenten der *higher application layer* sich während des Betriebs verhalten und in den genannten Ausnahmefällen reagieren, wird in den folgenden Unterkapitel 5.3.2.1 und 5.3.2.2 erläutert.

5.3.2.1 Network configurator

Für das Netzwerk wird eine grafische Benutzerschnittstelle bereitgestellt, die zur Übersicht, Konfiguration und Steuerung des WSANs dient. Diese ist in Sichten aufgeteilt, welche in 5.3.2.1.4 genauer beschrieben werden. Doch zunächst beschreibt 5.3.2.1.1 den Vorgang der *Initialisierung*, welcher nötig ist um den *network configurator* in *Betrieb* zu setzen (vgl. 5.3.2.1.2). Im Anschluss behandelt 5.3.2.1.3 die Ausnahmefälle.

5.3.2.1.1 Initialisierung

Nach Neustart richtet der *network configurator* zunächst im lokalen Netzwerk (ebenfalls auf IP-Adresse „127.0.0.1“) jeweils einen Port zum Empfangen und einen zum Senden von Nachrichten ein, wobei Empfangs- und Sendeports genau entgegengesetzt zu den des *network controllers* sind. Anschließend wartet der *network configurator* auf die *alive* Nachricht des *network controllers*. Trifft diese ein, wird der Watchdog Timer gestartet und beim *network controller* erfragt, welche Knoten aktiv sind. Daraufhin werden die Dienste sowie die aktuellen Konfigurationen aller aktiven Knoten angefordert[27]. Zuletzt wird jede Sicht auf den aktuellen Stand gesetzt und der Watchdog Timer gestartet. Der *network configurator* ist nun initialisiert und kann verwendet werden. Er wechselt in den Zustand Betrieb.

5.3.2.1.2 Betrieb

Während des Betriebs reagiert der *network configurator* auf Eingaben an der graphischen Benutzerschnittstelle und leitet die entsprechenden Kommandos an die jeweiligen Knoten weiter. Treffen Nachrichten vom *network controller* oder von Knoten ein, werden diese abhängig von der Nachricht unterschiedlich behandelt. Diese können z.B. angeben, welcher Sensortwert zuletzt gemessen wurde, dass der Watchdog Timer zurückgesetzt werden kann (*alive*) oder dass ein Ausnahmefall (z.B. der Ausfall eines Knotens) eingetreten ist.

5.3.2.1.3 Ausnahmefälle

Während des Betriebs muss der *network configurator* auf Änderungen des WSANs reagieren. Kommen neue Netzwerkknoten hinzu oder fallen welche aus, merkt er sich diese und aktualisiert die betroffenen Sichten (z.B. Markieren der betroffenen Knoten in der Netzwerksicht). Er sorgt auch dafür, dass keine Nachrichten an einen ausgefallenen Knoten gesendet werden (z.B. durch Deaktivieren/Einschränken von Sichten). War ein Knoten bereits angemeldet und wurde wieder

27 Die Bekanntgabe der Knotendienste geschieht durch OSC Nachrichten. Die enthaltenen Daten der OSC Nachrichten stellen die letzten Konfigurationen dar. Vgl. 5.3.1.1.1

neugestartet, werden ihm seine letzten Konfigurationen für Sensoren bzw. Aktoren erneut gesendet. Es kann nicht davon ausgegangen werden, dass ein Knoten sich diese merkt; er könnte ja durch einen anderen ersetzt worden sein.
Fällt der *network controller* aus (oder ist noch gar nicht gestartet), werden zunächst alle Sichten eingeschränkt (im Detail vgl. 5.3.2.1.4) und der Benutzer über den Ausfall benachrichtigt. Während des Ausfalls können keine Nachrichten an das WSAN geschickt werden. Erst wenn er (wieder) gestartet ist (also nach Empfangen der *alive* Nachricht), kann die Kommunikation zwischen *network configurator* und WSAN wieder stattfinden. Dazu wird zunächst beim *sound controller* erfragt, welche Knoten noch aktiv sind. Anschließend werden die letzten Konfiguration an die Knoten geschickt. Zwar sind diese im Normalfall im Knoten gespeichert, jedoch hätte ein Knoten währenddessen zurückgesetzt werden können.

5.3.2.1.4 Sichten

Der *network configurator* ist, um den in Kapitel 3.1.4 beschriebenen Anforderungen gerecht zu werden, in vier Sichten aufgeteilt.
Die erste Sicht ist die Netzwerksicht. Sie ist die Übersichtskarte über das Netzwerk und stellt die Knoten virtuell dar. Die Netzwerksicht ist immer sichtbar und zeigt je nachdem, welche der anderen Sichten gerade aktiv ist, ein anderes Verhalten.
Die anderen drei Sichten sind die Knotensicht , die Sensorsicht und die Aktorsicht , von denen genau eine zur Zeit aktiv ist. Es kann zwischen diesen Sichten hin und her gewechselt werden.
Die Knotensicht stellt Informationen über Knoten bereit und lässt Knotendienste in Anspruch nehmen. In der Sensorsicht können Einstellmöglichkeiten bezüglich einzelner Sensoren vorgenommen werden. Außerdem können hier Sensoren zu Messgruppen zusammengefasst und das Messverhalten aller Gruppen eingestellt werden. Die letzte Sicht ist die Aktorsicht . Diese ermöglicht das Konfigurieren von einzelnen Aktoren.
Die Kombination der ersten beiden Sichten (Netzwerksicht und Knotensicht) funktioniert unabhängig vom Knotentyp. Das heißt, dass sie für jeden beliebigen Knoten, der die Schnittstelle seiner Knotendienste bekannt gibt[28], genutzt werden kann. Die Sensor- und Aktorsicht setzen allerdings voraus, dass ein Knoten nach Knotentyp unterschieden werden können muss und die Schnittstelle seiner Dienste in Abhängigkeit vom Knotentyp bekannt ist. Es muss bekannt sein, wie Sensoren angesprochen und konfiguriert werden und wie selbiges für Aktoren geschieht. Dies wird, wie bereits erwähnt, bereits während der Programmierung festgelegt. Sensor- und Aktorsicht behandeln daher nur den Spezialfall für diese Installation. Netzwerksicht und Knotensicht können jedoch allgemein für jegliche Anwendung des WSANs genutzt werden.

5.3.2.1.4.1 Netzwerksicht

In der Netzwerksicht, werden Knoten graphisch in einer Art Übersichtskarte angezeigt. Diese wird durch ein zwei-dimensionales Koordinatensystem dargestellt. Knoten erscheinen hier als Rechteck und können - der realen Position nachempfunden – innerhalb des Koordinatensystems platziert werden.
Knoten werden zudem in einer bestimmten Farbe dargestellt, die den den jeweiligen Zustand des Knotens angibt. Diese Zustände sind *neu*, *aktiv* und *inaktiv*:
Meldet sich ein neuer Knoten an, erscheint dieser zunächst als *neu* (blau) im Koordinatensystem. Wird der Knoten positioniert (oder in den anderen Sichten verwendet), wechselt er in den Zustand *aktiv* (grün). Fällt ein Knoten aus, wechselt er in den Zustand in *inaktiv (rot)* und bleibt solange dort, bis er entweder durch Neustart wieder *aktiv* wird oder manuell entfernt wurde. Die

28 Dies geschieht während der *Initialisierung* eines Knotens durch Mitteilen seiner Dienste über OSC Nachrichten

Netzwerksicht dient außerdem dem schnellen Selektieren von Knoten und macht das Bedienen der anderen Sichten benutzerfreundlicher.

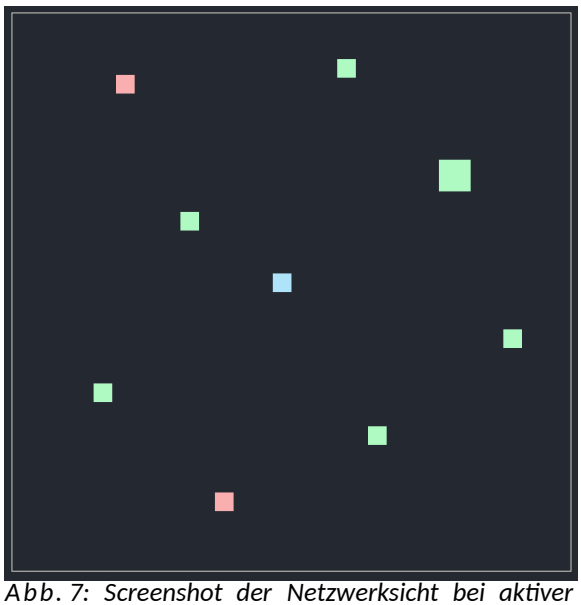

Abb. 7: Screenshot der Netzwerksicht bei aktiver Knotensicht

Abbildung 7 zeigt beispielhaft einen Screenshot der Netzwerksicht bei aktiver Knotensicht. Dargestellt werden ein *neuer* (mittig), zwei *inaktive* sowie mehrere *aktive* Knoten. Der selektierte Knoten (rechts oben) wird vergrößert angezeigt.
Wie bereits angedeutet, weist die Netzwerksicht je nachdem, welche der anderen Sichten ausgewählt ist, ein anderes Verhalten auf:
Ist die Knotensicht aktiv, erscheinen alle Knoten in der Netzwerksicht. Hier können nun alle Knoten in allen Zuständen selektiert und positioniert werden. Wurde ein Knoten ausgewählt, wird dieser in der Knotensicht angezeigt. Handelt es sich um einen *inaktiven* Knoten, muss die Knotensicht die Kommunikation mit diesem verbieten.
Bei aktiver Sensorsicht werden alle Sensorknoten sowie ihre Sensoren angezeigt. Sensoren können zudem virtuell platziert werden und zeigen das letzte Messergebnis an. Da Sensoren zu Messgruppen zusammengefasst werden können sollen (vgl. 3.1), bietet die Netzwerksicht die Möglichkeit mehrere Sensoren zu selektieren und zu gruppieren. Gruppen von Sensoren werden erkennbar in der Netzwerksicht dargestellt und erscheinen außerdem als solche in der Sensorsicht. Das Bedienen von Sensoren, dessen Knoten *inaktiv* sind, ist nicht möglich.
Ist die Aktorsicht aktiv, werden lediglich Aktorknoten sowie ihre Aktoren (Klangstäbe) angezeigt. Diese können – wie Sensoren auch – platziert werden. Selektierte Aktorknoten werden in der Aktorsicht angezeigt, so dass ihre Klangstäbe konfiguriert werden können. Das Bedienen von Klangstäben, dessen Knoten *inaktiv* sind, ist nicht möglich.
Unabhängig von der Auswahl der Sicht wird bei Ausfall des *network controllers* die Netzwerksicht unbenutzbar gemacht und alle Knoten auf *inaktiv* gesetzt. Erst nach Neustart des *network controllers* (Empfang einer *alive* Nachricht) wird die Netzwerksicht wieder benutzbar und alle noch vorhandenen Knoten werden wieder *aktiv*.

5.3.2.1.4.2 Knotensicht

Die Knotensicht gibt den aktuellen Zustand (*neu*, *aktiv* oder *inaktiv*) eines ausgewählten[29] Knotens wieder und listet alle Dienste dieses Knotens auf. Dazu werden die *OSC Address Patterns* sowie der *OSC Type Tag String* der Knotendienste, welche der Knoten nach Neustart bekannt gibt, angezeigt. Des Weiteren können diese Dienste zu Testzwecken und zur Funktionsüberprüfung ausgeführt werden. So erscheint für jedes Argument eines Dienstes ein eigenes Textfeld, welches die Eingabe von Werten erlaubt. Da die Argumente einen bestimmten Datentyp verlangen und diese gewissen Beschränkungen unterliegen, werden nicht alle Eingaben akzeptiert (so kann bspw. auch der höchste Wert einer vorzeichenbehafteten 32-Bit Integer Zahl nicht von einer +2.147.483.648 überschritten werden).
Im Falle eines ausgefallenen (inaktiven) Knotens wird dieser in der Knotensicht zwar angezeigt, das Anfordern von Diensten wird jedoch unbenutzbar gemacht. Ist der *network controller* ausgefallen, bleibt die Knotensicht bis zum Neustart des *network controllers* unbenutzbar.

5.3.2.1.4.3 Sensorsicht

In der Sensorsicht werden alle Sensorknoten sowie die zugehörigen Sensordienste aufgelistet. Sie ermöglicht das Konfigurieren einzelner Sensoren und lässt Einstellungen am Messverhalten vornehmen. So können bspw. Messperiode und *threshold* (vgl. 5.3.1.1.2) für Sensoren eingestellt werden. Dies ist zwar auch in der Knotensicht möglich (da sie alle Dienste auflistet und ausführen lässt), jedoch bietet die Sensorsicht zum Einen eine benutzerfreundlichere Bedienung der Sensordienste und ermöglicht zum Anderen das Speichern und Laden der Konfigurationen.
Um Interferenzen bei Sensormessungen zu vermeiden können zudem einzelne Sensoren zu Messgruppen zusammengefasst werden. Eine Messperiode kann in diesem Fall nicht für jede Gruppe einzeln, sondern - um Überschneidungen bei unterschiedlichen Periodendauern zu vermeiden - nur global für alle Gruppen eingestellt werden. Gruppen messen nun zwar in der selben Messperiode, werden aber so konfiguriert, dass die Messperioden der einzelnen Gruppen versetzt nacheinander starten.[30] Das Speichern und Laden eingestellter Sensorgruppen-Konfigurationen ist auch möglich.
Bei Ausfall eines Sensorknotens müssen seine Konfigurationen erhalten bleiben. Das Konfigurieren dieses *inaktiven* Knotens ist jedoch nicht möglich (die Kommunikation mit diesem Knoten wird unterbunden). Nach Neustart des Sensorknotens wechselt dieser in den Zustand *aktiv*. Alle seine vorigen Konfigurationen werden ihm nun automatisch zugesandt.
Im Falle eines ausgefallenen *network controllers* wird die Sensorsicht unbenutzbar gemacht. Lediglich das Speichern von Konfigurationen ist noch möglich.

5.3.2.1.4.4 Aktorsicht

Die Aktorsicht stellt - analog zur Sensorsicht – alle Aktorknoten sowie die zugehörigen Aktordienste dar. Sie ermöglicht das Konfigurieren einzelner Aktoren[31] sowie das Speichern und Laden dieser Konfigurationen.

---

29 Das Auswählen findet durch das Anklicken eines Knotens in der Netzwerksicht statt. Zusätzlich findet sich in der Knotensicht eine Dropdown-Liste, die ein Selektieren aus allen Knoten ermöglicht.

30 Der *network configurator* startet dazu die Messperioden der jeweiligen Gruppenmitglieder, so dass alle Gruppen nacheinander und in gleichmäßigen, auf die Anzahl aller Gruppen zeitlich verteilten Abständen messen. Damit der *network controller* erkennen kann, ob die Sensoren zur (für ihre Gruppe) richtigen Zeit gemessen haben, muss ein Sensorknoten den Zeitpunkt der Messung mitschicken. Findet eine Messung außerhalb eines bestimmten Toleranzmaßes statt (z.B. wenn die Zeit einer Sensormessung näher an der einer anderen Gruppe liegt oder zwei Sensormessungen unterschiedlicher Gruppen zu nah beieinander liegen), startet der *network configurator* alle Messperioden erneut.

Alle *inaktiven* Aktorknoten sind von der Bedienung ausgeschlossen. Ihre gemerkten Konfigurationen werden ihnen bei Wiedereintritt automatisch zugesandt.
Im Falle eines ausgefallenen *network controllers* wird die Sensorsicht unbenutzbar gemacht. Lediglich das Speichern von Konfigurationen ist noch möglich.

5.3.2.2 Sound controller

Der *sound controller* ist die Steuereinheit, die die Interaktion zwischen Besucher und Installation realisiert. Dieser wird logisch in zwei Komponenten aufgeteilt: die Komposition und die Klangauslösung.
In der Komposition werden gewisse Abläufe und Regeln festgelegt, die - abhängig von Sensordaten – Notenmuster[32] erzeugen. Die Regeln sollen einem Regelwerk entnommen werden können, welches mathematische Rechenoperatoren und Operatoren imperativer Programmierung beinhaltet. Es soll möglich sein, diese Regeln während der laufenden Installation zu ändern um ein „live“ Komponieren zu ermöglichen. Die Komposition soll zudem zeit- bzw. taktabhängige Strukturen erlauben können (vgl. Kapitel 3.2.1).
Die Klangauslösung muss diese Komposition verstehen und dafür sorgen, dass die dort erzeugten Noten in Klang umgesetzt werden. Dazu müssen die jeweiligen Aktorknoten benachrichtigt werden diejenigen Klangstäbe anzuschlagen, die den Noten zugewiesen wurden (vgl. 3.2.2). Dabei kann vorausgesetzt werden, dass die zur Verwendung von Aktoren und Sensoren notwendigen Schnittstellen dem *sound controller* vor Laufzeit bekannt sind (vgl. 5.3.2). Gleiches gilt für die Zuordnung von Noten zu Klangstäben.
Zur Realisierung des *sound controllers* bedarf es daher einer Softwarelösung, die den in Kapitel 3.2 beschriebenen Anforderungen gerecht wird und die Komposition und Klangauslösung umsetzen lässt. Wie sich Komposition und Klangauslösung während des Betriebs des *sound controllers* und in Ausnahmefällen verhalten, wird in 5.3.2.1.4 näher beschrieben. Zunächst werden jedoch die Initialisierung (5.3.2.2.1), der Betrieb (5.3.2.2.2) sowie Ausnahmefälle (5.3.2.2.3) des *sound controllers* im Allgemeinen erläutert.

5.3.2.2.1 Initialisierung

Nach Starten des *sound controllers* richtet dieser – wie der *network configurator* auch - zunächst (im lokalen Netzwerk auf IP-Adresse „127.0.0.1“) einen Port zum Empfangen und einen zum Senden von Nachrichten ein. Sende- und Empfangports sind wieder entgegengesetzt zu den des *network controllers*.
Anschließend prüft der *sound controller* nach dem Starten (wie der *network configurator* auch), welche der angegebenen Knoten bereits vorhanden sind, und merkt sich diese (durch Setzen auf *aktiv* oder *inaktiv*). Sind zu Beginn nicht alle Sensorknoten *aktiv*, greifen Standardwerte für die Sensoren *inaktiver* Knoten.
Zuletzt wird der Watchdog Timer zur Ausfallerkennung des *network controllers* gestartet. Der *sound controller* ist nun betriebsbereit. (Die Wiedergabe der Komposition wird noch nicht gestartet.)

31 Die verwendeten Aktoren sind Hubmagnete; im Detail vgl. 5.4.1.1.2. Diese werden durch Setzen einer minimalen und maximalen Einschaltdauer konfiguriert.
32 „Note“ als Repräsentant eines bestimmten Klangstabs; vgl. Definition in 3.2.

5.3.2.2.2 Betrieb

Während des Betriebs können die Kompositionsregeln gesetzt, geändert, gespeichert oder geladen werden. Dabei können nur diejenigen Sensoren und Noten genutzt werden, die bereits durch Vorgabe der Knotennummer und der jeweiligen Schnittstelle der Knotendienste bekannt sind. Ein Sensor bzw. eine Note ist dann eindeutig definiert, wenn die Knotennummer des Knotens, der ihn bedient sowie die Nummer des Sensors bzw. Klangstabs (innerhalb dieses Knotens) bekannt ist. Die Zuordnung eines Sensors bzw. einer Note zu einem Klangstab kann während des Betriebs geändert werden. Die Wiedergabe der Komposition kann durch Benutzereingabe gestartet oder gestoppt werden. Das Starten der Wiedergabe erfolgt nur, wenn der *network controller* vorhanden ist.
Um dies zu erkennen nimmt der *sound controller* während des Betriebs außerdem die *alive* Nachrichten des *network controllers* entgegen und setzt ggf. den Watchdog Timer zurück.

5.3.2.2.3 Ausnahmefälle

Kommen im laufenden Betrieb (neue) Knoten hinzu oder fallen aus, werden diese ebenfalls auf *aktiv* bzw. *inaktiv* gesetzt. Komposition und Klangauslösung können nur mit *aktiven* Knoten kommunizieren und treffen im Ausnahmefall geeignete Maßnahmen, damit die Installation weiterlaufen kann (im Detail siehe 5.3.2.2.4.1 und 5.3.2.2.4.2).
Bei Ausfall des *network controllers* wird die Wiedergabe der Komposition angehalten und bei Wiedereintritt fortgesetzt.

5.3.2.2.4 Komponenten

Der *sound controller* wird logisch in zwei Komponenten aufgeteilt: Komposition und Klangauslösung.

5.3.2.2.4.1 Komposition

Die Komposition erzeugt - durch Anwenden von Regeln auf Sensordaten - einen zeitlichen Ablauf von Noten. Dazu werden die Sensorknoten so konfiguriert[33], dass die Komposition periodisch Sensordaten von aktiven Sensorknoten erhält. Wie und durch welche Regeln die Komposition für die Installation geschieht, wird im Rahmen dieser Arbeit nicht ausgeführt. Wie jedoch eine solche Notengenerierung aussehen könnte, zeigt folgendes Beispiel:

33 Die Konfiguration erfolgt über den *network configurator* und ist nicht Teil des *sound controllers*.

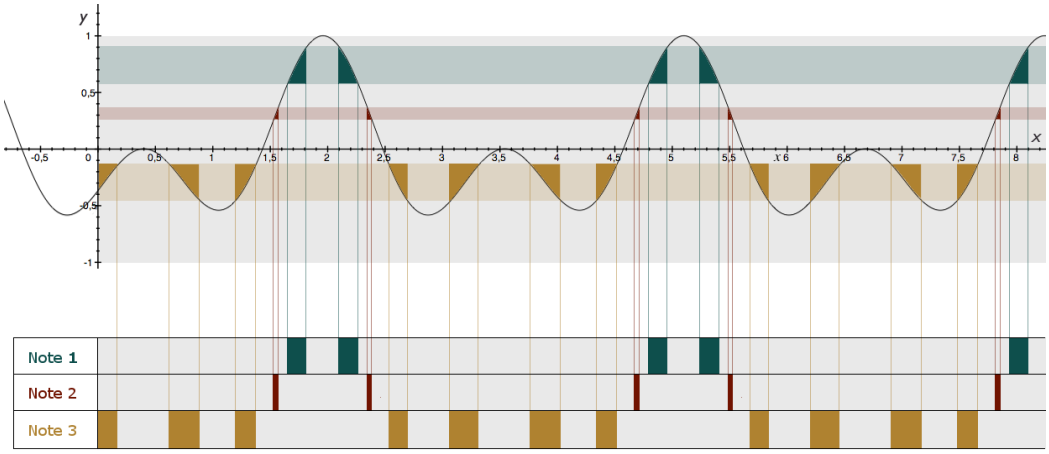

Abb. 8: Taktgenerierung mittels mathematischer Funktion

Es wird der Abschnitt einer mathematischen Funktion (siehe oberer Teil von Abb. 8) genutzt um Noten (Notenwert/Dauer einer Note; Position etc.) zu erzeugen. Dazu werden Noten bestimmten Wertebereichen (y-Achsenabschnitt) zugewiesen und über die Zeit betrachtet (x-Achsenabschnitt) auf einen musikalischen Takt übertragen. Sobald sich die Funktion in dem Wertebereich einer Note befindet beginnt die Note, sobald die Funktion den Wertebereich verlässt endet sie. Wenn nun die Funktion in einem bestimmten Tempo abgespielt werden würde, könnten so fortlaufend Noten erzeugt werden. Abhängig von der Funktion würde ein gleichbleibendes Muster (wie bspw. oben durch Überlagerung einer Sinus und Cosinus Funktion) oder eine eher zufällige Anordnung von Noten (z.B. bei Rauschen) erzeugt werden. Die Funktion könnte aber auch statt fortlaufend abgespielt zu werden innerhalb eines bestimmten Zeitbereichen wiederholt werden. Dies würde selbst eine zufällige Anordnung nicht mehr allzu zufällig erscheinen lassen. Um eine Interaktion zu ermöglichen, könnten nun bspw. Sensorwerte auf bestimmte Funktionsparameter abgebildet werden. Bei Sensorwertänderung würde sich dann der Funktionsverlauf und folglich der zeitliche Verlauf von Noten ändern.
Im Allgemeinen reagiert die Komposition nur auf Änderung der Sensorwerte. Ist ein Sensorknoten inaktiv, findet demnach keine Sensorwertänderung mehr statt. Zwar findet nun auch keine Reaktion der Komposition statt, jedoch können die letzten Sensorwerte nach wie vor verwendet werden. Die Komposition greift bei Ausfall eines Sensorknotens daher auf die letzten Sensorwerte zurück und kann weiterhin funktionieren ohne angehalten werden zu müssen. [34]
Dies kann sich - in Abhängigkeit von den Regeln - unterschiedlich auf die Installation auswirken: Es könnte bspw. sein, dass zuletzt Messwerte geliefert wurden, die viel Auswirkung auf das Klangbild hatten. Dieser Zustand würde so lange anhalten, bis der Sensorknoten nach Austausch/Neustart wieder neue Werte liefert - selbst wenn in Wirklichkeit keine Interaktion mehr stattfände (z.B. da der Besucher bereits aus dem vom Sensor erfassbaren Bereich verschwunden wäre). Eine Fehlfunktion der Installation wäre also offensichtlich.
Die Regeln könnten allerdings auch ein entgegengesetztes Verhalten vorgeben, welches nur bei stark variierenden Sensorwerten eine Auswirkung auf das Klangbild hätte. Da bis zum Austausch/Neustart der Sensorknoten jedoch nur konstante Werte benutzt werden können,

34 Der Ausfall von Aktorknoten ist für die Komposition irrelevant, da sie nicht mit diesen kommuniziert. Ausgefallene Noten sollen nicht durch verfügbare ersetzt werden (vgl. 3.2.1). Zwar entstünden nun Lücken im Klangbild, jedoch könnten durch geschickte Anordnung der Klangstäbe (z.B. keine aufeinander folgenden Tonhöhen an einem Knoten) die Noten so verteilt sein, dass ein Ausfall weniger auffällig wäre.

würde, selbst wenn in Wirklichkeit eine Interaktion stattfände, keine Reaktion auf das Klangbild gezeigt werden. Auch in diesem Fall wäre eine Fehlfunktion der Installation offensichtlich.
Zwar soll dem Besucher der Ausfall von Knoten weitestgehend verborgen bleiben (vgl. 3.2), jedoch ist die Auswirkung, die der Ausfall von Sensorknoten hätte, abhängig von den in der Komposition verwendeten Regeln. Eine für alle Kompositionen funktionierende Fehlerbehandlung ist daher nicht möglich, sondern muss für jede Komposition individuell berücksichtigt werden.

5.3.2.2.4.2 Klangauslösung

Die Klangauslösung ist zuständig für die Benachrichtigung der Aktorknoten zum Anschlagen der zu den Noten zugehörigen Klangstäbe. Dazu schickt die Klangauslösung eine OSC Nachricht, welche die für die Note bestimmte Klangstabnummer enthält, an einen Knoten mit der für die Note bestimmten Knotennummer. Die Verknüpfung von Noten und Klangstäben ist bereits bekannt und kann vorausgesetzt werden (vgl. 5.3.2).
Die Klangauslösung ist unabhängig von Sensorknoten und muss daher vor Abschicken der Nachricht lediglich prüfen, ob die jeweiligen Aktorknoten *aktiv* oder *inaktiv* sind. Ist der *network controller* ausgefallen, findet kein Nachrichtenverkehr mehr statt.

5.4 Aktueller Stand der Hardware-Software-Lösung

Im Folgenden wird der aktuelle Stand der Hardware-Software-Lösung vorgestellt. Dazu werden zunächst die Wahl der Hardware sowie die Softwareprogramme beschrieben, welche für die Komponenten verwendet werden. Anschließend wird aufgezeigt, wie weit diese bereits implementiert sind.

5.4.1 WSAN

Das WSAN setzt sich zusammen aus dem *network controller* und mehreren Knoten. Beide müssen durch ein Computersytem gesteuert werden, welches so erweiterbar ist, dass einerseits eine kabellose Kommunikation zwischen beiden ermöglicht und andererseits die Rolle des *network controllers* als gateway sowie die der *nodes* als Sensor- bzw. Aktorknoten umgesetzt werden kann. Um den Entwicklungsaufwand gering zu halten sollte dieses System für beide auf der selben Entwicklungsplattform basieren. Es sollte zudem klein, kostengünstig und energieeffizient sein.
Arduino[35] ist eine solche Entwicklungsplattform, die sowohl eine Hardwarelösung als auch eine Softwarelösung stellt und typischerweise in Interaktiven Projekten Verwendung findet. „Arduino is a tool for making computers that can sense and control more of the physical world than your desktop computer.“ (vgl. http://arduino.cc/en/Guide/Introduction) Arduino bietet den Vorteil, dass durch verschiedene Hardware Boards (typ. Mikrocontrollerboard mit digitalen/analogen Ein-/Ausgängen) eine je nach Anwendungszweck möglichst geeignete Hardwarelösung gewählt werden kann. So könnte beispielsweise der *network controller* auf einer leistungsstärkeren und die *nodes*, die in einer recht hohen Anzahl (von rund 10-20 Stück) zum Einsatz kommen, auf einer kostengünstigeren Variante laufen und trotzdem die selbe Software nutzen. Zur Programmierung dieser Boards liefert Arduino eine Entwicklungsumgebung, die (dank ihrer Community) von zahlreichen Bibliotheken unterstützt wird und daher auf bereits vorhandene Implementierungen von bspw. Sensortreibern oder auch dem OSC Protokoll (s.o.) zurückgreifen lässt.

35 http://arduino.cc/

Eine Alternative zu Arduino Microcontrollern ist der Raspberry Pi Microprocessor[36]. Im Gegensatz zum Arduino ist er von Leistung und Hardwareausstattung vergleichbar mit einem PC. Zwar sind beide (Arduino und Raspberry Pi) mit CPU, Memory, Timer und I/O Pins ausgestattet, jedoch stellt der Raspberry Pi diese (bis auf letztere) in einer leistungsstärkeren Variante: So ist z.B. der Arduino Uno mit einem 16MHz Prozessor (AVR ATmega328p) und der Raspberry Pi Modell B mit einem (über 40mal schnelleren!) 700MHz Prozessor (ARM11) ausgestattet. Der Raspberry Pi kann deshalb für solche Anwendungen genutzt werden, bei denen viele Daten verarbeitet werden müssen (wie z.B. Bilddatenverarbeitung). Arduino hingegen ist dazu konzipiert physische Hardware (wie Sensoren, Motoren, LEDs, etc.) zu bedienen und bietet dank des Microcontrollers eine direkte I/O Schnittstelle. Zwar sind Arduinos mit leistungsschwächeren Prozessoren bestückt, verbrauchen aufgrund dessen jedoch auch weniger Strom.
Da das WSAN für die Installation keine großen Datenmengen (wie Videodatenstreams, etc.) verarbeiten, sondern lediglich das Auslesen von Sensoren und das Ansteuern von Klangstäben bewerkstelligen muss, sollte dieses mit Hilfe von Arduinos umgesetzt werden. Auch die Netzwerkknoten, die kabellos und daher energieeffizient betrieben werden müssen, lassen auf die Wahl von Arduinos schließen. Zwar sollte der *network controller* auf eine leistungsstarke Plattform aufbauen, jedoch lässt sich diese in einer der zahlreichen Arduino Boards finden.
D a *nodes* und *network controller* miteinander kommunizieren, wird zudem ein Funkmodul benötigt, welches die kabellose, bidirektionale Kommunikation zwischen beiden ermöglicht. Sogenannte *radio transceiver* sind solche Sender-Empfänger-Einheiten, die im Radiofrequenzbereich (z.B. 2,4 Ghz) arbeiten. Ein Beispiel eines solchen radio transceivers geben die ZigBee Radiomodule. „ZigBee is the only standards-based wireless technology designed to address the unique needs of low-cost, low-power wireless sensor and control networks in just about any market." (vgl. http://www.zigbee.org/About/AboutTechnology/ZigBeeTechnology.aspx)
ZigBees geben zudem in Benutzung mit Arduinos eine robuste und leicht zu verwendende Schnittstelle zur kabellosen Kommunikation und zum Aufbau eines kabellosen Netzwerks. Sie sind jedoch für die benötigte Anzahl an Netzwerkknoten (10-20) zu kostenintensiv[37].
Eine weitaus günstigere Variante stellen die nRF24l01+ Radiomodule von Nordic Semiconductor[38]. Diese haben zum Einen eine hohe Übertragungsrate (von bis zu 2 Mbps) und können zum Anderen (durch Nutzung eines Power-Down Modus) energieeffizient betrieben werden. Die nRF24l01+ Radiomodule stellen eine einfache Sender/Empfänger Einheit dar, die eine Implementierung der Netzwerkstruktur erfordern. Da das Netzwerk jedoch (nur) in Sterntopologie aufgebaut sein soll, ist der Entwicklungsaufwand hierfür schätzungsweise gering[39].
Zwar bieten die ZigBee Module eine robuste und feature-reiche Technologie sowie eine bereits vorhandene Netzwerkstruktur, jedoch stellt das nRF24l01+ Radiomodul aufgrund der geringen Stückkosten (auch wenn theoretisch der zusätzliche Entwicklungsaufwand gegenzurechnen wäre) eine geeignete Wahl.
Im Folgenden (Abb. 9) wird nun die Hardware-Konfiguration abgebildet, auf welche der *network controller* und die Netzwerkknoten aufgebaut sind:

36 http://www.raspberrypi.org/
37 ca. 20 € und aufwärts, siehe z.B. http://www.exp-tech.de/Shields/XBee-Module-ZB-Series-2-XB24-Z7WIT-004.html
38 Im Detail siehe https://www.nordicsemi.com/kor/Products/2.4GHz-RF/nRF24L01P; Stückpreis eines auf dem nRF24l01+ basierenden Funkmoduls liegt bei ca. 3€, z.B. http://www.exp-tech.de/Shields/ITead-Studio-2-4G-Wireless-nRF24L01-Module.html
39 Die Implementierung der Netzwerkstruktur baut auf dem RF24 Arduino Treiber (vgl. https://github.com/TMRh20/RF24) auf. Dieser stellt eine Optimierung des ursprünglichen und wohl weit verbreitetsten Arduino Treibers für das nRF24l01+ Modul dar (vgl. https://github.com/maniacbug/RF24).

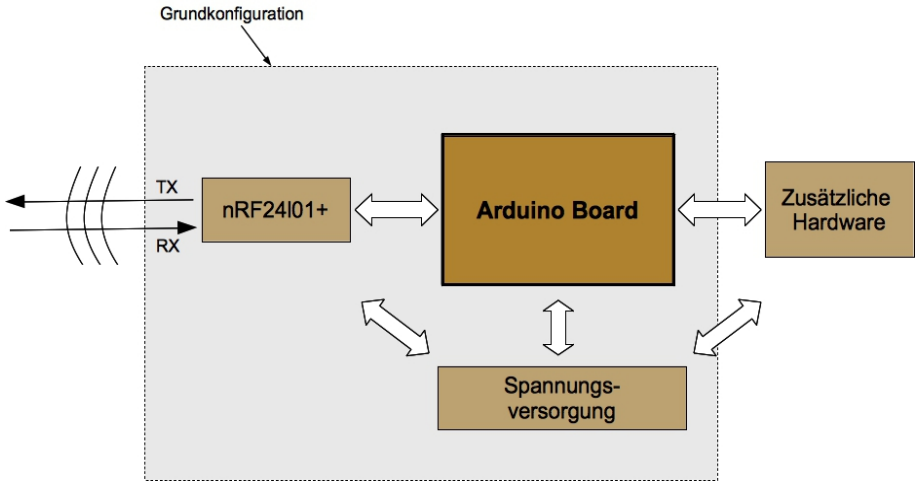

Abb. 9: Hardware-Konfiguration von network controller und Netzwerkknoten

Abgebildet ist hier zum Einen die oben beschriebene Grundkonfiguration und zum Anderen zusätzliche Hardware, die zwischen *network controller* und Netzwerkknoten variiert. Diese wird im Folgenden beschrieben.

5.4.1.1 Node

Ein *node* wird durch einen Arduino Pro Mini 5V/16MHz[40] gesteuert. Dieser stellt eine der kleinsten und günstigsten[41] Variante der Arduino-Boards dar. Er ist mit einem ATmega328p ausgestattet, welcher neben der Ansteuerung des nRF24l01+ Radiomoduls genügend analoge Eingänge sowie digitale Ein- und Ausgänge bietet um Sensoren oder Aktoren zu bedienen.
Der folgende Programmcode demonstriert, wie ein Knoten programmiert und beispielhaft um den Knotendienst aus Kapitel 5.2 erweitert werden kann.

40 http://arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardProMini
41 ca. 10 € ; http://www.exp-tech.de/Mainboards/Arduino-Pro-Mini-328---5V-8MHz.html

```
#include „Node.h“


// Erzeugt Radiomodul fuer Pin 9, 10 an SPI Schnittstelle
RF24 radio(9, 10);

// Erzeugt einen Knoten
Node node(radio);

// die auszufuehrende Funktion des Knotendienstes
void chime(OSCMessage &msg, int addrOffset);



void setup(){

      // Hinzufuegen eines Knotendienstes
      node.addCallback(„/chime“, chime);


      // Anmeldung beim network controller als Knoten 2
      node.begin(2);

}

void loop(){

      // update regelmaeßig um Nachrichten zu Empfangen
      node.update();
}

void chime(OSCMessage &msg, int addrOffset){

      // hole chime
      int chime_pin = msg.getInt(0);

      // versorge den Hubmagneten mit Spannung..
      digitalWrite(chime_pin, HIGH);

      // .. warte 20 ms ..
      delay(20);

      // .. und unterbreche die Spannungsvesorgung
      digitalWrite(chime_pin, LOW);

}
```

Abb. 10: Beispiel Programmierung eines Aktorknotens

Der in Abb. 10 dargestellte Code stellt lediglich das Beispiel eines primitiven Aktorknotens dar und dient nur zur Veranschaulichung. Auf eine Fehlerprüfung (z.B. ob die Anmeldung beim *network controller* fehlschlug oder ob es sich bei `chime_pin` möglicherweise um einen ungültigen Pin handelt) wurde deshalb nicht eingegangen.
In den nachfolgenden Unterkapiteln werden die aktuell für die Installation verwendeten Sensoren und Aktoren beschrieben.

5.4.1.1.1 Sensoren

Für die Installation werden Berührungssensoren verwendet, welche eine Interaktion mit Hilfe von berührungsempfindlichen Flächen erlauben. Diese werden auf dem Boden verteilt und lassen ein

Umherwandern der Besucher erkennen. Sie bieten den Vorteil, dass sie leicht zu installieren sind und ihre erfassbaren Bereiche (die berührungsempfindlichen Flächen) in der äußeren Form änderbar und gestaltbar sind. Ein Nachteil ist, dass die Sensoren durch Berührung lediglich „aktiviert“ oder „deaktiviert“ werden. Jedoch gibt es einige Hardwarelösungen, die eine Varianz in der Stärke der Berührung ermitteln oder bereits bei Näherung reagieren können. Ein Sensorchip, der eine solche Berührungserkennung ermöglicht, ist *freescale's* MPR121[42]. Der MPR121 kann bis zu 12 berührungsempfindliche Flächen[43] bedienen und lässt die Empfindlichkeit der Sensoren konfigurieren.

5.4.1.1.2 Aktoren

Der Anschlag der Klangstäbe funktioniert auf Basis von elektronischen Hubmagneten. Diese lassen bei anliegender, elektrischer Spannung einen am sog. Tauchkern montierten Druckstift nach vorne schnellen. Liegt keine Spannung am Magneten an, zieht eine Rückholfeder den Tauchkern (inkl. Druckstift) zurück. Um einen Klangstab zum Schwingen zu bringen darf dieser nur kurz angeschlagen werden. Der Hubmagnet darf deshalb nur für eine bestimmte Zeitspanne eingeschaltet sein. Ist die Einschaltdauer zu kurz erreicht der Druckstift den Klangstab nicht. Ist die Einschaltdauer zu lang, drückt der Druckstift zu lange gegen den Klangstab, so dass seine Schwingung stark gedämpft wird. Es wäre nur ein kurzer, lauter Anschlag zu hören.
Verwendet wird der HMF-2016d.002[44] der Firma *Tremba*. Dieser Hubmagnet ermöglicht einen kraftvollen (lauten) Anschlag bei vergleichsweise niedrigem Stromfluss (gemessen ca. 170 mA); das Vorgänger Modell[45] konnte zwar mit einer geringen Spannung betrieben werden, verbrauchte jedoch knapp 1A und erzeugte einen deutlichen leiseren Anschlag!
Nachfolgend zeigt Abb. 11 den Prototyp eines Klangstabs. Dieser setzt sich zusammen aus dem Hubmagneten (oben), einer Halterung (mittig) und einer Aluminiumstange (unten), die vom Hubmagneten angeschlagen Klang erzeugt.

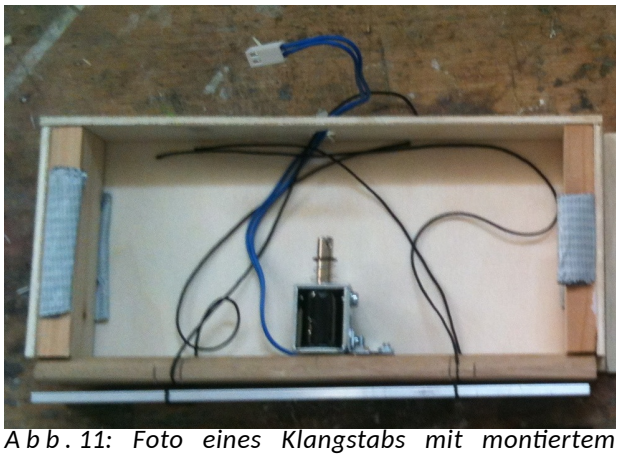

Abb. 11: Foto eines Klangstabs mit montiertem Hubmagneten

42 http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPR121
43 Diese müssen aus einem elektrisch leitfähigen Material bestehen.
44 http://tremba.de/hubmagnete/db-hubmagnete-HMF-2016d.002.pdf
45 http://www.exp-tech.de/Servos-und-Motoren/Solenoid-5v-small.html)

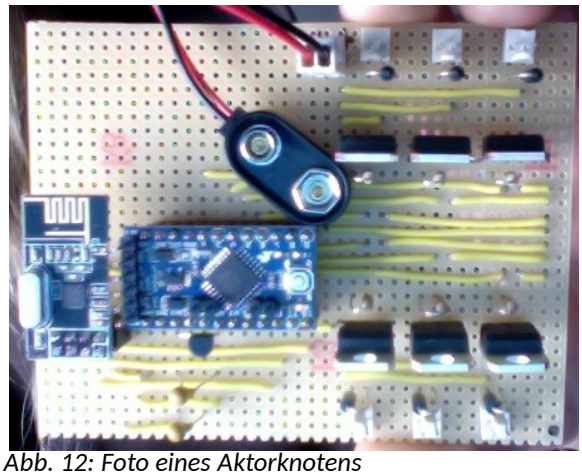

Abb. 12: Foto eines Aktorknotens

Abb. 12 zeigt das Foto eines Aktorknotens. Zu sehen ist das Funkmodul (links), der Arduino (mittig), ein Anschluss für eine 9V Blockbatterie und sechs Steckplätze für die Hubmagnete sowie zusätzlicher Bauteile (wie Transistoren, Widerstände und Dioden) zur Ansteuerung dieser Magnete.

5.4.1.2 Network controller

Der *network controller* basiert auf dem arduinokompatiblen Teensy 3.1[46]. Teensy ist ein Mikrocontroller Board, welches (in dieser Variante) um ein Vielfaches schneller ist als die Arduino Pro Mini's der *nodes* und Dank der USB Schnittstelle eine schnelle Kommunikation mit der *higher application layer* erlaubt: „Currently best performance is achieved with Arduinos with built-in USB Serial, i.e. Teensy 3.0, Teensy 2.0 and 2.0++ and Leanardo variants (12Mbps max)“ (vgl. Mann 2013)
Der Austausch von OSC Nachrichten mit der *higher application layer* findet deshalb kabelgebunden und seriell über die USB Schnittstelle des Teensy Boards statt. Da die Kommunikation jedoch nicht direkt mit dem lokalen Netzwerk der *higher application layer* stattfinden kann, wird eine zusätzliche Software verwendet, welche Nachrichten des *network controllers* in UDP Pakete verpackt und ins lokale Netzwerk sendet sowie umgekehrt Nachrichten aus dem lokalen Netz an die serielle Schnittstelle weiterleitet. Diese ist in der beiliegenden DVD unter „/Quellcode/SLIPSerialToUDP/src“ zu finden.
Eine zusätzliche Hardware benötigt der *network controller* nicht.

5.4.2 Network configurator

Die Realisierung des *network configurators* findet auf Basis der *Processing*[47] Entwicklungsumgebung statt. Diese ermöglicht durch Unterstützung sämtlicher Bibliotheken [48] eine simple Umsetzung des *network configurators* als graphischen Benutzerschnittstelle. So bietet *ControlP5*[49], eine GUI Bibliothek für Processing, sämtliche GUI Bausteine (wie Dropdown-Listen,

46 https://www.pjrc.com/teensy/
47 http://processing.org/
48 http://processing.org/reference/libraries/
49 http://www.sojamo.de/libraries/controlP5/

Slider, etc.). Des Weiteren wird *oscP5*[50] verwendet – eine Bibliothek, welche das OSC Protokoll für Processing implementiert und die Kommunikation mit dem WSAN möglich macht.

5.4.3 Sound controller

Die Umsetzung des *sound controllers* findet mit Hilfe von *SuperCollider* statt. SuperCollider ist in der Dokumentation folgendermaßen beschrieben:

> „SuperCollider is an environment and programming language for real time audio synthesis and algorithmic composition. It provides an interpreted object-oriented language which functions as a network client to a state of the art, realtime sound synthesis server." (vgl. http://doc.sccode.org/)

Die SuperCollider Umgebung wird in zwei Komponenten aufgeteilt: dem *scsynth* Server und dem *sclang* Client. Der *scsynth* Server dient als Sound Generator und bietet Dienste an um eine Echtzeit Klangsynthese zu ermöglichen. Der *sclang* Client interpretiert den Programmcode und ermöglicht das algorithmische Komponieren. *Sclang* beschreibt sowohl die SuperCollider Programmiersprache, den Interpreter dieser Sprache als auch die Client Anwendung, die typischerweise mit dem *scsynth* Server kommuniziert. Der *sclang* Client kann jedoch auch innerhalb des Netzwerks mit anderen Servern kommunizieren. Die Kommunikation zwischen *sclang* Client und *scsynth* Server findet auf dem OSC Protokoll innerhalb des Netzwerks statt. Die folgende Abbildung zeigt den oben beschriebenen Aufbau von SuperCollider:

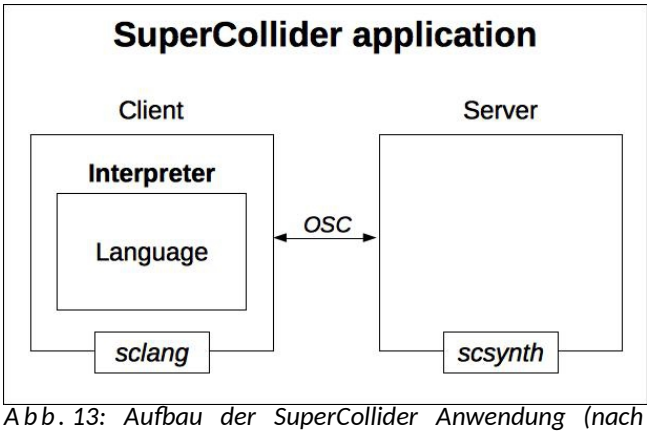

Abb. 13: Aufbau der SuperCollider Anwendung (nach http://doc.sccode.org/Guides/ClientVsServer.html)

Zwar baut die Installation nicht auf synthetisch erzeugten, künstlichen Klängen auf, jedoch ermöglicht *sclang* die Ansteuerung einer realen Klangerzeugung durch algorithmische Komposition. *Sclang* stellt dazu das geforderte Regelwerk (vgl. 3.2) bereit und ermöglicht die Umsetzung der Komposition des *sound controllers*. Zudem bietet *sclang* eine Kommunikationsschnittstelle, die auf dem OSC Protokoll basiert und sogleich eine Kommunikation mit dem WSAN erleichtert. Zur Umsetzung des *sound controllers* (sowie seiner Komponenten Komposition und Klangauslösung) wird daher lediglich *sclang* genutzt.
Dazu wird der *sclang* Client so eingerichtet (wie in 5.3.2.2.1 Initialisierung beschrieben), dass der *sound controller* Nachrichten des *network controllers* empfängt (z.B. Sensordaten für die Komposition), aber auch Nachrichten an ihn senden kann (bspw. sendet die Klangauslösung eine

50 http://www.sojamo.de/libraries/oscP5/

Nachricht zum Anschlagen eines Klangstabs). Folgendes Codebeispiel zeigt, wie das Senden und Empfangen von OSC Nachrichten in SuperCollider implementiert werden kann:

```
// Erzeugt eine Netzwerk Adresse, die sclang selbst repraesentiert
n = NetAddr("localhost", NetAddr.langPort);

// Sendet eine OSC Nachricht, die bspw. alle (drei) Sensordaten des
zweiten Knotens enthaelt
n.sendMsg("/node/2/sensor/", 0.123, 0.456, 0.789);

// Empaengt eine Nachricht, dessen OSC Address Pattern auf
"/node/2/sensor" zutrifft
OSCFunc.new(
  {|msg| msg.postln},
  '/node/2/sensor/',
  n
);
```

Zunächst wird ein Objekt einer Netzwerkadresse (`NetAddr`) erzeugt. `NetAddr.langPort` gibt dabei die Portnummer des *sclang* Clients an. Das Senden erfolgt mit Hilfe der Instanzmethode `sendMsg` des `NetAddr`-Objekts, welche als Argumente zunächst das OSC Address Pattern und anschließend alle Daten erhält, welche die Nachricht enthalten soll (in diesem Beispiel alle Sensordaten des Knotens). Gesendet wird die Nachricht an die vom `NetAddr`-Objekt angegebene Adresse. In diesem Fall senden wir uns zwar selbst die Nachricht zu, der *sound controller* wird jedoch so konfiguriert, dass er auf zwei verschiedenen Ports sendet und empfängt. Zum Empfangen einer OSC Nachricht, ist es möglich eine Funktion mittels `OSCFunc` zu registrieren, die ausgeführt wird, wenn eine Nachricht empfangen wurde, dessen OSC Address Pattern auf die übergebene Adresse zutrifft.
Ein weiterer Vorteil, den SuperCollider bietet, ist die dynamische Programmiersprache von *sclang*. Diese erlaubt ein „just in time programming“, sodass Teile des Codes einzeln, nach Belieben zur Laufzeit ausgeführt werden können. Im obigen Codebeispiel werden deshalb die Programmzeilen nicht (notwendigerweise) alle nacheinander ausgeführt, sondern könnten in einer geeignete Reihenfolge manuell ausgeführt werden (z.B. *`OSCFunc.new`* vor `sendMsg` aufrufen). Des Weiteren ermöglicht dies eine Ansteuerung der Knoten zur Laufzeit und ermöglicht daher ein „live“ Komponieren (vgl. 3.2.1).

5.5 Zusammenfassung

In diesem Kapitel wurde eine Software-/Hardware Lösung zur Umsetzung der Klanginstallation entwickelt. Dazu wurde zunächst in 5.1 eine Architektur, ihre Komponenten sowie die Kommunikationswege zwischen den Komponenten vorgestellt. In Kapitel 5.2 wurde das Nachrichtenprotokoll OSC vorgestellt, welches die Komponenten zur Kommunikation (auf Applikationsebene) nutzen. Anschließend wurden in Kapitel 5.3 die Komponenten genauer spezifiziert. Zuletzt beschrieb Kapitel 5.4 den aktuellen Stand der Hardware-Software-Lösung.

6 Fazit

Aus der Vorstellung einer verteilten Klanginstallation wurde ein Hardware-Software-Konzept entwickelt, welches die Installation sowohl umsetzen als auch nach Bedarf umgestalten lässt.
Dazu wurde zunächst in Kapitel 2 die Installation beschrieben und ihre drei Einflussgrößen Akustik, Optik und Interaktion näher erläutert. Die daraus resultierenden technischen Anforderungen wurden in Kapitel 3 aufgeführt. Dieses umfasste Anforderungen an das Netzwerk und die Klangsteuerung und stellte nichtfunktionale Anforderungen auf. Kapitel 4 stellte das Projekt Sense/Stage vor, welches eine vollständige Hardware- und Software-Infrastruktur zur Errichtung eines kabellosen Sensor/Aktor Netzwerks bereitstellt. Da die erforderliche Hardware zur Umsetzung der Klanginstallation zu kostenintensiv war, wurde in Kapitel 5 ein kostengünstigeres Hard-Software-Konzept entwickelt.
Dieses Konzept umfasst das WSAN, den *network configurator* und den *sound controller*.
Das WSAN setzt sich zusammen aus Netzwerkknoten, welche mit Sensoren und Aktoren ausgestattet werden können, und einem *network controller*, welcher als Koordinator für den Aufbau sowie die Instandhaltung des Netzwerks zuständig ist und als Gateway Netzwerkknoten auch außerhalb des WSANs erreichbar macht. Es baut auf Basis von Arduino Mikrocontrollerboards und nRF24l01+ Funkmodulen auf und erlaubt eine verteilte Anordnung von Sensoren und Aktoren.
Netzwerkknoten und *network controller* sind zur Zeit noch nicht vollständig implementiert. Zwar können sich Knoten beim *network controller* anmelden und sind auch von außerhalb des WSANs ansteuerbar, jedoch sind die Dienste von Sensor- und Aktorknoten (z.B. das Konfigurieren der Hubmagneten) noch nicht vollständig ausgebaut. Auch ist ein energiesparender Sleep-Mode noch nicht eingerichtet. Der *network controller* ist fast vollständig implementiert, kann zur Zeit jedoch auf Grund von Speicherproblemen nur eingeschränkt laufen.
Der *network configurator* ist eine auf Processing basierende, graphische Benutzerschnittstelle zur Übersicht und Konfiguration des Netzwerks. Er umfasst die vier Sichten: Netzwerksicht, Knotensicht, Sensorsicht und Aktorsicht. Die Netzwerksicht bietet eine Übersichtskarte, welche einen schnellen Überblick über die aktuellen Zustände von Knoten gibt und das Selektieren einzelner Knoten ermöglicht. Knoten-, Sensor- und Aktorsicht dienen zum Konfigurieren des Netzwerks.
Die aktuelle Implementierung beinhaltet lediglich die Netzwerksicht. Aktive Knoten erscheinen so bspw. in dieser Sicht und können selektiert und verschoben werden. Ausgefallene Knoten werden ihrem Zustand entsprechend markiert. Die Implementierung von Knotens-, Sensor- und Aktorsicht sowie eine Erweiterung der Netzwerksicht steht noch aus.
Der *sound controller* ist die Klangsteuerung. Er basiert auf SuperCollider und ist für die Umsetzung der Interaktion und Ansteuerung der Klangstäbe zuständig.
Eine Implementierung des *sound controllers* war bislang noch nicht möglich. Jedoch wurden verschiedene Ansätze der algorithmischen Komposition mit SuperCollider erforscht und vereinzelt an Netzwerkknoten getestet.
Da alle Komponenten (Netzwerkknoten, *network controller*, *network configurator* und *sound controller)* zum jetzigen Stand nicht vollständig implementiert sind, konnte ein vollständiger Systemtest noch nicht durchgeführt werden. Es fand jedoch bei jeder Änderung ein Modultest sowie ein Integrationstest der betroffenen Komponenten statt.
Das verwendete Nachrichtenprotokoll zur Kommunikation zwischen den Komponenten findet (auf Anwendungsebene) auf dem OSC Protokoll statt. OSC ermöglicht eine flexible, individuelle Adressierung der Knotendienste und unterstützt somit die Erweiterbarkeit eines Knotens. Zudem

sind bereits sämtliche Implementierungen des OSC Protokolls verfügbar, sodass eine Ansteuerung des WSANs auch von anderen Anwendungen aus möglich ist.
Die aktuellen Quellcodes befinden sich auf der beiliegenden DVD. Zusätzlich sind dort Klangaufnahmen der Klangstäbe enthalten.
Neben dem Entwickeln des Hardware-Software-Konzepts fand im Zuge dieser Arbeit auch ein praktischer Entwicklungsprozess der Installation statt. So wurden bspw. Klangeigenschaften verschiedener Materialien (z.B. Holz, Metall, Kunststoff) untersucht, erarbeitet, wie diese - elektronisch angesteuert - zum Klingen gebracht werden können, und das Verhalten unterschiedlicher Sensor- und Aktorsysteme erforscht. Die in dieser Arbeit beschriebene Hardware Lösung stellt den aktuellen Stand der verwendeten Bauteile dar (vgl. 5.4.1).

6.1 Ausblick

Diese Arbeit bietet einige Weiterentwicklungsmöglichkeiten, welche im Folgenden für die Software- sowie die Hardwareseite betrachtet werden.

6.1.1 Software

Da bis zum jetzigen Stand die Architektur Komponenten noch nicht vollständig implementiert sind, werden die noch ausstehenden Implementierungen zunächst vervollständigt. Überlegungen zu möglichen Erweiterungen sind jedoch bereits vorhanden:

So könnte bspw. der *sound controller* benutzerkomfortabler gestaltet werden, da bislang die Zuweisung bestimmter Sensoren oder Klangstäbe manuell durch Vorgabe der vorgesehenen Knotennummer sowie Sensor- bzw. Klangstabnummer erfolgt. Das bedeutet, dass der Benutzer stets wissen muss, welchem Knoten welche Knotennummer zugewiesen ist. Dies stellt möglicherweise bei einer kleineren Anzahl an Knoten kein Problem dar, kann jedoch bei einer größeren Anzahl schon unübersichtlich werden. Es wäre deshalb eine solche Lösung erstrebenswert, bei welcher in SuperCollider eine (eventuell auch graphische) Umgebung geschaffen wird, die Knoten automatisch erkennt und abhängig vom Knotentyp (Sensor, Aktor, Hybrid) als Objekte eigenen Datentyps darstellt, welche bspw. Sensordaten als Datenströme bereitstellen.

Auch wäre eine Erweiterung des *network controllers* denkbar, welche sich mit der Problematik von Interferenzen beim kabellosen Übertragen von Daten auseinander setzt: Da der nRF24l01+ wie viele andere kabellose Technologien (z.B. WLAN, Bluetooth und einige ZigBee Module) im 2,4GHz ISM Band[51] operiert, kann es möglicherweise zu Störungen bei der kabellosen Kommunikation kommen - vor Allem dann, wenn unterschiedliche Geräte in der selben Umgebung auf ähnlichen Frequenzen arbeiten. Der n*etwork controller* könnte deshalb dahingehend erweitert werden, dass er solche Interferenzen zur Laufzeit erkennt und geeignete Maßnahmen trifft. So könnte er z.B. eine „freie“ Frequenz suchen und den Wechsel aller Knoten auf diese Frequenz koordinieren.

51 Das 2,4 GHz ISM Band (Industrial, Scientific and Medical Band) beschreibt den Frequenzbereich von 2.400 MHz bis 2.500 MHz (vgl. http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Telekommunikation/Unternehmen_Institutionen/Frequenzen/Allgemeinzuteilungen/2003_76_ISM_pdf.pdf)

6.1.2 Hardware

Inspiriert von den MiniBees könnte für Netzwerkknoten ein kleines Hardware-Board entwickelt werden, welches mit integriertem Mikrocontroller sowie integriertem nRF24l01+ Funkmodul ausgestattet ist. Dieses könnte z.B. wie die in 5.4.1.1 vorgestellten Netzwerkknoten auf dem Design eines Arduino Pro Mini basieren, Schnittstellen für analoge Ein- sowie digitale Ein- und Ausgänge bereitstellen und durch ein über USB aufladbaren Akku betrieben werden. Ein solches Board bietet den Vorteil, dass es auf Grund seiner Größe vielfältig einsetzbar wäre und für die Verwendung lediglich mit Sensoren oder Aktoren ausgestattet werden müsste.

Auch wäre eine Weiterentwicklung der Installation interessant, bei welcher andere Mechanismen der Klangerzeugung eingesetzt werden. So könnten z.B. elektromagnetischen Aktoren (meist eine Spule mit ferromagnetischem Kern) verwendet werden um magnetische Materialien zum Schwingen zu bringen (vgl. McPherson 2012, S. 1). Auf diese Weise kann etwa der Klang eines Pianos ohne perkussiven Anschlag erzeugt werden, wie das Beispiel des Magnetic Resonator Pianos zeigt (vgl. McPherson 2010).

Anhang A

DVD

Die folgende Abbildung zeigt die Verzeichnisstrukur der DVD:

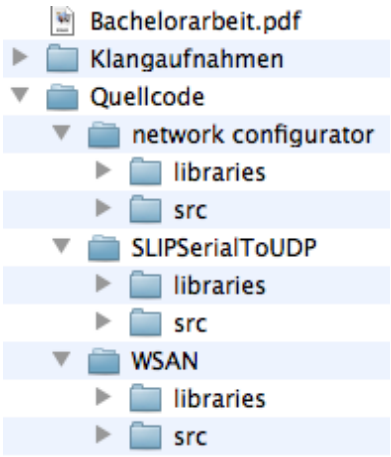

Abb. 14: Verzeichnisstruktur der DVD

Die digitale Version dieser Arbeit ist im Hauptverzeichnis unter dem Dateinamen „Bachelorarbeit.pdf“ zu finden.

Der Ordner „Klangaufnahmen“ enthält einige Tonaufnahmen von Klangstäben.

Der aktuelle Quellcode befindet sich im gleichnamigen Ordner. Dieser enthält drei Unterordner, die jeweils „src“ sowie „libraries“ beinhalten. In „src“ ist der jeweilige Programmcode hinterlegt. „libraries“ enthält zusätzlich benötigte Bibliotheken.

Der in „network configurator“ enthaltene Quellcode ist in Java geschrieben. Die verwendete Processing core Bibliothek ist beigefügt.

„SLIPSerialToUDP“ enthält eine Processing Anwendung, welche UDP Pakete in serielle Nachrichten umwandelt und umgekehrt. Sie wird benötigt um den *network controller* (Arduino) über die serielle Schnittstelle mit dem lokalen Netzwerk zu verbinden. Dieses Beispiel wurde der Arduino OSC Bibliothek entnommen.[52]

Der Quellcode des WSANs kann als Bibliothek der Arduino IDE genutzt werden. Diese benötigt zusätzliche Bibliotheken (wie z.B. den Treiber für das Funkmodul), welche in „libraries“ enthalten sind. Die Bibliothek enthält zudem jeweils ein Codebeispiel für den *network controller*, einen Sensorknoten sowie für einen Aktorknoten.

Der von mir verfasste Quellcode befindet sich in den Verzeichnissen „/Quellcode/network configurator/src“ und „/Quellcode/WSAN/src“.

52 Siehe https://github.com/CNMAT/OSC

Literaturverzeichnis

Angliss 2003
ANGLISS, Sarah: *INFRASONIC – SUMMARY OF RESULTS*. London, 2003. - Online verfügbar unter: http://www.sarahangliss.com/extras/Infrasonic/infrasonicResults.htm Abruf: 2014-09-19

Baalman 2010
BAALMAN, Marije A.J. ; SMOAK, Harry C. ; DE BELLEVAL, Vincent ; BERGMANN, Brett ; SALTER, Christopher L. ; MALLOCH, Joseph ; THIBODEAU, Joseph ; WANDERLEY, Marcelo M.: *Sense/Stage : low cost, open source wireless sensor and data sharing infrastructure for live performance and interactive realtime environments*. In: *Linux Audio Conference*. 2010. - Online verfügbar unter: http://sensestage.hexagram.ca/wordpress/wp-content/files_flutter/1281449432_28_1_1_13_file.pdf Abruf: 2014-09-19

Denegri 2007
DENEGRI, Livio ; ZAPPATORE, Sandro ; DAVOLI, Franco: Sensor network based localization for continuous tracking applications : [invited paper]. In: INSTITUTE FOR COMPUTER SCIENCES, SOCIAL-INFORMATICS AND TELECOMMUNICATIONS ENGINEERING (ICST) (Hrsg.): *Proceedings of the First International Conference on Immersive Telecommunications (ImmersCom '07)*. Brüssel, 2007. - ISBN 978-963-9799-06-6. - Article No. 16. - Online verfügbar unter: http://dl.acm.org/citation.cfm?id=1538981.1539002 Abruf: 2014-09-19

Galanter 2003
GALANTER, Phillip: *What is Generative Art? : Complexity Theory as a Context for Art Theory*. In: *GA 2003 6th Generative Art Conference*. 2003. - Online verfügbar unter: http://philipgalanter.com/downloads/ga2003_what_is_genart.pdf Abruf: 2014-09-19

Mandelbrot 1991
MANDELBROT, Benoît B.: *Die fraktale Geometrie der Natur*. Birkhäuser, 1991. - ISBN 3-7643-2646-8

Mann 2013
MANN, Yotam ; FREED, Adrian: *CNMAT/OSC · GitHub*. - Online verfügbar unter: https://github.com/CNMAT/OSC Abruf: 2014-09-19

Matischek 2012
MATISCHEK, Rainer: *Real-Time Communication MAC Protocols for Wireless Sensor Networks : For Automotive and Industrial Applications*. Hamburg : Verlag Dr. Kovač GmbH, 2012 (*Schriftenreihe : Technische Forschungsergebnisse. Band 12)*. - ISBN 978-3-8300-6349-0. - Zugl.: Überarbeitete Version der Dissertation, Technische Universität Wien, 2011

McPherson 2010
MCPHERSON, Andrew: The Magnetic Resonator Piano : Electronic Augmentation of an Acoustic Grand Piano. In: *Journal of New Music Research* 39 (2010), Nr. 3, S. 189-202. - Online verfügbar unter: *http://www.tandfonline.com/doi/full/10.1080/09298211003695587#.UqiN4I2mgy4* Abruf: 2014-09-19

McPherson 2012
McPherson, Andrew: Techniques and Circuits for Electromagnetic Instrument Actuation. In: University of Michigan (Hrsg.): *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. Ann Arbor, Michigan, 2012. - Online verfügbar unter: *http://www.eecs.umich.edu/nime2012/Proceedings/papers/117_Final_Manuscript.pdf* Abruf: 2014-09-19

Novello 2013
Novello, Alberto: *Rosa Dei Venti | TRASFORMATORIO*. Italien, 2013. - Online verfügbar unter: http://www.trasformatorio.net/?p=831 Abruf: 2014-09-19

Saha 2003
Saha, Ddebashis; Mukherjee, Amitava: Pervasive Computing : A Paradigm for the 21st Century. In: *Computer* 36 (2003), Nr. 3, S. 25-31. - Online verfügbar unter: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1185214 Abruf: 2014-09-19

Verdone 2008
Verdone, Roberto ; Dardari, Davide ; Mazzini, Gianluca ; Conti, Andrea: *Wireless Sensor and Actuator Networks : Technologies, Analysis and Design*. Academic Press, 2008. - ISBN 978-0-12-372539-4

Wright 2003
Wright, Matthew ; Freed, Adrian ; Momeni, Ali: *Open Sound Control : State of the Art 2003*. In: *Proceedings of the 2003 conference on New interfaces for musical expression*, S. 153-159. Montreal, Canada, 2003. - Online verfügbar unter: http://opensoundcontrol.org/files/Open+Sound+Control-state+of+the+art.pdf Abruf: 2014-09-19

Wright 2010
Wright, Matthew: *OpenSoundControl Application Areas*. - Online verfügbar unter: http://archive.cnmat.berkeley.edu/OpenSoundControl/application-areas.html Abruf: 2014-09-19

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____