DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Pepper VR – Teleopperation eines humanuiden Roboter auf Basis der Analyse menschlicher Bewegung

Studienarbeit

für die Prüfung zum

Bachelor of Science

des Studienganges Informatik / Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

von

Matthias Schuhmacher und Marlene Rieder

Abgabedatum 20. Mai 2024

Bearbeitungszeitraum	300 Stunden
Matrikelnummer	4128647 und 8261867
Kurs	tinf21b3 und tinf21b5
Gutachter der Studienakademie	Prof. Dr. Marcus Strand

Erklärung

Wir versicheren hiermit, dass wir unsere Studienarbeitmit dem Thema: »Pepper VR – Teleopperation eines humanuiden Roboter auf Basis der Analyse menschlicher Bewegung« selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt haben. Wir versicheren zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort Datum

Unterschrift

Zusammenfassung

Das Ziel der vorliegenden Studienarbeit ist es, eine Verbindung zwischen einer Virtual-Reality Brille und dem humanoiden Roboter Pepper herzustellen. Das Kamerabild des Roboters soll auf der Brille angezeigt werden, ebenfalls soll es möglich sein, den Roboter mit Hilfe der Controller der Brille zu steuern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Liste der Algorithmen

Formelverzeichnis

Abkürzungsverzeichnis

Kapitel 1

Einleitung

Der technologische Fortschritt steht nie still, deshalb ist es auch uns ein wichtiges Anliegen daran teilzuhaben. Mit dieser Arbeit

Kapitel 2

Grundlagen

2.1 Pepper

Pepper ist ein humanoider Roboter, der entwickelt wurde, um die Gefühle und Gesten von Menschen zu analysieren und basierend auf diesen, darauf zu reagieren. Das Projekt entstand durch eine Zusammenarbeit des französischen Unternehmens Aldebaran Robotics Société par actions simplifiée (SAS) und des japanischen Telekommunikations- und Medienkonzerns SoftBank Mobile Corp. Ziel diese Projektes war es, einen humanuiden "Roboter-Gefährten" oder einen "persönlichen Roboter-Freund" zu schaffen, der zunächst im Gewerbesektor in Verkaufsräumen, an Empfangstischen oder in Bildungs- und Gesundheitseinrichtungen eingesetzt werden sollte. Die Produktion wurde jedoch aufgrund geringer Nachfrage bis auf Weiteres pausiert.
Das Konzept von Pepper distanziert sich von herkömmlichen Industrierobotern und reinen Spielzeugrobotern, indem er als informativer und kommunikativer Begleiter konzipiert wurde. Sein Aussehen, das im etwa an die Größe eines Kindes angelehnt ist, sowie ein freundliches Gesicht und eine kindliche Stimme sind im ästhetischen Konzept von "kawaii" (japanisch für "niedlich" oder auch "liebenswert") gehalten.
Pepper wurde im Rahmen einer Präsentation am 5. Juni 2014 als der "erste persönliche Roboter der Welt mit Emotionen" vorgestellt. Die Vermarktung begann damit, dass SoftBank Pepper-Geräte in ihren Verkaufsräumen einsetzte, um Kunden zu unterhalten und zu informieren. Die Roboter sollte dabei den Umgang mit Kunden erlernen, um zukünftige Anwendungsmöglichkeiten zu erforschen. Verkauft wurde offiziell ab dem 3. Juli 2015 zu einem Preis von 198.000 Yen pro Einheit, zuzüglich monatlicher Gebühren für Zusatzleistungen. Im Laufe der Zeit wurde Pepper auch für den Einsatz in weiteren Unternehmen und Einrichtungen verfügbar gemacht.
Pepper wird mit einer Grundausstattung an Anwendungen geliefert, jedoch sind für spezifische Anwendungen, individuell entwickelte Softwarelösungen erforderlich wie auch zum Beispiel in diesem. SoftBank ermöglichte unabhängigen Entwicklern durch die Veröffentlichung der Schnittstellen den Zugang zu einem Interface für Applikationsprogramme, um zusätzliche Anwendungen für Pepper zu erstellen. Das NAOqi-Framework welches für diesen Nutzen bereitgestellt wurde, beinhaltet eine Application Programming In-

terface (API), eine Software Development Kit (SDK) und weitere Tools, welche in den Sprachen Python und C++ uneingeschränkten Zugriff auf die Komponenten, Sensoren und Aktoren des Roboters bieten, dazu später Ausführlicheres in Abschnitt 2.3. Mit Hilfe diese Interfaces haben verschiedene Unternehmen integrierte Lösungen entwickelt, die Pepper beispielsweise bei der Kundenberatung unterstützen können.
Das Design von Pepper ist dem Menschen ähnlich und umfasst einen Kopf mit integrierten Mikrofonen und Kameras sowie einen Torso mit weiteren Sensoren für Stabilität und Sicherheit. Der Roboter verfügt über verschiedene Mechaniken, die es ihm ermöglichen, sich flüssig zu bewegen und mit Personen zu interagieren. Durch die Verwendung von Kameras und bereitgestellter Software ist Pepper in der Lage, Emotionen bei seinen Gesprächspartnern zu erkennen und darauf zu reagieren, obwohl er selbst keine Mimik besitzt. Sicherheitsvorkehrungen wie Abstandssensoren und Stabilisatoren gewährleisten einen sicheren Einsatz von Pepper in verschiedenen Umgebungen. Diese können jedoch bedingt durch den Entwickler deaktiviert werden, um den Roboter in komplexeren oder laborähnlichen Umgebungen zu betreiben.

2.2 Publish and Subscribe

Bevor wir uns mit den Technologien und Herangehensweisen des Pepper Roboters beziehungsweise dessen Betriebssystem beschäftigen, müssen wir uns mit dem Publish-Subscribe-Modell auseinandersetzen.
Das Publish-Subscribe-Modell ist ein Paradigma für einen effektiven Nachrichtenaustausch in verteilten Systemen. Erstmals publiziert in einem Paper der Association for Computing Machinery (ACM) von 1987[**wiki_publish_subscribe_pattern**], ermöglicht es die flexible Kommunikation zwischen verschiedenen Komponenten, indem es einen Mechanismus bereitstellt, über den Nachrichten von einem Sender, dem Publisher, an einen oder mehrere Empfänger, den Subscribern, verteilt werden können. Ein zentrales Element dieses Modells ist dabei Nachrichtenbroker oftmals auch nur als Borker referenziert, der als Vermittler zwischen Publishern und Subscribenden fungiert.
Der Broker empfängt Nachrichten vom Publisher und organisiert sie in verschiedene Kategorien, welche als Topics bezeichnet werden. Diese Topics dienen als thematische Selektionsmerkmale, die den Inhalt der Nachrichten beschreiben. Sie können in den verschiedensten Datentypen erfolgen, einfache Ganzzahlen, Texte bis hin zu Bildern oder weitaus komplexeren Datenstrukturen. Durch diese Kategorisierung in den Topics, wird eine gezielte Auswahl und Weiterleitung der Nachrichten ermöglicht.
Subscriber können bestimmte Topics abonnieren, die ihren Gewünschten entsprechen. Dadurch erhalten sie nur die Nachrichten, die zu den von ihnen gewählten Themen gehören. Dieser Prozess der Nachrichtenauswahl und -verarbeitung wird als Filterung bezeichnet und kann auf zwei Arten erfolgen: themenbasiert und inhaltsbasiert.
Im themenbasierten Ansatz erhalten Abonnenten alle Nachrichten zu den Topics, welche sie abonniert haben. Der Publisher definiert die verfügbaren Topics, aus welchen die Abonnenten selbst wählen können. Im inhaltsbasierten Ansatz dagegen, werden Nach-

richten nur an die Abonnenten weitergeleitet, wenn sie den vom dessen festgelegten Kriterien entsprechen. Dabei ist der Abonnent für die Spezifikation dieser Kriterien verantwortlich[**itwissen_publish_subscribe_model**].
Das Publish-Subscribe-Modell bietetdurch seine Herangehensweise eine hohe Flexibilität und Skalierbarkeit, und wird daher in verschiedenen Anwendungsbereichen eingesetzt. Es ermöglicht eine Entkopplung von Nachrichtenerzeugung und -verarbeitung, was gerade in der Entwicklung verteilter Systeme Prozesse erleichtert. Die vermutlich bekannteste Implementierung des Modell ist das MQTT-Protokoll, welches konzipiert wurde um Telemetriedeaten zwischen Sensoren und Servern zu übertragen speziell in unzuverlässigen Umgebungen[**elektronik_kompendium_publish_subscribe**].
Weiter findet das Publish-Subscribe-Modell Anwendung in einer Vielzahl von weiteren Systemen und Technologien. Beispiele dafür sind der Java Message Service (JMS), der Data Distribution Service (DDS) oder wie im Fall diese Projektes die Message Oriented Middleware (MOM) zu denen auch Robot Operating System (ROS) gehört.

2.3 NAOqi

Im folgenden Abschnitt wird das NAOqi-Framework, welches auf dem Pepper Roboter läuft, genauer erläutert.

2.3.1 Was ist NAOqi?

NAOqi ist die Bezeichnung für die Hauptsoftware, die auf dem Pepper Roboter ausgeführt wird und ihn intern steuert. Diese kann mit persönlichen Modulen weiterentwickelt und angepasst werden. Dazu können mit Hilfe der Aldebaran SDK auch NQOqi-SDK genannt, eigene Module oder Bibliotheken entwickelt werden. Diese NAOqi-SDK ist die Basis des NAOqi Frameworks und ist in C++ implementiert[**nao_dev_install_guide**].
Das NAOqi Framework ist das Programmiergerüst, welches zur Programmierung von NAO und Pepper Robotern verwendet wird. Es implementiert alle allgemeinen Anforderungen der Robotik, einschließlich: Parallelität, Ressourcen-Management, Synchronisation und Ereignisse. Dieses Framework ermöglicht eine homogene Kommunikation zwischen verschiedenen Modulen wie etwa die Bewegung, Audio oder Video sowie eine homogene Programmierung und einen homogenen Informationsaustausch. Das Framework ist:

- plattformübergreifend, wie bereits erwähnt, basiert die NAOqi-SDK auf C++ und bietet damit die Möglichkeit auf Windows, Linux oder sogar auch Mac zu entwickeln.
- sprachübergreifend, mit einer identischen API für C++ und Python. Weitere Details dazu sind in Abschnitt 2.3.1 aufgeführt.

- fähig auf Introspektion, was bedeutet, dass das Framework weiß, welche Funktionen in den verschiedenen Modulen verfügbar sind und wo. Für Details diesbezüglich siehe Abschnitt 2.3.1.

Sprachübergreifend

Software kann in C++ und Python entwickelt werden. Eine Übersicht über die Sprachen selbst in den Abschnitten Unterabschnitt 2.5.1 und Unterabschnitt 2.5.2. In allen Fällen sind die Programmiermethoden genau die gleichen, alle vorhandenen APIs können unabhängig von den unterstützten Sprachen aufgerufen werden:

- Wird ein neues C++-Modul erstellt, können die C++-API-Funktionen von überall aus aufgerufen werden,
- Sind sie richtig definiert, können auch die API-Funktionen eines Python-Moduls von überall aus aufgerufen werden.

Normalerweise werden die Verhaltensweisen in Python und Ihre Dienste in C++ entwickelt.

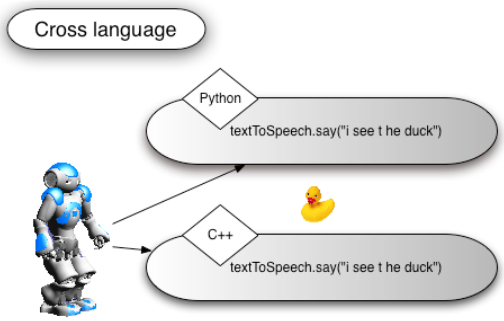

Abbildung 2.1: NAOqi Framework Sprachübergreifend

Introspektion

Die Introspektion ist die Grundlage der Roboter-API, der Fähigkeiten, der Überwachung und der Maßnahmen bei überwachten Funktionen. Der Roboter selbst kennt alle verfügbaren API-Funktionen. Wird eine Bibliothek entladen, werden die entsprechenden API-Funktionen automatisch ebenfalls entfernt. Eine in einem Modul definierte Funktion kann der API mit einem `BIND_METHOD` hinzugefügt werden.
Wird eine Funktion gebunden, werden automatisch folgende Funktionen ausgeführt:

- Funktionsaufruf in C++ und Python, wie in Abschnitt 2.3.1 beschrieben

- Erkennen der Funktion, wenn sie gerade ausgeführt wird
- Funktion lokal oder aus der Ferne, z.B. von einem Computer oder einem anderen Roboter, ausführen weiter im Detail beschrieben in Abschnitt 2.3.1
- Generierung und Aufruf von `wait`, `stop`, `isRunning` in Funktionen

Die API wird im Webbrowser angezeigt wenn auf das Gerät per Uniform Resource Locator (URL) oder Internet Protocol (IP)-Addresse auf dem Port 9559 zugegriffen wird. In dieser Übersicht, zeigt der Roboter seine Modulliste, Methodenliste, Methodenparameter, Beschreibungen und Beispiele an. Der Browser zeigt auch parallele Methoden an, die überwacht, zum Warten veranlasst und gestoppt werden können.
Die Introspektion und derer Implementation im NAOqi-Framework, ist also ein nützliches Werkzeug, welches es ermöglicht, die Roboter-API zu verstehen und zu verwenden aber auch zu überwachenund zu steuern.

Abbildung 2.2: NAOqi Framework Introspektion

Verteilter Baum und Kommunikation

Eine Echtzeitanwendung kann aus einer einzelnen ausführbaren Datei oder einem Baum von mehreren Systemen wie etwa Robotern, Prozessen oder Modulen bestehen. Unabhängig davon sind die Aufrufmethoden immer dieselben. Eine ausführbare Datei kann durch eine Verbindung mit einem anderen Roboter mit IP-Adresse und Port verbunden werden, sodass alle API-Methoden von anderen ausführbaren Dateien sind auf die gleiche Weise verfügbar sind, genau wie bei einer lokalen Methode. NAOqi trifft dabei selbst die Wahl zwischen schnellem Direktaufruf Local Procedure Call (LPC) und Fernaufruf Remote Procedure Call (RPC).

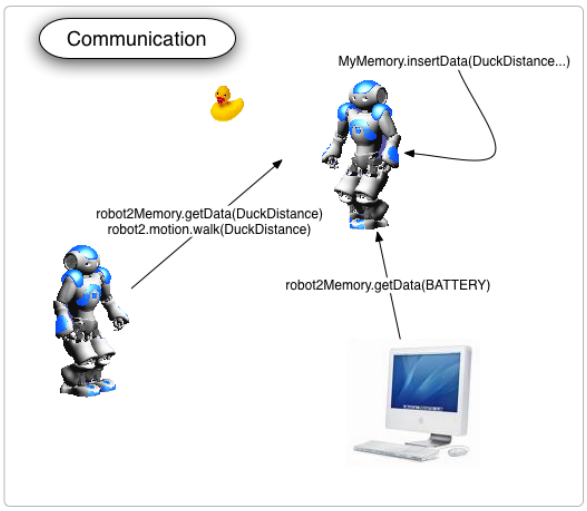

Abbildung 2.3: NAOqi Framework Kommunikation

2.3.2 NAOqi Vorgehensweise

Die NAOqi Software, welche auf dem Roboter läuft, ist ein Broker. Wenn dieser startet, lädt er eine Voreinstellungsdatei in den Speicher, in der festgelegt ist, welche Bibliotheken in dieser Konfiguration geladen werden sollen. Jede Bibliothek enthält ein oder mehrere Module, die den Broker benutzen, um ihre Methoden bereitzustellen.

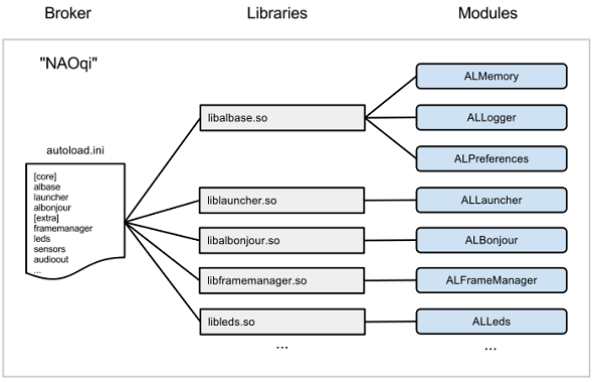

Abbildung 2.4: NAOqi Broker Bibliotheken Module

Der Broker selbst bietet Nachschlagdienste an, so dass jedes Modul im Baum oder im Netzwerk jede Methode finden kann, die an dem Broker bekannt gegeben wurde.
Das Laden von Modulen bildet dann einen Baum von Methoden, die mit Modulen verbunden sind, und von Modulen, welche mit dem Broker verbunden sind.

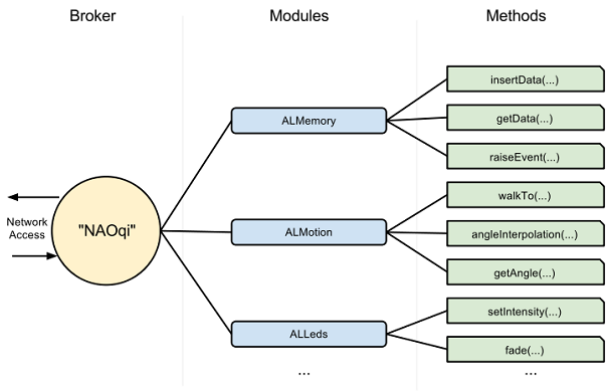

Abbildung 2.5: NAOqi Broker Modul Methoden

NAOqi Proxy

Ein weiterer wichtiger Bestandteil des NAOqi Broker ist der Proxy welcher grundlegend die Aufgabe eines Modules, näher beschrieben im folgenden Unterabschnitt 2.3.3, repräsentiert. Dies kann lokal oder entfernt enstehen der einzige Unterschied dabei ist, dass bei einer entfernten Refernz die IP des entfernten Broker mit angegeben werden muss, gleich bleibt aber im internen, dass der Proxy die Methoden des Moduls an den Broker weiterleitet.

2.3.3 NAOqi Module

Module sind die Grundbausteine der personalisierten Gestaltung von NAOqi. In ihnen kann der Nutzer eigene funktionaitäten implementieren und diese dem Broker bereitstellen. Ein Modul kann dabei aus einer oder mehreren Klassen bestehen, die wiederum Methoden enthalten.
Standardmäßig ist jedes Modul eine Klasse innerhalb einer Bibliothek, welche über eine `autoload.ini`-Datei geladen wird, worauf die Modulklasse automatisch instanziiert wird. Weiter können auch Module von übergeordneten Module abgeleitet werden, ähnlich wie etwa bei der Vererbung in der objektorientierten Programmierung. Wird eine Klasse voneiner anderen abgeleitet, können die Methoden gebunden werden, wodurch ihre Namen und MethodenSignaturen direkt dem Broker bekannt gemacht werden.
Weiter können, wie bereits erwähnt, Module sowohl lokal als auch entfernt implementiert werden. Ist es ein solches entferntes Modul, wird es als ausführbare Datei kompiliert, und kann auch außerhalb des Roboters ausgeführt werden.
Unabhängig jedoch von der Lagerung der Module, enthält jedes Modul eine Bandbreite von Methoden, welche wie im Abschnitt 2.3.1 beschrieben, gebunden werden können und damit nach außen hin nutzbar gemacht werden. Es ist also unabhängig der Lagerung der Module möglich, diese in der gleichen Weise aufzurufen. Module passen sich also automatisch selbst an.

Entfernte Module

Entfernte Module sind Module, die über das Netzwerk kommunizieren, dies kann auf demselben Robotter sein, also auch auf einem anderen Gerät im Netzwerk. Ein entferntes Modul braucht einen entfernten Broker, um mit anderen Modulen zu kommunizieren, da es selbst keinen eigenen besitzt. Broker sind dann für den gesamten Netzwerkteil verantwortlich. Über diesen Umweg über das Netzwerk können keine schnellen Zugriffe über Remote-Module durchgeführt werden wie etwa direkte Speicherzugriffe.
Diese entfernten Module sind trivialer in der Handhabung, da sie außerhalb des Systems entwickelt und debugged werden. Dagegen sind sie aber in der Laufzeit selbst deutlich schwächer in den Punkten Geschwindigkeit und Speichernutzung, gegenüber iherer lokalen Gegenstücke. Diese Funktionalität der entfernten entwicklung spielt speziell in der Implementierung dieses Projektes eine grundlegende Rolle.

Lokale Module

Lokale Module werden als Bibliotheken kompiliert und nur auf dem Roboter verwendet. Dadurch sind sie schneller und effizienter in der Laufzeit als auch in der Speicherverwaltung, dagegen ist die Entwicklung und das Debugging auf dem Roboter selbst deutlich aufwendiger.
Sie bestehen aus zwei oder mehr Module, welche auf dem Roboter im selben Prozess gestartet werden. Sie kommunizieren miteinander über denselben Broker.
Da sich lokale Module im selben Prozess befinden, können diese Variablen gemeinsam nutzen und die Methoden des jeweils anderen ohne Serialisierung oder Vernetzung aufrufen. Dies ermöglicht die schnellstmögliche und maximal effiziente Kommunikation zwischen den Modulen.

2.3.4 NAOqi Speicherverwaltung

Die Speicherverwaltung in NAOqi wird von dem Modul `ALMemory` übernommen. Dieses Modul ist ein Speicher welcher sowohl Daten als auch Ereignisse beinhaltet. Dieser Speicher wird von allen Modulen geteilt und ermöglicht es, Daten zwischen den Modulen auszutauschen. Zusätzlich werden in diesem Speicher auch alle Ereignisse zwischengespeichert, auf welchen dann die abonierten Module zugreifen können, sobald das Ereigniss ausgelöst wurde. Alle Module haben auf diesen Bereich sowohl Lese- als auch Schreibzugriff. Jedoch ist das `ALMemory` Modul kein Echtzeitsynchronisationstool. Abonieren auf Bewegungsdatendaten oder Echtzeitvariablen in diesem Speicher ist also risikobehaftet.
Das Modul selbst ist ein Array aus sogenannten `ALValue`-Objekten, auf welche threadsicher zugegriffen werden kann. In diese Speicherbausteine können alle gängigen Datentypen gelegt werden. Dazu zählen einfache Datentypen wie einzelne Bits, Ganzzahlen, Fließkommazahlen und Zeichenketten, aber auch komplexere Datenstrukturen wie Arrays, Matrizen und Binaries.

Standardmäßig wird zwischen drei verschiedenen Datentypen unterschieden:

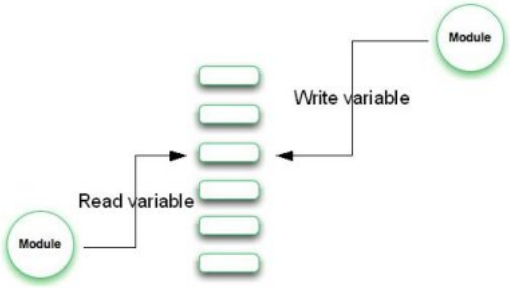

Abbildung 2.6: NAOqi Framework Speicherverwaltung

- Daten von Sensoren und Gelenken: diese Daten werden vom Roboter selbst in den Speicher geschrieben und haben keine Historie. Sie sin nicht größer als 32-Bit und können direkt per Pointer aufgerufen werden.
- Ereignisse: auf welche aboniert werden muss. Sobald eines dieser Ereignisse ausgelöst wird, wird es im Speicher gespeichert und kann von den abonierten Modulen abgerufen werden. Im Gegensatz zu den Daten von Sensoren und Gelenken sowie den Micro-Ereignissen, haben diese Ereignisse eine Historie und können auch nach dem Auslösen noch abgerufen werden.
- Micro-Ereignisse: Diese Ereignisse sind sehr kurzlebig und werden sofort nach dem Auslösen wieder gelöscht. Sie werden nicht im Speicher gespeichert und haben daher keine Historie.

2.4 Robot Operating System

2.5 Programming Languages

2.5.1 C++

2.5.2 Python

2.6 Virtual Reality

2.7 Entwicklung für Virtual Reality

2.8 TCP

Kapitel 3

Technologieauswahl

3.1 VR-Brillen

3.2 Entwicklung für VR-Brillen

Kapitel 4

Umsetzung

4.1 Pepper

4.2 MetaQuest3

4.3 Verbindung

Kapitel 5

Anwendungsgebiete

Das entstandene Produkt kann in verschiedenen Fällen eingesetzt werden.

5.1 Pflege

In Pflegeeinrichtungen können Beispielsweise Bewohner, die selbst nicht mehr so gut zu Fuß sind sich gegenseitig Besuchen und kommunizieren. Ebenfalls können Routinebesuche bei den Besuchern durch Pflegekräfte von einer zentralen Stelle aus getätigt werden, was die Pflegekräfte entlasten würde.

Kapitel 6

Fazit

Kapitel 7

Fortsetzung des Projekts