HAW HAMBURG

Bachelorarbeit

Jakob Kalwar

Konstruktion eines intelligenten Weckmechanismus und dessen Integration in ein Smart Home

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Computer Science and Engineering
Department of Automotive and Aeronautical Engineering

Jakob Kalwar

Konstruktion eines intelligenten Weckmechanismus und dessen Integration in ein Smart Home

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Mechatronik*
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Thomas Lehmann

Eingereicht am: 01. Oktober 2020

Jakob Kalwar

Thema der Arbeit

Konstruktion eines intelligenten Weckmechanismus und dessen Integration in ein Smart Home

Stichworte

Smart, Home, Wecker

Kurzzusammenfassung

Diese Bachelorarbeit beschäftigt sich mit der Konstruktion eines Wecksystems im Smart Home Kontext. Im LivingPlace der HAW wird mit diesem System versucht die Schlafphasen des Menschen zu erkennen. Diese Erkennung soll dem Menschen zu gute kommen, der so immer in einem richtig gewählten Moment geweckt wird. Als weiteres Thema wird die Bedienung eines Smart Homes behandelt. Alle erzeugten Daten werden zudem in einer Datenbank innerhalb des LivingPlaces gespeichert. Die Auswahl von Hardware und das Herstellen eines Prototypen werden ebenso behandelt.

Jakob Kalwar

Title of Thesis

Construction of an intelligent Alarm Clock and it's integration in a SmartHome Environment

Keywords

Smart, Home, Alarm, Clock

Abstract

This thesis is about Construction of a System to wake up people in a smart home. The system will be tested and used in the LivingPlace of HAW Hamburg. Sleep phases of humans will be detected to determind a suitable point in time to wake him up. Furthermore, topics related to the operation of a smart home are dealt with. All data generated

are also stored in a database within the LivingPlaces. Selecting hardware and making a prototype are also covered.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AAL Ambient Assisted Living.

ADC Analog Digital Converter.

BuA Bewegung und Anwesenheit.

CNC Computerized Numerical Control.

CPS Cyber Physical System.

DALI Digital Addressable Lighting Interface.

FDM Fused Deposition Modeling.

HCI Human-Computer Interaction.

I2C Inter-Integrated Circuit.

LAN Local Area Network.

MQTT Message Queuing Telemetry Transport.

PoE Power over Ethernet.

REM Rapid Eye Movement.

SSID Service Set Identifier.

WLAN Wireless Local Area Network.

WZ Weckzeitpunkt.

1 Einleitung

Der Gedanke, Häuser, Wohnungen und weitere Umfelder in unseres täglichen Lebens mehr und mehr von Computern steuern zu lassen, prägt die Forschung seit mehr als 25 Jahren. Stetige Entwicklung im Bereich Computer und Netzwerke der letzten Jahre haben neue und einfachere Möglichkeiten ergeben, die Visionäre damals schon untersucht haben. Der Begriff Ubiquitous Computing beschrieb schon damals das Zusammenarbeiten einer Vielzahl an Computern zur Analyse von Vorgängen z.B. in einem Büro. Mark Weiser [42] hat 1993 schon darüber berichtet, dass das Lokalisieren von Personen in einem Büro mit Hilfe von Computern möglich ist und wofür diese Anwendung von Daten genutzt werden kann. Dieser und weitere Schritte können als Beginn der Smart Home Ära gesehen werden. Schnell wurde erkannt, dass die Interaktion zwischen Computern und dem Menschen in einer solchen Umgebung von gewichtiger Rolle ist, womit ein weiteres Forschungsgebiet entstand. Human Computer Interaction ist ein Themengebiet, welches genau diese Interaktion untersuchen soll. Dabei sollen Fragen geklärt werden in welcher Weise Computer mit Menschen agieren können und andersherum. Für diesen Kontext wurde nicht nur an der HAW ein Labor, wie das LivingPlace [13], errichtet, sondern an Universitäten auf der ganzen Welt wie z.B. in Atlanta, Georgia an dem Georgia Institute of Technology mit dem Namen "The Aware Home" [22]. Das "Aware Home " soll dafür genutzt werden Ubiquitous Computing, HCI und Adaptive Verfahren in einem häuslichen Umfeld zu erforschen. Adaptive Verfahren sollen dafür genutzt werden dem Nutzer eines Smart Homes Entscheidungen abzunehmen. Diese Entscheidungen werden dabei auf Grundlage von Eingangsdaten aus der Wohnumgebung getroffen und erfordern kein äußeres Einwirken von Menschen. In einer Smart Home Umgebung könnten Bewegungsdaten von Personen in einer Wohnung analysiert werden, um z.B. automatisiert Lichter zu schalten.

Der Fortschritt in der Entwicklung von programmierbaren Prozessoren lies diese kleiner in ihrem Formfaktor werden, was dazu führte, dass sie in immer mehr technischen Systemen verbaut wurden. Elektromechanische Maschinen und Komponenten konnten so mit

neuen Methoden und mehr Präzision gesteuert werden. Der Schritt Eingebettete Systeme [21] wie diese zu vernetzen und miteinander kommunizieren zu lassen ist Anlass aktueller Forschungsereignisse. Cyber Physical Systems ist ein Begriff, der eben diese Eingebetteten Systeme mit der Vernetzung von Computern aus Ubiquitous Computing zusammenbringen soll. Diese Systeme können nicht nur in der Industrie Anwendung finden, wo immer mehr Herstellungsprozesse vernetzt miteinander arbeiten, um zeitlich besser abgestimmt zu sein oder Daten während des Herstellungsprozesses aufzuzeichnen. Diese können dann zu Dokumentation des Erzeugnisses benutzt werden. Auch in der Hausautomatisierung spielen diese Systeme eine immer größer werdende Rolle. Besonders dem Forschungsgebiet AAL, welches stark von Datensammlungen abhängt, kann diese Art der vernetzten Komponenten zugutekommen. Die gesammelten Daten sollen dafür genutzt werden, Veränderungen am Gesundheitszustand eines hilfsbedürftigen Patienten zu ermitteln, um bei Auffälligkeiten z.B. einen Pfleger zu alarmieren. Zudem können in einem CPS unterstützende automatische Abläufe programmiert werden, die einer gesundheitlich beeinträchtigten Person gewisse tägliche Aufgaben abnehmen kann.

Ein Smart Home, wie es an der HAW Hamburg im LivingPlace entsteht, ist ein CPS, denn hier agieren viele kleine Subsysteme, die Sensoren und Aktoren bereitstellen, zusammen. Untereinander vernetzt können alle Geräte dazu beitragen, dass Wohnen in einem solchen Umfeld angenehmer zu gestalten.

Diese Arbeit basiert auf dieser Forschungsrichtung und soll einen Beitrag dazu leisten.

2 Analyse

2.1 Ziel und Aufgaben der Arbeit

Die rasante Entwicklung der Welt sowie wirtschaftliche Interessen bewegen die Menschheit seit Jahren dazu, Prozesse und Tätigkeiten zu automatisieren. Erst waren es die großen Fabriken im industriellen Zeitalter, die durch Maschinen Arbeit erleichterten und somit einen wirtschaftlichen Aufschwung erzielten. Seither gewinnen kleine elektrische Endgeräte sowie Verbraucher-Elektronik in unserem Alltag an Bedeutung. Maschinen, die dem Menschen einfache und lästige Aufgaben abnehmen sind für uns ein Teil des normalen Lebens geworden. Seit einigen Jahren zieht die Automatisierung auch in unsere Wohnungen und Häuser ein. Die Geräte im Haushalt sollen intelligenter und miteinander vernetzt werden. Um das Jahr 2012 befasste sich eine Reihe von Universitäten und Professoren mit diesem Thema. Es wurde der Name Cyber Physical Systems gefunden [15]. Mit diesem Begriff soll die Entwicklung, dass große Anzahlen von Klein- und Großcomputern zusammenarbeiten, beschrieben werden. Systeme dieser Art können deutlich mehr Daten erfassen und verarbeiten als zuvor herkömmliche. Durch die höhere Rechenkapazität und mögliche Umverteilung der Rechenleistung können bessere und intelligentere Systeme erschaffen werden, die Mechanik auf Grundlage von präziseren Daten ansteuert. Dieser Fortschritt kann zum Beispiel dazu führen, dass ein Mensch nach dem Aufstehen nicht mehr das Fenster öffnen muss um frische Luft herein zu lassen.

Von dem betreuenden Professor, Prof. Dr. Kai von Luck, wurden folgende Anforderungen gestellt. Zum einen soll ein Mechanismus entwickelt werden, der eine Person in einem günstigen Moment weckt. Dafür soll ein Gesamtsystem entwickelt werden, welches unter anderem einen konstruierten Wecker enthält, der als Interface zwischen Smart Home und Person dienen kann. Dieses Gesamtsystem muss die Möglichkeit aufweisen, sich berührungslos bedienen zu lassen. Eine weitere Forderung ist das Erstellen einer Morgenroutine.

Bewegungen im Bett sollen erkannt werden, um eine grobe Einschätzung der Schlafphasen machen zu können. Dies dient dazu, den Nutzer möglichst in einer Leichtschlafphase zu wecken, da dies die optimale Phase ist, in der ein Mensch geweckt werden kann.

Weiterhin sollen andere Geräte in einer SmartHome Umgebung zum Weckzeitpunkt bzw. kurz davor oder danach angesteuert werden können, um eine Morgenroutine abzuspielen. Dabei sollen Fenster automatisch öffnen oder den Kaffee schon fertig zu haben, wenn der Wecker klingelt. Dies führt zu einer weiteren Anforderung. Das Gesamtsystem muss mit einem Smart Home agieren können.

Ein weiterer Aspekt ist die Bedienmöglichkeiten eines Smart Homes. Die Anforderung nach einer berührungslosen Bedienmöglichkeit des Gesamtsystems erfordert Recherche auf diesem Gebiet.

2.2 Recherche zum Thema Smart Home Services

Das folgende Kapitel soll einige für diese Arbeit wichtige Grundlagen aufgreifen und erklären. Es werden Begriffe kurz erklärt, die in dieser Arbeit eine wichtige Rolle spielen. Außerdem wird Recherche betrieben, um Erkenntnisse zu erlangen, die bei der Umsetzung der Anforderung von Nöten sind.

2.2.1 Was ist ein Smart Home ?

Nicht eine gute Architektur, die dem Objekt viel Stauraum bietet, oder Solarzellen, die das Haus mit Energie versorgt, lässt eine Wohnung oder Haus Intelligent bzw. Smart werden. Es sind Sensoren und Aktoren, die in allen vorstellbaren Gegenständen in einer Wohnung verbaut sein können. Erst durch das Verbinden dieser Endgeräte über ein Netzwerk kann ein Smart Home entstehen. Ein solches Netzwerk wird auch Heterogenes System bezeichnet in dem viele verschiedene Komponenten Daten verarbeiten und austauschen [43].

Ein zentraler Computer führt alle verfügbaren Daten und Funktionen zusammen. Er macht alle Dienste für weitere Endgeräte erreichbar und ermöglicht so eine automatische Steuerung des Hauses als auch das Schalten von Geräten per Hand durch eine Vielzahl von Endgeräten. Schalter in der Wand können genauso in eine Smart Home Steuerung eingebunden werden wie eine Steuerung des Hauses über ein Smartphone.

Auch Bewegungsmelder oder Anwesenheitssensoren anderer Art können in die Steuerung mit einbezogen werden.

Die von der Politik angestrebte Klimawende ist auch ein wichtiger Faktor, der bei der Entwicklung eines intelligenten Hauses beachtet werden muss. Eine automatische Heizungsregelung zum Beispiel, kann Räumlichkeiten gleichmäßig für ein angenehmes Temperaturempfinden heizen und dabei Heizkosten sparen [3]. Der zentrale Computer sowie die vielen kleinen Komponenten des heterogenen Systems verbrauchen jedoch mehr Energie als eine herkömmliche moderne Haustechnik, da hier kein Datenverkehr stattfinden muss. Dies ist ein wichtiger Faktor, der beim Entwickeln von Smart Home Equipment auch eine große Rolle spielen sollte.

Eine Datenbank ermöglicht das Sammeln von Daten, welche mit passender Software visualisiert werden können. So kann der Nutzer Einblick in seinen Stromverbrauch über den Tag oder aber auch die Temperaturkurven von draußen zu drinnen vergleichen. Diese und weitere Daten können wiederum einer künstlichen Intelligenz dabei helfen die Wohnung automatisch zu steuern.

2.2.2 Was ist Ambient Assisted Living ?

Das stetig steigende Durchschnittsalter der Weltbevölkerung [41] führt zu Veränderungen in der Welt und erfordert neue Denkansätze. Eine Vielzahl älterer Menschen benötigen ab einem gewissen Zeitpunkt Hilfe in ihrem Haushalt sowie bei alltäglichen Tätigkeiten, um die Grundbedürfnisse zu befriedigen. Durch die höhere Lebenserwartung der Menschheit wird es immer schwieriger werden genügend Pflegepersonal zu beschäftigen, um allen die gleichen Möglichkeiten bieten zu können. Zudem fühlen sich Personen in ihrem eigenen Heim am wohlsten und möchten solange wie möglich dort wohnen bleiben [35]. An dieser Stelle möchte *Ambient Assisted Living* ansetzen. Mit intelligenter Technik soll versucht werden die Lebensqualität alter oder behinderter Menschen zu verbessern [26]. Der Begriff Ambient Assisted Living baut auf den Begriff Ambient Intelligence auf. Darunter versteht man eine Umgebung, die viele vernetzte Sensoren hat wie zum Beispiel ein Smart Home. Der Ansatz, der hier verfolgt wird, ist, dass die Technik für den Menschen immer weniger sichtbar wird. Heute nutzen wir häufig mechanische oder digitale Schalter, um die Umgebung für uns anzupassen. Die Heizung wird wärmer gedreht und das Licht wird auch per Hand geschaltet. Solche Vorgänge sollen von Ambient Intelligence automatisiert oder vereinfacht werden. Eine klassische Benutzeroberfläche, wie wir sie vom Desktop des

Computers kennen soll vermieden werden [2], um die Bedienung für geistig weniger fähige Menschen zu vereinfachen. Bei der Forschung auf dem Themenbereich AAL wird also versucht alten- und behindertengerechte Hard- und Software zu entwickeln, die durch Einsatz von einer Vielzahl an vernetzten Computern das alltägliche Leben verbessern. Im Folgenden sollen drei kleine Szenarien vorgestellt werden, die nahe legen wie AAL helfen könnte. Ideen für diese Szenarien wurden der Literatur [10] entnommen.

Szenario 1: Überwachung der Einnahme von Medikamenten Einige Personen müssen täglich Medikamente zu sich nehmen, um ihre Vitalfunktionen zu stärken. Bei Nichteinnahme der Medikamente kann es zu schweren Vorfällen kommen, weswegen hier eine Überwachung besonders bei widerwilligen Personen interessant sein könnte. Ein intelligenter Medikamentenspender könnte dabei helfen die Einnahme gewissermaßen zu überwachen. Durch akustische oder optische Signale könnte ein solches Gerät auf sich aufmerksam machen und die Tablette bis zur Entnahme verfolgen. Ein gewisses Restrisiko bleibt natürlich, da der Proband die Tablette anderweitig entsorgen könnte. Bei Unregelmäßigkeiten kann der mit einem Smart Home verbundene Medikamenten Spender einen Arzt oder die ambulante Pflege benachrichtigen.

Szenario 2: Orientierungshilfe durch Lichtsteuerung Wer nachts durch Harndrang geweckt wird, muss erstmal die Toilette finden. Besonders interessant wird es auch hier wieder bei beeinträchtigten Menschen. Auch für demente Personen ist diese Situation nicht leicht. Ein denkbarer Ansatz dafür ist wieder Sensorik im Haus. Das Aufstehen aus dem Bett könnte der Auslöser für ein Event in der Lichtanlage des Hauses sein. Das Licht der Toilette sowie die Beleuchtung auf dem Weg dahin könnte angesteuert werden. Durch Aufzeichnung dieser Ereignisse über eine längere Zeit kann künstliche Intelligenz diese Entscheidung präzisieren, falls der Computer erkannt hat, dass die Person nur ein Glas Wasser trinken wollte.

Szenario 3: Überwachung des Körpergewichts Ein rapider Abfall des Körpergewichts eines Menschen geht sehr häufig mit einer Verschlechterung der Gesundheit einher. Durch intelligente Möbel, die das Gewicht einer Person ermitteln und an eine Datenbank weiterleiten, kann eine Software Auffälligkeiten im Gewichtsverlauf erkennen und gegebenenfalls einschreiten, indem Angehörige oder Pflegepersonal benachrichtigt werden.

Mit dieser Arbeit soll versucht werden auch auf diesem Gebiet einen kleinen Beitrag zu leisten. Da es um das Erkennen von Bewegungen im Bett geht, kann dieser Ort, das Bett, für weitere Zwecke genutzt werden. Eine Überprüfung auf Anwesenheit einer Person im Bett könnte eine Möglichkeit sein Szenario 2 umzusetzen.

2.2.3 Schlafphasen des Menschen

Im Rahmen dieser Arbeit sollen Schlafphasen erkannt werden, um den Wecker in einem günstigen Moment läuten zu lassen. Hier soll ein gewisses Verständnis für den Schlaf erlangt werden, indem einige wichtige Informationen zusammengetragen werden. Bevor die Schlafphasen des Menschen besprochen werden, wird Schlaf sowie Normwerte des Schlafs Grundsätzlich definiert ([39] Kap.1.2.2).

Schlaf wird als Zustand bezeichnet, in dem der Mensch

- Geschlossene Augen hat
- Ruhige Atembewegungen ausführt
- Verminderte Reaktionsfähigkeit auf Sinneswahrnehmungen besitzt

Für Erwachsene Menschen wurden Normwerte für den Schlaf festgelegt. Eine Abweichung von dieser Norm muss nicht zwangsläufig bedeuten, dass eine Person schlecht schläft ([39] Kap.1.2.4).

Normwerte für Schlafparameter eines Erwachsenen

- Schlafdauer 5-9 h
- Schlaflatenz 1-20 min
- Tiefschlafanteil: ca. 20%
- REM-Schlafanteil: ca. 20%

Die Schlafdauer kann unterteilt werden in Schlafzyklen und Schlafphasen. Ein Schlafzyklus wird in bis zu 5 Schlafphasen eingeteilt. ([17] Kap.2.1)

Einschlafphase In dieser Phase beruhigt sich der Körper. Das Wachempfinden schwindet langsam und es treten eventuelle Einschlafzuckungen auf.

Leichtschlafphase Der Muskeltonus ist hier wie in der Einschlafphasen relativ hoch. Die Wahrnehmung ist jedoch bereits gehemmt. Das Wecken lassen und Aufwachen in dieser Phase fällt leicht.

Mittlerer Schlaf Diese Phase kommt nach und vor dem Leichtschlaf. Es ist die Übergangsphase zum Tiefschlaf. Der Muskeltonus und die Wahrnehmung nehmen weiter ab. Es kann jedoch zu Bewegungen kommen.

REM Schlaf In dieser Phase ist der Muskeltonus auf seinem Minimum, wobei das Gehirn in dieser Phase besondere Aktivität zeigt. Diese Phase wird auch als Traumphase bezeichnet. Der Schläfer verarbeitet in dieser Phase Geschehen aus der Vergangenheit. Trotz der hohen Aktivität des Hirns ist diese Phase genauso wichtig wie die Tiefschlafphase für einen erholsamen Schlaf. Die Abkürzung REM steht dabei für Rapid Eye Movement, denn in dieser Phase bewegen sich die Augen viel. Um Bewegungen aus dem Traum nicht in die Realität umzusetzen und uns so gegebenenfalls zu verletzen, schaltet der Körper die Muskeln ab. In dieser Phase können also keine Bewegungen verzeichnet werden. Das Wecken aus der REM-Phase kann zu unangenehmen Traumberichten führen. ([39] Kap.1.2.3)

Tiefschlafphase Der Muskeltonus hat stark abgenommen, es kommt also vermindert zu Bewegungen im Bett. Das Wecken aus der Tiefschlafphase führt zu einer gewissen Verwirrtheit. Die geweckte Person muss sich erst wieder an der neuen Situation orientieren. ([39] Kap.1.2.3)

Bei einer Schlafdauer von 7-8 Stunden durchläuft der Schläfer 4-5 Schlafzyklen. Ein Schlafzyklus beginnt und endet mit der Leichtschlafphase, nach der die REM-Phase eintritt. Daraufhin gerät der Schlafende in den mittleren Schlaf um als nächstes eine gewisse Zeit im Tiefschlaf zu sein. Nach dem Tiefschlaf folgt wieder mittlerer Schlaf und Leichtschlaf. Dieser Zyklus dauert ca. 80-110 Minuten. Forschungen ([39] Kap.1.2.2) haben ergeben, dass der Mensch nur in den ersten 2-3 Phasen in den Tiefschlaf gelangt. Wer also über diese 3 Schlafphasen hinauskommt, somit länger als 5:30 Std schläft, läuft nur noch Gefahr in der REM-Phase geweckt zu werden. Alle anderen Phasen sind unkritisch.

Diese Erkenntnisse sollen bei der Entwicklung genutzt werden, um einen Algorithmus zu erschaffen, der Menschen entweder in der Leicht- oder Mittleren Schlafphase weckt. Grund dafür ist das bessere Wohlbefinden nach dem Wecken in einer der genannten Phasen.

2.2.4 Entwicklungsstand im Bereich Smart Home Wecker

Auf dem Markt sind verschiedene Produkte verfügbar. Die herausstechenden Produkte basieren auf Sprachassistenten von den Marktführern [29]. Sie verfügen über eine Anzeige für die Uhrzeit, Lautsprecher und sind mit Smart Home Diensten verbunden. Auch ein alltäglicher Gegenstand, den viele von uns bei sich tragen, kann als Smart Home-fähiger Wecker bezeichnet werden, das Smartphone. Weckzeiten sind über Spracheingabe einzustellen und die Weckmelodie kann über einen im Smart Home integrierten Lautsprecher wiedergeben werden.

Wecker zur Erkennung von Schlafphasen basieren häufig auf einer Kombination aus Armband/Sensor und Wecker oder Smartphones. Auch Fitnessarmbänder und Uhren können in einem Smart Home ein Teil eines Wecksystems genutzt werden, da sie teilweise schon als Überwacher im Schlaf Verwendung finden [14]. Diese benutzen Bewegungs- und Herzfrequenzsensoren zur Bestimmung der Schlafphase. Armbänder haben den Vorteil die Herzfrequenz messen zu können, um diese beim Klassifizieren der Schlafphase zu berücksichtigen. Es ist vorstellbar eines dieser bereits vorhandenen Produkte in eine Smart Home Umgebung zu integrieren. So könnte ein Gesamtsystem aus Fitnessarmband, Server und Lautsprechersystem erstellt werden, welches zum Wecken in der richtigen Schlafphase genutzt wird.

2.2.5 Wecker als ein Service

Wie kann der Wecker als Service in einem Smart Home arbeiten? Hier soll besprochen werden, welche Dienste ein Wecker bieten könnte und welche er in Anspruch nehmen kann. Die herkömmlichen Dienste eines Weckers sind das Anzeigen einer Uhrzeit sowie das Abspielen eines Wecktons zu einer eingestellten Uhrzeit. Ein Wecker mit der Fähigkeit in einem Smart Home Netzwerk zu kommunizieren kann weitaus mehr Funktionen bieten als ein herkömmlicher.

Durch eine Publish-Subscribe Architektur wie z.b. MQTT (siehe 2.3.1) können alle Geräte in einem Smart Home Dienste anbieten sowie abonnieren. So können nicht nur die herkömmlichen, sondern auch zusätzliche Dienste angeboten werden. Durch diese Architektur sind Dienste eines herkömmlichen Weckers nicht mehr an ihn gebunden und können stattdessen auf einem Server bearbeitet werden. Der Wecker selbst benötigt nicht zwangsläufig eine Uhr, weil die von einem Server zur Verfügung gestellt werden kann. Die Wiedergabe von Tönen kann zudem genutzt werden, um Personen auf das Empfangen einer Nachricht oder Klingeln an der Tür aufmerksam zu machen. Durch den Einsatz eines Displays, welchess nicht nur Zahlen sondern auch Buchstaben darstellen kann, können unter anderem Nachrichten auf den Wecker gesendet werden oder das erwartete Wetter für den Tag angezeigt werden.

Ein Wecksystem aus mehreren Komponenten könnte so in einem Netzwerk zusammenarbeiten und beim ins Bett gehen sowie beim Aufstehen weitere Dienste ansprechen. Das System kann auf Dienste wie Fenster öffnen und schließen oder die Lichtsteuerung Einfluss nehmen, um so dem Menschen einen angenehmen Morgen zu bereiten.

Ein Wecksystem kann aber auch vom Smart Home gebotene Dienste benutzen. Zum Beispiel wird es erst benötigt, wenn jemand im Bett liegt. Diese und weitere Information könnten im Smart Home bereitgestellt sein, um vom Wecksystem genutzt zu werden.

Ein weiterer Dienst eines Wecksystems kann die Überwachung des Schlafs sein. Durch Aufzeichnen von Bewegungen oder Herzfrequenzen kann eine im Smart Home laufende Prozedur jeden Morgen die Qualität des Schlafes beurteilen und gegebenenfalls Aktionen einleiten. Bei schlechtem Schlaf gibt es z.B. einen extra starken Kaffee und am kommenden Abend wird die Lichtstimmung relativ früh dunkel eingestellt. Davon wird der Nutzer subtil beeinflusst, um ihn dazu zu bewegen das Bett früher aufzusuchen.

2.2.6 Alternative Bedienmöglichkeiten im Smart Home

Ein weiterer wichtiger Gesichtspunkt ist die Bedienung des Smart Homes. Auch auf diesem Bereich wird seit vielen Jahren geforscht [5]. Es wird unter anderem versucht die Bedienung so intuitiv wie möglich zu gestalten, damit jeder Nutzer auf Anhieb die Bedienung versteht. Auch Gäste sollen sich im Haus wohlfühlen und beim Gang zur Toilette nicht erst den Lichtschalter suchen müssen.

Viele Smart Homes lassen sich heutzutage über ein beliebiges Gerät mit Webbrowser steuern. So kann zwar sichergestellt werden, dass mehrere Benutzer Eingriff nehmen können, aber es setzt die Anwesenheit eines Tablets oder Smartphones voraussetzt. Auch ein seit Jahren bewährter Schalter in der Wand kann nur dann bedient werden, wenn man in seiner Nähe ist. Diese und weitere Probleme zu beheben ist Gegenstand von Technikfirmen und Universitäten weltweit. Dabei werden verschiedene Ansätze verfolgt. Ein Ansatz sind Tangible User Interfaces (kurz TUI), bei dem Nutzer mit realen Gegenständen kollaborativ Zustandsveränderung vornehmen können [23]. Diese seien bei gut entwickelter Software sehr intuitiv zu bedienen, sind jedoch ortsgebunden. Ein weiterer Ansatz ist die Erkennung von Gesten, die mit Hand oder Körper ausgeführt werden [4]. Auch Fernseher nutzen diese Technik bereits und könnten durch eine offene Schnittstelle in ein Smart Home eingebunden werden. Durch Montage von einer Vielzahl an Kameras kann eine Abdeckung der ganzen Wohnung erfolgen. Die möglichen Gesten sind jedoch begrenzt und eventuell nicht jedem bekannt, auch wenn sie intuitiv gestaltet sind ist nicht jeder in der Lage diese auf Anhieb zu verstehen. Ein weiterer Gedanke geht dahin die Position von Personen im Haus zu bestimmen, um so zugeordnete Szenarien abspielen zu können. Denkbar ist es die Leselampe einzuschalten, wenn man sich auf den Lesesessel setzt.

Die immer einfacher werdende Bedienung unseres Lebensraumes kommt auch der Entwicklung im AAL Bereich zugute wie in 2.2.2 schon erwähnt. Die einfache bis automatisierte Bedienung des Hauses hat für körperlich oder geistig Eingeschränkten einen besonderen Nutzen.

2.3 Szenario

2.3.1 Ist Analyse aktueller Stand LivingPlace

In diesem Kapitel wird das LivingPlace der HAW Hamburg vorgestellt sowie die technischen Gegebenheiten besprochen.

Vorstellung LivingPlace

Das LivingPlace ist eine voll eingerichtete Wohnung an der HAW Hamburg die zur Entwicklung von Smart Home Equipment dienen soll. Es können neue Lösungsansätze für

die Steuerung oder neue Methoden zur Automatisierung einer Wohnung erprobt werden. Weiterhin können Vor- und Nachteile von unterschiedlichen Protokollen oder Funkverbindung untersucht werden, sowie deren Zusammenspiel. Ein weiterer Zweck dieser Wohnung ist die Entwicklung auf dem AAL Bereich, dieser könnte in Zukunft immer mehr Tragweite gewinnen. Da die Lebenserwartung der Menschheit stetig steigt, wird es zunehmend mehr alte Menschen geben, die vom Ambient Assisted Living profitieren können, wie auch in 2.2.2 beschrieben.

Einschränkungen

Das LivingPlace wurde als Wohnung für eine Person eingerichtet. Die in dieser Arbeit entwickelten Komponenten und Software wird dieser Gegebenheit folgen und nicht kompatibel mit der Benutzung durch mehrere Personen sein.

Technische Gegebenheiten

Da diese Bachelorarbeit auf die aktuellen Rahmenbedingungen im LivingPlace aufbaut werden diese hier etwas erläutert. Ein Smart Home benötigt immer mindestens einen zentralen Server, von dem aus die Wohnung gesteuert wird. An der HAW wird ein Computer, welcher als Betriebssystem Linux ausführt, verwendet. Dieser Server führt fünf wichtige Programme oder Software Pakete aus, welche im Folgenden kurz beschrieben werden, um die Grundfunktionen zu verstehen. Im Verlauf dieser Arbeit werden sie immer wieder erwähnt.

NodeRed Diese Software ist ein Programmierwerkzeug, dass sich besonders dafür eignet ereignisgesteuerte Programmabläufe zu entwickeln [28]. Für eine Bedienung auf niedrigem Niveau benötigt der Nutzer keine Programmierkenntnisse, da ein Großteil der Funktionen schon durch das Verbinden einzelner Bausteine abgerufen werden kann. NodeRed basiert auf Node.js, somit auf JavaScript und kann unter anderem dadurch auch auf Rechnern mit geringer Leistung, z.b. Einplatinencomputern verwendet werden. Diese und weitere Gründe sprechen dafür, diese Software in einem Smart Home zu verwenden.

HomeAssistant Jeder Nutzer eines Smart Homes möchte die Möglichkeit genießen sein Haus oder die Wohnung über das Smartphone zu steuern. HomeAssistant eignet sich für dieses Vorhaben sehr gut. Es ist OpenSource und lässt sich von jedem Nutzer individuell einstellen [18]. HomeAssistant ermöglicht die Zusammenarbeit von Smart Home Systemen verschiedener Hersteller. So können mit einer Anwendung Marken übergreifend alle Geräte gesteuert werden. Im LivingPlace an der HAW wird diese Software außerdem verwendet, um Daten von Sensoren an eine Datenbank weiter zu leiten. Mehr dazu im nächsten Abschnitt.

InfluxDB Sensordaten, die im LivingPlace der HAW generiert werden, müssen zur späteren Auswertung gespeichert werden. In einem Smart Home sind die meistens Daten an eine Uhrzeit gebunden. Deswegen ist InfluxDB eine geeignete Datenbank für diese Anwendung. Sie wurde entwickelt um Daten in einem Zeitstrahl mit hoher Abtastrate aufzunehmen [19].

Grafana Gespeicherte Daten in einer Tabelle sind nicht besonders anschaulich und lassen sich nur schwer analysieren. Es gibt viele Tools zum Veranschaulichen von Daten, Grafana ist eines davon [16]. InfluxDB arbeitet mit Grafana zusammen, um all die aufgenommenen Daten aus dem LivingPlace zu visualisieren. Es kann von Temperaturkurven bis hin zu statistischen Auswertungen, z.B. von Anwesenheiten in Räumen, vieles ausprobiert werden.

Mosquitto - MQTT Seit 2013 wird MQTT als Protokoll der „Internet der Dinge“ standardisiert [7]. Um ein MQTT Netzwerk aufzubauen wir immer ein Server benötigt, auf dem ein MQTT-Broker ausgeführt wird. Alle Nachrichten aus dem MQTT Netzwerk gehen über den Broker. Im MQTT Protokoll gibt es Topics, auf die jedes Endgerät „subscriben“ oder „publishen“ kann. Als Broker wird im LivingPlace eine Software mit dem Name "Mosquitto" verwendet. Mosquitto [11] ist ein relativ kleines Open Source Software Paket, welches durch seine Kompaktheit auch auf langsamen Prozessoren gut läuft.

Nun wurden die Software Komponenten besprochen, doch dass alleine reicht nicht. Zur Vervollständigung des Smart Homes fehlt es noch an Hardware, die angesprochen werden kann. Teile der bereits vorhandenen Hardware sollen hier erläutert werden. Es wird dabei ausschließlich Hardware vorgestellt, die für diese Arbeit von Interesse ist.

Homematic IP Eine Produktserie der eQ-3 AG [1]. Diese Produkte werden entwickelt um im kommerziellen sowie privaten Gebrauch Anwendung zu finden. Alle Homematic Produkte werden über die eigene Zentrale von Homematic genannt „CCU3“ gesteuert. Die CCU3 kann wiederum über MQTT Befehle bedient werden. So können über verschiedene Anwendungen die MQTT Befehle senden können alle Homematic IP Produkte gesteuert werden.

Lichtsteuerung Aus den Anfängen des LivingPlace befindet sich noch immer eine DMX Steuerung in der Wohnung. Dies ist ein kabelgebundener serieller Bus, der mit 3 Leitern auskommt. Der DMX Bus wird überwiegend in der Veranstaltungstechnik zur Steuerung von Lichtanlagen verwendet. Dieses System bietet verschiedene Universen, in welchem 512 Kanäle vorhanden sind. Jeder Kanal kann einen Wert zwischen 0 und 255 annehmen. Des Weiteren wird DALI verwendet um Beleuchtungselemente zu steuern.

Auf diese Dienste des LivingPlaces soll im Laufe der Arbeit zugegriffen werden. Nur so kann die von Prof. Dr. Kai von Luck geforderte Morgenroutine realisiert werden.

2.3.2 Exemplarischer Ablauf eines Weckvorgangs

In diesem Kapitel wird exemplarisch beschrieben wie die Nacht vom ins Bett gehen bis kurz nach dem Aufstehen verlaufen kann. Dafür wurde eine Beispielperson „Kal“ erfunden.

Kal geht ins Bett. Ein Detektor erkannt dies und fragt Kal nach der Weckzeit oder ob die aus einem Kalender übernommene Weckzeit stimmt.
Kal stellt die Weckzeit auf 8 Uhr. Die Weckzeit wird hinterlegt.
Kal schläft ein. Der BuA-Detektor beginnt Kals Schlafphasen zu ermitteln.
6:45: Kal dreht sich im Bett. Der BuA-Detektor erkennt die Bewegung und weiß, dass Kal ca. die nächsten 20 Minuten im Leichtschlaf ist, um danach langsam in eine Tiefschlafphase zu gelangen.

Version 1: Kal bewegt sich um 7:45 im Bett. Der Wecker weiß nun das Kal zum Weckzeitpunkt in einer Leichtschlafphase sein wird.

Version 2: Kal bewegt sich bis 7:50 nicht. Der Wecker weiß, dass Kal in einer Tiefschlafphase ist. Es wird versucht Kal möglichst sanft aus dem Schlaf zu holen.
7:55: Der Wecker spielt einen leisen beruhigenden WeckTon ab um Kal in eine Leichtschlafphase zu bringen. Zusätzlich wird das Licht leicht angedimmt, um einen Sonnenaufgang zu simulieren.

8:00: Der richtige Weckton wird abgespielt.
Die Kaffeemaschine geht an und lässt einen Kaffee durchlaufen.
Kal wacht auf und entschiedet sich noch 10 Minuten zu dösen. Er löst die Snooze-Funktion des Weckers aus. Der Weckton wird gestoppt.
8:10: Die Snooze-Zeit ist abgelaufen, der Wecker klingelt.
8:11 Kal verlässt das Bett.
Ein Detektor erkennt das Verlassen des Betts und schaltet den Wecker ab.
Automatisch gehen die Rollos auf und die Fenster fahren auf Kipp Position.

Dieser oder ein ähnlicher Ablauf soll im Laufe dieser Bachelorarbeit umgesetzt werden, um den Anforderungen von Prof. Dr. Kai von Luck gerecht zu werden.

2.3.3 Anforderungen

In diesem Kapitel werden Anforderungen an den Wecker gestellt und erläutert.

Anforderung:	Weckzeit einstellbar	Nr.:	1
Beschreibung:	Die Weckzeit muss einstellbar sein		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.1: Anforderung: Weckzeit einstellbar

Anforderung:	Kontaktlose Quittierung	Nr.:	2
Beschreibung:	Der Wecker muss kontaktlos zu quittieren sein.		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.2: Anforderung: Kontaktlose Quittierung

Anforderung:	Anwesenheit im Bett erkennen	Nr.:	3
Beschreibung:	Das Gesamtsystem muss die Möglichkeit bieten, die Anwesenheit einer Person im Bett zu erkennen.		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.3: Anforderung: Anwesenheit im Bett erkennen

Anforderung:	Bewegungen im Bett erkennen	Nr.:	4
Beschreibung:	Der Wecker muss die Möglichkeit bieten, Bewegungen im Bett zu erkennen.		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.4: Anforderung: Bewegungen im Bett erkennen

Anforderung:	Kommunikation mit Smart Home	Nr.:	5
Beschreibung:	Der Wecker muss mit einem Smart Home Server kommunizieren können		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.5: Anforderung: Kommunikation mit Smart Home

Anforderung:	Display	Nr.:	6
Beschreibung:	Der Wecker muss über ein Display verfügen, um Uhrzeit oder anderen Text anzuzeigen.		
Stakeholder:	Student		

Tabelle 2.6: Anforderung: Display

Anforderung:	Audiosausgabe	Nr.:	7
Beschreibung:	Der Wecker muss Töne abspielen können. z.B. ein Weck-Ton		
Stakeholder:	Student		

Tabelle 2.7: Anforderung: Audioausgabe

Anforderung:	Schlafphasen erkennen	Nr.:	8
Beschreibung:	Es muss eine Möglichkeit geben, die in Anforderung 4 erkannten Bewegungen, in Schlafphasen zu überführen		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.8: Anforderung: Schlafphasen erkennen

Anforderung:	Sanftes Wecken	Nr.:	9
Beschreibung:	Mithilfe der in #8 erkannten Schlafphasen soll der Nutzer, wenn möglich in einer Leichtschlafphase geweckt werden		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.9: Anforderung: Sanftes Wecken

Anforderung:	Morgenroutine	Nr.:	10
Beschreibung:	Der Wecker muss mithilfe der in #5 genannten Smart Home Kommunikation eine Morgenroutine ausführen wie z.B.: • t-5min -> Fenster öffnen für frische Luft • t-2min -> Licht langsam an dimmen • t -> Wecker klingelt, Kaffeemaschine anmachen • t+3min -> Rollläden öffnen		
Stakeholder:	Prof. Dr. Kai von Luck		

Tabelle 2.10: Anforderung: Morgenroutine

2.4 Zentrale/Dezentrale Datenverarbeitung

Die uns heutzutage zur Verfügung stehende Rechenkapazität durch immer schneller werdende Prozessoren ermöglicht es immer neue Ansätze auszuprobieren. Ein seit Jahren an Popularität gewinnender Zweig ist das sogenannte Cloud-Computing. Dabei sitzt der Endverbraucher an einem Computer mit geringer Rechenleistung, während die Hauptarbeit von einem zentralem Server erledigt wird, auf dem gerade weitere Endverbraucher Zugriff haben. Diese Vorgehensweise kann gut auf eine SmartHome Umgebung übertragen werden, da im Smart Home viele kleine Endgeräte mit geringer Rechenleistung zum Einsatz kommen. Während ein Server Daten von den Endgeräten verarbeitet und Befehle zurück an Endgeräte sendet. Im Folgenden sollen zwei Szenarien besprochen werden.

Szenario 1: Der Wecker verarbeitet die Bewegungs- und Anwesenheitsdaten auf allen Ebenen. Von den rohen Messdaten bis zum "Wahr" oder "Falsch" für das Auslösen des Wecksignals.

Szenario 2: Der Wecker verarbeitet die Bewegungs- und Anwesenheitsdaten auf den ersten beiden Ebenen und gibt eine große Vielfalt an Daten an den Server, dieser schaltet dann die Ausgänge in der ganzen SmartHome Umgebung.

Durch die Integration der Datenverarbeitung sowie einer internen Uhr in den Wecker ist dieser weniger an ein Smart Home gebunden. Er würde auch beim Ausfall des Servers über Nacht die Uhrzeit kennen und den Schlafenden wecken. Dieses Problem kann durch Sicherheitsmechanismen behoben werden, die im Fall eines Serverausfalls pünktlich wecken. Bei einem serienreifen Produkt muss die Installation für jedermann durchführbar sein. Da eine dezentrale Datenverarbeitung wie in Szenario 2 auch immer eine Softwarekomponente mit sich bringt, muss hier geschaut werden wie diese Software in eine bestehende SmartHome Umgebung integriert werden kann. Szenario 2 bietet jedoch weitaus mehr Möglichkeiten während der Entwicklungsphase. Durch die Einsicht von allen Daten verschiedener Teilsysteme auf einem Server ist es einfacher Fehler zu verfolgen und zu beheben. Des Weiteren ist die Programmierung auf Basis von NodeRed, welches hier Verwendung finden wird, sehr einfach und effektiv. Aus diesem Grund wurde sich für eine dezentrale Version mit einem nicht unbedingt notwendigen Server entschieden.

2.5 Mögliche Lösungsansätze zur Erfüllung der Anforderungen

Bevor ein Konzept zur Umsetzung dieser Arbeit erarbeitet werden kann, sollen hier Ideen gesammelt werden, die als Entscheidungshilfe dienen sollen. Dafür werden verschiedene Möglichkeiten zur Erfüllung der Anforderungen aufgeführt. Im Verlauf dieser Arbeit werden diese wieder aufgegriffen und sich für eine der Varianten entschieden. Zur besseren Übersicht wurden die möglichen Lösungsansätze an die Anforderungstabelle angehängt und im Anhang A.1 platziert.

2.6 Konzept

Da nun die Anforderungen festgelegt und mögliche Lösungsansätze aufgelistet wurden, geht es nun daran die verfolgten Ansätze umzusetzen und in einem Gesamtkonzept zu verpacken.

Die Funktionen des Gesamtsystems können grundlegend wie folgt zusammengefasst werden. Es soll ein Weckmechanismus geschaffen werden, der Schlafphasen ermitteln kann und auf Basis dieser den Weckzeitpunkt festlegt. Weiterhin muss der Weckton durch eine berührungslose Schnittstelle zwischen Mensch und Gesamtsystem gestoppt werden

können. Die vom Weckmechanismus ermittelte Weckzeit ist außerdem ein Zeitpunkt um den herum Geräte im LivingPlace angesteuert werden sollen. Um all diese Fähigkeiten zu erfüllen wird das Gesamtsystem in verschiedene Baugruppen aufgeteilt.

Dazu gehört der Wecker, welcher als Schnittstelle zwischen Mensch und Smart Home bzw. LivingPlace dient. Ein Display kann Uhrzeiten und Texte sowie Symbole anzeigen. Ein Lautsprecher spielt Töne ab und eine Sensorik dient zur Erkennung von berührungslosen Quittierungen die zum Beispiel den Weckton stoppt.

Ein weiteres Subsystem ist ein Detektor, welcher Bewegungen sowie Anwesenheit im Bett erkennt. Diese Daten werden dem Gesamtsystem und somit dem Weckmechanismus zu Verfügung gestellt, um Aussage über die Schlafphase treffen zu können. Dieser Detektor wird von nun an BuA-Detektor genannt. Die Abkürzung "BuA" steht für Bewegung und Anwesenheit.

Nun müssen die Daten noch verarbeitet werden. Dafür wird ein Server verwendet, der mit den beiden oben genannten Geräten kommuniziert und so einen Weckmechanismus zur Verfügung stellt. Eine weitere Aufgabe des Servers ist das Ansteuern des Living Places.

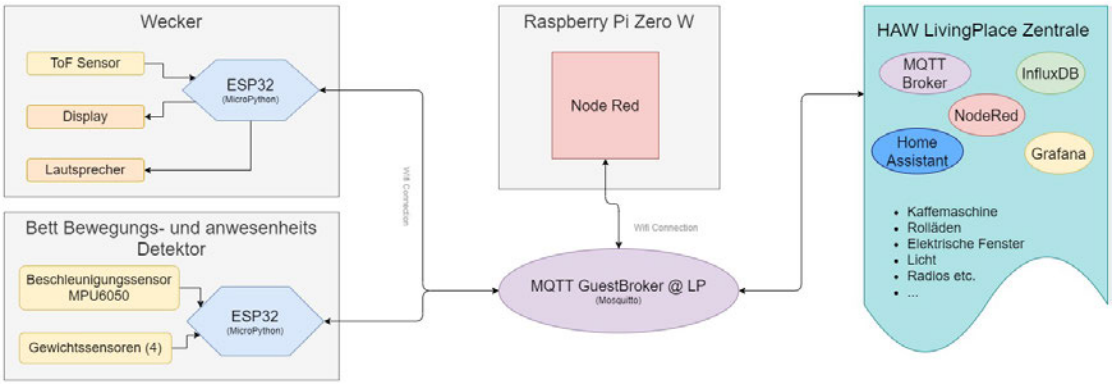

Abbildung 2.1: Grundlegender Aufbau. Einteilung in Komponenten

Eine grundlegende Entscheidung galt es darüber zu treffen, wie die Interaktion zwischen Server, Bett und Wecker funktionieren soll. Einerseits sind Kabel zur Übertragung von Daten eine bewährte Methode, andererseits könnte die Optik herumliegender Kabel die Ansprüche an einen modernen Wohnstil nicht erfüllen. Ein weiterer Punkt, der gegen

Kabel und für eine Funkverbindung spricht, ist das ein Bewegungs- und Anwesenheitsdetektor nicht nur an einem Bett benutzt werden kann. Er könnte auch in Sessel und Sofas verbaut werden. Die Anforderungen können mit dem Entwickeln von zwei Baugruppen, welche über eine Funkverbindung miteinander kommunizieren können, erfüllt werden. Dazu kommt, dass so möglicherweise ein größerer Nutzen durch die vielseitige Nutzbarkeit entsteht.

Die beiden Geräte arbeiten also unabhängig voneinander und senden die erzeugten Daten an eine Zentrale. Hier werden die Daten verarbeitet und so z.B. der Weckzeitpunkt bestimmt. Dieses Vorgehen entspricht einer modernen Herangehensweise eines dezentralisiertem Systems wie in 2.4 beschrieben. Die Daten können im Netzwerk des LivingPlace auf der Datenbank InfluxDB gespeichert und über die Anwendung Grafana visuell dargestellt werden. Ein Raspberry Pi Zero W wird mit NodeRed ausgestattet, um Funktionen zum Einlesen von Weckzeiten und zum Bestimmen der Schlafphase schreiben zu können. Dieser Raspberry Pi dient zudem als Schnittstelle zwischen dem hier entwickeltem Wecksystem und dem LivingPlace Server.

Die Anforderungen drei, vier und fünf werden mit der Baugruppe „BuA-Detektor“ umgesetzt. Ein kleines Gerät, das unter dem Bett montiert werden kann. Dort zeichnet es Bewegungs- sowie Belastungsdaten auf und verbreitet diese im Netzwerk. Belastungsdaten können genutzt werden, um Anwesenheit eines Gewichtes im Bett zu erkennen. Ab einer bestimmten Schwelle wird die Belastung als Person erkannt.

Die Baugruppe „Wecker“ erfüllt die Anforderungen zwei, fünf, sechs und sieben. Er meldet die eine berührungslose Quittierung an NodeRed und spielt auf Abruf Wecktöne ab. Gesteuert von NodeRed über den Raspberry Pi kann das Display Uhrzeiten und Nachrichten anzeigen.

Die restlichen Anforderungen eins, fünf, acht, neun und zehn werden mit Hilfe von der Software NodeRed erfüllt. Durch Heranziehen der Daten des BuA-Detektors kann hier die Schlafphase berechnet werden. Der tatsächliche Weckzeitpunkt wird anhand der Schlafphase und der eingestellten Uhrzeit berechnet. Wecktöne werden ausgelöst und gestoppt, durch das Ermitteln von Anwesenheit im Bett oder einer berührungslosen Quittierung.

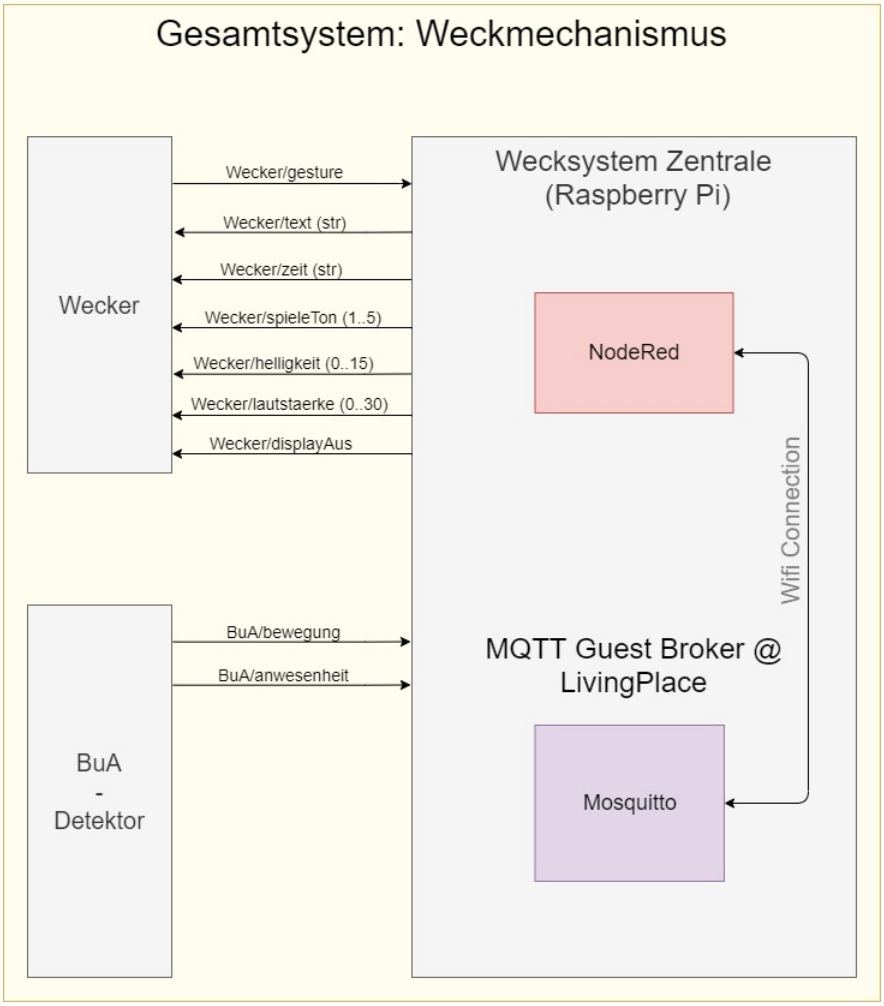

Abbildung 2.2: MQTT Topics

Beide Geräte, der BuA Detektor sowie der Wecker, benötigen einen Mikrocontroller um Sensordaten einlesen sowie aufbereiten zu können. Die Auswahl des Mikrocontrollers sowie weiterer Hardware, die im Gesamtsystem genutzt werden soll, wird im Kapitel 3 Umsetzung behandelt. Weiterführend werden Versuche mit den Komponenten durchgeführt und erläutert.

2.7 Fazit

Die Ziele dieser Arbeit und einige hilfreiche Informationen wurden in diesem Kapitel aufgeführt. Hier sollen sie noch einmal in einem Überblick zusammengefasst werden. Besonders bei der Recherche sind Fragen aufgetaucht, die dazu führten, die Anforderungen an diese Arbeit genauer zu betrachten. Dies trifft besonders auf die Schlafphasenerkennung zu, denn eine genaue Aussage über die Schlafphase kann nur in einem Schlaflabor erfolgen [39]. Da die angestrebte Erkennung von Schlafphasen so entwickelt werden soll, dass der Mensch sich nicht beim Schlafen gestört fühlt, ist das Messen von Hirnströmen und anderen körperlichen Aktivitäten nicht umsetzbar. Der Hauptgedanke dieser Arbeit ist also das Erfassen von Messwerten und deren Verarbeitung in einer Smart Home Umgebung, sowie das Einbinden eines intelligenten Gerätes in das LivingPlace der HAW. Dabei soll unter anderem erklärt werden wie aus Messwerten und einer festgelegten Zeit der wirkliche Weckzeitpunkt ermittelt wird. Als weiteren intelligenten Mechanismus des Weckers werden außerdem Versuche zur kontaktlosen Quittierung durchgeführt. Um das ganze Projekt anschaulich zu machen wird ein Demonstrator konstruiert, der alle in den Versuchen erforschten Ergebnisse in einem Produkt zusammenbringt. Dieses Endprodukt soll anschließend in das LivingPlace der HAW eingebunden und dort auf seine Funktionen überprüft werden.

3 Umsetzung

3.1 Auswahl von Sensoren, Aktoren und Module

Für jeden Anwendungsbereich gibt es eine Vielzahl an verschiedenen Sensoren, die sich je nach ihrem Einsatzgebiet mehr oder weniger für ihre Aufgabe eignen. In dieser Arbeit sollen keine neuen Sensoren oder Komponenten entwickelt werden, stattdessen geht es mehr um die Integration vorhandener Hard- und Software in ein neu entwickeltes Gesamtsystem. Zum einen geht es um die Erkennung von Bewegungen sowie Anwesenheit im Bett, zum anderen um die kontaktlose Quittierung. Es sollen Vor- und Nachteile besprochen und eine begründete Entscheidung getroffen werden.

3.1.1 Mikroprozessor Entwicklungsboard

Durch die Entwicklung im Bereich "Internet der Dinge" in den letzten Jahren hat sich eine Vielfalt an Mikroprozessoren gebildet, die für diese Zweck verwendbar sind. Die Anforderung #5 fordert eine Kommunikationsmöglichkeit zu einem Smart Home Server. Dieser ist im LivingPlace in das eigene LivingPlace LAN-Netzwerk eingebunden. Die Kommunikation über LAN ist also eine günstige Herangehensweise, weil diese schon vorhanden ist. Im LivingPlace wird neben dem kabelgebundenen LAN-Netzwerk auch ein WLAN-Netzwerk zur Verfügung gestellt. Durch die Nutzung des WLANs muss keine neue Funkstrecke, die wiederum eine Schnittstelle zwischen Funkverbindung und LAN benötigt, aufgebaut werden. Außerdem benötigen die Subsysteme Wecker und BuA-Detektor beim Einsatz WLAN kompatibler Hardware nur ein Kabel zur Stromversorgung. Eine weitere Herangehensweise ist das Benutzen von PoE [27]. Hier wird der Strom im Ethernet Kabel mitgeliefert, es ist also nur ein Kabel für 2 Funktionen notwendig. Für diese Methode wird jedoch ein besonderer Ethernet-Switch sowie eine Reihe von Ethernet Steckdosen in der Wohnung benötigt. Die Entscheidung ist demnach für die WLAN-Funkverbindung

getroffen worden. Bei der Suche nach WLAN fähigen Mikroprozessorboards stachen 4 verschiedene Entwicklungs-Boards heraus:

- ESP8266 Entwicklungsboards
- ESP32 Entwicklungsboards
- Raspberry Pi Zero W
- Arduino Nano 33 IoT

Der ESP32 ist eine Weiterentwicklung des ESP8266 und besitzt mehr I/O's sowie mehr Onboard Hardware, weshalb zwischen diesen beiden der ESP32 mit seinem marginal höheren Preis immer bevorzugt werden sollte. Durch eine zweite Hardware UART Schnittstelle kann er auch mit MicroPython gut "debugged" werden. Die Prozessorfrequenz und der vorhandene Speicher sind bei beiden dieser Modelle ausreichend für dieses Vorhaben. Der Raspberry Pi Zero W hat einen ähnlichen Formatfaktor, ist aber mehr ein Mini Computer als eine Mikroprozessor Board. Für den Raspberry gibt es eine Vielzahl an Betriebssystemen, welche überwiegend auf Linux basieren. Er bietet viele I/O's sowie Schnittstellen, hat aber Probleme dabei Anwendungen mit Echtzeitanforderungen auszuführen. Kurze Versuche mit einem Bewegungssensor haben ergeben, dass die Abtastrate zu gering ist um aussagekräftige Daten aus dem Sensor über I2C auszulesen. Der Arduino Nano 33 IoT ist wie der Name schon sagt dafür gedacht "Internet of Things" zu entwickeln. Seine Spezifikationen sind für das Vorhaben ausreichend, der Formfaktor passt und er hat sogar einen Bewegungssensor integriert. Wie die ESP Familie kann der Arduino Nano 33 IoT auch mit Python statt C programmiert werden. Der ausschlaggebende Punkt zur Entscheidung für den ESP32 war die Verfügbarkeit sowie der Preis des Produktes. Außerdem überzeugte die Vielfalt an verschiedenen Entwicklungsboards und Software Bibliotheken.

ESP32 Microcontroller Board Der ESP32 von Espressif ist ein 32bit Microcontroller mit hoher Funktionalität. Durch sein integriertes W-LAN eignet er sich hervorragend für IoT aber auch Smart Home Anwendungen. Der ESP32 wird in vielen verschiedenen Versionen angeboten, die hier nicht genau erwähnt werden. Die Speicherkapazität reicht von 4-16 Mbyte, während die CPU-Frequenz zwischen 80 und 240MHz frei eingestellt werden kann. Der Chip bietet Analoge Ein- und Ausgänge, Digitalte I/O's, Bluetooth,

Bluetooth Low Energy sowie serielle Schnittstellen (SPI, I2C, UART). Die Programmierung des Chips kann mit Assembler, C(++) sowie mit einer abgewandelten Version von Python (MicroPython) erfolgen. Der Chip arbeitet mit 3,3V Versorgungsspannung hat aber einige 5V tolerante Eingänge. Viele Entwicklungsboards haben bereits einen USB-Serial Chip sowie einen DC-DC Wandler von 5V auf 3,3V verbaut. So kann der Chip schnell und einfach per USB beschrieben werden.

Durch die Aufteilung des Projektes in zwei verschiedene Baugruppen werden auch zwei Mikroprozessoren benötigt, welche sich möglichst nicht voneinander unterscheiden. So muss sich nicht mit zwei unterschiedlichen Systemen auseinandergesetzt werden. Die Entscheidung fiel auf das WEMOS LOLIN32 v1.0.0. Entwicklungsboard. Dieses bietet genügend Speicher, I/O's, Schnittstellen und hat die vorgestellte Größe.

3.1.2 Bewegungssensor

Zur Erkennung von Bewegungen im Bett soll ein elektrischer Sensor zum Einsatz kommen. Es gibt verschiedene Ansätze, die verfolgt werden können. Ein Ansatz ist das Messen von Verformungen am Lattenrost des Betts wie es z.B. in [17] erprobt wurde. Ein weiter Ansatz ist das Messen von Bewegungen durch einen Beschleunigungssensor. Dieser kann zum einen am Lattenrost angebracht zum anderen aber auch direkt vom Probanden am Körper getragen werden.

Die folgende Tabelle zeigt eine kleine Übersicht an Sensortypen und deren Eignung durch ein Punktesystem von ganz schlecht (–) bis sehr gut (++). Unter dem Punkt "OpenSource" soll verstanden werden, ob OpenSource Software zur Verwendung in einer Bachelorarbeit verfügbar ist. Der Punkt "Drift" ist eine Messwertabweichung durch äußere Einflüsse wie zum Beispiel die Änderung der Temperatur eines Sensors. Unter dem Punkt "Änderung des Mittelwerts" soll verstanden werden wie sich der Sensor verhält wenn sich der Proband im Bett umdreht und die Belastung neu verteilt wird. Unter Genauigkeit soll verstanden werden wie akkurat der Sensor Bewegungen erkennt und ob er verschieden starke Bewegungen unterscheiden kann.

Typ	Preis	OpenSource	Drift	Änd. Mittelwert	Genauigkeit
Dehnmesstreifen	+	++	+	-	o
Kapazitiv	-	+	-	-	-
Wäge Zellen	o	++	++	+	-
Smart Watch	- -	o	++	++	++
Fitness Armband	o	o	++	+	++
Beschl. Sensor	++	++	+	+	++

Tabelle 3.1: Mögliche Sensortypen zur Erkennung von Bewegungen im Bett

Dehnmessstreifen Dehnmessstreifen gibt es in vielen Ausführungen. Häufig sind sie als Wheatstone'sche Messbrücke ausgeführt und benötigen somit mehr Hardware als einen üblichen ADC eines Mikroprozessors. Nicht nur durch die Brückenschaltung, auch weil sich die Spannungsänderung in einem sehr kleinen Fenster bewegt. Aus diesem Grund benötigt der ADC eine sehr hohe Auflösung. Häufig werden hier 24bit ADC's verwendet.

Vorteile

- schon ab einem geringen Preis verfügbar
- viele OpenSource Projekte zur Inspiration
- kaum Drift durch Wheatstone'sche Brückenschaltung

Nachteile

- eine Drehung des Probanden im Bett führt zu einer Änderung des Mittelwerts, dies könnte beim Auswerten der Daten zu Fehlern führen kann
- nicht besonders genau durch die unterschiedliche Verformung des Lattenrostes bei verschiedenen Schlafpositionen

Kapazitiv Ein kapazitives Messverfahren benötigt immer eine sehr genaue Elektronik zur Auswertung. Durch Beeinflussung des Sensors verändert sich die Kapazität des Sensors. Die Messung von Kapazitäten ist durch das Messen der Entladungszeit möglich. Diese Zeit ändert sich mit der Größe der Kapazität, hängt jedoch auch stark von Leitungswiderständen und Temperaturen ab. Es hängt also von vielen Faktoren ab, Kapazitäten genau messen zu können.

Vorteile

- Es gibt OpenSource Lösungen zur Messung von Kapazitäten

Nachteile

- der Drift ist bei Messungen von Kapazitäten relativ hoch
- nicht besonders genau durch die kleine Kapazitätsänderung
- die Auswerteelektronik zur Messung der Kapazität ist durch den Anspruch an ihre Genauigkeit nicht besonders günstig
- eine Drehung des Probanden im Bett führt zu einer Änderung des Mittelwerts, dies könnte beim Auswerten der Daten zu Fehlern führen kann

Wäge Zellen Diese können die Kraft messen die auf sie wirkt. Diese könnten unter den Beinen des Bettes oder unter dem Lattenrost montiert werden. Eine Bewegung hat häufig auch eine Verlagerung des Gewichtes im Bett zufolge, welche verarbeitet und als Bewegeung aufgefasst werden könnte. Wäge Zellen beruhen meist auf Dehnmessstreifen, benötigen somit besondere ADC's.

Vorteile

- schon ab einem geringen Preis verfügbar
- viele OpenSource Projekte zur Inspiration
- kaum Drift durch Wheatstone'sche Brückenschaltung
- die Änderung des Mittelwerts kann hier als Vorteil gesehen werden, da die Auswertung softwareseitig genau darauf baut

Nachteile

- eine Drehung des Probanden im Bett führt zu einer Änderung des Mittelwerts, dies könnte beim Auswerten der Daten zu Fehlern führen kann
- nicht besonders genau, da nur der Veränderung der Lage des Probanden im Bett gewertet werden kann. Kleine Bewegungen und Zuckungen können nicht erfasst werden.

Smart Watch Es gibt eine große Auswahl an Smart Watches, die mit Handy oder Smart Homes verbunden werden können. Sie haben fast alle einen Beschleunigungssensor verbaut und können diese Daten durch über Jahre entwickelte Algorithmen gut verarbeiten. Die Uhr muss jedoch während des Schlafens getragen werden, was nicht jeder als angenehm empfindet. Zudem sind viele dieser Uhren nicht quelloffen, dass heißt man kann die Daten nicht in eigener Software verarbeiten.

Vorteile

- sehr genaue Analyse der Bewegungen, da viele Hersteller jahrelange Erfahrung damit gesammelt haben (z.B. Garmin mit Schrittzähler Uhren)
- kein Drift durch clevere Programmierung und das Einbeziehen eines Temperatursensors im Beschleunigungssensor
- keine Änderung des Mittelwerts ebenfalls durch eine intelligente Programmierung und viel Erfahrung der Hersteller

Nachteile

- nicht quelloffen
- hoher Preis: Wer einen solchen Wecker haben möchte muss auch eine Uhr kaufen
- unbequem beim Schlafen

Fitness Armband Es gibt eine große Auswahl an Fitness Armbändern, die wenigsten sind quelloffen. Sie sind deutlich weniger unbequem als eine Smart Watch.

Vorteile

- sehr genaue Analyse der Bewegungen, da viele Hersteller jahrelange Erfahrung damit gesammelt haben (z.B. Garmin mit Schrittzähler Uhren)
- kein Drift durch clevere Programmierung und das Einbeziehen eines Temperatursensors im Beschleunigungssensor
- keine Änderung des Mittelwerts ebenfalls durch eine intelligente Programmierung und viel Erfahrung der Hersteller

Nachteile

- nicht quelloffen
- teurer als ein Mikroprozessor und Beschleunigungssensor
- unbequem beim Schlafen

Beschleunigungssensor Jedes Smartphone, jede SmartWatch oder Fitnessarmband ist mit einem Beschleunigungssensor ausgestattet. Die meisten Beschleunigungssensoren können translatorische als auch rotatorische Bewegungen erfassen. Ein solcher Sensor kann direkt an eine oder mehrere Latten des Lattenrostes angebracht werden.

Vorteile

- sehr kostengünstig verfügbar
- kein Drift durch clevere Programmierung und das Einbeziehen eines Temperatursensors im Beschleunigungssensor erzielbar
- sehr geringe Änderung des Mittelwerts durch äußere Einflüsse
- viele OpenSource Lösungen für verschiedene Sensortypen verfügbar
- hohe Genauigkeit, erkennt auch kleine Bewegungen des Lattenrostes

Nachteile

- eigene Softwareentwicklung zum Auswerten der Beschleunigungsdaten

Entscheidung

Aufgrund der praktischen Größe, dem geringen Entwicklungsaufwand und der stabil bleibenden Daten wurde sich dafür entschieden einen Beschleunigungssensor zu verwenden.

3.1.3 Anwesenheitssensor

Eine zweite wichtige Information, die ein intelligentes Wecksystem gebrauchen kann, ist, ob eine Person im Bett liegt. Denn ohne eine Person muss niemand geweckt werden. Um die Bereitstellung dieser Information soll sich eine Sensor kümmern, der die Anwesenheit einer Person im Bett erkennt. Dafür gibt es viele Möglichkeiten, wovon hier einige besprochen werden sollen.

Die Bewertungsskala kann aus 3.1.2 übernommen werden.

Typ	Preis	OpenSource	Drift	Genauigkeit
Wäge Zellen	+	++	+	++
Kraft sensibler Widerstand	++	++	-	-
Kamera	-	++	++	+
Wärmebild Kamera	- -	-	++	++
Luftdruck Sensor	o	+	+	+

Tabelle 3.2: Mögliche Sensortypen zur Erkennung von Anwesenheit im Bett

Wäge Zellen Werden in Küchen oder Körperwagen verwendet. Können somit das Gewicht genau bestimmen. Siehe 3.1.2 "Wäge Zellen".

Vorteile

- kostengünstig verfügbar
- wenig Drift durch Wheatstonesche Messbrücke
- messen bis auf wenige Gramm genau
- viele OpenSource Lösungen für verschiedene Sensortypen verfügbar

Nachteile

- zusätzlicher ADC mit 24-bit Auflösung erforderlich

Kraft Sensibler Widerstand Diese Sensoren regieren auf die Einwirkung durch ein Kraft mit einer Veränderung des Widerstands. Diese können zwischen Boden und Bett oder Bett und Lattenrost montiert werden um so Auskunft darüber zu geben ob das Lattenrost belastet ist.

Vorteile

- kostengünstig verfügbar
- viele OpenSource Lösungen vorhanden

Nachteile

- starker Drift
- geringe Genauigkeit seitens des Sensors

Kamera / Wärmebild Kamera Eine auf das Bett gerichtete Kamera kann durch Verwendung von künstlicher Intelligenz Personen klassifizieren. Dies ist auch mit Wärmebildkameras möglich. Diese haben zudem den Vorteil, die abgestrahlte Wärme eines Menschen zu erkennen.

Vorteile

- kein Drift, Kamerabild verändert sich kaum
- viele OpenSource Lösungen verfügbar z.B. OpenCV, eventuell nicht für Wärmebilder
- erzielt eine sehr hohe Trefferquote
- Wärmebildkamera benötigt kein Licht

Nachteile

- hoher Preis
- dauerhaftes Auswerten der Kamerabilder
- normale Kamera benötigt Licht

Luftdruck Sensor Mithilfe von kleinen Luftblasen aus Gummi und einem Luftdrucksensor ist es vorstellbar eine Art Waage zu bauen. Durch Zusammendrücken der Luftblase steigt der Luftdruck. Dieser Unterschied wird vom Sensor wahrgenommen. Eine solche Blase könnte in vielen Möbeln wir Sofas, Betten und vielleicht sogar an einem Toilettensitz montiert werden.

Vorteile

- die Sensoren sind durch OpenSource Lösungen auslesbar
- wenig Drift durch ausgereifte Sensortechnik
- kann durch korrekte Auswahl von Sensortyp und Luftblasenfläche genaue Ergebnisse liefern

Nachteile

- Luftdrucksensoren sind verglichen zu anderen Sensoren der Auswahl teuer
- Berechnungen der Größe der Luftblase notwendig
- Entwicklungsaufwand ist hoch

Entscheidung

Es wurde trotz des nicht sehr positiven Urteils entschieden, erst die Kraft Sensiblen Widerstände zu verwenden. Diese waren bereits vorhanden und konnten erprobt werden. Wenn diese sich nicht als praktikabel erweisen, werden die etwas teureren aber lang erprobten Wäge Zellen verwendet. Wichtig ist bei beiden dieser Sensoren, dass sie ausschließlich das Gewicht des im Bett liegenden Gegenstands beurteilen können. Es ist also vorstellbar, dass durch Belasten des Bettes durch einen Gegenstand eine Person erkannt wird, obwohl nur etwas abgelegt wurde.

3.1.4 Sensor zur Erkennung von kontaktlosen Quittierungen

Ein weiterer Aspekt, der diese Arbeit begleitet, ist seine Bedienbarkeit. Grundsätzlich wird in einem Smart Home versucht möglichst viel zu automatisieren, doch auch hier gibt es Grenzen und so müssen die einzelnen Geräte zwangsläufig mit dem Menschen interagieren können. Die Anforderungen (2.3.3) besagen, dass der Wecker die Fähigkeit besitzen soll sich kontaktlos quittieren zu lassen. Die Tabelle (3.3) vergleicht drei verschiedene Arten von Entfernungssensoren mit einer Kamera, die eine solche Möglichkeit zur Verfügung stellen können.

Unter dem Punkt "Preis" wird der Preis im Verhältnis zu den anderen Sensoren behandelt. Die Empfindlichkeit auf Licht ist hier wichtig, da der Sensor bei den ersten Sonnenstrahlen am Morgen immer noch eindeutige Daten liefern soll. Die Entfernung spielt hier nur eine nebensächliche Rolle, denn alle Sensoren können mindestens bis zur Mitte des Bettes messen, mehr ist hier nicht von Nöten. Die Anwendbarkeit ist das Resultat aus den verschiedenen Gesichtspunkten. Die Geschwindigkeit soll die Zeit darstellen, die ein System benötigt um ein Quittieren zu erkennen.

Typ	Preis	Open Source	Empf. auf Licht	Entfernung	Anwend-barkeit	Geschw.
Kamera	-	++	o	++	++	-
GP2Y0A02YK0F	++	++	- -	o	o	++
Ping)))	+	++	++	+	o	+
VL53L0X	++	++	+	+	+	++
Edisen MT0.6-U	+	++	++	- -	+	++
Spracheingabe	o	++	++	++	+	-

Tabelle 3.3: Mögliche Sensoren zur Erkennung berührungsloser Quittierungen

Kamera Sie kann mithilfe von einem Computer und künstlicher Intelligenz Bilderreihen auswerten und darin Gesten erkennen. Sie muss ständig laufen und das Bild muss überwacht werden. Kameras können nur mit Licht arbeiten und reagiert je nach Sensortyp sehr empfindlich auf Gegenlicht. Einige Kameras können nachts mit Infrarot Licht arbeiten.

Vorteile

- einige OpenSource Lösungen verfügbar, wie z.b. OpenCV
- kann mit einem gut gewählten Objektiv Gesten in einem großen Umkreis erkennen
- ist für diesen Zweck gut geeignet

Nachteile

- hoher Preis, unter anderem auch im Stromverbrauch
- durch die komplexe Berechnung die für künstliche Intelligenz notwendig ist , vergeht einige Zeit (abhängig von der Schnelligkeit des Rechners) bis die Gestenerkennung auslöst.

Sharp GP2Y0A02YK0F Dieser Sensor arbeitet mit Infrarot Licht. Die Entfernung wird Analog ausgegeben [36].

Vorteile

- kostengünstig verfügbar
- viele OpenSource Lösungen vorhanden

Nachteile

- starker Drift
- geringe Genauigkeit seitens des Sensors

Parallax Ping))) Ultraschall- Entfernungssensor Ultraschallsensoren dieser Art sind schon seit vielen Jahren verbreitet. Sie haben ein großes Sichtfeld [31] und die Entfernung kann durch Reflexionen vom Realwert abweichen.

Vorteile

- geringer Preis
- viele OpenSource Lösungen verfügbar
- nicht empfindlich auf Licht durch Benutzung von Schallwellen
- funktioniert auch ohne Licht
- Wärmebildkamera benötigt kein Licht im Dunkeln
- schnelle Messwerterfassung, schnelle Auswertung (kaum Rechenleistung benötigt)

Nachteile

- hoher Sichtbereich
- Messwerte nicht immer eindeutig durch Reflektion

VL53L0X LiDar ToF Sensors Dieser kleine Time of Flight Sensor misst Entfernungen von bis zu 2 Meter [38]. Er kann über I2C ausgelesen und programmiert werden. Das Sichtfeld des Sensors beträgt 25 Grad, was bei zwei Meter Entfernung einer Kreisfläche mit einem Durchmesser von 89cm entspricht. Er kann mit 3,3 Volt betrieben werden.

Vorteile

- diese Sensoren spielen in der Preisklasse für Hobbyelektroniker mit
- Es gibt einiges an OpenSource Bibliotheken für diesen Sensor für verschiedene Programmiersprachen.
- dieser Sensor ist kaum empfindlich auf Licht, da er mit einer exakten Wellenlänge arbeitet.
- die Geste kann in einer hohen Geschwindigkeit erkannt werden (wenig Rechenaufwand)
- funktioniert ohne Licht im Dunkeln

Nachteile

- großes Sichtfeld

Edisen MT0.6-U Dieser Sensor arbeitet kapazitiv und kann die Annäherung einer Hand, je nach Ausführung, laut Datenblatt [12] bis auf eine Entfernung von 40cm erkennen.

Vorteile

- Daten zum Auslesen des Sensor werden vom Hersteller bereit gestellt
- dieser Sensor ist unempfindlich auf Licht
- die Geschwindigkeit ist sehr hoch
- funktioniert ohne Licht im Dunkeln

Nachteile

- hohe Entfernung kann nur mit großer Sensorfläche erzielt werden

Spracheingabe Auf dem Markt erhältliche Hardware zu Erkennung von Sprache zur Steuerung eines Smart Homes können für das Quittieren des Wecktons verwendet werden.

Vorteile

- es gibt OpenSource Lösungen für dieses Vorhaben
- diese Art der Quittierung ist nicht empfindlich auf Umgebungslicht
- arbeitet auch bei hoher Entfernung zum Empfangsgerät/Mikrofon
- funktioniert ohne Licht im Dunkeln

Nachteile

- preislich höher als eine Version mit Sensoren, kann allerdings auch andere Funktionen übernehmen
- Spracheingaben müssen verarbeitet werden, dies dauert eine gewisse Zeit

Entscheidung

In der Recherche nach alternativen Bedienmöglichkeiten 2.2.6 war die Erkennung von Gesten ein interessantes Thema. Um diesen Gedanken weiter zu führen und Erprobungen daran durchzuführen wurde sich für die Erkennung von eindimensionalen Gesten für ein berührungslose Quittierung entschieden. Der Vorteil davon ist das unterschiedliche Gesten unterschiedlichen Einfluss auf das Smart Home nehmen kann. Alle vorgestellten Sensoren, ausgenommen Kamera und Spracheingabe, sind mögliche Kandidaten um dieses Vorhaben umzusetzen.

Aufgrund der hohen Geschwindigkeit, mit der die Sensoren ausgelesen werden können, sowie des geringen Preises und der Empfindlichkeit auf Licht wurde sich jedoch für den VL53L0X entschieden. Dieser wird in seinem Datenblatt [38] zudem damit beworben für eindimensionale Gesten Erkennung verwendbar zu sein. Eine weitere Bedienungsanleitung [37] erläutert den Prozess der Erkennung von Gesten unter Verwendung der ST eigenen Entwicklungsumgebung.

3.1.5 Aktoren

Für die Erfüllung von Anforderung #6 und #7 werden Aktoren benötigt. Da diese nicht von Prof. Dr. Kai von Luck gefordert wurden und nur verbaut wurden, um ein funktionierendes Gesamtsystem zu erstellen, wird hier ausschließlich erläutert wie es zu der Entscheidung kam und welche Funktionen die gewählten Aktoren bieten.

Display Da für diese Arbeit wenig Ausgaben auf dem geforderten Display geplant sind, wurde sich hier für eine einfache Variante entschieden. Ein Dotmatrix Display mit 8x32 Bildpunkten ist für alle Vorhaben ausreichend und passt vom Formfaktor gut in einen Wecker, der einem handelsüblichen nahe kommt. Außerdem ist die Programmierung sehr

einfach umsetzbar. Diese Gründe führten zu einem 8x32 LED Dotmatrix Display, welches über MAX7219 IC's zur Steuerung eines 8x8 Moduls verfügt. Für ein 8x32 Display wurden 4 dieser Module verbunden.

Audioausgabe Da die im LivingPlace verbauten Lautsprecher noch nicht funktionsfähig eingebunden wurden, musste hier eine Alternative einfache Möglichkeit gefunden wurde. Die Recherche im Internet ergab, dass der ESP32 die Möglichkeit bieten würde Audiodateien abzuspielen. Dafür müssten jedoch zusätzlich eine SD-Karte zum Speichern der Audiodateien verwendet werden, welche die SPI Schnittstelle belegen würde. Diese ist aber schon vom Display in Benutzung. Dieser Ansatz ist dementsprechend nicht durchführbar. Ein kleines Soundmodul, welches sich über UART ansprechen lässt und über eigenen Speicher verfügt war die Lösung. Dieses Modul mit dem Namen JQ6500 verfügt zudem über einen Verstärker. So ist nur der Anschluss von Lautsprechern nötig. Hier wurden zwei Lautsprecher aus einem alten Laptop verwendet.

3.2 Versuche mit Sensoren und Modulen

Bevor ein System mit mehreren Komponenten entstehen kann, müssen diese einzeln auf ihre Funktion überprüft werden. So kann ein Fehler erkannt werden, der beim Zusammenführen der Komponenten in ein Gesamtsystem entsteht. Es soll sichergestellt werden, dass alle Komponenten an den ESP32 ohne eine Kollision der Pins anzuschließen sind. Dafür wurden Pin-Belegungen bestimmt, die später in den Kapiteln 3.3 und 3.4 genauer aufgezeigt werden. Diese Belegungen haben sich bei den Versuchen ergeben.

3.2.1 BuA-Detektor Versuchsaufbauten

In diesem Kapitel werden Versuche mit Sensoren für den Bewegungs- und Anwesenheitsdetektor gemacht. Dafür wurden verschiedene Sensortypen untersucht und bewertet. Dabei gemachte Beobachtungen und Erkenntnisse sollen hier erläutert werden.

Iteration 1

Der erste Versuch galt dem Beschleunigungssensor. Aufgrund seiner hohen Verbreitung und die damit einhergehende Vielfalt an Software Bibliotheken fiel die Entscheidung für die ersten Tests auf das "MPU6050" Modul.

MPU6050 Beschleunigungssensor Dieser Chip kann physische Beschleunigungen sowie Rotationen in digitale Zahlen übersetzen. Durch ständiges Abfragen des Sensors über den integrierten I2C-Bus können Daten mit einem Mikrocontroller aufgenommen und verarbeitet werden. Diese Daten zeigen Beschleunigungswerte entlang (translatorisch) der 3 Achsen (X-Y-Z) sowie Beschleunigungen um die Achsen (rotatorisch). Der MPU6050 Chip von Invensesense wird auf verschiedenen Prototypen Boards verkauft, welche den Anschluss an einen Microcontroller vereinfachen. Die Versorgungsspannung soll im Idealfall 3,3V betragen. Somit arbeitet dieser Chip hervorragend mit dem ESP32 zusammen.

Für diesen Versuch wurde die MPU6050 Library für MicroPython [20] auf einen ESP32 gespielt. Diese kann die Werte aller sechs Achsen aus dem Beschleunigungssensor über I2C auslesen. Durch das Konvertieren der Daten in einen String konnten die Sensordaten über die zweite UART-Schnittstelle des ESP32 bereitgestellt werden. Mit Hilfe von einem FTDI Adapter, der eine serielle Schnittstelle an einem USB-Port eines Computers bereitstellt, können die Daten am Computer ausgelesen werden. Dafür wurde eine Reihe von Python Skripten geschrieben. Das erste Python Skript diente zur direkten Einsicht der Daten, es war ein Livegraph. Dieser funktionierte ähnlich wie ein Oszilloskop und half grundsätzlich beim Bewerten der Daten. Bei der Umsetzung dieses Plots mit Python half die Bibliothek „matplot.lib“ [40] die über „pip“ installiert werden kann. Die Bibliothek „serial“ [24] ermöglicht die serielle Kommunikation über den FTDI Adapter mit dem ESP32. Aus diesem Plot wurden Informationen zum maximalen Ausschlag sowie dem Zusammenspiel der Achsen entnommen. Während dieser Versuche wurde auch deutlich wie die Erdbeschleunigung die Messungen beeinflusst.

Folgende Vorüberlegung wurde getroffen: Da der Beschleunigungssensor fest am Lattenrost des Bettes befestigt werden soll, wird er keiner großen rotatorischen Bewegung ausgesetzt sein. Aus diesem Grund werden nur die translatorischen Werte berücksichtigt. Weiterhin bewegt sich der Lattenrost eines Bettes vorwiegend auf und ab, also in

Richtung der Erdbeschleunigung und von ihr weg. Durch diese Gegebenheiten wurde entschieden, dass der Beschleunigungssensor nur auf einer Achse ausgelesen wird. Abhängig von der Montage auf dem Breadboard oder einer Platine wurde sich für die Z-Achse des MPU6050 entschieden. Wobei hier jede weitere Achse genutzt werden könnte, wenn der Chip dementsprechend platziert wird.

Durch die Erdbeschleunigung wird der vom Sensor gemessene Wert beeinflusst, da diese durchgehend auf ihn wirkt. Um trotzdem einen Wert um Null herum messen zu können, wird beim Start des MicroPython Programms auf dem ESP32 eine Initialmessung ausgeführt. Dieser Initialwert wird von jedem weiteren gemessenen Wert abgezogen, um einen Messwert um Null zu generieren. So ist es einfacher die Ausschläge zu bewerten.

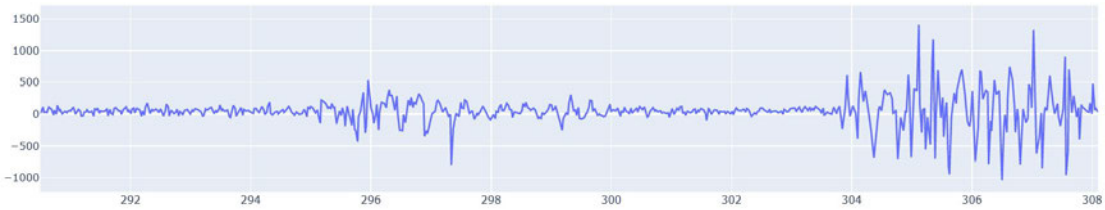

Abbildung 3.1: Daten des Beschleunigungssensors in Z-Achse mit Offset

In Abb. 3.1 wird eine Aufnahme gezeigt, bei der erst leichte, spätere stärkere Bewegungen auf das Lattenrost ausgeübt wurden. Nach einigen Versuchen dieser Art entstand folgendes Bild. Der Sensor misst auch im unbewegten Zustand Werte von ±120 – 120. Der maximale Ausschlag, der beobachtet werden konnte, lag bei bis zu ±6000. Dabei ist aber zu beachten, dass bedingt durch eine Abtastrate von 15 Werten/Sekunde wahrscheinlich nicht alle Spitzen bis zum Maximalwert aufgezeichnet werden. Die maximalen Werte können in Realität also höher sein. Dieses Verhalten ändert aber nichts an der Funktionalität des späteren Weckers, da ab einer bestimmten Schwelle die Werte als starke Bewegung aufgenommen werden.

Iteration 2

Die zweite Stufe der Evaluation beinhaltet das Einbringen des Anwesenheit-Sensors sowie eine längere Aufnahme von Daten über eine ganze Nacht. Dabei soll unter anderem geklärt werden ob die Daten einem Drift unterliegen, also ob über längere Zeit Abweichungen entstehen und ob es zu Inkonsistenzen der Daten führt.

Um die Anwesenheit einer Person im Bett zu erkennen sollen Sensoren zum Einsatz kommen, die auf das Gewicht der Person reagieren. Wie in 3.1.3 beschrieben wird im ersten Versuch der Ansatz mit Kraft Sensiblen Widerständen verfolgt.

Kraft Sensibler Widerstand Diese Sensoren sind klein und dünn, in vielen Fällen sind sie flexibel. Sie arbeiten auf der Basis von zwei verschiedenen übereinander gelegten Materialien, die unter Druck mehr Elektronen passieren lassen und somit den Widerstand reduzieren. Mit einem weiteren Widerstand kann so dieser Sensor zu einem variablen Spannungsteiler geschaltet werden, um diesen über einen ADC eines Mikrocontrollers auszulesen. Diese Sensoren gibt es in verschiedenen Ausführungen. Rechteckig sowie rund und mit verschiedenen Flächeninhalten. Die maximalen Belastungen pro Sensor reichen von wenigen Gramm bis mehrere kg.

Für diesen Versuch wurden vier Sensoren mit ca. $1cm^2$ Fläche und einer Belastung von 0-20kg verwendet. Die Sensoren für wurden wie folgt verschaltet.

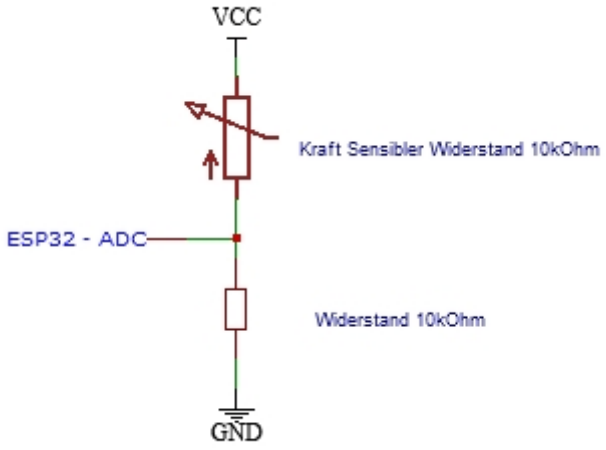

Abbildung 3.2: Kraft Sensibler Widerstand Schaltplan

Die Sensoren wurden unter die Beine eines Bettes gestellt. Um möglichst die gesamte Kraft aufzufangen, wurden kleine Holzunterlegscheiben über den Sensoren platziert. So konnte garantiert werden, dass die Kraft auf den gesamten Sensor wirkt und nicht punktuell. Da hier versucht werden soll Daten, über einen längeren Zeitraum aufzunehmen, sollte sichergestellt sein, dass möglichst viele Fehlerquellen beseitigt werden. Eine große Fehlerquelle, die bei Versuchen in Iteration 1 auffiel, war die lose Verkabelung auf einem

Breadboard. Um diesen Umstand zu vermeiden wurde ein kleiner Prototyp des BuA-Detektors auf Lochraster-Platine gebaut, siehe Abb. A.1. So konnten die Komponenten für eine bessere, sichere Verbindung verlötet werden. Nun war der erste Prototyp des BuA-Detektors einsatzbereit und es konnte eine Messreihe über eine Nacht aufgenommen werden.

Dafür wurden zwei weitere Skripte verwendet, die Daten über einen längeren Zeitraum in eine CSV-Datei aufzuzeichnen, um diese schließlich zu visualisieren. Zum Einlesen der Daten vom seriellen Port wurden Teile des ersten Skripts aus Iteration 1 verwendet. Zum Schreiben der Daten in eine CSV-Datei wurde ein FileStream verwendet. Als Bibliotheken zum Plotten von Daten aus CSV Dateien wird in dieser Arbeit überwiegend „pandas“ [30] zum Erstellen von DatenFrames aus CSV Dateien benutzt und „plotly“ [33] zum letztendlichen Plotten dieser DatenFrames verwendet.

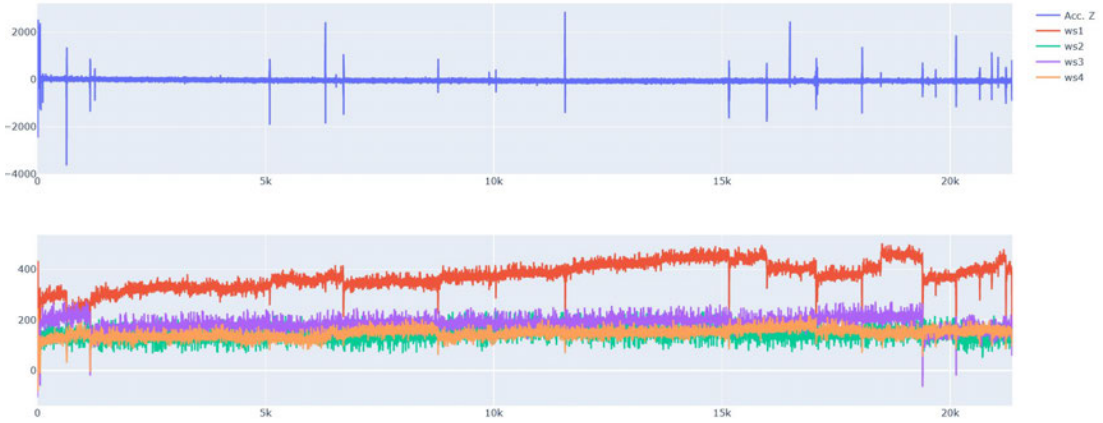

Abbildung 3.3: Aufzeichnung einer Nacht

Die Messreihe 3.3 zeigt im oberen Graph die Beschleunigung in der Z-Achse (Acc. Z), im unteren werden die Rohdaten der vier Kraftsensoren gezeigt (ws1-ws4). Wobei ws1 und ws2 die linke Seite des Betts repräsentieren und ws3 und ws4 die rechte.

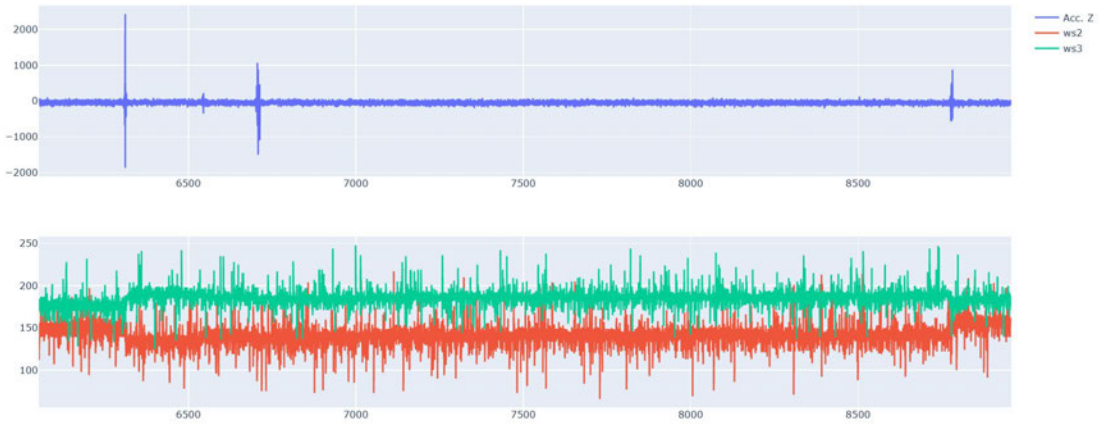

Abbildung 3.4: Nähere Betrachtung der Aufzeichnung einer Nacht

Bei genauer Betrachtung des Bereichs von 6000-9000 Sekunden sowie das Ausblenden von ws1 und ws4, die ohnehin einen starken Drift aufweisen, kann erkannt werden, dass jeweils bei einem starken Ausschlag der Beschleunigungskurve eine Veränderung der Gewichtsverteilung im Bett stattfindet. Es ist anzunehmen, dass der Schlafende sich in diesem Moment im Bett gedreht hat. Diese Information könnte später beim Programmieren der Schlafphasenerkennung interessant sein, um sie mit den Daten des Beschleunigungssensor abzugleichen.

Die Ursache für den starken Drift konnte nicht festgestellt werden. Die Eignung dieser Sensoren für diesen Einsatzzweck ist also fragwürdig. Sie sind darauf ausgelegt, zu erkennen ob sie belastet werden oder nicht, aber weniger dafür einen genauen Wert auszugeben. Des Weiteren ist das starke Rauschen der Messwerte einer schlechten Verarbeitung der Sensordaten im ESP32 geschuldet. Spätere Versuche mit einer Mittelwertbildung, über mehrere Messungen, haben bessere Ergebnisse erzielt. Dies hat jedoch keinen Einfluss auf den Drift der Sensoren.

Die Aufnahme von Daten über einen längeren Zeitraum lief jedoch problemlos. Es wurden pro Sekunde 10 Datensätze in die CSV-Datei geschrieben. Dies ist für einen Beschleunigungssensor, der für akkurate Daten sehr viel schneller ausgelesen werden muss, nicht besonders schnell, aber schnell genug um eine Idee davon zu bekommen, was überhaupt gelesen wird. Anhand dieser Langzeitaufnahme von Daten konnten wichtige Informationen gewonnen werden. Die über I2C eingelesenen Daten des MPU6050 Chips sind sehr stabil, haben also kaum Drift. Schwanken jedoch wie in Iteration eins bereits erwähnt mit ± 120 um den Mittelpunkt. Die Ausschläge des Sensors variieren jedoch in ihrer Stärke.

Die verschieden starken Ausschläge können bei einer späteren Programmierung klassifiziert werden, um mehr als nur ein "wahr" oder "falsch" zum Zustand der Bewegungen auszugeben.

Iteration 3

Da die Messwerte der Kraft sensiblen Widerstände keine präzisen Daten geliefert haben, wird ein weiterer Versuch mit anderen Sensoren unternommen. Es sollen wie in Abb. 3.1.3 bereits erwähnt Wäge Zellen verwendet werden.

Wäge Zellen Herkömmliche Wäge Zellen basieren meist auf Dehnmessstreifen in Verbindung mit einem spezifisch konstruiertem Metallteil, dass sich bei Belastung verformt. Diese Verformung ändert den Widerstand in den Dehnmessstreifen. Die in diesem Versuch verwendeten Wäge Zellen sind mit zwei Dehnmessstreifen die zu einer Halbbrücke verschaltet sind ausgestattet. Jede Wäge Zelle kann mit bis zu 50kg belastet werden.

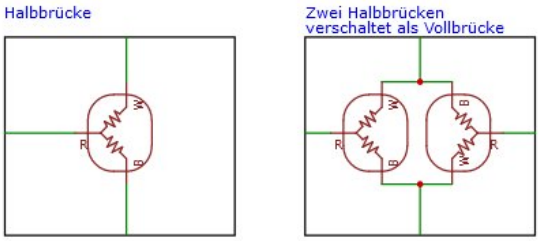

Abbildung 3.5: Verschaltung von Halbbrücken zu Vollbrücke

Die Kraft soll an 4 Punkten gemessen werden, z.B. an den Beinen des Bettes. Dafür werden vier Sensoren verwendet wovon jeweils zwei einer Seite (rechts und links) des Bettes zugeordnet sind. Durch das Verschalten zweier Sensoren die jeweils eine Halbbrücke von Dehnmessstreifen darstellen erhält man eine Vollbrücke (oder Wheatstonsche Messbrücke, siehe 3.5). Diese kann dann mit einem dafür ausgelegten ADC ausgelesen werden. Die Widerstandsänderung durch Verformung von Dehnmessstreifen ist sehr gering, weshalb der ADC eines ESP32 nicht ausreichend genau ist. Aus diesem Grund kommt hier ein spezieller ADC zum Einsatz.

HX711 Der HX711 wurde für speziell den Einsatz in Waagen entwickelt und ist auf sog. Prototypingboards mit verbauter Peripherie erhältlich. Das Auslesen erfolgt seriell und benötigt keine dedizierte Schnittstelle. Die Versorgungsspannung beträgt 2,7-5,5V. Der ADC hat eine Auflösung von 24-bit. Die Prototypingboards sind für den Anschluss von Vollbrücken vorbereitet.

Da der HX711 nur eine Vollbrücke messen kann, bot es sich an zwei der Wäge Zellen zu eine Vollbrücke zusammen zu schalten. Jede Seite des Bettes (rechts und links) wurde also mit einem Set aus zwei Wäge Zellen ausgestattet.

Als Programmcode für diesen Versuch wurde der Code aus Iteration 2 verwendet und angepasst. Der Teil des Programmcodes zum Auslesen der ADC's für die Kraft Sensiblen Sensoren wurde ersetzt. Zum Auslesen der HX711 wurde eine Bibliothek von Sergey Piskunov [32] verwendet. Über eine serielle Schnittstelle werden die Daten des Bewegungssensors und die Gewichte rechts und links übertragen.

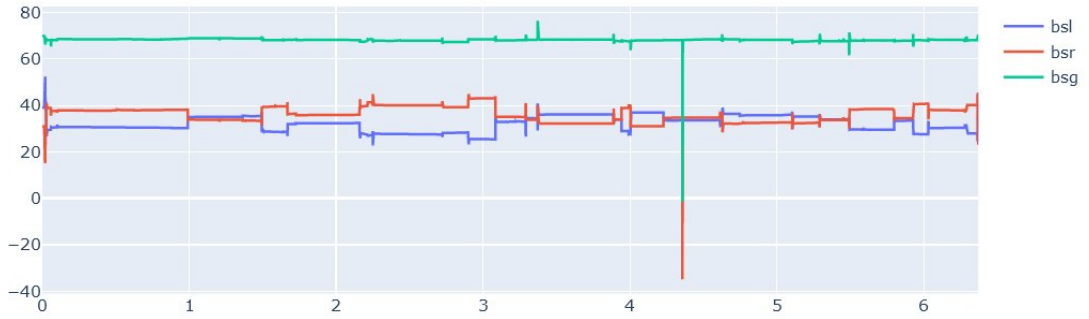

Abbildung 3.6: Betrachtung der Daten der Wäge Zellen über eine Nacht

Wie in Abb. 3.6 zu sehen, ist diese Messreihe deutlich besser ausgefallen als in 3.2.1. Die Gesamtbelastung des Lattenrostes wird durch "bsg" also dem grünen Graphen wieder gespiegelt. Hier gibt es nur kleine Abweichungen, die für das geplante Vorhaben nicht von Relevanz sind. Die rote und blaue Linie spiegeln die rechte (bsr) und die linke (bsl) Seite des Bettes wieder. Hier ist deutlich zu erkennen, dass während des Schlafs die Belastung der beiden Seiten schwankt. Auf Basis dieser Daten kann also nicht nur bestimmt werden ob sich ein Gegenstand oder eine Person im Bett befindet oder nicht sondern auch welcher Seite des Bettes der Gegenstand mehr zugewandt ist. Ein weiterer Vorteil die Seiten rechts und links getrennt zu messen ist, dass nach Erfassen einer Bewegung im Bett eine

Evaluation möglich ist, weil sich das Verhältnis zwischen "bsl" und "bsr" verändert hat. Selbst das umlegen eines Arms im Bett hat gereicht um geänderte Werte zu erfassen.

Betrachtung der Sensordaten

Abschließend zu diesem Versuch sollen die aufgezeichneten Daten besprochen werden. Hierzu wird eine gut formatierte Datenreihe verwendet 3.7. Als x-Asche ist hier nicht mehr die Sekundenzahl sondern die Stunden im Schlaf aufgezeigt. Weiterhin wurde im oberen Plot eine Cosinus-Kurve implementiert, die ungefähr aufzeigen soll, wie die Schlafphasen verlaufen würden, wenn ein Schlafzyklus 90 Minuten dauert. Im positiven Bereich sind REM-Phase sowie Leichtschlafphase, um den Nulldurchgang herum ist die Mittlere Schlafphase. Der stark negative Bereich ist als Tiefschlafphase zu interpretieren. Wie in Kap. 2.2.3 erwähnt wird nach 2-3 Schlafzyklen der Tiefschlaf nicht mehr erreicht, dieses Verhalten wird in dieser Kurve jedoch nicht wieder gespiegelt. Die Abstimmung der Cosinus-Kurve auf den wirklichen Einschlafzeitpunkt anzupassen war ebenfalls eine Schwierigkeit und konnte nicht mit voller Zufriedenheit umgesetzt werden.

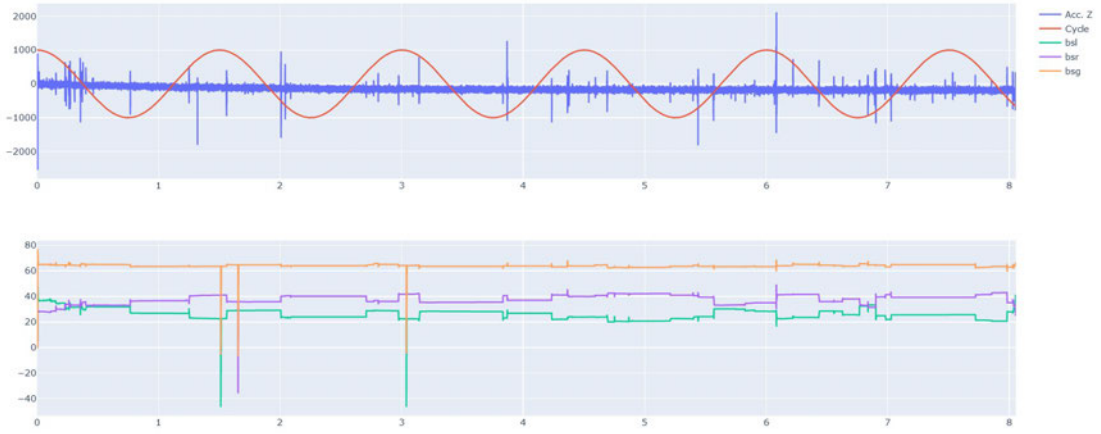

Abbildung 3.7: Aufzeichnung einer Nacht mit Wäge Zellen, Bewegungssensor und eingefügtem Schlafzyklus in Form einer Cosinus-Kurve

Die blaue Kurve Acc. Z zeigt die Beschleunigungsdaten auf. Hier ist auffällig, das im Gegensatz zu 3.3 ein Drift der Werte stattfindet. Dies kann nur durch einen Temperaturunterschied im Raum erklärbar sein, der in der vorherigen Messung nicht gegeben war. Ganz eindeutig zu erkennen ist die Zunahme an Bewegungen im letzten Drittel der

Messreihe. Dieses Verhalten ist wie in 2.2.3 erläutert darauf zurückzuführen, dass die Tiefschlafphase im Verlauf des Schlafs nicht mehr erreicht wird.

Ein weiterer interessanter Punkt ist der Abgleich von Daten des Beschleunigungssensors mit denen der Wäge Zellen. "bsl" im unteren Plot repräsentiert Bed Sensor Left und "bsr " Bed Sensor Right. Der Gesamtwert, also beide Seiten addiert wird von "bsg" repräsentiert. Mit fast jedem Ausschlag der Bewegungssensor Daten findet auch eine Veränderung der Kräfteverteilung im Bett statt. Es ist also möglich die Daten des Bewegungssensor mit denen der Wäge Zellen zu evaluieren und anders herum.

Ein negativer Punkt der in dieser Messung auffällt, die Ausschläge von "bsr" und "bsl" gen Null. Hier hat es Messfehler gegeben. Durch die relativ langen nicht geschirmten Kabel zwischen ESP32 und HX711 Boards sind vermutlich Fehler aufgetreten, die im Plot sofort auffallen. Dieses Problem muss bei der späteren Programmierung berücksichtigt werden, um nicht aus Versehen ein Aufstehen aus dem Bett zu erkennen, obwohl dies nicht stattgefunden hat.

3.2.2 Wecker Versuchsaufbauten

Der Wecker beinhaltet 4 wichtige elektronische Baugruppen, die in diesem Kapitel beschrieben werden. Die verschiedenen Baugruppen werden untersucht, sowie eine grundlegende Programmierung realisiert, um deren Funktion zu überprüfen.

Versuche in Verbindung mit der Gestenerkennung

Wie in Kap. 3.1.4 beschrieben, wurde sich für den VL53L0X Sensor entschieden. Dieser soll in diesem Kapitel untersucht werden.

VL53L0X ToF Sensor Time of Flight Sensoren starten eine Zeitmessung in dem Moment indem sie einen kurzen Lichtimpuls aussenden, warten dann auf das Eintreffen einer Reflexion des Lichtimpulses und stoppen die Zeitmessung. Die vergangene Zeit zwischen Aussenden und Eintreffen ist linear zur Entfernung des Gegenstandes der reflektiert hat. So kann dieser Sensor mit seiner sehr präzisen Zeitmessung auf einen Millimeter genau Entfernungen messen. Der Messbereich liegt zwischen 70mm und 2000mm. Die Daten gibt der Sensor über den I2C-Bus weiter. Der Vl53L0X hat ein Sichtfeld von 25°, mit steigender Entfernung steigt die abgedeckte Fläche.

Für diesen Versuch und den weiteren Verlauf wurde der VL53L0X Sensor verbaut auf einem Breakoutboard verwendet. Die ersten Versuche galten den Grundfunktionen. Dabei wurde untersucht, welche Entfernungen bei welchen Lichtverhältnissen und verschieden stark reflektierenden Gegenständen gemessen werden können. Außerdem wurde die Reaktionszeit sowie das Sichtfeld überprüft. Die Versuche ergaben, dass die Distanz welche der Sensor zuverlässig messen kann in einem Bereich zwischen 10 und 100cm liegt. Dabei wurden stark reflektierende bis matte (z.B. die Hand eines Menschen) Gegenstände verwendet um sicher zu stellen, dass unterschiedlich stark reflektierende Gegenstände die gleichen Ergebnisse erzielen. Die Überprüfung des Sichtfelds gestaltete sich schwierig, da keine Messeinrichtungen in der Nähe des Sensor angebracht werden konnten. Durch eine Untersuchung, bei der der Sensor in einen Raum ausgerichtet war, sollen die Kenngrößen des Sensor überprüft werden. Von außen wurde sich an das Sichtfeld des Sensors heran getastet. Bei Auslösen des Sensors wurde eine Markierung auf dem Boden gemacht. Es wurden die rechte und linke Seite des Sensors untersucht. Eine zusätzliche Markierung wurde genau unter dem Sensor auf dem Boden angebracht. Das so erzeugte Dreieck auf dem Boden konnte dann vermessen werden und ergab ein ähnliches Ergebnis wie die im Datenblatt [38] angegebenen 25°.

Um eine eindimensionale Geste zu erkennen können ein oder zwei Sensoren verwendet werden. Bei der Verwendung von einem Sensor ist man darauf beschränkt die Veränderung der gemessenen Distanz zu bewerten während zwei Sensoren auch das Wischen von einem zum anderen Sensor ermöglicht. Auf Grund des höheren Nutzens wurde sich hier für eine Version mit zwei Sensoren entschieden. Beide dieser Sensoren wurden über die I2C Schnittstellen des ESP32 ausgelesen. Durch das Sichtfeld von 25°, können die Sichtfelder je nach Anordnung überschneiden. Bei den Versuchen wurde festgestellt, dass die Positionierung der Sensoren eine erhebliche Rolle spielt.

Es sollen zwei Gesten erkannt werden. Das Wischen mit der Hand vor dem Wecker nach rechts und links. Hierfür wurde eine Bibliothek geschrieben, die mithilfe von endlichen Automaten (engl. state-machine) die verschiedenen Gesten auswertet. Überlegungen ergaben zwei mögliche Lösungsansätze:

Ansatz 1 Die Hand löst erst den einen Sensor aus, dann keinen, um schließlich den Zweiten auszulösen.

Ansatz 2 Die Hand löst erst den einen Sensor aus, dann beide, um schließlich nur noch den Zweiten auszulösen.

Der Erste Ansatz wurde erprobt, hat sich aber als schwierig erwiesen. Zum einen müssen die Sensoren, durch ihr Sichtfeld, entweder sehr weit voneinander entfernt sein, oder durch Verdrehung der beiden Sensoren so ausgerichtet sein, dass die beide Sichtfelder in jeder Entfernung eine Lücke aufweisen. Diese Lücke muss groß genug sein um einen Arm oder eine Hand nicht zu erkennen. Ein weiteres Problem war der Entwurf eines endlichen Automaten für diesen Ansatz. Dadurch, dass ein Zustand des Automaten der gleiche ist wie wenn gerade keine Geste ausgeführt wird, ist es schwierig ein Abbruchkriterium festzulegen. Der Übergang von "erster Sensor ist ausgelöst" zu "kein Sensor ist ausgelöst" stellt hier das Problem dar. Dieses Problem wurde durch den zweiten Ansatz behoben. Jeder Zustand dieses Automaten ist eindeutig. Falls etwas während des Ausführen der Geste schief läuft, bleibt der Automat nicht in einem Zustand des Automaten hängen. Er kann durch das Abbruchkriterium "kein Sensor ausgelöst" immer beendet werden und auf neue Betätigung warten.

Die geschriebene Bibliothek funktioniert wie folgt:

Erst werden beide Sensoren nacheinander ausgelesen. Befindet sich ein Objekt in einem voreingestellten Entfernungsfenster oder nicht, wird ein Status der Entfernungssensoren bestimmt. Diese Status sind:

- left
- right
- leftright
- nothing
- to close

Der ermittelte Status wird am Ende jedes Durchlaufs abgespeichert um beim nächsten Durchlauf bestimmen zu können, ob sich etwas geändert hat.

Wenn zwei aufeinander folgende Messungen den Status "nothing" haben, wird der Zustand des endlichen Automaten wieder auf Null gesetzt. Er muss somit von vorne Beginnen. 3.8 zeigt auf, wie der endliche Automat funktioniert. Die Bibliothek wird im digitalen Anhang im Ordner "Programmcode/Wecker " zu finden sein. Der Dateiname lautet "gesturelib.py"

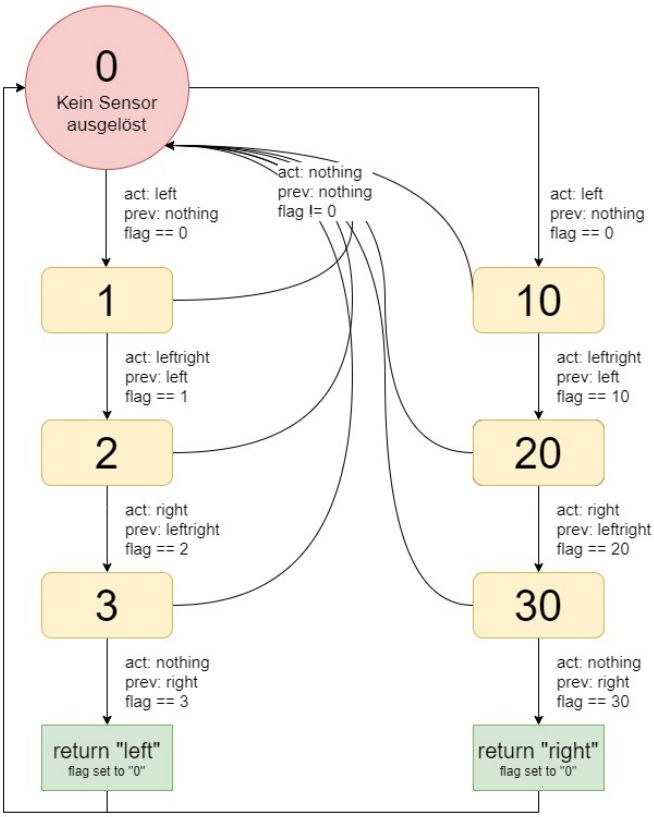

Abbildung 3.8: Endlicher Automat zur Erkennung von Gesten

Versuche in Verbindung mit dem Alarmton

Ein Wecker muss auch Wecken können. Personen können durch laute Geräusche, viel Licht oder Bewegungen am Bett geweckt werden. Im LivingPlace ist bereits einiges an Lichttechnik installiert, welches über MQTT angesteuert werden kann. Bewegungen am Bett werden außen vor gelassen, so muss sich nur auf das Erzeugen von Tönen konzentriert werden. Um hier nicht von anderen Systemen wie zum Beispiel einem Bluetooth Lautsprecher abhängig zu sein, wurde die Entscheidung getroffen ein kleines Soundmodul und Lautsprecher im Wecker zu verbauen.

JQ6500 Sound Player Dies ist ebenfalls ein kleines Prototypen Board, welches über einen USB Anschluss zum Bespielen des Chips mit Audiodateien, einen kleinen internen

3 Watt Verstärker, ein Speicherbaustein mit 16Mbit (ca. 2Mbyte) sowie einer seriellen Schnittstelle (UART) zum Bedienen des Moduls verfügt.

Dieses kleine Entwicklungsboard ist ausreichend um es in einem Demonstrator zu verbauen. Im Versuchsaufbau wurde ein ESP32 mit entsprechender Software [8] verwendet. Dieser wurde auf einem Breadboard mit dem JQ6500 Board verbunden. Die verwendeten Lautsprecher stammen aus einem alten Notebook.

Der Versuch war erfolgreich. Die Töne werden wie erwartet abgespielt. Der nächste Schritt wäre nun die Wiedergabe der Töne ausgelöst durch eine MQTT Nachricht. Dies wird im Kap. 3.4 beschrieben.

Versuche in Verbindung mit der Anzeige

Keiner mag aufgeweckt werden. Noch schlimmer ist es, wenn man dabei die Uhrzeit nicht erfährt. Aus diesem Grund und auch anhand der Anforderung #6 wurde sich entschieden ein Display zu verbauen. Größe und Nutzbarkeit sprechen für ein LED-Dotmatrix Display. Dieses kann kleine Textnachrichten sowie Uhrzeit und z.B. Temperaturen ausreichend gut wiedergeben. Es gibt viele Displays, die einen größeren Nutzen haben. Doch ist hier die Notwendigkeit nicht gegeben. Der Wecker wird pro Tag vielleicht bis zu 10 mal betrachtet, da man die meiste Zeit, die man im Bett verbringt, schläft.

MAX7219 Dot-Matrix Display Der MAX7219 ist ein Chip, um LED-Anzeigen zu steuern. Er empfängt serielle Daten, z.B. über den SPI-Bus mit bis zu 10MHz. Ein einzelner Chip kann eine Matrix von 8X8 LEDs ansteuern. Der Chip kann mühelos mit weitern Chips gleicher Bauart in Reihe geschaltet werden. So können große LED-Matrizen gebaut werden. Auch dieser Chip wird auf verschiedensten Prototypen Boards angeboten. Von 7-Segment Anzeigen bis zu LED Dot-Matrix anzeigen. In diesem Projekt wird eine einfarbige LED Dot-Matrix mit 8x32 Bildpunkten verwendet. Dafür werden Vier MAX7219 benötigt.

Auch hier ist eine Bibliothek von Mike Causer verfügbar [6], die das Anzeigen von Texten sehr einfach macht. Für lange Texte wurde eine for-Schleife entwickelt, die durch alle Buchstaben scrollt. Es entsteht eine Laufschrift. Für das Display werden die Hardware SPI Pin's des ESP32 verwendet. Diese können mit ihrer Geschwindigkeit eine hohe Bildwiederholrate gewährleisten.

3.3 Der BuA Detektor

Diese Baugruppe erfasst Bewegungs- und Anwesenheitsdaten und sendet diese über MQTT in ein Smart Home Netzwerk. Es wird alle 30 Sekunden die Belastung des Lattenrostes auf der Rechten und Linken Seite des Bettes gemessen. Innerhalb dieser 30 Sekunden wird außerdem der Bewegungssensor MPU6050 überwacht. Bewegungen werden jeweils in ihrer Stärke bewertet und gezählt, nach Ablauf der 30 Sekunden wird das ermittelte Gewicht sowie die Anzahl an Bewegungen an das MQTT Netzwerk übermittelt.

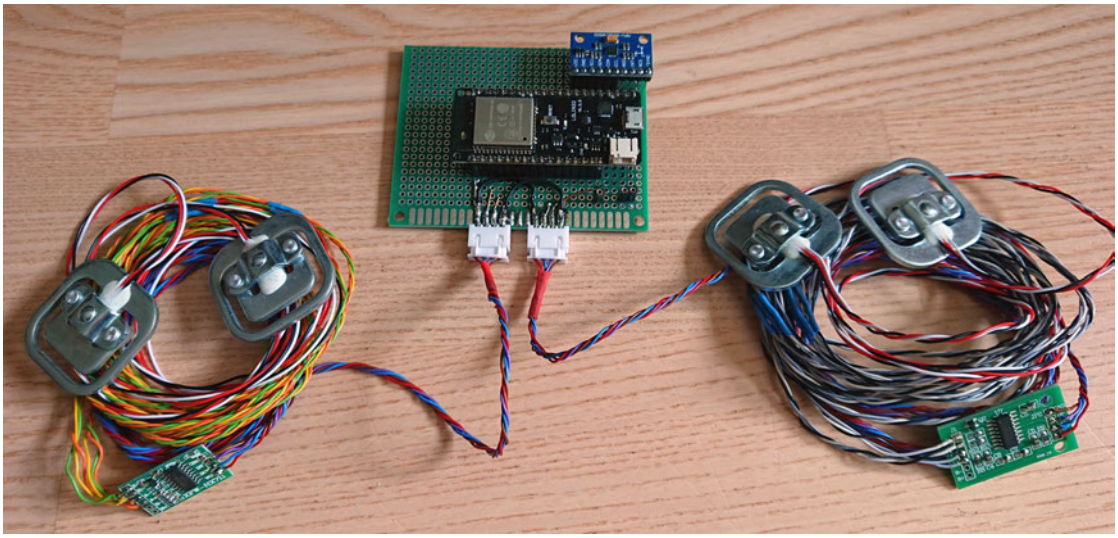

Abbildung 3.9: BuA-Detekor in finaler Austattung ohne Gehäuse

3.3.1 Schaltung

Der BuA-Detektor wurde auf einer Lochraster Platine aufgebaut. Für diese Baugruppe wurden folgende Komponenten benötigt:

- 1x Wemos LOLIN32 (ESP32 Entwicklungsboard)
- 1x Stück Lochrasterplatine
- 4x Wäge Zelle
- 2x HX711 Entwicklungsboard

- 2x 4-pol JST Stecker/Buchse
- Draht und Kabel

Der Schaltplan des BuA-Detektors ist im Anhang A.3.2 dieses Dokumentes und im digitalen Anhang im Ordner "Schaltpläne" zu finden. Dieser und alle andere Schaltpläne dieser Bachelorarbeit wurden mit dem Online-Editor EasyEDA [9] angefertigt.

Der Schaltplan zeigt, wie die einzelnen Komponenten miteinander verschaltet sind. Als Stromquelle wird ein Mikro USB Kabel verwendet. Dieses kann an einen beliebigen USB Port gesteckt werden um den BuA-Detektor mit 5V zu versorgen.

3.3.2 Konstruktion

Hier wird besprochen wie für den BuA-Detektor ein Gehäuse entworfen wurden und warum nicht die gesamte Elektronik auf einer Platine untergebracht wurde.

Sensoren an einem Bett zu verbauen zieht ein Problem nach sich. Durch die verhältnismäßig langen Strecken die von Kopf bis Fußende zurückgelegt werden, muss die Verlegung der Kabel wohl bedacht sein. In diesem Fall werden Wäge-Zellen verwendet. Wie in 3.2.1 bereits erwähnt, ist die Änderung des Widerstandes in einer Wäge-Zelle sehr klein. Somit könnte der Widerstand in einem langen Kabel zu Messabweichungen führen. Aus diesem Grund wurden die HX711 Entwicklungsboard extern verbaut, so müssen die Kabel von Sensor zu HX711 Entwicklungsboard nur knapp über einem Meter lang sein um am Bett verbaut werden zu können. Die Kabel zur seriellen Schnittstelle des HX711 vom ESP32 dürfen ruhig etwas länger sein, da hier der Widerstand keine so große Rolle spielt.

Der BuA Detektor soll am Lattenrost des Bettes befestigt werden können. Das Gehäuse wurde so entwickelt, dass ein Anbringen möglichst einfach ist. Durch die Anordnung von Ösen soll es dem Nutzer einfach gemacht werden den Detektor mit Kabelbindern am Lattenrost zu fixieren. Symbole auf dem Gehäuse sollen außerdem verdeutlichen, in welcher Orientierung das Gerät verbaut werden muss, um eindeutige Messergebnisse zu erzielen, sowie welcher Steckplatz welcher Seite des Bettes zugehörig ist. Wie auf Abb. 3.9 zu sehen ist gibt es die Grundplatine (oben im Bild) und zwei Kabelbäume mit eingebundenem HX711 Prototypingboard. Die zwei weißen Steckplätze sind fest an eine Seite des Bettes gebunden, wobei die Kabelbäume an jeden der Steckplätze funktionieren.

3.3.3 Funktionsbeschreibung

Bevor dieses Gerät in einem MQTT Netzwerk nutzbar ist, müssen Einstellungen im Programmcode vorgenommen werden. Als erster wichtiger Schritt wird die richtige WLAN SSID sowie das Passwort des Netzwerkes angegeben. Daraufhin muss auch der richtige MQTT Broker eingerichtet werden. Hierzu wird Port und IP-Adresse des MQTT Brokers eingestellt. Bevor das Gerät endgültig am Strom angeschlossen wird, sollte es bereits am Lattenrost montiert sein. Auch die Wäge Zellen sollten platziert sein, da beim Start des Gerätes eine Initialisierung stattfindet. Wie bei einer handelsüblichen Küchenwaage wird die aktuelle Belastung auf die Lattenrost als 0kg angenommen. Auch der Bewegungssensor, der durch die Erdbeschleunigung sowie durch geringe Verdrehung der Z-Achse zum Erdmittelpunkt leicht geänderte Werte erfasst, wird beim Start einmalig ausgelesen. Dieser ausgelesene Wert wird von jedem weiteren Wert abgezogen um so Werte um Null zu erlangen.

Nach dem die Initialisierung beendet wurde, wird regelmäßig eine Nachricht im MQTT Netzwerk publiziert. Die enthaltenen Informationen sollen hier besprochen werden.

Die übermittelten Daten sind einerseits die gemessenen Gewichte auf der rechten sowie linken Seite des Betts. Andererseits werden Daten, die auf die Aktivität im Bett schließen lassen, übermittelt. Diese Sammlung an Daten wird alle 30 Sekunden über MQTT versendet.

3.3.4 Programmcode

Hier soll kurz auf die Funktionsweise des Programmcodes eingegangen werden. Der eigentliche Programmcode wird im digitalen Anhang der Bachelorarbeit im Ordner "Programmcode/BuA-Detektor" zu finden sein.

In einem endlosen Loop wird der Bewegungssensor ausgelesen, hierfür wird die MPU6050 Library für MicroPython [20] verwendet. Dieser Vorgang wird alle 30 Sekunden durch einen Timer unterbrochen. Während dieser Zeit wird die Anzahl kleiner und großer Ausschläge gezählt. Ein kleiner Ausschlag wird gezählt, wenn er zwischen 250 und 500 liegt, ein großer Ausschlag wird gezählt wenn er über 500 liegt. Bevor die gesammelten Daten über MQTT übermittelt werden, wird das Gewicht über die HX711 Boards, welche mit der [32] Bibliothek ausgelesen werden, ermittelt. Nun können die gezählten Ausschläge sowie das Gewicht in ein MQTT Netzwerk publiziert werden.

Jede MicroPython Installation auf einem ESP32 funktioniert wie folgt. Generell können, bis der Speicher voll ist, unbegrenzt Dateien gespeichert werden. In diesen Dateien können Bibliotheken oder verschiedene Teile eines Programmes gespeichert sein. Die beiden Hauptdateien jedoch sind immer mit den Namen "main.py" und "boot.py" gespeichert. Bei jedem Hochfahren des ESP32 wird erst die "boot.py", dann die "main.py" Datei ausgeführt.

3.4 Der Wecker

Die Entwicklung eines Smart Home Weckers bietet die Möglichkeit den Nutzen, über die üblichen Funktionen eines Weckers hinaus, zu erweitern. Dieses Thema wurde in Kap. 2.2.5 bereits aufgegriffen. So soll dieser Wecker als Schnittstelle zwischen Mensch und Weckmechanismus dienen. Er trägt keine besondere Intelligenz in sich.

Abbildung 3.10: Vollständig zusammengebauter Wecker

Um alternative Bedienmöglichkeiten, siehe Kap. 2.2.6, zu erforschen, wurden Sensoren verbaut, die zur Erkennung von Gesten genutzt werden können. Erkannte Gesten können dann im MQTT Netzwerk publiziert werden können, um eine Aktion im Smart Home Netwerk auszulösen. Außerdem dient der Wecker auch als Aktor im Smart Home. Durch sein integriertes Dotmatrix LED Display kann der Wecker Symbole, Texte und Uhrzeiten wiedergeben. Ein kleines Soundmodul kann vorprogrammierte Töne und Melodien im

*.mp3-Format abspielen. Alle Funktionen, die der Wecker bietet, können über MQTT bedient werden.

3.4.1 Schaltung

Auch der Wecker basiert auf einem ESP32 Entwicklungsboard (WEMOS LOLIN32). Dieses wurde mit Stiftleisten auf einer Lochrasterplatine verbaut, um die Ein- und Ausgänge mit der verwendeten Hardware verbinden zu können. Im Wecker wurden folgende Komponente verbaut.

- 1x Wemos LOLIN32 (ESP32 Entwicklungsboard)
- 1x Stück Lochrasterplatine
- 2x VL53L0X Breakout Board (GY-VL53L0XV2)
- 1x JQ6500 Soundmodul (2 MB Flash-Speicher)
- 2x Lautsprecher von altem Laptop
- 1x LED Dotmatrix Anzeige 8x32 Pixel mit MAX7219 Chips
- 4x 4-pol JST Stecker/Buchse
- Draht und Kabel

Der Schaltplan des Weckers ist im Anhang A.3.2 dieses Dokumentes und im digitalen Anhang im Ordner "Schaltpläne" zu finden.

Der Schaltplan zeigt, wie die einzelnen Komponenten miteinander verschaltet sind. Als Stromquelle wird ein Mikro USB Kabel verwendet. Dieses kann an einen beliebigen USB Port gesteckt werden um den BuA-Detektor mit 5V zu versorgen.

3.4.2 Konstruktion

Ein Wecker ist kein Produkt, was ausschließlich auf die Funktionalität abzielt. Er soll optisch in einer Wohnumgebung nicht negativ auffallen. Da der Wecker für das LivingPlace der HAW gebaut wurde, welches sich mit einer zeitlosen Inneneinrichtung präsentiert, wurde auch beim Konstruieren ein gewisser Wert auf die Optik gelegt. Ein weiterer wichtiger Punkt den es zu beachten galt, war die Möglichkeit zur Verstellung der VL53L0X Sensoren. So konnte bei Versuchen verschiedene Winkel und Ansätze zur Auswertung erprobt werden. Mit diesen beiden Faktoren begann dann die Konstruktion des Weckers in einem CAD Programm, Autodesk Inventor. Alle gegebenen Maße von Display, ESP32 Entwicklungsboard usw. wurden gemessen und bei der Konstruktion berücksichtigt. Das Ergebnis dieser Entwicklung ist in Abb. 3.11 zu sehen.

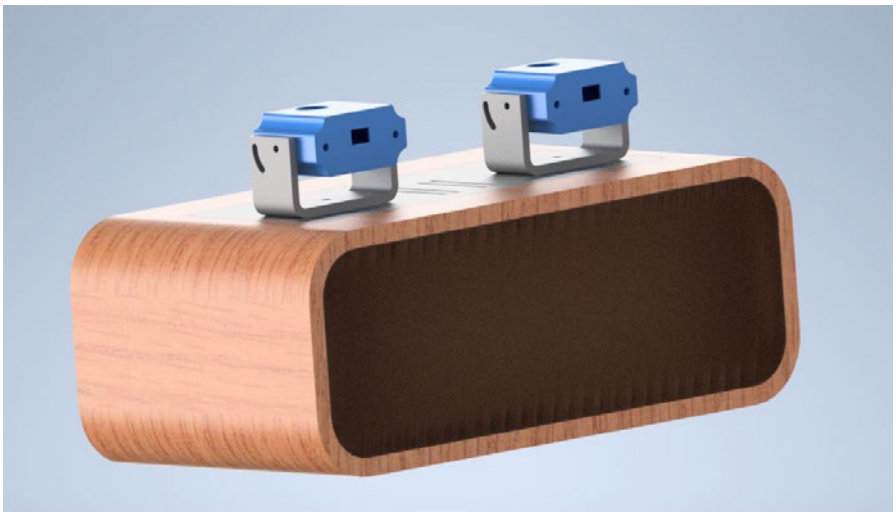

Abbildung 3.11: Vollständig zusammengebauter Wecker in Autodesk Inventor

Die Herstellung der Bauteile erfolgte auf einem 3D-Drucker (FDM) und einer CNC Portalfräse. Der Grundkörper des Weckers besteht aus Holz, genauer 18mm Multiplex Platten. Um mit den gegebenen Maschinen ein Gehäuse bauen zu können wurde der Wecker in 5 Teile geschnitten. Diese konnten dann einzeln gefertigt werden, um sie anschließend zu verkleben. Ein Einblick in die Fertigung wird im Anhang von A.5 bis A.8 gegeben. Front- und Rückseite des Weckers werden von transparenten aber milchigen Plexiglas-Scheiben geziert. Auf der Oberseite des Weckers befindet sich eine weitere Öffnung. Diese soll eine Weiterentwicklung des Weckers ermöglichen. Die Abdeckung dieser Öffnung ist im 3D Drucker entstanden und bietet die Möglichkeit Lautsprecher sowie die VL53L0X

Sensoren zu befestigen. Dieses Modul könnte in Zukunft auch durch ein anderes Modul mit anderen Sensoren wie z.b. einer Kamera ersetzt werden. Der hergestellte Wecker ist in Abb. 3.10 zu sehen.

Eine Technische Zeichnung des Wecker ist im digitalen Anhang diese Dokumentes in dem Ordner "Zeichnungen" zu finden.

3.4.3 Funktionsbeschreibung

Auch dieses Gerät muss für die Nutzung im Programmcode eingestellt werden. WLAN SSID, Passwort sowie MQTT Broker müssen händisch eingetragen werden. Nun kann der Wecker mit einem Mikro USB Kabel mit Strom versorgt werden. Wenn sich der Wecker im Netzwerk eingewählt hat, stellt er folgende Funktionen zur Verfügung:

- Geste "Links" erkennen und an MQTT Netzwerk senden
- Geste "Rechts" erkennen und an MQTT Netzwerk senden
- Töne abspielen
- Lautstärke einstellen
- Text/Uhrzeit auf Display anzeigen
- Laufschrift auf Display anzeigen
- Display Helligkeit einstellen

Die dafür verwendeten Topics im MQTT Netzwerk sind in Abb. 2.2 aufgezeigt worden und werden hier noch etwas näher beschrieben. Alle Topics, die dem Wecker angehören, beginnen mit "Wecker/"

Wecker/text Auf dieses Topic kann eine Textnachricht gesendet werden, also eine Zeichenkette. Ab einer Länge von fünf Zeichen wird diese Nachricht als Lauftext auf dem Wecker angezeigt.

Wecker/zeit Auf dieses Topic kann ebenfalls eine Textnachricht gesendet werden. Diese Zeichenkette muss eine vierstellige Zahl enthalten, die als erste zwei Zahlen die Uhrzeit in Stunden enthält und als zweite zwei Zahlen die Minuten. Hier ist darauf zu achten das eine führende Null dieser beiden Zahlen sehr wichtig ist, da sonst das Format nicht passt.

Wecker/helligkeit Auf dieses Topic reagiert der Wecker mit einer Veränderung der Bildschirmhelligkeit. Die Payload für dieses Topic muss eine Zahl zwischen 0 und 15 enthalten.

Wecker/displayAus Auf dieses Topic reagiert der Wecker unabhängig von der Payload mit einem Ausschalten des Displays. Es werden alle Punkte der Matrix ausgeschaltet.

Wecker/spieleTon Mit diesem Topic kann das Soundmodul des Weckers angesprochen werden. Als Payload wird eine Zahl übermittelt, die den Titeln des auf dem Modul gespeicherten Melodien entspricht. Eine "1" als Payload spielt den ersten Titel ab. Eine übertragene "0" stoppt das Abspielen des gerade gespielten Titels.

Wecker/lautstaerke Mit diesem Topic kann die Lautstärke im Soundmodul des Weckers angesprochen werden. Als Payload wird eine Zahl zwischen 1 und 30 übermittelt. 1 ist die leisten Stufe, 30 die Lauteste.

3.4.4 Programmcode

Hier soll kurz auf die Funktionsweise des Programmcodes eingegangen werden. Der eigentliche Programmcode wird im digitalen Anhang der Bachelorarbeit im Ordner "Programmcode/Wecker" zu finden sein.

Im endlosen Loop werden die beiden VL53L0X Sensoren über eine von [25] geschriebene Bibliothek ausgelesen und über die Bibliothek "gesturelib.py" werden die gemessenen Daten bewertet. Die "gesturelib.py" gibt als Rückgabewert zurück ob, eine Geste ausgeführt wurde und wenn ja in welche Richtung (rechts oder links). Eine weitere Funktion, die in

jedem Durchgang des Loops ausgeführt wird, ist das Überprüfen auf neue MQTT Nachrichten. Nach Eintreffen einer Nachricht wird in einer Funktion durch eine if-Abfrage auf das MQTT Topic überprüft, welche Funktion diese Nachricht ansprechen möchte. Die Payload wird anschließend dafür benutzt, die in Kap. 3.4.3 erwähnten Funktionen zu bedienen.

3.5 Weckmechanismus

Durch die Entwicklung zwei getrennter Systeme wird eine Schnittstelle benötigt mit der Wecker und BuA-Detektor untereinander kommunizieren können. Beide Geräte werden mit dem MQTT Guest Broker des HAW LivingPlace verbunden. Über diese können sie miteinander kommunizieren. Des weiteren wird ein projekteigener Raspberry Pi mit installiertem NodeRed verwendet, um die Daten der Sensoren auswerten und verarbeiten zu können. Über diesen können dann alle für das LivingPlace wichtige Daten, wie z.B. Nachrichten an Rolläden oder Bewegungsdaten des BuA-Detektors an die Datenbank InfluxDB, publiziert werden.

Die NodeRed erfasst die Daten aus dem BuA-Detektor und verarbeitet diese weiter um Annahmen über die aktuelle Schlafphase zu machen. Sobald der Weckmechanismus gestartet wurde, werden Lampen und Geräte im LivingPlace angesteuert, um eine möglichst automatisierte Morgenroutine abzuspielen. Auch das Ansteuern des Weckers wird über diese NodeRed ermöglicht. Der Aufbau des Systems wurde bereits in Abbildung 2.1 aufgezeigt.

Alle vom BuA-Detektor gesammelten Daten werden auf der Datenbank InfluxDB gespeichert. Dabei wird alle 30 Sekunden von einem Algorithmus in NodeRed anhand der BuA-Detektor Daten bestimmt, ob die Aktivität im Bett nicht vorhanden (0), schwach (1) oder stark (2) war. Dadurch entsteht eine Datenreihe über die ganze Nacht die den Schlaf bewertet. Dies sind die einzigen Daten, die zur Erkennung der Schlafphase dienen. Der in 2.2.3 erklärte Rhythmus wird nicht bei jeder Person eingehalten werden können, dient jedoch als Anhaltspunkt für folgende Überlegungen. Durch Betrachtung der letzten 60 Minuten aufgeteilt in drei 20 Minuten Blöcke könnte darüber Aussage getroffen werden, welche Phasen vermutlich als nächstes kommt. Grund für die Wahl von 20 Minuten Blöcken ist die Schlaflatenz, die in Kap. 2.2.3 erläutert wurde. Diese gibt an, wie lange sich ein Mensch in einer der 5 Schlafphasen aufhält. Der Mechanismus zur Berechnung

des tatsächlichen Weckzeitpunktes wird erst 20 Minuten vor dem eingestellten Weckzeitpunkt (WZ) gestartet. Abb. A.4 zeigt auf wie die Entscheidung getroffen wird. Als Beispiel wird eine der Entscheidung hier kurz erklärt. Wenn die letzten 60 Minuten ausschließlich viel Aktivität im Bett erkannt wurde, geht der Algorithmus davon aus, dass als nächste Phase die Tiefschlaf- oder REM-Phase durchlaufen wird. Dementsprechend wird versucht den Schlafenden noch in einer der drei anderen Phasen zu wecken.

Mithilfe von NodeRed wird die Datenbank ausgelesen. Daraufhin müssen die Werte der letzten 60 Minuten in drei Abschnitte aufgeteilt werden. Innerhalb dieser Abschnitte wird eine Mittelwertbildung durchgeführt. Die Werte dieser Mittelwertbildung wird dann auf eine Art Wahrheitstabelle wie in Abb. A.4 gezeigt angewendet. Wert `X` ist variabel einstellbar um bei Tests noch Einstellungen vornehmen zu können.

3.5.1 Die verschiedenen Weckmodi

Wie in den Anforderungen unter Kap. 2.3.3 bereits erwähnt soll es die Möglichkeit geben den Schlafenden sanft zu Wecken, siehe Tab. 2.9. Es müssen also verschiedene Weckmodi programmiert werden.

Normal Dieser Modus ist sehr einfach umzusetzen, denn es wird die eingestellte Weckzeit als tatsächlicher Weckzeitpunkt übernommen.

Sanftes Wecken Hier wird versucht den Schlafenden möglichst sanft zu wecken. Dafür wird versucht durch Abspielen von leisen Tönen und sehr gedimmten einschalten von Lichtern eine schlafende Person in eine Leichtschlafphase zu überführen. Daraufhin wird einige Minuten später der normale Weckvorgang gestartet. Es folgt eine kurze Erläuterung in Schritten:

1. Der Weckmechanismus 3.5 erkennt die Notwendigkeit für Sanftes Wecken
2. Indirekte Beleuchtung in der Nähe des Bettes wird auf sehr geringer Helligkeit eingeschaltet.
3. Leise beruhigende Geräusche werden auf dem Wecker 3.4 abgespielt.
4. Weckzeitpunkt wurde erreicht

5. Normale Weckmelodie wird abgespielt

3.5.2 Morgenroutine

Hier kommen wir zurück auf einen Punkt, auf welchen diese Arbeit unter anderem fokussiert. Das Bedürfnis dem Menschen ein möglichst sorgenfreies Leben zu ermöglichen treibt uns dazu, dass immer mehr Technik Einzug in unser Leben erhält. Nach einem erholsamen Schlaf und intelligent gewähltem Weckzeitpunkt, soll zudem das Aufstehen erleichtert werden. Dabei sollen einige morgendliche Handlungsabläufe vom Smart Home übernommen werde. Generell ist es vorstellbar, jedes erreichbare Gerät in einem Smart Home zu steuern und an die Bedürfnisse eines Menschens am Morgen anzupassen. Dazu gehören zum Beispiel Lichtszenarien oder das Öffnen von Fenstern um frische Luft herein zu lassen. Auch das Öffnen von Jalousien oder Einschalten der Kaffeemaschine oder Teemaschine kann durch diese Morgenroutine gesteuert werden. Um Punkte wie diese umzusetzen wird auf NodeRed zurückgegriffen. Hier wurde eine Routine erstellt, die einige diese Dinge aufgreift und im LivingPlace umsetzt.

Die verschiedenen Weckmodi sowie das Verschieben des Weckzeitpunktes durch die Schlafphasenerkennung haben Einfluss auf die Morgenroutine. Durch das vorzeitige Wecken können eventuell nicht alle Punkte in der Morgenroutine wie geplant umgesetzt werden. Hier wird exemplarisch eine im Livingplace umsetzbare Morgenroutine aufgezeigt, die bei normalem Weckmodi ablaufen könnte. Der Weckzeitpunkt wird in 3.12 als t abgebildet.

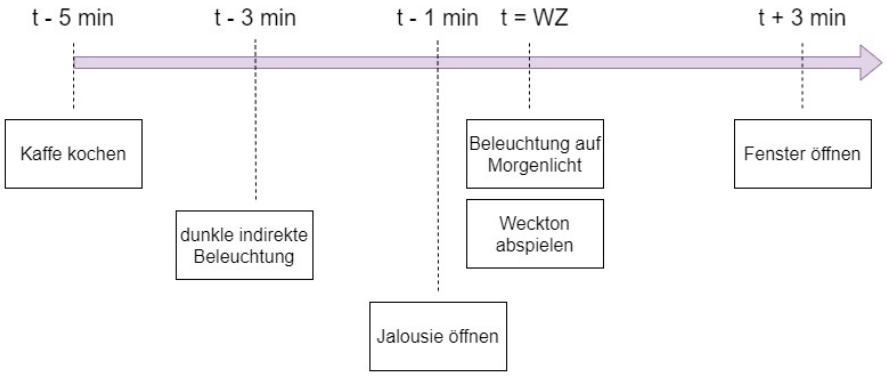

Abbildung 3.12: Morgenroutine

3.5.3 User Interface

Auch das Bedienen des Weckers soll leicht von der Hand gehen. Einerseits kann die Snooze Funktion des Wecker mit der Gestenerkennung des Weckers ausgelöst werden, andererseits müssen auch Einstellungen z.B. über den WZ (siehe 2.1) vorgenommen werden können. Um diese und weitere Einstellungen vornehmen zu können, wurde mit Hilfe der NodeRed Dashboard-Node ein brwoserbasiertes Einstellungsblatt entworfen.

Es können folgende Einstellungen vorgenommen werden:

- Weckzeitpunkt
- Snooze Zeit
- Welche Geräte sollen morgens bedient werden ?
- Zu welchem Zeitpunkt soll welches Gerät bedient werden ?

3.6 Einbringen des Wecksystems in das LivingPlace

Dieses Kapitel soll zusammenfassen, was alles beachtet werden muss, wenn ein neues Projekt im LivingPlace installiert werden soll. Generell ist zu sagen, dass viele Aufgaben vom Admin des LivingPlace übernommen werden. Dies ist wichtig, da sonst ein Durcheinander an Servern und Software im LivingPlace entstehen würde. Für Projekte wie dieses wurde eine MQTT Guest Bridge eingerichtet. Diese ist der MQTT Broker über den alle MQTT Daten des Projektes laufen. So wird der eigentliche MQTT Broker nicht mit vorerst unwichtigen Datenstreams belastet. MQTT Daten die wirklich im bestehendem LivingPlace Setup benötigt werden, werden vom Admin an den Haupt-Broker weiter gesendet.

Um MQTT Daten in einem Graph in Grafana darzustellen, muss in der Software HomeAssistant ein Sensor angelegt werden. Dieses Modul wird mit MQTT Daten gefüttert und HomeAssistant sendet diese Daten dann an die LivingPlace interne Datenbank InfluxDB. Grafana kann dann auf die in der Datenbank gespeicherten Werte zugreifen und Grafen erstellen. Auch diese Schritte werden vom Admin des LivingPlace durchgeführt um eine möglichst einheitliche Programmierung des LivingPlace zu gewährleisten.

Im Fall dieser Arbeit wird wie in 3.5 bereits erläutert ein Raspberry Pi verwendet. Dieser wird dafür verwendet projekteigene Daten zu verarbeiten und daraus folgende Nachrichten im LivingPlace Netzwerk zu publizieren. Diese oder ähnliche Vorgehensweisen sollten bei Projekten im LivingPlace angestrebt werden. Auf diesem Weg können kleine Subsysteme erstellt werden, die einzeln und unabhängig vom LivingPlace getestet werden können.

3.7 Evaluation

Ein wichtiger Schritt jeder Arbeit ist das Rekapitulieren der erbrachten Leistung. Dieses Kapitel soll die Anforderungen aus 2.3.3 wieder aufgreifen und betrachten inwieweit diese umgesetzt werden konnten.

Für alle in Kap. 2.3.3 gestellten Anforderung wurde eine Lösung gefunden. Die Tabellen wurden in Anhang A.6 mit dem Lösungsweg sowie Ergebnis vervollständigt. Grundsätzlich kann gesagt werden, dass alle Anforderungen die mit Aktoren zu tun haben in vollem Umfang umgesetzt werden. Bei den Sensoren hingegen gab es einige Hürden, die hier noch genauer behandelt werden sollen.

Des Weiteren soll außerdem etwas auf die verwendete Software MicroPython in Verbindung mit dem ESP32 gesagt werden. Durch ausgiebige Recherche war bekannt, das diese Software für alle Vorhaben sehr gut gerüstet ist. Besonders die MQTT Funktionen sind sehr einfach zu bedienen. Dennoch gab es einige Probleme mit MicroPython und den verwendeten Bibliotheken. Insbesondere der BuA-Detektor stellte sich als schwierig heraus. Da sich der Mensch quasi zu jeder Millisekunde im Bett bewegen könnte, muss der Bewegungssensor durchgehend ohne Pause ausgelesen werden. Genau hier liegt das Problem, denn das serielle Auslesen der HX711 Bausteine erfordert eine relativ lange Durchlaufzeit. Zum einen werden zur genaueren Mittelwertbildung mehrere Messungen durchgeführt, zum anderen müssen zwei HX711 Bausteine nacheinander ausgelesen werden. Um ein gutes Nutzererlebnis des BuA-Detektors zu gewährleisten ist es wichtig eine instantane Nachricht darüber zu bekommen, ob jemand das Bett belegt oder verlassen hat. Die Voraussetzung dafür ist, dass in jedem Durchlauf der Software die Gewichtsmessung über die HX711 Bausteine ausgeführt wird. Angenommen das Auslesen dauert 500ms für die Gewichtsmessung und 5ms für die Bewegungsmessung, so kann der Bewegungssensor nur einen Bruchteil der Bewegungen aufnehmen die er müsste um ein aussagekräftiges Bild zu erzeugen. Ein Grund für das langsame Auslesen mag die verwendete Bibliothek sein.

Ein anderen Grund ist in MicroPython, das mehr Ressourcen auf dem ESP32 verwendet als gedacht. Besonders das Einschalten der WLAN Funktionen und das Überprüfen auf neue MQTT Nachrichten hat den Gesamtprozess verlangsamt.

Um dieses Problem zu umgehen, sind mehrere Lösungsansätze denkbar. Eine einfache Lösung wäre es einen weiteren ESP32 zu verwenden, der nur dafür benutzt wird die HX711 Bausteine auszulesen und diese Daten über MQTT ins Netzwerk zu senden. Ein etwas aufwändigerer Ansatz ist das Auslesen der Gewichtsdaten aus dem HX711 auszulösen wenn eine Bewegung im Bett über den Bewegungssensor erkannt wurde. Dies erfordert jedoch einen komplexen Algorithmus, der viele Fehlerprüfungen beinhaltet um ein zu häufiges Auslesen zu verhindern. Bewegungen die während des Auslesens stattfinden können nach wie vor nicht berücksichtigt werden. Solange im Bett keine Person liegt, wird kein Bewegungssensor benötigt. So könnten verschiedene Stadien per MQTT an den BuA-Detektor gesendet werden, die bestimmen ob er gerade Bewegungsdaten ausliest oder nur Gewichtsmessungen durchführt. Diese Herangehensweise kann sicherstellen, dass ein Belegen des Bettes sofort erkannt wird. Auch nach dem Abspielen einer Weckmelodie kann davon ausgegangen werden, dass die Aktivität im Bett steigt und der BuA-Detektor somit auf das Verlassen des Bettes vorbereitet werden kann.

Obwohl die Versuche mit den LiDar ToF Sensoren vorerst für Optimismus sorgten, war die Umsetzung einer Gestenerkennung etwas schwieriger als gedacht. Die Anordnung der Sensoren ist durch die Größe des Weckers festgesetzt, der Winkel kann jedoch verstellt werden. Dies half zwar Hindernisse wie Bett und Wände aus dem Sichtfeld zu bekommen, änderte aber nichts am Sichtfeld insgesamt. Zu niedriger oder zu hoher Abstand zum Sensor führte zu erschwerten Bedingungen beim Erkennen von Gesten, was zu einem Fehlverhalten führte. Ein weiterer Faktor war die Abtastrate, die mit dem ständig aktivem WLAN zum Empfangen von MQTT Nachrichten, sankt. Eine schnelle Bewegung der Hand wird dadurch nicht immer richtig als Geste erkannt.

Ein Punkt der positiv empfunden wurde war das Erstellen von Subsystemen. Wecker, BuA-Detektor und Server (Raspberry Pi) konnte unabhängig voneinander getestet und programmiert werden. Auch das Installieren einer eigenen InfluxDB Datenbank war ein Schlüsselpunkt. Denn nur so konnte das Gesamtsystem wirklich unabhängig vom Living-Place getestet werden.

4 Fazit und Ausblick

4.1 Fazit

Das Entwickeln eines Weckmechanismus in einem Smart Home war Thema dieser Arbeit. Dieser Mechanismus besteht aus einem Bewegungs- und Anwesenheitsdetektor, der Daten zum Schlaf bereitstellt. Einen Server der diese verarbeitet sowie einem Wecker, um Gesten zu erkennen, Töne abzuspielen und Uhrzeiten oder Nachrichten anzuzeigen. Der unter Zusammenarbeit dieser drei Bausteine gebildete Weckmechanismus dient nicht nur einem sanften Wecken eines Schlafenden aus der richtigen Schlafphase, siehe Kap. 2.2.3, sondern kann auch Auslöser für weitere Aktionen im Smart Home sein. Um diese Ziele zu erlangen wurden Sensordaten untersucht, Programmcode entwickelt und Bauteile konstruiert.

Nach einer ausgiebigen Recherche und Analyse, war die Auswahl von Hardwarekomponenten ein ebenso relevanter Anteil des Geleisteten. Denn das selbst gesetzte Ziel einen optisch ansprechenden Wecker zu konstruieren, der sich zudem gut für Versuche eignete, trotzdem alle Hardwarekomponenten gut unterbringt, galt es zu bewältigen. Die verwendete Hardware zur Erkennung der Schlafphasen wurde so gewählt, dass sich keine Einschränkungen für den Schlafenden ergaben. Es wurden also keine am Körper zu tragenden Geräte verwendet, um einen hohen Schlafkomfort zu gewährleisten. Zudem galt es die Ästhetik des Raumes nicht zu verändern. Die Hardware wurde also so gestaltet, dass sie möglichst unsichtbar im Raum und am Bett verbaut werden kann. Ausgenommen davon war der Wecker, denn dieser muss vom Bett aus im Sichtbereich stehen, da sonst einerseits die Erkennung von Gesten, aufgrund der Reichweite, nicht erfolgen kann und andererseits die Uhrzeit nicht abgelesen werden kann.

Eine kontaktlose Quittierung des Weckers wurde durch eine Gestenerkennung ermöglicht, siehe Kap. 3.2.2. Dafür wurden zwei LiDar Sensoren verbaut, welche Abstände zu Objekten ermitteln können. Die Sensoren wurden auf dem Subsystem Wecker verbaut, da dieser herkömmlicherweise neben dem Bett steht und somit in Reichweite ist.

Bei der Recherche zum Schlaf des Menschen ist deutlich geworden, dass mit zunehmender Zeit die im Schlaf verbracht wird eine Abnahme der Tiefschlafphasen zu erwarten ist. Die Betrachtung der Sensordaten des BuA-Detektors in 3.2.1 zeigt das diese Annahme aus [39] auf meinen Schlaf anzuwenden ist. In Betrachtung dieser Annahme und bei einem normalen Schlafrhythmus eines Menschen, das heißt er schläft pro Tag die erforderliche Zeit, spielt die Erfassung der Schlafphasen für ein sanftes Wecken keine hoch gewichtete Rolle. Der Anteil des Schlafes, in dem der Mensch sich ungern wecken lässt, wird sich im Laufe des Schlafes verringern. Dies führt dazu, dass der Mechanismus für sanftes Wecken nur selten genutzt wird. Jedoch können die gewonnen Daten über einen langen Zeitraum gespeichert und bewertet werden, woraufhin ein intelligentes System Veränderungen im Schlaf feststellen könnte, um bei einem bedrohlichen Verhalten darüber aufmerksam machen. Dies könnte besonders im Bereich AAL interessant sein, denn schlechter Schlaf ist ein Indikator für eine Veränderung der Gesundheit.

4.2 Ausblick

In folgendem Kapitel wird aufgeführt, wie das Gesamtsystem weiterentwickelt sowie verbessert werden könnte. Außerdem werden Einsatzszenarien besprochen, die über die in der Arbeit besprochenen hinaus gehen könnten.

Besonders bei der Betrachtung der Sensordaten vom BuA-Detektor ist aufgefallen, dass sich die Aktivität im Verlauf des Schlafs erhöht. Dieses Verhalten ist wie in Kap. 2.2.3 beschrieben zwar zu erwarten, erschwert aber das Erkennen von Schlafphasen auf Basis von einem Bewegungssensor. Eine Verbesserung könnte durch das Überwachen der Herzfrequenz erreicht werden. Ein Fitnessarmband mit offener API (z.B. Uhren von Fitbit, siehe [14]) würde diese Möglichkeit bieten. Je mehr Daten während des Schlafes aufgezeichnet werden können, desto besser kann die Bewertung der Schlafphasen funktionieren.

Jeden Tag verschlafen Personen, weil sie vergessen haben ihren Wecker einzustellen. Mithilfe des BuA-Detektors kann beim Betreten des Bettes eine Routine ausgeführt werden, die die Weckzeit aus einem Cloud-Kalender aufruft. Dies setzt voraus, dass die nutzende Person seinen eigenen Kalender immer auf neuestem Stand hält. Falls kein Termin vorhanden ist, könnte der Wecker dem Nutzer eine Nachricht auf dem Display anzeigen und das Einstellen der Weckzeit einfordern.

Der BuA-Detektor ermöglicht eine Bewertung, auf welcher Seite des Bettes eine Person schläft, so könnte man zwei Wecker am Bett positionieren und immer nur einen, den näheren, ansteuern um mit den Nutzer zu interagieren. Auch Lichtszenarien können anhand diese Positionsangaben abgespielt werden um z.B. Leselampen auf der richtigen Seite des Bettes einzuschalten. Auch könnte der Detektor in anderen Sitz- oder Liegemöbeln Anwendung finden. Dies könnte nicht nur einer automatisierten Lichtsteuerung zugute kommen, sondern auch im AAL Kontext helfen. Die Gewichtserfassung des BuA-Detektors könnte dafür genutzt werden, das Gewicht einer Person zu überwachen und bei Auffälligkeiten einen stillen Alarm auslösen.

Literaturverzeichnis

[1] 3 eQ: *HomeMatic - IP*. – URL https://www.homematic-ip.com/. – Zugriffsdatum: 14.09.2020

[2] Aarts, E.: Ambient intelligence: a multimedia perspective. In: *IEEE MultiMedia* 11 (2004), Nr. 1

[3] Anvari-Moghaddam, A. ; Monsef, H. ; Rahimi-Kian, A.: Optimal Smart Home Energy Management Considering Energy Saving and a Comfortable Lifestyle. In: *IEEE Transactions on Smart Grid* 6 (2015), Nr. 1, S. 324–332

[4] Bernin, Arne: *Masterarbeit - Einsatz von 3D-Kameras zur Interpretation vonräumlichen Gesten im Smart Home Kontext*. 2012. – URL https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/bernin.pdf

[5] Blumendorf, Marco ; Feuerstack, Sebastian ; Albayrak, Sahin: Multimodal Smart Home User Interfaces. In: *TU-Berlin* (2008). – URL https://www.researchgate.net/profile/Sebastian_Feuerstack/publication/268338418_Multimodal_Smart_Home_User_Interfaces/links/54ec76ad0cf27fbfd770ba38/Multimodal-Smart-Home-User-Interfaces.pdf

[6] Causer, Mike: *MicroPython MAX7219 8x8 LED Matrix*. – URL https://github.com/mcauser/micropython-max7219. – Zugriffsdatum: 11.08.2020

[7] Coppen, Richard: Message Queuing Telemetry Transport (MQTT) TC. In: *OASIS* (2017). – URL https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt

[8] Dagger, R.: *MicroPython JQ6500 sound module library*. – URL https://github.com/rdagger/micropython-jq6500. – Zugriffsdatum: 11.08.2020

[9] EasyEDA: *EasyEDA - The Next Generation of PCB Design Tool.* – URL https://easyeda.com/login?from=editor. – Zugriffsdatum: 01.09.2020

[10] Eberhardt, Birgid: *AAL-Anwendungsszenarien.* VDE Ambient Assisted Living, 2011. – URL http://partner.vde.com/bmbf-aal/Publikationen/Fachbeitraege/Documents/AAL-Szenarien-final.pdf. – ISBN 978-3-925512-22-3

[11] Eclipse, Mosquitto™: *An open source MQTT broker.* – URL https://mosquitto.org/. – Zugriffsdatum: 16.07.2020

[12] EDISEN SENSOR, SYSTEME: *MT0.6-U.* – URL https://www.edisen.de/level9_cms/download_user/Datenbl%E4tter/MT0.6-U/deutsch/Datenblatt-Universalsensor-MT0.6-U.pdf. – Zugriffsdatum: 14.09.2020

[13] Engineering, HAW Hamburg F. of ; TI, Computer S.: *LivingPlace.* – URL https://livingplace.haw-hamburg.de/. – Zugriffsdatum: 16.09.2020

[14] Fitbit, Inc.: *Fitbit.* – URL https://www.fitbit.com/de/home. – Zugriffsdatum: 01.09.2020

[15] Geisberger, Eva ; (Hrsg.), Manfred B.: agendaCPS - Integrierte Forschungsagenda Cyber-Physical Systems. In: *acatech – Deutsche Akademie der Technikwissenschafte* (2012). – URL https://www.bmbf.de/files/acatech_STUDIE_agendaCPS_Web_20120312_superfinal.pdf

[16] Grafana, Labs: *Grafana Features.* 2020. – URL https://grafana.com/grafana/. – Zugriffsdatum: 20.07.2020

[17] Hardenack, Frank: *Masterarbeit - Das intelligente Bett -Sensorbasierte Detektion von Schlafphasen.* 2011. – URL https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/hardenack.pdf

[18] Home Assistant, Inc.: *Home Assistand.* – URL https://www.home-assistant.io/. – Zugriffsdatum: 20.07.2020

[19] InfluxData, Inc.: *InfluxDB.* – URL https://www.influxdata.com/products/influxdb-overview/. – Zugriffsdatum: 20.07.2020

[20] Jezek, Adam: *MPU6050-ESP8266-MicroPython.* – URL https://github.com/adamjezek98/MPU6050-ESP8266-MicroPython. – Zugriffsdatum: 06.08.2020

[21] Karsten Berns, Mario T.: *Eingebettete Systeme - Systemgrundlagen und Entwicklung eingebetteter Software*. VDE Ambient Assisted Living, 2010. – URL https://link.springer.com/content/pdf/10.1007/978-3-8348-9661-2.pdf. – ISBN 978-3-8348-0422-

[22] Kidd, Cory D. ; Orr, Robert ; Abowd, Gregory D. ; Atkeson, Christopher G. ; Essa, Irfan A.: *The Aware Home: A Living Laboratory for Ubiquitous Computing Research*. 1999. – URL https://doi.org/10.1007/10705432_17

[23] Koelle, Marion: Tangible User Interfaces - Ein kurzer Überblick über Forschungsfeld und Literatur. In: *LMU München - LFE Medieninformatik* (2019). – URL https://www.medien.ifi.lmu.de/lehre/ws1011/mmi2/mmi2_uebungsblatt1_loesung_koelle.pdf

[24] Liechti, Chris: *pySerial*. – URL https://pyserial.readthedocs.io/en/latest/pyserial.html. – Zugriffsdatum: 07.08.2020

[25] Matzner, Robin: *VL53L0X Bibliothek*. – URL https://github.com/uceeatz/VL53L0X. – Zugriffsdatum: 01.09.2020

[26] Michael Pieper, Margherita A. ; Cortés, Ulises: *Introduction to the Special Theme Ambient Assisted Living*. 2011. – URL https://projects.ics.forth.gr/files/publications/antona/2011/Pieper_et_al.pdf

[27] Netgear: *Power over Ethernet*. – URL https://kb.netgear.com/de/209/Was-ist-PoE-Power-over-Ethernet. – Zugriffsdatum: 14.09.2020

[28] OpenJS, Foundation: *NodeRed*. – URL https://nodered.org/. – Zugriffsdatum: 20.07.2020

[29] Pakalski, Ingo: Smarte Wecker im Test: Unter den Blinden ist der Einäugige König. In: *Golem* (2019). – URL https://www.golem.de/news/smarte-wecker-im-test-unter-den-blinden-ist-der-einaeugige-koenig-1908-143178.html

[30] pandas, development t.: *pandas*. – URL https://pandas.pydata.org/docs/. – Zugriffsdatum: 06.08.2020

[31] Parallax, Inc: PING))) Ultrasonic Distance Sensor. In: *Parallax Inc* (2013). – URL https://docs.rs-online.com/ca69/0900766b81382974.pdf

[32] Piskunov, Sergey: *micropython-hx711*. – URL https://github.com/SergeyPiskunov/micropython-hx711. – Zugriffsdatum: 11.08.2020

[33] Plotly: *Plotly*. – URL https://plotly.com/. – Zugriffsdatum: 07.08.2020

[34] Quast, Oliver: *Bachelorarbeit - Berührungslose Schlafphasenerkennung zur Integration in einSmart-Home*. 2012. – URL https://edoc.sub.uni-hamburg.de/haw/volltexte/2013/1957/pdf/Bachelorarbeit.pdf

[35] Rashidi, P. ; Mihailidis, A.: A Survey on Ambient-Assisted Living Tools for Older Adults. In: *IEEE Journal of Biomedical and Health Informatics* 17 (2013), Nr. 3

[36] Sharp: GP2Y0A02YK0F - Distance Measuring Sensor Unit Measuring distance: 20 to 150 cm Analog output type. In: *Sharp* (2006). – URL https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

[37] STMicroelectronics: UM2046 User manual - Getting started with VL53L0X ranging and gesture detection sensorsoftware expansion for STM32Cube. In: *ST* (2016). – URL https://www.st.com/content/ccc/resource/technical/document/user_manual/group0/44/17/0e/ed/f7/51/44/67/DM00285084/files/DM00285084.pdf/jcr:content/translations/en.DM00285084.pdf

[38] STMicroelectronics: VL53L0X - This is information on a product in full production. April 2018DocID029104 Rev 21/10VL53L0XWorld's smallest Time-of-Flight ranging and gesture detection sensor - Datasheet. In: *ST* (2018). – URL https://www.st.com/resource/en/datasheet/vl53l0x.pdf

[39] Tatjana Crönlein, Peter Y.: *Schlafmedizin 1×1 : Praxisorientiertes Basiswissen / von Tatjana Crönlein, Wolfgang Galetke, Peter Young*. Springer Verlag, 2017. – URL https://link.springer.com/book/10.1007%2F978-3-662-49789-0. – ISBN 978-3-662-49789-0

[40] The Matplotlib, development t.: *Matplotlib - Visualization with Python*. – URL https://matplotlib.org/. – Zugriffsdatum: 07.08.2020

[41] United Nations, Department of E. ; Social Affairs, Population Division (.: *World Population Ageing 2013. ST/ESA/SER.A/348*. URL https://www.un.org/en/development/desa/population/publications/pdf/ageing/WorldPopulationAgeing2013.pdf, 2011

[42] Weiser, Mark: Some Compter Science Issues in Ubiquitous Computing. In: *Communications of the ACM* (1993). – URL https://dl.acm.org/doi/pdf/10.1145/159544.159617

[43] Wrede, Matthias: *Bachelorarbeit - Dezentrale Echtzeitdatenverarbeitung inheterogenen Systemen.* 2019. – URL https://www-ai.cs.tu-dortmund.de/PublicPublicationFiles/wrede_2019a.pdf

A Anhang

A.1 Mögliche Lösungsansätze

Anforderung:	Weckzeit einstellbar	Nr.:	1
Beschreibung:	Die Weckzeit muss einstellbar sein		
Lösungsansätze:	• Tasten am Wecker (Button mechanisch, Touchdisplay) • Smartphone Applikation • Abfrage an einen Cloud Kalender • Spracheingabe (Alexa, Cortana, usw)		

Tabelle A.1: Anforderung: Weckzeit einstellbar

Anforderung:	Kontaktlose Quittierung	Nr.:	2
Beschreibung:	Der Wecker muss kontaktlos zu quittieren sein.		
Lösungsansätze:	• Zwei optische Sensoren, um eine ein-dimensionale Geste mithilfe von Software erkennen zu können – Infrarot Sensoren – LiDar Sensoren – Ultraschall Sensoren • Eine Kamera mit Bilderkennungssoftware welche Gesten erkennt • Kapazitiver Sensor der auch ohne Berührung reagiert • Spracheingabe		

Tabelle A.2: Anforderung: Kontaktlose Quittierung

Anforderung:	Anwesenheit im Bett erkennen	Nr.:	3
Beschreibung:	Das Gesamtsystem muss die Möglichkeit bieten, die Anwesenheit einer Person im Bett zu erkennen.		
Lösungsansätze:	• Eine Kamera mit Bilderkennungssoftware welche Personen im Bett erkennt • Wärmebildkamera ausgerichtet auf das Bett mit Auswertungssoftware • Wäge Zellen verbaut im oder unter dem Bett • Druck Sensoren in verschiedenen Ausführungen		

Tabelle A.3: Anforderung: Anwesenheit im Bett erkennen

Anforderung:	Bewegungen im Bett erkennen	Nr.:	4
Beschreibung:	Der Wecker muss die Möglichkeit bieten, Bewegungen im Bett zu erkennen.		
Lösungsansätze:	• Kapazitiver Sensor am Lattenrost um Verformung dieser zu erkennen [17, Hardenack 2012] • Dehnmessstreifen am Lattenrost um Verformung dieser zu erkennen [34] • Anbindung an eine Smartwatch mit Bewegungssensor über den SmartHome Server • Fest verbauter Bewegungssensor am Lattenrost des Bettes • Auswertung von Drucksensoren die im oder unter dem Bett verbaut sind		

Tabelle A.4: Anforderung: Bewegungen im Bett erkennen

Anforderung:	Kommunikation mit Smart Home	Nr.:	5
Beschreibung:	Der Wecker muss mit einem Smart Home Server kommunizieren können		
Lösungsansätze:	• Funk-Verbindung (WLAN, LoRaWAN, ZigBee) • Kabel-Verbindung (Ethernet) • Kommunikation über SmartHome spezifische Protokolle		

Tabelle A.5: Anforderung: Kommunikation mit Smart Home

Anforderung:	Display	Nr.:	6
Beschreibung:	Der Wecker muss über ein Display verfügen, um Uhrzeit oder anderen Text anzuzeigen.		
Lösungsansätze:	• LED-Anzeigen – 7-Segment Anzeige – Dotmatrix-Anezige • LCD-Anzeigen • LCD-Display • RGB-LED-Display / Matrix • Mini LED-Bildprojektor • Smartphone		

Tabelle A.6: Anforderung: Display

Anforderung:	Audiosausgabe	Nr.:	7
Beschreibung:	Der Wecker muss Töne abspielen können. z.B. ein Weck-Ton		
Lösungsansätze:	• Diese Aufgabe kann von einem Smart Home Software, wie z.B. NodeRed übernommen werden. Diese würde dann eine Audio Datei auf einem WLAN- oder Bluetooth Lautsprecher abspielen • Soundmodul mit integrierter Verstärker mit zusätzlichem Lautsprecher im Wecker		

Tabelle A.7: Anforderung: Audioausgabe

Anforderung:	Schlafphasen erkennen	Nr.:	8
Beschreibung:	Es muss eine Möglichkeit geben, die in Anforderung 4 erkannten Bewegungen, in Schlafphasen zu überführen		
Lösungsansätze:	• Ansätze mit künstlicher Intelligenz, um die Daten von den in #4 genannten Sensoren auszuwerten und in Schlafphasen zu überführen • Algorithmischer Ansatz zur Auswertung von Sensordaten aus #4		

Tabelle A.8: Anforderung: Schlafphasen erkennen

Anforderung:	Sanftes Wecken	Nr.:	9
Beschreibung:	Mithilfe der in #8 erkannten Schlafphasen soll der Nutzer, wenn möglich in einer Leichtschlafphase geweckt werden		
Lösungsansätze:	• Programmieren einer festen Weckroutine auf Basis der eingestellten Weckzeit und der aktuellen Schlafphase im Wecker • Dezentrales programmieren einer Weckroutine in einer SmartHome Umgebung mit Einstellmöglichkeiten für den Endnutzer		

Tabelle A.9: Anforderung: Sanftes Wecken

Anforderung:	Morgenroutine	Nr.:	10
Beschreibung:	Der Wecker muss mithilfe der in #5 genannten Smart Home Kommunikation eine Morgenroutine ausführen wie z.B.: • t-5min -> Fenster öffnen für frische Luft • t-2min -> Licht langsam an dimmen • t -> Wecker klingelt, Kaffeemaschine anmachen • t+3min -> Rollläden öffnen		
Lösungsansätze:	• Programmierung einer Morgenroutine in einer Event basierten Programmierumgebung • Programmierung in Python, Java oder C		

Tabelle A.10: Anforderung: Morgenroutine

A.2 Versuche BuA-Detektor

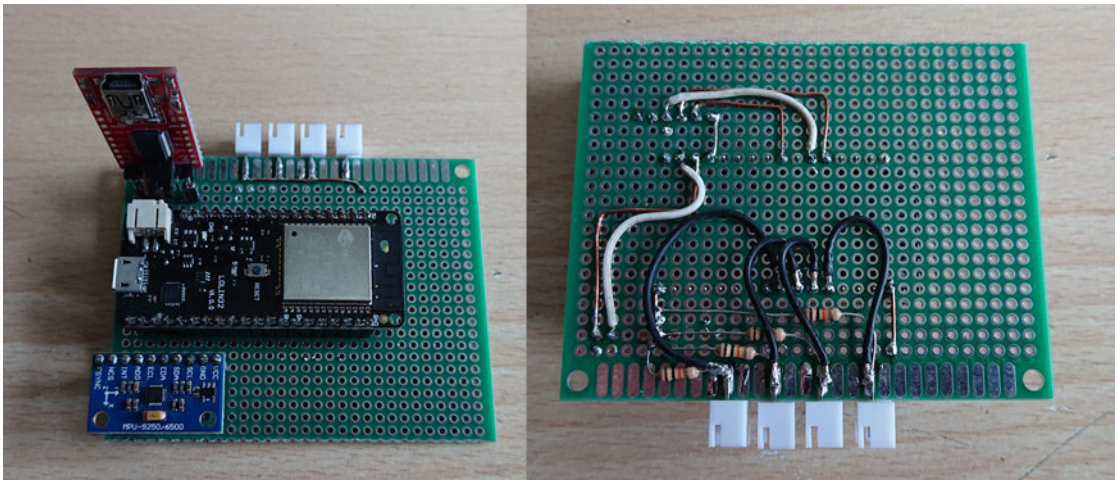

Abbildung A.1: Erster Prototyp des BuA Detektors

A.3 Schaltpläne

Schaltpläne in voller Größe sind zusätzlich als PDF im digitalen Anhang der Bachelorarbeit zu finden. Zu finden sind diese im Ordner SSchaltpläne"

A.3.1 Schaltplan Wecker

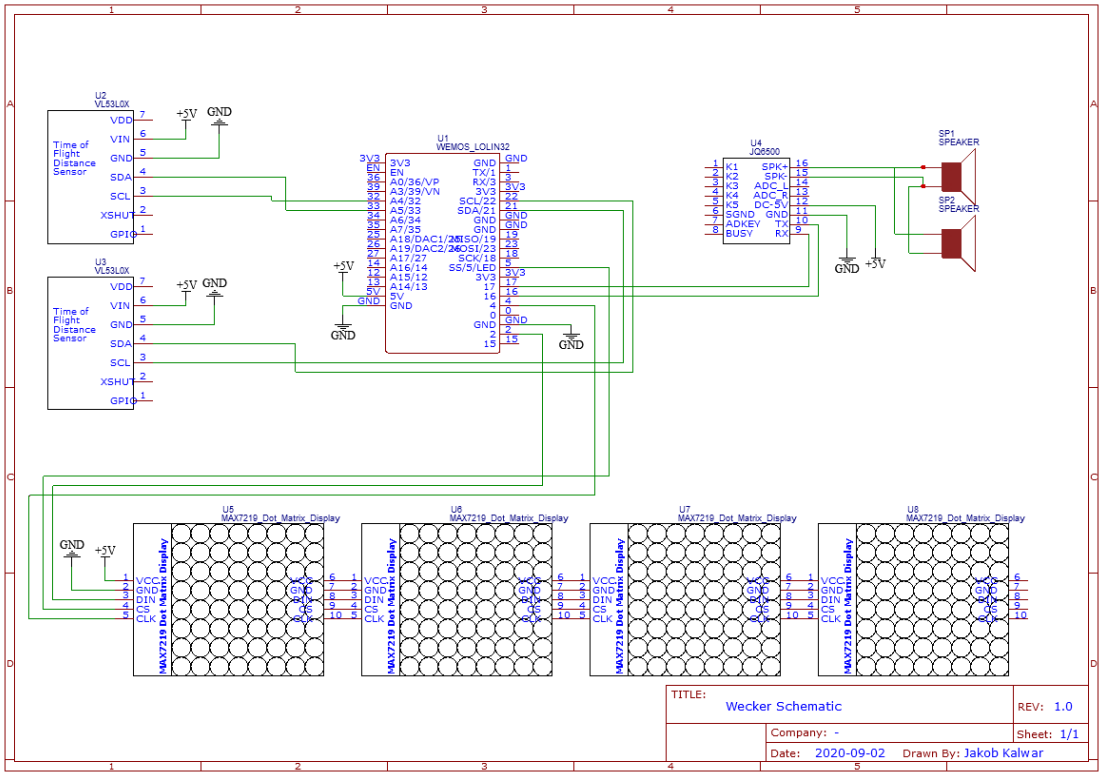

Abbildung A.2: Schaltplan Wecker

A.3.2 Schaltplan BuA-Detektor

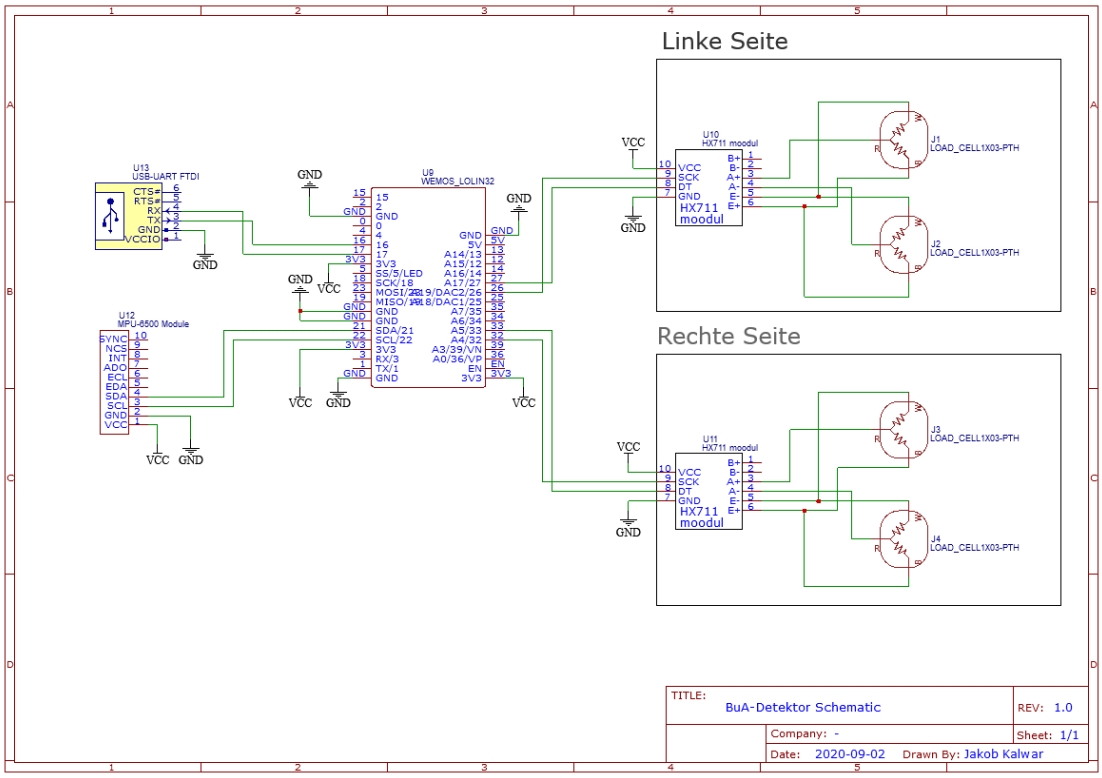

Abbildung A.3: Schaltplan BuA-Detektor

A.4 Weckmechanismus

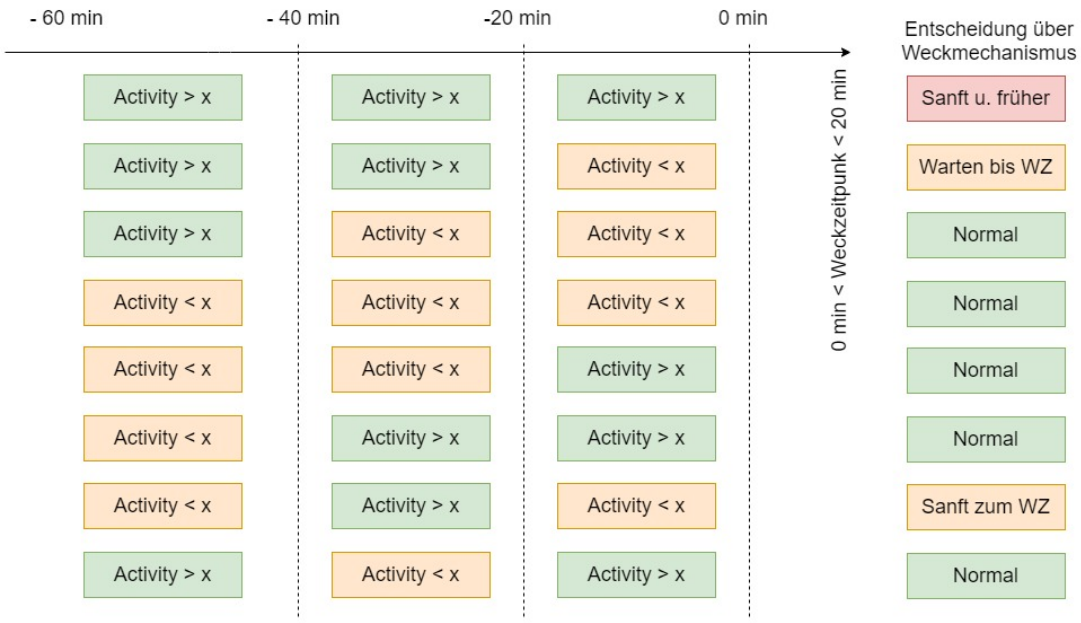

Abbildung A.4: Weckmechanismus Schaubild

A.5 Herstellung des Weckers

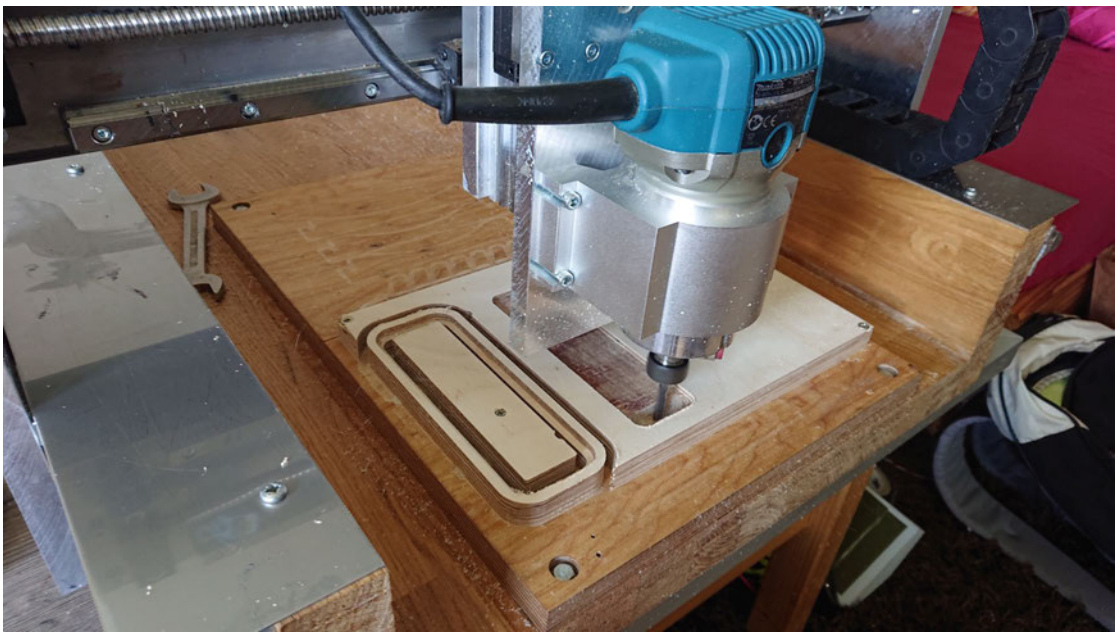

Abbildung A.5: Fertigung einer Sektion des Weckers

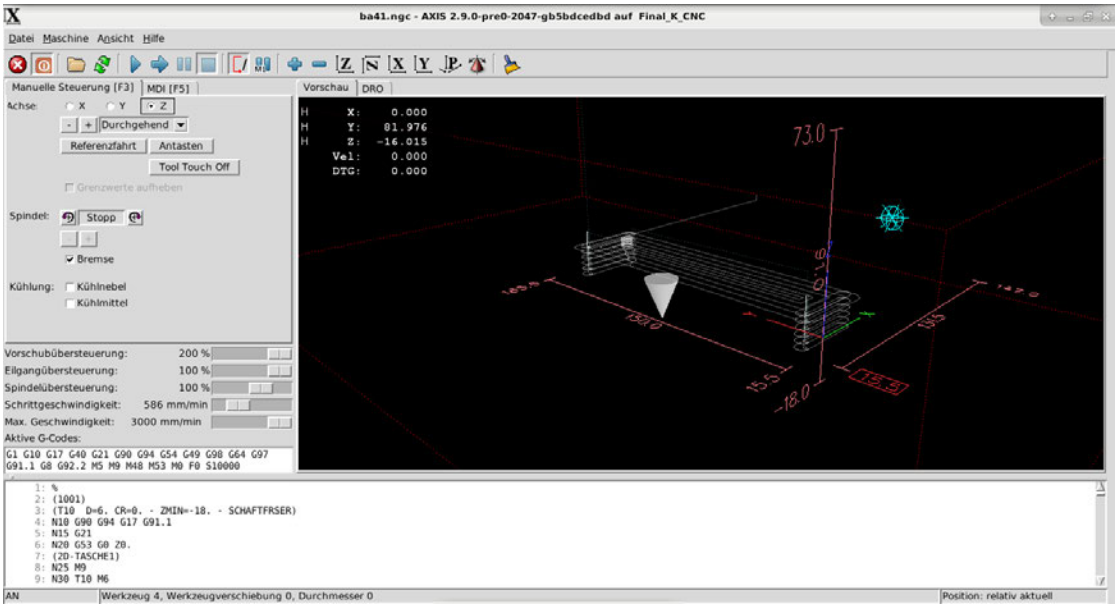

Abbildung A.6: LinuxCNC Screenshot

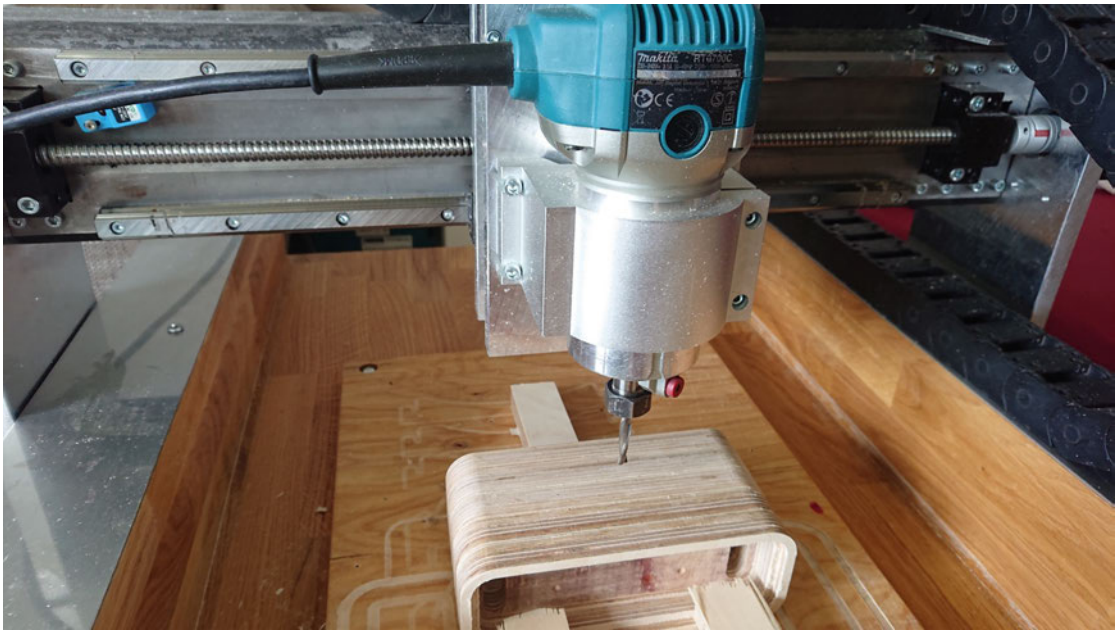

Abbildung A.7: Weitere Aufspannung des Weckers

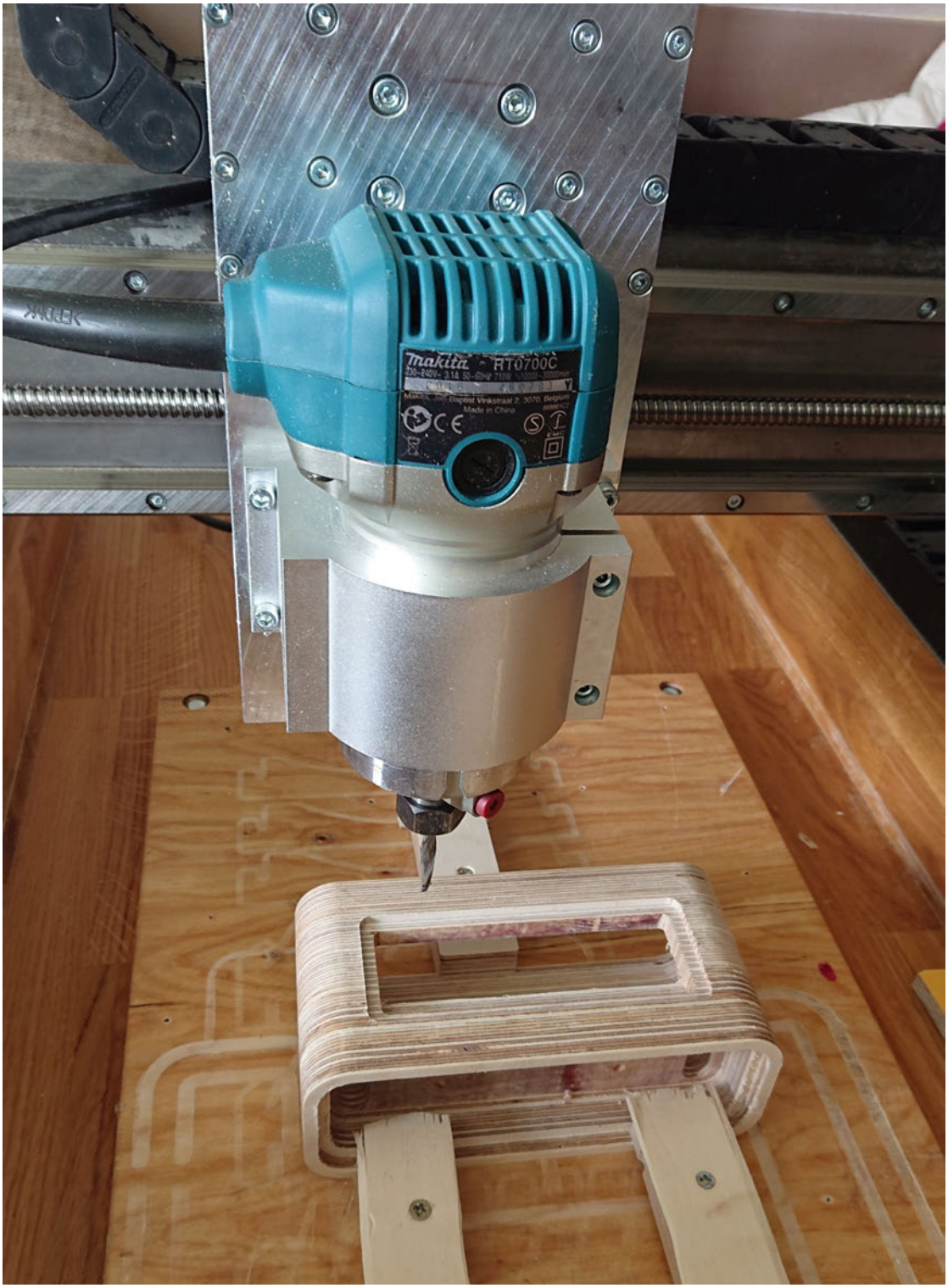

Abbildung A.8: Ausschnitt für Deckel

A.6 Evaluation

Hier wird die gezeigt wie die Anforderungen umgesetzt wurden und wie das Ergebnis ist.

Anforderung:	Weckzeit einstellbar	Nr.:	1
Beschreibung:	Die Weckzeit muss einstellbar sein		
Lösung:	Smartphone Applikation		
Ergebnis:	Über einen Browser kann eine Seite zum einstellen der Weckzeiten geöffnet werden		

Tabelle A.11: Evaluation: Weckzeit einstellbar

Anforderung:	Kontaktlose Quittierung	Nr.:	2
Beschreibung:	Der Wecker muss kontaktlos zu quittieren sein.		
Lösung:	Zwei Optische LiDar Sensoren mit ToF Prinzip. Eigene MicroPython Bibliothek.		
Ergebnis:	Die Sensoren können mithilfe der MicroPython Bibliothek 2 verschiedene Gesten zu erkennen. Das Wischen mit der Hand von Links nach Rechts am Wecker vorbei sowie das Wischen von Rechts nach Links.		

Tabelle A.12: Evaluation: Kontaktlose Quittierung

Anforderung:	Anwesenheit im Bett erkennen	Nr.:	3
Beschreibung:	Der Wecker muss die Möglichkeit bieten, die Anwesenheit einer Person im Bett zu erkennen.		
Lösung:	Wäge-Zellen		
Ergebnis:	Wäge-Zellen erfassen die Belastung die auf das Bett ausgeübt wird. NodeRed kann diesen Anstieg erfassen und beurteilen ob jemand im Bett liegt. Eine 100 Prozentige aussage kann dies jedoch nicht Treffen, da auch andere schwere Gegenstände auf ein Bett gelegt werden könnten. Hier ist es erforderlich eine gewisse Schwelle zu setzen.		

Tabelle A.13: Evaluation: Anwesenheit im Bett erkennen

Anforderung:	Bewegungen im Bett erkennen	Nr.:	4
Beschreibung:	Der Wecker muss die Möglichkeit bieten, Bewegungen im Bett zu erkennen.		
Lösung:	Fest verbauter Bewegungssensor am Lattenrost des Bettes		
Ergebnis:	Der BuA-Detektor erfasst Bewegungsdaten und übermittelt diese an das MQTT Netzwerk.		

Tabelle A.14: Evaluation: Bewegungen im Bett erkennen

Anforderung:	Kommunikation mit Smart Home	Nr.:	5
Beschreibung:	Der Wecker muss mit einem Smart Home Server kommunizieren können		
Lösung:	Funkverbindung: WLAN		
Ergebnis:	Alle verwendeten und konstruierten Geräte verfügen über WLAN und können so miteinander kommunizieren.		

Tabelle A.15: Evaluation: Kommunikation mit Smart Home

Anforderung:	Display	Nr.:	6
Beschreibung:	Der Wecker muss über ein Display verfügen, um Uhrzeit oder anderen Text anzuzeigen.		
Lösung:	LED Dotmatrix Anzeige		
Ergebnis:	Die verwendete Anzeige kann Lauftexte sowie Uhrzeiten wiedergeben. Kleine Symbole sind auch vorstellbar, müssen aber von Hand programmiert werden. Zum Beispiel kleine Symbole für das bevorstehende Wetter.		

Tabelle A.16: Evaluation: Display

Anforderung:	Audiosausgabe	Nr.:	7
Beschreibung:	Der Wecker muss Töne abspielen können. z.B. ein Weck-Ton		
Lösung:	Soundmodul und Lautsrpecher		
Ergebnis:	Der Wecker kann verschiedene Töne abspielen. Über MQTT stellt der Wecker ein Topic bereit, dass für das Abspielen von kleinen Audiodateien dient. Die Payload muss eine Zahl zwischen 1 und 4 enthalten.		

Tabelle A.17: Evaluation: Audioausgabe

Anforderung:	Schlafphasen erkennen	Nr.:	8
Beschreibung:	Es muss eine Möglichkeit geben, die in Anforderung 4 erkannten Bewegungen, in Schlafphasen zu überführen		
Lösung:	Algorithmischer Ansatz		
Ergebnis:	Mithilfe von NodeRed werden die letzten übertrageenen Daten vom BuA-Detektor ausgewertet und auf Basis dessen die Schlafphase bestimmt. Diese Bestimmung der Schlafphasen ist aufgrund des algorithmischen Ansatzes nicht immer genau. Des weiteren wäre für eine genaue Bestimmung der Schlafphasen mehr Daten notwendig. Das Messen und übermitteln der Herzfrequenz z.B. mit einem Fitnessarmband könnte hier schon einen großen Unterschied machen.		

Tabelle A.18: Evaluation: Schlafphasen erkennen

Anforderung:	Sanftes Wecken	Nr.:	9
Beschreibung:	Mithilfe der in #8 erkannten Schlafphasen soll der Nutzer, wenn möglich in einer Leichtschlafphase geweckt werden		
Lösung:	Dezentrale Programmierung		
Ergebnis:	In NodeRed wird entschieden ob beim derzeitigen Schlafzustand ein Sanftes Wecken gefordert ist. Wenn gefordert wird eine Routine Abgespielt die versucht den Nutzer sanft zu Wecken.		

Tabelle A.19: Evaluation: Sanftes Wecken

Anforderung:	Morgenroutine	Nr.:	10
Beschreibung:	Der Wecker muss mithilfe der in #5 genannten Smart Home Kommunikation eine Morgenroutine ausführen wie z.B.: • t-5min -> Fenster öffnen für frische Luft • t-2min -> Licht langsam an dimmen • t -> Wecker klingelt, Kaffeemaschine anmachen • t+3min -> Rollläden öffnen		
Lösung:	Programmierung einer Morgenroutine in einer Event basierten Programmierumgebung		
Ergebnis:	In NodeRed wurde eine Morgenroutine Programmiert		

Tabelle A.20: Evaluation: Morgenroutine

Glossar

Breadboard Ein Breadboard in Deutsch Steckbrett, ist Entwicklungshilfe die es ermöglicht elektrische Komponenten auf schnelle Art und Weise miteinander zu Verbinden. Diese Breadboards verringern den entwicklungsaufwand in der Prototypen Phasen..

Breakoutboard Ein Breakoutboard ist eine Kleine Platine auf der ein Chipsatz mit Perepherie ausgestattet ist um den Einsatz an einem Mikrocontroller zu vereinfachen.

BuA-Detektor Der Bewegungs- und Anwesenheitsdetektor ist ein Produkt welches während dieser Arbeit entwickelt wurde. Es ist für die Erfassung und Übermittlung von Bewegungs und Gewichtsdaten. Mithilfe dieser Daten kann die Anwesenheit auf einem Liege oder Sitzmöbel erkannt werden und Bewegungen Klassifiziert weden..

FTDI Adapter Ein Gerät zum Anschluss an einen USB-Port des Computers zur Bereitstellung einer UART Schnittstelle..

HAW Hamburg Die HAW Hamburg ist die vormalige Fachhochschule am Berliner Tor.

Tablet Ein kleiner Computer, der ohne Tastatur und Maus bedient werden kann. Ähnlich eines Smartphones kann er von seinem Touchdisplay aus bedient werden. Die Displaygröße liegt meist zwischen 8 und 12 Zoll..

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Konstruktion eines intelligenten Weckmechanismus und dessen Integration in ein Smart Home

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Bachelorarbeit ist erfolgt durch:

Ort	Datum	Unterschrift im Original