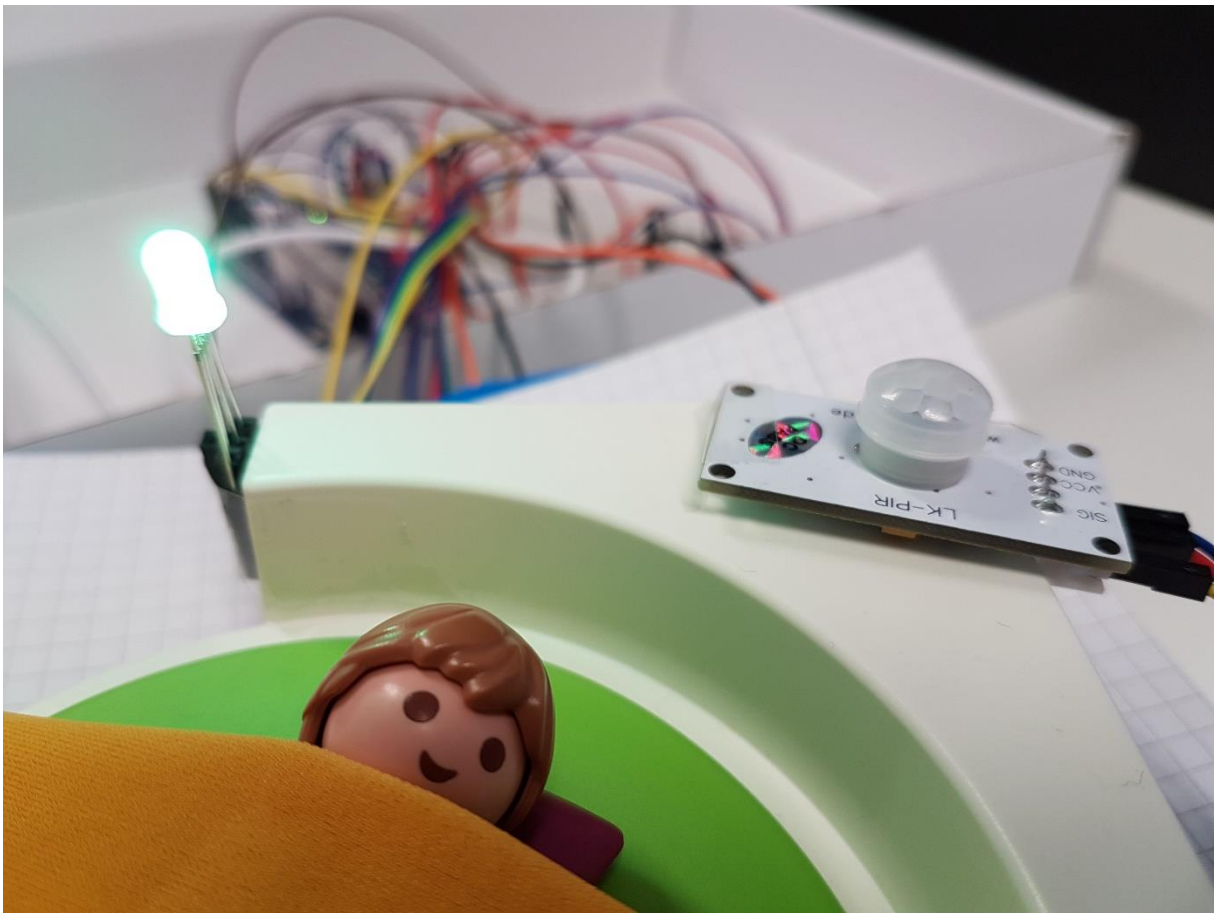
WPF „INTERNET OF THINGS (IOT)“ IM SOMMERSEMESTER 2016

PROJEKTBERICHT
BRIGHTWAKE – DER SMARTE WECKER

Autoren:
Sören Jesse-Grün, 11107639
Christopher Mainzer, 11104792

Dozent:
Prof. Dr. Matthias Böhmer



INHALT

EINLEITUNG

Viele Menschen haben heutzutage Probleme morgens aufzustehen. Sie sind genervt von ihrem Wecker der täglich die gleiche Laier abspielt. Unterschiedliche Arbeitszeiten (Früh- oder Spätschicht) bringen den natürlichen Schlafrhythmus durcheinander. Man ist morgens nach dem Wecken müde oder sogar schlecht gelaunt, obwohl man genug geschlafen hat. Dies liegt daran, dass man ggf. direkt aus einer Tiefschlafphase „gerissen“ wird. Dies ist sogar auf Dauer gesundheitsschädlich.

Einen ersten Ansatz bilden hier sogenannte Tageslichtwecker. Jedoch sind diese recht unflexibel, wenn man zum Zeitpunkt, wo dieser beginnt zu leuchten, schon fast wach ist. Man wacht direkt auf und liegt wach, bis man aufstehen muss.

Es wäre wünschenswert, wenn der Wecker sich nach dem Nutzer und seinem Schlaf richten würde und nicht umgekehrt. Der Nutzer sollte so geweckt werden, wie er es möchte.

Hier kommt unsere Idee ins Spiel.

DIE IDEE

BrightWake ist ein smarter Wecker, der sich nach dem Nutzer richtet. Er registriert die Bewegungen des Nutzers und wertet diese aus. In Abhängigkeit von den gemessenen Werten wird automatisch ein bestimmter, zuvor vom Nutzer definierter Weckmodus gestartet. Hierbei gibt es drei Modi: den Lichtmodus, den Soundmodus und den Vibrationsmodus.

BrightWake wird über ein Web-Interface, einen Bewegungsmelder und einen Taster bedient und eingestellt. Dabei ist es dem Nutzer möglich, individuelle Wecker für jeden Tag der Woche zu stellen. Des Weiteren kann der Nutzer jeder Schlafphase einen Modus zuordnen.

Durch die Bewegungen im Schlaf lassen sich bestimmte Profile ermitteln: Ist der Nutzer vor dem Weckvorgang im Tiefschlaf bewegt er sich kaum bis gar nicht. Ist der Nutzer kurz vor dem Weckvorgang im Leichtschlaf, bewegt er sich mehr.

Eine weitere innovative Idee von BrightWake beinhaltet die intelligente „snooze“-Funktion. Diese wird über eine Handbewegung aktiviert bzw. der Weckvorgang wird so pausiert. So muss der Nutzer sich nicht physisch zum Wecker bewegen, um den Alarm zu „snoozen“, sondern kann diesen einfach mit einer Handbewegung deaktivieren. Dies gilt sinnvollerweise nicht für das endgültige Ausstellen des Alarms, welches per Knopfdruck geschieht.

BrightWake zeichnet sich ebenfalls durch eine Simulation des natürlichen Sonnenaufgangs aus, um dem Nutzer auf sanfte Weise aus dem Schlaf zu holen, bevor der eigentliche Weckvorgang beginnt.

Die zuvor beschriebene künstliche Intelligenz von BrightWake ist ein Merkmal dafür, dass es sich hierbei um ein innovatives Produkt handelt. Bisher gibt es zwar Lichtwecker und die üblichen Wecker, die über einen Weckton wecken, jedoch nicht in einem „Komplettpaket“, welches alle Vorteile mitbringt und die Nachteile mit Hilfe einer künstlichen Intelligenz eliminiert.

Es gibt bis dato keinen Wecker, dass die gleichen Merkmale wie BrightWake aufweist und sich nach dem Schlaf des Nutzers richtet. Somit stellt BrightWake eine innovative Idee dar, die zudem noch neu und nicht käuflich zu erwerben ist.

WESENTLICHE MERKMALE IM ÜBERBLICK

- Einstellung per Web-Interface
- Individueller Wecker für jeden Tag
- Verschiedene individuelle Weckmodi
- Weckmodus wird je nach Schlafphase gewählt
- Bewegungen im Schlaf werden über Sensor überwacht
- Pausieren („snoozen“) des Alarms per Handbewegung, endgültiges Ausstellen per Knopfdruck
- Simulation des Sonnenaufgangs vor dem eigentlichen Wecken

Schlafmodi

Im Folgenden werden die zu Grunde liegenden Schlafmodi definiert. Diese sind nicht komplett wissenschaftlich fundiert, sprich sind nicht nach aktuellem Stand der Schlafwissenschaft erstellt worden. Sie dienen hauptsächlich der groben Unterscheidung des Schlafs des Nutzers.

Leichtschlaf
Der Nutzer bewegt sich viel und ist entweder bereits wach oder sein Aufwachen steht kurz bevor.

Normalschlaf
Der Nutzer bewegt sich hin und wieder. Im Grunde befindet er sich in einer normalen Schlafphase, in der er leicht geweckt werden kann.

Tiefschlaf
Der Nutzer bewegt sich kaum bis gar nicht im Schlaf. Er ist also sehr schwer zu Wecken und muss daher je nach Belieben entweder „hart“ oder sanft geweckt werden.

Weckmodi

Die folgenden Weckmodi wurden in Hinsicht auf möglichst interessante bzw. übliche Weckmethoden erstellt. Diese sind selbstverständlich mit entsprechenden Aktoren und der nötigen Logik erweiterbar. Beispiele dafür befinden sich im Abschnitt „Mögliche Erweiterungen des Prototyps / Umsetzung eines marktreifen Produkts“.

Lichtmodus
Dieser Modus imitiert den natürlichen Sonnenaufgang am Morgen. Der Farbverlauf geht hierbei von einem dunklen Rot über Orange bis hin zu einem hellen Sonnengelb. Er wird eine bestimmte Zeit vor dem eigentlichen Weckzeitpunkt ausgeführt. Somit wird der Nutzer auf eine angenehme und natürliche Methode langsam aus dem Schlaf geholt. Anwendung könnte dieser Modus beim Tiefschlaf finden.

Soundmodus
Dies ist die übliche Methode geweckt zu werden. Der Wecker spielt hierbei eine vorher vom Nutzer definierte Melodie zum Weckzeitpunkt ab.

Vibrationsmodus
Der Vibrationsmodus stellt einen recht „groben“ Weckmodus dar. Hierbei wird der Nutzer aus dem Schlaf „gerüttelt“. In unserem Prototyp wird dies durch einen Vibrationsmotor realisiert. Anwendung könnte dieser Modus bei Nutzern finden, die nur schwer geweckt werden können.

HARDWARE DES PROTOTYPEN

BrightWake verfügt im Wesentlichen über zwei Einheiten: Das Hauptmodul, auf dem das Webinterface erreichbar ist und das Haupt-Skript läuft, welches mit der zweiten Einheit, dem Sekundär-Modul interagiert. In diesem Modul sind die Sensoren und Aktoren integriert, worüber u. a. die Schlafdaten ermittelt werden.

Sehen wir uns zunächst die verwendeten Hardware-Komponenten im Überblick an.

Sensoren

- Bewegungssensor zur Aktivierung der „snooze“-Funktion
- Gyroskop zur Ermittlung der Bewegungen im Schlaf

Aktoren

- RGB-LED zur Simulation des Sonnenaufgangs
- Vibrationsmotor (Shake-Funktion)
- LCD-Anzeige für Uhrzeit und Datum
- Lautsprecher zur Weckton-Ausgabe
- Taster zum Ausstellen des Alarms

Entwicklerboard

- *Hauptmodul:* Raspberry Pi 3
- *Sekundär-Modul:* Arduino Yun

Modul-Komponenten-Zuordnung

Hauptmodul

- Raspberry Pi 3
- RGB-LED
- LCD-Anzeige
- Bewegungssensor
- Lautsprecher

Sekundär-Modul

- Arduino Yun
- Gyroskop
- Vibrationsmotor
- Taster

Sehen wir uns nun die verwendeten Softwarekomponenten an, welche die Hardware miteinander interagieren lassen und „zum Leben erwecken“.

SOFTWAREKOMPONENTEN

- Betriebssystem des Hauptmoduls: Raspbian (Linux-Distribution, basierend auf Debian, angepasst auf die Hardware des Raspberry Pi)
- HTML und CSS für das Userinterface
- PHP und mysqli für die Datenbank-Zugriffe des Userinterfaces
- PhpMyAdmin zum Bereitstellen des Userinterfaces
- MySQL-Datenbank zur Speicherung der Weckzeiten und -einstellungen
- NodeJS mit Johnny Five, mysql, raspi-io und node-aplay

AUSWAHL DER KOMPONENTEN UND TECHNOLOGIEN

Mit welcher Begründung wurden die einzelnen Komponenten für den Bau des Prototyps ausgewählt?

Die Idee „BrightWake“ stellt bestimmte Anforderungen an die Hard- und Software, mit der sie umgesetzt werden soll. Beispielsweise sollte der Prototyp in der Lage sein Ton abzuspielen, und zwar nicht den eines einfachen Summers, den man sehr einfach über einen digitalen Pin ansprechen würde. Sondern es sollen Audiodateien abgespielt werden können. Die Hardware und das System sollten also in der Lage sein, solche Dateien abspielen zu können und die Tonausgabe über einen Lautsprecher zu ermöglichen.

Des Weiteren ist es für die Idee von zentraler Bedeutung, mit Zeit arbeiten zu können. Die Hard- und Software sollte also in der Lage sein, bestimmte Aktionen oder Prozesse zeitgesteuert ausführen zu können.

Somit bietet sich für die Haupteinheit ein vollwertiger Rechner mit Betriebssystem an. Die Wahl fiel hier auf einen Raspberry Pi 3, ein günstiger Einplatinencomputer, der sich ideal für IoT-Projekte eignet. Es gibt mittlerweile eine Menge Konkurrenzprodukte zum Raspberry Pi, die teils mit leistungsstärkeren Prozessoren, mehr Arbeitsspeicher und schnelleren Schnittstellen zur Datenübertragung (Gbit-LAN, S-ATA, USB 3.0, …) ausgestattet sind, dementsprechend aber auch teilweise deutlich teurer sind, als der Raspberry Pi 3.

Die Haupteinheit sollte auf dem Nachttisch des Nutzers stehen und die Uhrzeit anzeigen, wie ein herkömmlicher Wecker. Das ist zumindest das, was der Nutzer von außen sieht – im Innern der Haupteinheit wird die gesamte Programmlogik von BrightWake ausgeführt. Ebenso wird hier der Webserver ausgeführt, der das Webinterface verfügbar macht. Im Falle eines Weckalarms wird über diese Einheit auch der Ton ausgegeben.

Zusätzlich zur Haupteinheit benötigt BrightWake noch zweites Modul, welches die Werte der Sensoren an die Haupteinheit übermittelt und die Aktoren direkt ansteuert, wenn die Hauptlogik des Hauptmoduls die Anweisung dazu erteilt. Theoretisch wäre das Dank der GPIO-Schnittstelle vermutlich alles alleine mit dem Raspberry Pi möglich – die Idee hinter BrightWake ist aber, dass dieses Hilfsmodul direkt am Bett des Nutzers angebracht ist (sowie die Sensoren/Aktoren). Die drahtlose Kommunikation der beiden Einheiten über WiFi soll BrightWake physikalisch gesehen eine gewisse Flexibilität verleihen.

Die konkrete Excel-Tabelle mit den verwendeten Komponenten sowie deren Preise und Bezugsquellen ist in unserem GitHub-Repository (Link s. u.) zu finden.

UMSETZUNG DES PROJEKTS

Ablauf der Umsetzung

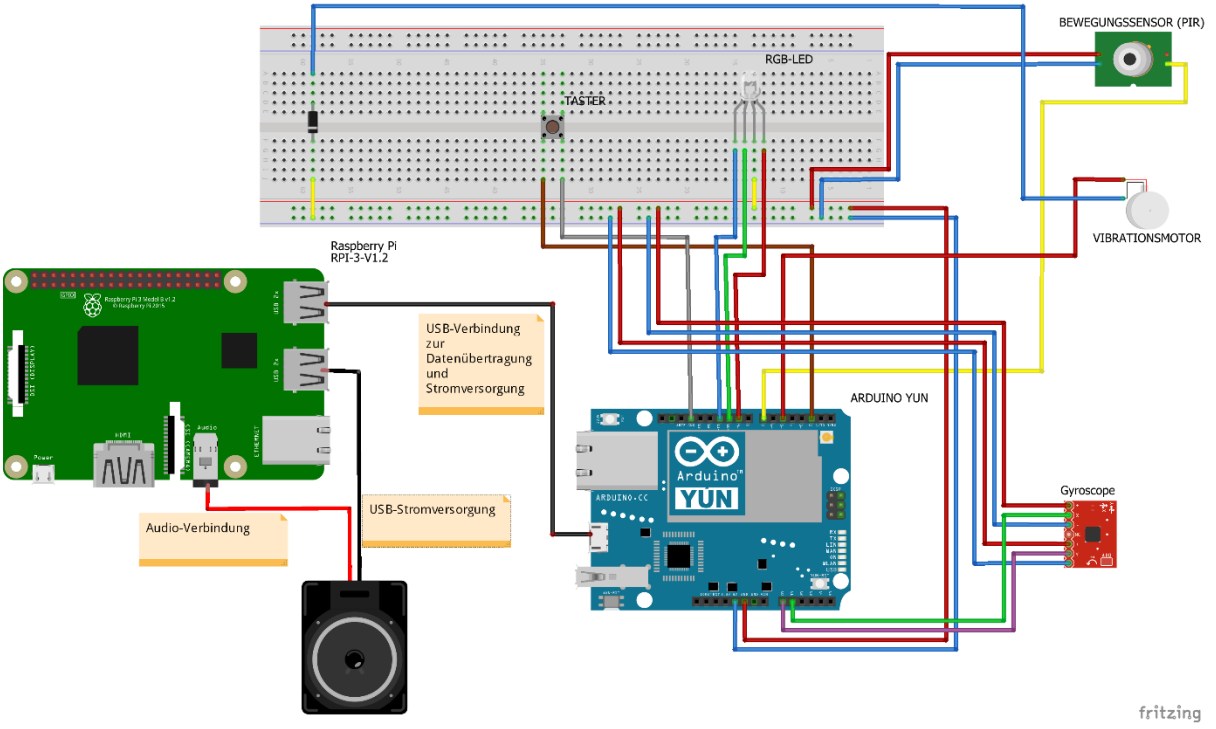
Nachdem die Komponenten ausgewählt, bestellt und geliefert wurden, haben wir zunächst damit begonnen, die Haupt-Einheit, also den Raspberry Pi, einzurichten und entsprechend unseren Anforderungen zu konfigurieren. Zum einen die Grundkonfiguration, wie Root- und Nutzerpasswort, Tastaturlayout, Sprache, etc., zum anderen das Installieren zusätzlicher Software, wie zum Beispiel einen Webserver und node-js.

Während der Bearbeitung des Projekts wurde das ursprüngliche Konzept etwas geändert, was zum einen darauf zurückzuführen ist, dass Hardware aufgrund von Inkompatibilitäten ausgetauscht werden musste, und zum anderen, da sich das erste Konzept als zu unflexibel erwies. Beispielsweise sollte die RGB-LED, der Bewegungsmelder und der Taster direkt an das Hauptmodul angeschlossen werden. Diese Komponenten kommunizieren nun allerdings alle direkt mit dem Zusatzmodul.

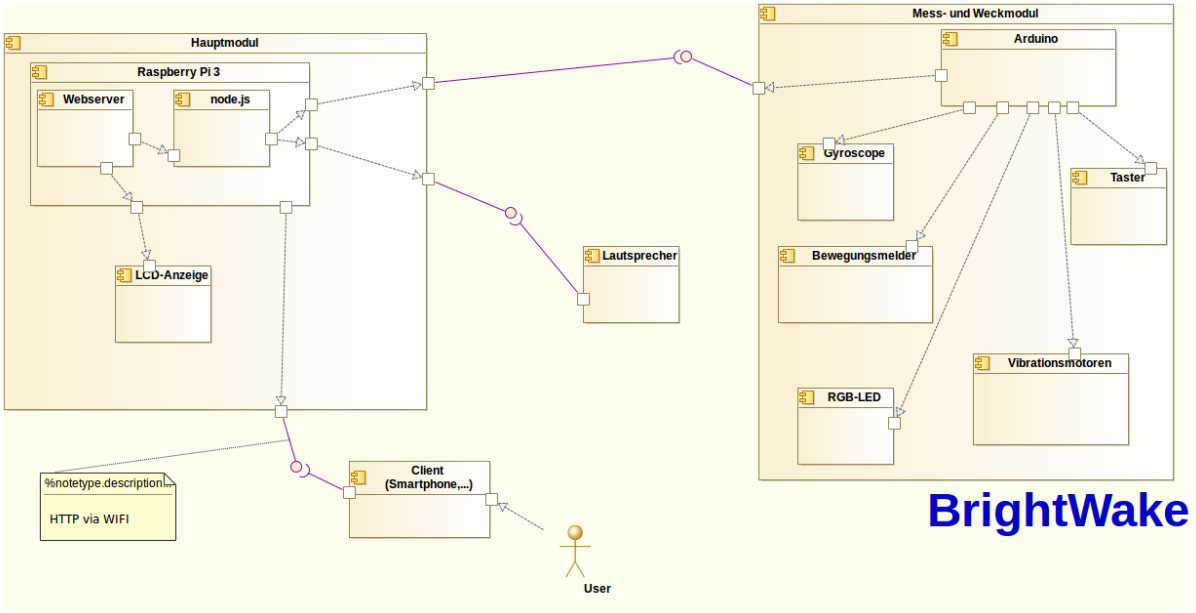
Dieses sollte zu Anfang durch ein Pretzelboard realisiert werden, da die Verfügbarkeit von drahtloser Kommunikation zwischen Haupt- und Zusatzmodul gegeben sein sollte. Aufgrund von Kommunikationsproblemen durch Inkompatibilitäten mit dem MQTT-Protokoll wurde das Pretzelboard jedoch gegen ein Arduino Yun ausgetauscht. Aus Zeitgründen wurden dieses via USB an den Raspberry Pi angeschlossen. Die gewünschte drahtlose Kommunikation ist beim Prototyp somit nicht umgesetzt worden. Jedoch kann diese Funktion mit wenig Aufwand nachträglich implementiert werden.

Der Prototyp: Aufbau und Schema

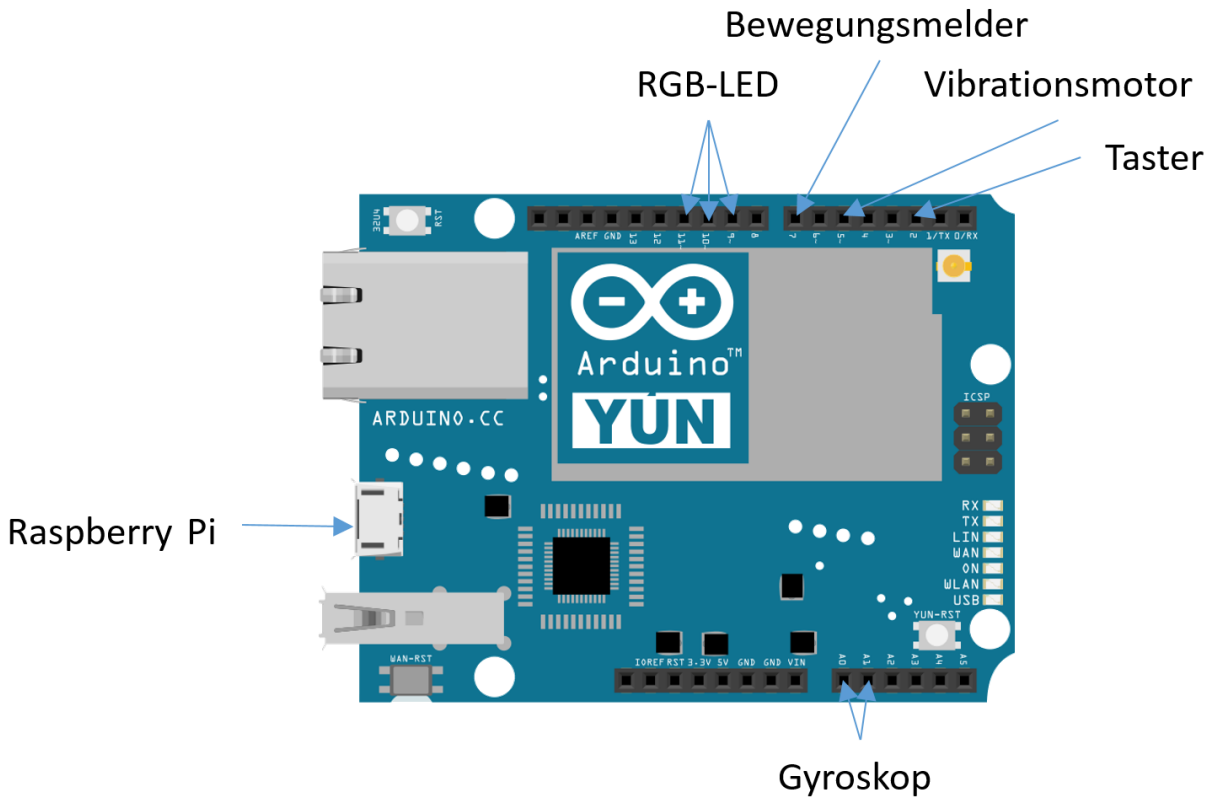
Das folgende Steckplatinendiagramm zeigt zunächst den Aufbau des Prototyps.



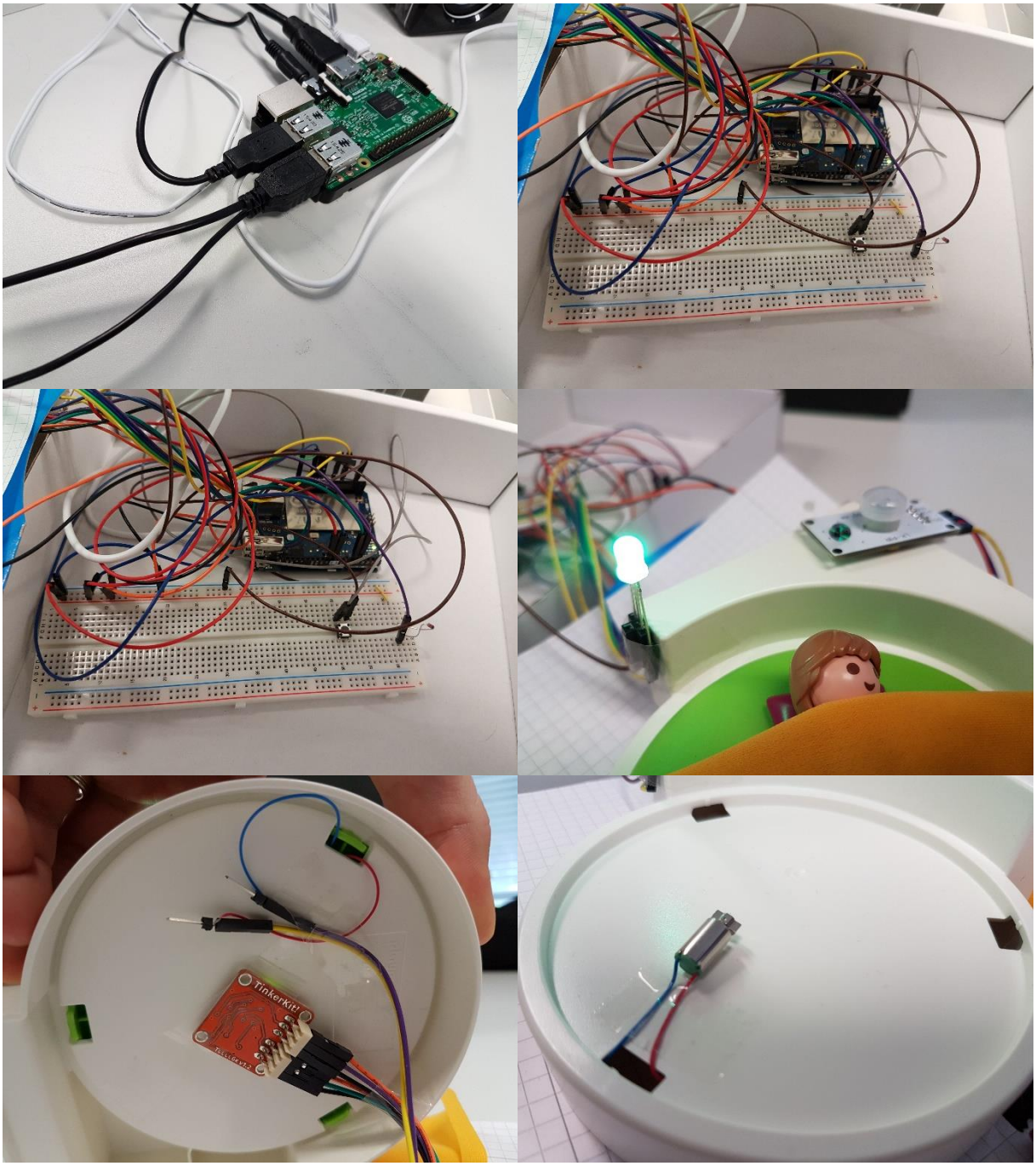
Das folgende Komponentendiagramm verdeutlicht nun, wie die einzelnen Hard- und Softwarekomponenten miteinander interagieren.



Die folgende Darstellung verdeutlicht noch einmal die Pinbelegung der Komponenten auf dem Arduino.



Die folgenden Fotos zeigen ein paar Impressionen des Prototyps. Alle Fotos und Diagramme sowie zwei Videos sind in unserem GitHub-Repository (Link s. u.) zu finden.



Programmlogik

Im nächsten Schritt haben wir die Sensoren und Aktoren mit Hilfe von kurzen Testskripten getestet und konfiguriert und anschließend das Hauptskript erstellt, welches die komplette Logik und „künstliche Intelligenz“ von BrightWake enthält.

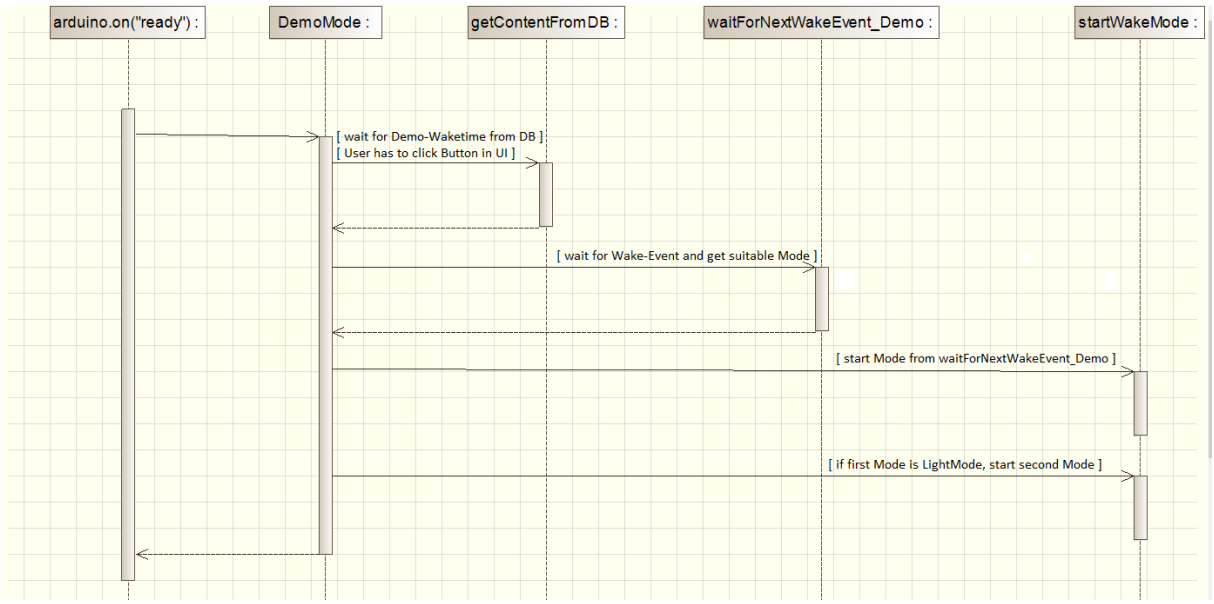
Die benötigten Skripte wurden mit Hilfe von JohnnyFive und weiteren oben genannten NodeJS-Modulen umgesetzt. Diese bieten uns die Möglichkeit die Logik recht einfach zu implementieren, ohne umständlich zu werden. Dank JohnnyFive ist das Einbinden der Komponenten sehr einfach (Vgl. Die übliche Programmierung von Arduino und Raspberry Pi ist recht umständlich).

Durch die „künstliche Intelligenz“ und die zwei Anwendungs-Modi (Demomodus zur Demonstration des Prototyps und ein DB-Modus zum „normalen“ Gebrauch) ist die Programmlogik relativ komplex geworden.

Da eins der Grundkonzepte von NodeJS Asynchronität ist, musste der überwiegende Teil des Codes in Intervalle und Timeouts gegliedert werden. Die Intervalle dienen hierbei ausschließlich zur regelmäßigen Überprüfung, ob beispielsweise bestimmte Variablen gesetzt sind oder die Daten aus der Datenbank bereits aufbereitet zu Verfügung stehen.

Im Folgenden wird hauptsächlich auf den Demo-Modus eingegangen, da an diesem die wesentliche Logik erläutert werden kann, ohne zu sehr in die Tiefe zu gehen. Der DB-Modus ist jedoch bis auf eine größere Anzahl an Datenbankzugriffen und eine somit höhere Komplexität größtenteils äquivalent.

Das folgende Sequenz-Diagramm verdeutlicht im Wesentlichen die Funktionsaufrufe des Demo-Modus. Parameter, konkrete Variablen und entsprechende SQL-Statements werden der Einfachheit halber nicht mit aufgeführt.



Jeder der aufgeführten Funktionsaufrufe erfolgt in einem Intervall, der unterbrochen und entfernt wird, wenn ein bestimmter Zustand (beispielsweise setzen der nächsten Weckzeit) eintritt.

Kurzbeschreibung und Ablauf der beiden Modi

Demomodus

Wie bereits beschrieben dient der Demomodus lediglich zur Demonstration und spiegelt die eigentliche Programmlogik nicht komplett wider.

Das Skript wartet im Hintergrund darauf, dass der Nutzer den Demo-Button im Userinterface anklickt. Geschieht dies, wird ein neuer Datensatz mit der Demoweckzeit in die Datenbank (nähere Beschreibung siehe Abschnitt „Datenbankanbindung“) geschrieben. Diese ist der Zeitpunkt des „Buttonclicks“ plus eine Minute.

Nun wartet das Skript auf den Zeitpunkt, an dem der Nutzer geweckt werden soll und fragt in bestimmten Intervallen ab, wie der Nutzer momentan schläft, sprich ob das Gyroskop Bewegungen wahrnimmt.

Wurde 20 Sekunden vor der Weckzeit keine Bewegung detektiert, wird zunächst der Lichtmodus und zum Zeitpunkt des eigentlichen Weckvorgangs schließlich der sekundäre Weckmodus gestartet. Der sekundäre Weckmodus ist hart-codiert und dient nur zur Demo eines der Aktoren (LED / Vibrationsmotor).

Wurde jedoch eine Bewegung wahrgenommen, wird zum Weckzeitpunkt geprüft, wie lange diese zurückliegt und der passende Modus (kürzer als 5 Sekunden → Soundmodus oder länger als 5 Sekunden → Vibrationsmodus) gestartet.

Der Nutzer kann die Weckmodi wie oben beschrieben „snoozen“ bzw. beenden.

Datenbankmodus

Der Datenbankmodus stellt die komplette Logik hinter BrightWake dar. Der Ablauf ist im Grunde äquivalent zum Demomodus.

Zu Beginn holt sich das Skript den nächsten Weckzeitpunkt aus der Datenbank und wartet auf diesen. Wie oben beschrieben wird die Bewegung des Nutzers in bestimmten Intervallen am Gyroskop abgefragt. Jedoch wird der Lichtmodus hier bereits 30 Minuten vor dem eigentlichen Weckzeitpunkt eingeleitet. Zum eigentlichen Weckzeitpunkt wird der vom Nutzer für den „Leichtschlaf" definierte Weckmodus aus der Datenbank geholt und ausgeführt.

Wurde jedoch eine Bewegung wahrgenommen, wird wie im Demomodus beschrieben zum Weckzeitpunkt geprüft, wie lange diese zurückliegt und der passende Modus gestartet. Jedoch ist die Zeitspanne auch hier verlängert (5 Minuten statt 5 Sekunden).

Auch hier ist es dem Nutzer möglich den Weckvorgang zu pausieren bzw. zu beenden.

Wesentliche Codeabschnitte

In diesem Abschnitt gehen wir auf die wesentlichen Variablen und Codefragmente als Pseudocode ein. Codeteile, die sich in den Modi wesentlich unterscheiden, werden kommentiert um auch die Hauptlogik zu erklären. Benötigte Intervalle und Timeouts werden der Einfachheit halber nur angedeutet. Der komplette Code ist in unserem GitHub-Repository (Link s. u.) einsehbar.

Zunächst werden kurz wichtige globale Variablen aus dem JavaScript erläutert, die verwendet werden. Diese spielen jedoch für den Pseudocode keine Rolle.

lastGyroDetection

- Zeitpunkt (Date) der letzten Bewegung des Nutzers
- Mit Hilfe des Gyroskops ermittelt
- Wird verwendet um den passenden Weckmodus zu ermitteln

motionStartTime

- Zeitpunkt (Date) an dem der Bewegungsmelder eine Bewegung wahrnimmt
- Ist dieser länger als eine Sekunde her, wird der Weckmodus pausiert („snooze"-Funktion)

isButtonPressed

- Boolean, der besagt, ob der Taster gedrückt wird oder nicht

Nun können die wesentlichen Codefragmente erläutert werden.

```
Funktion DemoMode () {
   Intervall 1 {
        Warte auf Buttonclick in GUI bzw. Datensatz in Datenbank
        Wenn erfolgreich ein Datensatz aus der Datenbank gelesen wird, diesen aufbereiten
        und in „demoWakeTime“ speichern und Intervall 1 löschen
   }
   Intervall 2 {
        Wenn „demoWakeTime“ gesetzt
             Dann auf Weckzeitpunkt warten (waitForNextWakeEvent_Demo in bestimmten Intervall
             aufrufen), „wakeMode“ wird je nach Nutzerschlaf gesetzt
   }
   Intervall 3 {
        Wenn „wakeMode“ gesetzt
             Dann diesen Weckmodus starten und Intervall 2 und 3 löschen
             Wenn „wakeMode“ ist „Lichtmodus“
                Dann sekundären Weckmodus („Vibrationsmodus“ oder „SoundModus“) zum
                eigentlichen Weckzeitpunkt starten
                // Im Datenbankmodus wird hier der Weckmodus für „Leichtschlaf“ ermittelt
   }
}
```

```
Funktion waitForNextWakeEvent_Demo (nextWakeTime) {
   // 1. Prüfzeitpunkt
   Wenn 20 Sekunden (DB-Modus 30 Minuten) vor dem Wecken keine Bewegung detektiert
        Dann wird „mode“ mit „Lichtmodus“ belegt
   // 2. Prüfzeitpunkt
   Wenn das Weckereignis jetzt ausgeführt werden soll
        Dann Zeitpunkt der letzten Bewegung überprüfen
        Wenn letzte Bewegung vor mehr als 5 Sekunden (DB-Modus 5 Minuten) vor dem Wecken
        (Nutzer in Normalschlaf)
            Dann ist „mode“ = „Vibrationsmodus“
        Wenn letzte Bewegung vor weniger als 5 Sekunden (DB-Modus 5 Minuten) vor dem
        Wecken wahrgenommen (Nutzer in Leichtschlaf)
            Dann ist „mode“ = „Soundmodus“
        // Im Datenbank-Modus wird hier der entsprechende Weckmodus passend zum Schlafmodus
        // aus der Datenbank geholt
   Passenden Weckmodus zurückgeben
}
```

Folgendes Codefragment erklärt den Ablauf eines Weckmodus und der intelligenten „snooze“-Funktion anhand des Beispiels „Soundmodus“. Der Ablauf ist äquivalent zu den anderen beiden Weckmodi.

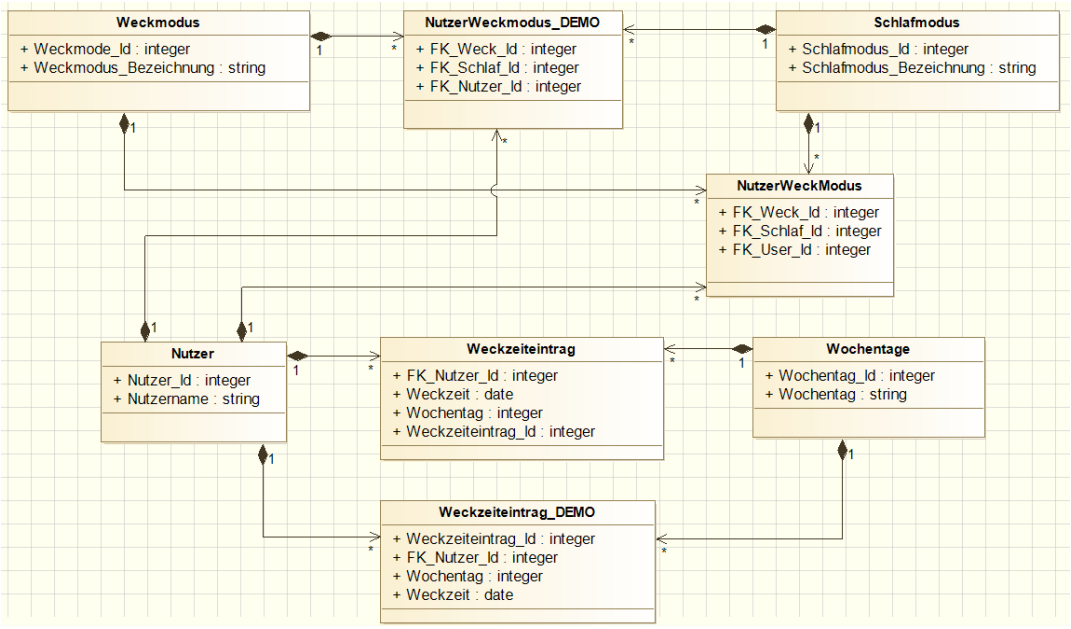
```
Funktion SoundMode () {
   Weckton in Schleife abspielen
   Intervall {
        Wenn Taster gedrückt wurde
            Dann Intervall löschen und Weckton stoppen
        Wenn Bewegung länger als 1 Sekunde detektiert
            Dann Weckton für eine bestimmte Zeit pausieren (Timeout) und anschließend weiter
            abspielen
   }
}
```

Datenbankanbindung

Wir haben uns konkret dazu entschieden, die benötigten Daten und Einstellungen in einer MySQL-Datenbank zu persistieren, da dies die einfachste und schnellste Methode der Datenspeicherung darstellt. Somit ist es auch die Interaktion zwischen Nutzer und BrightWake wesentlich einfacher.

Auch in Hinblick auf die Zukunft ist dies von Vorteil. Ich könnte mich beispielsweise auf Reisen in einem Hotel mit meinen Zugangsdaten auf einem dortigen BrightWake einloggen und würde wie gewohnt geweckt werden.

Das folgende ER-Diagramm zeigt die Datenbankstruktur. Hierin sind beide Modi realisiert.



Ein SQL-Skript zur Erstellung der Datenbank ist in unserem GitHub-Repository (Link s. u.) einsehbar.

USERINTERFACE

Das Userinterface, welches beispielsweise per Smartphone oder PC über das Web erreichbar ist, wurde mit HTML und CSS umgesetzt. Als Datenbankschnittstelle haben wir PHP verwendet. Dieses besteht aus drei Oberflächen.

Hauptoberfläche
Die Hauptoberfläche zeigt die aktuelle Zeit und das Datum an. Über je einen Button sind von hier aus die Oberflächen zur Weckzeit- und Weckmoduseinstellung erreichbar. Über das Feld „Demo-Modus“ können die Weckmodi für den Demomodus eingestellt und dieser über einen Button gestartet werden.

Weckzeit
Über diese Oberfläche kann der Nutzer seine gewünschten Weckzeiten verwalten.

Weckmodus
Hier kann der Nutzer jedem Schlafmodus einen gewünschten Weckmodus per Dropdown-Menü zuweisen.

Die Dateien des Userinterfaces sind in unserem GitHub-Repository im Ordner GUI (Link s. u.) einsehbar. Zum Ausführen benötigt man einen üblichen Browser und einen PHP-Interpreter wie z. B. XAMPP.

PROBLEME UND HÜRDEN BEI DER UMSETZUNG

Wie bereits in einem vorherigen Abschnitt erwähnt, musste das Pretzelboard aufgrund von Inkompatibilitäten gegen ein Arduino Yun ausgetauscht werden. Nach dem Austausch funktionierte die Kommunikation wie gewünscht. Jedoch wurde wie oben beschrieben auf die drahtlose Funktion verzichtet und eine Kommunikation über USB gewählt.

Ein weiteres Hardwareproblem ergab sich mit dem Display zur Anzeige der Uhrzeit am Hauptmodul. Bei der Kompilierung der dafür vorgesehenen Bibliotheken gab es Fehler. Eine mögliche Alternative wäre die Sieben-Segment-Anzeige von Adafruit (4-fach, I2C).

Eine Hürde stellte wie in Abschnitt „Programmlogik“ beschrieben die Asynchronität in NodeJS dar. Diese konnte jedoch mit den Routinen Intervall und Timeout umgangen werden. Somit stellte diese Hürde kein Problem dar.

ZUKÜNFTIGE ÄNDERUNGEN FÜR EIN MARKTREIFES PRODUKT

Umsetzung eines marktreifen Produkts

Der Prototyp von BrightWake stellt an sich noch kein komplett marktreifes Produkt dar. Die Demonstration erfolgte an einem Miniatur-Versuchsaufbau. Folgende Bauteile wären für ein marktreifes Produkt auszutauschen bzw. hinzuzufügen.

- Philips HUE statt RGB-LED
- Vibrationsmotoren vergrößern oder Wearable vibrieren lassen
- Größere Anzahl an Gyroskopen
- LCD-Display integrieren
- Drahtlose Kommunikation zwischen Modulen mit MQTT implementieren

Mögliche Erweiterungen des Prototyps

Um die Innovation von BrightWake zu erweitern, sind folgende ergänzende Funktionen denkbar.

- Bewegungen über ein Wearable (Smartwatch o. ä.) überwachen
- Erweiterte Weckmodi (modulare Erweiterung)
    - Z. B. wegziehen der Decke oder automatisches Öffnen der Jalousien
- Prüfung nach dem Weckvorgang, ob der Nutzer aufgestanden ist, beispielsweise per Gewichtssensor
- Mehrnutzerbetrieb / Login auf „fremden“ BrightWake-Weckern im Hotel o. ä.
- Einbinden in Smart-Home / mit weiteren IoT-Geräten koppeln

Mögliches Aussehen eines marktreifen Produkts

Die folgenden Bilder zeigen ein Beispiel dafür, wie BrightWake als marktreifes Produkt aussehen könnte. Diese Arbeiten stammen aus einem privaten Wecker-Projekt, welches allerdings auf die Funktionen verzichtet, Bewegungen im Schlaf zu messen und Vibrationsalarm auszulösen. Weiterhin besteht dieser Wecker nur aus dem hier abgebildeten Hauptmodul. Ansonsten sind die Ziele dieses Projekts ähnlich der von BrightWake, der Nutzer soll individuelle Wecker und Weckmodi für jeden Tag über ein Webinterface einstellen können, welches hier auch über den 7-Zoll-Touchscreen zu erreichen ist. Ebenfalls soll die Snooze-Funktion per Bewegungsmelder aktiviert werden können (Gelbe LED soll leuchten, solange „Snooze“ bei Alarm aktiv ist). Über das Webinterface soll ein Button „Alarm aus“ betätigt werden können, um den Alarm endgültig auszuschalten (Grüne LED soll für einige Sekunden aufleuchten). Der Wecker soll ebenfalls über das Webinterface auf „Alarm aktiv“ oder „Alarm inaktiv“ geschaltet werden können (Rote LED leuchtet, solange „Alarm aktiv“ aktiviert ist).

Das Gehäuse dieses Wecker-Projekts wurde aus Hartschaumplatten gefertigt. Die einzelnen Teile wurden dazu mit einem Bastelmesser sauber in Form gebracht. Danach wurden die Teile mit Metallwinkeln und Schrauben zu einem hinterseitig offenen Gehäuse verbunden. Die LEDs werden durch LED-Halterungen in den entsprechenden Bohrlöchern gehalten. Als Touchscreen wurde hier das offizielle Raspberry-Pi-Touchscreenmodul mit Treiberplatine (am rückseitig befestigten Raspberry Pi angebracht) verwendet. In dem hier abgebildeten Aufbau ist die vierfache Sieben-Segment-Anzeige zur Darstellung der Uhrzeit noch nicht in das Gehäuse eingebaut und auch noch nicht in Betrieb genommen. Für diese Komponente und den Bewegungssensor wären beispielsweise zwei weitere Öffnungen oberhalb des Touchscreens erforderlich.

FAZIT

BrightWake, der smarte Wecker, ist ein innovatives Projekt des Internet of Things, welches die Möglichkeit bietet, in beliebiger Form erweitert oder optimiert zu werden. Sei es auf funktionaler oder optischer Ebene (Gehäuse, Userinterface). Es gibt zahlreiche Funktionen und Komponenten, um die BrightWake erweitert werden könnte, wie im Abschnitten „Zukünftige Änderungen für ein marktreifes Produkt“ erläutert wurde.

Der Verlauf des Projekts hat uns gezeigt, dass eine genaue Planung wichtig ist und, dass trotz einer guten Planung dennoch Probleme und Hürden auftreten können. Zu Beginn unserer Planung hatten wir ein bestimmtes Bild von unserem Prototyp und dessen Funktionalitäten. Wir mussten zwar diversen Planänderungen vornehmen, jedoch kommt das fertige Produkt „BrightWake“ unseren Vorstellungen sehr nahe. Wir konnten fast alle Funktionalitäten, wie z. B. die Weckmodi, mit den bestellten Komponenten umsetzen. Es musste lediglich die Phillips HUE gegen eine RGB-Led getauscht werden, was jedoch dem modellhaften Aufbau des Prototyps zu schulden ist. Der Austausch des Pretzelboards gegen ein Arduino Yun war der Inkompatibilität geschuldet. Leider mussten wir daher auch aus Zeitgründen auf die drahtlose Kommunikation verzichten. Diese kann jedoch nachträglich schnell implementiert werden.

Trotz der kleinen Änderungen sind wir mit unserem Budget unter der 200€-Marke geblieben.

Am Ende des Projekts haben wir einen lauffähigen Prototyp, der zwar hier und da noch kleine Schwächen aufweist, diese aber mit relativ geringem Aufwand an Tests und kleinen Änderungen zu beheben sind.

LINKS UND WESENTLICHE QUELLEN

https://github.com/SoerenJesseGruen/BrightWake

http://johnny-five.io/

https://www.npmjs.com/

https://nodejs.org/en/