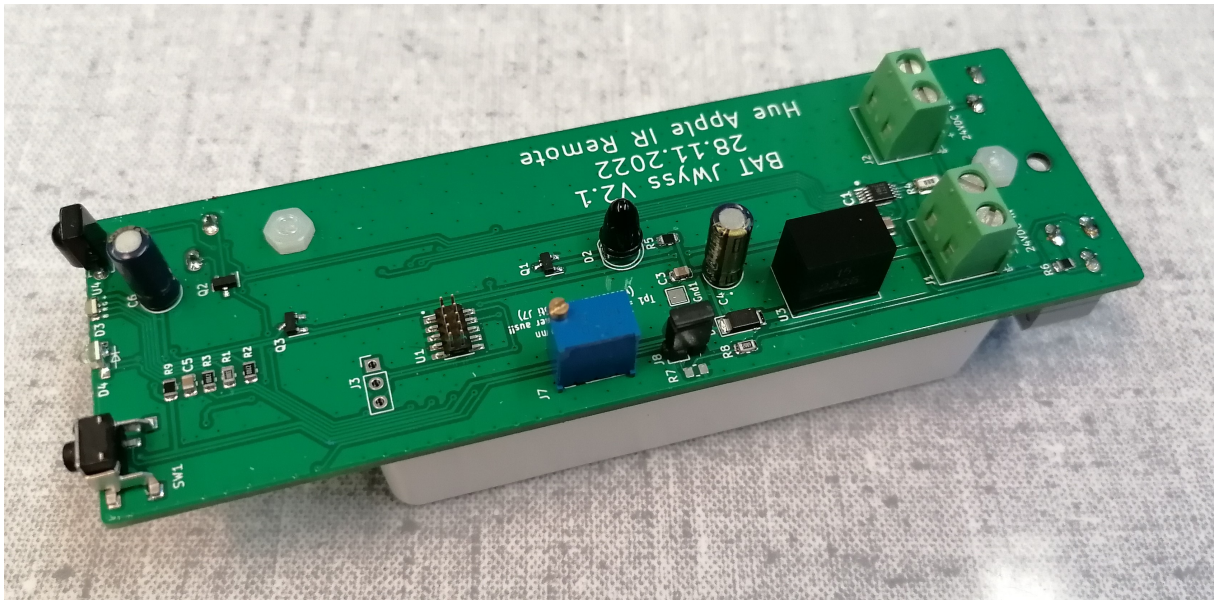
Hue Apple IR Remote

Bachelor-Thesis

Herbstsemester 2022



Autor:

Joshua Wyss

Datum: 29. Dezember 2022

Betreuender Dozent:	Prof. Styger, Erich
Experte:	Dr. Vetterli, Christian

Bachelor-Thesis an der Hochschule Luzern - Technik & Architektur

Titel	**Hue Apple IR Remote**
Diplomandin/Diplomand	**Wyss, Joshua**
Bachelor-Studiengang	**Bachelor Elektrotechnik und Informationstechnologie**
Semester	**HS22**
Dozentin/Dozent	**Styger, Erich**
Expertin/Experte	**Vetterli, Christian**

Abstract Deutsch
In dieser Bachelorarbeit wird eine Mikrocontrollerschaltung und die dazugehörige Software entwickelt, welche es ermöglicht, eine nicht smarte Bürostehleuchte in das Philips Hue Automatisierungssystem zu integrieren. In der Dokumentation wird beschrieben, wie das Lösungskonzept erarbeitet wurde, gefolgt von einer detaillierten Beschreibung der entwickelten Hardware und Software. Das Ergebnis dieser Arbeit ist ein funktionsfähiger Prototyp, welcher am Leuchtenständer befestigt und zwischen dem Netzteil der Leuchte und der Leuchte selber eingesteckt wird. Dies ermöglicht es, die Stehleuchte wie ein originales Philips Hue Produkt in das Philips Hue System zu integrieren. Durch den Prototypen kann die Stehleuchte mit der Philips Hue App von überall her gesteuert werden. Wird die Stehleuchte mit der bestehenden Fernbedienung ein- oder ausgeschaltet, wird der Zustand der Stehleuchte in der Hue App ebenfalls aktualisiert. Die Frage, ob man die Bürostehleuchte wirklich ausgeschaltet hat, gehört somit der Vergangenheit an, da dies einfach auf dem Smartphone überprüft und je nachdem ausgeführt werden kann.

Abstract Englisch
In this thesis, a microcontroller circuit and the associated software is developed, which makes it possible to integrate a non-smart office floor lamp into the Philips Hue automation system. The documentation describes how the solution concept was developed, followed by a detailed description of the developed hardware and software. The result of this work is a functional prototype, which is attached to the light stand and plugged in between the power supply of the light and the light itself. This makes it possible to integrate the floor lamp into the Philips Hue system like an original Philips Hue product. Thanks to the prototype, the floor lamp can be controlled from anywhere using the Philips Hue app. If the floor lamp is switched on or off by the existing remote control, the state of the floor lamp is also updated in the Hue app. The question of whether you really switched off the office floor lamp is a thing of the past, as this can be easily checked on your smartphone and executed accordingly.

Ort, Datum Horw, 29.12.2022

© Joshua Wyss, Hochschule Luzern – Technik & Architektur

Alle Rechte vorbehalten. Die Arbeit oder Teile davon dürfen ohne schriftliche Genehmigung der Rechteinhaber weder in irgendeiner Form reproduziert noch elektronisch gespeichert, verarbeitet, vervielfältigt oder verbreitet werden.

Sofern die Arbeit auf der Website der Hochschule Luzern online veröffentlicht wird, können abweichende Nutzungsbedingungen unter Creative-Commons-Lizenzen gelten. Massgebend ist in diesem Fall die auf der Website angezeigte Creative-Commons-Lizenz.

Inhaltsverzeichnis

1 Einleitung und Zielsetzung 1
1.1 Einleitung . . . 1
1.2 Aufgabenstellung . . . 1
1.2.1 Gottardo Stehleuchte . . . 2
1.2.2 Philips Hue Automatisierungssystem . . . 5
1.3 Anforderungen . . . 6

2 Projektmanagement und Projektplanung 7
2.1 Planung und Vorgehen . . . 7
2.2 Risiken . . . 8
2.3 Verwendete Programme . . . 8

3 Recherche 9
3.1 Infrarot Fernbedienungen . . . 9
3.1.1 NEC Infrarot Übertragungsprotokoll . . . 9
3.1.2 Protokoll Apple IR Remote . . . 11
3.2 NEC IR Signale mit dem Raspberry Pi Pico versenden und empfangen . . . 11
3.3 SPI . . . 12

4 Erarbeitung und Beschreibung des Lösungskonzepts 13
4.1 Aufteilung in Teilbereiche . . . 13
4.1.1 Einbindung in das Hue System . . . 13
4.1.2 Senden und Empfangen von Infrarot Signalen . . . 14
4.1.3 Statuserkennung der Stehleuchte . . . 14
4.1.4 Platzierung der Hue Apple IR Remote . . . 14
4.1.5 Speisung der Hue Apple IR Remote . . . 14
4.1.6 Auswahl des Mikrocontrollers . . . 14
4.2 Einbindung in Philips Hue . . . 15
4.2.1 Hue kompatible Geräte . . . 15
4.2.2 Umbau eines bestehenden Hue Geräts . . . 15
4.2.3 ZigBee Light Link SDK . . . 16
4.2.4 Hue Wallswitch zur Statusänderung in der App . . . 17
4.2.5 Verwendung eines Hue Smart Plugs . . . 18
4.2.6 Verwendung von Hue kompatiblen Schaltaktoren . . . 18
4.3 Statuserkennung der Leuchte . . . 19
4.3.1 Statuserkennung mit einem Lichtsensor . . . 19
4.3.2 Statuserkennung durch die Multianzeige der Leuchte . . . 20
4.3.3 Statuserkennung über Strommessung in der Zuleitung . . . 20
4.3.4 Bildverarbeitung . . . 21
4.3.5 Indirekte Statuserkennung durch bekannte Referenzpunkte . . . 21
4.4 Speisung der Hue Apple IR Remote . . . 22
4.4.1 Speisung über eine Batterie / einen Akku . . . 22
4.4.2 Kabelgebundene Speisung . . . 22
4.5 Platzierung der Hue Apple IR Remote . . . 22
4.5.1 Platzierung am Leuchtenkopf . . . 22

4.5.2 Platzierung am Ständer . . . 22
4.5.3 Platzierung auf Lampenfuss . . . 23
4.5.4 Platzierung im Leuchtenkopf . . . 23
4.5.5 Platzierung losgelöst von der Leuchte . . . 23
4.6 Vorstellung Lösungskonzept Hardware . . . 24
4.7 Begründung Lösungskonzept . . . 24

5 Erarbeitung und Beschreibung der Hardware 26
5.1 Beschreibung Komponenten . . . 26
5.1.1 Raspberry Pi Pico Breakoutboard . . . 26
5.1.2 ZigBee LED Dimmer SR-ZG9101CS . . . 27
5.1.3 Stromsensor INA239 . . . 27
5.1.4 Stromsensor INA260 . . . 28
5.1.5 DC-DC-Wandler TSR 1-2450 . . . 28
5.1.6 IR Diode SFH 4545 . . . 29
5.1.7 IR Diode APDA3020F3C-P22 . . . 29
5.1.8 IR-Receiver TSOP38238 . . . 30
5.1.9 RGB-LED - R6GHBHW . . . 30
5.2 Erklärung PCB und Workflow in KiCad . . . 31
5.3 Schaltplan PCB V1.1 . . . 32
5.3.1 Mikrocontroller . . . 32
5.3.2 Debug Schnittstelle . . . 33
5.3.3 Speisung Pico . . . 33
5.3.4 Strommessung . . . 34
5.3.5 Anschlussklemme . . . 34
5.3.6 Kommunikation mit ZigBee LED Dimmer . . . 35
5.3.7 IR rx / tx . . . 36
5.3.8 User Interface . . . 36
5.3.9 Montagelöcher . . . 37
5.4 PCB Version 1.1 . . . 38
5.4.1 Festlegung der Grösse des PCB V1.1 . . . 38
5.4.2 Komponentenplatzierung . . . 38
5.4.3 Routing . . . 39
5.4.4 Leiterbahnbreiten . . . 39
5.4.5 PCB Review 1 . . . 40
5.4.6 Testergebnisse PCB V1.1 . . . 41
5.5 Schaltplan PCB V2.1 . . . 42
5.5.1 Anpassungen Kommunikation mit ZigBee LED Dimmer . . . 42
5.5.2 Anpassungen Strommessung . . . 43
5.5.3 Anpassungen Anschlussklemme . . . 43
5.5.4 Anpassung IR rx / tx . . . 44
5.5.5 Anpassung User Interface . . . 44
5.6 PCB Version 2.1 . . . 45
5.6.1 Komponentenplatzierung . . . 45
5.6.2 Routing und Leiterbahnbreiten . . . 46
5.6.3 PCB Review 2 . . . 46

5.6.4 Bestücktes PCB V2.1 . . . 47
5.7 Gehäuse . . . 48
5.7.1 Gehäuse V1 Mock-Up . . . 48
5.7.2 Gehäuse V2 . . . 49
5.7.3 Gehäuse V3 . . . 50
5.8 Kosten Hardware . . . 51

6 Erarbeitung und Beschreibung der Software 52
6.1 Übersicht Softwarekonzept . . . 52
6.1.1 Beschreibung der Zustände . . . 53
6.2 Verwendung von FreeRTOS . . . 54
6.3 Aufbau Software . . . 55
6.4 Softwaremodule . . . 55
6.4.1 MainTask.c . . . 55
6.4.2 States.c . . . 56
6.4.3 HUEInterface.c . . . 58
6.4.4 IR.c . . . 59
6.4.5 SPI.c . . . 61
6.4.6 LED.c . . . 62
6.4.7 MyButton.c . . . 62
6.5 Dimmen der Stehleuchte mit der H-A-IR-Remote . . . 64
6.6 Strommessung mit dem INA239 . . . 65
6.7 Source Review . . . 67
6.8 Debugging das Picos mit einem Segger JLink und Eclipse . . . 67
6.8.1 Aufbau zum Debuggen des Picos . . . 67
6.8.2 Blink Programm auf dem Pico . . . 69
6.8.3 Free RTOS auf dem Pico . . . 69

7 Testberichte 70
7.1 Test: Lichtsensor LDR . . . 70
7.2 Test: Stromverbrauch Leuchte . . . 71
7.3 Test: ZigBee LED Dimmer . . . 72
7.3.1 Test: Einbindung in Philips Hue . . . 72
7.3.2 Test: Rückmeldung über den Taster . . . 73
7.3.3 Test: Stromverbrauch Standby . . . 74
7.3.4 Test: Speisung über Leuchtennetzteil . . . 75
7.3.5 Test: Tastereingang mit Microcontroller betätigen . . . 75
7.3.6 Test: Generierung eines 0 - 3.3 Volt Signals aus dem LED Dimmer Ausgang . . . 76
7.3.7 Lösungsansätze um das Ausgangssignal des LED Dimmers mit dem Pico zu messen . . . 77
7.4 Test bestücktes PCB V1.1 . . . 79
7.4.1 Test: Strommessung mit dem INA239 . . . 79
7.4.2 Test: IR Signale versenden und empfangen . . . 80
7.4.3 Test: Kommunikation LED Dimmer . . . 81
7.4.4 Test: User Interface . . . 82
7.5 IR Signale der Apple IR Remote aufzeichnen . . . 82

7.6 Test: Energieverbrauch Hardware V2.1 . . . 83

8 Bedienungsanleitung H-A-IR-Remote 84
8.1 Legende . . . 84
8.2 Inbetriebnahme . . . 85
8.3 Bedienungshinweise . . . 85
8.4 Softwareupdate . . . 86
8.5 Fehlerbehandlung . . . 86

9 Schlussteil 87
9.1 Erfüllungsgrad der Anforderungen . . . 87
9.2 Fachliches Fazit . . . 88
9.3 Ausblick und Erweiterungsideen . . . 89
9.4 Persönliches Fazit . . . 91
9.5 Danksagung . . . 91

10 Verzeichnisse 92
Abbildungsverzeichnis . . . 92
Tabellenverzeichnis . . . 95
Literaturverzeichnis . . . 96
Abkürzungsverzeichnis . . . 98

A Anhang 99
A.1 Struktur Elektronischer Anhang . . . 99
A.1.1 Aufgabenstellung . . . 99
A.1.2 Datenblätter . . . 99
A.1.3 KiCAD Projekte . . . 99
A.1.4 Software Projekt . . . 99
A.1.5 CAD Dateien Gehäuse . . . 99
A.2 Aufgabenstellung . . . 100
A.3 Schema PCB V1.1 . . . 102
A.4 Schema PCB V2.1 . . . 103
A.5 Kostenauflistung Bauteile . . . 104
A.6 Strombelastbarkeit von Leiterbahnen . . . 105
A.7 ZigBee . . . 106

1 Einleitung und Zielsetzung

1.1 Einleitung

Begriffe wie Smarthome und Internet of Things (IoT) sind heutzutage fast jedem ein Begriff und es ist nichts Neues mehr, dass smarte Geräte und Leuchten von überall her mit dem Smartphone gesteuert werden können. Dennoch können viele Leuchten in Büros noch nicht mit einem Smarthome System gesteuert werden. Damit solche nicht smarte Leuchten in ein Smarthome System integriert werden können, soll im Rahmen dieser Bachelorthesis eine Mikrocontrollerschaltung und die dazugehörige Software entwickelt werden, welche das Einbinden einer nicht smarten Leuchte in ein Smart Home System ermöglicht. Konkret soll eine Gottardo Stehleuchte in ein Philips Hue Automatisierungssystem integriert werden. Das Ziel ist es, dass die Stehleuchte neben der bestehenden Ansteuerung über die Infrarotfernbedienung auch mit der Philips Hue App gesteuert werden kann.

Der Hauptteil der vorliegenden Arbeit ist folgendermassen aufgebaut: In einem ersten Teil wird detailliert beschrieben, wie das Lösungskonzept erarbeitet wurde. Im Anschluss wird gezeigt, wie das Lösungskonzept in Form eines Prototypen umgesetzt wurde und wie die Software dazu aufgebaut ist. Am Ende des Hauptteils ist eine Bedienungsanleitung für den entwickelten Prototypen zu finden.

1.2 Aufgabenstellung

Wie in der Einleitung bereits angedeutet, soll eine Schaltung mit einem Mikrocontroller und der dazugehörigen Software entwickelt werden, welche es ermöglicht, eine Gottardo Stehleuchte in das Philips Hue Automatisierungssystem einzubinden. Die Gottardo Stehleuchte kann ab Werk nur mit einer Apple Infrarotfernbedienung (Apple IR Remote) oder mit den Bedientasten am Leuchtenkopf gesteuert werde. Die zu entwickelnde Mikrocontrollerschaltung (Hue Apple IR Remote, kurz H-A-IR-Remote) soll also die Leuchte wie eine Fernbedienung mit Infrarot Signalen steuern und gleichzeitig den Status der Leuchte dem Hue System bereitstellen können. In Kapitel 1.3 sind die Anforderungen an die H-A-IR-Remote etwas detaillierter aufgeführt. Die Abbildung 1 zeigt die Aufgabenstellung als Blockdiagramm. Das originale Dokument der Aufgabenstellung ist im Anhang A.2 zu finden.

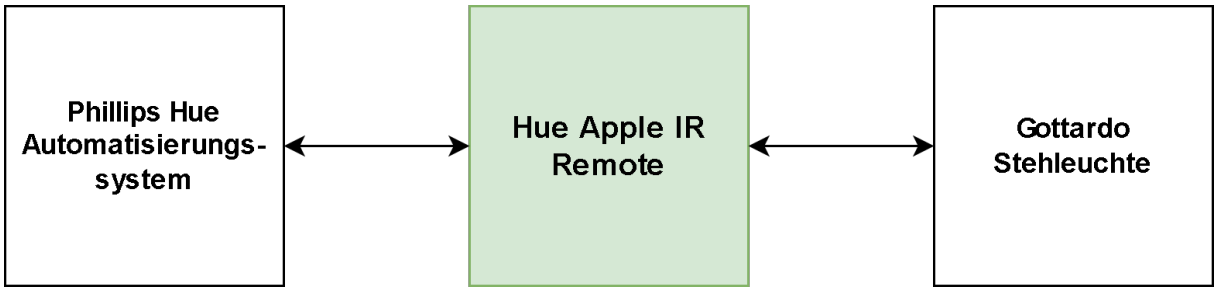


Abbildung 1: Blockdiagramm Aufgabenstellung

Die nachfolgenden Kapitel 1.2.1 und 1.2.2 geben eine Übersicht über die Gottardo Stehleuchte und das Philips Hue Automatisierungssystem.

1.2.1 Gottardo Stehleuchte

Die Gottardo Stehleuchte besteht aus der Stehleuchte selber und einem separaten 24 Volt Netzteil, an welchem die Leuchte eingesteckt wird. Die Leuchte kann entweder mit einer Apple IR Remote (siehe Abbildung 4) oder über das Bedienfeld (siehe Abbildung 3), welches sich am Leuchtenkopf befindet, gesteuert werden. Die Stehleuchte verfügt über zwei Lichtsysteme, welche nach oben und nach unten Licht abgeben. Die Gottardo Stehleuchte ist in der Abbildung 2 zu sehen.

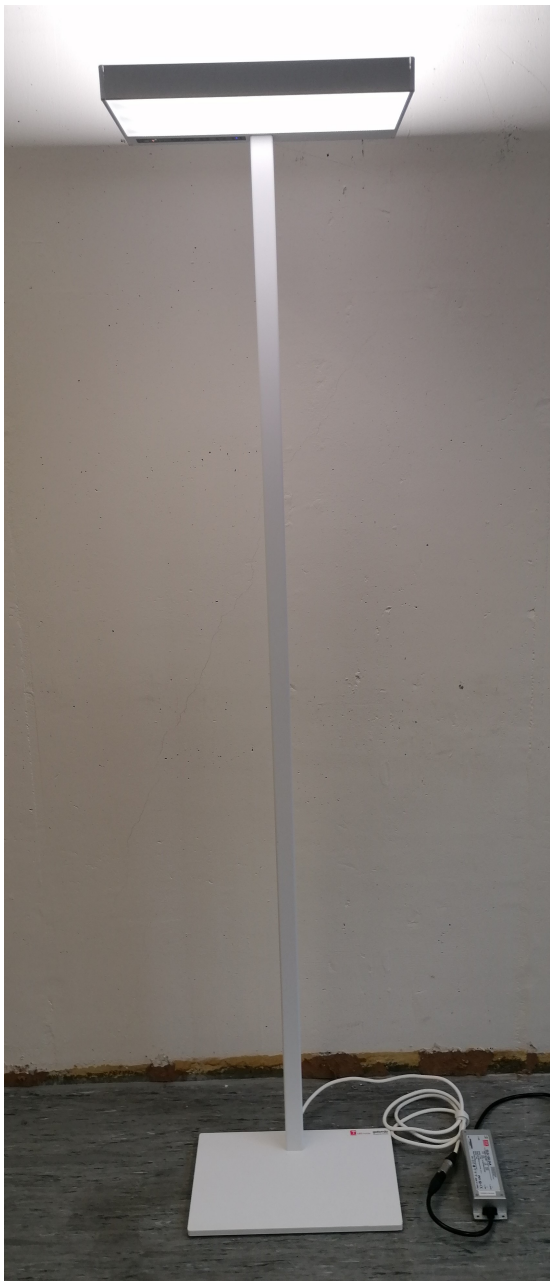


Abbildung 2: Gottardo Stehleuchte

Bedienung der Gottardo Stehleuchte
Die Stehleuchte kann mit der Taste Power ein- und ausgeschaltet werden. Wird die Powertaste gedrückt und gehalten, beginnt die Leuchte stufenlos hoch- oder runterzudimmen. Über die Taste Hoch- und Herunterdimmen kann die Stehleuchte entweder stufenlos durch drücken und halten der Taste oder in drei verschiedenen Helligkeitsstufen, durch kurzes betätigen der Taste gedimmt werden. Mit den Pfeiltasten links und rechts kann die Lichtszene geändert werden und mit der Taste Szenengruppe die Szenengruppe. Die verschiedenen Lichtszenen und Szenengruppen sind in der Abbildung 5 ersichtlich. Durch das Betätigen der Taste Synchronisieren gelangt man direkt in die Szenengruppe Wohnen. Neben der Steuerung mit der Fernbedienung und dem Bedienfeld kann auch der integrierte Bewegungssensor aktiviert werden, dann schaltet die Leuchte ein, sobald eine Bewegung registriert wird.

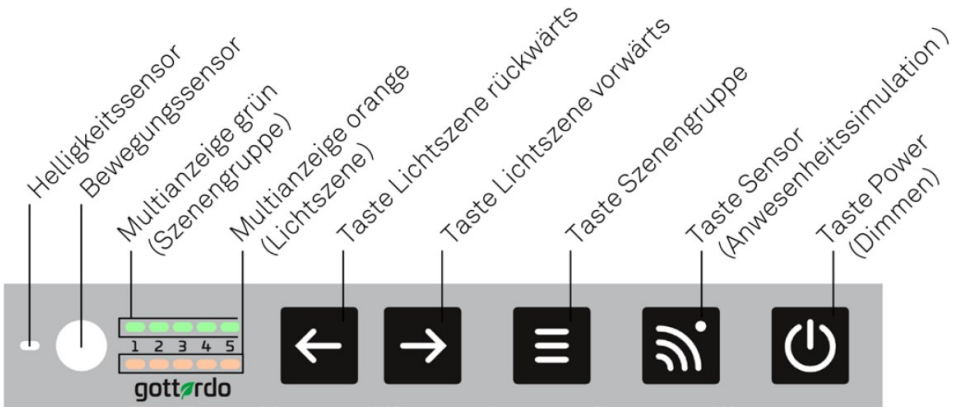


Abbildung 3: Bedienfeld [1, S. 4]

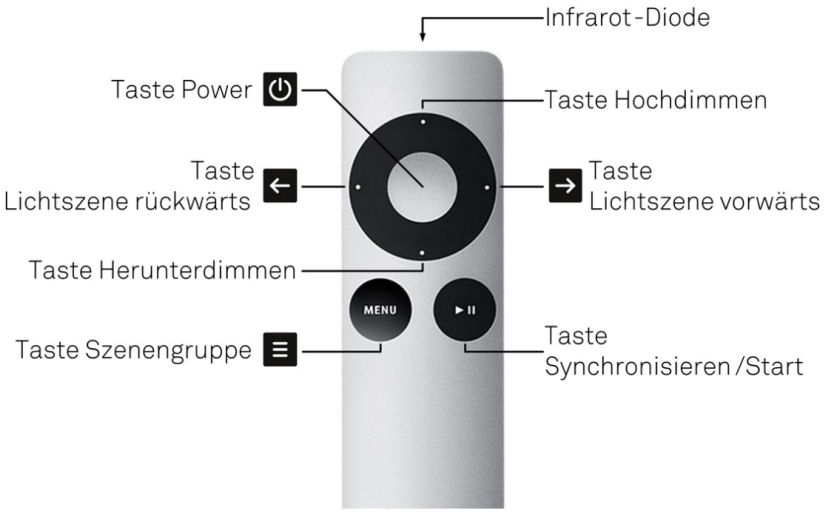


Abbildung 4: Apple IR Remote [1, S. 4]

Übersicht Szenen
Die Stehleuchte verfügt über fünf Szenengruppen mit jeweils fünf Lichtszenen. Die Abbildung 5 bietet eine Übersicht über die Szenengruppen und Lichtszenen. Je nach Lichtszene leuchten das obere- und untere Lichtsystem in anderen Lichtfarben. Es ist zu beachten, dass in den Szenen 2.1 und 2.2 der Lichtregler werksmässig aktiviert ist. Das bedeutet, dass eine Sollbeleuchtungsstärke eingestellt werden kann und die Leuchte dafür sorgt, dass die gewünschte Lichtstärke immer vorhanden ist.

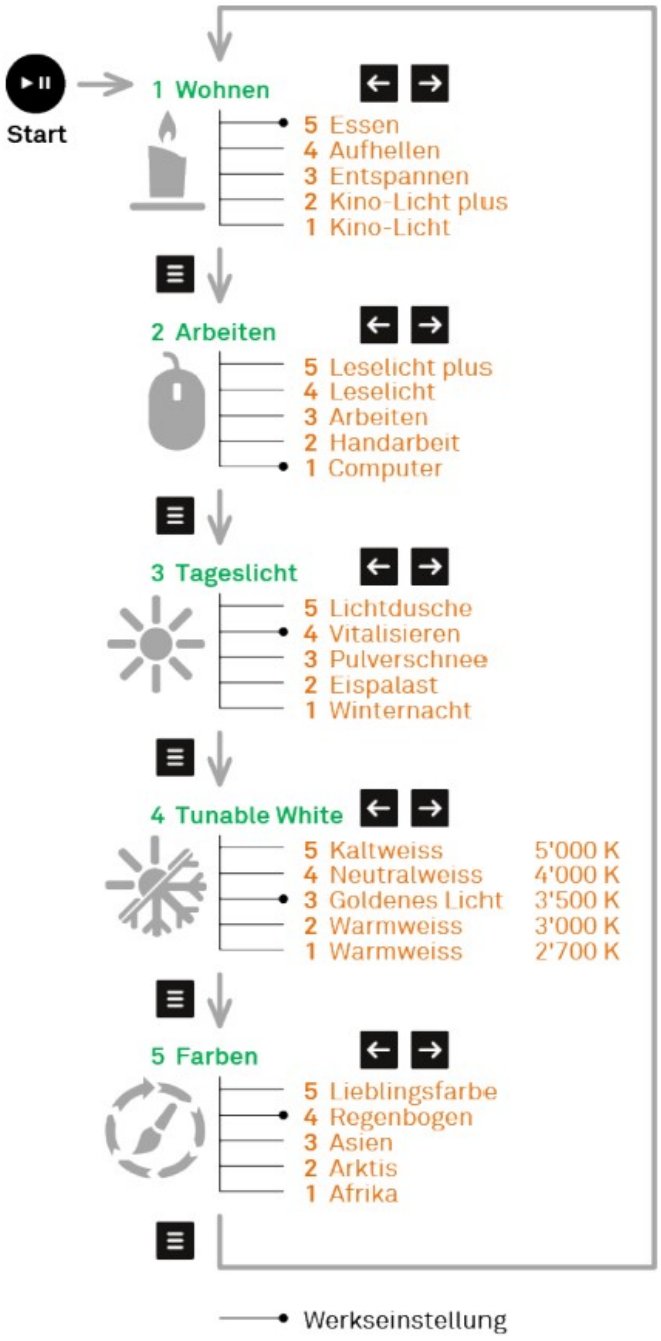


Abbildung 5: Szenengruppen und Lichtszenen [1, S. 15]

1.2.2 Philips Hue Automatisierungssystem

Das Philips Hue Automatisierungssystem ermöglicht das Steuern von Hue kompatiblen Leuchten und Leuchtmitteln durch das Smartphone mit einer App. Die Leuchten können entweder per Bluetooth mit dem Smartphone verbunden werden oder an einer Hue Bridge (Abbildung 6) angemeldet und dann über die Hue App gesteuert werden. Da über Bluetooth nur zehn Lampen und ein Raum gesteuert werden können und die Steuerung von unterwegs nicht möglich ist, wird bei dieser Arbeit mit der Hue Bridge gearbeitet. Somit können bis zu 50 Geräte eingebunden werden und alle Geräte können auch von unterwegs über die Hue App gesteuert werden [2]. Die Kommunikation zwischen Leuchten und Bridge basiert auf dem ZigBee Light Link Protokoll [3]. Genauere Informationen zum ZigBee Protokoll sind in Kapitel A.7 zu finden.

Abbildung 6: Philips Hue Bridge

Abbildung 7: Philips Hue Birne

Die Abbildung 8 zeigt den Aufbau des Philips Hue Systems mit allen Komponenten und Schnittstellen. Ausserdem ist ersichtlich, wie der Benutzer mit dem System interagieren kann.

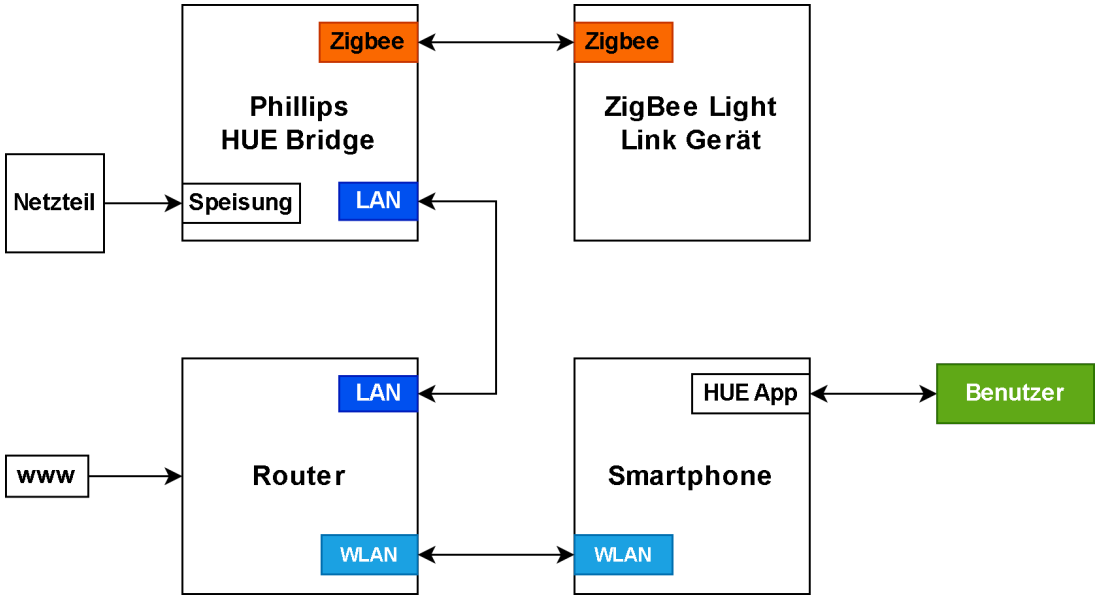


Abbildung 8: Blockschaltbild Philips Hue System

1.3 Anforderungen

In diesem Kapitel werden die Anforderungen an die Hue Apple IR Remote definiert.

1. **Einfache Integration in das Philips Hue System:** Die H-A-IR-Remote muss wie ein originales Philips Hue Produkt in das Philips Hue System integrierbar sein. Es dürfen keine zusätzlichen Schritte zur Integration notwendig sein.
2. **Leuchte ein- und ausschalten:** Die Gottardo Stehleuchte muss in der Hue App ein- und ausgeschaltet werden können.
3. **Leuchte Dimmen:** Die Gottardo Stehleuchte muss in der Hue App gedimmt werden können.
4. **Anzeige des aktuellen Leuchtenzustands in der Hue App:** Der Zustand der Stehleuchte in der App muss zu jeder Zeit dem Zustand der Stehleuchte entsprechen. Wird die Stehleuchte durch die Fernbedienung, die Bedientasten an der Leuchte oder durch den integrierten Bewegungsmelder ein- oder ausgeschaltet, muss der Zustand innerhalb von fünf Sekunden in der App aktualisiert werden.
5. **Keine Funktionseinschränkung der Stehleuchte:** Die Bedienung der Stehleuchte mit der originalen Fernbedienung oder den Bedientasten an der Leuchte darf durch die H-A-IR-Remote nicht verändert oder eingeschränkt werden.
6. **Energieverbrauch:** Die H-A-IR-Remote darf nicht mehr als 1 Watt verbrauchen
7. **Bedienungsanleitung:** Es muss eine Bedienungsanleitung für die H-A-IR-Remote vorhanden sein, welche die Montage, die Inbetriebnahme und die Verwendung der H-A-IR-Remote für Laien verständlich erklärt.
8. **Einfache Montage:** Die Montage der H-A-IR-Remote muss nach dem Lesen der Bedienungsanleitung innerhalb von einer Minute und ohne Werkzeuge durch einen Laien durchgeführt werden können.
9. **Einfache Demontage:** Die H-A-IR-Remote muss innerhalb von einer Minute Rückstandslos und ohne Werkzeuge von der Gottardo Stehleuchte entfernt werden können.
10. **Keine Veränderung der Stehleuchte:** Die Gottardo Stehleuchte darf durch die Montage oder Demontage der H-A-IR-Remote in keinster Weise beschädigt oder verändert werden.
11. **Softwareupdate:** Es muss einem Laien möglich sein, mit der Bedienungsanleitung ein Software-Update auf die H-A-IR-Remote zu laden.
12. **Anwendbarkeit auf ähnliche Leuchten:** Die H-A-IR-Remote muss mit Leuchten, ähnlich der Gottardo Stehleuchte, kompatibel sein. Damit eine Leuchte ähnlich ist, gelten die folgenden Bedingungen: Die Leuchte muss durch ein externes Netzgerät mit 12 - 36 Volt DC gespiesen werden, durch einen Stecker mit dem Netzgerät verbunden sein und mit einer NEC-Infrarotfernbedienung gesteuert werden können.

2 Projektmanagement und Projektplanung

2.1 Planung und Vorgehen

Am Anfang des Projekts wurden Meilensteine definiert und darauf aufbauend ein Zeitplan erstellt. Der Zeitplan ist in der Abbildung 9 zu sehen. Die roten Kästchen zeigen an, bis wann der jeweilige Meilenstein erreicht werden soll.

Meilensteine	SW	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Kickoffmeeting																
Lösungskonzept fertig																
Zwischenpräsentation																
PCB 1 und Komponenten bestellt																
Softwareblöcke Fertig																
Testing Prototyp 1 abgeschlossen																
Finales PCB bestellt																
Finale Software fertig																
Testing Prototyp 2 abgeschlossen																
Fertigstellung Schlussbericht																

Abbildung 9: Meilensteine und Zeitplan

In der ersten Woche findet das Kick-off-Meeting statt, in welchem die Aufgabenstellung detailliert besprochen wird. Im Anschluss wird jede Woche ein Meeting durchgeführt, bei welchem der Stand der Arbeit vorgestellt wird. Auch Fragen und Probleme können in diesen wöchentlichen Meetings diskutiert werden. Nach dem Kick-off-Meeting sieht das Vorgehen wie folgt aus: Bis in der fünften Semesterwoche (SW) ist das Ziel, ein Lösungskonzept für die H-A-IR-Remote erarbeitet zu haben. Dieses soll dann bis in SW7 in Form eines Prototypen V1 umgesetzt werden. In SW7 ist eine Zwischenpräsentation vorgesehen, um auch den Experten auf den aktuellen Stand der Arbeit zu bringen. Parallel zur Erstellung des Prototypen V1 soll bereits mit der Software begonnen werden, damit die einzelnen Softwaremodule bis in SW8 fertig gestellt werden können. Mit den Softwaremodulen kann dann der erste Prototyp getestet werden. Es besteht das Risiko, dass bei dem PCB für den ersten Prototypen Fehler gemacht werden, deshalb ist ein Redesign des PCBs bis in SW10 vorgesehen. Während die finale PCB produziert und geliefert wird, können die einzelnen Softwaremodule zu einer Software zusammengefügt werden, welche die Aufgabenstellung löst. Das Ziel ist, dass der finale Prototyp bis in SW13 getestet und lauffähig ist. Es ist geplant, den Prozess laufend zu dokumentieren, die Erfahrung hat jedoch gezeigt, dass der grösste Teil der Dokumentation am Ende der der Arbeit anfällt. Deshalb werden am Ende des Semesters zwei Wochen Zeit eingeplant, um die Dokumentation abzuschliessen. Diese Zeit kann im Notfall auch genutzt werden, um letzte Probleme zu beheben.

2.2 Risiken

In diesem Kapitel werden die für dieses Projekt relevanten Risiken aufgeführt und es wird erläutert, wie diese minimiert werden können.

1. **Lange Lieferfristen und reduzierte Verfügbarkeit von Bauteilen:** Im Moment sind gewisse Bauteile begrenzt oder gar nicht verfügbar und Lieferfristen können sich kurzfristig ändern, was das Projekt verzögern kann.

   (a) Es werden nur Bauteile gewählt, welche noch in grösserer Stückzahl beim Distributor verfügbar sind.

   (b) Damit das Projekt nicht verzögert wird, werden die Bauteile so früh wie möglich ausgewählt und in mehrfacher Stückzahl bestellt. Dies verhindert, dass Bauteile für die zweite Version der Hardware nicht mehr verfügbar sind.

2. **Fehler in der Hardware:** Beim Designen und Layouten des PCBs können Fehler passieren und zu einer nicht funktionierenden Schaltung führen.

   (a) Um Fehler zu vermeiden, wird die Hardware vor dem bestellen einer Drittperson gezeigt.

   (b) Für den Fall, dass sich dennoch Fehler auf der Hardware einschleichen, wird eine zweite Iteration eingeplant.

3. **Fehler in der Software:** Bei der Software, insbesondere bei den einzelnen Modulen können Fehler passieren, deren Behebung das Projekt verzögern können.

   (a) Die Software wird so früh wie möglich mit einer Testschaltung geprüft, um Fehlern vorzubeugen.

   (b) Um Fehler effektiv beheben zu können, wird mit einem Debugger gearbeitet.

2.3 Verwendete Programme

In diesem Kapitel wird aufgezeigt, mit welchen Programmen gearbeitet wurde.

- **KiCad 6.0:** Dieses Programm wurde verwendet, um die Schaltpläne zu zeichnen und die PCBs zu designen.
- **Autodesk Inventor Proffesional 2023:** Dieses Programm wurde verwendet, um das 3D Modell des Gehäuses zu zeichnen.
- **Texmaker:** Dieses Programm wurde verwendet um die Dokumentation in LaTeX zu erstellen.
- **MCU Xpresso IDE v11.5:** Diese IDE wurde verwendet, um Programme für den Mikrocontroller zu schreiben und diese zu debuggen.
- **draw.io:** Mit diesem Programm wurden sämtliche Blockdiagramme erstellt.
- **Dev-C++:** Dieses Programm wurde verwendet, um kleine Teile des Codes zu testen.

3 Recherche

In diesem Kapitel werden alle Rechercheergebnisse aufgeführt, welche im Verlauf der Arbeit benötigt werden oder welche wichtig sind für das Verständnis der Arbeit.

3.1 Infrarot Fernbedienungen

Infrarot Fernbedienungen senden Licht im unsichtbaren Infrarotbereich aus. Die Strahlungsquelle ist dabei eine Infrarotdiode [4]. Damit die Datenübertragung sicherer wird, werden die Datenbits nicht einfach als an und aus gesendet. Die Infrarotdiode wird mit einer Trägerfrequenz von 30 - 40 kHz moduliert. Dies verbessert die Störsicherheit des Empfängers erheblich, da zufällige Störsignale herausgefiltert werden können [5]. Die Abbildung 10 zeigt vereinfacht, wie ein IR Signal demoduliert wird.

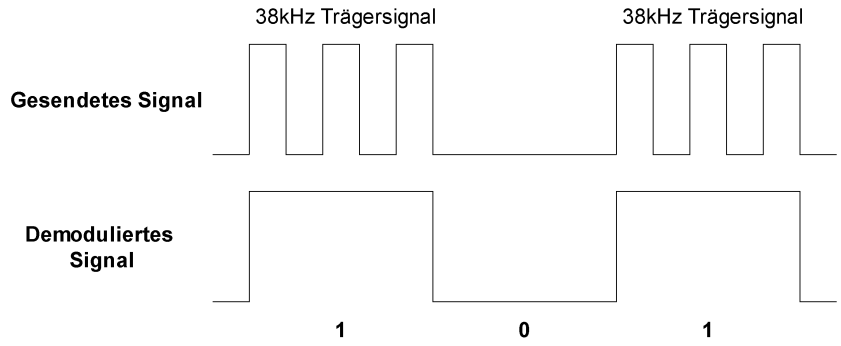


Abbildung 10: Vereinfachte Darstellung eines demodulierten IR Signals [5]

Für die Demodulation gibt es geeignete Empfänger, welche gleich das demodulierte Signal zur Verfügung stellen. Es gibt viele verschiedene Infrarot Protokolle, welche unterschiedliche Formate aufweisen. Die gesendete Bitmenge, die Trägerfrequenz und auch die Definition der Zustände 0 und 1 sind dabei unterschiedlich. Die Zustände 1 und 0 können zum Beispiel als Bitwechsel oder durch das Impuls / Pausenverhältnis definiert werden [5].

3.1.1 NEC Infrarot Übertragungsprotokoll

Die Informationen des Kapitels 3.1.1 stammen aus der Quelle [6]. Das NEC-Protokoll arbeitet mit einer Trägerfrequenz von 38 kHz. Die Zustände 1 und 0 sind durch das Impuls / Pausenverhältnis definiert. Logische Bits werden wie folgt übertragen:

	ON[ms]	OFF[ms]	Total[ms]
Logisch 0	0.5625	562.5	1.125
Logisch 1	0.5625	1.6875	2.25

Wird eine Taste auf der Fernbedienung gedrückt, wird eine Nachricht versendet, welche wie folgt aufgebaut ist.

- ein 9 ms langer führender Impulsstoss
- eine 4.5 ms lange Pause

- eine 8 Bit Adresse für den Empfänger
- die logisch invertierte Adresse
- ein 8 Bit Befehl
- der logisch invertierte Befehl
- ein 0.5625 ms langer Puls, um das Ende der Nachricht anzuzeigen

Die Abbildung 11 zeigt das Beispiel einer Nachricht für die Adresse 0x00 (0b00000000) mit dem Befehl 0xAD (0b10101101).

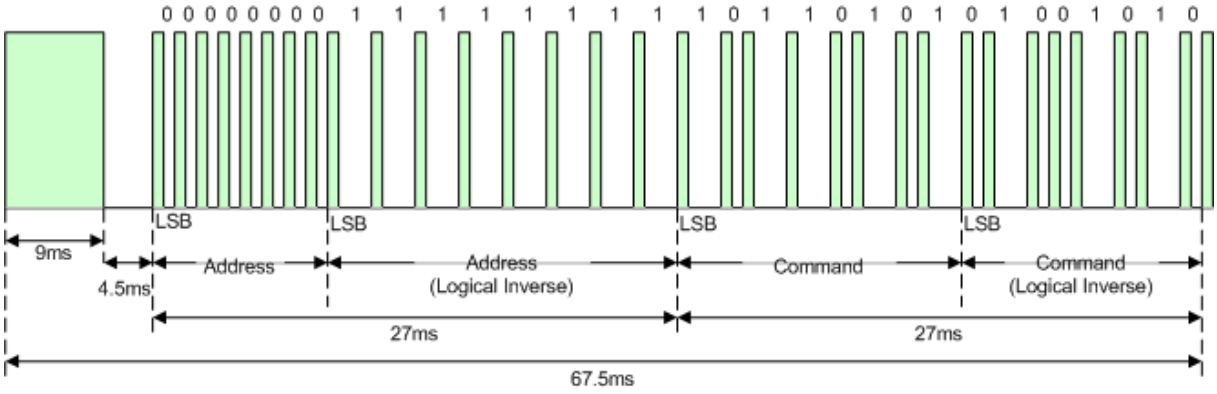


Abbildung 11: Beispiel für ein NEC-Frame [6]

Durch die Invertierung der Adresse und des Befehls dauert jede Nachricht gleich lang (67.5 ms).

Repeatcodes
Wenn eine Taste der Fernbedienung gedrückt gehalten wird, wird typischerweise 40 Milisekunden nach der Nachricht ein Repeatcode ausgesendet, welcher sich alle 108 ms wiederholt, bis die Taste losgelassen wird. Der Repeatcode ist wie folgt aufgebaut.

- ein 9 ms langer führender Impulsstoss
- eine 2.25 ms lange Pause
- ein 0.5625 ms langer Puls um das Ende der Nachricht anzuzeigen

Die Abbildung 12 zeigt beispielhaft, wie ein Nachricht mit anschliessendem Repeatcode aussieht.

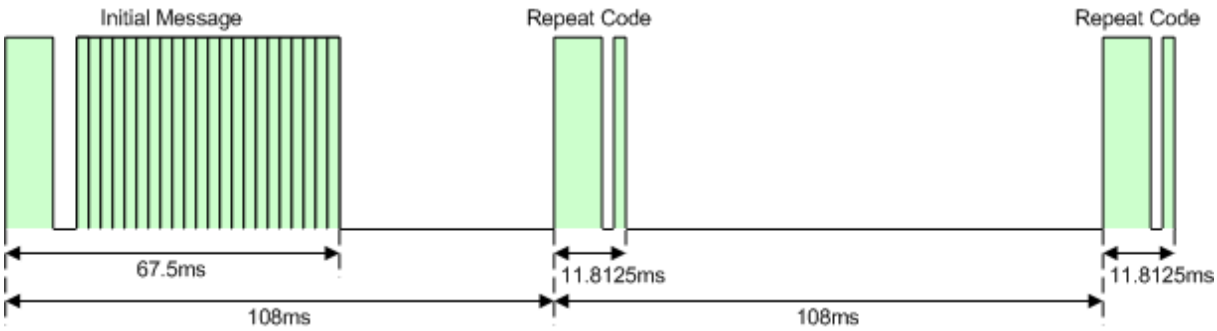


Abbildung 12: NEC-Frame mit Repeat Code (Taste wird gedrückt gehalten) [6]

3.1.2 Protokoll Apple IR Remote

Die Apple IR Remote arbeitet mit einem modifizierten NEC-Protokoll. Die Trägerfrequenz beträgt 38 kHz und es wird ein IR Emitter mit einer Wellenlänge von 950 nm verwendet. Wie bei der Standard NEC-Datenübertragung werden zwischen dem Start- und dem Stopbit 32-Bit Daten übertragen. Das von Apple verwendete modifizierte IR-Signal arbeitet jedoch mit einem anderen Datenformat. Die Daten sind wie folgt aufgebaut.

- 11 Bits um den Verkäufer (Apple) zu identifizieren
- 5 Bits für die Befehlsseite
- 8 Bits für eine gerätespezifische ID
- 7 Bits für den tatsächlichen Befehl
- 1 Odd Parity Bit (alle Bits zusammen müssen 1 ergeben)

Diese Informationen stammen aus der Quelle [7].

3.2 NEC IR Signale mit dem Raspberry Pi Pico versenden und empfangen

In diesem Kapitel wird gezeigt, wie IR Signale mit dem Raspberry Pi Pico versendet und empfangen werden können. Auf dem offiziellen GitHub Repository von Raspberry Pi befindet sich ein Beispiel, wie mit dem Raspberry Pi Pico IR NEC Signale versendet und empfangen werden können (siehe Quelle [5]). Die Abbildung 13 zeigt die Schaltung, welche dazu verwendet wird.

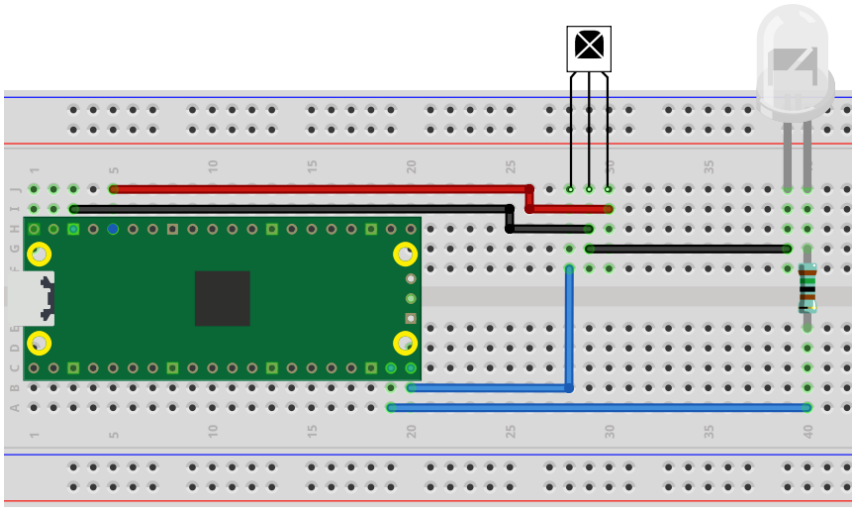


Abbildung 13: Schaltung um IR Signale mit dem Pico zu versenden und empfangen [5]

Das Beispiel zeigt, wie der Raspberry Pi Pico verwendet werden kann, um Infrarot-Frames im NEC-Format zu senden und zu empfangen. Der Hauptteil des Codes führt einen Loopback-Test durch, indem es IR-Codes über eine IR-Diode sendet und sie auf einem IR-Detektor empfängt. Die Ergebnisse werden an das serielle Standardterminal gesendet. Die Aufgaben der Codierung und Decodierung der Daten werden an die RP2040 PIO-Zustandsmaschinen ausgelagert. Dadurch werden die Hauptprozessorkerne entlastet, damit sie sich auf andere Aufgaben konzentrieren können. [5]

3.3 SPI

Die Informationen aus dem folgenden Abschnitt stammen aus der Quelle [8]. Beim Serial Peripheral Interface (SPI) handelt es sich um ein Bussystem, mit welchem digitale Schaltungen nach dem Master Slave Prinzip miteinander verbunden werden können. Für die Verbindung werden vier Leitungen benötigt.

- SCLK: Serial Clock
- MOSI: Master Output Slave Input
- MISO: Master Input Slave Output
- CS/SS: Chip Select / Slave Select

Die Abbildung 14 zeigt beispielsmässig auf, wie Master und Slaves verbunden werden.

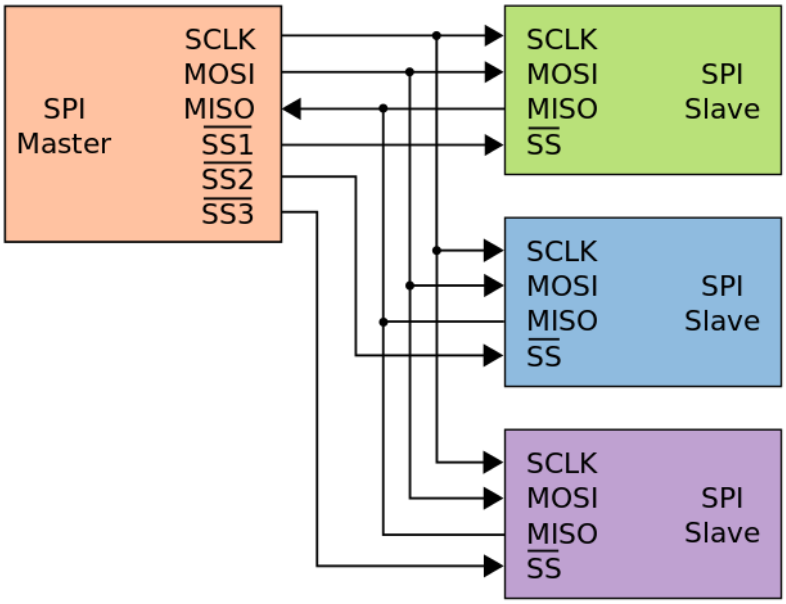


Abbildung 14: Sternförmige SPI Verbindung [8]

Die Anzahl von am Bus angeschlossenen Slaves wird nur durch die Anzahl der zur Verfügung stehenden CS Leitungen begrenzt. Zur Kommunikation erzeugt der Master ein Clock Signal auf der SCLK Leitung. Der Master entscheidet, mit welchem Slave kommuniziert werden soll, indem er die entsprechende CS Leitung auf Ground zieht. Dadurch wird der entsprechende Slave aktiv und lauscht auf der Leitung MOSI. Der Master sendet anschliessend eine Bitfolge, welche die Adresse eines Registers des Slaves enthält, welches der Master beschreiben oder lesen möchte. Auch ob der Master lesen oder Schreiben möchte, ist in dieser Bitfolge enthalten. Je nachdem, ob der Master lesen oder schreiben möchte, sendet der Master die Daten, die er in das Register schreiben möchte, oder erhält über die MISO Leitung die gewünschten Daten.

4 Erarbeitung und Beschreibung des Lösungskonzepts

In diesem Kapitel wird beschrieben, wie vorgegangen wurde, um das Lösungskonzept für die H-A-IR-Remote zu erarbeiten.

4.1 Aufteilung in Teilbereiche

In der Aufgabenstellung wird das Gesamtproblem beschrieben. Es soll eine nicht smarte Stehleuchte durch eine Mikrocontrollerschaltung in ein Philips Hue Automatisierungssystem integriert werden. Um strukturiert an das Erarbeiten eines Lösungskonzepts heranzugehen, wird das Gesamtproblem in einem ersten Schritt in mehrere Teilbereiche unterteilt. In diesem Kapitel werden die erarbeiteten Teilbereiche vorgestellt. Die folgenden Kapitel zeigen dann, welche Lösungsmöglichkeiten zu den Teilbereichen erarbeitet oder durch Recherche gefunden wurden. Die Abbildung 15 zeigt die Teilbereiche als Blockdiagramm auf.

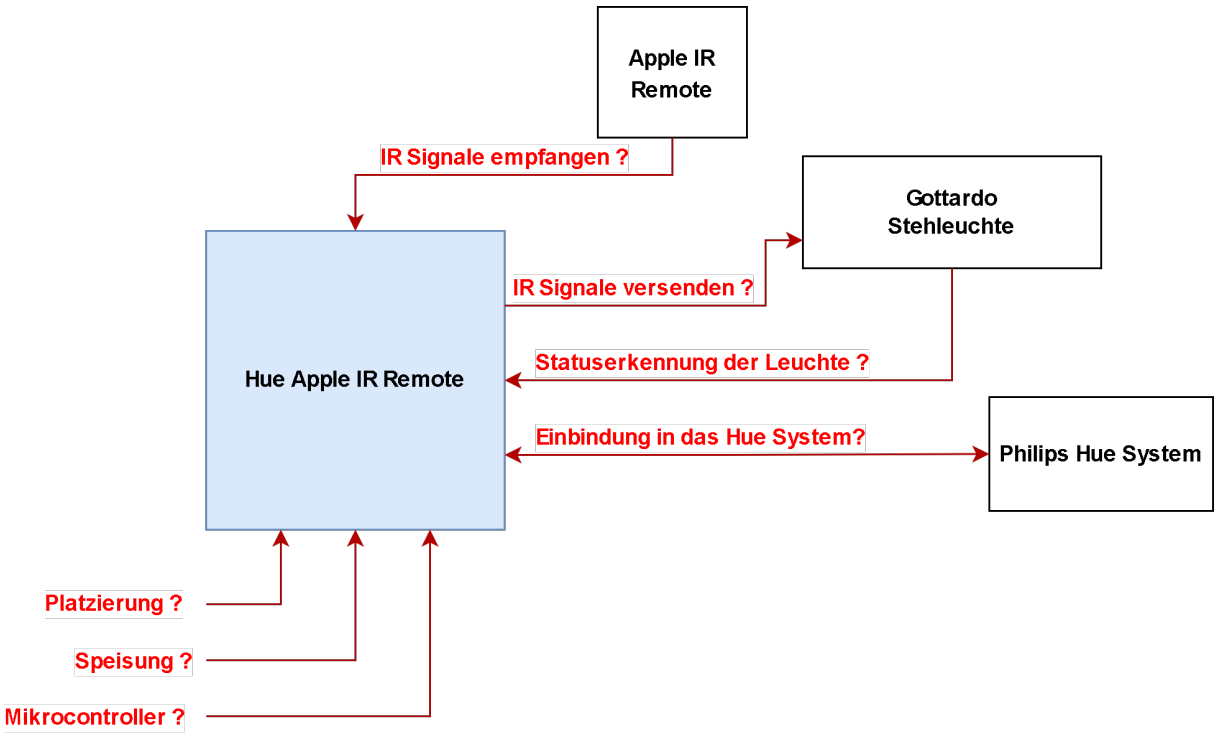


Abbildung 15: Übersicht Teilbereiche

4.1.1 Einbindung in das Hue System

Die Einbindung der Hue Apple IR Remote in das Philips Hue System ist eine Schlüsselfunktion, klappt die Einbindung nicht, so kann die Aufgabe nicht erfüllt werden. Mit der Einbindung ist gemeint, dass die H-A-IR-Remote wie jedes andere Gerät im Hue System angemeldet und verwendet werden kann. Die Stehleuchte muss gemäss den Anforderungen ein- und ausgeschaltet werden können. Dazu ist wichtig, dass die Ein- und Ausschaltbefehle von der Hue App dem Mikrocontroller bereitgestellt werden. Ausserdem muss es dem Mikrocontroller möglich sein, den Zustand der Leuchte in der App zu aktualisieren. Dies wird benötigt, wenn der Benutzer die Leuchte mit der Apple IR Remote oder mit den Bedientasten steuert. Es muss also eine

Lösung gefunden werden, welche eine bidirektionale Kommunikation zwischen Hue System und H-A-IR-Remote zulässt.

4.1.2 Senden und Empfangen von Infrarot Signalen

Damit die Leuchte überhaupt gesteuert werden kann, muss die H-A-IR-Remote in der Lage sein, Infrarotsignale zu versenden. Dazu muss herausgefunden werden, wie die Signale der Apple IR Remote aussehen und wie sie mit einem Mikrocontroller versendet werden können. Es gilt auch abzuklären, ob Infrarot Signale empfangen und verarbeitet werden müssen.

4.1.3 Statuserkennung der Stehleuchte

Im Kapitel 4.1.1 wurde bereits angedeutet, dass der Mikrocontroller den Status der Leuchte erkennen können muss, um Statusänderungen der Leuchte über die Apple IR Remote oder die Bedientasten zu erkennen und an das Hue System weiterzuleiten. Ausserdem wird die Leuchte mit der gleichen Taste ein- und ausgeschaltet, somit kann anhand des von der H-A-IR-Remote gesendeten Signals nicht eindeutig bestimmt werden, ob die Leuchte nun ein oder aus ist. Es muss also auch nach dem Senden eines Ein- oder Ausbefehls geprüft werden können, ob die Leuchte auch wirklich ein oder aus ist.

4.1.4 Platzierung der Hue Apple IR Remote

Ein weiterer wichtiger Aspekt ist, wo die H-A-IR-Remote platziert wird, denn Kommunikation mit der Leuchte geschieht über IR-Befehle, welche durch eine IR-Diode ausgesendet werden und beim Empfänger der Leuchte ankommen müssen.

4.1.5 Speisung der Hue Apple IR Remote

Bei der H-A-IR-Remote handelt es sich um eine Mikrocontrollerschaltung. Diese muss mit Energie versorgt werden. Es gilt eine geeignete Speisung zu finden.

4.1.6 Auswahl des Mikrocontrollers

Der Mikrocontroller wird das Gehirn der H-A-IR-Remote. Bei ihm laufen alle Signale zusammen, werden verarbeitet und es wird je nach Ereignis reagiert. Die Chipkrise hat zur Folge, dass nicht mehr alle Mikrocontroller verfügbar sind. Es gilt deshalb einen Mikrocontroller zu finden, welcher verfügbar und geeignet ist, um die Aufgaben zu erledigen.

4.2 Einbindung in Philips Hue

In diesem Kapitel geht es darum, herauszufinden, welche Möglichkeiten es gibt, um ein eigenes Gerät in Philips Hue einzubinden und was die Vor- und Nachteile der jeweiligen Methoden sind. In einem ersten Schritt wurde dazu eine Recherche durchgeführt, was das ZigBee Protokoll überhaupt ist. Denn die Geräte werden über das ZigBee Protokoll mit der Hue Bridge verbunden. Die Recherche Ergebnisse zum ZigBee Protokoll sind im Anhang A.7 zu finden.

4.2.1 Hue kompatible Geräte

Philips Hue verwendet den ZigBee Light Link (ZLL) Standard und sollte somit gemäss Quelle [3] auch mit derartigen Leuchten und Geräten kompatibel sein. Produkte, die den ZigBee HA Standard (siehe Anhang A.7) benutzen, sind deshalb nicht mit Phillips Hue kompatibel. Gemäss Quelle [9] sind bei Schaltern und Sensoren nebst den offiziellen Philips Hue Produkten, vor allem Produkte mit dem Label Friends of Hue kompatibel. Bei Lampen, Controllern und Smartplugs gibt es diverse Marken, welche kompatibel sind (z.B IKEA Tradfri, Innr ...). In der Quelle [9] wurden diverse Produkte von verschiedensten Herstellern getestet und beschrieben, wie gut sie mit dem Hue System funktionieren.

4.2.2 Umbau eines bestehenden Hue Geräts

Bei der Recherche nach Einbindungsmöglichkeiten wurde ein Video auf Youtube gefunden (Quelle [10]), in welchem das ZigBee Modul aus einem dimmbare IKEA-Tradfri-Leuchtmittel ausgebaut wurde. Dieses kann mit 3.3 Volt gespiesen werden und hat einen Outputpin, welcher je nach Dimmstufe, welche in der Hue App eingestellt werden kann, ein 3.3 Volt PWM-Signal von 0 bis 100 % herausgibt. Im Video wurde dieses PWM Signal mit einem Arduino ausgewertet und dazu genutzt, um je nach Dimmstufe unterschiedliche Szenen mit einem LED-Band zu realisieren. Das also zum Beispiel bei einem PWM Signal von 90 % das LED-Band immer abwechselnd rot und blau leuchtet. Die Abbildung 16 zeigt das ausgebaute ZigBee Modul aus dem Video.

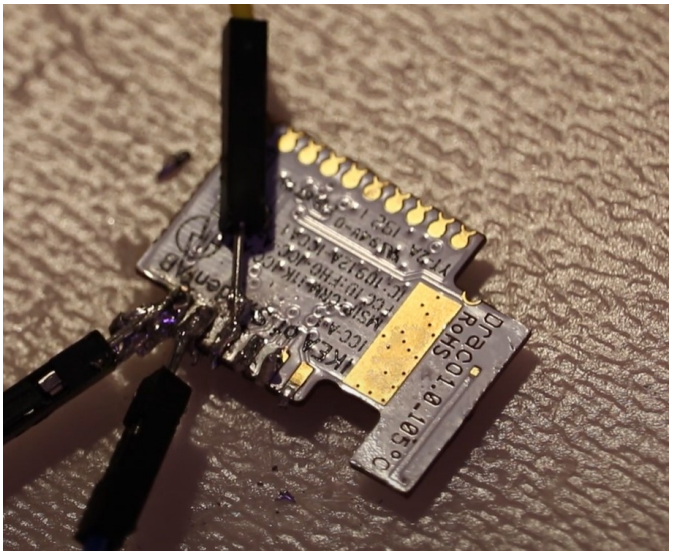


Abbildung 16: ZigBee Modul aus einem IKEA Tradfri Leuchtmittel [10]

Auch auf einer anderen Website (Quelle [11]) wurde sehr ähnlich vorgegangen. Hier wurde jedoch ein originales Philips Hue Leuchtmittel aufgebrochen. Das PCB wurde ausgemessen und der Autor hat einen 3.3 Volt PWM-Ausgang und einen Pin gefunden, welcher je nach Zustand in der App null oder 3.3 Volt hat. Hier wurde mit dem Signal direkt ein 230 Volt Relais mit einem Optokoppler angesteuert, um eine 230 Volt Leuchte ein- und auszuschalten.

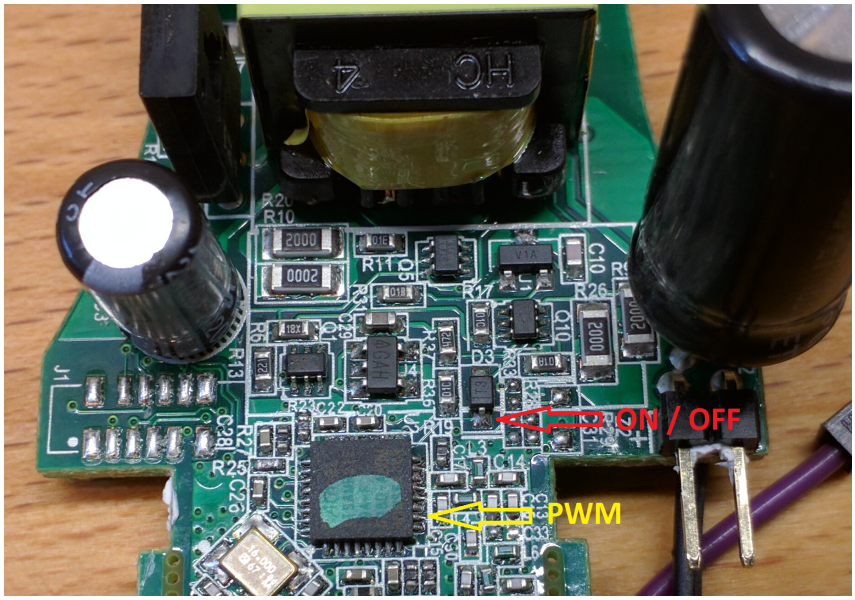


Abbildung 17: PCB eines originalen Philips Hue Leuchtmittels [11]

Für dieses Projekt könnte das ein / aus oder das PWM-Signal ebenfalls mit einem Mikrocontroller ausgewertet werden. Ist das PWM-Signal grösser als Null muss die Leuchte eingeschaltet werden und umgekehrt. Der Nachteil bei diesem Lösungsansatz liegt zum einen darin, dass das Leuchtmittel aufgebrochen werden muss. Es müsste rechtlich abgeklärt werden, ob dies erlaubt ist. Zum anderen kommt hinzu, dass der Zustand in der Philips Hue App vom Mikrocontroller aus nicht geändert werden kann. Dies ist jedoch nötig, da der Zustand in der App aktualisiert werden muss, wenn zum Beispiel jemand die Leuchte an den Bedientasten einschaltet.

4.2.3 ZigBee Light Link SDK

Bei der Recherche wurde auf der Website [12] ein Beitrag gefunden, in welchem ein eigenes Hue kompatibles Gerät gebaut wurde. Dazu wurde mit einem NXP-Programmer ein Binary auf einen JN5168 Mikrocontroller geflasht. Auf Grundlage des NXP Light Link Demoprojekts hat der Autor zwei Binaries erstellt, mit welchen die Steuerung einer monochromen dimmbaren Lichtquelle oder eines dimmbaren RGB-Lichts möglich waren. Die Binaries dazu stellt der Autor auf seiner Website zur Verfügung. Die Abbildung 18 zeigt den Versuchsaufbau aus der Quelle [12] zur Steuerung eines dimmbaren RGB Lichts.

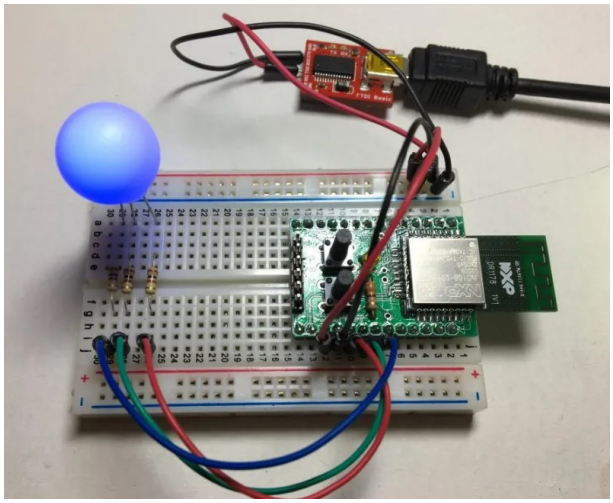


Abbildung 18: Testschaltung mit JN5168 Breakoutboard [12]

Wie bei der Lösungsmöglichkeit im Kapitel 4.2.2, fehlt auch hier die Möglichkeit, den Zustand in der App zu ändern. Dies könnte jedoch unter Umständen durch das Erstellen eines eigenen Binaries behoben werden. Ein weiteres Problem besteht darin, dass der JN5168 Mikrocontroller im Moment nicht verfügbar ist.

4.2.4 Hue Wallswitch zur Statusänderung in der App

In den vorherigen zwei Lösungsmöglichkeiten war das Problem, dass vom Mikrocontroller der Zustand in der App nicht verändert werden kann. Eine Lösung dafür wäre der Hue Wallswitch. Dabei handelt es sich um ein kleines batteriebetriebenes Gerät, welches hinter einem Wandschalter in einer bestehenden Elektroinstallation angebracht wird. Mit dem Wall Switch lassen sich Hue Lampen steuern. In diesem Projekt könnte der Wallswitch verwendet werden, um mit dem Mikrocontroller den Zustand der Leuchte in der App zu ändern.

Abbildung 19: Hue Wall Switch [13]

4.2.5 Verwendung eines Hue Smart Plugs

Eine weitere Option, ein eigenes Gerät in Philips Hue zu integrieren, wäre der Hue Smart Plug. Dabei handelt es sich um einen Zwischenstecker, welcher normalerweise zwischen Verbraucher und 230 Volt Steckdose gesteckt wird. Am Smart Plug befindet sich ein Taster, mit welchem der Verbraucher ein- und ausgeschaltet werden kann. Dies würde es ermöglichen, den Zustand in der App zu ändern. Der Smart Plug könnte aufgebrochen werden, damit das Ein- und Aussignal von der App dem Mikrocontroller zur Verfügung gestellt werden kann. Der Taster könnte mit dem Mikrocontroller betätigt werden, um den Zustand der Leuchte in der App zu ändern. Mit dem Smart Plug ist es jedoch nicht möglich, die Stehleuchte zu dimmen. Den Smart Plug einfach vor die Lampe zu stecken, ist keine Option, da die Lampe dann nur noch von der App aus ein- und ausgeschaltet werden kann. Denn wenn der Smart Plug ausgeschaltet ist, wird die Leuchte nicht mehr gespiesen und es können keine IR-Signale oder Befehle vom Bedienfeld verarbeitet werden. Wenn der Smart Plug aufgebrochen wird, müssten auch hier die rechtlichen Grundlagen geprüft werden. Die Abbildung 20 zeigt den Hue Smart Plug.

Abbildung 20: Hue Smart Plug [14]

4.2.6 Verwendung von Hue kompatiblen Schaltaktoren

Eine weitere Möglichkeit, ein eigenes Gerät in Philips Hue zu integrieren, sind Hue kompatible Schaltaktoren. Die Abbildung 21 zeigt ein Beispiel eines solchen Schaltaktors. Der Vorteil eines solchen Schaltaktors liegt darin, dass das Gerät nicht geöffnet werden muss, da der Ausgang und der Tastereingang des Aktors von aussen über Schraubklemmen erreichbar ist. Ein weiterer Vorteil ist, dass durch den Tastereingang eine Zustandsänderung der App möglich ist. Die Schaltaktoren sind mit AC oder DC-Speisung erhältlich.

Abbildung 21: Hue Schaltaktor mit 230 Volt AC Speisung [15]

4.3 Statuserkennung der Leuchte

In diesem Kapitel geht es darum, wie der Lampenstatus (an oder aus) eindeutig erkannt werden kann. Dies ist wichtig, damit der Zustand der Stehleuchte in der Hue App stets aktuell gehalten werden kann. Wird die Lampe mit der Fernbedienung, dem Bedienfeld oder durch den Bewegungsmelder gesteuert, muss diese Information in irgendeiner Weise zu dem Mikrocontroller gelangen, damit dieser die Information an die Hue App weiterleiten kann. Ausserdem soll überprüft werden können, dass wenn der uC das Signal zum Einschalten der Leuchte aussendet, die Leuchte auch wirklich angeschaltet wird.

4.3.1 Statuserkennung mit einem Lichtsensor

Bei diesem Ansatz geht es darum, den Lampenstatus über einen Lichtsensor zu erkennen. Eine einfache Variante wäre es, einen LDR (Light Dependent Resistor) zu verwenden. Bei einem LDR handelt es sich um einen Fotowiderstand, welcher mit zunehmenden Lichteinfall niederohmiger wird.
Der Test Statuserkennung durch LDR in Kapitel 7.1 hat ergeben, dass es sehr schwierig ist, den Unterschied zwischen Leuchte-An in der tiefsten Dimmstufe und Leuchte-Aus zu detektieren, da das Umgebungslicht einen erheblichen Einfluss auf die Messung hat. Ein weiterer Nachteil ist, dass der Lichtsensor direkt an der Leuchte angebracht werden muss, um den Zustand der Leuchte zu erkennen und dass der Widerstandswert des LDR mit einer Messschaltung zu einem für den Mikrocontroller verwertbaren Signal umgewandelt werden kann.

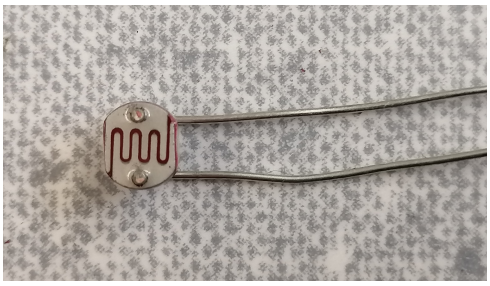


Abbildung 22: Light Dependent Resistor

Um eine eigene Messschaltung zu umgehen, kann auch ein erprobter Lichtsensor wie der TSL 2591 von Adafruit, verwendet werden (siehe Abbildung 23). Dieser kann über eine I2C Schnittstelle ausgelesen werden und kann Lichtstärken im Bereich von 188 Lux bis 88'000 Lux messen. Um das Problem mit dem Umgebungslicht zu minimieren, könnte mit zwei Sensoren gearbeitet werden. Wobei einer das Umgebungslicht plus das Licht der Leuchte misst und der andere nur das Umgebungslicht. Dies ermöglicht es, das Licht der Leuchte zu extrahieren. Aber auch bei dieser Variante muss ein Lichtsensor sehr nahe bei der Leuchte angebracht werden, was die Platzierung der H-A-IR-Remote einschränkt.

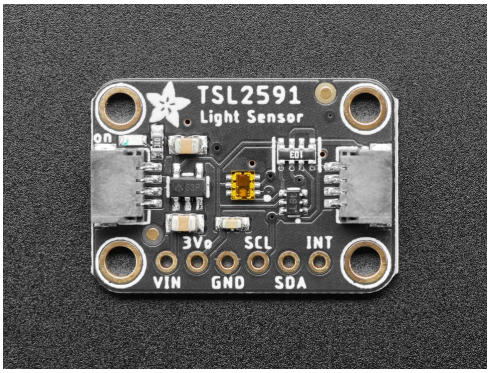


Abbildung 23: TSL 2591 I2C Lichtsensor [16]

4.3.2 Statuserkennung durch die Multianzeige der Leuchte

Die Multianzeige der Stehleuchte zeigt an, in welcher Lichtszene (orange) und in welcher Szenengruppe (grün) sich die Leuchte befindet. Ist die Leuchte aus, leuchtet kein LED der Multianzeige. Ein weiterer Ansatz, um den Status der Leuchte zu erkennen, wäre es, die zehn LEDs der Multianzeige mit 10 Photodetektoren abzutasten. Der Vorteil wäre, dass neben der Information, ob die Leuchte an oder aus ist, auch gleich die Information, in welcher Szene sich die Leuchte befindet, verfügbar wäre. Der Nachteil ist, dass die Statusleds durch die Detektoren verdeckt werden. Um die Detektoren abzutasten, müsste eine Abdeckung an die Leuchte geklammert / geklebt werden, verrutscht diese Abdeckung kann keine richtige Schlussfolgerung mehr aus den Messungen gezogen werden. Ausserdem belegen zehn Sensoren zehn GPIO Pins des Mikrocontrollers.

Abbildung 24: Multianzeige Gottardo Stehleuchte [1, S. 5]

4.3.3 Statuserkennung über Strommessung in der Zuleitung

Um den Status der Lampe zu erkennen, kann über den Strom, den die Leuchte bezieht, ein Rückschluss gezogen werden, ob die Leuchte im Stand-by-Modus aus oder an ist. Der Strom kann sowohl auf der 230-Volt-Seite des Netzgeräts als auch auf der 24-Volt-Seite des Netzgeräts gemessen werden. Dies ist möglich, da die Leuchte auf der 24-Volt-Seite durch einen Stecker mit dem Netzgerät verbunden ist. Um zu überprüfen, ob eine genügend grosse Differenz zwischen Stand-by-strom und Strom in der kleinsten Dimmstufe besteht, wurde eine Messung durchgeführt. Die Messergebnisse sind in Kapitel 7.2 aufgeführt. Die Messung hat gezeigt, dass auf der 24-Volt-Seite eine minimale Stromdifferenz von 30 mA und auf der 230-Volt-Seite von 36 mA zwischen Stand-by-Modus und tiefste Dimmstufe vorliegt. Diese Differenzen sind genug gross, um sie mit einem geeigneten Stromsensor zu messen.

4.3.4 Bildverarbeitung

Eine weitere Idee für die Statuserkennung ist es, durch Bildverarbeitung auf den Lampenstatus zu schliessen. Es gibt Systeme wie Open MV, die es ermöglichen, Bildverarbeitung mit einem Mikrocontroller durchzuführen. Der Vorteil an Bildverarbeitung wäre, dass sehr viele Informationen bereitgestellt werden können. Mann könnte die Kamera zum Beispiel auf die Multianzeige der Stehleuchte richten, um so auf den Zustand der Leuchte zu schliessen. Der Nachteil an Bildverarbeitung ist, dass es im Vergleich zu anderen Statuserkennungsmöglichkeiten sehr komplex und aufwendig ist und den Zeitrahmen dieser Arbeit sprengen würde.

Abbildung 25: OpenMV Camera [17]

4.3.5 Indirekte Statuserkennung durch bekannte Referenzpunkte

Beim bedienen der Leuchte fällt auf, dass die Leuchte neben dem Betätigen der Powertaste auch mit der Taste Hochdimmen anschalten lässt. Wird die Taste Hochdimmen betätigt, schaltet die Leuchte auch an und ist in der tiefsten Dimmstufe. Wird die Taste Hochdimmen erneut betätigt ist die Leuchte etwa auf 50 % der maximalen Helligkeit und wird die Taste Hochdimmen ein drittes Mal betätigt, ist die Leuchte auf der maximalen Lichtstufe. Mit der Taste Herunterdimmem funktioniert das ganze gleich, aber in die andere Richtung. Dies Bedeutet, dass die Leuchte nach dem dreimaligen Betätigen der Taste Herunterdimmen sicher aus ist. Durch diese Variante ist es jedoch nicht möglich, mit dem Mikrocontroller einen Statuswechsel der Stehleuchte beim Betätigen der Bedientasten zu erkennen. Somit kann nicht garantiert werden, dass der Zustand in der App immer dem Zustand der Stehleuchte entspricht.

4.4 Speisung der Hue Apple IR Remote

In diesem Kapitel wird aufgezeigt, welche Möglichkeiten es gibt, um die H-A-IR-Remote zu speisen und was die Vor- und Nachteile der jeweiligen Methoden sind.

4.4.1 Speisung über eine Batterie / einen Akku

Eine Möglichkeit, die H-A-IR-Remote zu speisen, wäre es, eine Batterie zu verwenden. Der Vorteil dabei ist, dass Kabel vermieden werden können. Der Nachteil ist, dass Batterien nach einer gewissen Zeit gewechselt werden müssen und dabei jedes Mal Abfall entsteht. Als Alternative dazu, könnte auch mit ein Akku verwendet werden. Somit würde kein Abfall anfallen, jedoch muss der Akku gelegentlich geladen werden. Um weniger oft laden zu müssen, wäre auch eine Kombination mit einer kleinen Solarzelle denkbar.

4.4.2 Kabelgebundene Speisung

Je nach Platzierung der H-A-IR-Remote ist es auch möglich, die Speisung über ein Kabel vorzunehmen. Das Gerät könnte zum Beispiel über ein USB Kabel oder sogar durch das Netzteil der Leuchte gespiesen werden. Der Vorteil ist, das so keine Batterie gewechselt oder kein Akku geladen werden muss.

4.5 Platzierung der Hue Apple IR Remote

In diesem Kapitel wird erarbeitet, wo die H-A-IR-Remote platziert werden kann, wie sie befestigt werden könnte und was die Vor- und Nachteile der jeweiligen Platzierungen sind.

4.5.1 Platzierung am Leuchtenkopf

Die H-A-IR-Remote könnte am Leuchtenkopf befestigt werden. Eine Möglichkeit wäre, es die H-A-IR-Remote wie eine Klammer aufzubauen, welche über die Multifunktionsanzeige geschoben wird und so gleich die Möglichkeit bietet, den Lampenstatus über diese auszuwerten. Es ist auch denkbar, die H-A-IR-Remote auf dem Leuchtenkopf anzubringen, wenn mit Lichtsensoren zur Lampenstatuserkennung gearbeitet wird. Zur sicheren Anbringung könnte mit Klettverschluss oder doppelseitigem Klebeband gearbeitet werden. Bei beiden Anwendungen wären bei einer kabelgebundenen Speisung die Zuleitung eher lang und müsste sauber hinter der Leuchte herunter geführt werden. Ausserdem verdecken beide Möglichkeiten einen Teil der Leuchte, was nicht der Anforderung entspricht, die Funktionalität der Stehleuchte nicht einzuschränken.

4.5.2 Platzierung am Ständer

Die Hardware könnte auch am Ständer der Leuchte angebracht werden. Somit wird nichts verdeckt und wenn die Hardware hinten am Ständer befestigt wird, fällt sie nicht gross auf. Als Befestigungsmöglichkeit käme auch hier ein Klammermechanismus, Klettverschluss oder doppelseitiges Klebeband zum Einsatz. Auch die Befestigung mit Gummibändern wäre denkbar. Da der Ständer nicht magnetisch ist, kann das Gehäuse nicht direkt mit Magneten befestigt werden. Aber es könnte eine Metallplatte am Ständer angebracht werden, an welcher die H-A-IR-Remote mit Magneten befestigt werden könnte.

4.5.3 Platzierung auf Lampenfuss

Als weiter möglicher Ort für die H-A-IR-Remote kommt eine Platzierung auf dem Leuchtenfuss infrage. Auf dem Sockel könnte die Hardware mit Magneten befestigt werden, da der Leuchtenfuss magnetisch ist.

4.5.4 Platzierung im Leuchtenkopf

Für dieses Projekt wurde der Leuchtenkopf geöffnet, um zu überprüfen, ob die H-A-IR-Remote auch im Lampenkopf verstaut werden könnte. Der Vorteil hiervon wäre, dass die Speisung direkt von innerhalb der Leuchte bezogen werden könnte und dass die H-A-IR-Remote von aussen nicht sichtbar wäre. Jedoch verletzt das Öffnen der Leuchte die Anforderung, dass nichts an der Leuchte verändert werden soll. Durch das Öffnen der Leuchte erlischt auch die Garantie, weshalb dies keine Option ist.

4.5.5 Platzierung losgelöst von der Leuchte

Die H-A-IR-Remote könnte auch losgelöst von der Leuchte platziert werden. Dies bedeutet, dass die H-A-IR-Remote zum Beispiel auf den Bürotisch gelegt werden kann. Die würde eine kabelgebundene Speisung vereinfachen (Verwendung eines USB Kabels). Bei dieser Variante ist zu beachten, dass es sich bei der IR-Kommunikation um ein gerichtetes Signal handelt und dass je nach Platzierung der H-A-IR-Remote IR-Befehle nicht bei der Leuchte ankommen.

4.6 Vorstellung Lösungskonzept Hardware

In diesem Abschnitt wird vorgestellt, aus welchen Teillösungen, das Lösungskonzept zusammengestellt wurde und wie die einzelnen Komponenten verknüpft werden. Anschliessend werden die einzelnen Komponenten vorgestellt.
Die Abbildung 26 zeigt das gesamte Hardware-Lösungskonzept als Blockdiagramm. Im oberen Teil des Bildes ist die Gottardo Stehleuchte mit allen Komponenten zu sehen. Im unteren Teil des Bildes ist die H-A-IR-Remote im gestrichelten Rahmen dargestellt. Die Hardware wird zwischen Netzteil und Leuchte gesteckt. Denn so kann die Hardware direkt über das Leuchtennetzteil gespiesen werden. Für die Leuchtenstatuserkennung misst die Hardware den Leuchtenstrom. Dies ist möglich, da der Leuchtenstrom durch die Hardware fliesst und so mit einem Stromsensor gemessen werden kann. Für die Einbindung in das Philips Hue System wird ein Hue kompatibler Schaltaktor ausgewählt. Die Hardware ist in der Lage, IR-Signale sowohl zu versenden als auch zu empfangen. Ein Raspberry Pi Pico Breakoutboard übernimmt die Steuerung der Hardware. Die Hardware wird am Ständer der Leuchte platziert und mit Klettbändern, welche die Hardware und den Ständer umschliessen, fixiert.

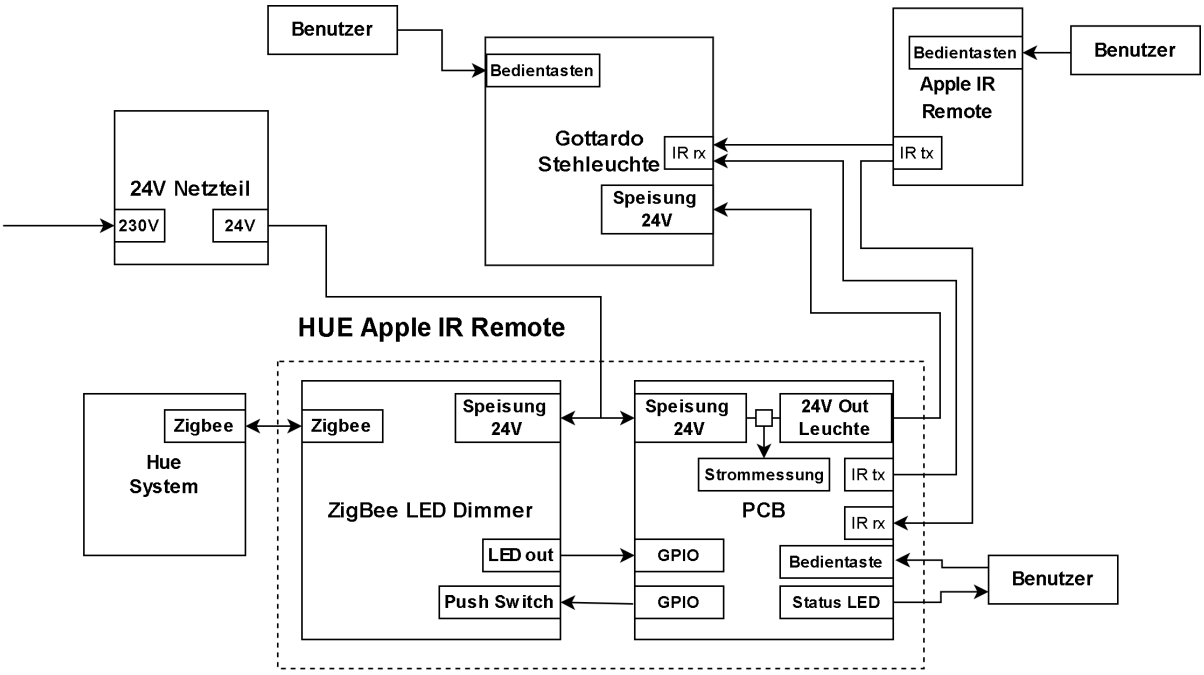


Abbildung 26: Lösungskonzept Hardware

4.7 Begründung Lösungskonzept

In diesem Kapitel wird erläutert, wie die Entscheidungen für die Teillösungen getroffen wurden und warum die Komponenten so ausgewählt wurden.

Einbindung in Phillips Hue mit ZigBee LED Dimmer
Für die Einbindung der H-A-IR-Remote wurde ein Hue tauglicher Schaltaktor gewählt (ZigBee LED Dimmer). Der ZigBee LED Dimmer wurde gewählt, weil so eine Einbindung in Philips Hue System ohne das Umbauen eines Geräts möglich ist. Ausserdem verfügt der LED Dimmer über einen Tastereingang, über welchen der Zustand in der App getoggelt werden kann. Es

ist somit mit nur einem Gerät möglich, den Zustand der Leuchte in der Hue App, dem Mikrocontroller zur Verfügung zu stellen und den Zustand in der App mit dem Mikrocontroller zu ändern. In Kapitel 7.3 wurden verschiedene Tests mit dem LED Dimmer durchgeführt, um zu prüfen, ob der ZigBee LED Dimmer für die Aufgabe geeignet ist. Ein weiterer Vorteil des LED Dimmers ist, dass dieser direkt über das Leuchtennetzteil gespiesen werden kann. Die technischen Daten des LED Dimmers sind in Kapitel 5.1.2 zu finden. Die Verwendung des ZigBee Light Link SDKs wurde verworfen, da zum einen der benötigte Mikrocontroller zurzeit nicht lieferbar ist und zum anderen der Zeitaufwand als zu hoch für den Zeitrahmen dieser Arbeit eingestuft worden ist.

Speisung über das 24V Leuchtennetzteil
Die Speisung über das Leuchtennetzteil wurde gewählt, da der LED Dimmer mit 12 - 36 Volt DC gespiesen werden muss. So bietet es sich natürlich sehr gut an, die Speisung des LED Dimmers und des Mikrocontrollers über das 24 Volt Netzteil der Leuchte vorzunehmen. Diese Variante bietet den Vorteil, dass keine Batterien getauscht oder Akkus geladen werden müssen. Das Zuleitungskabel kann durch eine geschickte Platzierung der Hardware versteckt werden.

Statuserkennung durch Strommessung mit INA239
Die Statuserkennung der Leuchte über die Strommessung wurde gewählt, da anhand des Stromverbrauches der Leuchte zuverlässig erkennbar ist, ob die Leuchte an oder aus ist (siehe Kapitel 7.2). Als Stromsensor wurde der INA239 gewählt, da dieser mit 16 Bit eine genügend feine Auflösung besitzt, um den Unterschied zwischen Leuchte an in der tiefsten Dimmstufe und Leuchte aus zu detektiern. Dies wurde auch durch einen Test bestätigt (siehe Kapitel 7.4.1). Dadurch, dass die Hardware über das Leuchten Netzteil gespiesen wird, kann die Hardware so ausgelegt werden, dass der Leuchtenstrom durch die Hardware fliesst, was eine Strommessung ermöglicht. Die Variante mit Lichtsensoren wurde aufgrund der höheren Komplexität und der kleineren Erfolgsgarantie verworfen. Ausserdem gestaltet sich die Platzierung der Hardware viel flexibler, da kein Sensor nahe am Leuchtenkopf platziert werden muss. Die Variante mit Bildverarbeitung auf den Lampenstatus zu schliessen wurde wegen des schwer einschätzbaren Zeitaufwands verworfen.

Platzierung der Hardware am Leuchten-Ständer
Die Platzierung der Hardware am Leuchtenständer wurde gewählt, da die Hardware hier unauffällig angebracht werden kann. Die Kabel können hinter dem Ständer zum Boden geführt werden und stören so die Optik der Leuchte nicht.

Verwendung des Raspberry Pi Pico Breakoutboards
Als Mikrocontroller wurde das Raspberry Pi Pico Breakoutboard gewählt. Der Grund dafür ist, dass das Pico im Moment lieferbar ist und eine sehr gute Dokumentation seitens des Herstellers verfügbar ist. Ausserdem wurden schon andere Arbeiten an dieser Schule mit diesem Mikrocontroller durchgeführt. Diese Arbeiten könnten bei Problemen konsultiert werden. Mit 26 GPIO Pins hat der Mikrocontroller genügend Ein- und Ausgänge für dieses Projekt. Es wird die Breakoutboard-Variante verwendet, da es keine Maximalgrösse für die Hardware definiert wurde. Weitere technische Daten sind in Kapitel 5.1.1 zu finden.

5 Erarbeitung und Beschreibung der Hardware

In diesem Kapitel wird beschrieben, wir die zwei Prototypen der H-A-IR-Remote entwickelt wurden.

5.1 Beschreibung Komponenten

In diesem Kapitel werden die wichtigsten verwendeten Komponenten mit den relevanten technischen Daten vorgestellt. Sämtliche Datenblätter der in diesem Kapitel aufgeführten Bauteile sind im elektronischen Anhang zu finden.

5.1.1 Raspberry Pi Pico Breakoutboard

Technische Daten
Versorgungsspannung: 1.8 - 5.5 VDC
Mikrocontroller: RP 2040 (Dual-Core-ARM M0+)
Speicher: 264 kB Ram + 2 MB Flash
Anzahl GPIO Pins: 26
ADC GPIO Pins: 3
Digitale Pheriferie: 2xSPI, 2xI2C, 2xUART, 12bit-ADC, 16xPWM-Channels
Debugging: 3 Pin ARM SWD Anschluss
Grösse (LxW): 51x21 mm

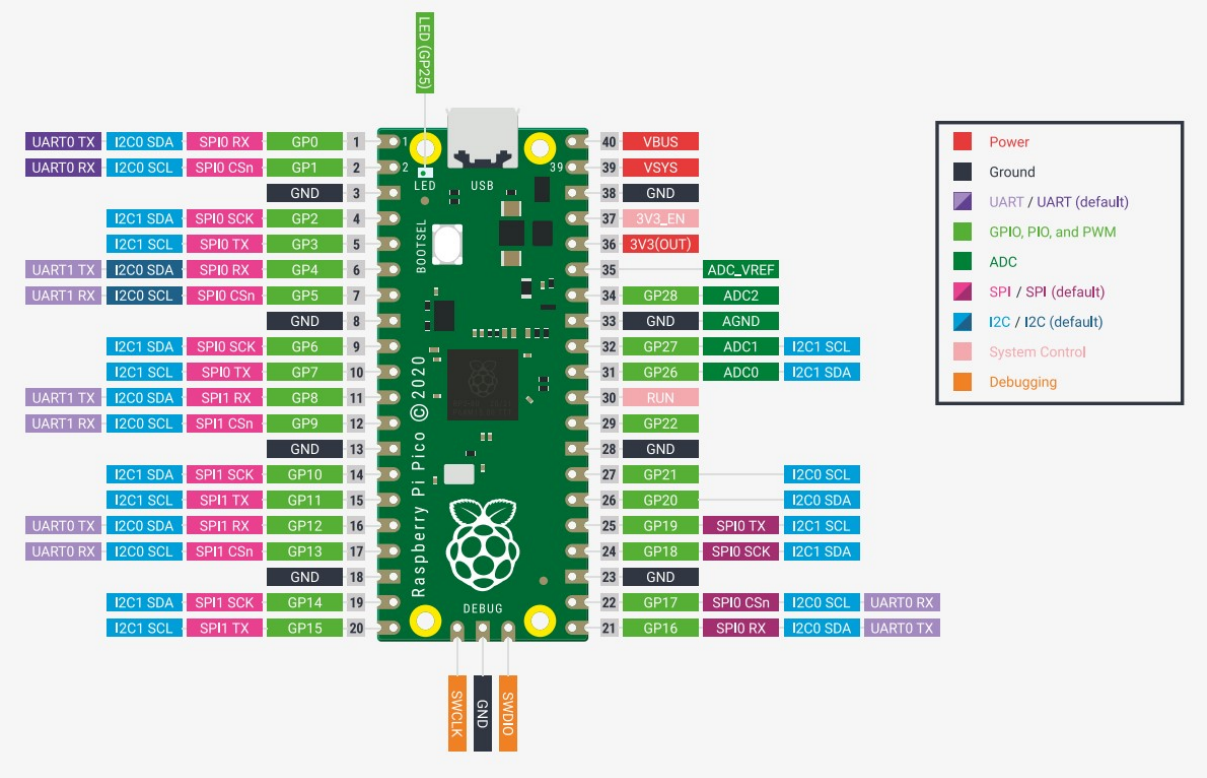


Abbildung 27: Raspberry Pi Pico Pinout [18]

5.1.2 ZigBee LED Dimmer SR-ZG9101CS

Technische Daten
Eingangsspannung: 12 - 36 VDC
Max. Ausgangsstrom: 8 A (Constant Voltage)
Max. Ausgangsleistung: 96 - 288 W
Grösse (LxWxH): 95x37x20 mm

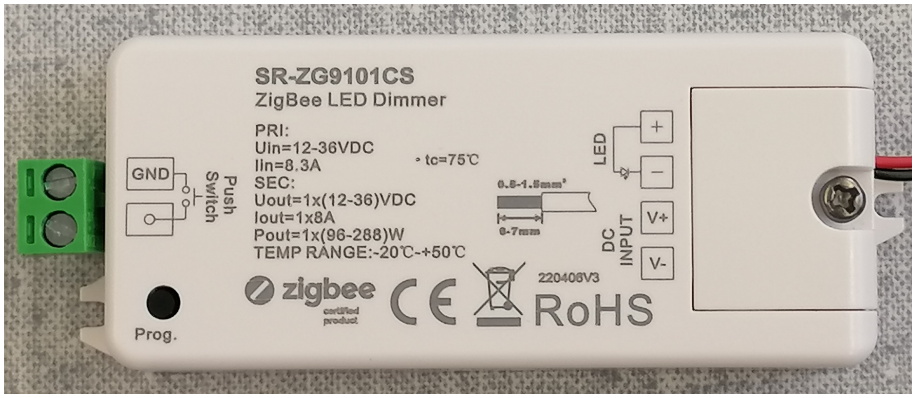


Abbildung 28: ZigBee LED Dimmer

5.1.3 Stromsensor INA239

Technische Daten
Versorgungsspannung: 2.7 - 5.5 V
Datenübertragung: 10 MHz SPI Interface
Max. Messpannung: 85 VDC
Max. Stromessung: durch externen Shunt Widerstand auf 5 A konfiguriert
Auflösung: 16-Bit (0.2 mA)
Grösse (LxW): 3x3 mm
Montageart: SMD VSSOP (10)

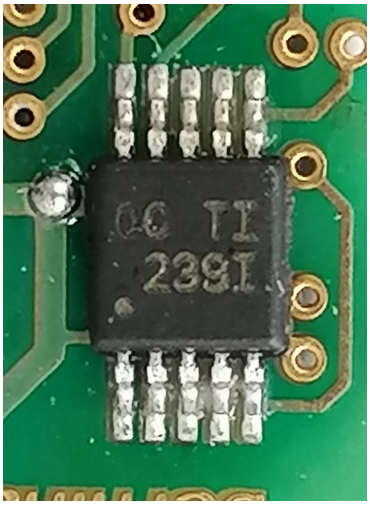


Abbildung 29: INA239 Stromsensor

5.1.4 Stromsensor INA260

Technische Daten
Versorgungsspannung: 2.7 - 5.5 V
Datenübertragung: I2C Interface
Max. Messspannung: 36 VDC
Max. Strommessung: 15 A
Integrierter Shunt Widerstand: 2 mOhm
Auflösung: 16-Bit (1.5 mA)
Montageart: THT (Breakoutboard)
Grösse (LxWxH): 22.9x22.8x2.7 mm

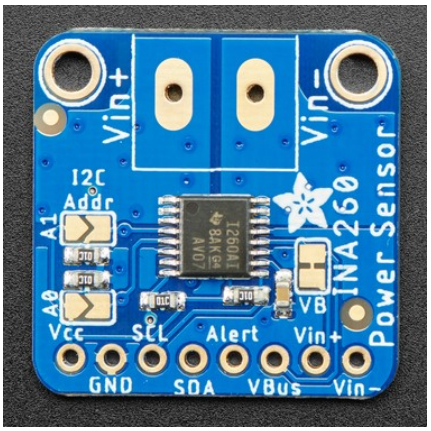


Abbildung 30: INA260 Stromsensor [19]

5.1.5 DC-DC-Wandler TSR 1-2450

Technische Daten
Eingangsspannung: 6.5 - 36 VDC
Ausgangsspannung: 5 VDC
Max. Ausgangsstrom: 1 A
Effizienz: 94 % (bei Vin min)
Montageart: THT
Grösse (LxWxH): 11.7x7.5x10.1 mm

Abbildung 31: DC-DC-Wandler TSR 1-2450

5.1.6 IR Diode SFH 4545

Technische Daten
Spitzenwellenlänge: 950 nm
Max. Vorwärtsstrom: 100 mA
Max Vorwärtsstrom gepulst: 1 A (tp < 400 us, Dutycycle < 0.005)
Durchlassspannung: 1.5 - 1.7 V
Durchlassspannung bei 1 A gepulst: 2.3 - 2.9 V
Sperrspannung: 5 V
Montageart: THT
Durchmesser: 5 mm
Höhe: 8.3 mm

Abbildung 32: IR Diode SFH 4545

5.1.7 IR Diode APDA3020F3C-P22

Technische Daten
Spitzenwellenlänge: 940 nm
Max. Vorwärtsstrom: 50 mA
Max Vorwärtsstrom gepulst: 1.2 A (tp < 10 us, Dutycycle<0.01)
Durchlassspannung: 1.2 - 1.6 V
Grösse (LxBxH): 3x2x2.8 mm
Montageart: SMD, seitliche Montage

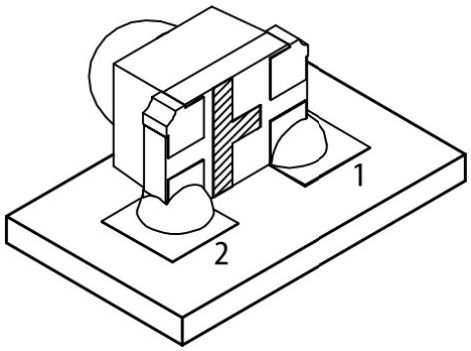


Abbildung 33: IR Diode APDA3020F3C-P22 [20]

5.1.8 IR-Receiver TSOP38238

Technische Daten
Versorgungsspannung: 2.5 - 5.5 V
Trägerfrequenz: 38 kHz
Peak Sensitivity Wavelength: 950 nm
Typ. Übertragungsdistanz: 24 m
Grösse (WxHxD): 5x6.95x4.8 mm

Abbildung 34: IR-Receiver TSOP38238

5.1.9 RGB-LED - R6GHBHW

Technische Daten
Max. Vorwärtsstrom(R,G,B): 25 mA
Typ. Vorwärtsstrom(R,G,B): 5 mA
Typ. Durchlassspannung (R,G,B); 1.85, 2.9 ,2.9 V
Montageart: SMD, seitliche Montage
Grösse (LxBxH): 2.7x1.35x0.5 mm

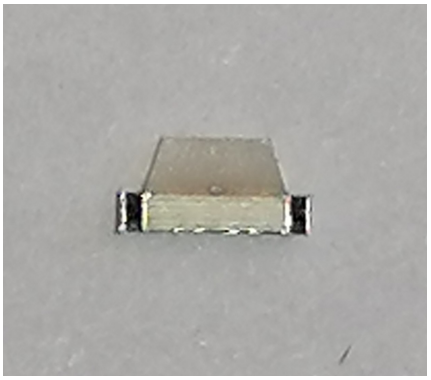


Abbildung 35: RGB LED - R6GHBHW

5.2 Erklärung PCB und Workflow in KiCad

In den nächsten Kapiteln wird beschrieben, wie das Printed Circuit Board (PCB) erstellt wurde. Bei einem PCB handelt es sich um eine Platine, auf welcher alle Komponenten angelötet werden und dann durch die Platine gemäss einem Schaltplan korrekt verbunden sind. Um diesen komplexen Prozess zu vereinfachen, wird in diesem Projekt mit KiCad V6.0 gearbeitet. Im Folgenden wird eine kurze Übersicht gegeben, wie der Arbeitsablauf mit KiCad funktioniert. Als Erstes wird mit dem Schaltplaneditor der Schaltplan gezeichnet. Anschliessend werden alle Bauteile im Schaltplan annotiert, sodass jedes Bauteil eine eigene Nummer hat. Danach wird jedem Bauteil der entsprechende Footprint zugewiesen. Bei einem Footprint handelt es sich um die Bauteillandepunkte auf der Leiterplatte. Generell werden zwei veschiedene Footprintarten unterschieden, Surface Mounted Device (SMD) und Through Hole Technology (THT) Footprints. Ist jedem Bauteil der richtige Footprint zugewiesen, wird in den Leiterplatteneditor gewechselt. Dort wird zuerst die grösse des PCBs, durch das Zeichnen des Umrisses definiert (Edge Cuts). Anschliessend werden die Footprints aller Bauteile auf dem PCB platziert. Zuletzt werden die Bauteile mit Leiterbahnen verbunden, hierbei hilft KiCad, indem es gemäss Schaltplan anzeigt, wie die Bauteile miteinander verbunden werden müssen. Schlussendlich können mit KiCad Fertigungsdaten erzeugt werden, mit welchen das PCB produziert werden kann. Die Abbildung 36 zeigt die Benutzeroberfläche und Werkzeuge von KiCad.

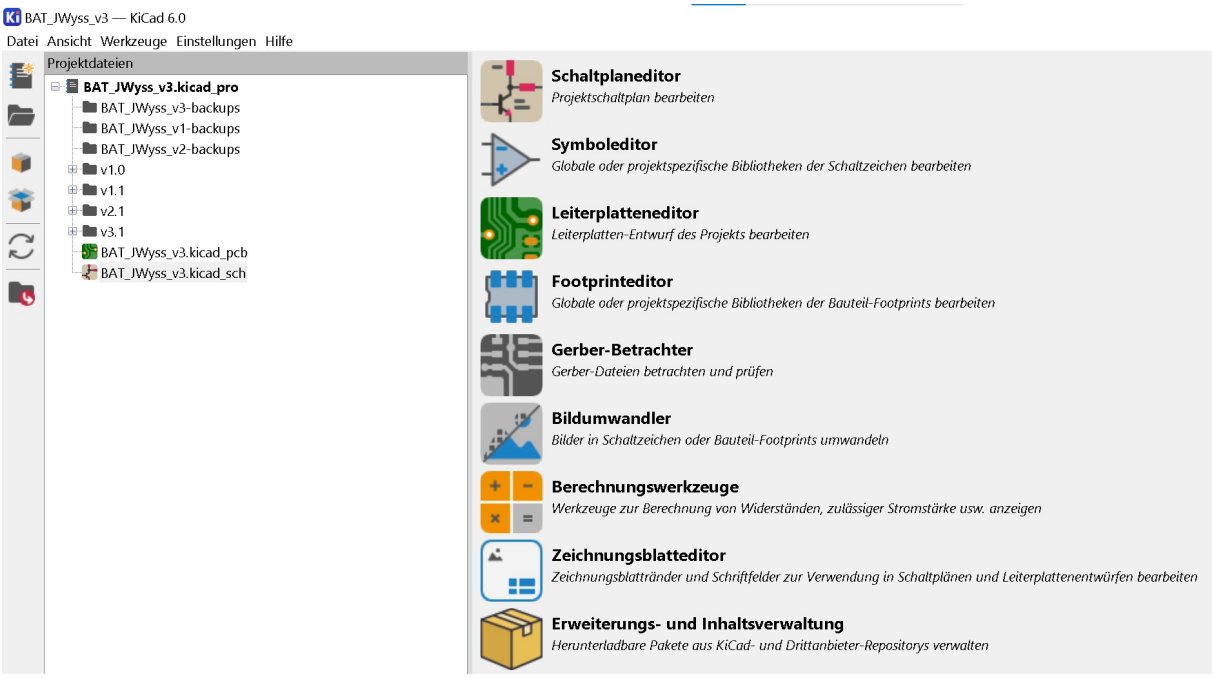


Abbildung 36: KiCad Benutzeroberfläche

5.3 Schaltplan PCB V1.1

In diesem Kapitel wird der Schaltplan für die erste Version des Prototypen detailliert beschrieben. Das PCB V1.0 wurde in einem ersten PCB Review mit dem betreuenden Dozenten Erich Styger besprochen (siehe Kapitel 5.4.5). Die PCB V1.1 beinhaltet schon einige der besprochenen Punkte (siehe Kapitel 5.4.5). Der gesamte Schaltplan des PCB V1.1 ist in Anhang A.3 zu finden.

5.3.1 Mikrocontroller

Die Abbildung 37 zeigt das Symbol des Raspberry Pi Picos. Es ist ersichtlich, an welchen Pins die verschiedenen Hardwarebauteile angeschlossen sind. Um die Übersichtlichkeit zu verbessern, wurde mit globalen Bezeichnern gearbeitet. Da viele Pins unbenutzt sind, wurde ein 11-poliger Pin-Header als Reserve vorgesehen. Dieser kann benutzt werden, falls es eine grössere Änderung für die V2 gibt, welche getestet werden muss.

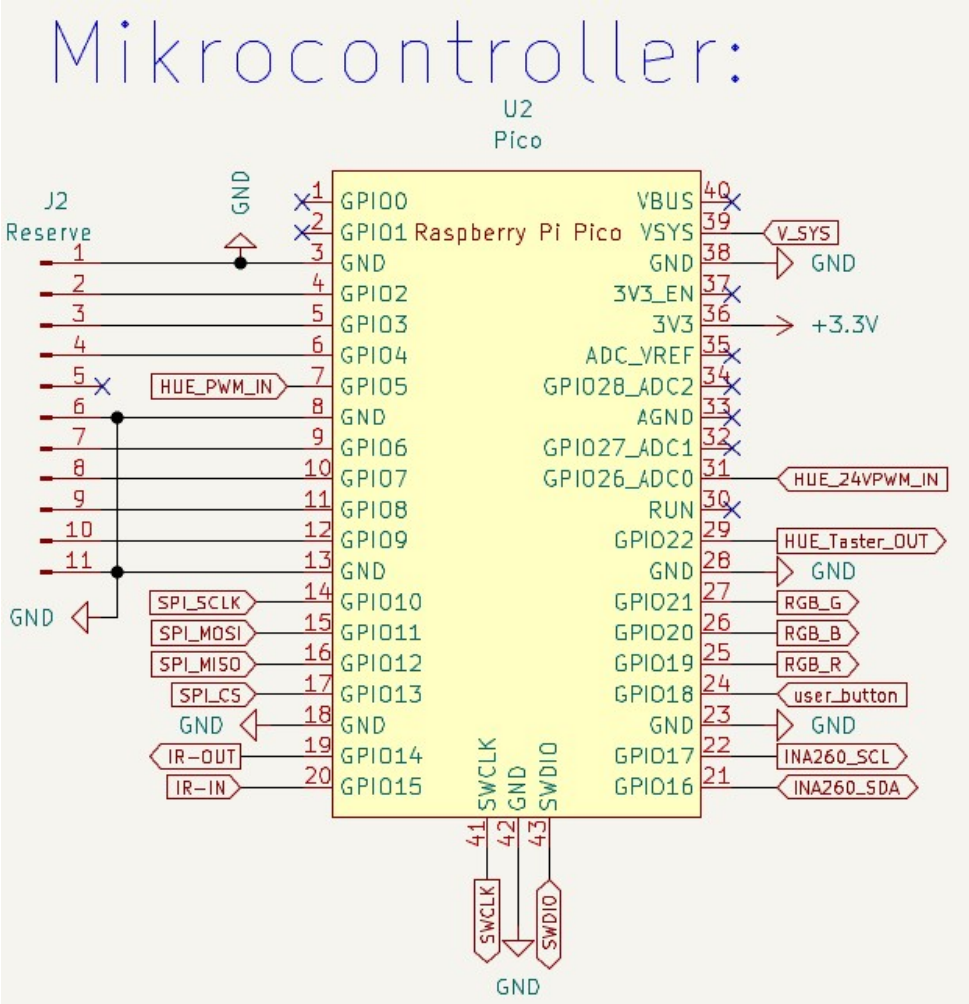


Abbildung 37: Raspberry Pi Pico

5.3.2 Debug Schnittstelle

Um die Mikrocontrollerschaltung zu Debuggen wird ein Segger J-Link Edu mit einer 20 Pin Interface Connection verwendet. Da 20-polige Schnittstelle viel Platz braucht und nur die Pins Vcc, SWDIO, SWCLK und GND verwendet werden, wird ein Cortex SWD Adapter von Adafruit verwendet. Dadurch kann ein 2x5 poliger Pin-Header mit 1.27 mm Lochabstand verwendet werden. Dieser wird als SMD Bauteil bestückt,um den Aufwand bei bestücken zu reduzieren. Die Vcc Leitung wird mit dem 3.3 Volt Spannungsausgang des Picos verbunden, somit kann der Debugger überprüfen, ob das Target mit Strom versorgt ist. Die Abbildung 38 zeigt das Symbol des 10-poligen Pin-Headers.

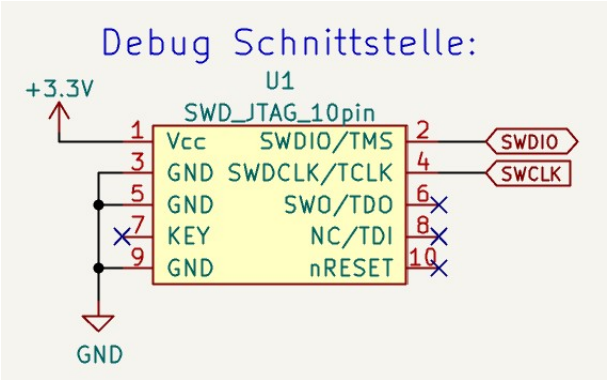


Abbildung 38: Debug Schnittstelle

5.3.3 Speisung Pico

Das Pico wird durch einen DC-DC-Wandler am 5-Volt-Eingang (V_SYS) gespiesen. Der DC-DC-Wandler wird gebraucht, um aus den 24 Volt vom Netzteil die 5 Volt zu machen. Eine Schottkeydiode sorgt dafür, dass bei einer gleichzeitigen Speisung über USB und das Stehleuchten Netzteil die beiden Quellen nicht gegeneinander Arbeiten können. Dies ist zum Beispiel der Fall, wenn die H-A-IR-Remote im Einsatz ist und über die Mikro-USB-Schnittstelle des Picos ein neues Programm auf das Pico geladen werden soll. Durch den grossen Eingangsspannungsbereich des DC-DC-Wandlers (6.5 - 36 Volt) ist die Mikrocontrollerschaltung flexibel, was die Speisespannung angeht. Die Abbildung 39 zeigt den DC-DC-Wandler in Schaltplan.

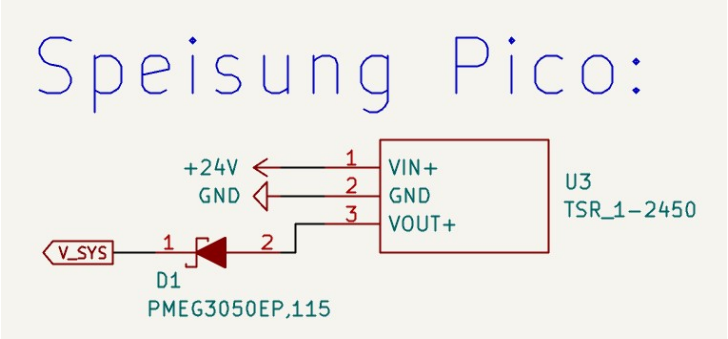


Abbildung 39: Speisung Pico

5.3.4 Strommessung

In Kapitel 4.7 wurde der INA239 zur Strommessung ausgewählt. Auf der linken Seite in der Abbildung 40 ist ersichtlich, wie der INA239 angeschlossen wird. Falls die Kommunikation mit dem INA239 über SPI nicht klappen sollte, wird als Reserve ein Header für ein INA260 Breakoutboard vorgesehen (siehe Abbildung 40 rechts). Das Problem bei dem INA260 ist, dass nur ein einziges Breakoutboard zur Verfügung steht und aufgrund der Chipkrise keine weiteren bestellt werden können. Deshalb ist diese Variante nur eine Notfalllösung, die Auslesung über den INA239 wird bevorzugt.

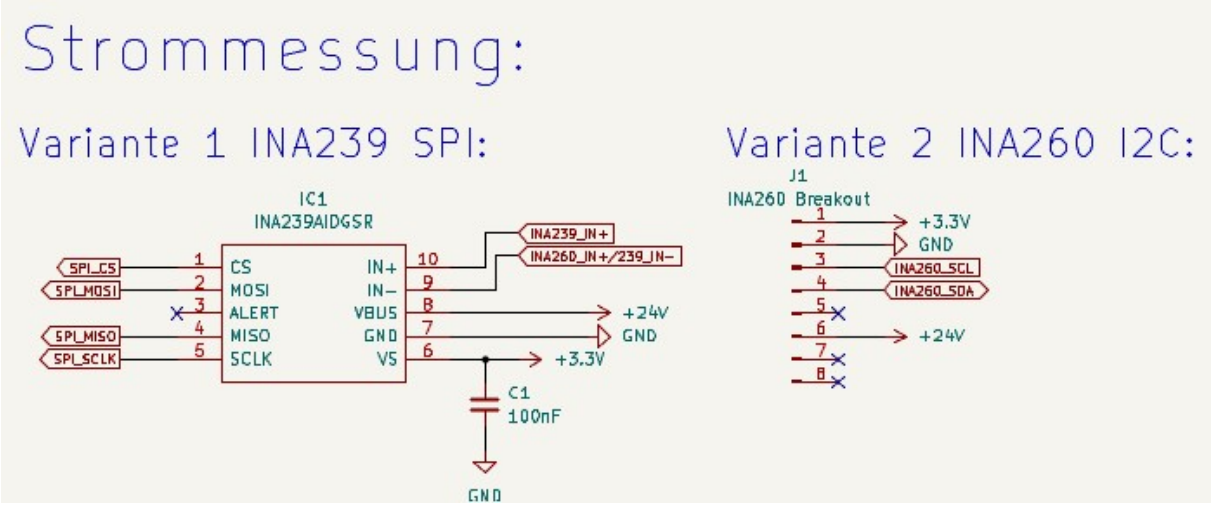


Abbildung 40: Strommessung

5.3.5 Anschlussklemme

An der Anschlussklemme werden später das Zuleitungskabel (24V IN) und das Kabel für die Stehleuchte (24V OUT) angeschlossen. Das Zuleitungskabel versorgt das PCB mit Spannung. Für die Strommessung mit dem INA239 fliesst der Leuchtenstrom durch einen 30 mOhm Widerstand (siehe Kapitel 6.6). Da bei dem INA260 Stromsensor der Shuntwiderstand intern verbaut ist, kann die Klemme I SENS benutzt werden, um den Strom zum INA260 Breakoutboard und zurück zuführen. Wenn der INA260 nicht benutzt wird, muss eine Drahtbrücke eingesetzt werden. Die Abbildung 41 zeigt den Schaltplan der Anschlussklemme.

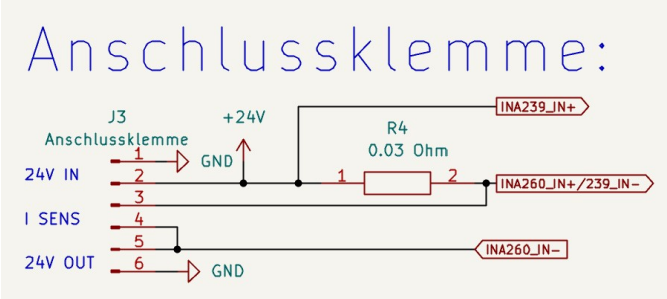


Abbildung 41: Anschlussklemme

5.3.6 Kommunikation mit ZigBee LED Dimmer

An der Klemme J6 kann der Tastereingang des LED Dimmers angeschlossen werden. Um den Tastereingang mit dem Pico zu betätigen, wird ein N-Mosfet zwischen die Kontakte geschalten. Wenn nun der GPIO Pin HUE_Taser_OUT auf high und anschliessend wieder auf low gesetzt wird, schaltet der LED Dimmer und der Zustand in der App wird automatisch aktualisiert. Diese Schaltung durch einen Test ermittelt (siehe Kapitel 7.3.5). Dieser Test wurde mit einem THT N-Mosfet durchgeführt. In der finalen Version soll jedoch ein SMD N-Mosfet zu Einsatz kommen. Aus diesem Grund befinden sich beide Footprints auf dem PCB V1.1.
Um das Ausgangssignal des LED Dimmers zu messen, wurden in Kapitel 7.3.7 verschiedene Varianten erarbeitet und gestestet. Bei einer Variante wird der LED Dimmer geöffnet und das 3.3 Volt PWM-Signal herausgeführt, welches der Dimmstufe in der App entspricht. Diese Variante funktioniert einwandfrei. Der Nachteil ist jedoch, dass der LED Dimmer aufgebrochen werden muss. Dennoch wird diese Variante für die erste Version des PCBs benutzt. Bei der zweiten Variante wird der reguläre Ausgang des LED Dimmers benutzt und mit einem Spannungsteiler und einem Tiefpassfilter aufbereitet, um das Signal mit einem ADC des Picos zu messen. Diese Variante vermeidet das Öffnen des LED Dimmers, ist jedoch nur als Reserve geplant und wurde dementsprechend nicht getestet. Um den LED Dimmer über das PCB mit 24 Volt DC zu speisen, wird die Klemme J4 vorgesehen. Die Abbildung 42 zeigt den Schaltplan zur Kommunikation mit dem LED Dimmer.

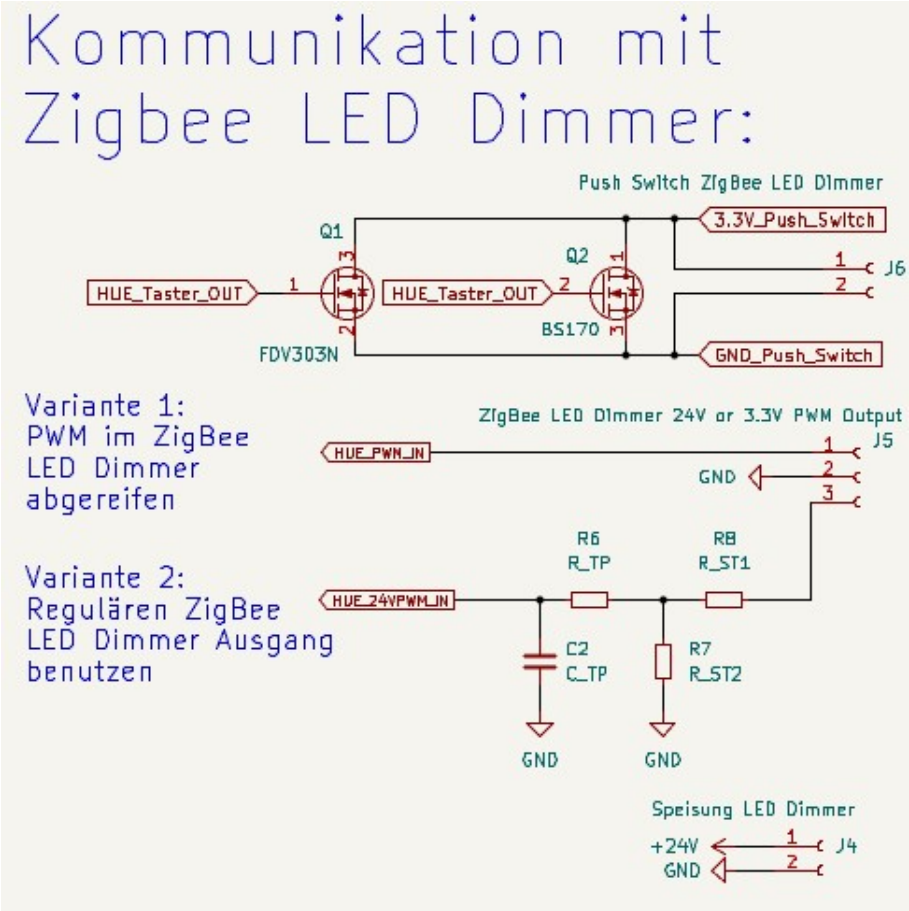


Abbildung 42: Kommunikation mit ZigBee LED Dimmer

5.3.7 IR rx / tx

Bei der Recherche hat sich herausgestellt, dass die Apple IR Remote das NEC-Protokoll verwendet, um IR Signale zu versenden (siehe Kapitel 3.1.2). In Kapitel 3.2 wurde herausgefunden, wie IR Signale in NEC-Format der Apple IR Remote mit dem Pico empfangen und versendet werden können und wie die Schaltung dazu aufgebaut werden muss. Diese Schaltung wurde für dieses Projekt übernommen (siehe Abbildung 43). Es wurden zwei verschiedene IR Dioden vorgesehen, damit die Hardware später zum Testen auf den Tisch gelegt werden oder auch in einem Gehäuse am Leuchtenständer montiert werden kann. In beiden Fällen soll einer der beiden IR Emitter richtig ausgerichtet sein. Die Vorwiderstände wurden so berechnet, dass beide IR Dioden mit dem maximalen kontinuierlichen Vorwärtsstrom betrieben werden (APA 50 mA, SFH 100mA). Zum Empfangen von IR Signalen wird ein TSOP38338 IR Receiver verwendet. Dieser stellt dem Mikrocontroller am Ausgang gleich das demodulierte IR Signal zur Verfügung.

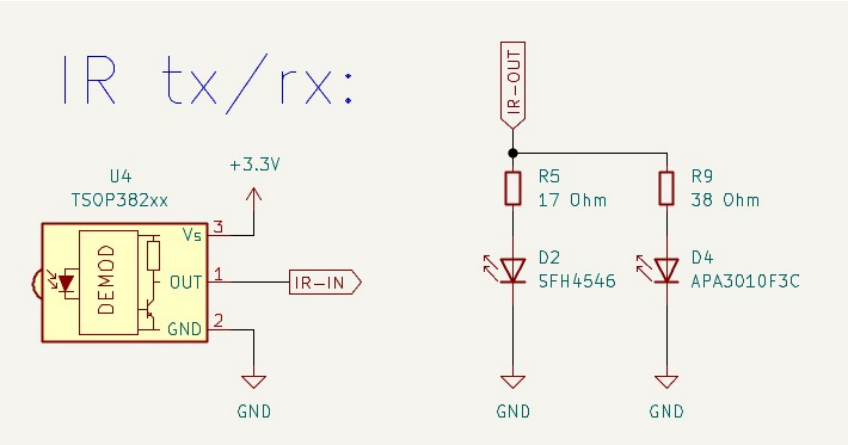


Abbildung 43: IR rx / tx

5.3.8 User Interface

Beim User Interface geht es darum, dass der Benutzer mit der Hardware interagieren kann. Dazu werden eine RGB-LED und ein Taster vorgesehen (siehe Abbildung 44). Auch beim Debuggen der Software ist es hilfreich, wenn über die RGB LED Ausgaben gemacht werden können. Wie genau der das User Interface am Schluss verwendet wird, wird erst beim Schreiben der Software klar definiert. Die Vorwiderstände wurden so berechnet, dass die LEDs einen Vorwärtsstrom von 5 mA aufnehmen. Dies ist gemäss Datenblatt der typische Vorwärtsstrom.

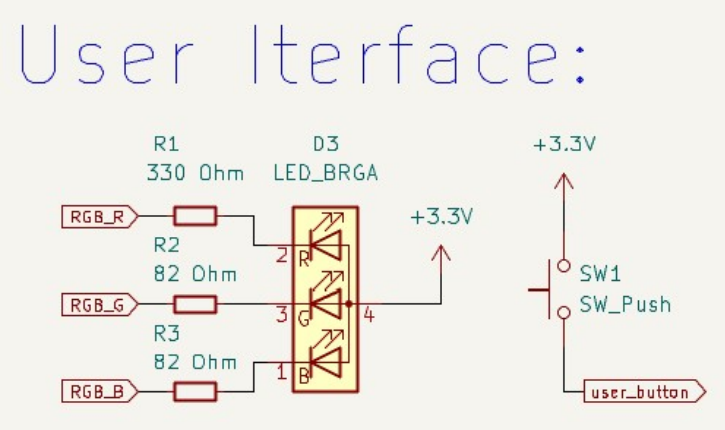


Abbildung 44: User Interface

5.3.9 Montagelöcher

In KiCAD werden die Löcher bereits im Schaltplan eingefügt, damit diese dann später wie normale Bauteile auf dem PCB platziert werden können. Es werden zwei Löcher zur Montage des LED Dimmers und zwei Löcher zur Befestigung des PCB im Gehäuse vorgesehen. Die Abbildung 45 zeigt, wie die Löcher im Schaltplan definiert werden.

Abbildung 45: Montagelöcher

5.4 PCB Version 1.1

Die Abbildungen 46 und 47 zeigen die Vorder- und Rückseite des PCB V1.1. Das PCB besteht aus zwei Kupferflächen (35 um), welche durch ein FR4-Substrat getrennt sind. Die Dimensionen des PCBs betragen 135 x 38 x 1.6 mm. In den nachfolgenden Kapiteln wird beschrieben, wie beim designen des PCBs vorgegangen wurde und was es zu beachten gab.

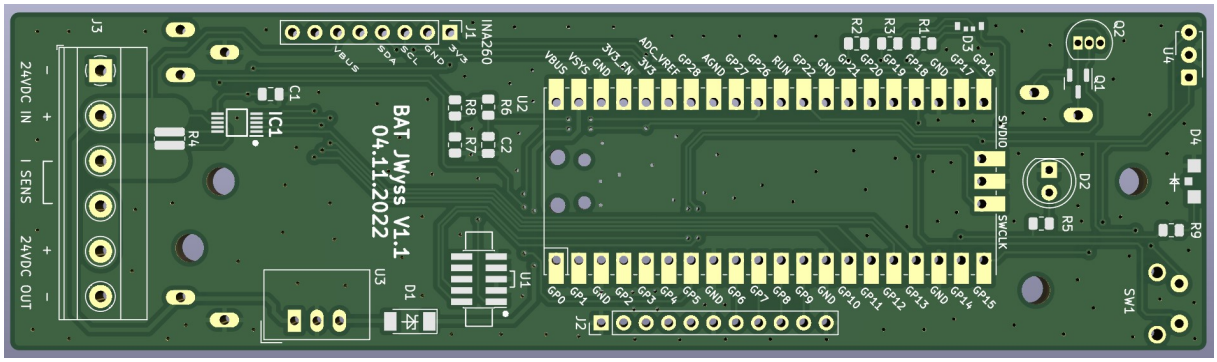


Abbildung 46: Vorderseite PCB V1.1

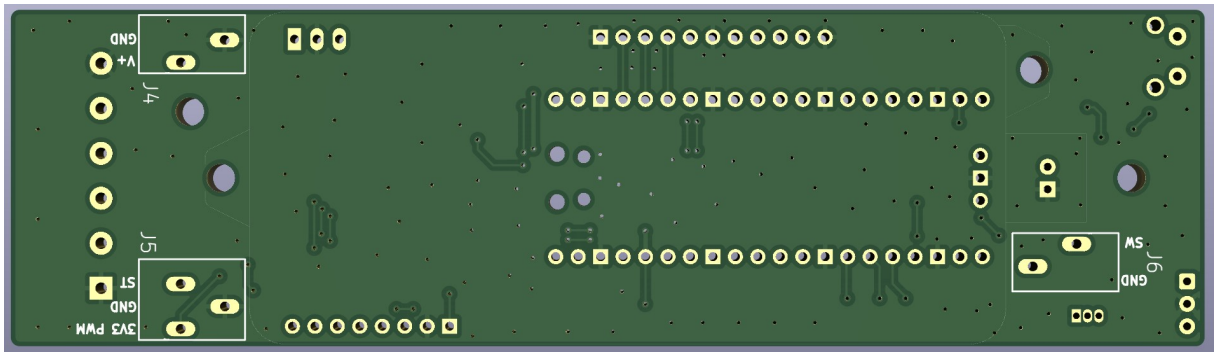


Abbildung 47: Rückseite PCB V1.1

5.4.1 Festlegung der Grösse des PCB V1.1

In einem ersten Schritt wurden die Grösse des PCBs festgelegt. Die Breite des PCBs wurde der Breite des LED Dimmers angepasst. Die Länge wurde so gewählt, dass der LED Dimmer und die Wire-to-Board Klemmen für den LED Dimmer Platz haben und noch genügend Platz zum anschliessen der Kabel vorhanden ist.

5.4.2 Komponentenplatzierung

Im nächsten Schritt mussten alle Komponenten in Form der Footprints der jeweiligen Komponenten auf dem PCB platziert werden. Die Anschlussklemme für Zuleitungs- und Leuchtenkabel wurden an einem der beiden schmalen Enden des PCBs platziert. Auf der gegenüberliegenden schmalen Seite wurde einer der IR Dioden platziert. Somit zeigt diese bei der Montage des Prototypen am Leuchtenständer nach oben. Ein kurzer Test mit der Apple IR Remote hat gezeigt, dass das IR Signal aus dieser Position sicher bei der Leuchte ankommt.

Auch der Taster und der IR Receiver wurden auf der gleichen Seite wie die IR Diode platziert. Die Überlegung dabei war, dass diese für den Benutzer gut erreichbar sind. Der zweite IR Emitter wurde auf der Oberseite des PCBs platziert, somit kann die Hardware zum Testen auf den Tisch gelegt werden. Das Raspberry Pi Pico wurde mittig auf der Vorderseite des PCBs platziert. Dabei wurde darauf geachtet, dass der USB Anschluss benutzbar bleibt, denn dieser wird für Software-Updates benötigt. Der LED Dimmer und die Anschlussklemmen des LED Dimmers, werden auf der Rückseite des PCBs angebracht. Die restlichen Komponenten wurden auf der Vorderseite des PCBs platziert. Dabei wurde darauf geachtet, dass die Leitungen möglichst kurz sind und nicht zu viele Kreuzungen entstehen.

5.4.3 Routing

Im letzten Schritt des PCB designens wurden alle Bauteile gemäss dem Schaltplan durch Leiterbahnen verbunden. Dazu wurde in einem ersten Schritt die Ober- und die Unterseite des PCBs als Grounplane definiert. Dann wurden die Bauteile miteinander verbunden. Dabei wurde darauf geachtet, dass die Groundplane auf der Rückseite des PCBs möglichst intakt bleibt. Um dies zu erreichen, wurden Leiterbahnen auf der Rückseite nur verwendet, um Leiterbahnen auszukreuzen. Am Schluss wurden die beiden Groundplanes durch viele Vias sauber miteinander verbunden (Ground-Stitching).

5.4.4 Leiterbahnbreiten

Im Folgenden ist aufgeführt, wie die Leiterbahnbreiten gewählt wurden.

- Signalleitungen: 0.25 mm
- 5V Speisung Pico: 0.4 mm
- Leiterbahn für Leuchtenstrom: 4 mm

Für die Signalleitungen wurde der Default Wert von KiCAD verwendet. Die Speiseleitung für das Pico wurde etwas breiter gewählt, um einem Spannungsabfall entgegenzuwirken. Die Leiterbahn für den Leuchtenstrom wurde gemäss Anhang A.6 für eine Strombelastung von 5 A mit einem Sicherheitsfaktor von zwei ausgelegt.

5.4.5 PCB Review 1

Nach dem Erstellen des PCB V1.0 wurde diese mit dem betreuenden Dozenten Erich Styger besprochen, um Fehler zu finden und Tipps für ein sauberes PCB Design zu erhalten. Die besprochenen Punkte sind im folgenden Abschnitt aufgelistet.

- Für eine angenehme Handhabung des PCBs wird empfohlen die Ecken abzurunden.
- Es wird empfohlen, die obere und untere Groundplane durch viele zusätzliche Vias miteinander zu verbinden (Ground-Stitching).
- Um die Montage von Dioden und ICs zu vereinfachen, wir empfohlen eine entsprechende Hilfe auf dem Silkscreen anzubringen (kleines Diodensymbol, Punkt bei Pin 1).
- Es wurden Bemassungen in der Lage F-Cu gemacht, diese sollen in die Schicht F-Fab verschoben werden.
- Es wird empfohlen, den Pin-Heder zum Debuggen als SMD Bauteil zu platzieren, dies erspart Montageaufwand.
- Auch beim Debug-Pin-Header wird ein Punkt zum Markieren des Pin 1 empfohlen, ausserdem sollte dieser nicht zu nahe am Rand platziert werden, damit er auch nach dem Einbau in einem Gehäuse noch gut erreichbar ist.
- Die Leitungen vom Shuntwiderstand zum INA239 sollten so kurz wie möglich gehalten werden, um Messfehlern vorzubeugen.
- Es soll nichts zu nahe am Rand platziert werden.
- Um die Bestückung zu vereinfachen, soll alles auf dem Silkscreen beschriftet werden.
- Da es mehrere Versionen des PCBs geben wird, wird empfohlen, die Versionsnummer auf das PCB zu Drucken.
- Der Name des Designers und das Datum sollten ebenfalls auf dem PCB ersichtlich sein.
- Das Raspberry Pi Pico sollte als SMD Bauteil platziert werden können, und es wäre von Vorteil, wenn der USB Anschluss erreichbar ist, ohne das Gehäuse zu öffnen.

Es wurden alle vorgeschlagenen Änderungen für die V1.1 umgesetzt bis auf den Punkt, dass der USB Anschluss des Picos von aussen zugänglich gemacht wird. Denn um den USB Anschluss des Raspberry Pi Picos von aussen zugänglich zu machen, müsste das Pico gedreht werden. Da es sich hierbei um eine sehr grosse Änderung handelt, wird diese erst in Version 2.0 umgesetzt.

5.4.6 Testergebnisse PCB V1.1

Nach dem Bestücken wurde die Funktionalität des PCBs getestet (siehe Kapitel 7.4). Der Test hat ergeben, dass die folgenden Punkte für die nächste Version angepasst werden müssen.

- Der GPIO Ausgang des Picos liefert zu wenig Strom für den IR Emitter, die Schaltung muss deshalb gemäss Kapitel 7.4.2 angepasst werden.
- Die Strommessung mit dem INA239 funktioniert (siehe Kapitel 7.4.1), die Alternative mit dem INA260 kann deshalb bei der nächsten Version des PCBs weggelassen werden.
- Obwohl die Schaltung zur Ausmessung des LED-Dimmerausgangs funktioniert, wird diese für die nächste Version gemäss Kapitel 7.4.3 angepasst, sodass der LED Dimmer nicht geöffnet werden muss.

Die Abbildungen 48 und 49 zeigen das bestückte PCB V1.1. Die SMD IR Diode wurde nicht geliefert und konnte deshalb nicht bestückt werden.

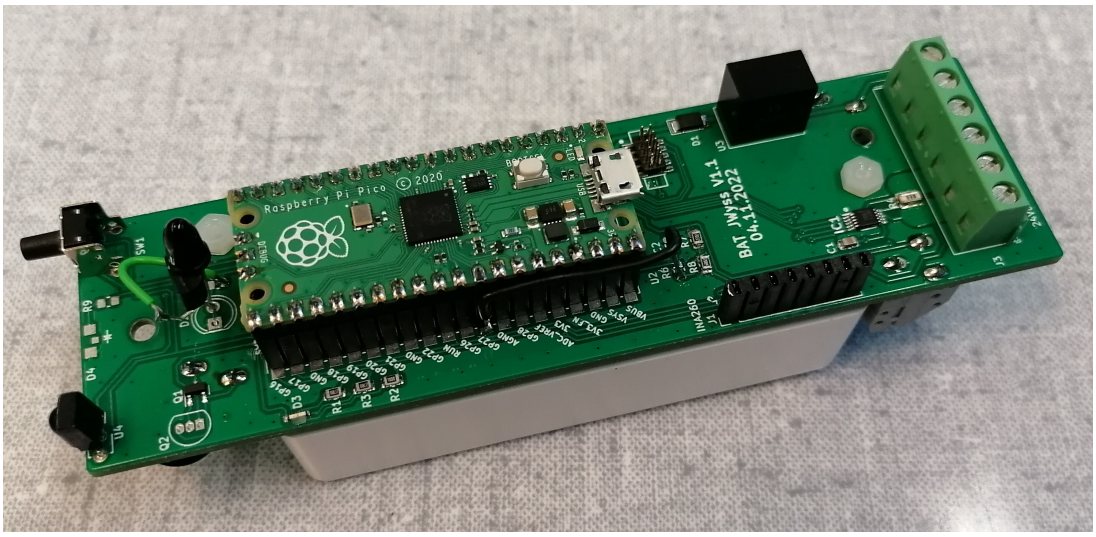


Abbildung 48: Vorderseite bestücktes PCB V1.1

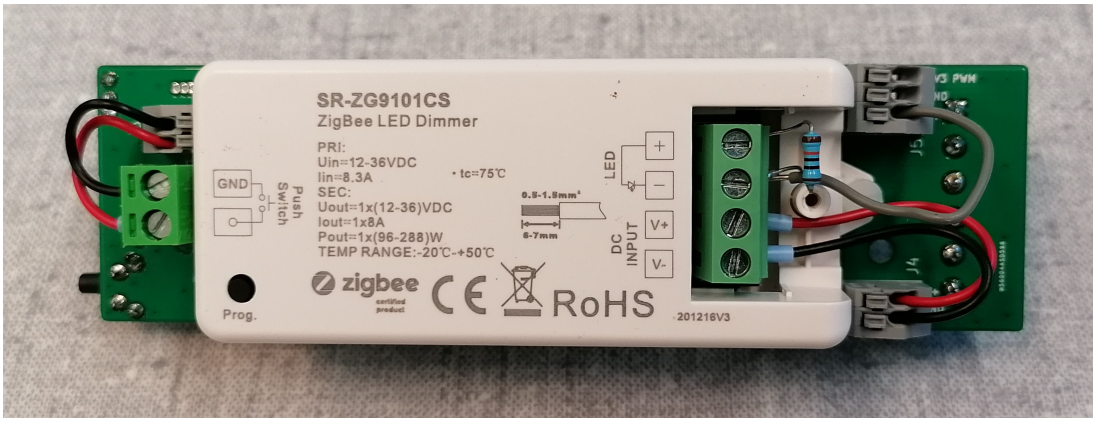


Abbildung 49: Rückseite bestücktes PCB V1.1

5.5 Schaltplan PCB V2.1

Der Test des PCB V1.1 hat ergeben, dass der Schaltplan an einiges Stellen angepasst werden muss. In diesem Kapitel werden die wichtigsten Anpassungen behandelt. Der gesamte Schaltplan des PCB V2.1 ist in Anhang A.4 zu finden. Auch das PCB 2.0 wurde in einem PCB Review mit dem betreuenden Dozenten Erich Styger besprochen (siehe Kapitel 5.6.3). Das PCB V2.1 beinhaltet die besprochenen Punkte bereits.

5.5.1 Anpassungen Kommunikation mit ZigBee LED Dimmer

Da die Betätigung des Tastereingangs der LED Dimmers über den SMD Mosfet funktioniert, wurde der THT Mosfet weggelassen.
Auf dem PCB V1.1 wird der Ausgang des LED Dimmers durch das Herausführen des 0 - 3.3 Volt PWM-Singals gemessen. Dies funktioniert sehr zuverlässig, jedoch muss dazu der LED Dimmer aufgebrochen werden. Um das Öffnen des LED Dimmers zu umgehen, wurde eine zweite Schaltung zur Messung des Ausgangssignals des LED Dimmers vorgesehen. Diese Schaltung wurde in Kapitel 7.4.3 erarbeitet. Mit der neuen Schaltung wird über einen Spannungsteiler ein 0 - 3.3 Volt PWM-Signal generiert, welches mit dem Mikrocontroller gemessen werden kann. Diese Variante wird bevorzugt, da der LED Dimmer nicht geöffnet werden muss. Damit die Schaltung mit einer Speisung von 12 -36 Volt betrieben werden kann, wurde ein Potentiometer vorgesehen, mit welchem der Ausgang des Spannungsteilers auf 3.3 Volt eingestellt werden kann. Damit das Potentiometer richtig eingestellt werden kann, ohne dabei eine zu hohe Spannung auf den GPIO Pin zu geben, wurde ein Jumper vorgesehen, welcher erst nach der Kalibrierung auf 3.3 Volt gesteckt werden darf. Wenn die Schaltung nur an einer Spannug betrieben werden soll, kann alternativ zum Potentiometer auch ein fester SMD Widerstand verbaut werden. Die Abbildung 50 zeigt die angepasste Schaltung zur Kommunikation mit dem LED Dimmer.

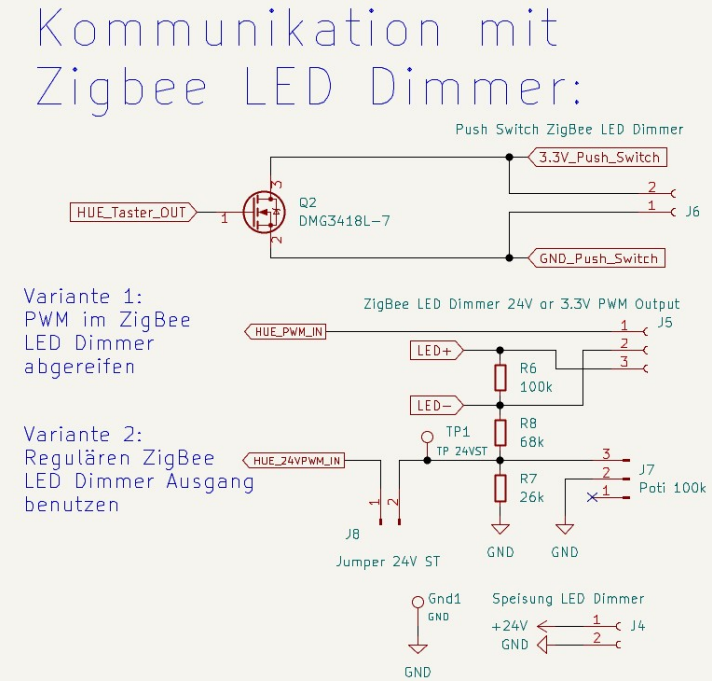


Abbildung 50: Anpassungen Kommunikation mit LED Dimmer

5.5.2 Anpassungen Strommessung

In Kapitel 7.4.1 wurde die Strommessung mit dem INA239 getestet. Des Test hat ergeben, dass die Strommessung mit dem INA239 gut funktioniert. Deshalb wurde bei der Version 2 des PCBs die Reservevariante mit dem INA260 Breakoutboard weggelassen. Ansonsten wurde die Strommessschaltung nicht verändert. Die Abbildung 51 zeigt den Schaltplan zur Strommessung.

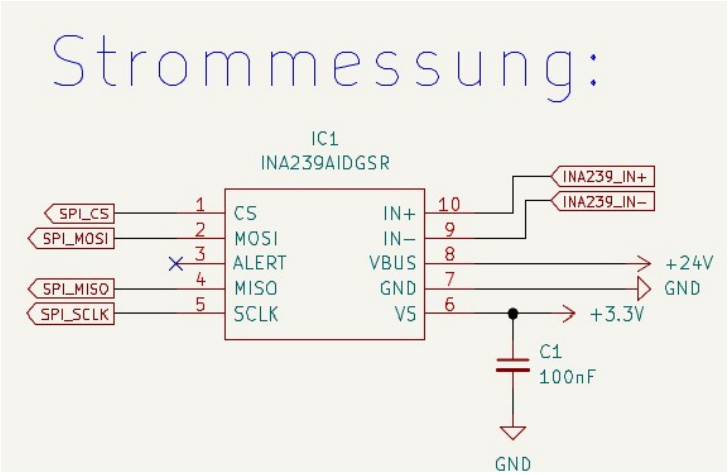


Abbildung 51: Anpassungen Strommessung

5.5.3 Anpassungen Anschlussklemme

Durch das Weglassen der Reservestrommessung mit dem INA260 Breakoutboard, entfällt auch die Klemme I Sens, welche für die Strommessung mit dem INA260 benötigt wurde. So werden nur noch zwei zweipolige Klemmen für die Zuleitung und das Leuchtenkabel benötigt.

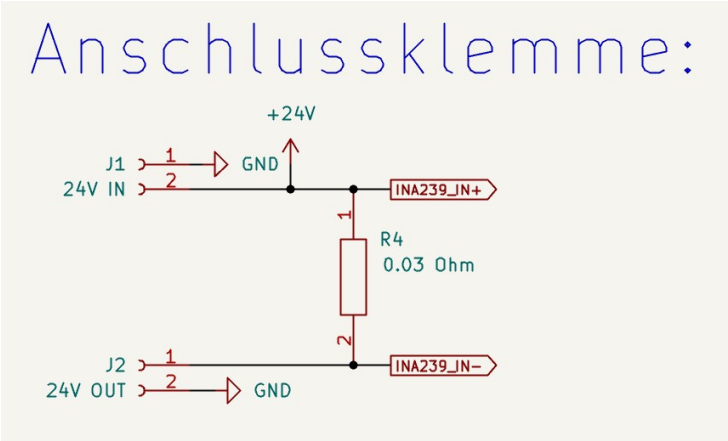


Abbildung 52: Anpassung Anschlussklemme

5.5.4 Anpassung IR rx / tx

Beim Empfängermodul für Infrarotsignale waren keine Anpassungen nötig. Beim Test des PCB V1.1 (siehe Kapitel 7.4.2) wurde festgestellt, dass das Pico nicht genügend Strom liefern kann, um ein genug starkes IRt Signal zu generieren. Deshalb wurde die Schaltung angepasst. Die IR Dioden werden nun mit einem N-Mosfet geschaltet und sind direkt mit der 5 Volt Speisung verbunden. Ausserdem wurden die Vorwiderstände angepasst, denn gemäss Datenblatt dürfen durch die THT IR Diode 1000 mA hohe Pulsströme und durch die SMD IR LED 1200ma hohe Pulsströme fliessen. Damit die 5V Speisung von den hohen Pulsströmen entlastet wird, wurden pro IR Diode jeweils ein Keramik- und ein Elektrolytkondensator vorgesehen. Die Abbildung 53 zeigt den angepassten Schaltplan.

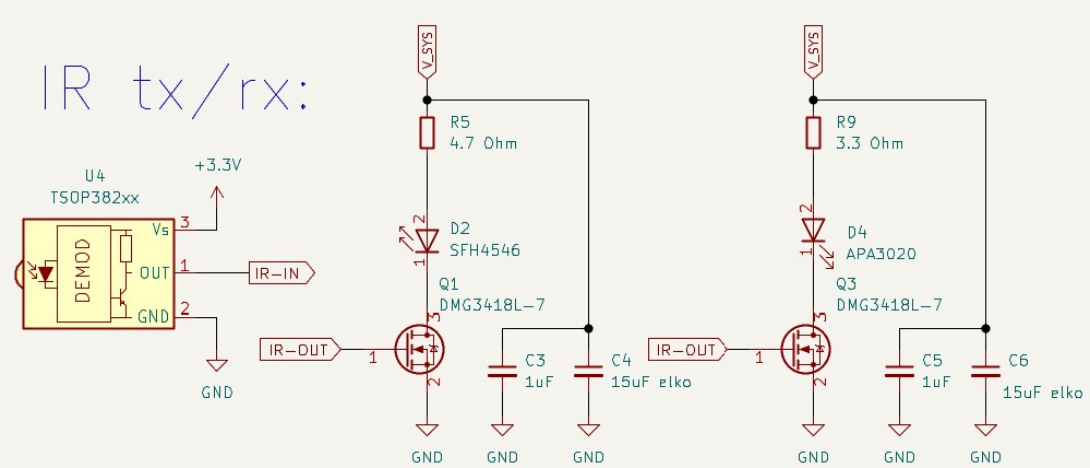


Abbildung 53: Anpassung IR rx / tx

5.5.5 Anpassung User Interface

Beim User Interface wurde der Taster auf Ground anstelle der 3.3 Volt gehängt. Bei der RGB LED waren keine Anpassungen nötig. Die Abbildung 54 zeigt die Anpassung beim Taster.

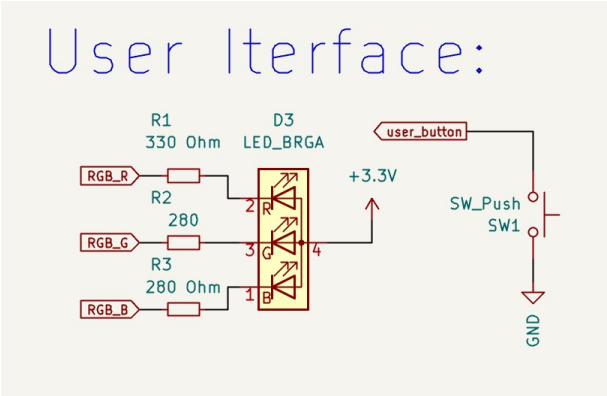


Abbildung 54: Anpassung User Interface

5.6 PCB Version 2.1

Die Abbildungen 55 und 56 zeigen die Vorder- und Rückseite des PCB V2.1. Das zweilagige Design wurde beibehalten. Die Grösse des PCBs ist neu 130 x 42 x 1.6 mm. Die Länge des PCBs V2.1 konnte reduziert werden, da die Anschlussklemmen des LED Dimmers um 90 Grad gedreht wurden. Das PCB wurde 4 mm breiter ausgeführt, damit links und rechts neben dem LED Dimmer noch ein kleiner Überhang des PCBs besteht. Dieser Überhang kann benutzt werden, um das PCB im Gehäuse durch eine entsprechende Nut zu befestigen.

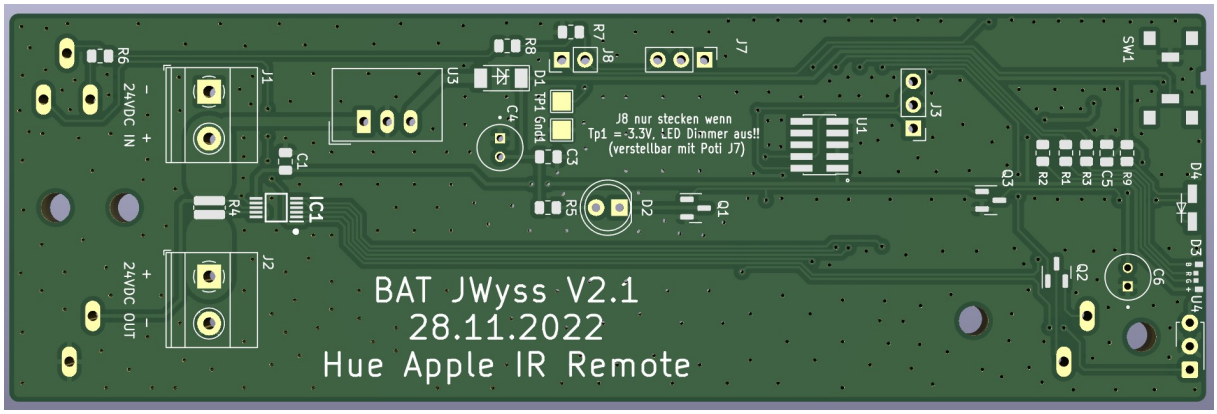


Abbildung 55: Vorderseite PCB V2.1

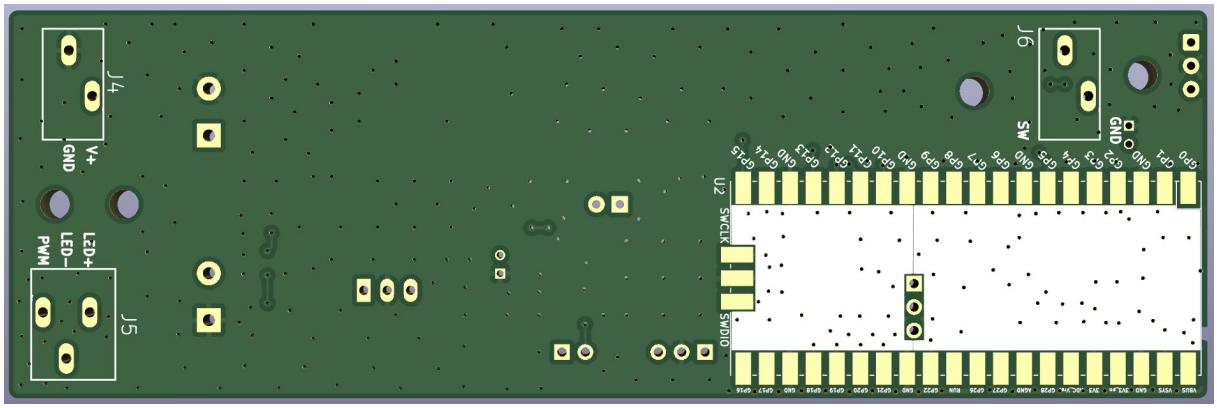


Abbildung 56: Rückseite PCB V2.1

5.6.1 Komponentenplatzierung

Die Komponenten wurden ähnlich angeordnet wie auf der PCB V1.1. Der grösste Unterschied liegt darin, dass das Raspberry Pi Pico um 180 Grad gedreht wurde. Ausserdem wird es neu auf der Rückseite des PCBs unter dem LED Dimmer angebracht. Durch diese Massnahme ist der USB Anschluss zugänglich, ohne das Gehäuse der H-A-IR-Remote zu öffnen. Das Pico wird als SMD-Bauteil montiert, da so der Bestückungsaufwand reduziert werden kann. Bei der Platzierung des Picos wurde darauf geachtet, dass der Knopf auf dem Pico nicht durch den LED Dimmer verdeckt wird. Denn dieser muss betätigt werden können, wenn ein Software-Update über den USB Anschluss auf das Pico geflasht werden soll. Durch die SMD-Montage

des Picos musste ein SMD-Taster verwendet werden. Die RGB-LED wurde auf die Vorderseite des PCBs verschoben. Ansonsten gab es bei der Platzierung der Komponenten keine grossen Änderungen. Beim Platzieren der Bauteile wurde darauf geachtet, keine SMD Bauteile auf der Rückseite des PCBs zu platzieren, da dies die Bestückung aufwendiger macht.

5.6.2 Routing und Leiterbahnbreiten

Beim Routing des PCB V2.1 wurde gleich vorgegangen wie beim PCB V1.1 (siehe Kapitel 5.4.3). Auch die Leiterbahnbreiten wurden beibehalten.

5.6.3 PCB Review 2

Auch die zweite Version des PCBs wurde mit dem betreuenden Dozenten Erich Styger besprochen. Folgende Punkte wurden dabei angemerkt.

- Es wurde gewünscht, dass das normale Raspberry Pi Pico W alternativ zum normalen Raspberry Pi Pico bestückt werden kann. Der Footprint stimmt bis auf die Debugganschlüsse überein und kann somit einfach angepasst werden.
- Da das Raspberry Pi Pico als SMD-Bauteil verbaut wird, wird empfohlen ein Silkscreen über die gesamte Kontaktfläche zu ziehen, um eine zusätzliche Isolation zu erreichen, die Debug-Pins für das Raspberry Pi Pico W werden mit einem Isolierband bedeckt, wenn das normale Pico bestückt wird.
- Das Raspberry Pi Pico sollte von Hand angelötet werden, um die Komponenten darauf zu schonen.
- Um die Montage zu vereinfachen, sollten nach Möglichkeit nur auf einer Seite SMD-Bauteile angebracht werden.

Das PCB V2.1 beinhaltet die aufgeführten Punkte aus beiden PCB Reviews.

5.6.4 Bestücktes PCB V2.1

Die Abbildungen 57 und 58 zeigen das bestückte PCB V2.1. Beim Testen der Funktionalität des PCB V2.1 wurden keine Fehler in der Schaltung festgestellt.

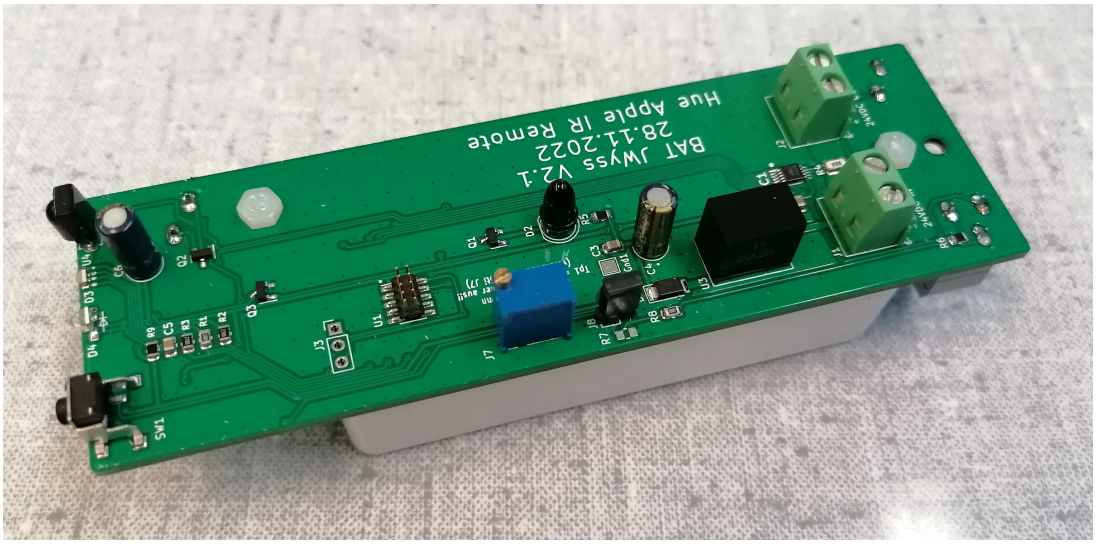


Abbildung 57: Vorderseite bestücktes PCB V2.1

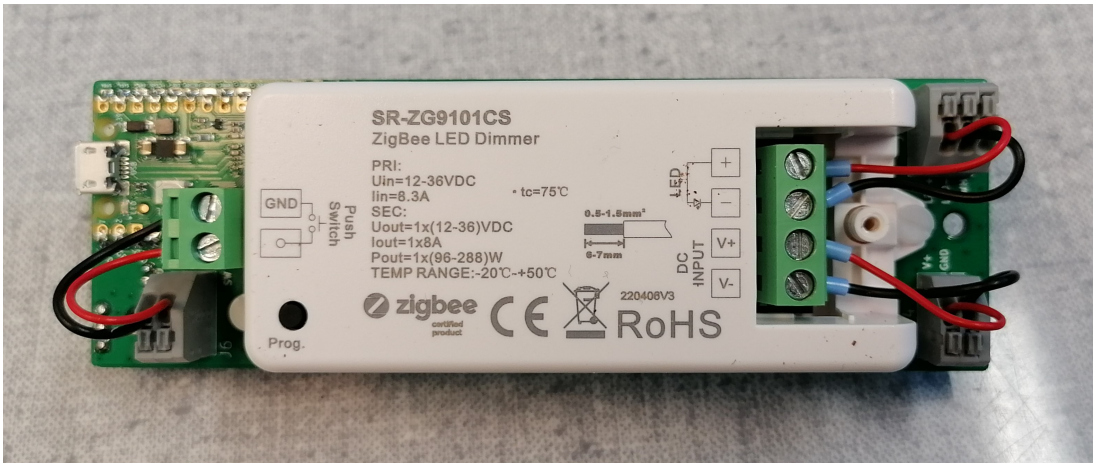


Abbildung 58: Rückseite bestücktes PCB V2.1

5.7 Gehäuse

Damit das PCB inklusive Mikrocontroller geschützt ist und am Ständer der Stehleuchte angebracht werden kann, wurde ein Gehäuse mit einem 3D-Drucker hergestellt. In diesem Kapitel (5.7) wird aufgezeigt, wie das Gehäuse entwickelt wurde.

5.7.1 Gehäuse V1 Mock-Up

Der erste Prototyp diente lediglich als Mock-Up, um das Befestigungskonzept zu veranschaulichen und zu verifizieren. Die Grösse des Gehäuses wurde abgeschätzt, da das PCB zu diesem Zeitpunkt noch nicht gezeichnet war. Da es sich bei der V1 nur um ein Mock-Up handelt, wurde eine Schichtdicke von 0.2 mm zum Drucken gewählt. In der Abbildung 59 ist zu sehen, wie das Gehäuse am Leuchtenständer angebracht wird. Zur Befestigung wurden Gummibänder verwendet, für die finale Version wird weisses Klettband verwendet. Um ein Verrutschen nach links und rechts zu Vermeiden, wurde eine Einbuchtung über die ganze Länge auf der Rückseite des Gehäuses vorgesehen. Um die Kabel in das Gehäuse zu führen, wurde eine Kabelverschraubung verwendet, welche zugleich als Zugentlastung dient.

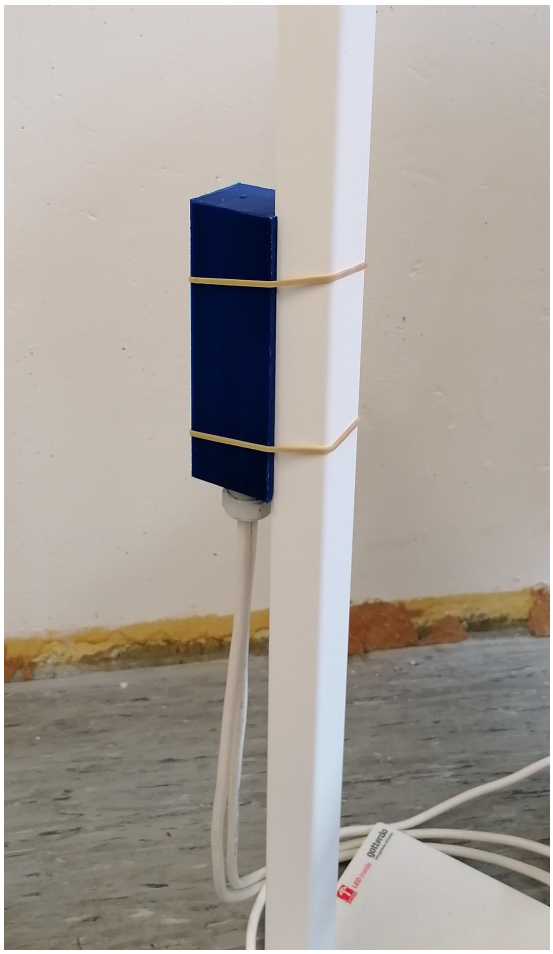


Abbildung 59: Gehäuse V1 am Ständer der Stehleuchte montiert

5.7.2 Gehäuse V2

Nachdem die PCB V2.1 gezeichnet war, konnte das Gehäuse mit den richtigen Massen designt werden. Denn nun war klar, wie gross das PCB ist und an welchen Stellen Aussparungen vorgenommen werden müssen. Da sich die Einbuchtung auf der Rückseite des Gehäuses beim Mock-Up bewährt hat, wurde diese für die V2 übernommen. Die eine klobige Kabelverschraubung wurde durch zwei kleinere ersetzt. Der Deckel des Gehäuse V2 wird durch einen Klemmmechanismus mit dem Gehäuse verbunden. Die Abbildung 60 zeigt das Gehäuses V2 ohne Deckel. Um Fehler in der Konstruktion vor dem Drucken zu bemerken, wurde eine Baugruppe aus allen Elementen (Deckel, PCB, Gehäuse) erstellt. Dadurch war es möglich, die Masse zu prüfen und anzupassen. Die Version zwei hatte dennoch einige Probleme. Das Gehäuse wurde mit weissem ABS gedruckt. Da die Druckkammer des verwendeten 3D-Druckers nicht beheizt ist, haben sich gewisse Schichten beim Auskühlen voneinander gelöst (siehe Abbildung 61). Der untere Teil des Gehäuses war durch die gewählte Geometrie zu flexibel, was zur Folge hatte, dass der Klemmmechanismus des Deckels nicht mehr sauber funktionierte. Ausserdem wurde festgestellt, dass 90 Grad Überhänge trotz Stützstruktur sehr unsauber gedruckt werden. Überhänge mit einem Winkel von bis zu 45 Grad wurden hingegen sehr sauber gedruckt. Mit dem Gehäuse V2 konnten die Aussparungen für LEDs, Taster etc. überprüft werden. Einige Aussparungen müssen noch leicht verschoben werden, damit sie sauber passen.

Abbildung 60: Gehäuse V2 ohne Deckel

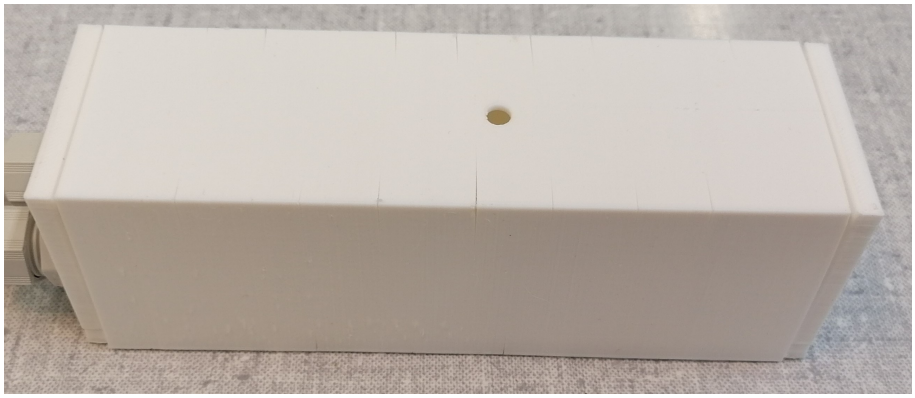


Abbildung 61: Gehäuse V2 mit Deckel

5.7.3 Gehäuse V3

Die Erkenntnisse aus den ersten beiden Versionen des Gehäuses flossen in das Design der V3 mit ein. Um das Gehäuse stabiler zu machen, wurden alle Wände bis oben gezogen. Der Deckel wurde so designt, dass er in eine dafür vorgesehene Nut geschoben und anschließend mit einer Schraube fixiert werden kann. Um das Problem mit den sich lösenden Schichten zu beheben, wurde die V3 wieder mit PLA gedruckt. Überhänge wurden mit einer 45 Grad Fase versehen, damit der Druck sauber wird. Damit der Taster besser gedrückt werden kann, wurde auch hier eine Fase vorgesehen. Die Schichtdicke wurde mit 0.1 mm gewählt, um eine schönere Oberfläche zu erhalten. Die Abbildungen 62 und 63 zeigen das Gehäuse V3, in welchem die Hardware bereits eingebaut ist.

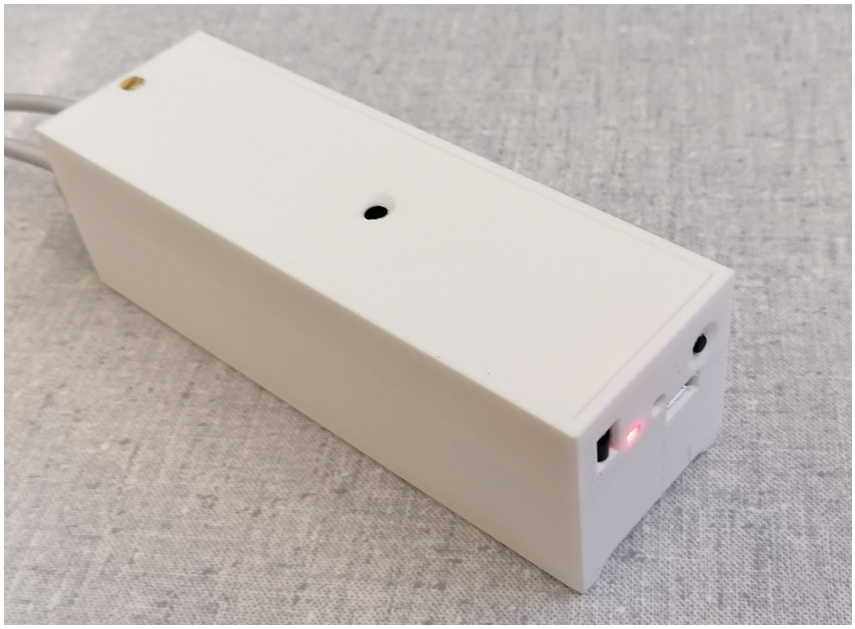


Abbildung 62: Gehäuse V3

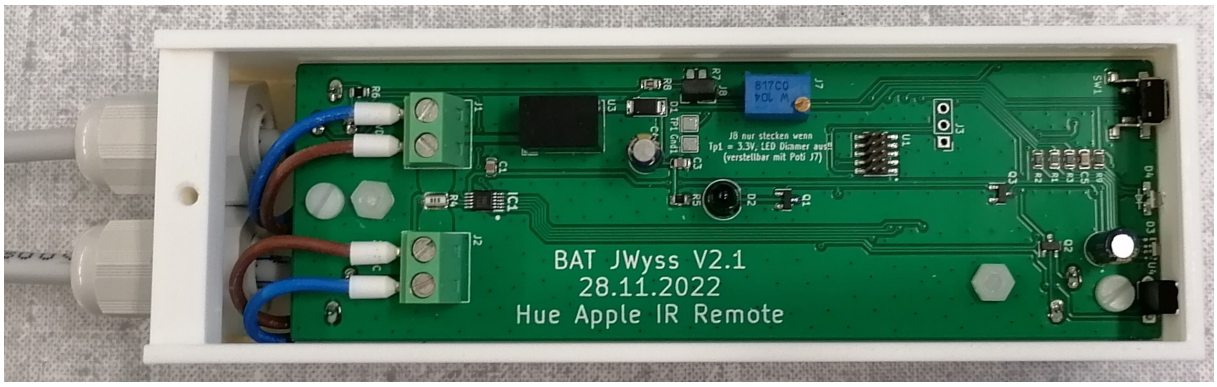


Abbildung 63: Gehäuse V3 ohne Deckel

Zur Montage des finalen Gehäuses am Leuchtenständer wurde weisses Klettband verwendet, welches Gehäuse und Ständer umschliesst. Diese Befestigung hinterlässt keine Spuren an der Leuchte und die Hardware lässt sich einfach anbringen und entfernen. Die Abbildung 64 zeigt das am Leuchtenständer montierte Gehäuse.

Abbildung 64: Gehäuse V3 am Ständer der Stehleuchte montiert

5.8 Kosten Hardware

Der Materialpreis der Hardware beträgt CHF 102,30 . Die Kosten wurden abgeschätzt, indem der Stückpreis aller verwendeten Bauteile und Materialien aufgelistet und aufsummiert wurde. Diese Auflistung befindet sich in Anhang A.5.

6 Erarbeitung und Beschreibung der Software

In diesem Kapitel wird vorgestellt, wie die Software für die H-A-IR-Remote erarbeitet wurde und wie sie funktioniert.

6.1 Übersicht Softwarekonzept

Die grundsätzliche Aufgabe der Software ist es, den Zustand der Leuchte in der App und den wirklichen Zustand der Leuchte synchron zu halten. Wird die Leuchte ausgeschaltet, muss der Zustand in der App aktualisiert werden. Wird die Leuchte in der App ausgeschaltet, muss die Stehleuchte ausgeschaltet werden und umgekehrt. Dies wird durch eine Finite State Machine (FSM) erreicht. Die Abbildung 65 zeigt das Blockdiagramm der FSM. Insgesamt gibt es sieben Zustände und zwei Exit Actions, welche einen Zustandswechsel erlauben.

Exit Action 1: Get App State (GAS): Diese Funktion gibt True zurück, wenn der LED Dimmer im Moment des Aufrufens an ist und false, wenn der LED Dimmer aus ist. Der Zustand des LED Dimmers entspricht immer dem Zustand in der App.

Exit Action 2: Get Lamp State (GLS): Diese Funktion gibt True zurück, wenn die Leuchte im Moment des Aufrufens an ist und False, wenn die Leuchte aus ist.

Das Ausrufezeichen steht dafür, dass das invertierte Signal benötigt wird, um den Zustand zu verlassen. Zwei &-Zeichen stehen für eine logische UND-Verknüpfung, es müssen also beide Sachen erfüllt sein, um den Zustand zu verlassen.

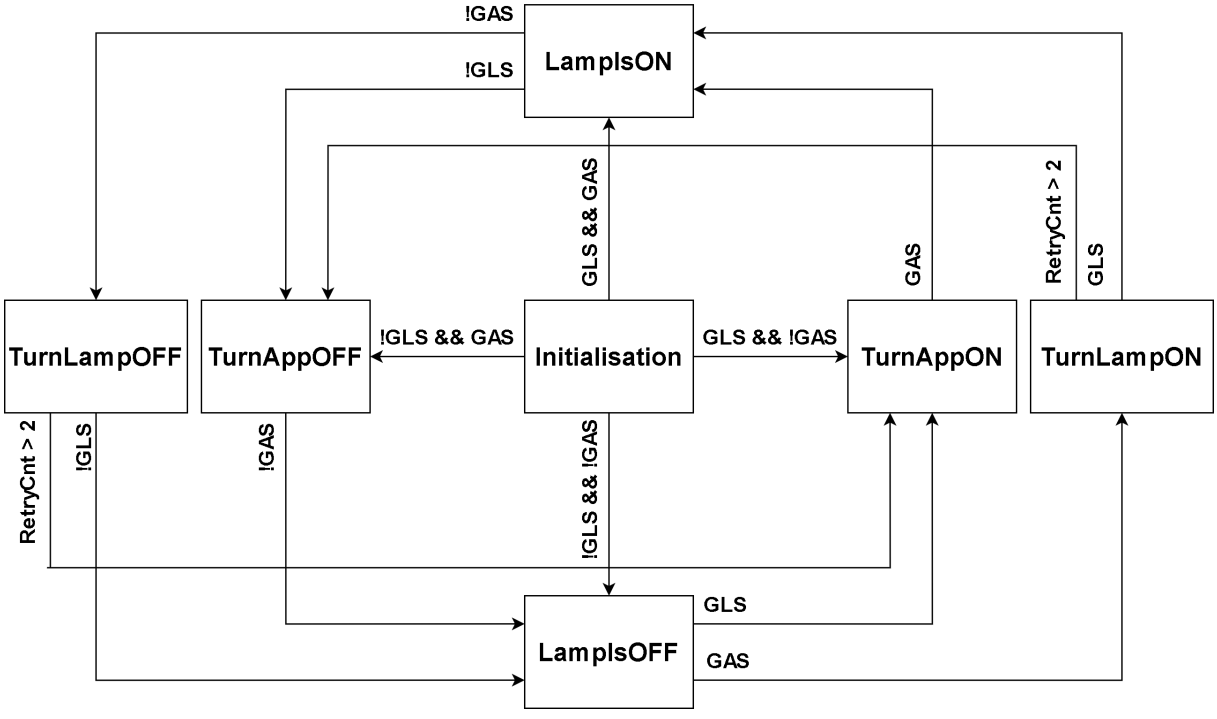


Abbildung 65: FSM Software

6.1.1 Beschreibung der Zustände

In diesem Kapitel wird genauer erklärt, was in den verschiedenen Zuständen aus Abbildung 65 passiert.

Initialisation
Wenn die Hardware mit Strom versorgt wird, befindet sich die FSM im Zustand Initialisation. In diesem Zustand leuchtet die RGB-LED weiss. In diesem Zustand wird die Hardware initialisiert. Anschliessend wird eine Sekunde gewartet, damit die Leuchte und der LED Dimmer aufstarten können. Nach dieser Wartezeit gibt es vier Möglichkeiten:

- Die Lampe und der LED Dimmer sind aus -> Wechsel in Zustand LampIsOFF
- Die Lampe und der LED Dimmer sind an -> Wechsel in Zustand LampIsON
- Die Lampe ist aus, der LED Dimmer aber an -> Wechsel in Zustand TurnAppOFF
- Die Lampe ist an, der LED Dimmer aber aus -> Wechsel in Zustand TurnAppON

Nach einem Stromunterbruch gelangt die Leuchte immer wieder in den Zustand, den sie vor dem Stromunterbruch hatte. Somit ist gegeben, dass die Leuchte mit und ohne H-A-IR-Remote gleich auf einen Stromunterbruch reagiert.

LampIsON / LampIsOFF
In diesen Zuständen sind Lampe und der LED Dimmer beide ein- oder ausgeschaltet. Das bedeutet Leuchte und App sind in einem synchronen Zustand. Alle 50 ms wird überprüft, ob die Zustände von Leuchte und App immer noch synchron sind. Es gibt zwei verschiedene Ereignisse, die Eintreten können:

- Die Leuchte wird zum Beispiel mit der Fernbedienung ein / ausgeschaltet, somit gibt die Funktion GLS True / False zurück und es wird in den Zustand TurnAppON / TurnAppOFF gewechselt.
- Der Benutzer schaltet die Leuchte in der App ein/aus, somit gibt die Funktion GAS True / False zurück und es wird in den Zustand TurnLampON / TurnLampOFF gewechselt.

Im Zustand LampIsON kann die Stehleuchte durch die App in 3 Stufen gedimmt werden. Genauere Informationen dazu, wie das Dimmen realisiert wurde, sind in Kapitel 6.5 zu finden.

TurnLampON / TurnLampOFF
Wenn der Benutzer die Leuchte in der App ein- oder ausschaltet, landet die Software je nach Ereignis entweder im TurnLampON oder TurnLampOFF Zustand. Die Leuchte soll ein- oder ausgeschaltet werden, um wieder in einen synchonen Zustand zu gelangen (App ON, Lamp On oder App OFF, Lamp OFF). Dazu wird der Befehl TLS (Toggle Lamp State) verwendet. Dieser versendet im Hintergrund den IR Befehl, um die Leuchte ein- und auszuschalten. Es gibt zwei Ereignisse welche eintreten können:

- Das Infrarot Signal ist bei der Leuchte angekommen und diese hat wie gewollt ein- oder ausgeschaltet -> GLS gibt je nach Zustand True oder False zurück -> Wechsel in Zustand LampIsON resp. LampIsOFF.
- Wenn das IR Signal nicht bei der Leuchte ankommt und diese deshalb nicht umschalten kann, wird noch zwei weitere Male versucht, die Leuchte ein- oder auszuschalten -> misslingt dies, wird der Zustand der App angepasst, dass dieser wieder dem Zustand der Leuchte entspricht.

Durch das Errorhandling, welches z.B. bei einem blockierten Pfad für das IR Signal zu Tragen kommt, gelangt das System immer wieder in einen synchronen Zustand. Bei einem Fehler merkt der Benutzer dies, da die Leuchte nicht reagiert, der Zustand von Leuchte und App wird aber automatisch synchronisiert. In diesen beiden Zuständen leuchtet die RGB LED blau.

TurnAppON / TurnAppOFF
Wenn die Leuchte mit der Fernbedienung, mit dem Bedienfeld oder durch den integrierten Bewegungsmelder ein- oder ausgeschaltet wird, muss der Zustand in der App aktualisiert werden, damit das System synchron ist. Dazu werden die beiden Zustände TurnAppON oder TurnAppOFF gebraucht. In diesen Zuständen wird der Taster des LED Dimmers betätigt, was einen Zustandswechsel in der App zur Folge hat. Ist der Zustandswechsel korrekt verlaufen (Überprüfung durch den Befehl GAS), wechselt die Software in den Zustand LampIsON rsp. LampIsOFF und das System ist wieder synchron. In diesen beiden Zuständen leuchtet die RGB-LED blau.

6.2 Verwendung von FreeRTOS

Da in der Software mehrere Prozesse gleichzeitig ablaufen sollen, wird in dieser Arbeit mit einem opensource Betriebssystem für Mikrocontroller gearbeitet (FreeRTOS). Die Verwendung von FreeRTOS erlaubt es, mehrere Tasks aufzusetzen, welche von einem Scheduler verwaltet werden.

6.3 Aufbau Software

In der Abbildung 66 ist der Aufbau der Software ersichtlich. Es werden die wichtigsten Module und wichtige Funktionen dargestellt. Der Einstiegspunkt ist das main.c - File. Hier werden alle Module initialisiert. Anschliessend wird der Scheduler gestartet. In Kapitel 6.4 werden die einzelnen Softwaremodule und ihre Schnittstellen genauer beschrieben.

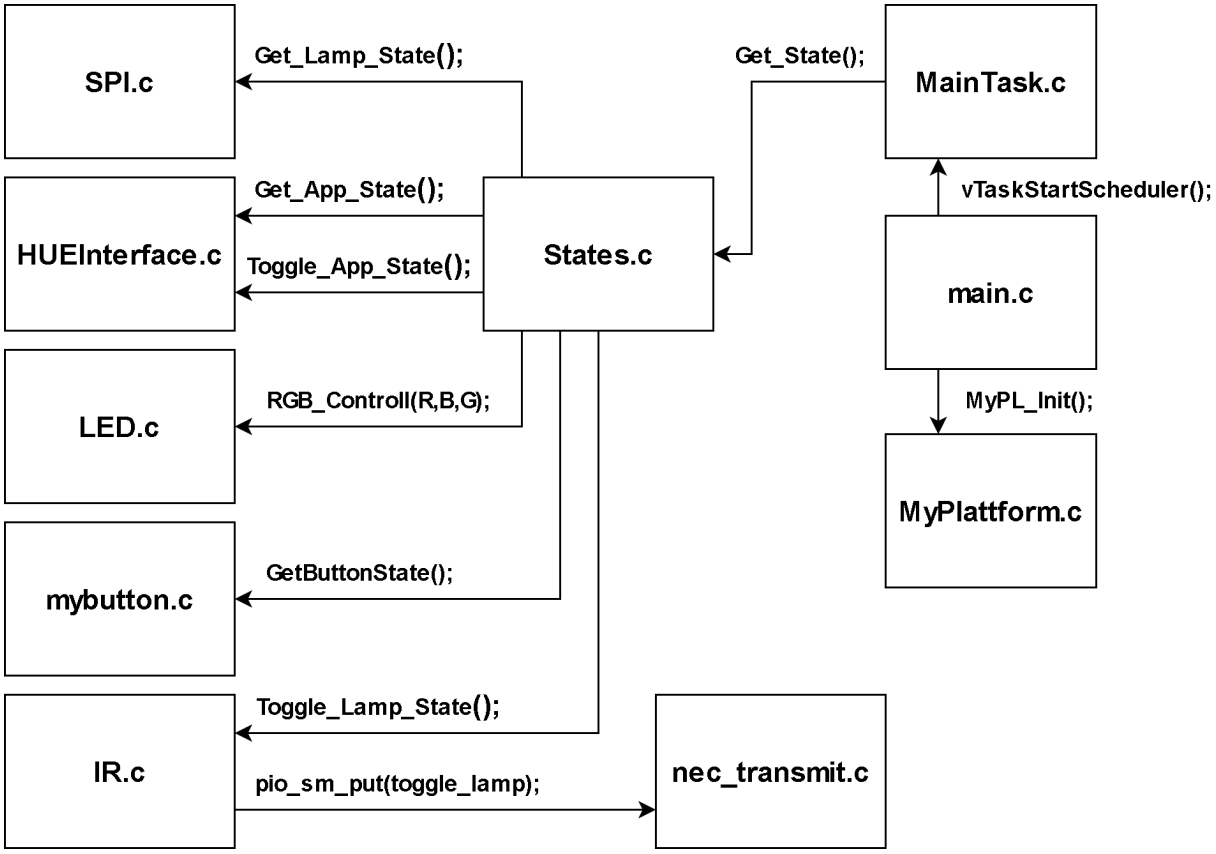


Abbildung 66: Aufbau Software

6.4 Softwaremodule

In diesem Kapitel werden die verschiedenen Softwaremodule und ihre Schnittstellen vorgestellt. Aus Gründen der Übersichtlichkeit werden nur die wichtigsten Auszüge aus dem Code präsentiert, der gesamte Code befindet sich im elektronischen Anhang.

6.4.1 MainTask.c

In diesem Modul befindet sich der Code für die FSM. Damit der Code übersichtlich bleibt, wurden die einzelnen Zustände als Funktionen in das Modul States.c ausgelagert. Die FSM wird im MainTask periodisch aufgerufen, mit einer Zykluszeit von 50 ms. Diese Zeit wurde so gewählt, da die Reaktionszeit des Menschen bei etwa 100 ms liegt. Durch die doppelt so hohe Geschwindigkeit soll sich das Programm dynamisch für den Benutzer anfühlen.
In jedem Zyklus bezieht die FSM den aktuellen Zustand aus dem Modul States.c über die Funktion GetState(). Die Abbildung 67 zeigt den MainTask mit der FSM.

```
14 static void MainTask(void * pv){
15
16     for(;;) {
17         switch(GetState()){ /*FSM*/
18             case Initialisation:
19                 State_Initialisation();
20                 break;
21             case LampIsON:
22                 State_LampIsON();
23                 break;
24             case LampIsOFF:
25                 State_LampIsOFF();
26                 break;
27             case TurnLampOFF:
28                 State_TurnLampOFF();
29                 break;
30             case TurnAppOFF:
31                 State_TurnAppOFF();
32                 break;
33             case TurnLampON:
34                 State_TurnLampON();
35                 break;
36             case TurnAppON:
37                 State_TurnAppON();
38                 break;
39             default:
40                 /*Code should never get here*/
41                 break;
42         }
43     vTaskDelay(pdMS_TO_TICKS(50));
44     }
45 }
```

Abbildung 67: MainTask

Schnittstellen
Dieses Modul hat nach aussen nur die Funktionen zu Initialisierung und Deinitialisierung als Schnittstelle.

6.4.2 States.c

In diesem Modul werden alle Zustände der Statemachine als Funktionen definiert. Jeder State besteht aus den folgenden drei Teilen:

- ENTRY -> Dieser Teil des Codes wird nur ausgeführt, wenn neu in diesen Zustand gewechselt wurde.
- DO -> Dieser Teil des Codes wird kontinuierlich ausgeführt, bis der State verlassen wird.
- EXIT -> In diesem Teil des States werden die Bedingungen definiert, unter welchen der State verlassen wird.

Die Abbildung 68 zeigt beispielhaft den State TurnLampOFF. Beim Betreten des States wird die RGB-LED auf blau gestellt. Anschliessend wird über den Befehl Toggle_Lamp_State() versucht,

die Stehleuchte auszuschalten. Da die Leuchte eine gewisse Zeit braucht, um auszuschalten, wird eine Sekunde gewartet. Falls das Ausschalten geklappt hat, wird in den Zustand LampIsOFF gewechselt. Klappt das Ausschalten der Leuchte nicht, wird es erneut versucht. Wenn eine gewisse Anzahl von gescheiterten Versuchen erreicht ist, wird in den Zustand TurnAppON gewechselt.

```
156 void State_TurnLampOFF(void){
157     //ENTRY -> this part of the code will only run once
158     //============================================================
159     if(ENTRY){
160         ENTRY = false;          //only do the entry statement once
161         RGB_Controll(0,0,1);    //turn the blue part of the RGB LED on, the other parts off
162     }
163 
164     //DO -> this part of the code is running continuous until an exit statement becomes true
165     //============================================================
166     Toggle_Lamp_State();        //Lamp needs to be turned OFF
167     IRRetryCounter++;
168     vTaskDelay(pdMS_TO_TICKS(TimeToWaitAfterLampToggleCommand)); //wait to let the lamp turn off
169 
170     //EXIT -> if one of these Statements becomes true, the state will be changed
171     //============================================================
172     if(!Get_Lamp_State()){          //Lamp was turned off
173         ENTRY = true;               //do the entry statement in the next State
174         IRRetryCounter = 0;         //reset the retry counter
175         actual_State = LampIsOFF;   //set the next state
176     }
177 
178     if(IRRetryCounter>=IRRetrys){   //Lamp couldn't be turned off
179         ENTRY = true;               //do the entry statement in the next State
180         IRRetryCounter = 0;         //reset the retry counter
181         actual_State = TurnAppON;   //set the next state
182     }
183 }
```

Abbildung 68: State TurnLampOFF

Schnittstellen

Dieses Modul hat zu einen die GetState() Funktion als Schnittstelle, diese wird benötigt, damit die FSM im MainTask Zugriff auf den aktuellen Zustand hat. Zum anderen müssen auch alle State-Funktionen als Schnittstellen nach aussen für die FSM im MainTask erreichbar sein. Die Abbildung 69 zeigt sämtliche Schnittstellen des Softwaremoduls States.c.

```
21 State_e GetState(void);              /* this function returns the actual state */
22 void State_Initialisation(void);     /* this function contains the state Initialization */
23 void State_LampIsON(void);           /* this function contains the state LampIsON */
24 void State_LampIsOFF(void);          /* this function contains the state LampIsOFF */
25 void State_TurnLampOFF(void);        /* this function contains the state TurnLampOFF */
26 void State_TurnAppOFF(void);         /* this function contains the state TurnAppOFF */
27 void State_TurnLampON(void);         /* this function contains the state TurnLampON */
28 void State_TurnAppON(void);          /* this function contains the state TurnAppON */
```

Abbildung 69: Schnittstellen des Moduls States.c

6.4.3 HUEInterface.c

Dieses Modul übernimmt die Kommunikation mit dem LED Dimmer und somit auch mit der Hue App. Die Kommunikation mit dem LED Dimmer besteht aus zwei Teilen. Zum einen wird der Ausgang des LED Dimmers gemessen und den Zustand der App zu erhalten, zum anderen kann der Zustand des LED Dimmers über den Tastereingang geändert werden. In Kapitel 7.3.5 wurde herausgefunden, dass der Tastereingang des LED Dimmers mindestens 20 Millisekunden betätigt werden muss, damit der LED Dimmer umschaltet. Die Abbildung 70 zeigt den Code, mit welchem der Tastereingang des LED Dimmers betätigt werden kann.

```
58 //this function allows to toggle the state of the Hue App
59 void Toggle_App_State(void){
60     gpio_put(HUE_Button_PIN, 1);
61     vTaskDelay(pdMS_TO_TICKS(20));
62     gpio_put(HUE_Button_PIN, 0);
63     vTaskDelay(pdMS_TO_TICKS(20));
64 }
```

Abbildung 70: Funktion Toggle_App_State()

Um das PWM Signal des LED Dimmers einzulesen, wird die Funktion measure_duty_cycle() benutzt. Diese Funktion wurde aus Quelle [21] übernommen. In Abbildung 71 ist der HueInterface Task ersichtlich. In diesem Task wird alle zehn Millisekunden der aktuelle Dutycycle des LED Dimmer Ausgangs gemessen. Sobald der Dutycycle einen Thresholdwert von 12 % überschreitet ist der Zustand der Leuchte in der App an (siehe Kapitel 7.4.3). Aufgrund der Schaltung zur Messung des Ausgangssignal der LED Dimmers, muss das gemessene Dutycycle Signal invertiert werden (siehe Kapitel 7.3.6).

```
66 //this task reads the dutycycle from the LED Dimmer every 10 Ms and provides the AppState
67 static void HUEInterfaceTask(void * pv){
68
69     //if the dutycycle is greater than this value the AppState is on
70     const static float threshold24V = 0.12;
71
72     //Read the AppDutyCycle every 10 MS
73     const static int DutyCycleReadFrequencyInMS = 10;
74
75     for(;;) {
76
77             //measure the dutycycle and invert it
78             DutyCycle = 1-(measure_duty_cycle(MEASURE_PWM_PIN));
79
80             //convert the dutycycle to an number in percent
81             DutyCycleInPercent = (uint8_t)(DutyCycle*100);
82
83             if((DutyCycle < threshold24V)){
84                 AppState = false;
85                 }else{
86                 AppState = true;
87                 }
88
89     vTaskDelay(pdMS_TO_TICKS(DutyCycleReadFrequencyInMS));
90     }
91 }
```

Abbildung 71: HueInterfaceTask

Die Schnittstellen des Moduls HUEInterface.c sind in Abbildung 72 ersichtlich. Über diese Schnittstellen kann zum einen der LED Dimmer und somit der Zustand in der App umgeschaltet werden. Zum anderen kann der Zustand der App über eine Getter-Funktion bezogen werden. Damit die Stehleuchte gedimmt werden kann (Informationen dazu im Kapitel 6.5), wird auch der Dutycycle des LED Dimmerausgangs über eine Getter-Funktion verfügbar gemacht.

```
14 void Toggle_App_State(void);        /*this function allows to toggle the state in the app*/
15 bool Get_App_State(void);           /*this function allows to get the actual app state*/
16 uint8_t Get_App_DutyCycle(void);    /*this function allows to get the actual dutycycle in percent*/
17
18 void HUEInterface_Init(void);
19 void HUEInterface_Deinit(void);
```

Abbildung 72: Schnittstellen des Moduls HUEInterface.c

6.4.4 IR.c

Dieses Modul ermöglicht das Empfangen und Versenden von Infrarotsignalen nach dem NEC-Protokoll (Informationen zum NEC-Protokoll sind in Kapitel 3.1.1 zu finden). Um IR Signale zu versenden und zu empfangen, werden die Bibliotheken necreceive.c und nectransmit.c verwendet. Die beiden Bibliotheken wurden von der Quelle [22] bezogen. Die Funktionsweise der Bibliotheken wird in Kapitel 3.2 genauer beschrieben. Um einen IR Signal zu versenden, kann die Funktion pio_sm_put(uint32_t transmit_command) verwendet werden. In Kapitel 7.5 wird beschrieben, wie die Signale der Apple IR Remote aufgezeichnet wurden, damit sie in diesem Modul verwendet werden können. Die Abbildung 73 zeigt den Code, um das Power On/Off Signal zu versenden. Dieser Code wird ausgeführt, wenn der Befehl Toggle_Lamp_State() aufgerufen wird. Für die Vollständigkeit wurde jeder IR Befehl der Apple IR Remote vorimplementiert.

```
38 /*send the power on/off IR command to toggle the lamp state*/
39 void IR_PowerOnOff_tx(void){
40     pio_sm_put(pio, tx_sm, apple_IR_power1_tx);
41     vTaskDelay(pdMS_TO_TICKS(100));
42     pio_sm_put(pio, tx_sm, apple_IR_power2_tx);
43     vTaskDelay(pdMS_TO_TICKS(100));
44 }
```

Abbildung 73: Code um das Power On / Off Signal zu versenden

Für mögliche Erweiterungen der H-A-IR-Remote wurde auch das Empfangen von IR Signalen implementiert. Hierzu wurde ein Task aufgesetzt, welcher alle 20 Milisekunden prüft, ob ein oder mehrere IR Signale empfangen wurden. Der Code wurde so geschrieben, dass der auszuführende Code, bei Empangen des entsprechenden IR Signals, nur noch eingefügt werden kann. Die Abbildung 74 zeigt den Code des IRreceiveTasks.

```
85  //this Task allows to react to received IR signals
86  static void IRReceiveTask(void * pv){
87
88      static uint32_t rx_frame; //received frames are stored in this variable
89
90      for(;;) {
91
92              while (!pio_sm_is_rx_fifo_empty(pio, rx_sm)) {
93
94                  rx_frame = pio_sm_get(pio, rx_sm);
95
96                  if(rx_frame == apple_IR_up_tx){
97                  //Action to do if button up was pressed
98                  }
99                  if(rx_frame == apple_IR_left_tx){
100                 //Action to do if button left was pressed
101                 }
102                 if(rx_frame == apple_IR_right_tx){
103                 //Action to do if button right was pressed
104                 }
105                 if(rx_frame == apple_IR_down_tx){
106                 //Action to do if button down was pressed
107                 }
108             }
109     vTaskDelay(pdMS_TO_TICKS(20));
110     }
111 }
```

Abbildung 74: Code IRreceiveTask

Die Schnittstellen des Moduls IR.c sind in Abbildung 75 ersichtlich. Zum einen kann der Zustand der Leuchte über die Funktion Toggle_Lamp_State() getoggelt werden. Zum anderen kann der Infrarotbefehl für jede Taste der Apple IR Remote durch die aufgelisteten Funktionen versendet werden.

```
13  /*this function allows to send the IR command to toggle the lamp state*/
14  void Toggle_Lamp_State(void);
15
16  /*the following functions allow to send every command of the apple IR Remote*/
17  void IR_PowerOnOff_tx(void);
18  void IR_Up_tx(uint8_t count);
19  void IR_Down_tx(uint8_t count);
20  void IR_Left_tx(uint8_t count);
21  void IR_Right_tx(uint8_t count);
22  void IR_Menu_tx(void);
23  void IR_Sync_tx(void);
24
25  void IR_Init(void);
26  void IR_Deinit(void);
```

Abbildung 75: Schnittstellen des Moduls IR.c

6.4.5 SPI.c

Dieses Modul stellt mit der Funktion Get_Lamp_State() die Information zur Verfügung, ob die Leuchte aktuell ein- oder ausgeschaltet ist. Dazu wird der Stromsensor INA239 alle 20 Millisekunden über SPI ausgelesen. Anschliessend wird anhand eines Thresholdwerts entschieden, ob die Leuchte aktuell ein- oder ausgeschaltet ist. Der Thresholdwert wurde in Kapitel 7.4.1 bestimmt. Um ein Flickern der Variable LampState zu vermeiden, wurde eine Hysterese von 3 mA eingebaut. Details zur Auslesung des INA239 über das SPI Protokoll sind in Kapitel 6.6 zu finden. Die Abbildung 76 zeigt den Code des SPITasks.

```
83  //this task reads the lamp current every 20 ms and provides the LampState
84  static void SPITask(void * pv){
85
86      /* storage for the actual current in A */
87      float current;
88
89      /* read the current every 20 ms */
90      int CurrentReadFrequencyInMS = 20;
91
92      /* if the current is greater than this value the lamp is on */
93      const float current_threshold = 0.014;
94
95      /* hysterese to avoid flickering in LampState while lamp turns on or off */
96      const float hysterese = 0.003;
97
98      for(;;) {
99
100         current = spi_get_currentA();
101
102         if(current>(current_threshold+hysterese)){
103             LampState = true;
104         }else if(current<(current_threshold-hysterese)){
105             LampState = false;
106         }
107
108     vTaskDelay(pdMS_TO_TICKS(CurrentReadFrequencyInMS));
109     }
110 }
```

Abbildung 76: SPITask

In der Abbildung 77 sind die Schnittstellen des Moduls SPI.c ersichtlich. Neben den Funktionen zur Initialisierung und Deinitialisierung, hat das Modul nur eine weitere Schnittstelle nach aussen. Die Funktion Get_Lamp_State() gibt zurück, ob die Leuchte aktuell ein- oder ausgeschaltet ist.

```
11  bool Get_Lamp_State(void); /* this function returns the actual state of the lamp */
12
13  void SPI_Init(void);
14  void SPI_Deinit(void);
```

Abbildung 77: Schnittstellen des Moduls SPI.c

6.4.6 LED.c

Dieses Modul ermöglicht das Ansteuern des RGB-LEDs. Es ist zu beachten, dass die RGB-LED low active ist und deshalb die Signale invertiert sein müssen (0 = Ein , 1 = Aus). Der Befehl gpio_put(int PIN, bool state) ermöglicht das digitale Ansteuern eines GPIO Pins. Die Abbildung 78 zeigt den Code um die RGB-LED zu steuern. Neben der Funktion, um alle LEDs individuell mit einem Befehl zu steuern, sind auch Funktionen implementiert, um die einzelenen Farbanteile des RGB LEDs einzeln zu steuern oder die RGB-LED komplett ein- oder auszuschalten.

```
49  //this function allows to control the RGB LED
50  void RGB_Controll(bool R, bool G, bool B){
51      gpio_put(RGB_R, !R);
52      gpio_put(RGB_G, !G);
53      gpio_put(RGB_B, !B);
54  }
```

Abbildung 78: Code um die RGB LED zu steuern

In der Abbildung 79 sind die Schnittstellen des Moduls LED.c ersichtlich.

```
12  void RGB_ON(void);                          /* turn all parts of the RGB LED ON */
13  void RGB_OFF(void);                         /* turn all parts of the RGB LED OFF */
14  void RGB_Controll(bool R, bool G, bool B);  /* control the individual parts of the RGB LED */
15  void LED_RED(bool onORoff);                 /* control only the red part from the RGB LED*/
16  void LED_GREEN(bool onORoff);               /* control only the green part from the RGB LED*/
17  void LED_BLUE(bool onORoff);                /* control only the blue part from the RGB LED*/
18
19  void LED_Init(void);
20  void LED_Deinit(void);
```

Abbildung 79: Schnittstellen des Moduls LED.c

6.4.7 MyButton.c

Dieses Modul wird verwendet, um den Taster durch Software zu entprellen und die Information zur Verfügung zu stellen, ob der Taster gedrückt wurde. Die Idee für das Debouncing stammt aus der Quelle [23], wurde jedoch angepasst, um auf dem Pico zu funktionieren. Die Funktionsweise ist wie folgt: Es wurde ein repeating-timer aufgesetzt, welcher alle 10 Millisekunden die Callbackfunktion Debounce_Button1() aufruft. Die Callbackfunktion ist in Abbildung 80 zu sehen.

```
33  //debouncing on rising edge for button 1
34  static bool Debounce_Button1(struct repeating_timer *t){
35      DebounceState = (DebounceState<<1);
36      DebounceState = DebounceState | gpio_get(BUTTON_PIN);
37      DebounceState = DebounceState | 0xe000;
38      if(DebounceState == 0xf000){
39          BUTTON_PRESSED_FLAG = true;
40          DebounceState = 0;
41      }
42      return true;
43  }
```

Abbildung 80: Code Debounce Callbackfunktion

In der Callbackfunktion wird bei jedem Aufruf die Variable DebounceState um ein Bit nach links geschoben, um anschliessend an die Position des LSBs, den aktuellen Zustand des Tasters zu schreiben. Es ist zu beachten, dass der Taster low active ist und somit eine 0 an die Position des LSBs geschrieben wird, wenn der Taster geschlossen ist. Anschliessend wird die Variable Debounce State maskiert, um das Prellen herauszufiltern. Nachdem eine steigende Flanke (1111 0000 0000 0000) detektiert wird, wird das BUTTON_PRESSED_FLAG auf true gesetzt. Die Funktion Get_Button_State() muss periodisch aufgerufen werden, damit detektiert werden kann, ob das Flag gesetzt wurde oder nicht. Die Abbildung 81 zeigt die Funktion Get_Button_State().

```
24  bool GetButtonState(void){
25      if(BUTTON_PRESSED_FLAG){
26          BUTTON_PRESSED_FLAG = false;
27          return true;
28      }else{
29          return false;
30      }
31  }
```

Abbildung 81: Code Funktion Get_Button_State()

In der Abbildung 82 sind die Schnittstellen des Moduls mybutton.c ersichtlich.

```
13  /* this function returns true when a rising edge was detected */
14  bool GetButtonState(void);
15
16  void MyButton_Init(void);
17  void MyButton_Deinit(void);
```

Abbildung 82: Schnittstellen des Moduls mybutton.c

6.5 Dimmen der Stehleuchte mit der H-A-IR-Remote

In diesem Kapitel wird beschrieben, wie der Code zum Dimmen der Stehleuchte über die App funktioniert, welche Schwierigkeiten dabei auftreten und warum die Stehleuchte über die App nur in drei Stufen gedimmt werden kann.

Schwierigkeit: Dimmstufe der Stehleuchte über die Strommessung bestimmen
In Kapitel 7.4.3 wird gezeigt, dass die Information über die Dimmstufe der App, mit dem Mikrocontroller ausgewertet werden kann. Es hat sich jedoch gezeigt, dass es nicht ganz einfach ist, die Stehleuchte über die IR Schnittstelle in eine spezifische Dimmstufe zu bringen. Denn dazu muss die aktuelle Dimmstufe der Stehleuchte mit dem Mikrocontroller gemessen werden können. Die Messung der Dimmstufe über die Strommessung ist jedoch nicht möglich, da sich die Gottardo Stehleuchte in 25 unterschiedlichen Szenen befinden kann, wobei die Stromaufnahme in jeder Szene unterschiedlich ist. Die Information, in welcher Szene sich die Stehleuchte gerade befindet, steht dem Mikrocontroller nicht zur Verfügung. Aus diesem Grund kann die Dimmstufe nicht durch die Strommessung bestimmt werden.

Alternativlösung zur Einstellung der Dimmstufe der Stehleuchte
Da die Bestimmung der Dimmstufe der Stehleuchte über die Strommessung nicht funktioniert, wurde eine alternative Lösung erarbeitet. Bei der alternativen Lösung wird umgangen, dass die Dimmstufe der Stehleuchte gemessen werden muss. Denn wenn die Taste hochdimmen bei eingeschalteter Stehleuchte drei Mal betätigt wird, befindet sich die Stehleuchte immer in der höchsten Dimmstufe. So kann die Leuchte in einen definierten Zustand gebracht werden, von welchem aus durch das ein oder zweimalige betätigen der Taste Herunterdimmen eindeutig in die mittlere oder tiefste Dimmstufe gewechselt werden kann. Diese Lösung ermöglicht es die Stehleuchte in eine definierte Dimmstufe zu bringen, ohne die Dimmstufe zu messen.

Implementation dimmen
Im vorherigen Abschnitt wurde beschrieben, wie die Stehleuchte mit dem Mikrocontroller in drei definierte Dimmstufen gebracht werden kann. Damit mit der App zwischen diesen Dimmstufen gewechselt werden kann, wurde der Bereich in der App in die folgenden drei Dimmbereiche unterteilt.

- 1 - 10 % in der App -> Die Stehleuchte soll in der tiefsten Dimmstufe sein.
- 11 - 90 % in der App -> Die Stehleuchte soll in der mittleren Dimmstufe sein.
- 91 -100 % in der App -> Die Stehleuchte soll in der höchsten Dimmstufe sein.

Im Zustand LampIsON wird kontinuierlich überprüft, ob der Benutzer den Dimmbereich gewechselt hat. Wenn ein Übergang zwischen den drei Dimmbereichen festgestellt wird, wird die Stehleuchte zuerst durch das dreimalige Versenden des IR Befehls Taste Hochdimmen in den definierten Zustand höchste Dimmstufe gebracht. Anschliessend kann durch das 0 bis 2-malige Versenden des IR Befehls Taste Herunterdimmen in die vom Benutzer eingestellte Dimmstufe gewechselt werden. Wird die Dimmstufe der Stehleuchte mit der Fernbedienung oder den Bedientasten eingestellt, wird dies mit dieser Lösung nicht in der App angezeigt. Doch wenn der Benutzer in der App einen der drei Dimmbereiche einstellt, gelangt die Leuchte immer in den gewünschten Dimmbereich.

6.6 Strommessung mit dem INA239

In diesem Kapitel wird beschrieben, wie mit einem INA239 Stromsensor der Strom gemessen werden kann. Der INA239 misst eine Spannungsdifferenz über einem Shuntwiderstand, durch welchen der zu messende Strom fliesst. In einem ersten Schritt musste der Wert des Shuntwiderstandes berechnet werden. Dazu wurde die Formel 6.3 aus dem Datenblatt des INA239 verwendet. Die maximale Differenzspannung $V_{sense-max}$ über dem Shuntwiderstand ist dabei gegeben. Der maximal zu messende Strom wurde mit 5 Ampere festgelegt, da der maximale Leuchtenstrom 3.3 Ampere entspricht und noch etwas Reserve einberechnet wurde. Die folgenden Gleichungen zeigen die Berechnung des Shuntwiderstandes.

$$V_{sense-max} = 0.16384V \tag{6.1}$$

$$I_{max} = 5A \tag{6.2}$$

$$R_{shunt} < \frac{V_{sense-max}}{I_{max}} = 0.0327Ohm \tag{6.3}$$

$$R_{shunt} = 30mOhm \tag{6.4}$$

$$Pv_{max} = R_{shunt} \cdot I_{max}^2 = 0.75W \tag{6.5}$$

Es wurde berechnet, dass ein 30 Milliohm Shuntwiderstand verwendet werden muss, um einen Strom von 0 - 5 Ampere mit dem INA239 zu messen. Der Shuntwiderstand muss gemäss Berechnung 6.5 für eine Verlustleistung von mindestens 0.75 Watt ausgelegt sein.

Um den INA239 zu kalibrieren, muss der Wert des Shuntwiderstands gemäss der Formel 6.7 umgerechnet werden.

$$Current_{LSB} = \frac{I_{max}}{2^{15}} = 15.3 \cdot 10^{-5} \tag{6.6}$$

$$SHUNT_{CAL} = 819.2 \cdot 10^6 \cdot Current_{LSB} \cdot R_{shunt} = 3750 = 0xEA6 \tag{6.7}$$

Der berechnete Wert von 0xEA6 muss ins SHUNT_CAL Register (Adresse: 0x02) des INA239 geschrieben werden.
Nun ist der INA239 einsatzbereit und der Stom kann von CURRENT Register (Adresse: 0x07) gelesen werden. Die Formel 6.6 zeigt die Auflösung des INA239. Die Formel 6.8 zeigt wie der ausgelesene Wert vom CURRENT Register in einen Strom in Ampere umgerechnet werden kann.

$$Current[A] = Current_{LSB} \cdot CURRENT \tag{6.8}$$

In Kapitel 3.3 wurde gezeigt, wie die SPI-Kommunikation im Allgemeinen funktioniert. Das Lesen eines Registers des INA239 funktioniert wie folgt: Zuerst wird die CS Leitung auf Ground gezogen, um dem Sensor den Beginn der Kommunikation zu signalisieren. Anschliessend sendet der Master auf der Leitung MOSI die Adresse des zu lesenden Registers (6 Bit), gefolgt von einer Null und einer Eins, um dem INA 239 zu signalisieren, dass das Register gelesen werden soll. Im Anschluss sendet der INA239 den Inhalt des angeforderten Registers auf der MISO Leitung. Um die Kommunikation zu beenden, wird die CS Leitung wieder auf high gezogen. Die Abbildung 83 zeigt, wie ein Read Frame aufgebaut ist.

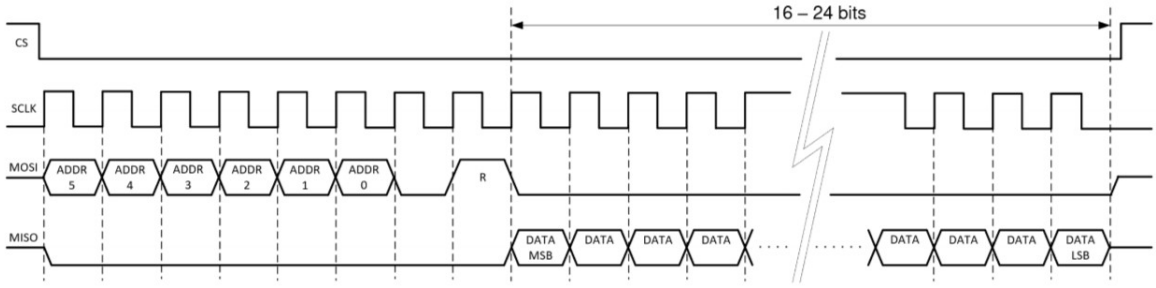


Abbildung 83: INA239 Read Frame [24, S. 18]

Das Beschreiben eines Registers des INA239 funktioniert wie folgt: Zuerst wird die CS Leitung auf Ground gezogen, um dem Sensor den Beginn der Kommunikation zu signalisieren. Anschliessend sendet der Master auf der Leitung MOSI die Adresse des zu beschreibenden Registers (6 Bit) gefolgt von zwei Nullen, um dem INA 239 zu signalisieren, dass das Register beschrieben werden soll. Im Anschluss sendet der Master die zu schreibenden Daten auf die MOSI Leitung und der INA 239 speichert diese im entsprechenden Register. Um die Kommunikation zu beenden, wird die CS Leitung wieder auf high gezogen. Die Abbildung 83 zeigt wie ein Write Frame aufgebaut ist.

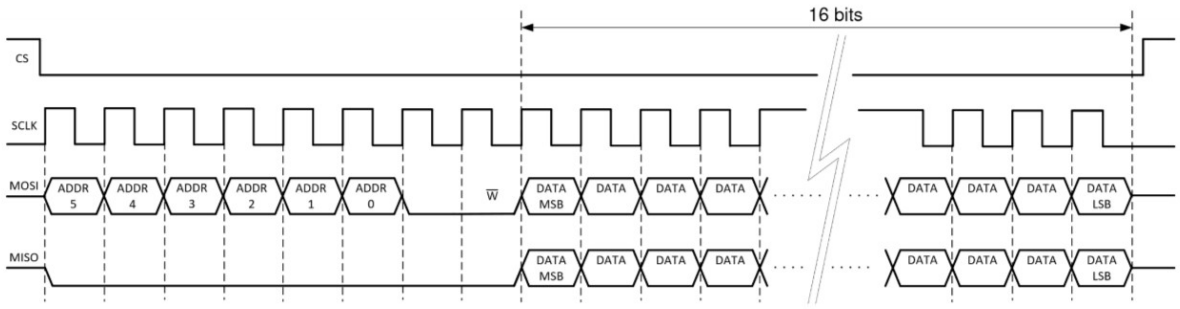


Abbildung 84: INA239 Write Frame [24, S. 18]

6.7 Source Review

Der Sourcecode der H-A-IR-Remote wurde mit dem betreuenden Dozenten Erich Styger besprochen. Dabei wurden die folgenden Punkte angemerkt:

- Globale Variablen und Funktionen, die nur im Modul gebraucht werden, sollten static sein.
- Globale Variablen sollten nur dann benutzt werden, wenn es sich nicht vermeiden lässt.
- Es ist sinnvoll, die Funktionen in den Headerfiles kurz zu kommentieren.
- // ist ein C++ Kommentar, damit die Software auch in einer reinen C Umgebung funktioniert, sollten besser C Kommentare verwendet werden (/* ..... */).
- C Bibliotheken werden mit eckigen Klammern inkludiert und sonstige Bibliotheken mit Hochkommas
- Variablen, welche sich zur Laufzeit nicht verändern, sollten mit dem Schlüsselwort const versehen werden.
- Es macht keinen Sinn, bei der Fehlerbehandlung einen Rückgabewert zu generieren, wenn nicht auf diesen reagiert wird. Als einfache Fehlerbehandlung kann eine unendliche For-Schleife aufgerufen werden (Hilfreich beim Debuggen).

Die wichtigsten besprochenen Punkte wurden in der Software umgesetzt. Einige Punkte konnten aus zeitlichen Gründen nicht umgesetzt werden, wie zum Beispiel, dass keine C++ Kommentare verwendet werden sollten.

6.8 Debugging das Picos mit einem Segger JLink und Eclipse

In diesem Kapitel wird beschrieben, wie vorgegangen wurde, um die Software auf dem Raspberry Pi Pico mit der Eclipse IDE zu übersetzen und zu debuggen.

6.8.1 Aufbau zum Debuggen des Picos

Um das Raspberry Pi Pico zu debuggen, wurde ein Segger JLink EDU verwendet. Um erste Software Module zu testen, bevor die Hardware V1.1 bestellt und bestückt war, wurde ein Testaufbau zusammengestellt. Dazu wurde ein Kabel angefertigt, durch welches sich das Pico mit dem JLink Debugger verbinden liess. Die Tabelle 1 zeigt, wie die Pins verbunden wurden. Die Information, wie das Pico mit dem JLink verbunden werden muss, wurde aus Quelle [25] entnommen.

JTAG Pin	Kabelfarbe	Pico Pin
1	Blau	3V3(Out)
4	Schwarz	GND
7	Grün	SWDIO
9	Braun	SWCLK
19	Rot	VBUS

Tabelle 1: Kabelverbindung Pico - JLink

Es ist zu beachten, dass beim anfertigen des Kabels vorgesehen war, das Pico über den JLink zu speisen. Diese Möglichkeit wurde jedoch nicht genutzt, stattdessen wurde das Pico über ein Mickro USB Kabel mit Strom versorgt. Die Abbildung 85 zeigt den Testaufbau zum Debuggen des Picos mit dem Segger JLink.

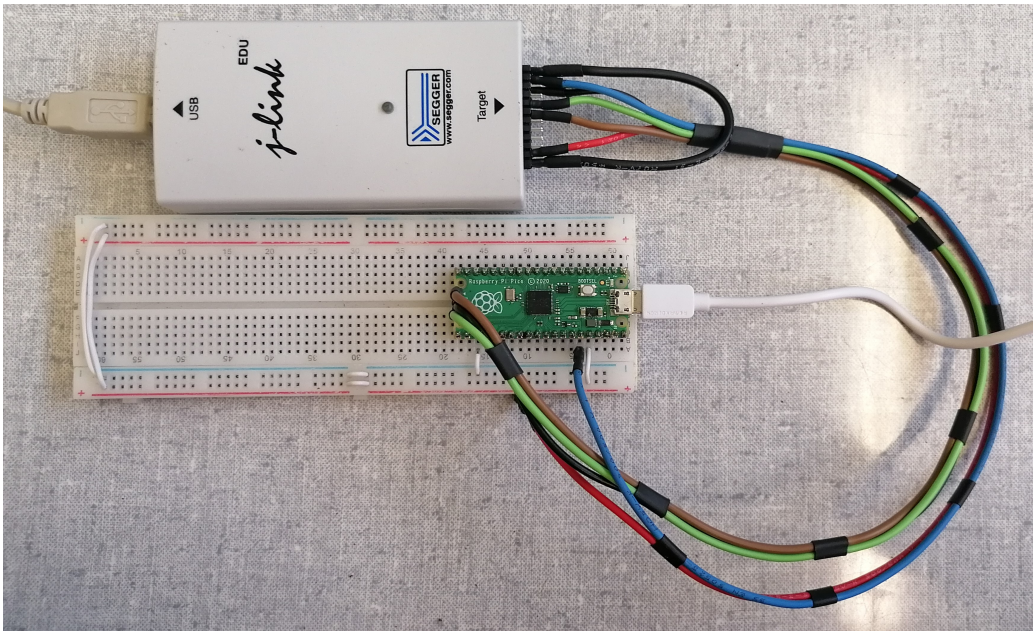


Abbildung 85: Testaufbau um das Pico zu debuggen

Um die Hardware V2.1 zu debuggen, wird ein Cortex SWD Adapter benutzt. Somit ist es möglich, ein 10-poliges Kabel zu verwenden, welches direkt bei der Hardware eingesteckt werden kann. Beim Debuggen der Hardware V2.1 wird das selber angefertigte Verbindungskabel nicht mehr benötigt. Die Abbildung 86 zeigt den Aufbau, um die Hardware V2.1 zu debuggen.

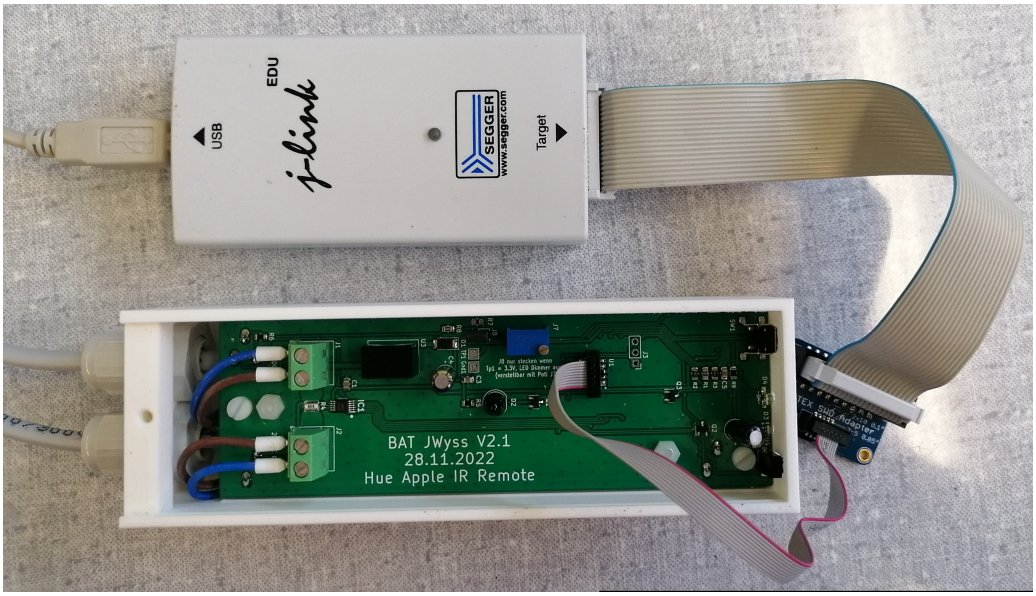


Abbildung 86: Debugging der finalen Hardware

6.8.2 Blink Programm auf dem Pico

Der erste Schritt beim Programmieren des Picos war, ein einfaches Programm für das Pico mit der Eclipse IDE zu erstellen, zu übersetzen und mit einem JLink EDU zu debuggen. Dazu wurde die Anleitung aus Quelle [25] befolgt. Am Anfang wird in der Anleitung beschrieben, wie das Pico mit dem JLink Debugger verbunden werden muss. Anschliessend wird aufgezeigt, welche Tools installiert werden müssen. Es wird auch aufgezeigt, welche Einstellungen in der IDE vorgenommen werden müssen, um das Programm mit dem JLink auf dem Pico zu debuggen und wie das CMakelists.txt - File aufgebaut sein muss. Die Abbildung 87 zeigt ein einfaches Programm, welches die Onboard-LED des Picos blinken lässt. Durch die Anleitung konnte dieses Programm erfolgreich auf dem Pico gedebuggt werden.

```
#include "hardware/gpio.h"
#include "pico/stdlib.h"

#define LED_PIN   (25)

int main(void) {
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);
    while (true) {
        gpio_put(LED_PIN, 1);
        sleep_ms(250);
        gpio_put(LED_PIN, 0);
        sleep_ms(250);
    }
    return 0;
}
```

Abbildung 87: Blink Code für das Pico [25]

6.8.3 Free RTOS auf dem Pico

Da bei der H-A-IR-Remote mehrere Dinge gleichzeitig passieren (Leuchtenstrom messen, LED Dimmer Ausgang messen, FSM betreiben) wird mit FreeRTOS gearbeitet. Dabei handelt es sich um ein opensource Real Time Operating System für Embedded-Systeme. Um das FreeRTOS auf dem Pico zu verwenden, wurde ein Pico Projekt vom betreuenden Dozenten Erich Styger übernommen, in welchem das FreeRTOS bereits eingebettet war. Das Projekt wurde umbenannt und es wurden sämtliche Softwaremodule entfernt, welche nicht für das FreeRTOS benötigt werden. In diesem Projekt wurden anschliessend sämtliche Softwaremodule für diese Arbeit erstellt und am Ende dann auch die Gesamtsoftware implementiert.

7 Testberichte

In diesem Kapitel sind alle Tests, welche während der Projektarbeit gemacht wurden, aufgeführt.

7.1 Test: Lichtsensor LDR

In diesem Test geht es darum, ob es realistisch ist den Lampenstatus mit einem Lichtsensor zu erfassen. Für einen einfachen Test wurde ein Light Dependent Resistor verwendet. Dieser wurde mit Klebeband direkt an der Leuchte montiert und der Widerstand wurde mit einem Ohmmeter gemessen.

Abbildung 88: Test Lichtsensor LDR

Messergebnisse in verschiedenen Situationen:

Situation	Widerstand LDR
Totale Dunkelheit (LDR mit Finger bedeckt)	1MOhm
Sensor auf Tisch bei normaler Raumbeleuchtung	2.5 kOhm
Leuchte aus	9.9 kOhm
Leuchte aus, Störlichtsimulation mit Taschenlampe	3.2 kOhm
Niedrigste Dimmstufe Szene 1.5	4.8 kOhm
Mittlere Dimmstufe Szene 1.5	650 Ohm
Höchste Dimmstufe Szene 1.5	330 Ohm
Niedrigste Dimmstufe Szene 3.4	4.33 kOhm
Mittlere Dimmstufe Szene 3.4	570 Ohm
Höchste Dimmstufe Szene 3.4	300 Ohm

Tabelle 2: Messergebnisse LDR

Dieser Test ergab, dass es sehr schwierig ist, mit einem LDR den Unterschied zwischen Leuchte an in der tiefsten Dimmstufe und Leuchte aus zu detektieren. Das Problem ist, dass die Messung durch Sonneneinstrahlung verfälscht werden kann und somit kein eindeutiger Rückschluss möglich ist, ob die Leuchte an oder aus ist.

7.2 Test: Stromverbrauch Leuchte

In diesem Test geht es darum, ob der Lampenstatus eindeutig anhand des Stromverbrauchs der Leuchte bestimmt werden kann. Dazu wird der Strom auf der 230-Volt- sowie auf der 24-Volt-Seite des Netzteils der Leuchte gemessen. Um den Stromverbrauch auf der 24-Volt-Seite zu messen, wurde ein Zwischenstecker aus einem NC3Fxx Stecker und einer Kupplung hergestellt (siehe Abbildung 89). Auf der 230 Volt Seite wurde mit einem Energiemessgerät gemessen (siehe Abbildung 90).

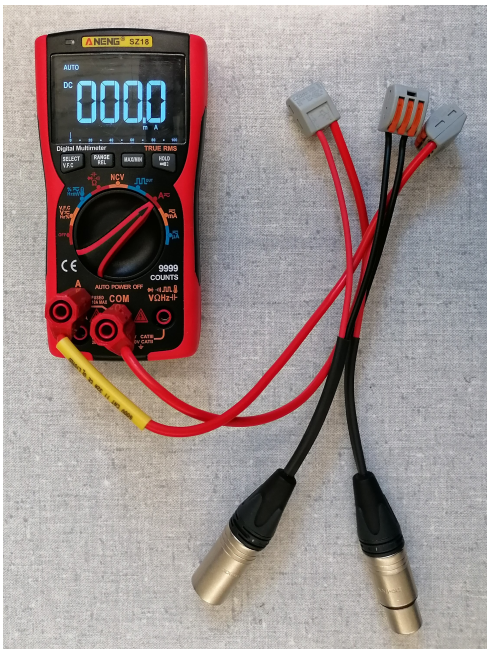


Abbildung 89: Zwischenstecker zur Strommessung der Leuchte

Abbildung 90: Energiemessgerät PM30 Mandolyn zur Strommessung der Leuchte

Messergebnisse:

Strom [mA]	Standby	Stufe 1	Stufe 2	Stufe 3
Szene 1.5	6.3	54	863	3303
Szene 3.4	6.3	49	838	3163
Szene 4.3	6.3	50	764	2761
Szene 5.4	6.3	36	100	308

Tabelle 3: Messergebnisse 24V Seite

Strom [mA]	Standby	Stufe 1	Stufe 2	Stufe 3
Szene 1.5	3.1	50	130	370
Szene 3.4	3.1	40	130	360
Szene 4.3	3.1	50	120	320
Szene 5.4	3.1	40	140	70

Tabelle 4: Messergebnisse 230V Seite

Die Messung hat gezeigt, dass auf der 24-Volt-Seite eine minimale Stromdifferenz von 30 mA und auf der 230-Volt-Seite eine minimale Stromdifferenz von 36 mA zwischen Standby und tiefste Dimmstufe vorliegen. Somit ist es eindeutig möglich, am Stromverbrauch zu erkennen, ob die Leuchte an oder aus ist. Ausserdem hat der Test ergeben, dass anhand der Strommessung nicht auf die Dimmstufe der Stehleuchte geschlossen werden kann, da der Stromverbrauch bei unterschiedlichen Szenen in der gleichen Dimmstufe variiert. Um anhand des Stromes auf die genaue Dimmstufe zu schliessen, muss die Information bekannt sein, in welcher Dimmstufe sich die Leuchte gerade befindet und jede der 25 Dimmstufen müsste einzeln ausgemessen werden.

7.3 Test: ZigBee LED Dimmer

In diesem Kapitel wird beschrieben, wie der LED Dimmer getestet wurde. Das Ziel ist, dass nach diesen Tests klar ist ob, und wie der LED Dimmer mit einer Mikrocontrollerschaltung kombiniert werden kann.

7.3.1 Test: Einbindung in Philips Hue

Da es sich bei dem ZigBee LED Dimmer nicht um ein originales Produkt von Philips Hue handelt, wurde in diesem Test überprüft, ob sich der LED Dimmer in Philips Hue integrieren lässt. Dazu wurde der LED Dimmer mit 24 Volt gespiesen. Anschliessend wurde es versucht, den LED Dimmer in das Philips Hue System zu integrieren. Die Abbildung 91 zeigt den Versuchsaufbau.

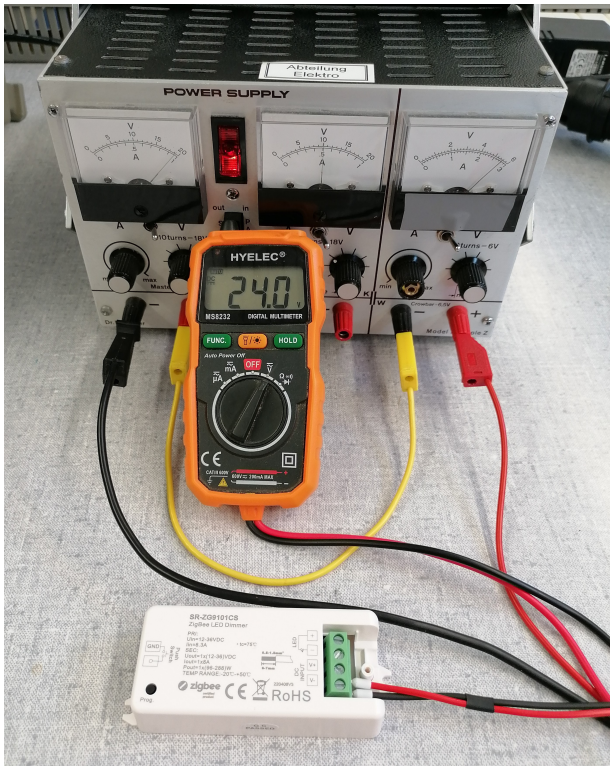


Abbildung 91: Speisung ZigBee LED Dimmer mit 24 V

Der Versuch hat ergeben, dass sich der Dimmaktor wie eine originale Hue Leuchte ins System integrieren lässt. Wird der Dimmaktor in der App geschaltet, wechselt der Ausgang von 0 auf 24 Volt. Es wurde ebenfalls festgestellt, dass die Ausgangsspannung des Dimmaktors der Eingangsspannung entspricht.

7.3.2 Test: Rückmeldung über den Taster

In diesem Test geht es darum, herauszufinden, ob Schaltungen, welche mit dem Taster durchgeführt werden, auch in der Hue App angezeigt werden. Zwischen den Tasterklemmen herrscht eine Spannung von 3.3 Volt unabhängig von der Eingangsspannung. Der Dimmaktor erkennt also, wenn der Tasteranschluss auf Ground gezogen wird. Für den Test wurde ein Taster zwischen die beiden Kontakte geschaltet (siehe Abbildung 93).

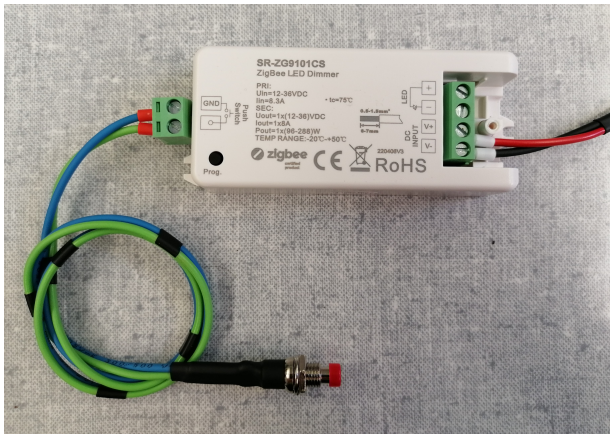


Abbildung 92: LED Dimmaktor mit angeschlossenem Taster

Der Versuch hat gezeigt, dass der Dimmaktor direkt nach dem Betätigen des Tasters den Ausgang ein- oder ausschaltet, die Aktualisierung des Zustandes in der App jedoch durchschnittlich 3.5 Sekunden verzögert ist. Die Dimmstufe des LED Dimmers kann durch das Drücken und Halten des Tasters verändert werden. Auch wenn über den Taster gedimmt wird, wird die Dimmstufe nach durchschnittlich 3.5 Sekunden in der App aktualisiert. Die Verzögerung der Aktualisierung kann nicht beeinflusst werden.

7.3.3 Test: Stromverbrauch Standby

In diesem Test geht es darum, den Stromverbrauch des LED Dimmmers zu ermitteln. Dazu wurde der LED Dimmer mit 24 Volt gespiesen und der Stromverbrauch gemessen.

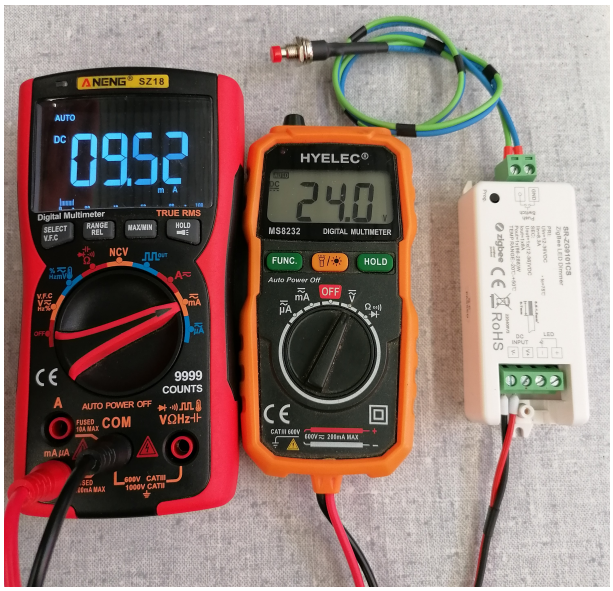


Abbildung 93: LED Dimmaktor mit angeschlossenem Taster

Die Messung hat einen Strom von etwa 10 mA ergeben, was bei der Speisung mit 24 Volt einer Leistung von 240 mW entspricht.

7.3.4 Test: Speisung über Leuchtennetzteil

Bei diesem Test geht es darum, ob der LED Dimmer sicher über das Leuchtennetzteil betrieben werden kann. Beim Leuchtennetzteil handelt es sich um eine 24-Volt-Konstantspannungsquelle mit einer Nennausgangsleistung von 96 Watt. In Test in Kapitel 7.2 wurde festgestellt, dass die Leuchte maximal 3.3 Ampere bei 24 Volt bezieht, was einer Leistung von 80 Watt entspricht. Somit dürfen zusätzlich zur Leuchte noch eine maximale Leistung von 16 Watt vom Leuchtennetzteil bezogen werden. Da der LED Dimmer nur 0.24 Watt bezieht (siehe Kapitel 7.3.3), wird die Nennleistung des Netzgeräts beim parallelen Betrieb von LED Dimmer und Leuchte zu keinem Zeitpunkt überschritten. Die Abbildung 94 zeigt den Betrieb des LED Dimmers am Leuchtennetzteil über einen Zwischenstecker.

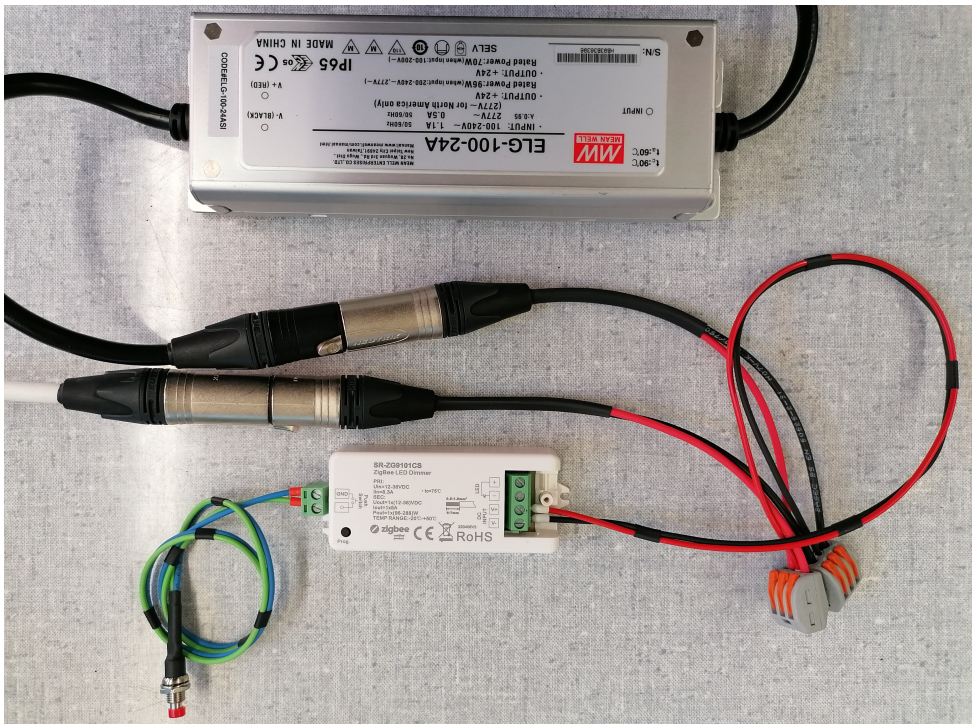


Abbildung 94: Betrieb LED Dimmer mit Leuchtennetzteil

Der Test hat gezeigt, dass der LED Dimmer ohne Probleme am Leuchtennetzteil betrieben werden kann. Ausserdem können noch bis zu 15.5 Watt am Leuchtennetzteil angehängt werden, ohne die Nennleistung des Netzteils zu überschreiten.

7.3.5 Test: Tastereingang mit Microcontroller betätigen

In diesem Test geht es darum herauszufinden, wie der Tastereingang des LED Dimmers mit dem Mikrocontroller betätigt werden kann. Die Idee besteht darin, einen N-Mosfet wie einen Schalter zu verwenden, um so den Tastereingang elektronisch zu betätigen. Die Abbildung 95 zeigt den Versuchsaufbau als Blockdiagramm. Die Schaltung wurde auf einem Breadboard aufgesteckt, um sie zu testen. Es wurde ein BS170-D75Z N-Mosfet verwendet, da dieser eine Gate Thresholdspannung von 2.1 Volt aufweist und somit bei 3.3 Volt nur noch einen Widerstand von etwa 1.2 Ohm aufweist. Für den Test wurde eine Funktion aufgesetzt, welche den GPIO Ausgang für eine gewisse Zeit auf High und anschliessend wieder auf Low schaltet. Es

wurde getestet, ob der LED Dimmer schaltet und wie lange der GPIO Pin mindestens auf High geschaltet sein muss, damit der LED Dimmer noch schaltet.

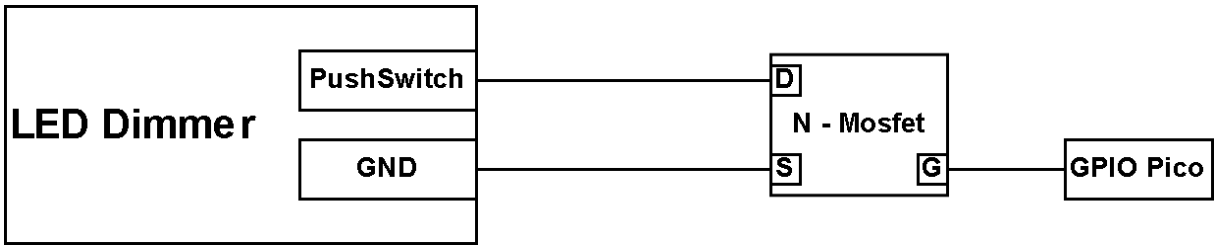


Abbildung 95: Blockschaltbild Versuchsaufbau

Der Test hat ergeben, dass der Tastereingang über einem N-Mosfet betätigt werden kann. Damit das Signal vom LED Dimmer erkannt wird, muss der N-Mosfet für 20 Millisekunden durchgeschaltet werden.

7.3.6 Test: Generierung eines 0 - 3.3 Volt Signals aus dem LED Dimmer Ausgang

In diesem Kapitel geht es darum ein 0 - 3.3 Volt Signal aus dem LED Dimmer Ausgang zu generieren. Dazu wurde in einem ersten Schritt der Ausgang des LED Dimmers mit einem Multimeter ausgemessen. Beim Ausmessen hat sich herausgestellt, das der Ausgang des LED Dimmers belastet sein muss, um ein Ausgangssignal zu generieren. Ausserdem wurde festgestellt, dass sich das Potential vom Anschluss LED - gegenüber Ground der Speisung je nach eingestelltem Wert in der App zwischen 0 und 24 Volt bewegt, wobei das Potenzial von LED + immer bei 24 Volt gegenüber dem Ground der Speisung bleibt. Abbildung 96 zeigt den LED Dimmer mit den für dieses Kapitel relevanten Ein- und Ausgängen. Die Abbildung 97 zeigt, wie der eingestellte Wert in der App und das Potenzial von LED - zusammenhängen.

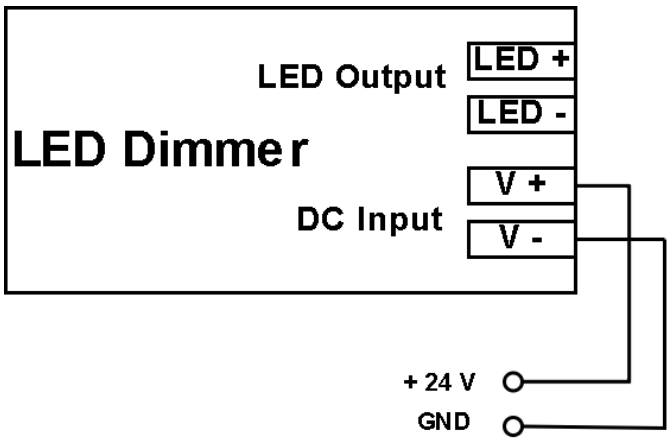


Abbildung 96: Ein- und Ausgänge des LED Dimmers

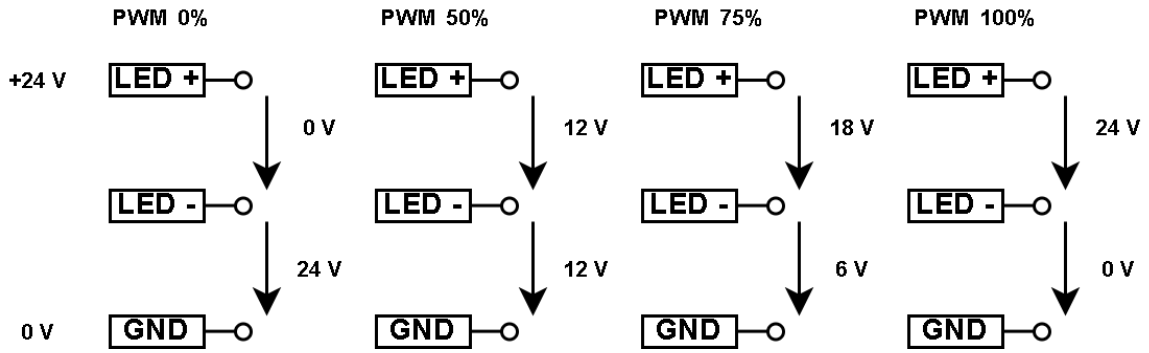


Abbildung 97: Potenzial des LED Dimmer Anschlusses LED -

Durch die Erkentnisse, wie der eingestellte Wert in der App und das Potenzial von LED - zusammenhängen, konnte eine Schaltung entwickelt werden. Der Ausgang des LED Dimmers wird mit einem 100 kOhm Widerstand belastet und es wird ein Spannungsteiler zwischen LED - und GND gehängt. Die Schaltung wurde so ausgelegt, dass der Ausgang des Spannungsteilers gegenüber dem Ground der Speisung ein Potenzial von 0 - 3.3 Volt aufweist. Wäre dies nicht der Fall, könnte der Ausgang des Spannungsteilers nicht mit dem Mikrocontroller gemessen werden, da der Ground des Mikrocontrollers dem Ground der Speisung entspricht. Die Messschaltung ist in Abbildung 98 zu sehen.

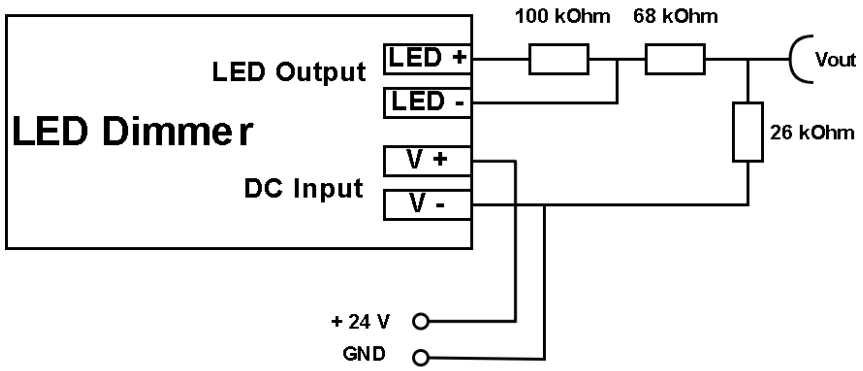


Abbildung 98: Schaltung zur Generierung eines 0 - 3.3 Volt PWM Signals

Mit dieser Schaltung konnte ein invertiertes PWM-Signal von 0 - 3.3 Volt generiert werden. Das Signal ist invertiert, da bei 100 % in der App 0 Volt zwischen dem Ausgang des Spannungsteiler und Ground anliegen und bei 0 % in der App 3.3 Volt zwischen dem Ausgang des Spannungsteilers und Ground anliegen. Durch eine Messung mit einem Oszilloskop wurde festgestellt, dass das PWM-Signal eine Frequenz von 1000 Herz aufweist.

7.3.7 Lösungsansätze um das Ausgangssignal des LED Dimmers mit dem Pico zu messen

In diesem Kapitel werden verschiedenen Ansätze beschrieben, um das Ausgangssignal des LED Dimmers mit dem Mikrocontroller zu messen. Das Ziel ist es, mit dem Mikrocontroller die Dimmstufe in der App auszulesen.

Ansatz 1: Spannungsteiler und Tiefpassfilter
Für diesen Ansatz wird die in Kapitel 7.3.6 entwickelte Schaltung mit einem Tiefpassfilter erweitert (siehe Abbildung 99).

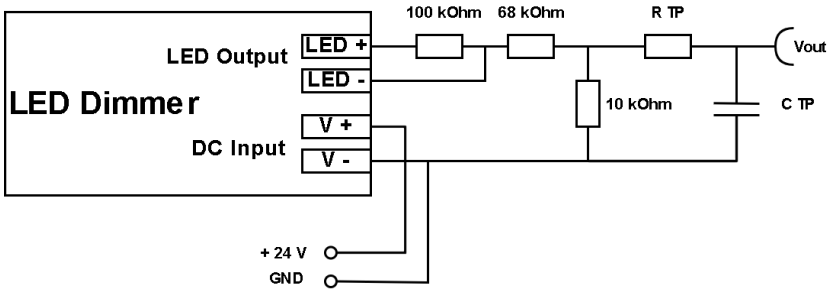


Abbildung 99: Durch Tiefpassfilter erweiterte Messchaltung

Das Ziel ist dabei, den PWM Ausgang des Spannungsteilers mit dem Tiefpassfilters zu Glätten und anschliessend über einen Analogpin des Mikrocontrollers einzulesen. Dieser Ansatz wurde auf der Hardware V1 nur als Reservelösung vorgesehen und wurde aus diesem Grund nicht weiter getestet.

Ansatz 2: Signal im LED Dimmer abgreifen
Für diesen Ansatz wurde der LED Dimmer geöffnet und nach einem 0 - 3.3 Volt PWM Signal gesucht, welches der Dimmstufe in der App entspricht. Nach einigen Messungen wurde ein solcher Anschluss gefunden und mit einer Litze nach aussen geführt. Die Abbildung 100 zeigt den offenen LED Dimmer mit den herausgeführten Litzen. Um den Dutycycle des herausgeführten PWM-Signals mit dem Mikrocontroller zu messen, wurde die Funktion measure_duty_cycle() aus der Quelle [21] verwendet.

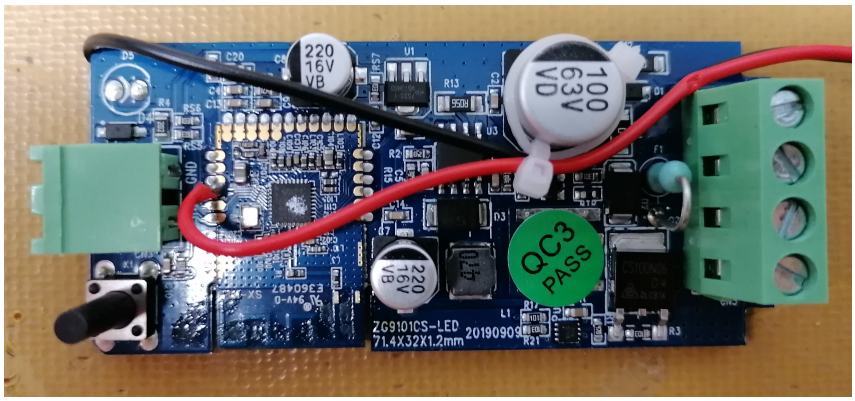


Abbildung 100: geöffneter LED Dimmer mit angelöteter Litze

Dieser Ansatz funktioniert sehr zuverlässig, der grosse Nachteil ist jedoch, dass der LED Dimmer geöffnet und verändert werden muss. Dennoch wurde dieser Ansatz auf der Hardware V1 als Lösung zur Messung des LED Dimmer Ausgangs verwendet.

Ansatz 3: Spannungsteiler und PWM Measure Funktion
Die Idee zu diesem Lösungsansatz kam kurz vor der Bestellung der Hardware V2. Das durch die

in Kapitel 7.3.6 entworfene Schaltung generierte PWM-Signal kann ebenfalls mit der Funktion measure_duty_cycle() gemessen werden. Somit entfällt das Öffnen und Verändern des LED Dimmers. Dieser Lösungsansatz konnte mit der Hardware V1 getestet werden, indem der Tiefpassfilter überbrückt wurde. Der Testbericht ist in Kapitel 7.4.3 zu finden.

7.4 Test bestücktes PCB V1.1

Nach dem Bestücken des PCB V1.1 wurde dieses getestet. Nachfolgend ist aufgeführt, wie die einzelnen Funktionalitäten der Hardware getestet wurden. Ausserdem wird beschrieben, welche Fehler bemerkt wurden und wie diese behoben werden konnten.

7.4.1 Test: Strommessung mit dem INA239

Um zu prüfen, ob mit dem INA239 über SPI kommuniziert werden kann, wurde zuerst die Device-ID über SPI ausgelesen. Das Auslesen der Device-ID hat wegen Fehlern in der Software nicht von Anfang an geklappt. Durch einen Logicanalyzer konnten die Fehler jedoch behoben werden. Um den Strom auszulesen, musste der INA 239 gemäss Kapitel 6.6 konfiguriert werden. Anschliessend wurde die Strommessung der Stehleuchte aus Kapitel 7.2 erneut durchgeführt, wobei der Strom hier mit dem INA239 gemessen wurde. Die Tabelle 5 zeigt die Messergebnisse mit dem INA 239 als Vergleich werden in Tabelle 6 die Messergebnisse der Strommessung mit dem Amperemeter aus Kapitel 7.2 aufgeführt.

Strom [mA]	Standby	Stufe 1	Stufe 2	Stufe 3
Szene 1.5	6.4	56.3	862	3343
Szene 3.4	6.4	50	834	3203
Szene 4.3	6.4	47.6	757	2789
Szene 5.4	6.4	80.1	245	291.1

Tabelle 5: Messergebnisse INA239

Strom [mA]	Standby	Stufe 1	Stufe 2	Stufe 3
Szene 1.5	6.3	54	863	3303
Szene 3.4	6.3	49	838	3163
Szene 4.3	6.3	50	764	2761
Szene 5.4	6.3	36	100	308

Tabelle 6: Messergebnisse mit Multimeter

Der Test hat zum einen gezeigt, dass die Strommessung mit dem INA239 funktioniert. Ausserdem hat sich gezeigt, dass mit dem INA239 sehr genau gemessen werden kann. Die grossen Differenzen bei der Messung in Szene 5.4 lassen sich dadurch erklären, dass in dieser Szene die Farbe des oberen Lichtsystems kontinuierlich zwischen rot, blau und grün wechselt. Somit ist diese Messung nicht aussagekräftig. Die Messung zeigt auch, dass die Leuchte ab einem Strom von 14 mA sicher eingeschaltet ist (Stand-by-Strom mal zwei).

7.4.2 Test: IR Signale versenden und empfangen

In diesem Test wurde geprüft, ob die Hardware in der Lage ist, IR Signale zu versenden und empfangen. Es konnte nur die THT IR Diode getestet werden, da die SMD IR Diode nicht geliefert wurde. Die Schaltung zum Empfangen von Infrarot Signalen hat wie erwartet funktioniert. Beim Versenden von IR Signalen gab es jedoch Schwierigkeiten. Es hat sich herausgestellt, dass der Mikrocontroller zu wenig Strom liefert, um ein ausreichend starkes Infrarot Signal zu generieren. Mit der in Kapitel 5.3.7 beschriebenen Schaltung konnte die Leuchte nur ein- und ausgeschaltet werden, wenn die Hardware nicht weiter als 50 cm vom IR Empfänger der Leuchte entfernt platziert wurde. Ausserdem hat sich herausgestellt, dass der Empfänger der Stehleuchte auf der Oberseite der Leuchte angebracht ist. Wenn die Hardware also am Ständer der Leuchte befestigt ist, muss das IR Signal genügend stark sein, um von der Raumdecke reflektiert zu werden. Um die Intensität des IR Signals zu verstärken, wurde die Schaltung angepasst, dass der Strom durch die IR LED höher ist. Gemäss Datenblatt dürfen bis zu 1000 mA Pulsströme durch die THT IR LED fliessen. Die Schaltung wurde also so angepasst, dass die IR LED den Strom direkt von dem DC-DC-Wandler bezieht. Dies wurde erreicht, indem ein N-Mosfet das Schalten der IR Diode übernimmt und der Mikrocontroller nur noch den Mosfet ein- und ausschalten muss. Die Abbildung 101 zeigt, wie die Hardware V1.1 angepasst wurde. Um den DC-DC-Wandler zu entlasten, wurden ein 1 uF Keramik- und ein 15 uF Elektrolytkondensator parallel zur IR Diode und Vorwiderstand gehängt.

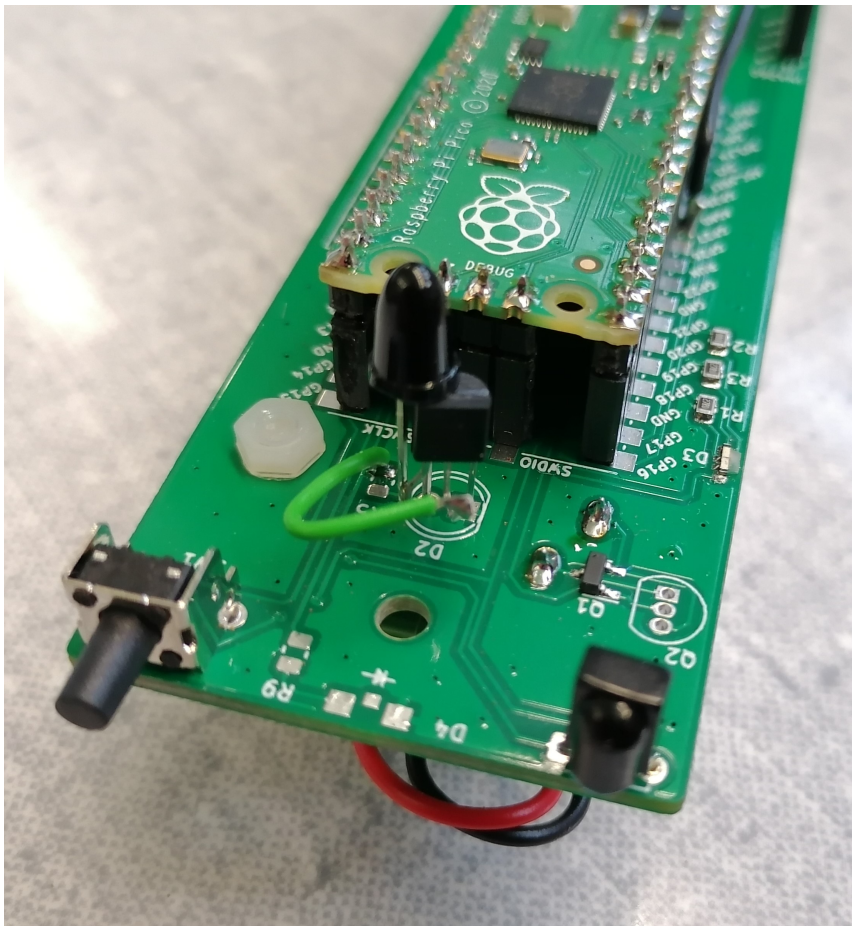


Abbildung 101: Anpassung der Hardware, um die Intensität des IR Signals zu erhöhen

7.4.3 Test: Kommunikation LED Dimmer

In diesem Test wurde geprüft, ob die Kommunikation mit dem LED Dimmer funktioniert. Die Schaltung zur Betätigung des Tasters hat einwandfrei funktioniert und musste nicht angepasst werden. In Kapitel 7.3.7 wurden verschiedene Ansätze zur Ausmessung des LED Dimmer Ausganges vorgestellt, wobei Ansatz 1 (Spannungsteiler und Tiefpassfilter) als Reserve und Ansatz 2 (Signal im LED Dimmer abgreifen), auf der Hardware V1.1 realisiert wurden. Der Ansatz 2 (Signal im LED Dimmer abgreifen) wurde mit der Hardware V1.1 getestet und hat wie erwartet funktioniert. Da bei diesem Ansatz jedoch der LED Dimmer geöffnet werden musste, wurde mit der Hardware V1.1 auch der Ansatz 3 (Spannungsteiler und PWM-Measure Funktion) getestet. Dazu wurde der auf dem PCB vorgesehene Tiefpassfilter überbrückt. Es hat sich herausgestellt, dass das Pico 2 PWM Channels besitzt, wobei nur mit dem Channel B PWM Signale gemessen werden können. Deshalb musste mit einer Drahtbrücke der Ausgang des Spannungsteilers auf den GPIO27 (Channel B) anstelle des GPIO26 (Channel A) geführt werden. So konnte der Ansatz getestet werden. Die Abbildung 102 zeigt die Anpassung auf der Hardware.

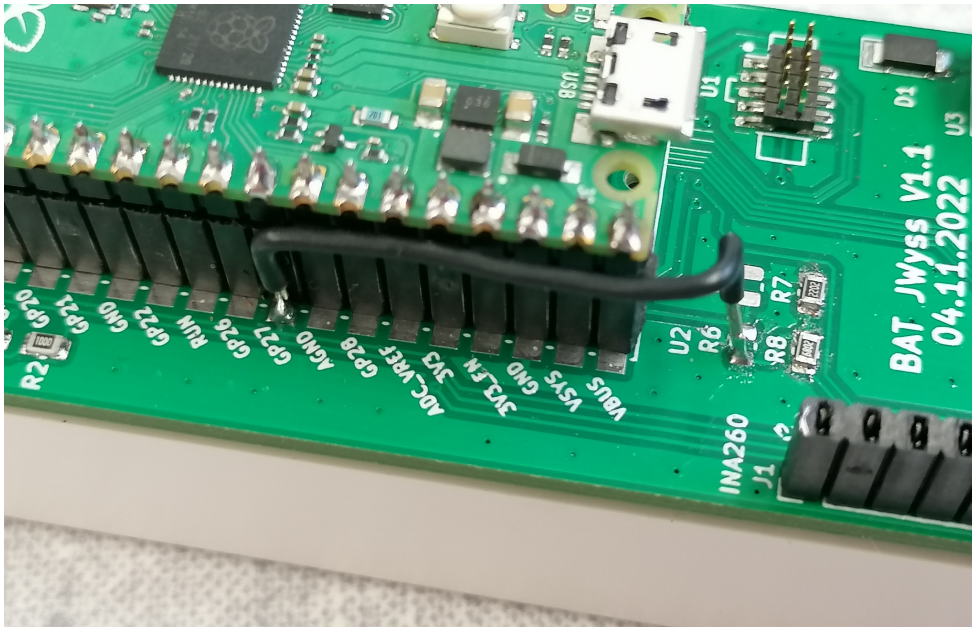


Abbildung 102: Anpassung um Ansatz 3 zu testen

Der Test hat ergeben, dass die Ausmessung des LED Dimmer Ausganges über den Spannungsteiler und die PWM Measure Funktion funktioniert. Es wurde ausgemessen, wie die Dimmstufe in der App mit dem gemessenen invertierten Dutycycle zusammenhängen, damit diese Information beim Programmieren zur Verfügung steht. Die Tabelle 7 zeigt die Messergebnisse.

Dimmmstufe in App [%]	gemessener und invertierter Dutycycle [%]
100	100
90	98
80	89
70	80
60	70
50	62
40	51
30	43
20	33
10	25
1	16
Aus	8

Tabelle 7: Messergebnisse Dutycycle des LED Dimmer Ausgangs über den Spannungsteiler

Durch diese Messung kann die Schwelle bestimmt werden, bei welchem gemessenen Wert der Zustand in der App aus ist. Dazu wird der Wert zwischen den gemessenen Werten aus und 1 % genommen. Für Werte kleiner als 12 % ist der Zustand in der App aus und für Werte grösser als 12 % ist der Zustand in der App an.

7.4.4 Test: User Interface

Beim Testen der RGB LED wurde festgestellt, dass diese verkehrt herum aufgelötet wurde. Dieser Fehler konnte jedoch einfach behoben werden. Danach hat die RGB LED wie erwartet funktioniert. Beim Testen des Tasters gab es keine Probleme.

7.5 IR Signale der Apple IR Remote aufzeichnen

Damit die Signale der Apple IR Remote versendet werden können, müssen diese zuerst ausgemessen werden. Dazu wurde der IR Receiver mit 3.3 Volt gespiesen und der Ausgang mit einem Logicanalyzer aufgezeichnet (siehe Abbildung 103).

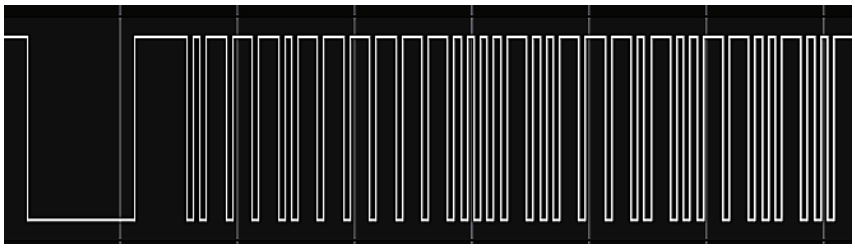


Abbildung 103: Aufgezeichnetes IR Signal mit Logicanalyzer

Zuerst wurde das Signal von Hand ausgewertet. Dazu wurden die Bit-Definitionen des NEC-Protokolls aus Kapitel 3.1.1 verwendet. Da dies sehr aufwändig und fehleranfällig war, wurde nach einer anderen Lösung gesucht. Es hat sich herausgestellt, dass Arduino ein Beispielprojekt zur Verfügung stellt, welches empfangene NEC-Befehle über die serielle Schnittstelle

ausgibt. Dies hat die Auswertung der IR Signale beschleunigt. Die Tabelle 8 zeigt die empfangenen Signale für die jeweiligen Tasten und den Bitweise gespiegelten Befehl, welcher zum Versenden der Befehle benötigt wird.

Taste	Befehl in Hex	Befehl gespiegelt
Power	77E13A64 77E1A064	265C87EE 260587EE
Hochdimmen	77E15064	260A87EE
Herunterdimmen	77E13064	260C87EE
Lichtszene Rückwärts	77E19064	260987EE
Lichtszene Vorwärts	77E16064	260687EE
Szenengruppe	77E1C064	260387EE
Synchronisieren/ Start	77E1FA64 77E1A064	265F87EE 260587EE
Fernbedienung Koppeln	7E14064	260287E0
Fernbedienung Entkoppeln	7E12064	260487E0

Tabelle 8: Apple IR Remote Frames

Um die Befehle zu spiegeln, wurde eine Funktion in C implementiert, welche diese Aufgabe übernimmt. Diese Funktion ist in der Abbildung 104 zu sehen.

```
156 static uint32_t mirror_frame(uint32_t original){
157     uint32_t target = 0x00000000;
158     uint32_t temp = 0x00000000;
159             for(uint32_t i=0;i<32;i++){
160                 temp = original & (1<<i);
161                 temp = (temp>>i);
162                 target = target | (temp<<(31-i));
163             }
164     return target;
165 }
```

Abbildung 104: Code zum Spiegeln von IR Frames

7.6 Test: Energieverbrauch Hardware V2.1

In diesem Test geht es darum, den Energieverbrauch der Hardware (HW) zu bestimmen. Dazu wird der aufgenommene Strom der Leuchte in der Szene 1.5 (tiefste Dimmstufe) einmal mit und einmal ohne Hardware bestimmt. Die Differenz der Messungen entsprechen dem von der Hardware aufgenommenen Strom. Mit der Formel $P = U * I$ kann anschliessend in die aufgenommene Leistung der Hardware berechnet werde. Die Ergebnisse des Tests sind in der Tabelle 9 aufgeführt.

HW + Leuchte [mA]	nur Leuchte [mA]	Differenz [mA]	Leistung HW [mW]
72.2	54.2	18	432

Tabelle 9: Strommessung Hardware

Der Test hat ergeben, dass die Hardware 0.43 Watt bezieht.

8 Bedienungsanleitung H-A-IR-Remote

In diesem Kapitel befindet sich die Bedienungsanleitung für die H-A-IR-Remote.

8.1 Legende

1. IR Receiver
2. Status LED
3. IR Diode 1
4. Schnittstelle für Software-Updates
5. Bedientaster
6. Taster für Software-Updates
7. IR Diode 2

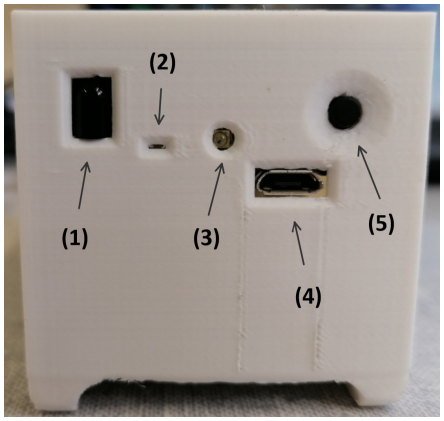


Abbildung 105: Frontansicht H-A-IR-Remote

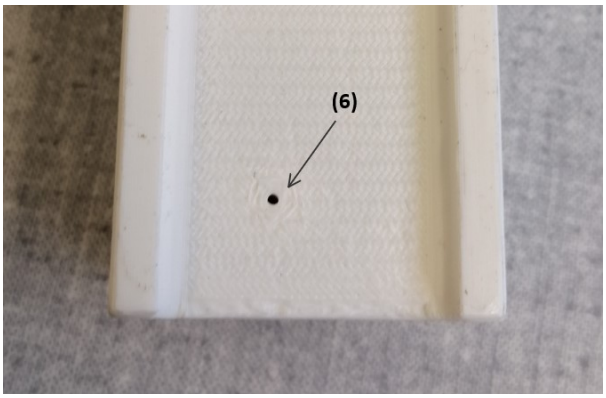


Abbildung 106: Unteransicht H-A-IR-Remote

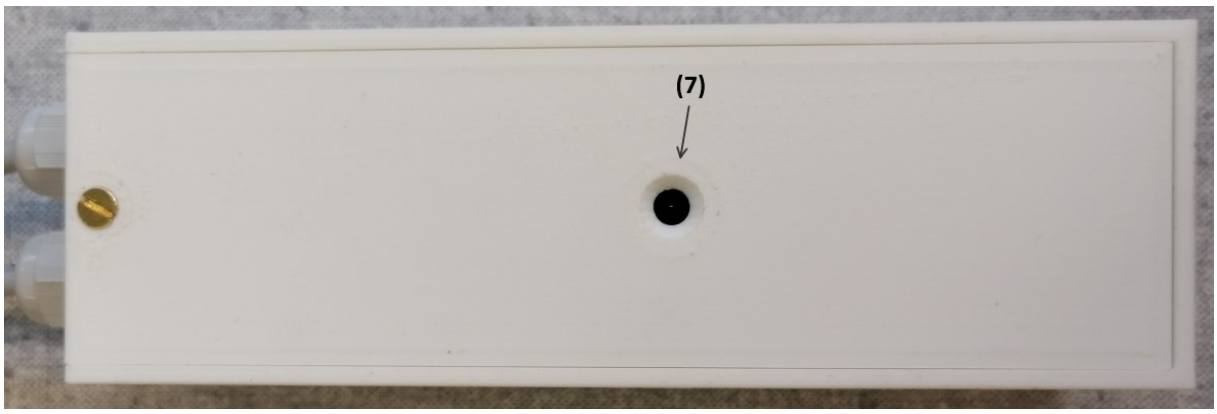


Abbildung 107: Draufsicht H-A-IR-Remote

8.2 Inbetriebnahme

Führen Sie die folgenden Schritte aus, um die H-A-IR-Remote in Betrieb zu nehmen.

- Trennen Sie die Gottardo Stehleuchte vom Stromnetz.
- Montieren Sie die H-A-IR-Remote mit den mitgelieferten Klettbändern auf einer Höhe von ungefähr 30 Zentimeter auf der Rückseite des Leuchtenständers. Der Kabelanschluss muss dabei nach unten zeigen.
- Die Klettbänder müssen den Leuchtenständer und die H-A-IR-Remote umschliessen und gut angezogen werden.
- Trennen Sie die Stehleuchte vom Netzteil (Stecker ausziehen).
- Verbinden Sie die Stecker der H-A-IR-Remote mit dem Netzteil und der Leuchte.
- Stecken Sie die Gottardo Stehleuchte nun wieder ein.
- Gehen Sie in die Philips Hue App und suchen sie nach neuen Geräten.
- Binden Sie das gefundene Gerät in die Hue App ein.
- Nun kann die Stehleuchte über die App ein- und ausgeschaltet werden. Wird die Stehleuchte ausserhalb der App ein- oder ausgeschaltet, wird der Zustand in der App aktualisiert.

8.3 Bedienungshinweise

Bitte beachten Sie die folgenden Bedienungshinweise.

- Die Leuchte kann mit der App in drei Stufen gedimmt werden. 100 % in der App entsprechen der maximalen Helligkeit der Stehleuchte, 50 % entsprechen einer mittleren Helligkeit der Stehleuchte und 1 % entspricht der minimalen Helligkeit der Stehleuchte.
- Wird die H-A-IR-Remote mit Strom versorgt, leuchtet die Status LED kurz weiss auf, anschliessend zeigt sie den Status der Leuchte an (grün = Leuchte an, rot = Leuchte aus).
- Wird die Stehleuchte ein- oder ausgeschaltet, leuchtet die Status LED kurz blau auf und zeigt anschliessend wieder den Zustand der Leuchte an.
- Mit dem Bedientaster kann die Stehleuchte ein- und ausgeschaltet werden.
- Es ist zu beachten, dass die Aktualisierung des Leuchtenstatuses in der App etwa 3.5 Sekunden dauert, wenn die Leuchte ausserhalb der App geschaltet wird.
- Bei einem Wiedereinschalten nach einem Stromunterbruch, befindet sich die Stehleuchte wieder im gleichen Zustand wie vor dem Stromunterbruch.
- Neben der Montage der H-A-IR-Remote am Leuchtenständer kann diese auch auf einem Tisch platziert werden, dabei ist zu beachten, dass die IR Diode 2 nach oben schaut und der horizontale Abstand zur Stehleuchte nicht mehr als zwei Meter beträgt.

8.4 Softwareupdate

Befolgen Sie die nachfolgenden Schritte, um ein Software-Update auf die H-A-IR-Remote zu laden.

- Trennen Sie die Gottardo Stehleuchte inklusive H-A-IR-Remote vom Stromnetz.
- Verbinden Sie die Schnittstelle für Software-Updates durch ein Mikro USB Kabel mit ihrem PC.
- Betätigen Sie den Taster für Software-Updates mit einer aufgebogenen Büroklammer, während sie die Gottardo Stehleuchte inklusive H-A-IR-Remote wieder ans Stromnetz anschliessen.
- Nun wird die H-A-IR-Remote auf ihrem PC als Laufwerk angezeigt, auf welches sie das Software-Update laden können.
- Trennen Sie das Mikro USB Kabel von der H-A-IR-Remote.

8.5 Fehlerbehandlung

- **Status LED leuchtet nicht:** Stellen sie sicher, dass Stecker der Stehleuchte und der H-A-IR-Remote richtig eingesteckt sind.
- **Status LED leuchtet dauerhaft blau:** Trennen sie die Stehleuchte vom Stromnetz und stecken sie sie nach zehn Sekunden wieder ein.
- **Die Stehleuchte reagiert nicht auf Befehle aus der App:** Stellen sie sicher, dass keine Gegenstände die beiden IR Dioden bedecken.

9 Schlussteil

In diesem Kapitel werden in einem ersten Teil die Anforderungen, welche zu Beginn der Arbeit definiert wurden, überprüft. Anschliessend werden die Resultate der Arbeit in einem fachlichen Fazit aufgezeigt und diskutiert. In einem nächsten Teil wird ein Ausblick gemacht, wie diese Arbeit weitergeführt werden könnte. Am Ende wird dann noch ein persönliches Fazit gezogen und eine Danksagung gemacht.

9.1 Erfüllungsgrad der Anforderungen

In diesem Kapitel werden die in Kapitel 1.3 definierten Anforderungen überprüft.

1. **Einfache Integration in das Philips Hue System:** Die H-A-IR-Remote kann durch den LED Dimmer wie ein originales Produkt in das Philips Hue System integriert werden, somit ist diese Anforderung erfüllt.

2. **Leuchte ein- und ausschalten:** Die Gottardo Stehleuchte kann in der Philips Hue App ein- und ausgeschaltet werden, somit ist diese Anforderung erfüllt.

3. **Leuchte Dimmen:** Die Gottardo Stehleuchte kann in der Philips Hue App in drei Stufen gedimmt werden, somit ist diese Anforderung erfüllt.

4. **Anzeige des aktuellen Leuchtenzustandes in der App:** Diese Anforderung wurde nur teilweise erfüllt. Der Test der Hardware hat ergeben, dass es durchschnittlich 3.5 Sekunden dauert, bis der Zustand in der App aktualisiert wird, es wurden aber bis zu 9 Sekunden Verzögerung gemessen. Somit liegt die Verzögerungszeit in gewissen Fällen über den geforderten 5 Sekunden. Doch nach dieser Verzögerungszeit ist der Zustand in der App immer aktuell, womit dieser Teil der Anforderung erfüllt ist.

5. **Keine Funktionseinschränkung der Stehleuchte:** Die Bedienung der Gottardo Stehleuchte mit der originalen Fernbedienung oder den Bedientasten an der Leuchte wird durch die H-A-IR-Remote in keinster Weise beeinträchtigt oder verändert, somit ist diese Anforderung erfüllt.

6. **Energieverbrauch:** Die H-A-IR-Remote benötigt im Betrieb 0.43 Watt. Da dies unter den geforderten 1 Watt liegt ist diese Anforderung erfüllt.

7. **Bedienungsanleitung:** Es ist eine Bedienungsanleitung mit den geforderten Punkten vorhanden, somit ist diese Anforderung erfüllt.

8. **Einfache Montage:** Die H-A-IR-Remote konnte innerhalb von einer Minute, ohne Werkzeuge durch einen Mitstudenten an der Stehleuchte montiert werden, somit ist diese Anforderung erfüllt.

9. **Einfache Demontage:** Die H-A-IR-Remote kann innerhalb von einer Minute, ohne Werkzeuge rückstandslos von der Stehleuchte demontiert werden, somit ist diese Anforderung erfüllt.

10. **Keine Veränderung der Stehleuchte:** Die Stehleuchte wird durch die Montage der H-A-IR-Remote in keinster Weise verändert oder beschädigt, deshalb ist diese Anforderung erfüllt.

11. **Softwareupdate:** Durch die Bedienungsanleitung kann ein Laie ein Software-Update auf die H-A-IR-Remote laden, somit ist diese Anforderung erfüllt.

12. **Anwendbarkeit auf ähnliche Leuchten:** Die H-A-IR-Remote ist mit Leuchten, welche ähnlich sind wie die Gottardostehleuchte, kompatibel. Es kann sein, dass bei einer anderen Leuchte die Montage am Ständer nicht möglich ist, dann muss die H-A-IR-Remote auf einem Tisch in der Nähe der Leuchte platziert werden. Dennoch ist die Anforderung erfüllt.

9.2 Fachliches Fazit

Das Ziel der vorliegenden Arbeit war es, eine Gottardo Stehleuchte in das Philips Hue Automatisierungssystem zu integrieren. Dieses Ziel darf als erfüllt betrachtet werden, denn der entwickelte Prototyp funktioniert einwandfrei und erfüllt die Anforderungen. Mit der entwickelten H-A-IR-Remote kann eine nicht smarte Gottardo Stehleuchte schnell und einfach in ein Philips Hue System integriert werden. Eine dafür angefertigte Bedienungsanleitung führt den Benutzer durch die Montage und zeigt auf, wie die Bedienung funktioniert. Die Montage ist dabei innerhalb von einer Minute möglich, da die H-A-IR-Remote sehr einfach mit zwei Klettbändern am Leuchtenständer montiert werden kann und anschliesssend nur noch zwischen Leuchtennetzteil und Stehleuchte gesteckt werden muss. Die Einbindung in das Philips Hue System unterscheidet sich dabei nicht von der Einbindung eines originalen Hue Produktes. Mit der H-A-IR-Remote ist es dem Benutzer möglich, die Gottardo Stehleuchte in der Philips Hue App ein- und auszuschalten und zu dimmen. Die Gottardo Stehleuchte kann aber weiterhin auch ganz normal mit der bestehenden Infrarotfernbedienung bedient werden, wobei der Zustand der Leuchte in der App durch die H-A-IR-Remote stets aktuell gehalten wird. Für den Fall, dass die IR Signale nicht bei der Leuchte ankommen, wurde eine Fehlerbehandlung vorgesehen, damit das System nicht in einen undefinierten Zustand gerät.
Bei einer kritischen Betrachtung des Projekts gibt es aber doch einige Punkte, die verbessert werden können. Beispielsweise kann die Leuchte mit der Hue App nicht wie mit der normalen Fernbedienung stufenlos, sondern nur in drei Stufen gedimmt werden und wenn die Dimmstufe in der App eingestellt wird, muss die Leuchte immer kurz in die höchste Dimmstufe gebracht werden. Ausserdem wird in der App nichts angezeigt, wenn die Dimmstufe der Stehleuchte mit der Fernbedienung geändert wird. Ein weiterer Punkt zur Verbesserung ist der folgende: Um die Hardware mit 12 - 36 Volt DC betreiben zu können, wurde ein Potentiometer im Spannungsteiler zur Messung des LED Dimmer Ausganges eingesetzt. Hier hätte auch eine andere Lösung eingesetzt werden können, bei welcher die H-A-IR-Remote nicht von Hand kalibriert werden muss.
Die Ergebnisse dieser Arbeit beschränken sich nicht nur auf die entwickelte Hardware und Software, denn auch das erarbeitete Fachwissen im Bereich der Infrarotkommunikation mit einem Mikrocontroller ist sehr wertvoll für zukünftige Arbeiten in diesem Bereich. Auch die in dieser Arbeit aufgezeigten Möglichkeiten, wie ein eigenes Gerät in ein Philips Hue System integriert werden kann, liefern wichtige Informationen, welche den Rechercheaufwand für eine Folgearbeit reduzieren können. Durch diese Arbeit ist auch eine Anleitung für die Benutzung

und Auslesung des INA239 Stromsensors entstanden. Wenn in zukünftigen Projekten eine Strommessung erforderlich ist, kann dann ebenfalls auf diese Arbeit zurückgegriffen werden. Als abschliessendes Fazit kann gesagt werden, dass die Ergebnisse dieser Arbeit sehr zufriedenstellend sind und in zukünftigen Projekten in diesem Bereich als Grundlage verwendet werden können.

9.3 Ausblick und Erweiterungsideen

In diesem Kapitel soll die Perspektive etwas erweitert werden, indem ein Ausblick gegeben wird, wie die H-A-IR-Remote weiter entwickelt werden kann.

Miniaturisierung der Hardware
Für eine Folgearbeit könnte ein Ziel sein, die Hardware zu miniaturisieren. Dazu könnte in einer nächsten Iteration der RP2040 Chip direkt auf dem PCB platziert werden. Dadurch können die Bauteile des Picos selber platziert werden und nicht benötigte GPIO Pins nehmen keinen Platz weg. Es könnte auch mit dem Hersteller des LED Dimmers abgeklärt werden, ob das ZigBee Modul aus dem LED Dimmer ausgebaut werden darf. Auch dieses könnte dann direkt auf dem PCB platziert werden, was wiederum Platz spart.

Reduktion des Energieverbrauchs
Der Energiebedarf der Hardware liegt mit 0.43 Watt unter dem Wert von 1 Watt aus den Anforderungen. Wenn beispielsweise 1000 Exemplare der H-A-IR-Remote im Einsatz sind, werden permanent 430 Watt verschwendet. Deshalb könnte in einer Folgearbeit untersucht werden, wie sich der Energieverbrauch auf ein absolutes Minimum reduzieren lässt. Dazu müsste die Software und auch die Hardware durch geeignete Massnahmen energieeffizienter gestaltet werden.

Ersatz Potentiometer
In der aktuellen Hardware ist ein Potentiometer verbaut, damit der Ausgang des Spannungsteilers am Ausgang des LED Dimmers je nach Speisung der Hardware auf 3.3 Volt für den Mikrocontroller eingestellt werden kann. Diese Lösung ist nicht optimal, da das Potentiometer von Hand eingestellt werden muss. Werden 1000 Exemplare der H-A-IR-Remote produziert, müssen auch 1000 Potentiometer eingestellt werden. In einer Folgearbeit könnte eine bessere Lösung für das Messen des Ausgangs des LED Dimmers gesucht werden. Eine Option wäre es, sich auf die üblichen Spannungen (12,24,36 Volt) zu beschränken und einen Jumper vorzusehen, um dies zu konfigurieren. Es könnte auch ein programmierbarer Widerstand im Spannungsteiler eingesetzt werden, wodurch eine Einstellung in der Software gemacht werden kann. Es könnte auch eine analoge Schaltung, welche immer eine Ausgangsspannung von 3.3 Volt liefert, eingesetzt werden.

Stufenloses Dimmen und Rückmeldung in die App
Mit der aktuellen Hardware kann die Leuchte in der App nur in drei Stufen gedimmt werden und bei jeder Änderung der Dimmstufe in der App wird die Stehleuchte kurz in die höchste Dimmstufe versetzt. Der Grund dafür liegt darin, dass mit der aktuellen Hardware nicht eindeutig auf die Dimmstufe der Stehleuchte geschlossen werden kann. Dadurch können Dimmaktionen, welche mit der Fernbedienung durchgeführt werden, auch nicht in der App ange-

zeigt werden. In einer Folgearbeit könnte die Hardware so angepasst und erweitert werden, dass die Leuchte in der App stufenlos gedimmt werden kann. Dazu muss eine Möglichkeit gefunden werden, die aktuelle Dimmstufe eindeutig mit dem Mikrocontroller messen zu können. Es könnte beispielsweise die Idee der Statuserkennung mit Lichtsensoren genauer untersucht werden.

Setup Prozess
Auch bei der Software gibt es Potenzial für Erweiterungen. Die aktuelle Software muss durch einen Programmierer umgeschrieben werden, wenn die Hardware mit einer ähnlichen, aber nicht identischen Stehleuchte verwendet wird. Dazu muss beispielsweise die Stromschwelle, anhand welcher bestimmt wird, ob die Leuchte an oder aus ist, neu ausgemessen und programmiert werden. Auch wenn die ähnliche Leuchte eine Fernbedienung mit leicht unterschiedlichen Befehlen verwendet, muss dieses ausgemessen und neu programmiert werden. Durch eine Anpassung des Programms in einer Folgearbeit könnte der Benutzer bei der Erstinbetriebnahme durch einen Setup Prozess geführt werden, bei welchem die Stromschwelle und die relevanten Signale der Fernbedienung geteacht und auf dem Flash Speicher der Picos abgespeichert werden. Der Ablauf könnte folgendermassen aussehen: In der Bedienungsanleitung steht, dass wenn das RGB-LED rot blinkt die Stehleuchte in den Standbymodus versetzt werden soll. Anschliessend bestätigt der Benutzer dies, indem er den Taster betätigt. Nach dem Tastendruck liest der Mikrocontroller den aktuellen Strom der Leuchte aus und kann so die Stromschwelle ermitteln. Auch die IR Befehle der Fernbedienung können nach diesem Prinzip geteacht werden. Wenn beim Setup etwas schief geht oder die H-A-IR-Remote mit einer anderen Leuchte verwendet werden soll, könnte implementiert werden, dass sich die H-A-IR-Remote durch das Betätigen des Tasters für zehn Sekunden zurücksetzt und der Setup Prozess erneut durchlaufen wird.

Universelles Konzept
Eine mögliche Erweiterung besteht auch darin, die H-A-IR-Remote so universell zu gestalten, dass sämtliche Geräte mit einer IR Fernbedienung in das Philips Hue System integriert werden können. Dazu müsste die Hardware so angepasst werden, dass sie auf der 230-Volt-Seite zwischen das zu steuernde Gerät und die Steckdose gesteckt werden kann. Die Strommessung müsste so angepasst werden, dass der Gerätestrom auf der 230-Volt-Seite gemessen werden kann. Durch diese Anpassung ist der Anwendungsbereich nicht mehr nur auf Bürostehleuchten beschränkt. So könnten beispielsweise auch eine Stereoanlage oder der Fernseher in das Philips Hue System integriert werden.

Geräte ohne Fernbedienung mit einer Fernbedienung steuerbar machen
Im Moment wird der IR Receiver auf der Hardware nicht genutzt. Doch die Hardware könnte so umgebaut werden, dass Leuchten und andere Haushaltsgeräte, welche über keine Infrarotschnittstelle verfügen, mit einer Infrarotfernbedienung und der Hue App gesteuert werden können. Dazu müsste die Hardware nur mit einem ein 230-Volt-Relais erweitert und etwas angepasst werden.

9.4 Persönliches Fazit

Für mich persönlich war diese Arbeit ein voller Erfolg. Es war eine sehr spannende Aufgabe, bei welcher ich sehr viel in verschiedensten Bereichen dazugelernt habe. Am Anfang erschien mir die Aufgabe nicht besonders komplex zu sein, doch beim Erarbeiten des Lösungskonzeptes hat sich schnell gezeigt, dass die Aufgabe sehr vielschichtig ist und viele unterschiedliche Aufgaben erledigt werden müssen, um das Projekt erfolgreich abzuschliessen. Vor dieser Arbeit habe ich beispielsweise nie hinterfragt, wie eine Infrarotfernbedienung funktioniert und plötzlich muss ich selber eine bauen. Ausserdem konnte ich sehr viel im Studium erlerntes praktisch umsetzen und dadurch festigen. Natürlich hat nicht immer alles auf Anhieb so funktioniert wie gedacht, doch aus Fehlern lernt man bekanntlicherweise am Besten. Während der Arbeit stand ich sehr oft vor Herausforderungen und musste mich in viele Themenbereiche einarbeiten. Ich habe beispielsweise noch nie mit LaTeX gearbeitet und musste mich deshalb während der Arbeit mit dem Programm vertraut machen. Auch die Arbeit mit dem Raspberry Pi Pico war komplett neu für mich. Beim Programmieren musste ich viel in der Dokumentation des Picos nachschlagen und zum Teil auch im Internet nach Lösungen suchen. Durch diese Arbeit habe ich auch gelernt, wie man strukturiert an ein grosses Projekt herangeht. Bisher habe ich noch an keinem so grossen Projekt gearbeitet, bei welchem von der Konzeption bis zur Realisierung alles dabei war. Durch diese Arbeit wurde mir auch klar, wie wichtig es ist, verschiedene Lösungen zu betrachten und nicht einfach das Erstbeste zu nehmen. Sonst wäre ich nämlich nie darauf gekommen, die Statuserkennung der Leuchte über die Strommessung zu machen.
Ich konnte sehr selbstständig arbeiten. Dennoch war ich sehr froh über die Tipps und Anregungen von Herr Erich Styger in den wöchentlichen Besprechungen. Diese haben mir oft weitergeholfen und liessen mich Probleme von einer anderen Seite betrachten. Ich habe den benötigten Zeitaufwand für diese Arbeit ein wenig unterschätzt. Doch durch den zu Beginn der Arbeit erstellten Zeitplan und die definierten Meilensteine konnte ich die Arbeit so einteilen, dass ich strukturiert am Projekt arbeiten konnte. Dies führte schlussendlich zu einem erfolgreichen Abschluss der Arbeit.

9.5 Danksagung

Am Ende dieser Arbeit möchte ich mich recht herzlich bei all denjenigen bedanken, die mich bei meiner Bachelorarbeit unterstützt und motiviert haben. Zuerst gebührt mein Dank, Herr Erich Stgyer, welcher meine Bachelorarbeit betreut und begutachtet hat. Für die hilfreichen Anregungen und die konstruktive Kritik möchte ich mich herzlich bedanken. Des Weiteren möchte ich mich bei den Mitarbeitern der Elektrowerkstatt an der Hochschule Horw bedanken, da sie mir bei Fragen zu der Bedienung der Gerätschaften in der Werkstatt immer geholfen haben. Ebenfalls möchte ich mich bei meiner Mutter und meiner Freundin bedanken, welche beide meine Arbeit gegengelesen haben. Abschliessend möchte ich mich auch bei meinen Mitstudenten bedanken, welche ich bei elektrotechnischen Problemen immer um Rat fragen konnte und mit welchen ich mich auch sonst gut über die Arbeit austauschen konnte.

10 Verzeichnisse

Abbildungsverzeichnis

1 Blockdiagramm Aufgabenstellung . . . 1
2 Gottardo Stehleuchte . . . 2
3 Bedienfeld [1, S. 4] . . . 3
4 Apple IR Remote [1, S. 4] . . . 3
5 Szenengruppen und Lichtszenen [1, S. 15] . . . 4
6 Philips Hue Bridge . . . 5
7 Philips Hue Birne . . . 5
8 Blockschaltbild Philips Hue System . . . 5
9 Meilensteine und Zeitplan . . . 7
10 Vereinfachte Darstellung eines demodulierten IR Signals [5] . . . 9
11 Beispiel für ein NEC-Frame [6] . . . 10
12 NEC-Frame mit Repeat Code (Taste wird gedrückt gehalten) [6] . . . 10
13 Schaltung um IR Signale mit dem Pico zu versenden und empfangen [5] . . . 11
14 Sternförmige SPI Verbindung [8] . . . 12
15 Übersicht Teilbereiche . . . 13
16 ZigBee Modul aus einem IKEA Tradfri Leuchtmittel [10] . . . 15
17 PCB eines originalen Philips Hue Leuchtmittels [11] . . . 16
18 Testschaltung mit JN5168 Breakoutboard [12] . . . 17
19 Hue Wall Switch [13] . . . 17
20 Hue Smart Plug [14] . . . 18
21 Hue Schaltaktor mit 230 Volt AC Speisung [15] . . . 18
22 Light Dependent Resistor . . . 19
23 TSL 2591 I2C Lichtsensor [16] . . . 20
24 Multianzeige Gottardo Stehleuchte [1, S. 5] . . . 20
25 OpenMV Camera [17] . . . 21
26 Lösungskonzept Hardware . . . 24
27 Raspberry Pi Pico Pinout [18] . . . 26
28 ZigBee LED Dimmer . . . 27
29 INA239 Stromsensor . . . 27
30 INA260 Stromsensor [19] . . . 28
31 DC-DC-Wandler TSR 1-2450 . . . 28
32 IR Diode SFH 4545 . . . 29
33 IR Diode APDA3020F3C-P22 [20] . . . 29
34 IR-Receiver TSOP38238 . . . 30
35 RGB LED - R6GHBHW . . . 30
36 KiCad Benutzeroberfläche . . . 31
37 Raspberry Pi Pico . . . 32
38 Debug Schnittstelle . . . 33
39 Speisung Pico . . . 33
40 Strommessung . . . 34
41 Anschlussklemme . . . 34
42 Kommunikation mit ZigBee LED Dimmer . . . 35

43 IR rx / tx . . . 36
44 User Interface . . . 37
45 Montagelöcher . . . 37
46 Vorderseite PCB V1.1 . . . 38
47 Rückseite PCB V1.1 . . . 38
48 Vorderseite bestücktes PCB V1.1 . . . 41
49 Rückseite bestücktes PCB V1.1 . . . 41
50 Anpassungen Kommunikation mit LED Dimmer . . . 42
51 Anpassungen Strommessung . . . 43
52 Anpassung Anschlussklemme . . . 43
53 Anpassung IR rx / tx . . . 44
54 Anpassung User Interface . . . 44
55 Vorderseite PCB V2.1 . . . 45
56 Rückseite PCB V2.1 . . . 45
57 Vorderseite bestücktes PCB V2.1 . . . 47
58 Rückseite bestücktes PCB V2.1 . . . 47
59 Gehäuse V1 am Ständer der Stehleuchte montiert . . . 48
60 Gehäuse V2 ohne Deckel . . . 49
61 Gehäuse V2 mit Deckel . . . 49
62 Gehäuse V3 . . . 50
63 Gehäuse V3 ohne Deckel . . . 50
64 Gehäuse V3 am Ständer der Stehleuchte montiert . . . 51
65 FSM Software . . . 52
66 Aufbau Software . . . 55
67 MainTask . . . 56
68 State TurnLampOFF . . . 57
69 Schnittstellen des Moduls States.c . . . 57
70 Funktion Toggle_App_State() . . . 58
71 HueInterfaceTask . . . 58
72 Schnittstellen des Moduls HUEInterface.c . . . 59
73 Code um das Power On / Off Signal zu versenden . . . 59
74 Code IRreceiveTask . . . 60
75 Schnittstellen des Moduls IR.c . . . 60
76 SPITask . . . 61
77 Schnittstellen des Moduls SPI.c . . . 61
78 Code um die RGB LED zu steuern . . . 62
79 Schnittstellen des Moduls LED.c . . . 62
80 Code Debounce Callbackfunktion . . . 62
81 Code Funktion Get_Button_State() . . . 63
82 Schnittstellen des Moduls mybutton.c . . . 63
83 INA239 Read Frame [24, S. 18] . . . 66
84 INA239 Write Frame [24, S. 18] . . . 66
85 Testaufbau um das Pico zu debuggen . . . 68
86 Debugging der finalen Hardware . . . 68
87 Blink Code für das Pico [25] . . . 69
88 Test Lichtsensor LDR . . . 70

89 Zwischenstecker zur Strommessung der Leuchte 71
90 Energiemessgerät PM30 Mandolyn zur Strommessung der Leuchte 71
91 Speisung ZigBee LED Dimmer mit 24 V 73
92 LED Dimmaktor mit angeschlossenem Taster 74
93 LED Dimmaktor mit angeschlossenem Taster 74
94 Betrieb LED Dimmer mit Leuchtennetzteil 75
95 Blockschaltbild Versuchsaufbau 76
96 Ein- und Ausgänge des LED Dimmers 76
97 Potenzial des LED Dimmer Anschlusses LED - 77
98 Schaltung zur Generierung eines 0 - 3.3 Volt PWM Signals 77
99 Durch Tiefpassfilter erweiterte Messchaltung 78
100 geöffneter LED Dimmer mit angelöteter Litze 78
101 Anpassung der Hardware, um die Intensität des IR Signals zu erhöhen 80
102 Anpassung um Ansatz 3 zu testen 81
103 Aufgezeichnetes IR Signal mit Logicanalyzer 82
104 Code zum Spiegeln von IR Frames 83
105 Frontansicht H-A-IR-Remote 84
106 Unteransicht H-A-IR-Remote 84
107 Draufsicht H-A-IR-Remote 84

Tabellenverzeichnis

1 Kabelverbindung Pico - JLink 67
2 Messergebnisse LDR 70
3 Messergebnisse 24V Seite 72
4 Messergebnisse 230V Seite 72
5 Messergebnisse INA239 79
6 Messergebnisse mit Multimeter 79
7 Messergebnisse Dutycycle des LED Dimmer Ausgangs über den Spannungsteiler 82
8 Apple IR Remote Frames 83
9 Strommessung Hardware 83

Literatur

[1] Gottardo. Bedianungsanleitung LED-Stehleuchte Gottardo Serie H. PDF. Online erhältlich unter https://www.gottardo-led.swiss/wp-content/uploads/2022/06/d109716-d-bedienungsanleitung-de-gottardo-serie-h.pdf, zuletzt abgerufen: Dezember 2022.

[2] Phillips Hue. Vergleich von Bluetooth und Hue Bridge. Website. Online erhältlich unter https://www.philips-hue.com/de-ch/explore-hue/how-it-works#bluetooth_or_bridge, zuletzt abgerufen: Dezember 2022.

[3] Hue Developer. Zigbee 3.0 support in Hue ecosystem. Website. Online erhältlich unter https://developers.meethue.com/zigbee-3-0-support-in-hue-ecosystem/, zuletzt abgerufen: Dezember 2022.

[4] Wikipedia. Fernbedienung: Infrarot-Fernbedienung. Website. Online erhältlich unter https://de.wikipedia.org/wiki/Fernbedienung#Infrarot-Fernbedienung, zuletzt abgerufen: Dezember 2022.

[5] Mikrocontroller.com. IR-Fernbedienungsprotokolle. Website. Online erhältlich unter https://mc.mikrocontroller.com/de/IR-Protokolle.php, zuletzt abgerufen: Dezember 2022.

[6] Altium Techdocs. NEC Infrared Transmission Protocol. Website. Online erhältlich unter https://techdocs.altium.com/display/FPGA/NEC+Infrared+Transmission+Protocol, zuletzt abgerufen: Dezember 2022.

[7] Wikipedia. Apple Remote. Website. Online erhältlich unter https://en.wikipedia.org/wiki/Apple_Remote, zuletzt abgerufen: Dezember 2022.

[8] Wikipedia. Serial Peripheral Interface. Website. Online erhältlich unter https://de.wikipedia.org/wiki/Serial_Peripheral_Interface, zuletzt abgerufen: Dezember 2022.

[9] IConnectHue. Philips Hue: Supported lights and devices. Website. Online erhältlich unter https://iconnecthue.com/supported-devices/, zuletzt abgerufen: Dezember 2022.

[10] Jakob Lyke. How to make your own custom Hue lights - for a fraction of the cost! Video. Online erhältlich unter https://www.youtube.com/watch?v=otqme0Ed8-I&t=336s, zuletzt abgerufen: Dezember 2022.

[11] fred27. Control ANY Light With Philips Hue. Website. Online erhältlich unter https://www.instructables.com/Control-ANY-Light-With-Philips-Hue/, zuletzt abgerufen: Dezember 2022.

[12] Peter. Custom firmware Hue lights. Website. Online erhältlich unter https://peeveeone.com/2016/11/custom-firmware-hue-lights/, zuletzt abgerufen: Dezember 2022.

[13] PhilipsHue. Hue Wandschaltermodul. Website. Online erhältlich unter https://www.philips-hue.com/de-ch/p/hue-philips-hue-wand-schaltermodul/8719514318045#overview, zuletzt abgerufen: Dezember 2022.

[14] PhilipsHue. Hue Smart Plug. Website. Online erhältlich unter https://www.philips-hue.com/de-ch/p/hue-smart-plug/8719514342385, zuletzt abgerufen: Dezember 2022.

[15] Zedfy. ZIGBEE UNTERPUTZMODULE IM ÜBERBLICK - SCHALT- & DIMMAKTOREN. Website. Online erhältlich unter https://www.zedfy.com/led-blog/zigbee-unterputzmodule#a-4, zuletzt abgerufen: Dezember 2022.

[16] Adafruit. Adafruit TSL2591 High Dynamic Range Digital Light Sensor. Website. Online erhältlich unter https://learn.adafruit.com/adafruit-tsl2591, zuletzt abgerufen: Dezember 2022.

[17] OpenMV. OpenMV Cam H7 R2. Website. Online erhältlich unter https://openmv.io/, zuletzt abgerufen: Dezember 2022.

[18] Raspberry Pi. Raspberry Pi Pico Datasheet. Datasheet. Online erhältlich unter https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf, zuletzt abgerufen: Dezember 2022.

[19] Adafruit. Adafruit INA260 High or Low Side Voltage, Current, Power Sensor. Website. Online erhältlich unter https://www.adafruit.com/product/4226, zuletzt abgerufen: Dezember 2022.

[20] Kingbright. APDA3020F3C-P22. Datasheet. Online erhältlich unter https://www.mouser.ch/datasheet/2/216/APDA3020F3C_P22-3000911.pdf, zuletzt abgerufen: Dezember 2022.

[21] raspberrypi. measure_duty_cycle. Website. Online erhältlich unter https://github.com/raspberrypi/pico-examples/blob/master/pwm/measure_duty_cycle/measure_duty_cycle.c, zuletzt abgerufen: Dezember 2022.

[22] raspberrypi. Sending and receiving IR (infra-red) codes using the PIO. Website. Online erhältlich unter https://github.com/raspberrypi/pico-examples/tree/master/pio/ir_nec, zuletzt abgerufen: Dezember 2022.

[23] Jack Ganssle. A Guide to Debouncing - Part 2. Website. Online erhältlich unter http://www.ganssle.com/debouncing-pt2.htm, zuletzt abgerufen: Dezember 2022.

[24] Texas Instruments. INA239 85-V, 16-Bit, High-Precision Power Monitor With SPI Interface. Datasheet. Online erhältlich unter https://www.ti.com/lit/ds/symlink/ina239.pdf, zuletzt abgerufen: Dezember 2022.

[25] Erich Styger. Getting Started: Raspberry Pi Pico RP2040 with Eclipse and J-Link. Website. Online erhältlich unter https://mcuoneclipse.com/2022/07/16/getting-started-raspberry-pi-pico-rp2040-with-eclipse-and-j-link/, zuletzt abgerufen: Dezember 2022.

[26] Wikipedia. ZigBee. Website. Online erhältlich unter https://de.wikipedia.org/wiki/ZigBee, zuletzt abgerufen: Dezember 2022.

Abkürzungsverzeichnis

H-A-IR-Remote	Hue Apple IR Remote
PCB	Printed Circuit Board
IR	Infrarot
SPI	Serial Peripheral Interface
PWM	Pulsweitenmodulation
LED	Light Emitting Diode
FSM	Finite State Machine
HW	Hardware
LDR	Light Dependent Resistor
DC	Direct Current
AC	Alternating Current
ARM	Advanced RISC Machine
SWD	Serial Wire Debug
SMD	Surface Mount Device
THT	Throgh Hole Technology
ADC	Analog to Digital Converter
LSB	Least Significant Bit

A Anhang

A.1 Struktur Elektronischer Anhang

Der elektronische Anhang befindet sich auf dem dieser Arbeit beigelegten Datenträger er ist folgendermassen strukturiert:

A.1.1 Aufgabenstellung

A.1.2 Datenblätter

A.1.3 KiCAD Projekte

A.1.4 Software Projekt

A.1.5 CAD Dateien Gehäuse

A.2 Aufgabenstellung

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Technik & Architektur

Horw, 19. September 2022
Seite 1/2

Bachelor Thesis im Studiengang
Elektrotechnik und Informationstechnologie

Aufgabe für Herrn Joshua Wyss

Hue Apple IR Remote

Fachliche Schwerpunkte

Nachrichtentechnik / Signal Processing

Technische Informatik (Embedded Systems)

Einleitung

Viele Leuchten in Büros haben eine Infrarot-(IR)-Fernbedienung, z.B. von Apple. Um diese Leuchten in ein Philips Hue Automatisierungssystem einzubinden, sollen die IR Signale abgefangen, verarbeitet und in einen Hue System bereitgestellt werden. Als Beispiel dazu dient eine 'gottardo' LED Stehleuchte.

Aufgabenstellung

Es soll eine Schaltung mit einem Mikrocontroller und der dazugehörigen Software entwickelt werden, welche als Gateway zwischen Büroleuchten und Infrarotfernbedienung agiert. Damit soll dem Philips Hue System der Status der Leuchte (Helligkeit, Farbe, etc.) mitgeteilt werden. Gleichzeitig soll das System auch als Fernbedienung direkt mit der Leuchte kommunizieren können. Damit soll es möglich sein, eine 'nicht-smarte' Stehleuchte in das Philips Hue System einzubinden. Dabei ist der Lösungsansatz offen. Das Philips Hue System verwendet einen ZigBee to LAN Gateway, es wäre auch eine Möglichkeit, die Einbindung indirekt über einen anderen ZigBee Gateway (z.B. Sonoff) oder mit 'Home Assistant' (https://www.home-assistant.io/) (HA, Raspberry Pi) zu realisieren. Das Augenmerk soll auf einfache Installation (möglichst ohne Veränderung des Leuchtmittels), Benutzbarkeit und niedrigen Energieverbrauch gelegt werden. Das Konzept sollte auf ähnliche Stehleuchten oder Beleuchtungsmittel anwendbar sein. Zu Beginn der Arbeit soll die Arbeitspakete zusammen mit einer Zeitplanung definiert werden.

Termine

Start der Arbeit:	Montag 19.9.2022
Zwischenpräsentation:	Zu vereinbaren im Zeitraum 17.10. – 19.11.2022
Abgabe Schlussbericht:	Montag 2. Januar 2023, vor 16:00 im D311
Abgabe Digitale Doku:	Gemäss separater Anweisung der Studiengangleitung
Abschlusspräsentation:	Zu vereinbaren im Zeitraum 2.1. – 21.1.2023
Diplomausstellung:	Freitag 7. Juli 2023 (Teilnahme obligatorisch!)

FH Zentralschweiz

Hochschule Luzern
Technik & Architektur

Horw, 19.9.2022
Seite 2/2
Diplomarbeit im Fachbereich
Elektrotechnik und Informationstechnologie

Dokumentation

Der gebundene Schlussbericht enthält keine Selbständigkeitserklärung und ist in einfacher Ausführung zu erstellen. Er enthält zudem zwingend

- einen sehr kurzen, englischen Abstract.
- Ein Titelblatt, gleich hinter dem Deckblatt, gemäss Weisungen der Studiengangleitung
- Eine SD-Hülle, innen, auf der Rückseite des Berichtes für den Betreuer

Alle Exemplare des Schlussberichtes müssen komplett und termingerecht gemäss Angaben der Studiengangleitung abgeben werden. Zusätzlich muss eine SD-Speicherkarte mit dem Bericht (inkl. Anhänge), dem Poster und den Präsentationen, Messdaten, Programmen, Auswertungen, usw. unmittelbar nach der Präsentation abgeben werden.
Die gesamte Dokumentation ist zudem gemäss Anweisungen der Studiengangleitung elektronisch auf einen Server zu laden. Sämtliche abzugebende Teile der Dokumentation sind Bestandteile der Beurteilung.

Fachliteratur/Web-Links/Hilfsmittel

https://www.apple.com/shop/product/MM4T2AM/A/apple-remote

Hardware/Software nach Bedarf

Stehleuchte mit Apple IR Remote

Philips Hue: https://www.philips-hue.com

Philips Hue ZigBee-LAN Gateway

Home Assistant: https://www.home-assistant.io/

Gottardo Steh-Leuchten: https://www.gottardo-led.swiss/

Geheimhaltungsstufe: Öffentlich

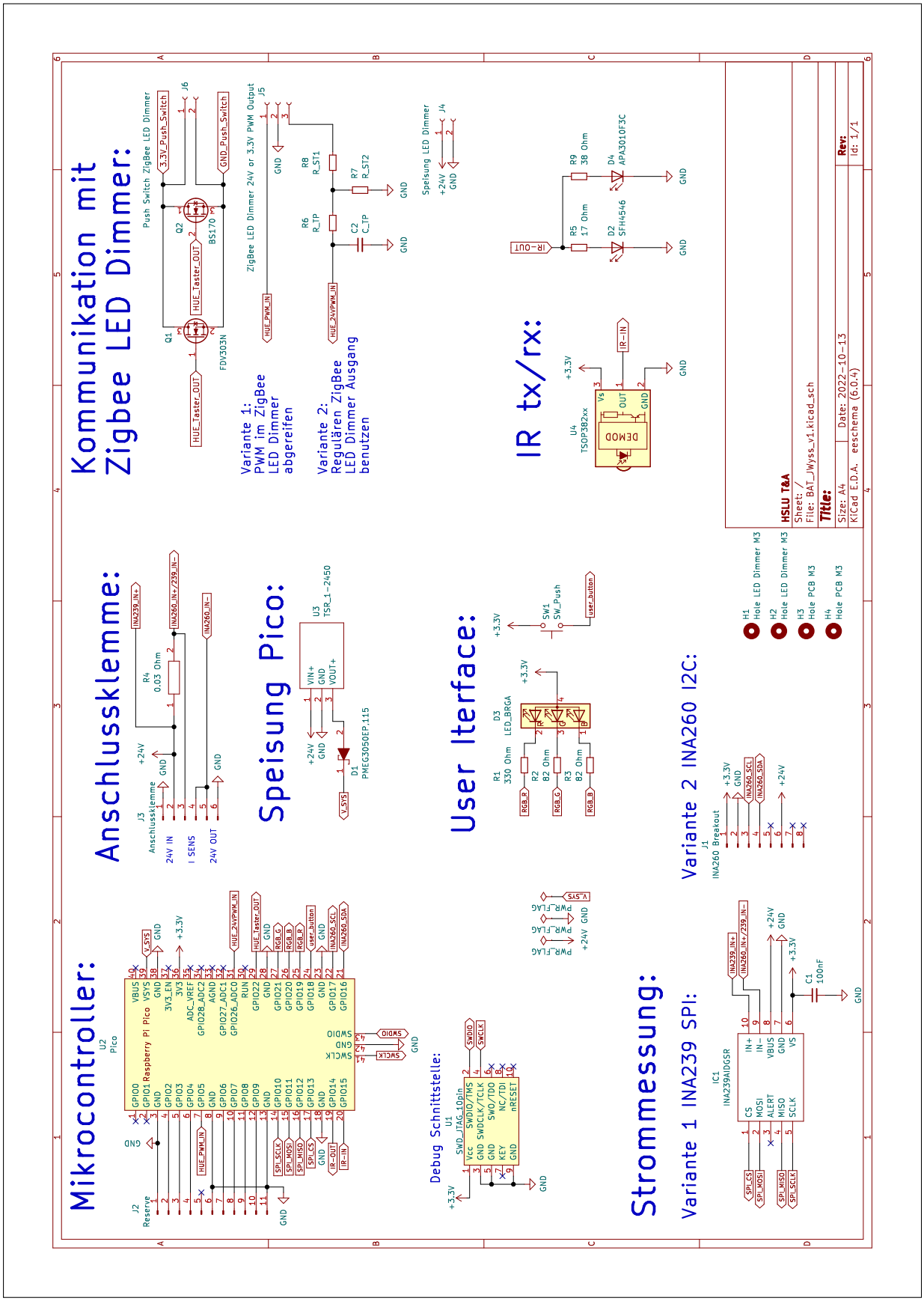
Verantwortlicher Dozent/Betreuungsteam, Industriepartner

Dozent	Prof. Erich Styger	erich.styger@hslu.ch
Industriepartner	IET S-TEC	
Experte	Dr. Christian Vetterli christian.vetterli@mt.com	

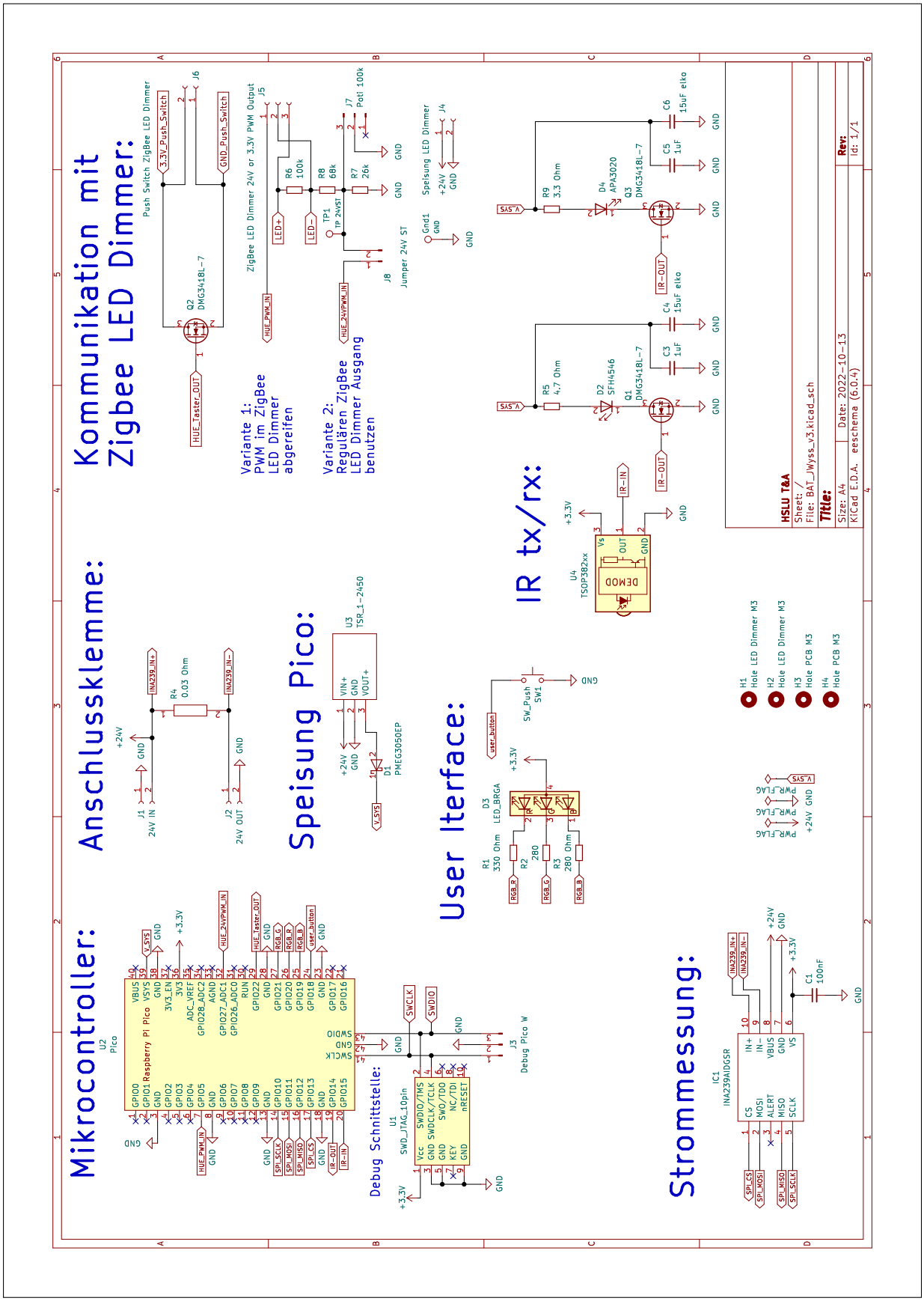
Hochschule Luzern
Technik & Architektur

Prof. Erich Styger

A.3 Schema PCB V1.1



A.4 Schema PCB V2.1



A.5 Kostenauflistung Bauteile

Bauteil	Distributor	Anzahl	1 Stk. [CHF]	Summe
IR Receiver - TSOP38238	Mouser	1	0.92	0.92
Shunt-Widerstand - WFCP0612R0300FE66	Mouser	1	0.634	0.634
Stromsensor - INA239AIDGST	Mouser	1	4.36	4.36
DCDC Wandler - TSR 1-2450	Mouser	1	5.74	5.74
Schottkey Diode - PMEG3050EP,115	Mouser	1	0.56	0.56
Elektrolytkondensator - EEU-FC1H150	Mouser	2	0.39	0.78
Kabelverschraubung M12 - 1417652	Mouser	2	0.61	1.22
Gegenmutter M12 - 7211 979	Mouser	2	0.32	0.64
IR Emitter - APDA3020F3C-P22	Mouser	1	0.64	0.64
Taster - TL3336AF160Q	Mouser	1	0.53	0.53
SMD N-Mosfet - DMG3418L-7	Mouser	3	0.45	1.35
0805 SMD Widerstand	Mouser	9	0.1	0.9
0805 SMD Keramikkondensator	Mouser	3	0.46	1.38
Abgleichwiderstand - 652-3296W-1-104LF	Mouser	1	2.45	2.45
Pinheader, 2 Pin, 2.54 mm Pitch	Mouser	1	0.7	0.7
Jumper, 2 Pin, 2.54 mm Pitch	Mouser	1	0.32	0.32
IR Emitter - SFH 4545	Mouser	1	0.74	0.74
Stecker - NC3MXX	Distrelec	1	3.56	3.56
Buchse - NC3FXX	Distrelec	1	4.52	4.52
Printklemme, 2Pole, 5.08mm Pitch - 1729128	Distrelec	2	1.02	2.04
RGB LED - R6GHBHW-C01	Distrelec	1	1.91	1.91
Wire to Board Reihenklemme - 250-1402	Distrelec	2	0.82	1.64
Wire to Board Reihenklemme - 250-1403	Distrelec	1	1.06	1.06
Pinheader 2 Rows, 10 Pins, 1.27mm Pitch	Distrelec	1	0.83	0.83
Klettverschluss Weiss, 0.3 m - 475-00460	Distrelec	2	0.4	0.8
Td Kabel 2x0.75 mm2 1m	-	3	0.95	2.85
Kleinmaterial (Aderendhülsen, Schrauben...)	-	1	3	3
Gehäuse- PLA White 130 gr	-	1	4.3	4.3
ZigBee LED Dimmer SR-ZG9101CS	Brack	1	44	44
Raspberry Pi Pico RP2040 Board	DigiKey	1	3.8	3.8
PCB	PCBWay	1	4.15	4.15
			Total [CHF]:	102.324

A.6 Strombelastbarkeit von Leiterbahnen

Strombelastbarkeit von Leiterbahnbreiten

Auch die nun folgende Tabelle enthält nur ungefähre Richtwerte. In die Strombelastbarkeit von Leiterbahnen geht eine große Anzahl von Faktoren ein. Die exakte Berechnung für den jeweiligen Fall ist sehr aufwendig. Die Angaben erfolgen ohne Gewähr! Die unten aufgeführten Werte gelten für "normale" mitteleuropäische Umweltbedingungen. Bei höheren Temperaturen gelten reduzierte zulässige Stromstärken. Denken Sie an die Wärmeabfuhr von der Leiterplatte! Nachfolgende Werte gelten für ca. 30 Grad Celsius. Beachten Sie die Strombelastbarkeit von "Durchsteigern", schalten Sie eventuell mehrere parallel! Planen Sie Fertigungstoleranzen und Umgebungseinflüsse als Sicherheiten ein! (Die Fertigungstoleranzen galvanischer Metallabscheidungs-Prozesse sind erheblich!)

Leiterbahnbreite (mm)	Leiterbahnbreite (mil)	Strombelastbarkeit bei 17,5 µm (A)	Strombelastbarkeit bei 35 µm (A)	Strombelastbarkeit bei 70 µm (A)	Strombelastbarkeit bei 105 µm (A)
0,1	4	0,25	0,5	0,75	1
0,2	8	0,5	1	1,5	2,2
0,5	20	1,2	2	3,2	5
1	40	2	4	6,2	9
1,5	60	3	5	8,1	11
2	80	3,3	6	10	12
4	160	6	10	16	22
6	240	8,3	14	22	26
10	400	12	21	nicht sinnvoll	nicht sinnvoll

Quelle: http://www.fs-leiterplatten.de/ebemess.htm

A.7 ZigBee

Üm zu verstehen was ZigBee genau ist und wie es funktioniert wurde eine Recherche durchgeführt. Die Informationen des Anhangs A.7 stammen aus der Quelle [26]. Bei ZigBee handelt es sich um eine Spezifikation für drahtlose Netzwerke mit geringem Datenaufkommen und geringem Stromverbrauch. ZigBee wird zum Beispiel in der Hausautomation, Sensornetzwerken oder der Lichttechnik verwendet. Die Geräte in einem ZigBee Netzwerk kommunizieren über ein Mesh-Netzwerk, was bedeutet, dass die einzelnen Geräte im Netzwerk untereinander vernetzt sind. Wenn also ein Kommunikationspfad im ZigBee Netzwerk ausfällt, kann ein anderer Pfad im Netzwerk verwendet werden. ZigBee Netzwerke werden vor allem in kurzreichweiten Netzwerken (100 – 300m) eingesetzt.

ZigBee Devices
Die ZigBee Spezifikation baut auf dem Standart IEEE 802.15.4 auf und erweitert dessen Funktionalität. Zigbee stellt dem Entwickler drei verschiedene Gerätearten zur Verfügung. Mit diesen Geäten wird dann ein ZigBee Wireless Personal Area Network aufgebaut.

ZigBee End Device, ZED
ZigBee Endgeräte benötigen nur einen Kleinen Teil der Funktionen der ZigBee Spezifikation, da sie nicht am Routing im Netzwerk teilnehmen und somit in einen Schlafmodus gehen können, wenn sie gerade nichts zu tun haben. Somit wird der Stromverbrauch minimiert, was sich sehr gut eignet für Batteriebetriebene Steuerungs- oder Sensormodule. Es ist zu beachten, dass ZigBee Endgeräte aus-schliesslich mit dem Router, über welchen Sie dem Netzwerk beigetreten sind, kommunizieren können.

ZigBee Router, ZR
ZigBee Router nehmen aktiv am Routing der Pakete durch das Netzwerk teil. Somit benötigen sie einen grösseren Funktionsumfang und mehr Hardwareressourcen. Um einem Netzwerk beizutreten, melden sich ZigBee Router an einem im Netzwerk befindlichen Router an. Dieser vergibt dann eine 16Bit Kurzadresse, welche bei Mesh Netzwerken zufällig ausgewählt wird. Das Routing im Netz-werk erfolgt entweder als Baumstruktur oder als Meshnetzwerk.

ZigBee Coordinator, ZC
Ein ZigBee Kordinator startet das Netzwerk mit festgelegten Parametern und übernimmt anschliessend dieselben Aufgaben wie ein Router.

ZigBee Profile
ZigBee Profile dienen dazu für verschiedene Anwendungsfälle Systemvoraussetzungen und Geräte zu definieren. Es gibt dabei veröffentlichte und Herstellerspezifische Profile. Herstellerspezifische Profile sind meist nicht öffentlich und werden vom Hersteller entworfen. Dies ermöglicht den Her-stellern Funktionen zu implementieren, welche nicht in den öffentlichen Profilen vorhanden sind. Der Nachteil von herstellerspezifischen Profilen ist, dass nur die Geräte des Herstellers das Profil sinnvoll nutzen können.

ZigBee Light Link, ZLL
Dieses Profil ermöglicht die Steuerung von Lichttechnik aller Art. Es ist vorgesehen Farbanteile, Helligkeiten und das An- und Ausschalten von Leuchten zu steuern. Der Einfachheit halber

wird auf einen Koordinator verzichtet, wodurch auch ein Trustcenter zur Schlüsselverteilung entfällt. Die Kommunikation wird jedoch immer durch einen Netzwerkschlüssel verschlüsselt. Tritt ein neues Funkmodul dem Netzwerk bei, wird ihm der Netzwerkschlüssel durch den sogenannten Master Key verschlüsselt mitgeteilt. Es wird versucht den Masterkey geheim zu halten, was bedeutet, dass der Privatanwender keine eigenen Produkte für zertifizierte Light-Link Produkte erstellen kann, solan-ge der Masterkey nicht bekannt ist.

ZigBee Home Automation, ZHA
Dieses Profil wird ebenfalls zur Steuerung von Lichttechnik verwendet, jedoch liegt der Fokus auf der allgemeinen Steuerung von Geräten in kleineren Gebäuden. Auch bei diesem Profil ist die Übertragung durch einen Netzwerkschlüssel verschlüsselt. Tritt ein neues Gerät dem Netzwerk bei, wird der Netzwerkschlüssel durch den Masterkey verteilt, wobei der Masterkey in diesem Fall be-kannt ist.

ZigBee Green Power, ZGP
Das ZigBee Green Power Profil wird für den Betrieb von Batterielosen Sensoren verwendet. Dazu wird Micro Energy Harvesting verwendet. Um das ZGP Profil zu verwenden, muss jeder ZigBee Router oder Koordinator die Funktion eines Green Power Basic Proxys (GPBP)implementieren, damit sogenannte Green Power Device Frames versendet werden können.