



Wifi-Gateway zur Steuerung smarter Innenraumleuchten

Bachelorarbeit von

Achraf Ben Rekaya

An der Fakultät für Elektrotechnik und Informationstechnik
Lichttechnisches Institut (LTI)

Erstgutachter: Prof. Dr. rer. nat. Cornelius Neumann
Betreuernder Mitarbeiter: Dr.-Ing. Christian Herbold

02. Mai 2018 – 01. November 2018

Karlsruher Institut für Technologie
Lichttechnisches Institut
Engesserstrasse 13, Geb. 30.34
D-76131 Karlsruhe

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Änderungen entnommen wurde.

Karlsruhe, 01.11.2018

.....

(Achraf Ben Rekaya)

Danksagung

Hiermit möchte ich mich bei meinem Betreuer Dr.-Ing. Christian Herbold für die Unterstützung, Motivation und das interessante Thema der Arbeit, welches meine Begeisterung für die Photon-Welt auch privat geweckt hat, bedanken.

Des Weiteren möchte ich mich herzlich bei Prof. Dr. rer. nat. Cornelius Neumann für die Möglichkeit bedanken, diese Arbeit an seinem Lehrstuhl durchzuführen und natürlich für seine wertvollen fachlichen Beiträge.

Selbstverständlich bin ich allen Freunden zu Dank verpflichtet, die mich bei dieser Aufgabe unterstützt haben.

Abschließend möchte ich mich bei meinen Eltern Kamel Ben Rekaya und Nada Lazrek bedanken, die mir mein Studium durch ihre Unterstützung ermöglicht haben und stets ein offenes Ohr für meine Sorgen hatten.

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Zielsetzung	2
2 Anforderungsanalyse	3
2.1 Aufgaben	3
2.2 Konzeptentwicklung	4
3 Hardwareanalyse	5
3.1 DALI-Protokoll und Steuerung der Smartleuchten	5
3.2 Der WiFi-Gateway	6
3.2.1 Auswahl des Bus-Ankopplers	6
3.2.2 Übersetzen des Dimmwerts in HTTP-Request	9
3.2.3 Kommunikationskanäle	10
4 Softwareanalyse	13
4.1 Entwicklungsumgebung	13
4.2 Particle Docs	14
4.3 Die TCP / SSDP Protokolle	14
4.4 Wireshark	15
5 Implementierung	17
5.1 Schaltungsaufbau	17
5.2 Kommunikation zwischen Bus-Ankoppler und Smartleuchte	19
5.2.1 I2C-Adresse des Bus-Ankopplers herausfinden	19
5.2.2 Ablesen der RAW-Data	19
5.2.3 Suche nach verfügbaren Smartleuchten	20
5.2.4 Funktionen zur Smartleuchtensteuerung	23
6 Tests	31
7 Fazit	33
7.1 Zusammenfassung	33
7.2 Ausblick	33
Literaturverzeichnis	35

1 Einleitung

1.1 Motivation

Mit der aktuellen Entwicklungsrate der Datenverarbeitung und Computerindustrie sind heutzutage Smartgeräte nicht mehr aus dem Leben wegzudenken.

Sie sind fast überall: Derzeit spricht man von intelligentem Zuhause wo die Haushaltsgesräte miteinander verbunden sind und sich ohne Berührung bedienen lassen.

Ein Beispiel dafür ist die Estelle Smartleuchte von Vanory, welche sich an die veränderbaren Lichtverhältnissen anpasst.

Anderes als bei den normalen Lampen, die auf die manuelle Bedienung über einen klassischen Lichtschalter beschränkt bleiben, sind Smartleuchten flexibel per App programmier- und steuerbar. Zur Steuerung der Estelle-Smartleuchte wird ein Android- oder iOS-Gerät benötigt. Die Vanory-App ermöglicht eine direkte Verbindung mit der Smartleuchte. Moodwechsel sowie ein- und ausschalten der Leuchte sind natürlich in der App enthalten.



Abbildung 1.1: Die Estelle Smartleuchte von Vanory [1]

Der Trend geht nun dahin, Smartgeräte autonom zu machen und die Automatisierung in jedes mögliche Feld zu bringen. Durch den Einzug der Mikroprozessortechnik in die Beleuchtungstechnik konnte sich eine neue digitale Kommunikation entwickeln. Die neue

Lichttechnik bringt mehr Möglichkeiten mit Licht sodass die Beleuchtung sich mit wenig Aufwand den eigenen Wünschen anpassen lässt und dabei sogar noch Strom spart. Mit der Existenz von intelligenten Lichtbussystemen können die Beleuchtungskomponenten eines Gebäudes zentral und in einigen Situationen auch kabellos gezielt elektronisch angesteuert werden. Damit ist die Frage nun, wie man diese Smartleuchten noch mehr an unser tägliches Leben anpassen kann.

1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist der Aufbau und die Entwicklung eines Systems, das Dimmwerte aus dem DALI-Bussystem in eine für die Smartleuchte passende HTTP-Anfrage umzuwandeln.

Gleichzeitig sollen auch Funktionen zur Smartleuchtensteuerung, die z.B. die Helligkeitsanpassung und den Moodwechsel erlauben, entwickelt werden.

Das WiFi-Gateway ermöglicht die Verbindung von Smartleuchten mit einem DALI-Bussystem ohne die Leuchtensoftware bearbeiten zu müssen. Das native Vanory-API ist hier bei der Anbindung an das Bussystem zu nutzen.

Ziel ist dem Benutzer die Möglichkeit zu bieten, die auf der Smartleuchte gespeicherten Moods mit DALI durchgehen zu können oder eine beliebige Animation ohne jegliche Interaktion mit der Smartleuchte abzuspielen.

2 Anforderungsanalyse

In diesem Kapitel werden die notwendigen Grundlagen und Anforderungen der Bachelorarbeit diskutiert.

Die Funktionalität des WiFi-Gateways soll mittels einer Konzeptentwicklung erklärt werden. Weiterhin werden die Aufgaben definiert und eine Gliederung festgesetzt.

Es ist nicht wichtig, sämtliche Details der einzelnen Komponenten des Gateways zu verstehen oder zu erörtern; Vielmehr ist ein Überblick der Module und deren Funktionen das Ziel dieses Kapitels.

2.1 Aufgaben

Die Aufgaben sind in 4 verschiedene Bereiche gegliedert:

- Auswahl des Bus-Ankopplers.
- Auswahl des Kommunikationsprotokolls zwischen Bus-Ankoppler und Photon.
- Verbindung Bus-Ankoppler / Photon aufbauen:
 - Ein Ablesen der DALI-Dimmwerte sollte hier möglich sein.
- Übertragung der vom DALI-Bussystem abgelesenen Werten an die Smartleuchte:
 - Die API-Funktionen von Vanory nutzen.

Zu Beginn wird im Abschnitt “Auswahl des Bus-Ankopplers” auf die für das WiFi-Gateway möglichen Ansätze eines Bus-Ankopplers eingegangen. Hier werden die Bauweise, Funktionalitäten sowie genutzten Kommunikationsschnittstellen verschiedener Module aufgelistet und erklärt.

Der Auswahl des Kommunikationsprotokolls wird gleich danach im Abschnitt “Kommunikationskanäle” beschrieben. Es wird zuerst auf die zwei wichtigsten Schnittstellen der Verbindung Bus-Ankoppler-Photon eingegangen. Eine Schlussfolgerung wird über die beste Übereinstimmung am Ende gezogen.

Das wichtigste Kapitel ist der “Implementierung” gewidmet. Hier werden die konkrete Umsetzung des WiFi-Gateways und die Entwicklung der Software erläutert. Es wird abschließend auf die Hauptfunktionen zur Steuerung der Smartleuchte im Abschnitt “Funktionen zur Smartleuchtensteuerung” eingegangen.

2.2 Konzeptentwicklung

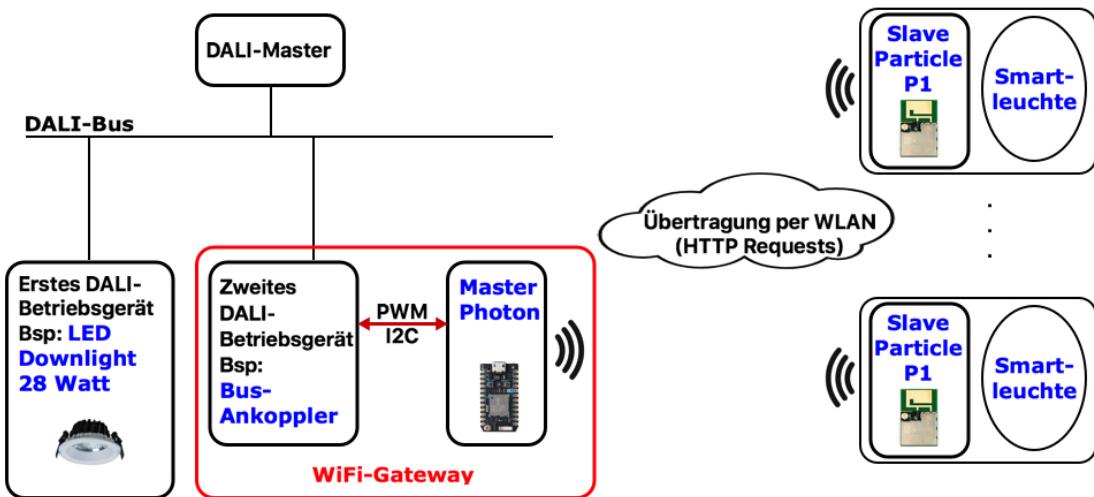


Abbildung 2.1: Konzept des WiFi-Gateways

Das aufzubauende WiFi-Gateway sollte sich einfach mit einem DALI-Bussystem verbinden. Um die vom DALI-Master gesendeten Werte zu lesen, sollte der Bus-Ankoppler bei der Bus-Anbindung als ein DALI-Betriebsgerät erscheinen.

Mittels eines geeigneten Kommunikationsprotokolls werden die Dimmwerte an das Master-Photon weitergegeben. Die Rolle des Photon hierbei ist die drahtlose Steuerung der Smartleuchten indem er die Dimmwerte in HTTP-Anfragen übersetzt.

Eine Suche nach verfügbaren Smartleuchten, welche mit dem gleichen WLAN-Zugangspunkt verbunden sind, sollte vor dem Lesen und Senden der Dimmwerte vom Gateway durchgeführt werden und nicht länger als eine Minute dauern.

Wie in der Abbildung 2.1 zu sehen ist, sollte ein WiFi-Gateway in der Lage sein, die DALI-Werte für mehrere Smartleuchten zu spiegeln.

3 Hardwareanalyse

3.1 DALI-Protokoll und Steuerung der Smartleuchten

Konventionell erfolgt die Lichtbedienung über einen klassischen Lichtschalter. Dies wurde mit der neuen Beleuchtungstechnik durch Lichtbussysteme ersetzt. Die Lichtbussysteme werden hauptsächlich in allgemeinen Beleuchtungsanwendungen wie Büro-, Museums- und Krankenhausbeleuchtung eingesetzt. In einem solchen System werden Sensoren (z.B. Lichtschranken, Infrarotsensoren, oder Helligkeitssensoren ...) und Aktoren (Schaltorgane) in Verbindung gebracht und fungieren "kommunizierend" miteinander.

Das aufzubauende WiFi-Gateway arbeitet mit der DALI-Schnittstelle. Die Abkürzung DALI steht für **Digital Addressable Lighting Interface**. Mittels diesem Bussystem ist eine einfache Steuerung elektronischer lichttechnischer Geräte, Dimmer und Leuchten möglich. Der Bus ist im Anhang E der Vorschaltgeräte-Norm IEC 60929 beschrieben: Dies wurde speziell entwickelt um die Beleuchtungsgeräte eines Gebäudes auf eine standardisierte Weise kommunizieren zu lassen, so dass universelle Controller (auch Master genannt) und Betriebsgreäte (auch Slaves genannt) leicht in Verbindung gebracht werden können. Es werden vor allem mit DALI Elektronische Vorschaltgeräte für Leuchtstoffröhren (EVGs) oder Leuchtdioden (LEDs) gesteuert, gedimmt und überwacht.

Für die Übertragung des DALI-Signals ist eine zweidrahtige Leitung mit einer Busspannung von 16V eingesetzt. Auf der Leitung werden die Daten in Frames übertragen.

Es gibt zwei verschiedene Frametypen: einen "Vorwärts" -Frame (2 Bytes, die vom Master an den Slave gesendet werden) und einen "Rückwärts" -Frame (1 Byte, der auf Anforderung des Masters vom Slave an den Master gesendet wird), so können Slaves z.B. Fehlerzustände zurückmelden. DALI verwendet eine zweiphasige (auch Manchester genannt) Kodierung, was bedeutet, dass die Daten über die Flanken des Signals übertragen werden. Eine steigende Flanke zeigt eine "1" an und eine fallende Flanke zeigt eine "0" an.

Das DALI-Bussystem lässt sich einfach und schnell aufbauen und im Gegensatz zu anderen Bussystemen sind mehrere Topologiearten möglich und können einzelne Leuchten oder Gruppen gezielt über diesen Bus angesteuert werden.

Im Fall des WiFi-Gateways verhält sich der Bus-Ankoppler wie ein DALI-Betriebsgerät. Er wird mit Hilfe von Drähten, die das DALI-Signal leiten, mit dem Bussystem verbunden. Dabei ist seine Rolle die Steuersignale zu empfangen und sie in Dimmwerte umzuwandeln. Die Werte werden danach vom Photon mittels eines geeigneten Kommunikationsprotokolls gelesen und an die Smartleuchten per WLAN gesendet.

3.2 Der WiFi-Gateway

3.2.1 Auswahl des Bus-Ankopplers

Eine wichtige Komponente zur Festlegung der Kommunikation zwischen dem DALI-Bus und dem Particle Photon ist der Bus-Ankoppler.

Eine mögliche Lösung sollte über einen durch den DALI-Bus steuerbaren Eingang und ein für Particle Photon angepasstes Ausgangssignal verfügen.

LED-Warrior12 als Bus-Ankoppler

Ein integrierter Schaltkreis, welcher die oben beschriebene Funktion erfüllt, wurde von der Firma "Code Mercenaries" hergestellt: Der LED-Warrior12 ermöglicht den Einsatz des kompletten DALI-Protokolls für LED-Betriebsgeräte oder für das WiFi-Gateway infrage. Damit lassen sich die Smartleuchten ohne großen Aufwand durch kleine Skripte steuern. Im Grunde genommen, wird das DALI-Signal an die Smartleuchten per WLAN übertragen, indem es zuerst in PWM- / I2C-Signal und anschließend in HTTP-Requests konvertiert wird.

Der LED-Warrior verfügt über 4 PWM-Ausgänge mit jeweils einer Dimmung von 0,1 bis 100% und kann direkt an ein I2C-Bussystem angeschlossen werden.

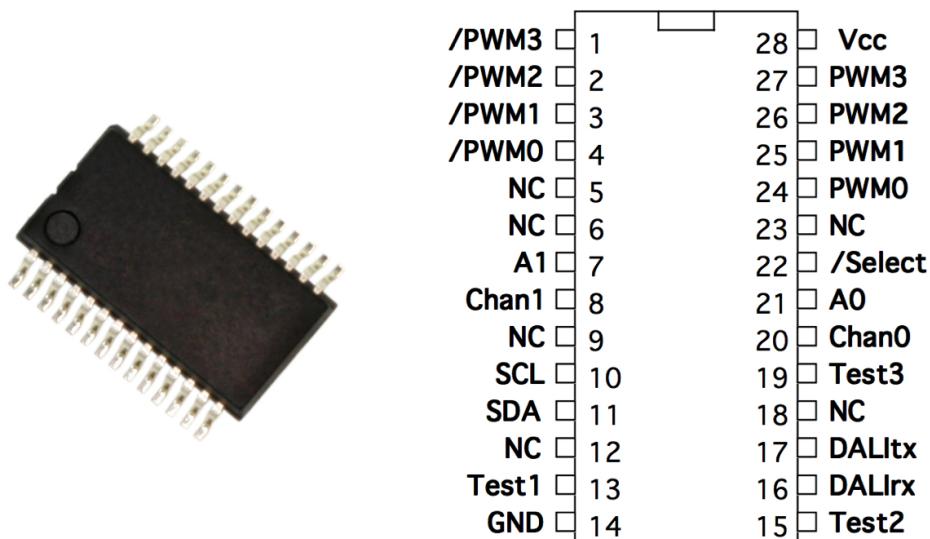


Abbildung 3.1: LW12 als Chip (links) [2] / Pin-Beschreibungen (rechts) [3]

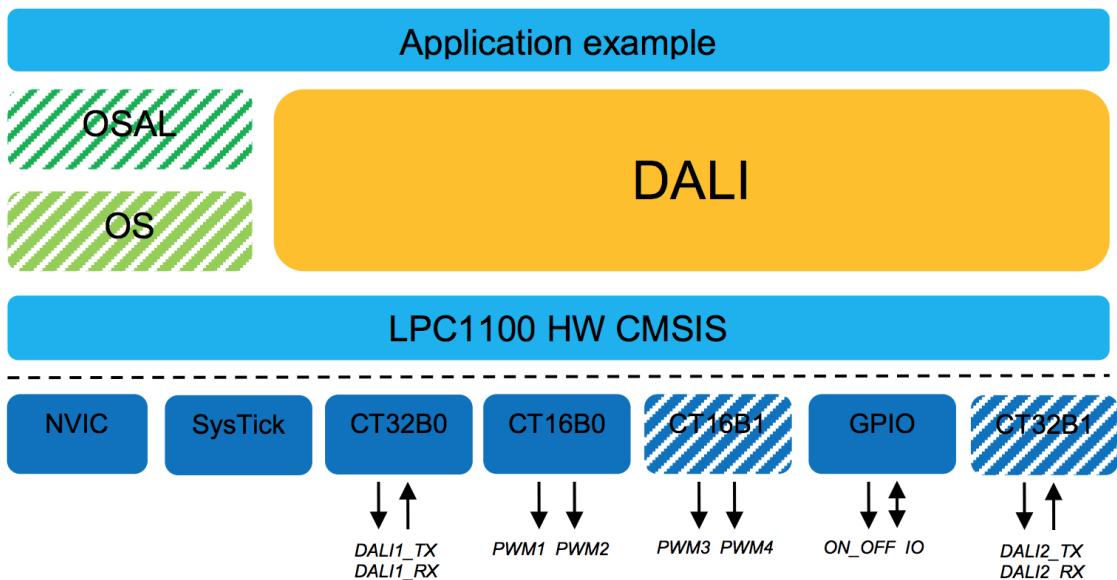
Der LED-Warrior12 ist ein mehrkanaliges DALI-Betriebsgerät, das sich als LED-Leuchte (Typ 6) ausweist. Er erscheint am DALI-Bus als 1 bis 4 Busteilnehmer: Die Anzahl der aktiven Kanäle wird beim Einschalten durch Setzen von zwei Pins (Chan0 und Chan1) ausgewählt.

Das AN11174 DALI-Betriebsgerät

Das von der Firma “NXP Semiconductors” hergestellte Board enthält viele I / O-Funktionen zur Steuerung externer Treiber für Anwendungen in der Solid-State- oder Kompaktleuchstoffbeleuchtung, ist mit der DALI-Schnittstelle und kann als eine mögliche Bus-Ankoppler-Lösung für das WiFi-Gateway angesehen werden.

Die DALI-Protokollbehandlung erfolgt mittels eines Cortex M0 LPC111x-Mikrocontrollers und einer isolierten physikalischen Schicht für den DALI-Bus. Es sind außerdem bis zu vier PWM-Ausgänge und zwei I2C-Pins (1x SCL und 1x SDA) zur gleichzeitigen Steuerung mehrere Endgeräten vorhanden.

Verglichen mit dem LW12, hat der AN11174 einen extra Ausgang: Ein ON/OFF-Signal kann unabhängig von den PWM-Signalen verwendet werden, um die Leuchten in einen EIN- oder AUS-Zustand zu schalten. (siehe Abbildung 3.2)



(1) shaded/striped items are optional

Abbildung 3.2: Einzelne Komponente des AN11174-Betriebsgeräts [4]

Anderes als der LED-Warrior12, kann das AN11174-Betriebsgerät unter Verwendung der mitgelieferten Software geflasht werden. Dies ermöglicht das Steuern grundlegender Vorgänge wie z.B. wie der Mikrocontroller die eingehenden und ausgehenden DALI-Nachrichten behandelt und die Beleuchtung regelt.

Die optisch isolierte DALI-Schnittstelle befindet sich auf der linken Seite des “NXP”-Symbols (siehe Abbildung 3.3). Auf der rechten Seite der Platine befindet sich der LPC1114 Mikrocontroller mit dem 10-Pin SWD Debug / Programmierkopf. Der mittlere Teil der Leiterplatte ist nicht bestückt und für den Einsatz als DALI-Betriebsgerät nicht erforderlich.

3 Hardwareanalyse



Abbildung 3.3: AN11174 Demo-Board [5]

Er dient zum Anschluss einer zweiten, anderen DALI-Physical-Layer oder kann für einen zusätzlichen GPIO- oder 32-Bit-Timer-Ausgang verwendet werden.

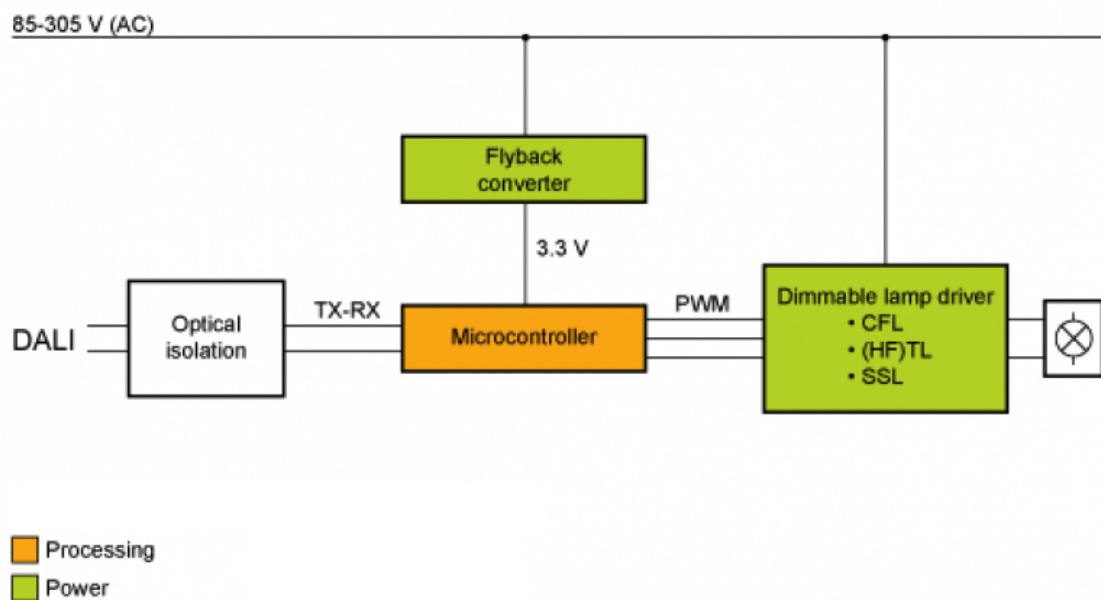


Abbildung 3.4: Anschließen des AN11174-Betriebsgeräts an den DALI-Bus [5]

In der Abbildung 3.4 ist der Schaltplan des AN11174-Betriebsgeräts dargestellt. Der Sperrwandler (Flyback converter) erzeugt aus einer regulären Wechselspannung zwischen 85V und 305V eine 3,3V-Gleichspannung für den Low-Power-Mikrocontroller. In diesem Beispiel wird eine normale Lampe (welche nicht direkt von DALI unterstützt wird) verwendet. Dabei ist die Verwendung eines dimmbaren Lichttreibers zu beachten. Die optische Isolierung ist in der Platine enthalten, was beim LED-Warrior12 nicht der Fall ist.

3.2.2 Übersetzen des Dimmwerts in HTTP-Request

Zur drahtlosen Übertragung des Dimmwerts an die Smartleuchten wurde ein Photon von Particle verwendet. Es handelt sich hier um eine Plattform, die aus einer Platine mit Mikrokontroller, Spannungsversorgungsanschluss und WLAN-Chip besteht. Der Mikrokontroller lässt sich mit der Particle-Software programmieren. Hierbei wird die "C++" Programmiersprache verwendet, die annähernd das gesamte Funktionsspektrum des Mikrocontrollers repräsentiert. Der Photon bietet zahlreiche Erweiterungsmöglichkeiten durch Verwendung von speziellen Sensoren und LED-Leuchten und kann über geeignete Kommunikationsprotokolle zum Ablesen und Senden der DALI-Dimmwerte verwendet werden.

Bei der Photon-Platine (Abbildung 3.5) handelt es sich um ein Entwicklungsboard mit dem Einkernprozessor STM32F205-ARM-Cortex-M3, welcher mit 120 MHz getaktet ist und zusammen mit einem BCM43362-Broadcom WLAN-Chip arbeitet. Das Board verfügt über 24 Pins, 18 davon sind GPIO-Pins mit bis zu 9 PWM-fähigen Pins. Weitere Schnittstellen sind einzelne Serial Peripheral Interface (SPI)-, Inter-IC Sound (I2S)-, Controller Area Network (CAN)- und Inter-Integrated Circuit (I2C)-Schnittstellen, sowie eine Antennenbuchse für eine bessere drahtlose Kommunikation und eine Micro-USB-Schnittstelle für Stromversorgung und Serial-Debugging.

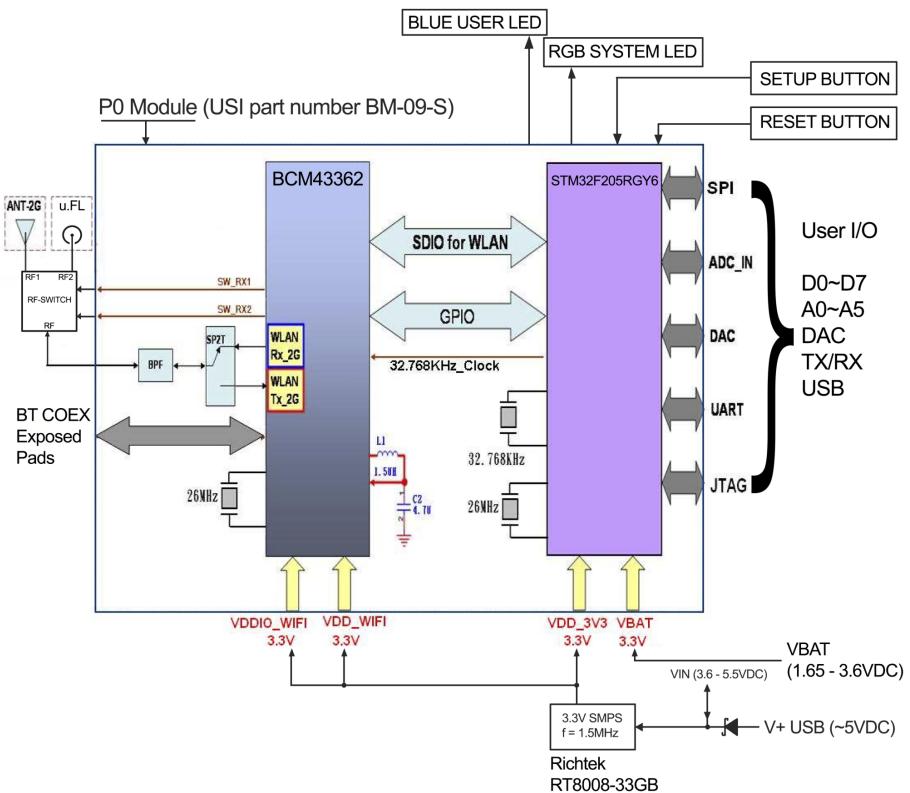


Abbildung 3.5: Blockdiagramm des Photon [6]

Vergleich Photon / Arduino Uno

Der Photon ist in vielerlei Hinsicht wie der Arduino Uno. Beide Geräte haben eine ähnliche Funktionalität mit kleinen Hardware-Unterschieden. Ein Vergleich macht hier Sinn, da ein Konzept schon festgelegt wurde: Das Gateway sollte in der Lage sein, die Dimmwerte per WLAN zu senden. Es gibt viele Entwicklungsboards, die von Particle hergestellt wurden wie z.B. das Electron oder Core. Der Photon kommt mit einem integrierten WLAN-Chip und kostet 19\$. Verglichen mit dem Arduino Uno (Rev3) hat der Particle Photon 1 MB Flashspeicher für Programme, sowie 128KB Ram. Im Gegensatz dazu, hat der Arduino Uno nur 32KB Flash und 2 KB Ram. Der Particle Photon kann, ähnlich wie der Arduino Nano, direkt in das Breadboard gesteckt werden.

Mit einem Startpreis von 20€ hat der Arduino Uno keinen WLAN-Chip. Eine Nachrüstung erhöht wesentlich die Kosten und somit beträgt die Gesamtsumme 79€.

Die Funktionsweise des Photon wird im 4. Kapitel "Softwareanalyse" weiter diskutiert. Es wird auf die Software-Entwicklungsumgebung und "Particle Docs" eingegangen. Im 5. Kapitel "Implementierung" wird darauf eingegangen, wie man das "Photon" programmiert um die HTTP-Requests an die Smartleuchten zu senden.

3.2.3 Kommunikationskanäle

Die I2C Schnittstelle

I2C steht für "Inter-Integrated Circuit" und wurde Anfang der 1980er Jahre von Philips Semiconductors (heute NXP Semiconductors) entwickelt. Mit dem I2C-Bussystem können gleichzeitig mehrere Geräte gesteuert werden. Sowohl Master als auch Slaves können mittels diesem Zweidrahtbus in Verbindung gebracht werden indem jedes Betriebsgerät mit Hilfe einer vordefinierten Adresse angesprochen wird. Betrachtet wird zunächst der Fall eines direkt mit dem Bus-Ankoppler verbundenen Photon, das danach das I2C-Signal in HTTP-Requests konvertiert.

Im Photon sind die Pins D0 (SDA) und D1 (SCL) für die I2C-Kommunikation gedacht. Sie unterstützen bis zu 5.0 V und sind über Pull-Up-Widerstände mit der Versorgungsspannung verbunden.

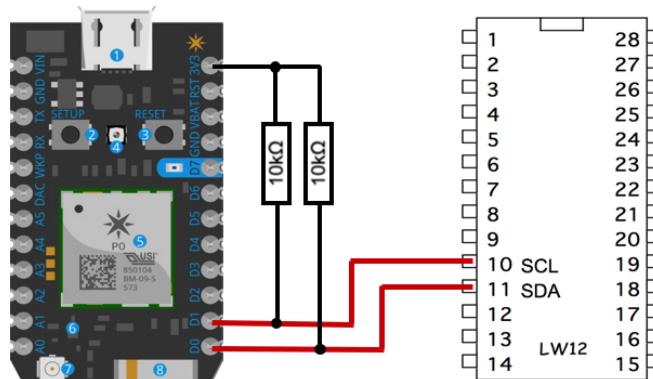


Abbildung 3.6: I2C-Schnittstelle und Pullup-Widerstände

Bei einem I2C-Bus wird die erste Leitung mit SDA (Serial Data Line) bezeichnet. Über diese Leitungen werden die eigentlichen Daten seriell übertragen. Die zweite Leitung wird mit SCL (Serial Clock Line) bezeichnet. Hier werden die Takt-Impulse gesendet. Um ein versehentliches Schalten von digitalen Schaltungen zu verhindern, sollten alle nicht verbundenen Eingänge, die als "potentialfreie Eingänge" bezeichnet werden, mit einer logischen "1" oder "0" verbunden sein. Das ist die Rolle der Pullup-Widerstände. Mittels der beiden Pullup-Widerstände befinden sich beide Leitungen im Ruhezustand auf HIGH-Pegel. Sie können nur durch die angeschlossenen IC's auf LOW-Pegel (Masse) gezogen werden.

Mit der "Wire"-Bibliothek von Particle wird eine Kommunikation zwischen den Photon und externen Betriebsgeräte ermöglicht. Nur die Adresse des Slaves wird dabei benötigt um eine Übertragung zu beginnen.

Helligkeitssteuerung via PWM

PWM (Kurz für Pulsweitenmodulation) beschreibt eine elektrische Größe die zwischen 0 und 1-Signal (Impulspause und -breite) bei konstanter Frequenz wechselt. Mit PWM lassen sich bequem Servomotoren und RGB-LEDs steuern.

Die Dauer der Einschaltzeit des Rechtecksignals (Duty Cycle) bestimmt die effektive Spannung, die für die Stärke der Beleuchtung der LEDs verantwortlich ist. Indem ein digitales Signal HIGH und LOW mit hoher Geschwindigkeit und mit einem bestimmten Tastverhältnis zyklisch durchlaufen wird, verhält sich das PWM-Ausgangssignal wie ein analoges Konstantspannungssignal, wenn die Geräte mit Strom versorgt werden.

Im Particle Photon sind die Pins: D0, D1, D2, D3, A4, A5, WKP, RX und TX für PWM für die Nutzung bereit. Die Softwareunterstützung wird von Particle Docs bereitgestellt um z.B. nach Befehlen zu suchen, die für PWM-Read und -Write implementiert sind. Für den Particle Photon wird die Funktion `analogWrite()` verwendet, um eine stetige Rechteckwelle des angegebenen Tastverhältnisses bis zum nächsten Aufruf der Funktion zu erzeugen. Für das WiFi-Gateway wird jedoch eine Read-Funktion benötigt welche die Einschaltzeit des Signals abruft. Diese Funktion muss manuell programmiert werden, da es keine bekannte Funktion gibt, die ein ordentliches Lesen der Einschaltzeit unterstützt. `AnalogRead()` liefert einen gemappten Wert von der tatsächlichen Spannung, die an einen Pin angelegt wird. Dabei sind nur positive Spannungen erlaubt, die die 3,3V-Schwelle nicht überschreiten.

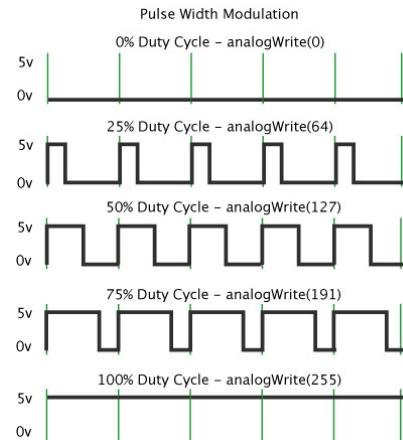


Abbildung 3.7: PWM und `analogWrite()` [7]

4 Softwareanalyse

4.1 Entwicklungsumgebung

Zur Entwicklung der Software für das WiFi-Gateway ist ein “Cross-Compiler” nötig, welcher aus dem Quellcode einen, auf dem Photon lauffähigen, Maschinencode generiert. Für Programme die in C++ geschrieben wurden, stehen für alle Plattformen (MacOS, Windows und Linux) neben der “Web IDE”-Lösung, die neben einem Compiler gleich eine Entwicklungsumgebung und eine Konsole zum Online-Debugging mitbringt, auch “Particle Dev” und “CLI” zur Verfügung.

Die Particle Web-IDE ist eine komplette Online-Plattform mit einem Quellcode-Editor und Compiler die direkt von einem Webbrower unter <https://build.particle.io/build/> abgerufen werden kann. Mittels dieser Plattform können Codezeilen von überall bearbeitet und schlussendlich mit einem Klick geflashed werden. Eine Voraussetzung dafür ist dass der Photon eine funktionierende Internetverbindung hat.

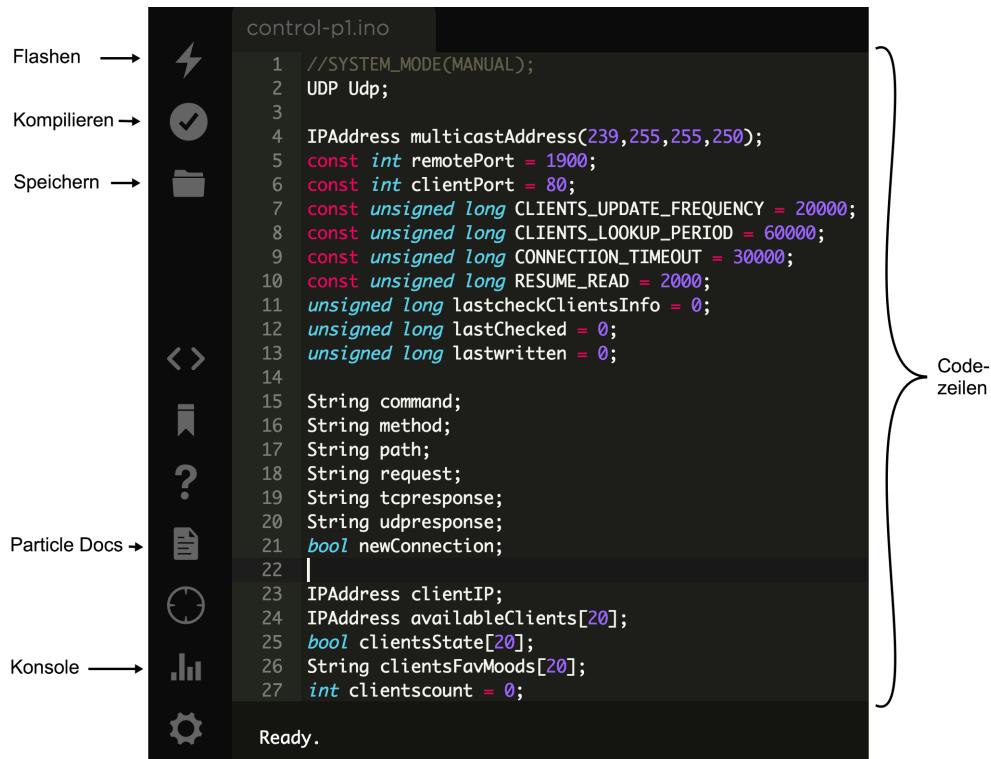


Abbildung 4.1: Die Particle.io Web-IDE-Schnittstelle

Abgesehen von der offiziellen Software und Web-IDE von Particle gibt es Lösungen von Drittanbietern. Po-util (Particle Offline Utility) ist ein Tool, das von der Particle-Community erstellt wurde um die lokale Particle-Entwicklung auf macOS- und Linux-Distributionen zu erleichtern. Es installiert alle notwendigen Tools für die Offline-Partikelentwicklung und bietet eine optimierte Benutzeroberfläche für die Erstellung von Particle-Firmware. Mit Po-util ist eine Internetverbindung nicht erforderlich da das Kompilieren lokal und nicht über die Cloud durchgeführt wird.

4.2 Particle Docs

Particle Docs ist die offizielle Dokumentation für Particle-Geräte. Sie enthält “Erste Schritte”-Anleitungen, Datenblätter, Lernprogramme und die verfügbaren Klassen und Funktionen für die Entwicklung der Software für jedes Gerät.

Hier ist ein Punkt erwähnenswert: Particle hat eine Vielzahl von Open-Source-Repositories auf GitHub. Das Unternehmen ermutigt seine Benutzer, zur Weiterentwicklung der Particle-Software beizutragen. Wann immer ein Benutzer möchte, dass eine neue Funktion implementiert oder ein Problem behoben wird, kann er eine Pull-Anforderung erstellen oder ein Problem direkt im gewünschten Github-Repository melden. Particle verfügt über eine hervorragende Dokumentation, um mit der Verwendung des Photon beginnen zu können. Der Prozess sollte für einen durchschnittlichen Benutzer mit wenig bis gar kein Wissen in der Photon / Arduino-Umgebung reibungslos verlaufen. Es gibt eine App für iOS und Android, mit der man den Photon mit dem Benutzer-Particle-Konto verbindet.

4.3 Die TCP / SSDP Protokolle

Das Transmission Control Protocol (kurz TCP) ist ein zuverlässiges, verbindungsorientiertes, Bytestrom Protokoll. Es nutzt die Dienste des Internet Protocols (IP) zur Herstellung einer unmittelbaren Verbindung zwischen zwei Endpunkten. Dieses Protokoll wird verwendet um die Daten sicher zwischen Server und Client zu übertragen. Es besteht aus zwei wichtigen Teilen: Headers und Body (Nutzdaten).

Die Headers geben den Typ der Anfrage, die Host-URL und verschiedene Verbindungsparameter an. Die Nutzdaten sind der Hauptteil des Requests und enthalten die zu übertragenden Daten.

Im Gateway wird das TCP in Korrelation mit einem zusätzlichen Protokoll verwendet um nach Smartleuchten zu suchen und an sie die Dimmwerte zu senden.

Das Simple Service Discovery Protocol (kurz SSDP) ist ein minimales Netzwerkprotokoll welches basierend auf das User Datagram Protocol (kurz UDP) besonders nützlich bei ungesicherten Netzwerkanwendungen ist und genauso wie TCP, die Dienste des IP-Protokolls nutzt.

Dies wird speziell verwendet um die Dienste eines Gerätes an die anderen Geräten, die mit dem gleichen WLAN-Zugangspunkt verbunden sind, anzukündigen.

Das WiFi-Gateway verwendet im ersten Schritt das SSDP-Protokoll, um eine Suche nach

verfügbar Smartleuchten durchzuführen. Es werden die IP-Adressen der gefundenen Smartleuchten gespeichert und sie danach für die Übertragung der Dimmwerte an jede Smartleuchte mithilfe des TCP-Protokolls verwendet.

4.4 Wireshark

Wireshark wird für die Netzwerkprotokollanalyse und das Erfassen von NOTIFY-Paketen verwendet. Es handelt sich hier um eine PC / Mac- Anwendung die es den Benutzern ermöglicht, eine Fehlerbehebung und Analyse der Protokolle und Netwerkanwendungen im Echtzeit durchzuführen.

Die NOTIFY-Pakete werden von den Smartleuchten als Broadcast-Nachricht gesendet. Das Gateway erfasst die Informationen, prüft ob die Pakete von einer Smartleuchte gesendet wurden und speichert die entsprechenden IP-Adressen ab.

5 Implementierung

5.1 Schaltungsaufbau

Für den Schaltungsaufbau (siehe Abbildung 5.1) wird der LED-Warrior12 als Bus-Ankoppler benutzt und der I2C-Bus für die Kommunikation verwendet. Die Gründe dafür sind:

- Simplizität: Der LED-Warrior12 hat die wichtigsten Funktionen, die zum Ablesen und Konvertieren vom DALI-Signal benötigt werden und verglichen mit dem AN11174-Betriebsgerät ist direkt ohne Programmieren einsatzbereit.
- Effizienz: Die I2C-Schnittstelle ist zuverlässiger für die Datenübertragung und effizienter bei der Implementierung und Verwendung.

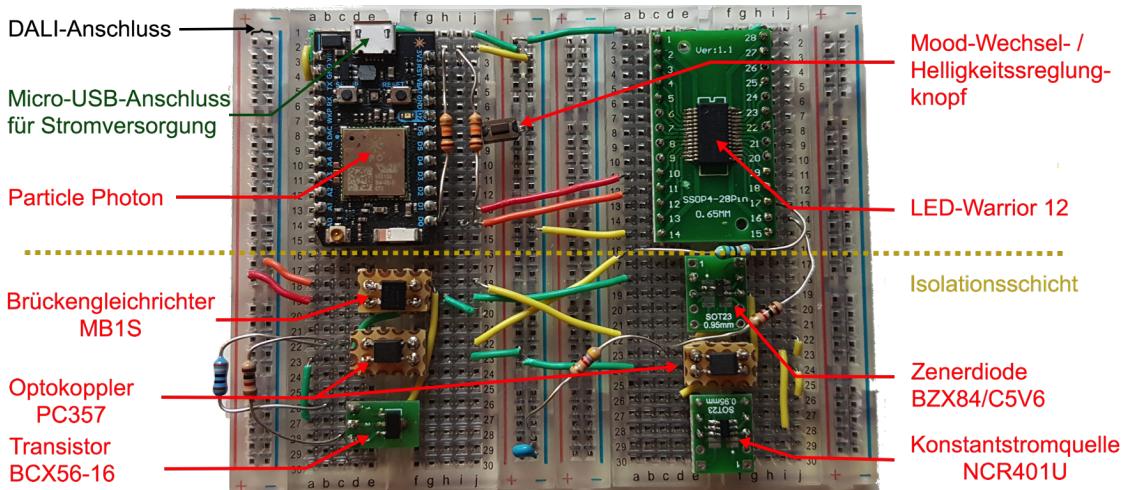


Abbildung 5.1: Aufbau des WiFi-Gateways

Isolationsschicht

Für den Bus-Ankoppler ist eine Isolationsschicht benötigt, um den Mikrocontroller oder IC vor Überspannungen zu schützen und das DALI-Signal vom Bus-Ankoppler optisch zu isolieren. Im Fall des AN11174 DALI-Betriebsgeräts, ist die Isolationsschicht bereits in der Platine enthalten. Der LED-Warrior12 kommt dagegen ohne Isolationsschicht. Aus diesem Grund muss eine Schutzschaltung mit den folgenden Komponenten aufgebaut werden:

Tabelle 5.1: Die zum Aufbau der Schaltung verwendeten Komponenten

Bauteil	Funktion	Menge
MB1S	Brückengleichrichter	1
BCX56-16	Transistor	1
BZX84/C5V6	Zenerdiode	1
NCR401U	Konstantstromquelle	1
PC357	Optokoppler	2
R: 390Ω	Widerstand	1
R: 560Ω	Widerstand	1
R: 1kΩ	Widerstand	2
R: 4,7kΩ	Widerstand	1
C1	Kondensator	1

Die Abbildung 5.2 zeigt das Schaltungsdesign der kompletten Schaltung:

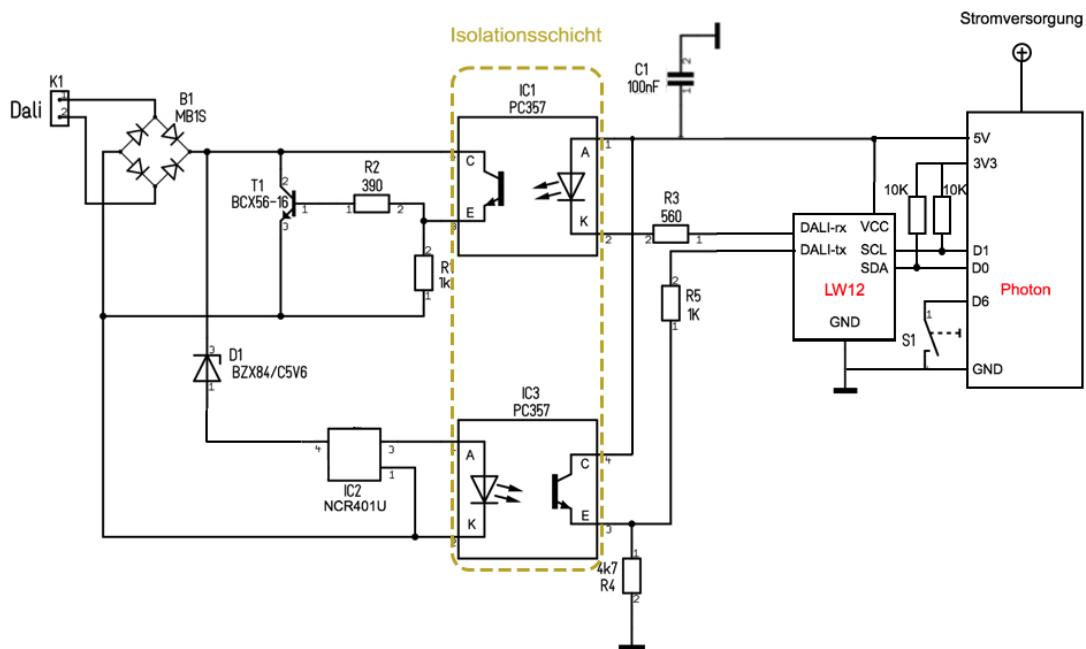


Abbildung 5.2: Schaltungsdesign

5.2 Kommunikation zwischen Bus-Ankoppler und Smartleuchte

5.2.1 I2C-Adresse des Bus-Ankopplers herausfinden

Um die Dimmwerte zu lesen, ist eine Kommunikation mit dem LED-Warrior erforderlich. Eine ordnungsgemäße Scan-Funktion, die nach der I2C-Adresse des Bus-Ankopplers sucht, ist wie folgt definiert:

Listing 5.1: searchforLW-Funktion zur Bestimmung der Bus-Ankoppler-Adresse

```

1 int searchforLW(){
2     byte address = 0;
3     for(address = 1; address < 127; address++) {
4         Wire.beginTransmission(address);
5         if(!Wire.endTransmission(false)){
6             LEDWarrior = address;
7         }
8     }
9     if (LEDWarrior) {
10        Serial.print("Found LW @ 0x");
11        Serial.println(LEDWarrior, HEX);
12    }else{
13        Serial.println("Could not find LW!");
14    }
15    return LEDWarrior;
16 }
```

Mit `Wire.beginTransmission(address)` und `Wire.endTransmission(false)` teilen wir dem Photon mit, ein Start- gefolgt von einem Restart-Bit an den LEDWarrior zu senden. Das Ergebnis des Befehls `endTransmission` ist genau "0", wenn eine erfolgreiche Übertragung stattgefunden hat, was bedeutet, dass ein I2C-kompatibles Gerät an dieser exakten Adresse gefunden wurde.

5.2.2 Ablesen der RAW-Data

Nach Ermitteln der richtigen Adresse des LEDWarriors, ist ein Zugriff auf die Dimmwerter möglich. Ein Blick auf die Konfiguration des Registers (Tabelle 5.2) reicht aus um zu wissen, wo die Rohwerte von DALI zu finden sind.

`Wire.write(0x04)` wird hier verwendet, um auf das Raw-Data-Register zuzugreifen. Gleich danach kommt ein weiterer Befehl (`Wire.requestFrom(_LEDWarrior, 4)`), mit dem 4 Bytes Daten vom LEDWarrior abgerufen werden können, wobei jedes Byte einen Kanal repräsentiert. Nur ein Byte wird hier benötigt um den tatsächlichen Dimmwert zu erhalten. Ein typischer Rohwert liegt zwischen 0 und 254, wobei 0 dem Minimum und 254 dem Maximum entspricht (255 sollte laut dem Hersteller des LED-Warrior12 nicht angezeigt werden). Dieser Wert wird einer Zahl zugeordnet, die den Index der entsprechenden Ani-

mation aus der Favorite-Moods-Tabelle darstellt.

Der abgerufene Wert wird in der `requestedRawData`-Variable gespeichert und bei jedem Lesevorgang mit `currentRawData` verglichen.

`currentRawData` enthält den zuletzt verwendeten Dimmwert direkt nach einer erfolgreichen Schreiboperation. Es wird also erst nach einer vollständigen Übertragung des Befehls über WLAN aktualisiert. Der Vergleich beider Werte ist hier sehr wichtig, damit ein Befehl nur einmal gesendet wird, wenn der Benutzer den Dimmwert ändern möchte. Dafür gibt es zwei Hauptgründe: Einer davon ist es, eine TCP-Verbindung mehrmals pro Sekunde zu vermeiden.

Tabelle 5.2: Registerkonfiguration des LEDWarriors [3]

Register	Funktion	Größe
\$00: Statusregister	Gibt die Anzahl der aktiven DALI-Kanäle zurück	1 Byte
\$04: Raw Data	Gibt die aktuellen Dimmwerte der aktiven Kanäle zurück	4 Bytes
	:	
\$FE: Set Addr	Den LEDWarrior auf eine andere I2C-Adresse verschieben	2 Bytes

Listing 5.2: Ablesen der Raw-Data mithilfe der `readRawData`-Funktion

```

1 int readRawData(byte _LEDWarrior){
2     Wire.beginTransmission(_LEDWarrior);
3     Wire.write(0x04);
4     if(!Wire.endTransmission(false)){
5         byte numbytes = Wire.requestFrom(_LEDWarrior, 4);
6         if(numbytes == 4){
7             int requestedRawData = 0;
8             while(Wire.available()){
9                 requestedRawData = Wire.read();
10            }
11            if(requestedRawData != currentRawData) {
12                return requestedRawData;
13            }else{
14                return -1;
15            }
16        }
17    }
18 }
```

5.2.3 Suche nach verfügbaren Smartleuchten

Für die Suche nach verfügbaren Smartleuchten wird das SSDP-Protokoll (Simple Service Discovery Protocol) eingesetzt. Dafür sollte der Photon erstmal eine Verbindung mit der Multicast-Adresse 239.255.255.250:1900 aufbauen. Gleich danach ist ein Ablesen der mit

der Multicast-Adresse geteilten Informationen möglich.
Die dafür benötigten Codezeilen sind:

```

1 IPAddress multicastAddress(239,255,255,250);
2 const int remotePort = 1900;
3 UDP Udp;
4 Udp.begin(remotePort);
5 Udp.joinMulticast(multicastAddress);
```

Zunächst betrachtet wird der Fall einer Kommunikation zwischen einem Master-Photon, welches für die drahtlose Übertragung des Steuerbefehls zuständig ist, und einer Smartleuchte, welche den Befehl empfangen wird. Es wird alle 30 Sekunden ein NOTIFY-Paket (discovery message) von der Smartleuchte an die Multicast-Adresse gesendet. Dieses Paket wird danach vom Photon empfangen und gelesen. Um sicherzustellen, dass es sich in dem Fall um ein von der Smartleuchte gesendetes Paket handelt, prüft der Photon, ob der Server-Typ der Zeichenfolge: Particle/1.0 UPnP/1.0 entspricht.

Bei Übereinstimmung wird ein Eintrag mit der IP-Adresse der Smartleuchte in die availableClients-Tabelle vorgenommen. Verwendet wird hier eine Tabelle, um eine gleichzeitige Verbindung mit mehreren Smartleuchten zu ermöglichen.

Die Dauer des Suchvorgangs ist in der CLIENTS_LOOKUP_PERIOD-Variable (in Millisekunden) vorgegeben:

```
1 const unsigned long CLIENTS_LOOKUP_PERIOD = 60000;
```

Listing 5.3: Suche nach verfügbaren Smartleuchten

```

1 void lookforClients(){
2     if (clientscount) listallClients();
3     Serial.println("Looking for clients ..");
4     RGB.control(true);
5     RGB.color(255, 255, 0);
6     lastChecked = millis();
7     while(millis() - lastChecked < CLIENTS_LOOKUP_PERIOD &&
8           lookingforClients){
9         if (Udp.parsePacket() > 0) {
10            udpreponse = "";
11            while(Udp.available()){
12                udpreponse += String(char(Udp.read()));
13            }
14            if(udpreponse.indexOf("SERVER: Particle/1.0 UPnP/1.0") != -1){
15                clientIP = Udp.remoteIP();
16                if(!clientregistered(clientIP)){
17                    Serial.println("Found: "+clientIP.toString());
18                    availableClients[clientscount] = clientIP;
19                    clientsState[clientscount] = true;
20                    atleastoneclientisup = true;
21                    clientscount++;
```

5 Implementierung

```
21     }
22 }
23 }
24 }
25 Serial.println("End of scan.\r\n");
26 RGB.control(false);
27 }
```

Die Funktion `clientregistered()` (Listing 5.4) prüft, ob dem `availableClients`-Array bereits die Client-IP-Adresse hinzugefügt wurde. Der Output der Funktion ist somit ein boolescher Wert.

Das `availableClients`-Array ist folgender Maßen definiert:

```
1 IPAddress availableClients[20];
```

Wobei 20 die maximale Anzahl an Client-Smartleuchten entspricht, die sich beim WiFi-Gateway anmelden können. Der Wert kann auch deutlich höher gewählt werden und wird außerdem für das `clientsState`-Array verwendet. Wenn die Client-IP-Adresse nicht im Array gefunden wird, wird sie hinzugefügt und die Anzahl der Clients (`clientscount`) wird um 1 erhöht.

Während des Suchvorgangs, weist das Statuslicht des Photons die Farbe gelb auf.

Andere Variablen, die in der `lookforClients()`-Funktion verwendet werden:

```
1 IPAddress clientIP;
2 bool lookingforClients = true;
3 bool clientsState[20];
4 int clientscount = 0;
```

Listing 5.4: `clientregistered()`-Funktion

```
1 bool clientregistered(IPAddress _client){
2     for(int iter=0; iter < clientscount; iter++){
3         if(availableClients[iter] == _client) return true;
4     }
5     return false;
6 }
```

Das NOTIFY-Paket

Das SSDP-Protokoll kann Plug & Play-Geräte mit UPnP (Universal Plug and Play) erkennen. Es verwendet Unicast- und Multicast (239.255.255.250), ist ein HTTP-ähnliches Protokoll und arbeitet mit den NOTIFY- und M-SEARCH-Methoden. Das NOTIFY-Paket wird verwendet, um die Dienste für alle SSDP-kompatiblen Geräte im lokalen Netzwerk anzukündigen. Mit Wireshark kann man ein NOTIFY-Paket erfassen, das alle 30 Sekunden von der Smartleuchte gesendet wird. Ein Beispiel dafür ist in der folgenden Abbildung gegeben:

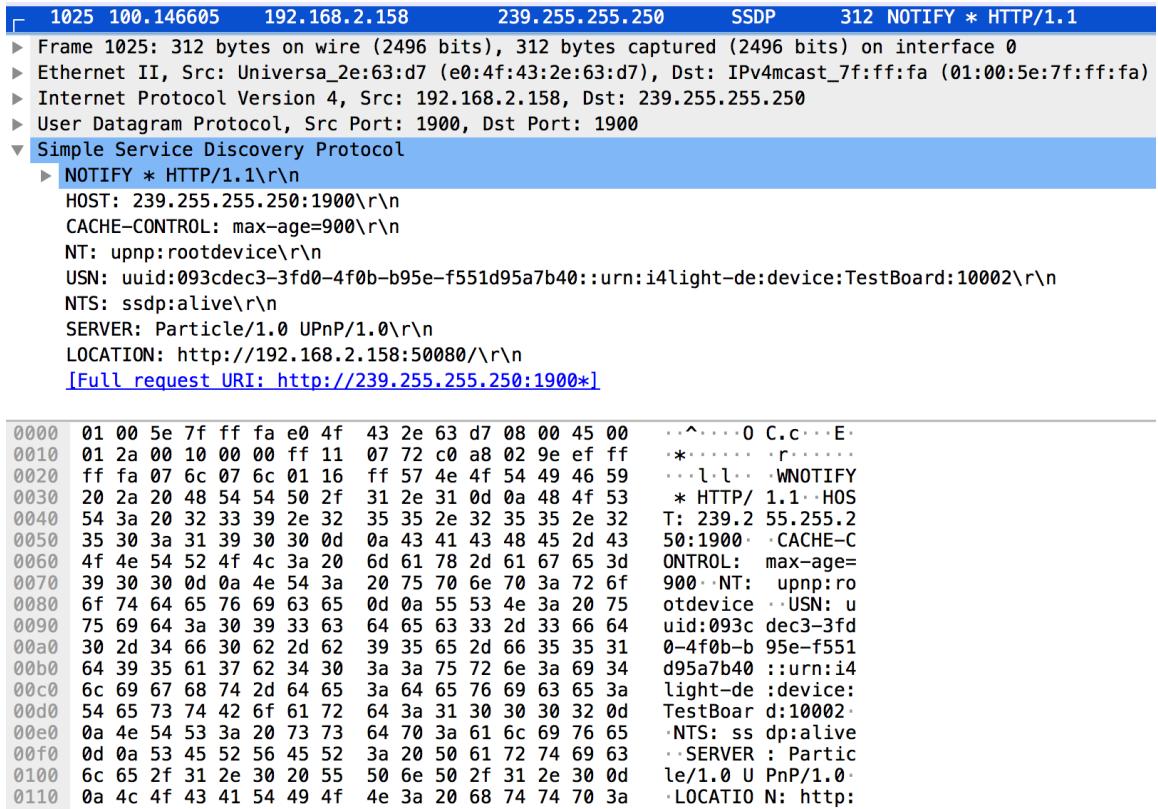


Abbildung 5.3: Beispiel eines NOTIFY-Pakets einer Smartleuchte (von Wireshark erfasst)

5.2.4 Funktionen zur Smartleuchtensteuerung

Die Funktionen zur Smartleuchtensteuerung folgen alle der gleichen Struktur:

- Die Sendemethode aktualisieren mit `updateMethod();`
 - Die URL aktualisieren mit `updatePath();`
 - Das Kommando aktualisieren mit `updateCommand();`
- Ausführen der gewünschten Funktion: Der Befehl wird an die Smartleuchte über eine TCP-Verbindung übertragen. (mit `doRequest();`)

Unter Verwendung der API-Funktionen von Vanory, ist man in der Lage folgende Aufgaben zu erledigen:

Helligkeit einstellen

Listing 5.5: Helligkeit einstellen

```

1 int setBrightness(String _bri){
2     updateMethod("PUT");
3     updatePath("/username/matrix/state");
4     updateCommand("{\"bri\":\"+_bri+"}");
5     return doRequest();
6 }
```

5 Implementierung

Mood einstellen

Listing 5.6: Mood einstellen

```
1 int setMood(String _mood){  
2     updateMethod("PUT");  
3     updatePath("/username/player/");  
4     updateCommand("{\"mood\": \""+_mood+"\"}");  
5     return doRequest();  
6 }
```

Animation steuern

Listing 5.7: Animation steuern

```
1 int setControl(String _ctrl){  
2     updateMethod("PUT");  
3     updatePath("/username/player/");  
4     updateCommand("{\"control\": \""+_ctrl+"\"}");  
5     return doRequest();  
6 }
```

Variablenwerte aktualisieren

Listing 5.8: Variablenwerte aktualisieren

```
1 int updateMethod(String _method) {  
2     method = _method;  
3     Serial.println("Method is set to: "+method);  
4     return 0;  
5 }  
6  
7 int updatePath(String _path) {  
8     path = _path;  
9     Serial.println("Path is set to: "+path);  
10    return 0;  
11 }  
12  
13 int updateCommand(String _command) {  
14     command = _command;  
15     Serial.println("Command is set to: "+command);  
16     return 0;  
17 }
```

Die Update-Funktionen sind wie folgt zu implementieren:

Die Variable erhält den neuen Wert. Eine Bestätigung wird danach über die serielle Schnittstelle an den Benutzer gesendet.

Die doRequest()-Funktion

Die Funktion durchläuft alle verfügbaren Smartleuchten, öffnet eine TCP-Verbindung und überträgt dann die gewünschte Aufgabe (Helligkeit-, Mood- oder Animationsteuerung) an jeweils eine Smartleuchte. Aufgrund der Art und Weise, wie die TCP-Verbindung funktioniert, die normalerweise weniger als eine Sekunde zur Ausführung der Aufgabe benötigt, erscheint es dem Benutzer so, als ob die Smartleuchten gleichzeitig die HTTP-Anfragen erhalten würden.

Der Benutzer erhält letztendlich den Output der `writetoClient()`-Funktion, die den Status der Übertragung anzeigt. Drei Hauptprobleme können dabei auftreten:

- `Connection to client timed out !`: Die Smartleuchte ist überlastet.
- `Empty response from client !`: Die Smartleuchte hat keine Antwort gesendet.
- `Could not connect to client !`: Die Verbindung zur Smartleuchte konnte nicht hergestellt werden → Es kann sein dass die Smartleuchte ausgeschaltet wurde.

Die Verbindung zur Smartleuchte geht dann automatisch verloren und wird erst nach Änderung des Dimmwerts wieder hergestellt.

Listing 5.9: Die doRequest-Funktion

```

1 int doRequest() {
2     for(int iter=0; iter < clientscount; iter++){
3         if (clientsState[iter]) {
4             clientIP = availableClients[iter];
5             buildRequest(clientIP, method, path, command);
6             Serial.println();
7             Serial.println("Output from: "+clientIP.toString());
8             Serial.println(writetoClient(clientIP));
9             Serial.println();
10        }
11    }
12    return 0;
13 }
14
15 void buildRequest(IPAddress _clientIP, String method, String path, String
16 command){
17     request = method+" "+path+" HTTP/1.1\r\nHost: "+_clientIP.toString()+"\r\nConnection: close\r\nContent-Length: "+command.length()+"\r\nContent-Type: application/json\r\n\r\n"+command;
18 }
```

5 Implementierung

```
18
19 String writetoClient(IPAddress _clientIP){
20     tcpresponse = "";
21     TCPClient client;
22     newConnection = client.connect(_clientIP, clientPort);
23     if(newConnection){
24         client.print(request);
25         lastChecked = millis();
26         while(newConnection && millis() - lastChecked < CONNECTION_TIMEOUT
27             ){
28             while (client.available()){
29                 tcpresponse += String(char(client.read()));
30             }
31             if (!client.connected()){
32                 tcpresponse = tcpresponse.substring(tcpresponse.indexOf("\r\n\r\n") + 4);
33                 tcpresponse = tcpresponse.substring(0, tcpresponse.indexOf("\r\n"));
34                 client.stop();
35                 newConnection = false;
36             }
37             if(newConnection) tcpresponse = "Connection to "+_clientIP.
38                 toString()+" timed out !";
39             if(!tcpresponse.length()) tcpresponse = "Empty response from "+
40                 _clientIP.toString()+" !";
41         }
42         return tcpresponse;
43     }
```

Die in der `doRequest()`-Funktion verwendeten Variablen:

```
1 const unsigned long CONNECTION_TIMEOUT = 30000;
2 unsigned long lastChecked = 0;
3 String command;
4 String method;
5 String path;
6 String request;
7 String tcpresponse;
8 bool newConnection;
```

Programmablauf des Master-Photon (WiFi-Gateway)

Die setup()-Funktion

Die Photon setup()-Funktion ist für die Initialisierung vorgesehen und wird einmal beim Start des Programms durchgeführt. Zunächst wird jedem Photon-Pin ein Pin-Modus zugewiesen. Die serielle Schnittstelle wird mit der gewählten Baud-Rate initialisiert und der Time-Out eingestellt. Außerdem werden die I2C-Schnittstelle mit `Wire.begin();` und die UDP-Verbindung mit `Udp.begin();` gestartet.

Listing 5.10: Die setup-Funktion

```

1 void setup() {
2     pinMode(modePin, INPUT_PULLUP);
3     Serial.begin(9600);
4     System.on(button_status, button_handler);
5
6     Udp.begin(remotePort);
7     Udp.joinMulticast(multicastAddress);
8
9     Wire.begin();
10 }
```

Die in der `setup()`-Funktion verwendeten Variablen und Funktionen:

```

1 int modePin = D6;
2
3 void button_handler(system_event_t event, int duration, void* )
4 {
5     if (!duration) {
6         lookingforClients = !lookingforClients;
7     }
8 }
```

Die `button_handler()`-Funktion wird hier verwendet um die Suche nach verfügbaren Smartleuchten zu starten. Während des Suchvorgangs wird durch einen zweiten Druck auf den Setup-Button des Photon der Vorgang beendet.

Die loop()-Funktion

Nach dem Initialisieren mit `setup()` folgt die Hauptschleifen-Funktion, bei Photon "loop()" genannt. Ein Photon-Programm muss zwingend neben der `setup()` eine `loop()`-Funktion enthalten. Diese stellt die Superloop des Mikrocontrollers dar, die während des Betriebs endlos durchlaufen wird, vergleichbar mit einer `while(true)`-Schleife, die bei reinem C für Mikrocontroller verwendet werden muss, da das Verhalten sonst nicht definiert ist. im ersten Abschnitt der Schleifenfunktion wählt der Benutzer den Arbeitsmodus des WiFi-Gateways. Er hat die Wahl, ob er die vom Bus-Ankoppler abgerufenen Werte verwendet, um das Licht zu dimmen oder die Animation zu ändern.

5 Implementierung

Die gespeicherten Smartleuchten-IP-Adressen werden alle 20 Sekunden gepingt, um zu sehen, ob die Smartleuchten immer noch eingeschaltet sind. Das Prüfintervall ist in der `CLIENTS_UPDATE_FREQUENCY`-Variable vorgegeben und kann beliebig groß eingestellt werden. Der wichtigste Abschnitt kommt gleich danach. Der Photon ruft die Rohwerte vom Bus-Ankoppler ab und sendet den vorher gewählten Befehl drahtlos an die Smartleuchten.

Listing 5.11: Die loop()-Funktion

```
1 void loop() {
2     if (!digitalRead(modePin)) {
3         if (mode == "Dimming") {
4             mode = "FavMoods";
5         }else{
6             mode = "Dimming";
7         }
8         currentRawData = -1;
9         Serial.println("Current mode is: "+mode+"\r\n");
10        while(!digitalRead(modePin));
11    }
12    if (lookingforClients) {
13        lookforClients();
14        lookingforClients = false;
15    }
16    if(millis() - lastcheckClientsInfo > CLIENTS_UPDATE_FREQUENCY){
17        checkClientsState();
18        checkClientsFavMoods();
19        lastcheckClientsInfo = millis();
20    }
21    if(clientscount && atleastoneclientisup){
22        if(LEDWarrior) {
23            if (millis() - lastwritten > RESUME_READ) {
24                int tmpRaw = readRawData(LEDWarrior);
25                if(tmpRaw > -1){
26                    delay(500);
27                    if(tmpRaw == readRawData(LEDWarrior)){
28                        if (mode == "Dimming") {
29                            setBrightness(String(tmpRaw));
30                        }else{
31                            for (int iter=0;iter < clientscount;iter++){
32                                String favMoods = clientsFavMoods[iter]+",
33                                ";
34                                String tblFavMoods[26];
35                                tblFavMoods[0] = "";
36                                int numfavmoods = 1;
37                                while(favMoods.indexOf(",") > -1) {
```

```

37         tblFavMoods[numfavmoods] = favMoods.
38             substring(0, favMoods.indexOf(","))
39             ;
40         favMoods = favMoods.substring(favMoods
41             .indexOf(",") + 1);
42         numfavmoods++;
43     }
44     int mappedValue = map(tmpRaw, 0, 254, 0,
45     numfavmoods - 1);
46     Serial.println("mappedValue: " + String(
47         mappedValue));
48     if (numfavmoods > 1 && map(currentRawData,
49         0, 254, 0, numfavmoods - 1) != mappedValue) {
50         if (!mappedValue) {
51             setControl("off");
52         } else{
53             setMood(tblFavMoods[mappedValue]);
54         }
55     }
56     currentRawData = tmpRaw;
57     lastwritten = millis();
58 }
59 }
60 }
61 }
```

Die in der `setup()`-Funktion verwendeten Variablen und Funktionen:

Listing 5.12: `checkClientsState()`-Funktion

```

1 void checkClientsState(){
2     atleastoneclientisup = false;
3     for (int iter=0;iter < clientscount;iter++){
4         clientsState[iter] = WiFi.ping(availableClients[iter]);
5         if (clientsState[iter]) atleastoneclientisup = true;
6     }
7 }
```

5 Implementierung

```
1 const unsigned long CLIENTS_UPDATE_FREQUENCY = 20000;
2 const unsigned long RESUME_READ = 2000;
3 unsigned long lastcheckClientsInfo = 0;
4 unsigned long lastwritten = 0;
5 String clientsFavMoods[20];
6 bool atleastoneclientisup = false;
7 String mode = "Dimming";
```

6 Tests

Das WiFi-Gateway wurde erfolgreich mit einem Demo Board und der Estelle Smartleuchte von Vanory getestet. Die serielle Schnittstelle war eine große Hilfe um einige Probleme debuggen zu können. Außerdem wurde die Particle Konsole mehrmals verwendet, um die Codezeilen in Echtzeit zu debuggen.

Als DALI-Master wurde der Casambi CBU-ASD (siehe Abbildung 6.1) verwendet.

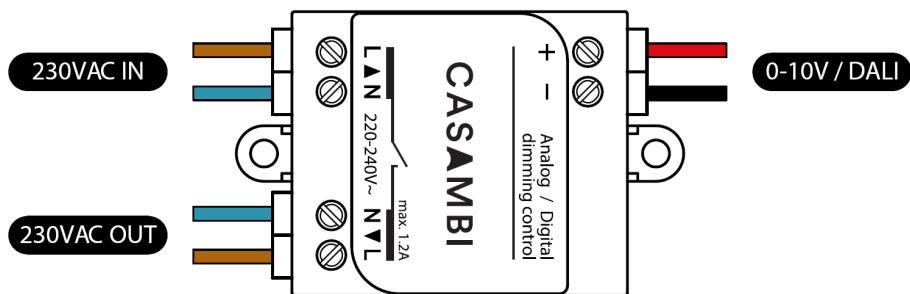


Abbildung 6.1: Casambi CBU-ASD DALI-Master [8]

Dank der Casambi-App (für Android und iOS) ist eine drahtlose Steuerung dieses Moduls möglich. Der Casambi CBU-ASD dient als Steuereinheit für LED- und Halogen-Treiber mit 0-10V, 1-10V oder DALI-Dimmerschnittstelle, unterstützt DALI-Szenen und passt somit perfekt für die Tests.

Getestet wurde war die TCP-Verbindung zwischen Gateway und Smartleuchte: Es konnte in meisten Fällen eine zuverlässige Verbindung aufgebaut werden und die entsprechende Aufgabe an die Smartleuchte gesendet werden. Ein Router wird auch benötigt um die drahtlosen Verbindungen herzustellen. Der DALI-Master ist mit der App per Bluetooth steuerbar. Ein DALI-Signal wird vom DALI-Master nach der Änderung der Beleuchtungsstärke von der App erzeugt. Dies wird zunächst vom LED-Warrior in einem Dimmwert konvertiert und vom Photon durch I2C abgelesen. Der Photon überträgt im nächsten Schritt die Aufgabe an die Smartleuchten / Demo-Board mithilfe des TCP-Protokolls. Die Aufgabe wird schlussendlich erledigt und eine Response für das Gateway mit { Success:True} gesendet.

Mit der seriellen Schnittstelle werden die Fehlermeldungen - falls vorhanden- angezeigt werden.

7 Fazit

7.1 Zusammenfassung

Rückblickend kann gesagt werden, dass aus dieser Arbeit ein funktionsfähiges WiFi-Gateway zur Steuerung der Estelle Smartleuchte von “Vanory” realisiert wurde.

Ein Prototyp aus dem entwickelten WiFi-Gateway und Demo Board weist die grundlegende Funktionalität nach. Das entwickelte WiFi-Gateway ist in der Lage, bis zu 20 Smartleuchten gleichzeitig zu steuern und bietet eine einfache Anbindung an ein DALI-Bussystem an. Die TCP-Verbindung vom Gateway zu den Smartleuchten läuft zuverlässig und schnell. Ein HTTP-Request vom Gateway wird unverzüglich und ohne Verzögerung bearbeitet. Somit zeigen sich Änderungen vom DALI-Bussystem sofort an den Smartleuchten.

7.2 Ausblick

Das entwickelte System bietet sich als sinnvolle Möglichkeit zur Steuerung der Estelle-Smartleuchten von Vanory an und bietet weitere flexible zukünftige Einsatzmöglichkeiten. Die gleiche Aufgabe wird bei einer Änderung des DALI-Dimmwerts an alle Smartleuchten gesendet. Es kann also z.B. ein Schalter verwendet werden, um jede Smartleuchte separat anzusteuern.

Eine Fallback-Funktion für die TCP-Verbindung könnte auch programmiert werden. Damit der Benutzer eine volle Kontrolle über das WiFi-Gateway bei einem Übertragungsfehler hat, somit kann er z.B. den Befehl erneuert senden oder einfach abbrechen.

Eine andere Verbesserungsidee ist der Anschluss des Gateways an andere Beleuchtungssysteme (wie z.B. KNX, Casambi, Zigbee ...).

Literaturverzeichnis

- [1] *Vivid Mood Light.* URL: <https://vanory.com/>. (Verfügbar am: 15.10.2018).
- [2] *1 to 4-channel DALI controller supports PWM and I2C.* URL: <http://www.eenewsled.com/news/1-4-channel-dali-controller-supports-pwm-and-i2c>. (Verfügbar am: 15.10.2018).
- [3] *DALI slave, one to four channels PWM and I2C output.* URL: https://www.codemercs.com/downloads/ledwarrior/LW12_Datasheet.pdf. (Verfügbar am: 15.10.2018).
- [4] *AN11174 DALI slave using the LPC111x.* URL: <https://www.mouser.com/pdfdocs/AN11174.pdf>. (Verfügbar am: 15.10.2018).
- [5] *LPC111x DALI slave demo board.* URL: <https://www.element14.com/community/docs/DOC-67524/l/lpc111x-dali-slave-demo-board>. (Verfügbar am: 15.10.2018).
- [6] *PHOTON DATASHEET (V016).* URL: <https://docs.particle.io/datasheets/wi-fi/photon-datasheet/>. (Verfügbar am: 15.10.2018).
- [7] *PWM.* URL: <https://www.arduino.cc/en/Tutorial/PWM>. (Verfügbar am: 15.10.2018).
- [8] *LIGHTING CONTROL VIA BLUETOOTH AND APP CASAMBI CBU ASD DALI.* URL: <https://www.watt24.com/en/Smart-Home/smart-home-controller/Lighting-control-via-Bluetooth-and-App-Casambi-CBU-ASD-DALI-watt24.html>. (Verfügbar am: 15.10.2018).
- [9] *Zentrale Leittechnik: BUS-Systeme.* URL: <https://www.licht.de/de/trends-wissen/licht-specials/lichtmanagement/bus-systeme/>. (Verfügbar am: 15.10.2018).
- [10] *Das Transmission Control Protocol (TCP) - Das Internet - Teil 7.* URL: <https://www.webschmoeker.de/grundlagen/tcp-transmission-control-protocol/>. (Verfügbar am: 15.10.2018).
- [11] *Das User Datagram Protocol (UDP) - Das Internet - Teil 8.* URL: <https://www.webschmoeker.de/grundlagen/udp-user-datagram-protocol/>. (Verfügbar am: 15.10.2018).
- [12] *Digital Addressable Lighting Interface, Eine Aktivität des Marktausschusses Betriebsgeräte im ZVEI.* URL: http://www.lichtart.de/8b7f2d87556fe9856eab367a697c342b_dali_erklaerungen.pdf. (Verfügbar am: 15.10.2018).

Abbildungsverzeichnis

1.1	Die Estelle Smartleuchte von Vanory [1]	1
2.1	Konzept des WiFi-Gateways	4
3.1	LW12 als Chip (links) [2] / Pin-Beschreibungen (rechts) [3]	6
3.2	Einzelne Komponente des AN11174-Betriebsgeräts [4]	7
3.3	AN11174 Demo-Board [5]	8
3.4	Anschließen des AN11174-Betriebsgeräts an den DALI-Bus [5]	8
3.5	Blockdiagramm des Photon [6]	9
3.6	I2C-Schnittstelle und Pullup-Widerstände	10
3.7	PWM und analogWrite() [7]	11
4.1	Die Particle.io Web-IDE-Schnittstelle	13
5.1	Aufbau des WiFi-Gateways	17
5.2	Schaltungsdesign	18
5.3	Beispiel eines NOTIFY-Pakets einer Smartleuchte (von Wireshark erfasst)	23
6.1	Casambi CBU-ASD DALI-Master [8]	31

Tabellenverzeichnis

5.1	Die zum Aufbau der Schaltung verwendeten Komponenten	18
5.2	Registerkonfiguration des LEDWarriors [3]	20

Listings

5.1	searchforLW-Funktion zur Bestimmung der Bus-Ankoppler-Adresse	19
5.2	Ablesen der Raw-Data mithilfe der readRawData-Funktion	20
5.3	Suche nach verfügbaren Smartleuchten	21
5.4	clientregistered()-Funktion	22
5.5	Helligkeit einstellen	23
5.6	Mood einstellen	24
5.7	Animation steuern	24
5.8	Variablenwerte aktualisieren	24
5.9	Die doRequest-Funktion	25
5.10	Die setup-Funktion	27
5.11	Die loop()-Funktion	28
5.12	checkClientsState()-Funktion	29