



Bachelorthesis LED-Mapper

Dokumentation Bachelorthesis LED-Mapper

Bachelorthesis
Patrik Aebscher, Elia Bösiger
Biel, 14. Juni 2018

Abstract

Ausgangslage

Lichterketten mit LEDs werden immer populärer. Während die Leute jedoch vor zehn Jahren noch damit zufrieden waren, einfach in einer festgelegten Reihenfolge die Farben wechseln zu lassen, wollen sie heute das Motiv gleich selber bestimmen. Dies ist jedoch mit einem erheblichen Aufwand verbunden. Die Konfiguration jeder einzelnen Leuchtquelle in der Lichterkette benötigt technisches Verständnis und ist aufwändig.

Zielsetzung

Das Zuweisen und Konfigurieren der einzelnen Leuchtquellen sollte doch eigentlich mit modernen Mitteln in wenigen Schritten möglich sein:

1. Eine Kamera auf die Lichterszene richten
2. Mobile-App ausführen und per Knopfdruck die Zuweisung der einzelnen Leuchtquellen starten
3. Lichterszene virtuell auf der Mobile-App darstellen
4. Muster zeichnen oder Bilder laden
5. Virtuelle Szenerie auf die Lichterkette laden

Umsetzung

Die einzelnen Teilbereiche des gesamten Systems wie beispielsweise die Mobile-App oder die Funktion für die Zuweisung der einzelnen Leuchtquellen werden in sogenannte Services unterteilt. Sie können auf verschiedenen Geräten, unabhängig von anderen Services laufen und über das MQTT-Protokoll kommunizieren. Das MQTT-Protokoll dient zur Kommunikation und gleichzeitig als Schnittstelle zu einem Agent. Dieser fungiert als Zentrale und verbindet sich mit jeder, von einem Service zur Verfügung gestellten, Schnittstelle. Liegt nun an einer beliebigen Schnittstelle ein Befehl an, zum Beispiel: „Schalte Leuchtquelle Nummer 10 ein“, leitet der Agent die entsprechende Aktion ein. Er meldet auf der Schnittstelle zur Lichterkette: „Schalte Leuchtquelle Nummer 10 ein“. Ist ein Service vorhanden, der diesen Befehl ausführen kann, wird die Leuchtquelle Nummer 10 eingeschaltet.

Dieser modulare Aufbau hat den Vorteil, dass ein einzelner Service durch einen völlig anderen ersetzt werden kann, solange er die entsprechende Schnittstelle bereitstellt. Dadurch kann unser Projekt für die verschiedensten Lichterketten eingesetzt werden. Es muss lediglich ein neuer Service für die entsprechende Hardware geschrieben werden.

Resultat

Bei einer Lichterkette mit zahlreichen Leuchtquellen (ca. 300) dauert es doch einige Minuten bis alle komplett zugewiesen sind. Auch ist es, trotz diversen Konfigurationsmöglichkeiten, schwierig, den verschiedenen Lichtverhältnissen Rechnung zu tragen. Mit dem richtigen Lichtverhältnis kann unser System das Zuweisen der einzelnen Leuchtquellen jedoch schnell und zuverlässig umsetzen. Das Darstellen von verschiedenen Motiven ist über die Mobile-App bequem und einfach. Dies ermöglicht auch einem technischen Laien, Bilder schnell und ohne Frust auf seiner Lichterkette darzustellen.

Inhaltsverzeichnis

1 Einleitung	6
1.1 Zielsetzung	6
1.1.1 Offizielle Aufgabenstellung	6
1.1.2 Detaillierte Aufgabenstellung	6
2 Pflichtenheft.....	7
2.1 Beschrieb.....	7
2.1.1 Produkteinsatz	7
2.1.2 Ablauf	7
2.1.3 Detaillierte Beschreibung.....	7
2.2 Vorgehen	9
2.3 Stakeholder.....	9
2.4 Systemabgrenzung	10
2.4.1 Use-Case-Diagramm.....	10
2.4.2 Was ist nicht Teil des Projektes.....	10
2.5 Anforderungen.....	11
2.5.1 Muss-Kriterien.....	11
2.5.2 Kann-Kriterien	12
2.6 Test-Cases	12
2.6.1 Muss-Kriterien.....	12
2.6.2 Kann-Kriterien	12
3 Systemübersicht	13
3.1 Grundidee	13
3.2 Einzelne Komponenten	13
3.3 Gliederung in Services.....	13
3.4 Serviceübersicht.....	14
3.4.1 User Interface.....	14
3.4.2 Kamera-Service	14
3.4.3 Leuchtquellen-Service	14
3.4.4 Datenverarbeitungs-Service.....	14
3.4.5 Bildumwandlungs-Service	14
3.4.6 Agent	15
3.5 Kommunikation via MQTT	15
4 Technische Systembeschreibung	16
4.1 Hilfsmittel und Werkzeuge	16
4.1.1 Verwendete Technologien	16
4.1.2 Verwendete Unterstützungstools.....	16
4.2 Grundsätzliche Servicearchitektur	16
4.2.1 Haupt-Controller	18
4.2.2 MQTT-Controller	18
4.3 Benutzerschnittstelle (User Interface)	18
4.3.1 Funktion	18
4.3.2 Technologie	18
4.3.3 Aufbau des User Interfaces	18
4.3.4 Befehle des User Interface	25
4.3.5 Rückmeldung an das User-Interface	25

4.4 Kamera-Service	26
4.4.1 Funktion	26
4.4.2 Technologie	26
4.4.3 Aufbau des Kamera-Services.....	26
4.4.4 Befehle an den Kamera-Service	27
4.4.5 Rückmeldung des Kamera-Service	27
4.5 Leuchtquellen-Service	28
4.5.1 Bildschirm-Service	28
4.5.2 Funktionsweise des Led-Strip-Service.....	29
4.6 Datenverarbeitungs-Service	32
4.6.1 Funktionsweise des Datenverarbeitungs-Service	32
4.6.2 View des Datenverarbeitungs-Service	34
4.6.3 Befehle an den Datenverarbeitungs-Service	34
4.6.4 Rückmeldungen des Datenverarbeitungs-Service	34
4.7 Bildumwandlungs-Service.....	35
4.7.1 Funktionsweise des Bildumwandlungs-Service	35
4.7.2 View des Bildumwandlungs-Service.....	36
4.7.3 Befehle an den Bildumwandlungs-Service.....	36
4.7.4 Rückmeldungen des Bildumwandlungs-Service.....	36
4.8 Agent	37
4.8.1 View des Agent.....	37
4.9 Kommunikationsaufbau (MQTT)	38
4.9.1 MQTT.....	38
4.9.2 Topics als Schnittstelle nutzen	38
4.9.3 Allgemeine Topic-Struktur	39
4.9.4 Gesamte Topic-Struktur des LED-Mapper	40
4.9.5 Nutzung der Topics durch den Agent.....	41
4.10 Mapping-Vorgang	42
4.11 Hardwarekonfiguration des Systems.....	43
4.12 Anleitung für die Inbetriebnahme.....	44
4.12.1 Anforderungen	44
4.12.2 Download und Installation	44
5 Umsetzung	46
5.1 Motivation	46
5.2 Vorgehensweise	46
5.3 Projektplan.....	47
5.3.1 Projektplan nach Abgabe der Bachelorthesis	49
5.4 Technologien.....	50
5.4.1 Auswahlmöglichkeiten	50
5.4.2 Entscheidung	51
5.4.3 Kommunikation mit MQTT	52
5.5 User Interface	53
5.5.1 Struktur der Android-Applikation	53
5.5.2 Datenspeicherung und Datenstruktur	53
5.5.3 Darstellen der virtuellen Leuchtquellen.....	53
5.5.4 Muster zeichnen / Farbauswahl.....	53
5.6 Kamera-Service	54
5.6.1 Livestream und Foto schiessen	54
5.7 Bildschirm-Service.....	54
5.8 Agent	54

<i>5.9 Datenverarbeitungs-Service</i>	55
5.9.1 Entdeckte Gefahr beim aktuellen Algorithmus.....	55
<i>5.10 Zusammenführen aller Services</i>	55
<i>5.11 Led-Strip-Service</i>	56
5.11.1 Unterstütze Konfigurationen	56
5.11.2 Latenzzeit der LED-Strips.....	56
<i>5.12 Bildumwandlungs-Service</i>	57
<i>5.13 Testprotokoll</i>	58
6 Fazit	65
6.1 Ausblick.....	66
7 Anhang.....	67
7.1 User Interface: Funktion <i>getCoordinateFactorForView</i>	67
7.2 User Interface: Funktion <i>getLayoutParamsFor</i>	67
7.3 Erklärung zur Bachelorthesis	68

Änderungsübersicht

Datum	Autor	Beschreibung der Änderung	Betroffene Kapitel
06.03.2018	Patrik Aebischer, Elia Bösiger	Fertigstellung des Pflichtenhefts	Pflichtenheft (Kapitel 2)
06.03.2018	Patrik Aebischer, Elia Bösiger	Fertigstellung des Projektplans	Projektplan (Kapitel 5.3)
14.06.2018	Patrik Aebischer, Elia Bösiger	Fertigstellung des gesamten Dokumentes	Alle

Abkürzungsverzeichnis

Abkürzung	Beschreibung
LED	Light Emitting Diode (Deutsch: lichtemittierende Diode)
App	Application Software (Deutsch: Anwendungsssoftware)
IoT	Internet of Things (Geräte, die über das Internet miteinander kommunizieren).
MQTT	Message Queue Telemetry Transport. Publish / Subscribe- Protokoll, das vor allem im IoT Verwendung findet.
JPEG	Bestimmtes Format einer Bilddatei.
TCP	Transmission Control Protocol ist ein Netzwerkprotokoll für den Datenaustausch.
TLS	Transport Layer Security ist ein hybrides Verschlüsselungsprotokoll zur Datenübertragung im Internet.
JAR	JAR ist eine ausführbare Java-Applikation und steht für Java Archive.
APK	Application Package Kit ist das Package-File-Format welches von Android für die Installation und Inbetriebnahme einer Android-Mobile-App benutzt wird.
GUI	Das Graphical User Interface (grafische Benutzeroberfläche) stellt die grafische Schnittstelle zum Benutzer dar.

Begriffsverzeichnis

Begriff	Beschreibung
User Interface (UI)	Über das User Interface (Benutzerschnittstelle) werden den Services Befehle erteilt, sowie wichtige Daten visualisiert.
Beleuchtungsanlage	Eine Beleuchtungsanlage ist die konkrete Anordnung von Leuchtquellen. Zum Beispiel kann das eine Lichterkette sein, die aus 100 Leuchtquellen besteht.
Leuchtquelle	Zum Beispiel ein Pixel, ein LED oder eine ähnliche Leuchtquelle, die unabhängig von anderen angesteuert werden kann.
Mapping-Vorgang	Als Mapping-Vorgang wird das Zuordnen der Koordinaten aller einzelnen Leuchtquellen einer Beleuchtungsanlage bezeichnet.
Virtuelle Beleuchtungsanlage	Die virtuelle Beleuchtungsanlage ist ein Abbild der Beleuchtungsanlage auf dem User Interface. Die virtuelle Beleuchtungsanlage entsteht nach dem Mapping-Vorgang.
Szenerie	Eine Szenerie ist eine spezifische Konfiguration einer virtuellen Beleuchtungsanlage. Zum Beispiel kann das eine Italienflagge sein, die durch die Beleuchtungsanlage angezeigt werden kann.
Agent	Der Agent ist das Herzstück des Systems. Er führt mehrere Services so aus, dass ein funktionstüchtiges Programm entsteht.
Service	Ein Service hat nur einen bestimmten Zweck (zum Beispiel Leuchtquelle ein-/ausschalten) und interessiert sich nicht, was um ihn sonst noch passiert.
Tinkerforge	Einfache Elektronikbauteile für diverse Anwendungszwecke. Zum Beispiel messen von Temperaturen oder steuern von LEDs.
Membervariable	Im objektorientierten Programmieren ist eine Membervariable eine Variable, die einem bestimmten Objekt zugeordnet ist.

Abbildungsverzeichnis

Abbildung 1 Kommunikationsübersicht der Service	8
Abbildung 2 User Interface: User-Case Diagramm	10
Abbildung 3 MVC-Pattern	17
Abbildung 4 Applikations-Struktur	17
Abbildung 5 UI: Model-Klassendiagramm	18
Abbildung 6 UI: Ansicht "Meine Beleuchtungen"	19
Abbildung 7 Dialog nach abgeschlossenem Mapping-Vorgang	20
Abbildung 8 UI: Verbindung zum MQTT-Broker prüfen	20
Abbildung 9 Ansicht mit verfügbaren Services	20
Abbildung 10 Dialog für das Erstellen einer neuen Szenerie	21
Abbildung 11 UI: Szenerien-Übersicht	21
Abbildung 12 UI: Szenerie-Ansicht	22
Abbildung 13 Konfigurationsansicht ohne Verbindung zum MQTT-Broker	23
Abbildung 14 UI: Ansicht Beleuchtungs-Konfiguration mit Verbindung zum MQTT-Broker	23
Abbildung 15- Kamera-Service: Einstellungs-Ansicht	26
Abbildung 16 Kamera-Service: Kamera-Ansicht	27
Abbildung 17 Bildschirm-Service: Start-View	28
Abbildung 18 Bildschirm-Service: View mit einzelnen Leuchtquellen	28
Abbildung 19 Led-Strip-Service: Grafische Benutzerschnittstelle	30
Abbildung 20 Referenzfoto	32
Abbildung 21 Vergleichsfoto	32
Abbildung 22 Referenzbild mit Weisskonturen	33
Abbildung 23 Vergleichsbild mit Weisskonturen	33
Abbildung 24 Datenverarbeitungs-Service: Grafische Benutzerschnittstelle	34
Abbildung 25 Bildumwandlungs-Service: Grafische Benutzeroberfläche	36
Abbildung 26 Agent: Grafische Benutzerschnittstelle	37
Abbildung 27 Kommunikationsübersicht der Services	38
Abbildung 28 Allgemeine Topic-Struktur	39
Abbildung 29 MQTT-Message-Ablauf vom Mapping-Vorgang	42
Abbildung 30 Hardware-Konfiguration der Realisierung	43
Abbildung 31 Weisskontur, wenn eine LED direkt auf die Kamera gerichtet ist	55

Zitate

Koenig, R. (2018). *Bachelorthesis-Aufgabe*. Biel.

Quellenverzeichnis

- [1] Android, «Android Activity,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/app/Activity>.
- [2] Android, «Android SQLiteOpenHelper,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper>.
- [3] Android, «Android FrameLayout,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/widget/FrameLayout>.
- [4] Android, «Android Button,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/widget/Button>.
- [5] Android, «Android View.OnTouchListener,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/view/View.OnTouchListener>.
- [6] Wikipedia, «Wikipedia RGB-Farbraum,» 21 Februar 2018. [Online]. Available: <https://de.wikipedia.org/wiki/RGB-Farbraum>.
- [7] Tinkerforge, «Tinkerforge Brick Daemon,» [Online]. Available: <https://www.tinkerforge.com/de/doc/Software/Brickd.html>.
- [8] OpenCV, «OpenCV-Dokumentation Funktion imgcodecs,» 23 Februar 2018. [Online]. Available: https://docs.opencv.org/3.4.1/d4/da8/group__imgcodecs.html.
- [9] Wikipedia, «Wikipedia MQTT,» 30 Mai 2018. [Online]. Available: <https://de.wikipedia.org/wiki/MQTT>.
- [10] Wikipedia, «Wikipedia Mobile App,» 1 Juni 2018. [Online]. Available: https://de.wikipedia.org/wiki/Mobile_App.
- [11] Tinkerforge, «Tinkerforge Led-Strip-Bricklet Java Dokumentation,» [Online]. Available: https://www.tinkerforge.com/de/doc/Software/Bricklets/LEDStrip_Bricklet_Java.html.
- [12] Eclipse, «Eclipse Paho Bibliothek,» [Online]. Available: <https://www.eclipse.org/paho/>.
- [13] Chiralcode, «Github Android Color Picker,» 2 Oktober 2014. [Online]. Available: <https://github.com/chiralcode/Android-Color-Picker/>.
- [14] Android, «Android Camera 2,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/hardware/camera2/package-summary>.
- [15] Android, «Android CameraCaptureSession,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/hardware/camera2/CameraCaptureSession>.
- [16] Android, «Android TextureView,» 6 Juni 2018. [Online]. Available: <https://developer.android.com/reference/android/view/TextureView>.
- [17] Wikipedia, «Wikipedia Gammakorrektur,» 23 Mai 2018. [Online]. Available: <https://de.wikipedia.org/wiki/Gammakorrektur>.
- [18] Tinkerforge, «Tinkerforge FrameRenderedListener Java Dokumentation,» [Online]. Available: https://www.tinkerforge.com/de/doc/Software/Bricklets/LEDStrip_Bricklet_Java.html#BrickletLEDStrip.FrameRenderedListener.

1 Einleitung

1.1 Zielsetzung

1.1.1 Offizielle Aufgabenstellung

Diese Arbeit beschäftigt sich mit der Ansteuerung von komplexen Lichtszenen (künstlicher Sternenhimmel / beleuchtete Krippenspiele / optische Werbung / ...), bei denen die räumliche Verteilung der einzelnen adressierbaren Leuchtmittel grundsätzlich unbekannt ist.

Dazu wird einem optischen Verfahren ad Hoc automatisch der relative Standort der einzelnen Leuchtmittel in Bezug zu einem Referenzpunkt ermittelt, sodass mit Hilfe der entstandenen Mapping Lichtszenen für den Referenzpunkt korrekt erstellt werden können.

Das Verfahren muss automatisch, mit Hilfe von handelsüblicher gut verfügbarer Hardware erfolgen, und muss in der Handhabung benutzerfreundlich und möglichst einfach zu bedienen sein.

(Koenig, R. 2018)

1.1.2 Detaillierte Aufgabenstellung

Die offizielle Aufgabenstellung ist relativ allgemein gehalten. Aus ihr ergeben sich aber weitere, etwas detailliertere Punkte. Grundsätzlich können verschiedene Beleuchtungsanlagen vorhanden sein. Ihre Leuchtquellen können unterschiedlich angeordnet sein. Das bedeutet, es ist nicht bekannt, wo im Raum sich welche Leuchtquelle befindet. Dadurch können die einzelnen Leuchtquellen solange kein Gesamtbild erzeugen, wie keine Zuordnung besteht. Eine Zuordnung besteht dann, wenn genau bekannt ist, welche Leuchtquelle sich an welcher Position im Koordinatensystem befindet. Diese Zuordnung zu erstellen, ist Aufgabe dieser Bachelorthesis.

2 Pflichtenheft

2.1 Beschrieb

2.1.1 Produkteinsatz

In herkömmlichen Bildschirmen ist jedes Pixel genau einem „geografischen“ Punkt zugeordnet. Ein Programm kann nun ganz einfach jedes einzelne Pixel ansteuern und ein Muster erzeugen. Bei vielen Anwendungen im täglichen Gebrauch ist dies jedoch nicht der Fall. Leuchtquellen wie LEDs von Lichterketten, Straßenlaternen oder Innenbeleuchtungen können zufällig angeordnet sein. Und jedes Mal, wenn etwas geändert wird, sind die Leuchtquellen wieder an anderen Positionen und müssen mühsam von Hand neu konfiguriert werden. Unsere Bachelorthesis soll diesen Mapping-Vorgang automatisieren.

Beispielsweise will jemand eine Party mit dem Thema Italien organisieren. Er besitzt eine Lichterkette mit beispielsweise 200 LEDs. Die einzelnen LEDs sind RGB-Wiedergabefähig. Das bedeutet, sie können die Farben Rot, Grün und Blau annehmen. Je nach Mischverhältnis können die meisten für das menschliche Auge sichtbaren Farben dargestellt werden. Mit diesen LEDs soll das Wohnzimmer für den Abend dekoriert werden. Verlegt und angeschlossen ist bereits alles. Man muss nur noch wissen, welche LED sich wo befindet. Hat man viele LEDs, gestaltet sich das sehr schwierig, da man jede einzelne ansteuern und sich deren Nummer merken muss. Mit der Mobile-App unserer Thesis können die einzelnen LEDs ganz einfach lokalisiert und anschliessend angesteuert werden. Dazu wird einfach eine Kamera so auf die ganze Beleuchtungsanlage gerichtet, dass alle LEDs in ihrem Sichtfeld sind. Anschliessend muss man nur noch auf „Mappen“ klicken. Nach dem automatischen Mapping-Vorgang können die LEDs ganz einfach, zum Beispiel in den Farben Italiens (grün, weiß, rot), zum Leuchten gebracht werden.

Wichtig zu beachten ist allerdings, dass das Ganze nur in einer Ebene funktioniert. Die Leuchtquelle müssen zwar nicht zwingend alle in der gleichen Ebene angeordnet sein, das Bild erscheint aber immer nur „zweidimensional“ und nur von einem einzigen Punkt aus als korrekt (Parallaxenfehler können auftreten). Es ist also schlussendlich eine zweidimensionale Projektion der tatsächlichen Anordnung der Leuchtquellen.

Zielgruppen sind alle Personen, die ein Smartphone bedienen können und Freude an Leuchtdekorationen haben. Alle die nicht immer die gleiche Dekoration haben, sich aber keine Zeit für ein mühsames manuelles Einstellen nehmen wollen, können auf den LED-Mapper zurückgreifen.

2.1.2 Ablauf

Nach dem Starten der Mobile-App befindet sich der Benutzer in einem Menu. Er kann hier entscheiden, ob er zum Beispiel eine alte Konfiguration laden oder einen neuen Mapping-Vorgang starten will. Sofern alle benötigten Services vorhanden sind, kann der Benutzer den Mapping-Vorgang starten. Dazu richtet er eine Kamera (die von einem Smartphone, oder eine externe) auf den gesamten Bereich, in welchem sich die Leuchtquellen befinden. Wichtig ist, dass die Kamera sehr ruhig gehalten wird. Der Benutzer betätigt einen Button auf dem Handy und der Mapping-Vorgang wird gestartet. Nacheinander leuchten die Leuchtquellen (zum Beispiel LEDs) auf und werden gemappt. Nachdem der Mapping-Vorgang erfolgreich abgeschlossen wurde, kann der Benutzer über das Menu zu verschiedenen Aktionen gelangen. Zum Beispiel kann er jeder Leuchtquelle eine bestimmte Farbe zuweisen und bestimmen, ob diese leuchten soll oder nicht. Weitere Aktionen sind im Kapitel „Anforderungen“ (Kapitel 2.5) weiter unten in diesem Dokument zu finden.

2.1.3 Detaillierte Beschreibung

Das Ziel des Projektes ist, verschiedene Services zur Verfügung zu stellen, die miteinander so kommunizieren, dass zufällig angeordnete Leuchtquellen über ein User Interface gemappt und anschliessend angesteuert werden können. Das gesamte Programm funktioniert nur genau dann, wenn alle erforderlichen Services zur Verfügung stehen. Ein einzelner Service alleine ist verhältnismässig nutzlos. Der Vorteil dieser Struktur ist jedoch, dass ein einzelner Service von einer Drittperson problemlos für ein anderes Gerät geschrieben werden kann, und wenn dieser die Schnittstellen richtig implementiert hat, kann der einzelne Service ganz einfach ausgetauscht werden. Im Zentrum dieser Struktur

steht ein Agent. Er nimmt Daten von den verschiedenen Services entgegen und leitet entsprechende Reaktionen ein.

Mit Hilfe eines Kamera-Services sollen alle Leuchtquellen, also die gesamte Beleuchtungsanlage, auf einmal erfasst werden. Der Benutzer stellt eine Kamera (in unserem Fall eine Handykamera) auf ein Stativ und richtet sie auf den Bereich, den er erfassen will. Nun kann er über ein User-Interface (in unserem Fall eine Mobile-App), den Befehl zum Starten des Mapping-Vorgangs geben. Der Agent nimmt diesen entgegen und meldet dem Leuchtquellen-Service, er soll einen Initialzustand einnehmen. Das bedeutet, alle Leuchtquellen werden ausgeschaltet. Der Initialzustand wird für das Schiessen eines Initialfotos benötigt. Sobald der Leuchtquellen-Service dem Agent meldet, dass der Initialzustand eingenommen ist, befiehlt der Agent dem Kamera-Service, dass er ein erstes Foto schießen soll. Sobald das Foto beim Agent eingetroffen ist, leitet er dieses an einen Datenverarbeitungs-Service weiter. Dieser wiederum merkt sich das Foto und bestätigt dem Agent den Erhalt. Der Agent meldet nun dem Leuchtquellen-Service, dass er die erste Leuchtquelle, zum Beispiel ein einzelnes LED, einschalten soll. Der Leuchtquellen-Service bestätigt dem Agent das Einschalten. Der Agent wiederum verlangt nun vom Kamera-Service ein nächstes Bild. Auch dieses leitet er an den Datenverarbeitungs-Service weiter. Der Datenverarbeitungs-Service vergleicht die beiden Bilder und stellt fest, an welcher Position sich die Helligkeit drastisch verändert hat. Dort muss sich also die erste Leuchtquelle befinden. Er meldet die Koordinaten dem Agent zurück, und dieser fügt ihnen die entsprechende LED-Nummer zu und merkt sich diese Information. Anschliessend meldet der Agent dem Leuchtquellen-Service, dass wiederum der Initialstatus eingenommen werden soll und alles beginnt von vorne. Dieser Mapping-Vorgang geht nun so lange weiter, bis alle Leuchtquellen erfasst wurden. Da jeder weitere Schritt immer vom Vorangegangenen abhängig ist und die Programmierung entsprechend erfolgt, werden Race-Conditions gleich von vornherein verhindert. Der Agent zum Beispiel kann erst weiter arbeiten, wenn er vom Datenverarbeitungs-Service gemeldet bekommen hat, dass der vorhergehende Schritt abgeschlossen wurde. Folgende Grafik zeigt das Zusammenspiel der einzelnen Services:

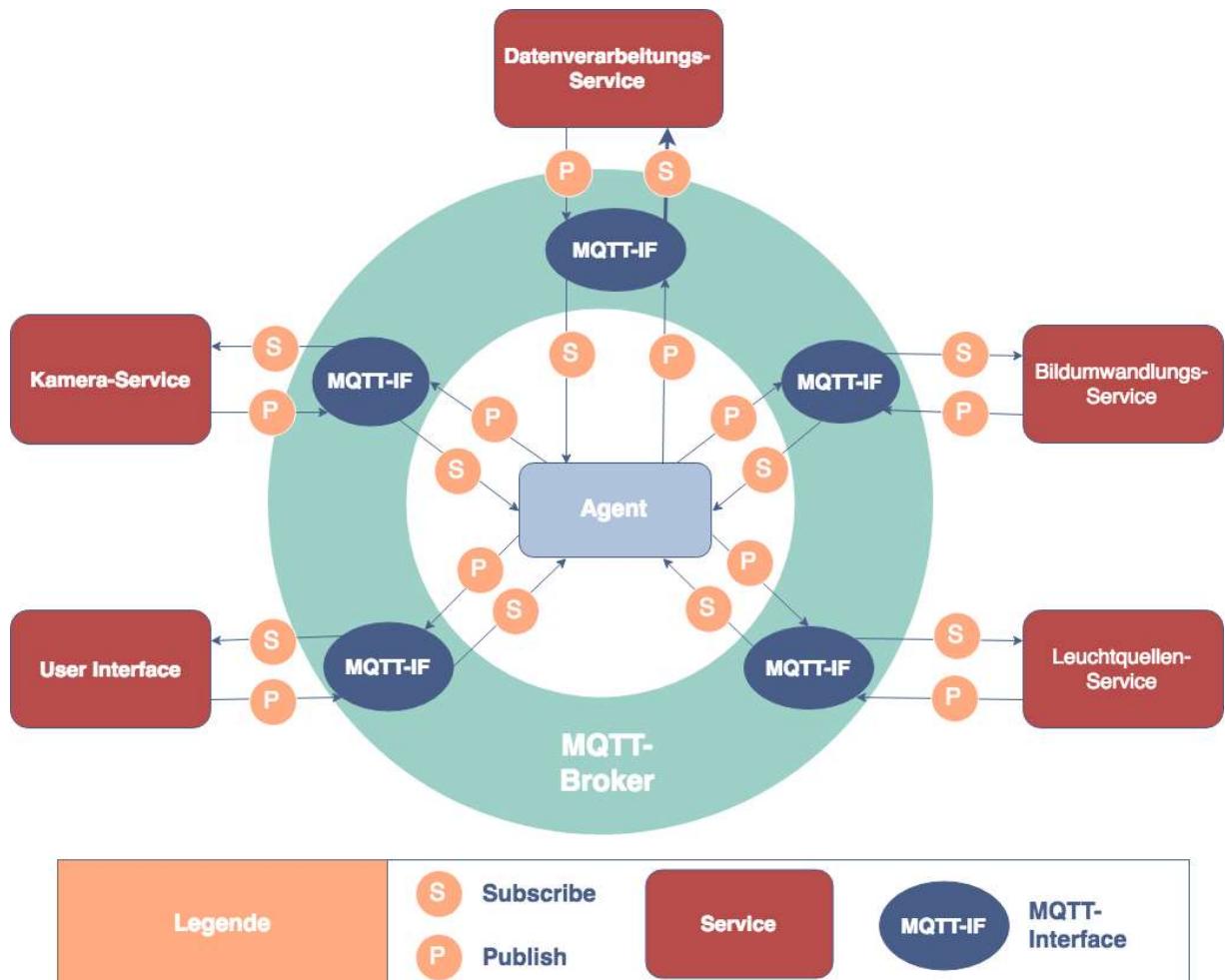


Abbildung 1 Kommunikationsübersicht der Service

2.2 Vorgehen

Das Endprodukt soll aus verschiedenen Services bestehen. Das bedeutet, die einzelnen Komponenten Kamera, User Interface und Leuchtquellenansteuerung funktionieren in sich geschlossen und stellen ausschliesslich benötigte Daten einer Schnittstelle zur Verfügung. Dies ermöglicht ein einfaches Austauschen der Komponenten, sofern sie die Anforderungen an die Schnittstelle erfüllen. Das „Managen“ der Schnittstelle übernimmt ein Agent. Er bestimmt, abhängig von Benutzereingaben, wann welcher Service was zu tun hat. Die Kommunikation läuft über das IoT-Protokoll MQTT (Message Queue Telemetry Transport).

Als erstes werden ein Agent, sowie die Services für die Kamera, die Kommunikation und die Mobile-App erstellt. Als Beleuchtungsanlage wird in einem ersten Schritt ein Bildschirm dienen. Auf ihm wird ein grobes Raster dargestellt, welches verschiedene Farben und Helligkeitsstufen annehmen kann. Damit wird eine einfache Beleuchtungsanlage simuliert, die einzelne Leuchtquellen individuell ansteuern kann. Diese „simulierte“ Beleuchtungsanlage dient hauptsächlich dem Sammeln von ersten Erfahrungen im Umgang mit der Bilderfassung. Sobald alle Tests dafür erfolgreich abgeschlossen worden sind, wird ein weiterer Leuchtquellen-Service, speziell für das Ansteuern von LED-Ketten implementiert. Diese LED-Ketten stellen dann eine weitere Beleuchtungsanlage dar, die vom LED-Mapper unterstützt wird. Ebenfalls wird die erste Version der Mobile-App nur grundsätzliche Funktionen wie das Erfassen der Pixel und deren Ein- und Ausschalten enthalten.

2.3 Stakeholder

Zielpublikum	Leute, die Freude an Beleuchtungsanlagen haben und sich nicht mit technischen Details zum Ansteuern dieser abgeben wollen.
Projektleiter	Elia Bösiger
Entwicklungsteam	Patrik Aebsicher Elia Bösiger
Betreuer	Reto Koenig
Experte	Igor Metz

2.4 Systemabgrenzung

2.4.1 Use-Case-Diagramm

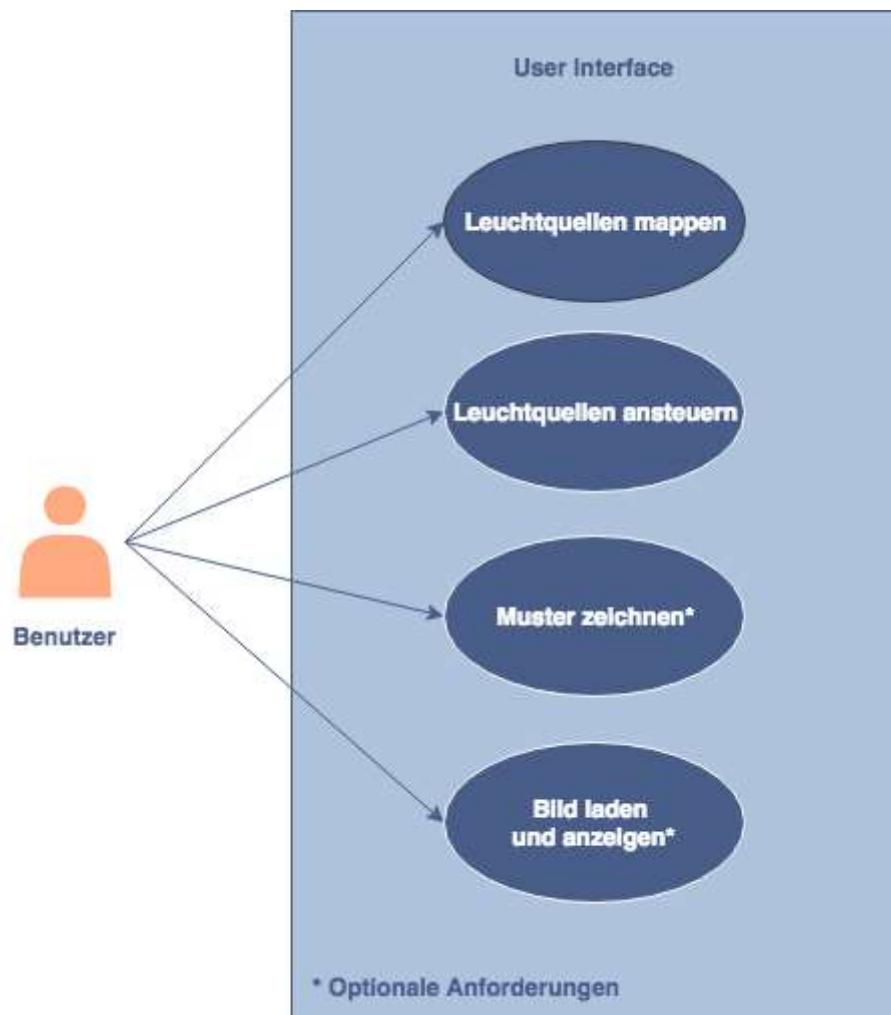


Abbildung 2 User Interface: User-Case Diagramm

2.4.2 Was ist nicht Teil des Projektes

Modul	Beschreibung
Wartung	Die Wartung der Applikation ist nicht Teil des Projektes.
Mehrsprachigkeit	Die Applikation wird lediglich in Deutsch realisiert.
Kompatibilität der Mobile-App	Die Mobile-App wird lediglich für Android-Geräte entwickelt.
3D	Die Motive, die von diesem Projekt schlussendlich dargestellt werden sind ausschliesslich zweidimensional (wie ein Foto). Die 3. Dimension ist kein Teil dieses Projektes.

2.5 Anforderungen

2.5.1 Muss-Kriterien

Modul	Beschreibung
Mapping-Vorgang	Der Benutzer kann eine Kamera auf eine Beleuchtungsanlage richten und ihre einzelnen Leuchtketten automatisch mappen lassen. Mappen bedeutet, die Leuchtketten werden einzeln durchgeschaltet und deren Positionen gespeichert. Die Lokalisierung der Leuchtketten funktioniert nur zweidimensional. Der Benutzer ist dafür verantwortlich, dass der Bereich ganz erfasst wird, er sich aber nicht so weit davon entfernt aufhält, dass einzelne Quellen nicht mehr unterschieden werden können. Hat er den Bereich gewählt, startet er durch Knopfdruck den Mapping-Vorgang. Ist dieser Vorgang abgeschlossen, gibt die Mobile-App dem Benutzer eine entsprechende Rückmeldung. Während des Mapping-Vorgangs ist der Benutzer dafür verantwortlich, dass die Kamera ruhig gehalten wird und sich nicht zu sehr bewegt.
Beleuchtungsanlagen ansteuern	Der Benutzer kann die einzelnen Leuchtketten von verschiedenen Beleuchtungsanlagen ansteuern und ihnen zum Beispiel eine fixe Farbe oder Helligkeit zuweisen. Auch kann er ihnen hier sagen, ob sie ein oder ausgeschaltet sind.
Mobile-App	Die Mobile-App wird für Android-Geräte entwickelt. Sie enthält grundsätzlich nur das User Interface, welches mobilefreundlich gestaltet wird. Es kann via Android-Tablets und -Smartphones auf die Mobile-App zugegriffen werden.
Kamera-Service	Der Kamera-Service ist für die Kamera zuständig. Er schießt Fotos und stellt diese dem Agent zur Verfügung.
Bildschirm-Service	Der Bildschirm-Service ist ein Leuchtketten-Service. Ein Leuchtketten-Service stellt eine spezifische Implementation einer Beleuchtungsanlage dar. Er ist für das korrekte Anzeigen des Rasters (welches in der ersten Testphase anstelle von anderen Leuchtketten benutzt wird) zuständig.
LED-Strip-Service	Der LED-Strip-Service ist in einer zweiten Phase zuständig für das Schalten der LED-Strips von Tinkerforge. Auch er ist ein Leuchtketten-Service.
Datenverarbeitungs-Service	Der Datenverarbeitungs-Service vergleicht die verschiedenen Bilder miteinander und definiert schlussendlich die Position der einzelnen Leuchtketten.
Agent	Der Agent verwaltet alle Services und ist für das Funktionieren der gesamten Anwendung verantwortlich. Er nimmt Befehle von der Mobile-App entgegen und ist zuständig, dass diese korrekt ausgeführt werden. Wurde ein solcher Befehl richtig ausgeführt, meldet er dies an die Mobile-App und somit an den Benutzer zurück.
Kommunikation	Die Kommunikation läuft über einen MQTT-Broker. MQTT ist die Schnittstelle zu sämtlichen Services. Hier ist definiert in welcher Form die Befehle und Daten abgelegt werden müssen, damit die anderen Services verstehen, was sie zu tun haben.

2.5.2 Kann-Kriterien

Modul	Beschreibung
Muster zeichnen	Der Benutzer kann, zum Beispiel mit dem Finger, einfache Muster auf den Bildschirm zeichnen, welche dann von den Leuchtquellen abgebildet werden.
Bild laden und anzeigen	Der Benutzer kann ein vorhandenes Bild laden (zum Beispiel eine Italien- flagge) und dies von einer Beleuchtungsanlage abbilden lassen. Dabei ist zu beachten, dass die "Auflösung", welche von einer Beleuchtungsanlage dargestellt werden kann, abhängig von ihrer Anzahl Leuchtquellen ist.

2.6 Test-Cases

2.6.1 Muss-Kriterien

Modul	Beschreibung
Mapping-Vorgang	Die einzelnen Leuchtquellen einer Beleuchtungsanlage werden erkannt und korrekt gemappt. Die Tests finden unter „Laborverhältnissen“ statt. Das bedeutet, wir können einen optimalen Kontrast von der Oberfläche, auf welcher die einzelnen Leuchtquellen liegen, zu den Leuchtquellen selber bilden. Auch das Lichtverhältnis können wir bestimmen (nicht zu hell und nicht zu dunkel).
Beleuchtungsanlage ansteuern	Die Leuchtquellen der Beleuchtungsanlage werden nach erfolgreichem Mapping auch korrekt angesteuert.
Mobile-App	Die Handhabung der Mobile-App ist intuitiv.
Kamera-Service	Der Kamera-Service liefert auf Anfrage jeweils ein Foto.
Bildschirm-Service	Der Bildschirm-Service schaltet eine bestimmte Kachel ein und aus. Auch die Farbe stimmt mit der gewünschten überein.
LED-Strip-Service	Der LED-Strip-Service schaltet eine bestimmte LED ein und aus. Auch die Farbe stimmt mit der gewünschten überein.
Datenverarbeitungs-Service	Der Datenverarbeitungs-Service erkennt die eingeschalteten Leuchtquellen korrekt und gibt dem Agent die entsprechenden Koordinaten zurück.
Agent	Der Agent nimmt die Befehle korrekt entgegen und leitet sie dem entsprechenden Service weiter.
Kommunikation	Der MQTT-Broker nimmt die Messages entgegen und veröffentlicht sie im entsprechenden Topic.
Race-Conditions	Während allen Tests wird darauf geachtet, dass keine Race-Conditions auftreten. Das bedeutet, bei der Kommunikation zwischen den einzelnen Services wird sichergestellt, dass die Nachrichten auch angekommen sind. Der Empfänger der Nachricht bestätigt dies dem Sender. Dadurch wird verhindert, dass zum Beispiel Bilder des Kamera-Service nicht in der richtigen Reihenfolge beim Datenverarbeitungs-Service ankommen und dieser dann falsche Schlüsse zieht.

2.6.2 Kann-Kriterien

Modul	Beschreibung
Muster zeichnen	Über die Mobile-App können mit dem Finger Muster gezeichnet werden.
Bild laden und anzeigen	Ein Bild (Dateityp ist noch nicht definiert) kann mit der Mobile-App geladen und korrekt auf einer Beleuchtungsanlage angezeigt werden.

3 Systemübersicht

3.1 Grundidee

Mit den Beleuchtungsanlagen soll schlussendlich ein zweidimensionales Bild dargestellt werden können. Die einzelnen Leuchtquellen können sich zwar theoretisch auch in einem dreidimensionalen Raum befinden, das Bild soll aber anschliessend nur von einem Beobachtungspunkt aus korrekt erscheinen. Die einzelnen Leuchtquellen werden also auf eine Ebene projiziert und das Problem kann auf zwei Dimensionen reduziert werden. Die Tatsache, dass nur ein einziger Beobachtungspunkt besteht, vereinfacht das Mappen erheblich. Die Idee ist, eine Kamera fix vor einer Beleuchtungsanlage zu installieren. Anschliessend wird ein Vorgang gestartet, der alle Leuchtquellen einmal ein- und wieder ausschaltet. Die Kamera schiesst jeweils ein Foto davon und ein Algorithmus kann die Koordinaten jeder Leuchtquelle auf der Ebene bestimmen. Dieser Vorgang ist das eigentliche Mapping und die Hauptaufgabe des LED-Mappers.

Danach soll das User Interface des LED-Mappers die gesamte Beleuchtungsanlage virtuell darstellen. Über dieses virtuelle Abbild sollen die einzelnen Leuchtquellen konfiguriert werden können. Zum Beispiel soll die Farbe der Leuchtquelle an Position X,Y grün sein. Damit bei vielen Leuchtquellen nicht alle Einstellungen manuell eingegeben werden müssen, soll der LED-Mapper um eine weitere Funktionalität erweitert werden. Dem LED-Mapper soll ein Bild (zum Beispiel eine JPEG-Datei) mitgegeben werden können, welches er auf der gemappten Beleuchtungsanlage abbildet. Soll zum Beispiel die Schweizerflagge abgebildet werden, weiss der LED-Mapper, welche Leuchtquelle er dazu Rot und welche Weiss einfärben muss. Wie detailliert diese Motive sein können, hängt natürlich von der Anzahl vorhandener Leuchtquellen ab. Mit bereits wenigen Leuchtquellen, können Länderflaggen oder Schriftzüge dargestellt werden. Besteht die Beleuchtungsanlage aus vielen Leuchtquellen, kann das Motiv auch eine komplexere Figur sein.

3.2 Einzelne Komponenten

Aus der Grundidee ergeben sich verschiedene Komponenten. Als erstes muss die Hardware, also die Beleuchtungsanlage selber, angesteuert werden können. Als zweites wird eine Kamera benötigt, die die gesamte Beleuchtungsanlage erfassen kann. Die Kamera muss Befehle entgegennehmen und Bilder weiterleiten können. Weiter muss irgendwo bestimmt werden, welche Leuchtquelle sich wo auf der Ebene befindet. Und Schlussendlich sollen all diese Funktionen möglichst einfach von einem Ort aus gesteuert werden. Dort sollen auch gleich die Motive ausgewählt und an die Beleuchtungsanlage übertragen werden können. Aus diesen Anforderungen ergeben sich also folgende Komponenten:

1. Beleuchtungsanlage
2. Kamera
3. Datenverarbeitung
4. Benutzerschnittstelle

3.3 Gliederung in Services

Die einzelnen Systemkomponenten werden in Services aufgeteilt. Das bedeutet, für jede Teilaufgabe des gesamten Systems wird ein eigenes „Programm“ (ein Service) geschrieben. Jeder dieser Services verfügt über eine spezifizierte Schnittstelle, über welche er mit dem Gesamtsystem kommunizieren kann. Über diese Schnittstelle erhält der Service Befehle und stellt im Gegenzug seine eigenen Ergebnisse bereit. Der Kamera-Service zum Beispiel kann den Befehl zum Schiessen eines Fotos erhalten. Er führt diesen Befehl aus, und wenn der abgeschlossen ist stellt er das geschossene Foto bereit. Die Anzahl Services im Gesamtsystem beträgt im Moment fünf.

1. Benutzerschnittstelle (User Interface)
2. Kamera-Service
3. Leuchtquellen-Service
4. Datenverarbeitungs-Service
5. Bildumwandlungs-Service

Diese Architekturform ist sehr offen und ermöglicht ein einfaches Hinzufügen von anderen Services. Ebenfalls können bestehende Services leicht ausgetauscht werden. Für ein korrektes Interagieren mit dem Gesamtsystem müssen diese lediglich die Schnittstelle korrekt implementieren. Als Beispiel kann ein Service, der für eine bestimmte Beleuchtungsanlage geschrieben wurde durch einen andern ausgetauscht werden. Dieser kann eine Hardwarekonfiguration unterstützen, die sich von der ersten unterscheidet.

Damit das Gesamtsystem nicht unübersichtlich wird, wurde für den LED-Mapper definiert, dass die einzelnen Services keine Kenntnis von den anderen Services haben. Jeder existiert nur für sich und führt ausschliesslich die Befehle aus, welche an seiner Schnittstelle anstehen. Das alles wird von einem Agent koordiniert. Er verbindet sich mit sämtlichen Schnittstellen, die von den Services zur Verfügung gestellt werden. Der Agent ist derjenige, der entscheidet was bei welchem Befehl getan werden muss und welcher Service ihn ausführt. Als Beispiel sendet die Benutzerschnittstelle den Befehl alle Leuchtketten einzuschalten. Sie weiss dabei nicht welche Hardware konkret angeschlossen ist. Für sie werden einfach Leuchtketten eingeschaltet. Da die Benutzerschnittstelle ein Service ist, wird der Befehl vom Agent entgegengenommen. Dieser leitet ihn an den Service weiter, der die Beleuchtungsanlage abbildet. Der Leuchtketten-Service schaltet alle Leuchtketten ein und meldet dem Agent zurück, dass der Befehl ausgeführt wurde. Auch diese Rückmeldung wird vom Agent an die Benutzerschnittstelle zurückgegeben. So wird der Benutzer stets informiert, ob sein Befehl auch ausgeführt werden konnten oder nicht.

3.4 Serviceübersicht

3.4.1 User Interface

Das User Interface (im Folgenden auch als Benutzerschnittstelle bezeichnet) nimmt alle Befehle vom Benutzer entgegen. Sie gibt ausserdem Auskunft darüber welche Services im Moment zur Verfügung stehen. Weiter verwaltet sie bereits erstellte Konfigurationen und speichert diese gegebenenfalls in einer Datenbank.

3.4.2 Kamera-Service

Der Kamera-Service ist für das korrekte Funktionieren einer Kamera zuständig. Er schießt auf Befehl ein Foto und stellt dieses an seiner Schnittstelle bereit.

3.4.3 Leuchtketten-Service

Der Leuchtketten-Service kommuniziert mit der jeweils spezifischen Hardware, also der Beleuchtungsanlage. Er kann auf sämtliche Leuchtketten zugreifen, sie ein- und ausschalten und gegebenenfalls ihre Farbe ändern.

3.4.4 Datenverarbeitungs-Service

Der Datenverarbeitungs-Service vergleicht jeweils zwei Fotos und überprüft sie auf Unterschiede. Das erste Foto zeigt immer die Beleuchtungsanlage in ausgeschaltetem Zustand. Beim zweiten Foto ist eine bestimmte Leuchtkette eingeschaltet, was im Vergleich zum ersten einen Helligkeitsunterschied ergibt. Dies stellt der Datenverarbeitungs-Service fest und errechnet daraus die Koordinate der Leuchtkette. Diese Information stellt er anschliessend an seiner Schnittstelle bereit.

3.4.5 Bildumwandlungs-Service

Der Bildumwandlungs-Service errechnet aus den Koordinaten eines vorhandenen Mappings und einem Bild (zum Beispiel JPEG) eine konkrete Szenerie (zum Beispiel „Italienflagge“). Er weist jeder Leuchtkette einer Beleuchtungsanlage eine Farbe möglichst so zu, dass das Bild auf der Beleuchtungsanlage durch den Betrachter erkannt wird.

3.4.6 Agent

Der Agent ist eigentlich kein Service. Er ist viel mehr für ihre Koordination zuständig. Der Agent entscheidet, was in welchem Systemzustand getan werden muss. Er fordert zum Beispiel während eines Mapping-Vorgangs ein Foto vom Kamera-Service an und leitet es entweder als Referenz- oder Vergleichsfoto an den Datenverarbeitungs-Service weiter. Dazwischen gibt der Agent dem Leuchtquellservice zum richtigen Zeitpunkt den Befehl, eine Leuchtquelle ein- oder auszuschalten. Er orchestriert also das ganze System und bringt die verschiedenen Services dazu, korrekt ineinander zu spielen.

3.5 Kommunikation via MQTT

Die Kommunikation zwischen den jeweiligen Services und dem Agent findet über das MQTT-Protokoll (Message Queue Telemetry Transport) statt. MQTT ist ein IoT-Protokoll (Internet of Things) und wird häufig für die Kommunikation zwischen verschiedenen Geräten verwendet. MQTT basiert auf dem „Publish- / Subscribe-Verfahren“. Der Kern ist ein Server (Broker) der von den einzelnen Clients zum Austauschen von Nachrichten verwendet wird. Jeder Client kann sich an einem Topic (Thema) des Brokers anmelden (Subscribe) und erhält anschliessend alle Nachrichten, die auf diesem Topic veröffentlicht (Publish) werden. Zum Veröffentlichen einer Nachricht kann der Client ein „Publish“ auf ein bestimmtes Topic durchführen. Um auf ein Topic „publishen“ zu können, muss der Client nicht zwingend auch auf dieses Topic „subscribed“ haben. Der Broker leitet anschliessend die veröffentlichte Nachricht an alle Clients weiter, die sich an diesem bestimmten Topic angemeldet haben.

4 Technische Systembeschreibung

4.1 Hilfsmittel und Werkzeuge

4.1.1 Verwendete Technologien

Java	Java ist eine Open-Source-Programmiersprache der Firma Oracle. Java ist weit verbreitet und wird deshalb von vielen Geräten und Plattformen unterstützt.
JavaFX	JavaFX ist ein Framework zum Erstellen von Java-Applikationen und wird häufig für das Erstellen von grafischen Benutzeroberflächen (GUI) verwendet.
MQTT	MQTT steht für Message Queue Telemetry Transport und ist ein Kommunikationsprotokoll, das vor allem im Internet of Things (IoT) stark verbreitet ist. Das Kommunikationsprotokoll basiert auf dem publish/subscribe -Prinzip. Ein Client subscribet ein bestimmtes Topic und erhält anschliessend alle Nachrichten, die unter diesem Topic published werden.
OpenCV	OpenCV ist eine Open Source Bibliothek die Methoden zur Bearbeitung von Bildern zur Verfügung stellt. Sie besitzt eine Schnittstelle zu Java.
Tinkerforge	Tinkerforge stellt diverse Hardware zur Verfügung, auf welche leicht von verschiedenen Softwareplattformen zugegriffen werden kann. Tinkerforge besitzt eine Schnittstelle zu Java.

4.1.2 Verwendete Unterstützungstools

Eclipse	Eclipse ist eine Open-Source-Programmierumgebung. Sie unterstützt die Entwicklung der Programmiersprache Java.
Android Studio	Android Studio ist eine Open-Source-Programmierumgebung von Google für mobile Applikationen. Android Studio ist gemacht für Mobile-Apps, die auf dem Android Betriebssystem laufen. Die Apps werden in Java geschrieben.
Draw.io	Mit der Webapplikation Draw.io können Diagramme sowie Darstellungen eines Systems erstellt werden.
Astah	Astah ist eine Open-Source-Applikation für die Erstellung von Diagrammen. Es eignet sich besonders für die Erstellung der unterschiedlichen Klassendiagramme.
Marvel	Marvel ist eine Webapplikation für die Erstellung von GUI-Entwürfen.

4.2 Grundsätzliche Servicearchitektur

Grundsätzlich sind alles Services ähnlich aufgebaut. Ein erster grober Unterschied besteht allerdings zwischen Android-Services (den Mobile-Apps) und der Standardsoftware (für herkömmliche Plattformen geschrieben). Obwohl sie eine ähnliche Systemstruktur aufweisen, ist dies auf den ersten Blick für den Benutzer nicht ersichtlich. Dies kommt daher, dass eine Android-App andere Anforderungen erfüllen muss, als herkömmliche Software. Der Aufbau richtet sich aber bei allen Services nach dem MVC-Pattern. Auch wenn das Model nicht in allen Services vollständig implementiert wurde. MVC steht für Model-View-Controller und bedeutet:

Model / Datenschicht

Das Model enthält die darzustellenden Daten. Es ist von der Präsentation und der Steuerung unabhängig.

View / Präsentationsschicht

Die Präsentationsschicht ist für die Darstellung der benötigten Daten des Modells sowie die Entgegennahme der Benutzerinteraktionen zuständig.

Controller / Steuerungsschicht

Die Steuerungsschicht verwaltet eine oder mehrere Präsentationen, nimmt von ihnen Benutzeraktionen entgegen und wertet diese aus. Zu jeder Präsentation existiert eine eigene Steuerung. Die Steuerung sorgt ebenfalls dafür, dass Benutzeraktionen wirksam werden, zum Beispiel durch Änderung der Präsentation oder durch Weiterleiten an das Modell. Weiter enthält die Steuerung Mechanismen um die Benutzerinteraktionen der Präsentation einzuschränken.

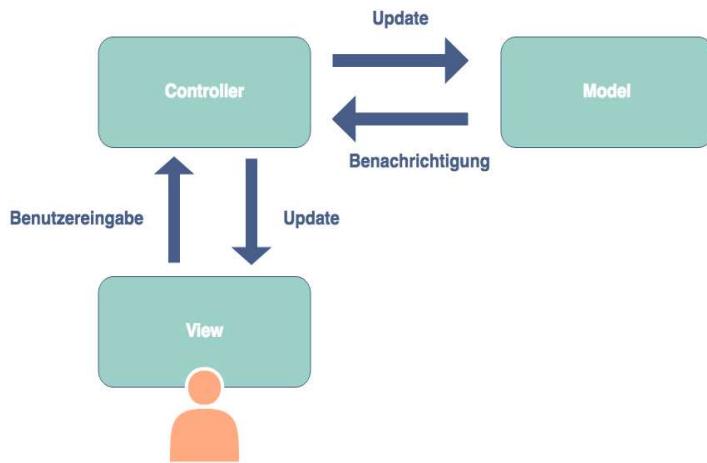


Abbildung 3 MVC-Pattern

Grundsätzlich gilt für alle Services, dass sie sich nach dem Starten erst einmal mit dem MQTT-Broker verbinden müssen. Auf den Mobile-Apps ist dafür ein Menüpunkt vorhanden. Wird ein Standardsoftware-Service gestartet, wird zuerst ein Fenster angezeigt, in welchem der Benutzer aufgefordert wird, die IP-Adresse des Brokers einzugeben. Weiter können hier Service-spezifische Einstellungen vorgenommen werden. Auch enthält dieses Fenster eine Textausgabe, in welcher grundlegende Informationen bereitgestellt werden. Zum Beispiel, ob eine Verbindung zu einem Broker möglich ist oder nicht. Sind die geforderten Einstellungen korrekt vorgenommen worden, lässt sich der eigentliche Service mittels eines „Start-Buttons“ starten. Das Fenster ist weiterhin sichtbar und ermöglicht ein allfälliges Stoppen und Anpassen der Konfiguration. Diese View-Klasse ist der Einstiegspunkt jedes Standardsoftware-Services. Sie instanziert jeweils ein „Haupt-Controller“, welcher für das weitere Funktionieren des Services zuständig ist. Ein solcher Haupt-Controller existiert auch bei den Mobile-App-Services. Der Haupt-Controller instanziert wiederum die MQTT-Controller-Klasse. Diese ist vom Aufbau her in jedem Service gleich. Allerdings gilt es zu beachten, dass sich jeder Service an seinen eigenen Topics anmeldet (Subscribe). Zusätzlich kann der Haupt-Controller noch weitere Unterklassen instanziieren. Auch hier ist die Anzahl aber abhängig von den Anforderungen an den einzelnen Service.

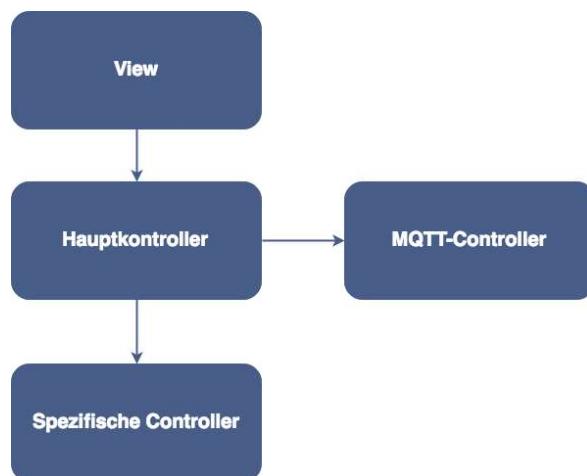


Abbildung 4 Applikations-Struktur

4.2.1 Haupt-Controller

Die Funktion des Haupt-Controllers besteht im Verwalten des Services. Er instanziert den MQTT-Controller und sorgt dafür, dass die vom MQTT-Controller erhaltenen Nachrichten, dem Service zur Verfügung stehen. Der Haupt-Controller entscheidet, was bei welcher ankommenden Nachricht getan werden muss und weist den MQTT-Controller an, entsprechende Nachrichten zu veröffentlichen (publish). Stehen Arbeiten wie zum Beispiel das Berechnen von Bildern an, weist der Haupt-Controller diese Aufgabe einem weiteren Controller zu.

4.2.2 MQTT-Controller

Der MQTT-Controller ist für die Kommunikation mit dem Broker zuständig. Der MQTT-Controller enthält den Client, welcher schlussendlich mit dem Broker kommuniziert. Der MQTT-Controller stellt Methoden zum Verbinden mit dem Broker, zum An- und Abmelden an den Topics, sowie zum Veröffentlichen von Nachrichten bereit. Wann welche Nachricht unter welchem Topic veröffentlicht wird, wird hingegen vom Haupt-Controller bestimmt.

4.3 Benutzerschnittstelle (User Interface)

4.3.1 Funktion

Das User Interface bietet die Benutzerschnittstelle zum gesamten System. Über das User Interface werden den Services Befehle erteilt, sowie die wichtigsten Daten visualisiert. Das User Interface hat folgende Funktionen:

- Verwaltung der Beleuchtungen und der eigenen MQTT-Anbindung
- Starten/Stoppen des Mapping-Vorgangs
- Verwaltung der Szenerien
- Konfiguration der Leuchtquellen

4.3.2 Technologie

Das User Interface ist in der Programmiersprache Java für Android-Geräte realisiert.

4.3.3 Aufbau des User Interfaces

Das User Interface wird nach dem Model-View-Controller (MVC) Pattern aufgebaut.

4.3.3.1 Model

Im folgenden Klassendiagramm wird aufgezeigt, wie die Model-Klassen aufgebaut sind und in welcher Relation sie zueinander stehen. Dies hilft der Verständlichkeit des Gesamtaufbaus der Mobile-App. Die Benennung der Klassen und Variablen wurden für die Verständlichkeit anders benannt als im Java-Code.

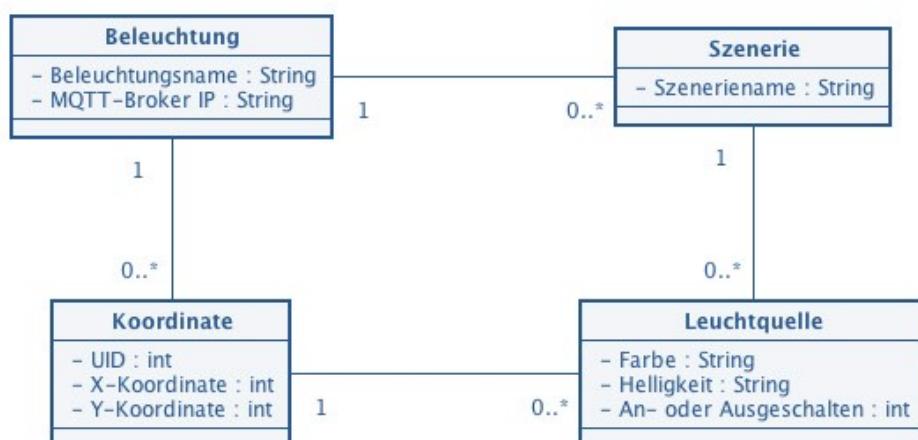


Abbildung 5 UI: Model-Klassendiagramm

- **Beleuchtung:** Die Beleuchtung-Klasse stellt ein Beleuchtungsanlage dar.
- **Szenerie:** Die Szenerie-Klasse stellt eine Szenerie für eine gemappte Beleuchtungsanlage dar.
- **Koordinate:** Die Koordinaten-Klasse stellt eine einzelne, gemappte Leuchtquelle dar.
- **Leuchtquelle:** Die Leuchtquellen-Klasse ist ein Abbild einer Leuchtquelle auf einer Szenerie.

Diese Model-Klassen dienen nicht nur der Zwischenspeicherung sondern werden auch so in der Datenbank abgelegt. Zusätzlich wird eine Service-Klasse definiert, welche alle Services enthält, die sich am MQTT-Broker angemeldet haben. Die Service-Klasse steht in keiner Relation zu den anderen Model-Klassen und wird jeweils vom aktuellsten Stand des MQTT-Brokers abgefüllt.

4.3.3.2 View / Präsentationsschicht

In der Android-Applikation wird jede Ansicht von einer Activity [1] implementiert. Das User Interface besteht aus den folgenden vier Activities:

- Übersicht Beleuchtungen
- Beleuchtungs-Konfiguration
- Übersicht Szenerien
- Szenerie

Ansicht: Meine Beleuchtungen

Wenn der Benutzer die Mobile-App startet, erscheint die Ansicht „Meine Beleuchtungen“. Auf dieser Ansicht sind alle Beleuchtungen, welche bereits erstellt wurden angezeigt.

Wenn die App zum ersten Mal gestartet wird oder sonst keine Beleuchtungen beinhaltet, erscheint der Text „Keine Beleuchtungen vorhanden“. Beim Klick auf eine Beleuchtung werden die dazugehörigen Szenerien angezeigt (Ansicht: Szenerien-Übersicht). Der Benutzer hat in dieser Ansicht ebenfalls die Möglichkeit mit einem Klick auf den gelben „+“ - Button in die Ansicht „Neue Beleuchtung“ zu gelangen um eine neue Beleuchtung hinzuzufügen.

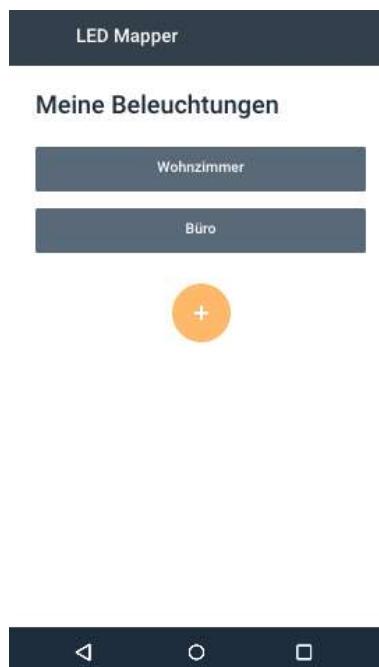


Abbildung 6 UI: Ansicht
"Meine Beleuchtungen"

Ansicht: Neue Beleuchtung

Eine neue Beleuchtung kann in der Ansicht „Neue Beleuchtung“ erstellt werden. Auf diese Ansicht gelangt der Benutzer über die Ansicht „Meine Beleuchtungen“. Bei einem Klick auf den Button „Erstellen und Mappen“ wird nach einer Validierung der Textfelder das Mapping der Leuchtketten gestartet. Jede Beleuchtung besitzt folgende Eigenschaften:

- **Name:** Um die Beleuchtung zu benennen.
- **MQTT-Broker:** IP-Adresse des gewünschten Brokers. Hierbei muss auf die Eingabe einer korrekten IPv4-Adresse geachtet werden

Wenn eine korrekte IP-Adresse eingegeben wurde, kann der Benutzer mit einem Klick auf den Button „Verbindung prüfen“ überprüfen ob eine Verbindung zum MQTT-Broker aufgebaut werden kann. Ist der MQTT-Broker mit der eingegebenen IP-Adresse nicht erreichbar, erscheint eine Meldung. Bei einem erfolgreichen Verbindungsauftakt erscheinen die Services. Die bereits mit dem MQTT-Broker verbundenen Services werden mit einem grünen Punkt, die nicht vorhandenen Services mit einem roten Punkt dargestellt. Während dem Mapping-Vorgang kann der Benutzer entweder das Mapping stoppen oder darauf warten, bis alle Leuchtketten gemappt wurden. In beiden Fällen wird dem Benutzer folgender Dialog mit der Anzahl gemappter Leuchtketten angezeigt.



Abbildung 7 Dialog nach abgeschlossenem Mapping-Vorgang

Der Benutzer kann diese Information mit einem Klick auf den Button „Ok“ bestätigen und wird in die Ansicht „Szenerien-Übersicht“ umgeleitet. Nach einem Mapping-Vorgang sind noch keine Szenerien vorhanden.

LED Mapper

Neue Beleuchtung

Name:
Innenbeleuchtung Wohnstube

MQTT-Broker:
192.168.5.2

Verbindung prüfen

Abbildung 8 UI: Verbindung zum MQTT-Broker prüfen

LED Mapper

Neue Beleuchtung

Name:
Innenbeleuchtung Wohnstube

MQTT-Broker:
192.168.5.2

- Kamera-Service
- Leuchtketten-Service
- Datenverarbeitungs-Service
- Agent

Erstellen und Mappen

Abbildung 9 Ansicht mit verfügbaren Services

Ansicht: Szenerien-Übersicht

Beim Klick auf eine Beleuchtung in der Ansicht „Meine Beleuchtungen“ werden alle Szenerien einer Beleuchtung aufgelistet. Im Beispiel besitzt der Benutzer für sein Büro zwei Szenerien. Eine um eine italienische und eine um eine japanische Flagge darzustellen. Beim Klick auf eine Szenerie wird die Ansicht „Szenerie“ mit den konfigurierten Leuchtketten gestartet. Um eine neue Szenerie zu erstellen kann der Benutzer den gelben „+“-Button anklicken, wodurch er in einem Popup aufgefordert wird, einen Namen für die Szenerie einzugeben:



Abbildung 10 Dialog für das Erstellen einer neuen Szenerie

Die Topbar der Ansicht enthält Icons für die Einstellungen und für das Löschen der Beleuchtung. Beim Klick auf die Einstellung erscheint die Ansicht „Beleuchtungs-Konfiguration“. Der Klick auf das Papierkorb-Icon löscht die Beleuchtung mit ihren Szenerien und führt den Benutzer in die Ansicht „Meine Beleuchtungen“ zurück.

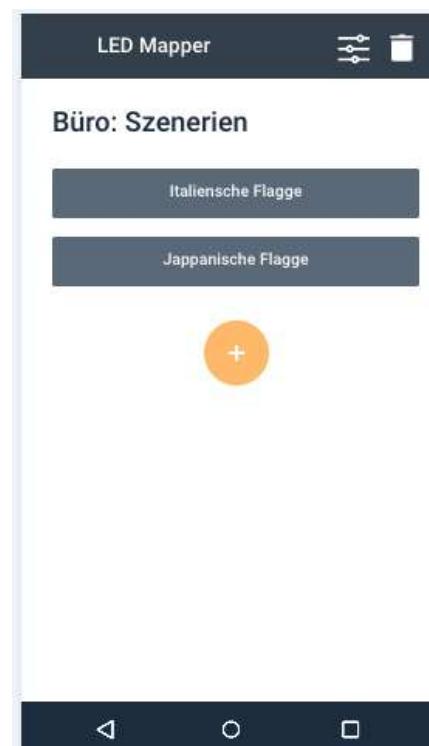


Abbildung 11 UI: Szenerien-Übersicht

Ansicht: Szenerie

In der Ansicht „Szenerie“ wird die gemappte Beleuchtungsanlage virtuell dargestellt. Der Benutzer kann in dieser Ansicht die Leuchtquellen einer Beleuchtungsanlage ansteuern. Dabei kann eine Leuchtquelle an- und ausgeschaltet, sowie die Farbe und Helligkeit konfiguriert werden.

Der Benutzer definiert nun die Farbe und Helligkeit bevor er auf dem virtuellen Abbild zu zeichnen beginnt.

Ist eine Farbe ausgewählt und wird anschliessend eine virtuelle Leuchtquelle mit dem Finger berührt, wird die neue Farbe sowohl auf die virtuelle wie auch auf die reale Leuchtquelle der Beleuchtungsanlage übertragen.

Wenn der Schalter „Leuchtquelle an/aus“ ausgeschalten ist, soll jede Leuchtquelle ausgeschalten werden. Zusätzlich kann eine einzelne Leuchtquelle mit einer Berührung von zwei Sekunden ebenfalls ausgeschalten werden. Die Topbar besteht aus einem Bild- und Papierkorb-Icon. Um die bestehende Szenerie zu löschen, kann der Benutzer das Papierkorb-Icon in der Topbar auswählen. Das Löschen einer Szenerie führt den Benutzer in die Ansicht „Szenerie-Übersicht“ zurück. Beim Klick auf das Bild-Icon, kann der Benutzer ein Bild laden und dies auf der Beleuchtung darstellen.

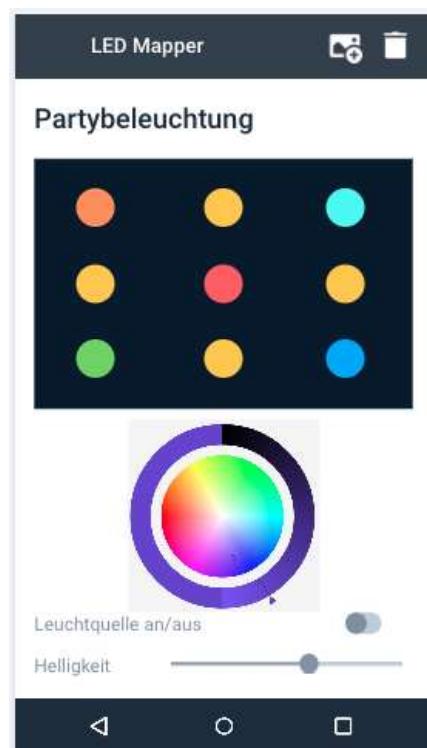


Abbildung 12 UI: Szenerie-Ansicht

Ansicht: Beleuchtungs-Konfiguration

In der Ansicht „Beleuchtungs-Konfiguration“ kann der Benutzer die gleichen Einstellungen vornehmen, wie in der Ansicht für eine neue Beleuchtung. Wenn die Verbindung zum gespeicherten MQTT-Broker aufgebaut werden konnte, werden die Services angezeigt. Zusätzlich kann der Benutzer einstellen, ob man ein neues Mapping der Lichtquellen vornehmen möchte oder nicht. Wenn die Verbindung zum MQTT-Broker nicht aufgebaut werden konnte, werden die Services sowie der Schieber für das erneute Mapping nicht angezeigt:



Abbildung 13 Konfigurationsansicht ohne Verbindung zum MQTT-Broker

Die Ansicht wird durch die gleiche Activity wie die „Neue Beleuchtung“-Ansicht implementiert. Die Activity entscheidet mit den Start-Parametern, ob es sich um eine bestehendes oder ein neues Beleuchtungssystem handelt.

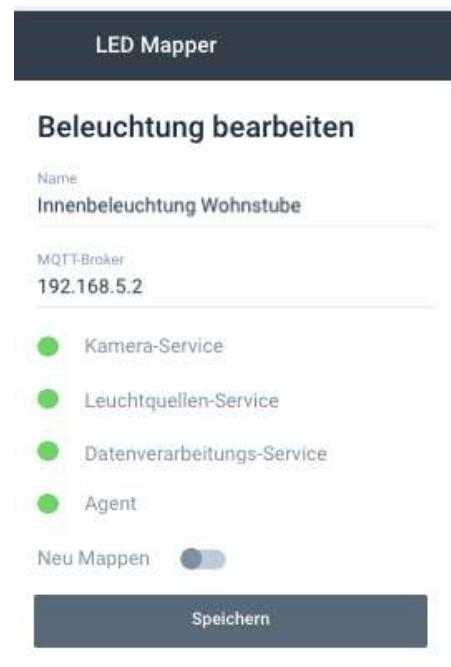


Abbildung 14 UI: Ansicht Beleuchtungs-Konfiguration mit Verbindung zum MQTT-Broker

4.3.3.3 Controller / Steuerungsschicht

Das User Interface besitzt für jede Activity einen Controller um die Benutzerinteraktionen entgegen zu nehmen oder die Ansicht mit Daten zu beliefern. In den folgenden Kapiteln werden einzelne Controller und deren Funktionen beschrieben.

DB-Controller

Um die Daten persistent abzulegen benutzt das Mobile-App eine SQL-Lite Datenbank. Der DB-Controller ist eine erweiterte Klasse der Android-Klasse SQLiteOpenHelper [2]. Die Klasse erstellt bei der ersten Instanz eine Datenbank. Die Model-Klassen dienen der Zwischenspeicherung der Datenbank-Inhalte und sind daher gleich aufgebaut wie die Datenbank-Tabellen. Folgende Daten werden in der SQL-Lite-Datenbank abgelegt:

- Beleuchtungen
- Koordinaten eines Mapping-Vorgangs
- Szenerien
- Leuchtketten einer Szenerie

Die Mobile-App beinhaltet kein Login und speichert somit auch keine Benutzerdaten.

Validierungs-Controller

Die Benutzerangaben in den Ansichten „Neue Beleuchtung“ und „Beleuchtung Konfigurieren“ müssen validiert werden. Zur Validierung gehören die folgenden Eingabefelder:

- Name der Beleuchtung: Der Name der Beleuchtung darf nicht leer sein.
- IP-Adresse des MQTT-Brokers: Die IP-Adresse des MQTT-Broker muss nach dem IPv4-Format eingegeben werden. Das IPv4-Format entspricht 4 Zahlen von 0-255 die jeweils mit einem Punkt getrennt werden (Bsp.: 192.168.255.34).

Der Mapping-Vorgang der Beleuchtung kann zur Validierung der Eingabefelder erst gestartet werden, wenn auch alle Services vorhanden und ausgewählt sind.

Darstellung der virtuellen Leuchtketten

Ein wichtiger Teil des Controllers ist das Darstellen der virtuellen Leuchtketten. Nach dem Mapping-Vorgang der Beleuchtungsanlage, holt sich die Mobile-App alle Koordinaten vom MQTT-Broker und legt sie in der internen Datenbank ab. Um die Leuchtketten auf dem Mobile-App darzustellen wird jeweils die minimale und maximale X- und Y-Koordinate der gemappten Beleuchtungsanlage verwendet. Mit diesen Koordinaten kann der minimale vom maximalen Wert subtrahiert werden. Aus der Subtraktion der X-Werte resultiert die Breite und für die Subtraktion der Y-Werte die Höhe der Szenerie. Auch Android liefert Funktionen um die Breite und Höhe der Android-FrameLayout-Komponente [3], auf der sich das virtuelle Abbild der Beleuchtung befindet, auszulesen. Mit diesen Informationen kann der Faktor für die Darstellung der virtuellen Leuchtketten berechnet werden.

$$xFaktor = \frac{(xMax - xMin)}{\text{Breite virtuelles Abbild}}$$

$$yFaktor = \frac{(yMax - yMin)}{\text{Höhe virtuelles Abbild}}$$

Die Funktion „getCoordinateFactorForView“ ist im Anhang beigelegt (Kapitel 7.1)

Mit dem jeweiligen Faktor kann nun die Koordinate der virtuellen Leuchtquelle berechnet werden. Der Android-Komponente FrameLayout kann der Abstand zum oberen und rechten Rand angegeben werden. Die kleinste X-Koordinate wird am rechten und die grösste X-Koordinate am linken Rand dargestellt. Die kleinste Y-Koordinate wird ganz unten und die grösste Y-Koordinate ganz oben dargestellt. Die Berechnung der Koordinate von einer virtuellen Leuchtquelle sieht folgendermassen aus:

$$\text{Abstand zum rechten Rand} = \text{Originale xKoordinate} * xFaktor$$

$$\text{Abstand zum oberen Rand} = \text{Originale yKoordinate} * yFaktor$$

Die Funktion „getLayoutParamsFor“ ist im Anhang beigelegt (Kapitel 7.2)

Die virtuellen Leuchtquellen werden schlussendlich als Android-Button-Komponenten [4] auf dem Android-FrameLayout dargestellt.

Muster auf die Beleuchtung zeichnen

Um ein Muster auf dem virtuellen Abbild der Mobile-App zu zeichnen wird detektiert, wo sich der Finger auf dem Bildschirm befindet. Bei der Berührung des virtuellen Abbilds wird jedes Mal die Liste aller virtuellen Leuchtquellen abgearbeitet. Befindet sich der Finger auf einer virtuellen Leuchtquelle, werden die Informationen der aktuell ausgewählte Farbe, Helligkeit sowie ob die Leuchtquelle an- oder ausgeschaltet werden soll, als Befehl an den MQTT-Broker übertragen. Um zu Überprüfen ob sich der Finger auf dem virtuellen Abbild befindet, wird ein erweiterter Android View.OnTouchListener [5] verwendet. Dabei unterstützt das User Interface den RGB-Farbraum.

Der RGB-Farbraum mischt die drei Grundfarben Rot, Grün und Blau so, dass alle vom menschlichen Auge wahrnehmbaren Farben dargestellt werden können. [6]

4.3.4 Befehle des User Interface

1. **Verfügbarkeit User Interface:** Meldet, dass das User Interface zur Verfügung steht.
2. **Mapping-Vorgang:** Fordert das Starten oder Stoppen eines Mapping-Vorgangs.
3. **Veränderte Leuchtquelle:** Stellt die Informationen der ausgewählten Leuchtquelle zur Verfügung und fordert, dass diese Leuchtquelle verändert wird. Das User Interface unterstützt den Farbraum RGB, also Rot, Grün, Blau.
4. **Koordinaten einer Szenerie:** Wenn ein Bild auf den Leuchtquellen abgebildet werden soll, stellt das User Interface alle Koordinaten einer Beleuchtungs-Szenerie zur Verfügung.
5. **Koordinaten erhalten:** Enthält die Bestätigung, dass alle Koordinaten übertragen wurden.
6. **Bild:** Wenn ein Bild auf den Leuchtquellen abgebildet werden soll, wird das ausgewählte Bild zur Verfügung gestellt. Dies ist erst nach der Bestätigung „Koordinaten erhalten“ möglich.

4.3.5 Rückmeldung an das User-Interface

1. **Verfügbare Services:** Das User Interface erhält alle verfügbaren Services.
2. **Gemappte Koordinaten:** Nach dem Mapping werden dem User-Interface alle gemappten Leuchtquellen als Koordinaten zur Verfügung gestellt.
3. **Koordinaten mit Konfiguration:** Nachdem das Bild, welches auf die Leuchtquellen abgebildet werden soll zur Verfügung gestellt wurde, wird eine Liste aller Leuchtquellen und deren Farbe bereitgestellt.

4.4 Kamera-Service

4.4.1 Funktion

Der Kamera-Service wird für den Mapping-Vorgang benötigt. Der Service erstellt auf Befehl ein Bild und stellt dies an seiner Schnittstelle zur Verfügung.

4.4.2 Technologie

Der Kamera-Service ist in der Programmiersprache Java für Android-Geräte realisiert.

4.4.3 Aufbau des Kamera-Services

Der Kamera-Service wird nach dem Model-View-Controller (MVC) Pattern aufgebaut. Der Service benötigt keine Zwischenspeicherung von Daten und verwendet daher keine Model-Klassen.

View / Präsentationsschicht

In der Android-Applikation wird jede Ansicht von einer Activity implementiert. Der Kamera-Service beinhaltet folgende Activities:

- Einstellungen
- Kamera

Ansicht: Einstellungen

In der Ansicht „Einstellung“ kann eine Verbindung zu einem MQTT-Broker hergestellt werden. Das Eingabefeld für die IP-Adresse des MQTT-Brokers wird validiert und bei einer ungültigen IP-Adresse entsprechend ausgegeben. Nach einem erfolgreichen Verbindungsauftbau zum MQTT-Broker wird die Kamera-Ansicht gestartet.

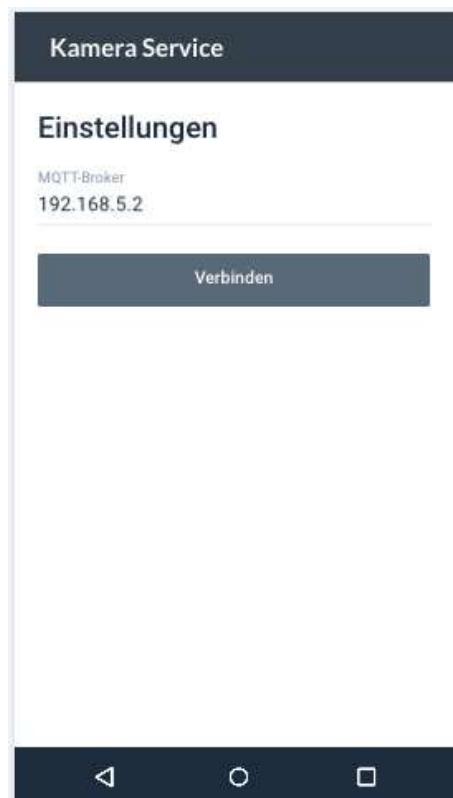


Abbildung 15- Kamera-Service: Einstellungs-Ansicht

Ansicht: Kamera

Die Kamera-Ansicht besteht zum Hauptteil aus einem Livestream, der von der internen Android-Kamera geliefert wird. Die Kamera wird auf eine Beleuchtungsanlage (im Abbild 16 unten ein LED-Strip) gehalten und ist somit einsatzbereit.

Die Topbar der Kamera-Ansicht beinhaltet einen grünen Punkt, wenn die Verbindung zum MQTT-Broker aufgebaut ist und einen roten Punkt, wenn keine Verbindung besteht. Das Icon für die Einstellungen führt in die Ansicht „Einstellungen“.



Abbildung 16 Kamera-Service: Kamera-Ansicht

4.4.4 Befehle an den Kamera-Service

1. **Foto schiessen:** Der Service erhält den Befehl ein Foto zu schiessen.

4.4.5 Rückmeldung des Kamera-Service

1. **Verfügbarkeit Kamera-Service:** Meldet, dass der Kamera-Service zur Verfügung steht.
2. **Foto:** Der Kamera-Service stellt das Foto zur Verfügung, welches er soeben geschossen hat.

4.5 Leuchtquellen-Service

Für den Leuchtquellen-Service wurden zwei Implementationen umgesetzt. Der Bildschirm-Service und der Led-Strip-Service. Sie repräsentieren zwei verschiedene Hardwarekonfigurationen. Der Bildschirm-Service stellt mehrere Leuchtquellen auf einem Bildschirm dar. Der Led-Strip-Service implementiert eine Lichterkette, die aus einzelnen LEDs besteht, welche diverse Farben annehmen können. Diese Lichterkette stammt von der Plattform Tinkerforge.

4.5.1 Bildschirm-Service

Der Bildschirm-Service ist eine konkrete Implementation für einen Leuchtquellen-Service. Er ist in Java geschrieben und implementiert das MVC-Pattern vollständig. Beim Starten des Services muss zuerst die IP-Adresse des gewünschten Brokers angegeben werden bevor der Service korrekt läuft. Ist dies getan, wechselt die View in den Vollbildmodus. Sie zeigt jetzt fünf Zeilen von jeweils acht Kreisen an. Die Kreise stellen die einzelnen Leuchtquellen dar. Der Bildschirm-Service kann die Farbe der Kreise genau anpassen.

4.5.1.1 View des Bildschirm-Service

Beim Starten des Services müssen zuerst diverse Einstellungen vorgenommen werden. Der Bildschirm-Service hat zwei Views. Eine wird beim Starten angezeigt und sieht ähnlich aus, wie die Start-Views der anderen Services. Nachdem aber auf den Start-Button geklickt wurde, erscheint eine weitere View. Sie enthält die einzelnen „Leuchtquellen“, dargestellt als Kreise. Diese View wird im Vollbildmodus geöffnet und kann mit der Escape-Taste wieder verlassen werden. In diesem Fall wird wieder die Start-View angezeigt.

Die Views des Bildschirm-Service sehen folgendermassen aus:

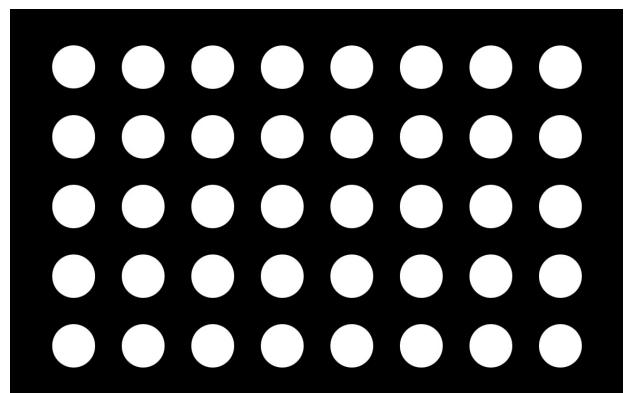
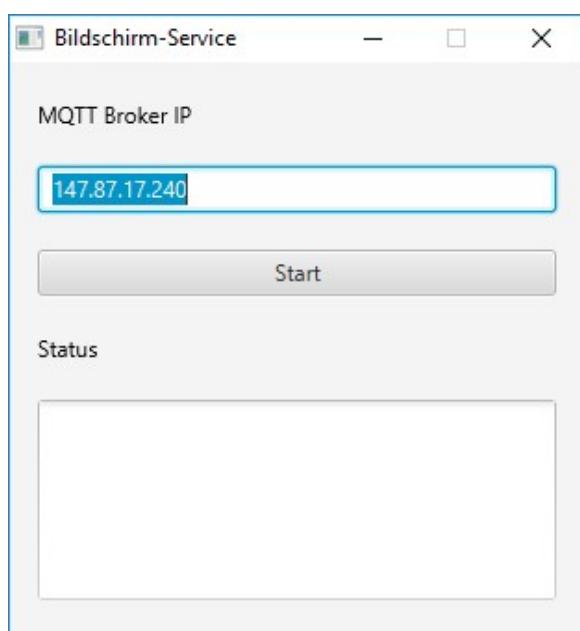


Abbildung 18 Bildschirm-Service: View mit einzelnen Leuchtquellen

Abbildung 17 Bildschirm-Service: Start-View

- **MQTT Broker IP:** IP-Adresse des gewünschten MQTT-Brokers.
- **Status:** Gibt Rückmeldung über den Zustand des Services. Zum Beispiel, ob eine Verbindung zum Broker besteht.

4.5.1.2 Befehle an den Bildschirm-Service

Der Bildschirm-Service kann folgende Befehle entgegennehmen.

1. **Initialstatus einnehmen:** Alle Leuchtquellen werden ausgeschaltet. Dies wird benötigt um ein Referenzfoto zu schiessen.
2. **Nächste Leuchtquelle einschalten:** Dieser Befehl enthält zusätzlich eine Leuchtquellennummer. Die Leuchtquelle mit dieser Nummer soll nun eingeschaltet werden. Diese Funktion wird während des Mapping-Vorgangs benötigt.
3. **Leuchtquelle ändern:** Dieser Befehl wird gegeben wenn die Konfiguration von einer oder mehreren Leuchtquellen geändert wurden. Das kann zum Beispiel sein, weil eine einzelne Leuchtquelle die Farbe gewechselt hat, oder wenn eine ganze, neue Szene dargestellt werden soll. Der Bildschirm-Service passt seine Leuchtquellen entsprechend an.

4.5.1.3 Rückmeldungen des Bildschirm-Service

Der Bildschirm-Service kann folgende Rückmeldungen geben:

1. **Nächste Leuchtquelle eingeschaltet:** Wenn die nächste Leuchtquelle eingeschaltet wurde, gibt der Bildschirm Service eine Bestätigungs Nachricht zurück.
2. **Service verfügbar:** Meldet, dass der Bildschirm-Service zur Verfügung steht.

4.5.2 Funktionsweise des Led-Strip-Service

Der Led-Strip-Service ist eine konkrete Implementation für einen Leuchtquellen-Service. Er ist in Java geschrieben und implementiert das MVC-Pattern vollständig. Der Led-Strip-Service ist für die Kommunikation mit dem LED-Strip von Tinkerforge verantwortlich. Er übersetzt dazu die Befehle, die an seiner Schnittstelle zum Agent anliegen in eine, für die Tinkerforge-Komponenten verständliche Form. Der Led-Strip-Service unterstützt den RGB-Farbraum.

4.5.2.1 Funktionsweise von Tinkerforge

Tinkerforge bietet Mikrocontroller für die verschiedensten Anwendungen an. So auch für das Ansteuern einer Lichterkette, die aus LEDs besteht. Diese LEDs stammen ebenfalls von Tinkerforge und können unter anderem die Farbe wechseln. Die verschiedenen Mikrocontroller werden Bricklets genannt. Jeder Bricklet ist auf eine definierte Anwendung spezialisiert. Für die Ansteuerung eines Led-Strips wird ein Led-Strip-Bricklet benötigt. Die einzelnen Bricklets sind nicht in der Lage direkt mit einem Anwenderprogramm zu kommunizieren. Dazu wird ein weiterer Mikrocontroller benötigt. Tinkerforge nennt diese Master-Bricks. Die Bricklets werden direkt mit einem Master-Brick verbunden. Dieser wiederum kann zum Beispiel über einen USB-Port mit einem Personal Computer kommunizieren. Die Kommunikation zwischen dem Master-Brick und dem Computer via USB wird von einem Deamon (ein Service) gesteuert. Dieser Deamon wird von Tinkerforge angeboten und heisst Brick-Deamon [7]. Der Brick-Deamon muss auf dem System installiert sein, wenn die Kommunikation zwischen dem Led-Strip-Service und den Tinkerforge-Komponenten funktionieren soll.

4.5.2.2 View des Led-Strip-Service

Beim Starten des Services müssen zuerst diverse Einstellungen vorgenommen werden. Die View des Led-Strip-Service sieht folgendermassen aus:

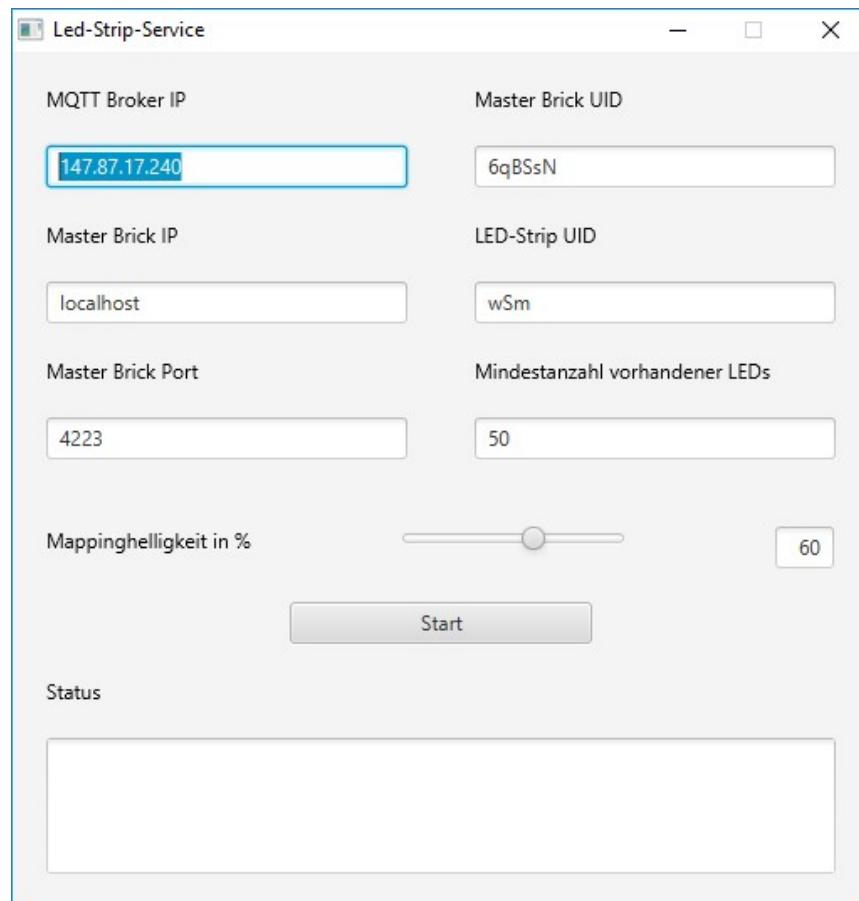


Abbildung 19 Led-Strip-Service: Grafische Benutzerschnittstelle

- **MQTT Broker IP:** IP-Adresse des gewünschten MQTT-Brokers.
- **Master Brick IP:** IP-Adresse des gewünschten Master-Bricks.
- **Master Brick Port:** Port, über welcher die Kommunikation mit dem Master-Brick laufen soll.
- **Master Brick UID:** Die UID wird von Tinkerforge für jeden Master-Brick individuell vergeben.
- **LED-Strip UID:** Die UID für das Bricklet (hier LED-Strip-Bricklet) wird von Tinkerforge für jedes Bricklet individuell vergeben.
- **Mindestanzahl vorhandener LEDs:** Anzahl von LEDs, die mindestens vorhanden sind. Es können auch mehr LEDs angegeben werden. Dann kann das Mappen aber unter Umständen länger dauern. Werden weniger angegeben, funktioniert das Mapping nicht ordnungsgemäß.
- **Mapping-Helligkeit in %:** Helligkeit der einzelnen LEDs während des Mapping-Vorgangs in %.
- **Status:** Gibt Rückmeldung über den Zustand des Services. Zum Beispiel, ob eine Verbindung zum Broker und den Tinkerforge-Komponenten besteht.

4.5.2.3 Befehle an den LED-Strip-Service

Der LED-Strip-Service kann folgende Befehle entgegennehmen.

1. **Initialstatus einnehmen:** Alle Leuchtquellen werden ausgeschaltet. Dies wird benötigt um ein Referenzfoto zu schiessen.
2. **Nächste LED einschalten:** Dieser Befehl enthält zusätzlich eine LED-Nummer. Die LED mit dieser Identifikations-Nummer soll nun eingeschaltet werden. Diese Funktion wird während des Mapping-Vorgangs benötigt.
3. **Leuchtquelle ändern:** Dieser Befehl wird gegeben wenn eine oder mehrere Leuchtquellen geändert wurden. Das kann zum Beispiel sein, weil eine einzelne LED die Farbe gewechselt hat, oder wenn eine ganze Szenerie dargestellt werden soll. Der Led-Strip-Service passt seine Leuchtquellen entsprechend an.

4.5.2.4 Rückmeldungen des LED-Strip-Service

Der LED-Strip-Service kann folgende Rückmeldungen geben:

1. **Nächste LED eingeschaltet:** Wenn die nächste LED eingeschaltet wurde, gibt der LED-Strip-Service eine Bestätigungs Nachricht zurück.
2. **Service verfügbar:** Meldet, ob der LED-Strip-Service zur Verfügung steht.

4.6 Datenverarbeitungs-Service

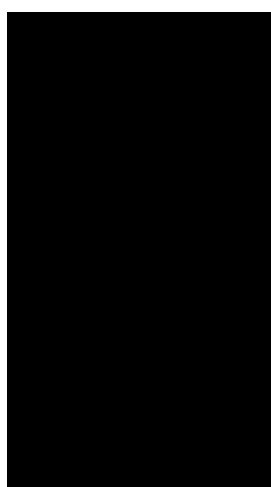
4.6.1 Funktionsweise des Datenverarbeitungs-Service

Der Datenverarbeitungs-Service ist eine konkrete Implementation. Er ist in Java geschrieben und implementiert das MVC-Pattern vollständig. Der Datenverarbeitungs-Service vergleicht jeweils zwei Fotos und überprüft, ob sich die Helligkeit geändert hat. Ist dies der Fall, hat er eine neue Leuchtquelle gefunden. Die Bilder müssen jeweils in Form eines Byte-Arrays an den entsprechenden Schnittstellen anliegen. Der Vergleich wird mit der OpenCV-Bibliothek erreicht. Diese stellt eine Vielzahl von verschiedenen Bildbearbeitungsfunktionen bereit.

4.6.1.1 Mapping im Idealfall

Der Datenverarbeitungs-Service erhält jeweils als erstes ein Referenzfoto. Auf diesem ist keine Leuchtquelle eingeschaltet. Dies wird mit der Hilfe der OpenCV-Bibliothek in ein reines Schwarz-/Weiss-Foto umgewandelt. Jeder Pixel des so bearbeiteten Fotos ist also entweder vollständig Schwarz oder vollständig Weiss. Der Umwandlungsmethode muss neben dem Foto ein Threshold-Wert mitgegeben werden. Dieser Wert bestimmt wo die Grenze zwischen schwarzen und weissen Pixel gesetzt werden soll. Sämtliche Pixelwerte die unter diesem Schwellwert sind, werden zu komplett schwarzen Pixel (Wert 0), alle anderen zu komplett weissen Pixel (Wert 255). Das so bearbeitete Foto wird anschliessend in einer Membervariable vorgemerkt. Sobald ein zweites Foto an der entsprechenden Schnittstelle bereitsteht, findet die eigentliche Datenverarbeitung statt. Das zweite Foto wurde geschossen, nachdem eine bestimmte Leuchtquelle eingeschaltet wurde. Dieses Foto ist das Vergleichsfoto.

Als erstes wird von diesem Vergleichsfoto auf die gleiche Art wie schon beim ersten Foto, ein Schwarz-/Weiss-Foto erstellt. Das zweite Foto wird auf Weisskonturen untersucht. Auch dafür bietet die OpenCV-Bibliothek eine Methode. Im Idealfall hat das Vergleichsfoto genau eine Weisskontur, nämlich dort, wo sich die jetzt eingeschaltete Leuchtquelle befindet. Der Datenverarbeitungs-Service errechnet aus den X- und Y-Koordinaten dieser Kontur das ungefähre Zentrum der Leuchtquelle. Anschliessend wird an genau der Koordinate des Zentrums, den Pixelwert auf dem Referenzbild ausgelesen. Ist dieser schwarz, wurde die Leuchtquelle gefunden. Das Zentrum dient in diesem Fall als Koordinate der Leuchtquelle und wird vom Datenverarbeitungs-Service an den Agent zurückgegeben. Diese Rückgabe dient zugleich als Bestätigung, dass das Finden der Leuchtquelle erfolgreich war.



Referenzfoto ohne Weisskontur.



Vergleichsfoto mit einer einzelnen Weisskontur an genau der Stelle, an der sich die Leuchtquelle befinden muss.

Abbildung 20 Referenzfoto

Abbildung 21 Vergleichsfoto

4.6.1.2 Mapping mit Weisskontur auf dem Referenzfoto

Sollte das Referenzfoto bereits eine Weisskontur enthalten, ist dies höchst wahrscheinlich aufgrund der vorliegenden Lichtverhältnisse der Fall. Das ist nicht ganz so optimal, kann aber bis zu einem gewissen Grad korrigiert werden. In diesem Fall haben die beiden Fotos an der gleichen Stelle, jeweils eine ähnliche Weisskontur. Beim überprüfen des Pixelwertes des Zentrums auf dem Referenzfoto wird festgestellt, dass beide Pixelwerte weiss sind. Erwartet würde aber, dass der Pixelwert des Vergleichsfotos weiss ist, derjenige vom Referenzfoto schwarz. In diesem Fall sucht der Datenverarbeitungs-Service auf dem Vergleichsfoto nach weiteren Konturen. Existieren solche, wird wieder der Pixelwert des Zentrums auf dem Referenzfoto verglichen. Solange sich die Leuchtquelle nicht in der unerwünschten Kontur, welche auch auf dem Referenzbild zu sehen ist, befindet, wird die Leuchtquelle trotzdem gefunden. Befindet sie sich jedoch innerhalb der Kontur, wird sie von den vorherrschenden Lichtverhältnissen überstrahlt und es existiert keine Möglichkeit aufgrund dieser beiden Bilder eine Leuchtquelle ausfindig zu machen. Der Datenverarbeitungs-Service schliesst daraus, dass keine weitere Leuchtquelle existiert und beendet den Auswertungsvorgang. In diesem Fall, kann durch das Korrigieren des Thresholds und / oder der Leuchtquellenhelligkeit vielleicht eine Verbesserung erzielt werden. Das Mapping muss aber mit diesen veränderten Werten neu gestartet werden.



Referenzbild mit mehreren Weisskonturen.



Vergleichsbild mit ähnlichen Weisskonturen wie auf dem Referenzbild, aber mit einer weiteren Weisskontur im unteren Bildbereich, welche auf dem Referenzbild nicht vorhanden ist. Dabei muss es sich um die gesuchte Leuchtquelle handeln.

Abbildung 22 Referenzbild mit Weisskonturen

Abbildung 23 Vergleichsbild mit Weisskonturen

Wäre die Leuchtquelle im obigen Bild in einer der Weisskonturen, würde sie nicht gefunden, da auf dem Regerenzbild am gleichen Ort auch eine Weisskontur existiert. Der Datenverarbeitungs-Service hätte keine Chance, die Leuchtquelle ausfindig zu machen.

4.6.1.3 Beenden des Mapping-Vorgangs

Der Datenverarbeitungs-Service entscheidet wann ein Mapping-Vorgang beendet wird. Dies kann entweder sein, weil schlechte Lichtverhältnisse bestehen, oder weil der Mapping-Vorgang regulär zu Ende gelaufen ist. In beiden Fällen kann der Datenverarbeitungs-Service zwischen dem Referenzfoto und dem Vergleichsfoto kein Unterschied mehr feststellen. Entweder sind beide Fotos gleich, weil keine weitere Leuchtquelle mehr eingeschaltet werden konnte, oder weil die Leuchtquelle von einer Weisskontur überstrahlt wird. Weil es aber vorkommen kann, dass ein Vergleichsfoto nur temporär ungünstige Lichtverhältnisse hat, wird der Abbruch nicht gleich nach dem ersten fehlgeschlagenen Vergleich eingeleitet. Zuerst wird ein weiteres Vergleichsfoto angefordert. Diesen Vorgang wiederholt der Datenverarbeitungs-Service insgesamt fünf Mal. Erst wenn dann immer noch keine Veränderung festgestellt werden kann, wird der Mapping-Vorgang beendet. In diesem Fall meldet der Datenverarbeitungs-Service an seiner Schnittstelle das Ende des Mapping-Vorgangs.

4.6.2 View des Datenverarbeitungs-Service

Beim Starten des Services müssen zuerst diverse Einstellungen vorgenommen werden. Die View des Datenverarbeitungs-Service sieht folgendermassen aus:



Abbildung 24 Datenverarbeitungs-Service: Grafische Benutzerschnittstelle

- **MQTT Broker IP:** IP-Adresse des gewünschten MQTT-Brokers.
- **Schwarz/Weiss Schwellwert (0 – 255):** Schwellwert für die Zuweisung ob ein Pixel als Schwarz oder Weiss gilt.
- **Status:** Gibt Rückmeldung über den Zustand des Services. Zum Beispiel, ob eine Verbindung zum Broker besteht.

4.6.3 Befehle an den Datenverarbeitungs-Service

Der Datenverarbeitungs-Service kann folgende Befehle erhalten:

1. **Referenzfoto:** Liegt an der Schnittstelle ein Referenzfoto an, wird dieses in ein Schwarz-/Weiss-Foto umgewandelt und in einer Membervariable zwischengespeichert.
2. **Vergleichsfoto:** Liegt an der Schnittstelle ein Vergleichsfoto an, wird ein Vergleich der beiden Fotos durchgeführt.

4.6.4 Rückmeldungen des Datenverarbeitungs-Service

Der Datenverarbeitungs-Service kann folgende Rückmeldungen geben:

1. **Koordinate:** Koordinate der Leuchtquelle vom letzten Vergleich.
2. **Status:** Meldet ob das Referenzfoto erfolgreich abgespeichert wurde, ob ein Wiederholen des Vergleichsfotos nötig ist oder ob der Vergleichsvorgang abgebrochen wurde.
3. **Service verfügbar:** Meldet, ob der Datenverarbeitungs-Service zur Verfügung steht

4.7 Bildumwandlungs-Service

4.7.1 Funktionsweise des Bildumwandlungs-Service

Der Bildumwandlungs-Service ist eine konkrete Implementation. Er ist in Java geschrieben und implementiert vom MVC-Pattern nur die Präsentations- (View) und die Steuerungsschicht (Controller). Der Bildumwandlungs-Service wandelt ein Bild so um, dass es möglichst gut auf das gewünschte Mapping passt. Die Bearbeitung der Bilder erfolgt mit der OpenCV-Bibliothek. Die unterstützten Formate laut der offiziellen Website von OpenCV sind:

- *Windows bitmaps - *.bmp, *.dib (always supported)*
- *JPEG files - *.jpeg, *.jpg, *.jpe (see the Notes section)*
- *JPEG 2000 files - *.jp2 (see the Notes section)*
- *Portable Network Graphics - *.png (see the Notes section)*
- *WebP - *.webp (see the Notes section)*
- *Portable image format - *.pbm, *.pgm, *.ppm *.pxm, *.pnm (always supported)*
- *Sun rasters - *.sr, *.ras (always supported)*
- *TIFF files - *.tiff, *.tif (see the Notes section)*
- *OpenEXR Image files - *.exr (see the Notes section)*
- *Radiance HDR - *.hdr, *.pic (always supported)*
- *Raster and Vector geospatial data supported by Gdal (see the Notes section)*

[8]

Der Bildumwandlungs-Service erhält jeweils als erstes alle Koordinaten eines konkreten Mappings. Diese speichert er in einer Membervariable ab und gibt an seiner Schnittstelle eine Bestätigung aus, dass er die Koordinaten erhalten hat. Danach ist er bereit ein Bild zu empfangen. Dieses muss in Form eines Byte-Arrays an der entsprechenden Schnittstelle anliegen. Sobald dies der Fall ist, beginnt der Bildumwandlungs-Service mit dem Konvertieren. Dazu errechnet er aus den Koordinaten die Länge und Breite des virtuellen Abbildes der Beleuchtungsanlage in Pixel. Diese ist um einen bestimmten Faktor anders (meist kleiner) als die Länge und Breite des Bildes, welches auf die Beleuchtungsanlage umgewandelt werden soll. Entsprechend diesem Faktor weist er anschliessend jeder Leuchtkquelle der Beleuchtungsanlage einen Pixel im Bild zu. Er liest die Farbwerte des entsprechenden Pixels aus und weist die Werte der Leuchtkquelle zu. Dies errechnet er für alle vorhandenen Leuchtketten und speichert jeweils die Farbwerte mit der Identifikationsnummer der Leuchtketten in einer Liste ab. Sind alle Leuchtketten zugewiesen, stellt er diese Liste an seiner Schnittstelle zur Verfügung.

4.7.2 View des Bildumwandlungs-Service

Beim Starten des Services müssen zuerst diverse Einstellungen vorgenommen werden. Die View des Bildumwandlungs-Service sieht folgendermassen aus:

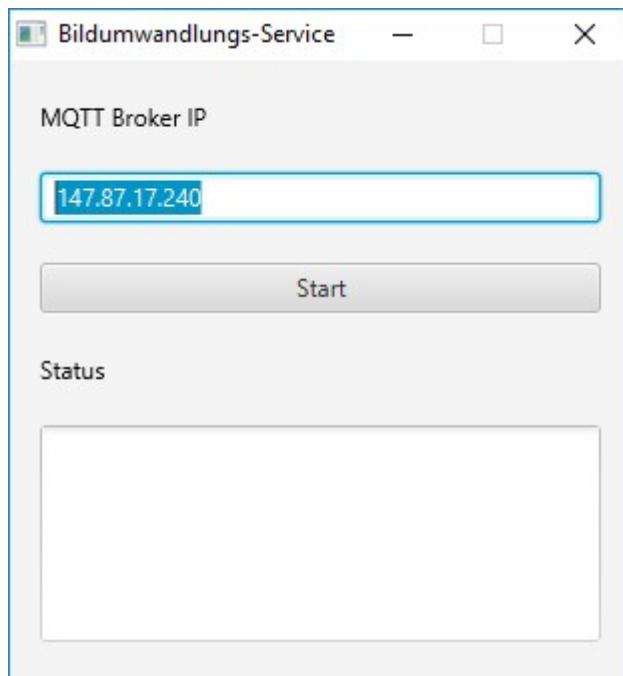


Abbildung 25 Bildumwandlungs-Service: Grafische Benutzeroberfläche

- **MQTT Broker IP:** IP-Adresse des gewünschten MQTT-Brokers.
- **Status:** Gibt Rückmeldung über den Zustand des Services. Zum Beispiel, ob eine Verbindung zum Broker besteht.

4.7.3 Befehle an den Bildumwandlungs-Service

Der Bildumwandlungs-Service kann folgende Befehle erhalten:

1. **Koordinaten:** Ist eine Liste von Koordinaten der Beleuchtungsanlage. Sie muss vom Bildumwandlungs-Service in einer gespeichert werden und wird nach dem Erhalten des Bildes für das Umwandeln benötigt.
2. **Bild:** Enthält das zu konvertierende Bild. Löst das Umwandeln auf die Szenerie aus.

4.7.4 Rückmeldungen des Bildumwandlungs-Service

Der Bildumwandlungs-Service kann folgende Rückmeldungen geben:

1. **Koordinaten erhalten:** Gibt die Bestätigung zurück, dass die Koordinaten erhalten wurden.
2. **Umgewandeltes Bild:** Gibt das umgewandelte Bild in Form einer Liste mit Koordinaten und den zugehörigen Farbwerten zurück.
3. **Service verfügbar:** Meldet, ob der Bildumwandlungs-Service zur Verfügung steht.

4.8 Agent

Der Agent ist eigentlich kein Service. Er ist in Java geschrieben und implementiert vom MVC-Pattern nur die Präsentations- (View) und die Steuerungsschicht (Controller). Der Agent ist für das korrekte Funktionieren des gesamten LED-Mapper zuständig. Er besitzt selber keine eigene Schnittstelle, sondern verbindet sich jeweils mit der Schnittstelle aller Services. Wird ein Service geschrieben der keine Funktionalität ersetzt sondern eine neue einführt, muss der Agent zwingend auch angepasst werden. Dies weil ein neuer Service automatisch auch eine neue Schnittstelle mit sich bringt, welche dem Agent nicht bekannt ist.

Da der Agent keine eigene Schnittstelle besitzt, muss er sich mit den Schnittstellen der Services verbinden. Auf jedes Topic, auf welches ein Service veröffentlicht (published), muss sich der Agent anmelden (subscribe). So erhält er sämtliche Mitteilungen aller Services. Umgekehrt veröffentlicht der Agent auf allen Topics, auf welche sich irgendein Service angemeldet hat. Dadurch entsteht zwischen dem Agent und dem jeweiligen Service eine bidirektionale Kommunikation. Jeder erhält alle Nachrichten des anderen. Die genaue Struktur der Kommunikation ist im Kapitel Kommunikation 3.5 beschrieben.

Bei jeder Nachricht, die ein Service an seiner Schnittstelle bereitstellt, muss der Agent entscheiden was zu tun ist. Fordert zum Beispiel die Benutzerschnittstelle, dass eine bestimmte LED eingeschaltet wird, nimmt der Agent das entgegen und leitet den Befehl an den entsprechenden Leuchtquellen-Service weiter. Etwas komplizierter ist der Ablauf wenn das User Interface ein Mapping-Befehl gibt. In diesem Falle muss der Agent als erstes dem Leuchtquellen-Service den Befehl geben, alle Leuchtquellen auszuschalten, also den Initialstatus einzunehmen. Dies wird dem Agent vom Leuchtquellen-Service bestätigt. Hat er diese Bestätigung, befiehlt er dem Kamera-Service ein Foto zu schiessen. Der Service liefert ihm das Foto und der Agent muss dies an den Datenverarbeitungs-Service als Initialbild weiterleiten. Anschliessen meldet der Agent dem Leuchtquellen-Service, dass er die erste Leuchtquelle einschalten soll. Danach steht das Schiessen des nächsten Fotos an. Jede Koordinate, die vom Datenverarbeitungs-Service gefunden wird, speichert der Agent in einer Liste ab. Erst wenn der Mapping-Vorgang beendet ist, wird das User Interface wieder involviert. Das User Interface erhält dann die Liste aller Koordinaten mit den zugehörigen Leuchtquellennummern.

4.8.1 View des Agent

Beim Starten des Agent müssen zuerst diverse Einstellungen vorgenommen werden. Die View sieht folgendermassen aus:

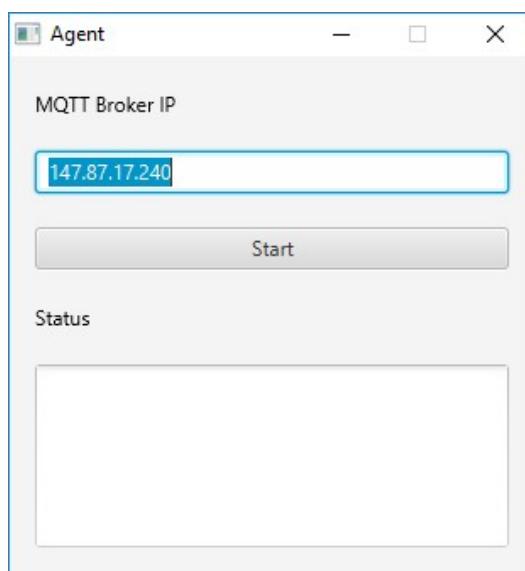


Abbildung 26 Agent: Grafische Benutzerschnittstelle

- **MQTT Broker IP:** IP-Adresse des gewünschten MQTT-Brokers.
- **Status:** Gibt Rückmeldung über den Zustand des Services. Zum Beispiel, ob eine Verbindung zum Broker besteht.

4.9 Kommunikationsaufbau (MQTT)

4.9.1 MQTT

Das IoT-Protokoll MQTT (Message Queue Telemetry Transport) basiert auf dem publish/subscribe-Prinzip. Ein Client kann sich mit einem Broker verbinden und anschliessend über diesen Nachrichten veröffentlichen (publishen). Dazu muss der Client ein bestimmtes Topic angeben, zum Beispiel „Kommunikation/Nachricht“. Meldet sich jetzt ein anderer Client an diesem Topic an (subscribe), erhält dieser sämtliche Nachrichten, die vom ersten Client unter genau diesem Topic veröffentlicht wurden. Natürlich können unter dem gleichen Topic beliebig viele Clients Nachrichten veröffentlichen und/oder empfangen. Auch kann ein Client auf einem Topic Nachrichten veröffentlichen und sich gleichzeitig mit diesem Topic verbinden, so dass auch er seine eigenen Nachrichten erhält. Das MQTT-Protokoll unterstützt sowohl die Kommunikation über TCP wie auch über TLS. Der LED-Mapper unterstützt nur die Kommunikation über TCP. Für MQTT ist der Port 1883 reserviert. [9]

4.9.2 Topics als Schnittstelle nutzen

Der LED-Mapper nutzt die Möglichkeit, Topics so zu strukturieren, dass sich alle Services am gleichen Broker anmelden, aber jeder Service eine eigene Topic-Struktur hat. Dies führt dazu, dass jeder Service nur die Nachrichten erhält, die ihn betreffen. Die Topic-Struktur dient also als Schnittstelle. Wie die Topic-Struktur für den LED-Mapper genau auszusehen hat, ist durch seine Architektur bestimmt. In der Praxis könnte diese auch anders aussehen. Die einzelnen Services könnten sogar direkt miteinander kommunizieren, indem sich ein Service direkt bei einem anderen Service anmeldet. Dies würde aber ein Verstoss gegen die Architektur bedeuten. Das Interfacediagramm des LED-Mapper sieht folgendermassen aus:

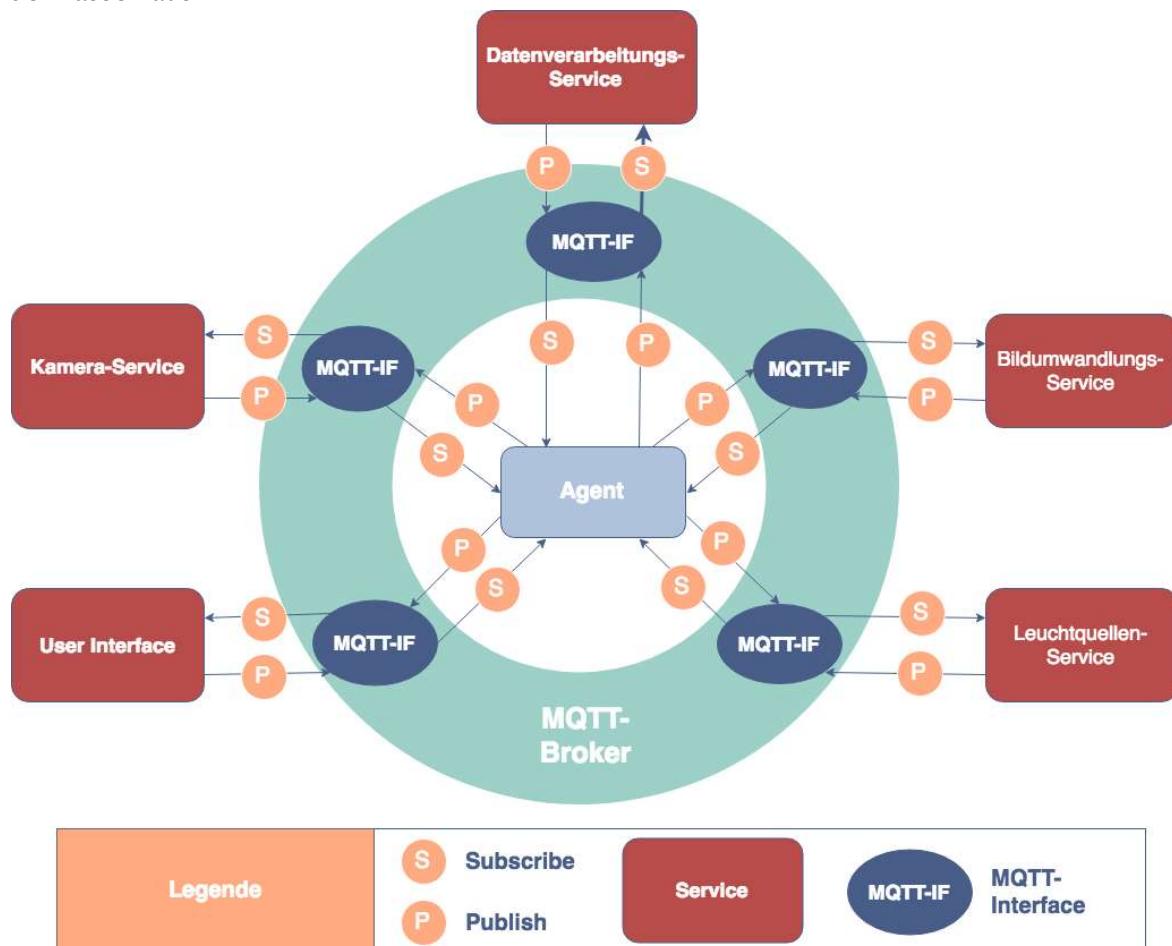


Abbildung 27 Kommunikationsübersicht der Services

4.9.3 Allgemeine Topic-Struktur

Das gesamte Topic, auf welchem bestimmte Nachrichten veröffentlicht (published) werden, besteht aus insgesamt vier Teilen. Der erste Teil (Topic-Head) bestimmt immer, dass es sich um das Topic des LED-Mapper handelt „ch/bfh/bachelorthesis/ledmapper/“. Der zweite Teil (Service-Topic) sagt aus, um welchen Service es sich handelt, zum Beispiel „camera_service/“. Der dritte Teil (Service In- und Output) sagt aus, ob es sich um ein Input- oder Output-Topic handelt „input/“ beziehungsweise „output/“. Dies geschieht immer aus der Sicht des Service. Input bedeutet also, der Service hat sich an diesem Topic angemeldet (subscribed). Der Service nimmt von dort seine Befehle oder Daten entgegen. Output bedeutet, der Service veröffentlicht auf dieses Topic. Er kann dort zum Beispiel sein Status oder Daten bereitstellen. Für den Kamera-Service heissen die Topics für den Input beispielsweise „do_picture/“, für den Output „picture/“ oder „service_available/“. Sie stellen den vierten Teil (Befehle) des gesamten Topics dar. Folgende Grafik soll Die Topic-Struktur am Beispiel des Kamera-Service veranschaulichen:

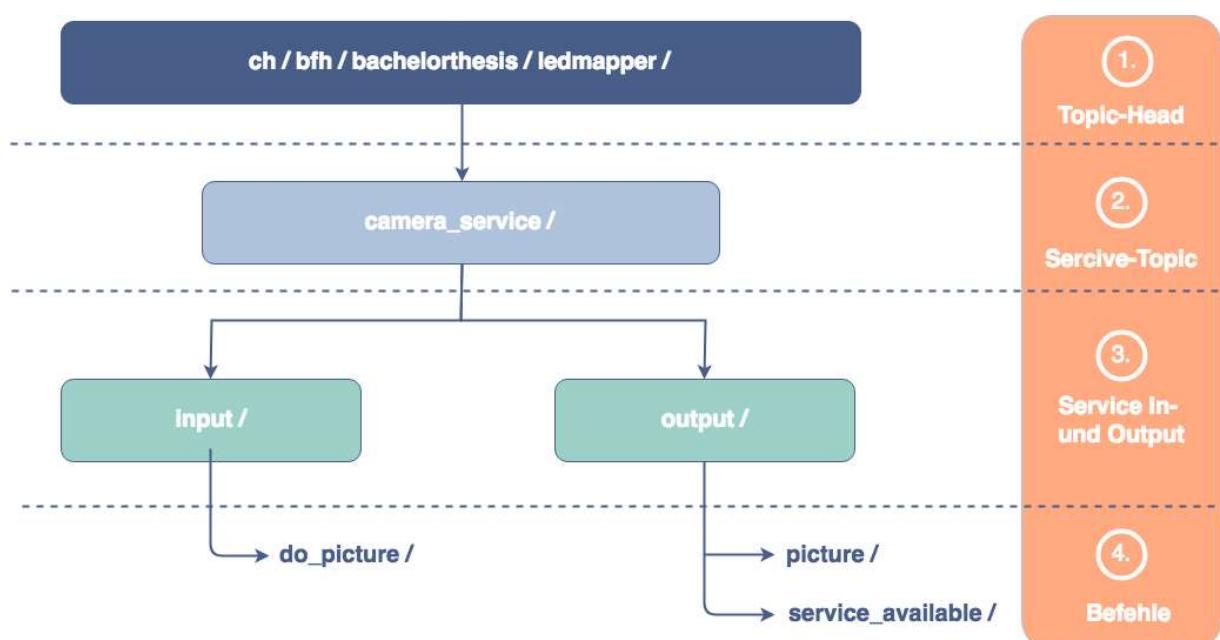


Abbildung 28 Allgemeine Topic-Struktur

4.9.4 Gesamte Topic-Struktur des LED-Mapper

Auf allen Topics können Nachrichten veröffentlicht werden. Ob in einem konkreten Topic nur eine oder mehrere Nachrichten vorhanden sein können, ist von Topic zu Topic verschieden. Die nachfolgende Tabelle enthält die gesamte Topic-Struktur mit allen Topics des LED-Mappers. Zusätzlich werden die zugehörigen Nachrichten, die über das jeweilige Topic laufen und vom LED-Mapper erkannt werden, aufgeführt. Es gilt dabei zu beachten, dass die Darstellung in der Tabelle erst ab dem zweiten Teil (Service-Topic) beginnt. Der erste Teil (ch/bfh/bachelorthesis/ledmapper/) muss in der Praxis vor den Pfad in der Tabelle gesetzt werden.

Service	Pfad (zweiter Teil/ dritter Teil/ vierter Teil)	Nachricht
User Interface	ui/input/coordinates/	List<String> von Koordinaten: [uid;X-Coordinate;Y-Coordinate;Radius; X-Resolution;Y-Resolution]
	ui/input/available_services/	List<String> von allen verfügbaren Services: [Typ;Name;ID, ...]
	ui/input/coordinates_received/	„done“
	ui/input/converted_file/	List<String> von Leuchtquellen: [luminaireId;color]
	ui/output/mapping/	„startMapping“ „stopMapping“
	ui/output/luminaire_changed/	List<String> von Leuchtquellen: [uid;color;brightness;on]
	ui/output/converting_coordinates/	List<String> von Koordinaten: [luminaireId;X-Coordinate;Y-Coordinate]
	ui/output/converting_file/	Byte-Array
Kamera-Service	camera/input/do_picture/	„doPicture“
	camera/output/picture/	Byte-Array
	camera/output/service_available/	„camerasHere“ „camerasGone“
Leuchtquellen-Service	luminaire/input/mapping/	„dolnitialState“ „next“ + aktuelle LED-Nummer
	luminaire/input/luminaire_changed/	List<String> von Leuchtquellen: [uid;color;brightness;on]
	luminaire/output/mapping/	„done“
	luminaire/output/service_available/	„luminaireIsHere“ „luminaireIsGone“

Datenverarbeitungs-Service	data_processing/input/reference_picture/	Byte-Array
	data_processing/input/comparison_picture/	Byte-Array
	data_processing/output/coordinate/	Koordinate: X-Coordinate;Y-Coordinate;Radius;X-Resolution;Y-Resolution
	data_processing/output/status/	„referencePictureDone“ „redoPicture“ „mappingDone“
	data_processing/output/service_available/	„dataProcessingIsHere“ „dataProcessingIsGone“
Bildumwandlungs-Service	picture_converting/input/converting_file/	Byte-Array
	picture_converting/input/converting_coordinates/	List<String> von Koordinaten: [luminaireId;X-Coordinate;Y-Coordinate]
	picture_converting/output/coordinates_received/	„done“
	picture_converting/output/converted_file/	List<String> von Leuchtquellen: [luminaireId;color]
	picture_converting/output/service_available/	„pictureConvertingIsHere“ „pictureConvertingIsGone“

Messages unter dem Topic „service_available/“ werden als sogenannte „Retained Message“ versandt. Das bedeutet, der Broker speichert jeweils die letzte erhaltene Nachricht und leitet diese auch einem Client weiter, der sich erst nach dem Veröffentlichen (publish) der Nachricht anmeldet. Dies ermöglicht dem Agent auch dann zu wissen, welche Services im Moment zur Verfügung stehen und welche nicht, wenn er erst nach einem Service gestartet wird. Genau gleich verhält es sich mit dem UI-Topic, „ui/input/available_services/“. Auch da erhält das UI die Nachricht, welche Services verfügbar sind, auch dann, wenn es später gestartet wird als der Agent.

4.9.5 Nutzung der Topics durch den Agent

Da der Agent selber keine Topics erstellt, verbindet er sich mit den Schnittstellen jeweils genau in umgekehrter Weise, wie dies die Services tun. Während ein Service auf sein Output-Topic Nachrichten veröffentlicht (publish), muss sich der Agent auf genau dem Topic anmelden (subscribe). Umgekehrt veröffentlicht (publish) der Agent seine Nachrichten auf einem Topic, auf welches sich ein Service angemeldet (subscribe) hat.

4.10 Mapping-Vorgang

Während dem Mapping-Vorgang werden viele Nachrichten hin und her geschickt, ohne dass der Benutzer etwas dazu beitragen muss. Dieser relativ komplexe Vorgang wird im nachfolgenden Diagramm verdeutlicht. Das Diagramm zeigt alle Messages, die während eines Mapping-Vorgangs zwischen dem Agent und den einzelnen Services ausgetauscht werden.

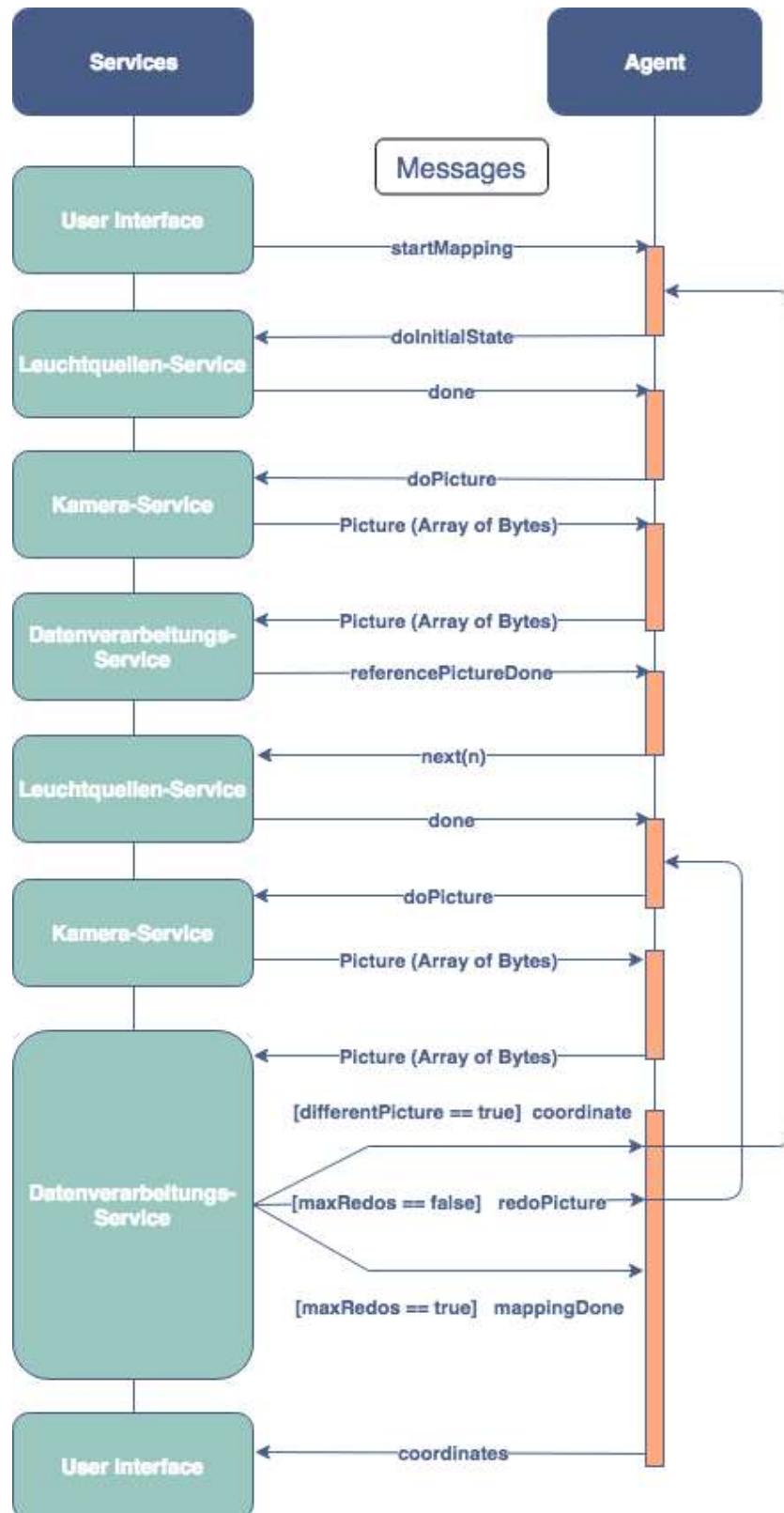


Abbildung 29 MQTT-Message-Ablauf vom Mapping-Vorgang

4.11 Hardwarekonfiguration des Systems

Durch die Aufteilung des Systems in verschiedene Services, existieren viele mögliche Hardwarekonfigurationen. Zum Beispiel können alle Services auf dem physikalisch gleichen Rechner laufen, oder einzelne Services laufen auf einem Server, der irgendwo sonst steht. Einige Einschränkungen existieren jedoch. So muss der Led-Strip-Service physikalisch am gleichen Ort laufen, wo sich auch der LED-Strip befindet. Dies kommt daher, dass der Master Brick von Tinkerforge mit einem USB-Kabel an den Computer angeschlossen wird. Auf diesem Computer muss sich dann auch der Led-Strip-Service befinden. Folgendes Bild zeigt einen möglichen Hardwareaufbau des LED-Mapper:

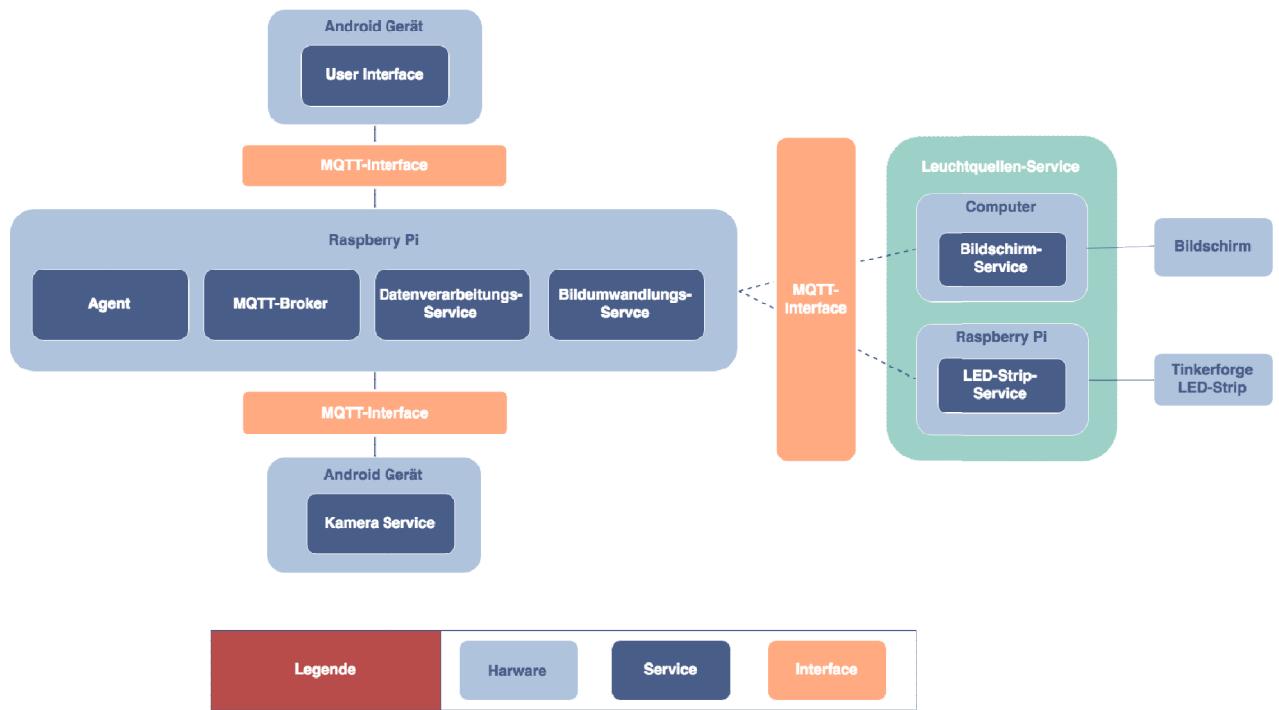


Abbildung 30 Hardware-Konfiguration der Realisierung

4.12 Anleitung für die Inbetriebnahme

4.12.1 Anforderungen

Mobile Applikationen <ul style="list-style-type: none">• User Interface• Kamera-Service	Um die mobilen Applikationen auszuführen wird ein Android-Gerät verwendet. Die native Mobile-App setzt eine Android Software Development Kit (SDK) Version 21 oder höher voraus. Die SDK Version 21 wird für die Version Android 5.0 „Lollipop“ verwendet.
Standardsoftware <ul style="list-style-type: none">• Agent• LED-Strip-Service• Bildschirm-Service• Datenverarbeitungs-Service• Bildumwandlungs-Service	<p>Die Ausführung der Standardsoftware setzt einen plattformunabhängigen Computer und einen Bildschirm sowie eine Java-Version 8 Installation voraus.</p> <p>OpenCV Die Services für die Datenverarbeitung und die Bildumwandlung benötigen die Native Bibliothek von OpenCV.</p> <p>Brick-Deamon Damit der LED-Strip-Service mit dem Master-Brick von Tinkerforge kommunizieren kann, wird ein Brick-Deamon benötigt.</p>
MQTT-Broker	Die Inbetriebnahme eines MQTT-Brokers setzt eine Installation eines Brokers wie beispielsweise Mosquitto voraus. Der MQTT-Broker muss auf einem Computer installiert werden, der eine Netzwerkanbindung hat. Der MQTT-Broker muss sich im gleichen Netzwerk wie die Services befinden. Zweite Option ist, dass der MQTT-Broker auf einem öffentlich erreichbaren Server läuft und die Services eine Internetverbindung haben.

4.12.2 Download und Installation

Sämtliche Downloads sind unter folgendem Link zu finden: <https://github.com/Haempu/LED-Confusion/Bachelorthesis/Downloads>

Mobile Applikationen <ul style="list-style-type: none">• User Interface• Kamera-Service	Nach der Installation der Mobilen Applikationen werden die Apps auf dem Android-Gerät angezeigt. Die Mobile Applikation für das User Interface ist mit dem Namen „LED-Mapper“ benannt. Der Kamera-Service mit dem Namen „Camera-Service“. Beim Anklicken der Apps werden diese gestartet.
Standardsoftware <ul style="list-style-type: none">• Agent• LED-Strip-Service• Bildschirm-Service• Datenverarbeitungs-Service• Bildumwandlungs-Service	<p>Die Standardsoftware können über das jeweilige JAR ausgeführt werden. Mit einem Doppelklick auf ein JAR wird der jeweilige Service auf dem Bildschirm angezeigt.</p> <p>OpenCV Die Services für die Datenverarbeitung und die Bildumwandlung benötigen die Native Bibliothek von OpenCV. Aus diesem Grund muss sich für Windows die DLL -Datei im gleichen Verzeichnis befinden wie die jeweilige JAR-Datei des Services. Die Version von OpenCV muss 3.4.1 entsprechen. Für Mac und Linux empfehlen wir die Installation über folgenden Link: http://opencv-java-tutorials.readthedocs.io/en/latest/01-installing-opencv-for-java.html</p> <p>Nach der Installation muss der Benutzer die Datei unter folgendem Pfad „/usr/local/Cellar/opencv/3.x.x/share/OpenCV/java/libopencv_java341.dylib“ in dasselbe Verzeichnis kopieren, wie sich auch die JAR-Datei des Services befindet.</p>

	<p>Eine Alternative zum Kopieren der Datei „java341.dylib“, ist das JAR über das Terminal auszuführen. Mit folgendem Befehl wird die OpenCV-Library für die Ausführung zum Java-Library-Path hinzugefügt:</p> <pre><code>java -Djava.library.path= "/usr/local/Cellar/opencv/3.x.x/share/OpenCV/java" -jar YourJar.jar</code></pre>
MQTT-Broker: Mosquitto	<p>Linux: Mosquitto kann auf einem Linux-System mit dem Kommandozeilenbefehl: „sudo apt-get install mosquitto“ installiert werden. Mit dem Befehl „service mosquitto start“ kann der Broker nun gestartet werden.</p> <p>Mac: Mosquitto kann auf dem Mac mit dem Kommandozeilenbefehl: „brew install mosquitto“ installiert werden. Danach kann die Mosquitto-Installation mit folgendem Befehl für den LauchAgent vorbereitet werden: „In -sfv /usr/local/opt/mosquitto/*.plist ~/Library/LaunchAgents“ Danach kann der Mosquitto-Broker mit dem Befehl „launchctl load ~/Library/LaunchAgents/homebrew.mxcl.mosquitto.plist“ gestartet werden.</p> <p>Windows: Für die Installation von Mosquitto auf einem Windows-System, wird empfohlen folgendes Verzeichnis herunterzuladen: https://github.com/Haempu/LED-Confusion/Bachelorthesis/Downloads/mosquitto.zip</p> <p>Um Mosquitto zu starten muss man über die Kommandozeile in das Verzeichnis navigieren, in welchem sich der soeben erhaltene Ordner „mosquitto“ befindet. Mit dem Befehl „mosquitto“ wird der Broker anschliessend gestartet.</p>
Brick-Deamon	Der Brick-Deamon kann für alle gängigen Plattformen unter folgendem Link heruntergeladen werden: https://www.tinkerforge.com/de/doc/Software/Brickd.html#brickd

5 Umsetzung

5.1 Motivation

Die Aufgabenstellung des LED-Mapper-Projektes hat uns bereits sehr angesprochen. Als wir uns erste Gedanken über das Projekt machten, stellte sich heraus, dass sowohl eine Mobile-Applikation, wie auch Standard-Applikationen geschrieben werden können. Da wir die Vertiefung Mobile Computing besuchen, war diese Aufgabenstellung ideal. Ausserdem war eine Implementation von Tinkerforge-Komponenten vorgesehen. Dies bedeutet, dass wir unter anderem auch verhältnismässig hardwarenah programmieren müssen. Auch das hatten wir in der Vertiefung bereits behandelt und wir wollten in der Bachelorthesis nochmal näher darauf eingehen.

5.2 Vorgehensweise

Unser Ziel war, diese Bachelorthesis von Beginn an praxisorientiert umzusetzen. Das bedeutet, als erstes mussten wir uns Gedanken darüber machen, was genau erreicht werden soll. Dafür erstellten wir ein Pflichtenheft. Nachdem dieses erstellt war, planten wir die einzelnen Komponenten ein und versuchten die dafür benötigte Zeit abzuschätzen. Dies wurde in einem Projektplan festgehalten. Anschliessend kümmerten wir uns um die Umsetzung. Im Folgenden gehen wir darauf ein, wie es uns dabei ergangen ist. Welche Schwierigkeiten aufgetaucht sind und welche Lösungen wir dafür gefunden haben.

5.3 Projektplan

- Das Projekt wird von zwei Personen realisiert
- Pro Woche werden zirka 45 Personenstunden an nicht definierten Tagen gearbeitet. Geplant sind bis zur Abgabe der Bachelorthesis 656 Stunden. Nach der Abgabe sind weitere 67 Stunden geplant für den Finaltag, den Film und die Vorbereitung der Verteidigung eingeplant. Zusammen ergibt das 723 Stunden, was ziemlich genau der Vorgabe für eine Bachelorthesis für zwei Personen entspricht.

Hauptaktivität	Nummer	Teilaktivität	Nummer	Abhängigkeiten	Geschätzter Aufwand (SOLL) [Pers. h]	Effektiver Aufwand (IST)	Aufwandsdifferenz (SOLL - IST)	Zeitraum SOLL
Start	1	Start Bachelor Thesis	1	-	1	1	0	KW 08 - 19.02.2018
Projektumriss	2	Pflichtenheft erstellen	3	1	16	24	-8	KW 09
		Projektplan erstellen	4	3	10	12	-2	KW 09
Technische Dokumentation	5	GUI-Entwurf	6	2	12	8	4	KW 10
		Systemübersicht erstellen	7	2	10	14	-4	KW 10
		Verwendete Technologien definieren	8	2	2	5	-3	KW 10
		Ablaufdiagramm erstellen	9	2	4	5	-1	KW 10
Realisierung	10	Raspberry Pi aufsetzen	11	5	10	9	1	KW 11
		MQTT-Broker aufsetzen	12	5	4	3	1	KW 11
		User Interface für Smartphone	13	5	70	122	-52	KW 11
		MQTT-Anbindung an Mobile-App	14	5	10	20	-10	KW 11
		Kamera-Service für Smartphone	15	5	30	45	-15	KW 12
		MQTT-Anbindung an Kamer-Service	16	5	10	21	-11	KW 12
		Bildschirm-Service	17	5	40	30	10	KW 13
		MQTT-Anbindung an Bildschirm-Service	18	5	8	5	3	KW 13
		Agent mit MQTT-Anbindung	19	5	16	24	-8	KW 13
		Datenverarbeitungs-Service: Mapping-Funktion	20	5	80	52	28	KW 16

Hauptaktivität	Nummer	Teilaktivität	Nummer	Abhängigkeiten	Geschätzter Aufwand (SOLL) [Pers. h]	Effektiver Aufwand (IST)	Aufwandsdifferenz (SOLL - IST)	Zeitraum SOLL
		Datenverarbeitungs-Service: Leuchtquelle ansteuern	21	5	25	12	13	KW 16
		Testphase Meilenstein 1	22	11-22	16	10	6	KW 17
		Meilenstein 1	23	22	0	0	0	KW 17
		Ort und Zeit der Prüfung definieren	24	-	0	0	0	KW 17 - 25.04.2018
		LED-Strip-Service	25	23	50	42	8	KW 18
		MQTT-Anbindung an LED-Strip-Service	26	23	8	4	4	KW 19
		Testphase Meilenstein 2	27	25-26	10	8	2	KW 19
		Meilenstein 2	28	27	0		0	KW 19
		App-Erweiterung: Muster zeichnen	29	28	60	30	30	KW 20
		Testphase Meilenstein 3	30	29	8	4	4	KW 21
		Meilenstein 3	31	20	0		0	KW 21
		Bildumwandlungs-Service: Bild laden und anzeigen	32	31	40	22	18	KW 22
		Testphase Meilenstein 4	33	32	10	6	4	KW 22
		Meilenstein 4	34	33	0	0	0	KW 22
Abschlussphase	35	Dokumentation fertigstellen	36		60	105	-45	KW24
		Präsentation vorbereiten	37		10		10	KW 24
		Ausstellung vorbereiten	38		10	8	2	KW 24
		Reserve und Feinschliff Thesis	39		16	22	-6	KW 24
Total					656	673	-17	

5.3.1 Projektplan nach Abgabe der Bachelorthesis

Hauptaktivität	Nummer	Teilaktivität	Nummer	Abhängigkeiten	Geschätzter Aufwand (SOLL) [Pers. h]	Effektiver Aufwand (IST)	Aufwandsdifferenz (SOLL - IST)	Zeitraum SOLL
Finaltag	40	Ausstellung	41		10			KW 24 - 15.06.2018
		Präsentation	42		1			KW 24 - 15.06.2018
Verteidigung	43	Verteidigung vorbereiten	44	12	20			KW 24
		Filmdreh und Schnitt	45	12	35			KW 23
		Verteidigung	46	38	1			KW 25 - 18.06.2018
Filmabgabe	47	Filmabgabe	47	39	0			KW 25 - 22.06.2018
Total					67			

5.4 Technologien

5.4.1 Auswahlmöglichkeiten

5.4.1.1 Applikationsarten

Es sind keine Technologien für die Entwicklung der Systemkomponenten vorgegeben. Die Systemkomponenten können somit als Standardsoftware (bspw.: Java-Applikation, welche lokal auf einem Computer läuft), Web-Applikation (bspw.: PHP-Applikation, die auf einem öffentlich erreichbaren Server läuft) oder als Mobile-Applikation (bspw.: iOS- oder Android-Applikation) realisiert werden. Die Mobilen Applikation gliedern sich in folgende Unterkategorien.

Native Applikationen:

Die unterschiedlichen Betriebssysteme, wie Android, Windows und iOS bieten alle ein anderes Format für ihre Applikationen. Plattformabhängige Applikationen werden für genau eine Zielplattform angepasst und können somit nur auf einem Betriebssystem gestartet werden. Da eine Native-Applikation die Programmierschnittstellen der Zielplattform benutzt, wird das Einbinden von Hard- und Software-spezifischen Funktionen vereinfacht.

Web-Applikation:

Mobile Web-Applikationen werden über einen Webbrowser auf dem mobilen Gerät abgerufen und müssen somit nicht installiert werden. Mobile Webapplikationen scheinen auf den Benutzer wie eine native Mobile-Applikation und nicht wie eine normale Webseite. Eine Mobile-Web-Applikation hat den Nachteil, dass eine langsame Internetverbindung die Geschwindigkeit der gesamten Applikation beeinflusst und es somit zu spürbaren Verzögerungen in der Interaktivität führen kann.

Hybrid-Applikation:

Eine Hybrid-Applikation kann unabhängig von der Plattform auf verschiedenen Betriebssystemen und Endgeräten ausgeführt werden. Hybrid-Applikationen können somit gleich wie Web-Applikationen auf allen Plattformen gestartet werden. Der Unterschied ist jedoch, dass die Applikation im Hintergrund (für den Benutzer nicht sichtbar) innerhalb des nativen Webbrowsers der jeweiligen Plattform läuft.

Cross-Plattform-Applikation:

Cross-Plattform-Applikationen sind plattformunabhängig und können somit auf verschiedenen Betriebssystemen und Endgeräten ausgeführt werden. Im Gegensatz zu hybriden Applikationen wird die Benutzeroberfläche meist mit den jeweiligen Programmierschnittstellen der Zielplattform gebaut und nicht in einem Webbrowser, sondern als unabhängige Mobile-Applikation angezeigt.

[10]

5.4.2 Entscheidung

5.4.2.1 Applikationsarten

Mobile-Applikationen <ul style="list-style-type: none"> • User Interface • Kamera-Service 	<p>Für Umsetzung des User Interfaces entscheiden wir uns für eine native Mobile-Applikation. Das User Interface soll auf einem portablen und handlichen Gerät verwaltet werden und schliesst somit eine Standardsoftware aus. Wir besuchen die Vertiefung „Mobile Computing“ und haben die theoretische Grundlage zur Android-Entwicklung erhalten. Die Bachelorarbeit bietet die optimale Möglichkeit, die Theorie in die Praxis umzusetzen und eine native Android-Applikation zu entwickeln. Die Android-Applikation werden wir aufgrund unserer Vorkenntnisse mit der Programmiersprache Java umsetzen.</p> <p>Für den Kamera-Service braucht es eine Kamera und eine Netzwerkverbindung sowie eine MQTT-Schnittstelle. Eine Mobile-Applikation erfüllt alle Kriterien und eignet sich daher für die Realisierung des Kamera-Services. Da wir das User-Interface für die Android-Plattform entwickeln, verwenden wir für die Umsetzung des Kamera-Services ebenfalls eine Android-Plattform. Der Kamera-Service wird ebenfalls mit der Programmiersprache Java entwickelt. Dies hat gegenüber von nicht-nativen Applikationen den Vorteil, dass Android uns eine einfache Schnittstelle für den Zugriff auf die Kamera zur Verfügung stellt.</p>
Standardsoftware <ul style="list-style-type: none"> • Datenverarbeitungs-Service • Bildschirm-Service • LED-Strip-Service • Bildumwandlungs-Service • Agent 	<p>Programmiersprache Jeder Service sowie der Agent werden als Standardsoftware mit der Programmiersprache Java umgesetzt. Java ist aufgrund der Plattformunabhängigkeit für die einzelnen Services geeignet. Somit spielt es keine Rolle ob ein Service auf einem Linux-, Windows- oder MacOS-Betriebssystem läuft. Zudem haben wir durch das Studium Erfahrungen mit Java gesammelt. Somit müssen wir uns nicht in eine neue Technologie einlesen, was zu einem zusätzlichen Zeitaufwand führen würde. Ein zusätzlicher Faktor für die Entscheidung für Java ist der Led-Strip-Service. Tinkerforge bietet eine Java-Schnittstelle zu ihrem LED-Strip an. [11]</p> <p>Benutzerschnittstelle Für die Verwaltung der Services Bildschirm-Service, LED-Strip-Service, Datenverarbeitungs-Service sowie dem Agent benutzen wir die in Java integrierte JavaFX-Bibliothek. JavaFX stellt uns alle benötigten Komponenten zur Verfügung um die Services mit einer Benutzerschnittstelle zu verwalten. Auch für die virtuellen Leuchtkörper auf dem Bildschirm-Service benutzen wir JavaFX-Komponenten. Auch in der Entscheidung der Technologie für die Benutzerschnittstelle kommt das Argument der Erfahrung mit JavaFX zum Zuge.</p>

5.4.2.2 Kommunikation der Services

Da die einzelnen Services in sich geschlossene Systeme darstellen, müssen sie irgendwie miteinander kommunizieren können. Die Anforderung war zudem, dass die Services nicht örtlich gebunden sein müssen. Das heißt, jeder Service könnte auf einem anderen Gerät ausgeführt werden. Trotzdem ist jeder Service auf die anderen angewiesen, da sie erst zusammen ein funktionierendes Gesamtsystem bilden. Dieser Anwendungsfall ähnelt stark demjenigen, der im IoT (Internet of Things) anzutreffen ist. Einzelne Geräte, in unserem Fall Services, laufen bis zu einem gewissen Grad unabhängig von einer menschlichen Eingabe, müssen aber untereinander kommunizieren. Im IoT hat sich das Kommunikationsprotokoll MQTT (Message Queue Telemetry Transport) etabliert. Da wir das bereits mehrmals in der Schule eingesetzt hatten, fiel die Wahl relativ leicht. MQTT überträgt Nachrichten in der Form von puren Byte-Arrays. Das ist sehr praktisch, da so einerseits Text (Strings) genau so gut verschickt werden kann, wie Daten von Bildern (Byte-Arrays). Damit haben wir bereits alle Anwendungsfälle des LED-Mapper abgedeckt. Zusätzlich bietet MQTT eine Art Dateistruktur in Form von Topics. Nachrichten werden unter einem speziellen Topic veröffentlicht und nur an die Clients weitergeleitet, die sich mit diesem Topic verbunden haben. Durch diese Struktur können wir das Interface der einzelnen Services sehr gut abbilden.

5.4.2.3 Bildverarbeitung

Für die Verarbeitung der Fotos/Bilder haben wir uns für die native OpenCV-Bibliothek entschieden. Die Bibliothek bietet genau die Funktionalitäten, die der Bildumwandlungs- sowie der Datenverarbeitungs-Service benötigen. Mit der OpenCV-Bibliothek können die lichtintensivsten Stellen eines Bildes hervorgehoben werden. Zudem kann jedes einzelne Pixel und die dazugehörigen Informationen, wie der RGB-Wert des Pixels, ausgelesen werden.

5.4.3 Kommunikation mit MQTT

Sowohl für Android als auch für herkömmliche Java-Applikationen existiert eine *Paho-MQTT-Bibliothek von Eclipse* [12]. Die Bibliothek war uns aus der Vertiefung und vorherigen Projekten bereits bekannt. Die Herausforderung war, einen MQTT-Controller zu schreiben, den wir später auch gleich bei den anderen Services genau so implementieren konnten. MQTT ruft bei jedem Erhalt einer Nachricht einen Callback auf. Der Callback auf den Anwendungen führt zu einer Reaktion. Beispielsweise wenn das Mapping fertig ist, was das User Interface anzeigen muss. Da der MQTT-Controller aber vom Haupt-Controller aufgerufen wird, können wir nicht direkt im MQTT-Controller Befehle ausführen. Deshalb geben wir dem MQTT-Client beim Verbinden mit dem Broker den Callback vom Haupt-Controller mit. So wird erreicht, dass der Callback in der Hauptmethode aufgerufen wird. Dort haben wir Zugriff auf alle anderen Controller. Der MQTT-Controller stellt jetzt die Methoden zum Verbinden oder Trennen des Clients und zum An- und Abmelden an den Topics zur Verfügung. Auch Methoden zum Veröffentlichen von Nachrichten stellt er bereit. Eine weitere wichtige Aufgabe des MQTT-Controllers besteht im „Verwalten“ der jeweiligen Topics seines Services. Sämtliche Topics, auf die ein Service Zugriff hat und alle Nachrichten die er verschicken kann, sind im MQTT-Controller als globale Konstanten definiert. So können Controller, die abhängig von einem bestimmten Zustand andere Nachrichten versenden wollen, immer darauf zugreifen und entsprechende Vorgänge einleiten. Dieses Konzept konnten wir auf alle Services übertragen. So sieht der MQTT-Controller des Kamera-Service sehr ähnlich aus, wie derjenige des Datenverarbeitungs-Service. Natürlich jeder mit seinen eigenen Topics und Nachrichten.

5.5 User Interface

5.5.1 Struktur der Android-Applikation

Eine Android Applikation mit dem Model-View-Controller-Pattern (MVC) umzusetzen, hat sich als schwieriger herausgestellt als erwartet. Eine Android-Activity (Ansicht), wird mit einem Interface-BUILDER zusammengesetzt. Dieser erstellt aus der zusammengeklickten grafischen Benutzerschnittstelle ein XML. Dieses XML ist die eigentliche View im MVC-Pattern. Der Laufzyklus eines XML-Layouts wird jedoch durch eine Android-Activity programmatisch definiert. Nach dem MVC-Pattern wäre der Lebenszyklus einer Activity bereits ein Controller. In der Umsetzung unserer Android-Struktur haben wir den Lebenszyklus einer Activity jedoch auch wie eine View behandelt. Diese Entscheidung haben wir getroffen, um die View-Komponenten vom Controller zu trennen. Die Lebenszyklen der Activities beinhalten nun alle Komponenten bezüglich der Benutzerschnittstelle. Der jeweilige Controller zu einem Activity-Lebenszyklus führt Berechnungen aus und bietet eine Schnittstelle zum Datenbank-Controller.

5.5.2 Datenspeicherung und Datenstruktur

Zu Beginn des Projektes war geplant, dass das User Interface keine Daten speichert. Wir wollten den Fokus auf den Mapping-Vorgang und die Benutzerfreundlichkeit der Mobilen-Applikation legen. Die Überlegung war, jeweils die aktuellsten Daten vom MQTT-Broker zu lesen. Die Umsetzung zeigte jedoch, dass unser Datensystem komplexer wird als erwartet. Geplant war, ein Mapping-Vorgang als ein gesamtes System ohne Szenerien dem MQTT-Broker zur Verfügung zu stellen und dies vom User Interface zu lesen. Während der Realisierung bemerkten wir jedoch, dass der Benutzer mehrere Szenerien für ein Mapping abspeichern möchte. Jede einzelne Szenerie auf dem MQTT-Broker abzulegen würde den MQTT-Datenaustausch unnötig auslasten. Deshalb wird nun nur das aktuelle Mapping eines Systems auf dem Broker abgelegt und in der Datenbank gespeichert. Jede Szenerie, wie beispielsweise das Darstellen einer Italienflagge auf einer Beleuchtungsanlage wird jeweils in der Datenbank abgelegt.

5.5.3 Darstellen der virtuellen Leuchtquellen

Eine Beleuchtung wird nun auf dem User Interface mit virtuellen Leuchtquellen dargestellt. Dabei stiessen wir auf einige Probleme. Die Datenstruktur beinhaltet den Radius einer gemappten Leuchtquelle. Nach unserem Plan sollten die virtuellen Leuchtquellen auch mit dem entsprechenden Radius dargestellt werden. Je nach Einfall des Lichtes einer Leuchtquelle auf die Kamera können jedoch eigentlich gleich grosse Leuchtquellen unterschiedliche Radian besitzen. Bei der virtuellen Darstellung der Beleuchtungsanlage des Led-Strip-Services konnten die Leuchtquellen daher nicht originalgetreu dargestellt werden. Aus diesem Grund haben wir für jede virtuelle Leuchtquelle einen fixen Radius von 6Pixel definiert.

Ein Vorschlag unseres Betreuers war, das virtuelle Abbild Beleuchtungsanlage so zu realisieren, dass der Benutzer hinein zoomen kann. Dies bedingt jedoch, dass bei jedem Zoom die Koordinaten neu berechnet werden müssten. Da diese Erweiterung mit einem grossen Aufwand verbunden ist, haben wir uns gegen eine Umsetzung dieser Aufgabe entschieden.

5.5.4 Muster zeichnen / Farbauswahl

Um mit einer bestimmten Farbe ein Muster auf die Beleuchtungsanlage zu zeichnen muss der Benutzer die Farbe auswählen. Geplant war, dass der Benutzer die Farbe mit drei Schiebern für die Farben Rot, Grün und Blau von 0 bis 255 auswählen kann. Nach der Umsetzung der Schieber für die Farben, haben wir mit dem Betreuer besprochen, dass man eine Farbe mittels der Komponente ColorPicker [13] auswählen kann. Grund für den Wechsel der Komponente ist die vereinfachte Auswahl einer Farbe für den Benutzer.

5.6 Kamera-Service

5.6.1 Livestream und Foto schiessen

Aufgabe des Kamera-Service ist, einen Live-Stream der Kamera auf dem Bildschirm anzuzeigen, sowie ein Schiessen und Verschicken eines Bildes zu einem bestimmten Zeitpunkt. Da eine Android-Bibliothek bereits kameraspezifische Klassen zur Verfügung stellt, dachten wir, dass das Implementieren relativ schnell gehen wird. Leider stellte sich heraus, dass es doch nicht ganz so einfach war und wir hatten viel zu wenig Zeit dafür eingeplant. Sämtliche Beispiele, die wir im Internet gefunden haben, bezogen sich auf die alte Kamera-API von Android. Nur leider wird diese nicht mehr unterstützt. Mit der offiziellen Beschreibung von Android, konnten wir den Live-Stream schlussendlich doch noch erfolgreich einrichten [14].

Auf den Livestream der Kamera kann mit einer Android-CameraCaptureSession [15] zugegriffen werden. Diese wird genutzt, um die Bilder der Kamera zu prozessieren. In unserem Fall um den Live-Stream auf dem Handybildschirm anzuzeigen. Anschliessend wird der Stream in einer *Android-TextureView* [16] angezeigt. Diese bietet eine Methode um direkt von der View eine Bitmap-Datei zu erstellen. Dazu wird immer das aktuelle Bild, welches gerade angezeigt wird, benutzt. Diese Methode wird genau dann aufgerufen, wenn ein Bild angefordert wird. Zuerst haben wir diese Methode mit den Standardeinstellungen laufen gelassen. Dies dauerte aber vom Erhalten des Befehls bis zum Ausgeben des vollständigen Fotos jeweils zwischen zwei und sechs Sekunden. Da wir aber für jede Leuchtquelle zwei Fotos schiessen müssen, einmal das Referenz- und anschliessend das Vergleichsfoto, ist diese Zeit nicht akzeptabel. Das Problem war, dass das Foto in einer hohen Auflösung geschossen wurde. Für unsere Anwendung reicht aber ein Foto mit viel geringerer Auflösung. Sie beträgt jetzt nur noch 112x200 Pixel. Das Schiessen des Fotos geht so viel schneller (einige Millisekunden) und führt zu keiner wesentlichen Verzögerung mehr.

5.7 Bildschirm-Service

Der Bildschirm-Service sollte möglichst einfach gestaltet werden und wurde vor allem darum geschrieben, weil wir noch nicht wussten, wie genau das Mapping schlussendlich funktionieren wird. Der Bildschirm-Service besteht aus fünf Zeilen, die jeweils acht Kreise beinhalten. Die Kreise stellen die Leuchtquellen dar und können vom Bildschirm-Service an und ausgeschaltet werden. Da die MQTT-Schnittstelle grösstenteils vom Kamera-Service übernommen werden konnte, konnte der Bildschirmservice relativ schnell und ohne nennenswerte Probleme realisiert werden.

5.8 Agent

Der Agent ist derjenige, der entscheidet, was bei welcher einkommenden Nachricht getan werden soll. Er choreographiert also die einzelnen Services so, dass ein gesamtes System entsteht. Der Kern ist ein Switch-Case-Block. Dieser trifft eine erste Vorauswahl aufgrund des Topics der angekommenen Nachricht. Im entsprechenden Abschnitt wird dann aufgrund des Inhalts der Nachricht entschieden, welchem Service welcher Auftrag gegeben wird. Bei der Implementierung konnten wir einfach alle Nachrichten, die wir von irgendeinem Service losschicken durchgehen und für jede einzelne entscheiden was zu tun ist. Man kann hier zwar viele Fehler machen, meistens ist ein Topic falsch geschrieben, aber sie werden auch sehr schnell entdeckt. Weil das System funktioniert ganz einfach nicht solange der Agent Fehler aufweist. Deshalb war die erste Version des Agent schnell implementiert. Allerdings mussten später noch diverse Anpassungen gemacht werden weil entweder Topics abgeändert wurden, oder neue dazu kamen.

5.9 Datenverarbeitungs-Service

Die erste Version des Datenverarbeitungs-Service wurde nur zum Testen der anderen Services geschrieben. Ziel war, erst einmal die Kommunikation zwischen den verschiedenen Services zum Laufen zu bringen. Deshalb implementierten wir erst nur die „Standard“-MQTT-Klasse und ein wenig Logik, die auf eine bestimmte Anfrage, einfach die richtige Antwort gab. So konnten wir einen Mapping-Vorgang durchführen, der unabhängig vom eigentlichen Mapping-Algorithmus war. Dies erleichterte uns die ersten Tests wesentlich. Auch konnten wir uns anschliessend vollständig mit dem Datenverarbeitungs-Algorithmus auseinandersetzen.

Um Fehler beim Übertragen der Bilder auszuschliessen, realisierten wir jeweils eine Hilfsmethode zum Laden und Speichern von Bildern. Somit konnten wir manuell ein optimales Bild aufnehmen und dies in einem ersten Schritt von der Festplatte laden. Anschliessend konnten wir an unserem Algorithmus schreiben und ihn bequem austesten, ohne immer einen ganzen Mapping-Vorgang machen zu müssen. Beim Mapping-Vorgang erstellen wir erst von den beiden Originalfotos (Referenz- und Vergleichsfoto) jeweils ein reines Schwarz- / Weissfoto. Dies geschieht mit der Opensource-Bibliothek OpenCV. Anschliessend wird das Vergleichsfoto auf Weisskonturen untersucht, geprüft ob sich an der selben Stelle auf dem Referenzfoto auch eine Weisskontur befindet und wenn nicht, wurde die Leuchtquelle gefunden. Diese Methode funktioniert sehr gut. Allerdings ist sie verhältnismässig teuer was die Rechenleistung angeht. Da die Rechenzeit allerdings immer noch zu kurz ist, als dass das bemerkt würde, ist dies nicht all zu schlimm. Im Verlaufe dieser Arbeit ist jedoch eine weiter mögliche Methode aufgetaucht. Diese Methode würde ebenfalls erst aus beiden Bildern ein Schwarz- / Weissfoto erstellen. Diese würden jedoch anschliessend nicht auf ihre Konturen untersucht, sondern die beiden Bilder jeweils Pixel für Pixel voneinander Subtrahieren. Ist das Ergebnis 0, wurde entweder Schwarz von Schwarz, oder Weiss von Weiss subtrahiert. Dies würde bedeuten, die Bilder unterscheiden sich an diesem Pixel nicht. In jedem anderen Fall wäre das Ergebnis ungleich 0, die Bilder würden sich an dieser Stelle also unterscheiden. Der Vorteil dieser Methode wäre, dass sie mit viel weniger Rechenaufwand gemacht werden könnte. Allerdings könnte ein Nachteil sein, dass sich eine Weisskontur auf den beiden Bildern nicht exakt an der gleichen Stelle befindet. Man müsste also einen Weg finden, eine Toleranz einzuberechnen. Ob dies funktioniert und schlussendlich wirklich eine bessere Lösung ist als unsere, müsste noch untersucht werden.

5.9.1 Entdeckte Gefahr beim aktuellen Algorithmus

Der aktuelle Algorithmus basiert auf dem Bestimmen des Zentrums von gefundenen Weisskonturen. Eine Gefahr besteht darin, dass der Algorithmus davon ausgeht, dass das Zentrum der Weisskontur immer auch Weiss ist.

Zu Beginn der Arbeit hatten wir die Helligkeit der LEDs während des Mapping-Vorgangs zu hoch eingestellt. Zusätzlich richteten wir zufälligerweise eine LED genau in das Objektiv der Kamera. Diese Kombination hat das Bild rechts ergeben. Deutlich ist zu sehen, dass hier der Pixel im Zentrum der Weisskontur nicht weiss ist. Der aktuelle Algorithmus würde das Zentrum auf diesem Bild mit dem gleichen Pixel auf dem Referenzbild vergleichen. Beide Pixel wären schwarz und es würde kein Unterschied festgestellt werden. Das bedeutet, die Leuchtquelle würde fälschlicherweise nicht entdeckt

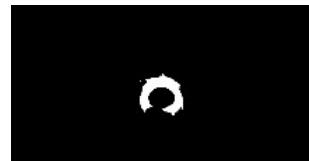


Abbildung 31 Weisskontur, wenn eine LED direkt auf die Kamera gerichtet ist

5.10 Zusammenführen aller Services

Mit dem User Interface so wie den Services der Kamera, der Datenverarbeitung, des Bildschirms und dem Agent, konnte das gesamte System erstmals zusammen getestet werden. Wie immer bei solchen frühen Tests, klappte nicht alles gleich auf Anhieb. Das System lief aber relativ schnell so, wie wir uns das vorgestellt hatten. Der Mapping-Vorgang funktionierte gut und relativ stabil. Allerdings nur unter sehr guten Bedingungen. Das bedeutet, keine störende Lichteinflüsse von ausserhalb und in relativ dunklen Räumen. Da wir dies aber erwartet hatten, konnten wir trotzdem mit dem Resultat zufrieden sein und der erste wichtige Meilenstein Planmäßig erreichen.

5.11 Led-Strip-Service

5.11.1 Unterstütze Konfigurationen

Nach dem Erreichen des ersten Meilensteins, ging es darum, den Bildschirm-Service durch den neuen Led-Strip-Service zu ersetzen. Durch den modularen Aufbau des LED-Mapper sollte dies relativ einfach möglich sein, da die anderen Services nicht von solchen Änderungen betroffen sind. Eine korrekt aufgebaute Kommunikation mit den Tinkerforge-Elementen ist jedoch etwas komplizierter als dies beim Bildschirm-Service der Fall war. Dabei mussten wir einen Kompromiss zwischen der Bedienungsfreundlichkeit des Services und einer möglichst breiten Unterstützung der verschiedenen Tinkerforge-Konfigurationen finden. Je mehr verschiedene Konfigurationen unterstützt werden, desto mehr Einstellungen müssen vor dem Starten des Services vorgenommen werden und desto komplizierter gestaltet sich die View des Services. Deshalb haben wir entschieden, dass nur ein bestimmter Treiber (WS 2801) des Led-Strip-Bricklets unterstützt wird. Auch unterstützen wir ausschliesslich die Farbkombination RGB in genau dieser Reihenfolge. Die Led-Strip-Bricklets von Tinkerforge würden auch andere Kombinationen unterstützen. Wir leiten diese aber bewusst nicht an die View des Led-Strip-Services weiter, eben darum, weil wir diese möglichst übersichtlich gestalten wollen. Sollten diese Einstellungen später einmal vorgenommen werden können, können sie sehr schnell auf die View genommen werden. Bis jetzt haben diese Konfigurationsmöglichkeiten für alle Led-Strip-Bricklets die wir getestet haben genügt.

5.11.2 Latenzzeit der LED-Strips

Eine relativ grosse Herausforderung bot uns die Latenzzeit zwischen dem Befehl zum Einschalten und dem tatsächlichen, physikalischen Einschalten der LEDs. Der Mapping-Algorithmus hat lange nicht funktioniert weil der Led-Strip-Service den Befehl zum Einschalten einer LED zwar gegeben und zurückgemeldet hat, die LED aber noch gar nicht eingeschaltet war. Dies hat dazu geführt, dass das Vergleichsbild vom Kamera-Service bereits geschossen wurde, bevor die LED auch tatsächlich geleuchtet hatte. Wir hatten das Problem lange auf die Seite geschoben, indem wir eine kurze Wartezeit zwischen dem Befehl zum Einschalten und dem Abschicken der Nachricht, dass der Befehl ausgeführt wurde, eingebaut hatten. Natürlich ist diese Lösung sehr unschön. Unser Betreuer hat uns aber darauf hingewiesen, dass Tinkerforge für dieses Problem eine Lösung bietet. Dazu muss man wissen, dass die Frames zum Aktualisieren der LEDs in gewissen Zeitabständen erfolgen. Tinkerforge bietet eine Methode, die immer genau dann aufgerufen wird, wenn sicher alle LEDs erfolgreich gerendert wurden. Die Methode heisst „frameRendered“. Der Led-Strip-Service darf also die Nachricht, dass er alle LEDs entsprechend eingeschaltet hat, erst dann schicken, wenn diese Methode aufgerufen wurde. Dies stellt eine ziemlich grosse Herausforderung dar, da der Led-Strip-Service für diesen Ansatz nicht gerade optimal aufgebaut ist. Auch war die Zeit sehr knapp und Änderungen an einem so wichtigen Teil der Applikation in letzter Minute vor zu nehmen, ist immer auch mit einem gewissen Risiko verbunden. Aus diesem Grunde haben wir uns entschieden, diese Änderung nicht mehr vor zu nehmen. Würde diese Arbeit aber weitergeführt, wäre das sicher einer der ersten Punkte die verbessert werden sollten.

5.11.2.1 Gammakorrektur

Während den Tests haben wir ab und zu die Erfahrung gemacht, dass wir über die Mobile-App eine Farbe für ein LED ausgewählt haben, die aber sobald sie von der LED angezeigt wurde ganz anders aussah. Da dieses Verhalten aber eine „Nebenerscheinung“ war und wir mit den anderen Tests genug zu tun hatten, ignorierten wir es erstmal. Wir schrieben es dem „ColorPicker“ von Android zu. Der ist relativ klein und wir nahmen an, dass wir die entsprechende Farbe einfach nicht gut genug mit dem Finger getroffen haben. Dies war ein ziemlicher Irrtum. Unser Betreuer wies uns darauf hin, wir sollen etwas zur Gammakorrektur im Internet nachschlagen. Sofort fiel uns wieder ein, was es mit der Gammakorrektur [17] auf sich hat, nämlich dass das menschliche Auge Farben nicht linear wahrnimmt. In dunklen Bereichen steigt die vom Menschen empfundene Helligkeit schneller an als in hellen. Dem Led-Strip von Tinkerforge aber übergeben wir die Werte einfach linear. Das führt dazu, dass ein Orange (z.B. 255, 127) zwar von der LED richtig dargestellt, vom menschlichen Auge aber nicht so wahrgenommen wird, wie wir das erwartet haben. Da der Bildschirm von Smartphones diese Gammakorrektur berücksichtigt, wird sie auf ihm anders dargestellt als auf den LEDs, die sie nicht berücksichti-

gen. Das führt dazu, dass wir dachten, wir würden die Farben nicht richtig „treffen“. Leider kam diese Erkenntnis zu spät, als dass wir sie noch hätten im Code korrigieren können.

5.12 Bildumwandlungs-Service

Der Bildumwandlungs-Service war von Beginn an nur als optionales Ziel gesetzt worden. Dies war auch gut so, denn als wir dazu kamen ihn zu programmieren, wurde die Zeit bis zum Abgabetermin schon langsam knapp. Wir waren aber der Meinung, dass diese Funktion den LED-Mapper erheblich verbessern würde und haben uns deshalb entschieden, eine rudimentäre Form des Bildumwandlungs-Service zu implementieren. Rudimentär deshalb, weil er „nur“ sehr einfach Formen korrekt übernehmen und darstellen kann. Der Bildumwandlungs-Service erhält einfach die Koordinaten eines bestimmten Mappings und ein Bild, welches zum Beispiel aus dem Internet heruntergeladen wurde. Da das Bild in einer viel grösseren Auflösung vorhanden ist als das Mapping darstellen kann, muss das Bild stark „komprimiert“ werden. Der Bildumwandlungs-Service weisst einfach jeder Leuchtquelle eines Mappings ein entsprechendes Pixel auf dem Bild zu. Dabei nimmt die Leuchtquelle die Farbwerte des ihm zugewiesenen Pixels ein. Durch diese Methode können einfach Motive wie zum Beispiel eine Italienflagge gut auf einem Mapping dargestellt werden. Schwieriger wird es bei Formen wie zum Beispiel der europäischen Flagge. Es ist reiner Zufall, ob der Bildumwandlungs-Service genau den Pixelbereich eines der Sterne trifft oder nicht. Wenn nicht, ist die Flagge einfach blau.

5.13 Testprotokoll

Verwendete Hardware:

- Computer: Think Pad mit Windows 10 Installation. Java-Version 1.8.0_112.
- Mobile: Nexus 5X mit Android-Betriebssystem. Android-Version 8.1.0.

Initialzustand:

- Services wurden neu auf die jeweiligen Plattformen installiert und sind ausgeschalten
- Kein MQTT-Broker gestartet
- Für den Led-Strip-Service muss der Brick-Deamon laufen

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
1	User Interface (UI) starten	<ul style="list-style-type: none"> • Applikation starten 	Die Applikation startet und zeigt die Ansicht „Meine Beleuchtungen“ an. Folgender Text erscheint auf der Ansicht: „Keine Beleuchtungen vorhanden“.	Ja
2	User Interface: Ansicht Neue Beleuchtung	<ul style="list-style-type: none"> • Klick auf den gelben, runden „+“-Button 	Die Ansicht „neue Beleuchtung“ wird angezeigt. Diese besteht aus einem Textfeld für den Namen der Beleuchtung und einem Textfeld für die IP des MQTT-Brokers.	Ja
3	User Interface: Verbindungsauflaufbau nicht möglich	<ul style="list-style-type: none"> • Name und IPv4-Adresse eingeben • Klick auf Button „Verbindung prüfen“ 	Nach einer kurzen Zeitspanne (5 Sekunden) erscheint die Nachricht „IP-Adresse ist nicht erreichbar“.	Ja
4	User Interface: Verbindungsauflaufbau möglich	<ul style="list-style-type: none"> • MQTT-Broker starten • IP des gestarteten Brokers im IP-Textfeld des UI eingeben • Klick auf Button „Verbindung prüfen“ 	Verbindung kann aufgebaut werden und eine Liste von erforderlichen Services wird angezeigt. Der Button im unteren Bildschirmbereich heisst nun „Erstellen und Mappen“.	Ja
5	Konsole: An Topic anmelden	<ul style="list-style-type: none"> • Mit gestartetem Broker verbinden und an das Topic „ch/bfh/bachelorthesis/ledmapper/#“ anmelden 	Anmeldung am Topic war erfolgreich. Konsole befindet sich in einem warten-Zustand und hat keine Fehlermeldung ausgegeben.	Ja
6	Datenverarbeitungs-Service (DV-S) starten	<ul style="list-style-type: none"> • Datenverarbeitungs-Service starten 	View des Datenverarbeitungs-Service erscheint.	Ja

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
7	DV-S: Verbindungs-aufbau nicht möglich	<ul style="list-style-type: none"> IP-Adresse 1.1.1.1 eingeben 	Nach einer kurzen Zeitspanne (5 Sekunden) erscheint im Textfeld „Status“ die Nachricht: „Fehler: MQTT-Verbindungsfehler“.	Ja
8	DV-S: Verbindungs-aufbau möglich	<ul style="list-style-type: none"> IP-Adresse des Brokers eingeben 	Verbindung kann aufgebaut werden und die Nachrichten: „Info: Mit MQTT-Broker verbunden“ und „Info: An allen MQTT-Topics angemeldet“ erscheinen. Alle Eingabefelder sind ausgegraut.	Ja
9	Konsole	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „dataProcessingIsHere;Datenverarbeitungs-Service;paho12345...“	Ja
10	DV-S: beenden	<ul style="list-style-type: none"> Auf Button „Stop“ drücken 	Verbindung wird unterbrochen und die Nachricht: „Info: Von MWTT-Broker abgemeldet“ erscheint. Alle Eingabefelder können wieder bearbeitet werden. Auf der Konsole erscheint die Zeile „dataProcessingIsGoing;Datenverarbeitungs-Service;paho12345...“.	Ja
11	Bildumwandlungs-Service (BU-S) starten	<ul style="list-style-type: none"> Bildumwandlungs-Service starten 	View des Bildumwandlungs-Service erscheint.	Ja
12	BU-S: Verbindungs-aufbau nicht möglich	<ul style="list-style-type: none"> IP-Adresse 1.1.1.1 eingeben 	Nach einer kurzen Zeitspanne (5 Sekunden) erscheint im Textfeld „Status“ die Nachricht: „Fehler: MQTT-Verbindungsfehler“. Alle Eingabefelder sind ausgegraut.	Ja
13	BU-S: Verbindungs-aufbau möglich	<ul style="list-style-type: none"> IP-Adresse des Brokers eingeben 	Verbindung kann aufgebaut werden und die Nachrichten: „Info: Mit MQTT-Broker verbunden“ und „Info: An allen MQTT-Topics angemeldet“ erscheinen.	Ja
14	Konsole:	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „pictureConvertingIsHere;Bildumwandlungs-Service; paho12345...“	Ja
15	BU-S: beenden	<ul style="list-style-type: none"> Auf Button „Stop“ drücken 	Verbindung wird unterbrochen und die Nachricht: „Info: Von MQTT-Broker abgemeldet“ erscheint. Alle Eingabefelder können wieder bearbeitet werden. Auf der Konsole erscheint die Zeile „pictureConvertingIsGoing;Bildumwandlungs-Service; paho12345...“.	Ja
16	Bildschirm-Service (BS-S) starten	<ul style="list-style-type: none"> Bildschirm -Service starten 	View des Bildschirm -Service erscheint.	Ja

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
17	BS -S: Verbindungs-aufbau nicht möglich	<ul style="list-style-type: none"> IP-Adresse 1.1.1.1 eingeben 	Nach einer kurzen Zeitspanne (5 Sekunden) erscheint im Textfeld „Status“ die Nachricht: „Fehler: MQTT-Verbindungsfehler“.	Ja
18	BS -S: Verbindungs-aufbau möglich	<ul style="list-style-type: none"> IP-Adresse des Brokers eingeben 	Verbindung kann aufgebaut werden. Ein neues Fenster erscheint im Vollbild-modus. Es ist ganz schwarz.	Ja
19	Konsole	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „luminaireIsHere;Bildschirm-Service; pa-ho12345...“	Ja
20	BS -S: beenden	<ul style="list-style-type: none"> Auf Button „Stop“ drücken 	Verbindung wird unterbrochen und die Nachricht: „Info: Von MQTT-Broker abgemeldet“ erscheint. Alle Eingabefelder können wieder bearbeitet werden. Auf der Konsole erscheint die Zeile „luminaireIsGone;Bildschirm-Service; pa-ho12345...“.	Ja
21	Led-Strip-Service (LS-S) starten	<ul style="list-style-type: none"> Led-Strip -Service starten 	View des Led-Strip -Service erscheint.	Ja
22	LS -S: Verbindungsau-fbau nicht möglich	<ul style="list-style-type: none"> IP-Adresse 1.1.1.1 eingeben Tinkerforge-Komponente darf nicht verbunden sein. 	Nach einer kurzen Zeitspanne (5 Sekunden) erscheinen im Textfeld „Status“ die Nachrichten: „Fehler: Verbindung zu Tinkerforge nicht möglich“ und „Fehler: MQTT-Verbindungsfehler“.	Ja
23	LS -S: Verbindungsau-fbau möglich	<ul style="list-style-type: none"> IP-Adresse des Brokers eingeben Tinkerforge-Komponenten verbin-den und richtige UIDs eingeben (Die benötigten UIDs können mit dem Brick-Viewer ausgelesen werden) 	Verbindung zu MQTT und Tinkerforge kann aufgebaut werden und die Nachrichten: „Info: Mit Tinkerforge-Komponenten verbunden“, „Info: Mit MQTT-Broker verbunden“ und „Info: An allen MQTT-Topics angemeldet“ erscheinen. Alle Eingabefelder sind ausgegraut.	Ja
24	Konsole	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „luminaireIsHere;Led-Strip-Service; pa-ho12345...“	Ja
25	LS -S: beenden	<ul style="list-style-type: none"> Auf Button „Stop“ drücken 	Verbindung wird unterbrochen und die Nachricht: „Info: Von MQTT-Broker abgemeldet“ erscheint. Alle Eingabefelder können wieder bearbeitet werden. Auf der Konsole erscheint die Zeile „luminaireIsGone;Bildschirm-Service; pa-ho12345...“.	Ja
26	Kamera-Service (K-S) starten	<ul style="list-style-type: none"> Applikation starten 	Ein Livestream des Kamerabildes erscheint. Die Farbe des Kreises oben rechts im Bild ist rot.	Ja

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
27	K-S: Verbindungsaufbau nicht möglich	<ul style="list-style-type: none"> Auf Einstellungsseite wechseln (oben rechts) und IP-Adresse 1.1.1.1 eingeben Verbinden drücken 	Einstellungsseite wird geladen. Keine Verbindung kann aufgebaut werden und es erscheint der Text: „Bitte geben Sie eine gültige IP-Adresse ein.“.	Ja
28	K-S: Verbindungsaufbau möglich	<ul style="list-style-type: none"> IP-Adresse des Brokers eingeben 	Bei erfolgreichem Verbindungsaufbau wechselt der Service wieder zum Livestream der Kamera. Der Punkt oben rechts hat die Farbe Grün.	Ja
29	Konsole	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „cameralsHere;Android Kamera;paho12345...“	Ja
30	K-S: Verbindung trennen	<ul style="list-style-type: none"> Einstellungsseite laden und „Verbindung beenden“ drücken 	Die Einstellungsseite ermöglicht nun das errichten einer neuen Verbindung. Die Verbindungsseite mit dem Namen „Einstellungen“ wurde geladen.	Ja
31	Konsole	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „cameralsGone;Android Kamera;paho12345...“	Ja
32	Agent starten	<ul style="list-style-type: none"> Agent starten 	View des Agents erscheint.	Ja
33	Agent: Verbindungs- aufbau nicht möglich	<ul style="list-style-type: none"> IP-Adresse 1.1.1.1 eingeben 	Nach einer kurzen Zeitspanne (5 Sekunden) erscheint im Textfeld „Status“ die Nachricht: „Fehler: MQTT-Verbindungsfehler“.	Ja
34	Agent: Verbindungs- aufbau möglich	<ul style="list-style-type: none"> IP-Adresse des Brokers eingeben 	Verbindung kann aufgebaut werden und die Nachrichten: „Info: Mit MQTT-Broker verbunden“ und „Info: An allen MQTT-Topics angemeldet“ erscheinen. Alle Eingabefelder sind ausgegraut.	Ja
35	Konsole	<ul style="list-style-type: none"> Ausgabewert auf Konsole überprüfen 	Auf der Konsole erscheint die Zeile: „dataProcessingIsHere; [agent;Agent;paho12345...]“.	Ja
36	Agent: beenden	<ul style="list-style-type: none"> Auf Button „Stop“ drücken 	Verbindung wird unterbrochen und die Nachricht: „Info: Von MQTT-Broker abgemeldet“ erscheint. Alle Eingabefelder können wieder bearbeitet werden. Auf der Konsole erscheint die Zeile „[]“.	Ja
37	Agent: starten	<ul style="list-style-type: none"> UI prüfen 	Auf dem UI wird der Punkt vor dem Agent grün.	Ja

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
38	Services starten	<ul style="list-style-type: none"> Nacheinander die Services: Kamera-Service, Datenverarbeitungs-Service und Leuchtquellen-Service starten 	Auf dem UI werden alle roten Punkte nacheinander grün.	Ja
39	Mapping starten	<ul style="list-style-type: none"> Auf dem UI wird der Befehl „Erstellen und Mappen“ gegeben. 	<p>Auf der Konsole müssen nacheinander folgende Nachrichten erscheinen:</p> <ul style="list-style-type: none"> startMapping dolInitialState done doPicture Picture (Byte-Array) referencePictureDone next0 done doPicture Picture (Byte-Array) Koordinate (z.B. 78;57;112;200) dolInitialState done doPicture Picture (Ist ein Array of Bytes) referencePictureDone next1 <p>Dieser Vorgang wiederholt sich so lange bis das Mapping abgebrochen wurde oder keine weitere Leuchtquelle mehr gefunden wurde.</p>	Ja
40	UI: Mapping stoppen	<ul style="list-style-type: none"> Auf dem UI den Button „Mapping stoppen“ 	<p>Auf der Konsole erscheinen die Nachrichten:</p> <ul style="list-style-type: none"> stopMapping Liste von Koordinaten ([0;97;..., ...,112;200]) dolInitialState <p>Das UI öffnet einen Dialog mit der Information darüber, wie viele Leuchtquellen gefunden wurden.</p>	Ja
41	UI: Mapping abschliessen	<ul style="list-style-type: none"> Dialog mit „OK“ bestätigen 	Die Ansicht mit der Überschrift „Szenerie“ erscheint.	Ja
42	UI: Szenerie-Dialog	<ul style="list-style-type: none"> Klick auf den runden gelben Button „+“ 	Ein Dialog wird geöffnet um eine Szenerie mit Namen zu erstellen.	Ja

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
43	UI: Szenerie erstellen	<ul style="list-style-type: none"> Name „Meine Szenerie“ im Dialog eingeben und Button „Erstellen“ klicken. 	Dem Benutzer wird die Ansicht „Übersicht Szenerien“ mit der erstellten Szene „Meine Szenerie“ dargestellt.	Ja
44	UI: Szenerie anwählen	<ul style="list-style-type: none"> Erstellte Szenerie auswählen. 	Die Ansicht mit der Überschrift „Meine Szenerie“ erscheint. Unter der Überschrift ist ein virtuelles Abbild der Beleuchtungsanlage zu sehen. Unter dem virtuellen Abbild ist der ColorPicker, ein Switch für die Konfiguration ob die Leuchtquellen an/aus geschaltet werden sollen und ein Schieber für die Helligkeit zu sehen.	Ja
45	UI: Leuchtquelle konfigurieren	<ul style="list-style-type: none"> Mit dem ColorPicker eine Farbe auswählen und eine oder mehrere Leuchtquellen berühren. Der Schalter „Leuchtquelle an/aus“ ist eingeschaltet und der Schieber für die Helligkeit voll auf. 	<p>Beim Berühren von einer oder mehreren Leuchtquellen werden diese auf dem virtuellen Abbild mit der ausgewählten Farbe „angemalt“. Die MQTT-Message mit den veränderten Leuchtquellen wird unter dem Topic „ch/bfh/bachelorthesis/ledmapper/ui/output/luminaire_changed/“ veröffentlicht.</p> <p>Die Leuchtquelle wird auf dem Leuchtquellen-Service physikalisch mit der ausgewählten Farbe angezeigt.</p>	Ja
46	UI: Leuchtquelle ausschalten	<ul style="list-style-type: none"> Variante 1 Der Schalter „Leuchtquelle an/aus“ ist aus. Eine oder mehrere Leuchtquellen berühren. Variante 2 Eine Leuchtquelle wird 2 Sekunden lang berührt. 	<p>Beim Berühren von einer oder mehreren Leuchtquellen werden diese auf dem virtuellen Abbild transparent dargestellt (Nur noch der Rand der Leuchtquelle ist zu sehen). Die MQTT-Message mit den veränderten Leuchtquellen wird unter dem Topic „ch/bfh/bachelorthesis/ledmapper/ui/output/luminaire_changed/“ veröffentlicht.</p> <p>Die Leuchtquelle wird auf dem Leuchtquellen-Service physikalisch ausgeschaltet.</p>	Ja
47	UI: Helligkeit konfigurieren	<ul style="list-style-type: none"> Der Schieber für die Helligkeit wird auf ca. einen Viertel verstellt. 	<p>Beim Berühren von einer oder mehreren Leuchtquellen werden diese auf dem virtuellen Abbild mit der ausgewählten Farbe „angemalt“. Die MQTT-Message mit den veränderten Leuchtquellen wird unter dem Topic „ch/bfh/bachelorthesis/ledmapper/ui/output/luminaire_changed/“ veröffentlicht.</p> <p>Die Leuchtquellen haben auf dem Leuchtquellen-Service eine deutlich schwächere Leuchtintensität als zuvor.</p>	Ja

Nr.	Beschreibung	Aktion	Zu erwartendes Ergebnis	Erfüllt
48	UI: Bild auswählen und Koordinaten schicken	<ul style="list-style-type: none"> In der Topbar das „Bild hinzufügen“-Icon wird ausgewählt und Foto auswählen. 	<p>Das Android-Gerät öffnet die Foto-Ansicht. Nun kann ein Foto ausgewählt werden. Für den Test wird eine Italienflagge ausgewählt. Die Koordinaten der Beleuchtungsanlage werden auf dem Topic „ch/bfh/bachelorthesis/ledmapper/ui/output/converting_coordinates/“ veröffentlicht.</p> <p>Sobald der Bildumwandlungs-Service die Koordinaten erhalten hat veröffentlicht er den Befehl auf das Topic „ch/bfh/bachelorthesis/ledmapper/ui/input/coordinates_received/“</p> <p>Nachdem die Koordinaten erhalten wurden, wird das ausgewählte Bild als Byte-Array auf das Topic „ch/bfh/bachelorthesis/ledmapper/ui/ output/converting_file/“ veröffentlicht.</p> <p>Das Bild wird vom Bildumwandlungs-Service nun auf die vorhandenen Koordinaten abgebildet und die Leuchtquellen werden mit der Identifikation und deren Farbe auf das Topic „ch/bfh/bachelorthesis/ledmapper/ui/ input/converted_file/“ veröffentlicht.</p> <p>Das Bild wird nun vom UI auf dem virtuellen Abbild geladen und auf den Leuchtquellen-Service übertragen. Die Italienflagge ist deutlich erkennbar.</p>	Ja
49	UI: Szenerie löschen	<ul style="list-style-type: none"> In der Szenerie-Ansicht in der Topbar das Papierkorb-Icon anklicken. 	Die Szenerie wird gelöscht und der Benutzer wird in die Ansicht „Szenerie-Übersicht“ geleitet. Da es die einzige Szenerie der Beleuchtung war, wird der Text „Keine Szenerien vorhanden“ angezeigt.	Ja
50	UI: Beleuchtung löschen	<ul style="list-style-type: none"> In der Ansicht „Szenerie-Übersicht“ in der Top-Bar das Papierkorb-Icon anklicken. 	Die Beleuchtung wird gelöscht und der Benutzer wird in die Ansicht mit der Überschrift „Meine Beleuchtungen“ geleitet. Da es die einzige Beleuchtung war, wird der Text „Keine Beleuchtungen vorhanden“ angezeigt.	Ja

6 Fazit

Zum Abschluss unserer Bachelorarbeit ist es an der Zeit, das Projekt zu analysieren und die geeigneten Lehren zu ziehen.

Die gesamte Bachelorarbeit hat uns nun fast sechs Monate beschäftigt. Die geleistete Arbeit ist jedoch mit neuem Wissen und interessanten Erkenntnissen belohnt worden.

Zu Beginn standen wir vor einer grossen Herausforderung, die nicht nur von uns, sondern auch von aussenstehenden Personen bestaunt wurde. Doch durch unsere bereits im Studium gesammelten Projekterfahrungen, konnten wir die Projektarbeit mit einem ansprechenden Resultat umsetzen. Die folgenden Punkte zeigen einen Überblick unserer wichtigsten Erfahrungen auf:

- **Service-Agent-Architektur**

Während der Bachelorarbeit konnten wir besonders von der Systemstruktur mit der Aufteilung der unterschiedlichen Services und einem Agent profitieren. Bis vor der Bachelorarbeit haben wir nur monolithische Applikationen geschrieben, wo alle Funktionen auf einem System laufen. Die Unabhängigkeit der Services und somit auch die simple Erweiterung des gesamten Systems, haben uns sehr beeindruckt. Die Service-Agent-Architektur ist ein Versprechen für die Zukunft, da immer mehr unabhängige Geräte miteinander kommunizieren müssen. Aufgrund der Unabhängigkeit der Services wird sich die Architektur wohl besonders im Internet-of-Things durchsetzen können. Die Architektur eignet sich zudem für eine Kommunikation über MQTT. Jeder Service kann sich zu den Topics subscriben und publishen, die für ihn wichtig sind. Bei einer entsprechenden Nachricht, kann der Service reagieren und seine Funktion zum gesamten System beitragen.

- **Android**

Bis zur Bachelorarbeit hatten wir nur theoretische Grundlagen zur Programmierung von Android-Applikationen. Das Projekt hat unser Wissen in diesem Bereich enorm gesteigert. Wir sind uns jedoch einig, dass wir in Zukunft, wenn nicht gefordert, keine nativen Mobile-Apps realisieren werden. In Zukunft werden wir Mobile-Applikationen umsetzen, die nur einmal für unterschiedliche Betriebssysteme geschrieben werden müssen.

- **Open-CV**

Für die Mapping-Funktion der Leuchtquellen, haben wir die Open-CV-Bibliothek verwendet. Das Erkennen einer Leuchtquelle mit der Bibliothek hat uns keine grosse Mühe zubereitet. Da Open-CV mit C++ geschrieben ist erwies sich die Einbindung der nativen Open-CV-Java-Bibliothek aufwändiger als erwartet. Die Native-Java-Bibliothek kann nicht in ein ausführbares JAR integriert werden und muss über Funktionen eines Windows-DLL oder unter Linux beziehungsweise Max OS X einer Shared-Library ausgeführt werden, die in C++ programmiert sind. Dennoch überwiegt die geniale Funktionalität der Bibliothek dem aufwändigen Einbinden.

- **Tests**

Vor jedem Meilenstein haben wir funktionale Tests zum jeweiligen Service im gesamten System mit verschiedenen Lichtverhältnissen durchgeführt. Die Tests haben sich als wichtigen Teil unseres Projektes erwiesen. Durch die Tests haben wir einige Fehler gefunden und korrigieren können. Ein Testprotokoll haben wir jedoch nur am Ende der Realisierungsphase für das gesamte System erstellt.

- **Projektplanung**

Der Zeitplan konnte nicht in allen Projektphasen gleich gut eingehalten werden. Durch das Aufteilen des gesamten Projektes in Teilaufgaben, hatten wir während den verschiedenen Projektschritten einen guten Überblick. Hohe Differenzen vom geschätzten und effektiven Aufwand gab es vor allem in der Realisierungsphase. Die grösste Differenz ist in der Umsetzung des User Interfaces mit einem geschätzten Aufwand von 70 und einem effektiven Aufwand von 122 Stunden festzustellen. Besonders die Ansicht für das Zeichnen auf eine Beleuchtungsanlage hat mehr Aufwand in Anspruch genommen als geplant. Glücklicherweise haben wir uns auch in die andere Richtung verschätzt und haben beispielsweise für die Funktion des Mapping-Vorgangs 28 Stunden zu viel eingeplant. Durch diese Differenzen aller Teilaufgaben haben wir über das gesamte Projekt gesehen nur 17 Stunden mehr investiert als geplant.

- **Qualität des Endproduktes**

Die Dokumentation sowie die Realisierung sind uns gelungen und wir können mit dem Endprodukt zufrieden sein.

6.1 Ausblick

Mit der Weiterentwicklung unseres Projektes, könnte folgende Systemkomponenten optimiert werden:

- **User Interface**

Das User Interface könnte aus unserer Sicht einfacher gestalten werden. In der Übersicht der Beleuchtungen und Szenerien werden diese als Buttons dargestellt. Mit einer Liste mit mehreren Spalten pro Zeilen könnten mehrere Beleuchtungen respektive Szenerien in der Übersicht dargestellt werden. Die wichtigste Erweiterung im User Interface ist jedoch Auswahl von mehreren Leuchtquellen- und Kamera-Services. Momentan kann die Mobile-App nur je einen Service auswählen. In einer weiteren Version des User Interfaces könnten die verschiedenen Services in einem Dropdown-Feld ausgewählt werden.

- **Kamera-Service**

Der Kamera-Service liefert die Daten nur im Hochformat. In der Weiterführung des Projektes könnte man die Kamera im Hoch- und Querformat auf die Beleuchtungsanlage richten und dieser würde die Fotos auch im entsprechenden Format liefern.

- **Led-Strip-Service**

Der Led-Strip-Service kann noch keine Gammakorrektur durchführen. Dies würde sich bei einer Weiterführung des Projektes relativ rasch erledigen lassen. Auch ist nach jedem Aktualisieren der LEDs ein fixes Timeout einprogrammiert. Das ist sehr unschön und wir würden das bei einem nächsten Mal mit dem FrameRenderedListener [18] von Tinkerforge lösen.

- **Datenverarbeitungs-Service**

Der Datenverarbeitungs-Service läuft ziemlich stabil und zuverlässig. Sicher würde es sich aber lohnen zu prüfen, ob die Vergleichsmethode mit dem Subtrahieren von zwei Bildern besser geeignet ist, als die aktuelle. Ebenfalls wäre sehr schön, wenn eine Lösung gefunden würde um die Schwarz/Weiss-Fotos anzuschauen. Im Moment ist das nur möglich, indem man in den Programmcode eingreift. Aber zum Überprüfen ob der Threshold richtig eingestellt ist, wäre es sehr praktisch, wenn das irgendwo sonst geschehen könnte.

7 Anhang

7.1 User Interface: Funktion getCoordinateFactorForView

```
/***
 * Funktion berechnet den Faktor um die Koordinaten auf der View richtig
 * anzuzeigen.
 *
 * @param resolutionSize
 * @param sizeImageView
 * @return
 */
private double getCoordinateFactorForView(int resolutionSize, int sizeImageView,
                                         boolean x) {
    if(x) {
        this.xPadding = (resolutionSize / PADDING_DIVIDE_FACTOR);
        double factor = (sizeImageView / (resolutionSize+(this.xPadding)));
        return factor;
    } else{
        this.yPadding = (resolutionSize / PADDING_DIVIDE_FACTOR);
        double factor = (sizeImageView / (resolutionSize+(this.yPadding)));
        return factor;
    }
}
```

7.2 User Interface: Funktion getLayoutParamsFor

```
/***
 * Funktion setzt alle Parameter wie Höhe/Breite und Position des Leuchtquellen-
 * Buttons.
 * @param coordinate
 * @param imageView
 * @return
 */
private FrameLayout.LayoutParams getLayoutParamsFor(Coordinate coordinate,
                                                 FrameLayout imageView, int width, int xMin, int xMax, int yMin, int yMax) {

    //Padding links und rechts
    width = width - Utils.pxFromDp(this.context, PADDING_IMAGE_VIEW);

    double xFactor = getCoordinateFactorForView((xMax-xMin), width, true);
    double yFactor = getCoordinateFactorForView((yMax-yMin),
                                                imageView.getLayoutParams().height, false);
    FrameLayout.LayoutParams params = new FrameLayout.LayoutParams(
        FrameLayout.LayoutParams.MATCH_PARENT,
        FrameLayout.LayoutParams.MATCH_PARENT);
    params.gravity = Gravity.TOP;
    params.topMargin = (int) Math.round(
        (coordinate.getyCoordinate()-yMin) * yFactor)
        +(int) Math.round(this.yPadding)+20;
    params.leftMargin = (int) Math.round(
        (coordinate.getxCoordinate()-xMin) * xFactor)
        +(int) Math.round(this.xPadding)+40;
    params.width = (int) Math.round(coordinate.getRadius());
    params.height = (int) Math.round(coordinate.getRadius());

    if (params.topMargin >= (this.yLast - X_IN_BETWEEN_THRESHOLD_IN_PX) &&
        params.topMargin <= (this.yLast + X_IN_BETWEEN_THRESHOLD_IN_PX)) {
        params.topMargin = this.yLast;
    }

    this.yLast = params.topMargin;
}

return params;
}
```

7.3 Erklärung zur Bachelorthesis



Erklärung der Diplandinnen und Diplanden *Déclaration des diplômant-e-s*

Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/Nom, Vorname/Prénom

Bösiger Elia

Datum/Date

14. Juni 2018

Unterschrift/Signature

A handwritten signature in black ink, appearing to read "Elia Bösiger".

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.



Erklärung der Diplomandinnen und Diplomanden *Déclaration des diplômant-e-s*

Selbständige Arbeit / *Travail autonome*

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbstständig durchgeführt habe. Alle Informationsquellen (Fachliteratur, Besprechungen mit Fachleuten, usw.) und anderen Hilfsmittel, die wesentlich zu meiner Arbeit beigetragen haben, sind in meinem Arbeitsbericht im Anhang vollständig aufgeführt. Sämtliche Inhalte, die nicht von mir stammen, sind mit dem genauen Hinweis auf ihre Quelle gekennzeichnet.

Par ma signature, je confirme avoir effectué ma présente thèse de bachelor de manière autonome. Toutes les sources d'information (littérature spécialisée, discussions avec spécialistes etc.) et autres ressources qui m'ont fortement aidé-e dans mon travail sont intégralement mentionnées dans l'annexe de ma thèse. Tous les contenus non rédigés par mes soins sont dûment référencés avec indication précise de leur provenance.

Name/Nom, Vorname/Prénom

Patrik Aebischer

Datum/Date

14. Juni 2018

Unterschrift/Signature



.....

Dieses Formular ist dem Bericht zur Bachelor-Thesis beizulegen.
Ce formulaire doit être joint au rapport de la thèse de bachelor.