Pflichtenheft

Projekt: LED-Cube

Version 1.0

Carlucci Ramon, Guntli Michael, Kuhn Thomas, Weber Remo

|  |  |  |
| --- | --- | --- |
| **Name** | **Datum** | **Unterschrift** |
| Carlucci Ramon | 29.09.2019 |  |
| Guntli Michael | 29.09.2019 |  |
| Kuhn Thomas | 29.09.2019 |  |
| Weber Remo | 29.09.2019 |  |

Änderungsgeschichte

| **Datum** | **Version** | **Autor** | **Beschreibung** |
| --- | --- | --- | --- |
| 2019-09-19 | 1.0 | R. Carlucci | Dokument erstellt  Die Information in dieser Tabelle muss nur dann aktualisiert werden, wenn eine neue Version des Dokumentes erzeugt, bzw. freigegeben wird, nicht aber jedes Mal wenn das Dokument 'angefasst' wird. |

Alle Abschnitte, die im Style "Comment" geschrieben sind (wie dieser Text), dienen nur für Erklärungen, die dem Autoren des Dokumentes helfen sollen. Diese Kommentare müssen im endgültigen Dokument entfernt oder unsichtbar gemacht werden.

Im Dokument werden verschiedene Felder, z.B. für das Datum, den Projektnamen, etc. verwendet. Diese sollen deshalb unbe­dingt zu Beginn unter den Doc properties (Dokumenteigenschaften) festgelegt werden.

Das Abschnittsraster soll nicht verändert werden. Wenn in einem bestimmten Abschnitt nichts zu definieren ist, dann soll das dort erwähnt, der Abschnitt aber nicht entfernt werden.

Die hier präsentierte Pflichtenheftvorlage ist angelehnt an **IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE Std 830-1998**. Sie kann aber auch für Projekte verwendet werden, die nicht nur aus Software beste­hen.

Gemäss DIN 69901-5 umfasst das Pflichtenheft die "vom Auftragnehmer erarbeiteten Realisierungsvorgaben aufgrund der Um­setzung des vom Auftraggeber vorgegebenen Lastenhefts", d.h. das Lastenheft beinhaltet die Kundenanforderungen, im Pflichten­heft sind technische Vorgaben an die Entwicklungsgruppe formuliert, z.B. allenfalls notwendige Vorgaben für die Programmier­sprache, die Plattformen, Betriebssystem, etc.. Im Pflichtenheft darf aber keinesfalls das Design beschrieben werden.

Im internationalen Umfeld werden statt der DIN-Normen eher die IEEE-Normen angewandt. Im IEEE Standard 830 wird eine "Software Requirements Specification" formuliert, welche sowohl das Lastenheft als auch das Pflichtenheft beinhaltet. Diese Vor­lage verfolgt diesen Ansatz. Teilweise werden Hinweise in Englisch direkt aus diesem Standard verwendet. Weitere Informa­tionen zu den einzelnen Punkten finden Sie direkt in [1].

The Requirements Specification should address the product, not the process of producing the product. Project requirements represent an understanding between the customer and the supplier about contractual matters pertaining to production of the product and thus should not be included in the Requirements Specification. These normally include items such as

* Cost;
* Delivery schedules;
* Reporting procedures;
* Development methods;
* Quality assurance;
* Validation and verification criteria;
* Acceptance procedures.

Project requirements are specified in other documents, typically in a software development plan, a software quality assurance plan, or a statement of work.

Inhaltsverzeichnis

1 Einleitung 5

1.1 Zweck 5

1.2 Produktüberblick 5

1.3 Definitionen, Akronyme und Abkürzungen 5

1.4 Referenzen 5

2 Allgemeine Beschreibung 6

2.1 Systemübersicht 6

2.2 Produktfunktionen 7

2.3 Benutzereigenschaften 7

2.4 Einschränkungen 7

2.5 Annahmen und Abhängigkeiten 7

2.6 Priorisierung der Anforderungen 7

3 Externe Schnittstellen 9

3.1 Hardwareschnittstellen 9

4 Funktionale Anforderungen 10

4.1 Überblick über die Systemfunktionen 10

4.2 Actors 10

4.3 Kurzbeschreibung der Use Cases 10

4.4 Use Case <Use Case Name> 10

5 Sonstige Anforderungen 12

A Referenzen 13

B Schema Hardware LED-Cube 14

C Schema Hardware Arduino Mega V3 15

Abbildungsverzeichnis

[Abbildung 1: Kontextdiagramm (Festlegung der Systemgrenze) 6](#_Toc476056215)

Tabellenverzeichnis

**Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.**

# Einleitung

## Zweck

Im vorliegenden Dokument sind die Anforderungen definiert, welche im Projekt *LED-Cube* um­ge­setzt wer­den müs­sen. Es beschreibt den Auftrag zwischen Auftraggeber und Auftragnehmer. Der Ausdruck *Pflich­ten­heft* ist hier im Sinne der *IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE Std 830-1998* ver­wen­det. Die dort definierte *Requirements Specification* beinhaltet sowohl die Benutzeranforderungen (*Las­ten­heft* ge­mäss DIN 69901-5) als auch Realisierungsvorgaben an die Entwicklungsgruppe (*Pflich­ten­heft* gemäss DIN 69901-5).

## Produktüberblick

Im Rahmen des Projekts *LED-Cube* entsteht ein GUI zu einem bestehenden LED-Cube. Der Nutzer soll über das GUI den LED-Cube steuern können. Die Abbildung 1 gibt eine grobe Übersicht über das ganze System.

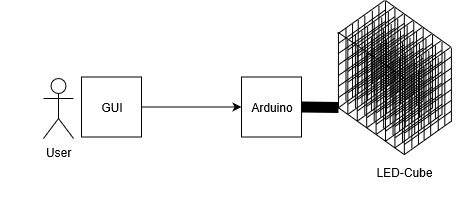


Abbildung 1: erste grobe Systemübersicht des LED-Cubes

Der zu entwickelnde Teil ist die Software, welche teils auf dem Rechner und teils auf dem Arduino ausgeführt wird.

Hier soll ein Überblick gegeben werden über die Teile, welche im Rahmen dieses Projekts entwickelt werden sollen, d.h. eine sum­marische Beschreibung darüber, welche Funktionalität das Produkt haben, und eventuell auch welche es nicht haben soll. Die zu entwick­elnden Teile sollen namentlich erwähnt werden. Ausserdem kann auch erläutert werden, wie das Produkt einge­setzt werden soll.

## Definitionen, Akronyme und Abkürzungen

GUI graphical user interface (grafische Benutzeroberfläche)

LED-Cube Würfel gitterstrukturartig aufgebaut mit den Dimensionen 8x8x8 Leds

oder

Name des Projekts

(es ist aus dem Zusammenhang klar, ob vom Projekt oder vom physischen Würfel die Rede ist)

Programm im Zusammenhang mit dem LED-Cube  
 *z.B. es wird ein bestimmtes Programm auf dem Würfel abgespielt*  
Das meint auf dem LED-Cube (physisch) werden verschiedene Zustände der Leds sequenziell nacheinander abgespielt.

(Optional) Define all terms, acronyms, and abbreviations used in this document. This often goes into a separate document in a larger project.

## Referenzen

siehe Anhang A auf Seite 12 dieses Dokumentes.

# Allgemeine Beschreibung

Der bestehende LED-Cube wird momentan über einen Arduino angesteuert, das bedeutet es kann ein fixes Programm auf dem Würfel abgespielt werden. Nun soll ein GUI entwickelt werden, über welches verschiedene Programme auf dem Würfel abgespielt werden können. Der Arduino wird weiterhin Bestandteil des Systems bleiben, da dieser die IOs zum Ansteuern des Würfels besitzt.

In diesem Kapitel sollen Hintergrundinformationen gegeben werden, keine spezifischen Anforderungen. Diese folgen im nächsten Kapitel. In diesem Kapitel muss genau geklärt werden, was zum System gehört und was nicht, d.h. die Systemgrenze muss hier zwingend festgelegt werden.

## Systemübersicht

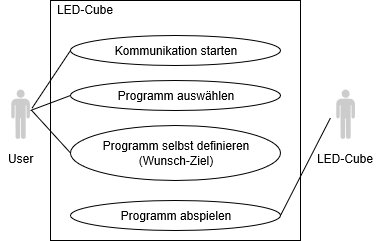


Abbildung 2: Kontextdiagramm (Festlegung der Systemgrenze)

Der ist die Steuerung über ein GUI des LED-Cubes. Der User kann auf dem GUI ein Programm auswählen, welches anschliessend auf dem LED-CUBE wiedergegeben wird. Zum Projekt gehört das Schreiben der Software zu diesem System.

### Software

Zu den Softwareaufgaben gehören folgende Punkte: entwerfen eines GUI, Implementation der USB-Schnittstelle vom Arduino, Handler, welcher Benutzereingaben verarbeitet und weiter zur Ausgabe an den Würfel gibt und das Anpassen der vorhandenen Software, damit diese implementiert werden kann.

### Arduino Mega V3

#### Schema

Schema des Arduino Mega V3 im Anhang C Schema Hardware Arduino Mega V3.

#### Software Bibliothek UART-Libary

Information zur UART-Libary vom Arduino Mega V3: <https://www.arduino.cc/en/Reference/SoftwareSerial>

### Hardware

Die Hardware ist vorhanden und wird nicht angepasst/geändert.  
Details zur Hardware im Anhang B Schema Hardware LED-Cube.

Definieren Sie das Umfeld des Systems mit den Schnittstellen des Systems zu seinem Umfeld und das System selbst. Hier muss die Systemgrenze in Form eines Kontextdiagramms gezogen werden. Aus diesem Abschnitt muss eindeutig hervorgehen, was inner­halb der Systemgrenze liegt (was muss entwickelt werden?) und was ausserhalb.

This subsection should also describe how the system operates inside various constraints. For example, these constraints could include

a) System interfaces;

b) User interfaces;

c) Hardware interfaces;

d) Software interfaces;

e) Communications interfaces;

f) Memory;

g) Operations;

h) Site adaptation requirements.

## Produktfunktionen

* Kommunikation starten (COM auswählen, Kontrollanzeige das eine Verbindung besteht)
* Programm auswählen, welches anschliessend auf dem LED-Cube wiedergegeben wird
* Programm selbst definieren

Hier soll eine **Zusammenfassung** der Hauptfunktionen angeboten werden, welche das System erfüllen soll.

## Benutzereigenschaften

Jeder Benutzer mit wenig technischem Hintergrund soll das Gerät verwenden können.

Festlegung der Benutzer (Benutzergruppen), welche mit dem System arbeiten sollen, z.B. Routinebenutzer, Servicetechniker, Ad­mi­nistrator. Dazu gehört auch die Definition der Kenntnisse und Erfahrungen der einzelnen Gruppen. Falls die funktionalen An­forderungen mittels Use Case – Diagrammen formuliert werden, dann entspricht dies den Actors, welche Personen repräsentie­ren.

## Einschränkungen

Einschränkungen sind gegeben durch den Clock-Speed 16 MHz und die maximale Baudrate von 115200 des Arduino Mega V3.0 und der Rechenleistung des verwendet PCs.

This subsection of the SRS should provide a general description of any other items that will limit the developer's options. These include

a) Regulatory policies;

b) Hardware limitations (e.g., signal timing requirements);

c) Interfaces to other applications;

d) Parallel operation;

e) Audit functions;

f) Control functions;

g) Higher-order language requirements;

h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);

i) Reliability requirements;

j) Criticality of the application;

k) Safety and security considerations.

## Annahmen und Abhängigkeiten

Es wird angenommen, dass die vorhandene Hardware einwandfrei läuft.

*This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the system but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.*

## Priorisierung der Anforderungen

### Muss-Anforderungen

* Programm auswählen, welches anschliessend auf dem LED-Cube wiedergegeben wird

### Soll-Anforderungen

* Kommunikation starten (COM auswählen, Kontrollanzeige das eine Verbindung besteht)

### Wunsch-Anforderungen

* Programm selbst definieren

Falls die Anforderungen unterschiedliche Prioritäten haben, bzw. einzelne Anforderungen erst in einer späteren Version imple­mentiert werden sollen, dann kann das hier aufgelistet werden, z.B.

* Muss-Anforderung: dies ist eine Anforderung, welche für das System essentiell und unabdingbar ist, bzw. das System würde keinen Sinn ergeben, wenn diese Anforderung nicht implementiert wäre.
* Soll-Anforderung: eine Soll-Anforderung ist nicht unabdingbar, trägt jedoch zur wesentlichen Verbesserung des Sy­stems bei. Sie soll wenn möglich realisiert werden.
* Wunsch-Anforderung (nice to have): diese Anforderung trägt zur Verbesserung des Systems bei, ist jedoch nicht unbe­dingt notwendig. Es ist ein Plus wenn diese Anforderung realisiert werden kann.

# Externe Schnittstellen

## Hardwareschnittstellen

### über Arduino zu LED-Cube

Die Schnittstelle von PC zu Arduino ist ein USB-Kabel realisiert. Die Schnittstelle zwischen Arduino und LED-Cube ist bereits bestehend und detailliert ersichtlich im Anhang B Schema Hardware LED-Cube. Zum besseren Verständnis ist unten noch eine Abbildung gezeigt.



Abbildung 3: Hardwareschnittstellen

## Benutzerschnittstellen

Dem User steht ein GUI auf dem PC zur Verfügung, welches einfach zu bedienen ist.

## Softwareschnittstellen

Die zu schreibenden Software wird in drei Blöcke aufgeteilt, wie die Abbildung unten zeigt.

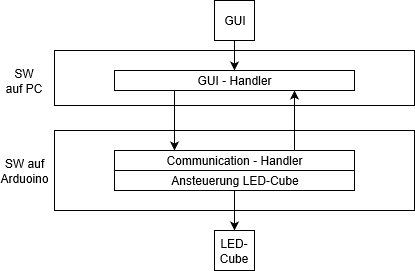


Abbildung 4: Softwareaufbau

Die Blöcke der Software haben folgende Aufgaben:

### GUI-Handler

Der GUI-Handler hat die Aufgabe die Eingaben vom GUI entgegen zu nehmen und diese per UART dem Communication-Handler zukommen zu lassen. Die Schnittstelle zwischen GUI-Handler und Communication ist noch zu definieren (in welchem Format die Daten geschickt werden und wie sie zu interpretieren sind). Weiter hat der GUI-Handler die Aufgabe die Kommunikation mit dem Arduino aufzunehmen.

### Communication–Handler

Der Communication-Handler nimmt vom GUI-Handler die gesendeten Informationen entgegen, interpretiert diese und macht die Ausgabe über den Block Ansteuerung LED-Cube.

### Ansteuerung LED-Cube

Dieser Block enthält Funktionen, die der Communication-Handler aufrufen kann um den LED-Cube anzusteuern.

Ab diesem Abschnitt sollen alle Anforderungen in einem Detaillierungsgrad geschrieben sein, der es den Entwicklern ermöglicht, ein System zu entwerfen, welches diese Anforderungen erfüllt, d.h. der Entwickler sollte nun klar wissen, welche Anforderungen umgesetzt werden müssen. Ebenfalls müssen Testingenieure aufgrund dieser Anforderungen ihre (System-)Testfälle definieren können.

Jede Anforderung sollte eindeutig identifizierbar sein, beispielsweise durch eine eindeutige Nummer.

In diesem Kapitel sollen, allenfalls in Abschnitten organisiert, die folgenden Schnittstellen beschrieben werden:

* Benutzerschnittstellen
* Hardwareschnittstellen
* Softwareschnittstellen
* Kommunikationsschnittstellen

# Funktionale Anforderungen

Die funktionalen Anforderungen sind die wichtigsten Teile zur Beschreibung des Systems. Für die Beschreibung der funktionalen Anforderungen bieten sich Use Cases an, nicht nur für Software, sondern auch für das ganze System. Jeder Use Case beschreibt dabei eine Funktion. Im folgenden Raster ist vorgesehen, dass bei jedem Use Case auch nicht funktionale Anforderungen wie Ant­wortzeiten, etc. direkt beim Use Case stehen, falls diese zu dieser Funktion gehören.

Bei der Use Case Definition ist wichtig, dass die Granularität nicht zu fein gewählt wird. Allfällige Ausnahmefälle einer Funk­tion sollen beispielsweise in der Beschreibung des entsprechenden Use Cases geschehen und nicht allenfalls in einem "Unter-Use Case".

Statt mittels Use Cases kann dieses Kapitel auch mit **Systemfunktion 1**, **Systemfunktion 2**, etc. gegliedert werden.

## Überblick über die Systemfunktionen

Die folgenden Use Cases beziehen sich auf die Abbildung 2 auf der Seite 6.

Hier kommt ein Use Case Diagramm hinein, das alle Funktionen zeigt. Allenfalls genügt ein Link zu Abschnitt 2.1 Systemübersicht.

## Actors

User: wählt gewünschtes Programm per Klick aus

Kurzbeschreibung der Actors.

## Kurzbeschreibung der Use Cases

|  |  |
| --- | --- |
| *Use Case* | *Beschreibung* |
| Kommunikation starten (Soll-Anforderung) | Das Programm erkennt, dass ein Arduino angeschlossen wurde und verbindet sich automatisch mit dem Arduino. Falls eine automatische Verbindung möglich ist, kann der COM manuell ausgewählt werden. Eine Anzeige signalisiert, ob das Arduino verbunden ist oder nicht. |
| Programm auswählen (Muss-Anforderung) | Das GUI besitzt für jedes mögliche abzuspielende Programm eine Taste, wird diese gedrückt, wird das entsprechende Programm auf dem LED-Cube abgespielt. |
| Programm selbst definieren (Wunsch-Anforderung) | Auf dem GUI befindet sich ein 8x8 Feld aus Tasten. Der User kann ein Muster mit den Tasten zeichnen und mit dem Knopf Start wird das Muster auf die vorderste Ebene des Würfels geschrieben und zyklisch eine Ebene nach hinten geschoben.  Auf dem GUI befindet sich ein Textfeld. Der User kann ein Text eingeben und mit dem Knopf Start wird der Text auf dem Würfel abgespielt. |

Jeder einzelne Use Case soll kurz beschrieben werden.

## Use Case “Kommunikation starten”

Ist für jeden Use zu beschreiben. Die folgenden Unterkapitel sollen für jeden einzelnen Use Case vollständig vorhanden sein. Falls bei einem Abschnitt nichts zu schreiben ist, dann soll dies entsprechend vermerkt werden, z.B. falls ein Use Case keine Vorbedingungen braucht oder keine nicht-funktionalen Anforderungen vorhanden sind, kann bei diesem Abschnitt einfach das Wort "keine" stehen.

### Vorbedingungen

Das USB-Kabel zum System ist noch nicht eingesteckt.

Zustand des Systems bevor der Use Case eintritt, z.B. kann hier stehen, dass ein System erfolgreich initialisiert sein muss, damit diese Funktion ausgeführt werden kann.

### Nachbedingungen

Die Kommunikation mit dem System wurde aufgenommen. Anzeige Verbunden ist grün.

Zustand des Systems nachdem der Use Case durchlaufen ist, z.B. kann hier bei einem Kalibrations-Use Case stehen, dass das System kalibriert oder allenfalls in einem Fehlerzustand ist.

### Nicht-funktionale Anforderungen

keine

Zusicherungen, die für Design und Realisierung wichtig sind, wie z.B. Antwortzeit, Häufigkeit, Priorität usw.

### Hauptszenario

Programm mit dem GUI wird gestartet. Dann wird nach dem Arduino gesucht und falls vorhanden verbunden.

Beschreibung des Use Cases, ggf. gegliedert in Einzelpunkte. Beschrieben wird der Normalfall. Variationen werden mit Unter­szenario-Nummer erwähnt (**[S-1]** , **[S-2]** usw.) und separat als Unterszenarien beschrieben. Fehlerfälle werden mit Fehlerszena­rio-Nummern angegeben ([E-1], [E-2] usw.) und separat als Fehlerszenarien beschrieben.

Beispiele:

Falls gewünscht können zusätzliche Informationen erfasst werden **[S-1]**.

Beim Einlesen der Daten können die Fehler **[E-1]**, **[E-2]** oder **[E-3]** auftreten.

### Unterszenarien

Programm mit dem GUI läuft bereits. Ein USB-Gerät wird eingesteckt. Das Programm prüft, ob es sich um das Arduino handelt und falls dies der Fall verbindet sich das Programm damit.

### Fehlerszenarien

#### Arduino wird nicht erkannt

Falls das das Arduino nicht erkannt wird, kann der User den COM manuell auswählen.

#### mehrere Arduinos sind am PC angehängt

Falls vorhanden bestimmte Nummer vom Arduino auslesen und anhand von dem mit dem richtigen verbinden.

Wenn dies nicht möglich ist, ist der User gezwungen den COM manuell auszuwählen.

### Regeln

keine

Gültigkeits- und Validierungsregeln, Berechnungsformeln usw.

### Anmerkungen

keine

### Beispiele

keine

## Use Case “Programm auswählen”

Ist für jeden Use zu beschreiben. Die folgenden Unterkapitel sollen für jeden einzelnen Use Case vollständig vorhanden sein. Falls bei einem Abschnitt nichts zu schreiben ist, dann soll dies entsprechend vermerkt werden, z.B. falls ein Use Case keine Vorbedingungen braucht oder keine nicht-funktionalen Anforderungen vorhanden sind, kann bei diesem Abschnitt einfach das Wort "keine" stehen.

### Vorbedingungen

Das Arduino ist mit dem PC verbunden und die Kommunikation zwischen Programm und Arduino steht.

Zustand des Systems bevor der Use Case eintritt, z.B. kann hier stehen, dass ein System erfolgreich initialisiert sein muss, damit diese Funktion ausgeführt werden kann.

### Nachbedingungen

Programm wird auf dem Würfel abgespielt.

Zustand des Systems nachdem der Use Case durchlaufen ist, z.B. kann hier bei einem Kalibrations-Use Case stehen, dass das System kalibriert oder allenfalls in einem Fehlerzustand ist.

### Nicht-funktionale Anforderungen

Die Anzahl Programme ist noch nicht bekannt, so programmieren damit es einfach erweiterbar ist.   
Für den ersten Wurf wird mit 6 Programmen gearbeitet.

Zusicherungen, die für Design und Realisierung wichtig sind, wie z.B. Antwortzeit, Häufigkeit, Priorität usw.

### Hauptszenario

Programm ist gestartet, Verbindung mit dem Arduino steht. Es wird die Taste für das Programm X gedrückt.

Beschreibung des Use Cases, ggf. gegliedert in Einzelpunkte. Beschrieben wird der Normalfall. Variationen werden mit Unter­szenario-Nummer erwähnt (**[S-1]** , **[S-2]** usw.) und separat als Unterszenarien beschrieben. Fehlerfälle werden mit Fehlerszena­rio-Nummern angegeben ([E-1], [E-2] usw.) und separat als Fehlerszenarien beschrieben.

Beispiele:

Falls gewünscht können zusätzliche Informationen erfasst werden **[S-1]**.

Beim Einlesen der Daten können die Fehler **[E-1]**, **[E-2]** oder **[E-3]** auftreten.

### Fehlerszenarien

#### Verbindung zum Arduino verloren

Aus irgendeinem Grund wird der gesendete Information vom Arduino nicht empfangen.

### Regeln

keine

Gültigkeits- und Validierungsregeln, Berechnungsformeln usw.

### Anmerkungen

keine

### Beispiele

keine

## Use Case “Programm selbst definieren”

Ist für jeden Use zu beschreiben. Die folgenden Unterkapitel sollen für jeden einzelnen Use Case vollständig vorhanden sein. Falls bei einem Abschnitt nichts zu schreiben ist, dann soll dies entsprechend vermerkt werden, z.B. falls ein Use Case keine Vorbedingungen braucht oder keine nicht-funktionalen Anforderungen vorhanden sind, kann bei diesem Abschnitt einfach das Wort "keine" stehen.

### Vorbedingungen

Das Arduino ist mit dem PC verbunden und die Kommunikation zwischen Programm und Arduino steht.

Zustand des Systems bevor der Use Case eintritt, z.B. kann hier stehen, dass ein System erfolgreich initialisiert sein muss, damit diese Funktion ausgeführt werden kann.

### Nachbedingungen

Programm wird auf dem Würfel abgespielt.

Zustand des Systems nachdem der Use Case durchlaufen ist, z.B. kann hier bei einem Kalibrations-Use Case stehen, dass das System kalibriert oder allenfalls in einem Fehlerzustand ist.

### Nicht-funktionale Anforderungen

keine

Zusicherungen, die für Design und Realisierung wichtig sind, wie z.B. Antwortzeit, Häufigkeit, Priorität usw.

### Hauptszenario

Programm ist gestartet, Verbindung mit dem Arduino steht. Es wird ein Muster auf dem 8x8 Tastenfeld erstellt. Es wird die Taste Start gedrückt.

Beschreibung des Use Cases, ggf. gegliedert in Einzelpunkte. Beschrieben wird der Normalfall. Variationen werden mit Unter­szenario-Nummer erwähnt (**[S-1]** , **[S-2]** usw.) und separat als Unterszenarien beschrieben. Fehlerfälle werden mit Fehlerszena­rio-Nummern angegeben ([E-1], [E-2] usw.) und separat als Fehlerszenarien beschrieben.

Beispiele:

Falls gewünscht können zusätzliche Informationen erfasst werden **[S-1]**.

Beim Einlesen der Daten können die Fehler **[E-1]**, **[E-2]** oder **[E-3]** auftreten.

### Fehlerszenarien

#### Verbindung zum Arduino verloren

Aus irgendeinem Grund wird die gesendete Information vom Arduino nicht empfangen.

#### gesendete Information kann nicht interpretiert werden

Es wird eine Information dem Communication-Handler gesendet, doch dieser kann die Information nicht interpretieren.

### Regeln

keine

Gültigkeits- und Validierungsregeln, Berechnungsformeln usw.

### Anmerkungen

keine

### Beispiele

keine

# Sonstige Anforderungen

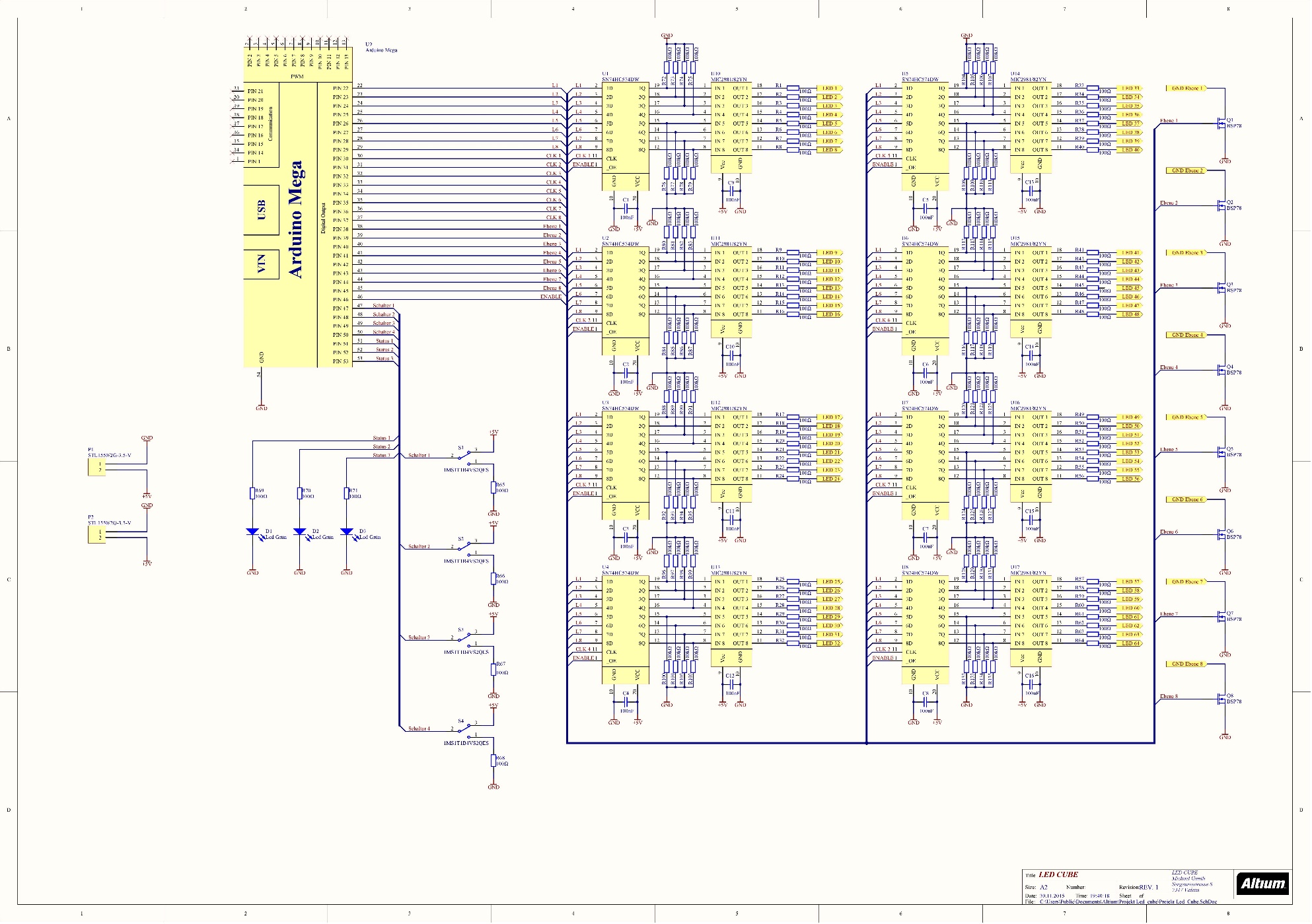
In diesesm Kapitel sollen alle bisher noch nicht spezifizierten Anforderungen, möglicherweise in Unterabschnitten, definiert werden, z.B.

* Leistungsanforderungen (Performance), die nicht direkt einer Systemfunktion zugeordnet werden konnten.
* Entwurfseinschränkungen (Design Constraints): Anforderungen, welche die Entwickler bei der Wahl des Designs einschränken.
* Qualitätsanforderungen wie Zuverlässigkeit, Verfügbarkeit, Sicherheit (Safety und Security, ist nicht dasselbe!)
* Wartbarkeit
* Portabilität
* Logging und Tracing
* Service und Support

1. Referenzen
2. IEEE Recommended Practice for Software Requirements Specification. ANSI/IEEE Std 830-1998

Do not repeat items covered in other documents or in a global project definitions and acronyms document

1. Schema Hardware LED-Cube



1. Schema Hardware Arduino Mega V3

