

Auftraggeber: gibb

Betreuer: Burri Matthias

Autor: David Jäggli, Nils Jäggi

Datum: 11.11.2020

Der Auftrag dieses Projektes ist es einen Fidget-Spinner zu realisieren, der mit dem PoV-Prinzip (Persistance of Vision) einen Text/Symbole anzeigen kann.

BüP

Hot-Board

Probelauf IPA

Inhalt

[1 Einleitung 3](#_Toc58149066)

[2 Hauptteil 4](#_Toc58149067)

[2.1 Blockschaltbild Hardware 4](#_Toc58149068)

[2.2 Schema 4](#_Toc58149069)

[2.2.1 Taster 4](#_Toc58149070)

[2.2.2 USB-Interface 4](#_Toc58149071)

[2.2.3 Spannungsregler 5](#_Toc58149072)

[2.2.4 Mikrocontroller 5](#_Toc58149073)

[2.3 Software 7](#_Toc58149074)

[2.3.1 Mikrocontoller 7](#_Toc58149075)

[2.3.2 Python-Skript 7](#_Toc58149076)

[3 Schlussteil 9](#_Toc58149077)

[3.1 Fazit 9](#_Toc58149078)

[3.2 Probleme 9](#_Toc58149079)

[3.3 Vergleich mit Aufgabenstellung 9](#_Toc58149080)

[3.4 Quellenverzeichnis 10](#_Toc58149081)

Anhang

[A Anhang 11](#_Toc528075874)

[A.1 Gesamtschema 11](#_Toc528075875)

[A.2 Blockschaltbild 12](#_Toc528075876)

[A.3 Pflichtenheft 13](#_Toc528075877)

[A.4 Bestückungsplan und Stückliste 14](#_Toc528075878)

[A.5 Bedienungsanleitung 16](#_Toc528075879)

[A.6 Zustandsdiagramm Hauptsoftware 17](#_Toc528075881)

[A.7 Zeitplan 18](#_Toc528075882)

**Abkürzungsverzeichnis**

|  |  |
| --- | --- |
| Abkürzung | Beschreibung |
| COM-Port | Communication-Port |
| ESD | Elektrostatische Entladung |
| GPIO | General Purpose Input/Output |
| GUI | Graphical User Interface |
| IC | Integrated Circuit |
| LDO | Low-Dropout |
| SWD | Serial Wire Debug |
| USB | Universal Serial BUS |
| XML | Textform zur hierarchischen Darstellung von Daten |

# Einleitung

**Open-Source**

Das gesamte Projekt mit allen benötigten Dateien ist zur freien Benützung öffentlich erhältlich auf Github

# Hauptteil

## Blockschaltbild Hardware

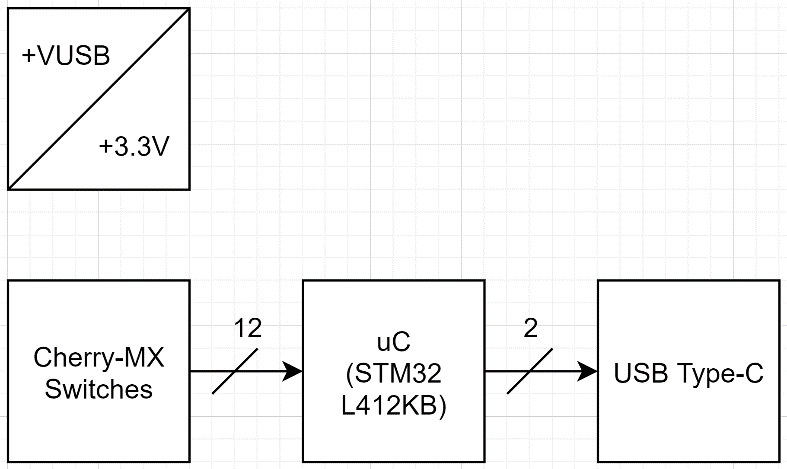
Die Hardware beinhaltet die 12 Taster und einen Mikrocontroller, welcher die Tastenanschläge in einen von der Software interpretierbaren Command umwandelt. Dafür wurde ein STM32L412KB verwendet. Dieser bringt bereits ein Protokoll zur Kommunikation über Virtual-COM-Port mit sich und somit ist kein externer Serial-to-USB Converter notwendig.

Abbildung : Blockschaltbild

## Schema

Abbildung : Schemaausschnitt Taster

### Taster

Als Taster werden Mechanische von der Marke «Cherry-MX» verwendet. Diese sind auf dem Tastaturen-Markt weit verbreitet und werden oft eingesetzt. In diesem Fall werden sie direkt an einen GPIO des Mikrocontrollers angeschlossen und per Pull-Down abgeschlossen, welcher für einen Definierten Zustand sorgt.

### USB-Interface

Die Ausgewerteten Daten werden vom Mikrocontroller über die Serielle Schnittstelle an einen Computer gesendet. Als Anschluss wurde eine USB TYPE-C Buchse verwendet.

IC3 dient als ESD-Schutz gegen Überspannung auf den Datenleitungen.

Zu beachten ist hier, dass alle Pins doppelt vorhanden sind. Dies liegt daran, dass USB-C auf 2 Wege eingesteckt werden kann.

Die beiden Widerstände R8/R10 dienen als Terminierungswiderstände und teilen dem Host mit, dass das Gerät mit Spannung versorgt werden soll.

Abbildung : Schemaausschnitt USB-Buchse

### Spannungsregler

Mithilfe eines LDO-Spannungsreglers werden die 5V der USB-Schnittstelle auf 3.3V geregelt. Diese tiefere Spannung ist nötig für den Mikrocontroller, da dieser mit 3.3V arbeitet.

Hierfür wurde ein sehr einfacher Regler verwendet, da die gesamte Schaltung nie über 5mA braucht.

Abbildung : Schemaausschnitt Spannungsregler

### Mikrocontroller

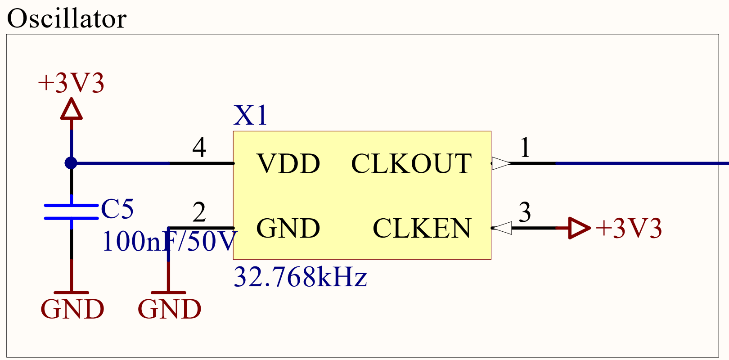
Bei der Auswahl des Controllers wurden die folgenden Mindestanforderungen gesetzt:

* Integrierte USB-Schnittstelle
* STM32L-Familie (Low-Power Familie)
* Mindestens 12 GPIO’s
* Handlötbarer Formfaktor

Das gewählte Modell ist der STM32L412KB. Dieser bietet genügend Pins und eine USB-Schnittstelle. Das Gehäuse ist ein LQFP mit 32 Pins.

Die Abbildung stellt die Beschaltung des IC’s dar. Die drei Kondensatoren dienen als Abblockkondensatoren an der Speisung.

Abbildung : Schemaausschnitt Mikrocontroller



Getaktet wird der Controller durch einen internen Oszillator, genannt MSI. Dieser versorgt die interne Peripherie mit 48MHz. Die Genauigkeit dieses Taktes ist jedoch ungenau und durch Temperatur sehr schwankend. Deshalb ist extern ein Taktgenerator (X1) angeschlossen, welcher den Controller mit einem 32.768kHz Signal speist. Mithilfe dieses Taktes wird danach der interne 48MHz Clock kalibriert. Die Genauigkeit ist wichtig, da die USB-Schnittstelle einen möglichst genauen Takt benötigt, um eine Fehlerfreie Datenübertragung zu gewähren.

Abbildung : Schemaausschnitt Taktgenerator

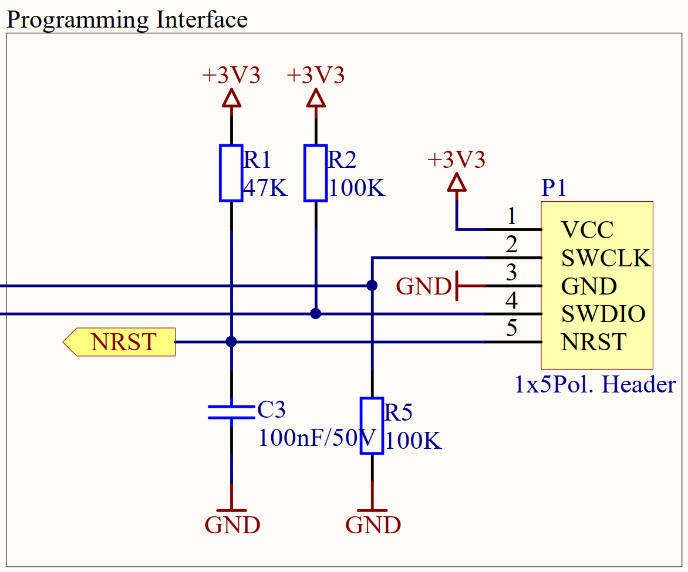
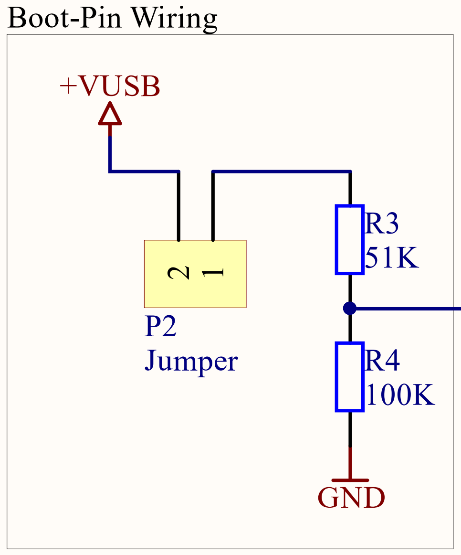
Um den Mikrocontroller zu Programmieren oder zu Debuggen, steht die SWD Schnittstelle zur Verfügung.

Abbildung : Schemaausschnitt Programmierinterface

Sie beinhaltet zudem die Reset-Beschaltung (R1/C3), welche auf VCC verbunden sein muss.



Der «BOOT0» Pin des Controllers bestimmt, wie er aufstartet. Standardmässig ist dieser mit einem Pull-Down Widerstand (R4) auf Masse gezogen, sodass er normal aufstartet und das Programm ausführt. Alternativ kann der Jumper P2 geschlossen werden und beim Startup wird dadurch der Bootloader gestartet. Dies erlaubt es, den Controller, alternativ zur SWD-Schnittstelle, per USB zu Programmieren. R3/R4 dient als Spannungsteiler, da hier direkt die Spannung der USB-Schnittstelle verwendet wird.

## Software

### Mikrocontoller

Auf dem Controller läuft ein Programm, welches bei einer gedrückten Taste den entsprechenden Command sendet. Insgesamt werden pro Tastendruck 3 Bytes geschickt im Format: «sXX», wobei «XX» der Nummer des gedrückten Tasters entspricht. Die Taster werden zusätzlich Softwareseitig entprellt.

Um später das GUI am Computer einfach öffnen zu können, wurde beim ersten Taster noch eine spezielle Funktion hinzugefügt. Wenn dieser länger als 2 Sekunden gedrückt wird, sendet es anstatt «s01», «s00».

### Python-Skript

Die Hauptsoftware läuft auf dem Computer und ist in Python geschrieben. Dazu wurden eigene Libraries geschrieben, um die einzelnen Aspekte aufzuteilen. Bereits vorhandene Module, wie «pySerial» (USB-Interface) oder «xml» konnten eingebunden und verwendet werden. Der grösste Teil des Codes in Python befasst sich mit den Prozessen, welche im Hintergrund ablaufen. Das beinhaltet Aspekte, welche dazu dienen die gewünschten Funktionen bei einem Tastendruck auszulösen.

#### GUI

Das GUI besteht momentan aus einer Konsole, bei der man mit multiple choice Fragen auswählen kann, was man ändern möchte. Das heisst es stehen Fragen und man kann dann mit vordefinierten Antworten auswählen welche Funktion bei welchem Taster ausgeführt wird. In Zukunft ist geplant ein eigenes GUI zu kreieren welches sehr viel übersichtlicher und auch intuitiver aussieht. Dies war aber im Moment aufgrund des Zeitdruckes nicht möglich.

#### Serielle Schnittstelle

Damit erkannt werden kann, wann ein Taster gedrückt wurde, wird bei einem Tastendruck auf dem Board ein individueller String geschickt. Die möglichen Kombinationen auf unserem Board sind daher «s01» bis «s12». Dazu kommt noch «s00», welcher das GUI öffnet.

Auf dem Computer wird mit dem laufenden Python Skript in einer unendlichen Schlaufe der serielle Buffer ausgelesen und nach einem gültigen Format String gesucht. Dazu wurde eine Virtual COM Port Library geschrieben. Diese durchsucht alle verfügbaren COM Ports auf dem PC nach der Hardware und versucht sich damit zu verbinden. Wenn dies erfolgreich ist, ist die Verbindung stabil und das Hot-Board funktioniert. Wenn kein gültiges Board verbunden wurde, oder die Verbindung nicht erfolgreich war, wird ein Fehler ausgegeben.

#### XML

Die Funktionen der einzelnen Switches werden in einem XML File namens «settings.xml» gespeichert. Somit kann das Programm nach einem Neustart, die eingestellten Funktionen wiederherstellen. Um all dies zusammenzufassen haben wir eine weitere Library geschrieben. Alle Funktionen, welche etwas mit den abgespeicherten Daten sowie der Ausführung der einzelnen Funktionen zu tun haben sind in dieser Library zusammengefasst. Sofern sich keine gültige Datei in diesem Pfad befindet, die den Namen settings hat, wird eine neue erstellt und mit den nötigen Daten gefüllt. Der Datenstamm sieht wie folgt aus:

* Root
  + User
    - Switch 1
    - Switch ...
    - Switch 12

Jeder Switch hat 3 Attribute: «name», «function» und «additional\_information». «name» ist dabei die gleiche Bezeichnung, welche vom Hot-Board über USB gesendet wird, also «s1» – «s12». «function» beinhaltet, die Funktion des Tasters welche ausgeführt werden soll. Und «additional\_information» beinhaltet die benötigten Informationen, um die Funktion auszuführen. Bei der Funktion «Datei ausführen» wäre dies zum Beispiel der Pfad zur Datei und bei einem Shortcut, die Tasten, welche gedrückt werden sollen.

# Schlussteil

## Fazit

## Probleme

## Vergleich mit Aufgabenstellung

## Quellenverzeichnis

Microchip: PIC18(L)F2X/45K50, Microchip, 27.07.2018, <http://ww1.microchip.com/downloads/en/DeviceDoc/PIC18F2X_45K50-30000684B.pdf>

Würth Elektronik: WL-SFTW SMD TOP LED full-color waterclear, Würth Elektronik eiSos GmbH & Co. KG, 27.7.2018,<http://www.farnell.com/datasheets/1911491.pdf?_ga=2.146832631.1915304690.1540276737-1213338847.1534140400>

Infineon: TLE4946-2K, infenion, 10.10.2018, <http://www.farnell.com/datasheets/1932173.pdf?_ga=2.190443627.1915304690.1540276737-1213338847.1534140400>

Texas Instruments: TPS6220x High-Efficiency, Texas Instruments, 27.10.2018, <http://www.ti.com/lit/ds/symlink/tps62203.pdf>

1. Anhang
   1. Gesamtschema
   2. Blockschaltbild
   3. Pflichtenheft

**Beschreibung**

Ziel des Projekts ist es einen Fidget-Spinner zu realisieren, der mit dem PoV (Persistance of Vision) - Prinzip einen Text anzeigen kann. Dazu ist es nötig LEDs auf einem selbst konstruierten Fidget-Spinner zu installieren und diese im richtigen Moment anzusteuern. Mit der Trägheit des Auges «sehen» wir dann einen Text. Der angezeigte Text und die Farbe sollen jederzeit über die Serielle, Virtual COM-Port Schnittstelle verändert werden können. Dazu soll, wenn möglich, eine eigene Windows Applikation geschrieben werden. Ansonsten wird der Fidget Spinner mit dem Programm TeraTerm konfiguriert.

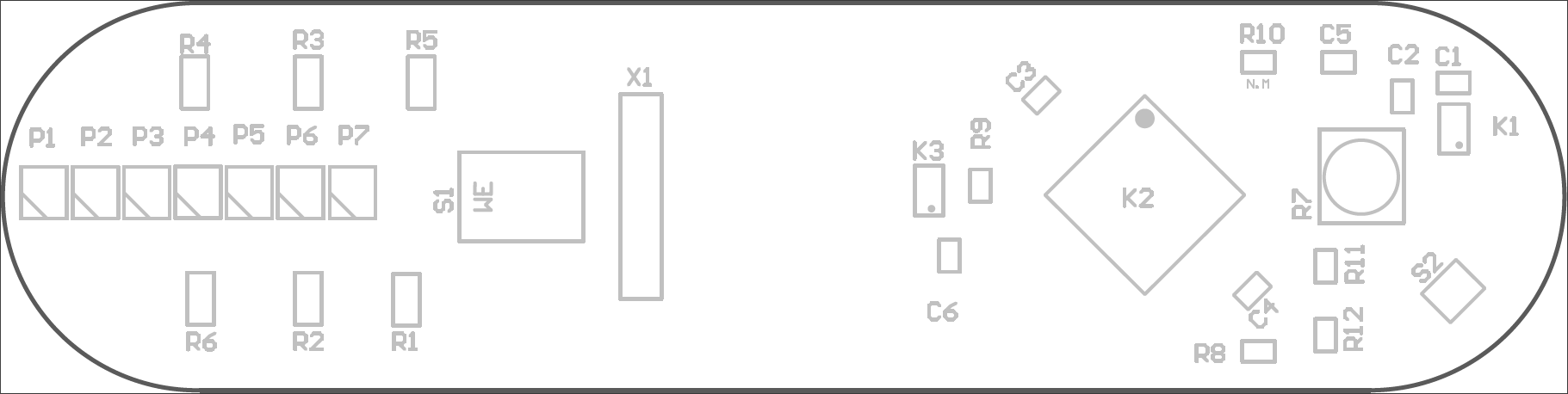
**Obligatorische Ziele**

* Einen Text anzeigen lassen (min. 15 Zeichen)
* Den Text verändern können (Tera Term)
* RGB LEDs für die Textanzeige verwenden
* Farben einstellbar (7 Farben)
* PIC-Mikrocontroller mit USB verwenden
* Gute Gewichtsverteilung auf den beiden Flügeln des Fidget-Spinners
* Speisung über Batterien
* Schrift bleibt kontinuierlich an derselben Stelle

**Optionale Ziele**

* Farben dimmbar (volles Farbspektrum)
* Eigene Windows-Software (C#) zum definieren des Textes bzw. der Farbe schreiben
  1. Bestückungsplan und Stückliste

**Oberseite**



**Stückliste Oberseite**

|  |  |  |  |
| --- | --- | --- | --- |
| Menge | Position | Bezeichnung | Händler |
| 1 | C1 | Cap, Cer, 4,7uF, 0805 | Distrelec |
| 1 | C2 | Cap, Cer, 10uF, 0805 | Distrelec |
| 3 | C3, C4, C6 | Cap, Cer, 100nF, 0805 | Distrelec |
| 1 | C5 | Cap, Cer, 470nF, 0805 | Distrelec |
| 1 | K1 | Buck-Converter, TPS62203, SOT23-5 | Distrelec |
| 1 | K2 | Microcontroller, PIC18F45K50, TQFP44 | Farnell |
| 1 | K3 | Hall-Effekt-Sensor, TLE4946-2K, SOT23-3 | Farnell |
| 7 | P1 – P7 | RGB-LED, 3528 | Farnell |
| 2 | R1, R2 | Widerstandsnetzwerk, 220R, 0612 | Distrelec |
| 2 | R3, R4 | Widerstandsnetzwerk, 330R, 0612 | Distrelec |
| 2 | R5, R6 | Widerstandsnetzwerk, 560R, 0612 | Farnell |
| 1 | R7 | Spule, 10uH | Farnell |
| 2 | R8, R9 | Res, 10k, 0805 | Distrelec |
| 1 | R11 | Res, 4,7k, 0805 | Distrelec |
| 1 | R12 | Res, 6,8k, 0805 | Distrelec |
| 1 | S1 | Switch on-on, SW DPDT | Distrelec |
| 1 | S2 | Taster, B3U | Farnell |
| 1 | X1 | ICSP-Programmierschnittstelle, 5-Pin Stiftleiste | Distrelec |
| 1 | X2 | Micro-USB Buchse, USB AB | Distrelec |

**Unterseite**



**Stückliste Unterseite**

|  |  |  |  |
| --- | --- | --- | --- |
| Menge | Position | Bezeichnung | Händler |
| 2 | G1, G2 | 16mm Knopfbatterie 3V | Farnell |
| 1 | X2 | Micro-USB Buchse, USB AB | Distrelec |

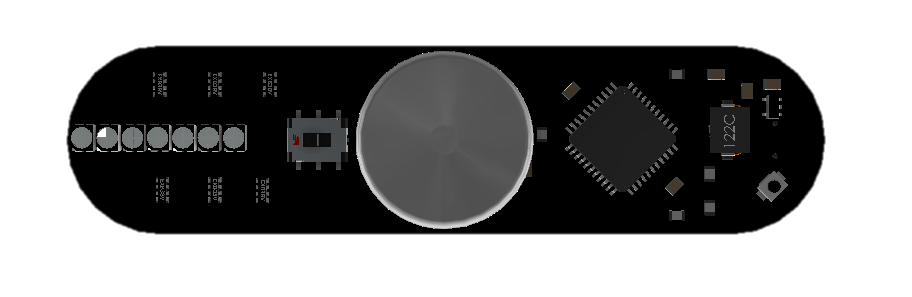
* 1. Bedienungsanleitung

**Text anzeigen**

Um einen Text anzeigen zu können muss nur der Schalter in Richtung Mitte gestossen werden und der Spinner gedreht werden.

**Farbe wechseln**

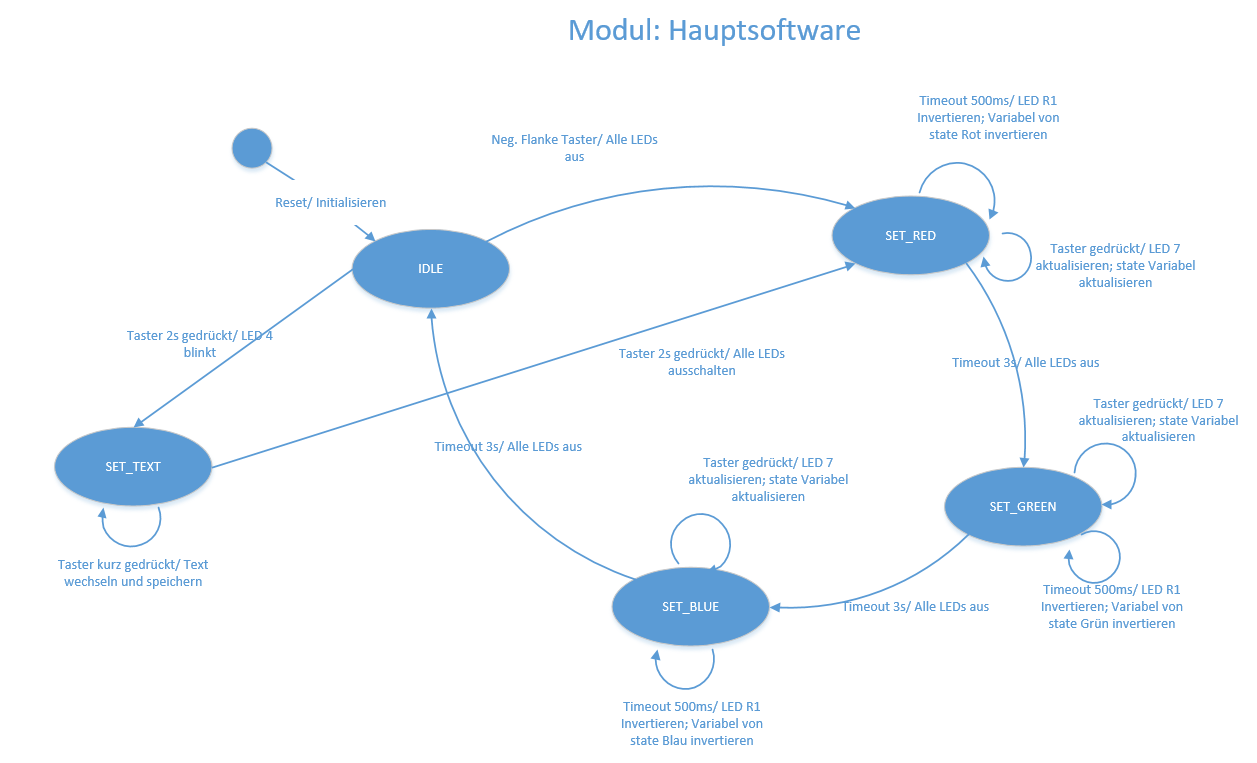
Um eine Farbe zu wechseln muss zuerst der Taster kurz gedrückt werden. Anschliessend blinkt die 1. LED in den Farben Rot, Grün und Blau. Bei jeder Farbe kann mit einem kurzen Tastendruck entschieden werden ob die Farbe leuchten soll oder nicht (Farbmischung ist möglich). Standartmässig leuchten die LEDs in Grün.



Taster

Power-Schalter OFF/ON

* 1. Zustandsdiagramm Hauptsoftware



* 1. Zeitplan

