Dokumentation Mikroprozessortechnik Simon Says

Inhalt

[Hintergrund des Projekts 3](#_Toc184837905)

[Machbarkeitsanalyse 3](#_Toc184837906)

[Terminplan 3](#_Toc184837907)

[Pflichtenheft 4](#_Toc184837908)

[Mögliche Erweiterungen 6](#_Toc184837909)

[Budget 6](#_Toc184837910)

[Softwareflussdiagramm 6](#_Toc184837911)

[Finalversion 7](#_Toc184837912)

[Hardware Layout 9](#_Toc184837913)

[Materialbeschaffung 9](#_Toc184837914)

[Software Regeln 10](#_Toc184837915)

[Software Dokumentation 10](#_Toc184837916)

[Variablen 10](#_Toc184837917)

[Funktionen 10](#_Toc184837918)

[Libraries 12](#_Toc184837919)

[Übrige 3rd-party Fragmente 12](#_Toc184837920)

[Sounds 12](#_Toc184837921)

[Button Funktion 12](#_Toc184837922)

[Versionierung 13](#_Toc184837923)

[Inbetriebnahme & Testprotokoll 13](#_Toc184837924)

[Bedienungsanleitung 13](#_Toc184837925)

[Herausforderungen 14](#_Toc184837926)

[Buttonreaktion 14](#_Toc184837927)

[Spielmodus / Einstellungen 14](#_Toc184837928)

[Gehäusebau 14](#_Toc184837929)

[Stückliste 14](#_Toc184837930)

[Arbeitsrapport 15](#_Toc184837931)

[Fazit 15](#_Toc184837932)

[Quellen 16](#_Toc184837933)

[Literaturverzeichnis 16](#_Toc184837934)

[Anhang 16](#_Toc184837935)

[Anhang 1: Terminplan Final 17](#_Toc184837936)

# Hintergrund des Projekts

Während meinem Lehrgang an der schweizerischen Fachschule TEKO Zürich müssen wir im Fach Mikroprozessortechnik eine Projektaufgabe absolvieren. Es geht darum ein einfaches Projekt mit Mikroprozessoren umzusetzen. Dazu kann beispielsweise Arduino, ESP32 oder RasperryPi benutzt werden. Da diese Aufgabe auch eine Vorbereitung auf die kommende Diplomarbeit sein soll muss sie als vollwertiges Projekt inklusive Projektplanung aufgezogen werden.

# Machbarkeitsanalyse

Nach einiger Recherche und Ideenfindung habe ich mich dazu entschieden ein **Simon Says** Spiel zu programmieren.

**Simon Says** ist ein Spiel, bei dem der Spieler eine immer länger werdende Sequenz aus farbigen Lichtern und dazu passenden Tönen wiederholen muss. Das Gerät zeigt eine Kombination an, die sich der Spieler merken und in der gleichen Reihenfolge durch Drücken der entsprechenden Tasten wiedergeben muss. Mit jeder Runde wird die Sequenz länger und schwieriger, bis ein Fehler gemacht wird. Ziel ist es, so viele Runden wie möglich korrekt zu meistern.

Solche Spiele sind bekannt und es gibt etliche Beispiele im Internet. Mein Ziel ist es aber dieses Spiel selbst zu programmieren und keine fremden Codeblocks zu verwenden. So musste ich im Vorfeld testen, wie ich den Arduino genau ansprechen kann und ob meine Idee überhaupt funktioniert. Die für mich wichtigste Funktion zur Spiellogik ist das Befüllen und Vergleichen von Arrays. Wie dies funktioniert, wollte ich im Vorfeld austesten. So ist schon ein grosser Codeschnipsel entstanden, auf dem ich später aufgebaut habe.

Ich kam auch in Absprache mit dem Dozenten zum Schluss, dass das Projekt umsetzbar ist und dem Anspruch der Schule genügt.

# Terminplan

Der Terminplan ist in mehrere Hauptphasen unterteilt. Darunter fallen die Organisation, die Entwicklung und die Dokumentation. Die einzelnen Arbeitsschritte sind den Hauptphasen unterstellt.

Da in diesem Ein-Mann-Projekt keine fixen Arbeitstage definiert sind, kann auch nicht genau prognostiziert werden, wann an der Umsetzung gearbeitet wird. Deshalb sind die meisten Arbeitspakete über einen Zeitraum von mehreren Tagen geplant. Obwohl sich Tätigkeiten überschneiden können, kann ein grundsätzlicher Ablauf der Entwicklung erkannt werden. Abhängigkeiten wurden im Terminplan nicht extra ausgewiesen, ergeben sich aber aus logischer Betrachtung. So kann zum Beispiel keine Hardware bestellt werden, wenn diese nicht zuvor definiert wurde.

Jedes Arbeitspaket verfügt über eine Aufwandabschätzung in Stunden. Dieser Aufwand soll im notierten Zeitraum geleistet werden. Zur Überprüfung der benötigten Zeitressourcen wird ein Arbeitsprotokoll erstellt. Die Aufwände aus dem Arbeitsprotokoll werden laufend in den Terminplan geschrieben, um so stets eine aktuelle Übersicht über den Stand des Projektes zu erhalten.

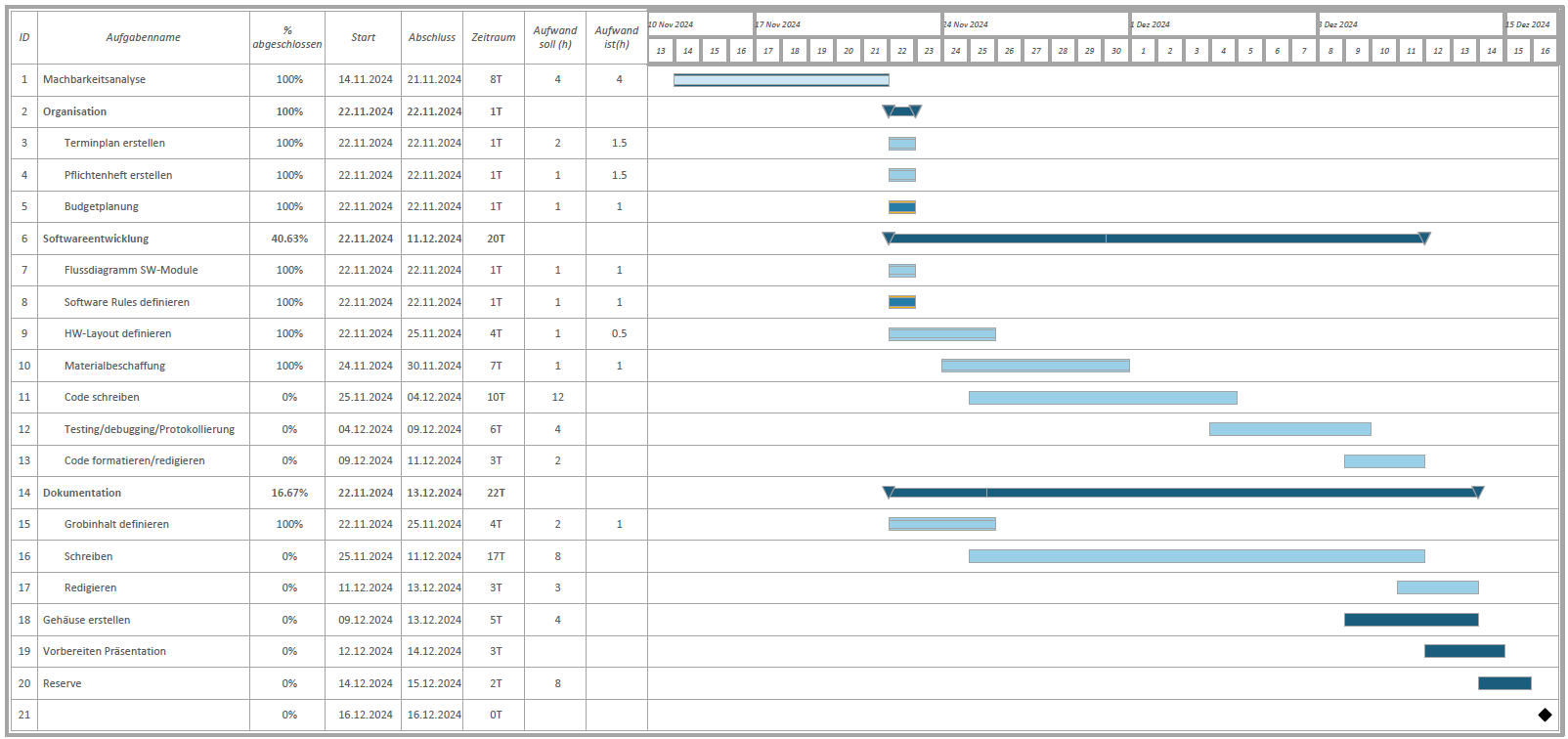


Abbildung 1: Terminplan stand 28.11.2024

<<Terminplan bei Projektabschluss>>

# Pflichtenheft

Aus dem nachfolgenden Pflichtenheft ist zu entnehmen, welche Funktionen bei der Planung des Projektes miteinberechnet wurden. Manche möglichen Erweiterungen, wie unten näher beschrieben, waren als Ideen schon zu diesem Zeitpunkt als Ideen vorhanden und sind als "optional" kategorisiert ins Pflichtenheft mit eingeflossen.

|  |
| --- |
| Zielbestimmung |
| Das Endziel ist ein funktionstüchtiges Minispiel, das bekannt ist unter dem Namen "Simon Says". Der Computer gibt dem Spieler optisch eine Abfolge an Farbenkombinationen vor, die der Spieler anschliessend wiederholen muss. Bei Spielbeginn startet die Sequenz bei einer Vorgabe. Schafft der Spieler es die Sequenz korrekt zu wiederholen steigt er ein Level auf und die Abfolge erhöht sich um eins. Schafft er es nicht startet das Spiel wieder bei einer Vorgabe. Computer und Spielereingaben werden optisch mittels farbiger LED und akustisch mittels Lautsprecherton quittiert. Zudem erfolgt eine optische und akustische Rückmeldung, wenn ein Level erreicht oder das Spiel verloren ist. |

|  |
| --- |
| Anwendungsbereich |
| Das Produkt ist in erster Linie ein Übungsprojekt welches an der höheren Fachschule als Projektarbeit dient. Das Endprodukt kann zum kurzweiligen und spielerischen Zeitvertreib genutzt werden. |

|  |
| --- |
| Zielgruppe |
| Alle Personen, bei denen der oben erwähnte Anwendungsbereich zutrifft. |

|  |
| --- |
| Betriebsbedingungen |
| Das Endprodukt kann von jedermann betrieben werden. Manipulationen an den elektrischen Bauteilen sind nur durch instruiertes Fachpersonal erlaubt. Kontakt mit flüssigen Substanzen ist nicht gestattet. Das Produkt darf nir mit der dafür vorgesehenen Stromversorgung benutzt werden. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Funktionen | | | | |
|  |  | Wichtigkeit | | |
| Gruppe | **Funktion** | **Hoch** | **Niedrig** | **Optional** |
| Ausgabe | LED leuchtet | x |  |  |
| Ausgabe | Buzzer tönt je nach Farbwahl unterschiedlich | x |  |  |
| Spielverhalten | Levelerhöhung funktioniert | x |  |  |
| Spielverhalten | Computer spielt Sequenz vor | x |  |  |
| Eingabe | Eingabe gesperrt, wenn Computer am Zug | x |  |  |
| Eingabe | System reagiert auf Knopfdruck mit Anzeige von LED und Ton von Buzzer | x |  |  |
| Spielverhalten | Startmelodie verfügbar |  | x |  |
| Spielverhalten | System erkennt Eingabefehler | x |  |  |
| Ausgabe | System zeigt Fehler optisch und akustisch an | x |  |  |
| Ausgabe | System zeigt Erfolge optisch und akustisch an | x |  |  |
| Spielverhalten | Spiel beginnt bei Fehler von vorne | x |  |  |
| Eingabe | schnelle Benutzereingabe möglich | x |  |  |
| Einstellungen | LED-Helligkeit kann angepasst werden |  |  | x |
| Einstellungen | Lautstärke kann angepasst werden |  |  | x |
| Einstellungen | verschiedene Schwierigkeitsstufen wählbar |  |  | x |
| Ausgabe | Display zeigt aktuelles Level an |  |  | x |
| Ausgabe | Display zeigt aktuellen Rekord an |  |  | x |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Qualität | | | | |
|  |  | Wichtigkeit | | |
| Gruppe | **Funktion** | **Hoch** | **Niedrig** | **Optional** |
| Programmcode | Für dritte verständlich |  | x |  |
| Dokumentation | Vorhanden | x |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Benutzungsoberfläche | | | | |
|  |  | Wichtigkeit | | |
| Gruppe | **Funktion** | **Hoch** | **Niedrig** | **Optional** |
| Eingabe | Buttons zur Eingabe vorhanden | x |  |  |
| Ansicht | System in Chassis verbaut |  |  | x |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nichtfunktionale Anforderungen | | | | |
|  |  | Wichtigkeit | | |
| Gruppe | **Funktion** | **Hoch** | **Niedrig** | **Optional** |
| Programmcode | Variablen extern definiert | x |  |  |
| Dokumentation | Die Dokumentation enthält eine Stückliste |  | x |  |
| Dokumentation | Das System wird live demonstriert | x |  |  |
| Programmcode | Versionisierung auf Github |  |  | x |

|  |  |  |
| --- | --- | --- |
| Technische Produktumgebung | | |
|  | | |
| Funktion | **Typ** | **Produkt** |
| Software | Entwicklungsumgebung | Arduino IDE |
| Software | Elektronische Schaltungen | Fritzing |
| Hardware | Mikrocontroller | Arduino UNO MEGA 2560 |
| Hardware | Elektronikbauteile | Gemäss Stückliste |

## Mögliche Erweiterungen

Da der Zeitrahmen begrenzt ist und ich bei der Aufgabe mehr Relevanz auf die Ausführung der organisatorischen Bestandteile gelegt habe, konnte ich einige Ideen nicht in die Umsetzung des Endproduktes miteinberechnen. So sind die möglichen Erweiterungen bestenfalls optional dazu entstanden, aber nicht teil des Projektumfanges:

* Einbau in ein Gehäuse
* Schwierigkeitsstufen wählbar
* Zwei-Spieler Modus
* Spielmodus Spieler gegen Spieler
* Displayintegration
* Einstellungsmöglichkeiten (LED Helligkeit, Lautstärke)
* Rekordanzeige
* Speichern der Einstellungswerte und Rekorde

# Budget

Das Budget kann bis auf einen kleinen Teil ausser Acht gelassen werden.

Alles Material, welches es zum Erstellen dieses Projektes benötigt ist im von der Teko Zürich zur Verfügung gestellten «Starter Kit für Arduino» vorhanden. Deshalb ist kein Budget für Materialkosten nötig.

Auch Arbeitsressourcen werden nicht budgetiert, da diese nicht verrechnet werden.

Einzige Kosten, welche berücksichtigt werden, sind diejenigen für das optionale Material, welches für die optische Aufwertung des Endproduktes benötigt werden. Das Material kann anhand der Stückliste nachvollzogen werden. Die Kosten für das Zusatzmaterial sind mit CHF 50.- budgetiert.

Nach Abschluss des Projektes konnte festgestellt werden, dass das Budget mit CHF 34.50 für alle zusätzlichen Bauteile und ohne Verbrauchsmaterial und Werkzeug eingehalten werden konnte.

# Softwareflussdiagramm

Mit dem Softwareflussdiagramm soll das enthaltene Programm auf einfache Art und Weise illustriert werden. Das erstellte und nachfolgend abgebildete Diagramm zeigt die zu Beginn des Projektes geplante Softwarearchitektur auf. Im Laufe des Projektes wurden diverse Zusatzfunktionen eingebaut, welche in diesem Bild nicht ersichtlich sind.

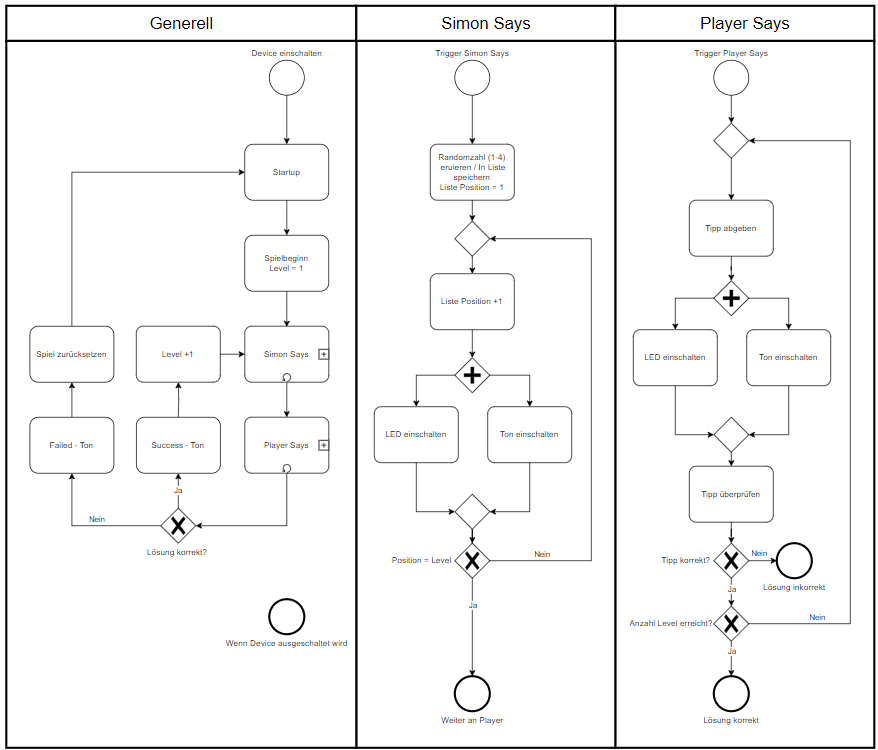


Abbildung 2: Softwareflussdiagramm V1.0

## Finalversion

Wegen der vielen Änderungen im Projektverlauf wurde das Softwareflussdiagramm nach Abschluss des Projektes noch einmal aktualisiert. Es sind einige Vereinfachungen der Darstellung eingeflossen. So wird nicht mehr die Schaltung jedes LED's oder jedes Tonsignals vermerkt. Vielmehr wird dies im Prozessschritt markiert.

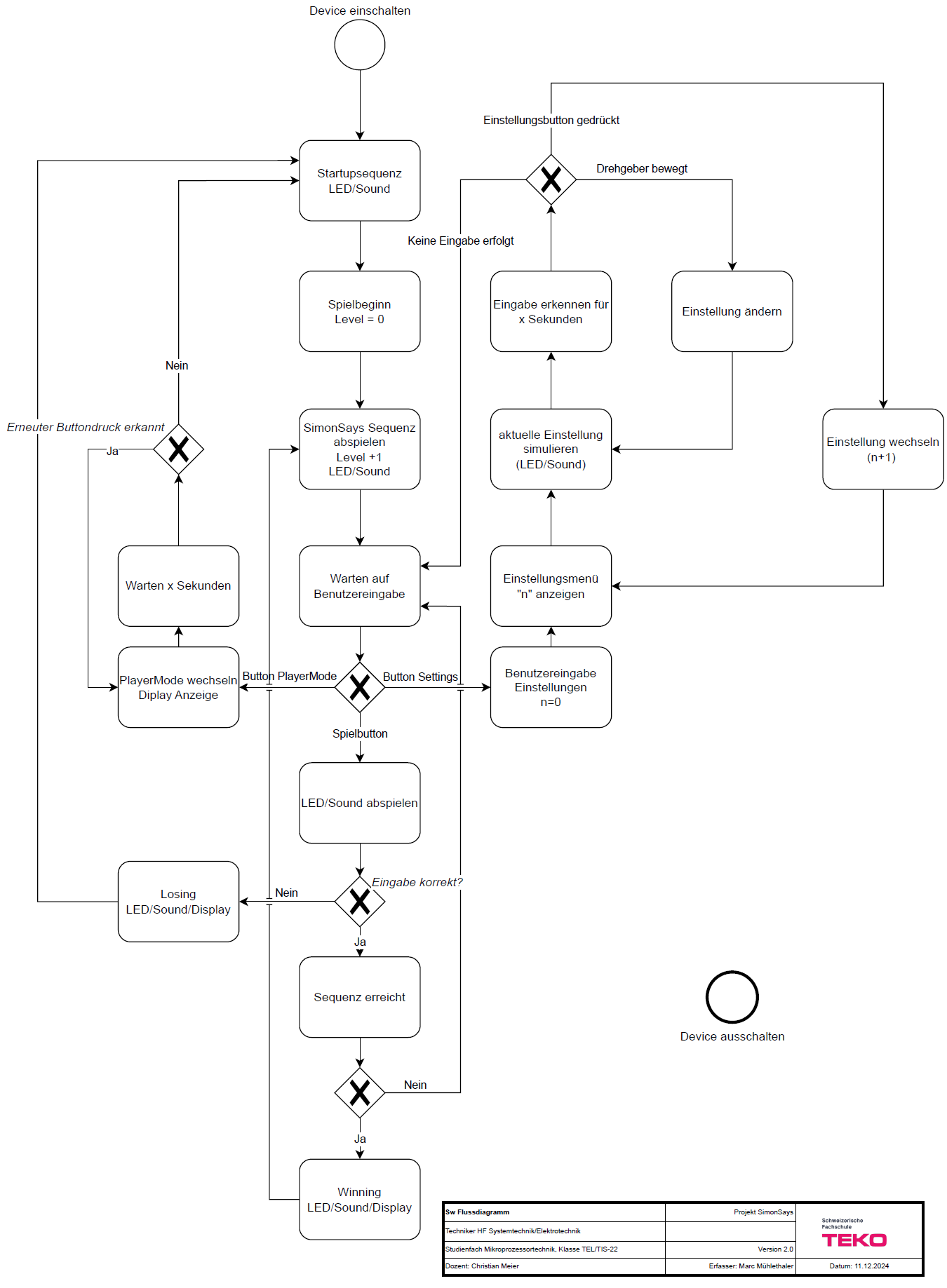
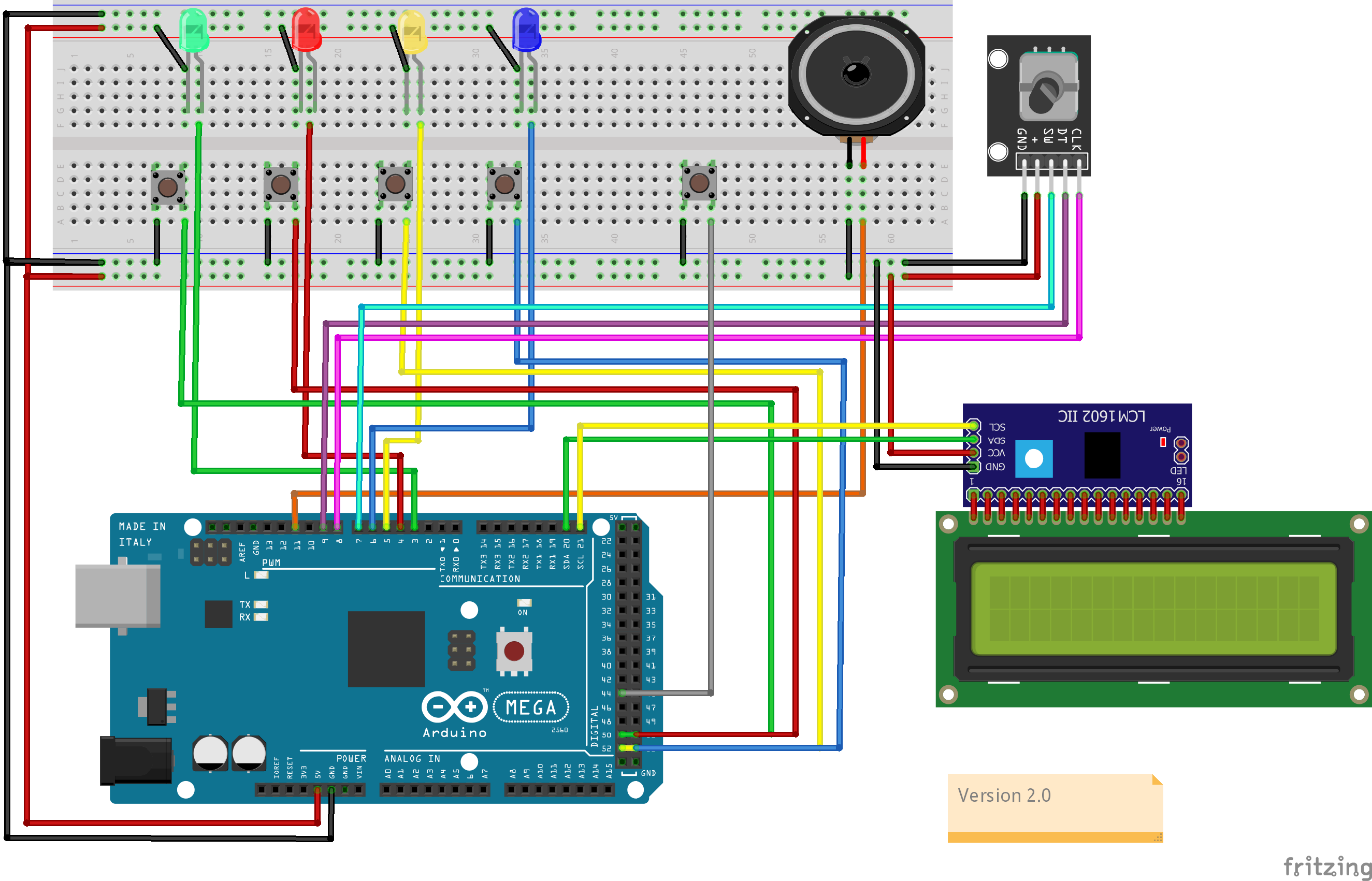


Abbildung 3: Softwareflussdiagramm V2.0

# Hardware Layout

Mit Hilfe vom Elektronikschema soll aufgezeigt werden wie die Bauteile untereinander verbunden werden müssen. Dies hilft zum einen die benötigte Hardware zu erfassen und auch im Fehlerfall eine effiziente Behebung zu gewährleisten.

Das nachfolgend aufgezeigte Hardware-Layout zeigt die Verkabelung des Endproduktes inklusive aller während dem Projekt hinzugefügten Bauteile auf. Der Übersichtlichkeit dienend wurde es in der Software "Fritzing" erstellt und mit einem generellen Breadboard umgesetzt. Die Endlösung verwendet zur Verbindung der einzelnen Komponenten andere Verbindungsmittel. Die Verschaltung der Elemente und PIN-Belegungen auf dem Arduino Mega 2560 entsprechen aber dem Original.



# Materialbeschaffung

Nachdem das Hardware Layout definiert wurde, konnte auch der Materialbedarf benannt werden.

Die Auswahl des Lieferanten erfolgte nach nachfolgenden Kriterien, welche durch Internetrecherchen beurteilt werden konnten:

* Verfügbarkeit der Artikel
* Lieferdauer
* Preis
* Reputation
* Lieferkosten
* Lagerhaltung
* Alle Artikel aus einer Hand

Das Ziel war keine Artikel direkt aus dem asiatischen Raum zu importieren, sondern auf Waren zu setzen, welche sich bereits in der Schweiz befinden. So konnte die Lieferfrist klein gehalten werden. Natürlich mussten die Artikel am Lager verfügbar sein. Da das Budget bereits festgelegt wurde spielte auch der Preis eine wichtige Rolle. Unbekannte Verkäufer wurden nicht berücksichtigt.

Die Materialbeschaffung erfolgte schlussendlich wegen neuer Ideen während dem Projekt in zwei Schritten. Es wurden pro Bestellung je ein Lieferant berücksichtigt. Die bestellten Artikel und deren Bezugsorte können in der Stückliste nachvollzogen werden.

# Software Regeln

Softwareregeln dienen dazu Ordnung und Konsistenz sicherzustellen. Sie umfassen Vorgaben und Richtlinien, die bei der Entwicklung und Wartung von Software einzuhalten sind. Durch Vorgaben kann zum Beispiel die Teamarbeit, Kompatibilität, Fehlervermeidung und Wartbarkeit erhöht und somit die Qualität verbessert werden.

Ich habe folgende Regeln definiert:

**Codestandards:**

* Variablen in externem File (ausser Schleifenvariablen)
* Variablen wenn möglich gruppieren
* Variablen im camelCase Style (beginnen mit Kleinbuchstaben, haben Grossbuchstaben in der Mitte) und bezeichnen den Nutzen der Variable sogut wie möglich.
* Beispiel: «buttonPin» definiert den Pin für einen Button.
* Kommentare: Klar und prägnant die Codeabsicht dokumentieren
* Wiederholende Programmschritte werden in Funktionen geschrieben

**Versionskontrolle:**

* Im .ino-File wird nach jedem grösseren Arbeitsschritt die Version erhöht. Zudem wird notiert welche Änderungen vorgenommen wurden.
* GIT: Nach jedem Arbeitsschritt erfolgt ein git push.

**Sicherheit:**

* Code kann ohne zusätzliche Kontrolle deployed werden

# Software Dokumentation

## Variablen

Alle verwendeten Variablen sind im File <arduino\_config.h> abgespeichert. Dort ist auch deren Verwendungszweck auch genauer beschrieben

## Funktionen

Um den Aufbau der Software besser verstehen zu können sind anschliessend die wichtigsten Funktionen und deren Verwendungszweck genauer beschrieben.

|  |
| --- |
| setup () |
| Anfangswerte (Variablen, Pins und Bibliotheken) werden initialisiert. |

|  |
| --- |
| loop () |
| Die funktion loop ist eine Endlosschleife, die nach jedem Durchlauf erneut aufgerufen wird.  Sie wird verwendet um das Spiel zu starten, die Computervorgabe abzuspielen und dann auf die Benutzereingabe zu warten. Je nach dem ob diese korrekt ist oder nicht wird angezeigt, ob das Level gewonnen oder verloren wurde. Danach wird entweder mit Erhöhung der Sequenz weitergefahren oder bei null begonnen. |

|  |
| --- |
| printLCD(String lcdLine1, String lcdLine2 = "none") |
| Nimmt die auf dem Display auszugebenden Strings auf und printet sie. Wenn das Spiel gestartet ist und für die zweite Linie keine Eingabe erfolgt werden die Variablen Level und Rekord ausgegeben. |

|  |
| --- |
| startupSeq() |
| Gibt eine Ton und LED Abfolge als Startsequenz aus |

|  |
| --- |
| simonPlays() |
| Die Funktion wird in jedem neuen Level durchschritten und speichert pro Durchlauf einen Zufallswert für die nächste Vorgabe der Sequenz in ein Array. Die gesamte Sequenz wird danach abgespielt. Dies ist die Spielsequenz die es danach vom Spieler zu lösen gilt. |

|  |
| --- |
| checkUserSequence() |
| In dieser Funktion wird über die verschachtelte Funktion readButtons() geprüft ob die Benutzereingabe mit dem Array das in der Funktion simonPlays() generiert wurde übereinstimmt. Wenn ja wird sofern noch nicht die ganze Sequenz nachgespielt wurde nochmals eine Eingabe abgefragt. Wenn eine fehlerhafte Eingabe gemacht wurde wird das Spiel mit der Funktion losing() beendet. |

|  |
| --- |
| readButtons() |
| Es wird abgefragt ob und wenn ja welcher Button gedrückt wurde. Für die beiden Einstellungsbuttons sind weitere Funktionen verschachtelt. |

|  |
| --- |
| settings(buttonPin) |
| In der Funktion settings() wird unterschieden ob der MultiPlayer- oder der Einstellungsbutton gedrückt wurde.  Multiplayer: mittels der Funktion changePlayerMode() wird der Spielmodus geändert. Der Button bleibt für gewisse Zeit aktiv und kann geändert werden bevor das Spiel neu gestartet wird.  Einstellungen: mittels der Funktion rotary() können Einstellungen der Hardware vorgenommen werden. |

|  |
| --- |
| changePlayerMode() |
| Hier wird der Zweispielermodus aktiviert oder deaktiviert. |

|  |
| --- |
| rotary() |
| In dieser speziellen Funktion können Einstellungen der Hardware vorgenommen werden. Dies beinhaltet Helligkeit der LED’s, Lautstärke der akustischen Signale und die Spielgeschwindigkeit. Pro Einstellung wird der neu gesetzte Wert jeweils simuliert und auf dem Display in Prozent-Schritten ausgegeben. |

|  |
| --- |
| toneLED(int button) |
| Wann immer möglich wird dieses Codefragment benutzt um LED und Töne zu schalten |

|  |
| --- |
| losing() |
| Notifikation via LED, Lautsprecher und Display über eine falsche Eingabe, die das Spiel anschliessend zurücksetzt. |

|  |
| --- |
| winning() |
| Notifikation via LED, Lautsprecher und Display über eine korrekte Eingabe der gesamten Sequenz. Das Level wird um 1 erhöht. |

## Libraries

Um dieses Projekt umzusetzen wurden folgende Libraries zu Hilfe genommen.

|  |
| --- |
| LiquidCrystal\_I2C by Frank de Brabander |
| Die Bibliothek ermöglicht die Steuerung von I2C-Displays mit Funktionen, die denen der LiquidCrystal-Bibliothek sehr ähnlich sind. |

|  |
| --- |
| RotaryEncoder by Matthias Hertel |
| Um den verwendeten Drehgeber KY-040 abzufragen wurde diese Library verwendet. |

|  |
| --- |
| toneAC by Tim Eckel |
| Um das Feature der Lautstärkeneinstellung Software massig zu implementieren wurde toneAC benutzt. |

## Übrige 3rd-party Fragmente

### Sounds

Die Noten und Spielabfolgen für die akustischen Rückmeldungen wurden teilweise von Fazie Romadhona und seinem Simon Says Game übernommen. Insbesondere die Startabfolge, der Sound beim Meistern eines Levels und der Game-Over Ton habe ich kopiert.

### Button Funktion

Mein erster Entwurf der Buttonfunktion war schlecht spielbar, da die Reaktionsgeschwindigkeit zu langsam ist. Ich habe die Idee von Urish übernommen, der auf der Webseite [www.Wokwi.com](http://www.Wokwi.com) sein Simon Says Game präsentiert. Dort wird in einer Schleife nur auf eine Benutzereingabe gewartet, was die Reaktionsgeschwindigkeit massiv verbessert. Meine Funktionen habe ich darum herum implementiert.

# [Ein Bild, das Text, Screenshot, Schrift, Grafiken enthält. Automatisch generierte Beschreibung](https://github.com/tekoler/Mikroprozessortechnik_SimonSays/commit/f6a4ed1bcb49a129313a702e81e14b1ebef3770c)Versionierung

Da ich noch nicht viel Erfahrung bezüglich Softwareentwicklung habe, kenne ich auch den Prozess der Versionierung nicht wirklich. Ich habe dieses Projekt aber zum Anlass genommen meine ersten Schritte auf Github zu tätigen. Da ich aber nicht sehr viel Zeit zum Einlesen in dieses Thema hatte, habe ich mich auf die Grundfunktionen wie Hinzufügen und ändern von Dateien beschränkt. Jede grössere Änderung des Codes oder anderer relevanter Dateien hat ein Update auf Github erfahren mit den entsprechenden Kommentaren im commit.

# Inbetriebnahme & Testprotokoll

# Bedienungsanleitung

|  |  |
| --- | --- |
| **Bedien- und Anzeigeelemente:**  1: Spielknöpfe  2: LED Anzeige  3: Display  4: Multiplayer Modus Knopf  5: Settings  6: Stromanschluss  7: Wartungsanschluss | **Bedienung**  **Einschalten:**  Einschalten: Gerät mit Stromversorgung verbinden. Hierfür entweder eine 5V Batterie oder das Netzteil verwenden  **Ziel des Spiels:**  Der Computer gibt eine Reihe an nachzuspielenden Farb- und Tonfolgen vor. Das Ziel des Spiels ist es die Sequenz mit Hilfe der Spielknöpfe nachzuspielen. Hat man eine Sequenz korrekt gespielt erhöht sich die Länge der Folge um eins. Wurde eine falsche Eingabe gemacht verliert man und das Spiel beginnt von neuem.  **Multiplayer Modus:**  Durch Drücken des Multiplayer Modus Buttons kann zwischen Ein- und Zweispieler Modus gewechselt werden. Das Gerät wechselt in den aktuell angezeigten Modus und startet neu so bald 5 Sekunden keine Eingabe mehr erfolgt.  **Settings-Button:**  Durch drücken des Einstellungsknopf kann die Helligkeit der LED, die Lautstärke des Lautsprechers und die Spielgeschwindigkeit angepasst werden. Drücken des Knopfes ändert die Einstellungsmöglichkeit, drehen des Knopfes den jeweiligen Wert. |

# Herausforderungen

## Buttonreaktion

Leider war mit dem ursprünglich geplanten Code die Usereingabe nicht allzu schnell möglich. Gerade die ersten kurzen Sequenzen werden bei diesem Spiel schnell eingegeben. Dies war in den ersten Versionen wegen des überladenen loops nicht möglich.

Die Eingabegeschwindigkeit musste also verbessert werden. Aus Zeitgründen wurde die bereits erwähnte Buttonfunktion von Urish adaptiert.

## Spielmodus / Einstellungen

Die bei der Buttonreaktion eingeführten Änderungen brachten zwar eine Verbesserung der Eingabeverzögerung, dann aber ein anders Problem zum Vorschein. Das Spiel sollte mit einem 2-Spielermodus attraktiver gestalten werden. Zudem sollte der Benutzer Einstellungen mithilfe eines Drehgebers verändern können. Diese Einstellung sollte zu jedem Zeitpunkt möglich sein, wenn nicht gerade der Computer etwas vorspielt. Dies führte dazu, dass im Code sehr viele Änderungen vorgenommen werden mussten, vor allem um die zeitkritische Abfrage eines erneuten Buttonklicks zu bewältigen.

Auch die Abfrage des Drehgebers war kompliziert umzusetzen. Einstellungen sollten ohne grosse Latenz funktionieren, die gemachten Änderungen anzeigen und simulieren und nach einem Timeout, wenn nichts eingegeben wurde, wieder zum Spiel zurückkehren.

Dies konnte aus Mangel von Programmiererfahrung und Zeitgründen nur durch eine etwas unschöne und wohl komplizierte Codestruktur umgesetzt werden. Das Endergebnis allerdings funktioniert.

## Gehäusebau

Der Bau des Gehäuses mit Haushaltsüblichen Werkzeugen war eine Herausforderung. Im speziellen die Ausschnitte für das Display und die Arduino-Anschlüsse sind nicht optimal gelungen. Dies wäre mit einer speziell dafür geeigneten Maschine auf CNC-Basis zu optimieren.

# Stückliste

Anhand folgender Stückliste kann eingesehen werden, welches Material für die Umsetzung des Projektes benötigt wurde. Die für das Spiel benötigten Bauteile wurden teilweise dem «Starter Kit für Arduino» von Funduino entnommen. Um das Endresultat etwas ansehnlicher zu gestalten kann es zum Beispiel in ein Gehäuse montiert werden. Dazu benötigt man im besten Fall besser geeignete Buttons und Fassungen für die Elektrobauteile. Es folgt die gesamte Stückliste des fertigen Projektes:

|  |  |  |  |
| --- | --- | --- | --- |
| Anzahl | Artikel | Bezugsort | Preis (CHF) |
| 1 | Arduino MEGA 2560 | Starterkit Funduino | k.A. |
| 1 | Breadboard | Starterkit Funduino | k.A. |
| 1 | LED 5mm blau | Starterkit Funduino | k.A. |
| 1 | LED 5mm rot | Starterkit Funduino | k.A. |
| 1 | LED 5mm grün | Starterkit Funduino | k.A. |
| 1 | LED 5mm gelb | Starterkit Funduino | k.A. |
| 4 | Taster | Starterkit Funduino | k.A. |
| 4 | Widerstand 1kOhm | Starterkit Funduino | k.A. |
| 1 | LCD Display | Starterkit Funduino | k.A. |
| einige | Breadboardkabel | Starterkit Funduino | k.A. |
| 1 | EURO Kunstoff-Gehäuse Mittel Schwarz | Conrad | 8.75 |
| 1 | Arcade Button Micro grün | Conrad | 2.20 |
| 1 | Arcade Button Micro blau | Conrad | 2.25 |
| 1 | Arcade Button Micro rot | Conrad | 2.85 |
| 1 | Arcade Button Micro gelb | Conrad | 2.85 |
| 4 | LED Fassung Innenreflektor Chrom 5mm | Conrad | 6.60 |
| 1 | KY-040 Rotary Encoder Drehgeber | Bastelgarage.ch | 3.90 |
| 1 | Mini Lautsprecher 8Ohm 0.25W | Bastelgarage.ch | 1.90 |
| 1 | Einbau Drucktaster schwarz 7mm 2P | Bastelgarage.ch | 1.40 |
|  |  |  |  |

Um alle Elemente zu verbauen und das Gehäuse zu präparieren wurde zudem folgendes Werkzeug und Verbrauchsmaterial verwendet:

* Bohrmaschine mit Bohrer (6mm, 7mm, 8mm, 20mm)
* Dekoupiersäge
* Schraubenzieher / Montageschrauben
* Lötkolben / Lötzinn
* Klebstoff

# Arbeitsrapport

# Fazit

Das einfache Spiel ist in der Basis simpel zu umzusetzen. Wenn man sich dann aber an Zeitangaben halten soll, die Arbeit im Projektmodus umsetzen möchte und während dem Verlauf der Umsetzung die Ansprüche an das Endprodukt erhöht, bedarf es schon ein bisschen mehr Aufwand.

Wenn man das fertige Spiel betrachtet und jemandem zum Spielen überlässt, kann man sich gar nicht vorstellen, wie viel Zeit in die Entwicklung der wenigen Features gesteckt wurde.

Ich konnte wieder viel neues dazulernen. Hier möchte ich besonders die Abwicklungsschritte eines Projektes, die Programmierung von Mikroprozessoren aber auch technische Details der verbauten Hardwarebestandteile benennen.

Würde ich das Projekt nochmals von vorne beginnen, müsste ich mir dazu Gedanken machen, ob ich wirklich so viel Zeit investieren möchte oder ich mich doch nicht lieber an das zu beginn erstellte Pflichtenheft halten sollte. Die wenigen Zusatzfunktionen kosteten enormen Entwicklungs- und Testaufwand.

Ich bin mit dem fertigen Projekt aber sehr zufrieden und bin froh, die Extrameile gegangen zu sein. Ich habe grösstenteils selbstständig ein funktionierendes Spiel entwickelt, das in einer ansehnlichen Form daherkommt und ich meiner Familie und Freunden zum Bespielen überlassen kann.

# Quellen

# Literaturverzeichnis

Romadhona, F. (2014). *Instructables - Arduino Simple Simon Says Game*. Von https://www.instructables.com/Arduino-Simple-Simon-Says-Game/ abgerufen

Urish, W. (kein Datum). *Wokwi simon-with-score*. Von https://wokwi.com/projects/328451800839488084 abgerufen

# Anhang

## Anhang 1: Terminplan Final