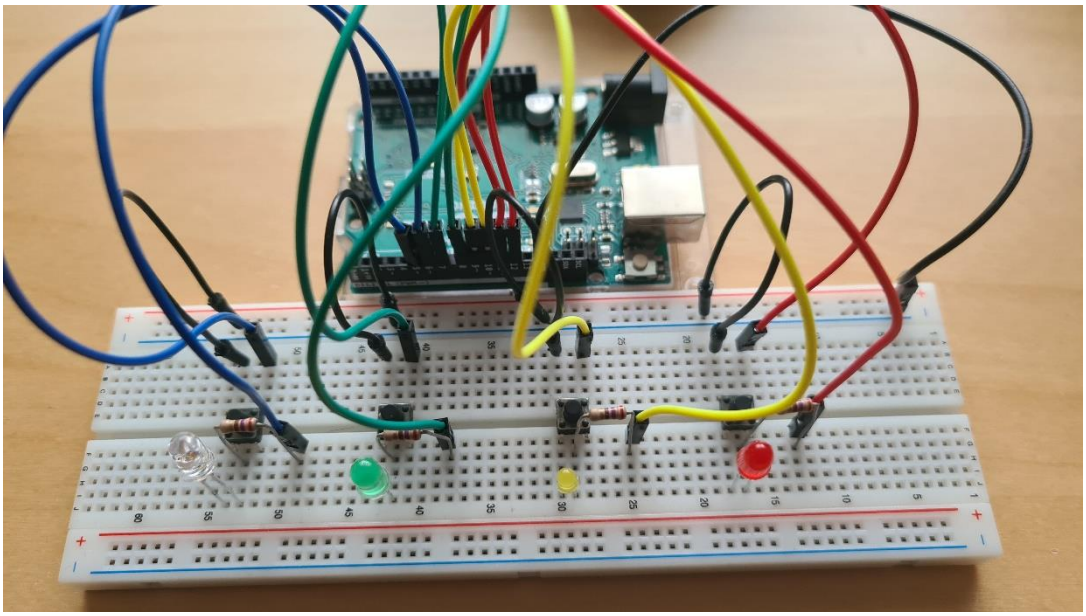
Entwicklungsdokumentation

Projektaufgabe aus dem 3. Semester
FH Wedel WS 2021

Projektaufgabe für den Fachbereich „eingebettete Software“
mit dem Ziel einen Gedächtnis-Trainer zu entwickeln



Kevin Feht
Rollberg 11
22880 Wedel
STEC 105106

Fachhochschule Wedel
Smart-Technology
WS 2021

Abgabetermin: 10.01.2022
Projektgeber: Prof. Dr. Ulrich Hoffmann

Inhaltsverzeichnis

Einleitung

Aufgabenvorstellung

Die vorgegebene Aufgabe war es, ein Memory Game zu bauen und zu programmieren. Zusätzlich musste eine Dokumentation und eine Bau-/ Bedienungsanleitung erstellt werden. Der Hauptfokus dieses Projektes sollte dabei auf der Dokumentation liegen. Welche Software und Hardware verwendet werden sollten, wurde nicht vorgegeben.
Die folgende Entwicklungsdokumentation enthält die von mir durchgeführten Schritte zum Projektabschluss, sowie alle notwendigen Informationen, um dieses Projekt selber durchzuführen.

Projekterläuterung

Ein Gedächtnistrainer soll das tun, was der Name schon sagt. Dem menschlichen Gehirn einen Anreiz geben, bestimmte Dinge besser zu behalten. In diesem Fall eine Reihenfolge von Farben in Form von LEDs. Nach einer Startsequenz leuchtet jede gespielte Runde eine neue, zufällige LED, die der Spieler sich merken muss. Daraufhin müssen die den LEDs zugehörigen Taster in der richtigen Reihenfolge bestätigt werden. Hier wird dem Spieler geholfen, da nach jedem erfolgreichen Betätigen der Knöpfe, und somit auch der richtigen Reihenfolge, die gesamte Sequenz noch einmal abgespielt wird mit einer neuen, zufälligen Farbe am Ende. Wenn der Spieler aber einen falschen Knopf und somit die falschen Farbe betätigt, dann gilt die Runde als gescheitert, was anhand einer Beendigungssequenz dargestellt wird.
Das Spiel kann dann, oder auch jederzeit während des Trainings, mit dem Betätigen des Reset-Knopfes auf dem Arduino neugestartet werden. Beim Beenden des Trainings muss nichts beachtet werden und das USB-Kabel kann einfach vom Arduino, oder der Stromquelle getrennt werden.
Diese Entwicklungsdokumentation beinhaltet alle Software und Hardware Informationen, einzelne durchgeführte Schritte, sowie mögliche Fehlerquellen und -behebungen.

Alle zum Projekt gehörenden Dateien, Bilder und Programme, sowie diese Entwicklerdokumentation und das Benutzerhandbuch sind in meinem GitHub Repository[1] zu finden.

Beim Nachbau ist lediglich zu beachten, dass man die Verkabelung wie in der Abbildung einhält. Falls man die Anzahl oder die Arduino-PINs verändert, dann müssen diese Informationen im Ersten Teil des Codes angepasst werden.

[1] https://github.com/Kvikne/MemoryGame

Hardware

Bauteile

Bei diesem Projekt habe ich mich dazu entschieden folgende Bauteile zu verwenden:

Bauteil	Verwendungszweck	Stückzahl
Abbildung 4: Arduino Uno Rev. 3[2]	Zur Stromversorgung und Programmierung	1
LED (Blau, Grün, Gelb, Rot)[3]	Zum Zeigen der Reihenfolge	4
Push-Taster (Knopf)[4]	Zur Eingabe der Reihenfolge	4
Widerstand[5]	Je 270Ω, zum Schutz der LEDs	4
Jumper Kabel[6]	Zum Verbinden der Bauteile	13
Laptop/PC	Zur Stromversorgung Zum Programmieren mit Arduino IDE	1

Tabelle 1: Bauteile Übersicht

Zusätzliche Hardware ist in der Theorie nicht nötig. Weitere LEDs, Taster, Widerstände und Jumper Kabel können aber jederzeit zusätzlich angeschlossen werden, falls man ein Memory Game mit mehr Herausforderung wünscht. Andererseits kann dieses Spiel aber auch mit nur 2 LEDs und den dazugehörigen Komponenten gespielt werden. Nur eine LED macht wenig Sinn, da man sich ansonsten nur merken müsste, wie häufig diese eine LED pro Sequenz aufgeleuchtet hat.

Die Stückzahl der einzelnen Komponenten ist also optional, welcher Arduino verwendet wird ebenso. Worauf man aber achten muss ist die Anzahl der verfügbaren Pins auf dem verwendetem Arduino. Je nach Auswahl des Arduino ist man also limitiert bei der Anzahl der verwendbaren Bauteile.

[2] Abbildung 4
[3] Abbildung 3
[4] Abbildung 5
[5] Abbildung 6
[6] Abbildung 7

Begründung der Auswahl der Bauteile

Alle verwendeten Bauteile stammen aus dem Lagerbestand der FH Wedel. Lediglich der Laptop stammt aus dem Eigenbedarf. Ich habe mich also bei der Auswahl der verwendeten Bauteile nach dem gerichtet, was bereits vorhanden war. Die Finale Auswahl an Bauteilen beruht auf vorherigen Versionen und verschiedenen Tests, sowohl beim Aufbau, als auch bei der Anzahl der verwendeten Bauteile. Anfänglich hatte ich noch mit mehr Widerständen experimentiert und bin dann aber zu dem Entschluss gekommen, dass ein Widerstand pro Stromkreislauf mit vorhandener LED vollkommen ausreicht. Ein Widerstand mit 270 Ω genügt, um pro Stromkreis jeweils eine LED und den dazugehörigen Push-Taster abzusichern. Es ist nicht notwendig jeden Taster zusätzlich mit einem Widerstand abzusichern, da diese lediglich erkennen, wenn ein geschlossener Stromkreis vorhanden ist. Nämlich beim Betätigen des Tasters. Zusätzlich bietet der Arduino UNO auch noch die Möglichkeit per internen PULLUP Widerständen diese Stromkreise abzusichern.
Diese Entscheidung macht das Programmieren der Software und dessen Struktur weniger komplex.

Verkabelung

Die Verkabelung bei meinem Aufbau sieht wie folgt aus:

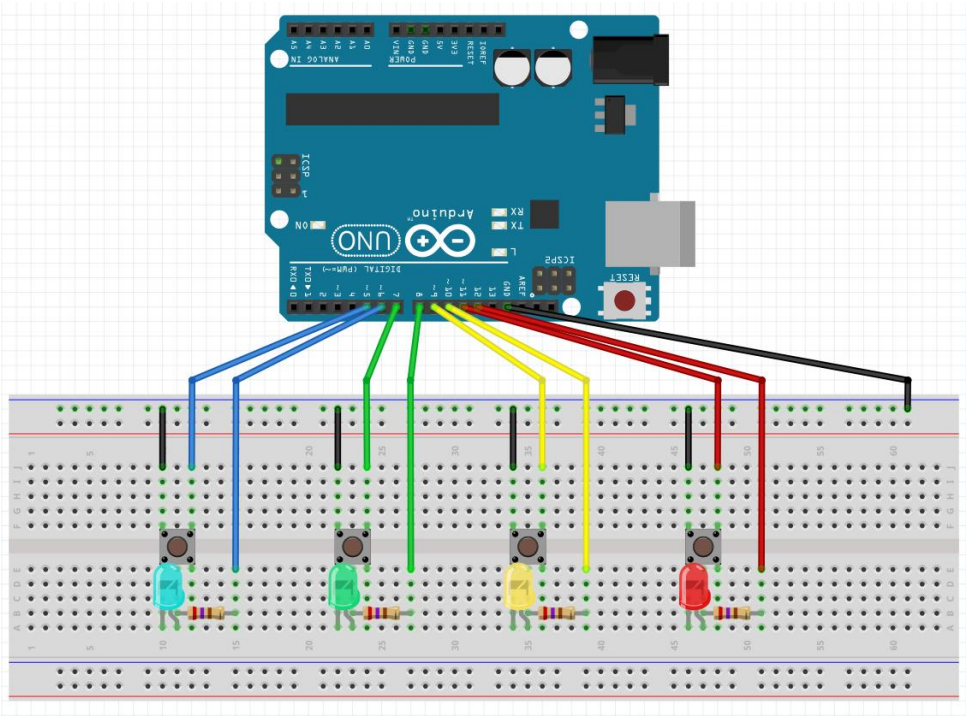


Abbildung 1: Verkabelung Arduino

Begründung der Verkabelung

Ich habe mich für diese Verkabelung auf Grund der Funktionsweise der Arduino Pins entscheiden. Jeder Push-Taster bildet einen Stromkreis, welcher dem Erkennen der Spielereingabe dient. Jedes Mal, wenn der Spieler einen Taster drückt, wird zwischen dem zu diesem Taster gehörenden Pin auf dem Arduino und der Erdung, welche durch die Verbindung zum GND Pin auf dem Arduino besteht, ein geschlossener Stromkreis erkannt. Dieser wird in der Software abgefangen, um zu erkennen, ob ein Taster betätigt wurden ist.
Die Verbindungen zu Ground auf dem Breadboard (4 Stück), welche in der Abbildung mit der Farbe Schwarz markiert sind, dienen jeweils zwei Stromkreisen. Einmal dem Stromkreis der LED und einmal dem Stromkreis des Tasters.
Jeder Stromkreis besteht zwischen einem Pin des Arduino und dem GND Pin. Alle Stromkreise sind in der Abbildung farblich voneinander abgetrennt.

Programmierung

Übersicht

Das gesamte Programmieren erfolgt in der Arduino IDE, welche extra von Arduino für ihre Produkte konzipiert wurden ist. Die Programmiersprache der IDE basiert auf C++. Alle Tests, sowie die finale Software sind im GitHub Repository komplett hinterlegt.

Das Programm funktioniert wie folgt:
Als erstes werden alle benötigten Variablen initialisiert, sowie Pins der LEDs und der Taster und die jeweilige Anzahl dieser verwendeten Bauteile.

Dann läuft das Setup, in dem die Taster und LEDs auf die zugehörigen Pins programmiert werden. Ebenfalls wird hier der Zufallsgenerator (randomSeed) zurückgesetzt, welcher später per random() Funktion eine zufällige LED auswählt.

Im Anschluss durchläuft das Programm dann den Hauptteil in einer Schleife. In dieser Schleife werden mehrere selbstdefinierte Funktionen aufgerufen. Diese Funktionen bilden das Spiel.

Folgende Funktionen existieren:

blinkOnce	lässt eine vorgegebene LED einmal aufblinken.
butWait	wartet, bis der zur LED gehörige Taster betätigt wird.
startSeq	lässt eine Startsequenz von LEDs aufblinken, die dem Spieler zeigt, dass gespielt werden kann.
wrongSeq	zeigt dem Spieler an, dass ein falscher Taster betätigt wurde und somit das Spiel zu Ende ist.
resetFunc	beendet das Spiel und startet das Programm neu. Diese Funktion muss jedoch im Setup-Teil definiert sein.

Tabelle 2: Funktionen Übersicht

Entwicklung

Bei der Softwarenentwicklung habe ich insgesamt 5 verschiedene, voneinander unabhängige Tests durchgeführt, um einerseits diverse Funktionen besser zu verstehen und um Fehler im Code zu eliminieren. Auf Grund dieser Tests basiert das finale Programm, mit dem der Gedächtnistrainer im aktuellen Zustand gespielt werden kann. Weitere Optimierungen, oder auch Spielfunktionen sind jederzeit noch möglich.

Softwaretest Nr.1:
Bei dem ersten Test ging es darum, sich mit der IDE, dem Arduino und der Stromversorgung zu beschäftigen. Im Detail bedeutet dies sowohl die auf dem Arduino integrierte LED, als auch eine externe LED blinken zu lassen, mit Hilfe eines Programms aus der Arduino Beispiel-Bibliothek.

Softwaretest Nr.2:[7]
Bei dem zweiten Test ging es darum, eine LED mit Hilfe eines Druck-Tasters leuchten zu lassen. Dies bedeutet in einer Schleife zu gucken ob der Taster betätigt wird und falls ja die am konfigurierten Pin angeschlossene LED mit Strom zu versorgen.

Softwaretest Nr.3:[8]
Bei dem dritten Test ging es darum, mehrere LEDs, welche in einem Array gespeichert wurden, in der immer gleichen Reihenfolge aufleuchten zu lassen. Bei diesem Test hat sich gezeigt, dass es deutlich simpler ist die Pins der Taster und LEDs in 2 Arrays zu speichern und diese im Programm aufzurufen.

Softwaretest Nr.4:[9]
Bei dem vierten Test ging es darum, von 4 LEDs eine per Zufall auszuwählen und den jeweiligen Pin in ein neues Array zu speichern. Dieser Array wurde nach jedem hinzufügen einer zufälligen LED im Serial Monitor der Arduino IDE überprüft.

Softwaretest Nr.5:
Bei dem fünften Test ging es darum, jeder LED einen Taster zuzuteilen. Es wurden mehrere Funktionen hinzugefügt und getestet, die für das finale Programm notwendig waren. Diese sind unteranderem: Reset, Startsequenz und Abbruchsequenz. Zusätzlich diente dieser Test der Sicherstellung, dass jede Spielrunde individuell betrachtet wird.

[7] GIF
[8] GIF
[9] Bild

Flussdiagramm

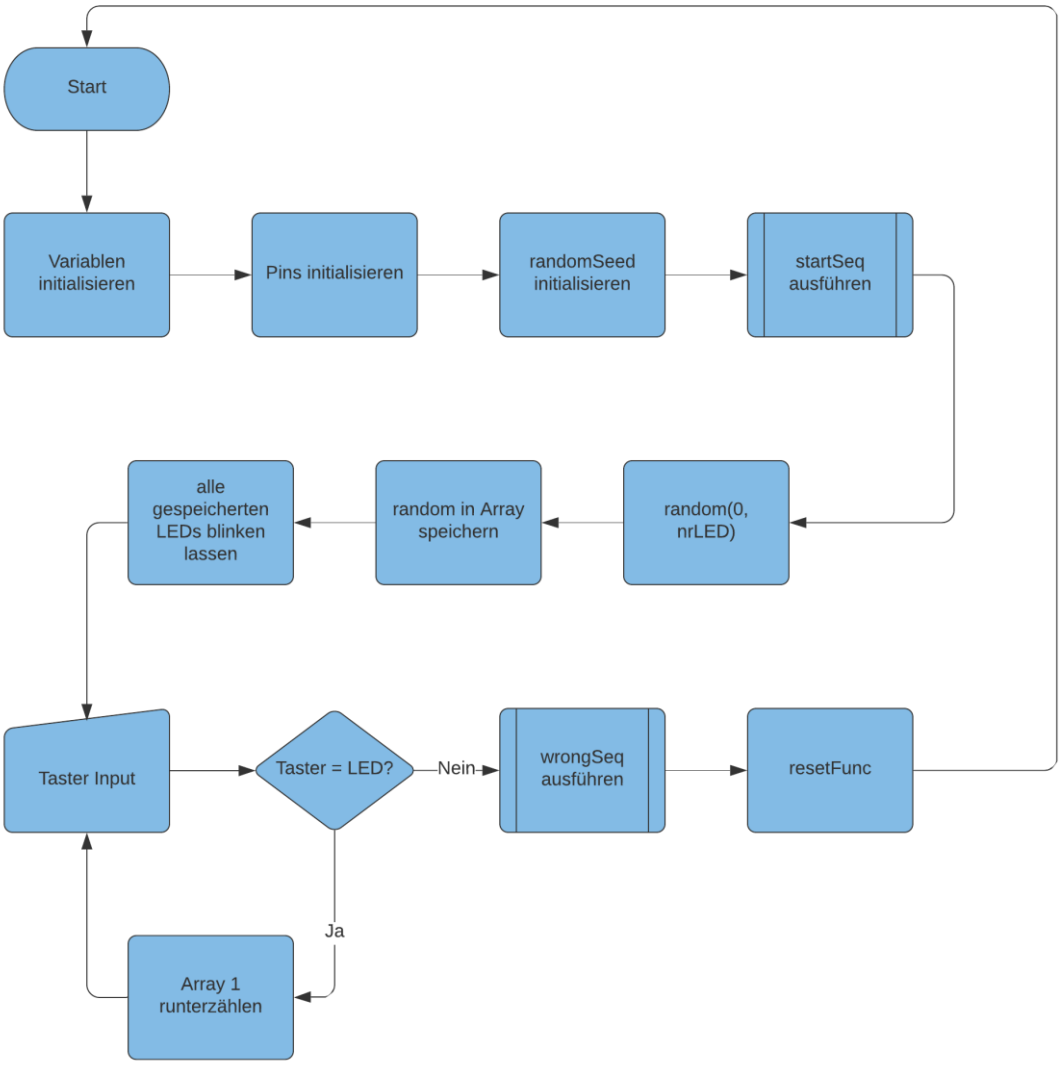


Abbildung 2: Programmablauf

Programmausschnitt

```
//Memory Game - Gedächtnistrainer

//-----------------------------------------------------------------------
//start variables
const int ledPin[] = {12, 10, 8, 6}; //LED pins
const int butPin[] = {11, 9, 7, 5}; //Button pins
const int nrLED = 4; //number of LEDs
const int nrBut = 4; //number of buttons
const int startButPin = 11; //Pin for start button
int saveNr[] = {}; //empty array to save random order of blinking LED
int count = 1; //startposition for counter //array pos 0 funktioniert
nicht?
int endCheck = false; //check value for user input check

//-----------------------------------------------------------------------
```

Vollständiger Programmcode zu finden im GitHub.

Kevin Feht

Anhang

Abbildung 4: Arduino Uno Rev. 3

Abbildung 3: LED versch. Farben

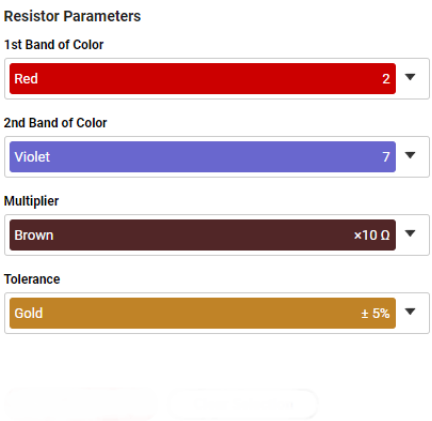


Abbildung 6: Widerstand

Abbildung 5: Push-Taster

Abbildung 7: Jumper Wires

Tabellenverzeichnis

Abbildungsverzeichnis

Quellenverzeichnis

https://create.arduino.cc/projecthub/Barqunics/the-memory-game-7a9f10 [02.11.2021, um 10:45]

https://www.instructables.com/two-ways-to-reset-arduino-in-software/ [15.11.2021, um 16:30]

https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code [19.11.2021, um 12:45]

Verwendete Programme

fritzing.0.9.3b – Verkabelung Abbildung
Microsoft Word – Dokumentation
Lucidchart – Flussdiagramm
Arduino DIE – Programmierung
Notepad++ – RTF Konvertierung