Pong

Bild???

TODO!!!

Luca Manuel Krause

Matrikelnummer: tinf104236

Fachsemester: 5

Wintersemester 2021/2022

Pinneberg, den 17.03.2022

Inhalt

[Aufgabe 2](#_Toc98670627)

[Abstrakte Erläuterung 2](#_Toc98670628)

[Umsetzung der Aufgabe 2](#_Toc98670629)

[Elemente auf dem Board 2](#_Toc98670630)

[Entwicklungsumgebung 2](#_Toc98670631)

[Installation Geany und MiniIDE 2](#_Toc98670632)

[Programmorganisationsplan 3](#_Toc98670633)

[Spielablauf / Beschreibung des POP 4](#_Toc98670634)

[Beschreibung wichtigster Dateien/Unterprogramme 4](#_Toc98670635)

[Ball.inc 4](#_Toc98670636)

[Programmtests 5](#_Toc98670637)

[Debug-Tests 5](#_Toc98670638)

[Spieltests 5](#_Toc98670639)

[Probleme 7](#_Toc98670640)

[Sichtbarkeit des Balls 7](#_Toc98670641)

[Aktualisierung Sieben-Segment-Anzeige 7](#_Toc98670642)

[Literaturverzeichnis 8](#_Toc98670643)

# Aufgabe

## Abstrakte Erläuterung

Ziel der Aufgabe ist eine Umsetzung eines komplexeren Programms auf dem trainer11-Board. Die Schwerpunkte liegen vor allem bei der klar strukturierten Programmierung und der Verständlichkeit der Dokumentation.

## Umsetzung der Aufgabe

Die Aufgabe ist realisiert in dem Programm Pong, einem 1972 von Atari veröffentlichten Videospiel, dass als weltweit erstes beliebtestes Videospiel gilt2.

## Elemente auf dem Board

Um das Spiel Pong auf dem Board zu realisieren wird das LCD-Display für das Spielfeld genutzt, die 7-Segment-Anzeige für den Punktestand und für einen Reset des Spielstandes eine Taste.

# Entwicklungsumgebung

Betriebssystem Windows 10

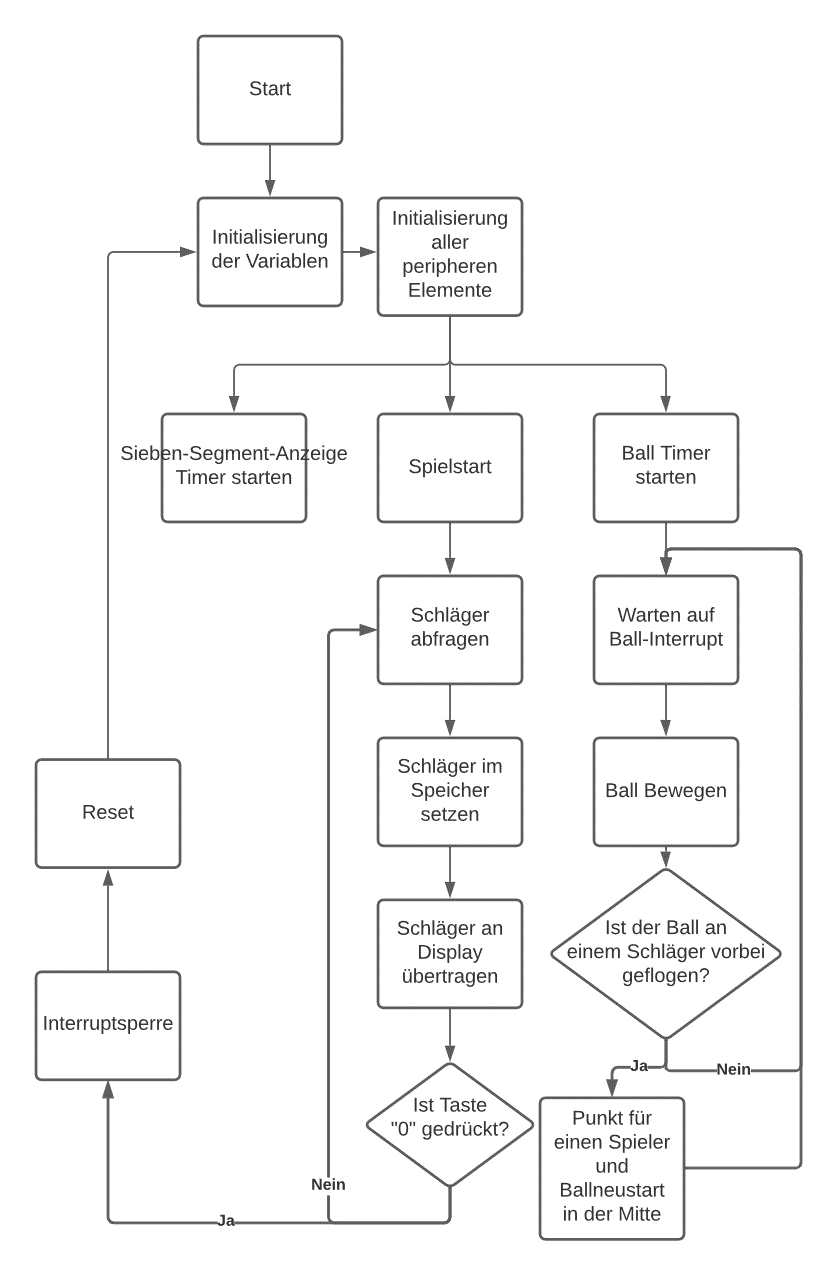
Software Geany IDE, MiniIDE

Sprache ASM11

## Installation Geany und MiniIDE

* Bei der Nutzung der Geany IDE sollte bei jedem Dokument bei >Dokument>Dateityp festlegen>Kompilersprachen>Assembler ausgewählt werden.
* Anschließend in dem Reiter >Erstellen>Kommandos links „Assembler“ und rechts „C:\miniIDE\ASM11 %f -l“ eintragen.
* Nun im gleichen Fenster unter „Befehle zum Ausführen“ bei links „Ausführen“, rechts „C:\Realterm\realterm "first" "display=1" "rows=40" "baud=9600" "sendfile=%d\%e.s19"“ eintragen.
  + Eventuell kann hier auch noch der Standardport mit “port=COMPort“ eingetragen werden.
* Die miniIDE von Mgtek sollte hierzu im Hauptverzeichnis C:/miniIDE installiert werden. Diese IDE nutzt nur den Assembler ASM11
* Realterm sollte auch im Hauptverzeichnis als C:/realterm installiert werden.[[1]](#footnote-1)

# Programmorganisationsplan



## Spielablauf / Beschreibung des POP

Nach dem Starten des Microcontrollers und Überspielen der Dateien werden die Variablen und peripheren Elemente, wie das Display, die Potentiometer und die 7-Segment-Anzeige initialisiert. Die Variablen werden vollständig mit Nullen gefüllt und im Anschluss von den eigenen Initialisierungsprogrammen, wie dem Setzen des Balles, auf spezielle Werte gesetzt.

Bei der Initialisierung werden außerdem die beiden Timer gestartet, wobei der eine Timer für die Aktualisierung des Balls und der andere für die Aktualisierung der 7-Segment-Anzeige zuständig ist. Der Timer für die Anzeige der Punkte liest nur die Punkte aus und veröffentlicht diese kontinuierlich.

Bei dem Timer des Balls wird zudem noch geprüft, ob er die Grenzen des Spielfeldes erreicht und deshalb entweder von Ihnen abprallen muss (obere und untere Grenze) oder an den Schlägern vorbeifliegen würde und deshalb einen Spieler einen Punkt erhält. In dem Moment wird dann auch die Anzahl der Punkte aktualisiert.

Nach den Initialisierungen wird die Hauptschleife des Spiels gestartet. In dieser wird immer die Position der Schläger abgefragt, diese auf einen Wertebereich transformiert und im Anschluss im Speicher abgelegt, der jedes Byte der Schläger beinhaltet. Anschließend werden die Bytes der Schläger an das Display übertragen. Als letzter Teil der Schleife wird die Taste „0“ überprüft, die eine Reset-Taste darstellt. Mit dieser Taste werden der Spielstand und das Spielfeld zurückgesetzt, wobei im Anschluss wieder mit der Initialisierung gestartet wird.

# Beschreibung wichtigster Dateien/Unterprogramme

## Ball.inc

In der Datei zum Ball werden alle Dinge zum Ball geklärt. Er wird hier initialisiert, dessen Timer gestartet und bewegt.

Bei der Bewegung des Balls gibt es verschiedene Fälle. Im einfachsten Fall wird der Ball von seiner Bewegung her einfach nur einen Punkt weitergesetzt. Fliegt der Ball also nach rechts unten, nimmt dessen x- und y-Position um jeweils +1 zu. Trifft der Ball die obere Grenze, muss er nach unten abprallen, trifft er die untere, muss er nach oben abprallen.

Pseudocode:

//Konstanten

POS\_BAT\_1 = 1; POS\_BAT\_2 = 126;

//Variablen

move\_x = 1; move\_y = 1;

x\_pos = Width/2; y\_pos = Height/2;

x\_pos += move\_x; y\_pos += move\_y;

If (y\_pos == 0) {

move\_y = 1

} else if (y\_pos == Height -1) {

move\_y = -1

}

//Befindet sich der Ball vor einem der Schläger?

If (x\_pos == POS\_BAT\_1 + 1) {

//Werte des ersten Schlägers

If (y\_pos >= SchlägerPosStart && y\_pos < SchlägerPosEnd) {

//Schläger getroffen

move\_x = 1

} else {

//Punkte verteilen

}

} else if (x\_pos == POS\_BAT\_2 - 1) {

//Werte des zweiten Schlägers

If (y\_pos >= SchlägerPosStart && y\_pos < SchlägerPosEnd) {

//Schläger getroffen

move\_x = -1

} else {

//Punkte verteilen

}

}

Für das Treffen des Schlägers wurde später noch eine komplexere Art der Berechnung eingeführt, um das Abprallen vom Schläger variabel zu gestalten. Hierbei kann der Ball im oberen Drittel des Schlägers nach oben abprallen, in der Mitte nach vorne und im unteren Drittel nach unten. Diese Ausgangswinkel sind dann unabhängig davon, aus welcher Richtung der Ball auf den Schläger getroffen ist.

# Programmtests

Das Programm wurde auf zwei verschiedene Weisen getestet. Zum Einen wurden Debug-Ausgaben auf der Textausgabe benutzt. Zum Anderen wurde das Spiel Pong gespielt und nach speziellen Fehlern gesucht.

TODO: vielleicht noch Bilder von einzelnen Testfällen

## Debug-Tests

|  |  |  |
| --- | --- | --- |
| Testfall | Erwartetes Ergebnis | Ergebnis |
| Vollständiges Drehen beider Potentiometer | Die Werte 0 – 55 werden nach der Wertetransformation ausgegeben | Nur die Werte 0 – 55 sind in der Ausgabe zu sehen |

## Spieltests

|  |  |  |
| --- | --- | --- |
| Testfall | Erwartetes Ergebnis | Ergebnis |
| Ball auf dem oberen Rand des Schlägers aufkommen lassen. | Der Ball prallt ab. | Der Ball prallt ab. |
| Ball auf dem unteren Rand des Schlägers aufkommen lassen. | Der Ball prallt ab. | Der Ball prallt ab. |
| Der Ball fliegt oben am Schläger vorbei. | Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt. | Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt. |
| Der Ball fliegt unten am Schläger vorbei. | Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt. | Der Gegner bekommt einen Punkt und der Ball wird zurückgesetzt. |
| Der Ball kann sich überall auf dem Spielfeld bewegen | Der Ball bewegt sich überall. | Der Ball bewegt sich überall. |
| Den Schläger mit dem Ball am oberen Ende treffen | Der Ball prallt am oberen Ende des Schlägers nach oben ab. | TODO |
| Den Schläger mit dem Ball in der Mitte treffen | Der Ball prallt in der Mitte des Schlägers nach vorne ab. | TODO |
| Den Schläger mit dem Ball am unteren Ende treffen | Der Ball prallt am unteren Ende des Schlägers nach unten ab. | TODO |

# Probleme

## Sichtbarkeit des Balls

Der Ball ist durch eine zu hohe Schnelligkeit des Balls leider nur stark gedimmt zu erkennen. Dazu führte erstmal eine Begrenzung des Hardware-Interrupt-Timers, der um maximal den Wert 216 – 1­ = 65.535 geändert werden kann. Um dieses Problem zu umgehen, gibt es nun eine Konstante, mit Hilfe derer nun dieser maximale Wert mit einer ganzzahligen Nummer multipliziert werden kann. Der Timer für die Aktualisierung des Balls kann jetzt zum Beispiel bei jedem zweiten Timer-Interrupt (Konstante = 2) den Ball auf dem Display aktualisieren. Sollte man eine Geschwindigkeit zwischen den Ganzzahligen Werten anstreben, kann man außerdem die Konstante für den Interrupt Timer ändern, bei der der Interrupt-Timer aufgerufen wird. Als Beispiel könnte man den Wert von 65.000 auf 40.000 ändern. Der Ball ist somit fast 40 % schneller.

## Aktualisierung Sieben-Segment-Anzeige

Leider war die Aktualisierung der Sieben-Segment-Anzeige nicht konstant, beziehungsweise konnte nicht so schnell durchgeführt werden, damit das menschliche Auge kein Flackern wahrnimmt. Das liegt an der Aktualisierung des Balls und des Schlägers auf dem Display, dessen Kommunikation viel langsamer als der Prozessor sind.

TODO:

Als Lösungsweg dafür wurde nun die Anzahl der Spaltenübertragungen an das Display reduziert, um die die Zeit zu verringern, an der kein Interrupt stattfinden kann. Dies wurde durch ein Kommando erreicht, bei dem die nächste angesprochene Spalte nicht automatisch weitergesetzt wird, wenn einmal Daten an das LCD-Display übertragen werden.

# Literaturverzeichnis

1. <https://stud.fh-wedel.de/handout/Voelker_Joerg/>:

wesentliche Grundbausteine zum Erstellen des Codes und für das Verständnis der Kommunikation zwischen den peripheren Einheiten

a. https://stud.fh-wedel.de/handout/Voelker\_Joerg/Folien/T2ErsteBefehle.pdf

1. <https://de.wikipedia.org/wiki/Pong>:

Grundinformationen zum Spiel Pong

1. <https://www.lucidchart.com/pages/>:

Zum Erstellen des Programmorganisationsplans

1. a: Diese Beschreibung entspricht im Wesentlichen der Beschreibung von Herrn Völkers Folien [↑](#footnote-ref-1)