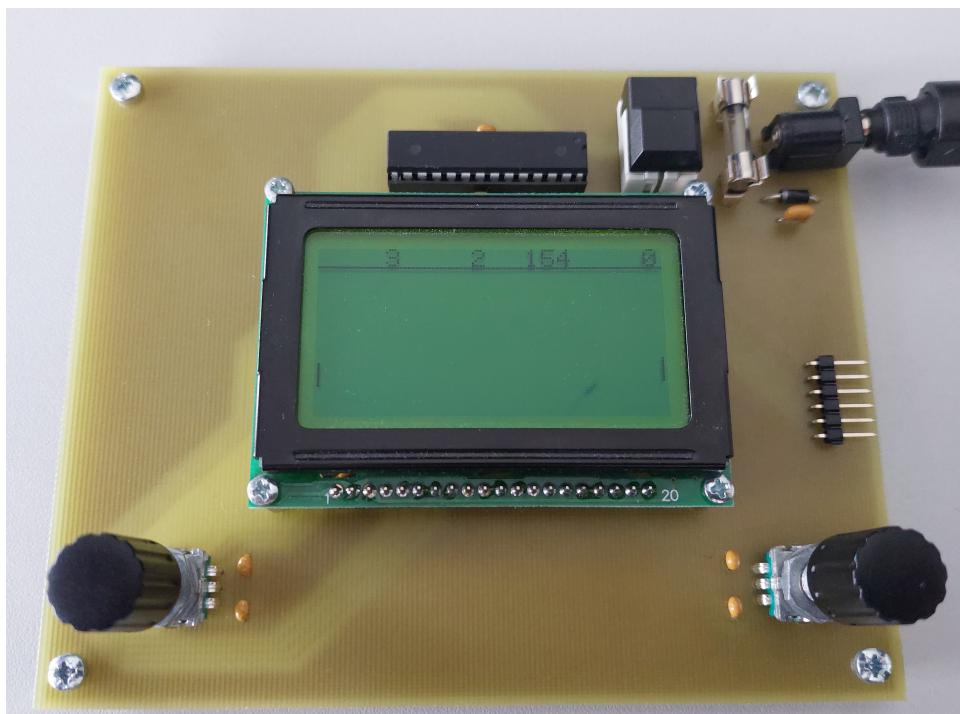


Projekt Mikrokontroller - „Pong“

DOKUMENTATION



FACHHOCHSCHULE WEDEL

*Luca Krause, tinf104236
Fachrichtung: Technische Infomatik
Fach-/Verwaltungssemester: 6*

31. Juli 2022

Inhalt

1 Einleitung	2
1.1 Aufgabe	2
2 Benutzerhandbuch	2
2.1 Einschalten des Gerätes	2
2.2 Steuerung	2
2.3 Modi	3
2.4 Zurücksetzen des Spielstandes	4
3 Programmierhandbuch	4
3.1 Entwicklungsumgebung	4
3.2 Problemanalyse	4
3.2.1 Modularisierung	4
3.3 Implementierung	5
3.3.1 Dimensionierung Display	5
3.3.2 Steuerung der Schläger	5
3.3.3 Spielablauf - Punkte	6
3.3.4 Punktanzeige	6
3.3.5 Speicherung des ewigen Highscores	8
3.3.6 Spielfeldbegrenzung	9
3.3.7 Kollisionen des Balls und der Schläger	10
3.3.8 Bewegungsrichtungen des Balls	10
3.3.9 Invertierung der Schläger	11
3.3.10 Schaltplan	11
3.4 Komponenten	14
3.4.1 Zusatzfunktionen	14
3.5 Test	17
3.6 Fazit und Ausblick	18
4 Literaturverzeichnis	18

1 Einleitung

1.1 Aufgabe

Im Rahmen des Modules „Projekt Mikrocontroller“ soll das Spiel Pong auf einem Mikrocontroller entwickelt werden. Dabei werden der Ball und die Schläger auf einem Grafikdisplay dargestellt und die Steuerung der Schläger soll mittels zweier Inkrementaldrehgeber oder Drehencoder erfolgen. Die genaue Umsetzung der Regeln, des Spielfeldes und der Anzeige des Punktestandes des Spiels Pong ist dem Studenten überlassen und eine möglichst individuelle Ausgestaltung des Projektes soll erreicht werden.

Die Bauteile sind selber zu wählen, in welcher Größe, oder überhaupt, welche Bauteile man verwenden möchte.¹

Das Spiel Pong wurde 1972 von Atari veröffentlicht und wurde damit zum ersten weltweit bekannten Videospiel.²

Die Regeln dieser Implementation ähneln der Implementierung des Originalspiels stark.

2 Benutzerhandbuch

2.1 Einschalten des Gerätes

Stecken Sie das 5V-DC Kabel in den Anschluss und seien sie sich dabei sicher, dass an der Innenseite des Pins 5V und an der Außenseite GND/0V anliegen. Ist Strom am Kabel vorhanden, startet das Spiel sofort, was bedeutet, dass sich der Ball direkt anfängt, sich zu bewegen. Entstehen hierdurch schon Punkte, kann man den Taster drücken, um den Spielstand zurückzusetzen (siehe Abschnitt 2.4). Der Anschluss ist oben rechts in rot auf der Abbildung 1 zu sehen.

2.2 Steuerung

Zur Steuerung bewegt der linke Spieler den linken Drehencoder und der rechte Spieler den rechten Drehencoder. Diese können unendlich weit nach links und rechts bewegt werden. Dabei ist aber die Bewegung der Schläger auf dem Display begrenzt. Durch die Drehung bewegen sich die beiden Schläger

¹Die Aufgabenbeschreibung ist an die Aufgabenstellung angelehnt.

²Quelle: <https://de.wikipedia.org/wiki/Pong>

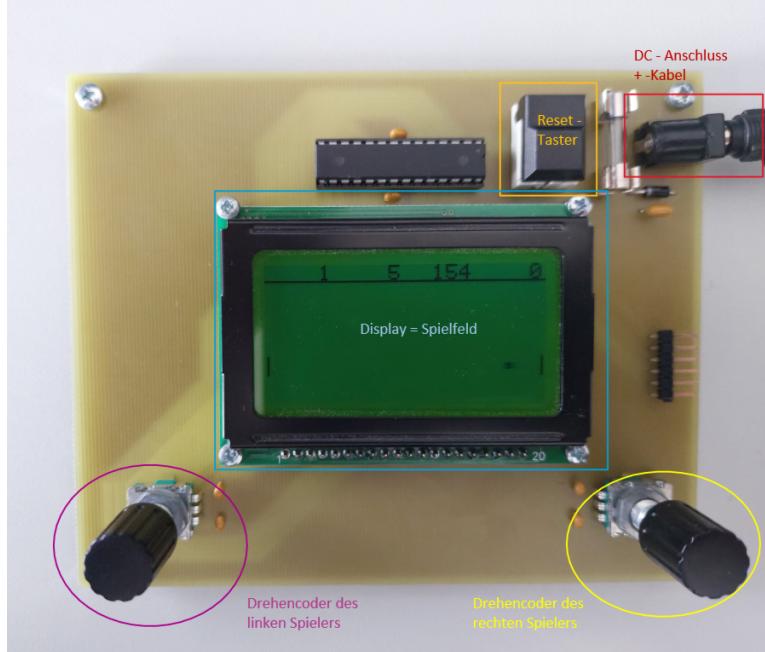


Abbildung 1: Alle Bauteile

auf dem Spielfeld nach oben und unten. Wird der Drehencoder im Uhrzeigersinn gedreht, bewegt sich der Schläger im normalen Modus nach unten. Wird eine Drehung gegen den Uhrzeigersinn durchgeführt, bewegt sich der Schläger nach oben.

Die Drehrichtungen des Schlägers des Gegners können durch ein Drücken auf den Drehencoder geändert werden. Dabei ist die Steuerung für eine konstante Zeit von ca. 5 Sekunden invertiert und nach der Invertierung kann der Schläger auch nicht wieder direkt invertiert werden, sondern erst wieder nach ca. 5 Sekunden.

Die beiden Drehencoder sind unten links und unten rechts auf der Abbildung 1 zu sehen.

2.3 Modi

Im Spiel gibt es zwei Modi, die beide durchgehend aktiv sind. Zum Einen kann man mit den Punkten gegeneinander spielen, also dass beide Spieler so viele Punkte erzielen, wie sie schaffen. Zum Anderen können beide Spieler miteinander spielen und versuchen, so viele Schlägerberührungen hintereinander durchzuführen, ohne dass der Ball am Schläger vorbeifliegt, wodurch

der ewige Highscore übertroffen werden kann, der anschließend gespeichert wird und dabei den alten Highscore überschreibt. Dabei wird immer nur der höchste Highscore gespeichert. Der maximale Highscore beträgt 65535 und kann nicht übertroffen werden.

Die verschiedenen Punkte sind auf dem oberen Teil des blau markierten Displays auf der Abbildung 1 zu sehen.

2.4 Zurücksetzen des Spielstandes

Möchte man den Spielstand zurücksetzen, drückt man den einzigen vorhandenen Knopf. Dabei werden das Spielfeld, die Punktzahlen und der zuletzt aktuelle (nicht ewiger) Highscore zurückgesetzt. Der ewige Highscore erhält intern erst ein Update, wenn ein Spieler einen Punkt erzielt hat. Also sollte der Knopf immer erst gedrückt werden, wenn ein Spieler einen Punkt erzielt hat und nicht, während versucht wird, den Highscore zu überbieten.

Der Reset kann über den orange markierten Taster der Abbildung 1, ausgelöst werden.

3 Programmierhandbuch

3.1 Entwicklungsumgebung

Betriebssystem:	Windows 10
Software:	MPLAB® X IDE
Programmiersprache:	C/C++

Betriebssystem:	Windows 10
Software:	Microsoft Visual Studio Code

3.2 Problemanalyse

3.2.1 Modularisierung

Das Projekt wird ein main-File besitzen, in dem der Hauptablauf des Spiels und die Aktualisierungen ablaufen. Zusätzlich dazu gibt es für fast jede Komponente eine Schnittstelle und ein cpp-File, welches diese Schnittstelle implementiert. Zu diesen Komponenten zählen Dinge, die etwas komplexer sind. Hierzu zählen das Display und die Drehencoder. Außerdem bekommt

der Ball eine eigene Schnittstelle, da er ebenfalls etwas komplexer ist und einen Timer besitzt, damit der Ball sich nach vorne bewegen kann.

3.3 Implementierung

3.3.1 Dimensionierung Display

Die LED - Hintergrundbeleuchtung für das Display benötigt einen Vorwiderstand, um den maximalen Strom zu begrenzen, der durch die LED's fließt. Dabei ist dem Datenblatt eine Spannung $U_{led} = 3 \text{ V}$ und ein Strom $I_v = 150 \text{ mA}$ zu entnehmen.

$$R_v = \frac{U_b - U_{led}}{I_v} \quad (1)$$

$$R_v = \frac{5 \text{ V} - 3 \text{ V}}{150 \text{ mA}} \quad (2)$$

$$R_v = 13.3 \text{ Ohm} \rightarrow 15 \text{ Ohm} \quad (3)$$

3.3.2 Steuerung der Schläger

Für die Steuerung der Schläger werden zwei Drehencoder genutzt. Die Wahl fiel auf die Drehencoder, weil ich vorher schon mit Potentiometern als Schläger gearbeitet habe und etwas Neues ausprobieren wollte. Da es, wie auch später in den Zusatzfunktionen beschrieben (siehe 3.4.1), eine Invertierung der gegnerischen Schläger geben soll, wird noch ein Taster benötigt. Dieser ergibt als Bauteil Sinn, da es nur die Zustände „Invertieren“ und „nicht Invertieren“ gibt. Dabei ist die Entscheidung nicht auf normale Taster gefallen, sondern auf Drehencoder mit eingebautem Taster. Durch die Verbindung dieser beiden Bauteile kommt es für den Benutzer zu einer Assozierung zwischen dem Taster und dem Invertieren des gegnerischen Schlägers.

Da Schalter, beziehungsweise Taster, immer ein Prellen aufweisen, muss hier entprellt werden. Getan wird dies in diesem Fall über ein Hardware-Entprellen, was durch Kondensatoren erreicht wird. Für diesen speziellen Fall wird hier für jedes Phasensignal ein 10 nF Kondensator. Da wir zwei Drehencoder mit jeweils zwei Phasensignalen haben, benötigen wir insgesamt vier 10 nF Kondensatoren. Die Auswahl des Kondensators basiert auf der Nutzung von 10 nF Kondensatoren in diesem Video: <https://www.youtube.com/watch?v=bBhbynj6NYM&t=495s>. Zusätzlich dazu hätte man hier die 10 nF noch über eine $t = 2 \text{ ms}$ Zeitkonstante aus dem Datenblatt auf

eine Kondensatorgröße von 2 uF gekommen, was allerdings einem Faktor 200 entspricht:

$$Iv = \frac{Uv}{Rv} \quad (4)$$

$$Iv = \frac{5V}{10k\Omega} \quad (5)$$

$$Iv = 0.5mA \quad (6)$$

$$c = \frac{I * t}{\Delta U} \quad (7)$$

$$c = \frac{0.5mA * 2ms}{5V} \quad (8)$$

$$c = 2\mu F \quad (9)$$

(10)

3.3.3 Spielablauf - Punkte

Entschieden wurde sich für einen endlosen Spielmodus, bei dem beide Spieler so lange gegeneinander spielen können, wie sie möchten und am Ende kann dann geschaut werden, wer gewonnen hat. Gleiches gilt für den Highscore. Dieser ist maximal zwei Byte breit, kann also maximal den Wert 65535 erreichen. Also kann der Highscore so weit erweitert werden, bis 65535 erreicht wurde.

3.3.4 Punktanzeige

Es gibt vier verschiedene Punkte. Dazu zählen die Punkte beider Spieler, der jetzige und der ewige Highscore.

Die Punkte befinden sich auf dem Display in den oberen 7 Bit der ersten Page und grenzen hier das Spielfeld durch eine 1 Bit breite Linie ab (siehe Abbildung 2), sodass die gesamte obere Page des Displays nicht für das Spielfeld genutzt wird. Die Punkte der beiden Spieler sind die beiden äußersten Punkte. Der jetzige Highscore ist links in der Mitte und der ewige Highscore ist rechts in der Mitte. Diese Anordnung ist so gewählt, da die Punkte jeweils dem linken oder rechten Spieler zugeordnet werden und der Highscore eine Errungenschaft beider Spieler ist, weshalb der Highscore in der Mitte angezeigt wird. Die Erklärung ist grafisch in Abbildung 3 zu sehen.

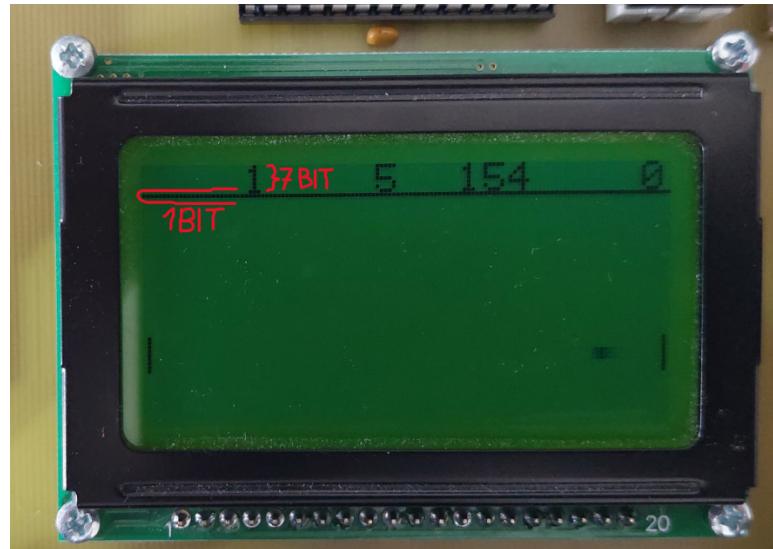


Abbildung 2: Punkteanzeige Höhe

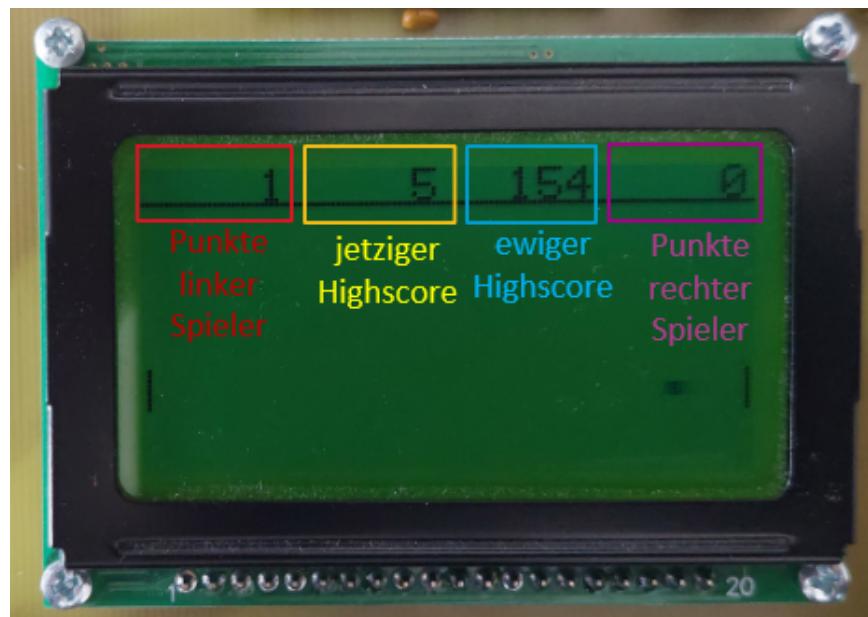


Abbildung 3: Punkte Erklärung

3.3.5 Speicherung des ewigen Highscores

Es gibt, wie auch später im Abschnitt 3.4.1 beschrieben, einen ewigen Highscore. Dieser ewige Highscore wird im EEPROM gespeichert, der schon im Microcontroller vorhanden ist. Allerdings hat EEPROM ein Problem. Die Haltbarkeit der EEPROM-Zellen ist auf eine bestimmte Anzahl an Schreib- und Lesezyklen begrenzt. Im Standardfall garantieren Hersteller rund 10.000 - 1.000.000 Schreibzyklen³. Die Lese-Operationen können wesentlich häufiger durchgeführt werden.

Um die Anzahl an Schreibzyklen nicht zu schnell zu erreichen, musste eine Art der Speichernutzung her, die wenig Schreib-Operationen auf dem Speicher durchführt.

Die einzige Information, die auf den EEPROM geschrieben wird, ist der ewige Highscore. Um diesen Wert zu speichern, gibt es mehrere Möglichkeiten. Alle gleich aufgezählten Möglichkeiten beziehen sich darauf, einen neuen Highscore zu speichern (alter Highscore wurde überboten).

Dazu zählt die schreibintensive Art zu aktualisieren, immer sofort, wenn einer neuer Höchststand erreicht ist. Dies hat den Vorteil, dass immer, kurz bevor der Strom wegfällt (zum Beispiel durchs Kabel ziehen oder durch das Drücken des Reset-Knopfes), der aktuellste Highscore gespeichert wurde.

Eine andere Art wäre das Speichern, wenn der Highscore-Versuch beendet wurde. Ein Highscore-Versuch ist beendet, sobald der Ball an einem Schläger vorbeifliegt. Dies hat den Vorteil, dass der Speicher nur beschrieben wird, wenn der Highscore in den nächsten Sekunden nicht direkt wieder hochzählt (Ein Highscore- Versuch wurde beendet). Der Nachteil hierbei ist allerdings, dass der letzte, noch nicht beendete Highscore-Versuch nicht gezählt wird. Hat man nun also einen sehr hohen Highscore erreicht, berührt aber den Reset-Knopf, wird der Wert nicht gespeichert.

Die dritte Alternative wäre ebenfalls das Speichern, immer wenn der Zug beendet wurde, aber mit einer Schutzschaltung, dass wenn der Strom wegfällt (eben auch bei einem Reset), alle wichtigen Werte gespeichert werden können. Diese Alternative verbindet die Vorteile beider vorrangeprogrammierten Varianten, ist allerdings schwieriger zu implementieren, da der Strom bei allen anderen Hardware-Bauteilen direkt abgeschaltet werden muss, sobald die Spannung unter einen bestimmten Wert fällt, damit der Mikrocontroller noch genug Strom erhält, um einen Wert im EEPROM speichern zu können.

Letzendlich habe ich mich für die zweite Art entschieden, da der Aspekt mit den Schreib-Operationen wichtiger erschien, als die Aktualität des ewi-

³https://de.wikipedia.org/wiki/Electrically_Erasable_Programmable_Read-Only_Memory

gen Highscores. Die dritte Alternative ist aufgrund des erhöhten Aufwandes nicht durchgeführt worden.

```
1 if (playerReceivedPoint()) {  
2     eeprom.writeWord(highscore, addr);  
3 }
```

3.3.6 Spielfeldbegrenzung

Wie auch schon im Kapitel zur Punktanzeige (3.3.4) beschrieben, benötigt die Punktanzeige sieben Bit zur Darstellung. Zusätzlich dazu gibt es noch eine ein Bit breite Abgrenzung zum Spielfeld, damit die Spieler vernünftig herleiten können, wie der Ball von der Bande abprallen wird. An den anderen Kanten des Spielfeldes wird diese Begrenzung nicht benötigt, da es hier eine natürliche Begrenzung durch die Displaygrenze gibt.

Durch die Punkteanzeige plus der manuellen Spielfeldbegrenzung bleiben dem Spielfeld also $8 - 1 = 7$ Byte in der Höhe für die vertikale Darstellung des Spielfeldes.

Das Spielfeld wird nach links und rechts hin von den Schlägern begrenzt. Die Schläger sind einen Pixel vom Displayrand entfernt und einen Pixel breit. Dadurch resultiert aus dem 128x64-Bit Display ein $(128-2*2) \times (64-8) = 124 \times 56$ -Bit großes Spielfeld, auf dem sich der Ball bewegen kann.

Der Aufbau des Spielfeldes ist in Abbildung 4 zu sehen.

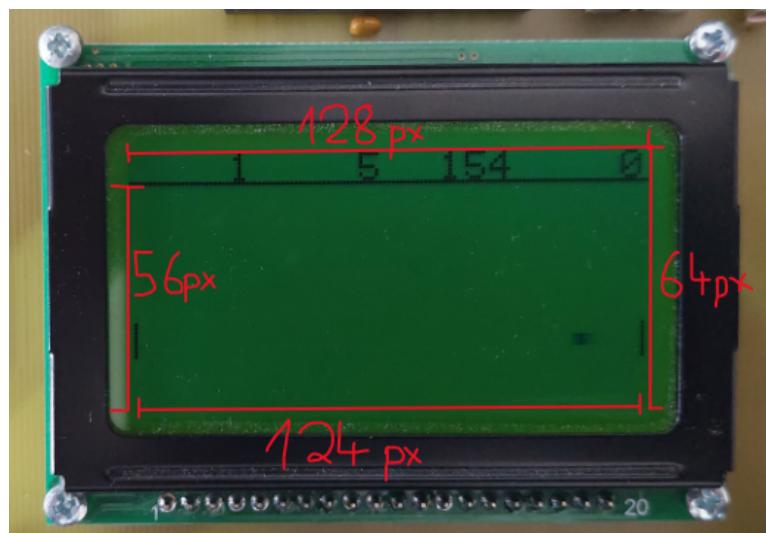


Abbildung 4: Spielfeldbreite

3.3.7 Kollisionen des Balls und der Schläger

Die Kollision des Balls mit dem Schläger findet nicht wie normal so statt, dass sich ein Pixel des Balls über einem Pixel des Schlägers befindet, sondern so, dass die Kollision errechnet wird, wenn der Ball einen Pixel vor dem Schläger ist (liegen, wie in Abbildung 5 zu sehen, direkt aneinander).

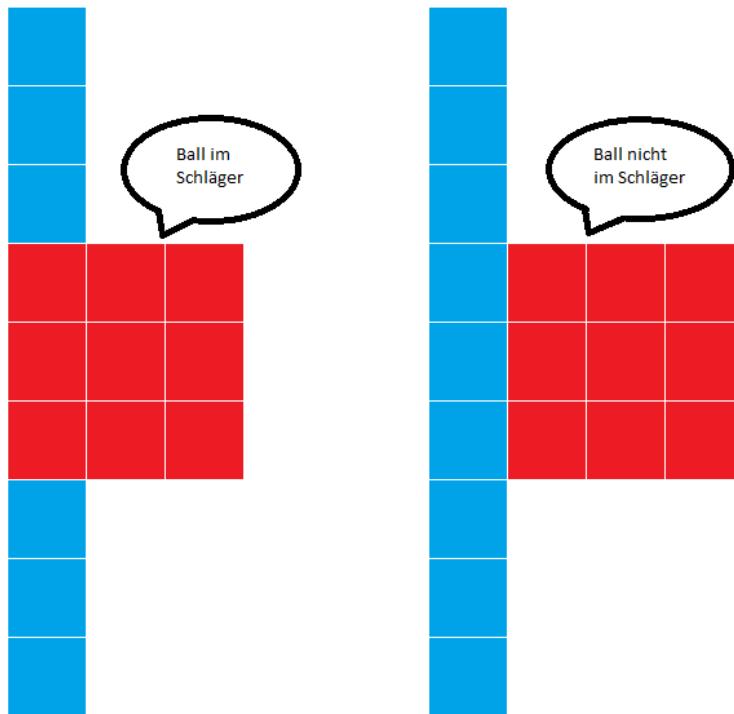


Abbildung 5: links: Ball kollidiert im Schläger, rechts: Ball kollidiert am Schläger

Da der Ball über einen Timer aktualisiert wird, lohnt es sich, die Kollision direkt im Timer zu errechnen, da sich so der Ball nicht durch unglückliche Verteilung der Rechenzeit zwischen Interrupts und Hauptablauf des Spiels an den Schlägern vorbeischleichen kann. Dabei ist auch wichtig, dass der Ball pro Timer-Interrupt immer nur einen x-Wert weitergehen kann, damit auch hier der Ball nicht am Schläger vorbeihuschen kann.

3.3.8 Bewegungsrichtungen des Balls

Die Bewegungsrichtungen des Balls an den Schlägern ist dadurch entstanden, dass sich der Ball idealerweise pro Timer-Interrupt einen Pixel weiter

bewegen kann, um den Problemfall zu umgehen, dass der Ball in manchen Interrupts zwei oder gar keinen x-Wert weitergeht. Dies vereinfacht die Berechnung enorm.

```

1 //Beispielcode linker Schläger
2 hitType hitRes = UNDEFINED_HIT;
3 //drei Abprallwinkel
4 int batIncrease = BAT_SIZE / 3;
5 if (ball.left == leftBat.x + 1) {
6     if (ball.y < leftBat.y || ball.y >= leftBat.y) {
7         //am Schläger vorbei
8         hitRes = NO_HIT;
9     } else if (ball.y < leftBat.y + batIncrease) {
10        hitRes = HIT_UPPER;
11    } else if (ball.y < leftBat.y + 2*batIncrease) {
12        hitRes = HIT_MID;
13    } else if (ball.y < leftBat.y + 3*batIncrease) {
14        hitRes = HIT_LOWER;
15    }
16 }
```

3.3.9 Invertierung der Schläger

Die Invertierung des Schlägers, wie auch hier: 3.4.1 noch beschreiben, wird über die Taster der gegnerischen Drehencoder gesteuert und werden über Pin Change Interrupts ausgelesen. Jede Invertierung wird etwa 5 Sekunden lang ausgeführt und anschließend etwa 5 Sekunden lang blockiert. Die 5 Sekunden beruhen auf dem Takt von 1 MHz einem Taktteiler von 1024 und einem Vergleichswert von 255 beim Timer, sowie einem internen Zähler, der bis 20 zählt.

$$5.224s = \frac{1}{1000000}s \cdot 1024 \cdot 255 \cdot 20 \quad (11)$$

3.3.10 Schaltplan

Für die Spannungsversorgung erhält der Mikrocontroller an den Pins VCC und AVCC 5 V. An diesen Ports gibt es jeweils, noch gegen Masse angegeschlossen, einen Kondensator zur Spannungsstabilisierung.

Der Port C und D werden beide zur Kommunikation mit dem Display verwendet. Dabei werden die Kommandos, wie auf dem Schaltplan (6) zu

sehen, über den Port C verschickt, da dieser 7 Bit breit ist und das Display 7 Kommando-Bits besitzt. Die Datenbits des Displays werden über den Port D des Mikrocontrollers verschickt, da hier erstens 8 Bit verfügbar waren und zweitens, dieser Port nicht für die Programmierung des Microcontrollers genutzt wird, was zu Problemen führen kann. Dadurch, dass hier die Anschlüsse nicht auf verschiedene Ports aufgeteilt wurden, hat man den programmativen Vorteil, dass die Übertragung zum Display nicht unnötig komplex gestaltet wird.

Das Display besitzt außerdem noch eine Hintergrundbeleuchtung, die mit einem Vorwiderstand von 15 Ohm an 5 V angeschlossen wird.

Um den Kontrast des Displays anpassen zu können, wird ein Potentiometer an den VEE-Pin des Displays angeschlossen, und mit dem Abnehmenden Ausgang davon an V0 des Displays verbunden. Hier wird eine höhere Spannung als 5 V erzeugt.

Das Display erhält auch noch einen Kondensator zur Spannungsstabilisierung an VCC.

Der Reset-Pin des Microcontrollers wird zusätzlich zu der Verbindung mit dem Programmer ebenfalls noch mit dem Eingabetaster (siehe Zusatzfunktion Spiel-Reset: 3.4.1) verbunden, um auch einen Reset manuell durchführen zu können. Der Eingabetaster hat noch einen 10 kOhm PullUp-Widerstand.

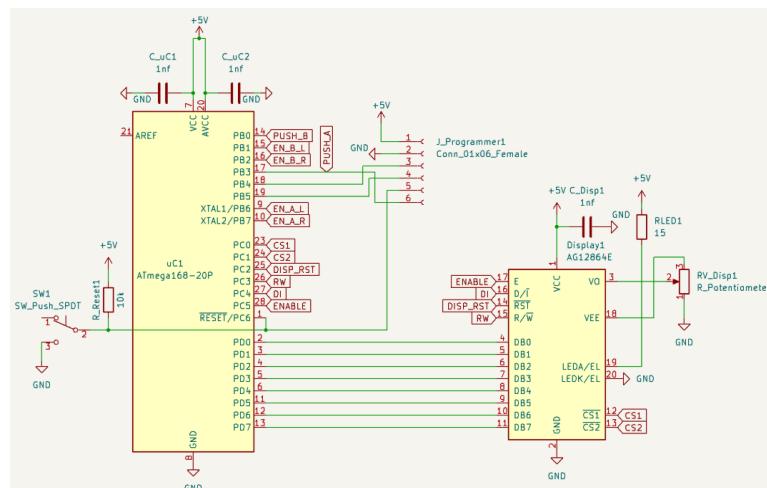


Abbildung 6: Schaltplan Mikrocontroller und Display

Die beiden Drehencoder erhalten jeweils an Phase A, B und an den Switch einen PullUp-Widerstand von 10 kOhm, der an 5V angeschlossen ist.

Die Signale für den Mikrocontroller werden jeweils zwischen dem PullUp-Widerstand und dem Anschluss des Drehencoders abgenommen.

Die Kondensatoren sind, wie schon im Abschnitt 3.3.2 beschrieben, zum Entprellen zwischen Phase A und B und Masse angeschlossen.

Der Drehencoder des linken Spielers ist an die Pins PB3, PB6 und PB7 angeschlossen und der Drehencoder des rechten Spielers ist an die Pins PB0, PB1 und PB2 angeschlossen.

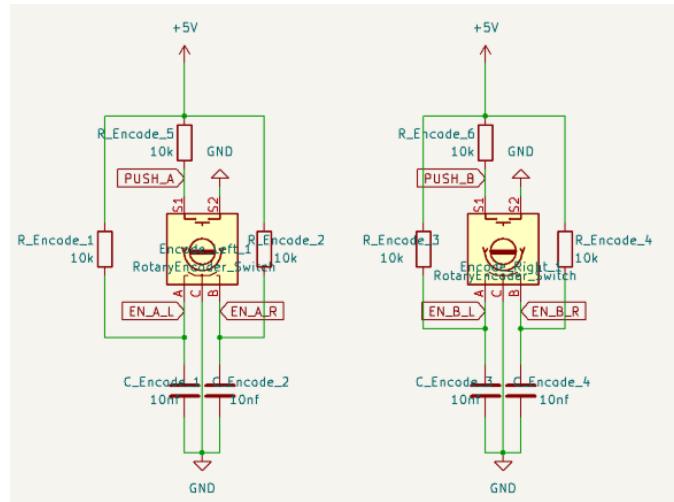


Abbildung 7: Schaltplan Drehencoder

Die Sicherungsschaltung besteht aus einem Barrel Jack-DC-Anschluss, einer Sicherung, einer Diode und einem Kondensator. Diese sind so geschaltet, dass im Fall eines Anschließens in die falsche Richtung die Diode viel Strom durchlässt und die Sicherung durchbrennt. Dadurch trennt sich der Stromkreis und die Schaltung dahinter ist geschützt.

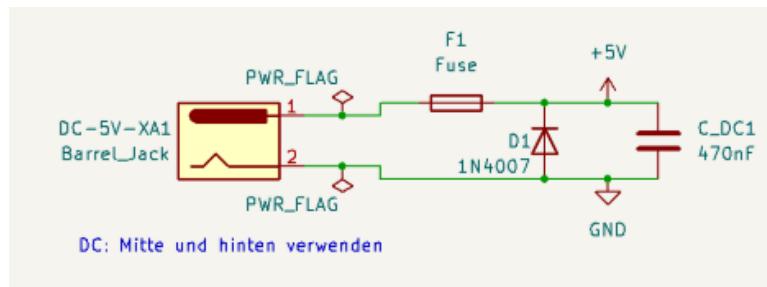


Abbildung 8: Schaltplan Sicherungssystemsteuerung

3.4 Komponenten

Die Komponenten sind speziell für den Einsatz auf dem gefertigten PCB (siehe Deckblatt) abgestimmt.

- ATMega168PA
- Sockel für ATMega168PA
- Drehencoder 2 * EC11E mit Switch
- Taster (hier Eingabetaster)
- Display DEM128064...
- 20er Pinleiste für das Display
- 6er Pinleiste für die Programmer-Schnittstelle
- Sicherung
- 2 Sicherungshalter
- Diode IN4007
- Potentiometer
- Kondensator 1*470 nF, 3*1 nF, 4*10 nF
- Widerstände 15 Ohm, 7*10 kOhm
- 5V-DC-Anschluss
- 5V-DC-Kabel mit 5V am Pin und GND außen

3.4.1 Zusatzfunktionen

3.4.1.1 Schneller werdender Ball

Der Ball wird mit der Zeit schneller, wobei hier nicht die Anzahl der Schläge zählt, sondern die Zeit, die der Ball bewegt wird, ohne dass er an einem Schläger vorbeifliegt. Die Geschwindigkeit des Balls wird zurückgesetzt, wenn der Ball an einem Schläger vorbeifliegt. Dabei wird der Ball nicht sonderlich schnell, um das Spiel nicht zu schwierig zu gestalten oder die Hardware zu überlasten (die Interrupts wären sonst zu oft am Rechnen). Der Ball wird schneller, indem der Vergleichswert des Ball-Timers kontinuierlich runtergesetzt wird. Dieser besitzt einen Wert zwischen 5000 und 2000.

3.4.1.2 Invertierte Schläger

Ein Spieler kann den Schläger des Gegners invertieren, indem er auf den Drehencoder drückt. Dadurch invertiert sich die Bewegungsrichtung des gegnerischen Schlägers für ca. 5 Sekunden und kann für diese Länge und ca. 5 Sekunden danach nicht wieder invertiert werden, damit der Knopf nicht durchgehend gedrückt wird.

3.4.1.3 Jetziger Highscore

Als zweite Zahl auf dem Display (von links) wird angezeigt, wie oft der Ball hintereinander einen Schläger getroffen hat, ohne an einem Schläger vorbeizufliegen. Dieser Wert wird zurückgesetzt, wenn der Ball an einem Schläger vorbeifliegt.

3.4.1.4 Ewiger Highscore

Es gibt einen ewigen Highscore, der auf dem EEPROM gespeichert ist und als dritte Zahl (von links) auf dem Display angezeigt wird. Dieser ewige Highscore kann durch den jetzigen Highscore aktualisiert werden, wenn dieser den ewigen Highscore übertrifft.

3.4.1.5 Spiel-Reset

Das gesamte Spiel kann durch das Drücken des einzigen vorhandenen Knopfes zurückgesetzt werden. Dazu zählen das Spielfeld und die Punktzahlen mit dem jetzigen Highscore. Der ewige Highscore ist davon nicht betroffen.

3.4.1.6 Abprallen in drei Richtungen

Der Ball kann in drei Richtungen (-45° , 0° , 45°) vom Schläger abprallen. Dies ist in der Abbildung 9 mit den lila Pfeilen zu sehen. Ausschlaggebend dafür ist der Trefferpunkt auf dem Schläger. Die verschiedenen Trefferflächen sind in der Abbildung mit einem orangefarbenen Strich getrennt. Wird der Schläger im oberen Drittel getroffen, prallt der Ball nach oben ab (-45°). Ist der Treffer im mittleren Drittel prallt der Ball nach vorne ab (0°). Die dritte Möglichkeit ist das Treffen im unteren Drittel, wobei hier der Ball nach unten abprallt (45°). Wie die grünen Pfeile zeigen, ist es egal, in welchem Winkel der Ball auf den Schläger prallt. Der Abprallwinkel hängt, wie gesagt, nur vom Trefferpunkt auf dem Schläger ab.

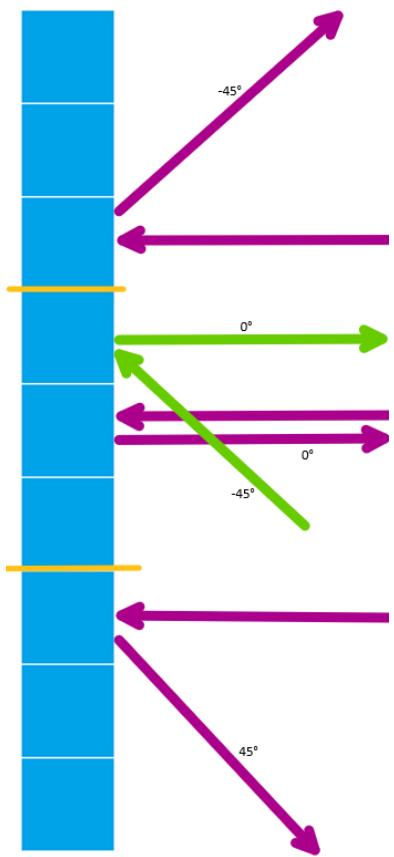


Abbildung 9: verschiedene Arten des Abprallens

3.5 Test

Test	Ergebnis
Drehencoder langsam nach rechts drehen	Schläger auf dem Display bewegt sich pro Rastung einen Pixel nach unten
Drehencoder langsam nach links drehen	Schläger auf dem Display bewegt sich pro Rastung einen Pixel nach oben
Eingabetaster drücken	Die Punktzahlen setzen sich bis auf den ewigen Highscore zurück
Ewigen Highscore übertreffen	Ewiger Highscore setzt sich um einen Punkt weiter
Der Ball trifft die obere und untere Grenze	Der Ball springt in entgegengesetzte Richtung zurück
Der Ball trifft einen der beiden Schläger	Der jetzige Highscore erhöht sich und der Ball springt in die entgegengesetzte Richtung zurück
Der Ball trifft einen Schläger im oberen Drittel	Der Ball springt nach oben zurück
Der Ball trifft einen Schläger im mittleren Drittel	Der Ball springt nach vorne zurück
Der Ball trifft einen Schläger im unteren Drittel	Der Ball springt nach unten zurück
Der Ball fliegt an einem Schläger vorbei	Der andere Spieler erhält einen Punkt und der Ball setzt sich in die Mitte zurück
Auf einen der Drehencoder drücken und den anderen Drehencoder nach rechts drehen	Der Schläger des gedrehten Drehencoders bewegt sich nun nach oben
Schlägerinvertierung durchführen wie im Test zuvor und 6 Sekunden warten und anschließend den Drehencoder nach rechts drehen	Der Schläger bewegt sich nach unten

3.6 Fazit und Ausblick

Im Großen und Ganzen hat das Projekt Mikrocontroller sehr viel Spaß gemacht, man konnte sich mit anderen Teilnehmern gut über Probleme austauschen und man hat auch gute individuelle Hilfe erhalten, wenn man in einer Klemme steckte oder wenn verschiedene Dinge abzuwägen waren, wie man ein bestimmtes Problem lösen könnte. Als Beispiel hierzu wäre der Einbau einer Sicherung in das PCB mit einer Diode, die im Falle eines falsch Anschließens des Kabels die Sicherung durchbrennen würde.

Als weiteres Feature könnte hier ein Menü eingebaut werden, in welchem der Spielmodus ausgewählt werden kann. Zusätzlich kann noch ein Schalter oder ein Taster eingebaut werden, mit dem die Bewegung des Balls gestartet werden kann, nachdem der Strom eingesteckt wurde.

4 Literaturverzeichnis

- <https://de.wikipedia.org/wiki/Pong>
- https://de.wikipedia.org/wiki/Electrically_Erasable_Programmable_Read-Only_Memory
- <https://youtu.be/bBhbynj6NYM?t=495>
- <https://www.nongnu.org/avr-libc/user-manual/modules.html>