dokumentation Projekt Snake Reproject

Entwicklung eines Snake-spiels basierend auf Arduino

Verantwortlich: Reindl Dominik, Ioannis Christodoulakis  
Revision: A  
Datum: 28.06.2017  
Temperatur: 23.9°C

Cooles ansprechendes Bild

Inhalt

[2 Zusammenfassung 3](#_Toc494379103)

[3 Zeitplan 3](#_Toc494379104)

[3.1 Testfälle 4](#_Toc494379105)

[4 Informieren Arbeitsstand 4](#_Toc494379106)

[5 Testen der Hardware 5](#_Toc494379107)

[5.1 Aufbau und Funktionsweise 5](#_Toc494379108)

[5.2 Test 6](#_Toc494379109)

[5.3 Zusammenfassung 6](#_Toc494379110)

[6 Testen der Auslesesoftware 7](#_Toc494379111)

[6.1 Schnittstelle 7](#_Toc494379112)

[6.2 Ausleselogik 7](#_Toc494379113)

[6.3 Spiellogik 7](#_Toc494379114)

[6.4 Nächste Schritte 8](#_Toc494379115)

[7 Anpassungen 8](#_Toc494379116)

[7.1 Ausleselogik 8](#_Toc494379117)

[7.1.1 Problemstellung 8](#_Toc494379118)

[7.1.2 Struktogramm 8](#_Toc494379119)

[7.1.3 Lösung 8](#_Toc494379120)

[7.2 Spiellogik 8](#_Toc494379121)

[8 Ausführung Testfälle 9](#_Toc494379122)

[8.1 Sonstige Abweichungen 10](#_Toc494379123)

[9 Offene Punkte 10](#_Toc494379124)

[10 Fazit 10](#_Toc494379125)

[11 Zeitplan 11](#_Toc494379126)

[12 Anhang 11](#_Toc494379127)

[12.1 Spieldefinition 11](#_Toc494379128)

[12.2 Gesprächsprotokoll 11](#_Toc494379129)

[12.3 Programmcode 11](#_Toc494379130)

[13 Identifizierung 11](#_Toc494379131)

# Zusammenfassung

Wir haben als Ausgangslage bereits ein Hardwareshield für ein Snake welches für ein Arduino UNO ausgelegt ist. Jedoch läuft das Programm nicht auf der gefertigten Hardware. Mein Auftrag ist es nun den Fehler zu finden und zu beheben. So das am Ende ein funktionsfähiges Snake vorhanden ist.  
Für ein möglichst effizientes Vorgehen haben ich mir folgende Schritte überlegt:

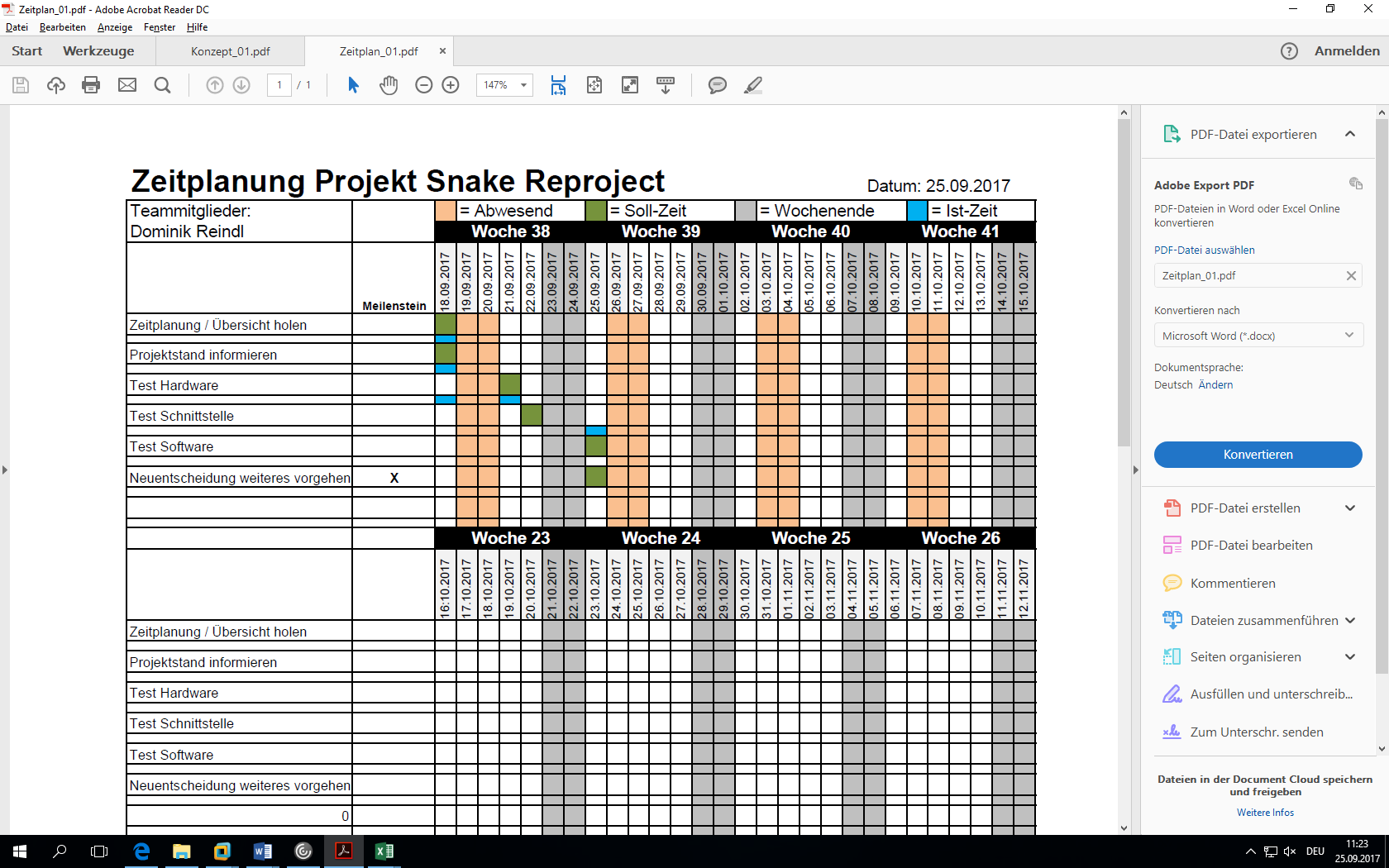
1. Zeitplan erstellen
2. Informieren über den genauen Stand des Projektes.
3. Einarbeiten / Testen der vorhandenen Hardware
4. Einarbeiten / Testen der vorhandenen Software
5. Entscheid weiteres Vorgehen

Nach diesen Schritten kann ich eine definitive Entscheidung treffen ob es für mich möglich ist in vernünftigem Zeitraum das Projekt Snake funktionsfähig zu machen.

Der Zeitplan ist im Anhang angefügt.

Um sich ein Bild zu machen wie der Stand am Anfang des Projektes war, können die Dokumentation vom Projekt «Snake the Game» konsultiert werden.  
Diese sind auf: H:\LL\ELO 2016\Snake\_Reproject zu finden. Mit dem Namen «Gesamtdokumentation Projekt Snake the Game»

# Zeitplan

  
Ich konnte in der Kalenderwoche 38 nicht mit dem Test der Schnittstelle beginnen. Für diesen Schritt benötige ich Hilfe vom Informatiker. Er ist ab Montag am 25.09.2017 wieder Verfügbar. Ich habe am Nachmittag mit ihm einen Termin um weiter voranzukommen. Durch die Verzögerung kann es sein, dass die Neuentscheidung für das weitere Vorgehen erst am 28.09.2017 gefällt wird.  
Die Zeit, welche ich nicht am Projekt arbeiten konnte, habe ich bei anderen Projekten verwendet.

Aktualisiert: 25.09.17

.17

## Testfälle

Zusätzlich haben wir Testfälle definiert um am Ende das Snake Spiel auch auf Funktionalität testen zu können.

* Das Spiel muss am Schluss von unserem Projekt, auf einer 8x8 Matrix mit vier Taster spielbar sein (gemäss Spieldefinition).
* Dieses Spiel soll auf einer Lochrasterplatine gelötet werden, welches wie ein *Shield* problemlos auf das *Arduino* aufsteckbar ist.
* Falls mehrere Taster auf einmal gedrückt und gehalten werden, soll der letzte gelesene Impuls übernommen werden.
* Wenn man nur kurz und ausserhalb von Takt der Schlangenbewegung den Taster betätigt, soll es diese Tasterrichtung speichern und zum Takt ausführen.
* Beim Betätigen von mehreren Tastern nacheinander, ausserhalb des Taktes der Schlangenbewegung, soll es den letzten eingegangener Taster Impuls übernehmen.
* Falls eine in entgegen gesetzter Richtungstaste geklickt wird (in Bezug auf die Schlangenbewegung), soll es keine Richtungsänderung geben.

# Informieren Arbeitsstand

Da ich nicht selbst der letzte Bearbeiter des Projektes bin, muss ich zuerst herausfinden wie weit das Projekt bereits ist. Dafür habe ich mit Herr Marcel Nemeth ein Gespräch geführt. Das Ausführliche Protokoll (Vom 18.09.2017) dazu ist im Anhang aufgelistet.

Die Hardware besteht bereits und wurde auch schon getestet. Jedoch ist das schon länger her und ist somit nicht 100% sicher, dass die Hardware immer noch funktioniert. Deswegen wird ein kleiner Hardwarecheck am Anfang von Nöten sein.

Es wurde nun eine neue Software geschrieben. Diese konnte jedoch noch nicht getestet werden. Deswegen werde ich nach dem Hardwarecheck die Software kontrollieren. Als erstes ohne die fertige Snake Hardware, sondern mithilfe der SerialPrint Funktion. Wenn nach diesem Check die Software in Ordnung ist kann diese mit der fertigen Hardware getestet werden.

Jetzt weiss ich was bereits vorhanden ist und was meine nächsten Schritte sind.

# Testen der Hardware

## Aufbau und Funktionsweise

Um die Hardware zu testen, muss ich mich zuerst mit dem Aufbau der Hardware befassen. Wenn mir der Aufbau klar ist kann ich erst einen sinnvollen Test überlegen für die Hardware.

Für dies habe ich den Schaltplan genauer angeschaut und analysiert.



Ersetzten mit korrektem Schema

Da ich nun wieder im Bilde bin, wie unsere Hardware angesteuert wird, kann ich einen Code Schreiben um diese zu testen.

Das Schieberegister im rotem Rahmen ist zuständig für das Durchschalten der einzelnen Reihen. Dieses Register wird so angesteuert das ein Bit jeweils von Anfang bis zum Schluss durchgeschoben wird. Wenn dieses Bit beim letzten Ausgang angekommen ist, muss ein frisches nachgeschoben werden. Angesteuert wird dieses Register mit den Arduino-Ausgänge 3 und 4.  
Digital Pin 3 = Shift-Register (SRCK) und Input (SER)  
Digital Pin 4 = Storage Register (RCK)  
Die Schiebregister im grünen Rahmen steuern welches LED auf der jeweils aktiven Reihen leuchten soll oder nicht. Es sind zwei Register da insgesamt 16 LEDs (8 rote und 8 grüne LEDs) angesteuert werden. Für diese Register sind die Arduino Ausgänge 5, 6 und 7 zuständig.  
Digital Pin 5 = Input (SER)  
Digital Pin 6 = Shift-Register (SRCK)  
Digital Pin 7 = Storage Register (RCK)  
Die verschiedene Richtungstasten sind auf den Analogen Eingängen A0 – A3 angeordnet. Dabei gilt A0 = Aufwärtstaste  
A1 = Taster für rechts  
A2 = Abwärtstaste  
A3 = Taster für links

## Test

Ich werde folgendes Testen:

1. Kann die LED-Matrix vollständig angesteuert werden? Das heisst jede LED kann entweder rot oder grün leuchten.
2. Sind die Taster noch funktionsfähig?
3. Funktioniert der Batteriebetrieb?
4. Kann beim Batteriebetrieb das Gerät ein und ausgeschalten werden?

Den gesamten Code für das Testen finden Sie im Verzeichnis: H:\LL\ELO 2016\Snake\_Reproject unter dem Namen «Hardware\_Test».

Hier ist nun eine Zusammenfassung der Testergebnisse:

|  |  |  |
| --- | --- | --- |
| Testfall | Ergebnis | |
| LED vollständig ansteuern | Jedes LED konnte die Farbe rot und grün annehmen. |  |
| Taster funktionsfähig? | Jeder Taster wurde erkannt |  |
| Test Batteriebetrieb | Die Hardware wurde auch mit Batterie ordnungsgemäss angesteuert |  |
| Ein- / Ausschalter Test | Die Stromzufuhr konnte wie gewollt zum Arduino unterbrochen werden. |  |

## Zusammenfassung

Die Hardware ist einwandfrei in Takt und einsatzfähig.

# Testen der Auslesesoftware

Da die Hardware keinen Defekt hat, muss das Problem bei der Software liegen. Bei der Software gibt es ebenfalls zwei verschiedene Aspekte. Zu einem gibt es die Spiellogik selbst, welche vom Informatiker erstellt wurde. Der zweite Punkt ist die Ausleselogik. Diese habe ich selbst erstellt. Diese Logik liest ein Array aus und wiedergibt das Bild auf der LED Matrix. Ein Array auszulesen für die LED-Matrix war kein Problem. Jedoch funktionierte das nicht mehr als ich die Ausleselogik in die Spiellogik implementierte. Deswegen habe ich als nächstes die Schnittstelle zwischen der Auslese-Software und Spiellogik genauer betrachten. Dort können viele Fehler entstehen, wenn zum Beispiel nicht sauber besprochen wurde was übergeben wird und wann etc. Genau um dieses Thema kümmere ich mich als erstes bei der Software.

## Schnittstelle

Um hier möglichst schnell Klarheit zu schaffen, machte ich eine Besprechung mit Herr Nemeth (Informatiker). Mit ihm kläre ich die Schnittstelle genau ab um dort auch die Übersicht zu verschaffen.

Ein ausführliches Protokoll ist im Anhang zu finden. Hier werden nun die wichtigsten Erkenntnisse des Gespräches aufgezeigt:

* Zuerst musste klar das Übergabearray definiert werden, aus welchem ich die Daten auslese für die LED-Matrix. Die Arraygrösse wurde nun auf 16x8 (X-Achse x Y-Achse) definiert.
* Das Array muss mindestens 25-mal in der Sekunde ausgelesen werden. Dies ist nötig für eine saubere Darstellung auf der LED-Matrix.

Es wurde schnell klar, dass wir verschiedene Array-Grössen definiert hatten für die Übergabe. So war der erste Fehler bereits gefunden. Für die genug schnelle rate um die Daten auf das Array auszuschreiben, wurde in der Spiellogik die Funktion des Ausgebens mehrmals aufgerufen.

## Ausleselogik

Nun war klar was Übergeben wird und wie ich dies dann bearbeiten muss für die LED-Anzeige. Ich habe ebenfalls die Ausleselogik nochmals angeschaut und Überarbeitet. Anfangs war es nicht möglich das Spiel visuell auf der LED-Matrix darzustellen.  
Ich fand heraus das bei implementieren unserer Ausleselogik in die Spiellogik, die Definition der Portmanipulation vergessen wurde. Nach dem das behoben war, konnte die Bewegung der Schlange auf der LED-Matrix angezeigt werden.

Leider wird auf dem Display die letzte Reihe stärker angezeigt als alle anderen Reihen. Dies muss dann noch angepasst werden.

## Spiellogik

Um das Zusammenspiel zwischen Auslese- und Spiellogik zu vereinfachen wurde am Anfang lediglich die Schlangenbewegung aktiviert.  
Nach den kleineren Anpassungen an der Ausleselogik funktioniert das Zusammenspiel zwischen Auslese- und Spiellogik sowie mit der Hardware. Das heisst es konnte eine Schlange die sich bewegt auf der LED-Matrix steuern.

## Nächste Schritte

Jetzt kommen die Nacharbeiten an die Reihe. Es müssen noch einige Punkte richtiggestellt werden, damit das Spiel vollumfänglich Funktioniert.

* Zeitplan aktualisieren.
* Die Ausleselogik Überarbeiten das alle Reihen gleich stark leuchten.
* In der Spiellogik die Frucht, das Game Over und die Punktestand-Anzeige aktivieren.

Ich selbst bin für die ersten beiden Punkte zuständig. Das Anpassen der Spiellogik macht Herr Nemeth Marcel, meine Informatiker-Ressource.

# Anpassungen

Hier werden nun die bereits beschriebenen Anpassungen dokumentiert.

## Ausleselogik

### Problemstellung

Ich hatte das Problem, das die letzte Reihe ein stärkeres leuchten hatte, als alle anderen Reihen. Ich habe mir den Code der Ausleselogik genauer angeschaut und analysiert. Dabei kam heraus, dass der Anzeigefehler mit der Hardware zusammenhängt. Das 16Bit Schieberegister (2 mal 8-Bit- Schieberegister) speichert welche LED pro Reihe leuchten soll. Diese 16Bit werden immer gespeichert bis die neuen 16Bit eingeschoben werden für die nächste Reihe. So wird die letzte Reihe ebenfalls gespeichert, bis für die erste Reihe die neuen Bits eingeschoben werden. Dies passiert jedoch erst wenn die Funktion für die Ausleselogik erneut aufgerufen wird. Das heisst die letzte Reihe wird im Vergleich zu den übrigen Reihen länger dargestellt. Aufgrund diesen Effekts leuchte die unterste Reihe für uns heller.

### Struktogramm

### Lösung

Also mein Ziel war es, dass auch die letzte Reihe gleich lang leuchtete wie die anderen. Dazu musste ich meine Logik so erweitern, dass die letzte Reihe ebenfalls wieder ausgeblendet wird zwischen den Funktionsaufrufen. Ich habe meine Ausleselogik mit einer Funktion erweitert welche am Ende, wenn alle 8 Reihen angezeigt wurde wieder 16-mal ein «Low» in das Schieberegister speichert. So werden nun alle Reihen gleich angezeigt.

## Spiellogik

Die Spiellogik wird von Marcel Nemeth verbessert und angepasst. Ich habe mit ihm die weiteren Punkte besprochen. Er selbst hat nicht mehr genug Ressourcen zur Verfügung um alle Punkte umzusetzen. Wir haben uns entschieden das die Grundfunktionen sicher noch implementiert werden.

**Wird implementiert:** -Die Frucht, welche zum Sammeln geholt werden müssen.  
 - Das Game Over wenn man sich selbst in den Körper fährt.

**Wird ausgelassen:** - Das Game Over wenn man in den Rand fährt. (Das heisst die Schlange kommt  
 auf der anderen Seite wieder hervor).  
 - Die Anzeige des Punktestandes am Ende des Spiels.

Die Punkte die jetzt noch ausgelassen sind, können auch zu einem späteren Zeitpunkt als eine Aufgabe noch implementiert werden.

# Ausführung Testfälle

Da nun alle Systeme ihre Grundfunktion erfüllen, habe ich die Testfälle, welche am Anfang definiert wurden, ausgeführt. Die Resultate habe ich tabellarisch zusammengetragen.

|  |  |  |  |
| --- | --- | --- | --- |
| Testfall | | Ist-Reaktion | Ok? |
| Das Spiel muss am Schluss von unserem Projekt, auf einer 8x8 Matrix mit vier Taster spielbar sein (gemäss Spieldefinition 🡪 Im Anhang). | | Es ist möglich die Schlange zu steuern mit den  Tastern. Wir lassen die Schlange schneller als der 500ms-Takt laufen, für ein wenig mehr Herausforderung. Die Schlange wird beim Fressen von Futter länger. Das Game Over bei fressen des eigenen Körper funktioniert. Jedoch ist das Game Over beim Berühren vom Rand nicht der Fall.  Die Punktezahlen sind ebenfalls nicht angezeigt am Ende des Spiels.  Entscheid: Auch wenn noch nicht alle Funktionen erfüllt wurden, funktioniert das Grundkonzept von Snake. Man kann durch Fressen von Nahrung grösser werden und die Schlange navigieren. Die weiteren Funktionen können noch im Nachhinein ergänzt werden. |  |
| Dieses Spiel soll auf einer Lochrasterplatine gelötet werden, welches wie ein *Shield* problemlos auf das *Arduino* aufsteckbar ist. | | Ich konnte ohne Probleme die Lochrasterplatine lösen und wieder am Arduino anstecken. |  |
| Falls mehrere Taster auf einmal gedrückt und gehalten werden oder auch nur gedrückt, soll der letzte gelesene Impuls übernommen werden. | | Beim Testen ist aufgefallen das der erste Tastendruck gespeichert und beim nächsten Takt ausgeführt wird..  Entscheid:  Leider ist dies aufgrund der Spiellogik. Für das wäre ein Informatiker zuständig. Da die Informatiker momentan keine Ressourcen zur Verfügung haben und das Spiel auch so gut spielbar ist, belassen wir das so. |  |
| Wenn man nur kurz und ausserhalb von Takt der Schlangenbewegung den Taster betätigt, soll es diese Tasterrichtung speichern und zum Takt ausführen. | | Die Richtung des Tastendrucks wird gespeichert. |  |
| Falls eine in entgegen gesetzter Richtungstaste geklickt wird (in Bezug auf die Schlangenbewegung), soll es keine Richtungsänderung geben. | | Drückt man nur die entgegen gesetzte Richtung, so nimmt die Schlang den Richtungswechseln nicht auf. |  |
|  | = Ist in Ordnung, jedoch gibt es noch Abweichungen | |  |
|  | = Testfall voll erfüllt | |  |

## Sonstige Abweichungen

Ich habe unabhängig der Testfälle das Spiel getestet und verschiedene Situationen ausprobiert. Dabei sind mir noch folgende Punkte aufgefallen:

Das direkte entgegengesetzte fahren der Schlange ist durch die Spiellogik verhindert. Diese prüft ob die geklickte Taste entgegengesetzt der letzten geklickten Taste entspricht. Wenn dies der Fall ist wird die Bewegung verhindert. So kann man nicht eine 180° Wendung machen.  
Jedoch (wir nehmen an das die Schlange nach oben geht) klickt man zwischen dem letzten und dem neuen Takt zuerst nach links und dann gleich nach unten, so wird die Schlange sich beim nächsten Takt nach unten Bewegen und sich selbst gleich fressen. Dies ist der Fall, weil mit dem klicken nach links und dann gleich nach unten bevor die Schlang die Bewegung ausgeführt hat, überlistet man die Prüfung der Spiellogik. Weill jetzt der letzte gedrückte Taster links ist, dann nimmt es den Tastendruck nach unten an und bei nächsten Takt wird die Schlange nach unten gehen.  
Also muss um dieses Problem ebenfalls in der Spiellogik richtig gestellt werden.

Zusätzlich habe ich noch bemerkt, dass wir keine Neustartmöglichkeit haben welcher schnell und einfach betätigt werden kann am Ende des Spiels. Um das Spiel neu zu starten muss entweder die Reset-Taste des Arduino benützt werden oder bei Batteriebetrieb der On/Off Schalter kurz betätigt werden.  
Schön wäre wenn nach dem Spiel eine bestimmte Taster gedrückt werden könnte um es neu von vorne zu beginnen.

# Offene Punkte

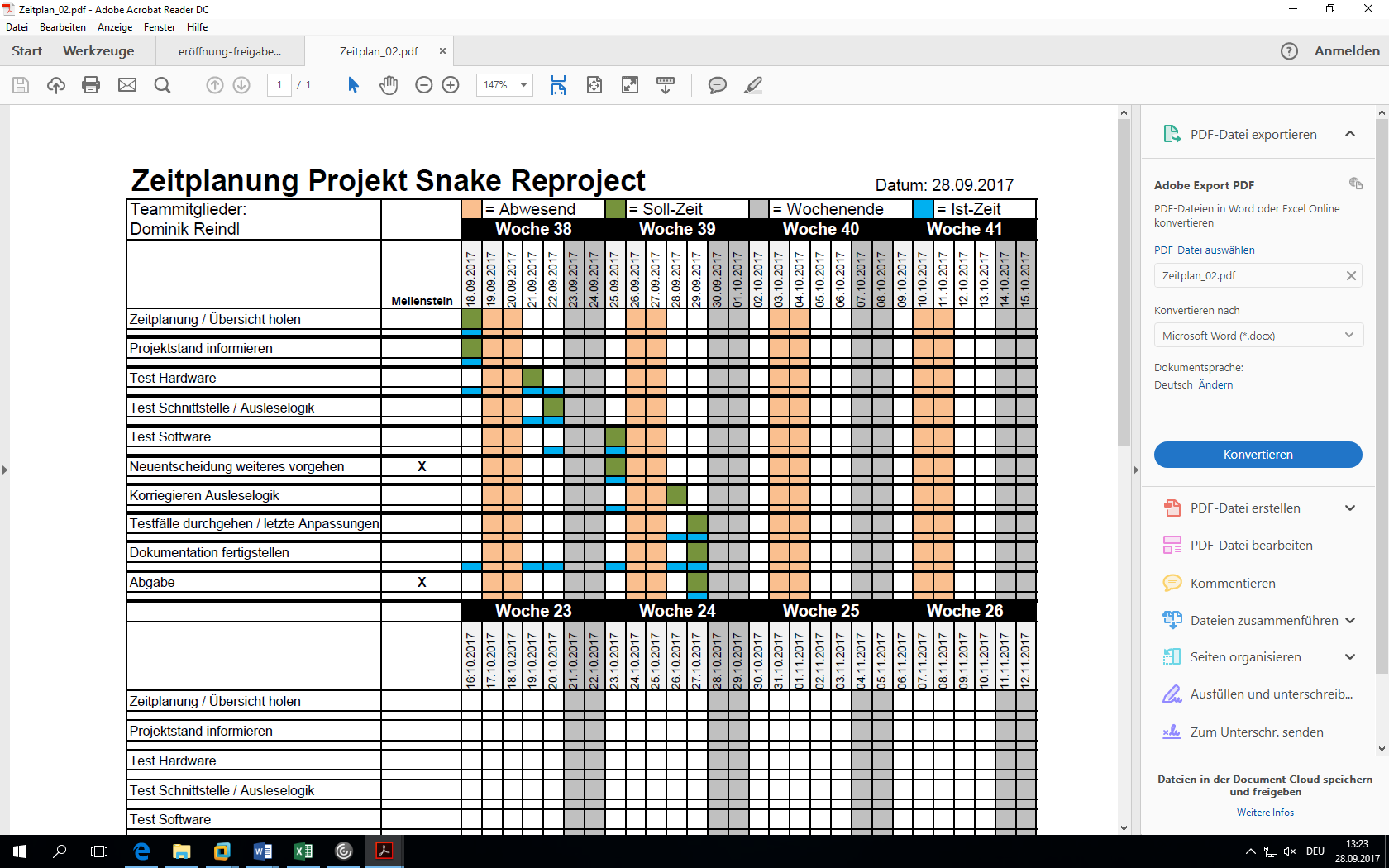
In diesem Kapitel ist nochmals zusammengefasst was am Spiel noch ergänzt werden kann. Dies kann als kleines Zusatzprojekt geschehen, falls jemand genügend Ressourcen zur Verfügung hat.

* Hinzufügen des Game Over beim Berühren der Wand.
* Hinzufügen der Punkteanzeige am Ende des Spiels
* Verhindern das durch Drücken der falschen Kombination die Schlang sich 180° wenden kann. (Im Kapitel 8.1 genauer beschrieben)
* Eine eigene Neustart-Funktion, wenn das Spiel beendet ist.

# Fazit

Man kann ein Snake Spielen. Es ist funktionsfähig und läuft auf einem selbst gelötetem und entwickeltem Shield für das Arduino.  
Jedoch konnten nicht alles Funktionen für das Spiel umgesetzt werden. Es war jedoch möglich mit starker Unterstützung vom Informatiker Marcel Nemeth ein Grundgerüst für Snake zu erstellen. Man kann eine Schlange auf einer LED-Matrix steuern und Futter fressen lassen.  
Die offenen Punkte können gut als Projekt für zukünftige Elektroniker / Informatiker genutzt werden. So kann Stück für Stück ein vollkommen ausgereiftes Snake entstehen.

# Zeitplan



# Anhang

## Spieldefinition

Die Steuerung der Schlange erfolgt durch *Cursor-Pfeiltasten*. Das Spielfeld besteht aus einer LED Matrix mit roten und grünen LED’s.  
Das Ziel ist es mit der Schlange (rote LED’s) die sich im 500ms Takt bewegt, möglichst viele Futterhappen (grüne LED) aufzunehmen.  
Mit der Aufnahme jedes Futterhappens wird die Schlange um ein LED länger.  
Zu beachten ist, dass Berührungen des Spielfeldrandes oder des eigenen Schwanzes zum Tod der Schlange führen und somit das Spiel beenden.   
Am Ende des Spiels werden die angesammelten Futterhappen als Punktzahl auf der Anzeige visualisiert.

## Gesprächsprotokoll

## Programmcode

# Identifizierung