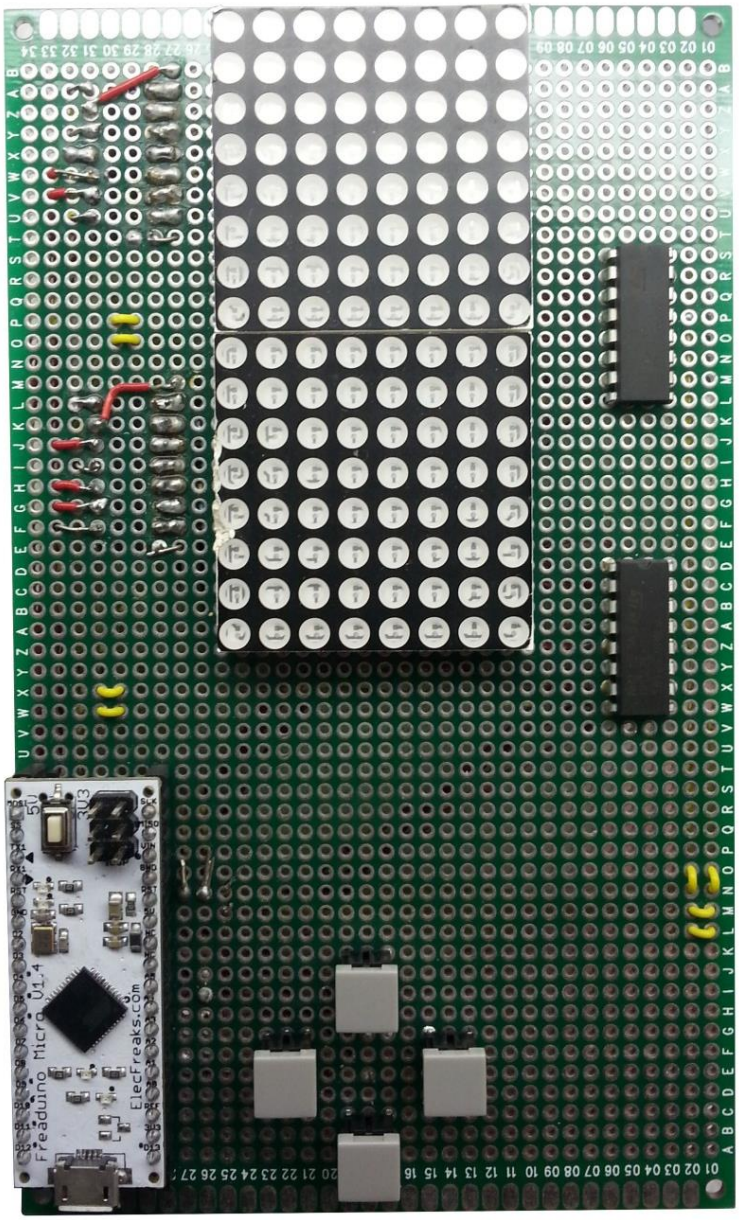
Bau eines Handheld Computerspiels aus elektronischen Komponenten



Gymnasium Kirchenfeld
Abteilung MN
Geschrieben von **Áron Szakács**
Begleitet durch Stefan Rothe
Bern 2014

Inhaltsverzeichnis

Einleitung

Inspiration

Bei der Themenwahl für meiner Maturaarbeit war mir von Anfang an klar, dass es sich um etwas Technisches oder Elektronisches handeln würde. Anfangs hatte ich keine genaue Vorstellung, aber ich war mir bewusst, dass das Programmieren eines Spiels für mich zu einseitig wäre, deshalb war ich auf der Suche nach einer vielseitigeren Aufgabe, wo ich Hard- und Software kombinieren kann.

Mein Informatiklehrer, Herr Rothe, zeigte mir einige interessante Projekte, als ich ihn danach fragte. Seine erste Idee war ein LED-Würfel, welcher aus Leuchtdioden besteht und dreidimensionale Animationen darstellen kann. Dies gefiel mir zwar, jedoch war es nicht das Wahre nach dem ich gesucht hatte. Schliesslich fanden wir auf der Website Youtube ein Video von einer Tetris-Konsole, welche mich sofort begeisterte. Insbesondere sprach mir zu, dass das Projekt zu einer Hälfte aus Hardwarearbeit wie Löten und Verkabeln und zur anderen Hälfte aus Programmieren besteht und somit etwas Abwechslung mit sich bringt. Danach habe ich auch nicht mehr weiter nach Themen gesucht, es war ziemlich klar dass mir dieses Projekt am besten passen würde. Die Fragestellung war schliesslich: „Wie baue ich eine Tetris-Konsole aus elektronischen Komponenten?“, wobei ich dies herauszufinden und auszuprobieren hatte.

Theoretische Grundlagen

Der erste Teil meiner Maturaarbeit bestand darin, die elektronischen Komponenten der Handheld-Konsole zusammenzufügen. Um die Komponenten funktionsfähig verwenden zu können, musste ich mich zuerst mit den theoretischen Grundlagen des Projektes befassen.

Folgende Komponenten sind für die Verwirklichung der Konsole essenziell:

- Zwei 8x8 LED-Matrix
- Pro Matrix zwei Schieberegister
- Eine Lötplatine mit Kabel, Lötzinn und Lötkolben
- Ein Arduino Mikrokontroller
- Tasten (mindestens 2, optimal 4)

Diese Komponenten sind bei play-zone.ch erhältlich.

Abb. 1: 8x8 LED-Matrix [5]

LED - Matrix

Die 8x8 LED-Matrix, einfacher gesagt das Display, wird über 2x8 Pins gesteuert. Eine 8-er Reihe von Pins sind die Eingänge, durch welche Strom ins Display kommt, die andere Reihe bildet die Ausgänge, wo der Strom wieder rausfliesst. Dadurch lassen sich die einzelnen Leuchtdioden ansteuern. Das Prinzip lässt sich vielleicht mit einer Abbildung veranschaulichen:

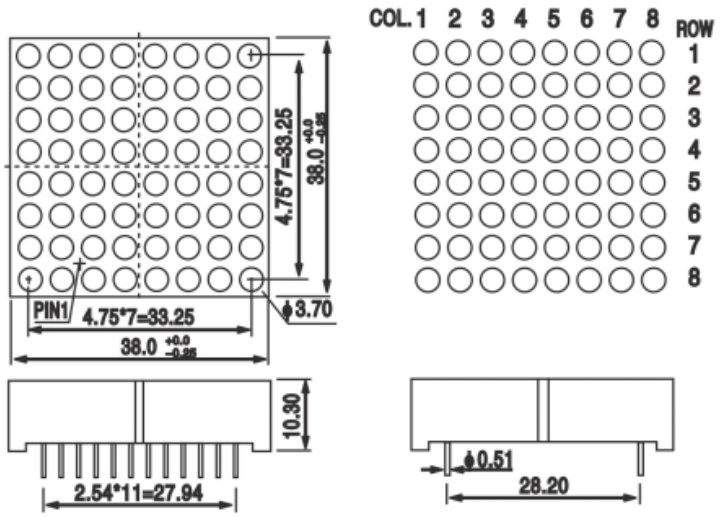


Abb. 2: Auszug aus dem Handbuch der Matrix [4]

Wird nun Strom auf das erste Pin in der Zeile (COL 1) gegeben und wird gleichzeitig das erste Pin aus der Reihe (ROW 1) der Erdung, also 0V, angeschlossen, so sollte nun das Pin (1,1) in der oberen linken Ecke Leuchten. In der Praxis ist dies etwas schwieriger, da die Pins nicht parallel zu den entsprechenden LED-Dioden angeordnet sind. Im Handbuch zur Matrix ist die Anordnung jedoch erklärt.

Schieberegister

Die Schieberegister, welche ich verwendet habe, werden über 3 Pins gesteuert. Dazu brauchen sie eine Stromversorgung über 2 Pins (Siehe Abb. 4: 5V und GND) welche der Bezeichnung entsprechend auf 5V und 0V geschaltet werden. Weiter hat das Schieberegister einen OE und einen MR Pin, welche nur Aktiviert werden, wenn sie keinen Strom haben. MR steht für Master Reset, hätte dieser Pin also keinen Strom, würde alles im Register gelöscht werden. Deshalb muss dieser mit 5V verbunden werden. OE steht für Output Enable, auf Deutsch: Ausgabe erlauben. Dieser muss natürlich auf 0V gesetzt werden, da wir ja Daten über das Register ausgeben können müssen. Die Pins STC, SHC und DS werden vom Arduino angesteuert und die Pins Q0-Q7 sind die Ausgabepins des Registers.

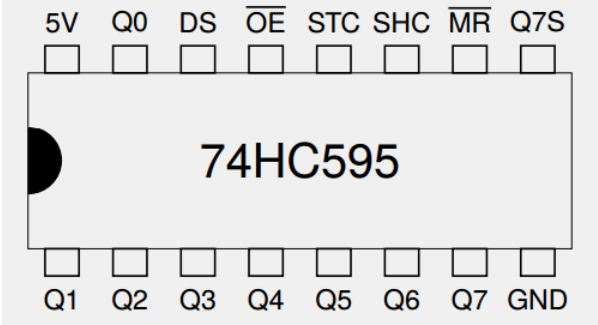


Abb. 3: Pinbelegung eines Schieberegisters [6]

Tasten

Die Tasten der Konsole sind ganz einfache Schalter, welche bei Knopfdruck geschlossen werden. Ein Ende ist an 5V angeschlossen, das andere Ende wird mit einem analogen Pin verbunden, welcher den eingehenden Strom misst. Bei Knopfdruck misst der analoge Pin also 5V, ansonsten 0V. Nebenbei kann er auch Werte dazwischen messen, er gibt diese Werte nämlich als Zahlen zwischen 0 und 1023 heraus.

形名 Part No.	スイッチ特性 Switching function		回路図 Circuit diagrams
		Push	
TR1−01	OFF	(ON)	
接続端子 Connecting terminals	——	1 — 3 2 — 4	

Abb. 4: Grafik zur Funktionsweise der Tasten [2]

Kontroller

Zur Verwirklichung des ganzen Projektes ist ein Arduino, also ein Kontroller, welcher das ganze steuert, essenziell. Mit dem Arduino wurde ich schon im fakultativen Informatikunterricht bekannt gemacht. Er wurde genau für solche Projekte konzipiert. Nun benutze ich einen Freaduino Micro (v1.4), welchen mir Stefan Rothe, mein Informatiklehrer, empfohlen hat, da dieser sehr kompakt und einfach an der Platine zu befestigen ist. Er hat genau 12 digitale Pins, welche ich zur Steuerung der Schieberegister brauche, 6 analoge Pins, wovon ich nur 4 für die Knöpfe brauche, und natürlich je einen 5V und GND (0V) Pin, welche immer und vom Program unabhängig operieren.

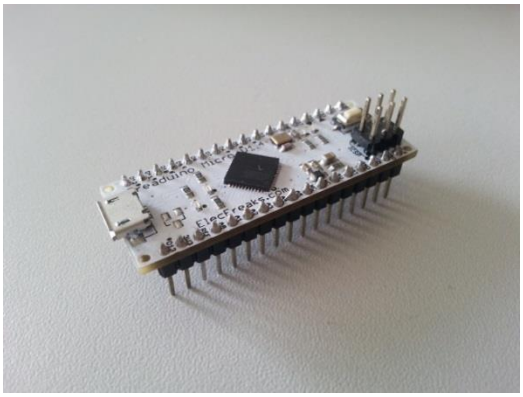


Abb. 5: Freaduino Micro (v1.4) [7]

Angesteuert wird der Freaduino Micro durch ein Micro-USB-Kabel mit dem Computer. Mit der Arduino-Applikation für Windows kann dann die Programmierung am Computer durchgeführt werden und anschliessend wird das Programm auf den Arduino geladen. Währenddessen wird das Programm überprüft und kompiliert, also von Wörtern und Gleichungen in für den Arduino verständliche Maschinensprache übersetzt. Dass ich einen Arduino von einem anderen Hersteller benutze (der Freaduino stammt von der Firma ElecFreaks), ändert nichts an der Projektumsetzung, da der Freaduino genau gleich aufgebaut ist wie der Arduino Leonardo (ein Arduino von vielen), er ist nur etwas kompakter und ist dafür gemacht, angelötet zu werden.

Realisierung

Machbarkeitsstudie

Bevor ich anfangen konnte mit dem Löten und Programmieren, musste ich zuerst wissen, wie ich das Ganze verwirklichen will. Ich musste herausfinden, welche Bauteile ich überhaupt brauchte um 2 LED-Matritzen ansteuern zu können. Um eine Matrix anzusteuern sind ja 16 Pins benötigt, um 2 anzusteuern sind es dementsprechend 32, für jede Display-Achse 8, da es ja 8x8 Matritzen sind. Dass ich also

Schieberegister, welche ein Byte (entspricht 8 Bits) darstellen können, brauchen werde, war mir somit klar. Nun musste ich das Ansteuern eines Displays ausprobieren um dessen Machbarkeit zu überprüfen. Mit einem Breadboard und einigen Kabeln kann man die Komponenten leicht zusammenstecken. Fehler sind so natürlich einfach zu beheben, weshalb es auch Sinn macht, zuerst einmal alles so auszuprobieren, bevor man es dann definitiv auf eine Platine lötet.

Vom Schieberegister zur Matrix

Es sollten also alle Pins der LED-Matrix dem Kontroller angeschlossen werden. Jedoch besitzt der Arduino nicht genügend Anschlüsse, um die 16 Pins anzuschliessen. Nun werden die 2 Schieberegister benötigt. Die Schieberegister sind integrierte Schaltkreise, also Chips, welche eine bestimmte Funktion erfüllen. Es werden Befehle über Pins eingegeben und Resultate über andere Pins erhalten. Ein solches Schieberegister wird über nur 3 Pins gesteuert und kann 8 Werte herausgeben. Somit können wir nun die vorher benötigten 16 Anschlüsse auf nur 6 zusammenfassen, um eine LED-Matrix anzusteuern.

Also kann ein Schieberegister ein Byte speichern, welches es dann auch wieder herausgeben kann, was essenziell ist für die Steuerung des Displays. Beispielsweise kann das Byte 10000000 betragen, also wird der erste Pin auf Strom gesetzt (5V). Nun wird es jedoch etwas schwieriger, denn um die Diode zum Leuchten zu bringen muss der Strom ja hindurchfliessen, also muss beim zweiten Schieberegister das Byte 01111111 betragen, da der Strom von 1 nach 0, von Plus nach Minus fliesst (technische Stromrichtung). Nochmals kurz zusammengefasst: Wird bei der Abbildung 2 in der Einleitung der Pin COL 1 auf 5V gesetzt und der Pin ROW 1 auf 0V, fliesst der Strom durch die Diode (1,1) hindurch und sie leuchtet. Dass bei den anderen Dioden beim Eingang 0V und beim Ausgang 5V herrscht stört sie nicht, sie lassen den Strom nämlich nur in eine Richtung fliessen.

Die Pins, welche unverändert bleiben, wurden schon in der Einleitung erklärt. Nun werden die Pins, welche zum Steuern des Schieberegisters gebraucht werden, angeschlossen, nämlich DS, STC und SHC. Wenn STC („STorage register Clock“) von 0V auf 5V wechselt, werden alle ins Schieberegister geschobene Bits auf den Ausgängen herausgegeben. Deshalb muss STC immer vor der Ausgabe des Bytes, wie im Codebeispiel auf der nächsten Seite, mit der Methode digitalWrite auf LOW (also 0V) gestellt werden. Dann darf mit der shiftOut Methode das Byte Bit für Bit ins Register geschoben werden, welches in der Abbildung 7 in der 2. Zeile an 4. Stelle in der Klammer steht. Das B vor den Bits signalisiert, dass nun ein Byte folgt. LSBFIRST steht für „Least Significant Bit FIRST“ und gibt einfach gesagt an, in welcher Richtung das Byte herausgegeben werden soll, also an welchem Ende des Bytes es mit der Ausgabe anfangen soll. DS und SHC sind die Pins, über welche die Ausgabe dann erfolgt, sie stehen für „Serial Data input“ und „SHift register Clock“. Über DS wird jeweils das Bit herausgegeben, mit SHC wird es dann um eine Stelle verschoben. Der Arduino macht dies automatisch mit der shiftOut Methode.

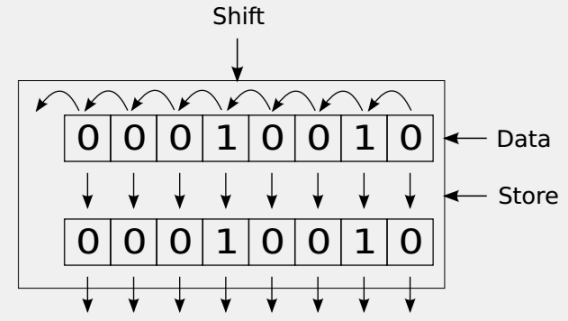


Abb. 6: Mechanik des Schieberegisters [6]

Wenn jedes Bit seriell ins Register geschoben wurde, wird STC wieder aktiviert um das Byte (alle Bits parallel) herauszugeben. Im Codebeispiel sind STC, DS und SHC Variabeln, welche für die Ziffer ihres entsprechenden Pins stehen.

```
digitalWrite(STC, LOW);
shiftOut(DS, SHC, LSBFIRST, B10000000);
digitalWrite(STC, HIGH);
```

Abb. 7: Codebeispiel für die Ausgabe eines Bytes über Schieberegister [1]

Breadboard

Auf der Abbildung 8 ist ein mit dem Programm Fritzing erstelltes Schema zu sehen. Die roten Kabel sind beim Arduino am 5 Volt Anschluss eingesteckt, die blauen am 0 Volt Anschluss. Die drei Pins zum Steuern des Registers sind an den ersten drei digitalen Ausgängen des Arduinos angeschlossen. Die Ausgänge des Schieberegisters sind mit hellgrünen Punkten markiert, es sind dieselben wie auf der Grafik zu den Schieberegistern in der Einleitung.

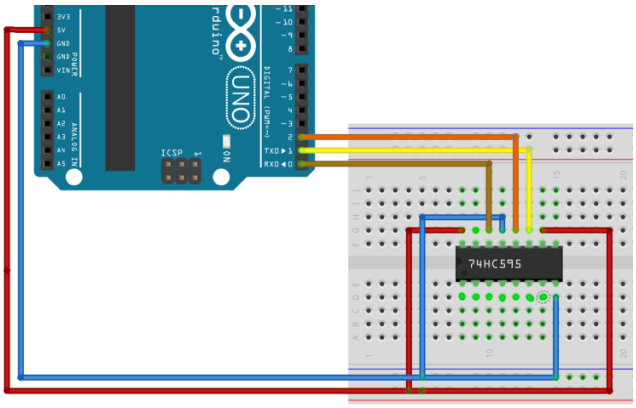


Abb. 8: Schieberegister auf einem Breadboard [7]

Jeder dieser Ausgänge wird nun an einen Pin der Matrix angeschlossen. Hierbei gilt, dass alle Ausgänge des Registers nur an die Pins einer Achse angeschlossen werden dürfen. Die Pins der anderen Achse werden mit einem zweiten Schieberegister angesteuert. Angenommen, über das erste Schieberegister würde die Achse angesteuert, welche den Eingang des Stromflusses bei der Matrix bildet, müsste man beim zweiten Schieberegister noch bei jedem Pin einen 220 Ohm Widerstand zwischen dem Pin der Matrix und dem Ausgang des Schieberegisters einstecken. Würde man dies vernachlässigen, würden die LED-Lämpchen der Matrix überhitzen, da sie maximal 2 Volt aushalten.

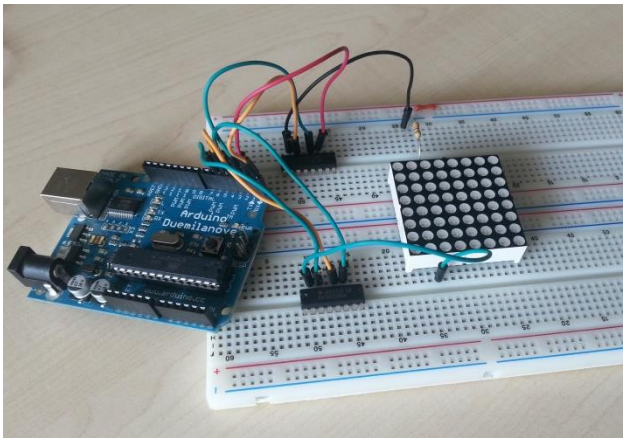


Abb. 9: Schieberegister angeschlossen an die Matrix [7]

Um dies etwas zu veranschaulichen, habe ich bei der Abbildung 9 das Ganze noch einmal für ein einzelnes LED-Lämpchen abgesteckt. Nun muss ich dies für jeden Pin der Matrix wiederholen.

Wie die Pins der Matrix, welche sich an der Unterseite der Matrix befinden, angeordnet sind, kann im Datenblatt zur Matrix nachgelesen werden. Dies ist jedoch bei jeder Matrix verschieden: Zuerst verwendete ich eine blaue Matrix, bei welcher die Pinbelegung ein riesiges Durcheinander war. Bei den

roten Matritzen, welche ich dann beim Projekt auch eingesetzt habe, war die Pinbelegung etwas verständlicher und hatte eine gewisse Reihenfolge.

Nun habe ich also die Pins der Matrix mit den Schieberegistern verbunden, die eine Achse mit dem einen Schieberegister, die andere mit dem anderen. Bei der ersten Achse müssen 5 Volt in die Matrix laufen, bei der anderen müssen 0 Volt angeschlossen sein damit der Strom durch die Matrix fliesst und ein Lämpchen leuchtet. Zwischen dem 0Volt-Register und der entsprechenden Achse der Matrix befindet sich für jeden Pin auch noch ein Widerstand.

Das erste Arduino Programm

Nun ist alles eingesteckt und Funktionsbereit, jedoch weiss der Arduino natürlich nicht, was ich vorhabe. Also muss ich ihn so programmieren, dass er weiss, was er zu tun hat. Eine Leuchtdiode zum Leuchten zu bringen ist nicht besonders schwierig. Wie schon in der Einleitung erklärt, kann dies ganz einfach mit der shiftOut Methode durchgeführt werden. Für das Lämpchen mit der Koordinate (1 , 1) gebe ich beim ersten Schieberegister das Byte (B10000000) heraus, beim zweiten Register (B01111111), genau das Gegenteil des ersten Bytes. Nun fliesst der Strom auf dem ersten Lämpchen von 5 Volt (1) nach 0 Volt (0). Dass auf allen anderen Pins der Strom in die entgegengesetzte Richtung fliesst ist egal: die Lämpchen leuchten nur wenn der Strom in die richtige Richtung fliesst.

Was ist aber, wenn ich mehr als nur ein Lämpchen aufs mal aufleuchten lassen will? Wenn ich die Diagonale der Matrix hell haben möchte, könnte ich nun einfach (B11111111) und (B00000000) herausgeben. Ich spreche ja so immer das Lämpchen mit der Koordinate (1, 1), (2, 2), (3, 3), etc. an. Was aber in der Realität passiert, ist folgendes: Da von jedem Pin einer Achse Strom auf jedes Pin der anderen Achse fliesst, würde einfach die ganze Matrix leuchten, nicht nur die Diagonale. Deshalb muss ein Weg um diesen Fehler herum gefunden werden.

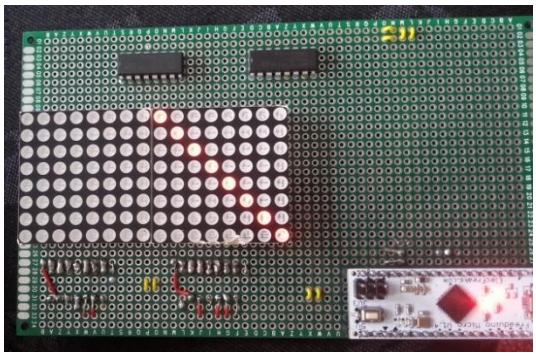


Abb. 10: Diagonale Linie dargestellt auf einer LED-Matrix [7]

Die Lösung dafür ist nicht wirklich schwierig: Anstatt alle benötigten Lämpchen aufs mal aufleuchten zu lassen, spreche ich jedes Lämpchen einzeln an. Also zuerst das Lämpchen (1, 1), dann (2, 2), und so weiter. Wenn ich bei dem Lämpchen mit den Koordinaten (8, 8) angekommen bin, fange ich einfach von vorne an. Dies wiederhole ich in einer unendlichen Schlaufe. Der gesamte Prozess wird vom Arduino schliesslich so schnell ausgeführt, dass wir die Einzelschritte mit unserem Menschlichen Auge gar nicht mehr sehen. Die einzelnen Punkte verschwimmen zu einer Linie auf der Diagonalen. Das ganze Prinzip ähnelt sich einem Daumenkino, wo jedes Bild nacheinander gezeigt wird, so schnell, dass sie dann zu einem Film verschmelzen. Dazu reichen schon rund 25 Bilder pro Sekunde, um aus Einzelbildern einen flüssigen Film zu machen. Je nach dem wieviel der Arduino neben der Darstellung zu rechnen hat, gibt er die Bilder schneller oder langsamer ab, was aber nicht wesentlich ist, da er trotzdem viel mehr als 25 Bilder pro Sekunde auf die Reihe bringt. Auf der Abbildung 10 wird genau dies auf der unteren der beiden Matritzen dargestellt.

Lötprozess

Nach der Machbarkeitsstudie, welche ich hauptsächlich im fakultativen Informatikunterricht durchgeführt habe, war ich also versichert, dass das Ansteuern der Matritzen und das Darstellen von Formen und Figuren möglich ist. Als nächstes war das Bauen der Tetriskonsole angesagt.

Als erstes werden die zwei LED-Matritzen auf die Platine gelötet, dann kommen die Schieberegister, zwei auf einer Seite, zwei auf der anderen Seite der Matritzen. Die Schieberegister, welche Widerstände benötigen, werden auf der Rückseite der Platine befestigt, damit dann diese auf der Vorderseite mit den Widerständen zusammengelötet werden können. Somit ist auf der Vorderseite nur eine Lötzinn-Verbindung sichtbar, die Elektronik befindet sich auf der Rückseite, wie in der Abbildung 11.

Abb. 11: Schieberegister mit Widerständen [7]

Die zwei anderen Schieberegister ohne Widerstände werden auf der Vorderseite befestigt, damit die Kabel dann auf der Rückseite angelötet werden können. Deshalb sieht man in der unteren Hälfte der Abbildung 12 keine Schieberegister, da diese sich auf der Vorderseite befinden. Ausserdem sind die Widerstände direkt an den Pins der Matritzen angelötet, deswegen führen keine Kabel von ihnen weg. Nun sind also die Schieberegisterausgänge mit den Displays verbunden. Jetzt müssen die Schieberegister mit dem Arduino verbunden werden, welcher auch noch aufgelötet werden muss.

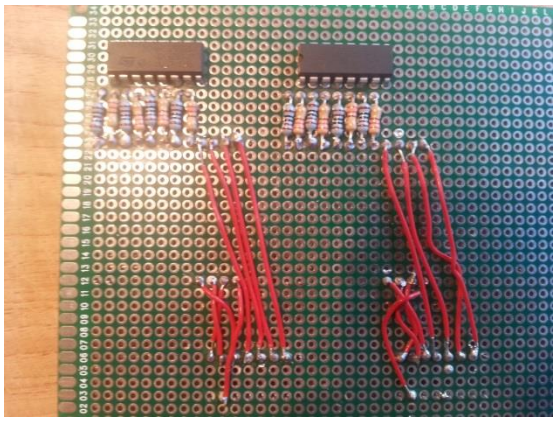


Abb. 12: Verkabelung der Register mit den Matritzen auf der Rückseite [7]

Die Schieberegister können während dem Lötprozess überhitzen, was bei mir zum Glück nicht geschah. Beim befestigen des Arduinos hingegen wollte ich dieses Risiko nicht mehr eingehen, deshalb habe ich für den Kontroller einen Sockel befestigt, in den er dann eingesteckt werden kann. Dieser Sockel ist auf der Abbildung 13 zu sehen. Die Pins des Sockels kann man dann gefahrlos erhitzen.

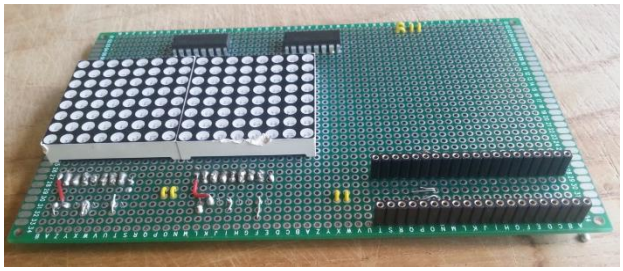


Abb. 13: Vorderseite mit angelötetem Sockel [7]

Auf der Abbildung 14 sind nun auch gelbe Kabel zu sehen, welche die Verbindung zwischen dem 5-Volt Pin des Arduinos mit allen Pins der Schieberegister, welche 5 Volt benötigen, herstellen. Dasselbe gilt auch für die 0 Volt Verbindungen. Unten links auf dem Foto sieht man wie alle Kabel zusammen kommen, der obere Ausgang ist der 5 Volt Ausgang des Arduinos, der Untere 0 Volt.

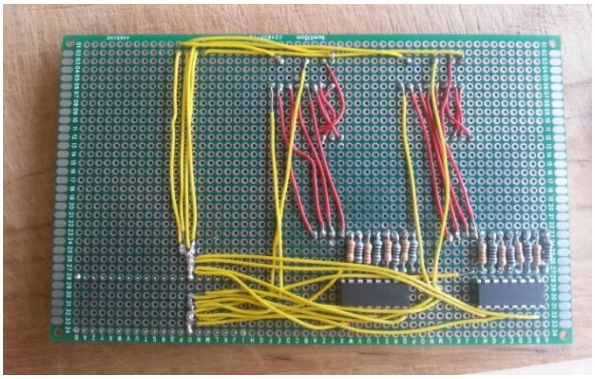


Abb. 14: Verkabelung der Stromversorgung [7]

Nun müssen die Eingänge der Schieberegister (DS, SHC und STC) auch mit dem Arduino-Sockel verbunden werden. Diese habe ich wiederum mit Roten Kabeln mit dem Sockel verbunden. Die vollständige Verkabelung ist auf der Abbildung 15 zu sehen. Die kurzen Kabel links im Bild sind die Verbindungen zwischen den Buttons und den analogen Eingängen, wobei die Buttons noch mit dem 5V-Ausgang verbunden sind, damit der analoge Eingang des Arduinos bei Knopfdruck dann auch 5V misst.

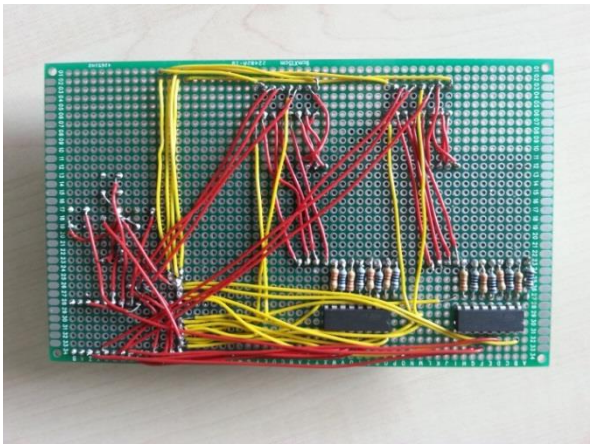


Abb. 15: Vollständige Verkabelung der Rückseite [7]

Programmieren

Das Programmieren des Arduinos erfolgt auf dem Arduino Programm für Windows. Die Arduino Sprache basiert auf C/C++. Auf die Verwendung von Bibliotheken (Libraries), die man selber herstellen kann, habe ich verzichtet, weil ich diese aufgrund des simplen Aufbaus meiner Arbeit nicht notwendig fand. Bibliotheken kann man auch auf der Arduino-Website herunterladen, falls man vor hat ein äusserst komplexes Programm zu schreiben.

Es dauerte einige Zeit, bis ich in der Arduino-Sprache die notwendige Erfahrung sammelte. Sehr viel gelernt habe ich durch die offizielle Arduino-Website (arduino.cc), wo alles sehr gut erklärt und verständlich dargestellt ist.

Display

Das Display wird in einem zweidimensionalen Boolean-Array gespeichert. Dies ist einfach gesagt eine Tabelle mit einer x und y Achse, in welcher die Werte entweder wahr oder falsch sein können. Also kann der Wert mit den Koordinaten (1, 1) entweder wahr oder falsch sein. Zahlen kann man keine einsetzen. So stelle ich dar, ob ein Lämpchen leuchtet (wahr) oder nicht (falsch). Nun wird bei der Display-Methode (der Code-Teil welcher das Display aufleuchten lässt) jeder Wert überprüft. Ist er wahr, wird er herausgegeben. Jeden Wert einzeln zu überprüfen würde aber heissen, dass ich dasselbe 128-mal programmieren müsste. Deswegen mache ich zwei for-Schleiffen, welche ich ineinander verschachtle: In

der ersten Schleiffe wird die Variable y als 0 definiert. Was in dieser Schleiffe geschrieben steht, wird ausgeführt solange y kleiner als 16 ist. Nach jeder Runde wird zu y eins hinzuaddiert. Nun geschieht in dieser Schlauffe dasselbe mit einer Schlauffe mit der Variable x. Für x gelten die gleichen Regeln, jedoch nur solange x kleiner als 8 ist. Also wird das, was in der zweiten Schleiffe steht, nun 128-mal (16 mal 8) ausgeführt. Dort kann ich nun definieren: Wenn in der vorher erklärten Tabelle der Wert mit den Koordinaten (x, y) wahr ist, soll das LED-Lämpchen mit denselben Koordinaten aufleuchten. Wenn der Wert nicht wahr ist, geschieht nichts. So wird also jedes y und jedes x einmal geprüft, und falls wahr, wird das entsprechende Lämpchen erleuchtet.

Da diese Methode etwas uneffizient ist, weil jedes Pixel maximal in einem 128-stel der Zeit angezeigt werden kann, wurde die Methode schlussendlich so verbessert, dass anstatt einem Pixel nun eine ganze y-Spalte aufs mal Dargestellt wird. Somit ist zuerst die x-Koordinate 1 aktiviert, dann werden alle y, welche wahr sind, auch aktiviert, die falschen bleiben unbeleuchtet. Dann geschieht dasselbe für x=2, usw. So wird das Programm ungefähr 16-mal schneller, wobei zu beachten gilt, dass immer noch 2 Schieberegister für die x-Koordinaten benutzt werden (pro Matrix eines).

Tetris Blöcke

In Tetris fallen bekanntlich Blöcke, auch bekannt als Tetrominos, welche man auf einer Reihe anordnen muss, um Punkte zu sammeln. Jeder Block hat einen Namen, es gibt insgesamt 6 Blöcke: O, S, Z, J, L, I und T. Die Blöcke sind nach ihrem Aussehen benannt, wie man es in der Abbildung 16 erkennen kann. Die Tetrominos J und Z sind die Umkehrvarianten von L und S.

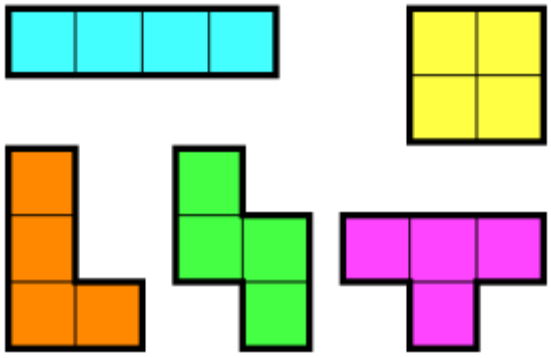


Abb. 16: die Tetrominos I, O, L, S und T [3]

Um die Tetris Blöcke darzustellen, brauche ich nun wieder ein Array, welches nun aber aus Integers, also ganzen Zahlen, anstatt Booleans besteht. Jeder Tetris Block besteht aus 4 Pixeln, deshalb enthält das Array 4 Koordinatenpaare, eines für jedes Pixel. Welche Koordinate jedes Pixel hat, damit die Blöcke auch so aussehen, wie sie sollten, ist im Programm definiert.

Die Tetrominos fallen jede halbe Sekunde um ein Pixel herunter. Dies wird erreicht, indem jede halbe Sekunde die y-Koordinaten der Pixel des Steines um 1 erhöht werden. Dabei muss gelten, dass die y-Achse nach unten zeigt, anstatt, wie es in der Mathematik üblich ist, nach oben zu zeigen.

Bewegt werden die Blöcke mit den Links- und Rechts-Tasten. Jedes mal, wenn eine Taste gedrückt wird, bewegt sich der Block um einen Pixel entweder nach links, oder nach rechts. Wird die nach Unten gerichtete Taste gedrückt und gehalten, wird die Geschwindigkeit, mit welcher der Block fällt, erhöht.

Rotiert werden die Tetrominos, indem jedes ihrer Pixel so verschoben wird, dass sie die erwünschte Form annhemen. Dafür habe ich für jede Rotation jedes Blockes eine Methode erstellt, die die

erwünschten Koordinaten verändert. Der O-Block muss nicht rotiert werden, da er immer gleich aussieht. Die Blöcke I, S und Z haben nur 2 mögliche Positionen, wogegen die Blöcke L, J und T ganze 4 mögliche Positionen besitzen.

Um die Blöcke darzustellen, werden immer vor dem Anzeigen die Koordinaten in das Display-Array übertragen. Immer vor dem Anzeigen wird das Display-Array neu aus den Koordinaten des Blocks und aus dem Stapel-Array, welches die schon vorhandenen Pixel unten auf dem Stapel beinhaltet, berechnet. So kann der Stapel verändert werden und der Tetromino verschoben werden oder fallen, ohne dass jemals etwas falsches angezeigt wird.

Stapel

Der Stapel besteht aus einem exakt gleichen Boolean-Array wie das Display. Der einzige Unterschied besteht darin, dass der Stapel den Block in Bewegung nicht beinhaltet. Im Stapel befinden sich nun alle Pixel der Blöcke, die unten auf der Anzeige angekommen sind.

Spielmechanik

Ob ein Block auf dem Stapel angekommen ist, wird laufend überprüft. Falls dies zutrifft, wird ein neuer Block generiert und der alte zum Stapel hinzugefügt. Gleichzeitig wird dauernd geprüft, ob sich im Stapel eine vollständige Zeile befindet. Trifft dies zu, wird diese gelöscht. Nun dürfen die Blöcke natürlich weder in den Stapel von der Seite hineinbewegt oder rotiert werden, noch aus der Anzeige herausbewegt oder rotiert werden. Deshalb wird bei jeder Bewegung oder Rotation geprüft, ob sich der Block innerhalb des Stapels oder ausserhalb des Spielfeldes befindet. Falls dies zutrifft, wird die ausgeführte Aktion sofort rückgängig gemacht. Dasselbe geschieht beim Fallen der Steine, nur wird bei dem Kontakt mit dem Stapel der Block wieder nach oben Bewegt und dann auf den Stapel gesetzt.

Fazit

Ziel dieser Maturaarbeit war es, herauszufinden, wie man eine funktionsfähige Tetriskonsole bauen und programmieren kann. Mithilfe eines Arduinos, 2 LED-Matritzen, 4 Schieberegistern und 4 Tasten konnte die Hardware aufgebaut werden. Mit der Arduino Entwicklungsumgebung wurde dann das Spiel programmiert.

Rückblickend auf die Bauphase bietet sich aus heutiger Perspektive vor allem eine Optimierungsmöglichkeit bei der Funktionsweise der Spieldarstellung. Diese besteht in der Aneinanderreihung der beiden Schiebereigster für die y-Achse mit dem bisher unbenutzten Q7S-Anschluss, wodurch ein 16-Bit Schieberegister erschaffen wird. Nach diesem Lösungsansatz stellt man mit nur einem Schieberegister die x-Achse beider Matritzen gleichzeitig dar, ein Schieberegister und somit 3 Arduino-Pins erspart. Dies resultiert auch in einer effizienteren Gestaltung des Programms mit einem Schieberegister weniger, welches den Aufwand für die Darstellung der x-Achse zur Hälfte reduziert.

Nach Beendigung der Arbeit mit der gesammelten Programmiererfahrung stellte ich beim Überblicken des Codes fest, dass mir oft effizientere und platzsparendere Lösungen einfielen. Aus der Sicht der Programmgestaltung habe ich beispielsweise rückblickend festgestellt, dass in einigen Fällen

Gleichungen oder mathematische Überlegungen in Methoden oder Formeln hätten zusammengefasst werden können, anstatt diese bei jeder Anwendung zu kopieren. Damit wäre der Code etwas übersichtlicher gewesen. Auch bei der Variabelbenennung bieten sich eindeutigere Namen, vor allem schon weil die Entwicklungsumgebung weder Variabeln noch Methoden hervorhebt.

Im Nachhinein stelle ich fest, dass ich während des Arbeitsprozesses sehr viele Erfahrungen gesammelt habe. Weiter sehe ich auch die Ungründlichkeiten, welche am Anfang der Arbeit entstanden sind, ein. Rückblickend liessen sich ein paar Sachen verbessern, jedoch werde ich die neu gesammelte Erfahrung und das Gelernte in Zukunft verwenden und das nächste ähnliche Projekt mit Sicherheit effizienter und besser machen können. Ausserdem ist mir der Bau einer vollständig fehlerfreien und funktionsfähigen Tetriskonsole gelungen.

Abbildungs- und Quellenverzeichnis

[1] Arduino, «shiftOut,» [Online]. Available: http://arduino.cc/en/Reference/shiftOut. [Zugriff am 2 10 2014].

[2] Copal Electronics, «Super-miniature Illuminated Pushbutton Switches TM & TR».

[3] Wikipedia, «Tetromino,» [Online]. Available: http://en.wikipedia.org/wiki/Tetromino. [Zugriff am 8 11 2014].

[4] Guangzhou Linong LightingTechnology Co., LTD, «TYPE: LNM1588BB2C».

[5] Play-Zone, «Ultra Bright 8x8 LED-Matrix Blau, 3.8 x 3.8 cm,» [Online]. Available: http://www.play-zone.ch/de/ultra-bright-8x8-led-matrix-blau-3-8-x-3-8-cm.html. [Zugriff am 16 4 2014].

[6] T. Jampen und S. Rothe, «Digitalelektronik 2: vom Transistor zum Byte,» Bern, 2014.

[7] Á. Szakács, *Abbildung.*

Eigenständigkeitserklärung

Hiermit bestätige ich, dass ich die Arbeit „Bau eines Handheld Computerspiels aus elektronischen Komponenten“ selber geschrieben, gestaltet und erstellt habe. Alle verwendeten Hilfsmittel und Grafiken sind im Abbildungs- und Quellenverzeichnis angegeben.

Ort und Datum

Unterschrift

Áron Szakács