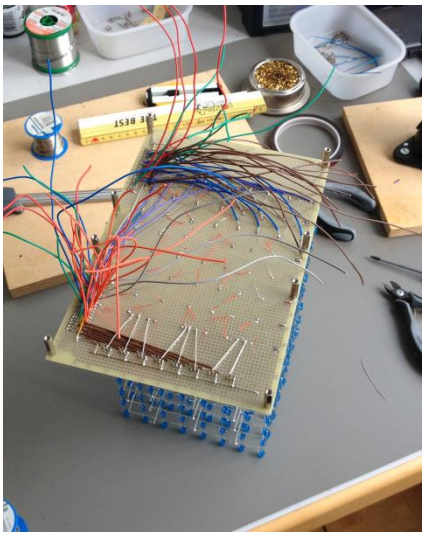
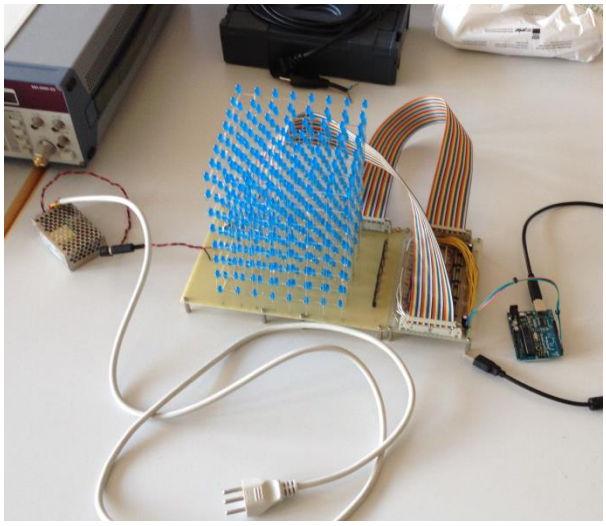
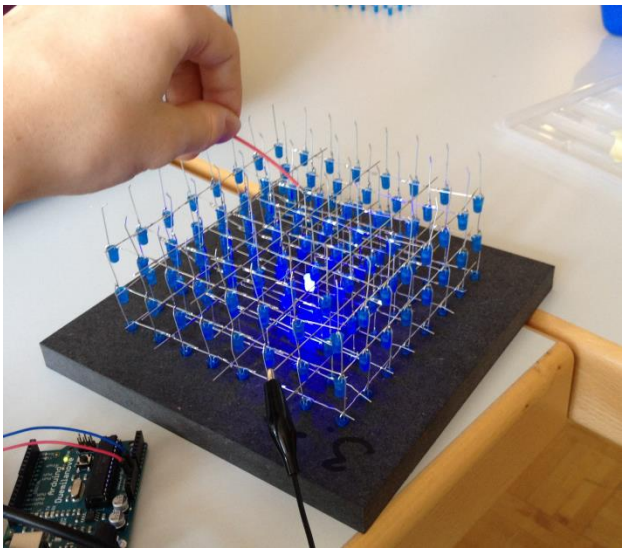
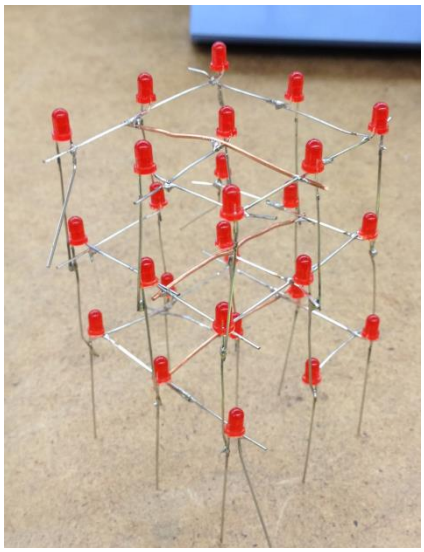
Der Bau eines LED-Würfels









Maturaarbeit am Gymnasium Kirchenfeld (MN) von Peter Werner

betreut von Stefan Rothe

9. Oktober 2014

Inhalt

Einleitung

LED Würfel sind dreidimensionale Matrizen aus Leuchtdioden, welche dreidimensionale Bilder generieren können. LED-Würfel mögen zuerst nach einer schönen Spielerei aussehen, aber darin sind essenzielle Methoden aus dem Alltag der Elektronik sehr anschaulich verkörpert. Zum Beispiel das Konvertieren von langen Datensträngen in eine parallele Ausgabe, die Ansteuerung von Speicher oder etwa Multiplexing, das Zusammenfassen von mehreren Signalen, kommen heutzutage überall vor. Das Praktische an den LED-Würfeln ist auch, dass die benötigten Komponenten dafür leicht erwerbbar sind und man LED-Würfel somit gut als „Do-it-yourself"-Projekt selber bauen kann. Die vielen LEDs stabil und funktionsfähig aneinander zu löten, eine Ansteuerung zu bauen und selber die Software dafür zu schreiben ist jedoch nicht einfach.

Ich beschloss, mir im Rahmen meiner Maturaarbeit einen LED-Würfel von Grund auf aufzubauen, die Schaltung zu entwerfen und die Software dafür zu schreiben.

In diesem Projekt baute ich einen 8x8x8 Würfel. Der Würfel besteht also aus 512 LEDs. Diese vielen Lämpchen müssen auf irgendeine Art sinnvoll und effizient angesteuert werden. Dies erfolgt über mehrere 8 Bit-Schieberegister. Die Schieberegister werden dann schlussendlich von einem Arduino- Mikrokontroller aus angesteuert.

Die Durchführung dieses Projekts verlief in verschiedenen Phasen, in welchen es immer neue und mir unbekannte Probleme zu überwinden gab. Beim Entwickeln und Durchführen dieses Projekts habe ich deshalb auch viele Gebiete der Digitalelektronik und den Entwicklungsprozess eines programmierbaren Objektes etwas kennengelernt. Die Durchführung des Projekts ist in den folgenden Kapiteln dokumentiert und erklärt.

1. Grundlagen

Um die folgenden Kapitel etwas besser verständlich zu machen, werden in diesem Kapitel die notwendigsten Grundlagen erklärt.

1.1 Ohm'sches Gesetz

Um Berechnungen für den Stromkries des Würfels machen zu können, werden einige Grundlagen aus der Elektronik verwendet.

Stromkreise bestehen aus einer Stromquelle, welche Elektronen zum Fliessen zwingt, und einem oder mehreren Verbrauchern, welche die Elektronen abbremsen. Das Ohm'sche Gesetz [1] besagt, dass die Anzahl Elektronen, die pro Zeiteinheit durch den Draht fliessen (Stromstärke [I] = Ampère), proportional zur Elektronendichtedifferenz (Spannung [U] = Volt) ist. Man fand schliesslich heraus, dass diese Proportionalitätskonstante, mit der richtigen Umformung, dem Widerstand ([R] = Ohm) in einem einfachen Stromkreis entspricht. Die Formel dafür sieht folgendermassen aus.

$$\frac{U}{I} = R\ (= konstant)$$

Nun lassen sich mit dieser Formel im Stromkreis alle wichtigen Werte, bis auf die Leistung, berechnen. Die Leistung ergibt sich aus dem Produkt der Spannung und Stromstärke.

$$UI = P$$

1.2 Leuchtdioden

Wie der Name schon sagt, ist das Hauptelement des Würfels die lichtemittierende Diode (**L**ight **E**mitting **D**iode). Sie besteht aus einer sogenannten Silizium-„PN-Junction". PN-Junctions sind Übergänge zwischen zwei verschiedenen Siliziumarten [2], nämlich P- und N-Silizium. P und N stehen für positiv und negativ. Die zwei Arten werden so hergestellt, dass man einfach Fremdatome, die entweder ein Valenz-Elektron mehr oder weniger haben als die Siliziumatome, mit den Siliziumatomen vermischt. Somit werden dann im Siliziumkristall entweder losgelöste Elektronen frei oder es entstehen sogenannte „holes", welche Lücken auf der Valenzschale der Atome sind. Die „holes" können ohne Energieaufwand durch die Aufnahme von anderen Elektronen eines benachbarten Atoms, an ein anderes Atom im Gitter weiter gegeben werden. Nun werden in Dioden eben diese zwei Arten von Silizium zusammen gefügt und es entsteht ein Halbleiter, welcher Strom nur in eine Richtung durchlässt, nämlich nur dann, wenn an der P-Seite eine positive Ladung und an der N-Seite eine negative Ladung angebracht werden. Bei gewissen Materialien wird durch den Prozess des Kombinierens von einem Hole mit einem Elektron für uns sichtbares Licht abgegeben. Verwendet man solche Materialien als Unreinheiten im Silizium, hat man den Grundbaustein für eine LED. Sie lässt also nur Strom in eine Richtung durch und gibt Licht ab, wenn Strom fliesst.

1.3 Schieberegister und Ansteuerung

Grundbaustein für die programmierbare Ansteuerung für den Würfel und eigentlich alle digitalen Ansteuerungen ist der Transistor. Transistoren sind eine Art Schalter [3], vergleichbar mit einem Ventil für Elektronen, welches mit Strom geöffnet und geschlossen werden kann. Dabei wird die Stromstärke zwischen E (Emitter, also dem Minuspol des Transistors) und C (Kollektor, Pluspol des Transistors) kleiner oder grösser. Das Praktische daran ist, dass man dadurch mit sehr kleinen elektrischen Strömen sehr grosse genau steuern kann. Es gibt unzählige Variationen, jede mit ihren Vor- und Nachteilen. Alle aber haben eine ähnliche Aufgabe, nämlich mit einem schwachen Strom einen starken zu regulieren. In Abb1 ist dieses Prinzip an einem Beispiel mit Wasser statt Strom visualisiert.

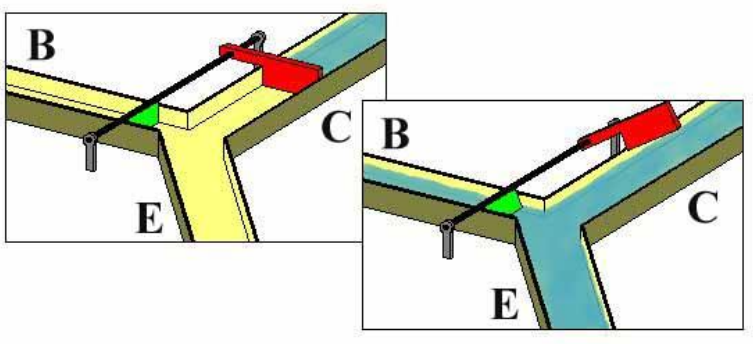


Abb. 1 Visualisierung der Funktion eines Transistors

Aus Transistoren lassen sich sogenannte Logikgatter aufbauen. Logikgatter sind Schaltkreise aus Transistoren, die immer eine oder mehrere Eingaben und Ausgaben haben. Je nach Eingabe verändert sich dann die Ausgabe. Nehmen wir als Beispiel das AND-Gatter. Das AND Gatter hat typischerweise 2 Eingänge und einen Ausgang. Nur wenn bei beiden Eingängen eine Spannung von 5V (eine übliche Spannung in der Digitalelektronik) angelegt wird, stellt sich der Ausgang auf 5V, sonst ist der Ausgang auf 0V (in Realität sind es nie genau 0V).In der Digitalelektronik ist oft auch die Rede von Einsen und Nullen, die durch 5V und 0V repräsentiert werden.

Schlussendlich gibt es viele verschiedene logische Schaltungen mit beliebiger Komplexität, sie lassen sich aber alle auf andere, einfachere Gatter reduzieren. Nun kann man aus diesen Gattern ganze Computer mit Rechner und Speicher aufbauen.

Im LED-Würfel sind nicht nur im Mikrocontroller, der den Würfel steuert, solche Gatter vorhanden, sondern um überhaupt alle Lämpchen an steuern zu können werden zusätzliche Zwischenspeicher verwendet. Für diese Aufgabe eignen sich sogenannte Schieberegister. Schieberegister sind Speicherchips [4], die serielle Ausgaben (Bits nacheinander) in parallele Ausgaben (Ausgabe gleichzeitig) umwandeln können. Sie funktionieren folgendermassen: Alle Schieberegister sind mit Taktflankenschaltungen ausgestattet. Taktflankenschaltungen werden durch ein Taktsignal aktiviert. Ein Taktsignal ist ein Signal, das immer von 0V zu 5V springt und wieder zurück. Die Schaltung registriert den Takt immer bei der ansteigenden Flanke, also wenn die Spannung ansteigt.

Um die Schieberegister anzusteuern, werden zuerst der Reihe nach die Datenbits in das Register hinein geschoben und danach parallel ausgegeben. Die Schieberegister haben grundsätzlich einen Ausgabekontakt für alle Speicherstellen. Dazu kommen 3 Eingabekontakte: Ein Datenkontakt (Datapin), ein Taktkontakt (Clockpin), ein Speicherkontakt (Latchpin) und andere Kontakte die z.B. den beschriebenen Speicher löschen. Um Daten zu speichern, stellt zuerst die Datenleitung das nächste Bit ein. Danach springt das Taktsignal, welches grundsätzlich auf 0 ist, kurz auf 1 und wieder zurück auf 0. Der Wert auf der Datenleitung wird in die erste Speicherstelle des Schieberegisters übertragen. Dieser Prozess wird solange wiederholt, bis alle Bits übertragen wurden und das Schieberegister voll ist. Danach kriegt der Speicher oder sogenannte „Latch“-Pin einen kurzen Impuls (gleich wie beim Taktsignal von 0 auf 1) und die hereingeschobenen Daten werden auf den Ausgabe-Pins gleichzeitig herausgegeben (Das erste Bit auf dem letzten Pin, das zweitletzte Bit auf dem zweitletzten Pin usw.).Die Speicherchips haben Einschränkungen die für den späteren Aufbau des Würfels von Bedeutung sein werden. Die Ausgabepins können nur gewisse Stromstärken in beide Richtungen durchlassen. Dabei spricht man von Stromsenke (wenn der Ausgabe Pin als Pluspol agiert)und Quellfunktion (wenn der Ausgabe Pin als Minuspol agiert).

Schieberegister können auch aneinander gekettet werden und so über drei programmierte Output Signale unendlich viele parallele Outputs generieren. Um das zu machen hat jedes Schieberegister einen Kontakt, welcher permanent den Wert an der letzten Speicherstelle herausgibt. Dieser Pin wird mit Serial Out bezeichnet. Um eine Kette zu machen schliesst man nun das erste Schieberegister ganz normal an. Bei den folgenden Gliedern schliesst man immer den Serial Out-Anschluss des vorderen Schieberegisters mit dem Daten-Input des nächsten Chips zusammen. Zum Schluss muss man nur noch die Takt- und die Speichersignale des programmierten Outputs mit allen Schieberegistern verbinden. Die ganze Kette kann dann gleich wie ein einziges Schieberegister angesteuert werden.

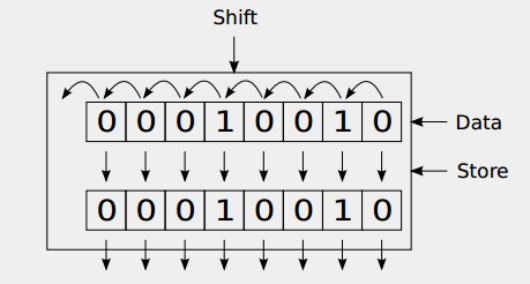


Abb. 2 Schieberegister Funktion

1.4 Arduino

Um sämtliche Schieberegister anzusteuern, wird in diesem Projekt der Arduino Duemilanove verwendet. Der Arduino ist ein Mikrocontroller. Ein Mikrocontroller ist ein kleiner Computer [5], welcher alle Komponenten in einem einzigen integrierten Schaltkreislauf hat. Er hat eine programmierbare Ein- und Ausgabe, eine CPU und einen Speicher. Schlussendlich hat er in diesem Projekt die Aufgabe, die Daten in die Schieberegister zu schieben und zu speichern. Der Arduino kann mit der Programmiersprache C oder C++ programmiert werden.

Um noch kurz auf diese Programmiersprachen einzugehen: Bei meinem Programm werden nur ganz einfache Operationen der Arduinobibliothek [6] verwendet. Darunter mathematische Operationen mit Variablen durchführen, Schleifen (wiederholte Ausführung von Befehlen) und einen Ausgabebefehl für die Pins des Arduinos.

2. Funktion des Würfels

Um die Funktion des Würfels zu verstehen, muss man beim Aufbau des Würfels beginnen. Alle Kathoden der LEDs sind beim 8x8x8 Würfel in 64 Spalten (Abb. 4 [7]) und alle Anoden in 8 Ebenen (Abb. 3 [7]) gelötet.

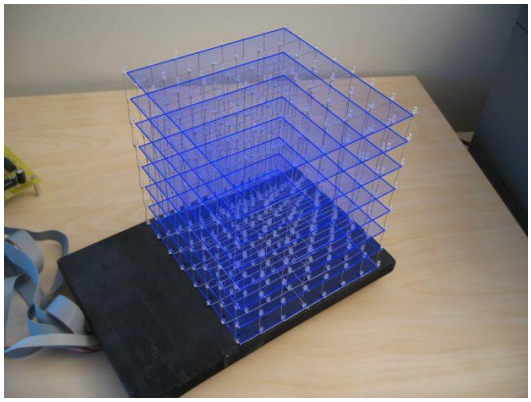


Abb. 4 Anode Ebenen am 8x8x8 LED Würfel

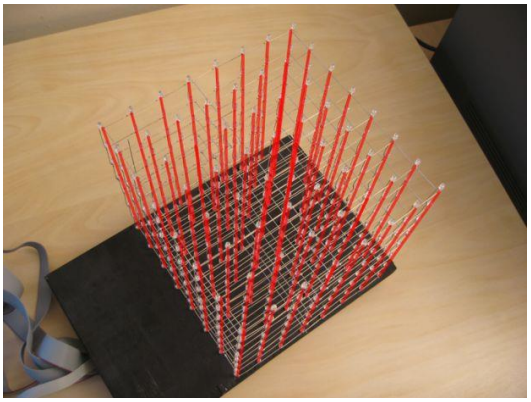


Abb. 3 Kathoden Spalten am 8x8x8 LED Würfel

Mit diesem Aufbau spart man Platz, da man nur noch 64 Kathoden und 8 Anoden ansteuern muss, statt je 512. Diese Architektur verhindert das unabhängige Aufleuchten von LEDs auf unterschiedlichen Ebenen (z.B. in einer Animation). Am besten sieht man dies an einem Beispiel. Nimmt man zwei Spalten des Würfels und lässt bei der einen Spalte immer abwechslungsweise ein Lämpchen leuchten und eines nicht (d.h. 4Lämpchen leuchten) beginnend bei einem leuchtenden Lämpchen. Das Muster wird einwandfrei angezeigt. Wiederholt man aber dieses Muster an der zweiten Spalte, aber um ein Lämpchen verschoben (d.h. beginnend bei einem erloschenen Lämpchen), so wird keines der beiden Muster richtig angezeigt. Es werden nämlich alle Lämpchen in den beiden Spalten leuchten. Das folgt daraus, dass alle Anoden mit 0V verbunden werden müssten, da mindestens 1 Lämpchen pro Ebene leuchten sollte und beide Kathoden mit 5V verbunden werden müssten, da es in jeder Spalte Lämpchen gibt, die Leuchten sollten. Das heisst, es gibt eine Spannung über alle Lämpchen in den beiden Spalten und sie leuchten alle.

Um alle Lämpchen gezielt und gleichzeitig aufleuchten zu lassen, greift man auf einen Trick zurück der fast überall in der heutigen Welt der Digitalelektronik verwendet wird. Dieser Trick heisst Multiplexing [8]. An diesem Beispiel funktioniert das so: Man leuchtet alle Ebenen abwechslungsweise auf. Geschieht dies mit einer genügend grossen Frequenz, so wird der ganze Würfel von unserem Auge als leuchtend empfunden. Dieses Prinzip wird bei jedem Monitor verwendet. Der einzige Nachteil von diesem Verfahren ist, dass der Würfel nicht mehr so hell leuchten kann, da die Lämpchen nur einen Achtel der normalen Zeit leuchten.

Um die LED-Matrix mit dem Mikrokontroller zu verbinden, braucht man bei diesem Aufbau 72 programmierbare Pins. Da der Ardunio aber nicht so viele Kontakte hat, ist die Ansteuerung auf Schieberegister ausgelagert. Wie bei den Grundlagen erklärt, kann man mittels der drei Pins des Mikrokontrollers sehr viele programmierbare Pins generieren. Ich habe eine Kette von neun 8-Bit-Schieberegistern ver-

wendet. Bereits hier muss etwas spezifiziert werden, damit weitere Teile der Ansteuerungselektronik verständlich werden. Die Lämpchen, die ich verwende, müssen mit einer Stromstärke von 20 mA versorgt werden. Das bedeutet, dass über die 8 Anoden des Würfels je eine maximale Stromstärke von (64x20 mA) 1.28 A fliessen wird. Da es ungünstig ist, mit einem Schieberegister diese Stromstärken auf 0V zu senken, wird dies über 8 Transistoren gemacht, die solche Stromstärken ohne weiteres ertragen können. Die 64 Kathoden hingegen können direkt mit den Schieberegister Outputs verbunden werden. Die verwendeten LEDs haben eine Spannungsgrenze von 3.4V und die Ausgabespannung beträgt jedoch 5V. Also müssen noch an jedem Kathodenstrang Widerstände eingebaut werden. Damit diese Erklärung etwas übersichtlicher wird ist in Abb. 5 das Schema einer Querschnittebene des Würfels abgebildet (d.h. ein Querschnitt mit 8 ganzen Kathodensträngen, also die roten aus Abb. 4). Das Schema des ganzen Würfels ist ganz analog dazu einfach mit 7 weiteren Querschnitten angehängt (die rechte Hälfte 7 Mal angehängt, mit geteilten Anoden 1-8). Weiterhin sind Kondensatoren im ganzen Schaltkreis verteilt, um die Spannungen zu glätten und um Signale zu filtrieren.

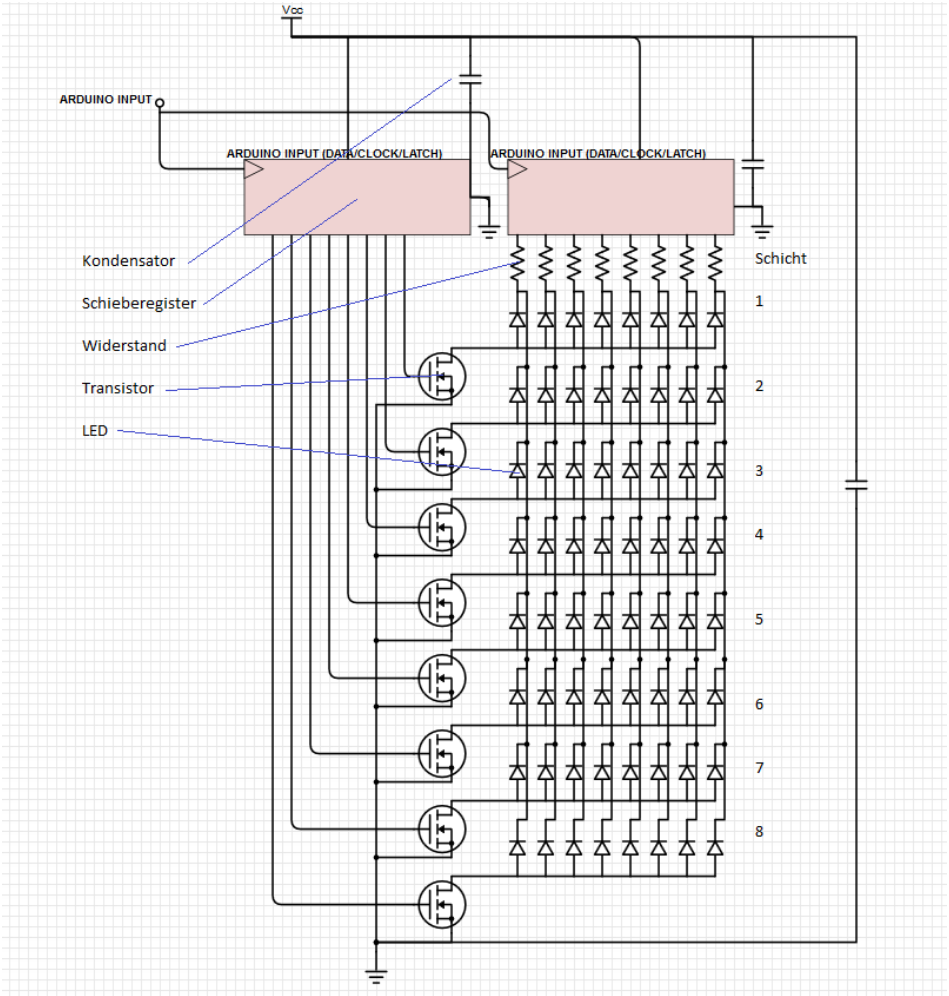
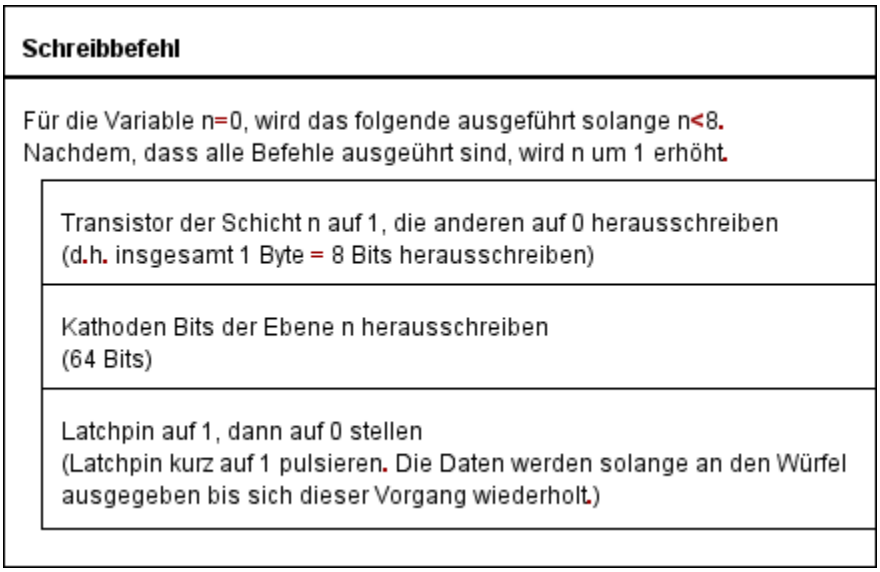


Abb. 5 Schema Querschnittebene des Würfels. (Der richtige Würfel wird mit zusammengeketteten Schieberegistern angesteuert. Wie im Kapitel "Schieberegister und Ansteuerung" erklärt.)

Die Transistoren, welche den Würfel mit 0V verbinden, sind so an den Schieberegistern angeschlossen, dass sie bei einer Ausgabe von 5V einen tiefen Widerstand haben. Der gesamte Würfel ist also so ausgerichtet, dass Strom durch die LEDs fliesst, wenn zwei Bedingungen zutreffen: Erstens müssen die Ausgaben der Kathodenschieberegister auf eine digitale 1 (5V) geschaltet werden. Zweitens muss die Ausgabe des Schieberegisters an die Transistoren ebenfalls 1 betragen, damit der Widerstand des Transistors verschwindet und die LEDs direkt mit 0V verbunden werden.

Das Programm, das den Würfel ansteuert, hat also die Aufgabe, die Daten über drei Leitungen des Arduinos an die Kette von neun Schieberegistern heraus zu schreiben. Die Schieberegister sind in meinem Schaltkreis so angeordnet, dass das Schieberegister, welches die Transistoren steuert, an letzter Stelle ist. Das Programm muss also immer zuerst die Daten für die Transistoren herausschreiben und dann die 64 Kathoden-Bits herausgeben. Zur Erläuterung ist unten ein Struktogramm des Programms angefügt, das den Würfel einmal aufleuchten lässt. Das Struktogramm liest sich von oben nach unten. Zusätzlich werden alle 3 Leitungen auf 0V initialisiert.



Dieses Programm wird immer wieder von neuem genau gleich aufgerufen und kann deshalb in eine sogenannte Subroutine verpackt werden. Das bedeutet, dass man dieses Programm unter einen vereinfachten Namen ausserhalb des Hauptprogramms definiert. Danach kann man diesen Namen aufrufen und die ganze Subroutine wird ausgeführt. Der Schritt zu den schönen Animationen im Würfel ist jetzt nur noch ein kleiner. Die Subroutine, die den Würfel aufleuchten lässt, muss in eine Schleife verpackt werden, die immer zuerst die Daten nach Animation verändert. Mit jedem Durchgang dieser Schleife wird dann schrittweise die Animation durchgearbeitet und am Würfel ausgegeben.

3. Der Bau des Würfels und der Ansteuerungselektronik

3.1 Der erste Prototyp

Bevor dieses Projekt durchgeführt wurde, wurde ein erster Würfel(Abb. 6) im kleinen Stil im Rahmen des Freifachs Informatik gebaut, der anschliessend als Prototyp dienen konnte. Dieser Würfel war die Inspiration für die ganze Maturaarbeit.

Der Prototyp war im Rückblick eine Machbarkeitsstudie, die mir Erfahrung mit den Materialien und dem Löten gab.

Der kleine Würfel ist gleich wie der grosse aufgebaut. Alle Kathoden der LEDs (die Pluspole) sind in 9 individuellen Spalten und die Anoden in drei Ebenen zusammengelötet. Durch diese Anordnung muss der Würfel analog zum grossen Würfel auch durch Multiplexing zum Leuchten gebracht werden. Der Prototyp hat aber im Gegensatz zum finalen Projekt nur zwölf ansteuerbare Pins. Dadurch kann er ohne Zwischenspeicher direkt mit dem Arduino verkabelt werden. Die grösste Schwierigkeit, die sich beim Bau des 3*3*3 Würfels ergab, war das Aneinanderlöten der drei Schichten. Schlussendlich funktionierte der Prototyp aber gut und war trotz Multiplexing genügend Hell.

Der Prototyp zeigte, dass der Aufbau und die Softwarestruktur grundsätzlich funktionierten.

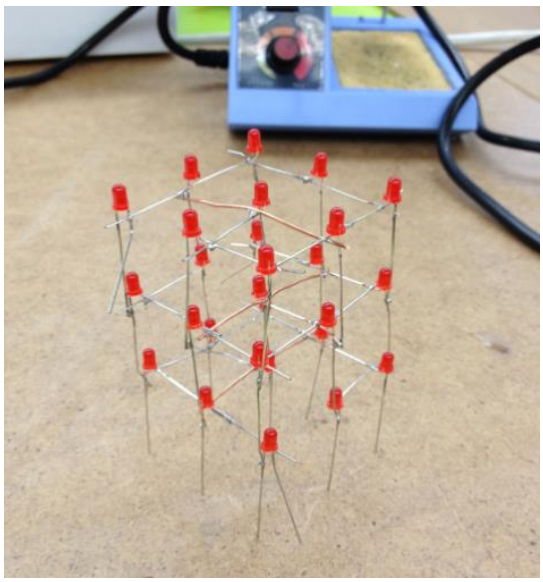


Abb. 6 Erster Würfel

3.2 Schablone

Der erste Schritt für den Bau des finalen Würfels, der die Grösse 8x8x8 haben sollte, war die LED-Grösse zu wählen und danach die Schablone (Abb. 7) herzustellen. In diesem Projekt wurden 3.4V 5mm (typische Grösse) blaue LEDs verwendet. Daraus ergaben sich dann die Grössenverhältnisse für die Schablone, nämlich 19mm Abstand zwischen den Löchern in beide Richtungen.

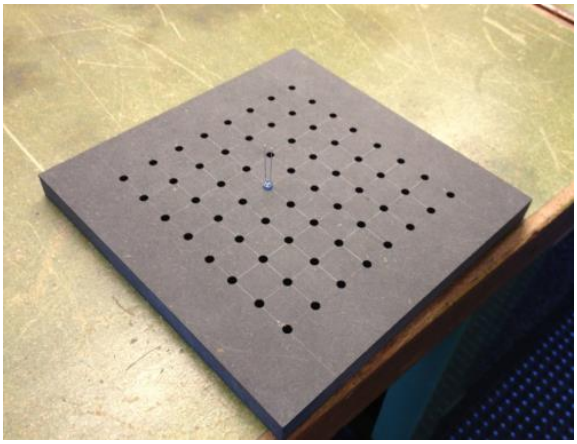


Abb. 7 Holzschablone für den grossen Würfel

3.3 Das Löten des Würfels

Alle LEDs mussten zu einem funktionierenden Würfel zusammen gelötet werden. Dies erfolgte in drei Schritten: In einem ersten Schritt wurde die Funktion jedes LEDs in einem einfachen Schaltkreis überprüft.

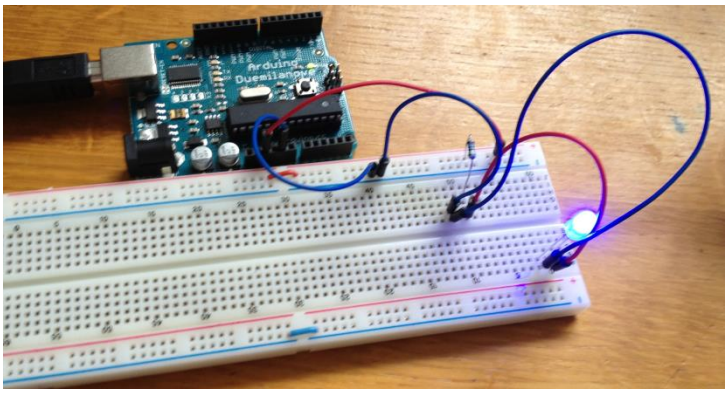


Abb. 8 Test Schaltkreis für LEDs mit Arduino

Durch diesen Schritt konnte verhindert werden, dass nicht funktionierende LEDs in den Würfel eingelötet werden. Danach wurden im zweiten Schritt mittels der gebohrten Schablone alle acht Schichten des Würfels einzeln zusammengelötet. Wichtig dabei war, dass alle Schichten gut verstrebt wurden, da sie sich sonst verbiegen könnten.

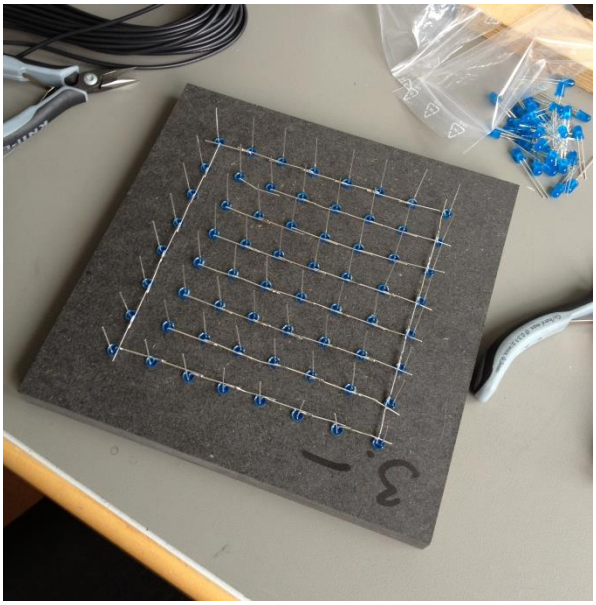


Abb. 9 Bauprozess einer Ebene

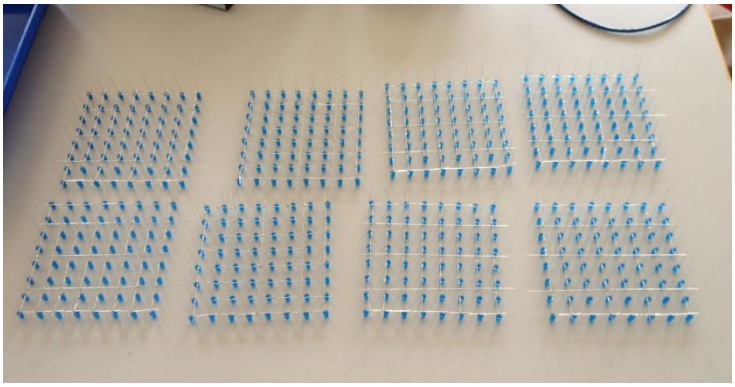


Abb. 10 Fortschritte am Bau, alle 8 Ebenen fertig

Im dritten Schritt wurden alle acht Schichten verbunden. Beim Zusammenlöten der Schichten musste auf verschiedenes geachtet werden. Vorerst musste die Ausrichtung stimmen. Die Schichten (also die Abstehenden Kathoden) mussten so angeordnet werden, dass die inneren, mechanischen Spannungen nicht in einem schrägen Würfel endeten. Im Weiteren mussten die Abstände zwischen den Ebenen überall gleich sein. Um dies zu erreichen, wurden LEGO-Klötze als Abstandhalter zwischen die Ebenen gelegt (es reichte aus, die Klötze in der Mitte hinzulegen, da die einzelnen Schichten relativ stabil waren). Mit der Zeit wurden die Klötze dann durch die Lötstellen entlastet.

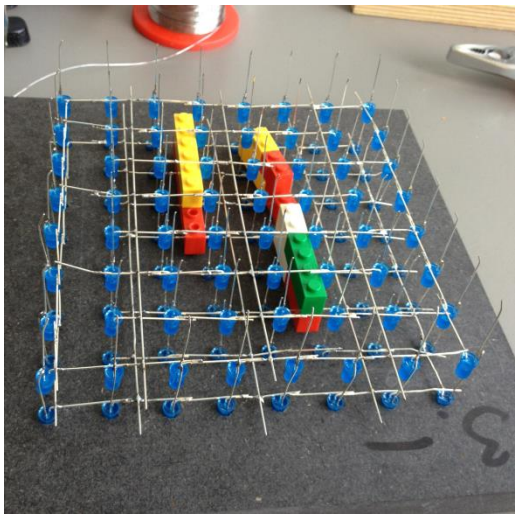


Abb. 11 Zusammenlöten von 2 Ebenen

Wichtig war noch, dass die Funktion jeder angelöteten Ebene sofort überprüft wurde, da das Austauschen von Lämpchen in späteren Phasen des Projekts mühsam gewesen wäre.

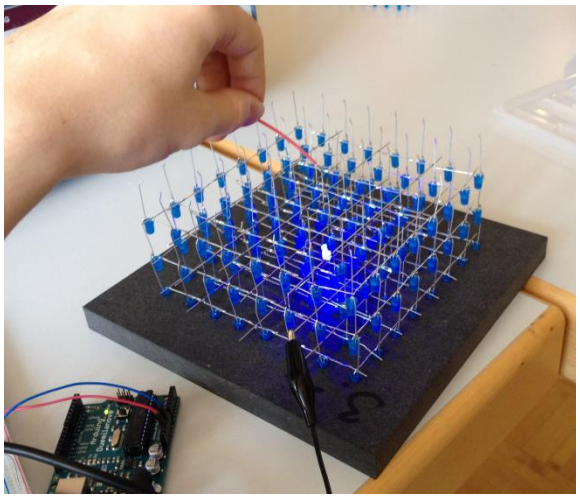


Abb. 12 Direktes Testen am Würfel

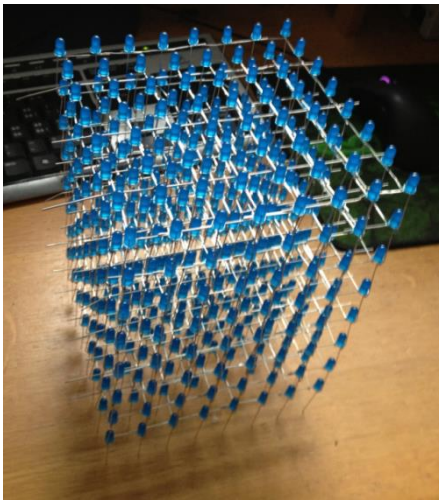


Abb. 13 Fertiger Würfel

3.4 Bau der Ansteuerungselektronik

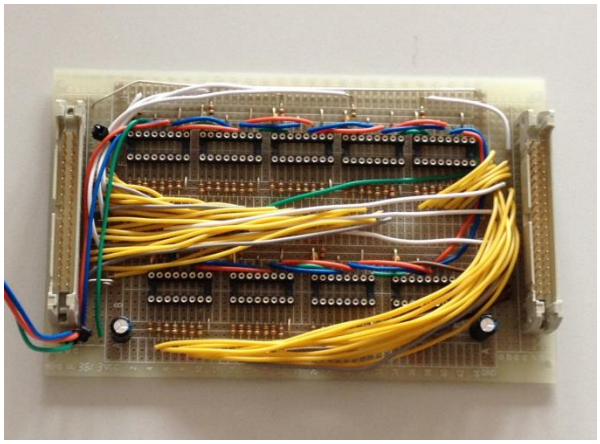


Abb. 14 Fertige Ansteuerungselektronik ohne Schieberegister

Ich entschied mich, aus praktischen Gründen, den Würfel und die Ansteuerungselektronik (Abb. 14) auf separaten Protoboards (kleine Platten zur Herstellung von Schaltkreis-Prototypen) zu löten. Zuerst wurde die Ansteuerungselektronik gebaut. Die verwendeten Schieberegister (74HC595) besitzen 8 Speicherstellen (1 Byte Speicher)und können pro Ausgabe-Pin eine vergleichsweise eher grosse Stromstärke von 70 mA durchlassen. Das ist darauf zurück zu führen das die Ausgaben über CMOS Schaltungen laufen (complementary-symmetry metal–oxide–semiconductor, also komplementäre Symmetrie-Metalloxid-Halbleiter). Normalerweise fliessen in einer logischen Schaltung wie dieser nur kleinere Ströme, weil die

Elektronik so kompakt gebaut ist. Der Vorteil der CMOS Schaltung ist, dass der Strom der Ausgaben nicht direkt über die Logik zur Ausgabe fliesst, sondern zu Transistoren, die den Strom regulieren. CMOS Schaltungen verstärken somit die Ausgabe.
Die CMOS Schaltung (Abb. 15) funktioniert so [9]: Die Ausgabe A wird mit 2 Transistoren verbunden. Der eine hat bei $A = 1$ einen tiefen Widerstand und bei $A = 0$ einen hohen Widerstand, der andere genau umgekehrt. Je nach Ausgabe wird dann Q entweder mit Vdd oder Vss direkt mit der Stromquelle verbunden. Vdd und Vss repräsentieren die Leitungen, die mit der Stromquelle verbunden werden.

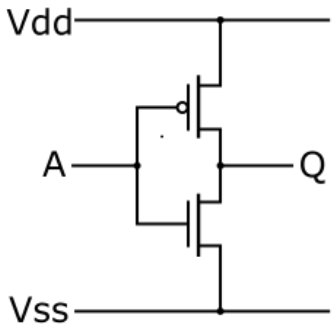


Abb. 15 Schema einer CMOS Schaltung, die im 74HC595 verwendet wird

Weil ich den Würfel aus Platzgründen von den 74HC595s trennen wollte, wurden zuerst die 9 Schieberegister an ein kleines Protoboard gelötet. Die 72 Ausgänge der Chips wurden dann mit zwei 40 Drähte breiten Flachbandkabeln verdrahtet, die mit dem anderen Brett verbunden sind. Bei den Ausgängen der Kathodenschieberegister wurden noch 100 Ohm Widerstände zwischengeschaltet. Das bedeutet bei 20 mA und 5V Ausgabespannung gibt es eine maximale Spannung von 3V über die LEDs. Die drei Eingabedrähte wurden zu einem Inputkabel, das in den Arduino gesteckt werden kann, zusammengefasst. Im Weiteren wurden Kondensatoren verbaut, um die Spannung des Netzgeräts zu glätten. Um Widerstand durch die Kabel zu minimieren, sind die Anoden-Transistoren und die Stromversorgung auf das Protoboard mit dem Würfel ausgelagert. Beim Testen der Ansteuerungselektronik mit einem kleinen Programm traten zuerst noch eigenartige Fehler auf den Ausgabe-Pins auf. Die Ausgaben nahmen nie einen fixen Wert an. Dieser Fehler konnte dann durch einen 100 nF Kondensator zwischen dem Speicherkabel und 0V behoben werden.

Als nächster Schritt wurden die Transistoren, die Stromversorgung und der Würfel an das zweite Protoboard gelötet und verkabelt. Ich verwendete 8 n-Kanal-MOSFETs. Ein n-Kanal-MOSFET wird zwischen einem Verbraucher und 0V geschaltet und hat einen kleinen Innenwiderstand solange die Gate-Source Spannung höher als die Schwellenspannung ist. Im Gegensatz zu anderen Transistoren, in welchen der Innenwiderstand proportional zur Gate-Source Spannung ist, gib es bei den MOSFETs eine Schwellenspannung die, wenn überschritten, den Transistor komplett aufmacht. MOSFETs sind also mit Schaltern vergleichbar.

4. Animationen

Animationen sind das Kernstück des Würfels. Sie sind Algorithmen, die angeben welche Lämpchen leuchten und welche nicht, um schöne oder interessante Muster zu generieren. Die Animationen für den Würfel sind immer nach dem gleichen Muster aufgebaut. Sie manipulieren schrittweise die Kathodendaten und rufen danach die Subroutine auf, die den Würfel aufleuchten lässt (ab jetzt Schreibbefehl genannt). Dieser Prozess wird dann solange wiederholt, bis die Animation fertig ist. Die Kathodendaten sind in einem 512-stelligen Array gespeichert (eine Stelle für jedes LED). Arrays sind Listen von Elementen, die nach dem Index ihrer Position abgeruft werden können. Eine Stelle im Array ruft man ab, indem man den Namen und dann den Index des gewünschten Elements in eckige Klammern hinschreibt (z.B. daten[5]). In diesem Beispiel ist das Array ein Zahlen Array, das Einsen und Nullen enthält. Der Schreibbefehl liest dann pro Ebene nur eines der acht 64-stelligen Zahlensegmente aus dem Array heraus. Die LEDs sind weiterhin, so mit den 8 Kathodenschieberegistern verbunden, dass die LEDs der untersten Ebene, von vorne und von oben gesehen, folgendermassen mit den Indizes des Arrays durchgezählt sind. (Die anderen Ebenen sind analog dazu.)

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Da es aber für die Animationen praktisch ist, die Koordinaten jedes Lämpchens ansteuern zu können, wurde eine Subroutine geschrieben, die aus den drei Komponenten der Koordinaten den Index im Array des Lämpchens errechnet. Die Koordinatenachsen schneiden sich beim Lämpchen mit dem Index 0(also dieses Lämpchen befindet sich im Ursprung).Die x-Achse verläuft vom Ursprung bis zum Index 7, die y-Achse bis zum Index 56 und die z-Achse bis zum Index 448 (sieben LEDs oberhalb vom Lämpchen mit dem Index 0). Dieser Vorgang muss natürlich in einer Animation tausende Male wiederholt werden und wird deshalb in eine Subroutine verpackt. Die Subroutine errechnet den Index so:

$$INDEX = x + y * 8 + z * 64$$

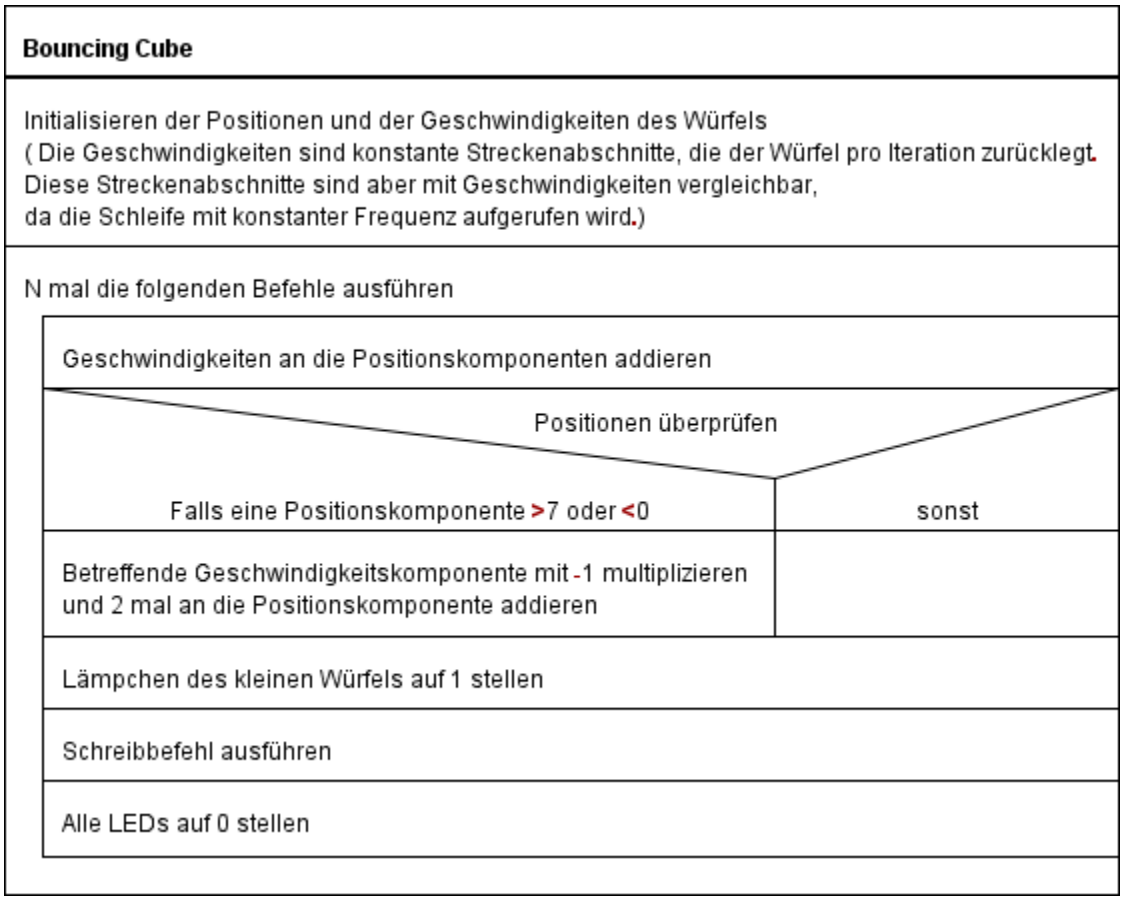
Um jetzt ein oder mehrere Lämpchen aufleuchten zu lassen, muss man die Bits der gewünschten Lämpchen auf 1 stellen und den Schreibbefehl ausführen.

4.1 Bouncing Cube

Bouncing Cube ist einer der einfachen und schönen Beispielanimationen, die ich programmiert habe. Als Einblick in die Animationen werde ich Ihnen Bouncing Cube erläutern

Am Anfang wird im Zentrum des Würfels ein kleiner Würfel generiert und dann in eine Richtung losgeschossen. Der kleine Würfel prallt dann immer an den Wänden ab, bis die Animation vorbei ist.

Das Prinzip der Animation ist eigentlich relativ einfach. Ein Würfel wird mit drei Geschwindigkeitskomponenten in alle drei Dimensionen initialisiert. Danach wird eine Schleife gestartet, die beliebig lange laufen kann. Man nimmt die x, y und z-Koordinaten und zählt zu jeder Komponente die jeweilige Stecke dazu, welche im vergangenen Zeitintervall zurückgelegt wurde (die Stecken sind immer gleich lang, also werden sie im Programm als fixe Werte auftauchen). Danach überprüft man, ob der kleine Würfel sich noch ganz in der Matrix befindet. Das geschieht, indem man die Komponenten der äusseren Punkte des Würfels überprüft. Ist eine Komponente zu klein oder zu gross, so wird die Geschwindigkeit dieser Komponente mal -1 gerechnet. Um einen fliessenden Abpraller zu machen, nimmt man die Komponente, die den Rahmen gesprengt hat und addiert 2 mal die neue Geschwindigkeit dazu, damit der kleine Würfel nicht eine Iteration zu lange am Rand der Matrix bleibt. Zur Erläuterung ist unten ein Struktogramm angefügt.



5. Fazit

Ich habe in Vorbereitung auf das Projekt einen Projektplan erstellt und habe zuvor jeden Arbeitsschritt eingeplant. Während des Projektes habe ich mich grösstenteils gut an diesen Projektplan gehalten und wurde rechtzeitig fertig. Während des Arbeitsprozesses sind unerwartete Ereignisse aufgetreten, sowie z.B. unsaubere Signalausgaben die geflickt werden mussten, oder etwa die Suche nach den optimalen Transistoren, die etwas länger als erwartet andauerte. Schlussendlich funktionierte der Würfel und ich bin gut durch gekommen.

Das Resultat war für mich grundsätzlich ein Erfolg. Der Würfel funktioniert und sieht gut aus.

Die Funktionalität des Würfels ist jedoch in Bezug auf höhere mathematische Operationen eingeschränkt. Das folgt aus der Struktur des Ansteuerungsprogramms. Das Programm berechnet mit jedem Durchgang die neuen Daten und gibt diese dann aus. Während dem Berechnen bleibt die letzte Ebene am Leuchten, da das Latchsingal bis zum nächsten Durchgang nicht ausschlägt. Daraus folgt, dass der Würfel zu flackern beginnt, wenn die Berechnung lange dauert. Dies tritt vor allem dann auf, wenn trigonometrische und Wurzeloperationen durchgeführt werden. Bereits ohne die Verwendung dieser Operationen können interessante und differenzierte Animationen programmiert werden, die nicht flackern. Es gibt aber viele Lösungsansätze zu dem oben genannten Problem, die ich Momentan noch verfolge. Zusammengefasst: Der Würfel funktioniert zuverlässig, aber es können noch Software-Optimierungen am Würfel gemacht werden.

Im Rückblick hätte ich den Hardware Teil der Arbeit gleich durchgeführt. Bei der Software hätte ich mich aber besser mit vertieften Funktionen des Arduinos auseinandergesetzt wie z.B. Interrupt-Routinen und somit das Programm besser geschrieben. Zum schriftlichen Teil der Arbeit: Es ist mir schwerer gefallen als erwartet die Arbeit zu verfassen. Bei einem zweiten Mal würde ich definitiv früher damit beginnen.

6. Quellenverzeichnis

[1] Wikipedia, „Ohmsches Gesetz,“ [Online]. Available: http://de.wikipedia.org/wiki/Ohmsches_Gesetz. [Zugriff am 10 10 2014].

[2] fmgomezcampos, „The PN Junction. How Diodes Work? (English version),“ [Online]. Available: https://www.youtube.com/watch?v=JBtEckh3L9Q. [Zugriff am 10 10 2014].

[3] L. Kneip, „H-Bridges for DC Motors,“ [Online]. Available: http://www.laurentkneip.de/H_bridges.html. [Zugriff am 10 10 2014].

[4] S. Rothe, *PowerPoint Präsentation: Digitalelektronik 2: Vom Transistor zum Byte,* Bern, 9.3.2014.

[5] Wikipedia, „Microcontroller,“ [Online]. Available: http://en.wikipedia.org/wiki/Microcontroller. [Zugriff am 10 10 2014].

[6] Arduino.cc, „Language reference,“ Arduino.cc, [Online]. Available: http://arduino.cc/en/Reference/HomePage. [Zugriff am 10 10 2014].

[7] chr, „LED Cube 8x8x8: Step 6: The anatomy of a LED cube,“ [Online]. Available: http://www.instructables.com/id/Led-Cube-8x8x8/step6/The-anatomy-of-a-LED-cube/. [Zugriff am 10 10 2014].

[8] Wikipedia, „Multiplexing,“ [Online]. Available: http://en.wikipedia.org/wiki/Multiplexing. [Zugriff am 10 10 2014].

[9] Wikipedia, „CMOS,“ [Online]. Available: http://en.wikipedia.org/wiki/CMOS. [Zugriff am 10 10 2014].

Bilder des Würfels wurden von mir, Peter Werner, während dem Bau des Würfels (2014) selber gemacht. Abb 5 wurde mittels der Freeware Schemeit (http://www.digikey.com/schemeit#) gezeichnet. Die Struktogramme (Schreibbefehl und Bouncingcube) wurden mit dem Programm Struktorizer (http://structorizer.fisch.lu/) gezeichnet. Die anderen Abbildungen stammen von den oben genannten Quellen ([1]-[9]).

Anhang

In diesem Kapitel sind Codebeispiele für die in Kapitel 2 & 4.1 genannten Programme. Beachten Sie, dass der Code nur der funktionierende Test-Code für den Würfel ist und noch Optimiert werden kann. Es ging nur darum, dass der Würfel vorerst funktioniert. Die Optimierung des Codes ist noch nicht erfolgt.

Schreibbefehl-Code

Mit diesem Code wird schlussendlich durch den Befehl writeCube(dp,cp,lp) der Würfel einmal aufgeleuchtet und Zeit die Daten im data[]-Array an.

```
int cp,lp,dp,tp; // Clockpin, Latchpin, Datapin und Triggerpin für Oszilloskop
int layerarray[8] = {                    //Transistoren Bytes
 B10000000,B01000000,B00100000,B00010000,
 B00001000,B00000100,B00000010,B00000001};
int data[512];
void setup(){
 cp=12;
 lp=11;
 dp=10;
//Die Verwendeten Pins auf dem Arduino auf Outputmodus stellen
 pinMode(cp,OUTPUT);
 pinMode(dp,OUTPUT);
 pinMode(lp,OUTPUT);
 digitalWrite(lp,LOW);  //sicherstellen,dass die Latchlinie und die Clocklinie auf 0 initialisiert sind, da fehler aufgetreten sind
 digitalWrite(cp,LOW);
}
void loop(){
//Animationen hier laufen lassen
}
// Subroutinen die den Würfel 1 Mal ganz aufleuchten lassen
void shiftOutInt(int datapin,int clockpin,int data ){
 digitalWrite(clockpin,LOW);  // braucht es theoretisch nicht, es sind aber fehler aufgetreten die durch diesen Befehl gelöst worden
 digitalWrite(datapin,Data);
 digitalWrite(clockpin,HIGH);
 digitalWrite(clockpin,LOW);
```

```
}
void writeLayer(int datapin,int clockpin,int latchpin,int layer){
 digitalWrite(latchpin,LOW);
 shiftOut(Datapin,Clockpin,LSBFIRST,layerarray[layer]); //Transistoren Daten-> Q0 auf dem letzten Schieberegister ist die unterste Ebene
 for(int i=0;i<64;i++){
  shiftOutInt(datapin,clockpin,data[i+layer*64]);
 }
 digitalWrite(latchpin,HIGH);
 digitalWrite(latchpin,LOW);
}
void writeCube(int datapin,int clockpin,int latchpin){
 for(int layer=0;Llayer<8;layer++){
  writeLayer(datapin,clockpin,latchpin,layer);
  //delay(1);   Multiplexdelay-> Bestimmt die Zeit die eine Ebene am Leuchten bleibt.
 }
}
```

Bouncing Cube-Code

In diesem Kapitel ist noch der Code der am Schreibbefehl -Code angehängt werden müsste um die Animation „Bouncing Cube“ auszuführen.

```
//setzt alle Werte im Daten Array auf 0
void clearData(){
 for(int i;i<512;i++){
  data[i]=0;
 }
}
//Wandelt x,y,z Koordinaten in den Index des Datenarrays um
int KzI(int x, int y,int z){
 int N=x+y*8+z*64;
 return N;
}
```

```
void bouncingCube(){
 clearData();
 int iterations;
 float vx,vy,vz,x = 3,y=3,z=3;
 //vx=random(-200, 200)/100 + 1;
 vx=1;
 //vy=random(-200, 200)/100 + 1;
 vy=-1;
 //vz=random(-200, 200)/100 + 1;
 vz=-0.5;
 iterations=300;
 data[KzI((int)x,(int)y,(int)z)]=1;
 data[KzI((int)x+1,(int)y,(int)z)]=1;
 data[KzI((int)x+1,(int)y+1,(int)z)]=1;
 data[KzI((int)x,(int)y+1,(int)z)]=1;
 data[KzI((int)x,(int)y,(int)z+1)]=1;
 data[KzI((int)x+1,(int)y,(int)z+1)]=1;
 data[KzI((int)x+1,(int)y+1,(int)z+1)]=1;
 data[KzI((int)x,(int)y+1,(int)z+1)]=1;
 for(int a=0;a<50;a++){
  writeCube(dp,cp,lp);
 }
 for(int i=0;i<iterations;i++){
  for(int a=0;a<3;a++){
   writeCube(dp,cp,lp);
  }
  clearData();
  x=(x+vx);
  y=y+vy;
  z=z+vz;
```

```
if(x>6){
  vx=-vx;
  x=x+2*vx;
}

if(x<0){
  vx=-vx;
  x=x+vx;
}

if(y>6){
  vy=-vy;
  y=y+2*vy;
}

if(y<0){
  vy=-vy;
  y=y+2*vy;
}

if(z>6){
  vz=-vz;
  z=z+2*vz;
}

if(z<0){
  vz=-vz;
  z=z+2*vz;
}
```

```
    data[KzI((int)x,(int)y,(int)z)]=1;
    data[KzI((int)x+1,(int)y,(int)z)]=1;
    data[KzI((int)x+1,(int)y+1,(int)z)]=1;
    data[KzI((int)x,(int)y+1,(int)z)]=1;

    data[KzI((int)x,(int)y,(int)z+1)]=1;
    data[KzI((int)x+1,(int)y,(int)z+1)]=1;
    data[KzI((int)x+1,(int)y+1,(int)z+1)]=1;
    data[KzI((int)x,(int)y+1,(int)z+1)]=1;
  }
}
```

Selbständigkeitserklärung

Hiermit bestätige ich, Peter Werner, dass ich die vorliegende Arbeit selbst geschrieben habe.

Weiterhin versichere ich hiermit, dass ich alle fremden Hilfsmittel deklariert habe und keine anderen Hilfsmittel als angegeben verwendet habe.

Ort, Datum: Unterschrift: