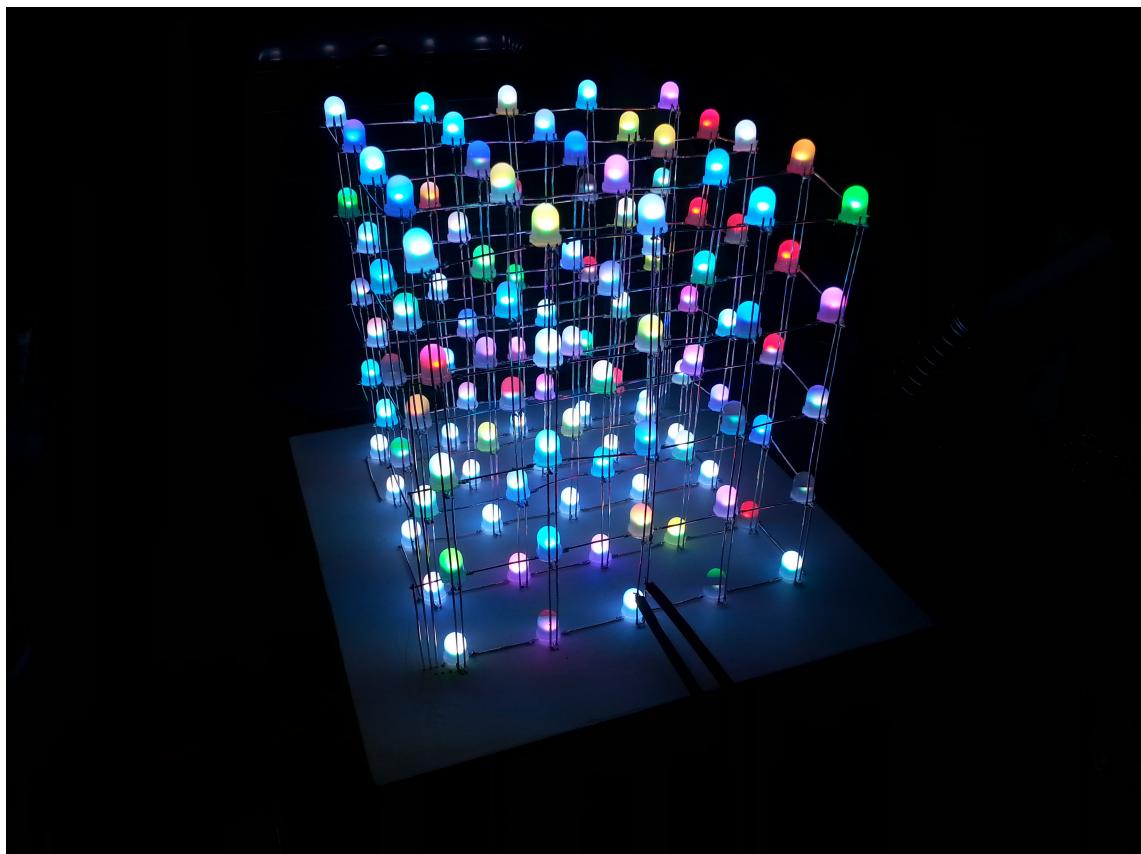


Ansteuerung einer LED-Matrix mit einem Mikrocontroller



Facharbeit im Fach Informatik
am Conrad von Soest Gymnasium
betreut von: Hr. Buyken

vorgelegt von: Jannik Schmöle,
Deiringser Weg 94, 59494 Soest

+49 160 5276634

fertiggestellt am: 20.03.2017

Inhaltsverzeichnis

1 Einleitung.....	3
2 MULTIPLEXING.....	5
2.1 Hintergrund:.....	5
2.2 Funktionsweise:.....	5
2.3 Vorteile:.....	6
2.4 Nachteile:.....	6
3 CHARLIEPLEXING.....	8
3.1 Funktionsweise:.....	8
3.2 Vorteile:.....	8
3.3 Nachteile:.....	8
4 DAISYCHAINING.....	9
4.1 Hintergrund:.....	9
4.2 Funktionsweise:.....	9
4.3 Vorteile:.....	9
4.4 Nachteile:.....	10
5 ANSTEUERUNG MIT DEM DAISYCHAINING.....	11
5.1 Mein LED-Würfel:.....	11
5.2 Pulssweitenmodulation (kurz PWM):.....	11
5.3 PL9823:.....	11
5.4 Probleme des Mikrocontrollers:.....	12
5.5 Timer:.....	12
5.6 Bitbanging:.....	12
5.7 Taktzyklenoptimierung:.....	13
5.8 Assembler:.....	13
5.9 Datenübertragung:.....	14
5.9.1 Tabelle mit den benötigten Werten für die Ansteuerung.....	14
5.10 Zahlensysteme:.....	15
6. Fazit.....	16
ANHANG.....	1
Literaturverzeichnis:.....	1
Materialliste für den LED-Würfel.....	2
Ansteuerungscode:.....	3
Fotos vom LED-Würfel.....	7
Erklärung der selbstständigen Anfertigung.....	8

1 Einleitung

Bildschirme sind aus dem heutigen Alltag gar nicht mehr wegzudenken. Sie werden z.B. im Smartphone, Computer oder Fernseher eingesetzt und haben unseren Alltag verändert. Sie haben meistens eine gleichmäßige Hintergrundbeleuchtung und schalten dann einzelne Pixel an oder aus. Dann gibt es aber auch noch größere Bildschirme, wie sie z.B. als Werbetafel oder auf großen öffentlichen Veranstaltungen eingesetzt werden. Diese wiederum blenden nicht einzelne Bildbereiche ab, sondern die Pixel sind einzelne LEDs. Doch alle haben sie das gleiche Problem. Rechnen wir mal durch, wie viele Drähte man zur Ansteuerung bei einem 720p Bildschirm benötigen würde:

$$1280 \text{ pro Reihe} * 720 \text{ pro Spalte} * 3 \text{ Farben} = 2.764.800 \text{ Anschlüsse}$$

Dies überschreitet die Anzahl der Ausgänge, die ein Mikroprozessor hat bei weitem, ganz abgesehen davon, dass die meisten Anschlusskabel wie HDMI nicht annähernd so viele Pins haben. Also muss man sich mit technischen Raffinessen behelfen. Es gibt grundsätzlich 2 verschiedene Verfahren dieses Problems zu umgehen: das Multiplexing oder das Daisychaining. Ersteres ist für Bildschirme praktikabel, Zweiteres eher für dekorative LED-Streifen oder Hobbyprojekte, weil zum Daisychaining jeder Pixel nochmal einen eigenen kleinen Schaltkreis benötigt, um das Signal zu dekodieren, was in einem Display, wo die Pixel dicht an dicht liegen, eng und teuer wird.

In dieser Facharbeit werde ich mich damit auseinandersetzen, wie die Ansteuerung solcher Displays, sowohl LED-Matrizen als auch LCD-Displays, funktioniert. Dafür werde ich mich mit dem Multiplexverfahren beschäftigen. Außerdem werde ich mich dem Daisychaining widmen, denn ich habe mich dazu entschieden, eine LED-Matrix selber zu bauen. Diese hat die Form eines Würfels und soll weniger als Display eingesetzt werden, sondern mehr als Tischdekor, denn die Auflösung von 5 mal 5 mal 5 Pixeln ist zu gering, um aufwendigere Animationen darzustellen. Die 125 LEDs des Würfels sind alle der Reihe nach über ein Datenkabel miteinander verbunden und lassen sich über

einen Port an einem Mikrocontroller ansteuern. Ich werde mich außerdem damit beschäftigen, wie ein Prozessor „tickt“, denn zur Ansteuerung müssen Zeiten im Mikrosekundenbereich eingehalten werden. Inspiriert zu dem Würfel wurde ich von dem Englisch/Deutschen YouTuber GreatScott¹. Ich wollte diesen erst als Hobbyprojekt bauen, aber im Rahmen der Facharbeit fand ich es sehr interessant, mich mit der Ansteuerung und Funktionsweise mal von wissenschaftlicher und nicht von Nutzerseite zu beschäftigen, denn hinter den heutigen Bildschirmen versteckt sich eine Menge Technik, die der Nutzer sonst nicht zu sehen bekommt.

¹ GreatScott: URL: www.youtube.com/user/greatscottlab

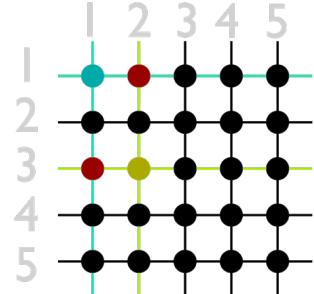
2 MULTIPLEXING

2.1 Hintergrund:²

Das Multiplexing wurde wie, oben bereits genannt, entwickelt, um die Zahl der benötigten Drähte für eine Signalübertragung zu reduzieren. In der Datenübertragung werden mehrere Signale mit verschiedenen Verfahren von einem Multiplexer gebündelt durch ein Kabel übertragen und danach mit einem Demultiplexer wieder in einzelne Signale umgewandelt.³ Dieses Verfahren wird teilweise auch beim Display Multiplexing genutzt (dazu später mehr), das Display Multiplexing bringt aber sonst einen eigenen Multiplex Ansatz mit.

2.2 Funktionsweise:⁴

Das Display ist in Reihen und Spalten unterteilt, die einzelnen Pixel lassen sich dann ähnlich wie Koordinaten ansteuern (zum Beispiel wäre Pixel P(10|20) dann Reihe 10, Spalte 20). Doch daraus resultiert folgendes Problem: Wenn man jetzt z.B. Pixel A(1|1) und B(2|3) ansteuert, leuchten auch Pixel C(2|1) und D(1|3), weil die erste Reihe von Pixel A aktiv ist und die dritte Spalte von Pixel B. Um diesem Problem zu entgehen, wird nacheinander jede Reihe (oder je nach Ansatz jede Spalte) kurz eingeschaltet, somit entfällt oben genanntes Problem. Wenn man diesen Prozess schnell genug hintereinander wiederholt, bei modernen Bildschirmen in der Regel mit 50 Hz oder 60 Hz (50 bzw. 60 Durchläufe pro Sekunde), erkennt das menschliche Auge nicht mehr, dass nur je eine Reihe aktiv ist. Allerdings erkennt das menschliche Auge bei LED-Matrizen einen Helligkeitsunterschied, da jede Reihe nur für 1/n (wobei n = Zahl der Spalten) der Zeit eingeschaltet ist. Grund dafür ist, dass die Erregung der Lichtrezeptoren im Auge noch kurze Zeit abklingt.⁵ Bei herkömmlichen LCD-Displays ist das weniger ein Problem, da



2 Patrick Schnabel: „Multiplex und Multiplexing“ URL: www.elektronik-kompendium.de/sites/kom/0211292.htm [Stand: 18.03.2017]

3 JIMBO: „Serial Communication“ URL: learn.sparkfun.com/tutorials/serial-communication [Stand: 18.03.2017]

4 „LED-Matrix“ URL: www.mikrocontroller.net/articles/LED-Matrix [Stand: 18.03.2017]

5 „Nachbildungswirkung“ URL: de.wikipedia.org/wiki/Nachbildungswirkung [Stand: 18.03.2017]

diese eine konstante Hintergrundbeleuchtung haben und die Pixel nicht selber leuchten.

Allerdings ist dies für die Augen nicht gesund, weshalb man möglichst nicht durchgehend vor dem Bildschirm sitzen sollte, darüber könnte man aber nochmals eine eigene Facharbeit schreiben.

Um noch mehr Pins einzusparen, können sogenannte Schieberegister⁶ angewandt werden. Diese haben mehrere Ausgänge, die alle entweder AN (1) oder AUS (0) sein können. Dazu wird am Dateneingang das nächste Datum angegeben (kein Strom für eine 0, Strom für eine 1) und durch einen kurzen Strompuls am Clock Pin schiebt das Register alle Einträge weiter und nimmt das anliegende Signal am Dateneingang in die erste Position des Schieberegisters. Die letzte Position entfällt oder kann, wenn gewünscht, am Datenausgang abgefangen werden, um ein weiteres Schieberegister anzuschließen (Kaskadieren⁷). Da das Schieberegister die Daten immer zum Nachfolger bei einem Signal weiter gibt, ist dies eine synchrone Daisychain. Auf das Daisy chaining gehe ich aber im entsprechenden Kapitel noch näher ein.

2.3 Vorteile:

Es können mit relativ wenigen Bauteilen eine große Anzahl an Pixeln angesteuert werden. Die Zahl der benötigten Anschlüsse berechnet sich aus Zeilen + Spalten und kann durch Anwendung von Schieberegistern signifikant verringert werden.

2.4 Nachteile:

Effektiv ist immer nur eine Reihe aktiv, was besonders bei LED-Matrizen auffällt, da die Pixel dunkler erscheinen. In gewissem Maße kann man dem entgehen, indem man die LEDs in der Zeit in der sie eingeschaltet sind, mit mehr Strom versorgt, als sie im Dauerbetrieb aushalten würden. Doch da die LEDs danach wieder ausgeschaltet sind, können sie in der Zwischenzeit wieder abkühlen. Bei

6 Havard: „Simple multiplexing with two shift registers“ URL: hblok.net/blog/posts/2013/06/30/simple-multiplexing-with-two-shift-registers/ [Stand: 18.03.2017]

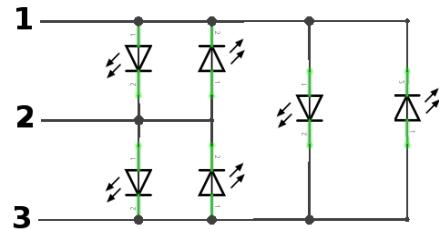
7 „AVR-Tutorial: Schieberegister“ URL: https://www.mikrocontroller.net/articles/AVR-Tutorial:_Schieberegister [Stand: 19.03.2017]

den meisten LEDs ist der Spitzstrom etwa doppelt so hoch, wie die normale Betriebsspannung, was in etwa der doppelten Helligkeit entspricht. Unendlich lässt sich damit also auch nicht ausgleichen, ohne die Lebensdauer der LEDs signifikant zu reduzieren.

3 CHARLIEPLEXING

3.1 Funktionsweise:⁸

Charlieplexing ist in der Grundidee eine Weiterentwicklung des Display Multiplex Ansatzes, funktioniert aber nur für LED-Matrizen, nicht aber für LCD-Displays. Denn man macht sich hier zunutze, dass LEDs Dioden sind und somit Strom nur in eine Richtung leiten, das heißt es werden zwei LEDs gegengleich angeschlossen, und je nach Polung leuchtet dann eine der beiden LEDs oder ohne Strom dann keine der beiden. Wenn man nun mehrere dieser Konstrukte kombiniert, können somit wieder beliebig große Displays gebaut werden.



3.2 Vorteile:

Die Zahl der Ausgänge berechnet sich aus Anzahl² - Anzahl und ist somit höher als bei dem Multiplex Ansatz.

3.3 Nachteile:

Es kann immer nur eine LED gleichzeitig eingeschaltet sein und nicht wie beim Multiplexing ganze Reihen, somit ist die Helligkeit auch noch geringer. Außerdem sucht sich der Strom unter Umständen einen anderen Weg als geplant, da ein ausgeschalteter Pin vom Mikrocontroller Strom durchlässt. Um dies zu verhindern, kann man gerade nicht genutzte Pins im Mikroprozessor auf „Eingang“ schalten, wodurch dieser dann nahezu keinen Strom mehr leitet.

8 Burkhard Kainka: „Charlieplexing“ URL: www.elektronik-labor.de/AVR/Charlieplexing.html [Stand: 18.03.2017]

4 DAISYCHAINING

4.1 Hintergrund:⁹

Beim Daisychaining werden mehrere Komponenten in Reihe geschaltet, wobei die erste Komponente immer mit der Ansteuerung verbunden ist. Es wurde entwickelt, um Anschlüsse zu sparen und wird zum Beispiel in der Bühnentechnik angewandt, um alle Scheinwerfer gesammelt an einem Punkt anzusteuern. Es nutzt ebenfalls verschiedene Übertragungsprotokolle aus dem Multiplex Bereich.

4.2 Funktionsweise:¹⁰

Es gibt mehrere Ansätze eine Daisychain zu realisieren, für mich relevant sind zwei Verfahren: synchron und asynchron. Beide benötigen mindestens eine Datenleitung zur Verständigung. Doch da nur das erste Mitglied in der Kette mit der Ansteuerung verbunden ist, muss es eine Möglichkeit geben, dass die jeweiligen Mitglieder ihr Datenpaket bekommen. Jetzt kommt der Unterschied zum Tragen. Das asynchrone Register schiebt sein Datenpaket immer an das nachfolgende Mitglied weiter, sobald es ein neues Datenpaket erhält. Die Datenpakete haben in der Regel eine konstante Länge, damit die Mitglieder wissen, wann ein neues Datenpaket anfängt. Beim synchronen Ansatz gibt es einen weiteren Anschluss, der sagt, wann jedes Mitglied der Kette sein Datenpaket weiterreichen soll. Die Größe des Datenpaketes kann somit variieren.

4.3 Vorteile:

Die Ansteuerung erfolgt gesammelt an einem Punkt. Es werden nur sehr wenige Anschlüsse benötigt, beim asynchronen Verfahren sogar nur ein einziger. Somit kann bei der Verdrahtung eine Menge Kabel eingespart werden.

9 „Daisy Chain“ URL: de.wikipedia.org/wiki/Daisy_Chain [Stand: 18.03.2017]

10 Görne, Thomas: „Tontechnik“. 4. Auflage, Hanser Verlag, 2014, S. 217f

4.4 Nachteile:

Das Übertragungsprotokoll ist störanfällig, da eine Menge Daten durch ein Kabel mit einem Multiplexer gebündelt übertragen werden. Kleine Umwelteinflüsse oder Wackelkontakte können die Datenpakete fehlerhaft machen und wenn ein Teil in der Kette ausfällt, bekommen die nachfolgenden gar keine Daten mehr. Außerdem benötigt jedes Mitglied in der Kette einen eigenen Schaltkreis, der die eingehenden Signale verwaltet und weitergibt.

5 ANSTEUERUNG MIT DEM DAISYCHAINING

5.1 Mein LED-Würfel:

Der Würfel ist zusammengelötet aus PL9823 LEDs. Diese haben einen eingebauten Schaltkreis (kurz IC für Integrated Circuit). Das heißt jede LED hat vier Pins, einen Dateneingang, einen Datenausgang, Plus-Pol und Minus-Pol. Die LEDs werden dann nach dem Daisychain Verfahren hintereinander geschaltet. Für die Ansteuerung verwende ich einen Mikrocontroller, den NanoESP. Dieser hat einen 16 Mhz Prozessor. Die LEDs sind in Form eines 5x5x5 Würfels angeordnet. Als Stromversorgung dient ein 5 Volt 8 Ampere Netzteil, welches Mikrocontroller und LEDs versorgt.

5.2 Pulsweitenmodulation (kurz PWM):¹¹

In der digitalen Technik findet die sogenannte Pulsweitenmodulation sehr häufig Anwendung. Da ein Mikroprozessor nur AN oder AUS kennt, kann man diesen schlecht nutzen, um analoge Werte auszugeben, wie es z.B. nötig wäre, um eine LED zu dimmen. Also schaltet man den Eingang schnell AN und AUS (englisch HIGH oder LOW). Die Länge der Zeit in der der Ausgang eingeschaltet ist im Verhältnis zu der Zeit, in der der Ausgang ausgeschaltet ist, wird Tastgrad (englisch: Duty Cycle) genannt. Dieser bestimmt zum Beispiel auch die Helligkeit von einer LED, da hier das oben genannte Abklingen des Reizes auf der Netzhaut genutzt wird. Durch Veränderung des Tastgrads können aber auch Daten übertragen werden. Der Empfänger kann dann die Tastgrade der gesendeten Daten auswerten.

5.3 PL9823:¹²

Der IC in den genannten PL9823 LEDs empfängt das Datenpaket bestehend aus 24 Bits und behält es für 50 µs im Speicher. Wenn die LED in dieser Zeit ein zweites Datenpaket erhält, gibt die LED das Datenpaket, was momentan

11 Patrick Schnabel: „Modulation / Modulationsverfahren“ URL: www.elektronik-kompendium.de/sites/kom/0211195.htm [Stand: 18.03.2017]

12 WS2812B Datasheet URL: cdn-shop.adafruit.com/datasheets/WS2812B.pdf [Stand: 18.03.2017]

noch im Speicher ist, weiter an die nächste LED. Sollten die 50 µs überschritten werden, wendet die LED das Datensignal an, die LED wechselt ihre Farbe. Allerdings wird die Helligkeit der LEDs nicht, wie man vielleicht vermuten würde, durch unterschiedliche Spannungen kontrolliert, sondern die Leuchtkristalle der drei Farben werden schnell an und wieder ausgeschaltet (PWM). Das menschliche Auge nimmt dies als Helligkeitsunterschied wahr.

Die PL9823 nutzen das Übertragungsprotokoll der etwas populäreren WS2812.

5.4 Probleme des Mikrocontrollers:

Der Mikrocontroller ist auf 16 MHz getaktet. Dies ist fest vorgegeben durch den Quarzkristall und muss in der Programmierung bedacht werden, denn die Länge der benötigten Datensignale für den IC der LED variiert von 6,4 µs bis 13,6 µs, dies entspricht 6 bis 14 Taktzyklen des Prozessors. (Der Prozessor tickt 16 mal in einer Mikrosekunde). Um diese genauen Zeiten zu erreichen, gibt es zwei verschiedene Verfahren: Das Bitbanging und die Nutzung der integrierten Timer.

5.5 Timer:¹³

Diese Methode setzt voraus, dass der Mikrocontroller über Timer verfügt, was aber bei den meisten der Fall ist. Der Timer wird dann programmiert und sobald er abgelaufen ist, unterbricht der Timer sämtliche anderen Prozesse, und der Mikrocontroller schaltet in der Zeit den Ausgang um, dann fährt der Mikrocontroller mit seiner Arbeit fort. Vorteile dieser Methode sind, dass der Mikrocontroller zwischendurch andere Aufgaben erledigen kann. Allerdings muss darauf geachtet werden, dass nicht ein zweiter Timer oder ein eingehendes Signal an einem der Pins den Sendevorgang unterbricht.

5.6 Bitbanging:

Die Grundidee des Bitbangings ist, den Prozessor während des Sendevorgangs nur für die Ansteuerung zu verwenden, das heißt der Prozessor schaltet den Ausgang und geht dann für eine bestimmte Zeit in den Leerlauf, bis die

13 „WS2812 Ansteuerung“ URL: www.mikrocontroller.net/articles/WS2812_Ansteuerung [Stand: 18.03.2017]

gewünschte Zeit erreicht ist, dann wird der Ausgang wieder geschaltet. Damit verschwendet man theoretisch einige Prozessortakte, in denen etwas anderes gemacht werden könnte. Der Prozessor ist dementsprechend zu 100% ausgelastet, allerdings ist so eine sehr präzise Ansteuerung möglich. Letztendlich wird aber ein Großteil der Zeit zwischen dem Umschalten des Ausgangs sowieso benötigt, um zu entscheiden, ob und wann der Ausgang wieder zwischen AN oder AUS wechseln muss. Außerdem muss hier darauf geachtet werden, dass kein anderer Timer den Sendevorgang unterbricht. Dies ist aber mit zwei Zeilen Programmcode erledigt. Außerdem ist der Code abhängig von der Prozessortaktung. Bei einem 32 MHz Prozessor würde der Code zum Beispiel doppelt so schnell ausgeführt, dies sollte ebenfalls bei der Programmierung bedacht werden.

5.7 Taktzyklenoptimierung:

Aufgrund der geringen Taktzahl des Mikrocontrollers muss sehr darauf geachtet werden, welche Zeile Programmcode in wie viele Maschinenanweisungen übersetzt wird. Zum Beispiel wird eine einfache Rechenoperation in eine Zeile Maschinencode übersetzt. Aber schon für eine Standard FOR-Schleife wird für die Initialisierung der Variable ein Taktzyklus und dann pro Schleifendurchlauf nochmal zwei Taktzyklen verwendet. Einer um zu überprüfen, ob die Abbruchbedingung erfüllt ist, und einer um die Variable um eins zu erhöhen. Möchte man jetzt eine Aufgabe innerhalb von Bruchteilen einer Mikrosekunde erfüllen, wird die Zahl der Maschinenanweisungen, die vom Compiler aus der Hochsprache erzeugt werden, auf einmal sehr relevant. Leider ist nicht klar ersichtlich, wie viele Maschinenanweisungen aus einer Zeile Code compiliert werden, wobei der Compiler aus der Hochsprache im Fall von Arduino C++ einen Zwischenschritt macht, und zwar über Assembler.

5.8 Assembler:¹⁴

Assembler ist eine Programmiersprache, die sehr nah an den Maschinenanweisungen ist. Jeder Prozessortyp hat einen eigenen Befehlssatz, somit ist der Code nahezu gar nicht auf andere Prozessoren portierbar.

14 „Assembly Programming Tutorial“ URL: www.tutorialspoint.com/assembly_programming/ [Stand: 18.03.2017]

5.9 Datenübertragung:¹⁵

Die softwaretechnische Umsetzung ist geklärt, schauen wir uns nun das Übertragungsprotokoll an. Ein Datenpaket ist 24 Bit lang und besteht aus drei Wörtern a 8 Bit. Diese repräsentieren die Farbwerte in der Reihenfolge rot, grün und blau. Es wird eine sogenannte Trägerfrequenz eingesetzt, das heißt das Signal ist innerhalb eines Bits immer einmal HIGH und einmal LOW und durch Variation des Tastgrades wird übermittelt, ob dieses Bit eine 1 oder eine 0 repräsentiert. Wenn die Datenleitung für mehr als 50 µs LOW ist, wird das Datenpaket angewandt. Unten stehen die benötigten Zeiten und die Zahl der Prozessortakte, die ein 16 MHz Prozessor dafür benötigt.

5.9.1 Tabelle mit den benötigten Werten für die Ansteuerung¹⁶

	Mikrosekunden	Prozessortakte
1 Zyklus bei 16Mhz	0,0625µs	1
Bit 0 HIGH Zeit	0,4µs	6
Bit 0 LOW Zeit	0,85µs	14
Bit 1 HIGH Zeit	0,8µs	13
Bit 1 LOW Zeit	0,45µs	7
Bit Gesamtzeit	1,25µs	20
Latch Code	50,0µs	800

15 Levine, Josh: „NeoPixels Revealed: How to (not need to) generate precisely timed signals“ URL: wp.josh.com/2014/05/13/ws2812-neopixels-are-not-so-finicky-once-you-get-to-know-them/ [Stand: 18.03.2017]

16 WS2812B Datasheet URL: cdn-shop.adafruit.com/datasheets/WS2812B.pdf [Stand: 18.03.2017]

5.10 Zahlensysteme:¹⁷

Das System, mit dem wir normalerweise zählen, nennt sich Dezimalsystem. Es gibt noch andere Systeme, zum Beispiel Hexadezimalzahl oder Binär.

Dezimal	42
Hexadezimal	2A
Binär	101010

Für die Facharbeit ist aber hauptsächlich das Binärsystem relevant, da es in der digitalen Technik nur 1 oder 0, AN oder AUS gibt. Das Binärsystem erscheint einem auf den ersten Blick sehr kompliziert, man kann sich dieses System aber sehr schnell vor Augen führen und damit sogar an einer Hand bis 31 Zählen.

Man stelle sich vor die Finger von links nach rechts repräsentieren die Zahlen 1, 2, 4, 8 und 16.

Zahl	Finger 1 (1)	Finger 2 (2)	Finger 3 (4)	Finger 4 (8)	Finger 5 (16)
1	✓				
2		✓			
3	✓	✓			
4			✓		
5	✓		✓		
6		✓	✓		
7	✓	✓	✓		
8				✓	
9	✓			✓	
10		✓		✓	

Das System lässt sich so beliebig weiter führen. Um aus Binärzahlen Dezimalzahlen zu machen, gibt es das Hornerschema. Man setzt in die Gleichung die Zahlen von links nach rechts ein: **Zahl · 2^{Index} + ...**

$$101010 \text{ nach } 42 \quad 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 42$$

¹⁷ CosineKitty: „Bit Math Tutorial“ URL: playground.arduino.cc/Code/BitMath#binary [Stand: 18.03.2017]

6. Fazit

Es gibt mehrere Möglichkeiten Displays anzusteuern. Ausgelegt war die Facharbeit ursprünglich nur für LED-Matrizen. Während der Recherche zur Ansteuerung fand ich aber außerdem heraus, dass herkömmliche LCD-Displays ebenfalls den Multiplex Ansatz nutzen und dass man zwischen dem Multiplexen in der Datenübertragung und der Displaytechnik unterscheiden muss, es aber auch Gemeinsamkeiten zwischen den beiden gibt. Über die Ansteuerung nach dem Daisychain Verfahren war ich im vornherein bestens informiert, da ich die Idee einen LED-Würfel zu bauen, schon länger hatte.

Für die Ansteuerung meiner LED-Matrix ist definitiv das Daisychain Verfahren am besten geeignet gewesen, da hierfür weniger externe Komponenten wie Shift Register, Lötstellen und Drähte erforderlich waren, sondern nur ein Beutel mit LEDs mit eingegossenem IC, einen Mikrocontroller und 12 m Draht. Bei der Programmierung für das Daisychainverfahren habe ich mich für den Bitbanging Ansatz entschieden, da dieser besser portierbar ist und zwischen den Sendezeiten nur wenige Prozessortakte wirklich ungenutzt sind. In anderen Fällen wäre das Verfahren mit Timern aber eventuell effizienter, besonders wenn der Prozessor mehr als 16 MHz hat und somit mehr Prozessortakte verschwendet werden als bei meinem.

Für die Ansteuerung größerer Displays ist dieses Verfahren aber ungeeignet, da es für LED-Matrizen zu teuer wäre jede LED mit einem IC auszustatten und kleinere LCD-Displays hätten den Platz sowieso nicht. Hier greift man dann auf das Multiplex Verfahren zurück.

Somit wäre dann geklärt wofür das Multiplexing genutzt wird, aber wo finden wir das Daisychaining im Alltag? Hauptsächlich in der Computertechnik, wo mehrere Festplatten oder Bildschirme angeschlossen werden sollen oder in der Bühnentechnik, aber auch in einigen LED-Streifen wie sie heutzutage in jedem Baumarkt zu haben sind.

ANHANG

Literaturverzeichnis:

- Internet (Stand aller Quellen: 18.03.2017)
 - www.youtube.com/user/greatscottlab
 - www.elektronik-kompendium.de/sites/kom/0211292.htm
 - www.mikrocontroller.net/articles/LED-Matrix
 - de.wikipedia.org/wiki/Nachbildwirkung
 - hblok.net/blog/posts/2013/06/30/simple-multiplexing-with-two-shift-registers/
 - learn.sparkfun.com/tutorials/serial-communication
 - www.elektronik-labor.de/AVR/Charlieplexing.html
 - de.wikipedia.org/wiki/Daisy_Chain
 - www.elektronik-kompendium.de/sites/kom/0211195.htm
 - cdn-shop.adafruit.com/datasheets/WS2812B.pdf
 - www.mikrocontroller.net/articles/WS2812_Ansteuerung
 - www.tutorialspoint.com/assembly_programming/
 - wp.josh.com/2014/05/13/ws2812-neopixels-are-not-so-finicky-once-you-get-to-know-them/
 - playground.arduino.cc/Code/BitMath#binary
 - elty.pl/upload/download/PL9823_pdf.pdf
- Buch
 - Görne, Thomas: „Tontechnik“. 4. Auflage, Hanser Verlag, 2014, S. 217f

Um herauszufinden welche Befehle in wie viele Maschinenanweisungen übersetzt werden, habe ich den Prozessor messen lassen, wie lange er für die Zeile Code benötigt und dies dann ausgegeben. Alle anderen Informationen dieser Facharbeit sind ansonsten oben genannten Quellen entnommen.

Materialliste für den LED-Würfel

Anz	Beschreibung	Ges.
125	PI9823 LEDs *	~30€
1	Schottky Gleichrichterdiode *	1€
2	6 Meter versilberter Kupferdraht	5€
1	NanoESP	23€
1	5 Volt, 8 Ampere Einbaunetzteil	15€
3	480cm ² 5mm dickes Spanholz	5€
1	Heizkörperlack	12€

* Beim Bauen musste ich feststellen, dass die LEDs, wenn man sie falsch herum anschließt, den Strom leider direkt durch den IC leiten, dieser aber kaum einen Innenwiderstand hat, also kurzgeschlossen ist. Somit fließt so viel Strom durch den IC, dass dieser zerstört wird (bei einem Test habe ich mit 5V und 1.5A als Grenze eine LED falsch gepolt angeschlossen, der IC in der LED wurde für wenige Sekunden zur neuen Lichtquelle, bevor ich den Versuch abgebrochen habe). Um nicht versehentlich 125 LEDs auf einmal zu zerstören, habe ich eine Diode eingelötet, um eine falsche Polung zu vermeiden. Die erste Diode, die ich verwendet habe, war aber leider nicht dafür ausgelegt 6 Ampere durchzulassen, weshalb diese schon nach wenigen Sekunden so heiß wurde, dass man sie nicht mehr anfassen konnte. Später habe ich dann eine entsprechend große Diode eingelötet, womit es dann funktioniert hat.

Ansteuerungscode:

```
#define dmHigh B00000000
#define dmLow B11111111
#define numLeds 125
#define nop __asm__("nop\n\t");
int datapin = 3;

byte leds[numLeds*3];

void setup()
{
    pinMode(datapin, OUTPUT);
}

void send()
{
    for(int j=0; j<numLeds; j++)
    {
        byte buff = leds[j];

        for(int i=0; i<8; i++)
        {
            PORTD = dmHigh;
            if(buff & 1)
            {
                nop nop nop nop nop nop nop nop
                PORTD = dmLow;
            }
            else
            {
                nop nop nop
                PORTD = dmLow;
            }
            if(buff & 1)
            {
                nop nop
            }
            else
            {
                nop nop nop nop nop nop nop
            }
            buff >>= 1;
        }
    }
}
```

Die Masken um anzugeben welche Pins geschaltet werden sollen
Zahl der LEDs
Assembler Zeile für einen Tick warten
Der Datenpin der genutzt werden soll

Das Array speichert die Farben aller LEDs

Der Datenpin wird als Output deklariert

Durch alle LEDs durchgehen

Eine Farbe von einer LED aus dem Array laden

Durch alle Bits (8) der Farbe durchgehen

Datenausgang HIGH schalten
Wenn das aktuelle Bit in der Farbe = 1

Bis zum 14. Tick warten
Datenausgang LOW schalten

Wenn das aktuelle Bit in der Farbe = 0

Bis zum 7. Tick warten
Datenausgang LOW schalten

Wenn das aktuelle Bit in der Farbe = 1

Bis zum 7. Tick warten

Wenn das aktuelle Bit in der Farbe = 0

Bis zum 14. Tick warten

Das Byte um 1 nach rechts verschieben,
dadurch wird das aktuelle Bit auf den linken
Nachbarn gesetzt

Die Idee ist, dass alle Farbwerte in einem Array gespeichert werden. Jede LED nutzt 3 Slots, für jede Farbe einen. Wenn die Funktion send() aufgerufen wird, holt er sich aus dem Array der Reihe nach alle Farben als Byte und geht diese dann Bit für Bit durch. Er prüft immer, ob das rechte Bit eine 1 ist und der Ausgang länger HIGH sein muss oder eine 0 ist und der Ausgang länger LOW sein muss. Dann schiebt er alle Bits der Zahl nach rechts. Dieser Code übernimmt nur das Ansteuern der LEDs, das Array leds sollte vorher mit sinnvollen Zahlwerten gefüllt werden.

$$15 \quad ns \quad 4s \quad ms$$

16.000.000		Arduino Speed
800.000		PL9823 Speed
1.000.000		= 1 Tick pro Milli

$$16 \times 10 \text{ } 4s$$

$$\frac{1}{16} = 0,0625 \quad 1 \text{ Tick alle } 0,0625 \text{ } 4s$$

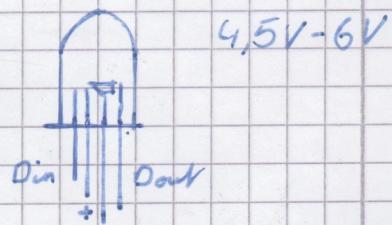
Benötigt: $0,35 \text{ } 4s$
 $1,36 \text{ } 4s$

$$\frac{0,35 \text{ } 4s}{0,0625 \text{ } 4s} = 5,6 \quad 5 \quad // \text{PL9823}$$

$$\frac{1,36 \text{ } 4s}{0,0625 \text{ } 4s} = 21,76 \quad 22$$

$$\frac{171 \text{ } 4s}{0,0625 \text{ } 4s} = 27,36 \quad 27$$

0 Bit: 5T low 22T high
 1 Bit: 22T high 5T low
 Reset: 800T low



8Bit	Rot	Bit 0
8Bit	Grün	
8Bit	Blau	Bit 1

Bit 0

High	$\frac{0,4 \text{ } 4s}{0,0625 \text{ } 4s} = 6,4$	6	// WS2812B
------	--	---	------------

Low	$\frac{0,85 \text{ } 4s}{0,0625 \text{ } 4s} = 13,6$	14
-----	--	----

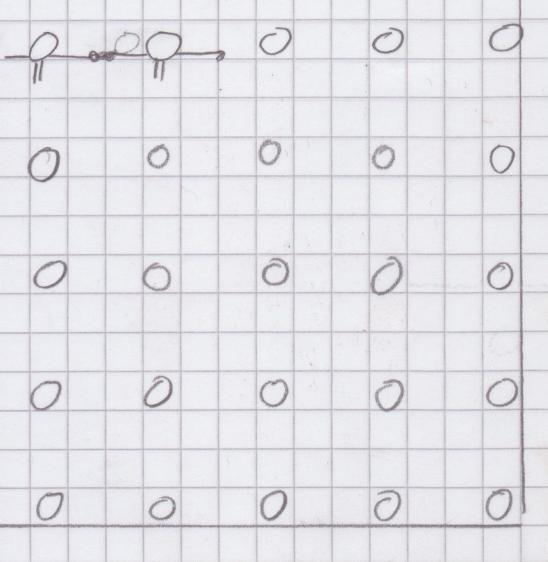
Bit 1

High	$\frac{0,8 \text{ } 4s}{0,0625 \text{ } 4s} = 12,8$	13
------	---	----

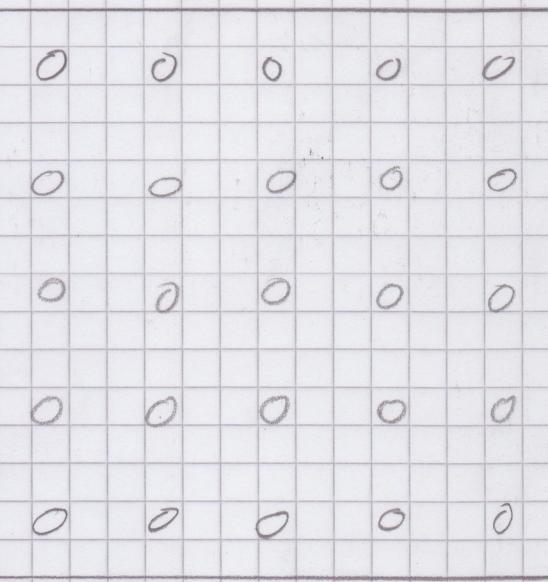
Low	$\frac{0,45 \text{ } 4s}{0,0625 \text{ } 4s} = 7,2$	7
-----	---	---

All	$\frac{1,25 \text{ } 4s}{0,0625 \text{ } 4s} = 20$	20
-----	--	----

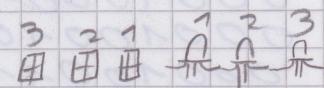
26.01.2017

1.	
	0 0 0 0 0
	0 0 0 0 0
	0 0 0 0 0
	0 0 0 0 0

TOP

2.	
	0 0 0 0 0
	0 0 0 0 0
	0 0 0 0 0
	0 0 0 0 0

25. no Ebene



Data
In

Data
Out

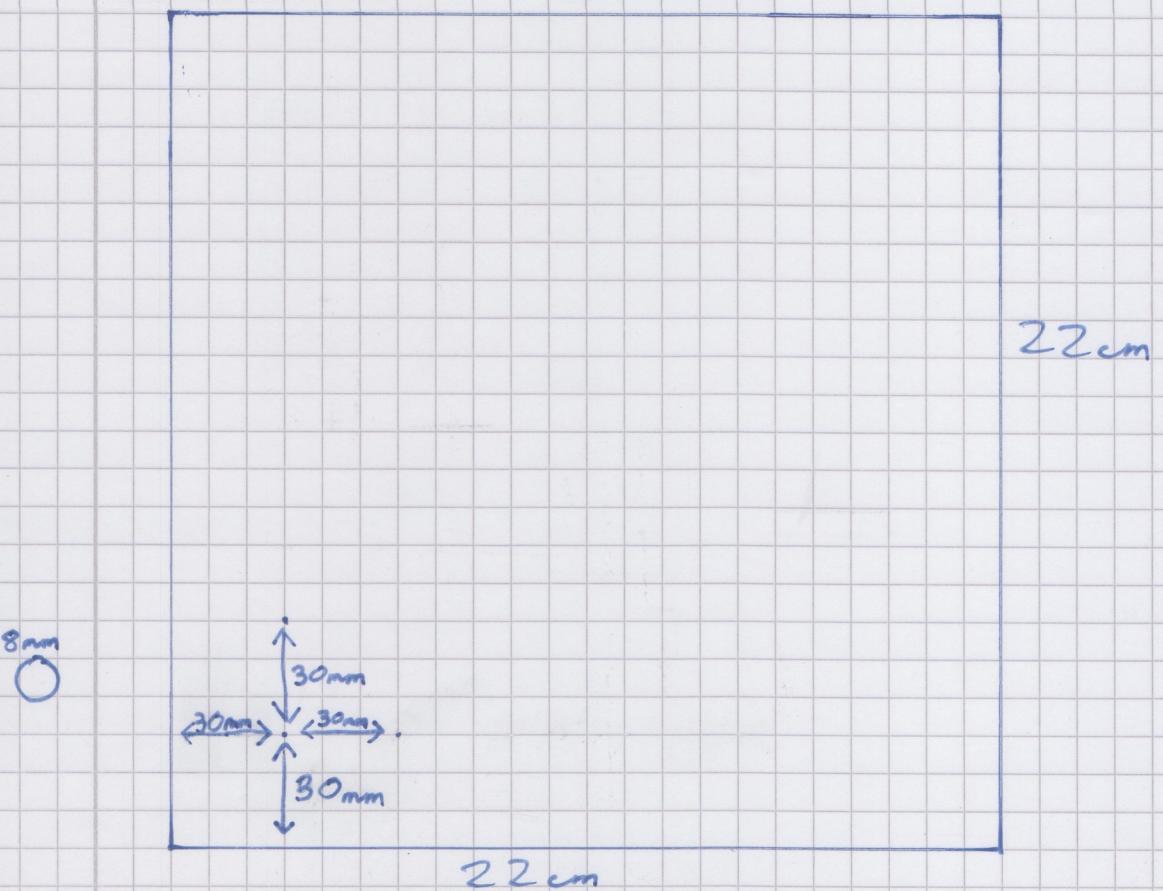
6 R B
8Bit 8Bit 8Bit
0 12 128

f f f f f
f f f f f
f f f f f
f f f f f
f f f f f

FRONT

5 Ebenen

$$5 \times 5 \times 5 = 125 \text{ LEDs}$$

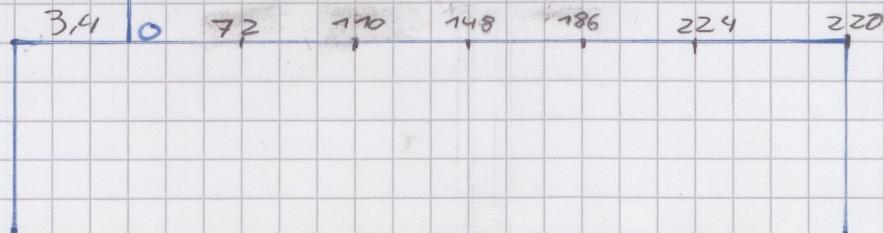


-15,5

-11,7

-7,9

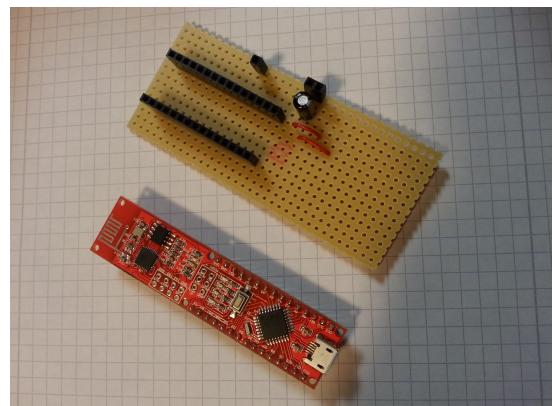
-4,1



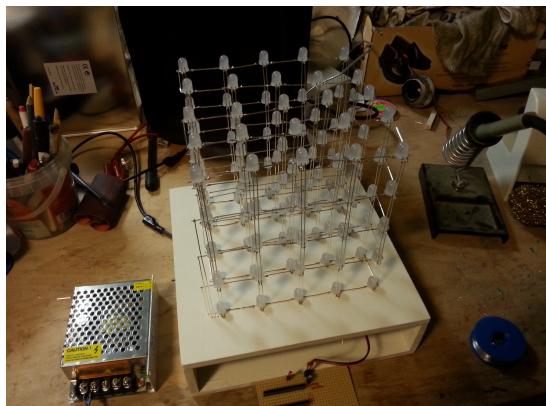
Fotos vom LED-Würfel



Beutel mit 150x PL9823 LEDs



NanoESP neben dem Steckbrett zur Ansteuerung



Der LED-Würfel, daneben das Netzteil



Der LED-Würfel mit zufälligen Farben

Erklärung der selbstständigen Anfertigung

Ich erkläre, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

Soest, den 20.03.2017

Jannik Schnide