

Elektrischer Aufbau einer LED-Matrix auf Punktrasterplatine und programmieren des Spiels „SNAKE“ auf einem Arduino „NANO“ Mikrocontroller.

Projekt angewandte Programmierung

Maik Joshua Wiegand & Max Schwengelbeck

Inhalt

[Motivation 3](#_Toc32073026)

[Elektrischer Aufbau 4](#_Toc32073027)

[Funktion der verwendeten ICs 5](#_Toc32073028)

[Ansteuerung der LEDs 6](#_Toc32073029)

[Das Spiel „Snake“ 7](#_Toc32073030)

[Resümee 8](#_Toc32073031)

[Quellen 9](#_Toc32073032)

# Motivation

Im Rahmen des Bachelor-Studiengangs *Elektrotechnik V6* angebotenen Moduls: *Grundlagen der Programmierung für Ingenieure*, werden die Grundlagen der Programmiersprache C++ vermittelt. Um diese zu vertiefen, wird innerhalb der vorlesungsfreien Zeit ein Projektpraktika angeboten, um die erlernten Kenntnisse zu vertiefen. Hierfür wird exemplarisch der Aufbau und die anschließende Programmierung des Arduino NANO Mikrocontroller gezeigt. So kann auf leichte, spielerische Weise, der Übergang von der Informatischen- zur Ingenieurstechnischen Disziplin geschaffen werden, wobei durch die direkte Steuerung der Schaltung mittels eines C++ Programms, Elektrotechnik und Programmierung nahtlos verknüpft werden. Hierdurch wird zudem die Bedeutung der Programmierfähigkeiten für den Ingenieur im Bereich Entwicklung deutlich.

Beim elektrischen Aufbau der Schaltung werden zudem erste handwerkliche Fähigkeiten vermitteltet, die im Bereich des elektrischen Prototypings verwendet werden können und ansonsten im Studium nur wenig vermittelt werden können.

Das Spiel Snake bietet sich dabei hervorragend für die Programmierung auf einer LED-Matrix an, da es nur wenige Pixel zur Darstellung des Spielfeldes braucht und im Codeding einfach zu realisieren ist. Nichtsdestotrotz müssen erweiterte Techniken zur Ansteuerung der LED-Matrix zum Einsatz kommen da selbst die am großzügigsten ausgestattetsten Mikrocontroller nicht über genügend Ports verfügen, um jede LED einer 8X8 Matrix einzeln mit einem Ausgang anzusteuern, erst recht nicht der hier verwendete Arduino NANO. Um dieses Problem zu umgehen wird hier die Methode des Multiplexings verwendet auf die im weiteren Verlauf auch noch genauer eingegangen wird. Begonnen wird in dieser Ausarbeitung mit der Erläuterung des elektrischen Aufbaus, bevor die Software beschrieben wird.

# Elektrischer Aufbau

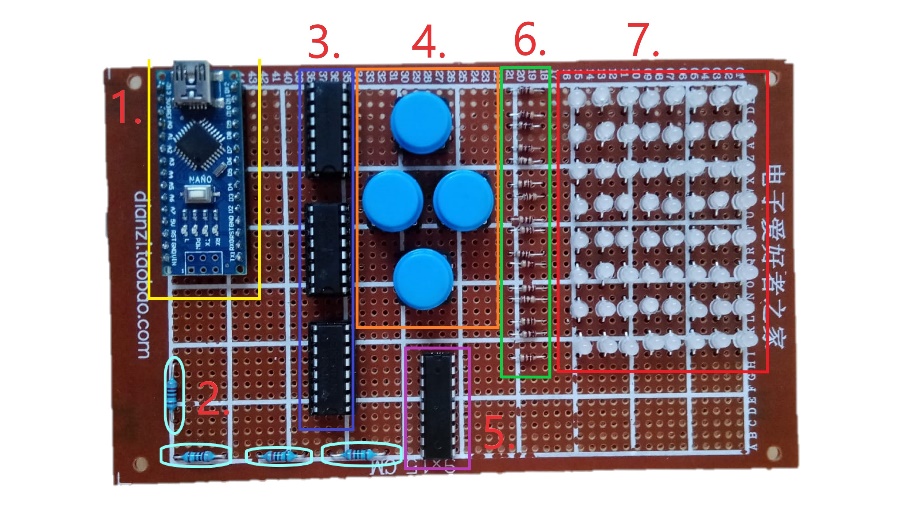
Der Elektrische Aufbau der Schaltung ist einfach zu beschreiben. Verwendet wurde hier eine Punktrasterplatine, es sei aber gesagt, dass eine Streifenrasterplatine einiges an Arbeit beim Löten ersparen kann. Die LED-Matrix besteht aus 8X8 bi-color Leds. Die einzelnen Farben (Rot und Grün) sind jeweils spaltenweise mit einander Verbunden und jeweils mit einem Vorwidertand versehen. Die gemeinsamen Minuspole werden reihenweise miteinander gebrückt.

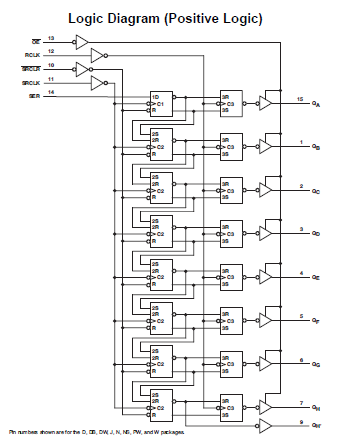
Abbildung : Platine mit Controller. 1. Arduino Nano 2. Pulldown Widerstände der Taster 3. IC 74HC595 4. Taster 5. IC ULN2803APG 6. LED-Vorwiderstände 7. LED-Matrix

Die Ansteuerung der einzelnen Farb-Spalten erfolgt direkt über 8 Ch. Schieberegister ICs vom Typ 74HC595. Die in Reihen geschalteten Minuspole werden auch über ein solches Schieberegister gesteuert, allerdings ist hier eine integrierte 8 Ch. Darlington-Schaltung im IC ULN2803APG zwischengeschaltet.

Die Schieberegister Eingänge *SER, RCLK* und *SRCLK* werden mit den digitalen Eingängen des Arduinos verbunden. Der Pin *SRCLR* wird mit 5V verbunden und der Pin *not OE* mit Ground. Die Ausgänge eines der Schieberegister werden auf die Eingänge der Darlington-Schaltung gelegt, während die Ausgänge der anderen Schieberegister auf die LED-Eingänge ansteuern.

Die Ausgänge der angebrachten Taster werden einfach auf die Eingänge des Arduino gelegt (Hier müssen jedoch Pulldown-Widerstände für die Arduino Eingänge verwendet werden.). Eventuell reicht die Anzahl der digitalen Eingänge des Arduinos nicht komplett aus, in diesem Fall können einfach die analogen Eingänge verwendet werden, welche dann digital eingelesen werden.

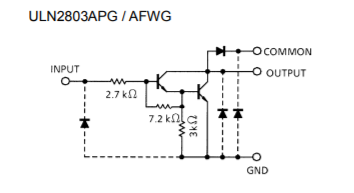
# Funktion der verwendeten ICs

**Funktionsweise des SN74HC595:**

Das verwendete Schieberegister ist die entscheidende Komponente der Schaltung für das steuern LED-Matrix. Es funktioniert folgendermaßen: Das Schieberegister verfügt über einen Speicher von 8 Bit. Bei einer positiven Flanke am RCLK Pin wird der aktuelle Zustand des SER Pins in die erste Speicherzelle des Registers geschoben. Bei einer weiteren positiven Flanke am RCLK Pin wird dieser Wert eine Zelle weitergeschoben, während wieder der Zustand von SER in Zelle eins gespeichert wird.

Ist der Speicher voll belegt, kann der aktuelle Zustand mit einer positiven Flanke an SRCLK an die Ausgänge des ICs übertragen werden. Auf diese Art und Wiese können wir nun alle Spalten der LED-Matrix gleichzeitig ansprechen, in dem der Arduino immer ein komplettes Byte mit der gewünschten Reihen/Zeilen-Kombination in die Schieberegister schiebt.

Abbildung : Logikdiagramm des SN74HC595

**Funktionsweise des ULN2803APG:**

Beim ULN2803APG handelt es sich um eine integrierte Darlington Schaltung. Die Outputs des ICs sind dabei so mit den Collectoren der internen Transistoren verschaltet, dass zwei Transistoren pro Output eine Emitterfolge bilden. Dadurch können die die Transistoren wie ein einziger Schalter verwendet werden, der mit der Collector-Spannung vorgesteuert wird. Alternativ zum IC kann die Schaltung auch händisch auf der Platine aufgebaut werden, was jedoch mit einem weitaus höheren Platzbedarf verbunden ist.

Abbildung : Ersatzschaltbild des ULN2803APG

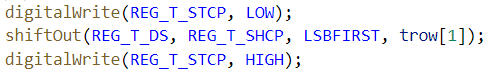
# Ansteuerung der LEDs

Die Ansteuerung der LEDs ist ein Key-Element der Software. Da wir hier die Methode des Multiplexings verwenden, wird pro ausgeführtem Programmloop nur eine LED gezielt angesteuert. Die hohe Taktrate lässt dann jedoch optisch beim Anwender den Eindruck erwecken, dass alle benötigten LEDs gleichzeitig eingeschaltet wären. Somit ist es essentiell das der Programmcode, besonders für das Ansteuern, besonders effektiv und schnell arbeitet, da ein längere Loop-Cycle Zeit eine (scheinbar) verringerte Leuchtintensität der LEDs zur Folge hat.

Die Arduino Standard-Library beinhaltet hierfür allerdings bereits einige nützliche Funktionen. Hierfür muss jedoch zunächst festgelegt werden, welche Bytes überhaupt in die Schieberegister geschoben werden sollen. Da wir jeweils nur eine LED ansteuern wollen und die Bytes im Binärcode in die Register übertragen werden, müssen wir für unserem Fall Potenzen der Zahl Zwei wählen. Ein Angelegter Array für die „Adressen“ der Transistor-Output Reihe sieht dann z.B. so aus:

wobei *trow[1]* z.B. das Byte der ersten Reihe ist.

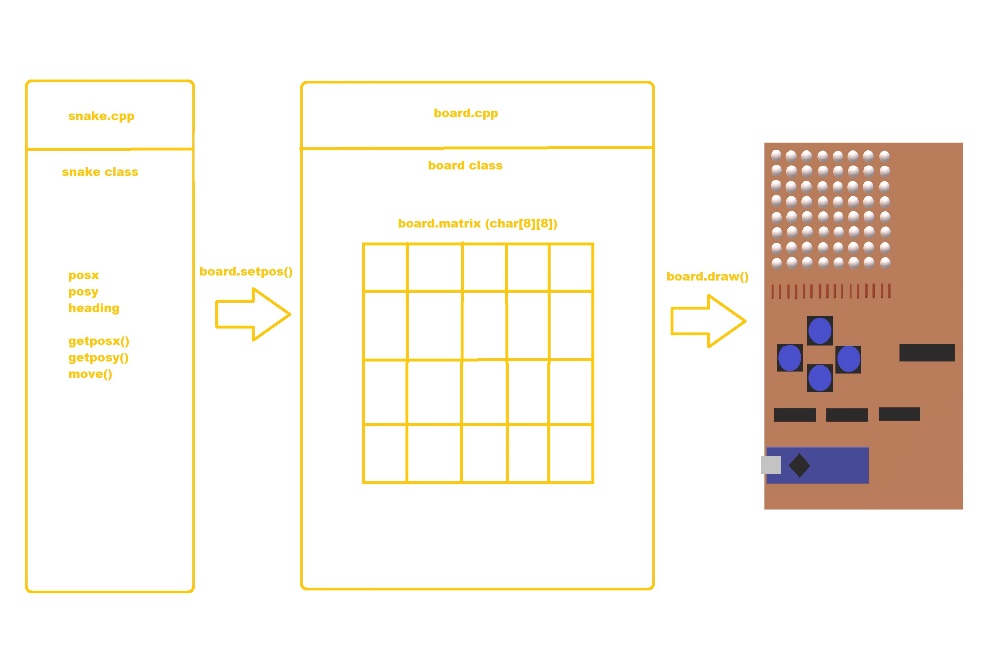
Um dieses Byte an die Schieberegister zu übertragen, hält die Arduino Standard Library den Befehl *shiftOut* bereit. Der Funktion müssen lediglich die Adresse des *RCLK-* und des *SER-*Pins übergeben werden, sowie das zu übertragende Byte. Nun kann die Funktion eine komplettes Byte in den Schiebereisterspeicher übertragen. Die postive Flanke am *SRCLK-*Pin muss allerdings noch händisch gesetzt werden. Ein Aufruf sähe dann Beispielsweise so aus:



Die einzelnen Bits könnten auch über Schleifen einzeln in die Register übertragen werden, diese würden dann jedoch weniger effektiv arbeiten als die *shiftOut* Funktion und somit wertvolle Loop-Cycle Zeit kosten.

# Das Spiel „Snake“

Abbildung : Funktionsprinzip des Programms

Das Spielprogramm „Snake“ besteht aus zwei großen Klassen, die miteinander interagieren. So existiert zunächst einmal die Klasse *Board*, die eine 8X8 Char Array-Matrix verwaltet, welche das Spielfeld darstellt. Diese kann mit dem Set und Get-Methoden beschrieben und ausgelesen werden. Mit Hilfe der *board.print()* Methode kann diese Matrix durch die im vorherigen Abschnitt beschriebenen Funktionen auf die LED-Matrix geschrieben werden.

Die zweite große Klasse *Snake*, erzeugt ein Objekt mit einer schlangenartigen Datenstruktur in deren einzelnen Glieder die X und Y Positionen der Schlangen-Elemente gespeichert sind. Durch die verwendete Datenstruktur können leicht weitere Methoden zum Wachsen (*snake.grow()* ) und Bewegen (*snake.move()* ) der Schlange erstellt werden. Ausgegeben werden die einzelnen Elemente mit Standard Get- und Set-Methoden.

Gesteuert werden die Schlangen-Interaktionen mit dem Board-Objekt durch die Main-Funktion. Sie spawnt Food Elemente auf der Matrix und erzeugt den Spielfeldrand. Hier wird zudem überprüft ob die Schlange etwa mit dem Spielfeldrand oder mit einem Food Element zusammengestoßen ist *(check\_collision()* ). Mit jedem Loop wird hier die Matrix für das Multiplexing neu gezeichnet. Allerdings findet nicht bei jedem Loop eine Bewegung der Schlange statt, um das Reflexvermögen des Spielers nicht zu überanspruchen.

# Resümee

Die erstellte Platine mit der auf ihr vorhandenen LED-Matrix erwies sich im Projektverlauf als durchaus Fähig ein Spiel, z.B. wie das hier gezeigte Snake zu beherbergen. Sobald eine flüssig Funktionierende Methode für das Multiplexing erstellt wurde, sind die einzigen vorhandenen Restriktionen die geringe Pixelzahl und die Kreativität des Programmierers. Durch den eigenen Aufbau werden zudem erste handwerkliche elektrotechnische Fähigkeiten vermittelt. So gelingt einfach der Transfer vom Codeding zur Elektrotechnik.

Nichtsdestotrotz ist das Spielfeld natürlich noch nicht perfekt. So nimmt die scheinbare Leuchtintensität der einzelnen LEDs bei einer Ansteuerung von vielen verschiedenen LEDs stark ab. Es ist möglich dies mit einer höheren LED-Spannung zu kompensieren, dann müssten aber weitere Bauteile verbaut werden und feste Abkühlzeiten für die einzelnen LEDs einprogrammiert werden. Auch muss der Spielprogrammcode für die bestmögliche Darstellung so performant wie möglich seinen, was eine gewisse Hürde für unerfahrene Programmierer seien kann.

Falls Sie das Snake-Programm einmal selbst auf Ihrer selbstgebauten Platine testen möchten, können Sie den Sourcecode auf GITHUB herunterladen:



<https://github.com/wigand-mj/led-matrix>

# Quellen

Abbildung 2: <https://www.alldatasheet.com/datasheet-pdf/pdf/182611/TOSHIBA/ULN2803APG.html>

© TOSHIBA CORPERATION

Abbildung 3: <http://www.ti.com/lit/gpn/sn74hc595>

© TEXAS INSTRUMENTS INCORPERATED