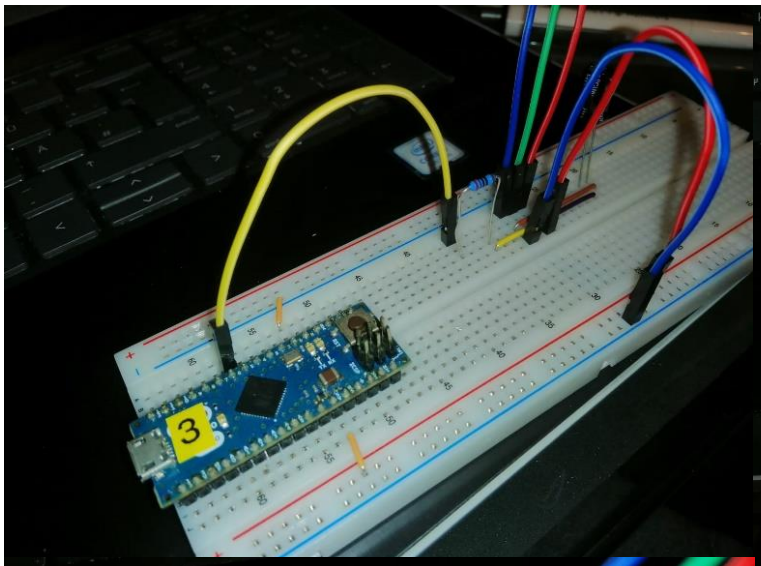
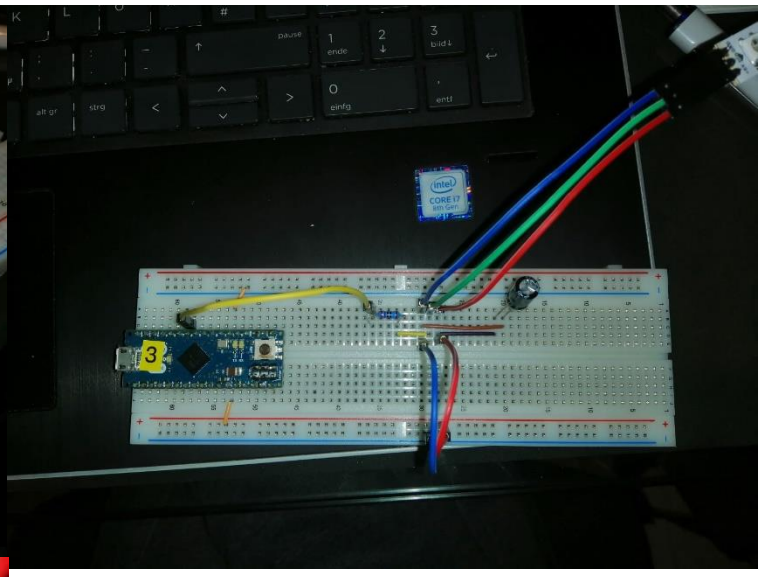
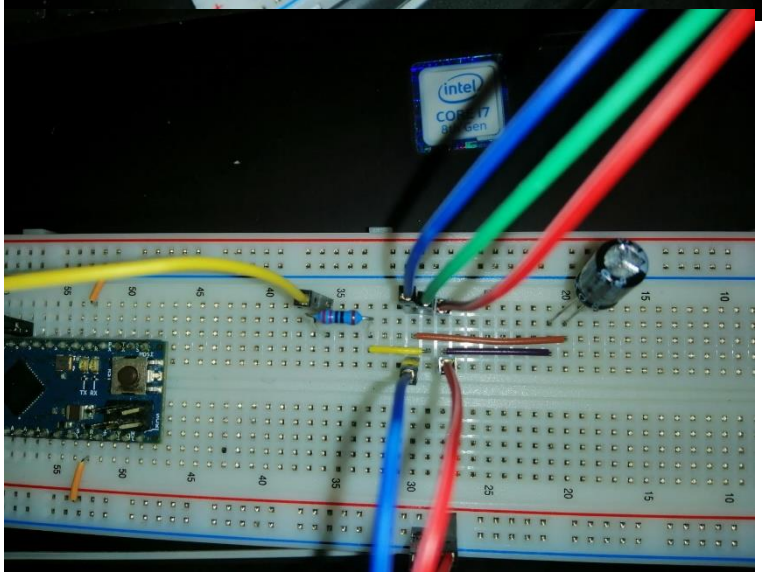
Übungsblatt 5 – Libraries, Serielle Protokolle, komplexe Komponenten

Für Code und Videos siehe Ordner „Code“ und „Anhang“

Aufgabe 1: LED-Strips

1.1

- Es wurde ein 100µF Kondensator verwendet, da sonst nur 4,7µF Kondensatoren oder kleiner vorhanden

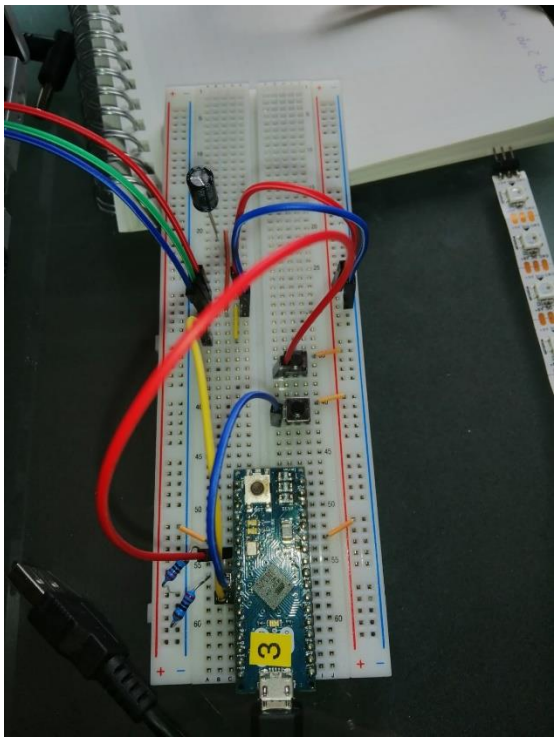




1.2 und 1.3

Siehe Ordner „Code“ und „Anhang“ für das Resultat.

1.4

Um Impulse setzen zu können, habe ich einfach zwei Buttons an die PIN 2 und 4 des Arduinos angeschlossen (mit Pull-down Widerstand). Sobald Button an Pin 2 gedrückt wird, ist ein roter Impuls am LED-Streifen zu sehen, bei Button an Pin 4 ein blauer. Drückt man beide gleichzeitig, so entsteht ein lila-farbiger Impuls. (siehe Video im Anhang)



Aufgabe 2: Binäruhr

Notizen zum SN74HC595N Schieberegister/ Shift register

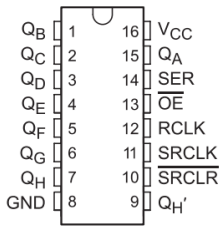
- Datenblatt: https://www.ti.com/lit/ds/scls041i/scls041i.pdf?ts=1616863656747
- Schöne Erklärung: https://www.youtube.com/watch?v=hUZCrba93pU
- kann man sich wie einen kleinen Zwischenspeicher vorstellen. Man schreibt zuerst Bit für Bit in ein Schieberegister (8 bit maximal, bit heißt HIGH oder LOW value), ohne dass diese gleich für die Ausgänge sichtbar sind. Man kann dann alle Bits auf einen Schlag an die Ausgangspins übertragen.
- Nützlich, da man zusätzliche Ein- und Ausgänge zu einem Mikrocontroller hinzufügen kann (ohne alle PINS des Mikrocontrollers zu belegen)
- Pins:

➔ Q_A – Q_H: Ausgangspins

➔ $Q_{H'}$: serieller Ausgang, also quasi das Überlauf-bit; dieses könnte man wiederum an den Datenpin eines weiteren Shift registers weitergeben usw.

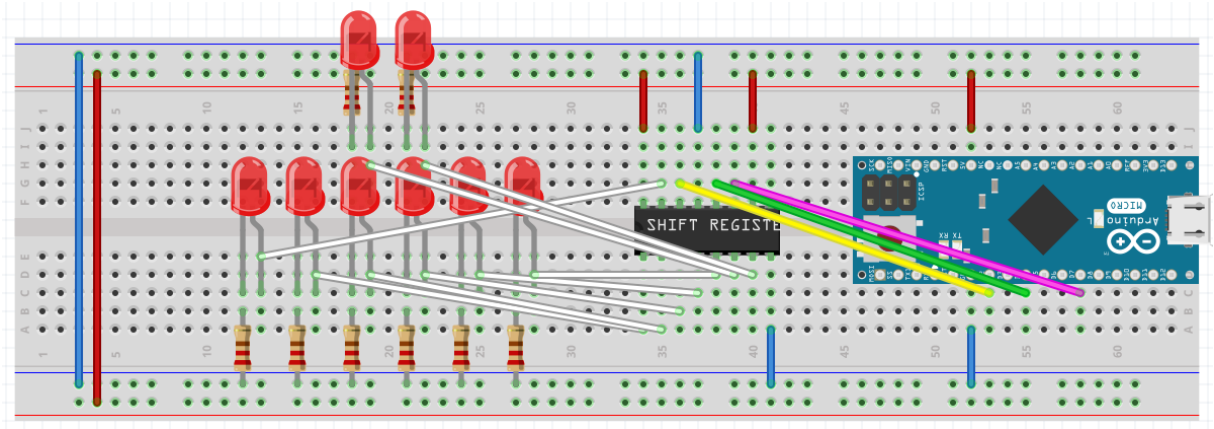
➔ SER: Datenpin/ serieller Eingang, an dem die eigentlichen Daten an das Schieberegister geschickt werden

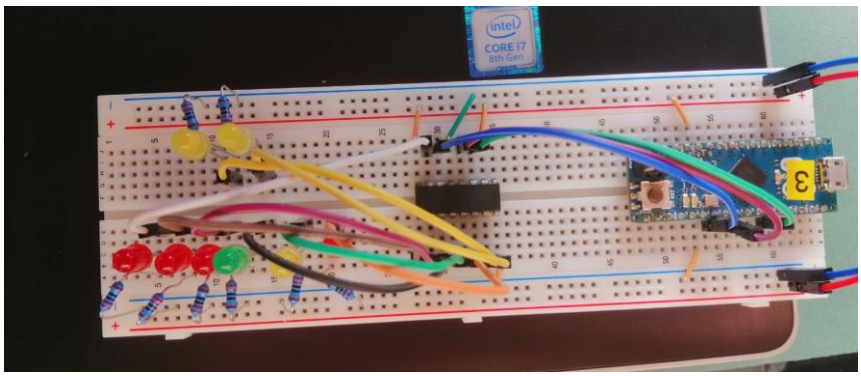
➔ OE: Output Enable; sorgt dafür, das die Ausgänge (QA-QH) auch aktiviert sind → muss mit GND verbunden sein, da LOW active
(der Strich über dem OE bedeutet, dass es LOW aktiv ist)



➔ SRCLK: Shift-clock Pin; Sorgt dafür, dass die Daten vom Datenpin in das Register geladen werden; beim Wechseln von LOW zu HIGH an diesem Pin wird das aktuelle Bit aus dem Datenstrom in das Register geladen.
➔ RCLK: Store-clock Pin; sorgt dafür, dass die 8 bit aus dem Register an die Ausgabepins geschickt werden (beim Wechsel von LOW zu HIGH)
➔ SRCLR: Master Reset (LOW aktiv): Resettet das Bauteil bei LOW → wollen wir nicht unbedingt, deswegen mit Vcc verbinden und nicht GND

- Die drei Pins SER, RCLK und SRCLK werden mit dem Arduino verbunden. SER liefert die Daten, SRCLK lädt diese in das Register, RCLK schickt die Daten des Registers an die Ausgabepins





➔ Weiße Drähte: Verbindungen zwischen Ausgangspins (QA-QH) des Shift Registers und den LEDs
➔ Gelber Draht: serieller Eingang des Shift Registers, verbunden mit Pin 2 des Arduino
➔ Grüner Draht: Verbindung zwischen Store-clock Pin und Pin 4 des Arduino
➔ Rosa Draht: Verbindung zwischen Shift-clock Pin und und Pin 7 des Arduino

Notizen zum Code

- Mit der shiftOut(dataPin, clockPin, bitOrder, value) Methode kann ein byte auf einmal ausgegeben werden (kann aber auch manuell implementiert werden)
➔ clockPin = shift-clock Pin
➔ bitOrder: LSBFIRST oder MSBFIRST
- da durch die delay() Methode der gesamte Code für eine bestimmte Zeit angehalten wird, habe ich mich für einen simplen Timer entschieden, um das Programm nicht zu blockieren

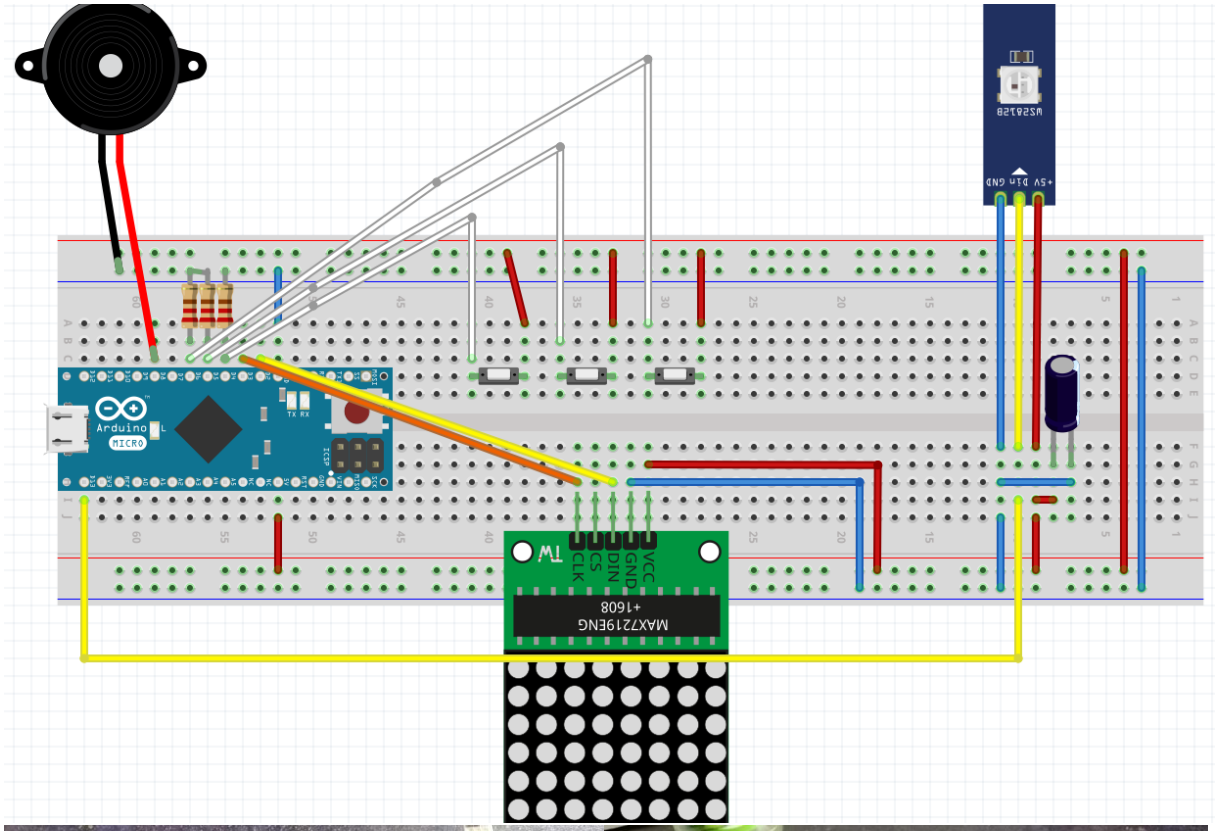
Aufgabe 3: Reaktionsspiel

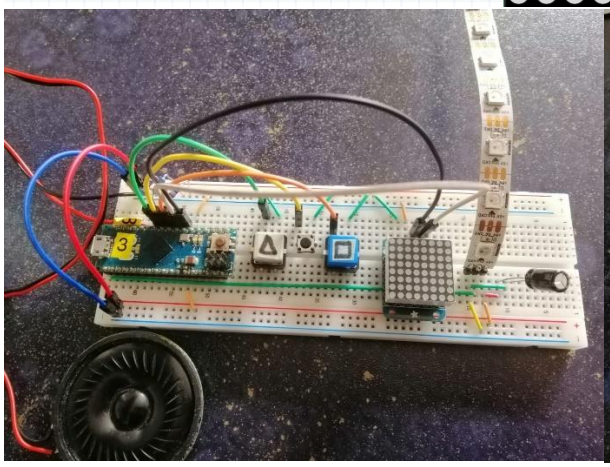
Anmerkungen

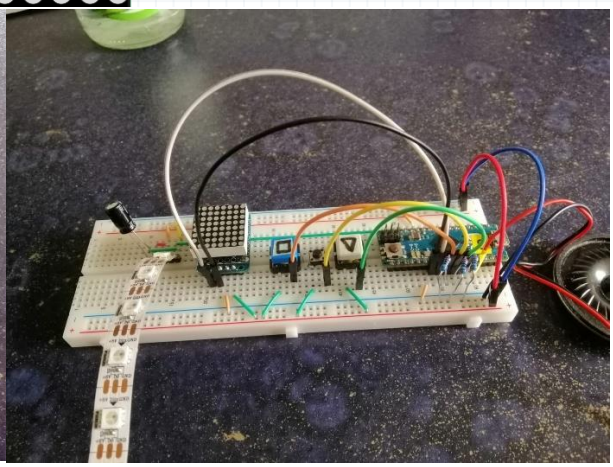
Größere Probleme gab es bei der Bearbeitung der Aufgabe nicht, da wir (bis auf die LED-Matrix) alle Komponenten bereits besprochen/ verwendet haben und man diese in der Aufgabe nur kombinieren musste. Nur bei der LED-Matrix gab es zwei Probleme, zum einen hatte ich zuerst die falsche Adresse (0x70) verwendet, obwohl eigentlich 0x71 richtig wäre (wie man die Adresse herausfindet/ ändert: siehe Links weiter unten). Das andere Problem war ein einfacher Fehler in der Schaltung, der dazu geführt hat, dass die Matrix nicht mit den 5V verbunden war.

➔ LED-Matrix PINs: SDA (Serial data) muss in PIN D2 des Arduino, SCL (Serial clock) muss in PIN D3 des Arduino.

Schaltung







➔ Da ich nur zwei größere Buttons zur Hand hatte, die man auch beschriften konnte, habe ich für die Form „Kreis“ einfach einen kleinen runden Button hergenommen.

Code

Die Aufgabe habe ich umgesetzt, indem ich das Programm in mehrere Status aufgeteilt habe:

- WAITING: Der Startzustand des Spiels, bei dem noch keine Nutzereingabe erwartet wird. Nach zufällig langer Wartezeit (1-10 Sekunden) wird dann eine zufällige Form (Kreis, Dreieck, Quadrat) bestimmt, welche dem Nutzer auf der LED-Matrix angezeigt wird und der Zustand ändert sich zu RUNNING. (Anmerkung: drückt der Nutzer in diesem Zustand schon einen Button, so wird dies als Fehler registriert und zu Status LOSING übergegangen)
- RUNNING: Der Zustand, wenn die Zeit für den Spieler läuft, in der er auf die angezeigte Form reagieren kann und den passenden Button drücken kann. Der Spieler hat dafür insgesamt 1 Sekunde lang Zeit. Drück der Spieler innerhalb dieses Zeitfensters den korrekten Button, so geht das Spiel über zum Status WINNING. Ist der Spieler zu langsam oder drückt den falschen Knopf (auch wenn mehrere Knöpfe gleichzeitig gedrückt werden), so geht das Spiel über zum Status LOSING.
- WINNING: Befindet sich das Spiel in diesem Zustand, wird die aktuelle Punktzahl erhöht, der „Win-Sound“ abgespielt, und danach dann wieder auf WAITING zurückgesetzt. Hat der Spieler den 5. Punkt erzielt, so wird eine kurze Animation beim LED-Streifen angezeigt und dann das Spiel und die Punktzahl komplett zurückgesetzt.
- LOSING: Befindet sich das Spiel in diesem Zustand, wird die aktuelle Punktzahl verworfen, der „Mööp“ Sound abgespielt und das Spiel in den Zustand WAITING zurückgesetzt.

Links

- Verwendete Library für LED-Matrix: https://github.com/adafruit/Adafruit_LED_Backpack
- HT16K33 I2C Adresse herausfinden/ ändern: https://learn.adafruit.com/adafruit-led-backpack/changing-i2c-address
- LED-Matrix Sprite Generator: http://embed.plnkr.co/3VUsekP3jC5xwSIQDVHx
- Code und Melodie für den „WIN-Sound“ wurde übernommen von: https://www.arduino.cc/en/Tutorial/BuiltInExamples/toneMelody