Lernhilfe Dualsystem

von Anna Wanders, Daniela Petkau und Simon Hinz

Konzept/Idee:

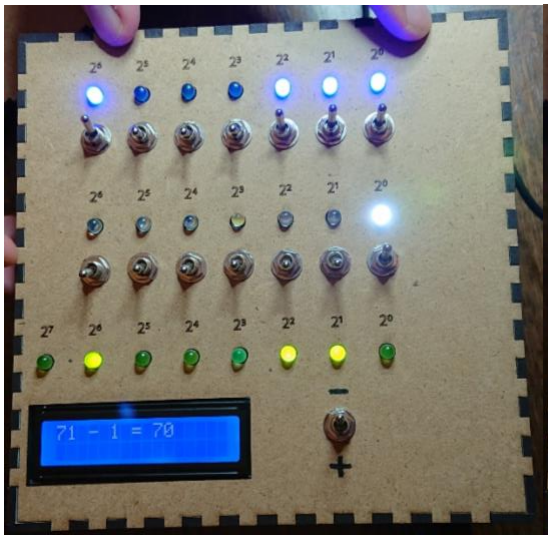
Mit Hilfe unserer Box können Additionen und Subtraktionen von sechsstelligen Dualzahlen durchgeführt werden. Die Eingabe der beiden zu addierenden/subtrahierenden Zahlen erfolgt über Kippschalter. Für eine 1 in der Darstellung wird der Schalter eingeschaltet, eine LED oberhalb des entsprechenden Schalters leuchtet auf. Die Operation kann über einen weiteren Kippschalter eingestellt werden. Das Ergebnis wird in einer weiteren Reihe durch LEDs angezeigt, zusätzlich erscheint die entsprechende Rechnung in dezimaler Darstellung auf einem LCD-Display. Zur Berechnung wird ein Arduino genutzt, der an einem Anschluss an der Seite der Box an den Strom angeschlossen wird.

Unsere Box hilft insbesondere Schülerinnen und Schülern mit Förderschwerpunkt Lernen den Umgang mit dem Dualsystem zu erlernen. Die visuelle Darstellung von Dualzahlen als ein- bzw. ausgeschalteten LEDs nimmt zunächst die Hürde, mit der wesentlich abstrakteren Darstellung mit Nullen und Einsen zu arbeiten. Außerdem werden durch die senkreckt untereinander angebrachten LEDs die Rechenregeln (1+0=1 bzw. „an+aus=an“, 0+0=0 bzw. „aus+aus=aus“ und 1+1=0 bzw. „an+an=aus“) ganz offensichtlich. Durch die Eingabe über die Kippschalter erhält die Bearbeitung einer Aufgabe eine inaktive Komponente und haptische Lerner werden angesprochen.

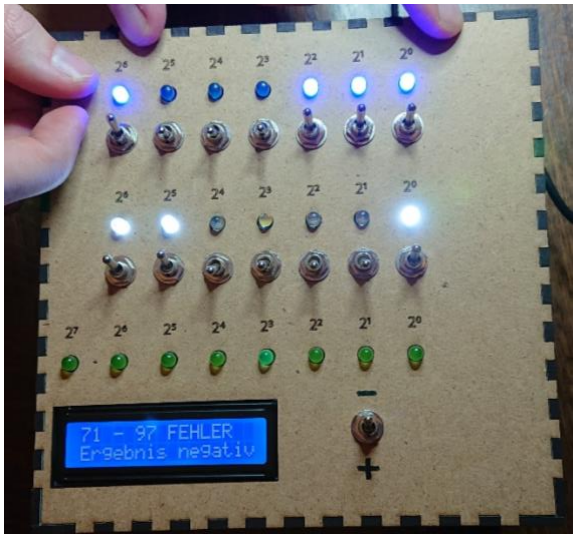
Das Display bietet die Möglichkeit die Eingabe noch einmal zu überprüfen. So werden den Schülerinnen und Schülern mögliche Fehler direkt offensichtlich. Über dieses direkte Feedback können sich auch evtl. Fehlvorstellungen nicht festsetzen, da sie direkt als falsch erkannt werden können. Durch die Gestaltung (Schalter, LEDs, Display) bekommt der Umgang mit der Box einen gewissen spielerischen Charakter der die Schülerinnen und Schüler motivieren kann und zum Ausprobieren anregt.
Als Hilfsmittel „nebenbei“ eingesetzt, bietet die Box unsicheren Schülerinnen und Schülern eine gewisse Sicherheit, wenn sie ihre Ergebnisse anschließend mit der Box überprüfen können.


71 + 65 = 136

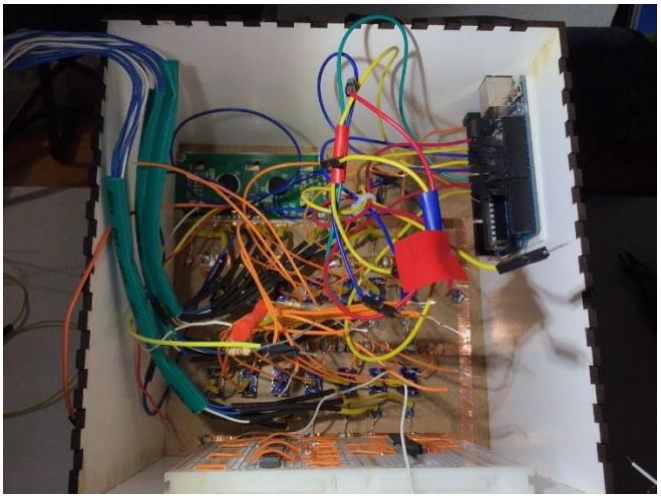
71 + 1 = 72


71 - 1 = 70


71 - 65 = 6


71 - 97 FEHLER
Ergebnis negativ

Konstruktionsanleitung



Benötigtes Material: 1 Arduino Uno, 7 blaue, 7 weiße und 8 grüne LEDs, 22 330Ω Widerstände, 15 3-polige Mini-Kippschalter, 1 1kΩ Widerstand, 1 4kΩ Widerstand, 1 220Ω Widerstand, 3 Schieberegister 74HC595, 2 Multiplexer HCF4051 und 1 LCD-Display Modul LCD1602, mind. 2 Steckbretter, Steckkabel, Kabel, eine (Holz-)box mit den Maaßen 16x16x12cm. Außerdem: Lötmaterial, Schrumpfschläuche, Kupferklebeband, Heißkleber.

Am besten geht man in der folgenden Reihenfolge vor. Die genauen Schaltpläne folgen weiter unten. Geschätzter Arbeitsaufwand: je nach Erfahrung bis zu 20h

1. Hülle entsprechend der Vorlage Lasern oder selber basteln
2. Code auf den Arduino überspielen.
3. Kippschalter einschrauben und Kabel anlöten, Schrumpfschläuche an dem Anschluss, der zum Arduino geht, anbringen.
4. LEDs mit Heißklebe einkleben.
5. Kupferband zwischen den Reihen in den Deckel einkleben. Alle Anschlüsse der Schalter und LEDs, die an die Erde gehen mit dem Band verlöten.
6. Kabel an die LEDs löten (ca. 25cm) können nach Farbe gebündelt werden zur besseren Übersicht, Beschriftung hilft später beim Stecken.
7. Widerstände und Kabel an den entsprechenden Pins des LCD-Displays anlöten. (Der Rest kann nachher mit Steckkabeln verbunden werden.
8. Multiplexer und Schieberegler auf den Steckbrettern anbringen, Verbindungen unter den Komponenten, zur Erde und 5V auf dem Brett (möglichst platzsparend) anbringen.
9. Seitenwände der Box am Deckel anbringen.
10. Arduino, LCD-Display und die Steckbretter an die Außenwände ankleben.
11. Übrige Verbindungen entsprechend des Schaltplans stecken.

Aufbau der Holzbox/Hülle

Im Deckel werden an folgenden Koordinaten (ausgehend davon, dass die obere linke Ecke (0/0) ist) Löcher mit einem Durchmesser von 5mm für die LEDs benötigt:
(2,4/2), (4/2), (5,6/2), (7,2/2), (8,8/2), (10,4/2), (12/2), (2,4/6,5), (4/6,5), (5,6/6,5), (7,2/6,5), (8,8/6,5), (10,4/6,5), (12/6,5), (0,3/11), (2,4/11), (4/11), (5,6/11), (7,2/11), (8,8/11), (10,4/11), (12/11).

Zusätzlich werden an folgenden Koordinaten Löcher mit einem Durchmesser von 6mm für die Schalter benötigt:
(2,4/3,5), (4/3,5), (5,6/3,5), (7,2/3,5), (8,8/3,5), (10,4/3,5), (12/3,5), (2,4/8), (4/8), (5,6/8), (7,2/8), (8,8/8), (10,4/8), (12/8), (10,4/13,7).

Außerdem bräuchten wir folgende Bezeichnungen an folgenden Koordinaten:

2^0: (12/1) (12/5,5), (12/10)
2^1: (10,4/1) (10,4/5,5), (10,4/10)
2^2: (8,8/1) (8,8/5,5), (8,8/10)
2^3: (7,2/1) (7,2/5,5), (7,2/10)
2^4: (5,6/1) (5,6/5,5), (5,6/10)
2^5: (4/1) (4/5,5), (4/10)
2^6: (2,4/1) (2,4/5,5), (2,4/10)
2^7: (0,8/10)
-: (10,4/12,5)
+: (10,4/15)
Erste Zahl: (16,5/2)
Zweite Zahl: (16,5/6,5)
Ergebnis: (16,5/11)

Dazu kommt noch ein Rechteck für das Display, das ausgeschnitten werden soll, mit folgenden Koordinaten:
(3/12,5), (10,2/12,5), (3/15), (10,2/15)

Im vorderen Seitenteil wird ein ausgeschnittenes Rechteck für den USB-Anschluss mit folgenden Koordinaten (ausgehend davon, dass die obere linke Ecke (0/0) ist) benötigt:

(0,8/7,1), (1,9/7,1), (0,8/8,4), (1,9/8,4)

Arduino: Bauteile, Schaltpläne und Quellcode

Erläuterung der einzelnen Bauteile und Schaltungen:

1) **3 Schieberegister 74HC595:**
   Es werden insgesamt drei Schieberegister benötigt, um alle LEDs ein- und auszuschalten. Ein Schieberegister kann 8 LEDs ansteuern. Wir benötigen für die beiden Zahlen, die über die Schalter eingestellt werden können, 7 LEDs und für die Darstellung des Ergebnisses 8 LEDs. Für die Erläuterung der Schieberegister wurden folgende Quellen verwendet:
   https://www.onsemi.com/pub/Collateral/MC74HC595-D.PDF (das Datenblatt)
   https://www.arduino.cc/en/tutorial/ShiftOut
   Für das Schieberegister wird eine vorhandene Arduino-Library verwendet:
   ShiftRegister74HC595.h (https://timodenk.com/blog/shift-register-arduino-library/)

   Die Anschlüsse:

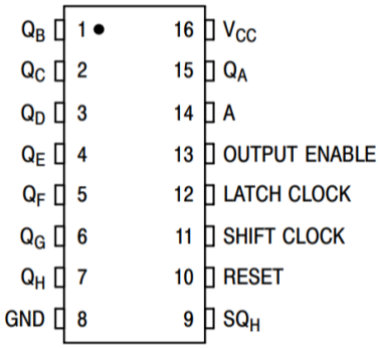


Abbildung 1: Die Anschlüsse des Schieberegister 74HC595 entnommen aus: https://www.onsemi.com/pub/Collateral/MC74HC595-D.PDF

V_{CC}: 5V (+)
GND: Ground (-)
Q_A - Q_H: Ausgänge, um LEDs ein- und auszuschalten. Dabei liegt die 2^0-Stelle an Q_A und die 2^6 Stelle an Q_H an.
A: Serial Data Input. Hierüber wird das Schieberegister vom Arduino angesprochen und mit Daten versorgt, also in unserem Fall mit einer 8-stellige binären Zahl (8Bit), bei der jede 1 bedeutet, dass die entsprechende LED angeschaltet werden soll. Um zu verstehen wie das funktioniert, ist das Logikdiagramm (siehe Abbildung 2) hilfreich. Über Serial Data Input A (oben links) wird entsprechend der Taktrate, die durch Shift Clock vorgegeben wird (unten links), eine binäre Stelle nach der anderen in das Schieberegister geschrieben. Dabei wird in der Taktrate die binäre Zahl von dem ersten Flip-Flop SR_A bis zum letzten Flip-Flop SR_H durchgegeben. Nach 8 Schritten befindet sich dann die 2^0-Stelle der binären Zahl in SR_A und die 2^7-Stelle in SR_H. Mit einer weiteren Taktrate, der Latch Clock (oben links), werden anschließend alle Werte synchron weitergegeben (Parallel Data Outputs rechts) und damit alle LEDs gleichzeitig ein- oder ausgeschaltet. Dies alles geschieht so schnell, dass es für das menschliche Auge nicht sichtbar ist (Die Shift Clock Rate wird bei 4,5 Volt mit maximal 30 MHz angegeben).

Output Enable: Liegt hier 5V an, wäre das Schieberegister ausgeschaltet (active low). Demnach wird es auf GND gesetzt.

Latch Clock und Shift Clock: geben die Taktraten vor, siehe Erläuterung zu Anschluss A.

Reset: Da wir keine Resetfunktion benötigen und dieser wie Output Enable nach dem active low Prinzip funktioniert, wird diese Funktion ausgeschaltet, indem es auf 5V gesetzt wird.

SQ_H: Dieser Anschluss wird benötigt, um mehrere Schieberegister hintereinander zu schalten. In dem Logikdiagramm kann man unten rechts sehen, dass über diesen Anschluss entsprechend der Taktrate die Daten an die angeschlossenen Schieberegister weitergeleitet werden. Dies werden wir nutzen, um wiederum Anschlüsse zu sparen und alle drei LED-Reihen (Zahl 1, Zahl 2, Ergebnis) mit drei hintereinander geschalteten Schieberegistern zu steuern. Es werden dann alle drei Zahlen direkt hintereinander in das erste Schieberegister gegeben. Nach 3*8 Taktraten der Shift Clock sind die drei Zahlen in die korrekten Flip-Flops der drei Schieberegister geschrieben und werden dann mit der Latch Clock synchron an alle LEDs weitergegeben.

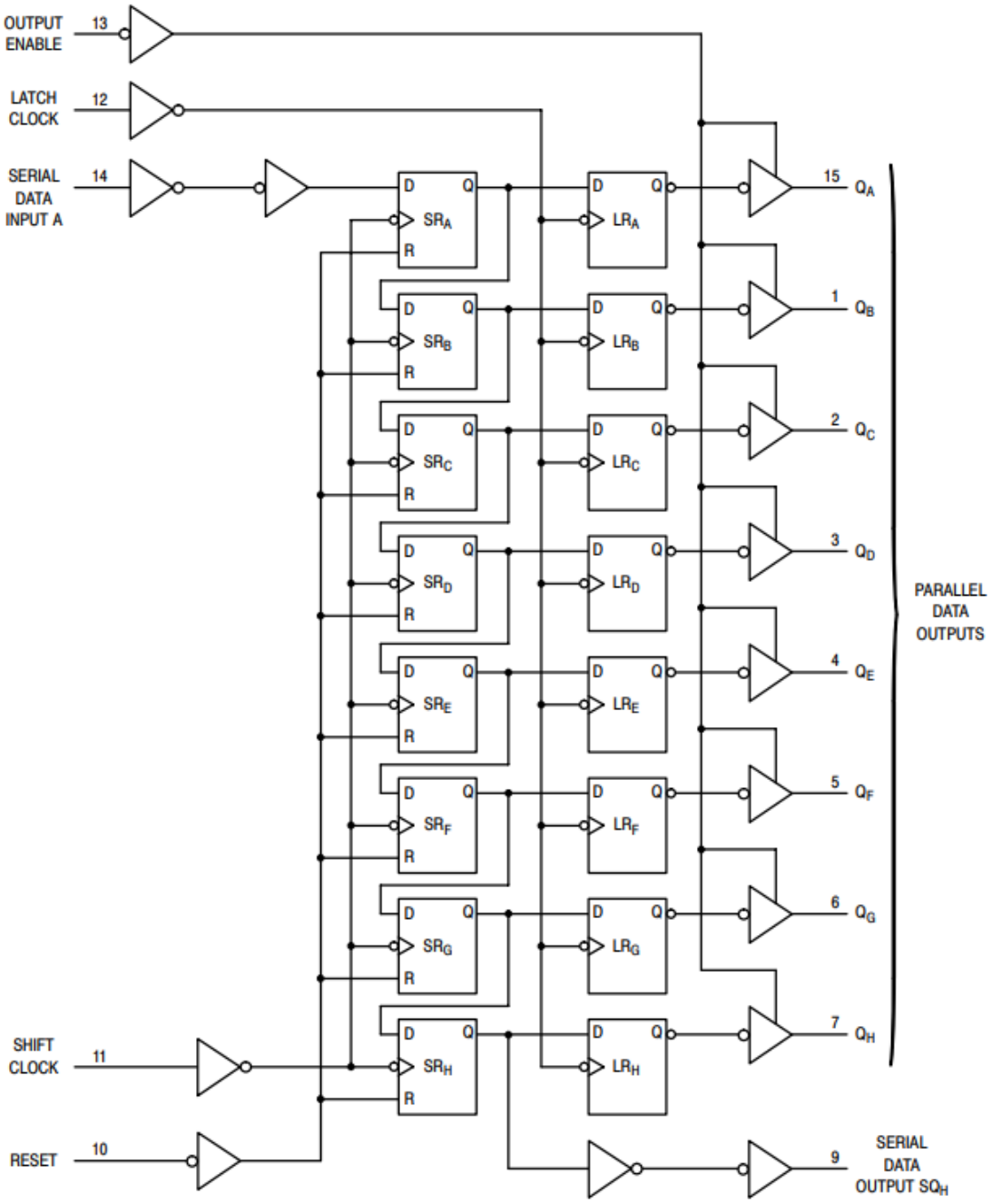


Abbildung 2: Das erweiterte Logikdiagramm des Schieberegisters 74HC595 entnommen aus: https://www.onsemi.com/pub/Collateral/MC74HC595-D.PDF

Schaltplan:

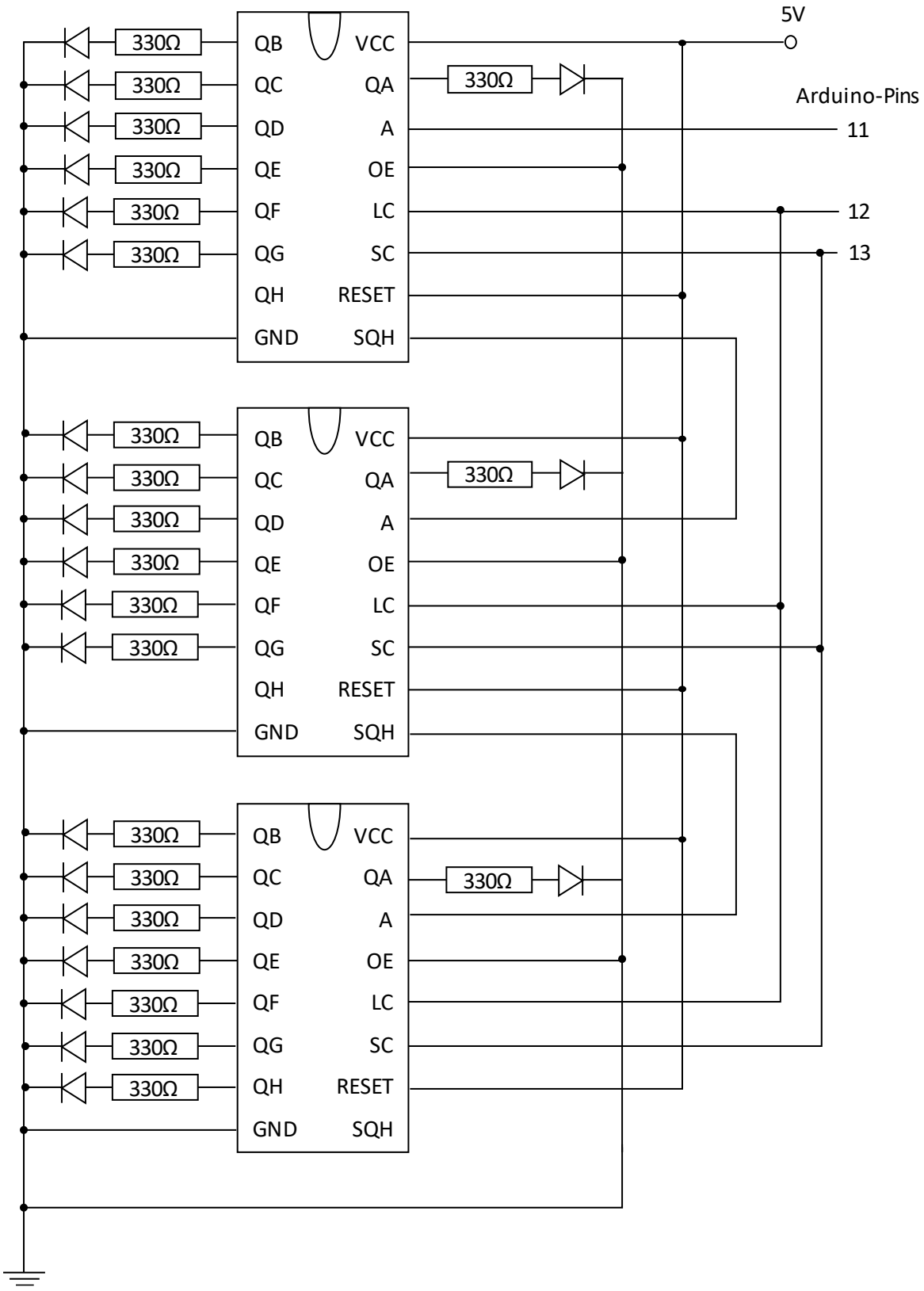


Abbildung 3: Schaltplan für die Schieberegister 74HC595 und die LEDs.

2) 15 3-polige Mini-Kippschalter:

Die Kippschalter verfügen über drei Anschlüsse, die wie in Abbildung 4 dargestellt, verschaltet sind:

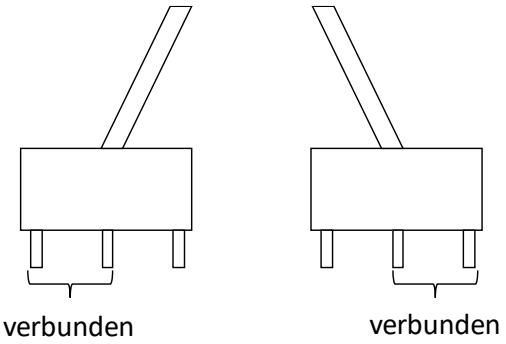


Abbildung 4: Schematische Darstellung der Verschaltung der 3-poligen Mini-Kippschalter.

Der mittlere Anschluss wird vom Arduino ausgelesen. Eine Seite wird mit 5V, die andere mit GND verbunden. Abhängig von der Position des Schalters misst der Arduino eine Spannung von 5V oder 0V. Um die Zahl der Anschlüsse von 3 auf 2 zu reduzieren, kann man den Anschluss zum Arduino als Pull-Up definieren. Dadurch ist eine Verbindung zu 5V nicht mehr notwendig, denn es wird immer 5V gemessen, es sei denn, der Schalter ist in der Stellung, in der der mittlere Anschluss mit GND verbunden ist. Dann werden 0V gemessen. Welcher der beiden Seiten mit GND verbunden wird, ist egal. Wichtig ist nur, dass die Schalter so angeschlossen werden, dass für alle Schalter die gleiche Stellung ein Anschalten bzw. Ausschalten der entsprechenden LED bezweckt. Wenn 0 V gemessen wird, wird die LED angeschaltet. Wird 5 V gemessen, wird die LED ausgeschaltet.

3) 2 Multiplexer HCF4051

Zum Auslesen der Kippschalter werden zwei Multiplexer verwendet. Für die Erläuterung der Multiplexer wurden folgende Quellen verwendet:
https://www.st.com/resource/en/datasheet/hcf4051.pdf (das Datenblatt)
https://playground.arduino.cc/Learning/4051/
Der HCF4051 kann als Multiplexer oder Demultiplexer verwendet werden. Wir möchten mehrere Kippschalter auslesen und benötigen daher mehrere Eingänge und nur einen Ausgang. Wir verwenden das Bauteil demnach als Multiplexer, kurz: MUX. Jeder Multiplexer kann 8 Kippschalter auslesen. Der erste Multiplexer wird die 7 Kippschalter der ersten Zahl und den Kippschalter zum Einstellen der Rechenoperation (+/-) auslesen, der zweite die 7 Kippschalter der zweiten Zahl.

Die Anschlüsse:

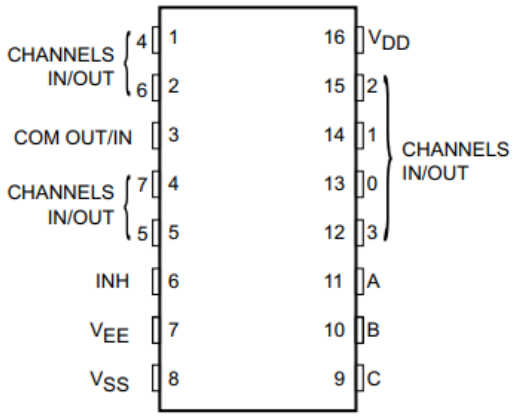


Abbildung 5: Die Anschlüsse des Multiplexers HCF4051 entnommen aus: https://www.st.com/resource/en/datasheet/hcf4051.pdf

V_{DD}: 5V (+). Dient außerdem als Vergleichspannung für alle gemessenen Spannungen.

V_{SS}: GND (-)

V_{EE}: Die Spanne an Werten, die das Bauteil lesen/interpretieren kann/soll, wird durch die minimale (V_{EE}) und maximale Spannung (V_{DD}) vorgegeben. Wird auf GND gesetzt.

COM OUT/IN: Bei der Nutzung des Bauteils als Multiplexer wird dieser Pin mit dem Arduino verbunden und dient damit als Ausgang des Multiplexers.

CHANNELS IN/OUT: Bei der Nutzung des Bauteils als Multiplexer werden diese Pins als Eingänge genutzt. Hier werden die Kippschalter angeschlossen. Die Reihenfolge ist dabei von 0 (Pin 13) für den Kippschalter der 2^0-Stelle einer Zahl bis 6 (Pin 2) für den Kippschalter der 2^6-Stelle einer Zahl. Bei dem ersten Multiplexer wird außerdem der Kippschalter für die Rechenoperation an Position 7 (Pin 4) angeschlossen. Diese Stelle bleibt bei dem zweiten Multiplexer ungenutzt.

INH: Inhibit Inputs, also Hemmung der Eingänge. Funktioniert nach dem active high Prinzip. Die Eingänge können also gelesen werden, wenn dieser Pin mit GND verbunden ist.

A,B,C: Dies sind die Kontrollpins, über die die jeweils ein Eingang mit dem Ausgang verbunden wird. Die drei Kontrollpins werden mit dem Arduino verbunden. Hierüber wird mit Hilfe einer dreistelligen binären Zahl ausgewählt, welcher der 8 Eingänge mit dem Ausgang verbunden wird. So werden nacheinander alle Eingänge mit dem Ausgang verbunden und können mit dem Arduino ausgelesen werden. In Abbildung 6 ist die Wahrheitstafel der Kontrollpins dargestellt. „ON“ channels (S) gibt an welcher Eingang für welche binäre Zahl (A,B,C) mit dem Ausgang verbunden ist. Inhibit zeigt nochmal, dass das Auslesen der Eingänge nur funktioniert, wenn der Inhibit Input auf GND gesetzt ist.

Bei der Verwendung von zwei Multiplexern können beide Multiplexer über dieselben Kontrollpins gesteuert werden. Lediglich die Ausgänge (COM OUT/IN) müssen von jedem Multiplexer separat angeschlossen werden. So werden die Multiplexer parallel ausgelesen.

Input states				"ON" channel (S)
Inhibit	C	B	A	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	None

Abbildung 6: Wahrheitstafel der Kontrollpins A,B,C des Multiplexers HCF4051 entnommen aus: https://www.st.com/resource/en/datasheet/hcf4051.pdf

Schaltplan:

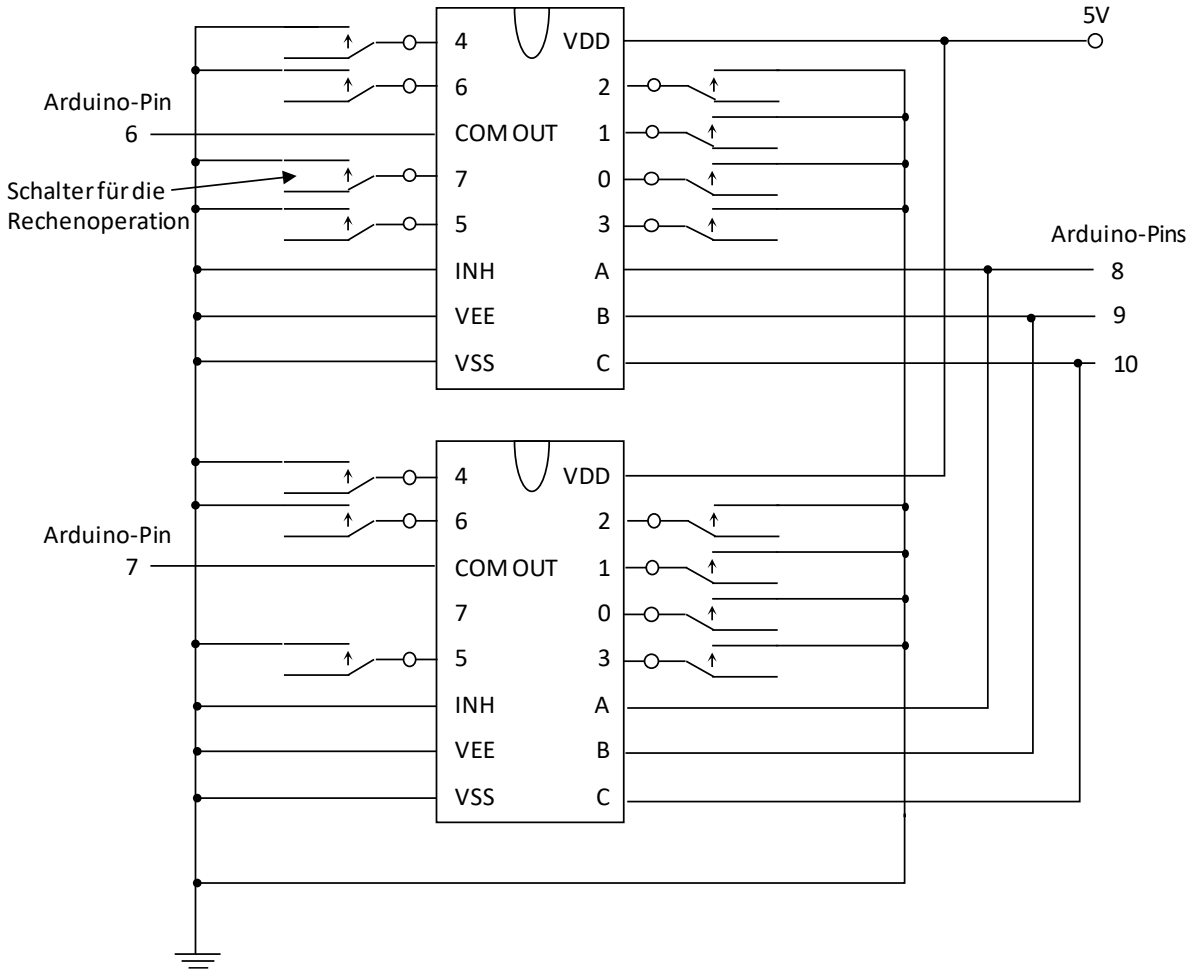


Abbildung 7: Schaltplan für die Multiplexer HCF4051 und die 3-poligen Mini-Kippschalter.

4) 1 LCD-Display Modul LCD1602

Für die Ausgabe und textliche Darstellung der Zahlen sowie der Rechnung wird das LCD-Display Modul LCD1602 verwendet, auf dem ein HITACHI-Display (HD44780U) verbaut ist. Dieses verfügt über einen großen Betrachtungswinkel und einen hohen Kontrast zur besseren Lesbarkeit. Für die Erläuterungen werden folgende Quellen verwendet:
https://ecksteinimg.de/Datasheet/CP02006/Schematic.pdf (das Datenblatt)
https://www.sparkfun.com/datasheets/LCD/HD44780.pdf (das Datenblatt speziell für HD44780)
https://www.arduino.cc/en/Tutorial/HelloWorld
Verwendet wird die Arduino-Library LiquidCrystal.h :
(https://www.arduino.cc/en/Reference/LiquidCrystal)

Die Anschlüsse:

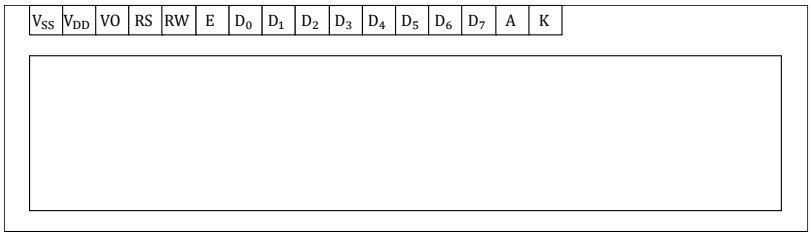


Abbildung 8: Die Anschlüsse des LCD-Displays LCD1602.

V_{SS}: GND (-)

V_{DD}: 5V (+)

VO: Der Kontrastpin. Hier kann der Kontrast des LCD-Displays eingestellt werden. Mit Hilfe eines Spannungsteilers werden wir hierüber den Kontrast optimieren. Der Spannungsteiler wird so gewählt, dass an dem Kontrastpin 1 Volt anliegt. Der Kontrast war bei diesem Wert optimal. Dafür werden wie in dem Schaltplan zu sehen zwei Widerstände, ein 1 kΩ und ein 4 kΩ Widerstand, benötigt.

RS: register select pin. Hierüber wird ausgewählt, ob Daten in das Datenregister oder Befehle in das Befehlsregister geschrieben werden (z.B. Setze den Cursor an den Anfang des Displays). Der Pin wird durch die Verwendung der LiquidCrystal.h automatisch verwendet und muss daher mit dem Arduino verbunden werden.

RW: read/write pin. Da wir nicht Auslesen wollen, was auf dem Display steht, benötigen wir nur den write-Modus. Dafür muss dieser Pin mit GND verbunden werden.

E: Enable signal. Hiermit signalisiert man den Start eines Schreibvorgangs. Wird mit dem Arduino verbunden und von der Bibliothek LiquidCrystal.h korrekt bedient.

D_0-D_7: Datenpins. Das Display kann in zwei Modi verwendet werden: im 4Bit- oder im 8Bit-Modus. im 8Bit-Modus stehen mehr Zeichen zur Verfügung als im 4Bit-Modus. Für unsere Zwecke reicht der 4Bit-Modus vollkommen aus. Daher müssen nur die Datenpins D_4-D_7 mit dem Arduino verbunden werden.

A: Anode der Hintergrundbeleuchtung. Wird über einen 220 Ω Widerstand an 5V angeschlossen.

K: Kathode der Hintergrundbeleuchtung. Wird an GND angeschlossen.

Schaltplan:

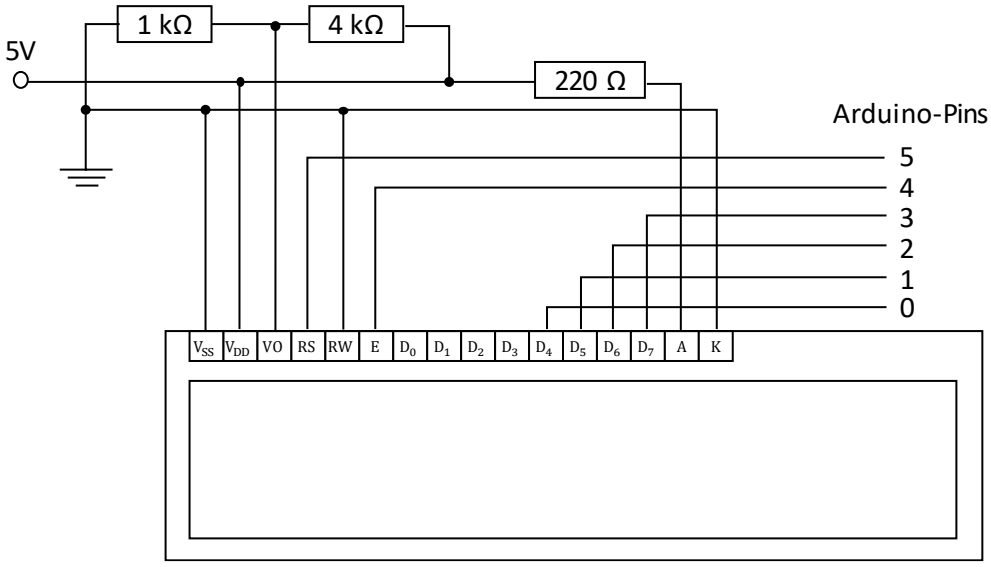


Abbildung 9: Schaltplan für das LCD-Display LCD1602.

5) Quellcode

Datei Bearbeiten Sketch Werkzeuge Hilfe

binaeresrechnen

```
#include <ShiftRegister74HC595.h>
#include <LiquidCrystal.h>

// erstelle ein globales Schieberegister-Objekt
// Parameter: <Anzahl der Schieberegister> (data pin, clock pin, latch pin)
ShiftRegister74HC595<3> sr(11, 13, 12);

// erstelle die globalen Variablen für den Multiplexer
// A, B, C sind Kontrollpins des Multiplexers
int A = 8; // A ist an digitalem Pin 8 des Arduinos angeschlossen
int B = 9;
int C = 10;
// Ausgänge der Multiplexer
int OUT1 = 6;
int OUT2 = 7;

// Variablen für die Binärdarstellung der Kontrollpins
int r0 = 0;
int r1 = 0;
int r2 = 0;

// Variablen für die eingestellten Zahlen
uint8_t zahl1 = 0; //vorzeichenlos, also von 0 bis 255 statt von -128 bis +127
uint8_t zahl2 = 0;

// Variable für Plus/Minus Rechnung
int plusMinus = 0;

// initalisiere LCD Display LCD1602
LiquidCrystal lcd(5, 4, 0, 1, 2, 3);

void setup() {
  // Definiere A, B, C als Ausgänge
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  // Definiere die Ausgänge der Multiplexer als Eingänge des Arduinos mit Pull-Up Widerstand
  pinMode(OUT1, INPUT_PULLUP);
  pinMode(OUT2, INPUT_PULLUP);
  // Initialisiere 16x2 LCD-Matrix
  lcd.begin(16, 2);
}

void loop() {
  // initialisiere Ausgabevariablen für Schieberegister
  uint8_t pinValues[] = { B00000000, B00000000, B00000000};
  // initialisiere String für LCD-Ausgabe
  char buffer[17];

  // Lese die Schalter mit Hilfe der Multiplexer aus
  // Durchlaufe die Eingänge 0 bis 6 an beiden Multiplexern parallel
  for (int count = 0; count <= 6; count++) {
    // Wandle den Zähler (0 bis 6) mit bitRead in Binärdarstellung um
    r0 = bitRead(count, 0);
    r1 = bitRead(count, 1);
    r2 = bitRead(count, 2);
```

```
    // Schreibe die Binärdarstellung in die Kontrollpins
    digitalWrite(A, r0);
    digitalWrite(B, r1);
    digitalWrite(C, r2);
    delay(1); // warte bis der Multiplexer den Kanal geschaltet hat
    // lese mit digitalRead den aktuell gewählten Eingang des Multiplexers aus
    // und setze das Bit an die richtige Stelle
    bitWrite(zahl1, count, !digitalRead(OUT1));
    bitWrite(zahl2, count, !digitalRead(OUT2));
  }

  // Lese Rechenoperation aus
  // Der Schalter für PlusMinus liegt am 8ten Eingang des ersten Multiplexers an
  digitalWrite(A, 1);
  digitalWrite(B, 1);
  digitalWrite(C, 1);
  delay(1); // warte bis der Multiplexer den Kanal geschaltet hat
  plusMinus = !digitalRead(OUT1);

  // Setze die LEDs über die Schieberegister
  pinValues[0] = zahl1; // Zahl 1
  pinValues[1] = zahl2; // Zahl 2
  // ist plusMinus = 1, wurde + gewählt
  if (plusMinus == 1) {
    pinValues[2] = pinValues[0] + pinValues[1];
    // Setze Zeichenfolge für LCD-Display
    sprintf(buffer, "%d + %d = %d", pinValues[0], pinValues[1], pinValues[2]);
    // Zeige Rechnung auf LCD-Display an
    lcd.setCursor(0, 0);
    lcd.clear();
    lcd.print(buffer);
  }
  // ist plusMinus = 0, wurde - gewählt
  else {
    // überprüfe, ob ein negatives Ergebnis raus käme.
    if (pinValues[1] > pinValues[0]) {
      pinValues[2] = 0;
      // Setze Zeichenfolge für LCD-Display
      sprintf(buffer, "%d - %d FEHLER", pinValues[0], pinValues[1]);
      // Zeige Fehler auf LCD-Display an
      lcd.setCursor(0, 0);
      lcd.clear();
      lcd.print(buffer);
      lcd.setCursor(0,1);
      sprintf(buffer, "Ergebnis negativ");
      lcd.print(buffer);
    }
    else {
      pinValues[2] = pinValues[0] - pinValues[1];
      // Setze Zeichenfolge für LCD-Display
      sprintf(buffer, "%d - %d = %d", pinValues[0], pinValues[1], pinValues[2]);
      // Zeige Rechnung auf LCD-Display an
      lcd.setCursor(0, 0);
      lcd.clear();
      lcd.print(buffer);
    }
  }
  // schalte LEDs
  sr.setAll(pinValues);
}
```

Aufgabenbeispiele

Die Lernhilfe soll vor allem Schüler*innen mit Förderbedarf bei ihren individuellen Lernwegen helfen. Die folgenden Aufgaben und Fragen sind Ideen für Aufgabentypen. Eine konkrete Auswahl sollte die Lehrkraft in Hinblick auf den*die individuelle*n Schüler*in vornehmen.

1.) Dualsystem verstehen

a) Probiere aus für welche Zahl die Bezeichnungen über den Schaltern stehen und schreibe diese gemeinsam mit den Bezeichnungen in eine Tabelle.
b) Wenn du die Lichter für die eins und für die zwei anmachst. Welche Zahl im Dualsystem wird dann dargestellt? Und warum?
c) Wie ist das bei den Lichtern der eins und der vier?
d) Überlege an Hand deiner Tabelle welche Lichter du für die Zahl sieben anmachen musst. Überprüfe dein Ergebnis.
e) Aus wie vielen Stellen besteht die Zahl 8?
f) Welche Zahl ist die größtmögliche, die sich mit der Lernhilfe einstellen lässt?
g) Warum leuchten bei der Zahl 8 weniger Lichter als bei der Zahl 7?
h) Lässt sich an der Anzahl der leuchtenden Lichter sagen, ob eine Zahl größer ist, als eine andere?
i) Beschreibe, wie du an Hand der Lichter erkennst, ob eine Zahl größer ist als die andere.

2.) Addieren/Subtrahieren im Dualsystem

a) Welche zwei Zahlen sollte man mit einander addieren, damit das Ergebnis 10 beträgt und möglichst wenige Lichter aufleuchten?
b) Mache bei beiden Zahlen jeweils direkt übereinanderstehende Lichter an? Was passiert mit den Lichtern des Ergebnisses, wenn du „+“ aktiviert hast und was bei „-"?
c) Welche Lichter musst du bei den Zahlen anmachen, damit im Ergebnis das Licht ganz links aufleuchtet?
d) Aktiviere bei beiden Zahlen übereinander liegende Licht. Was musst du tun, damit das Ergebnislicht darunter ebenfalls leuchtet?

Dokumentation des Arbeitsprozess

Meilensteine:

Idee steht:	05.12.19
Prototyp funktionsfähig:	16.01.20
Hülle fertig:	23.01.20
Dokumentation, Anleitung und Beispiele fertig:	30.01.20

Vorgehen

- Suche nach Ideen, Festlegung auf ein Projekt
- Entwurf eines groben Aufbaus. Verworfene Ideen:
    - Neben Addition und Subtraktion auch Multiplikation und Division
    - Eingabe von 8Bit Zahlen. (Zahlen 1 und 2 haben aus folgenden Gründen 7 Stellen, das Ergebnis 8: 1) Addiert man 1111111 mit 1111111 erhält man 11111110, was mit 8 LEDs also noch darstellbar ist. 2) Jeder Multiplexer kann 8 Schalter ansteuern. Da wir noch einen Schalter für die Auswahl der Rechenoperation (+/-) benötigen und keine Anschlüsse am Arduino mehr frei sind, müssen wir mit den beiden Multiplexern auskommen. Also steuert der erste die Zahl 1 mit 7 Stellen und die Rechenoperation und der zweite die Zahl 2 mit 7 Stellen an.
    - Eigene Displays zur Anzeige der Eingegebenen Zahlen in Dezimaldarstellung.
    - Weitere LED-Reihe zur Anzeige des Übertrags (zu wenig Anschlüsse und auch so klar)
- Einarbeitung in die Funktion von Arduino, Schieberegister, Multiplexer
- Entwicklung eines funktionalen Prototypens, Code und Schaltplan
- Entwurf Hülle und Einbau der Technik. (Hier haben wie festgestellt, dass wenn Schaltplan und Code bekannt sind, es sinnvoll ist, erst die Hülle zu konstruieren, da wir unseren ersten Prototypen noch einmal komplett auseinander nehmen mussten, um ihn in der Hülle einzubauen.

Zeitplan/Dokumentation:

29.11	3h	Erste Schritte mit Arduino
04.12	1h	Konzeptplanung Schaltung
05.12	7h	Mini-Prototyp (Simon), gemeinsame Planung der benötigten Materialien
12.12	3h	Weitere Schritte mit Arduino
16.12	2h	Erste Schritte mit dem Multiplexer
19.12	3h	Schalter löten
20.12	1h	Schalter fertig löten (Simon)

27.12	3h	Schieberegister verstehen und Testprogramm (Daniela)
28.12	9h	Multiplexer verstehen und Testprogramm (Daniela)
02.01	5h	Schieberegister in unsere Schaltung integrieren und Code schreiben (Daniela)
03.01	9h	Multiplexer in unsere Schaltung integrieren und Code schreiben (Daniela)
04.01	8h	Display integrieren und Code schreiben (Daniela), Beginn Dokumentation (Daniela)
05.01	8h	Dokumentation mit Schaltplan und Quellcode (Daniela)
06.01	4h	Dokumentation mit Schaltplan und Quellcode fertigstellen (Daniela)
09.01	2h	Konstruktion Hülle (Simon + Anna)
11.01	4h	Anschlüsse am Display löten (Anna)
14.01.	4h	Zeichnung Hülle + Konstruktionsplan an Matthias (Simon)
16.01	2h	Kippschalter in Hülle einschrauben, Kabel anlöten (Anna)
16.01	1h	Anschluss des neuen Displays, Fehlersuche (Display kaputt)(Daniela + Anna)
18.01	5h	LEDs einkleben, Schalter und LEDs an Kupferband anlöten (Anna)
19.01	5h	Kabel an LEDs gelötet, Funktionstest, Anordnung der Komponenten in der Box, Entwurf neuer Außenwände (Anna)
23.01	4h	Einbau des neuen Displays, Stecken der Kabel, Fehlersuche (Display funktioniert immer noch nicht, Vermutung Widerstände defekt)
28.01	4h	Austausch der Widerstände am Display, Dokumentation weiter schreiben (Anna)
29.01.	2h	Fehlerbehebung Display (Daniela + Anna)
30.01.	4h	Abschluss Dokumentation, Aufgabentypen für den Unterricht (Simon)