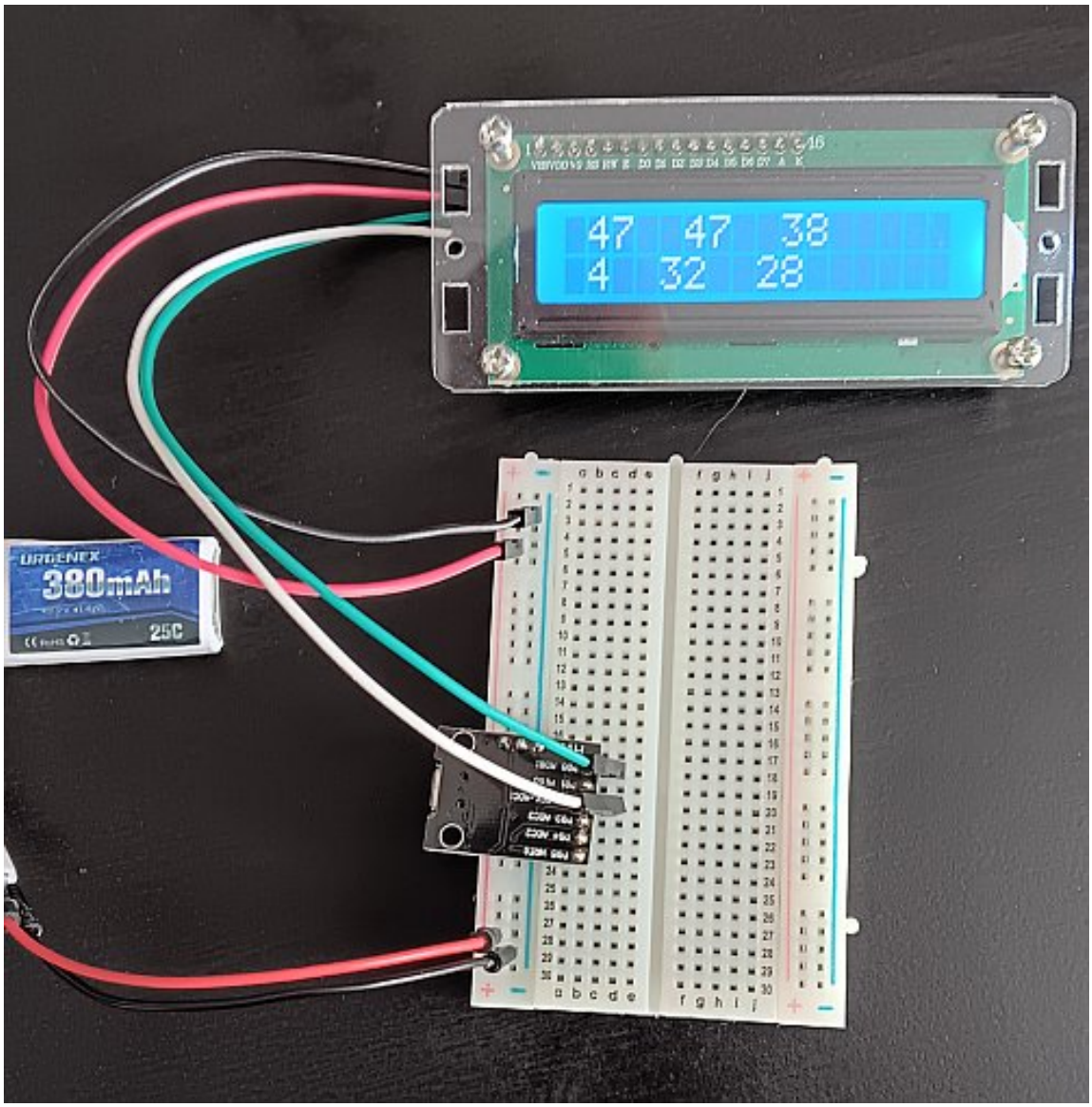
Lottozahlengenerator

Vers.2

Unzulänglichkeiten in Version 1

Bei Ausführung der Version 1 des Programmes gibt es einige Dinge, die uns nicht gefallen:



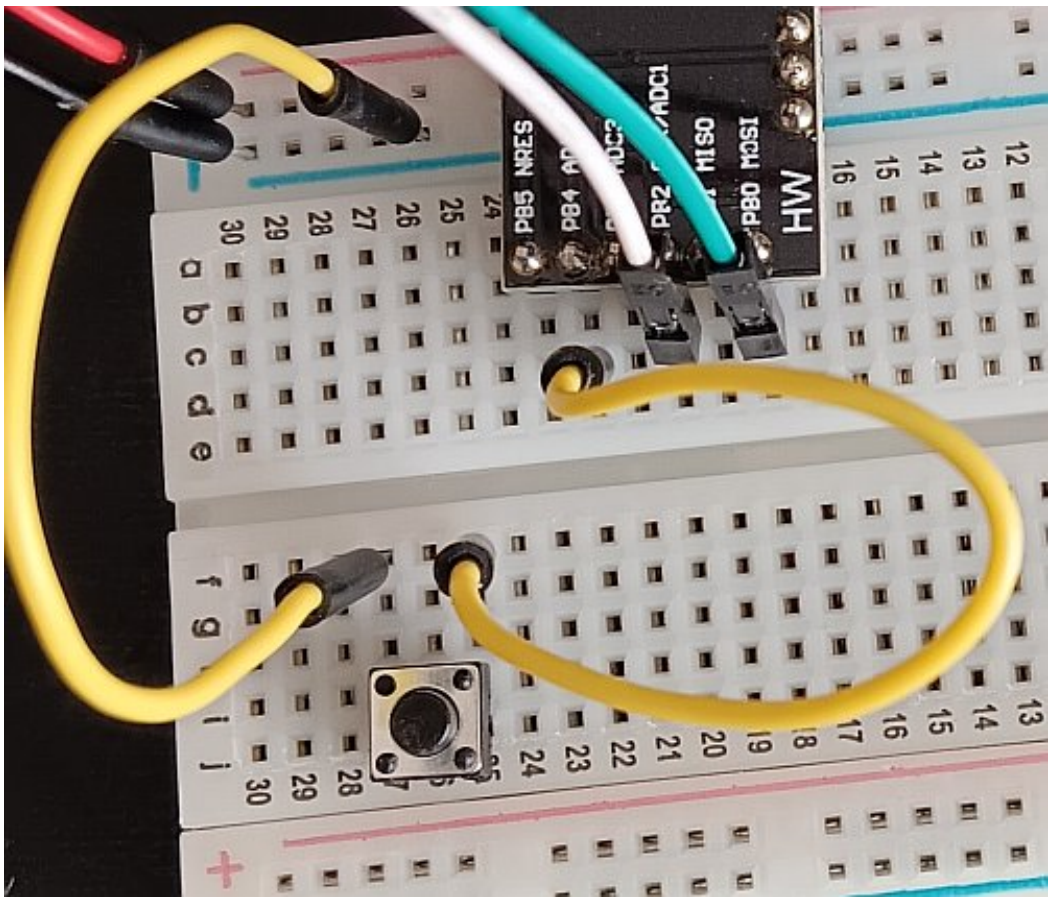
1. Es kann passieren, dass eine Zahl doppelt gezogen wird
2. Das Ergebnis wird nicht sortiert angezeigt
3. Die nächste Ziehung erfolgt automatisch nach einer vorgegebenen Zeitspanne von 5 Sekunden.

Lottozahlengenerator (Version 2)

In der zweiten Version des Lottozahlengenerators sollen diese Unzulänglichkeiten beseitigt werden.

Erweiterung der Schaltung: Mikro-Tastschalter

Die Schaltung wird mit einem Mikro-Tastschalter erweitert.
Durch Betätigung kann der Benutzer signalisieren, wann eine neue Ziehung erfolgen soll.
Der Anschluss erfolgt an Pin 4 des Digispark und wird über den Minuspol geschlossen.



Achtung:
Der Mikro-Tastschalter funktioniert nicht, wenn die Stromversorgung des Controllers über die USB-Schnittstelle realisiert wird!

In diesem Fall ist der Pin 4 des Digispark mit dem USB-Minus-Anschluss verbunden - Pin 4 liegt also dauerhaft auf "LOW" (was für das Programm im Digispark dann so aussieht, also ob der Taster dauerhaft betätigt wird).

Das Programm funktioniert also nur, wenn eine externe Stromversorgung (Akku oder Batterie) verwendet wird!

Erweiterungen am Programmcode

Variablen

Folgende Variablen werden definiert, die im Programm verwendet werden

```
int zahlen[50];
```
Ein sogenanntes "Array": Es gibt nicht nur eine Variable "zahlen", sondern 50 Stück:
zahlen[0], zahlen[1], zahlen[2], ... , zahlen[49]

```
int gezogen;
```
In der Variablen "gezogen" merke ich mir, welche zufällige Zahl gerade gezogen wurde

```
int anzahl;
```
In "anzahl" merke ich mir, wieviele Zahlen in der aktuellen Ziehung schon gezogen wurden.
"anzahl" muss also am Ende einer Ziehung den Wert "6" haben.

```
int x1;
int x2;
```
Damit die Ausgabe des Ergebnisses einer Ziehung in einer einzigen Schleife ausgegeben werden kann, werden die Variablen "x1" und "x2" eingesetzt, die die Zeile und die Spalte angeben, in der die nächste gezogene Zahl angezeigt werden soll.

Änderungen im Setup-Modul

```
void setup() {
```

```
  pinMode(4, INPUT_PULLUP);
```
Der Taster ist an Pin 4 des Digispark angeschlossen.
Der Zustand dieses Pin soll gelesen werden, er wird folglich als "Input"-Pin definiert.
Außerdem wird er mittels eines im Digispark eingebauten Pullup-Widerstandes auf "HIGH" gesetzt, das heißt, es liegt immer die Ausgangsspannung an. Erst bei Betätigung des Tasters (der ja mit "Minus" verbunden ist), wird Pin 4 auf "LOW" gezogen. Das kann im Programm abgefragt und damit die Betätigung des Tasters erkannt werden.
Ohne den Parameter "PULLUP" wäre der Zustand des Pin "undefiniert", das heißt, man könnte zum Programmstart nichts über dessen Zustand sagen!

```
  lcd.init();
  lcd.backlight();
  lcd.setCursor(2, 0);
  lcd.print("Lottozahlen");
  delay(2000);
  lcd.setCursor(0, 0);
  lcd.print("   Zum  Start   ");
  lcd.setCursor(0, 1);
  lcd.print("  Taste druecken  ");
```
Nichts Neues - das kennen wir alles schon...

```
  randomSeed(millis());
```
Hier setze ich den Startwert für die Berechnung der Zufallszahlen auf die Anzahl der Millisekunden, die seit Start des Controllers vergangen sind

```
}
```

```
void loop() {
```

```
  gezogen = random(1, 50);
```
Hiermit "ziehe" ich bei jedem Durchlauf der "Loop"-Schleife (also sehr, sehr oft...) eine zufällige Zahl, obwohl das überhaupt nicht nötig ist. Wenn dann tatsächlich irgendwann der Taster gedrückt wird und damit eine neue Ziehung ausgelöst wird, sind also vorher beliebig viele zufällige Zahlen gezogen worden und es ist nicht vorhersehbar, an welcher Stelle der "Zufallszahlenkette" ich mich dann gerade befinde. Selbst wenn also bei jedem Start des Controllers im Setup-Block mit "millis()" derselbe Startwert für die Zufallszahlenberechnung verwendet wird, sind auf diese Weise dennoch die bei der Ziehung ermittelten Zufallszahlen nicht vorhersehbar.

```
  if (digitalRead(4) == LOW) {
    delay(100);
```
Hier wird abgefragt, ob der Taster gedrückt wurde. In diesem Fall wird Pin 4 mit dem Minuspol verbunden und auf "LOW" gezogen.
Ich warte dann eine Zehntelsekunde damit ich Zeit habe, den Taster wieder loszulassen: Selbst ein "kurzer" Tastendruck dauert nämlich viel zu lange und würde zu vielen weiteren Durchläufen des "Loop"-Blockes führen...

```
    for (int x = 0; x < 50; x++) {
      zahlen[x] = 0;
    }
```
Zu Beginn einer Ziehung setze ich das "Zahlen"-Array auf Null, das heißt, alle "zahlen" (zahlen[0], zahlen[1], ... , zahlen[49]) bekommen den Wert "0".

Jetzt passiert die eigentliche Ziehung:

```
anzahl = 0;
```
Die "anzahl" der gezogenen Zahlen ist am Anfang "0".

```
    while (anzahl < 6) {
```
Solange "anzahl" noch nicht "6" ist...

```
      gezogen = random(1, 50);
```
... wird eine zufällige Zahl gezogen.

```
      if (zahlen[gezogen] == 0) {
```
Wenn im "Zahlen"-Array dieser Zahl eine "0" steht, wurde diese Zahl bisher noch nicht gezogen...

```
        zahlen[gezogen] = 1;
```
... und ich schreibe da eine "1" rein (als Zeichen, dass die Zahl nun gezogen wurde)

```
        anzahl = anzahl + 1;
```
Außerdem erhöhe ich "anzahl" um 1, denn ich habe eine weitere Zahl in meiner Ziehung gefunden

```
      }
    }
```

Wenn ich hier angelangt bin, ist die "while"-Schleife von oben durchgelaufen, das heißt, es wurden sechs Zufallszahlen "gezogen". Im "Zahlen"-Array der betreffenden Zahlen steht eine "1".

```
  lcd.clear();
```
Als erstes lösche ich das Display.

```
    x1 = 2;
```
x1 ist die Spalte im Display, in der die nächste gezogene Zahl angezeigt werden soll
```
    x2 = 0;
```
x2 ist die Zeile im Display, in der die nächste gezogene Zahl angezeigt werden soll

Die erste Zahl wird also in Zeile 0, Position 2 angezeigt - entspricht also "**lcd.setCursor(2,0)**"

Jetzt wird in einer Schleife das "Zahlen"-Array analysiert und die Positionen, in denen eine "1" steht, auf dem Display ausgegeben
```
    for (int x = 0; x < 50; x++) {
      if (zahlen[x] == 1) {
```
Wenn in "zahlen[x]" eine "1" steht, wurde diese Zahl "gezogen"...
```
        lcd.setCursor(x1, x2);
        lcd.print(x);
```
... und wird an Position "x1,x2" auf dem Display ausgegeben
```
        x1 = x1 + 3;
```
Danach wird die "Spalte" um 3 erhöht (Display-Position für die nächste Ausgabe)
```
        if (x1 == 11) {
```
Es sei denn, ich bin in Spalte 11 angekommen. Das passiert, wenn ich x1 dreimal erhöht habe (2+3+3+3) - also dann, wenn ich drei Zahlen in Zeile 1 ausgegeben habe.
```
          x1 = 2;
```
Dann wechsel ich für die Ausgabe wieder in Spalte 2...
```
          x2 = 1;
```
Aber jetzt in der zweiten Zeile!
```
        }
      }
    }
```

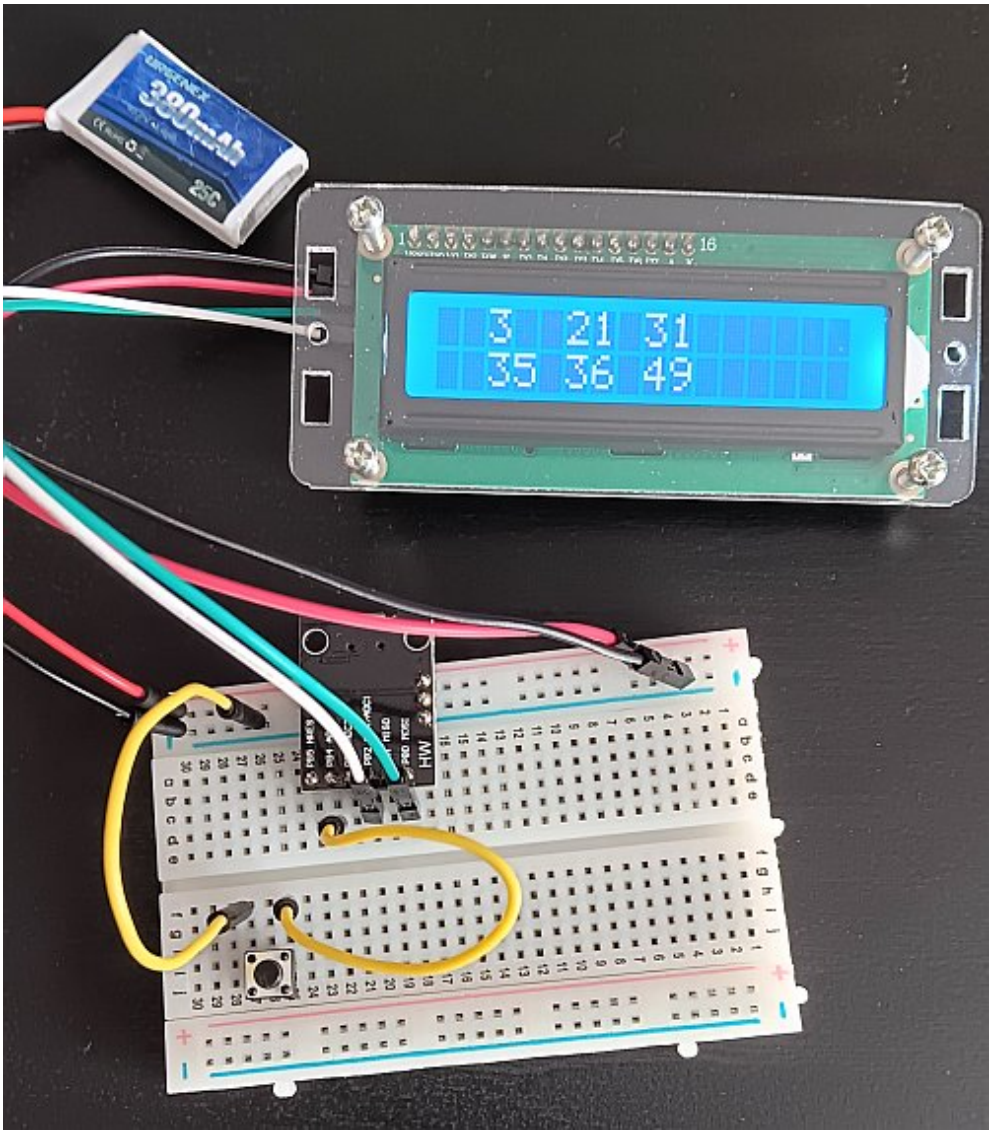
Durch die Prüfung auf "1" im Zahlen-Array wird sichergestellt, dass jede Zahl nur genau einmal gezogen wird.

Da das Zahlen-Array numerisch von 0 bis 49 analysiert wird, werden dadurch automatisch auch die "gezogenen" Zahlen numerisch sortiert ausgegeben.

Der Taster ermöglicht die Interaktion mit dem Anwender.

Aufbau

Wie gesagt: Die Stromversorgung muss extern über Batterie/Akku erfolgen.
Wird der USB-Anschluss des Digispark als Spannungsquelle verwendet, funktioniert der Taster nicht!



Code

```
// Lottozahlen (Version 2)
// Jede Zahl kann nur einmal gezogen werden
// und die Ausgabe des Ergebnisses erfolgt sortiert


#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);

int zahlen[50];
int gezogen;
int anzahl;
int x1;
int x2;

void setup() {
  pinMode(4, INPUT_PULLUP);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(2, 0);
  lcd.print("Lottozahlen");
  delay(2000);
  lcd.setCursor(0, 0);
  lcd.print("   Zum  Start   ");
  lcd.setCursor(0, 1);
  lcd.print("  Taste druecken  ");

  randomSeed(millis());
}


void loop() {

  gezogen = random(1, 50);

  if (digitalRead(4) == LOW) {
    delay(100);

    for (int x = 0; x < 50; x++) {
      zahlen[x] = 0;
    }

    anzahl = 0;
    while (anzahl < 6) {
      gezogen = random(1, 50);
      if (zahlen[gezogen] == 0) {
        zahlen[gezogen] = 1;
        anzahl = anzahl + 1;
      }
    }

    lcd.clear();

    x1 = 2;
    x2 = 0;

    for (int x = 0; x < 50; x++) {
      if (zahlen[x] == 1) {
        lcd.setCursor(x1, x2);
        lcd.print(x);
        x1 = x1 + 3;
        if (x1 == 11) {
          x1 = 2;
          x2 = 1;
        }
      }
    }
  }
}
```