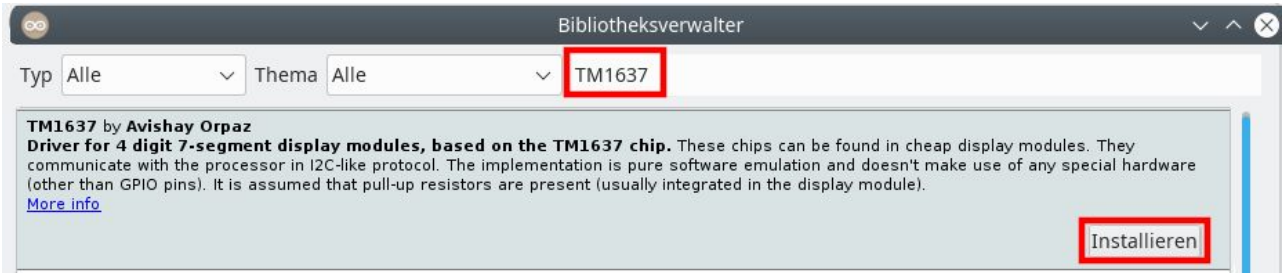
Würfelspiel für zwei Spieler mit einer 7-Segment-Anzeige

Die 7-Segment-Anzeige besteht aus 4 Zeichen. Jedes dieser Zeichen besteht aus 7 sogenannten Segmenten, die einzeln angesteuert werden können. Die Anschlüsse:

CLK → beliebiger digitaler Pin
DIO → beliebiger digitaler Pin
VCC → 5 V
GND → Ground

Als Erstes musst du die Bibliothek TM1637 installieren
Sketch → Bibliothek einbinden → Bibliotheken verwalten



Das Programm simuliert ein Würfelspiel für zwei Spieler. Jeder Spieler verfügt über einen Taster. Wird dieser gedrückt, wird die Summe der bis dahin gewürfelten Zahlen angezeigt.

Nach sechs Würfen ermittelt das Programm den Sieger und zeigt die Summe der gewürfelten Zahlen an.

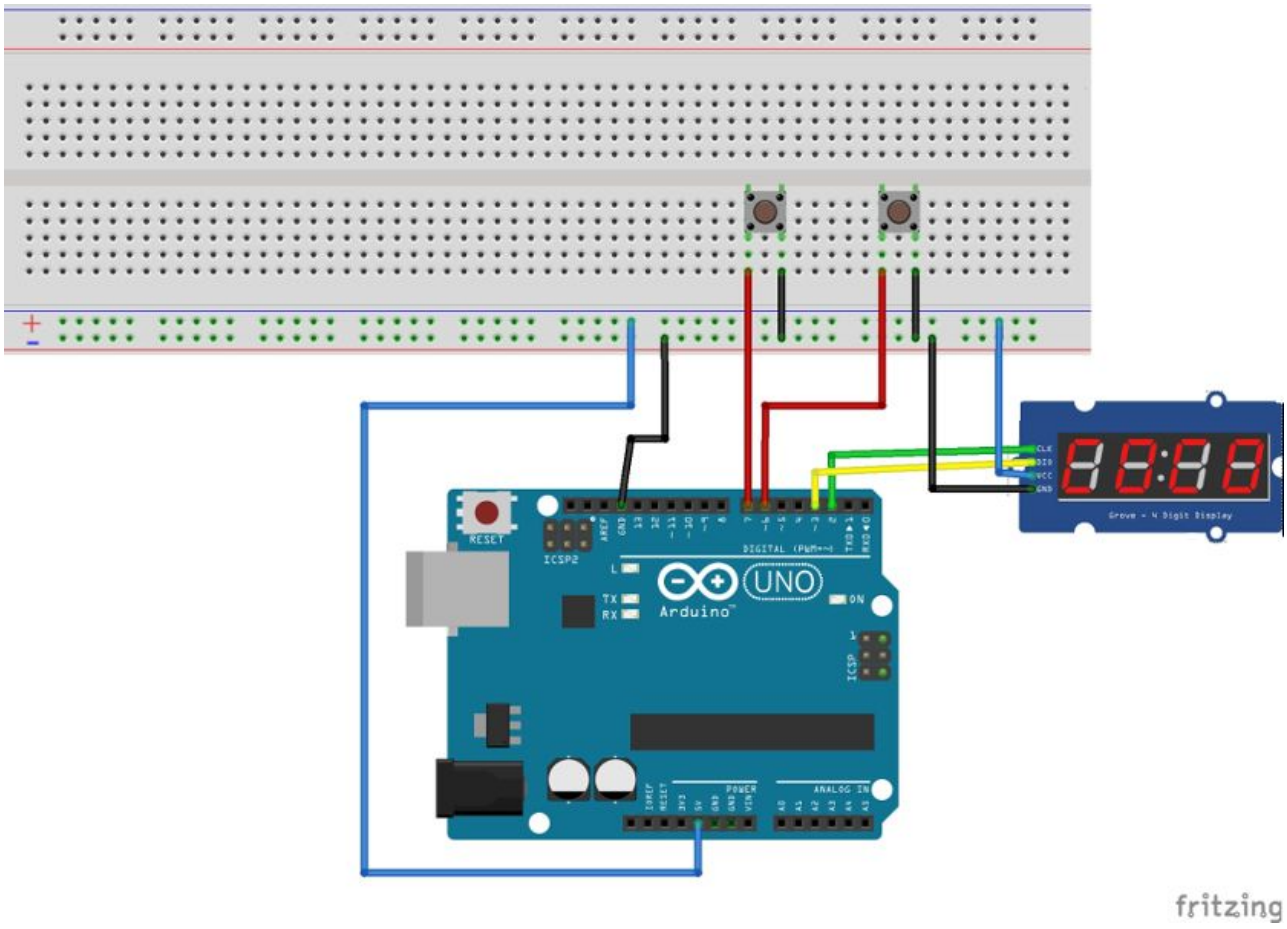
So sieht es aus:

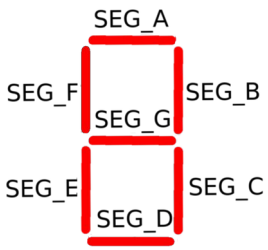
erster Spieler	Punktestand	zweiter Spieler	Punktestand	Anzeige des Siegers		Summe
SP-1	06:13	SP-2	15:24	SIEG	SP-2	24

Benötigte Bauteile:

- 2 Taster
- 7-Segment-Anzeige
- Leitungsdrähte

Baue die Schaltung auf.





Jedes der vier Zeichen besteht aus 7 Segmenten. Damit können alle Ziffern und auch einige Buchstaben dargestellt werden.

Die Buchstaben müssen in einem Array dargestellt werden:

```
// Array für die Anzeige StoP
byte STOP[] =
{
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D, // S
  SEG_F | SEG_E | SEG_G | SEG_D,         // t
  SEG_E | SEG_G | SEG_C | SEG_D,         // o
  SEG_E | SEG_F | SEG_A | SEG_B | SEG_G, // P
};
```

Beispiel: Auf Tasterdruck startet ein Countdown von 10 bis 0. Am Ende zeigt das Display „StoP“, nach einer kurzen Wartezeit erscheint „LOS“.

Befehl	**Parameter**
setBrightness(int)	0-15 → Helligkeit des Displays setzen
clear()	Anzeige löschen
showNumberDec(int, bool, int, int)	int → Zahl, die angezeigt wird false/true → führende 0 verbergen/anzeigen int → Position 1-4 (1 → links, 4 → rechts) int → Position der niederwertigsten Ziffer 0 → ganz links, 3 → rechts die beiden letzten Parameter sind optional werden sie weggelassen, erfolgt die Anzeige ganz rechts
showNumberDecEx(int, int, bool, int, int)	int → Zahl, die angezeigt wird int → 64 → Doppelpunkt anzeigen oder hexadezimal 0x40 false/true → führende 0 verbergen/anzeigen int → Position 1-4 (1 → links, 4 → rechts) int → Position der niederwertigsten Ziffer 0 → ganz links, 3 → rechts die beiden letzten Parameter sind optional werden sie weggelassen, erfolgt die Anzeige ganz rechts
setSegments(Daten_Array)	zeigt das unter Daten_Array definierte Array an

```
// Bibliothek einbinden
# include <TM1637Display.h>

// Pins für das Display definieren
int CLK =  2;
int DIO = 3;

// Pin für den Taster
int TASTER = 7;

// Name und Pins für das Display festlegen
TM1637Display Anzeige(CLK, DIO);
// Array für die Anzeige LOS
byte LOS[] =
{
  SEG_F | SEG_E | SEG_D,                          // L
  SEG_A | SEG_F | SEG_E | SEG_D | SEG_C | SEG_B,  // O
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D,          // S
  0 | 0 | 0 | 0 | 0 | 0 | 0,                     // leer
};
```

```
// Array für die Anzeige StoP
byte STOP[] =
{
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D, // S
  SEG_F | SEG_E | SEG_G | SEG_D,         // t
  SEG_E | SEG_G | SEG_C | SEG_D,         // o
  SEG_E | SEG_F | SEG_A | SEG_B | SEG_G, // P
};

void setup()
{
  pinMode(TASTER, INPUT_PULLUP);

  // Anzeige löschen
  Anzeige.clear();

  // Helligkeit des Displays setzen (1-15)
  Anzeige.setBrightness(2);

  // LOS anzeigen
  Anzeige.setSegments(LOS);
}

void loop()
{
  int Tasterlesen = digitalRead(TASTER);
  if (Tasterlesen == LOW)
  {
    Anzeige.clear();

    for (int i = 10; i >= 0; i--)
    {
      Anzeige.showNumberDec(i, false, 4, 0);
      delay(1000);
    }
    Anzeige.setSegments(STOP);
    delay(3000);
  }
  Anzeige.setSegments(LOS);
}
```

Der Kopf für das Spiel muss mit den Arrays für die Meldungen ergänzt werden:

```
// Array für SP-1
byte SPIELER1[] =
{
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D, // S
  SEG_E | SEG_F | SEG_A | SEG_B | SEG_G, // P
  SEG_G,                                 // -
  SEG_B | SEG_C,                         // 1
};
// Array für SP-2
byte SPIELER2[] =
{
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D, // S
  SEG_E | SEG_F | SEG_A | SEG_B | SEG_G, // P
  SEG_G,                                 // -
  SEG_A | SEG_B | SEG_G | SEG_E | SEG_D, // 2
};

// Array für SIEG
byte SIEG[] =
{
  SEG_A | SEG_F | SEG_G | SEG_C | SEG_D,         // S
  SEG_B | SEG_C,                                 // I
  SEG_A | SEG_F | SEG_G | SEG_E | SEG_D,         // E
  SEG_A | SEG_F | SEG_E | SEG_D | SEG_C | SEG_G, // G
};

// Array für UNENTSCHIEDEN
byte UNENTSCHIEDEN[] =
{
  SEG_G | SEG_D,        // =
};
```

Außerdem müssen noch die Definition der Taster, die Variablen für die erzielten Punkte, die Anzahl der gespielten Runden und die Reihenfolge der Spieler hinzugefügt werden:

```
// die Taster
int TASTER1 = 6;
int TASTER2 = 7;

// Summe der Augen der beiden Spieler
int SummeSpieler1;
int SummeSpieler2;

// Anzahl der Runden, Start mit 1
int Runde = 1;
```

```
int Zahl;
int TasterStatus;

// Reihenfolge der Spieler festlegen, Spieler1 startet
bool StartSpieler1 = true;
bool StartSpieler2 = false;
```

Im setup-Teil wird der Zufallsgenerator gestartet.

```
void setup()
{
  pinMode(TASTER1, INPUT_PULLUP);
  pinMode(TASTER2, INPUT_PULLUP);

  // Anzeige.clear();
  Anzeige.setSegments(AnzeigeLoeschen);

  // Helligkeit setzen(0-15)
  Anzeige.setBrightness(2);

  Anzeige.setSegments(SPIELER1);

  // Zufallsgenerator starten
  randomSeed(analogRead(0));
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  int Minimum = 1;
  int Maximum = 7;

  // 6 Runden spielen
  while (Runde < 7)
  {
    // Spieler 1 ist an der Reihe StartSpieler1 → true
    if (StartSpieler1)
    {
      TasterStatus = digitalRead(TASTER1);

      if (TasterStatus == LOW)
      {
        // Wechsel → Spieler2 ist an der Reihe
        StartSpieler1 = !StartSpieler1;
```

```
      StartSpieler2 = !StartSpieler2;

      delay(200);
      Anzeige.setBrightness(2);
      Anzeige.clear();

      // Wechsel → Spieler2 ist an der Reihe
      StartSpieler1 = !StartSpieler1;
      StartSpieler2 = !StartSpieler2;

      // Aufruf der Funktion ZufallsZahl
      SummeSpieler1 = SummeSpieler1 + ZufallsZahl(Minimum, Maximum);
      /*
        Punkte anzeigen
        showNumberDecEx(Zahl, Doppelpunkt anzeigen
        (64 oder hexadezimal 0x40),
        false/true(führende 0 verbergen/anzeigen),
        Position (1-4), 0 (rechtsbündig))
      */
      Anzeige.showNumberDecEx(SummeSpieler2, 64, true, 4, 0);
      Anzeige.showNumberDecEx(SummeSpieler1, 64, true, 2, 0);

      delay(2000);
      Anzeige.setSegments(SPIELER2);
    }
  }

  // Spieler 2 ist an der Reihe StartSpieler2 → true
  if (StartSpieler2)
  {
    TasterStatus = digitalRead(TASTER2);
    if (TasterStatus == LOW)
    {
      // Wechsel -> Spieler1 ist an der Reihe
      StartSpieler1 = !StartSpieler1;
      StartSpieler2 = !StartSpieler2;

      delay(200);
      Anzeige.setBrightness(2);
      Anzeige.clear();

      // Aufruf der Funktion ZufallsZahl
      SummeSpieler2 = SummeSpieler2 + ZufallsZahl(Minimum, Maximum);

      Anzeige.showNumberDecEx(SummeSpieler2, 64, true, 4, 0);
      Anzeige.showNumberDecEx(SummeSpieler1, 64, true, 2, 0);

      delay(2000);
```

```
      Anzeige.setSegments(SPIELER1);

      // nur bei Spieler2 Runde hochzählen, Spieler1 hat angefangen
      Runde ++;

      // Wechsel → Spieler1 ist an der Reihe
      StartSpieler1 = !StartSpieler1;
      StartSpieler2 = !StartSpieler2;
    }
  }
}
// unentschieden
if (SummeSpieler1 == SummeSpieler2)
{
  Anzeige.clear();
  Anzeige.setSegments(UNENTSCHIEDEN, 1 , 2);
  delay(2000);
  Anzeige.setSegments(SPIELER1);

  // alle Werte zurücksetzen
  Runde = 1;
  SummeSpieler1 = 0;
  SummeSpieler2 = 0;
}

// Sieger Spieler1
if (SummeSpieler1 > SummeSpieler2)
{
  Anzeige.setSegments(SIEG);
  delay(2000);
  Anzeige.setSegments(SPIELER1);
  delay(2000);
  Anzeige.showNumberDec(SummeSpieler1, false);
  delay(2000);
  Anzeige.setSegments(SPIELER1);

  // alle Werte zurücksetzen
  Runde = 1;
  SummeSpieler1 = 0;
  SummeSpieler2 = 0;
}

// Sieger Spieler2
if (SummeSpieler1 < SummeSpieler2)
{
  Anzeige.setSegments(SIEG);
  delay(2000);
  Anzeige.setSegments(SPIELER2);
  delay(2000);
  Anzeige.showNumberDec(SummeSpieler2, false);
  delay(2000);
```

```
    Anzeige.setSegments(SPIELER1);

    // alle Werte zurücksetzen
    Runde = 1;
    SummeSpieler1 = 0;
    SummeSpieler2 = 0;
  }
}
```

Jetzt fehlt nur noch die Funktion ZufallsZahl.