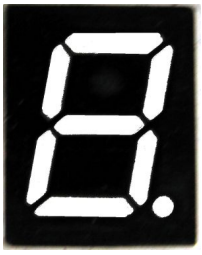
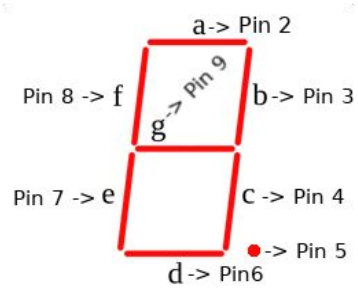
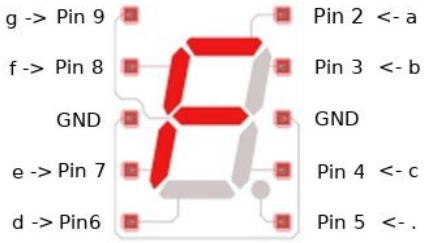
Würfeln mit einer einstelligen 7-Segment-Anzeige



Die 7-Segment-Anzeige besteht aus sieben horizontal und vertikal verlaufenden Segmenten und einem Punkt in der rechten unteren Ecke, die einzeln angesteuert werden. Es lassen sich alle Zahlen und eine Reihe von Buchstaben darstellen.



Die Segmente sind von a bis g gekennzeichnet. Jedes Segment muss mit einem Pin des Arduinos verbunden werden.



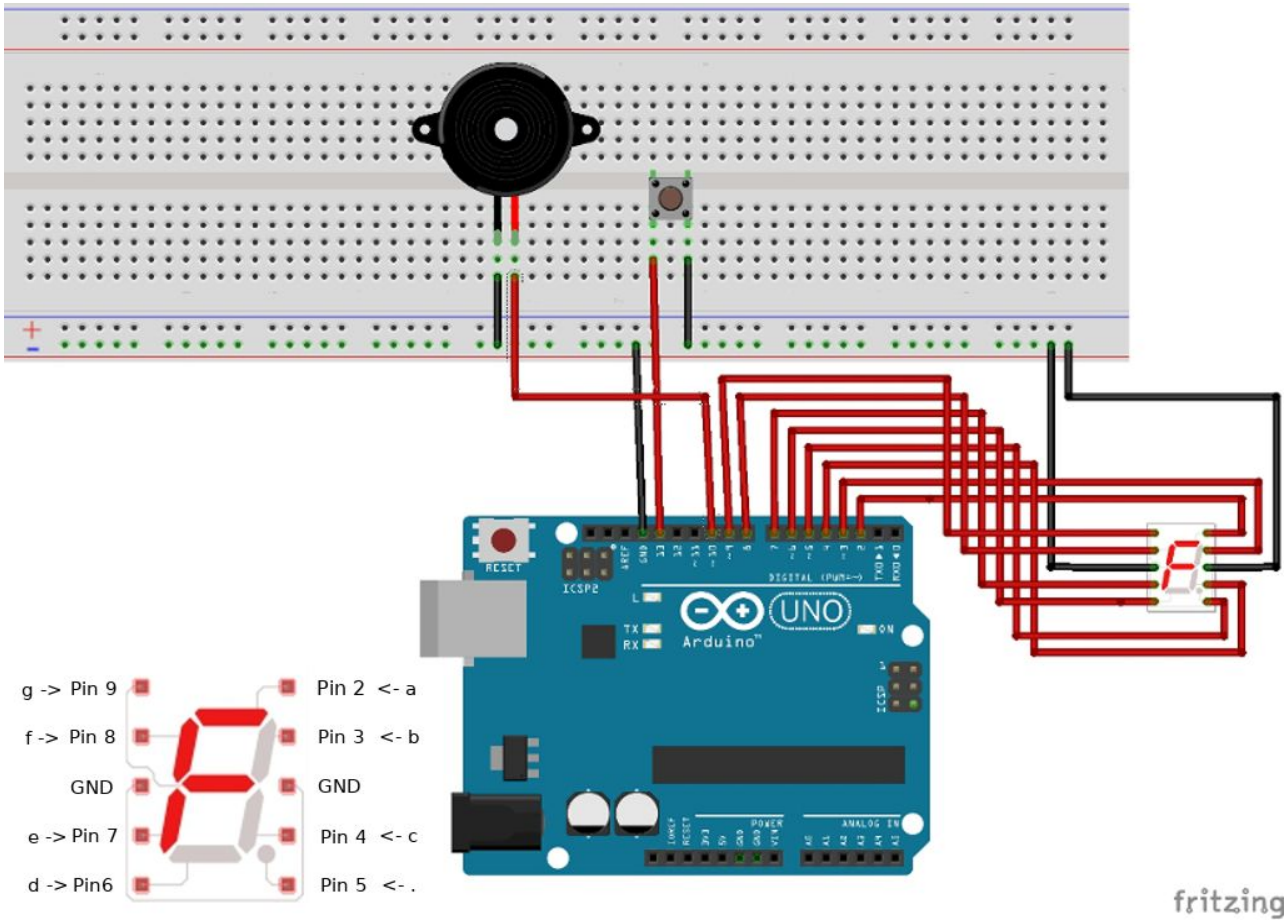
Es gibt die 7-Segmente-Anzeige in zwei Ausführungen:
entweder - (Common Cathode → GND) oder + (Common Anode → 5V).
Die verwendete Version kannst du durch einfaches Umstecken (GND/5V) herausfinden.

Das Programm würfelt eine Zahl und zeigt sie auf dem 7-Segment-Display an. Kurz angezeigte Zufallszahlen simulieren den Würfelvorgang, bevor die endgültig gewürfelte Zahl angezeigt wird.

Benötigte Bauteile:

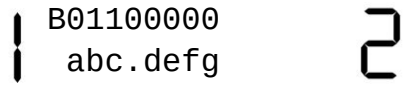
- Taster
- Einstellige 7-Segment-Anzeige
- Lautsprecher
- Leitungsdrähte

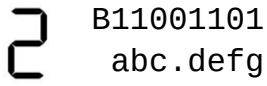
Baue die Schaltung auf.



Die Zahlen, die dargestellt werden sollen, werden als Binärwert notiert. Eine 1 steht für Segment einschalten, eine 0 zeigt das Segment nicht an. Die Reihenfolge der Ziffern entspricht der Reihenfolge der Pins. Die erste Ziffer schaltet Pin 2, die zweite Pin 3 und die letzte Pin 9.

Beispiele:





Lege die Variablen fest.

```
byte Zahlen[6] =
{
  B01100000, // 1
  B11001101, // 2
  B11101001, // 3
  B01100011, // 4
  B10101011, // 5
  B10101111, // 6
};
```

```
int TASTER = 13;
int LAUTSPRECHER = 10;
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
  // Pins auf OUTPUT setzen
  for (int i = 2; i <= 9; i++)
  {
    pinMode(i, OUTPUT);
  }

  pinMode(TASTER, INPUT_PULLUP);

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

Der loop-Teil. Beachte die Kommentare.

```
void loop()
{
  /*
   der Bereich der Zahlen 1 bis 6
   als oberer Wert muss 7 angegeben werden,
   weil immer nach unten gerundet wird
  */
  int Minimum = 1;
  int Maximum = 7;

  int TasterLesen = digitalRead(TASTER);
  if (!TasterLesen)
  {
    // Würfeleffekt
    // in schneller Folge werden 10 Zufallszahlen angezeigt
    for (int i = 0; i < 10; i++)
    {
      /*
        das Array der Zahlen beginnt mit 0 und endet bei 5
        die Würfelzahlen beginnen mit 1
        -> 1 von der gewürfelten Zahl abziehen,
        um das richtige Array anzuzeigen
      */
      ZahlZeigen(Zahlen[ZufallsZahl(Minimum, Maximum) - 1]);
      delay(100);
    }
```

```
    // gewürfelte Zahl anzeigen
    byte Zahl = ZufallsZahl(Minimum, Maximum);
    ZahlZeigen(Zahlen[Zahl - 1]);
    tone(LAUTSPRECHER, 1000, 10);
  }
}
```

Im loop-Teil wird die Methode ZahlZeigen() aufgerufen. Als Parameter wird ihr ein Element des Arrays Zahlen – einer der Binärwerte für die Zahl 9 bis 0 – übergeben.

```
void ZahlZeigen(byte ArrayZahl)
{
  // Bits des Arrays ArrayZahl prüfen
  // von Pin 2 bis Pin 9 durchlaufen
  for (int i = 2; i <= 9; i++)
  {
    /*
      vergleicht das Byte ArrayZahl mit dem Byte B10000000
      befindet sich an beiden Positionen eine 1
      das Ergebnis der Prüfung ist also nicht 0
      -> Segment einschalten
      ist eine der Positionen eine 0
      das Ergebnis der Prüfung ist 0
      -> Segment ausschalten
      1 Bit nach links schieben -> nächstes Bit prüfen
      nach 8 Durchläufen sind alle Segmente (Pins) richtig geschaltet
    */
    if ((ArrayZahl & B10000000) != 0) digitalWrite(i, HIGH);
    else  digitalWrite(i, LOW);

    ArrayZahl = ArrayZahl << 1;
  }
}
```

Die Funktion Zufallszahl ermittelt die Zufallszahl.

```
int ZufallsZahl(int Minimum, int Maximum)
{
  int Zahl =  random(Minimum, Maximum);
  return Zahl;
}
```