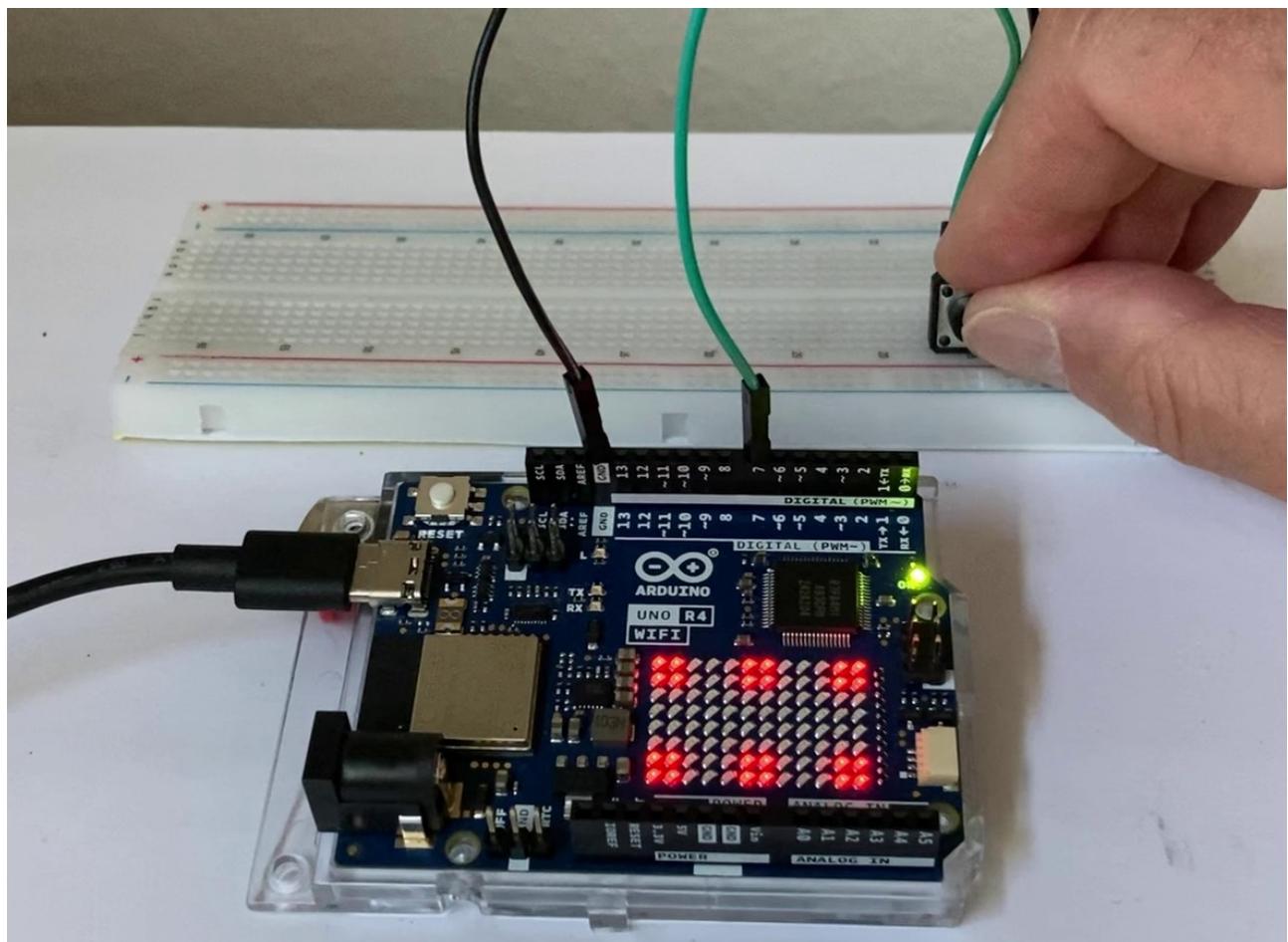


Der UNO R4 WiFi verfügt über eine 8x12 große LED-Matrix auf dem Board. Ein Taster startet den Würfelsvorgang und die LED-Matrix zeigt die Augenzahl an. Zusätzlich wird das Würfeln durch eine Folge von gewürfelten Zahlen simuliert.



Vorbereitung

Zunächst musst du über den Boardverwalter das Board installieren:

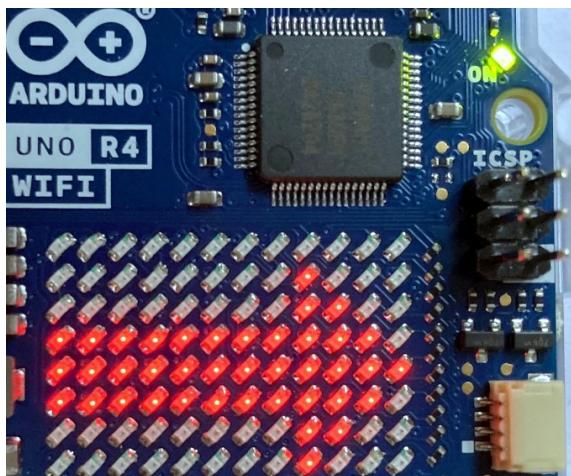
The screenshot shows the Arduino Board Manager interface. On the left is a sidebar with icons for folder, package, search, and refresh. The main area has a title bar 'BOARD-VERWALTUNG' and a search bar containing 'Arduino Uno R4'. Below the search bar is a dropdown menu 'Typ:' set to 'Alle'. A list of boards is titled 'Arduino Uno R4 Boards von Arduino' with a '...' button. Underneath the list, it says 'Boards included in this package: Arduino Uno R4 Minima, Arduino Uno R4 WiFi' and a 'Mehr Information' link. At the bottom is a large green button labeled 'INSTALLIEREN' with a small arrow icon pointing to the right. The 'Arduino Uno R4' entry in the search bar and the 'INSTALLIEREN' button are both highlighted with red boxes.



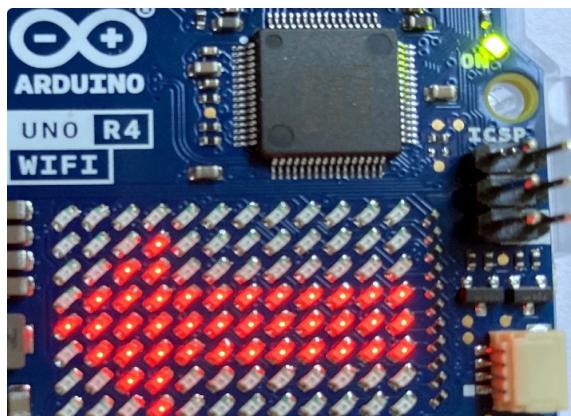
Wenn das Board angeschlossen ist, kann der USB-Anschluss ausgewählt werden. Der Name des Anschlusses unterscheidet sich je nach verwendeten Betriebssystem.

Die LED-Matrix kann als zweidimensionales Array definiert werden. Diese Schreibweise hat den Vorteil, dass der Aufbau der Matrix sichtbar wird und eine Änderung schnell möglich ist.

Beispiele:



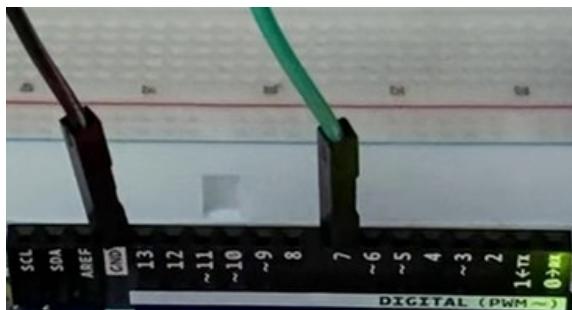
```
byte PfeilRechts[8][12] =  
{  
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
    { 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 },  
    { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0 },  
    { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0 },  
    { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },  
    { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0 },  
    { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0 },  
    { 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0 }  
};
```



```
byte PfeilLinks[8][12] =  
{  
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
    { 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },  
    { 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0 },  
    { 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },  
    { 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },  
    { 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 },  
    { 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0 },  
    { 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 }  
};
```

Benötigte Bauteile:

- Taster
- Leitungsdrähte



Verbinde den einen Pin des Tasters mit GND und den anderen mit dem digitalen Pin 7.

Binde die benötigte Bibliothek ein und definiere die Variablen.

```
#include "Arduino_LED_Matrix.h"

int TASTER = 7;

/*
 Minimum und Maximum der Zufallszahlen
 ermittelte Zahl wird immer nach unten gerundet
 -> maximaler Wert muss 7 sein
*/
int Minimum = 1;
int Maximum = 7;

// Name der Matrix
ArduinoLEDMatrix Matrix;

// Start-Button
byte StartButton[8][12] =
{
    { 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0 },
    { 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
    { 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0 },
    { 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0 },
    { 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0 },
    { 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 }
};

byte eins[8][12] =
{
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
    { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};
```

```
byte zwei[8][12] =
{
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};

byte drei[8][12] =
{
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 }
};

byte vier[8][12] =
{
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 }
};

byte fuenf[8][12] =
{
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0 },
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 },
  { 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 }
};
```

```
byte sechs[8][12] = {  
  { 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1 },  
  { 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1 },  
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
  { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },  
  { 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1 },  
  { 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1 }  
};
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()  
{  
  Matrix.begin();  
  
  // Zufallsgenerator starten  
  randomSeed(A0);  
  
  pinMode(TASTER, INPUT_PULLUP);  
  
  // Start-Button anzeigen  
  Matrix.renderBitmap(StartButton, 8, 12);  
}
```

Im loop-Teil wird die Funktion Wuerfeln() aufgerufen:

```
void Wuerfeln()  
{  
  // Zufallszahl ermitteln  
  int Zahl = random(Minimum, Maximum);  
  
  // Abfrage der gewürfelten Zahl  
  switch (Zahl)  
  {  
    case 1:  
      Matrix.renderBitmap(eins, 8, 12);  
      break;  
  
    case 2:  
      Matrix.renderBitmap(zwei, 8, 12);  
      break;  
  
    case 3:  
      Matrix.renderBitmap(drei, 8, 12);  
      break;  
  
    case 4:  
      Matrix.renderBitmap(vier, 8, 12);  
      break;  
  }
```

```
case 5:  
    Matrix.renderBitmap(vier, 8, 12);  
    break;  
  
case 6:  
    Matrix.renderBitmap(sechs, 8, 12);  
    break;  
}  
}
```

Der loop-Teil. Beachte die Kommentare.

```
void loop()  
{  
    // Zustand des Tasters lesen  
    int TasterLesen = digitalRead(TASTER);  
  
    if (TasterLesen == LOW)  
    {  
        delay(200);  
  
        // Würfeleffekt  
        for (int i = 0; i < 5; i++)  
        {  
            Wuerfeln();  
            delay(200);  
        }  
    }  
}
```

Hartmut Waller (hartmut-waller.info/arduinoblog) Letzte Änderung: 18.07.23