# Würfeln mit einem OLED-Display


OLED-Displays (Organic Light Emitting Diode) benötigen im Unterschied zu LCDs keine Hintergrundbeleuchtung, sie leuchten selbst. Eine OLED besteht aus zwei Elektroden, von denen mindestens eine transparent sein muss**.**
Im Zwischenraum befinden sich organische Halbleiterschichten aus natürlichen Farbstoffen.
Die organischen Schichten leuchten, wenn sie von Gleichstrom durchflossen werden.

Basis der Technik ist die Entdeckung der Elektrolumineszenz: ein Festkörper wird durch Anlegen einer elektrischen Spannung dazu angeregt Licht zu erzeugen.

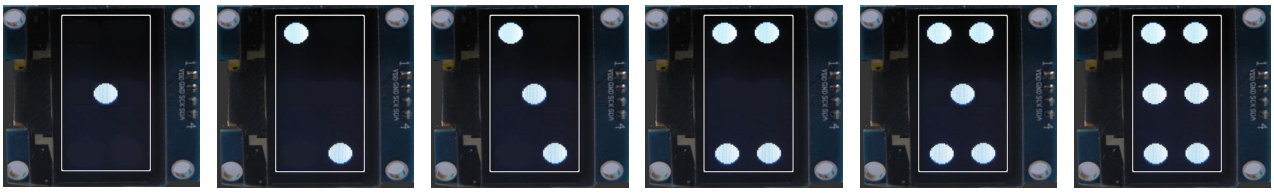
Die gebräuchlichsten Formate der Displays sind 0,96 Zoll und 1,3 Zoll. Sie unterscheiden sich beim verwendeten Chip:

0,96 Zoll Chipsatz SSD1306

1,3 Zoll Chipsatz SH1106

Das Programm würfelt auf Tasterdruck eine Zahl zwischen 1 und 6, simuliert im OLED-Display den Würfelvorgang und zeigt anschließend die gewürfelte Zahl an.

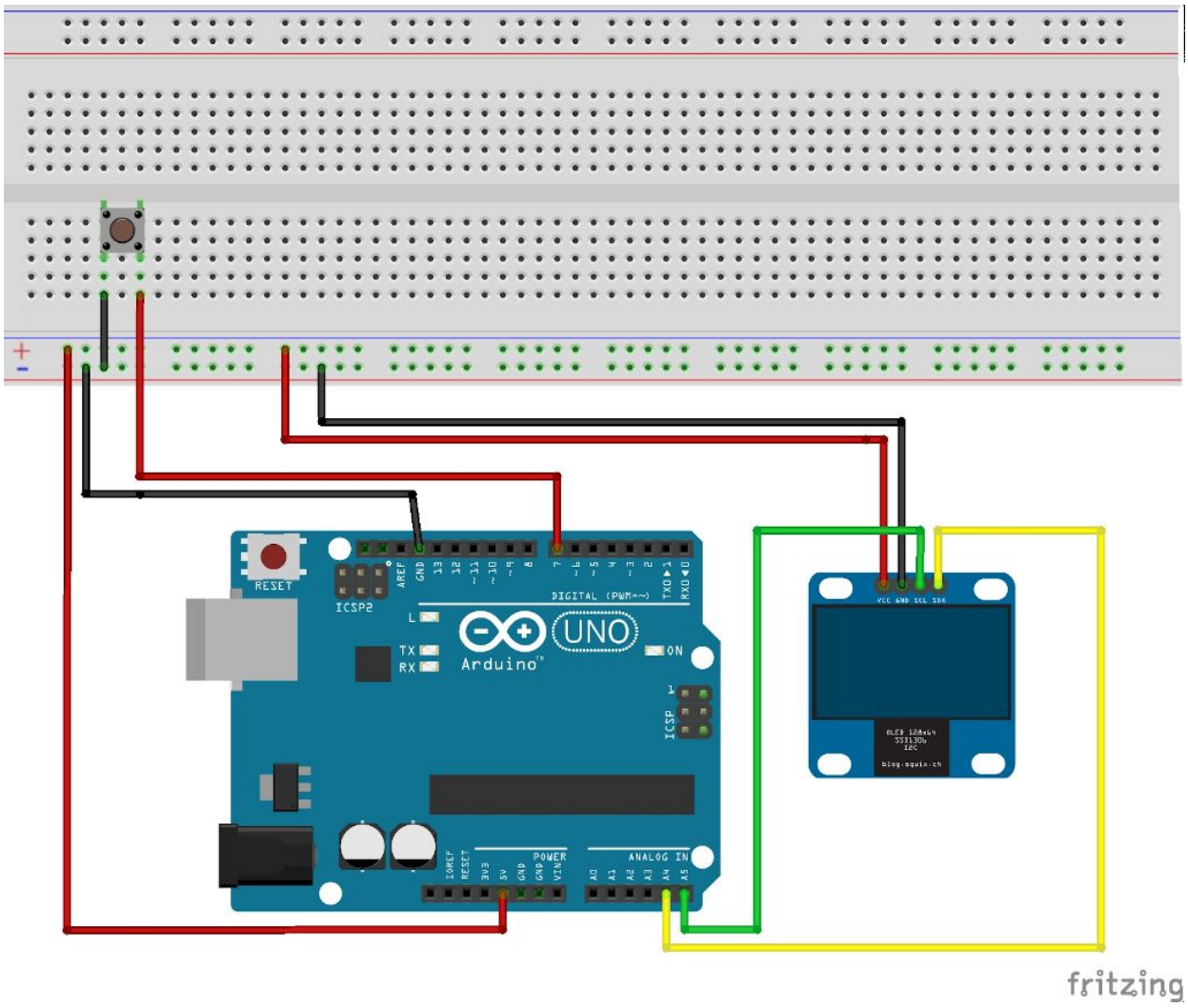
So sieht es aus:



**Benötigte Bauteile:**

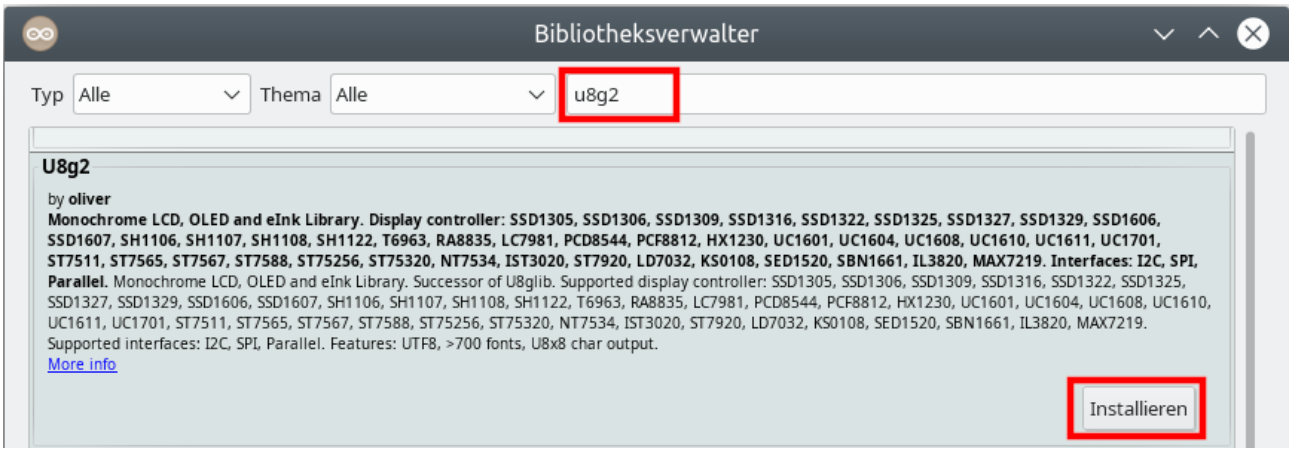
- Berührungssensor
- OLED-Display 1,3 Zoll/0,96 Zoll
- Leitungsdrähte

Baue die Schaltung auf.




**Benötigte Bibliothek:**

**Sketch → Bibliothek einbinden → Bibliotheken verwalten**

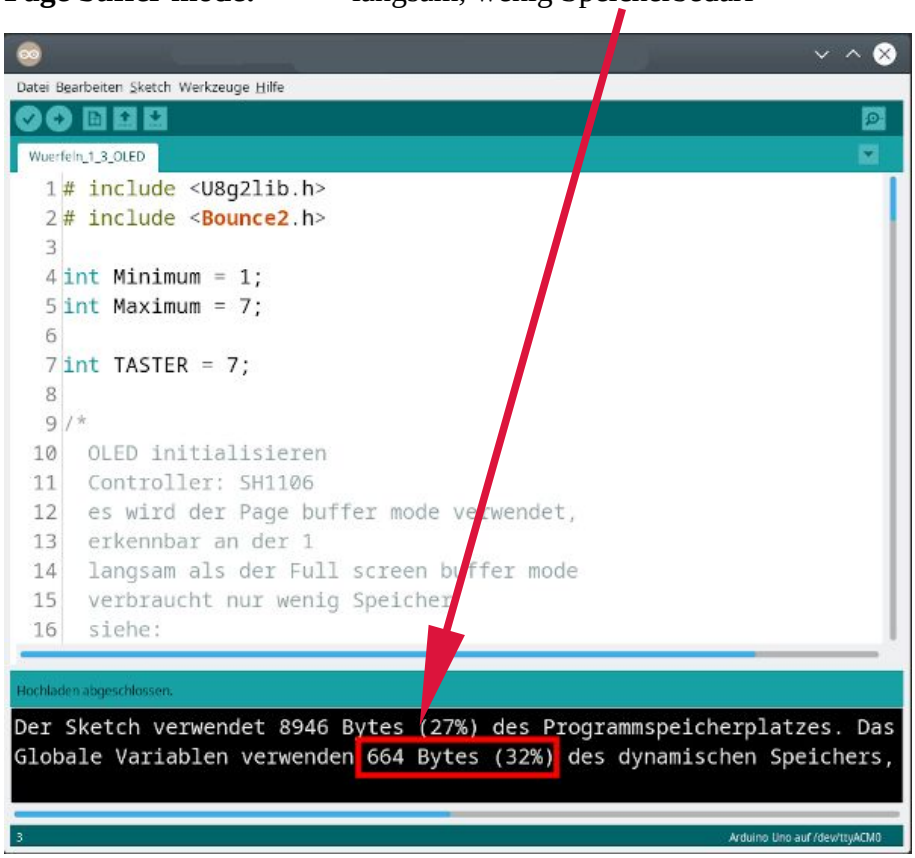


Die Bibliothek u8g2 kann eine Vielzahl von OLED-Displays ansteuern.

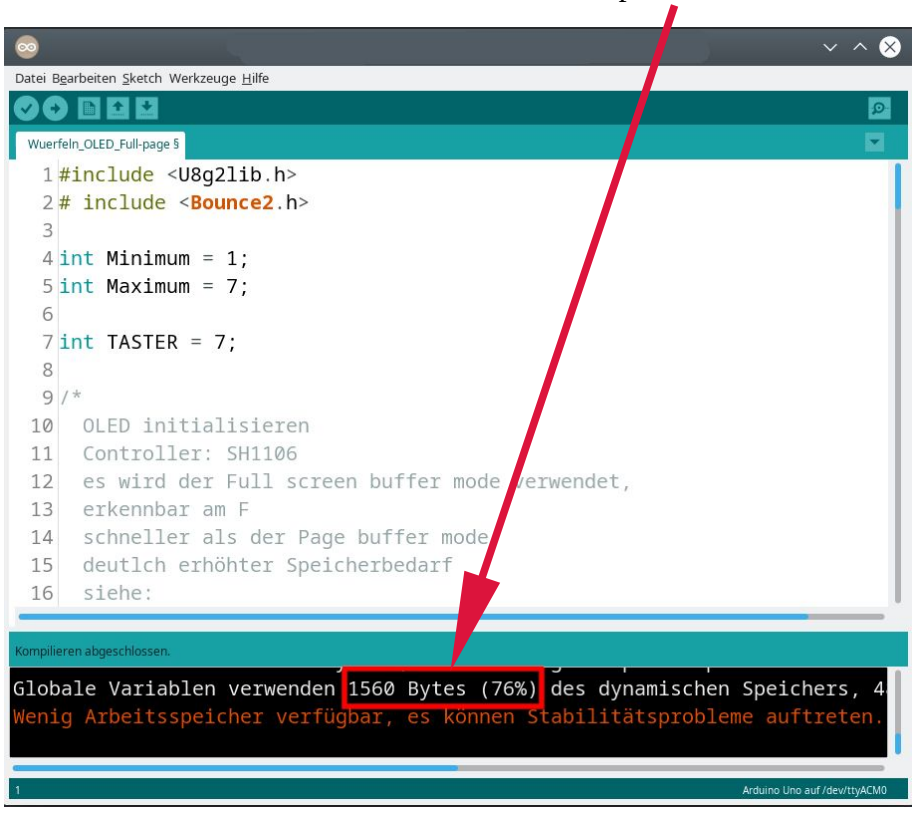
Hier sollen die Displays mit den Chipsätzen SSD1306 und SH1106 und I2C betrachtet werden.

Die Bibliothek kennt zwei verschiedene Modi den Bildschirm anzusprechen:

**Page buffer mode:** langsam, wenig Speicherbedarf




**Full screen buffer mode** schnell, sehr hoher Speicherbedarf



Sie unterscheiden sich in der Initialisierung und in der Programmierung:

Page buffer mode (1 hinter NONAME)

```
// 1,3 Zoll SH1106
U8G2_SH1106_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

// 0,96 Zoll SSD1306
U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);
```

Full screen buffer mode (F hinter NONAME)

```
// 1,3 Zoll SH1106
U8G2_SH1106_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

// 0,96 Zoll SSD1306
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);
```

**Darstellung der Inhalte:**

Page buffer mode:

```
// Farbe weiß
u8g2.setDrawColor(1);

u8g2.firstPage();
do
{
  // Rahmen mit abgerundeten Ecken an den Bildschirmrändern zeichnen
  u8g2.drawRFrame(0, 0, 128, 64, 5);
}
while (u8g2.nextPage());
```

Full screen buffer mode:

```
// Farbe weiß
u8g2.setDrawColor(1);

u8g2.clearBuffer();

// Rahmen mit abgerundeten Ecken an den Bildschirmrändern zeichnen
u8g2.drawRFrame(0, 0, 128, 64, 5);

u8g2.sendBuffer();
```

| Methode Bibliothek u8g2 | Anweisung | Parameter |
|---|---|---|
| OLED-Display starten | begin(); | |
| Bildschirmbreite feststellen | getDisplayWidth(); | |
| Bildschirmhöhe feststellen | getDisplayHeight(); | |
| Zeichenfarbe festlegen | setdrawColor(Parameter) | 0 → schwarz<br>1 → weiß |
| Bildschirm dunkel schalten | clearDisplay(); | |
| Kontrast einstellen | setContrast(Parameter) | 0 … 255 |
| Anzeige drehen | setDisplayRotation(U8G2_*); | U8G2_0 → 0 Grad<br>U8G2_1 → 90 Grad<br>U8G2_2 → 180 Grad<br>U8G2_3 → 270 Grad |
| Anzeige spiegeln (180 Grad) | flipMode(Parameter); | 0 → normale Ausrichtung<br>1 → 180 Grad drehen<br>wirksam erst bei einer Rotation |
| Cursor in die linke obere Ecke setzen | home(); | |
| einzelnen Pixel zeichnen | drawPixel(x-Achse, y-Achse) | |
| Linie zeichnen | drawLine(StartX, StartX, EndeX, EndeY); | |
| horizontale Linie zeichnen | drawHLine(StartX, StartY, Länge); | |
| vertikale Linie zeichnen | drawVLine(StartX, StartY, Länge); | |
| Rechteck zeichnen | drawFrame(StartX, StartY,, Breite, Höhe); | |
| abgerundetes Rechteck zeichnen | drawRFrame(StartX, StartY, Breite, Höhe, Eckenradius); | |
| ausgefülltes Rechteck zeichnen | drawBox(StartX, StartY, Breite, Höhe); | |
| abgerundetes ausgefülltes Rechteck zeichnen | drawRBox(StartX, StartY, Breite, Höhe, Eckenradius); | |
| Kreis zeichnen | drawCircle(MittelpunkX, MittelpunktY, Radius); | |
| ausgefüllten Kreis zeichnen | drawDisc(MittelpunkX, MittelpunktY, Radius); | |
| XBM-Bild anzeigen | drawXBM(StartX, StartY, Breite, Höhe, Array_Bilddatei); | |
| Ellipse zeichnen | drawEllipse(StartX, StartY, RadiusX, RadiusY); | |
| Cursor setzen | setCursor(x-Achse, y-Achse); | |

| Methode Bibliothek u8g2 | Anweisung | Parameter |
|---|---|---|
| Schriftart | u8g2.setFont(Schriftart) | **Beispiele für funktionierende Schriftarten:**<br>6px: u8g2_font_5x7_tr<br>7px: u8g2_font_torussansbold8_8r<br>8px: u8g2_font_ncenB08_tr<br>10px: u8g2_font_t0_15b_me<br>12px: u8g2_font_helvB12_tf<br>13px: u8g2_font_t0_22_te<br>14px: u8g2_font_helvB14_tf<br>17px: u8g2_font_timB18_tf<br>18px: u8g2_font_lubB18_tr<br>20px: u8g2_font_courB24_tf<br>23px: u8g2_font_timB24_tf<br>25px: u8g2_font_helvR24_tf<br>32px: u8g2_font_logisoso32_tf<br>42px: u8g2_font_fub42_tf<br>58px: u8g2_font_logisoso58_tf<br>62px: u8g2_font_logisoso62_tn |
| Text schreiben | print("Text");<br>drawStr(StartX, StartY ,"Text"); | |
| Schreibrichtung | setFontDirection(Wert); | 0 → normal ausgerichtet<br>1 → 90 ° gedreht<br>2 → 180 ° gedreht<br>3 → 270 ° gedreht |

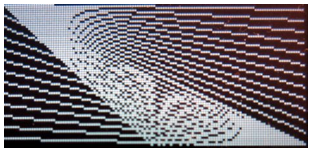
weitere Schriftarten: https://github.com/olikraus/u8g2/wiki/fntlistall

Du musst ausprobieren, welche Schriftarten dargestellt werden können!

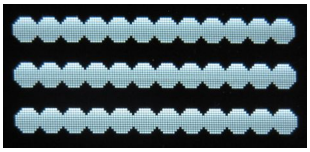
Beispiele für grafische Funktionen:

So sieht es aus:


drawXBM()


drawLine()


drawCircle()


drawDisc()

```
# include <U8g2lib.h>

// 1,3 Zoll SH1106
U8G2_SH1106_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

// 0,96 Zoll SSD1306
// U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);

int BildschirmBreite = u8g2.getDisplayWidth();
int BildschirmHoehe = u8g2.getDisplayHeight();

// Smiley XBM erstellt mit GIMP
# define SmileyBreite 46
# define SmileyHoehe 45
static unsigned char Smiley[] =
{
   0x00, 0x00, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0xc0, 0xff, 0xff, 0x00, 0x00,
   0x00, 0xf0, 0x07, 0xf8, 0x03, 0x00, 0x00, 0xfc, 0x00, 0xc0, 0x0f, 0x00,
   0x00, 0x3e, 0x00, 0x00, 0x1f, 0x00, 0x80, 0x0f, 0x00, 0x00, 0x7c, 0x00,
   0xc0, 0x07, 0x00, 0x00, 0xf8, 0x00, 0xe0, 0x01, 0x00, 0x00, 0xe0, 0x01,
   0xf0, 0x00, 0x00, 0x00, 0xc0, 0x03, 0x70, 0x00, 0x00, 0x00, 0x80, 0x03,
   0x38, 0x7e, 0x00, 0x80, 0x1f, 0x07, 0x38, 0xff, 0x00, 0xc0, 0x3f, 0x07,
   0x9c, 0xff, 0x01, 0xc0, 0x3f, 0x0e, 0x9c, 0xe7, 0x01, 0xc0, 0x39, 0x0e,
   0x8e, 0xc3, 0x01, 0xc0, 0x30, 0x1c, 0x8e, 0xe3, 0x01, 0xc0, 0x31, 0x1c,
   0x86, 0xf7, 0x01, 0xc0, 0x3b, 0x18, 0x87, 0xff, 0x01, 0xc0, 0x3f, 0x38,
   0x07, 0xff, 0x00, 0x80, 0x3f, 0x38, 0x03, 0x7e, 0x00, 0x80, 0x1f, 0x30,
   0x03, 0x00, 0x00, 0x00, 0x00, 0x30, 0x03, 0x00, 0x00, 0x00, 0x00, 0x30,
   0x03, 0x00, 0x00, 0x00, 0x00, 0x30, 0x03, 0x00, 0x00, 0x00, 0x00, 0x30,
   0x03, 0x00, 0x00, 0x00, 0x00, 0x30, 0x03, 0x00, 0x00, 0x00, 0x00, 0x30,
   0x07, 0x00, 0x00, 0x00, 0x00, 0x38, 0x07, 0x20, 0x00, 0x00, 0x01, 0x38,
   0x06, 0x70, 0x00, 0x80, 0x03, 0x18, 0x0e, 0xf0, 0x00, 0xc0, 0x01, 0x1c,
   0x0e, 0xe0, 0x03, 0xf0, 0x01, 0x1c, 0x1c, 0xc0, 0x3f, 0xfc, 0x00, 0x0e,
   0x1c, 0x80, 0xff, 0x7f, 0x00, 0x0e, 0x38, 0x00, 0xfc, 0x1f, 0x00, 0x07,
   0x38, 0x00, 0xc0, 0x03, 0x00, 0x07, 0x70, 0x00, 0x00, 0x00, 0x80, 0x03,
   0xf0, 0x00, 0x00, 0x00, 0xc0, 0x03, 0xe0, 0x01, 0x00, 0x00, 0xe0, 0x01,
   0xc0, 0x07, 0x00, 0x00, 0xf8, 0x00, 0x80, 0x0f, 0x00, 0x00, 0x7c, 0x00,
   0x00, 0x3e, 0x00, 0x00, 0x1f, 0x00, 0x00, 0xfc, 0x00, 0xc0, 0x0f, 0x00,
   0x00, 0xf0, 0x07, 0xf8, 0x03, 0x00, 0x00, 0xc0, 0xff, 0xff, 0x00, 0x00,
   0x00, 0x00, 0xfe, 0x1f, 0x00, 0x00
};

// Schneemann XBM erstellt mit GIMP
# define SchneemannBreite 28
# define SchneemannHoehe 62
static unsigned char Schneemann[] =
{
   0x00, 0xf0, 0x01, 0x00, 0x00, 0xfc, 0x07, 0x00, 0x00, 0x0e, 0x06, 0x00,
   0x00, 0x06, 0x0c, 0x00, 0x00, 0x02, 0x08, 0x00, 0x00, 0x03, 0x18, 0x00,
   0x00, 0x03, 0x18, 0x00, 0x00, 0x03, 0x18, 0x00, 0x00, 0x03, 0x38, 0x00,
   0xe0, 0xff, 0xff, 0x03, 0x00, 0xfe, 0x0f, 0x00, 0x00, 0x0f, 0x1e, 0x00,
```

```
   0x80, 0x03, 0x1c, 0x00, 0x80, 0x01, 0x38, 0x00, 0xc0, 0x19, 0x37, 0x00,
   0xc0, 0x1c, 0x37, 0x00, 0xc0, 0x18, 0x27, 0x00, 0xc0, 0x00, 0x20, 0x00,
   0xc0, 0x08, 0x21, 0x00, 0xc0, 0xf9, 0x31, 0x00, 0xc0, 0xf1, 0x38, 0x00,
   0x80, 0x03, 0x38, 0x00, 0x80, 0x07, 0x1c, 0x00, 0x00, 0x1f, 0x0f, 0x00,
   0x00, 0xfc, 0x07, 0x00, 0x00, 0xfc, 0x03, 0x04, 0x00, 0xff, 0x0f, 0x06,
   0x80, 0x07, 0x1e, 0x06, 0xc0, 0x03, 0x3c, 0x03, 0xe0, 0x00, 0x70, 0x03,
   0xf0, 0x00, 0xf0, 0x01, 0x70, 0x00, 0xe0, 0x00, 0x38, 0x00, 0xc0, 0x01,
   0x38, 0xf0, 0xc0, 0x01, 0x18, 0xf0, 0xe0, 0x01, 0x18, 0xf0, 0xb0, 0x01,
   0x0c, 0x70, 0x30, 0x03, 0x0c, 0x00, 0x18, 0x03, 0x0c, 0x00, 0x18, 0x03,
   0x0e, 0x00, 0x08, 0x07, 0x06, 0x00, 0x0f, 0x06, 0x06, 0x00, 0x0f, 0x06,
   0x06, 0x30, 0x06, 0x06, 0x06, 0x70, 0x00, 0x06, 0x06, 0xf0, 0x00, 0x06,
   0x06, 0xf0, 0x00, 0x06, 0x0e, 0x60, 0x00, 0x07, 0x0c, 0x00, 0x00, 0x03,
   0x0c, 0x00, 0x00, 0x03, 0x0c, 0x00, 0x00, 0x03, 0x0c, 0x00, 0x00, 0x03,
   0x18, 0x00, 0x80, 0x01, 0x38, 0x60, 0xc0, 0x01, 0x38, 0xf0, 0xc0, 0x01,
   0x78, 0xf0, 0xe0, 0x00, 0xf0, 0xf0, 0xf0, 0x00, 0xf0, 0x00, 0x70, 0x00,
   0xe0, 0x03, 0x7c, 0x00, 0xe0, 0x07, 0x3e, 0x00, 0xe0, 0xff, 0x3f, 0x00,
   0xc0, 0xff, 0x1f, 0x00, 0x00, 0x00, 0x00, 0x00
};

void setup()
{
  // Zufallsgenerator starten
  randomSeed(A0);

  // Display starten
  u8g2.begin();

  // Kontrast maximal 255
  u8g2.setContrast(200);
}

void loop()
{
  // Farbe weiß
  u8g2.setDrawColor(1);

  // Smiley
  u8g2.firstPage();
  do
  {
    u8g2.drawXBM(40, 10, SmileyBreite, SmileyHoehe, Smiley);
  }
  while (u8g2.nextPage());
  delay(2000);
```

```
// Schneemann
u8g2.firstPage();
do
{
  u8g2.drawXBM(50, 1, SchneemannBreite, SchneemannHoehe, Schneemann);
}
while (u8g2.nextPage());
delay(2000);

// Pixelmuster
u8g2.firstPage();
do
{
  for (int i = 0; i < 500; i ++)
  {
    int x = random(1, BildschirmBreite);
    int y = random(1, BildschirmHoehe);
    u8g2.drawPixel(x, y);
  }
}
while (u8g2.nextPage());
delay(2000);

// u8g2.setFont(u8g2_font_t0_22_te);
u8g2.setFont(u8g2_font_unifont_t_symbols);

// Text horizontal
u8g2.firstPage();
do
{
  u8g2.setFontDirection(0);
  u8g2.setCursor(2, BildschirmHoehe / 2);
  u8g2.print("Text");
}
while (u8g2.nextPage());
delay(2000);

// Text 90 Grad gedreht
u8g2.firstPage();
do
{
  u8g2.setFontDirection(1);
  u8g2.setCursor(BildschirmBreite / 2, 2);
  u8g2.print("Text");
}
while (u8g2.nextPage());
delay(2000);
```

```
// Text 180 Grad gedreht
u8g2.firstPage();
do
{
  u8g2.setFontDirection(2);
  u8g2.setCursor(BildschirmBreite - 2, BildschirmHoehe / 2);
  u8g2.print("Text");
}
while (u8g2.nextPage());
delay(2000);

// Text 270 Grad gedreht
u8g2.firstPage();
do
{
  u8g2.setFontDirection(3);
  u8g2.setCursor(BildschirmBreite / 2,  BildschirmHoehe - 2);
  u8g2.print("Text");
}
while (u8g2.nextPage());
delay(2000);

// Kreise
u8g2.firstPage();
do
{
  for (int i = 2; i < BildschirmHoehe / 2; i += 3)
  {
    u8g2.drawCircle(BildschirmBreite / 2, BildschirmHoehe / 2, i);
  }
}
while (u8g2.nextPage());
delay(2000);

// Rahmen
u8g2.firstPage();
do
{
  for (int i = 2; i < BildschirmHoehe; i += 4)
  {
    u8g2.drawFrame(0, 0, i, i);
  }
}
while (u8g2.nextPage());
delay(2000);
```

```
// vertikale Linie
u8g2.firstPage();
do
{
  for (int i = 0; i < BildschirmBreite; i += 4)
  {
    u8g2.drawVLine(i, 0, BildschirmBreite - 1);
  }
}
while (u8g2.nextPage());
delay(2000);

// horizontale Linie
u8g2.firstPage();
do
{
  for (int i = 0; i < BildschirmHoehe; i += 4)
  {
    u8g2.drawHLine(0, i, BildschirmBreite - 1);
  }
}
while (u8g2.nextPage());
delay(2000);

// ausgefüllte Kreise
u8g2.firstPage();
do
{
  int Radius = 5;
  int StartX = 10;
  int StartY = 10;
  while (StartX < BildschirmBreite - Radius)
  {
    for (int i = StartY; i < BildschirmBreite - Radius; i += 20)
    {
      u8g2.drawDisc(StartX, i, Radius);
    }
    StartX += 10;
  }
}
while (u8g2.nextPage());
delay(2000);
```

```
  // Linien
  u8g2.firstPage();
  do
  {
    for (int i = 0; i < BildschirmBreite; i += 5)
    {
      u8g2.drawLine(0, i, 128, 64);
    }
    for (int i = BildschirmBreite; i > 0; i -= 5)
    {
      u8g2.drawLine(BildschirmBreite, i, 0, 0);
    }
  }
  while (u8g2.nextPage());
  delay(2000);
}
```

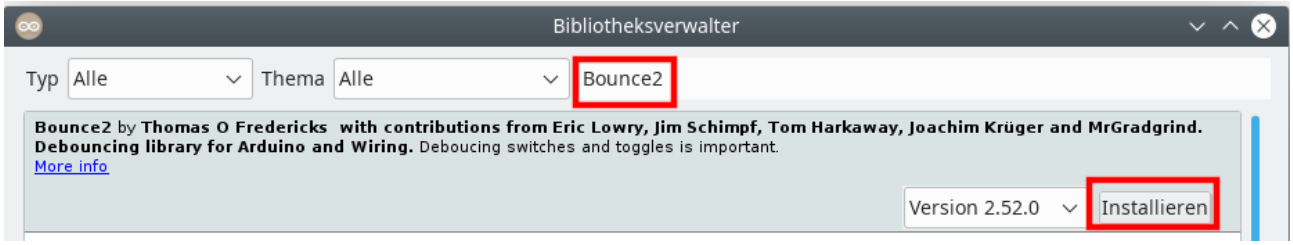
Beispiel Bildschirm drehen:

```
# include <U8g2lib.h>

// 0,96 Zoll SSD1306
U8G2_SH1106_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

// 1,3 Zoll SH1106
// U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);

int BildschirmBreite = u8g2.getDisplayWidth();
int BildschirmHoehe = u8g2.getDisplayHeight();

// Smiley
# define SmileyBreite 46
# define SmileyHoehe 45
static unsigned char Smiley[] =
{
  0x00, 0x00, 0xfe, 0x1f, 0x00, 0x00, 0x00, 0xc0, 0xff, 0xff, 0x00, 0x00,
  0x00, 0xf0, 0x07, 0xf8, 0x03, 0x00, 0x00, 0xfc, 0x00, 0xc0, 0x0f, 0x00,
  0x00, 0x3e, 0x00, 0x00, 0x1f, 0x00, 0x80, 0x0f, 0x00, 0x00, 0x7c, 0x00,
  0xc0, 0x07, 0x00, 0x00, 0xf8, 0x00, 0xe0, 0x01, 0x00, 0x00, 0xe0, 0x01,
  0xf0, 0x00, 0x00, 0x00, 0xc0, 0x03, 0x70, 0x00, 0x00, 0x00, 0x80, 0x03,
  0x38, 0x7e, 0x00, 0x80, 0x1f, 0x07, 0x38, 0xff, 0x00, 0xc0, 0x3f, 0x07,
  0x9c, 0xff, 0x01, 0xc0, 0x3f, 0x0e, 0x9c, 0xe7, 0x01, 0xc0, 0x39, 0x0e,
  0x8e, 0xc3, 0x01, 0xc0, 0x30, 0x1c, 0x8e, 0xe3, 0x01, 0xc0, 0x31, 0x1c,
  0x86, 0xf7, 0x01, 0xc0, 0x3b, 0x18, 0x87, 0xff, 0x01, 0xc0, 0x3f, 0x38,
  0x07, 0xff, 0x00, 0x80, 0x3f, 0x38, 0x03, 0x7e, 0x00, 0x80, 0x1f, 0x30,
  0x03, 0x00, 0x00, 0x00, 0x00, 0x30, 0x03, 0x00, 0x00, 0x00, 0x00, 0x30,
  0x03, 0x00, 0x00, 0x00, 0x00, 0x30, 0x03, 0x00, 0x00, 0x00, 0x00, 0x30,
  0x03, 0x00, 0x00, 0x00, 0x00, 0x30, 0x03, 0x00, 0x00, 0x00, 0x00, 0x30,
  0x07, 0x00, 0x00, 0x00, 0x00, 0x38, 0x07, 0x20, 0x00, 0x00, 0x01, 0x38,
```

```
  0x06, 0x70, 0x00, 0x80, 0x03, 0x18, 0x0e, 0xf0, 0x00, 0xc0, 0x01, 0x1c,
  0x0e, 0xe0, 0x03, 0xf0, 0x01, 0x1c, 0x1c, 0xc0, 0x3f, 0xfc, 0x00, 0x0e,
  0x1c, 0x80, 0xff, 0x7f, 0x00, 0x0e, 0x38, 0x00, 0xfc, 0x1f, 0x00, 0x07,
  0x38, 0x00, 0xc0, 0x03, 0x00, 0x07, 0x70, 0x00, 0x00, 0x00, 0x80, 0x03,
  0xf0, 0x00, 0x00, 0x00, 0xc0, 0x03, 0xe0, 0x01, 0x00, 0x00, 0xe0, 0x01,
  0xc0, 0x07, 0x00, 0x00, 0xf8, 0x00, 0x80, 0x0f, 0x00, 0x00, 0x7c, 0x00,
  0x00, 0x3e, 0x00, 0x00, 0x1f, 0x00, 0x00, 0xfc, 0x00, 0xc0, 0x0f, 0x00,
  0x00, 0xf0, 0x07, 0xf8, 0x03, 0x00, 0x00, 0xc0, 0xff, 0xff, 0x00, 0x00,
  0x00, 0x00, 0xfe, 0x1f, 0x00, 0x00
};

void setup()
{
  // Display starten
  u8g2.begin();

  // Farbe weiß
  u8g2.setDrawColor(1);
}

void loop()
{
  // Position 0 Grad
  u8g2.clearDisplay();
  u8g2.setDisplayRotation(U8G2_R0);
  u8g2.firstPage();
  do
  {
    u8g2.drawXBM(40, 10, SmileyBreite, SmileyHoehe, Smiley);
  }
  while (u8g2.nextPage());
  delay(2000);

  // Position 90 Grad
  u8g2.clearDisplay();
  u8g2.setDisplayRotation(U8G2_R1);
  u8g2.firstPage();
  do
  {
    u8g2.drawXBM(10, 30, SmileyBreite, SmileyHoehe, Smiley);
  }
  while (u8g2.nextPage());
  delay(2000);

  // Position 180 Grad
  u8g2.clearDisplay();
  u8g2.setDisplayRotation(U8G2_R2);
```

```
  u8g2.firstPage();
  do
  {
    u8g2.drawXBM(40, 10, SmileyBreite, SmileyHoehe, Smiley);
  }
  while (u8g2.nextPage());
  delay(2000);

  // Position 270 Grad
  u8g2.clearDisplay();
  u8g2.setDisplayRotation(U8G2_R3);
  u8g2.firstPage();
  do
  {
    u8g2.drawXBM(10, 30, SmileyBreite, SmileyHoehe, Smiley);
  }
  while (u8g2.nextPage());
  delay(2000);
}
```

**Zusätzlich benötigte Bibliothek:**
**Sketch → Bibliothek einbinden → Bibliotheken verwalten**




**Das eigentliche Programm:**
Binde die benötigten Bibliotheken ein und definiere die Variablen. Beachte die Kommentare.

```
# include <U8g2lib.h>
# include <Bounce2.h>

int Minimum = 1;
int Maximum = 7;
int TASTER = 7;
/*
  OLED initialisieren
  Controller: SH1106 oder SSD1306
  es wird der Page buffer mode verwendet
*/
// 1,3 Zoll SH1106
U8G2_SH1106_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

// 0,96 Zoll SSD1306
// U8G2_SSD1306_128X64_NONAME_1_HW_I2C u8g2(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);
```

```
// Bounce initialisieren
Bounce Wuerfel = Bounce();
```

Der setup-Teil. Beachte die Kommentare.

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

  // Taster Bounce zuordnen
  Wuerfel.attach(TASTER);
  Wuerfel.interval(20);

  u8g2.begin();

  // Zufallsgenerator starten
  randomSeed(A0);

  // Farbe weiß
  u8g2.setDrawColor(1);

  // Position 90 Grad
  u8g2.clearDisplay();
  u8g2.setFont(u8g2_font_t0_22_te);

  u8g2.setDisplayRotation(U8G2_R1);
  u8g2.setFlipMode(1);

  // Hinweis anzeigen
  u8g2.firstPage();
  do
  {
    u8g2.drawStr(2, 20, "Start");
    u8g2.drawStr(2, 50, "->");
    u8g2.drawStr(2, 80, "Taste");
  }
  while (u8g2.nextPage());

  u8g2.setDisplayRotation(U8G2_R0);
  u8g2.setFlipMode(1);
}
```

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

```
void Wuerfeln()
{
  int Zahl =  random(Minimum, Maximum);
  u8g2.firstPage();

  // Würfelaugen zeichnen
  // 1
  if (Zahl == 1)
  {
    do
    {
      u8g2.drawRFrame(0, 0, 128, 64, 5);
      u8g2.drawDisc(60, 32, 8);
    }
    while (u8g2.nextPage());
  }

  // 2
  if (Zahl == 2)
  {
    do
    {
      u8g2.drawRFrame(0, 0, 128, 64, 5);
      u8g2.drawDisc(14, 14, 8);
      u8g2.drawDisc(112, 50, 8);
    }
    while (u8g2.nextPage());
  }

  // 3
  if (Zahl == 3)
  {
    do
    {
      u8g2.drawRFrame(0, 0, 128, 64, 5);
      u8g2.drawDisc(14, 14, 8);
      u8g2.drawDisc(60, 32, 8);
      u8g2.drawDisc(112, 50, 8);
    }
    while (u8g2.nextPage());
  }

  // 4
  if (Zahl == 4)
  {
    do
    {
      u8g2.drawRFrame(0, 0, 128, 64, 5);
      u8g2.drawDisc(14, 14, 8);
```

```
      u8g2.drawDisc(14, 50, 8);
      u8g2.drawDisc(112, 14, 8);
      u8g2.drawDisc(112, 50, 8);
    }
    while (u8g2.nextPage());
  }

  // 5
  if (Zahl == 5)
  {
    do
    {
      u8g2.drawRFrame(0, 0, 128, 64, 5);
      u8g2.drawDisc(14, 14, 8);
      u8g2.drawDisc(60, 32, 8);
      u8g2.drawDisc(112, 14, 8);
      u8g2.drawDisc(14, 50, 8);
      u8g2.drawDisc(112, 50, 8);
    }
    while (u8g2.nextPage());
  }

  // 6
  if (Zahl == 6)
  {
    do
    {
      u8g2.drawRFrame(0, 0, 128, 64, 5);
      u8g2.drawDisc(14, 14, 8);
      u8g2.drawDisc(60, 14, 8);
      u8g2.drawDisc(112, 14, 8);
      u8g2.drawDisc(14, 50, 8);
      u8g2.drawDisc(60, 50, 8);
      u8g2.drawDisc(112, 50, 8);
    }
    while (u8g2.nextPage());
  }
}

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

Der loop-Teil:

```
void loop()
{
  if (Wuerfel.update())
  {
    if (Wuerfel.read() == LOW)
    {
      // Würfeleffekt: Zufallszahlen in schneller Folge anzeigen
      // bedingt durch den Page buffer mode nicht sehr schnell
      for (int i = 0; i < 5; i++)
      {
        int Zahl =  random(Minimum, Maximum);

        Wuerfeln();
        delay(50);
      }
    }
  }
}
```