Auf dem OLED-Display wird eine Stoppuhr als Startsymbol angezeigt, ein Druck auf den Taster startet den Countdown. Eine Zielflagge zeigt das Ende des Countdowns an.

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.

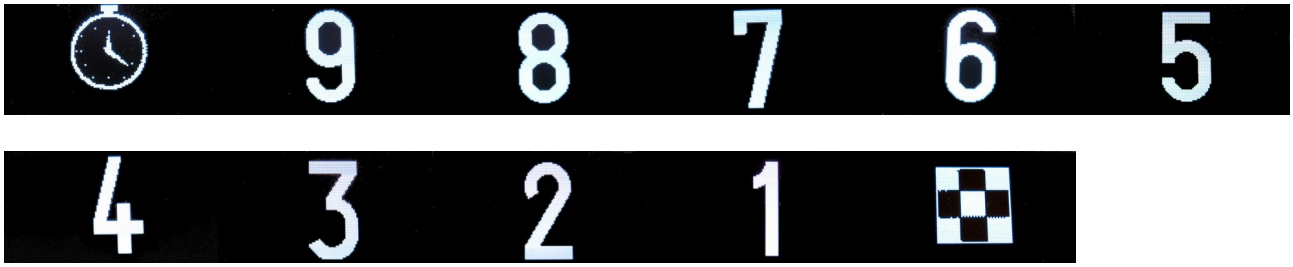
Basis der Technik ist die Entdeckung der Elektrolumineszenz: ein Festkörper wird durch Anlegen einer elektrischen Spannung dazu angeregt Licht zu erzeugen.

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

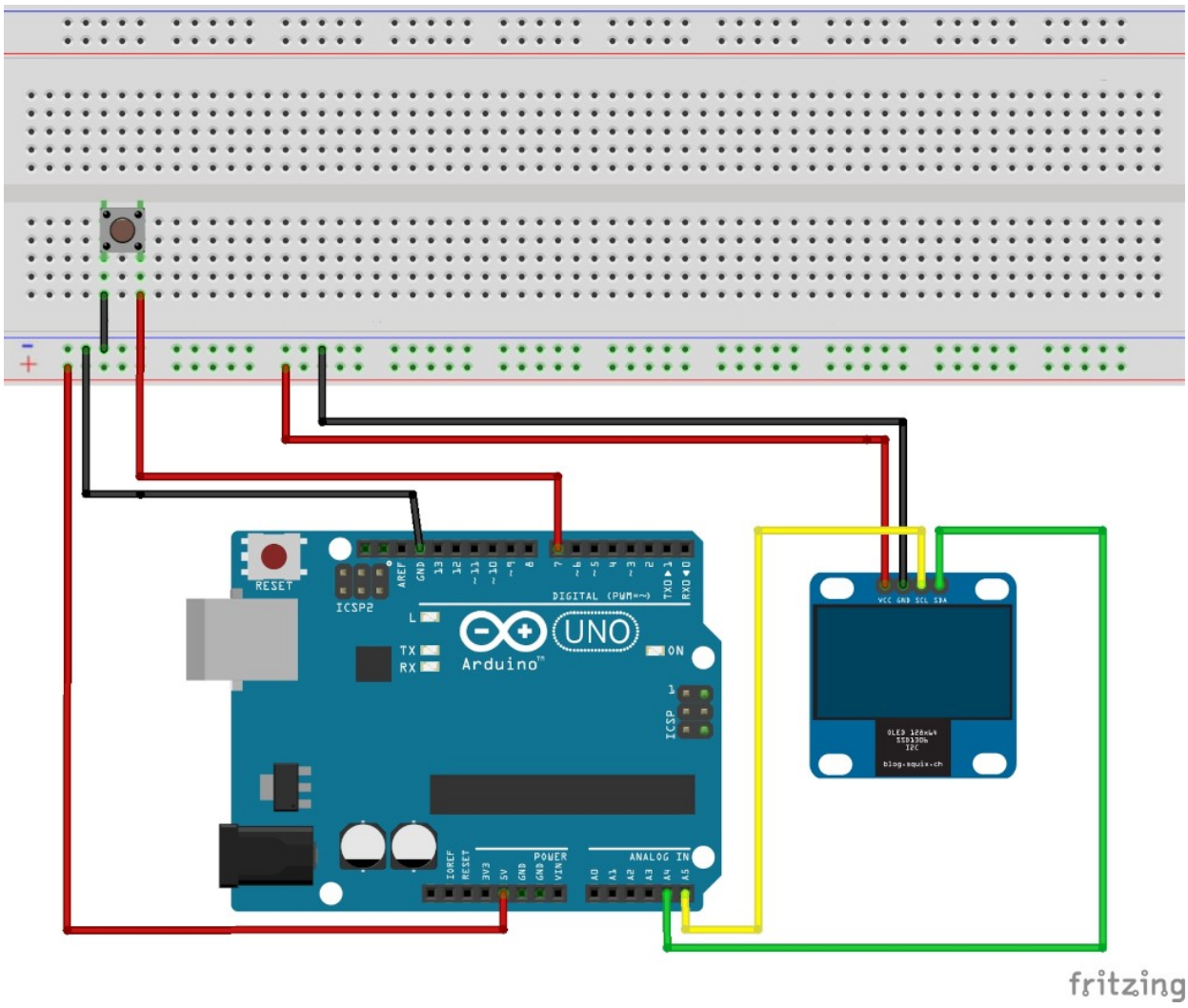
Auf dem OLED-Display wird eine Stoppuhr als Startsymbol angezeigt, ein Druck auf den Taster startet den Countdown. Eine Zielflagge zeigt das Ende des Countdowns an



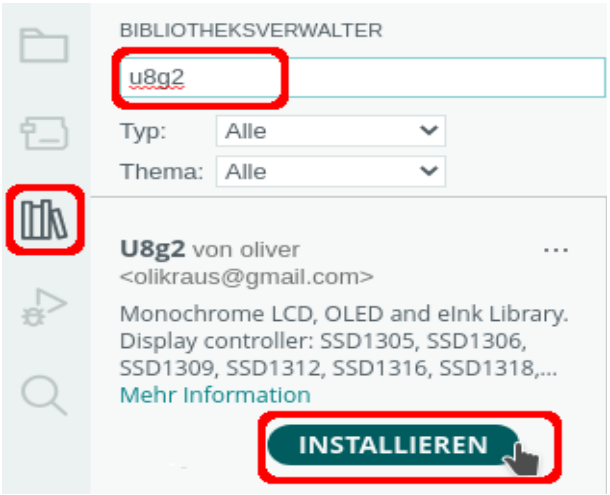
Benötigte Bauteile:

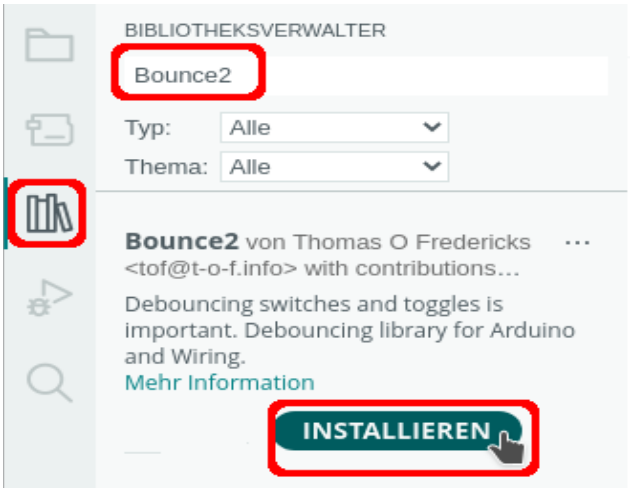
- Taster
- OLED-Display 1,3 Zoll/0,96 Zoll
- Leitungsdrähte

Baue die Schaltung auf.



Benötigte Bibliotheken:





Ansteuerung

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 mit dem Hinweis Speicherplatz- und Stabilitätsprobleme beim UNO R3

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();
```

Schlüsselwort	Parameter	Aktion
begin();		OLED-Display starten
getDisplayWidth();		Bildschirmbreite feststellen
getDisplayHeight();		Bildschirmhöhe feststellen
setdrawColor(Parameter)	0 → schwarz 1 → weiß	Zeichenfarbe festlegen
clearDisplay();		Bildschirm dunkel schalten
setContrast(Parameter)	0 … 255	Kontrast einstellen
setDisplayRotation(U8G2_*);	U8G2_0 → 0 Grad U8G2_1 → 90 Grad U8G2_2 → 180 Grad U8G2_3 → 270 Grad	Anzeige drehen
flipMode(Parameter);	0 → normale Ausrichtung 1 → 180 Grad drehen wirksam erst bei einer Rotation	Anzeige spiegeln (180 Grad)
home();		Cursor in die linke obere Ecke setzen
drawPixel(x-Achse, y-Achse)		einzelnen Pixel zeichnen
drawLine(StartX, StartX, EndeX, EndeY);		Linie zeichnen
drawHLine(StartX, StartY, Länge);		horizontale Linie zeichnen
drawVLine(StartX, StartY, Länge);		vertikale Linie zeichnen
drawFrame(StartX, StartY,, Breite, Höhe);		Rechteck zeichnen
drawRFrame(StartX, StartY, Breite, Höhe, Eckenradius);		abgerundetes Rechteck zeichnen
drawBox(StartX, StartY, Breite, Höhe);		ausgefülltes Rechteck zeichnen
drawRBox(StartX, StartY, Breite, Höhe, Eckenradius);		abgerundetes ausgefülltes Rechteck zeichnen
drawCircle(MittelpunkX, MittelpunktY, Radius);		Kreis zeichnen
drawDisc(MittelpunkX, MittelpunktY, Radius);		ausgefüllten Kreis zeichnen
drawXBM(StartX, StartY, Breite, Höhe, Array_Bilddatei);		XBM-Bild anzeigen
drawEllipse(StartX, StartY, RadiusX, RadiusY);		Ellipse zeichnen
print("Text"); drawStr(StartX, StartY ,"Text");		Text schreiben
setFontDirection(Wert);	0 → normal ausgerichtet 1 → 90 ° gedreht 2 → 180 ° gedreht 3 → 270 ° gedreht	Schreibrichtung
setCursor(x-Achse, y-Achse);		Cursor setzen

Schlüsselwort	Parameter	Aktion
u8g2.setFont(Schriftart)	**Beispiele für funktionierende Schriftarten:** 6px: u8g2_font_5x7_tr 7px: u8g2_font_torussansbold8_8r 8px: u8g2_font_ncenB08_tr 10px: u8g2_font_t0_15b_me 12px: u8g2_font_helvB12_tf 13px: u8g2_font_t0_22_te 14px: u8g2_font_helvB14_tf 17px: u8g2_font_timB18_tf 18px: u8g2_font_lubB18_tr 20px: u8g2_font_courB24_tf 23px: u8g2_font_timB24_tf 25px: u8g2_font_helvR24_tf 32px: u8g2_font_logisoso32_tf 42px: u8g2_font_fub42_tf 58px: u8g2_font_logisoso58_tf 62px: u8g2_font_logisoso62_tn	Schriftart bestimmen

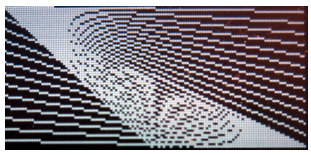
weitere Schriftarten: https://github.com/olikraus/u8g2/wiki/fntlistall

Du musst ausprobieren, welche Schriftarten dargestellt werden können!

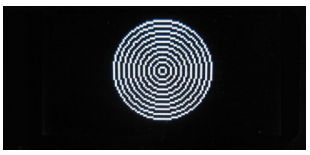
Beispiele für grafische Funktionen:

So sieht es aus:

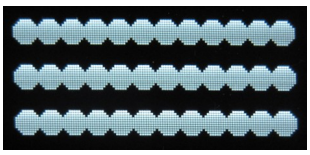
drawXBM()



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);
}
```

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

```
# include <U8g2lib.h>
# include <Bounce2.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();

int TASTER = 7;
Bounce StoppUhr = Bounce();
```

Die xbm-Bilder:

```
// Bild Stoppuhr
# define StoppuhrBreite 50
# define StoppuhrHoehe 56
static unsigned char Stoppuhr[] =
{
  0x00, 0x00, 0xf0, 0x7f, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x7f, 0x00,
  0x00, 0x00, 0x00, 0x00, 0xf0, 0x7f, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf0,
  0x7f, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc0, 0x0f, 0x00, 0x00, 0x00, 0x00,
  0x00, 0xc0, 0x0f, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0xff, 0x00, 0x00,
  0x00, 0x00, 0x00, 0xff, 0xff, 0x07, 0x00, 0x00, 0x00, 0xe0, 0x0f, 0x80,
  0x1f, 0x00, 0x00, 0x00, 0xf0, 0x01, 0x00, 0x7e, 0x00, 0x00, 0x00, 0x78,
  0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x1e, 0x00, 0x00, 0xe0, 0x01, 0x00,
  0x00, 0x0f, 0x00, 0x00, 0x80, 0x03, 0x00, 0x00, 0x07, 0x00, 0x00, 0x04,
  0x07, 0x00, 0xc0, 0x03, 0x00, 0x00, 0x00, 0x0e, 0x00, 0xc0, 0x01, 0x01,
  0x00, 0x00, 0x1c, 0x00, 0xe0, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x70,
  0x00, 0x00, 0x02, 0x00, 0x38, 0x00, 0x30, 0x00, 0x00, 0x02, 0x00, 0x70,
  0x00, 0x38, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x00, 0x98, 0x00, 0x00, 0x02,
  0x00, 0xe2, 0x00, 0x1c, 0x00, 0x00, 0x03, 0x00, 0xc0, 0x00, 0x1c, 0x00,
  0x00, 0x02, 0x00, 0xc0, 0x00, 0x0c, 0x00, 0x00, 0x03, 0x00, 0xc0, 0x01,
  0x0e, 0x00, 0x00, 0x03, 0x00, 0x80, 0x01, 0x0e, 0x00, 0x00, 0x07, 0x00,
  0x80, 0x01, 0x06, 0x00, 0x00, 0x07, 0x00, 0x80, 0x03, 0x06, 0x00, 0x00,
  0x07, 0x00, 0x80, 0x01, 0x06, 0x00, 0xc0, 0x07, 0x00, 0x00, 0x03, 0x06,
  0x00, 0x80, 0x07, 0x00, 0x80, 0x01, 0x96, 0x00, 0x80, 0x0f, 0x00, 0x80,
  0x03, 0x06, 0x00, 0x00, 0x1f, 0x00, 0x80, 0x01, 0x06, 0x00, 0x00, 0x3e,
  0x00, 0x00, 0x03, 0x06, 0x00, 0x00, 0xf8, 0x00, 0x80, 0x03, 0x06, 0x00,
  0x00, 0xf0, 0x01, 0x80, 0x01, 0x0e, 0x00, 0x00, 0x80, 0x01, 0x80, 0x01,
  0x0e, 0x00, 0x00, 0x00, 0x07, 0xc0, 0x01, 0x0c, 0x00, 0x00, 0x00, 0x0e,
  0x80, 0x01, 0x0c, 0x00, 0x00, 0x00, 0x18, 0xc1, 0x00, 0x1c, 0x04, 0x00,
  0x00, 0x20, 0xe0, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00, 0xe0, 0x00, 0x38,
  0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x70,
  0x00, 0x70, 0x00, 0x00, 0x00, 0x00, 0x38, 0x00, 0xe0, 0x00, 0x00, 0x00,
  0x00, 0x3c, 0x00, 0xc0, 0x01, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x80, 0x01,
  0x00, 0x00, 0x00, 0x0f, 0x00, 0x80, 0x87, 0x00, 0x02, 0x08, 0x07, 0x00,
  0x00, 0x1f, 0x00, 0x00, 0xc0, 0x03, 0x00, 0x00, 0x3c, 0x00, 0x00, 0xf0,
  0x01, 0x00, 0x00, 0x78, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0xf0, 0x01,
  0x00, 0x3e, 0x00, 0x00, 0x00, 0xc0, 0x7f, 0xe0, 0x0f, 0x00, 0x00, 0x00,
  0x00, 0xff, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0xd0, 0x7f, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
```

```
// Bild Zielflagge
# define ZielBreite 50
# define ZielHoehe 53
static unsigned char Zielflagge[] =
{
  0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0x03, 0xfe, 0xff, 0x01, 0x00, 0xfe,
  0xff, 0x03, 0xfe, 0x7f, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00,
  0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe,
  0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff,
  0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00,
  0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff,
  0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03,
  0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc,
  0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00,
  0x00, 0xfc, 0xff, 0x03, 0x8a, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x02,
  0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00,
  0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff,
  0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00,
  0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02,
  0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01,
  0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe,
  0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02,
  0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00,
  0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff,
  0x01, 0x00, 0x02, 0x02, 0x00, 0xfe, 0xff, 0x01, 0x00, 0x02, 0x02, 0x00,
  0xaa, 0xaa, 0x00, 0x00, 0x02, 0x06, 0x44, 0x00, 0x00, 0x54, 0xd5, 0x03,
  0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc,
  0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00,
  0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe,
  0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff,
  0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00,
  0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff,
  0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03,
  0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc,
  0xff, 0x03, 0xfe, 0xff, 0x00, 0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0x01,
  0x00, 0xfc, 0xff, 0x03, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0x03
};
```

Der setup-Teil. Beachte die Kommentare.

```
void setup()
{
  StoppUhr.attach(TASTER);

  // Display starten
  u8g2.begin();

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

  pinMode(TASTER, INPUT_PULLUP);

  // Stoppuhr anzeigen
  u8g2.firstPage();
  do
  {
    u8g2.drawXBM(40, 1, StoppuhrBreite, StoppuhrHoehe, Stoppuhr);
  }
  while (u8g2.nextPage());
}
```

Der loop-Teil. Beachte die Kommentare.

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

  // Beginn des Countdowns
  int Nummer = 9;

  int TasterLesen = digitalRead(TASTER);

  // Taster gedrückt
  if (StoppUhr.update())
  {
    if (StoppUhr.read() == LOW)
    {
      u8g2.setFont(u8g2_font_logisoso62_tn);
      while (Nummer > 0)
      {
        u8g2.firstPage();
        do
        {
          u8g2.setFontDirection(0);
          u8g2.setCursor(50, BildschirmHoehe);
          u8g2.print(Nummer);
        }
```

```
        while (u8g2.nextPage());

        delay(1000);
        Nummer --;
      }

      // Ziel anzeigen
      u8g2.firstPage();
      do
      {
        u8g2.drawXBM(40, 5, ZielBreite, ZielHoehe, Zielflagge);
      }
      while (u8g2.nextPage());
    }
  }
}
```