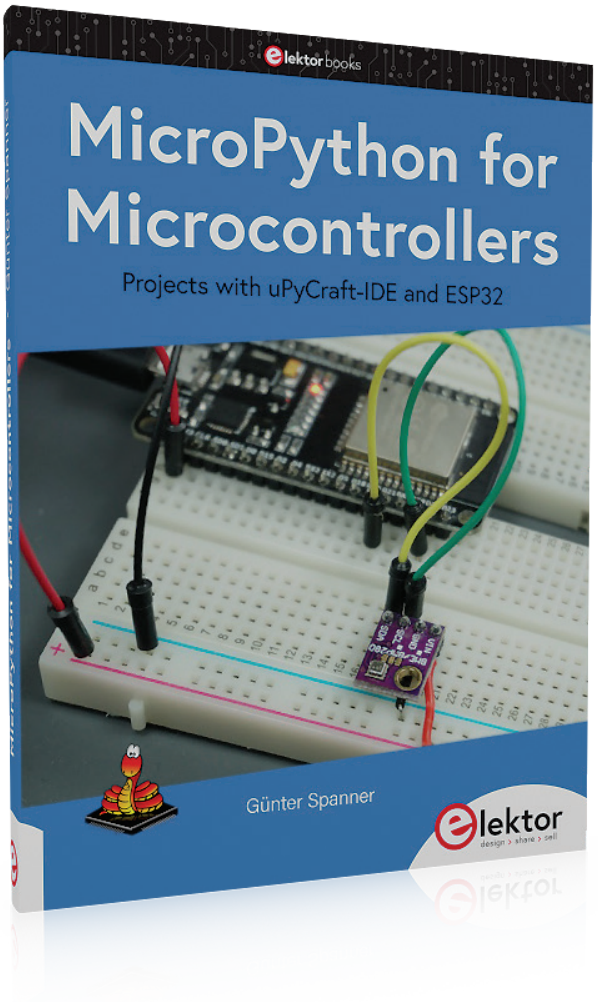
MicroPython
voor microcontrollers
technieken voor kleine displays



Dr.Günter Spanner (Duitsland)

Dit is een deel van een hoofdstuk uit het boek MicroPython for Microcontrollers (Elektor 2021) van Günter Spanner. Het materiaal dat hier is afgedrukt heeft betrekking op de geslaagde combinatie van het populaire ESP32-microcontrollerboard met een SSD1306 OLED display onder gebruikmaking van de programmeertaal MicroPython. Ontdek de kracht en het gebruiksgemak van MicroPython, ga programmeren en maak prototypes van een ECG-display en een digitale klok.

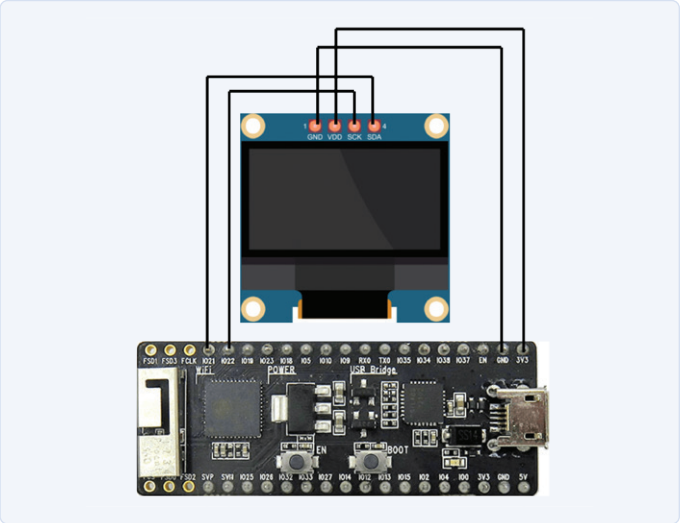
Als het om tests en eenvoudige toepassingen gaat, voldoet de console volledig voor het weergeven van tekst en getallen. Deze methode heeft echter het grote nadeel dat er altijd een PC of in ieder geval een laptop bij nodig is. Bij permanent gebruik gedurende dagen of weken veroorzaken deze apparaten onnodig elektriciteitsverbruik. Bovendien is het niet altijd wenselijk of praktisch om voor elke µC-applicatie een extra computer te gebruiken.
Als het alleen een kwestie is van het weergeven van een tijd of de meetwaarden van klimaatsensoren, dan is het veel beter om een apart klein display op de controller aan te sluiten. Een veelgebruikt display is de SSD1306. Dit meet slechts 0,96 inch (2,4 cm) en biedt een resolutie van 128 × 64 pixels.
MicroPython wordt standaard geleverd met een stuurprogramma voor dit displaytype. Dit stuurprogramma is al geladen wanneer de bestanden worden overgebracht naar de ESP32. Het stuurprogramma maakt het mogelijk tekst en numerieke gegevens weer te geven, alsmede eenvoudige graphics. Het SSD1306-display heeft een interne RAM en een ingebouwde oscillator. Dit betekent dat het geen externe componenten nodig heeft. Bovendien beschikt het over een helderheidsregeling met 256 instelbare niveaus.

De belangrijkste kenmerken van het SSD1306-display zijn:

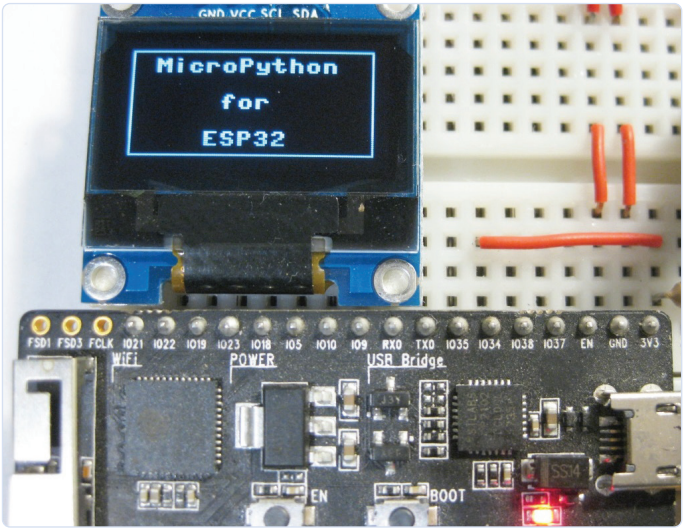
- resolutie 128 × 64 dot matrix;
- voedingsspanning 1,65...3,3 V;
- temperatuurbereik –40 °C tot +85 °C;
- geïntegreerde 128 x 64-bit SRAM displaybuffer;
- continue scroll-functie, zowel horizontaal als verticaal;
- on-chip oscillator.

OLED-displays hebben geen achtergrondverlichting nodig omdat elke pixel licht kan uitstralen. Daarom is deze variant ook bij ongunstige lichtomstandigheden goed afleesbaar. Bovendien is het contrast aanzienlijk hoger in vergelijking met LCD's. Omdat een pixel alleen energie verbruikt als hij daadwerkelijk oplicht, zijn OLED-schermen zeer energiezuinig.

De eenvoudigste versies van SSD1306-boards hebben slechts vier aansluitpinnen. Dat is echter genoeg om het display via de I²C-bus te besturen. Andere varianten hebben reset-pinnen of een extra SPI-interface. Voor de meeste toepassingen is de eenvoudige versie



Figuur 1. SSD1306-display aangesloten op het ESP32-microcontrollerboard.



Figuur 2. SSD1306-display met tekstuele en grafische uitvoer.

echter voldoende. De onderstaande tabel toont alle benodigde aansluitingen.

OLED-pin	ESP32
VDD	3V3
GND	GND
SCK	GPIO 22
SDA	GPIO 21

Het bijbehorende bedradingsschema is afgebeeld in **figuur 1**. Het volgende script geeft een tekstbericht en een eenvoudig grafisch element in de vorm van een kader op het display weer:

```
# SSD1306_DEMO.py

from machine import Pin, I2C
from ssd1306 import SSD1306_I2C

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))
# I2C Pin assignment
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width, oled_height, i2c)
lin_hight = 9
col_width = 8
def text_write(text,lin, col):
oled.text(text,col*col_width,lin*lin_hight)

oled.fill(0)
text_write("MicroPython", 1, 2)
text_write("for", 3, 6)
text_write("ESP32", 5, 5)

oled.rect(5, 5, 116, 52, 1)
oled.show()
```

Figuur 2 toont de resulterende uitvoer. Om het display aan te sturen, moeten vervolgens de benodigde modules worden geïmporteerd. Zoals al opgemerkt zijn de bibliotheken *machine* en *SSD1306* normaal gesproken al beschikbaar als standaardbibliotheken. Indien nodig kan echter een *ssd1306.py*-bestand afzonderlijk in het board worden geladen. De pinspecificatie voor de I²C-bus verloopt via:

```
i2c = I2C (-1, scl = Pin (22), sda = Pin (21))
```

Het aantal beschikbare pixels in de aangesloten module wordt vastgelegd met behulp van deze variabelen:

```
oled_width = 128
oled_height = 64
```

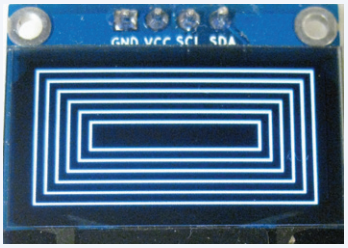
De parameter '–1' geeft aan dat de gebruikte module geen reset- of interruptpinnen heeft. Met deze informatie kan een `SSD1306_I2C` object met de naam `oled` worden gemaakt. Hier worden de eerder gedefinieerde gegevens gekopieerd:

```
oled = ssd1306.SSD1306_I2C (oled_width, oled_height,
   i2c)
```

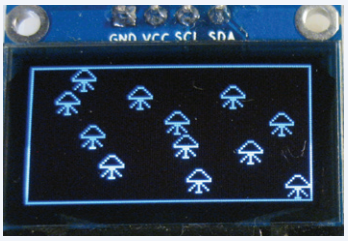
Het display is nu klaar voor gebruik. De functie `text()` wordt gebruikt om informatie weer te geven op het display. Met de methode `show()` wordt het display bijgewerkt. De functie `text()` accepteert de volgende argumenten:

- bericht (type String);
- X- en Y-positie van het tekstveld in pixel-eenheden;
- optionele tekstkleur: 0 = zwart (donker) en 1 = wit (licht).

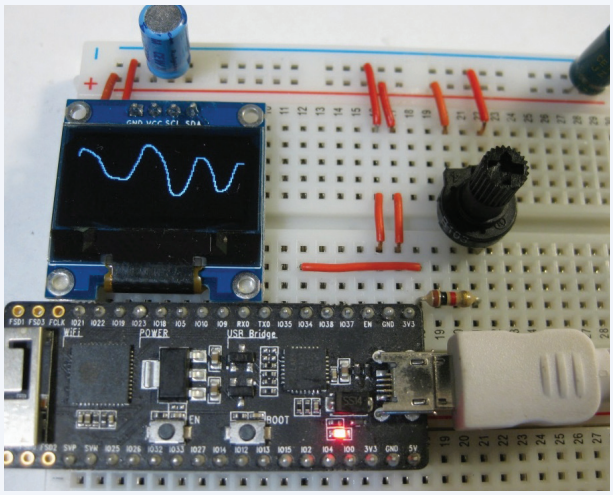
De volgende instructie geeft een bericht in witte of blauwe kleur op een donkere achtergrond uit. De tekst begint op positie x = 0 en y = 0:`oled.text ('MicroPython!', 0, 0)`



Figuur 3. Concentrische frames op het OLED-display getekend.



Figuur 4. Grafische elementen op het OLED-display.



Figuur 5. Gegevensuitvoer op het OLED-display.

Listing 1: SSD1306 Bitmap Demo.

```
# SSD1306_bitmap_DEMO.py

from machine import Pin, I2C
import ssd1306
import urandom

i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height,
   i2c)

# frame
oled.hline(0, 0, oled_width-1, 1)
oled.hline(0, oled_height-1, oled_width-1, 1)
oled.vline(0, 0, oled_height-1, 1)
oled.vline(oled_width-1, 0, oled_height, 1)
oled.show()
```

```
ICON = [
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0],
    [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
    [0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0],
    [0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0],
    [0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0],
]
for n in range(12):
  xofs = urandom.randint(1, oled_width-12)
  yofs = urandom.randint(1, oled_height-12)
  for y, row in enumerate(ICON):
    for x, c in enumerate(row):
      oled.pixel(x+xofs, y+yofs, c)
oled.show()
```

Listing 2: Rolling ECG Display.

```
# Rolling_ECG_display.py
from machine import Pin, ADC, I2C from time import
   sleep
import ssd1306

i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
  oled_width = 128
  oled_height = 64
  oled = ssd1306.SSD1306_I2C(oled_width, oled_
   height, i2c)

  pot = ADC(Pin(34))
  pot.atten(ADC.ATTN_11DB) #Full range: 3.3v vMax =
   3.4
  dotPos_old = int(oled_height/2)
while True:
  pot_value = pot.read()
  voltage = 0.000816*pot_value + 0.037822 #
   print(voltage)
dotPos_new = int(voltage/vMax*oled_height)

oled.line(0, dotPos_new, 0, dotPos_old, 1)
oled.scroll(1, 0)
oled.line(0, dotPos_new, 0, dotPos_old, 0)

dotPos_old = dotPos_new oled.pixel(0, int(oled_
   height/2), 1)
oled.show()
```

De `show()` methode maakt de wijzigingen zichtbaar op het display. De bibliotheek bevat ook andere handige methoden. De functie `fill (1)` creëert een volledig wit scherm, dat wil zeggen alle pixels zijn verlicht. Met `oled.fill (0)` worden alle pixels zwart of donker gemaakt.
De `pixel()` methode maakt grafische weergaven mogelijk. Hij accepteert de volgende argumenten:

- X-coördinaat: horizontale pixelpositie;
- Y-coördinaat: verticale pixelpositie;
- pixelkleur: 0 = zwart, 1 = wit.

Zo kan een enkele witte pixel in de linkerbovenhoek worden gemaakt:

```
oled.pixel (0, 0, 1)
```

Met behulp van `oled.invert (True)` worden de OLED-kleuren geïnverteerd. Wit wordt zwart en omgekeerd. Om terug te keren naar de originele kleuren, kunt u `oled.invert (False)` gebruiken.

Grafische weergave

Naast de pixelinstructies zijn er ook andere grafische commando's beschikbaar. Met behulp van `.hline ()` of `.vline ()` kunnen horizontale en verticale lijnen worden getekend. De XY-startpositie evenals de lijnlengte en kleur worden gespecificeerd.

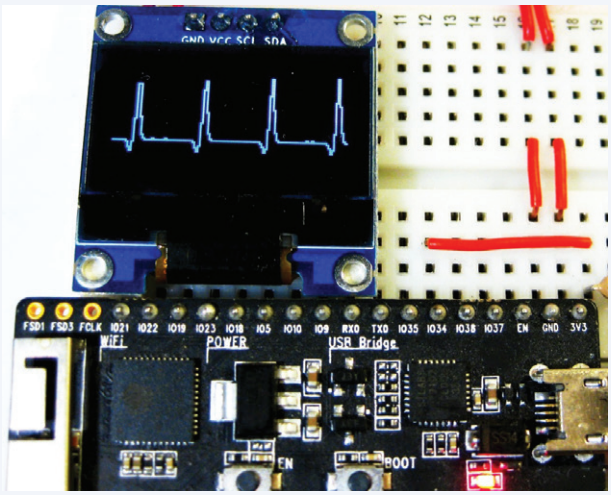
Het volgende kleine programma tekent bijvoorbeeld concentrische rechthoekige kaders op het display. Het resultaat kan worden bewonderd in **figuur 3**.

```
# SSD1306_frames.py
from machine import Pin, I2C
import ssd1306

i2c = I2C(-1,scl=Pin(22),sda=Pin(21))
oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height,
   i2c)

for n in [0,5,10,15,20,25]:
oled.hline(n, n, oled_width-1-2*n, 1-2*n)
oled.hline(n, oled_height-1-n, oled_width-1-2*n,
   1-2*n)
oled.vline(n, n, oled_height-1-2*n, 1-2*n)
oled.vline(oled_width-1-n, n, oled_height-2*n, 1-2*n)
oled.show()
```

Diagonalen worden getekend met `.line (x1, y1, x2, y2, c)` tussen de twee punten (x1, y1) en (x2, y2). De parameter `c` bepaalt de kleur van de getekende lijn. Voor eenvoudige grafische afbeeldingen kunnen bitmaps pixel voor pixel in de weergavebuffer worden geschreven. Het programma in **listing 1** toont een voorbeeld, resulterend in de weergave van **figuur 4**.

OLED-display als plotter

Naast bitmaps kunnen ook meetwaarden grafisch worden weergegeven op het OLED-display. Dit betekent dat u niet langer afhankelijk bent van de plotterfunctie in Thonny (de IDE die voor MicroPython wordt gebruikt) maar ook stand-alone apparaten kunt gebruiken om grafische uitvoer weer te geven. De software in **listing 2** toont een doorlopende weergave zoals bekend van een ECG-monitor in een ziekenhuis.
Op de ADC-ingang 34 kan een potentiometer worden aangesloten voor het registreren van de meetwaarde. Na het starten van het programma worden de spanningswaarden continu op het display weergegeven. Dit werkt dankzij de ingebouwde scroll-functie. Met behulp van de instructie



Figuur 6. Elektrocardiogram op het display.

```
oled.scroll(1, 0)
```

wordt de volledige scherminhoud één pixel verplaatst. Als u geen losse pixels maar een doorlopende curve wilt opnemen, moet u de punten met lijnen verbinden. Hiervoor zijn twee variabelen nodig:

```
dotPos_old and dotPos_new
```

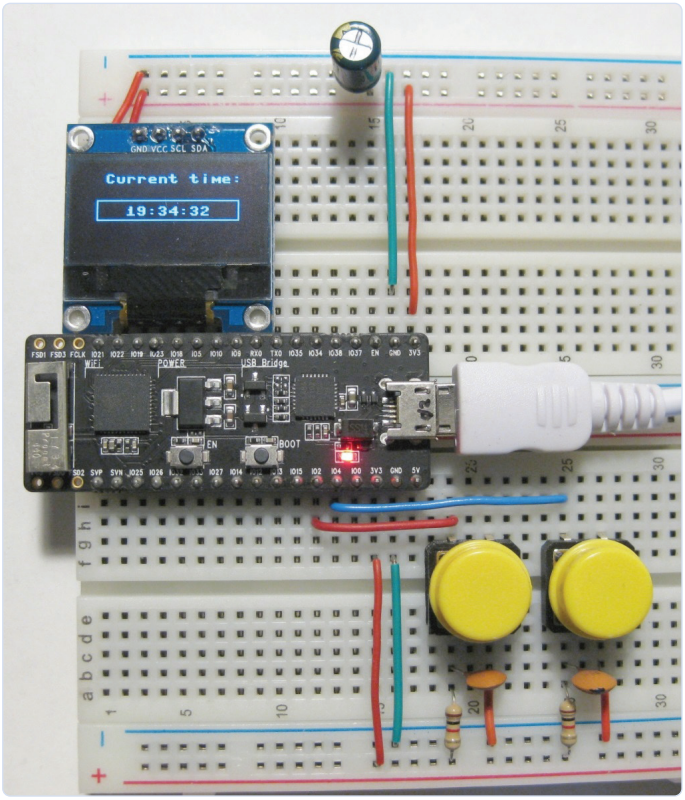
Hiermee wordt een lijn getekend tussen de huidige en de laatst gemeten waarde. Vervolgens wordt de scherminhoud met één pixel verschoven. Ten slotte wordt de nieuwe positie overgebracht naar het buffergeheugen:

```
dotPos_old = dotPos_new
```

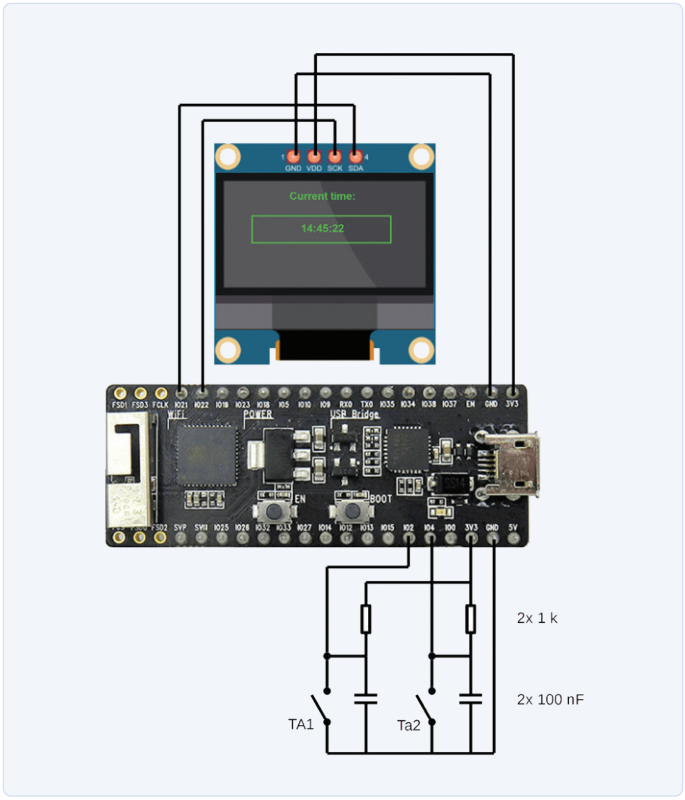
en het spel begint opnieuw. **Figuur 5** toont een voorbeeld van continu veranderende potentiometerwaarden. Als u een ECG-versterker hebt, kunt u op deze manier ook de elektrische signalen van het menselijk hart opnemen. Het resultaat is het typische elektrocardiogram dat we kennen van de intensive care-afdelingen in ziekenhuizen (**figuur 6**).

Digitale klok met OLED-display

Een andere nuttige toepassing is het tonen van de tijd. Dit verandert de ESP32 samen met de SSD1307 in een praktische digitale klok. De functie `time()` van de tijdmodule geeft het aantal seconden weer sinds het inschakelen of resetten van het board. Met behulp van een variabele



Figuur 7. Digitale klok met OLED-display.



Figuur 8: Schema van de in MicroPython geprogrammeerde digitale klok.

Listing 3. ESP32/SSD1306-klok in MicroPython.

```
# setable_clock.py

from machine import Pin,I2C
import time
from ssd1306 import SSD1306_I2C

i2c = I2C(-1,scl=Pin(22),sda=Pin(21))
oled_width=128
oled_height=64
oled = SSD1306_I2C(oled_width,oled_height,i2c)

time_offset=20*3600+00*60+0 # hh+mm+ss
lin_hight=5
col_width=8

def handle_interrupt_min(pin):
  global time_offset
  time_offset+=60
  time.sleep(.2)

def handle_interrupt_hr(pin):
  global time_offset
  time_offset+=3600
  time.sleep(.2)

button_min = Pin(4, Pin.IN)
button_min.irq(trigger=Pin.IRQ_RISING,
   handler=handle_interrupt_min)

button_hr = Pin(2, Pin.IN)
button_hr.irq(trigger=Pin.IRQ_RISING,
   handler=handle_interrupt_hr)

def text_write(text, lin, col=0):
  oled.text(text, col*col_width, lin*lin_hight)

def time_text(time):
  secs=time%60
  mins=(time//60)%60
  hours=(time//3600)%24
  return "{:02d}:{:02d}:{:02d}".
   format(hours,mins,secs)

def show():
  oled.fill(0)
  text_write("Current time:",1,2)
  current_text = time_text(current_time)
  text_write(current_text,6,4)
  oled.rect(10,25,108,16,1)
  oled.show()

while True:
  current_time=time_offset+time.time()
  show()
```

```
time_offset=20*3600+00*60+0 # hh+mm+ss
```

kan een starttijd kan worden gespecificeerd. In dit geval begint de klok om 20:00:00. De routine `time_text()`:

```
def time_text(time):
secs=time%60
mins=(time//60)%60
hours=(time//3600)%24
return "{:02d}:{:02d}:{:02d}".format(hours,mins,secs)
```

zet de opeenvolgende seconden om in uren, minuten en seconden, dus in de gebruikelijke hh:mm:ss-weergave, zoals in **figuur 7**. De klok kan worden ingesteld met behulp van twee drukknoppen (Ta1 en Ta2) op poorten 02 en 04, aangesloten zoals in **figuur 8**. Over de knoppen zijn 100nF-condensatoren aangesloten ter ontdendering. De 1kΩ-weerstanden dienen als pull-ups. De bijbehorende interruptroutines

```
def handle_interrupt_min(pin):
global time_offset
time_offset+=60
time.sleep(.2)
```

en

```
def handle_interrupt_hr(pin):
global time_offset
time_offset+=3600
time.sleep(.2)
```

zorgen ervoor dat bij elke druk op de toets de minuten (60 seconden) of uren (3600 seconden) met één worden opgehoogd. Het volledige programma voor de digitale klok is in **listing 3** afgedrukt.

200581-03

Opmerking van de redactie: een Engelstalige versie van het boek is beschikbaar op www.elektor.nl/micropython-for-microcontrollers. Het softwarepakket bij het boek is gratis beschikbaar op de projectpagina bij dit artikel [1].

Vragen of opmerkingen?
Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur of naar de redactie van Elektor, via redactie@elektor.com

Een bijdrage van
Tekst en illustraties:
dr. Günter Spanner

Redactie: **Jan Buiting**
Vertaling: **Hans Adams**
Layout: **Giel Dols**

WEBLINK

[1] **Software-archief bij het boek MicroPython for Microcontrollers**: www.elektormagazine.nl/200581-03