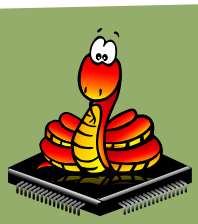


MicroPython
voor de ESP32 en consorten
deel 1: installatie en onze eerste programma's

Dr. Günter Spanner (Duitsland)

Python komt meer en meer in de plaats van C als programmeertaal en deze trend is nu ook te zien in de microcontrollerwereld. Met de ESP32 als voorbeeld bekijken we in dit artikel hoe u een moderne microcontroller kunt programmeren in MicroPython.

Benodigde onderdelen

1. Klein breadboard *
2. ESP32-PICO-KIT V4 *
3. Rode LED
4. Weerstand 150 Ω
5. SSD1306-gebaseerde displaymodule *
6. Draadverbindingen

* Verkrijgbaar in de Elektor-shop

De laatste jaren is de populariteit van Python enorm toegenomen, niet in het minst door de beschikbaarheid van single board-systemen zoals de Raspberry Pi. Maar Python heeft ook een bredere toepassing gevonden op andere gebieden zoals kunstmatige intelligentie en machine learning. Het ligt daarom voor de hand om te kijken hoe we Python (of in ieder geval de MicroPython-variant) kunnen gebruiken in microcontroller projecten.
Dit artikel bespreekt de basis van MicroPython, en in het bijzonder de belangrijkste beschikbare commando's en bibliotheken. Met een ESP32-microcontroller demonstreren we een paar kleine toepassingen: de microcontroller wordt geprogrammeerd met firmware die de Python-commando's interpreteert. Het Python-programma zelf wordt geschreven met behulp van een ontwikkelomgeving op een PC, en Python-commando's kunnen van de PC naar de microcontroller worden gestuurd als een heel programma of één voor één. Dat kan zowel via USB als via een draadloos netwerk. Laten we deze aspecten een voor een bekijken.

Programmeer- en ontwikkelomgevingen

In tegenstelling tot het Arduino-ecosysteem kan MicroPython werken met een reeks van geïntegreerde ontwikkelomgevingen (IDE's). Momenteel zijn de twee meest gebruikte omgevingen:

1. µPyCraft
2. Thonny

Ook kunnen we overwegen Anaconda Navigator te installeren, dat vooral wordt gebruikt voor het programmeren van microcontrollers op het gebied van kunstmatige intelligentie. Elke aanpak heeft zijn eigen voor- en nadelen. Zo heeft de µPyCraft IDE een minder verfijnde gebruikersinterface, met eenvoudige grafische elementen die doen denken aan tekstgeoriënteerde besturingssystemen. Thonny daarentegen biedt een complete grafische gebruikersinterface in de stijl van Windows (zie **figuur 1**). Deze IDE is zeer populair in de maker-cummunity, met name omdat hij beschikbaar is voor

een Raspberry Pi die draait onder het besturingssysteem Raspbian. Veel Raspberry Pi-gebruikers zijn dan ook al zeer bekend met Thonny. Thonny draait op alle belangrijke besturingssystemen, waaronder Windows, Mac OS X en Ubuntu Linux, en is gratis te downloaden [1]. Voor het gebruik van Thonny is het noodzakelijk dat Python 3 geïnstalleerd is op de machine die gebruikt zal worden voor ontwikkeling. Installatie-instructies zijn te vinden op de betreffende website [2].
Vervolgens kunnen we Thonny zelf installeren vanaf [1]. Selecteer rechtsboven op de pagina eerst het juiste besturingssysteem. Starten van het gedownloade bestand zal het gebruikelijke installatieproces starten. De IDE is nu klaar om onze eerste Python-applicatie te ontwikkelen.

De interpreter installeren

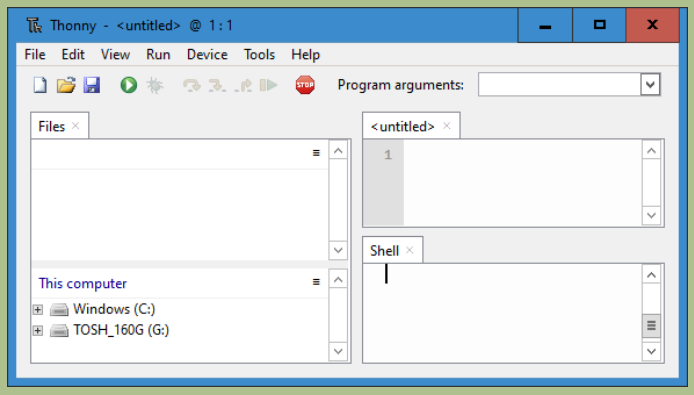
De volgende stap is het installeren van de MicroPython-firmware zelf. Deze kan worden verkregen van de officiële MicroPython-website [3], waar een reeks microcontroller-opties beschikbaar is. In dit artikel kijken we specifiek naar de ESP32, en dus moeten we de laatste versie downloaden die compatibel is met dit type microcontroller. Klik onder *Espressif ESP-based boards* op de afbeelding met als onderschrift *Generic ESP32 module*. De pagina die nu opent biedt een aantal verschillende firmware-varianten. Merk op dat de meest recente versie van de firmware vaak gemarkeerd is als 'unstable'. Deze optie is eerder bedoeld voor de ontwikkelaars van de interpreter-firmware zelf en voor mensen met een avontuurlijke inslag. Degenen onder ons die de voorkeur geven aan een stabieler systeem kunnen de meest recente niet als 'unstable' gemarkeerde versie kiezen:

GENERIC : esp32-idf3-20200902-v1.13.bin

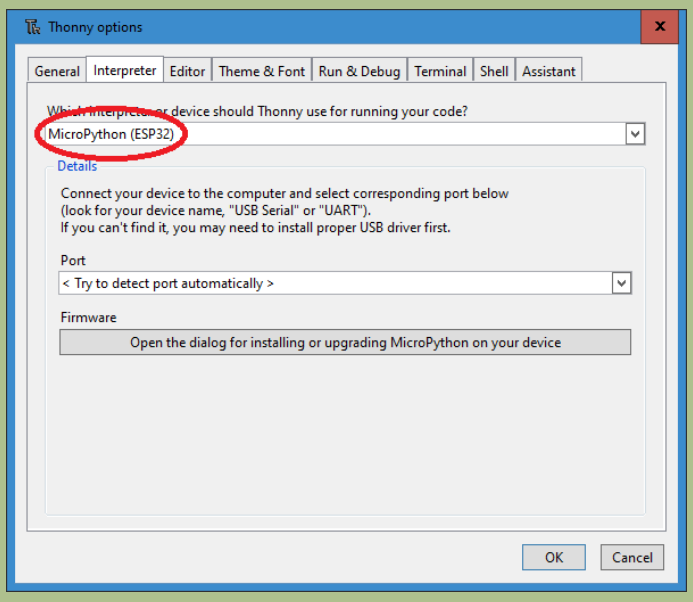
Klik op de juiste link en de firmware wordt gedownload.
Nu kunnen we het microcontrollerboard (bijvoorbeeld een ESP32-PICO-KIT: zie **Benodigde onderdelen**) via USB aansluiten op de PC en vervolgens de Thonny IDE starten. Hier moeten we *Select interpreter* kiezen in het *Run*-menu, wat het Thonny options venster opent: zie **figuur 2**. Op het tabblad *Interpreter* kunnen we kiezen uit een reeks opties, waaronder enkele die specifiek zijn voor de ESP32. Selecteer onder *Port* de USB-poort waarop de ESP32 is aangesloten. Als alternatief kunt u de optie *Try to detect port automatically* selecteren. Dit werkt echter niet in alle gevallen betrouwbaar.
Door in het vakje onder *Firmware* te klikken wordt nu de installer gestart. De poort wordt niet automatisch geconfigureerd en moet opnieuw worden ingevoerd. Navigeer vervolgens naar de firmware die we hierboven hebben gedownload. Door op *Install* te klikken wordt het installatieproces gestart. Als het proces is voltooid blijft het venster open: sluit het en u bent klaar om te beginnen met het programmeren van uw ESP32 in MicroPython.

Bibliotheken

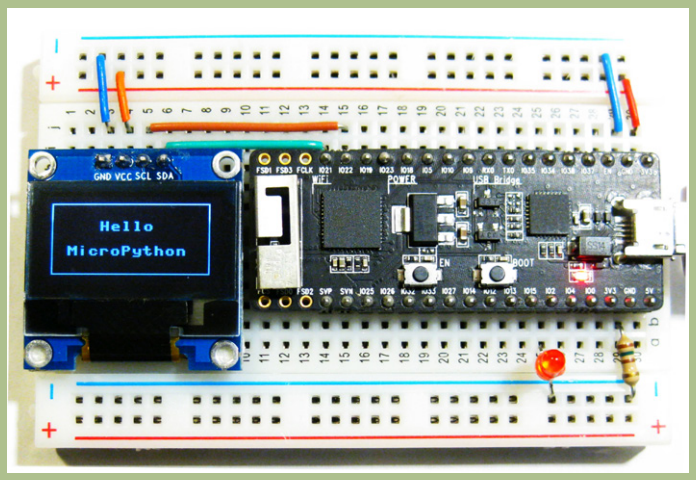
Werken met Python betekent werken met bibliotheken. Elk programma vanaf nul schrijven is op zijn zachtst gezegd zeer inefficiënt, en het overweldigende succes van Python is voor een groot deel te danken aan de vele bibliotheken die ervoor beschikbaar zijn. Helaas kunnen binnen de beperkte mogelijkheden van een microcontroller niet alle bibliotheken worden gebruikt, en daarom is een speciale selectie bibliotheken ontwikkeld voor gebruik met MicroPython. Veel van deze bibliotheken zijn standaard opgenomen in de Thonny IDE-download.
De twee belangrijkste standaardbibliotheken in MicroPython zijn *machine* en *time*. Het `import`-commando wordt gebruikt om de



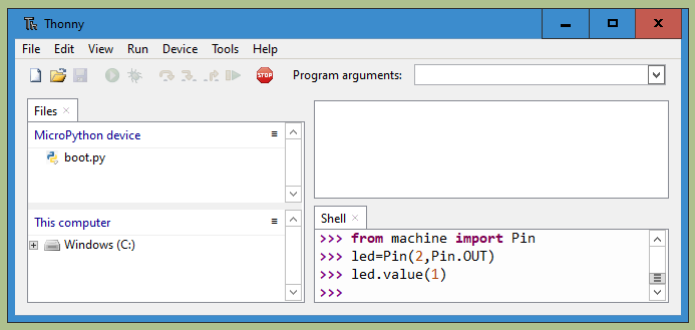
Figuur 1. De Thonny IDE.



Figuur 2. Opties voor de Thonny firmware-installer.



Figuur 3. ESP32-microcontroller met een OLED-display en een LED aangesloten op pin 2.



Figuur 4. De toestand van een I/O-poort wijzigen met behulp van de REPL-console.

bibliotheken beschikbaar te maken voor gebruik op de microcontroller. De volgende benaderingen (onder andere) kunnen worden gebruikt om een bibliotheekfunctie toegankelijk te maken:

```
> import module
> from module import name
```

In het eerste geval wordt de gehele module geïmporteerd; in het tweede geval worden alleen de gespecificeerde functies binnengehaald.
De *machine*-module bevat functies die specifiek zijn voor de hardware van een bepaalde microcontroller. Deze functies geven direct en ongehinderd toegang tot en controle over hardwarecomponenten, waaronder de CPU, timers, bussen en in- en uitgangspinnen. Wees ervan bewust dat verkeerd gebruik van deze module fouten, crashes en in extreme gevallen zelfs schade aan de hardware kan veroorzaken.
De Pin-klasse is een van de belangrijkste onderdelen van de *machine*-module. Een Pin-object wordt gebruikt om een I/O-pin aan te sturen en wordt toegewezen aan een fysieke pin op de microcontroller. Met het object kunnen uitgangsniveaus worden aangestuurd en ingangsniveaus worden gelezen.
De Pin-klasse voorziet in methoden om de modus van de pin in te stellen. Bijvoorbeeld: `IN` en `OUT` kunnen worden gebruikt om een pin te configureren als respectievelijk een ingang of een uitgang.
De *time*-module biedt een reeks tijdgerelateerde functies. De `sleep`-klasse uit deze module pauzeert de uitvoering van het lopende programma gedurende een gespecificeerd aantal seconden. Het argument kan zelfs een floating-point waarde zijn, zodat exacte vertragingen van een fractie van een seconde kunnen worden gespecificeerd. De commando's

```
from machine import Pin
from time import sleep
```

maken de `Pin`- en `sleep`-functies beschikbaar. Met de eerste kunnen we praten met de individuele poortpinnen van de microcontroller, en met de tweede kunnen we tijd-gebaseerde functionaliteit implementeren. Met behulp van het commando

```
led = Pin(2, Pin.OUT)
```

kunnen we een object `led` maken dat gekoppeld is aan I/O-pin nummer 2, en deze pin configureren als een uitgang. We kunnen nu verschillende waarden aan dit object doorgeven. Als we bijvoorbeeld het commando

```
led.value(1)
```

uitvoeren dan zal de waarde `1` naar het object worden geschreven. Dat betekent weer dat de bijbehorende I/O-pin, pin 2, op een hoog spanningsniveau wordt gezet. In het geval van de ESP32 is dat 3,3 V.

De REPL-console

Het is mogelijk om de toestand van een poort-pin te controleren door er eenvoudig via een stroombegrenzende weerstand een LED op aan te sluiten. (**Figuur 3** laat zien hoe dit gedaan wordt, en ook hoe we een stap verder kunnen gaan door een volledig OLED-paneel aan te sluiten: daarover later meer).
De poort, en dus ook de LED, kan heel eenvoudig worden aangestuurd met behulp van wat de 'REPL-console' wordt genoemd. REPL staat voor *Read Evaluate Print Loop*, wat een interactieve MicroPython-invoermethode is waarmee we rechtstreeks op de

ESP32 commando's kunnen uitvoeren, een zeer eenvoudige manier om commando's en programma's te testen.
De REPL-console wordt in Thonny de 'shell' genoemd en is beschikbaar rechtsonder in het hoofdvenster. U kunt de commando's uit de uittreksels hierboven direct in deze box invoeren: zie **figuur 4**. Na het invoeren van het laatste van de bovenstaande commando's zou de LED moeten oplichten. Het commando `led.value(0)` kan worden gebruikt om de LED weer uit te schakelen.
De REPL console heeft een paar interessante functies die het werken met MicroPython erg handig maken. Eerder ingevoerde commandoregels worden bijvoorbeeld opgeslagen, en kunnen indien nodig met de pijltjestoetsen weer worden opgeroepen.
Een andere handige functie is het aanvullen met tabs. Door op de 'Tab'-toets van het toetsenbord te drukken, wordt automatisch geprobeerd een gedeeltelijk ingevoerd woord te vervolledigen. Dit kan zelfs worden gebruikt om informatie te verkrijgen over de functies en methoden die beschikbaar zijn in een module of die op een object van toepassing zijn.
Als u bijvoorbeeld "ma" invoert en op de 'Tab'-toets drukt, wordt automatisch "machine" ingevuld (ervan uitgaande dat de machine-module reeds is geïmporteerd zoals hierboven beschreven). Door nu op de punt-toets ('.') en vervolgens weer op 'Tab' te drukken, verschijnt een complete lijst van alle functies die in de machine-module beschikbaar zijn: zie **figuur 5**. Deze functie maakt het in veel gevallen overbodig om commando's, objecten en methoden op te zoeken in de documentatie.

Draadloze toegang met WebREPL

Een van de grote voordelen van de ESP32 is zijn uitstekende ondersteuning voor draadloze connectiviteit. Wat is er dan logischer dan de REPL-console beschikbaar te maken via zo'n verbinding? Om dit te doen kunnen we gebruik maken van de WebREPL-interface.
De eerste stap om WebREPL aan de praat te krijgen, is zorgen dat het op de ESP32 geïnstalleerd en ingeschakeld is. WebREPL is standaard niet ingeschakeld en moet worden ingeschakeld met het eenmalige commando

```
import webrepl_setup
```

via de seriële poort. U krijgt de mogelijkheid om de functie in of uit te schakelen, en om een wachtwoord in te stellen. Na de configuratie moet de ESP32 opnieuw worden opgestart.
Om WebREPL via een draadloos netwerk te gebruiken, moet de ESP32 eerst verbonden zijn met dat netwerk. Geef daartoe de volgende commando's via de seriële REPL-console.

```
import network
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect('ssid', 'password')
```

De plaatshouders `ssid` en `password` moeten natuurlijk worden vervangen door de gegevens voor uw lokale draadloze netwerk. De opdracht

```
wlan.ifconfig()
```

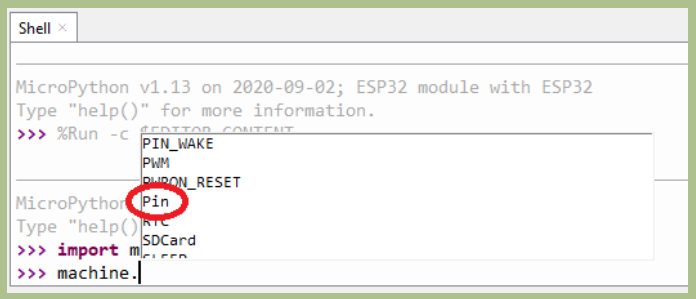
geeft vervolgens de IP-adresinstellingen weer die de ESP32 gebruikt om met het netwerk te communiceren: zie **figuur 6**. De commando's

```
import webrepl
webrepl.start()
```

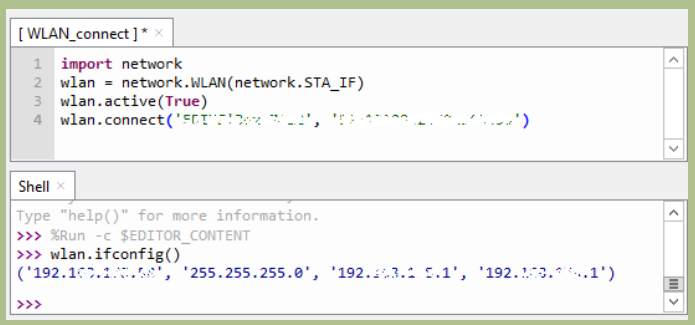
zullen nu de WebREPL-client inschakelen. In een webbrowser kunt u naar het adres

```
http://micropython.org/webrepl/#xxx.xxx.xxx.xxx:8266
```

gaan waarbij u `xxx.xxx.xxx.xxx` moet vervangen door het IP-adres dat u eerder met het commando `wlan.ifconfig()` hebt bepaald. U kunt dan het tabblad *Connect* gebruiken om verbinding te maken met de ESP32 met het wachtwoord dat u eerder tijdens de installatieprocedure hebt ingesteld.
Wanneer WebREPL wordt opgestart, kan het zijn dat de console in de 'raw REPL'-modus staat. Dit maakt het mogelijk om commando's direct in te voeren met kopiëren en plakken. In zulke situaties kunt u op control-B drukken om over te schakelen naar de normale



Figuur 5. Automatisch aanvullen.



Figuur 6. De ESP32 is verbonden met het lokale draadloze netwerk.

modus, waarin u normaal commando's kunt invoeren.
We zijn nu in staat om de ESP32 volledig draadloos te besturen. Met de eenvoudige schakelcommando's voor de uitgang kunnen we al basisfuncties voor domotica uitvoeren: zie **figuur 7**. Als er een geschikt bestandssysteem op de ESP32 is geïnstalleerd, is het ook mogelijk om draadloos software-updates uit te voeren, ook wel over-the-air (OTA) updates genoemd. Wanneer u met WebREPL werkt, moet u er rekening mee houden dat dit een experimentele functie is, waarvan dus niet verwacht kan worden dat hij in alle situaties absoluut betrouwbaar werkt.

Programmacontrole

De REPL- of WebREPL-console is het meest geschikt voor het uittesten van ideeën. Voor conventionele programmaontwikkeling is het editorpaneel (boven de REPL-console in het Thonny-venster) meer geschikt. Daar kunnen we bijvoorbeeld het hieronder beschreven programma voorbereiden dat is ontworpen voor automatische controle van nachtverlichting en verlichting van trappenhuizen: zie **figuur 8**. Zodra de code is ingevoerd en gestart met het start-icoon (witte pijl in een groene cirkel), wordt u gevraagd om het programma op te slaan. Kies hier de optie *MicroPython-device* en voer een programmanaam in (bijvoorbeeld *Automatic_LED*) en bevestig het opslaan. De LED zal nu gedurende drie seconden oplichten en daarna automatisch doven. Als u nogmaals op de startknop drukt, kan het programma onmiddellijk opnieuw worden uitgevoerd.
Om dit in het klassieke 'blinky' knipperLED-demonstratieprogramma te veranderen, hoeven we alleen maar een `while`-statement toe te voegen. Dit zorgt ervoor dat een bepaald commando of blok van commando's herhaaldelijk wordt uitgevoerd. Het specifieke geval van `while True:` creëert een zich eindeloos herhalende lus: in dit geval betekent dat dat de LED continu zal knipperen.

```
from machine import Pin
from time import sleep

led = Pin(2, Pin.OUT)

while True:
  led.on()
  sleep(0.5)
  led.off()
  sleep(0.5)
```

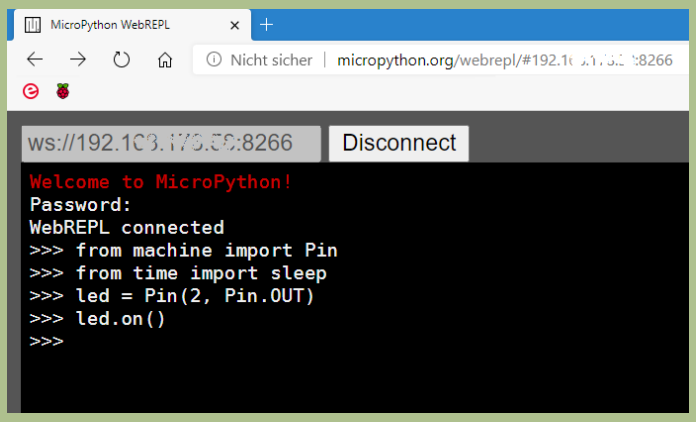
Merk hier op dat het `while`-statement moet worden afgesloten met een dubbele punt. Het volgende blok commando's is, volgens de standaard Python-conventie, ingesprongen met een constante hoeveelheid spaties. Thonny zorgt automatisch voor deze inspringing: zodra u een dubbele punt invoert, gevolgd door een druk op Enter, verschijnt de cursor op de volgende regel, één niveau ingesprongen. En natuurlijk biedt de programma-editor ook de functie tab-aanvullen. Een lopend programma kan worden gestopt door op control-C te drukken.

MicroPython in een notendop

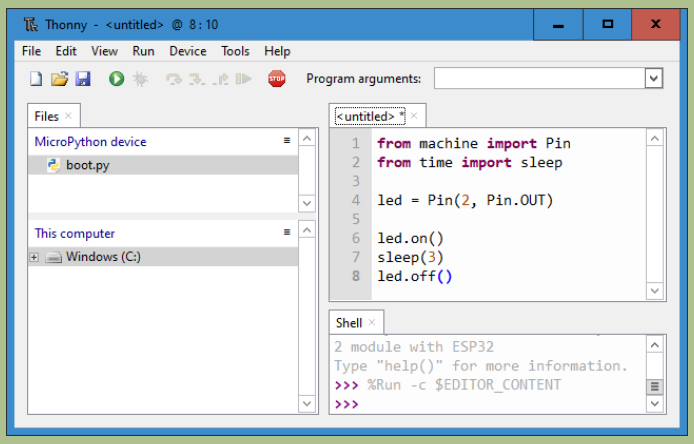
Hoewel de verzameling bibliotheken de levensader van MicroPython is, is een behoorlijke beheersing van de basiscommando's noodzakelijk om programma's van anderen te begrijpen en om uw eigen programma's te schrijven. Daarom zullen we nu kort kijken naar de krachtigste MicroPython-commando's.
Eenvoudig commentaar wordt voorafgegaan door het '`#`' symbool. Het commentaar loopt van '`#`' tot het einde van de regel.

```
>>> print("hello ESP32")          # this is a comment
  hello ESP32
```

Commentaar wordt tijdens de uitvoering van het programma genegeerd; het is er alleen om de ontwikkelaar van het programma informatie te geven. Met het hier gebruikte commando `print()` kan informatie worden uitgevoerd naar de console, zowel tekst- als



Figuur 7. WebREPL in de browser.



Figuur 8. Automatische LED.

Gerelateerde producten

- **Boek: MicroPython for Microcontrollers**
  www.elektor.nl/micropython-for-microcontrollers
- **ESP32-PICO-KIT V4**
  www.elektor.nl/esp32-pico-kit-v4
- **Breadboard**
  www.elektor.nl/breadboard-830-tie-points
- **SSD1306-based display module**
  www.elektor.nlm/blue-0-96-oled-display-i2c-4-pin

numerieke informatie; anderzijds kan `print()` ook rechtstreeks vanuit het terminalvenster worden aangeroepen.
Zoals we hierboven hebben gezien in het knipper-LED voorbeeldprogramma, kunnen statements gegroepeerd worden in blokken die geïdentificeerd worden door hun inspringing. Dat betekent dat accolades ('{' en '}') en consorten niet nodig zijn. Het voordeel van deze aanpak is dat u min of meer gedwongen wordt tot een gestructureerde programmeerstijl.
Het is heel eenvoudig om een variabele aan te maken in MicroPython, en het is met name niet nodig om het type van een variabele op te geven. Variabelen kunnen ook direct in de console worden gebruikt, en wel als volgt.

```
>>> a=17
>>> b=12
>>> print(a*b)

           204
```

In MicroPython hebben de rekenkundige operatoren hun conventionele wiskundige betekenis. Naast optellen, aftrekken, vermenigvuldigen en delen hebben we ook de `//`-operator voor integer-deling, `%` voor modulo (of rest van deling), en `**` voor machtsverheffen. De gebruikelijke vertakkings- en looping-statements zijn ook in MicroPython aanwezig. Vertakking wordt ondersteund door het sleutelwoord `if`, gevolgd door een voorwaarde; als de voorwaarde waar is dan worden de volgende statements uitgevoerd. Een `else`-sleutelwoord kan volgen, en een set van statements introduceren die in plaats daarvan worden uitgevoerd als de voorwaarde onwaar is. Bijvoorbeeld:

```
if True:
  # block 01
  print ("True")
else:
  # block 02
  print ("False")
```

Loops bieden de mogelijkheid om instructies herhaaldelijk uit te voeren. Een reeks instructies wordt uitgevoerd zolang aan een bepaalde voorwaarde wordt voldaan. Er zijn twee varianten beschikbaar:

- 'while' loops
- 'for' loops

Om de getallen van 1 tot 9 op de console af te drukken, kunt u een `while`-lus als volgt gebruiken.

```
number=1
while number<10:
  print(number)
  number=number+1
```

De reeks te herhalen instructies wordt aangegeven door de inspringing. Deze taak kan ook als volgt worden uitgevoerd met behulp van een for-loop.

```
for number in range(1, 10):
  print(number)
```

Automatisch alarmsignaal

Met deze basis programmeerstructuren onder de knie, kunnen we nu ons eerste praktische toepassingsprogramma maken. Het volgende programma is een automatisch SOS-baken dat kan worden gebruikt om een noodsignaal te geven bij zeilen of klimmen.

```
from machine import Pin
from time import sleep
led=Pin(2,Pin.OUT)

while True:
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.4)
    led.value(0)
    sleep(.5)
  sleep(1)
  for n in range(3):
    led.value(1)
    sleep(.1)
    led.value(0)
    sleep(.5)
  sleep(2)
```

Het OLED-display: een klein-formaat beeldscherm

Het is zeker mogelijk om een enkele LED te gebruiken om nuttige informatie over te brengen, bijvoorbeeld met behulp van morsecode zoals in de SOS-toepassing hierboven. Een aanzienlijk modernere benadering is natuurlijk het weergeven van gegevens op een OLED-display. Vaak maken dergelijke panelen gebruik van de SSD1306-displaycontroller. Voor dit voorbeeld gebruiken we een display met een diagonaal van slechts 0,96 inch (ongeveer 2,5 cm) en een resolutie van 128 x 64 pixels.
MicroPython is standaard voorzien van een bibliotheek voor SSD1306-gebaseerde displays. De bibliotheek laat de weergave

toe van tekst, numerieke gegevens en eenvoudige graphics.
De eenvoudigste SSD1306 modules hebben een interface met slechts vier pinnen, wat voldoende is om het display aan te sturen met behulp van het I²C-busprotocol. De aansluiting op het display wordt geïllustreerd in **figuur 3**; de vereiste aansluitingen zijn hieronder ook in tabelvorm weergegeven.

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

Het script in **listing 1** voert een tekstbericht uit naar het scherm en tekent enkele eenvoudige grafische elementen in de vorm van een kader dat het bericht omlijnt (zie figuur 3).
De relevante bibliotheek is beschikbaar als een standaard-package (*ssd1306.py* in het download-archief [5]) en kan apart naar het board worden geladen. De pinnen die gebruikt worden voor de I²C-interface worden als volgt gedeclareerd:

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

De eerste parameter, '`-1`', geeft aan dat de gebruikte module niet over een reset- of een interruptpin beschikt. Het aantal horizontale en verticale pixels in de module wordt opgegeven met het commando

```
oled = SSD1306_I2C(128, 64, i2c)
```

Het display is nu klaar voor gebruik. De functie `text()` kan worden gebruikt om informatie naar de displaybuffer te schrijven, en het display zelf wordt bijgewerkt met de methode `show()`. De functie `text()` accepteert de volgende argumenten.

- het bericht (een string)
- x- en y-coördinaten voor de tekst in pixeleenheden
- optionele tekstkleur: 0 voor zwart (niet verlicht) en 1 voor wit (verlicht)

De methode `show()` maakt de veranderingen zichtbaar op het scherm. Met de methode `rect()` kunt u een rechthoek tekenen op het scherm. Deze accepteert de volgende argumenten.

- x- en y-coördinaten voor de linkerbenedenhoek van de rechthoek
- x- en y-coördinaten voor de rechterbovenhoek van de rechthoek
- pixelkleur: 0 voor zwart en 1 voor wit

De instructie

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

tovert dus een rechthoekig kader langs de rand van het scherm. Met alleen deze eenvoudige commando's is het voor een toepassing al mogelijk om allerlei soorten informatie weer te geven, van basistekst tot complexe sensorwaarden.

Listing 1. Een bericht op het OLED-display.

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

i2c=I2C(-1,scl=Pin(22),sda=Pin(21))
oled = SSD1306_I2C(128, 64, i2c)
lin_hight = 9
col_width = 8

oled.fill(0)
oled.text("Hello", 5*col_width, 2*lin_hight)
oled.text("MicroPython", 2*col_width,
4*lin_hight)

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

Conclusie

MicroPython is een moderne en krachtige programmeertaal. Bovendien kunnen met zijn bibliotheken complexe projecten snel en eenvoudig worden gerealiseerd. In dit artikel hebben we laten zien hoe u een geschikte IDE installeert en een aantal eenvoudige applicaties maakt. In het tweede deel van deze serie zullen we verdere aspecten van MicroPython bekijken en, als praktische demonstratie, een voorbeeld geven van het aansturen van een grootformaat dotmatrix LED-display.
Meer informatie over MicroPython, over de ESP32-microcontroller en over de hier gepresenteerde voorbeelden is te vinden in het boek *MicroPython for Microcontrollers* [4].

210179-03

Een bijdrage van
Tekst en illustraties: **dr. Günter Spanner**
Redactie: **Jens Nickel**
Vertaling: **Jelle Aarnoudse**
Layout: **Harmen Heida**

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

WEBLINKS

[1] **Thonny**: https://thonny.org/
[2] **Python**: www.python.org/downloads
[3] **MicroPython**: http://micropython.org/download
[4] **Dr. G. Spanner, MicroPython for Microcontrollers, Elektor 2020**: www.elektor.nl/micropython-for-microcontrollers
[5] **Downloads**: www.elektormagazine.nl/210179-03