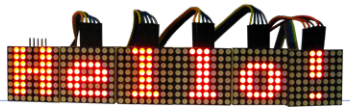
MicroPython voor de ESP32 c.s.

(Deel 2)

Eenvoudig Matrix Displays besturen

Dr. Günter Spanner (Duitsland)

Het eerste artikel in deze serie ging over de ontwikkelinterface en enkele eenvoudige commando's. Nu presenteren we praktische toepassingen. Met behulp van geschikte programmabibliotheken bijvoorbeeld, wordt het bouwen en programmeren van grote LED dot-matrix displays kinderspel. Naast het scrollen van teksten, kunt u ook de tijd of de lokale temperatuur weergeven.



Het mini-display uit het eerste artikel [1] was geschikt voor kleinere apparaten. In dit artikel zullen we laten zien hoe je met Python ook grote displays kunt realiseren. In dit geval LED-matrix modules met elk 8 x 8 = 64 individuele LED's. Door meerdere modules aan elkaar te rijgen, kunnen displays van bijna onbeperkte grootte worden gebouwd. Deze kunnen bijvoorbeeld worden gebruikt als 'live tickers' voor beurskoersen, of als displays voor scores op het sportveld thuis. Met een paar Python-instructies kunnen grote tijd- of temperatuurdisplays worden gemaakt, die bij vele gelegenheden zeer effectief voor reclamedoeleinden kunnen worden gebruikt.

Dot Matrix Displays

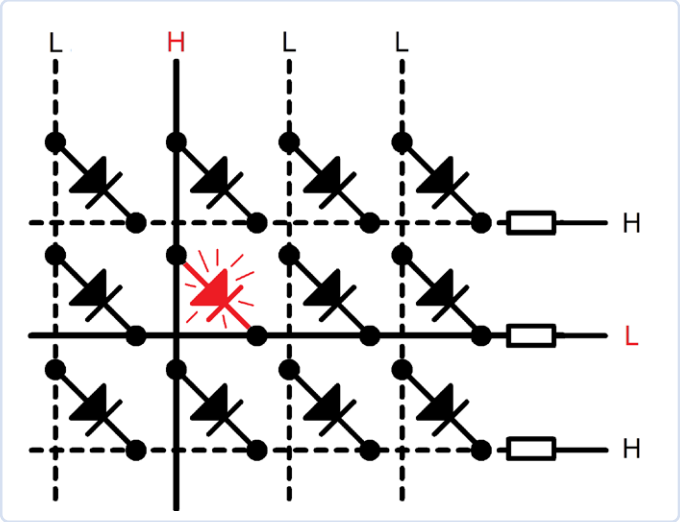
Dot matrix displays maken -naast de weergave van cijfers en letters- de uitvoer mogelijk van symbolen en eenvoudige afbeeldingen. Dit maakt ze superieur aan klassieke 7-segment displays, die alleen de weergave van cijfers van 0 tot 9 toestaan. Bovendien zijn 7-segment displays beperkt tot de beschikbare afmetingen. Dot-matrices daarentegen kunnen naar wens worden geschaald. Bovendien kan een veel grotere helderheid worden bereikt in vergelijking met OLED-displays en kunnen met LED-matrices display-oppervlakken van meerdere meters in lengte en breedte worden bereikt. Deze display-variant wordt daarom ook vaak gebruikt voor display- of reclameborden met grote oppervlakken op drukke pleinen, treinstations of luchthavens, op openbare bussen

Figuur 1: LED dot-matrices zijn zeer populair in Azië.

en treinen, enz. In de beurshallen van de wereld geven ze koersen weer als zogenaamde 'live tickers'. Overigens zijn dot-matrices bijzonder populair in de Aziatische regio, omdat er probleemloos tekens uit het Verre Oosten mee kunnen worden weergegeven, zoals **figuur 1** laat zien.

Besturing

LED-matrices met 5 x 7 = 35 LED's worden veel gebruikt. Er is echter ook een grote verscheidenheid van andere ontwerpen beschikbaar. Elementen met 8 x 8 LED's komen vaak voor. Voor de directe aansturing van 8 x 8 punten zouden 65 lijnen nodig zijn, omdat 64 anode-aansluitingen en een gemeenschappelijke kathode zouden moeten worden aangesloten. Met een matrix kan men met veel minder aansluitingen toe. Hier zijn 8 LED's met elkaar verbonden in een kolom en een rij. Er zijn dus maar 8 + 8 = 16 aansluitingen nodig. **Figuur 2** toont het principe voor een 3 x 4 matrix. In plaats van 13 aansluitingen voor de afzonderlijke aansturing van de LED's zijn er nu slechts 7 nodig. Om bijvoorbeeld de tweede LED op de tweede rij in een matrix te activeren, moeten alle rij-aansluitingen, behalve de tweede, op HIGH worden gezet en de tweede op GND-potentiaal. Voor de kolommen mag alleen de tweede kolom een hoog potentiaal hebben.



Figuur 2: Principe van de dot-matrixbesturing.

De directe aansturing van dot-matrices legt een groot beslag op de resources van de controller. Als ook nog sensorwaarden moeten worden geregistreerd of actuatoren moeten worden aangestuurd, bereikt zelfs een krachtige ESP32-controller al snel zijn grenzen. Ook de aansturing van grotere displays met honderd of meer LED's zou al snel een probleem worden - enerzijds door de vereiste rekenkracht, anderzijds door het beperkte aantal beschikbare pinnen op een controller. Bovendien zou de relatief lage uitvoersnelheid van Python-code hier ook een negatief effect hebben.

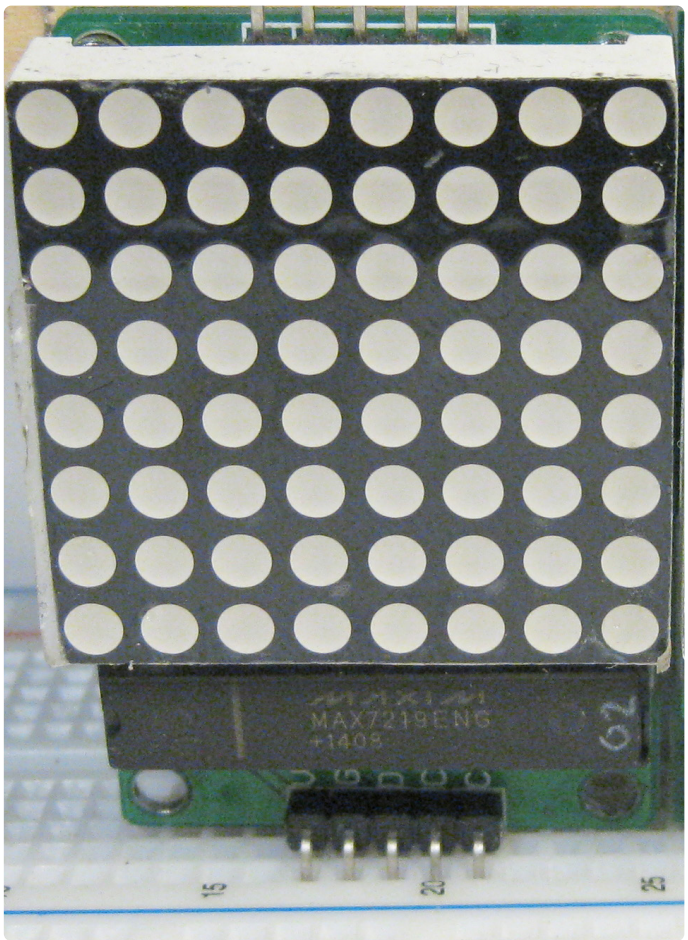
Het is daarom raadzaam goedkope display-drivers te gebruiken, zoals de MAX7219. Deze hebben een SPI-compatibele interface en kunnen displays met maximaal 8 x 8 = 64 pixels aansturen met slechts drie digitale pennen. De Serial Peripheral Interface (SPI) is wijdverbreid in gebruik en wordt met name in consumentenelektronica veelvuldig toegepast. Het bussysteem en vooral het gebruik ervan onder MicroPython zijn uitvoerig beschreven in een Elektor-boek van de auteur [2].

Deze drivers zijn samen met het eigenlijke matrixelement als complete modules verkrijgbaar. **Figuur 3** toont een voorbeeld.

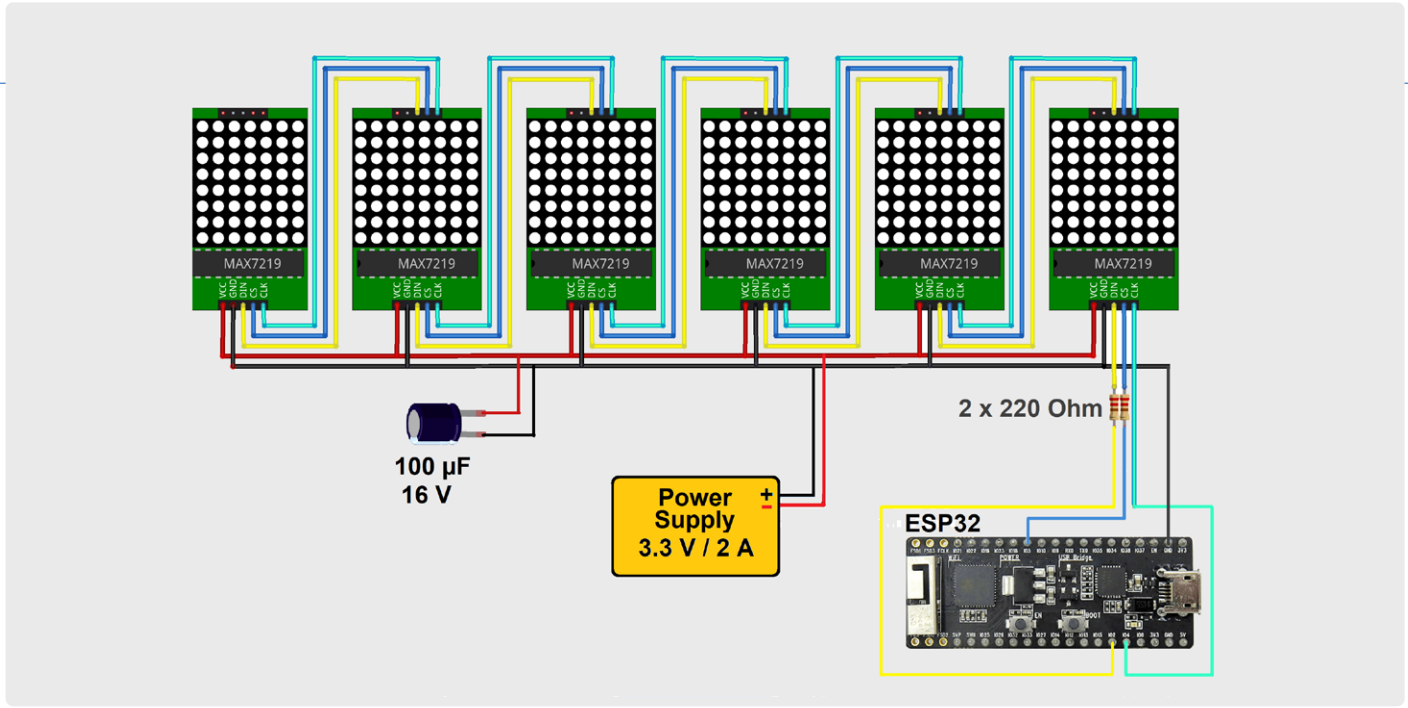
De aansluiting van de driver-modules op de ESP32 is zeer eenvoudig. Alleen de drie SPI-pennen moeten op het controllerbord worden aangesloten:

MAX7219_data (DIN) Port D02
MAX7219_load (CS) Port D05
MAX7219_clock (CLK) Port D04
MAX7219_GND ESP32 GND

Gezien het relatief hoge stroomverbruik is het aan te bevelen de voeding extern te verzorgen. De voedingsspanning van de modules moet ongeveer 3,3 tot 4 V bedragen om compatibiliteit met de ESP32 te garanderen. De modules krijgen zo de minimum toegelaten voedingsspanning volgens de datasheet en anderzijds wordt overbelasting van de controller-ingangen vermeden. Als alternatief kan ook een bidirectionele 3,3/5 V level-converter worden gebruikt.



Figuur 3: Enkelvoudige LED-matrix module.



Figuur 4: LED-matrix module array.

Daarnaast moet worden gezorgd voor voldoende buffercondensatoren. Laagohmige VCC- en massaleidingen zijn eveneens essentieel. Als met deze voorzorgsmaatregelen rekening wordt gehouden, kunnen displays van vrijwel elke grootte worden gebouwd. De controller wordt nauwelijks meer belast, omdat slechts enkele commando's via de SPI-bus hoeven te worden verzonden. Bovendien blijven voldoende pinnen vrij om externe sensoren of andere randapparatuur aan te sturen. De realisatie van displays met een groot oppervlak, bijvoorbeeld voor reclamedoeleinden of sportevenementen, is dan geen belemmering meer. **Figuur 4** toont het volledige schakelschema voor de opbouw van een matrixdisplay met zes cijfers met één ESP32-besturingsmodule.

Succes met de juiste bibliotheek

Een link voor een geschikte Python-bibliotheek voor het aansturen van de Maxim IC's is te vinden in het download-package [4]. Nadat het driver-bestand *Max7219.py* op de controller is geladen, zijn de volgende instructies beschikbaar:

```
spi = SPI(1, baudrate=10000000, polarity=1, phase=0,
sck=Pin(CLK), mosi=Pin(DIN))
ss = Pin(CS, Pin.OUT)
```

Voor de bovenstaande pin-toewijzing moet:

```
CLK = 4
DIN = 2
CS = 5
```

Worden ingesteld. Dan kan een display-object worden gemaakt:

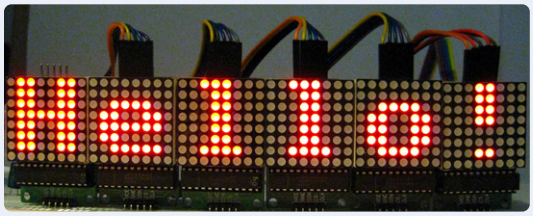
```
display = max7219.Matrix8x8(spi, ss, MN)
```

waarbij MN het aantal gebruikte matrixmodules is. Vervolgens kunnen teksten en afbeeldingen worden ontworpen met de commando's in **Listing 1.** Dit maakt het mogelijk om effectieve advertenties te maken die zelfs op enkele meters afstand nog goed leesbaar zijn.

LED Matrix in Actie

De volgende code toont een toepassingsvoorbeeld voor een display met zes 8 x 8 matrixelementen:

```
# LED_matrix_test.py
```



Figuur 5: Dot-matrixdisplay met zes 8x8 matrixelementen.

Figuur 6: Dot-matrixdisplay met twaalf matrixelementen.

Listing 1: Display Commando's.

```
display.pixel(x,y,1)          # stel een pixel in op de coördinaten x,y
display.pixel(x,y,0)          # verwijder een pixel op de coördinaten x,y
display.hline(x,y,l,1)        # horizontale lijn van x,y - lengte l
display.vline(x,y,l,1)        # verticale lijn van x,y - lengte l
display.line(a,b,c,d,1)       # lijn van a,b tot c,d
display.rect(a,b,c,d,1)       # rechthoek met hoeken a,b, c,d
display.text('Text',x,y,1)    # tekst op positie x,y
display.scroll(x,0)           # scroll met x pixels
display.show()                # ververs display
```

```
importeren max7219
van machine importeren Pin, SPI

spi = SPI(1, baudrate=10000000, polarity=1, phase=0,
sck=pin(4), mosi=pin(2))
ss = Pin(5, Pin.OUT)

display = max7219.Matrix8x8(spi, ss, 6)
display.text('Python',0,0,1)
display.show()
```

Nadat het programma in de ESP-controller is geladen, verschijnt de tekst op het display (**figuur 5**).
Het display kan gemakkelijk worden uitgebreid met nog meer matrixmodules. **Figuur 6** toont een matrix met 12 modules. Er kan echter meer dan alleen statische weergaven en teksten worden weergegeven. Ook bewegende afbeeldingen kunnen zonder problemen in Python worden geïmplementeerd. De volgende sectie toont een voorbeeld.

Tickers en Aandelentickers

De scroll-functie kan worden gebruikt om tickers en dergelijke te maken. Dit maakt het mogelijk om langere teksten uit te voeren, zelfs op kleine of korte beeldschermen. Het programma in **listing 2** scrollt de tekst "Python" van rechts naar links over een dot-matrix display.
De afzonderlijke letters worden daarbij toegewezen via

```
display.text('...',40,0,1)
```

en aan de rechterrand van het scherm aangemaakt. Vervolgens worden deze naar links verplaatst met de `pixelDistance` waarde via de `moveLeft` functie:

```
voor i in range(8):
display.scroll(-1,0)
sleep(speedFactor)
```

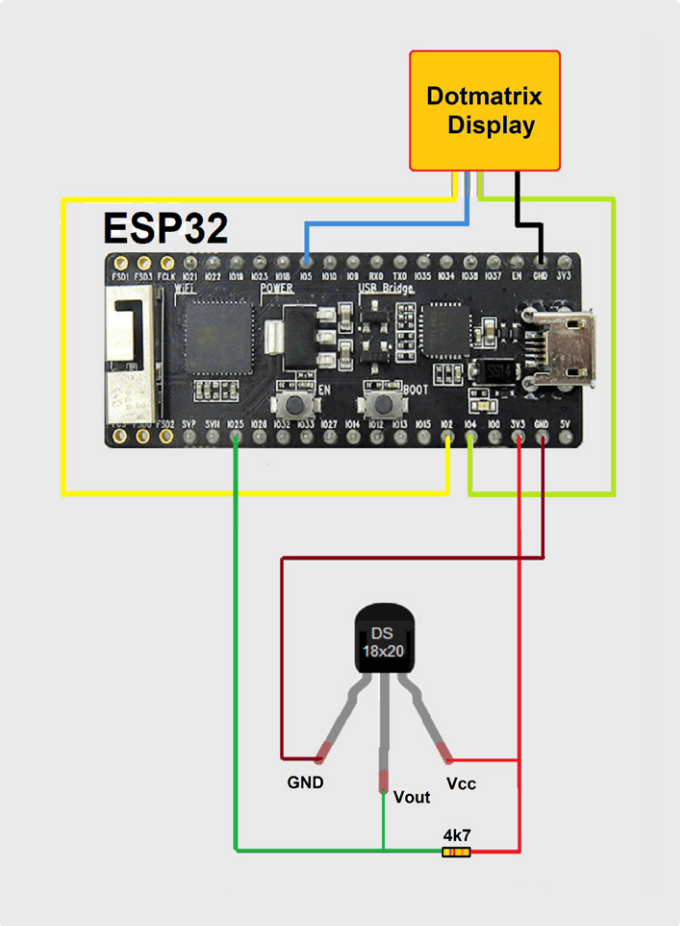
De snelheid kan worden gewijzigd via de speedFactor-waarde. Het scherm is live in actie te zien in een YouTube-video [3].

Listing 2: Ticker.

```
# LED_matrix_ticker.py

import max7219
from machine import Pin, SPI
from time import sleep

spi = SPI(1,baudrate=10000000, polarity=1, phase=0,
    sck=Pin(4), mosi=Pin(2))
ss = Pin(5,Pin.OUT)
speedFactor=0.05
pixelDistance=7
display = max7219.Matrix8x8(spi, ss, 6)

def moveLeft(Pixel):
    for i in range(Pixel):
        display.scroll(-1,0)
        sleep(speedFactor)
        display.show()

while True:
    display.text('P',40,0,1)
    moveLeft(pixelDistance)
    display.text('y',40,0,1)
    moveLeft(pixelDistance)
    display.text('t',40,0,1)
    moveLeft(pixelDistance)
    display.text('h',40,0,1)
    moveLeft(pixelDistance)
    display.text('o',40,0,1)
    moveLeft(pixelDistance)
    display.text('n',40,0,1)
    moveLeft(pixelDistance)
    display.text(' ',40,0,1)
    moveLeft(pixelDistance)
```

Temperatuurdisplay in Groot Formaat

Veel apotheken, banken en detailhandelaren proberen de aandacht van potentiële klanten te trekken door reclame te maken met grote temperatuur- of tijddisplays. Dergelijke displays zijn ook blikvangers op sportevenementen, beurzen, tentoonstellingen en voor FabLabs. Om bijvoorbeeld een temperatuurdisplay te realiseren, hoeft u alleen maar een geschikte sensor toe te voegen aan de hierboven gepresenteerde opstelling. Een bijzonder geschikte variant is hier de DS18x20 van Maxim Integrated (voorheen Dallas). Met de juiste Python-bibliotheek kan de sensor zonder problemen worden uitgelezen.



Figuur 7: DS18x20 temperatuursensor op ESP32-controller.

Deze sensoren communiceren via de 1-Wire bus en nemen dus slechts één enkele I/O-pin van de ESP32 in beslag. Bovendien is er een kant-en-klare Python-bibliotheek beschikbaar voor deze sensorserie. Naast de sensor zelf is slechts een 4,7 kΩ pull-up weerstand nodig. Bij gebruik van de interne pull-up van de ESP32 kan zelfs deze worden weggelaten. De sensor heeft de volgende kenmerken:

- Voedingsspanning: 3,0 V tot 5,5 V
- Temperatuurbereik: -55 °C tot +125 °C
- Meetnauwkeurigheid: ±0,5 °C (-10 °C tot +85 °C)
- Resolutie: 9-bit, komt overeen met ca. 1/10 °C
- Duur van de meetperiode: 750 ms (max.)

Via de 1-Wire bus kunnen meer dan één sensor per pin worden geëvalueerd. Het speciale protocol van het 1-Wire bussysteem maakt het mogelijk vrijwel elk aantal temperatuursensoren parallel via een enkele controller-pen op te vragen. In het onderstaande wordt echter slechts een enkele sensor gebruikt.
Figuur 7 toont de aansluiting van de DS18x20 op de ESP32. Een evaluatieprogramma voor de DS18x20 geeft de geregistreerde temperatuurwaarden weer op de console:

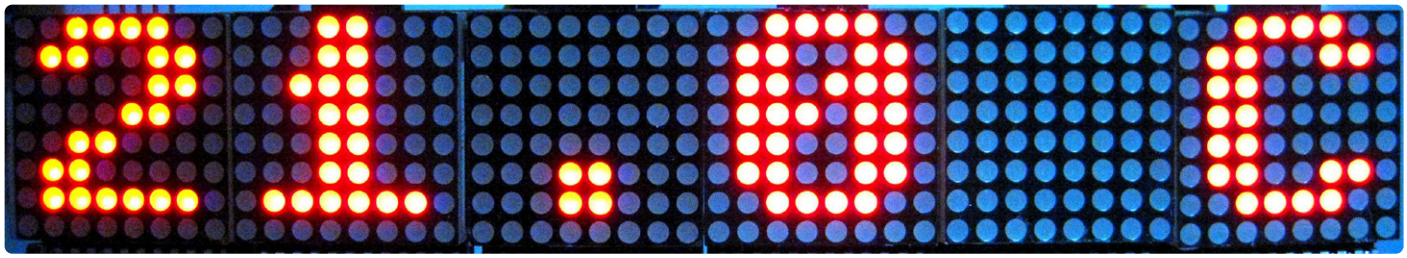
```
# DS18x20_TempSens_demo.py

from machine import pin
import onewire
import ds18x20
import time

ow = onewire.OneWire(Pin(25))  # init one wire bus
ow.scan()
ds=ds18x20.DS18X20(ow) # create ds18x20 object

while True:
  units=ds.scan() # scan for ds18x20 units
  ds.convert_temp() # convert temperature
  for unit in units:
    print(ds.read_temp(unit)) #display
  time.sleep(1)
  print()
```

De modules voor het uitlezen van de sensor zijn weer standaard beschikbaar in de MicroPython firmware.
De uitbreiding van het programma naar het LED-matrix display wordt gedaan met een paar regels, zie **Listing 3**. **Figuur 8** toont het resultaat. Verdere details over de aansluiting van diverse sensoren voor o.a. lichtintensiteit, vochtigheid of magnetische velden, zijn te vinden in het boek [2].

Geanimeerde afbeeldingen

Naast letters en numerieke gegevens kunnen ook afbeeldingen en zelfs animaties op de LED matrix worden weergegeven. Het programma in **listing 4** tovert een willekeurige pixelafbeelding op het display.
De grafiek wordt gegenereerd via het bitmap `icoon`. Dots die later moeten oplichten, krijgen een “1”. Donkere punten krijgen een “0”:

```
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 1, 0, 0, 1, 1, 0],
[0, 1, 1, 0, 0, 1, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
```



Figuur 8: Een mega-temperatuurweergave.

Listing 3: Temperatuur Display.

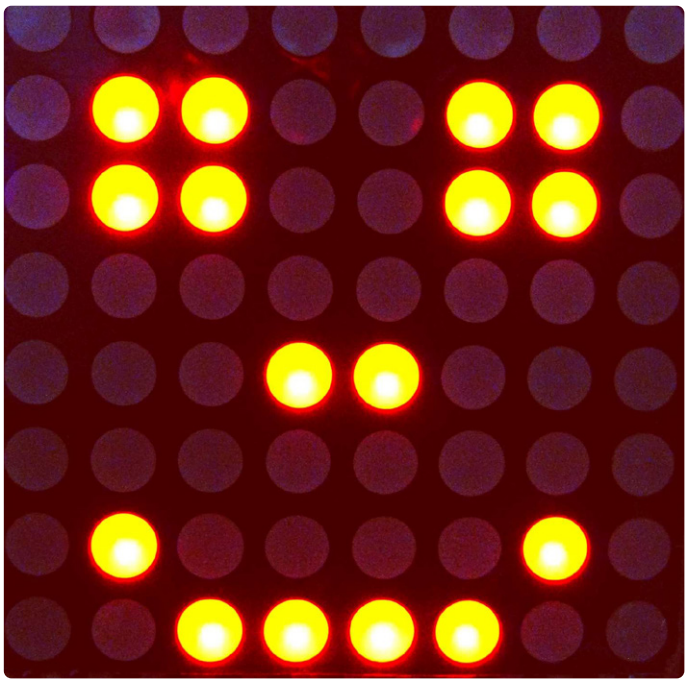
```
# DS18x20_to_LED_matrix.py

import max7219
from machine import Pin, SPI
import onewire
import ds18x20
import time

# pin assignment for LED matrix
spi = SPI(1, baudrate=10000000, polarity=1, phase=0, sck=Pin(4), mosi=Pin(2))
ss = Pin(5, Pin.OUT)

ow = onewire.OneWire(Pin(25))                          # init one wire bus
ow.scan()
ds=ds18x20.DS18X20(ow)                                 # create ds18x20 object

while True:
    units=ds.scan()                                    # scan for ds18x20 units
    ds.convert_temp()                                  # convert temperature
    for unit in units:
        T=ds.read_temp(unit)
        output="T = {:5.1f} °C"                        # generate formated string
        print(output.format(T))

        output=str(T)
        output="{:4.1f} C"                             # generate formated string for LED matrix
        display = max7219.Matrix8x8(spi, ss, 6)
        display.text(output.format(T),0,0,1)           # write temperature to LED matrix
        display.show()
    time.sleep(1)
```

```
[0, 0, 0, 1, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 1, 0],
[0, 0, 1, 1, 1, 1, 0, 0],
```

Op deze manier kunnen alle iconen worden gemaakt. De functie *enumerate* zet de bitmap om in een weer te geven grafiek:

```
for y, row in enumerate(icon):
  for x, c in enumerate(row):
    display.pixel(x, y, c)
```

Dit resulteert in de LED-matrix in **figuur 9**. De scroll-functie in de hoofdlus kan vervolgens worden gebruikt om het op deze manier gegenereerde beeld over de display-unit te verplaatsen (zie ook de YouTube-video [3]).



Figuur 9: Zelf-gedefinieerde pixelafbeelding op het dot matrix display.

Samenvatting en vooruitzichten

Nadat in het eerste artikel van de serie [1] de elementaire commando's waren geïntroduceerd, konden deze nu in het tweede deel in diverse praktische toepassingen worden gebruikt. Krachtige bibliotheken maken het mogelijk om indrukwekkende projecten te implementeren met slechts een paar regels programma-code. Dit bevestigt de opmerkelijke kracht van MicroPython ook voor controller-toepassingen. Meer informatie en vele praktische projecten zijn te vinden in het boek *MicroPython voor Microcontrollers* [2]. Onder andere de besturing van servo's, draadloze datatransmissie

Listing 4: Pixel Graphic.

```
#LED_matrix_icon.py

import max7219
from machine import Pin, SPI
from time import sleep

# sck -> serial clock - CLK; mosi -> master out slave in - DIN
# ss -> chip select - CS
spi=SPI(1,baudrate=10000000,polarity=1,phase=0,sck=Pin(4),mosi=Pin(2))
ss=Pin(5,Pin.OUT)

display=max7219.Matrix8x8(spi,ss,6)
display.brightness(0)

icon = [
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 1, 0, 0, 1, 1, 0],
    [0, 1, 1, 0, 0, 1, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 1, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0],
    [0, 1, 0, 0, 0, 0, 1, 0],
    [0, 0, 1, 1, 1, 1, 0, 0],
]

for y, row in enumerate(icon):
    for x, c in enumerate(row):
        display.pixel(x, y, c)

display.show()

while True:
    for i in range(40):      #move square left to right
        display.show()
        display.scroll(1, 0)
        sleep(0.05)

    for i in range(40):      #move square right to left
        display.show()
        display.scroll(-1, 0)
        sleep(0.05)
```

via RFID, het MQTT protocol en de transmissie van sensorwaarden naar het Internet komen daar ook uitgebreid aan bod.

Bovendien zal de combinatie van Python met het snel groeiende gebied van machine-learning (ML) en artificial intelligence (AI) in de nabije toekomst toepassingen mogelijk maken die voorheen ondenkbaar waren. Nieuwe controller-generaties, zoals de Kendryte K210, maken bijvoorbeeld nu al de directe evaluatie van audio- en camerasignalen mogelijk. Samen met op Python gebaseerde beeldverwerkingsalgoritmen opent dit een breed scala aan toekomstgerichte mogelijkheden voor ontwikkelaars.

210179-B-03

GERELATEERDE PRODUCTEN

- **Boek en E-book: MicroPython for Microcontrollers (SKU 19736, 19737)**
  www.elektor.nl/19736
  www.elektor.nl/19737
- **ESP32-PICO-KIT V4 (SKU 18423)**
  www.elektor.nl/18423
- **Breadboard (SKU 17092)**
  www.elektor.nl/17092

Een bijdrage van
Tekst en illustraties: **Dr. Günter Spanner**
Redactie: **Jens Nickel, C.J. Abate**
Layout: **Sylvia Sopamena**
Vertaling: **Jelle Aarnoudse**

Vragen of opmerkingen?
Indien u technische vragen of opmerkingen heeft over dit artikel, kunt u ons e-mailen op editor@elektor.com.

WEBLINKS

[1] Dr. Günter Spanner, "MicroPython for the ESP32 and Friends," ElektorMag 7-8/2021: www.elektormagazine.com/200179-01
[2] Dr. G. Spanner, MicroPython for Microcontrollers, Elektor 2020.: www.elektor.com/micropython-for-microcontrollers
[3] Demo video on YouTube: http://www.youtube.com/watch?v=5uligjyKMEk
[4] Software Download: http://www.elektormagazine.com/210179-B-01