|  |
| --- |
| MicroPython  deel 2  *Github organiseren en sensoren* |

Geschreven door

Arjan Kamberg  
<https://www.linkedin.com/in/arjankamberg/>

Leerjaar: 2023/ 2024

Copyright  
Arjan Kamberg

Van stroomdiagram naar Python code  
© 2023, Arjan Kamberg  
Uitgegeven in eigen beheer  
(Python@AKamberg.nl)

Alle rechten voorbehouden.  
Niets uit deze uitgave mag worden verveelvoudigd, opgeslagen in een geautomatiseerd gegevensbestand en/of openbaar gemaakt in enige vorm of op enige wijze, hetzij elektronisch, mechanisch, door fotokopieën, opnamen of op enige andere manier zonder voorafgaande schriftelijke toestemming van de uitgever. Het gebruik maken van dit boek voor opleidingsdoeleinden mag alleen gedaan worden met uitdrukkelijke toestemming van de uitgever.

Versie Datum: 11-10-2023

Inhoudsopgave

[Voorwoord 3](#_Toc147947769)

[1 Welke kennis heb je nodig. 4](#_Toc147947770)

[2 Componenten 5](#_Toc147947771)

[2.1 Buzzer 5](#_Toc147947772)

[2.2 7 segment display 5](#_Toc147947773)

[2.3 Ultrasoon sensor 5](#_Toc147947774)

[2.4 Lcd-Display (I2C) 5](#_Toc147947775)

[3 Webserver 6](#_Toc147947776)

[4 Python op een Raspberry pi 7](#_Toc147947777)

[4.1 Kalender.py op de Raspberry pi 7](#_Toc147947778)

[4.2 Python programmeren op de Raspberry Pi 8](#_Toc147947779)

[4.3 GPIO Emulator gebruiken 9](#_Toc147947780)

[5 Max 7219 11](#_Toc147947781)

[6 Neopixels 12](#_Toc147947782)

# Voorwoord

Alle apparaten om ons heen hebben tegenwoordig wel een computerprogramma in zich dat de functies bepaald. Dit boek is geschreven voor MBO-studenten die programmeren leren aan de hand van de taal Python. Python is een taal die op steeds meer gebieden ingezet wordt door zijn kracht en eenvoud.

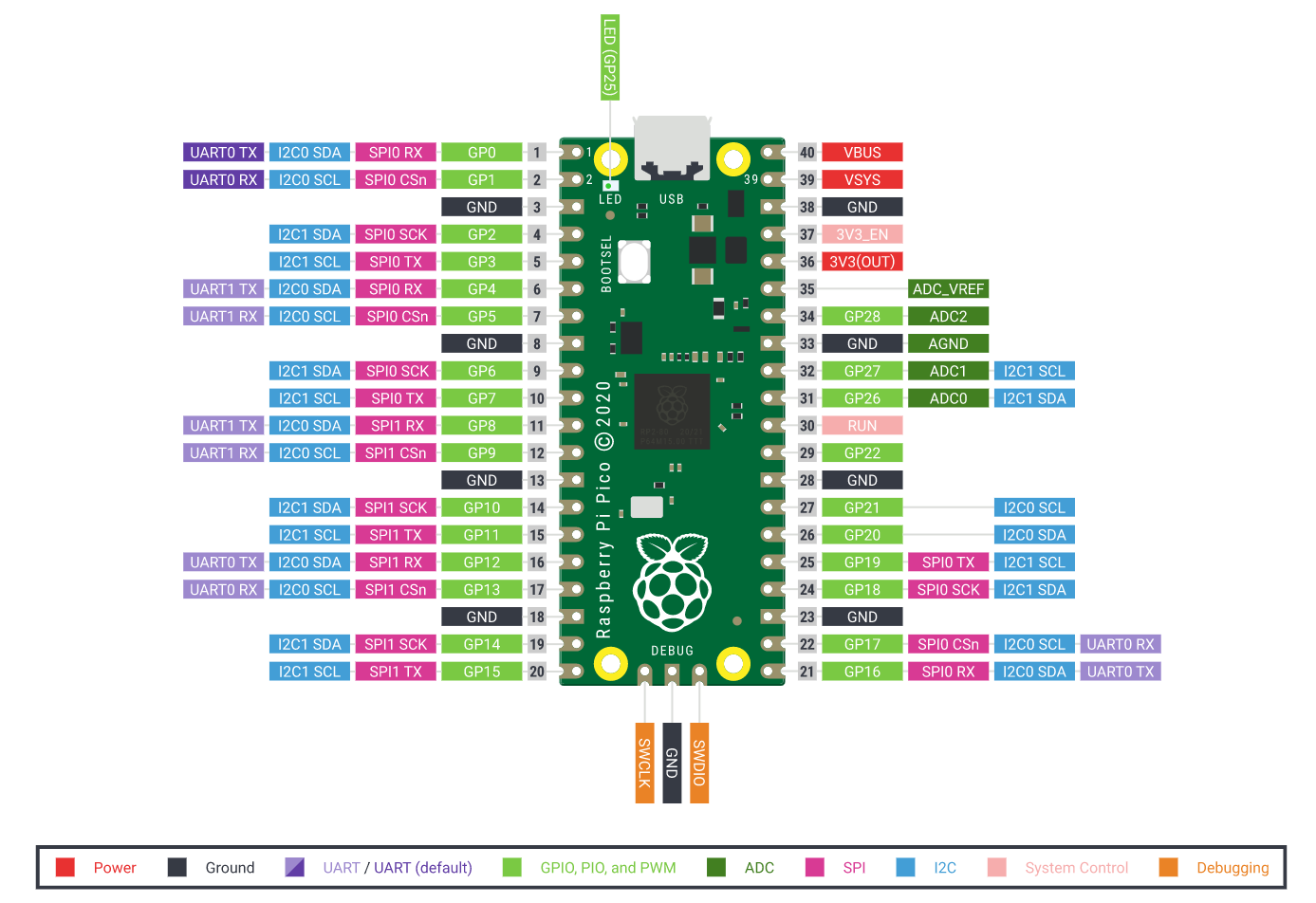
In dit boek worden niet alle dingen behandeld, de focus zit op het begrip van programmeren leren. Zo worden lussen bewust alleen met while-statements gemaakt. De opbouw van het programmeren moet eenduidig zijn en er moet zo min mogelijk keuzestress zijn bij het programmeren. De overwegingen en didactische uitleg wordt beschreven in de docenten versie van dit boek.

Bij een goed begrip van programmeren, en een opbouw die duidelijk en testbaar is wordt bereikt dat het eindproduct ook uitbreidbaar is en dat het zich aan de conventies houdt die bruikbaar zijn bij programmeurs met jaren ervaring.



Figuur 1: Programmeurs die iemand anders code niet snappen. (C) CSEstack.org

# Welke kennis heb je nodig.



# Componenten

## Buzzer

## 7 segment display

## Ultrasoon sensor

## Lcd-Display (I2C)

# Webserver

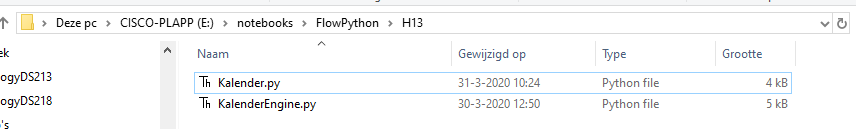
<https://electrocredible.com/raspberry-pi-pico-relay-control-using-web-server/>

# Python op een Raspberry pi

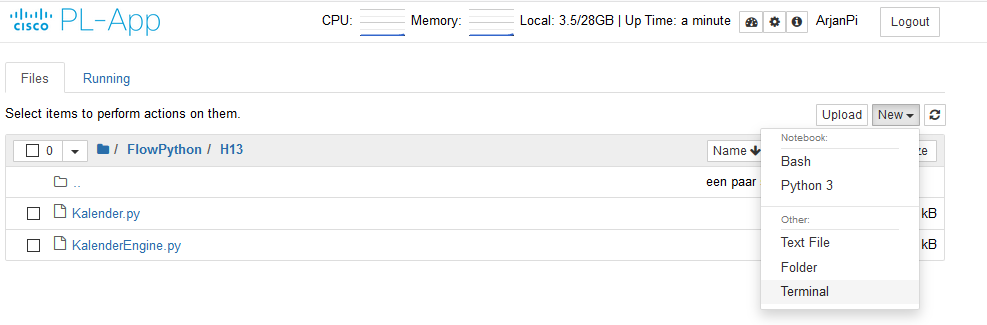
In dit hoofdstuk gaan we de code die we hebben gemaakt gebruiken op een Raspberry pi.   
Thonny staat standaard geïnstalleerd op onder anderen Ubuntu. Daar kunnen we programmeren zoals we dat al tot nu toe hebben gedaan. Hebben we dat niet op de raspberry pi staan dan kunnen we met onderstaande methode de code op de Pi zetten.  
Om contact te maken met de Raspberry pi maken we gebruik van Cisco PL-App.

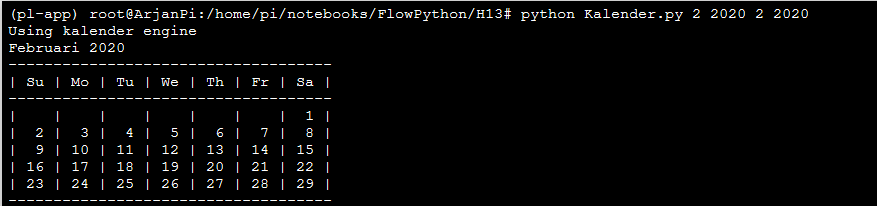
## Kalender.py op de Raspberry pi

Voor deze opdracht gebruiken we de Raspberry pi om je gemaakte code uit te voeren. We hoeven niets aan de code te veranderen, we hoeven alleen de code naar de Raspberry pi te kopiëren.



We kunnen via PL-App naar de code kijken Bekijk deze folder via de Terminal.

We kijken eerst of de code ook op de Raspberry pi werkt.



Figuur 2: Python code starten op Raspberry pi

Op dezelfde manier kan je ook voor andere operating systems een bestand maken dat uitgevoerd kan worden.

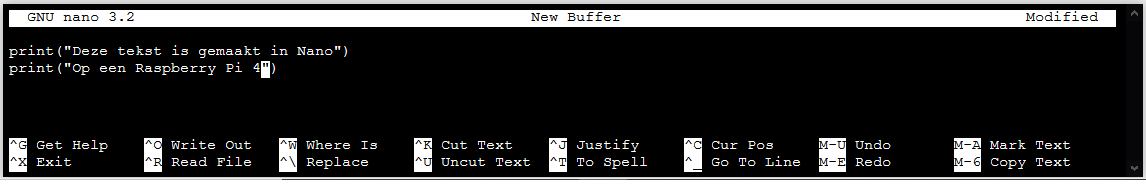
## Python programmeren op de Raspberry Pi

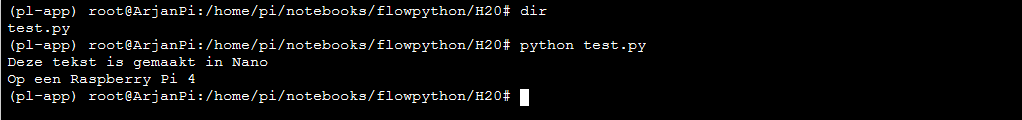
Vanuit de Terminal kan je met iedere editor python code schrijven. Dit is niet alleen voor de Raspberry pi, maar voor iedere operating system. Dit komt dat python code een tekstbestand is met de extensie .py.

Als we het bestand hebben opgeslagen kunnen we de python code starten zoals in het voorbeeld “Figuur 2: Python code starten op Raspberry pi”.

Een groot nadeel van een tekst editor is dat we niet geholpen worden met kleuren of de commando’s correct zijn. De IDE (“Integrated development environment”) is niet aanwezig bij een tekst editor.

We kunnen wel eens kijken hoe het eruit ziet als we code maken in bijvoorbeeld Nano.





Het grote voordeel van programmeren op de Raspberry pi is dat op deze microcomputer GPIO-pinnen zitten (General Purpose Input Output). Dit zijn vergelijkbare digitale poorten als op een Arduino.

Onderstaande code zorgt ervoor dat de pin 17 hoog en laag wordt iedere seconden. Als je op de schakelaar drukt op poort 23, dan stopt het knipperen, en eindigt het programma.



## GPIO Emulator gebruiken

Als je alleen een Terminal tot je beschikking hebt is dit best lastig om de software te ontwikkelen. Het makkelijkste is als we met Thonny de code kunnen ontwikkelen en dan als we het af hebben op de Raspberry pi kunnen neerzetten. Voor deze situaties zijn er vaak emulators gemaakt, en zo ook voor de GPUI. Het verdere voordeel is dat je met een emulator geen componenten kapot kan maken. Een Raspberry pi is minder robuust dan een Arduino, die wel tegen een kortsluiting kan.

We gebruiken de Pi GPUI Emulator ( <https://sourceforge.net/projects/pi-gpio-emulator/> ) gemaakt door [Roderick Vella](https://sourceforge.net/u/vellarod/). Het zip bestand moet gedownload worden en naast je project uitgepakt worden. Je hebt dan extra modules tot je gebruik die de GPIO emuleert.



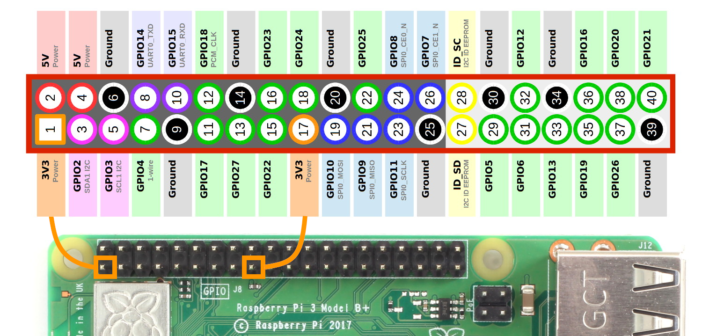
Het enige wat je in je code moet wijzigen is regel 1 en regel 2, zie de code in ”Figuur 3: Code wijziging voor emulator”. Als we klaar zijn met het ontwikkelen dan wordt regel 1 als commentaar gemaakt, en wordt regel 2 weer actief gemaakt, en de code kan op de Raspberry pi gezet worden.



Figuur 3: Code wijziging voor emulator

Als we ons programma nogmaals runnen dan zien we het volgende gebeuren. Er kont een balk bovenaan het scherm die overeenkomt met de pinnen van de Raspberry pi.

De oranje GPIO17 zal de waarde 0 en 1 krijgen. Indien je op GPIO23 drukt zal het knipperen van GPIO17 stoppen. Let op: GPIO17 is niet pin 17!!!!



Figuur 4: Pin layout. bron: www.raspberrypi-spy.co.uk



Figuur 5: EmulatorGUI in actie

De code zelf heeft een paar nieuwe dingen. Op regel 3 wordt de module time geïmporteerd, die module heeft de sleep() functie die zorgt dat op regel 15 en 17 de code even wacht.

We hebben al in hoofdstuk “**Error! Reference source not found.**” de try-except structuur gezien, waarbij als er een fout optrad gelijk naar except gesprongen werd. Na alle excepts (of in plaats van zoals hier) kan ook nog een finally opgenomen worden. Deze wordt ervoor dat het blok erna altijd uitgevoerd wordt. Dus zowel na een except als nadat de code zonder problemen is doorlopen.

De GPIO-methoden zijn vergelijkbaar met hoe er bij een Arduino geprogrammeerd wordt. Omdat de Raspberry pi alleen digitale poorten heeft is het resultaat van het uitlezen van een poort altijd True of False, zodat op regel 18 het resultaat van de functie gebruikt kan worden in een if-statement.