Multifunctionele slimme kamerklok

januari 26, 2025 · Electronica, Mechatronica, Meten, Microcontroller, Raspberry Pi, Regelen, Software

Auteurs

ZEH Otten

Klokken blijven een bron van fascinatie binnen de wereld van elektronica en design. Door de jaren heen zijn talloze mechanische en elektronische klokken verschenen op hobby- en elektronica platformen, elk met hun eigen unieke benadering van tijdweergave. Vanuit mijn eigen interesse (zie ook publicatie in PC-Active 313, juli 2020 : link) ontstond het idee voor een originele tekstklok – een visuele interpretatie van tijd die verder gaat dan traditionele cijfers en wijzers.

Dit ontwerp sluit aan bij de moderne wensen: duurzaam, energiezuinig en functioneel. De klok presenteert namelijk niet alleen de tijd op een eigentijdse manier, maar is ook veelzijdig genoeg om data uit een database weer te geven.
Met zijn royale formaat van 12 x 60 cm is hij bovendien een echte blikvanger in huis.

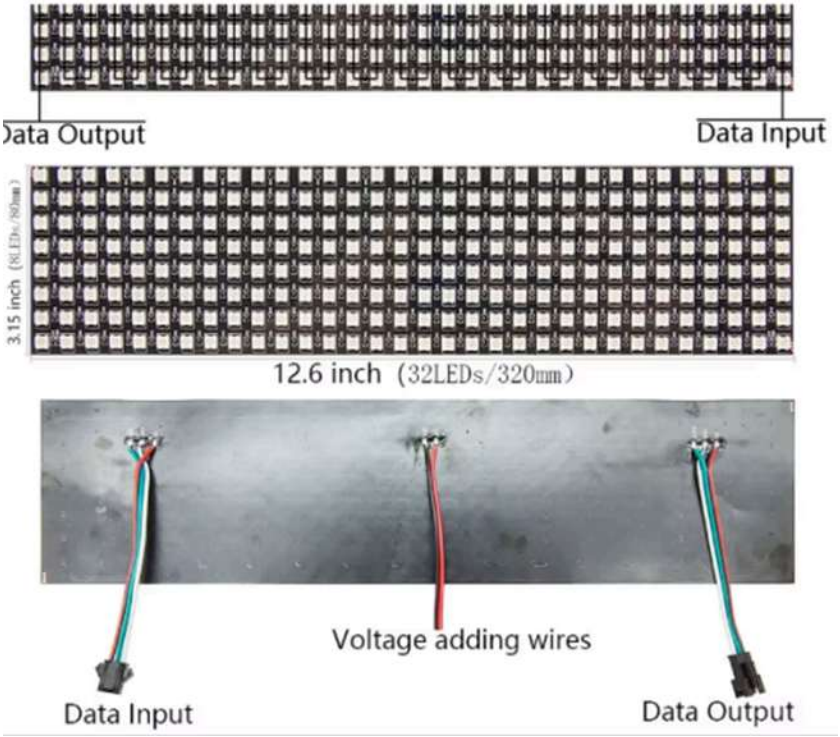
Dit artikel is ook gepubliceerd in PC-Active 346 (feb 2026)

Inleiding
Na een recente renovatie van mijn woonkamer ontstond de behoefte aan een nieuwe klok. De oude, klassieke wijzerklok met glasplaat en batterijen was gesneuveld en niet meer te repareren. Dat moment vormde het begin van een nieuw idee: een klok die niet alleen de tijd weergeeft, maar ook mijn passie voor elektronica weerspiegelt. Als hobbyist wilde ik een klok bouwen met een matrixdisplay van kleurrijke, programmeerbare leds. Het doel was om de tijd weer te geven in een eenvoudige 24-uurs notatie. Maar al snel groeide het idee verder: waarom zou de klok niet ook andere nuttige informatie kunnen tonen?

Naast het tonen van de tijd wilde ik data uit mijn domotica database kunnen visualiseren in de vorm van meldingen. Denk aan meldingen zoals: Wanneer het verstandig is om mijn opgewekte zonne-energie direct te gebruiken in plaats van deze ongemerkt terug te leveren aan de energieleverancier of een waarschuwing geven wanneer iemand aanbelt, of een weergave van het momentane energieverbruik. De klok moet dus ook tekst weergeven.
Zo ontstond het concept van een multifunctionele kamerklok: een combinatie van design, techniek en slimme domotica toepassing.

Uitgangspunten van het ontwerp
Het display moet voldoende pixels bevatten om tekst te kunnen weergeven. De pixels moeten in meerdere kleuren kunnen oplichten om bijvoorbeeld visueel onderscheid te kunnen maken tussen een melding en een zeer dringende melding (alarm). Bovendien kan door het aanpassen van kleuren de klok makkelijk in het interieur van de kamer worden opgenomen.
Als basis materiaal gebruik ik het liefst hout, eventueel daaraan toegevoegd 3D geprinte onderdelen zoals bijvoorbeeld de licht diffusor.
Verder is het wenselijk om de helderheid van het display automatisch aan te passen aan de hoeveelheid omgevingslicht. Het display hoeft in het donker niet de gehele kamer te verlichten.

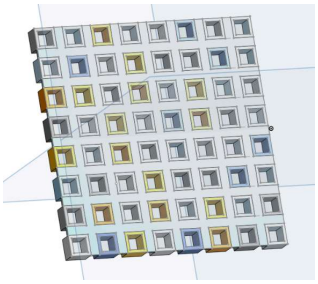
Al deze wensen leiden ertoe om ws2812B leds te gaan gebruiken. Deze leds zijn in allerlei uitvoeringsvormen te krijgen. Ik heb gekozen voor een ’12.6 inch matje’ waarop 8×32 leds zijn gemonteerd. Twee matjes in serie geschakeld geeft een matrix display van 8×64 = 512 individueel aanstuurbare pixels.



Het gekozen tekstfont en dus de karakters moet passen in een raster van maximaal 8×8 pixels.
Gelukkig is er een ruime keuze aan fonts die speciaal ontworpen zijn voor matrix display toepassingen, bijvoorbeeld een font van 5×8 pixels. Het font geeft met een beperkt aantal pixels een goed leesbare tekst en cijfers, inclusief hoofd- en kleine letters.

Belangrijke eigenschap van de WS2812B led is de intense hoeveelheid licht van de led die soms bijna verblindend kan zijn. Het stroomverbruik kan dan zeker tot 60mA per led oplopen. De toegepaste voeding moet dit vermogen kunnen leveren aan 5 volt: 60mA x 512 = 30A ! Echter door niet alle leds tegelijk op volle sterkte aan te laten gaan is het stroomverbruik zeer te beperken (softwarematig) en kan worden volstaan met een voeding die enkele ampères kan leveren.

Een tweede mogelijkheid om intense lichtopbrengst van de leds in te dammen is het gebruik maken van een licht diffusor. Deze zorgt voor een betere verspreiding van het licht uit een enkele led en dempt daarmee de lichtopbrengst. Een blad papier heeft reeds effect (zie figuur 4), maar een 3D geprint raster voor de leds geeft ook al een goede diffuse uitstraling van het licht uit de leds.



De elektronica in de klok meet het omgevingslicht. De stroom door een licht gevoelige weerstand (LDR) wordt gedigitaliseerd en kan wordt gebruikt om de intensiteit van de leds in te stellen. In een donkere omgeving wordt de helderheid verlaagd terwijl bij veel omgevingslicht de intensiteit voor betere zichtbaarheid wordt verhoogd.

Om energie te besparen, is een PIR sensor ingebouwd die beweging detecteert in de buurt van de klok. Wanneer beweging wordt gedetecteerd wordt het display van stroom voorzien zolang een ingestelde tijd niet is verstreken en er opnieuw beweging wordt gedetecteerd. Anders dooft het display. Er is een (blauwe) status led ingebouwd om af en toe een signaal te krijgen ter controle van het goed functioneren van de hardware van de klok.

De klok is meer dan een tijdweergave: ze fungeert als een slim display voor meldingen en informatie uit het huis. De meldingen komen uit een database, waarin, in mijn geval. allerlei informatie van mijn huis wordt verzameld, zoals energieverbruik, status van de zonnepanelen, temperaturen in huis, weerstation data en deurbel activiteit.
Deze database is gehost op mijn persoonlijke website en is toegankelijk via het internet. WiFi-connectiviteit is dus essentieel voor de klok. De klok haalt ook de actuele tijd van het internet via een NTP-server en is daardoor perfect gesynchroniseerd. Er zijn geen afwijkingen meer door lege batterijen of stroomuitval. Hoewel de klok is opgebouwd uit verschillende functionele hardware modules zoals het display, de LDR, PIR sensor, WiFi-module en voeding, ligt in dit artikel de nadruk op de software die deze onderdelen aanstuurt en met elkaar laat samenwerken.
Voor de ondersteuning, de vele ideeën en voorbeelden, een actieve community en de eenvoudige integratie met libraries, heeft de programmeertaal ‘Python’ de voorkeur.

De klok is meer dan een tijdweergave: ze fungeert als een slim display voor meldingen en informatie uit het huis. De meldingen komen uit een database, waarin, in mijn geval. allerlei informatie van mijn huis wordt verzameld, zoals energieverbruik, status van de zonnepanelen, temperaturen in huis, weerstation data en deurbel activiteit.
Deze database is gehost op mijn persoonlijke website en is toegankelijk via het internet. WiFi-connectiviteit is dus essentieel voor de klok. De klok haalt ook de actuele tijd van het internet via een NTP-server en is daardoor perfect gesynchroniseerd. Er zijn geen afwijkingen meer door lege batterijen of stroomuitval. Hoewel de klok is opgebouwd uit verschillende functionele hardware modules zoals het display, de LDR, PIR sensor, WiFi-module en voeding, ligt in dit artikel de nadruk op de software die deze onderdelen aanstuurt en met elkaar laat samenwerken.
Voor de ondersteuning, de vele ideeën en voorbeelden, een actieve community en de eenvoudige integratie met libraries, heeft de programmeertaal ‘Python’ de voorkeur.

De klok is meer dan een tijdweergave: ze fungeert als een slim display voor meldingen en informatie uit het huis. De meldingen komen uit een database, waarin, in mijn geval. allerlei informatie van mijn huis wordt verzameld, zoals energieverbruik, status van de zonnepanelen, temperaturen in huis, weerstation data en deurbel activiteit.
Deze database is gehost op mijn persoonlijke website en is toegankelijk via het internet. WiFi-connectiviteit is dus essentieel voor de klok. De klok haalt ook de actuele tijd van het internet via een NTP-server en is

daardoor perfect gesynchroniseerd. Er zijn geen afwijkingen meer door lege batterijen of stroomuitval. Hoewel de klok is opgebouwd uit verschillende functionele hardware modules zoals het display, de LDR, PIR sensor, WiFi-module en voeding, ligt in dit artikel de nadruk op de software die deze onderdelen aanstuurt en met elkaar laat samenwerken.
Voor de ondersteuning, de vele ideeën en voorbeelden, een actieve community en de eenvoudige integratie met libraries, heeft de programmeertaal ‘Python’ de voorkeur.

De klok is meer dan een tijdweergave: ze fungeert als een slim display voor meldingen en informatie uit het huis. De meldingen komen uit een database, waarin, in mijn geval. allerlei informatie van mijn huis wordt verzameld, zoals energieverbruik, status van de zonnepanelen, temperaturen in huis, weerstation data en deurbel activiteit.
Deze database is gehost op mijn persoonlijke website en is toegankelijk via het internet. WiFi-connectiviteit is dus essentieel voor de klok. De klok haalt ook de actuele tijd van het internet via een NTP-server en is
daardoor perfect gesynchroniseerd. Er zijn geen afwijkingen meer door lege batterijen of stroomuitval. Hoewel de klok is opgebouwd uit verschillende functionele hardware modules zoals het display, de LDR, PIR sensor, WiFi-module en voeding, ligt in dit artikel de nadruk op de software die deze onderdelen aanstuurt en met elkaar laat samenwerken.
Voor de ondersteuning, de vele ideeën en voorbeelden, een actieve community en de eenvoudige integratie met libraries, heeft de programmeertaal ‘Python’ de voorkeur.

Voor de ondersteuning, de vele ideeën en voorbeelden, een actieve community en de eenvoudige integratie met libraries, heeft de programmeertaal ‘Python’ de voorkeur.

Voor mijn eerste versie van de klok heb ik een Raspberry Pi Pico toegepast. Deze heeft echter geen real-time klok (RTC). Een extra tijdmodule zoals een DS3231 moet voor registratie en onthouden van de tijd worden gebruikt. Een ander nadeel van een Pico is het feit dat het niet mogelijk is om de kamerklok zonder meer op afstand en in headless mode te bedienen.
Daarom is er voor dit project gekozen voor een goedkope Raspberry Pi Zero W met Wifi toegang. Een Raspberry Pi 4 of 5 zou qua prestaties overkill zijn voor dit specifieke project, en bovendien duurder en energie-intensiever.

De hardware
Het maken van een overzichtelijk schema met behulp van AI-hulpmiddelen bleek helaas geen succes en de gegenereerde resultaten waren onbruikbaar voor dit specifieke project. Daarom heb ik gekozen voor een meer praktische aanpak: het schema is opgesteld met behulp van de Fritzing tool en een aanvullend tekenprogramma.

Het onderstaande schema toont de volledige opstelling van de elektronica en componenten in een breadboard-overzicht. Hierin zijn opgenomen:

De RPi Zero W
De WS2812B led matrix
De LDR voor lichtmeting
De PIR sensor voor aanwezigheidsdetectie
De voeding en spanningsregeling
De PCF8591

GPIO2 (pin3) I2C SDA: pin 1 +3.3V
GPIO3 (pin5) I2C SCL: pin 2 +5V
GPIO17 (pin 11) PIR: pin 39 GND
GPIO18 (pin 12): matrix display

Beschrijving hardware:
Omdat de RPi Zero W geen ingebouwde A/D (analoog naar digitaal) of D/A (digitaal naar analoog) converter heeft, is gekozen voor een externe oplossing: de PCF8591, een 8-bits converter die communiceert via het I²C-protocol.De I²C-bus bestaat de lijn SDA (data, geel) en SCL (klok, blauw). De PCF8591 is via I²C-adres 00×48 aan te spreken in de software.

D/A conversie voor status-led
Via de uitgang A0 van de PCF8591 wordt een spanning aangeboden aan de status-led, via weerstand R2. Hierdoor kan de helderheid en knipperfrequentie van de led softwarematig worden aangepast, afhankelijk van de status of meldingen.

A/D conversie voor lichtmeting
De LDR levert een analoge spanning die via ingang A1 wordt omgezet naar een digitaal signaal.
Dit signaal wordt via I²C aangeboden aan de Raspberry Pi. De software leest deze waarde uit als een maat voor het omgevingslicht. Een hoge waarde = weinig licht (donkere omgeving) en een lage waarde = veel licht (heldere omgeving).

Een PIR sensor registreert beweging in de omgeving op basis van IR reflectie. Het uitgangssignaal wordt hoog gemaakt wanneer beweging wordt gedetecteerd. Vervolgens blijft het signaal een in te stellen tijd actief waarna het signaal weer nul wordt. De software in de RPi leest het signaal op de GPIO pin17 (pin 11).

Omdat de pinnen van de I/O poorten van de Rpi 3.3 volt compatible zijn en die van PIR sensor en van de PCF op 5 volt werken, is er gebruik gemaakt van een level convertor. Hierdoor worden de verschillende spanningen toch goed in- en uitgelezen.
De level convertor wordt gevoed met zowel 3.3 volt en 5 volt van de Rpi.De level conversie is ook nodig voor de datalijn van het neopixel matrix display. Hiervoor wordt de uitgang GPIO18 (pin12 van de Rpi) gebruikt die 3.3 volt bedraagt. De neopixels willen graag 5 volt!

Als voeding is een *LRS 50-5* toegepast. Deze levert 5 volt aan de neopixels en aan de Rpi.
De 3.3 volt wordt geleverd door de Rpi.

Beschrijving software:
Het aansturen van neopixels is in veel bibliotheken beschreven en voor meerdere programmeeromgevingen geschikt gemaakt. De basis van de software in deze klok is een specifieke Python-library, gebaseerd op de bekende NeoPixel-bibliotheken van Adafruit [2]. Ieder Python programma begint met het inlezen van de toegepaste bibliotheken. Voor de kamerklok zijn dat:

```
import os
```

```
import time
import RPi.GPIO as GPIO
import smbus
import requests
import json
import adafruit_framebuf
from neopixel_matrix import NeoPixelMatrix, Color
```

Voor algemene taken worden os en time geïmporteerd. De wat oudere bibliotheek Rpi. GPIO en smbus worden gebruikt om eenvoudig en overzichtelijk de I2C bus tekunnen gebruiken voor de PCF8591. De requests en json bibliotheek worden voor de communicatie met de database op de website gebruikt.
Dan blijft de adafruit_framebuf en de neopixel_matrix bibliotheek over die samen worden toegepast. Deze zijn te vinden in de github repository: [2 , 3]. De bibliotheken zijn geschreven voor circuitpython. Voor het gebruik van python3 zijn enkele kleine aanpassingen in de bibliotheek noodzakelijk die ik eenvoudig kon oplossen met hulp van AI (microsoft copilot).

Het fontbestand is lastiger te vinden echter via de link [4] is het font uiteindelijk geplaatst in de directory waar alle programma's en bibliotheken zijn geplaatst:

```
font_path = "/home/pi/Progs/font5x8.bin"
```

Het matrix display is als volgt gedefinieerd:

```
np_matrix = NeoPixelMatrix(pin=18, width=64, height=8, brightness=0.01, bg_color=Color.BLACK)
```

Pin=18 is de GPIO pin op de RPi zero (fysieke pin 12), width en height is het aantal pixels in de matrix display voor de breedte en hoogte, de brightness (dus energieverbruik) kan tussen 0 en 1 worden ingesteld en is met 0.01 wel tot een minimum ingesteld. Door black te kiezen als pixel kleur geeft dit aan dat de pixel gedoofd is.

Om iets op het matrix display te zetten wordt de np_print()' procedure gebruikt:

```
#————————————————— melding printen
def np_mprint (s):
    np_matrix.brightness = 0.01 # 0.05 is minimale brithness voor groen!
    np_matrix.text(s, 0, 0, color=Color.WHITE, center = False)
    return
```

Voordat de leds worden aangestuurd wordt eerst de brightness ingesteld op een (minimaal) niveau:

```
np_matrix.brightness = 0.01 # 0.05 is minimale brithness voor groen!
np_matrix.text(s, 0, 0, color=Color.WHITE, center = False)
```

Vervolgens kan de tekst s in kleur WHITE geprint worden. Met center= False begint het eerste karakter op positie (0,0).

Nu kan dus al snel de tijd en tekst worden geprint. Voor de definitie van de tijd is een volgende procedure gemaakt. Afhankelijk van de keuze van f wordt de tijd weergegeven in een bepaald format:

```
# Haal de afzonderlijke tijdcomponenten op uit localtime()
def real_time(f):
    year = time.localtime()[0]
    month = time.localtime()[1]
```

```
    day = time.localtime()[2]
    dw  = weekdag[time.localtime()[6]]
    hours =  time.localtime()[3]
    minutes =  time.localtime()[4]
    seconds =  time.localtime()[5]

# Formatteer de tijd als een string en maak een keuze in het format wat je wilt zien
if f==0:
        return “{:02d}-{:02d}-{:02d} {:02d}:{:02d}:{:02d}”.format(year, month, day, hours, minutes, seconds)
elif (f==1):
        return “{:s} {:02d}:{:02d}”.format(dw, hours, minutes)
elif (f==2):
        return “{:02d}:{:02d}”.format(hours, minutes)
elif (f==3):
return “{:02}:{:02d}:{:02d}”.format(hours, minutes, seconds)
else:
        return
```

Deze procedure levert dus de tekst ‘2025-08-01 12:01:20’ indien f=0 en indien f=3 levert dan de tekst bijvoorbeeld ‘16:32:35’ op. In dit voorbeeld is dit op de klok zichtbaar:

Structuur van het programma

Het python programma doorloopt continue de volgende loop:

```
while True:
  # 1. snelle loop
time.sleep(0.5)  # Matig de belasting van de CPU
# 2. display aan of uit
if presence_detected and ((time.time() – last_detected_time ) >= min_on_time):
    last_detected_time=time.time()
    if not GPIO.input(pir_pin):      # Geen bewegingmeer
        presence_detected = False
        # Display uitschakelen
       display_on = False
       VerzamelData(data_lijst)
       np_matrix.clear()
# 3. Klokfunctie
If time.time() – t0 >= 10:  # Elke 10 seconde uitvoeren
      t0 = time.time()
      np_dprint(data_lijst[0][“tijd”])
 # 4. Database melding
if time.time() – t1 >= 60:  # Elke 60 seconde uitvoeren
      t1 = time.time()
      meldingen = filter(GetLastDataUitDatabase())
      np_print (data_lijst[0][“melding”])
```

Voor deze structuur kan worden gekozen wanneer niet tijd kritische processen plaats vinden (op mille seconden basis). In de snelle loop van de lus kunnen alsnog processen die bijvoorbeeld op mS basis moeten plaatsvinden aangeroepen worden. Vervolgens wordt eens in de min_on_tijd (minimum aan tijd van de display) gekeken of er nog iemand aanwezig is en of dus het display moet aanblijven of mag doven (display_on = False).

Elke 10 seconde wordt de tijd op het display ververst. De tijd wordt ook bijgehouden in de datalijst van het programma (VerzamelData(data_lijst)). In deze lijst worden ook nog andere zaken bijgehouden zoals meldingen en lichtmetingen.

Dan wordt elke 60 seconde gecontroleerd of er een melding van een item uit de database moet worden geprint op het display.
De procedure GetLastDataUitDatabase() haalt de laatst ingevoerde data van de sensoren uit huis uit de database op en kijkt of een grenswaarde wordt overschreden die moet worden gemeld (bv. wordt er elektriciteit geproduceerd door de zonnepanelen, of is het CO2 gehalte in de woonkamer te hoog of is het buiten erg koud).

Het ophalen van data uit een (mysql) database gaat relatief eenvoudig via een request:

```
# haal data van de website , bijvoorbeeld url=https://www.mysite.nl/lastin.php
response = requests.get(url)
if response.status_code == 200:
    data = response.json()  # JSON omzetten naar een Python-dictionary
```

Op mijn website met de domotica database draait een php script lastin.php met een sql-query dat laatst ingevoerde data uit de verschillende tabellen haalt. Het resultaat wordt geleverd aan de response van het python script. Indien de status van de data ‘oke’ is dan wordt van de response een goed leesbaar json formaat gemaakt.

De melding wordt vervolgens met witte karakters op het display geprint. Er zijn enkele figuren weergegeven met verschillende meldingen (fig. 5a,b,c,d)

```
np_matrix.text(melding 0, 0, color=Color.WHITE, center = False)
```

Resultaten
Het hier getoonde ontwerp van een kamerklok is makkelijk na te bouwen. Het volledige programma is te downloaden via deze link [5] en is goed gedokumenteerd.
Het ontwerp is makkelijk uit te breiden met extra meldingen of alarmeringen met geluid dankzij het gebruik van de Rpi, die dit volledig ondersteunt.

Links

[1]: Adafruit_CircuitPython_framebuf : https://github.com/adafruit/Adafruit_CircuitPython_framebuf/tree/main/examples

[2]: micropython_neopixel_matrix: https://github.com/leosok/micropython_neopixel_matrix

[3] font5x8.bin bestand: https://github.com/adafruit/micropython-adafruit-bitmap-font?tab=readme-ov-file

[4] neopixels on raspbery Pi: https://learn.adafruit.com/neopixels-on-raspberry-pi/raspberry-pi-wiring

[5] download link software

1 reactie

zotten
29 januari 2026

Leuk project. De nadruk ligt wel op het programmeren. De toegepaste elektronica is relatief eenvoudig. Dit project kan makkelijk worden nagebouwd en worden uitgebreid met eigen ideeën.