

4 Lcd-Display (I2C)

4.1 Einddoel

Deze opdracht heeft meerdere eindopdrachten.

- Laat een getal van 100 naar 0 aflopen, en erna weer oplopen.
- Met 4 knoppen laat je een pijl bewegen over een lcd-scherm.

Als je klaar bent met de opdrachten lever je in It's learning in:

- De definitieve python code waar de verschillende opdrachten in verschillende functies staan.
- Een foto van de opstelling waarop duidelijk te zien is hoe alles in aangesloten.
- Een mp4-filmpje met een demo van je programma.

Als dit gedaan is laat je de opdracht zien aan de docent zodat deze de opdracht kan goedkeuren.

4.2 Kennis

Voor deze opdracht heb je kennis nodig van microPython.

4.3 Benodigdheden

Voor deze opdracht heb je de onderstaande materialen nodig. Controleer aan het begin of al deze spullen aanwezig zijn. Bij het opruimen dien je weer te controleren of alles aanwezig is. Indien er iets defect is geraakt moet je de docent op de hoogte brengen.

- Raspberry Pi Pico – H
- UBS-usb mini kabel
- 4 drukknoppen
- Pul down weerstanden
- LCD 1602 met hd44780 controller
-

4.4 Opdracht

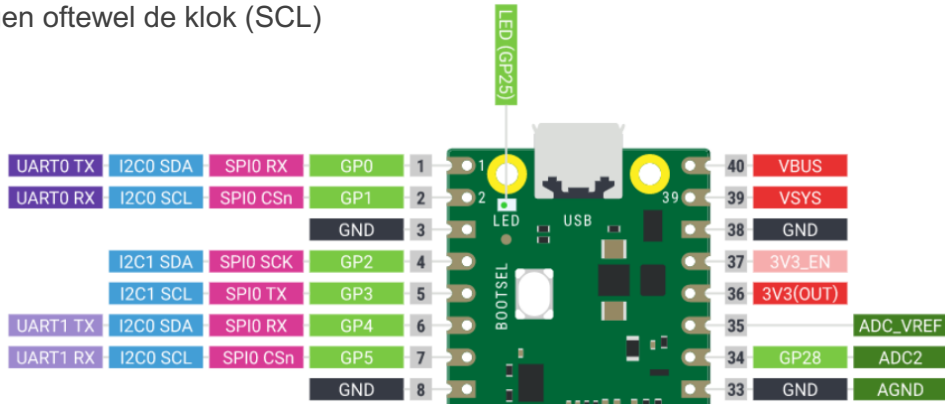


4.4.1 Aansluiten

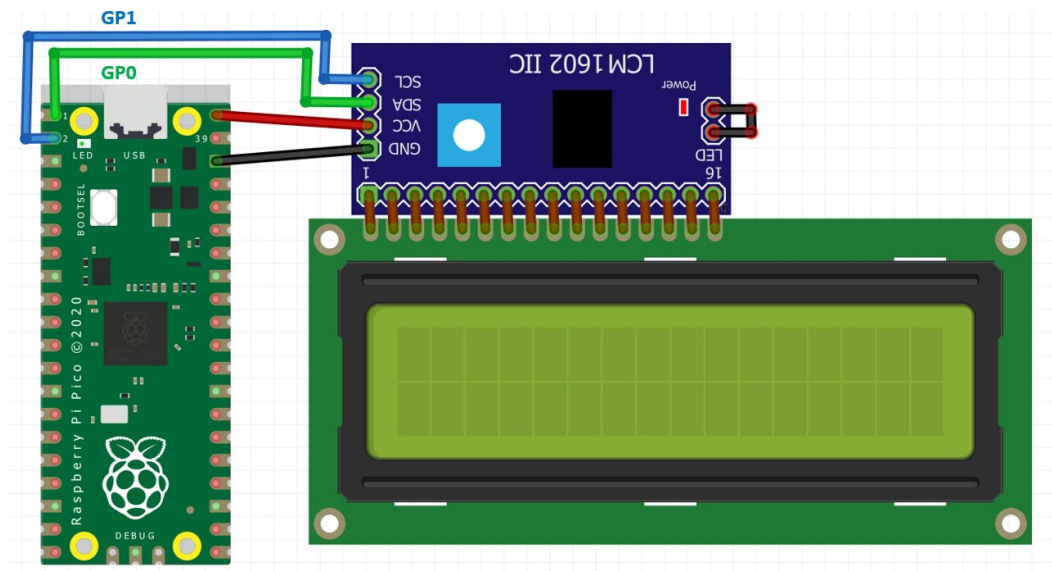
Als je een lcd-scherm bekijkt dan kan je zien dat er 16 aansluitpinnen aanzitten om de lcd aan te sturen. Er zijn 8 pinnen die op een GP-pin op de Pico aangesloten moeten worden. Dit is niet zo wenselijk omdat je vaak meerdere pinnen nodig hebt om het een en ander te laten werken.

Hiervoor is op de achterkant van de LCD1602 een hd44780 gesoldeerd. Dit component zorgt ervoor dat je de lcd-display via i2c kan benaderen. I2c is een serieel communicatieprotocol waarmee ieder een apparaat naast de plus en de GND maar 2 GP pinnen nodig heeft. Dit moeten we GP pinnen zijn die I2C ondersteunen, zoals GP0 en 1, of GP2 en 3.

Over de ene pin wordt de data gestuurd (SDA) en de andere pin geeft aan wat de snelheid is van de veranderingen oftewel de klok (SCL)



Als eerste gaan we het display aansluiten op de pico zoals hieronder is weergegeven.



Seriële communicatie werkt hetzelfde als een klas vol studenten. Als de docent iets tegen een student wil zeggen dat zal hij eerst de naam roepen en dan de boodschap vertellen.

Ieder apparaat dat aangesloten zit via seriële communicatie moet dus een naam (een adres) hebben. Als we de LCD hebben aangesloten dan kunnen we de naam van het lcd opvragen met de onderstaande code. Sla deze code op in een nieuw bestand met de naam **i2c_scanner.py**, want je zult hem ook nodig hebben bij andere opdrachten.

```

1 from machine import I2C, Pin
2
3 i2c = I2C(id = 0, sda=Pin(0), scl=Pin(1))
4 devices = i2c.scan()
5
6 print('Scan i2c bus...')
7 if len(devices) == 0:
8     print("No i2c device !")
9 else:
10    print('i2c devices found:',len(devices))
11
12    for device in devices:
13        print("Decimal address: ",device," | Hexa address: ",hex(device))
14

```

Voor het adres is 8 bits gereserveerd, dat betekent dat je $2^8 = 512$ verschillende apparaten aan die twee draadjes kan aansluiten. Deze apparaten moeten dan wel allemaal een andere naam hebben, anders geven er twee tegelijk antwoord als er iets gevraagd wordt. Het hoeven ook niet allemaal dezelfde apparaten te zijn. Als een apparaat aangesproken wordt weten zowel ontvanger als verzender wat ze met de instructies bedoelen.

Nu we het adres van de display weten kunnen we instructies naar de display sturen. Hiervoor moeten we **eerst** van It's learning de volgende twee bestanden op de pico zetten.

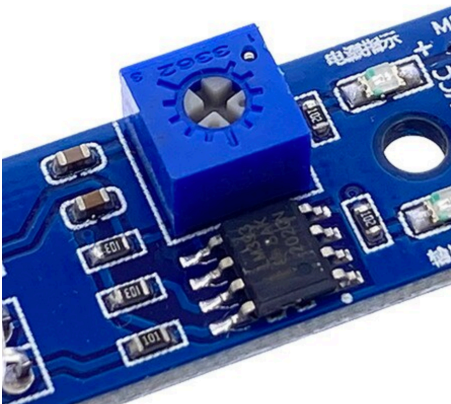
- lcd_api.py
- pico_i2c_lcd.py

Deze twee bestanden helpen ons bij het aansturen van de display. Hieronder staat een nieuw bestand met een voorbeeld om de tekst op het scherm te tonen.

```
1 from machine import I2C, Pin
2 from pico_i2c_lcd import I2cLcd
3
4 NUMBER_OF_ROWS = 2
5 NUMBER_OF_COLUMNS = 16
6
7 i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
8 I2C_ADDR = 0x00 #Vul hier het adres in
9 lcd = I2cLcd(i2c, I2C_ADDR, NUMBER_OF_ROWS, NUMBER_OF_COLUMNS)
10
11 lcd.clear()
12 lcd.blink_cursor_on()
13 lcd.putstr("daVinci")
14 lcd.putstr("I2C Address:"+str(I2C_ADDR)+"\n")
15
16
```

Als je onderstaande fout ziet, dan is het ingevulde adres niet correct. Gebruik het programma ic_scanner.py nogmaals.

```
MPY: soft reboot
Traceback (most recent call last):
  File "<stdin>", line 9, in <module>
  File "pico_i2c_lcd.py", line 22, in init
OSError: [Errno 5] EIO
```



Als deze code uitgevoerd wordt zal er een tekst op het scherm getoond worden. Indien de tekst niet zichtbaar is kan je met de potentiometer aan de achterkant het contrast instellen.

Als de display geen back-light geeft dat ontbreekt waarschijnlijk de jumper op de hd44780.

4.4.2 Meer functionaliteit

In het bestand `lcd_api.py` staan de verschillende methoden die we kunnen uitvoeren op de display zoals:

- `clear`
- `show_cursor / hide_cursor`
- `blink_cursor_on / blink_cursor_off`
- `display_on / display_off`
- `backlight_on / backlight_off`
- `move_to`
- `putchar`
- `putstr`
- `custom_char`

Je kan herleiden aan de hand van de functies `putstr` die hieronder is weergegeven hoe die andere functies werken.

```
1 from machine import I2C, Pin
2 from pico_i2c_lcd import I2cLcd
3 from time import sleep # ADDED
4 NUMBER_OF_ROWS = 2
5 NUMBER_OF_COLUMNS = 16
6
7 i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
8 I2C_ADDR =
9 lcd = I2cLcd(i2c, I2C_ADDR, NUMBER_OF_ROWS, NUMBER_OF_COLUMNS)
10
11 lcd.clear()
12 lcd.blink_cursor_on()
13 lcd.putstr("daVinci")
14 lcd.putstr("I2C Address:"+str(I2C_ADDR)+"\n")
15
16 lcd.blink_cursor_off() # ADDED
17 lcd.clear() # ADDED
18 lcd.putstr("Backlight Test") # ADDED
19 for i in range(10): # ADDED
20     lcd.backlight_on() # ADDED
21     sleep(0.2) # ADDED
22     lcd.backlight_off() # ADDED
23     sleep(0.2) # ADDED
24 lcd.backlight_on() # ADDED
25 lcd.hide_cursor() # ADDED
```

4.4.3 Opdracht 1: Countdown

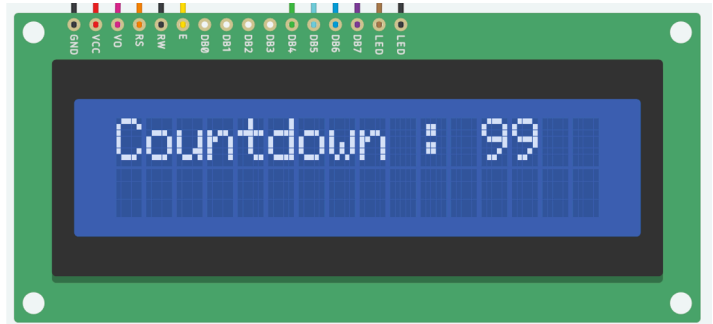
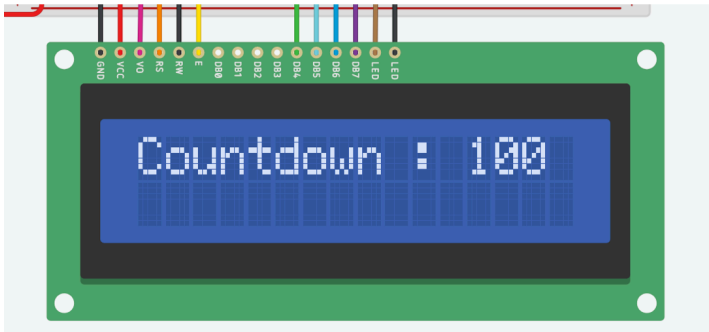
Als je op een positie een tekst wil neerzetten kan moet je het commando: `move_to` gebruiken voor de lcd.

```
move_to(cursor_x, cursor_y):  
    """Moves the cursor position to the indicated position. The cursor  
    position is zero based (i.e. cursor_x == 0 indicates first column).  
    """
```

Zet op het scherm de tekst neer "Countdown : "

Deze tekst wordt maar 1x naar het scherm gestuurd, en zit dus niet in een lus die steeds opnieuw getond wordt. Vervolgens laat je een teller van 100 naar 0 om de 4/10^e seconden aftellen. Deze teller staat ACHTER de dubbele punt van countdown. Zorg ervoor dat alles mooi uitgelijnd wordt. Je mag dus GEEN commando `clear` gebruiken.

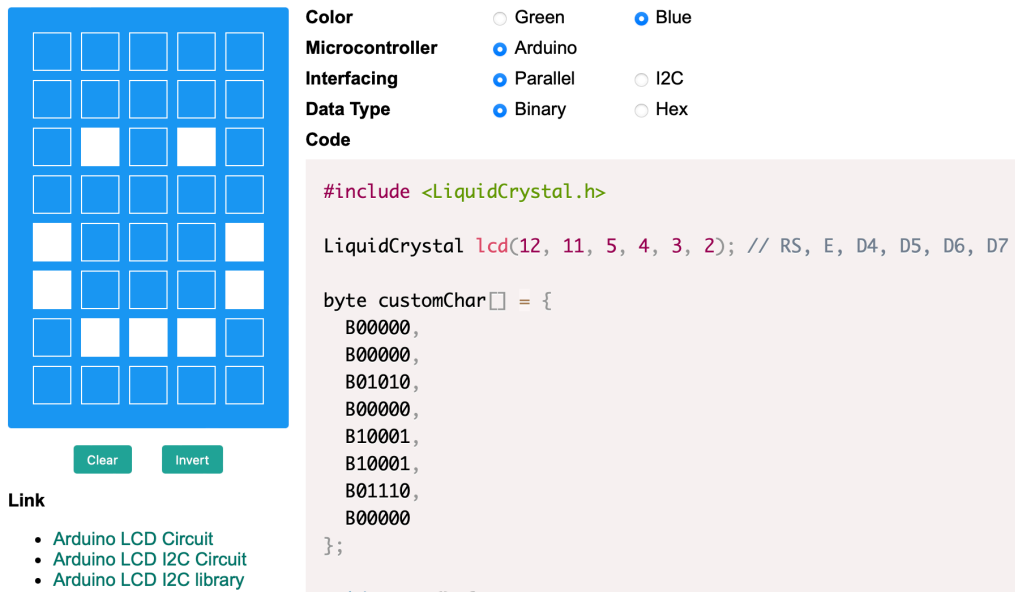
Op het scherm zal je dus het volgende zien:



4.4.4 Opdracht 2: Beweegbare karakters

Hieronder zie je een voorbeeld van een eigen gemaakt karakter dat je op het display kan gebruiken.

<https://maxpromer.github.io/LCD-Character-Creator/>



```
#include <LiquidCrystal.h>

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // RS, E, D4, D5, D6, D7

byte customChar[] = {
  B00000,
  B00000,
  B01010,
  B00000,
  B10001,
  B10001,
  B01110,
  B00000
};
```

Aan de rechterkant zien we de code voor de Arduino om dat karakter op het scherm te tonen. We werken met microPython, dus zullen we de code iets moeten aanpassen.

```
1 from machine import I2C, Pin
2 from pico_i2c_lcd import I2cLcd
3 from time import sleep # ADDED
4 NUMBER_OF_ROWS = 2
5 NUMBER_OF_COLUMNS = 16
6
7 i2c = I2C(0, sda=Pin(0), scl=Pin(1), freq=400000)
8 I2C_ADDR = <Vu1 hier het adres in> #i2c.scan()[0]
9 lcd = I2cLcd(i2c, I2C_ADDR, NUMBER_OF_ROWS, NUMBER_OF_COLUMNS)
10
11 SMILE = ( 0b00000,
12           0b00000,
13           0b01010,
14           0b00000,
15           0b10001,
16           0b10001,
17           0b01110,
18           0b00000)
19
20 lcd.clear()
21 lcd.hide_cursor()
22 lcd.custom_char(0, SMILE)
23 lcd.move_to(8, 1)
24 lcd.putchar(chr(0))
25
```

Kan je een pacman maken die van links naar rechts over het scherm gaat. Je maakt dan een pacman met mond open, en een met mond dicht. In de bovenstaande code op regel 22 zal dan gekopieerd moeten worden op de regel eronder. De 0 zal voor het tweede karakter een 1 worden, en op de huidige regel 24 kan met chr(1) het andere karakter getoond worden. Op deze manier kan je maximaal 8 (0 t/m 7) karakters maken. Let op dat je maar 1 maal de custom_char aanmaakt per plaatje. Je zet het plaatje eenmaal in het geheugen en erna kan je iedere keer dat karakter weer opvragen.

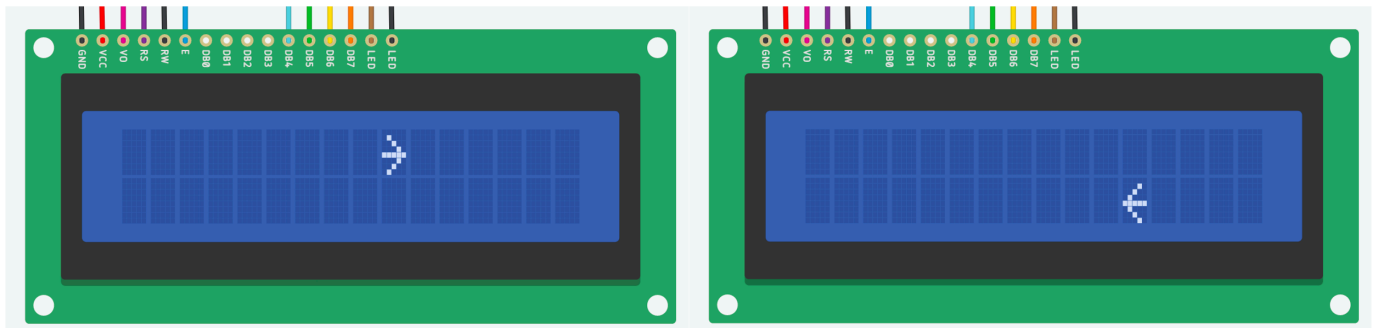
<https://www.google.com/logos/2010/pacman10-i.html>

Let op dat je GEEN clear gebruikt bij iedere beweging, maar maak de oude positie leeg en zet de pacman op de nieuwe positie.

4.4.5 Opdracht 3: Verplaatsen met knoppen

Voor de laatste opdracht sluit je 4 knoppen aan op de pico. De vier knoppen zijn achtereenvolgend naar boven, naar rechts, naar links, naar onder. Door op deze knoppen te drukken zal de pacman over het scherm verplaatsen. Als de pacman naar links gaat, kijkt deze naar links, als de pacman naar rechts gaat kijkt hij naar rechts. Naar onder en naar boven zal hij de oude kijkrichting hebben.

De pacman mag niet van het scherm aflopen. Op het scherm staan aan het begin allemaal puntjes. Laat zien dat je alle punten opeet met de pacman. Let op dat je eenmaal je scherm opbouwt met de puntjes. Daarna heb je een lus waar alles gebeurt. **In die lus mag GEEN clear staan.** Een lcd-scherm is heel traag en je moet alleen de wijzigingen op het scherm doorgeven. Dus waar de pacman was zet je een spatie, en op de nieuwe plek zet je de pacman.



4.4.6 Opdracht 4: Groter display

Er is ook een groter display aanwezig bij de opdracht. Sluit je pacman spel aan op het grotere display en kijk of het gelijk helemaal werkt. Als je goed hebt geprogrammeerd hoeft je alleen de `number_of_rows` en `number_of_columns` aan te passen om alles te laten werken.

Kan je ook nog een obstakel maken waar de pacman NIET overheen mag komen, als je dat wel doet ben je af. Je krijg dan "GAME OVER" te zien op het scherm.