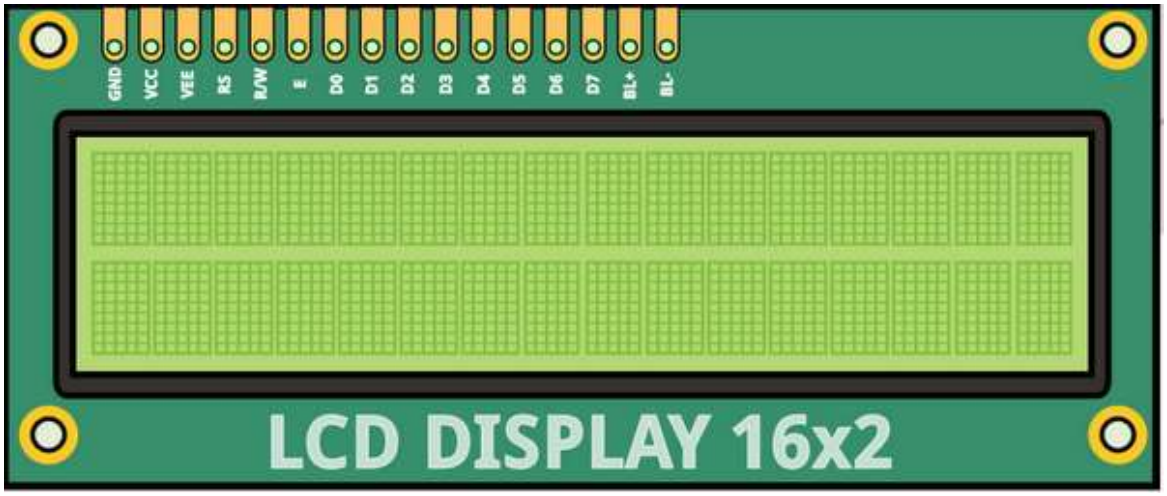
Lcd-display (2 lines x16 ch)

Doelstelling

- Communicatie opbouwen met een tekst - Lcd-display
- Gebruik van bit-operaties
- Datasheet van de controller leren implementeren.
- Timing diagram van de controller kunnen omzetten naar code.
- Werken met hexadecimale getallen.
- Protocol opbouwen in 8-bits
- Aanmaken en gebruiken van klassen in Python.

Inleiding



Er bestaan verschillende libraries om het Lcd-display aan te sturen, in dit labo zullen wij echter ZELF een klasse schrijven om het Lcd-display aan te sturen. Op deze manier leren we ook eens een parallel protocol. Het Lcd-display kan je zowel in vier- of in acht data bits aan sturen. Het voordeel van de 4 bits modus is dat je vier GPIO pinnen minder nodig hebt. Het nadeel is dat je wat meer programmeerwerk hebt en dat het aansturen van het Lcd-display wat trager wordt.

De controller van de Lcd-displays is een HD44780, deze controller wordt gebruikt om verschillende types van Lcd-displays aan te sturen o.a. 4*20 display, 2*16,…

Pinnummering Lcd-display

Pin 1 en 2 zijn de voeding van het Lcd-display. De voedingspinnen van de Raspberry Pi kunnen voldoende vermogen leveren om het Lcd-display te voeden. Maar we raden aan om toch via de externe power supply te voeden.

Pin 3 is de contrast pin van het Lcd-display. Aan pin 3 kunnen we een trimmer aansluiten zodat we de spanning aan pin 3 kunnen regelen tussen de 0V en de 5V (als de display op 5 volt gevoed word).

Pin 4,5 en 6 zijn de stuursignalen. Pin 5 de R/W wordt dikwijls rechtstreeks aan de massa verbonden omdat er meestal niks van het Lcd-display moet worden ingelezen.

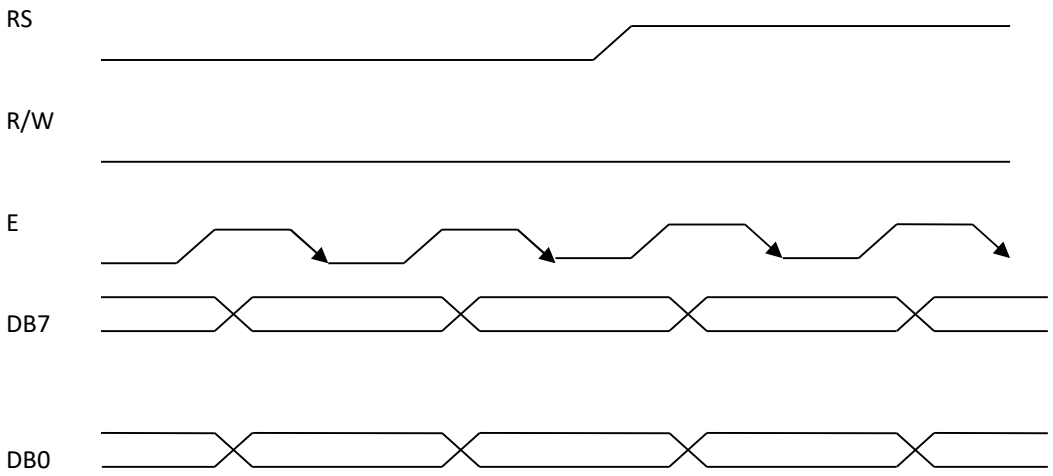
Vervolgens hebben we de 8 datalijnen. Waarvan we er vier of acht met de GPIO pinnen moeten verbinden, dit volgens de keuze 8 of 4 bits sturen.

Met pin 15 en 16 kunnen we de backlight laten oplichten. Dit zijn respectievelijk de anode en de kathode.

Pin number	Symbol	Level	Function
1	Vss	-	Power supply (GND)
2	Vcc	-	Power supply (+3.3V or 5V)
3	Vo	-	Contrast adjust
4	RS	0/1	0 = Instruction input 1 = Data input
5	R/W	0/1	0 = Write to LCD module 1 = Read from LCD module
6	E	0/1, Active:1->0	Enable signal
7	DB0	0/1	Data bus line 0 (LSB)
8	DB1	0/1	Data bus line 1
9	DB2	0/1	Data bus line 2
10	DB3	0/1	Data bus line 3
11	DB4	0/1	Data bus line 4
12	DB5	0/1	Data bus line 5
13	DB6	0/1	Data bus line 6
14	DB7	0/1	Data bus line 7 (MSB)
15	+Vcc	Only on some models	Backlight +Vcc, if available
16	Vss (0V)	Only on some models	Backlight GND, , if available

Aansturen Lcd-displaycontroller (HD44780/KS0066)

Onderstaande grafieken kunnen we terugvinden in de datasheets van het display. Het enable signaal E kunnen we vergelijken met het kloksignaal SCL van de SPI-bus of het shift register. Het enable signaal bepaalt wanneer de displaycontroller de data gaat verwerken. In tegenstelling tot de voorbije weken wordt de data nu verwerkt tijdens de **dalende** flank. Per klokpuls worden er nu ook 8 bits data verwerkt.



De VOLGORDE van de signalen is ook zeer belangrijk. We brengen eerst het enable signaal naar een hoog spanningsniveau, vervolgens zetten we de data klaar.

Als we nu het enable signaal terug naar 0V brengen wordt de data door de displaycontroller verwerkt. Via een wait (sleep) instructie moet de displaycontroller wat extra tijd krijgen om de data te verwerken.

Samengevat moet ons programma er als volgt uitzien :

=> E = hoog => Data klaarzetten => E = laag => wait => E = hoog => ….

Er zijn nu echter nog twee controle lijnen:

- Het controle signaal R/W bepaalt of we data lezen of schrijven naar het display. Een laag signaal is schrijven, een hoog signaal is lezen. Wij beperken ons tot het schrijven van data. Zodat deze lijn op laag mag blijven staan voor onze les.
- Tenslotte hebben we nog het register select signaal (RS), dat signaal bepaalt of er een instructie of een karakter gestuurd wordt. De cursor verplaatsen, de vorm van de cursor,… zijn voorbeelden van instructies. Als het RS signaal een waarde laag heeft sturen we instructies naar de displaycontroller, bij een hoog signaal sturen we karakters naar de displaycontroller.

Hexadecimaal rekenen (Per 4 bits) herhaling

Het grote voordeel van het hexadecimale rekenen is dat je bijna geen rekenwerk hebt bij het omzetten van een binair getal naar een hexadecimaal getal. Eén hexadecimaal cijfer komt immers overeen met 4 bits.

Number	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal	0	1	2	3	4	5	6	7

Number	8	9	10	11	12	13	14	15
Binary	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal	8	9	A	B	C	D	E	F

Circuit Globe

B.v.: 1111 0100 0011 1101

= 0x F 4 3 D

Protocol LCD display

Aangezien we ons beperken tot het schrijven naar het LCD display, blijft het signaal R/W altijd op een laag spanningsniveau staan en kunnen we die zonder GPIO pin rechtsreeks verbinden in onze schakeling.

We kunnen de volgende subroutines gaan programmeren :

```
def send_instruction(value):
    ...
    time.sleep(0.01)


def send_character(value):
    ...
    time.sleep(0.01)
```

Deze functies gaan het E en RS signaal voor hun rekening nemen en op hun beurt allebei gebruik maken van een functie om de 8 datapins op het juiste niveau te brengen. Op het einde van een instructie moeten we een kleine delay in bouwen.

```
def set_data_bits(value):
    ...
```

Dit moet een routine worden die de bits uit de byte haalt, maar in plaats van deze serieel door te sturen zetten we ze nu parallel klaar op de 8 GPIO pinnen van de databus

TIP: Zet je GPIO-pinnen voor de databus **in de juiste volgorde** in een list, dan kan je met 1 regel in een for-loop de pinnen klaarzetten! List-index en bit-nummer komen dan namelijk mooi overeen. Je zal uiteraard van bit operaties gebruik moeten maken om de waarde van die bit eruit te filteren.

Enkele basisinstructies

	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Clear display & Cursor home	0	0	0	0	0	0	0	0	0	1
Cursor home	0	0	0	0	0	0	0	0	1	*
Display on Sets On/Off of all display (D), cursor On/Off (C) and blink of cursor position character (B).	0	0	0	0	0	0	1	D I=ON 0=off	C I=ON 0=off	B I=ON 0=off
Function set Sets interface data length (DL), number of display line (N) and character font(F).	0	0	0	0	1	DL I=8bit 0=4bit	N I=2lines 0=1line	F 0:5x7 I:5x10	*	*
Sets DDRAM	0	0	1	DDRAM adres						

In de bovenstaande tabel vind je de belangrijkste instructies terug om het display te initialiseren. In de datasheets vind je de volledige lijst met instructies terug. Aan de eerste, meest beduidende bit herkent de controller de instructie, met de volgende bits kunnen we dan verschillende opties meegeven of een cursorpositie weergeven.

Om het display te initialiseren moeten we de volgende instructies aanroepen:

1) Function set
2) Display on
3) Clear display en cursor home.

Met de instructie “function set” moeten we het type van het display meegeven. Ons display beschikt over 2 lijnen en karakters van 5x7 pixels. De datalengte mag je instellen op 8 bit. Omdat we met deze instructie bepalen of we in vier bits of acht bits modus werken moet dit de eerste instructie zijn. Als we b.v. de “function set” willen instellen, herkent de controller de functie omdat we bit 5 (DB5) hoog zetten. Met DB2,3,4 kunnen we dan het type van ons display bepalen. Op de plaats van het sterretje mogen we om het even wat instellen.

Met de instructie “display on” kunnen we het display aan en uit schakelen, tevens kunnen we ook de cursor instellen. Eventueel kunnen we ook nog de cursor naar zijn home positie brengen en het display wissen.

Geheugen adressen van het Lcd-display

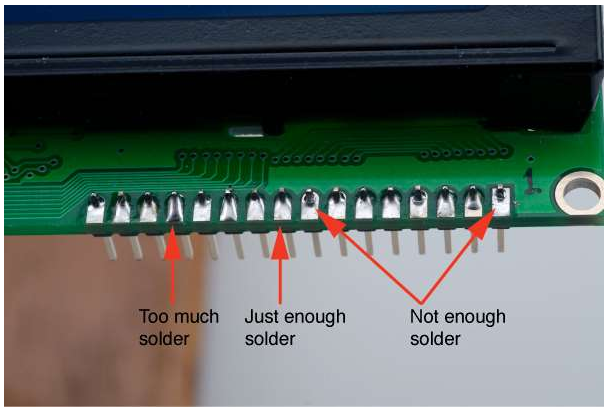
De controller van het Lcd-display kan ook een display aansturen van 4 x 20 karakters, daarom beschikt de controller ook over 40 geheugen adressen. De zichtbare geheugenlocaties van het display zijn de adressen 0x00 t.e.m. 0x0F voor de eerste lijn. De tweede lijn heeft de adressen 0x40 t.e.m. 0x4F

De adressen van de cursorposities kunnen we uit de onderstaande tabel halen.

DDRAM adres:

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21
40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61

Tip: bij deze DDRAM moet je data bit 7 toevoegen. Dat kan je makkelijk doen m.b.v. bitoperaties.

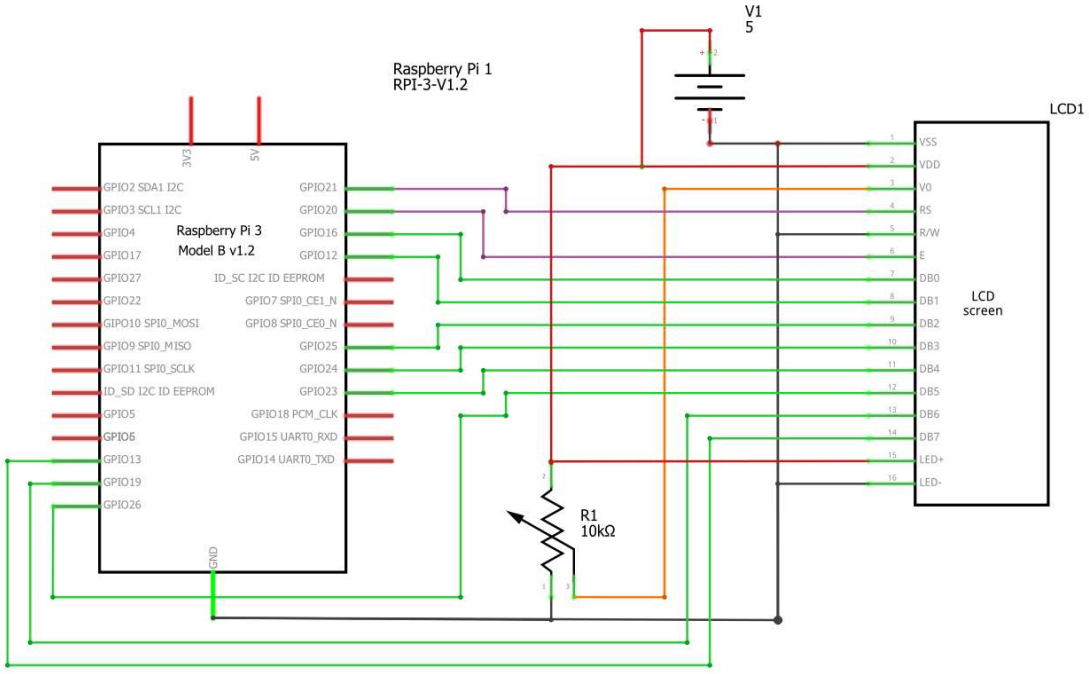


Opdracht 1: Solderen en testen

a) We zullen van start gaan om het display aan te sturen in de acht bits modus. Misschien zal je nog eerst de pin header moet vast solderen aan het Lcd-display. Bekijk aan welke kant je de korte kant van de pinnen plaatst. Het is best om deze te solderen, zodat de lange pinnen over zijn, om in het breadboard, of via kabels verbonden te worden. (kijk ook waar je de header soldeert : achteraan is gemakkelijker om later het LCD display in te bouwen, en je kan dit ook op het breadboard inpluggen)

b) Maak de schakeling voor 8 bits, we gebruiken ook een trimmer om het contrast te regelen. Gezien we veel pinnen nodig hebben, is het best om eens te kijken welke nog vrij zijn. Vb Gebruik GPIO pin 4 NIET als uw one-wire bus nog aanstaat . Die pin hebben we immers gebruikt als de one-wire bus. Wil je toch die pin gebruiken dan moet je de one-wire disabelen. Zet eventueel ook de SPI bus terug af in de raspi-config, zodat er meer pinnen vrij komen. De pinnen van het LCD display kunnen we sturen met 3V3, we hebben dus geen level shifter nodig.

c) Wel plaatsen we de VDD van het LCD display, voeding van de backlight van de LCD display en de trimmer op de 5V spanning van de externe voeding!.



d) Nu controleer/ meet je best eens het soldeerwerk. Maak daarvoor alle nodige aansluitingen op het breadboard, plaats alle stuursignalen en databits op een HOOG niveau (maak daarvoor een kleine programma) . Nu zou je op alle stuur-en datalijnen van het Lcd-display een 3V3 moeten METEN.

e) Draai aan uw potentiometer(trimmer) zodat er net op de bovenste lijn licht grijze hokjes tevoorschijn komen. Deze zullen straks verdwijnen als we het display juist initialiseren. Dit is de contrast regeling, die je

straks, als er tekst op staat kan optimaliseren, maar om te starten zijn licht grijze hokjes in de bovenste lijn ideaal. (het zou jammer zijn mocht het contrast helemaal toe staan, … dan zie je nooit iets…)

Opdracht 2: Aansturen in 8-bits modus

Schrijf de eerder vernoemde hulpfuncties - code:

a) Stop alle 8 de databits in de correcte volgorde in een list.

b) Initialiseer alle GPIO pinnen.

c) Schrijf de set_data_bits(value) functie. De parameter van de deze functie is een byte. Deze functie moet de value waarde parallel op de 8 databus lijnen plaatsen. Schrijf hiervoor een for loop die de parameter value bit per bit overloopt. Je checkt telkens de waarde van die bit om de overeenkomstige databus bitlijn hoog of laag te plaatsen. In deze functie gaan we de RS lijn NIET aanpassen.

d) Schrijf de send_instruction(value). Deze methode zet de RS lijn op het correcte niveau en maakt een klokpuls met de E lijn. (van hoog naar laag = inlezen). Deze methode roept ook de methode set_data_bits(value) aan.

e) Schrijf de send_character(value). Idem als daarnet maar met de RS lijn op data input.

f) Schrijf een init_LCD(). In deze definition roep je de instructie function set, display on en clear display & cursor home aan. Dus 3 instructies na elkaar doorsturen (in de juiste volgorde)

g) Nu kan je de code voor de eerste keer uittesten. Als het display aansloten is, zie je op de eerste rij (lichtjes) een zwarte rij balken. Als het display correct geïnitialiseerd is verdwijnt die rij balken en staat de cursor te knipperen.

h) Test de methode send_character(value) uit. Geef b.v. de ascii code 65 mee en de letter ‘A’ zou op het display moeten verschijnen.

i) Schrijf een def write_message(message). Deze methode overloopt het bericht en schrijft karakter per karakter naar het LCD display.

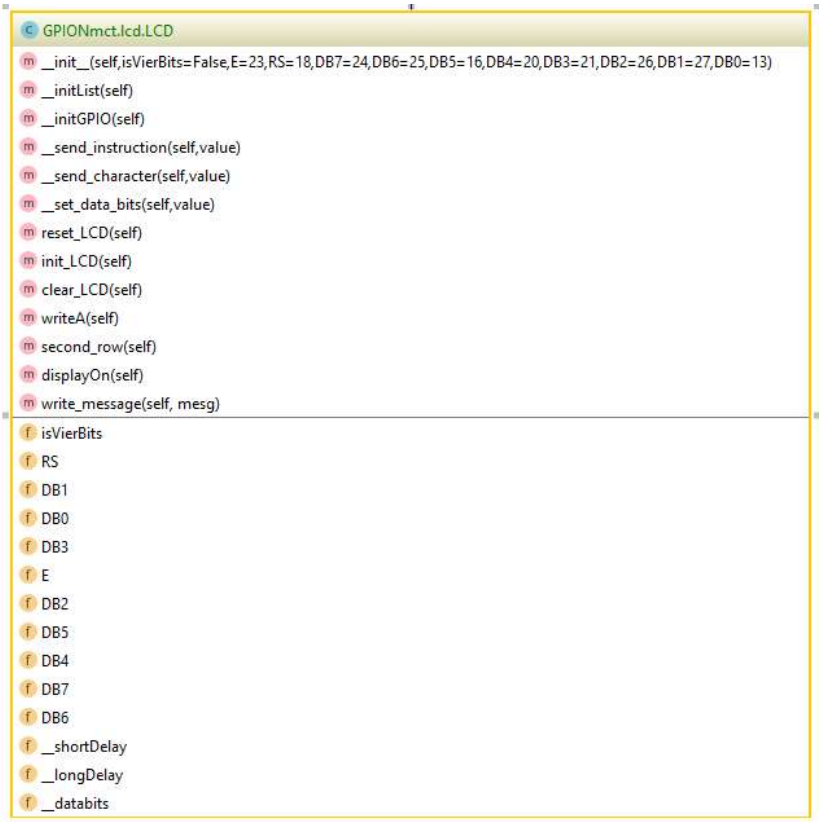
https://docs.python.org/3.5/library/functions.html

j) Vraag een input van de gebruiker. De ingegeven tekst druk je af op het display. Zorg er ook voor dat de tekst correct wordt weergeven als de tekst meer dan 16 karakters bedraagt. Los dit op door gebruik te maken van een instructie.

k) Laat de gebruiker een aantal opties van de cursor instellen. Los dit op door gebruik te maken van bitoperaties.

l) Maak de klasse LCD , een voorbeeld staat hiernaast, maar jullie moeten met een list werken ipv afzonderlijke pinnen… Het staat jullie vrij om deze aan te passen en uit te breiden (zoals bijvoorbeeld met een extra argument/parameter om de displayOn aan en af te zetten; of bijvoorbeeld ook een cursor aan en af te zetten…

m) Is de gevraagde tekst langer dan 32 karakters, dan laat je het display scrollen.



Opdracht 3: Aansturen in 4-bits modus

Probeer de LCD display nu in bits modus aan te sturen.

Gebruik daarvoor alleen de hoogte 4 bits.

De eerste instructie die je moet doorgeven is het in 4 bits plaatsen , maar moet je doorgeven in 8 bits,

daarna alle bytes (zowel instructies als data)in 2 keer doorgeven. (dus hoogste 4 bits data klaarzetten op hoogste 4 lijnen, E pulsen, onderste 4 bits klaarzetten op 4 hoogste lijnen, E pulsen

Opletten bij het stoppen. Het LCD display staat waarschijnlijk nog in 4 bits modus, dus wanneer je nogmaals runt, en je begint met een 8 bits instructie zal uw display niet juist reageren. Probeer daarom bij het afsluiten van uw programma eerst de display terug in 8 bits te zetten, met behulp van 4 bits instructies