LCD-Display

Arduino-experiment	130315-EN	*Trefwoorden*	Libraries, 16x2 display	
Versie	2018-06-21 / HS	*Niveau*	Gevorderd	*p.* 1/4

Wat je leert:

- Sluit een LCD-display aan op Arduino
- Functies van externe bibliotheken gebruiken

Displays met twee regels van 16 karakters bestaan in een groot aantal versies van vele verschillende leveranciers. Ze worden aangeduid als 16x2 of 1602 displays. Gelukkig zijn ze qua programmering vrijwel identiek. Je kunt er niet 100 % zeker van zijn dat een willekeurig gekozen display zich zal gedragen zoals hieronder beschreven – maar de kans is groot.

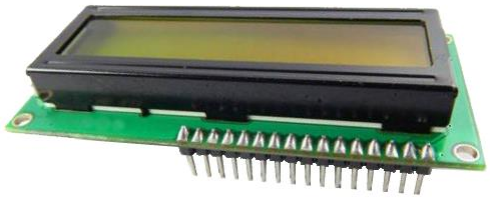
Hier wordt echter wel verondersteld dat het display gebruik maakt van parallelle communicatie via de 16 pins langs de bovenrand van de printplaat (zoals op de afbeelding in onderdeel 1 hieronder).

Als er een hulpplaat onder de hoofdprintplaat is gemonteerd (zoals hiernaast getoond), wordt er gebruik gemaakt van seriële communicatie. Dit vermindert het aantal pins als dat nodig is, maar is een beetje ingewikkelder en wordt hier niet behandeld.

1 – Soldeer een pin header op de display-printplaat

Indien het bord reeds voorzien is van een header (zoals hiernaast afgebeeld) – ga dan verder naar onderdeel 2.

Als je niet veel soldeertraining hebt is het mogelijk verstandig om iemand met meer ervaring om hulp te vragen. Het solderen van de header is een eenmalige handeling. In de toekomst wordt het display via het breadboard aangesloten.



De in-line headers worden meestal in rijen van 40 verkocht. Breek de rij voorzichtig af zodat je een rij van 16 pins overhoudt.
Het kortste eind van de pins gaat in de display-printplaat.

Het solderen is het gemakkelijkst met een assistent die de pinnen loodrecht op het bord kan houden tijdens het solderen. Begin met pin 1 en pin 16, en controleer of alles er goed uitziet voordat je de rest soldeert.

Als je geen assistent hebt kun je eventueel de lange uiteinden van de header pins in de breadboard plaatsen. De display-printplaat komt daar bovenop en laat de korte uiteinden van de pins erdoorheen. *Als je dit doet, let er dan op dat je niet te veel verhit tijdens het solderen – anders zal de breadboard smelten*. Soldeer eerst alle andere pins en pauzeer af en toe om de breadboard af te laten koelen.

2 – Sluit het display aan op Arduino

Zoals je zo meteen zult zien is er een hoge mate van flexibiliteit met betrekking tot welke van de *Arduino pins* je kunt gebruiken. Je kunt bijvoorbeeld kijken naar de oriëntatie van je opstelling – of naar welke pins je al hebt gereserveerd. Overweeg om een spreadsheet te gebruiken om de verbindingen bij te houden.

Op het display worden de volgende pins gebruikt:

1	VSS	0 V (GND)
2	VDD	+ 5 V
3	Vo	Contrast – zie hieronder
4	RS	Besturingssignaal
5	RW	0 V (GND)
6	E	Besturingssignaal
11	D4	Data bit 4
12	D5	Data bit 5
13	D6	Data bit 6
14	D7	Data bit 7
15	+LED	+ 5 V

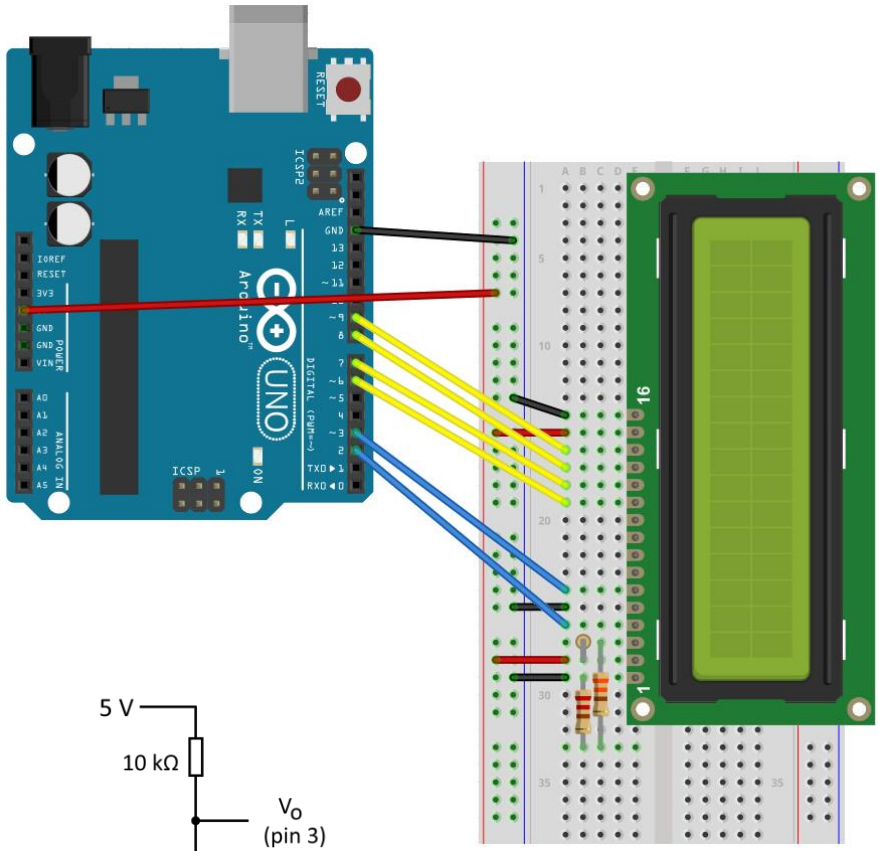
16 -LED 0 V (GND)

Zoals je ziet zijn pin 7 tot en met 10 weggelaten en worden sommige van de resterende pins gebruikt voor vaste spanningen. De daadwerkelijke communicatie met de Arduino verloopt via deze pins: RS, E, D4, D5, D6 en D7.

Pin 3 (contrast) is een beetje apart. Deze heeft een kleine DC-spanning nodig die we met behulp van een spanningsdeler zullen genereren. Soms wordt hier een trimmerpotentiometer gebruikt om het contrast aan te passen – maar voor nu zijn we tevreden met een vaste opstelling.

Nu zijn we klaar om de opstelling daadwerkelijk te bouwen.

Het is verstandig om de USB-aansluiting op de Arduino los te koppelen om spanningen in het circuit te vermijden tijdens het werken.



Net als voorheen zullen we de twee lange, buitenste rijen voor 5 V en GND gebruiken. Deze zijn verbonden zoals afgebeeld: display pin 2 en 15 aan 5 V, display pins 1, 5 en 16 aan GND.

De spanningsdeler op pin 3 wordt als volgt aangesloten:

Plaats een weerstand van 10 kΩ tussen display pins 2 en 3 – plaats deze verticaal (vanwege de korte afstand).

Tussen display pins 1 en 3 plaats je een weerstand van ca. 560 Ω. Op de tekening is te zien hoe twee weerstanden in serie geschakeld kunnen worden: 220 Ω + 330 Ω = 550 Ω. Deze waarde is OK als er geen 560 Ω-weerstanden beschikbaar zijn.

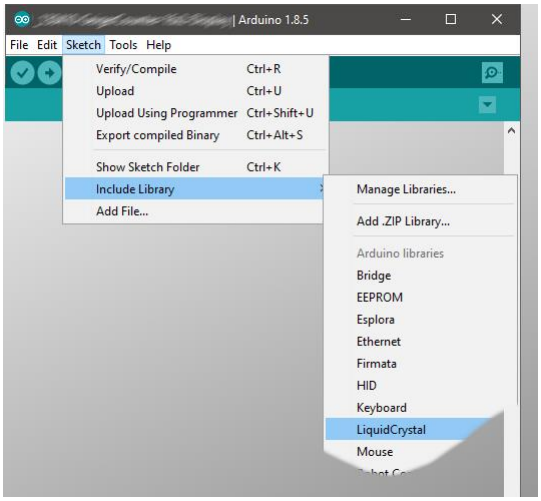
Sluit de signaaldraden aan volgens deze pintabel:

Arduino	Display	Functie
2	4	RS
3	6	E
6	11	D4
7	12	D5
8	13	D6
9	14	D7

Uitdaging 1

Maak het circuit zoals afgebeeld – en controleer grondig op fouten. Sluit de USB-kabel aan; de displayverlichting moet branden.

3 – De LiquidCrystal library



Tot nu toe hebben we een aantal van de ingebouwde functies van het Arduino systeem gebruikt. Bediening als LCD-display is een meer gespecialiseerde taak, die niet wordt gedekt door de systeemkern. Dit soort code wordt in zogenaamde *libraries* geplaatst.

Om een *library* te kunnen gebruiken, moet deze in de code worden opgenomen:
Start een nieuw programma. Gebruik het menu Sketch /*Include Library* en selecteer *LiquidCrystal*.

Dit zal de volgende regel aan de bovenkant van de programmacode toevoegen:

```
#include <LiquidCrystal.h>
```

(Als je van tevoren weet wat je moet schrijven, kun je deze regel handmatig toevoegen zonder de menu's te gebruiken.)

Vanaf hier lijkt de situatie op het gebruik van het `Serial` object (dat werd gebruikt om te communiceren met de PC).

De functies die we zullen gebruiken zijn beschikbaar als methoden van het object en worden gerefereerd met een punt tussen de namen van het object en de functie – zoals in bijvoorbeeld:

```
Serial.println(”Hello”);
```

Het object `Serial` bestaat op voorhand. Echter moet het object `LiquidCrystal` dat we gaan gebruiken eerst worden aangemaakt (zie het stukje code op de volgende pagina). Tijdens het aanmaken van dit object specificeren we welke pins we gaan gebruiken.

Zoals altijd loont het de moeite om constanten met zinvolle namen te gebruiken in plaats van direct getallen in te voeren. Al met al ziet dit er zo uit:

```
const int pinRS = 2;
const int pinE  = 3;
const int pinD4 = 6;
const int pinD5 = 7;
const int pinD6 = 8;
const int pinD7 = 9;

LiquidCrystal lcd(pinRS, pinE, pinD4, pinD5, pinD6, pinD7);
```

Zoals je ziet is `LiquidCrystal` het *type* van het object, terwijl het object zelf als `lcd` wordt gemaakt. We maken `lcd` aan als globale variabele, buiten zowel `setup()` als `loop()`.

Nu “weet” het `lcd`-object welke pinnen gebruikt moeten worden voor de signalering. De LiquidCrystal library is zeer algemeen, dus we moeten ook het displaytype specificeren. Dit doe je door de volgende regel in te voegen in `setup()`:

```
lcd.begin(16,2);
```

Hierna kunnen we tekst en nummers weergeven – precies zoals we dat in het monitorscherm op de PC konden doen. Er wordt alleen geen gebruik gemaakt van nieuwe regels – in plaats daarvan wordt de cursor verplaatst naar de plaats waar we de tekst willen plaatsen. Zie het voorbeeld hieronder:

```
lcd.setCursor(0,0);       // Left border, first line
lcd.print(”Hello World!”);
lcd.setCursor(0,1);       // Left border, second line
lcd.print(”Time:”);
lcd.setCursor(7,1);       // Center of second line
lcd.print(millis()/1000);
```

Uitdaging 2
Maak het programma af. De zes zojuist genoemde regels gaan in de `loop()`-functie.

Testen maar! De getallen op de onderste regel moeten elke seconde met één worden verhoogd.

4 – Kennismaken met het display
In de LiquidCrystal library zijn nog veel meer functies gedefinieerd, zoals:

```
lcd.home();       // As setCursor(0,0);
lcd.clear();      // Blanks the display, move to 0,0.
lcd.noDisplay(); // Blanks the display, but remembers the text
lcd.Display();    // Shows the text again
```

Uitdaging 3
Herschrijf het programma om de tijd te laten aftellen van 100 naar 0 en dan opnieuw te beginnen vanaf 100.

Je zult zien dat de “oude” tekst niet vanzelf verdwijnt. Als er ongewenste tekens zichtbaar zijn na iets dat wordt weergegeven, kun je eenvoudigweg een regel toevoegen die enkele blanco karakters op de bovenkant weergeeft: `lcd.print(”  ”);`

Als alternatief kun je het scherm volledig wissen met `lcd.clear();` Probeer maar. Het hangt van de context af of je een knipperende weergave zult ervaren wanneer alles wordt gewist voordat de nieuwe output wordt uitgevoerd. Vaak is de beste oplossing om te overschrijven met uitvulspaties.

Uitdaging 4
Onderzoek wat er gebeurt als je tekst uitvoert die iets verder gaat dan de rechterrand van het scherm.

Controleer ook wat er gebeurt als je 40-50 karakters over de grens uitvoert.

Er zijn andere, nog exotischere mogelijkheden met de LiquidCrystal library – die kun je op het net vinden:
https://www.arduino.cc/en/Reference/LiquidCrystal

Er staat bijvoorbeeld een beschrijving van hoe je je eigen karakters kunt definiëren (smileys, speciale letters...)

Merk overigens op dat dit display een vrij trage component is – het is amper te doen om deze constant bij te laten werken. Vaker dan 10 keer per seconde zal alleen leiden tot flikkeren, 5 keer is geschikt voor de meeste doeleinden.

Uitdaging 4
Schrijf een programma om de uitvoer naar het scherm te timen. (Het resultaat moet natuurlijk op het display worden weergegeven.)

Probeer het met zowel 16 karakters tegelijk als met een stapsgewijze benadering, waarbij de cursor meerdere keren handmatig wordt verplaatst.

Plaats het volledige programma in de `setup()`-functie – het hoeft maar één keer te draaien.