HAN_UNIVERSITY
OF APPLIED SCIENCES

EMBEDDED PROGRAM DEVELOPMENT (EPD)

WEEKOPDRACHT 2: DOBBELSTEEN

Versie: 22-08-19 16:01:00

INHOUDSOPGAVE

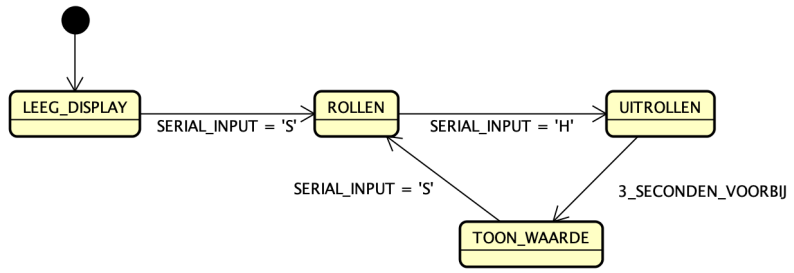
1 DOBBELSTEEN

Maak m.b.v. een **schuifregister** en een **7 segment-display** een dobbelsteen waarmee je de getallen 0_{16} t/m F_{16} kunt gooien. Zodra er via de seriële interface een 'S' (van *start*) binnenkomt, gaat de dobbelsteen 'rollen' met 25 standen per seconde (d.w.z. 25 keer per seconde verschijnt achtereenvolgens het volgende getal (0, 1, 2, …, E, F, 0, 1, …) op het scherm). Zodra er een 'H' (van *halt*) binnenkomt op de seriële poort zal de dobbelsteen 'uitrollen' in 3 seconden (d.w.z. er verschijnen langzaam steeds minder getallen per seconde, tot er uiteindelijk een waarde stil blijft staan). Zodra er vervolgens weer een 'S' binnenkomt, begint het rollen weer opnieuw.

Uiteraard programmeer je je code zonder gebruik van delay(), maar met softwaretimers. Verder moet je je code volgens de beschrijving in paragaaf 1.1 structureren. In paragrafen 1.2 en 1.3 kun je lezen hoe je je 7 segment-display kunt aansluiten en hoe je de cijfers en letters kunt laten zien.

1.1 TOESTANDSDIAGRAM

Tot nu toe hebben we toestandsdiagrammen gebruikt om de functionaliteit van een programma te beschrijven, waarbij we ons in de code zelf niet zo veel aantrokken van toestandsdiagram. In deze opdracht ga je je code echter structureren volgens het toestandsdiagram. Je zult zien dat het uitwerken van de opdracht hierdoor eenvoudiger wordt, en dat je code bovendien duidelijker gestructureerd is. Hieronder zie je het toestandsdiagram:



Gebruik onderstaande code als startpunt voor je uitwerking. Merk op dat de namen van de toestanden zijn gekoppeld aan constante getallen. Je zou natuurlijk ook gewoon de getallen 0, 1, 2 en 3 kunnen gebruiken zonder die constanten, maar op deze manier maak je je code beter leesbaar (en hoef je niet zelf bij te houden welk getal nu ook alweer bij welke toestand hoort).

```
const int LEEG_DISPLAY = 1;
const int ROLLEN = 2;
const int UITROLLEN = 3;
const int TOON_WAARDE = 4;

int huidigeToestand = LEEG_DISPLAY;

void loop() {
  switch(huidigeToestand) {
    case LEEG_DISPLAY:
    // Schrijf hier de code die moet worden uitgevoerd
    // in toestand LEEG_DISPLAY
    break;
    case ROLLEN:
    // Schrijf hier de code die moet worden uitgevoerd
    // in toestand ROLLEN
```

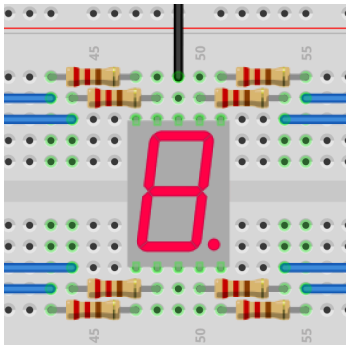
```
    break;
    case UITROLLEN:
    // Schrijf hier de code die moet worden uitgevoerd
    // in toestand UITROLLEN
    break;
    case TOON_WAARDE:
    // Schrijf hier de code die moet worden uitgevoerd
    // in toestand TOON_WAARDE
    break;
  }
}
```

In deze code moet je zelf dus nog alle functionaliteiten toevoegen, en ook het initialiseren van de hardware zit er nog niet in. Wat ook mist is, zijn de toestandsovergangen: ergens moet de waarde van de variabele huidigeToestand nog worden aangepast, anders blijven we voor eeuwig in de toestand LEEG_SCHERM. Later gaan we een mooiere plaats bedenken om de toestandsovergangen uit te programmeren, maar nu zou je ze gewoon binnen de CASE-stukken kunnen zetten (in de CASE van LEEG_SCHERM moet dus gecheckt worden of er een 'S' is binnengekomen via de seriële poort, en als dat het geval is, moet de toestand naar ROLLEN overgaan, etc.).

1.2 AANSLUITEN

Het 7-segment display bevat acht[1] leds die individueel kunnen worden aangestuurd via de pinnen. De GND-kant van de leds zijn met elkaar verbonden en aan te sluiten via de middelste pin aan de onderkant of bovenkant (die zijn met elkaar verbonden) van het display.

Je sluit de leds verder op dezelfde manier aan als "gewone" leds: je moet dus weerstanden toevoegen voor elke led. Het relevante deel van je opstelling zou er dus zo uit kunnen zien als in Afbeelding 1 (waarbij de blauwe kabels worden aangesloten op je shiftregister). Je kunt ervoor kiezen de decimal point niet aan te sluiten.

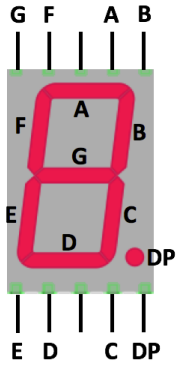


Afbeelding 1 Voorbeeld van opstelling

In Afbeelding 2 zie je welke pin met welke led verbonden is. Het ligt voor de hand om je opstelling zo te maken dat de pin van segment A te verbinden met Q0 op je shiftregister, B met Q1, C met Q2, et cetera.

1.3 FONT MAKEN

Sla de getallen 0_{16} t/m F_{16} op als bitpatroon (waarbij een 0 aangeeft dat een led niet brandt, en een 1 dat een led wel brandt). Zet deze bytes in een array en je hebt je font (lettertype) klaar. Je kunt eventueel het onderstaande ontwerp gebruiken voor de vormgeving van je cijfers en letters.

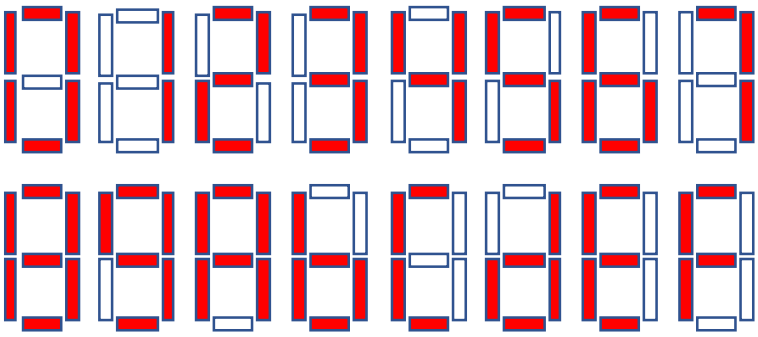


1.4 INLEVEREN

Lever in:

- Hardwareopstelling (Fritzing)
- Softwareontwerp (tabbladendiagram)
- Code

[1] De naam "7 segment" is dus een beetje verwarrend, maar wel te verklaren: er zijn zeven "langwerpige" leds waarmee je een vorm kunt tekenen. De achtste led is de punt ("decimal point").



Afbeelding 2 Koppeling pinnen met segmenten

2 COULD

2.1 DECIMAL POINT KNIPPERT

Laat de decimal point de hele tijd knipperen met 1 Hz.

2.2 MELODIETJE NA UITROLLEN

Direct na het uitrollen klinkt een melodietje uit de buzzer.

2.3 MELODIETJE TIJDENS UITROLLEN

Tijdens het uitrollen klinkt het melodietje.