Druk op de knop...

Tekst en schema's **Peter Knijff**; tekstbewerking **Hans van de Ven**

Op de club werd mij gevraagd om een geluidje (omroepbericht) te maken bij een station wanneer er een trein aankomt. Geluidsmodules zijn er te kust en te keur; je kunt ook aan de gang met een afgedankt mp3-spelertje. Maar … een geluid komt ook in de modelwereld nooit alleen! Hóór je het klateren bij het waterrad? En de zaagmachine produceert nog meer decibellen!

Zagerij met watermolen uit de Faller catalogus

Vóór, tijdens of ná een geluid vindt bijvoorbeeld het volgende plaats. De slagbomen gaat dicht of het armsein wordt gesteld (beweging). Het lichtsein wijzigt van kleur of er worden knipperlichten ingeschakeld (licht). De tekst op de CTA-bak wordt gewijzigd (display). En hierop zijn ontelbaar veel variaties te bedenken of in de werkelijkheid waar te nemen. Ik wil dus met een druk op de knop geluid, beweging, licht én een display in een zelfgekozen volgorde en interval realiseren.

Ik ben begonnen met een demo boardje waarop verschillende zaken met een druk op de knop het openen en sluiten van twee dubbele treinloodsdeuren simuleerden. Het volgende scenario werd met een druk op de knop uitgevoerd:

1. Een stem meldt dat de rechter deur open/gesloten wordt
2. Een knipperlicht gaat knipperen
3. De rechter deur gaat piepend open/sluiten (servo gaat bewegen)
4. Een stem meldt dat de deur geopend/gesloten is
5. Een stem meldt dat de linker deur open/gesloten wordt
6. De tweede deur gaat piepend open/sluiten (servo gaat bewegen)
7. Een stem meldt dat de deur geopend/gesloten is
8. Het knipperlicht gaat uit.

Parallel hieraan worden bijpassende teksten in een LCD-display getoond. Een foto van dit demo board met toelichting vind je op de laatste pagina van dit artikel.

Dit stukje is wel erg technisch en ik begrijp dat verschillende mensen dit niet kunnen volgen. Begrippen zoals Arduino, loop(), setup(), maar ook pre-emptive state machine zullen voor velen onduidelijk zijn. Toch denk ik dat dit artikel nieuwsgierige hobbyisten op weg kan helpen. Voor beginners: je kunt met dit opstapje de (geweldige wereld van de) Arduino leren kennen. Wie meer ervaren is kan aan de hand van dit artikel zijn Arduinoproject verbeteren. Eventueel met mijn hulp kun je betere sketches in de Arduino schrijven.

Bron: Het Zijspoor - clubblad van Modelbouwvereniging Arnhem e.o.

Ik zal ook delen van bestaande code bespreken. Deze code kan men van het internet afhalen. Hij staat namelijk in mijn persoonlijke GitHub repository. De code kan gedownload worden via de link: https://github.com/knijff1961/MVA. Hier is een knopje genaamd Code waar een zipfile gedownload kan worden. Ik ga mijn best doen om deze repository (of bibliotheek) bij te houden.

Mensen die met de Arduino vertrouwd zijn weten dat er twee routines in de sketch voorkomen:

1. De setup() die de Arduino vertelt wat er allemaal aan de Arduino "hangt" en hoe al deze hardware geïnitialiseerd dient te worden.
2. De loop() welke continue door de Arduino wordt aangeroepen. Deze loop blijft aangeroepen worden tot de stroom van de Arduino afgaat.

Laten we een eenvoudig programmaatje nemen: een knipperlicht; elke seconde aan-uit-aan-uit- etc. Stel een ledje is verbonden met pin 9 van de Arduino. Dit is een output dus in de setup zal de code moeten staan zoals:

```
void setup() {
  pinMode(9, OUTPUT);
}
```

En de loop kan dan zijn:

```
void loop() {
  digitalWrite(9, HIGH);
  delay(1000);
  digitalWrite(9, LOW);
  delay(1000);
}
```

Eerst wordt de led aangezet, dan wordt er een seconde (1000 ms) gewacht. Hierna zal de led uitgezet worden en wordt er weer een seconde gewacht. En dit blijft zich herhalen. Heel eenvoudig dus en inderdaad de led zal om de seconde aan- of uitgaan. Stel nu, we hebben twee knipperlichten (led pin 9 & led pin 10). De ene moet om de seconde knipperen, de ander om de 700 ms. Nu wordt het wat lastiger; we kunnen niet zomaar vertragingen erin plakken want hoe lang moeten deze delays zijn. Toch is er een oplossing: pre-emptive state machines.

De pre-emptive state machine

Om het begrip pre-emptive state te verduidelijken gaan we een café bekijken. In de keuken hangt een klok en in het café zijn twee tafels met mensen. De ene tafel wil koffie en tien minuten later thee. Daarna, na tien minuten koffie en na tien minuten weer thee, etc. Een mogelijke pseudo-code is:

- Breng koffie
- Wacht tien minuten
- Breng thee
- Wacht tien minuten

Koffie of thee? Foto Johan Vos

Maar je kunt ook zeggen:

- Breng koffie en bepaal de tijd voor de thee (tien minuten later dus).
- Doe allerlei andere dingen maar controleer wel regelmatig de tijd.
- Wanneer de tijd om is breng thee en bepaal de tijd voor de volgende koffie.
- Doe allerlei andere dingen maar controleer wel regelmatig de tijd.

En doe dit voor een onbepaalde tijd. Twee dingen moeten er dus bijgehouden worden: De tijd dat je iets moet doen - dit noemen we de timeout() - en wat je zal moeten doen (de state). Dit laatste bepaalt wat je gedaan hebt. Dus zolang er geen timeout is: ga de keuken in, bepaal de huidige tijd en ga weer het café (loop()) in.

Bovenstaande code kan dus herschreven worden door middel van:

```
Is er op deze tafel een timeout() {
  Bepaal de status:
  Is de status "koffie gehad" {
    Geef thee
    Zet de timeout 10 minuten later
  } anders is de status "thee gehad"
    Geef koffie
    Zet de timeout 10 minuten later
  }
}
```

NOTE: Dit is dus een state machine met twee statussen! Deze code moet in de loop() geplaatst worden (net zoals de tafel in de café staat). Zolang er niets gedaan hoeft te worden - er is dus geen timeout() - zal de loop() verlaten moeten worden om de nieuwe tijd te bepalen. Deze tijd kan gebruikt worden om een tweede state machine te maken en deze ook uitvoeren. Wanneer een tweede tafel (state machine) bezet is en deze persoon ook om en om koffie dan wel thee wil hebben maar dan om de zeven minuten, dan kan de code hiervoor gewoon in de loop gezet worden. Mocht de (zeldzame) situatie zich voordoen dat beide tijden precies op hetzelfde moment aflopen, dan zal het toch lijken alsof de twee tafels binnen de tien, resp. zeven minuten afgehandeld wordt. Elke opdracht wordt namelijk in 0,002 milliseconden gerealiseerd!

Iedere modelbouwer kent wel die scenes op een modelbaan met politie, brandweer en ambulance. Alle zwaailichten zijn aangesloten op een gezamenlijke knippermodule, waardoor ze met hetzelfde interval aan en uit gaan. Maar zo gaat het in de werkelijkheid natuurlijk niet. En onze pre-emptive state machine gaat de werkelijkheid nabootsen!

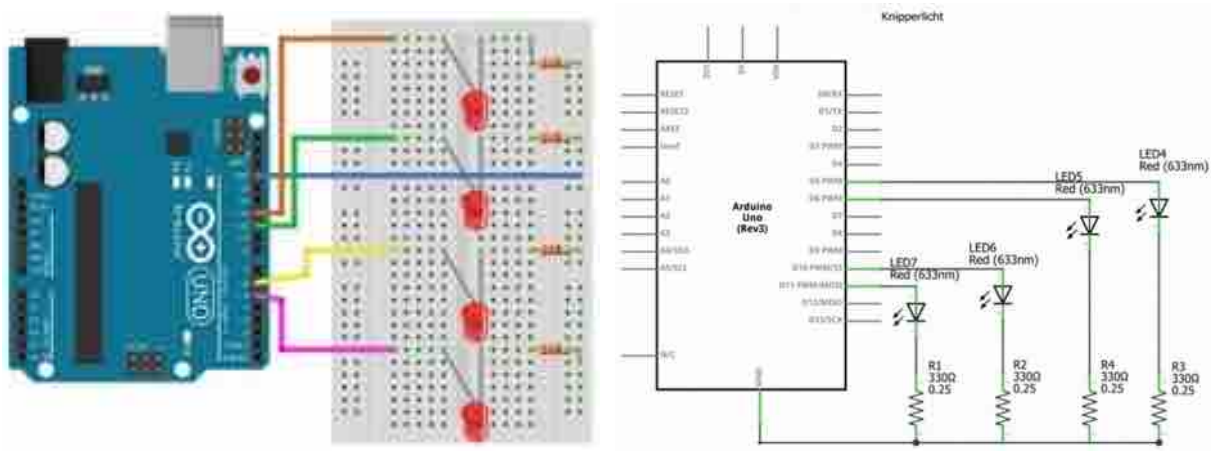
Het leuke is dat er nog een derde, vierde etc. tafel bij kunnen komen. De tafels kunnen gewoon allemaal afgehandeld worden zolang het café open is.

We wisselen koffie en thee nu even in voor twee ledjes. Die kun je niet drinken, maar je kunt er wel goed aan zien wat onze pre-emptive state machine doet. Het ene ledje moet elke seconde aan/uit gaan, het andere knippert iets sneller, namelijk in 0,7 seconde. De code met de twee knipperlichten waarvan de eerste om de 1000 ms knippert en de tweede hetzelfde doet om de 700 ms beslaat bijna een hele pagina en plaatsen we daarom niet in ons Zijspoor.

Ik heb een plaats "in de cloud" waar ik mijn MVA-documenten (dus ook de code) plaats. Iedereen kan deze documenten en Arduino codes downloaden en installeren. Dit alles is te vinden op https://github.com/knijff1961/MVA. Er is een groene "code" knop. Zoek daar naar CPreEmptiveTimerSimple.ino en je kunt de code voor dit dubbele knipperlicht bekijken en downloaden. Om deze code uit te kunnen voeren moet dit in de libraries directory van de Arduino IDE gezet worden. Deze is meestal te vinden in: My Documents\Arduino\libraries. Meer informatie hierover is te vinden in https://docs.arduino.cc/hacking/software/Libraries. Deze is wel in het Engels.

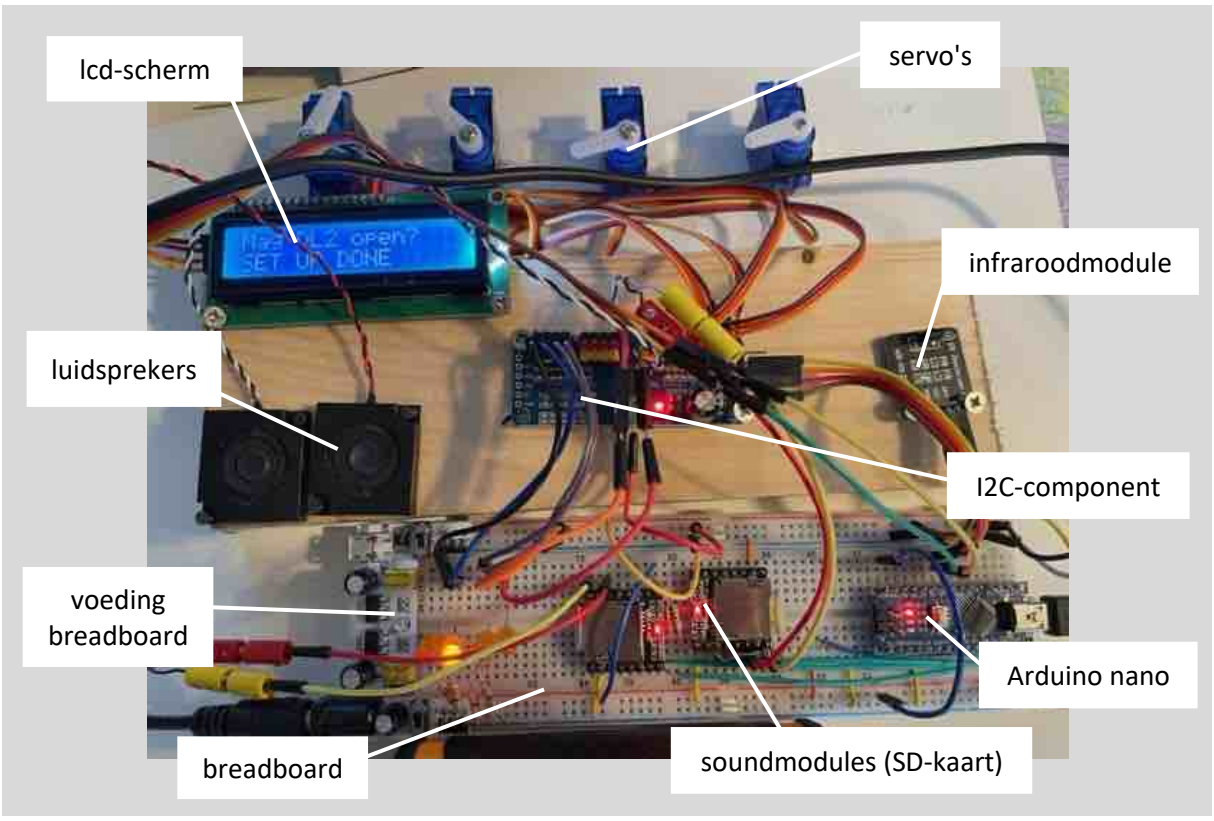
Hieronder staat onze eerste pre-emptive state machine in schema, gemaakt in Fritzing. Je vindt ook deze schema's in mijn github-omgeving onder de naam preemptive.fzz. Fritzing is

open-source EDA-software voor mensen die geen ingenieur zijn. Het perfecte hulpmiddel voor ontwerpers, uitvinders, hobbyisten en docenten voor het maken van een prototype of zelfs het maken van printplaten. Vaak wordt van dit hulpmiddel gebruik gemaakt om eenvoudig een schakeling te laten nabouwen.



Wie de code of schema's goed bekijkt zal opmerken dat er hier sprake is van vier ledjes. De twee ledjes, aangesloten op pin 5 resp. 6 doen hier nog niet mee in de bovenstaande code. Ze zullen de volgende keer worden gebruikt. Dan gaan we een (overweg) knipperlicht biblio-theek maken. Deze is natuurlijk ook pre-emptive. Het knipperlicht zal de oude gloeilampen nabootsen door langzamer aan en uit te gaan. Dit knipperlicht heeft dan 4(!) statussen.

Langzaamaan werken we dan toe naar de demo waarmee ik dit verhaal begon en die ik op de club meermalen mocht presenteren voor allerlei geïnteresseerde (en geïmponeerde) leden. Hier alvast een drone-view met toelichtingen.



Bron: Het Zijspoor - clubblad van Modelbouwvereniging Arnhem e.o.