Opgave Tussentijdse Oefeningen Jaarproject I
Reeks 3: Tijd, licht en warmte

Voor deze oefeningenles heb je de handleiding van de uitgedeelde ARM processor nodig. Je kan deze vinden op de website van het vak.

PCLK: De Oertijd

Elke processor wordt aangestuurd door een interne klok. Men spreekt dan ook over de 'kloksnelheid' van procesoren als men hun snelheden probeert te vergelijken. De uitgedeelde Olimex bordjes hebben elk een kristalletje dat aan een snelheid van 14.7456 mHz trilt. Dit signaal wordt binnen de processor vermenigvuldigd tot de uiteindelijke processorsnelheid van 60 mHz. In de handleiding van de processor wordt naar deze snelheid verwezen aan de hand van `pclk`. De processor leeft op het ritme van deze `pclk`. De volgende oefeningen behandelen hoe men met reële tijd kan werken door middel van de timer functionaliteit op de processor.

Oefening 1

Maak een 'naïeve' `wait` functie die iteratief aftelt tot nul. Hoe groter het getal waarvan je start, hoe meer werk de processor moet verrichten. Deze eenvoudige techniek kan daarom gebruikt worden voor de implementatie van een `wait` functie.

Gebruik je `wait` in het volgende stuk code:

```
> (begin (display "een")
               (wait 1000)  ; dit zou even moeten wachten
               (display "twee")
               (wait 1000)
               (display "wie niet weg is, is gezien!"))
```

Wat is het effect van `(wait 1000)` hier? Kan je op voorhand weten hoeveel je aan `wait` moet meegeven om exact 1 seconde te wachten? Wat beïnvloedt de duur van zulk een `(wait 1000)` oproep?

Oefening 2

In deze oefening proberen we een correctere benadering van de tijd te krijgen aan de hand van de opgegeven kloksnelheid van de processor.

Er zijn twee modules die de timer functionaliteit behandelen, zie p. 171 in de handleiding. Deze twee zijn identiek: TIMER0 en TIMER1. Dit laat je toe om twee tellers tegelijkertijd te gebruiken.

Om deze modules aan te spreken zijn er een hoop registers. Er is veel functionaliteit aanwezig die je nu niet nodig hebt. Zoek de volgende registers op.

- Timer Control Register (TCR), laat je toe de timer te starten, stoppen en resetten.
- Timer Counter Register (TC), dit is de teller die wordt opgehoogd.
- Prescale Register (PR), de *clock divider*. De bovenstaand teller wordt om de PR+1 klokslagen verhoogd.

Als Time Counter (TC) elke klokslag van `pclk` verhoogd wordt komen we in de problemen. Een 32-bit getal kan namelijk maar tellen van 0 tot 4.294.967.295. Hiermee kan je, aan de kloksnelheid van 60MHz, dus maar tot 71,5 seconden tellen. Daarom heeft men de timer functionaliteit van de processor uitgerust met een *prescale register*. De timer zal de teller (TC) pas verhogen nadat het aantal klokslagen verlopen zijn, dit aantal wordt aangegeven door het Prescale Register (PR). Zo zal bijvoorbeeld een *PR* van 60.000 ervoor zorgen dat de teller duizend maal (kHz) per seconde telt. Hiermee kunnen we veel langer tellen zonder dat de timer, een 32-bit getal, aan zijn maximum geraakt. Dit is dan wel aan een lagere precisie, 1 milliseconde in plaats van microseconde.

Oefening 2.1

Maak de volgende functies om TIMER0 aan te spreken:
- (timer-scale 60000) zal de waarde veranderen in PR door 60000.
- (timer-start) start de timer door gebruik te maken van TCR.
- (timer-stop) stopt de timer door gebruik te maken van TCR.
- (timer-read) geef de inhoud weer van TC.

tip: *Voor de registers aan te spreken zal je de functies read en write moeten gebruiken.*
(read <register address> <offset>) en (write object <register address> <offset>)
De juiste adressen staan in de handleiding op pagina 171.

Oefening 2.2

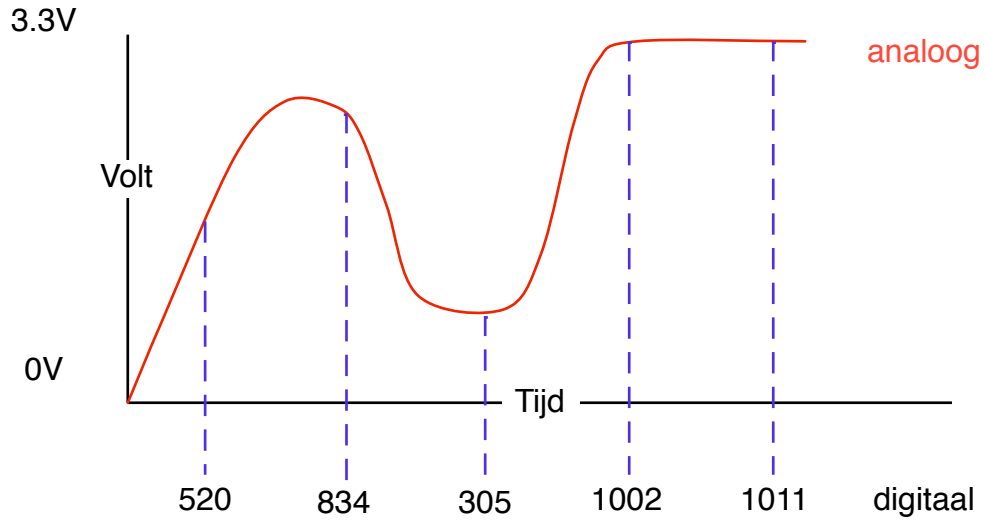
Maak een nieuwe `wait` functie die van de bovenstaande functies gebruik maakt. Vergelijk met je uurwerk of `(wait 1000)` wel degelijk 1 seconde wacht.

Het Digitale tijdperk: van Analoog naar Digitaal.

De enige sensor die tot nu toe gebruikt werd, was een drukknop. Door een pin uit te lezen als input krijgen we een *laag* of *hoog* voltage. Dit wordt omgezet naar bit 0 of 1, een digitaal signaal dat de processor kan uitlezen.
Vele sensoren werken op een andere manier, ze meten een *analoog* signaal. Een thermometer met kwik, een geluidsgolf, een licht dimmer; deze werken allemaal analoog. Een processor werkt met *digitale* signalen, dit wil zeggen dat het discrete waarden nodig heeft. Zo kan je op een digitale thermometer alleen de graden Celsius uitlezen op discrete intervallen: 22, 36

Om een analoog signaal om te vormen naar een digitaal kan je gebruik maken van een zogenaamde **analoog/digitaal converter** module op de processor. Deze zal metingen

uitvoeren, zoals een normale input pin. Alleen krijg je niet een 0 (laag signaal) of 1(hoog signaal), maar een 10-bit getal (0 tot 1023). Hoe groter dit getal, hoe hoger het originele signaal.



Op pagina 193 van de handleiding kan je lezen hoe je de analoog/digitaal converter moet aanspreken. Hiervoor zijn er twee registers: A/D Control Register (ADCR) en A/D Data Register (ADDR). Deze zijn nog verder onderverdeeld. Zo kan je met de eerste 8 bits van ACDR aangeven op welke pin er gemeten moet worden. Op de uitgedeelde bordjes staan vier *analog input* (ain) pinnen die je hiervoor kan gebruiken.
Wij zullen de module aanspreken om simpele 10-bit metingen te verrichten. De module ondersteunt wel geavanceerdere meettechnieken, zoals een BURST mode om gemiddelde nemen over een bepaalde tijd.

Voor deze oefening hebben we al een functie gemaakt die deze registers aanspreekt. Je kan deze code vinden op de website van het vak.
Voor de volgende oefeningen heb je alleen de functie `measure` nodig. Je gebruikt hem op de volgende manier:

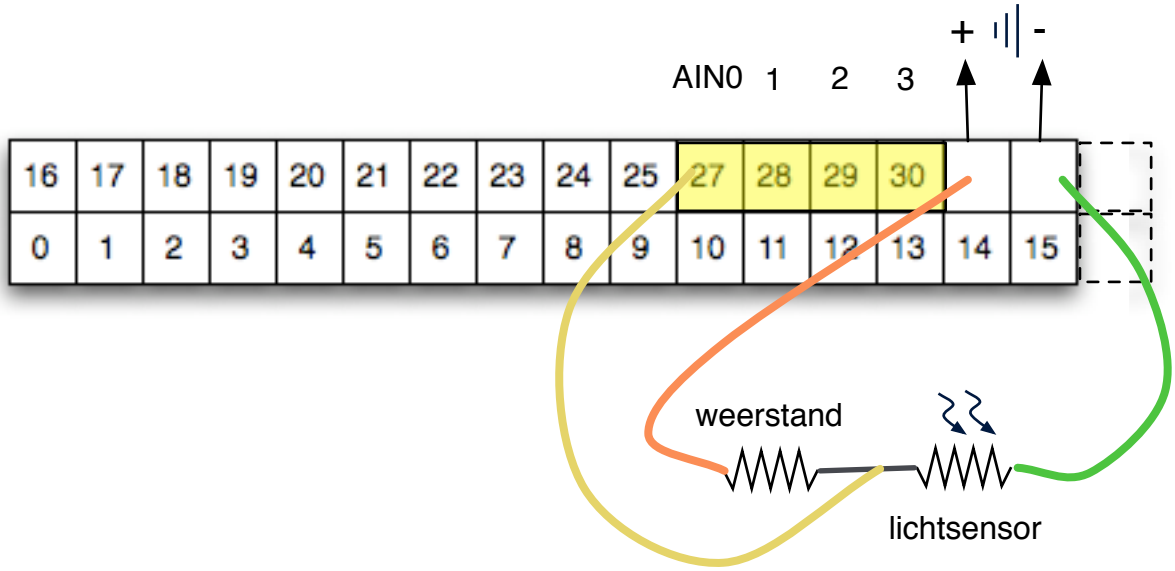
```
> (measure 'ain0)
0.52
> (measure 'ain0)
0.834
```

Zoals je ziet wordt het uitgelezen 10-bit getal omgevormd tot een getal tussen [0,1]. Dit noemt men *normaliseren*, een vaak gebruikte techniek.

Oefening 3

In deze oefening maken we een opstelling die je in staat stelt de waarde van een *lichtsensor* uit te lezen. De lichtsensor is een sensor die minder stroom doorlaat wanneer er min-

der licht op valt. Je kan dit uitlezen door de spanning (Voltage) uit te meten van een pin met de A/D converter. Maak hiervoor het volgende circuit.



Roep nu het volgende op.

```
> (measure 'ain0)
```

Doe dit een paar keer met je hand over of weg van de lichtsensor. Je zou de uitgelezen waarde moeten zien veranderen.

Oefening 4

De *Theremin* is één van de eerste elektronische muziekinstrumenten. Je bespeelt het door je hand op een afstand van de antenne te houden, zonder deze aan te raken. Hoe dichter je hand, hoe hoger de geproduceerde toon.

Maak zelf een *theremin* door gebruik te maken van een lichtsensor en een buzzer. Gebruik een lichtsensor in plaats van een antenne om hetzelfde effect te bekomen. Hoe dichter je hand bij de lichtsensor, hoe hoger de toon.

Gebruik de uitgelezen waarde van de lichtsensor om de toonhoogte van de buzzer (piezo) te controleren. Voeg hiervoor een buzzer toe aan je opstelling van oef. 3. Je mag kiezen welke output pin je gebruikt om de buzzer aan te sturen.

De sensormetingen moeten voortdurend gebeuren zodat de toon onmiddellijk verandert bij een verduistering van de lichtsensor.

nb. Let op dat de meting met `measure` *een floating point waarde geeft tussen 0 en 1. Deze moet je nog transformeren en afronden naar een nuttig geheel getal om de buzzer aan te sturen. Hiervoor bestaan scheme functies voor, zoals:*

```
> (round 374.2)
374
```