

Innovation Lab

Ingenieu(r)ze STEM-projecten voor secundair onderwijs

Je hartslagmonitor altijd op zak

JE HARTSLAGMONITOR
ALTIJD OP ZAK !



Hanne Deprez, Dimitri Coppens



Inhoudsopgave

Module 1. Inleiding	3
1 Wat en waarom?	3
2 De hartslag	5
3 Overzicht van het project	8
3.1 Hoe pakken we dat aan?	8
3.2 De strategie als blokdiagram	9
3.3 De kleurwaarde als functie van de tijd	10
Module 2. Apps maken met MIT App Inventor 2	11
1 MIT App Inventor 2	11
1.1 Maken van de app	11
1.2 Testen en uitvoeren van de app	15
2 Blokkens in de blocks view	16
2.1 Controle (control)	17
2.2 Logica (Logic)	19
2.3 Wiskunde (Math)	20
2.4 Tekst (Text)	20
2.5 Variabelen (Variables)	20
2.6 Lijsten (Lists)	20
Module 3. Digitale afbeeldingen en video	22
1 Digitale afbeeldingen	22
1.1 Hoe zijn digitale afbeeldingen opgebouwd?	22
1.2 Toepassing op de hartslagmonitor: roodwaarde berekenen in MIT App Inventor 2	24
2 Digitale video	26
3 Toepassing: de hartslagmonitor	28



Module 4. Automatische piekdetectie	33
1 Wat is een piek in een signaal?	33
2 Intermezzo: differentiequotiënt	35
3 Intermezzo: Frequentieanalyse	37
3.1 Geluid: zuivere en samengestelde tonen	38
3.2 Frequentieanalyse: het idee van Fourier	40
3.3 Toepassing: fourieranalyse voor onze hartslagmonitor	40
Module 5. Bouwen van de complete app	43
1 Video opnemen	43
2 Video naar afbeeldingen m.b.v. app VideoNaarAfbeeldingen	43
3 Gemiddelde roodwaarde voor elke afbeelding berekenen	44
4 Grafiek van de gemiddelde roodwaarde i.f.v. de tijd	44
5 Automatische piekdetectie en berekening van de hartslag	44
6 Extra: frequentie-analyse m.b.v. Fouriertransformatie	45
Module 6. Experimenteren met de app	46
1 Invloed van sport en rust	46
2 Vergelijking met bloeddrukmeter/hartslagmeter/smartwatch/app	46
3 Evaluatie: wat kan beter?	47
Oplossingen van de oefeningen zonder *	48
Module 1. Inleiding	48
Module 2. Apps maken met MIT App Inventor 2	48
Module 3. Digitale afbeeldingen en video	48
Module 4. Automatische piekdetectie	49
Module 5. Bouwen van de complete app	49
Module 6. Experimenteren met de app	49
Oplossingen van alle oefeningen	50
Module 1. Inleiding	50
Module 2. Apps maken met MIT App Inventor 2	50
Module 3. Digitale afbeeldingen en video	50
Module 4. Automatische piekdetectie	51
Module 5. Bouwen van de complete app	51
Module 6. Experimenteren met de app	51

Module 1

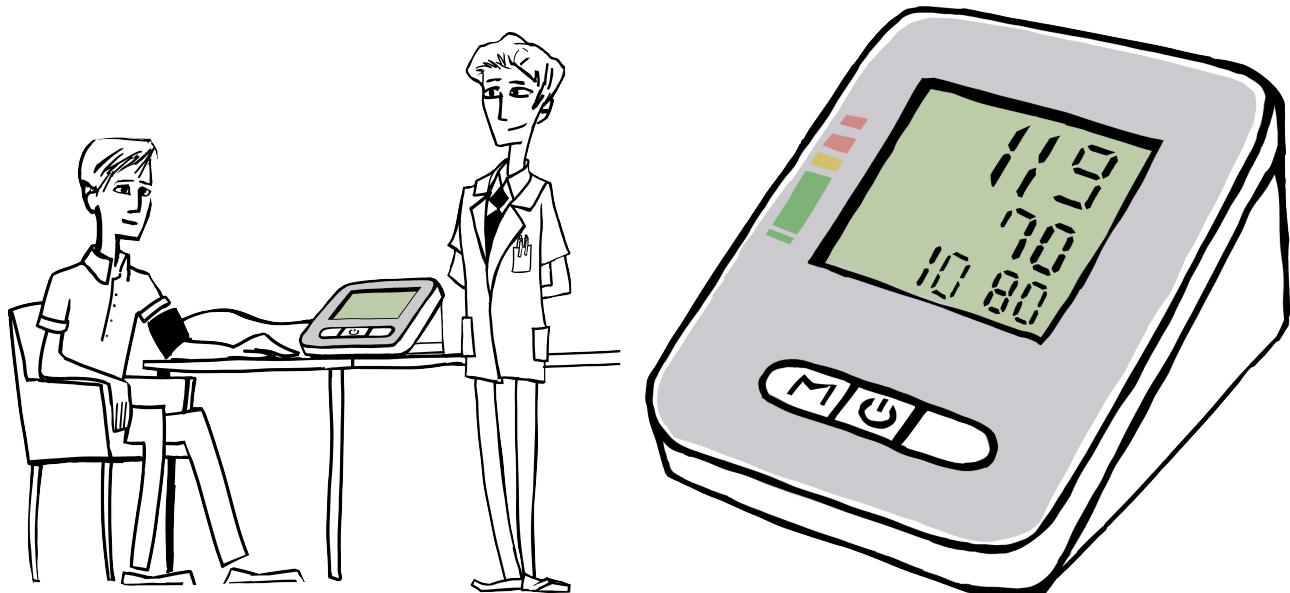
Inleiding

Tijdens dit project maken we een smartphone app waarmee je je hartslag kan meten. We leggen later uit waarom het belangrijk is af en toe je hartslag te meten en waarom we ervoor kiezen dit met een app te doen. Daarna bekijken we wat een hartslag precies is en hoe we die met onze smartphone kunnen meten.

1 Wat en waarom?

De hartslag duidt aan hoe vaak je hart klopt per minuut. Tijdens het sporten is het belangrijk je hartslag in het oog te houden. Een te hoge hartslag wijst op een te hoge inspanning, en kan mogelijks gevaarlijk zijn. Bovendien kan een (te) hoge hartslag in rust wijzen op aandoeningen zoals een overactieve schildklier, bloedarmoede of bloedverlies, een longaandoening enz. Regelmatig je hartslag controleren is dus de boodschap.

Om je hartslag te meten gebruiken we meestal een bloeddrukmeter.



Zo'n bloeddrukmeter is een gespecialiseerd toestel dat meestal enkel een dokter of iemand met hartklachten ter beschikking heeft. Bovendien is het nogal omslachtig een bloeddrukmeter tijdens het sporten mee te nemen, gezien dit niet echt een draagbaar toestel is.



Een alternatief voor de bloeddrukmeter is de hartslagmeter.



Een hartslagmeter kan je wel makkelijk meenemen bij het sporten. Bovendien geeft de hartslagmeter ook "real time feedback"; dat wil zeggen dat je op elk moment quasi onmiddellijk je hartslag kan zien. Maar wellicht ligt de aanschaf van een hartslagmeter niet binnen het budget van elke leerling of student?

Daarom bouwen wij in dit project onze eigen smartphone app, waarmee we snel, eenvoudig en redelijk accuraat onze hartslag kunnen meten, zonder nood aan gespecialiseerde toestellen.



2 De hartslag

Je hart is een spier die samentrekt om je bloed in je lichaam rond te pompen. De hartslag is de pompbeweging van je hart. In de volksmond is de hartslag ook het aantal samentrekkingen van je hart per minuut, of anders gezegd het aantal keer dat je hart klopt per minuut.

De hartslag in rust van een volwassene varieert tussen 60 en 100 slagen per minuut. De Maximale Hartslag (aangeduid als MH) is de hoogste hartslag die je lichaam aan kan gedurende fysische activiteit. De maximale hartslag varieert naargelang je leeftijd en kan berekend worden met de volgende vuistregel

$$MH = 220 - LT$$

met volgende notatie:

- MH: maximale hartslag met als eenheid slagen per minuut of “beats per minute” (kortweg: bpm). Dit noteren we als MH [bpm].
- LT [jaar]: leeftijd

Tijdens het sporten houd je best je hartslag tussen 55 en 85% van je maximale hartslag.

Voor een volwassene van 20 jaar oud is het aangeraden hartslagbereik tijdens het sporten dus tussen 110 en 170 bpm. Voor een volwassene van 50 jaar oud is het aangeraden hartslagbereik tijdens het sporten lager, nl. tussen 94 en 145 bpm.

Dit kunnen we in tabelvorm schrijven als volgt:

LT	MH	$0,55 \cdot MH$	$0,85 \cdot MH$
20	200	110	170
50	170	94	145

- 1** Bereken de maximale hartslag MH en het ideale hartslagbereik tijdens het sporten van een volwassene van
 - 18 jaar
 - 30 jaar
 - 65 jaar
- 2** Maak ook een grafiek die de maximale hartslag MH uitzet in functie van de leeftijd LT . Welke grootheid komt op de x -as? Wat is de eenheid?
Welke grootheid komt op de y -as? Wat is de eenheid?

De hartslag, of het rondpompen van je bloed, kan je voelen wanneer je je hand op je hart legt. Maar ook in je hals of aan je pols kan je je hartslag voelen. Handmatig meten van de hartslag kan je doen door twee gestrekte vingers op de pols (van jezelf of van een ander persoon) of in de hals (tegen de halsslagader) te plaatsen. Hier voel je de hartslag heel goed. Tel je gedurende 15 seconden het aantal hartslagen en vermenigvuldig je dit getal met vier, dan heb je een goede indicator van het aantal keer dat je hart pompt per minuut.

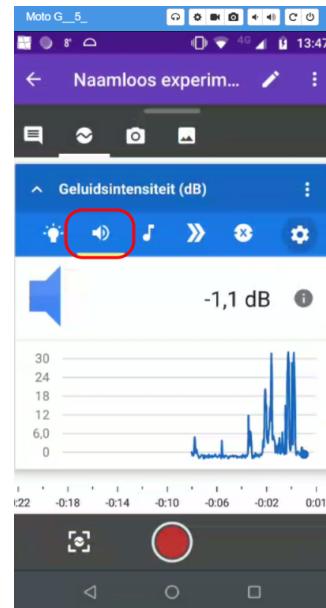
Opdracht: Je eigen hartslag meten

1. Handmatig
 - Meet je eigen hartslag door twee gestrekte vingers in je hals te plaatsen.
 - Tel het aantal keer dat je hart pompt gedurende 15 seconden en vermenigvuldig dit getal met vier.
2. Met de science journal app ¹

¹Tip voor leerkachten: Het gebruik van de app Science Journal kan je demonstreren door je smartphone scherm te projecteren op je computer. Hiervoor kan je het programma (op je pc) en de app (op je smartphone) Vysor gebruiken wanneer je smartphone met een USB kabel met de pc verbonden is.

Als je moeite hebt om het aantal hartslagen te tellen, kan je ook gebruik maken van de smartphone app ‘Science journal’. De app kan je downloaden in de Google Play Store.

- Start een nieuw experiment en meet het omgevingsgeluid, zoals in de figuur hiernaast.
- Plaats twee gestrekte vingers in je hals en zeg ‘ja’ elke keer als je een hartslag voelt.
- Doe dit gedurende 15 seconden.
- Tel nadien het aantal pieken dat geregistreerd werd en vermenigvuldig dit aantal met vier om de hartslag te bekomen.



Figuur 1. De app Science journal. Gebruik de luidspreker om je stem en het omgevingsgeluid op te nemen.



Figuur 2. Voorbeeld van een geluidsmeting in Science Journal.

- 3** Stel dat je gedurende 10 tellen in plaats van 15 seconden het aantal hartslagen bijhoudt.
- Met welke factor moet je dit aantal vermenigvuldigen om een correcte hartslag in bpm te bekomen?
 - Wat is de factor bij een telperiode van 20 seconden?

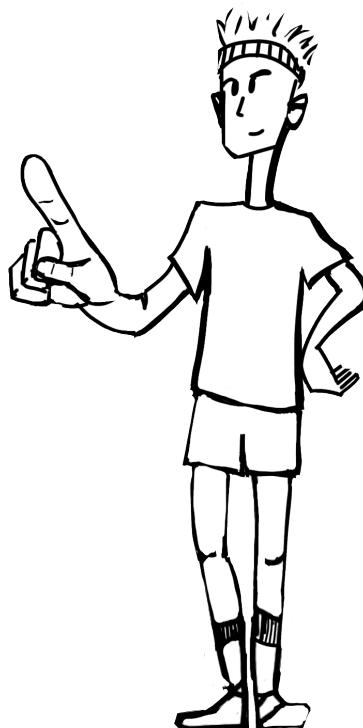
- Welk voordeel biedt een langere telperiode?

Voor heel hoge hartslagen ($> 160 \text{ bpm}$) wordt het quasi onmogelijk de hartslag nog zelf te tellen. Dan moet je een meettoestel gebruiken. In wat volgt maken wij ons eigen meettoestel: een smartphone app waarmee je goedkoop, snel en gemakkelijk je hartslag kan meten.

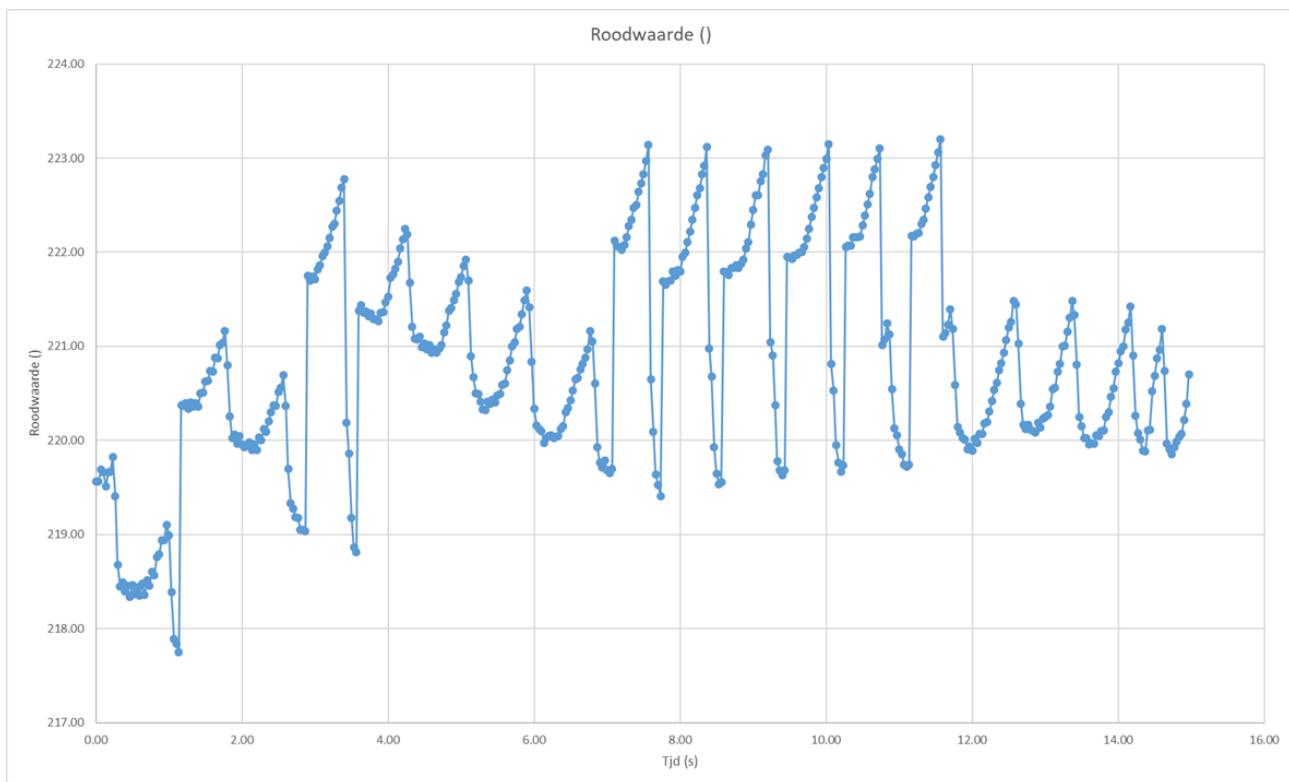
3 Overzicht van het project

3.1 Hoe pakken we dat aan?

We hebben al gezien dat je het pompen van je hart goed kan voelen aan je pols of in de hals. Een andere “handige” plek voor het waarnemen van je hartslag is je vingertop; ook daar doen zich kleine pulsaties in je aders voor door het pompen van je hart.

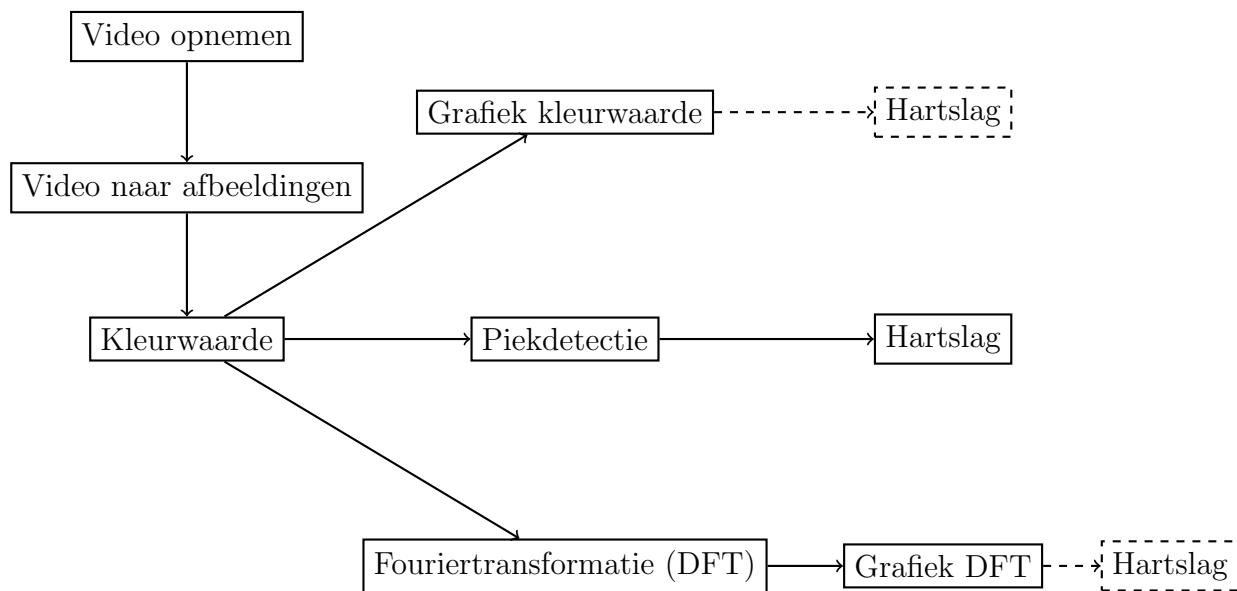


We gaan hiervoor slim gebruik maken van de camera die in je smartphone ingebouwd is. We zullen onze vingertop zachtjes op de lens van de camera plaatsen en daar een video van maken. De kleine pulsaties in de aders van je vinger resulteren in kleine veranderingen in de kleur van je vinger. Met het blote oog kan je die minuscule kleurveranderingen niet waarnemen, maar de camera van je smartphone kan die subtiele veranderingen wel detecteren. De kleurveranderingen in functie van de tijd zijn gerelateerd aan het pompen van je hart. Een grafiek van de kleurverandering in functie van de tijd zie je hieronder. We gebruiken dezelfde strategie als voorheen: we tellen het aantal pieken gedurende 15 seconden en vermenigvuldigen dit getal met vier.



3.2 De strategie als blokdiagram

De opeenvolging van uit te voeren stappen kunnen we voorstellen in een blokdiagram, zoals hieronder. Zo zien we grafisch welke stappen we moeten doorlopen om de hartslag te bepalen. Dit geeft een bondig overzicht van het project, zonder dat we nu al precies en in alle detail moeten weten wat in elke stap moet gebeuren. Deze stapsgewijze aanpak verhindert dat we zouden verdwalen in een overweldigende hoop denkwerk.

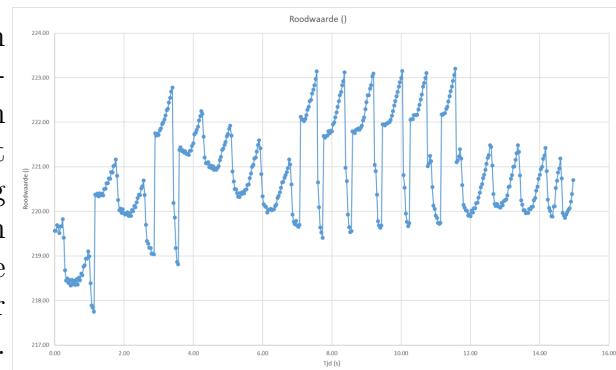


3.3 De kleurwaarde als functie van de tijd

We willen de kleurwaarde van je vingertop in functie van de tijd analyseren. De camera van je smartphone filmt de kleur van je vingertop. Een video of een filmpje is in feite niet meer dan een aantal foto's die heel snel na elkaar te voorschijn komen op een scherm. Die foto's laten ons toe om te bepalen wat de kleurwaarde van je vingertop was op het moment dat de foto werd genomen. We hebben drie verschillende manieren om vanuit die kleurwaarden de hartslag te bepalen.

1. Grafiek

Een eerste optie is het achter elkaar zetten van de kleurwaarde van alle foto's op die verschillende tijdstippen in een grafiek. Zo zien we het aantal pieken zoals hiernaast. Daaruit kunnen we zelf berekenen wat onze hartslag was op het moment van opname. We doen dit door zoals voorheen het aantal pieken te tellen en te vermenigvuldigen met een factor om het aantal pieken per minuut te bekomen.



2. Automatische piekdetectie

Een tweede manier bestaat uit het automatisch laten herkennen van die pieken door de smartphone. We sparen zo zelf heel wat rekenwerk: de smartphone telt immers het aantal pieken en herleidt dit aantal naar het aantal pieken per minuut. Dit bespaart ons heel wat rekenwerk.

3. Fouriertransformatie

Tenslotte kunnen we beroep doen op de zogenaamde *Fouriertransformatie* die gebaseerd is op wat complexere wiskunde. Vooral wanneer de meting een beetje ruizig is, wanneer we de pieken moeilijk kunnen herkennen, zal deze methode nauwkeuriger zijn. Hoe deze methode concreet werkt, wordt uitgelegd in Module 4 Sectie 3.

Bij methode 1 en 3 hebben we zelf nog wat denkwerk: we moeten nog gegevens aflezen op een grafiek en daaruit de hartslag bepalen. Daarom staan die twee opties in stippellijn in het blokdiagram. Bij methode 2 doet de smartphone al het lastige reken- en denkwerk; daarom staat die optie in volle lijnen in het blokdiagram.

Module 2

Apps maken met MIT App Inventor 2

We gaan een hartslagmonitor maken in de vorm van een smartphone app! MIT, een bekende universiteit in de Verenigde Staten, ontwikkelde een programmeeromgeving waarmee je zeer eenvoudig apps kan maken en testen. In dit hoofdstuk bespreken we de belangrijkste structuren voor het maken van een app. Ook in andere programmeertalen en -omgevingen zal je deze structuren kunnen gebruiken.

1 MIT App Inventor 2

Met behulp van MIT App Inventor 2 kan je zelf apps maken. Alles wat je nodig hebt, is een computer met wifi-verbinding en een smartphone geconnecteerd met hetzelfde wifi-netwerk.

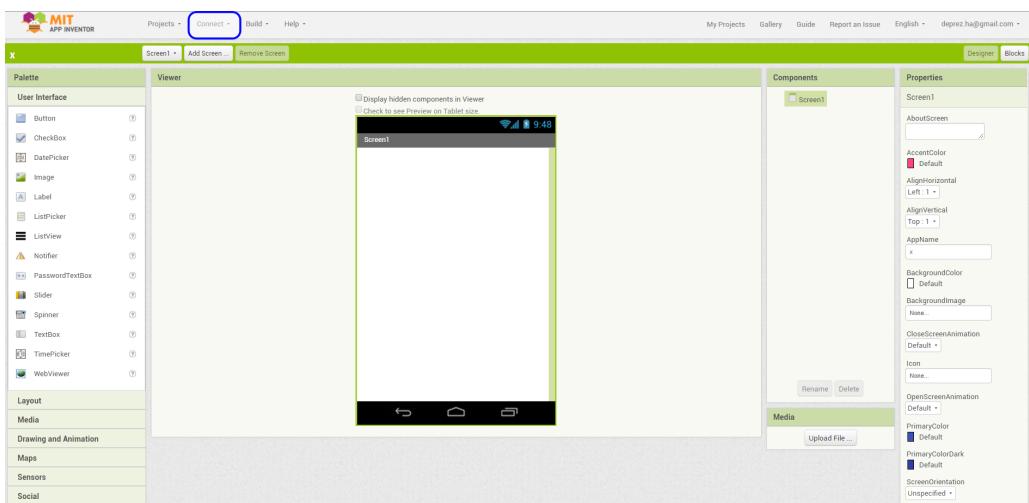
1.1 Maken van de app

Surf naar de website <http://appinventor.mit.edu/explore/> en klik rechtsboven op **Create apps!** Je zal moeten inloggen met een google account. Als je nog geen google account hebt, kan je die aanmaken via <https://accounts.google.com/signup>. Na succesvol aanmelden kom je op de pagina met je projecten terecht.

Creëer een nieuw project, noem het bvb. “HelloWorld”, en klik erop om het project te openen. Je komt op het startscherm terecht. Er zijn twee ‘views’.

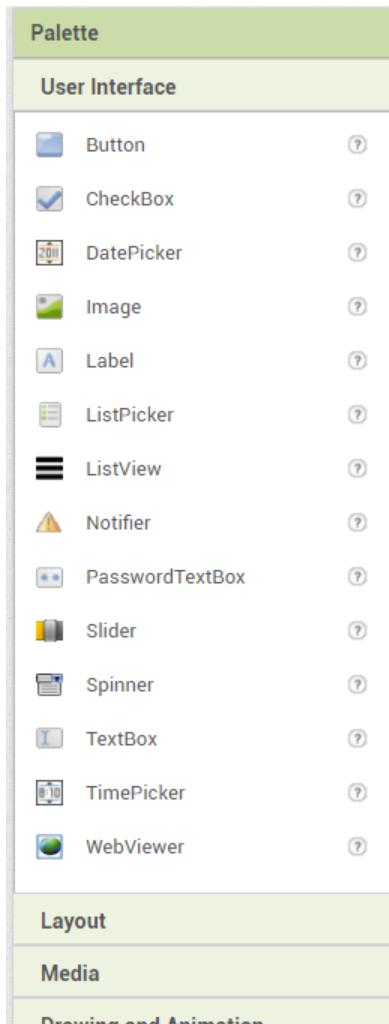
Designer view

De eerste view is de *designer view*, weergegeven in Figuur 1.



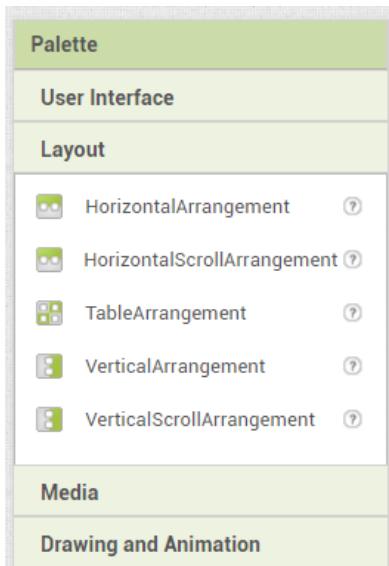
Figuur 1. Designer view in MIT App Inventor 2. Overschakelen tussen views doe je rechtsbovenaan (blauwe rechthoek). Tabs zijn aangeduid in rood.

In deze view kan je de componenten toevoegen die je zal zien op het scherm tijdens het uitvoeren van je app. Een component kan zijn: een **knop** (Engels: **button**), een **label**, een **text** enz.



In de *Palette* tab (links) vind je alle componenten die je kan toevoegen. Je voegt componenten toe door ze vast te nemen en ze naar het scherm van de afgebeelde smartphone te slepen.

Wat kan je allemaal doen met deze componenten?



Je kan je componenten grafisch schikken door ze in een **HorizontalArrangement**, **VerticalArrangement** of **TableArrangement** te plaatsen.

Components

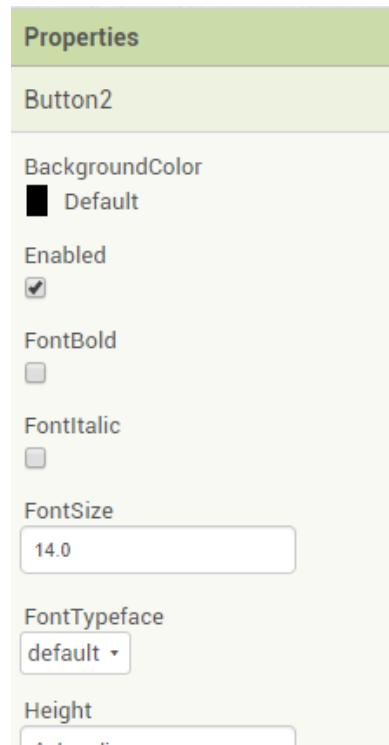
- Screen1
 - Button1
 - Button2**
 - Image1

Rename Delete

Om een overzicht te kunnen houden, moet je al snel je componenten benoemen. Dit doe je in de tab *Components* door de component te selecteren en op **Rename** te drukken.

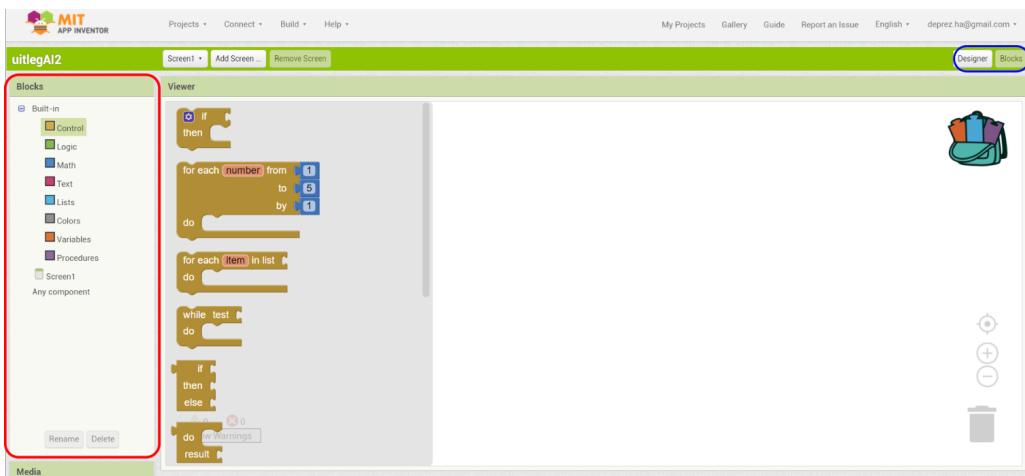
Je zal de eigenschappen van de componenten ook wijzigen voor de specifieke taak die ze zullen krijgen. Dit doe je in de *Properties* tab helemaal rechts.

Welke tekst moet op de knop staan? Wat is de lettergrootte? Wil je de tekst in een andere kleur weergeven? ...



Blocks view

De andere view is de *blocks view*, weergegeven in Figuur 2.



Figuur 2. Block view in MIT App Inventor 2.

De *Designer view* kan je vergelijken met het dashboard van een auto. De bestuurder ziet er alles waarmee hij de wagen kan besturen. Door over te gaan naar de *Blocks view*, duiken we als het ware in de motorkap van de wagen. Hier gaan we de eigenlijke werking van de app programmeren.

Wat moet er gebeuren als je op de knop drukt? Misschien wil je de tekst in een label aanpassen, of wil je een andere afbeelding tonen. Er kan veel; wat je implementeert, hangt af van je verbeelding en van hoe jij wil dat de app zal werken.

Experimenteren met MIT App Inventor 2

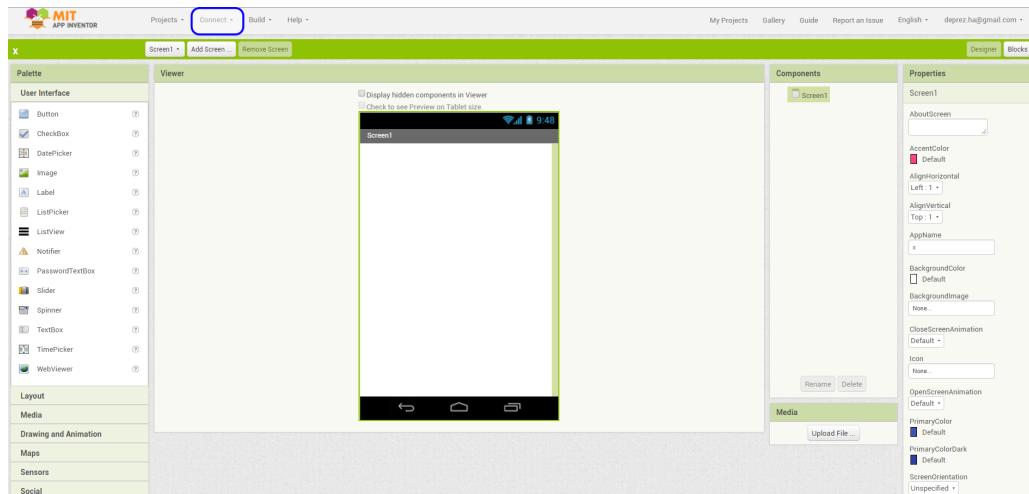
Ga na wat je allemaal kan doen in de programmeeromgeving. Welke componenten bestaan er allemaal? Welke eigenschappen kan je aanpassen?

- Probeer een eerste app te maken met een knop waarop de tekst “Zeg hallo” staat en een label zonder tekst.
- Als je op de knop drukt, moet de tekst “Hello world!” in het label verschijnen.

Dit “Hello world!”-programma is een van de simpelste programmatjes die je kan schrijven. Wanneer programmeurs een nieuwe programmeertaal aanleren, schrijven ze steeds “Hello world!”-programma om te leren werken met de nieuwe taal, die we met een geleerde woord syntax noemen.

1.2 Testen en uitvoeren van de app

Om de app te testen moet je de MIT App Inventor 2 Companion App (te vinden in de Google Play Store) installeren op je smartphone. Zorg ervoor dat je computer en smartphone met hetzelfde wifi-netwerk verbonden zijn. Klik op **Connect**, linksboven op het scherm zoals aangeduid op de figuur.



Kies de eerste optie **AI companion**. Er verschijnt een QR-code die je kan scannen met je smartphone met de MIT App Inventor 2 Companion App. De app opent op je smartphone (even geduld, nergens op klikken!), zodat je de werking van de app kan testen.

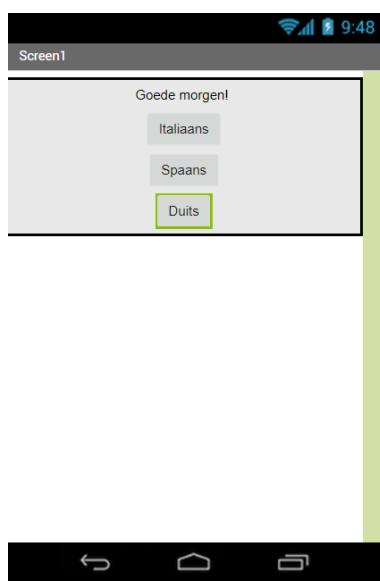
Zolang je smartphone via wifi gekoppeld is aan de Companion App, verandert de app op je smartphone tegelijkertijd met de aanpassingen die je doorvoert op het scherm van je computer. Soms werkt dit echter niet zo goed. Breek dan de connectie met de smartphone af via **Connection - reset connection** en maak de verbinding opnieuw via **Connection - AI companion**.

Als je de app die je ontworpen hebt getest hebt via de App Inventor 2 Companion App en de app aan de vereisten voldoet, kan je de app downloaden. Zo heb je een zelfstandig werkende app op

je smartphone, zonder dat je een wifi-verbinding met je computer of de MIT ontwikkelomgeving nodig hebt. Dan hoef je ook niet meer steeds de QR code te scannen als je de app wilt gebruiken. De app kan je installeren via **Build – App (provide QR code for .apk)**. Door een QR code te scannen met je smartphone kan je het installatie-bestand (.apk) downloaden op je smartphone. Wanneer je het .apk-bestand opent, installeert de app.

Good morning app

Toon “Good morning!” in drie verschillende talen op het scherm, nl. Italiaans, Spaans en Duits.



1. Creëer een nieuw project.
2. Start met de *Designer view*:
 - Welke layout kies je?
 - Welke componenten heb je nodig?
 - Welke naam geef je aan iedere component?
 - Welke eigenschappen heeft elke component nodig?
3. Ga over naar de *Blocks view* om ervoor te zorgen dat:
 - een klik op de **knopItaliaans** “Buongiorno!” toont
 - een klik op de **knopItaliaans** “Buenos dias!” toont
 - een klik op de **knopItaliaans** “Guten Morgen!” toont

Test de werking van je app grondig. Doet de app op elk moment wat je ervan verwacht? Indien niet, ga dan op zoek naar de onderliggende oorzaak. Het proces waarin je ongewenste fouten in de werking van de app opspoort en wegwerkt, heet “debuggen”. Zelfs de beste programmeurs moeten hun apps debuggen. Dit is soms een frustrerend proces, maar het is helaas wel een wezenlijk onderdeel van programmeren.

2 Blokken in de blocks view

Met MIT App Inventor 2 kan je heel gemakkelijk zelf aan de slag. De betekenis van de componenten is vaak intuïtief duidelijk en via het vraagteken kan je steeds extra uitleg oproepen.

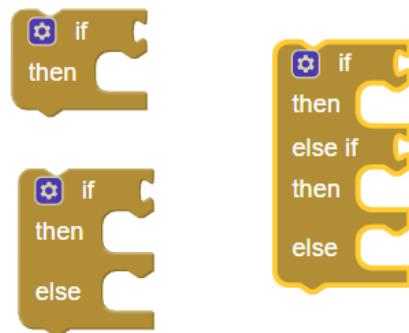
Hieronder bespreken we enkele belangrijke blokken uit de blocks view, waarvan de bedoeling misschien iets minder vanzelfsprekend is.

2.1 Controle (control)

De meeste programmeertalen zijn opgebouwd rond drie belangrijke controlestructuren:

1. ‘if’-loops om beslissingen te nemen,
2. ‘for’-loops om herhaling in te bouwen, en
3. ‘while’-loops om herhaling in te bouwen.

Beslissingen nemen: *If then* blok



‘If’-lussen laten toe een programma dynamisch te maken. Een ‘If’-lus kan je vergelijken met het kiezen van een weg op een kruispunt: als een voorwaarde voldaan is (YES), ga je naar links; anders (NO) ga je rechts.

Het testen of een voorwaarde voldaan is, gebeurt met een Boolean expressie. Een Boolean expressie is een uitdrukking die ofwel waar of ofwel vals is. Het resultaat van Boolean expressie is een Boolean, een variabele die maar twee waarden kan aannemen: **true** als de voorwaarde voldaan is, en **false** als de voorwaarde niet voldaan is.

In App Inventor zijn er verschillende implementaties van deze structuur mogelijk:

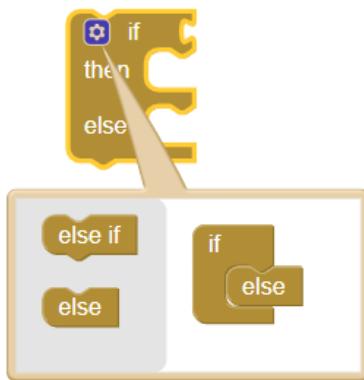
- *If then* blok

Je kan kiezen om een bewerking enkel uit te voeren als een voorwaarde voldaan is (*If then* blok).



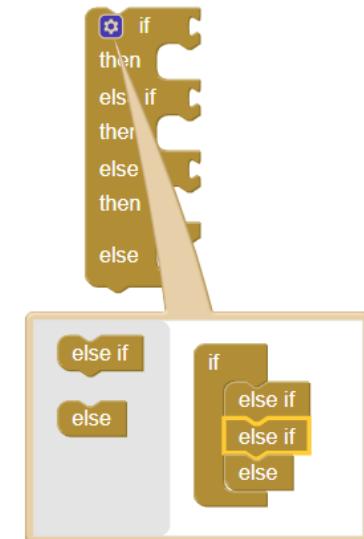
- *If then else* blok

Het kan zijn dat je een eerste bewerking wenst uit te voeren als aan de voorwaarde voldaan is, maar dat er een andere bewerking zal gebeuren indien nodig. De “else” toevoegen doe je door op het blauwe icoontje met het tandwielteeltje te klikken en een “else”-blokje in het *If then*-blok te slepen.



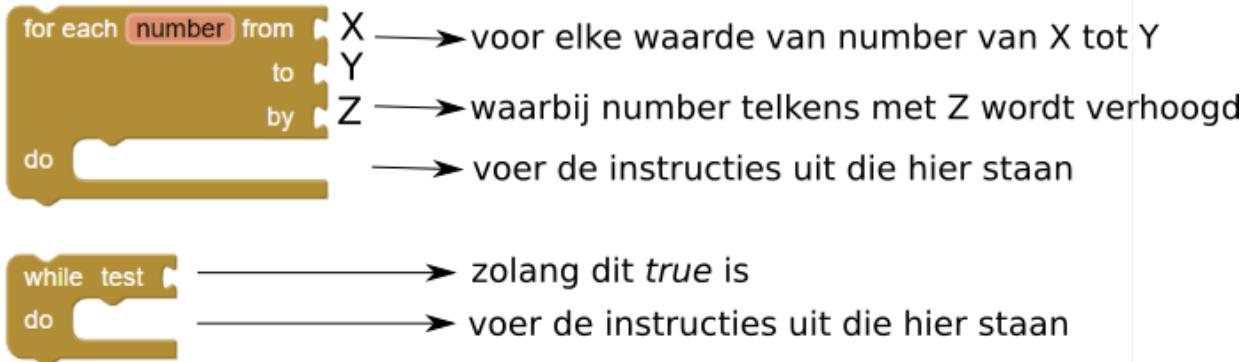
- *If then, else if then, else then* blok

Als je echt complexe dingen wil doen, kan je extra voorwaarden toevoegen. Dit doe je door op het blauwe icoontje met het tandwielteeltje te klikken en “else if”-blokjes en een “else”-blokje in het *If then*-blok te slepen..



Herhaling: *While test* lus of *for each* lus

Voor zaken die je veelvuldig wil uitvoeren, kan je gebruik maken van een een repetitie-blok, zoals een *While test* lus of *for each* lus. Zo'n herhaling zou kunnen zijn: de tekst op een label laten optellen van 1 tot 100.



- Bij een *for each*-lus wordt een *teller* (in ons voorbeeld **number**) steeds verhoogd (in ons voorbeeld met **Z**) binnen een bereik van waarden (in ons voorbeeld tussen **X** en **Y**). Een *for*-lus is ideaal om een probleem op te lossen waarbij een lus een vast aantal keer herhaald moet worden.
- Een *While test* lus heeft twee delen:
 1. een *test*-socket: Boolean expressie (waar of niet waar?) die getest wordt, en
 2. een *do*-socket: een verzameling instructies die uitgevoerd wordt zolang de Boolean expressie **true** is.

Bij het programmeren van een herhalingslus moet de booleaanse uitdrukking ooit **false** worden. Anders blijft het programma eindeloos in cirkeltjes draaien en zal het lijken alsof er niets gebeurt, ook al is de computer aan het rekenen! Pas dus op dat je geen oneindige lussen maakt: ooit moet de Boolean expressie **false** worden, anders blijf je vastzitten in de *while*-lus!!!

2.2 Logica (Logic)

Zoals eerder gezegd is een Boolean expressie een expressie die ofwel waar of ofwel vals is. We kunnen op een aparte manier gaan rekenen met de mogelijkheden waar en vals. Hiervoor beschikken we over zogenaamde ‘logische operatoren’. Dit zijn bewerkingen waarmee we booleaanse uitdrukkingen kunnen combineren. Zo kunnen we meer complexe voorwaarden testen.

- De logische operator **AND** geeft als output waar als **beide** inputs waar zijn, en onwaar als een of meerdere inputs onwaar zijn.
- De logische operator **OR** geeft als output waar als **minstens één** van beide inputs waar is, en onwaar als **beide** inputs onwaar zijn.
- De logische operator **NOT** geeft als output onwaar als de **input waar is** en geeft als output waar als de **input onwaar is**.

2.3 Wiskunde (Math)

Uiteraard kunnen bewerkingen op getallen uitgevoerd worden. Je vindt het volledige overzicht van mogelijke bewerkingen onder *Blocks - Math*.

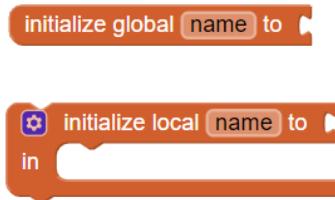
2.4 Tekst (Text)

Naast wiskundige bewerkingen kan je ook tekst input van de gebruiker verwerken. De mogelijke bewerkingen die je op tekst kan uitvoeren vind je onder *Blocks - Text*.

2.5 Variabelen (Variables)

Variabelen worden gebruikt om waarden bij te houden. Een bij te houden waarde kan bvb. het resultaat van een wiskunde bewerking zijn, of een tekst input van de gebruiker die je later nog wilt aanpassen.

Belangrijk is dat er een onderscheid gemaakt wordt tussen *lokale* en *globale* variabelen, zie Figuur 3. Globale variabelen worden aangemaakt bij de opstart van het programma en kunnen doorheen het hele programma gebruikt worden. We zouden kunnen zeggen dat je vanuit elk blok “aan de globale variabele kan”. Lokale variabelen worden aangemaakt binnen een blok en kunnen enkel door blokken binnen de grenzen van de lokale variabele gebruikt worden. Je “kan er dus enkel aan” binnen het oranje blok zelf. Vandaar dat het blok voor een lokale variabele onderaan nog een oranje staartje heeft. Dit staartje duidt aan binnen welke grenzen “je aan de variabele kan”.



Figuur 3. Het bovenste blok toont hoe je een globale variabele kan aanmaken, die doorheen het hele programma gebruikt kan worden. Het onderste blok toont hoe je een lokale variabele kan aanmaken, die enkel toegankelijk is binnen het blok van de variabele.

De **Get**-instructies worden gebruikt om de waarde van een variabele op te halen. De **Set**-instructie wordt gebruikt om de waarde van een variabele te wijzigen.

2.6 Lijsten (Lists)

Een lijst kan meerdere items met gelijkaardige data bevatten. Zo kan je bijvoorbeeld een lijst van getallen hebben, of een lijst van tekstwaarden. Als je 100 getallen moet bijhouden, is het makkelijker een lijst te gebruiken dan 100 verschillende variabelen.

Nadat een lijst wordt aangemaakt zoals in Figuur 4, kan je extra items aan de lijst toevoegen, of er items uit verwijderen. De volledige lijst van bewerkingen die je op een lijst kan toepassen, vind je onder *Blocks - list*.



Figuur 4. Je kan een lege lijst aanmaken (boven), of een lijst aanmaken die reeds een aantal elementen bevat.

Module 3

Digitale afbeeldingen en video

Het meten van je hartslag met de smartphone is gebaseerd op kleurveranderingen in je vingertop, die we met de smartphone camera registreren in een video. Om goed te begrijpen hoe we die kleurveranderingen precies kunnen detecteren, is het belangrijk de basisprincipes achter digitale afbeeldingen en video te begrijpen. Daar gaan we in deze module dieper op in.

1 Digitale afbeeldingen

1.1 Hoe zijn digitale afbeeldingen opgebouwd?

Om jullie te laten kennis maken met de basisprincipes achter digitale afbeeldingen, raden we jullie aan onderstaand filmpje te bekijken. Het is een Engelstalig filmpje, maar je kan het ondertitelen (in het Engels) en eventueel ook vertraagd bekijken.



In het filmpje wordt uitgelegd dat digitale afbeeldingen opgebouwd zijn als een rooster. De vakjes uit het rooster noemen we “pixels”.

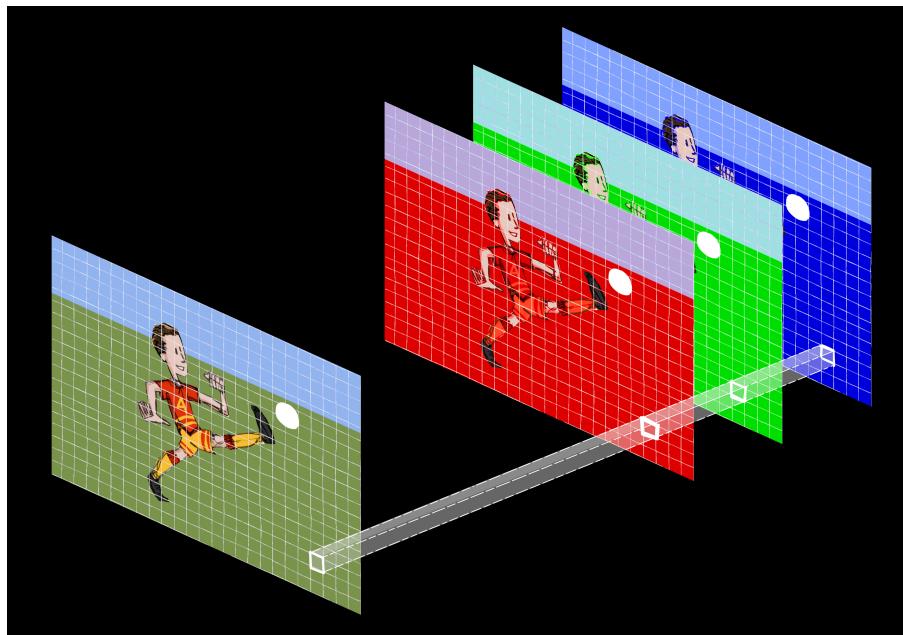


Elk pixel heeft een roodwaarde, een groenwaarde en een blauwwaarde. Met de combinatie van drie kleuren, nl. rood, groen en blauw, kan je quasi elke bestaande kleur genereren.

De rood-, groen- en blauwwaarde van een pixel is steeds een getal tussen 0 en 255. Hoe kleiner het getal, hoe donkerder de kleur. Een roodwaarde van 0 komt dus overeen met een rood dat bijna zwart is. Omgekeerd: hoe groter het getal, hoe lichter de kleur. Een roodwaarde van 255 komt dus overeen met een rood dat bijna helemaal wit is. De drie getalwaarden voor rood, groen en zwart samen bepalen volledig de kleur die je zal zien. Dit drietal noemen we de RGB-waarde (R van rood/red, G van groen/green en B van blauw/blue) van de pixel.

- 1** Hoeveel verschillende RGB-kleuren kan je genereren als de rood-, groen- en blauwwaarden steeds een waarde tussen 0 en 255 aannemen?

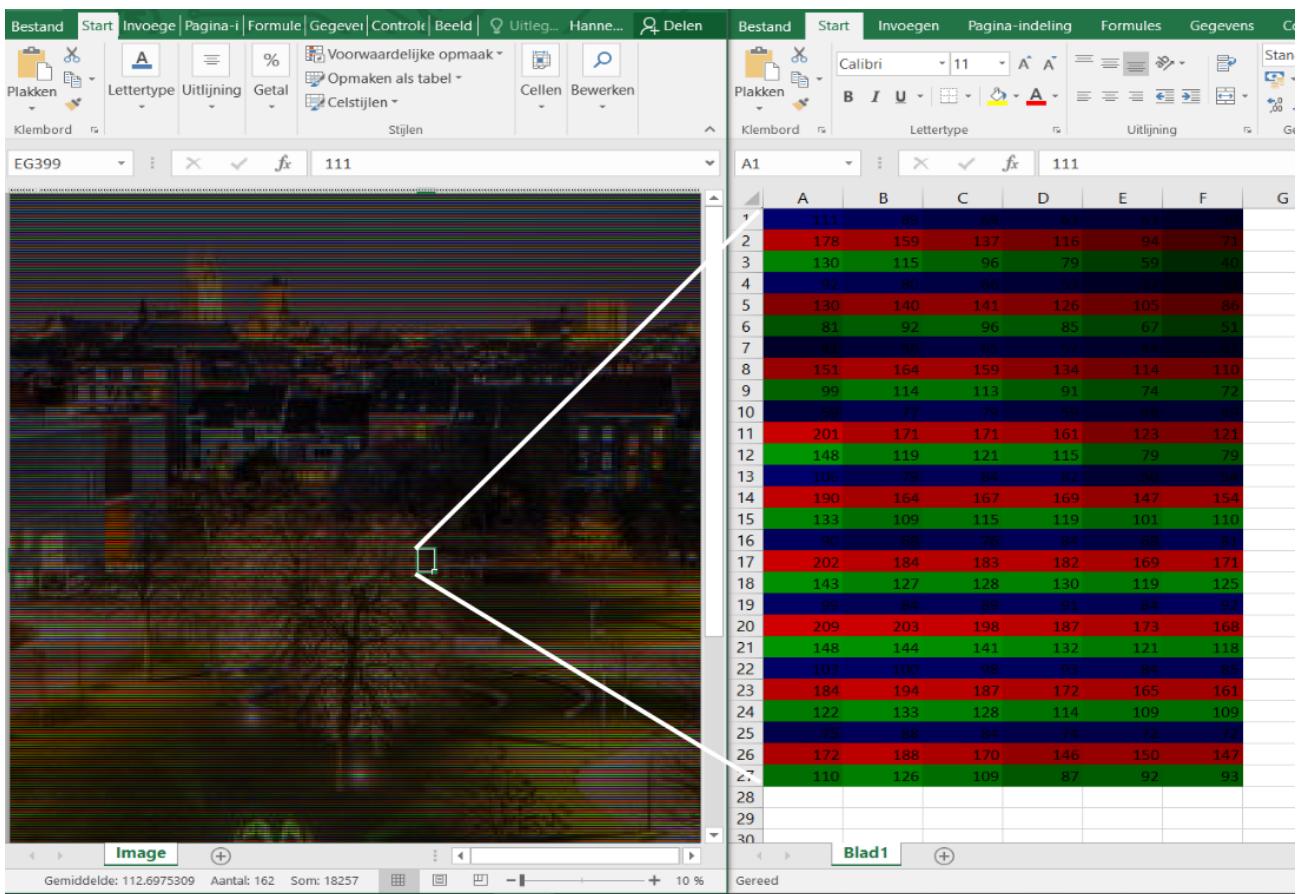
Hieronder zie je hoe een digitale afbeelding in een computer voorgesteld wordt: van elk pixel wordt bijgehouden hoeveel rood, groen en blauw die bevat.



Je kan ook zelf experimenteren met de RGB-waarden van een foto.

Via <http://makeanddo4d.com/spreadsheet/> of onderstaande QR-code kan je een foto omzetten in een Excel-werkblad. Inzoomen in het werkblad toont je de RBG-waarden van de individuele pixels. Uitzoomen in het werkblad toont je de volledige foto, weliswaar met een beperkte resolutie.





Als je de RGB waarden van enkele pixels aanpast en een beetje uitzoomt, zal je een andere kleur zien.

1.2 Toepassing op de hartslagmonitor: roodwaarde berekenen in MIT App Inventor 2

Voor de hartslagmonitor gaan we de roodwaarde van je vingertop analyseren.

Gedurende 1 hartslag verandert je vingertop heel subtiel van kleur. Wanneer het bloed door je aders gepompt wordt, zetten je aders een heel klein beetje uit. Daardoor komt het bloed net iets dichter bij de oppervlakte van je huid en kleurt je vingertop net iets meer rood dan daarvoor. Die verandering in roodwaarde van je vinger kan je met het blote oog niet detecteren, maar als je een filmpje maakt van je vinger, kan een computer, of zelfs je smartphone, die subtile kleurverschillen wel waarnemen.

We maken een filmpje van onze vingertop. In dit filmpje worden de kleurveranderingen van je vingertop ten gevolge van je hartslag geregistreerd. Een filmpje dat afspeelt is in feite niet meer dan een aantal foto's die heel snel na elkaar getoond worden. Hier gaan we zo dadelijk, in Sectie 2 verder op in. Als we het opgenomen filmpje splitsen in de achterliggende foto's en van elke foto bepalen "hoe rood" die is, kennen de roodwaarde van je vingertop in functie van de tijd. Dit is een grafiek, die net zoals de grafiek in hieronder (die je opmat via *Science Journal*) een piek heeft voor elke hartslag. Door het aantal pieken in een beperkt tijdsinterval (bv 15 s) te tellen en dit aantal te herleiden naar het aantal pieken per minuut (bv door het aantal pieken in 15 s te vermenigvuldigen met vier), kan je de hartslag berekenen.



Om het project tot een goed einde te brengen, moeten we dus “de roodwaarde” van een foto kunnen berekenen. De roodwaarde van een foto zullen we bepalen als het gemiddelde van de roodwaarde van elke pixel van de foto.

■ Opdracht: roodwaarde van een foto bepalen in App Inventor 2 ■

Bepaal de roodwaarde van een foto in App Inventor 2.

Volg daarvoor de volgende stappen:

1. Voeg een *canvas(E)/doek(NL)* component toe aan je scherm.
2. Upload een foto naar keuze en stel de foto in als achtergrond van het canvas. Pas de afmetingen van het canvas aan zodat de verhoudingen van de foto goed zijn.
Tip: Voor de meeste rechtopstaande foto's is de verhouding tussen de hoogte en de breedte 16:9 (voor liggende foto's 9:16). Als de hoogte van een rechtopstaande foto vastgelegd wordt op 200 pixels, moet de breedte dus ongeveer $\frac{9}{16} \cdot 200$ pixels zijn om de oorspronkelijke verhouding te respecteren.
3. Voeg een knop en een label component toe.
4. Ga nu naar de Designer view. Voeg de nodige blokken toe zodat de roodwaarde van de foto berekend wordt als de knop ingedrukt wordt. De berekende roodwaarde verschijnt in de label component.
 - *Tip:* De canvas component heeft een methode/blok waarmee je de RGB-waarde van een pixel op breedte x en hoogte y in het rooster van de foto kan berekenen. De RGB-waarde wordt opgeslaan als een lijst met drie elementen.

- *Tip:* Om de roodwaarde te bekomen, selecteer je het eerste element uit de lijst met de drie kleurwaarden. Je krijgt dus de R van de RGB-waarde.
- *Tip:* Om de gemiddelde roodwaarde van een foto te berekenen, gebruik je twee for-lussen: één lus loopt over alle pixels in de breedte van een foto (x varieert), de andere lus loopt over alle pixels in de hoogte van de foto (y varieert). De grenzen waarbinnen x en y variëren, kan je bepalen a.d.h.v. de breedte en de hoogte van je canvas component.
- *Tip:* om een gemiddelde van een rij getallen te berekenen, maak je eerst de som van alle getallen, en die som deel je vervolgens door het aantal getallen dat je opgeteld hebt:

$$\text{Gemiddelde van } 2, 7 \text{ en } 10 = \frac{2 + 7 + 10}{3} = 6,33$$

Iets algemener, voor het gemiddelde van 5 willekeurige getallen:

$$\overline{a_5} = \text{Gemiddelde van } a_1, a_2, \dots, a_5 = \frac{a_1 + a_2 + \dots + a_5}{n} = \frac{\sum_{i=1}^5 a_i}{5}$$

Nog algemener, voor het gemiddelde van n willekeurige getallen:

$$\overline{a_n} = \text{Gemiddelde van } a_1, a_2, \dots, a_n = \frac{a_1 + a_2 + \dots + a_n}{n} = \frac{\sum_{i=1}^n a_i}{n}$$

Opmerking. *De gemiddelde roodwaarde van een foto berekenen kan even duren!*

Daarom gaan we in de toekomst niet werken via deze omslachtige manier, maar gebruiken we een apart blok dat een foto als input neemt en meteen de gemiddelde roodwaarde van de foto als output teruggeeft.

2 Digitale video

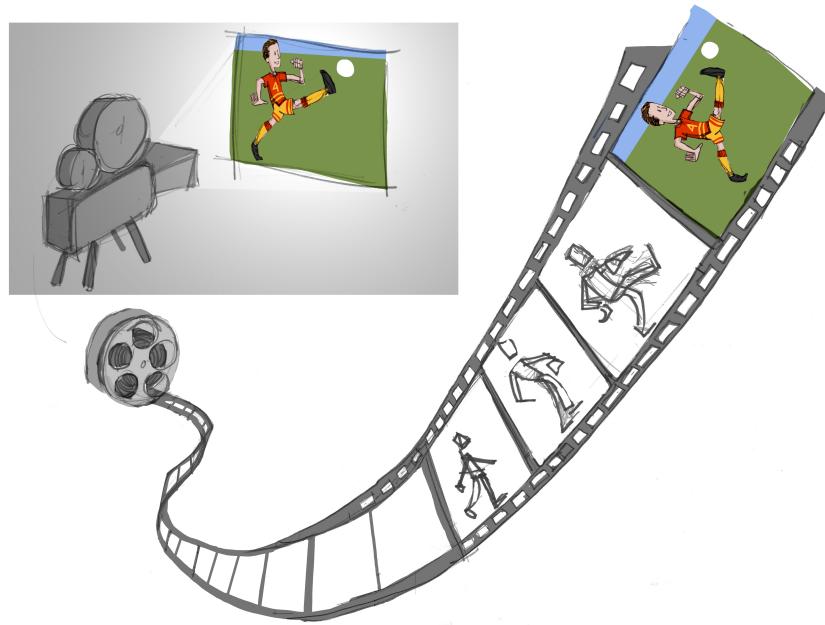
In onderstaand filmpje (Engelstalig, met de mogelijkheid te ondertitelen en indien nodig te vertragen) worden de basisprincipes achter digitale video uitgelegd.



We vatten de inhoud van het filmpje hieronder nog even samen. Zoals eerder gezegd is digitale video een opeenvolging van digitale afbeeldingen, die snel na elkaar getoond worden met een vaste snelheid. De digitale afbeeldingen waaruit een video is opgebouwd, noemen we frames. De snelheid waarmee frames na elkaar getoond worden is de frame rate, uitgedrukt in frames per seconde (of kortweg *fps*). Onderzoek heeft aangetoond dat een gemiddelde mens individuele afbeeldingen niet meer kan onderscheiden als ze op een snelheid van 24 *fps* of sneller getoond worden.

- 2** Hoelang is elke afbeelding zichtbaar als afbeeldingen in een video elkaar opvolgen aan een snelheid van 24 *fps*? En wat als de frame rate 30 *fps* is?

Elk frame heeft een breedte B en een hoogte H , uitgedrukt in pixels. We weten uit de vorige sectie dat een pixel in feite het kleinste bouwblokje van een digitale afbeelding is en dat elk pixel een RBG-waarde heeft.



De hoogte H bepaalt de kwaliteit van het beeld. Hoe groter H , hoe scherper het beeld. Sommigen onder jullie hebben misschien al gehoord van standard definition (SD), high definition (HD) en full-HD beeldresolutie. De hoogte H van de frames bepaalt of de beeldresolutie SD (als $H = 480$ pixels), dan wel HD (als $H = 720$ pixels), dan wel full-HD (als $H = 1080$ pixels) is.

Encoding is een proces dat gebruikt wordt om de grootte van de video te reduceren m.b.v. wiskundige bewerkingen. Met de grootte van de video bedoelen we hoeveel ruimte nodig is op je computer of smartphone om de video op te slaan. De video-grootte wordt meestal uitgedrukt in megabytes (MB). Ongecomprimeerde video is video waarop geen encoding is toegepast en neemt veel geheugenruimte in op je smartphone of computer. Encoding past een wiskundige bewerking toe op je video die ervoor zorgt dat de geheugenruimte beperkt blijft, zonder de kwaliteit van de video te veel aan te tasten. Afhankelijk van de gebruikte wiskundige bewerkingen krijgen we een ander encodingsformaat. Vaak gebruikte encodingsformaten zijn .mp4 en .mov, maar er bestaan nog veel andere formaten.

Opdracht: een willekeurig filmpje splitsen in foto's

Pluk een filmpje van het internet of maak zelf een filmpje. Gebruik de app “Video naar afbeeldingen” om het filmpje te splitsen in opeenvolgende foto's.

Opmerking. *Het splitsen van video in afbeeldingen kan even duren!*

De foto's worden na het splitsen getoond. Deze foto's worden ook apart op de smartphone opgeslaan. De map waar de foto's worden opgeslaan wordt ook weergegeven. Je kan deze foto's dus ook achteraf nog raadplegen of gebruiken.

Als je een willekeurig filmpje in foto's splitst, zie je hoe de foto's veranderen doorheen de tijd. Als de foto's snel genoeg na elkaar getoond worden, kan je oog de individuele foto's niet meer onderscheiden, waardoor je een continue beweging ziet.

Opdracht: een filmpje van je vingertop splitsen in foto's

Maak een filmpje van 15 s van de vingertop van je wijsvinger. Zet de flits van je smartphone aan. Leg je vinger op de lens van je camera en zorg dat je vinger de lens volledig bedekt. Het beeld zou volledig rood moeten zijn. Laat de camera gewoon rusten op je vingertop, je hoeft je vinger niet tegen de lens te duwen.

Splits het filmpje in foto's.

Zie jij de kleurverschillen ten gevolge van je hartslag?

3 Toepassing: de hartslagmonitor

Opdracht: roodwaarden bepalen en weergeven in MS Excel

In de vorige twee opdrachten hebben jullie een filmpje gemaakt van je vingertop en dit gesplitst in afzonderlijke afbeeldingen. Daarnaast hebben jullie geleerd hoe je in app inventor de roodwaarde van een individuele foto kan bepalen. Hier zullen we die twee combineren om de roodwaarden van de afzonderlijke foto's van je filmpje te bepalen. Die waarden slaan we op, en analyseren we in MS Excel, zodat we de hartslag kunnen berekenen.

1. Maak een filmpje van 15 s van je vingertop of gebruik het filmpje uit de vorige opdracht. Zet de flits van je smartphone aan. Leg je vinger op de lens van je camera en zorg dat je vinger de lens volledig bedekt. Het beeld zou volledig rood moeten zijn. Laat de camera gewoon rusten op je vingertop, je hoeft je vinger niet tegen de lens te duwen.
2. Splits het filmpje in foto's met de app “Video naar afbeeldingen”.
Als je dit in de vorige opdracht al gedaan hebt, mag je deze stap overslaan. Noteer de map waarin de afzonderlijke digitale afbeeldingen werden opgeslaan, dit heb je straks nodig.

3. Maak een nieuw project aan in MIT App Inventor 2 en geef het een passende naam, bv `Hartslagmonitor_redToExcel`.
4. Schrijf een app die, wanneer je de knop *BerekenRoodwaarden* klikt, de roodwaarde van alle digitale afbeeldingen in de map berekent en toevoegt aan een lijst. Zet vervolgens deze lijst om naar een CSV-tabel en sla de tabel op op je smartphone.

- Voeg de knoppen/labels/... die je denkt nodig te hebben toe aan je scherm.
- Wanneer de knop *BerekenRoodwaarden* wordt ingedrukt, moet je de roodwaarden van alle afbeeldingen in de map bepalen. Maak daarom eerst een lijst van alle afbeeldingen die in de genoteerde map opgeslaan zijn.

Hiervoor moet je een extensie gebruiken: *TaifunFile*. Voeg in de *Designer-view* de extensie toe in de tab *Palette* via *Import extension*. Sleep de extensie dan naar het afgebeelde smartphone-scherm. De extensie zou onder de afgebeelde smartphone moeten verschijnen.

In de *Blocks-view* gebruik je het blok *TaifunFile - FileListAsync* om alle bestanden in de genoteerde map op te lijsten. Dit blok gaat alle bestanden in de gevraagde map *directoryName* en met de gevraagde extensie *extension* oplijsten. Als je bij *includeSubDirectories* **true** zet, worden ook submappen meegenomen in de oplijsting. Als *includeSubDirectories* **false** is, wordt enkel de map zelf bekeken.

De oplijsting is *asynchroon*: dat wil zeggen dat de smartphone andere processen kan uitvoeren tijdens de oplijsting. Maar dat betekent ook dat we niet weten wanneer de bestanden opgelijst zijn. Daarom gebruiken we ook het blok *TaifunFile - After-FileListAsync*. Dit blok wordt geactiveerd wanneer alle bestanden opgelijst zijn, en dan kunnen we verder met onze berekeningen. Binnen dit blok hebben we een variabele *fileList* waarin alle bestanden in de gevraagde map met de gevraagde extensie opgenomen zijn.

Ga even na dat de bestanden in de gevraagde map correct opgelijst worden, door de variabele *filelist* als label te tonen op het scherm van je smartphone. Ga pas verder naar de volgende stap als dit correct werkt.

- Je zal merken dat de bestanden niet noodzakelijk in numeriek oplopende volgorde in *fileList* staan. Daarom gaan we de lijst eerst sorteren. We willen immers eerst de roodwaarde van de eerste foto (met naam 001.jpg) berekenen, vervolgens van de tweede foto (met naam 002.jpg), enzovoort.

Om de lijst te sorteren, hebben we een tweede extensie gemaakt: *ListProcessing*. Voeg de extensie toe zoals voorheen via *Import extension* en door de extensie naar het scherm van de afgebeelde smartphone te slepen.

Gebruik het blok *ListProcessing - SortList* om de lijst te sorteren. Ga na dat dit correct gebeurde door de bekomen lijst als label te tonen op het scherm van je smartphone. Ga pas verder naar de volgende stap als dit correct werkt.

- Itereer over de items/bestanden in de lijst met gesorteerde bestanden en bereken van elk bestand (en dus van elke afbeelding) de roodwaarde. Voeg de bekomen roodwaarden telkens toe aan een lijst, die je bijvoorbeeld *redValueList* noemt.

Ga even na dat de roodwaarden berekend worden, door de variabele *redValueList* als label te tonen op het scherm van je smartphone. Ga pas verder naar de volgende stap als dit correct werkt.

- Ten slotte willen we de roodwaarden opslaan in een bestand dat we kunnen inlezen in MS Excel. Daarom moet de lijst omgezet worden in een CSV tabel. CSV staat voor *Comma Separated Value*. Dit is een formaat dat vaak gebruikt wordt om gegevens op te slaan in een tekstbestand, dat eenvoudig kan ingelezen worden in MS Excel.
Voeg een nieuwe *File*-component toe aan je scherm. De component zou onder de afgebeelde smartphone moeten verschijnen.

Gebruik het blok *File - SaveFile* en het blok *List - list to csv table* om de lijst met roodwaarden (*redValueList*) eerst om te zetten in een CSV tabel. Die CSV tabel is de *text* input van het blok *SaveFile*. De *filename*-input mag je zelf kiezen, maar zorg dat de bestandsnaam eindigt op “.csv”. Het csv-bestand zou moeten opgeslaan worden op je smartphone in de map AppInventor/data.

Ga na dat het CSV-bestand inderdaad op de voorziene plaats en onder de voorziene naam opgeslagen is op je smartphone. Je kan navigeren door de bestanden op je smartphone via de app “Bestanden”, die normaal gezien standaard geïnstalleerd is op Android smartphones.

5. Kopieer het CSV-bestand van je smartphone naar je computer. Je kan dit doen door het bestand naar jezelf te mailen of het bestand op te laden naar je google drive en dan te downloaden vanaf je computer. Alternatief sluit je je smartphone aan op je computer via een USB-kabel en kopieer je het bestand rechtstreeks.

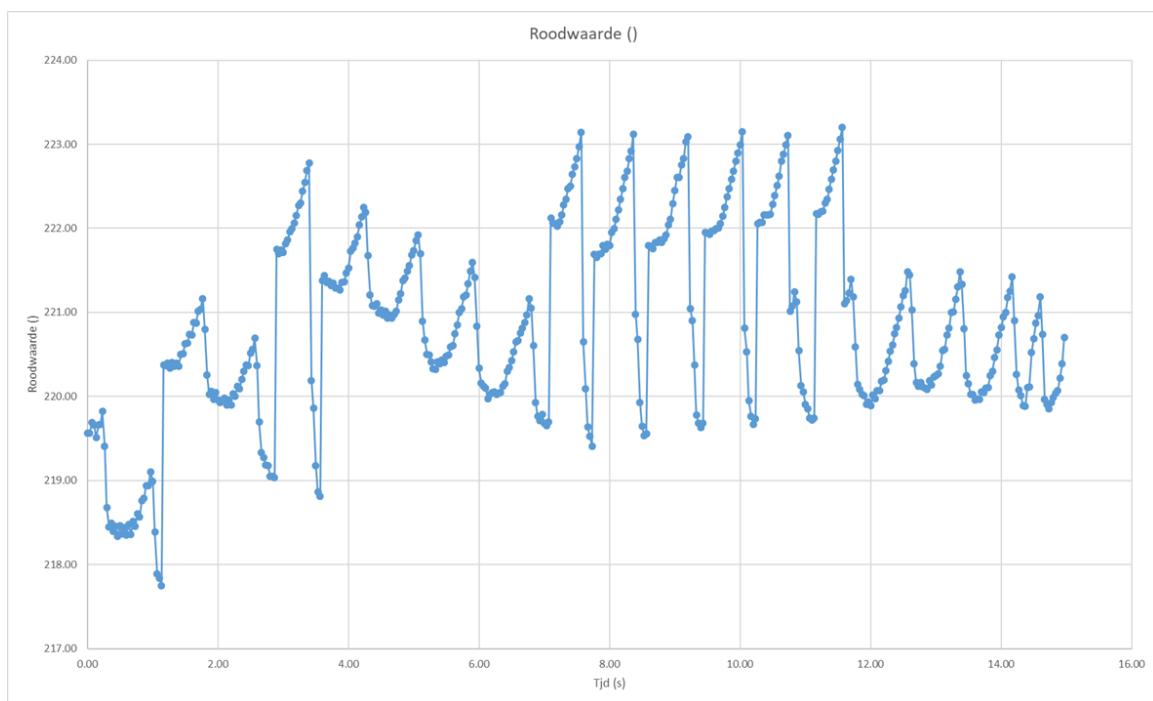
Het CSV-bestand bevat zoals gezegd de roodwaarden van de digitale afbeeldingen waaruit de video bestaat. Je kan deze waarden inlezen in MS Excel.

- Open een nieuwe MS Excel werkmap.
- Lees de gegevens in via het tabblad *Gegevens - Van tekst*. Verplaats de roodwaarden naar kolom B. Voeg bovenaan een lege rij toe en geef kolom B een passende titel.
- Voeg de tijdstippen toe in kolom A.

De eerste foto werd geregistreerd op tijdstip 0,00. Gebruik de frame rate om de registratietijden van de volgende foto's toe te voegen. Ga indien nodig even terug naar oefening 1 in Sectie 2 om de duurtijd per foto d te berekenen a.d.h.v. de frame rate.

Als je de frame rate niet kent, kan je die berekenen. Hiervoor moet je weten hoelang je filmpje duurt. Duid dit aan met D . Dit kan je zien op je smartphone. In MS Excel kan je nagaan hoeveel foto's geëxtraheerd werden. Dit duid je aan met N . De totale duur van de video D gedeeld door het aantal foto's in de video N geeft je het aantal seconden per foto/frame $d = \frac{D}{N}$.

- Geef kolom A een passende titel.
- Maak een grafiek die de roodwaarde in functie van de tijd weergeeft. Welke kolom levert de waarden die op de x -as komen? En welke gegevens komen op de y -as?
Hieronder zie je een voorbeeld.



- Tel het aantal pieken in de grafiek. Noteer ook hoelang het filmpje duurde. Gebruik beide gegevens om de hartslag, het aantal pieken per minuut, te berekenen.
In de grafiek hierboven tellen we 19 pieken op 15 seconden. De hartslag is dus $19 \cdot 4 = 76$ bpm.
- Als de grafiek niet mooi genoeg is om pieken te kunnen tellen, maak dan een nieuw filmpje en herhaal de bovenstaande stappen.
Ga na welke factoren (bv flits aan/uit, hoe hard je op de lens drukt, hoe je je vinger houdt, ...) de kwaliteit van de meting beïnvloeden. Hou hiermee rekening als je later nog pogingen onderneemt om de hartslag te meten. Schrijf indien nodig je bevindingen ergens op.

Opdracht: roodwaarden bepalen en weergeven in de app

In de vorige opdracht konden we voor het eerst onze hartslag bepalen o.b.v. een opgenomen video. De omweg via MS Excel is wel een beetje vervelend. In deze opdracht wijzigen we de app uit de vorige opdracht, zodat we de grafiek rechtstreeks in onze app kunnen maken en bekijken.

1. Kopieer het vorige project en geef het een passende naam, bv `Hartslagmonitor_redToGraph`.
2. Om de tijd op de x -as correct te kunnen berekenen, moeten we de frame rate kennen. In de meeste nieuwe smartphones is de frame rate $30\text{ }fps$. Dit is dus de standaardwaarde die we in de app zullen gebruiken.

We moeten wel functionaliteit aan de app toevoegen die toelaat de frame rate te veranderen, mocht die niet overeenkomen met onze standaardwaarde van $30\text{ }fps$. Voeg daarom

een tekstinput *fsInput* en een knop *fsChanged* toe aan je app. Wanneer je de tekstin-put verandert, moet je de op knop *fsChanged* drukken, waarna een globale variabele *fs* aangepast wordt.

In de vorige opdracht berekende je de frame rate voor de smartphone die jij gebruikte. Ga na of je de standaardwaarde van 30 *fps* inderdaad moet veranderen.

3. Voeg een nieuwe knop *plotValues* en een webviewer toe. Voeg ook de extensie *Chartmaker* toe door de extensie te importeren en dan de componenten te slepen naar het afgebeelde smartphonescherm.

Gebruik het blok *Chartmaker - DrawLineGraph* om een lijngrafiek te maken. Je kan de grafiek een titel, *x*- en *y*-labels geven. De input *scalingFactor* geeft aan met welke stapgrootte de waarden op de *x*-as veranderen. Dit is de duurtijd per foto *d*, die je in de vorige opdracht bepaalde. De input *maxXValue* limiteert het bereik dat afgebeeld wordt op de *x*-as. Ten slotte moet je een label-naam meegeven, welke waarden je wilt uitzetten in de grafiek, en in welke webviewer je de grafiek wilt weergeven.

Controleer dat de grafiek correct wordt weergegeven alvorens verder te gaan!

Opmerking. Als je geen internet connectie hebt, zal je de grafiek niet kunnen weergeven!

Module 4

Automatische piekdetectie

In de vorige modules leerde je werken met App Inventor 2 en leerde je de basis van digitale afbeeldingen en video. In de slotopdracht van module 3 maakte je een grafiek waaruit je je hartslag kon afleiden door zelf pieken te tellen. In deze module gaan we na hoe de smartphone zelf die pieken kan detecteren, zodat wij van dat rekenwerk gespaard blijven. Daarvoor staan we stil bij wat een piek precies is, en hoe we de smartphone kunnen leren pieken te detecteren.

1 Wat is een piek in een signaal?

I.p.v. zelf pieken te tellen op een grafiek willen we dit automatisch doen in de app. Dit houdt in dat we de code moeten schrijven om pieken te detecteren. Door onze ervaringen in het dagelijkse leven kunnen wij mensen redelijk intuïtief pieken herkennen. De smartphone daarentegen moet door ons gezegd worden wanneer hij een extra piek moet tellen. Daarom moeten we zelf ook even stilstaan bij wat een piek nu eigenlijk is.

- * 1 Omschrijf in eigen woorden wat een piek is.

Een woordelijke omschrijving van een piek hebben we gevonden, maar je smartphone redeneert wiskundiger.

- 2 Stel een wiskundige uitdrukking op die je in de code kan gebruiken om pieken te detecteren.

Hou er rekening mee dat je smartphone enkel een lijst met roodwaarden als input heeft. Die lijst kunnen we voorstellen als een verzameling $\{u_1, u_2, u_3, \dots, u_n\}$ met n elementen, waarbij u_i de i -de berekende gemiddelde roodwaarde voorstelt.

Als je dit niet meteen ziet, kan je zelf een aantal voorbeeldlijsten maken en a.d.h.v. de voorbeelden nagaan wanneer het signaal een piek bereikt.

Automatische piekdetectie in App Inventor 2

Schrijf een app die automatisch pieken detecteert in een lijst. Hou bij (1) op welk volgnummer i een piek u_i gedetecteerd wordt, (2) hoeveel pieken in totaal gedetecteerd worden. Gebruik labels om die resultaten op je scherm weer te geven ter controle.

1. Gebruik eerst een lijst die je zelf ingeeft met *Lists - make a list*. Worden de pieken gedetecteerd op de plek die je verwacht? Pas je code aan tot je het verwachte resultaat bekomt.

2. Gebruik nu een lijst met berekende roodwaarden, die we in de vorige module hebben opgeslaan in een CSV-bestand. Voeg daarom een *File*-component toe je aan je app. Wanneer een knop *Read CSV* ingedrukt wordt, moet het CSV-bestand die je opgeslagen hebt in de vorige module ingelezen worden. Kijk indien nodig even terug in je vorige project waar en onder welke naam je het CSV-bestand hebt opgeslagen. Als het CSV-bestand is ingelezen, moet de inhoud omgezet worden in een lijst en in een lijst geplaatst worden. Geef die lijst weer in een label en vergelijk de tekst met het CSV-bestand op een computer om te controleren dat dit correct gebeurde.

Opmerking. Afhankelijk van het CSV-bestand dat je inleest, kan het zijn dat de lijst met roodwaarden in plaats van getallen als elementen lijsten met 1 element als elementen bevat. Om aan de roodwaarde te raken, moet je dan het eerste element halen uit elke lijst (die ook weer een element van de oorspronkelijke lijst is).

3. Gebruik de code uit stap 1 om automatisch pieken te detecteren in de lijst met roodwaarden. Komt dit overeen met het aantal pieken dat jij telt. Waarom (niet)?

Opmerking. Vaak is de lijst met roodwaarden een beetje “ruizig”. Dit zorgt ervoor dat de app meer pieken detecteert dan wij zouden doen. Soms zien we een klein piekje in het signaal, dat we negeren, omdat het maar een kleine piek is, of omdat de piek te snel na een grote hartslagpiek komt.

4. We moeten de app dus leren dat sommige pieken moeten genegeerd worden, wanneer de piek niet echt hoog is, of heel snel komt. Hoe zou jij dit implementeren? Je hoeft dit niet meteen te doen, maar zorg wel dat je een plan hebt.

5. De makkelijkste manier om de kleine piekjes te negeren is door de lijst met roodwaarden een beetje op te kuisen. De lijst met roodwaarden noemen we het signaal. De ruis die aanwezig is, is ongewenst. De ruis verandert vaak sneller dan het signaal zelf. Daardoor kunnen we de ruis verminderen door die snelle veranderingen weg te werken. Een eenvoudige manier om snelle veranderingen weg te werken is door opeenvolgende roodwaarden uit te middelen. Dit wordt ook wel laagdoorlaatfiltering genoemd: de snelle veranderingen, vaak ook hoge frequenties genoemd, worden weggefilterd uit het signaal.

Schrijf code om opeenvolgende roodwaarden uit te middelen. Maak een nieuwe lijst, bv *smoothRedValuesList*, waarin elk element een gemiddelde is van vier opeenvolgende elementen uit de originele lijst. Herinner je dat we een gemiddelde berekenen als een som van elementen, gedeeld door het aantal elementen dat opgeteld wordt. Voor het gemiddelde van 4 willekeurige getallen:

$$\overline{a_4} = \text{Gemiddelde van } a_1, a_2, a_3, a_4 = \frac{a_1 + a_2 + a_3 + a_4}{4} = \frac{\sum_{i=1}^4 a_i}{4}$$

Opmerking. Tip: gebruik een for-lus om de elementen van de nieuwe lijst *smoothRedValuesList* te berekenen.

6. Gebruik de code uit stap 1 om de pieken in de nieuwe lijst *smoothRedValuesList* te berekenen. Bekom je een aanvaardbaar resultaat? Is het nodig meer of minder elementen van de originele lijst uit te middelen om tot een aanvaardbaar resultaat?

Opmerking. We spreken hier over een aanvaardbaar resultaat en niet over een correct resultaat. Het is goed mogelijk dat het aantal pieken dat je code detecteert licht afwijkt van het aantal pieken dat jij telt. Hoe sterk de afwijking is, hangt af van de kwaliteit van de opgenomen video. In Module 3 Sectie 3 gingen jullie al na welke factoren invloed hebben op de kwaliteit van de video.

7. Als je tevreden bent over de automatische piekdetectie, kan je het aantal pieken gedetecteerd tijdens de duur van de video herleiden naar een waarde voor de hartslag, het aantal pieken per minuut.

Gebruik de frame rate en het aantal elementen in de lijst met roodwaarden om de duur van de video te berekenen. Gebruik de regel van drie om het aantal pieken gedurende de videoduur te herleiden naar het aantal pieken per seconde. Dit is de gemeten hartslag. Geef die waarde weer in een label op het scherm. Bekom je een realistische waarde voor de gemeten hartslag?

2 Intermezzo: differentiequotiënt

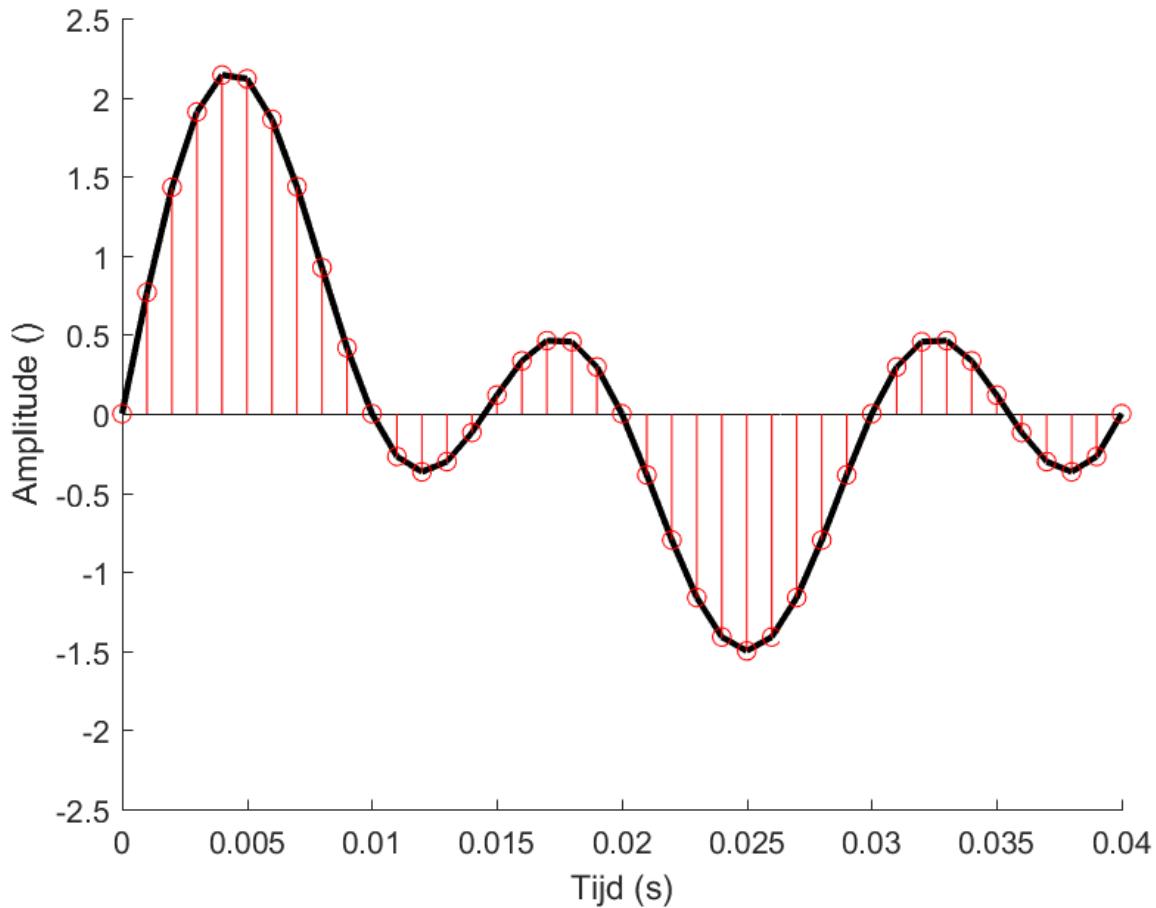
Bedoeld voor leerlingen uit de 3e graad.

Voor de bepaling of een piek in het signaal bereikt wordt, maakte je gebruik van een wiskundige uitdrukking gerelateerd aan het differentiequotiënt.

Stel dat $\{u_1, u_2, \dots, u_n\}$ een signaal is, bv. een lijst met roodwaarden.

Opmerking. We noemen dit een discreet signaal (in de figuur hieronder in rood aangeduid): de meetwaarden (bv roodwaarden) zijn slechts op bepaalde tijdstippen bepaald.

Naast discrete signalen bestaan ook continue signalen (in de figuur hieronder in zwart aangeduid), waarvoor op elk moment een meetwaarde beschikbaar is. Het signaal loopt dus - zoals de naam het zegt - continu door.



Vaak spreekt men ook over digitale en analoge signalen. Digitale signalen zijn vergelijkbaar met discrete signalen, terwijl analoge signalen te vergelijken zijn met continue signalen, hoewel niet exact dezelfde zaken bedoeld worden met de termen digitaal/discreet en analoog/continu.

- 3** In de lessen wiskunde leerde je al over functies. Een grafiek van de functie met functievoorschrift $y = f(x)$ toont hoe de grootheid y verandert i.f.v. de grootheid x . Vaak kan de grootheid x een continu bereik van waarden aannemen. In dat geval spreken we over een discreet/continu signaal. (*Schrap wat niet past.*)

De signalen waar we hier mee werken, kan je dus enigszins vergelijken met functies die je kent uit de wiskundelessen. Het verschil is dat we hier slechts op discrete momenten een functiewaarde kunnen bepalen, terwijl in de wiskundeles meestal met continue signalen gewerkt wordt.

We gaan terug naar ons discreet signaal $\{u_1, u_2, \dots, u_n\}$. De uitdrukking die je kan gebruiken om na te gaan of een piek bereikt was is:

$$u_i - u_{i-1}$$

Immers, als $u_i - u_{i-1} > 0$ en $u_{i+1} - u_i < 0$ wordt een lokaal maximum, of een piek, bereikt op het tijdstip i .

Deze uitdrukking is sterk gerelateerd aan het differentiequotiënt D voor een functie f in het punt $(a, f(a))$:

$$D = \frac{f(a + \Delta h) - f(a)}{\Delta h} \quad (1)$$

Voor zeer kleine waarden van Δh benadert het differentiequotiënt de richtingscoëfficiënt van de raaklijn in het punt $(a, f(a))$. We zeggen ook weleens dat de afgeleide van de functie f in het punt $(a, f(a))$ gelijk is aan

$$\begin{aligned} f'(a) &= \lim_{\Delta h \rightarrow 0} D \\ &= \lim_{\Delta h \rightarrow 0} \frac{f(a + \Delta h) - f(a)}{\Delta h} \end{aligned}$$

Uit de wiskundelessen weet je dat een functie een extremum (= een maximum of een minimum) bereikt waar de afgeleide nul wordt. Om een *minimum* te vinden zoeken we voor welke waarde(n) van x de functie overgaat van dalend (afgeleide $f'(x) < 0$) naar stijgend (afgeleide $f'(x) > 0$). Om een *maximum* te vinden zoeken we voor welke waarde van x de functie overgaat van stijgend (afgeleide $f'(x) > 0$) naar dalend (afgeleide $f'(x) < 0$).

Wat we in dit project doen, is zeer gelijkaardig: we kijken voor welke waarde van i de waarde van $u_i - u_{i-1}$ overgaat van een positieve naar een negatieve waarde om een piek (= een maximum) te detecteren.

- 4** Nu we achterhaald hebben hoe onze aanpak in dit project vergelijkbaar is met afgeleiden en extrema-vraagstukken, kunnen we ons ook afvragen welke verschillen bestaan. Som op welke verschillen je ziet.

3 Intermezzo: Frequentieanalyse

Bedoeld voor leerlingen uit de 3e graad. ¹

Je zal al ondervonden hebben dat de nauwkeurigheid van de hartslagmeting afhangt van de kwaliteit van het opgenomen filmpje. Als je vingertop of de smartphone bewoog tijdens de opname, is de grafiek van de roodwaarde in functie van de tijd ruiziger. Hoe ruiziger het signaal, hoe moeilijker de detectie van pieken wordt. En hoe moeilijker de piekdetectie, hoe meer kans op fouten en hoe onnauwkeuriger de hartslagmeting.

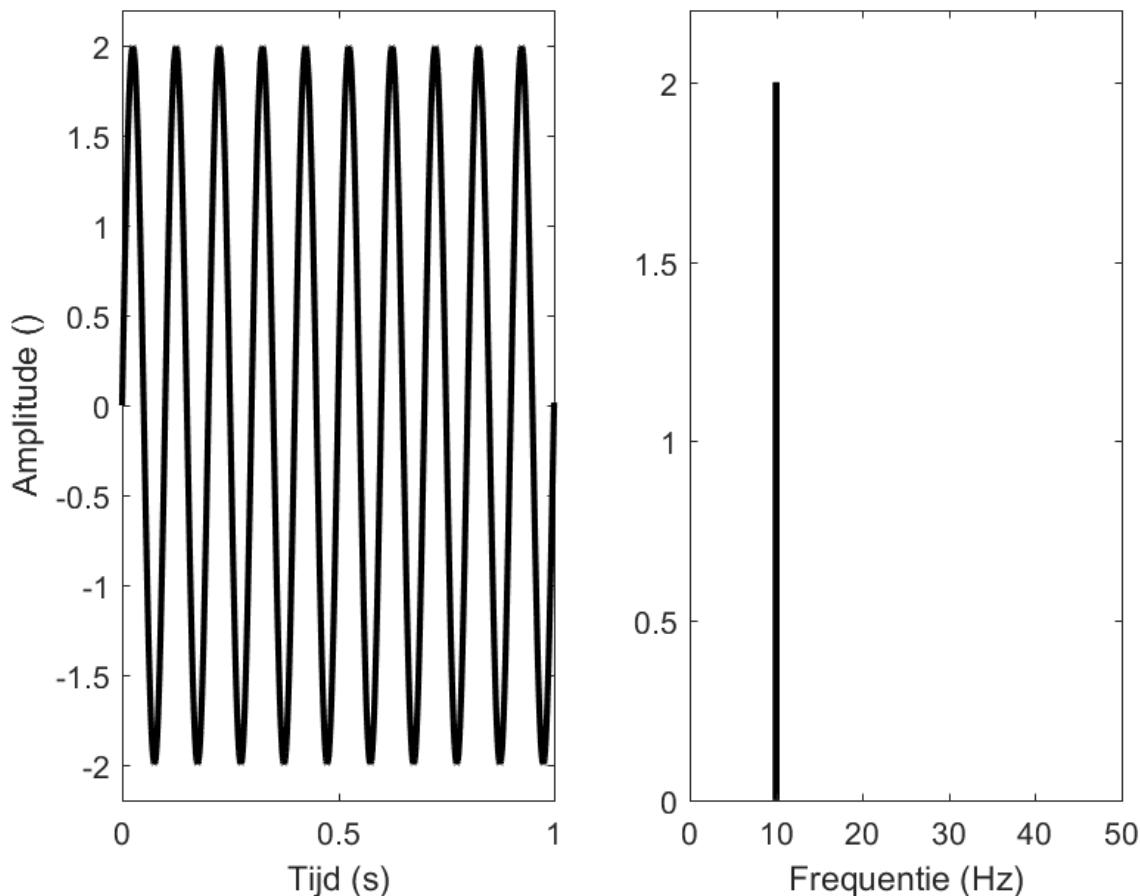
Er bestaat een andere manier om de hartslag te bepalen, die niet gebaseerd is op piekdetectie en daardoor iets nauwkeuriger is. Deze alternatieve manier is gebaseerd op frequentieanalyse.

¹Gebaseerd op de Junior College STEM over cochleaire implantaten.

3.1 Geluid: zuivere en samengestelde tonen

We nemen even afstand van onze hartslagmonitor. Om goed te kunnen uitleggen wat we met frequentieanalyse bedoelen, kijken we even naar geluid.

Geluid is een kleine verandering in luchtdruk, die zich door de lucht voortplant. Elk luchtdeeltje ondervindt een trilling en geeft die trilling door aan de luchtdeeltjes in zijn buurt. De trilling van een luchtdeeltje in functie van de tijd kan er uitzien zoals hieronder links.



De trilling hierboven (links) is een harmonische trilling, die als een sinusvormig signaal voorgesteld wordt en met als functievoorschrift

$$y(t) = A \sin(2\pi ft).$$

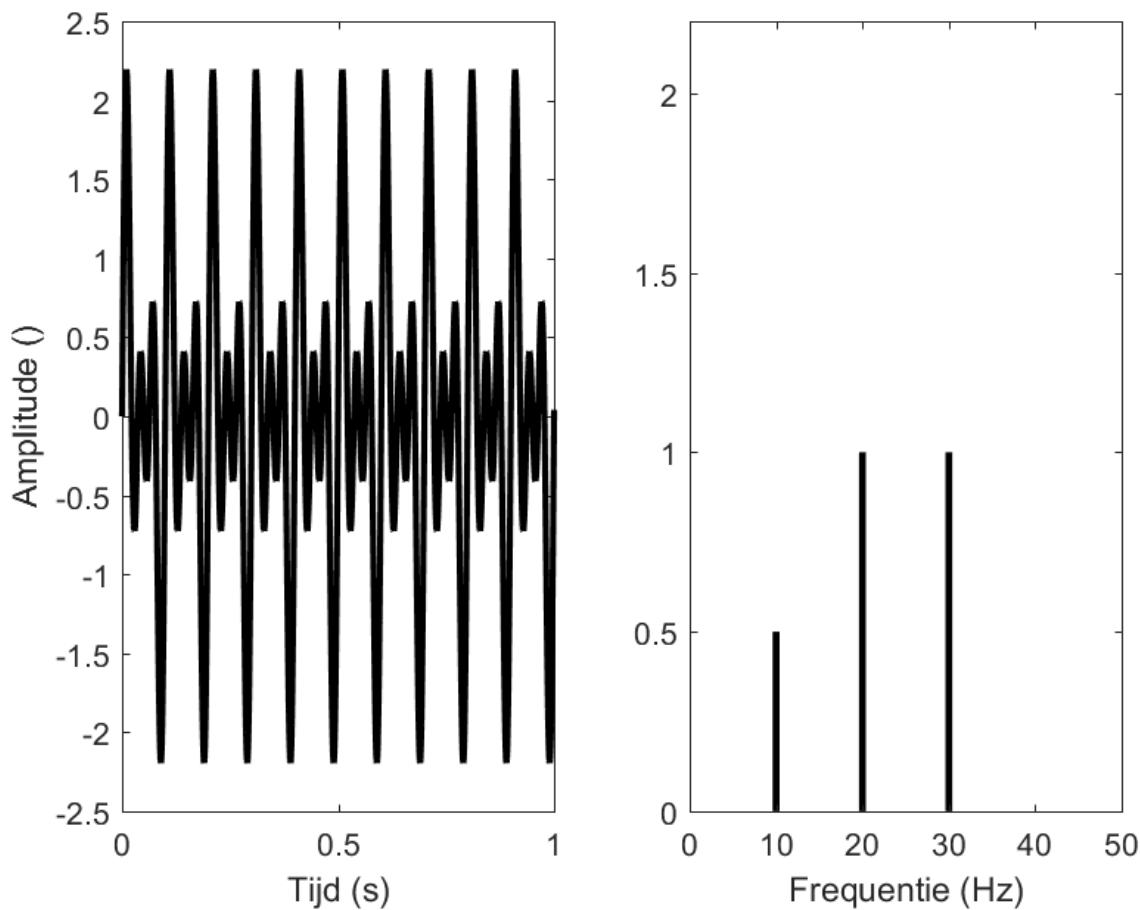
Links is de trilling voorgesteld in het tijdsdomein: de uitwijking in functie van de tijd is een sinusoidaal verloop. Rechts is de trilling voorgesteld in het frequentiedomein. Deze voorstelling noemen we dan ook het *frequentiespectrum*. De frequentie van een trilling is gedefineerd als het aantal trillingen per seconde. De eenheid is $1/s$ of Hertz (Hz). Hoe het frequentiespectrum gevonden wordt, bespreken we later.

Deze trilling is een *zuivere toon*: er is slechts 1 frequentie aanwezig. In het frequentiespectrum zien we maar 1 lijn. Het is het geluid dat je hoort als je een stemvork aanslaat.

5

- Wat is de uitwijking, of amplitude A van de trilling?
- Hoe verandert het geluid als de amplitude verhoogt?
- Wat is de frequentie f van de trilling? Vergeet de eenheid niet!
- Hoe verandert het geluid als de frequentie verhoogt?
- De periode is $T = 1/f$ is het tijdsverschil tussen 2 opeenvolgende maximale (of minimale) uitwijkingen. Het is dus de tijd die 1 volledige trilling inneemt.

Naast zuivere tonen bestaan ook samengestelde tonen. De meeste geluiden die je kent, zijn samengestelde tonen, waarin meerdere frequenties voorkomen. Hieronder zie je een voorbeeld.



Het functievoorschrift van de afgebeelde trilling is

$$y(t) = 0.5 \sin(2\pi 10t) + 1 \sin(2\pi 20t) + 1 \sin(2\pi 30t).$$

De amplitude en frequentie van de samengestelde toon kan je uit het functievoorschrift afleiden: $A_1 = 0.5$, $f_1 = 10$ Hz, $A_2 = 1$, $f_2 = 20$ Hz, $A_3 = 1$ en $f_3 = 30$ Hz. Uit de linkergrafiek in het tijdsdomein kunnen die grootheden echter niet zo eenvoudig afgeleid worden. Gelukkig toont het frequentiespectrum ons wel de nodige informatie: de samengestelde trilling is opgebouwd

uit drie zuivere trillingen, één op 10 Hz met amplitude 0.5, één op 20 Hz met amplitude 1 en één op 30 Hz met amplitude 1.

In het algemeen kan een samengestelde toon geschreven worden als een som van N zuivere tonen, elk met amplitude A_n en frequentie f_n :

$$y(t) = \sum_{n=1}^N A_n \sin(2\pi f_n t).$$

3.2 Frequentieanalyse: het idee van Fourier

Meer algemeen kan je om het even welk periodiek signaal opbouwen door sinussen met veelvouden van een bepaalde grondfrequentie f_0 en verschillende amplituden op te tellen. Naarmate je meer termen optelt, krijgt het signaal een strakkere vorm.

Je kan elk periodiek signaal dus schrijven als een reeks

$$y(t) = a_0 + \sum_{k=1}^{\infty} (a_k \cos(2\pi k f_0 t) + b_k \sin(2\pi k f_0 t)).$$

Frequentieanalyse of fourieranalyse is het ontleden van een oneindig lang periodiek signaal in de verschillende frequentiecomponenten waaruit het bestaat.

Fourieranalyse bestaat er dus in de coëfficiënten a_0 , a_k en b_k te zoeken, die bij elke frequentie $k f_0$ horen.

Fourieranalyse op geluidsignalen toont ons welke frequenties in een geluid voorkomen.

3.3 Toepassing: fourieranalyse voor onze hartslagmonitor

Terug naar onze hartslagmonitor. Wij zijn op zoek naar onze hartslag: het aantal keer dat ons hart klopt per minuut. Dit is eigenlijk ook een soort van frequentie: het kloppen van ons hart kunnen we vergelijken met een geluidstrilling. Elke keer ons hart samentrekt, bevindt het bloed in onze vingertop zich iets verder van de huid en is de roodwaarde in onze video minimaal. En elke keer ons hart maximaal uitgezet is, bevindt het bloed in onze vingertop zich iets dichter bij huid en is de roodwaarde in onze video maximaal. Onze hartslag is dus de frequentie waarmee ons hart klopt en is meteen ook de frequentie waarmee de roodwaarde in de opgenomen video fluctueert. Als we dus het frequentiespectrum van ons roodwaarde-signaal kunnen opstellen, kunnen we uit de pieken hierin de hartslag afleiden.

Fourieranalyse in App Inventor 2

Gebruik Fourieranalyse om de hartslag in een roodwaarde signaal te bepalen.

- Maak een nieuw project `IL_hartslagmonitor_fourieranalyse`.

- Voorzie een aantal knoppen: *readCSV*, *calculateFFT* en *plotFFT*. Voorzie ook een textbox waarin de gebruiker de frame rate (*fs*) kan aanpassen en een knop *fsChanged* waarmee de gebruiker kan aangeven dat hij de frame rate aangepast heeft. Als de knop *fsChanged* aangeklikt wordt, moet een globale variabele aangepast worden aan de inhoud van de textbox.
- Gebruik nu een lijst met berekende roodwaarden, die we in de vorige module hebben opgeslaan in een CSV-bestand. Gebruik zoveel mogelijk code uit vorige opdrachten. Code kopiëren tussen verschillende projecten doe je door de blokken code binnen het ene project vanuit het witte venster naar de rugzak rechtsboven te slepen en die vervolgens binnen een ander project vanuit de rugzak naar het witte venster te slepen.

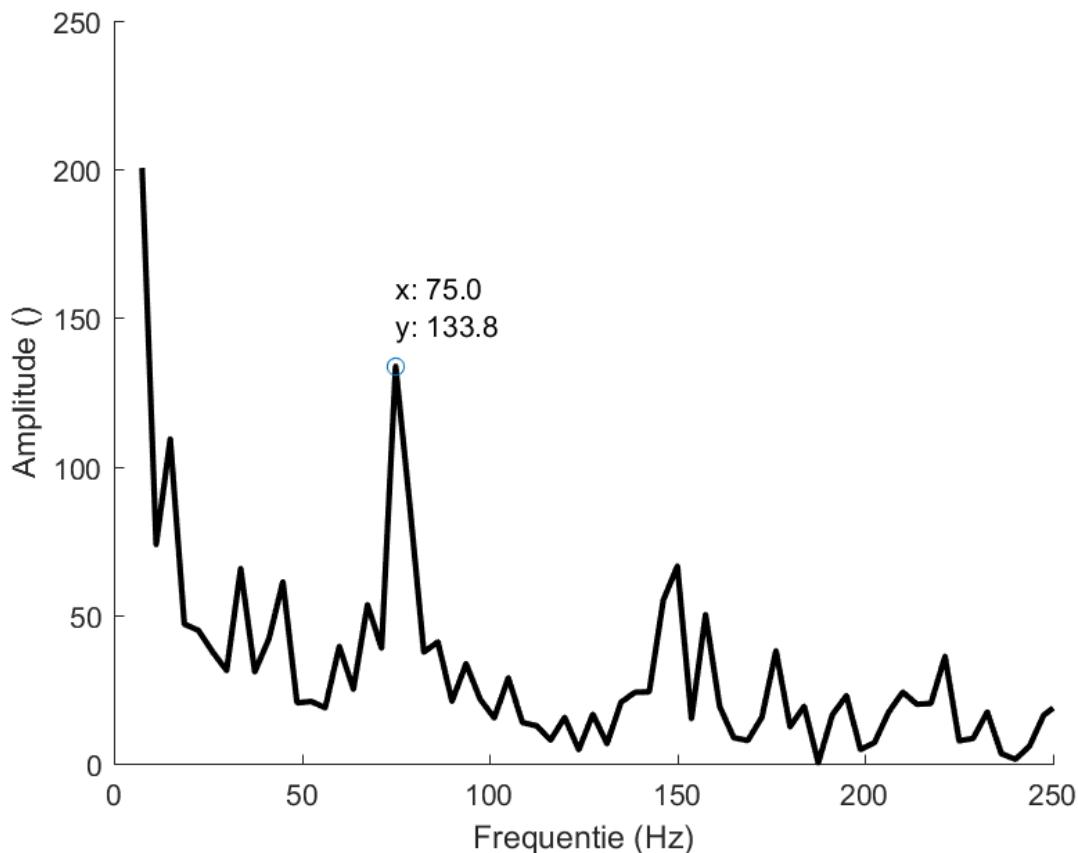
Voeg een *File*-component toe je aan je app om het CSV-bestand in te lezen. Wanneer een knop *Read CSV* ingedrukt wordt, moet het CSV-bestand dat je opgeslaan hebt in de vorige module ingelezen worden. Kijk indien nodig even terug in je vorige project waar en onder welke naam je het CSV-bestand hebt opgeslaan. Als het CSV-bestand is ingelezen, moet de inhoud omgezet worden in een lijst en in een lijst geplaatst worden. Geef die lijst weer in een label en vergelijk de tekst met het CSV-bestand op een computer om te controleren dat dit correct gebeurde.

Opmerking. *Afhankelijk van het CSV-bestand dat je inleest, kan het zijn dat de lijst met roodwaarden in plaats van getallen als elementen lijsten met 1 element als elementen bevat. Om aan de roodwaarde te raken, moet je dan het eerste element halen uit elke lijst (die ook weer een element van de oorspronkelijke lijst is).*

- Kopieer de code uit een van de vorige opdrachten om opeenvolgende roodwaarden uit te middelen. Maak een nieuwe lijst, bv *smoothRedValuesList*, waarin elk element een gemiddelde is van vier opeenvolgende elementen uit de originele lijst.
- De implementatie van de Fourieranalyse hebben we al voor jullie gedaan. Voeg daarom de extensie *FFTCalculator* toe aan je app.
- Gebruik het blok *FFTCalculator - CalculateFft* om een Fourieranalyse uit te voeren wanneer de knop *calculateCSV* aangeklikt wordt. Het resultaat is een lijst *redValuesFFT*.
- Verwijder het eerste element uit de lijst. Dit is nodig om een correct frequentiespectrum te kunnen maken.
- Wanneer op de knop *plotFFT* gedrukt wordt, moet het frequentiespectrum getoond worden. Gebruik hiervoor het blok *Chartmaker - DrawLineGraph*. Voeg opnieuw een *Webviewer* toe aan je app. De titel van de grafiek en van de assen en het label kies je zelf. Maak de grafiek eerst met *scalingFactor* gelijk aan 1.
- Op de *x*-as staan de mogelijke hartslagfrequenties. De sprong tussen twee opeenvolgende hartslagfrequenties is $\frac{60 \cdot fs}{\text{aantal roodwaarden}}$. Pas de *scalingFactor* hieraan aan.
- Welke hartslag meten we nu op? De grootste piek in de grafiek toont welke frequentie “het meeste in het signaal aanwezig is”. De *x*-waarde van de grootste piek toont dus de meest waarschijnlijke hartslag. Je kan de *x*- en *y*-waarde van een punt op de grafiek zien door erop te klikken. Komt de hoogste piek overeen met de hartslag die je zelf berekende? En met de hartslag die je app berekende via automatische piekdetectie?

Opmerking. Het is mogelijk dat de grootste piek zichtbaar is voor een heel kleine x -waarde. In dat geval is de piek wellicht veroorzaakt door ruis in het signaal. Komt de 2e grootste piek wel in de buurt bij een meer waarschijnlijke hartslag?

Voor de roodwaarden in het voorbeeld CSV-bestand zou je een grafiek moeten zien zoals hieronder:



In dit geval is de meest waarschijnlijke hartslag ongeveer 75 bpm.

- Wat is de stapgrootte waarmee de frequenties op de x -as verspringen? Wat zegt dit over de nauwkeurigheid van de hartslagmeting? Heeft de stapgrootte verband met een variabele die we eerder al gebruikten? Welke? Hoe kan je de nauwkeurigheid van de hartslagmeting verbeteren?

Module 5

Bouwen van de complete app

Op dit moment heb je al alle stappen overlopen die nodig zijn om je hartslag te meten m.b.v. je smartphone: (1) een video opnemen, (2) de video splitsen in afbeelden, (3) de gemiddelde roodwaarde voor elke afbeelding berekenen, (4) een graiek van de gemiddelde roodwaarde i.f.v. de tijd maken, (5) automatische piekdetectie en berekening van de hartslag, en eventueel (6) frequentie-analyse m.b.v. de Fouriertransformatie. Elk bouwblok uit het schema uit Module 1 hebben we apart geïmplementeerd, waarbij we abstractie maakten van wat in eerdere en latere bouwblokken moest gebeuren. Bij de implementatie van de functionaliteit van een bouwblok houden we enkel rekening met welke inputs en outputs dat bouwblok heeft; verder negeren we de complexiteit en de functionaliteit van de andere bouwblokken. Die segmentering (het opdelen van een probleem in segmenten of deelproblemen) laat ons toe onze volledige aandacht te richten op 1 deelprobleem van het meer complexe hartslagmonitor-probleem. Natuurlijk moet alles op een bepaald moment samenkommen, en dat moment is nu gekomen.

1 Video opnemen

Importeer een bestaand project, nl. *IL_startapp_allessamen* in MIT App Inventor 2. In dit project zijn alvast de nodige knoppen en extensies voorzien die nodig zijn voor de volledige hartslagmonitor. De functionaliteit achter de knoppen moet je zelf nog implementeren.

We starten met de eerste knop “1. Video opnemen”. Wanneer op de knop geklikt wordt moet de camcorder geopend worden zodat de gebruiker een video kan opnemen. De gebruiker moet zelf de flits aanzetten en beslissen hoelang hij video zal opnemen.

Nadat de video opgenomen is, keurt de gebruiker de video goed of af door op het vinkje of op het kruisje te klikken. Als de gebruiker de video goedkeurt, wordt de video opgeslaan op de smartphone.

2 Video naar afbeeldingen m.b.v. app VideoNaarAfbeeldingen

Nadat een video is opgenomen, moet de video gesplitst worden in afbeeldingen. Opnieuw gebruiken we de app “Video naar afbeeldingen” die je in Module 3 Sectie 2 installeerde. Het

verschil met vroeger is dat we de app “Video naar afbeeldingen” niet handmatig openen, maar dat de app “Video naar afbeeldingen” automatisch geopend wordt vanuit onze hartslagmonitor app wanneer we de knop “2. Video naar afb” aanklikken.

Hiervoor gebruiken we een component “ActivityStarter”. In het project *IL_startapp_allessamen* staat de ActivityStarter normaal gezien al correct ingesteld. Wanneer de knop “2. Video naar afb” aangeklikt wordt, moet de activiteit van de ActivityStarter gestart worden.

Hierdoor opent de app “Video naar afbeeldingen”, zodat de gebruiker de video die hij in afbeeldingen wil splitsen kan selecteren. Als hij klikt op “Extract images” wordt de video gesplitst in afbeeldingen. Na enige tijd worden de afbeeldingen getoond, net als de map waar de afbeeldingen opgeslaan worden. Schrijf ergens neer welke map dit is.

3 Gemiddelde roodwaarde voor elke afbeelding berekenen

Nadat de video gesplitst is in afbeeldingen, bereken je de roodwaarde van elke afbeelding. Gebruik hiervoor de code die je schreef in Module 3 Sectie 3 - opdracht 1. Code kopiëren tussen verschillende projecten doe je door de blokken code binnen het ene project vanuit het witte venster naar de rugzak rechtsboven te slepen en die vervolgens binnen een ander project vanuit de rugzak naar het witte venster te slepen. De roodwaarden worden opgeslagen in een lijst *redValueList*.

4 Grafiek van de gemiddelde roodwaarde i.f.v. de tijd

Nadat de roodwaarden van de afbeeldingen berekend werden, kan je de grafiek van de roodwaarden in functie van de tijd weergeven. Gebruik hiervoor de code die je schreef in Module 3 Sectie 3 - opdracht 2. Op basis van de grafiek kan je de hartslag berekenen.

5 Automatische piekdetectie en berekening van de hartslag

Als de knop “5. Piekdetectie” aangeklikt wordt, moet de hartslag automatisch berekend worden d.m.v. automatische piekdetectie. Gebruik hiervoor de code die je schreef in Module 4 Sectie 1 - opdracht 1. Denk eraan opeenvolgende roodwaarden uit te middelen als je signaal heel ruizig is, zoals je ook deed in Module 4 Sectie 1 - opdracht 1. De ruizigheid van het signaal kan je op basis van de grafiek inschatten.

6 Extra: frequentie-analyse m.b.v. Fouriertransformatie

Als de knop “6. DFT berekenen” aangeklikt wordt, moet een frequentieanalyse uitgevoerd worden en het frequentiespectrum getoond worden. Gebruik hiervoor de code die je schreef in Module 4 Sectie 3 - opdracht 1. Op basis van het frequentiespectrum kan je opnieuw de meest waarschijnlijke hartslag bepalen.

Module 6

Experimenteren met de app

Nu we een app hebben gemaakt waarmee we onze hartslag kunnen meten is het tijd om de app te evalueren. Net zoals bij elk ontwikkeld product zijn er wellicht verbeteringen mogelijk. In deze module gaan we met de app aan de slag en vergelijken we de app met alternatieven om je hartslag te meten, met als bedoeling eventuele fouten te achterhalen en verbeterpunten te ontdekken. Kritisch stilstaan bij je ontwerp en bij je product is noodzakelijk als je je product wilt blijven verbeteren.

1 Invloed van sport en rust

Het is natuurlijk interessant om te weten wat je hartslag bij rust is, zoals besproken in Module 1, maar de meeste mensen gebruiken hartslagmeters om na te gaan wat je hartslag is bij inspanning.

Hartslag bij rust en na inspanning meten

Meet je hartslag vlak voor en vlak na een inspanning. Zit je hartslag na inspanning binnen het aanbeloven hartslagbereik voor jouw leeftijd?

Verschilt je hartslag bij rust als je rechtop staat, rechtop zit of ligt? Waaraan zou dit kunnen liggen?

2 Vergelijking met bloeddrukmeter/hartslagmeter/smartwatch/app

Zoals we in Module 1 besproken hebben, bestaan er alternatieven om je hartslag te meten. We spraken over bloeddrukmeters en hartslagmeters. Maar ook apps zoals de onze zijn al te verkrijgen in de Google Play Store.

Vergelijking met alternatieve hartslagmonitors

Verzamel een aantal alternatieven voor onze hartslagmonitor-app.

Som eerst op welke factoren je belangrijk vindt bij het gebruik van een hartslagmonitor.

Ga na hoe goed de verschillende alternatieven naar jouw mening scoren voor elk van die factoren. Dit is een *kwalitatieve evaluatie*. Jij beoordeelt de kwaliteit van elk alternatief, op basis van jouw aanvoelen.

Misschien kan je sommige van die scores ook objectiever maken. Objectieve maatstaven zijn het tegengestelde van subjectieve maatstaven. Objectiever wil zeggen dat het oordeel minder afhankelijk is van jouw aanvoelen of van jouw persoonlijke mening, maar meer neutraal, zonder vooroordeel, kan vastgesteld worden. Vaak worden hiervoor *kwantitatieve maten* gebruikt: dit zijn factoren die meetbaar zijn.

Wellicht is de snelheid van de hartslagmeting voor jou belangrijk. Je subjectieve aanvoelen is wellicht dat de app die wij ontworpen hebben minder snel is dan een commercieel beschikbare hartslagmeter. (Dit is natuurlijk omdat het ook doenbaar moet zijn om de hartslagmonitor binnen een beperkt aantal lessen te maken, op basis van de kennis die jullie nu al hebben.) Maar als objectieve maat volstaat dit aanvoelen niet. Je zou met een chronometer kunnen nagaan hoelang de hartslagmeting duurt voor elk van de alternatieven. Dit is een objectieve maatstaf voor de snelheid van de hartslagmeting.

Kan je ook voor de andere factoren die jij belangrijk vindt bepalen hoe je elke factor een objectieve score zou kunnen geven?

3 Evaluatie: wat kan beter?

Plus- en minputen en suggesties voor verbetering bepalen

Uit de vergelijking met de alternatieven heb je zelf kunnen ervaren hoe goed of slecht onze hartslagmonitor werkt.

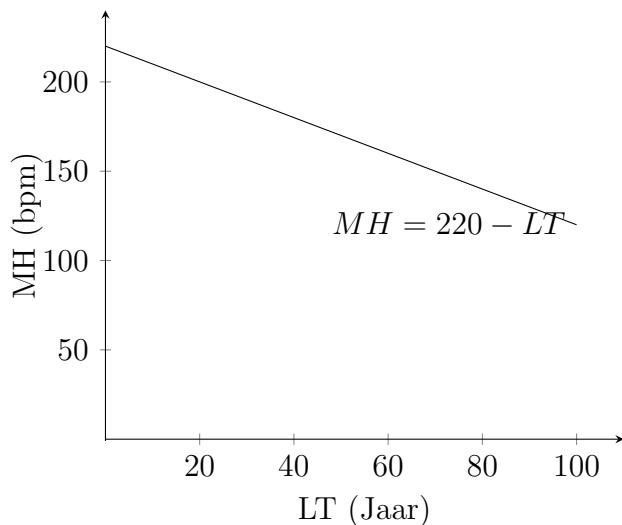
- * 1 Op welke vlakken scoort de hartslagmonitor goed? Wat kan beter?
- * 2 Als ontwerper moet je niet alleen bepalen wat de minpunten van je ontwerp zijn; je moet ook nadrukken over hoe je het ontwerp kan verbeteren. Welke aanpassingen zou jij doen?
- * 3 In een ideale wereld kan je uiteraard veel aanpassingen doen, maar als echte ontwerper ben je meestal meer beperkt. Welke beperkingen zie jij?
- * 4 De beperkingen bepalen ook de haalbaarheid van de verbetering. Welke aanpassingen zijn haalbaar, welke zijn moeilijker te implementeren?

Opmerking. Merk op dat voor dit probleem (*net als voor veel andere problemen!*) geen modeloplossing beschikbaar is. Vaak is een probleem open-ended; dit wil zeggen dat niet op voorhand vastligt wat de beste oplossing voor het probleem is. Soms kan je door ervaring inschatten welke ontwerpen de beste zullen zijn, maar soms moet je het ook gewoon uitproberen en via evaluatie achterhalen wat goed en wat minder goed loopt en welke aanpassingen de kwaliteit van je producten goede beïnvloeden. Deze probleemoplossende vaardigheden zijn cruciaal in veel beroepen en vergen een zekere kritische houding en creativiteit.

Oplossingen van de oefeningen zonder *

Oplossingen module 1

- 1
- $MH_{18} = 202$ en hartslagbereik tussen 111,1 en 171,7
 - $MH_{30} = 190$ en hartslagbereik tussen 104,5 en 161,5
 - $MH_{65} = 155$ en hartslagbereik tussen 85,25 en 131,75



2

- 3
- 10 s: factor 6
 - 20 s: factor 3
 - Hoe korter de telperiode, hoe nauwkeuriger je moet tellen. Eventuele telfouten worden immers vermenigvuldigd met de factor. De factor is groter, naarmate de telperiode korter is. Telfouten worden dus groter als de telperiode korter is.

Oplossingen module 2

Oplossingen module 3

- 1 Je kan $255 \cdot 255 \cdot 255 = 16581375$ verschillende kleuren genereren.
- 2 Voor een frame rate van 24 *fps* wordt elke afbeelding gedurende 0,042 s (of 42 ms) getoond. Voor een frame rate van 30 *fps* wordt elke afbeelding gedurende 0,033 s (of 33 ms) getoond.

Oplossingen module 4

- 1 Wellicht komen hier heel diverse antwoorden, afhankelijk van de achtergrond van de leerlingen. Leerlingen uit de 3e graad zullen wellicht over maxima spreken, terwijl leerlingen uit de 2e graad misschien meer woordelijke/minder wiskundige omschrijvingen zullen geven.
- 2 Er is een maximum voor element u_i als en slechts als: $u_i > u_{i-1}$ en $u_i < u_{i+1}$.
- 3 Als de grootheid x een continu bereik van waarden kan aannemen, spreken we over een continu signaal.
- 4
 - Bij afgeleiden gebruiken we continue signalen, en hier discrete.
 - Bij afgeleiden is er nog een deling door Δh . Dit laten we hier achterwege.
Opmerking. *Merk op dat je ook de uitdrukking $u_i - u_{i-1}$ zou kunnen delen door de tijdsduur tussen 2 foto's. Die tijdsduur is $\Delta i = 1/fs$, het omgekeerde van de frame rate.*
Die deling heeft geen invloed op het resultaat (de gedetecteerde pieken), omdat we zoeken voor welke waarde van i de uitdrukking $u_i - u_{i-1}$ of $\frac{u_i - u_{i-1}}{\Delta i} = [u_i - u_{i-1}]fs$ overgaat van een positieve naar een negatieve waarde. Die extra (positieve) factor fs beïnvloedt het teken, en dus ook de uitkomst, niet.

Oplossingen module 5

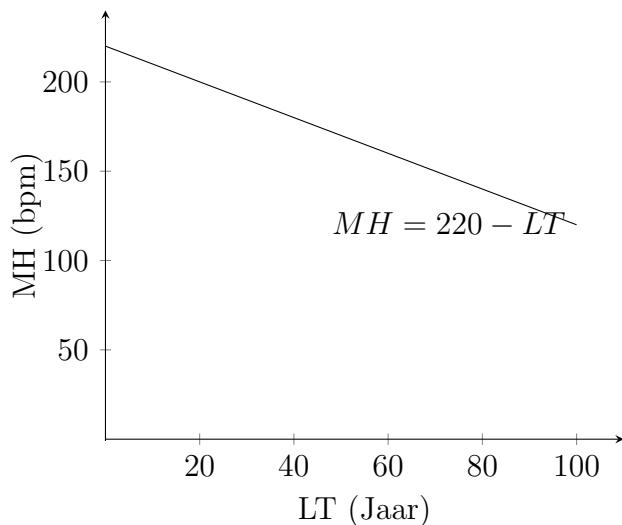
Oplossingen module 6

- 3
 - Budget
 - Tijd
 - Wiskundige achtergrond
 - Beperkingen hardware: snelle vs trage smartphone, geheugen, wel of geen flash, ...

Oplossingen van alle oefeningen

Oplossingen module 1

- 1
- $MH_{18} = 202$ en hartslagbereik tussen 111,1 en 171,7
 - $MH_{30} = 190$ en hartslagbereik tussen 104,5 en 161,5
 - $MH_{65} = 155$ en hartslagbereik tussen 85,25 en 131,75



2

- 3
- 10 s: factor 6
 - 20 s: factor 3
 - Hoe korter de telperiode, hoe nauwkeuriger je moet tellen. Eventuele telfouten worden immers vermenigvuldigd met de factor. De factor is groter, naarmate de telperiode korter is. Telfouten worden dus groter als de telperiode korter is.

Oplossingen module 2

Oplossingen module 3

- 1 Je kan $255 \cdot 255 \cdot 255 = 16581375$ verschillende kleuren genereren.
- 2 Voor een frame rate van $24 \text{ } fps$ wordt elke afbeelding gedurende $0.042 \text{ } s$ (of $42 \text{ } ms$) getoond. Voor een frame rate van $30 \text{ } fps$ wordt elke afbeelding gedurende $0.033 \text{ } s$ (of $33 \text{ } ms$) getoond.

Oplossingen module 4

- 1 Wellicht komen hier heel diverse antwoorden, afhankelijk van de achtergrond van de leerlingen. Leerlingen uit de 3e graad zullen wellicht over maxima spreken, terwijl leerlingen uit de 2e graad misschien meer woordelijke/minder wiskundige omschrijvingen zullen geven.
- 2 Er is een maximum voor element u_i als en slechts als: $u_i > u_{i-1}$ en $u_i < u_{i+1}$.
- 3 Als de grootheid x een continu bereik van waarden kan aannemen, spreken we over een continu signaal.
- 4
 - Bij afgeleiden gebruiken we continue signalen, en hier discrete.
 - Bij afgeleiden is er nog een deling door Δh . Dit laten we hier achterwege.
Opmerking. *Merk op dat je ook de uitdrukking $u_i - u_{i-1}$ zou kunnen delen door de tijdsduur tussen 2 foto's. Die tijdsduur is $\Delta i = 1/fs$, het omgekeerde van de frame rate.*
Die deling heeft geen invloed op het resultaat (de gedetecteerde pieken), omdat we zoeken voor welke waarde van i de uitdrukking $u_i - u_{i-1}$ of $\frac{u_i - u_{i-1}}{\Delta i} = [u_i - u_{i-1}]fs$ overgaat van een positieve naar een negatieve waarde. Die extra (positieve) factor fs beïnvloedt het teken, en dus ook de uitkomst, niet.

Oplossingen module 5

Oplossingen module 6

- 3
 - Budget
 - Tijd
 - Wiskundige achtergrond
 - Beperkingen hardware: snelle vs trage smartphone, geheugen, wel of geen flash, ...

INNOVATION LAB

Gebroeders De Smetstraat 1
9000 GENT, België
innovationlab@kuleuven.be
<https://eng.kuleuven.be/innovationlab>

