Verschillende detectors

Bij het maken van onze werkende stappenteller moesten we daarbij natuurlijk gebruik maken van verschillende soorten detectors om de stappen zo optimaal mogelijk te kunnen detecteren en tellen. Voor dat we begonnen aan onze detectors te implementeren moesten we eerst onze data filteren door de valversnelling eruit te halen en dat hadden we gedaan met de volgende formule die we reeds gebruikt hadden in labsessie 2: r= sqrt(x²+y²+z²)-9.81.FORMULE In onze vierde labsessie hebben we daarna onze eerste detector in java geïmplementeerd de zogenaamde ‘ Dummydetector ‘ en wat deze detector doet is heel simpel. Deze detector detecteert namelijk niks anders dan het aantal metingen die er gedaan worden, dus hier worden er nog geen stappen geteld. Daarna hadden we een functie addAccelerationData() aangemaakt om de stappen te tellen en daarin hadden we dus ook onze formule eringestoken om de data te filteren. Na het implementeren van onze eerste detector in java konden we beginnen aan onze tweede detector die we nodig hadden om de stappen te kunnen detecteren. Deze tweede detector heet de ‘SimpleTresholdDetector’. Met deze detector konden we een drempelwaarde aanmaken voor onze metingen en daarbij ook voorwaarden voor opstellen. In labsessie 5 en 6 hadden we geëxperimenteerd met simpele en complexe detectors en hebben we ze daarna in labsessie 8 met elkaar vergeleken. In deze labsessies hadden we ook nog een ‘DynamicTresholdDetector’ geïmplementeerd. Bij deze detector kan de drempelwaarde bepaalt worden als een fractie van het maximum van de laatste gemeten datapunten. We hadden daarbij deze 2 detectors uitgetest en zagen dat de DynamicTresholdDetector nauwkeuriger stappen kon tellen dan de andere detector. Deze detector verschilt dus niet heel veel van de ‘SimpleTresholdDetector’ , maar is een stuk preciezer en efficiënter.

Pseudocode

Nadat we onze detectors in java hebben geïmplementeerd, moesten we nog voorwaarden opstellen voor onze drempelwaarde in onze pseudocode die onze stappen zal tellen. Voor deze drempelwaarde hadden we een parameter ‘a’ gekozen. Bij het kiezen van onze drempelwaarde hadden we eerst gekeken naar de grafiek van onze metingen die we in de vorige labsessies hadden gedaan. Als eerst hadden we een waarde van 10 toegekend aan onze parameter, maar we zagen meteen dat deze waarde een te hoge drempelwaarde was en dat het vrijwel geen enkele stap kon detecteren. Daarna kozen we een waarde van 1,5 en deze waarde was al meteen veel beter voor onze parameter, omdat het vrijwel de meeste stappen kon detecteren en tellen. Dus hierbij konden we nu onze voorwaarden voor opstellen. Dus we willen dat onze gefilterde data deze drempel overschrijdt en dat het eenmalig geregistreerd wordt. Daarbij hadden we een gedefinieerde waarde *vorige* aangemaakt. Onze eerste voorwaarde die we hadden aangelegd zegt dat als onze grafiek deze horizontale asymptoot of drempel met waarde 1,5 overschrijdt in de stijgende richting, dat het een stap bijtelt bij onze waarde *vorige* en staat deze waarde nu op 1.GRAFIEK Als het onder deze drempelwaarde blijft dan telt het natuurlijk geen stappen bij. Voor onze tweede voorwaarde hebben we het omgekeerde gedaan en hebben we gezegd dat wanneer onze grafiek deze drempelwaarde via boven naar beneden overschrijdt dat het een stap aftrekt van onze waarde *vorige* en op die manier staat deze waarde nu weer op 0. Op deze werkwijze blijft ons algoritme werken en stappen bijtellen waar nodig. CODE