Portfolio MAP

Fonteyne Joost

Contents

[Low-tech proposal (not used) 1](#_Toc127018924)

[Advantages 5](#_Toc127018925)

[Disadvantages 5](#_Toc127018926)

[Questions 5](#_Toc127018927)

[Hardware verzamelen 5](#_Toc127018928)

[Software selecteren 7](#_Toc127018929)

[Initial power profiling 7](#_Toc127018930)

[Blink power consumption 7](#_Toc127018931)

[Deep sleep 8](#_Toc127018932)

[Arduino ide vs esp-idf 9](#_Toc127018933)

[First model on esp32-cam 9](#_Toc127018934)

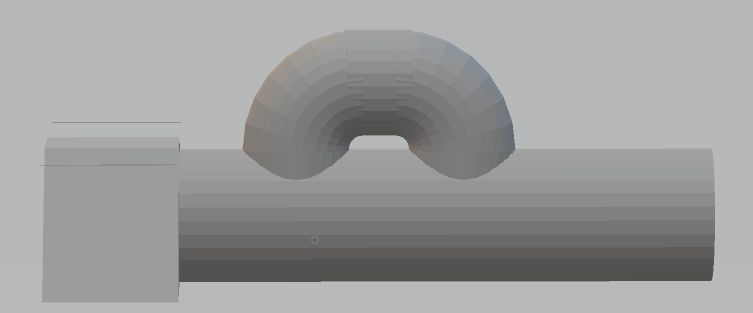
Invasion of the Chinese mitten crab

Development of a monitoring sensor: sensor alternative

# Low-tech proposal (not used)

Het doel is het ontwikkelen van een monitoring sensor die de populaties van de invasieve wolhandkrab in kaart kan brengen. Voor mij is hier al een jaar onderzoek naar gedaan. Ze vonden toen dat een camera met een raspbery pi, een object detectie algoritme en een tracking algoritme een goede oplossing was. Ze hadden echter geen goede concluderende resultaten omdat de dataset waarmee ze hun model getrainde hadden niet uitgebreid genoeg was. Na het lezen van de paper dacht ik dat het misschien mogelijk was om een simpelere meer low-tech manier uit te werken. Ik heb dan een klein beetje tijd besteed aan het uitwerken van een 3d model voor een val die de krabben zou kunnen tellen zonder al te veel ingewikkelde elektronica. Na het pitchen van dit idee aan mijn begeleiders bleken er toch wat gaten in dit idee te zitten. De grootste zijnde dat ik uiteraard een master thesis doe in de elektronica-ict richting en dan is het expres vermijden van elektronica en ICT misschien niet het beste idee. Er is dan ook niets meer gekomen van dit idee, maar ik voeg het hier toch toe aan mijn portfolio omdat ik er wel wat tijd in had gestoken en het deel was van het proces als het waren.

I think it is possible to make a decent crab counting sensor without the need for more advanced technologies like machine learning etc. I hope to accomplish this by making use of some of the natural characteristics of the crabs. The main ones being: they can survive outside of water, they can climb vertically on a mesh and they are the only species found in Flemish waters with these two characteristics. This sensor would consist of a tube that is laid flat on the river bed aligned with the flow of the water to not disrupt the natural flow. Above the tube is a smaller tube filled with air. The crabs can reach this air pocket by climbing a mesh that prevents them from going straight in the ground tube. This mesh stops the crabs but lets water, mud, small stones and small organisms pass freely. Once in the air pocket the crabs would move forward and fall into the water tube again. While they are falling they can be counted with a sensor that works based on the disruption of a light beam. They can not go back since the exist does not have a mesh, preventing a crab from being counted twice. The whole setup is held in place by a heavy piece of stone.



: side view

The front of the picture above is the weight to keep the contraption at its desired position. The middle is the ground tube, which rests on the river surface and is filled with water. The top is the air/counting chamber. It is important that the contraption never rolls over otherwise the top chamber can fill with water. That is why the weight at the front has a trapezoid shape.

A picture containing shape

Description automatically generated

: front view

The mesh is located right after the first hole leading to the top chamber. Giving access to the entrance of the top chamber but not to the exit. Icon

Description automatically generated with medium confidence

3: top chamber entrance

A picture containing building, tower

Description automatically generated

4: top chamber exit

It might be necessary to make the top chamber longer to prevent the crabs from climbing up at the exit again. But in general because of the crabs migrating behaviour I do not think many will try to go against the alignment of the trap (trap has to be aligned correctly ofc.) The trap in the pictures is configured to count crabs going with the flow of water. The trap can also be used for crabs going in the other direction by changing the position of the weight from the front to the back.

The crabs are counted when they interrupt a light source while falling from the end of the air chambre back into the water tube. They can not go back up because there is no mesh to climb at the exit.

Diagram

Description automatically generated

5: route illustration

Blue: waterflow, black: waterline, white: mesh, red: counting sensor, orange: crab route, green: crab trying to go back.

The light beam sensor sends signals trough a waterproof cable to the shore. The data arriving at the shore is processed by a microcontroller (add 1 for each pulse to count). From here on the data can be stored on site and await collection or can be transmitted trough a low power wide area network.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| Low-tech | Tube can clog |
| Easy to place/ maintain | Air chamber can fill with water destroying equipment |
| Natural waterflow not disrupted | Mesh would catch bigger rocks or would not catch smaller crabs |
| Low maintenance | Crab might prefer other route |
| Low cost | A lot of crabs might clog the system |
| Crab can not keep standing in front of sensor | A lot of crabs might give some inaccuracies when counting |
| Crab only counted once if we can assume that they don’t turn back and go around the contraption back to the entrance | Crab can be counted twice if it for some reason decides to go around the contraption and enters from the entrance again. |

## Questions

1. Can we assume the crabs to be evenly distributed over the ground surface of the water body?
   1. If yes -> can use math to calculate approximation of crabs passing point in river.
   2. If no -> need some sort of funnelling system to only allow crabs to pass trough certain points without disrupting natural water flow.
2. Do crabs prefer to go trough tubes? Or do they hate it?
3. Are crabs smart enough (or maybe instinct or something) to go to the upper air chamber instead of just all clogging the entrance or even preferring to move back?
   1. If no -> Can angle the mesh a bit to smooth transition
4. What is the smallest space a small crab can fit through?
   1. To decide optimal mesh size
5. How long can we assume this system to remain operational until segment build-up, rocks and plants clog it?
6. …

Dus tot zover dat idee. Hier na was ik met mijn begeleider tot de conclusie gekomen dat ik best gewoon verder werkte op het werk van de 2 mensen voor mij. Met de nadruk op zo min mogelijk energie te verbruiken. Dit door een zo licht mogelijk viable model op een zo licht mogelijk platform te gebruiken en aan een soort van energie harvesting te gaan doen met bv een zonnepaneel.

# Hardware verzamelen

Dit is een praktische doelstelling en ik heb hier dan ook heel wat hardware voor nodig. Dit kan soms lastig zijn omdat dingen bestellen en wachten tot ze aankomen soms lang kan duren. Het is ook moeilijk om op voorhand exact te bepalen wat je allemaal nodig zal hebben.

Het eerste dat ik moest bepalen is op welk platform ik wil gaan werken. De mensen voor mij gebruikten een raspberry pi, ik wil echter zo min mogelijk energie gebruiken en daar is de raspberry pi nu gewoon niet efficiënt genoeg voor. Na het overwegen van verschillende mogelijkheden ben ik uiteindelijk beland op een kitje genaamd ESP32-cam. Wat, zoals de naam al doet vermoeden, bestaat uit een ESP32 bordje verbonden met een OV2640 camera module. Dit bordje is klein, goedkoop en zuinig maar belooft toch taken zoals object herkenning etc uit te kunnen voeren. Er zijn ook enkele tutorials te vinden online waarin dit setje gebruikt wordt om soortgelijke dingen als mijn doel te doen, ideaal dus. De vraag blijft natuurlijk of het echt zo goed is, maar dat kan ik alleen praktisch ondervinden. De ESP32 is een dual core processor die claimt beduidend minder stroom te verbruiken dan vergelijkbare processors en heeft bovendien meerdere low power modes. Hij beschikt ook over pinnen waarop de ESP32 wakker kan worden gemaakt zodat je deze kan laten slapen tot wanneer hij echt nodig is. De ESP32 heeft ook wifi, bluetooth en low power bluetooth. Dit kan eventueel interessant zijn om verwerkte data door te sturen of foto’s naar de cloud te sturen om ze daar te verwerken om zo accuratere resultaten te bekomen. De ESP32 beschikt over 520 kb ram en 4 mb flash memory.

Een nadeel van de ESP32-cam kit is wel dat de bijgeleverde camera geen night vision heeft. Wat wel belangrijk is aangezien de krabben niet van fel licht houden en het onderwater, zeker troebel water best donker kan worden. Gelukkig kan de OV2640 gemod worden om ook IR licht op te vangen. De krabben zelf zijn koudbloedig, daarom bestel ik ook een stuk of 5 IR ledjes. Deze gaan natuurlijk ook weer extra stroom verbruiken dus we willen er zo min mogelijk gebruiken, maar deze zijn niet echt duur dus ik heb er beter een paar te veel voor als ik merk dat 1 ledje niet genoeg is om de krabben goed te zien.

De ESP32 kan ook niet zomaar via USB geprogrammeerd worden. Daarom bestel ik ook een programmer. Deze bied langs de ene kant een usb poort aan, waar je dan mee kan verbinden. Langs de andere kant bied deze pinnen aan waar je de ESP32 dan aan kan hangen om deze zo te programeren. De specificaties van deze programmer zijn niet belangrijk en ook niet echt relevant voor dit project. Hij moet natuurlijk wel de juiste pinnen aanbieden om de ESP32 te programeren

Voor het energie harvesting deel heb ik besloten om het simpel te houden en gewoon een zonnepaneel te gebruiken met een regulator bordje dat ook een lithium batterij kan opladen. Gelukkig voor mij had ik thuis nog een 9v zonnepaneel liggen. Bovendien had ik ook nog een bordje dat de energie van een zonepaneel tussen de 6 en 24v kan opslaan in een batterij en tegelijk een usb output met 5v en max 1a heeft. De ESP32-cam wil ook 5v dus dit komt goed uit.

Om het energie verbruik zo laag mogelijk te krijgen leek het mij ook een goed idee om een beweging sensor aan te schaffen. Zo hoeft de setup enkel actief te draaien als er daadwerkelijk iets voorbijkomt. De klassieke PIR sensors zijn hier echter geen optie, omdat de krabben koud bloedig zijn. Daarom heb ik voor een microwave beweging sensor gekozen. Deze werkt als een soort radar door constant microwave golven uit te sturen en dan terug op te vangen. De sensor die ik gekozen heb is de SEN0192. Deze kan makkelijk aan een van de pinnen van de ESP32 gekoppeld worden. Het idee is dan dat de ESP32-cam in deep sleep gaat en de IR lichtjes uit zijn tot de beweging sensor iets op pikt. Hoe goed dit werkt in onze setup (zijnde onderwater en soms snel bewegend water) valt nog af te wachten. Een probleem dat ik nu al mogelijks voorzie is dat het bewegende water de sensor constant gaat triggeren waardoor we uiteindelijk meer stroom verbruiken dan als we de setup gewoon zonder de sensor gebruiken en constant aan laten. Maar ik denk dat dit zeker de moeite waard is om eens te testen als al de rest klaar is. Omdat er potentieel is om deze manier toch maar weer wat extra stroom te besparen.

# Software selecteren

De 2 studenten voor mij gebruikten voor hun krab herkenning yolov3-tiny. Om te bepalen wat ik ga gebruiken ben ik dus eerst eens gaan kijken of er ondertussen al nieuwere versies van zijn. Dit was het geval met de nieuwste al op versie 7. Van yoloV7 is ook een tiny variant dus dit is zeker een kandidaat. De nieuwere yolo modellen vereisen een uitgebreidere dataset om optimaal te werken, wat nu net het probleem was bij de vorige thesis. Daarom besloot ik om toch nog wat te blijven rondkijken. Ik vond ook een paper over onderwater detectie genaamd: “ULO: An Underwater Light-Weight Object Detector for Edge Computing” wat dus perfect is voor mijn use case. De mensen van deze paper hadden hun eigen algoritme ontwikkeld specifiek voor onder water en stelde dit ook ter beschikking. Helaas bleek dat zij als platform de Nvidea jetson nano gebruikte. Een platform dat ik eerder al had afgekeurd omdat het ten eerste veel geld kost en al meer verbruikt dan een simpele arduino. Dit maakte mij wat sceptisch over het reproduceren van de paper zijn beloftes op de ESP32, vooral voor mijn toepassing waar low power echt wel super belangrijk is. Uiteindelijk liet ik mijn oog vallen op FOMO (faster objects more objects). Dit is een algoritme dat tijd bespaart door niet telkens de bounding boxes van objecten te berekenen maar enkel hun middelpunt. FOMO is gespecialiseerd in het vinden van meerdere objecten van ongeveer dezelfde grootte, perfect dus voor de krabben. Een probleem is wel dat FOMO werkt met een instelbaar grid, een heel grote krab kan dus meerdere vakjes innemen in het grid waardoor kleinere krabben in dezelfde vakjes niet gedetecteerd worden. FOMO claimt heel schaalbaar te zijn van zware GPU’s tot tinyML (wat ik doe dus). Dit heeft als extra vordeel dat ik een zwaardere FOMO versie dus eventueel ook kan runnen in de cloud en de ESP32 fotos laten opsturen via zijn wifi of bluetooth. Dit is een handig alternatief voor moest de ESP32 toch niet toereikend zijn voor mijn toepassing. FOMO heeft bovendien een ingebouwde tracker. Al bij al dus zeker een heel goede kandidaat voor mijn toepassing.

Om mijn ESP32 te programmeren heb ik een ontwikkeling omgeving nodig. Ik was eerst van plan om dit via de arduino IDE te doen. Er zijn veel tutorials die uitleggen hoe je dit moet doen dus dat is geen probleem. Mijn begeleider rade mij echter aan om ESP IDF te gebruiken omdat je via arduino IDE geen toegang hebt tot low power en deep sleep modus etc. Wat uiteraard wel belangrijke features zijn in mijn context. Ik heb er dus uitijndelijk voor gekozen om te starten met in vs code de esp-idf extensie te gebruiken om mijn esp32 te programeren.

Om de power consumptie van alles in kaart te brengen maak ik gebruik van een nordic semiconductors power profiler kit v2. Deze kan gebruikt worden als ampere meter maar ook als source meter. Ik gebruik hem in source mode, wat inhoud dat de stroom voor het te meten device door de profiler wordt aangeleverd.

# Initial power profiling

## Blink power consumption

Om de werking van de ppkv2 onder te knie te krijgen start ik met het flashen van een standaard blink programma op de esp32. Dit programa zal om de 2 seconden het blauwe ledje aan of uit zetten. In principe zouden we dan dus om de 2 seconden een klein verschil in power consumptie moeten waarnemen met de ppkv2. Na het instaleren van de juiste software blijkt dit dan ook het geval.

Chart

Description automatically generated

We zien duidelijk dat er om de 2 seconden even wat meer of minder stroom wordt verbruikt. Dit komt overeen met het knipperen van het ledje. We zien ook dat de esp32 met een heel simpel programmatje gemiddeld 38.65mA verbruikt.

## Deep sleep

Esp32 heeft verschillende power saving modes. Deep sleep zet bijna alles af waardoor er heel weinig stroom wordt verbruikt. Maar nog wel net genoeg om de low power coprocessor te laten draaien zodat het commando om terug wakker te worden kan gegeven worden. Dit kan een extern interupt event zijn via bepaalde pinnen maar ook een ingebouwde timer die afloopt. Onderstaande grafiek is deep sleep met een timer als wake up event. De esp32 is ingesteld om 5 seconden in deep sleep te gaan en dan 1 second terug wakker te worden. Chart, waterfall chart

Description automatically generated with medium confidence

We zien dat de consumptie in deep sleep rond de 10mA ligt, wat overeen komt met wat er online geadverteerd wordt. In de actieve second zitten we gemiddeld rond de 50mA. Chart, waterfall chart

Description automatically generated

## Arduino ide vs esp-idf

De arduino ide is veel makkelijker in het gebruik dan visual code met de isp-idf extensie. Maar bij het vergelijken van de actieve power consumptie bij het eerste blink voorbeeldje (geprogrammeerd met esp-idf) en het 2de deep sleep voorbeeldje (geprogrammeerd met arduino ide) blijkt dat hier toch een beduidend verschil op zit. Iets wat ik eigenlijk niet verwacht had. Om mijn vermoeden te bevestigen doe ik het eerste blink voorbeeldje nog eens, maar nu met arduino ide.

Timeline

Description automatically generated

We zien hier nu een hoger gebruik voor wat essentieel dezelfde opdracht is.

# First model on esp32-cam

Model om de aanwezigheid van een auto in een afbeelding te herkennen. Dit is een model dat wordt aangereikt door edgeimpuls zelf als een soort van tutorial. Deze sketch maakt gebruik van een webserver etc, dus veel extra opsmuk die eigenlijk niet nodig is. Maar dit is een goed startpunt om te leren hoe deze modellen gegenereerd worden en hou je ze kan uploaden. Dit voorbeeld is bovendien geupload met de arduino IDE, die door gebrek aan hardware acceleratie support niet dezelfde benchmarks as de officiële idf ide kan halen. Dit model is dan ook een soort van baseline. Later zal ik het verbruik zo laag mogelijk proberen krijgen door alle onnodige functies uit te zetten. Als ik dan later wil weten hoeveel vooruitgang ik nu juist gemaakt heb kan ik terug naar dit model komen om een vergelijking te maken.

Power consumptie webserver:

Graphical user interface, chart

Description automatically generated

De korte piek komt voor wanneer er een client met de webserver verbinding maakt. De volgende afbeelding toont de consumptie piek wanneer de esp32-cam een foto maakt: Graphical user interface, chart

Description automatically generated

We zien een duidelijke groep pieken verschijnen in onze power profiler alvorens de afbeelding verschijnt op de webserver. Soms is deze groep veel langer en dan duurd het ook veel langer alvorens de afbeelding verschijnt. Maar aangezien dit uitzonderlijk is ga ik er vanuit dat dit ergens een fout is. De volgende afbeelding toont het maken van een foto via de webserver en vervolgens het model loslaten op deze foto: Chart

Description automatically generated

Wat we hier zien is de consumptie bij het maken van een foto plus het loslaten van het model op deze foto. Ook hier is deze groep in uitzonderlijke gevallen veel langer, zoals bij het nemen van een foto. Bij deze grafiek werd er geen object gedetecteerd om zeker te zijn dat dit profiel ook klopt als er wel een gedecteerd wordt heb ik dit ook eens getest en dit was gewoon hetzelfde:

Graphical user interface, chart

Description automatically generated

Het valt wel op dat de grafiek er bij elke nieuwe iteratie altijd net iets anders uitziet, wat vervelend is voor mij. Omdat het voor mij leuk is als het uitvoeren van dezelfde actie altijd hetzelfde resultaat geeft. Maar over het algemeen kan over alle grafieken wel gezegd worden dat er pieken tot 300 mA zijn. Het maken van een foto is altijd een duidelijke groep en het runnen van het model is ook altijd een duidelijk te onderscheiden groep. Soms zijn de groepen wat compacter dan anders. De volgende foto toont de consumptie bij het streamen van video van de camera via de webserver:

Chart

Description automatically generated

Dit is een vrij duidelijk resultaat denk ik. Het draaien van het model op de live feed van de stream is helaas geen optie van dit standaard programmatje.

# Het werk opdelen

De opdracht: het ontwikkelen van een monitoring sensor is een heel breed gegeven. Daarom wil ik de opdracht voor mezelf wat vernauwen. Ik wil hetzelfde principe gebruiken als de groep voor mij. Dus een camera met een machine learning algoritme achter. Maar mijn focus zal liggen op dit idee beter aan te passen aan de use case. Dat wil concreet zeggen dat ik mij ga toespitsen op dit systeem zo energie efficiënt mogelijk te maken. Om dit te realiseren en hier nuttige vragen bij te kunnen stellen deel ik het systeem op in een paar onderdelen waarvan de haalbaarheid en nuttigheid individueel onderzocht kunnen worden. Deze opdeling resulteert in onderstaand schema.

Diagram

Description automatically generated

Zoals je ziet zijn de verschillende onderdelen: power harvesting -> Het verzamelen van energie uit de omgeving om zo de operationele duur van het systeem te verlengen. Movement detection -> Het detecteren van beweging verbruikt beduidend minder energie dan het nemen van een foto en het draaien van het algoritme, dit kan dus een goede trigger voor het systeem zijn. Ir led array -> We hebben licht nodig om een foto te maken. Aangezien de krabben niet van gewoon licht houden kiezen we hier voor ir. Esp32 -> hier gebeurt de echte magie, hier gaan we een foto nemen en het algoritme laten draaien en de tel bijhouden. Het doel is om dit deel zo lang mogelijk operationeel te houden.

Nu de opdracht opgedeeld is in stukjes kunnen we elk stuk apart gaan bekijken en beoordelen. Maar we houden wel in het achterhoofd dat het uiteindelijke doel de totale energie kost minimaliseren is en niet elk stuk apart minimaliseren.

# Movement detection

Zoals reeds vermeld kiezen we hier voor een microwave sensor ipv. Een standaard PIR sensor, dit omdat de krabben koudbloedig zijn. Bij dit deel komen er meteen enkele vragen naar boven die we hier gaan proberen beantwoorden. 1) Kan onze sensor door water penetreren zonder al teveel invloed van het water te ondervinden. 2) Wat is het energie verbruik van deze sensor en kunnen we dit nog verlagen? 3) Is het interessant om met een interval te werken ipv de sensor altijd aan te laten?

## Sensor water penetratie

De sensor werd boven een bak met 15 centimeter water gehouden om volgende tests uit te voeren.

* Beweging onder bak met water wordt correct geregistreerd? -> sporadisch, niet super betrouwbaar
* Beweging in bak met water wordt correct geregistreerd? -> Ja, meestal wel
* Bewegingen van het water zelf in de bak worden geregistreerd? -> enkel grote schokken
* Heel kleine bewegend element (steentje 1 op 0.5 cm) wordt geregistreerd? -> niet testbaar
* Heel kleine bewegend element (steentje 1.5 op 0.9 cm) wordt geregistreerd? -> niet testbaar

## Sensor energie verbruik

De maker van de sensor adverteert een max verbruik van 60mA en een typisch verbruik van 37mA. Zelf ontdekken we dat de sensor rond de 47mA verbruikt als er geen beweging wordt waargenomen en de interrupt pin niet verbonden is aan de esp32. Graphical user interface, application, table

Description automatically generated

Bij beweging ziet onze grafiek bij deze setup er zo uit Graphical user interface

Description automatically generated

Rond de 50 mA verbruik. Na de beweging blijft dit nog enkele seconden rond de 50 maar met minder grote pieken om vervolgens terug naar 47 mA te dalen. Dit is hellaas 10mA meer dan geadverteerd op de site van de verkoper. Bij het opstarten zien we ook steeds een kleine piek. Graphical user interface, chart

Description automatically generated with medium confidence

Maar deze piek wordt pas relevant als we met een interval gaan werken.

We zouden deze consumptie nog kunnen verlagen door de de power indicator led en movement indicator led los te solderen. Alhoewel dit geen groote besparing is helpen alle kleine beetjes in een project als dit. Na dit gedaan te hebben merken we dat we in idle nog maar rond de 43 mA verbruiken en bij bewegings detectie zien we enkel nog korte hogere pieken. Dit is omdat de indicatie led nu niet meer aan en uit springt. Chart

Description automatically generated with low confidence

Als we nu de interupt pin van onze sensor aansluiten op de esp32 zien we dat het verbruik terug naar 47 mA gaat. Maar zonder het lossolderen zou dit dus ongeveer 51 mA zijn. Het verwijderen van de bewegings indicatie led bespaard ook nog eens rond de 4mA als er beweging gedetecteerd wordt. Het gekke is dat we zien dat bij het aansluiten van de interrupt pin aan de esp32 we zien dat het algemene energie verbruik nu lager ligt als er beweging is tegenover als er geen beweging is. Ook de initiële opstart piek is nu hoger (80 mA) maar korter. Het verbruik in de opstart fase is nu ook lager dan het algemene verbruik. Ter illustratie foto bij het uit en vervolgens weer aanzetten van de senor (zonder indicatie leds en met interrupt aangesloten):

Chart, scatter chart

Description automatically generated

Hier ter illustraite van het verbruik bij beweging:

Chart

Description automatically generated

De resultaten hier zijn een beetje teleurstellend aangezien 47mA nog steeds best veel is voor onze toepassing. Maar deze 47mA + de 10mA van de esp32 in deep sleep is nog steeds een pak minder dan de consumptie tussen de 100 en 200mA die we bij het runnen van het model op de esp32 zagen. Dus dit is zeker een verbetering, alhoewel het niet de grote verbetering die ik hier graag gezien had is. Misschien kan het volgende deel, het gebruiken van een interval hier nog wat positieve verandering in brengen.

## Beweging sensor met interval

We kunnen proberen om de beweging sensor enkel om de x seconden te voor y seconden lang te gebruiken om zo stroom te besparen. Belangrijk hier is dan dat we weten hoelang het duurt vanaf het opstarten voor de sensor beweging kan detecteren. We moeten ook uitrekenen hoe groot x minimaal moet zijn om winst te maken op ons stroom verbruik. Hoe groter x is hoe beter uiteraard, maar hier moet ook een grens opgelegd worden omdat we anders krabben kunnen missen. Uiteindelijk moeten we de totale latency van het systeem ook kennen. Want het heeft geen zin om x kleiner te maken dan de tijd die het systeem nodig heeft om 1 trigger volledig te verwerken.

### Interval tijd

Zoals we hiervoor al zagen hebben we een kleine piek bij het opstarten, hier moeten we rekening mee houden bij het berekenen van de minimum interval tijd.

Chart

Description automatically generated

Het vervelende hier is dat dit stuk niet altijd even lang doorloopt. Er zit zelfs heel veel verschil op. Daarom denk ik dat het veiligste is om in onze berekening deze initiële daling te negeren en overal gewoon het idle verbruik te nemen. Dit omdat deze piek soms maar enkele milliseconden duurt en soms enkele seconden. We nemen ons verbruik dus best iets te ruim dan te smal omdat we deze duur niet op voorhand kunnen voorspellen. We zullen dus overal 47mA gebruiken. Bij het testen hier viel het ook op dat de sensor niet meteen na het opstarten klaar is om beweging te detecteren. Ook zien we dat de sensor vlak na het opstarten altijd beweging detecteert, wat dat eerste, iets lagere stuk kan verklaren. Dit is enorm vervelend voor het bepalen van een ideaal interval omdat zoals eerder vermeld de duur van deze piek niet voorspelbaar is. Na vele malen testen denk ik dat we de initiële opstarttijd kunnen leggen op 2s. Vaak is het veel minder dan 2s maar we moeten hier voorzichtig zijn als we uiteindelijk willen besparen met onze esp32 zelf. Op dezelfde manier is het ook niet erg als we eens over de 2s zitten, dan draaien we ons systeem maar een keer voor niks. y de duur van elk moment in het interval moet dus al minstens 2s zijn. Als we dan nog 1s nemen voor het detecteren van beweging hebben we een y van 3s aan max 47mA.

### Tijd tussen interval

Aangezien we ontdekt hebben dat de initiële opstart fase minder verbruikt dan de normale staat en bovendien bij het detecteren van beweging gemiddeld minder verbruikt wordt dan normaal maakt het eigenlijk niet uit hoeveel tijd er tussen onze intervallen zit. We gaan altijd winst maken tegenover als we de sensor full-time laten werken. We moeten wel onthouden dat we de sensor altijd minstens 3s laten werken. We moeten natuurlijk wel nog een logisch gekozen redelijke tijd kiezen. (TODO gemiddelde snelheid krab?)

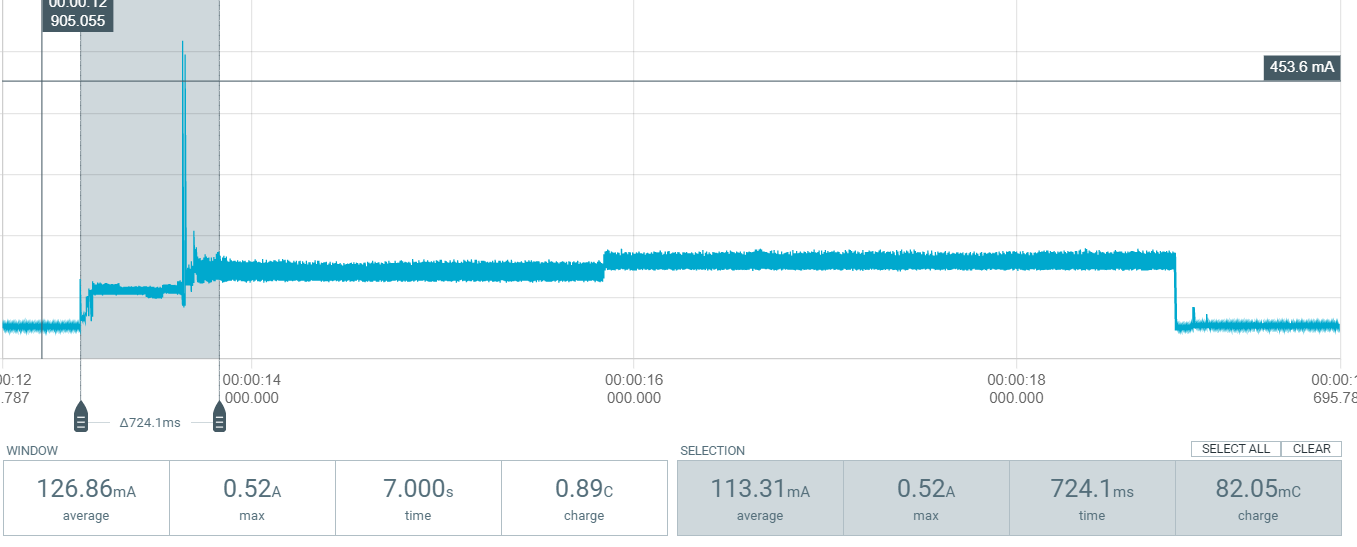
## Experiment 1

Parameters: tijd tussen interval = 5s, interval length = 5s

Table

Description automatically generated with medium confidence

Shows => esp32 in deepsleep, motion detector on



Shows => esp32 comming out of deep sleep, running setup, time not accounted for in 5s intervals

Table

Description automatically generated

Shows => esp32 active and microwave active, no motion detected

Table, timeline

Description automatically generated with medium confidence

Shows => esp32 active and microwave active, picture taken and infernce ran at 2s. (Motion is evaluated every 1s so this means that motion was detected between 1s-2s, no more motion detected between 2s-3s because the signal does not start another peak at 3s)

Timeline

Description automatically generated

Shows => time and power needed to take picture and run inferance.



Shows => inferance result (no crab detected in picture)

# Exp1

Compare power with use of movementsensor to without movement sensor

## No movement sensor

Take picture and run inference every second for 10 seconds, between 0 and 10 crabs detected do 10 times for every amount of crabs to get average and standard deviation.

### 0 crabs detected

Below screenshot of first run average of 114.75ma

Timeline

Description automatically generated with low confidence

Lets do this 9 more times and see what we get

* 2nd run -> 114.45ma
* 3d run -> 114.50ma
* 4d run -> 114.36ma
* 5d run -> 114.54ma
* 6d run -> 114.84ma
* 7d run -> 114.78ma
* 8d run -> 114.70ma
* 9d run -> 114.86ma
* 10d run -> 115.04ma

Allemaal samen geeft dat een gemiddeld gemiddeld verbruik van 114.682ma met een standaard deviatie van 0.21347912. Wat wel vervelend is is dat als we de setup langer laten runnen we een hoger gemmidelde waarde vinden. Hoelanger de setup runt hoe hoger het gemiddeld verbruik wordt (klein verschil wel). Dit is doenbaar met 0 krabben gedetecteerd elke 10s maar voor meer dan 0 wordt dit bijna ondoenbaar om dit over een langere periode dan 10s te doen. Over 290s zien we in dit geval een gemiddeld verbruik van 115.25ma. 🡪VRAGEN

### 10 crabs detected

I suspect that there is no real difference in power consumption between wether an inference detects a crab or not. So to possibly save some time lets go straight to 10 crabs and see if this is the same as 0. To clarrify: 10 crabs means that every second a picture gets taken with a crab in it, we do this 10 times.

The first run gives an average power consumption of 114.30ma, which is a little bit less than with 0 crabs. Timeline

Description automatically generated

Lets run it 9 more times:

* 2 -> 114.51ma
* 3 -> 114.66ma
* 4 -> 114.73ma
* 5 -> 114.79ma
* 6 -> 114.71ma
* 7 -> 114.86ma
* 8 -> 114.91ma
* 9 -> 114.86ma
* 10 -> 114.97ma

This gives us an average of 114.73ma with a standard deviation of 0.20165977949672 which is close enough tot he values of the test with 0 crabs detected to conclude that for his setup it does not matter if a crab is detected or not. Which is what I sort of expected because in both cases you take a picture and run the inference and wether a crab is detected or not is only the result of the inference.

## With movement sensor

Now our esp32 is in deepsleep while our movement sensor is constantly on. Esp32 only wakes up when movement sensor detects movement and sends an interupt. We do this again 10 times for each number of crabs between 0 and 10. We proved earlier that wether a crab is actually detected or not during a scan has no effect on the actual power consumption. So here we care about the number of scans and not the actual amount of detected crabs since here it does not matter.

### 0 scans

So this is bassicly just the power consumption of the sensor + esp32 in deepsleep. Values measuret => [49.54, 49.53, 51.85, 49.54, 49.57, 52.48, 52.22, 49.57, 49.56, 52.21]

This gives a mean of 50.607 ma and a standard deviation of 1.3706612353978 . This is ofcourse less than without the movement sensor. But I want to know exactly where the tipping point lies. The value 50.607 ma is completly within my expectations.

### 1 scan in 10s

Example photo:

Chart

Description automatically generated

***Looking at this photo I just realized that this is not a fair comparisson since it takes longer with the movement sensor to scan for crabs then without. I should later redo the measurements without the sensor and increase the time between pictures to match the time of the setup with the movement sensor (1.612s). With 1 scan taking longer than 1s it is also not possible to do 10 scans in 10s zo I will test from 0 to 5 scans in 10 seconds from now on.***

Measurements => [65.81, 65.95, 63.84, 63.57, 64.22, 66.13, 65.87, 65.95, 66.08, 65.84]

M => 65.326

S => 1.0166961744352

### 2 scans in 10s

Measurements => [79.85, 79.74, 79.43, 77.40, 77.92, 78.47, 78.43, 78.32, 78.40, 78.43]

M => 78.639

S => 0.79101130768712

### 3 scans in 10s

Measurements => [91.70, 91.71, 92.10, 91.78, 91.50, 91.15, 91.74, 91.86, 92.20, 91.83]

M => 91.757

S => 0.29261464989527

### 4 scans in 10s

Measurements => [106.09, 106.12, 106.07, 105.51, 105.64, 105.93, 106.16, 105.92, 105.95, 106.02]

M => 105.941

S => 0.21136855647581

### 5 scans in 10s

Measurements => [119.37, 118.92, 118.68, 118.84, 118.67, 118.74, 118.54, 118.41, 118.28, 118.67]

M => 118.712

S => 0.29913950665943

## Redoing without movement sensor

As previously stated the comparisson was not fair since the setup with movement sensor took way longer for one detection. So lets redo the measurement. Luckily we can just change the picture and inferance interval to match the rate of the other setup. We already proved that wether a scan actually detects a crab or not does not matter, so we can just do the measurement for 0 actual crabs detected

### 0 crabs V2

Measurements => [107.94, 108.22, 107.94, 107.64, 107.82, 107.9, 108.18, 107.59, 107.52, 107.95]

M => 107.87

S => 0.23380903889

## Experiment resultaat