# Ontwerp IN2710-D/TI2710-D

Jeroen Bareman 4035186
Roelof Sol 4012194
Mick van Gelderen 4091566
Luyt Visser 4016603
Jelle Licht 410694
Groep 6
December 6, 2011

## Inhoudsopgave

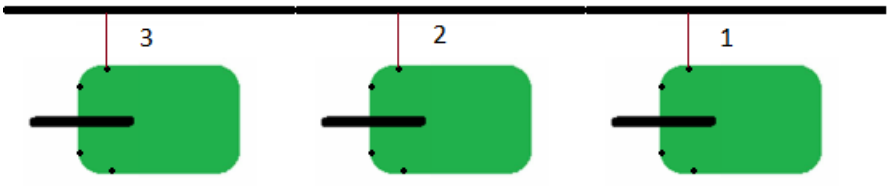
# Signaalverwerking

## Afstandsensoren

Uit het plot van de sensorwaarden is gekomen dat de afstandsensoren soms niet de goede waarden geven. Het kan dus voorkomen dat de tank geen muur ziet terwijl deze muur er wel is.

Voor dit probleem heeft de projectgroep de volgende oplossing bedacht. Zodra een afstand sensor een waarde meet die sterk afwijkt van de vorige waarde moet de meting gecontroleerd worden.

Voorbeeldsituatie: De tank rijdt rechts langs een muur in de volgmuur status en meet uit zijn rechter afstand sensor de volgende waarden: Eerste meting: 30cm. Tweede meting: 31cm. Laatste meting 100cm. Zoals in Figuur 0 te zien is moet de laatste meeting, aangegeven met een 3, ook in de buurt van 30 cm liggen. Deze meting is dus fout.




Figuur 0

Volgens de oplossing van de projectgroep wordt meting 3 nu gecontroleerd omdat deze meting sterk afwijkt van de vorige metingen. De meting zal als volgt gecontroleerd worden:  De tank stuurt al rijdende naar rechts, hierbij zal de tank zijn sensoren extra vaak uitlezen door een voorzichtigheid vlag aan te zetten. De tank gaat nu draaien en naar de muur toe rijden, omdat de hoek die de afstand sensor met de muur maakt veranderd en de afstand met muur kleiner wordt, zal er na een aantal metingen een getal onder de 30cm gemeten moeten worden. Omdat de tank zich in de volgmuur status bevind zal de tank naar links bijgestuurd worden. De voorzichtigheidsvlag zal automatisch na een aantal cycli van het Pledge algoritme worden uitgezet en het probleem is opgelost.

Er kunnen meerdere voorbeeld scenario's bedacht worden, bijvoorbeeld als de tank niet langs een linker muur rijdt maar langs een rechtermuur en opeens geen muur meer ziet. Er kan ook een situatie ontstaan waarbij de tank een tijdje niks ziet en opeens wel wat. De oplossing van deze problemen zal analoog zijn aan de uitgelegde voorbeeld situatie; De voorzichtigheid vlag wordt aangezet en de tank stuurt naar de afwijkende meting toe om deze meting te controleren.

## Compassensor

Het compas geeft het aantal graden aan in een integer in de range 0 - 3600. Aan ons is de taak om dit te interpreteren. We definieren 0 als noord, 900 als oost, 1800 als zuid en 2700 als oost.

Om het pledge algoritme uit te voeren is nodig om bij te houden hoeveel de tank draait. Tijdens het maken van een bocht zal het compas een aantal keer aangeroepen moeten worden en het totaal aantal graden dat de tank draait in een bocht zal afgeleid moeten worden uit het compas.

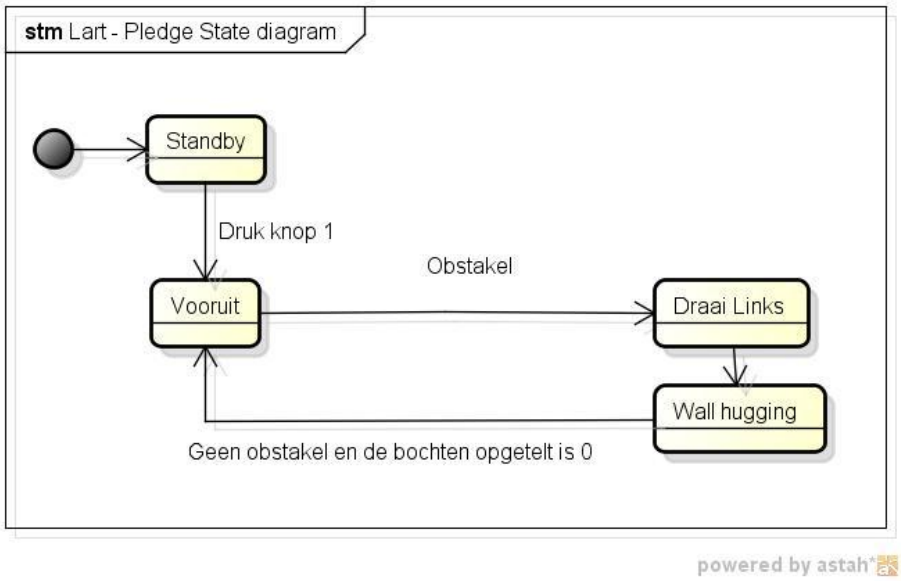
Om te weten hoeveel graden de tank gedraaid is tussen 2 compasmetingen nemen we het verschil van de compasmetingen. We voorzien wel een probleem, namelijk als het compas een meting heeft van 3500 en een volgende meting van 100. Nu zal het verschil van de metingen uitwijzen dat de tank 340 graden gedraaid is. Dit verschil klopt niet.

De oplossing die de projectgroep heeft bedacht gaat als volgt: Als een het verschil van 2 waarden meer dan 1800 is dan wordt bij het kleinste getal 3600 opgeteld en wordt van dit getal en het grootste getal het verschil genomen. Dit verschil is het juiste aantal graden. Dit zal werken omdat de tank terwijl hij een bocht maakt steeds het compas uitleest, het

verschil van 180 graden zal dus alleen voorkomen als het compas over de 0/3600 grens loopt. Hierdoor kan altijd het juiste aantal graden bepaald worden dat de tank draait en is dit probleem opgelost.

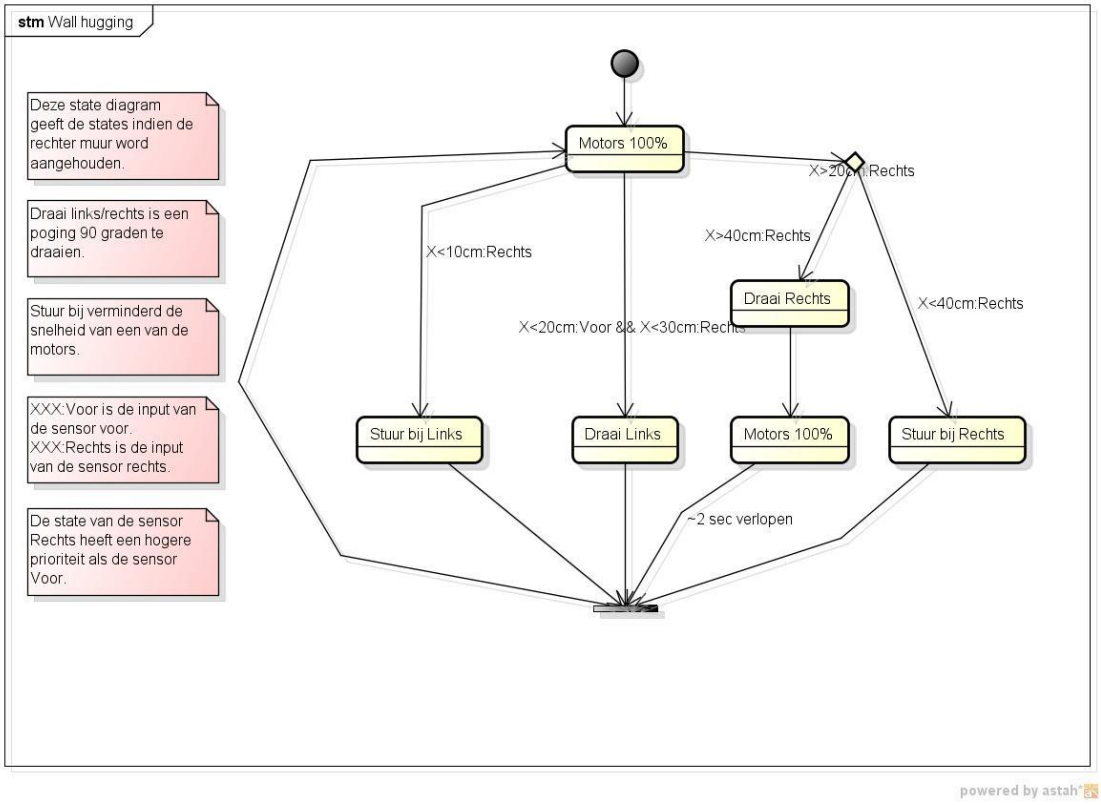
## Statemachine Pledge

De versimpelde state diagram voor de robot is weergegeven in de onderstaande afbeelding. Dit ontwerp is de essentie van Fidge algoritme. Aangenomen is dat de tank de rechter muur zal volgen.




Figuur 1

De uitbreiding van de Wall hugging state is als volgt:




Figuur 2

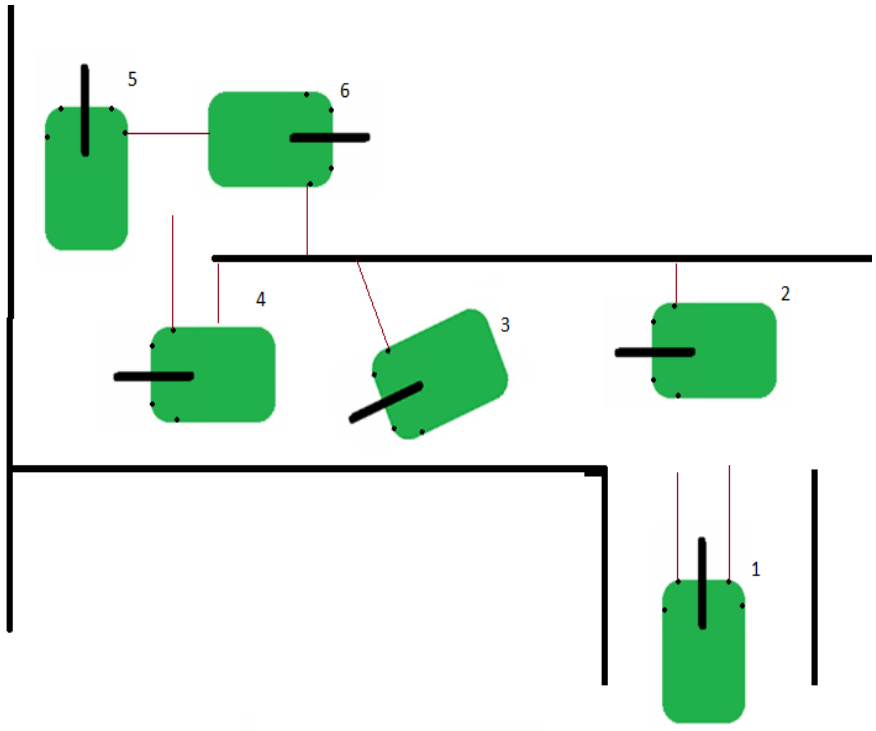
In de volgende alinea's wordt een voorbeeldscenario beschreven. De superscript cijfertjes verwijzen naar de positie van de tank in figuur 3.

De tank wordt aangezet. Zodra knop 1 wordt ingedrukt zal de tank de waarde van het kompas opslaan als Dit is de basis righting[1.] Hij zal nu gaan rijden totdat er een obstakel wordt gedetecteerd.
Op dat moment zal de tank gaan bijhouden hoeveel hij gedraaid heeft. Om de muur aan de rechter kant te krijgen wordt er 90 graden naar links gedraaid[2]. Nu zal de robot vooruitgaan. Om te zorgen dat de tank niet te ver van de muur afdwaalt of juist te dicht bij komt zal er met pulse width modulation de motor op verschillende snelheden worden aangedreven [3].

Uit tests moet nog blijken wat de mogelijkheden zijn van de pulse width modulation. Zo weten we nog niet wat de mogelijke snelheden zijn en of de rupsbanden wel vast blijven zitten. Mocht het heel effectief zijn dan zou het misschien mogelijk zijn de statemachine te versimpelen door de Draai functie te vervangen door de Stuur bij functie.

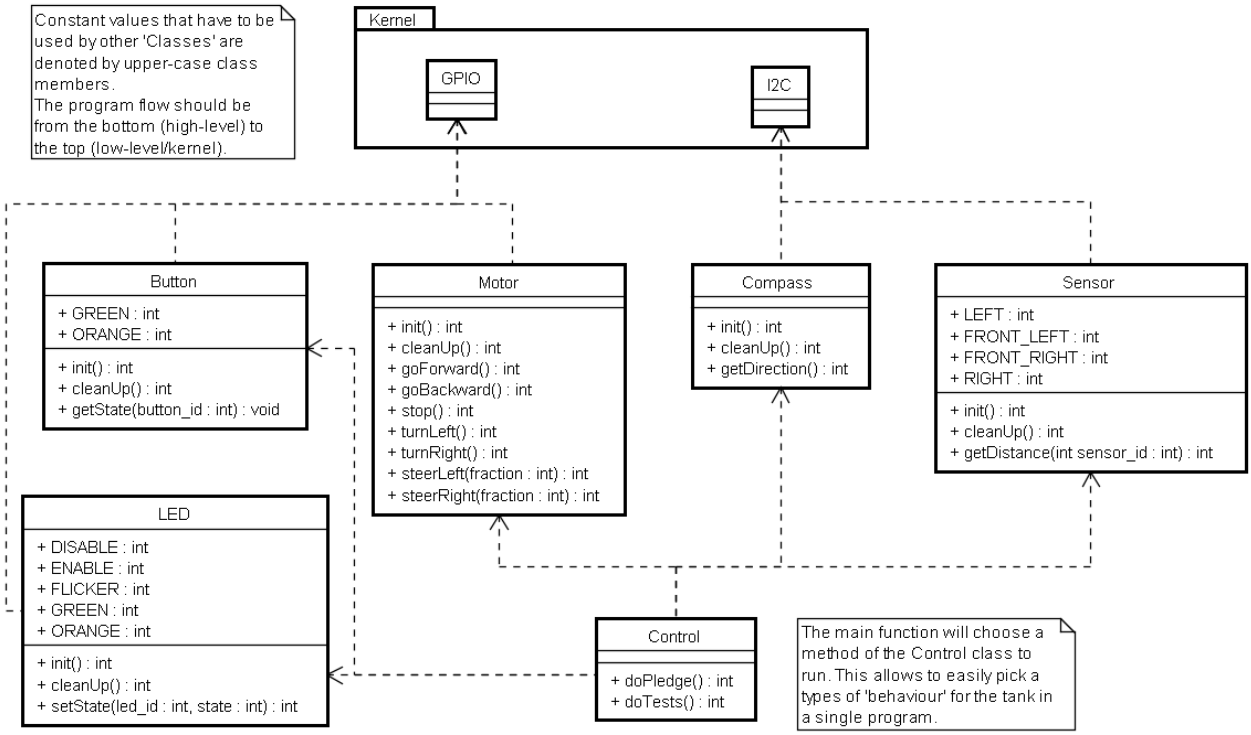
Zodra de tank merkt dat er geen muur meer aan de rechter kant zit zal de tank draaien[4]. Daarna zal de tank enkele seconden vooruit rijden om te kijken of het mogelijk is om weer rechts te draaien.

Er zijn nu twee mogelijkheden. Als punt 1 het begin is dan zal de tank bij punt 5 weer terug switchen na de 'vooruit' state. Immers is de som van de bochten 0 en staat hij weer in dezelfde richting als aan het begin. Mocht de som van de bochten niet 0 zijn dan zal de tank door krijgen dat er geen muur rechts van hem zit en weer 90 graden bij draaien en rijden zodat er weer een muur gedetecteerd word door de sensor.




Figuur 3

## Het klasse diagram




Het klasse diagram kun je opdelen drie lagen, 1) de controle laag, 2) de componenten laag en 3) de kernel laag. In het diagram is echter alleen de kernel laag apart weergeven. Laag 1 en 2 liggen namelijk zo dicht bij elkaar dat het programmatig gezien niets toevoegt om ze anders te behandelen. Het doel van diagram is om te laten zien hoe de tank bestuurd kan worden.

### De controle laag

De controle laag bevat functies die de tank een bepaald gedrag laten vertonen. Hier is voor gekozen omdat er op deze manier maar één programma nodig is voor het testen van de kwaliteit van een tank en voor het vinden van de uitweg van een doolhof. Zo worden ook altijd de meest recente implementaties van de componenten en kernel gebruikt.

### De componenten laag

De componenten laag bevat software om de geassocieerde hardware te besturen. De Motor klasse valt daar een klein beetje buiten omdat deze in feite twee motoren bestuurt en daarvoor nog een aantal geavanceerde functies voor bevat. Het project is echter zo specifiek en klein dat deze kleine inconsistentie geen problemen op zal leveren.

De geavanceerde functies in de Motor klasse zijn de functies die de tank laten bijsturen. Een van die functies is ‘steerLeft(fraction : int)’. Hierin representeert het argument fraction een waarde tussen 0 en 100. Een waarde van 50 zal de linker motor op 50% en de rechter motor op 100% laten draaien zodat de tank langzaam bijstuurt.

De klassen Button, Sensor en LED bevatten constante waarden die ingevuld kunnen worden als ID of state bij hun eigen functies. Dit is prettiger dan bijvoorbeeld 6 functies om twee LEDs op een bepaalde state te zetten.

### De kernel laag

De kernel laag bevat interfaces om met een aantal device drivers te communiceren. De precieze implementatie ervan zal grotendeels hetzelfde zijn als de standaard linux gpio en i2c drivers.

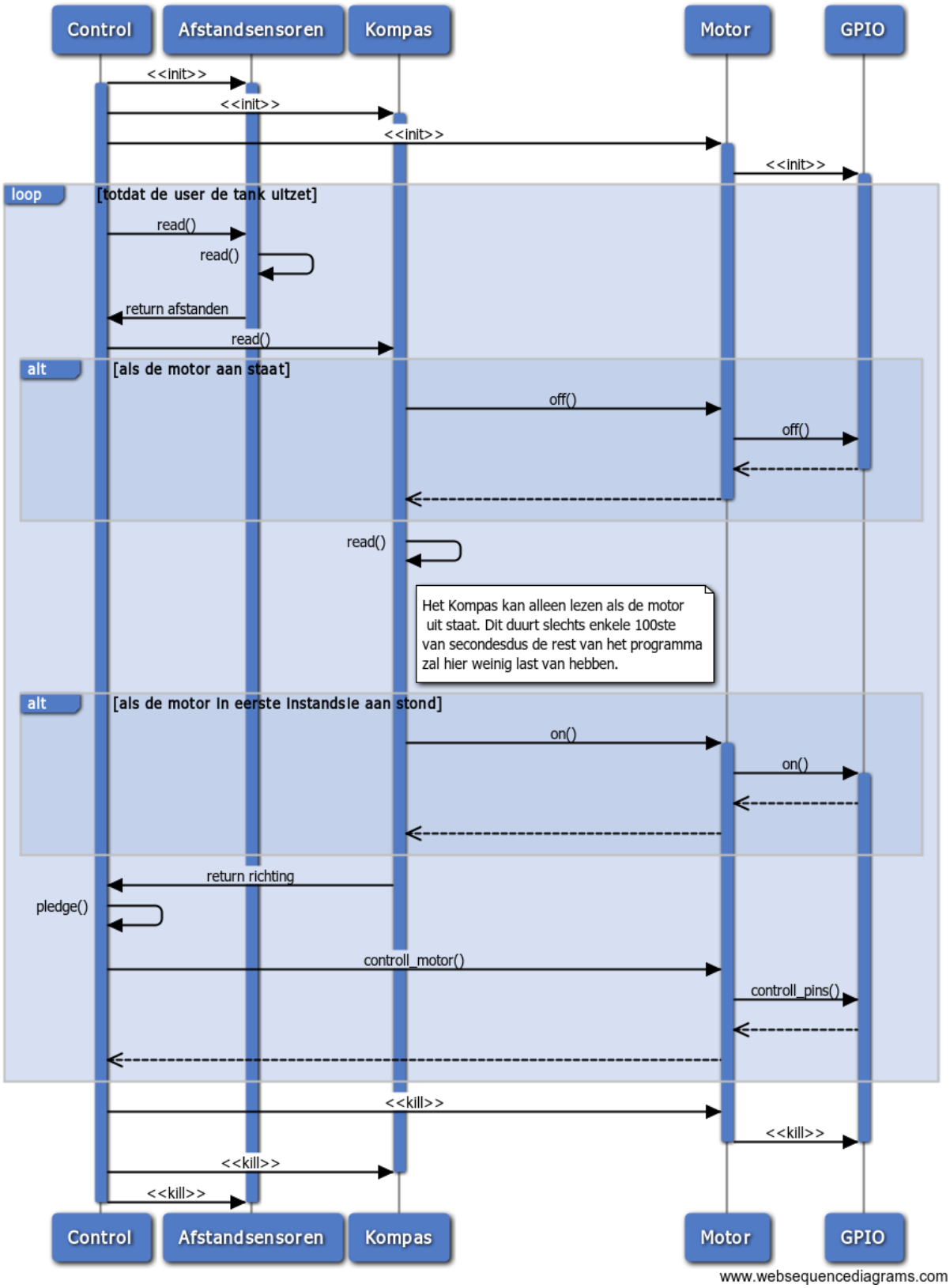
## Extra opmerkingen

Het is nog mogelijk om een Log klasse te maken. Deze zal dan functies bevatten waarmee data weggeschreven kan worden naar de tank om deze gegevens later terug te kunnen zien. Dit is verder niet interessant genoeg voor het bovenstaande diagram dat zich op de tank functionaliteit concentreert.

## Sequence Diagram main loop

```
Control -> Afstandsensoren : <<init>>
activate Control
activate Afstandsensoren
Control -> Kompas : <<init>>
activate Kompas
Control -> Motor : <<init>>
activate Motor
Motor -> GPIO : <<init>>
activate GPIO

loop totdat de user de tank uitzet
Control -> Afstandsensoren : read()
Afstandsensoren -> Afstandsensoren : read()
Afstandsensoren -> Control : return afstanden
Control -> Kompas : read()

alt als de motor aan staat
Kompas -> Motor : off()
Motor -> GPIO : off()
GPIO --> Motor :
deactivate GPIO
Motor --> Kompas :
deactivate Motor
end

Kompas -> Kompas : read()

note left of Motor
Het Kompas kan alleen lezen als de motor\n uit staat. Dit duurt slechts enkele
100ste \nvan secondesdus de rest van het programma \nzal hier weinig last van
hebben.
end note

alt als de motor in eerste instandsie aan stond
Kompas -> Motor : on()
activate Motor
Motor -> GPIO : on()
activate GPIO
GPIO --> Motor :
Motor --> Kompas :
end

Kompas -> Control : return richting
Control -> Control : pledge()
Control -> Motor : controll_motor()
Motor -> GPIO : controll_pins()
GPIO --> Motor :
Motor --> Control :
end


Control -> Motor : <<kill>>
```

```
Motor -> GPIO : <<kill>>
deactivate GPIO
deactivate Motor
Control -> Kompas : <<kill>>
deactivate Kompas
Control -> Afstandsensoren : <<kill>>
deactivate Afstandsensoren
deactivate Control
```


Control
Afstandsensoren
Kompas
Motor
GPIO
<<init>>
<<init>>
<<init>>
<<init>>
loop
[totdat de user de tank uitzet]
read()
read()
return afstanden
read()
alt
[als de motor aan staat]
off()
off()
read()
Het Kompas kan alleen lezen als de motor uit staat. Dit duurt slechts enkele 100ste van secondesdus de rest van het programma zal hier weinig last van hebben.
alt
[als de motor in eerste instandsie aan stond]
on()
on()
return richting
pledge()
controll_motor()
controll_pins()
<<kill>>
<<kill>>
<<kill>>
<<kill>>
Control
Afstandsensoren
Kompas
Motor
GPIO
www.websequencediagrams.com

## Gebruikersscenario's

### Scenario 1 : starten.

Pre: de tank is aan maar nog niet gestart.
Post: de tank blijft rijden totdat hij wordt uitgezet

| User | System |
|---|---|
| 1) De gebruiker klikt op één van de twee knoppen | 2) Het systeem zal alle componenten (afstand sensoren, kompas, motor, GPIO) initialiseren.<br>3) De tank leest de afstand sensoren uit<br>4) De tank leest het kompas uit (en zet eventueel de Motor kort uit)<br>5) De tank berekend volgens het Pledge algoritme hoe hij verder moet rijden.<br>6) De motoren van de tank worden gebruikt zodat de tank de juist berekende route gaat rijden.<br>7) -> stap 3 |

### Scenario2: afsluiten.

Pre: de tank moet aan zijn en gestart.
Post: de tank blijft aan maar blijft stil staan.

| | |
|---|---|
| 1) De gebruiker klikt op één van de twee knoppen | 2) De tank zal alle componenten opschonen en uitschakelen. |

## Planning

| | *05-12-11* | *08-12-11* | *12-12-11* | *15/12/11* | *19/12/11* | *22/12/11* | *09-01-12* | *12-01-12* | Totaal |
|---|---|---|---|---|---|---|---|---|---|
| **uren** | 3 | 3,75 | 3,75 | 3,75 | 3,75 | 3,75 | 3,75 | 0 | 25,5 |
| **aanwezig** | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 40 |
| **bezigheid** | opdelen kernel/userspace | uitwerken implementatie | uitwerken implementatie | testen en verbeteren | testen en verbeteren | oefenparcours | laatste aanpassingen | eindwedstrijd | werkende robot |
| **extra** | begin implementatie | | | verslag | verslag | verslag | eindverslag | | prachtig verslag |
| **uurbesteding** | 15 | 18,75 | 18,75 | 18,75 | 18,75 | 18,75 | 18,75 | 0 | 127,5 |