

Een kruipende, zelflerende robot

Arne Vanheesbeke, Jonas Van Der Donckt, Jules Noppe

21 mei 2017

Voorwoord

Voor het vak Micrcontrollers werd van ons verlangd een project te maken met een Dwenguino-Board. Tijdens de eerste twee lesweken van het tweede semester werden labo oefeningen opgegeven om vertrouwd te geraken met het Dwenguino-Board en de Atom programmeeromgeving.

Het project dat wij - Jonas Van Der Donckt, Arne Vanheesbeke en Jules Noppe - hebben gekozen is de Q-learning robot. Andere mogelijkheden waren: een IR afstandsbediening, een tekentafel of een HDD klok. De keuze om voor de Q-learning robot te gaan, was snel gemaakt, aangezien we allen een grote interesse vertoonden in dit onderwerp.

In dit project werden we bijgestaan door prof. dr. ir. Francis wyffels en ir. Tom Neutens, die we willen bedanken voor hun bijdragen.

Inhoudsopgave

1 Inleiding	1
1.1 Probleem en doelstelling	1
1.2 Vereisten	2
2 Aansturing van de verschillende componenten	3
2.1 Servo's	3
2.2 Hoekencoder	5
3 Design	7
3.1 Materiaal	7
3.2 Kruiparm	7
3.3 Frame	8
4 Wat is Q-learning?	9
4.1 theoretische achtergrond	9
4.2 De formule	10
4.3 Algoritme	11
5 Visualisatie van de Q-tabel	12
5.1 USART communicatie	12
5.2 Software	12
5.3 Bluetooth communicatie	13
5.3.1 Protocol	13
5.3.2 Bluetooth module	14
6 Meetresultaten en besluit	15
6.1 Meetresultaten bij variërende parameterwaarden	15
6.1.1 Random geïnitialiseerde Q-tabel	16
6.1.2 Niet geïnitialiseerde Q-tabel	17
6.2 Besluit	19

Hoofdstuk 1

Inleiding

1.1 Probleem en doelstelling

De opdracht van dit project bestaat eruit een robot te ontwerpen die zelfstandig kan zoeken naar een efficiënte manier om vooruit te gaan. Hiervoor moet een robot worden gemaakt, bestaande uit één arm en een onderstel met wielen. Deze moet in staat zijn om zichzelf vooruit te trekken door de op de arm gemonteerde servo's, op een zo correct mogelijke manier te benutten. Het geheel wordt aangestuurd met behulp van een Dwenguino-board.

We kunnen deze topic kaderen binnen het domein van 'Machine learning'. Machine learning is in feite het vermogen om zich aan te passen aan een nieuwe situatie. Iedere situatie brengt informatie mee en machine learning zal hierbij gebruikt worden om een patroon te ontdekken en dit toe te passen. De machine kan door middel van een algoritme zelf een soort regelsysteem ontwikkelen om de gepaste output te koppelen aan een gegeven input. Machine learning vormt hierdoor een niet te omzeilen onderdeel van het onderzoeksgebied van kunstmatige intelligentie, omdat dit ervoor zal zorgen dat de machine kan evolueren.

In dit project zullen we het probleem benaderen aan de hand van het 'Q-learning algoritme'. Door middel van dit algoritme zal de robot in staat zijn een efficiënte manier te achterhalen om vooruit te komen. Het volledige Q-learning proces wordt verder in dit verslag volledig uitgediept.



Figuur 1.1: Kruipende, zelflerende robot

1.2 Vereisten

Volgende basisvereisten werden doorgrond en benaderd doorheen het project:

- Nagaan hoe de hoekencoder werkt en deze op een correcte manier kunnen gebruiken. Het is belangrijk dat zowel kleine als grote veranderingen door de hoekencoder kunnen waargenomen worden.
- De servo's (zichtbaar op figuur 1.1), die instaan voor de beweging van de arm, moeten afzonderlijk kunnen aangestuurd worden.
- De robot mag niet onmiddellijk beginnen bewegen wanneer deze van stroom wordt voorzien. Er moet een knop worden geïmplementeerd dat de robot laat starten met het leerproces.
- Het Q-learning algoritme moet correct werken met voor elke servo vier mogelijke acties die de robot kan gebruiken. Dit laat de robot toe om 16 verschillende toestanden te bereiken. De robot moet bijgevolg in staat zijn om autonoom te achterhalen wat de meest efficiënte manier is om vooruit te komen.

Hoofdstuk 2

Aansturing van de verschillende componenten

2.1 Servo's

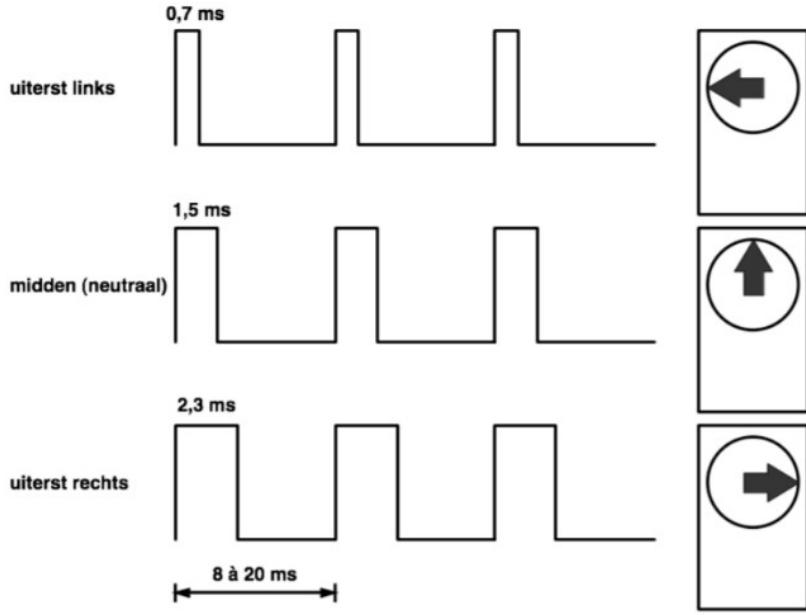
Om de arm te vervaardigen, werd gebruik gemaakt van 2 servo's, die los van elkaar kunnen bewegen. De plaatsing van beide servo's is duidelijk te zien op figuur 2.1.



Figuur 2.1: Servo's worden gebruikt om de arm te construeren

Het aansturen van de servo's gebeurt via een PWM-signaal (Pulse Width Modulation). De breedte van een puls is op zich een maat voor de hoek die wordt verdraaid. Hoe langer de pulse duurt, des te groter de draaihoek wordt. Figuur 2.2 geeft een mooi idee van hoe de aansturing van de servo's in elkaar zit.

De periode bedraagt 20 milliseconden. Deze periode wordt dan opgedeeld naar gelang het aantal servo's die men wenst te gebruiken.



Figuur 2.2: Servo-aansturing via PWM-signalen

Zoals te zien is op figuur 2.2 wordt de hoek die de servo moet verdraaien dus ingesteld met de breedte van de puls. Een puls van 0.7ms zal de servo uiterst links gaan plaatsen terwijl een puls van 2.3ms de servo uiterst rechts zal plaatsen. Dit zijn uiteraard de theoretisch beschouwde waarden en kunnen dus wat afwijking vertonen ten opzicht van de praktisch waarden, waar er slechts maximum per graad kan aangestuurd worden. De periode van 20ms wordt opgesplitst in 2 delen: voor elke servo een halve periode (10ms). Dit stelt ons in staat om 2 servo's aan te sturen met één timer. Aangezien de theoretisch maximale hoekverdraaiing 2.3ms bedraagt, zullen de aansturingssignalen van beide servo's niet in elkaar's vaarwater komen.

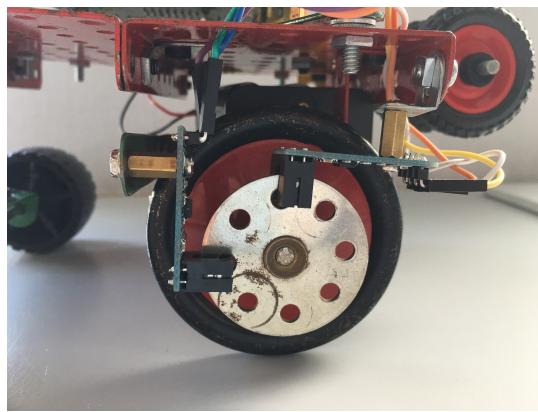
De posities van beide servo's worden dus gestuurd via één gemeenschappelijke timer, met bijhorende interruptafhandeling. Voor het aansturen van de servo's werd gekozen voor een 16 bit timer. Dit met als reden de nauwkeurigheid van de hoekverdraaiingen te optimaliseren. Bij een 16 bit timer kan een hoek met een nauwkeurigheid van om en bij de 0.1 graad worden gewerkt. Wanneer men een 8 bit timer had gekozen, zou men maar met een nauwkeurigheid van 7 graden kunnen werken. Dit laatste argument heeft ervoor gezorgd dat geopteerd werd voor een 16 bit timer.

De interruptafhandeling van deze timer bestaat eruit het correct doorsturen van het PWM-signalen naar de servo. Het signaal moet een zekere tijdsduur hoog en laag worden gezet en aan de hand van de interrupts die gegenereerd worden, kan dit op een zeer efficiënte manier.

Om de snelheid, waarmee de servo's bewegen, aan te passen, werd een interpolatie toegepast. Om een vaste interpolatietijd wordt nagegaan of de streefwaarde van de servo's reeds bereikt werd. Indien dit niet zo is, worden de servo's elk 1 graad bijgestuurd met oog op de doelhoek van beide servo's.

2.2 Hoekencoder

Om na te gaan of de robot, tijdens een actie, vooruit of achteruit ging, werd gebruik gemaakt van een hoekencoderopstelling. Een hoekencoder bestaat uit 2 optosensoren die geplaatst worden op een ronde schijf met daarin boringen voorzien. Figuur 2.3 geeft dit weer.

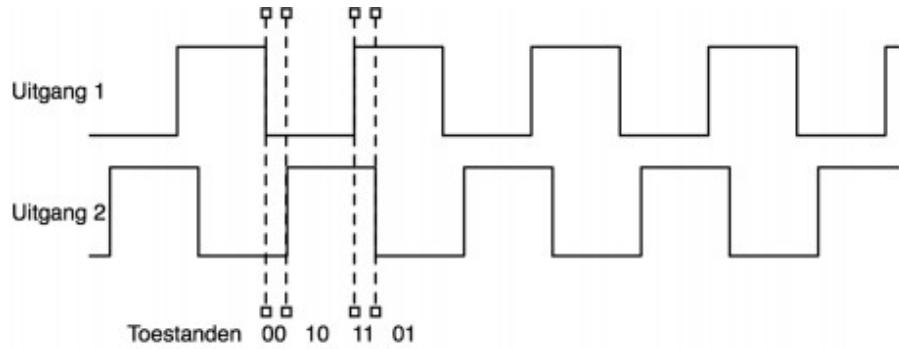


Figuur 2.3: Plaatsing van de Optosensoren

De schijf is tevens bevestigd op een as, waarop de wielen zijn vastgemaakt. Op deze manier draait de schijf met dezelfde hoeksnelheid als de wielen en in dezelfde richting. Zo kan later de bewegingsrichting (vooruit/achteruit) en de afgelegde afstand worden bepaald.

De optosensor bestaat uit een infrarood LED dat licht uitstuurd en vervolgens wordt opgevangen door een fototransistor. Wanneer men deze optosensor aanbrengt op een schijf waarin boringen zijn aangebracht, wordt het infrarood licht, wanneer de robot beweegt, telkens onderbroken en doorgelaten. Aangezien één optosensor niet volstaat om de bewegingsrichting van de robot te achterhalen, zal een tweede moeten worden geplaatst. Op deze manier krijgt men twee signalen die licht verschoven zijn ten opzichte van elkaar. Dankzij deze twee signalen kan men nu 4 verschillende toestanden afleiden. Aan de hand van de sequentie waarmee de vier toestanden elkaar opvolgen kan worden nagegaan wat de bewegingsrichting van de robot is. Figuur 2.4 geeft een beeld weer van deze kwadratuur waarbij de vier toestanden duidelijk waarneembaar zijn.

Om dit te verwezenlijken, werden externe pin interrupts gebruikt, die bij elke flank (stijgende en dalende) een interrupt genereren. Bij elke interrupt wordt eerst gecontroleerd of de sequentie al dan niet aangepast werd ten opzichte van de vorige genereerde interrupt. Indien de sequentie veranderd is, zal een counter geïncrementeerd worden, die bijhoudt in welke mate (hoe ver) de robot vooruit/achteruit ging.



Figuur 2.4: Kwadratuur van de optosensor-signalen

De sequentie van de kwadratuur is te vergelijken met een twee bit gray-counter. Dit is een counter waarbij telkens slechts één enkele bit verandert bij de overgang van de ene naar de andere toestand. Onderstaande tabel geeft een overzicht van de twee bit gray-counter sequenties en de bijhorende bewegingsrichting die eruit kan afgeleid worden.

gray-counter vooruit	gray-counter achteruit
00	00
01	10
11	11
10	01

Tabel 2.1: Gray-counters voor bewegingsrichtingsbepaling

Deze gray-waarden worden dan naar binair omgezet en vergeleken met de vorige waarden.

Hoofdstuk 3

Design

3.1 Materiaal

De gehele robot, met uitzondering van de kruiparm, werd vervaardigd uit Meccano. Meccano is een soort montagespeelgoed, dat wordt vervaardigd uit tal van verschillende onderdelen, die moeten worden bevestigd met moertjes en boutjes.

Het wagentje op zich heeft zes losstaande wielen. Onder het frame zijn 4 wielende bevestigd om stabiliteit te waarborgen. Aan de achterkant werden nog een extra paar wielen bevestigd, om het omkantelen van de robot tegen te gaan.

Het Dwenguino board zelf werd gemonteerd bovenop het frame. Net onder het frame is een uitsparing voorzien om de batterijhouder te plaatsen.

De hoekencoderopstelling, 2 optosensoren, werd bevestigd rond één van de achterwielen.

3.2 Kruiparm

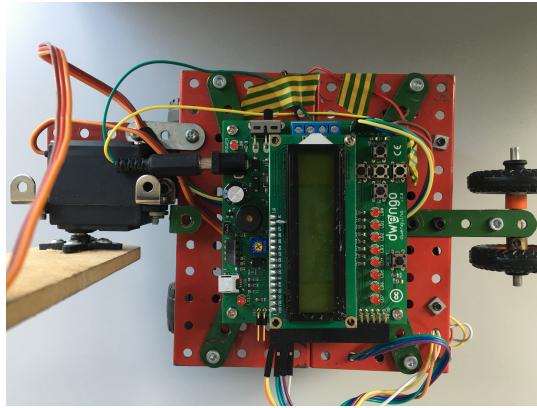
De arm, waarmee de robot zich kan verplaatsen, is vervaardigd uit MDF, een stevig hout-type. Op de arm zelf werden twee servo's geplaatst. Eén op het frame van de wagen en één in het midden van de arm.

Beide servo's kunnen 4 verschillende posities innemen, waardoor de arm zich in 16 verschillende toestanden kan bevinden. Door de correcte sequentie van bewegingen na elkaar uit te voeren, zal de robot erin slagen om heel efficiënt vooruit te gaan.

Op het uiteinde van de arm werd een rubber bevestigd. Dit met als reden een grote verscheidenheid aan oppervlakken te kunnen bewandelen. Wanneer de robot immers zou wegglijden op een gladde ondergrond, zullen de bekomen afstanden een minder goede waarde opleveren met als gevolg dat de robot zich misschien in een suboptimale oplossing zou kunnen bevinden.

3.3 Frame

Het Dwenguino-Board werd geplaatst op de bovenkant van het frame. Dit laat ons toe telkens de output weer te geven op het LCD-scherm, dat eveneens bevestigd is op de bovenzijde van het board. Het board werd georiënteerd in die hoedanigheid dat extra I/O-pinnen kunnen worden bevestigd aan de zijkant van het frame. Figuur 3.1 geeft de oriëntatie van het Dwenguino-Board weer.



Figuur 3.1: Frame van de robot met Dwenguino-Board erop geplaatst

Om de robot van stroom te voorzien, werd aan de onderkant van het frame, tussen de achterste wielen, een batterijhouder voorzien voor 6 AA-batterijen (6 batterijen van 1.5V leveren 9V, dit is genoeg om de robot te voeden).

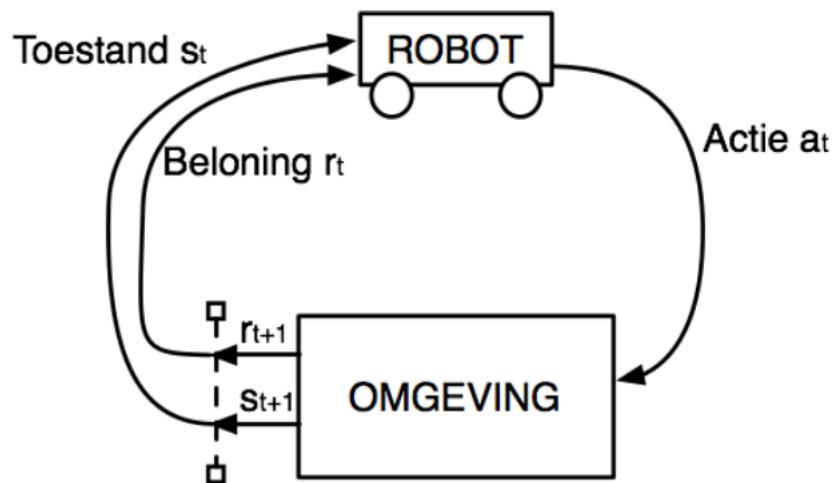
De twee wielen die achteraan op het frame zijn gemonteerd (en dus extra gewicht op de achterkant verzekeren), zorgen er tevens voor dat de robot zich optimaal kan opduwen, wanneer deze zich vooruit wil trekken.

Hoofdstuk 4

Wat is Q-learning?

4.1 theoretische achtergrond

Q-learning is een techniek, die het mogelijk maakt, om in een gegeven situatie het beste actie-plan te achterhalen en dit ook uit te voeren. In dit project wordt het begrip “beste actie-plan” ingevuld door: zo efficiënt mogelijk vooruit te komen. Het Q-learning algoritme, schematisch weergeven op figuur 4.1, bestaat eruit om vanuit de huidige toestand een actie uit te voeren, waardoor de robot in een andere toestand terechtkomt. Door de nieuwe toestand te gaan vergelijken met de vorige toestand, kan de robot beloningen verdienen of, wanneer de actie nadelig was voor het doel dat voor ogen is, een afstraffing verkrijgen. Door deze beloning/straf toe te kennen aan een actie, zal de robot kunnen opteren om telkens de actie te kiezen, die hem de hoogste beloning oplevert. Uiteindelijk zal de robot een vaste sequentie van acties blijven uitvoeren. Dit is dan de optimale oplossing voor het gegeven probleem.



Figuur 4.1: Cyclus van het Q-learning proces

De waarden die de robot gebruikt om zijn leerproces te bevorderen, zijn de zogenaamde “Q-waarden”. Deze Q-waarden staan garant voor de ‘Quality’ van een bepaalde actie en

worden bijgehouden in een tabel: de Q-tabel. De tabel is opgedeeld in 64 vakjes (16 rijen en 4 kolommen). De 4 kolommen staan voor de 2 servo's van de robot, die elk 2 verschillende bewegingen bevatten (naar boven of naar beneden). Elk vakje van de tabel vormt een toestand-actie paar (s_t, a_t) . Met andere woorden: Bij een gegeven toestand heeft de robot telkens keuze uit 4 mogelijke acties: Servo 1 naar boven of naar beneden bewegen en zelfde opties gelden voor Servo 2. De code is zo generisch mogelijk geschreven, met de bedoeling om het aantal servo's en het aantal toestanden per servo aan te passen.

4.2 De formule

De Q-waarden worden bijgewerkt telkens wanneer de robot een nieuwe zet maakt. Aan de hand van formule 4.1 wordt de vorige Q-waarden overschreven, met informatie die werd bekomen uit de recent uitgevoerde actie:

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(s_{t+1}, a)). \quad (4.1)$$

Hierin staat $Q(s_t, a_t)$ voor de Q-waarde op een gegeven tijdstip t . Daarbij staat α voor de learning-rate. De learning-rate kan een waarde aannemen tussen 0 en 1¹ en is een indicatie voor de robot in hoeverre hij “mag” bijleren (gevoelig is voor nieuwe informatie). Een learning-rate waarde van 0 betekent dat de robot niets leert, terwijl een learning-rate waarde van 1 weerlegt dat de robot enkel kijkt naar de meest recent verworven informatie.

De beloning/afstraffing die wordt verkregen, wordt vervat in r_{t+1} . Daarnaast staat $\gamma \max_a Q(s_{t+1}, a_t)$ voor de maximale beloning, van de toestand waar de robot in verkeerd, na het uitvoeren van de actie.

Met γ wordt de discount factor bedoeld. Met deze factor kan worden ingesteld hoe relevant toekomstige beloningen zijn voor de robot.

Wanneer de robot dus in een toestand terechtkomt, zal hij steeds opteren voor de actie die hem de grootste beloning oplevert. Om ervoor te zorgen dat de robot niet blijft vastzitten in een suboptimale oplossing, werd de greedy-factor ε geïntroduceerd. Deze greedy-factor zorgt ervoor dat de robot niet steeds de actie kiest die hem de grootste beloning (op het gegeven tijdstip) oplevert, maar dat ook de kans wordt gegeven aan de acties die een minder hoge beloning opleveren. Wanneer bijvoorbeeld een greedy-factor van 0.1 wordt gekozen, betekent dit concreet dat de robot in 10 procent van de gevallen niet de actie zal prefereren die hem de grootste beloning oplevert, maar willekeurig een andere actie zal uitvoeren.

¹Alle parameters krijgen een waarde tussen 0 en 1.

4.3 Algoritme

Volgende stappen moeten dus worden geïmplementeerd in het Q-learning algoritme:

1. Eerst wordt de vorige toestand, waarin de robot zich bevond, opgeslagen. Dit maakt het mogelijk om deze toestand terug op te halen wanneer dit wordt verlangd.
2. Daarna wordt gekeken in welke toestand de robot is terechtgekomen. Ook deze toestand wordt opgeslagen. Zodat beide toestanden met elkaar kunnen worden vergeleken.
3. Het toekennen van een beloning/straf gebeurt op basis van de mate, waarin de robot vooruit of achteruit bewoog, tijdens de overgang van de vorige toestand naar de huidige toestand. Het spreekt voor zich dat vooruit bewegen een positieve beloning zal opleveren en achteruit bewegen een negatieve.
4. Er moet worden berekend wat de grootst mogelijke beloning is voor de toekomstige toestand waarin de robot zich zal bevinden.
5. De Q-waarde van het vorige ‘toestand-actie’ paar moet in de Q-tabel worden overschreven met de nieuw bekomen Q-waarde.
6. Tot slot wordt een nieuwe actie gekozen die de robot zal uitvoeren. Op basis van de greedy-factor zal de robot dan opteren voor de optimale actie of een minder optimale actie.
7. De actie die werd gekozen, wordt nu ook effectief uitgevoerd.
8. Deze sequentie van acties wordt continu uitgevoerd. Op deze manier zal de robot in staat zijn om het leerproces te voltooien.

Hoofdstuk 5

Visualisatie van de Q-tabel

5.1 USART communicatie

Als uitbreiding werd ervoor gekozen om de Q-tabel grafisch te visualiseren met behulp van USART communicatietechnieken. USART staat voor 'Universal Synchronous/Asynchronous Receiver/Transmitter' en is een soort Seriële interface die geprogrammeerd kan worden met de bedoeling om synchroon/asynchroon te communiceren tussen twee apparaten.

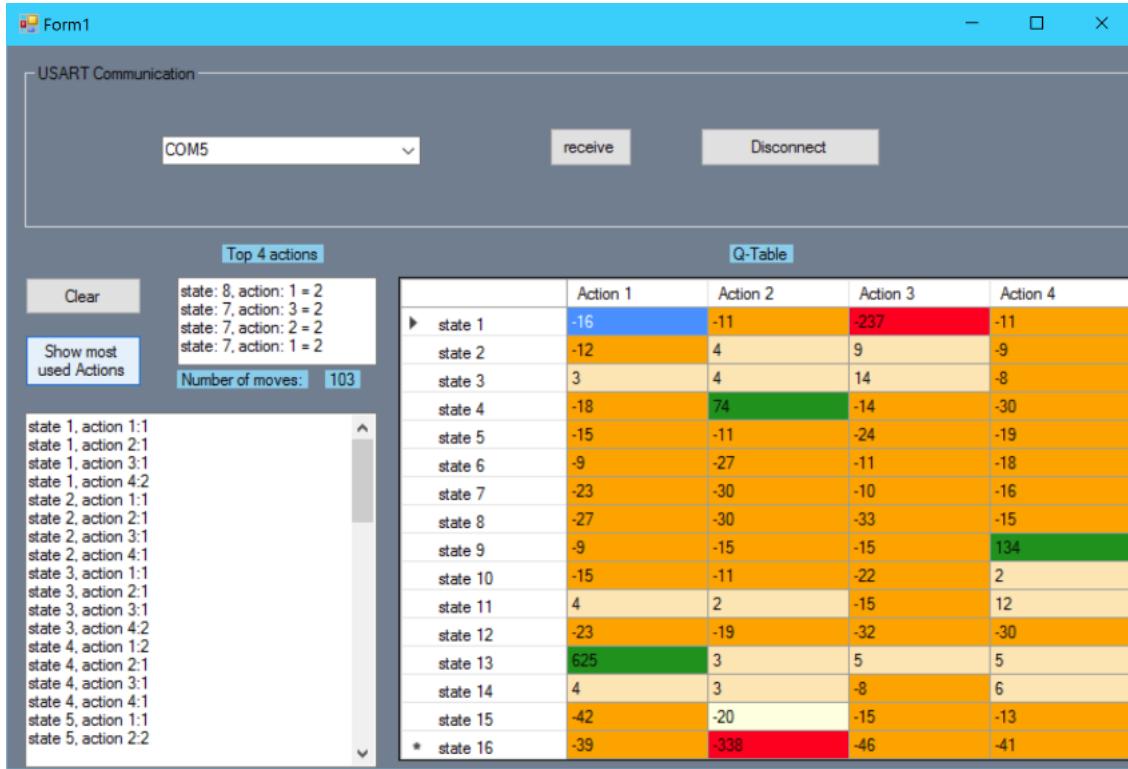
5.2 Software

Door middel van software geprogrammeerd in Microsoft Windows Visual Studio¹ worden de, door de robot verworven Q-waarden, grafisch gevisualiseerd in een tabel. Dit stelt ons in staat om nauwkeurig te volgen welke actie de robot uitvoert en hoeveer het leerproces gevorderd is. Voor vakjes waarvan de Q-waarde hoger dan 50 wordt, zullen een donker-groene kleur dragen. Hoe lager de Q-waarde wordt, des te roder het vakje van deze actie zal worden. uiteindelijk zal een patroon zichtbaar worden van groen opgelichte vakjes. Dit betekent dat de robot zijn leerproces bijna afgerond heeft en bijgevolg de optimale oplossing beet heeft.

Zoals te bemerken valt op figuur 5.1 werd naast de visualisatie van de Q-tabel tevens een output venster geplaatst om na te gaan hoeveel maal een bepaalde actie door de robot werd uitgevoerd. In dit venster zal dus ook een patroon kunnen waargenomen worden van de gerichte graaf die de robot bewandeld.

Een tweede output venster werd nog toegevoegd om een top vier van de meest gekozen acties te zien. Dit zullen na verloop van tijd de vier acties zijn, die zeker voorkomen in de optimale oplossing. Wanneer de robot aan het leren is, zal deze top vier natuurlijk continu variëren. Om het aantal acties te weten, die de robot nodig had om zijn leerproces te voltooien, werd nog label geplaatst naast de tabel met deze gegevens erin.

¹Als programmeertaal werd geopteerd om c# te gebruiken.



Figuur 5.1: Q-waarde tabel, visualisatie vanuit Microsoft Windows Visual Studio

5.3 Bluetooth communicatie

Om te communiceren tussen het dwenguino-board en de PC werd gebruikt gemaakt van een bluetooth-module, die op het dwenguino-board kan worden aangesloten. De data werd via het USART-protocol naar een USART-to-Bluetooooth module gestuurd en via de PC ontvangen. Om te weten met welk datatype we te maken hadden werd er nog een extra protocol ontworpen.

5.3.1 Protocol

Wanneer een groep data wordt doorgestuurd, zal eerst en vooral een startkarakter 's' worden doorgestuurd. Zo weet de software dat er een groep data zal worden doorgestuurd door de microcontroller (Dit is één byte lang).

Daarna wordt een typekarakter doorgestuurd. Er is keuze uit: een positie 'p', een q-waarde 'q' of de grootte van de tabel 't' (dit is één byte lang).

Hierna worden de waarden, horende bij elk van de drie types doorgestuurd. Bij een positie worden twee waarden doorgestuurd: een state (rij in de q-tabel) en een actie (kolom in de q-tabel). Bij een q-waarde wordt een enkele waarde doorgestuurd en bij een tabel het aantal rijen en kolommen. (Dit wordt allemaal doorgestuurd in twee bytes)²

Tot slot wordt een eindkarakter 'e' doorgestuurd. Zo weet de software dat de groep data helemaal werd ontvangen. Op figuur 5.2 is dit protocol mooi te zien.

```

data received
numb bytes:      1
char: s
startChar sent
data received
numb bytes:      9
data 0: 112
data 1: 15
data 2: 0
char: e
endChar sent

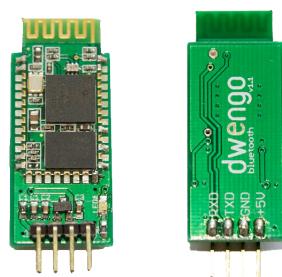
char: s
startChar sent
data 0: 113
data 1: 212
data 2: 255
char: e
endChar sent

```

Figuur 5.2: console output met protocol van de bluetoothcommunicatie

5.3.2 Bluetooth module

Om alle bluetooth communicaties te verwezenlijken werd een bluetooth module van Dwengo gebruikt (zie figuur 5.3). Deze module is gebaseerd op klasse 2 HC-06 bluetooth module en ondersteunt bluetooth v2.0. De baud rate kan softwarematig eenvoudig worden ingesteld en aangepast. In dit project werd gewerkt met een baud rate van 9600 (baud is een eenheid die het aantal signaalwisselingen (symbolen per seconde) op een datatransmissiekanaal of meer algemeen: op een informatieverbinding, aangeeft).



Figuur 5.3: Dwengo bluetooth module

²Een q-waarde wordt doorgestuurd als een 16 bit karakter, daarom zal deze waarde ook moeten door- gestuurd worden in twee bytes met behulp van Little Endian. Eerst wordt de LSB (least significant byte) daarna de MSB (most significant byte) doorgestuurd.

Hoofdstuk 6

Meetresultaten en besluit

6.1 Meetresultaten bij variërende parameterwaarden

Aan de verschillende parameters die voorkomen in de formule voor q-learning (α , γ , ε) werd oorspronkelijk een vaste waarde toegekend. Voor α (learning-rate) werd gekozen voor een waarde van 0.5, aan γ (discount-factor) werd de waarde 0.99 toegekend en voor ε (greedy-factor) werd 0.1 genomen.

Wanneer er niet voor een vaste waarde gekozen wordt voor elk van de drie parameters, kan ervoor geopteerd worden om de parameterwaarden exponentieel te laten variëren. Om na te gaan wat de invloed hiervan is op de tijdspanne waarin de optimale oplossing wordt gevonden door de robot, worden verschillende metingen uitgevoerd. Tevens wordt ook nagegaan wat de invloed zou kunnen zijn van een geïnitialiseerde Q-tabel met random waarden tussen 0 en 15 en een niet geïnitialiseerde Q-tabel.

De exponentiële functies waarmee de parameterwaarden bepaald worden, krijgen softwarematig een begin- en eindwaarde toegekend. Tabel 6.1 geeft een overzicht van deze begin- en eindwaarden.

	beginwaarde	eindwaarde
α (learning-rate)	0.90	0.50
ε (greedy-factor)	0.40	0.00
γ (discount-factor)	0.90	0.50

Tabel 6.1: Begin- en eindwaarden van de verschillende parameters

In een tweede deel van het onderzoek wordt nagegaan hoeveel acties, gemiddeld, minimaal nodig zijn om het leerproces te volmaken. Opnieuw wordt gekeken naar de invloed van exponentieel variërende parameterwaarden ten opzichte van vaste parameterwaarden en een geïnitialiseerde q-tabel ten opzichte van een niet geïnitialiseerde.

De resultaten die bekomen werden bij deze twee onderzoeken worden dan samen in telkens één tabel vervat.

6.1.1 Random geïnitialiseerde Q-tabel

Vaste parameterwaarden

Wanneer, bij aanvang van het leerproces, een vaste waarde werd toegekend aan de parameters en de Q-tabel geïnitialiseerd werd met random waarden tussen 0 en 15, werden volgende tijden en aantal acties bekomen:

	tijdsduur(m:s,cs)	aantal acties
meting 1	07:31,82	321
meting 2	05:41,40	238
meting 3	05:59,64	257
meting 4	01:47,90	78
meting 5	06:17,56	268
gemiddeld	05:19,66 ± 02:09,82	232 ± 92

Tabel 6.2: resultaten bij een geïnitialiseerde Q-tabel met vaste parameterwaarden ($\alpha = 0.5$, $\varepsilon = 0.1$, $\gamma = 0.99$)

Uit tabel 6.2 wordt gevonden dat de gemiddelde tijdsduur bij deze metingen 5 minuten, 19 seconden en 66 centiseconden bedraagt. Het gemiddelde aantal acties hierbij bedraagt respectievelijk 232. De bekomen waarden liggen hier behoorlijk ver van, maar dit is te verklaren. Wanneer men terugkijkt naar figuur 5.1, is het vakje bij 'state 13' en 'action 1' donkerblauw gekleurd. Dit is één van de belangrijkste acties in het leerproces. Indien doorheen het leerproces een andere actie in deze state eerst wordt gekozen en deze blijkt ook een positieve Q-waarde op te leveren, is het goed mogelijk dat deze "beste actie" (actie 1) maar zelden wordt gekozen. Na verloop van tijd zullen de Q-waarden van actie 2 tot en met actie 4 aanzienlijk genoeg dalen ten opzichte van actie 1 in deze state, zodat deze actie toch aan bod komt.

Exponentieel variërende parameterwaarden

In een volgende set van metingen werd, in plaats van een vaste waarde voor elk van de drie parameters, gekozen voor een exponentieel variërende functie om aan de parameters toe te kennen. Bij elke meting werd telkens de eindwaarde van de parameters opgeslagen. In tabel 6.3 staat het resultaat van de vijf metingen die werden uitgevoerd en een gemiddelde waarde over al deze metingen:

	tijdsduur(m:s,cs)	aantal acties	α	ε	γ
meting 1	02:50,71	109	0.57	0.31	0.66
meting 2	03:45,13	158	0.50	0.19	0.60
meting 3	06:49,45	298	0.42	0.04	0.52
meting 4	04:08,65	172	0.49	0.16	0.59
meting 5	04:12,33	177	0.48	0.16	0.58
gemiddeld	04:13,25 ± 01:29,03	183 ± 70	0.49 ± 0.05	0.17 ± 0.1	0.58 ± 0.05

Tabel 6.3: resultaten bij geïnitialiseerde Q-tabel met exponentieel variërende parameterwaarden

Bij een random geïnitialiseerde Q-tabel met exponentieel variërende parameterwaarden werd een gemiddelde tijdsduur van 4 minuten 13 seconden 25 miliseconden genoteerd en dit in gemiddeld 183 zetten.

Voor de parameters α (learning-rate) en γ (discount-factor) werd een exponentiële functie genomen die begint bij een relatief grote waarden en zal dalen naar ongeveer de helft van deze oorspronkelijke waarde (van 0.9 naar 0.5).

Voor de parameter ε (greedy-factor) werd een exponentiële functie genomen, die start bij 0.4 en zal dalen naar de waarde 0. Dit met de bedoeling om na een bepaald aantal acties (ongeveer 300) voortdurend dezelfde sequentie van acties uit te voeren. Na berekening bleek dat na ongeveer 300 de robot zijn leerproces voltooid had. Met dit in het achterhoofd is het dan optimaal om de 'drang' naar nieuwe informatie te reduceren naar nul, omdat dit niet meer aan de orde is.

6.1.2 Niet geïnitialiseerde Q-tabel

In een tweede reeks onderzoeken werd gebruik gemaakt van een aanvankelijk lege Q-tabel in plaats van te werken met een geïnitialiseerde Q-tabel met random waarden. Opnieuw zal zowel de invloed van vaste als exponentieel variërende parameterwaarden worden bekeken.

Vaste parameterwaarden

Vijf metingen uitgevoerd met vaste parameterwaarden en werden de bekomen resultaten voor de tijdsduur en het aantal acties hieronder in tabel 6.4 samengebracht:

	tijdsduur(m:s,cs)	aantal acties
meting 1	00:12,50	12
meting 2	09:37,81	413
meting 3	00:54,03	36
meting 4	03:30,01	154
meting 5	02:40,88	114
gemiddeld	03:15,05 ± 03:44,05	146 ± 160

Tabel 6.4: resultaten bij een niet geïnitialiseerde Q-tabel met vaste parameterwaarden ($\alpha = 0.5$, $\varepsilon = 0.1$, $\gamma = 0.99$)

Uit tabel 6.4 kan worden afgeleid dat de gemiddelde tijdsduur van het leerproces bij een niet geïnitialiseerde Q-tabel met vaste waarden gelijk is aan 3 minuten 15 seconden en 5 centiseconden. Dit met een gemiddelde van 146 acties. Wanneer deze tijden en aantal acties vergeleken worden met de bekomen waarden uit tabel 6.1 is er toch wel een aanzienlijk verschil te merken: De robot leert het snelst wanneer de Q-tabel niet random werd geïnitialiseerd.

Deze resultaten zijn te wijten aan het feit dat bij een random geïnitialiseerde Q-tabel, de kans dat een optimale actie een lagere random waarde toegewezen krijgt dan minder optimale acties, vrij groot is. Aangezien voor vaste parameterwaarden werd gekozen en de greedy-factor (ε) dus 0.99 bedroeg, zullen de kleinere waarden meestal genegeerd worden.

Exponentieel variërende parameterwaarden

Wanneer de vaste parameterwaarden nu worden vervangen door exponentieel variërende parameterwaarden, werden opnieuw vijf metingen uitgevoerd. De resultaten van deze metingen staan in tabel 6.5 weergeven:

	tijdsduur(m:s,cs)	aantal acties	α	ε	γ
meting 1	02:54,42	125	0.54	0.26	0.64
meting 2	03:23,13	150	0.51	0.20	0.62
meting 3	02:22,63	101	0.58	0.33	0.68
meting 4	01:50,34	88	0.61	0.38	0.71
meting 5	03:16,47	134	0.53	0.24	0.63
gemiddeld	02:53,40 ± 00:38.80	120 ± 25	0.55 ± 0.04	0.28 ± 0.07	0.66 ± 0.04

Tabel 6.5: resultaten bij een niet geïnitialiseerde Q-tabel met exponentieel variërende parameterwaarden

Voor exponentieel variërende parameterwaarden bij een niet geïnitialiseerde Q-tabel werd als gemiddelde waarde voor de tijdsduur van het leerproces 2 minuten 53 seconden en 40 centiseconden gevonden met gemiddeld 120 acties.

Wat uit tabel 6.3 en tabel 6.5 kan besloten worden, is dat het leerproces sneller voltooid wordt wanneer de Q-tabel niet random wordt geïnitialiseerd, wat uit de meetresultaten met vaste parameterwaarden ook al bleek.

6.2 Besluit

Wanneer de gemiddelde resultaten van alle metingen worden gebundeld tot één tabel (tabel 6.6), wordt een mooi overzicht gecreëerd om hierover enkele besluiten te trekken.

		tijdsduur (m;s,cs)	aantal acties
geïnitialiseerd	vaste waarden	05:19,66 ± 02:09,82	232 ± 92
	exponentieel variërende waarden	04:13,25 ± 01:29,03	183 ± 70
niet geïnitialiseerd	vaste waarden	03:15,05 ± 03:44,05	146 ± 160
	exponentieel variërende waarden	02:53,40 ± 00:38,80	120 ± 25

Tabel 6.6: gemiddelde resultaten over alle metingen

Uit deze meetresultaten blijkt dat een niet geïnitialiseerde Q-tabel een beter resultaat oplevert qua tijdsduur en op vlak van het aantal zetten. Dit is, zoals eerder al vermeld, te weiten aan het feit dat optimale acties een grote kans hebben om een initieel kleine Q-waarde te krijgen, zodat ze maar later in het leerproces worden ontdekt. De beste manier om het leerproces te voltooien blijkt in dit geval de niet geïnitialiseerde Q-tabel met exponentieel variërende parameterwaarden te zijn. Dit met een gemiddelde tijdsduur van 2 minuten 53 seconden en 40 centiseconden en gemiddeld 120 acties.

Uit tabel 6.6 kan ook worden afgeleid dat bij een niet geïnitialiseerde Q-tabel met exponentieel variërende parameterwaarden de resultaten het meest consistent zijn, aangezien hier de standaardafwijking het kleinste is. Dit valt ook op wanneer de tabel van deze metingen erbij gehaald wordt.

Nawoord

Doorheen dit project kregen we meer inzicht inzage "Machine Learning" en bepaalde zij-takken binnen dit onderzoeks domein. Door de invloed na te gaan van de verschillende parameters op de tijdsduur van het leerproces en het aantal acties waarbinnen dit leerproces werd bereikt, werd meer kennis verworven in het Q-learning algoritme en de verschillende parameters die hieraan verbonden zijn.

Wat ook bleek, is dat het algoritme zeer robuust is tegen verschillende verstoring van buitenaf. Tijdens het testen van de robot, die achteraf de bijnaam 'Ramon' kreeg, verloor hij tijdens één van de metingen het voorstel met de wielen. Desondanks dit toedoen slaagde de robot er toch in om zijn leerproces te voltooien, maar dan met een licht andere set van acties.

Relevante topics die niet behandeld werden:

- De Q-tabel opslaan en terug opnieuw inladen.
- Communiceren met andere robots en een gedeelde Q-tabel gebruiken¹.

Hopelijk werden onduidelijkheden in verband met deze topic opgehelderd en werd u warm gemaakt voor dit onderwerp.

ondergetekende, Arne, Jonas en Jules.

¹Een voorwaarde hiervoor is dat alle robots dezelfde omvang moeten hebben en met dezelfde voorgedefinieerde states moeten werken.

Bibliografie

- [1]S. Maessen, "Machine Learning: een korte toelichting op de techniek en toepassing", *e-sites.nl*, 2015. [Online], Beschikbaar: <https://www.e-sites.nl/blog/476-machine-learning-een-korte-toelichting-op-de-techniek-en-toepassing.html>. [geraadpleegd op 14/05/2017]
- [2]L. Frenzel, "What's The Difference Between Bit Rate And Baud Rate?", *electronicdesign.com*, 2012. [Online], Beschikbaar: <http://www.electronicdesign.com/communications/what-s-difference-between-bit-rate-and-baud-rate>. [geraadpleegd op 12/05/2017]