Introductie

Welkom! Dit boekwerkje geeft een gedetailleerd voorbeeld van een curriculum 'inleiding tot programmeren en robotica' met behulp van de AERobot.

Voordat je gaat beginnen, lees over de robot op zijn website. Er staat informatie over de hardware en software. Let in het bijzonder op mogelijk technische zaken die je tegen kan komen, met name het kallibreren van de robot.

De volgende pagina geeft een overzicht van alle lessen in het curriculum, met daarin een onderscheid tussen de kernlessen en optionele* lessen. Verderop staan alle lessen uitgewerkt. De lessen kunnen geordend of anderszins aangepast worden aan de behoeften van de klas.

Materialen en lokaal: leerlingen zullen een computer nodig hebben - idealiter één per leerling, maar delen kan ook - alsook een werkplek om hun robots op te laten rijden (een tafel of de vloer). Omdat de sensoren van de robot gevoelig zijn voor fel licht, in het bijzonder zonlicht, hebben gordijnen, luxaflex en/of gedempt licht de voorkeur. Wanneer je grotere groepen les geeft, kunnen multi-port USB laders nuttig om meerdere robots tegelijk te kunnen opladen. Sommige lessen vragen om bijzonder lesmateriaal zoals apart aangegeven.

Een cursus gebaseerd op dit curriculum was populair en effectief in proefsessies met leerlingen van 10-14 jaar in het STEM zomerkamp van i2 Camp, één van de ondersteuners van dit initiatief.

*) Optionele les
**) Aanbevolen optionele les

Discussie: Wat is een robot? *

Doel:	Nadenken over de vraag wat iets tot een robot maakt, klassieke en ongewone voorbeelden van robots, en de grenzen van robot-achtigheid
Basiskennis:	Geen
Mogelijke materialen:	Schoolbord (voor het maken van aantekeningen); computer, beamer, en internet (om video's van robots te tonen)
Tijdsduur:	30 minuten

Vraag de leerlingen wat iets precies tot een robot maakt. Maak een lijst van deze eigenschappen. Praat over een paar voorbeelden van robots - vraag de leerlingen eventueel naar hun favoriete robot - en hoe ze in de lijst passen. Bespreek wat je krijgt als bepaalde eigenschappen van de lijst zouden ontbreken, en geef een voorbeeld van zo'n machine (er is een grijs gebied tussen wel/niet een robot). Plaatjes of films van robots kunnen hier waardevol zijn (zie lijst achterin).

Onze lijst van eigenschappen van een robot, en voorbeelden van dingen die er sommige missen, zijn:

- Fysiek - het moet in de echte wereld bestaan - een vorm hebben. Software, zoals de Google webcrawler dat websites bezoekt om te indexeren, wordt soms ook een "robot" genoemd, terwijl het geen fysieke vorm heeft.
- Aandrijving - het moet uit zichzelf kunnen bewegen, of invloed kunnen uitoefenen op de 'echte wereld'. Een computer op zichzelf is geen robot.
- Zintuigen - het moet op één of andere manier input krijgen van buiten om daarop te kunnen reageren. Robots in fabrieken missen deze eigenschap vaak: ze doen hetzelfde over en over, zonder op hun omgeving te reageren.
- Autonomie - de robot moet in staat zijn zelfstandig beslissingen te kunnen nemen op basis van de input van zijn zintuigen. Sommige robots missen deze eigenschap: de 'iRobot military Packbot' is niet autonoom, maar wordt bestuurd door een mens. Zulke robots noemen we tele-operable.

Het gaat niet zozeer om de bovenstaande woorden, als wel de begrippen waar ze voor staan. Deze vier begrippen moeten in ieder geval besproken worden. In dit kader is de Roomba (een robot stofzuiger) een goed voorbeeld van de robot - omdat iedereen dit voorbeeld kent, zodat wanneer we de AERobot introduceren ze het meer zien als een Roomba dan als een Optimus Prime (transformer personage)

Er kunnen zaken genoemd worden die geen wezenlijke karakteristieken zijn voor robots, zoals:

- Er menselijk uitzien - R2D2 is net zoveel een robot als C3PO.
- Houterige bewegingen - veel moderne robots kunnen als vloeiende bewegingen maken. Goede voorbeelden hiervan zijn filmpjes van robots gemaakt door Boston Dynamics, zoals de BigDog en Petman.

AERobot: een introductie

Doel:	Leren over de AERobot en zijn input en output
Basiskennis:	Geen
Mogelijke materialen:	AERobots; hardware beschrijving (achterin)
Tijdsduur:	30 minuten

Geef elke leerling een AERobot kit en de hardware beschrijving.

Let op: Soms kan het handig zijn om de batterij alvast in de robot te plaatsen.

Laat de leerlingen hun robot in elkaar zetten.

Bespreek de eigenschappen -met bekende voorbeelden- van de robot, met behulp van de hardware beschrijving:

- Bewegen: de robot beweegt met twee telefoon-trilmotoren. Net zoals een mobieltje over de tafel beweegt op trilstand, kan de AERobot met één of twee motoren naar voren, rechts of links bewegen.
- Bots sensoren: deze infra-rood lampjes en ontvangers (zoals van de TV afstandsbediening), kunnen meten hoe ver een object van de robot af staat aan de hand van de sterkte van het reflecterende licht.
- Licht sensoren: hiermee kan je meten hoe licht het is in de kamer. een voorbeeld van zo'n licht sensor dat de leerlingen misschien kennen, is het scherm van een laptop dat zich aanpast aan hoe licht het in de kamer is.
- LED (light-emitting diode): dit lampje kan in verschillende kleuren schijnen. We zien deze terug in veel elektronische apparaten, al is het misschien nieuw dat één lampje dit in zich kan.

De klas heeft eerder gesproken wat iets tot een robot maakt (vorige les), loop met de klas door deze lijst en bespreek hoe de AERobot hieraan kan voldoen.

Minibloq: een introductie

Doel:	Bekend worden met het Minibloq programma en dit gebruiken om de robot te programmeren (zet LED aan)
Basiskennis:	AERobot: een introductie'
Materialen:	Computers met Minibloq geïnstalleerd; (bij voorkeur) computer van de leerkracht met beamer om te demonstreren
Tijdsduur:	30 minuten

Laat de leerlingen zien hoe je (1) een programma in Minibloq schrijft dat het lampje van de robot aanzet, (2) de robot aan de computer verbindt, (3) het programma in de robot laadt, (4) de robot loskoppelt en het programma afspeelt. Toon hoe de LED oplicht, en hoe je het programma kan veranderen zodat het lampje een andere kleur krijgt.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 10.ledOn, en ten slotte op ledOn.mbqc.

Geef de leerlingen een computer waar Minibloq op staat en laat ze zelf een programma schrijven dat hetzelfde doet met hun eigen robots.

Als extra activiteit, voor snelle leerlingen -terwijl je de leerlingen helpt die nog worstelen- is om hen te vertellen over wachttijden en hen te vragen om hun robots Morse berichten te laten knipperen. Je zou eventueel andere leerlingen kunnen vragen deze weer te ontcijferen.

Minibloq heeft een voorbeeld programma met LEDs en wachttijd. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 20.delay, en ten slotte op delay.mbqc.

Let op: Als de klas later ook in tekst zal leren te programmeren, kan het nuttig zijn om alvast het scherm in Minibloq met de code in C die overeenkomt met de bouwstenen zichtbaar te maken, zodat ze al vertrouwd kunnen raken met de code. Wanneer de klas blijft werken met de bouwstenen, kan het scherm beter verborgen blijven.

Rijden en afstelling van de motoren

Doel:	Leren de robot te laten rijden, en dit af te stellen
Basiskennis:	'Minibloq: een introductie'
Materialen:	Computers, robots
Tijdsduur:	30 minuten

Introductie van de motor bouwsteen binnen Minibloq.
Laat leerlingen een programma schrijven om recht vooruit te rijden.

Je zult zien dat de robot (hoogstwaarschijnlijk) niet goed vooruit zal rijden, en dat de verschillende robots allemaal (ook in de tijd) anders zullen bewegen.

Vertel over het kallibreren van de motoren (de snelheid zo instellen dat de robots bewegen zoals je wilt), laat de routine zien hoe je de robot kallibreert en laat de leerlingen het ook doen.

Herhaal dit voor de andere richtingen (achteruit, links- en rechtsaf draaien).

Opmerking 1: Waarom zijn de robots niet al goed afgesteld? Omdat de ideale snelheid van de motoren om vooruit te komen, afhankelijk is van het oppervlak waarop het rijdt. Je kunt dit zien door de robot op verschillende oppervlakten te laten rijden (tafel, vloer, papier).

Opmerking 2: Deze variatie in beweging toont het belang van feedback ("gesloten-lus" controle - het gebruik van input om te helpen keuzes te maken - in tegenstelling tot "open-lus" controle, wanneer een robot een alleen een rijtje opdrachten afwerkt). Als de robot bijvoorbeeld moeite heeft om zelfstandig rechtuit te rijden, zou het volgen van een lijn op de grond of een muur naast hem helpen als referentie.

Wachttijd bepalen

Doel:	Leren hoe de wachttijd te gebruiken
Basiskennis:	Minibloq: een introductie'
Materialen:	Computers, robots
Tijdsduur:	10 minuten

Introductie tot wachttijden in Minibloq.

Laat de leerlingen hun robots programmeren ze vooruit te laten rijden voor een bepaalde tijd om dan te stoppen.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 30.moveForward, en ten slotte op moveForward.mbqc. (Het volgende voorbeeld, 40.basiMovements, breidt dit uit naar alle elementaire bewegings-commando's.)

Kijk of ze de goede wachttijd kunnen vinden om hun robot een bepaalde afstand te kunnen laten afleggen. Probeer vervolgens de juiste wachttijd te vinden om de robot een draai van precies 90° te laten maken.

Laat de leerlingen de robot programmeren om een vierkant te maken. Zorg voor rechte hoeken en gelijke afstanden met behulp van de beweeg-bouwstenen en de wachttijd.

Door een doolhof lopen **

Doel:	De robot programmeren om een parcours af te leggen
Basiskennis:	Rijden en afstelling van de motoren', 'Wachttijd bepalen'
Materialen:	Doolhof onderdelen (getekend met stiften, gebouwd met karton of objecten); computers, robots
Tijdsduur:	1 uur

Laat de leerlingen een doolhof voor hun (of iemands anders!) robot maken. hou het simpel: niet teveel bochten, brede paden, niet te lang (de robot rijdt niet al te snel, waardoor het dan lang kan duren).

Laar de leerlingen hun robot programmeren om erdoor te rijden. Zonder feedback van de sensoren zal het lastig zijn met veel pogingen om van fouten te leren.

Opmerking: de moeite die het kost om de robot het parcours af te laten leggen illustreert het belang van feedback ("gesloten-lus" controle met gebruik van input om gedrag te sturen - als tegenhanger van "open-lus" controle, waarbij de robot een vast rijtje handelingen afwerkt). Als we met een latere les terugkomen bij dit doolhof, en de leerlingen de sensoren van de robot kunnen gebruiken, zal het veel eenvoudiger zijn.

Mensen “programmeren” **

Doel:	Leerlingen “programmeren” elkaar om door een levensgroot doolhof te lopen, om zo beter besef van de uitdagingen voor de robot te krijgen
Basiskennis:	Rijden en instellen van de motoren’, ‘Wachttijd bepalen’
Materialen:	Open ruimte; blinddoek; maskeertape
Tijdsduur:	45 minuten

Maak een (eenvoudig) doolhof op ware grootte voor de leerlingen met tape, obstakels of andere leerlingen als “muren”. Je kunt ‘voelen’ of je dichtbij iemand staat, dus het laatste is niet ideaal.

Zorg dat de ‘robot’-leerlingen het doolhof niet van tevoren zien.

Bepaal welke leerling de robot zal spelen. Laat de andere leerlingen een ‘programma’ schrijven - net zoals met de robot - met commando’s voor bewegingen (loop vooruit, draai, stop) en wachttijden.

Bepaal welke leerling het ‘programma’ gaat spelen. Blinddoek de ‘robot’ en plaats hem/haar bij het begin van het doolhof. Zorg dat ‘het programma’ de acties van de robot niet kan zien, en laat hem/haar het programma oplezen. (Het ‘programma’ kan een horloge gebruiken om en wachttijden goed door te geven.)

Laat de klas het programma verbeteren en de ‘robot’ steeds opnieuw het doolhof doorlopen tot het gelukt is.

Probeer het programma uit op verschillende ‘robots’. Het kan de eerste keren mis gaan omdat de ‘robot’ eerst nog ‘gekallibreerd’ moet worden!

De “IF” (als) voorwaarde

Doel:	Begrip krijgen van voorwaardelijke bepalingen
Basiskennis:	Wachttijd bepalen’
Materialen:	Computers, robots
Tijdsduur:	30 minuten

De acties die de robots tot nu toe uitgevoerd hebben, zijn allen van tevoren bepaald (“open-lus”), zonder de mogelijkheid van de robots om in te spelen op hun omgeving. In deze les gaan we de hier de eerste stap in maken, zodat ze op verschillende manieren kunnen reageren op verschillende situaties.

Introductie tot voorwaardelijk gedrag - als iets bepaalds gebeurt, doe dan iets als reactie daarop.

Toon de leerlingen hoe je de IF (als) bouwsteen in Minibloq gebruikt. Kies een voor de hand liggende voorwaarde (“IF 1=1”, “IF 1=5”, IF 3>2”), en als actie dat de LED aangaat (zodat je gelijk ziet of het werkt).

Leg uit hoe je met logische operatoren “AND” (en) en “OR” (of) meerdere voorwaarden kan combineren. (De logica achter de woorden ‘en’ en ‘of’ zijn hetzelfde als in onze taal, dus goed te begrijpen.)

Opmerking: Deze voorbeelden zijn voorwaardelijk, maar reageren niet daadwerkelijk op input van buiten. Later zullen we het hebben over sensoren en deze, samen met IF-stellingen, gebruiken voor echt reactief gedrag.

Botsen

Doel:	Bots sensoren leren begrijpen
Basiskennis:	De "IF" (als) voorwaarde'
Materialen:	Computers, robots; hardware beschrijving
Tijdsduur:	30 minuten

Toon, met de hardware beschrijving in de hand (of geprojecteerd door een beamer), de drie paar IR-LEDs en -sensoren aan de voorkant. Zodra een programma ze activeert, zenden de IR-LEDs IR-licht uit. Als dit IR-licht reflecteert op een object kan dit door de sensoren opgevangen worden en de sterkte gemeten worden, en dus de afstand tot de robot. Hoe dichterbij het object, des te meer IR-licht er gereflecteerd wordt.

De bots-sensor bouwsteen in Minibloq retourneert TRUE (ja) of FALSE (nee) in hoeverre het licht dat hij meet (links, rechts, of midden) sterk genoeg is om een object in de buurt te herkennen. Er is een drempelwaarde: boven deze waarde JA, onder deze waarde NEE.

Opmerking 1: Als het in de kamer heel licht is (zonlicht), zal dat de sensoren kunnen beïnvloeden, dus ook of deze bots-sensoren denken dat er een object in de buurt is.

Laat de leerlingen een programma schrijven voor de bots-sensoren. Zo zou bijvoorbeeld de LED een kleur kunnen geven als iets in de buurt is, en een andere kleur als er niets in de buurt is van de sensor is.

Gebruik dan de "OR"-bouwsteen (of) zodat het voor alle drie de sensoren tegelijk werkt: één kleur als er ergens iets staat, zo niet een andere kleur.

Opmerking 2: De "WHILE"-bouwsteen (doe zolang), meer daarover hierna, zal de ht mogelijk maken programma's te schrijven die eeuwig doorlopen in plaats van één keer. (Zonder dit commando zou het programma de LED aan kunnen zetten in een bepaalde kleur, afhankelijk of er iets in de buurt is, en in die stand blijven. Met behulp van de "WHILE"-bouwsteen, kan de robot de LED van kleur blijven veranderen als er een object weggehaald wordt of teruggezet, zonder het programma opnieuw te moeten starten.)

Opmerking 3: Er is nog een bouwsteen die gebruik maakt van deze transmitter-sensor paren als afstandsmeter. In plaats van de waarde TRUE (ja) of FALSE (nee) afhankelijk ervan of de gemeten waarde hoger of lager is dan de drempelwaarde, geeft deze de gemeten waarde terug. (Deze waarde is hoog als iets dichtbij staat en laag als iets veraf staat - niet wat je verwacht!) Als sommige leerlingen snel klaar zijn, zouden ze als extra opdracht kunnen proberen de afstands-sensor te gebruiken om de robot een groen licht te laten schijnen als er geen objecten in de buurt zijn, oranje als er iets redelijk dichtbij is, en rood als het object heel dichtbij is.

De “WHILE” (doe-zolang) lus

Doel:	Leren de “WHILE” (doe zolang) lus te begrijpen
Basiskennis:	De “IF” (als) voorwaarde’
Materialen:	Computers, robots
Tijdsduur:	20 minuten

Introductie van de “WHILE”-bouwsteen (doe zolang) in Minibloq.

De meest eenvoudige toepassing is de eeuwige lus. Tot nu toe was een programma beperkt tot een serie commando’s - doe alles achter elkaar en stop dan. De “IF”-bouwsteen (als) introduceerde de mogelijkheid van voorwaardelijk gedrag, namelijk dat verschillende onderdelen van het programma actief worden (of niet) afhankelijk van wat er gebeurt - terwijl het programma gewoon verder loopt, van begin tot eind. Vervolgens zorgt de “WHILE” lus dat dit tot in de eeuwigheid herhaald kan worden.

Laat de leerlingen een LED-lampje eeuwig in diverse kleuren knipperen met behulp van de WHILE-lus.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 50.blinkColors, en ten slotte op blinkColors.mbqc (Het volgende voorbeeld, 60.mixingColors, toont hoe je de LED elke kleur kan geven, buiten de negen opties.)

Gebruik nu de WHILE-lus met daarin een IF voorwaarde, om de robot zijn LED een kleur aan te doen als er een object in de buurt is en een andere kleur als dat niet zo is. (Dit lijkt sterk op het programmavan de vorige les, maar in dat geval moest het programma elke keer opnieuw gestart worden, terwijl deze door blijft lopen.)

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 80.bumpCenter, en ten slotte op bumpCenter.mbqc (Het volgende voorbeeld, 90.bumpAndColors, laat de LED voor elke van de drie sensoren een eigen kleur branden als er een object voor staat. Ten slotte laat 100.bumpThreshold zien hoe je de afstands-sensor bouwsteen kan gebruiken om een fijnere feedback te krijgen van de sensoren dan via de bots-sensor bouwsteen - zie Opmerking 3 van de vorige les.)

Ten slotte, schrijf een programma met een voorwaarde in een WHILE-lus die meer doet dan alleen TRUE (ja) terug geven. Een voorbeeld: zolang er geen object gedetecteerd wordt, knippert de LED rood-blauw. Zodra er een object gedetecteerd wordt, stopt de LED met knipperen.

Bots-en-draai **

Doel:	Schrijf een programma om de robot obstakels te laten ontwijken
Basiskennis:	De "WHILE (doe-zolang) lus'
Materialen:	Computers, robots
Tijdsduur:	1 uur

Nu kennen de leerlingen alle onderdelen om de robots rond te laten rijden en obstakels te ontwijken. De bots-sensor vertelt wanneer ze iets tegenkomen; de IF-voorwaarde laat ze kiezen wat te doen als ze wel of niet een object tegenkomen; en de "WHILE TRUE"-lus laat ze dit voor altijd blijven doen.

Leerlingen die dit vlot onder de knie hebben, kunnen een stapje verder gaan. Ze kunnen bijvoorbeeld de robot naar links laten uitwijken als het object rechts staat; als het object recht voor staat, stoppen of omkeren, wat ze maar willen.

Leerlingen kunnen ook speels hun robots besturen: over de tafel te leiden met hun vinger als obstakel door deze steeds zo neer te zetten dat de robot daarheen gaat waar je wilt.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 160.wallFollow1, en ten slotte op wallFollow1.mbqc (Dit voorbeeld, en 170.wallFollow2, gebruiken de afstands-sensor bouwsteen in plaats van de bots-sensor bouwsteen - zie Opmerking 3 van de Bots les.)

Volg een vinger **

Doel:	Schrijf een programma om de robot je vinger te laten volgen
Basiskennis:	De “WHILE (doe-zolang) lus’
Materialen:	Computers, robots
Tijdsduur:	45 minuten

In de bots-en-draai les moest de robot objecten ontwijken, zodat leerlingen de robot konden opjagen met hun vinger als obstakel. Wanneer we dat gedrag omdraaien, besturen we de robot door hem juist naar onze vinger te lokken.

Laat de leerlingen een programma schrijven - of als ze net het bots-en-draai programma geschreven hebben, dit te wijzigen - zodat de robot naar objecten toe rijdt: als een object vóór de robot gedetecteerd wordt, vooruit; rechts, rechtsaf; links, linksaf; en anders - sta stil.

Leerlingen kunnen hierop door bouwen: bijvoorbeeld door de LED verschillende kleuren te geven afhankelijk van de sensoren, of anders te laten reageren op de input (zodat je vinger voor de verschillende sensoren gewoon een andere manier is om de robot te vertellen wat hij moet doen: rechtsom draaien, vooruit of stoppen).

De “FOR” (voor) lus

Doel:	Pas de “FOR” (voor) lus toe, om de robot iets een paar keer achter elkaar te laten doen
Basiskennis:	De “WHILE (doe-zolang) lus’
Materialen:	Computers, robots
Tijdsduur:	30 minuten

Introductie van de “FOR” (voor) lus in Minibloq.

Laat de leerlingen een programma schrijven waar de LED kleuren knippert, 3 keer achter elkaar, en dan stopt.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 70.blink3Times, en ten slotte op blink3Times.mbqc

Ga terug naar de keer dat leerlingen de robot in een vierkant moest rijden. Herschrijf dat programma, maar in plaats van in zestien regels (‘vooruit’, ‘wacht’, ‘draai’, ‘wacht’ en dat vier keer), nu met een lus in vijf regels (de lus herhaalt herhaalt de reeks vier keer).

Maak nu een vorm met meer zijden!

Variabelen

Doel:	Begrip krijgen voor het gebruik van variabelen in een programma
Basiskennis:	De “FOR” (voor) lus
Materialen:	Computers, robots
Tijdsduur:	1 uur

Tot nu toe reageren de robots gelijk op iets dat ze tegenkomen. Maar robots kunnen ook iets onthouden wat ze gemerkt hebben om later pas te gebruiken.

Een variabele is een manier voor robots om zoiets te onthouden. Toon de leerlingen hoe je variabelen gebruikt in Minibloq, met als voorbeeld door een waarde in een variabele te stoppen die bepaalt hoe lang een LED aanstaat.

Laat de leerlingen nu zelf gedrag verzinnen met hulp van variabelen. Bijvoorbeeld:

- bots-en-draai, maar nu om-en-om naar rechts en links afbuigen;
- wissel LED kleuren af, aan de hand van hoeveel obstakels de robot gedetecteerd heeft;
- draai steeds verder, elke keer dat de robot een obstakel detecteert.

Laat de leerlingen hun leukste ideeën aan de klas tonen,

Debuggen *

Doel:	Ervaring opdoen in debuggen, met voorbeeld programma's: waarom werken ze niet als bedoeld, kan je ze goed laten werken
Basiskennis:	Variabelen
Materialen:	Computers; programma's om met debuggen te oefenen
Tijdsduur:	1 uur

Er zijn (minstens) twee soorten programmeerfouten (bugs): fouten die zorgen dat het programma niet kan draaien, en fouten die het programma (of de robot) verkeerde dingen laat doen. Minibloq zorgt met zijn blokken dat je de eerste soort fout niet kan maken. De tweede soort kunnen we helaas niet voorkomen.

Geef de leerlingen de zes debug-oefenbestanden (Ex1.mbqc—Ex6.mbqc), die te downloaden zijn met de link hierboven, en de omschrijvingen van het gewenste gedrag hieronder. Laat ze (alleen of samen) door de voorbeeld programma's gaan die niet werken zoals bedoeld. Door stap-voor-stap door het programma te lopen en te bedenken wat in verschillende situaties zal gebeuren, moeten de leerlingen in staat zijn uit te vissen wat er niet klopt en hoe het te repareren. (Vaak zal dat op verschillende manieren kunnen.)

Opmerking: De vraag zal kunnen komen waarom programmeerfouten 'bugs' heten. Er is een verhaal dat lang geleden er een mot gevonden werd in een computer die het niet deed. Dat verhaal is waargebeurd, en heeft zeker bijgedragen aan de populariteit van de naam, maar was niet de echte bron - het woord 'bug' werd al vele jaren eerder gebruikt: http://en.wikipedia.org/wiki/Software_bug#Etymology .)

Ex1.mbqc	
Gewenst gedrag:	The robot moet een vierkant maken (rond rijden).
Programma fout:	De ‘index variabele’ in de lus wordt niet elke keer hoger.
Fout gedrag:	De robot blijft rondjes rijden en stopt niet na 4 keer.
Correctie:	Laat de variabele in de lus groter worden; of gebruik een “FOR” (voor) lus in plaats van een “WHILE” (doe zolang) lus.

Ex2.mbqc	
Gewenst gedrag:	Bots-en-draai: rij vooruit; als een sensor en object ziet, draai.
Programma fout:	Er staat een AND (en) bouwsteen in plaats van OR (of).
Fout gedrag:	De robot blijft vooruit rijden totdat alle drie sensoren een object tegelijk zien, wat niet vaak gebeurt.
Correctie:	Vervang de AND-bouwsteen met een OR-bouwsteen.

Ex3.mbqc	
Gewenst gedrag:	“Frustratie”: rij vooruit tot je een object ziet, draai dan; elke volgende keer dat de robot een obstakel tegenkomt, draai een korter moment, totdat na het tiende obstakel, geef het op en stop met rijden.
Programma fout:	The programmer verwisselde de variabelen x en y op een plek.
Fout gedrag:	The robot draait elke keer juist steeds langer in plaats van korter.
Correctie:	Gebruik bij de wachttijd x in plaats van y; of verwijder de variabele y, en wacht “10-x”.
Extra lesje:	Gebruik duidelijke namen voor je variabelen! Dan is de kans kleiner dat je ze met elkaar verwart.

Ex4.mbqc	
Gewenst gedrag:	De robot moet naar het licht toe draaien.
Fout gedrag:	De robot draait weg van het licht in plaats van er naar toe.
Correctie:	Vervang de < met een > ; of verwissel de linksaf en rechtsaf draairichtingen; of verwissel de linker en rechter sensor input.

Ex5.mbqc	
Gewenst gedrag:	De robot laat de LED oplichten in een kleur die de gemiddelde afstand die de voorste afstandssensor in de tijd gemeten heeft: rood als object dichtbij waren, geel gemiddeld en groen voor ver of geen obstakels. (Hoe langer het programma draait, des te lastiger het is om de kleur te veranderen, omdat meer metingen het gemiddelde ‘gewicht’ geven.)
Programma fout:	Het programma maakt een 0/0 deel fout right vlak bij het begin, omdat het total_measured door num_samples probeert te delen voordat deze niet-nul gemaakt zijn.
Correctie:	Orden de code om eerst de meting the doen en daarna de deling en het bepalen van de LED kleur; of initialiseer total_measured als de afstandsmeting en num_samples als 1.

Ex6.mbqc	
Gewenst gedrag:	De robot zou vooruit moeten bewegen als de gemiddeld gemeten afstand van de drie sensoren geen objecten aangeeft. (Dus er wordt nu gemiddeld over afstand (3 sensoren tegelijkertijd), niet over de tijd (een sensor door de tijd heen).)
Programma fout:	In de eerste bouwsteen wordt - gebruikt in plaats van /.
Fout gedrag:	De robot stopt veel eerder dan de bedoeling is (of beweegt niet eens), omdat het “gemiddelde” veel hoger is dan men wil. Hierdoor lijken de objecten dichterbij dan ze zijn.
Correctie:	Vervang de - bouwsteen met een / bouwsteen.

Met tekst programmeren *

Doel:	Gewend raken aan het gelijk in C schrijven van eenvoudige programma's
Basiskennis:	Variabelen
Materialen:	Computers
Tijdsduur:	1.5 uur

Minibloq heeft een apart scherm dat de C code toont die hoort bij de ordening van de bouwstenen.

Laat de leerlingen een heel eenvoudig programma schrijven en kijken wat er rechts aan code getoond wordt om een beeld te krijgen van hoe een programma eruit ziet. Ze hoeven niet te begrijpen wat er staat - alleen dat er in het midden ruimte gemaakt wordt om een bepaalde code te zetten.

Laat leerlingen hun bouwstenen neerzetten en zien hoe deze er in code uitzien.

Laat leerlingen zelf C code schrijven van hun eerdere opdrachten. De makkelijkste manier om dit te doen is door een voorbeeld programma te openen: ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 210.textCoding1, en ten slotte op textCoding1.mbqc. Dit voorbeeld heeft al wat code om mee te beginnen, of alles weg te halen op de volgende vijf regels na:

```
#include <mbq.h>
void go()
{
      initBoard();
}
```

(Let op dat als je het programma opslaat, je Minibloq's voorbeeld programma overschrijft!)

Met een paar extra ingrepen, kan je een nieuw programma maken dat je gelijk C commando's laat typen in het tekstscherm. Instructies hiervoor vind je onder "Denkfouten" in de volgende les.

Tekst debuggen * (lastig!)

Doel:	Ervaring opdoen met debuggen van tekst programma's, met voorbeeld programma's in C. Waarom werken ze niet zoals gepland? Hoe kan je ze goed werkend krijgen.
Basiskennis:	Debuggen, Met tekst programmeren
Materialen:	Computers; debug oefenprogramma's
Tijdsduur:	1 uur

In de les over debuggen zagen we twee soorten bugs. Als aanvulling op de soort die fout gedrag heeft, kan een programma in C ook schrijffouten hebben, waarbij het niet werkt omdat een letterteken fout is (zoals een typefout).

Geef de leerlingen de bestanden om tekst te debuggen (bump_and_turn_ex.c, Ex7.mbqc— Ex10.mbqc), die je in bovenstaande link kan downloaden, en de beschrijvingen van het gewenste gedrag op de volgende pagina's. Laat ze werken (in groepjes of alleen) aan deze programma voorbeelden met bugs: eerst eenvoudige schrijffouten, later met denkfouten. Door de stappen van het programma te doorlopen, en te bedenken wat zal gebeuren in verschillende situaties voor de robot, zullen de leerlingen kunnen achterhalen wat er mis is, en hoe ze dat kunnen herstellen.

1. Schrijffouten

Deze oefening is gebseerd op een programma geschreven in C en bedoeld om bots-en-draai gedrag te coderen. The logica is in orde, maar er zijn een paar schrijffouten. Leerlingen kunnen een print bekijken of het programma zelf om de fouten eruit te halen, eventueel met de grafische visualisatie ernaast van eerdere programma's als hulp. De goede code (grafisch en als tekst) staat in bump_and_turn_correct.mbqc, en bump_and_turn_correct.c; Laat deze programma's nog niet aan de leerlingen zien.

De versie met de schrijffouten staat in bump_and_turn_ex.c. De fouten zijn hieronder uitgelicht:

```
void setup()
{
	initBoard();
	//Bump-and-turn
```
* de declaraties en initialisaties van de variabelen ontbreken (float left_bumper = (bumpSens(LEFT));, etc.)
```
	while(true)
	{
		mov(FORWARD);  <- 'move' mist een e op het eind
		left_bumper = (bumpSens(LEFT));
		right_bumper = (bumpSens(RIGHT));
		front_bumper = (bumpSens(CENTER));
		if((left_bumper!=0))
		{
			move(TURN_RIGHT);
			delay(1);  <- de hoofdletter D ontbreekt in Delay
		* accolade ontbreekt }
		else
		{
			if right_bumper!=0  <- haakjes ontbreken om de voorwaarde
			{
				move(TURN_LEFT)  <- punt-komma ontbreekt
				Delay(1)  <- punt-komma ontbreekt
			}
			else
			{
				if((front-bumper!=0))  <- er staat een min-teken in plaats van een liggend
streepje
				{
					move(BACK);  <- de naam van de constante is BACKWARD
					Delay(1);
				}
				else
				{
				}
			}
		}
	}
}
```

2. Denkfouten

Deze oefeningen, zoals de eerste set hiervoor, kunnen wel draaien, maar geven verkeerd gedrag. Net als de vorige oefening zie je alleen de C code. De volgende twee zijn knap lastig, omdat Mninbloq de fouten voorkomt, waardoor beginnende programmeurs moeite kunnen hebben om deze te vinden.Maar belangrijk voor C programmeurs om te leren: probeer het programma zelf te schrijven in Minibloq en vergelijk dit met de code - dat zou de leerlingen kunnen helpen bij het vinden van de fouten.

Je kunt deze programma's in C (of anderen die je zelf maakt) gelijk op de robot zetten via Minibloq - ietwat omslachtig, maar toch:

1. Maak een nieuw programma in Minibloq aan. Klik in het scherm met de grafische bouwstenen op het keuzemenu van het beginblok, waar "initBoard" staat, en klik vervolgens op "go".

2. Ga naar File -> Save All, en bewaar het programma waar je dat wil.

3. Ga naar Component -> Build. (Er zijn dan errors; maak je geen zorgen.)

4. Ga naar Component -> Open Folder, wat een nieuwe folder opent in je file manager (Windows Explorer of File Explorer) waar je het programma bewaard hebt. Je zal het programma zien (een .mbqc bestand) en een folder met dezelfde naam.

5. Open het .mbqc bestand met een tekst-programma. Op de tweede regel, staat "<files/>". Verander dit in "<files> <f name="userCode.cpp"/> </files>". Sla nu het bestand op.

6. Ga terug naar Minibloq, en open je programma via File -> Open. In het tekstveld, zal je twee tabs zien: de eerste met userCode.cpp, de tweede met (je programmanaam).cpp. Nu kan je elke code in het userCode.cpp scherm typen. Je kan ook code van andere programma's voor deze oefening erin kopie-plakken, of eigen code schrijven.

Belangrijk: als je een programma in userCode.cpp, plakt, moet je de regel "void setup()" in "void go()" veranderen; en de laatste drie regels "void loop() { }" weg halen. Anders krijg je een error als je de code wil bewaren.

Ex7.c	
Gewenst gedrag:	De robot moet een vierkant rijden, dan dezelfde (in een vierkant) weg terug, and dat alles herhalen.
Fout gedrag:	De robot rijdt bij eerste vierkant 5 zijden in plaats van 4, en hij stopt na het tweede vierkant in plaats van het weer te herhalen.
Programma fout:	Twee verschillende fouten in de lus. (Als je het verschil tussen de twee lussen weet, kan je makkelijker de fouten vinden.) In de buitenste lus loop begint de lus variabele met 1 in plaats van 0; omdat de lus loopt tot i<2, zal het maar één keer rondgaan in plaats van twee. De eerste binnen lus het een eind voorwaarde met <= in plaats van <, en zal dus een extra rondje lopen. De tweede binnen lus heeft beide fouten en loopt dus goed!
Correctie:	Zoals de derde lus toont, zijn er meer oplossingen mogelijk, maar het is het beste om alles steeds op dezelfde wijze te doen: begin met 0, en sluit af met < in de eind voorwaarde.

Ex8.c	
Gewenst gedrag:	De robot moet zijn LED rood laten branden als iets te dichtbij zijn voorste sensor is, anders staat de LED uit.
Fout gedrag:	De LED gaat juist aan als er niets staat en uit als er wel iets is.
Programma fout:	Herinner je dat de afstandssensor hoger waarden aangeeft als objecten dichterbij zijn.
Correctie:	Vervang < door >.

Ex9.c	
Gewenst gedrag:	The robot moet vooruit bewegen als er niets voor staat, en stoppen als iets te dicht bij zijn voorste afstandssensor is.
Fout gedrag:	De robot beweegt niet.
Programma fout:	Punt-komma's geven het eind aan van een stelling (een beetje als zinnen die een punt op het eind hebben). Als je de punt-komma weglaat (zoals in dit voorbeeld) laat de 'C compiler' weten dat je een schrijffout hebt gemaakt: alsof je de zin niet beëindigd hebt. Maar als je een punt-komma per ongeluk neer zet waar die niet hoort, beëindig je een stelling te vroeg en verander je diens bedoeling. (Het is zoals het verschil tussen "Ik doe het zelf" en "Ik? Doe het zelf!") Een voorwaardelijke stelling (IF of WHILE), werkt als volgt "ALS (bla bla is waar), doe dan dit." Als de punt-komma te vroeg staat, dan stopt de zin daar; je zegt niet wat de robot onder voorwaarde moet doen, en de zin na de punt-komma wordt een nieuwe opdracht zonder relatie met de zin ervoor. In dit geval, zorgt de punt-komma na WHILE() voor de volgende bewering "WHILE (TRUE), doe niets." En het programma loopt hier vast, en gaat niet door naar de volgende regel.
Correctie:	Haal de punt-komma weg na while(true).

Ex10 (MOEILIJK!)	
Gewenst gedrag:	De robot moet tellen hoe vaak een object voor zijn voorste sensor komt. Na 5 keer wordt de LED oranje. Na 10 keer wordt de LED wit.
Fout gedrag:	De LED wordt meteen wit.
Programma fout:	Er zitten eigenlijk twee verschillende fouten in, één logische en één hele stiekeme. De eerste is dat de robot te snel achter elkaar kijkt of er iets voor staat. Dus als je er iets voor zet, telt het al heel snel op. Daardoor wordt de LED als snel wit zodra je er iets voor zet. Maar er is nog een probleem. Let op het verschil tussen de twee voorwaarden om de LED aan te sturen. De eerste heeft een dubbele =; de tweede een enkele =. (Det is de reden dat we twee voorwaarden in deze opdracht, zodat een oplettende leerling het verschil kan ontdekken, al hebben ze nog niet geleerd wat het betekent.) De eerste twee onderdelen van het programma met een enkele = zijn opdrachten: je vertelt het programma de nieuwe waarde van de variabele x. Het onderdeel met de dubbele = is een vergelijking, waar je vraagt of de twee delen dezelfde waarde hebben. Je zou denken dat alles binnen een voorwaarde IF(*) herkent wordt als vergelijking, maar nee, C kijkt alleen naar hoeveel =-tekens er achter elkaar staan. Dus (x=10) is een opdracht: x wordt 10, en omdat voorwaarden zo werken, is dat vervolgens 'waar' en gaat de LED wordt wit. Je zou kunnen zeggen dat deze opdracht niet eerlijk was voor leerlingen zonder ervaring met C, maar het is belangrijk omdat deze fout heel vaak gemaakt wordt - zelfs door ervaren programmeurs, en je ziet het snel over het hoofd dus belangrijk om te weten. (En als er toch leerlingen zijn met programmeer ervaring, is dit een goede oefening voor hen - sterker, het zou nog beter zijn om hen een versie te geven zonder het middelste blok dat de LED na 5x op oranje zet, zodat ze geen hint hebben met de goede == .)
Correctie:	1) Voeg een vertraging van een seconde of zo binnen de IF bumpSens(CENTER) bouwsteen, bij de regel waarin x wordt verhoogd, zodat het object weggehaald kan worden. 2) Vervang in de laatste voorwaarde de enkele = door een dubbele = .

Volg de wand

Doel:	Schrijf een programma om de robot de rand van een groot object te laten volgen
Basiskennis:	Variabelen
Materialen:	Computers, robots; losse obstakels of ander materiaal om muren te maken
Tijdsduur:	1 uur

Bespreek met de klas of in groepjes ideeën hoe je de bots-sensor kan gebruiken om de robot om een groot object te laten rijden. (Een combinatie van vooruit rijden en draaien moet werken: rij vooruit met een afwijking naar rechts; draai naar links als de rechter bots-sensor een object herkent.)

Laat leerlingen een programma schrijven dat dit doet. Als het niet gelijk werkt, zoek uit waar de fout zit en hoe je het programma kan verbeteren.

Los een doolhof op **

Doel:	Schrijf een programma om de robot zelfstandig door een doolhof de weg te laten vinden
Basiskennis:	Volg de wand
Materialen:	Computers, robots; losse obstakels of ander materiaal om muren te maken
Tijdsduur:	1 uur

Begin met een discussie of brainstorm sessie: als je bij een doolhof zou komen, wat zou de makkelijkste manier zijn om de weg erdoor te vinden? (Denk aan ideeën als “hou je rechter hand aan de wand en volg deze”, “neem elke bocht naar rechts die je tegenkomt” of “neem een gok bij elke kruising”.)

Laat de leerlingen een programma schrijven om de weg door het doolhof te vinden, met behulp van de bots-sensoren om de wanden te zien.

Laat de leerlingen een eenvoudig doolhof voor zichzelf (of anderen) maken. (Hergebruik eventueel de doolhoven van de eerdere doolhof les!) Bekijk hoe succesvol de robots zijn: probeer te achterhalen wat de robot nog niet goed doet en het programma te verbeteren.

Licht herkenning

Doel:	Leer de werking en het gebruik van de licht sensoren.
Basiskennis:	De “IF” (als) voorwaarde
Materialen:	Computers, robots; zaklampje
Tijdsduur:	30 minuten

Als de klas de bots-les nog niet gedaan heeft, gebruik de hardware beschrijving achteraan om de drie IR-sensoren voor op de robot te tonen. Herinner ze anders eventueel op deze sensoren.

Deze sensoren kunnen we gebruiken als bots- en afstandssensoren, door de IR transmitters licht te laten stralen, en de sensoren de reflectie op objecten in de buurt ervan te meten. Maar de sensoren kunnen ook de lichtsterkte meten.

Laat leerlingen een programma schrijven met de lichtsensor bouwsteen, bijvoorbeeld door de LED aan te zetten als hij licht ziet. Test het in een donkere kamer met een zaklamp. Probeer proefondervindelijk de waarde van de lichtsterkte te kiezen voor wanneer de zaklamp dichtbij is.

Schrijf vervolgens een programma dat een LED een andere kleur geeft voor de sensor die het meeste licht meet.

Ten slotte, verander dit programma zodat de robot de kant op rijdt waar het meeste licht gemeten wordt. Leid de robot nu rond met een zaklamp.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 140.lightSensors1, en ten slotte op lightSensors1.mbqc (Dit voorbeeld, en 150.lightSensors2, zorgen dat je niet hoeft te weten welke waarde de lichtsterkte heeft, alleen welke de sensor hoogste heeft.)

Discussie: Gedrag, intelligentie en leven *

Doel:	Bespreek de overeenkomsten tussen robots en dieren, en wat intelligentie betekent.
Basiskennis:	Licht herkenning
Mogelijke materialen:	Hexbug Nanos en leefomgeving (voorbeeld)
Tijdsduur:	45 minuten

Leid een discussie met de leerlingen langs de volgende punten:

1. Wat is het verschil tussen een robot en een dier?

2. Focus eerst op zaken als voorgeprogrammeerd, stereotype gedrag versus intelligent denken en doen, en later op metalen circuits versus cellen.

3. Denk na over heel eenvoudige dieren. Een mens lijkt niet veel op een robot, net als een hond, maar een goudvis? kever? mug? amoebe?

4. Een beroemd experiment (voor het eerst gedaan door Henri Fabre in 1879) met graafwespen maakt het wat concreter. Een bepaalde soort graafwesp graaft een holletje, zoekt en vangt een krekel, legt haar eitjes in het holletje met de krekel om op te eten als ze uitkomen. Maar tussen het graven van het holletje en terugkeren met de krekel, zou iets met het holletje gebeurd kunnen zijn—een ander insect zou zich er verstopt kunnen hebben, om de krekel en eitjes op te eten. Dus als ze terugkomt met de krekel, legt ze die eerst op de rand van de ingang en controleert het holletje. Vervolgens sleept ze de krekel naar binnen, legt de eitjes, sluit het holletje af, en vertrekt. Dit lijkt slim, beredeneerd gedrag. Maar als je, terwijl ze in het holletje zit voor de inspectie, de krekel een beetje verschuift, legt ze de krekel weer op de oorspronkelijke plek, en inspecteert ze het holletje opnieuw. Verplaats je de krekel opnieuw een stukje, dan herhaalt ze het weer; dit kan tientallen keren doorgaan, tot de onderzoeker het genoeg vindt. Nu lijkt het opeens robot gedrag—als er aan een voorwaarde voldaan is, ga naar de volgende stap, zo niet ga terug naar de vorige stap!

5. Nu lijkt intelligentie gezien te worden waar het niet is. We vinden de graafwesp intelligent omdat zijn gedrag goed bedacht lijkt in zijn normale omgeving. Maar in bijzondere situaties, lijkt zijn gedrag mechanisch, een beetje 'dom'.

6. Ken je meer voorbeelden van simpel gedrag dat er toch slim uitziet, vooral bij eenvoudige dieren? Andersom, kan je het gedrag van een robot soms intelligent vinden?

7. Laat de Hexbug Nano zien - een heel eenvoudige 'robot'. Niets meer dan een trilmotor op een tandenborstel kop—het beweegt een beetje als de AERobot, maar heeft geen sensoren, reageert niet op zijn omgeving, en doet maar één ding: altijd recht vooruit rijden (als het ergens tegenaan botst, stuitert het verder). Het gaat te ver dit een robot te noemen (lees onze eerdere discussie over de vier eigenschappen van robots).

8. Bekijk een filmpje van één of meer rondrijdende hexbugs, of maak ze zelf. Het lijkt ingewikkeld gedrag, en je hebt het gevoel naar insecten te kijken, maar de zogenaamde complexiteit van hun gedrag is niets meer dan mechanische invloeden van hun omgeving.

9. Een voorbeeld waarbij een iets steeds door toeval anders reageert op dezelfde soort situatie. Een robot kan ook steeds anders reageren in dezelfde situatie door toevallige "keuzen" die het maakt, alsof hij in zijn hoofd een muntje opgooit.

10. Hoe kan je bepalen of een insect of robot zich mechanisch gedraagt, of dat er meer aan de hand is? Wat betekent intelligentie eigenlijk?

Braitenberg voertuigen *

Doel:	Eenvoudige Braitenberg voertuigen bespreken
Basiskennis:	Discussie: Gedrag, intelligentie, en leven
Materialen:	Computer, robots; schoolbord
Tijdsduur:	1 uur

Leid een discussie met hulp van tekeningen op het schoolbord langs de volgende punten:

1. Een wetenschapper genaamd Valentino Braitenberg beschreef een aantal hypothetische "voertuigen" waarbij input van sensoren op diverse manieren was gekoppeld aan de motor output.

2. Een van de eenvoudigste voertuigen heeft één wiel en één licht sensor voorop. Als het licht sterkte wordt, gaat het wiel harder. Wat gebeurt er als je een lamp voor dit voertuig zet? Het gaat steeds harder op de lamp af, tot het botst.

3. Als we het wiel juist langzamer laten draaien als de lichtsterkte toeneemt, en we zetten er een lamp voor komt hij er steeds langzamer erop af.

4. Heel eenvoudig, mar niet zo interessant. Nu iets ingewikkelder: stel er zijn 2 sensoren, linksvoor en rechtsvoor, en 2 wielen. Stel de rechter sensor is verbonden aan het rechter wiel, en de linker aan het linker, en hoe sterker het licht, des te harder draait het wiel. Wat gebeurt er als je er nu een lamp schuin voor zet? De wagen rijdt erop af, maar draait weg van het licht.

5. En als het wiel juist langzamer gaat draaien? Naar het licht draaien, maar steeds langzamer rijden!

6. Stel je voor dat de sensoren juist het andere wiel beïnvloeden. Als het licht het wiel harder laat draaien, draait het voertuig naar het licht toe en botst erop (of, met wat afstellen, dat hij er omheen draait als een planeet). Als het licht het wiel langzamer laat draaien, keert het voertuig zich af en vertraagt.

7. Als je deze voertuigen bekijkt, zou je ze hun 'karakter' kunnen zien. Hoe zou je het eerste twee-wiel wagentje willen omschrijven die wegdraait van het licht naar het donker, en hoe sterker het licht des te sneller? (Misschien verlegen, of bang?) De tweede, die naar het licht toe gaat en vertraagt? (Nieuwsgierig?) En de derde, die harder op het licht afkomt? (Agressief?)

8. En weer, al deze voertuigen zijn simpel en mechanisch: het lijkt alleen dat er meer is. Het zou kunnen zijn dat insecten ook simpele regeltjes volgen, of dat we zelf in insecten en robots intelligentie zien.

Probeer de AERobot met de licht sensoren te programmeren als één van de Braitenberg voertuigen,. (Bij de AERobots kan je niet de snelheid bepalen, de leerlingen moeten in dit geval iets anders verzinnen of de snelheid negeren.)

Lijn herkenning

Doel:	Leren over de werking en het gebruik van lijn sensoren.
Basiskennis:	De "IF" (als) voorwaarde
Materialen:	Computers, robots; hardware beschrijving; wit papier en stiften markers
Tijdsduur:	30 minuten

Toon, met de hardware beschrijving in de hand (of geprojecteerd door een beamer), de lijn sensoren op de onderkant van de AERobot: een IR-transmitter in het midden en twee IR-sensoren links en rechts. Deze kijken omlaag naar het oppervlak onder de robot.

De "lijn sensor"-bouwsteen in Minibloq geeft een waarde die aangeeft of er een zwarte lijn links, midden of rechts onder de robot is.

De lijn sensor moet gekallibreerd worden om goed te werken op een bepaald oppervlak bij bepaald omgevingslicht, net zoals de motoren gekallibreerd moeten worden. Demonstreer de kallibreer-routine met een 1cm dikke lijn van een zwarte stift op wit papier en laat de leerlingen dit nadoen.

Laat leerlingen een programma schrijven waarin de robots LED een kleur geeft afhankelijk van waar hij de lijn detecteert.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 180.lineAndColors, en ten slotte op lineAndColors.mbqc.

Lijn volgen

Doel:	Programmeer de robot om een zwarte lijn op de grond te volgen.
Basiskennis:	Lijn herkenning
Materialen:	Computer, robots; wit papier en zwarte stiften
Tijdsduur:	1 uur

Laat leerlingen met papier en stiften een dikke zwarte lijn trekken onder de robot. De lijn moet minstens zo dik zijn als de USB-poort, en slingeren (recht is saai) maar niet té veel (dan raakt de robot de weg kwijt).

Bespreek met de klas hoe je een programma zou maken om de robot de lijn te volgen. Hoe zou het gedrag van de robot eruit moeten zien? (Ons antwoord: als de sensor vindt dat de lijn midden-onder is, rechtdoor rijden. Als de lijn links ligt, draai naar links. Als de lijn rechts ligt, draai naar rechts.)

Laat de leerlingen dit gedrag programmeren.

Opmerking: het kan helpen om de robots beweging te kallibreren zodat hij bij "linksom" en "rechtsom" ook vooruit gaat en niet op zijn plaats keert.

Leerlingen die tijd over hebben, zouden erop door kunnen bouwen. Bijvoorbeeld dat de robot stopt en met zijn LED knippert als hij de weg kwijt is, zodat je hem dan kan oppakken en weer op de lijn kan zetten.

Minibloq heeft een voorbeeld programma voor deze les. Ga naar het FILE menu en kies EXAMPLES: dubbelklik dan op de AERobot folder in het scherm dat zich opent, vervolgens op 190.lineFollower, en ten slotte op lineFollower.mbqc.

Multi-robot interactie *

Doel:	Ontdek manieren om verschillende AERobots met elkaar te laten communiceren
Basiskennis:	Hoe meer, hoe beter!
Materialen:	Computers, robots
Tijdsduur:	1 uur

Begin de les met een discussie over hoe - in het algemeen - dieren nooit helemaal geïsoleerd zijn, maar altijd wel met soortgenoten inter-acteren. (Zelfs solitaire diersoorten communiceren met elkaar, al is het maar om te vertellen wat hun gebied is.) Hoe kunnen onze robots inter-acteren?

Sommige ideeën om mee te beginnen:

1. Zouden we de bots of afstand-sensoren kunnen gebruiken om andere robots te herkennen, door ze als obstakel te zien?

2. Denk eraan dat deze sensoren werken door IR-licht uit te zenden en de sterkte van de reflectie op een object te meten. Dus één robot zou IR-licht, met zijn bots of afstand-sensoren, uitzenden dat een ander kan opvangen.

3. De LED schijnt licht, wat een licht-sensor kan opvangen. Kan één robot in een donkere kamer met zijn LED licht schijnen en rond lopen, en een andere robot hem volgen met zijn licht-sensoren? Zou je zo een polonaise van robots kunnen maken?

Als de leerlingen leuke ideeën realiseren zouden we dat dolgraag van jullie willen horen!

Versier je robot **

Doel:	Versier de robot zoals je zelf wil
Basiskennis:	Geen
Materialen:	Robots; knutselspullen (zie hieronder)
Tijdsduur:	1 uur

In deze les kunnen de leerlingen hun robot versieren zoals ze zelf willen! Ze kunnen de robot op een machine, insect, voertuig, of wat dan ook laten lijken.

We raden het gebruik van stickervellen aan: laat de leerlingen de vorm van de printplaat overtrekken op het papier en uitknippen: versier het stickervel in plaats van de printplaat zelf. Zo voorkom je lijm op de printplaat wat problemen zou geven: het is zo ook makkelijker designs te vervangen en robots onderling te delen.

Naast stickervel en schaar, kan je gebruik maken van plakoogjes, pijpenraggers, stiften, glittertjes, veren, kralen, crêpe-papier etc.

Opmerking 1: Dit was de populairste les toen we deze cursus voor het eerste testten, met unanieme stem.

Opmerking 2: Leerlingen zullen weten (of ontdekken) dat hoe ze hun robot zullen versieren effect kan hebben op het gedrag. Ze moeten weten waar de sensoren en motoren zitten, en dat deze misschien last van de versieringen hebben. Teveel gewicht, of onevenredig veel materiaal aan één kant kan zorgen dat de robot vreemd beweegt (langzamer, moeilijker draaien, vastlopen, etc.)

Schrijf je eigen programma **

Doel:	Zelf ontdekken van programmeerbaar robot gedrag
Basiskennis:	Hoe meer, hoe beter!
Materialen:	Computers, robots
Tijdsduur:	1 uur

Nu hebben de leerlingen genoeg geleerd over programmeren, gedrag hardware en intelligentie. En nu is de kans om zelf iets te verzinnen en programmeren om de robots te laten doen.

Deze les kan gevolgd worden met een presentatie van de resultaten of een discussie over hoe het ging, met antwoorden op vragen als:

- Wat ging er goed?
- Wat ging er mis?
- Hoe voelde je je toen de dingen niet zo goed liepen als je had gehoopt?
- Waren er ook problemen die je aan voelde komen?
- Zo ja, waarom kwam je ze toch tegen?
- Ben je ook iets onverwachts tegengekomen?
- Wat was het leukste aan samenwerken in een groep of met anderen?
- Wat was het moeilijkste?
- Wat was het meest uitdagende van de robot om te programmeren?

Diverse video's

- Robots in een autofabriek
- iRobot PackBot (reclamefilmpje)
- iRobot Roomba toont hoe hij in een kleine kamer objecten ontwijkt

Natuurlijke, vloeiende bewegingen

- Boston Dynamics BigDog
- BigDog parodie
- Boston Dynamics PETMAN (nog een link)

Verschillende manieren om vooruit te komen

De filmpjes hierboven tonen lopen (BigDog, PETMAN) rijden (Roomba) of stappen (PackBot)

- Iets tussen lopen en rollen: RHex
- Meer van hetzelfde, meer als wielen: "whegs"
- Klimmen: Stickybot
- Springen: Zandvlo
- Slangbewegingen
- Zwemmen
- Vliegen
- Glijden (net als de AERobots)

…en veel meer!

Robots geïnspireerd door insecten

- Termieten (gebouwen)
- Mieren (dragen samen grote objecten)
- Vliegen/bijen (vliegen)

…en veel meer!

De AERobot hardware

A. Licht/afstand/bots sensoren (3).
   Deze foto-transistors kunnen licht detecteren en zijn naar buiten gericht, op hun omgeving.
   - Je kan ze gebruiken om de lichtsterkte te meten van het licht dat ze opvangen - als lichtsensor.
   - Je kan ze gebruiken om de sterkte van infra-rood licht te meten (van lampje B, gereflecteerd door een object in de buurt van de robot) - als afstandsmeter.
   - Je kan ze ook gebruiken om te meten of een object dichterbij de robot is dan een bepaalde afstand - als bots sensors.

B. Infra-rood lampjes (3).
   Naar buiten gericht, op de omgeving. Infra-rood licht dat ze uitzenden wordt door objecten in de buurt gereflecteerd en kan gedetecteerd worden door de licht sensoren A. Hoe dichterbij het object, des te sterker het gereflecteerde licht.

C. Lijn sensoren (2).
   Deze foto-transistoren zijn omlaag gericht, op het tafelblad, en meten de lichtsterkte van D dat door het grondoppervlak gereflecteerd wordt.

D. Infra-rood lampje (1).
   Omlaag gericht, op het tafelblad.

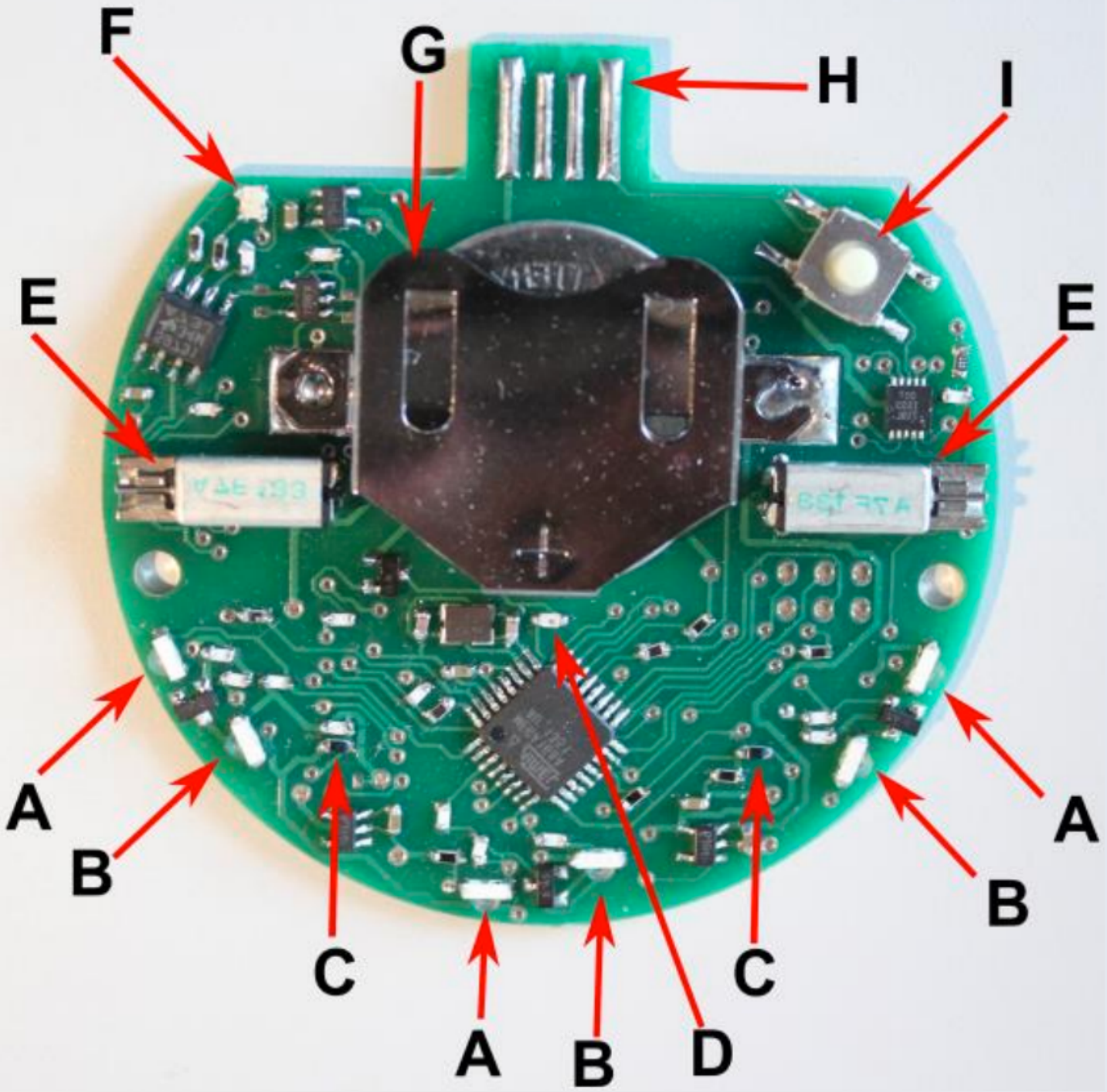
E. Trilmotoren (2).

F. Light-emitting diode (LED).
   Deze lamp kan elke kleur hebben.

G. Batterij (lithium-ion, oplaadbaar).

H. USB-poort.

I. Aan/uit knop.


F
G
H
I
E
E
A
B
C
A
B
D
C
A
B