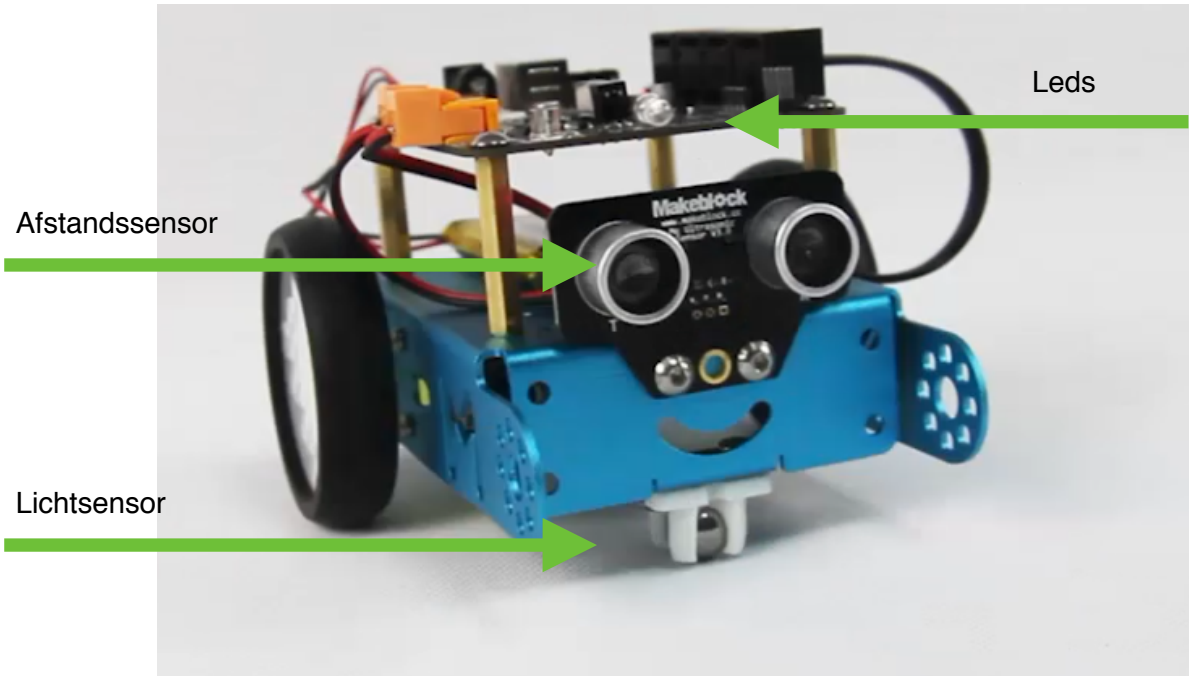
Opgave Project
Functioneel Programmeren

Achtergrond

In dit project zal je zelf een *mini programmeertaal* maken voor het programmeren van een robot. Deze mini taal zal je zo uitwerken dat de robot enkele leuke taken kan uitvoeren zoals het volgen van een lijn en het uit de weggaan van obstakels. Om te valideren of je programmeertaal goed werkt zal je drie verschillende programma's maken in je eigen programmeertaal. Ten eerste een programma om van de robot een brandweerwagen of politiewagen te maken door *de leds van de robot te doen blinken*. Ten tweede een programma die de robot *een lijn laat volgen* en ten derde een programma die de robot *obstakels uit de weg laat gaan*.

Robot

De robot die je zal gebruiken is een didactische robot die ook gebruikt wordt om kinderen de basis beginsels van programmeren aan te leren. We gaan deze robot echter niet programmeren in scratch maar in Haskell. Om dat te doen staat er op minerva een hardware bibliotheek ter beschikking. De robot heeft twee motoren die je toelaten om de wielen van de robot individueel te laten bewegen. Verder heeft de robot ook twee leds bovenaan op de printplaat. Deze leds kan je elke RBG waarde laten aannemen. De robot heeft ook verschillende sensoren waarvan we voor dit project twee zullen gebruiken. Een afstand sensor en een lichtsensor. De afstandssensor laat toe om na te gaan of er een voorwerp voor de robot staat. De lichtsensor is naar de grond gericht en geeft de gebruiker de mogelijkheid om na te gaan of de grond net onder de robot wit of zwart is.



Functionele vereisten

Het project bestaat eruit om een mini programmeertaal te maken om zo een robot te programmeren. Je bent vrij om zelf de syntax van je programmeertaal te bepalen en je mag deze maken in het Engels of in het Nederlands (maar wees wel consistent). Zoals reeds in de inleiding vermeld zal je verschillende instructies moeten maken zodat je op zijn minst de drie gevraagde programma's kan implementeren in jou programmeertaal.

We verwachten dus op zijn allerminst de volgende functionaliteit voor je programmeertaal:

1. *Variabelen*: In je taaltje zal het mogelijk zijn om de uitgelezen waardes van de sensoren bij te houden.
2. *Getallen*: Het uitlezen van de afstandssensor zal natuurlijk een getal als waarde hebben. Je zal dus in je taaltje getallen moeten ondersteunen.
3. *Booleans*: Om te beslissen of je naar links of rechts gaat moet je taaltje kunnen omspringen met booleans.
4. *Operatoren*: Je zal allerhande primitieve operatoren moeten voorzien, bijvoorbeeld (>,<,==, +, -, *).
5. *Loops* : De programmeertaal zal moeten toelaten om instructies in een loop te herhalen.
6. *Conditionals* : Je zal op zijn minst een if test moeten toevoegen zodat je kan beslissen dat de robot andere acties uitvoert afhankelijk van de staat van de sensoren.
7. *Sturen van de Motors*: Je zal de motors moeten kunnen controleren. Je mag zelf kiezen of je dat doet op een statische manier bijvoorbeeld: TURN_LEFT, TURN_RIGHT, FORWARD of meer generisch (Turn LEFT) (Turn RIGHT).
8. *Uitlezen sensoren:* De sensoren van de robot moeten kunnen uitgelezen worden. Voor de lichtsensor zal je een boolean of een "richting" moeten teruggeven. Voor de afstand zal je een getal moeten teruggeven die de afstand uitdrukt.
9. *Sturen van de leds*: De leds kan je programmeren in de verschillende kleuren. Je zal dus primitieven moeten voorzien om de leds een kleur te geven.
10. *Commentaar :* Zorg ervoor dat de programmeur code commentaar kan toevoegen in zijn code.

Rapportering

Naast je implementatie verwachten we ook een verslag met de volgende structuur:

1. *Inleiding* : In je inleiding geef je een overzicht van je project wat je verwezenlijk hebt. Geef ook aan op welke bestaande programmeertalen je eigen programmeertaal gebaseerd is.
2. *Syntax van de taal:* Geeft een overzicht van de constructies in je taal in (informele) BNF vorm.
3. *Semantiek van de taal*: Voor elk van je taalconstructies geef een korte uitleg wat de taalconstructies doen en hoe je deze gebruikt.
4. *Voorbeelden programma's:* Geef volledige uitleg bij de programma's die je geïmplementeerd hebt in je eigen programmeertaal.
5. *Implementatie*: Geef een overzicht van de belangrijke punten van de implementatie. Refereer naar de lijnnummers in je code. Kleine stukjes code die heel belangrijk zijn kan je ook inline in je rapport plaatsen. Het is echter niet de bedoeling dat je verslag een kopie van je broncode is.
6. Conclusie : Geef een overzicht van wat je gerealiseerd hebt en hoe je de bestaande code eventueel nog zou kunnen verbeteren.
7. *Appendix Broncode*: Geef de volledige code van je project, zorg ervoor dat hierbij lijnnummers staan zodat je hier makkelijk naar kan refereren.

Niet Functionele vereisten

Naast de functionele vereisten zijn er ook een aantal niet functionele vereisten die dienen om na te gaan of je de concepten tijdens de hoorcolleges goed begrepen hebt. Hoewel deze niet functionele vereisten normaal gezien zullen opduiken tijdens de implementatie van het project lijsten we deze hier nog eens expliciet op:

Gebruik van parser monad: Om je taal te implementeren zal je beginnen van een tekst file, deze tekstfile zal je moeten parsen om zo de taal te kunnen uitvoeren. Voor het implementeren van deze parser verwachten we dat je gebruik zal maken van de parser monad. **Bestaande parser bibliotheken mogen enkel gebruikt worden als inspiratie voor je eigen bibliotheek.**

Gebruik van monad transformers: Je zal tijdens de implementatie van je project zowel IO als State moeten gebruiken. Daarom verwachten we dan ook dat je gebruik zal maken van de monad transformer bibliotheek.

Code Kwaliteit: Gebruik beschikbare tools om je code op te kuisen, gebruik “**hlint**” om de meest gebruikelijke bad smells uit je code te halen.

Commentaar: Schrijf voldoende commentaar bij je code.