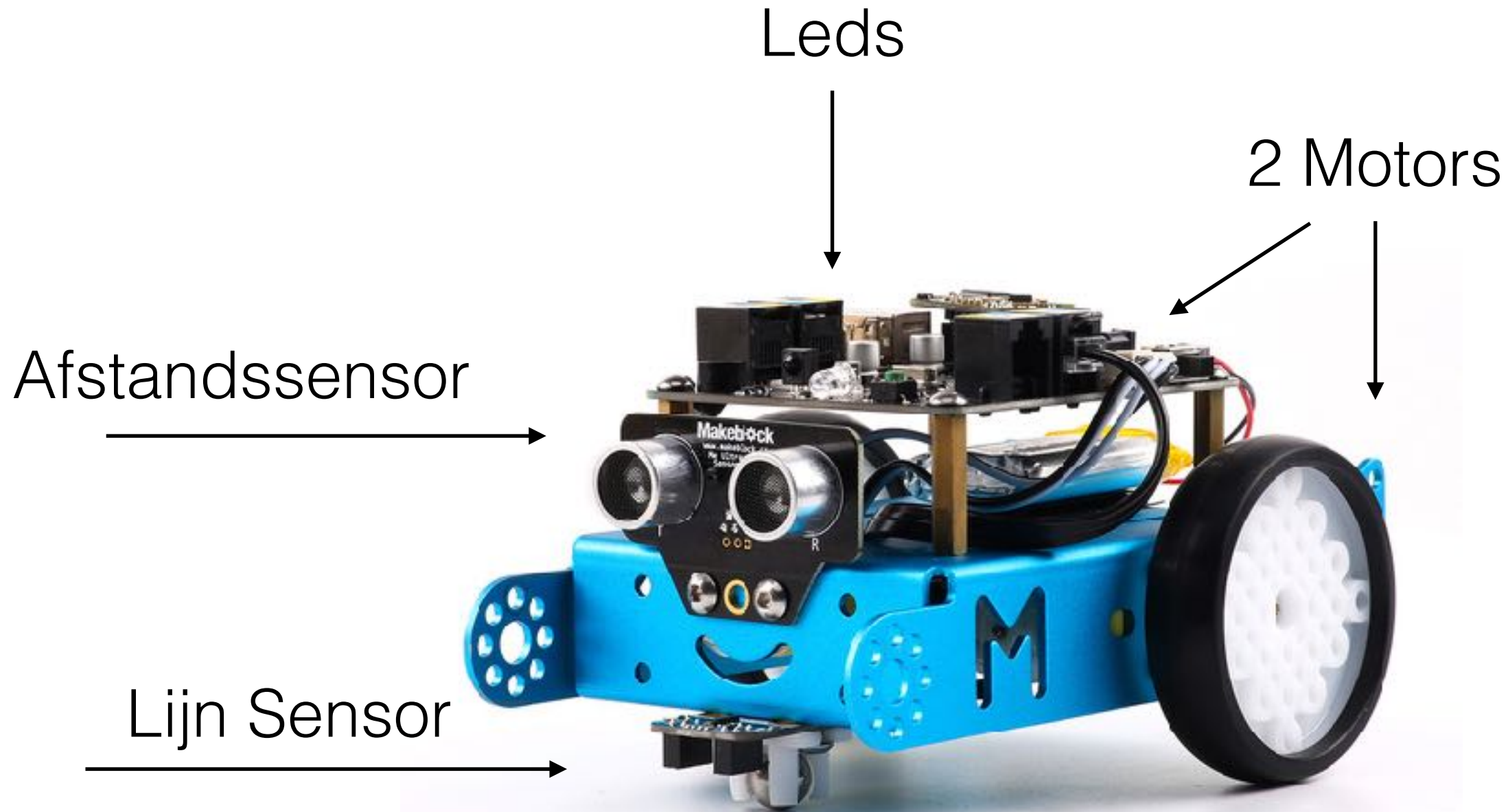
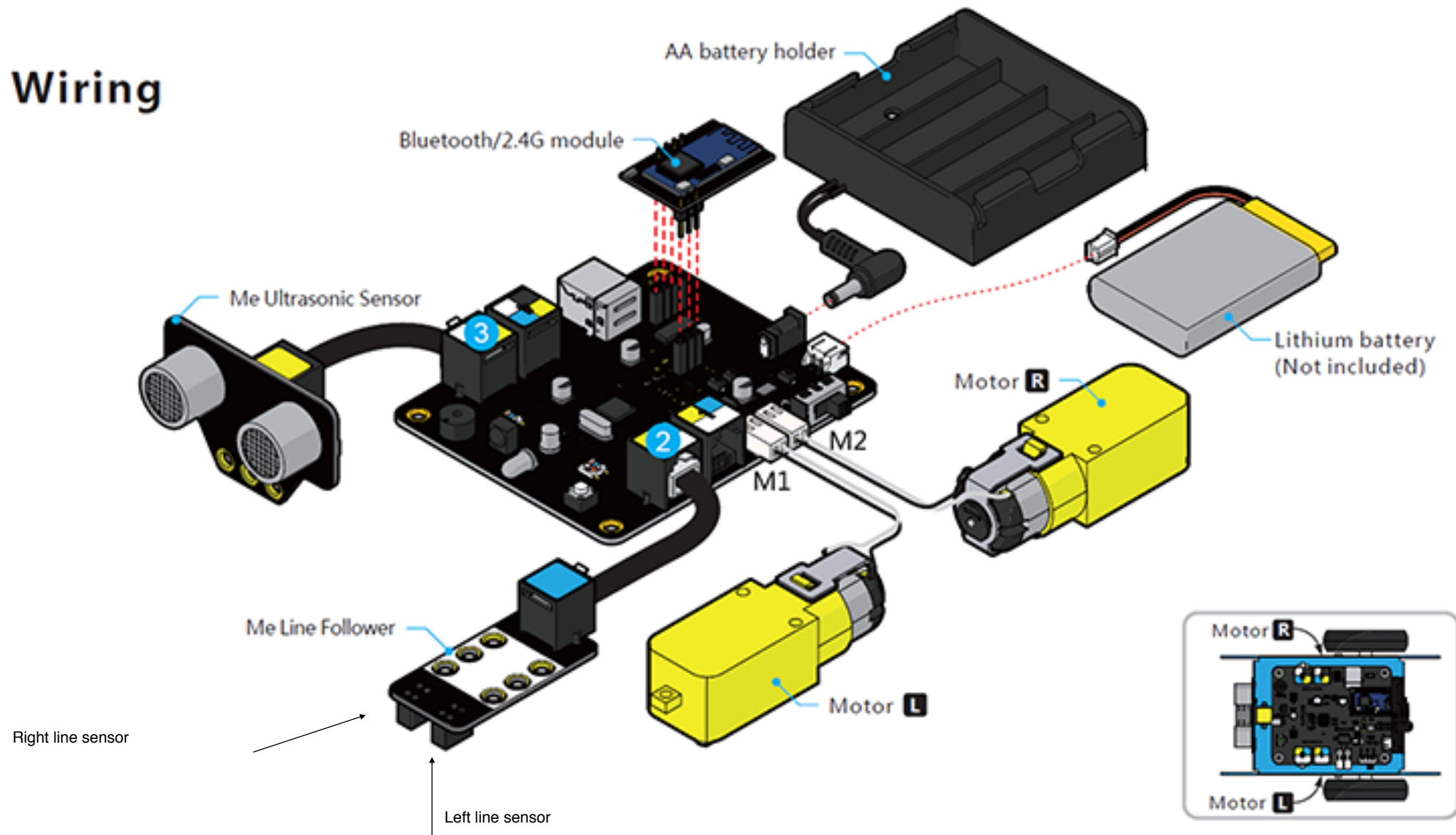


# Functioneel Programmeren Project



# De robot

## Wiring



# Installeren van de bibliotheek

`Cabal install mbot`

## Windows

(na installatie)

```
ghc Minimal.hs
```

## Linux

Het zou kunnen dat hidapi niet geïnstalleerd is.

Dat kan je installeren met “**apt-get install hidapi**” of de package manager van jouw linux systeem.

Om de code te gebruiken als normale gebruiker kan je een file

“/etc/udev/rules.d/99-hidraw.rules” aanmaken met de volgende lijn:

```
KERNEL=="hidraw*", ATTRS{busnum}=="1", ATTRS{idVendor}=="0416" ATTRS{idProduct}=="ffff", MODE="0666"
```

Nadien kan je de code compileren en uitvoeren zoals normaal:

```
ghc Minimal.hs
```

## Osx

Op mac moeten we expliciet meegeven welke frameworks er gebruikt worden.

In ons geval is dat IOKit en CoreFoundation.

```
ghc Minimal.hs -framework IOKit -framework CoreFoundation
```

# Problemen

 **Minerva**  [Bekijken als gewone gebruiker](#) | [Aangemeld als Christophe Scholliers](#) | [UGent CAS logout](#)


[Mijn Minerva](#) [Mijn profiel](#) [Mijn agenda](#) [Aanbod 2016-2017](#) [Aanbod 2015-2016](#) [Aanbod 2014-2015](#) [Aanbod 2013-2014](#) [Aanbod 2012-2013](#)

[Functioneel programm...](#) > [Forum](#)      

## Functioneel programmeren

[+ Inleiding toevoegen](#)  
[+ Forumcategorie maken](#) [+ Nieuw forum toevoegen](#)  [Zoeken](#)

### Algemene categorie

	Forum	Onderwerpen	Berichten	Laatste bericht	Acties
	Algemeen Forum	0	0		     
	MBot Software Problemen	0	0		     

### MBot

Forum	Onderwerpen	Berichten	Laatste bericht	Acties
Geen gegevens weer te geven				

# Voorbeeld programma

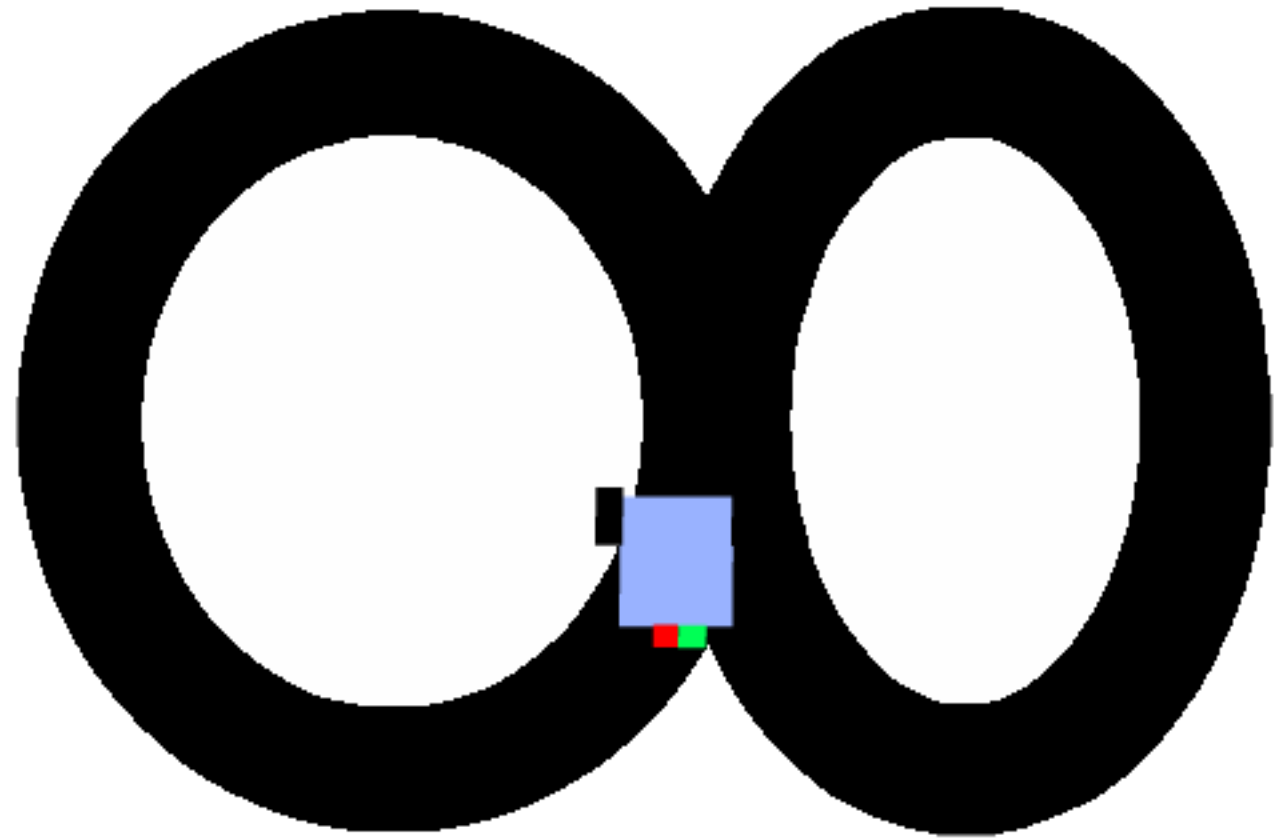
```
import MBot

main = do
  d <- openMBot
  sendCommand d $ setRGB 1 0 0 100
  sendCommand d $ setRGB 2 100 0 0
  closeMBot d
```

# Twée delen

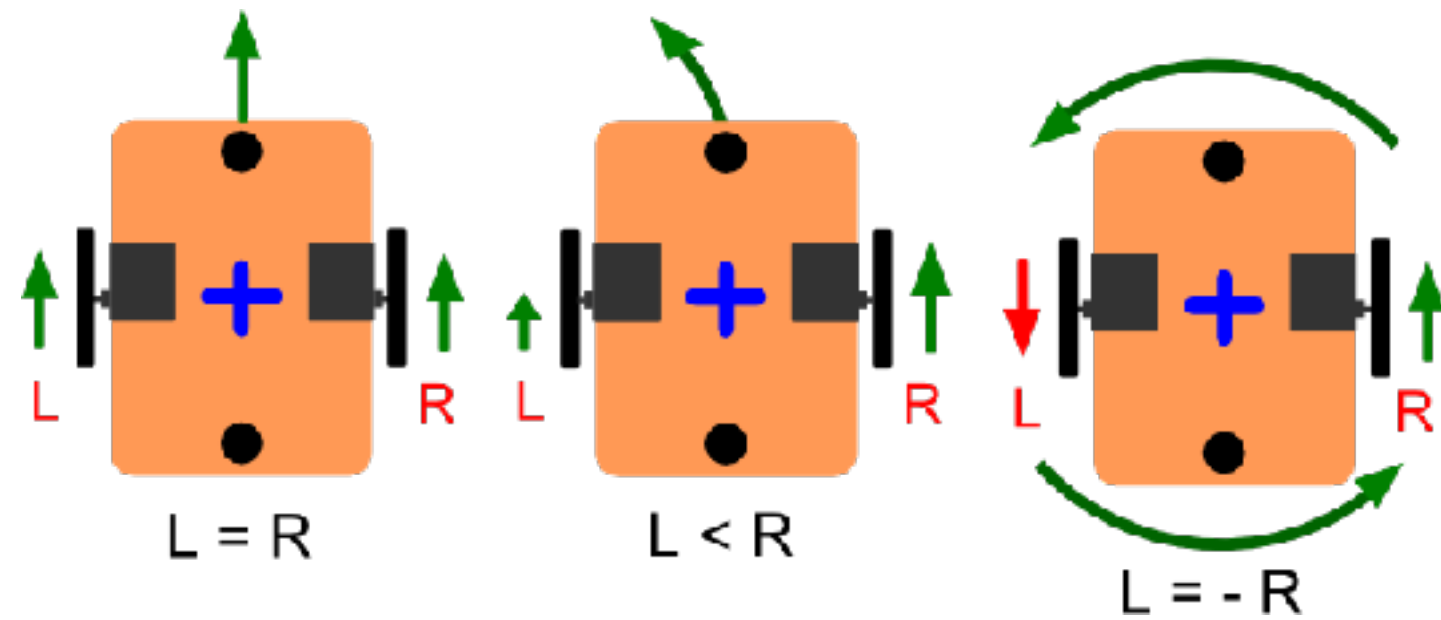


# Maak je **eigen** taal

[illegible]

# Maak een simulator

# Simulator



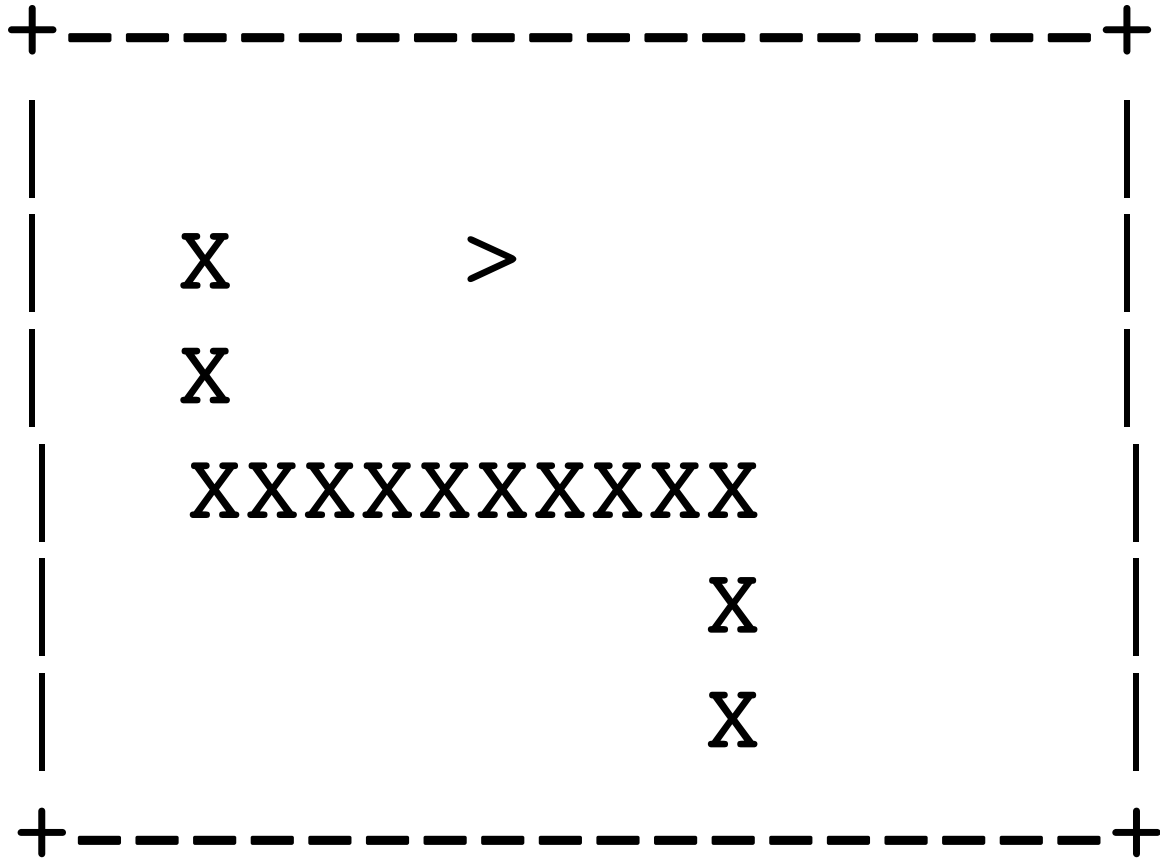
$$\Delta x = \Delta t \frac{W_r}{2} (v_L + v_R) \cos \theta$$

$$\Delta y = \Delta t \frac{W_r}{2} (v_L + v_R) \sin \theta$$

$$\Delta \theta = \Delta t \frac{W_s}{W_a} (v_R - v_L)$$



# Simulator



( 0 , 0 )    ( 10 , 10 )  
( 10 , 10 )    ( 20 , 10 )

Gridwereld formaat	
+	Hoek van de wereld
-	Horizontale grens van de wereld
	Verticale grens van de wereld
< > ^ v	Plaats en richting van de robot
X	Muur
(x,y)	Coördinaat voor het voorstellen van een lijnstuk.

# Maak je **eigen** taal

BASIC 5.1 gI

Copyright (C) 1984 by Edward T. Grochowski

```
>load "PATTERN15A"
```

```
>list 1-30
```

```

1 ! Pattern 15a
2 ! by Edward T. Grochowski
3 ! Revised 7/28/83, 3/9/85
4 ! 750 bytes reserved
5 !
6 CLS
7 PRINT LINE(0);TAB(0);"Pattern 15
8 PRINT LINE(1);TAB(0);"by Edward
9 N=5 ! Number of sides in
10 K=50 ! Number of polygons
11 Q=255.5 ! Center in X direct
12 W=191.5 ! Center in Y direct
13 MR=180 ! Maximum radius
14 FOR G=0 TO K
15     T=G/K
16     R=G/K*MR
17     FOR H=0 TO N-1
18         TH=(H/N*2+T-0.5)*PI
19         X=INT(COS(TH)*R+Q)
20         Y=INT((0-SIN(TH))*R+W)
21         IF H>0
22             THEN CONNECT (I,J) TO (X,Y)
23             ELSE A=X
24                 B=Y
25             ENDF
26         I=X
27         J=Y
28     NEXT H
29     CONNECT (X,Y) TO (A,B)
30 NEXT G

```

```

10 INPUT "What is your name: "; U$
20 PRINT "Hello "; U$
25 REM
30 INPUT "How many stars do you want
35 S$ = ""
40 FOR I = 1 TO N
50 S$ = S$ + "*"
55 NEXT I
60 PRINT S$
65 REM
70 INPUT "Do you want more stars? ";
80 IF LEN(A$) = 0 THEN GOTO 70
90 A$ = LEFT$(A$, 1)
100 IF (A$ = "Y") OR (A$ = "y") THEN GOTO 30
110 PRINT "Goodbye ";
120 FOR I = 1 TO 200
130 PRINT U$; " ";
140 NEXT I
150 PRINT

```

This class will use the ghost to insert a collection of particles which have no net charge. This is used to calculate chemical potential and activity coefficients.

```

class widom : public analysis {
private:
    average<double> expsum; //!< Average of
protected:
    ghostin;                //!< count test
    cnt;                    //!< List of gh

```

```
private:
    average<double> av;
protected:
    int ghostin;
    long long int cnt;
    vector<particle> g;
public:
    sim(int n=10);
    ~sim();
```

```
public:
    widom(int n=10);
    string info();
    void add(particle);
    void add(container &);
    void insert(container &, energybase &);
    void check(checkValue &);
    void sum() { return exp(muex()); }
    void sum() { return -log(expsum.ans); }
```

```

void insert(int value) {
    if (isFull()) {
        return;
    }
    for (int innerIndex = 0; innerIndex < outerIndex; innerIndex++) {
        if (data[innerIndex].Comp
            // found value
            largest = data[innerIndex];
            indexLargest = innerIndex;
        }
    }
}

```

```
// Find Value  
largest = data[indexLargest];  
  
}  
  
/* Interchange data[indexLargest]  
   data[outerIndex]; */  
temp = data[indexLargest];  
data[indexLargest] = data[outerIndex];  
data[outerIndex] = temp;
```

```

    temp = data[indexLongest]
    data[indexLongest] = data[
        indexShortest]
    data[indexShortest] = temp
}

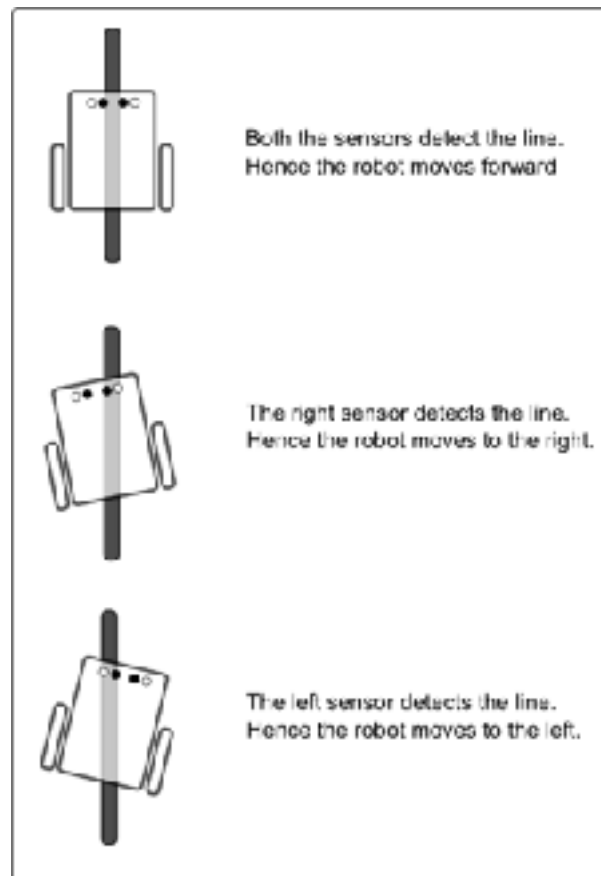
}

```

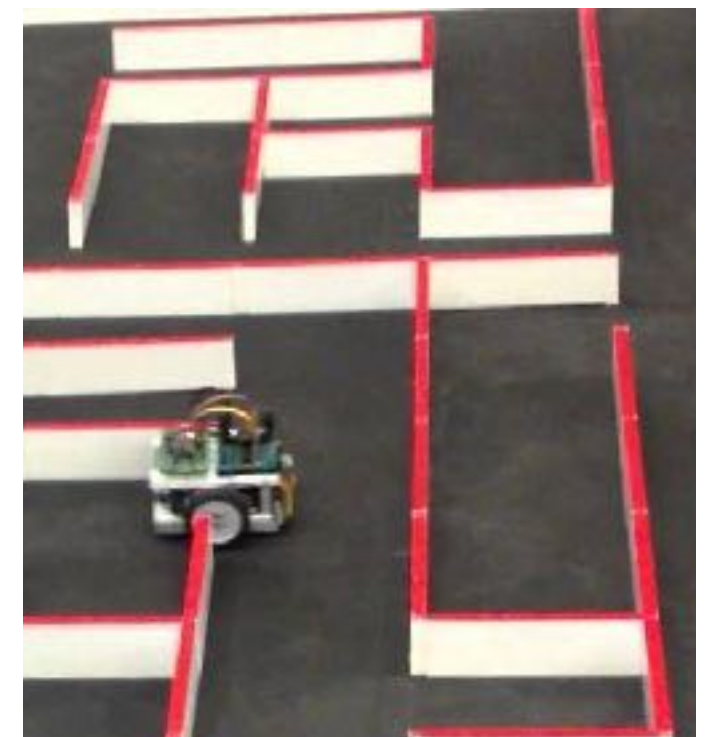
# Implementeer in je eigen taal



Blinking leds

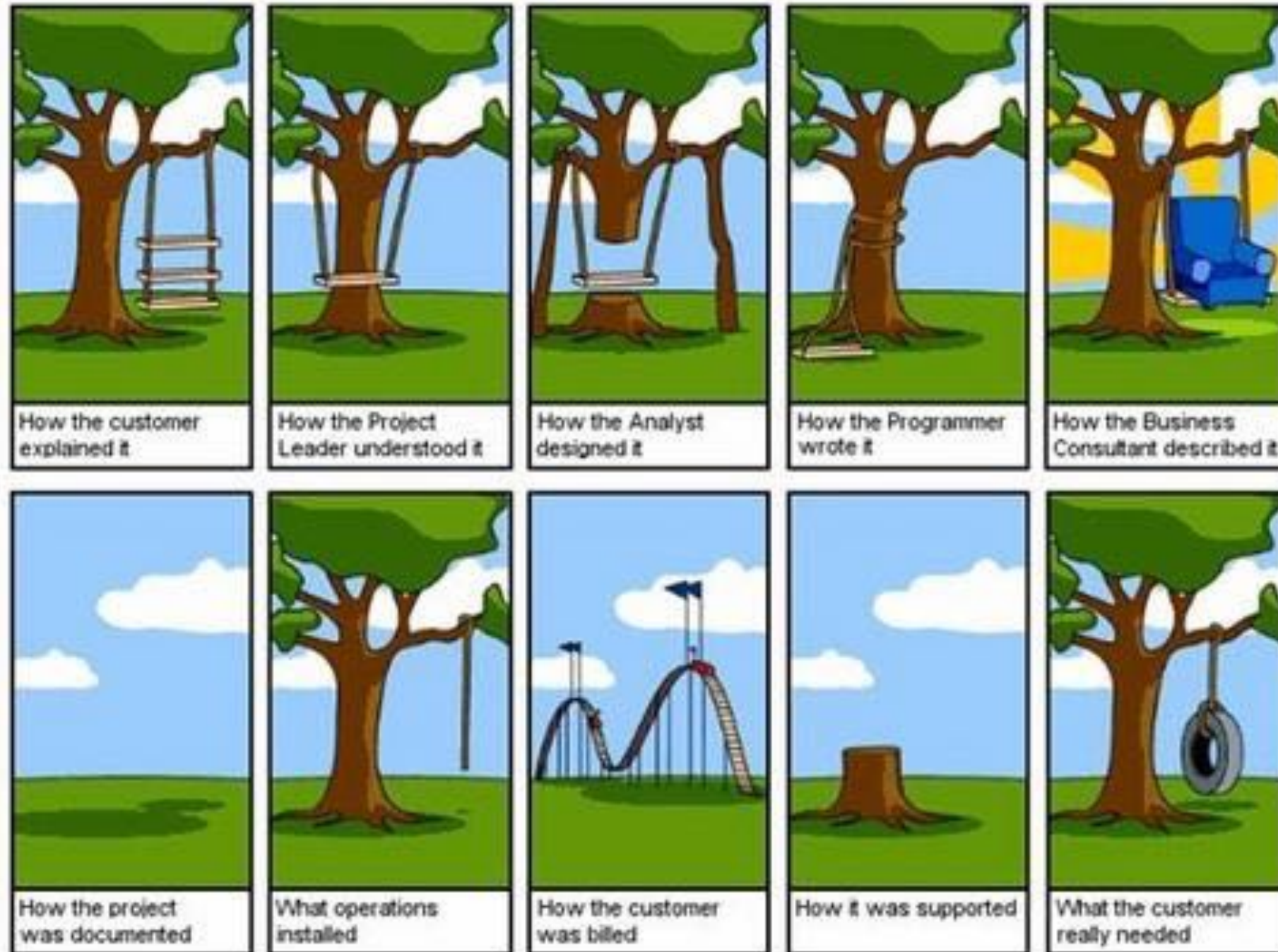


Line following



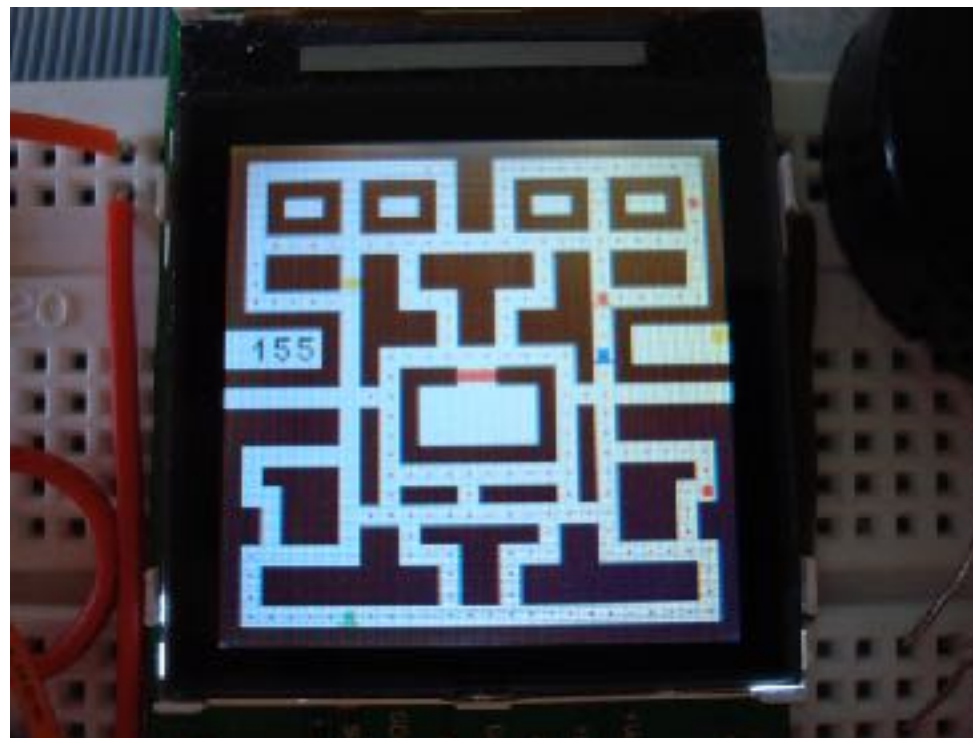
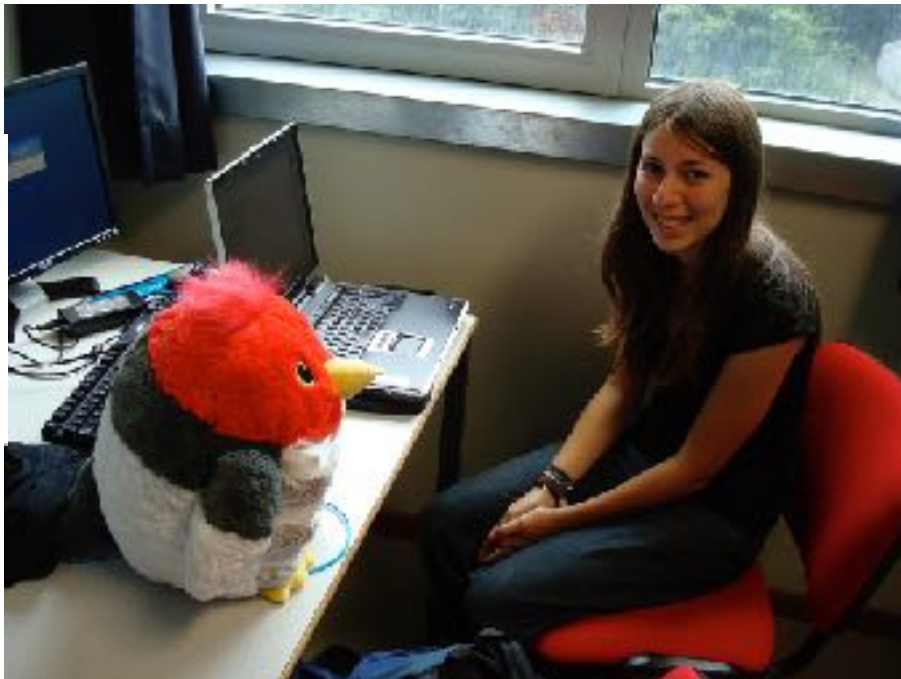
Obstacle avoidance

# Makkelijke manieren om te falen voor het project





# 2007-2008 Virtueel Huisdier



# Functionele vereisten

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 lichtsensoren 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.

# Niet functionele vereisten

*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.

Voor het tekenen van de robot moet je gebruik maken van de **gloss** bibliotheek (<https://hackage.haskell.org/package/gloss>).

*Je simulator moet multithreaded zijn.* Dit wilt concreet zeggen dat je simulator en het programma dat gebruik maakt van de simulator in een aparte thread draaien.

# Rapportering

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.



# BNF Voorbeeld

AExp ::= Int  
      | Id  
      | AExp + AExp  
      | AExp / AExp

BExp ::= Bool  
      | AExp <= AExp  
      | not BExp  
      | BExp and BExp

Stmt ::= skip  
      | Id := AExp  
      | Stmt ; Stmt  
      | if BExp then Stmt else Stmt  
      | while BExp do Stmt

Pgm ::= var List{Id } ; Stmt

Bool ::= true | false  
Int   ::= integers  
Id     ::= variabelen

Het is niet nodig om  
standaard primitieven  
expliciet uit te schrijven

# Focus !

- Parser
- Evaluator
- Use of monad transformers
- Code comments
- Code Quality
- Report

- My robot algorithm  
is the best

Meer documentatie op  
minerva