*Declarations*:

*sFront* = sensor voor

*sLeft* = sensor links

*sRight* = sensor rechts

*sBack* = sensor achter

*corner* = iterator

*specifiedDistance* = ...

*diagonal* = 0

1. **Level 1 – Bekend parcours met hindernissen**

**Loop**

**If (sVoor)**

*Go right*

*End loop*

**Else**

*Go straight*

**Loop**

**If (sVoor ‘bridge-distance’)**

*Go straight*

*End loop*

**Else**

*Go straight*

**Loop**

**If (sVoor)**  
 *Go left*

*End loop*

**Else**

*Go straight*

**Loop**

**If (sVoor)**

*Go left  
 End loop*

**Else**

*Go straight*

**Loop**

**If (sVoor)** *Go right*

*End loop*

**Else**

*Go straight*

1. **Level 2 – Onbekend hindernissenparcours**

**Loop**

**If (sVoor ‘bridge-distance’)**

*Go straight*

**Else if (sVoor)**

**If (sLeft)**

*Go right*

**If (sRight)**

*Go left*

**If (sRight && sLeft)**

*Go back*

**Else**

*Go right*

**Else** *Go straight*

1. **Level 3 – Zoek object in een open ruimte**

Kanten vinden en afzoeken, dan diagonaal van hoek tot hoek, dan zigzaggen.

**Loop**

**Loop: for (corner = 0; corner < 6; corner ++)**

**If (sVoor)**

*Go right* **If (sLeft > specifiedDistance)**

*Correct to the left*

**If (sLeft < specifiedDistance)**

*Correct to the right*

**If (!sLeft)**

*Go left*

**Loop: while (diagonal < 2)**

**If (sVoor)**

*Go diagonally (Pythagoras)*

*diagonal++*

**If (diagonalLength == Pythagoras && sVoor && diagonal == 1)**

*Go right*

**If (diagonalLength == Pythagoras && sVoor && diagonal == 2)**

*Go left*

**Loop: Do...**

**If (sVoor)**

*Go left*

*Go left*

**If (sVoor)**

*Go right*

*Go right*

**While (!sLeft || !sRight)**

1. **Level 4 – Zoek een object in een open ruimte met hindernissen**

1. **Level 5 – Zoek een object in een onbekend hindernissenparcours met tijdlimiet**