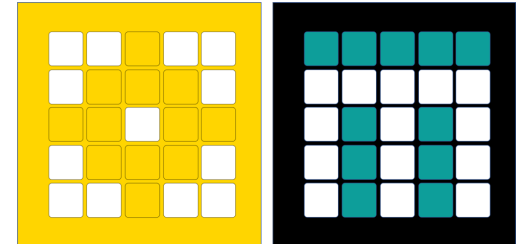


# PRIME LESSONS

By the Makers of EV3Lessons



## PID-LIJN VOLGER

DOOR SANJAY EN ARVIND SESHAN

VERTAALD ROY KRIKKE EN HENRIËTTE VAN DORP

Deze les maakt gebruik van SPIKE 3-  
software

# LESDOELSTELLINGEN

- Leer de beperkingen van proportionele regeling
- Ontdek wat PID betekent
- Leer hoe u PID programmeert en afstemt

# WANNEER LEVERT PROPORTIONELE CONTROLE PROBLEMEN OP?

Let op: de volgende paar dia's zijn geanimeerd. Gebruik de PowerPoint-presentatiemodus om ze te bekijken

## Wat zou een mens doen?

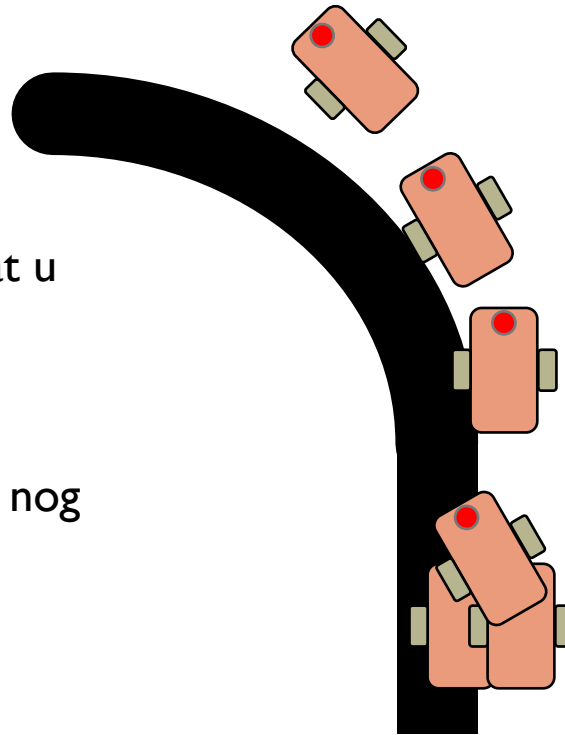
Op lijn → ga rechtdoor

Op wit → linksaf

Als u de lijn overschrijdt, gaat u  
rechtsaf

Op wit → linksaf

Verder van de lijn komen → nog  
meer draaien!



## Wat zou proportionele controle doen?

Op lijn → ga rechtdoor

Op wit → linksaf

**Ga over de lijn → ga  
rechtdoor!**

Op wit → linksaf

**Verder van lijn komen →  
evenveel linksaf slaan!**

LICHT LEZEN = **500%**

# HOE KUNNEN WE PROPORTIONELE CONTROLE OPLOSSEN?

Wat zou een mens doen?

Linksaf/op lijn → rechtsaf

Verder van de lijn komen → nog meer draaien!

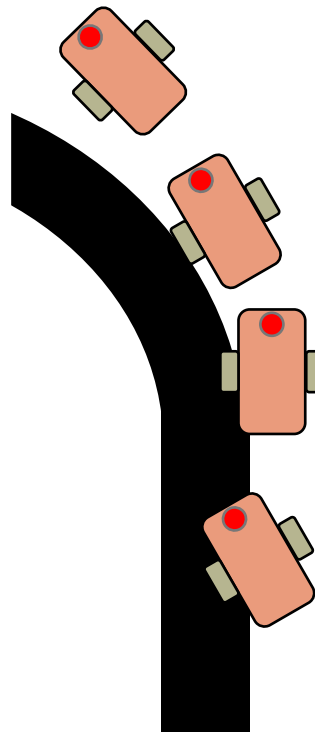
1. Voorspel wat de volgende sensormeting zal zijn

Wat zou proportionele controle doen?

**Sla linksaf/op lijn → ga rechtdoor!**

**Verder van lijn komen → evenveel linksaf slaan!**

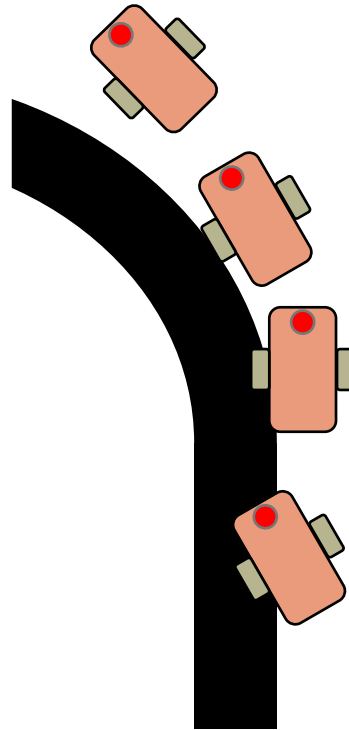
2. Hebben eerdere correcties op het gebied van stuurbedrag bijgedragen tot het verminderen van fouten?



# LESDOELSTELLINGEN

## 1. Voorspellen wat de volgende sensormeting zal zijn?

- Als de meetwaarden zijn: 75, 65, 55 → wat denk je dat de volgende meetwaarde zal zijn?
  - Wat als de meetwaarden 57, 56, 55 waren...
- Welke informatie heb je gebruikt om te raden?
- Afgeleide → de snelheid waarmee een waarde verandert



## 2. Hebben eerdere correcties op het gebied van stuurgedrag bijgedragen tot het verminderen van fouten?

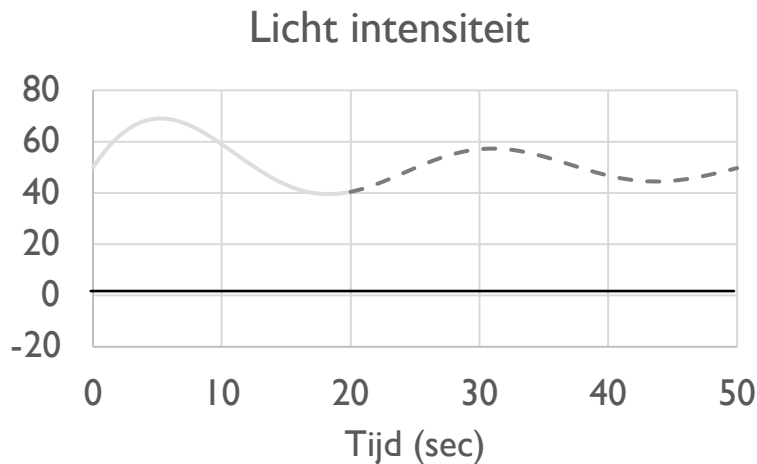
- Hoe zien de foutmetingen eruit als de correctie goed werkt?
  - +5, -6, +4 -3.... dat wil zeggen stuiten rond 0
- Hoe ziet de fout eruit als de besturing niet werkt?
  - +5, +5, +6, +5... dat wil zeggen altijd aan één kant van 0
- Hoe kunnen we dit eenvoudig detecteren?
  - Tip: kijk naar de som van alle fouten uit het verleden
- Wat is een ideale waarde voor dit bedrag? Wat betekent het als de som groot is?
- Integraal → de 'som' van waarden

# WAT IS PID?

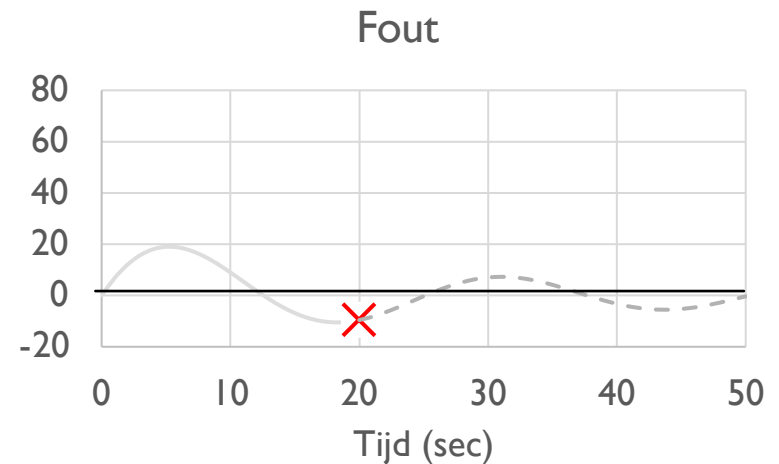
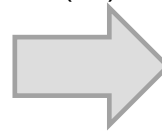
- **Proportioneel** [fout] → Hoe erg is de situatie nu?
- **Integraal** → Hebben mijn eerdere oplossingen geholpen om dingen op te lossen
- **D**erivatief → Hoe verandert de situatie?
- PID-regeling → combineert de fout-, integrale en afgeleide waarden om te beslissen hoe de robot moet worden bestuurd

# FOUT

- De ononderbroken lijn vertegenwoordigt wat je hebt gezien, de stippellijn is de toekomst
- Op tijdstip 20 zie je lichtwaarde = 40 en fout = -10 (rode X)

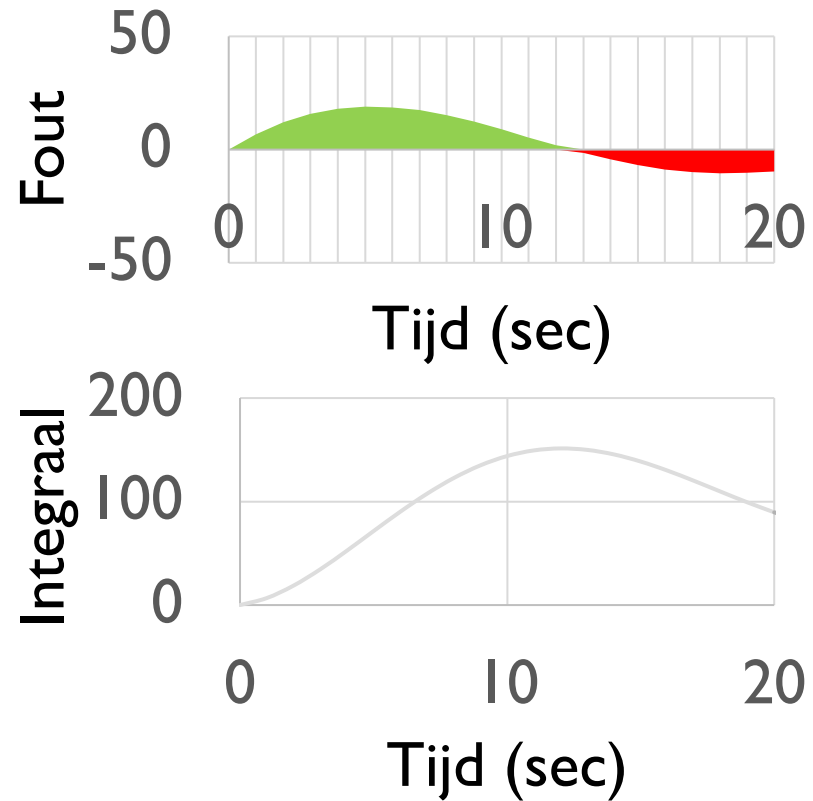


af trekken  
(50)



# INTEGRAAL

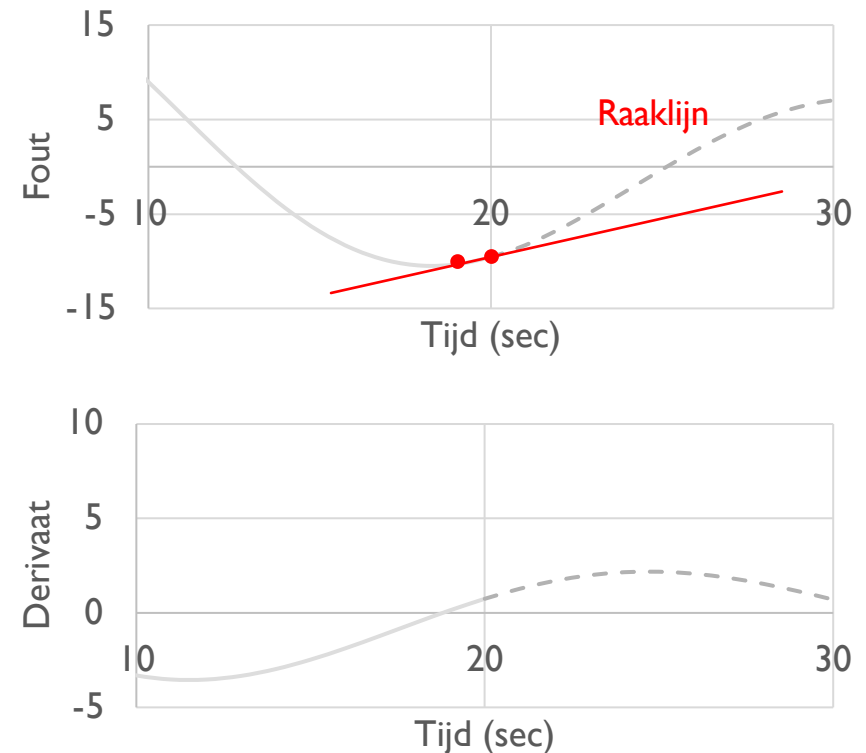
- Kijkt naar de geschiedenis van lijnvolger in het verleden
- Som van fouten uit het verleden
- Zoals gebied onder de curve in grafiek (integraal)
  - Groen = positief gebied
  - Rood = negatief gebied





# DERIVAAT

- Hoe snel verandert de positie?
  - Voorspelt waar de robot zich in de nabije toekomst zal bevinden
  - Hetzelfde als hoe snel fouten veranderen
- Kan worden gemeten met behulp van een raaklijn aan metingen → afgeleide
  - Geschat op basis van twee nabijgelegen punten in de grafiek



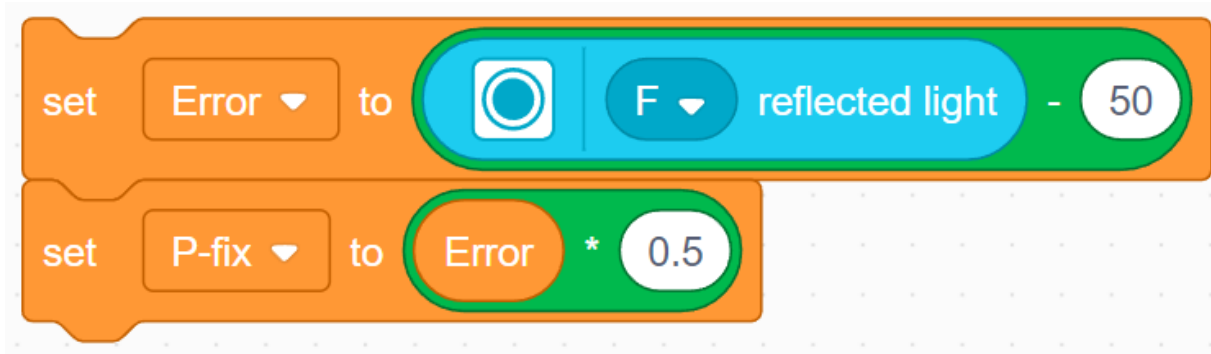
# PSEUDOCODE

1. Voer een nieuwe lichtsensormeting uit
2. Bereken de “fout”
3. Schaalfout om bijdrage aan stuurupdate te bepalen (proportionele regeling)
4. Gebruik fout om integraal bij te werken (som van alle fouten uit het verleden)
5. Schaalintegraal om bijdrage aan stuurupdate te bepalen (integrale sturing)
6. Gebruik fout om afgeleide bij te werken (verschil met laatste fout)
7. Schaalafgeleide om bijdrage aan stuurupdate te bepalen (afgeleide controle)
8. Combineer P-, I- en D-feedback en stuur de robot

# CODE - PROPORTIONEEL

- Dit is hetzelfde als de proportionele besturingscode

Fout = afstand tot lijn = aflezing - doel



Correctie (  $P_{fix}$  ) = Fout geschaald door proportionele constante (  $K_p$  ) = 0,5

# CODE - INTEGRAAL

- In deze sectie wordt de integraal berekend. Het voegt de huidige fout toe aan een variabele die de som heeft van alle voorgaande fouten.
- De schaalconstante is meestal klein, omdat Integraal groot kan zijn

Integraal = som van alle fouten uit het verleden = laatste

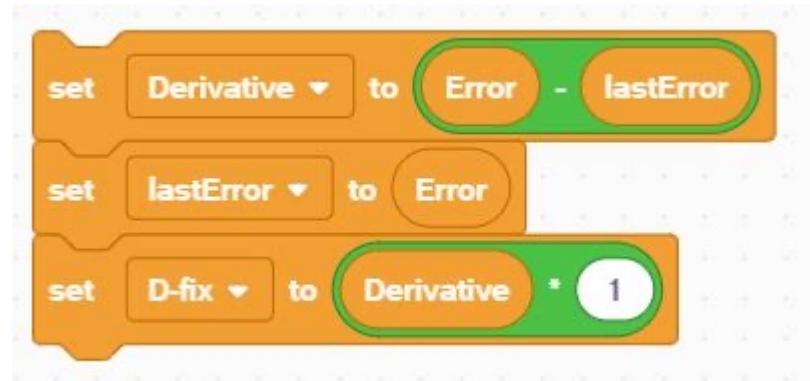


Correctie (  $I_{fix}$  ) = Integraal geschaald door proportionele constante (  $K_i$  ) = 0,001

# CODE - AFGELEIDE

- Dit codegedeelte berekent de afgeleide. Het trekt de huidige fout af van de fout uit het verleden om de verandering in de fout te vinden.

Afgeleide = mate van foutverandering = huidige fout – laatste fout

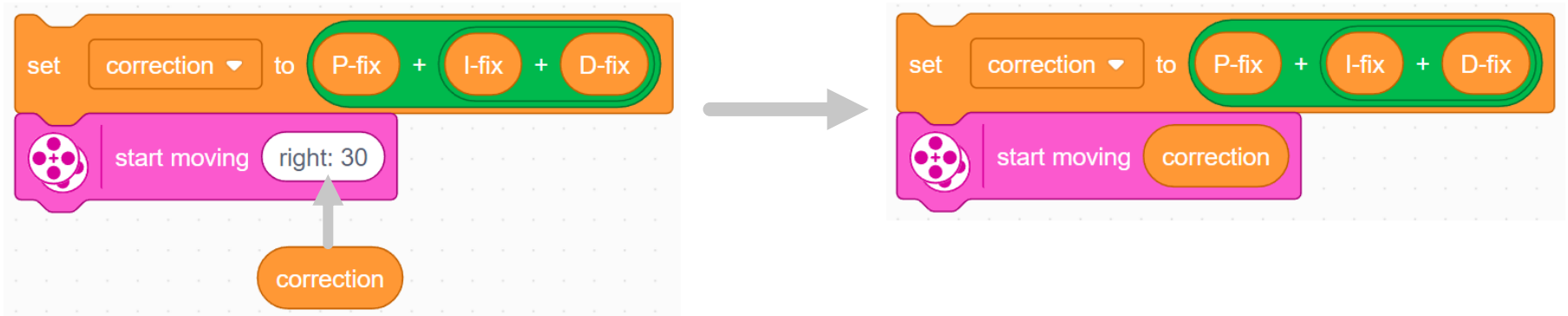


Correctie (  $D_{\text{fix}}$  ) = Afgeleide geschaald door proportionele constante (  $K_d$  ) = 1,0

# ALLES SAMENBRENGEN

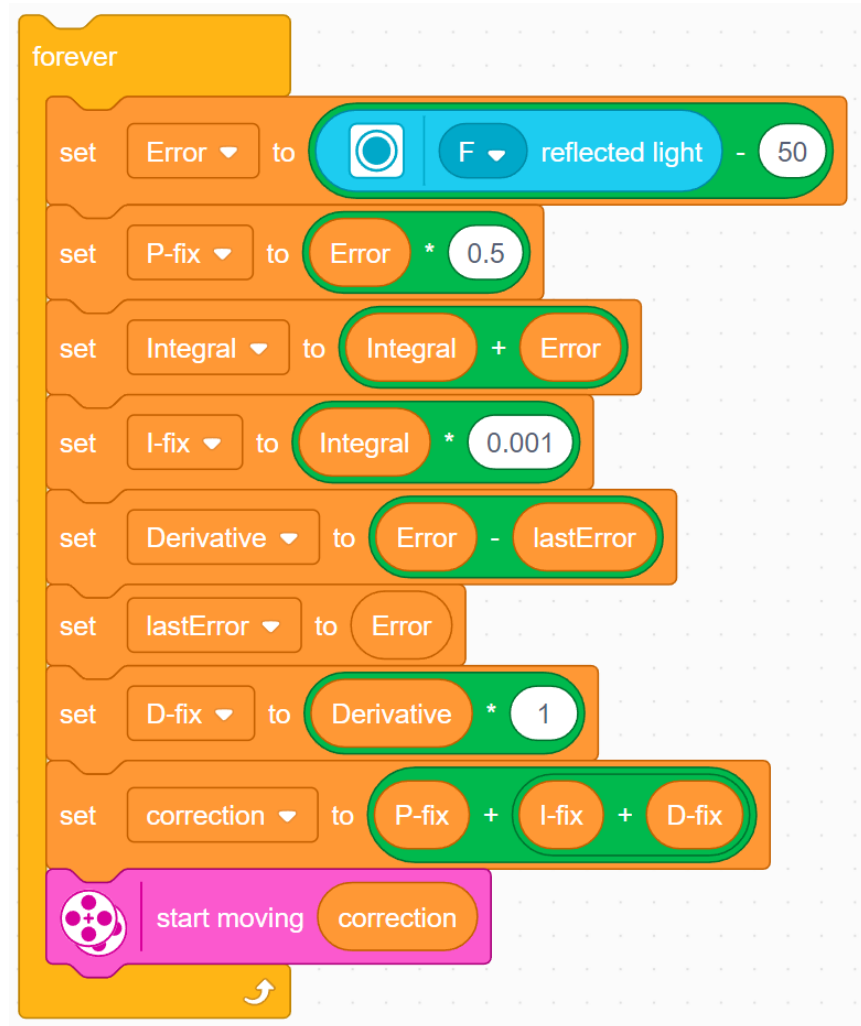
- Elk van de componenten is al geschaald. Op dit punt kunnen we ze eenvoudig bij elkaar optellen.
- Voeg de drie oplossingen voor P, I en D samen. Hiermee wordt de definitieve correctie berekend

Pas de correctie toe op de besturing van een bewegend stuurblok



# VOLLEDIGE CODE

- Dit is wat je krijgt als je al deze onderdelen samenvoegt.
- We hopen dat je nu begrijpt hoe PID een beetje beter werkt.



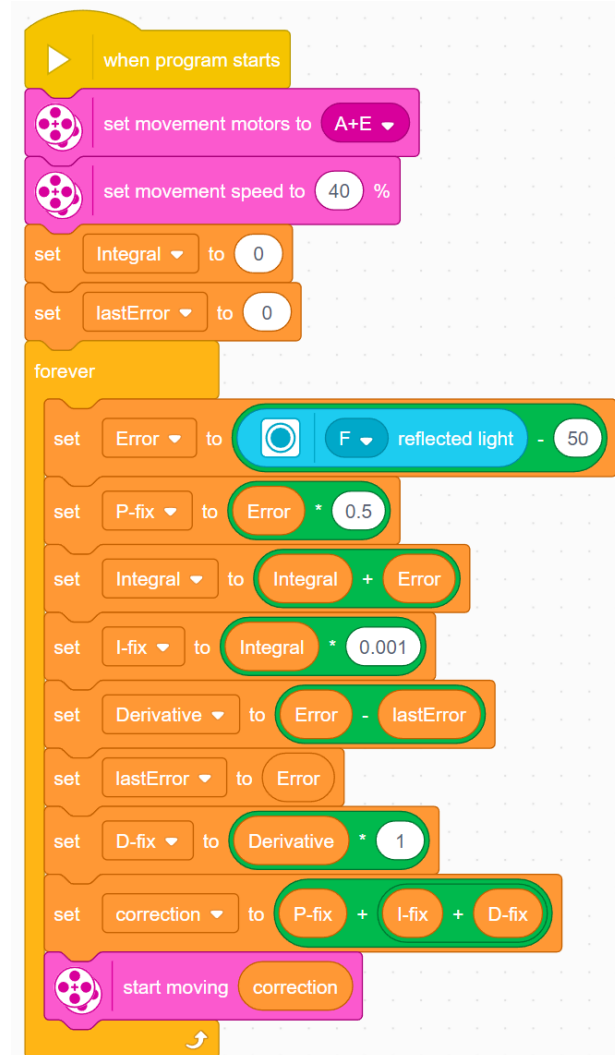
```
forever
  set Error to F reflected light - 50
  set P-fix to Error * 0.5
  set Integral to Integral + Error
  set I-fix to Integral * 0.001
  set Derivative to Error - lastError
  set lastError to Error
  set D-fix to Derivative * 1
  set correction to P-fix + I-fix + D-fix
  start moving correction
```

The image shows a Scratch script for a PID controller. It is enclosed in a 'forever' loop. The script consists of the following blocks:

- set Error to**  $F \text{ reflected light} - 50$
- set P-fix to**  $\text{Error} * 0.5$
- set Integral to**  $\text{Integral} + \text{Error}$
- set I-fix to**  $\text{Integral} * 0.001$
- set Derivative to**  $\text{Error} - \text{lastError}$
- set lastError to**  $\text{Error}$
- set D-fix to**  $\text{Derivative} * 1$
- set correction to**  $\text{P-fix} + \text{I-fix} + \text{D-fix}$
- start moving**  $\text{correction}$

# VOLLEDIGE CODE

Stel de variabelen in voor de laatste fout en integraal vóór de lus en initialiseer deze op 0, omdat ze worden gelezen voordat ze worden geschreven. Stel bovendien de bewegingsmotoren en snelheid in.



```
when program starts
  set movement motors to A+E
  set movement speed to 40 %
  set Integral to 0
  set lastError to 0
  forever
    set Error to F reflected light - 50
    set P-fix to Error * 0.5
    set Integral to Integral + Error
    set I-fix to Integral * 0.001
    set Derivative to Error - lastError
    set lastError to Error
    set D-fix to Derivative * 1
    set correction to P-fix + I-fix + D-fix
    start moving correction
```

The image shows a Scratch script for a PID controller. It starts with a 'when program starts' block, followed by three 'set' blocks for 'movement motors' (A+E), 'movement speed' (40%), 'Integral' (0), and 'lastError' (0). A 'forever' loop contains several 'set' blocks: 'Error' (F reflected light - 50), 'P-fix' (Error \* 0.5), 'Integral' (Integral + Error), 'I-fix' (Integral \* 0.001), 'Derivative' (Error - lastError), 'lastError' (Error), 'D-fix' (Derivative \* 1), and 'correction' (P-fix + I-fix + D-fix). The loop ends with a 'start moving' block using the 'correction' variable.



# BELANGRIJKE STAP: AFSTEMMEN VAN DE PID-CONSTANTEN

- De meest gebruikelijke manier om uw PID-constanten af te stemmen is met vallen en opstaan.
- Dit kan enige tijd duren. Hier zijn een paar tips:
  - Schakel alles uit behalve het proportionele deel (zet de andere constanten op nul). Pas alleen de proportionele constante aan totdat de robot de lijn goed volgt.
  - Schakel vervolgens de integraal in en pas deze aan totdat deze goede prestaties levert op een reeks lijnen.
  - Schakel ten slotte de afgeleide in en pas deze aan totdat u tevreden bent met de volgende regel.
  - Bij het inschakelen van elk segment zijn hier enkele goede cijfers om mee te beginnen voor de constanten:
    - P: 1,0 aanvankelijk aangepast met  $\pm 0,5$  en  $\pm 0,1$  voor fijnafstemming
    - I: 0,05 aanvankelijk aangepast met  $\pm 0,01$  en  $\pm 0,005$  voor fijnafstemming
    - D: 1,0 aanvankelijk aangepast met  $\pm 0,5$  en  $\pm 0,1$  voor fijnafstemming

# EVALUEREN VAN LIJNVOLGERS

## Proportioneel

- Gebruikt de “P” in PID
- Maakt proportionele bochten
- Werkt goed op zowel rechte als gebogen lijnen
- Goed voor gemiddelde tot gevorderde teams die wiskundeblokken moeten kennen

## PID

- Het is beter dan proportionele besturing op een zeer gebogen lijn, omdat de robot zich aanpast aan de bochten
- Voor de FIRST LEGO League, die meestal rechte lijnen heeft, kan proportionele controle echter voldoende zijn

# CREDITS

- Deze les is gemaakt door Sanjay Seshan en Arvind Seshan voor Prime Lessons
- Deze lessen zijn door Roy Krikke en Henriëtte van Dorp vertaald in het Nederlands
- Meer lessen zijn beschikbaar op [www.primelessons.org](http://www.primelessons.org)



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).