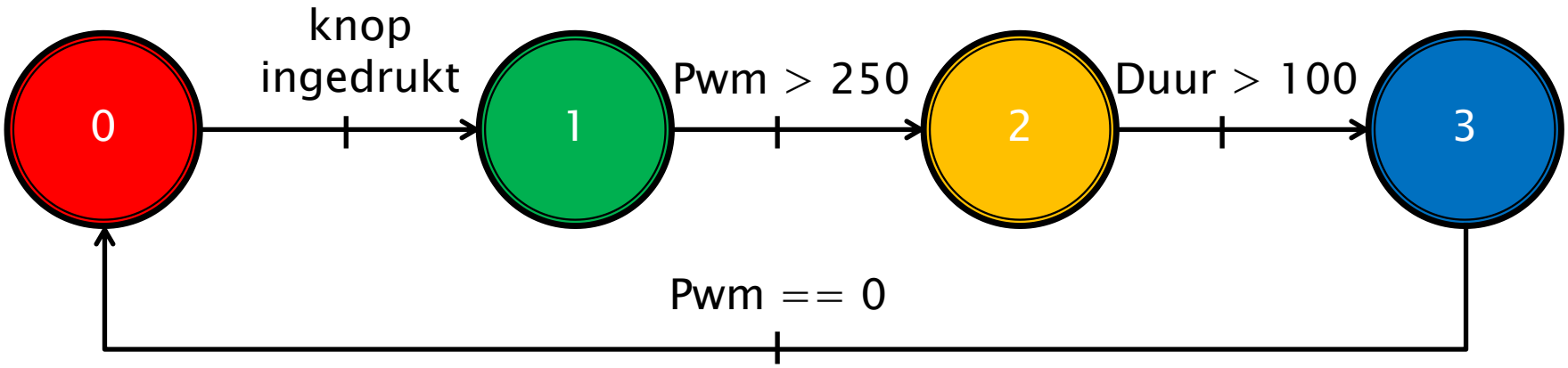
CAR – Dag 5

C, Arduino & Robots

3 mei 2014

Huiswerk

- 1. Maak de onderstaande statemachine voor PWM aansturing van de LED.
- Loop tijd is 10 ms.



State acties:

- State 0: Pwm = 0
- State 1: verhoog Pwm in stapjes van 2
- State 2: verhoog teller Duur met 1, te beginnen bij 0.
- State 3: verlaag Pwm in stapjes van 1.

Huiswerk – state machine Pwm

les_5_p10_state_machine_pwm

```
// to support printf
int my_putc(char c, FILE *t) {
  Serial.write(c);
}

// the setup routine runs once wh
void setup() {

  // start serial
  Serial.begin(115200);

  fdevopen( &my_putc, 0);  // dev
  pinMode(11,  INPUT_PULLUP);  //

  printf("Opstarten gereed.\n");
}

int Toestand = 0;
int Pwm, Duur;

void loop() {
```

```
void loop() {
  delay(10);

  printf("T %d, P %d, D %d\n", Toestand, Pwm, Duur);

  switch(Toestand) {
    case 0 : {  // led uit
      // State actie
      analogWrite(6, 0);

      // Conditie
      if (digitalRead(2) == 0) { // knop ingedrukt?
        Pwm = 0;
        Toestand = 1;
      }
      break;
    }
  case 1 : {  // led wordt feller
      // State actie
      Pwm = Pwm + 2;
      analogWrite(6, Pwm);

      // Conditie
      if (Pwm > 250) { // (bijna) volle sterkte?
        Duur = 0;
        Toestand = 2;
      }
      break;
    }
  case 2 : { // led 100 ticks op volle sterkte
```

```
  case 2 :
    {  // led 100 ticks op volle sterkte
      // State actie
      Duur++;       // Duur = Duur + 1;

      // Conditie
      if (Duur > 100) { // tijd voorbij?
        Toestand = 3;
      }
      break;
    }
  case 3 :
    {  // led wordt minder fel
      // State actie
      Pwm = Pwm - 1;
      analogWrite(6, Pwm);

      // Conditie
      if (Pwm == 0) { // led uit / pwm 0?
        Toestand = 0;
      }
      break;
    }
  default :
    {
      Serial.println("Ongeldige state");
      Toestand = 0;
      break;
    }
  } // einde van switch
}
```

Huiswerk

- 2. Voorspel de output

les_4_p80_puzzel

```
  printf("* %d *\n", 123);
  printf("* %6d *\n", 123);
  printf("* %06d *\n", 123);
  printf("* %-6d *\n", 123);
  printf("* 0x%02x *\n", 123);
  printf("* 0x%02X *\n", 123);

  int x = 7;
  while (x) {
    printf("x: %d\n", x);
    x--;
  }

  int y = -3;
  while (y<100) {
    printf("y: %d\n", y);
    y += 7;
  }

  while (1) {
    delay(1000);
    printf("lus \"while (1)\"\n");
  }
}

void loop() {
  delay(1000);
  printf("Hoofdlus.\n");
}
```

Huiswerk

- 2. Voorspel de output

COM16

```
Opstarten gereed.
* 123 *
*    123 *
* 000123 *
* 123    *
* 0x7b *
* 0x7B *
x: 7
x: 6
x: 5
x: 4
x: 3
x: 2
x: 1
```

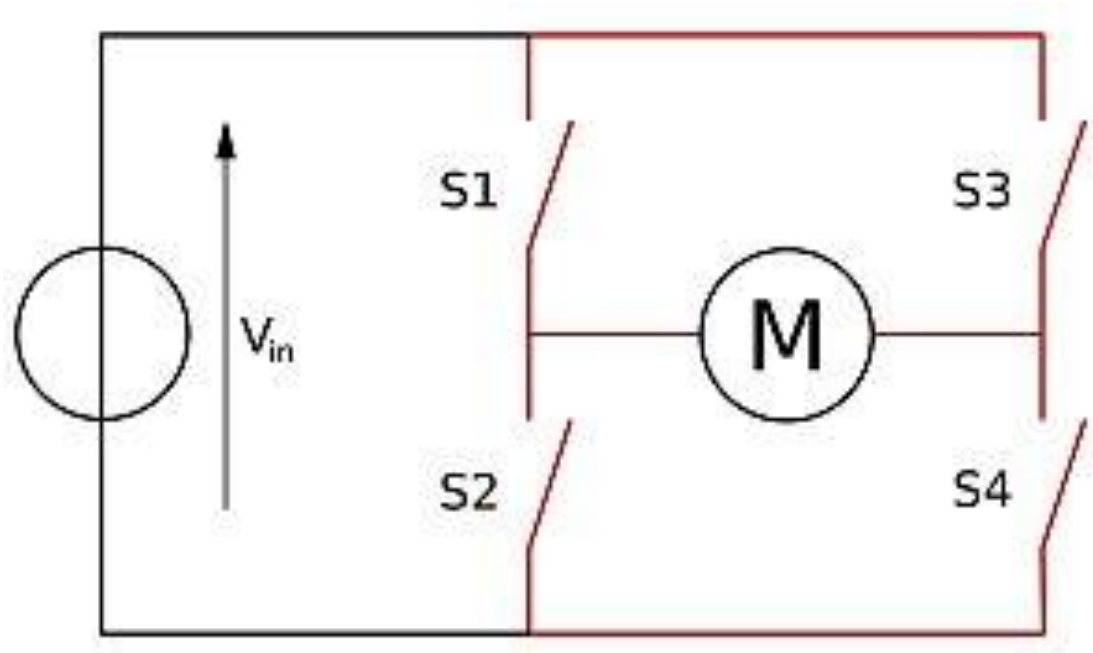
```
y: -3
y: 4
y: 11
y: 18
y: 25
y: 32
y: 39
y: 46
y: 53
y: 60
y: 67
y: 74
y: 81
y: 88
y: 95
lus "while (1)"
lus "while (1)"
```

les_4_p80_puzzel

```
int my_putc(char c, FILE *t) {
  return Serial.write(c);
}

void setup() {
  Serial.begin(115200);
  fdevopen( &my_putc, 0);  // device 0 (stdout) output naar my_pu
  printf("Opstarten gereed.\n");

  printf("* %d *\n", 123);
  printf("* %6d *\n", 123);
  printf("* %06d *\n", 123);
  printf("* %-6d *\n", 123);
  printf("* 0x%02x *\n", 123);
  printf("* 0x%02X *\n", 123);

  int x = 7;
  while (x) {
    printf("x: %d\n", x);
    x--;
  }

  int y = -3;
  while (y<100) {
    printf("y: %d\n", y);
    y += 7;
  }

  while (1) {
    delay(1000);
    printf("lus \"while (1)\"\n");
  }
}

void loop() {
  delay(1000);
  printf("Hoofdlus.\n");
}
```

Motorsturing

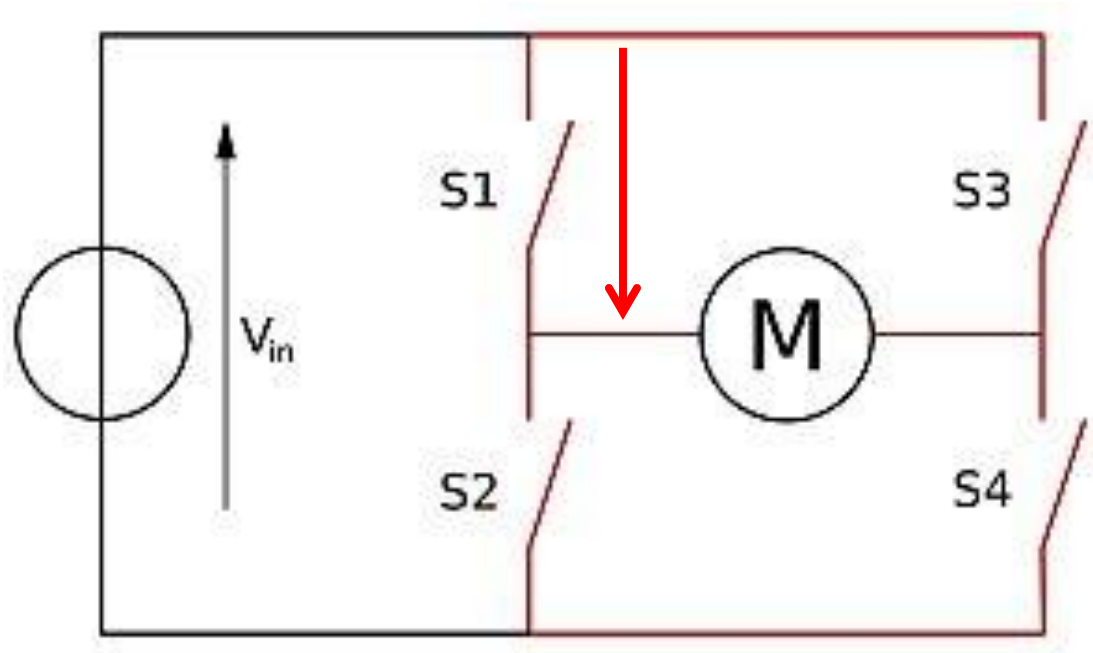
- PWM – langzaam en snel.
- Richting (DIR, direction) – vooruit of achteruit.



Richting	PWM	Schakelaar
0	0	S1
0	1	S1 + S4
1	0	S3
1	1	S3 + S2

Motorsturing

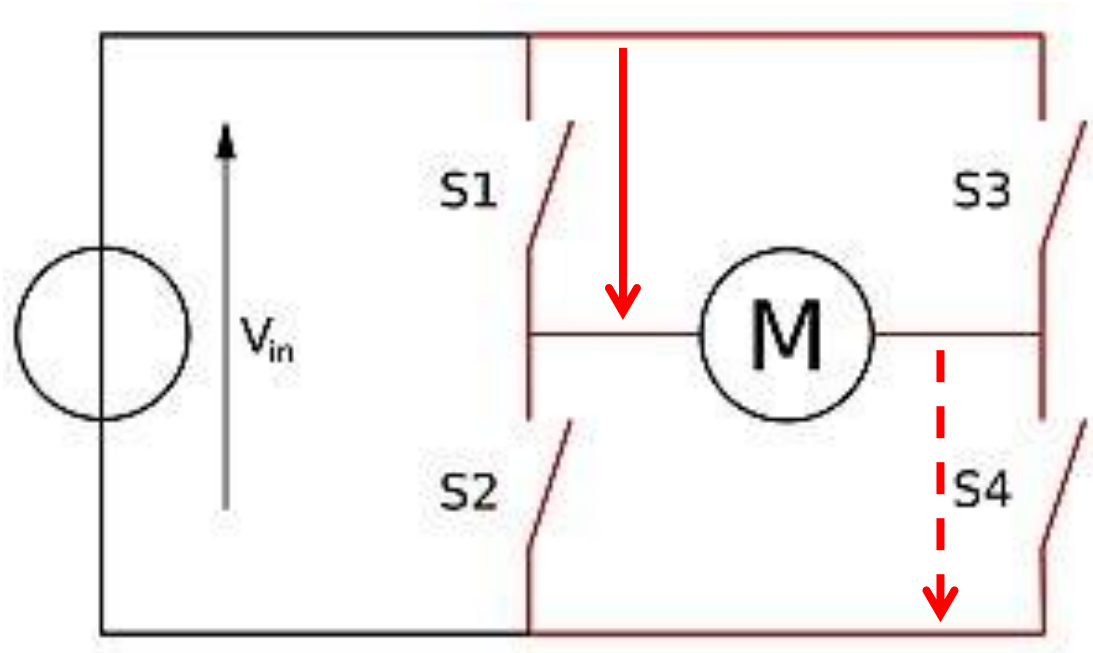
- PWM – langzaam en snel.
- Richting (DIR, direction) – vooruit of achteruit.



Richting	PWM	Schakelaar
0	0	S1
0	1	S1 + S4
1	0	S3
1	1	S3 + S2

Motorsturing

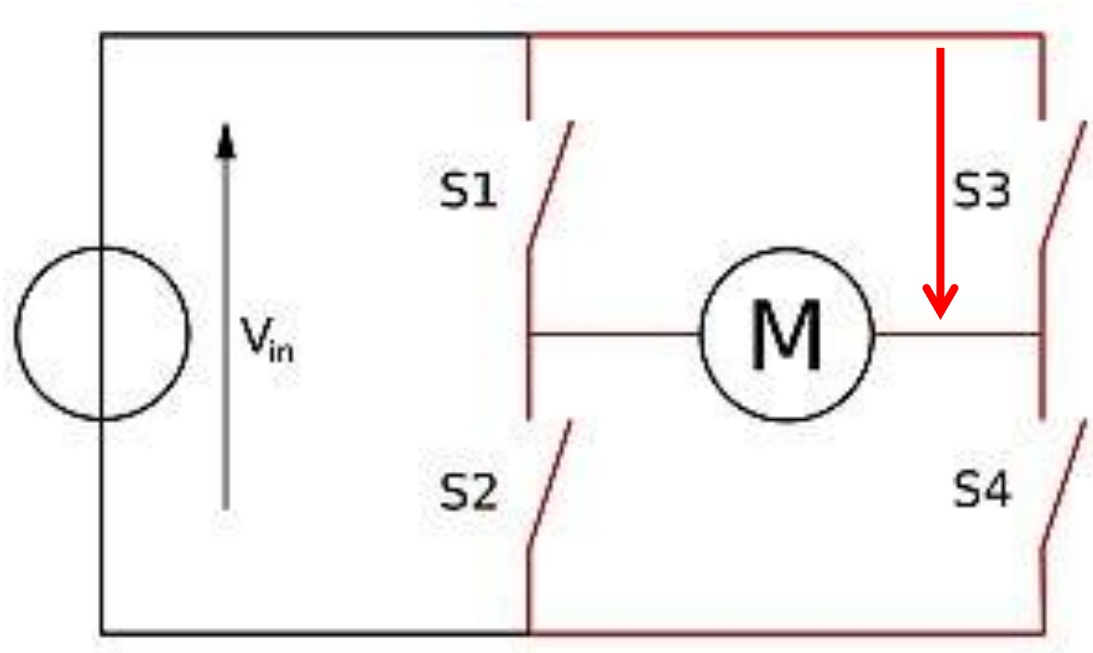
- PWM – langzaam en snel.
- Richting (DIR, direction) – vooruit of achteruit.



Richting	PWM	Schakelaar
0	0	S1
0	1	S1 + S4
1	0	S3
1	1	S3 + S2

Motorsturing

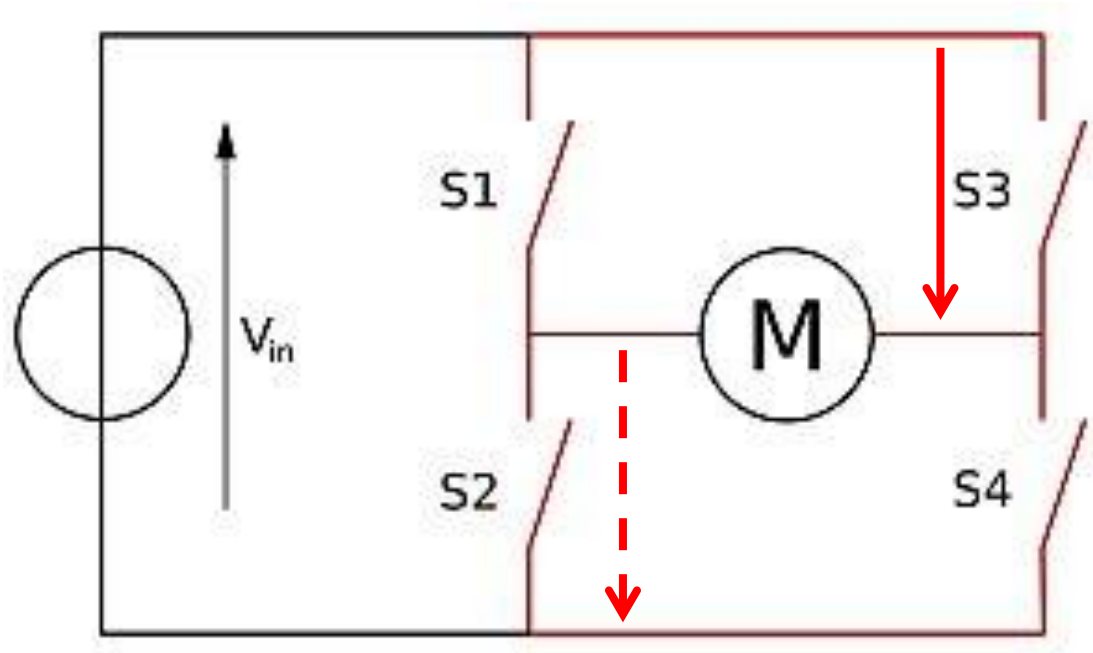
- PWM – langzaam en snel.
- Richting (DIR, direction) – vooruit of achteruit.



Richting	PWM	Schakelaar
0	0	S1
0	1	S1 + S4
1	0	S3
1	1	S3 + S2

Motorsturing

- PWM – langzaam en snel.
- Richting (DIR, direction) – vooruit of achteruit.



Richting	PWM	Schakelaar
0	0	S1
0	1	S1 + S4
1	0	S3
1	1	S3 + S2

Aansturing motorcontroller

les_5_p20_motorcontroller

```
const int PWML = 11;
const int PWMR = 3;

const int DIRL = 13;
const int DIRR = 12;

const int KNOP = 2;

// to support printf
int my_putc(char c, FILE *t)
  return Serial.write(c);
}
```

```
// the setup routine runs once when you press reset:
void setup() {
  Serial.begin(115200);            // start serial
  fdevopen( &my_putc, 0);         // device 0 (stdout) output naar my_putc()
  pinMode(KNOP,  INPUT_PULLUP);  // knop input & pull-up

  // Break gebruiken we verder niet...
  pinMode( 8, OUTPUT);          // brakeB
  digitalWrite(8, 0);
  pinMode( 9, OUTPUT);          // brakeA
  digitalWrite(9, 0);

  // Rijrichting
  pinMode(DIRL, OUTPUT);
  pinMode(DIRR, OUTPUT);
  digitalWrite(DIRL, 0);  // vooruit
  digitalWrite(DIRR, 0);  // vooruit

  // duty cycle
  analogWrite(PWML, 0);    // uit, 0 of 150..255
  analogWrite(PWMR, 0);    // uit

  printf("Opstarten gereed.\n");
}
```

```
void loop() {
  delay(10);

  if (digitalRead(KNOP) == 0) {
    // hier de motor code
  }
}
```

Oefening – laat motortjes draaien

- Maak het programma af
- Start met knop, stop met reset
- Probeer andere PWM waarden & richting

les_5_p20_motorcontroller

```
const int PWML = 11;
const int PWMR = 3;

const int DIRL = 13;
const int DIRR = 12;

const int KNOP = 2;

// to support printf
int my_putc(char c, FILE *t) {
  return Serial.write(c);
}
```

```
up routine runs once when you press reset:
() {
egin(115200);            // start serial
( &my_putc, 0);          // device 0 (stdout) output naar my_putc()
KNOP,  INPUT_PULLUP);    // kn

gebruiken we verder niet...
8, OUTPUT);              // brak
rite(8, 0);
9, OUTPUT);              // brak
rite(9, 0);

chting
pinMode(DIRL, OUTPUT);
pinMode(DIRR, OUTPUT);
digitalWrite(DIRL, 0);   // voorruit
digitalWrite(DIRR, 0);   // voorruit

// duty cycle
analogWrite(PWML, 0);    // uit, 0 o
analogWrite(PWMR, 0);    // uit

printf("Opstarten gereed.\n");
}
```

```
void loop() {
  delay(10);

  if (digitalRead(KNOP) == 0) {
    digitalWrite(DIRL, 0);   // voorruit
    digitalWrite(DIRR, 1);   // achterruit
    analogWrite(PWML, 200);
    analogWrite(PWMR, 200);
  }
}
```

Motorsturing via een functie – 1

- Input:

Per motor: Kracht van +255 tot -255

- Output:

Per motor: PWM en Richting

- Prototype

```
void SetupMotors();
void Motors(int PwmL, int PwmR);
```

Motorsturing via een functie – 2

MotorTest_Rev3 | MotorTest_Rev3_sm | Motors

```
//-----------------------------------------------------------------------------
// Motors.ino  - aansturing van de RedBot motors via enable (standaard methode)
//-----------------------------------------------------------------------------

// Constantes
const int PWML    = 11;
const int PWMR    =  3;

const int DIRL    = 13;
const int DIRR    = 12;

const int BREAKL  =  8;
const int BREAKR  =  9;
```

```
//-----------------------------------------------------------------------------
// SetupMotors - Stel IO in voor aansturing motoren
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
void SetupMotors()
{
  // Break gebruiken we (nog) niet...
  pinMode(BREAKL, OUTPUT);
  digitalWrite(BREAKL, 0);
  pinMode(BREAKR, OUTPUT);
  digitalWrite(BREAKR, 0);

  // Rijrichting
  pinMode(DIRL, OUTPUT);
  pinMode(DIRR, OUTPUT);

  // Pwm pins
  pinMode( PWML, OUTPUT);
  pinMode( PWMR, OUTPUT);

  // zet output op 0 via Motors()
  Motors(0, 0);
}
```

Motorsturing via een functie – 3

```
//-----------------------------------------------------------------------------
// Motors - Stel PWM duty cycle in voor beide motoren
//-----------------------------------------------------------------------------
//-----------------------------------------------------------------------------
void Motors(int PwmL, int PwmR)
{
  //--------
  // Motor L
  //--------

  // begrens waarden
  if (PwmL >  255) {
    PwmL =  255;
  }
  if (PwmL < -255) {
    PwmL = -255;
  }

  // Set PWM en richting
  if (PwmL >= 0) {
    digitalWrite(DIRL, 0);  // vooruit
    analogWrite(PWML, PwmL);
  }
  else {
    digitalWrite(DIRL, 1);  // achteruit
    analogWrite(PWML, -PwmL);
  }
```

```
  //--------
  // Motor R
  //--------

  // begrens waarden
  if (PwmR >  255) {
    PwmR =  255;
  }
  if (PwmR < -255) {
    PwmR = -255;
  }

  //Set PWM en richting
  if (PwmR >= 0) {
    digitalWrite(DIRR, 0);  // vooruit
    analogWrite(PWMR, PwmR);
  }
  else {
    digitalWrite(DIRR, 1);  // achteruit
    analogWrite(PWMR, -PwmR);
  }
}
```

Motorsturing via een functie – 4

les_5_p30_motorcontroller | Motors

```
const int KNOP = 2;

// to support printf
int my_putc(char c, FILE *t) {
  return Serial.write(c);
}

void setup() {
  Serial.begin(115200);            // start serial
  fdevopen( &my_putc, 0);         // device 0 (stdout) output naar my_putc()
  pinMode(KNOP,  INPUT_PULLUP);   // knop input & pull-up

  SetupMotors();
  printf("Opstarten gereed.\n");
}

void loop() {
  delay(10);

  if (digitalRead(KNOP) == 0) {
    Motors(200, -200);
  }
}
```

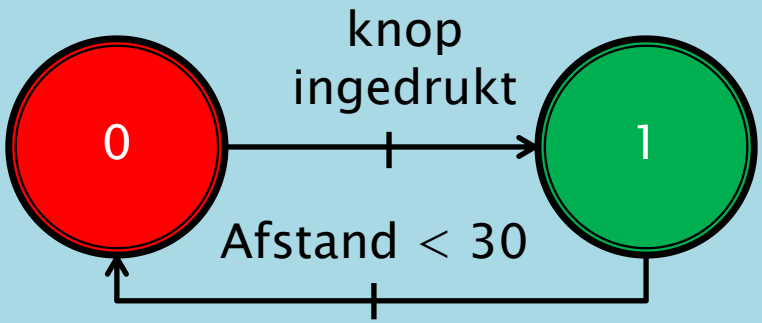
Oefening – Laat robot rijden (1)

- Knop -> start rijden.
- Object < 30 cm (Sharp sensor1) -> Stoppen.

les_5_p31_motorcontroller | Motors | StateMachine

```
const int KNOP = 2;
int Centimeters;

// to support printf
int my_putc(char c, FILE *t) {
  return Serial.write(c);
}

void setup() {
  Serial.begin(115200);           // start serial
  fdevopen( &my_putc, 0);        // device 0 (stdout) output naar my_putc()
  pinMode(KNOP,  INPUT_PULLUP);  // knop input & pull-up

  SetupMotors();
  printf("Opstarten gereed.\n");
}

void loop() {
  delay(10);

  Centimeters = SharpAfstand(A2);
  printf("Sharp afstand is %d\n", Centimeters);

  MotorTakt();
}

int SharpAfstand(int Pin)
{
  int SensorValue = analogRead(Pin);
  int Afstand = (40*148) / SensorValue;
  return Afstand;
}
```

Oefening – Laat robot rijden (2)



State acties:

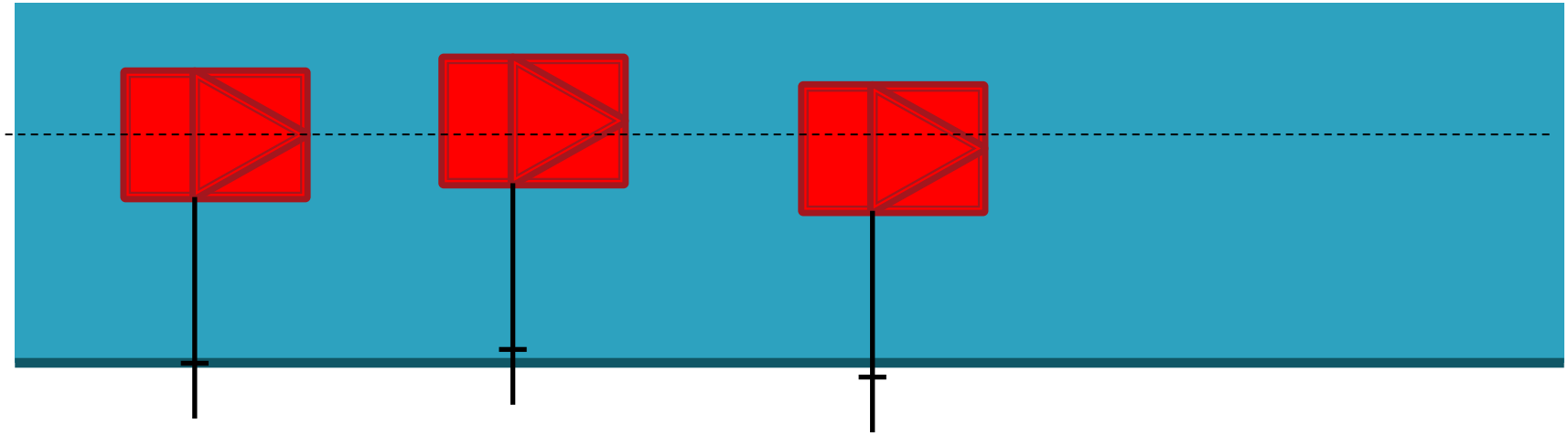
- State 0: Rust, Pwm = 0
- State 1: Rijden,
  PwmL = PwmR = 200

```
Centimeters = SharpAfstand(A2);
printf("Sharp afstand is %d\n", Centimeters);
```

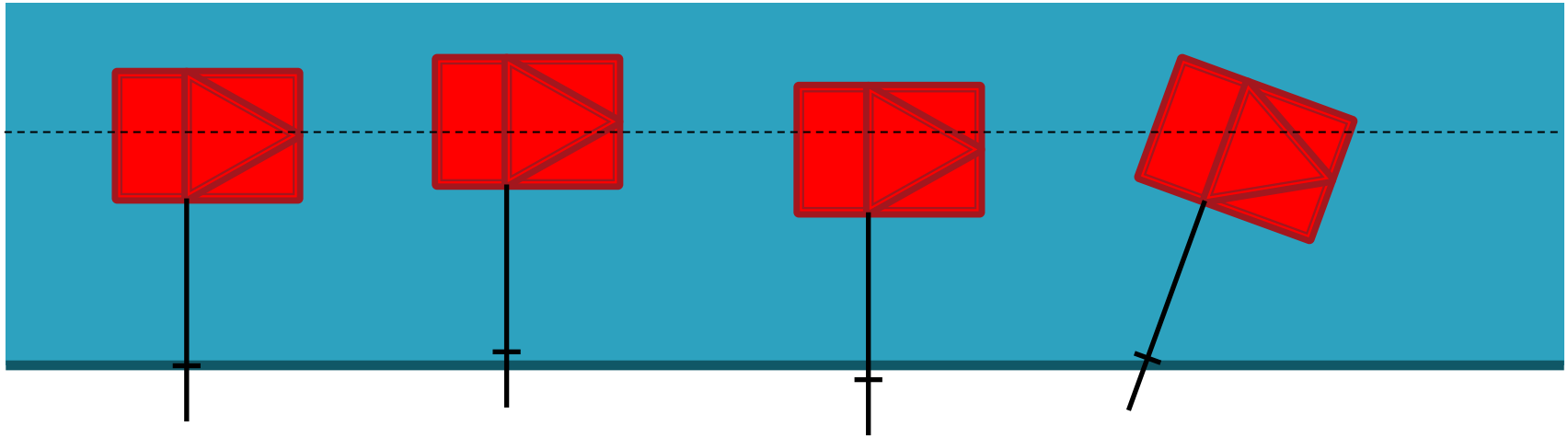
les_5_p31_motorcontroller | Motors | StateMachine

```
void MotorTakt()
{
  static int State, PrevState = -1;

  // rapporteer status bij state overgang
  if (PrevState != State) {
    PrevState = State;
    printf("MotorTakt State: %d\n", State);
  }

  // state machine
  switch(State) {
  case 0 : // State: rust
    {
      // State actie

      // Condities
      if (1) { // Knop ingedrukt
      }
      break;
    }

  default :
    {
      printf("MotorTakt: ongeldige state %d\n", State);
      State = 0;
      break;
    }
  } // einde van switch
}
```

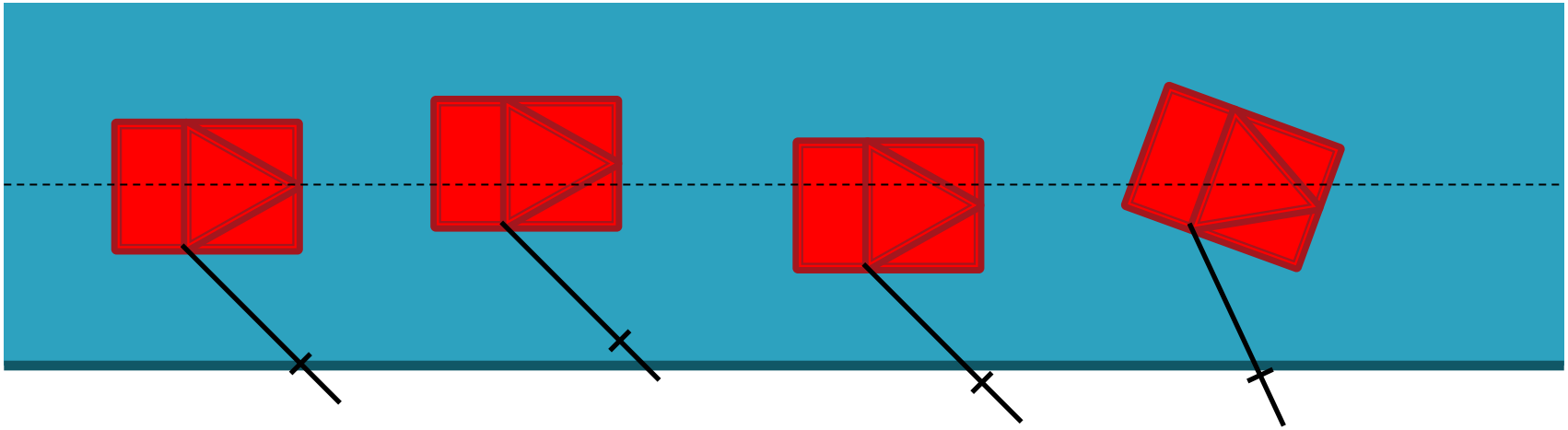
Wandvolgen (1)



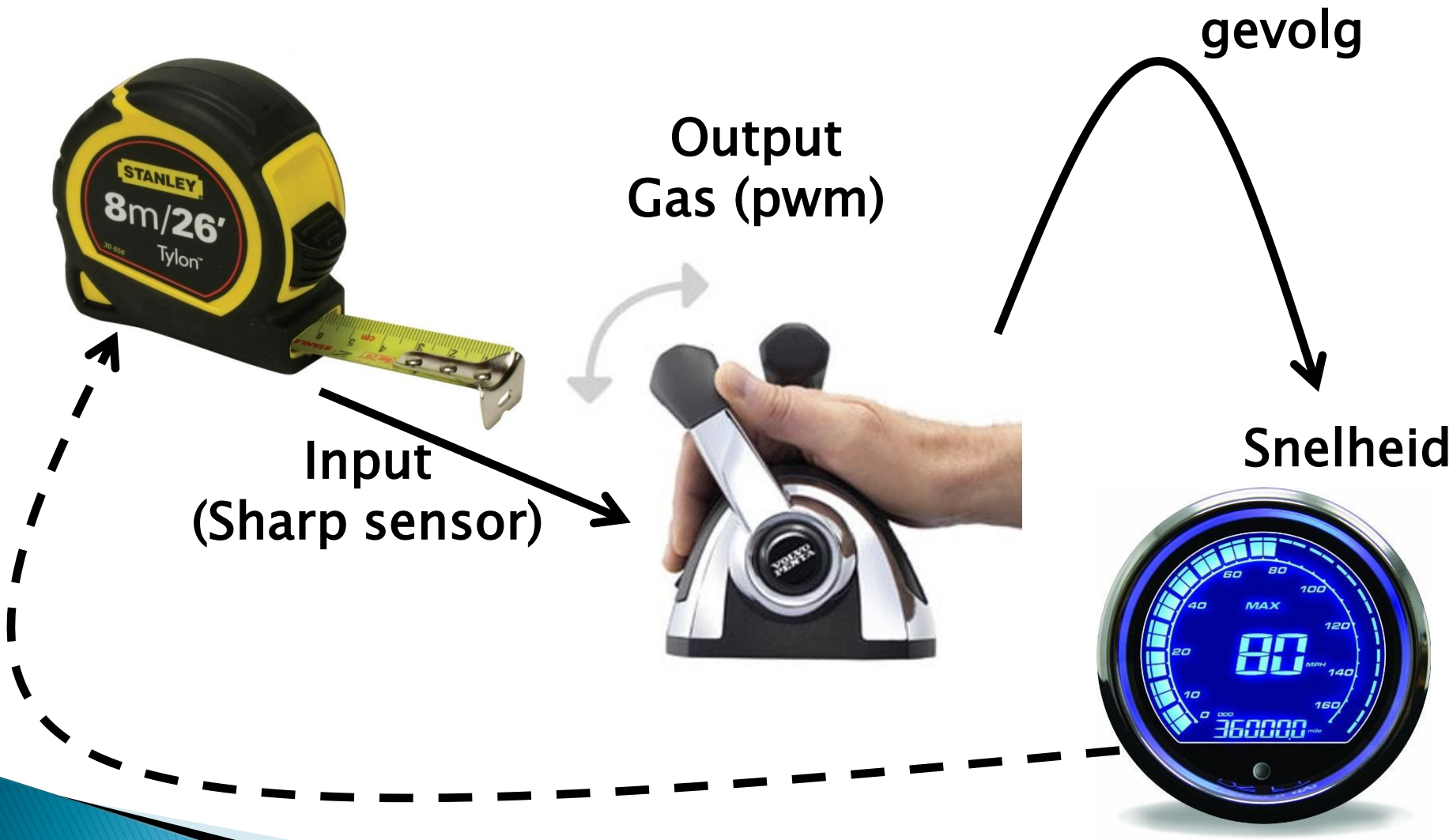
Wandvolgen (2)



Wandvolgen (3)



Regeling (1)


gevolg
Output
Gas (pwm)
Input
(Sharp sensor)
Snelheid
STANLEY
8m/26'
Tylon
MAX
80
MPH
360000

Regeling (2)

meting
30 cm

Offset
150

Fout: 0

Links: 150

Rechts:150

Setpoint
30 cm

Gas (pwm)

Snelheid

Regeling (3)

meting
40 cm

Offset
150

Fout: 10

Links: 160

Rechts:140

Setpoint
30 cm

Gas (pwm)

Snelheid

Regeling (4)

meting
20 cm

Offset
150

Fout: -10

Links: 140

Rechts:160

Setpoint
30 cm

Gas (pwm)

Snelheid

Regeling (5)

Offset
150

Fout: 5

Versterking
7

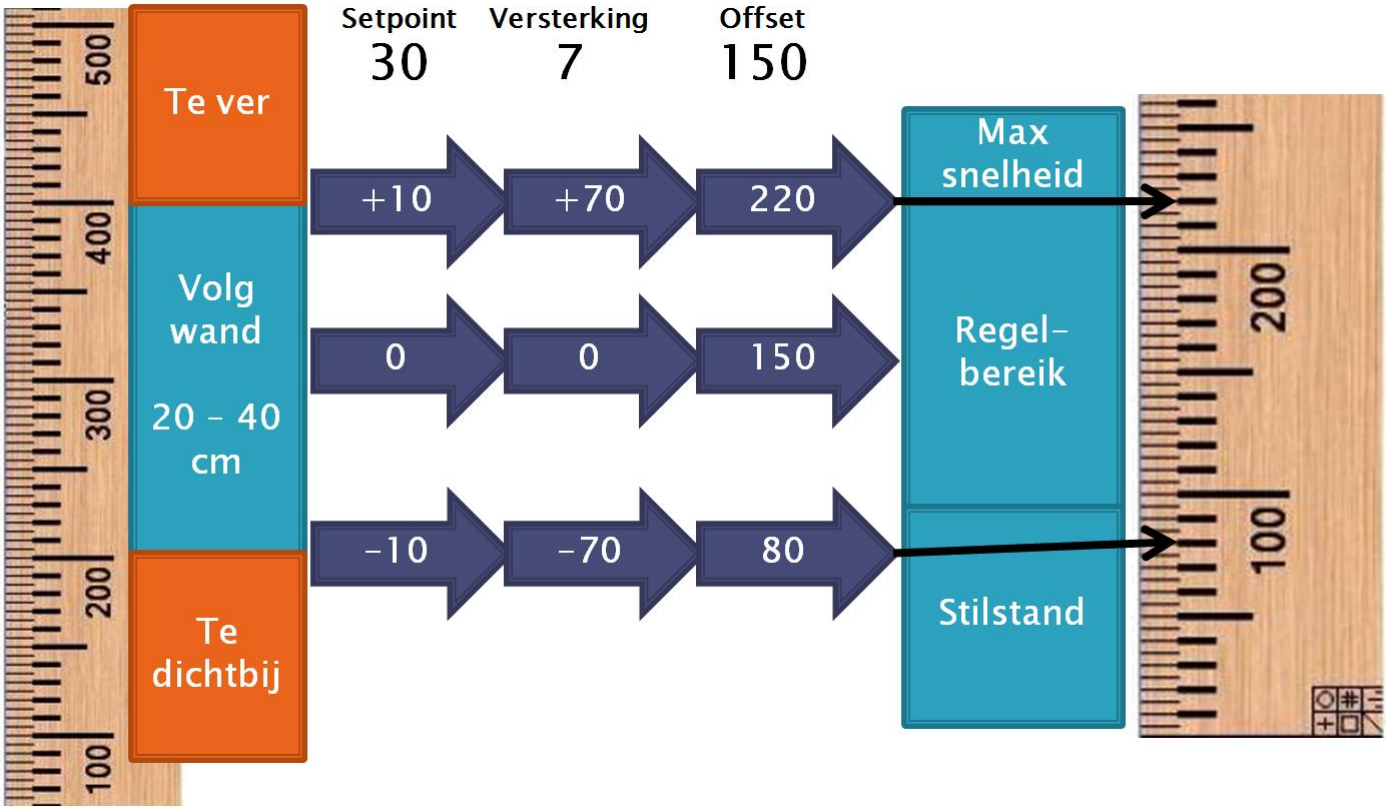
Fout: 35

Links: 185

Rechts:115

Regelbereik

- Afstand tussen 20 en 40 cm
- PWM tussen 80 en 220



Oefening Wandvolgen

- State machine; stop als afstand buiten regelbereik (tussen 20 en 40 cm)
- regel beide motoren (ene +, andere –)