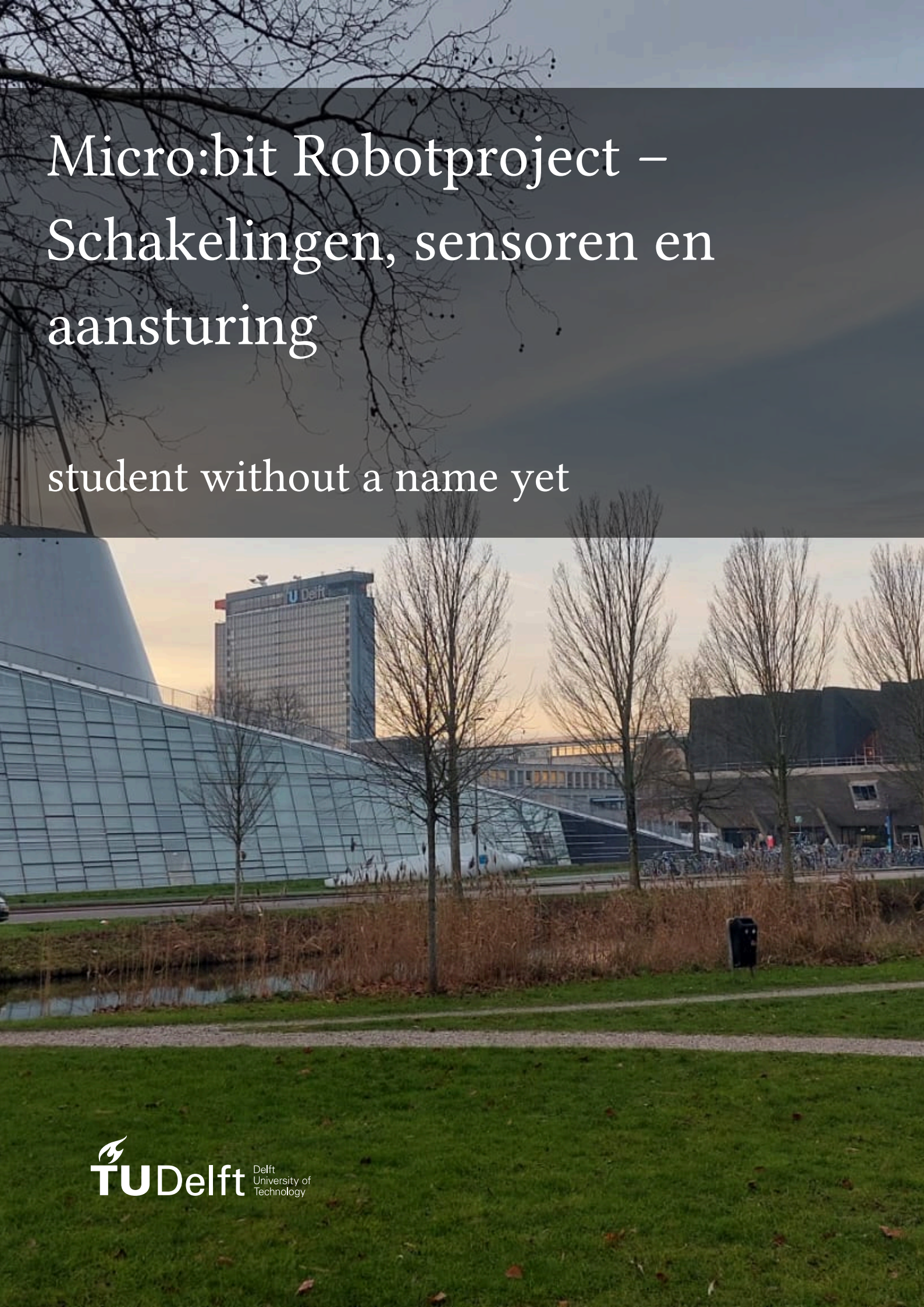
Micro:bit Robotproject – Schakelingen, sensoren en aansturing

student without a name yet



TU Delft Delft University of Technology

Micro:bit Robotproject – Schakelingen, sensoren en aansturing

by

student without a name yet

Supervisor: prof. dr. John Doe
Delft University of Technology
Committee: Prof. Dr. C. Examiner
Delft University of Technology
Dr. D. Examiner
Delft University of Technology

TU Delft

TU Delft

Publication date: January 31, 2026

Abstract

Keywords: Jupyter Book, Open Science, TU Delft, Starterkit, micro:bit

Contents

1. Gebruik van dit tekstboek

Dit online tekstboek ondersteunt je tijdens het micro:bit-robotproject bij het bouwen en testen van schakelingen met sensoren en actuatoren. Je vindt hier korte uitleg, voorbeeldschakelingen en Python-voorbeelden die je helpen wanneer je tijdens het project ergens vastloopt.

Gebruik dit boek als eerste hulpmiddel bij technische vragen, zodat je zelfstandig verder kunt werken aan je robot.

1.1 Hoe gebruiken we Python in dit robotproject?

In dit robotproject schrijven we Python-code om sensoren uit te lezen en actuatoren aan te sturen met de micro:bit. Om de code overzichtelijk en begrijpelijk te houden gebruiken we steeds dezelfde manier van programmeren.Zo kun je makkelijker fouten vinden en elkaars code begrijpen.

1.1.1 1. We definiëren pins bovenaan het programma

In plaats van direct `pin0` te gebruiken, geven we pins eerst een naam.Voorbeeld:

```
from microbit import *

LED_PIN = pin0
```

Waarom doen we dit? Omdat je dan meteen ziet waar de pin voor gebruikt wordt.

Vergelijk:

```
pin0.write_digital(1)
```

met:

```
LED_PIN.write_digital(1)
```

De tweede versie is duidelijker.

1.1.2 2. We gebruiken variabelen voor vaste waarden

Getallen zoals `1`, `0` of `77` zeggen op zichzelf weinig.

Daarom geven we ze een naam.

Voorbeeld:

```
LED_AAN = 1
LED_UIT = 0
```

Of bij een servo:

```
GRIJPER_OPEN = 26
GRIJPER_DICHT = 77
```

Nu wordt de code makkelijker te lezen:

```
SERVO_PIN.write_analog(GRIJPER_OPEN)
```

1.1.3 3. We zetten instellingen altijd bovenaan het programma

Bovenaan het programma staat dus:

- welke pins we gebruiken
- welke vaste waarden belangrijk zijn

Bijvoorbeeld:

```
from microbit import *
SERVO_PIN = pin0
GRIJPER_OPEN = 26
GRIJPER_DICHT = 77
```

Daaronder komt pas de rest van het programma.

1.1.4 4. We gebruiken duidelijke namen voor variabelen

Gebruik namen die uitleggen wat iets doet.

Goede voorbeelden:

```
MOTOR_PIN
LED_PIN
LICHT_SENSOR
AFSTAND_SENSOR
```

Minder goede voorbeelden:

```
pin
waarde
x
test
```

1.1.5 5. We werken met een vaste structuur in elk programma

Onze programma's hebben meestal deze opbouw:

```
from microbit import *

# instellingen

LED_PIN = pin0
LED_AAN = 1
LED_UIT = 0

# programma
while True:
  LED_PIN.write_digital(LED_AAN)
```

Hierdoor kun je snel zien:

- waar instellingen staan
- waar het programma begint
- wat de micro:bit doet

1.1.6 Waarom werken we zo?

Deze manier van programmeren helpt je om:

- minder fouten te maken
- sneller fouten te vinden
- duidelijkere programma's te schrijven
- beter samen te werken in je groep
- grotere programma's te begrijpen

Professionele programmeurs werken ook op deze manier.

2. Elektronica

Om een robot te bouwen met de micro:bit werk je met elektronische schakelingen. Daarin lopen elektrische stromen door draden en componenten. In dit hoofdstuk leer je hoe stroom werkt en welke rol **plus**, **min**, **weerstanden** en **transistoren** spelen.

2.1 Stroom: van plus naar min

Elektrische stroom kun je vergelijken met water dat door een slang stroomt. De batterij of voeding zorgt ervoor dat stroom kan bewegen.

In een schakeling geldt:

- stroom loopt **van plus naar min**
- er moet altijd een **gesloten kring** zijn
- zonder gesloten kring werkt een component niet

Bijvoorbeeld:

een led gaat alleen branden als:

plus arrow.r led arrow.r min

correct is aangesloten.

Als ergens de kring onderbroken is, gebeurt er niets.

De micro:bit heeft ook plus- en min-aansluitingen. Die herken je als:

- **3V** (plus)
- **GND** (min)

2.2 Weerstanden: bescherming en controle van stroom

Een weerstand zorgt ervoor dat er **niet te veel stroom** door een component loopt.

Zonder weerstand kan bijvoorbeeld:

- een led kapot gaan
- een sensor verkeerd werken
- een schakeling onbetrouwbaar worden

Een weerstand werkt dus als een soort **kraan in een waterleiding**: hij beperkt de hoeveelheid stroom.

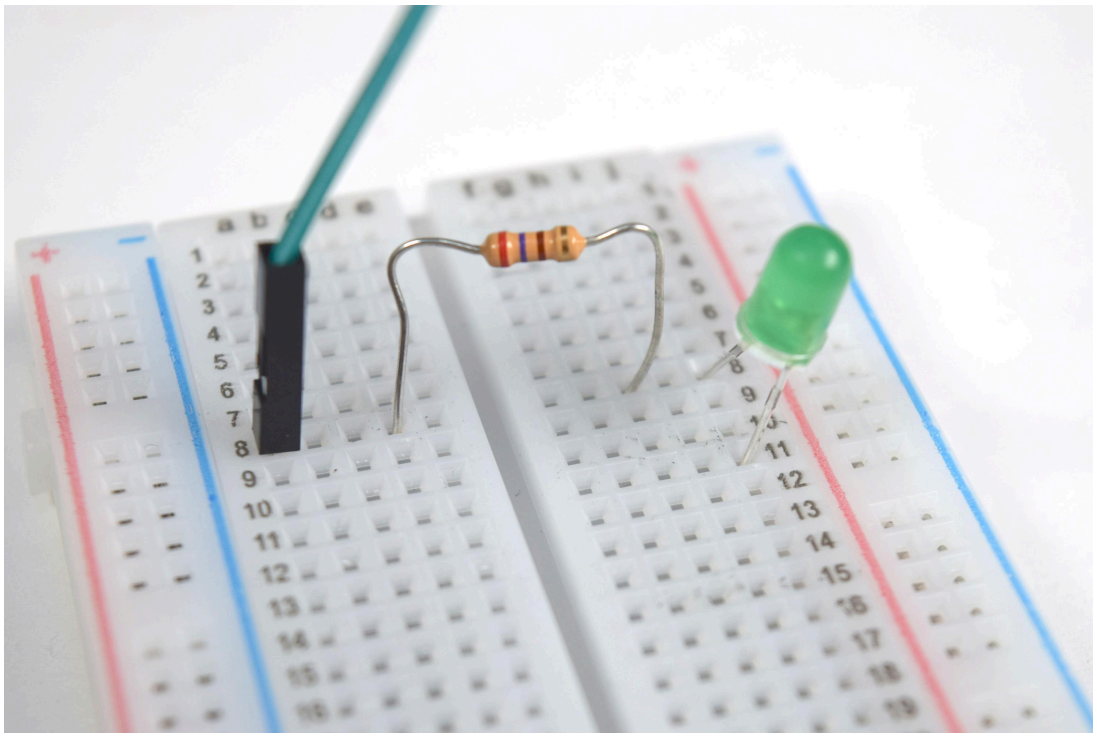
Daarom zie je vaak:

micro:bit arrow.r weerstand arrow.r led arrow.r GND

De weerstand beschermt hier de led.

Vuistregel in dit project:

gebruik altijd een weerstand bij een led.



2.3 Transistoren: schakelen met een klein signaal

Soms kan de micro:bit een onderdeel niet direct aansturen. Bijvoorbeeld:

- motor
- grijparm
- grotere actuator

Dat komt omdat deze onderdelen **meer stroom nodig hebben** dan de micro:bit veilig kan leveren.

Daarom gebruiken we een transistor.

Een transistor werkt als een **elektronische schakelaar**:

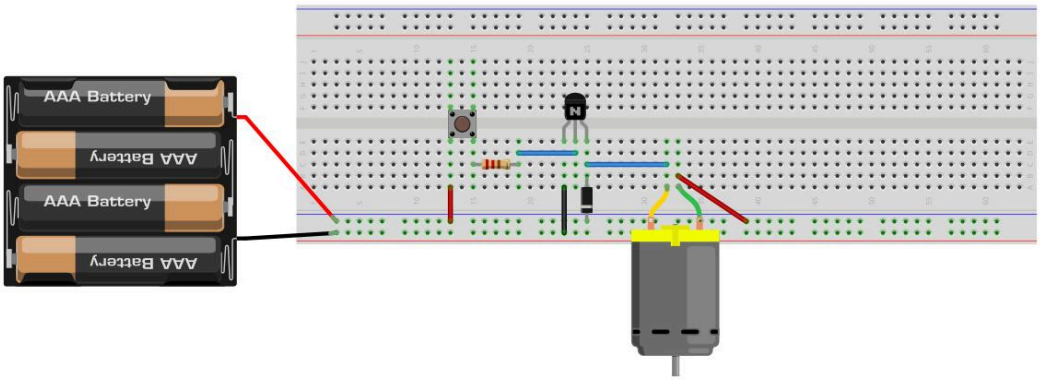
een klein signaal van de micro:bit
stuurt een grotere stroom aan uit een externe batterij.

Dus:

micro:bit arrow.r transistor arrow.r motor

De micro:bit bestuurt dan de transistor, en de transistor bestuurt de motor.

Zo bescherm je de micro:bit én kun je toch sterke onderdelen gebruiken.



2.3.1 De drie aansluitingen van een transistor

Een transistor heeft drie pootjes:

naam	functie
Base (B)	stuursignaal van de micro:bit
Collector (C)	verbonden met motor
Emitter (E)	verbonden met GND

De **base** bepaalt of de transistor aan of uit staat.

2.3.2 Hoe herken je de juiste kant van de transistor?

Bij de transistor die we in dit project gebruiken (bijvoorbeeld BC547):

- de platte kant zit aan de voorkant
- de pootjes wijzen naar beneden

Van links naar rechts zijn de pootjes:

Collector – Base – Emitter

Dus:

positie	aansluiting
links	Collector
midden	Base
rechts	Emitter

Let op: dit geldt alleen als de **platte kant naar je toe wijst**.

2.3.3 Hoe sluit je een transistor aan?

Sluit de transistor zo aan:

micro:bit pin arrow.r weerstand arrow.r Base Emitter arrow.r GND Collector arrow.r motor motor arrow.r batterij + batterij - arrow.r GND

De weerstand tussen pin en base beschermt de micro:bit.

2.3.4 Waarom zit er een weerstand bij de base?

De base mag niet rechtstreeks op de micro:bit worden aangesloten.

De weerstand:

- beschermt de micro:bit
- zorgt dat de transistor goed schakelt

Gebruik meestal een weerstand van ongeveer:

220Ω – 1kΩ

2.3.5 Wat gebeurt er als de transistor verkeerd om zit?

Dan gebeurt meestal:

- de motor draait niet
- of de schakeling werkt onbetrouwbaar

De transistor gaat meestal niet direct kapot, maar de schakeling werkt niet zoals bedoeld.

Controleer daarom altijd:

1. staat de platte kant in de juiste richting?
2. zitten de pootjes goed aangesloten?
3. zit er een weerstand tussen pin en base?
4. is GND goed verbonden?

2.4 Samenvatting

In dit project gebruik je drie belangrijke ideeën:

- stroom loopt alleen in een **gesloten kring**
- een **weerstand beschermt** componenten tegen te veel stroom
- een **transistor schakelt** grotere onderdelen veilig aan met behulp van de micro:bit

3. Werken met de pins van de micro:bit

De micro:bit heeft aansluitpunten (pins) waarmee je sensoren en onderdelen kunt **uitlezen (input)** of **aansturen (output)**. In deze sectie leer je:

- welke pins je kunt gebruiken
- wat het verschil is tussen digitaal en analoog
- hoe input en output werken
- hoe je pins gebruikt in Python

3.1 Wat is een pin?

Een pin is een aansluitpunt op de rand van de micro:bit waarmee je signalen kunt lezen of sturen.

Belangrijke pins in dit project zijn:

- **P0**
- **P1**
- **P2**
- **3V** (plus)
- **GND** (min)

Gebruik meestal:

- **3V** arrow.r voeding
- **GND** arrow.r terug naar min
- **P0** / **P1** / **P2** arrow.r signalen

3.2 Input en output

Pins kunnen twee dingen doen:

3.2.1 Input (invoer)

De micro:bit **meet** een signaal.

Bijvoorbeeld:

- knop
- lichtsensor
- afstandssensor

De micro:bit leest dan wat er gebeurt.

3.2.2 Output (uitvoer)

De micro:bit **stuurt** een signaal.

Bijvoorbeeld:

- led
- motor (via transistor)
- grijparm

De micro:bit bepaalt dan wat er gebeurt.

3.2.3 Digitale signalen

Digitale signalen kennen maar twee waarden:

- 0 = uit
- 1 = aan

Voorbeelden:

- knop ingedrukt of niet
- led aan of uit
- motor aan of uit

Python voorbeeld:

```
from microbit import *

pin0.write_digital(1)
```

Dit zet pin0 **aan**.

En:

```
from microbit import *

pin0.write_digital(0)
```

zet pin0 **uit**.

3.2.4 Analoge signalen

Analoge signalen kunnen **veel waarden tegelijk hebben**.

Bijvoorbeeld:

0 – 1023

Dat gebruik je bij sensoren zoals:

- lichtsensor
- draaiknop
- afstandssensor

Python voorbeeld:

```
from microbit import *

waarde = pin1.read_analog()
display.scroll(waarde)
```

3.2.5 Analoge output

De micro:bit kan ook analoge signalen sturen.

Gebruik dit bijvoorbeeld voor:

- motorsnelheid
- servo-aansturing
- helderheid van een led

```
from microbit import *

pin0.write_analog(512)
```

Waarden liggen tussen:

0 – 1023

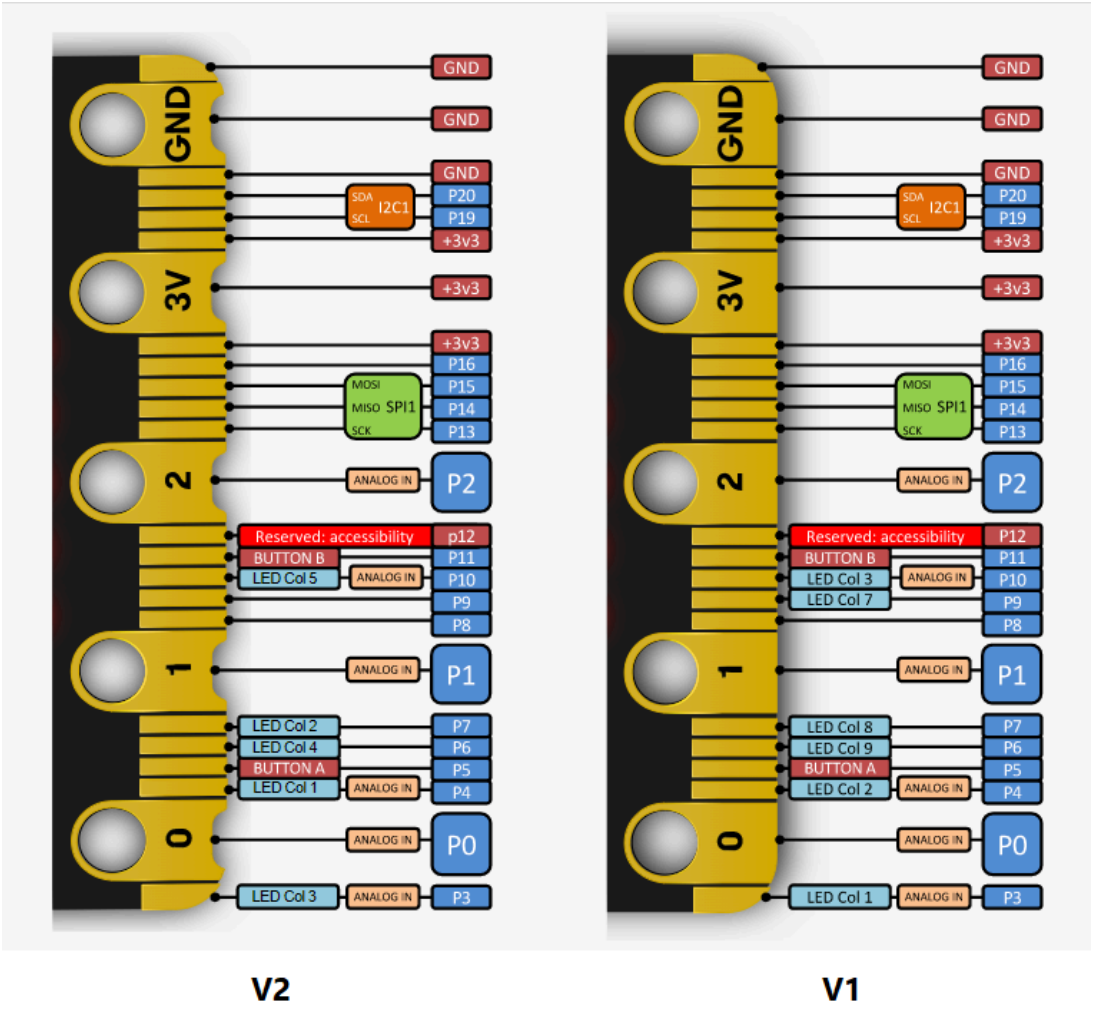
Dus:

- 0 = uit

- 1023 = maximaal

3.2.6 Welke pins zijn digitaal en analoog?

Het verschilt enigzins in de versie van de micro:bit welke pins je kan gebruiken.



Sommige pins hebben extra functies.

Bijvoorbeeld:

- verbonden met knoppen
- verbonden met led-matrix
- gebruikt voor communicatie

De micro:bit heeft een ingebouwd 5×5 led-scherm.

Dit scherm gebruikt intern ook pins.

Daardoor geldt:

sommige pins werken anders wanneer het scherm actief is.

Als je nauwkeurige metingen doet (bijvoorbeeld analoge sensoren), kan het handig zijn om het scherm uit te schakelen:

```
display.off()
```

3.2.7 Voorbeeld: knop als input en led als output

```
from  microbit  import  *

while  True:
  if  pin1.read_digital():
    pin0.write_digital(1)
  else:
    pin0.write_digital(0)
```

Hier gebeurt:

knop ingedrukt arrow.r led aan
knop los arrow.r led uit

4. Actuatoren aansturen met de micro:bit

Actuatoren zijn onderdelen die **iets laten bewegen of zichtbaar maken**.
In dit robotproject gebruiken we onder andere:

- leds (licht)
- motoren (beweging)
- grijparmen (servo)

De micro:bit stuurt actuatoren aan via pins zoals **P0, P1 en P2**.

4.1 Led aansturen

Een led is de eenvoudigste actuator. Hij kan alleen:

- aan
- uit

4.1.1 Hoe sluit je een led goed aan?

Een led werkt maar in **één richting**.
Daarom moet je goed opletten welke kant naar **plus** en welke kant naar **min (GND)** gaat.

Een led moet **altijd via een weerstand** worden aangesloten.
Zonder weerstand kan de led kapot gaan.

Aansluiten:

P0 arrow.r weerstand arrow.r led arrow.r GND

4.1.2 Welke kant van een led is plus en min?

Je kunt dit op drie manieren herkennen:

kenmerk	betekenis
lange poot	plus (anode)
korte poot	min (kathode)
platte rand van de led	min (kathode)

Vuistregel:

lange poot arrow.r via weerstand naar pin
platte kant arrow.r naar GND

Dus meestal sluit je een led zo aan:

pin arrow.r weerstand arrow.r lange poot led platte kant led arrow.r GND

4.1.3 Waarom moet een led de juiste richting hebben?

Een led is een **diode**.
Dat betekent dat stroom er maar in één richting doorheen kan lopen.

Als je hem omdraait:

- gaat de led niet branden
- maar meestal gaat hij niet kapot (als je een weerstand gebruikt)

4.1.4 Altijd een weerstand gebruiken

Een led moet **altijd via een weerstand** worden aangesloten.

Zonder weerstand kan:

- de led kapot gaan
- of de micro:bit beschadigd raken

Gebruik daarom altijd:

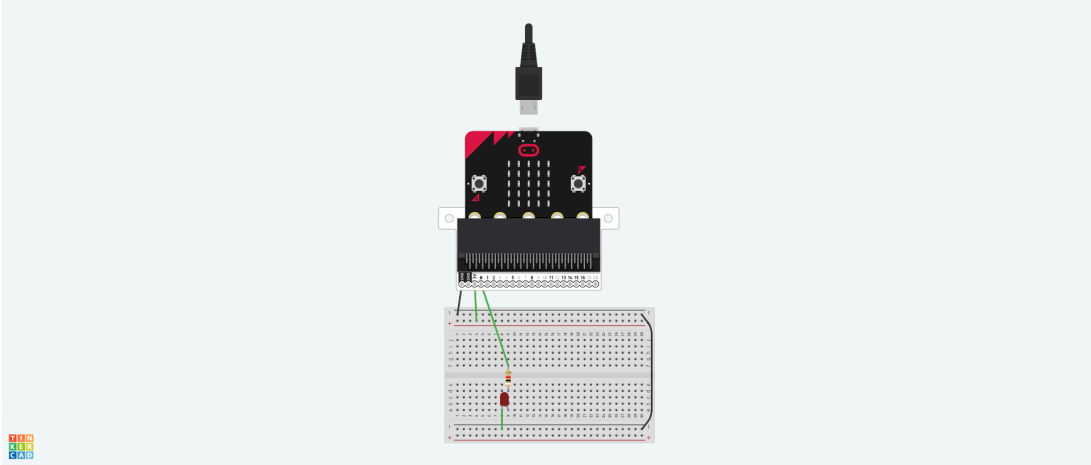
pin arrow.r weerstand arrow.r led arrow.r GND

4.1.5 Werkt de led niet?

Controleer dan:

1. zit de platte kant aan GND?
2. zit er een weerstand in de schakeling?
3. zit de led goed in het breadboard?
4. gebruik je de juiste pin in je programma?

In veel gevallen hoef je de led alleen even om te draaien.



4.1.6 Voorbeeldprogramma:

```
from microbit import *

# instellingen
LED_PIN = pin0

LED_AAN = 1
LED_UIT = 0


# programma
while True:
    LED_PIN.write_digital(LED_AAN)
```

Led bedienen met knop A en knop B:

```
from  microbit  import  *

# instellingen
LED_PIN  =  pin0

LED_AAN  =  1
```

```
LED_UIT  =  0


# programma
while  True:
  if  button_a.was_pressed():
    LED_PIN.write_digital(LED_AAN)
  if  button_b.was_pressed():
    LED_PIN.write_digital(LED_UIT)
```

4.2 Motor aansturen

Een motor heeft meer stroom nodig dan de micro:bit kan leveren.
Daarom gebruik je:

- een **transistor**
- een **externe batterij**

De micro:bit stuurt de transistor aan.
De transistor schakelt vervolgens de motor.

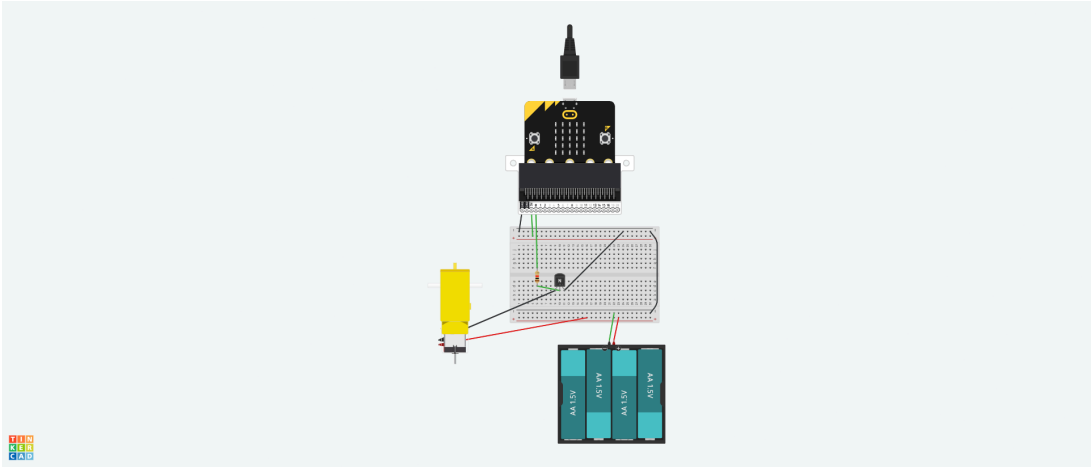
Aansluiten (vereenvoudigd):

micro:bit arrow.r transistor arrow.r motor arrow.r batterij

De base mag niet rechtstreeks op de micro:bit worden aangesloten.

De weerstand:

- beschermt de micro:bit
- zorgt dat de transistor goed schakelt



4.2.1 Voorbeeldprogramma

```
from microbit import *

# instellingen
MOTOR_PIN = pin0
MOTOR_AAN = 1
MOTOR_UIT = 0

# programma
```

```
while True:
  MOTOR_PIN.write_digital(MOTOR_AAN)
```

Motor snelheid regelen (PWM):

```
from microbit import *

# instellingen

MOTOR_PIN = pin0
MOTOR_LANGZAAM = 300
MOTOR_SNEL = 700

# programma
while True:
  MOTOR_PIN.write_analog(MOTOR_SNEL)
```

Waarden liggen tussen:

0 – 1023

Dus:

- lage waarde = langzaam
- hoge waarde = snel

4.2.2 Grijparm aansturen (servo)

Een grijparm wordt meestal bestuurd met een **servo-motor**.

Een servo kan draaien naar een **specifieke positie**, bijvoorbeeld:

- open
- dicht

Servo aansluiten:

rood arrow.r 3V
zwart arrow.r GND
geel arrow.r P0

Voorbeeldprogramma:

```
from microbit import *

# instellingen
SERVO_PIN = pin0
GRIJPER_OPEN = 26
GRIJPER_DICHT = 77

# programma
SERVO_PIN.write_analog(GRIJPER_OPEN)
```

Grijper bedienen met knop A en B:

```
from microbit import *

# instellingen

SERVO_PIN = pin0
```

```
GRIJPER_OPEN = 26
GRIJPER_DICHT = 77

# programma
while True:
  if button_a.was_pressed():
    SERVO_PIN.write_analog(GRIJPER_OPEN)
  if button_b.was_pressed():
    SERVO_PIN.write_analog(GRIJPER_DICHT)
```

5. Sensoren uitlezen met de micro:bit

Sensoren zijn onderdelen die informatie meten uit de omgeving. In dit robotproject gebruiken we sensoren om te bepalen wat de robot moet doen. Voorbeelden:

- lichtsensor arrow.r meet hoe licht of donker het is
- afstandssensor arrow.r meet hoe ver een object weg is

Sensoren sturen informatie **naar** de micro:bit. Dit noemen we **input**.

5.1 Digitale en analoge sensoren

Er zijn twee soorten signalen die sensoren kunnen geven.

5.1.1 Digitale sensoren

Digitale sensoren hebben maar twee waarden:

- 0 = geen signaal
- 1 = wel signaal

Bijvoorbeeld:een knop ingedrukt of niet.

5.1.2 Analoge sensoren

Analoge sensoren kunnen veel verschillende waarden geven. Bij de micro:bit liggen deze waarden tussen: 0 – 1023 Bijvoorbeeld:

- hoe licht het is
- hoe ver iets weg is De meeste sensoren in dit project gebruiken **analoge signalen**.

5.2 Lichtsensor uitlezen

Een lichtsensor meet hoeveel licht er op de sensor valt. Meer licht arrow.r hogere waarde
Minder licht arrow.r lagere waarde

Voorbeeldprogramma:

```
from microbit import *

# instellingen

LICHT_SENSOR_PIN = pin1

# programma
while True:
  lichtwaarde = LICHT_SENSOR_PIN.read_analog()
  display.scroll(lichtwaarde)
```

De micro:bit meet hier steeds hoeveel licht er binnenkomt en toont dit op het display.

5.2.1 Reageren op licht (beslissing maken)

Je kunt de robot laten reageren op licht.

Bijvoorbeeld:

bij weinig licht gaat een led aan.

```
from microbit import *

# instellingen

LICHT_SENSOR_PIN = pin1
LED_PIN = pin0

DONKER_DREMPEL = 400
LED_AAN = 1
LED_UIT = 0

# programma

while True:
  lichtwaarde = LICHT_SENSOR_PIN.read_analog()
  if lichtwaarde < DONKER_DREMPEL:
    LED_PIN.write_digital(LED_AAN)
  else:
    LED_PIN.write_digital(LED_UIT)
```

Hier bepaalt de variabele `DONKER_DREMPEL` wanneer het donker genoeg is.

5.3 Afstandssensor uitlezen

Een afstandssensor meet hoe ver een object van de robot staat.

Bijvoorbeeld:

- dichtbij obstakel
- ver weg van obstakel

Veel afstandssensoren geven een **analoge waarde** terug.

Voorbeeldprogramma:

```
from microbit import *

# instellingen

AFSTAND_SENSOR_PIN = pin2

# programma
while True:
  afstandwaarde = AFSTAND_SENSOR_PIN.read_analog()
  display.scroll(afstandwaarde)
```

Hoe dichter een object bij de sensor komt, hoe meer de waarde verandert.

Let op:

de exacte betekenis van de waarde hangt af van het type sensor.

5.3.1 Reageren op afstand

De robot kan stoppen wanneer een object te dichtbij komt.

```
from microbit import *

# instellingen
AFSTAND_SENSOR_PIN = pin2
MOTOR_PIN = pin0
OBSTAKEL_DREMPEL = 500
MOTOR_AAN = 1
MOTOR_UIT = 0

# programma
while True:
  afstandwaarde = AFSTAND_SENSOR_PIN.read_analog()
  if afstandwaarde > OBSTAKEL_DREMPEL:
    MOTOR_PIN.write_digital(MOTOR_UIT)
  else:
    MOTOR_PIN.write_digital(MOTOR_AAN)
```

Hier stopt de motor wanneer een obstakel dichtbij komt.

5.3.2 Sensorwaarden bekijken tijdens testen

Tijdens het bouwen is het handig om eerst alleen de sensorwaarde te bekijken.

Zo kun je bepalen welke drempelwaarde goed werkt:

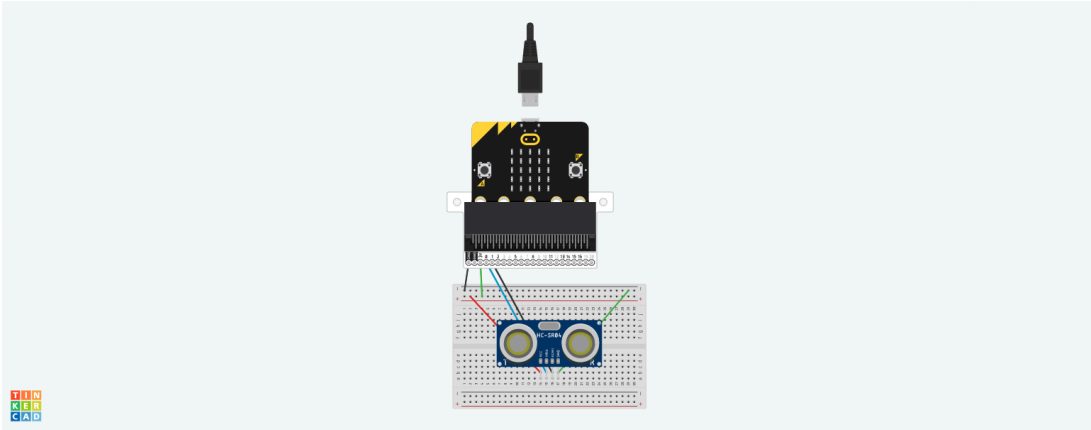
```
from microbit import *

# instellingen

SENSOR_PIN = pin1

# programma
while True:
  sensorwaarde = SENSOR_PIN.read_analog()
  display.scroll(sensorwaarde)
```

Test dit eerst voordat je beslissingen toevoegt aan je programma.

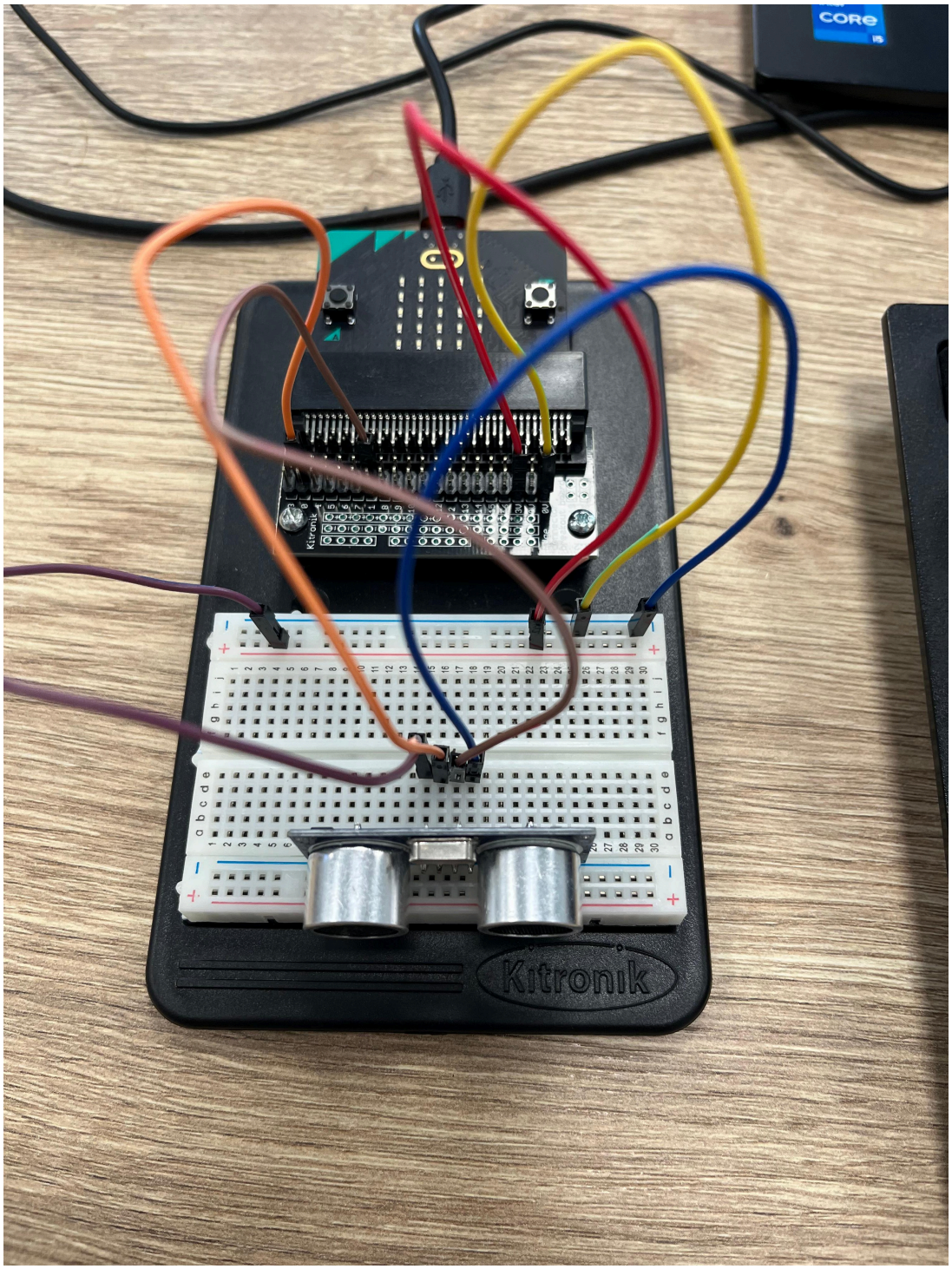
5.4 Ultrasone afstandssensor gebruiken

De ultrasone afstandssensor (met 4 pinnetjes) verzendt een ultrasoon geluid en meet de tijd totdat deze terugkomt. Door deze tijd om te rekenen kom je tot een aantal centimeters.

De 2 buitenste pinnen van de sensor zijn voor de GND en VCC, de binnenste voor de trigger (om het geluid te versturen) en de echo (om het geluid op te vangen).



```
from microbit import *
import machine
import utime

# Constants. Stel hier in welke pinnen je gebruikt.
SONAR_SIGNAL_PIN = pin0
SONOR_ECHO_PIN   = pin1

# Variablen

def get_afstand():
    # Berekent afstand in cm
    SONAR_SIGNAL_PIN.write_digital(0)   # Zet speaker sonar uit
    utime.sleep_us(2)                   # Wacht 2 microseconden
    SONAR_SIGNAL_PIN.write_digital(1)   # Zet speaker sonar aan (verzend
geluid)
    utime.sleep_us(10)                  # Laat het geluid aan voor 10
microseconden
    SONAR_SIGNAL_PIN.write_digital(0)   # Zet speaker sonar uit
    afstandtijd = machine.time_pulse_us(SONOR_ECHO_PIN,1,11600) # Meet tijd
tot geluid wordt gemeten
    afstand_cm = afstandtijd / 58       # Deel door 2 (heen en terug) en
door snelheid geluid
    return afstand_cm

# Hoofdprogramma
SONOR_ECHO_PIN.set_pull(SONOR_ECHO_PIN.NO_PULL)

while True:
    if button_a.was_pressed():        # als op knop A wordt gedrukt, gaan
we de afstand meten
        afstand = int(get_afstand())
        display.scroll(afstand)
```



5.5 PIR-sensor met externe voeding gebruiken

Sommige PIR-sensoren (zoals de HC-SR505) werken **niet altijd betrouwbaar op 3V**.
In dat geval kun je een **externe batterij** gebruiken.

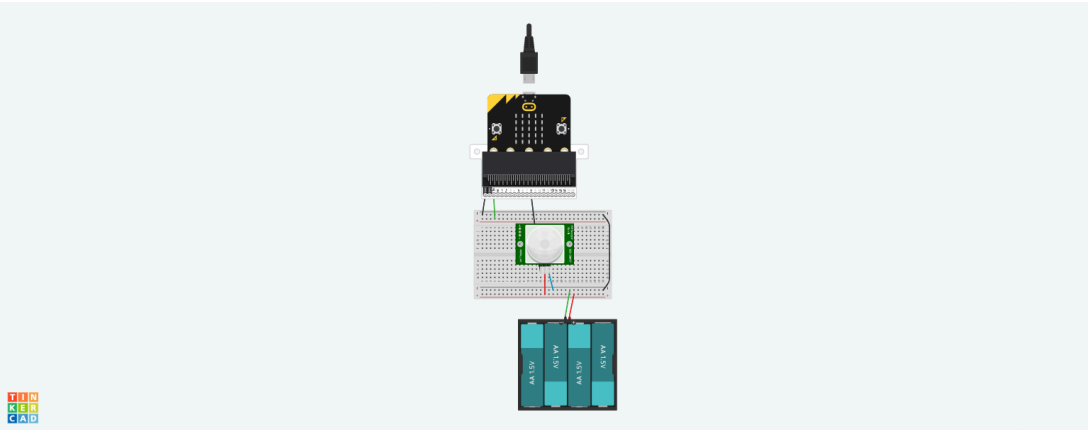
5.5.1 Aansluiten met externe voeding

Sluit de PIR-sensor zo aan:

VCC arrow.r externe batterij (+) (bijv. 3x AA ≈ 4.5V) GND arrow.r GND batterij én GND micro:bit OUT arrow.r P1

Belangrijk:

- De **GND van de batterij en de micro:bit moeten verbonden zijn**
- Anders kan de micro:bit het signaal niet goed lezen



5.5.2 Waarom moet GND gedeeld zijn?

De micro:bit meet spanning ten opzichte van GND.

Als de GND's niet verbonden zijn:

- krijgt de micro:bit geen stabiel signaal
- werkt de sensor niet goed

5.5.3 PIR uitlezen (zelfde code)

De code verandert niet als je externe voeding gebruikt:

```
from microbit import *

# instellingen
PIR_SENSOR_PIN = pin1

BEWEGING = 1
GEEN_BEWEGING = 0


# programma
while True:
    sensorwaarde = PIR_SENSOR_PIN.read_digital()

    if sensorwaarde == BEWEGING:
        display.show(Image.YES)
    else:
        display.clear()
```

Let op bij hogere spanning

De uitgang (OUT) van de PIR is meestal: ongeveer 3.3V (veilig voor micro:bit)

Daarom kun je deze meestal direct aansluiten op een pin.

Gebruik geen 5V direct op een micro:bit pin.

Wanneer gebruik je externe voeding?

Gebruik een externe batterij als:

- de sensor altijd 1 geeft
- de sensor niet reageert
- de sensor instabiel werkt

5.6 Kleurensensor VEML6040 gebruiken met de micro:bit

De VEML6040 RGBW Color Sensor Module is een kleurensensor die licht kan meten. De sensor kan vier verschillende waarden uitlezen:

- R = rood
- G = groen
- B = blauw
- W = wit / totale lichtsterkte

De sensor werkt via een communicatieprotocol dat we ook bij andere sensoren tegenkomen: I2C.

Met deze sensor kun je bijvoorbeeld:

- kleuren herkennen
- een lijnvolgende robot maken
- lichtintensiteit meten
- een sorteersysteem bouwen
- onderzoeken hoe RGB-kleuren werken

5.6.1 Aansluiten van de sensor

De sensor heeft vier aansluitingen, die je op de micro:bit kunt koppelen:

- VIN/VCC –> 3V (dus geen 6V!)
- GND –> GND
- SDA –> P20 (aan de zijkant van het breadboard)
- SCL –> P19 (aan de zijkant van het breadboard)

SDA en SCL zijn de twee datalijnen van I2C. We gebruiken hiervoor dus 2 speciale pinnen, P19 en P20. Let er op dat de gaatjes van deze pinnen, en de gaatjes van de sensor, wat groter zijn dan je gewend bent, de stekkertjes blijven hierdoor minder goed vast zitten.

5.6.2 Hoe werkt I2C?

Bij I2C praten apparaten met elkaar via twee draadjes:

- SDA arrow.r data
- SCL arrow.r kloksignaal

De micro:bit is hierbij de “baas” (master). De sensor luistert als “slave”.

Iedere I2C-sensor heeft een eigen adres. De VEML6040 gebruikt standaard adres: 0x10 (hexadecimaal)

5.6.3 Controleren of de sensor werkt

Voordat we waarden uitlezen, controleren we eerst of de micro:bit de sensor ziet.

```
from microbit import *

while True:
```

```
    print(i2c.scan())
    sleep(1000)
```

Zie je: [16], dan is de verbinding goed. Dit is namelijk gelijk aan 0x10. Hexadecimale getallen worden vaak gebruikt bij I2C-apparaten.

5.6.4 Sensor inschakelen

De VEML6040 staat standaard soms in een soort slaapstand. Daarom moeten we hem eerst aanzetten.

```
i2c.write(0x10, bytes([0x00, 0x00, 0x00]))
```

5.6.5 RGBW-waarden uitlezen

De sensor bewaart de meetwaarden in zogenaamde registers. We lezen telkens 2 bytes uit een register.

Register	Betekenis
0x08	Red
0x09	Green
0x0A	Blue
0x0B	White

```
from microbit import *

ADDRESS = 0x10

def read16(register):

    i2c.write(ADDRESS, bytes([register]))
    data = i2c.read(ADDRESS, 2)

    value = data[0] | (data[1] << 8)

    return value

# VEML6040 inschakelen
# Register 0x00 = config
# 0x00, 0x00 betekent: sensor aan, normale meting
i2c.write(ADDRESS, bytes([0x00, 0x00, 0x00]))

sleep(100)

devices = i2c.scan()
display.scroll(str(devices))

while True:
    red   = read16(0x08)
    green = read16(0x09)
    blue  = read16(0x0A)
    white = read16(0x0B)

    print("R:", red,
```

```
            "G:", green,
            "B:", blue,
            "W:", white)

    sleep(500)
```

5.6.6 Wat betekenen de waarden?

De R, G en B geven de rode, groene en blauwe waarde aan van het licht. Je kunt kleuren herkennen door waarden met elkaar te vergelijken. Hoe hoger een bepaalde waarde, hoe meer de kleur die richting in gaat.

5.6.7 Omgevingslicht

De sensor meet licht. Dus:

- zonlicht beïnvloedt de meting
- afstand maakt uit
- schaduw beïnvloedt de waarden

Daarom krijg je niet altijd exact dezelfde waarden.

5.6.8 De sensor zelf

De echte sensor is het kleine zwarte chipje midden op het printplaatje.

Richt dat chipje naar het object dat je wilt meten.

Geen waarden? Of zie je alleen nullen?

Controleer:

- juiste pinnen?
- P19 en P20 niet omgedraaid?
- GND aangesloten?
- sensor aangezet?
- voldoende licht?

6. Communiceren met twee micro:bits via radio

Je kunt twee micro:bits met elkaar laten communiceren via radiosignalen. In dit robotproject gebruiken we:

- één micro:bit als afstandsbediening
- één micro:bit op de robot

Beide micro:bits krijgen **exact dezelfde programmacode**. Daardoor hoef je maar één programma te onderhouden.

6.1 Hoe werkt dit?

We gebruiken twee micro:bits:

micro:bit	functie
micro:bit 1	afstandsbediening
micro:bit 2	robotbesturing

Wanneer je op knop A drukt:

- stuurt de micro:bit een radiosignaal
- voert de micro:bit zelf ook dezelfde actie uit

De andere micro:bit ontvangt het radiosignaal en voert dezelfde actie uit. Zo kun je de robot:

- rechtstreeks bedienen
- én via een afstandsbediening

met hetzelfde programma.

6.2 Radio aanzetten

Om radio te gebruiken moet je eerst de radio-module importeren en inschakelen.

```
from microbit import *
import radio

# instellingen

RADIO_GROEP = 7
radio.on()
radio.config(group=RADIO_GROEP)
```

De `RADIO_GROEP` zorgt ervoor dat micro:bits alleen luisteren naar berichten binnen dezelfde groep.

Gebruik voor jouw robotgroep dus steeds hetzelfde groepsnummer.

6.3 Signalen afspreken

We spreken vaste berichten af.

```
RIJD_VOORUIT = "vooruit"
STOP = "stop"
```

Deze berichten sturen we via radio.

6.4 Eén actie uitvoeren op twee manieren

De robot moet vooruit rijden als:

- knop A wordt ingedrukt
- of het radiosignaal `"vooruit"` wordt ontvangen

Dat schrijven we zo:

```
from microbit import *
import radio

# instellingen

RADIO_GROEP = 7
LINKER_MOTOR_PIN = pin0
RECHTER_MOTOR_PIN = pin1
MOTOR_AAN = 1
MOTOR_UIT = 0
RIJD_VOORUIT = "vooruit"
STOP = "stop"
radio.on()
radio.config(group=RADIO_GROEP)

# functies

def rij_vooruit():
  LINKER_MOTOR_PIN.write_digital(MOTOR_AAN)
  RECHTER_MOTOR_PIN.write_digital(MOTOR_AAN)

def stop_robot():
  LINKER_MOTOR_PIN.write_digital(MOTOR_UIT)
  RECHTER_MOTOR_PIN.write_digital(MOTOR_UIT)
# programma

while True:
  bericht = radio.receive()
  if button_a.was_pressed() or bericht == RIJD_VOORUIT:
    radio.send(RIJD_VOORUIT)
    rij_vooruit()

  if button_b.was_pressed() or bericht == STOP:
    radio.send(STOP)
    stop_robot()
```

Hier gebeurt dus hetzelfde bij:

```
button_a.was_pressed()
```

en bij:

```
bericht == RIJD_VOORUIT
```

Dat betekent:

> lokale knop = radiosignaal = dezelfde actie

6.5 Waarom gebruiken we dezelfde code?

Beide micro:bits krijgen dezelfde code.

Dat heeft voordelen:

- je hoeft maar één programma te onderhouden
- je kunt de robot direct bedienen
- je kunt de robot ook via afstandsbediening bedienen
- je hoeft niet steeds te onthouden welke micro:bit welke versie van de code heeft

De micro:bit op de robot reageert dus op:

- eigen knoppen
- radiosignalen van de afstandsbediening

De afstandsbediening stuurt signalen, maar kan dezelfde actie ook lokaal uitvoeren.
Als daar geen motor op aangesloten is, gebeurt er lokaal gewoon niets zichtbaars.