



Datapunt 3G – Fysiek technisch product

Studenten nummer:2152618 - Stamgroep:1A - Jaar: 2025

Leeruitkomsten

BC 3.2.1 Je ontwikkelt een werkend technisch prototype waarmee je de basisprincipes van programmeren kunt aantonen.

BC 3.2.2 Je maakt in je uitleg navolgbaar dat je de basisprincipes van programmeren begrijpt.

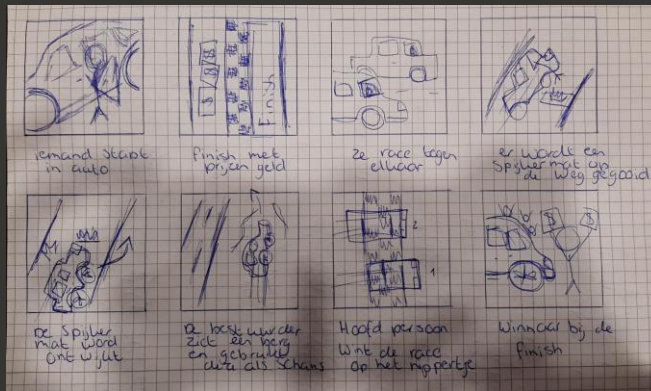
BC 3.2.1

Inspiratie

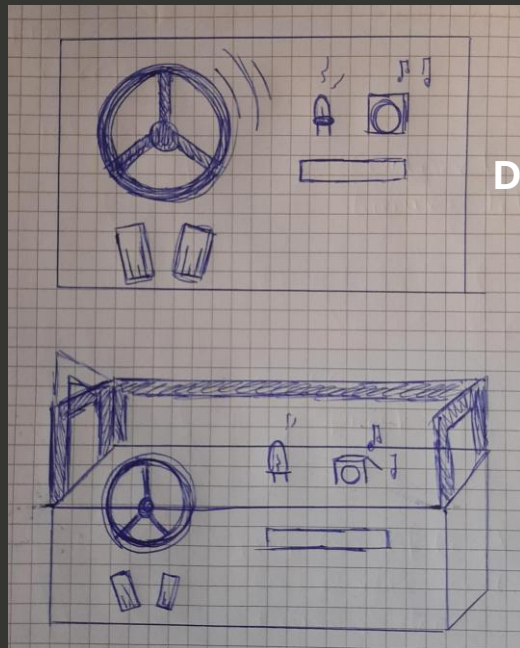


BC 3.2.1

Met mijn **scratch** verhaal **als basis** (dit was een race game met straat race als verhaallijn) heb ik besloten om een **interactief auto dashboard** te maken.



Schetsen

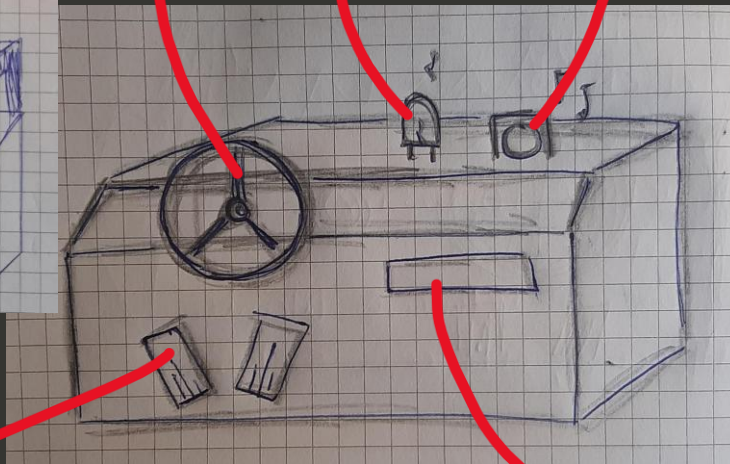


Gaspedaal

Lampje

Draaiend stuur

Speaker

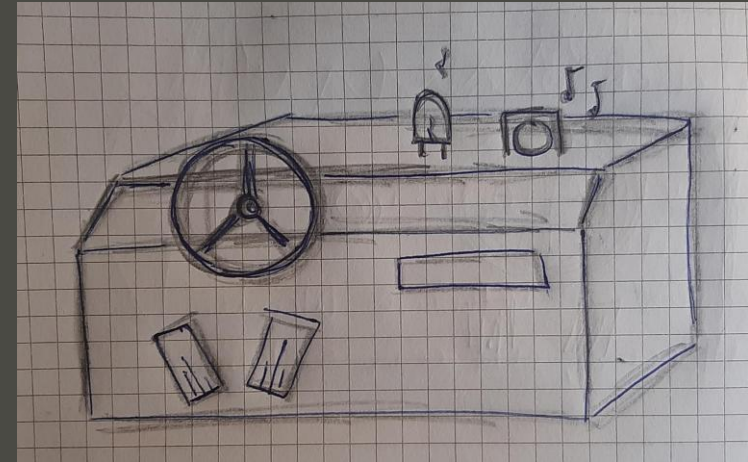


CD speler

BC 3.2.1

BC 3.2.2

Benodigheden



Draaiend stuur → *Output*

-Draait door Servo krijgt input door gaspedaal

Gaspedaal → *Input*

-Geeft input via button

Lampje → *Output*

-Reageert op input van gaspedaal

Speaker → *Output*

-Reageert op beweging in de CD speler

CD speler → *Input*

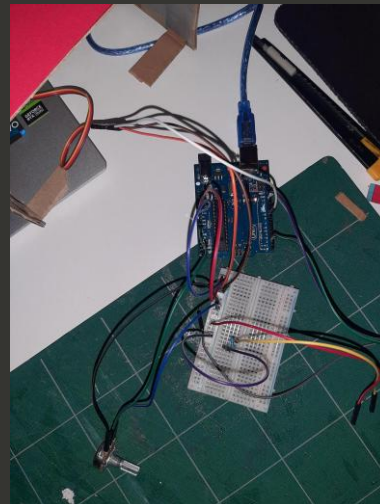
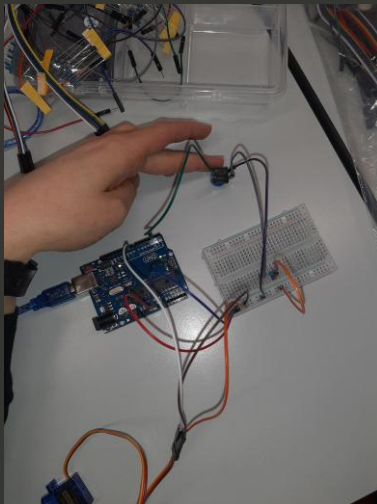
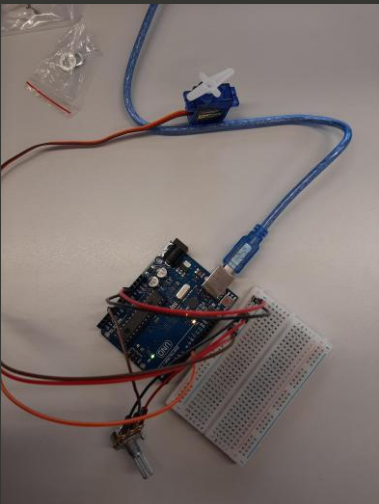
-Krijgt input door ultrasonic sensor

Testen

1 Voor dat ik het “dashboard” ben gaan bouwen ben ik eerst de verschillend in-put out-put mogelijkheden gaan bekijken. Om zo te zien wat ik met wat kan doen.

2 Ik onder vond toen al snel dat de cd-speler die ik wou gaan maken erg lastig zou zijn en dat ik ook niet precies wist hoe ik dit zou moeten gaan bouwen.

3 Ik heb toen besloten om het cd-speler idee los te laten en in plaats daarvan een soort speakertje te maken waar uit het geluid komt.



In elkaar zetten

4 Vervolgens ben ik de buitenkant gaan bouwen uit een schoenendoos. Ik heb eerst de afmetingen bepaald en het toen alles in elkaar gezet met lijm en duct-tape.

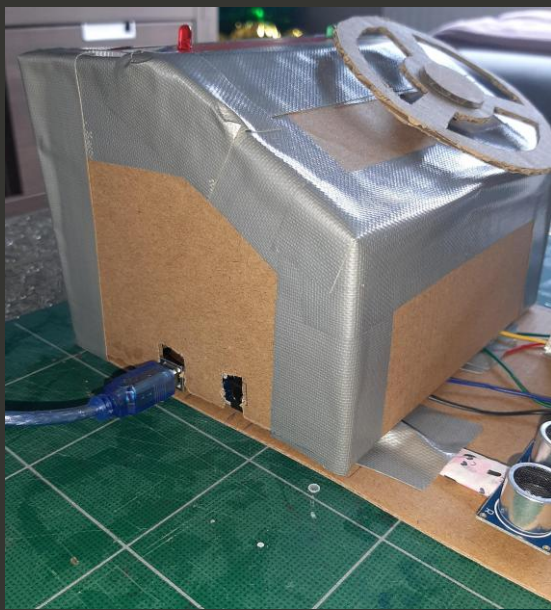
5 Toen ik de Arduino in het “dashboard” wou plaatsen kwam ik er achter dat de bedrading erg chaotisch was. Ik heb deze toen volledig opnieuw gedaan en heb alle draden toen met washi-tape aan elkaar verbonden en vast gemaakt.

6

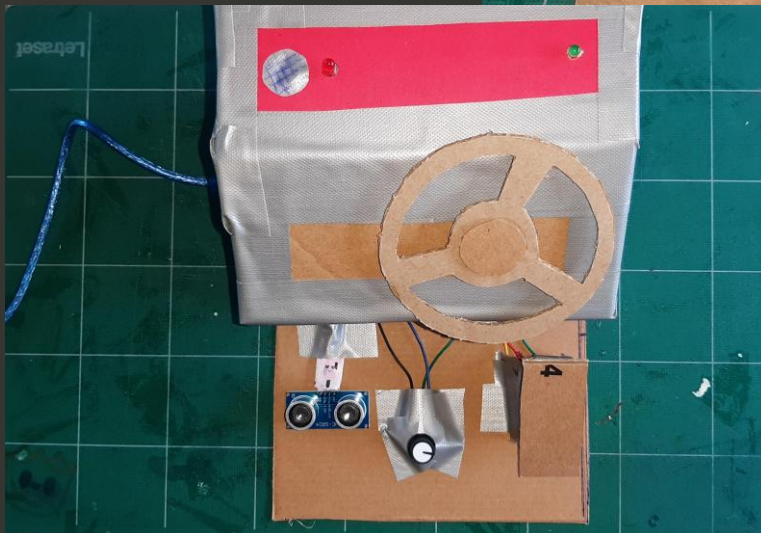
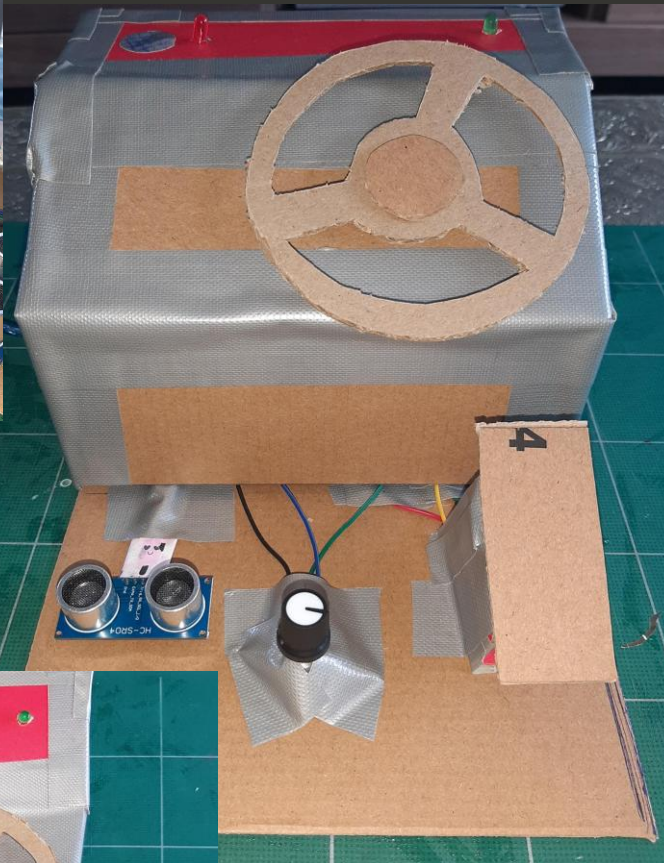


Daarna ben ik de Arduino en het breadboard in de binnenkant van het kartonnen ‘dashboard’ gaan vast maken.

BC 3.2.1



7 De buitenkant van het 'dashboard' in elkaar zetten.



BC 3.2.1

BC 3.2.2

Pseudocode

Gaspedaal:

- 1 Als het gaspedaal (Button) wordt ingedrukt
- 2 Dan gaat een van de lampjes aan
- 3 Na 2 seconden gaat het lampje uit
- 4 Als je nog een keer het gaspedaal (Button) indrukt
- 5 Dan gaat het andere lampje aan
- 6 Na 2 seconden gaat het lampje uit
- 7 Anders wanneer er niks wordt ingedrukt
- 8 Dan gaat er geen lampje aan

Stuur:

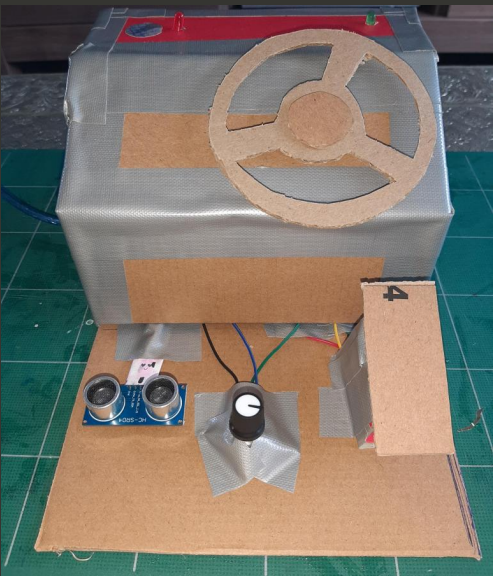
- 1 Als je aan het knopje draait (potentiometer)
- 2 Dan beweegt het stuur (Servo)
- 3 Anders wanneer er niet wordt gedraaid
- 4 Dan beweegt er niks

Ultrasonic sensor:

- 1 Als je je hand boven de sensor houdt (op $< 3\text{cm}$ afstand)
- 2 Dan speelt er een piepjes af
- 3 Anders wanneer je buiten bereik bent
- 4 Dan blijft het stil

Buitenkant verbeteren

Om het echt op een autodashboard te laten lijken heb ik aan de buitenkant nog wat dingen toegevoegd die een echte auto ook heeft.



Inspiratie



BC 3.2.1

Prototype in werking

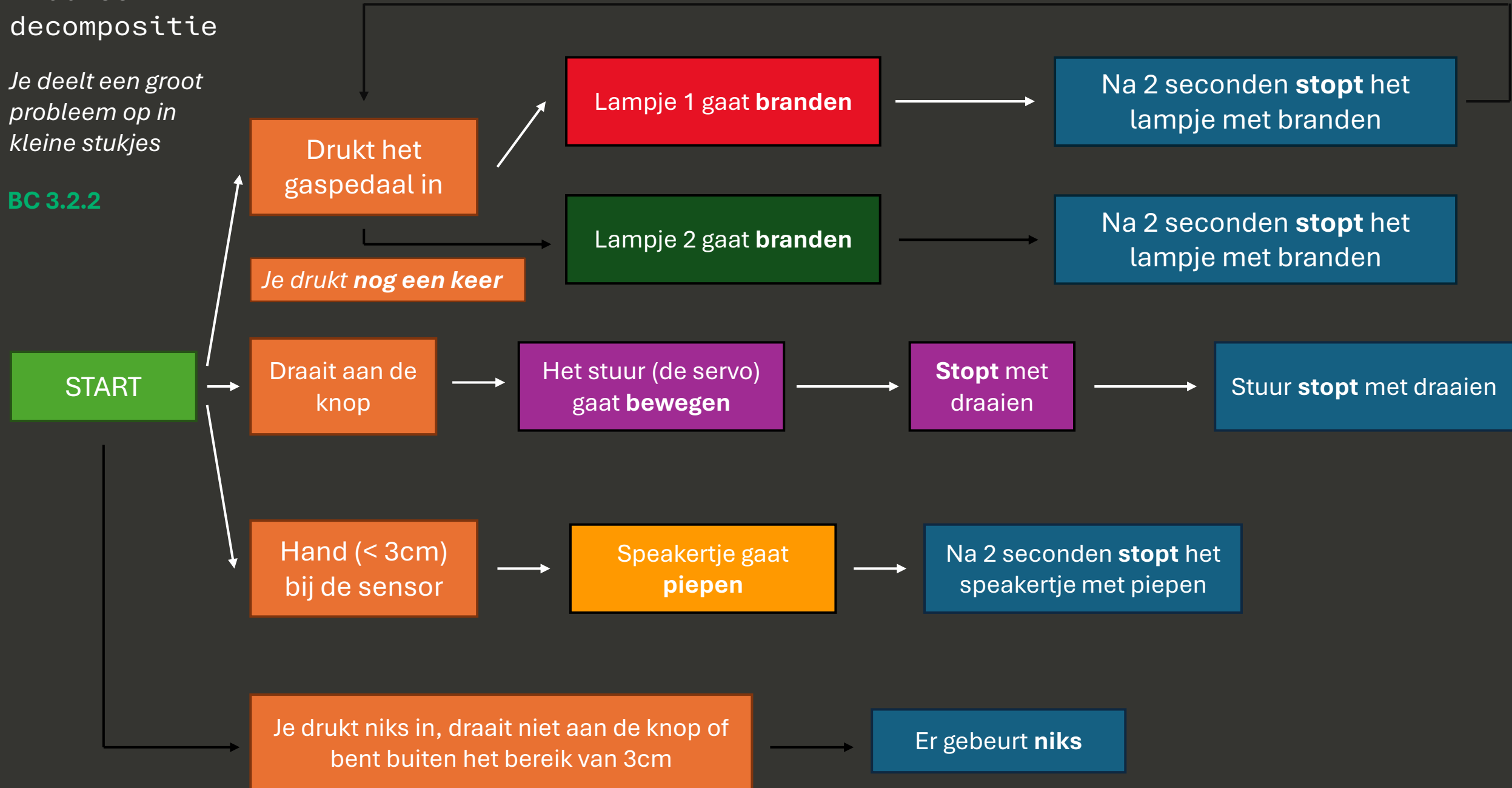
**Filmpje staat in de ZIP map



Probleem decompositie

*Je deelt een groot
probleem op in
kleine stukjes*

BC 3.2.2



Algoritme

Een stappenplan dat wordt gebruikt om een specifieke taak uit te voeren of problemen op te lossen.

```
// === LED aansturen via knop ===
buttonState = digitalRead(buttonPin); //Leest de status

if (buttonState == HIGH && lastButtonState == LOW) { //
  digitalWrite(ledPin1, LOW);
  digitalWrite(ledPin2, LOW);

  if (currentLed == 0) {
    digitalWrite(ledPin1, HIGH);
    delay(700);
    digitalWrite(ledPin1, LOW);
    currentLed = 1; //Wanneer knop wordt ingedrukt branden
  } else {
    digitalWrite(ledPin2, HIGH);
    delay(700);
    digitalWrite(ledPin2, LOW);
    currentLed = 0; //Wanneer de knop nóg een keer wordt
  }
  delay(50);
}
```

Dit is het algoritme voor de button en de 2 LED pins

Dit algoritme **leest de status** van de button (of hij wel of niet wordt ingedrukt). Wanneer deze **wordt ingedrukt** gaat er een **lampje branden** voor 700ms. Hierna **gaat hij uit** als je dan **nog een keer klikt** gaat het **andere** lampje branden voor de zelfde tijd. Als je niet drukt gaat er ook **niks** branden.

Algoritme voor buzzer

```
if (distance < 3) { //Afstand kleiner dan 3cm
  digitalWrite(buzzerPin, HIGH);
  delay(400); //De buzzer piept voor 400ms
  digitalWrite(buzzerPin, LOW);
}
```

Algoritme voor potentiometer om Servo aan te sturen

```
// === Servo aansturen via potentiometer ===
val = analogRead(potPin);
val = map(val, 0, 1023, 0, 180); //De gelezen
myServo.write(val);

Serial.print("Potentiometerwaarde: ");
Serial.println(val);
```

Algoritme voor buzzer en sensor

```
void loop() {
  // === Ultrasoon sensor met buzzer === Hiermee v
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

Statement

Een enkele uitvoerbare instructie

```
Serial.print("Afstand: "); 1
Serial.print(distance); 2
Serial.println(" cm"); //Zorgt 3
```

Je ziet in dit screenshot **3 verschillende statements** die allemaal een **enkele uitvoerbare instructie** zijn.

```
// Ultrasoon sensor - Besch
const int trigPin = 6; 1
const int echoPin = 7; 2
const int buzzerPin = 4; 3
```

Je ziet in dit screenshot **3 verschillende statements** iedere regel die eindigt met ; is een actie die wordt uitvoert dus een statement.

Conditioneel statement

*Een statement met een bepaalde voorwaarde.
Het stukje code controleert of de voorwaarde
waar of niet is en reageert hierop.*

```
if (distance < 3) { //Afstand kleiner dan 3cm
  digitalWrite(buzzerPin, HIGH);
  delay(400); //De buzzer piept voor 400ms
  digitalWrite(buzzerPin, LOW);
}
```

Je ziet in dit screenshot een **conditioneel statement**. Er wordt gecontroleerd of er wordt voldaan aan een bepaald statement. **In dit geval of de sensor een echo binnen 3cm ontvangt.** Wanneer deze **word ontvangen** (Het statement is dus waar) word de **status HIGH geprint** en piept de buzzer voor 400ms. Wanneer de 400ms klaar zijn word de **status LOW geprint** en stopt het piepen. Wanneer er geen echo is binnen 3cm gebeurt er niks omdat de **voorwaarde niet waar is**.

Variabele

Een stukje code dat een waarde opslaat

```
// LED en button pins - Beschrijving
const int buttonPin = 2;
const int ledPin1 = 12;
const int ledPin2 = 13;
int buttonState = 0; //Maakt de variabele
int lastButtonState = 0;
int currentLed = 0;
```

```
// Ultrasonic en buzzer setup
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(buzzerPin, OUTPUT);
```

In deze screenshots zie je **verschillende variable**. Op het eerste screenshot worden er verschillende waardes (**Const en int**) toegekend aan LEDS en buttons. Op het tweede screenshot worden er verschillende waardes (**INPUT en OUTPUT**) toegekend aan verschillende pins van de ultrasonic sensor.

Operator

Een bewerking die uitgevoerd wordt op een variable

```
// == Servo aansturen via potentiometer
val = analogRead(potPin);
val = map(val, 0, 1023, 0, 180);
```

Er word een functie aan geroepen

```
if (currentLed == 0) {
    digitalWrite(ledPin1, HIGH);
}
```

Hier word een vergelijking (==) gemaakt

```
if (distance < 3) {
    digitalWrite(buzzer, HIGH);
}
```

Een kleiner dan (<) vergelijking

Met operators kun je **rekenen of vergelijkingen** maken met **variabele of waardes**. In deze screenshots zie je hier verschillende voorbeelden van.

Val is een **variabele**

Data types

Data Types geven aan tot welk soort gegevens een waarde behoort

```
#include <Servo.h>
```

Een object uit de servo bibliotheek

```
Servo myServo; //M
```

```
// LED en button pins - E
```

```
const int buttonPin = 2;
```

```
const int ledPin1 = 12;
```

```
const int ledPin2 = 13;
```

Een constante **integer** die niet mag veranderen

```
long duration = pulseIn(echoPin, HIGH);
```

```
int distance = duration * 0.034 / 2;
```

Groter gehele getal

```
Serial.print("Afstand: ");
```

```
Serial.print(distance);
```

```
Serial.println(" cm"); //7
```

Dit zijn 2 **string** datatypes dit zijn **tekst elementen** en herken je doordat ze tussen enkele of dubbele **aanhaling tekens** staan.

Loop

Een stuk code dat wordt herhaald totdat aan een bepaalde voorwaarde wordt voldaan

```
void loop() {
  // === Ultrasoon sensor met buzzer === Hiermee
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH); //Hiermee
  int distance = duration * 0.034 / 2;

  Serial.print("Afstand: ");
  Serial.print(distance);
  Serial.println(" cm"); //Zorgt ervoor dat er g

  if (distance < 3) { //Afstand kleiner dan 3cm
    digitalWrite(buzzerPin, HIGH);
    delay(400); //De buzzer piept voor 400ms
    digitalWrite(buzzerPin, LOW);
  }
}
```

De loop voor de buzzer en ultrasonic sensor

****Dit een maar een gedeelte van de loop. Alles in deze void loop word oneindige herhaald.**