

Datapunt 3G

Jochem Bakker
2153973
Stamgroep 1A
2024-25

Omvat:

- Demonstratie van prototype
- Uitleg en onderbouwing van de code

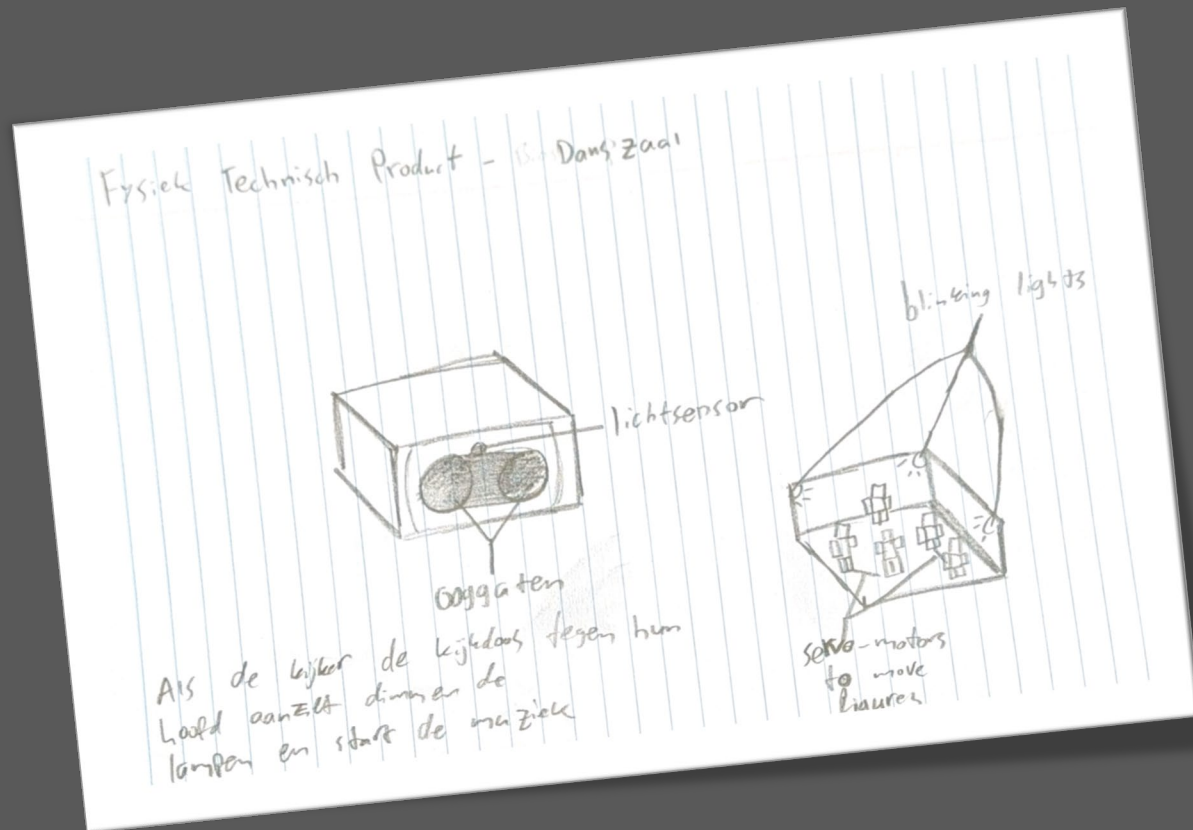
Bijhorende Leeruitkomsten:

LUK 3.2 De student betreft medestudenten om zich een beeld te vormen van haar ontwikkeling als ontwerper en kan deze inzichten omzetten naar persoonlijke leerdoelen.

- **BC 3.2.1** Je ontwikkelt een werkend technisch prototype waarmee je de basisprincipes van programmeren kunt aantonen [Ontwerpen]
- **BC 3.2.2** Je maakt in je uitleg navolgbaar dat je de basisprincipes van programmeren begrijpt. [Ontwerpen]

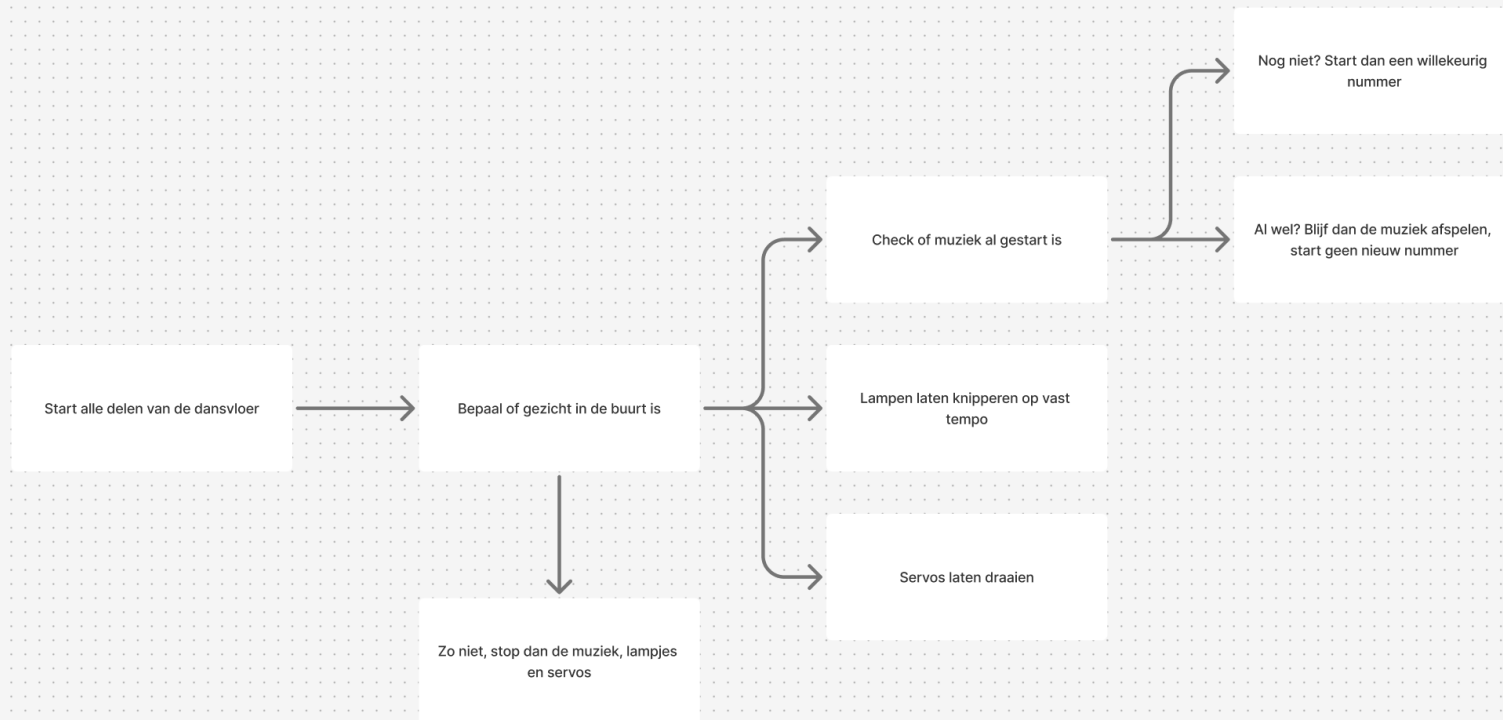
Daar waar het begrip van Leeruitkomsten (LUK's) wordt bewezen door middel van het behalen van Beoordelingscriteria (BC's) wordt de BC aangegeven tussen haakjes.

Prototype concept



Mijn Prototype zou bestaan als een interactieve kijkdooz van een danszaal. Zodra iemand erin zou kijken hoorde de muziek te starten, de lampen te knipperen en de servos waar de karakters op staan te draaien.

Probleemdecompositie

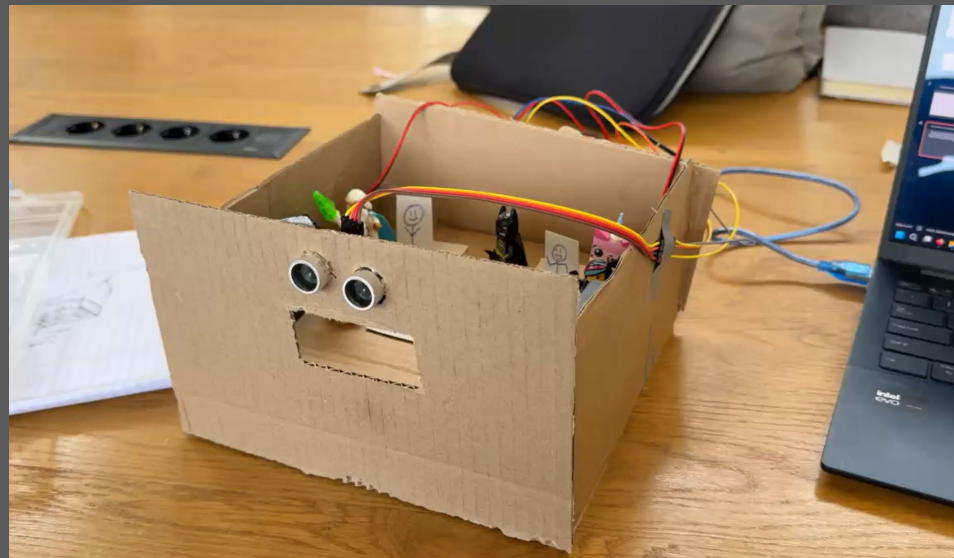


De probleemdecompositie hielp om vast te stellen en te begrijpen welke stappen elkaar moesten opvolgen.

Demonstratie van prototype (BC 3.2.1)

Tijdens het maken van het prototype liep ik tegen meerdere obstakels aan. Hierdoor bevat mijn uiteindelijke prototype de code voor de muziek en de lampjes, en alleen de muziek komt ook daadwerkelijk de zaal in. De servos kreeg ik los van de muziek en lampjes te werken, maar ze draaide niet meer toen alle drie bij elkaar kwamen, dus heb ik ze ook uit de code gelaten. De code voor de lampjes is nog wel geladen maar ik ben er niet meer aan toegekomen om de lampjes de zaal in te krijgen.

[Klik hier om de demonstratie te bekijken](#)



Uitleg en onderbouwing van code – Variabelen (BC 3.2.2)

Variabelen zijn waardes die zullen veranderen terwijl de code zijn gang gaat. Ik gebruik variabelen onder andere om de waardes bij te houden voor de lengte van een ultrasonische pulse (duration) en de afstand van een object tot de ultrasonische sensor die daarmee wordt berekend (distance).

Ook gebruik ik een variabele om bij te houden of er al een nummer wordt afgespeeld. Sinds deze variabele maar twee waardes nodig heeft, *true* of *false*, gebruik ik hiervoor een boolean variabele.

```
// create variables to hold the calculations for duration measured by Ultrasonic sensor and the distance calculated from that
float duration, distance;

bool songPlaying = false; // Variable to check whether a song is already playing, used to prevent skipping when the code loops
```

Uitleg en onderbouwing van code – Operatoren (BC 3.2.2)

Operatoren laten zien welke wiskundige acties worden ondernomen in de code. Ik gebruik Operatoren in mijn Conditionele Statements (hierover later meer informatie), bijvoorbeeld om een variabele met een constante te vergelijken om te bepalen of een object dichtbij genoeg is. Ook wordt een Operator gebruikt om de waarde van de Boolean *songPlaying* te vergelijken.

```
if (distance < detectFace && (songPlaying == false)) {
```

Ook gebruik ik Operatoren om het laatste moment dat de LED's knipperden te updaten nadat een update is gemaakt. Deze waarde wordt opgeslagen in *previousLedMillis*. Door de LED Interval op te tellen bij de vorige waarde van *previousLedMillis* zal de volgende loop deze nieuwe waarde gebruiken in de code er net boven.

```
if (LedState == LOW) { // if the Led is off, wait for the LED Interval to pass before turning it on
  if (currentMillis - previousLedMillis >= LedInterval) {
    LedState = HIGH;
    previousLedMillis += LedInterval; // save the time the change was made so the loop can run again
```

Uitleg en onderbouwing van code – Conditionele Statements (BC 3.2.2)

Conditionele Statements zorgen ervoor dat een stuk code alleen wordt uitgevoerd als een bepaalde conditie wordt voldaan.

```
if (distance < detectFace) {  
    digitalWrite(LedPin, LedState);  
} else {  
    digitalWrite(LedPin, LOW);  
}
```

Zo gebruik ik hier een Conditioneel Statement om de lampjes alleen te updaten met de correcte waarde als een object dichtbij de ultrasonische sensor is

Hier gebruik ik een Conditioneel Statement waar meerdere variabelen worden gecontroleerd om te bepalen wat gedaan moet worden. Als een object dichtbij is en er geen nummer afspeelt wordt een willekeurig nummer gekozen om af te spelen.

Is een object dichtbij en speelt er al een nummer af? Dan is de loop herhaald terwijl dezelfde persoon nog steeds kijkt, doe dus niks en blijf het nummer afspelen.

Is er geen object dichtbij en speelt er een nummer af? De persoon heeft hun gezicht weggetrokken, stop met het afspelen van muziek.

```
if (distance < detectFace && (songPlaying == false)) {  
    songPlaying = true; // start playing song when face is detected  
    myMP3.play(random(1, 5));  
    delay(1000);  
} else if (distance < detectFace && (songPlaying == true)) {  
    songPlaying = true; // do nothing if these conditions are met, just keep playing the song  
} else if (distance > detectFace && (songPlaying == true)) {  
    songPlaying = false; // stop playing when face is pulled away  
    myMP3.stop();  
}
```

Uitleg en onderbouwing van code – Data Types (BC 3.2.2)

Data Types zijn de benamingen voor verschillende soorten data dat zich in de code kan bevinden.

```
int const trigPin = 11;  
int const echoPin = 10;  
int const LedPin = 13;  
int sensorValue = 0;  
int detectFace = 10; //
```

Zo bestaan *integers* altijd uit volledige nummers, dus gebruik ik *floats* voor variabelen die mogelijk met decimalen moeten worden bewaard voor berekeningen

```
float duration, distance;
```

Ook gebruik ik een *string* om een stuk tekst naar de Serial Monitor te printen.

```
Serial.println("Afstand tot sensor");  
Serial.println(distance); // Print de
```


Uitleg en onderbouwing van code – Loops (BC 3.2.2)

```
void loop() {  
  
    currentMillis = millis(); // capture t  
  
    digitalWrite(trigPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(trigPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW);  
  
    duration = pulseIn(echoPin, HIGH);  
  
    distance = (duration * .0343) / 2;  
  
    Serial.println("Afstand tot sensor");  
    Serial.println(distance); // Print de  
  
    playMusic();  
    updateLEDs();  
    switchLEDs();  
}
```

```
void updateLEDs() {  
    if (LedState == LOW) { // if the Led is off, wait for the L  
        if (currentMillis - previousLedMillis >= LedInterval) {  
            LedState = HIGH;  
            previousLedMillis += LedInterval; // save the time the  
        }  
    } else {  
        if (currentMillis - previousLedMillis >= blinkDuration) {  
            LedState = LOW;  
            previousLedMillis += blinkDuration; // save the time th  
        }  
    }  
}  
  
void switchLEDs() {  
    if (distance < detectFace) {  
        digitalWrite(LedPin, LedState);  
    } else {  
        digitalWrite(LedPin, LOW);  
    }  
}
```

In mijn void loop wordt alleen de afstand tot de ultrasonische sensor gemeten. Deze waarde wordt tijdens elke stap van het proces gebruikt, maar de rest van de stappen staan in aparte functies die achter elkaar worden geroepen in de void loop.

Ik heb geen andere loops gebruikt in mijn code, maar dat had ik kunnen doen als ik de stappen op een andere manier wilde zetten om mijn doel te behalen.

```
void playMusic() {  
  
    if (distance > 0) {  
        if (distance < detectFace && (songPlaying == false)) {  
            songPlaying = true; // start playing song when face is dete  
            myMP3.play(random(1, 5));  
            delay(1000);  
        } else if (distance < detectFace && (songPlaying == true)) {  
            songPlaying = true; // do nothing if these conditions are n  
        } else if (distance > detectFace && (songPlaying == true)) {  
            songPlaying = false; // stop playing when face is pulled av  
            myMP3.stop();  
        }  
    }  
}
```

Zo had ik bijvoorbeeld mijn ultrasonische meting in een while loop kunnen zetten om ervoor te zorgen dat de andere functies niet eens gecalled werden totdat ze nodig waren.

```
while (distance < detectFace) {  
    digitalWrite(trigPin, LOW);  
    delayMicroseconds(2);  
    digitalWrite(trigPin, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(trigPin, LOW);  
}
```

Uitleg en onderbouwing van code – Algoritmen (BC 3.2.2)

Een Algoritme is een collectie stappen die de code neemt om een resultaat te bereiken. De hele code is één groot Algoritme om de dansvloer te activeren als iemand de kijkdoos in kijkt, maar hij bestaat ook uit kleinere algoritmen die hun eigen resultaat creëren die wordt gebruikt door andere delen.

```
digitalWrite(trigPin, LOW);  
delayMicroseconds(2);  
digitalWrite(trigPin, HIGH);  
delayMicroseconds(10);  
digitalWrite(trigPin, LOW);  
  
duration = pulseIn(echoPin, HIGH);  
  
distance = (duration * .0343) / 2;
```

Zo is dit het algoritme waarmee ik de afstand van een object tot de ultrasonische sensor meet. Eerst creëert de Trigger Pin een korte puls die door de door de Echo Pin wordt opgevangen. De lengte van die puls wordt vervolgens opgeslagen in een variabele die weer wordt gebruikt om daaruit de afstand tot de sensor te berekenen.