

BASISPRINCIPES PROGRAMMEREN

FYSIEK TECHNISCH PRODUCT

3.2.2 Basisprincipes programmeren

In deze uitleg zal ik het gebruik van de basisprincipes van programmeren bespreken aan de hand van de aangepaste code voor het interactieve fysieke prototype. Het systeem maakt gebruik van een servo om een prullenbak deksel te openen en te sluiten, waarbij een aantal visuele en auditieve indicaties worden gegeven door LED's en een buzzer. De code maakt gebruik van sensoren (input) en actuatoren (output) om de interactie te realiseren.

Basisprincipes van Programmeren

1. Probleemdecompositie

Het probleem is opgedeeld in kleinere taken. De sensor meet de afstand tot een object, en op basis daarvan wordt de servo gestuurd om het deksel van de prullenbak te openen of te sluiten. Daarnaast worden er visuele en audio elementen toegevoegd via de LED's en de buzzer. Door het systeem op deze manier op te splitsen, kan elke component eenvoudig worden beheerd en getest.

2. Variabelen (afbeelding 5)

Ik gebruik verschillende variabelen om belangrijke waarden op te slaan, zoals de afstand en de duur van de echopuls van de ultrasone sensor.

- **long distance = 100;** De gemeten afstand.
- **long duration;** De duur van de echo puls die wordt gebruikt om de afstand te berekenen. Deze variabelen zorgen ervoor dat ik de resultaten van de sensor kan opslaan en gebruiken in de verdere logica van de code.
- Daarnaast worden er variabelen voor de pinnen van de LED's en buzzer gedefinieerd (bijvoorbeeld **const int greenLed = 6;**), zodat het makkelijk is om deze te wijzigen als dat nodig is.

```
3  Servo servo1;
4  const int trigPin = 9;
5  const int echoPin = 8;
6  const int greenLed = 6; // Groene LED voor open-indicatie
7  const int redLed = 5;   // Rode LED voor sluit-indicatie
8  const int whiteLed = 4; // Witte LED voor normale positie
9  const int buzzer = 3;   // Buzzer voor geluidsindicatie
10 long distance = 100;
11 long duration;
```

Afbeelding 5: Variabelen code

3. Operatoren (afbeelding 6)

De code maakt gebruik van de vermenigvuldigings en delingsoperatoren om de afstand te berekenen met de ultrasone sensor:

- **distance = duration * 0.034 / 2;**

Hier wordt de duur van de echopuls omgezet in een afstand. De snelheid van geluid (34 cm) wordt vermenigvuldigd met de tijd en gedeeld door 2 om de afstand van het object naar de sensor te berekenen.

```
52 duration = pulseIn(echoPin, HIGH, 30000);
53 if (duration > 0) {
54     distance = duration * 0.034 / 2;
55 } else {
56     distance = 100;
57 }
58 }
```

Afbeelding 6: Operatoren code

4. Conditionele Statements (afbeelding 7)

if en **else statements** worden gebruikt om de logica van het systeem te sturen. De belangrijkste voorwaarde is of de gemeten afstand kleiner dan of gelijk aan 20 cm is:

- **if (distance > 0 && distance <= 20) {**
- **moveServo(180); // Open de prullenbak**
- **delay(5000); // Wacht 5 seconden**
- **} else {**
- **moveServo(90); // Sluit de prullenbak**
- **}**

Als de afstand kleiner dan of gelijk aan 20 cm is, wordt de servo naar 180° gestuurd om het deksel te openen. Daarna wordt er 5 seconden gewacht voordat het systeem weer terugkeert naar de normale positie. Als de afstand groter is dan 20 cm, wordt het deksel gesloten door de servo naar 90° te bewegen.

```
30
31 if (distance > 0 && distance <= 20) {
32     digitalWrite(whiteLed, LOW); // Zet de witte LED uit wanneer de servo beweegt
33     digitalWrite(greenLed, HIGH); // Zet de groene LED aan wanneer de servo opent
34     moveServo(180); // Beweeg naar 180°
35     delay(5000); // Blijf 5 seconden op 180°
36     digitalWrite(greenLed, LOW); // Zet de groene LED uit na 5 seconden
37 } else {
38     moveServo(90); // Beweeg terug naar 90°
39     digitalWrite(whiteLed, HIGH); // Zet de witte LED weer aan bij normale positie
40 }
```

Afbeelding 7: Conditionele Statements code

5. Data Types (afbeelding 8&9)

In deze code zijn de juiste datatypes gebruikt om verschillende soorten gegevens op te slaan:

- **long** voor de variabelen **distance** en **duration**, omdat deze grote waarden moeten kunnen opslaan (bijvoorbeeld voor het meten van de afstand met de sensor).
- **int** voor de servo hoeken en pininstellingen, omdat de hoeken en pinnen binnen het bereik van een integer vallen.

```
3  Servo servo1;
4  const int trigPin = 9;
5  const int echoPin = 8;
6  long distance = 100;
7  long duration;
```

Afbeelding 8: Data Types code

```
47 if (currentAngle < targetAngle) {
48     for (int pos = currentAngle; pos <= targetAngle; pos++) {
49         servol.write(pos);
50         delay(20);
51     }
52 } else if (currentAngle > targetAngle) {
53     for (int pos = currentAngle; pos >= targetAngle; pos--) {
54         servol.write(pos);
55         delay(20);
56     }
57 }
```

Afbeelding 9: Data Types code

6. Loops (afbeelding 10)

De **loop()** functie wordt voortdurend uitgevoerd, waardoor het systeem continu reageert op veranderingen in de omgeving. Binnen de **loop()** worden de sensormetingen uitgevoerd en worden de servo en andere componenten gestuurd op basis van de gemeten waarde:

- **void loop() {**
- **ultra(); // Meet de afstand**
- **if (distance > 0 && distance <= 20) {**
- **digitalWrite(whiteLed, LOW); // Zet de witte LED uit**
- **digitalWrite(greenLed, HIGH); // Zet de groene LED aan**
- **moveServo(180); // Open de prullenbak**
- **delay(5000); // Wacht 5 seconden**
- **} else {**
- **moveServo(90); // Sluit de prullenbak**
- **digitalWrite(whiteLed, HIGH); // Zet de witte LED weer aan**
- **}**
- **delay(100); // Korte vertraging om het systeem stabiel te houden**
- **}**

De **loop()** zorgt ervoor dat de interactie steeds opnieuw wordt uitgevoerd, afhankelijk van de afstand die door de sensor wordt gemeten.

```
27
28 void loop() {
29     ultra();
30
31     if (distance > 0 && distance <= 20) {
32         digitalWrite(whiteLed, LOW); // Zet de witte LED uit wanneer de servo beweegt
33         digitalWrite(greenLed, HIGH); // Zet de groene LED aan wanneer de servo opent
34         moveServo(180); // Beweeg naar 180°
35         delay(5000); // Blijf 5 seconden op 180°
36         digitalWrite(greenLed, LOW); // Zet de groene LED uit na 5 seconden
37     } else {
38         moveServo(90); // Beweeg terug naar 90°
39         digitalWrite(whiteLed, HIGH); // Zet de witte LED weer aan bij normale positie
40     }
41
42     delay(100);
43 }
```

Afbeelding 10: Loops code

7. Functies

moveServo(int targetAngle): Deze functie zorgt ervoor dat de servo soepel beweegt van de ene hoek naar de andere. Als de servo naar een grotere hoek moet bewegen, wordt het in stappen van 1 graad gedaan, en hetzelfde geldt voor het terugbewegen naar een kleinere hoek. Er worden visuele (LED) en auditieve (buzzer) feedbacksignalen gegeven bij de verschillende hoeken. De buzzer maakt een geluid bij het openen (180°) en vlak voor het sluiten (90°), terwijl de LED's de status van de prullenbak aangeven.

8. Systemen

LED's: Er worden drie LED's gebruikt om de status van de prullenbak aan te geven:

- **Groene LED** (aan als de prullenbak open is),
- **Rode LED** (aan tijdens het sluiten van de prullenbak),
- **Witte LED** (aan wanneer de prullenbak in de normale, gesloten positie is).

Buzzer: De buzzer geeft een geluid af als de prullenbak opent of sluit, en een waarschuwingsgeluid net voor het sluiten. De combinatie van visuele en auditieve feedback maakt het voor de gebruiker duidelijk in welke fase van de interactie het systeem zich bevindt, wat de gebruikerservaring verbetert.

Conclusie

De code maakt gebruik van veel programmeerprincipes, zoals probleemdecompositie, conditionele statements, loops, variabelen, operatoren, en functies. Het gebruik van feedbacksystemen via LED's en een buzzer verhoogt de interactie en maakt de werking van het prototype visueel en auditief duidelijk voor de gebruiker. De code is ontworpen op een manier die het mogelijk maakt om zijn prototype te testen en de werking ervan gemakkelijk aan te passen indien nodig. De principes die in deze code zijn toegepast tonen mijn begrip van de basisprincipes van programmeren en hoe ze kunnen worden gebruikt in een fysiek prototype.