Bijlage 2: Praktische voorbeelden

Lijnsensor

Voor het volgen van de zwarte lijn van het parcours maakt de Asuro Robot gebruik van optische sensoren. Deze bestaan uit twee fototransistoren (T9 en T10), en een rode LED (D11). De hoeveelheid stroom die door een fototransistor gaat, wordt bepaald door de hoeveelheid opvallend licht. Als de Asuro Robot de kleurbaan volgt (in dit geval de donkere lijn op een lichte ondergrond) dan gebruikt de robot deze sensoren.

De LED straalt licht naar de ondergrond (het parcours) dat door de fototransistor wordt opgevangen. De zwarte lijn die gevolgd moet worden, reflecteert veel minder licht dan de witte achtergrond. Boven een witte ondergrond zal er meer stroom door de transistor lopen dan boven een zwarte. Zo kan via de sensor informatie worden verkregen over de kleur van de ondergrond; op basis van deze informatie worden de motoren aangestuurd.
In een ideale situatie zijn de fototransistoren evenredig opgesteld ten opzichte van de lijn van het parcours, waarbij de reflectie voor beide transistoren gelijk is, waardoor de robot een rechte lijn nauwkeurig kan volgen, met een minimale afwijkingen.

De robot dient ook bochten te kunnen nemen, waarbij het voor kan komen dat één van de fototransistoren zich net boven de donkere gebogen lijn bevindt. Daardoor ontstaat er verschil tussen de reflecties, waardoor de motoren met verschillende snelheden aangestuurd worden. De route van de robot kan dan een te grote afwijking krijgen, waardoor de kans ontstaat dat na zo'n bocht de donkere lijn van het parcours niet meer wordt gevolgd.
Om dit te voorkomen is het handig om de waarden van de twee sensoren uit te lezen en deze weer te geven op een computerscherm. Voor het verschil tussen deze waarden kunt u de volgende algoritme op nemen in uw code:

[Verschil] = [Sensor waarde Links] – [Sensor waarde Rechts]

Hiervoor kunt u de programma's "Hyperterminal" of "Putty" gebruiken.

Neem de onderstaande instellingen over voor communicatie tussen de USB IR-ontvanger en de computer:

Bits per seconde	: 2400
Databits	: 8
Pariteit	: geen
Stopbits	: 1
Datatransportbesturing	: geen

Om de waarden vanuit de Asuro Robot naar het computerscherm te sturen dient de USB IR ontvanger verbonden te zijn met de computer en binnen het bereik te zijn van de Asuro Robot.

De waarden die nu weergeven worden voor respectievelijk de [Rechter sensor] en de [Linker sensor] kunt u in uw programma gebruiken.

Voor het uitlezen van de waarden van de fototransistoren kunt u de onderstaande code gebruiken:

```
#include "asuro.h"
int main(void)

{

        unsigned int data[2];          //Geheugenruimte reserveren
        Init();
        LCD_clear_screen();
        while(1); // Endless loop
        {
            LineData(data);    //Sensor data uitlezen

    // Print waarde van de Linker sensor op het computerscherm

            LCD_move_cursor_1st_line(0);
            SerWrite("LEFT ", 5);
            PrintInt(data[LEFT])

    // Print waarde van de Rechter sensor op het computerscherm

            LCD_move_cursor_2nd_line(0);
            SerWrite("RIGHT ", 6);
            PrintInt(data[RIGHT]);
            MSleep(200);

        }

    return 0;

}
```

TIP:
Meet op verschillende punten, en bepaal hiervan een gemiddelde waarde. Schakel eventueel de verlichting uit om eventuele reflecties te vermijden.

PID controller
Om de zwarte lijn te volgen en de robot niet te veel te laten uitwijken kunt u een PID regeling toepassen. De PID-regelaar gebruikt het volgende regelalgoritme voor de uitgang y(t):

$$y(t) = K_P \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt}$$

waarin:

- y(t) uitgang;
- Kp Proportionele actie van de regelaar (P-actie);
- Ki Integrale actie van de regelaar (I-actie);
- Kp Differentiële actie van de regelaar (D-actie);
- e(t) foutsignaal, het verschil van de proceswaarde PV(t) en het setpunt SP(t): e(t) = PV(t) – SP(t).

Op hoofdlijnen kan de code er bijvoorbeeld als volgt uitzien:

```
// PID-Controller

e_sum = 0; //begin waarde

loop{
        e = w – x; // bereken de Fout e
        e_sum = e_sum + e; // I-actie
        y = KP · e + KI · e_sum + KD · (e – e_old); //PID alg.
        e_old = e; // opslaan huidige fout voor de volgende differentiatie
    }
```

In het programma wordt:

- y uitgang
- e foutsignaal
- KI constante waarde KI = Ki · Ta
- KD constante waarde KD = Kd / Ta

Met een paar simpele veranderingen kan een PID-algoritme veranderd worden in een P-, PI- of PD-regelaar, bijvoorbeeld met Kd = 0, Ki = 0 krijg je een P-regelaar, de formule wordt y = KP · e

Het is aan te raden om te beginnen met een "Trial and Error" om te bepalen of u een PD-, PI- of een PID-controller dient te programmeren.

Onderstaand een globaal voorbeeld van een P-Controller die u kunt gebruiken:

```
#define MAXSPEED 200
#define KP 1
unsigned int data[2];
while (1)                    //eindeloze loop
{
//uitlezen van waarden T9 en T10
      LineData(data);

//Error
      error = data[LEFT] - data[RIGHT];

// P-Controller
      setwaarde = KP * error;

// Motoren aansturen

      if
      {
      speed_left = MAXSPEED;
      speed_right MAXSPEED – abs(setwaarde);
      }
      else
      {
      ......
      }
      MotorSpeed(speed_left,speed_right);
}
```

HINT:

- Vergeet niet om de variabelen setwaarde, error, speed_left, speed_right enz. te definiëren;
- error en setwaarde kunnen een negatieve waarde teruggeven;
- De P_Waarde moet gedeclareerd worden tussen 0.1 en 10.

Odometrie

De odometer bestaat uit:

- een LED (D13) en een fototransistor (T11) voor het linker wiel;
- een LED (D14) en een fototransistor (T12) voor het rechter wiel;
- een encoder.

Deze componenten voeren de meting uit van de zwarte - en witte markeringen op de wielen van de robot.

Om Odometrie toe te passen kunt u de volgende functies gebruiken:

- **encoder[LEFT], encoder[RRIGHT]:**
  Elke zwart-wit markering op de wielen geeft de decimale waarde 1. De huidige som van de getelde tellingen wordt opgeslagen in de globale variabele *encoder [ ]*.
  Door de robot naar voren te laten rijden wordt de som verhoogd, en door de robot achteruit te laten rijden wordt de som verlaagd.
- **void Encoder_Init(void):**
  Start meting, dit is tevens de initialisatie.
- **void Encoder_Set(int seti, int setr):**
  Stel de waarde in van encoder[LEFT] en encoder[RIGHT].
- **void Encoder_Stop(void):**
  Meting stoppen.
- **void Encoder_Start(void):**
  Opnieuw starten van de meting.

Voorbeeld:

```
Encoder_Init();
MotorDir(FWD,FWD);
MotoSpeed(150,150);
    while(1)
    {
        if(encoder[LEFT]>encoder[RIGHT])
        {
            StatusLED(GREEN);
        }
        else
        {
            StatusLED(RED);
        }
        SerWrite("L ",2);
        PrintInt(encoder[LEFT]);
        SerWrite("R ",3);
        PrintInt(encoder[RIGHT]);
    }
```

Bijlage 3: Beoordeling praktijkopdracht Robotica met RUML

Dirksen Hogeschool

Student:
Studentnummer:
Datum beoordeling:
Beoordeling door:

Checklist rapportage	Onvoldoende / Voldoende / Goed	
	Opmerkingen	**O/V/G**
Eisen aan de robot		
1. De robot onderbreekt de werkzaamheden direct als er een gevaarlijke situatie ontstaat.		
2. De robot ontwijkt obstakels.		
3. De robot is voorzien van een noodstopvoorziening, de werkzaamheden direct worden gestopt als die wordt bediend.		
4. Vanuit een reset na de noodstop situatie worden de werkzaamheden weer hervat.		
5. De robot rijdt een route van A naar B, volgens parcours.		
6. De robot rijdt een route van A naar B met obstakels.		
7. De robot bepaalt de snelste route van A naar B, volgens parcours.		
8. De robot rijdt de snelste route van A naar B met obstakels.		
Totaal onderdeel Robot: ***Voldoende:*** *Wanneer aan eisen 1 t/m 6 is voldaan.* ***Goed:*** *Wanneer aan alle voorwaarden is voldaan.*		