Stuhlroboter Struktogramme

DataSol 8:

Paul Winkler, Manuel Wischnat,

Oskar Wortmann, Tobias Ziegler

task main()

Variablendefinition

int linienFarbton, int zeitFuer90GradDrehung

- sensorenInitialisieren()
- **linienFarbton** ← kalibrierungSW()
- **zeitFuer90GradDrehung** ← kalibrierung90()
- startVorgang()
- sortierVorgang(**linienFarbton**, **zeitFuer90GradDrehung**)

void sensorenInitialisieren()

- SetSensorColorFull(IN_1) (Farbsensor initialisieren)
- SetSensorLowspeed(IN_2) (Ultraschallsensor initialisieren)
- SetSensorLight(IN_3) (Lichtsensor initialisieren)
- SetSensorTouch(IN_4) (Berührungssensor initialisieren)

void startVorgang()

- Bildschirm löschen
- Bedienungsanleitung ausgeben
- solange Sensor(IN_4) ≠ 1
  - %
- Bildschirm löschen
- Ton ausgeben
- 2 Sekunden warten

void sortierVorgang(int linienFarbton, int zeitFuer90GradDrehung)

Übergabeparameter

int linienFarbton, **int** zeitFuer90GradDrehung

Stuhlzähler initialisieren

int stuhlZaehler = 0

stuhlZaehler anzeigen

solange wahr

- **int stuhlFarbe** ← stuhlFarbeHerausfinden()
- zuFarbeFahren(linienFarbton, **stuhlFarbe**, zeitFuer90GradDrehung)
- stuhlZaehler = stuhlZaehler + 1
- rechtsDrehung(zeitFuer90GradDrehung * 2)
- zuStartpunktFahren(linienFarbton, **stuhlFarbe**, zeitFuer90GradDrehung)
- rechtsDrehung(zeitFuer90GradDrehung * 2)

int stuhlFarbeHerausfinden()

Variablendefinition

int stuhlFarbe

solange wahr

- **stuhlFarbe** ← Sensor(IN_1)

stuhlFarbe = schwarz oder
stuhlFarbe = rot oder
stuhlFarbe = gelb ?

wahr | falsch

- Schleife abbrechen
- fehlerMeldungAnzeigen("Farbe nicht erkannt!")

stuhlFarbe zurückgeben

void zuFarbeFahren(int linienFarbton, int stuhlFarbe, int zeitFuer90GradDrehung)

Übergabeparameter

int linienFarbton, **int** stuhlFarbe, **int** zeitFuer90GradDrehung

stuhlFarbe

Schwarz	Rot	Gelb	default
linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)	%
geradeAusFahren(SEC_1 + MS_200)	geradeAusFahren(SEC_1 + MS_200)	geradeAusFahren(SEC_1 + MS_200)	
rechtsDrehung(zeitFuer90GradDrehung)	linksDrehung(zeitFuer90GradDrehung)	linienVerfolgung(linienFarbton)	
linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)	Switch abbrechen	
geradeAusFahren(MS_400)	geradeAusFahren(MS_400)		
linksDrehung(zeitFuer90GradDrehung)	rechtsDrehung(zeitFuer90GradDrehung)		
linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)		
Switch abbrechen	Switch abbrechen		

void zuStartpunktFahren(int linienFarbton, int stuhlFarbe, int zeitFuer90GradDrehung)

Übergabeparameter

int linienFarbton, **int** stuhlFarbe, **int** zeitFuer90GradDrehung

stuhlFarbe

Schwarz	Rot	Gelb	default
linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)	%
geradeAusFahren(MS_400)	geradeAusFahren(MS_400)	geradeAusFahren(SEC_1 + MS_200)	
rechtsDrehung(zeitFuer90GradDrehung)	linksDrehung(zeitFuer90GradDrehung)	linienVerfolgung(linienFarbton)	
linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)	Switch abbrechen	
geradeAusFahren(SEC_1 + MS_200)	geradeAusFahren(SEC_1 + MS_200)		
linksDrehung(zeitFuer90GradDrehung)	rechtsDrehung(zeitFuer90GradDrehung)		
linienVerfolgung(linienFarbton)	linienVerfolgung(linienFarbton)		
Switch abbrechen	Switch abbrechen		

void linienVerfolgung(int linienFarbton)

Globale Konstanten

POWER == 40,
LINIENVERFOLGUNG_LINKSDREHUNG_ZEIT == 120,
LINIENVERFOLGUNG_RECHTSDREHUNG_ZEIT == 240

Übergabeparameter

int linienFarbton

solange wahr

Sensor(IN_3) ≤ linienFarbton ?

wahr

- arbeitUnterbrechenFallsWegBlockiert()
- Motoren A und B vorwärts mit der Leistung POWER einschalten
- Neuen Schleifendurchlauf anstoßen

falsch

- linksDrehung(LINIENVERFOLGUNG_LINKSDREHUNG_ZEIT)

Sensor(IN_3) ≤ linienFarbton ?

wahr

- Neuen Schleifendurchlauf anstoßen

falsch

- rechtsDrehung(LINIENVERFOLGUNG_RECHTSDREHUNG_ZEIT)

Sensor(IN_3) ≤ linienFarbton ?

wahr

- Neuen Schleifendurchlauf anstoßen

falsch

- linksDrehung(LINIENVERFOLGUNG_LINKSDREHUNG_ZEIT)
- Schleife abbrechen

void linksDrehung(int zeitspanne)

Globale Konstanten

POWER == 40

Übergabeparameter

int zeitspanne

- arbeitUnterbrechenFallsWegBlockiert()

Motor B vorwärts mit der Leistung POWER einschalten

Motor A rückwärts mit der Leistung POWER einschalten

zeitspanne lang warten

Motoren A und B abschalten

void rechtsDrehung(int zeitspanne)

Globale Konstanten

POWER == 40

Übergabeparameter

int zeitspanne

arbeitUnterbrechenFallsWegBlockiert()

Motor A vorwärts mit der Leistung POWER einschalten

Motor B rückwärts mit der Leistung POWER einschalten

zeitspanne lang warten

Motoren A und B abschalten

void geradeAusFahren(int zeitspanne)

Globale Konstanten

POWER == 40

Übergabeparameter

int zeitspanne

arbeitUnterbrechenFallsWegBlockiert()

Motoren A und B vorwärts mit der Leistung POWER einschalten

zeitspanne lang warten

Motoren A und B abschalten

void arbeitUnterbrechenFallsWegBlockiert()

Globale Konstanten

MINIMALE_DISTANZ_ZU_OBJEKT == 5

solange SensorUS(IN_2) < MINIMALE_DISTANZ_ZU_OBJEKT

fehlerMeldungAnzeigen("Der Weg ist blockiert!")

void fehlerMeldungAnzeigen(string nachricht)

Übergabeparameter **string** nachricht
Zeile 3 auf Bildschirm löschen
Zeile 4 auf Bildschirm löschen
Zeile 5 auf Bildschirm löschen
"FEHLER" auf Zeile 3 ausgeben
nachricht auf Zeile 4 ausgeben
"Taste druecken" auf Zeile 5 ausgeben
solange Sensor(IN_4) ≠ 1   %
Zeile 3 auf Bildschirm löschen
Zeile 4 auf Bildschirm löschen
Zeile 5 auf Bildschirm löschen

int kalibrierungSW(void)

Variablendefinition bzw. -initialisierung

long end_time, **long** t0 = CurrentTick(),
int iDurchschnitt, **int** imin = 999, **int** imax = -1, **int iHelligkeitKal** = 0

Bildschirm löschen

"KALIBRIERUNG LINKS" auf Zeile 1 des Bildschirms ausgeben

Motor B vorwärts mit Leistung 50% einschalten

Motor A rückwärts mit Leistung 50% einschalten

solange CurrentTick()-t0 < 990

- **iHelligkeitKal** ← Sensor(IN_3)
- **iHelligkeitKal** > imax ?
  - wahr:
    - imax = **iHelligkeitKal**
  - falsch:
    - **iHelligkeitKal** < imin ?
      - wahr:
        - imin = **iHelligkeitKal**
      - falsch:
        - %

"KALIBRIERUNG RECHTS" auf Zeile 2 des Bildschirm ausgeben

- **end_time** ← CurrentTick() + 1200

Motor A vorwärts mit Leistung 50% einschalten

Motor B rückwärts mit Leistung 50% einschalten

solange CurrentTick() < **end_time**

- **iHelligkeitKal** ← Sensor(IN_3)
- **iHelligkeitKal** > imax ?
  - wahr:
    - imax = **iHelligkeitKal**
  - falsch:
    - **iHelligkeitKal** < imin ?
      - wahr:
        - imin = **iHelligkeitKal**
      - falsch:
        - %

iDurchschnitt = (imin + imax) / 2

Zeile 3 auf Bildschirm löschen

"SW Linienkante:" auf Zeile 3 des Bildschirms ausgeben

iDurchschnitt auf Zeile 3 des Bildschirms ausgeben

solange Sensor(IN_3) > iDurchschnitt

- Motor B vorwärts mit Leistung 50% einschalten
- Motor A rückwärts mit Leistung 50% einschalten

Motoren A und B abschalten

iDurchschnitt zurückgeben

long kalibrierung90()

```
Globale Konstanten

POWER == 40
```

```
Bildschirm löschen
```

```
"KALIBR. LÄUFT" auf Zeile 1 des Bildschirms ausgeben
```

```
Variablendefinition bzw. -initialisierung

int iHelligkeitkal = 0,
long t0kal = CurrentTick(), turntime180 = 0
```

```
Motor A vorwärts mit der Leistung POWER einschalten
```

```
Motor B rückwärts mit der Leistung POWER einschalten
```

```
300ms warten
```

```
    iHelligkeitkal ← Sensor(IN_3)
```

```
solange iHelligkeitkal ≥ 50
```

```
"LINIE GEFUNDEN" auf Zeile 2 des Bildschirms ausgeben
```

```
  turntime180 ← CurrentTick() - t0kal
```

```
Motor B vorwärts mit der Leistung POWER einschalten
```

```
Motor A rückwärts mit der Leistung POWER einschalten
```

```
300ms warten
```

```
    iHelligkeitkal ← Sensor(IN_3)
```

```
solange iHelligkeitkal ≥ 50
```

```
"KALIBR. ABGESCHLOSSEN" auf Zeile 3 des Bildschirms ausgeben
```

```
turntime180 / 2 zurückgeben
```