HOCHSCHULE FURTWANGEN UNIVERSITY HFU

Dokumentation

SensorCar OpenPEARL

Semesterprojekt WS17/18

Referent	:	Prof. Dr. Rainer Müller
Korreferent	:	-
Vorgelegt am	:	18.12.2017
Vorgelegt von	:	Kevin Hertfelder Stefan Kienzler Patrick Kroner Daniel Petrusic Daniel Schlageter

Inhaltsverzeichnis

1 Einleitung

Das Semesterprojekt OpenPEARL im Wintersemester 2017/18 beschäftigt sich mit der Anwendung von OpenPEARL in einem Beispielprojekt.

1.1 OpenPEARL

PEARL steht für *Process and Experiment Automation Realtime Language*, eine höhere Programmiersprache, die speziell dafür entworfen wurde, Multitasking-Aufgaben bei der Steuerung technischer Prozesse zu steuern. Die Sprache wurde um 1975 am IRT Institut der Leibniz Universität in Hannover entwickelt und 1998 vom Deutschen Institut für Normung in der DIN66253-2 standardisiert.
OpenPEARL ist eine quelloffene Implementierung eines Kompilers und einer Laufzeitumgebung für PEARL. OpenPEARL befindet sich aktuell noch in der Entwicklung, ist aber weit genug fortgeschritten, um erste Beispielprojekte damit umzusetzen. In dieser Dokumentation werden deshalb auch in OpenPEARL gefundene Fehler festgehalten.

1.2 SensorCar

Ziel des Projektes ist es, ein autonomes Modellauto zu bauen, das, mit verschiedenen Sensoren ausgestattet, einer Linie folgen und auf Abruf weitere Manöver ausführen kann. Zur Umsetzung wird ein Raspberry Pi 3 mit mehreren Sensoren und Aktoren verbunden auf einem eigens konstruierten Fahrzeug aufgebracht und mit OpenPEARL programmiert.

2 Funktionsbeschreibung

2.1 Grundfunktionalität

Das OpenPEARL SensorCar ist ein mit zwei Motoren und unterschiedlichen Sensoren ausgestattetes Modellauto, das einer schwarzen Linie auf hellem Untergrund folgen soll.
Um dies zu erreichen, werden folgende Komponenten eingesetzt:

- **Raspberry Pi mit OpenPEARL**: Kernkomponente des SensorCar ist ein Raspberry Pi, auf dem ein OpenPEARL Programm läuft, das alle angeschlossenen Sensoren und Aktoren verwaltet und steuert.
- **Schrittmotor**: Zwei Schrittmotoren steuern jeweils zwei über ein Gummi verbundene Räder (ein Motor für eine Seite). Dadurch lässt sich das Auto mit unterschiedlichen Geschwindigkeiten sowohl vorwärts wie auch rückwärts bewegen und Kurven mit beliebigen Radius fahren.
- **Lichtrechen**: Ein Lichtrechen, bestehend aus mehreren binären Helligkeitssensoren, erfasst den Untergrund in zwei Helligkeitsstufen. Die hier erfassten Informationen werden verwendet, um den Motor so anzusteueren, dass das SensorCar der schwarzen Linie folgt.
- **Modellauto**: Alle Komponenten werden auf einem selbstgedruckten Modellauto aufgebracht.
- **Beschleunigungssensor**: Ein Beschleunigungssensor erfasst die auftretende Beschleunigung und kann daraus Bewegungen ableiten. Diese können eventuell als Eingabe für eine Regelung genutzt werden.

2.2 Erweiterte Funktionalität

Zusätzlich zum einfachen Nachfahren einer schwarzen Linie soll das SensorCar verschiedene Aktionen auf Abruf – durch Erkennung von Farbpunkten neben der Linie – durchführen können. Dies sind:

- **Umdrehen**: Das SensorCar dreht auf der Stelle und fährt in die entgegengesetzte Richtung weiter.
- **Richtungsänderung**: Das SensorCar ändert die Fahrtrichtung ohne umzudrehen.
- **Abbiegen**: Das SensorCar biegt an einer Kreuzung in eine bestimmte Richtung ab.

2.3 Kommunikation

Aktuelle Informationen über das SensorCar können über einen Browser mittel einer *http* Anfrage abgefragt werden. Dazu läuft in OpenPEARL ein Webserver, der die Informationen abfragt und ausgibt.
Eingehende Kommunikation erfolgt über ssh.

2.4 Sonstiges

Zur Beleuchtung des SensorCar werden LEDs eingesetzt:

- **Weiße LEDs**: Zwei weiße LEDs dienen als Frontscheinwerfer.
- **Rote LEDs**: Zwei rote LEDs dienen als Rückleuchten.
- **Gelbe LEDs**: Vier gelbe LEDs dienen als Blinker.

3 Planung

3.1 Teilprojekte

Das Projekt wird in folgende Phasen bzw. Teilprojekte unterteilt:

- **Projektstart**
  Einführung in das Projekt und Festlegung der Organisation. Grundlegende Abstimmung über Ziel und Umfang.
- **Einführung OpenPEARL und Philosophenproblem**
  Zur Einarbeitung in die Programmiersprache OpenPEARL implementiert das Projektteam jeweils das Philosophenproblem.
- **Einrichtung der Raspberry Pis**
  Für die weitere Arbeit am Projekt werden zwei Rasperberry Pi 3 eingerichtet, um mit OpenPEARL und NFS zu arbeiten. Der Zugriff auf die Rechner soll mittels *ssh* möglich sein.
- **Inbetriebnahme und Test der Hardwarekomponente**
  Die einzelnen Komponenten (Sensoren und Aktoren) werden als Teilprojekte separat am Raspberry Pi in Betrieb genommen und getestet.
- **Entwurf und Druck des Modells**
  Das Modell für den mechanischen Aufbau wird entwickelt und mit dem 3D-Drucker ausgedruckt und getestet.
- **Detailplanung des Gesamtsystems**
  Die genaue Funktionalität und der Hardwareaufbau des Gesamtsystems werden festgelegt und dokumentiert.
- **Zusammensetzung Gesamtsystems**
  Nachdem alle Komponenten erfolgreich in Betrieb genommmen und getestet wurden, wird das Gesamtsystem im Sinne des Ziels zusammengebaut. Daraufhin wird die Funktionalität des Gesamtsystems programmiert. Im Sinne eines inkrementell iterativen Vorgehens erfolgen Anforderungsanalyse, Implementierung und Tests bis das System die Anforderungen erfüllt.

- **Projektabschluss**
  Zum Abschluss des Projekts erfolgt die Fertigstellung der Dokumentation und Abnahme des Ergebnisses.

- **Projektvorstellung**
  Das Ergebnis des Projektes wird im Rahmen der Projektpräsentationen vorgestellt.

3.2 Zeitplanung

Die untenstehende Tabelle detailliert die grobe Zeitplanung für den Projektablauf. Änderungen sind vorbehalten.

Zeitraum	**Teilprojekt / Aufgabe**
09.10.17 – 16.10.17	Projektstart
16.10.17 – 23.10.17	Einführung in OpenPEARL und Philosophenproblem
23.10.17 – 30.10.17	Einrichtung der Raspberry Pis
30.10.17 – 11.12.17	Inbetriebnahme und Test der Hardwarekomponenten
30.10.17 – 18.12.17	Entwurf und Druck des Modells
11.12.17 – 15.01.18	Zusammensetzung des Gesamtsystems
15.01.18 – 22.01.18	Projektabschluss
26.01.18	Projektpräsentation

4 Philosophenproblem

Um mit der Sprache OpenPEARL und ihren Konstrukten insbesondere zur Nebenläufigkeit vertraut zu werden, wurde als einführendes Beispiel das Philosophenproblem gelöst.

4.1 Das Philosophenproblem

Das Philosophenproblem ist ein bekanntes Fallbeispiel für Nebenläufigkeit und Verklemmung. Fünf Philosophen sitzen um einen runden Tisch, jeder der Philosophen hat einen Teller mit Spaghetti vor sich. Um diese zu essen, benötigt jeder Philosoph zwei Gabeln, es sind jedoch nur fünf Gabeln, jeweils eine zwischen zwei Philosophen, vorhanden. Es können also nicht alle Philosophen gleichzeitig essen.
Die Philosophen werden als Prozesse oder Threads betrachtet, die gleichzeitig auf mehrere Ressourcen, die Gabeln, zugreifen möchten.

4.2 Umsetzung

Die Gabeln werden als Semaphoren realisiert. Diese bieten nur einer bestimmten Anzahl Threads bzw. im Kontext von OpenPEARL *Tasks* die Möglichkeit der gleichzeitigen Belegung. Beim Philosophenproblem kann jede Gabel, also jede Semaphore, nur einmal belegt werden.
Zum Essen benötigt ein Philosoph zwei Gabeln, also auch zwei Semaphoren. Erfolgt der Zugriff auf die Semaphoren nicht-atomar, das heißt voneinander getrennt und nacheinander, so kann es zur Verklemmung kommen: Jeder Philosoph hält eine Gabel und wartet auf die Freigabe der zweiten — die Philosophen verhungern.
OpenPEARL bietet zur einfachen Lösung des Problems die Möglichkeit, mehrere Semaphoren in einer atomaren Aktion gleichzeitig aufzunehmen. Dies ist nur dann erfolgreich, wenn alle Semaphoren belegbar sind, und verhindert eine Verklemmung.

4.3 Codebeispiel

Im Folgenden sind die relevanten Codeausschnitte einer Lösung des Philosophenproblems in OpenPEARL aufgeführt.

4.3.1 Definition der Semaphoren

```
! Semaphoren ("Gabeln"), die maximal ein mal (gleichzeitig) belegt werden koennen
DCL (g1, g2, g3, g4, g5) SEMA PRESET(1,1,1,1,1);
```

4.3.2 Aktivität eines Philosophen als Task

```
philosopher1: TASK;
        REPEAT;
                ! Gleichzeitige Anfrage fuer die Belegung der Semaphoren
                REQUEST g1, g2;
                PUT 'Philosopher 1 starts eating...' TO termout BY A, SKIP;
                AFTER timeone RESUME;
                PUT 'Philosopher 1 stops eating...' TO termout BY A, SKIP;
                ! Gleichzeitige Freigabe der Semaphoren
                RELEASE g1, g2;
        END;
END;
```

Komplette Lösungen können im Ordner *Code/Philosophenproblem* des Repositories eingesehen werden.

4.3.3 Kompilerfehler

Während der Implementierung des Philosophenproblems wurde ein Fehler im Kompiler entdeckt, der unter bestimmten Umständen das gleichezeitige Belegen zweier Semaphoren verhindert hat. Der Fehler wurde inzwischen behoben.

5 Mechanischer Aufbau

6 Komponenten

6.1 Raspberry Pi 3

mit Anschlussplan...

6.2 Schrittmotor

Die Schrittmotoren werden jeweils über einen Motortreiber und vier GPIO-Pins des Raspberry Pi angesteuert. Energie erhält der Motor durch ein an den Treiber angeschlossenes Netzteil mit 12V Gleichstrom. Durch Alternieren der Bits an den vier GPIO-Pins wird der Schrittmotor um jeweils einen Schritt bewegt.

7 Gesamtsystem