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.

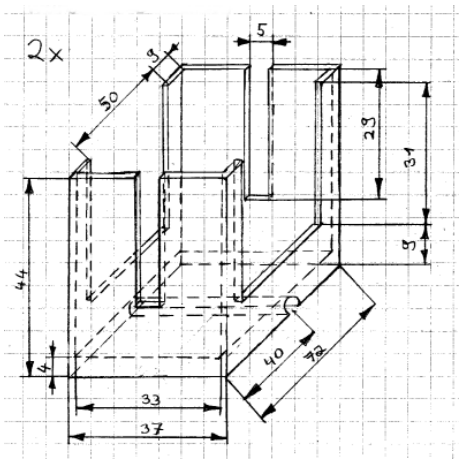
5 Mechanischer Aufbau

5.1 Rahmenbedingungen

Aufgrund des Vorhandenseins eines 3D-Druckers an der Hochschule und der damit verbundenen Flexibilität, haben wir uns entschlossen, das Gehäuse und die Halterungen für die einzelnen Komponenten zu drucken. Allerdings gilt es auch hierbei einige Dinge zu beachten. Dadurch, dass der Drucker die Modelle schichtenweise von unten nach oben aufbaut, können beispielsweise Löcher in Wänden nur umständlich oder überhaupt nicht realisiert werden. Theoretisch kann dies durch ein Kippen des zu druckenden Körpers oder das Mitdrucken von Hilfselementen umgangen werden. In der Praxis und vor allem dann, wenn solche Probleme auf mehreren Seiten beziehungsweise an sehr filigranen Stellen eines Modells auftreten, nutzen auch diese Methoden nichts mehr.
Zudem kann das Innenleben der gedruckten Teile entweder durch eine Art Wabenstruktur oder komplett gefüllt aufgebaut sein. Durch den Aufbau in Form einer Wabenstruktur wird zum einen ein geringerer Verbrauch von Druckmaterialen und zum anderen eine wesentlich geringere Druckzeit erzielt. Vor allem der Zeitfaktor ist dabei nicht unrelevant, da selbst der Druck von einfachen und mittelgroßen Teilen in der Regel mehrere Stunden in Anspruch nimmt. Wenn man jedoch beabsichtigt die Komponenten zu verschrauben, oder ein sehr hohe Stabilität vorsieht, sollte man auf diese Druckart verzichten und einen gefüllten Druck vorziehen.
Um in dem Fall, dass ein Teil unseres Modells beschädigt wird, nicht alles erneut drucken zu müssen, ist das Gesamtmodell modular aufgebaut. Jedes Modul weißt entweder den weiblichen oder den männlichen Teil einer Steckverbindung auf.
Außerdem ist der Druckbereich eines 3D-Druckers beschränkt. Bei dem von uns genutzten 3D-Drucker sind wir vor allem auf einer der horizonzalen Achsen eingeschränkt. Der hier zur Verfügung stehende Druckbereich von maximal 21,4cm hat somit die Länge unseres Modells bestimmt.

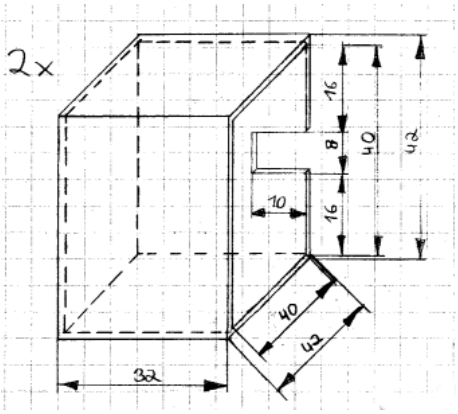
5.2 Motorenführung

Da wir uns dazu entschlossen haben, jeweils beide Räder auf einer Seite über einen O-Ring miteinander zu verbinden, muss es eine möglichkeit geben, diese ein- und nachspannen zu können. Aus diesem Grund haben wir eine Art Schiene modelliert, in der ein Motor über jeweils eine Schraube mit zwei Muttern an der Vorder- sowie Hinterseite befestigt werden kann. Sie weißt im Inneren eine effektive Länge von 68mm auf und erlaubt es uns somit je nach Größe der Muttern die Motoren mit einer Länge von 40mm um 15mm bis 20mm zu verstellen. Auf der hier gezeigten Skizze ist ein Schraubendurchmesser von 5mm vorgesehen. Dieser kann jedoch beliebig angepasst werden, wenn man die entsprechenden Maße der Motorenführung abändert. Über den weiblichen Teil einer Steckverbindung am unteren Ende wird die Motorenführung später seitlich mit später in der Dokumentation beschriebenen Bodenplatte verbunden.



5.3 Motorenfassung

Die Motoren werden jeweils in einer Halterung eingefasst, welche später in der Motorenführung beweglich gelagert sein wird, um die O-Ringe ein- und nachspannen zu können. Der Primärgrund für die Verwendung solcher Motorenfassungen ist jedoch der, dass die zu den Motoren gehörenden Treiberstufen über ein weiteres Bauteil an den Oberseiten der Motorenfassungen befestigt werden können. Auf diese Weise kann der für die Treiberstufen benötigte Platz auf der Bodenplatte eingespart werden. Zudem ist die Länge der Kabel zwischen Motor und Treiberstufe somit konstant. Um die Kabel, die seitlich aus den Motoren kommen zu berücksichtigen, befindet sich an einer Seite der Motorenfassungen eine entsprechende Aussparung.



6 Komponenten

6.1 Raspberry Pi 3

Als Grundlage für das SensorCar wird ein Raspberry Pi 3 genutzt, auf dem OpenPEARL installiert ist. Die Installation erfolgt nach der Anleitung im OpenPEARL Repository. Die Sensoren und Aktoren des SensorCars werden wie folgt am Raspberry Pi angeschlossen:

Gerät	**BCM**	**Versorgung**
Lichtrechen	6 – 13	3,5 / 5V
Motortreiber	16 – 19, 20 – 23	extern
Farbsensor	2, 3 (I^2C)	3,3 / 5V
Gyroskop	2, 3 (I^2C)	5V
Kompass	2, 3 (I^2C)	5V
LEDs	24 –	5V

Der zugrundeliegende Belegungsplan ist im Anhang 8.1 zu finden.

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.

6.3 Lichtrechen

Der Lichtrechen ist essenzieller Bestandteil des Autos und ist dafür zuständig die Position des Autos relativ zum vorgegebenen Pfad zu ermitteln. Der Lichtrechen QTR-8A der Marke Pololu besteht aus 8 Reflektionssensoren, die mit infrarot LED's die Reflexivität des Untergrundes auf einer Linie von ca. 7cm ermitteln. In unseren Anwendungsfall soll der Sensor ein schwatzes Klebeband auf einem hellen Untergrund erkennen und diese Information zur Steuerung des Autos weiterleiten.

Die Verbindung zum System geschieht über acht separate Datenpins, die jeweils konstant ein einzelnes Bit aussendet das die Ausgabe eines einzelnen Sensors binär repräsentiert. Des Weiteren gibt es noch einen Abschluss für die Stromzufuhr für 5 und 3.3V, einen Pin für die Erdung und einen Pin mit dem die Infrarot-LED's an oder abgeschaltet werden können. In unserem Aufbau nutzen wir die 5V Stromversorgung und lassen die LED-Steuerungspin unangeschlossen, in welchem Fall die Sensoren dauerhaft eigeschaltet bleiben.

Im Anschlussplan ist vorgesehen, dass der Lichtrechen über die Datenpins 6 -13 am Raspberry Pi ausgelesen wird.

Bei der Auswertung der Datenpins werden den Pins von rechts nach links Gewichtungen von 4 bis -4 gegeben und jeder Pin der wenig Reflexion misst wird seiner Wertung nach aufsummiert. Diese Summe Wird dann durch die Anzahl des Summanden geteilt und ergibt somit einen Durchschnittswert für die Position des Leitstreifens. Dieses Verfahren schwächt auch fehlerhafte Werte einzelner Sensoren ab.

7 Gesamtsystem

8 Anhang

8.1 Belegungsplan Raspberry Pi 3