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

Inhaltsverzeichnis ii

1 Einleitung 1

1.1 OpenPEARL 1

1.2 SensorCar 1

2 Funktionsbeschreibung 3

2.1 Grundfunktionalität 3

2.2 Erweiterte Funktionalität 3

2.3 Kommunikation 4

2.4 Sonstiges 4

3 Planung 5

3.1 Teilprojekte 5

3.2 Zeitplanung 6

4 Philosophenproblem 7

4.1 Das Philosophenproblem 7

4.2 Umsetzung 7

4.3 Codebeispiel 8

4.3.1 Definition der Semaphoren 8

4.3.2 Aktivität eines Philosophen als Task 8

4.3.3 Kompilerfehler 8

5 Mechanischer Aufbau 9

5.1 Rahmenbedingungen 9
5.2 Motorenführung 10
5.3 Motorenfassung 10
5.4 Treiberstufenhalteung 11
5.5 Vorderachsenhalterung 11
5.6 Bodenplatte 12
6 Komponenten 15
6.1 Raspberry Pi 3 15
6.2 Schrittmotor 15
6.3 Lichtrechen 16
6.4 Farbsensor 17
6.5 TCP/IP Socket 18
7 Gesamtsystem 19
8 Anhang 21
8.1 Belegungsplan Raspberry Pi 3 22

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.

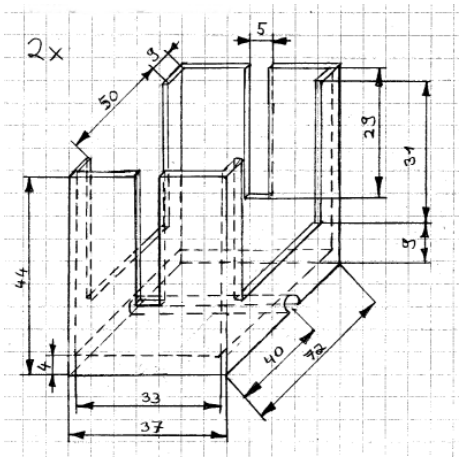
5 Mechanischer Aufbau

5.1 Rahmenbedingungen

Aufgrund des Vorhandenseins eines 3D-Druckers an der Hochschule und der damit verbundenen Flexibilität, haben wir uns entschlossen, die Bodenplatte und die Halterungen für die einzelnen Komponenten zu drucken. Allerdings müssen auch hierbei einige Dinge beachtet werden. 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 zu unterschätzen, da selbst der Druck von einfachen und mittelgroßen Teilen in der Regel mehrere Stunden in Anspruch nimmt und ein gefüllter Druck die benötigte Zeit nicht selten verdoppelt. 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 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. Bis auf wenige Ausnahmen werden von uns alle Bauteile mit einer Wabenstruktur im Inneren gedruckt.

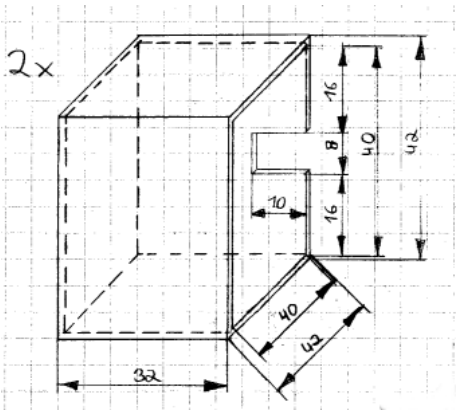
5.2 Motorenführung

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. Eine Motorenführung weißt im Inneren eine effektive Länge von 68mm auf und erlaubt es uns somit je nach Größe der verwendeten Muttern die Motoren mit einer Länge von 40mm um ca. 15mm 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. Durch eine Wandstärke von 2mm wird eine ausreichende Stabilität erreicht. Über den weiblichen Teil einer Steckverbindung am unteren Ende werden die Motorenführungen seitlich mit der weiter unten beschriebenen Bodenplatte verbunden.



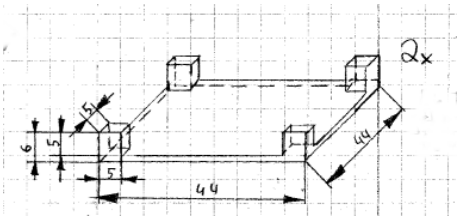
5.3 Motorenfassung

Die beiden Motoren werden jeweils in einer Halterung eingefasst, welche später in den Motorenführungen beweglich gelagert sein werden, 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.



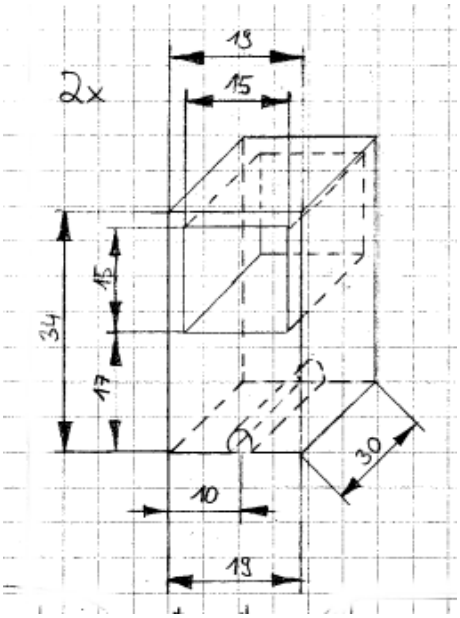
5.4 Treiberstufenhalteung

Die Treiberstufenhalterung ist im Grunde lediglich eine quadratische Grundplatte mit einer Erhöhungen an jeder Ecke. Diese Erhöhungen dienen dazu, die Treiberstufen mit jeweils vier Schrauben an den Treiberstufenhalterungen befestigen zu können. Durch den Einsatz von Schrauben kommt hier ein Druck mit Wabenstruktur nicht in Frage. Deshalb werden die Treiberstufenhalterungen gefüllt gedruckt. Sie werden mit den Oberseiten der Motorenfassungen verklebt.



5.5 Vorderachsenhalterung

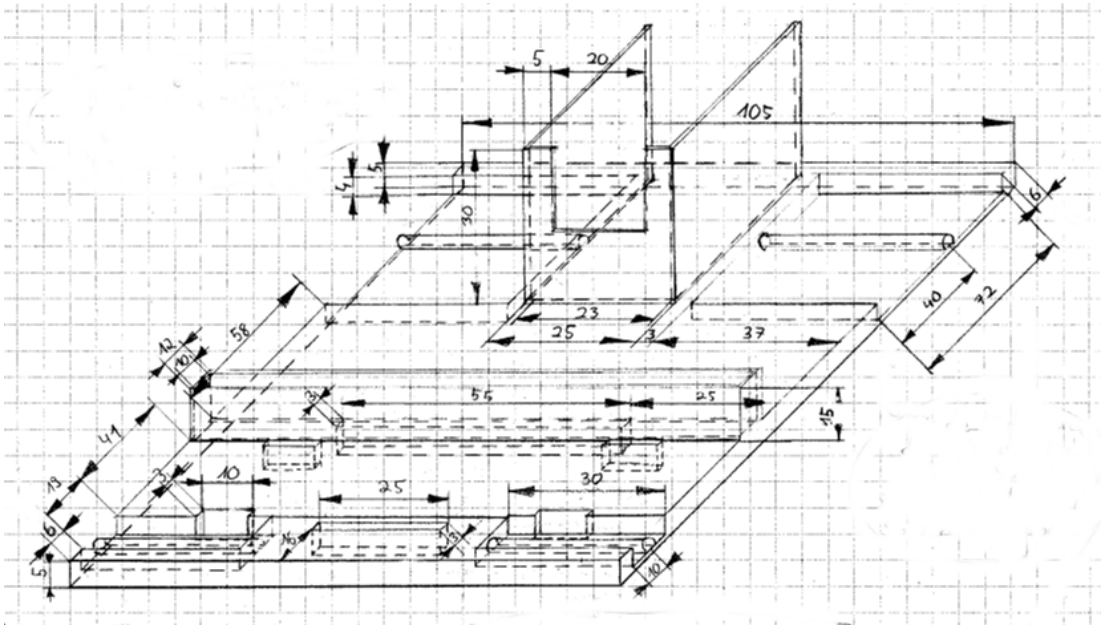
Um die vorderen Räder an unserem Modell befestigen zu können, ist eine entsprechende Halterung notwendig. Die beiden Vorderräder sind auf voneinander getrennten Achsen befestigt. Diese wiederum sind in jeweils einem Fischertechnik-Baustein frei drehbar gelagert. Um ein seitliches herausrutschen der Achsen zu verhindern werden an den Achsen befestigte Fischertechnik-Klemmen verwendet. Damit die Vorderräder auf dem selben Niveau wie die durch die relativ großen Schrittmotoren sehr hoch gelegenen Hinterräder sind, sind die Vorderachsenhalterungen 34mm hoch. Am oberen Ende befindet sich eine für die bereits zuvor genannten Fischertechnik-Bausteine vorgesehene Öffnung. Um eine ausreichende Stabilität gewährleisten zu können, ist auch hier eine Wandstärke von 2mm vorgesehen. Am unteren Ende befindet sich der weibliche Teil einer Steckverbindung, um die Vorderachsenhalterungen mit den auf der Bodenplatte befindlichen Gegenstücken zu verbinden.



5.6 Bodenplatte

Um ein funktionierendes Gesamtsystem zu erhalten, ist eine Bodenplatte von nöten. Auf ihr werden schließlich alle anderen Komponenten befestigt. Um keine Stabilität zu verlieren, wird sie aus einem einzigen Teil gedruckt. Die daraus resultierenden Nachteile, wie zum Beispiel die eingeschränkten Möglichkeiten durch den begrenzten Druckbereich des 3D-Druckers, mussten wir daher in Kauf nehmen.
Die Bodenplatte hat unter anderem aus Stabilitätsgründen eine Dicke von 5mm. Außerdem werden die anderen Bauteile über Steckverbindungen in abgesenkten Bereichen der Bodenplatte seitlich eingeschoben. Eine Breite von 105mm ergibt sich aus der breitesten Komponente, die auf der Bodenplatte befestigt werden muss, dem Akku. Die Länge von 214mm wurde durch den maximalen Druckbereich des von uns verwendeten 3D-Druckers bestimmt.
Jeweils vorne (auf der zugehörogen Skizze: unten) und hinten auf der Bodenplatte werden die Platinen mit den LEDs befestigt. Direkt hinter der vorderen LED-Platine finden sich jeweils links und rechts die Stellen, an denen die Vorderachsenhalterungen eingeschoben werden können. Dazwischen ist ein Durchbruch um die Kabel der vorderen LEDs nach unten durchführen zu können.
Im Bereich dahinter findet der für die Stromversorgung benötigte Akku seinen Platz. Durch die vier Durchbrücke können zwei Kabelbinder oder ähnliches gezogen werden, um den Akku an der Bodenplatte zu befestigen. Dahinter befindet sich eine oben geöffnete Box mit einem langen Durchbruch im inneren. Dort können die Kabel der Sensoren, welche an der Unterseite der Bodenplatte befestigt werden, und die Kabel der vorderen LEDs wieder nach oben geführt werden. So können sie einfach an dem direkt dahinter befestigten Raspberry Pi angeschlossen werden. Zu lange Kabel können dann in dieser Box verstaut werden.
Der RaspberryPi wird später an separat gedruckten Stelzen festgeschraubt, welche mit der Bodenplatte verklebt werden. Sie werden separat gedruckt, da auch diese Teile aufgrund der Tatsache, dass hier Schrauben zum Einsatz kommen, im Gegensatz zur Bodenplatte gefüllt gedruckt.
Die beiden darauffolgenden seitlichen Absenkungen mit den männlichen Teilen der Steckverbindungen dienen zur Befestigung der beiden Motorenführungen. Dazwischen ist wieder eine nach oben geöffnete Box, die zur Verkabelung dient. Diese ist zusätzlich teilweise nach vorne und komplett nach hinten geöffnet. So können auch bei der Verwendung eines Deckels Kabel nach vorne und hinten durchgeführt werden. Auch zum Anbringen der LED-

Platine am hinteren Ende der Bodenplatte ist die Öffnung an der Hinterseite der Box aus Platzmangel notwendig. Am hintersten Bereich der Bodenplatte wird wie bereits oben beschrieben die Platine mit den Rücklichtern befestigt.



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 paarweise Alternieren der Bits an den vier GPIO-Pins wird der Schrittmotor um jeweils einen Schritt bewegt.

Das nachstehende Codebeispiel zeigt die Ansteuerung eines Schrittmotors.

```
1 SYSTEM;
        stepmotor_out: RPiDigitalOut(19, 4);
3
PROBLEM;
5       SPC stepmotor_out DATION OUT SYSTEM BASIC BIT
            (4) GLOBAL;
        DCL motor DATION OUT BASIC BIT(4) CREATED (
            stepmotor_out);
```

```
7
main: TASK MAIN;
9       DCL time DUR INIT (0.01 SEC);
        DCL steps FIXED INIT (100);
11      DCL (a, b, c, d) BIT(4) INIT('1010'B1, '1001'B1
            , '0101'B1, '0110'B1);
        OPEN motor;
13      REPEAT
                SEND a TO motor;
15              AFTER time RESUME;

17              SEND b TO motor;
                AFTER time RESUME;
19
                SEND c TO motor;
21              AFTER time RESUME;

23              SEND d TO motor;
                AFTER time RESUME;
25      END;
        CLOSE motor;
27 END;
```

Dabei ist zu beachten, dass die Belegung mehrerer Pins des Raspberry Pi abwärts erfolgt, d.h. `RPiDigitalOut(19, 4)` belegt die Pins `16, 17, 18, 19`. Die Geschwindigkeit des Schrittmotors kann durch die Wartezeit `time`, die Richtung durch die Reihenfolge der Signale `a, b, c, d` geändert werden.

Das Speichern von `BIT(4)` Variablen in einem Array funktioniert in OpenPEARL aktuell noch nicht; aufgrund der wenigen Zustände wurde auf einen Workaround verzichtet.

Das komplette Schrittmotormodul kann im Ordner *Code/Komponenten/-Schrittmotor* des Repositories eingesehen werden.

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 er-

mitteln. 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.

6.4 Farbsensor

Der Farbsensor TCS 34725 Der Marke Adafruit besitzt einen Sensor, der sowohl einen RGB Farbwert als auch die gesamt Helligkeit ermitteln lässt. Um die Farben Best möglichst zu erkennen besitzt der Sensor eine sehr helle LED mit weißem Licht. In unserem Projekt soll der Farbsensor Farbige Signal-Punkte auf der Strecke erkennen die denen das Auto beispielsweise das Ende der Stecke erkennen soll.

Verbunden wir der Sensor im Gegensatz zum Lichtrechen erfolgt die Datenverbindung zum Farbsensor über einen I2C-Bus. Des Weiteren Besitzt der Sensor ebenfalls einen Stromverbindung für 5 und 3.3V, eine Erdung und Einen Pin zur Steuerung der LED.

Die Datenpins 2 und 3 am Raspberry Pi sind standartmäßig für diesen Bus vorkonfiguriert. Die Unterstützung von I2C auf dem Raspberry Pi muss aber erst konfiguriert werden. Dazu muss der Befehl "raspi-config" ausgeführt werden und dort unter "Advanced Options" Die Unterstützung in Kernel eingeschaltet werden. Zusätzlich musste in der Datei "/boot/config.txt" noch die Zeile " dtparam=i2c1=on" angefügt werden.

Um den Farbsensor in Pearl benutzen zu können musste dafür noch ein Treiben in C++ geschrieben werden. Dieser umfasst die Dateien "TCS34725.cc",

“TCS34725.h“ und “TCS34725.xml“ dieser Treiber wird in eine Headerfile von open Pearl integriert werden und stellt alle Methoden für die Nutzung des Sensors als DATION in OpenPEARL zur Verfügung.

Bei der Auswertung der Rohdaten vom Sensor konvertieren wir die Daten in Standard RGB Werte mit einer Spanne von 0 bis 255.

6.5 TCP/IP Socket

Über das Socket werden die Daten des SensorCars auf einer Webseite angezeigt. Hierbei bekommt das Socket einen html request vom Browser, an den er dann den HTML-Code schickt. Es werden die Geschwindigkeit, die Anzahl der Schritte und die Werte des Lichtrechens und des Farbsensors angezeigt. Es können außerdem Informationen zum Projekt und die Dokumentation aufgerufen werden.

Die Webseite kann im gleichen Netz über <ipadresse>/index.html aufgerufen werden. Damit die Daten immer aktuell sind, wird die Seite alle zwei Sekunden aktualisiert.

OpenPEARL hat von sich aus keine Möglichkeit ein Socket aufzusetzen. Es musste daher zuerst implementiert werden. Hierzu wurde eine C++ Datei mit dazugehöriger XML Datei geschrieben und dem System hinzugefügt. Die Dateien wurden im Makefile und im "PearlIncludes.h"registriert.

Man kann nun im Systemteil von OpenPEARL "TcpIpServer(portnummer)ängeben. Die Kommunikation läuft über "DATION INOUT". Somit ist eine beidseitige Kommunikation möglich.

7 Gesamtsystem

8 Anhang

8.1 Belegungsplan Raspberry Pi 3