hochschule mannheim

# LegoPI - Steuerung eines Lego Mindstorms Roboters mit Raspberry PI

Andreas Reichel, Jonas Weber, Joshua Vecsei

Robotik - Projektdokumentation

Dokumentation des Arbeitsergebnis eines Robotikprojekts durchgeführt im SS 2015

Studiengang Informatik

Fakultät für Informatik

Hochschule Mannheim

24.07.2015

Betreuer

Prof. Thomas Ihme, Hochschule Mannheim

Ute Ihme, Hochschule Mannheim

# Abstract


### *LegoPI - Steuerung eines Lego Mindstorms Roboters mit Raspberry PI*

Jemand musste Josef K. verleumdet haben, denn ohne dass er etwas Böses getan hätte, wurde er eines Morgens verhaftet. Wie ein Hund! sagte er, es war, als sollte die Scham ihn überleben. Als Gregor Samsa eines Morgens aus unruhigen Träumen erwachte, fand er sich in seinem Bett zu einem ungeheueren Ungeziefer verwandelt. Und es war ihnen wie eine Bestätigung ihrer neuen Träume und guten Absichten, als am Ziele ihrer Fahrt die Tochter als erste sich erhob und ihren jungen Körper dehnte. Es ist ein eigentümlicher Apparat, sagte der Offizier zu dem Forschungsreisenden und überblickte mit einem gewissermaßen bewundernden Blick den ihm doch wohlbekannten Apparat. Sie hätten noch ins Boot springen können, aber der Reisende hob ein schweres, geknotetes Tau vom Boden, drohte ihnen damit und hielt sie dadurch von dem Sprunge ab. In den letzten Jahrzehnten ist das Interesse an Hungerkünstlern sehr zurückgegangen. Aber sie überwanden sich, umdrängten den Käfig und wollten sich gar nicht fortrühren.

### *Application of a flux compensator for timetravel with a maximum velocity of warp 7*

The European languages are members of the same family. Their separate existence is a myth. For science, music, sport, etc, Europe uses the same vocabulary. The languages only differ in their grammar, their pronunciation and their most common words. Everyone realizes why a new common language would be desirable: one could refuse to pay expensive translators. To achieve this, it would be necessary to have uniform grammar, pronunciation and more common words. If several languages coalesce, the grammar of the resulting language is more simple and regular than that of the individual languages. The new common language will be more simple and regular than the existing European languages. It will be as simple as Occidental; in fact, it will be Occidental. To an English person, it will seem like simplified English, as a skeptical Cambridge friend of mine told me what Occidental is.

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Motivation

Um im Unterricht Schülern und Studenten dass Programmieren und Konstruieren von Robotern zu erläutern, ist Lego Mindstorms mit dem programmierbaren Baustein NXT eine gute Wahl. Das Ziel dieser Arbeit ist es das programmieren von Lego Mindstorms Robotern zu vereinfachen und gleichzeitig die Grenzen die duch den Baustein NXT gesetzt sind zu sprengen. Wir wollen die Möglichkeiten der Programmierung des NXT eins zu eins mit dem Minicomputer Raspberry PI abbilden. Damit ist es uns möglich eine einfachere Programmierschnittstelle anzubieten und zusätzlich die vielen Möglichkeiten des Raspberry PIs für Schüler zugänglich machen.

In der Folgenden Arbeit werden wir den, vom Raspberry PI gesteuerten, Lego Mindstorms Roboter LegoPI nennen.

Wir sehen den Raspberry PI als ein dem NXT überlegenes Steuerungsmodul, da auf dem Raspberry ein vollständiges Ubuntu Linux Betriebssystem läuft. Eine Überlegung der Autoren hierbei war es auf dem Ubuntu einen Webserver laufen zu lassen und dem Roboter über diesen aus der Ferne zu steuern. Weiterhin soll es in dieser Arbeit ermöglicht werden, weitere Sensoren an den Raspberry PI anschließen zu können. Zwar gibt es für Lego Mindstorms eine Liste von Sensoren, wie z.B einen Lichtsensor, Tastsensor und Ultraschallsensor, dieses Set an Sensoren ist allerdings eingeschränkt.

Das Bereitstellen eines programmierbaren Roboters für jeden Schüler kann schnell teuer werden. Wir sehen die Verwendung eines Raspberry PI auch deshalb als Vor-

teil, weil ein Raspberry B+ mit 39,90 Eur den NXT mit 9841 mit 148,99 Eur preislich schlägt. [1]

In dieser Arbeit werden wir uns darauf beschränken alle Sensoren des NXT Bausatzes mit dem Raspberry PI zu verbinden. Zusätzlich soll es möglich sein Motoren über den Raspberry zu steuern. Um zu Prüfen ob der Raspberry PI den NXT im Unterricht ersetzen kann, werden wir einen Parqour der auch von Schülern der Hochschule Mannheim absolviert werden muss mit dem LegoPI durchlaufen.

[1] Preisvergleich des Raspberry PI 2 B+ und NXT 9841 auf www.amazon.de

# Kapitel 2

# Grundlagen

## 2.1 Raspberry PI

Raspberry PI ist ein günstiger Computer in Kreditkartengröße, welches viele Schnittstellen bietet, wie z.B HDMI, USB, Audio, GPIOs und Ethernet. Was den Raspberry PI so beliebt macht um eigene Systeme zu bauen, ist dass auf ihm ein vollständiges Linux Betriebssystem läuft. Dies führt dazu, dass nahezu alles was auf einem Desktoprechner läuft auch auf einem Raspberry PI ausführbar ist.

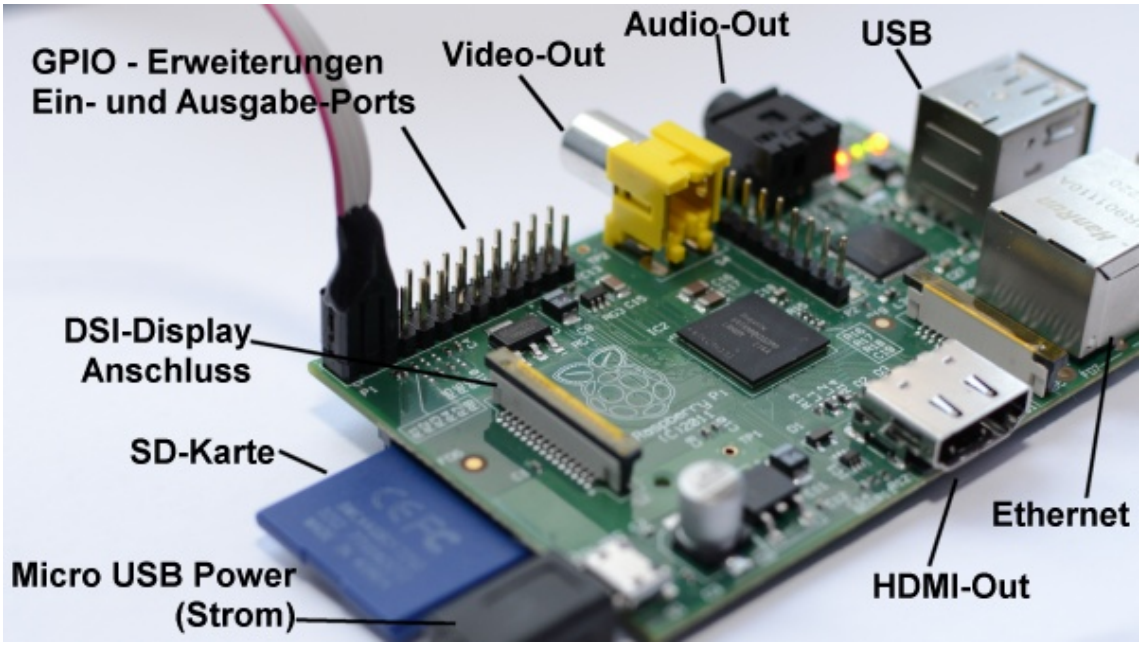


**Abbildung 2.1:** Ein RaspberryPI 2 mit Beschrifteten Schnittstellen. Quelle: http://www.portunity.de/blog/2013/februar/raspberry-pi-warum-ist-der-mini-computer-bei-unseren-mitarbeitern-so-beliebt.html

## 2.2 NXT

Lego Mindstorms NXT ist ein Steuerungscomputer der Produktserie Lego Mindstorms. Es hat ebenso wie der Raspberry PI Anschlüsse für USB- und Bluetooth Schnittstellen. Im Gegensatz zu Raspberry PI bietet der NXT schon von vornerein Anschlüsse für Sensoren und Aktoren.


**Abbildung 2.2:** Ein NXT Baustein mit Anschlüssen für Motoren und Sensoren. Quelle: https://de.wikipedia.org/wiki/Lego-Mindstorms-NXT

## 2.3 A/D-Wandler

Ein Analog-Digital-Umsetzer, kurz A/D-Wandler, ist ein elektronisches Baustein, bei dem ein zeit-kontinierliches Eingangssignal in einzelne diskrete Abtastwerte abgetastet werden.

In folgender Abbildung ist ein MCP3008 abgebildet. In dieser Arbeit wird dieser Baustein dazu verwendet, analoge Signale von Lego Sensoren in digitale Daten abzutasten, um diese im Raspberry PI zu verarbeiten. Der MCP3008 bietet 8 Eingänge um Sensoren Anzuschließen.

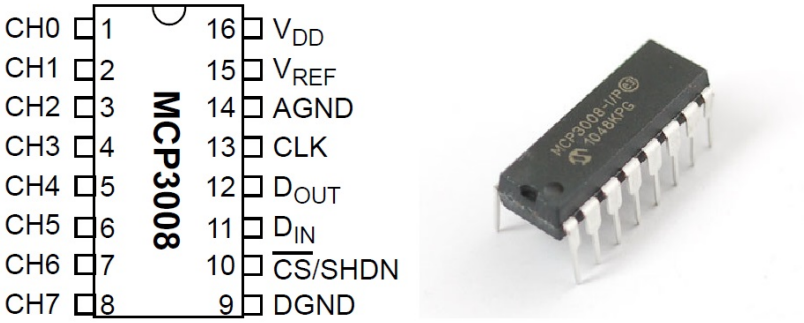


**Abbildung 2.3:** MCP3008 A/D-Wandler mit 8 Eingängen für Sensoren CH0-CH7

## 2.4 Bussysteme

In der Computerarchitekture ist ein Bus ein System, das Daten zwischen einzelnen Computerbestandteile überträgt.

### 2.4.1 SPI

Der Serial Peripheral Interface (SPI) Bus ist eine von Motorola entwickelte, synchrone serielle Kommunikationsschnittstelle, welche für die Übertragung von Daten über kurze Distanzen entworfen ist. Häufige Anwendung findet der SPI in Embedded Systems.

Eine Kommunikation über SPI erfolgt über einen SPI Master und einen SPI Slave. Der SPI Master startet eine Kommunikation in dem er die SS Verbindung auf 0 Volt zieht und eine clock signal mit einer bestimmten clock frequency aktiviert. Die zu sendenden Signale werden dann über den MOSI (Master Out Slave In) übertragen, in der Frequenz die von SCLK vorgeben ist. Der SPI ist ein single-master Protokoll, da lediglich ein Master die Kommunikation über alle Slaves steuert.

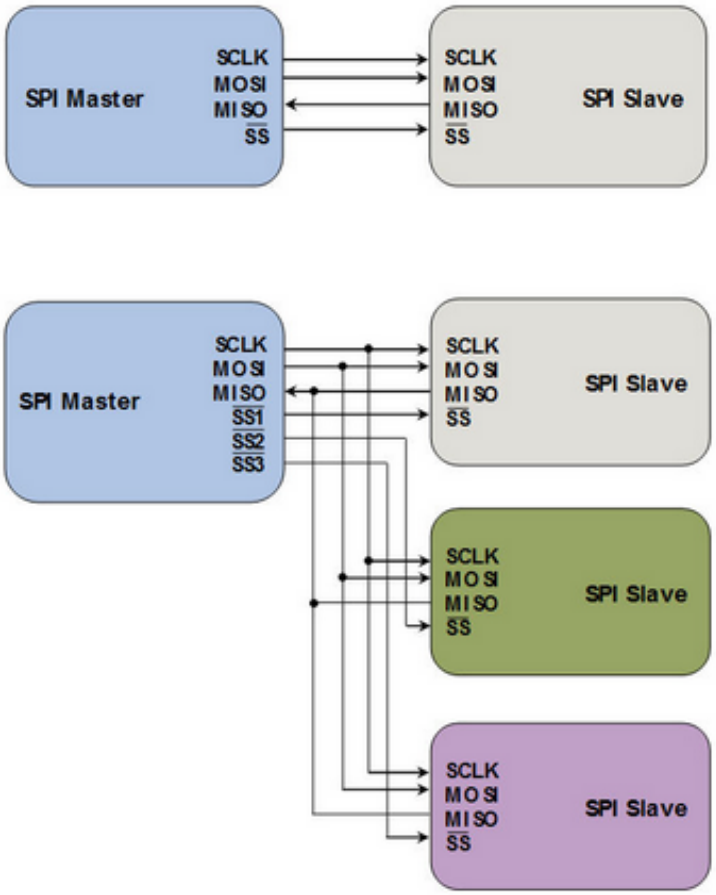


**Abbildung 2.4:** SPI Kommunikation erfolgt über einen Puls der von SCLK gegeben ist und den Master und Slave synchronisiert. Über MOSI (Master IN Slave OUT) sendet der Master Signale an den Slave. Quelle: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols

### 2.4.2 I2C

I2C ist ein multi-master Protokoll mit zwei Datenleitungen. Über Serial Data (SDA) werden die Daten übertragen. Der SCL (Serial Clock) gibt die Frequenz der Datenübertragung an und synchronisiert somit die Kommunikationspartner. Jedes Gerät, welches an den Bus angeschlossen ist bekommt eine 7 -bit slave address. Es können beliebig viele Geräte an den I2C angeschlossen werden. Die Daten werden in 8-bit Paketen übertragen. Die Daten Raten können 100 kbps, 400 kbps oder 3.4 Mbps betragen. Diese verschiedenen Übertragungsraten werden auch standard mode, fast mode und high speed mode genannt.

Physikalisch besteht der I2C Bus aus zwei Leitungen und einer Ground Leitung (FIXME:siehe Abbildung x). Die aktiven Leitungen sind bi-direktional. Im I2C Standard ist definiert, dass jedes I2C Gerät welches Daten übersenden will, die

Rolle eines Masters annimmt und alle anderen Geräte in diesem Moment Slaves sind.

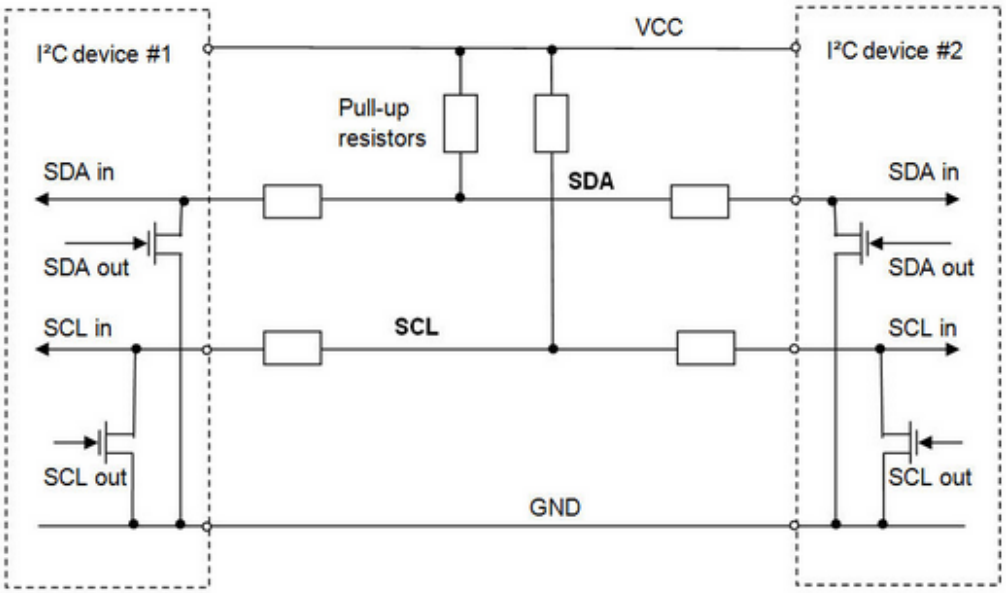


**Abbildung 2.5:** Zwei I2C Geräte sind über SDA und SCL verbunden. Diese beiden Leitungen sind über pull-up Widerständen mit VCC verbunden. Quelle: http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/

### 2.4.3 Sensoren

Unter Sensoren versteht man Komponenten, in denen eine physikalische oder chemische Veränderung in ein geeignetes Nutzsignal erfasst oder gemessen wird. **?**

Im folgenden werden die Sensoren beschrieben, die in dieser Arbeit in Betrieb genommen werden.

#### *NXT Lightsensor*

Der NXT Lichtsensor misst wie viel Licht in den Sensor fällt. Dies geschieht über einen light-dependent resistor (LDR), der ein variabler Widerstand ist der niedriger wird, je mehr Licht in hereinfällt. Der Lichtsensor verfügt ebenfalls über eine externe Lichtquelle. Mit dieser ist es möglich auch in dunklen Lichtverhältnissen zwischen Oberflächen zu unterscheiden.

**Abbildung 2.6:** NXT Lightsensor mit dem LDR oben und einer externen Lichtquelle unten.

### 2.4.4 Tastsensor

Der Tastsensor ist ein einfacher Sensor der die Werte 1 für gedrückt und 0 für nicht gedrückt darstellen kann. In dieser Arbeit wird dieser Sensor beispielsweise dazu verwendet den Roboter zu starten. Ist der Schalter gedrückt, wird ein Startskript ausgeführt.

**Abbildung 2.7:** NXT Tastsensor

## 2.5 PWM

## 2.6 GPIO

Ein General Purpose Input/Output (GPIO) ist eine einfache digitale Schnittstelle mit der eine Folge von einsen und nullen übertragen werden können. Der Raspberry PI 2 bietet 40 GPIOs, welche jeweils als Eingabe oder Ausgabe Pins geschaltet werden können.

# Kapitel 3

# Implementierung

## 3.1 Herausforderungen beim Auslesen der Sensoren

Den ersten Sensor den wir ausgelesen haben war der Tastsensor. Diesen haben wir über einen GPIO direkt mit dem Raspberry PI verbunden. Über das öffnen des GPIOs via RPi.GPIO konnten wir auslesen, ob der Tastsensor gedrückt ist oder nicht.

Die Idee war den Lichtsensor auch über GPIO auslesen zu können. Eine solche Testkonfiguration zeigte allerdings, dass wir keine sinnvollen Werte über den GPIO auslesen können.

Eine weitere möglichkeit sahen wir darin, über I2C den Sensor anzuschließen. Leider zeigte auch diese Option keinen Erfolg, da die NXT Sensoren eine eigenes I2C Protokoll verwenden, welches nicht kompatibel ist mit dem I2C Bus am Raspberry PI.

Eine funktionierende Lösung sah so aus, den Sensor mit einem A/D-Wandler zu verbinden und dann per GPIO die digital abgetasteten Daten zu empfangen. Da wir keinen A/D-Wandler zur Verfügung hatten, bauten wir uns einen eigenen A/D-Wandler. (vgl. Abbildung xx FIXME). Der Nachteil dieser Schaltung ist, dass lediglich ein Sensor ausgelessen werden kann. Um den Anforderungen gerecht zu werden und einen Parqour wie in der Robotik Vorlesung zu durchlaufen ist der Einsatz von mehreren Sensoren notwendig. Dies erreichten wir mit dem Einsatz eines MCP3008 A/D-Wandler. Mit diesem ist das Auslesen von bis zu 8 Sensoren gleichzeitig möglich. (vergleiche xx FIXME).

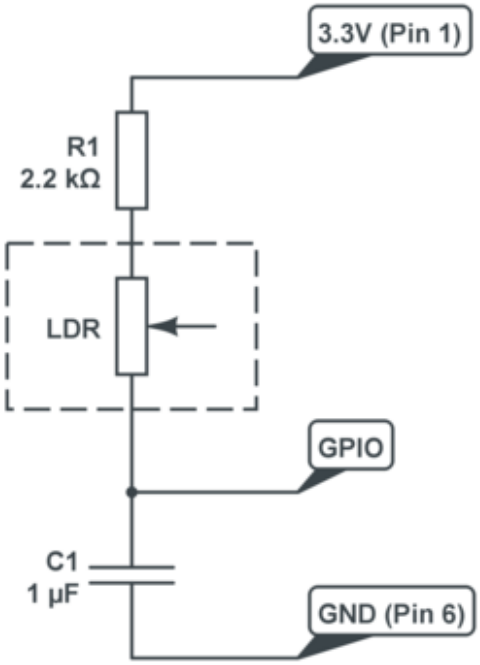


**Abbildung 3.1:** A/D-Wandler realisiert über einen LDR der in Reihe geschaltet ist mit einem 2.2 kOhm Widerstand und 1$\mu$F Kondensator

## Kapitel 4

# Fazit

In dieser Arbeit haben wir es geschafft LegoPI zu bauen. Einen Roboter der mit allen analogen Sensoren des Lego Mindstorms Bausatzes kompatibel ist. LegoPI ist ebenso in der Lage zwei Motoren zu steuern, um die Fortbewegung des Roboters zu ermöglichen. Wir konnten einen einfachen Parqour durchlaufen, in welchem der Roboter einer schwarzen Linie folgt und am Ende an einer Wand stehen bleibt. Das Signal zum Stehenbleiben wird hier über einen Tastsensor ausgelöst, welcher an der vorderen Front des Roboters angebracht ist. FIXME: Bild?

# Kapitel 5

# LegoPi-Software

Für die einfache Benutzung des LegoPi-Roboters wurde eine individuelle API entwickelt. Um den Wechsel zwischen der LeJOS Bibliothek, welche direkt auf den Lego Mindstorm-Komponenten ausgeführt wird, und der für den RaspberryPi entwickelten API so einfach wie möglich zu gestalten, ist diese stark an die LeJOS Software-Architektur angelehnt. Entwickelt wurde die API in der Programmiersprache Python und mithilfe der Bibliotheken RPi.GPIO, welche eine einfache Verwendung der RaspberryPi GPIO-Pins ermöglicht, sowie SpiDev um den SPI-Bus ansprechen zu können. Mithilfe der entwickelten API ist es möglich bis zu zwei Motoren und acht Sensoren zu steuern.

## 5.1 Software-Komponenten

Die LegoPi Software besteht aus vier grundlegenden Komponenten die einen einfachen Einstieg in die Benutzung gewährleisten.

### 5.1.1 LegoPi.py - API

In dem Python-Modul LegoPi sind alle in Abbildung 5.1 gezeigten Funktionalitäten eingebaut. Dieses Modul kann vom Entwickler genutzt werden um den individuellen Code für die Steuerung des LegoPi-Roboters zu kreieren.

### 5.1.2 Run.py - Individueller Code des Entwicklers

Die *Run.py*-Datei enthält den individuellen Code des entwicklers. Grund für die Vorgabe eines Dateinamens ist der in Unterabschnitt 5.1.3 beschriebene ExecDaemon.

### 5.1.3 ExecDaemon.py - Automatischer Start des Entwickler-Codes

Der ExecDaemon wird beim Starten des Systems automatisch ausgeführt. Er läuft im Hintergrund und wartet auf ein Signal des an *Channel 3* angeschlossenen Tastsensors. Sobald dieses Signal empfangen wird, startet der ExecDaemon den in *Run.py* eingetragenen Code. Dies ermöglicht es den Entwickler den RaspberryPi vom Netzwerk zu trennen und den Code einfach zu testen, ohne ihn, zum Beispiel, direkt über eine SSH-Verbindung ausführen zu müssen.

### 5.1.4 startup.sh - Vorbereitung des Systems

Um das System unmittelbar nach Systemstart nutzen zu können, wird automatisch das Bash-Script *startup.sh* ausgeführt. Dieses wird beim Neustart des Systems, durch einen in Crontab konfigurierten Job, automatisch ausgeführt. Dieses Script stellt zum einen sicher, dass der SPI-Bus aktiviert ist und zum anderen, dass der ExecDaemon ausgeführt wird.

## 5.2 API

Die API ist sehr einfach gestaltet und besteht aus wenigen Klassen um die verschiedenen Sensoren und Motoren ansprechen zu können.

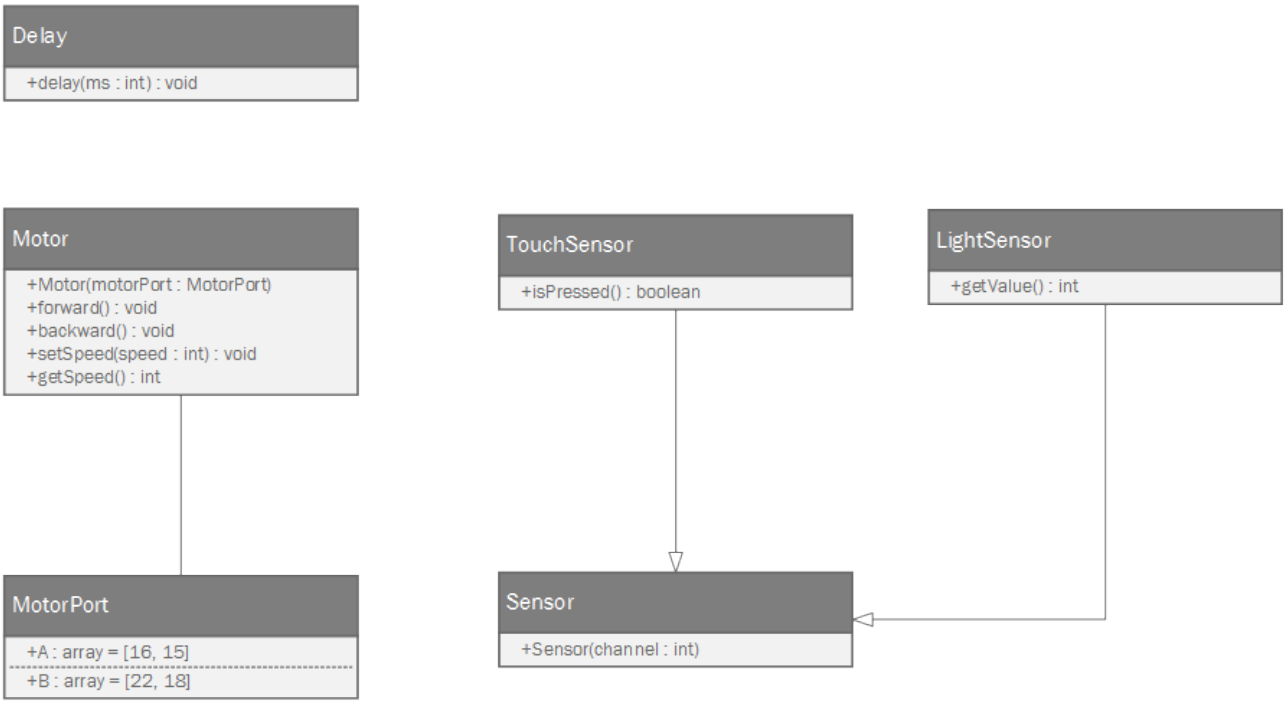


**Abbildung 5.1:** API - Klassendiagramm

#### *Motoren*

Die Motoren werden mithilfe eines übergebenen MotorPorts instanziiert. Dieser MotorPort beschreibt, an welchen GPIO Pins der Motor angeschlossen ist. Durch *forward()* und *backward()* lassen sich die Motoren vorwärts und rückwärts bewegen. Ebenfalls ist es möglich die Geschwindigkeit individuell anzupassen. Durch *setSpeed(speed: int)* lässt sich eine Geschwindigkeit zwischen 0 und 100 angeben.

#### *Sensoren*

Sensoren hingegen gibt es in zwei unterschiedlichen Spezialisierungen, dem Tastsensor (TouchSensor), sowie dem Lichtsensor (LightSensor). Beide sind eine Erweiterung der Klasse Sensor, welche Grundlegende Aufgaben übernimmt die für alle Sensor-Arten benötigt werden. Der Lichtsensor liefert mit seiner einzigen öffentlichen Methode *getValue()* einen numerischen Wert zurück, der die Lichtreflektion beschreibt. Je geringer die Reflektion, desto kleiner ist auch der Rückgabewert. Zu beachten ist hier, dass durch unterschiedliche Lichtverhältnisse bedingt, verschiedene Werte zurückgeliefert werden können. Deshalb ist zu empfehlen, nicht nach einem konkreten Wert zu suchen, sondern ein gewisses Delta zuzulassen. Der Tastsensor bestitzt ebenfalls lediglich eine öffentliche Methode *isPressed()*, welche einen booleschen Wert zurückliefert, ob der Tastsensor aktiviert wurde oder nicht. Sensoren werden mit einem übergebenen *Channel* instanziiert. Dieser ist äquivalent zu den Channels am AD-Wandler, und beschreibt wo der Sensor angeschlossen ist. Da bist zu acht Sensoren genutzt werden können, wird eine Zahl zwischen null und sieben erwartet.

### 5.2.1 Beispielanwendung

Nachfolgend eine einfache Beispielanwendung, um das Verständnis der Programmierschnittstelle zu fördern.

```
# instanziiere Lichtsensoren für Channel 0 und 1
lightSensorA = LightSensor(0)
lightSensorB = LightSensor(1)
# instanziiere Tastsensor für Channel 2
touchSensor = TouchSensor(2)

# instanziiere Motoren A und B
motorA = Motor(MotorPort.A)
motorB = Motor(MotorPort.B)
```

```
# setze Geschwindigkeit beider Motoren auf 30
motorA.setSpeed(30)
motorB.setSpeed(30)

# fahre mit beiden Motoren vorwärts
motorA.forward()
motorB.forward()

# solange der Tastsensor nicht gedrück ist
while not touchSensor.isPressed():
  # warte 50 ms
  delay(50)
  # Ist der Wert von LichtsensorA geringer 150
  if lightSensorA.getValue() <= 150:
    # stoppe MotorA
    motorA.stop()
  # Ist der Wert größer 80, fahre vorwärts
  else:
    motorA.forward()

  # Ist der Wert von LichtsensorB geringer 80
  if lightSensorB.getValue() <= 150:
    motorB.stop()
  else:
    # Ist der Wert größer 80, fahre vorwärts
    motorB.forward()

# Stoppe Motor A und B
motorA.stop()
motorB.stop()
```