Institut für Softwaretechnologie

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 230

Bachelorarbeit

Entwicklung eines Simulators für Adaptive-Cruise-Control Systeme mit LEGO Mindstorm Robotern

Dennis Maseluk

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. rer. nat. Stefan Wagner
Betreuer/in:	Asim Abdulkhaleq, M. Sc.
Beginn am:	6. Mai 2015
Beendet am:	5. November 2015
CR-Nummer:	B.2.2, D.2.1, D.2.2, I.2.9

Kurzfassung

Das Adaptive Cruise Control System ist ein fahrerunterstützendes System in Fahrzeugen und wird von vielen Automobilherstellern in ihren Produkten verwendet. Es sorgt dafür das ein Fahrzeug eine gewünschte Geschwindigkeit konstant hält und wenn nötig durch automatisches Bremsen und Beschleunigen einen Sicherheitsabstand zum vorausfahrenden Fahrzeug einhält. Im Rahmen dieser Bachelorarbeit wird dieses System für zwei Lego Mindstorms Roboter implementiert. Die Implementierung besteht aus zwei Programmen. Das erste Programm ist für den Lego Mindstorms Roboter, der das Adaptive Cruise Control System verendet, und das andere Programm ist für den Roboter der vorausfährt. Für das Programm mit dem Adaptive Cruise Control wurden zwei unterschiedliche Algorithmen verwendet. Ein Algorithmus arbeitet mit der Information, wie schnell das vorausfahrende Fahrzeug fährt, und der andere Algorithmus arbeitet ohne diese Information. Diese Programme sollen das Adapive Cruise Control System simulieren und in Studien für Sicherheitsanalysen verwendet werden.

Abkürzungsverzeichnis

ACC	Adaptive Cruise Control
CC	Cruise Control
RPM	Rotation pro Minute

Begriffsverzeichnis

ACCcar	Das Fahrzeug mit dem ACC System.
accelerationratio	Die Beschleunigung des Fahrzeugs mit dem ACC System.
currentspeed	Die aktuelle Geschwindigkeit des Fahrzeuges mit dem ACC System.
deltaX	Der Abstand, der vom Fahrzeug mit dem ACC System benötigt wird, um sich der Geschwindigkeit des vorausfahrenden Fahrzeugs bis zum Sicherheitsabstand anzupassen.
desiredspeed	Die gewünschte Geschwindigkeit, die das Fahrzeug mit dem ACC System fahren soll.
desiredTimeGap	Eine timeGap die einen gewissen Abstand zum vorausfahrenden Fahrzeug gewährt.
frontcar	Das Fahrzeug das vor dem Fahrzeug mit dem ACC System fährt.
frontspeed	Die Geschwindigkeit des vorausfahrenden Fahrzeugs.
time	Die Dauer der Simulation.
minimumspeed	Die Mindestgeschwindigkeit, die benötigt wird, um das ACC System zu aktivieren.
safedistance	Ein Abstand zwischen dem Fahrzeug mit dem ACC System und dem vorausfahrenden Fahrzeug. Wenn dieser Abstand unterschritten wird, muss das Fahrzeug mit dem ACC System sofort zum Stillstand kommen.
timeGap	Die Zeit die benötigt wird die Strecke zwischen dem Fahrzeug mit dem ACC System und dem vorausfahrenden Fahrzeug abzufahren.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Dieses Kapitel soll einen Überblick über den Aufbau und das Thema der Bachelorarbeit verschaffen. Als erstes wird im Abschnitt Überblick der größere Zusammenhang von fahrerunterstützenden Systemen wie Adaptive Cruise Control beschrieben. Im nächsten Abschnitt wird die Motivation erläutert und danach die Zielstellung der Bachelorarbeit dargestellt. Der letzte Abschnitt erläutert den Aufbau dieses Dokumentes, um einen Gesamtüberblick zu verschaffen.

1.1 Überblick

Der Mensch ist durch seine begrenzte Reaktionszeit und fehleranfälliges Verhalten ein Grund für Unfälle im Straßenverkehr. Um die Sicherheit im Straßenverkehr zu erhöhen werden immer mehr Advanced Driver Assistance Systeme [GP05] entwickelt, die den Fahrer eines Fahrzeuges in unfallkritischen Situationen unterstützen sollen. Diese Systeme können das Risiko eines Unfalls eines menschlichen Fahrers durch unbeabsichtigtes oder unachtsames Verhalten minimieren.
Ein Beispiel ist das Adaptive Cruise Control System, das nachfolgend als ACC abgekürzt wird. Das ACC System kann Auffahrunfälle verhindern, indem es dafür sorgt das ein gewisser Sicherheitsabstand zum vorausfahrenden Fahrzeug eingehalten wird.

1.2 Motivation

Der ACC Simulator soll als open source Anwendung veröffentlicht werden und in Studien für Sicherheitsanalysen verwendet werden.

1.3 Ziel

Das Ziel der Arbeit ist es, eine Simulation von ACC und deren Erweiterung ACC Stop and Go zu realisieren. Für die Simulation sollen zwei Lego Mindstorm Roboter verwendet werden. Diese müssen in verschiedenen Szenarien die Simulation durchführen können.

1.4 Aufbau der Bachelorarbeit

Das erste Kapitel der Bachelorarbeit ist die Einleitung und soll einen Überblick über das Thema verschaffen. Es beschreibt die Motivation und das Ziel der Arbeit. Das zweite Kapitel Grundlagen befasst sich mit den fahrerunterstützenden Systemen und den Lego Mindstorms Robotern, auf die ausführlich eingegangen wird. Das nächste Kapitel Analyse und Spezifikation umfasst die Modellierung und den Aufbau des ACC Simulators. Es werden ebenfalls die technischen Voraussetzungen beschrieben. Danach wird in Detail auf die einzelnen Formeln des ACC Simulatots im Kapitel Algorithmen und Formeln eingegangen. Das fünfte Kapitel Evaluation vergleicht zwei Algorithmen innerhalb des ACC Simulators und visualisiert diese. Das Kapitel Implementierung befasst sich mit der Entwicklungsumgebung und dem Quellcode des ACC Simulators. Im Kapitel Installation/User Guide wird erklärt, was man für den ACC Simulator benötigt und wie man ihn startet. Zusätzlich wird in die Bedienung des ACC Simulators eingeführt. Das letzte Kapitel ist die Zusammenfassung der Bachelorarbeit. Nach dem letzten Kapitel folgen zwei Anhänge, die Bauanleitung der Lego Mindstorms Roboter und diverse Streckenvorlagen zum ausdrucken.

2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der fahrerunterstützenden Systeme und den Lego Mindstorms Robotern. Es gibt verschiedene fahrerunterstützende Systeme, sowie verschiedene Lego Mindstorms Serien Versionen und Komponenten. Deshalb wird nachfolgend auf die Grundlagen dieser fahrerunterstützenden Systeme und der Lego Mindstorms Versionen in Detail eingegangen, die in dieser Bachelorarbeit Verwendung finden.

2.1 Cruise Control

Cruise Control [Gol] ist ein System in Kraftfahrzeugen, das es ermöglicht die Geschwindigkeit eines fahrenden Fahrzeugs konstant zuhalten. Es wird über drei Knöpfe gesteuert: resume, on und off.
Der Fahrer eines Fahrzeugs mit einem Cruise Control System, kann das System während einer Fahrt manuell aktivieren. Um das Cruise Control System zu deaktivieren, muss der Fahrer nur das Gaspedal, die Bremse oder den off Knopf betätigen. Wenn das System deaktiviert wird, wird die aktuelle Geschwindigkeit abgespeichert. Durch den resume Knopf beschleunigt oder verlangsamt das System das Fahrzeug auf die abgespeicherte Geschwindigkeit zurück. Die meisten Cruise Control Systeme erlauben es nicht das System zu aktivieren, bevor eine Mindestgeschwindigkeit (zum Beispiel 40 km/h) erreicht wird.
Ein Fahrer kann durch Cruise Control entspannter fahren, da er nicht ständig auf den Tachometer schauen muss um eine bestimmte Geschwindigkeit einzuhalten. Ein weiterer Vorteil ist zudem ein effizienterer Treibstoffverbrauch.

2.2 Adaptive Cruise Control

Das Adaptive Cruise Control System [Gur04] ist eine Verbesserung des Cruise Control Systems. Wie beim CC System sorgt das ACC System, wenn es aktiviert wird, dass das Auto mit einer vorgegebenen konstanten Geschwindigkeit fährt und zusätzlich einen Sicherheitsabstand zum vorausfahrenden Fahrzeug beibehält. Ein Fahrzeug mit einem ACC System besitzt ein Radar oder einen Lasersensor mit dem es den Abstand und die Geschwindigkeit eines vorausfahrenden Fahrzeugs messen kann. Dadurch kann das ACC System den benötigten Sicherheitsabstand zwischen zwei Fahrzeugen berechnen und einhalten.
Das Fahrzeug mit dem ACC System würde, um den Sicherheitsabstand einzuhalten, automatisch die Geschwindigkeit durchs Abbremsen verringern, wenn das vorausfahrende Fahrzeug zu nah ist. Das ACC System ohne Stop & Go Funktion würde die Geschwindigkeit, aber nur bis zur Mindestgeschwindigkeit abbremsen, die nötig ist das System zu aktivieren. Wenn der Weg vor dem Fahrzeug wieder

frei ist, würde das ACC System auf die voreingestellte Geschwindigkeit wieder zurück beschleunigen. Dieses System wurde konzipiert um die Fahrsicherheit und den Fahrkomfort zu erhöhen.

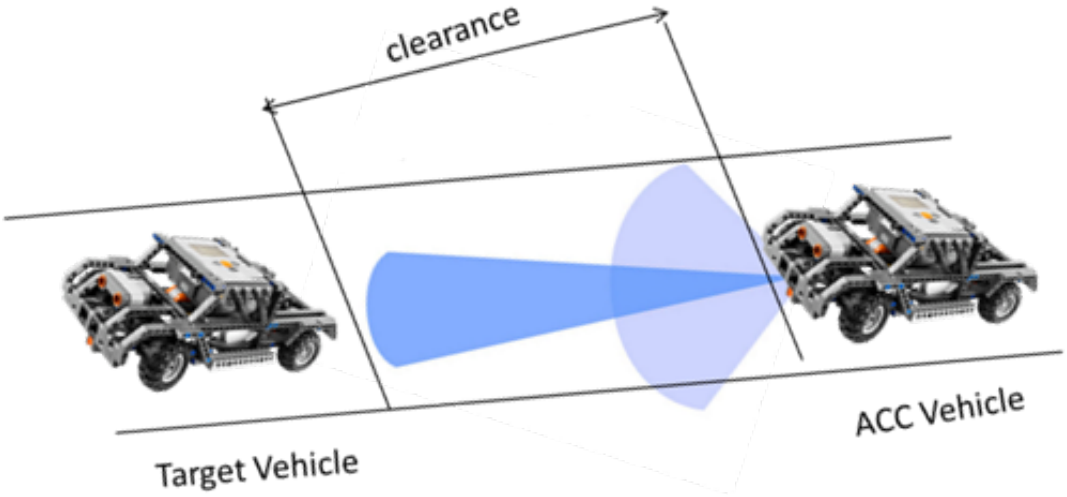


Abbildung 2.1: ACC Überblick

In Abbildung 2.1 ist ein Überblick für ein ACC System dargestellt. Ein Fahrzeug (ACC Vehicle) fährt einem anderen Fahrzeug (Target Vehicle) mit einem gewissen Abstand hinterher. Das hinterherfahrende Fahrzeug (ACC Vehicle) misst den Abstand (clearance) zwischen ihnen mit einem Radar.

2.3 ACC Stop & Go

Das ACC System mit Stop & Go-Funktion [VNA00] ist in der Lage, wie das normale ACC, eine bestimmte Geschwindigkeit zu halten und dafür zu sorgen, dass ein gewisser Sicherheitsabstand zum vorausfahrenden Fahrzeug eingehalten wird. Das System kann zudem durch selbsttätiges Bremsen, das Fahrzeug zum vollständigen Stillstand bringen, wenn die Situation es erfordert. Wie zum Beispiel, das vorausfahrende Fahrzeug hält an. Sobald das davorstehende Fahrzeug wieder losfährt, kann das System das angehaltene Fahrzeug automatisch wieder losfahren lassen und dem vorausfahrenden Fahrzeug wieder folgen.
Es existieren Versionen dieses Systems, die es ermöglichen festzulegen, wie zügig das System beschleunigen soll. Dadurch kann der Fahrer eines solchen Fahrzeugs, die Beschleunigung festlegen, die ihm am komfortabelsten ist.

2.4 Lego Mindstorms

Die Lego Mindstorms[1] Serie besteht aus Software und Hardware, die es ermöglicht eigene Roboter zu bauen und zu programmieren. Die Roboter können über sogenannte programmierbare Bricks auf spezielle Motoren oder Sensoren zugreifen und steuern.
1998 kam die erste Generation der Lego Mindstorms Roboter auf dem Markt mit dem Namen RCX (Robotic Command eXplorers). Das Nachfolgesystem NXT erschien 2006 und löste das Vorgängermodell ab. Die neuste Generation mit dem Namen EV3 kam 2013 auf den Markt und übertraf das NXT System mit zahlreichen Neuerungen.
Es gibt viele verschiedene Lego Mindstorms EV3 Sets, die mit unterschiedlichen Komponenten ausgeliefert werden. Für die Bachelorarbeit wurde das Lego Mindstorms Education EV3 Core Set[2] verwendet.

Abbildung 2.2: Lego Mindstorms Education EV3 Core Set

2.5 EV3

Der ACC Simulator läuft nur auf den Lego Mindstorms EV3 Robotern, deshalb geht dieser Abschnitt näher auf das EV3 Modell ein. Es folgen Informationen und technische Details zu den verschiedenen verwendeten Bauelementen. Der Zusammenbau aller Komponenten ist im Anhang A beschrieben.

[1] http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com
[2] https://education.lego.com/en-us/lego-education-product-database/mindstorms-ev3/45544-lego-mindstorms-education-ev3-core-set/

EV3 Brick:
Der EV3 Brick[3] ist das zentrale Element eines Lego Mindstorms EV3 Roboter. Er ist programmierbar und kann andere Hardwarekomponenten steuern. Von diesem zentralen Element des Roboters wird die ACC Simulator Software ausgeführt. Er verfügt sechs Tasten, die mit verschiedenen Farben beleuchtet werden können.

Abbildung 2.3: EV3 Brick

ARM 9 Prozessor
16 MB Flash-Speicher
SD-Karten Slot
Anschlussboxen für 4 Sensoren und 4 Motoren
Bluetooth 2.1

Tabelle 2.1: Technische Daten zum EV3 Brick

Großer Servomotor:
Der große EV3 Servomotor[4] wird für den Antrieb der Räder der Fahrzeuge im ACC Simulator verwendet. Pro Fahrzeug werden zwei große Servomotoren benötigt, um die Fahrzeuge zu lenken.

[3] http://shop.lego.com/en-US/EV3-Intelligent-Brick-45500
[4] http://shop.lego.com/de-DE/Gro%C3%9Fer-EV3-Servomotor-45502?fromListing=listing

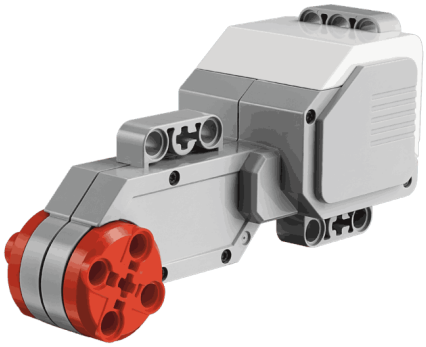


Abbildung 2.4: EV3 großer Servomotor

160 - 170 RPM (Rotation per Minute)
Tacho feedback (1 Grad Auflösung)

Tabelle 2.2: Technische Daten zum EV3 großen Servomotor

Ultraschallsensor:
Der EV3 Ultraschallsensor[5] wird im ACC Simulator für die Abstandsmessung verwendet und wird nur für ein Fahrzeug benötigt.

Abbildung 2.5: EV3 Ultraschallsensor

[5] http://shop.lego.com/de-DE/EV3-Ultraschallsensor-45504

Distanz 3 - 250 cm (weiter als 255 cm = kein Objekt)
Genauigkeit: +/- 1 cm
Abfragemöglichkeit nach anderen Ultraschallsensoren Sensoren
LED Beleuchtung
Auto-Identifikation

Tabelle 2.3: Technische Daten zum EV3 Ultraschallsensor

Farbsensor:
Falls die Simulation einem Weg nachfahren soll, wird ein EV3 Farbsensor[6] pro Fahrzeug benötigt. Der Farbsensor kann entweder Farben erkennen oder die Helligkeit messen. Im ACC Simulator ist der Farbsensor auf Helligkeit eingestellt.

Abbildung 2.6: EV3 Farbsensor

Kann acht Farben erkennen
Abtastrate 1 kHz
Auto-Identifikation

Tabelle 2.4: Technische Daten zum EV3 Farbsensor

[6] http://shop.lego.com/de-DE/EV3-Farbsensor-45506?fromListing=listing

Vergleich zwischen EV3 und NXT Brick:
Dieser Vergleich der technischen Details zwischen dem EV3 Brick und NXT Brick (Vorgängermodell) zeigt auf, dass es sich lohnt mit dem neusten Modell der Lego Mindstorms Robotern zu arbeiten. Falls man einen EV3 Brick besitzt, aber keine EV3 Komponenten wie zum Beispiel der EV3 Ultraschall-sensor. Dann wäre es möglich durch wenige Änderungen im Code des ACC Simnulators mit NXT Komponenten zu arbeiten, da der EV3 Brick zu NXT-Komponenten kompatibel ist.

	EV3	**NXT**
Prozessor	ARM9 300MHz 16MB Flash 64MB RAM	Atmel 32-Bit ARM AT91SAM7S256 48MHz 256 KB FLASH-RAM 64 KB RAM
Betriebssystem	LINUX	Properitär
Motoranschlüsse	4, digital	3 digital
Sensoranschlüsse	4 Analog Digital: bis zu 460.8 Kbit/s	4 Analog Digital: 9600 bit/s (IIC)
Display	LCD Matrix, monochrom 178 x 128 Pixel	LCD Marix, monochrom 100 x 64 Pixel

Tabelle 2.5: Vergleich zwischen EV3 und NXT Brick

3 Analyse und Spezifikation

Dieses Kapitel befasst sich mit den Voraussetzungen und dem Aufbau des ACC Simulators. Es wird im Detail erklärt was der ACC Simulator ist und wie er arbeitet.

3.1 ACC Simulator

Der ACC Simulator besteht aus zwei Programmen. Das eine Programm ist für das Fahrzeug das vorausfährt und wird nachfolgend als frontcar bezeichnet. Das andere Programm ist für das Fahrzeug, das dem frontcar hinterherfährt und wird nachfolgend als ACCcar bezeichnet. Das Programm für das ACCcar ist das eigentliche Programm auf dem der ACC Algorithmus abläuft.
Beide Programme haben unterschiedliche Funktionen und Konfigurationen, die nach dem Start der Programme ausgewählt werden können. Hierfür navigiert man sich durch das Menü des EV3 Bricks (siehe: User Guide) und stellt die gewünschte Simulation ein.

Ein Überblick über die möglichen Funktionen des ACCcars:

	Auswahl	Erklärung
Time	0..3000	Dauer der Simulation in Sekunden
DesSpeed	0..40	Gewünschte zufahrende Geschwindigkeit (cm/s)
Modus	ACC/ACC Stop & Go	ACC oder die Erweiterung Stop & GO starten
Line Follow	Yes/No	Soll das Fahrzeug einer Linie nachfahren
Algorithm	Known/Unknown	Ist die Geschwindigkeit des frontcars bekannt
FrontSpeed*	0..40	Geschwindigkeit des frontcars (cm/s)

Tabelle 3.1: Überblick über die Konfigurationen des ACCcars

* nur sichtbar, wenn known bei Algorithm ausgewählt wurde

Ein Überblick über die möglichen Funktionen des frontcars:

	Auswahl	Erklärung
Time	0..3000	Dauer der Simulation in Sekunden
Speed	0..40	Gewünschte zufahrende Geschwindigkeit (cm/s)
Line Follow	Yes/No	Soll das Fahrzeug einer Linie nachfahren
Scenario	1..3	Welches Szenario soll gefahren werden
Stoptime**	0..5	Dauer der Standzeit in Sekunden
MinSpeed***	0..MaxSpeed	Untere Grenze der zufälligen Geschwindigkeit
MaxSpeed***	MinSpeed..40	Obere Grenze der zufälligen Geschwindigkeit

Tabelle 3.2: Überblick über die Konfigurationen des frontcars

** nur sichtbar, wenn 2 bei Scenario ausgewählt wurde
*** nur sichtbar, wenn 3 bei Scenraio ausgewählt wurde

Unter der Auswahl Scenario kann man die möglichen Fahrsequenzen auswählen:

Auswahl 1: Fahre mit der Geschwindigkeit, die bei Speed eingestellt wurde, bis die Zeit abläuft.

Auswahl 2: Fahre mit der Geschwindigkeit, die bei Speed eingestellt wurde, bis die Hälfte der Zeit abgelaufen ist. Dann bleibe solange stehen, bis die Stoptime abgelaufen ist. Fahre wieder los mit der Geschwindigkeit, die bei Speed eingestellt wurde, bis der Rest der Zeit abläuft.

Auswahl 3: Fahre mit der Geschwindigkeit, die bei Speed eingestellt wurde, bis ein Drittel der Zeit abgelaufen ist. Dann fahre mit einer zufälligen Geschwindigkeit die zwischen MinSpeed und MaxSpeed liegt weiter. Wenn wieder ein Drittel der Zeit abgelaufen ist, fahre mit einer zufälligen Geschwindigkeit die zwischen MinSpeed und MaxSpeed liegt weiter, bis die Zeit abgelaufen ist.

3.2 Spezifische Voraussetzungen

Um die ACC Simulation durchzuführen werden zwei EV3 Lego Mindstorms Roboter benötigt (Zusammenbau der Autos, siehe Anhang) mit folgenden technischen Voraussetzungen pro Lego Mindstorms Roboter:

- 2x Große EV3 Servomotoren
- EV3 Ultraschallsensor (nur für das ACCcar)
- EV3 Brick mit einem auf Linux basierenden Betriebssystem
- EV3 Farbsensor (falls die Roboter einer Linie nachfahren sollen)

Zusätzlich wird jeweils eine Mikro SD Karte pro Lego Mindstorms Roboter benötigt, um die Simulationsprogramme auszuführen.

3.3 Funktionale Anforderungen

Der ACC Simulator soll das ACC System und deren Erweiterung ACC Stop & Go simulieren können. Es müssen dafür zwei Programme erstellt werden, für das ACCcar und das frontcar. Um die Fahrzeuge in einer Spur zu halten, sollen die beiden Fahrzeuge, die Option haben, einer Linie zu folgen. Dafür soll ein Farbsensor verwendet werden.

3.4 Sicherheitsanforderungen

Der ACC Simulator muss in der Lage sein, den korrekten Sicherheitsabstand zu gewährleisten. Falls es doch zu einer Überschreitung des Sicherheitsabstandes kommen sollte, muss der Lego Mindstorms Roboter sofort zum Stillstand kommen. Dies gilt aber nur für den ACC Stop & Go Algorithmus, da der ACC Algorithmus ohne Stop & Go Funktion nur bis zur Mindestgeschwindigkeit abbremst und nicht zum vollständigen Stillstand kommt.

3.5 Block Diagramm

Das folgende Block Diagramm veranschaulicht die verschiedenen Komponenten, die in der Struktur des ACC Simulator vorkommen. Die Komponente Display befindet sich auf dem EV3 Brick und wird über die ACC Simulator Software gesteuert. Die Komponenten Ultraschallsensor, großer Servomotor A und großer Servomotor B befinden sich außerhalb des EV3 Brick und werden ebenfalls über die ACC Simulator Software gesteuert.

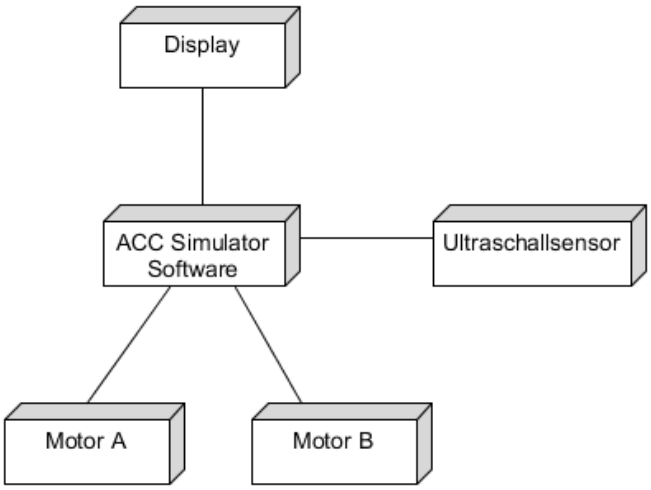


Abbildung 3.1: ACC Simulator Block Diagramm

3.6 Aktivitätsdiagramm

Das Aktivitätsdiagramm beschreibt einen Prozess aus Aktionen und Aktivitäten und zeigt grafisch das Verhalten des Programms. In der Abbildung 3.2 wird der Ablauf des ACC Simulators dargestellt.

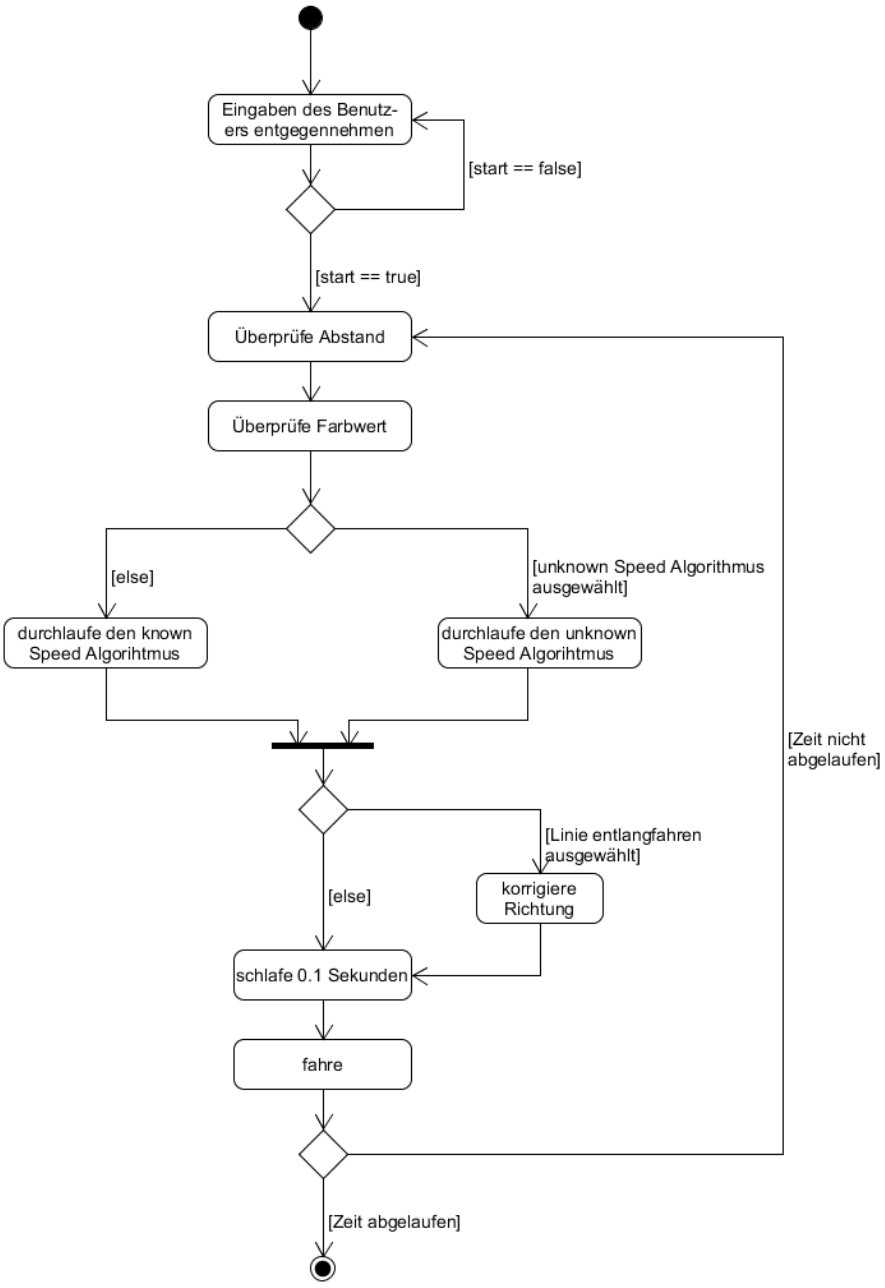


Abbildung 3.2: ACC Simulator Aktivitätsdiagramm

3.7 Use Case Diagramm

Das Use Case Diagramm visualisiert aus Anwendersicht den Systemnutzen und zeigt die Anwendungsfälle aller externen Akteure auf. Das Use Case Diagramm in Abbildung 3.3 zeigt den ACC Simulator als großes Rechteck und den Akteur, der mit System interagiert, als Figur daneben. Die Interaktionen sind als Ovale innerhalb des Rechtecks visualisiert und sind überschaubar. Der Benutzer des ACC Simulators kann vor dem Start der Simulation verschiedene Einstellungen vornehmen und dann die Simulation starten. Er kann ebenfalls jederzeit die Simulation beenden.

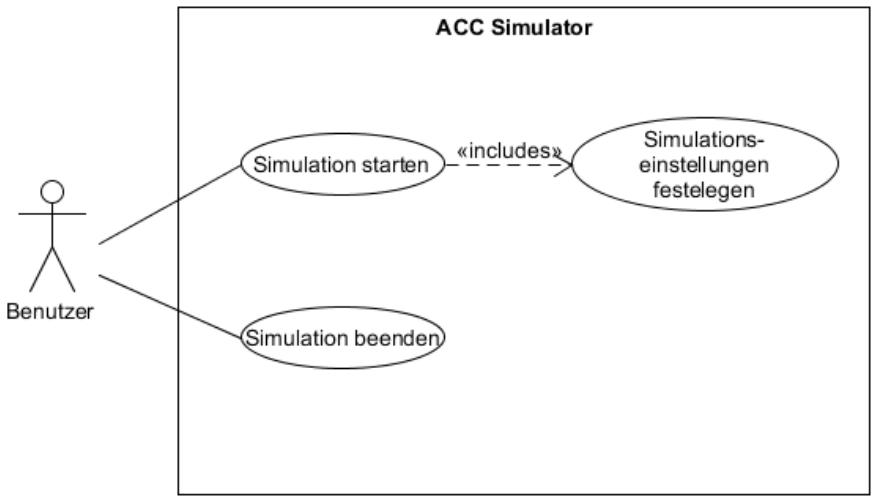


Abbildung 3.3: ACC Simulator Use Case Diagramm

3.8 Zustandsübergangsdiagramm

Das folgende Zustandsübergangsdiagramm veranschaulicht die verschiedenen Zustände, die von der ACC Software angenommen werden können. Wird die ACC Simulator Software auf dem EV3 Brick gestartet, befindet sich die Software in einem Zustand in der sie auf die Eingaben des Benutzers wartet. Der Benutzer kann in diesem Zustand verschiedene Einstellungen vornehmen, wie die Simulation ablaufen soll. Wenn der Benutzer die gewünschten Einstellungen vorgenommen hat, kann er die Simulation aktivieren. Ist die Simulation aktiviert, läuft die Simulation so lange bis die eingestellte Dauer der Simulation abgelaufen ist oder der Benutzer die Simulation abbricht. Das Programm beendet sich danach.

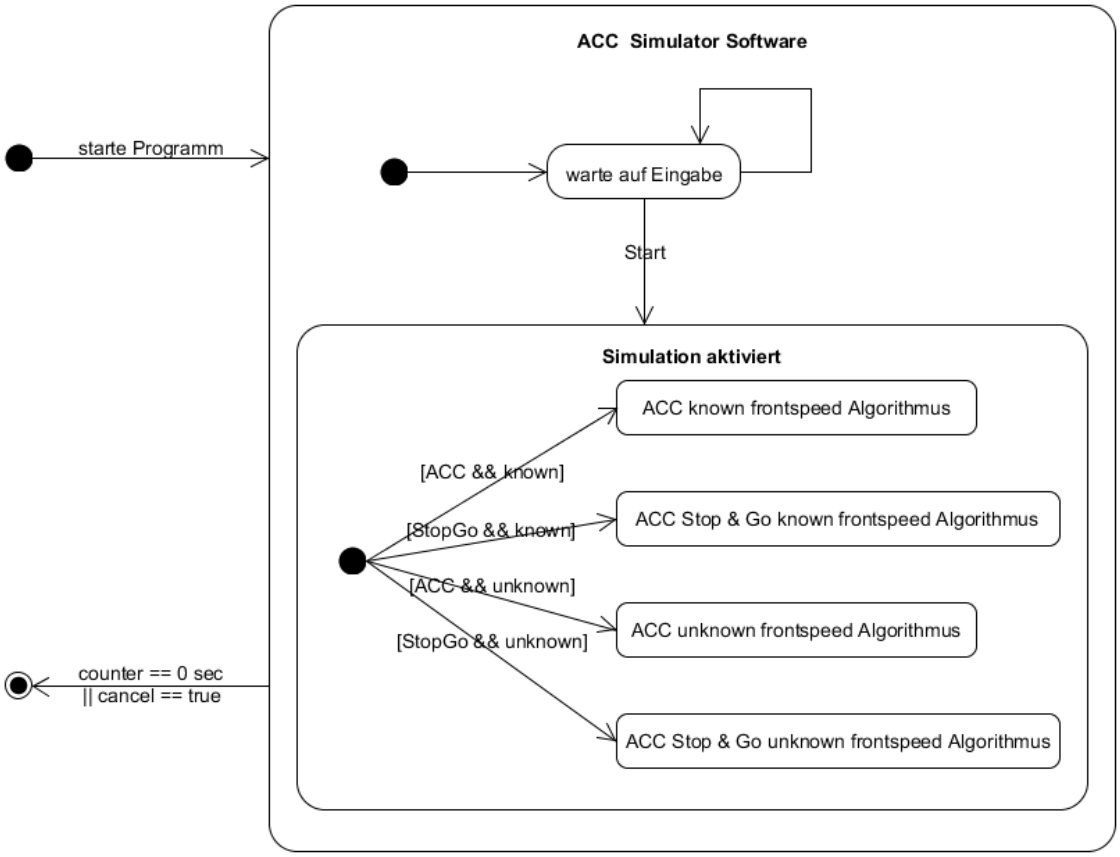


Abbildung 3.4: ACC Simulator Zustandsübergangsdiagramm

Auf die Zustände, die während der Simulation von der Software angenommen werden, werden im nächsten Abschnitt eingegangen.

3.9 Endliche Zustandsautomaten

Die folgenden endlichen Zustandsautomaten veranschaulichen die verschiedenen Zustände, die während der laufenden Simulation angenommen werden können. Zustände sind als Rechtecke visualisiert und Übergänge als Pfeile Ein Übergang von einem Zustand zum nächsten erfolgt, wenn bestimmte Bedingungen erfüllt sind.

3.9.1 ACC endliche Zustandsautomaten

Die zwei Zustandsautomaten in Abbildung 3.5 und in Abbildung 3.6 zeigen die zwei unterschiedlichen ACC Algorithmen. Einmal mit der Information der Geschwindigkeit des vorausfahrenden Fahrzeugs (known frontspeed Algorithmus) und einmal ohne die Information (unknown frontspeed Algorithmus).

Die folgenden Zustände können angenommen werden:

accoff: In diesem Zustand startet das System und wechselt sofort zum nächsten Zustand.

standby: In diesem Zustand wird das Fahrzeug solange beschleunigt, bis es die Mindestgeschwindigkeit erreicht hat. Es soll die Beschleunigung eines Fahrers imitieren.

resume: Wird zum Zustand resume gewechselt, wird das Fahrzeug beschleunigt, solange bis der Zustand gewechselt wird oder die desiredspeed erreicht wird.

follow: Im Zustand follow wird die Geschwindigkeit bis zur Mindestgeschwindigkeit reduziert.

cruise: Der Zustand cruise hält die Geschwindigkeit des Fahrzeugs konstant.

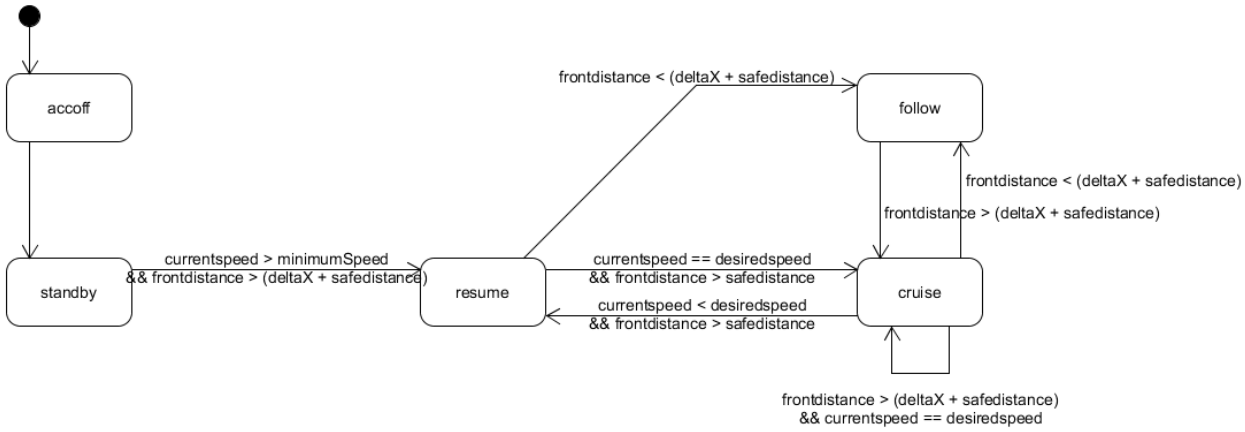


Abbildung 3.5: ACC Zustandsautomat (known frontspeed Algorithmus)

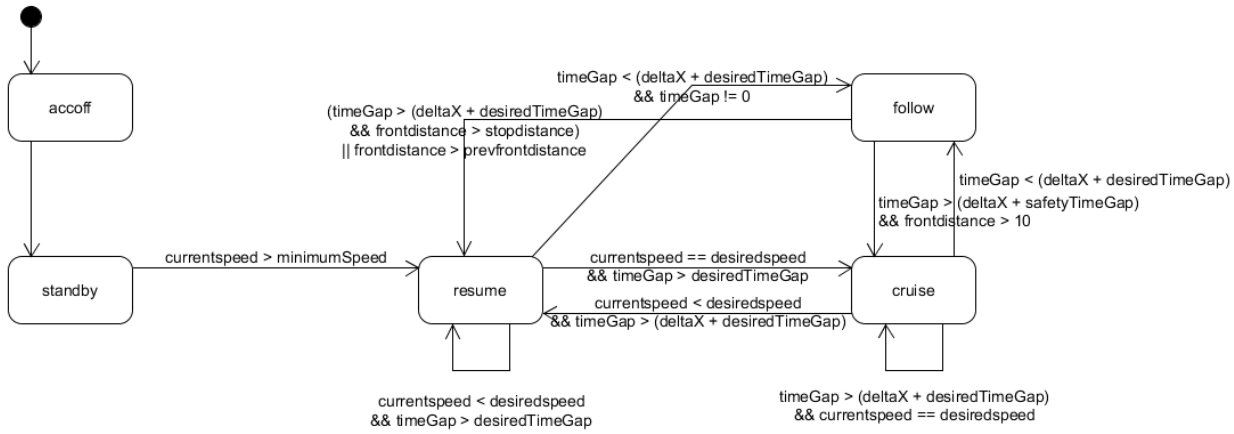


Abbildung 3.6: ACC Zustandsautomat (unknown frontspeed Algorithmus)

3.9.2 ACC Stop & Go endliche Zustandsautomaten

Die nachfolgenden zwei Zustandsautomaten in Abbildung 3.7 und in Abbildung 3.8 zeigen die zwei unterschiedlichen ACC Stop & Go Algorithmen. Einmal mit der Information der Geschwindigkeit des vorausfahrenden Fahrzeugs (known frontspeed Algorithmus) und einmal ohne die Information (unknown frontspeed Algorithmus).

Die folgenden Zustände können angenommen werden:

accoff: In diesem Zustand startet das System und wechselt sofort zum nächsten Zustand.

standby: In diesem Zustand wird das Fahrzeug solange beschleunigt, bis es die Mindestgeschwindigkeit erreicht hat. Es soll die Beschleunigung eines Fahrers imitieren.

resume: Wird zum Zustand resume gewechselt, wird das Fahrzeug beschleunigt, solange bis der Zustand gewechselt wird oder die desiredspeed erreicht wird.

follow: Im Zustand follow wird die Geschwindigkeit reduziert.

cruise: Der Zustand cruise hält die Geschwindigkeit des Fahrzeugs konstant.

stop: Im Zustand stop wird das Fahrzeug sofort zum stehen gebracht.

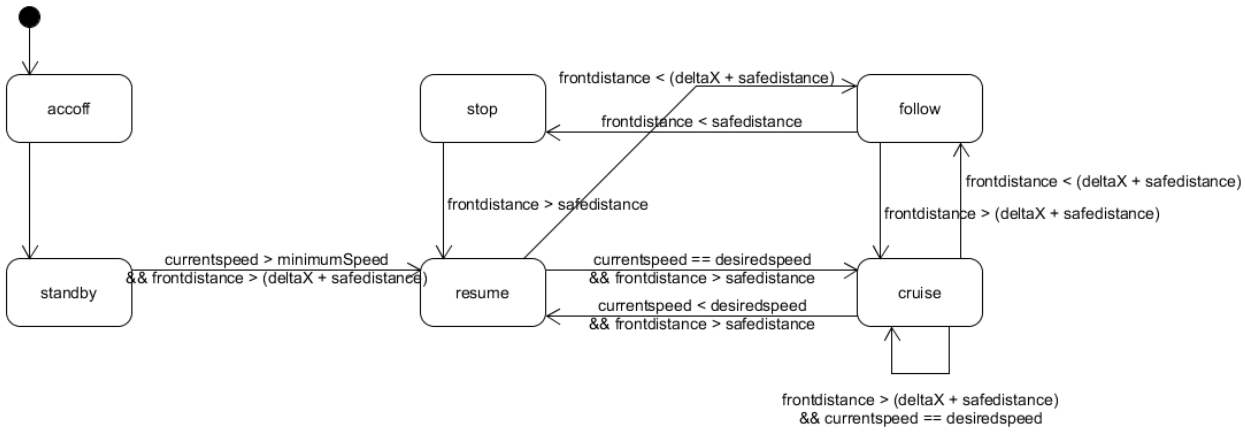


Abbildung 3.7: ACC Stop & Go Zustandsautomat (known frontspeed Algorithmus)

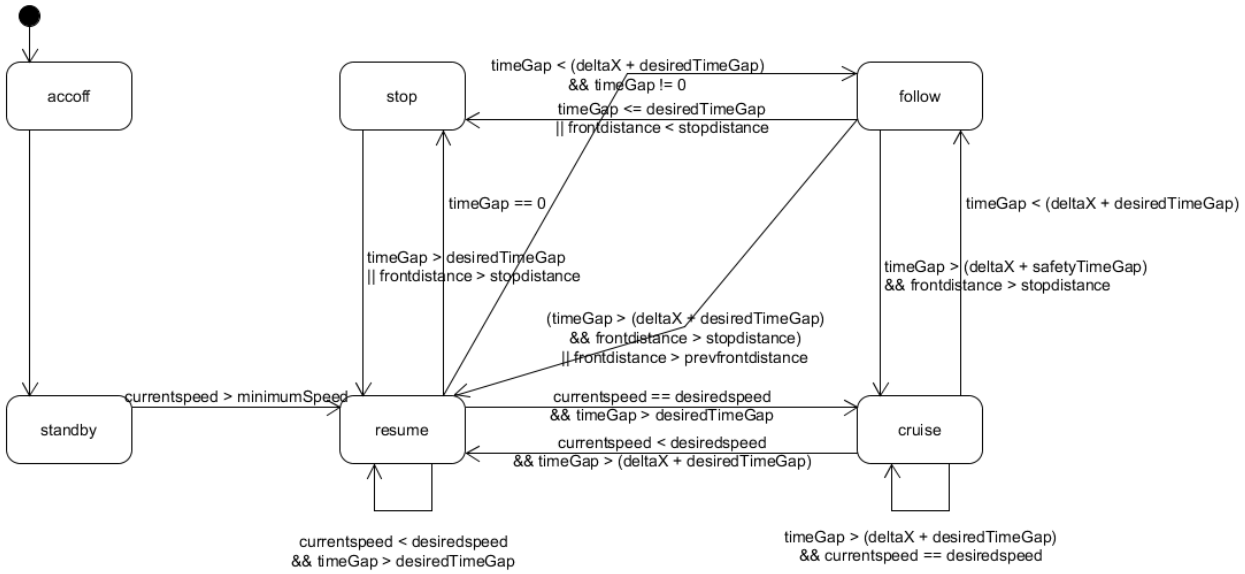


Abbildung 3.8: ACC Stop & Go Zustandsautomat (unknown frontspeed Algorithmus)

3.10 Simulink Model

Das Simulink Model in Abbildung 3.9 veranschaulicht die Prozesse im ACC Simulator. Im Rechteck mit dem Namen ACC STOP and GO Simulator läuft der Algorithmus des ACC Simulators statt.

Die benötigten Daten werden verarbeitet und es wird eine Zielgeschwindigkeit berechnet. Diese Zielgeschwindigkeit wird dem PID Speed Controller übergeben. Der PID Speed Controller berechnet die Geschwindigkeit, die zur currentspeed dazugerechnet werden muss.

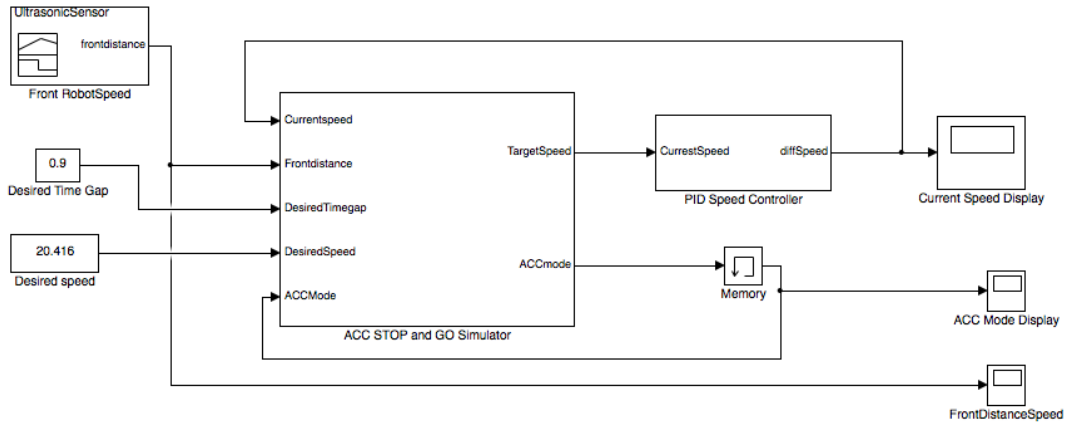


Abbildung 3.9: ACC Simulator Simulink

3.11 Methode zur Linienverfolgung

Der ACC Simulator hat die Option einer schwarzen Linie nachzufahren. Dafür wird der EV3 Farbsensor verwendet. Er ist im ACC Simulator auf Helligkeit eingestellt. Das bedeutet, dass der Sensor die Helligkeit der Fläche, die er anstrahlt, messen kann. Umso heller die Fläche, desto höher der Wert, der vom Sensor zurückgegeben wird. In Abbildung 3.10 sind mögliche Werte abgebildet, die der Sensor zurückgibt, wenn er diese anstrahlt.

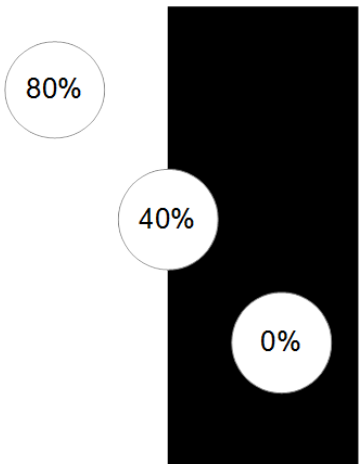


Abbildung 3.10: Farbsensor Messwerte auf einer schwarzen Linie

Die Linienverfolgung im ACC Simulator benutzt folgende Methode:

- Wenn der Farbsensor im ACC Simulator einen Wert zurückgibt, der kleiner ist als 30, biegt das Lego Fahrzeug nach links ab.
- Gibt der Farbsensor einen Wert zurück, der größer ist als 50, biegt das Lego Fahrzeug nach rechts ab.
- Umso größer der zurückgegebene Wert von 40 abweicht, umso stärker wird abgebogen.

4 Algorithmen und Formeln

Dieser Abschnitt geht auf die verschiedenen Algorithmen und Formeln ein, die im ACC Simulator verwendet werden. Der ACC Simulator kann das ACC System und die Erweiterung ACC Stop & Go simulieren. Diese zwei Systeme haben jeweils zwei unterschiedliche Algorithmen im ACC Simulator implementiert mit teilweise unterschiedlichen Formeln.
Der erste Algorithmus arbeitet mit der Information, mit welcher Geschwindigkeit das vorausfahrende Fahrzeug fährt. Hierfür muss vor dem Start die Geschwindigkeit des vorausfahrenden Fahrzeugs eingegeben werden. Dieser Algorithmus orientiert sich an dem ACC Simulator von Asim Abdulkhaleq und Stefan Wagner[1] und wird im folgenden als known frontspeed Algorithmus bezeichnet. Der zweite Algorithmus funktioniert ohne die Information der Geschwindigkeit des vorausfahrenden Fahrzeugs. Dieser Algorithmus wird nachfolgend als unknown frontspeed Algorithmus bezeichnet.

4.1 Szenarien für known frontspeed Algorithmus

In der Abbildung 4.1 sind die Fahrzeuge ACCcar und frontcar zu sehen. Zwischen den beiden Fahrzeugen sind zwei Abstände eingetragen.
Der erste Abstand zwischen Markierung A und Markierung B ist deltaX. Der Abstand deltaX ist der Bremsweg, der nötig ist, um sich an die Geschwindigkeit zum frontcar anzupassen. Der zweite Abstand ist safedistance und befindet sich zwischen Markierung B und Markierung C. Er soll vom ACCcar eingehalten werden. Falls dieser Abstand unterschritten wird, hält das ACCcar sofort an.

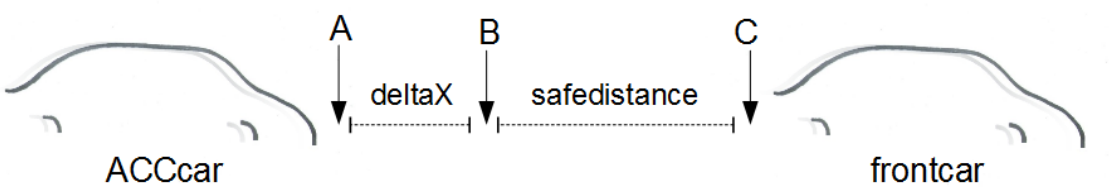


Abbildung 4.1: ACC Simulator known frontspeed Algorithmus Übersicht

Als Erstes wird deltaX berechnet. Die Berechnung im ACC Simulator erfolgt folgendermaßen:

$$deltaX = \frac{(frontspeed * frontspeed) - (currentspeed * currentspeed)}{2 * (-accelerationratio)}$$

[1] urlhttps://github.com/asimabdulkhaleq/STPA-and-Software-Model-Checking

frontspeed:	Die Geschwindigkeit des vorausfahrenden Fahrzeugs.
currentspeed:	Die aktuelle Geschwindigkeit des Fahrzeuges mit dem ACC System.
accelerationratio:	Die Beschleunigung des Fahrzeugs mit dem ACC System.

1. Szenario: frontdistance > deltaX + safedistance

Das ACCcar befindet sich in der Abbildung 4.1 vor der Markierung A. Falls das ACCcar seine gewünschte Geschwindigkeit noch nicht erreicht hat, beschleunigt es. Die Beschleunigung erfolgt folgendermaßen:

$$currentspeed+ = calPID(\sqrt{(currentspeed * currentspeed) + 2 * |(deltaX + safedistance + 1 - frontdistance)|} - currentspeed)$$

calPID:	Ein PID Controller wird aufgerufen.
frontdistance:	Der aktuelle Abstand, der vom Ultraschallsensor gemessen wird.

2. Szenario: frontdistance < deltaX + safedistance && frontdistance > safedistance

Das ACCcar befindet sich in der Abbildung 4.1 zwischen der Markierung A und der Markierung B. In diesem Bereich reduziert das ACCcar seine Geschwindigkeit. Die Geschwindigkeit wird solange reduziert, bis der Abstand einen Zentimeter über dem Sicherheitsabstand beträgt. Dieser Zentimeter soll als Puffer für die ungenaue Abstandsmessung des EV3 Ultraschallsensors dienen. Die Formel für die Geschwindigkeit ist folgende:

$$currentspeed- = calPID(\sqrt{(currentspeed * currentspeed) + 2 * |(deltaX + safedistance + 1 - frontdistance)|} - currentspeed)$$

3. Szenario: frontdistance < safedistance
Das ACCcar befindet sich in der Abbildung 4.1 zwischen der Markierung B und der Markierung C. In diesem Bereich kommt das ACCcar sofort zum Stillstand und zwar folgendermaßen:

$$currentspeed = 0$$

4.2 Szenarien für unknown frontspeed Algorithmus

In der Abbildung 4.2 sind die Fahrzeuge ACCcar und frontcar zu sehen. Zwischen den beiden Fahrzeugen sind drei Abstände eingetragen.
Der erste Abstand zwischen Markierung A und Markierung B ist, wie beim known frontspeed Algorithmus, deltaX. Der Abstand deltaX ist der Bremsweg, der nötig ist, um sich an die Geschwindigkeit zum

frontcar anzupassen. Der zweite Abstand ist desiredTimeGap und befindet sich zwischen Markierung B und Markierung C. Falls dieser Abstand unterschritten wird, hält das ACCcar sofort an. Im Gegensatz zum safedistance Abstand aus dem known frontspeed Algorithmus, ist dieser Abstand variabel. Der Bereich zwischen Markierung B und Markierung C wird immer kleiner, umso langsamer das ACCcar fährt. Aus diesem Grund gibt es einen dritten Abstand stopdistance. Dieser Abstand ist fest und ist deshalb nötig, damit der Ultraschallsensor korrekt arbeitet.

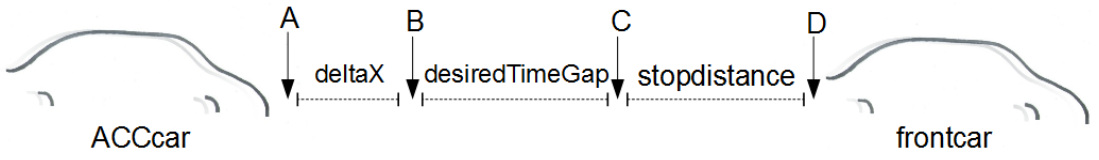


Abbildung 4.2: ACC Simulator unknown frontspeed Algorithmus Übersicht

Als Erstes wird deltaX berechnet. Die Berechnung im ACC Simulator erfolgt folgendermaßen:

$$deltaX = 0.5 + \sqrt{timeGap}$$

timeGap: Die Zeit die benötigt wird die Strecke zwischen dem ACCcar und dem frontcar abzufahren.

1. Szenario: timeGap > deltaX + desiredTimeGap

Das ACCcar befindet sich in der Abbildung 4.2 vor der Markierung A. Falls das ACCcar seine gewünschte Geschwindigkeit noch nicht erreicht hat, beschleunigt es. Die Beschleunigung erfolgt folgendermaßen:

$$currentspeed += 1$$

2. Szenario: timeGap < deltaX + desiredTimeGap && timeGap > desiredTimeGap

Das ACCcar befindet sich in der Abbildung 4.2 zwischen der Markierung A und der Markierung B. In diesem Bereich reduziert das ACCcar seine Geschwindigkeit und zwar folgendermaßen:

$$currentspeed -= calPID(\sqrt{(currentspeed * currentspeed) + 2 * |(deltaX + desiredTimeGap - timeGap)|} - currentspeed)$$

desiredTimeGap: Eine timeGap die einen gewissen Abstand zum frontcar gewährt.

3. Szenario: timeGap < desiredTimeGap || frontdistance < stopdistance

Das ACCcar befindet sich in der Abbildung 4.2 zwischen der Markierung B und der Markierung D. In diesem Bereich kommt das ACCcar sofort zum Stillstand ind zwar folgendermaßen:

$currentspeed = 0$

stopdistance:	Ein fester Sicherheitsabstand der nicht unterboten werden darf, um keine falschen Werte des Ultraschallsensors zu erhalten.

5 Evaluation

In diesem Kapitel werden die Daten der beiden Algorithmen im ACCcar verglichen und visualisiert. Dafür wurde folgendes Testszenario für beide Algorithmen verwendet:
Das ACCcar und das frontcar fuhren eine gerade Strecke entlang. Das frontcar fuhr mit 10 cm/s und wurde mit einem Vorsprung gestartet. Das ACCcar wurde mit einer Zielgeschwindigkeit von 15 cm/s eingestellt und ist nach dem frontcar gestartet. Das Testszenario lief 20 Sekunden. Die Daten currentspeed, frontdistance und accmode wurden jeweils alle 0,1 Sekunden gespeichert und in ein Diagramm bzw. Tabelle übertragen.

5.1 Daten eines Testszenarios mit known frontspeed Algorithmus

In der Abbildung 5.1 ist zu erkennen, dass der Ultraschallsensor die Abstände ungenau erfasst. Die Abstände haben über dem Zeitraum des Testszenarios kleine Ausschläge. Trotzdem passt sich die Geschwindigkeit relativ flexibel an, wie in Abbildung 5.2 zu sehen ist.
Zwischen dem Zeitabschnitt 2 und 6 beschleunigt das ACCcar bis zur desiredspeed, die 15 cm/s beträgt. Von dem Zeitpunkt 7 bis 25 befindet sich das ACCcar im cruise Zustand. Deshalb hält das ACCcar die Geschwindigkeit konstant. Zwischen dem Zeitabschnitt 26 und 91 wechselt der Zustand zwischen follow und cruise. Dies hat zur Folge, dass die Geschwindigkeit des ACCcars reduziert wird, bis es sich der Geschwindigkeit des frontcars angepasst hat. Ab dem Zeitpunkt 92 wechselt der Zustand ständig zwischen cruise, follow und resume. Das ACCcar stellt damit sicher das es den Sicherheitsabstand einhält und nicht zurückfällt.
Der Sicherheitsabstand beträgt 20 cm, aber der Algorithmus lässt 1 cm als Puffer. Der Puffer ist nötig, da der EV3 Ultraschallsensor zu ungenau ist. Deshalb versucht das ACCcar dem frontcar mit dem Abstand von 21 cm zu folgen.

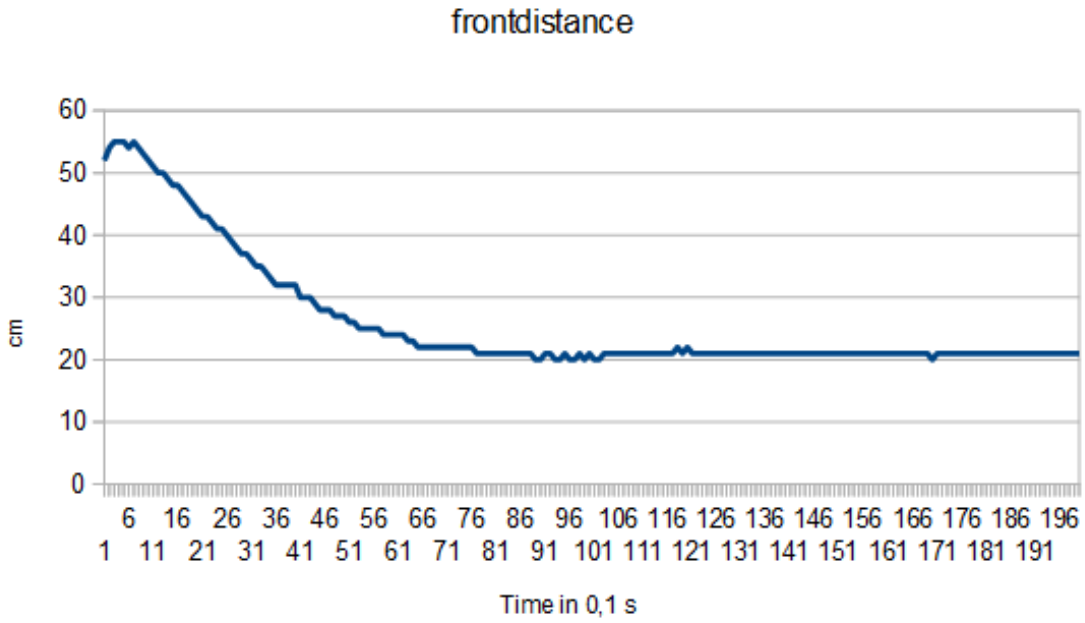


Abbildung 5.1: frontdistance Daten des known frontspeed Algorithmus

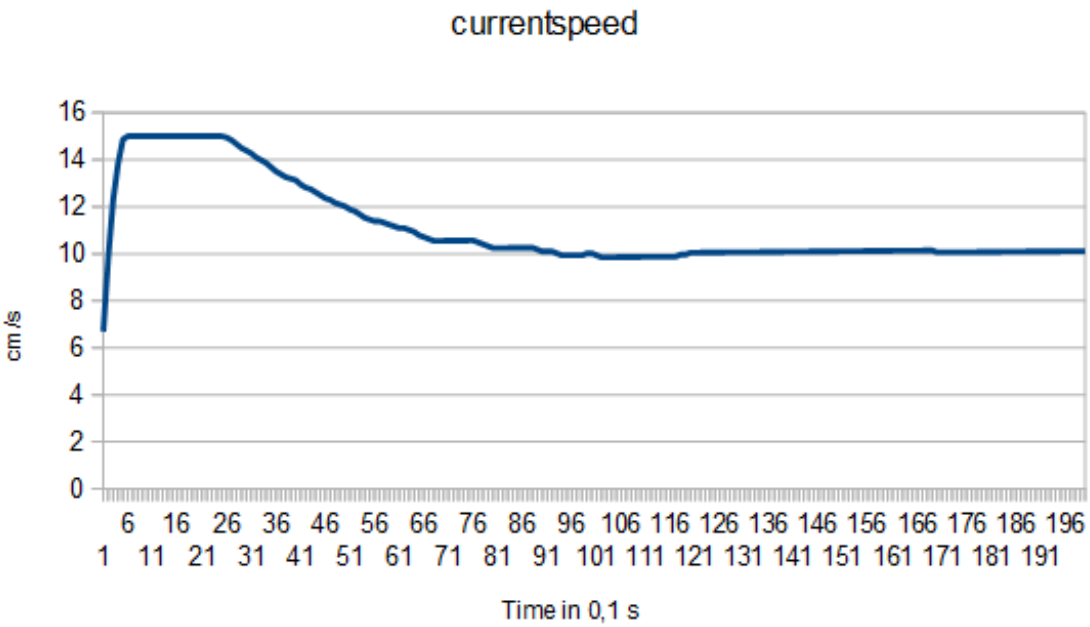


Abbildung 5.2: currentspeed Daten des known frontspeed Algorithmus

Time in 0,1 s	**accmode**
1	standby
2-6	resume
7-25	cruise
26-56	follow
57	cruise
58-61	follow
62	cruise
63-68	follow
69	cruise
70-76	resume
81	cruise
82-88	resume
89-90	follow
91	cruise
92	resume
93-94	follow
95-97	cruise
98-100	resume
101-102	follow
103	cruise
104-169	resume
170	follow
171	cruise
172-200	resume

Tabelle 5.1: accmode Daten des known frontspeed Algorithmus

5.2 Daten eines Testszenarios mit unknown frontspeed Algorithmus

Wie beim Testszenario mit dem known frontspeed Algorithmus, ist in der Abbildung 5.3 zu erkennen, dass die Abstandsmessung in Zentimeter zu grob ist. Dies hat zur Folge, dass die Geschwindigkeit des ACCcars sich abrupter ändert. Der Graph aus Abbildung 5.4 schlägt an den selben Stellen aus wie der Graph aus Abbildung 5.3.
Zwischen dem Zeitpunkt 2 und 14 beschleunigt das ACCcar, bis es im Zeitpunkt 15 in den cruise Zustand übergeht. Ab dem Zeitpunkt 16 wird in den follow Zustand übergegangen. Die Geschwindigkeit wird dann bis zum Zeitpunkt 80 reduziert. Zwischen dem Zeitabschnitt 81 und 200 wechselt der Zustand zwischen cruise, follow und resume, um sich der Geschwindigkeit des frontcars anzupassen.
Die Geschwindigkeit des ACCcars schwankt ab dem Zeitpunkt 81 zwischen 9 cm/s und 11 cm/s, wie in Abbildung 5.4 zu sehen ist. Der Abstand, in Abbildung 5.3, schwankt ebenfalls und zwar zwischen 17 cm und 19 cm.
Der Sicherheitsabstand ist im Gegensatz zum known frontspeed Algorithmus nicht fest und wird aus deltaX und desiredTimeGap berechnet.

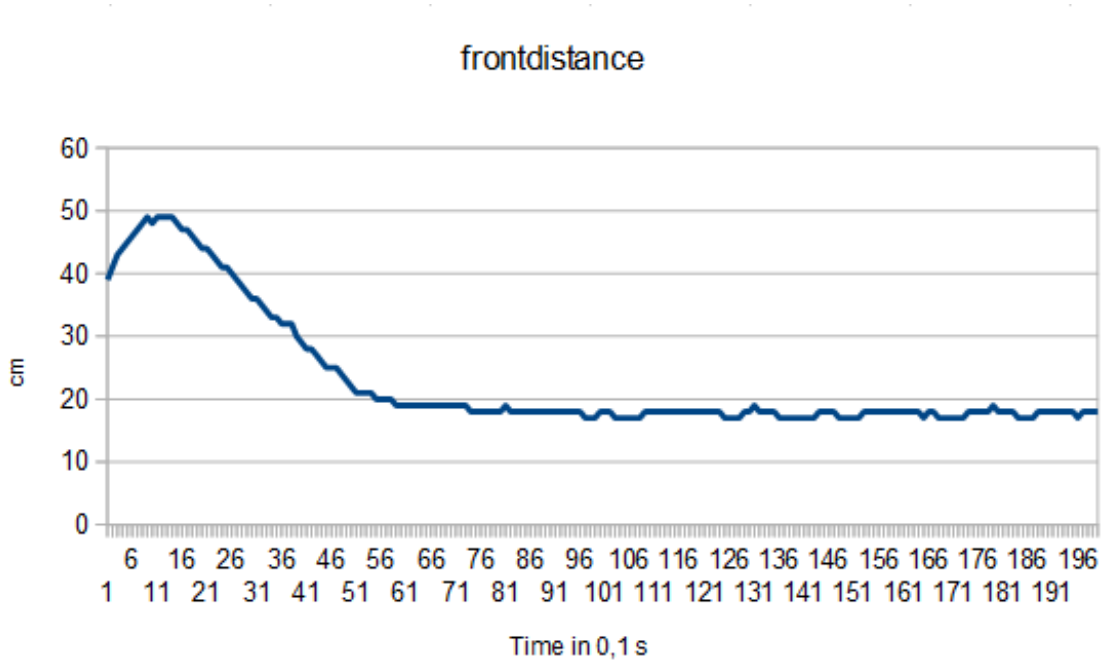


Abbildung 5.3: frontdistance Daten des unknown frontspeed Algorithmus

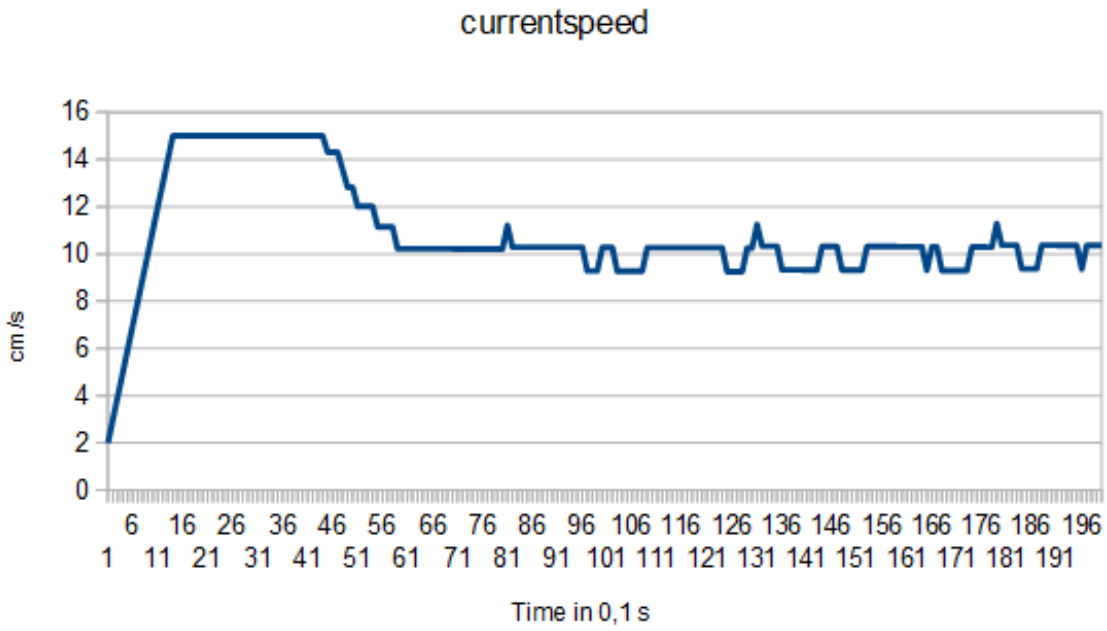


Abbildung 5.4: currentspeed Daten des unknown frontspeed Algorithmus

Time in 0,1 s	accmode
1	standby
2-14	resume
15	cruise
16-80	follow
81	resume
82-99	follow
100	resume
101-108	follow
109	resume
110-128	follow
129	resume
130	follow
131	resume
132-143	follow
144	resume
145-152	follow
153	resume
154-165	follow
166	resume
167-173	follow
174	resume
175-178	follow
179	resume
180-196	follow
197	resume
198-200	follow

Tabelle 5.2: accmode Daten des unknown frontspeed Algorithmus

6 Implementierung

Dieses Kapitel soll einen Überblick über die Entwicklung des ACC Simulators bieten. Es wird auf die verwendete Entwicklungsumgebung und auf den Quellcode eingegangen.

6.1 Entwicklungsumgebung

Der ACC Simulator wurde auf der Entwicklungsumgebung EV3 IDE[1] Version 1.1.0.0 entwickelt und kann für das Betriebssystem Windows 7 kostenlos heruntergeladen werden. Der Download ist auf der Homepage Seite verlinkt und befindet sich auf folgender Internetseite:
https://onedrive.live.com/?id=5B144ED4386B5E6A!773&cid=5B144ED4386B5E6A&group=0&authkey=!AIiP-RE_hsNu9WM

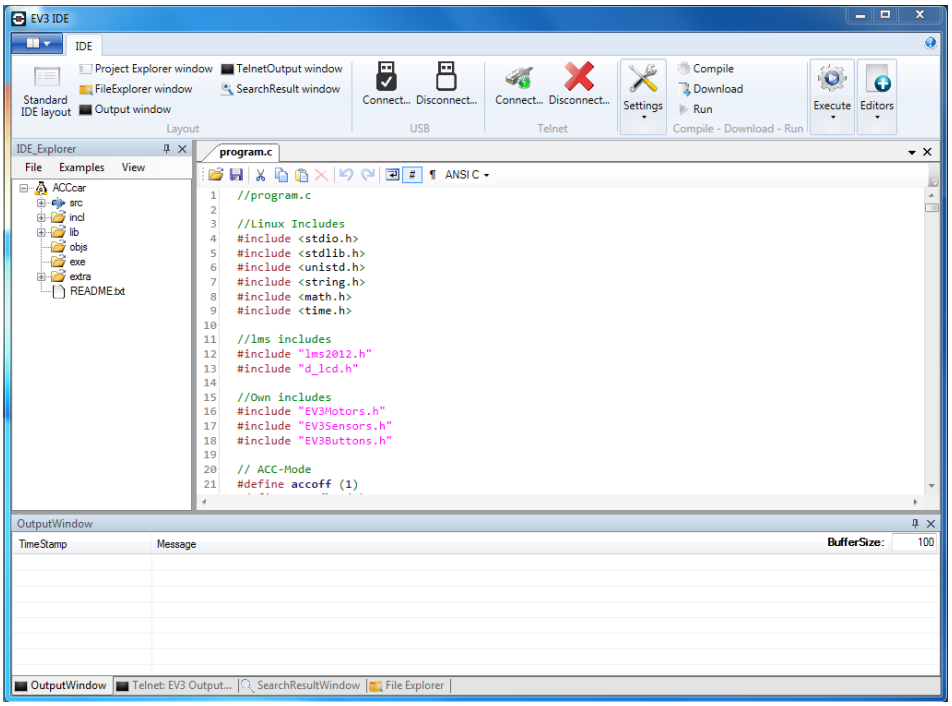


Abbildung 6.1: EV3 IDE Screenshot

[1] http://jasvap.somee.com/

Die Entwicklungsumgebung ermöglicht es Lego Mindstorms EV3 Roboter mit ANSI C zu programmieren. Es ist in der Lage den geschriebenen Code zu kompilieren, zu downloaden und zu starten.

6.2 Quellcode

Der Quellcode des ACC Simulators ist open source und wird auf der folgenden sourceforge Internetseite bereit gestellt:
https://sourceforge.net/projects/acc-with-stop-and-go-simulator/files/

Die Programme für das ACCcar und das frontcar sind beide in ANSI C geschrieben und haben eine ähnliche Programmstruktur.
In der c-Datei program.c findet die Berechnung der Simulation statt. In den anderen c-Dateien, EV3Motors.c, EV3Sensors.c und EV3Buttons.c, wird der Zugriff auf die Komponenten wie den Motoren, den Knöpfen und den Sensoren ermöglicht. Die c-Dateien EV3Motors.c, EV3Sensors.c und EV3Buttons.c basieren teilweise auf den Beispiel Programmen, die mit der EV3 IDE heruntergeladen werden.

7 Installation/User Guide

In diesem Kapitel wird beschrieben was man braucht, um den ACC Simulator auf den Lego Mindstorms Robotern zu starten und wie man dafür vorgehen muss. Es wird ebenfalls beschrieben wie man den ACC Simulator bedienen kann.

7.1 Installation

Um die ACC Simulator Software auf den EV3 Brick zu kopieren, gibt es zwei Möglichkeiten. Die erste Möglichkeit ist sehr einfach und ist für Personen gedacht, die nur die Simulation ausführen wollen. Die zweite Möglichkeit, ist für Personen, die Veränderungen im Quellcode vornehmen wollen.

7.1.1 Möglichkeit 1

Folgen Sie den Anweisungen, falls Sie nur die Simulation ausführen wollen.

ACCcar:

1. Laden Sie den Ordner „ForACCcar.zip“ herunter: https://sourceforge.net/projects/acc-with-stop-and-go-simulator/files/SDCardCopy/
2. Entpacken Sie den Ordner „ForACCcar.zip“
3. Stecken Sie die Mikro SD Karte in ihren PC
4. Kopieren Sie den Inhalt des Ordners „Forfrontcar“ auf die Mikro SD Karte
5. Stecken Sie die Mikro SD Karte in ihren ACCcar EV3 Brick

frontcar:

1. Laden Sie den Ordner „Forfrontcar.zip“ herunter: https://sourceforge.net/projects/acc-with-stop-and-go-simulator/files/SDCardCopy/
2. Entpacken Sie den Ordner „Forfrontcar.zip“
3. Stecken Sie die Mikro SD Karte in ihren PC
4. Kopieren Sie den Inhalt des Ordners „Forfrontcar“ auf die Mikro SD Karte
5. Stecken Sie die Mikro SD Karte in ihren frontcar EV3 Brick

7.1.2 Möglichkeit 2

Folgen Sie den Anweisungen, falls Sie mit dem Quellcode arbeiten wollen.

Benötigte Programme und Dateien:

1. Laden Sie die EV3 IDE Version 1.1.0.0 herunter: https://onedrive.live.com/?id=5B144ED4386B5E6A!773&cid=5B144ED4386B5E6A&group=0&authkey=!AIiP-RE_hsNu9WM
2. Laden Sie den Ordner „accsimulator.zip“ herunter: https://sourceforge.net/projects/acc-with-stop-and-go-simulator/files/?source=navbar
3. Entpacken Sie den Ordner „accsimulator.zip“

Die Anleitungen, „Mikro SD Karte vorbereiten“ und „Installation des Compilers“, sind beim EV3 IDE Download auf englischer Sprache mit dabei.

Mikro SD Karte vorbereiten:
Als nächstes muss die Mikro SD Karte vorbereitet werden, auf dem die ACC Simulator Software heruntergeladen wird.

1. Stecken Sie die Mikro SD Karte in ihren PC
2. Erstellen Sie auf der SD Karte zwei Ordner namens „lib“ und „myapps“
3. Platzieren Sie die Datei „libstdc++.so.6.0.10“ in den „lib“ Ordner
4. Platzieren Sie die Datei „run_ progran.rbf“ in den „myapps“ Ordner
5. Stecken Sie die Mikro SD Karte in ihren EV3 Brick

Hinweis: Wählen Sie „File“ -> „New project“ -> „Basic project...“ im IDE Explorer Fenster aus und alle Dateien die Sie auf die Mikro SD Karte platzieren müssen, befinden sich im Ordner „EV3_ Files“ .

Installation des Compilers:

1. Laden Sie CodeSourcery Lite herunter: https://sourcery.mentor.com/GNUToolchain/package4574/public/arm-none-linux-gnueabi/arm-2009q1-203-arm-none-linux-gnueabi.exe
2. Installieren Sie „arm-2009q1-203-arm-none-linux-gnueabi.exe“ CodeSourcery Lite crosscompiler für arm-linux in einen Ordner, mit einem kurzen Pfad ohne Leerzeichen, wie zum Beispiel: „C:\CSLite“
3. Bearbeiten Sie die PC PATH Umgebungsvariable und fügen Sie den Pfad für den CodeSourcery Lite’s bin Ordner an den Anfang des Pfades, zum Beispiel: „C:\CSLite\bin;“ %andere Pfadvariablen%

4. Installieren Sie die „linux_ tools.zip“ Datei, enthalten in „EV3IDE.zip“ -> Extrahiere „linux_ tools.zip“ nach „C:\tools“
5. Aktualisieren Sie die PC PATH Umgebungsvariable in dem Sie „C:\tools;“ nach „C:\CSLite\bin;“ hinzufügen
6. Sie müssen eventuell den PC neu starten, damit die PATH Variable aktiv wird

Programm auf den EV3 Brick herunterladen:

1. Starten Sie die EV3 IDE
2. Wählen Sie „File“ -> „Open project“ im IDE Explorer Fenster aus und öffnen Sie das ACCcar bzw. frontcar Projekt im „accsimulator“ Ordner
3. Schalten Sie den EV3 Brick ein und verbinden Sie es mit ihren PC über ein USB Kabel
4. Klicken auf der Menüleiste der EV3 IDE auf „Connect...“
5. Klicken auf der Menüleiste der EV3 IDE auf „Compile“ und danach auf „Downlaod“

7.2 User Guide

Dieser Abschnitt soll in die Bedienung des EV3 Bricks und des ACC Simulators einführen. Die Anleitung ist identisch für das ACCcar und das frontcar Programm. Die Bedienung erfolgt über die EV3 Brick Tasten, die in Abbildung 7.1 zu sehen sind. Die Tasten heißen nachfolgend:

- **1:** Abbruchtaste
- **2:** Starttaste
- **3:** Richtungstaste

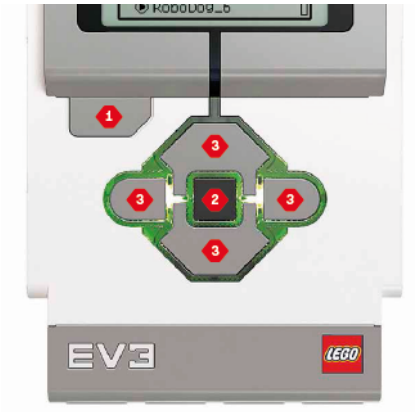


Abbildung 7.1: EV3 Brick Tasten

Um den EV3 Brick einzuschalten halten Sie die Starttaste so lange gedrückt bis die Tasten rot aufleuchten. Leuchten die Tasten nicht auf, versichern Sie sich das der Akku des EV3 Bricks aufgeladen ist. Nach ein paar Sekunden werden die Tasten grün beleuchtet und der EV3 Brick ist eingeschaltet.
Der Bildschirm zeigt nun das Lego Menü an, ähnlich wie in Abbildung 7.2. Der Tab „Zuletzt verwendetes Programm ausführen“ ist ausgewählt und die Dateien, die man zuletzt ausgeführt hat sind zu sehen.

Abbildung 7.2: EV3 Brick Menü

Drücken Sie einmal die reche Richtungstaste, um zur „Datei-Navigation“ zu gelangen. Der Tab zeigt sämtliche Dateien auf dem EV3 Brick und der Mikro SD Karte an. Der Bildschirm des EV3 Bricks sollte so ähnlich wie in Abbildung 7.3 aussehen.

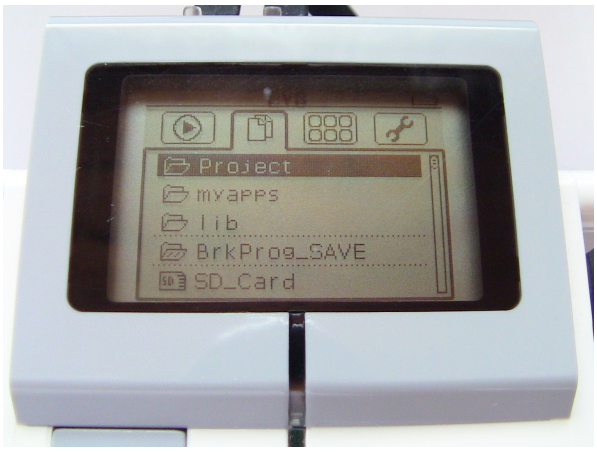


Abbildung 7.3: EV3 Brick Menü

Als nächstes navigieren Sie mit den oberen und unteren Richtungstasten auf den Ordner „SD_ Card“.

Abbildung 7.4: EV3 Brick Menü

Drücken Sie auf die Starttaste, damit öffnen Sie den Ordner.

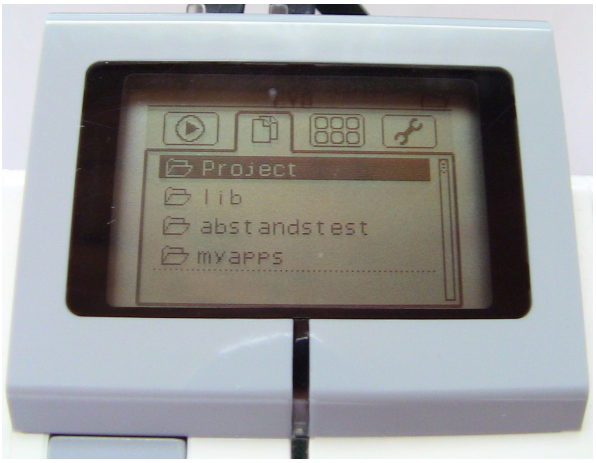


Abbildung 7.5: EV3 Brick Menü

Navigieren Sie nun mit den oberen und unteren Richtungstasten zum Ordner „myapps“.

Abbildung 7.6: EV3 Brick Menü

Drücken Sie auf die Starttaste und Sie sehen nun den gleichen Bildschirm wie in Abbildung 7.7.

Abbildung 7.7: EV3 Brick Menü

Betätigen Sie einmal die untere Richtungstaste, damit Sie „run_ program“ ausgewählt haben.

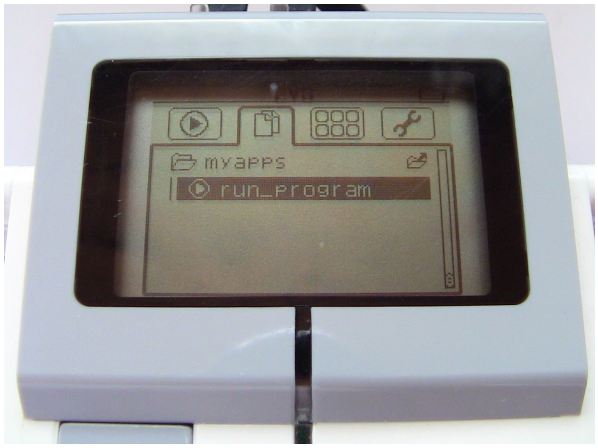


Abbildung 7.8: EV3 Brick Menü

Drücken Sie erneut die Starttaste und der ACC Simulator wird gestartet. Haben Sie das Programm für das ACCcar gestartet, dann wird Ihnen der Bildschirm, wie in Abbildung 7.9, angezeigt.

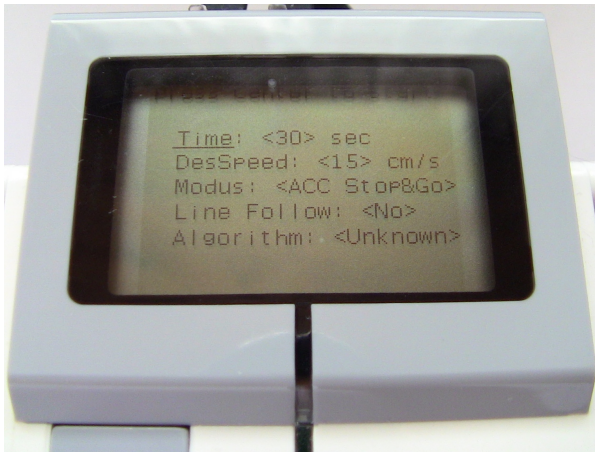


Abbildung 7.9: ACC Simulator Menü

In dem ACC Simulator Menü wählen Sie mit den oberen und unteren Richtungstasten die Einstellungen aus, die Sie verändern wollen. Mit den rechten und linken Richtungstasten ändern Sie den Wert für eine Einstellung. Wenn Sie die Simulation starten wollen, drücken Sie die Starttaste. Sie können jederzeit die Simulation abbrechen, indem Sie die Abbruchtaste drücken. Die Abbruchtaste beendet das Programm und Sie gelangen wieder in das EV3 Brick Menü.
Von hier aus genügt es die Starttaste zu drücken, um das ACC Simulator Programm erneut zu starten. Das Programm befände sich dann beim Tab „Zuletzt verwendetes Programm ausführen“ ganz oben.

8 Zusammenfassung

Im Rahmen dieser Bachelorarbeit wurde ein ACC Simulator für Lego Mindstorms Roboter entwickelt. Der ACC Simulator ist in der Lage ein Adaptive Cruise Control System mit und ohne Stop & Go Funktion zu simulieren. Dieser Simulator besteht aus zwei Programmen. Ein Programm ist für den Lego Mindstorms Roboter mit dem ACC System und das andere Programm ist für den vorausfahrenden Lego Mindstorms Roboter. Das ACC System reguliert die Geschwindigkeit und sorgt dafür, dass ein gewisser Sicherheitsabstand zum vorausfahrenden Roboter eingehalten wird. Der ACC Simulator bietet dafür zwei unterschiedliche Algorithmen.
Der erste Algorithmus heißt known frontspeed Algorithmus und benötigt die Geschwindigkeit des vorausfahrenden Roboters. Diese Information muss vor dem Start der Simulation eingegeben werden. Der zweite Algorithmus heißt unknown frontspeed Algorithmus und benötigt diese Information nicht. Beide Algorithmen passen die Geschwindigkeit unterschiedlich an. Der known frontspeed Algorithmus reduziert und beschleunigt die Geschwindigkeit feiner als der unknown frontspeed Algorithmus.
Das liegt vor allem daran, dass der EV3 Ultraschallsensor den Abstand misst. Der Ultraschallsensor gibt den Abstand in Zentimetern an und diese Abstandsmessung ist zu grob, um eine feinere Geschwindigkeitsanpassung für den unknown frontspeed Algorithmus zu ermöglichen. Zusätzlich hat der Ultraschallsensor eine Genauigkeit von +/- 1 cm. Das bedeutet das der gemessene Abstand von dem echten Abstand abweichen kann und somit nicht optimal für ein sicherheitskritisches System ist.

Es gibt weitere Erweiterungen des ACC Systems, wie zum Beispiel die Erweiterung Cooperative Adaptive Cruise Control [NVP+09]. Dieses System kommuniziert zwischen Fahrzeugen. Dadurch können mehrere Fahrzeuge auf einer Spur die Informationen ihrer Geschwindigkeiten austauschen und ihre Geschwindigkeiten anpassen. Mit diesem System können Staus reduziert werden. Dies könnte eine Anregung für die Weiterentwicklung des ACC Simulators sein.

A Bauanleitung

Die Bauanleitung ist in drei Teile unterteilt. Der erste Teil, die Bauanleitung für das Fahrzeug, erklärt den Aufbau des Fahrzeugs ohne Sensoren. Die zweite und dritte Bauanleitung erklärt wie man die zwei Sensoren, Farbsensor und Ultraschallsensor, an das Fahrzeug befestigt.

Um das ACCcar zu bauen, folgen Sie den Anleitungen:

- Bauanleitung: Fahrzeug
- Bauanleitung: Farbsensor
- Bauanleitung: Ultraschallsensor

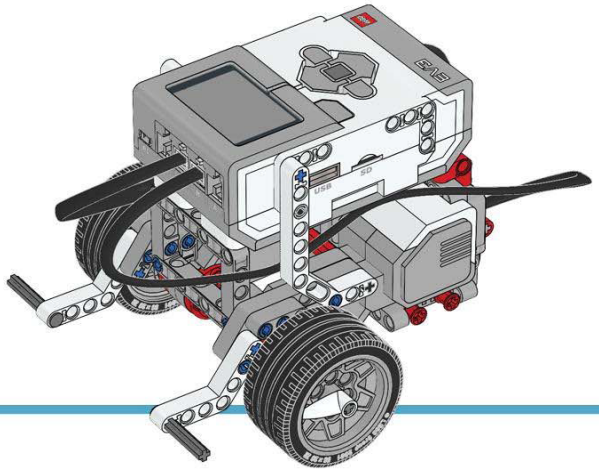
Um das frontcar zu bauen, folgen Sie den Anleitungen:

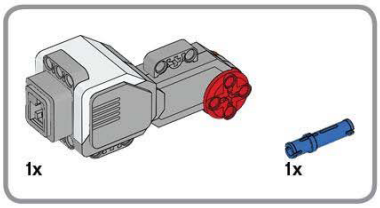
- Bauanleitung: Fahrzeug
- Bauanleitung: Farbsensor

Die nachfolgenden Bauanleitungen stammen aus dem Lego Mindstorms education EV3 Set und können ebenfalls im Internet[1] heruntergeladen werden.

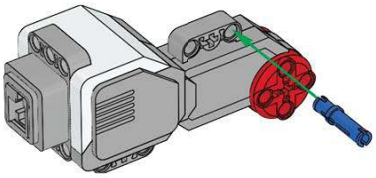
[1] http://robotsquare.com/2013/10/01/education-ev3-45544-instruction/

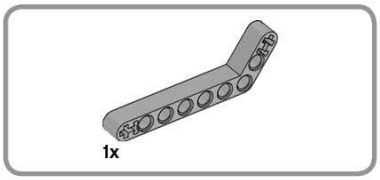
Bauanleitung: Fahrzeug



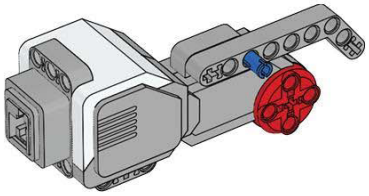


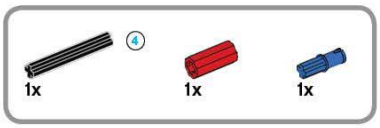
1



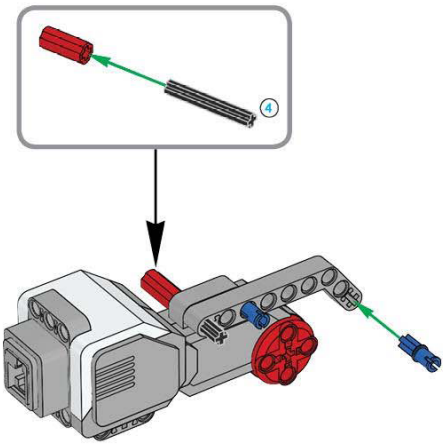


2

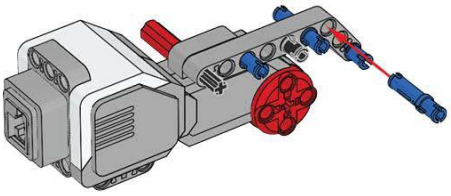


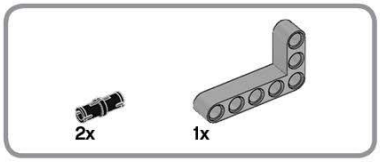


3

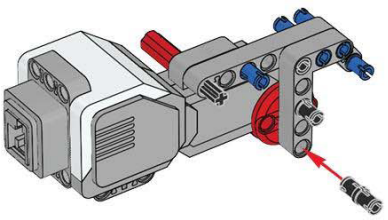


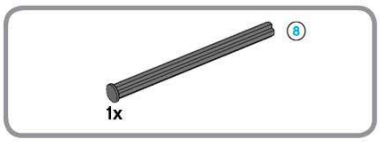
4



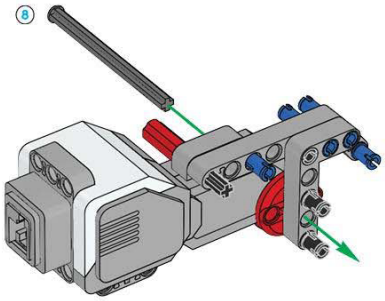


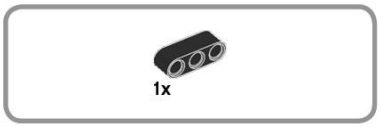
5



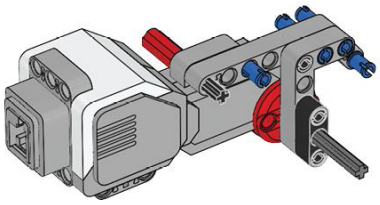


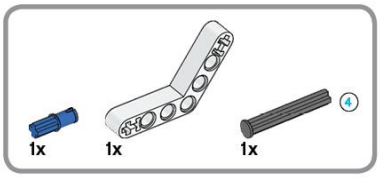
6



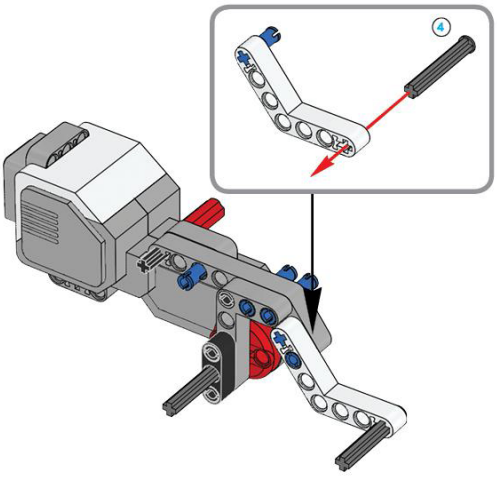


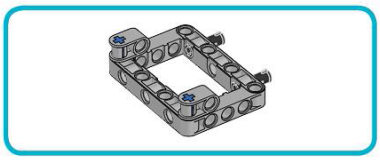
7

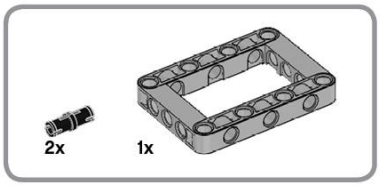




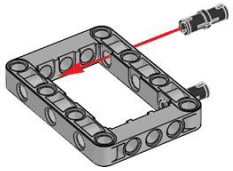
8

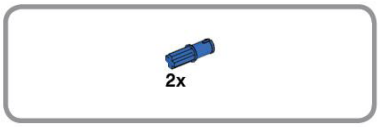




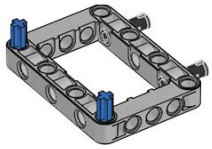


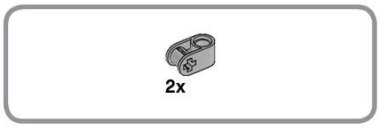
9



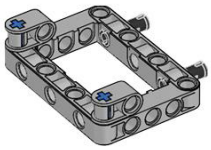


10

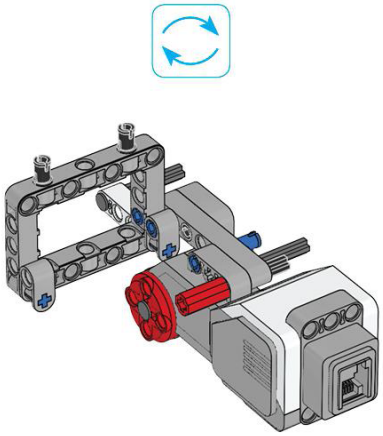


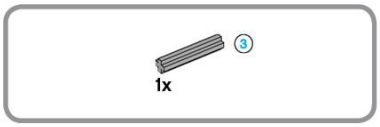


11

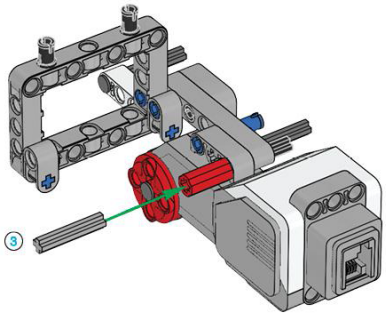


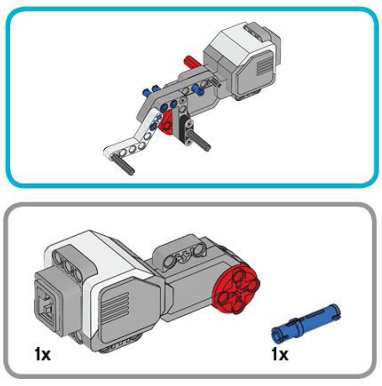
12

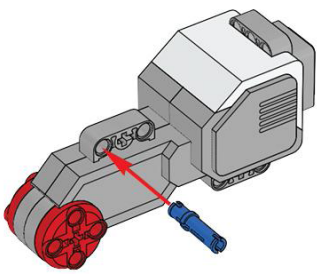




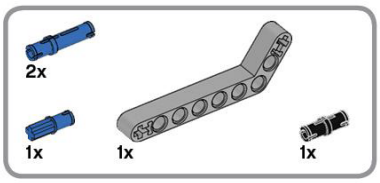
13



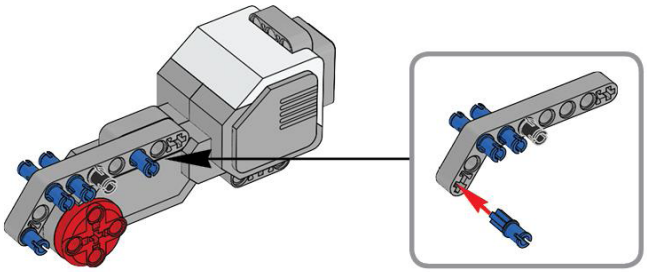


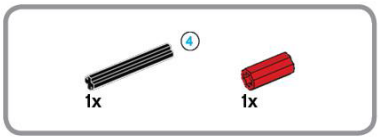


14

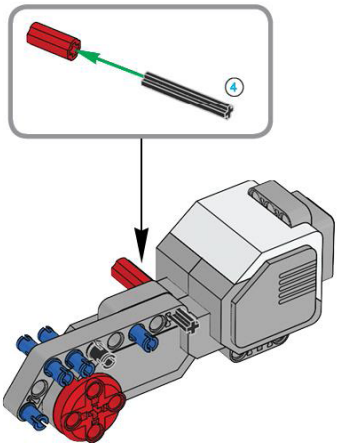


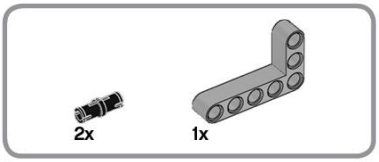
15



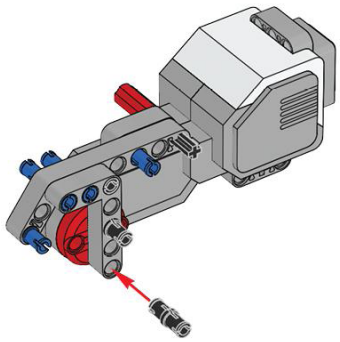


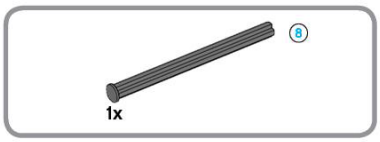
16



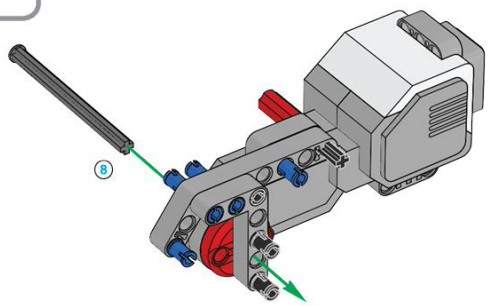


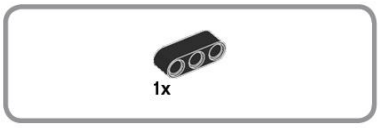
17



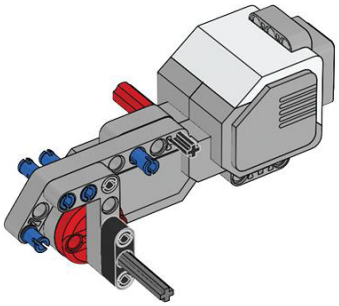


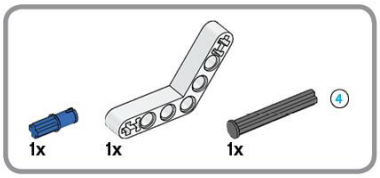
18



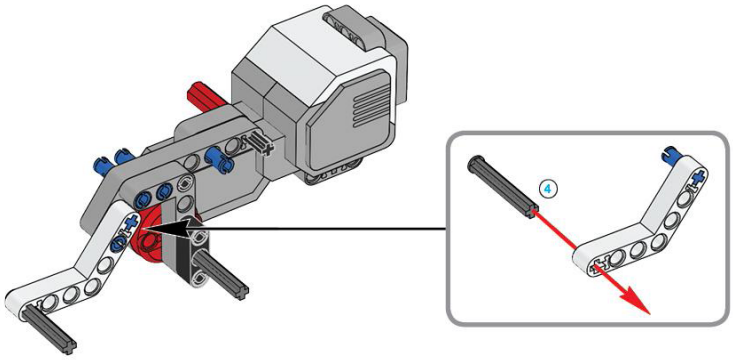


19

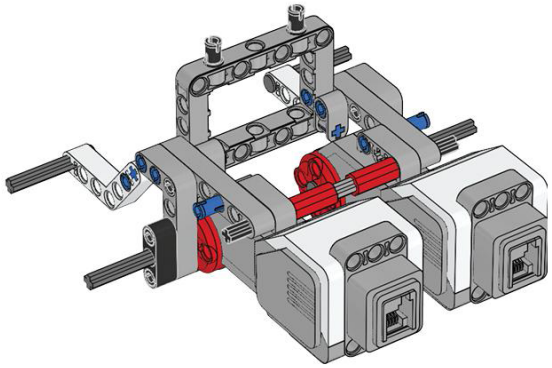




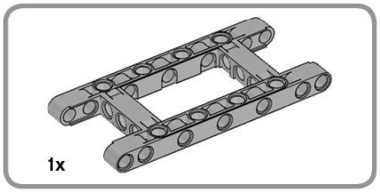
20



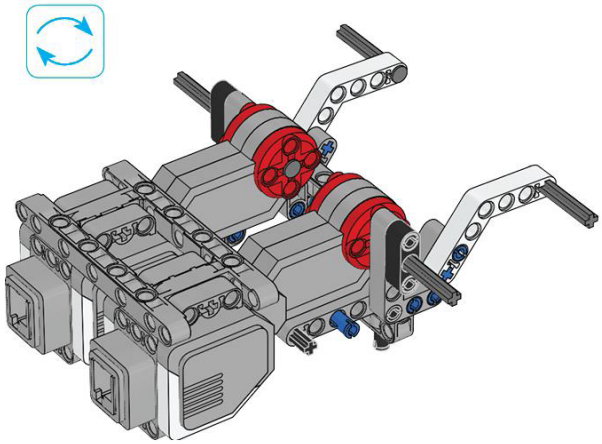
21



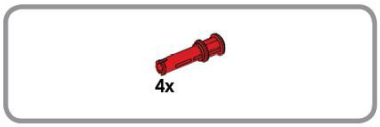
1x



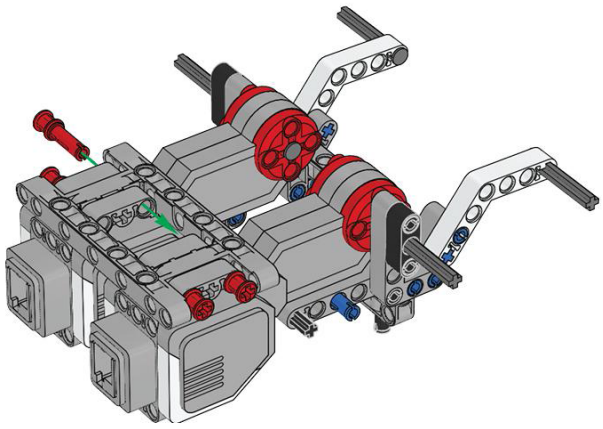
22

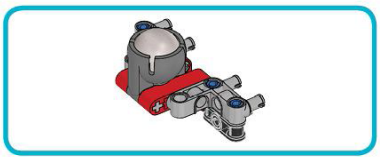


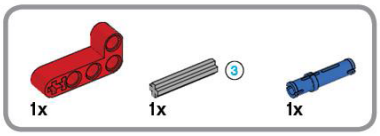
4x



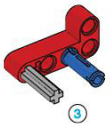
23

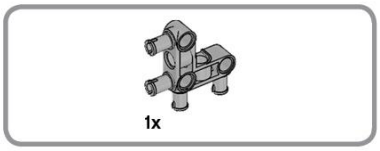




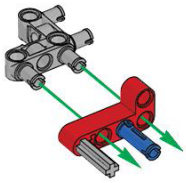


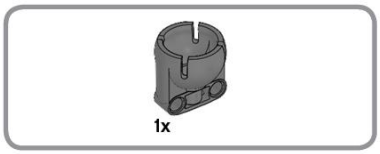
24



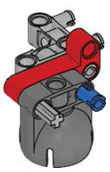


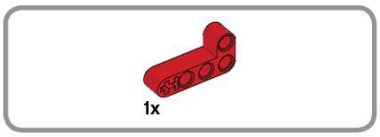
25



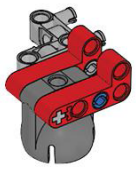


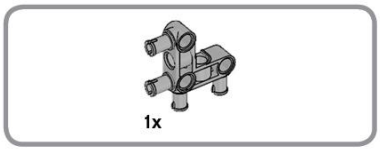
26



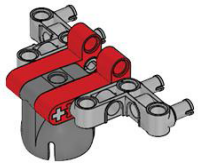


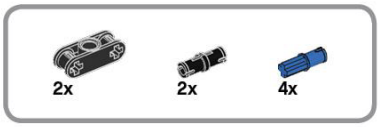
27



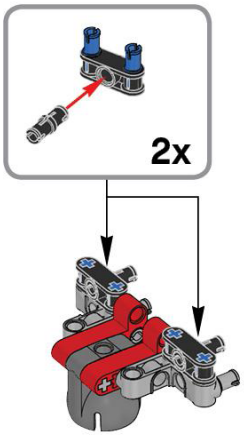


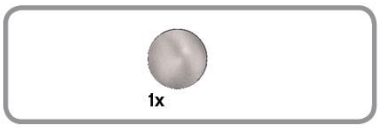
28



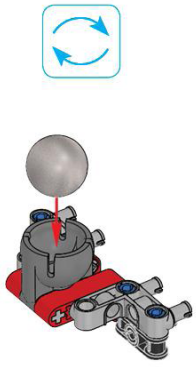


29

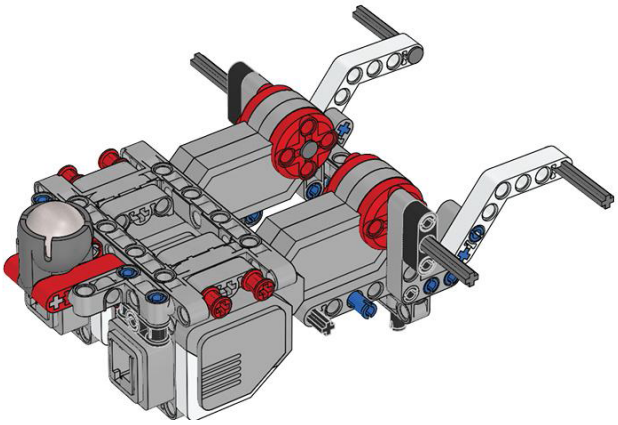


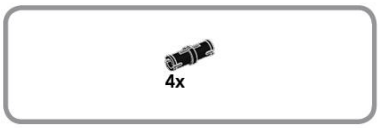


30

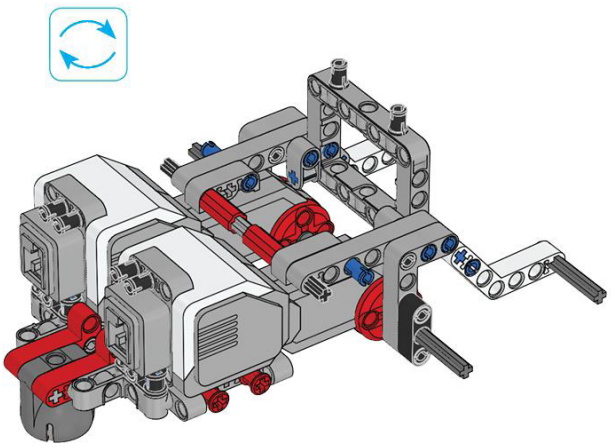


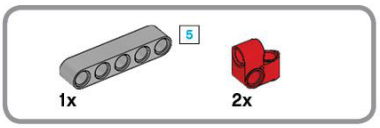
31



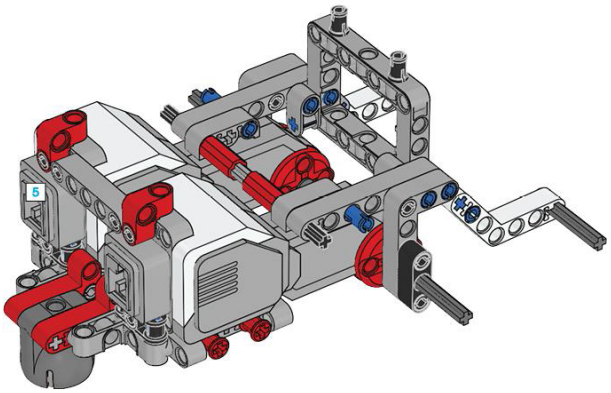


32

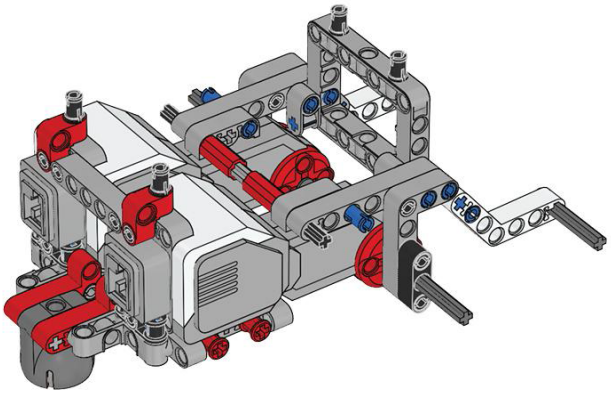


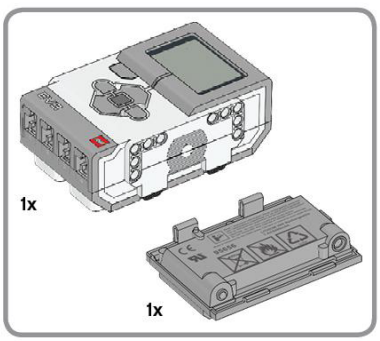


33

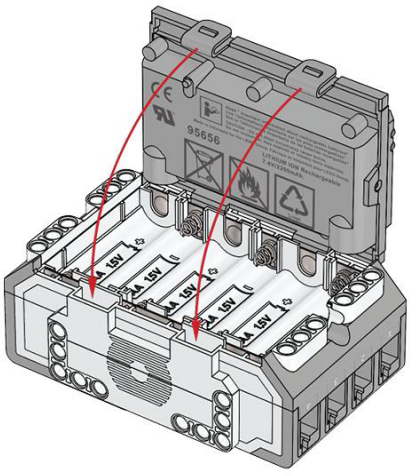


34

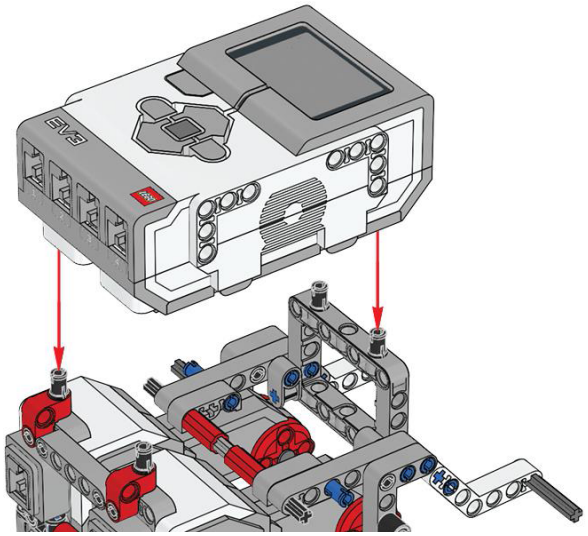


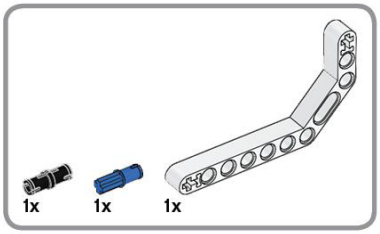


35

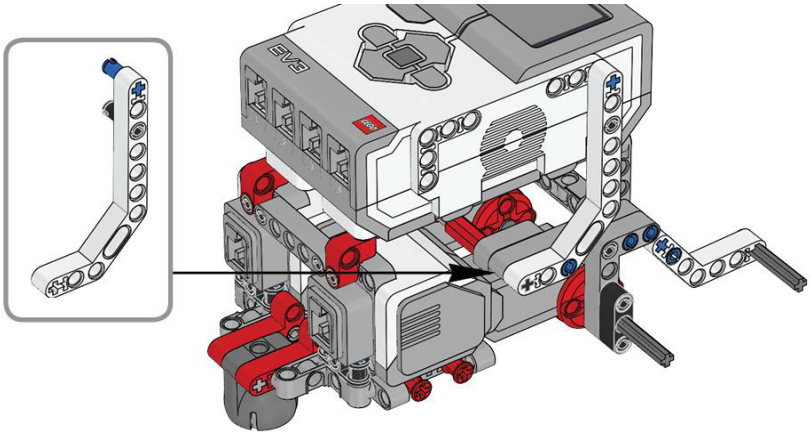


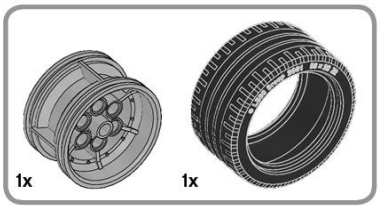
36



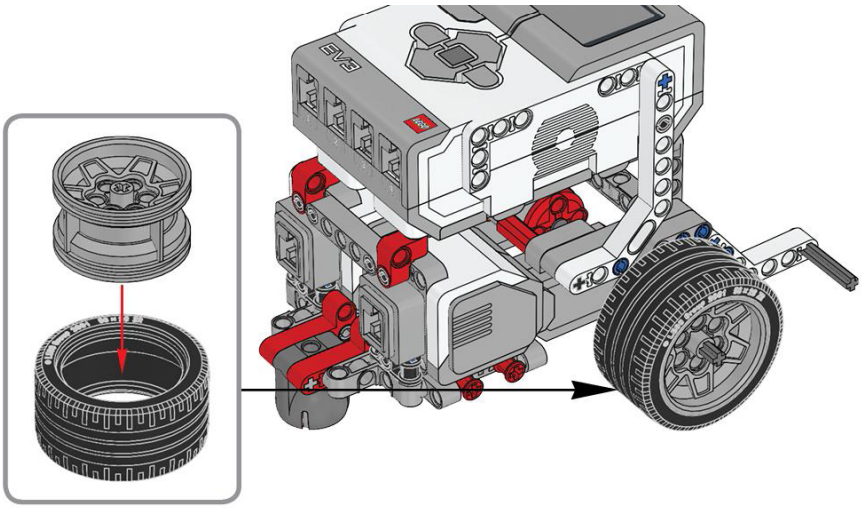


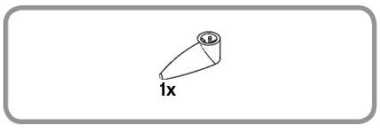
37



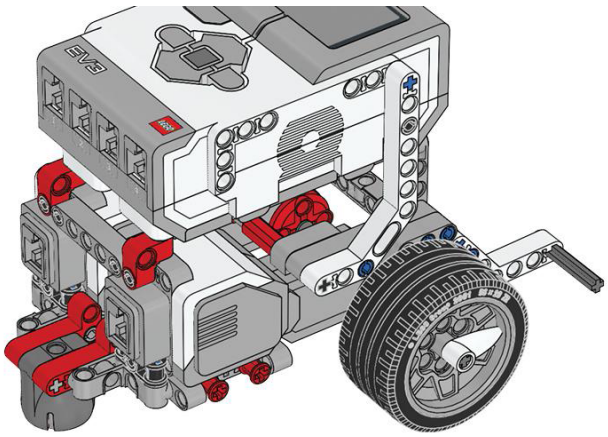


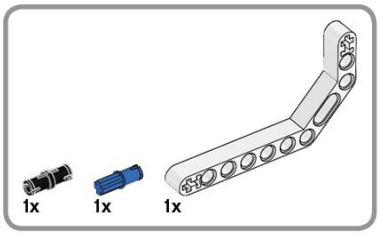
38



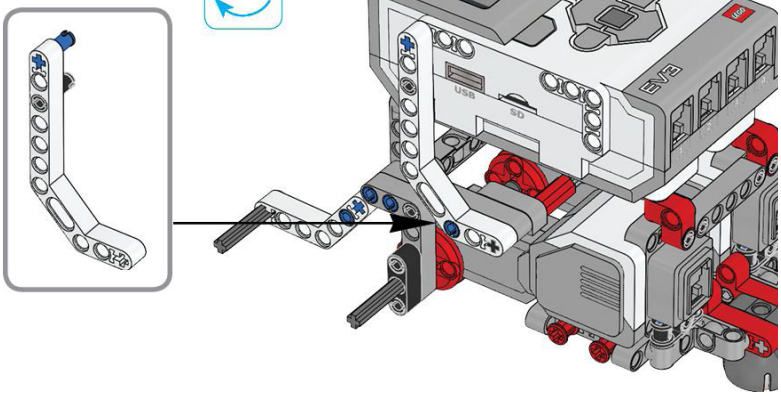


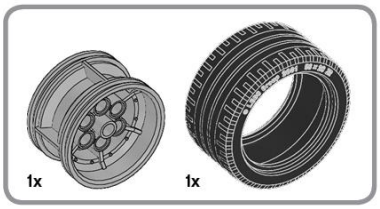
39



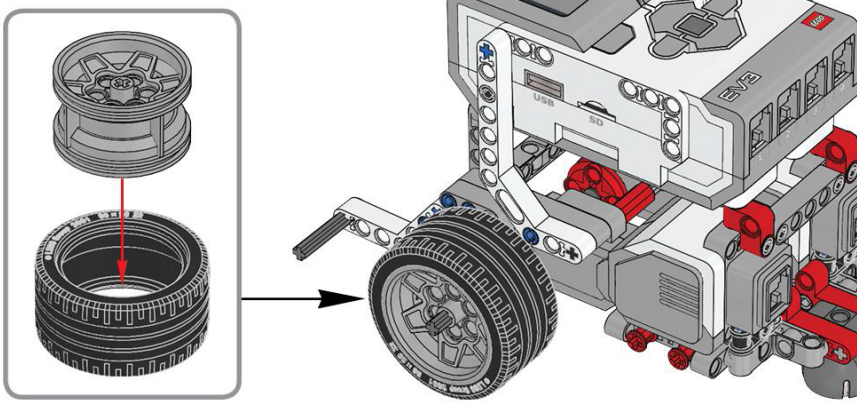


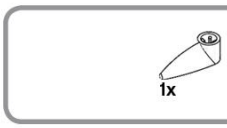
40



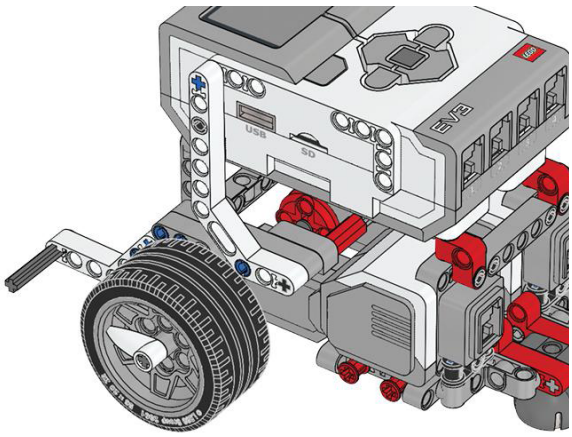


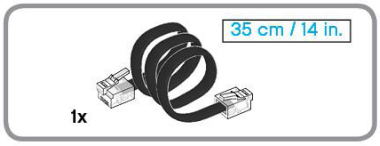
41



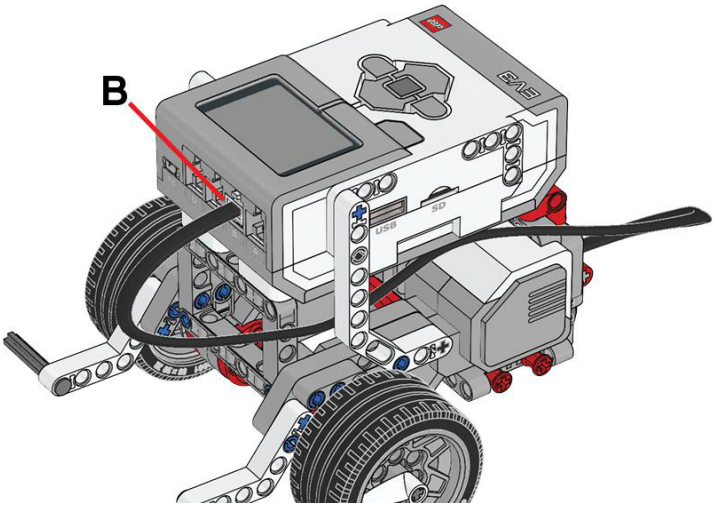


42

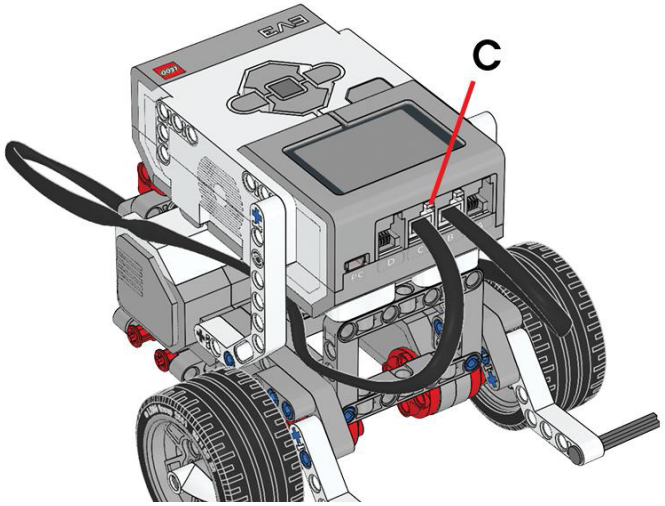




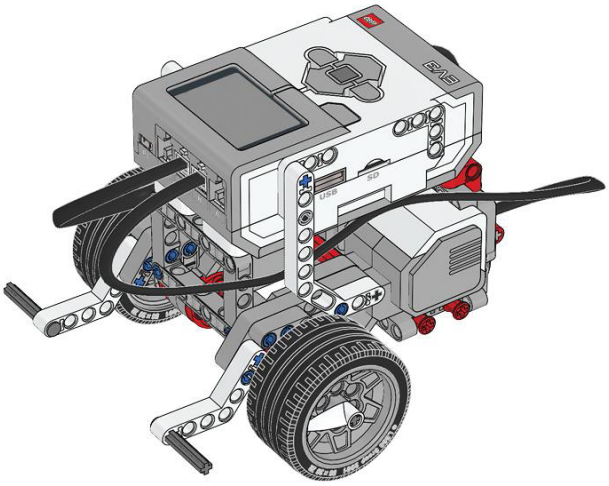
43



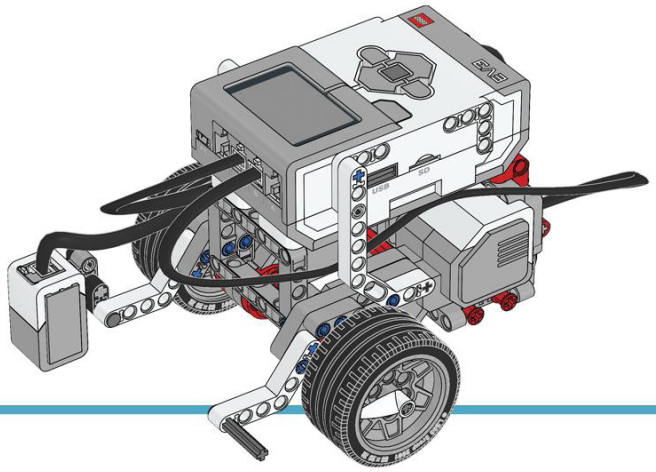
44

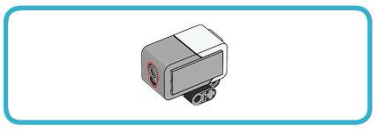


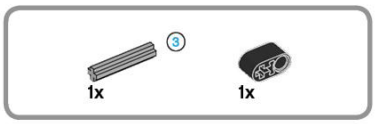
45



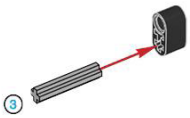
Bauanleitung: Farbsensor

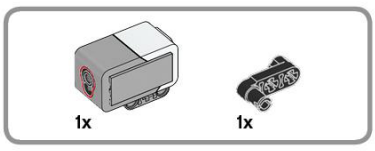




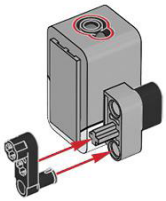


1



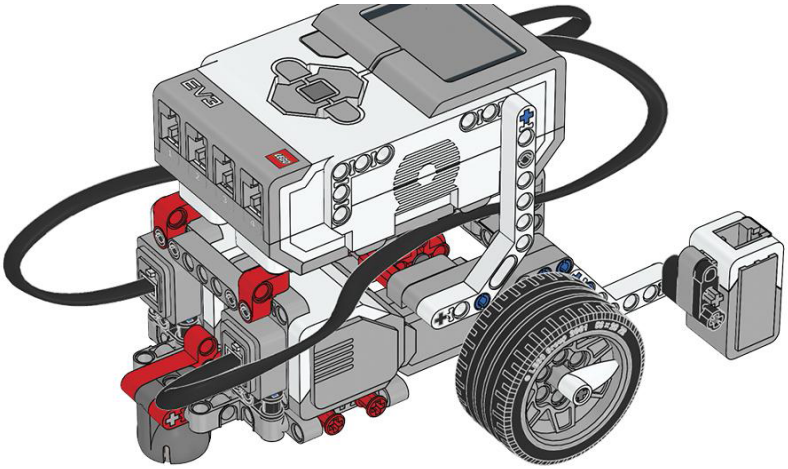


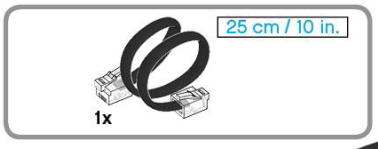
2



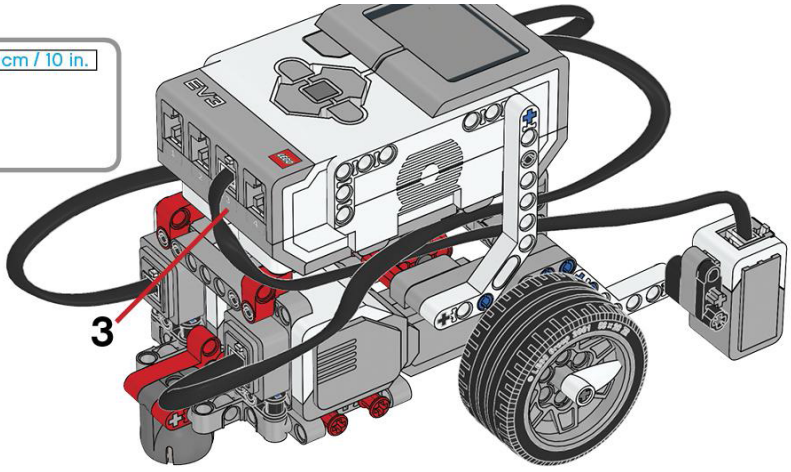
3

4

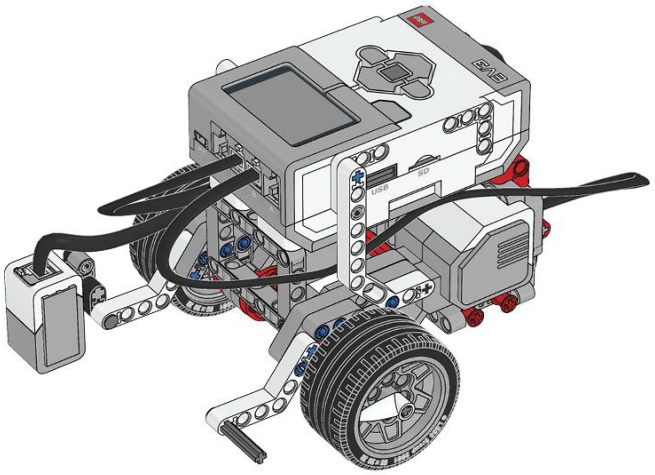




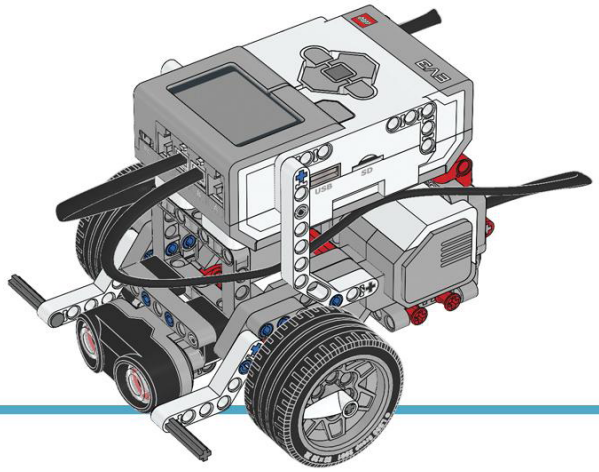
5

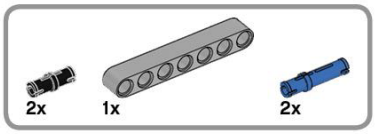


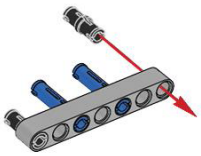
6



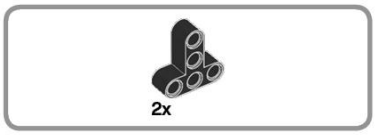
Bauanleitung: Ultraschallsensor

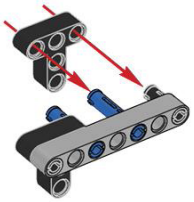






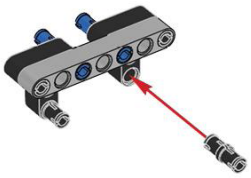
1

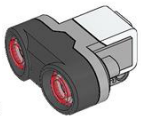




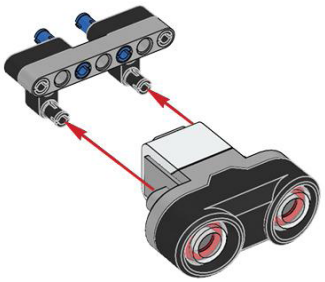
2

3

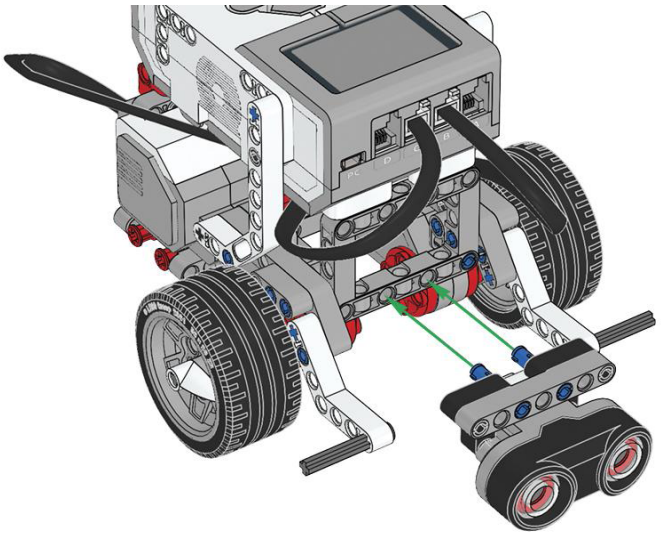


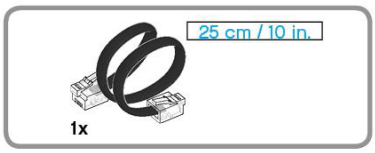


4

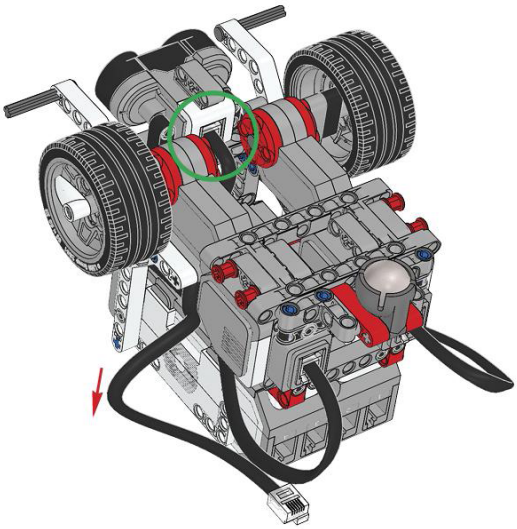


5

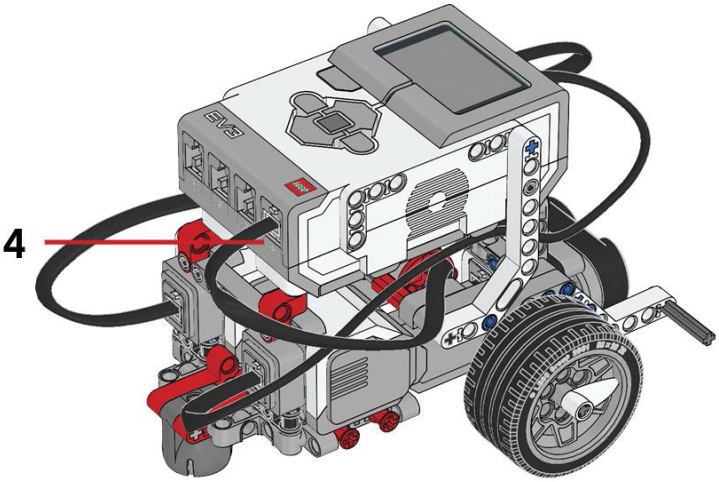




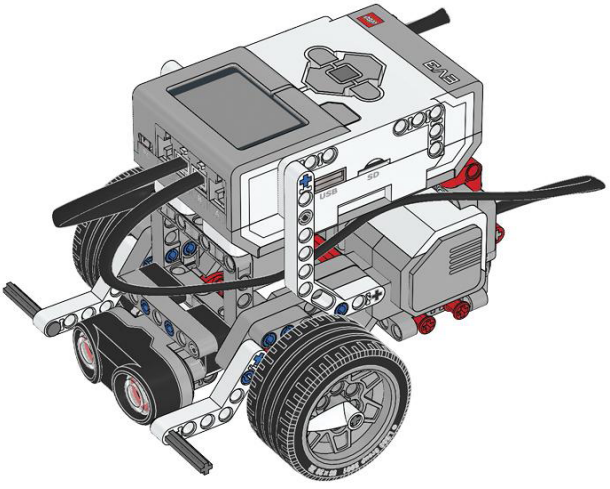
6



7



8



B Strecken

Möchte man die Simulation auf eine Strecke durchführen, kann man die nachfolgenden Strecken verwenden. Die Streckenvorlagen ausdrucken, ausschneiden und auf einem Untergrund befestigen, wie zum Beispiel Pappe. Achten Sie darauf es gut fest zu kleben und die Simulation in einem gut beleuchteten Ort zu starten. Platzieren den Lego Mindstorms Roboter so, dass der Farbsensor sich über dem linken Rand der schwarzen Linie befindet. Die Strecken sind für Geschwindigkeiten bis zu 25 cm/s geeignet.

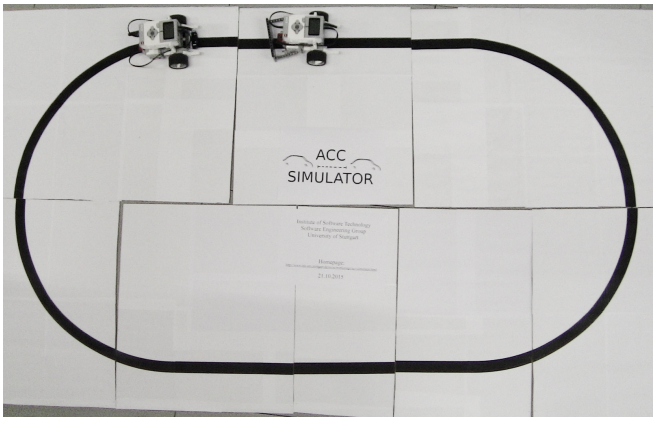
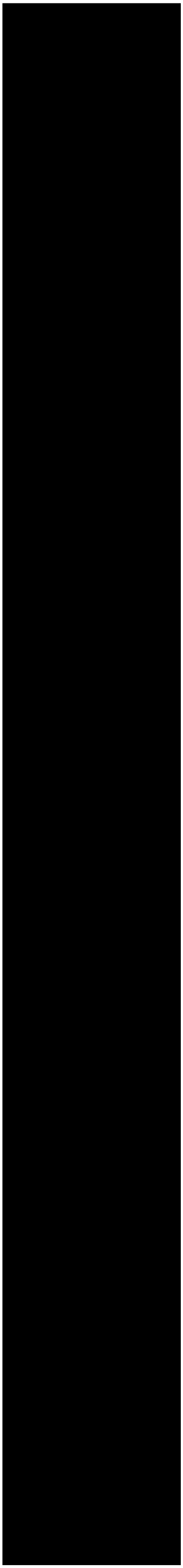
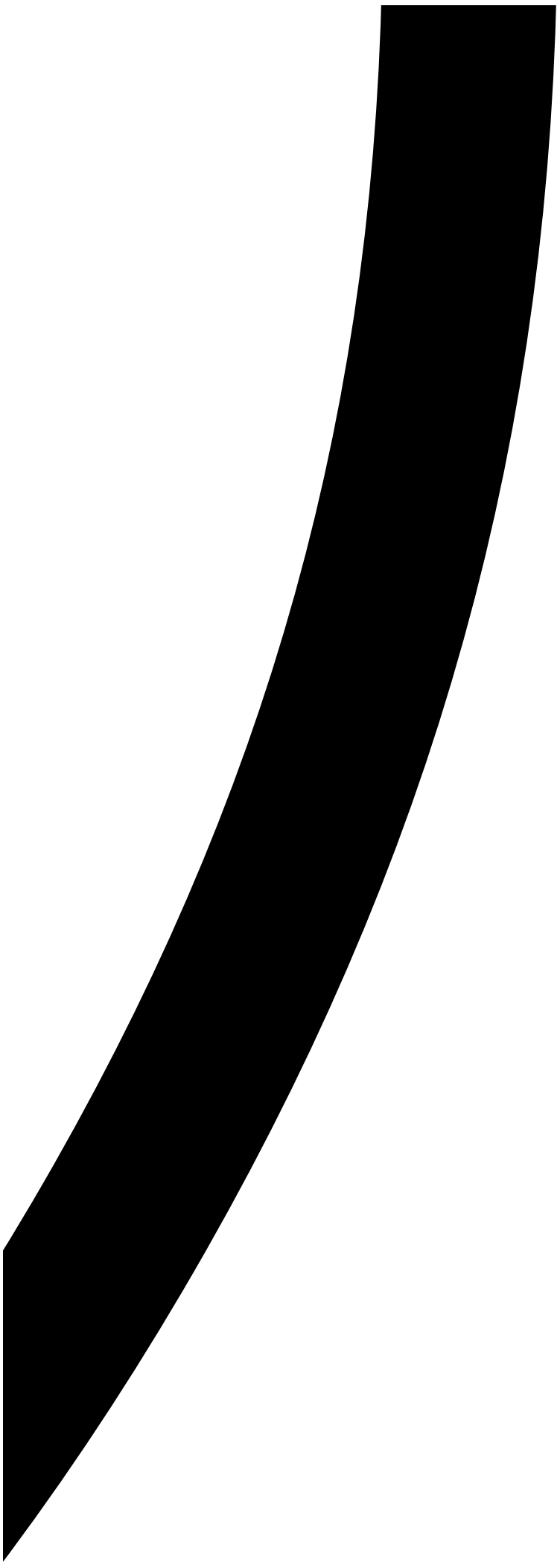


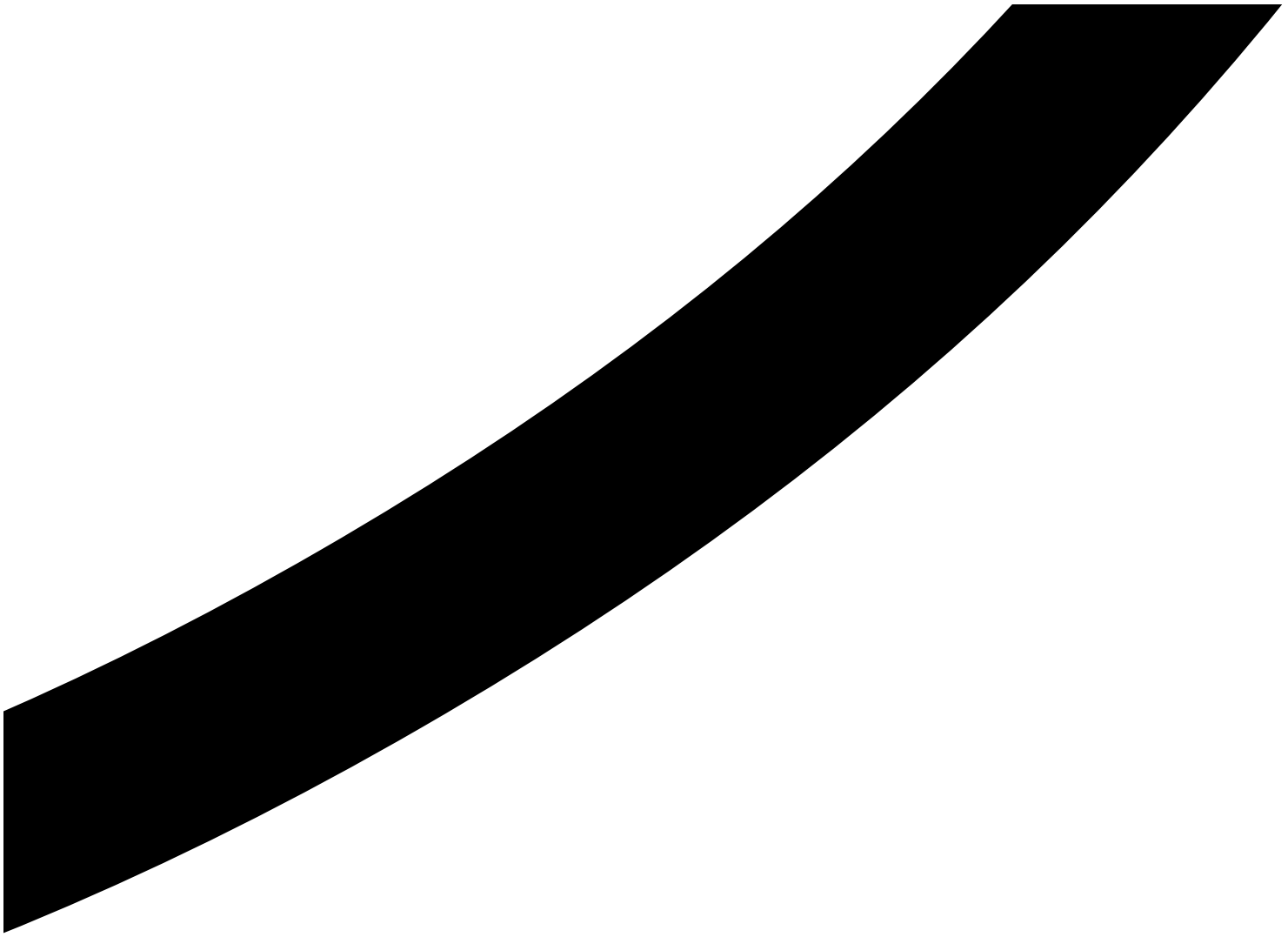
Abbildung B.1: Beispiel für einen Streckenaufbau

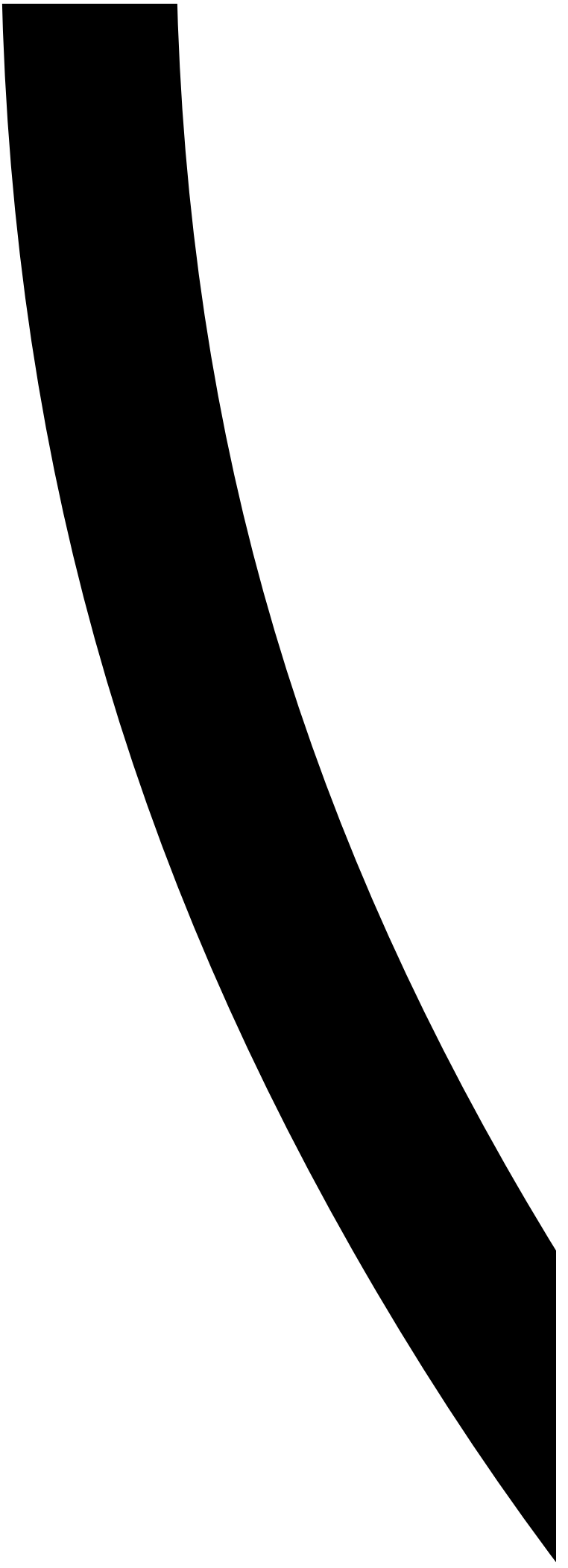
Die Streckenvorlagen sind in der folgenden Reihenfolge sortiert:

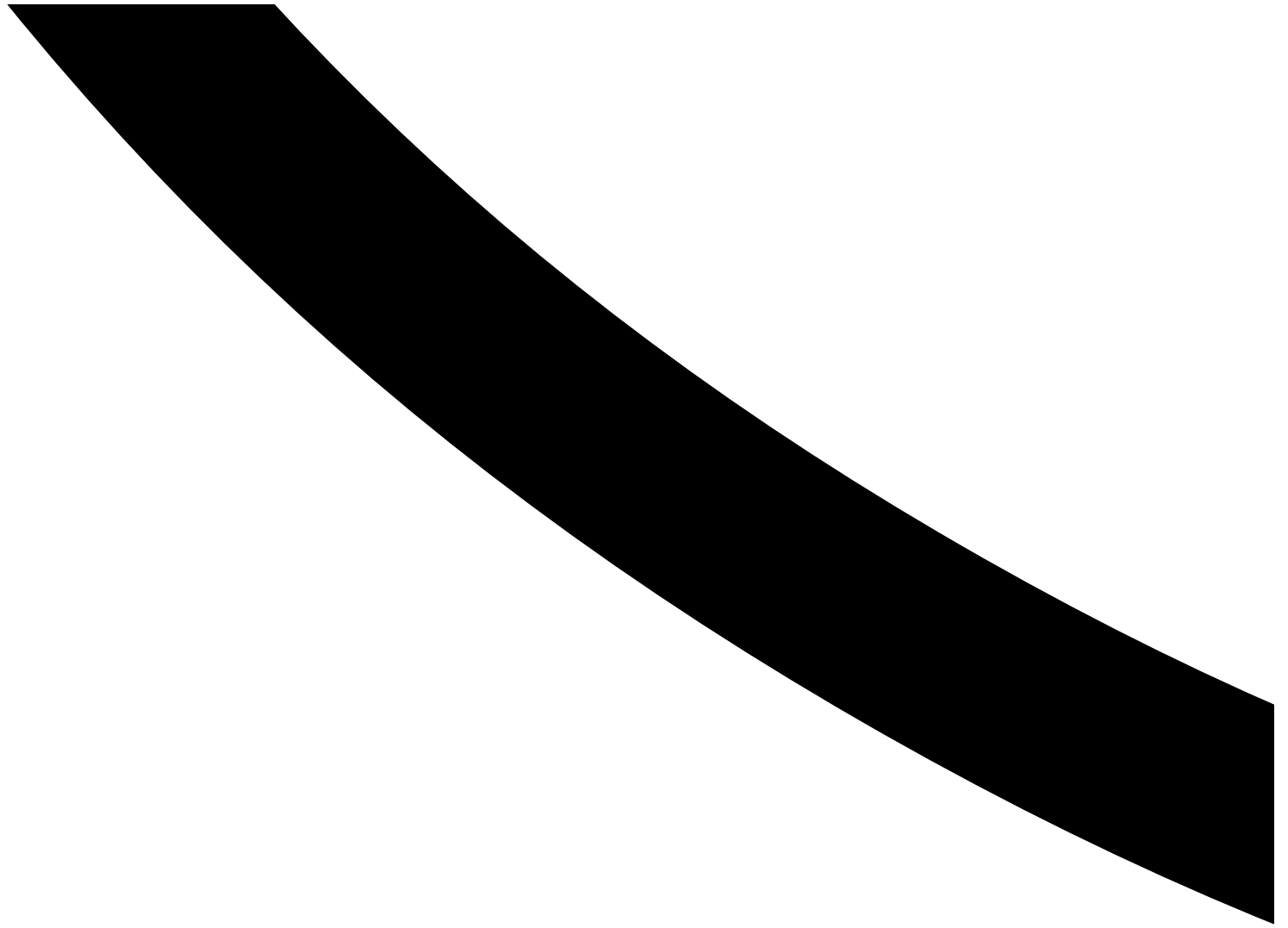
- gerade Strecke
- Kurve Links a
- Kurve Links b
- Kurve Links c
- Kurve Links d
- Kurve Rechts a
- Kurve Rechts b
- Kurve Rechts c
- Kurve Rechts d

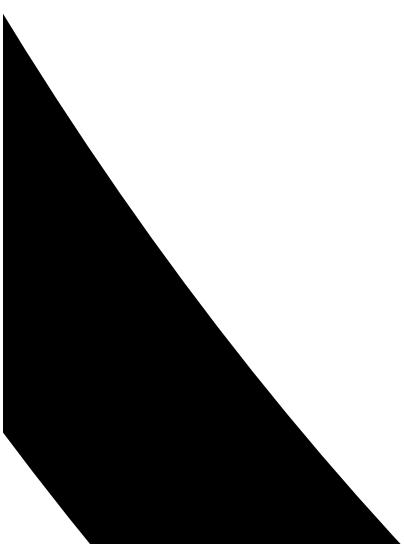












Literaturverzeichnis

[Gol] G. R. S.-U. Goltz. Allgemeine Beschreibung. URL http://www.ips.tu-braunschweig.de/docs/teaching/ws11-12/grs/vorlesung/Folien-2.5.pdf. (Zitiert auf Seite 13)

[GP05] M. Genzel, J. Padron. Adaptive Cruise Control. In *Autonome Fahrzeuge Seminar. Freie Universität Berlin*. 2005. (Zitiert auf Seite 11)

[Gur04] R. Gurulingesh. *Adaptive Cruise Control*. Dissertation, Indian Institute of Technology Bombay, 2004. (Zitiert auf Seite 13)

[NVP+09] G. Naus, R. Vugts, J. Ploeg, R. van de Molengraft, M. Steinbuch. Cooperative adaptive cruise control. In *IEEE automotive engineering symposium Eindhoven, The Netherlands*, Band 6. 2009. (Zitiert auf Seite 50)

[VNA00] P. Venhovens, K. NAAV, B. Adiprasito. Stop and go cruise control. *International journal of automotive technology*, 1(2):61–69, 2000. (Zitiert auf Seite 14)

Alle URLs wurden zuletzt am 3. 11. 2015 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift