Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Studienarbeit

Daniel Lorenz

HIL basierte Kalibrierung anhand des HAWKS Rennwagens

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Daniel Lorenz

Thema der Studienarbeit

HIL basierte Kalibrierung anhand des HAWKS Rennwagens

Stichworte

HIL, Hardware-in-the-Loop, SIL, MIL, PIL, HAWKS, Modellierung, Smart Transducer

Kurzzusammenfassung

Gegenstand dieser Arbeit ist, die Begrifferklärung der HIL und das Kalibrieren mittels der HIL anhand eines Beispieles aus dem HAWKS Rannwagen zu untersuchen.

Daniel Lorenz

Title of the paper

HIL basing calibration on basis of HAWKS racing car

Keywords

HIL, Hardware-in-the-Loop, SIL, MIL, PIL, HAWKS, modelling, Smart Transducer

Abstract

The object of the assignment is the definition of HIL and to analyse calibration using HIL on the basis of an example of the HAWKS racing car.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

In der Industrie ist die Technik Hardware-in-the-Loop (HIL^G) seit Jahren etabliert und wird genutzt um Vorabtests von Hardware durchzuführen. Diese Studienarbeit beschäftigt sich mit der Begriffserklärung von HIL und untersucht in welchem Maße und mit welchem Vorgehen HIL in den Entwicklungsprozess eingebunden wird.
Anhand einer Marktanalyse werden die Werkzeuge der HIL erklärend aufgegriffen. Daraufhin wird die Telemetrie des HAWKS Rennwagens grob analysiert und anhand eines Beispielproblems versucht ein Versuchsaufbau zur Kalibrierung einer Hardware mittels HIL zu konstruieren.

1.1. Definition

Hardware-in-the-Loop ist eine Technik zum Testen von Hardware bei der Eingaben, Signaleingänge oder Sensoren simuliert, an das zu testenden System übergeben und die Ausgänge oder Zustandsübergänge des zu testenden Systems aufgezeichnet oder rückgeführt werden. Wie der Name HIL schon suggeriert wird bei HIL Realtime-Hardware getestet. „HIL wurde zunehmend für rapid control prototyping RCP^G, Validierung und Kalibrierung von elektronischen Kontrolleinheiten entwickelt“ (siehe ur Rehman, 2007).

Mit HIL-Simulationen ist es möglich einzelne Hardwarekomponenten, aber auch ganze Verteilte Systeme zu validieren. Man unterscheidet in der Regel zwischen „open loop“ und „closed loop“ HIL.
In einer open loop HIL-Simulation wird die Zustandänderung des zu testenden Systems oder dessen ausgehenden Signale entweder aufgezeichnet oder für frühzeitige Warnungen genutzt.
Anders als bei der open loop, haben die Zustandsänderungen oder Ausgangssignale bei closed loop Simulationen einen direkten Einfluss auf die Simulation selber. Es gibt eine Rückkopplung auf die Simulation, was einen anderen Simulationsablauf zufolge haben kann, indem die die Simulation auf den Ausgang der zu testenden Hardware reagiert (siehe Schlager, 2008).

1.2. Grundlagen

Die Zugrunde liegende Technik der HIL kann auch allgemein **X-in-the-Loop** (XIL^G) genannt werden. Das X gibt an, dass die Technik auf verschiedenen Ebenen verwendet werden kann, also austauschbar ist.
Eine XIL-Simulation besteht grundsätzlich aus mindestens zwei Komponenten, dem **System-under-test** (SUT^G), also das zu testende System, und dem **Simulator** der die Stimuli generiert und die Signale oder Zustandsänderungen des SUT's liest.

Model-in-the-Loop (MIL^G) ist eine Technik bei der statt Hardware, ein physikalisches Modell einer Hardware genutzt wird. Das Modell steht stellvertretend für die Hardware und kann gleichermaßen über entsprechenden Schnittstellen, wie bei der HIL, stimuliert und ausgelesen werden. Die Grafik A.1 verdeutlicht den Aufbau einer MIL.

Software-in-the-Loop (SIL^G) validiert Software. Dabei besteht das SUT aus einem Emulator in dem die besagte Software läuft (siehe Abb. A.2).

Processor-in-the-Loop (PIL^G) ist der HIL sehr ähnlich. Bei PIL ist das SUT Hardware, wobei die Hardware nicht ein Prototyp des zu implementierenden Systems sein muss, sondern die Hardware lediglich vom gleichen Prozessortyp sein muss. Das kann auch ein einfaches Evaluationsboard eines Prozessor-Herstellers sein.

Das mischen der verschiedenen X-in-the-Loop Techniken nennt man Hybride-Simulations-Modelle. Hier muss darauf geachtet werden, ob die Simulationen zueinander kompatibel sind. Bei HIL bzw. PIL ist eine kontinuierliche Simulation in der Regel notwendig, weil man das real-time Verhalten validieren möchte. Bei MIL und SIL darf man auch zeitdiskret Simulieren, was den Vorteil hat, nicht so schnelle und dadurch meist teure Hardware für die Simulation zu benötigen. HIL-Simulatoren sollten ca. um den Faktor 10 schneller sein als die zu testende Hardware. Dabei ergeben sich beispielsweise für moderne Steuergeräte im Automotive-Bereich Simulationsschrittweiten von ca. 500 μs (siehe Gühmann, 2007b).

Eine Simulation sollte eine hohe Testabdeckung haben, damit man verschiedene SUT-Konfigurationen miteinander Vergleichen kann. Die Testabdeckung ist abhängig von den Realtime-Anforderungen. Bei einer soft Realtime-Anforderung darf ab und an, eine Nachricht später ankommen oder vielleicht auch ausfallen. Ganz anders bei firm und besonders hard Realtime-Anforderungen. Die Signalverarbeitung oder das Zeitverhalten von Nachrichten muss klar definiert sein und darf in keinem Fall verspätet sein oder sogar ausfallen, ganz besonders bei hard Realtime-Anforderungen die Sicherheitskritisch sein können.

Für eine X-in-the-Loop Simulation sind zwei weitere Anforderungen wichtig:

- Beobachtbarkeit

- Reproduzierbarkeit

Während einer Simulation sollte man Daten Aufzeichnen oder sich mittels einer Live Anzeigen lassen können, ohne Einfluss auf die Simulation aus zu üben (Beobachtbarkeit). Das X-in-the-Loop sollte auch bis auf die SUT-Komponenten in ihrem Verhalten deterministisch sein, um Ergebnisse vergleichen oder reproduzieren zu können (Reproduzierbarkeit).

1.3. Motivation

Die Motivation HIL zu verwenden sind die Kosten bei der Entwicklung gering zu halten und komplexe verteilte Systeme in den Griff zu bekommen. Besonders die Automobil-Branche die Milliarden von Euro für die Entwicklung eines Fahrzeuges ausgibt, um den heutigen Sicherheitsanforderungen zu genügen und neue Technologien zu etablieren, möchte die Kosten so gering wie möglich halten. Audi gibt ca. 2 Milliarden Euro für die Entwicklung eines Fahrzeuges aus (Gesele 2008, mdl. Mitt.).
„The need of early concept decisions and comprehensive functional validation coupled with a rise in complexity and a reduction of prototyping hardware has lead to an increasing use of Model-in-the-Loop (MIL) and Hardware-in-the-loop (HIL) methods in the control unit development.“(siehe S.246 Gühmann, 2008, BMW Autoren: Franz Durstberger, Erwin Kranawetter).

Durch die HIL können hard-realtime Tests ohne Gefährdung von Personen oder großen Verschleiß von Ressourcen durchgeführt werden. Durch das kontrollierte Testen lassen sich Tests auf verschiedenen Ebenen durchführen und mit einander verbinden, Fehlerfälle kontrolliert simulieren und dadurch Zeit sparen. Schnellere Testdurchläufe lassen sich mit der Testautomatisierung realisieren.

1.4. Vorgehen

XIL steht für eine Methode oder Technik Tests durchzuführen und haben keinen spezifisches Vorgehen. Wie genau die XIL Simulationen in den Entwicklungszyklus eingebunden werden, ist Betriebsgeheimnis.
Eine etablierte und meist genutzte Vergehensweise ist bei BMW (vgl. S.249 Gühmann, 2008, BMW) oder Audi das V-Modell (Gesele 2008, mdl. Mitt.). Anhand des V-Modells kann man nach den Testen erkennen, in welcher Stufe des Entwicklungsprozesses der Fehler gemacht wurde und setzt kontrolliert dort an. Eine weitere und mit dem V-Modell mischbare Vergehensweise ist das Rapid-control-prototyping (RCP). Folgend die Schritte von RCP:

1. Analyse und Spezifikation (Modellierung)
2. temporäre Implentierung (Rapid Prototyping)
3. Testdurchführung
4. Auswertung der Testergebnisse
5. Dokumentation

1.5. Vorteile der HIL

HIL hat eine Vielzahl von Vorteilen. Folgend einige aufgelistet:

- komplexe Systeme in Griff bekommen
- schnelle Entwicklungszeiten
- in RCP integrierbar
- Vorabtests anhand von Modellen
- Testen in verschiedenen Detailtiefen
- mögliche schnelle Fehlererkennung
- Reproduzierbarkeit der Testergebnisse
- Wiederverwendbarkeit und somit Vergleichbarkeit
- einheitliche automatische Testergebnisse für Dokumentation
- kontrollierte fault Tests
- Kalibrierung mittels der HIL

2. Analyse

In diesem Kapitel sollen die verschiedenen im Handel erhältichen Tools, die man für RCP bzw. MIL und HIL braucht, anhand einer Marktanalyse erörtert werden. Anschließend folgt eine grobe Beschreibung der Telemetrie im HAWKS-Rennwagen. Es wird eine Beispielproblematik heraus gegriffen und näher betrachtet.

2.1. Marktanalyse

2.1.1. Modellierungstools

Simulink^G ist eine Plattform für das Model-Based Design dynamischer Systeme. Es bietet eine interaktive, grafische Entwicklungsumgebung mit individuell anpassbaren Blockbibliotheken, für die eine Reihe von Erweiterungen für spezielle Anwendungsgebiete zur Verfügung stehen (siehe The Mathworks, 2008).

Dymola^G ist ein Werkzeug zum Modellieren und Simulieren von komplexen Systemen. Es basiert auf der deklarativen Modellierungssprache Modellica.

2.1.2. DAQ-Cards, Smart Transducer & Co.

Für die Kommunikation zwischen SUT und HIL-Simulator braucht man geeignete Schnittstellen. Diese Schnittstellen müssen immer öfter Aufgaben wie das Aufzeichnen von Daten oder das Konvertieren von Signalen übernehmen. **Data-aquisition-cards** (DAQ^G) werden von verschiedensten Herstellern angeboten oder selbst hersgestellt. Sie nehmen Daten auf oder konvertieren Signale von analog zu digital oder umgekehrt, wie z.B. AD-Wandler.
Zum kommunizieren im Netzwerk müssen die Daten von Sensoren oder Aktoren in ein bestimmtes Format umgewandelt werden. In einigen Fällen gibt es auch zeitliche Anforderungen die eingehalten werden müssen. An dieser Stelle kommen **Smart Transducer**^{G} zum Einsatz. Sie besitzen eine Schnittstelle zum Sensor, eine Logik, die die Signale in ein für das Netzwerk angebrachtes Format umwandelt, und eine intelligente Schnittstelle zum Netzwerk,

wie z.B CAN-Bus. Smart Transducer werden von Herstellern mit standarisierten Schnittstellen angeboten.

2.1.3. Simulatoren

Simulatoren gibt es von verschiedenen Firmen. Die Simulatoren sind in der regel mit schnellen auf das Testen optimal angepassten Prozessoren ausgestattet und bieten damit die Möglichkeit echtzeitfähige Tests durchzuführen.
Solch Simulatoren werden häufig mit DAQ-Cards oder Smart Transducern und unter Umständen auch mit einem Realtime-Betriebssystem ausgeliefert.

2.1.4. Echtzeit- und Live-Anzeigen

Über sogenannte Human Machine Interfaces (**HMI**^{G}) kann man eine Simulation verfolgen oder ggf. manuell Eingreifen, wenn die Bedienung es erlaubt. Damit der Benutzer die Daten in einer für ihn gut verständlichen Form schnell und effizient ablesen kann, gibt es Werkzeuge für solche Live-Anzeigen.
Simulink oder **LabVIEW** bietet grafische Elemente zum Anzeigen der Daten in einer für den Menschen angemessenen lesbaren Form.

2.2. Einführung in die Telemetrie des HAWKS Rennwagen

Die Telemtrie des HAWKS Rennwagens HAWK08 ist der eines normalen PKWs sehr ähnlich. Sensoren, die über den Rennwagen verteilt sind, geben Daten über Raddrehzahl, Kühlwassertemperatur, Öldruck u.s.w. an. Diese Daten werden von einem Mikrocontroller für die Kommunikation vorbereitet und über einen CAN-Bus geschickt. Der Sensor und der Mikrocontroller ergeben zusammen einen Smart Transducer. Die Subsysteme, die an diesen Daten interessiert sind, lesen zur gegebenen Zeit die Daten vom CAN-Bus ab.

2.2.1. Verteiltes System

Das vorliegende System ist ein verteiltes System und hat daher bestimmte Anforderungen an Signallaufzeiten, somit auch an eine Simulation. Der Vorteil an solch einem verteilten System liegt in dem Komponentenweise Testen der Hardware. Damit ist es leichter Komponenten- oder Modultests durchzuführen und dadurch Vorabtests mit Subsystemen zu machen und

Fahlerquellen besser eingrenzen zu können. Im Anhang A.4 finden Sie eine grobe Darstellung des Systems.

2.2.2. TTA

Die Architektur, die hier implementiert wurde, ist eine Time-triggered-Architektur (**TTA**^{G}). Der CAN-Bus (**TTCAN**^{G}) ist in sogenannten Matrix-Zyklen organisiert, die 40ms lang sind. In einen Matrix-Zyklus gibt es Zeit-Slots in denen nur ein Teilnehmer auf dem Bus schreiben darf. Die Zeit-Slots sind den Teilnehmern offline zugewiesen. Das heißt, dass vor Inbetriebnahme des CAN-Busses die Teilnehmer des Gesammtsystems bekannt sind und einen oder mehr Zeit-Slots zugewiesen bekommen haben. Es wird nicht dynamisch während der Laufzeit entschieden, welcher Teilnehmer auf den Bus schreiben darf (siehe Haase, 2007).

Dadurch ergeben sich Zeitaussagen über die Kommunikation. Ein Teilnehmer, der z.B. einen Zeit-Slot hat, kann nur alle 40ms eine Nachricht versenden. Somit würde sich für die Simulation eine Simulationsschrittweite von ca. 40ms anbieten.

2.2.3. Sicherheitsanforderungen

Die Sicherheitsanforderungen an die elektronischen Komponenten sind bei dem Rennwagen sehr gering, weil die Vorgaben der Formula Student - dafür wurde der Rennwagen konstruiert - besagen, dass sicherheitsrelevante Eigenschaften des Fahrzeuges wie Bremse oder Lenkung nicht elektronisch realisiert werden dürfen. Unter anderem greift momentan kein Steuergerät oder ähnliches während der Fahrt dynamisch auf das Fahrverhalten des Rennwagens ein.

2.3. Beispiel: Automatische Gangschaltung

Um zu verdeutlichen wie das Kalibrieren mittels der HIL funktionieren soll, wird an dieser Stelle eine beispielhafte Problematik am HAWKS Rennwagen herausgegriffen und erörtert.

Bisher schaltet der Fahrer selber die Gänge mittels Schaltwippen am Lenkrad. Der Fahrer muss während der Fahrt sich darauf konzentrieren zum optimalen Schaltmoment zu schalten. Man könnte eine automatische Gangschaltung implementieren, die einerseits den Fahrer vom Schalten der Gänge entlasten würde und andererseits schneller und präziser

auf den optimalen Schaltmoment reagiert.

Für die automatische Gangschaltung soll zwischen zwei verschiedenen Fahr-Szenarien unterschieden werden, geradeaus Beschleunigen (Acceleration) und den kurvenreichen Rundkurs (Endurance). Auf die markenten Unterschiede wird weiter unten eingegangen.
Für das Sybsystem, automatische Gangschaltung, wurden folgende Sensoren bzw. Aktoren herausgearbeitet (siehe Anhang Abb. A.5):

- Gear-Control-Unit (**GCU**^{G})
- Raddrehzahlsensoren
- Motor bzw. die sensorischen Daten der Motorsteuerung
- Fahrer bzw. Gas- und Bremspedal
- Mechanische Schaltung

Die GCU gibt den Befehl an die mechanische Schaltung den Gang zu wechseln. Damit die GCU weiss wann sie den Befehl zum schalten geben muss, brauch sie die Geschwindigkeit des Fahrzeuges und die Motordrehzahl. Die Motordrehzahl erhält sie über die Motorsteuerung. Für die Geschwindigkeit werden die Daten der Raddrehzahlsensoren gemittelt. Um nicht falsche Werte zu erhalten, weil die hinteren Räder Schlupf haben, werden nur die Raddrehzahlen der vorderen Räder genommen.

Für das Fahr-Szenario **Acceleration** ist das Schalten der Gänge einfacher als beim Rundkurs. Beim Schalten der Gänge muss darauf geachtet, dass man nach dem Gängeschalten nicht aus dem optimalen Leistungsbereich fällt und somit keine optimale Beschleunigung hat (siehe Abb. 2.1).

In dem Zeitraum in dem der Gang gewechselt wird, gibt es einen Geschwindigkeitsabfall der auf der Abb. 2.1 mit den Pfeilen dargestellt wird. Jeder Gang besitzt eine charakteristische Drehmoment-RPM^G-Kurve die angibt, bei welcher Drehzahl das Drehmoment, also die momentane Kraft des Motors, am stärksten ist (siehe Anhang Abb. A.6). Beim Gangwechsel sollte hier nicht nur darauf geachtet werden, dass man im optimalen Leistungsbereich bleibt (in der Abb. A.6 mit der Ellipse eingezeichnet), sondern der nächste Gang nach dem Wechsel ein möglichst gleiches oder höheres Drehmoment hat.

Beim Fahrer-Szenario **Rundkurs** sollten mehrere Faktoren bezogen werden. Die Problematik die sich bei diesem Szenario äußert, sind die variabel langen geraden Strecken im wechsel mit den Kurven. Folgendes Problem könnte auftreten: Die Gangschaltung kennt die Strecke nicht und könnte womöglich kurz vor einer Kurve in den nächst höheren Gang

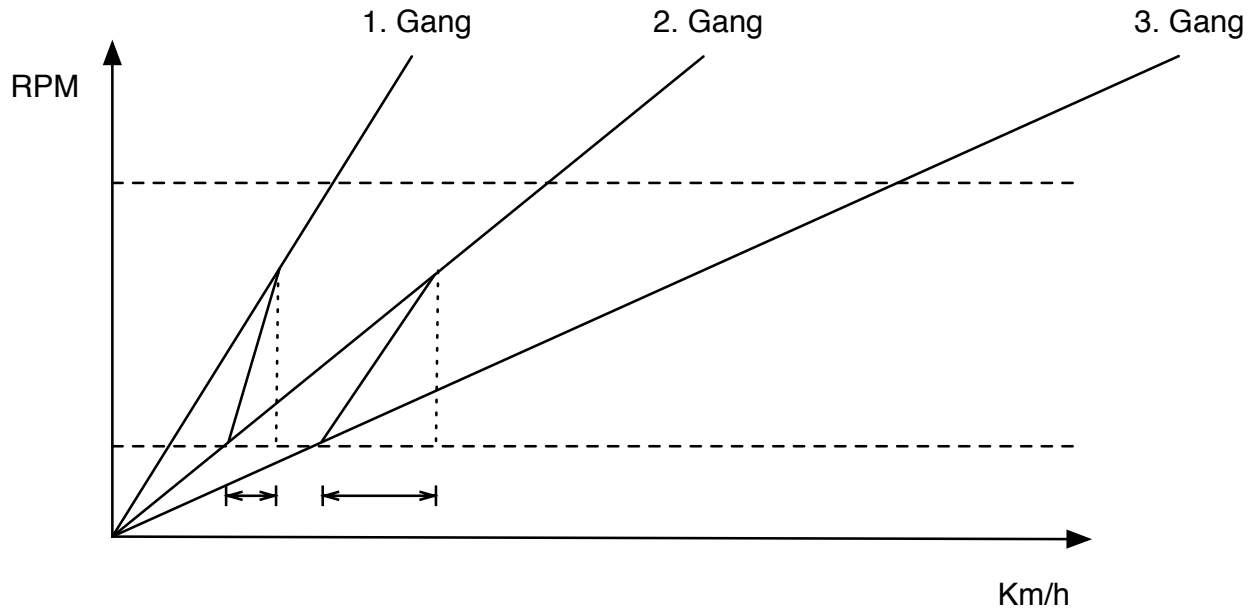

Abbildung 2.1.: Verhältnis RPM zu Km/h für das Gängeschalten (der Raum zwischen den beiden gestrichelten Linien ist der optimale Leistungsbereich)

schalten. Der Fahrer geht auf die Bremse, um die Kurve optimal anzufahren und nicht raus zu fliegen, und ist im nächsten Moment nicht mehr im optimalen Leistungsbereich. Die Gangschaltung bemerkt das und schaltet wieder runter.

Dadurch ergaben sich zwei unnötige Gangwechsel, die einen Zeit- und somit auch einen Geschwindigkeitsverlust verursachten. Man kann um das Problem aus dem Weg zu räumen den Moment des Schaltens in einer höheren Motodrehzahl ansätzen, dabei stellt sich aber schnell die Frage wie weit hoch und hat sich das Problem nicht einfach nur verlagert.

Um das Problem nicht zu verlagern und den optimalen Schaltmoment zu finden sollte noch die Stellung des Brems- und Gaspedals mit einbezogen werden. Dazu mehr im nächsten Kapitel.

3. Versuchsaufbau

Dieses Kapitel soll eine kleine Anregung bieten, wie man HIL zum Kalibrieren einer Kennlinie für das Beipiel aus dem letzten Kapitel anwenden könnte.

3.1. Anforderungen an den Versuchsaufbau

Die Anforderungen sind eine closed Loop HIL mit einer Testabdeckung für ein firm real-time System. Das System sollte keine ausfallende oder zu spät kommende Nachrichten haben, sonst bringt der Aufwand einer automatischen Gangschaltung nichts, wenn sie ab und an langsamer reagieren würde als der Fahrer selber. Um einmal zu verdeutlichen in welchen zeitlichen Anforderungen sich das System in seinem Nutzungskontext ungefähr befindet, die folgende Rechnung 3.1 (Beispielrechnung für 100 Km/h):

$$\frac{100 * 1000}{3600 * 1000} m/ms \tag{3.1}$$

Nach der Rechnung fahren wir also innerhalb von 100ms ca. 3 Meter weit.

3.2. SUT

Das zu testende System besteht aus der GCU. Die soll anhand einer gegebenen Kennlinie validiert werden. Die Kennlinie gibt vor, wann geschaltet werden soll. Die Kennlinie besteht aus den einstellbaren Variablen das vorherigen Kapitels (Motordrehzahl, Raddrehzahl, u.s.w.). Nach einer Testsimulation soll das SUT mit einer neuen Kennlinie validiert werden. Die Testsimulation wird so oft durchgeführt, bis die neue Kennlinie sich nur zu einem kleinen δ zur vorherigen in den Ergebnissen unterscheidet. Die große des δ kann zu diesem Zeitpunkt noch nicht beschrieben werden.

3.3. HIL Simulator

Der HIL-Simulator ist um einiges komplexer. Er besteht aus einer Fahrer-Simulation, einem Modell des Antriebsstranges und der mechanischen Gangschaltung, sowie einen Smart Transducer.

3.3.1. Fahrer-Simulation

Die Fahrer-Simulation simuliert ein bestimmtes Fahrverhalten eines Fahrers auf einem Rundkurs. Dabei werden Gas- und Bremspedal simuliert und in das Modell geführt (siehe Abb. A.7). Die Fahrer-Simulation ist abhängig von dem Ort der Strecke, der von dem Model übergeben wird.

3.3.2. Modell

Das Modell errechnet anhand des Ganges und der Stellung des Gas- und Bremspedals die Geschwindigkeit, den Ort auf der Strecke (idealisiert) und die Drehzahl. Diese Daten werden an Fahrer-Simulation und Smart Transducer weiter gegeben. Das Modell berücksichtigt außerdem das Verhalten des SUT indem es den Gang wechselt, nachdem es das Signal von dem SUT erhalten hat. Dabei berücksichtigt das Modell die Zeit, die die mechanische Gangschaltung zum schalten braucht.

3.3.3. Smart Transducer

Der Smart Transducer verarbeitet die Daten aus dem Modell so, dass sie korrekt, zeitlich und in der Form, oder absichtlich falsch auf dem realen CAN-Bus - die Verbindung zu dem SUT - liegen.

3.4. Datenaufzeichnung

Während der Simulation werden alle Daten für weitere Auswertungen oder spätere Emulationen aufgezeichnet.

3.5. Kennlinienberechner

Der Kennlinienberechner wertet die Ergebnisse aus und errechnet eine neue Kennlinie nach jeder Test-Simulation. Dabei stehen dem Kennlienenberechner alle Daten von der Datenaufzeichnung bereit.
Nach welchen Kriterien und wie der Kennlinienberechner rechnet, muss noch untersucht werden.

4. Ausblick

Für weitere Untersuchungen sollte der Kennlinienberechner erörtert werden. Außerdem steht die Definition der Schnittstellen noch offen. Taktraten und Art müssen untersucht werden.
Das Kalibrieren mittels HIL hat ein großes Potenzial. Gerade bei komplexen Systemen, die nicht so einfach mathematisch zu lösen sind, hilft dieser Ansatz eine gute Kennlinie zu erhalten. Dieser Aufbau sollte nicht als ein Testaufbau nur für die GCU gesehen werden, sondern als allgemeiner Ansatz. Es könnte auch die Motorsteuerung für das SUT stehen, nur das dieser Ansatz um einiges komplexer in der Simulation und Modellierung wäre.
Ansonsten sollte dieser Ansatz alle Vorzüge einer HIL besitzen.

4.1. Risiken

Ein Risiko dieses Ansatzes wäre die Annahme ein falsches Modell wäre richtig. Zudem kann das Modell auch unvollständig sein und somit nicht alle Möglichkeiten abdecken. Desweiteren kann so ein Modell auch unendlich komplex werden (Beispiel Motorsteuerung mit mind. 15 Stellgrößen) und damit gar nicht in den Griff zu bekommen sein.

Für zeitkritische Anwendungen braucht der HIL-Simulator eine gute Performance. Es kann aber dazu führen, dass es keine oder nur sehr teure Hardware gibt, die diese Performance leisten kann.
Zuletzt bleibt auch noch das Problem, dass Sensordaten nicht richtig sind oder verrückt spielen und damit die Berechnung des Kennlinienberechners stark verfälschen.

4.2. Möglichkeiten

Wenn der Kennlinienberechner gut funktioniert, könnte man so einen auch in den HAWKS-Rennwagen einbauen. So könnte man diesen nutzen, um dynamisch nach einer abgefahrenen Runde im Fahr-Szenario Rundkurs während der Fahrt die Kennlinie der GCU einzustellen. Die Machbarkeit bleibt aber noch zu untersuchen.

Literaturverzeichnis

[Gühmann 2007a] GÜHMANN, Dr.-Ing. C.: HIL-Simulation für die automatisierte Prüfstandsapplikation. In: *ACM* (2007)

[Gühmann 2007b] GÜHMANN, Dr.-Ing. C.: Modellbildung und Testautomatisierung für die Hardware-in-the-Loop Simulation. In: *ACM* (2007)

[Gühmann 2008] GÜHMANN, Dr.-Ing. C.: *Simulation und Test in der Funktions- und Softwareentwicklung für die Automoilelektronik II*. Expert Verlag, 2008. – ISBN 978-3-8169-2818-8

[Haase 2007] HAASE, Sebastian: *Telemetrie im Formula Student Rennwagen auf Basis von CAN Bus, DatenspeicherungundWireless LAN Technologien*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2007

[King 2007] KING, P.J.: HARDWARE IN THE LOOP FOR AUTOMOTIVE VEHICLE CONTROL SYSTEMS DEVELOPMENT. In: *IEEE Explore* (2007)

[ur Rehman 2007] REHMAN, Naveed ur: RTLinux based Simulator for Hardware-in-the-Loop Simulations. In: *IEEE Explore* (2007)

[Schlager 2008] SCHLAGER, Martin: *Hardware-in-the-Loop Simulation*. VDM Verlag Dr. Müller, 2008. – ISBN 978-3-8364-6216-7

[The Mathworks 2008] THE MATHWORKS, Inc.: *Simulink*. 2008. – URL http://www.mathworks.de/products/simulink/

A. Anhang

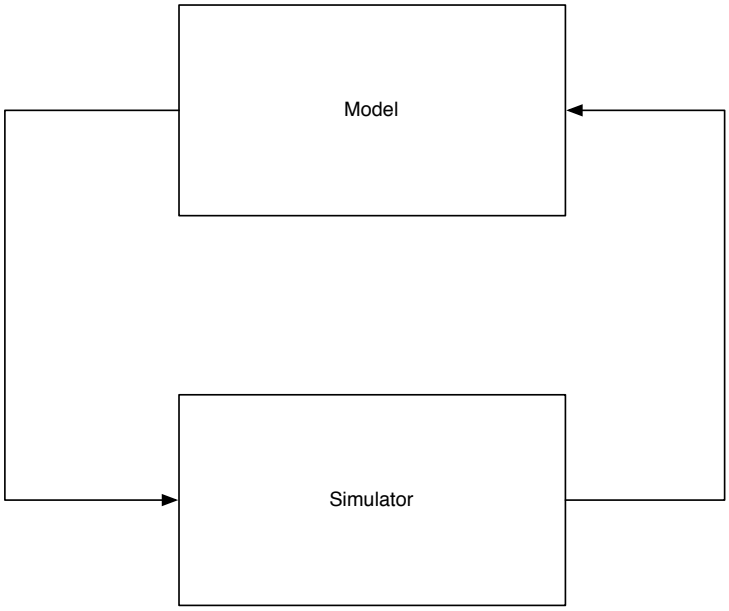

Abbildung A.1.: Model-in-the-Loop (MIL)

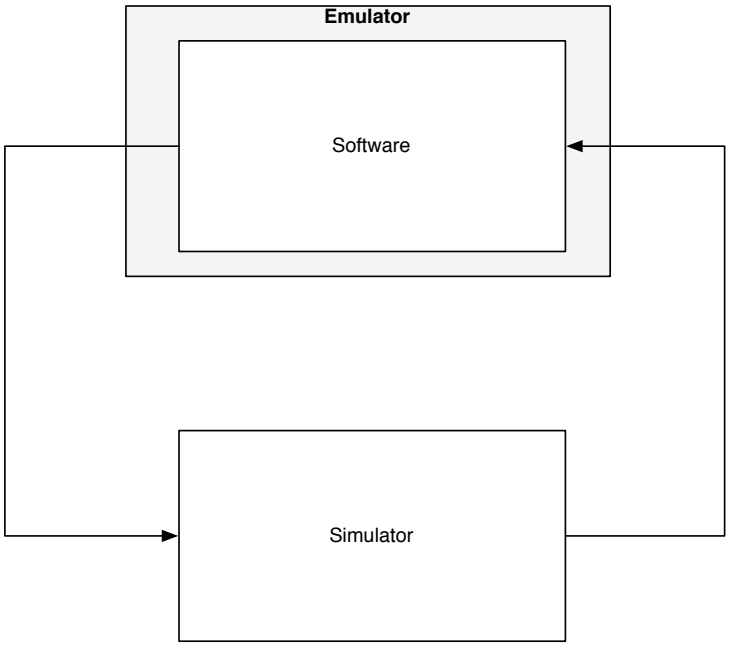

Abbildung A.2.: Software-in-the-Loop

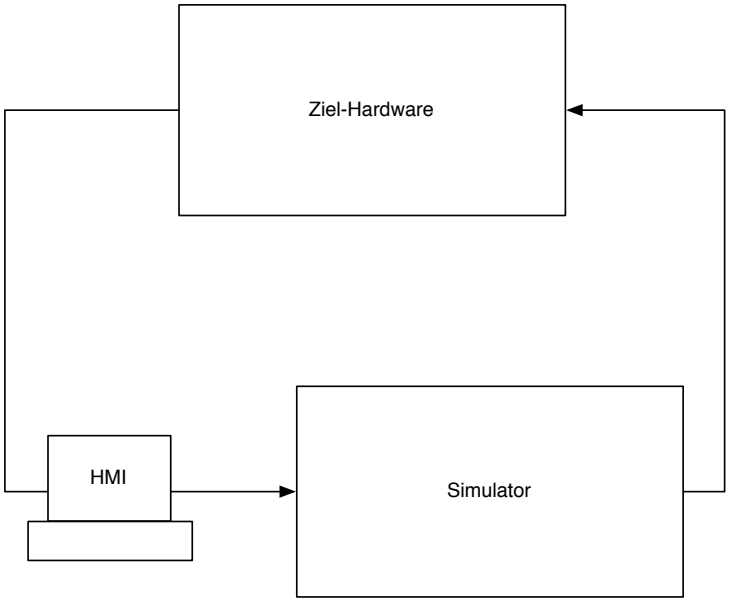

Abbildung A.3.: HIL bzw. PIL

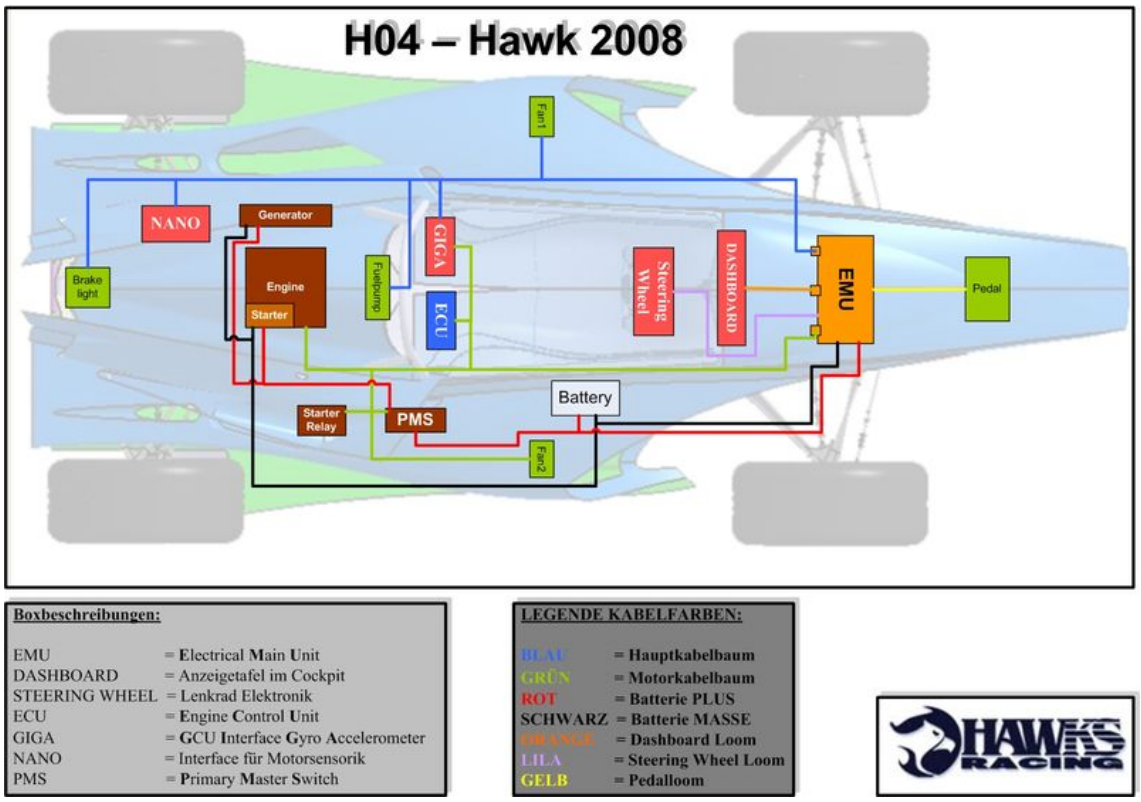

Abbildung A.4.: Grobe Darstellung der Telemetrie im HAWK08

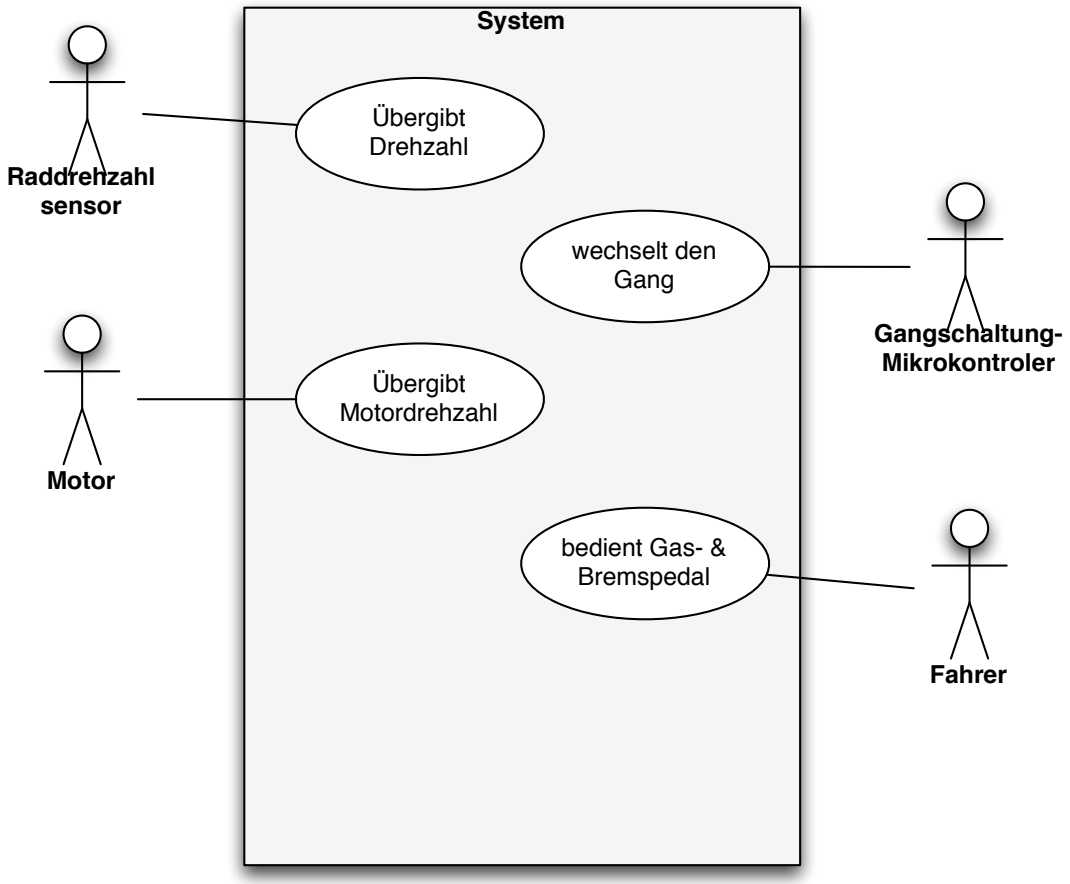

Abbildung A.5.: UseCase des relavanten Subsystem für die automatische Gangschaltung

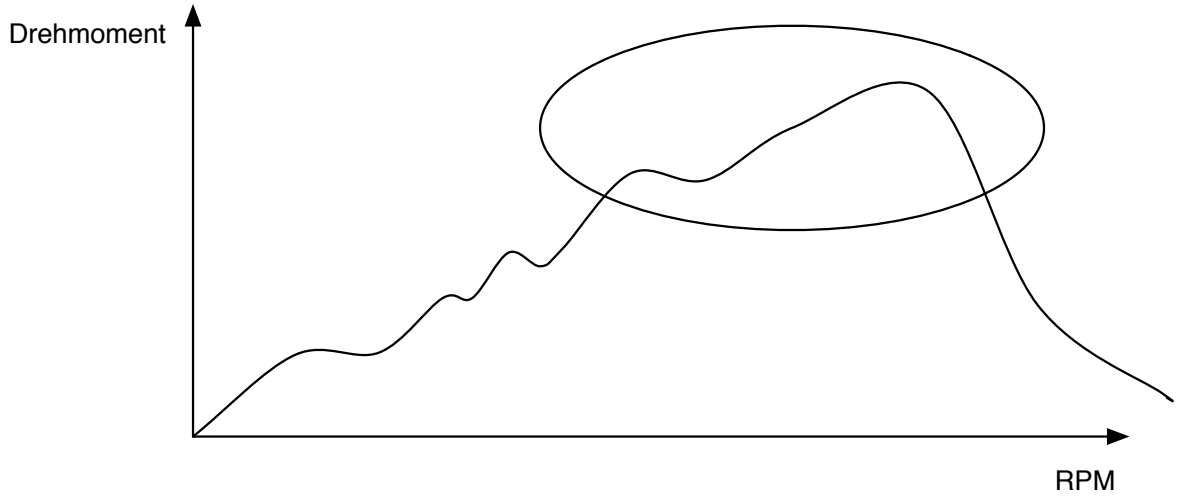

Abbildung A.6.: Beispiel einer Drehmoment-RPM-Kurve

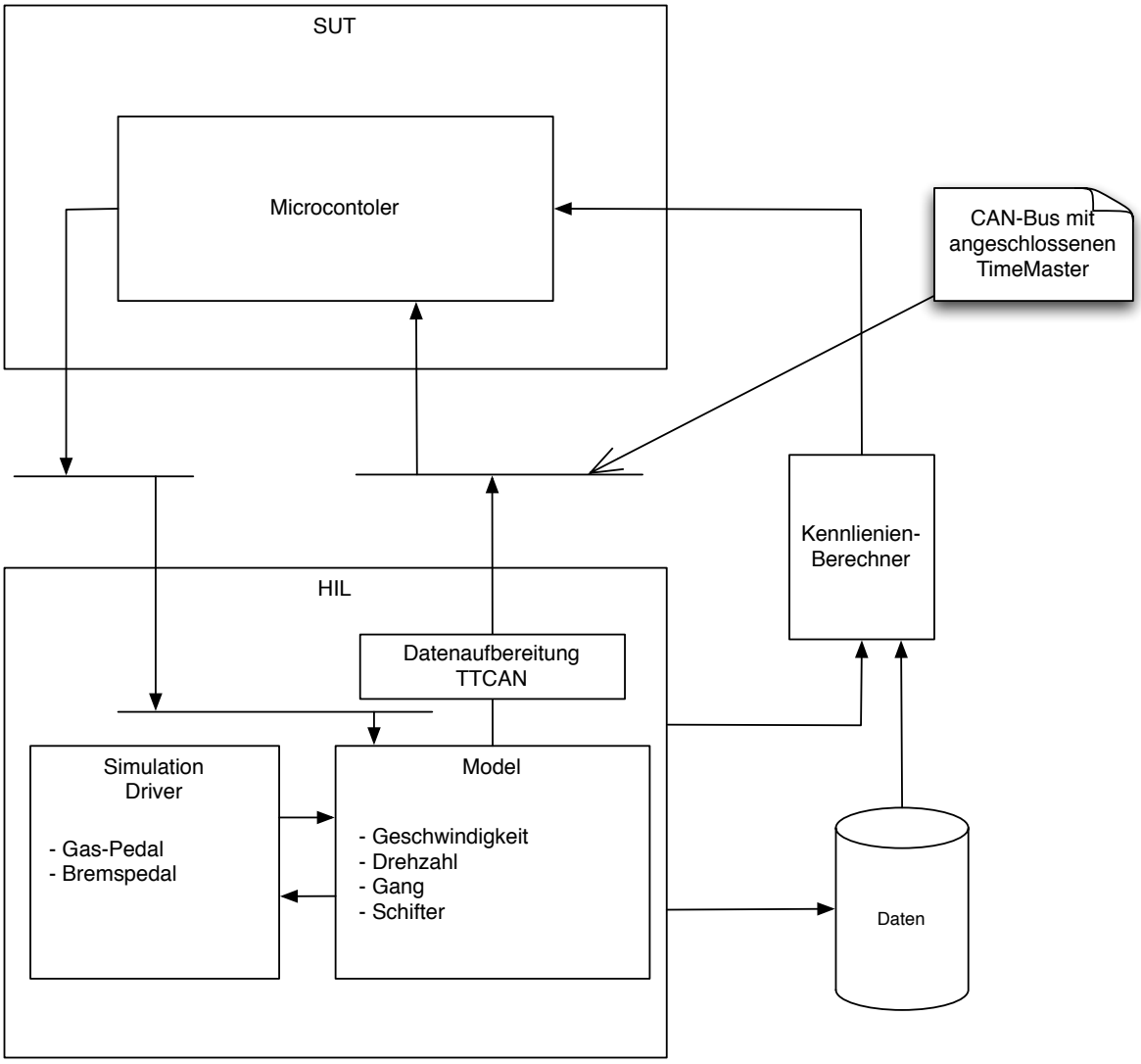

Abbildung A.7.: Versuchsaufbau einer HIL für die Kalibrierung

Glossar

DAQ Data-aquisition

Dymola Tool zur Modellierung wie z.B dynamischer physikalischer Modelle

GCU Gear-Control-Unit - ein Mikrokontroller, der für das Schalten der gänge zuständig ist

HIL Hardware-in-the-Loop

HMI Human Machine Interface

MIL Model-in-the-Loop - siehe Kapitel 1.2

PIL Processor-in-the-Loop

RCP Rapid control prototyping - siehe Kapitel 1.4

RPM Rounds per minute

SIL Software-in-the-Loop

Simulink Tool zur Modellierung wie z.B. physikalischen Modellen

Smart Transducer siehe Kapitel 2.1.2

SUT System-under-test - siehe Kapitel 1.2

TTA Time-triggered-Architektur

TTCAN Time-triggered CAN

XIL X-in-the-Loop

Index