Control-over-the-Air: Verteilte Regelungen durch cloudbasierte Fahrzeugfunktionen

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Elektrotechnik und Informationstechnik
des Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

M. Sc. Martin Sommer
(geb. Böhme)
geb. in Freiburg im Breisgau

Tag der mündlichen Prüfung: 05.02.2026

Hauptreferent: Prof. Dr.-Ing. Eric Sax
Korreferent: Prof. Dr. Andreas Oberweis

Kurzfassung

Die Automobilindustrie steht vor der Herausforderung, softwarebasierte Fahrzeugfunktionen flexibler und ressourceneffizienter zu gestalten. Mit bis zu 150 Steuergeräten in Premiumfahrzeugen und Lebenszyklen von bis zu 35 Jahren bei Stadtbussen entsteht eine kritische Diskrepanz zwischen statisch verbauter Hardware und exponentiell wachsenden Softwareanforderungen. Diese Dissertation entwickelt einen systematischen Ansatz zur Erweiterung von Fahrzeug-E/E-Architekturen um cloudbasierte Funktionen, insbesondere den Control-over-the-Air (COTA) Ansatz, bei dem regelnde Softwarekomponenten in der Cloud ausgeführt werden und die Regelschleife vom Fahrzeug über die Cloud geschlossen wird.

Die Forschungsarbeit adressiert drei zentrale Fragestellungen: die systematische Identifikation cloudfähiger Funktionen, die Auswahl optimaler Deploymentmodelle und die quantifizierbare Bewertung der Cloudverlagerungspotenziale. Zur Identifikation geeigneter Funktionen wird ein zweistufiger Bewertungsprozess entwickelt, der zunächst die Realisierbarkeit anhand von Safety- und Echtzeitanforderungen prüft und anschließend mittels eines Eignungs-Scores aus fünf Bewertungsmetriken die optimalen Kandidaten identifiziert. Die Anwendung dieses Prozesses auf eine moderne Stadtbus-E/E-Architektur identifiziert die Heizung, Lüftung, Klimatisierung (HLK)-Regelung als besonders geeigneten Kandidaten, da sie weder sicherheitskritisch (kein ASIL zugewiesen) noch echtzeitkritisch ist, aber gleichzeitig erhebliche Energieeinsparpotenziale aufweist.

Für die identifizierte Funktion wird eine multikriterielle Entscheidungsanalyse (engl. MCDA) durchgeführt, um ein passendes Cloud-Deploymentmodell

zu bestimmen. Das *Nur Cloud*-Modell erzielt ein knapp besseres Ergebnis als das *Fallback*-Modell. Beide Ansätze werden daher praktisch umgesetzt, basierend auf einer rechenintensiven modellprädiktiven Regelung, die sich besonders für die Cloud eignet. Ein Software Orchestrator im Fahrzeug koordiniert dabei die dynamische Integration von Services wie dem cloudbasierten Regler, unabhängig davon, ob sie lokal im Fahrzeug oder in externen Netzwerken bereitgestellt werden.

Die Validierung der beiden identifizierten Deploymentmodelle erfolgt auf einer Testplattform am Institut, die eine vereinfachte zentralisierte E/E-Architektur nachbildet. Die Evaluierung umfasst vier repräsentative Klimaszenarien von durchschnittlichen Bedingungen bis zu extremen Temperaturen von $-15\,^{\circ}\mathrm{C}$. Die Ergebnisse demonstrieren signifikante Verbesserungen: Der cloudbasierte MPC erreicht realistische Energieeinsparungen von $11{,}06\,\%$ gegenüber einem fahrzeuginternen PID-Regler, entsprechend $2{,}4\,\mathrm{kW\,h}$ täglich oder einer zusätzlichen Reichweite von $2{,}1\,\mathrm{km}$. Gleichzeitig wird der thermische Komfort gemäß VDV-236 Vorgaben in allen Szenarien ohne Verbindungsabbruch eingehalten, während die Vorgaben bei Verbindungsabbrüchen im -15 °C-Szenario sowohl im *Nur Cloud*- als auch im *Fallback*-Modell leicht überschritten werden. Bei der praktischen Erprobung zeigt das *Fallback*-Modell bessere Robustheit gegenüber Verbindungsabbrüchen als das in der MCDA besser bewertete *Nur Cloud*-Modell.

Die Dissertation leistet einen systematischen Beitrag zur Integration von Cloud Computing in automotive E/E-Architekturen durch die Entwicklung strukturierter Bewertungsprozesse, methodischer Deploymentmodell-Auswahl und den praktischen Nachweis messbarer Potenziale. Die Ergebnisse zeigen, dass der COTA-Ansatz technisch realisierbar ist und klare Vorteile in Energieeffizienz, Wartbarkeit und Skalierbarkeit bietet. Fallback-Strategien im Fahrzeug sind kurzfristig zwar sinnvoll, könnten mit zukünftigen drahtlosen Kommunikationstechnologien jedoch entbehrlich werden.

Abstract

The automotive industry faces the challenge of making software-based vehicle functions more flexible and resource-efficient. With up to 150 electronic control units (ECUs) in premium vehicles and life cycles of up to 35 years for city buses, there is a critical discrepancy between statically installed hardware and exponentially growing software requirements. This dissertation develops a systematic approach to extending vehicle E/E architectures with cloud-based functions, in particular the Control-over-the-Air (COTA) approach, in which control software components are executed in the cloud and the control loop is closed from the vehicle via the cloud.

The research addresses three central questions: the systematic identification of cloud-enabled functions, the selection of optimal deployment models, and the quantifiable evaluation of cloud migration potential. To identify suitable functions, a two-stage evaluation process is developed that first checks feasibility based on safety and real-time requirements and then identifies the optimal candidates using a suitability score based on five evaluation metrics. Applying this process to a modern city bus E/E architecture identifies heating, ventilation, and air conditioning (HVAC) control as a particularly suitable candidate, as it is neither safety-critical (no ASIL assigned) nor real-time-critical, but at the same time offers considerable energy-saving potential.

A multi-criteria decision analysis (MCDA) is performed for the identified function to determine a suitable cloud deployment model. The *Only Cloud* model achieves a slightly better result than the *Fallback* model. Both approaches are therefore implemented in practice, based on a computationally intensive model predictive control system that is particularly suitable for the

cloud. A software orchestrator in the vehicle coordinates the dynamic integration of services such as the cloud-based controller, regardless of whether they are provided locally in the vehicle or in external networks.

The two identified deployment models are validated on a test platform at the institute that replicates a simplified centralized E/E architecture. The evaluation covers four representative climate scenarios ranging from average conditions to extreme temperatures of -15 degrees Celsius. The results demonstrate significant improvements: The cloud-based MPC achieves realistic energy savings of 11.06 percent compared to an in-vehicle PID controller, corresponding to 2.4 kilowatt hours per day or an additional range of 2.1 kilometers. At the same time, thermal comfort is maintained in accordance with VDV-236 specifications in all scenarios without connection interruptions, while the specifications are slightly exceeded in the -15 °C scenario in both the *Only Cloud* and *Fallback* models in the event of connection interruptions. In practical testing, the *Fallback* model shows better robustness against connection interruptions than the *Only Cloud* model, which was rated higher in the MCDA.

The dissertation makes a systematic contribution to the integration of cloud computing into automotive E/E architectures through the development of structured evaluation processes, methodical deployment model selection, and practical verification of measurable potential. The results show that the COTA approach is technically feasible and offers clear advantages in energy efficiency, maintainability, and scalability. Fallback strategies in vehicles make sense in the short term, but could become unnecessary with future wireless communication technologies.

Danksagung

Diese Dissertation wäre ohne die Unterstützung vieler Menschen nicht möglich gewesen.

An erster Stelle möchte ich meinem Doktorvater, Herrn Prof. Dr.-Ing. Eric Sax, für seine hervorragende Betreuung, seine fachliche Unterstützung und die zahlreichen wertvollen Impulse während der gesamten Entstehung dieser Dissertation danken. Mein besonderer Dank gilt ebenso Herrn Prof. Dr. Oberweis für die freundliche Übernahme des Korreferats.

Meinen Kolleginnen und Kollegen am Institut für Technik der Informationsverarbeitung (ITIV) danke ich aufrichtig für die vertrauensvolle Zusammenarbeit und die vielen konstruktiven Gespräche, die wesentlich zum Gelingen dieser Arbeit beigetragen und meine Zeit am ITIV persönlich wie fachlich sehr bereichert haben.

Von ganzem Herzen danke ich meiner Familie, insbesondere meiner Frau Simone, für ihre Geduld, ihre stetige Unterstützung und ihren Zuspruch, die mich während der gesamten Promotionszeit getragen und bestärkt haben. Ohne euch wäre diese Dissertation nicht möglich gewesen.

Meiner Tochter Rosa in Liebe gewidmet.

Offenburg, im Februar 2026

Martin Sommer

Hinweise

Erklärung zu generativer KI

Bei der Erstellung dieser Arbeit wurde generative KI verwendet. Daher wird eine Erklärung zu generativer KI gemäß den Regeln der CEUR-WS-Richtlinie zu KI-unterstützenden Tools[1] in ihrer aktuellen Fassung (2025) abgegeben: Bei der Erstellung dieser Arbeit habe ich LanguageTool verwendet, um: Grammatik und Rechtschreibung zu überprüfen. Zusätzlich wurde ChatGPT verwendet für: Grammatik- und Rechtschreibprüfung, Formatierungshilfe, Paraphrasierung und Umformulierung. Nach der Verwendung dieser Tools habe ich den Inhalt überprüft und bei Bedarf überarbeitet und übernehme die volle Verantwortung für den Inhalt der Veröffentlichung.

Gender-Hinweis

In dieser Arbeit wird aus Gründen der Lesbarkeit bei Personenbezeichnungen die männliche Form verwendet. Sie schließt ausdrücklich alle Geschlechter ein.

[1] `https://ceur-ws.org/GenAI/Policy.html`

Inhaltsverzeichnis

1 Einleitung

Der Automobilsektor, der traditionell vom Maschinenbau geprägt ist, wird zunehmend durch Themen der Informationstechnologie vorangetrieben. Die Kunden haben sich an Produkte gewöhnt, deren Funktionalitäten sich im Laufe ihres Lebenszyklus ändern können. Neue Feature und Featureaktualisierungen müssen dem Kunden schnell zur Verfügung gestellt werden. Diese softwarebasierten Feature zeigen sich in den Trends der Automobilindustrie, die sich zu den folgenden Kernthemen zusammenfassen lassen:

Elektrifizierung: Die Elektromobilität wird nach allem, was wir heute wissen, kurz- bis mittelfristig als sinnvollste technologische Antwort betrachtet, um die externen Auswirkungen des Automobils, insbesondere Luft- und Lärmemissionen, zu minimieren [23]. Neben den Anforderungen an Fahrzeuge mit herkömmlichen Verbrennungsmotoren ist die Energieeffizienz für elektrische Fahrzeuge von entscheidender Bedeutung, um eine möglichst hohe Reichweite zu erzielen. Eine der Möglichkeiten zur Reduktion des Energieverbrauchs ist es, nicht benötigte Funktionen und Steuergeräte in einen stromsparenden Standby-Modus zu versetzen und durch Zentralisierung von Features die Gesamtzahl der Steuergeräte zu reduzieren. Ein weiterer in dieser Dissertation betrachteter Ansatz zur Verbesserung der Energieeffizienz ist die Entwicklung cloudbasierter Fahrzeugfunktionen.

Automatisiertes Fahren: Das hochautomatisierte Fahren erfordert eine schnelle Verarbeitung von umfangreichen Sensordaten (z.B. Radar und Kamerabilder) und den Austausch dieser Daten zwischen

verschiedenen Teilsystemen. Die schnelle Verarbeitung setzt eine erhebliche Rechenleistung der verbauten Hardware und Bandbreite der fahrzeuginternen Kommunikation voraus. Bereits in aktuellen Fahrzeugen werden technisch immer ausgereiftere Fahrerassistenzsysteme eingesetzt.

Vernetztes Fahrzeug: Die Vernetzung der Fahrzeuge stellt einen Weg dar, um in einem starken Wettbewerbsgefüge neue Akzente und Anreize für die Kunden zu setzen. Beim sogenannten Vehicle-2-X wird das Fahrzeug mit anderen Fahrzeugen oder mit Infrastruktur (Ampelanlagen etc.) vernetzt. Bereits flächendeckend verfügbare Techniken wie das Mobilfunknetz werden zunehmend für eine Anbindung des Fahrzeugs an die Cloud verwendet. Dadurch sind Hersteller in der Lage, kontinuierlich Daten aus dem Fahrzeug zu sammeln und in der Cloud aufzuarbeiten. Diese Daten können herangezogen werden, um einzelne Features zu verbessern und in der Cloud bereitzustellen oder neu auf die Fahrzeuge zu verteilen. Das Fahrzeug lässt sich zudem in ein digitales Ökosystem integrieren, in dem Feature für einen kurzen Zeitraum einzeln gebucht werden können [69].

Diese Trends sind einerseits durch politische Initiativen getrieben, etwa die Elektrifizierung als Folge der strengeren Abgasgesetzgebungen, andererseits leiten sie sich aus neuen technologischen Innovationen sowie aus Kundenwünschen ab [103]. Die Elektrik/Elektronik (E/E)-Architektur der Fahrzeuge muss diesen Trends gerecht werden und eine hohe Flexibilität bezüglich der Implementierung neuer, innovativer Features über den Lebenszyklus ermöglichen. Zukünftige E/E-Architekturen gehen daher zunehmend weg von verteilten Steuergeräten und hin zur logischen Zentralisierung auf zentralen Steuergeräten oder Fahrzeugcomputern. Die logische Zentralisierung von Funktionen in wenigen leistungsstarken Steuergeräten in Kombination mit einer hohen Vernetzung der Fahrzeuge mit der Cloud macht eine physische Verteilung der E/E-Architektur denkbar, wie in der zentralisierten Architektur der Zukunft aus Abbildung 1.1 dargestellt. Funktionen, die bisher fest

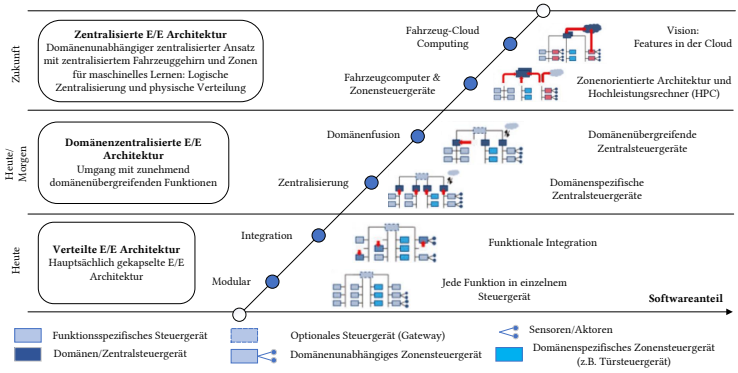


Abbildung 1.1: Evolution der E/E-Architekturen im Automobil (angelehnt an [103])

in einem Steuergerät im Fahrzeug ausgeführt wurden, werden durch die Vernetzung des Fahrzeugs mit der Cloud in diese verlagert. Im Rahmen der vorliegenden Dissertation wird eine solche Verlagerung anhand des Beispiels einer Klimaregelung eines elektrischen Stadtbusses prototypisch umgesetzt.

1.1 Hintergrund und Motivation

Ein modernes Premiumfahrzeug verfügt über 150 Steuergeräte [124], die das Fahrzeugverhalten überwachen und zahlreiche Komfort- sowie Fahrerassistenzfunktionen ermöglichen.

Diese Steuergeräte werden an den Produktionsbändern der Automobilhersteller in Fahrzeuge montiert, deren Lebenszyklen im Fall von Omnibussen bei bis zu 35 Jahren liegen (s. Abbildung 1.2). Das bedeutet, dass statisch

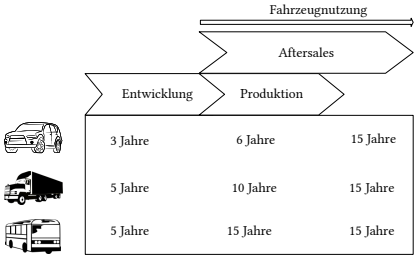


Abbildung 1.2: Produktlebenszyklen verschiedener Fahrzeugtypen in Jahren [102]

verbaute Rechen- und Speicherressourcen in Fahrzeugen die Anforderungen zukünftiger Software über mehrere Jahrzehnte erfüllen müssen. Betrachtet man den rasant steigenden Technologieumfang im Fahrzeug (s. Abbildung 1.3), der zunehmend durch Software-Features angeführt wird, so ist davon auszugehen, dass diese Annahme nicht haltbar ist. Stattdessen werden Over-The-Air (OTA)-Software-Updates, wie auch im Smartphone-Bereich, ab einer definierten Version gar nicht mehr oder in reduzierter Form angeboten, da die Kompatibilität nicht gegeben ist. Ein Ansatz, Hardware und Software über den Lebenszyklus gleichbleibend aktuell zu halten, ist die (re-)konfigurierbare Fahrzeugarchitektur [111], bei der Hardwarekomponenten über den Lebenszyklus ersetzt oder hinzugefügt werden können.

Eine andere Möglichkeit, Hardware und Software über den Lebenszyklus aktuell zu halten, stellt die Erweiterung der E/E-Architektur in die Cloud dar. Diese Erweiterung beinhaltet die Nutzung der skalierbaren Ressourcen der Cloud zur Ausführung von Funktionen, deren Output wiederum im Fahrzeug genutzt werden kann.

Im Kontext von Control-Over-The-Air (COTA) (s. Tabelle 5.1) handelt es sich um regelnde Funktionen, bei denen die Regelschleife vom Fahrzeug über die Cloud und zurück zum Fahrzeug geschlossen wird. Die Umsetzung dieser Idee führt zu folgenden Herausforderungen:

HF-1 Dynamische Integration von lose gekoppelten Funktionen

HF-2 Kontextadaptive und robuste Regelung trotz intermittierender Konnektivität oder Serviceausfällen

HF-3 Lebenszyklusfähige und wartbare Regelalgorithmen

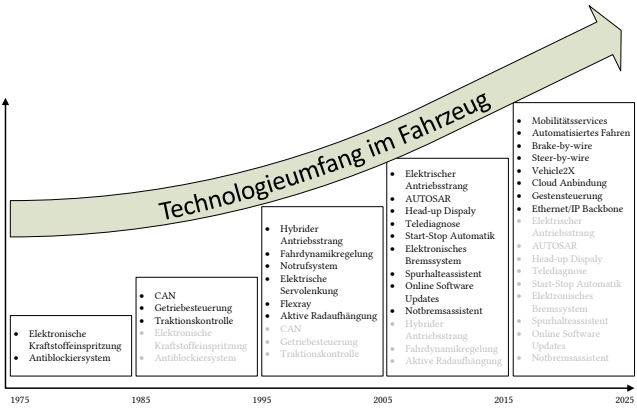


Abbildung 1.3: Zeitliche Entwicklung des Technologieumfangs im Automobil (angelehnt an [110])

1.2 Cloudbasierte Fahrzeugfunktionen

Ziel dieser Dissertation ist es, darzulegen, wie die bestehende E/E-Architektur eines Fahrzeugs durch den Einsatz cloudbasierter Fahrzeugfunktionen systematisch erweitert werden kann. Die Integration einer dynamischen Cloudumgebung in die traditionell statische Fahrzeugarchitektur führt zu einer grundlegenden Veränderung der Architekturentwicklung – insbesondere im Hinblick auf die Funktionsverteilung. Die Möglichkeit, eine Funktion (vgl. Definition 1.5) nicht nur im Fahrzeug, sondern auch in der Cloud bereitzustellen, eröffnet neue Freiheitsgrade im Entwurf. Dies betrifft sowohl die Migration bestehender Funktionen in die Cloud als auch die Realisierung neuartiger Funktionen, die aufgrund begrenzter Rechen-, Speicher- oder Energieressourcen im Fahrzeug bislang nicht umsetzbar waren.

Zu diesen neuartigen Funktionen zählen insbesondere regelungstechnische Komponenten, deren Stellgrößen direkt zur Optimierung fahrzeuginterner Funktionen beitragen können, beispielsweise im Bereich des Energiemanagements, der Klimatisierung oder der Fahrdynamik. Lernende Funktionen, die in der Cloud betrieben werden, können zudem umfangreiche Daten aus Fahrzeugflotten nutzen, um ihre Strategien kontinuierlich zu verbessern und so eine weitere Steigerung von Effizienz, Komfort und Sicherheit zu erzielen.

Ein weiterer zentraler Vorteil cloudbasierter Funktionen liegt in deren Lebenszyklusflexibilität. Sie können unabhängig von der im Fahrzeug fest verbauten Hardware dynamisch bereitgestellt, aktualisiert oder ersetzt werden. Im Gegensatz zu OTA-Update-Prozessen, bei denen Softwarepakete explizit an die Fahrzeugflotte verteilt werden müssen, erfolgt die Aktualisierung cloudbasierter Funktionen zentral. Fahrzeuge, die mit der Cloud verbunden sind, greifen automatisch auf die jeweils aktuelle Version zu. Dies ermöglicht eine erhebliche Reduktion der Update-Komplexität, senkt Betriebs- und Wartungskosten und minimiert potenzielle Ausfallzeiten infolge softwareseitiger Aktualisierungen.

1.3 Allgemeine Begriffsdefinitionen

Diese Dissertation verwendet die Begriffe System, Komponente, Funktion und Feature gemäß der in Abbildung 1.4 dargestellten ontologischen Beziehung und den zugehörigen Definitionen. Demnach kann ein System aus null oder mehr Subsystemen bestehen und setzt sich mindestens aus einer Komponente zusammen. Eine Komponente als generalisierte Form von Hard- und Softwarekomponenten stellt einen funktional sowie logisch abgegrenzten Teil eines Systems dar. Sie kann dabei aus einer oder mehreren Funktionen bestehen. Funktionen sind hier als logische Softwarekomponenten zu verstehen, die Eingaben verarbeiten und Ausgaben erzeugen; sie können dabei optional in Komponenten vorhanden sein. Aus Sicht des Anwenders werden Funktionen in Form von Features wahrgenommen, die die für den Kunden erlebbaren Eigenschaften des Systems darstellen.

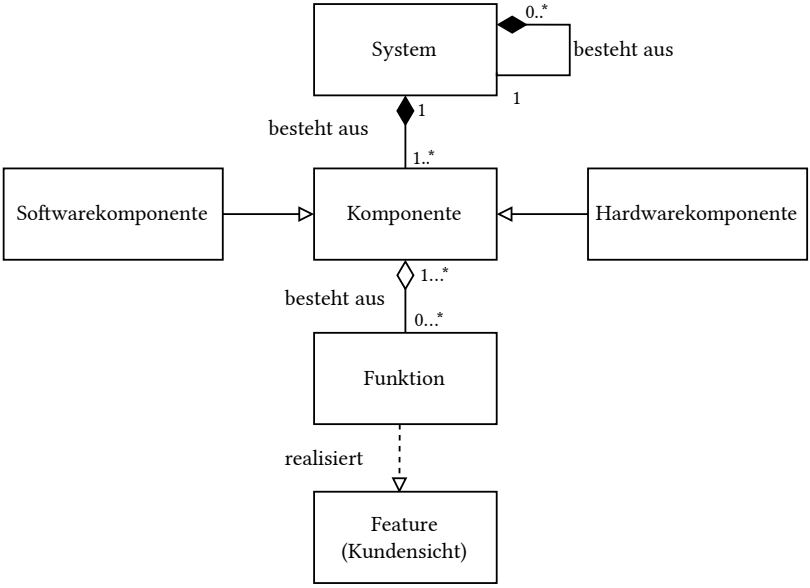


Abbildung 1.4: Ontologie der Begriffe System, Komponente, Funktion und Feature

Definition 1.1 System: Die ISO/IEC/IEEE 24765:2017 „Systems and software engineering — Vocabulary" [57] definiert ein System als eine Kombination interagierender Elemente, die so organisiert sind, dass sie einen oder mehrere definierte Zwecke erfüllen. Im Kontext dieser Dissertation kann der Begriff System, abhängig vom jeweiligen Betrachtungsrahmen, unterschiedliche Bedeutungen annehmen (vgl. [101]):

- das Gesamtfahrzeug, bestehend aus verschiedenen Komponenten wie Elektrik und Antrieb,
- ein Subsystem innerhalb des Fahrzeugs, bspw. das Heizung, Lüftung, Klimatisierung (HLK)-System in Kapitel 2.4,
- eine einzelne Fahrzeugkomponente, wie z. B. ein Verdichter oder ein Steuergerät.

Definition 1.2 Komponente: Ein System besteht aus Komponenten, wobei eine Komponente einen funktional und logisch abgegrenzten Teil des Systems darstellt. Dabei kann es sich sowohl um HWCs (s. Definition 1.3) als auch um SWCs (s. Definition 1.4) handeln, die selbst weiter in Subkomponenten unterteilt werden können. (nach [57])

Definition 1.3 Hardwarekomponente (HWC): Eine HWC ist eine technische Einheit, die eine Funktion durch in Hardware implementierte Logik ausführt:

- Festgelegte Logik (Application-specific integrated circuit (ASIC), dedizierte Schaltungen)
- Konfigurierbare Logik (Field-Programmable Gate Array (FPGA))
- Prozessoren und Mikrocontroller (als Ausführungsplattform)

Die Funktionalität wird direkt durch die Hardwarearchitektur bereitgestellt.

Definition 1.4 Softwarekomponente: Eine Softwarekomponente (SWC) ist eine technische Einheit, die eine Funktion durch ausführbaren Code realisiert. Dieser Code wird auf einer HWC (Prozessor, Mikrocontroller) interpretiert und ausgeführt. Die SWC bestimmt das funktionale Verhalten durch programmierbaren Code.

Definition 1.5 Funktion: Im Kontext dieser Dissertation ist eine Funktion eine logisch abgegrenzte Ausführungsanweisung, die Eingaben entgegennimmt, verarbeitet und Ausgaben zurückliefert. Sie wird in einer Softwarekomponente realisiert. (nach [58])

Definition 1.6 Feature: Features, auch Kundenfunktionen, sind nicht mit Funktionen im Sinne von Definition 1.5 zu verwechseln. Kundenfunktionen stellen abstrakte funktionale Charakteristika eines Systems dar, die für Anwender und andere Stakeholder greifbar sind. (nach [56])

1.4 Forschungsfragen

Die Integration von Cloud-Technologien in die Fahrzeugarchitektur stellt einen vielversprechenden Ansatz dar, um die Softwareentwicklung flexibler zu gestalten, neue Features schneller bereitzustellen und die Rechenleistung bedarfsgerecht zu skalieren. Gleichzeitig wirft diese Entwicklung neue Fragen auf, da bestehende E/E-Architekturen in Fahrzeugen primär für lokale Steuergeräte optimiert sind und eine Migration in die Cloud umfassend analysiert werden muss.

Daher ist zunächst zu klären, wie in Fahrzeug-E/E-Architekturen potenziell cloudfähige Funktionen systematisch identifiziert und deren Eignung für eine Cloud-Migration bewertet werden können. Aufbauend auf einer solchen Identifikation stellt sich die Frage, welche Deploymentmodelle (s. Definition 5.1) für cloudbasierte Funktionen überhaupt denkbar sind und wie sich das am besten geeignete Modell finden lässt. Schließlich gilt es nachvollziehbar zu machen, welche Vorteile sich daraus ergeben und wie diese messbar belegt werden können.

Forschungsfrage 1: Identifikation cloudfähiger Funktionen

Wie können in Fahrzeug-E/E-Architekturen potenziell cloudfähige Funktionen systematisch identifiziert und deren Eignung für eine Cloud-Migration bewertet werden?

Forschungsfrage 2: Deploymentmodelle

Welche Deploymentmodelle cloudbasierter Fahrzeugfunktionen sind vorstellbar und wie lässt sich das am besten geeignete finden?

Forschungsfrage 3: Potenziale der Cloudverlagerung

Welche Vorteile ergeben sich aus der Cloudverlagerung und wie lassen sich diese messbar nachvollziehen?

2 Technische und begriffliche Grundlagen

2.1 Regelungstechnik

2.1.1 Modellprädiktive Regelung

Eine modellprädiktive Regelung (engl. Model Predictive Control (MPC)) nutzt ein Anlagenmodell, um das zukünftige Verhalten der realen Anlage bzw. des Systems vorherzusagen.

Das Modell bildet die Systemdynamik möglichst realitätsnah ab und kann dabei in verschiedenen Formen vorliegen, wie z. B. als Zustandsraumdarstellung, Übertragungsfunktion oder auf Basis der Sprungantwort. Mithilfe des Modells wird aus dem aktuellen Zustand $x(k)$ sowie einer gegebenen Referenz ein prädizierter Output berechnet, der mit dem gewünschten Sollverlauf der Ausgangsgröße $y(k)$ verglichen wird. Aus diesem Vergleich ergibt sich ein prädizierter Fehler.

Ein integrierter Optimierer berechnet anschließend auf Basis dieses Fehlers, einer zugehörigen Kostenfunktion sowie definierter Beschränkungen die optimale zukünftige Stellgrößenfolge (bzw. prädizierten Input) $u(k)$, welche das Regelziel bestmöglich erfüllt.

Obwohl der Optimierer eine gesamte Stellgrößenfolge berechnet, wird in der Praxis jeweils nur der erste Eintrag an die reale Anlage übergeben. Der

daraus resultierende neue Systemzustand $x(k)$ wird zurückgeführt und im nächsten Zeitschritt zur erneuten Optimierung verwendet – ein Prinzip, das als Receding Horizon bezeichnet wird.

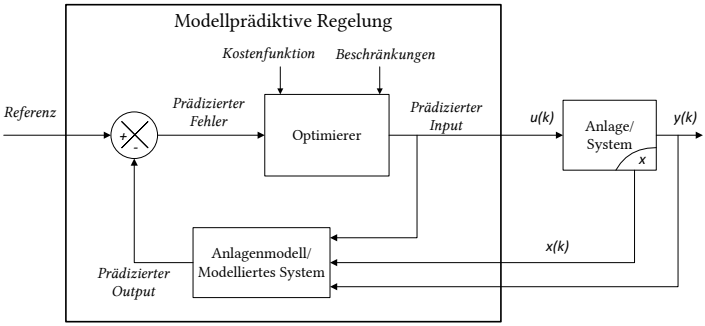


Abbildung 2.1: Struktur einer modellprädiktiven Regelung

Dieser Vorgang wird stetig wiederholt. Die Struktur eines Systems mit modellprädiktiver Regelung ist in Abbildung 2.1 zu sehen. Da angenommen wird, dass die Zustände der Anlage vollständig beobachtbar sind, ist in dieser Abbildung kein Zustandsschätzer in den Regelkreis integriert.

In Abbildung 2.2 ist die Strategie einer modellprädiktiven Regelung schematisch dargestellt. Der Ablauf kann durch folgende Schritte in zeitdiskreter Form beschrieben werden [27]:

- **1. Schritt:**
  Für den Zeitschritt k wird die zukünftige Stellgrößenfolge über einen endlichen Horizont N_p (Prädiktionshorizont) bestimmt, die zum optimalen Verlauf der Ausgangsgröße führt (s. Abbildung 2.2a). Die Vorhersage erfolgt anhand des Modells und des aktuellen Zustands des realen Systems. Der Optimierungsvorgang zur Bestimmung der optimalen Stellgrößenfolge wird durch Minimierung der Kostenfunktion umgesetzt.

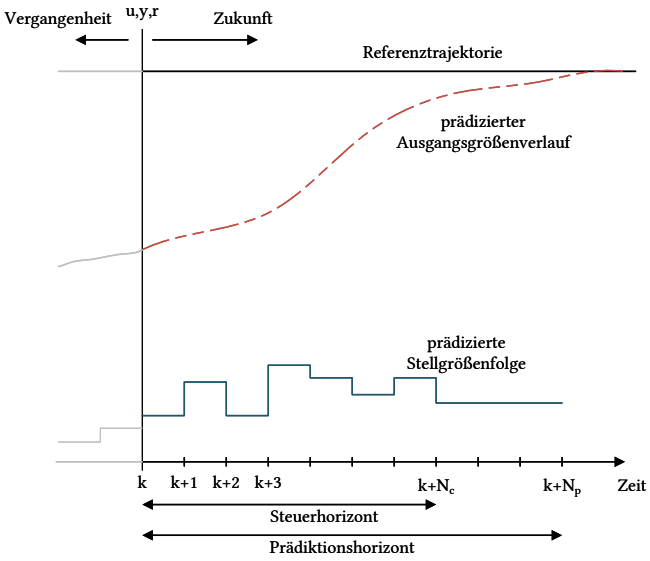
- **2. Schritt:**
  Der erste Wert der für den Zeitschritt k ermittelten zukünftigen Stellgrößenfolge wird an das reale System übermittelt und angewendet. Die anderen Werte der vorhergesagten Stellgrößenfolge werden verworfen. Es ergibt sich ein neuer Zustand des Systems. Die tatsächliche Ausgangsgröße des Systems kann dabei von der prädizierten Ausgangsgröße abweichen, wenn das verwendete Modell die Dynamik des Systems nicht exakt widerspiegelt oder Störungen auftreten, die im Modell nicht berücksichtigt werden können (s. Abbildung 2.2b).

- **3. Schritt:**
  Im nächsten Zeitschritt $k + 1$ wird für den verschobenen Prädiktionshorizont und mit dem neuen Zustand des Systems eine neue Stellgrößenfolge vorhergesagt (s. Abbildung 2.2c). Dieses Vorgehen wird für die folgenden Zeitschritte wiederholt.

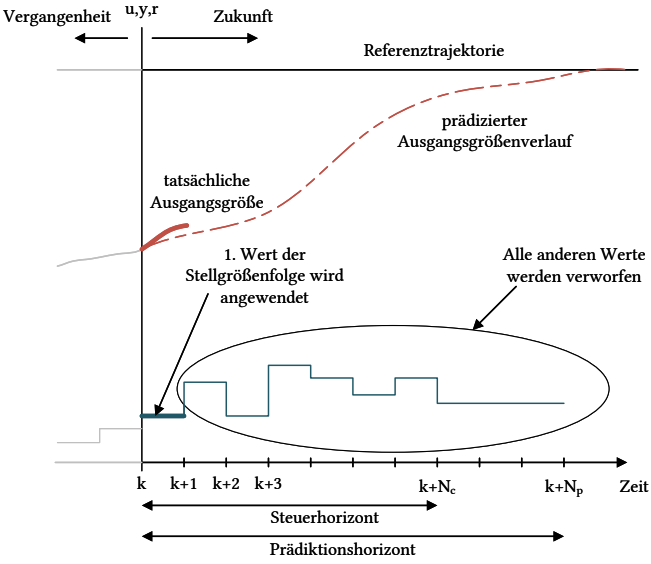
a) Kostenfunktion Bei einer MPC wird zu jedem Zeitschritt eine optimale zukünftige Stellgrößenfolge gesucht. Dazu erfolgt ein Optimierungsvorgang innerhalb des Reglers. Die Optimierung basiert dabei auf der Minimierung einer Kostenfunktion. Eine solche Kostenfunktion berücksichtigt typischerweise die Differenz der Ausgangsgröße von einer Referenz sowie die Änderungen der Stellgröße [95]:

$$J = \sum_{j=1}^{N_\mathrm{p}} q \cdot [y(k+j) - y_\mathrm{ref}(k+j)]^2 + \sum_{j=1}^{N_\mathrm{c}} r \cdot [\Delta u(k+j-1)]^2 \quad (2.1)$$

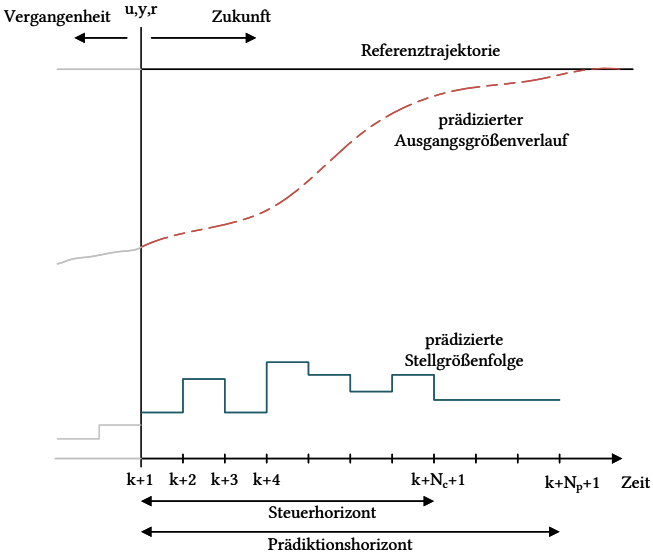
Die Faktoren q und r aus Gleichung 2.1 dienen der Gewichtung der Optimierungsziele einer Kostenfunktion (s. Abbildung 2.1). Durch diese Größen kann eingestellt werden, ob die Abweichung des Ausgangs von der Referenz oder aggressive Änderungen der Stellgrößen stärker bestraft werden sollen.


Vergangenheit
u,y,r
Zukunft
Referenztrajektorie
prädizierter
Ausgangsgrößenverlauf
prädizierte
Stellgrößenfolge
k
k+1
k+2
k+3
k+Nc
k+Np
Zeit
Steuerhorizont
Prädiktionshorizont

(a)


Vergangenheit
u,y,r
Zukunft
Referenztrajektorie
prädizierter
Ausgangsgrößenverlauf
tatsächliche
Ausgangsgröße
1. Wert der
Stellgrößenfolge wird
angewendet
Alle anderen Werte
werden verworfen
k
k+1
k+2
k+3
k+Nc
k+Np
Zeit
Steuerhorizont
Prädiktionshorizont

(b)



(c)

Abbildung 2.2: Strategie einer modellprädiktiven Regelung
a) Vorhersage der optimalen Stellgrößenfolge
b) Anwendung des ersten Wertes der Stellgrößenfolge
c) Horizont verschieben und neue Stellgrößenfolge bestimmen

b) Beschränkungen Einer der größten Vorteile einer modellprädiktiven Regelung ist die Berücksichtigung von Beschränkungen der Stellgrößen. Mechanische Grenzen von Maschinen können dadurch ebenso berücksichtigt werden wie konstruktive oder sicherheitstechnische Begrenzungen. Beispielsweise kann das maximale Füllniveau eines Tanks oder ein einzuhaltender Druckbereich eines Behälters über Beschränkungen einkalkuliert werden. Dabei wird zwischen *harten* und *weichen* Beschränkungen unterschieden. Während harte Beschränkungen zu jeder Zeit eingehalten werden müssen, können weiche Beschränkungen kurzzeitig verletzt werden.

c) Prädiktionshorizont Bei der MPC wird in jedem Zeitschritt die optimale zukünftige Stellgrößenfolge gesucht, die innerhalb eines bestimmten Zeitfensters die Abweichung der Ausgangsgröße von einer gewünschten Referenz minimiert. Dieses Zeitfenster, für das die optimale Stellgrößenfolge vorhergesagt wird, ist der Prädiktionshorizont N_p. Die Wahl des Prädiktionshorizonts muss den Gegebenheiten des zu regelnden System angepasst werden. Er sollte länger als die signifikanten Dynamiken des Prozesses sein, um rechtzeitig auf mögliche Störungen reagieren zu können. Allerdings sollte auch berücksichtigt werden, dass der Rechenaufwand des Reglers mit zunehmendem Prädiktionshorizont zunimmt.

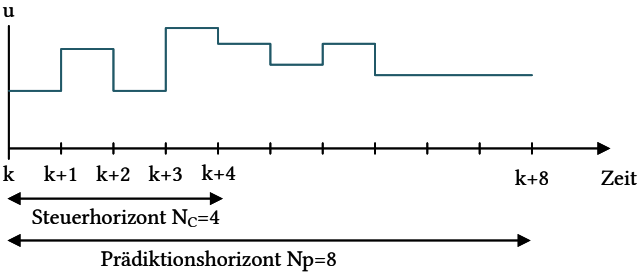


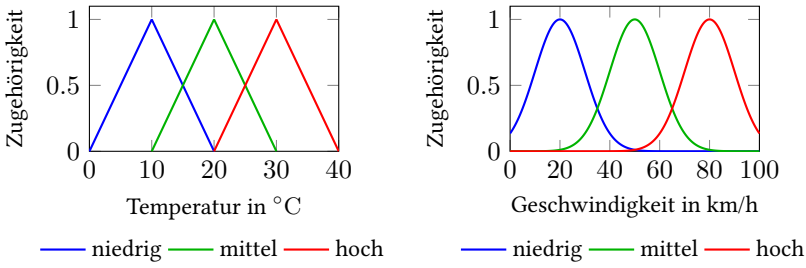
Abbildung 2.3: Beispiel einer prädizierten Stellgrößenfolge mit $N_p = 8$ und $N_c = 4$

d) Steuerhorizont Der Steuerhorizont N_c gibt an, bis zu welchem Zeitschritt des Prädiktionshorizonts die Stellgröße verschiedene Werte annimmt. Über den Steuerhorizont kann somit eingestellt werden, ab welchem Zeitschritt die Stellgrößenfolge konstante Werte annehmen soll. In Abbildung 2.3 ist beispielhaft eine prädizierte Stellgrößenfolge mit dem Prädiktionshorizont $N_\mathrm{p} = 8$ und Steuerhorizont $N_\mathrm{c} = 4$ dargestellt. Ab dem Zeitpunkt $k + N_\mathrm{c} = k + 4$ ist die Stellgrößenfolge bis $k + N_\mathrm{p}$ konstant. Die Wahl des Steuerhorizonts beeinflusst den Rechenaufwand des Reglers. Entspricht der Steuerhorizont dem Prädiktionshorizont, muss für jeden vorhergesagten Zeitschritt eine Optimierung durchgeführt werden. Bei kürzeren Steuerhorizonten

entfällt die Optimierung für die Zeitschritte zwischen Steuer- und Prädiktionshorizont, da in diesem Bereich die vorhergesagten Stellgrößenwerte konstant gehalten werden.

2.1.2 Fuzzy-Regler

Ein Fuzzy-Regler arbeitet mit unscharfen Zugehörigkeitsfunktionen, die zusammen eine Fuzzy-Menge bilden, um Ein- und Ausgangsgrößen mittels linguistisch formulierter Regeln zu verknüpfen. Dabei werden konkrete Eingangssignale durch definierte Funktionen in unscharfe linguistische Begriffe überführt (Fuzzifizierung). So kann beispielsweise eine exakte Temperaturangabe (in °C) durch Begriffe wie „niedrig“, „mittel“ oder „hoch“ beschrieben werden. Diese Abbildung eines scharfen auf einen unscharfen Eingangswert erfolgt über Zugehörigkeitsfunktionen, die für jeden Eingangswert einen Zugehörigkeitsgrad (zwischen 0 und 1) zu den jeweiligen Begriffen angeben. Die Zugehörigkeitsfunktionen der einzelnen Begriffe dürfen sich dabei überlappen (s. Abbildung 2.4).



(a) Dreieckszugehörigkeitsfunktion

(b) Gaußsche Zugehörigkeitsfunktion

Abbildung 2.4: Fuzzifizierung der Eingangsgrößen Temperatur und Geschwindigkeit mittels Dreiecks- und Gaußscher Zugehörigkeitsfunktionen

Für die so fuzzifizierten Eingangswerte werden anschließend mithilfe von Regeln der Form

WENN <*Bedingung*>, DANN <*Anweisung*>

die zugehörigen unscharfen Ausgangswerte bestimmt.

Aus diesen unscharfen Ausgangswerten wird anschließend durch verschiedene Verfahren (bspw. Flächenschwerpunkt-Bestimmung) ein scharfer Ausgangswert berechnet (Defuzzifizierung) (s. Abbildung 2.5) [5]. Die Regelbasis basiert dabei häufig auf Expertenwissen, es existieren jedoch auch Ansätze zur automatischen Generierung bzw. adaptiven Anpassung, etwa durch den Einsatz künstlicher neuronaler Netze.

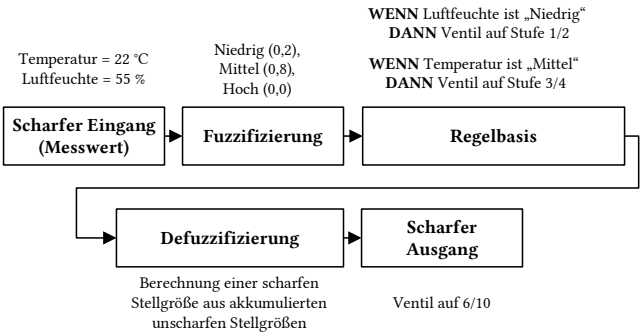


Abbildung 2.5: Ablauf eines Fuzzy-Reglers

2.1.3 PID-Regler

Der PID-Regler zeichnet sich durch Robustheit, Intuitivität und Einfachheit aus und wird auch als Universalregler bezeichnet. Ein PID-Regler besteht aus einem Proportionalteil (P-Glied), einem Integralteil (I-Glied) sowie einem Ableitungsteil (D-Glied). Dabei sorgt das P-Glied für direkte Reaktion auf Regelabweichungen und der I-Anteil für Elimination von stationären Fehlern. Der D-Anteil reagiert auf Änderungen der Regelabweichung:

$$u(t) = K_p \cdot e(t) + K_I \cdot \int_0^t e(t)\,dt + K_D \cdot \frac{d}{dt} e(t) \tag{2.2}$$

mit

u: Stellgröße

e: Regelabweichung (Sollwert – Istwert)

K: Verstärkungsfaktoren der jeweiligen Anteile

Der Hauptnachteil von PID-Reglern liegt in der Vielfalt möglicher Parameterwerte, die es auszuwählen gilt, um einen optimalen Regler zu gestalten. Oftmals wird deshalb, je nach Regelstrecke, auf einzelne Glieder verzichtet.

2.2 Steuergeräte

Elektronische Steuergeräte (fortan Electronic Control Unit (ECU)), Sollwertgeber, Sensoren und Aktoren bilden gemeinsam ein elektronisches System zur Beeinflussung des Zustands der Strecke. Dieses System wirkt innerhalb des Gesamtsystems „Fahrer – Fahrzeug – Umwelt“ häufig im Verborgenen. Besonders Steuergeräte, die ausschließlich Steuerungs- und Regelungsaufgaben übernehmen, besitzen keine direkten Benutzerschnittstellen und bleiben für die Fahrzeuginsassen unsichtbar.

Der Einfluss des Fahrers auf diese Steuergeräte erfolgt nur indirekt, etwa über Gaspedalstellung oder Lenkwinkel, sogenannte Führungsgrößen. Diese werden über Sensoren oder Sollwertgeber (als Sonderform von Sensoren zur Erfassung von Benutzerwünschen) erfasst und im Steuergerät überwacht, geregelt oder gesteuert (s. Abbildung 2.6). Aktoren wandeln die von der ECU kommenden Stellgrößen in physikalische Reaktionen auf die Strecke um. Unter der „Strecke“ versteht man den physikalischen Teil des Systems. Im Fahrzeugkontext umfasst die Strecke beispielsweise:

- das mechanische System des Antriebs,

- die Bewegung des Fahrzeugs (z. B. Beschleunigung, Bremsung),
- das Verhalten der Räder oder Dämpfer,
- die Innenraumtemperatur in einem Klimaregelungssystem.

Sensoren erfassen den sich neu ergebenden Systemzustand und geben diesen an die ECU zurück, um in diesem Fall einen Regelkreis zu schließen.

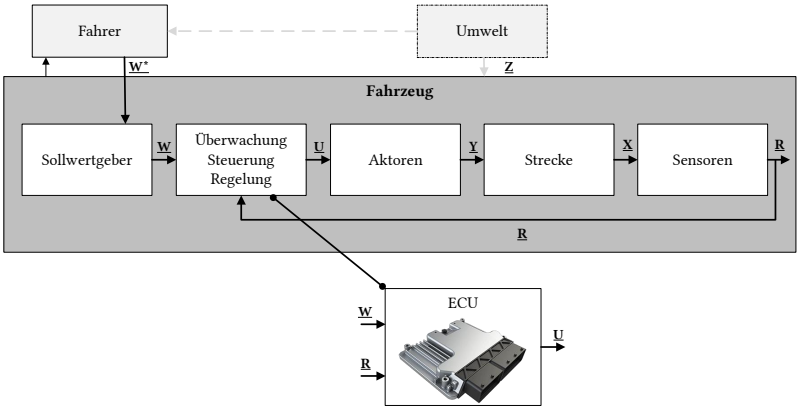


Abbildung 2.6: Steuergerät als eingebettetes System im Fahrzeug [104]

Die ECUs im Fahrzeug sind über ein Kommunikationsbussystem miteinander vernetzt. Dieses Kommunikationssystem ermöglicht den Datenaustausch zwischen den verschiedenen ECU in Echtzeit (s. Definition 2.1) und bildet somit das Rückgrat für viele elektronische Features moderner Fahrzeuge.

> **Definition 2.1 Echtzeitsystem:** Verarbeitet ein System seine Eingangs- und Zustandsgrößen innerhalb eines vorgegebenen Zeitintervalls garantiert und erzeugt in diesem Zeitintervall die zugehörigen Ausgangs- und Zustandsgrößen, so spricht man von einem Echtzeitsystem [101].

Die zunehmende Anzahl an ECUs wird in der Automobilindustrie durch den Ansatz „teile und herrsche“ realisierbar gemacht. Mittels dieses Ansatzes wird das Fahrzeug üblicherweise in die Subsysteme bzw. Domänen unterteilt (s. Abbildung 2.7):

- Antriebsstrang
- Fahrwerk
- Karosserie (mit Komfort und Passive Sicherheit)
- Infotainment
- Fahrerassistenz

Leistungskategorien von ECUs Eine generelle Aussage über die Rechenleistung und den Energieverbrauch von ECUs ist nicht realistisch. Vielmehr kann die Vielzahl an ECUs in Fahrzeugen in Kategorien eingeteilt werden. Die nachfolgenden Leistungsklassen basieren auf der Dissertation „Energiemanagement Ethernet-basierter Fahrzeugnetze“ von Balbierer [13]. Die Leistungsklasse *Hoch* wurde aufgrund aktueller leistungsfähiger Mikrocontroller für Machine Vision im Bereich der Fahrerassistenz angepasst. Dazu gehört zum Beispiel der i.MX 8Quad Max des Herstellers NXP, dessen Kennwerte hier übernommen wurden [88]. Die Leistungsklasse *Sehr hoch* orientiert sich am NVIDIA DRIVE AGX Orin [86].

Einfluss der CPU-Auslastung auf den Energieverbrauch der ECU

Die unteren zwei Leistungsklassen aus Tabelle 2.1 liegen in Bezug auf den Stromverbrauch im Bereich eines Raspberry Pi Model B[1]. Daher wird die in Abbildung 2.8 gemessene Kurve näherungsweise auf diese Leistungsklassen angewendet. Auf diese Weise kann die Leistungsaufnahme aus der Central

[1] `https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#voltages`

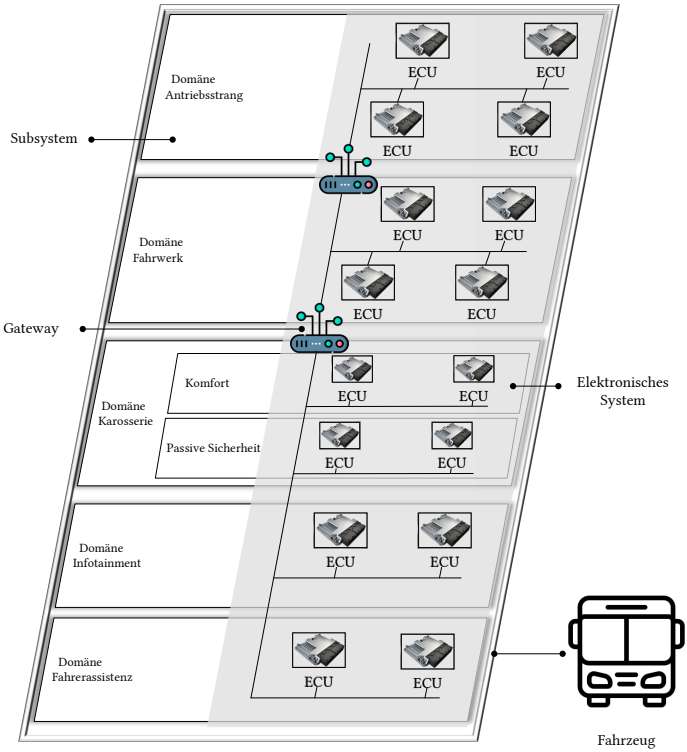


Abbildung 2.7: Zuordnung der elektronischen ECUs zu den funktionalen Domänen des Fahrzeugs [104]

Processing Unit (CPU)-Last abgeleitet werden. Um die Kurve anzunähern, muss zunächst bestimmt werden, welcher Prozentsatz der maximalen Leistung (CPU-Auslastung 100%) im Leerlauf, d.h. wenn sie nicht genutzt wird, verbraucht wird. Der Stromverbrauch im Ruhezustand entspricht etwa 89% der maximalen Last. Damit lässt sich eine allgemeine Berechnungsregel für den Stromverbrauch in Abhängigkeit von der CPU-Last δ_{CPU} aufstellen:

$$P_{\mathrm{CPU,usage}} = 0,11 \cdot P_{\max} \cdot \delta_{\mathrm{CPU}} + 0,89 \cdot P_{\max} \tag{2.3}$$

Tabelle 2.1: Leistungsklassen typischer ECU im Fahrzeug (angelehnt an [13])

Kategorisierung						
Leistungs-klasse	Einsatz-gebiet	Rechen-performanz	Mikro-controller	Peripherie	Netzwerk	Summe (50 % Effizienz)
Sehr hoch	HPC	>200 TOPS >50.000 DMIPS	40 W	8,0 W	2 W	150 W
Hoch	Fahrer-assistenz	10.000-50.000 DMIPS	15 W	4,0 W	1 W	40 W
Medium	Antriebs-strang	1.000-10.000 DMIPS	0,8 W	1,0 W	0,12 W	4 W
Gering	ESP	500-1.000 DMIPS	0,5 W	0,5 W	0,04 W	2 W

DMIPS: Dhrystone million instructions per second TOPS: Tera operations per second
ESP: Elektronisches Stabilitätsprogramm

Typischerweise werden in modernen E/E-Architekturen mehrere Funktionen auf einer ECU lokalisiert (s. Kapitel 2.3). Um nur eine Funktion zu berücksichtigen, muss daher die durchschnittliche CPU-Auslastung der im Fokus stehenden Funktion in die Formel eingesetzt werden und nicht die gesamte CPU-Auslastung der ECU. Für die zwei untersten Leistungsklassen von Mikrocontrollern ergeben sich daraus die folgenden Gleichungen:

$$\textbf{Medium Leistungsklasse: } P_{\text{Medium}} = 0,11 \cdot 1{,}6\,\text{W} \cdot \delta_{\text{Funktion}} \tag{2.4}$$

$$\textbf{Geringe Leistungsklasse: } P_{\text{Niedrig}} = 0,11 \cdot 1\,\text{W} \cdot \delta_{\text{Funktion}} \tag{2.5}$$

Für die hohe und sehr hohe Leistungsklasse liegen keine vergleichbaren Kurven der Leistungsaufnahme in Abhängigkeit von der CPU bzw. Graphics Processing Unit (GPU)-Auslastung vor, es wird aber ein linearer Zusammenhang angenommen. Um sich der Formeln anzunähern, muss die Leistungsaufnahme eines Low-Power-Zustandes ermittelt werden. Hierfür wird der Leistungsbedarf des NXP i.MX 8QuadMax [88] und des NVIDIA Drive AGX

Orin [87] im Low-Power-Zustand auf Basis des Datenblatts ermittelt. Es ergeben sich folgende Formeln für die Leistungsklassen:

$$\textbf{Hohe Leitungsklasse: } P_{\text{Sehr hoch}} = 0,8 \cdot 30\,\text{W} \cdot \delta_{\text{Funktion}} \tag{2.6}$$

$$\textbf{Sehr hohe Leitungsklasse: } P_{\text{Sehr hoch}} = 1,1 \cdot 80\,\text{W} \cdot \delta_{\text{Funktion}} \tag{2.7}$$

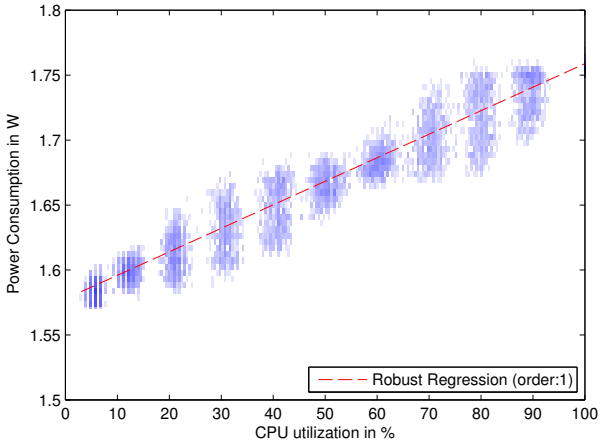


Abbildung 2.8: Energieverbrauch in Abhängigkeit von der CPU-Last eines Raspberry Pi Model B [62]

2.3 Die Elektrik/Elektronik (E/E)-Architektur

2.3.1 Historische Entwicklung der E/E-Architektur

Verteilte Architektur

Die konventionelle Herangehensweise setzt auf funktional-verteilte Systeme, bei denen jede einzelne Funktion auf eine eigene ECU partitioniert wird. Um Informationen miteinander auszutauschen, sind die Komponenten beispielsweise über Controller Area Network (CAN)- und Local Interconnect Network (LIN)-Bussysteme miteinander vernetzt. Beide Kommunikationssysteme verfolgen dabei das Prinzip der signalorientierten Datenübertragung. Das heißt, ein Messwert wird von einem Sensor erfasst und per Broadcast an alle ECUs ausgegeben, die den Messwert wiederum an entsprechende Aktoren weiterleiten (s. Abbildung 2.9). Dieses Prinzip beruht auf einer statischen Kommunikationsmatrix, die die Beziehungen zwischen Sendern und Empfängern definiert [137]. Die Modularität der signalorientierten Architektur macht den Aufbau einfach, allerdings steigt der Umfang an ECUs mit neu hinzugefügten Funktionalitäten kontinuierlich [25].

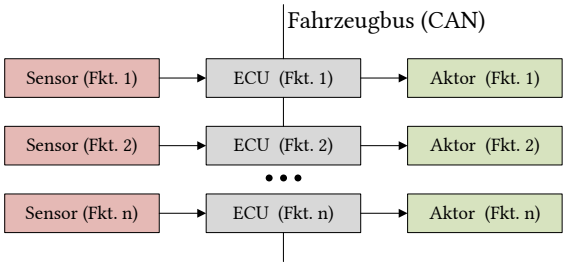


Abbildung 2.9: Verteilte E/E-Architektur [111]

Domänenzentralisierte Architektur

Inzwischen enthalten Personenkraftwagen (PKW) zwischen 70 und 150 ECUs [124], die mit Kabeln einer Länge von bis zu einem Kilometer und einem Gewicht von bis zu 70 Kilogramm miteinander verbunden sind [59]. Die hohe Anzahl an ECUs in Kombination mit dem Broadcast Prinzip führen zu einer hohen Ineffizienz, da eine Vielzahl der sich auf dem Bus befindenden Nachrichten für die meisten ECUs irrelevant sind. Eine Reduzierung der Gesamtlänge der Datenkabel bei gleichzeitiger besserer Einhaltung von Zeitanforderungen brachte das Konzept der Zentralisierung, bei dem ECUs nach Aufgabenbereichen (bspw. Antriebsstrang) zusammengefasst werden. Nach Möglichkeit finden die Aufnahme von Sensordaten, die Verarbeitung und die anschließende Steuerentscheidung in einem sogenannten domänenspezifischen oder gar domänenübergreifenden Gerät statt. Entsprechend steigen aber auch die Anforderungen an die einzelne ECU, beispielsweise müssen die Prozessoren deutlich leistungsfähiger sein [98].

Zentralisierte Architektur

Die zentralisierte Fahrzeugarchitektur verfolgt das Konzept der Datenfusion aus heterogenen Quellen, um ein umfassenderes, systemisches Verständnis des Fahrzeugs und seiner Umgebung zu ermöglichen. Dies stellt eine essenzielle Grundlage für die Realisierung zukünftiger Mobilitätskonzepte wie das automatisierte Fahren dar. In diesem Kontext gewinnen Konzepte aus der Informationstechnik (IT)-Branche zunehmend an Bedeutung. Besonders leistungsfähige ECUs bilden die Grundlage für eine bedarfs- und ressourcengerechte Allokation sowie Ausführung von Funktionen (vgl. Kapitel 3.3) [25].

Die dynamische Zuweisung von Funktionen stellt einen Paradigmenwechsel innerhalb der automobilen E/E-Architekturen dar. Sie ermöglicht die Entwicklung und Integration cloudbasierter Fahrzeugfunktionen, welche den zentralen Forschungsgegenstand dieser Arbeit bilden.

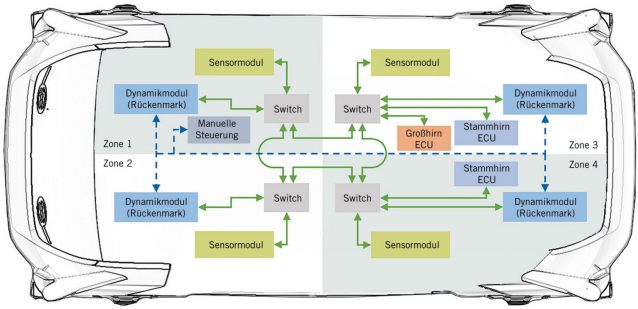


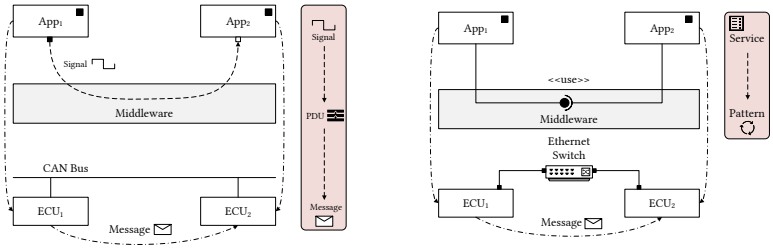
Abbildung 2.10: Zonenarchitektur des UNICARagil Projekts [135]

Im Unterschied zu klassischen, domänenbasierten und überwiegend signalorientierten Architekturen beruhen zentralisierte Architekturen auf serviceorientierten Prinzipien. Ein evolutionärer Zwischenschritt zwischen domänenbasierter und vollständig zentralisierter Architektur ist die sogenannte Zonenarchitektur (vgl. Abbildung 2.10). Dabei erfolgt die Gruppierung von Fahrzeugkomponenten primär basierend auf ihrer physischen Position innerhalb definierter Zonen. In diesem Architekturansatz übernehmen zonale ECU (Zonencontroller) die Erfassung von Sensordaten sowie die Ansteuerung von Aktoren. Diese Zonencontroller sind mit zentralen Hochleistungsrechnern (engl. High-Performance Computer (HPC)) verbunden, welche die übergeordneten Funktionen koordinieren und ausführen.

2.3.2 Die serviceorientierte Architektur

Eine serviceorientierte Architektur (SOA) versteht das System als zusammenhängendes Gebilde von Komponenten, die untereinander kooperieren müssen. Diese Zusammenarbeit basiert darauf, dass eine Komponente einen Service anbietet (Service Provider) und eine andere ihn nachfragt (Service Consumer). Die Beschreibung stellt vielmehr die Anwendung ins Zentrum der Betrachtung, als den technischen Aufbau der Bauelemente und ihrer

Schnittstellen. Diese Herangehensweise erleichtert auch die Wiederverwendung von Software, die durch einheitliche Schnittstellen zusätzlich unterstützt wird. SOAs bieten die Option, den Datenverkehr zwischen verschiedenen Teilnehmern erst während der Laufzeit dynamisch festzulegen. Eine starre Kommunikationsmatrix, wie bei signalorientierten Architekturen, kann vermieden werden [45]. In Abbildung 2.11 wird der Unterschied der beiden Architekturansätze dargestellt. Auf der linken Seite der Abbildung ist eine signalorientierte Kommunikation, bei der logische Signale in Protocol Data Unit (PDU) gruppiert und anschließend in CAN Nachrichten abgebildet werden, dargestellt. Demgegenüber werden bei SOA sowohl Informationen (z.B. „Außentemperatur“) als auch Funktionen (z.B. „Sitz belüften“) als Services angeboten und z.B. mit Internet Protocol (IP)-basierten Protokollen ausgetauscht. Die Definition der sendenden und empfangenden Komponente sowie des Kommunikationsmusters (engl. communication pattern) sind ausreichend (s. Abbildung 2.11b). Dadurch verringert sich der Arbeitsaufwand und überflüssiger Kommunikations-Overhead wird vermieden.



(a) Signalorientiert

(b) Serviceorientiert

Abbildung 2.11: Vergleich zwischen signalorientierter und serviceorientierter Architektur (in Anlehnung an [73]):
a) Klassische Zuordnung von Signalen zu Nachrichten (engl. Messages): Logische Signale werden in PDUs (Protocol Data Units) zusammengefasst, die wiederum CAN-Nachrichten zugeordnet werden.
b) Service-Bindung in einem SOA-Ansatz. Hier werden nur die Interaktionsmuster definiert, während der Rest automatisch erfolgt.

Da sich das Kommunikationsverhalten bei SOA über die Lebenszeit des Produkts verändern kann, sind Updates von Funktionen oder sogar das Hinzufügen gänzlich neuer Funktionen auch nach Auslieferung möglich.

Komponenten einer serviceorientierten Architektur

Die in Tabelle A.2 beschriebenen Open Systems Interconnection (OSI) Schichten und deren Zusammenfassung finden auch in der Automotive-Softwarearchitektur in Abbildung 2.12 Anwendung. Diese Schichten bleiben auch in einer SOA erhalten, mit dem Unterschied, dass die Ebene der Applikations-Software in weitere Ebenen unterteilt wird (s. Abbildung 2.13).

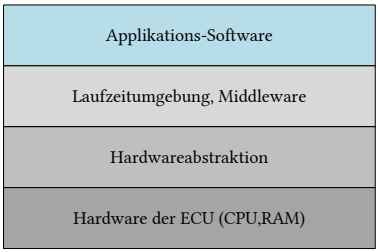


Abbildung 2.12: Grundlegende Komponenten einer Automotive-Softwarearchitektur

Der Konsument greift auf Services zu, die von Softwarekomponenten (engl. software component (SWC)) ausgeführt werden. Die Servicekomponenten Schicht verbindet den Service Contract mit der Implementierung des Service in der SWC. Die einzelnen Services (atomar oder zusammengesetzt) werden durch die Prozessebene verschaltet und zur Anwendung bzw. Applikation zusammengefügt, die dann dem Konsumenten angeboten wird. Der Service ist die entscheidende Ebene innerhalb der SOA und besteht wiederum aus kleineren Komponenten, die nachfolgend beschrieben werden.

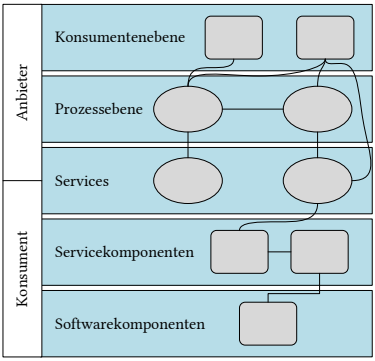


Abbildung 2.13: Ebenen der SOA, angelehnt an [90]

Service Roger Heutschi hat in der Veröffentlichung „Serviceorientierte Architektur: Architekturprinzipien und Umsetzung in die Praxis“ [51] die Definition 2.2 eines Service eingeführt.

> **Definition 2.2 Service:** Ein Service ist ein abstraktes Software-Element bzw. eine Schnittstelle, die anderen Applikationen über ein Netzwerk einen standardisierten Zugriff auf Anwendungsfunktionen bietet.

Dieser Definition wird ein SOA-Service durch die Einführung dreier Komponenten gerecht. Ein Service besteht aus einem Contract (A), der Implementierung (B) und Schnittstellen (C). Die Schnittstelle (C) nach außen zum Client beschreibt die Funktionalität des Services. Der Service-Contract (A) definiert den Service an sich. Diese Definition wird meist auf formaler Ebene in einer Sprache wie IDL[2](Verwendung bei Data Distribution Service (DDS)[3] und damit auch Robot Operating System (ROS) 2 und AUTomotive Open

[2] Interface Definition Language
[3] Data Distribution Service

System ARchitecture (AUTOSAR) Adaptive) oder WSDL[4] durchgeführt. Die Contract Definition abstrahiert und erzeugt dadurch eine weitestgehende Technologieunabhängigkeit. Diese Unabhängigkeit beinhaltet dabei die Aspekte Programmiersprache, Middleware, Laufzeitumgebung und Netzwerkprotokoll.

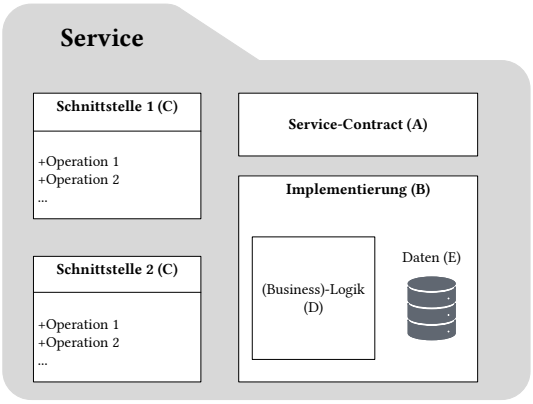


Abbildung 2.14: Komponenten eines Services, nach Krafzig [67]

Die Implementierung (B) ist die tatsächliche technische Umsetzung zur Erfüllung des Contracts eines Services (in ROS 2 ein Node). Die Implementierung beinhaltet die Funktionalität bzw. Logik (D), die dann mittels Service-Schnittstellen zur Verfügung gestellt werden. Die Daten (E) sind nicht zwangsläufig Teil eines Service. Die meisten Services sind *stateless*, das heißt, sie verfügen über keinen Speicher und werden bei jedem Aufruf auf einem gleichen Zustand gestartet.

[4] Web Service Description Language

2.4 Klimatisierung im Stadtbus

2.4.1 Typenvielfalt im Busverkehr

Im öffentlichen Personennahverkehr (ÖPNV) werden Kraftomnibusse zur Beförderung von Fahrgästen eingesetzt. Diese Fahrzeuge lassen sich in drei Fahrzeugklassen einteilen (s. Definition 2.3).

Definition 2.3 Kraftomnibus: Ein Kraftomnibus ist ein Kraftfahrzeug der Klassen M_2 (zGM[a] $\leq 5\,\mathrm{t}$) oder M_3 (zGM $> 5\,\mathrm{t}$), das für die Beförderung von mehr als acht Personen (zusätzlich zum Fahrersitz) ausgelegt ist [30], [31]. Fahrzeuge mit einer Beförderungskapazität von mehr als 22 Fahrgästen werden in drei Fahrzeugklassen untergliedert [30]:

- **Klasse I:** Fahrzeuge, die über Stehplätze verfügen und für den Transport von Fahrgästen auf Strecken mit zahlreichen Haltestellen ausgelegt sind.
- **Klasse II:** Fahrzeuge, die primär für die Beförderung sitzender Fahrgäste konzipiert sind, jedoch auch die Mitnahme stehender Fahrgäste im Gang oder in einem auf maximal zwei Sitzbankreihen begrenzten Bereich zulassen.
- **Klasse III:** Fahrzeuge, die ausschließlich für die Beförderung sitzender Fahrgäste bestimmt sind.

Ein Kraftomnibus kann mehreren Klassen gleichzeitig zugeordnet werden, sofern er für die jeweiligen Klassen zugelassen ist.

[a] zulässige Gesamtmasse (zGM)

In der Praxis erfolgt eine Einteilung der Fahrzeuge nach ihrem vorherrschenden Einsatzzweck. So werden Fahrzeuge der Klasse I üblicherweise als Stadtbusse, Fahrzeuge der Klasse II als Regional- oder Überlandbusse und

Fahrzeuge der Klasse III als Reisebusse bezeichnet. Die Regelung Nr. 107 der Wirtschaftskommission für Europa der Vereinten Nationen (engl. United Nations Economic Commission for Europe (UNECE)) legt die grundlegenden Konstruktionsmerkmale fest, die für die Genehmigung von Fahrzeugen in den verschiedenen Fahrzeugklassen erforderlich sind [130]. Dazu zählen unter anderem die Flächen für stehende und sitzende Fahrgäste, die Anzahl und Anordnung von Betriebstüren und Notausstiegen, die Gestaltung der Fenster, Gangbreite, Stufenhöhe, Einstiegshöhe sowie der Abstand und die Ausführung von Sitzen. Auch Anforderungen an Brandunterdrückungssysteme, Beleuchtung und Kennzeichnung sind Teil dieser Regelung.

Definition 2.4 Solobus: Ein einteiliger Kraftomnibus ohne Gelenk, typischerweise etwa 12 Meter lang [122].

Definition 2.5 Gelenkbus: Ein mindestens zweiteiliger Kraftomnibus, verbunden durch ein Gelenk, mit einer Länge von etwa 18 Metern [122]. Die Straßenverkehrs-Zulassungs-Ordnung (StVZO) sieht eine Länge von $18{,}75\,\mathrm{m}$ vor und erfordert beim Einsatz im öffentlichen Straßenverkehr von Fahrzeugen, die diese Länge überschreiten oder über mehr als ein Gelenk verfügen, eine Ausnahmegenehmigung entsprechend §70 der StVZO.

Batterieelektrische Stadtbusse Im batterieelektrischen Bus (BEB) wird alle benötigte Energie für den Antrieb und alle sonstigen Verbraucher aus der Hochvoltbatterie (Abk. HV-Batterie) entnommen. Aufgrund der begrenzten Batteriekapazität stellt die effiziente Gestaltung der Verbraucher die größte Herausforderung im BEB dar. Ein Vergleich der Energiemengen in einem dieselbetriebenen *Citaro* und einem batterieelektrischen *eCitaro* des Herstellers Mercedes-Benz unterstreicht diese Herausforderung. Mithilfe des

Heizwerts von Diesel und dem Tankinhalt eines *Citaro*[5] Busses bestimmt sich der Energieinhalt einer Tankfüllung zu:

$$H_{\mathrm{i,vol,\,Diesel}} = \begin{cases} 34{,}7\,\mathrm{MJ\,L^{-1}} \\ 9{,}64\,\mathrm{kW\,h\,L^{-1}} \end{cases} \tag{2.8}$$

$$E_{\mathrm{Tank}} = V_{\mathrm{Tank}} \cdot H_{\mathrm{i,vol,\,Diesel}} = 2.506\,\mathrm{kW\,h} \tag{2.9}$$

Die größte im *eCitaro* Solobus erhältliche Batteriekonfiguration verfügt über eine Batteriekapazität von:

$$E_{\mathrm{Batterie}} = 588\,\mathrm{kW\,h}^{6} \tag{2.10}$$

Dem BEB steht also nur $23{,}5\,\%$ der Energiemenge eines Dieselbusses zur Verfügung. Die mit dieser Energiemenge erzielbare Reichweite hängt stark von der Umgebungstemperatur ab, da die Klimatisierung einen erheblichen Anteil am Gesamtenergiebedarf des BEB ausmacht (vgl. Kapitel 3.5). Das in Abbildung 2.15 dargestellte Sankey-Diagramm der Energieflüsse im BEB verdeutlicht den großen Anteil thermischer Wandler wie der Wärmepumpe oder dem Zusatzheizgerät, die Energie für die Klimatisierung bereitstellen. Zur Bewertung des Energieverbrauchs von Fahrzeugen und deren Komponenten wird typischerweise die Energiemenge in Relation zur gefahrenen Strecke gesetzt (Energieverbrauch e in $\mathrm{kW\,h\,km^{-1}}$).

Bei einer Umgebungstemperatur von $15\,^{\circ}\mathrm{C}$ ist der streckenbezogene Gesamtenergieverbrauch mit durchschnittlich $e_{\mathrm{Basis}} = 1\,\mathrm{kW\,h\,km^{-1}}$ am geringsten [15]. Bei diesen Temperaturen kann mit dem Solobus mit der maximalen Batteriekapazität also eine theoretische Reichweite von $s_{\mathrm{max}} = 588\,\mathrm{km}$

[5] https://www.mercedes-benz-bus.com/de_DE/models/citaro/facts/facts-citaro.pdf
[6] https://www.mercedes-benz-bus.com/de_DE/models/ecitaro/technology/battery-technology.html

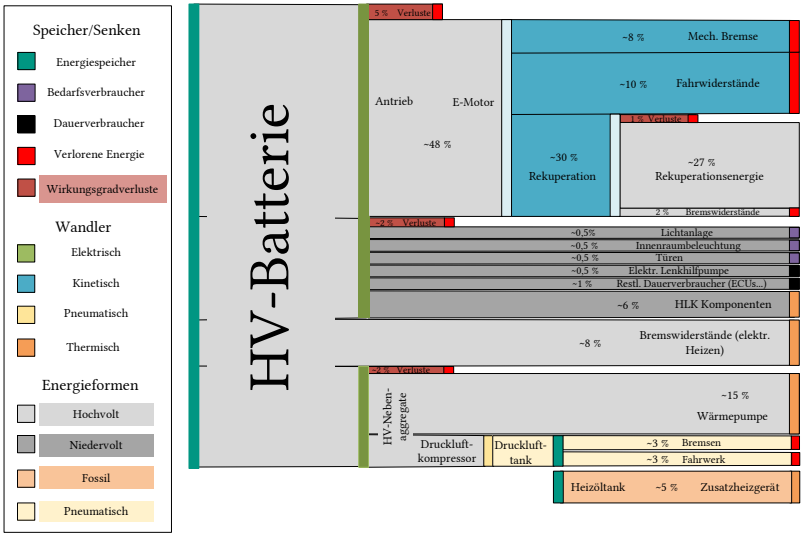


Abbildung 2.15: Sankey-Diagramm des Energieflusses im BEB (angelehnt an [71])

zurückgelegt werden. Im Heizbetrieb unterhalb von 10 °C reduziert sich die genannte maximale Reichweite um bis zu 58 % [100] auf:

$$s_{\max,\mathrm{Heiz}} = 247\,\mathrm{km} \tag{2.11}$$

Im Kühlbetrieb fällt der Abfall mit 47 % [100] nur marginal geringer aus und es ergibt sich eine Reichweite von:

$$s_{\max,\mathrm{Kühl}} = 311\,\mathrm{km} \tag{2.12}$$

Mit $s_{\max}$ können alle Umlaufpläne für Solobusse des deutschen Busverkehrssystems abgedeckt werden (s. Abbildung 2.16). Reduziert sich die Reichweite auf $s_{\max,\mathrm{Kühl}}$ können noch 82 % der Umläufe realisiert werden, während bei $s_{\max,\mathrm{Heiz}}$ sogar nur noch 50 % abgedeckt werden können.

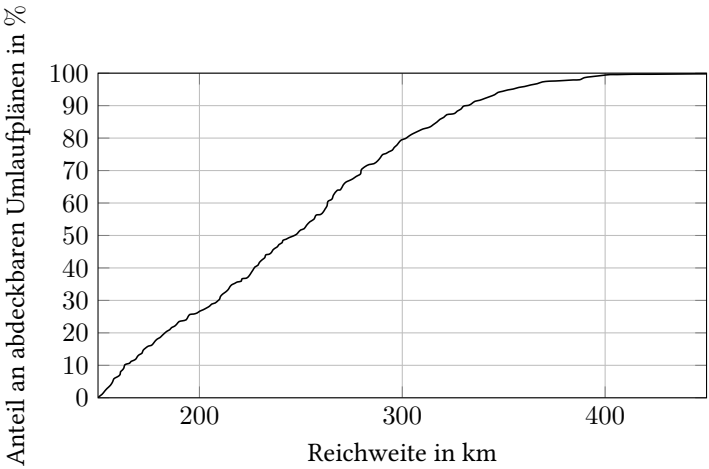


Abbildung 2.16: Anteil an abdeckbaren Umlaufplänen des deutschen Busverkehrssystems bei gegebener Reichweite eines Busses [64]

2.4.2 HLK-Systeme im Stadtbus

Die Klimatisierung eines Elektrofahrzeugs stellt im Vergleich zu einem Fahrzeug mit Verbrennungsmotor eine deutlich größere Herausforderung dar. Durch die fehlende Abwärme des Verbrennungsmotors muss in einem Elektrofahrzeug der komplette Heizbedarf durch zusätzliche Heizkomponenten bereitgestellt werden. Neben passiven Maßnahmen, wie z. B. bessere Isolierung des Fahrzeugs zur Verringerung des Wärmeverlustes im Winter oder Wärmeeintrags im Sommer, gibt es verschiedene Möglichkeiten und Varianten, die in elektrisch betriebenen Fahrzeugen zur Klimatisierung zum Einsatz kommen. Dazu zählen für Heizzwecke elektrisch-, mechanisch- und kraftstoffbetriebene Zuheizer. Ein anderer Ansatzpunkt ist die Verwendung einer Wärmepumpe zur Bereitstellung der benötigten Heizleistung. Die Kühlung erfolgt sowohl bei Fahrzeugen mit Verbrennungsmotor als auch mit alternativen Antriebstechniken durch den Einsatz von Kompressionskältemaschinen [24]. Außer im Fall des Wärmepumpensystems sind die Heiz-

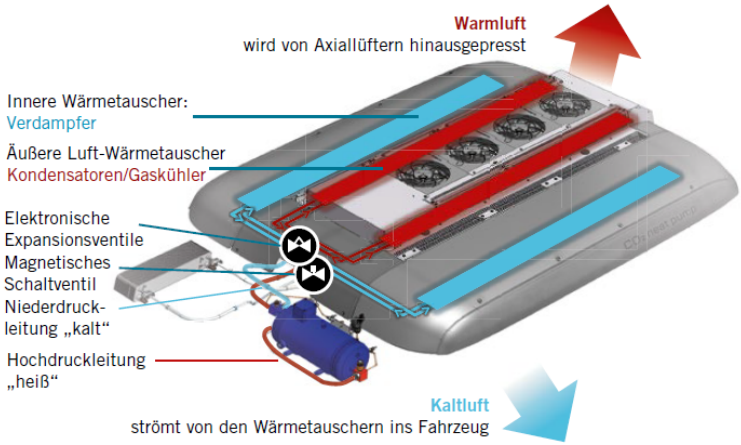


Abbildung 2.17: Darstellung der Aufdachanlage mit dem Wirkprinzip des Kühlkreislaufs [15]

und Kühlkreisläufe voneinander getrennt [61].

Tabelle 2.2: Übliche HLK-Systeme in Elektrobussen (in Anlehnung an [61])

Variante	Kühlung	Heizung
1	Aufdachklimaanlage	Elektrische Widerstandsheizung
2	Aufdachklimaanlage	Kraftstoffheizung
3	Aufdachwärmepumpe für Kühl- und Heizbetrieb (ggf. mit Zuheizer)	

In Tabelle 2.2 sind die in Elektrobussen (vgl. batterieelektrischer Bus in Kapitel 2.4.1) am häufigsten eingesetzten HLK-Systeme zu sehen [61]. Die Kühlung erfolgt jeweils durch eine Kompressionskältemaschine, welche in einer Aufdachanlage untergebracht ist (s. Abbildung 2.17). Für den Einsatz in

einem Elektrobus werden Aufdachklimaanlagen mit unterschiedlichen Kältemitteln angeboten. Häufig eingesetzt wird das Kältemittel R134a, welches allerdings aufgrund seines hohen Treibhauspotentials in der Kritik steht und für neu entwickelte PKW sowie Transporter bereits verboten ist [119]. Als Alternative wurden in den vergangenen Jahren natürliche Kältemittel für den Einsatz in Klimaanlagen untersucht. Dabei hat sich Kohlenstoffdioxid (CO_2), auch als R744 bezeichnet, als vielversprechende Option herauskristallisiert. Aus ökologischer Sicht ist CO_2 ein nahezu ideales Kältemittel, da es weder giftig noch brennbar ist sowie ein geringes Treibhauspotential besitzt. Ein Austritt in die Atmosphäre ist daher unbedenklich und hat im Vergleich zu anderen Kohlenstoffdioxidemittenten einen vernachlässigbaren Einfluss auf die globale Erwärmung. Nachteilig beim Einsatz von Kohlenstoffdioxid als Kältemittel sind die erforderlichen hohen Drücke und die damit einhergehenden höheren Kosten für die Bauteile der Klimaanlage [99].
Für die Heizung eines Elektrobusses greifen die meisten Hersteller auf eine elektrische Widerstandsheizung, eine Kraftstoffheizung oder eine Wärmepumpe zurück. Durch den Einsatz von kraftstoffbetriebenen Heizungen sind auch Elektrobusse häufig mit einem Kraftstofftank ausgestattet und daher nicht vollständig emissionsfrei. In Abbildung 2.18 sind die Verschaltungen der Heiz- und Kühlsysteme sowie die Anordnung der Komponenten der drei üblichen HLK-Systemvarianten schematisch dargestellt. Die Varianten 1 und 2 aus Tabelle 2.2 werden in Abbildung 2.18a dargestellt. Diese nutzen einen Flüssigkeitskreislauf, der mithilfe eines elektrischen bzw. kraftstoffbetriebenen Zuheizers erwärmt wird. Dieser Kreislauf ist an bodennahe Konvektoren angeschlossen, welche die Luft in der Fahrzeugkabine erwärmen. In Stadtbussen sind diese Konvektoren zumeist als Seitenwandheizer mit Gebläse realisiert. Zusätzlich ist der Flüssigkeitskreislauf mit einem Heizungswärmeübertrager in der Aufdachanlage verbunden. Dort wird je nach Stellung der Umluftklappe Frisch-, Misch- oder Umluft erwärmt und schließlich der Fahrzeugkabine zugeführt. Durch die Stellung der Umluftklappe kann der Frischluftanteil innerhalb des Busses gesteuert werden. Im Heizfall ist die Klimaanlage ausgeschaltet und der Luft wird somit bei der Durchströmung des Verdampfers der Kompressionskältemaschine kei-

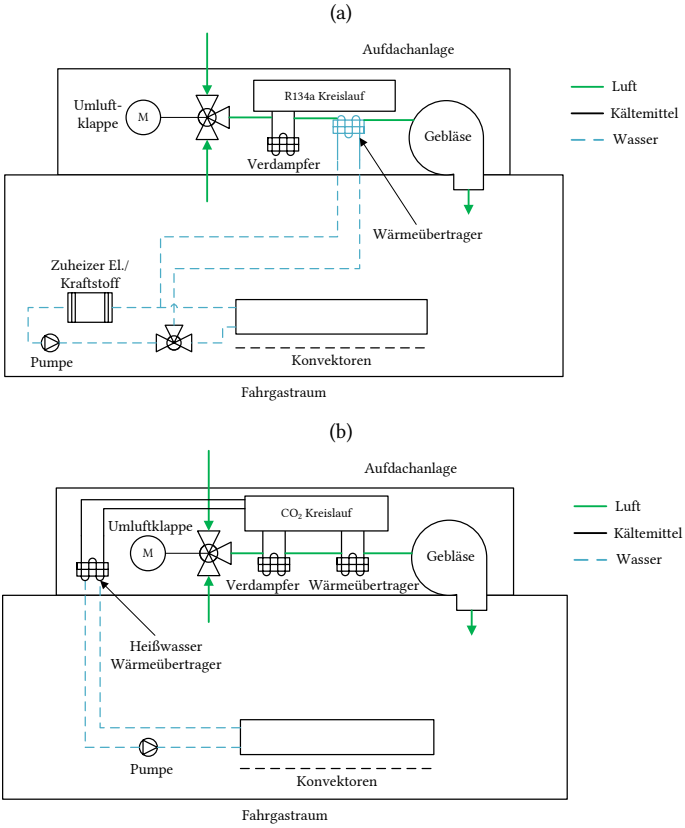


Abbildung 2.18: Schematischer Aufbau unterschiedlicher HLK-Systeme eines Elektrobusses (in Anlehnung an [61]):
a) Aufdachklimaanlage mit elektrischem bzw. kraftstoffbetriebenem Heizgerät (verwendetes Kältemittel R134a)
b) Aufdachwärmepumpe für Heiz- und Kühlbetrieb (verwendetes Kältemittel R744 (CO_2))

ne Wärme entzogen. Ist die Klimaanlage eingeschaltet, wird die Luft nach der Umluftklappe im Verdampfer gekühlt. Im Kühlfall ist der Zuheizer ausgeschaltet und der Flüssigkeitskreislauf gestoppt, wodurch die Luft, ohne

erwärmt zu werden, durch den Heizungswärmeübertrager in die Kabine geleitet wird. Es gibt auch Verschaltungen, in denen die Luft über einen Bypass am Heizungswärmeübertrager vorbeigeleitet wird [109]. Alternativ zu den oben beschriebenen Heizmethoden mit Flüssigkeitskreislauf existieren auch Heizkonzepte, bei denen sowohl der Heizungswärmeübertrager als auch die bodennahen Konvektoren als reine elektrische Positive Temperature Coefficient (PTC)-Luftheizungen ausgeführt sind.
Durch das häufige Öffnen der Türen an den Haltestellen eines Stadtbusses findet in kurzen Abständen ein erheblicher Wärmeaustausch zwischen Fahrzeugkabine und Umgebung statt. Daher werden für den Einsatz von Stadtbussen in Gebieten mit extremen Außentemperaturen teilweise zusätzlich Türluftschleier eingesetzt. Diese erzeugen vor den Türen des Busses mittels starker Gebläse eine Barriere aus strömender Luft. Dadurch wird der Austausch von sehr kalter oder sehr warmer Außenluft mit der klimatisierten Luft der Fahrzeugkabine reduziert [61].
Bei Variante 3 aus Tabelle 2.2 (s. Abbildung 2.18b) erfolgt sowohl die Kühlung als auch die Heizung mittels einer Wärmepumpe. Durch Umkehrung des Kältemittelkreislaufs oder Luftumkehr kann eine Wärmepumpe als Kompressionskältemaschine fungieren. Diese Umkehrung wird genutzt, um zwischen Heiz- und Kühlbetrieb umschalten zu können und somit sowohl den Heiz- als auch den Kühlbedarf durch die Aufdachwärmepumpe bereitzustellen. Beim Einsatz von R134a als Kältemittel der Wärmepumpe muss aufgrund der thermodynamischen Eigenschaften bei niedrigen Außentemperaturen auf eine Zusatzheizung zurückgegriffen werden. Besser geeignet für Temperaturbereiche unter $0\,^{\circ}\mathrm{C}$ ist dagegen Kohlenstoffdioxid. Allerdings kann es auch beim Einsatz von CO_2-Wärmepumpen sinnvoll sein, eine kraftstoffbetriebene Zusatzheizung zu integrieren, um diese bei extremen Witterungsverhältnissen oder zur Verlängerung der Reichweite bedarfsgerecht zuzuschalten. Zusätzlich wird oftmals die Abwärme des Bremswiderstandes als Wärmequelle genutzt.

Eigenschaften verschiedener Wärmequellen

Folgende Eigenschaften dienen der Beschreibung von Wärmequellen:

- **Elektrische Effizienz:** Der Coefficient of Performance (COP) gilt als Maß für die Effizienz von Wärmepumpen (s. Definition A.5). Dieser Ansatz wird verwendet, um die Effizienz elektrisch betriebener Wärmequellen zu berechnen (s. Definition 2.6).

  > **Definition 2.6 Electrical Coefficient of Performance:** Der *electrical coefficient of performance* bezeichnet bei einer elektrisch betriebenen Wärmequelle das Verhältnis des nutzbaren Wärmestroms $\dot{Q}_{\mathrm{ab}}$ zu eingebrachter elektrischer Leistung $P_{\mathrm{elektrisch}}$:
  >
  > $$COP_{\mathrm{el}} = \frac{\dot{Q}_{\mathrm{ab}}}{P_{\mathrm{elektrisch}}} \tag{2.13}$$

- **Leistungsbereich:** Wärmequellen können einen nutzbaren Wärmestrom in einem bestimmten Wärmebereich zur Verfügung stellen.

$$\dot{Q}_{\mathrm{ab,\,min}} < \dot{Q}_{\mathrm{ab}} < \dot{Q}_{\mathrm{ab,\,max}} \tag{2.14}$$

- **Temperaturniveau:** Für den Transport und die anschließende Verwendung der Wärme ist deren Temperaturniveau ausschlaggebend.

$$T_{\mathrm{ab,\,min}} < T_{\mathrm{ab}} < T_{\mathrm{ab,\,max}} \tag{2.15}$$

- **Betriebliche Einschränkungen:** Wärmequellen können zusätzliche Abhängigkeiten haben, die je nach Betriebszustand die Verfügbarkeit beeinflussen.

2.4.3 Regelwerk zur Klimatisierung von Linienbussen des Verbands deutscher Verkehrsunternehmen

Der Verband Deutscher Verkehrsunternehmen (VDV) beschäftigt sich in seiner Schrift 236 mit der Klimatisierung von Linienbussen und definiert dabei unter anderem die Anforderungen an einzuhaltende Temperaturbereiche in der Fahrzeugkabine (s. Abbildung 2.19). Es werden Komfort und Economy Kennlinien definiert. Die Economy Kennlinie ist für Verkehre mit geringeren Reichweiten bzw. kurzer Aufenthaltsdauer der Fahrgäste und E-Mobilität sinnvoll und ist gekennzeichnet durch:

- Heizen im Bereich der unteren Kennfeldgrenze
- Kühlen im Bereich der oberen Kennfeldgrenze

Die Kennlinien müssen geringfügig anpassbar sein. Die Soll-Innenraumtemperatur darf im Heizbetrieb um $\pm 2\ K$ verschoben werden. Die Starttemperatur für den Kühlbetrieb darf ebenfalls um $\pm 2\ K$ angepasst werden.

Die Fahrzeugkabine wird in verschiedene Klimazonen unterteilt (s. Definition 2.7).

> **Definition 2.7 Klimazone:** Eine Klimazone umfasst jeweils den Bereich eines Gesamtsystems, der eigenständig kontrollierbar klimatisiert werden kann. Ein Solo- Linienbus wird in die Klimazonen Fahrerarbeitsplatz (FAP) und Fahrgastraum (FGR) aufgeteilt, bei einem Gelenkbus kommt entsprechend eine zweite FGR-Zone hinzu.

Typischerweise wird im Linienbus jeder Zone eine Aufdachanlage zugeordnet, während der FAP über ein zusätzliches System, genannt Frontbox, klimatisiert wird.

Neben Vorgaben zu Temperaturbereichen legt die Vorschrift ebenfalls einen Mindestwert für den Frischluftanteil von $15\,\mathrm{m^3\,h^{-1}}$ pro Person in der Fahr-

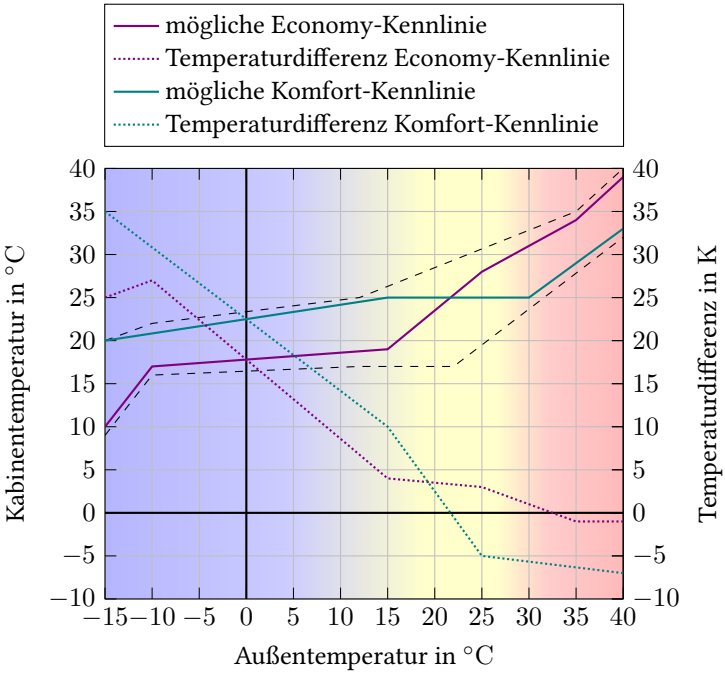


Abbildung 2.19: Mögliche Economy- und Komfort-Kennlinien nach VDV-Schrift 236 mit den Betriebszuständen Heizen (blau), Lüften (gelb) und Kühlen (rot) [123], sowie jeweils davon abgeleitete Temperaturdifferenz zwischen Kabinen- und Umgebungstemperatur

zeugkabine fest. Dieser Wert darf unterschritten werden, wenn die Luftgüte durch eine entsprechende Messung des CO_2 Gehalts oder der relativen Luftfeuchte oder der Luftreinheit (Partikel) sichergestellt wird.

2.4.4 Die thermische Modellierung der HLK-Vorgänge im Stadtbus

Das thermische Modell der HLK-Vorgänge orientiert sich an der Aufteilung des Stadtbusses in Klimazonen (s. Definition 2.7). Die Enthalpie- und Wärmeströme in einer Klimazone sind in Abbildung 2.20 abgebildet.

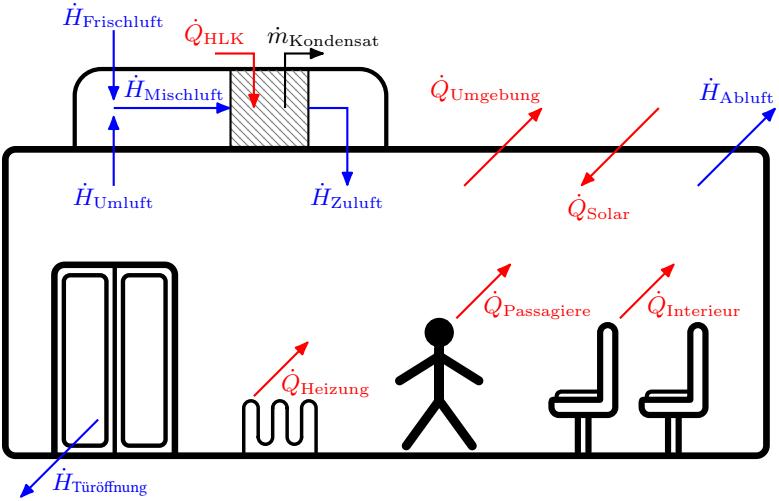


Abbildung 2.20: Wärme- und Massenströme in einem Stadtbus (angelehnt an [46])

Die in Abbildung 2.20 blau dargestellten Enthalpieströme deuten den Transport feuchter Luft an, während die rot dargestellten Wärmeströme einen Wärmetransport durch Konduktion oder Strahlung aufzeigen. Die Luftmenge innerhalb einer Klimazone der Fahrzeugkabine wird als homogene Masse mit den Zustandsgrößen (s. auch Kapitel A.1)

- Temperatur T_{Kabine}
- Feuchtegrad x_{Kabine}
- Spezifische Enthalpie h_{Kabine}

und entsprechend mit T_{Umgebung}, x_{Umgebung} und h_{Umgebung} betrachtet.

In der HLK-Anlage werden Umluft und Frischluft abhängig von der Umluftklappe (s. Abbildung 2.18) gemischt. Der Frischluftanteil a beschreibt dabei den Anteil des Frischluftmassenstroms am gesamten Mischluftmassenstrom,

$$a = \frac{\dot{m}_{\text{Frischluft}}}{\dot{m}_{\text{Mischluft}}} = \frac{\dot{m}_{\text{Frischluft}}}{\dot{m}_{\text{Frischluft}} + \dot{m}_{\text{Umluft}}} \tag{2.16}$$

sodass die Mischluft wie folgt berechnet wird:

$$x_{\text{Mischluft}} = (1 - a) \cdot x_{\text{Kabine}} + a \cdot x_{\text{Umgebung}} \tag{2.17}$$

$$h_{\text{Mischluft}} = (1 - a) \cdot h_{\text{Kabine}} + a \cdot h_{\text{Umgebung}} \tag{2.18}$$

Definition 2.8 Enthalpie: Die Enthalpie H (Einheit: J) ist eine Zustandsgröße eines thermodynamischen Systems. Sie berechnet sich aus der inneren Energie U, dem Druck p und dem Volumen V und kann nicht direkt gemessen werden:

$$H = U + pV \tag{2.19}$$

Wird die Enthalpie H auf eine Masse m bezogen, entsteht eine intensive Größe, die spezifische Enthalpie h (Einheit: $\mathrm{J\,kg^{-1}}$):

$$h = \frac{H}{m} \tag{2.20}$$

Definition 2.9 Enthalpiestrom: Wird Luft mit der spezifischen Enthalpie h durch einen Luftmassenstrom $\dot{m}_{\mathrm{L}}$ bewegt, entsteht ein Enthalpiestrom $\dot{H}$:

$$\dot{H} = h \cdot \dot{m}_{\mathrm{L}} \tag{2.21}$$

Ein Enthalpiestrom $\dot{H}$ ist gleichbedeutend mit einem Wärmestrom $\dot{Q}$ und wird wie dieser in einen sensiblen Anteil (Änderung der Temperatur) und einen latenten Anteil (Änderung des Feuchtegrads) aufgeteilt. Im Folgenden wird der Begriff Enthalpiestrom zur besseren Unterscheidung verwendet, um so einen Wärmestrom zu kennzeichnen, der explizit durch den Transport von feuchter Luft hervorgerufen wird.

In HLK-Systemen wird der Frischluftanteil a durch die Umluftklappe und der Mischluftmassenstrom $\dot{m}_{\mathrm{Mischluft}}$ durch das Gebläse vorgegeben. Somit ergeben sich folgende Enthalpieströme (s. Definition 2.9):

$$\dot{H}_{\mathrm{Frischluft}} = h_{\mathrm{Umgebung}} \cdot \dot{m}_{\mathrm{Frischluft}} = a \cdot h_{\mathrm{Umgebung}} \cdot \dot{m}_{\mathrm{Mischluft}} \tag{2.22}$$

$$\dot{H}_{\mathrm{Umluft}} = h_{\mathrm{Kabine}} \cdot \dot{m}_{\mathrm{Umluft}} = (1-a) \cdot h_{\mathrm{Kabine}} \cdot \dot{m}_{\mathrm{Mischluft}} \tag{2.23}$$

$$\dot{H}_{\mathrm{Mischluft}} = \dot{H}_{\mathrm{Frischluft}} + \dot{H}_{\mathrm{Umluft}} \tag{2.24}$$

Der Mischluft wird in der HLK-Einheit der Wärmestrom $\dot{Q}_{\mathrm{HLK}}$ zugeführt. Die dabei entstehende Zuluft in die Fahrzeugkabine lässt sich folgendermaßen berechnen:

$$\dot{H}_{\mathrm{Zuluft}} = \dot{H}_{\mathrm{Frischluft}} + \dot{H}_{\mathrm{Umluft}} + \dot{Q}_{\mathrm{HLK}} \tag{2.25}$$

Der Massenstrom $\dot{m}_{\mathrm{Abluft}}$ beschreibt, wie viel Luft aus der Kabine an die Umgebung abgegeben wird und bestimmt den zugehörigen Enthalpiestrom $\dot{H}_{\mathrm{Abluft}}$:

$$\dot{H}_{\mathrm{Abluft}} = h_{\mathrm{Kabine}} \cdot \dot{m}_{\mathrm{Abluft}} \tag{2.26}$$

Türöffnungen führen zu einem bidirektionalen Austausch von Luft zwischen Fahrzeugkabine und Umgebung, der zu einem Enthalpiestrom $\dot{H}_{\text{Türe}}$ pro Anzahl der geöffneten Türen $N_{\text{Türen, geöffnet}}$ führt:

$$\dot{H}_{\text{Türe}} = h_{\text{Kabine}} \cdot \dot{m}_{\text{Tür, ausströmend}} - h_{\text{Umgebung}} \cdot \dot{m}_{\text{Tür, einströmend}} \tag{2.27}$$

$$\dot{H}_{\text{Türöffnung}} = N_{\text{Türen, geöffnet}} \cdot \dot{H}_{\text{Türe}} \tag{2.28}$$

Besteht zwischen T_{Kabine} und T_{Umgebung} eine Temperaturdifferenz, so wird abhängig von der Oberfläche A und dem Wärmedurchgangskoeffizient U (in $\mathrm{W\,m^{-2}\,K^{-1}}$) [116] der Wärmestrom $\dot{Q}_{\text{Umgebung}}$ von der Kabine an die Umgebung abgegeben:

$$\dot{Q}_{\text{Umgebung}} = A \cdot U \cdot (T_{\text{Kabine}} - T_{\text{Umgebung}}) \tag{2.29}$$

$$\frac{1}{U} = \frac{1}{U_{\text{Konvektion}}} + \frac{1}{U_{\text{Konduktion}}} = \frac{1}{h_{\text{Innen}}} + \frac{1}{h_{\text{Außen}}} + \sum_{i=1}^{n} \frac{d_i}{k_i} \tag{2.30}$$

Eine ausführliche Darstellung der Berechnung des Wärmedurchgangs findet sich bei Schild [105].
Der Wärmeeintrag durch Sonneneinstrahlung setzt sich aus direkter, diffuser und reflektierter Solarstrahlung zusammen:

$$\dot{Q}_{\text{Solar}} = \dot{Q}_{\text{direkt}} + \dot{Q}_{\text{diffus}} + \dot{Q}_{\text{reflektiert}} \tag{2.31}$$

Die Fahrzeugkabine kann durch zusätzliche Heizungen mit Umwälzgebläse ausgestattet werden. Diese führt dem Raum den Wärmestrom $\dot{Q}_{\text{Heizung}}$ zu.

Eine Temperaturdifferenz der Innenausstattung $T_{\text{Interieur}}$ (vereinfacht als eine homogene thermische Masse $m_{\text{Interieur}}$) von T_{Kabine} erzeugt den Wärmestrom $\dot{Q}_{\text{Interieur}}$:

$$\dot{Q}_{\text{Interieur}} = m_{\text{Interieur}} \cdot c_{\text{p, Interieur}} \cdot \dot{T}_{\text{Interieur}} \tag{2.32}$$

Dieser ist hauptsächlich für Aufheiz- und Abkühlvorgänge relevant, da sich $T_{\mathrm{Interieur}}$ und T_{Kabine} im stationären Zustand annähern und $\dot{Q}_{\mathrm{Interieur}}$ damit vernachlässigbar klein wird.
Ein normal bekleideter, sitzender Mensch gibt einen Wärmestrom $\dot{Q}_{\mathrm{P}}$ zwischen 110 W und 120 W ab, welcher sich abhängig von der Raumtemperatur aus sensibler und latenter Wärme zusammensetzt (s. Abbildung A.20). Je höher die Raumtemperatur ist, desto größer ist der Anteil der Wärmeabgabe, der durch Verdunstung von Schweiß erreicht wird und somit zu latenter Wärme führt. Der Wärmestrom durch die Passagiere ergibt sich mit deren Anzahl $N_{\mathrm{Passagiere}}$ zu:

$$\dot{Q}_{\mathrm{Passagiere}} = N_{\mathrm{Passagiere}} \cdot \dot{Q}_{\mathrm{P}} = N_{\mathrm{Passagiere}} \cdot \left(\dot{Q}_{\mathrm{P,\,sen}} + \dot{Q}_{\mathrm{P,\,lat}} \right) \tag{2.33}$$

Der gesamte auf die Fahrzeugkabine einwirkende Wärmestrom $\dot{Q}_{\mathrm{Gesamt}}$ berechnet sich wie folgt:

$$\begin{aligned} \dot{Q}_{\mathrm{Gesamt}} = \dot{H}_{\mathrm{Zuluft}} - \dot{H}_{\mathrm{Umluft}} - \dot{H}_{\mathrm{Abluft}} - \dot{H}_{\mathrm{Türöffnung}} - \dot{Q}_{\mathrm{Umgebung}} \\ + \dot{Q}_{\mathrm{Solar}} + \dot{Q}_{\mathrm{Heizung}} + \dot{Q}_{\mathrm{Interieur}} + \dot{Q}_{\mathrm{Passagiere}} \end{aligned} \tag{2.34}$$

Durch Lösen der folgenden Differentialgleichung kann schließlich die Kabinentemperatur bestimmt werden:

$$\dot{Q}_{\mathrm{Gesamt}} = m_{\mathrm{Kabine,L}} \cdot c_{\mathrm{p,L}} \cdot \frac{dT_{\mathrm{Kabine}}}{dt} \tag{2.35}$$

Die Fahrzeugkabine weist dabei eine thermische Masse der Luft $m_{\mathrm{Kabine,L}}$ mit der spezifischen Wärmekapazität $c_{\mathrm{p,L}}$ auf.

Beim Zusammenführen der einzelnen Klimazonen (s. Definition 2.7) in einem Fahrzeug entstehen zusätzliche Enthalpieströme, die den Luftaustausch zwischen den Zonen abbilden. In einem Solobus tritt dabei ein Enthalpiestrom $\dot{H}_{\mathrm{FAP,FGR1}}$ zwischen dem FAP und dem FGR auf. In einem Gelenkbus

kommt ein weiterer Strom $\dot{H}_{\mathrm{FGR1,FGR2}}$ zwischen dem ersten und dem zweiten FGR hinzu (s. Abbildung 2.21).

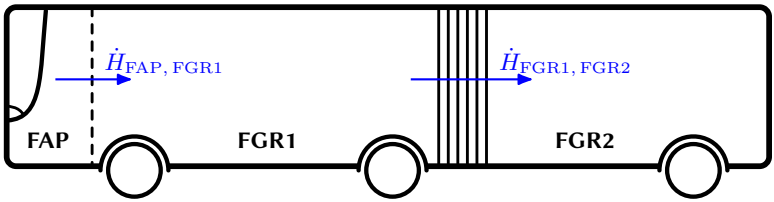


Abbildung 2.21: Enthalpieströme zwischen den Klimazonen im Stadtbus [100]

2.5 Cloud Computing

Mit der zunehmenden Bedeutung der IT kann die Verfügbarkeit von Rechenkapazität als ein Grundbedürfnis ähnlich der Verfügbarkeit von Strom angesehen werden. Von diesem Standpunkt aus können Benutzer unabhängig von Dienststandort und Technologie auf Rechenressourcen zugreifen. Cloud Computing (CC) ist ein Paradigma zur Bereitstellung von On-Demand-Computing-Dienstprogrammen aus Rechenzentren für Benutzer an verschiedenen Standorten. Eine Cloud besteht aus einer Sammlung von Computern, die miteinander verbunden und virtualisiert sind [26]. Es gibt mehrere Definitionen für Cloud Computing. Eine der etablierten stammt vom National Institute of Standards and Technology (NIST), die CC wie folgt definieren:

> **Definition 2.10 Cloud Computing:** Cloud Computing ist ein Modell zur Ermöglichung allgegenwärtiger, bequemer On-Demand-Netzwerkzugriffe auf einen gemeinsam genutzten Pool konfigurierbarer Rechenressourcen (z. B. Netzwerke, Server, Speicher, Anwendungen und Dienste), die mit minimalem Verwaltungsaufwand oder Interaktion mit dem Dienstanbieter schnell bereitgestellt und freigegeben werden können [79].

Vergleich Cloud vs. ECU Die in Tabelle 2.3 beschriebenen Eigenschaften zeigen die Flexibilität und Leistungsfähigkeit des Cloud Computing. Tabelle 2.4 zeigt einen Vergleich zwischen Cloud und ECU hinsichtlich der Funktionsausführung. Während ECUs nur begrenzte Rechenkapazitäten bieten, vergleichbar mit einem Raspberry Pi, kann in der Cloud potenziell auf spezialisierte Hochleistungs-Hardware wie die NVIDIA A100[7] zurückgegriffen werden, die über 100 Tera Floating Point Operations Per Second (TFLOPS) Rechenleistung bereitstellt. Dem gegenüber steht jedoch ein er-

[7] `https://www.nvidia.com/de-de/data-center/a100/`

Tabelle 2.3: Wesentliche Eigenschaften des Cloud Computing [79]

Eigenschaft	**Beschreibung**
On-Demand-Self-Service	Ein Verbraucher kann Rechenkapazitäten wie Serverzeit und Netzwerkspeicher einseitig nach Bedarf automatisch bereitstellen, ohne dass eine menschliche Interaktion mit jedem Dienstanbieter erforderlich ist.
Breiter Netzwerkzugriff	Auf Computer- und Speicherkapazitäten kann über das Netzwerk durch Standard-Benutzerplattformen, wie z. B. Mobiltelefone, zugegriffen werden.
Ressourcenzusammenlegung	Die bereitgestellten Ressourcen sind standortunabhängig. Mehrere physische und virtuelle Ressourcen werden den Kundenanforderungen dynamisch zugewiesen.
Hohe Elastizität	Die bereitgestellten Ressourcen lassen sich elastisch skalieren und wirken oft unbegrenzt.
Messbarer Service	Die verwendeten Services können vom Anbieter und vom Kunden überwacht, kontrolliert und gemeldet werden.

Tabelle 2.4: Vergleich der Funktionsausführung auf Steuergerät und Cloud [7]

Eigenschaft	**ECU**	**Cloud**
Rechenkapazität	Gering (vgl. Raspberry Pi 4 Model B)	Skalierbar (bis zu >100 TFLOPS z.B. in NVIDIA A100)
Speicherressourcen	Beschränkt (<100 GB)	Skalierbar bis Petabyte
Kommunikationslatenz	Gering (vorhersagbar)	Hoch >50 ms
Funktionale Sicherheit	Sehr hoch	Gering bis Mittel
Verfügbarkeit	Sehr hoch	Abhängig von Komm.-kanal
Security	Sehr hoch	Gering
Geographische Abdeckung	National oder global	Lokal

heblicher Nachteil in der Kommunikationslatenz. ECUs zeichnen sich durch eine vorhersagbar niedrige Latenz aus, was für zeitkritische Anwendungen unerlässlich ist. Die Cloud hingegen verursacht durch den notwendigen Datenverkehr über Netzwerke typischerweise Latenzen von über $50\,\mathrm{ms}$.

Auch bei funktionaler Sicherheit (vgl. Definition 4.2) und Security (s. Definition 4.3) zeigen sich signifikante Unterschiede. Lokale Steuergeräte unter-

liegen häufig strengen Sicherheitsstandards und bieten entsprechend hohe Sicherheitseigenschaften. Cloud-Infrastrukturen hingegen sind potenziell anfälliger für Angriffe, was sich in einer im Vergleich geringeren funktionalen Sicherheit und Datenschutzbewertung widerspiegelt.

2.6 Vehicle-to-X-Kommunikation

Vehicle-to-X- oder Vehicle-to-Everything-Kommunikation beschreibt im Allgemeinen den Internet of Things (IoT)-Ansatz, bei dem Fahrzeuge Informationen mit anderen Akteuren in ihrer lokalen Umgebung über Messaging austauschen (s. Abbildung 2.22). Die V2X Kommunikation lässt sich, abhängig von den beteiligten Parteien, unterschiedlich kategorisieren:

- **Vehicle-to-Vehicle (V2V):** Die Fähigkeit von Fahrzeugen, miteinander zu kommunizieren. Dies ist eine der häufigsten Instanziierungen der Vehicle-to-Everything-Kommunikation.
- **Vehicle-to-Infrastructure (V2I):** Die Fähigkeit von Fahrzeugen, mit jeder Art von Verkehrsinfrastruktur zu kommunizieren. Ein oft genannter Use Case ist der bilaterale Austausch von Fahrzeugen mit Ampeln, um den Verkehrsfluss zu optimieren. Weitere Beispiele sind die automatisierte Mautabrechnung oder das Freihalten von Fahrbahnen für Einsatzfahrzeuge.
- **Vehicle-to-Pedestrian (V2P):** Die Fähigkeit von Fahrzeugen, indirekt mit Fußgängern in der Nähe zu kommunizieren, die hauptsächlich zur Vermeidung von Kollisionen verwendet wird. Fußgänger müssen mit Smartphones oder tragbaren Geräten ausgestattet sein, um an der Kommunikation teilnehmen zu können.
- **Vehicle-to-Network (V2N):** Die Fähigkeit von Fahrzeugen, mit allen Arten externer Netzwerke zu kommunizieren, z.B. um Echtzeit-Verkehrsinformationen zu erhalten, Parkplätze zu finden, OTA-Software-

Updates zu erhalten oder um Zugriff auf Multimedia-Dienste zu erhalten.

Im Jahr 2016 wurde von der 3GPP Organisation[8] eine erste Spezifikation der Cellular-V2X-Technologie (C-V2X) (s. Kapitel 2.6.1) unter Verwendung von Long Term Evolution (LTE)-Netzwerken veröffentlicht und erste V2X Services beschrieben, die mit diesem Standard ermöglicht werden sollen (s. Abbildung 2.23). Insbesondere mit der Etablierung von 5G-Netzen, deren Eigenschaften in 2.6.1 dargestellt sind, wird C-V2X immer attraktiver, da eine verbesserte Latenz und ein höherer Durchsatz hochleistungsfähige Anwendungen ermöglichen. Der Einsatz der jeweiligen Mobilfunknetze ist

[8] `https://www.3gpp.org`

Abbildung 2.22: Vehicle-to-X Kommunikationsarten [1]

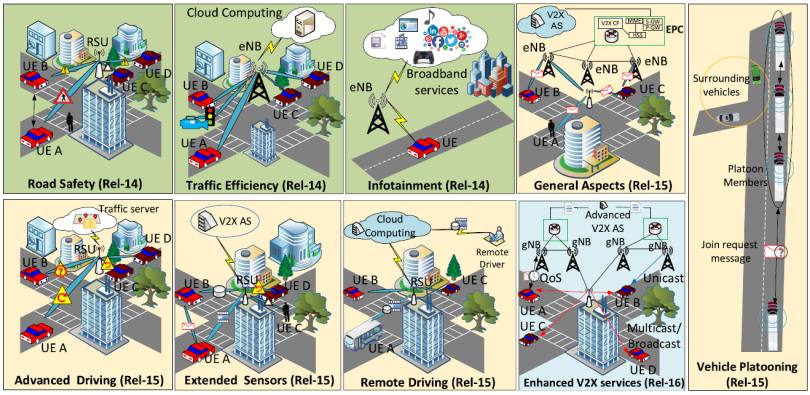


Abbildung 2.23: Zusammenfassung der V2X Service Kategorien (Release 14-16) der 3GPP Spezifikationen

sehr stark von dem im Fahrzeug verbauten Telematiksystem (s. Abbildung A.21) abhängig.

2.6.1 Cellular-V2X

Mobilfunk ist dank flächendeckender Verfügbarkeit, hoher Sicherheit und steigender Netzkapazität der aussichtsreichste Übertragungskanal für cloudbasierte Fahrzeugfunktionen. Die C-V2X Technologie setzt aktuell entweder auf die LTE- oder 5G-Technologie. Die wesentlichen Unterschiede beider Technologien werden in Abbildung 2.24 in einem Netzdiagramm hinsichtlich folgender Kriterien zusammengefasst:

- **Datenrate beim Benutzer:** Minimale erreichbare Datenrate für einen Benutzer in der tatsächlichen Netzwerkumgebung
- **Anschlussdichte:** Gesamtanzahl angeschlossener Geräte pro Flächeneinheit
- **Datenkapazität pro Flächeneinheit:** Gesamtdatenrate aller Benutzer pro Flächeneinheit

- **Maximale Datenrate:** Maximal erreichbare Datenrate pro Benutzer
- **Energieeffizienz des Netzwerks:** Anzahl an Bits, die pro Energieeinheit übertragen werden können
- **Latenz:** Dauer der Datenübertragung vom Sender zum Empfänger
- **Mobilität:** Relative Geschwindigkeit zwischen Empfänger und Sender unter bestimmten Performanzanforderungen
- **Bandbreiteneffizienz:** Verhältnis zwischen Datenübertragungsrate und Bandbreite

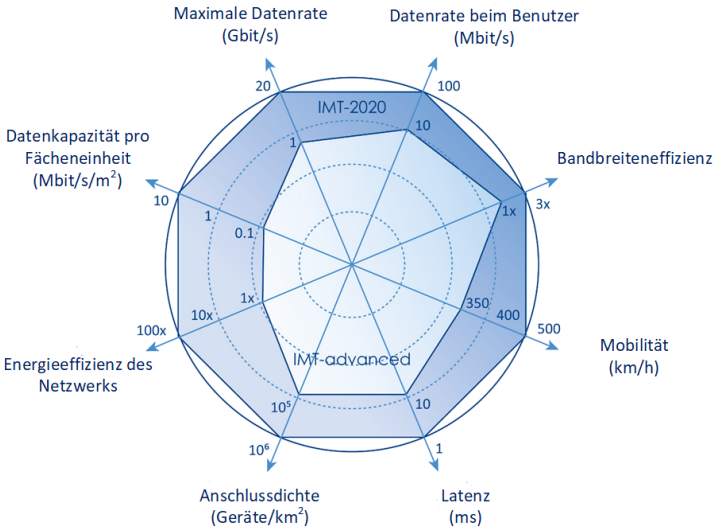


Abbildung 2.24: Vergleich der Schlüsselfunktionen von IMT-Advanced (4G) mit IMT-2020 (5G) [38]

2.6.2 Einflüsse auf die erreichbaren Datenraten im Mobilfunknetz

Die Datenrate des Mobilfunknetzes wird von verschiedenen Faktoren beeinflusst, die somit auch Einfluss auf die in die Cloud verlagerte SWC haben. Die Entfernung der Basisstationen und ihr Einflussbereich wirken sich direkt auf die Datenrate aus, da das Signal-to-Interference-plus-Noise Ratio (SINR) an den Grenzen einer Zelle abnimmt. Das SINR steht in direktem Zusammenhang mit der Datenrate und ist definiert als das Verhältnis zwischen der Leistung des Signals und der kombinierten Leistung von Störungen und Rauschen [83]. Auch die Umgebung und die Position des Fahrzeugs sind entscheidend. Ein großes Gebäude zum Beispiel begrenzt die erreichbare Datenrate. Hohe Geschwindigkeiten eines Fahrzeugs beeinflussen ebenfalls die erreichbaren Datenraten und führen zu Paketverlusten und Kommunikationsverzögerungen. Ein weiteres Problem, das bei Großveranstaltungen beobachtet werden kann, ist der Einfluss der Anzahl der Nutzer auf die Datenrate. Sobald die Kapazitätsgrenze eines Netzes erreicht wird, kann sich die Round-Trip Time (RTT) im Vergleich zur normalen Nutzung um den Faktor 1,5 bis 7 erhöhen [83].
Nutzerstatistiken aus dem deutschen Mobilfunknetz der Deutschen Telekom zeigen eine durchschnittliche Download-Datenrate von $65{,}2\,\mathrm{Mbit\,s^{-1}}$ und eine Upload-Datenrate von $13{,}9\,\mathrm{Mbit\,s^{-1}}$ [93]. Der Unterschied in der Upload- und Download-Geschwindigkeit spielt im weiteren Verlauf der Dissertation bei der Bewertung der Eignung einer Funktion für die Verlagerung in die Cloud eine Rolle.

Stromverbrauch für den Datentransfer über das Mobilfunknetz

Der Stromverbrauch für den Datentransfer über LTE wurde von Huang et al. [54] anhand von Nutzerdaten in den USA untersucht. Abbildung 2.25 zeigt die Ergebnisse als das Verhältnis zwischen Datendurchsatz R und Stromverbrauch für Up- und Download im LTE-Netz. Für die Berechnung

des Stromverbrauchs in Abhängigkeit vom Datendurchsatz beim Up- und Download werden die folgenden vereinfachten Geradengleichungen eingeführt [54]:

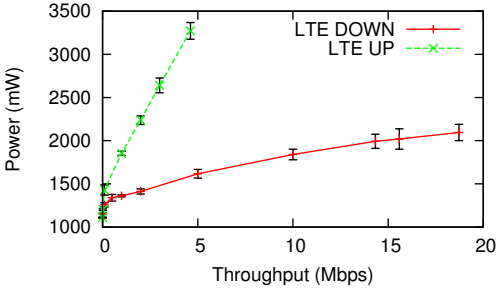


Abbildung 2.25: Energieverbrauch im LTE-Netz abhängig vom Datendurchsatz im Up- und Download [54]

$$P_{\mathrm{Trans,Up}} = \alpha_{\mathrm{Up}} R_{\mathrm{Up}} + \beta_{\mathrm{Up}} \tag{2.36}$$

$$P_{\mathrm{Trans,Down}} = \alpha_{\mathrm{Down}} R_{\mathrm{Down}} + \beta_{\mathrm{Down}} \tag{2.37}$$

α_{Up} und α_{Down} sowie β_{Up} und β_{Down} sind feste Modellparameter, die die Autoren im Rahmen der durchgeführten Messungen extrahieren konnten. Das Verhältnis $\alpha_{\mathrm{Up}}/\alpha_{\mathrm{Down}}$ beträgt 8,44 [54]. Demnach ist der Stromverbrauch der gleichen Datenmenge bei gleichem Datendurchsatz im Upload wesentlich höher als im Download. β ist der Basisstromverbrauch ohne Datendurchsatz. Durch Einsetzen der durchschnittlichen Up- und Downloaddatenraten aus Kapitel 2.6.2 ergeben sich folgende Leistungswerte:

$$P_{\mathrm{Trans,Up}} = 438,39 \frac{\mathrm{mW}}{\mathrm{Mbit\,s^{-1}}} \cdot 13{,}9\,\mathrm{Mbit\,s^{-1}} + 1.288{,}04\,\mathrm{mW} = 7{,}38\,\mathrm{W} \tag{2.38}$$

$$P_{\mathrm{Trans,Down}} = 51,97 \frac{\mathrm{mW}}{\mathrm{Mbit\,s^{-1}}} \cdot 65{,}2\,\mathrm{Mbit\,s^{-1}} + 1.288{,}04\,\mathrm{mW} = 4{,}67\,\mathrm{W} \tag{2.39}$$

2.7 Multikriterielle Entscheidungsanalyse

Die multikriterielle Entscheidungsanalyse (engl. Multiple-Criteria Decision Analysis (MCDA)) ist eine Methode der Entscheidungsfindung für jegliche Art von Alternativen und Kriterien. Der Zweck der multikriteriellen Entscheidungsanalyse ist es, den Entscheidungsprozess anhand der folgenden fünf Schritte zu strukturieren [18]:

I. Identifizierung des Entscheidungsproblems - Analyse des betrachteten Systems, Formulierung des Problems und Identifizierung der gewünschten Ziele.

II. Strukturierung des Entscheidungsproblems - Die Bestandteile des Entscheidungsproblems (Ziele, Kriterien und Alternativen) werden genau definiert. Darüber hinaus werden die Wertebereiche für die Kriterien festgelegt.

III. Erstellung des Modells - Die Komponenten des Entscheidungsproblems werden in einem System zusammengefasst, mit dem eine zielorientierte Bewertung der Alternativen möglich ist.

IV. Bewertung der Alternativen - Die Alternativen werden mithilfe des Modells bewertet.

V. Analyse der Ergebnisse.

Für die Bewertung von Alternativen aus Schritt IV stehen vielfältige Methoden zur Verfügung (s. dazu auch [115]). Das Weighted Sum Model (WSM) ist aufgrund der schnellen Implementierung weit verbreitet.

Weighted Sum Model (WSM) Das WSM bewertet die Alternativen A_i anhand der gewichteten Summe der Kriterienbewertungen. Jede Alternativlösung A_i wird hinsichtlich der definierten Kriterien C_j in Form eines Kriterienwertes a_ij beurteilt. Mithilfe eines relativen Gewichts w_j können die einzelnen Kriterien C_j nach Wichtigkeit eingestuft werden. Die finale Bewer-

tung der Alternative A_i, bezeichnet als Gesamtnutzwert $A_\mathrm{i}^{\mathrm{WSM}}$, berechnet sich schlussendlich wie in Gleichung 2.40 definiert, wobei die Kriterienwerte normiert werden müssen, bevor sie in die Alternativenbewertung einfließen.

$$A_\mathrm{i}^{\mathrm{WSM}} = \sum_{j=1}^{N} w_j a_{ij}, i \in I \tag{2.40}$$

Dabei sind alle $w_\mathrm{j} > 0$ und es gilt:

$$\sum_{j=1}^{N} w_j = 1 \tag{2.41}$$

3 Stand der Technik und Wissenschaft

3.1 Cloudbasierte Fahrzeugfunktionen

Cloudbasierte Fahrzeugfunktionen orientieren sich an Trends der Informationstechnik, die zunehmend in die Automobilbranche einfließen. Besonders das Internet der Dinge (engl. IoT), also die Vernetzung von Soft- und Hardwarekomponenten über das Internet, spielt hierbei eine zentrale Rolle. Da moderne Fahrzeuge internetfähig sind, stellt sich die Frage nach dem Mehrwert, den sie als Teil des IoT bieten. Das Forschungsthema cloudbasierter Fahrzeugfunktionen adressiert genau diese Fragestellung und wird von Milani [82] wie folgt definiert:

> **Cloud-based vehicle functions**
>
> Cloud-based vehicle functions (CBVF) are entire or parts of vehicle applications, which run temporarily or permanently in the cloud and use cloud capability (computing and/or storage resources) instead of vehicle's on-board capacity. CBVFs can use as input both information from vehicle e.g. on-board sensors and data from cloud data center. Their outputs are destined for on-board ECU functions to improve the vehicle control decision and strategy. According to this definition, functions in the cloud without influence on the vehicle on-board ECU control, are not part of CBVFs.

3.2 Deploymentmodelle cloudbasierter Fahrzeugfunktionen

Die Autoren Kovachev et al. [65] versuchen in der Arbeit „Mobile Cloud Computing: A comparison of Application Models“ eine Unterteilung von Deploymentmodellen (vgl. Definition 5.1) für mobile Cloud-Anwendungen durchzuführen. Die Autoren entwickeln Modelle, die von einer reinen Cloud-Ausführung bis hin zu unterschiedlichen Formen der Multiplizität in der Cloud reichen. Diese Kategorien wurden von Milani [82] auf den Automotive-Bereich übertragen und dort auf folgende Modelle für CBVF festgelegt:

- Nur Cloud: Die Funktion ist nur in der Cloud verfügbar und verfügt über keinerlei Fallback im Fahrzeug.
- Fallback: Die Funktion wird in der Cloud ausgeführt, es existiert aber eine Fallback Funktion im Fahrzeug-Steuergerät.
- Dupliziert: Eine Kopie der Funktion existiert in der Cloud. Die Funktion wird entweder auf dem Steuergerät oder in der Cloud ausgeführt.
- Elastisch: Die Funktion kann nahtlos zwischen dem Fahrzeug und der Cloud wechseln.

3.3 Statische und dynamische Funktionsverteilung im Fahrzeug

Die Veröffentlichung „Ontology for Vehicle Function Distribution“ von Ruhnau et al. [RtH+23] gibt einen Überblick über die statische und dynamische Funktionsverteilung (s. Definition 3.1) in der Fahrzeugdomäne. Eine Funktion wird in diesem Fall als die „Erfüllung spezieller Aufgaben wie Signalverarbeitung, Algorithmenberechnung und Steuerung von E/E-Komponenten“ definiert.

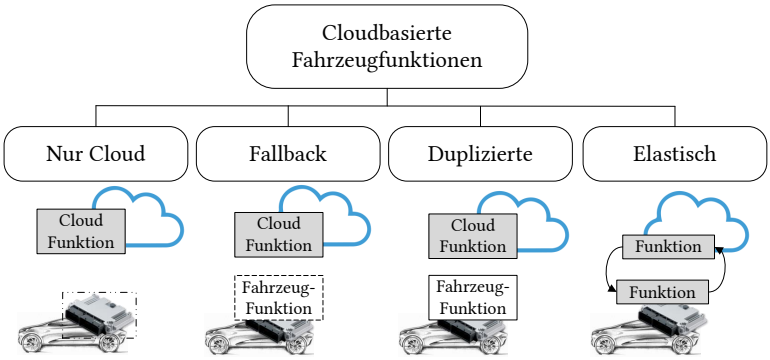


Abbildung 3.1: Arten cloudbasierter Fahrzeugfunktionen, nach [82]

Definition 3.1 Funktionsverteilung: Ergebnis des Prozesses der Verteilung von Funktionen und Erstellung einer Konfiguration. Die Funktionsverteilung setzt voraus, dass zuvor die Schritte der Funktionsallokation und des Deployments durchgeführt wurden. [RtH+23]

Jede Funktionsverteilung (s. Abbildung 3.2) definiert eine gültige Kombination aus Software und Hardware für das gesamte Fahrzeug. Sie besteht aus einem Deployment (s. Definition 3.2) und einer Allokation (s. Definition 3.3).

Definition 3.2 Deployment: Der Vorgang der tatsächlichen Bereitstellung der SWC auf einer Ausführungsplattform. Voraussetzung ist, dass die Funktionsallokation vorher abgeschlossen worden ist. [RtH+23]

Definition 3.3 Funktionsallokation: Prozess der Verteilung von Funktionen auf die Hardwarekomponenten des Systems [57].

Letzteres muss als Ergebnis eines Planungsprozesses vorhanden sein, bevor das eigentliche Deployment stattfinden kann. Sowohl das Deployment als auch die Allokation betreffen den gesamten Satz von SWCs und sind daher für jede Funktionsverteilung eindeutig, während Deployment- und Allokations-Mappings die individuellen Beziehungen beschreiben. Das heißt, eine Vielzahl an Mappings formen das Deployment bzw. die Allokation.

Das Allokations-Mapping beschreibt den geplanten Ausführungsknoten, auf dem die einzelne SWC laufen soll, auch wenn die Software bisher nicht physisch bereitgestellt ist. Das Deployment-Mapping beschreibt jedoch, wo die Artefakte, die die SWC implementieren, installiert sind und sich somit befinden. Bei diesen Artefakten handelt es sich um die eigentlichen Binärdateien oder ausführbaren Dateien der SWC. Das Deployment der Artefakte wird meist auf einem einzelnen Ausführungsknoten durchgeführt, es kann jedoch Fälle geben, in denen die SWCs auf mehreren Ausführungsknoten bereitgestellt (engl. deployed) werden. So kann etwa eine Software in der Cloud aus Gründen der Ausfallsicherheit über eine *Fallback-Version* im Fahrzeug verfügen.

Statische Funktionsverteilung

Die statische Funktionsverteilung ist der einmalig geplante Schritt innerhalb des Allokationsprozesses (Definition 3.3) bei der Entwicklung von Fahrzeugarchitekturen. Das entsprechende Deployment (Definition 3.2) erfolgt dann bei der Herstellung des Fahrzeugs und der Installation bzw. dem Flashen der Software auf den Steuergeräten oder anderen Ausführungsknoten wie der Cloud. Nach diesem Schritt ist die Funktionsverteilung mit genau einem Deployment und einer Allokation (und entsprechenden Mappings) abgeschlossen. Betrachtet man die Cloud-Deploymentmodelle (s. Kapitel 3.2) so fällt einzig das „Nur Cloud“ Modell in diese Kategorie, während alle anderen Modelle in das nachfolgende dynamische Modell der Funktionsverteilung eingeordnet werden.

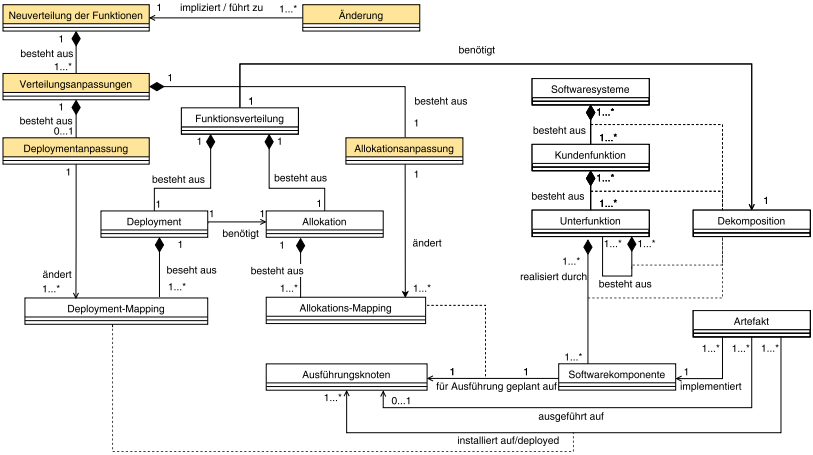


Abbildung 3.2: Ontologie für die statische und dynamische Funktionsverteilung. Ohne die gelben Elemente und ihre Beziehungen wird der statische Fall dargestellt (angepasst an [RtH+23])

Dynamische Funktionsverteilung

Bei einer dynamischen Funktionsverteilung ist der gesamte Prozess flexibler, da sich die Allokation und das Deployment von Software zur Laufzeit ändern können. Dies führt möglicherweise zu mehreren Deploymentanpassungen (engl. redeployment) mit vorhergehenden Reallokationsprozessen. Die Neuverteilung (engl. redistribution) der Softwarefunktionen wird durch eine Änderung ausgelöst. Diese Änderung könnte entweder eine Änderung der Umgebungsbedingungen, z.B. die Verfügbarkeit von Bandbreite für den Up- und Download zur Cloud, oder die Veränderung von Software zur Laufzeit sein. Die Neuverteilung besteht aus einer oder mehreren einzelnen Verteilungsanpassungen. Eine Verteilungsanpassung besteht aus genau einer Allokationsanpassung, aber nicht zwingend aus einer Deploymentanpassung. Dies ermöglicht den Fall, dass verschiebbare SWC gleichzeitig auf mehreren Ausführungsknoten verfügbar (engl. deployed) sind (bspw. in Form einer Backup-Version) und daher eine Funktionsneuverteilung nicht unbedingt ei-

ne Deploymentanpassung erfordert. Die notwendige Allokationsanpassung ändert jedoch den Ausführungsknoten, auf dem die SWC ausgeführt werden soll. Dies führt schließlich zu einer geänderten Funktionsverteilung, sobald die neue Allokation aktiviert ist.

3.3.1 Frameworks für die Funktionsverteilung

Die Veröffentlichung „Elastic Service Provision for Intelligent Vehicle Functions“ von Stefan Kugele et al. [72] versucht die beschränkte Rechenleistung des Fahrzeugs mithilfe eines konzeptuellen Frameworks zu lösen, das in der Lage ist, die E/E-Architektur des Fahrzeugs in die Cloud zu erweitern. Fahrzeugfunktionen sollen in einem entfernten Rechenzentrum genauso wie im Fahrzeug selbst ausgeführt werden können. Dadurch wird ein hohes Maß an Standorttransparenz erreicht: Fahrzeugdiensten soll es gleichgültig sein, ob sich ihre Gegenstellen auf demselben Host-Rechner oder einem entfernten Server befinden. Hierfür entwickeln die Autoren neben einem Ansatz zur Verbindung des Fahrzeugs mit der Cloud auch Strategien für die Verteilung der Funktionen zwischen Fahrzeug und Cloud abhängig vom Betriebszustand des Fahrzeugs. Die Autoren setzen auf Docker[1] für die Containerisierung von Diensten in der Cloud und setzen auf eine Kommunikation basierend auf der serviceorientierten Publish/Subscribe Technik. Abhängig von verfügbarer Rechenpower im Fahrzeug, der Fahrzeuggeschwindigkeit oder der verfügbaren Batteriekapazität entscheidet das System, wo Dienste ausgeführt werden sollen. Die Verbindungsqualität der Luftschnittstelle wird dabei außen vor gelassen. Der genaue Inhalt eines Dienstes, der elastisch zwischen Fahrzeug, Steuergerät oder Cloud verschoben werden soll, ist nicht von entscheidender Rolle, sondern stellt ein Programm dar, das in einem gewissen Rahmen Daten zwischen zwei Endpunkten schicken will.

[1] https://www.docker.com/

Die Änderungen des Betriebszustandes des Fahrzeugs werden auch von Banijamali et al. in der Veröffentlichung „Kuksa: Self-Adaptive Microservices in Automotive Systems“ aus dem Jahr 2020 betrachtet [14]. Die Autoren präsentieren ein Microservice-Framework vom Fahrzeug bis in die Cloud (s. Abbildung 3.3), das in der Lage ist, sich den Fahrzeuggegebenheiten (z.B. sich ändernde Geschwindigkeiten) anzupassen. So sollen Funktionen (oder Microservices) in der Cloud durch Parameteranpassungen in einer Feedback-Schleife (mittels des sogenannten *Autonomic Controllers*) kontinuierlich angepasst werden, um zu jedem Zeitpunkt in der Lage zu sein, Echtzeitentscheidungen treffen zu können. Diese möglichen Anpassungen werden aus theoretischer Sicht betrachtet und beinhalten beispielsweise ein Upgrade des Antriebsstrangs des Fahrzeugs aufgrund eines angehängten Wohnwagens oder das Hinzufügen von neuen Sicherheitsanforderungen aufgrund schlechter Wetterverhältnisse. Das Framework wird anhand eines Video-Streaming Use Case in einem Laboraufbau mit einem mobilen Roboter getestet. Die adaptive Anpassung des Microservices wurde dabei anhand der Kriterien Zeit für die Rekonfiguration des Services an sich ändernde Gegebenheiten und der Qualität des Services in Form von Bildrate und Bildqualität beurteilt. Beide Parameter können zudem unterschiedlich gewichtet werden. Die Ergebnisse der Arbeit zeigen, dass ein adaptiver Microservice sinnvoll ist, um die Qualität des Services deutlich zu verbessern. Allerdings zeigt sich auch, dass diese Anpassungsfähigkeit eine weitere Komplexität zum Gesamtsystem hinzufügt, die nicht immer sinnvoll und demnach stark vom Use Case abhängig ist.

3.3.2 Use Cases cloudbasierter Applikationen

Ashok et al. [8] beschreiben 2015 ein System, das in der Lage ist, dem Fahrzeug Cloud-Rechenkapazitäten zur Verfügung zu stellen. Dabei werden die Konzepte der Serviceorientierung (s. Kapitel 2.3.2) verwendet. Das bedeutet, dass die Cloud-Funktionalität dem Fahrzeug als Service angeboten wird. Die betrachteten Use Cases für eine Verlagerung sind aus dem Bereich Compu-

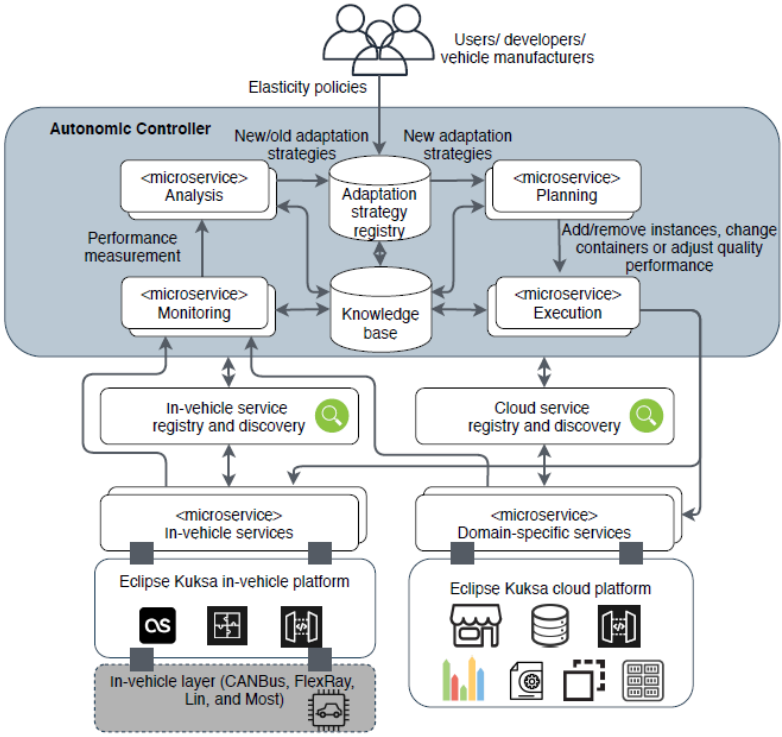


Abbildung 3.3: KUKSA Framework [14]

ter Vision, da dieser Bereich aufgrund der großen Datenmengen als sehr rechenintensiv gilt.

1. Ein System zur Erkennung von Handgesten, bei dem ein Benutzer Handbewegungen vollzieht (Hand bewegen, Finger nach oben/unten zeigen etc.), die von einer Kamera aufgezeichnet werden. Die Gesten werden den entsprechenden Ausgaben für die Nutzung im Auto zugeordnet. Zum Beispiel entspricht Winken dem Öffnen der Karten-

App, während das Bewegen des Zeigefingers nach oben/unten dem Auslösen eines Telefonanrufs entspricht.

2. Ein Ampel- und Verkehrszeichenerkennungssystem, bei dem die Kamera (auf die Straße gerichtet) den Zustand der Ampel und Verkehrsschilder in ihrem Sichtfeld erkennt.

Beide Applikationen wurden nicht in einem Fahrzeug integriert, sondern prototypisch als Android Apps auf einem Smartphone mit Cloud-Anbindung umgesetzt. Für beide Systeme wurde untersucht, inwiefern eine Verlagerung der rechenintensiven Schritte der Bilderkennung in die Cloud die Reaktionszeit der Applikation gegenüber einer vollständigen lokalen Ausführung auf einem Smartphone verbessert oder verschlechtert. Die Ergebnisse zeigen eine mindestens dreifach schnellere Ende-zu-Ende-Reaktionszeit der Applikation bei einer Cloudverlagerung der Bilderkennungsschritte.

Deng et al. [33] beschreiben in ihrem Paper aus dem Jahr 2020 die Möglichkeit, cloudbasierte Fahrzeugfunktionen mithilfe der Cloud-Infrastruktur von Amazon Web Services (AWS) umzusetzen. Es wird in einem ersten Schritt gezeigt, über welche Infrastruktur und Architektur die Cloud verfügen muss. Die Cloud soll die zentrale Schnittstelle für vernetzte Fahrzeuge und andere Verkehrsteilnehmer, die Verkehrsinfrastruktur (Road Side Units) und Services (Wettervorhersagen) aus dem Internet darstellen. Die Autoren haben diese Infrastruktur in AWS umgesetzt und eine prototypische Verkehrsaufkommensüberwachung in Echtzeit auf Basis von Fahrzeugdaten in der Cloud implementiert. Fahrzeuge übertragen Informationen wie die aktuelle Geschwindigkeit und den Standort an den Datenbankdienst in der Cloud (NoSQL-Datenbank). Anschließend wird auf Basis dieser Daten eine Applikation zur Berechnung der Durchschnittsgeschwindigkeit auf einzelnen Streckenabschnitten ausgeführt und an Abonnenten dieser Applikation entsprechend ihres aktuellen Standorts für vorausliegende Abschnitte verteilt. Der Hauptfokus der Arbeit lag bei der anschließenden Untersuchung der Cloud-Applikation hinsichtlich der Paketumlaufzeit zwischen Fahrzeug, Cloud und zurück.

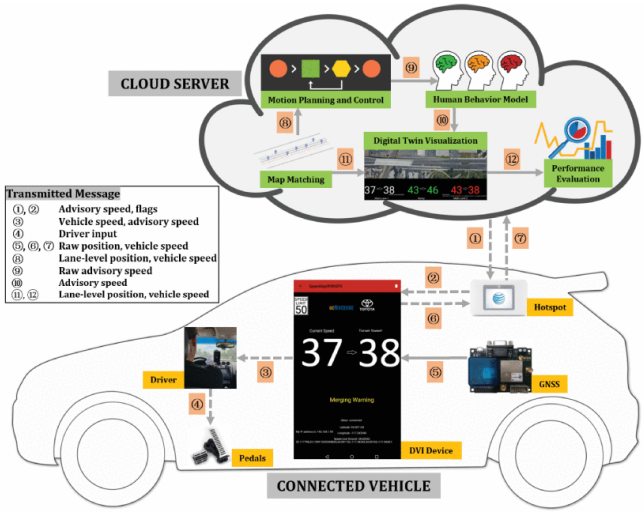


Abbildung 3.4: Architektur des cloudbasierten Fahrerassistenzsystems [125]

Die Arbeiten an einem cloudbasierten Fahrerassistenzsystem von Wang et al. [125] aus der Veröffentlichung „A Digital Twin Paradigm: Vehicle-to-Cloud Based Advanced Driver Assistance Systems“ beschreiben ein System, das dem Fahrer Vorschläge für die Einstellungen von Aktoren mittels Driver-Vehicle-Interface (DVI) sichtbar macht, diese aber nicht automatisch (bspw. durch direktes Schreiben der Werte auf den CAN Bus) für den Fahrer ausführt. Der Fahrer soll diese Vorgaben bestmöglich durch selbständiges Betätigen der Aktoren (in diesem Fall durch die Bedienung des Fahrpedals) einhalten. Die Architektur des Fahrerassistenzsystems mit den einzelnen Komponenten im Fahrzeug sowie in der Cloud und dem Informationsfluss ist in Abbildung 3.4 dargestellt.

Auf der Fahrzeugseite besteht das System im Wesentlichen aus der DVI Einheit, die Sensorinformationen (Fahrzeuggeschwindigkeit, Position) sammelt und die Nachrichten des Fahrerassistenzsystems in der Cloud dem Fahrer zur Verfügung stellt. Alle Berechnungen des Systems werden in der

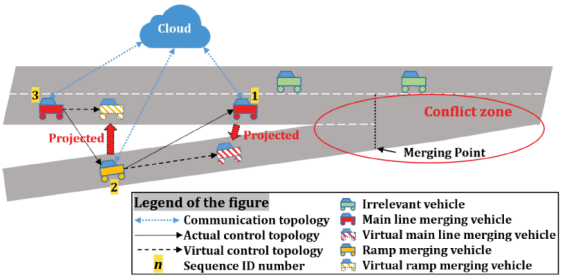


Abbildung 3.5: Use Case der Rampenzusammenführung von Straßen für den Test des cloudbasierten Fahrerassistenzsystems [125]

Cloud ausgeführt, in der digitale Nachbildungen der realen physikalischen Welt erstellt werden. Dazu gehören unter anderem eine digitale Karte des Testfelds und ein Algorithmus, der anhand der gemessenen Position des realen Fahrzeugs die Position des Fahrzeugs auf der digitalen Karte anpasst. Entscheidend ist die Bewegungsplanung und -steuerung, die auf Basis des Ist-Zustandes (Geschwindigkeit etc.) des Ego-Fahrzeugs eine Bewegungsabfolge in Form von „rohen“ Geschwindigkeitsvorgaben berechnet. Da der reale Fahrer des Fahrzeugs nicht in der Lage ist, die „rohen“ Vorgaben genau umzusetzen, wurde ein Fahrer-Verhaltensmodell entwickelt, das den menschlichen Fehler kompensieren soll. Die tatsächlich an das physikalische System übertragenen Vorgaben sind der Ausgang dieses Modells. Schlussendlich wird das Verhalten des Ego-Fahrzeugs und anderer Fahrzeuge digital dargestellt und kann auch direkt anhand von Kriterien wie Energieverbrauch, Beschleunigungswerten etc. in der Cloud ausgewertet werden.

Die Effektivität des entwickelten Systems haben die Autoren anhand eines realen Praxisbeispiels einer Straßenzusammenführung in Riverside in den USA getestet (Abbildung 3.5 zeigt den Test Use Case vereinfacht). Dabei wurde eine spezielle Rampenzusammenführung gewählt, bei der die beteiligten Fahrzeuge bis zuletzt keine Sicht auf die jeweils andere zusammenführende Fahrbahn haben. Ein Fahrzeug auf der Hauptspur und ein Fahrzeug auf der zusammenführenden Rampe sollen zusammengeführt werden. Das Manöver

wurde auf digitaler Ebene in der Cloud zu einem Fahrzeug-Folge-Problem vereinfacht, indem man ein virtuelles Fahrzeug der jeweils anderen Fahrbahn mit denselben Parametern (Geschwindigkeit und Distanz bis zum Zusammenführungspunkt) auf der anderen Fahrbahn gespiegelt hat. Anhand dieses Beispiels konnten die Autoren schlussendlich zeigen, dass das cloudbasierte System im Testszenario trotz Paketverlusts und Latenz zwischen den realen Fahrzeugen und der Cloud funktioniert und die Fahrzeuge ohne drastische Geschwindigkeitsveränderungen durch das cloudbasierte Assistenzsystem zusammengeführt werden können.

Chao Yang et al. beschreiben einen Use Case, der ebenfalls das Ziel der Energieoptimierung (in diesem Fall von Stadtbussen) mittels eines cloudbasierten Frameworks verfolgt [134]. Das Framework sammelt Positions- und Geschwindigkeitsdaten aus Stadtbussen mit Plug-In-Hybrid-Antrieben und versucht, diese zunächst in verschiedene Fahrstreckenprofile einzuteilen (Cluster). Mithilfe dieses Streckenprofils und aktueller Brems- und Beschleunigungsdaten aus dem Fahrzeug ermittelt ein Online-Algorithmus in der Cloud eine energieoptimierte Strategie für die Aufteilung des Drehmoments des Elektromotors und des Verbrennungsmotors für den weiteren Verlauf auf der Busroute. Der weitere Verlauf der Route wird überwacht und kontinuierlich in einer Regelschleife eine neue Aufteilung der Werte berechnet. Das Framework wird in einem Hardware-in-the-Loop (HiL)-Test überprüft. Die Aussagekraft der Arbeit wird von den Autoren durch eine Analyse der jeweils durch den Cloud-Algorithmus berechneten Drehmomente der Motoren auf der Busroute bezüglich der Energieeffizienz in einem Wirkungsgradkennfeld dargelegt. Dabei zeigen die Autoren, dass das von Ihnen entwickelte Verfahren einen, verglichen mit dem bisherigen regelbasierten Verfahren im Motorsteuergerät, geringeren Energieverbrauch auf der beispielhaften Route aufweist.

3.3.3 Fazit zu den Anwendungsfällen und Frameworks aus der Wissenschaft

Die vorgestellten Veröffentlichungen zeigen, dass die grundsätzliche Idee, Funktionen in die Cloud zu verlagern oder diese zumindest durch Wissen aus der Cloud anzureichern, bereits wissenschaftlich diskutiert und exemplarisch umgesetzt wurde. Eine elastische Applikation, wie in Kapitel 3.2 beschrieben, wird durch das Framework von Stefan Kugele et al. beschrieben. Dabei werden aber zum einen Parameter, die einen Einfluss auf die Verlagerung haben (wie die Mobilfunkverbindung), vernachlässigt und zum anderen nicht ein Feature (s. Definition 1.6) verlagert, sondern beispielhafte nicht weiter spezifizierte Dienste betrachtet, die Daten zwischen Cloud und Fahrzeug austauschen. Das KUKSA Framework von Banijamali et al. beschreibt die selbstanpassende Fähigkeit von Microservices in der Cloud, um eine hohe Verfügbarkeit und Qualität sicherzustellen.
Die Veröffentlichungen zu den Frameworks und die Erkenntnisse aus den Anwendungsfällen für cloudbasierte Fahrzeugfunktionen bieten gemeinsam eine Grundlage für die prototypische Umsetzung cloudbasierter Use Cases.

3.3.4 Wissenschaftliche Methode zur Bewertung des Ausführungsortes einer Fahrzeugfunktion

In ihrer Dissertation „Suitability Analysis Methodology for Cloud-based Vehicle Functions“ entwickelt Farzaneh Milani [81] eine Methodik zur Bewertung der Eignung von Fahrzeugfunktionen für die Migration in Cloud-Umgebungen. Diese Methodik, bekannt als „Suitability Analysis Methodology“ (SAM) (s. Abbildung 3.6), stellt einen Ansatz dar, um die Machbarkeit (engl. Feasability) und Eignung (engl. Suitability) der Verlagerung von Fahrzeugfunktionen in die Cloud systematisch zu untersuchen. Die Machbarkeit der Auslagerung im ersten Teil der Methodik beschreibt vier relevante Kriterien:

- Ausreichende Speicher- und Rechenressourcen
- Einhaltung von Deadline Beschränkungen
- Einhaltung von Safety- und Securitybeschränkungen
- Ausreichende Energie für die Ausführung von Funktionen und die Datenübertragung

Die Eignung der beiden Ausführungsorte Fahrzeug und Cloud in Form eines Bewertungs-Scores wird anhand von fünf Kriterien ausgemacht:

- Kapazitätsbedarf
- Datenabhängigkeit
- Reaktionszeit
- Sicherheitsmaßnahmen
- Energieverbrauch

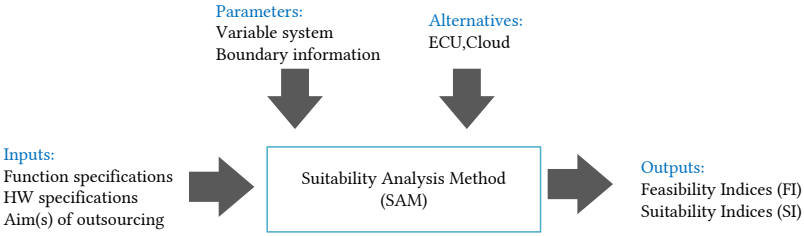


Abbildung 3.6: Suitability Analysis Methodology (SAM) für die Funktionsverlagerung in die Cloud [81]

Weiterer Bestandteil der Methodik ist ein Partitionierungsalgorithmus, der dazu dient, die optimale Aufteilung von Fahrzeugfunktionen zwischen On-Board-Systemen und der Cloud zu bestimmen. Dieser Algorithmus berücksichtigt sowohl technische als auch betriebliche Anforderungen und ermöglicht eine Entscheidungsfindung bei der Planung und Implementierung cloudbasierter Funktionen.

Bewertung von SAM

Die Suitability Analysis Methodology (SAM) verfügt nicht über ein einheitliches Prozessschaubild mit einer nachvollziehbaren Vorgehensweise zur Bewertung der Cloud-Eignung von Fahrzeugfunktionen. Dies zeigt sich auch daran, dass die Arbeit Kriterien anführt und diese ausführlich beschreibt, diese anschließend aber nicht mehr in der schlussendlichen Realisierbarkeit bzw. Eignung miteinbezieht.

Zudem fehlen wesentliche Kriterien wie eine umfassende Wirtschaftlichkeitsanalyse, beispielsweise in Form einer Total Cost of Ownership (TCO)-Betrachtung, die sowohl die langfristigen Kosten als auch mögliche Einsparpotenziale der Cloud-Ausführung berücksichtigt. Ebenso bleibt das Potenzial der Cloud für lernende Systeme und künstliche Intelligenz (KI) unberücksichtigt, insbesondere in Bezug auf adaptive Optimierungen, datengetriebene Verbesserungen und verteiltes Lernen, die für moderne Fahrzeugfunktionen von großer Bedeutung sind.
SAM bietet dennoch die Grundlage für den Prozess zur Identifikation cloudbasierter Fahrzeugfunktionen in Kapitel 4.

3.3.5 Stand der Wissenschaft hinsichtlich der Ziele einer Funktionsverlagerung

Die dynamische Verteilung von Smartphone Applikationen (s. auch Kapitel 3.3) zwischen Cloud und lokalem Endgerät ist Untersuchungsbestandteil des Journalbeitrags „Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions“ von Liu et al. [77] aus dem Jahr 2015. Im Rahmen dieses Beitrags wurden Ziele einer solchen Verteilung im Mobile Cloud Computing (MCC) definiert:

- **Verteilung auf verschiedene Cloud Server**: Durch die Verteilung von Funktionen auf verschiedene Server kann die Verfügbarkeit der Funktion erhöht werden.
- **Reduzierung der Speicherbeschränkungen**: Die Speicherbeschränkungen eines lokalen Geräts können durch eine Verlagerung umgangen werden.
- **Dynamische Updates von Funktionen**: Die Funktion kann in der Cloud Updates erhalten, ohne dass das lokale Endgerät davon betroffen sein muss (bspw. durch Ausschalten).

Huaming Wu [131] und Karthik Kumar [74] haben neben diesen Zielen insbesondere die Möglichkeiten der Energieeinsparung und zeitliche Performanzsteigerungen als Ziele identifiziert und beschreiben diese wie folgt:

- **Zeitliche Performanzsteigerung**: Die Gesamtzeit für die Remote-Ausführung ist die Summe der Ausführungszeit in der Cloud (t_{ExeCloud}) und der Übertragungszeit (t_{Trans}), um die Datenmenge (D) zwischen der Cloud und dem Endgerät auszutauschen. Die Remote Ausführung spart Zeit, wenn die Ausführungszeit auf einem lokalen End-

gerät (t_{ExeMob}) größer ist, als die aufgewendete Zeit für die Remote-Ausführung:

$$t_{\mathrm{ExeMob}} > t_{\mathrm{ExeCloud}} + t_{\mathrm{Trans}} > t_{\mathrm{ExeCloud}} + \frac{D}{B} \tag{3.1}$$

B stellt dabei die Datenrate zwischen Endgerät und Cloud dar. Eine hohe Ausführungsgeschwindigkeit in der Cloud, eine geringe Datenmenge zwischen Endgerät und Cloud und eine hohe Datenrate sprechen demnach für eine Ausführung in der Cloud.

- **Reduzierung des Energieverbrauchs**: Durch die Verlagerung von Funktionen kann der Energieverbrauch lokaler Endgeräte gesenkt werden. Das Endgerät benötigt für Berechnungen P_{Mob}, im Leerlauf P_{IdleMob} und für die Datenübertragung P_{Trans}. Eine Energieeinsparung durch Auslagerung ist nur dann möglich, wenn folgende Bedingung gilt:

$$P_{\mathrm{Mob}} \cdot t_{\mathrm{exeMob}} > P_{\mathrm{IdleMob}} \cdot t_{\mathrm{ExeCloud}} + P_{\mathrm{Trans}} \cdot \frac{D}{B} \tag{3.2}$$

Das Paper mit dem Titel „A partial offloading technique for wireless mobile cloud computing in smart cities“ von Mazza et al. [78] behandelt den Trade-off zwischen den genannten Zielen Energieverbrauch und zeitlicher Performanz für die Funktionsverlagerung. Im Rahmen des Beitrags wird eine Berechnungsaufgabe dann in die Cloud verlagert, wenn die Ausführungskosten (Zeit, Energie) auf dem lokalen Endgerät bedeutend höher sind als in der Cloud. Für diese Entscheidung wird eine Kostenfunktion herangezogen, die beide genannten Faktoren beinhaltet und somit optimal berechnen soll, wie viele Aufgaben in die Cloud verlagert werden sollen. Kumar et al. [74] schlagen vor, den Trade-Off zwischen Rechenanforderungen und der Transferdatenmenge (D) als Entscheidungsmerkmal zu betrachten. Die Autoren haben dabei zwei beispielhafte Applikationen untersucht. Ein Schachcomputer benötigt eine hohe Rechenpower bei gleichzeitig geringem Datentransfer

zwischen Cloud und Endgerät. Eine solche Applikation ist entsprechend der Autoren für eine Verlagerung geeignet. Das Gegenstück stellt die Funktion für einen inhaltsbasierten Bildabruf (engl. content-based image retrieval) dar. Eine solche Funktion fordert einen hohen Datentransfer der Bilder bei gleichzeitig moderatem Rechenaufwand (die Feature-Erkennung innerhalb der Bilder erfolgt direkt nach der Erstellung des Bildes). Eine solche Funktion sollte demnach immer lokal ausgeführt werden. Die Autoren haben den Trade-Off entsprechend Abbildung 3.7 dargestellt.

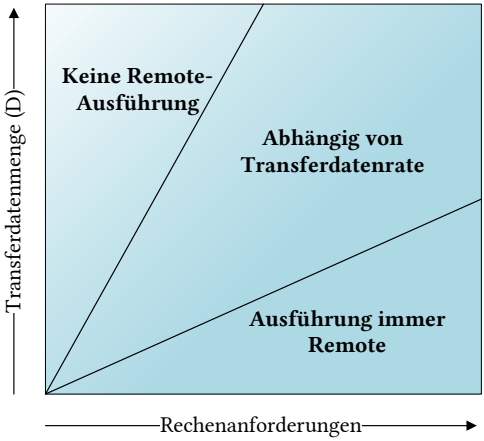


Abbildung 3.7: Trade-Off zwischen Transferdatenmenge und Rechenanforderungen bei einer Funktionsverlagerung im Bereich des Mobile Cloud Computing [74]

3.4 Regelungsstrategien für HLK-Systeme

Im Bereich der Stadtbusse (s. Kapitel 2.4.1) hat sich die Aufteilung des vollständigen HLK-Systems in Subsysteme durchgesetzt. Das zu regelnde System – genauer gesagt die zu regelnde Anlage – wird dabei in die Fahrzeugkabine sowie das HLK-System (Dachklimaanlage bzw. Wärmepumpe) unterteilt, welches die thermischen Bedingungen in der Kabine regelt. Dadurch ent-

steht eine Kaskadenregelung (s. Abbildung 3.8), die zum einen eine höhere Genauigkeit ermöglicht und zum anderen für eine bessere Koordinierung der einzelnen Regelkreise sorgt, indem sichergestellt wird, dass zu einem bestimmten Zeitpunkt immer nur ein Regelkreis für eine Regelgröße in Betrieb ist [84].

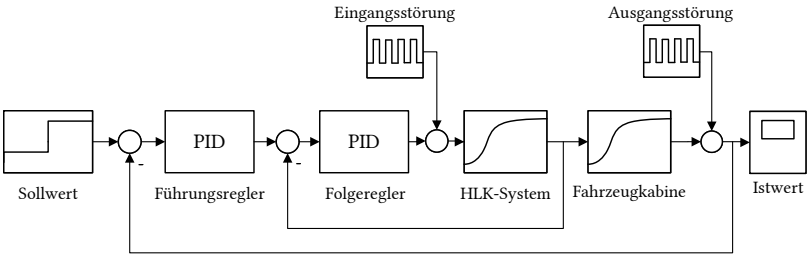


Abbildung 3.8: Typische Kaskadenregelung der Klimatisierung im Stadtbus

Die Kaskadenregelung ist so aufgebaut, dass der äußere Regelkreis den optimalen thermischen Energieeintrag für die Fahrzeugkabine berechnet, wobei dieser thermische Energieeintrag $\dot{Q}_{\mathrm{Soll}}$ von der Dynamik und den Beschränkungen des HLK-Systems abhängt. Der innere Regelkreis regelt das HLK-System so, dass der vom äußeren Regelkreis vorgegebene Wärmestrom $\dot{Q}_{\mathrm{Soll}}$ mit dem tatsächlichen Wärmestrom des HLK-Systems, $\dot{Q}_{\mathrm{HLK}}$, möglichst genau übereinstimmt.

Sowohl für den äußeren Regelkreis der Kaskadenregelung als auch für den inneren Regelkreis stehen aus technischer Sicht vielfältige Umsetzungsmöglichkeiten für die einzelnen zu regelnden Größen zur Auswahl. PID-Regler (s. Kapitel 2.1.3) sind im Bereich von HLK-Systemen ein weitverbreiteter Regelkreismechanismus. Allerdings stößt ein solcher Regler für Multiple Input Multiple Output (MIMO)-Systeme sowie für Systeme mit großer Nichtlinearität an Grenzen.
Regelbasierte Algorithmen, wie der in Kapitel 2.1.2 beschriebene Fuzzy-Regler, eignen sich besonders für Prozesse, bei denen kein exaktes Modell

erstellbar ist und daher vage oder intuitive Einschätzungen zur Modellierung herangezogen werden müssen [85]. Das Potenzial solcher Systeme im Bereich von HLK-Anwendungen wurde unter anderem von Abuhussain et al. [2], Pasha et al. [94] und Khalid et al. [63] untersucht. Fuzzy-Regler zeigten sich darin besonders geeignet, Unsicherheiten zu handhaben und gleichzeitig eine robuste Regelgüte zu erreichen. Die Wahl des Fuzzy-Regelansatzes erfolgte in allen Fällen aufgrund typischer Herausforderungen in HLK-Systemen: schwer modellierbares Nutzerverhalten, nichtlineare Dynamiken, externe Störgrößen sowie variierende Nutzeranforderungen. Durch die regelbasierte Struktur konnten Unsicherheiten (z. B. subjektives Temperaturempfinden) kompensiert und dynamische Umgebungsbedingungen flexibel berücksichtigt werden. In einem Vergleich zwischen Fuzzy- und PID-Reglern belegten Pasha et al. [94] eine signifikant geringere Überschwingung sowie ein schnelleres Einschwingverhalten beim Fuzzy-System. Khalid et al. [63] entwickelten darüber hinaus einen adaptiven Fuzzy-Regler, der sich dynamisch an wechselnde Nutzerprofile und Umweltbedingungen anpassen konnte.

Das Potenzial der modellprädiktiven Regelung (s. Kapitel 2.1.1) im Bereich von HLK-Systemen wurde unter anderem in den Veröffentlichungen von Eckstein et al. [36], He et al. [50] sowie Yan et al. [133] bewiesen. Dabei konnte der MPC seine Stärken in Form von

- Guter Eignung für MIMO-Systeme
- Expliziter Behandlung von Störgrößen
- Berücksichtigung von Beschränkungen auf Stellgrößen und deren Änderungsraten

entfalten. Die typischerweise hohen Zeitkonstanten von HLK-Systemen wirken sich zudem positiv auf den Einsatz von modellprädiktiven Regelverfahren aus. Aufgrund der trägen Dynamik genügt ein grobes zeitliches Abtastintervall, wodurch sich der Rechenaufwand reduziert. Gleichzeitig wirken sich Modellungenauigkeiten weniger kritisch auf die Regelgüte aus und die Optimierung kann energieeffiziente, kontinuierliche Stellgrößenver-

läufe erzeugen.
Dennoch befindet sich der industrielle Einsatz weiterhin im Anfangsstadium. Ein wesentlicher Hemmnisfaktor ist der hohe Aufwand für die Erstellung eines präzisen Anlagenmodells sowie die aufwendige Parametrierung des Reglers. Insbesondere bei komplexen Systemen – etwa großen Nichtwohngebäuden mit mehreren Zonen, variablen Luftvolumenströmen und thermischen Speichern – kann ein MPC auf mehrere hundert bis über tausend Zustände und Parameter angewiesen sein [89], was im Vergleich zu klassischen PID-Reglern mit deutlich höheren Anforderungen an Datenverfügbarkeit, Modellgenauigkeit und Rechenleistung einhergeht.

Zur Reduktion dieses Modellierungsaufwands werden zunehmend Verfahren auf Basis künstlicher neuronaler Netze eingesetzt [4]. Diese ermöglichen es, die zeitintensive physikbasierte Modellierung zu umgehen, welche tiefgehendes Verständnis der thermischen Systemdynamik erfordert.

In den vergangenen Jahren hat sich Reinforcement Learning (RL) aus dem Bereich der künstlichen Intelligenz (KI) als vielversprechender Ansatz zur Regelung von HLK-Systemen hervorgetan. Als modellfreier Lernansatz ist RL besonders geeignet für hochgradig nichtlineare und dynamische Systeme, bei denen klassische modellbasierte Regler wie PID oder MPC an ihre Grenzen stoßen. Durch Interaktion mit der Umgebung lernt ein RL-Agent, langfristig optimale Steuerstrategien zu entwickeln – ohne dass ein explizites physikalisches Modell erforderlich ist.

Wei et al. [126] zeigten in einer simulierten Bürogebäudeumgebung, dass ein auf Deep Q-Learning basierender Agent den Energieverbrauch gegenüber einem regelbasierten Baseline-Regler um 22 % senken kann, während die Komfortkriterien nach ASHRAE-Standard[2] eingehalten werden.
Li et al. [76] stellten einen Deep RL-basierten Regler für das Klimasystem eines Elektrobusses vor. Als konventioneller Vergleichsmaßstab dient

[2] https://www.ashrae.org/technical-resources/bookstore/standard-55-thermal-environmental-conditions-for-human-occupancy

ein PID-Regler. Die Ergebnisse zeigen, dass der Deep RL-Regler den HLK-Energieverbrauch im Vergleich zum PID-Regler um 7,3 % reduzieren kann.

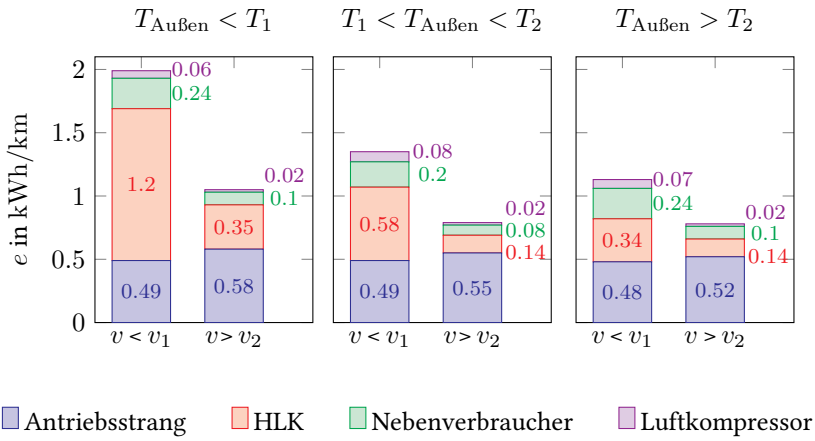
Trotz vielversprechender Simulationsergebnisse wird RL bislang kaum in realen HLK-Systemen eingesetzt. Die Gründe liegen primär in langen Trainingszeiten, fehlender Sicherheit während des Lernens sowie der begrenzten Übertragbarkeit von simulierten auf reale Systeme. Darüber hinaus erschweren mangelnde Interpretierbarkeit und die eingeschränkte Generalisierbarkeit den praktischen Einsatz. Aktuell findet RL vorwiegend in simulationsgestützten Architekturen Anwendung.

3.5 Beitrag der Klimatisierung auf den Gesamtenergieverbrauch eines BEB

Bei niedrigen Außentemperaturen kann der Beitrag der Klimatisierung im BEB den energetischen Beitrag des Antriebsstrangs übertreffen. Die Veröffentlichung von Rösch et al. liefert eine tiefere Analyse auf Basis von Realdaten aus drei Solo-BEB aus dem laufenden Betrieb über den Zeitraum eines gesamten Jahres [RRt+23]. Die Daten wurden über das Jahr gemittelt und anschließend in drei verschiedene Szenarien abhängig von der Außentemperatur unterteilt. Dabei konnte auf die Daten der VDV-236 Vorschrift zurückgegriffen werden, die Kenngrößen für Kühl- und Heizbetrieb eines Stadtbusses abhängig von Fahrzeuginnen- und Außentemperatur vorgibt [20]. Innerhalb der drei Szenarien wurden die Daten erneut in die drei definierten Geschwindigkeitsprofile des Standardized on Road Test Cycles (SORT) [120] unterteilt.

Bei einer Außentemperatur unterhalb von $10\,^{\circ}\mathrm{C}$ und einer Durchschnittsgeschwindigkeit unter $15\,\mathrm{km\,h^{-1}}$ (SORT 1) liegt der Beitrag der Klimatisierung bei $1{,}2\,\mathrm{kW\,h\,km^{-1}}$ (60 % des Gesamtenergieverbrauchs). Dieser Beitrag sinkt bei steigender Temperatur und steigender Geschwindigkeit des Fahr-

zeugs, da die Klimatisierung eines Fahrzeugs nicht von der zurückgelegten Distanz, sondern von der Zeit abhängt (s. Abbildung 3.9).

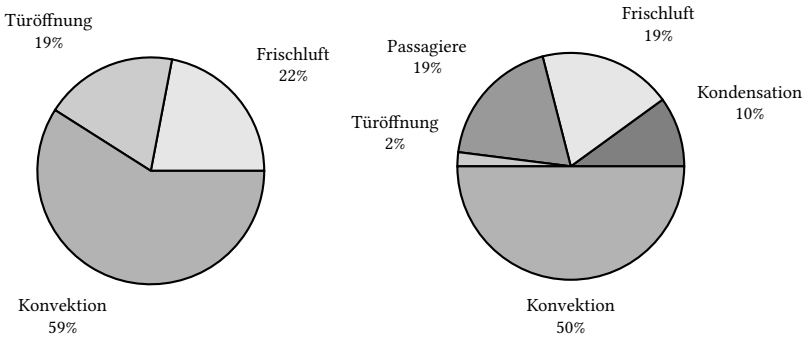


$T_1 = 10\,°\text{C}, T_2 = 25\,°\text{C}, v_1 = 15\,\text{km}\,\text{h}^{-1}, v_2 = 21\,\text{km}\,\text{h}^{-1}$
Die Geschwindigkeitswerte basieren auf den Standardized on Road Test Cycles (SORT) 1-3 [120]
Die Temperaturwerte basieren auf den in der VDV-236 [20] definierten Temperaturbereichen

Abbildung 3.9: Energieverbrauch einzelner Fahrzeugkomponenten im BEB (Solobus) [RRt+23]

Einflussfaktoren auf den Heiz- und Kühlbedarf

Die Analyse der Verbrauchsstruktur des Heiz- und Kühlbedarfs von Jefferies et al. [61] zeigt, dass in Stadtbussen die kombinierten Anteile der Konvektions- und der solaren Strahlungswärmelast maßgeblich sind: Sie machen in Kühlszenarien etwa 59 % und in Heizszenarien etwa 50 % der von einem typischen HLK-System bereitgestellten Leistung aus (s. Abbildung 3.10). Generell wird davon ausgegangen, dass für mitteleuropäische Klimazonen der Heizbetrieb von HLK-Systemen in batterieelektrischen Stadtbussen signifikanter ist als der Kühlbetrieb in Bezug auf den geschätzten



(a) Heizbetrieb, $-15\,^{\circ}\mathrm{C}$ (b) Kühlbetrieb, $30\,^{\circ}\mathrm{C}$

Abbildung 3.10: Relative Anteile der einzelnen Einflussgrößen zum Heiz- und Kühlbetrieb im Stadtbus [61]

Energieverbrauch pro Kilometer und daher den kritischeren Auslegungsfall darstellt [46]. Neben den beiden Hauptfaktoren Umgebungslufttemperatur und Sonneneinstrahlung stellen die Umgebungsfeuchte und die relative Luftströmungsgeschwindigkeit in Bezug auf die Fahrzeuggeschwindigkeit zusätzliche Faktoren dar, die die gesamte Umgebungswärmelast beeinflussen [121]. Solange die Bustüren geschlossen sind, wirkt sich die Umgebungsfeuchte nicht direkt auf das Businnere aus, aber sie beeinflusst durch ihre Eigenenergie die Enthalpie der in die Lüftungsanlage angesaugten Frischluft und kann somit die erforderliche Luftbe- bzw. -entfeuchtung beeinflussen, was sich folglich auch auf die Leistungsaufnahme des HLK-Systems auswirkt. Da sich die Enthalpie der angesaugten Frischluft aus der Summe des Wasserdampfanteils der Luft und der Raumlufttemperatur ergibt, stellt letztere eine doppelte Belastung dar. Da die Umgebungslufttemperatur die Wärmeleitung der Fahrzeugkarosserie beeinflusst, führt sie letztlich auch zu einem exponentiell ansteigenden Leistungsbedarf des HLK-Systems bei steigenden Umgebungslufttemperaturen [49]. Die auf die Buskabine einwirkende relative Luftströmungsgeschwindigkeit setzt sich aus der natürlichen absoluten Luftströmungsgeschwindigkeit, vereinfacht gesagt der Windge-

schwindigkeit, und der durch die Busgeschwindigkeit verursachten Luftströmungsgeschwindigkeit zusammen. Beide gemeinsam beeinflussen die Wärmeleitbelastung der Buskarosserie [49].

3.6 Lücken des Standes der Wissenschaft und Technik

Die Analyse des aktuellen Stands der Technik zeigt, dass die Thematik der Funktionsverlagerung im Automotive-Bereich bereits in verschiedenen wissenschaftlichen Arbeiten adressiert wurde. Die identifizierten Frameworks und Use Cases basieren dabei überwiegend auf Konzepten der SOA.

Eine Gegenüberstellung der zu Beginn definierten Herausforderungen (Kapitel 1.1) mit den in Kapitel 3.3.1 dargestellten Frameworks sowie den in Kapitel 3.3.2 beschriebenen praktischen Use Cases zeigt, dass keine der betrachteten Veröffentlichungen sämtliche Herausforderungen vollständig adressiert (vgl. Tabelle 3.1).

Die *Suitability Analysis Methodology* (SAM) nach Milani (s. Kapitel 3.3.4) konzentriert sich primär auf technische Kriterien zur Bewertung der Eignung von Fahrzeugfunktionen für eine Cloud-Migration. Eine ganzheitliche Betrachtung, die auch ökonomische Faktoren wie die *Total Cost of Ownership* (TCO) oder Potenziale für adaptives Lernen berücksichtigt, bleibt jedoch unberücksichtigt.

Die im Stand der Technik identifizierten Deploymentmodelle cloudbasierter Fahrzeugfunktionen (Kapitel 5.6) entstammen überwiegend dem Bereich des MCC. Ein systematisches, strukturiertes Verfahren zur Auswahl des jeweils optimalen Deploymentmodells für spezifische Funktionen fehlt bislang. Ein solches Vorgehen müsste unterschiedliche Anforderungen an Performance, Latenz, Verfügbarkeit und Ressourcennutzung abwägen und als methodische Entscheidungsgrundlage dienen. Aufgrund der hohen Sicherheitsanforde-

Tabelle 3.1: Bewertung der Frameworks und Use Cases hinsichtlich der Herausforderungen HF-1 bis HF-3

Publikation	**HF-1**	**HF-2**	**HF-3**
Kugele et al. (Elastic Service Provision)	ja	ja	nein
Banijamali et al. (KUKSA)	ja	teilweise[1]	teilweise[2]
Ashok et al. (Computation Offloading)	ja	teilweise[3]	teilweise[4]
Deng et al. (Commercial Cloud Applications)	ja	teilweise[5]	teilweise[6]
Wang et al. (ADAS Digital Twin)	ja	teilweise[7]	teilweise[8]
Yang et al. (Bus Energy Optimization)	ja	teilweise[9]	nein

Legende:
HF-1 = Dynamische Integration von lose gekoppelten Softwarekomponenten
HF-2 = Kontextadaptive und robuste Regelung trotz intermittierender Konnektivität und Serviceausfällen
HF-3 = Lebenszyklusfähige und wartbare Regelalgorithmen für Cloud-Plattformen

[1] Selbstadaptivität durch Microservice-Architektur, jedoch keine tiefergehende Betrachtung von Connectivity-Ausfällen.
[2] OTA-Updatefähigkeiten und servicebasierte Entwicklung vorhanden, aber kein vollständiges Lifecycle-Management.
[3] Ad-hoc-Offloading abhängig von Netzbedingungen, jedoch keine Resilienzstrategie bei Verbindungsverlust.
[4] Rechenlastverteilung zur Effizienzsteigerung, jedoch keine Erwähnung der Wartbarkeit.
[5] Nutzung kommerzieller Clouds zur Skalierung, aber keine Absicherung gegen spezifische Serviceausfälle oder Kontextabweichungen.
[6] Adaptive Ressourcenverteilung thematisiert, aber keine Betrachtung von Wartbarkeit.
[7] Digitale Zwillinge ermöglichen kontextsensitives Verhalten, aber keine robusten Fallback-Strategien bei Connectivity-Verlust.
[8] Updatefähigkeit theoretisch angesprochen.
[9] Regelung berücksichtigt Fahrbedingungen, aber keine Ausfallsicherheit bei Cloudverbindungsausfällen.

rungen im Automotive-Bereich ist zudem anzunehmen, dass weitere, über das klassische MCC hinausgehende Deploymentmodelle erforderlich sind.

3.7 Beitrag dieser Dissertation im Kontext der Forschungsfragen

Die Dissertation schließt Forschungslücken zur Identifikation cloudfähiger Funktionen (FF 1), Auswahl geeigneter Deploymentmodelle (FF 2) und Evaluation von Cloudverlagerungspotenzialen (FF 3). Zunächst wird ein systematischer Prozess zur Identifikation cloudbasierter Fahrzeugfunktionen entwickelt, der technische, energetische und ökonomische Kriterien (z.B. TCO) sowie Potenziale wie maschinelles Lernen in Fahrzeugflotten berücksichtigt (→ FF 1).

Eine multikriterielle Entscheidungsanalyse (MCDA) integriert technische, ökonomische und energetische Dimensionen für die fundierte Auswahl geeigneter Deploymentmodelle (s. Kapitel 5.6) cloudbasierter Fahrzeugfunktionen (→ FF 2).

Die Evaluation erfolgt auf einer Testplattform, die die Simulationsumgebung *Carla* mit einer vereinfachten E/E-Architektur bestehend aus Standard-ECUs und einem HPC kombiniert. Eine serviceorientierte Softwarearchitektur auf ROS 2-Basis ermöglicht die nahtlose Integration externer Services (s. Definition 2.2). Als Fallbeispiel wird die cloudbasierte HLK-Regelung für einen batterieelektrischen Stadtbus prototypisch umgesetzt - sowohl als *Nur Cloud*- als auch als *Fallback*-Deploymentmodell. Die Evaluation verdeutlicht die Auswirkungen auf Energieeffizienz und Betriebsfähigkeit unter variierenden Bedingungen (→ FF 3).

4 Definition und Identifikation cloudfähiger Fahrzeugfunktionen

4.1 Die Definition einer cloudbasierten Fahrzeugfunktion

Die in Kapitel 3.1 vorgestellte Beschreibung einer cloudbasierten Fahrzeugfunktion dient als Ausgangspunkt für die eigene Definition.

Die Definition von Milani wird angepasst, da diese „Funktionen in der Cloud ohne Einfluss auf die fahrzeuginterne Regelung bzw. Steuerung auf den entsprechenden Steuergeräten“ nicht als cloudbasierte Fahrzeugfunktionen betrachtet. Diese Grenze wird in der eigenen Definition nicht gezogen (vgl. Tabelle 5.1). Zudem wird klargestellt, dass es sich um eine SWC (vgl. Eigenschaften einer SWC in Tabelle A.3) handelt, die Eingabewerte empfängt, mit diesen eine spezifische Handlung ausführt und Ausgabewerte zurückgibt (vgl. Definition 1.5).

Definition 4.1 Cloudbasierte Fahrzeugfunktionen: Cloudbasierte Fahrzeugfunktionen sind Softwarekomponenten, die anstelle der verfügbaren Rechenkapazitäten des Fahrzeugs Rechen- und/oder Speicherkapazitäten der Cloud nutzen. Sie können sowohl Informationen aus dem Fahrzeug als auch Daten aus der Cloud als Eingabe verwenden. Die Ausgaben der verlagerten Funktionen optimieren bestehende Funktionen im Fahrzeug, ersetzen sie oder bilden selbst neue Funktionen.

4.2 Identifikation cloudfähiger Fahrzeugfunktionen

4.2.1 Anforderungen

Die Eignung einer Fahrzeugfunktion, im Sinne einer Funktion als SWC (vgl. Definition 1.5) zur Auslagerung in die Cloud wird anhand der folgenden Anforderungen analysiert:

Req-1 (Funktion) **Echtzeitanforderungen:** Deadline Beschränkungen bei der Fnktionsausführung (s. Definition 2.1)

Req-2 (Funktion) **Safety- und Securityanforderungen:** Erfüllung von Sicherheits- und Securityanforderungen (s. Definitionen 4.2 & 4.3)

Req-3 (Funktion) **Ressourcenanforderungen:** Verfügbarkeit notwendiger Rechen- und energetischer Ressourcen für die Ausführung (und den Datentransfer)

Req-4 (Funktion) **Hardwareabhängigkeiten:** Erfüllung hardwarebedingter und baulicher Integrationsanforderungen

Req-5 (Funktion) **Ökonomische Anforderungen:** Hard-/Softwarekosten und Servicekosten

Req-6 (Funktion) **Qualitätsanforderungen:** Generierung von Mehrwerten (wie z.B. Flottenlernen) durch die Ausführung der Funktion in der Cloud

> **Definition 4.2 Safety:** Erwartung, dass ein System unter definierten Bedingungen nicht zu einem Zustand führt, in dem Leben, Gesundheit oder Eigentum von Menschen oder die Umwelt gefährdet werden [57].

> **Definition 4.3 Security:** Schutz von Fahrzeughardware oder -software vor versehentlichem oder böswilligem Zugriff, Verwendung, Änderung, Zerstörung oder Offenlegung [57].

Securityanforderungen (**Req-2 (Funktion)**) betreffen Themen der Datensicherheit, den Verlust sensibler Informationen durch den Datentransfer und die Ausführung von Funktionen in der Cloud. Dieser Aspekt ist nicht Teil dieser Abhandlung.
Hardwareabhängigkeiten (**Req-4 (Funktion)**) sind bei einer Funktion dann vorhanden, wenn diese direkt auf spezifische HWC (s. Definition 1.3) zugreift, beispielsweise durch Register-Zugriffe, Timing-kritische Operationen oder die unmittelbare Verarbeitung von Sensor- und Aktorschnittstellen. Funktionen, die auf höheren Abstraktionsebenen arbeiten und über standardisierte APIs oder Middleware kommunizieren, weisen geringe Hardwareabhängigkeiten auf. Im nachfolgenden Prozess wird eine geringe Hardwareabhängigkeit der betrachteten Funktionen vorausgesetzt, da diese primär algorithmische Verarbeitung von bereits aufbereiteten Sensordaten durchführen.

4.2.2 Bewertung der Realisierbarkeit

Die technische Realisierbarkeit R_{Cloud} einer Cloudverlagerung wird durch die Erfüllung von Echtzeit- (**Req-1 (Funktion)**) und Safety- und Securityanforderungen (**Req-2 (Funktion)**) bestimmt.

R_S: Realisierbarkeitsanalyse der Sicherheitsanforderungen
Sicherheitsanforderungen im Fahrzeug beziehen sich auf die in Kapitel A.13 beschriebene Gefährdungs- und Risikoanalyse und die damit einhergehende Einstufung von *items* in Automotive Safety Integrity Level (ASIL). Dieses *item* wird im ersten Schritt als eine Funktion im Sinne einer SWC (vgl. Definition 1.5) definiert.

Die Zuweisung einer SWC zu einem ASIL bedingt die Umsetzung von Maßnahmen zur Risikominderung auf Systemebene. Teil 4 der ISO 26262 (s. Kapitel A.13) fordert die Entwicklung einer technischen Sicherheitskonzeption und die Ableitung einer funktionalen Systemarchitektur, in der sicherheitsrelevante Funktionen auf Systemelemente wie Steuergeräte, Sensoren, Aktoren und Kommunikationsschnittstellen verteilt werden. Wird eine SWC in die Cloud ausgelagert, erweitern sich diese Systemelemente um die Kommunikationsstrecke zwischen Fahrzeug und Cloud sowie die Cloud-Infrastruktur selbst. Beide stellen zusätzliche potenzielle Fehlerquellen dar, die in der Systemarchitektur und im Sicherheitskonzept explizit berücksichtigt werden müssen.

Diese beiden zusätzlichen Systemelemente und die zugehörigen möglichen Fehlerursachen führen dazu, dass die Maßnahmen der ISO 26262 nicht mehr genügen, um das Risikopotenzial auf ein Restrisiko zu senken, das unterhalb des tolerierbaren Risikos liegt (s. Abbildung 4.1).

Die in Tabelle 4.1 aufgeführte mögliche Fehlerursache „Unterbrechung“ wird beispielhaft näher betrachtet, um zu verdeutlichen, mit welchen Ausfallwahrscheinlichkeiten auf den hinzugefügten Schichten gerechnet werden kann.

Tabelle 4.1: Potenzielle Fehlerursachen cloudbasierter Fahrzeugfunktionen (angelehnt an [75])

Systemelement	Mögliche Fehlerursachen
Cloud	Cloud nicht verfügbar aufgrund von physikalischen Ausfällen der Hardware
	Fehler der Prozessabläufe in der Cloud durch Ressourcenschwankungen und unzureichende Priorisierung
	Serviceausfälle aufgrund von abgelaufenen Servicezeiten
Kommunikation	Hohe Latenz (Netzwerküberlastung, hohe Entfernung) und niedrige Datenrate (begrenzte Bandbreite)
	Unterbrechung (Abdeckung, Verfügbarkeit)
	Niedrige Zuverlässigkeit (Hoher Paketverlust)

Beim Datenaustausch über Mobilfunknetze wird aktuell eine Betriebszeit von $99{,}999\,\%$ erreicht, was zu einer Ausfallwahrscheinlichkeit von $<10^{-5}$ [37] führt. In der Realität ist das Fahrzeug jedoch nicht als stationäres Objekt zu betrachten. Bei sich bewegenden Objekten sinkt die mobile Breitbandverfügbarkeit je nach Dienstanbieter auf etwa $99{,}99\,\%$ [37], was einer Unterbrechungszeit von etwa $360\,\mathrm{ms\,h^{-1}}$ (bzw. einer Ausfallwahrscheinlichkeit von $<10^{-4}\,\mathrm{h^{-1}}$) entspricht[1]. Die Anzahl der Fahrzeuge in der unmittelbaren Umgebung, die Landschaft und die Entfernung zu den Basisstationen können die Ausfallzeit noch weiter erhöhen.

Der Einfluss dieser Ausfallwahrscheinlichkeiten auf die finale Restfehlerrate einer Funktion ist kaum spezifizierbar, die ISO 26262 gibt hier aber keine genauen Hinweise zur Ableitung solcher Restfehlerraten. Wilhelm et al. [128] schlussfolgern aus der ISO bereits für ASIL A *Funktionen*, je nach Herleitung des akzeptablen Restrisikos, die Notwendigkeit eines Nachweises von Ausfallwahrscheinlichkeiten $<10^{-5}\,\mathrm{h^{-1}}$ (alle $36\,\mathrm{ms\,h^{-1}}$). Die genannte Ausfallwahrscheinlichkeit von $10^{-4}\,\mathrm{h^{-1}}$ auf der neu hinzuge-

[1] mit weiteren Verbesserungen ist in den kommenden Jahren zu rechnen

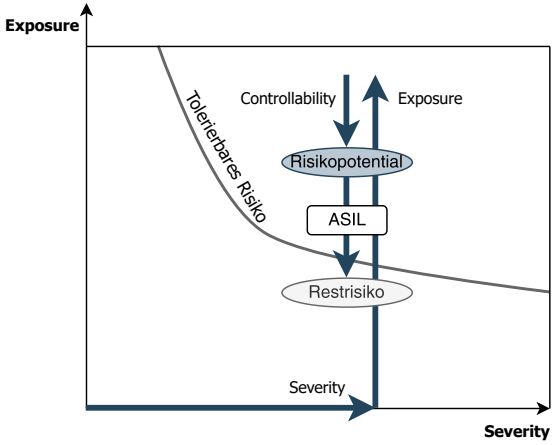


Abbildung 4.1: Verringerung des Risikopotenzials durch Umsetzung der erforderlichen Sicherheitsmaßnahmen des zugewiesenen ASIL

fügten Schicht *Kommunikation* ist demnach zu hoch für ASIL. Aus diesem Grund werden Funktionen, denen auch bisher ein ASIL A bis D zugewiesen wurde, als nicht in die Cloud verlagerbar eingestuft und im nachfolgenden Prozess nicht weiter betrachtet.

R_{EZ}: Realisierbarkeitsanalyse der Echtzeitanforderungen
Harte Echtzeitsysteme (s. Definition 2.1) benötigen garantiert Antworten innerhalb eines vorgegebenen Zeitintervalls (Deadline-Zeitpunkt). Wird dieses Intervall überschritten, liegt ein Verstoß gegen die Echtzeitanforderung vor, was zu schwerwiegenden Fehlfunktionen führen kann. Betrachtet man ein System, das auf Antworten einer cloudbasierten Funktion wartet, ist die Umsetzung harter Echtzeitanforderungen abhängig von der konkret geforderten Deadline-Zeit. Die drahtlose Kommunikationsanbindung zur Cloud stellt eine Herausforderung im Systemdesign dar. Die genaue Beschreibung der Einflussfaktoren auf die drahtlose Mobilfunkkommunikation wird in Kapitel 2.6.2 zusammengefasst. Je nach geforderter Reaktionszeit und implementierten Sicherheitsmaßnahmen könnten einige

Funktionen mit harten Echtzeitanforderungen möglicherweise in der Cloud realisierbar sein. Aufgrund der genannten Gründe und der Schwierigkeit, absolute Garantien für die Kommunikationslatenz zu geben, werden diese Funktionen in der vorliegenden Arbeit jedoch nicht weiter betrachtet. Als Beispiel für solche kritischen Funktionen seien hier regelnde oder steuernde Komponenten des Motors genannt [104].

4.2.3 Bewertung der Eignung

Die Eignung einer Funktion für die Auslagerung wird mithilfe von Bewertungsmetriken in einem Prozess ermittelt. Der Prozess orientiert sich an den in Kapitel 4.2 beschriebenen Anforderungen und ermittelt einen Eignungs-Score zur Auslagerung der Funktionen. Dabei wird auf der vorgestellten Methodik von Milani aus Kapitel 3.3.4 aufgebaut und die genannten Lücken bzw. Erkenntnisse aus der Arbeit für den eigenen Prozess (s. Abbildung 4.2) überarbeitet.

Eignungs-Score Die Berechnung des Eignungs-Score E_{Cloud} beruht auf fünf Bewertungen:

1. $\mathbf{B_{RB}}$: Diese Bewertung orientiert sich an den in Kapitel 3.3.5 beschriebenen Trade-Off zwischen benötigter Bandbreite (B) und Rechenanforderungen (R) (**Req-3 (Funktion)**). Die Transferdatenmenge wird allerdings durch die aussagekräftigere Kennzahl der benötigten Bandbreite (B) ersetzt.

2. $\mathbf{B_Z}$: Bewertung der potenziellen Verkürzung der Ausführungszeit (Z) einer Funktion bei einer Verlagerung in die Cloud.

3. $\mathbf{B_{EE}}$: Energieeinsparpotenziale (EE) im Fahrzeug durch die Verlagerung des Ausführungsortes einer Funktion in die Cloud (**Req-3 (Funktion)**).

4. $\mathbf{B_{Kosten}}$: Bewertung der Kosteneinsparpotenziale (Kosten) durch eine Cloudverlagerung (**Req-5 (Funktion)**).

5. $\mathbf{B_L}$: Bewertung der Potenziale durch die Ausführung der Funktion in der Cloud und der Möglichkeit des (maschinellen) Lernens (L) aus Flottendaten (**Req-6 (Funktion)**).

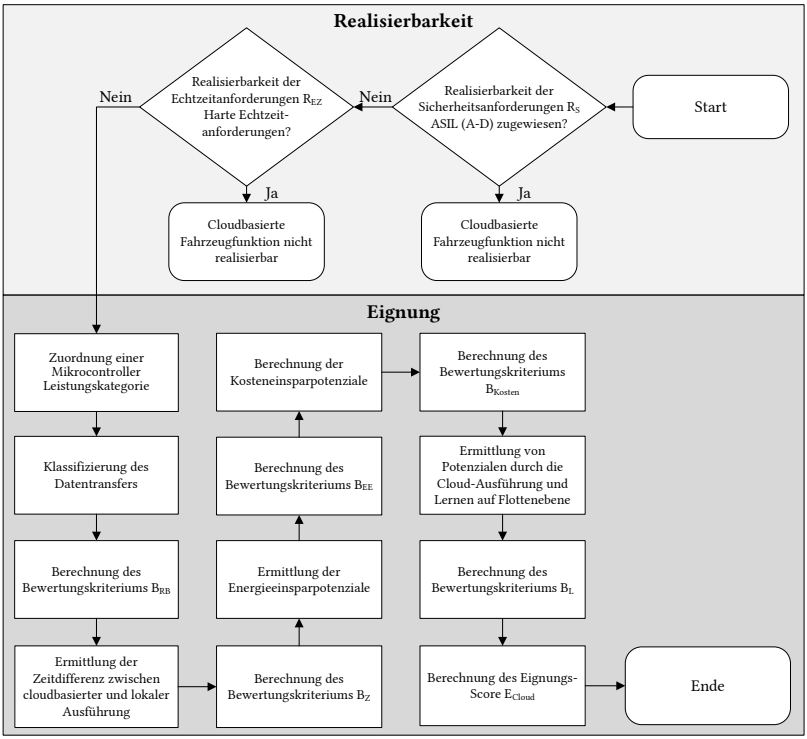


Abbildung 4.2: Prozess zur Bewertung der Realisierbarkeit und Eignung von cloudbasierten Fahrzeugfunktionen

B_{RB}: Bewertungsmetrik für den Trade-Off zwischen Bandbreitenbedarf und Rechenaufwand

Die Rechenanforderungen (R) werden anhand der in Tabelle 4.2 dargestellten Kategorisierung bewertet.

Tabelle 4.2: Kategorisierung von Funktionen nach Rechenanforderungen (angelehnt an [21])

Kategorie	Rechenanforderung	Typische Funktion
Sehr hoch (4)	Multithreading, Parallelisierung, hoher RAM- und CPU-Bedarf	Bildverarbeitung, Objekterkennung, Sensorfusion, Pfadplanung
Hoch (3)	Gleitkommaoperationen, regelbasierte oder modellgestützte Algorithmen	Fahrdynamikregelung, modellbasierte Diagnose, modellprädiktive Regelung
Medium (2)	Zyklusgesteuerte Verarbeitung, geringe Parallelität, moderate Speicher- und CPU-Last	PID-Regelung, Licht- bzw. Fenstersteuerung, einfache Logikfunktionen
Gering (1)	minimaler Speicherbedarf, Sub-Millisekunden bis wenige Millisekunden Rechenzeit	Signalweiterleitung, GPIO-Handling, einfache Filter oder Triggerlogik

Die Einordnung des Bandbreitenbedarfs (B) orientiert sich an den typischen Datenübertragungsraten innerhalb der verschiedenen Fahrzeugdomänen (s. Tabelle 4.3) und erfolgt ebenfalls in vier Kategorien. Die Kombination aus Rechenaufwand und Bandbreitenbedarf erlaubt anschließend die Bestimmung des Bewertungsfaktors B_{RB} gemäß Abbildung 4.3.

Tabelle 4.3: Klassifizierung der benötigten Bandbreiten in Fahrzeugnetzwerken [96] [97]

Kategorie	**Benötigte Bandbreite (B)**	**Fzg.-Domäne**
Sehr hoch (4)	>25 $\mathrm{Mbit\,s^{-1}}$	Fahrerassistenz
Hoch (3)	10 $\mathrm{Mbit\,s^{-1}}$ bis 25 $\mathrm{Mbit\,s^{-1}}$	Infotainment
Medium (2)	1 $\mathrm{Mbit\,s^{-1}}$ bis 10 $\mathrm{Mbit\,s^{-1}}$	Fahrwerk, Karosserie
Gering (1)	<1 $\mathrm{Mbit\,s^{-1}}$	Antriebsstrang

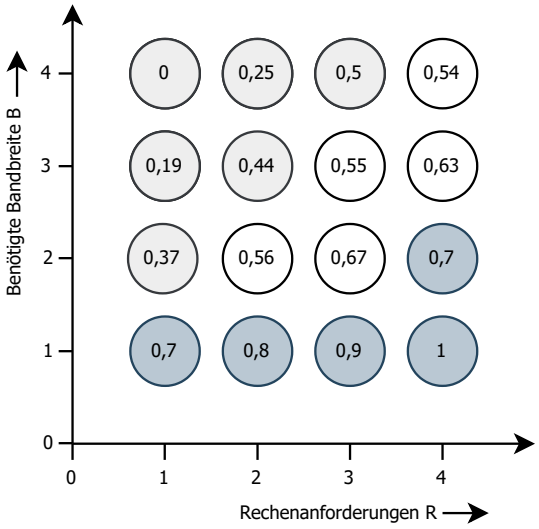


Abbildung 4.3: Mögliche Werte für B_{RB} entsprechend der Einordnung der Rechenanforderungen (s. Tabelle 4.2) und der Transferdatenmenge (s. Tabelle 4.3)

B_Z: Bewertungsmetrik der Ausführungszeit
Wenn der Datenaustausch zwischen dem Fahrzeug und der Cloud nicht streaming-basiert (s. Kapitel A.5) ist und keine dauerhaft erhöhte CPU-Auslastung vorliegt, kann die statische Zeitdifferenz zwischen der Ausführung der Funktion im Fahrzeug und in der Cloud ermittelt werden.
Daher müssen die Formeln zur Berechnung der Antwortzeit der lokalen Funktion und der cloudbasierten Funktion eingeführt werden.

> **Definition 4.4 Antwortzeit:** Die Antwortzeit gibt an, wie viel Zeit ein System benötigt, um auf eine bestimmte Situation zu reagieren. Genauer gesagt gibt sie die Zeit an, die ein System benötigt, um auf eingehende Eingangsdaten mit der entsprechenden Ausgabe zu reagieren.

Die jeweiligen Antwortzeiten können verwendet werden, um die potenziellen Zeiteinsparungen durch Auslagerung der Funktion in die Cloud zu ermitteln. Da potenziell mehrere Funktionen auf einem Steuergerät lokalisiert sind, wird angenommen, dass $t_{\text{Aktivierung}}$ vernachlässigbar ist, da das Steuergerät zwischen den Ausführungsintervallen nicht in den Ruhezustand versetzt wird. Es werden die folgenden Formeln eingeführt:

$$t_{\text{Antwort,Lokal}} = t_{\text{Exe,ECU}} + t_{\text{Bus,Lokal}} \tag{4.1}$$

$$t_{\text{Antwort,Cloud}} = t_{\text{Exe,Cloud}} + t_{\text{Trans,Cloud}} + t_{\text{Latenz}} + t_{\text{Bus,Lokal}} \tag{4.2}$$

Die Ausführungszeit auf dem lokalen Steuergerät $t_{\text{Exe,ECU}}$ ist von der Leistungsklasse der CPU, der Arbeitslast etc. abhängig. Basierend auf der Arbeit von Ashok et al. [9] wird von einer 10-mal schnelleren Ausführung der Funktion in der Cloud im Vergleich zum lokalen Steuergerät ausgegangen. Die benötigte Zeit zur Übertragung von Sensordaten an das lokale Steuergerät $t_{\text{Bus,Lokal}}$ bzw. bei der Cloud-Variante an das Übertragungsmodul

wird vernachlässigt, da davon ausgegangen wird, dass die Zeit in beiden Fällen identisch ist. $t_{\mathrm{Trans,Cloud}}$ wird folgendermaßen genauer definiert:

$$t_{\mathrm{Trans,Cloud}} = t_{\mathrm{Upload}} + t_{\mathrm{Download}} \tag{4.3}$$

t_{Upload} und t_{Download} können unter Verwendung der durchschnittlichen Datenraten für Upload R_{Up} und Download R_{Down} im deutschen LTE-Mobilfunknetz (s. Kapitel 2.6.2) und der Datenmenge D_{Up} und D_{Down} (als Orientierung der Transferdatenmengen kann hier die Quelle [83] herangezogen werden) berechnet werden.

$$t_{\mathrm{Upload}} + t_{\mathrm{Download}} = \frac{D_{\mathrm{Up}}}{R_{\mathrm{Up}}} + \frac{D_{\mathrm{Down}}}{R_{\mathrm{Down}}} \tag{4.4}$$

Die Latenzzeit t_{Latenz} im LTE-Netz ist abhängig von vielen Parametern, wird aber im Rahmen dieses Prozesses angelehnt an [132] auf $25\,\mathrm{ms}$ festgesetzt. Die Zeitdifferenz zwischen der lokalen und der Cloud-Variante wird daher mithilfe der folgenden Gleichung ermittelt:

$$t_{\mathrm{Diff}} = t_{\mathrm{Antwort,Lokal}} - t_{\mathrm{Antwort,Cloud}} \tag{4.5}$$

$$t_{\mathrm{Diff}} = \frac{9}{10} t_{\mathrm{Exe,ECU}} - \left(\frac{D_{\mathrm{Up}}}{R_{\mathrm{Up}}} + \frac{D_{\mathrm{Down}}}{R_{\mathrm{Down}}} + 25\,\mathrm{ms}\right) \tag{4.6}$$

Handelt es sich um eine zyklische Funktion mit unregelmäßigen Intervallen (s. Kapitel A.5), muss die Bedeutung der Gleichung genauer erklärt werden. In diesem Fall wird die Antwortzeit $t_{\mathrm{Antwort,Cloud}}$ ausschließlich ab dem Beginn der letzten Upload-Sequenz mit anschließender Berechnung und Download berechnet. Die Zeitspanne, in der nur Daten in die Cloud hochgeladen werden, ohne dass anschließend ein erhöhter Rechenaufwand entsteht, wird nicht in die Analyse einbezogen.
Das Bewertungskriterium B_{Z} wird mithilfe der in Abbildung 4.4 dargestellten Beziehung ermittelt.

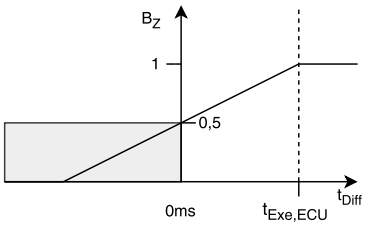


Abbildung 4.4: Bewertungsmetrik der Ausführungszeit bei der Funktionsverlagerung in die Cloud

Die Obergrenze der zeitlichen Ersparnis orientiert sich an der lokalen Ausführungszeit der Funktion $t_{\mathrm{Exe,ECU}}$. Im Falle einer höheren Zeitersparnis wird B_{Z} entsprechend auf 1 gesetzt:

$$-t_{\mathrm{Exe,ECU}} \leq t_{\mathrm{Diff}} \leq t_{\mathrm{Exe,ECU}} : B_{\mathrm{Z}} = \frac{t_{\mathrm{Diff}}[ms]}{t_{\mathrm{Exe,ECU}}} + 0,5 \tag{4.7}$$

$$t_{\mathrm{Exe,ECU}} < t_{\mathrm{Diff}} : B_{\mathrm{Z}} = 1 \tag{4.8}$$

$$-t_{\mathrm{Exe,ECU}} > t_{\mathrm{Diff}} : B_{\mathrm{Z}} = 0 \tag{4.9}$$

B_{EE}: Bewertungsmetrik der Energieeinsparpotenziale im Fahrzeug
Insofern die Funktion in der Cloud und nicht mehr im Fahrzeug ausgeführt wird, entstehen Energieeinsparpotenziale im Fahrzeug. Im Fahrzeug fällt nur noch Energie für das Senden und Empfangen der notwendigen Input- und Outputsignale der Funktion an.

Die Einsparpotenziale im Fahrzeug berechnen sich aus der Differenz zwischen der benötigten Energie zur lokalen Ausführung der Funktion und der benötigten Energie für den Datentransfer mit dem Backend.

Der Energieverbrauch des lokalen Steuergeräts für die Dauer der Ausführung $t_{\mathrm{Exe,ECU}}$ der Funktion ist abhängig von der durchschnittlichen CPU/GPU-Auslastung $\delta_{\mathrm{Funktion}}$ der Funktion (s. auch Kapitel 2.2). Der Ener-

gieverbrauch der Funktion auf einem lokalen Steuergerät ermittelt sich folgendermaßen:

$$E_{\text{lokal}} = P_{\text{ECU}} \cdot t_{\text{Exe,ECU}} \tag{4.10}$$

P_{ECU} entspricht der Leistungsklasse des Mikrocontrollers (s. Tabelle 2.1), auf dem die Funktion ausgeführt wird.
Die benötigte Energie für den Datentransfer $E_{\text{Trans,Cloud}}$ berechnet sich wie folgt:

$$E_{\text{Trans,Cloud}} = P_{\text{Trans,Up}} \cdot \frac{D_{\text{Up}}}{R_{\text{Up}}} + P_{\text{Trans,Down}} \cdot \frac{D_{\text{Down}}}{R_{\text{Down}}} \tag{4.11}$$

Dabei sind folgende Aspekte für *zyklische* und *streaming-basierte* Funktionen zu beachten:

- Zyklisch Fall 2: Aufgrund der Leerlaufzeit zwischen dem erneuten Senden und dem Empfangen von Daten wird davon ausgegangen, dass das Sendegerät in diesem Zeitraum in den Ruhezustand wechselt und keine Energie verbraucht [54]. Da das Uploadintervall nicht identisch zum Berechnungs- und Downloadintervall ist (s. Kapitel A.5), sollten deshalb beide Intervalle getrennt betrachtet werden.
- Streaming-basiert: Das Intervall zwischen Senden und Empfangen kann als zu kurz betrachtet werden, um das Steuergerät in den Ruhezustand zu wechseln [54]. Es wird von einer Grundlast β (s. Kapitel 2.6.2) ausgegangen.

Die Differenz E_{Diff} aus der lokalen Ausführung und der benötigten Energie für den Datentransfer

$$E_{\text{Diff}} = E_{\text{Lokal}} - E_{\text{Trans,Cloud}} \tag{4.12}$$

dient zur Berechnung der Relation zur Gesamtenergie E_{Total}, die die Summe von lokaler Ausführung und Datentransfer darstellt

$$E_{\mathrm{Diff,Ratio}} = \frac{E_{\mathrm{Diff}}}{E_{\mathrm{Total}}} \tag{4.13}$$

und wird schlussendlich zur Berechnung der Bewertungsmetrik der Energieeinsparpotenziale B_{EE} herangezogen (s. Abbildung 4.5):

$$B_{\mathrm{EE}} = \frac{E_{\mathrm{Diff,Ratio}}}{2} + 0,5 \tag{4.14}$$

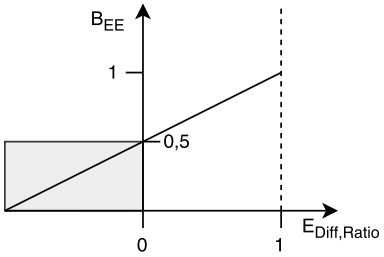


Abbildung 4.5: Bewertungsmetrik der Energieeinsparung der Funktionsverlagerung in die Cloud

Die dargestellte Betrachtung von Energieeinsparpotenzialen behandelt ausschließlich die Ebene der Steuergeräte und deren Energieverbrauch. Die bereits in der Motivation (Kapitel 1.1) genannten Potenziale der optimierten Gestaltung, bspw. in Form einer effizienteren Regelung, von Features in der Cloud kann nicht bemessen werden, soll aber am Ende durch Forschungsfrage 3 beantwortet werden.

$\mathbf{B_{Kosten}}$: Kosteneinsparpotenziale durch die Cloudverlagerung
Die Kosten der Bereitstellung einer Funktion können nach dem TCO Modell berechnet werden (s. [BtS+24]). Sowohl im Fahrzeug als auch in der Cloud entstehen Kosten für die Entwicklung $C_{\mathrm{Entwicklung}}$, das Deployment $C_{\mathrm{Deployment}}$ und die Ausführung C_{Exe} der Funktion.

- $\text{TCO}_{\text{Fkt.,Fzg.}} = C_{\text{Entwicklung,Fzg.}} + C_{\text{Depl,Fzg.}} + C_{\text{Exe,Fzg.}}$
  Die Entwicklung der Funktion stellt Capital Expenditures (CapEx) dar. Deploymentkosten im Fahrzeug beinhalten Investitionsausgaben (CapEx) der Steuergeräte, deren Zusammensetzung Schäuffele in [104] beschreibt. Die Ausführungskosten $C_{\text{Exe,Fzg.}}$ beinhalten laufende Kosten im Betrieb, z.B. Servicekosten durch OTA-Updates, Diagnose, Wartung oder Supportmaßnahmen.
- $\text{TCO}_{\text{Fkt.,Cloud}} = C_{\text{Entwicklung,Cloud}} + C_{\text{Depl,Cloud}} + C_{\text{Exe,Cloud}}$
  Eine Verlagerung der Funktion in die Cloud stellt einen Wechsel von CapEx zu Operating Expenses (OpEx) dar. Während $C_{\text{Depl,Cloud}}$ mit einem Wert von 0€ angenommen werden kann, entstehen OpEx hauptsächlich durch die Bereitstellung der Rechenressourcen, entweder intern oder durch einen Cloud Service Provider (CSP). CapEx für die Entwicklung der Funktion bleiben erhalten.

Die Bewertung des Kosteneinsparpotenzials kann mithilfe einer Break-Even-Analyse durchgeführt werden. Dabei wird die folgende Gleichung

$$\begin{aligned}&(\text{CapEx}_{\text{Cloud}} + \text{OpEx}_{\text{Cloud}} \cdot t_{\text{Break-Even,Fzg.}})\\ -&(\text{CapEx}_{\text{Fzg.}} + \text{OpEx}_{\text{Fzg.}} \cdot t_{\text{Break-Even,Fzg.}}) = 0\end{aligned} \tag{4.15}$$

nach $t_{\text{Break-Even;Fahrzeug}}$ in Anzahl Monaten (aufgrund der monatlichen Rechnungsstellung der OpEx) aufgelöst. Das Bewertungskriterium B_{Kosten} wird mithilfe der Break-Even Dauer bis zu dem Zeitpunkt, ab dem die Bereitstellung der Funktion im Fahrzeug wieder kostengünstiger ist (in Jahren) und dem Durchschnittsalter von Personenkraftwagen (bzw. Kraftomnibussen) in Deutschland berechnet. Dieses Durchschnittsalter lag zum 1. Januar 2022 bei 10,1 Jahren für PKW und bei 8,3 Jahren für Omnibusse [66]. Übersteigt die Break-Even Dauer den Wert von 10,1 Jahren (bzw. 8,3 Jahren), so wird folgerichtig die maximale Bewertung von 1 angenommen.

Tabelle 4.4: Kategorisierung der Kosteneinsparpotenziale einer Funktionsverlagerung

Break-Even Dauer in Jahren	**Bewertungskriterium B_{Kosten}**
$0 \geq t_{\text{Break-Even,Fzg.}} \leq 2,5$ (bzw. 2)	0
$2,5(2) > t_{\text{Break-Even,Fzg.}} \leq 5$ (4)	0,25
$5(4) > t_{\text{Break-Even,Fzg.}} \leq 7,5$ (6)	0,5
$7,5(6) > t_{\text{Break-Even,Fzg.}} \leq 10,1$ (8,3)	0,75
$t_{\text{Break-Even,Fzg.}} > 10,1$ (8,3)	1

B_L: Bewertung der Potenziale durch die Ausführung der Funktion in der Cloud und der Möglichkeit des (maschinellen) Lernens (L) aus Flottendaten

Neben den genannten quantifizierbaren Bewertungskriterien wird die qualitative Bewertung der Cloudverlagerung berücksichtigt, die sich aus den Potenzialen einer Verlagerung ergibt. In Kapitel 3.3.5 wurden bereits einige Potenziale unter den Oberbegriffen des „Function Offloading“ und des „Function partitioning“ vorgestellt. Diese werden nachfolgend aufgegriffen und genauer spezifiziert sowie ergänzt:

- **Ressourcenbeschränkungen reduzieren:** Die Ressourcenbeschränkungen im Fahrzeug können durch die Verwendung der skalierbaren Ressourcen der Cloud überwunden werden. Die vorhandenen Ressourcen im Fahrzeug können für Funktionen verwendet werden, die unumgänglich im Fahrzeug ausgeführt werden müssen.
- **Reduzieren der Entwicklungs-/Servicekosten:** Eine Funktion, die partiell oder vollständig in der Cloud ausgeführt wird, kann dort wesentlich unproblematischer weiterentwickelt, parametriert und getestet werden ohne jeglichen Zugriff auf das Steuergerät im Fahrzeug. Durch diesen einfacheren Zugriff auf die Funktion sinken beim Original Equipment Manufacturer (OEM), genauer gesagt dem Funk-

tionsentwickler, die Kosten, da keine Rückrufe von Fahrzeugen in Werkstätten vonnöten sind.

- **Erstellen von Datenbanken und Flottenlernen:** Die kontinuierlich generierten Daten ganzer Fahrzeugflotten bieten die Möglichkeit, Datenbanken in der Cloud immer aktuell und aussagekräftig zu halten (aufgrund der potenziell hohen Anzahl an Fahrzeugen), wovon wiederum alle Fahrzeuge profitieren können. Beispielhaft bietet die Speicherung der Passagieranzahl in einer Stadtbusflotte viele energetische Einsparpotenziale für die Regelung des Fahrzeugkomforts.
- **Anbinden von externen Echtzeit-Datenquellen:** Externe Datenquellen können von einer cloudbasierten Funktion direkt angebunden und weiterverarbeitet werden. Es entfällt somit die zusätzliche Datenübertragung der externen Datenquellen in das Fahrzeug.

Die Potenziale können nicht direkt bewertet werden. Eine qualitative Aussage wird innerhalb des Prozesses anhand Tabelle 4.5 getroffen.

Tabelle 4.5: Kategorisierung der Potenziale der Cloudverlagerung

Potenziale der Verlagerung	**Bewertungskriterium** B_{L}
Keine	0
Gering	0,25
Medium	0,5
Hoch	0,75
Sehr Hoch	1

Berechnung des Eignungs-Score

Der Eignungs-Score E_{Cloud} ermöglicht es, die einzelnen Bewertungsmetriken zu gewichten.

$$E_{\mathrm{Cloud}} = w_{\mathrm{RB}} B_{\mathrm{RB}} + w_{\mathrm{Z}} B_{\mathrm{Z}} + w_{\mathrm{EE}} B_{\mathrm{EE}} + w_{\mathrm{Kosten}} B_{\mathrm{Kosten}} + w_{\mathrm{L}} B_{\mathrm{L}} \quad (4.16)$$

Dabei gilt für die Gewichte:

$$\sum_{i=1}^{n} w_{\mathrm{i}} = 1 \quad (4.17)$$

4.2.4 Anwendung des Prozesses auf die E/E-Architektur eines Stadtbusses

Der vorgestellte Prozess dient dazu, die E/E-Architektur eines BEB im Hinblick auf die Cloud-Realisierbarkeit von Funktionen zu analysieren. Im ersten Schritt wird die Verlagerbarkeit von Funktionen in die Cloud bewertet, wobei insbesondere ASIL-Zuweisungen sowie Echtzeitanforderungen berücksichtigt werden. Grundlage dieser Bewertung ist die Analyse einer modernen E/E-Architektur eines Stadtbusses, wie sie im Projekt *INDU2-OTrace* des Innovationscampus Mobilität der Zukunft[2] durchgeführt wurde.

Mehr als 60 % der in der Architektur aufgeführten Fahrzeug-Features sind einem ASIL zugeordnet. Dies bedeutet jedoch nicht zwangsläufig, dass auch die Funktionen, welche diese Features implementieren, einem ASIL unterliegen. Gemäß Teil 9 der ISO 26262 ist eine ASIL-Dekomposition möglich, wodurch einzelne Software-Items keinem ASIL, sondern lediglich dem Qualitätsmanagement (QM) unterstellt sein können. Diese Möglichkeit wird im Rahmen dieser Analyse nicht weiter verfolgt. Stattdessen werden ausschließ-

[2] `https://www.icm-bw.de/forschung/projektuebersicht/detailseite/indu2-otrace`

Tabelle 4.6: Beurteilung der Cloud-Realisierbarkeit von Funktionen exemplarischer E/E-Features im Stadtbus

Domäne	**Exempl. Feature**	**Exempl. Funktion des Features**	**Cloud-Real.**
Antriebsstrang	Motorsteuerung	Drehmoment-begrenzung	✗
	Rekuperations-steuerung	Prädiktive Rekuperation	✓
	Prädiktive Instand-haltung	Modelle der prädiktiven Instandhaltung	✓(E)
	Reichweitenschätzung	Energiever-brauchsmodell	✓(B)
Fahrwerk	Wankstabilisierung	Dämpferregelung	✗
Infotainment	Navigation	Echtzeit-Verkehrsinfos	✓(A)
Karosserie	Heizung, Lüftung, Klimatisierung	Kaskadierter HLK-Regler	✓(D)
Fahrerassistenz	Müdigkeitserkennung	Lidschlagfrequenz-erkennung	✗
	Abstandsregel-tempomat	Abstandsregelung	✗
	Notbremsassistent	Echtzeit-Objekterkennung	✗
	Spurhalteassistent	Spurzentrierung	✗
	Erstellung von Fahrerprofilen	Verhaltensanalyse	✓(C)

lich Features und die zugehörigen Funktionen betrachtet, denen kein ASIL zugewiesen wurde.

Einen Auszug der untersuchten Funktionen und deren Cloud-Realisierbarkeit, geordnet nach den in Abbildung 2.7 dargestellten Fahrzeugdomänen, zeigt Tabelle 4.6.

Für die übrigen Funktionen erfolgt die Bewertung in der zweiten Stufe des Prozesses unter Anwendung der Gewichtungen der jeweiligen Bewertungskriterien, die im Rahmen des *INDU2-OTrace* Projekts in Zusammenarbeit mit den Projektpartnern erarbeitet wurden:
$w_{\mathrm{RB}} = 0,3$, $w_{\mathrm{Z}} = 0,1$, $w_{\mathrm{EE}} = 0,15$, $w_{\mathrm{Kosten}} = 0,1$ und $w_{\mathrm{L}} = 0,35$
Tabelle 4.7 enthält die fünf Funktionen mit den höchsten Eignungs-Scores für eine potenzielle Cloud-Auslagerung im Kontext eines Stadtbusses.

Tabelle 4.7: Cloud-Eignungs-Score der am höchsten bewerteten Funktionen im Stadtbus

Bezeichnung	**Eignungs-Score** E_{Cloud}
(A) Echtzeit-Verkehrslage	0,9175
(B) Energieverbrauchsmodellierung	0,911
(C) Fahrerverhaltensanalyse	0,875
(D) Kaskadierter HLK-Regler	0,798
(E) Modelle der prädiktiven Instandhaltung	0,740

Die beiden am höchsten bewerteten Funktionen, Echtzeit-Verkehrslage und Energieverbrauchsmodellierung, werden nicht weiter vertieft behandelt. Erstere ist bereits in mehreren Serienfahrzeugen etabliert (vgl. [16]), während Letztere in der wissenschaftlichen Literatur umfassend untersucht wurde (vgl. [117]). Auch die Analyse des Fahrverhaltens (Funktion C) wird aufgrund der Verarbeitung personenbezogener Daten und der damit verbundenen Anforderungen der Datenschutz-Grundverordnung (DSGVO) nicht weiter verfolgt. Damit rückt der kaskadierte HLK-Regler (Funktion D) in den Mit-

telpunkt der weiteren Betrachtung. Er bietet ein praxisnahes und zugleich wissenschaftlich relevantes Anwendungsszenario, da die HLK-Regelung bei BEB sowohl maßgeblich zur Energieeffizienz beiträgt als auch den Fahrgastkomfort sicherstellt. Im folgenden Kapitel werden daher die Konzeption und prototypische Umsetzung eines cloudbasierten HLK-Reglers für batterielelektrische Stadtbusse vorgestellt.

Eignungs-Score der kaskadierten HLK-Regelung Die nachfolgenden Daten, beispielsweise zu Berechnungszeiten oder Leistungsaufnahmen, beziehen sich auf den Stand der Technik einer HLK-Regelung im Stadtbus (s. Kapitel 3.4), in diesem Fall mittels PID-Regler. Wie diese in Zukunft durch den cloudbasierten Ansatz aussehen könnte, wird durch das Bewertungskriterium B_{L} „Potenziale durch die Ausführung der Funktion in der Cloud“ widergespiegelt.

$\mathbf{B_{RB}}$: Die Datenübertragung für die HLK-Regelung beschränkt sich auf das Senden und Empfangen von Sensor- und Aktorwerten (B auf den Wert 1 gesetzt). Die Rechenanforderungen der State of the Art PID Regler (Kapitel 3.4) im Stadtbus sind gering (R auf den Wert 2 gesetzt). Aus Abbildung 4.3 ergibt sich hiermit ein Wert von 0,8 für B_{RB}.

$\mathbf{B_Z}$: Eine mögliche Reduzierung der Ausführungszeit t_{Diff} wird mithilfe der Gleichung 4.6 berechnet. Die Ausführungszeit der HLK-Regelung auf dem Steuergerät im Stadtbus wird mit $25\,\mathrm{ms}$ angenommen. Bei einem Übertragungsdatenvolumen für Up- und Download von jeweils $1\,\mathrm{kB}$ und der durchschnittlichen Up- und Downloadrate im deutschen Mobilfunknetz (s. Kapitel 2.6.2) wird ein Wert von $-2{,}5\,\mathrm{ms}$ für t_{Diff} errechnet. B_{Z} berechnet sich demnach auf den Wert 0,45.

$\mathbf{B_{EE}}$: Die Energie zur Ausführung der HLK-Regelung auf einem Steuergerät wird anhand der Leistungsklassen typischer Steuergeräte im Fahrzeug berechnet (Tabelle 2.1). Die HLK-Regelung wird auf einem Steuergerät der Leistungsklasse *Medium* ausgeführt und lastet dieses zu $30\,\%$ aus. Die ein-

malige Ausführung der Regelung berechnet sich anhand der Gleichungen 2.3 und 2.4 und der Ausführungszeit von $25\,\mathrm{ms}$ auf $0{,}037.5\,\mathrm{W\,s}$. Die für den einmaligen Upload und Download der Daten ($1\,\mathrm{kB}$) der HLK-Regelung benötigte Energie wird mithilfe von Gleichung 4.11 berechnet, da es sich um eine zyklische Funktion des ersten Falls handelt (vgl. Kapitel A.5). $E_{\mathrm{Trans.Cloud}}$ summiert sich auf den Wert von $0{,}05\,\mathrm{W\,s}$. Die Differenz E_{Diff} beträgt $-0{,}012.5\,\mathrm{W\,s}$. Infolgedessen wird B_{EE} auf den Wert 0,42 bestimmt.

B_{Kosten}: Formel 4.15 wird herangezogen, um B_{Kosten} zu berechnen. Hierfür müssen folgende Annahmen getroffen werden:

1. Die Entwicklung einer Funktion für die Cloud ist aufgrund ihrer einfacheren Skalierbarkeit und Flexibilität kostengünstiger als die Entwicklung für ein definiertes Steuergerät mit einer automobilspezifischen Laufzeitumgebung. Hier wird die Annahme getroffen, dass der Kostenunterschied $20\,\%$ beträgt.

2. Die Regelung wird in einer virtuellen Maschine (VM) oder als serverless Anwendung[3] ohne weitere Komponenten wie Datenbanken etc. bei einem CSP wie Microsoft Azure ausgeführt.

3. Die HLK-Regelung wird auf einem dedizierten Steuergerät ausgeführt.

Unter Berücksichtigung dieser Annahmen werden die Entwicklungskosten des Features für das Fahrzeug auf den Wert 200.000€ und für die Cloud auf 180.000€ gesetzt. Diese Kosten werden auf eine fiktive Flotte von 1.000 Fahrzeugen aufgeteilt. Die Kosten für das Steuergerät im Fahrzeug $C_{\mathrm{Depl,Fahrzeug}}$ werden angelehnt an die Kosten eines typischen Steuergeräts der *Medium* Leistungsklasse aus [112] unter Berücksichtigung der Inflation auf 60€ festgelegt. Die OpEx der Cloud werden auf 3,50€ im Monat bzw. 2€ im Monat im Fahrzeug definiert. Der Break-Even-Point, ab dem ein Deployment im Fahrzeug wirtschaftlicher ist als in der Cloud, wird

[3] s. auch https://www.redhat.com/de/topics/cloud-native-apps/what-is-serverless

nach 10,9 Jahren erreicht. Nach Tabelle 4.4 ergibt das einen Wert von 1 für B_{Kosten}.

$\mathbf{B_L}$: Die HLK-Regelung von Stadtbussen ist stark von Störfaktoren wie Türöffnungen und variierenden Fahrgastzahlen beeinflusst, die sich dynamisch während der Fahrt ändern. Um die Auswirkungen dieser Störfaktoren zu minimieren, können fortschrittliche Regelungstechniken genutzt werden, die aufgrund von Ressourcenbeschränkungen oder aus Kostengründen nicht direkt im Fahrzeug realisierbar sind. Der Austausch von Echtzeitdaten und die Nutzung von cloudbasierten Datenbanken ermöglichen es, diese Störfaktoren bereits im Vorfeld der Fahrt zu antizipieren und entsprechende Anpassungen vorzunehmen. Dies trägt nicht nur zur Verbesserung der Regelungsgenauigkeit bei, sondern auch zur Reduzierung der Entwicklungs- und Wartungskosten, da weniger Rechenleistung und Hardwareressourcen im Fahrzeug benötigt werden. Durch den Einsatz solcher Systeme lassen sich auch die langfristigen Betriebskosten senken und die Effizienz des gesamten Flottenmanagements steigern. Die Bewertungsstufe dieser Maßnahmen wird als *Sehr hoch* eingestuft, mit einem Wert von 1 für B_{L}.

Durch Einsetzen der einzelnen Werte der Bewertungskriterien in Gleichung 4.16 mit den oben genannten Gewichten ergibt sich folgende Gleichung für den kaskadierten HLK-Regler:

$$\begin{aligned} E_{\text{Cloud}} &= w_{\text{RB}} B_{\text{RB}} + w_{\text{Z}} B_{\text{Z}} + w_{\text{EE}} B_{\text{EE}} + w_{\text{Kosten}} B_{\text{Kosten}} + w_{\text{L}} B_{\text{L}} \\ &= 0,3 \cdot 0,8 + 0,1 \cdot 0,45 + 0,15 \cdot 0,42 + 0,1 \cdot 1 + 0,35 \cdot 1 \\ &= 0,798 \end{aligned}$$

5 Die cloudbasierte HLK-Regelung eines BEB

Kapitel 2.4 gibt einen Überblick über verschiedene Klimatisierungskonzepte für Elektrobusse. Unter den Begriff Elektrobus fallen auch hybride Varianten sowie Brennstoffzellenbusse. In der vorliegenden Dissertation werden jedoch ausschließlich BEB betrachtet. Bei diesen Bussen ist der Einsatz von Aufdachwärmepumpen (Variante 3 in Abbildung 2.2) mit dem Kältemittel R744 aufgrund ihrer hohen Effizienz üblich. Des Weiteren wird die Klimatisierung einer Fahrzeugkabine betrachtet, die als eine einzelne Klimazone modelliert ist und mit der genannten Wärmepumpentechnologie betrieben wird (vgl. Definition 2.7). Zusatzheizer, beispielsweise fossile Zusatzheizgeräte oder thermische Speicher (vgl. [100]), sowie der Fahrerarbeitsplatz bleiben unberücksichtigt.

5.1 Der Use Case

Cloudbasierte Fahrzeugfunktionen werden in die vier Use Case-Kategorien aus Tabelle 5.1 eingeteilt. Die Use Cases beziehen sich auf die Definition einer cloudbasierten Fahrzeugfunktion (Definition 4.1).

Die im Stand der Wissenschaft und Technik (s. Kapitel 3.3.2) beschriebenen Anwendungsfälle werden in Tabelle 5.2 den Use Case-Kategorien aus Tabelle 5.1 zugeordnet. Dabei zeigt sich, dass der Anwendungsfall von Yang et al. der Kategorie COTA zugeordnet werden kann. Die kaskadierte HLK-Regelung

Tabelle 5.1: Use Case-Kategorien und Beispiele für cloudbasierte Fahrzeufunktionen

Use Case-Kategorie	**Beschreibung**	**Beispiel**
(1) Eingabedaten für fahrzeuginterne Features	Die Rechenressourcen der Cloud dienen als Datenvorverarbeitungsstelle für Feature im Fahrzeug	Datenvorverarbeitung mehrerer Sensorquellen
(2) Vorschläge für fahrzeuginterne Features	Entscheidungsalgorithmen in der Cloud berechnen Vorschläge als Input für Feature im Fahrzeug	Fahrzeuggeschwindigkeit auf Basis von Verkehrsdaten, Fahrwerkmodus auf Basis von Straßendaten
(3) Regelstrategie - COTA	Regelungen, deren Regelschleifen vom Fahrzeug über die Cloud ausgeführt werden	HLK-Regelung
(4) Vorhersagemodellierung	Rechenintensive Feature auf Basis einer Vielzahl an Inputquellen	Lebensdauermodelle von Batterien, Reichweitenmodelle

eines Stadtbusses wird ebenfalls diesem Use Case COTA zugeordnet. Dabei soll der äußere Regelkreis der kaskadierten Klimaregelung (s. Abbildung 3.8) in die Cloud verlagert und somit eine Regelschleife vom Fahrzeug über die Cloud und zurück geschlossen werden [100].

5.2 Die Auswahl des Reglers

Der kaskadierte HLK-Regler bietet den Vorteil, dass nur der äußere Regelkreis ersetzt werden kann, ohne dabei Einfluss auf die tatsächliche innere Regelung einzelner Komponenten innerhalb des HLK-Systems Einfluss nehmen zu müssen (vgl. Kapitel 3.4). Dieser Umstand ermöglicht den Einsatz eines neuen Reglers im äußeren Regelkreis, welcher für ein breites Spektrum von HLK-Systemen in Stadtbussen mit unterschiedlichen Antriebssträngen

Tabelle 5.2: Einordnung der Use Cases aus dem Stand der Wissenschaft und Technik in die Use Case-Kategorien aus Tabelle 5.1

Publikation	**Beschreibung**	**Kategorie**	**Begründung**
Ashok et al. [9]	Gestenerkennung	(1)	Cloud als Rechenressource für Fahrzeug-Infotainment
Deng et al. [33]	Verkehrsaufkommensüberwachung in der Cloud	(1)	Cloud als Rechenressource für Fahrzeug-Navigation
Wang et al. [125]	Fahrerassistenzsystem mit digitalem Zwilling	(2)	Anzeige von Vorschlägen für Aktorwerte
Yang et al. [134]	Energieoptimierung für Plug-in-Hybrid-Stadtbusse	(3)	Regelschleife über die Cloud

einsetzbar ist, solange die benötigten Messwerte verfügbar sind, die Regelgrößen des HLK-Systems identisch bleiben und die kaskadierte Regelstruktur beibehalten wird.

Für die Umsetzung des cloudbasierten HLK-Reglers des äußeren Regelkreises kommen die in Kapitel 3.4 beschriebenen Ansätze der *Fuzzy-Regelung*, der *modellprädiktiven Regelung*, der *PID-Regelung* und des *RL-Agenten* in Betracht.

Typische Metriken zur Bewertung der Performanz von HLK-Reglern sind [3]:

- Energie- und Kosteneinsparungen
- Verbesserung des Einschwingverhaltens (Verkürzung der Anstiegszeit, Einschwingzeit und Spitzenzeit)

- Steuerung von Variablen innerhalb von Grenzen
- Verringerung der Schwankungen von einem Sollwert
- Verbesserung des Wirkungsgrads und des COP
- Robustheit gegenüber Störungen und Änderungen der Betriebsbedingungen
- Verbesserung der Raumluftqualität und des thermischen Komforts
- Verkürzung der Berechnungszeit

Diese Metriken werden für die Definition eigener Anforderungen herangezogen. Die Anforderungen sind in *MUSS* und *SOLL* Anforderungen aufgeteilt, die entsprechend doppelt oder einfach in der Bewertungstabelle (s. Tabelle 5.3) gewichtet werden:

Req-1 (Regler) **Energieeffizienz [MUSS]:** Aufgrund des hohen Beitrags der HLK zum Gesamtenergieverbrauch des BEB (s. Kapitel 3.5) muss die elektrische Leistungsaufnahme des HLK-Systems möglichst gering gehalten werden.

Req-2 (Regler) **Thermischer Komfort [MUSS]:** Der thermische Komfort für die Passagiere ist die Hauptaufgabe des HLK-Systems und muss den Vorgaben der VDV-Schrift 236 (s. Kapitel 2.4.3) entsprechen.

Req-3 (Regler) **Berücksichtigung von Beschränkungen [MUSS]:** Die Regelung muss physikalische, technische und sicherheitsrelevante Einschränkungen der HLK-Komponenten, wie beispielsweise maximaler Wärmestrom der Wärmepumpe, einhalten. Diese Systembeschränkungen müssen beim Entwurf und in der Laufzeit des Reglers berücksichtigt werden können, um eine realisierbare, sichere und verschleißarme Betriebsführung zu gewährleisten.

Req-4 (Regler) **Nutzbarkeit von Störgrößenprädiktionen [SOLL]:** Störgrößenprädiktionen, wie beispielsweise die Passagieranzahl oder Türöffnungen, können die Regelung optimieren (vgl. [tSR21]). Insbesondere im Stadtbus kann durch festgelegte Routen und Haltestellen eine Prädiktion ein großes Potenzial erzeugen und soll deshalb genutzt werden können.

Req-5 (Regler) **Konfigurier- und Übertragbarkeit [SOLL]:** Der Regler soll auf verschiedene HLK-Anlagen konfigurierbar und bei Bedarf erweiterbar sein.

Tabelle 5.3: Bewertung der potenziellen HLK-Regler auf Basis von [17]

Anforderung	Gewicht	Fuzzy	PID	MPC	RL
		Punkte	Punkte	Punkte	Punkte
Energieeffizienz [MUSS]	2	8	10	18	16
Thermischer Komfort [MUSS]	2	10	14	18	16
Berücksichtigung von Beschränkungen [MUSS]	2	4	8	20	16
Nutzbarkeit von Störgrößenprädiktionen [SOLL]	1	4	2	10	10
Konfigurier- und Übertragbarkeit [SOLL]	1	8	7	5	3
Gesamtpunkte	-	**34**	**41**	**71**	**61**

Bewertungsskala: 1 Punkt (nicht erfüllt/ungenügend) bis 10 Punkte (vollständig erfüllt/sehr gut)

Die Wahl des cloudbasierten Reglers fällt zugunsten des MPC Ansatzes. Diese Entscheidung basiert zum einen auf den in Kapitel 3.4 erwähnten wissenschaftlichen Erkenntnissen in der Anwendung des MPC, aber hauptsächlich auf der nachfolgenden Erfüllung der gestellten Anforderungen

durch den MPC. Hinsichtlich der Energieeffizienz (**Req-1 (Regler)**) lässt sich die elektrische Leistungsaufnahme des HLK-Systems direkt als Kostenkriterium in die Optimierung integrieren. Der MPC kann somit energetisch günstige Stellgrößenverläufe bevorzugen, ohne den thermischen Komfort zu beeinträchtigen, der im Mittelpunkt der Anforderung (**Req-2 (Regler)**) steht. Die Komfortbedingungen gemäß der VDV-Schrift 236 lassen sich dabei entweder als Nebenbedingungen oder als Komfortmetriken in der Zielfunktion abbilden. Die Berücksichtigung physikalischer Beschränkungen (**Req-3 (Regler)**) stellt eine Kernkompetenz des MPC dar. Systemgrenzen wie maximale Kompressorleistung, Temperaturbandbreiten oder Stellgrößenänderungsraten werden in Form von Nebenbedingungen berücksichtigt und in jeder Optimierungsiteration eingehalten. Ähnliches gilt für die Nutzung von Störgrößenprädiktionen (**Req-4 (Regler)**), die beim MPC explizit in das Optimierungsproblem eingebunden werden können, während die anderen genannten Regler entweder reaktiv arbeiten (wie der PID) oder heuristische Regeln verwenden (z.B. Fuzzy). RL-Agenten bieten durch das Lernen von Störgrößen eine vergleichbare Nutzbarkeit der Störgrößenprädiktion. Nicht zuletzt trägt die Konfigurier- und Übertragbarkeit (**Req-5 (Regler)**) zur Flexibilität des Reglers bei. Da der MPC auf einem mathematischen Modell basiert, kann er durch Anpassung dieses Modells und der entsprechenden Parameter auf andere HLK-Systeme oder Fahrzeugplattformen übertragen werden. Die regelungstechnische Struktur bleibt erhalten, wodurch eine einfache Skalierbarkeit und Erweiterbarkeit ermöglicht werden. Hier schneidet insbesondere der RL-Ansatz aufgrund der Notwendigkeit eines erneuten Lernvorgangs des Agenten nicht gut ab.

5.3 Die Regelstrecke

5.3.1 Das Fahrzeugkabinenmodell

Die Fahrzeugkabine und die sich einstellende Kabinentemperatur werden mithilfe der thermischen Beschreibung aus Kapitel 2.4.4 modelliert. Es wird ein Solobus (s. Kapitel 2.4.1) mit den Eigenschaften aus Tabelle 5.4 modelliert. Das Systemverhalten der Fahrzeugkabine kann in der Form eines impliziten differenzial-algebraischen Gleichungssystems beschrieben werden. Dieses berücksichtigt die Energiebilanz in Form der Differenzialgleichung 2.34, die Feuchtebilanz und den Zusammenhang zwischen allen Wärmeströmen aus Gleichung 2.35:

$$F\left(\dot{x}(t), x(t), y(t), u(t), \Theta\right) = 0 \tag{5.1}$$

mit:

$$x = \begin{bmatrix} T_{\mathrm{Kab}} \\ x_{\mathrm{Kab}} \end{bmatrix}, \qquad u = \begin{bmatrix} T_{\mathrm{Umgebung}} \\ x_{\mathrm{Umgebung}} \\ \dot{Q}_{\mathrm{HLK}} \\ \dot{Q}_{\mathrm{Solar}} \\ \dot{m}_{\mathrm{Frischluft}} \\ N_{\mathrm{Pass}} \\ N_{\mathrm{Türen}} \end{bmatrix} \tag{5.2}$$

$$y = \begin{bmatrix} T_{\mathrm{Kab}} \end{bmatrix}, \qquad \Theta = \begin{bmatrix} \Theta_1 \\ \vdots \\ \Theta_{\mathrm{n}} \end{bmatrix} \tag{5.3}$$

Abbildung 5.1: Batterieelektrischer Stadtbus mit 3 Türen [41]

Dabei ist $\boldsymbol{x}$ der Zustandsvektor, $\boldsymbol{y}$ der Ausgangsgrößenvektor, $\boldsymbol{u}$ der Eingangsgrößenvektor und $\boldsymbol{\Theta}$ der Parametervektor. Das Kabinenmodell hat insgesamt sieben Eingangsgrößen: die Umgebungstemperatur T_{Umgebung}, den Feuchtegrad der Umgebung x_{Umgebung}, den bereitgestellten Wärmestrom des HLK-Systems $\dot{Q}_{\text{HLK}}$, die Solarstrahlung $\dot{Q}_{\text{Solar}}$, den Frischluftmassenstrom $\dot{m}_{\text{Frischluft}}$, die Anzahl der Passagiere $N_{\text{Passagiere}}$, und die Anzahl der geöffneten Türen $N_{\text{Türen}}$ (s. Abbildung 5.1). Die Ausgangsgröße ist die Temperatur in der Fahrzeugkabine T_{Kab}. Die Modellparameter $\boldsymbol{\Theta}$ (s. Tabelle 5.4) werden als zeitlich konstant angenommen.

5.3.2 Wärmepumpe

Das HLK-System besteht aus einer Aufdachwärmepumpe für den Kühl- und Heizbetrieb ohne Zuheizer (s. Tabelle 2.2). Die Modellierung der Wärmepumpe basiert auf dem Ansatz aus der Dissertation von Rösch [100], die anstatt der vollständigen physikalischen Modellierung auf ein COP-Kennfeld setzt (s. Kapitel A.2). Mittels Herstellerangaben[1] konnte das Kennfeld für den Umgebungstemperaturbereich $-15\,^{\circ}\text{C}$ bis $25\,^{\circ}\text{C}$ erstellt werden (s. Abbildung 5.3). Mittels des Kennfelds lässt sich durch Interpolation, basierend auf der gegebenen Umgebungstemperatur und dem angeforderten Wärmestrom,

[1] Konvekta 700EM CO_2 HP: `https://www.konvekta.de/fileadmin/user_upload/docs/datenblaetter/busse/UltraLight_500-700EM_CO2_HP_dt_engl_0722.pdf`

Tabelle 5.4: Fahrzeugparameter und Modellparameter der Fahrzeugkabine nach [39]

Fahrzeugparameter	**Wert**
Minimale Passagieranzahl	1
Maximale Passagieranzahl	85
Anzahl Türen	3
Modellparameter der Fahrzeugkabine	**Wert**
Masse der Luft in der Fahrzeugkabine	$55\,\mathrm{kg}$
spezifische Wärmekapazität Luft Fahrzeugkabine	$1.005\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Oberfläche Fahrzeugkabine	$100\,\mathrm{m^2}$
Masse Interieur	$400\,\mathrm{kg}$
Konvektionskoeffizient Interieur	$2\,\mathrm{kW\,K^{-1}}$
spezifische Wärmekapazität Interieur	$966\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Emissionskonstante der Kabinenoberfläche	0.95

der tatsächlich bereitgestellte Wärmestrom sowie die dafür erforderliche elektrische Leistung bestimmen. Liegt der angeforderte Wärmestrom jedoch unterhalb der minimal möglichen Heizleistung $\dot{Q}_{\min}$, so liefert die Wärmepumpe weiterhin mindestens $\dot{Q}_{\min}$. Ein linearer Betriebspunkt unterhalb dieser Grenze ist somit nicht erreichbar. In der Praxis wird überschüssige Wärme zunächst in das Heizmedium eingebracht; überschreitet dieses die Solltemperatur, schaltet die Wärmepumpe ab und bei erneutem Bedarf wieder zu. Das resultiert in einem Takten der Anlage, sofern keine weiteren Regelstrategien wie die Abgabe der Überschusswärme an andere Kreise oder eine vollständige Abschaltung greifen. Das dynamische Verhalten der Wärmepumpe kann durch ein Verzögerungsglied erster Ordnung (PT1-Glied) mit einer Zeitkonstante $\tau_{\text{Wärmepumpe}} = 20\,\mathrm{s}$ beschrieben werden (s. Abbildung A.19).

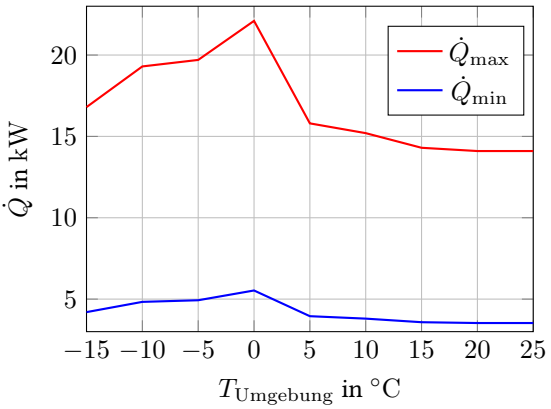


Abbildung 5.2: Minimale und maximale Wärmeleistung der modellierten Wärmepumpe

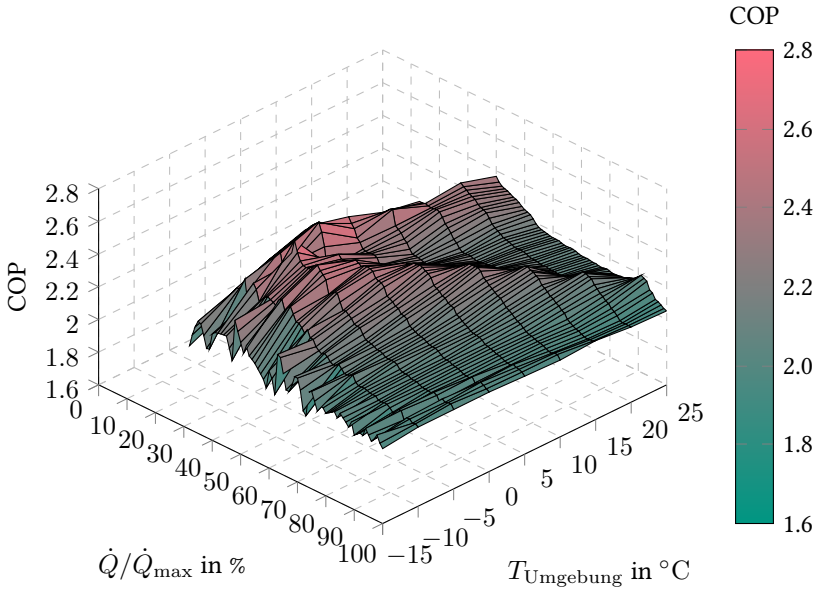


Abbildung 5.3: COP-Kennfeld der modellierten Wärmepumpe

5.4 Aufbau des MPC-Reglers

In Abbildung 5.4 ist das Funktionsschema des MPC dargestellt. Die wesentlichen Bestandteile des Reglers sind: Optimierung der Kostenfunktion und Prädiktion des dynamischen Systemverhaltens der Regelstrecke bestehend aus Fahrzeugkabine und Wärmepumpe. Im Folgenden wird das im MPC verwendete Modell als Anlagenmodell bezeichnet, während das Modell, das das Verhalten der realen Anlage (Regelstrecke) in der Simulation abbildet, als Simulationsmodell bezeichnet wird. Die zugrunde liegenden Modelle, das Kabinenmodell und die Wärmepumpe, sind in beiden Fällen identisch, wobei im Simulationsmodell zur Berücksichtigung realer Betriebsbedingungen ein additives gaußsches Rauschen hinzugefügt wird. Die Optimierung berücksichtigt Nebenbedingungen der Stellgrößen wie die minimale und maximale Leistung der Wärmepumpe. Die Kostenfunktion beinhaltet die Definition von Komfort (s. Kapitel 2.4.3) und die Festlegung der Regelparameter wie der Gewichtung einzelner Bestandteile der Kostenfunktion. Die Prädiktion von Störgrößen kann in das Anlagenmodell einfließen und die Performance des Reglers erhöhen. Die Rückkopplung von Messdaten aus der Regelstrecke schließt den Regelkreis.

Die in Kapitel 5.3.1 eingeführten Größen zur Beschreibung des Kabinenmodells werden verwendet, um eine Notation des MPC Schemas zu entwickeln (s. Abbildung 5.4). Die Eingangsgrößen $\boldsymbol{u}$ werden in manipulierbare Stellgrößen $\boldsymbol{u}$ und nicht-manipulierbare Störgrößen unterteilt. Die Störgrößen wiederum werden in messbare (vorhersagbare) $\boldsymbol{d}$ und nicht-messbare (nicht-vorhersagbare) Störgrößen $\boldsymbol{z}$ unterteilt. Vorhergesagte Größen werden mit dem Dach-Symbol gekennzeichnet (z.B. vorhergesagte Ausgangsgrößen des Anlagenmodells).

Die Systemgleichungen des Kabinenmodells werden aufgrund der zeitdiskreten Funktionsweise des Reglers diskretisiert. Das bedeutet, die kontinuierliche Zeit $t \in [t_0, t_\mathrm{p}]$ mit dem Prädiktionshorizont t_p wird mit dem

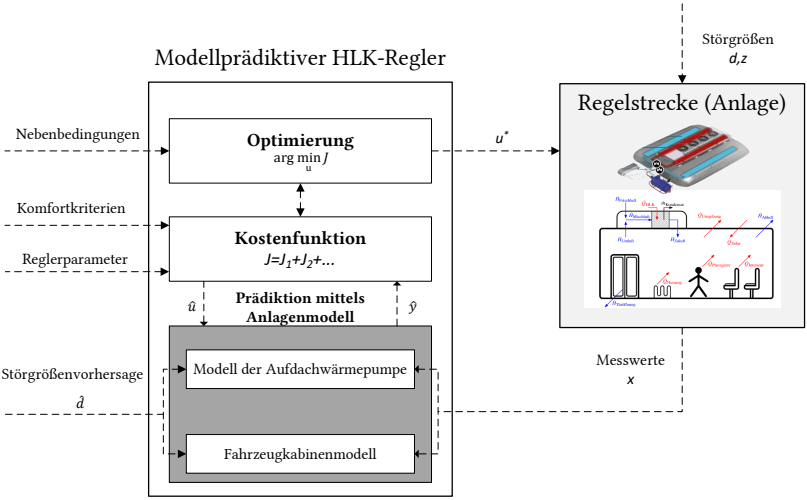


Abbildung 5.4: Schematischer Aufbau der modellprädiktiven HLK-Regelung im Stadtbus (angelehnt an [42])

Tabelle 5.5: Beschreibung der Notationen des kaskadierten äußeren HLK-Regelkreises

Notation	Beschreibung	Zugeordnete Größe
$\boldsymbol{x}$	Zustandsgrößen	Temperatur in der Fahrzeugkabine, Feuchtegrad der Kabine
$\boldsymbol{y}$	Ausgangsgrößen	Bereitgestellter Wärmestrom der Wärmepumpe
$\boldsymbol{u}$	Stellgrößen	Soll-Wärmestrom der Wärmepumpe
$\boldsymbol{d}$	messbare Störgrößen	Außentemperatur, Solarstrahlung, Feuchtegrad der Umgebung
$\boldsymbol{z}$	nicht-messbare Störgrößen	Verhalten der Passagieranzahl, Türöffnungen
$\boldsymbol{\Theta}$	Modellparameter	Werte der thermischen Kapazitäten

Abtastintervall des Reglers Δt_{MPC} in die Zeit $k \in [0, 1, ..., N_{\mathrm{p}} - 1]$ überführt (es gilt: $N_{\mathrm{p}} = \frac{t_{\mathrm{p}}}{\Delta t_{\mathrm{MPC}}}$). Alle nachfolgenden Gleichungen werden deshalb in zeitdiskreter Form dargestellt.

Die Kostenfunktion des MPC-Reglers

Die Kostenfunktion besteht aus den Bestandteilen Energiekosten $J_{\mathrm{EK}}(k)$, den Komfortkosten $J_{\mathrm{KK}}(k)$ und den Kosten für die Änderung der Stellgrößen $J_{\Delta \mathrm{u}}(k)$. Das Optimierungsproblem lässt sich dadurch folgendermaßen formulieren:

$$\boldsymbol{u}^*(k) = \arg\min_{\hat{u}(k)} \gamma_{\mathrm{EK}} J_{\mathrm{EK}}(k) + \gamma_{\mathrm{KK}} J_{\mathrm{KK}}(k) + \gamma_{\Delta \mathrm{u}} J_{\Delta \mathrm{u}}(k) \tag{5.4}$$

mit den Nebenbedingungen:

$$F\left(\hat{\dot{x}}_{\mathrm{i}}(k), \hat{x}_{\mathrm{i}}(k), \hat{y}_{\mathrm{i}}(k), \hat{u}(k), \hat{d}_{\mathrm{i}}(k), \Theta_{\mathrm{i}}\right) = 0 \tag{5.5}$$

$$u_{\min} \leq \hat{u}(k) \leq u_{\max} \tag{5.6}$$

Die Nebenbedingungen basieren auf der Modellierung des Kabinenmodells (s. Kapitel 5.3.1) als zeitdiskretes System und den Nebenbedingungen der Stellgröße $\hat{u}(k)$. Die Grenzen der Stellgröße der Wärmepumpe (s. Abbildung 5.2) sind als harte Beschränkung zu jedem Zeitpunkt einzuhalten. Die Minimierung der Kostenfunktion liefert die optimierte Stellgrößenfolge $\boldsymbol{u}^*(k)$ über den Prädiktionshorizont. Die Gewichtungsfaktoren γ_{EK},γ_{KK} und $\gamma_{\Delta \mathrm{u}}$ müssen aufgrund unterschiedlicher Größenordnungen der Bestandteile der Kostenfunktion skaliert werden.

Energiekosten

Die Energiekosten berechnen sich aus der Leistungsaufnahme der Aufdachwärmepumpe:

$$J_{\mathrm{EK}} = \sum_{k=0}^{N_{\mathrm{p}-1}} \hat{u}_{\mathrm{WP}}(k) \tag{5.7}$$

Komfortkosten

Komfortkosten berechnen sich aus der weichen Beschränkung der Regelgröße Temperatur in der Fahrzeugkabine T_{Kab}. Die Komfortgrenzen orientieren sich an den Vorgaben der VDV-Schrift 236 (s. Kapitel 2.4.3). Diese beinhaltet die Unterscheidung zwischen Heiz- und Kühlbetrieb. Während im Heizbetrieb die Solltemperatur T_{Soll} um $\pm 2\ K$ angepasst werden darf, ist im Kühlfall nur eine Änderung der Starttemperatur um $\pm 2\ K$ möglich. Es wird also die Variable m für den Modus eingeführt:

- $m = 1$ im Heizbetrieb
- $m = 0$ im Kühlbetrieb

Die Kostenfunktion gestaltet sich folgendermaßen:

$$J_{\mathrm{KK}} = \sum_{k=0}^{N_{\mathrm{p}-1}} \hat{s}_{\mathrm{Kab}}(k)^2 \tag{5.8}$$

mit

$$\hat{s}_{\mathrm{Kab}}(k) = m \cdot \max(0, |T_{\mathrm{Kab}} - T_{\mathrm{Soll}}^{\mathrm{Heiz}}| - 2) + (1 - m) \cdot \max(0, |T_{\mathrm{Kab}} - T_{\mathrm{Soll}}^{\mathrm{Kühl}}| - 2) \tag{5.9}$$

$$\hat{s}_{\mathrm{Kab}}(k) \geq 0 \tag{5.10}$$

Kosten der Stellgrößenänderung

Die Kosten für Stellgrößenänderungen verhindern, dass sich die optimierte Stellgröße sehr schnell oder häufig ändert. Dadurch kann ein übermäßig

schnelles Hoch- und Runterfahren des Kompressors der Wärmepumpe verhindert werden.

$$J_{\Delta u} = \sum_{k=0}^{N_{\mathrm{p}-1}} |\Delta \hat{u}(k)| \tag{5.11}$$

mit

$$\Delta \hat{u}(k) = \hat{u}(k) - \hat{u}(k-1) \tag{5.12}$$

5.5 Die serviceorientierte Architektur

Die in Kapitel 2.3.1 erläuterte signalorientierte Architektur erweist sich aufgrund ihrer statischen Kommunikationsstruktur als ungeeignet für eine dynamische Erweiterung des Funktionsumfangs über eine Cloudanbindung (vgl. [107]). In dieser klassischen Architektur sind die Kommunikationsbeziehungen zwischen Steuergeräten fest in einer Kommunikationsmatrix hinterlegt. Das Hinzufügen eines Features bedingt einen K-Matrix Release. Infolgedessen muss in Steuergeräteprojekten die Basissoftware, insbesondere die Komponenten, die die Netzwerkschnittstellen betreffen, an die veränderte Kommunikation angepasst werden [124]. Dieser Aufwand fällt in der SOA (s. Kapitel 2.3.2), bei der das System aus lose gekoppelten Komponenten besteht, weg.

Zentraler Vorteil der SOA ist die Fähigkeit, Kommunikationsbeziehungen dynamisch zur Laufzeit zu etablieren, anstatt sie im Vorfeld definieren zu müssen. Diese Herangehensweise erleichtert die Integration neuer cloudbasierter Fahrzeugfunktionen.

Für die Umsetzung einer solchen SOA wird die Implementierung von ROS 2 (s. Kapitel A.7) in Kombination mit der zugrunde liegenden DDS-Middleware bevorzugt. ROS 2 ermöglicht die dynamische Registrierung, Entdeckung und Konfiguration von Services (s. Definition 2.2) sowohl im Fahrzeug als

auch in der Cloud. Für die Kommunikation zwischen Fahrzeug und Cloud kommt anstelle von DDS das leichtgewichtigere Message Queuing Telemetry Transport (MQTT)-Protokoll (s. Kapitel A.9) zum Einsatz, um Bandbreite zu schonen und robust Daten auszutauschen.

Insgesamt bietet die SOA damit die notwendige technologische Grundlage, um bestehende E/E-Architekturen in Richtung Cloudfähigkeit weiterzuentwickeln – und zwar skalierbar, wartbar und zukunftssicher.

Software Orchestrator

Der *Software Orchestrator* (s. Abbildung 5.5) dient als zentrales Modul innerhalb der SOA im Fahrzeug. Das nachfolgende Konzept erweitert die Arbeit von Schindewolf [106] um die Anbindung des Fahrzeugs an die Cloud [tGSS25]. Dabei muss der *Orchestrator* nicht nur das interne Netzwerk und die zugehörigen Services verwalten, sondern auch eine sichere Verbindung zu externen Netzwerken herstellen können. Diese Netzwerke müssen verwaltet und deren erreichbare Services dem Fahrzeug verfügbar gemacht werden. Der *Software Orchestrator* besteht aus den folgenden Komponenten:

- **Connection-Manager**: Verwaltet die Verbindung zu externen Netzwerken auf Basis der Informationen aus der Network-Database. Er führt Statusprüfungen durch, registriert verfügbare Services in der Service-Database und übernimmt sowohl die Authentifizierung als auch die Sicherstellung einer verschlüsselten Kommunikation.
- **Execution-Manager**: Verantwortlich für die Ausführung von Features. Vor der Ausführung prüft er anhand der Feature- und Service-Database, ob die benötigten Services am vorgesehenen Ausführungsort verfügbar sind.
- **Feature-Updater**: Verwaltet die Feature-Database und ermöglicht die dynamische Anpassung des Funktionsumfangs über OTA-Updates.

Features können dadurch hinzugefügt, modifiziert oder entfernt werden.

- **Status-Check und Fault-Manager**: Der Status-Check überprüft regelmäßig die Verfügbarkeit bekannter Services und erkennt neue, integrierbare Services. Die Ergebnisse werden in der Service-Database aktualisiert. Bei einem Serviceausfall wird der Fault-Manager aktiviert, um geeignete Maßnahmen einzuleiten.
- **Orchestrator-Plotter**: Visualisiert Systemdaten zur Laufzeit und ermöglicht so die Überwachung und Evaluierung des Gesamtsystems.
- **Network-Database**: Enthält alle bekannten Netzwerke einschließlich zugehöriger Authentifizierungsinformationen.
- **Service-Database**: Enthält alle registrierten Services mit ihrer Bezeichnung, dem zugehörigen Service-Typ sowie dem aktuellen Status (z.B. aktiv, inaktiv).
- **Feature-Database**: Listet alle vom Orchestrator ausführbaren Features.
- **Arbiter**: Wird vom Execution-Manager aufgerufen, wenn ein Service aufgrund des Deploymentmodells sowohl lokal im Fahrzeug als auch in der Cloud ausgeführt werden kann. Der Arbiter entscheidet auf Basis aktueller Netzwerkparameter wie der Round-Trip-Time (RTT) über den geeignetsten Ausführungsort.

Neben dem eigentlichen Orchestrator existiert der **Orchestrator-Manager** als eigenständige Komponente außerhalb des zentralen Moduls. Er stellt Service-Clients für den *Connection-Manager* und den *Execution-Manager* bereit und ermöglicht es externen Komponenten, auf deren Funktionalität zuzugreifen. Der Orchestrator-Manager übernimmt damit eine koordinierende Rolle und fungiert als Schnittstelle zwischen außenstehenden Modulen (wie z.B. cloudbasierten Alternativen oder Diagnosesystemen) und den internen Steuerkomponenten des Orchestrators. Über ihn können gezielt Aktionen

wie das Starten von Features oder das Anstoßen von Netzwerkverbindungen initiiert werden, ohne dass direkte Zugriffe auf den Orchestrator selbst notwendig sind.

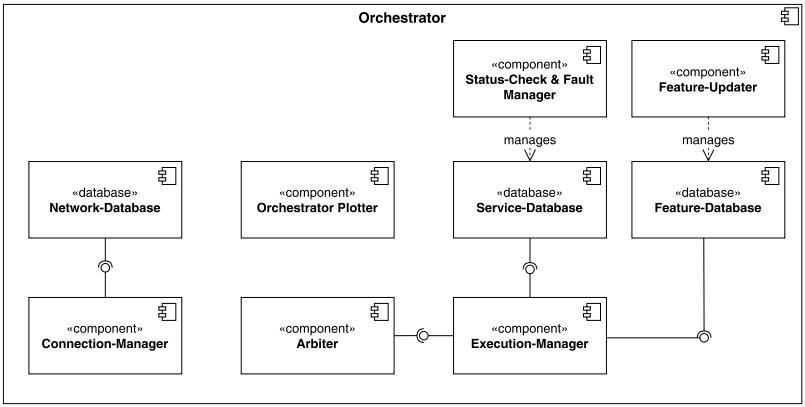


Abbildung 5.5: Komponentendiagramm des Orchestrators

5.6 Das Deploymentmodell

Das Konzept der cloudbasierten HLK-Regelung wird durch die Entscheidung des Deploymentmodells (s. Definition 5.1) beeinflusst.

> **Definition 5.1 Deploymentmodell cloudbasierter Funktionen:** Ein Deploymentmodell beschreibt die strukturelle und funktionale Architektur einer Softwareanwendung – insbesondere, wie und wo eine SWC bereitgestellt (engl. deployed) und ausgeführt wird. Im Kontext cloudbasierter Fahrzeugfunktionen legt es fest, ob eine Funktion ausschließlich in der Cloud, lokal im Fahrzeug oder in einer Kombination aus beidem bereitgestellt und ausgeführt wird.

Die vier bekannten Deploymentmodelle *Nur Cloud*, *Fallback*, *Dupliziert* und *Elastisch* (vgl. Kapitel 3.2) werden um das Deploymentmodell *Parallele Ausführung* [RtD+25] erweitert. Dabei wird die Funktion gleichzeitig auf zwei Ausführungsknoten, im Fahrzeug und in der Cloud, betrieben. Ziel ist die Erzeugung redundanter Ergebnisse, die zur Laufzeit abgeglichen werden können.

Dieses Modell erhöht die Ausfallsicherheit und erlaubt die sofortige Detektion von Abweichungen. Im Vergleich zum duplizierten Modell, bei dem nur ein Knoten (Cloud oder Fahrzeug) aktiv ist, sind hier beide Knoten permanent aktiv. Dadurch eignet sich das Modell besonders für sicherheitskritische, latenzempfindliche oder fehlertolerante Anwendungen, wie z. B. Fahrerassistenzsysteme oder KI-gestützte Entscheidungsprozesse.

- Parallele Ausführung: Die Funktion wird auf zwei Ausführungsknoten (Fahrzeug und Cloud) parallel bereitgestellt und ausgeführt, um redundante Ergebnisse zu erzielen.

Entsprechend der Ontologie aus Kapitel 3.3 können alle Deploymentmodelle bis auf das *Nur Cloud* Modell als *dynamisch* eingestuft werden.

5.7 MCDA zur Identifikation des optimalen Deploymentmodells cloudbasierter Funktionen

5.7.1 Die Komponenten der MCDA

Die Komponenten der MCDA (s. Kapitel 2.7) sind in vier hierarchische Ebenen gegliedert (s. Abbildung 5.6). Auf der obersten Ebene (Ebene 0) steht das Hauptziel: die Identifikation eines optimalen Deploymentmodells, das den spezifischen Anforderungen im Fahrzeugkontext gerecht wird.

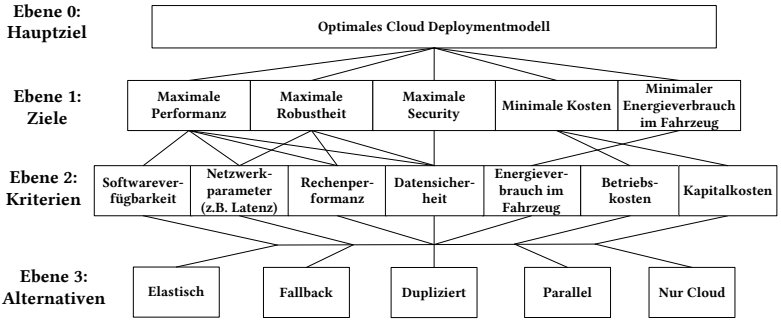


Abbildung 5.6: Aufbau der MCDA für Deploymentmodelle cloudbasierter Funktionen

Ebene 1 umfasst daraus abgeleitete zentrale Zielsetzungen. Diese reichen von der Maximierung von Performanz, Robustheit und Sicherheit bis hin zur Minimierung von Kosten und Energieverbrauch im Fahrzeug. Die Ziele spiegeln die unterschiedlichen Anforderungen an eine cloudbasierte Fahrzeugfunktion wider und stehen häufig in einem Zielkonflikt zueinander.

In Ebene 2 werden diese Ziele durch konkrete Bewertungskriterien operationalisiert. Dazu gehören unter anderem Softwareverfügbarkeit, Netzwerkparameter (z.B. Latenz), Rechenleistung, Datensicherheit, Energieverbrauch im Fahrzeug sowie Betriebs- und Kapitalkosten. Die Verbindungen zwischen Zielen und Kriterien in Abbildung 5.6 zeigen, dass ein Kriterium mehrere Ziele beeinflussen kann und umgekehrt.

Ebene 3 bildet schließlich die Handlungsebene ab, auf der die konkreten Alternativen der Deploymentmodelle bewertet werden.

Im Folgenden werden ausschließlich die Bewertungskriterien aus Ebene 2 im Detail erläutert, da sie die Grundlage für die Bewertung der Alternativen bilden, während die übergeordneten Ziele bereits implizit über deren Verknüpfung berücksichtigt sind:

Krit-1 **Verfügbarkeit/Robustheit**: Verfügbarkeit einer Funktion innerhalb des Systems und Robustheit gegenüber Komponenten- oder Netzausfällen. Eine redundant ausgeführte Funktion, die sowohl im Fahrzeug als auch in der Cloud betrieben wird, weist im Vergleich zu rein lokal oder nur cloudbasierten Deploymentmodellen eine erhöhte Verfügbarkeit und Robustheit auf. Durch die verteilte Ausführung kann bei Ausfall einer Umgebung (z.B. fehlende Netzwerkanbindung oder lokale Systemstörung) auf die jeweils andere Instanz zurückgegriffen werden.

Krit-2 **Latenz**: Die Latenz der Datenübertragung erhöht sich unter anderem durch Faktoren wie Netzüberlastung, den zu benutzenden Routing-Pfad sowie die physische Entfernung zwischen den Kommunikationspartnern. Folglich hat eine Funktion, die auf einer ECU ohne externe Netzwerkkommunikation ausgeführt wird, die geringste Latenz, während eine dynamische Funktionsverteilung (s. Kapitel 3.3) die höchste Latenz aufweist.

Krit-3 **Performanz der Ausführungsknoten**: Leistung und Skalierbarkeit der Ausführungsknoten (s. Abbildung 3.2). Die Rechen- und Speicherressourcen im Fahrzeug sind nicht skalierbar und daher begrenzt. Andererseits sind virtualisierte Einheiten wie virtuelle Maschinen in einer Cloud skalierbar.

Krit-4 **Datensicherheit**: Beschreibt die Möglichkeit, dass die Funktion selbst oder Eingabe-/Ausgabedaten von externen Parteien kompromittiert werden können. Daher weisen Funktionen, die nur im Fahrzeug verfügbar sind, den höchsten Sicherheitsgrad auf. Dynamisch verteilte Funktionen, die zwischen Ausführungsknoten hin- und herbewegt werden, weisen das höchste Risiko der Manipulationen

auf und haben daher das niedrigste Security-Level.

Krit-5 **Energieverbrauch im Fahrzeug**: Energie, die für die Ausführung einer Funktion erforderlich ist, einschließlich der Energie, die für die Datenübertragung benötigt wird, wenn eine externe Funktion angesprochen werden muss (s. Kapitel 2.6.2).

Krit-6 **Betriebskosten (OpEx)**: Kosten, die bei der Ausführung der Funktion anfallen. Die Ausführung im Fahrzeug umfasst potenzielle Kosten für OTA-Updates und Nutzungskosten wie Stromverbrauch. Die Ausführung in der Cloud führt zu Kosten beim Cloud Service Provider (s. Kapitel 4.2.3).

Krit-7 **Kapitalkosten (CapEx)**: Umfasst Software-Entwicklungs- und Hardwarekosten. Eine Funktion im Fahrzeug erfordert eine zugehörige ECU (s. Kapitel 2.2), die einmalig gekauft wird. Diese Kapitalkosten fallen in der Cloudumgebung vollständig weg. Die Softwareentwicklungskosten zwischen Cloud und Fahrzeug unterscheiden sich ebenfalls (s. Kapitel 4.2.3).

Die Erfüllung der einzelnen Kriterien für die Deploymentmodelle wird mittels einer Bewertungsskala von 0 (nicht erfüllt) bis 100 (vollständig erfüllt) durchgeführt (verbale Ordinalskala in Tabelle 5.6).

Tabelle 5.6: Entscheidungsmatrix der MCDA für Deploymentmodelle

Kriterium	Elastisch	Fallback	Dupliziert	Parallel	Nur Cloud
Krit-1 Verfügbarkeit/Robustheit	mittel (Redeployment)	sehr hoch	sehr hoch	am höchsten	am geringsten
Krit-2 Latenz	abhängig von Internetverbindung	abhängig von Internetverbindung (insb. beim Redeployment)	abhängig von Internetverbindung	gering	abhängig von Internetverbindung
Krit-3 Performanz der Ausführungsknoten	nahezu unbegrenzt	nahezu unbegrenzt	nahezu unbegrenzt	beschränkt auf Fahrzeug-Speicher- & Rechenressourcen	nahezu unbegrenzt
Krit-4 Datensicherheit	mittel	hoch	mittel	mittel	mittel
Krit-5 Energieverbrauch im Fahrzeug	mittel (Transfer der Funktion)	gering	sehr gering	maximaler Verbrauch	am geringsten
Krit-6 Betriebskosten	hoch	hoch	hoch	am höchsten	mittel
Krit-7 Kapitalkosten	hoch	hoch	am höchsten	hoch	am geringsten

5.7.2 Bewertung der Alternativen für die HLK-Regelung

Die Bewertung der Alternativlösungen des Deploymentmodells für die HLK-Regelung wird durch die Anforderungen der Funktion selbst beeinflusst (s. Kapitel 4.2.1), wobei die Qualitätsanforderungen **Req-6 (Funktion)** in Form von Flottenlernen für die Wahl des Deploymentmodells nicht von Bedeutung sind. Die Anforderungen der modellprädiktiven HLK-Regelung (s. Tabelle 5.7) werden entsprechend den Kriterien auf Ebene 2 der MCDA zugeordnet und ebenfalls mittels einer Skala von 0 (keine/niedrigste Anforderungen) bis 100 (hohe Anforderungen) erhoben. Erfüllt ein Deploymentmodell eine Anforderung vollständig, wird der zugehörige Kriterienwert auf das Maximum von 100 gesetzt. Bei einer Unterschreitung des Anforderungswerts erfolgt eine lineare Abwertung (vgl. Abbildung 5.7). Wird ein Kriterium durch mehrere Anforderungen beeinflusst, etwa *Latenz* durch Echtzeitanforderungen und Hardwareabhängigkeit, wird der Mittelwert der beteiligten Anforderungswerte verwendet, um das Deploymentmodell zu bewerten. Die Berechnung des Gesamtnutzwerts erfolgt gemäß dem WSM nach Gleichung 2.40, wobei alle Kriterienwerte a_{ij} mit identischem Gewicht w_{j} in die Bewertung eingehen. Eine prozentuale Bewertung der Deploymentmodelle kann mithilfe einer hypothetischen Funktion berechnet werden, die über die niedrigsten Anforderungen verfügt und somit von jedem Deploymentmodell vollumfänglich erfüllt wird. Die aufsummierte maximale Gesamtsumme bei den entsprechenden Gewichten wird als Referenz herangezogen.

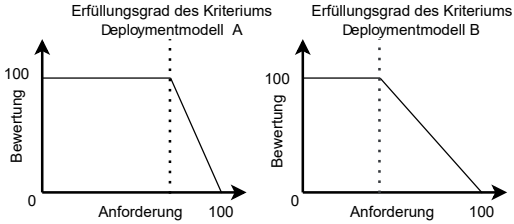


Abbildung 5.7: Bewertungsmetrik eines Kriteriums für zwei verschiedene Deploymentmodelle

Tabelle 5.7: Anforderungen der modellprädiktiven HLK-Regelung

Anforderung	Wert	Begründung
Echtzeitanforderungen (Krit-1, Krit-2)	20	Für die Regelung des thermischen Komforts ist eine zeitnahe Berechnung der Stellgrößen erforderlich. Aufgrund der trägen Dynamik der HLK-Regelung im Stadtbus (vgl. Kapitel 6.4.5) sind Mobilfunklatenzen im Millisekundenbereich (vgl. Kapitel 2.6.1) unkritisch. Die Verfügbarkeit der Funktion beeinflusst den Komfort, ist jedoch nicht sicherheitskritisch.
Safety- und Securityanforderungen (Krit-4)	20	Die Sensordaten (Temperaturmesswerte etc.) und Regelgrößen sind für den Komfort relevant und nicht sicherheitskritisch. Standard-Securitymaßnahmen bei Cloud-Deployment wie verschlüsselte Kommunikation, sichere Authentifizierung und Datenschutz.
Ressourcenanforderungen (Krit-3, Krit-5, Krit-6)	70	Die modellprädiktive Regelung erfordert die Lösung eines Optimierungsproblems in jedem Regelungsintervall, was im Vergleich zu klassischen Reglern (z.B. PID) signifikante Rechenressourcen **(Krit-3)** (vgl. auch Tabelle 4.2) beansprucht. Dies führt gleichzeitig zu einem erhöhten Energieverbrauch der ECU im Vergleich zum bisherigen PID-Regler.
Hardwareabhängigkeiten (Krit-2)	10	Die Funktion der HLK-Regelung im Stadtbus ist im Vergleich zu typischen hardwareabhängigen Treiber- oder Sensorschnittstellen nicht hardwareabhängig. Die HLK-Funktion arbeitet auf einer höheren Abstraktionsebene und berechnet Sollwerte oder Reglerausgänge basierend auf physikalischen Messgrößen (Temperatur, Feuchte etc.), ohne dabei direkt auf Register oder spezifische Hardwareschnittstellen zuzugreifen. Der Einfluss von Latenz **(Krit-2)** ist, verglichen mit der Zeitkonstante der Regelstrecke, gering.
Ökonomische Anforderungen (Krit-6, Krit-7)	50	Während Kapitalkosten aufgrund der benötigten Rechenressourcen der MPC als hoch einzustufen sind, sind die Betriebskosten aufgrund des hohen Abtastintervalls einer HLK-Regelung im Sekundenbereich (vgl. Millisekunden in der Motorsteuerung) als gering einzustufen.

Für die in Kapitel 5.4 beschriebene modellprädiktive HLK-Regelung ergibt sich folgende prozentuale Eignung der fünf verschiedenen Deploymentmodelle cloudbasierter Funktionen:

- Elastisch: 74,35 %
- Fallback: 91,68 %
- Dupliziert: 81,69 %
- Parallel: 52,12 %
- **Nur Cloud: 96,55 %**

5.8 Fazit zur Konzeptentwicklung

Das vorliegende Kapitel entwickelte ein umfassendes Konzept für eine cloudbasierte HLK-Regelung in BEB, das sowohl technische als auch architektonische Herausforderungen adressiert.

Die Wahl der modellprädiktiven Regelung (MPC) als cloudbasierter Regler erwies sich durch die systematische Anforderungsanalyse (s. Kapitel 5.2) als optimal geeignet. Mit 71 von 80 möglichen Punkten übertrifft der MPC-Ansatz deutlich die Alternativen Fuzzy (34), PID (41) und Reinforcement Learning (61). Der MPC erfüllt alle kritischen Anforderungen durch direkte Integration der Leistungsaufnahme in die Kostenfunktion, Abbildung der VDV-236-Vorgaben sowie explizite Berücksichtigung von Systembeschränkungen und Störgrößenprädiktionen.

Die serviceorientierte Architektur (SOA) (s. Kapitel 5.5) mit ROS 2 und DDS-Middleware ermöglicht die notwendige Flexibilität für cloudbasierte Fahrzeugfunktionen. Der entwickelte Software Orchestrator bildet das zentrale Modul für die dynamische Service-Verwaltung zwischen Fahrzeug und Cloud.

Die MCDA (s. Kapitel 5.7) identifizierte das *Nur Cloud*-Deploymentmodell mit $96{,}55\,\%$ als am besten geeignet für die HLK-Regelung, gefolgt vom *Fallback*-Modell mit $91{,}68\,\%$. Der geringe Bewertungsunterschied zwischen beiden Deploymentmodellen zeigt, dass das *Fallback*-Modell eine nahezu gleichwertige Alternative darstellt. Diese Bewertung resultiert aus den spezifischen Anforderungen der Anwendung: moderate Echtzeitanforderungen, geringe Security-Kritikalität, aber hohe Ressourcenanforderungen aufgrund der rechenintensiven MPC-Optimierung (s. Tabelle 5.7).

6 Prototypische Umsetzung

E/E-Architekturen befinden sich derzeit in einem tiefgreifenden Wandel. Während in der Vergangenheit stark domänenorientierte Architekturen mit einer Vielzahl dezentral verteilter ECUs vorherrschend waren, zeichnet sich im PKW-Bereich zunehmend ein Übergang hin zu zentralisierten Architekturen über den Zwischenschritt der Zonenarchitektur ab (s. Kapitel 2.3.1). Diese setzen auf zentrale Hochleistungsrechner (engl. HPC) und das Konzept der Serviceorientierung (s. Kapitel 2.3.2).

Der Stadtbus folgt dem technologischen Wandel, wenn auch mit zeitlicher Verzögerung. Dennoch ist absehbar, dass auch hier zentralisierte E/E-Architekturen Einzug halten. Parallel dazu etabliert sich die Anbindung an übergeordnete Systeme, insbesondere an Cloud-Infrastrukturen, zunehmend als Standard. Diese Verbindung wird in der Regel über eine im Fahrzeug verbaute Telematikeinheit realisiert (vgl. Abbildung A.21). Die Telematikeinheit fungiert als zentrales Gateway zwischen der fahrzeuginternen E/E-Architektur und externen Systemen. Wie in Abbildung 6.1 dargestellt, verbindet sie die verschiedenen fahrzeuginternen Sub-Netzwerke und ermöglicht über verschiedene Inter-Fahrzeug-Kommunikationsprotokolle (WLAN IEEE 802.11, LTE/5G, Dedicated Short-Range Communications (DSRC)) die Anbindung an cloudbasierte Funktionen oder andere Fahrzeuge (V2V) und Infrastrukturelemente wie Road Side Unit (RSU). Die Cloud-Infrastruktur selbst besteht aus verschiedenen cloudbasierten Funktionen, die über eine Cloud-Application Programming Interface (API) zugänglich sind. An dieser Stelle übernimmt wieder ein Gateway die Funktion der Protokollübersetzung, Zugriffskontrolle und Weiterleitung an die entsprechenden Funktionen in

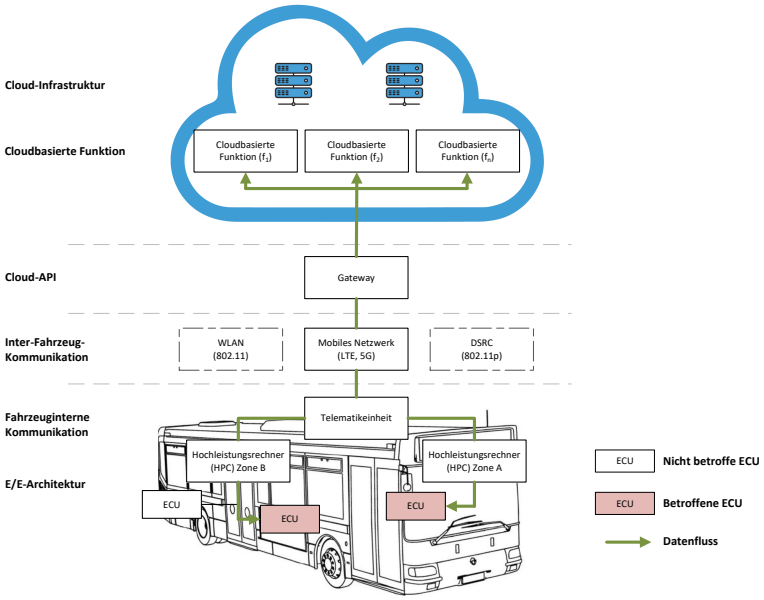


Abbildung 6.1: Cloudbasierte Funktionen im vernetzten Stadtbus (angelehnt an [81])

der Cloud. In der dargestellten Version cloudbasierter Funktionen im vernetzten Stadtbus ist ein Fall abgebildet, bei dem die cloudbasierte Funktion einen Einfluss auf einzelne ECUs im Fahrzeug hat, was jedoch laut Definition (vgl. Kapitel 4.1) nicht zwingend der Fall sein muss.

Vor dem Hintergrund dieser technologischen Entwicklungen wird im Folgenden eine E/E-Testplattform vorgestellt, die als Grundlage für die prototypische Umsetzung der in dieser Dissertation entwickelten Konzepte dient. Die Plattform bildet zentrale Aspekte moderner Fahrzeugarchitekturen ab und erlaubt deren experimentelle Validierung im Kontext des COTA-Ansatzes. Aufgrund der im vorangegangenen Kapitel 5.6 dargelegten hohen Eignung der beiden Deploymentmodelle *Nur Cloud* und *Fallback* werden auf der Plattform beide Deploymentmodelle umgesetzt. Ziel ist es, die zuvor theoretisch bewerteten Konzepte zu validieren, ihre Auswirkungen auf die Regelungs-

performanz zu untersuchen und zu klären, ob ein Fallback-Regler trotz der Ergebnisse der MCDA notwendig ist.

6.1 ATLAS Testplattform

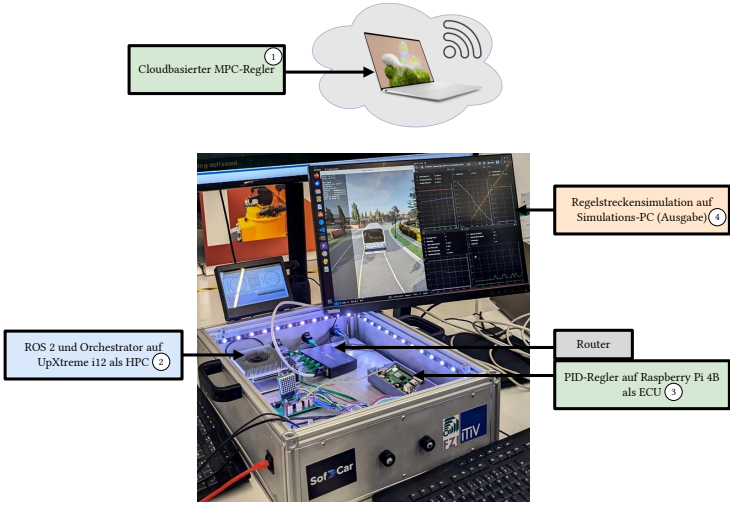


Abbildung 6.2: Die zur Validierung verwendete Testplattform ATLAS

Die ATLAS Testplattform am ITIV (s. Abbildung 6.2) stellt einen Verbund von Mikrocontroller-Boards zur Nachbildung einer vereinfachten zentralisierten Fahrzeug E/E-Architektur dar. Die Kommunikation der Boards setzt sowohl auf den CAN-Bus als auch auf IP-basierte Ethernet-Kommunikation. Der Demonstrator dient als Plattform für die Umsetzung der cloudbasierten HLK-Regelung. Zur Anwendung kommen nur die in Abbildung 6.3 dargestellten Komponenten (s. auch Tabelle 6.1).

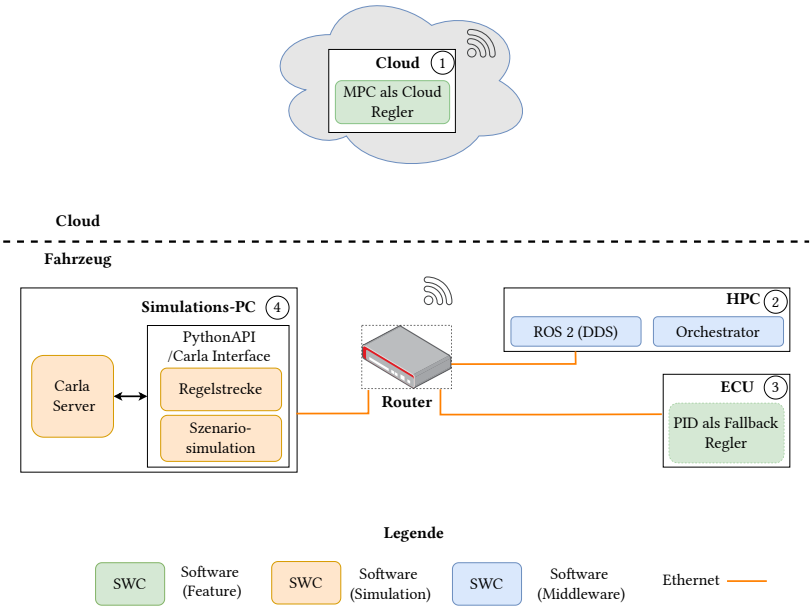


Abbildung 6.3: Hard- und Softwarekomponenten für die cloudbasierte HLK-Regelung auf der ATLAS Testplattform

Zentraler Bestandteil der Plattform ist ein UpXtreme i12 Entwicklungsboard[1], das einen HPC (2) im Fahrzeug darstellt (ca. 204.000 DMIPS vgl. Tabelle 2.1). Die Performanz des Boards bietet die Möglichkeit, verteilte und quervernetzte Features (s. Kapitel A.15) auszuführen. Der Software Orchestrator (s. Kapitel 5.5) wird hier als zentrale Middleware-Komponente (s. Kapitel A.11) ausgeführt. Eine klassische ECU (3), die hinsichtlich ihrer Leistungsfähigkeit mit einer im Fahrzeug eingesetzten ECU der hohen Leistungsklasse (ca. 24.000 DMIPS vgl. Tabelle 2.1) vergleichbar ist, wird herangezogen, um das *Fallback*-Deploymentmodell mit einem PID-Regler im Fahrzeug umzusetzen.

[1] `https://up-shop.org/DE/up-xtreme-i12-series.html`

Weiterer Bestandteil ist eine Regelstreckensimulation, ausgeführt auf einem Simulations-PC ④. Auf diesem Rechner wird die Open-Source Simulationsumgebung *Carla* [35] ausgeführt. In *Carla* wurde die Regelstrecke, bestehend aus Fahrzeugkabine und Aufdachklimaanlage, mittels des *Carla* Python API integriert. Alle Boards sind über einen *Router* in einem gemeinsamen Netzwerk organisiert.
Die *Cloud* ① ist ein performanter Laptop in einem externen Netzwerk zur Ausführung des entwickelten MPC. Die Netzwerkverbindungen zwischen ATLAS Demonstrator und *Cloud* können über Mobilfunk (4G/LTE oder 5G) hergestellt werden, sodass eine „Cellular-V2N" Kommunikation (s. Kapitel 2.6) zustande kommt. Diese Kommunikationsart stellt aufgrund der hohen Verfügbarkeit und Reichweite die stabilste Verbindung für eine V2N-Kommunikation bei einem sich mit hoher Geschwindigkeit bewegenden Fahrzeug dar.

Das POSIX-kompatible (s. Definition A.6) Betriebssystem *Ubuntu 22.04.5* wird auf dem UpXtreme i12, der *Cloud* und dem Simulations-PC ausgeführt.

Tabelle 6.1: Spezifikation der Hardware des ATLAS Testplattform

Bezeichnung	**Beschreibung**	**Prozessor**	**Kerne**	**RAM**
Cloud-PC ①	Laptop	AMD Ryzen™ 5 3500U	4	16 GB
HPC ②	UpXtreme i12 Board	12th Gen Intel ® Core™ i7-1270PE	12	16 GB
ECU ③	Raspberry Pi Model 4 B	Cortex-A72	4	8 GB
Simulations-PC ④	Desktop-PC	AMD Ryzen™ 9 7950X	16	64 GB

6.2 Die integrierten Softwarekomponenten des COTA-Ansatzes

Die Applikation der HLK-Regelung wird vom *Orchestrator* durch den Start des *HVAC-Client* mittels ROS-Launch Datei aufgerufen. Dieser beinhaltet für die beiden Deploymentmodelle jeweils einen *HVAC-Cloud-Client* und einen *HVAC-Vehicle-Client*. Die Clients (s. Abbildung 6.4) kommunizieren mit den *HVAC-Servern* (s. Abbildung 6.5) im Fall des MPC in der Cloud asynchron (Publish-Subscribe) über das MQTT Protokoll und im Fall des *HVAC-PID-Controllers* im Fahrzeug synchron (Request-Response) via dem ROS-Service *HvacControl*. Der *HVAC-MPC-Controller* in der Cloud muss seine Services aktiv dem Orchestrator kommunizieren. Dazu sendet er im aktiven Zustand die nutzbaren Services über das MQTT-Topic *cloud_services* an den *Connection-Manager* des Orchestrators. Dieser stellt sicher, dass die Services in der *Service-Database* korrekt verwaltet werden. Schlussendlich wird die Regelschleife über die beiden Topics *HvacData* und *HvacControlValues* zwischen der Simulation und den HVAC-Clients geschlossen.

Die Zuverlässigkeit der asynchronen Nachrichtenübermittlung zwischen *HVAC-Cloud-Client* und *HVAC-MPC-Controller* kann jeweils für den MQTT-Publisher und MQTT-Subscriber festgelegt werden. Die Quality of Service (QOS)-Levels der Topics sind im System wie folgt spezifiziert:

- **hvac_readings** QOS-Level 0: Die Daten werden unter möglichen Verlusten schnellstmöglich übertragen. Um sicherzustellen, dass der *HVAC-MPC-Controller* in der Cloud die Berechnung der nächsten Stellgrößen auf Basis der aktuellsten Messwerte durchführt, muss die Nachrichtenübermittlung so hochfrequent wie möglich erfolgen.
- **hvac_control_values** QOS-Level 2: Jede Nachricht wird genau einmal übermittelt. Ein wiederholtes Anwenden von Stellgrößen im Simulati-

onsmodell muss ebenso vermieden werden wie ein möglicher Verlust einer Stellgröße während der Nachrichtenübermittlung.

- **cloud_services** QOS-Level 1: Jede Nachricht wird mindestens einmal übermittelt. Die Übermittlung der Cloud-Services muss in bestimmten Zeitabständen erfolgen, da sonst der Orchestrator davon ausgeht, dass der jeweilige Cloud-Service nicht mehr zur Verfügung steht.

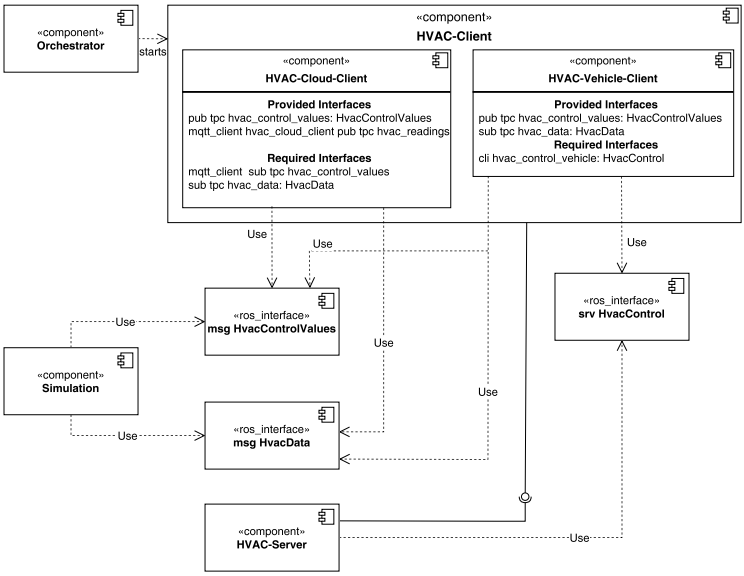


Abbildung 6.4: Komponentendiagramm des integrierten HVAC-Clients

Die Simulationsumgebung (s. Abbildung 6.6) in *Carla* umfasst neben dem Simulationsmodell der Regelstrecke (s. Kapitel 5.3) auch eine Erweiterung um Wettermesswerte über die Komponente *Dynamic_Weather*. Hier lassen sich Temperaturverläufe eines Tages inklusive des Startzeitpunkts definieren. Mithilfe des *Bus_Agent* können Bushaltestellen mit entsprechender Haltedauer konfiguriert werden (s. Abbildung A.15). Während eines Halts öffnen sich die Bustüren und das Ein- und Aussteigen von Passagieren wird

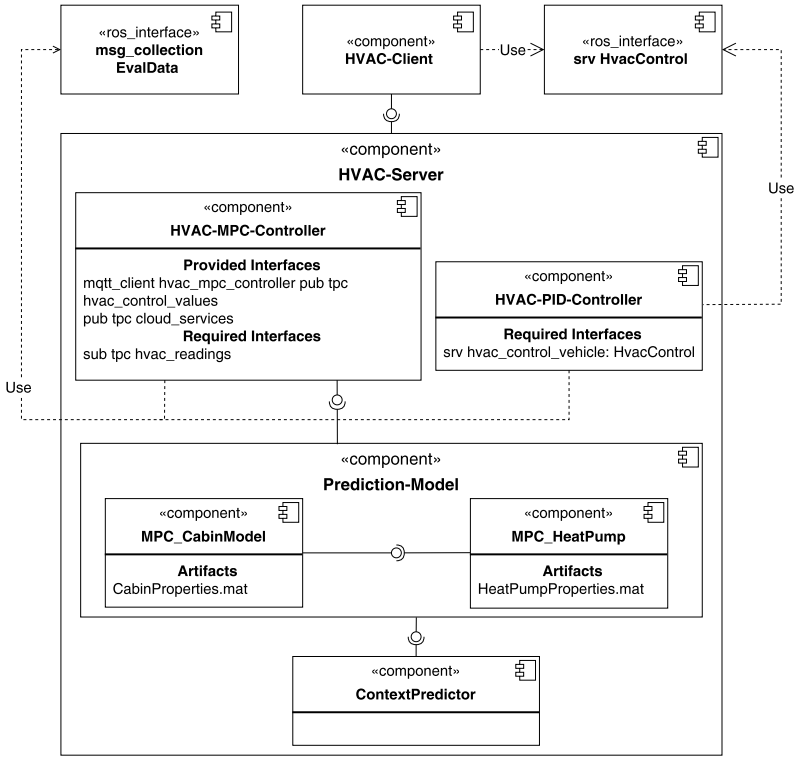


Abbildung 6.5: Komponentendiagramm des integrierten HVAC-Servers

simuliert. Damit lassen sich die in Kapitel 3.5 identifizierten Einflussfaktoren auf die HLK-Systeme innerhalb der Simulation abbilden.

Das Simulationsmodell wird mit der Methode *NextTimeStep()* und dem Zeitschritt *delta_t_sim* synchron zur CARLA-Simulation ausgeführt. Vor jedem Simulationsschritt werden über die Methode *SetContext()* die aktuellen Eingangsgrößen (s. Kapitel 5.3.1) an das Modell übergeben (vgl. Abbildung A.16 und Abbildung A.17). Die Prädiktion von Störgrößen (z.B. Passagieranzahl) kann über die Komponente *ContextPredictor* (s. Abbildung 6.5, unten) erfol-

gen. In diesem Zusammenhang wird auf die Publikation „Model Predictive HVAC Control with disturbance variable forecasting for city buses“ [tSR21] verwiesen.

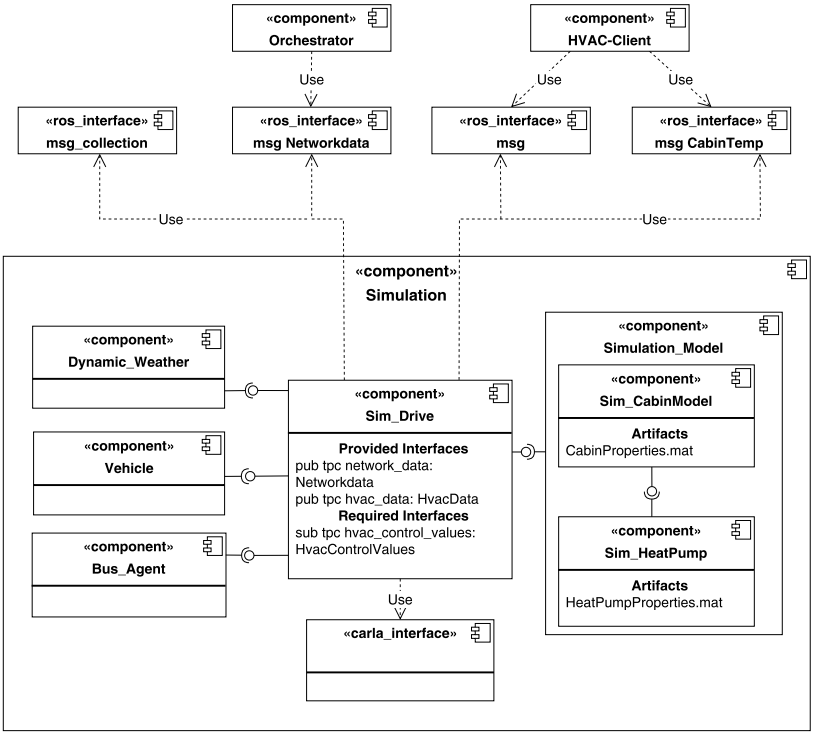


Abbildung 6.6: Komponentendiagramm der integrierten Simulationsumgebung

Ein Ausfall bzw. die Inaktivität eines Services wird vom *Status-Check und Fault-Manager* des Orchestrators (vgl. Kapitel 5.5) erkannt. Dieser vergleicht die aktiven und die registrierten Services in der *Service-Database*. Dadurch lassen sich neue Services registrieren und der Status bereits bekannter Services überprüfen. Außerdem nutzt der *Execution-Manager* einen Callback, um zu prüfen, ob alle Services zur Ausführung des Features für den jeweiligen Ausführungsort noch aktiv sind (s. Abbildung 6.7). Verbindungsausfälle werden hingegen vom *Connection-Manager* detektiert. Wurde die Verbindung zu einem anderen Netzwerk unterbrochen, versucht der *Connection-Manager* für einen definierbaren Zeitraum die Verbindung wieder aufzubauen (s. Abbildung 6.8). Kann die Verbindung nicht wiederhergestellt werden, löscht der *Connection-Manager* die entsprechenden Services aus der *Service-Database*. Sind die Services, die zur Ausführung des HLK-Cloud-Features benötigt werden, nicht mehr aktiv oder wurde die Cloud-Verbindung für den definierten Zeitraum unterbrochen, versucht der *Execution-Manager* das Feature weiterhin bereitzustellen. Hierfür wurden im *Execution-Manager*, abhängig vom Deploymentmodell, die folgenden Mechanismen implementiert:

- *Nur Cloud*: Es werden die zwischengespeicherten Stellgrößen aus dem Prädiktionshorizont des MPC zur Überbrückung der Unterbrechung verwendet. Sobald dieser *Buffer* überschritten ist, wird die letzte Stellgröße konstant gesetzt. Da die Regelschleife in diesem Fall nicht mehr geschlossen ist, handelt es sich hierbei nicht um eine Regelung, sondern eine Steuerung. Dieser Betriebsmodus wird als Buffered Prediction Horizon (BPH) bezeichnet.
- *Fallback*: Wechsel auf den lokalen Fallback PID-Regler.

Steht das HLK-Cloud-Feature nach einem Ausfall wieder zur Verfügung, können die notwendigen Services erneut dynamisch integriert werden. Die Cloud-Services sind für den *Connection-Manager* bei bestehender Verbindung zur Cloud stets sichtbar, da eine Subscription auf das Topic *cloud_services* besteht. Außerdem besitzen Services für gewöhnlich keinen Zustand, weshalb sich das Wiederherstellen eines Services auf das erneute Aufrufen

reduziert.
Innerhalb des cloudbasierten MPC erfolgt die Detektion inkonsistenter oder verzögerter Mess- und Stellgrößen über eine Plausibilitätsprüfung. Dabei wird die Gültigkeit der Stellgrößen vor ihrer Weitergabe im System sichergestellt (vgl. Abbildung A.17). Zu diesem Zweck wird ein Buffer mit historischen Stellgrößen angelegt. Dieser dient sowohl zur Überwachung der Änderungsrate zwischen aufeinanderfolgenden Stellgrößen als auch zur Kontrolle der Zeitabstände, in denen diese bereitgestellt werden. Verzögerungen bei der Übertragung von Messwerten oder bei der Berechnung der Stellgrößen wirken sich auf diese Zeitabstände aus. Wird eine berechnete Stellgröße von der Plausibilitätsprüfung als ungültig erkannt, so wird stattdessen die zuletzt als plausibel eingestufte Stellgröße verwendet.

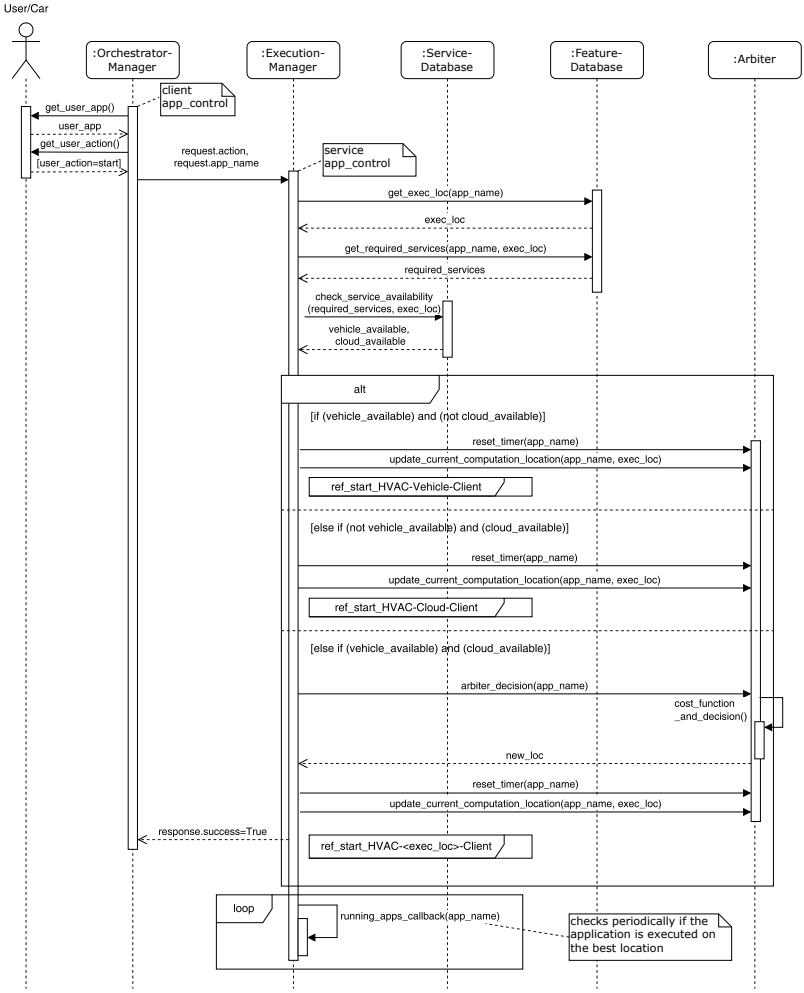


Abbildung 6.7: Sequenzdiagramm der Ausführung eines Features im integrierten System

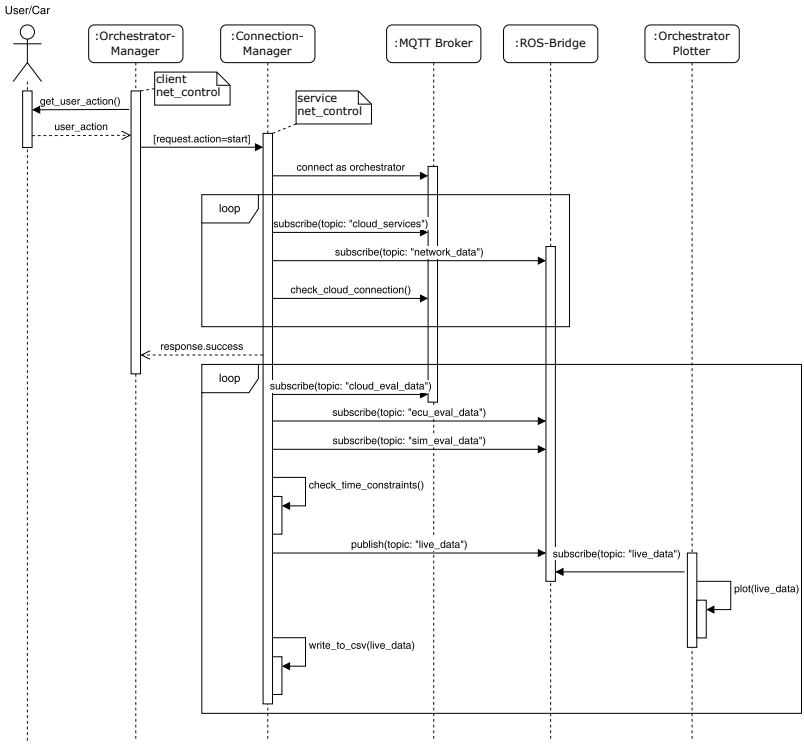


Abbildung 6.8: Sequenzdiagramm der Cloud Verbindung im integrierten System

6.3 Definition der Systemanforderungen und Testszenarien

Die Validierung der cloudbasierten HLK-Regelung wird durch eine Systemintegration und den anschließenden Test (s. Kapitel A.6) auf der ATLAS-Plattform durchgeführt.
Die Anforderungen der Validierung teilen sich folgendermaßen in funktional **F** und nichtfunktional **NF** und zugehörige Testszenarien **TS**:

Funktionale Anforderungen und zugehörige Testszenarien

F-Req-1 **Energieeffizienz:** Der Einsatz des cloudbasierten Reglers muss unter identischen thermischen Bedingungen sowie ohne Verbindungsabbrüche oder Serviceausfälle energetische Einsparpotenziale gegenüber dem Stand der Technik, einem fahrzeuginternen PID-Regler, aufzeigen.

TS 1.a Simulationsszenarien: Variation der thermischen Umgebungsbedingungen

F-Req-2 **Thermischer Komfort:** Der cloudbasierte Regler soll den thermischen Komfort bei Verbindungsabbrüchen bzw. Serviceausfällen und unter schwierigsten thermischen Umgebungsbedingungen entsprechend der Regleranforderung **Req-2 (Regler)** (s. Kapitel 5.2) erfüllen.

TS 2.a Simulationsszenarien: Variation der thermischen Umgebungsbedingungen bei Verbindungsabbrüchen- und Serviceausfällen

F-Req-3 **Fehlerbehandlung und Wiederherstellung:** Das System muss Fehlerbehandlungsmechanismen umsetzen, um mit Serviceausfällen, Netzverbindungsausfällen, Datenfehlern (inkonsistente Daten)

und Verzögerungen bei der Datenbereitstellung umgehen zu können. Darüber hinaus muss das System in der Lage sein, den Betrieb von Services nach einem Fehler wiederherzustellen.

TS 3.a Serviceausfall und Wiederherstellung: Stopp und Neustart des HLK-Cloud-Features

TS 3.b Verbindungsausfall: Trennung der Cloud-Verbindung

TS 3.c Datenfehler: Erzeugung von inkonsistenten Reglerstellgrößen

TS 3.d Datenverzögerungen: Einfügen synthetischer Verzögerungen der Datenübertragung

F-Req-4 Einhaltung des Regler Abtastintervalls: Der cloudbasierte MPC-Regler muss Stellgrößen in einem Zeitfenster kleiner als das Abtastintervall berechnen. Es muss also folgende Gleichung erfüllt werden:

$$\Delta t_{\mathrm{MPC}} > t_{\mathrm{Exe,Cloud}} + t_{\mathrm{Trans,Cloud}} + t_{\mathrm{Latenz}} \tag{6.1}$$

Es wird davon ausgegangen, dass der PID-Regler im *Fallback* Deploymentmodell aufgrund geringer Rechenanforderungen diese Zeitanforderung jederzeit erfüllt.

TS 4.a Überprüfung der Zeitanforderung: Variation des Prädiktions- bzw. Kontrollhorizonts des cloudbasierten MPC ausgehend von einem Basisfall.

Nichtfunktionale Anforderungen

NF-Req-1 Wartbarkeit und Erweiterbarkeit: Das System soll modular aufgebaut sein, um Services problemlos zu aktualisieren. Die Architektur soll so ausgelegt sein, dass zukünftige Erweiterungen

oder Anpassungen ohne Umstrukturierungsaufwand erfolgen können.

- keine Testszenarien

NF-Req-2 **Security:** Die Übertragung von Daten zwischen Cloud und Fahrzeug/Simulation soll verschlüsselt erfolgen. Zusätzlich darf eine Verbindung mit der Cloud nur nach erfolgreicher Authentifizierung stattfinden.

- keine Testszenarien

6.4 Validierung der Anforderungen

6.4.1 Die Simulationsszenarien

Mit Hilfe der *Dynamic_Weather*-Komponente der Simulationsumgebung (s. Kapitel 6.2) wurden zunächst drei repräsentative Klimaszenarien erstellt, die typische Wetterverläufe abbilden. Grundlage dafür bilden Temperaturdaten des Deutschen Wetterdienstes aus den Jahren 1960 bis 2020 für die Städte Berlin, Hamburg, Köln, Mannheim und München. Die Auswertung dieser Daten zeigt eine annähernd normalverteilte Temperatur mit einem Mittelwert von $\mu_{\text{Temperatur}} = 9{,}805\,^{\circ}\text{C}$ und einer Standardabweichung von $\sigma_{\text{Temperatur}} = 8{,}007\,^{\circ}\text{C}$. Auf Basis dieser Verteilung wurden, unter Verwendung realer Wetterdaten der Stadt Köln, die in der Mitte der betrachteten Städte liegt, drei Szenarien definiert.

Zur umfassenden Bewertung der Systemperformanz wurde zusätzlich ein weiteres Szenario bei der unteren Grenztemperatur des COP-Modells der Wärmepumpe berücksichtigt (vgl. Kapitel 5.3.2). Diese Simulation (Sim 4) wurde bei einer nominalen Außentemperatur von $-15\,^{\circ}\text{C}$ durchgeführt (nicht auf Basis von realen Wetterdaten der Stadt Köln), um das Verhalten des Systems am Rand des definierten Betriebsbereichs zu charakterisieren

und die Robustheit der beiden Deploymentmodelle unter extremen Witterungsbedingungen zu evaluieren.

Insgesamt entstanden die in den folgenden Klimaszenarien beschriebenen Simulationen:

Sim 1 Szenario mit durchschnittlichen nominalen Außentemperaturen,

Sim 2 Szenario mit Temperaturen eine Standardabweichung unterhalb des Mittelwerts,

Sim 3 Szenario mit Temperaturen eine Standardabweichung oberhalb des Mittelwerts,

Sim 4 Szenario bei der unteren Grenztemperatur des COP-Modells der Wärmepumpe.

Da alle Klimaszenarien realistische Tagestemperaturverläufe mit natürlichen Schwankungen simulieren, weichen die resultierenden Durchschnittstemperaturen geringfügig von den angestrebten Zielwerten ab. Die *Bus_Agent* Komponente diente zur Konfiguration der Haltedauern an Bushaltestellen und zur Parametrierung der Passagieranzahl, die sich an Haltestellen ändert. Alle Parameter zusammen ergeben die Simulationsszenarien in Tabelle 6.2.

Zur Validierung der Einhaltung des Abtastintervalls (vgl. **F-Req-4**) sowie zur Bewertung des Einflusses des Prädiktionshorizonts (s. Kapitel 2.1.1 auf die Energieeffizienz (vgl. **F-Req-2**) wurde Sim 1 in drei Unterszenarien unterteilt, die sich durch unterschiedliche Prädiktionshorizonte des MPC unterscheiden.

Abtastung von Simulation und Regelung In der Simulationsumgebung *Carla* wird eine Abtastrate von $\Delta t_{\mathrm{Sim}} = 0{,}33\,\mathrm{s}$ verwendet. Diese hohe zeitliche Auflösung ist ausreichend, um die Dynamik sämtlicher thermischer Massen im Fahrzeug präzise abzubilden. Der übergeordnete MPC wird mit einem Abtastintervall von $\Delta t_{\mathrm{MPC}} = 10\,\mathrm{s}$ implementiert. Dieses Intervall

wurde auf Grundlage der Regelstreckendynamik gewählt, die maßgeblich durch die thermische Trägheit der Fahrzeugkabine bestimmt wird. Aus den in Tabelle 5.4 aufgeführten Modellparametern ergibt sich eine Zeitkonstante der Fahrzeugkabine von $\tau_{\text{Kabine}} = 193\,\text{s}$, was gut mit experimentellen Messungen aus [Lin21] übereinstimmt.

Das gewählte Regler-Abtastintervall ist damit klein genug, um die Dynamik der Kabine hinreichend genau zu erfassen. Gleichzeitig erlaubt es eine mindestens zweifache Abtastung der Wärmepumpe innerhalb ihrer typischen Zeitkonstante, wie in Abbildung A.19 dargestellt. Dadurch wird eine angemessene Regelgüte sowohl für das Gesamtsystem als auch für die untergeordneten thermischen Komponenten sichergestellt.

Tabelle 6.2: Rahmenbedingungen der Simulationen

	Sim 1 (a,b,c)‡	**Sim 2**	**Sim 3**	**Sim 4**
$\overline{T}_{\text{Umgebung}}$ in $^\circ C$	9,58	0,94	16,44	-14,87
$\overline{t}_{\text{Busstopp}}$ in s	16,5	33	8,25	22,0
$\overline{\dot{Q}}_{\text{Solar}}$ in W	290	0	330	60
$\overline{\varphi}_{\text{Umgebung}}$ in %	34	51	33	80
$N_{\text{Passagiere}}$†	1–88	1–88	1–88	1–88

‡ Variation des Prädiktionshorizonts des MPC: a: $N_{\text{P}} = 10$, b: $N_{\text{P}} = 20$, c: $N_{\text{P}} = 30$

† Die Anzahl der Passagiere variiert bei jeder Bushaltestelle im Bereich von 1 bis 88.

6.4.2 F-Req-1: Energieeffizienz

Durchführung des Testszenarios Die funktionale Anforderung **F-Req-1** wird auf Basis der in Tabelle 6.2 dargestellten Simulationsszenarien validiert. Die Variation der Umgebungstemperatur in diesen Szenarien deckt das Testszenario **TS 1.a** ab. Tabelle 6.3 zeigt den direkten Vergleich zwischen einem PID-Regler, ausgeführt auf der ATLAS-ECU, und dem MPC im *Nur Cloud*-Deploymentmodell (vgl. Kapitel 3.2) auf dem ATLAS Cloud PC (vgl.

Tabelle 6.1). Die Verbindung zur Cloud wurde in diesen Szenarien nicht unterbrochen oder ein Serviceausfall generiert (vgl. Kapitel 6.4.4).

Tabelle 6.3: Vergleich der fahrzeuginternen PID- und der cloudbasierten MPC-Regelung im Nur Cloud-Deploymentmodell ohne Service- oder Verbindungsausfälle

Simulation (Szenariendauer 620 Sekunden)	**PID (fahrzeugintern)**	**MPC (Nur Cloud)**
Sim 1a $\overline{T}_{\text{Umgebung}} = 9{,}58\,^{\circ}\text{C}$	$\overline{\Delta T} = 0{,}75\,\text{K}$ $E_{\text{WP,elektrisch}} = 1.392\,\text{kW s}$ $\bar{t}_{\text{Exe,ECU}} = 0{,}03\,\text{ms}$	$\mathbf{N_P = N_C = 10}$ $\overline{\Delta T} = 0{,}65\,\text{K}$ $E_{\text{WP,elektrisch}} = 1.238\,\text{kW s}$ $\bar{t}_{\text{Exe,Cloud}} = 797{,}94\,\text{ms}$
Sim 1b $\overline{T}_{\text{Umgebung}} = 9{,}58\,^{\circ}\text{C}$	identisch wie Zeile 1	$\mathbf{N_P = N_C = 20}$ $\overline{\Delta T} = 1{,}13\,\text{K}$ $E_{\text{WP,elektrisch}} = 1.080\,\text{kW s}$ $\bar{t}_{\text{Exe,Cloud}} = 1.870\,\text{ms}$
Sim 1c $\overline{T}_{\text{Umgebung}} = 9{,}58\,^{\circ}\text{C}$	identisch wie Zeile 1	$\mathbf{N_P = N_C = 30}$ $\overline{\Delta T} = 0{,}98\,\text{K}$ $E_{\text{WP,elektrisch}} = 1.062\,\text{kW s}$ $\bar{t}_{\text{Exe,Cloud}} = 2.023{,}77\,\text{ms}$
Sim 2: $\overline{T}_{\text{Umgebung}} = 0{,}94\,^{\circ}\text{C}$	$\overline{\Delta T} = 1{,}34\,\text{K}$ $E_{\text{WP,elektrisch}} = 2.834\,\text{kW s}$ $\bar{t}_{\text{Exe,ECU}} = 0{,}03\,\text{ms}$	$\mathbf{N_P = N_C = 10}$ $\overline{\Delta T} = 1{,}47\,\text{K}$ $E_{\text{WP,elektrisch}} = 2.672\,\text{kW s}$ $\bar{t}_{\text{Exe,Cloud}} = 600{,}26\,\text{ms}$
Sim 3: $\overline{T}_{\text{Umgebung}} = 16{,}44\,^{\circ}\text{C}$	$\overline{\Delta T} = 0{,}59\,\text{K}$ $E_{\text{WP,elektrisch}} = 1.046\,\text{kW s}$ $\bar{t}_{\text{Exe,ECU}} = 0{,}03\,\text{ms}$	$\mathbf{N_P = N_C = 10}$ $\overline{\Delta T} = 0{,}54\,\text{K}$ $E_{\text{WP,elektrisch}} = 1.130\,\text{kW s}$ $\bar{t}_{\text{Exe,Cloud}} = 858{,}14\,\text{ms}$
Sim 4: $\overline{T}_{\text{Umgebung}} = -14{,}87\,^{\circ}\text{C}$	$\overline{\Delta T} = 2{,}05\,\text{K}$ $E_{\text{WP,elektrisch}} = 5.740\,\text{kW s}$ $\bar{t}_{\text{Exe,ECU}} = 0{,}03\,\text{ms}$	$\mathbf{N_P = N_C = 10}$ $\overline{\Delta T} = 1{,}93\,\text{K}$ $E_{\text{WP,elektrisch}} = 5.580\,\text{kW s}$ $\bar{t}_{\text{Exe,Cloud}} = 680{,}25\,\text{ms}$

Hinsichtlich des elektrischen Energieverbrauchs der Wärmepumpe des HLK-Systems demonstriert der MPC deutliche Vorteile. In den Simulationen 1a bis 1c kann der Energieverbrauch des MPC durch eine Erhöhung des Prädiktions- und Kontrollhorizonts von 10 auf maximal 30 erheblich reduziert werden. Den größten Effizienzvorteil zeigt Simulation 1c mit einer Einsparung von $330\,\text{kW s}$ (von $1.392\,\text{kW s}$ auf $1.062\,\text{kW s}$), was einer Reduzierung um bis zu $23{,}7\,\%$ entspricht. Auch in Simulation 2 und Simulation 4 bleibt der MPC energieeffizienter. Lediglich in Simulation 3 bei wärmeren Umgebungsbedingungen zeigt der MPC einen leicht höheren Verbrauch ($1.130\,\text{kW s}$

gegenüber $1.046\,\mathrm{kW\,s}$), kompensiert dies jedoch durch eine verbesserte Regelgüte.

Die Ergebnisse zeigen, dass Anforderung **F-Req-1** als erfüllt gekennzeichnet werden kann.

Anmerkung: Betrachtet man die Energieeffizienz des COTA-Ansatzes unter Berücksichtigung von Verbindungs- und Serviceausfällen in beiden Deploymentmodellen (vgl. Tabelle 6.4) im Vergleich zum fahrzeuginternen PID-Regler (vgl. Tabelle 6.3), zeigt sich, dass der PID-Regler in allen Szenarien energieeffizienter ist, mit Einsparungen zwischen $1{,}9\,\%$ und $8{,}6\,\%$ gegenüber beiden cloudbasierten Deploymentmodellen des MPC-Reglers. Dabei ist zu beachten, dass Service- bzw. Verbindungsausfälle rund ein Drittel der Gesamtzeit der Simulationsszenarien ausmachen. Unter den beiden Deploymentmodellen weist das *Fallback*-Modell in allen Szenarien einen geringeren Energieverbrauch auf als das *Nur Cloud*-Modell.

6.4.3 F-Req-2: Thermischer Komfort

Durchführung des Testszenarios Für das Testszenario **TS 2.a** werden erneut die vier Simulationsszenarien aus Tabelle 6.2 herangezogen, innerhalb derer definierte Verbindungs- und Serviceausfälle erzeugt werden (s. nachfolgende Validierung der Systemanforderung **F-Req-3**). Jedes Simulationsszenario wurde mit beiden Deploymentmodellen durchgeführt, wobei die Ausfälle immer identisch implementiert wurden, sodass ein direkter Vergleich möglich ist.
Der thermische Komfort wird in drei der vier Simulationsszenarien in beiden Deploymentmodellen entsprechend den VDV-Vorgaben (s. Kapitel 2.4.3) eingehalten (vgl. Tabelle 6.4). Im Kälteszenario von Sim 4 ($\overline{T}_{\mathrm{Umgebung}} = -14{,}87\,^{\circ}\mathrm{C}$) wird die Vorgabe jedoch in beiden Modellen leicht überschritten, da das mittlere Temperaturdelta über $2\,\mathrm{K}$ liegt.

Insgesamt kann **F-Req-2** nicht als vollständig erfüllt gekennzeichnet werden, da bei extremen Wetterbedingungen die VDV-236 Vorgaben um $0{,}12\,\mathrm{K}$ bzw. $0{,}25\,\mathrm{K}$ überschritten werden.

Tabelle 6.4: Vergleich des *Fallback*- und *Nur Cloud*-Deploymentmodells in den Simulationsszenarien mit Verbindungs- bzw. Serviceausfällen

Simulation (Szenariendauer 620 Sekunden)	**Fallback**	**Nur Cloud**
Sim 1a $\overline{T}_{\mathrm{Umgebung}} = 9{,}58\,^{\circ}\mathrm{C}$	$\overline{\Delta T} = 0{,}76\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 1.509\,\mathrm{kW\,s}$	$\overline{\Delta T} = 1{,}24\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 1.760\,\mathrm{kW\,s}$
Sim 2: $\overline{T}_{\mathrm{Umgebung}} = 0{,}94\,^{\circ}\mathrm{C}$	$\overline{\Delta T} = 1{,}44\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 2.978\,\mathrm{kW\,s}$	$\overline{\Delta T} = 1{,}5\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 3.050\,\mathrm{kW\,s}$
Sim 3: $\overline{T}_{\mathrm{Umgebung}} = 16{,}44\,^{\circ}\mathrm{C}$	$\overline{\Delta T} = 0{,}61\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 1.140\,\mathrm{kW\,s}$	$\overline{\Delta T} = 0{,}64\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 1.180\,\mathrm{kW\,s}$
Sim 4: $\overline{T}_{\mathrm{Umgebung}} = -14{,}87\,^{\circ}\mathrm{C}$	$\overline{\Delta T} = 2{,}12\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 5.850\,\mathrm{kW\,s}$	$\overline{\Delta T} = 2{,}25\,\mathrm{K}$ $E_{\mathrm{WP,elektrisch}} = 6.100\,\mathrm{kW\,s}$

6.4.4 F-Req-3: Fehlerbehandlung und Wiederherstellung

Durchführung der Testszenarien zu F-Req-3 Der Einfluss von Service- und Verbindungsausfällen (s. **TS 3.a** und **TS 3.b**) auf die cloudbasierte Regelung im *Nur Cloud* Deploymentmodell wird in Abbildung 6.10 in Simulationsszenario 1a (vgl. Tabelle 6.2) dargestellt. Nach der *Startphase* befindet sich das System in der Aufheizphase bis die Solltemperatur nach $160\,\mathrm{s}$ erreicht wird. 50 Sekunden später ist der HLK-Cloud-Service nicht mehr abrufbar und das Regelsystem wechselt automatisch auf den Cloud-BPH-Modus und wieder zurück (s. Kapitel 6.2), sobald der Cloud-Service wieder verfügbar ist. Bei Sekunde 440 geht die Verbindung zur Cloud verloren (s. Cloud-Verbindung in Abbildung 6.10). Der *Connection-Manager* versucht anschließend über einen Zeitraum von 20 Sekunden in regelmäßigen Intervallen die Verbindung erneut herzustellen, bevor die Cloud-Services aus der Datenbank gelöscht werden und auf den BPH gewechselt wird (s. oberste Darstellung bei Sekunde 460 in Abbildung 6.10). In dieser Übergangsphase bleibt die Stellgröße der Wärmepumpe konstant. Anschließend wird auf den zeitlich angepassten

BPH zurückgegriffen. Der Wechsel zurück in den regelnden Betrieb, sobald die Cloud-Verbindung zum Ende des Szenarios wieder vorhanden ist, funktioniert reibungslos.
Die angewendeten Stellgrößen der Wärmepumpe sind im BPH-Modus aufgrund des fehlenden Feedbacks mit größeren Schwankungen behaftet.

Im *Fallback*-Deploymentmodell besteht für das System die Möglichkeit, bei einem Verbindungsabbruch auf einen fahrzeuginternen PID-Regler zu wechseln. Hierbei kommt der *Arbiter* (s. Kapitel 5.5) des Orchestrators, der den Wechselvorgang in die Wege leitet, zum Einsatz. Abbildung 6.11 (ebenfalls in Simulation 1a) zeigt diesen Fall, bei dem der *Arbiter* aufgrund hoher gemessener RTT der MQTT Nachrichten vom Fahrzeug zur Cloud und zurück vom MPC- zum PID-Regler im Fahrzeug wechselt. Der *Arbiter* initiiert zweimal den Wechsel auf den fahrzeuginternen PID-Regler, was sich unmittelbar in den Stellgrößen der Wärmepumpe widerspiegelt. Diese schwanken beim Einsatz des PID-Reglers wesentlich stärker, da diese Schwankungen nicht wie beim MPC durch eine Kostenfunktion bestraft werden können. Die Kabinentemperatur oszilliert um den Sollwert stärker, als dies unter stabiler Verbindung der Fall ist (vgl. Abbildung 6.9). Der Übergang zwischen lokalem PID und cloudbasiertem MPC ist so realisiert, dass die aktuelle Stellgröße so lange auf die Regelstrecke angewendet wird, bis eine neue Stellgröße zur Verfügung steht.

Die Plausibilitätsprüfung wurde durch das Setzen von inkonsistenten Stellgrößen (s. **TS 3.c**) zur Laufzeit und durch das Hinzufügen von Verzögerungen bei der Berechnung der Stellgrößen (s. **TS 3.d**) getestet (s. Abbildung 6.12). Die zu Beginn des Szenarios auffällig hohen Berechnungszeiten des cloudbasierten Reglers werden korrekt erkannt, und stattdessen wird die zuletzt plausible Stellgröße angewendet. Dasselbe gilt für berechnete Stellgrößen, die entweder aufgrund zu starker Änderungsrate oder aufgrund eines zu hohen Wertes physikalisch nicht von der Wärmepumpe umgesetzt werden können.
F-Req-3 kann als erfüllt gekennzeichnet werden.

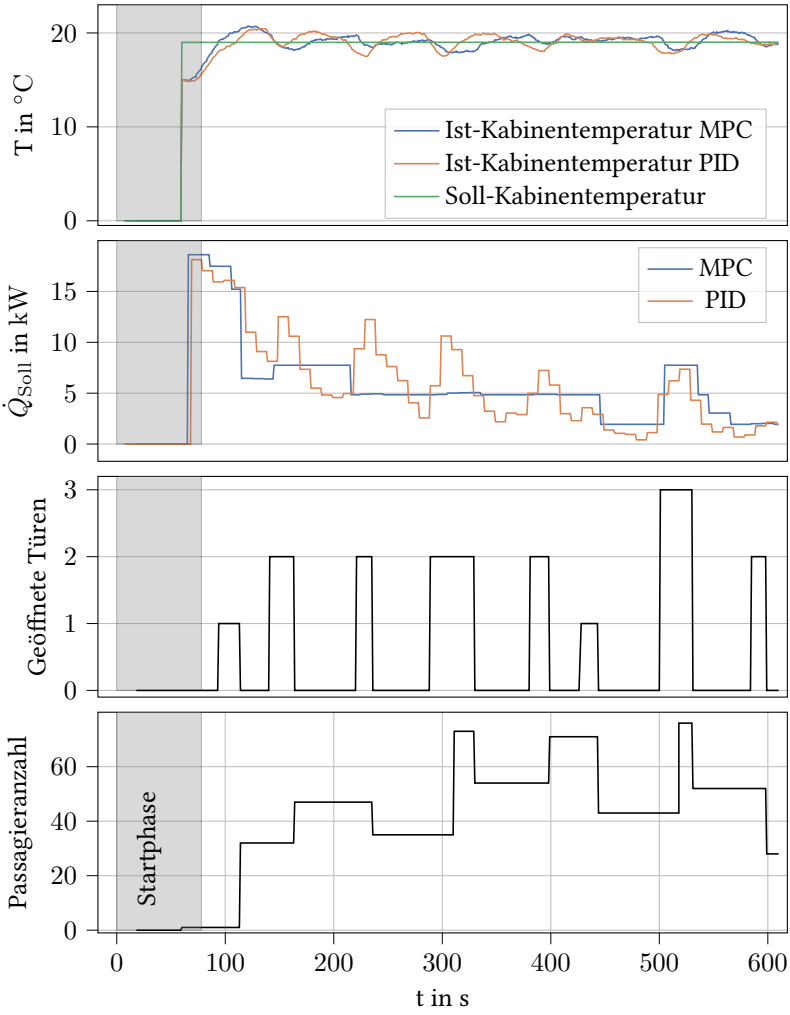


Abbildung 6.9: Vergleich der fahrzeuginternen PID- und cloudbasierten MPC-Regelung (Nur Cloud) in Simulation 1a

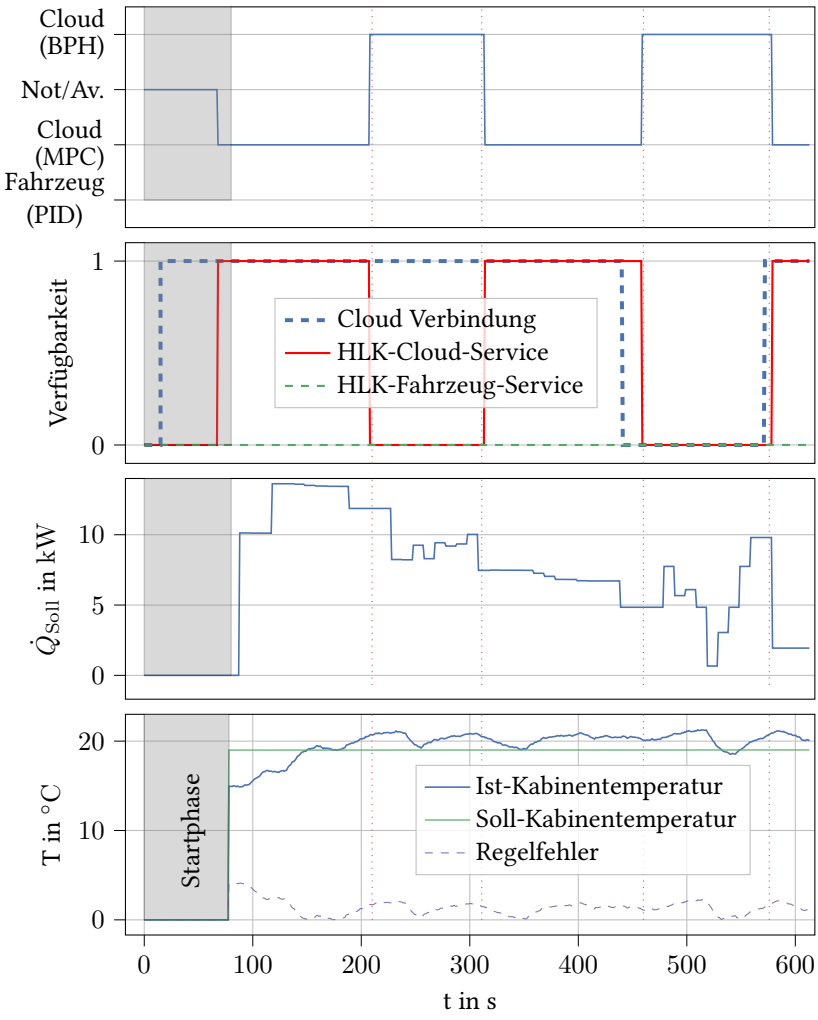


Abbildung 6.10: Umgang des cloudbasierten Regelungssystems mit Service- und Verbindungsausfällen im *Nur Cloud* Deploymentmodell in Sim 1a

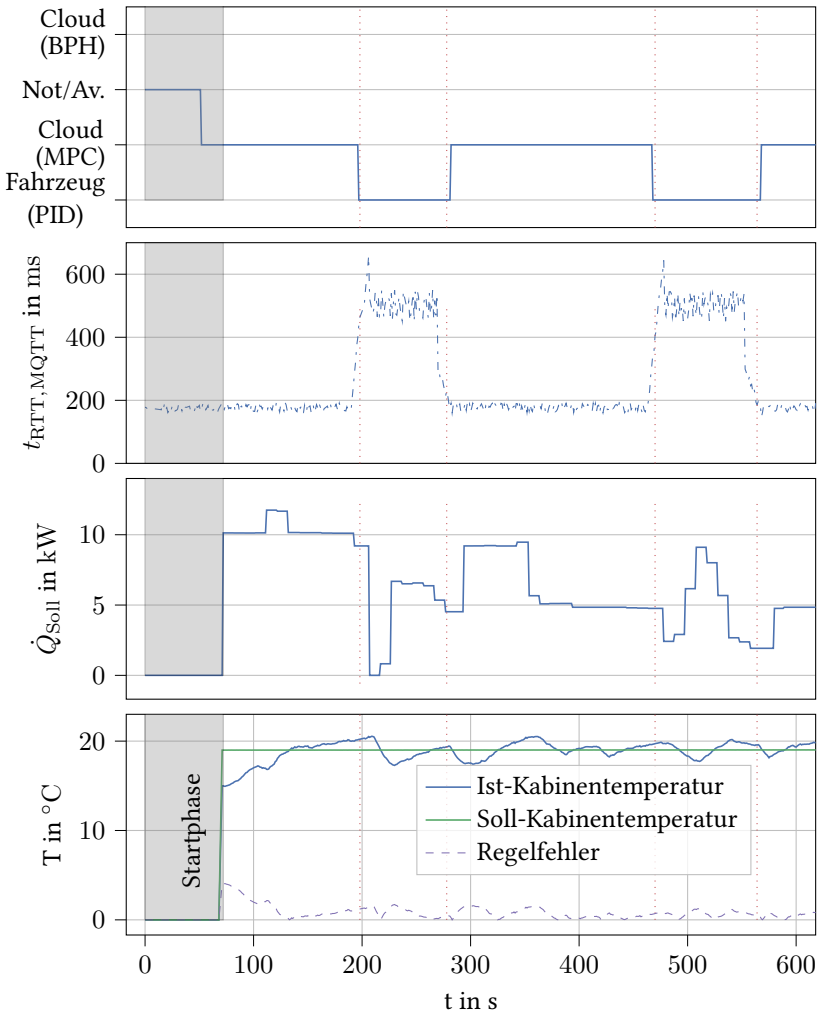


Abbildung 6.11: Umgang des cloudbasierten Regelungssystems mit Verbindungsausfällen im *Fallback*-Deploymentmodell in Simulation 1a

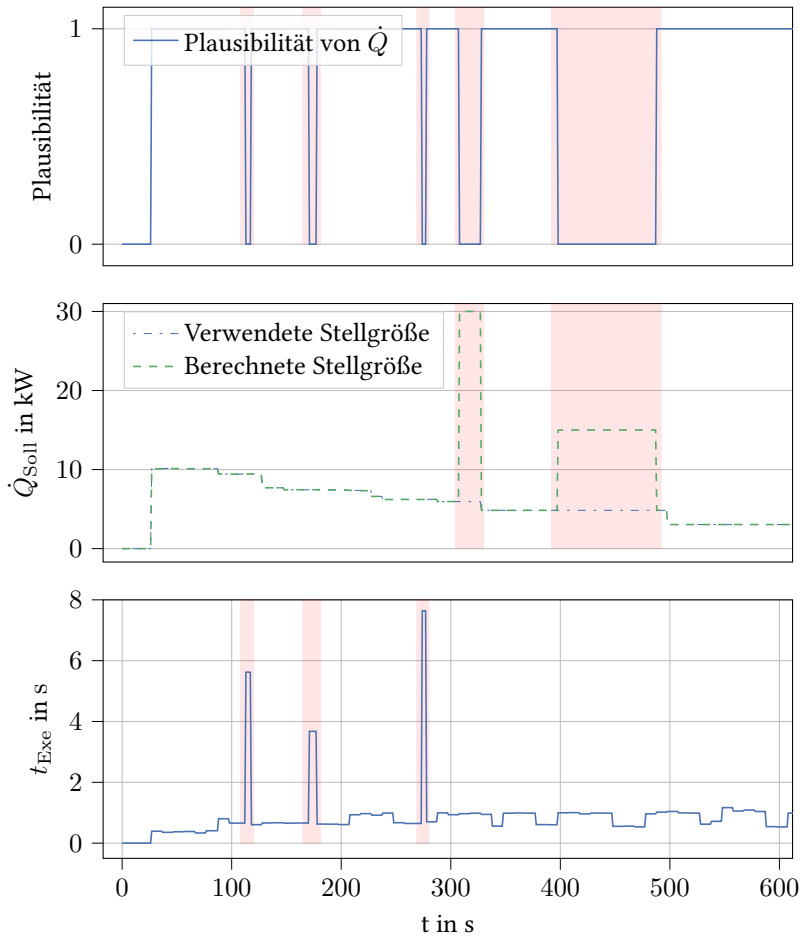


Abbildung 6.12: Plausibilitätsprüfung der cloudbasierten HLK-Regelung

6.4.5 F-Req-4: Einhaltung des Regler Abtastintervalls

Die Zeitanforderung aus Gleichung 6.1 wird durch den *Connection-Manager* des Orchestrators überwacht. Zur Bestimmung von $t_{\text{Exe,Cloud}}$ wird die Zeit erfasst, die der MPC benötigt, um die nächste Stellgrößenfolge zu berechnen. Das Abtastintervall Δt_{MPC} ist auf $10\,\text{s}$ festgelegt (vgl. Kapitel 6.4.1). Die Datenrate für das Mobilfunknetz wird entsprechend den Durchschnittswerten aus dem deutschen Mobilfunknetz in Kapitel 2.6.2 im System festgesetzt.

Durchführung des Testszenarios zu F-Req-4 Gleichung 6.1 wird ausgehend von einem Basisfall des MPC mit $N_{\text{P}} = N_{\text{C}} = 20$ (s. **TS 4.a**) und dem lokalen Optimierungsalgorithmus *Powell*[2] überprüft. Dabei wurde die RTT auf Basis des implementierten MQTT Protokolls gemessen und diese nicht weiter in die einzelnen Bestandteile Latenzzeit und Transferzeit, wie in der genannten Gleichung genannt, aufgeteilt. Die gemessene durchschnittliche RTT von $173\,\text{ms}$ ist dabei vergleichbar mit Messwerten aus der aktuellen Veröffentlichung von Dettinger et al. [DWW+25]. Die Ergebnisse aus Tabelle 6.5 zeigen, dass die Zeitanforderung mit einem Puffer von $7{,}96\,\text{s}$ eingehalten werden kann. Dieser Puffer ermöglicht es, N_{P} und N_{C} des MPC bei einer Cloud-Ausführung auf maximal 60 zu erhöhen oder auf einen globalen Optimierungsalgorithmus zu wechseln. Bei einer Ausführung des MPC auf der ATLAS ECU (s. Tabelle 6.1) ist $N_{\text{P}} = N_{\text{C}}$ aufgrund der hohen Berechnungszeit auf 20 beschränkt.

F-Req-4 kann als erfüllt gekennzeichnet werden.

[2] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-powell.html

Tabelle 6.5: Messwerte der Ausführung des MPC ($N_\mathrm{P} = N_\mathrm{C} = 20$) auf der ATLAS ECU vs. dem ATLAS Cloud PC

Parameter	Wert
$\bar{t}_{\mathrm{Exe,Cloud}}$	1,87 s
$\bar{t}_{\mathrm{Exe,ECU}}$	6,3 s
$\bar{t}_{\mathrm{RTT,MQTT}}$	173,00 ms
$L_{\mathrm{Exe,ECU}}$	26,13 %
$L_{\mathrm{Idle,ECU}}$	0,5 %
L_{Transfer}	11,17 %
$P_{\mathrm{Exe,ECU}}$	1,96 W
$P_{\mathrm{Idle,ECU}}$	0,04 W
P_{Transfer}	0,84 W

6.4.6 NF-Req-1: Wartbarkeit und Erweiterbarkeit

Bedingt durch die SOA-Architektur und die Möglichkeit von OTA-Updates durch den *Feature-Updater* (s. Kapitel 5.5), ist der Aufbau des Systems modular gestaltet und Services sowie Features können leicht aktualisiert werden. Zusätzlich erfolgt die Konfiguration der Regelparameter und der Simulation über YAML Ain't Markup Language (YAML)-Dateien. Der Pfad der YAML-Dateien kann beim Starten der zugehörigen Nodes angegeben werden. Die Parametrisierung der Regelstrecke (Fahrzeugkabine und Wärmepumpe) kann ebenfalls durch die jeweilige Konfigurationsdatei angepasst werden.

NF-Req-1 kann als erfüllt gekennzeichnet werden.

6.4.7 NF-Req-2: Security

Die Authentifizierung zwischen Fahrzeug (Client) und Cloud (Broker) wird durch die Mechanismen des MQTT Protokolls realisiert. Dazu authentifiziert sich der Client beim Broker mittels Client-ID und einem Token. Der Client erhält dazu einen Token, welchen er symmetrisch verschlüsselt an den Broker sendet. Der Broker validiert diesen Token und überprüft infolgedessen seine Berechtigungen. Eine zyklische Erneuerung der Token ist nicht realisiert. Für die Anwendung in der Produktion müsste neben der sicheren Speicherung der Token zusätzlich eine asymmetrische Verschlüsselung der Tokens zwischen Client und Broker erfolgen. Die Nachrichten werden mit dem Transport Layer Security (TLS) Protokoll unter der Verwendung von Zertifikaten übermittelt.

NF-Req-2 kann als erfüllt gekennzeichnet werden.

6.5 Bewertung der Ergebnisse

Die Ergebnisse des cloudbasierten HLK-Reglers belegen die Eignung des COTA-Ansatzes, sowohl in wirtschaftlicher Hinsicht als auch im Hinblick auf die Performanz des Reglers.

- **Performanz:** Energetische Einsparungen (vgl. **Req-2 (Regler)**) im Fahrzeug lassen sich sowohl auf der in Kapitel 4.2.3 beschriebenen Steuergeräteebene als auch durch eine effizientere Auslegung und Regelung von Fahrzeugfunktionen realisieren.

  Die Einsparungen auf Steuergeräteebene können mithilfe der Gleichung 3.2 quantifiziert werden. Grundlage dafür bilden die Messwerte aus Tabelle 6.5, auf deren Basis eine Energieeinsparung von $12{,}13\,\mathrm{W\,s}$ pro Regelungsschritt ermittelt wurde. Unter Annahme einer durchschnittlichen täglichen Betriebszeit von $9{,}35\,\mathrm{h}$ gemäß dem öffentlichen

Datensatz ZTBus [127] und einem Abtastintervall von $\Delta t_{\mathrm{MPC}} = 10\,\mathrm{s}$ ergibt sich eine kumulierte Einsparung von $11{,}62\,\mathrm{W\,h}$ pro Tag. Bezogen auf die Batteriekapazität eines Stadtbusses von bis zu $588\,\mathrm{kW\,h}$ (vgl. Kapitel 2.4.1) ist dieser Wert als vernachlässigbar einzustufen – er entspricht einer zusätzlichen Reichweite von lediglich rund $10\,\mathrm{m}$.

Deutlich größere Einsparpotenziale ergeben sich hingegen durch den Einsatz effizienterer Regelungsstrategien. Die Ergebnisse des Vergleichs zwischen dem fahrzeuginternen PID-Regler und dem cloudbasierten MPC im Szenario *Nur Cloud* ohne Verbindungsabbrüche (Simulation 1a, $N_{\mathrm{P}} = 10$) zeigen eine relative Energieeinsparung von $11{,}06\,\%$ (vgl. Tabelle 6.3). Hochgerechnet auf die angenommene tägliche Betriebszeit entspricht dies einer absoluten Einsparung von $2{,}4\,\mathrm{kW\,h}$. Bei einem spezifischen Energieverbrauch von $1{,}15\,\mathrm{kW\,h\,km^{-1}}$ (extrapoliert aus [15]) bei einer durchschnittlichen Umgebungstemperatur von $9{,}58\,^{\circ}\mathrm{C}$ (entspricht der Außentemperatur in Simulation 1a) resultiert daraus eine rechnerische Reichweitenerhöhung von $2{,}1\,\mathrm{km}$. Treten über längere Zeiträume Verbindungsabbrüche zur Cloud auf oder ist der Cloud-Service nicht verfügbar, verringern sich die Einsparpotenziale erheblich, bis hin zu dem Punkt, an dem der fahrzeuginterne PID-Regler dem cloudbasierten Ansatz überlegen ist. Dies zeigt sich deutlich im Vergleich der Tabellen 6.3 und 6.4: In diesem Szenario erweist sich der lokale PID-Regler als effizienter als beide cloudbasierten Deployment-Modelle des MPC, da die Verbindungs- bzw. Serviceausfälle nahezu ein Drittel der Gesamtzeit ausmachen.

Der thermische Komfort (vgl. **Req-2 (Regler)**) wird in drei von vier Simulationsszenarien auch bei Verbindungsabbrüchen zur Cloud durch beide Deploymentmodelle zuverlässig gewährleistet und entspricht den Anforderungen der VDV-236. Einzig bei extremen Außentemperaturen von $\overline{T}_{\mathrm{Umgebung}} = -14{,}87\,^{\circ}\mathrm{C}$ werden die Vorgaben leicht

überschritten. Das *Fallback*-Deploymentmodell zeigt dabei eine höhere Regelgüte und Energieeffizienz im Vergleich zum *Nur Cloud*-Modell.

Zudem zeigt sich, dass der Cloud-PC (vgl. Tabelle 6.1) die Stellgrößen des MPC signifikant schneller berechnet (vgl. **Req-4 (Regler)**) als ein lokales Steuergerät, in diesem Fall ein Raspberry Pi 4 B. Der Vergleich in Tabelle 6.5 verdeutlicht eine 3,3-fach schnellere Berechnungszeit der Stellgrößen in der Cloud im Vergleich zur lokalen Ausführung, wobei sich der Faktor auf 3,08 reduziert, wenn man auf die Berechnungszeit in der Cloud auch die durchschnittliche RTT zur Cloud und zurück addiert. Dies unterstreicht die Sinnhaftigkeit der Cloud-Nutzung, da dadurch zusätzlicher Spielraum sowohl beim Prädiktionshorizont des MPC als auch bei der Abtastrate des MPC entsteht, der im Fahrzeug aufgrund der dort begrenzt verfügbaren Rechenleistung nicht realisierbar wäre.

- **Wirtschaftlichkeit:** Das *Fallback*-Modell ist dem *Nur Cloud* Modell hinsichtlich Komfort und Energieeffizienz überlegen. Dennoch stellt sich aus wirtschaftlicher Sicht die Frage, ob sich der Mehraufwand dieser Lösung auch langfristig rechtfertigen lässt.

  Aus betriebswirtschaftlicher Perspektive eröffnet sich somit die Notwendigkeit, beide Ansätze im Rahmen einer TCO-Betrachtung gegenüberzustellen. Das *Fallback*-Modell verursacht zusätzliche Kosten durch das Deployment einer Regelung im Fahrzeug, erhöhten Entwicklungsaufwand für die parallele Ausführung von Funktionen in Fahrzeug und Cloud sowie durch potenziell höheren Wartungs- und Validierungsaufwand über den Lebenszyklus hinweg. Das *Nur Cloud*-Modell hingegen erfordert eine zuverlässige Netzwerkinfrastruktur sowie eine hochverfügbare Cloud-Umgebung, was zu laufenden Betriebs- und Kommunikationskosten führt, jedoch mit deutlich reduziertem Aufwand auf Fahrzeugseite.

Für eine detaillierte Bewertung dieser Aspekte wird auf das in [BtS+24] vorgestellte TCO-Modell verwiesen. Es bietet einen strukturierten Rahmen zur Analyse der wirtschaftlichen Auswirkungen verschiedener Softwareverteilungsszenarien im Fahrzeugkontext, einschließlich Entwicklung, Infrastruktur, Betrieb, Wartung und Skalierung.

7 Fazit und Ausblick

Cloudbasierte Fahrzeugfunktionen (s. Definition 4.1) verändern nicht nur die Fahrzeugarchitektur grundlegend, sondern stellen insbesondere den bisherigen E/E-Entwicklungsprozess infrage. Künftig muss bereits in frühen Entwicklungsphasen bewertet werden, ob eine Funktion im Fahrzeug oder in der Cloud ausgeführt werden soll. Vor diesem Hintergrund wird in der Dissertation der Fokus auf den Ansatz des Control-Over-The-Air (COTA) gesetzt, bei dem regelnde Funktionen aus dem Fahrzeug herausgelöst und in die Cloud verlagert werden.

7.1 Beantwortung der Forschungsfragen

Forschungsfrage 1: Identifikation cloudfähiger Funktionen

Wie können in Fahrzeug-E/E-Architekturen potenziell cloudfähige Funktionen systematisch identifiziert und deren Eignung für eine Cloud-Migration bewertet werden?

Die Identifikation cloudfähiger Funktionen kann mithilfe des in Kapitel 4.2 vorgestellten zweistufigen Prozesses durchgeführt werden. Die erste Stufe des Prozesses prüft, ob die Funktion in der Cloud realisierbar ist, während in der zweiten Stufe ein Eignungs-Score zur Anwendung kommt. Der Eignungs-Score beinhaltet insgesamt fünf Bewertungsmetriken (s. Kapitel 4.2.3), deren Ergebnisse im finalen Eignungs-Score einzeln gewichtet werden können. Anhand des Ergebnisses des Scores zwischen 0 (nicht für die Cloud geeignet)

und 1 (optimal geeignet) kann eine Entscheidung getroffen werden, ob eine Funktion im Fahrzeug verbleiben sollte oder ausgelagert werden kann.

Forschungsfrage 2: Deploymentmodelle

Welche Deploymentmodelle cloudbasierter Fahrzeugfunktionen sind vorstellbar und wie lässt sich das am besten geeignete finden?

Die Wahl des Deploymentmodells erfolgt, nachdem eine cloudfähige Funktion identifiziert wurde. Die Literaturrecherche ergab vier verschiedene Möglichkeiten für das Deployment und die Ausführung cloudbasierter Funktionen: *Nur Cloud*, *Fallback*, *Dupliziert* und *Elastisch*. Diese wurden um das Modell *Parallel* erweitert. Mit der in Kapitel 5.7 vorgestellten MCDA wird das passende Deploymentmodell abhängig von den Anforderungen der Funktion bestimmt. Dabei berücksichtigt die MCDA sowohl technische (Latenz, Verfügbarkeit) als auch wirtschaftliche (CapEx und OpEx) Kriterien. Auf Basis der durchgeführten MCDA ging das Deploymentmodell *Nur Cloud* mit geringem Vorsprung vor dem *Fallback*-Modell als optimale Lösung für die cloudbasierte HLK-Regelung eines Stadtbusses mittels MPC hervor. Aufgrund des knappen Ergebnisses wurden beide Modelle im weiteren Verlauf implementiert und untersucht. Dabei zeigte das *Fallback*-Deploymentmodell aus technischer Sicht, insbesondere hinsichtlich Regelgüte und Energieeffizienz, die besseren Resultate.

Aus wirtschaftlicher Sicht sind beim *Fallback*-Modell jedoch zusätzliche Aufwände durch die Umsetzung eines zusätzlichen Reglers im Fahrzeug zu berücksichtigen. Die Entscheidung zwischen den beiden Varianten stellt somit einen Abwägungsprozess zwischen technischer Leistungsfähigkeit und wirtschaftlichem Aufwand dar. Welche Lösung letztlich umgesetzt wird, hängt von den strategischen Zielen, der Infrastruktur sowie den betriebswirtschaftlichen Prioritäten des OEM ab.

Die MCDA unterstützt diesen Entscheidungsprozess, indem sie technische, wirtschaftliche und betriebliche Kriterien transparent gegenüberstellt. Bei

knappen Ergebnissen, wie im vorliegenden Fall, ist es jedoch sinnvoll, die Entscheidung durch eine nachgelagerte Validierung abzusichern. Dies kann etwa in Form von prototypischen Implementierungen, Simulationsstudien oder Praxistests erfolgen, um die Eignung des gewählten Deploymentmodells unter realen Bedingungen zu bestätigen.

Forschungsfrage 3: Potenziale der Cloudverlagerung

Welche Vorteile ergeben sich aus der Cloudverlagerung und wie lassen sich diese messbar nachvollziehen?

Die Einbindung der Cloud erweitert das Fahrzeug um eine flexible und anpassbare Rechenplattform. Dadurch ergeben sich zahlreiche Vorteile, insbesondere im Hinblick auf die Reduzierung von Ressourcenbeschränkungen sowie eine vereinfachte Entwicklung und Wartbarkeit von Software (vgl. Abschnitt „Potenziale durch die Ausführung der Funktion in der Cloud“ in Kapitel 4.2.3). Besonders deutlich wurde das Potenzial durch die Verwendung eines intelligenten Reglers in der Cloud: Der cloudbasierte MPC-Regler ist rechenintensiv und profitiert daher stark von den verfügbaren Cloud-Ressourcen. Die optimierte cloudbasierte Regelung ermöglichte eine Reduktion des Energieverbrauchs der Wärmepumpe des HLK-Systems um bis zu $23{,}7\,\%$ mit einem maximalen Prädiktionshorizont von $N_\mathrm{P} = 30$ bzw. $11{,}06\,\%$ mit einem kleineren Prädiktionshorizont des MPC in moderaten klimatischen Bedingungen von ca $10\,^\circ\mathrm{C}$ ohne Service- oder Verbindungsabbrüche zur Cloud.

Gleichzeitig wurde deutlich, dass der Energieverbrauch einzelner Steuergeräte bzw. die Ausführung typischer Funktionen auf diesen Steuergeräten (vgl. Tabelle 2.1) im Vergleich zur Batteriekapazität moderner BEB vernachlässigbar ist. Eine Auslagerung dieser Funktionen zur Energieeinsparung auf dieser Ebene ist daher nicht sinnvoll. Hinzu kommt der Energieaufwand für die Übertragung von Daten zwischen Fahrzeug und Cloud, der ebenfalls berücksichtigt werden muss (vgl. Kapitel 2.6.2).

7.2 Ausblick

Der in Kapitel 4.2 beschriebene Prozess zur Identifikation cloudbasierter Funktionen schließt sicherheitskritische und Funktionen mit harten Echtzeitanforderungen für eine Cloudverlagerung aus. Dies könnte sich mit der Weiterentwicklung neuer drahtloser Netzwerktechnologien ändern. Wenn diese Technologien die Zuverlässigkeit und Latenzzeit so weit verbessern, dass ASIL und harte Echtzeitanforderungen erfüllt werden können, könnte die erste Stufe des Prozesses entfallen.

Das Feld auslagerbarer Funktionen könnte durch die Erweiterung potenzieller Ausführungsknoten erhöht werden. Edge-Nodes bieten eine zusätzliche, dezentrale Rechenebene näher am Fahrzeug und eröffnen neue Möglichkeiten für die Verlagerung von Features, in der Hauptsache solcher, die in der Cloud aufgrund von Latenz, Verfügbarkeit oder Sicherheitsanforderungen nicht effizient genug betrieben werden können. Somit könnte ein gestufter Auslagerungsansatz entstehen: Sicherheitskritische Funktionen verbleiben im Fahrzeug, Funktionen mit geringer Zeitkritikalität wandern in die Cloud und dazwischen bietet die Edge eine Zwischenebene, beispielsweise zur Sensordatenvorverarbeitung.

Ein weiterer vielversprechender Ansatz für zukünftige Arbeiten ist die Mehrfachnutzung cloudbasierter Funktionen für ganze Fahrzeugflotten. Anstatt jede Fahrzeuginstanz individuell mit dedizierten Funktionen zu versorgen, können zentrale, skalierbare Funktionsinstanzen in der Cloud bereitgestellt werden, die gleichzeitig mehrere Fahrzeuge bedienen (Stichwort Matching Theory).

Das laufende Promotionsvorhaben von Baumann [68] zur *cloudbasierten Rekonfiguration lernender Softwarekomponenten* zeigt, wie sich durch die kontinuierliche Analyse von Umgebungs- und Fahrzeugdaten Funktionen dynamisch an wechselnde Einsatzbedingungen anpassen lassen, etwa an den Fahrmodus, die Wetterlage oder das Nutzerverhalten (s. Abbildung 7.1). Insbesondere im Stadtbusbetrieb mit Flotten, wiederkehrenden Routen und

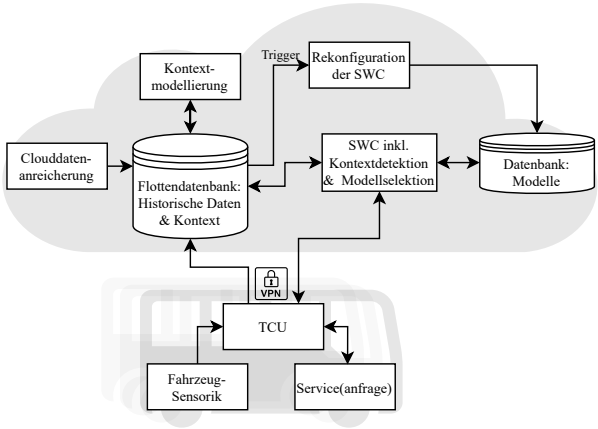


Abbildung 7.1: Konzept rekonfigurierbarer, lernender cloudbasierter Fahrzeugfunktionen (Darstellung aus dem laufenden Promotionsvorhaben von Daniel Baumann)

festen Fahrplänen können lernende Funktionen dynamisch an spezifische Einsatzbedingungen angepasst werden. Beispielsweise lassen sich Antriebsstrategien, Rekuperationsprofile oder Klimatisierungsparameter auf Basis gesammelter Fahr- und Umweltdaten für einzelne Linien oder Haltestellen dynamisch anpassen, wodurch Energieeffizienz und Fahrgastkomfort gesteigert werden. Dies stellt ein weiteres Potenzial der Cloudverlagerung im Sinne von **Forschungsfrage 3** dar.

7.3 Das Barebone-Fahrzeug

Ein zukunftsweisendes Konzept in der Fahrzeugentwicklung könnte das Barebone-Fahrzeug sein, das sich auf die grundlegende Funktion des Fahrens beschränkt. In dieser Architektur dient das Fahrzeug primär als Hardwareplattform, auf der fortgeschrittene Softwarekomponenten nicht lokal installiert sind, sondern dynamisch über die Cloud oder Edge-Computing-Plattformen bereitgestellt werden.

Im Wesentlichen wäre das Barebone-Fahrzeug mit minimaler, robuster Infrastruktur für grundlegende Fahrzeugoperationen ausgestattet, etwa zur Steuerung der Antriebseinheit, der Bremsen und grundlegend sicherheitsrelevanter Systeme. Alle weiteren Features, wie Infotainment, Fahrerassistenzsysteme, Navigationssoftware und Regelungsalgorithmen, werden dynamisch heruntergeladen oder direkt als cloud- oder edgebasierte Funktionen hinzugefügt.

A Anhang

A.1 Einige Grundbegriffe der Thermodynamik

Definition A.1 Wasserdampfpartialdruck: Der Wasserdampfpartialdruck p_{D} (Einheit: Pa) ist der Teildruck des in der Luft vorhandenen Wasserdampfes und bildet zusammen mit dem Teildruck der trockenen Luft p_{L} den Gesamtdruck p (ugs. Luftdruck). [32]

Definition A.2 Sättigungsdruck: Der Sättigungsdruck p_{DS} (Einheit: Pa) beschreibt den maximalen Wasserdampfpartialdruck in feuchter Luft. Der Sättigungsdruck ist eine Funktion der Temperatur und unabhängig vom Luftdruck. [32]

Definition A.3 Relative Feuchte: Die relative Feuchte φ (Einheit: $\%$) beschreibt den Anteil der möglichen Wasserdampfaufnahme in ungesättigter Luft:

$$\varphi = \frac{p_{\mathrm{D}}}{p_{\mathrm{DS}}} \tag{A.1}$$

Definition A.4 Feuchtegrad: Der Feuchtegrad x (Einheit: $\mathrm{kg\,kg^{-1}}$) beschreibt die Masse an Wasserdampf im Verhältnis zur Masse der trockenen Luft:

$$x = \frac{m_{\mathrm{D}}}{m_{\mathrm{L}}} \tag{A.2}$$

A.2 Coefficient of Performance einer Wärmepumpe

Definition A.5 Coefficient of Performance: Der Coefficient of Performance (COP) bezeichnet bei einer Wärmepumpe das Verhältnis von nutzbarer Wärme-/Kälteleistung $\dot{Q}_{\mathrm{Kond/Verd}}$ zu eingebrachter Kompressorleistung $\dot{W}_{\mathrm{Komp}}$ [47]. Im Heiz- bzw. Kühlbetrieb berechnet sich der COP folgendermaßen:

$$COP_{\mathrm{Kühl}} = \frac{\dot{Q}_{\mathrm{Verd}}}{\dot{W}_{\mathrm{Komp}}} \tag{A.3}$$

$$COP_{\mathrm{Heiz}} = \frac{\dot{Q}_{\mathrm{Kond}}}{\dot{W}_{\mathrm{Komp}}} = \frac{\dot{Q}_{\mathrm{Verd}} + \dot{W}_{\mathrm{Komp}}}{\dot{W}_{\mathrm{Komp}}} = 1 + COP_{\mathrm{Kühl}} \tag{A.4}$$

A.3 Thermischer Komfort: Predicted Mean Vote

Das Predicted Mean Vote (PMV)-Modell ist ein weitverbreitetes Verfahren zur Bewertung des thermischen Komforts in Innenräumen. Es wurde von P. O. Fanger [40] im Jahr 1970 entwickelt und basiert auf einem thermophysiologischen Modell des menschlichen Körpers. Das PMV-Modell berechnet

eine thermische Komfortbewertung auf einer siebenstufigen Skala von -3 (kalt) bis +3 (heiß), wobei 0 als thermisch neutral gilt (vgl. Abbildung A.1).

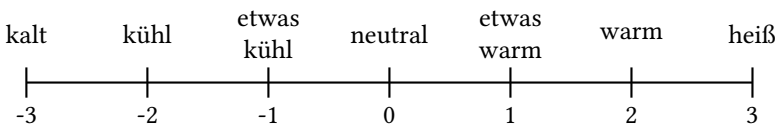


Abbildung A.1: Benennung der sieben Stufen der PMV-Skala

Grundlagen des PMV-Modells

Das PMV-Modell berücksichtigt sechs Hauptfaktoren, die den thermischen Komfort eines Menschen beeinflussen:

- **Lufttemperatur** – Die Temperatur der Raumluft.
- **Mittelstrahlungstemperatur** – Die mittlere Temperatur der umgebenden Oberflächen.
- **Luftgeschwindigkeit** – Die Geschwindigkeit der Luftbewegung im Raum.
- **Relative Luftfeuchtigkeit** – Der Feuchtegehalt der Luft.
- **Metabolische Rate (MET)** – Die Wärmeerzeugung durch den menschlichen Stoffwechsel, abhängig von der Aktivität.
- **Bekleidungswiderstand (CLO)** – Der thermische Widerstand der getragenen Kleidung.

Auf Basis dieser Parameter berechnet das PMV-Modell eine Durchschnittsbewertung des thermischen Empfindens einer Gruppe von Personen unter den gegebenen Bedingungen.

Bedeutung des PMV für die Raumklimatisierung

Das PMV-Modell ist von großer Bedeutung für die Gestaltung und Steuerung von Heizung, Lüftung, Klimatisierung (HLK)-Systemen. Ziel ist es, einen PMV-Wert nahe 0 zu erreichen, um ein angenehmes Raumklima zu gewährleisten. In internationalen Normen wie der ISO 7730[1] wird das PMV-Modell als Maßstab für den thermischen Komfort herangezogen.

A.4 Mikrocontroller

Das Herz einer jeden Electronic Control Unit (ECU) bildet ein oder mehrere Mikrocontroller, Application-specific integrated circuit (ASIC)s oder Field-Programmable Gate Array (FPGA)s. Diese verschiedenen Technologien werden je nach Anwendungsanforderungen eingesetzt: Mikrocontroller dienen als universelle Rechenplattformen für programmierbare software components (SWCs), ASICs implementieren hochoptimierte, anwendungsspezifische Funktionen direkt in Hardware, und FPGAs ermöglichen rekonfigurierbare Hardware-Implementierungen für spezielle Verarbeitungsaufgaben wie Signalverarbeitung oder parallele Algorithmen. Mikrocontroller bestehen aus den in Abbildung A.2 dargestellten Komponenten.

Dabei ist zu beachten, dass eine zunehmende Integration der Komponenten auf einen Chip durchgeführt wird. Dadurch wird der Mikrocontroller für sich alleine funktionsfähig und kann je nach Anforderung durch zusätzliche externe Bausteine wie Speicher ausgebaut werden. Moderne Mikrocontroller zeichnen sich durch folgende Eigenschaften aus:

- Einsatz von Mehrkernprozessoren
- Einsatz von Grafikprozessoren
- Unterstützung von Kryptografie zur Verschlüsselung der Daten

[1] `https://www.din.de/de/meta/suche/62730!search?query=DIN+EN+ISO+7730`

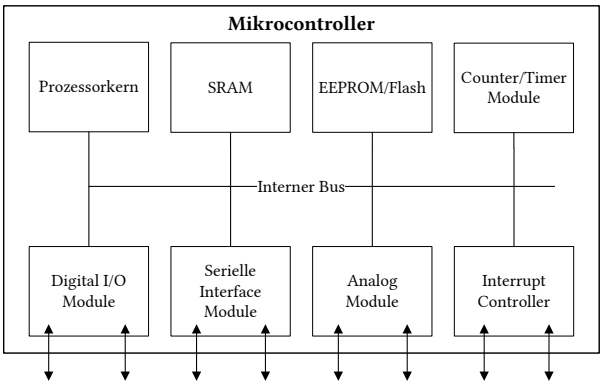


Abbildung A.2: Basislayout eines Mikrocontrollers [44]

- Möglichkeit der drahtlosen Vernetzung über WLAN oder andere Schnittstellen

Die Motivation für den Einsatz von Mehrkernsystemen ist auch hier bedingt durch das Ziel der Kostenersparnis. Mehr und mehr Funktionen sollen auf weniger ECUs zusammengeführt werden, sodass weniger Gehäuse, Elektronik, Verkabelung und Bauaufwand benötigt wird [70]. Die Zusammenlegung von Funktionen (unterschiedlicher Kritikalitätsstufen) auf eine ECU führt allerdings zu höherer Komplexität und damit auch zu einem höheren Zertifizierungsaufwand. Komplex wird in diesem Zusammenhang von der *European Aviation Safety Agency* folgendermaßen definiert [43]:

- „Mehr als eine eingebettete CPU und es wird ein gemeinsames Bussystem verwendet“
- „Mehrere Interfaces sind abhängig voneinander und tauschen Daten miteinander aus“

A.5 Zyklische und Streaming-basierte Funktionen im Automotive-Kontext

Im modernen Fahrzeug- und Cloud-Computing-Umfeld lassen sich Funktionen grundlegend nach ihrem Ausführungs- und Kommunikationsverhalten unterscheiden: **zyklische** und **streaming-basierte** Funktionen. Beide Konzepte adressieren unterschiedliche Anforderungen an Rechenleistung, Timing und Datenverarbeitung.

Streaming-basierte Funktionen

Streaming-basierte Funktionen zeichnen sich durch einen *kontinuierlichen Datenfluss* aus, der über ein Netzwerk transportiert wird. Die eingehenden Datenströme werden laufend und nahezu in Echtzeit verarbeitet, was zu einer kontinuierlich erhöhten CPU-Auslastung führt. Diese Art der Funktionen finden sich typischerweise bei Anwendungen, die große Datenmengen oder fortlaufende Sensordaten verarbeiten müssen.

Ein klassisches Beispiel ist das *Video-Streaming* von Kameras im Fahrzeug. Hier werden Videodaten permanent übertragen und verarbeitet, wodurch eine nachhaltige, hohe Rechen- und Kommunikationslast entsteht.

Zyklische Funktionen

Im Gegensatz dazu zeichnen sich zyklische Funktionen durch eine *periodische, zeitgesteuerte Ausführung* aus. Die CPU-Auslastung steigt dabei nur für die Dauer der Funktionsverarbeitung kurz an und fällt danach auf ein niedriges Niveau zurück, da keine permanente Berechnung erforderlich ist.

Die zyklische Verarbeitung lässt sich in zwei Falltypen unterteilen:

1. Fall 1 – Regulärer Zyklus mit festen Intervallen

Hierbei werden die Eingabedaten des Fahrzeugs in regelmäßigen, festen Zeitintervallen $t_{\text{intervall}}$ in die Cloud hochgeladen. Unmittelbar nach dem Upload wird eine Berechnung durchgeführt und das Ergebnis an das Fahrzeug zurückgesendet. Die gesamte Kette aus Upload, Verarbeitung und Download wiederholt sich in genau definierten Zeitabständen, wodurch ein deterministisches Verhalten entsteht.

Beispiel:
Die HLK-Regelung

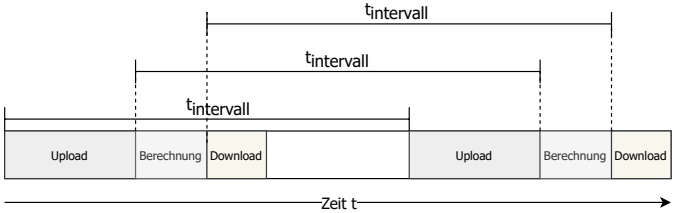


Abbildung A.3: Zeitlicher Ablauf einer zyklischen Funktion: Fall 1

2. Fall 2 – Zyklische Verarbeitung mit unterschiedlichen Intervallen

   In diesem Szenario unterscheiden sich das Upload-Intervall t_{Up} und das Intervall für Berechnung und Download t_{CD}. Beispielsweise werden Daten zunächst über einen längeren Zeitraum gesammelt, bevor sie für die Berechnung verwendet werden. Das Ergebnis wird dann in einem anderen Rhythmus zurückgesendet.

   Beispiel:
   Bestimmung des Fahrstils → Um den Fahrstil zu bestimmen, müssen zunächst Informationen über einen bestimmten Zeitraum gesammelt werden, bevor sie in die Berechnung einfließen können.

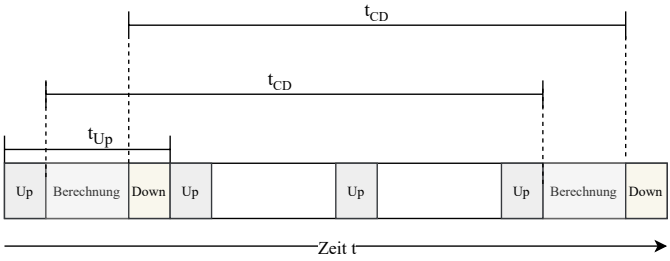


Abbildung A.4: Zeitlicher Ablauf einer zyklischen Funktion: Fall 2

A.6 X-in-the-Loop Testmethoden

Simulationen bzw. individuelle Simulationsmodelle sind ein wesentlicher Bestandteil des Softwareentwicklungsprozesses. Im Folgenden werden wichtige Elemente simulationsbasierter Methoden nach dem XiL-Ansatz aus Sicht der Testmethodik vorgestellt. Die Integration von Softwarefunktionen in das Fahrzeug folgt einem schrittweisen Ansatz, bei dem noch nicht verfügbare Hardwarekomponenten simuliert oder modelliert werden. Typischerweise wird die folgende Sequenz ausgeführt:

- Model-in-the-Loop (MiL): In der frühen Phase des MiL-Testens liegen nicht nur die Umgebungskomponenten, sondern auch das Testobjekt nur als ausführbares oder mathematisches Modell vor. MiL-Tests können auf Standard-PCs ausgeführt werden und für die Entwicklung von Funktionen genutzt werden, die zukünftig nicht mehr mit Software realisiert werden [104].
- Software-in-the-Loop (SiL): Der nächste Schritt erfolgt mit dem generierten Code der Vorgängermodelle. Dieser Code wird auf einer Plattform in der Nähe der Zielplattform getestet. SiL-Tests verwenden keine realen elektronischen Steuergeräte, sodass die Simulationsmodelle der Umgebung keine Echtzeitanforderungen erfüllen müssen [53].

- Hardware-in-the-Loop (HiL): Ein HiL-Test unterscheidet sich von den zuvor genannten Verfahren dadurch, dass das zu testende System in der Regel als reale Komponente vorliegt und in eine Simulationsumgebung integriert ist. Einzelne Umgebungsteile wie Aktoren oder Sensoren können auch als reale Hardware verfügbar sein, falls sie aufwändig zu modellieren sind oder die Echtzeitfähigkeit ihrer entsprechenden Modelle nicht gewährleistet ist [60].

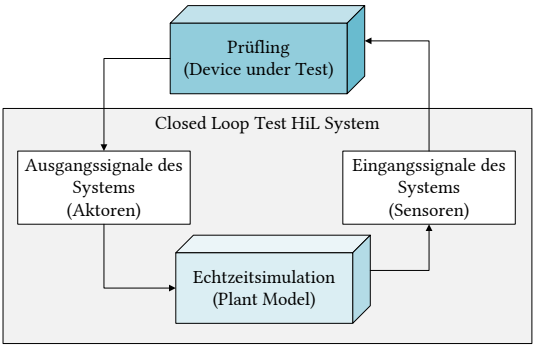


Abbildung A.5: Konzept Closed Loop Tests am HiL Prüfstand

Die genannten XiL-Verfahren lassen sich wiederum unterscheiden in sogenannte Closed-Loop und Open-Loop Tests. In der Open-Loop Testumgebung werden im XiL-Kontext die zu testenden Modelle oder Algorithmen mit Input-Daten simuliert. Die Output-Daten des Testobjekts werden von der durchführenden Testperson untersucht und mit den erwarteten Ergebnissen verglichen. Aufgrund der Einfachheit der Implementierung und des geringen Aufwands bei der Testdurchführung ist dieser Ansatz weit verbreitet [118]. Das Closed-Loop Testen beinhaltet das Zurückspeisen der Ausgangssignale des Systems an ein simuliertes Echtzeitmodell inklusive der Weiterleitung der Signale dieses Modells zurück in das System (Abbildung A.5). Closed-

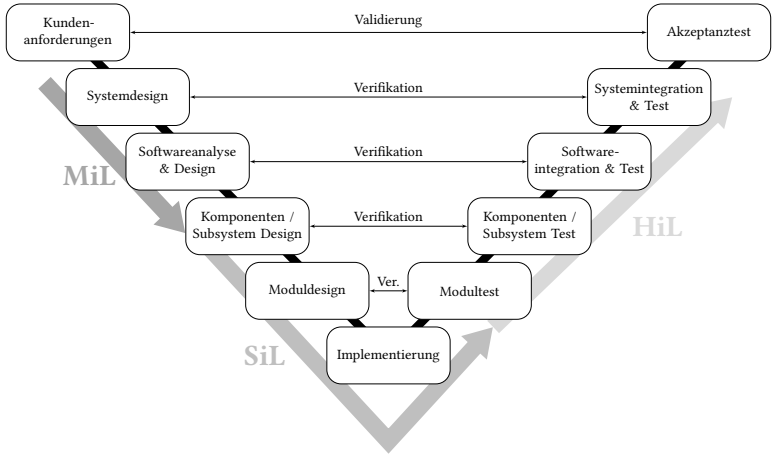


Abbildung A.6: V-Modell basierend auf [129] und [101]

Loop-Tests erfordern Echtzeitmodelle[2], die der realen Test-Hardware ein Feedback geben können, was mit einem hohen Implementierungsaufwand verbunden ist.

Unter Berücksichtigung des V-Modells lässt sich der beschriebene Integrationsablauf im Entwicklungsprozess von links nach rechts abbilden [129]. Die Positionierung von MiL, SiL und HiL in den einzelnen Schritten in Abbildung A.6 kann dabei variieren, da der Übergang zwischen ihnen nicht eindeutig einem bestimmten Schritt zugeordnet ist.

[2] Unter Echtzeit wird der Betrieb eines Rechensystems verstanden, bei dem die Datenverarbeitung der anfallenden Daten derart durchgeführt wird, dass die Ergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind [108].

A.7 ROS 2

Robot Operating System (ROS) 2 ist die Überarbeitung des ROS, das bei Willow Garage für den Forschungsroboter PR2 entwickelt wurde. Der PR2 ist ein humanoid wirkender Roboter mit zwei Greifarmen. ROS unterstützte dabei keine Echtzeitanforderungen und war auf stabile Netzwerke (kein Mobilfunk) angewiesen. Die weitreichende Überarbeitung ROS 2 behebt diese Thematiken und hat dadurch zu einem Durchbruch von ROS 2 aus dem Bereich der Forschung in die Industrie[3] ermöglicht.

Aufbau der ROS 2 API

Die ROS 2 Schichten können den OSI Schichten (s. Tabelle A.2) zugeordnet werden. Dabei wurde bei ROS 2 auf eine dezentrale Kommunikation gesetzt, während die ursprüngliche ROS Implementierung noch auf einem zentralen ROS-Master aufbaut. ROS Nodes bieten Services selbständig an oder abonnieren andere Services. Dabei wendet sich ROS 2 von der selbst entwickelten Middleware ab und es kommt Data Distribution Service (DDS) als Kommunikationsumgebung zum Einsatz (Abbildung A.7). Analog zu AUTOSAR Adaptive (Kapitel A.8) wird auch in ROS 2 jedes POSIX[4]-kompatible Betriebssystem unterstützt. Dazu gehört neben Windows auch die verschiedenen Linux Distributionen.

Die ROS 2 API verfügt über die zwei wesentlichen Schnittstellen der Middleware *rmw* und des Benutzers *rcl*.

Die *rmw* API ist die Schnittstelle zwischen dem ROS 2 Software-Stack und der zugrunde liegenden Middleware-Implementierung bestehend aus einer DDS- oder RTPS-Implementierung. Die Rolle der *rmw* Schicht ist das Finden, Abonnieren und Nutzen von Services mithilfe der unterschiedlichen

[3] https://rosindustrial.org/ric/current-members
[4] Portable Operating System Interface

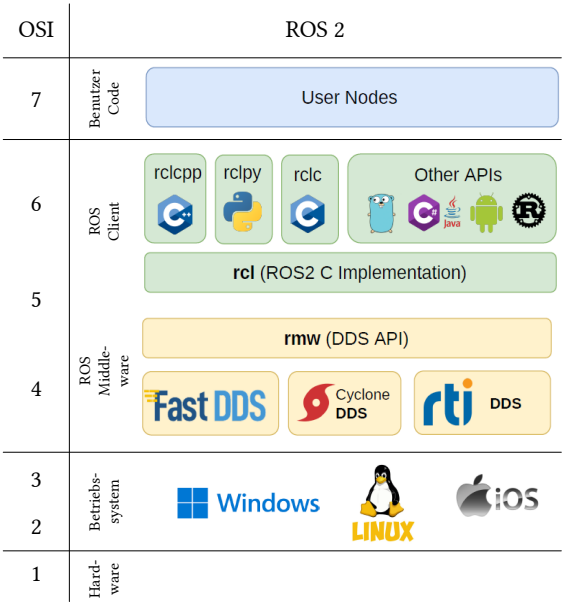


Abbildung A.7: ROS 2 Schichten und deren Zuordnung in das OSI-Modell

Mechanismen (Publish/Subscribe).
Die höher gelagerte *rcl* API stellt die Schnittstelle zu den Nutzeranwendungen dar. Entscheidend ist hierbei die Möglichkeit für Anwender, in den Sprachen C++, Python oder auch Java entwickeln zu können. ROS 2 bietet hierfür sprachspezifische Schnittstellen (rclpy etc.), die die Implementierungen der entsprechenden Datentypen zwischen den Programmiersprachen übersetzen können.

Die Nachrichtentypen in ROS 2

ROS 2 stellt Übersetzungsmechanismen zur Verfügung, die eine Unabhängigkeit der Nachrichtentypdefinition von der Middleware der Applikationssoft-

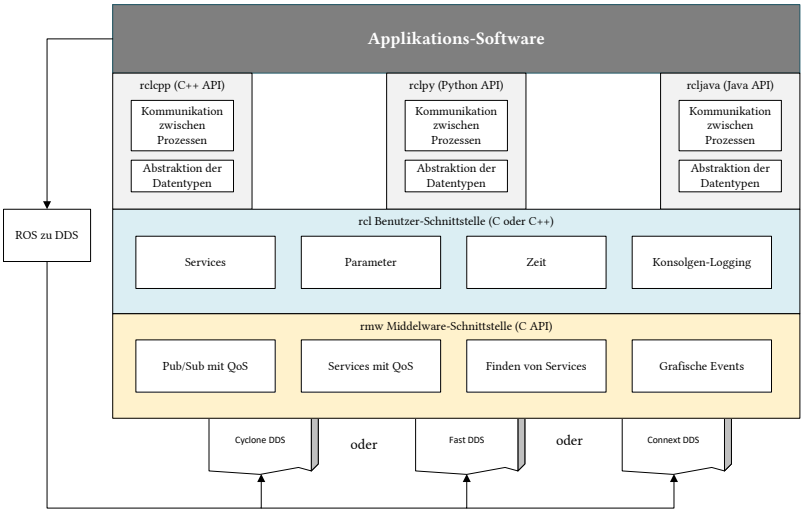


Abbildung A.8: ROS 2 API Überblick, nach [92]

ware gestatten. Die Mechanismen umfassen die Definition des Nachrichtenformats in Form einer *.msg* Datei bis hin zur sprachspezifischen Umsetzung in Form von Programmcode und Header-Files [111]. Die Übersetzung kann statisch oder dynamisch erfolgen.

.msg-Datei

Messages sind zentraler Bestandteil des ROS 2 Interface. Innerhalb der *.msg*-Dateien werden die Nachrichtentypen definiert, aus denen wiederum mithilfe von ROS 2 Tools automatisch interpretierbarer Quelltext für verschiedene Programmiersprachen erzeugt wird. *.msg*-Dateien beinhalten *Felder* und *Konstanten*. Felder bestehen aus dem Datentyp und der Bezeichnung. Die Datentypen entsprechen den typischen Datentypen aus der C++ Umgebung. Definierte Nachrichtentypdefinitionen können selbst wieder Datentypen darstellen und somit verschachtelt werden. Konstanten entsprechen Feldern mit einem Standardwert, der zur Laufzeit nicht verändert werden kann.

Kommunikationsarten in ROS 2

Der Publisher-Subscriber-Mechanismus ermöglicht eine asynchrone, viele-zu-viele Kommunikation über sogenannte *Topics*.

- **Publisher** senden Nachrichten zu einem bestimmten Topic.
- **Subscriber** empfangen Nachrichten von einem Topic.
- Mehrere Publisher und Subscriber können dasselbe Topic verwenden.
- Nachrichten werden nur empfangen, wenn ein Subscriber aktiv ist.

Das Service-Client-Modell ermöglicht synchrone Anfragen und Antworten über definierte *Services*.

- **Client-Knoten** sendet eine Anfrage.
- **Service-Knoten** verarbeitet die Anfrage und sendet eine Antwort zurück.
- Die Kommunikation erfolgt über eine Service-Schnittstelle mit einer festgelegten Struktur (Request/Response).

Actions sind eine Erweiterung des Service-Client-Modells für langlaufende Prozesse mit Zwischenupdates.

- Der **Client** sendet eine Anfrage zur Ausführung einer Aktion.
- Der **Server** verarbeitet die Anfrage und sendet regelmäßige Statusupdates.
- Die Aktion kann während der Ausführung abgebrochen oder neu gestartet werden.

Parameter bieten eine Möglichkeit zur Verwaltung von Konfigurationswerten für Knoten.

- Jeder Knoten kann **Parameter setzen, lesen und aktualisieren**.
- Parameter sind oft für Einstellungen wie Sensorkalibrierung oder Steuerungsparameter relevant.
- Sie können über die Kommandozeile oder über eine API geändert werden.

Lifecycle Nodes ermöglichen eine gezielte Steuerung des Knotenzustands.

- Standardzustände: *unconfigured, inactive, active, finalized.*
- Ein Knoten kann erst aktiv werden, wenn alle Abhängigkeiten erfüllt sind.
- Erhöht die Kontrolle über die Systeminitialisierung und das Fehlerhandling.

A.8 AUTOSAR Adaptive

Die AUTOSAR Adaptive Plattform [11] ist die Definition einer serviceorientierten Architektur bestehend aus Laufzeitsystem, Schnittstellen und

Funktionalitäten, die durch das AUTOSAR (AUTomotive Open System ARchitecture) Konsortium entwickelt wurde. Die Plattform versucht vor allem, den steigenden Anforderungen der Rechenleistung im Fahrzeug gerecht zu werden [136], für die die AUTOSAR Classic Plattform (Beginn der Arbeiten im Jahr 2003) nicht ausgelegt ist. Die stark gestiegenen Anforderungen an Rechenleistung und mögliche kontinuierliche Aktualisierbarkeit der Software sind durch Funktionen wie automatisiertes Fahren und künstliche Intelligenz, die im Fahrzeug Einzug halten, entstanden. AUTomotive Open System ARchitecture (AUTOSAR) Classic wurde für Mikrocontroller mit einem Prozessorkern entwickelt, während Adaptive die zunehmende Parallelisierung durch Erhöhung der Anzahl der Prozessorkerne durch Multi-Core Prozessoren unterstützt [111]. Neben der Möglichkeit, leistungsstärkere Prozessoren zu verwenden, ist der Einsatz von Ethernet ein Technologietreiber, der mit Adaptive im Fahrzeug stärker als bisher in Classic umgesetzt wird. Der Ethernet-Standard ermöglicht nicht nur eine höhere Bandbreite gegenüber den bisher verbreiteten signalbasierten Standards Controller Area Network (CAN) und Local Interconnect Network (LIN) der Kommunikation im Fahrzeug, sondern auch den Aufbau einer SOA.

Struktur von AUTOSAR Adaptive

Die Grundlage der logischen Struktur von AUTOSAR Adaptive (s. Abbildung A.9) bildet die Hardware eines Steuergeräts oder einer virtuellen Maschine. Darauf wird ein POSIX-kompatibles Betriebssystem aufgesetzt (s. Definition A.6).

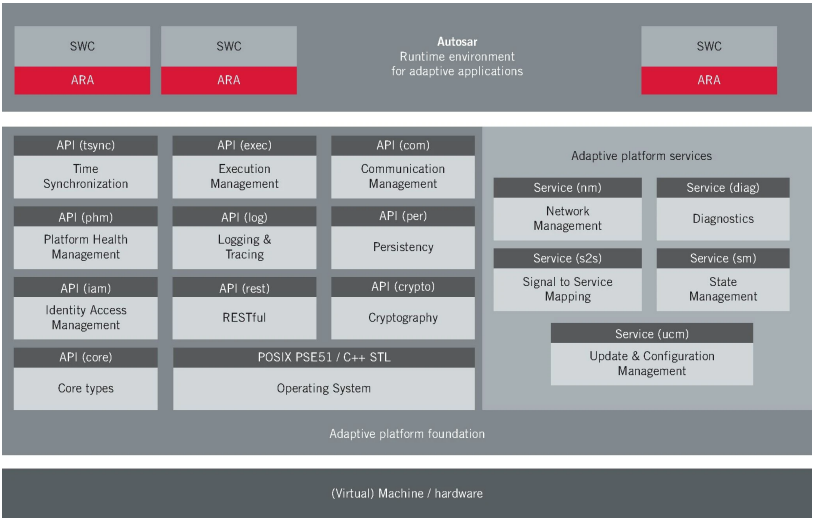


Abbildung A.9: Komponenten der AUTOSAR Adaptive Software [136]

> **Definition A.6 POSIX:** IEEE 1003.1 „IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) Base Sepcifications“ bezeichnet POSIX als eine Standard-Betriebssystemschnittstelle und -umgebung, inklusive Befehlsinterpreter und allgemeinen Dienstprogrammen, die die Portabilität von Programmen auf Quellcodeebene gewährleisten soll. [55]

Oberhalb des Betriebssystems sind die funktionellen Cluster angesiedelt. Diese Cluster bilden die Schnittstelle zu den Adaptive Services. Dabei wird zwischen *Adaptive Platform Foundation* und *Adaptive Platform Services* unterschieden. Die *Foundation* umfasst die fundamentalen Funktionalitäten des Adaptive Stacks, während letztere Standard-Services von AUTOSAR Adaptive beschrieben werden. Die oberste Ebene ist die AUTOSAR Runtime Environment, auf der schließlich die Adaptive Applikationen ausgeführt werden können.

Relevante funktionelle Cluster der AUTOSAR Adaptive Plattform

Communication Management

Das *Communication Management* ist für die Kommunikation zwischen Anwendungen auf derselben ECU, also auch für die Kommunikation zwischen Anwendungen in verteilten Systemen auf verschiedenen ECU zuständig. AUTOSAR Adaptive (AUTOSAR Adaptive R22-11) unterstützt die serviceorientierte Kommunikation mittels der Middleware Protokolle SOME/IP (s. Anhang A.12) und DDS (s. Anhang A.12), sowie IPC[5] und Signal PDU (als signalbasierte Kommunikationsform). Die serviceorientierte Kommunikation von AUTOSAR bietet einen dynamischen Aufbau von Kommunikationspfaden. Die zentrale Komponente für die Laufzeiterkennung ist die *Service Registry*. Applikationen, die Dienste anbieten, registrieren diese in der *Service Registry*. Konsumenten finden diese Dienste, indem sie die Registry abfragen. Danach können sie Dienste direkt beim Anbieter aufrufen. Dieser Vorgang wird als Service Discovery bezeichnet. Ähnlich wie beim Dienstzugriff werden auch die Service-Discovery-Aufrufe auf das spezifische Middleware-Protokoll abgebildet [6].

Execution Management

Das *Execution Management* ist für die Verwaltung der Ausführung der Plattform und der Anwendungen zuständig. Dazu gehört unter anderem die Initialisierung der Plattform sowie das Starten und Herunterfahren von Anwendungen. Das *Execution Management* konfiguriert die Plattform und die Anwendungen gemäß einem Manifest, dem *Execution Manifest*. AUTOSAR-Manifeste sind XML-basierte Dateien, die Dienste, Anwendungen, die zugrunde liegenden Maschinen und deren Konfiguration beschreiben. Ein

[5] Interprozesskommunikation

wichtiger Aspekt ist zudem die Zusammenarbeit zwischen dem *Execution Manager* und dem Betriebssystem für die Konfiguration von *Scheduling Policies*. Mithilfe dieser können kritischere Anwendungen mit einer höheren Priorität eingeplant werden.

Entwicklungsprozess in AUTOSAR Adaptive

Das Entwicklungsvorgehen in AUTOSAR Adaptive beschreibt Services, Anwendungen, Maschinen sowie deren Konfiguration und Verhalten untereinander. Der gesamte Entwicklungsprozess ist in Abbildung A.10 abgebildet und wird farblich zwischen ausführbaren Anwendungen in Orange, Beschreibungen und Manifesten in Hellorange und Arbeitsschritten in Grau unterschieden.

A.9 Weitere Netzwerkprotokolle

Für die Integration der Cloud in die Architektur des Fahrzeuges ist eine netzwerkübergreifende Kommunikation auf der Applikationsschicht notwendig. Message Queuing Telemetry Transport (MQTT) und Hypertext Transfer Protocol (HTTP) sind zwei gängige Protokolle für die Datenübertragung in netzwerkbasierten Anwendungen [22]. Während MQTT analog zu den Topics aus ROS 2 dem Publish-Subscribe-Modell zugrunde liegt, basiert HTTP auf dem Request-Response-Modell wie auch die ROS 2-Services. Beide Protokolle unterstützen das Verschlüsselungsprotokoll Transport Layer Security (TLS), wobei lediglich MQTT eine Konfiguration der Quality of Service (QOS)-Levels bietet. HTTP ist auf das Versenden von Textnachrichten beschränkt und erlaubt die Kommunikation immer nur zwischen genau einem Server und einem Client. Die Nachrichtengröße ist bei MQTT auf 256 MB beschränkt, wohingegen bei HTTP diesbezüglich kein Limit vorliegt [28, 29].

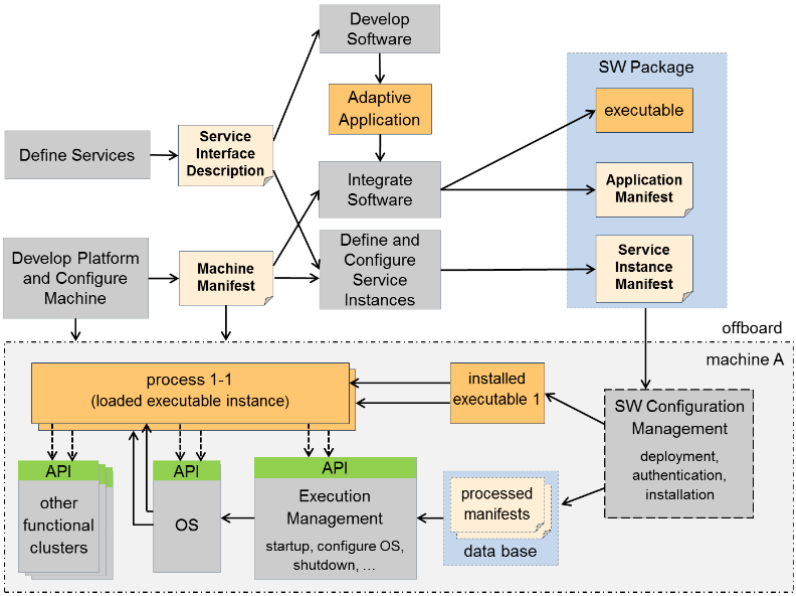


Abbildung A.10: Entwicklungsprozess in AUTOSAR Adaptive [11]

A.10 Software Architektur von Steuergeräten

Das Software Design innerhalb der Automobilindustrie wird durch die Randbedingungen der Robustheit und Zuverlässigkeit geprägt, weswegen einige Softwarearchitekturstile aus der IT-Welt nicht in der Automobilwelt Einzug halten [110]. Die Grundlage aller Automotive Software Architekturen bildet die Schichtenarchitektur (engl. layered architecture), die auf das Prinzip der Hierarchie aufbaut (vgl. AUTOSAR Classic in Abbildung A.11). Komponenten werden aufeinander aufgesetzt und Funktionsaufrufe sind nur in eine Richtung von höheren Ebenen zu niedrigeren Ebenen möglich (s. Abbildung 2.12). Die Applikationssoftware auf der obersten Ebene ist klar von der Hardware durch verschiedene Abstraktionsschichten getrennt.

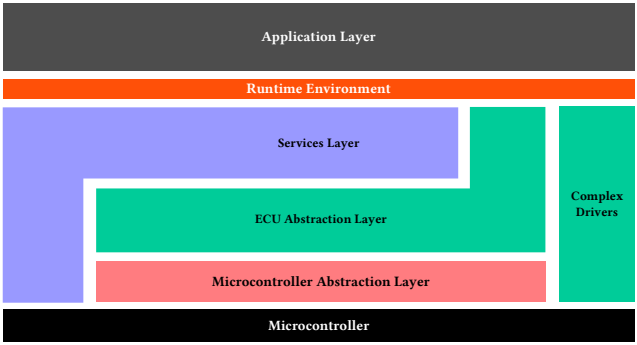


Abbildung A.11: Das AUTOSAR Classic Schichtenmodell

A.11 Middleware

Eine Middleware Architektur postuliert einen zentralen Broker, der die Kommunikation zwischen Anwendungen und Ressourcen so vermittelt, dass die Komplexität dieser Anwendungen und ihrer Infrastruktur für den Benutzer nicht sichtbar ist. Die Middleware stellt also die Vermittlungsplattform zwischen verschiedensten Anwendungen dar, die auf unterschiedlichen Rechenumgebungen innerhalb eines verteilten Informationssystems ausgeführt werden können [113].

Die typischen Dienste einer Middleware sind [19]:

- Kommunikationsmanagement: Peer-to-Peer messaging, remote procedure call...
- Systemmanagement: Event notification service, recovery coordinator...
- Informationsmanagement: Dateimanager, Datenbanksystem...
- Ablaufkontrollmanagement: Thread manager, resource broker...
- Berechnungsdienste: Datenkonvertierung, Sortierverfahren...

Middleware Architekturen sind im Bereich des Cloud Computing nicht mehr wegzudenken und spielen durch die Vernetzung des Fahrzeugs mit

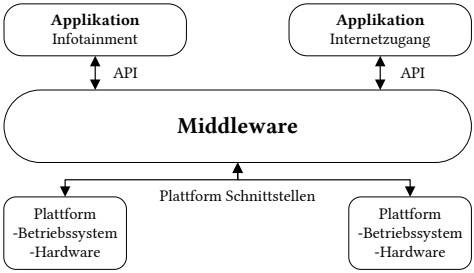


Abbildung A.12: Struktur einer Middleware Architektur

der Cloud auch innerhalb des Fahrzeugs eine immer wichtigere Rolle. Die Einordnung einer Middleware in das OSI-Schichtenmodell kann in Anhang A.14 nachgelesen werden.

A.12 Middleware Kommunikationsprotokolle

Data Distribution Service (DDS)

DDS wurde für den Einsatz im Umfeld von Industrie 4.0 entwickelt und erhält durch die Umsetzung in ROS 2 und AUTOSAR Adaptive auch Einzug in Fahrzeuge. Zentrales Element ist das Publish-Subscriber Konzept, welches zudem die Einstellung verschiedener QoS-Policies ermöglicht (s. Kapitel A.4). Das tatsächliche Protokoll, mit den Implementierungen über verschiedene Netzwerkprotokolle Daten austauschen, wird nicht von DDS vorgegeben. Somit ist DDS über der Transportschicht des OSI Schichtenmodells einzuordnen (s. Kapitel A.14) und baut auf einem entsprechenden Protokoll zur Übertragung der Nutzdaten (auch *wire protocol* genannt) auf.

Scalable Service-Oriented Middleware over IP (SOME/IP)

SOME/IP stellt ein Middleware Protokoll auf den Open Systems Interconnection (OSI) Schichten (s. Kapitel A.14) Session, Presentation und Application dar und ist Teil des AUTOSAR Adaptive-Standards. SOME/IP ist für den Einsatz in serviceorientierten Architekturen geeignet und bietet somit die grundsätzlichen Möglichkeiten, Daten schnell und effizient zwischen Services auszutauschen. SOME/IP unterstützt die folgenden vier Kommunikationskonzepte:

- **RPCs** Entfernte Methodenaufrufe (engl. Remote Procedure Calls) werden in *Fire & Forget* und *Request-Response* unterschieden. Bei der ersten Variante ruft der Client die angebotene Methode auf und erwartet keinen Rückgabewert. Bei der *Request-Response* Variante erhält der Client eine Antwortnachricht.
- **Ereignis (Event)** Ein Service kann eine oder mehrere Eventgruppen anbieten, welche von interessierten Clients abonniert werden können. Sobald sich der Zustand der zur Eventgruppe gehörenden Felder in einer definierten Weise ändert, sendet der Service ein Benachrichtigungsframe (Event) mit dem neuen Zustand an alle abonnierenden Clients.
- **Felder** Datenfelder, die einen Wert repräsentieren und ein Event auslösen, falls sich der Wert verändert. Felder können durch den Client ausgelesen oder geändert werden.

Die Services finden und registrieren sich durch das Service Discovery Protokoll des Standards [10]. Das Protokoll unterscheidet auch hier zwischen zwei Mechanismen für die dynamische Adressermittlung von Services. Die erste Möglichkeit besteht darin, dass der Server mittels zyklischer Multicast Nachrichten den Service im gesamten Netzwerk offeriert. Die Alternative besteht in der aktiven Suche eines Services durch den Client mithilfe des Find-Packet.

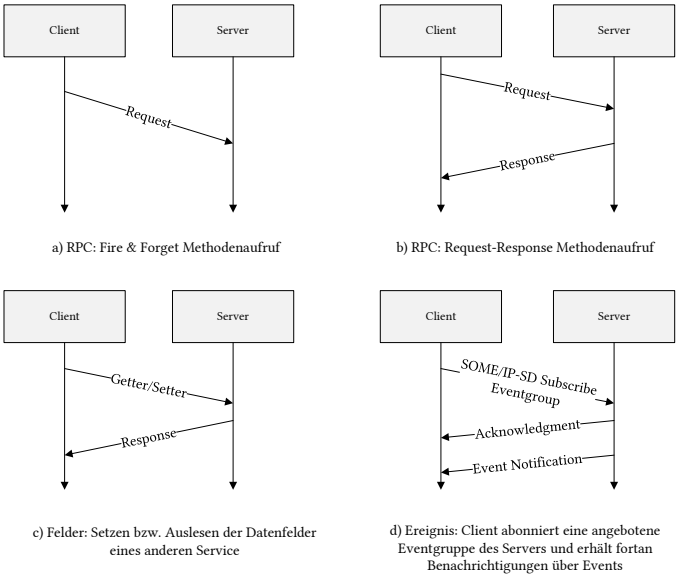


Abbildung A.13: SOME/IP Kommunikationsparadigmen

A.13 Funktionale Sicherheit im Automobil

Die ISO 26262 (Functional Safety for Road Vehicles) ist der internationale Standard für die funktionale Sicherheit von Straßenfahrzeugen mit einem zulässigen Gesamtgewicht von max. $3{,}5\,\mathrm{t}$. Entsprechend dieses Standards wird die funktionale Sicherheit als das „Nichtvorhandensein eines unangemessenen Risikos aufgrund von Gefahren, die durch fehlerhaftes Verhalten von E/E Systemen verursacht werden“ definiert. Das Rahmenwerk ist in zehn Teile gegliedert. Die Teile 3 bis 7 beziehen sich auf den Lebenszyklus eines Fahrzeugs, während die übrigen Teile hiervon unabhängig sind:

- Teil 1: Glossar
- Teil 2: Management der funktionalen Sicherheit

- Teil 3: Konzeptphase
- Teil 4: Produktentwicklung Systemebene
- Teil 5: Produktentwicklung Hardware-Ebene
- Teil 6: Produktentwicklung Software-Ebene
- Teil 7: Produktion und Betrieb
- Teil 8: Unterstützende Prozesse
- Teil 9: ASIL und sicherheitsorientierte Analysen
- Teil 10: Orientierungshilfen

Die Konzeptphase des dritten Teils des Standards ist für diese Dissertation relevant. Dieser Teil befasst sich mit der Betrachtung von Gefährdungen und der Einschätzung von Risiken, welche im Zusammenhang mit der funktionalen Sicherheit von sicherheitsbezogenen Fahrzeugsystemen bestehen [52]. Hauptbestandteil ist die Klassifizierung von Gefährdungen in sogenannte Automotive Safety Integrity Level (ASIL), die wiederum als Basis für die Anforderungen an die betrachteten Systeme oder Teilsysteme dienen. Innerhalb der ISO wird dabei der Begriff *item* benutzt. Im ersten Schritt ist das *Item* zu definieren. Unter einem *Item* versteht die ISO 26262 eine Funktion, ein System oder eine Kombination von Systemen, die auf Fahrzeugebene eine bestimmte Aufgabe realisieren. Die Definition des *item* steht laut ISO 26262 am Anfang eines jeden Entwicklungsprojekts.

Gefährdungs- und Risikoanalyse

Die Bestimmung eines ASIL erfolgt durch die Gefährdungs- und Risikoanalyse (engl. Hazard analysis and risk assessment (HARA)) des *item*.
Gefährdungen werden anhand folgender Faktoren und deren Bewertungsstufen klassifiziert:

- Schwere eines möglichen Schadens (engl. Severity *S*)
  S0: Keine Verletzungen
  S1: Leichte bis mittlere Verletzungen
  S2: Schwere Verletzungen, Überleben wahrscheinlich

S3: Lebensgefährliche Verletzungen, Überleben unwahrscheinlich

- Häufigkeit der Fahrsituationen (engl. Probability of Exposure *E*)
  - E0: Unvorstellbar
  - E1: Sehr niedrige Wahrscheinlichkeit
  - E2: Niedrige Wahrscheinlichkeit
  - E3: Mittlere Wahrscheinlichkeit
  - E4: Hohe Wahrscheinlichkeit
- Beherrschbarkeit durch den Fahrer (engl. Controllability *C*)
  - C0: Im Allgemeinen beherrschbar
  - C1: Einfach beherrschbar
  - C2: Normalerweise beherrschbar
  - C3: Schwierig oder nicht beherrschbar

Auf Grundlage der Einstufung des *Items* in den Faktoren *Severity*, *Exposure* und *Controllability* erfolgt die Festlegung des ASIL (s. Tabelle A.1), welches die Werte QM, A, B, C oder D annehmen kann. QM steht für „Qualitätsmanagement“ und repräsentiert den niedrigsten Kritikalitätsgrad. In Fällen, in denen der ASIL-Wert mit QM eingestuft wird, sind keine speziellen sicherheitsgerichteten Maßnahmen gemäß ISO 26262 erforderlich. Die Entwicklung erfolgt hier lediglich unter Berücksichtigung allgemeiner Qualitätsanforderungen.

Anders verhält es sich bei ASIL A bis D: Hier sind spezifische sicherheitsbezogene Maßnahmen verpflichtend, um das mit der jeweiligen Gefährdung verbundene Risiko auf ein akzeptables Maß zu reduzieren. ASIL D stellt dabei die höchste Stufe dar und erfordert den umfassendsten Maßnahmenkatalog im Hinblick auf funktionale Sicherheit. Zu den typischen sicherheitsgerichteten Maßnahmen zählen unter anderem: systematische Risikoanalysen, sicherheitsrelevante Design- und Architekturentscheidungen, formale Reviews sowie Verifikations- und Validierungsaktivitäten. Ergänzend werden Maßnahmen auf Hardware- und Softwareebene ergriffen, wie etwa die Implementierung von Fehlererkennungsmechanismen, Redundanzkonzepten oder Sicherheitsüberwachungsfunktionen. Ziel all dieser Maßnahmen ist es,

Tabelle A.1: ASIL Bestimmung nach ISO 26262

		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

potenzielle Fehler frühzeitig zu erkennen, ihre Auswirkungen zu begrenzen oder das System in einen sicheren Zustand zu überführen.

A.14 OSI Referenzmodell

Das OSI-Modell ist ein von der International Telecommunication Union (ITU) entwickelte und herausgegebene Spezifikation zur Klassifikation von Kommunikation zwischen verschiedenen Netzwerkteilnehmern. Das Modell unterteilt sich in 7 Schichten, mit einer steigenden Abstraktion von Schicht 1 bis 7 (s. Tabelle A.2). Grundsätzlich kann das OSI-Modell auf verschiedenste Systeme angewendet werden. Die Software-Architektur im Fahrzeug (s. Kapitel A.10) oder in der Cloud kann ebenfalls durch die sieben Schichten des OSI-Modells abgebildet werden. Oftmals werden in diesem Zusammenhang die Schichten 4 bis 6 zu einer sogenannten Middleware-Schicht zusammengefasst (Kapitel A.12), die auf den Schichten Betriebssystem (Schichten 2 und 3 des OSI-Modells) und Hardware (Schicht 1 des OSI-Modells) aufbaut.

Tabelle A.2: Schichten des OSI-Modells [80]

Schicht	Titel	Funktionen	Standard (ISO-OSI)	Beispiel Protokoll
7	Application	Austausch der Nutzdaten spezieller Anwendungen	ISO 8649	HTTP, CANopen
6	Presentation	Systemunabhängige Bereitstellung	ISO 8823	ASN.1
5	Session	Steuerung logischer Verbindungen	ISO 8326	ISO 8327
4	Transport	Übertragung der Nutzdaten	ISO 8073	TCP, UDP
3	Network	Schalten von Verbindungen	ISO 8348	IP
2	Data Link	Sicherung der Übertragung	ISO 8886	Ethernet
1	Physical	Elektrische Konnektivität	ISO 10022	WLAN, RS232

A.15 Hierarchieebenen von E/E-Features

Die nachfolgenden Hierarchieebenen basieren auf der Dissertation von Bach [12]. Beispielhafte Features der Ebenen sind in Abbildung A.14 dargestellt.

- **Integrierte Features** Eng mit der mechanischen Domäne des Fahrzeugs verknüpft. Dadurch entsteht eine unmittelbare Nähe zu spezifischen mechanischen Einheiten, die häufig eine Ausführung auf einer dedizierten ECU mit sich bringt. Sensoren und Aktoren, die für die Aufgabe der ECU erforderlich sind, sind direkt an die dedizierte ECU angeschlossen.
- **Verteilte Fatures** Kombination einzelner Komponenten unterschiedlicher Domänen, um neue Funktionalitäten zu ermöglichen. Diese Features erfordern nicht zwingend mechanische Hardwarekomponenten.
- **Quervernetzte Features** Verbindung mehrerer Funktionselemente miteinander. Das funktionale Verhalten hängt von der koordinierten Beeinflussung der unabhängigen Komponenten getrennter Domänen ab. Nutzen des Sensornetzwerks des gesamten Fahrzeugs für eine vollständige Repräsentation des Fahrzeugzustandes.

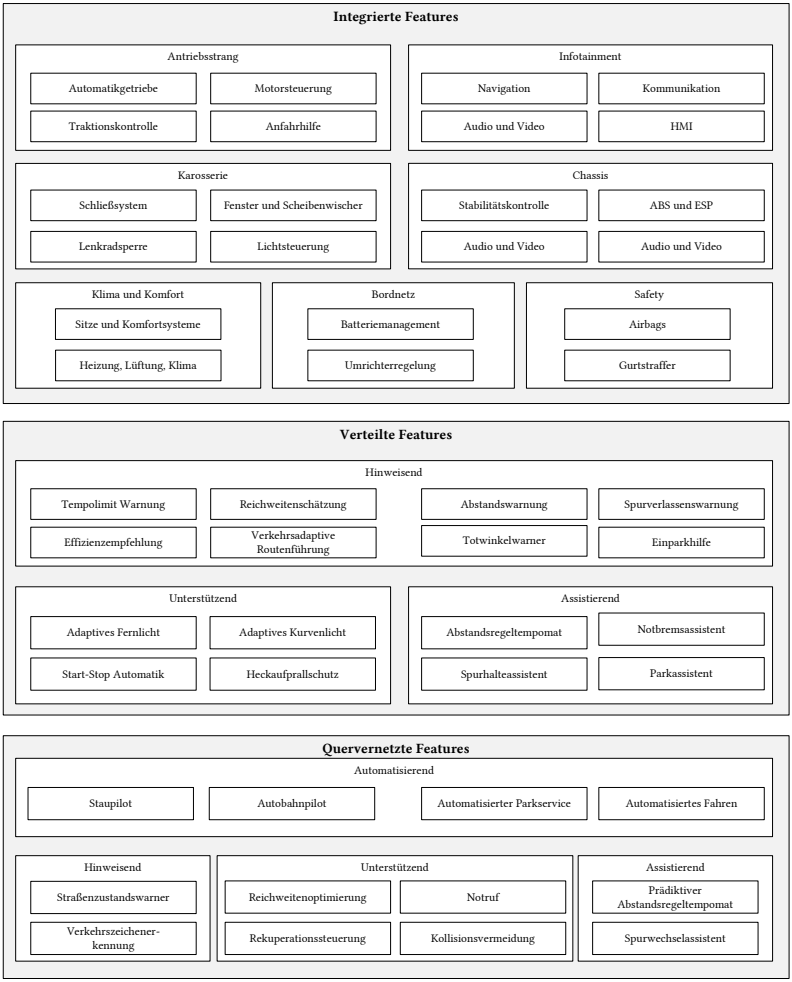


Abbildung A.14: Features im Fahrzeug und deren Einordnung bezüglich der Vernetzung [12]

A.16 Tabellen

Tabelle A.3: Schlüsselmerkmale einer generischen Softwarekomponente [114]

Merkmal	Beschreibung
Modularität	Unterteilt ein System in eigenständige, austauschbare Einheiten, die jeweils eine spezifische Funktion erfüllen.
Kapselung	Bündelt Daten und die darauf operierenden Methoden, schränkt den direkten Zugriff ein und verbirgt interne Details hinter einer öffentlichen Schnittstelle.
Abstraktion	Vereinfacht komplexe Systeme, indem nur wesentliche Eigenschaften offengelegt und irrelevante Details verborgen werden.
Wiederverwendbarkeit	Ermöglicht die Integration von Komponenten in verschiedene Anwendungen oder deren Nutzung in verschiedenen Teilen derselben Anwendung.
Substituierbarkeit	Eine Komponente kann durch eine andere ersetzt werden, die dieselbe Schnittstelle einhält, ohne die Systemfunktionalität zu beeinträchtigen.
Interoperabilität	Komponenten können miteinander kommunizieren und zusammenarbeiten, auch wenn sie mit unterschiedlichen Technologien oder Plattformen erstellt wurden.
Unabhängigkeit	Komponenten haben minimale Abhängigkeiten von anderen Komponenten und können in verschiedenen Umgebungen und Kontexten funktionieren.
Klar definierte Schnittstellen	Die Kommunikation zwischen Komponenten erfolgt ausschließlich über explizite, präzise spezifizierte Schnittstellen, die die Interaktion regeln.

Tabelle A.4: Die QOS Policies des Basis Profils in ROS 2 [91]

Bezeichnung	Beschreibung
History	**Keep last:** Maximal N Werte werden gespeichert **Keep all:** Alle Werte bis zum Limit der zugrundeliegenden Middleware werden gespeichert.
Depth	Im *History* Fall *keep last* wird die Anzahl zu speichernder Werte angegeben.
Reliability	**Best effort:** Es wird versucht Werte zu senden, es können jedoch Werte bei unzuverlässiger Netzwerkverbindung verloren gehen. **Reliable:** Garantie, dass alle Werte zugestellt werden. Notfalls werden Werte mehrfach gesendet.
Durability	**Transient local:** Der Publisher ist dafür verantwortlich Werte für später beitretende Subscriber zu vorzuhalten. **Volatile:** Werte für noch nicht verbundene Subscriber werden nicht zwischengespeichert.
Deadline	**Duration:** Die erwartete maximale Zeitspanne zwischen zwei aufeinanderfolgenden Nachrichten eines Topics
Lifespan	**Lifespan:** Die maximale Zeitspanne zwischen dem Senden und Empfangen einer Nachricht, bevor diese als ungültig erachtet und verworfen wird.
Liveliness	**Automatic:** Das System nimmt an, dass alle Publisher eines Node lebendig sind, sobald ein Publisher eine Nachricht versendet hat. **Manual by topic:** Publisher muss dem System aktiv mitteilen, dass er noch lebendig ist.
Lease Duration	**Duration:** Maximale Zeitdauer, die das System wartet, bevor es einen Publisher als tot erachtet.

A.17 Grafiken

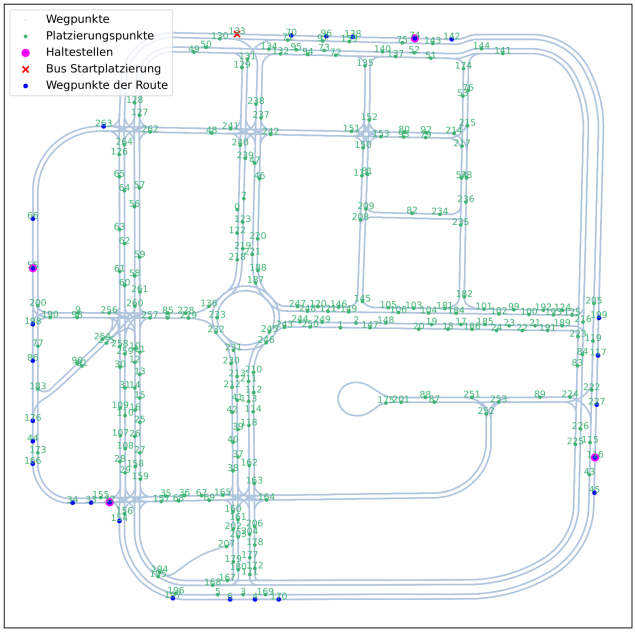


Abbildung A.15: Busroute mit Bushaltestellen in der CARLA Simulationsumgebung

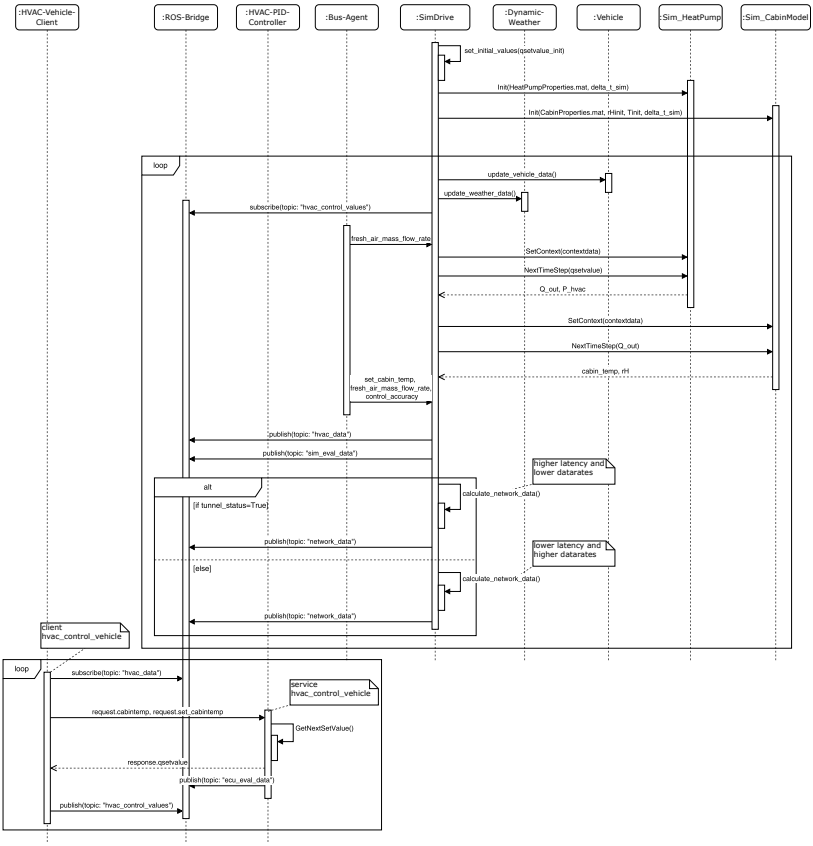


Abbildung A.16: Sequenzdiagramm der Ausführung des integrierten fahrzeuginternen PID-Reglers

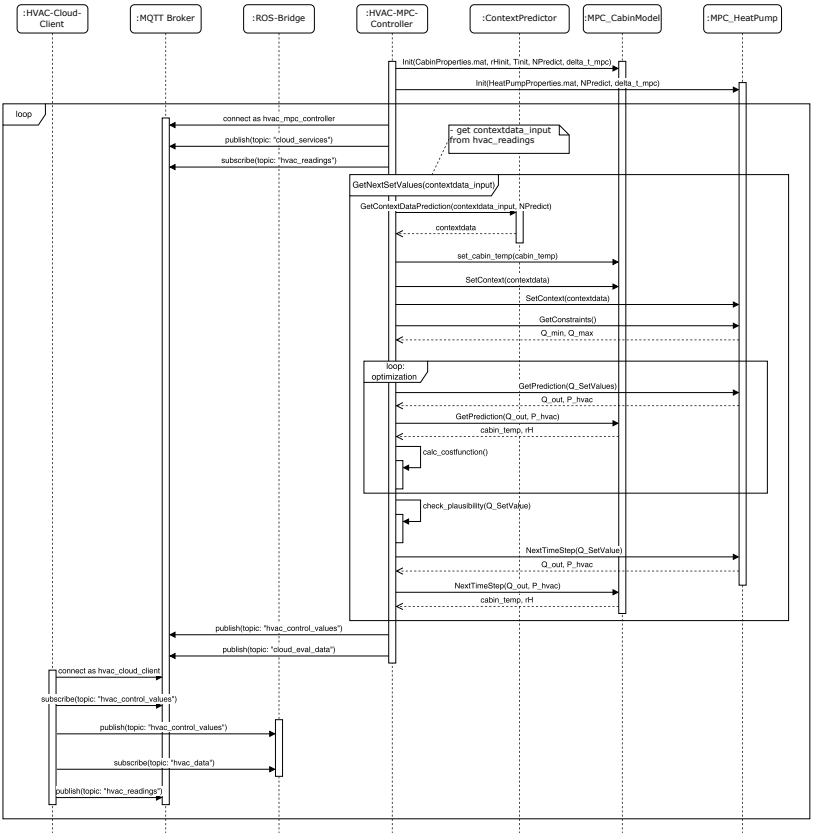


Abbildung A.17: Sequenzdiagramm der Ausführung des integrierten cloudbasierten MPC

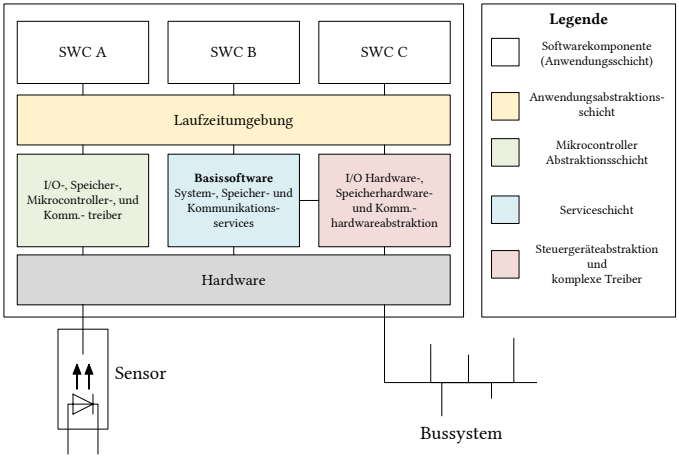


Abbildung A.18: Softwarekomponenten innerhalb der AUTOSAR Classic Schichtenarchitektur [112]

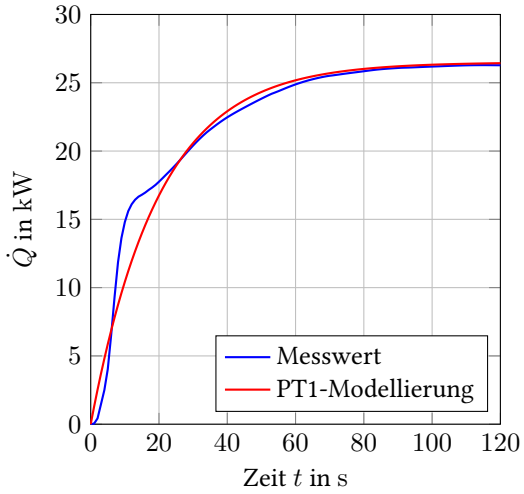


Abbildung A.19: Sprungantwort der modellierten Wärmepumpe [100]

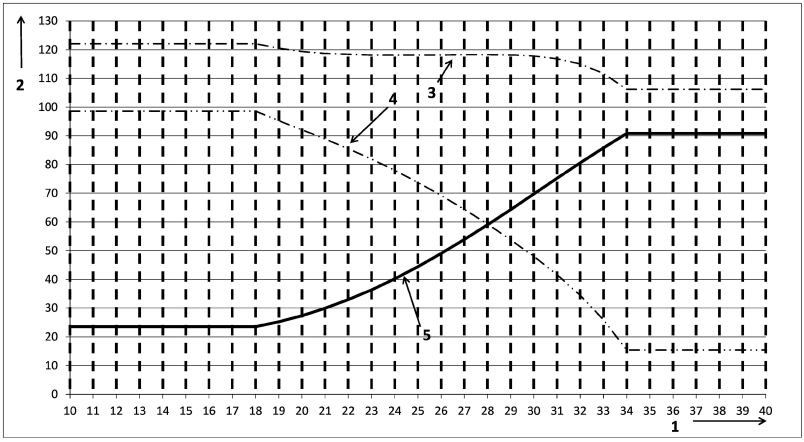


Abbildung A.20: Wärmeabgabe eines Menschen [34]

1: mittlere Raumtemperatur $T_{\mathrm{Mittel}}[℃]$

2: Wärmeabgabe [W]

3: Wärme gesamt [W]

4: sensible Wärme [W]

5: latente Wärme [W]

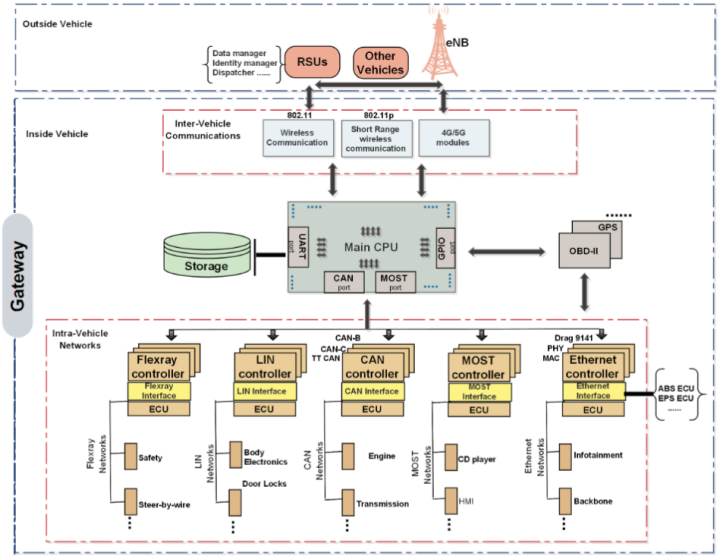


Abbildung A.21: Aufbau der fahrzeuginternen Telematikeinheit als Gateway [48]

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface 141, 145, *Glossar:* API

ASIC Application-specific integrated circuit 9, 182, *Glossar:* ASIC

ASIL Automotive Safety Integrity Level 92–94, 107, 109, 176, 203, 204

AUTOSAR AUTomotive Open System ARchitecture 30, 194, 196, 197, 200

AWS Amazon Web Services 69

BEB batterieelektrischer Bus 33, 34, 82, 107, 110, 113, 116, 138, 175

BPH Buffered Prediction Horizon 150, 161, 162

CAN Controller Area Network 25, 28, 70, 143, 194

CapEx Capital Expenditures 104, 134, 174, *Glossar:* CapEx

CBVF Cloud-based vehicle functions 61, 62

CC Cloud Computing 50

COP Coefficient of Performance 41, 116, 120, 156, 157, 180

COTA Control-Over-The-Air 5, 113, 114, 142, 160, 169, 173, *Glossar:* COTA

CPU Central Processing Unit 21–23, 99, 101, *Glossar:* CPU

CSP Cloud Service Provider 104, 111

DDS Data Distribution Service 30, 127, 128, 138, 189, 200

DSGVO Datenschutz-Grundverordnung 109

DSRC Dedicated Short-Range Communications 141

DVI Driver-Vehicle-Interface 70

E/E Elektrik/Elektronik 2, 5, 6, 10, 23, 26, 66, 87, 107, 128, 141, 143

ECU Electronic Control Unit 19–21, 23, 25–27, 50, 51, 61, 87, 133, 134, 137, 141, 142, 144, 158, 167, 182, 183, 196, 207, *Glossar:* ECU

FAP Fahrerarbeitsplatz 42, 48

FGR Fahrgastraum 42, 48, 49

FPGA Field-Programmable Gate Array 9, 182, *Glossar:* FPGA

GPU Graphics Processing Unit 23, 101

HARA Hazard analysis and risk assessment 203

HiL Hardware-in-the-Loop 72, 187, 188

HLK Heizung, Lüftung, Klimatisierung 8, 37, 38, 45, 46, 78–84, 87, 108–116, 118, 120, 136–139, 143, 148, 150, 154, 155, 159, 169, 174, 175, 182, 185

HPC High-Performance Computer 27, 87, 141, 144, *Glossar:* HPC

HTTP Hypertext Transfer Protocol 197

IoT Internet of Things 52, 61

IP Internet Protocol 28, 143

IT Informationstechnik 26, 50

KI künstliche Intelligenz 81, 131

LIN Local Interconnect Network 25, 194

LTE Long Term Evolution 53, 54, 56, 100

MCC Mobile Cloud Computing 76, 85, 86, *Glossar:* MCC

MCDA Multiple-Criteria Decision Analysis 58, 87, 131, 136, 139, 143, 174, *Glossar:* MCDA

MiL Model-in-the-Loop 186, 188

MIMO Multiple Input Multiple Output 79, 80

MPC Model Predictive Control 11, 13, 16, 80, 81, 117, 118, 123, 137–139, 145, 146, 150, 151, 155, 157–160, 162, 167, 170, 171, 174, 175

MQTT Message Queuing Telemetry Transport 128, 146, 162, 167, 169, 197

OEM Original Equipment Manufacturer 105, 174

OpEx Operating Expenses 104, 111, 134, 174, *Glossar:* OpEx

OSI Open Systems Interconnection 29, 201, 206

OTA Over-The-Air 4, 6, 52, 104, 128, 134, 168, *Glossar:* OTA

PDU Protocol Data Unit 28

PKW Personenkraftwagen 26, 38, 104, 141

PMV Predicted Mean Vote 180

PTC Positive Temperature Coefficient 40

QOS Quality of Service 146, 147, 197, *Glossar:* QOS

RL Reinforcement Learning 81, 82, 115, 118, *Glossar:* RL

ROS Robot Operating System 30, 31, 87, 127, 138, 189, 190, 192, 200, 220

RSU Road Side Unit 141

RTT Round-Trip Time 56, 129, 162, 167, 171

SiL Software-in-the-Loop 186, 188

SINR Signal-to-Interference-plus-Noise Ratio 56

SOA serviceorientierte Architektur 27–30, 85, 127, 128, 138, 168, *Glossar:* SOA

SORT Standardized on Road Test Cycles 82, *Glossar:* SORT

StVZO Straßenverkehrs-Zulassungs-Ordnung 33

SWC software component 29, 56, 63–66, 89, 90, 92, 130, 182

TCO Total Cost of Ownership 75, 87, 103, 171, 172, *Glossar:* TCO

TFLOPS Tera Floating Point Operations Per Second 50, 51

TLS Transport Layer Security 169, 197

UNECE United Nations Economic Commission for Europe 33

VDV Verband Deutscher Verkehrsunternehmen 42, 43, 82, 126, 218, *Glossar:* VDV

VM virtuelle Maschine 111

WSM Weighted Sum Model 58, 136

YAML YAML Ain't Markup Language 168, *Glossar:* YAML

zGM zulässige Gesamtmasse 32

ÖPNV öffentlicher Personennahverkehr 32

Glossar

API Programmierschnittstelle, die Funktionalitäten der zugrundeliegenden Implementierung abstrahiert und anderen Anwendungen zur Verfügung stellt.

ASIC Integrierte Schaltung, die für eine spezifische Anwendung oder Funktion entwickelt und hergestellt wird. Im Gegensatz zu universellen Prozessoren ist die Logik eines ASICs fest implementiert und nicht veränderbar, was zu hoher Effizienz bei spezialisierten Aufgaben führt.

CapEx CapEx (dt. Investitionsausgaben) bezeichnet Ausgaben für langlebige Vermögenswerte, die zur Erweiterung, Verbesserung oder Erhaltung der betrieblichen Infrastruktur eines Unternehmens oder einer Organisation dienen. Dazu zählen insbesondere Investitionen in physische Anlagen wie Gebäude, Maschinen, Fahrzeuge oder technische Infrastruktur.

COTA Ansatz bei dem regelnde Softwarekomponenten in der Cloud ausgeführt werden und somit eine Regelschleife vom Fahrzeug über die Cloud und zurück geschlossen wird.

CPU Die zentrale Verarbeitungseinheit eines digitalen Rechensystems. Sie stellt die wesentliche Recheneinheit eines Computers dar und ist ver-

antwortlich für die Ausführung von Befehlen, die Steuerung der Abläufe im System sowie die Verarbeitung von Daten.

ECU Auch Steuergerät genannt. Es handelt sich hierbei um ein beliebiges eingebettetes System im Fahrzeug, das Regelungs-, Steuerungs- oder Überwachungsaufgaben basierend auf Eingabewerten von Sensoren übernimmt.

Edge-Node Ein Edge-Node in Automobilanwendungen ist ein stationärer oder mobiler Rechenknoten, der in unmittelbarer Netzwerk- oder geografischer Nähe zu Fahrzeugen eingesetzt wird. Er bietet lokale Kommunikationsschnittstellen sowie begrenzte Rechen- und Speicherressourcen, um latenzkritische Kooperation und/oder die teilweise bzw. vollständige Auslagerung von Fahrzeugfunktionen zu ermöglichen.

FPGA Ein FPGA ist ein integrierter Schaltkreis, der nach seiner Herstellung beim Kunden mittels einer Hardware Description Language (HDL) programmiert werden kann. Daher die Bezeichnung „field programmable“.

HPC Ein Automotive High-Performance Computer bezeichnet einen zentralisierten Hochleistungsrechner innerhalb der Fahrzeugarchitektur, der Rechen- und Steuerungsaufgaben verschiedener Domänen integriert. Er stellt die notwendige Rechenleistung für Fahrerassistenzsysteme, Infotainment, Konnektivitätsdienste sowie automatisierte Fahrfunktionen bereit. Durch die Konsolidierung zuvor verteilter Steuergeräte gilt der Automotive HPC als Schlüsseltechnologie für die Entwicklung softwaredefinierter Fahrzeuge.

Matching Theory Matching Theory ist ein Teilgebiet der Spieltheorie und der algorithmischen Ökonomie, das sich mit der optimalen Zuordnung zweier Mengen von Akteuren oder Ressourcen befasst, beispielsweise

Studenten zu Universitäten oder Arbeitskräfte zu Stellenangeboten. Ziel ist es, stabile und effiziente Paarungen unter Berücksichtigung individueller Präferenzen, Kapazitäten und Rahmenbedingungen zu finden. Im Kontext cloudbasierter Fahrzeugsoftwarekomponenten ermöglicht Matching Theory die modellbasierte Zuweisung von Softwarekomponenten zu Fahrzeugen, wobei kontextuelle Anforderungen (z.B. Fahrzeugsensorik, Einsatzszenarien, Rechenressourcen) berücksichtigt werden können. Sie bietet damit eine methodische Grundlage für die dynamische, bedarfsgerechte und skalierbare Verteilung von Softwarefunktionen in vernetzten Fahrzeugflotten.

MCC Mobile Cloud Computing bezeichnet die Nutzung von Cloud-Computing-Diensten über mobile Endgeräte wie Smartphones oder Tablets. Dabei werden Rechenleistung, Speicher und Anwendungen nicht lokal auf dem Gerät, sondern in der Cloud bereitgestellt, um die begrenzten Ressourcen mobiler Geräte zu erweitern.

MCDA Die Multi-Criteria Decision Analysis (MCDA) bezeichnet eine Klasse methodischer Ansätze zur strukturierten Entscheidungsunterstützung in multidimensionalen Problemsituationen, bei denen mehrere, teils konkurrierende Zielkriterien simultan berücksichtigt werden müssen. MCDA zielt darauf ab, Entscheidungsprozesse durch formalisierte Bewertungs- und Aggregationsmechanismen transparenter, nachvollziehbarer und systematischer zu gestalten.

OpEx OpEx (dt. Betriebsausgaben) bezeichnet die laufenden Ausgaben, die im Rahmen des täglichen Geschäftsbetriebs anfallen. Dazu zählen Kosten für Wartung, Energieverbrauch, Personal, Dienstleistungen, Verbrauchsmaterialien oder Softwarelizenzen im Rahmen des operativen Betriebs.

OTA Over-the-Air bezeichnet die drahtlose Übertragung von Softwareaktualisierungen, Konfigurationen oder Daten an Endgeräte wie Smartphones, IoT-Geräte oder eingebettete Systeme. Diese Technologie er-

möglicht die Fernwartung und -aktualisierung von Gerätesoftware, einschließlich Betriebssystemen, Sicherheitspatches und Anwendungssoftware, ohne physischen Zugriff auf das Gerät.

QOS Einsatz von Mechanismen oder Technologien in einem Netzwerk, um den Datenverkehr zu kontrollieren und die Leistung wichtiger Anwendungen bei begrenzter Netzwerkkapazität sicherzustellen.

RL Reinforcement Learning ist ein Teilbereich des maschinellen Lernens, bei dem ein Agent durch Interaktion mit einer Umgebung lernt, optimale Handlungen auszuführen. Der Agent erhält dabei Rückmeldungen in Form von Belohnungen oder Strafen und passt sein Verhalten so an, dass langfristig die kumulative Belohnung maximiert wird. RL wird häufig in Bereichen wie Robotik, automatisiertes Fahren oder zur Erstellung von KI-Agenten für Spiele eingesetzt.

SOA Definiert eine Möglichkeit, Softwarekomponenten über Serviceschnittstellen wiederverwendbar und interoperabel zu machen. Services verwenden gemeinsame Schnittstellenstandards und ein Architekturmuster, sodass sie schnell in neue Anwendungen integriert werden können.

SORT Standardisierter, reproduzierbarer Fahrzyklus für den Vergleich des Energeiverbrauchs von Bussen.

TCO Bezeichnet die gesamten Kosten, die mit dem Erwerb, dem Betrieb und der Wartung eines Produkts oder Systems über dessen gesamte Lebensdauer verbunden sind. TCO umfasst nicht nur die anfänglichen Anschaffungskosten, sondern auch alle folgenden Betriebskosten, wie etwa Energieverbrauch, Wartungs- und Reparaturkosten, Betriebsressourcen (z.B. Personalaufwand) sowie Kosten für Ersatzteile und Software. Besonders in technischen und industriellen Bereichen ist TCO

eine wichtige Kennzahl, um die langfristige Rentabilität und Effizienz von Investitionen zu bewerten.

VDV Der Verband Deutscher Verkehrsunternehmen (VDV) ist der Branchenverband des öffentlichen Verkehrs, in dem über 630 Unternehmen des öffentlichen Personenverkehrs (ÖPV) und des Schienengüterverkehrs organisiert sind.

YAML YAML ist ein menschenlesbares Datenformat zur Konfiguration und zum Datenaustausc. Es wird häufig in Konfigurationsdateien (z.B. für Docker, Kubernetes, CI/CD-Pipelines) sowie für den Datenaustausch in Webanwendungen verwendet.

Literaturverzeichnis

[1] 5G Automotive Association e.V.: *Connected Mobility: C-V2X explained.* Technischer Bericht, 2025. https://5gaa.org/c-v2x-explained/.

[2] Abuhussain, H., Alqahtani, A., Alharthi, A. und Alshahrani, S.: *Adaptive HVAC System Based on Fuzzy Controller Approach.* Applied Sciences, 13(20):11354, 2023.

[3] Afram, Abdul und Janabi-Sharifi, Farrokh: *Theory and applications of HVAC control systems–A review of model predictive control (MPC).* Building and Environment, 72:343–355, 2014.

[4] Afram, Abdul, Janabi-Sharifi, Farrokh, Fung, Alan S. und Raahemifar, Kaamran: *Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system.* Energy and Buildings, 141:96 – 113, 2017, ISSN 0378-7788. http://www.sciencedirect.com/science/article/pii/S0378778816310799.

[5] Andrew Putrayudha, S., Kang, Eun Chul, Evgueniy, E., Libing, Y. und Lee, Euy Joon: *A study of photovoltaic/thermal (PVT)-ground source heat pump hybrid system by using fuzzy logic control.* Applied Thermal Engineering, 2015.

[6] Arestova, Anna, Martin, Maximilian, Hielscher, Kai Steffen Jens und German, Reinhard: *A service-oriented real-time communication scheme for AUTOSAR adaptive using OPC UA and time-sensitive networking.* Sensors, 21(7):2337, 2021.

[7] Arthurs, Peter, Gillam, Lee, Krause, Paul, Wang, Ning, Halder, Kaushik und Mouzakitis, Alexandros: *A Taxonomy and Survey of Edge Cloud Computing for Intelligent Transportation Systems and Connected Vehicles.* IEEE Transactions on Intelligent Transportation Systems, 23(7):6206–6221, 2022.

[8] Ashok, Ashwin, Steenkiste, Peter und Bai, Fan: *Enabling Vehicular Applications Using Cloud Services through Adaptive Computation Offloading.* In: *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, MCS '15, Seite 1–7, New York, NY, USA, 2015. Association for Computing Machinery, ISBN 9781450335454.

[9] Ashok, Ashwin, Steenkiste, Peter und Bai, Fan: *Vehicular Cloud Computing through Dynamic Computation Offloading.* Computer Communications, 120:125–137, 2018.

[10] AUTOSAR: *SOME/IP Service Discovery Protocol Sepcification*, 2017. `https://www.autosar.org/fileadmin/standards/R22-11/FO/AUTOSAR_PRS_SOMEIPServiceDiscoveryProtocol.pdf`, abgerufen am 03.03.2022.

[11] AUTOSAR: *Explanation of Adaptive Platform Design*, 2019. `https://www.autosar.org/fileadmin/standards/R24-11/AP/AUTOSAR_AP_EXP_PlatformDesign.pdf`, abgerufen am 10.02.2026.

[12] Bach, Johannes: *Methoden und Ansätze für die Entwicklung und den Test prädiktiver Fahrzeugregelungsfunktionen.* Dissertation, Karlsruher Institut für Technologie (KIT), 2018.

[13] Balbierer, Norbert: *Energiemanagement Ethernet-basierter Fahrzeugnetze.* Dissertation, TU Ilmenau, 2018.

[14] Banijamali, Ahmad, Kuvaja, Pasi, Oivo, Markku und Jamshidi, Pooyan: *Kuksa: Self-adaptive Microservices in Automotive Systems.* In: *International Conference on Product-Focused Software Process Improvement*, Seiten 367–384. Springer, 2020.

[15] Bareiß, Mario und Vorgerd, Daniel: *Thermomanagement für elektrisch angetriebene Stadtbusse.* ATZ - Automobiltechnische Zeitschrift, 121(2):52–55, 2019, ISSN 2192-8800. https://doi.org/10.1007/s35148-018-0227-9.

[16] Bayerische Motorenwerke (BMW) AG: *Real Time Traffic Information*, 2025. https://www.bmw.de/de/shop/ls/dp/Base_RTTIOffer_de.

[17] Behrooz, Farinaz, Mariun, Norman, Marhaban, Mohammad Hamiruce, Mohd Radzi, Mohd Amran und Ramli, Abdul Rahman: *Review of Control Techniques for HVAC Systems—Nonlinearity Approaches Based on Fuzzy Cognitive Maps.* Energies, 11(3), 2018, ISSN 1996-1073. https://www.mdpi.com/1996-1073/11/3/495.

[18] Belton, Valerie und Stewart, Theodor: *Multiple criteria decision analysis: an integrated approach.* Springer Science & Business Media, 2002.

[19] Bernstein, Philip A: *Middleware: a model for distributed system services.* Communications of the ACM, 39(2):86–98, 1996.

[20] Birkel, Harald, Bronnenberg, Peter, Krawinkel, Elmar, Roch, Robert und Walter, Karsten: *Klimatisierung von Linienbussen der Zulassungsklassen I (Stadtbus) und II (Überlandbus), für konventionell angetriebene Diesel- und Gasbusse als auch für Hybrid-, Brennstoffzellen- und Elektrobusse.* VDV-Schrift 236, November 2018.

[21] Blanco, David Fernández, Le Mouël, Frédéric, Lin, Trista und Escudié, Marie Pierre: *A comprehensive survey on Software as a Service (SaaS) transformation for the automotive systems.* IEEE Access, 11:73688–73753, 2023.

[22] Blanco, David Fernández, Le Mouël, Frédéric, Lin, Trista und Escudié, Marie Pierre: *A Comprehensive Survey on Software as a Service (SaaS) Transformation for the Automotive Systems.* IEEE Access, 11:73688–73753, 2023.

[23] Bormann, René, Fink, Philipp, Holzapfel, Helmut, Rammler, Stephan, Sauter-Servaes, Thomas, Tiemann, Heinrich, Waschke, Thomas und Weirauch, Boris: *The future of the German automotive industry*. Friedrich-Ebert-Stiftung Shanghai Representative Office, 2018.

[24] Braess, Hans Hermann und Seiffert, Ulrich: *Vieweg Handbuch Kraftfahrzeugtechnik (ATZ/MTZ Fachbuch)*. Springer Vieweg, ISBN 9783834810113. `http://dnb.d-nb.de`, 7. Auflage, 2013.

[25] Bucaioni, Alessio und Pelliccione, Patrizio: *Technical architectures for automotive systems*. In: *2020 IEEE International Conference on Software Architecture (ICSA)*, Seiten 46–57. IEEE, 2020.

[26] Buyya, Rajkumar, Yeo, Chee Shin, Venugopal, Srikumar, Broberg, James und Brandic, Ivona: *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation computer systems, 25(6):599–616, 2009.

[27] Camacho, Eduardo F und Bordons, Carlos: *Introduction to model predictive control*. In: *Model Predictive Control*, Seiten 1–11. Springer, 2007.

[28] Craggs, Ian: *MQTT Vs. HTTP for IoT*, 2022. `https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/`, besucht: 01.10.2024.

[29] Dallinger, Laurenz: *HTTP vs MQTT: Choose the Best Protocol for your IoT Project*. Cedalo, 12.10.2022. `https://cedalo.com/blog/http-vs-mqtt-for-iot/`, besucht: 01.10.2024.

[30] Das Europäische Parlament und der Rat der Europäischen Union: *Richtlinie 2001/85/EG über besondere Vorschriften für Fahrzeuge zur Personenbeförderung mit mehr als acht Sitzplätzen außer dem Fahrersitz und zur Änderung der Richtlinien 70/156/EWG und 97/27/EG*, 2001.

[31] Das Europäische Parlament und der Rat der Europäischen Union: *Richtlinie 2007/46/EG zur Schaffung eines Rahmens für die Genehmigung von Kraftfahrzeugen und Kraftfahrzeuganhängern sowie von Systemen,*

Bauteilen und selbstständigen technischen Einheiten für diese Fahrzeuge, 2007.

[32] Dehli, Martin, Doering, Ernst und Schedwill, Herbert: *Grundlagen der Technischen Thermodynamik: Für eine praxisorientierte Lehre.* Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 10. auflage 2023 Auflage, 2023, ISBN 9783658412500. https://link.springer.com/978-3-658-41250-0.

[33] Deng, Hsien Wen, Rahman, Mizanur, Chowdhury, Mashrur, Salek, M Sabbir und Shue, Mitch: *Commercial Cloud Computing for Connected Vehicle Applications in Transportation Cyberphysical Systems: A Case Study.* IEEE Intelligent Transportation Systems Magazine, 13(1):6–19, 2021.

[34] Deutsches Institut für Normung: *DIN EN 14750 - 2025-02 Bahnanwendungen – Luftbehandlung in Schienenfahrzeugen des städtischen, Vorort- und Regionalverkehrs – Behaglichkeitsparameter und Typprüfungen.* Norm, 2025.

[35] Dosovitskiy, Alexey, Ros, German, Codevilla, Felipe, Lopez, Antonio und Koltun, Vladlen: *CARLA: An Open Urban Driving Simulator.* In: *Proceedings of the 1st Annual Conference on Robot Learning*, Seiten 1–16, 2017.

[36] Eckstein, Julian, Lüke, Christopher, Brunstein, Frederik, Friedel, Patrick, Köhler, Ulrich und Trächtler, Ansgar: *A novel approach using model predictive control to enhance the range of electric vehicles.* Procedia Technology, 26:177–184, 2016.

[37] Elmokashfi, Ahmed, Zhou, Dong und Baltrünas, Džiugas: *Adding the Next Nine: An Investigation of Mobile Broadband Networks Availability.* In: *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, MobiCom '17, Seite 88–100, New York, NY, USA, 2017. Association for Computing Machinery, ISBN 9781450349161. https://doi.org/10.1145/3117811.3117842.

[38] European Telecommunications Standards Institute (ETSI): *Why do we need 5G?* https://www.etsi.org/technologies/5G?jjj=1649405459019, 29.07.2015. abgerufen am: 08.04.2022.

[39] EvoBus GmbH: *Der eCitaro. Technische Informationen.* https://www.mercedes-benz-bus.com/de_DE/models/ecitaro/facts/facts-ecitaro.pdf, abgerufen am 10.06.2020.

[40] Fanger, P. O.: *Thermal comfort. Analysis and applications in environmental engineering.* Copenhagen: Danish Technical Press., 1970.

[41] Free3D: *Stadtbus mit drei Türen 3D-Modell.* https://free3d.com/de/3d-model/city-bus-three-doors-1776.html, 2025. Online; abgerufen am: 03-Juli-2025.

[42] Freund, Svenne: *Modellbasierte prädiktive Regelung komplexer gebäudetechnischer Anlagen zur Optimierung der Energieeffizienz und des Komforts.* Dissertation, TU Hamburg, 2023.

[43] Fulton, Randall und Vandermolen, Roy: *Airborne Electronic Hardware Design Assurance: A Practitioner's Guide to RTCA/DO-254.* CRC Press, 2017.

[44] Girdling, Gunther und Weiss, Bettina: *Introduction to Microcontrollers.* Vienna University of Technology, Institute of Computer Engineering, 2007.

[45] Goeb, Andreas: *SOA und Softwarequalität.* Dissertation, Technische Universität München, 2013.

[46] Göhlich, Dietmar, Fay, Tu Anh, Jefferies, Dominic, Lauth, Enrico, Kunith, Alexander und Zhang, Xudong: *Design of urban electric bus systems.* Design Science, 4:1–28, 2018, ISSN 20534701.

[47] Grassi, Walter: *Heat pumps: fundamentals and applications.* Springer, 2017.

[48] Hbaieb, Amal, Rhaiem, Olfa Ben und Chaari, Lamia: *In-car Gateway Architecture for Intra and Inter-vehicular Networks.* In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, Seiten 1489–1494, 2018.

[49] He, H, Jia, H, Sun, C und Sun, F: *Stochastic Model Predictive Control of Air Conditioning System for Electric Vehicles: Sensitivity Study, Comparison, and Improvement.* IEEE Transactions on Industrial Informatics, 14(9):4179–4189, 2018, ISSN 1941-0050.

[50] He, Hongwen, Yan, Mei, Sun, Chao, Peng, Jiankun, Li, Menglin und Jia, Hui: *Predictive air-conditioner control for electric buses with passenger amount variation forecast.* Applied Energy, 227:249–261, 2018, ISSN 0306-2619. `https://www.sciencedirect.com/science/article/pii/S0306261917312084`, Transformative Innovations for a Sustainable Future – Part III.

[51] Heutschi, Roger: *Architekturmanagement und Servicedesign.* Serviceorientierte Architektur: Architekturprinzipien und Umsetzung in die Praxis, Seiten 119–181, 2007.

[52] Hillenbrand, Martin: *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen.* Dissertation, Karlsruher Institut für Technologie (KIT), 2012.

[53] Holzmann, Henning: *Anwendungsorientierte Übersicht kommerzieller Fahrzeug-Simulations-Systeme.* In: *Fahrdynamik-Regelung*, Seiten 93–116. Springer, 2006.

[54] Huang, Junxian, Qian, Feng, Gerber, Alexandre, Mao, Z Morley, Sen, Subhabrata und Spatscheck, Oliver: *A close examination of performance and power characteristics of 4G LTE networks.* In: *Proceedings of the 10th international conference on Mobile systems, applications, and services*, Seiten 225–238, 2012.

[55] IEEE: *IEEE Standard for Information Technology–Portable Operating System Interface (POSIX) Base Specifications, Issue 7.* Standard, 2018.

[56] International Organization for Standardization: *ISO/IEC 26550:2015 Software and systems engineering — Reference model for product line engineering and management.* Standard, 2015.

[57] International Organization for Standardization: *ISO/IEC/IEEE 24765:2017(E) International Standard - Systems and software engineering–Vocabulary.* Standard, 2017.

[58] International Organization for Standardization: *ISO/IEC/IEEE 26514:2022 Systems and software engineering — Design and development of information for users.* Standard, 2022.

[59] Isermann, R.: *Automotive Control - Modeling and Control of Vehicles.* Springer-Verlag GmbH, Berlin, 2022.

[60] Isermann, R., Schaffnit, J. und Sinsel, S.: *Hardware-in-the-loop simulation for the design and testing of engine-control systems.* Control Engineering Practice, 7(5):643–653, 1999, ISSN 0967-0661. https://www.sciencedirect.com/science/article/pii/S0967066198002056.

[61] Jefferies, Dominic, Tu-Anh Ly, Alexander Kunith und Göhlich, Dietmar: *Energiebedarf verschiedener Klimatisierungssysteme für Elektro-Linienbusse.* In: *Deutsche Kälte- und Klimatagung*, 2015.

[62] Kaup, Fabian, Gottschling, Philip und Hausheer, David: *PowerPi: Measuring and modeling the power consumption of the Raspberry Pi.* In: *39th Annual IEEE Conference on Local Computer Networks*, Seiten 236–243. IEEE, 2014.

[63] Khalid, W. *et al.*: *Fuzzy Energy Management Controller and Scheduler for Smart Homes.* Sustainable Cities and Society, 47:101475, 2019.

[64] Knote, T., Haufe, B. und Saroch, L.: *E-Bus-Standard: Ansätze zur Standardisierung und Zielkosten für Elektrobusse.* Fraunhofer-Institut für Verkehrs- und Infrastruktursysteme (IVI), 2017. https://books.google.de/books?id=SVLhuQEACAAJ.

[65] Kovachev, Dejan, Cao, Yiwei und Klamma, Ralf: *Mobile Cloud Computing: A Comparison of Application Models*, 2011. https://arxiv.org/abs/1107.4940.

[66] Kraftfahrt-Bundesamt: *Bestand an Kraftfahrzeugen und Kraftfahrzeuganhängern nach Fahrzeugalter, 1. Januar 2022 (FZ15).* https://www.kba.de/DE/Statistik/Produktkatalog/produkte/Fahrzeuge/fz15_b_uebersicht.html, 2024. Online; abgerufen am: 27-Februar-2024.

[67] Krafzig, Dirk, Banke, Karl und Slama, Dirk: *Enterprise SOA: Wege und Best Practices für serviceorientierte Architekturen;[Einführung, Umsetzung, Praxis; SOA-Definition, Architektur, Infrastruktur; Organisation, Strategie, Projektmanagement; vier konkrete Fallstudien].* MITP-Verlags GmbH & Co. KG, 2010.

[68] Kraus, David, Baumann, Daniel, Vučinić, Veljko und Sax, Eric: *Cloud-Enabled Reconfiguration of Electrical/Electronic Architectures for Modular Electric Vehicles.* World Electric Vehicle Journal, 16(2), 2025, ISSN 2032-6653. https://www.mdpi.com/2032-6653/16/2/111.

[69] Kreissl, Jochen: *Absicherung der SOME/IP Kommunikation bei Adaptive Autosar.* Diplomarbeit, Universität Stuttgart, 2017.

[70] Kreuzberger, Christoph: *Integration und Evaluierung eines 3-Ebenen Sicherheitskonzepts auf einer Echtzeit Mehrkern-Plattform.* Diplomarbeit, 2015.

[71] Kruppok, Kurt und Otten, Stefan: *Analyse der Energieverbraucher im batterieelektrischen Stadtbus.* Projektbericht in Kooperation mit Evo-Bus GmbH, 2019.

[72] Kugele, Stefan, Hettler, David und Shafaei, Sina: *Elastic Service Provision for Intelligent Vehicle Functions.* In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Seiten 3183–3190, 2018.

[73] Kugele, Stefan, Obergfell, Philipp und Sax, Eric: *Model-based resource analysis and synthesis of service-oriented automotive software architectures.* In: *Software and Systems Modeling (2021)*, March 2020. https://doi.org/10.1007/s10270-021-00896-9.

[74] Kumar, Karthik und Lu, Yung Hsiang: *Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?* Computer, 43(4):51–56, 2010.

[75] Kumari, Priti und Kaur, Parmeet: *A survey of fault tolerance in cloud computing.* Journal of King Saud University - Computer and Information Sciences, 33(10):1159–1176, 2021, ISSN 1319-1578. https://www.sciencedirect.com/science/article/pii/S1319157818306438.

[76] Li, Liqiang, Ma, Hao und Gao, Yong: *Reinforcement Learning-Based Control for Electric Bus HVAC Systems under Dynamic Conditions.* IEEE Transactions on Intelligent Transportation Systems, 24(2):1523–1534, 2023.

[77] Liu, Jieyao, Ahmed, Ejaz, Shiraz, Muhammad, Gani, Abdullah, Buyya, Rajkumar und Qureshi, Ahsan: *Application partitioning algorithms in mobile cloud computing: Taxonomy, review and future directions.* Journal of Network and Computer Applications, 48:99–117, 2015.

[78] Mazza, Daniela, Tarchi, Daniele und Corazza, Giovanni E.: *A partial offloading technique for wireless mobile cloud computing in smart cities.* In: *2014 European Conference on Networks and Communications (EuCNC)*, Seiten 1–5, 2014.

[79] Mell, Peter und Grance, Timothy: *The NIST Definition of Cloud Computing.* Recommendations of the National Institute of Standards and Technology NIST.–2011.–SP, Seiten 800–145, 2011.

[80] Meroth, Ansgar und Sora, Petre: *Sensornetzwerke in Theorie und Praxis.* Springer, 2018.

[81] Milani, Farzaneh: *Suitability Analysis Methodology for Cloud-based Vehicle Functions.* Dissertation, Technische Universität Darmstadt, 2020.

[82] Milani, Farzaneh und Beidl, Christian: *Cloud-based Vehicle Functions: Motivation, Use-cases and Classification.* In: *2018 IEEE Vehicular Networking Conference (VNC)*, Seiten 1–4, 2018.

[83] Milani, Farzaneh, Foell, Mike und Beidl, Christian: *A Data-based Approach to Predict the Response Time of Cloud-based Vehicle Functions.* In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, Seiten 1–6. IEEE, 2019.

[84] Montgomery, Ross und McDowall, Robert: *Fundamentals of HVAC control systems.* Elsevier, 2008.

[85] Nellessen, Philipp: *Vortriebssynchrone Prognose der Setzungen bei Flüssigkeitsschildvortrieben auf Basis der Auswertung der Betriebsdaten mit Hilfe eines Neuro-Fuzzy-Systems.* Cuvillier Verlag. 1. Auflage, 2005.

[86] NVIDIA: *Jetson AGX Orin Series.* `https://developer.nvidia.com/downloads/drive/secure/drive-orin/drive-agx-orin-devkit-qsg-du-11049-001.pdf`. Online; abgerufen am: 27.01.2025.

[87] NVIDIA: *Jetson AGX Orin Series (including AGX Orin Industrial) Thermal Design Guide.* `https://developer.nvidia.com/downloads/jetson-agx-orin-series-thermal-design-guide`. Online; abgerufen am: 27.01.2025.

[88] NXP: *AN13249 i.MX 8QuadMax Current Drain with Low- and High-Power Use Cases* . `https://www.nxp.com/docs/en/application-note/AN13249.pdf`. abgerufen am: 09.05.2023.

[89] Oldewurtel, Frauke, Sturzenegger, David und Morari, Manfred: *Importance of occupancy information for building climate control.* Applied Energy, 101:521–532, 2013.

[90] Open Group: *Initial Architecture Repository and the SOA Reference Architecture.* `https://www.opengroup.org/soa/source-book/togaf/p4.htm`, 2022. abgerufen am: 06.05.2022.

[91] Open Source Robotics Foundation: *ROS 2 Documentation: About Quality of Service settings.* `https://docs.ros.org/en/humble/Concepts/About-Quality-of-Service-Settings.html`. abgerufen am: 13.01.2023.

[92] Open Source Robotics Foundation: *ROS 2 Documentation: Internal API Architecture Overview.* `https://docs.ros.org/en/humble/Concepts/About-Internal-Interfaces.html`. abgerufen am: 11.01.2023.

[93] OPENSIGNAL: *Deutschland Erlebnisbericht zum Mobilfunknetz - November 2024 .* `https://www.opensignal.com/de/reports/2024/11/germany/mobile-network-experience`. abgerufen am: 22.01.2025.

[94] Pasha, M. *et al.*: *Comparison of Fuzzy and PID Techniques in Controlling a HVAC System.* International Journal of Engineering Research and Technology, 2(5):1–6, 2009.

[95] Patan, Krzysztof: *Robust and Fault-Tolerant Control.* In: *Robust and Fault-Tolerant Control*, Seiten 59–76. Springer, 2019.

[96] Protzmann, Robert, Huebner, Karl Karl, Ascheuer, Norbert, Bauknecht, Uwe, Enderle, Tobias, Gebhard, Ulrich, Raack, Christian und Witt, Arthur: *Large-scale modeling of future automotive data traffic towards the edge cloud.* In: *Photonic Networks; 20th ITG-Symposium*, Seiten 1–3. VDE, 2019.

[97] Ragesh, NK: *Data traffic in new generation vehicles.* CSI Communications, 35:12, 2012.

[98] Reinhardt, D. und Kucera, M.: *Domain Controlled Architecture.* In: *Domain Controlled Architecture - A New Approach for Large Scale Software Integrated Automotive Systems*, 2015.

[99] Rony, Rajib Uddin, Yang, Huojun, Krishnan, Sumathy und Song, Jongchul: *Recent Advances in Transcritical CO2 (R744) Heat Pump System: A Review.* Energies, 12(3), 2019, ISSN 1996-1073. `https://www.mdpi.com/1996-1073/12/3/457`.

[100] Rösch, Tobias: *Optimierung der Wärmebereitstellung in Thermomanagementsystemen elektrisch betriebener Stadtbusse.* Dissertation, Karlsruher Institut für Technologie (KIT), 2024.

[101] Sax, Eric (Herausgeber): *Automatisiertes Testen Eingebetteter Systeme in der Automobilindustrie.* Hanser eLibrary. Hanser, München and Wien, 2008, ISBN 9783446419018. http://www.hanser-elibrary.com/doi/book/10.3139/9783446419018.

[102] Sax, Eric: *Wagen, fahr schon mal den Harry vor.* FKFS Stuttgart, Auto-Test, 27.09.2018.

[103] Sax, Eric, Reussner, Ralf, Henle, Jacqueline, Otten, Stefan, Krach, Sebastian, Henß, Jörg, Hohl, Carl Philipp, Guissouma, Houssem und Saglam, Timur: *Themenpapier Cluster Elektromobilität Süd-West: Analyse der Aktivitäten und Entwicklungsfortschritte im Bereich der Fahrzeugelektronik mit Fokus auf fahrzeugeigene Betriebssysteme*, 2020.

[104] Schäuffele, Jörg und Zurawka, Thomas: *Automotive Software Engineering: Grundlagen, Prozesse, Methoden und Werkzeuge.* Springer, 2024.

[105] Schild, Kai: *Wärmeschutz: Grundlagen – Berechnung – Bewertung.* SpringerLink Bücher. Vieweg+Teubner Verlag / Springer Fachmedien Wiesbaden GmbH Wiesbaden, Wiesbaden, 2011, ISBN 9783834881458.

[106] Schindewolf, Marc, Grimm, Daniel, Lingor, Christian und Sax, Eric: *Toward a Resilient Automotive Service-Oriented Architecture by using Dynamic Orchestration.* In: *2022 IEEE 1st International Conference on Cognitive Mobility (CogMob)*, Seiten 147–154, 2022.

[107] Schindewolf, Marc, Stoll, Hannes, Guissouma, Houssem, Puder, Andreas, Sax, Eric, Vetter, Andreas, Rumez, Marcel und Henle, Jacqueline: *A Comparison of Architecture Paradigms for Dynamic Reconfigurable Automotive Networks.* In: *2022 International Conference on Connected Vehicle and Expo (ICCVE)*, Seiten 1–7, 2022.

[108] Scholz, Peter: *Echtzeit, Echtzeitsysteme, Echtzeitbetriebssysteme.* Softwareentwicklung eingebetteter Systeme: Grundlagen, Modellierung, Qualitätssicherung, Seiten 39–73, 2005.

[109] Schüppel, Fabian: *Optimierung des Heiz- und Klimakonzepts zur Reduktion der Wärme- und Kälteleistung im Fahrzeug.* Cuvillier Verlag, 2015, ISBN 9783736990296.

[110] Staron, Miroslaw: *Automotive software architectures - An Introduction.* Springer Cham, 2021.

[111] Stoll, Hannes Frank: *Die (re-)konfigurierbare Fahrzeugarchitektur.* Dissertation, Karlsruher Institut für Technologie (KIT), 2021.

[112] Streichert, Thilo und Traub, Matthias: *Elektrik/Elektronik-Architekturen im Kraftfahrzeug: Modellierung und Bewertung von Echtzeitsystemen.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, ISBN 978-3-642-25477-2.

[113] Sunyaev, Ali: *Internet Computing – Principles of Distributed Systems and Emerging Internet-Based Technologies.* Springer International Publishing, 1. ed. edition, 2020, ISBN 978-3-030-34956-1.

[114] Szyperski, Clemens, Gruntz, Dominik und Murer, Stephan: *Component software: beyond object-oriented programming.* Pearson Education, 2002.

[115] Taherdoost, Hamed und Madanchian, Mitra: *Multi-Criteria Decision Making (MCDM) Methods and Concepts.* Encyclopedia, 3(1):77–87, 2023, ISSN 2673-8392. `https://www.mdpi.com/2673-8392/3/1/6`.

[116] Tanyeri, Mustafa und Baslamisli, S.: *Prediction of the annual heat load of an articulated electric urban bus.* Journal of Thermal Sciences and Technology, 2019.

[117] Thorgeirsson, Adam: *Probabilistic Prediction of Energy Demand and Driving Range for Electric Vehicles with Federated Learning.* KIT Scientific Publishing, Karlsruhe, Sep 2024.

[118] Ulsoy, Galip, Peng, Huei und Çakmakci, Melih: *Automotive control systems.* Cambridge University Press, 2012.

[119] Umweltbundesamt: *EU-Richtlinie zu Emissionen aus Pkw-Klimaanlagen.* `https://www.umweltbundesamt.de/themen/klima-energie/fluorierte-treibhausgase-fckw/rechtliche-regelungen/eu-richtlinie-zu-emissionen-aus-pkw-klimaanlagen`. abgerufen am 04.07.2024.

[120] Union Internationale des Transports Publics: *UITP SORT & E-SORT brochures.* `https://www.uitp.org/publications/uitp-sort-e-sort-brochures/`. abgerufen am: 20.12.2022.

[121] Vatanparvar, K und Al Faruque, M A: *Design and analysis of battery-aware automotive climate control for electric vehicles.* ACM Transactions on Embedded Computing Systems, 17(4), 2018.

[122] Verband Deutscher Verkehrsunternehmen: *VDV-Schrift 230: Rahmenempfehlung für Stadt-Niederflur-Linienbusse*, Ausgabe 07/14.

[123] Verband Deutscher Verkehrsunternehmen: *VDV-Schrift 236: Klimatisierung von Linienbussen der Zulassungsklassen I (Stadtbus) und II (Überlandbus), für konventionell angetriebene Diesel-und Gasbusse als auch für Hybrid-, Brennstoffzellen- und Elektrobusse*, Ausgabe 11/2018.

[124] Vetter, Andreas: *Hierarchische Versionierung in der Entwicklung von Fahrzeugnetzwerken.* Dissertation, Karlsruher Institut für Technologie (KIT), 2025.

[125] Wang, Ziran, Liao, Xishun, Zhao, Xuanpeng, Han, Kyungtae, Tiwari, Prashant, Barth, Matthew J. und Wu, Guoyuan: *A Digital Twin Paradigm: Vehicle-to-Cloud Based Advanced Driver Assistance Systems.* In: *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, Seiten 1–6, 2020.

[126] Wei, Tianjun, Wang, Yanzhi und Zhu, Qing: *Deep Reinforcement Learning for Building HVAC Control.* In: *Proceedings of the 54th Annual Design Automation Conference (DAC).* ACM, 2017.

[127] Widmer, Fabio, Ritter, Andreas und Onder, Christopher H: *ZTBus: A large dataset of time-resolved city bus driving missions.* Scientific Data, 10(1):687, 2023.

[128] Wilhelm, Ulf, Ebel, Susanne und Weitzel, Alexander: *Funktionale Sicherheit und ISO 26262*, Seiten 85–103. Springer Fachmedien Wiesbaden, Wiesbaden, 2015, ISBN 978-3-658-05734-3. https://doi.org/10.1007/978-3-658-05734-3_6.

[129] Winner, Hermann, Hakuli, Stephan, Lotz, Felix und Singer, Christina: *Handbuch Fahrerassistenzsysteme*. Springer Fachmedien Wiesbaden, Wiesbaden, 2015, ISBN 978-3-658-05733-6.

[130] Wirtschaftskommission für Europa der Vereinten Nationen: *Regelung Nr. 107: Einheitliche Bestimmungen für die Genehmigung von Fahrzeugen der Klassen M2 oder M3 hinsichtlich ihrer allgemeinen Konstruktionsmerkmale*, 2018-02-23.

[131] Wu, Huaming: *Multi-objective decision-making for mobile cloud offloading: A survey*. IEEE Access, 6:3962–3976, 2018.

[132] Xu, Yuzhe: *Latency and bandwidth analysis of LTE for a smart grid*. Diplomarbeit, Königlich Technische Hochschule, Stockholm, 2011.

[133] Yan, Mei, He, Hongwen, Jia, Hui, Li, Menglin und Xue, Xue: *Model Predictive Control of the Air-conditioning System for Electric Bus*. Energy Procedia, 105:2415–2421, 2017, ISSN 1876-6102. https://www.sciencedirect.com/science/article/pii/S1876610217307579, 8th International Conference on Applied Energy, ICAE2016, 8-11 October 2016, Beijing, China.

[134] Yang, Chao, Li, Liang, You, Sixiong, Yan, Bingjie und Du, Xian: *Cloud computing-based energy optimization control framework for plug-in hybrid electric bus*. Energy, 125:11–26, 2017, ISSN 0360-5442. https://www.sciencedirect.com/science/article/pii/S0360544217302840.

[135] Zaheri, Dorsa, Niedballa, Dennis, Leuffen, Marc und Bilkei-Gorzo, Gergely: *Praktische Implementierung einer Zonenarchitektur für automatisierte Fahrzeuge in UNICARagil*. ATZelektronik, 18(1):16–21, 2023.

[136] Zimmer, Bastian und Oertel, Markus: *Mehr Leistung mit Autosar Adaptive.* ATZelektronik, 14(5):38–43, 2019, ISSN 1862-1791.

[137] Zimmermann, Werner und Schmidgall, Ralf: *Bussysteme in der Fahrzeugtechnik.* Springer, 2006.

Eigene Veröffentlichungen

[BtD+24] Baumann, Daniel ; **Sommer, Martin** ; Dettinger, Falk ; Rösch, Tobias ; Weyrich, Michael ; Sax, Eric: Connected Vehicle: Ontology, Taxonomy and Use Cases. In: *2024 IEEE International Systems Conference (SysCon)*, 2024, S. 1–6

[BtS+24] Baumann, Daniel ; **Sommer, Martin** ; Sax, Eric ; Dettinger, Falk ; Weyrich, Michael: Total Cost of Ownership: Cloud-based vs. Onboard Vehicle Software Components. In: *The 1st International Conference on Systems Scalability and Expandability*, International Academy, Research, and Industry Association (IARIA), 2024. – ISBN 978–1–68558–216–6, S. 6 S.

[BtSS22] Baumann, Daniel ; **Sommer, Martin** ; Schrempp, Yannick ; Sax, Eric: Use of Deep Learning Methods for People Counting in Public Transport. In: *2022 International Conference on Connected Vehicle and Expo (ICCVE)*, 2022, S. 1–6

[DWW+25] Dettinger, Falk ; Weiss, Matthias ; Weyrich, Michael ; Baumann, Daniel ; **Sommer, Martin**: Directives for Function Offloading in 5G Networks Based on a Performance Characteristics Analysis. In: *IEEE International Automated Vehicle Validation Conference*, 2025

[GLt+20] Grimm, Daniel ; Leiner, Simon ; **Sommer, Martin** ; Pistorius, Felix ; Sax, Eric: Flow-based aggregation of CAN frames with compressed payload. In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)* IEEE, 2020, S. 202–207

[LStS23] Lüntzel, Vitus ; Schade, Florian ; **Sommer, Martin** ; Sax, Eric: Modularized Platform for an Embedded Systems Case Study: Concept and Design. In: *Human Interaction & Emerging Technologies (IHIET 2023): Artificial Intelligence & Future Applications* 111 (2023), Nr. 111

[LtFS20] Lauber, Andreas ; **Sommer, Martin** ; Fuchs, Kevin ; Sax, Eric: Evolutionary Algorithms to Generate Test Cases for Safety and IT-Security in Automotive Systems. In: *2020 IEEE International Systems Conference (SysCon)* IEEE, 2020, S. 1–6

[RRt+23] Rösch, Tobias ; Raghuraman, Sunilkumar ; **Sommer, Martin** ; Junk, Carolin ; Baumann, Daniel ; Sax, Eric: Multi-layer Approach for Energy Consumption Optimization in Electric Buses. In: *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, Institute of Electrical and Electronics Engineers (IEEE), 2023 (IEEE Vehicular Technology Conference). – ISBN 979–83–503–1114–3, S. 1–6

[RtD+25] Ruhnau, Jan ; **Sommer, Martin** ; Dettinger, Falk ; Kraus, David ; Becker, Steffen ; Sax, Eric ; Weyrich, Michael: Vehicle Function Offloading: Finding the Optimal Cloud/Edge Application Model (OptiCAM). In: Arai, Kohei (Hrsg.): *Advances in Information and Communication.* Cham : Springer Nature Switzerland, 2025. – ISBN 978–3–031–84457–7, S. 556–571

[RtH+23] Ruhnau, Jan ; **Sommer, Martin** ; Henle, Jacqueline ; Walz, Alexander ; Becker, Steffen ; Sax, Eric: Ontology for Vehicle Function Distribution. In: *2023 IEEE International Systems Conference (SysCon), Vancouver, Canada, 17-20 April 2023*, Institute of Electrical and Electronics Engineers (IEEE), 2023. – ISBN 978–1–66543–994–7, S. 1–6

[RtS21] Rossel, Nicole ; **Sommer, Martin** ; Sax, Eric: Automated and networked city buses – Optimized, demand-oriented service through intelligent use of data. Version: 2021. `http:`

//dx.doi.org/10.51202/9783181023808-215. In: VDI Wissensforum GmbH (Hrsg.): *Commercial Vehicles 2021: Truck, Bus, Van, Trailor.* Düsseldorf : VDI Verlag, 2021. – DOI 10.51202/9783181023808–215. – ISBN 978–3–18–102380–8, S. 215–228

[RtS22] Rösch, Tobias ; **Sommer, Martin** ; Sax, Eric: Adaptive application development and integration process for modern automotive software. In: *ICCTA '22: Proceedings of the 2022 8th International Conference on Computer Technology Applications*, Association for Computing Machinery (ACM), 2022. – ISBN 978–1–4503–9622–6, S. 85–90

[SBS+20] Stock, Simon ; Bertemes, Alain ; Stang, Marco ; **Böhme, Martin** ; Grimm, Daniel ; Stork, Wilhelm: Feedi-a smart wearable foot-band for navigation and guidance using haptic feedback. In: *International Conference on Human Interaction and Emerging Technologies* Springer, 2020, S. 349–355

[StB+20] Stang, Marco ; **Sommer, Martin** ; Baumann, Daniel ; Zijia, Yuan ; Sax, Eric: Adaptive Customized Forward Collision Warning System Through Driver Monitoring. In: *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 2. Ed.: K. Arai* Bd. 1289, Springer International Publishing, 2020 (Advances in Intelligent Systems and Computing). – ISBN 978–3–030–63088–1, S. 757–772

[StKS24] Stang, Marco ; **Sommer, Martin** ; Kraus, David ; Sax, Eric: Improving the Validation of Automotive Self-Learning Systems through the Synergy of Scenario-Based Testing and Metamorphic Relations. In: *Proceedings of the IEEE/ACM 10th International Conference on Big Data Computing, Applications and Technologies.* New York, NY, USA : Association for Computing Machinery, 2024 (BDCAT '23). – ISBN 9798400704734

[StS19] Stang, Marco ; **Böhme, Martin** ; Sax, Eric: Applied Machine Learning: Reconstruction of Spectral Data for the Classifica-

tion of Oil-Quality Levels. In: *5th International Conference on Research in Engineering, Technology and Science (ICRETS 2019), Lissabon, P, February 3-7, 2019* Bd. 5, ISRES Publishing, 2019 (The Eurasia Proceedings of Science, Technology, Engineering & Mathematics (EPSTEM)). – ISSN 2602–3199, S. 1–13

[tBR+24] **Sommer, Martin** ; Baumann, Daniel ; Rösch, Tobias ; Dettinger, Falk ; Sax, Eric ; Weyrich, Michael: Process for the Identification of Vehicle Functions for Cloud Offloading. In: Arai, Kohei (Hrsg.): *Intelligent Computing*. Cham : Springer Nature Switzerland, 2024, S. 596–608

[tGSS25] **Sommer, Martin** ; Guissouma, Houssem ; Schindewolf, Marc ; Sax, Eric: An Orchestrator for the Dynamic Extension of Automotive E/E Architectures to the Cloud. In: Aiello, Marco (Hrsg.) ; Barzen, Johanna (Hrsg.) ; Dustdar, Schahram (Hrsg.) ; Leymann, Frank (Hrsg.): *Service-Oriented Computing*. Cham : Springer Nature Switzerland, 2025, S. 24–41

[tJRS21] **Sommer, Martin** ; Junk, Carolin ; Rösch, Tobias ; Sax, Eric: Intelligent Control of HVAC Systems in Electric Buses. In: *Human Interaction, Emerging Technologies and Future Applications IV : Proceedings of the 4th International Conference on Human Interaction and Emerging Technologies: Future Applications (IHIET – AI 2021), April 28-30, 2021, Strasbourg, France. Ed.: T. Ahram* Bd. 1378, Springer International Publishing, 2021 (Advances in Intelligent Systems and Computing). – ISBN 978–3–030–74009–2, S. 68–75

[tLS+20] **Böhme, Martin** ; Lauber, Andreas ; Stang, Marco ; Pan, Luyi ; Sax, Eric: Using Machine Learning to Optimize Energy Consumption of HVAC Systems in Vehicles. In: *Human Interaction and Emerging Technologies. Ed.: T. Ahram* Bd. 1018, Springer International Publishing, 2020 (Advances in Intelligent Systems and Computing). – ISBN 978–3–030–25629–6, S. 706–712

[tRS23] **Sommer, Martin** ; Rösch, Tobias ; Sax, Eric: Fleet data used for self-learning functions in commercial vehicles. In: *17th International Conference Commercial Vehicles 2023* Bd. 2417, VDI Verlag, 2023 (VDI-Berichte). – ISBN 978–3–18–092417–5, S. 81–91

[tSMS20] **Sommer, Martin** ; Stang, Marco ; Muetsch. Ferdinand ; Sax, Eric: TalkyCars: A Distributed Software Platform for Cooperative Perception among Connected Autonomous Vehicles based on Cellular-V2X Communication. In: *IEEE Intelligent Vehicles Symposium, October 19 - November 13, 2020, (Virtual) Las Vegas, NV*, 2020

[tSR21] **Sommer, Martin** ; Sax, Eric ; Rösch, Tobias: Model Predictive HVAC Control with disturbance variable forecasting for city buses. In: *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME) Mauritius, 7-8 Oct. 2021*, Institute of Electrical and Electronics Engineers (IEEE), 2021. – ISBN 978–1–6654–2943–6, S. 1–7

[tSS25] **Sommer, Martin** ; Seidel, Luca ; Sax, Eric: Control-over-the-air (COTA) for automotive comfort functions. In: *11th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2025

Betreute studentische Arbeiten

[Bau24] Bausch, Jens: *Auslegung und techno-ökonomische Analyse von Schnellladeparks unter Einbindung von automatisiertem Fahren und Laderobotern*, Karlsruher Institut für Technologie, Masterarbeit, 2024

[Bre23] Brenk, Joshua B.: *Entwurf und Implementierung eines Orchestrator Middleware Moduls innnerhalb der serviceorientierten Fahrzeugarchitektur*, Karlsruher Institut für Technologie, Bachelorarbeit, 2023

[Che21] Chen, Yalin: *Continuously Learning of Deep Neural Networks for Applications utilizing Incremental Data*, Karlsruher Institut für Technologie, Masterarbeit, 2021

[Eic20] Eichhorn, Nadine: *Modellierung eines Fahrgastraummodells innerhalb einer Modellprädiktiven Klimaregelung für einen Elektrobus*, Karlsruher Institut für Technologie, Bachelorarbeit, 2020

[Gan23] Ganz, Jan-Luca: *Process for identifying functions in a vehicle for possible relocation to the cloud*, Karlsruher Institut für Technologie, Bachelorarbeit, 2023

[Gu20] Gu, Renxin: *A Reinforcement-Learning Approach for Optimal Heating, Ventilation and Air-Conditioning Control*, Karlsruher Institut für Technologie, Masterarbeit, 2020

[Hen20] Hennhöfer, Aaron: *Entwicklung und Modellierung einer modellprädiktiven Klimaregelung für einen Elektrobus*, Karlsruher Institut für Technologie, Masterarbeit, 2020

[Het20] Hetzel, Moritz: *Knowledge Discovery in Databases: Entwicklung eines Algorithmus zur Berechnung von Passagieraufkommen*, Karlsruher Institut für Technologie, Bachelorarbeit, 2020

[Hor22] Hornstein, Justus: *Integration und Evaluation eines vernetzten HLK-Regelungssystems für einen Stadtbus in einer MiL-Umgebung*, Karlsruher Institut für Technologie, Bachelorarbeit, 2022

[Jun20] Junk, Carolin: *Vergleich von Reinforcement Learning und Modellprädiktiver Regelung zur Klimaregelung in Elektrobussen*, Karlsruher Institut für Technologie, Masterarbeit, 2020

[Jun23] Jung, Ulf: *Modellierung, Simulation und energetische Optimierung von Wärmequellen elektrisch betriebener Stadtbusse*, Karlsruher Institut für Technologie, Masterarbeit, 2023

[Koc24] Koch, Laurin: *Gesamtintegration eines „Control over the Air" Ansatzes für eine automotive Komfortfunktion*, Karlsruher Institut für Technologie, Masterarbeit, 2024

[Kuh23] Kuhn, Michael: *Erhöhung der Portabilität von Cloudsystemen mittels eines generalisierten Orchestrierungsinterfaces am Beispiel eines Fahrzeugbackends*, Karlsruher Institut für Technologie, Masterarbeit, 2023

[Li20] Li, Hui: *Predicting of Thermal Disturbance Variables to improve Thermal Management of Electric Vehicles*, Karlsruher Institut für Technologie, Masterarbeit, 2020

[Lin21] Linek, Simon: *Entwicklung einer auf neuronalen Netzen basierenden modellprädiktiven Klimaregelung für Stadtbusse*, Karlsruher Institut für Technologie, Masterarbeit, 2021

[Lom22] Lomen Árpád: *Modellierung der Regelstrecke einer Klimaregelung für einen Stadtbus an einem HiL Prüfstand*, Karlsruher Institut für Technologie, Masterarbeit, 2022

[Mit21] Mitsios, Dionysis: *Modellierung einer Stadtbus HLK-Funktion mithilfe von künstlichen neuronalen Netzen*, Karlsruher Institut für Technologie, Bachelorarbeit, 2021

[Mü20] Mütsch, Ferdinand: *TalkyCars: A Distributed Software Platform for Cooperative Perception among Connected Autonomous Vehicles based on Cellular-V2X Communication*, Karlsruher Institut für Technologie, Masterarbeit, 2020

[Osw23] Oswald, Kai: *Entwicklung eines Testkonzepts für serviceorientierte Architekturen im Automobilbereich*, Karlsruher Institut für Technologie, Bachelorarbeit, 2023

[Rau21] Rau, Robin: *Implementing a federated averaging algorithm for a neural network model predictive control of the HVAC system in city buses*, Karlsruher Institut für Technologie, Bachelorarbeit, 2021

[Rom21] Romier, Lucas: *Disturbance Forecasting for a Model Predictive Control of HVAC Systems*, Karlsruher Institut für Technologie, Bachelorarbeit, 2021

[Sch20] Schrempp, Yannick: *Einsatz von Deep Learning Methoden zur Personenzählung im öffentlichen Nahverkehr*, Karlsruher Institut für Technologie, Masterarbeit, 2020

[Sch22a] Schmid, Tobias: *Systemintegration und Bereitstellung eines modellprädiktiven Reglers im Kontext serviceorientierter Architekturen*, Karlsruher Institut für Technologie, Bachelorarbeit, 2022

[Sch22b] Schmidt, Sören: *Design eines Entscheidungsalgorithmus für die Verschiebung von cloudbasierten Funktionen in vernetzten Fahrzeugen*, Karlsruher Institut für Technologie, Bachelorarbeit, 2022

[Sto22] Stolbrink, Vincent: *Erweiterung eines HIL-Demonstrators um ein Closed-Loop-Fahrzeugmodell für das Testen cloudbasierter Funktionen*, Karlsruher Institut für Technologie, Bachelorarbeit, 2022