Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 1227

IT-Unterstützung von BHKW-Prüfständen in der angewandten Forschung

Dominik Schöner[1], Richard Pump[1], Christian Schmicke[2], Jan P. Minnrich[2], Henrik Rüscher[2], Volker Ahlers[1], Arne Koschel[1]

Abstract: In modernen Forschungsprüfständen spielt Software bei deren Überwachung und Steuerung sowie bei der Analyse von Messdaten eine große Rolle. Das dynamische Anforderungsumfeld und die interdisziplinäre Zusammenarbeit, sowie die Integration domänenspezifischer Werkzeuge und Komponenten stellen dabei hohe Ansprüche an Flexibilität, Wartbarkeit und Nutzerfreundlichkeit von Softwarearchitektur und Systemkomponenten.

Die in diesem Beitrag vorgestellte Architektur löst diese Herausforderungen mittels eines serviceorientierten Ansatzes und einer klaren Aufteilung in drei Schichten: Einer hardwarenahen Ebene zur Anbindung der Sensoren und Aktoren des Prüfstands, dem virtuellen LabVIEW-Leitstand als Client und der diese verbindenden Serviceinfrastruktur mit Geschäftslogik und Datenhaltung.

Keywords: Elektromobilität, Mobiles Mikro-BHKW, Prüfstand, Embedded, RTOS-UH, SOA, LabVIEW, Automatisierung, KWKK

1 Einleitung

Mit verschiedenen Initiativen soll die Bundesrepublik Deutschland zum „Leitmarkt und Leitanbieter für Elektromobilität“ [RB16] werden, steht jedoch vor Diskussionen um die nicht ausreichend vorhandene Ladeinfrastruktur und die vergleichsweise geringe Reichweite von Elektrofahrzeugen. Durch die Integration von Range Extendern (RE) in reinen Elektrofahrzeugen als zusätzliches, verbrennungsmotorisches Aggregat zur Ladung der Traktionsbatterie, kann die Reichweite von Elektrofahrzeugen deutlich erhöht werden. RE stellen eine Brückentechnologie dar, deren Einsatzschwerpunkt sich von der Unterstützung des Antriebs hin zur überwiegenden Nutzung im Thermo- und Energiemanagement des E-Fahrzeugs verlagert, um deren Energieeffizienz zu erhöhen und ihren Betrieb so ressourcenschonender und CO_2-effizienter zu gestalten. Zudem kann die Nachhaltigkeit von RE und vergleichbaren Konzepten durch den Einsatz von Kraftstoffen aus nachwachsenden Rohstoffen zusätzlich erhöht werden.

Bei der Weiterentwicklung von RE steht die thermische Konditionierung (Kühlen und Heizen) sowohl des Fahrgastinnenraums als auch der Batterie im Fokus, da dies nicht nur den Komfort für den Fahrgast erhöht, sondern verbessert durch Vorkonditionierung auch die Batteriereichweite, wie in Simulationen [Mi14] nachgewiesen. Neue Konzepte

[1] Hochschule Hannover, Fakultät IV, Abteilung Informatik, Ricklinger Stadtweg 120, 30459 Hannover, Dominik.Schoener@hs-hannover.de
[2] Hochschule Hannover, Fakultät II, Abteilung Maschinenbau, Bismarckstraße 2, 30173 Hannover

für verbrennungsmotorische RE setzen auf die Kraft-Wärme-Kopplung (KWK) und können daher auch als Mikro-Blockheizkraftwerk (Mikro-BHKW) bezeichnet werden.

Seit 2013 wird an der Hochschule Hannover der Ansatz verfolgt, eine Nutzung des REs als mobiles mikro-BHKW (mmBHKW) zur dezentralen KWK oder Kraft-Wärme-Kälte-Kopplung (KWKK), auch im Gebäude, zu ermöglichen [Rü14]. Durch die Bereitstellung von thermischer Energie zum Heizen oder Kühlen zusätzlich zum Drehmoment lässt sich so der Gesamtwirkungsgrad der im Kraftstoff gespeicherten Energie erhöhen [Mi15] und somit die CO_2-Bilanz verbessern. Die Skalierung des BHKWs zu kompakteren, mobil im Fahrzeug integrierten Aggregaten, die auch in anderen Bereichen (z. B. im Freizeitbereich oder der Gebäudetechnik) angewendet werden können, ist Teil des an den Hochschulen Hannover und Ostfalia ansässigen Forschungsschwerpunktes „Skalierbarkeit mobiler mikro-Blockheizkraftwerke“ [Sc14].

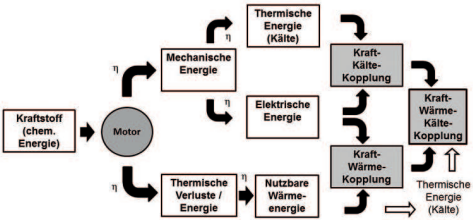

Abb. 1: Wirkungsgradketten für die Kennfelduntersuchung. Grafik: HS-Hannover

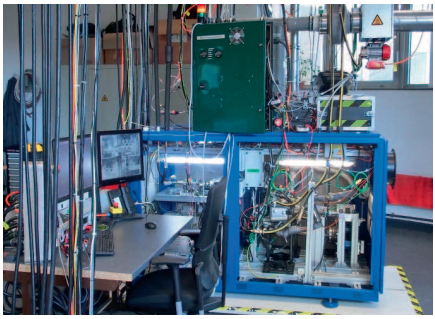

Abb. 2: 15kW Prüfstand an der Hochschule Hannover. Foto: HS-Hannover

Die an der Hochschule Hannover untersuchten mmBHKW-Konzepte grenzen sich dabei von üblichen, klassischen RE-Konzepten dadurch ab, dass ihre Hauptaufgabe in der Bereitstellung von Wärme- und Kälteströmen zur thermischen Konditionierung der Batterie und des Fahrgastraumes liegt. Um diese Technologie in Elektrofahrzeuge und Gebäude integrieren zu können, ist die Ermittlung relevanter Kennwerte (Abb. 1) notwendig, aus welchen sich Anforderungen hinsichtlich der Nutzung, Baugröße und Leistungsfähigkeit solcher Aggregate ergeben. Daraus abgeleitete Betriebsstrategien für mmBHKW-Konzepte in E-Fahrzeugen können dabei durch Steigerung der Energieeffizienz und des Komforts einen Beitrag zur Schonung von Ressourcen wie z.B. fossilen Brennstoffen leisten.

Im Rahmen des Forschungsschwerpunktes besteht hierzu eine enge Kooperation der Fachrichtungen Elektrotechnik, Maschinenbau und Informatik, wodurch es Studierenden dieser Fachrichtungen ermöglicht wird, sich in studentischen Projekten oder Abschlussarbeiten an der Bearbeitung von fachübergreifenden Aufgabenstellungen zu beteiligen.

Um die Machbarkeit dieses mmBHKW-Konzeptes zu überprüfen, wurden an der Hochschule Hannover zwei Versuchsstände im Leistungsbereich 1 – 15 kW (Abb. 2) aufgebaut. Mit Hilfe dieser können verschiedene Betriebsmodi, wie z. B. das Heizen mittels des Verbrennungsmotors bei gleichzeitigem Laden der Batterie oder das Laden der Batterie mit paralleler Kühlung des Innenraums, simuliert werden. Die Ergebnisse aus den Messun-

gen während verschiedener Versuche und Tests dienen hierbei u.a. der Einschätzung von Wirkungsgraden, Leistungen, Packaging und Gewicht [Ha16] für den Bau eines Prototyps.

In dieser Arbeit wird ein Konzept vorgestellt, mit dem die bisherige Aufzeichnung der Messwerte über einen Datenlogger und anschließender Offline-Analyse auf ein eingebettetes System umgestellt werden kann, um Messdaten im laufenden Betrieb analysieren und die Steuerung automatisieren zu können. Hierfür war es nötig, eine Systemarchitektur zu entwerfen, welche mit möglichst geringen Anpassungen dynamisch auf Veränderungen am Prüfstand reagieren kann und dennoch einen hohen Bedienkomfort bietet.

Im folgenden Abschnitt 2 werden die sich aus diesem Szenario ergebenden Anforderungen und das dafür entworfene Architekturkonzept vorgestellt. Die darauffolgenden Abschnitte gehen auf dessen Hauptkomponenten, das Mikrocontrollerboard (Abs. 3.1), die Servicearchitektur mit der Geschäftslogik (Abs. 3.2) und den Leitstand (Abs. 3.3), ein. Anschließend ordnet Abschnitt 4 diese Arbeit in den wissenschaftlichen Kontext ein, bevor Abschnitt 5 ein Fazit zu den erzielten Ergebnissen zieht und einen Ausblick auf die geplante Weiterführung und mögliche Weiterentwicklungen gibt.

2 Architekturkonzept

Der Prüfstandssoftware liegen umfangreiche funktionale Anforderungen und Rahmenbedingungen zugrunde, die einen großen Einfluss auf die Architektur haben. Insbesondere der Kontext des Forschungsprojektes beeinflusste Komponenten und Aufbau der Softwarearchitektur stark. Bei der Softwarearchitektur für den 15 kW Prüfstand steht vor allem die spätere Verwendung und Weiterentwicklung durch Mitarbeiter und Studierende aus den Fachbereichen Maschinenbau, Elektrotechnik und Informatik im Mittelpunkt.

Im Folgenden werden die Anforderungen an das System vorgestellt und mit einem Bezeichner (A1. . . An) versehen, mit dem sie im restlichen Dokument referenziert werden.

2.1 Anforderungen

Die Prüfstandssoftware soll die Untersuchungen am Prüfstand umfangreich unterstützen und dessen Anbindung an weiterverarbeitende Systeme vereinfachen (A1). Dazu muss die Software die Vielzahl an verschiedenen Messdaten des Prüfstandes visualisieren (A2) und die Steuerung der am Prüfstand vorhandenen Aktoren ermöglichen (A3). Durch einheitliche Schnittstellen (A4) soll die Software die Möglichkeit bieten, Steuer- und Regelprozesse mit Hilfe von weiteren Software-Komponenten zu automatisieren (A5), sowie die parallele Arbeit von mehreren Workstations erlauben (A6). Zur erneuten Auswertung muss die Software zudem die gefahrenen Versuche aufzeichnen und jederzeit zur weiteren Analyse wiedergeben können (A7), ohne dass der eigentliche Prüfstand benötigt wird. Um den sicheren Betrieb zu gewährleisten, muss die Software außerdem in der Lage sein, Über- oder Unterschreitungen von Grenzwerten zu erkennen (A8) und auf diese entsprechend, z.B. durch eine Warnmeldung an den Prüfstandsbediener, reagieren können.

Ferner muss das System auch ohne weitreichende Informatikkenntnisse leicht und verständlich bedienbar sein (A9), weshalb Werkzeuge verwendet werden sollen, die bereits in den entsprechenden Domänen genutzt werden (A10). Dabei muss es jedoch auch so flexibel und anpassbar sein, dass Änderungen an der Konfiguration des Prüfstandes, z.B. durch den Umbau von Sensoren, ohne großen Aufwand eingepflegt werden können (A11).

Da der Prüfstand im Rahmen eines bis 2018 angelegten Forschungsprojektes betrieben wird, ist außerdem zu gewährleisten, dass eine Erweiterung des Funktionsumfangs der Software ohne Beeinträchtigung der laufenden Systeme möglich ist, was eine starke Kapselung und lose Kopplung der einzelnen Komponenten voraussetzt (A12). Zusätzliche Komponenten zur Bilanzierung von Regelkreisen (A13), Aufzeichnung und Wiedergabe von Testläufen oder Automatisierung der Prüfstandssteuerung können so Schritt für Schritt eingebaut oder ausgetauscht werden.

2.2 Resultierende Architektur

Um die teils analogen, teils digitalen Sensor- und Steuersignale von der Hardwareebene in eine übersichtliche, schematische Visualisierung des Prüfstands zu überführen, bietet sich unter den gegebenen Voraussetzungen eine Aufteilung des Systems auf drei Kernkomponenten (Abb. 3) an [Du08]. Eine Oberfläche in LabVIEW bietet ein dem Anwender bekanntes Frontend (vgl. Anforderung A2), ein Application Server stellt eine Serviceinfrastruktur mit Geschäftslogik (vgl. A12) und ein Mikrocontrollerboard dient als hardwareseitiges Backend (vgl. A1). Der Application Server und das LabVIEW-Plugin entlasten zudem den Microcontroller, sodass problemlos komplexe Berechnungen zur Steuerung und Regelungen möglich sind, ohne kritische Abläufe auf dem Board zu beeinträchtigen.

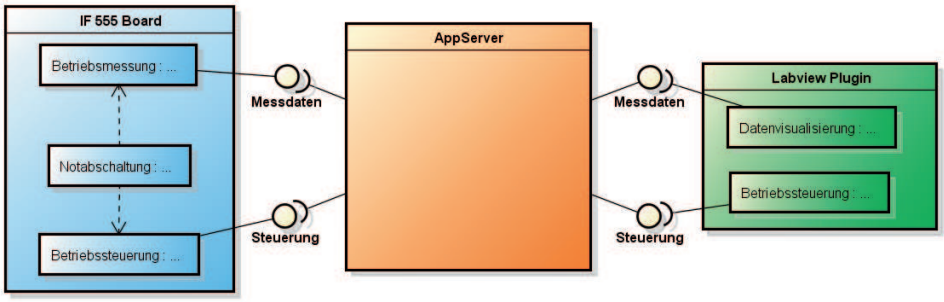

Abb. 3: Die drei Hauptkomponenten des Architekturkonzepts.

Ein IF-555-Board [Ha12] der Firma IEP aus Langenhagen, welches direkt am Prüfstand installiert wurde, bildet die Grundlage des Backends. Auf diesem befindet sich eine Motorola MPC563-CPU, auf welcher das Echtzeitbetriebssystem RTOS-UH [Ge06] (vgl. A10) läuft, wodurch Softwaremodule nebenläufig ausgeführt werden können. Dies spiegelt sich im Design und der Implementierung der Boardkomponenten (siehe Abschnitt 3.1) wieder. An das Board sind mehrere I/O-Module über CAN-Bus angeschlossen, um die Anzahl verfügbarer Sensor- und Aktoranschlüsse signifikant zu erhöhen.

Vom Board werden zwei Interfaces über TCP/IP zur Verfügung gestellt: Zum einen die Messdaten der über 100, an verschiedenen Stellen angeschlossenen, Sensoren (z.B. PT-100-Temperaturfühler und Drehzahlmesser) und zum anderen die Steuerung von mehr als 50 möglichen Aktoren (z.B. Servomotoren und Ventile) (vgl. A3). Dadurch ist eine komplette Überwachung des Prüfstandes und Automatisierung durch eine externe Regelungskomponente möglich (vgl. A1). Die Interfaces bieten dabei wenig Abstraktion; so werden Sensordaten z.B. als Rohdaten (nicht in SI-Basiseinheiten überführt) anhand der Anschlussstelle aggregiert übertragen. Um später Sensoren und Aktoren beliebig hinzufügen oder entfernen zu können (vgl. A11), spielt es bei den Interfaces des Boards keine Rolle, ob tatsächlich ein Sensor oder Aktor angeschlossen ist. Somit bilden die Interfaces die Anschlussmöglichkeiten des IF-555-Boards [Ha12] und dessen Erweiterungen ab.

Die TCP/IP Interfaces des Boards werden durch einen Application Server genutzt, welcher die Schnittstelle des Prüfstands zu sämtlichen anderen Systemen, wie z.B. dem in LabVIEW realisierten Leitstand, darstellt. Services auf dem Application Server nehmen hierbei u.a. die Rolle eines Adapters ein, welcher zum einen eine stärker abstrahierte und standardisierte RESTful API zum Zugriff auf Sensoren und Aktoren des Prüfstands bietet (vgl. A4) und zum anderen auch den parallelen Zugriff mehrerer Clients auf diesen ermöglicht (vgl. A6), ohne dass die begrenzten Ressourcen des Boards hierdurch zusätzlich beansprucht werden. Des Weiteren bietet der Server zusätzliche Services an, welche Nutzern z.B. erlauben, die Konfiguration für Sensoren und Aktoren über ein leicht zu bedienendes Webinterface zu bearbeiten (vgl. A9) und zuvor in einer Datenbank aufgezeichnete Testläufe erneut wiederzugeben (vgl. A7). Die serviceorientierte Architektur der einzelnen, in Form von Microservices implementierten, Softwarekomponenten auf dem Application Server bietet dabei eine hohe Flexibilität (vgl. A12), sowohl was fachliche Anforderungen und die Integration mit anderen Systemen [St15] als auch die Skalierbarkeit des Systems und dessen Portierbarkeit anbelangt. Dies wird in Abschnitt 3.2 näher betrachtet.

Als virtuelle Bedienoberfläche des Prüfstands dient schließlich ein in LabVIEW (vgl. A10) realisierter Leitstand, welcher die aufbereiteten Sensordaten über die Interfaces des Application Servers bezieht und diese auch nutzt, um Steuerbefehle an die Aktoren zu übermitteln (vgl. A3). Die zentrale Ansicht des Leitstands stellt dabei die Visualisierung sämtlicher Komponenten des Prüfstands in Form eines Fließbildes, wie in Abbildung 8 zu sehen, dar (vgl. A2), welches in enger Zusammenarbeit mit der Abteilung Maschinenbau erstellt wurde. Mit der Entscheidung, für diesen Teil des Systems auf LabVIEW zurückzugreifen, wurde der Grundstein für ein nutzerfreundliches Frontend gelegt, welches sowohl von der Zielgruppe an Nutzern aus Maschinenbau und Elektrotechnik als auch den Entwicklern aus dem Bereich Informatik ohne große Einstiegshürden genutzt, gewartet und erweitert werden kann (vgl. A9). Parallel zu diesem in Abschnitt 3.3 näher beschrieben Leitstand, wird im Rahmen einer Masterarbeit im Bereich Elektrotechnik eine ebenfalls in LabVIEW umgesetzte automatisierte Steuerung für den Prüfstand erarbeitet (vgl. A5), welche auf dieselben Schnittstellen zurückgreift.

3 Umsetzung

3.1 Hardwarenahes Backend

Das Backend der Prüfstandsoftware wurde in enger Zusammenarbeit mit Endanwendern des Systems entwickelt, wodurch der rege Austausch domänenspezifischer Informationen gefördert wurde. Dementsprechend bilden die Anforderungen an die Software des IF-555-Boards (Abb. 4) die zur Prüfstandoperation grundlegend notwendigen Operationen ab.

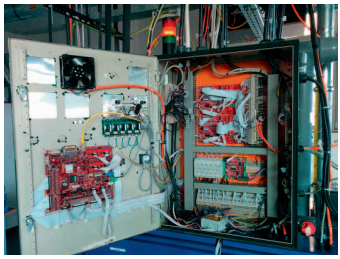

Abb. 4: Montiertes IF-555 Board mit Zusatzmodulen am Prüfstand.

Neben den bereits genannten funktionalen Anforderungen existieren aufgrund der Projektstruktur weitere Qualitätsanforderungen, welche Einfluss auf das Architekturdesign und die Implementierung haben. Die komplette Boardsoftware muss, wie bereits erwähnt, einfach veränderbar und erweiterbar sein um Studierenden ohne große Einarbeitungszeit die Arbeit an der Software zu erlauben (vgl. A12). Durch das IF-555 Board besteht zudem die Rahmenbedingung der Implementierung in Ansi-C-89 unter der Verwendung von speziellen Bibliotheken. Auch die gute Unterstützung von nebenläufigen Tasks durch das Betriebssystem RTOS-UH [Ge06] (vgl. A10) soll im Architekturdesign genutzt werden, um Module – ähnlich zu Klassen in Java – semantisch orientiert zu entwerfen (vgl. A12).

Anhand der Anforderungen an die Software und die Architektur entstand ein modulares Design, in welchem die Komponenten wenig Abhängigkeiten untereinander besitzen und die Funktionalität über wenige, die anderen Komponenten verwendenden, Programme erfüllt wird. Abbildung 5 zeigt die Komponenten des Mess-Steuer-Regelsystems (MSR).

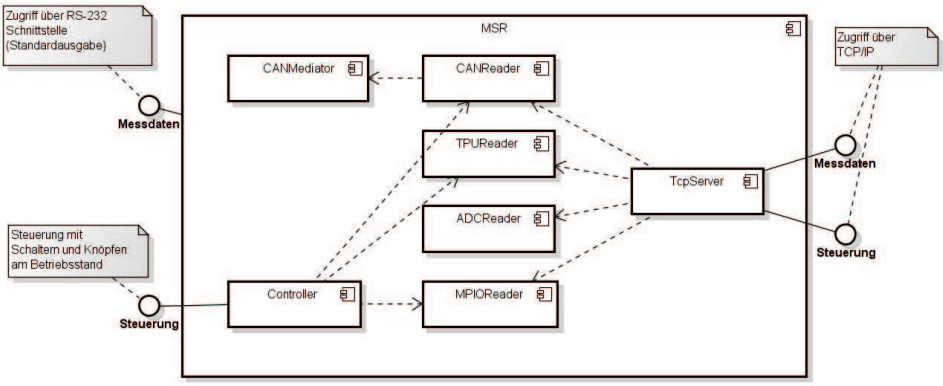

Abb. 5: Komponenten der Boardsoftware.

Die Grundlage des MSRs bilden die verschiedenen *Reader*, welche anhand der Ein-/Ausgabestellen des IF-555 Boards gruppiert sind. Die Reader enthalten nebenläufige Tasks, welche die Sensordaten der abstrahierten Ein-/Ausgabestelle zur Verfügung stellen (vgl.

A2). Weiterhin bieten sie Funktionen zur Ansteuerung eventueller Ausgänge (vgl. A3). Die Reader verwenden für den nebenläufigen Betrieb die umfangreichen Mittel des RTOS-UH [Ge06], um die Stärken des Betriebssystems auszunutzen.

Der TCPServer greift auf alle Reader zu, um deren Mess- und Steuerfunktionen über eine FTP-ähnliche Schnittstelle zur Verfügung zu stellen (vgl. A1). Der Server bietet softwaretechnische Interfaces für Messdaten und Steuerung, welche vom AppServer, aber auch anderen Anwendungen genutzt werden können (vgl. A5).

Zur Steuerung des Prüfstandes über ein Kontrollpanel wird das physikalische Interface des Prüfstandes durch den Controller bereitstellt. Der Controller besteht aus einem Eventhandler, um auf Signale des Panels zu reagieren und entsprechende Steuersignale zu senden.

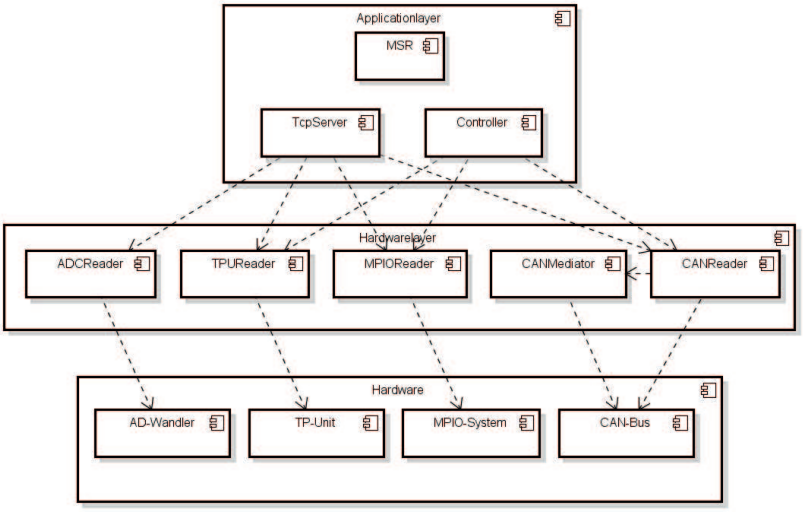

Abb. 6: Schichtarchitektur des Systems.

Durch die lose Kopplung und den schichtartigen Aufbau (Abb. 6) der Systemarchitektur sind Änderungen am System einfach durchzuführen und nachzuvollziehen (vgl. A12), wodurch eine zukünftige Weiterentwicklung durch Studierende möglich ist und das Testen der Komponenten erleichtert wird.

3.2 Servicearchitektur

Durch seine serviceorientierte Architektur bietet der Application Server (vgl. A10) einen hohen Grad an Modularität und Flexibilität, da sämtliche Services untereinander und nach außen über RESTful APIs (vgl. A4) kommunizieren (Abb. 7). Die Anbindung von Komponenten mit anderen, ggf. nicht standardisierten Protokollen kann dabei leicht in eigene Adapter gekapselt und über vereinheitlichte Schnittstellen verfügbar gemacht werden.

Der eng gefasste Aufgabenbereich und die dadurch kompakt gehaltene Codebasis der einzelnen Microservices ermöglichen es hierbei, mit geringem Aufwand auf Änderungen z.B.

am Kommunikationsprotokoll mit dem Board oder den Anforderungen an die Aufbereitung von Sensordaten zu reagieren [Ne15] (vgl. A12). Ein derart hoher Grad an Adaptivität innerhalb der Anwendungssoftware ist besonders in einem Forschungsszenario wie dem hier zugrunde liegenden Mikro-BHKW Prüfstand wichtig, um mit dem stetigen Wandel der Anforderungen, welche sich aus den verschiedenen Arbeiten am Prüfstand durch verschiedene Nutzer ergeben, Schritt zu halten.

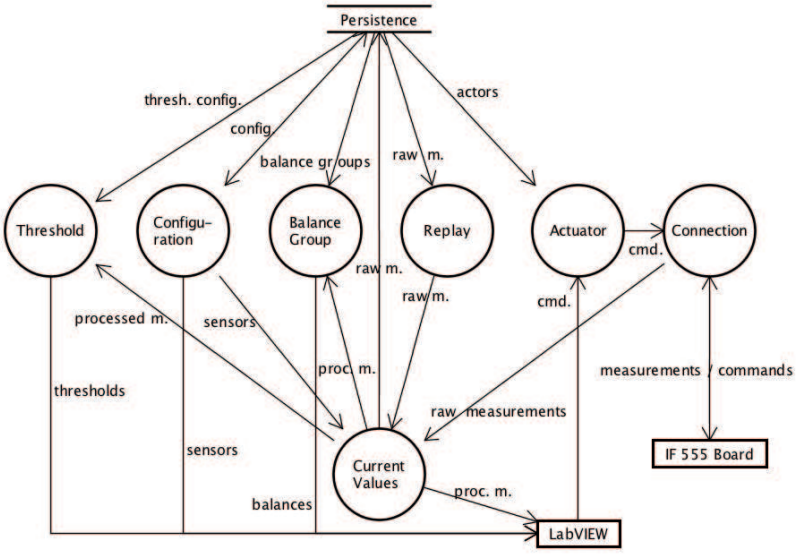

Abb. 7: Datenflüsse zwischen den einzelnen Services des Application Servers.

3.2.1 Basisdienste

Für dieses Szenario wurde ein Satz an Basisdiensten erarbeitet, der die Infrastruktur für sämtliche weiteren Dienste bildet und aus zwei zentralen Komponenten besteht:

- Der `Connection` Service stellt die Schnittstelle zwischen der Servicearchitektur des Application Servers und der Software auf dem Board dar und ist für die Verwaltung der Verbindungen zwischen beiden Systemen zuständig (vgl. A4). Er kann mehrere Verbindungen zu verschiedenen (ggf. virtuellen) Boards handhaben und sowohl deren Daten anderen Services zur Verfügung stellen als auch Steuerbefehle von diesen an die Boards weiterleiten (vgl. A1).
- Die aktuellen Rohdaten von Sensoren der jeweiligen Prüfstände werden vom Service `CurrentValues` in ggf. aufbereiteter Form anderen Services oder Clients zur Verfügung gestellt (vgl. A6).

Ergänzt werden die zentralen Services um einen generischen und agnostischen [Er14] `Persistence` Service, der u.a. zur Speicherung der Messwerte (vgl. A7) und zugehörigen Metadaten wie SI-Einheiten genutzt wird.

3.2.2 Erweiterte Funktionalitäten

Weitere Services erweitern den Funktionsumfang, wie u.a. der `BalanceGroup` Service zur Berechnung von Energie- oder Stoffbilanzen (vgl. A13) im Prüfstand. Um Störungen frühzeitig erkennen zu können, überwacht der `Threshold` Service kontinuierlich die Messwerte von `CurrentValues` auf die Über-/Unterschreitung von Messwerten (vgl. A8).

In der Datenbank aufgezeichnete Testläufe können mit Hilfe des `Replay` Service wiedergegeben werden (vgl. A7), um diese im Nachgang ausführlicher analysieren zu können. Durch die Speicherung von Rohdaten können dabei auch neue Bilanzkreise eingeführt oder bereits vorhandene Formeln korrigiert werden. Zudem kann direkt zu relevanten Zeitpunkten gesprungen werden, ohne z.B. die Aufwärmphase abwarten zu müssen.

Die über ein Webinterface änderbaren (vgl. A9) Sensorkonfigurationen werden versioniert durch den `Configuration` Service verwaltet (vgl. A11), wodurch diese auch für aufgezeichnete Testläufen verfügbar bleiben. Als Rückkanal für Steuerbefehle zur Regelung des Prüfstands mittels der Aktoren dient der `Actuator` Service (vgl. A3), welcher diese auf ihre Plausibilität prüfen kann. Hierdurch kann der Prüfstand von beliebigen Clients oder Services z.B. bei Grenzwertüberschreitungen automatisiert gesteuert werden (vgl. A5).

3.3 Leitstand

Zur Überwachung und (teilweise manuellen) Steuerung des Prüfstands wurde in enger Zusammenarbeit mit den Anwendern aus Maschinenbau und Elektrotechnik ein virtueller Leitstand in LabVIEW entwickelt, welcher die bisher genutzten Messinstrumente und Bedienelemente weitestgehend ersetzt (vgl. A2). Die Anbindung der LabVIEW-Oberfläche an den Prüfstand erfolgt dabei mittels HTTP-Aufrufen an die entsprechenden RESTful APIs der Services auf dem Application Server (vgl. A4).

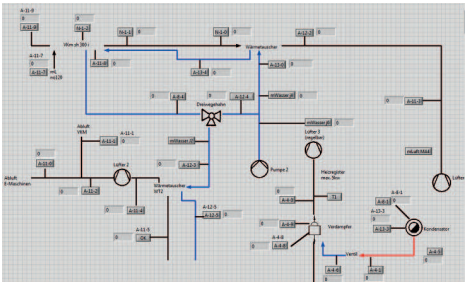

Abb. 8: Fließbild des Prüfstands in LabVIEW

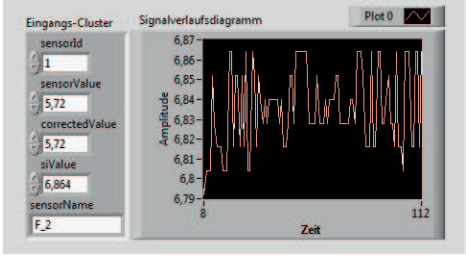

Abb. 9: Detailansicht eines Sensors

Nach dem Verbindungsaufbau werden die über `CurrentValues` ausgelesenen Messwerte sowie die Bilanzen von `BalanceGroup` (vgl. A13) in einem schematischen Fließbild des Prüfstands (Abb. 8) dargestellt (vgl. A10) und in einem konfigurierbaren Intervall aktualisiert. LabVIEW zeichnet dabei zusätzlich den Verlauf der Messwerte der einzelnen Sensoren auf, sodass dieser in der Detailansicht (Abb. 9) als Plot dargestellt werden kann. Zusätzlich werden in dieser Ansicht auch die rohen, die korrigierten und die in die ent-

sprechenden SI-Einheiten umgerechneten Sensorwerte angezeigt, um mögliche Fehler in den Korrektur- und Umrechnungsformeln erkennen zu können.

Weitere Bedienelemente im Fließbild erlauben zudem die Ansteuerung der Aktoren (vgl. A3), welche im Hintergrund über den `Actuator` Service läuft und so bei Bedarf die Regelung verschiedener Teilsysteme des Prüfstands von mehreren, verschiedenen Workstations aus ermöglicht (vgl. A6). Warnmeldungen zu von `Threshold` erkannten Grenzwertüberschreitungen werden ebenfalls in dieser Oberfläche angezeigt (vgl. A8), sodass auf diese umgehend reagiert werden kann.

Bei Änderungen (z.B. der Sensorkonfiguration) müssen diese manuell in LabVIEW nachgezogen werden (vgl. A11), was durch wiederverwendbare Bausteine mit geringem Aufwand möglich ist. Dies erlaubt die Wartung des Frontends und die Verwaltung der Sensor-, Aktor- und Bilanzkonfigurationen ohne weitreichende Programmierkenntnisse (vgl. A9).

4 Verwandte Arbeiten

In [Rü14] werden die grundlegenden Konzepte und Ideen zur Skalierbarkeit und Anwendung von mmBHKW in Fahrzeugen vorgestellt, welche die Basis der Forschungsarbeiten rund um die Prüfstände der Hochschule Hannover bilden, für welche die hier präsentierte Softwarearchitektur ausgelegt wurde. Ein Werkzeug zur Simulation des Einsatzes von mmBHKW in konkreten Szenarien hinsichtlich dessen Auswirkung auf die CO_2-Bilanz wurde vorgestellt [RSG15].

Ein ähnliches Konzept zu dem des mmBHKWs wurde in [Bo12] dargelegt, wobei hier lediglich die KWK ohne die zusätzliche Nutzung des Aggregats zur Kühlung betrachtet wird und dieses auch fest im Fahrzeug verbaut ist.

[PL14] beschreibt eine allgemeine Softwarearchitektur zur Testautomatisierung für Prüfstände, jedoch mit einem Fokus auf komplette Antriebsstränge. Die dort vorgestellte Architektur weist zwar eine in Teilen vergleichbare Struktur auf, ist in ihrem Anwendungsschwerpunkt jedoch auf Szenarien mit einem anderen Maßstab an Anforderungen ausgelegt als das hier präsentierte Konzept.

Zur Smart-Grid-Integration wurde in [Ap12] eine serviceorientierte Referenzarchitektur vorgestellt, die auch Komponenten für flexible Erzeuger, wie z.B. mmBHKW enthält. Die hier vorgestellte Prüfstandsarchitektur kann mit geringen Modifikationen in die Referenzarchitektur eingepasst werden, um ein mmBHKW direkt in ein Smart-Grid zu integrieren.

Eine cloudbasierte, serviceorientierte Architektur mit Microservices wird in [Ho16] auf Grund ihrer hohen Flexibilität und Adaptivität als Lösung für IT-Systeme mit hohen Ansprüchen an Robustheit und Stabilität gegenüber unerwarteten Ereignissen präsentiert, mit der diese in einen als „anti-fragil“ [Ho16] bezeichneten Zustand gebracht werden können. Aus diesem Grund wird dort auch deren Einsatz z.B. in Smart Grid Szenarien als besonders kritische Infrastruktur diskutiert.

Die in diesem Beitrag eingeführte Architektur ist auf die Integration der Systeme eines Prüfstands für mmBHKW ausgerichtet, bietet aber auch gute Anknüpfungspunkte für weitere Entwicklungen hinsichtlich der Anbindung eines Prototypen an bestehende und in Entwicklung befindliche Smart Grid Infrastrukturen und vergleichbare Systeme.

5 Fazit und Ausblick

In diesem Beitrag wurde ein Architekturkonzept zur softwaregestützten Überwachung und Steuerung des mmBHKW-Prüfstands der Hochschule Hannover vorgestellt. Dabei wurden die Anforderungen im Rahmen des interdisziplinären Forschungsprojekts analysiert und darauf aufbauend eine dreischichtige Softwarearchitektur aus hardwarenaher Prüfstandssteuerung, serviceorientierter Geschäftslogik und LabVIEW-Frontend entworfen.

Es wurde dargelegt, wie durch die klar definierten und abgegrenzten Zuständigkeitsbereiche der einzelnen Komponenten die Ansprüche an die Flexibilität und Anpassbarkeit des Systems an das dynamische Anforderungsumfeld eines Forschungsprüfstands erfüllt werden können. Die verwendete Servicearchitektur erlaubt zudem die Integration und Anbindung domänenspezifischer Anwendungen, wie am Beispiel der RTOS-basierten Prüfstandssteuerung sowie des virtuellen Leitstands in LabVIEW gezeigt wurde.

In unseren zukünftigen Arbeiten sollen u.a. folgende Themen betrachtet werden: Vollautomatisierung des Prüfstandes; Integration in Smart Grid Szenarien; Möglichkeiten zur Emissionsreduktion und Effizienzsteigerung; Performanz- und Skalierbarkeitsanalysen sowie der Einsatz von Complex Event Processing und Cloud Computing Technologien, ggf. basierend auf einer ereignisgetriebenen Architektur [BD10].

6 Danksagungen

Dieses Projekt wurde gefördert durch die VolkswagenStiftung und das Nds. Ministerium für Wissenschaft und Kultur (Fkz. VWZN2891). Wir bedanken uns bei den Studierenden des Bachelorprojekts und den Mitgliedern des Forschungsschwerpunkts „Skalierbarkeit mobiler Mikro-Blockheizkraftwerke“ an der Hochschule Hannover für ihre Unterstützung und die gute Zusammenarbeit.

Literaturverzeichnis

[Ap12] Appelrath, Hans-Jürgen; Bischofs, Ludger; Beenken, Petra; Uslar, Mathias: IT-Architekturentwicklung im Smart Grid. Springer, 2012.

[BD10] Bruns, Ralf; Dunkel, Jürgen: Event-Driven Architecture: Softwarearchitektur für ereignisgesteuerte Geschäftsprozesse. Xpert.press. Springer, Berlin [u.a.], 2010.

[Bo12] Bouvy, Claude; Baltzer, Sidney; Ernst, Christian; Eckstein, Lutz: Range Extender als Mobile Kraft-Wärme-Kopplungseinheit. ATZ - Automobiltechnische Zeitschrift, 114(10):764–769, 2012.

[Du08] Dunkel, Jürgen; Eberhart, Andreas; Fischer, Stefan; Kleiner, Carsten; Koschel, Arne: Systemarchitekturen für Verteilte Anwendungen: Client-Server, Multi-Tier, SOA, Event Driven Architectures, P2P, Grid, Web 2.0. Carl Hanser, München, 2008.

[Er14] Erl, Thomas: Next generation SOA: A concise introduction to service technology & service-orientation. Prentice Hall/PearsonPTR, 2014.

[Ge06] Gerth, Wilfried: , RTOS-UH, 2006. IEP.

[Ha12] Hadler, Andreas und Koerth, Klaus: , Dokumentation IF555-3, 2012. IEP.

[Ha16] Hanif, Haider Iqbal; Minnrich, Jan Peter; R. P. Schmicke, Christian; Rüscher, Henrik; Gusig, Lars-Oliver: Bauraum- und gewichttechnische Untersuchung einer mobilen mikro-PCU zur Bereitstellung von Wärme, Kälte und elektrischen Strom im E-Fahrzeug. In: 10. Tagung – Wärmemanagement des Kraftfahrzeugs, 09. - 10.06.2016. Potsdam, 2016.

[Ho16] Hole, Kjell Jørgen: Anti-fragile ICT Systems. Springer International Publishing, Cham, 2016.

[Mi14] Minnrich, Jan P.; Rüscher, Henrik; Schmicke, Christian R. P.; Gusig, Lars-Oliver: Integration einer mikro-PCU im Thermomanagement eines Elektrofahrzeugs unter Berücksichtigung von Reichweite und Emissionen. In: 9. Tagung — Wärmemanagement des Kraftfahrzeugs, 05 - 06.06.2014. Potsdam, 2014.

[Mi15] Minnrich, J. P.; Schmicke, Christian R.P.; Rüscher, Henrik; Gusig, Lars-Oliver: Multifuel Application in Alternative Range-Extender-Concepts. In: International Conference IDTechEx, 28. - 29.04.2015. Berlin, 2015.

[Ne15] Newman, Sam: Building microservices: Designing fine-grained systems. O'Reilly, 2015.

[PL14] Paulweber, Michael; Lebert, Klaus: Mess- und Prüfstandstechnik: Antriebsstrangentwicklung · Hybridisierung · Elektrifizierung. Springer Fachmedien Wiesbaden, Wiesbaden, Kapitel Softwaresicht – Prüfstand, S. 273–376, 2014.

[RB16] Referat Öffentlichkeitsarbeit; Bundesministerium für Wirtschaft und Energie: , BMWi - Leitmarkt und Leitanbieter. http://www.bmwi.de/DE/Themen/Industrie/Elektromobiltaet/leitmarkt-und-leitanbieter.html, 2016. online; abgerufen 10/05/2016.

[RSG15] Rüscher, Henrik; Schmicke, Christian R.P.; Gusig, Lars-Oliver: Applicability and scalability of mobile mCHP units in mid-size battery electric vehicles and detached houses with different energy standards. In: International Conference on Sustainability in Energy and Buildings 2015, 01. - 03.07.2015. Lisabon, Portugal, 2015.

[Rü14] Rüscher, Henrik; Schmicke, Christian R. P.; Minnrich, Jan P.; Gusig, Lars-Oliver: Weiterentwicklung von Range Extendern zu mobilen mikro-Blockheizkraftwerken in Fahrzeugen und Gebäuden. In: Techniktagung Kraft-Wärme-Kopplungssysteme, 29. - 30.04.2014. Berlin, 2014.

[Sc14] Schmicke, Christian R.P.; Rüscher, Henrik; Minnrich, Jan P.; Gusig, Lars-Oliver: Strategies for combined use of power conditioning units in vehicles and buildings. In: International Conference on Sustainability in Energy and Buildings 2014, 25 - 27.06.14. Cardiff, Wales, 2014.

[St15] Starke, Gernot: Effektive Softwarearchitekturen: Ein praktischer Leitfaden. Hanser, München, 7.. Auflage, 2015.