4 Zusammenfassung

Abschließend werden in Kapitel 4.1 die Ergebnisse dieser Arbeit zusammengefasst und in Kapitel 4.2 ab Seite 85 verschiedene Optimierungsmöglichkeiten diskutiert wonach in Kapitel 4.3 ab Seite 86 ein Ausblick auf weitere Aufgaben gegeben wird.

4.1 Ergebnisse

Für das In-situ-Diagnosesystem war ein modulares Software-Konzept zu entwickeln, dass die softwareseitige Infrastruktur für die Signalauswertung von Dehnungsmessstreifen aufbaut. Dazu zählen der Datenaustausch zwischen den Hardwaremodulen via CAN, die Möglichkeit der Parametrisierung und des Datenabgriffs via UART, wie auch die Signalerfassung und Umrechnung von Messdaten auf den Messmodulen und die Ermittlung der Belastungen auf dem Hauptmodul.

Die während der Arbeit vorgestellten Konzepte wurden in der Programmiersprache C in der Programmierumgebung Keil µVision implementiert und mithilfe des zugehörigen Keil-Compilers für die Mikrocontroller TI Stellaris und TIVA Mikrocontroller übersetzt. Bei der modularen Umsetzung des Software-Konzepts in C-Methoden, wurde auf einen hohen Abstraktionsgrad zur Erreichung einer hohen Wiederverwendbarkeit der einzelnen Software-Module geachtet. So wurden die von allen Messmodulen gleichermaßen, aber unterschiedlich parametrisierten Funktionen des Mikrocontrollers gegen Schnittstellen entwickelt und im Projektverzeichnis einem gemeinsamen Quellverzeichnis zugeordnet. Nur modulspezifische Implementierungen finden sich in den moduleigenen Ordnern. Die Quelldateien finden sich ausführlich kommentiert und mit der Versionskontrolle HG Mercurial verwaltet auf dem beiliegenden Datenträger im Anhang C.1[17].

Das für die Kommunikation entwickelte CAN-Modul enthält ein effizientes Kommunikationsprotokoll, dass durch Einführung von Broadcast- und Unicast-Kommunikationstypen und die effiziente Nutzung der Nachrichten-Identifier zur Identifikation dieser Kommunikationstypen, Modulen und Kommandos sowie Nachrichtentypen eine Abtastrate von bis zu 100 Hz ermöglicht, wenn die Daten via UART ausgegeben und keine Belastungsermittlung durchgeführt wird.

Für die Ermittlung der Belastungen aus den gemessenen Dehnungen wurden zwei der drei vorgestellten Varianten von Algorithmen implementiert und auf das Hauptmodul portiert.

[17] Die Projekte, Quelldateien und Ressourcen finden sich auf dem Datenträger unter „CD-Laufwerk:\3 Implementierung\Haupt-Repository\“

Im Test-Betrieb des In-situ-Diagnosesystem haben sich für die Variante A mit optimierten Speicherzugriff bei einer Diskretisierung von $\mathbf{s} = (11,11,11)$ und Nachbarschaft von $\mathbf{n} = (7,7,7)$ Abtastraten der gesamten Messkette von bis zu 5 Hz realisieren lassen. Die maximale Abtastrate ohne die algorithmische Bestimmung der Belastungen beträgt hingegen bis zu 100 Hz. Dies entspricht einer Verringerung der Abtastrate auf fünf Prozent. Erklären lässt sich diese geringe Abtastrate durch die teuren Speicherzugriffe auf die SD-Karte.

Bei dem Algorithmus Variante B hingegen, lässt bereits eine Diskretisierung von $\mathbf{s} = (13, 9,4)$ eine höhere Genauigkeit (relative Messabweichung vom Messbereichsendwert von 0,34 % für die Momente um die x-, 0,16 % für die um die y- und 1,2 % für die Momente um die z-Achse) zu gegenüber Variante A mit einer erheblich feineren Diskretisierung. Allerdings benötigt die Berechnung aufgrund des Speicherzugriffs erheblich mehr Zeit. So werden für die genannte Diskretisierung in Matlab 4,63 Sekunden benötigt. Doch auch auf dem Mikrocontroller stellten sich im Test-Betrieb der gesamten Messkette des In-situ-Diagnosesystems Umsetzungszeiten zwischen einer und vier Sekunden ein, was einer Abtastrate von 0,25 Hz entspricht.

Algorithmus Variante B erzielt also höhere Genauigkeiten durch geringeren Speicherbedarf. Dieses Mehr an Genauigkeit wird jedoch durch eine geringe Abtastrate erkauft.

4.2 Optimierungsmöglichkeiten

Während der Arbeit wurden seitens der Hard- als auch Software verschiedene Probleme identifiziert, die bei einer Weiterentwicklung des In-situ-Diagnosesystems zu berücksichtigen sind, um einen zuverlässigen und schnellen Betrieb zu ermöglichen.

Seitens der Software hat sich der Brückenabgleich auf den Messmodulen als wenig performant herausgestellt und kann im Worst-Case bei abgetrennten DMS mit 153,6 s zu Buche schlagen. Der Brückenabgleich könnte daher mit dem in [25] vorgestellten Verfahren für Wägeumsetzer beschleunigt werden, bei der man den veränderlichen Widerstand der Messbrücke nicht mit äquidistanten Stufen verändert, sondern mit einer großen Stufe beginnt und sich mit sukzessive verfeinerten Stufen einer abgeglichenen Messbrücke nähert.

Weiterhin könnte sich das während der Implementierung vorgenommene Scheduling per Hand als problematisch erweisen, wenn das Software-Konzept um weitere Komponenten erweitert werden muss, bspw. um weitere Berechnungen, die viel Rechenzeit beanspruchen. Ein geschicktes Aufteilen der Rechenschritte in kleinere Prozesse erscheint sich gerade bei Algorithmen als fehleranfällig. Das hat sich bereits bei den Umrechnungsalgorithmen von Belastungen aus Dehnungen gezeigt. Wenn die Rechenzeiten sehr lange werden, verliert der

Mikrocontroller zu viel Zeit für andere Prozesse. Diese verhungern und die Funktion anderer Software-Module wird eingeschränkt wie etwa das Senden und Empfangen von Nachrichten via CAN und UART. Daher empfiehlt es sich bei steigender Komplexität des Gesamtsystems, ein Echtzeitbetriebssystem einzusetzen, dass das Scheduling und die Kontextwechsel übernimmt und so die Fehleranfälligkeit der händischen Methode reduziert.

Auch seitens der Hardware lassen Optimierungsmöglichkeiten erkennen. Obwohl die Mikrocontroller der DMS-Messmodule zwei Analog-Digital-Umsetzer enthalten, wird nur einer verwendet. Dies kann software-seitig nicht anders konfiguriert werden, da die verwendeten Pins fest einem Kanal und ein Kanal fest einem ADU zugewiesen ist. Bei Verwendung eines Pins eines Kanals des zweiten ADU, könnte die effektive Abtastrate der DMS-Messmodule verdoppelt werden [21], [44].

Bei dem für den CAN-Bus produzierten Kabelbus für den Testbetrieb, wurde noch keine Schirmung angebracht, wodurch es zur Einstreuung elektromagnetischer Felder und damit zur Beeinträchtigung der Nachrichtenübertragung via CAN kommen kann. Die Verwendung einer abgeschirmten Leitung mit der Abschirmung gegen Masse, kann dieses Problem beheben.

Schlussendlich hat sich die Auswahl der Mikrocontroller für das Hauptmodul als optimierbar herausgestellt. Um eine effiziente Implementierung der Algorithmen zu gewährleisten, deren Laufzeiten signifikant von Speicherzugriffszeiten abhängen, sollten Mikrocontroller ausgewählt werden, welche die software-seitige Verwendung eines größeren Stack- bzw. Heap-Speichers erlauben. Dadurch könnte die Nachschlagetabelle initial von der SD-karte oder aus dem Flash in den Stack geladen werden, was zu einer deutlich geringeren Laufzeit der Algorithmen führen könnte. Auch die Verwendung von extern an den Mikrocontroller anzubindenden SRAM oder FRAM könnte sich als ähnlich vorteilhaft herausstellen. Neben der Verringerung der Speicherzugriffszeiten wären mit der erhöhten Speicherkapazität auch die Umsetzung größerer Datenstrukturen wie Warteschlangen oder Datenpuffer für den Scheduler und Software-Filter denkbar, wodurch sich alle erarbeiteten Konzepte aus der Arbeit effizient umsetzen ließen.

4.3 Ausblick

Nachdem in dieser Arbeit das grundlegende Software-Konzept für das In-situ-Diagnosesystem geschaffen und Algorithmen zur effizienten Belastungsermittlung entwickelt wurden, können diese Daten als Basis für zukünftige Optimierungen und Weiterentwicklungen dienen. Wenn die vorhergehenden Optimierungen umgesetzt und damit die Laufzeit der Algorithmen verringert wird, steht einer anschließenden Lebensdaueranalyse bzw. der Feststellung des Verschleißzustands des Wechslers mithilfe der ermittelten Belastungen nichts mehr im Wege.