Das leistungsfähige und modulare Werkzeug *Maglap++* zur Zustandsdiagnose und Überwachung magnetgelagerter Maschinen

Mikhail SHMACHKOV, Ivo NOACK, Frank WORLITZ
Institut für Prozeßtechnik, Prozeßautomatisierung und Meßtechnik (IPM)
Hochschule Zittau/Görlitz
Theodor-Körner-Allee 16
02763 Zittau, Germany
Tel.: +49 3583 612 4383, Fax: +49 3583 612 3449
Email: Mikhail.Shmachkov@hszg.de, I.Noack@hszg.de, F.Worlitz@hszg.de

Kurzfassung

Moderne aktive Magnetlager sind hoch präzise mechatronische Systeme, deren Überwachung, Zustandsdiagnose und Analyse spezielle Anforderungen mit sich bringen. Diese müssen auch von der zu verwendenden Software zur Messwerterfassung, Datenspeicherung und -verarbeitung erfüllt werden. Ein solches Messwerterfassungssystem ist eine komplexe Software, die vor allem schnell und genau arbeiten muss. Die bisherigen Umsetzungen unterlagen einigen Einschränkungen, die sich nachteilig auf Effizienz, Erweiterbarkeit und Programmdesign auswirkten. Ebenfalls problematisch sind die verwendeten Lizenzmodelle und die fehlende Quellcodetransparenz in der Software. Für das Forschungspersonal ist nicht ersichtlich, wie interne Funktionen (auch Module usw.) arbeiten und wie sie in bestimmten Situationen reagieren. Um diesen Problemen und Begrenzungen entgegenzuwirken, wurde am Institut für Prozeßtechnik, Prozeßautomatisierung und Meßtechnik (IPM) mit der Entwicklung der Software „Magnetlagerprogramm“ (*Maglap++*, Logo: Abb. 1) begonnen.

Abb. 1: Logo von Maglap++

Maglap++ ist ein Cross-Plattform-Projekt in der Programmiersprache C++ und kann derzeit auf Windows und Linux gleichermaßen eingesetzt werden. Das System ist sehr modular aufgebaut, wodurch sich nahezu beliebige Verarbeitungsketten erzeugen lassen. So kann das Programm, allein durch die Anordnung der zu verwendenden Module und hier speziell der Datenquelle und der Datensenke, zu einem Server oder einer Client-Applikation konfiguriert werden. So ist zum Beispiel ein performantes Aufzeichnen und Überwachen der Anlage möglich, während zeitgleich auch Techniker von anderen Geräten aus zu Diagnosezwecken auf die Daten schauen können. Denkbar wäre hierfür auch eine Android-App, die auf einem Tablet als Analysewerkzeug fungiert.

Da Herkunft und Ziel der Daten durch eigenständige Module bestimmt werden und alle verarbeitenden Module weitestgehend frei kombiniert werden können, beschränkt sich der Einsatz von *Maglap++* nicht nur auf die Magnetlagertechnik. Vielmehr bietet es einen Raum zur kreativen Gestaltung von Verarbeitungsketten, bis hin zur modellgestützten Simulation von Anlagenteilen. Die Stärken von *Maglap++* liegen in einer effizienten, plattformunabhängigen Programmierung und dem äußerst modularen Aufbau.

1 Einleitung

1.1 Motivation

Die Magnetlagertechnik findet große Verwendung in Industrie und Wissenschaft, vor allem im Bereich der Energie- und Weltraumtechnik. Im Rahmen verschiedener Projekte finden am IPM die theoretischen und experimentellen Untersuchungen an den Versuchsanlagen statt. Für die Arbeit an derartigen Versuchsständen wird Software benötigt, die den speziellen Anforderungen gewachsen ist. Im Rahmen von Industrie 4.0 spielt die Vernetzung der Systemkomponenten sowie deren Zustandsdiagnose und Überwachung eine entscheidende Rolle. Die Magnetlagerregelkreise besitzen aufgrund ihres Funktionsprinzips inhärent das Potenzial, derartig ausgerüstete Applikationen in ein Netzwerk im Sinne von Industrie 4.0 zu integrieren.

Die Durchführung der theoretischen und experimentellen Forschung beeinflusst mehrere Bereiche der Industrie. Die wissenschaftlichen Untersuchungen an den Versuchsanlagen bedürfen der sensorischen Aufnahme von Daten durch die Technik mithilfe von Messgeräten. Anschließend müssen die erfassten, analogen Werte, zur weiteren Verarbeitung, in eine digitale Form überführt werden. Die erstellten Messwerte müssen mithilfe von mathematischen und physikalischen Algorithmen verarbeitet und analysiert werden. Diese Methoden stellen sich als komplizierte und langwierige Kalkulationen dar. Dazu ist eine spezielle analytische Software notwendig. Abbildung 2 stellt diesen Themenkomplex schematisch dar und dient als Orientierungspunkt bei der Entwicklung von *Maglap++*.

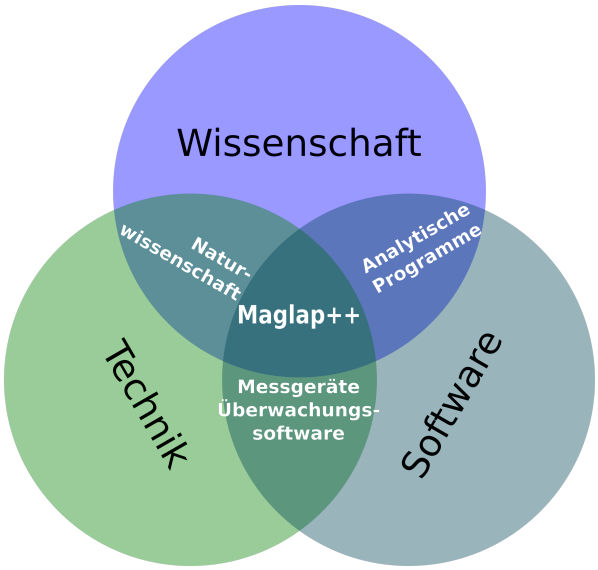


Abb. 2: thematische Einordnung von Maglap++

1.2 Maglap++

Das Projekt *Maglap++* wurde als studentisches Programm zur simplen Messwerterfassung gegründet. Während der Entwicklung (2014-2016) wurde dieses Programm von Studenten aus verschiedenen Ländern und Studienrichtungen erweitert. Dank des Enthusiasmus der Entwickler und durch die Hilfe der hochqualifizierten Mitarbeiter des IPMs wurde das Ziel gesetzt, *Maglap++* als qualitativ hochwertige und anwendungsfreundliche Software aufzubauen. Überdies wurde die Entscheidung getroffen, ein modulares Konzept in dem Programm umzusetzen, welches den Entwicklungsprozess, vor allem mit Blick auf die nachfolgende Erweiterung, verbessert. Die dem Programm zugewiesenen Aufgaben können jetzt als unabhängige Module realisiert werden. Außerdem ist es damit möglich, die Arbeit zwischen dem Entwicklungspersonal aufzuteilen und sauber zu trennen. Die Erweiterung des Projektes erfolgt mithilfe des integrierten Plugin-Systems. Die dazu entwickelte Programm- und Elementorganisation erlaubt es, die neuen Module leicht und unabhängig zu entwickeln und in das Hauptprogramm zu integrieren.

1.3 Zielstellung

Formulierung

Für theoretische und experimentelle Untersuchengen an Magnetgenlagerversuchsständen ist es notwendig, eine leistungsfähige und funktionale Software zur

- Messwerterfassung,
- Datenverarbeitung,
- Client-Server-Verbindung,
- Darstellung,
- Analyse,
- Datenspeicherung und
- Zustandsdiagnose

zur Verfügung zu haben. Dafür wird am IPM (Fachgebiet „Mechatronische Systeme“) das Projekt *Maglap++* bearbeitet. Das Messwerterfassungssystem *Maglap++* ist eine Software zur Prozessverarbeitung, Prozessanalyse, Prozessauswertung und Visualisierung von Prozessen und Prozesszuständen. Dieses System besteht aus zwei Teilen: Client und Server (Abb. 3). Das Konzept einer Client-Server-Struktur sah vor, die Daten per Netzwerkverbindung an einen angeschlossenen Client zu übertragen, um die „Mensch-Maschine-Schnittstelle“ örtlich von der Messwerterfassung zu trennen.

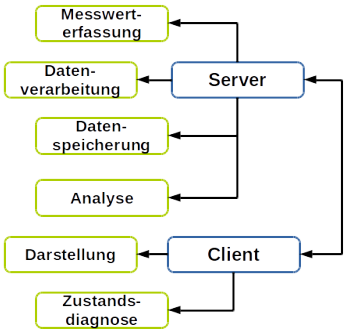


Abb. 3: Maglap++ System

Programmanforderungen

Maglap++ soll eine universelle, modulare und flexible Cross-Plattform-Software werden. Es soll aus mehreren Modulen bestehen, die jeweils bestimmte Aufgaben erfüllen. Dazu müssen zunächst Module existieren, die in der Lage sind, Messwertdaten zu erfassen. Als Quellen können dabei verschiedene Geräte (z.B. Messwerterfassungskarte), Netzverbindungen und Dateien dienen. Die erfassten Daten sind anschließend zu verarbeiten (z.B. Skalierung, Reduzierung usw.). Die Ergebnisse müssen dargestellt, gespeichert und protokolliert werden. Je nach Wunsch des Benutzers müssen verschiedene Module eingesetzt und konfiguriert werden. Für die Verwendung, Erweiterung und Integration der Module muss ein Extension-/Plugin-System existieren. *Maglap++* soll sich nicht nur auf Magnetlagertechnik fokussieren, sondern ein gutes Forschung-Tools für alle naturwissenschaftlichen Richtungen sein.

Um eine hohe Qualität des Programms sicherzustellen, müssen die Regeln der Softwaretechnik beachtet werden. Das System muss die Möglichkeit bieten, die verschiedenen Aufgaben (und ihre Kombinationen) realisieren zu können. Alle Einzelteile des Programms sollen eine ereignisgesteuerte Nutzeroberfläche (eng. Graphical User Interface) und eine Konfigurationsdatei besitzen. Bei der Entwicklung der Komponenten ist zu beachten, dass die zu verarbeitenden Prozesse unterschiedlich sind (z.B. Anzahl der Kanäle, verschiedene Datenmengen, Abtastfrequenz usw.). Dazu ist eine Systematisierung der Eingangsdaten zu erstellen. Für die Entwicklung solcher Schnittstellen sind entsprechende Regeln zu definieren. Die Programmiervorschriften basieren auf der allgemeinen Programmierrichtlinie des IPM. Für Test und Verifikation ist eine Testanleitung für die Software zu verfassen. Notwendig ist die Erstellung eines Test-Frameworks mit allgemeinen Testverfahren und Testszenarien. Für das gesamte Projekt sollen eine Dokumentation und eine Bedienungsanleitung erstellt werden.

2 Projektvorstellung

2.1 Notwendige Definitionen

Durch die modulare Struktur von *Maglap++*, ist es notwendig, die Hauptschnittstellen exakt zu beschreiben. Das heißt, um die Dokumentation klar, deutlich und verständlich zu präsentieren, müssen die verschiedenen Programmteile definiert werden.

Das Programm muss eine benutzerfreundliche Bedienoberfläche und eine dynamische Integration der Module mithilfe eines Plugin-Systems bieten. Dafür ist „*General Core*“ verantwortlich. Dieses Element stellt sich als Basis von *Maglap++* dar. Außerdem wird das Erweiterungssystem des Programms und der Module unterstützt. Später kann auch „*General Core*“ wegen des modularen Aufbaus erweitert oder verändert werden.

Die Module, welche spezielle Aufgaben realisierende, werden als „*Units*“ festgelegt. Das heißt, die Module, welche die Programmfunktionen von *Maglap++*, wie Messwerterfassung, Datenverarbeitung usw., darstellen, bilden einen speziellen Modulbereich. Die *Maglap++-Units* unterteilen sich in drei bestimmten Gruppen:

Receiver — Die Units, welche für die Erfassung der Daten (Messwerte usw.) von verschiedenen Quellen und Übergabe dieser an interne Module (Units/Core) verantwortlich sind;
Processing — Die Units, welche die Daten verarbeiten und die notwendigen Kalkulationen realisieren;
Sender — Die Units, welche die verarbeiteten (berechneten) Daten oder Messwerte an externe Objekte (Dateien, Clients usw.) übergeben;

Und die letzte Definition ist „*Core*“. Sie stellt sich als komplexe Unit dar. Dort kann die Kombination der zu verwendenden Units erstellt werden. Anderes genannt, *Core* ist ein Unterprogramm, welches komplizierte Aufgaben verwirklicht. Zusätzlich gehören die Module dazu, welche sich nicht in eine der drei Grundgruppen der Units einordnen lassen. Mit anderen Worten, es handelt sich dabei um besondere Units.

2.2 Arbeitspunkte

Die Aufgabenbearbeitung erfolgt nach den Regeln der objektorientierten Programmierung (OOP) und des Software Engineering (SE). Für die Erfüllung der Zielstellung müssen die folgende Arbeitspunkte (AP) bearbeitet werden:

- **Programm-AP:**
  - AP-1. Betriebssystemunabhängiges Basisprogramm
  - AP-2. Dynamisches Modulsystem
  - AP-3. Test und Verifikation
  - AP-4. Dokumentation der Ergebnisse
- **Unit/Core-AP:**
  - AP-5. Messwerterfassung
  - AP-6. Datenbearbeitung
  - AP-7. Client-Server-Verbindung
  - AP-8. visuelle Darstellung
  - AP-9. Messwertanalyse/-auswertung
  - AP-10. Messwertarchivierung
  - AP-11. Zustandsdiagnose

Dies sind die aktuellen Aufgaben des Projekts. Die Funktionalität wird in Zukunft erweitert. Außerdem ermöglicht es das aktuelles Konzept, Module zu erstellen und in das Programm zu integrieren, die nicht auf Magnetlagertechnik beschränkt sind. Damit kann *Maglap++* verschiedene Modifikationen unterstützen. Abschließend kann man das Hauptziel des Projekts als universales, betriebssystemunabhängiges modulares Forschung-Werkzeug mit örtlicher Trennung der Mensch-Maschine-Schnittstelle definieren.

2.3 Programm-Arbeitspunkte

AP-1. Betriebssystemunabhängiges Basisprogramm
Das Programm soll weitestgehend betriebssystemunabhängig sein, d.h. *Maglap++* wird für die Betriebssysteme Windows und Linux als Cross-Plattform-Applikation umgesetzt. Dazu soll ein universaler Quellcode verwendet werden. Für die Entwicklung des Programms wird die Programmiersprache „C++“ genutzt. Um die Unabhängigkeit vom Betriebssystem zu erreichen und eine grafische Benutzeroberfläche (GUI) zu erstellen, wird die Programmbibliothek „wxWidgets“ verwendet. Für die Erstellung des Quellcodes und des Kompilats wird „Code::Blocks“ als integrierte Entwicklungsumgebung (IDE) in Kombination mit dem „C++“-Compiler der GNU Compiler Collection („GCC“) genutzt.

AP-2. Dynamisches Modulsystem
Ein allgemeines, unabhängiges und modulares Umsetzungskonzept ist zu entwickeln. Dieses Konzept soll die Möglichkeit bieten, neue Elemente und Komponenten leicht und schnell zu integrieren. Es ist als API zu realisieren. *Maglap++* ist nicht nur auf Magnetlagertechnik fokussiert, sondern soll als universelle, modulare und flexible Cross-Plattform-Software mit Extension-/Plugin-System zur Prozessverarbeitung, Prozessanalyse, Prozessauswertung und Visualisierung von Prozessen und Prozesszuständen realisiert werden.

AP-3. Test und Verifikation der Software
Um die Funktionsfähigkeit der erstellten Software nachzuweisen und die Ergebnisse zu verifizieren, sind verschiedene Tests und Vergleiche mit Referenzprogrammen durchzuführen. Für die Tests der Software sind Testanleitungen und Testszenarien zu erstellen. Ein Test und eine Verifikation müssen laut SE-Regeln durchgeführt werden. Die erhaltenen Ergebnisse sind zu dokumentieren und zu protokollieren.

AP-4. Dokumentation der Ergebnisse
Alle Ergebnisse sind zu dokumentieren. Die Dokumentationssprache ist deutsch und in Teilen englisch. Insbesondere werden die Quellcodedokumentation und ausgewählte Diagramme englisch dokumentiert. Damit ist sichergestellt, dass fremdsprachiges Personal barrierefrei damit arbeiten kann. Darüber hinaus ist Mehrsprachigkeit sinnvoll, da Literaturquellen und Syntax im Bereich der Softwareentwicklung grundsätzlich in englisch ausgeführt sind. So fügt sich die Dokumentation nahtlos in die Entwicklungsumgebung ein. Als Dokumentation zu realisieren:

- Lastenheft
- Pflichtenheft
- Balkenplan
- Arbeitspaketbeschreibung
- Projektdokumentation
- Quellcodedokumentation

2.4 Unit/Core-Arbeitspunkte

AP-5. Messwerterfassung
Das Hauptziel der Messwerterfassung ist die Erfassung der Daten von der Technik. Ausgehend vom Ziel gehört sie zum zweiten Bereich: die Messtechnik und die Software. Die Messwerterfassung unterteilt sich in drei Schritte und bildet eine Messkette (Abb. 4):

- Erfassung der Messgröße
- Umwandlung des Messsignals in ein normiertes Ausgangssignal
- Erfassung der Messwerte (Digitalisierung)

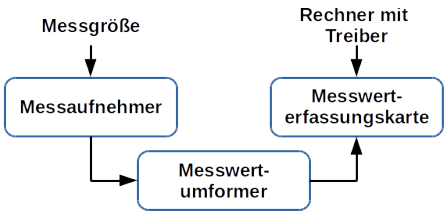


Abb. 4: Messkette

Die Charakteristiken des Versuchsstandes (z.B. Rotorpositionen) werden sensorisch erfasst. Der Messaufnehmer wandelt die physikalische Messgröße in ein elektrisches Strom- oder Spannungssignal um.

Die Signale der Sensoren werden durch Kabel (Sensorleitung) an ein Umformungsgerät übertragen. Der Messumformer speist einerseits das Sensorelement mit elektrischer Hilfsenergie und normiert andererseits das Messsignal in ein analoges Ausgangssignal. Das heißt, Eingangsignale werden mithilfe des analogen Messwertumformers in definierte, analoge Ausgangssignale umgewandelt.

Dieses Ausgangssignal wird mithilfe der Messwerterfassungskarte erfasst und in eine digitale Form umgewandelt. Diese Daten werden im Puffer der Karte zwischengespeichert, von wo aus sie mithilfe des Treibers in das Programm übernommen werden können. Diese Aufgabe übernimmt das Messwerterfassungsmodul. Hierbei handelt es sich um eine Receiver-Unit von *Maglap++*. Diese Unit muss ständig die Messdaten erfassen und an die nachfolgenden Units (oder Cores) übertragen. Die Messwerterfassung soll vollständig konfigurierbar sein, um den Messwerterfassungsprozess wie gewünscht steuern zu können.

AP-6. Datenbearbeitung
Die Zustandsdiagnose und insbesondere die Visualisierung an der Mensch-Maschine-Schnittstelle erfolgt auf der Basis physikalischer Messgrößen. Die dazu erforderlichen Umwandlungen und Umrechnungen einschließlich einer ggf. notwendigen Filterung erfolgen in der Datenverarbeitung. Dazu werden Units aus der Gruppe der Processing-Units verwendet. Diese Units müssen ihre Daten von einer vorangestellten Unit bekommen, dann verarbeiten (umrechnen, usw.) und die erstellten Ergebnisse weiter an nachfolgende Units übergeben. Dieser Prozess muss vom Benutzer kontrolliert werden können und ereignisgesteuert ablaufen.

AP-7. Client-Server-Verbindung
Der Server, im Sinne der Messwerterfassung und Verarbeitung, und der Client, im Sinne des Beobachters, können räumlich voneinander getrennt eingesetzt werden. Die Kommunikation zwischen Server und Client soll dazu unter Anwendung industrieller Netzwerktechnologien erfolgen. Dazu zählen insbesondere die Kombination aus Transmission Control Protocol und Internet Protocol (TCP/IP). Auf der Server-Seite muss dazu das Server-Modul eingesetzt werden, welches die Netzwerkverbindung erstellt und an den relevanten Ports lauscht (Bind und Listening). Das Beobachter-Programm muss sich mithilfe des Client-Moduls mit dem Server verbinden und kommunizieren. Nach der Erstellung der Client-Server-Verbindung müssen die Server-Daten auf die Clients übertragen werden.

AP-8. Visuelle Darstellung
Die von einem Server mit Messwerterfassungskarte oder einem Mirror-Server empfangenen Daten sollen auf der Client-Seite geeignet dargestellt werden. Die Informationen sind so aufzubereiten, dass eine Anzeige tabellarisch, im kartesischen Koordinatensystem und/oder als Orbit auswählbar ist. Diagramme mit einem kartesischen Koordinatensystem stellen auf der Abzisse die Zeit und auf der Ordinate den Funktionswert (physikalische Messgröße) dar. Die Orbitdarstellung eignet sich zur Visualisierung von Wellenlagen und Spulenströmen.

AP-9. Messwertanalyse/-auswertung
Ein Auswertungsmodul ist zu erstellen, das die Daten auswerten und die Resultat protokollieren kann. Zu den Aufgaben der Auswertung gehören die Berechnung, Bewertung, Analyse und Einschätzung der Ergebnisse. Außerdem ist eine allgemeine TEX-Vorlage zu entwickeln, die für die Erstellung des Protokolls in der Form einer PDF-Datei (PDF) genutzt werden kann.

AP-10. Messwertarchivierung
Ein Archivierungsmodul ist zu implementieren, das in einer geeigneten Form verschiedene Messdaten speichern kann. Die Messwertarchivierung soll verschiedene Varianten der Speicherung und Komprimierung anbieten. Innerhalb der Entwicklung der Messwertarchivierung wird das Ziel verfolgt, den TDC, der die gruppierten Daten (Messwerte) und eine Beschreibung dazu enthält, und das Verfahren TT, das den Kompressionskoeffizienten mithilfe der spezifischen Kodierung verbessert, zu erstellen.

AP-11. Zustandsdiagnose
Mit „Maglap++“ soll auch der Zustand einer magnetgelagerten Maschine diagnostizierbar sein. Hierzu sind Informationen durch geeignete Algorithmen so zu verdichten, dass ausgehend von abstrakten Messwerten Bewertungsaussagen möglich werden. Die Algorithmen sollen um Codeteile zur Visualisierung erweitert werden. Insbesondere soll auf mobilen Endgeräten (Android-Client) dem Anwender/Bediener ein Werkzeug in die Hand gegeben werden, mit dem auf eine intuitive Art und Weise der Gesamtzustand sowie die Zustände in den Teilanlagen angezeigt werden. Im einfachsten Fall sind hier Ampelfarben zu wählen.

3 Konzeption

3.1 Programmkonzept

Von Mitte 2015 bis zur ersten Hälfte des Jahres 2016 wurde das Projekt *Maglap++* als eine Programmfamilie aus Server- und Client-Programmen entwickelt. Aber dieses Konzept wird den steigenden Anforderungen nicht gerecht. Die Programmfamilie *Maglap++* konnte nicht mit vertretbarem Aufwand erweitert werden. Weiterhin sieht das ursprüngliche Umsetzungskonzept keine dynamische Einbindung von Programmbibliotheken vor. Deswegen wurde als Ziel gesetzt, das Umsetzungskonzept zu modernisieren. Die softwaretechnische Lösung und die Idee des Modulkonzepts ergaben sich aus der Analyse der Gesamtfunktion der Server- und Client-Programme.

Der Server bildet die Basis des Messwerterfassungssystems. Laut den formulierten Anforderungen zu *Maglap++*:
Der Server

- erfasst die Daten aus einer Datenquelle (z. B. Messwerterfassungskarte),
- verarbeitet diese (Datenvorverarbeitung, Datenverarbeitung),
- speichert,
- archiviert,
- analysiert,
- diagnostiziert und
- stellt sie einem Client zur Verfügung.

Die Abbildung 5 zeigt die Funktionsverkettung des Servers. Die Verarbeitungskette besteht aus drei Abschnitten:
• **Erfassung** — Programm muss die Daten aus dem Puffer des Messwerterfassungskarte abholen. Als Alternative können die Daten auch aus Dateien übernommen werden.
• **Verarbeitung** — Die erfassten Daten (Messwerte) müssen umgewandelt oder kalkuliert werden. Außerdem muss die Verarbeitung der Daten, sowie deren Analyse und Transformation durchgeführt werden.
• **Übergabe** — Programm bietet die folgenden Möglichkeiten für die verarbeiteten Daten: Archivierung (Speicherung in Dateien), Auswertung (ereignisgesteuerte Speicherung mit Protokoll) und Übertragung an die Clients (Netzwerk) – nämlich die Übergabe der Daten an externe Objekte (Dateien, Netzwerk).

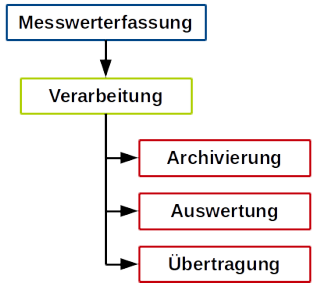


Abb. 5: Server-Aufgaben

Die Verwendung eines Clients setzt die Einrichtung eines Servers voraus. Der Client mit seinen notwendigen Funktionen stellt die Mensch-Maschine-Schnittstelle dar. Somit ist er aus Sicht des Werkzeuganwenders eine wesentliche Komponente im Programmpaket *Maglap++*. Zu seinen wesentlichen Funktionen zählen die

- Initialisierung der Netzwerkverbindung zum Server,
- Datenabholung,
- Datenverarbeitung und
- Darstellung/Visualisierung von Informationen.

Abbildung 6 zeigt die Funktionsverkettung des Clients. Der Client vollzieht die folgende Aufgaben:
• **Datenempfang** — Erfassung der Daten vom Server über eine Netzwerkverbindung.
• **Datenvorbereitung** — Die erfassten Daten müssen zur Weiterarbeit vorbereitet werden, zB. Umwandlung in Messgrößen, Erstellung Zeitverlauf für die graphische Darstellung, usw.
• **Darstellung** — Darstellung der vorbereiteten Daten als Diagramm oder Tabelle
• **Zustandsdiagnose** — Note: Android-App, die Ergebnisse der Zustandsdiagnose werden von der

App in geeigneter Form angezeigt.

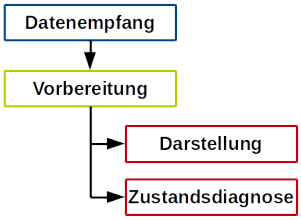


Abb. 6: Client-Aufgabe

Der Vergleich der Funktionsketten von Server und Client zeigt, dass sie in wesentlichen Teilen übereinstimmen. Während sich der Anwendungsnutzen grundsätzlich unterscheidet ist die Funktionsstruktur – Eingabe, Verarbeitung, Ausgabe – identisch. In diesem Zusammenhang sei auf die Abbildung 7 verwiesen. Die Programme sollen

- von einer frei wählbaren Informationsquelle (Messwerterfassungskarte, Dateien, Netz) die Daten abrufen (erfassen),
- den Datensatz verarbeiten (Skalierung, Reduktion, Aufbereitung zur Darstellung) und
- die verdichteten Informationen übertragen (senden, speichern, anzeigen).

Es ist das Umsetzungskonzept so weiterzuentwickeln, dass die Funktionskette nach Abbildung 7 in Server und Client Anwendung findet.

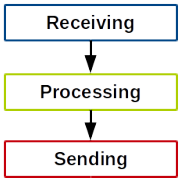


Abb. 7: Programm-Aufgabe

Das Konzept macht Funktionsmodule mit definierten Ein- und Ausgabeschnittstellen erforderlich. Diese werden durch den Anwender in logischen Ketten (Datenfluss- und Verarbeitungskette) angeordnet, mit denen er jeweils eine Server- oder Clientfunktionsstruktur abbildet. Das heißt, aufgrund des neuen Konzepts kann das Programm durch dem Benutzer mit den erstellten Modulen dynamisch konfiguriert und aufgebaut werden (Abb. 8).

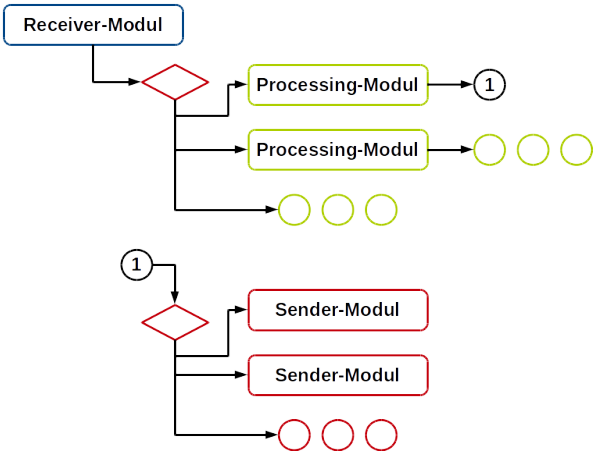


Abb. 8: Logische Kette

3.2 Modulsystem

Maglap++ basiert auf dem Prinzip „Receiver-Processing-Sender“. Außerdem bietet dieses Modulsystem die Möglichkeit, die Module durch Extensions zu erweitern oder neue Module in das Programm einzufügen. Dafür ist der *General Core* verantwortlich. Die in den *Logischen Ketten* verwendeten Module sind *Units* und *Cores* (siehe Kap. 2.1).

Die modulare Programmierung — eine Organisation des Programms (Software) als Gesamtheit der unabhängigen Blöcke (Module), deren Struktur und Funktionalität definierte Regeln folgt. Das verbessert die Mobilität des Programms. Außerdem vereinfacht die modulare Programmierung den Entwicklungsprozess und die Verifikation (auch Test usw.) der Software. Man versteht das Modul als fertiges Einzelteil (Komponente) des Programms, welches eine definierte Aufgabe erfüllt.

Bei der Erstellung und Weiterentwicklung der Programme entsteht die Notwendigkeit, die Funktionalität zu ändern oder zu ergänzen. Dies führt häufig dazu, dass man den Quellcode teilweise oder stark verändern muss. Programmentwickler müssen das Programm dann vollständig neu kompilieren. Die häufigste Lösung ist das Plugin-System. Ein Plugin (häufig auch Plug-in) ist ein unabhängig kompiliertes optionales Software-Modul, das dem Hauptprogramm dynamisch angeschlossen wird und die bestehende Software erweitert bzw. verändert. Das Plugin-System liefert neben den Erweiterungen ein Datenaustauschprotokoll und die Möglichkeit das Plugin im Programm zu registrieren. Softwarehersteller definieren Programmierschnittstellen (Ap-

plication Programming Interface (API)) zu ihren Produkten, mit denen Erweiterungen für diese Software entwickelt werden können.

Das Modulsystem in *Maglap++* wird mithilfe der Vererbung (Objektorientierte Programmierung) erstellt. Die Notwendige API wird in der Basisklasse definiert. Die Erstellung oder Weiterentwicklung der Module realisiert man mithilfe abgeleiteter Klassen. Eine dynamische Erweiterung des Programms (Plugin-System) während Arbeit der Applikation (Explizites Verknüpfen) wird durch die Verwendung dynamischer Bibliotheken erreicht. Die dynamische Bibliotheksdatei wird bei Unix-Betriebssystemen als Shared Object (SO) bezeichnet. Sie verfügen meist über die Dateiendung „.so“. Im Betriebssystem Windows wird eine solche Datei als Dynamic Link Library (DLL) bezeichnet und trägt die Dateiendung „.dll“.

3.3 Archivierung

Datenspeicherung

In letzter Zeit vergrößert sich die Anzahl der Informationen weltweit und die Frage der optimalen Datenspeicherung wird relevant. Es gibt viele Formate (Möglichkeiten), die Daten zu speichern, z.B. Audio-, Video-, Text-Formate usw. Im vorliegenden Fall ist die Lagerung der technischen Angaben interessant. Als Kriterien spielen die folgenden Charakteristiken eine große Rolle:

- Geschwindigkeit des Lesens/Schreibens,
- Art der Kodierung/Dekodierung,
- Optimierung auf Platzbedarf,
- Möglichkeiten der Benutzung/des Zugriffs (Zugänglichkeit)

Zurzeit existieren zwei Basisdateiformatsrichtungen:

• **Textformat** — Speicherung der Information als symbolische Zeile. Diese Variante ist verfügbar, sicher und stabil. Aber das größte Problem ist die ineffiziente Speicherung der Daten (insbesondere von Zahlen).
• **Binär Format** — verschlüsselte oder komprimierte Speicherung der Daten als binärer Code. Wegen des Fehlens der Verbildlichung sind binäre Dateien schnell und platzsparend.
Maglap++ bietet beide Möglichkeiten. Dazu existieren die TDC-Units (Receiver - zum Lesen, Sender - zum Speichern). Diese Units verwirklichen „Technical Data Container“, welcher mit verschieden Dateiformaten arbeiten kann.

Datenspeicherungstrategie

Während der Forschungsarbeiten erhält man eine große Menge an Daten. Daraus entsteht das Problem der platzsparenden Datenspeicherung. Die erste und einfachste Lösung ist Datenreduktion. Diese Methode reduziert die Anzahl der Werte durch zB. Mittelwertbildung. Die Reduzierung ist eine Lösung um die Dateigröße zu verringern, aber dies ist nicht für alle Gelegenheiten geeignet. Zum Beispiel soll man Daten, die nicht im Normalbetrieb liegen, ohne Verlust speichern. Deswegen spielt die ereignisgesteuerte Speicherung eine große Rolle. Das heißt, die Unit muss die Daten analysieren und eine Entscheidung treffen, wie die Daten zu speichern sind. Dazu müssen die entsprechenden Bedienungen durch den Benutzer eingesetzt werden. Für *Maglap++* werden *TDC-Units* entwickelt, die Daten einfach speichern/lesen können. Die ereignisgesteuerte Speicherung wird in *Processing-Units* realisiert. Zurzeit existiert eine solche *Unit*, welche die Daten im Fehlerfall vollständig (eireignisgesteuert) an die nachfolgende Unit übergibt und die Mittelwerte immer erstellt.

Diese Datenspeicherungstrategie ist schon sehr effektiv, aber der Speicherbedarf kann noch weiter reduziert werden. Die Archivierung der Daten (komprimierte Speicherung) hilft die Datengröße zu verringern und zu organisieren. Deswegen spielt die Erweiterung der Komprimierungskoeffizienten eine wichtige Rolle. Dazu wird die *TT-Unit* für *Maglap++* entwickelt, welche auf einer speziellen Datencodierung basiert (siehe Kap. Table-Transposition (TT)).

Table-Transposition (TT)

Das Archivierungsprinzip basiert auf einer Codierung mithilfe eines Wörterbuches, welches auf Grundlage einer Analyse der zu komprimierenden Daten erstellt wird. Dazu werden, statt der eigentlichen Datenketten, die entsprechenden, kurzen Schlüsselworte des Wörterbuches gespeichert. Bei technischen Daten treten in der Regel keine hohen Frequenzen mit gleichzeitig hohen Amplituden auf. Das heißt, der Signalverlauf unterliegt keinen enormen Schwankungen, sodass benachbarte Werte meist dicht beieinander liegen. Da es sich bei Messwerten in der Regel um technische Daten handelt, bei denen die Differenzen zwischen den aufeinanderfolgenden Einzelwerten eher gering ausfallen,

kann hier durch eine geschickte Reorganisation ein Datenstrom erzeugt werden, der relativ viele gleiche Datenketten enthält und damit gut komprimierbar ist.

Das Grundprinzip der Datenreorganisation, welche in *Maglap++* realisiert wurde, ist einfach die Erzeugung einer Datentabelle und seiner Transposition. TT-Algorithmen unterteilen sich in zwei Gruppen:

• **Binär** — die Werte haben eine binäre Darstellung
• **Symbolisch** — die Werte werden in symbolischer Form geschrieben (als Ziffern)

Nachfolgend soll an einem Beispiel (TT-binär) die detaillierte Funktionsweise des TT-Algorithmus demonstriert werden. Der Unterschied zwischen den beiden Gruppen besteht nur in der Darstellung der Zahlen. Das Grundprinzip ist davon unabhängig.

Bemerkung: Für TT-binär werden Integer-Werte verwendet, weil bei Double-Werten das Exponent-Fraction-Zahl-Format verwendet wird, was die Erweiterung der Methode erfordert. Die Realzahlen werden dazu in ganze Zahlen umgewandelt (zB. durch Multiplikation mit 1000000).

Zu Beginn befinden sich alle Werte normalerweise in einem Puffer als Zeile (in Informatik: Array). Mit verschieden Farben werden das nullte, erste, zweite usw. Bit (bis 31 für *int*) markiert (Abb. 9). Zuerst baut man die spezielle Matrix (wie in Abb. 9) auf. Dazu werden alle nullten Bits in der ersten Spalte eingesetzt (erste in zweite usw.).

Dann transponiert man diese erstellte Matrix. Als Ergebnis entsteht eine Matrix (Abb. 10), wo die nullten Bits unserer Zahlen die erste Zeile darstellen und jede weitere Bit-Gruppe entsprechend eine nächste Zeile. Anschließend müssen diese Zeilen in einem Puffer nacheinander gespeichert werden. Anderes gesagt, wird eine gruppierte Bits-Folge erstellt. Weil die Unterschiede zwischen den Werten gering sind, werden sich in diesem Puffer viele gleiche Zahlenfolgen befinden. Diese Daten-Umwandlung verbessert den Kompressionskoeffizienten.

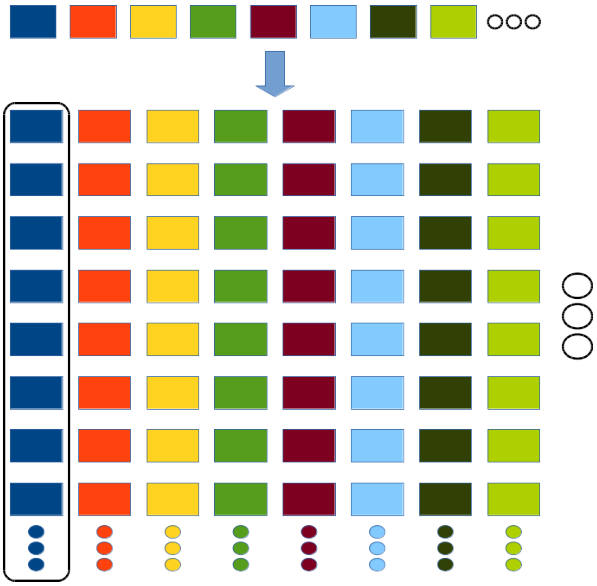


Abb. 9: TT. Erstellung der Matrix

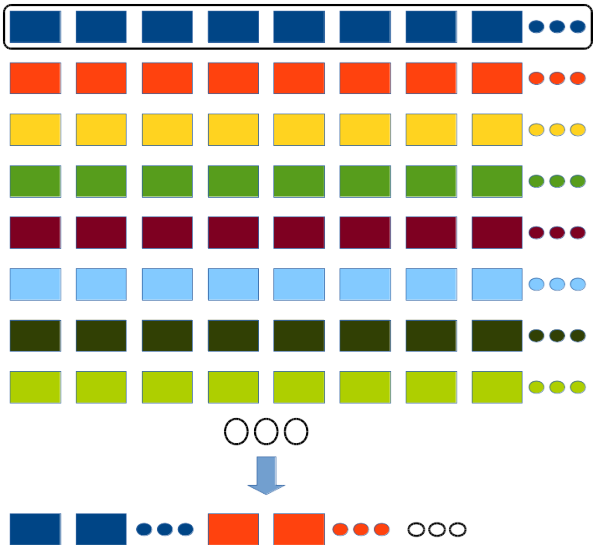


Abb. 10: TT. Erstellung der Ausgangsdaten

4 Test und Verifikation

4.1 Softwaretest

Um die Software auf die Erfüllung der Anforderungen hin zu bewerten, müssen Software-Tests nach den Regeln der Softwaretechnik durchgeführt werden. Das Hauptziel der Tests stellt sich als eine Prüfung des Programms auf korrekte Verarbeitung dar. Aber das ist nicht ganz korrekt. Ein Test muss nicht zeigen, dass ein Programm fehlerfrei funktioniert, sondern er muss die Anzahl der Fehler minimieren. Deswegen kann man das Ziel besser mit der Suche nach Situationen beschreiben, in denen das Produkt fehlerhaft arbeitet. Das Testen selbst ist ein destruktiver Prozess und beginnt mit der

Vermutung, dass Fehler vorhanden sind. Ein Testprozess ist gegensätzlich zum Entwicklungsprozess. Dies ist vor allem zeitaufwendig und kompliziert. In der Zeitplanung kann für das Testen normalerweise ca. 50% der gesamten Entwicklungszeit angesetzt werden. Aus organisatorischen Gründen aber vor allem aufgrund psychologischer Aspekte darf ein Programm nicht vollständig vom Entwickler selbst getestet werden.

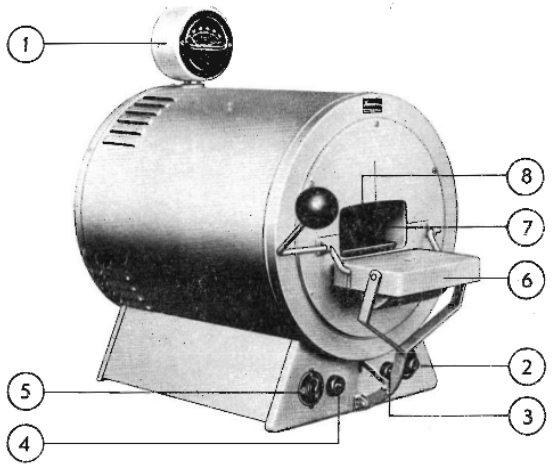
Die notwendigen Tests und Testszenarien wurden durchgeführt. Mit dem Funktionsnachweis wurde die Qualität der Arbeiten sichergestellt. Außerdem wurden die Forschungs-Tests (Experimente) auf Versuchsanlagen mithilfe von *Maglap++* erstellt.

4.2 Muffeloffen-Test

Beschreibung
Für die Verifikation des Programms *Maglap++* wurde ein Test an der Versuchsanlage „Muffelofen MR 170“ (Abb. 11) durchgeführt. Ein „Muffelofen MR 170“ (Temperaturbereich: zwischen ca. 600 und 1000^{o}C) ist ein Standard-Heizgerät. Das Ziel des Tests ist die Temperaturmessung im „Muffelofen“. Während des Versuchs wurde die Herdschale des „Muffelofens“ von ca. 120 auf ca. 500^{o}C erwärmt. Der Versuch lief ca. zwei Stunden.

Aufbau
Die Abbildung 12 zeigt die Struktur des Testszenarios und die notwendigen Einrichtungen. Die Temperatursensoren wurden in dem „Muffelofen“ installiert (in der Herdschale) und mithilfe der Leitungen mit einem SAE-Systemgehäuse „Bedobox“ verbunden. Die analogen Spannungen wurden von der „Bedobox“ in normierte, analoge Signale umgewandelt. Diese wurden dann an die Messwerterfassungskarte des Rechners mit den NI-Treibern (von National Instruments) weitergeleitet, wo sie digitalisiert und über einen Pufferspeicher dem System verfügbar gemacht wurden. Auf diesem Rechner wurde das Programm *Maglap++* als ein Messwerterfassungssystem mit Archivierung und Auswertung gestartet.

Auswetung
Nach ca. 2 Stunden wurde der Versuch beendet, die erfassten Daten gespeichert (Darstellung der Daten, Abb. 13) und ein Auswertungsprotokoll erstellt. Es bestand bei diesem Test technisch keine Möglichkeit parallel ein Referenzprogramm laufen zu lassen. Es ist nicht möglich, dass zwei Programm



1 Temperatur-Anzeigeinstrument
2 Temperatur-Einstellknopf
3 Regel-Kontrollampe (gelb)
4 Haupt-Kontrollampe (rot)
5 Netzschalter
6 Wendetür
7 Herdschale
8 Heizmuffel

Abb. 11: Test Muffelofen, Quelle: o.V. "messerforum.net"

Abb. 12: Test Muffelofen. Struktur

gleichzeitig mit der Messwerterfassungskarte arbeiten können. Auch war es mit der vorhandenen Ausrüstung nicht möglich die Signale zu verdoppeln um sie auch noch auf einem anderen Rechner aufzuzeichnen. Für eine Überprüfung der Messwerte wurde deshalb das analoge Thermometer am Muffelofen (siehe Abb. 11) genutzt. Die erfassten Daten wurden im Auswertungsmodul als Mittelwerte dargestellt. Die Messwerte wurden in Temperaturwerte umgerechnet und mit den abgelesenen Werten des Thermometers verglichen. Daraus konnte geschlussfolgert werden, dass die Daten korrekt erfasst wurden. Ein Eingang der „Bedobox“ war beschädigt, was mit späteren Tests und Überprüfungen nachgewiesen wurde.

4.3 Archivierung

Beschreibung
Um den TDC und die TT-Verfahren zu prüfen, wurden zwei Tests mit „realen“ (Quelle: Magnet- und Fanglagerprüfstand (MFLP)) und generierten Daten durchgeführt.

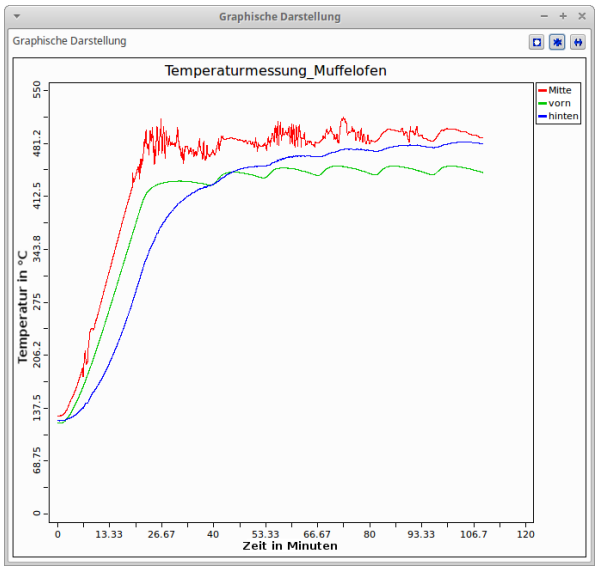


Abb. 13: Test Muffelofen. Datendarstellung

Aufbau

Die Abbildung 14 zeigt die allgemeine Struktur zum Test. Im ersten Fall wurden die Daten mithilfe der Unit übernommen, welche die Werte aus Dateien lesen kann. Diese Daten stellten die Messwerte vom Versuchsstand MFLP dar. Für die zweite Variante wurde die Unit verwendet, welche die Daten generiert. Die Daten hatten die nachfolgende Charakteristik. Dabei steht die erste Zahl für „real“ und Zweite für „generierte“ Daten.

- 10 Dateien (je 120/180 Mb groß)
- 40/60 Sekunden
- 12 Kanäle
- 2500 Werte pro Sekunde

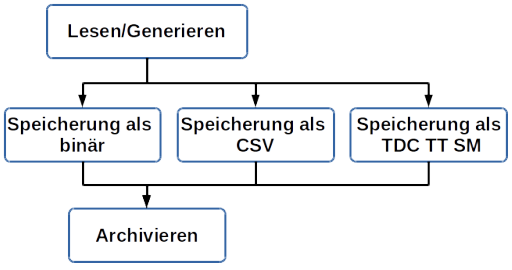


Abb. 14: Archivierung - Teststruktur

Zuerst werden die Daten von einer Quelle erfasst (Dateien/Generator). Dann müssen diese in drei Gruppen als Dateien gespeichert werden, wobei jede Gruppe in einem eigenen Ordner abgelegt wird. Dann werden die Dateien separat mithilfe einem Archivierungsprogramms archiviert. Für diese Aufgabe wurde das Programm „7-Zip“ ausgewählt. Schließlich muss eine Vergleich zwischen den Archiven und den nicht komprimierten Dateien erstellt werden.

Auswetung

Für den Vergleich und die Analyse wurden die Tabellen 1 und 2 erstellt.

	Binär	CSV	TT-SM
real	80%	100%	50%
generiert	80%	100%	50%

Tab. 1: Dateigröße. Versionsvergleich.

Die Tabelle 1 zeigt, wie viel Platz nach der Speicherung im jeweils speziellen Datenformat im Vergleich zur „Originalgröße“ notwendig ist. In dieser Tabelle wurden die Werte in den Zellen mithilfe der Formel 1 erstellt. Für die notwendige Berechnung wurde der Wert „Größe der originalen Dateien“ verwendet:

- Anfangsdateien (Test eins) = 1,2 Gb (10 Dateien, 40 Sekunden pro Datei, 2500 Wert pro Sekunde)
- Generierte Dateien (Test zwei) = 1,8 Gb (10 Dateien, 60 Sekunden pro Datei, 2500 Wert pro Sekunde)

$$Codierungskoeffizient = \\ = \frac{Größe\ nach\ der\ Codierung}{Größe\ der\ originalen\ Dateien} \quad (1)$$

Obwohl es sich jedes mal um die selben Daten handelt, unterscheiden sich die resultierenden Dateien der verschiedenen Formate in ihrer Größe. Dies ist auf die Besonderheiten des jeweils verwendeten Formats zurückzuführen. In der „binären“ Form wurden die Werte zum Beispiel als „Double-Werte“ (8 Bytes) gespeichert. Zum Vergleich, in CSV wurden die Werte als „Text“ gespeichert. Für jeden Wert werden dabei mindestens „10 Bytes“ verwendet. Ein Byte für das Vorzeichen, eins für das Dezimalzeichen, mindestens ein Byte für den ganzzahligen Teil und 6 Bytes für den Bruchteil).

Die Tabelle 2 stellt die Resultate nach der Archivierung dar. Das heißt, jede Zelle zeigt, wie viel

	Binär	CSV	TT-SM
real	12,9%	18,9%	19,3%
generiert	66,6%	31,6%	24,6%

Tab. 2: Archivierung. Versionsvergleich.

Platz nach der Codierung und der Archivierung im Vergleich zur „Originalgröße“ der Anfangsdaten notwendig ist. Mithilfe der Formel 2 wurden diese Werte berechnet.

$$Kompressionskoeffizient = \\ = \frac{Größe\ nach\ der\ Archivierung}{Größe\ der\ originalen\ Dateien} \quad (2)$$

Der Container TDC-TT-SM hat fast den gleichen Koeffizienten für „Real“, wie der Container mit CSV. Aber ersterer speichert die Werte zusammen mit einer Beschreibung, was etwas mehr Platz benötigt. Deswegen sind diese zwei Container als gleich anzusehen. In diesem Test sind die Resultate des Containers TDC-Binär am besten. Im zweiten Test hingegen hat er die schlechtesten Resultate (es wird mehr als der doppelte Speicherplatz benötigt). Der Container TDC-TT-SM lieferte im zweiten Test die beste Resultate.

Für die Analyse der Datenspeicherung sind diese zwei Test zu wenig. Für ein repräsentatives Ergebnis sind mindestens 1000000 Tests mit unterschiedlichen Daten notwendig. Die Archivierung ist eine komplizierte Aufgabe und bislang existiert noch keine allgemeine Lösung mit guten Ergebnissen dazu. Die entwickelten Container sind zwar gut für die Archivierung, aber nur für die Messwertarchivierung.

5 Zusammenfassung und Ausblick

5.1 Zusammenfassung

Maglap++ ist ein gutes Werkzeug zur Zustandsdiagnose und Überwachung magnetgelagerter Maschinen. Außerdem beschränkt sich dieses Programm nicht nur auf Magnetlagertechnik und die genannten Aufgaben. *Maglap++* verwendet einen benutzerdefinierten dynamischen Aufbau, was es ermöglicht, frei kombinierte komplexe Verarbeitungs- und Datenstromketten zu verwirklichen.

Die Verwendung der „Open Source“-Ressourcen entbindet weitestgehend von lizenzrechtlichen Problemen. Der modulare Aufbau ermöglicht es, die Module, das Design und das Programm selbst leicht erweitern und anpassen zu können. Die notwendigen Units zur Messwerterfassung, Datenverarbeitung, Client-Server-Verbindung, Darstellung, Analyse, Datenspeicherung und Zustandsdiagnose wurden erstellt. Die dazu notwendigen Methoden wurden im Quellcode implementiert.

5.2 Ausblick

Maglap++ ist ein gutes Messwerterfassungssystem. Diese Software stellt sich als ein Programm zur Verarbeitung und Analyse von Daten dar. Natürlich liegt das zukünftige Ziel auf der Weiterentwicklung des Konzepts und des Programms und die Implementierung neuer Möglichkeiten in *Maglap++*. Zusätzlich ist eine Erweiterung des Modulkonzepts notwendig. Dazu sind die Weiterentwicklung des Plugin-Systems wichtig. Das Programm muss auch in Zukunft weiter optimiert und an die Fortschritte in der Technik und der Softwaretechnik angepasst werden (Funktionalität, Threads, GUI usw.).