HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

MASTERARBEIT

Herr
Martin Karing

Entwicklung einer .NET™-Softwareumgebung für ein Sensorsystem

2012

Fakultät **Elektro- und Informationstechnik**

MASTERARBEIT

Entwicklung einer .NET™-Softwareumgebung für ein Sensorsystem

Autor:
Martin Karing

Studiengang:
Elektrotechnik

Seminargruppe:
ET11sS-M

Erstprüfer:
Prof. Dr.-Ing. habil. Heinz Döring

Zweitprüfer:
Dipl.-Ing. Norbert Göbel

Mittweida, 2012

Bibliografische Angaben

Karing, Martin: Entwicklung einer .NET™-Softwareumgebung für ein Sensorsystem, 55 Seiten, 28 Abbildungen, Hochschule Mittweida (FH), Fakultät Elektro- und Informationstechnik

Masterarbeit, 2012

Satz: LuaLaTeX

Referat

Diese Masterarbeit soll einen Überblick über die Softwareumgebung geben die für das Messsystem der Lehr- und Forschungsgruppe Optronik an der Hochschule Mittweida entworfen wurde. Dabei sollen die Bestandteile der neuen Softwareumgebung beschrieben werden und die Konzepte verwendet wurden. Außerdem sollen mögliche Anwendungen und Anwendungskonzepte beschrieben werden.

I. Inhaltsverzeichnis

Inhaltsverzeichnis I

Abbildungsverzeichnis II

Akronyme III

Danksagung IV

1 Einleitung 1

1.1 Ausgangslage 1

1.2 Hintergrund der Aufgabenstellung 1

1.3 Aufgabenstellung 1

2 Vorbetrachtung 3

2.1 Beschreibung der aktuellen Softwareversionen 3

2.2 Anforderungen an die neue Software 3

3 Grundlagen 5

3.1 Beschreibung des aktuellen ILM 5

3.2 Beschreibung der aktuellen Kommunikationsschnittstelle 6

3.3 Entwicklungs-Hilfsmittel 7

4 Kommunikationsschicht der neuen Software 13

4.1 Problemdarstellung 13

4.2 Konzeptbeschreibung 13

4.3 Neuerungen im Kommunikationsprotokoll 14

4.4 Implementierung des Kommunikationsprotokolls in .NET™ 15

5 Werkzeugkasten der neuen Software 23

5.1 Konzeptbeschreibung 23

5.2 Hilfsklassen 23

5.3 Hardware-orientierte Werkzeuge 25

5.4 Software-orientierte Werkzeuge 33

5.5 WPF-Anbindung 40

6 Anwendungsentwicklung 45

6.1 Fertige Anwendungen 45

6.2 Konzept für neue Software 46

7 Zusammenfassung und Ausblick 47

Literaturverzeichnis 49

Glossar 51

II. Abbildungsverzeichnis

3.1 Schematischer Aufbau des ILM 5
3.2 Regelkreis des ILM 6
3.3 Kommunikationsschema über RS-232 7
3.4 Kommunikationsschema über USB-HID 7
4.1 Aufbau eines Befehlspaketes für RS-232 15
4.2 .NET™-Implementierung des Kommunikationsprotokolls 16
4.3 Klassendefinition für den Befehl zum Einstellen der Frequenz 17
4.4 Abhängigkeitsdiagramm der Schnittstellen der Abstraktionsschicht 18
4.5 Ablaufdiagramm für die Suche nach einer RS-232-Verbindung 20
5.1 Profilkurve mit SFP LS38-A3S-TC-N-DD 26
5.2 Profilkurve mit SFP F413S17415-D 26
5.3 Klassendiagramm für Frequenzgenerator-Werkzeug 27
5.4 Klassendiagramm für ADC-Werkzeug 28
5.5 Klassendiagramm für SFP-Werkzeug 30
5.6 Klassendiagramm für RTC-Werkzeug 31
5.7 Klassendiagramm für Phasendiskriminator-Werkzeug 32
5.8 Klassendiagramm für LWL-Werkzeug 34
5.9 Profilkurve einer 50m-Strecke 35
5.10 Klassendiagramm für Profil-lesen-Werkzeug 35
5.11 Klassendiagramm für Rampen-Erkennungs-Werkzeug 36
5.12 Abhängigkeit Phasendifferenz zu Ausgabe des Phasendiskriminators 37
5.13 Profilfunktion mit angenäherter Kurve 38
5.14 Klassendiagramm für das Profilfunktion-suchen-Werkzeug 38
5.15 Profilfunktion mit angenäherter Kurve und Arbeitspunkt 39
5.16 Klassendiagramm für das Phasen-Stabilisierungs-Werkzeug 40
5.17 Objekt-Verbindung in WPF 41
5.18 Verbindungsarten in WPF 41
5.19 Konvertierung in WPF 43

III. Abkürzungsverzeichnis

ADC Analog-to-Digital-Converter.

CAN Controller Area Network.

DDM Digital Diagnostics Monitoring.
DDS Direct-Digital-Synthesizer.
DLL Dynamic Link Library.

GPG GNU Privacy Guard.

HID Human Interface Device.
HTTP Hypertext Transfer Protocol.
HTTPS Hypertext Transfer Protocol Secure.

ID Identifikationsnummer.
ILM Integrales Längenmesssystem.

LWL Lichtwellenleiter.

OOP Objektorientierte Programmierung.

PC Personal Computer.
PLL Phase-Locked Loop.
POF Polymeric optical fiber.

RTC Real-Time Clock.

SFP Small Form-Factor Pluggable.
SSH Secure Shell.

USB Universal Serial Bus.

WPF Windows Presentation Foundation.

IV. Danksagung

Ich möchte mich an dieser Stelle bei der gesamten Forschungsgruppe Optronik der Hochschule Mittweida und den Betreuern dieser Masterarbeit, Prof. Dr.-Ing. habil. Heinz Döring und Dipl.-Ing. Norbert Göbel für die umfangreiche Unterstützung bedanken.

Außerdem möchte ich mich bei Steffen Ebersbach, Andreas Grob, Werner Mothes und Stefan Wolf für das Korrekturlesen bedanken.

Ein großer Dank auch an meine Eltern, die mich in der gesamten Studienzeit unterstützt haben.

1 Einleitung

1.1 Ausgangslage

In der Lehr- und Forschungsgruppe Optronik wird seit mehreren Jahren ein Sensormesssystem entwickelt, das vor allem zur Beobachtung von Bodenbewegungen verwendet wird. Dieses Messsystem benutzt das Prinzip der integralen, optischen Längenmessung. Dafür werden Lichtwellenleiter (LWL) in den Boden eingebracht. Die Bewegungen des Bodens werden mechanisch auf den LWL übertragen und verursachen eine Dehnung oder Stauchung des LWL. Diese mechanischen Belastungen äußern sich in einer Veränderung der Laufzeit des Lichts im LWL. Die Änderung wird von dem Messsystem gemessen und dadurch kann die Änderung der Gesamtlänge des LWL gemessen werden.

Die Steuerung jedes Messsystems wird von einem Personal Computer (PC) übernommen. Dieser PC ist über eine RS-232-Verbindung oder eine Universal Serial Bus (USB)-Verbindung mit dem Messsystem verbunden. Auf dem PC läuft eine entsprechende Messsoftware, die das Messsystem steuert und die Daten empfängt, auswertet und speichert.

1.2 Hintergrund der Aufgabenstellung

Eines der größten Probleme in Bezug auf die Hostsoftware besteht darin, dass es sehr viele Varianten der Messsoftware gibt. Diese verschiedenen Versionen sind eigenständige Entwicklungen, die allerdings Kernkomponenten immer gleich oder ähnlich implementieren. Zu diesen Kernkomponenten zählen zum Beispiel die Kommunikation mit dem Messsystem, das Nachregeln der Frequenz und das Auslesen der Messdaten, die das Messsystem aufgezeichnet hat.

Jede Änderung die das integrale Längenmesssystem (ILM) erfährt muss in jede einzelne Version der Software separat eingepflegt werden. Damit muss die gleiche Arbeit mehrfach erledigt werden um alle Softwarevarianten auf dem aktuellen Stand zu halten.

Dieses Problem tritt auf, weil die Kommunikation mit dem ILM um eine USB-Verbindung und eine mit Prüfsummen gesicherte RS-232-Verbindung erweitert wurde. Damit jede Softwarevariante diese neue Möglichkeit nutzen kann, muss jede Variante einzeln und von Hand geändert werden.

1.3 Aufgabenstellung

Die Aufgabenstellung ist es eine Lösung für die PC-Software zu finden, die die im vorherigen Abschnitt genannten Probleme löst. Eine Vorgabe ist, die Erstellung mit Visual Basic.NET™ und damit die Nutzung des .NET™-Framework.

Die Lösung muss den Entwicklungsaufwand, der sich aus Änderungen am ILM ergibt, reduzieren. Das Ziel ist es, dass derartige Änderungen nur noch an einer Stelle in der PC-Software erledigt werden müssen und jedes Programm das mit dem ILM arbeitet diese Änderungen sofort verwenden kann.

Zusätzlich sollte die Software-Lösung bereits die Funktionen enthalten die bereits in nahezu jeder Softwarevariante verwendet werden.

2 Vorbetrachtung

2.1 Beschreibung der aktuellen Softwareversionen

Es gibt eine Reihe von Versionen der Messsoftware, die das ILM nutzt, um Messungen auszuführen. Diese Software-Versionen basieren zum Teil auf Visual Basic 6.0 und zum Teil bereits auf dem .NET™-Framework. Jede Softwarevariante ist dabei eine vollständig eigenständige Entwicklung, die keinerlei Abhängigkeit zu anderen Software-Paketen hat. Darum sind grundlegende Komponenten der Software, wie zum Beispiel die Kommunikation mit dem Messsystem, in jeder Anwendungsvariante neu implementiert. Deshalb ist es sehr kompliziert, Änderungen in das Kommunikationsprotokoll einzufügen, ohne die bestehende Software in ihrer Funktionalität zu stören. Fehler in den Kernfunktionalitäten der Software stellen ein besonderes Problem dar. In den meisten Fällen sind diese Fehler in allen Software-Versionen und müssen einzeln behoben werden. Vor allem bei Software die noch auf Visual Basic 6.0 basiert, sind solche Korrekturen gar nicht mehr möglich, weil diese Entwicklungsumgebung nicht mehr verwendet wird.

Die Pflege der bestehenden Softwareversionen ist insgesamt sehr aufwändig, weil die Software teilweise sehr undurchsichtig geschrieben ist. Das liegt daran, dass oft verschiedene Personen die Software nacheinander, ohne Absprache, entwickelt und weiterentwickelt haben. Außerdem wird bei Softwareversionen, die auf das .NET™-Framework aufbauen, noch fast vollständig die prozedurale Programmierung genutzt wird. Das .NET™-Framework ist aber grundsätzlich auf die Verwendung der objektorientierten Programmierung (OOP) ausgelegt. Darunter leidet die Wartbarkeit und auch die Erweiterbarkeit der Software. Das ist auch einer der Gründe, warum für jedes Problem eine neue Softwarevariante entwickelt wurde. Bestehende Software zu erweitern, um neue Probleme lösen zu können, ist ein größerer Aufwand als die Software neu zu entwickeln.

2.2 Anforderungen an die neue Software

Es sind zwei Lösungsansätze für die Lösung der Aufgabenstellung denkbar. Die erste ist eine komplette Software zu schreiben die jeden Anwendungsfall des ILM abdeckt. Da das ILM allerdings sehr flexibel eingesetzt wird, müsste die Software so viele Anwendungsfälle abdecken können, dass die Bedienbarkeit der Software darunter leiden würde. Außerdem ist eine Software mit sehr vielen verschiedenen Anwendungsfällen sehr schwer zu warten.

Die zweite mögliche Lösung ist es ein zentrales Softwarepaket zu schreiben. Dieses Softwarepaket muss die Elemente enthalten, die jede Software die mit dem ILM arbeitet benötigt. Der Vorteil dieser Lösung ist, das die Flexibilität von verschiedenen Programmen gewährleistet wird. Jedes Programm kann auf eine spezielle Messaufgabe ausgerichtet werden. Die Probleme die durch einen zu großen Funktionsumfang entstehen können damit umgangen werden.

Die grundsätzliche Anforderung an die Software ist, dass sie auf das .NET™-Framework aufsetzt und auf absehbare Zeit mit künftigen Versionen des .NET™-Framework kompatibel ist. Die neue Software soll die Kern- Funktionalitäten in abgegrenzten Klassen anbieten die in anderen Versionen der Software verwendet werden können. Dabei soll die OOP soweit es sinnvoll ist ausgenutzt werden um eine möglichst effiziente Programmierung zu ermöglichen.

Das Softwarepaket soll die komplette Kommunikation mit dem ILM abdecken und einige Werkzeuge anbieten die in vielen Anwendungen verwendet werden können. Für alle Komponenten aus denen sich die Software zusammensetzt müssen entsprechende Schnittstellen entworfen werden, auf die die eigentliche Messsoftware dann zurückgreifen kann. Diese Schnittstellen müssen die eigentlichen Implementierungsdetails verbergen, die für die übergeordnete Anwendung nicht relevant sind. Zum Beispiel ist es für eine Messsoftware nicht relevant ob das Messgerät über eine RS-232- oder über eine USB-Verbindung mit dem Computer verbunden ist. Solche Details sollten in dem Softwarepaket intern behandelt werden. Zu den Werkzeugen, die die Software enthalten soll, zählen unter anderen Klassen die die Steuerung des Frequenzgenerators oder des Analog-Digital-Konverters (ADC) der im ILM integriert ist vereinfachen.

Dazu kommt, dass die Software grundsätzlich in der Lage sein muss mehrere Messsysteme gleichzeitig innerhalb von einem Programm zu verwalten, ohne dass mehrere Instanzen des selben Programms gestartet werden müssen.

3 Grundlagen

3.1 Beschreibung des aktuellen ILM

Das ILM der Lehr- und Forschungsgruppe Optronik ist ein Längenmessgerät für LWL-Strecken. Das Gerät basiert auf der Laufzeitmessung des Lichtes im LWL. Für diese Messung wird ein optisches Signal in den LWL eingekoppelt, das mit einer bestimmten Frequenz moduliert wird. Der LWL ist in einer Schlaufe ausgelegt. Das Licht, das wieder im Gerät ankommt, hat im Vergleich zum ausgesendeten Licht die gleiche Frequenz aber eine verschobene Phase. Diese Phasenverschiebung wird gemessen. Wenn der LWL gedehnt oder gestaucht wird, ändert sich die Phasenverschiebung. Aus dieser Änderung kann die Längenänderung am LWL berechnet werden. Der interne Aufbau des ILM ist schematisch in der Abbildung 3.1 dargestellt.

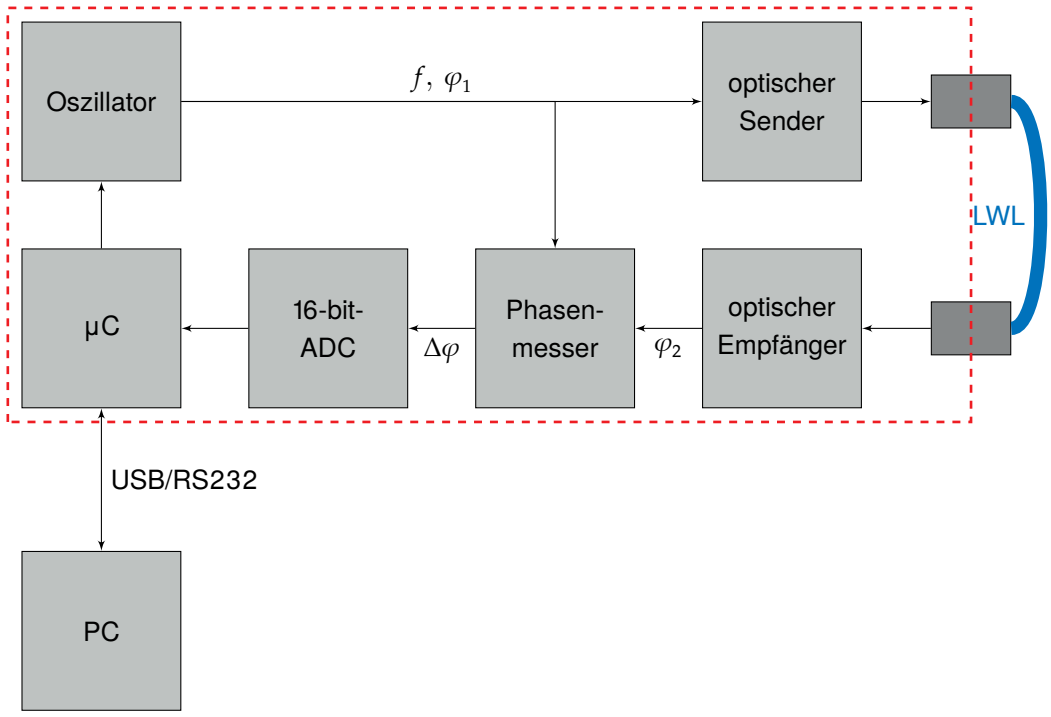

Abbildung 3.1: Schematischer Aufbau des ILM

Die Steuerung und Reglung des Messsystems erfolgen über den angeschlossenen PC. Dieser empfängt die aktuelle Phasendifferenz vom Messsystem. Der Regelalgorithmus für das Messsystem sieht vor, dass die Phasendifferenz immer auf einem konstanten Wert gehalten wird (z. B. 90°). Das wird durch eine Korrektur der Frequenz erreicht, die der Oszillator erzeugt. Der PC berechnet den neuen Frequenzwert und schickt diesen zurück an das Messsystem. Dieser Regelkreis ist in Abbildung 3.2 dargestellt.

Um das elektrische Signal des Oszillators in optische Signale umzuwandeln, werden Small Formfactor Pluggable (SFP)-Module eingesetzt. Diese Module enthalten einen optischen Sender und

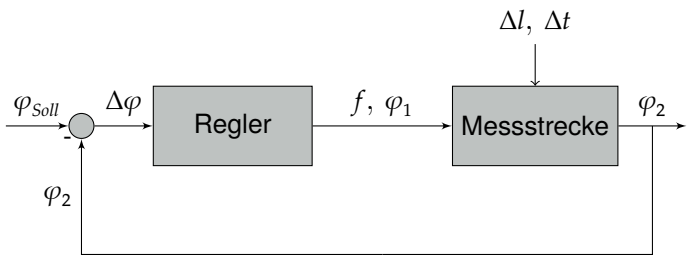

Abbildung 3.2: Regelkreis des ILM

Empfänger. Der Vorteil dieser Module ist, dass sie die notwendigen optischen Komponenten vollständig abdecken. Außerdem sind die SFP-Module durch ihre Verwendung in der Netzwerktechnik sehr weit verbreitet, gut erhältlich und technisch ausgereift. Die SFP-Module können jederzeit ausgetauscht werden, um das Gerät mit anderen LWL-Typen zu benutzen. Das ILM funktioniert problemlos mit allen optischen SFP-Modulen. Es kann Singlemode-, Multimode-Glasfasern und auch polymere optische Fasern (POF) verwenden.

3.2 Beschreibung der aktuellen Kommunikationsschnittstelle

Bei der Betrachtung der Kommunikationsschnittstelle des ILM muss zum einen der gemeinsame Befehlssatz betrachtet werden, als auch die speziellen Implementierungen des Übertragungsprotokolls für USB und RS-232. Obwohl sich beide Kommunikationsschnittstellen den selben Befehlssatz teilen, werden die Daten aufgrund der Besonderheiten der jeweiligen Schnittstelle unterschiedlich zwischen ILM und PC ausgetauscht.

Das Kommunikationsprotokoll basiert darauf, dass jeder Befehl eine eindeutige Identifikationsnummer (ID) hat. Diese Nummer wird sowohl für den Befehl selbst, als auch für die passende Antwort verwendet. Die Kommunikation wird immer durch den PC ausgelöst. Der PC schickt einen Befehl und das ILM schickt eine Antwort, wenn diese vorgesehen ist. Das ILM schickt niemals automatisch Daten.

3.2.1 RS-232

Bei der Übertragung über das RS-232-Protokoll wird immer eine einfache Bytefolge geschickt. Das erste Byte ist die ID des Befehls. Die folgenden Bytes sind die Daten des Befehls. Es ist wichtig, dass sowohl PC als auch ILM die Länge der Befehle und Antworten kennen, weil es keine Start- und keine Ende-Kennung gibt. Dieses Schema ist in Abbildung 3.3 dargestellt.

Das Problem dieser Kommunikation besteht darin, Übertragungsfehler zu erkennen. Es ist praktisch unmöglich, den richtigen Anfang eines Befehls zu finden, wenn die Kommunikation fehlerbehaftet ist. Die Zahlenwerte einer ID können auch innerhalb der Daten des Befehls vorkommen. Wenn

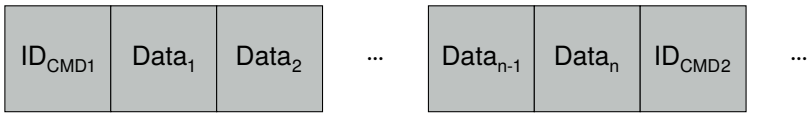

Abbildung 3.3: Kommunikationsschema über RS-232

das Gerät nicht sicher weiß, dass ein Datenbyte eine ID ist, dann kann ein falsches Datenbyte als ID erkannt und die folgenden Bytes völlig falsch interpretiert werden.

3.2.2 USB

Die USB-Verbindung mit dem ILM erfolgt über das Human Interface Device (HID)-Protokoll von USB. In diesem Protokoll ist vorgesehen, dass in gleichen zeitlichen Abständen von minimal $1\,\mathrm{ms}$ die Übertragung von einem Datenpaket erfolgt. Die Größe des Datenpakets wird festgelegt wenn sich das HID-Gerät am USB anmeldet. Das ILM verwendet eine Paketgröße von 8 Byte.

Jeder übertragene Befehl wird daher auf Pakete von je 8 Byte aufgeteilt. Die ID steht immer im ersten Byte eines Paketes gefolgt von den Daten. Wenn ein Befehl kleiner ist als 8 Byte, werden die restlichen Bytes mit Nullen gefüllt. Ist der Befehl länger als 8 Byte, dann werden die restlichen Daten in einem weitem Datenpaket verpackt. In diesem Fall werden alle 8 Byte des folgenden Paketes für Daten verwendet. PC und ILM müssen wissen, welche Länge jeder Befehl hat, damit die verschiedenen Datenpakete richtig verknüpft werden können.

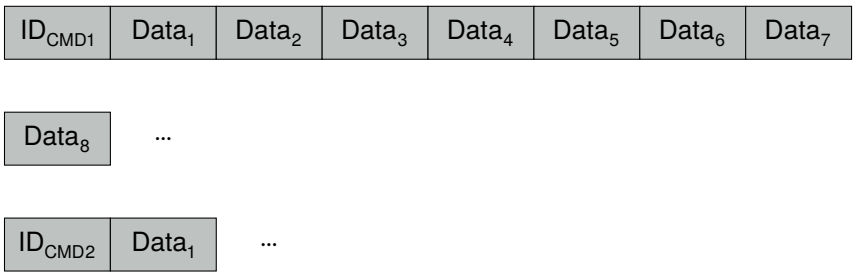

Abbildung 3.4: Kommunikationsschema über USB-HID

Bei dieser Implementierung ist es erheblich leichter, den Anfang eines Befehls zu finden, als bei der RS-232-Implementierung. Der Anfang und damit die ID eines Befehls ist immer das erste Byte eines Datenpaketes. Wenn dieses Byte nicht zugeordnet werden kann, kann das komplette Paket verworfen werden. Übertragungsfehler sind sehr unwahrscheinlich, denn das USB-Protokoll ist generell mit Prüfsummen abgesichert. Wenn ein Datenpaket falsch übertragen wird, wird es durch den USB-Treiber automatisch neu angefordert. Dieses Schema ist in Abbildung 3.4 dargestellt.

3.3 Entwicklungs-Hilfsmittel

Für die Entwicklung der Softwareumgebung wurden einige Technologien verwendet, die bei der Softwareentwicklung innerhalb der Forschungsgruppe Optronik neu sind. Diese werden in diesem

Abschnitt kurz beschrieben.

3.3.1 Versionsverwaltung

Für die Entwicklung der Software wurde das Versionsverwaltungssystem *Git* verwendet. Dieses Versionsverwaltungssystem ist eines der neusten Systeme dieser Art. Git gibt es erst seit 2005 und wurde vom Initiator des Linux-Kernels Linus Torvalds ins Leben gerufen. Git wurde entwickelt, weil es kein alternatives Versionsverwaltungssystem gab, das verteilte Arbeitsabläufe unterstützte, eine sehr hohe Sicherheit gegen Verfälschung bot und sich durch eine hohe Effizienz auszeichnete[13].

Allgemeine Informationen

Git ist stark auf eine nicht lineare Entwicklung ausgelegt. Die Möglichkeit, neue Entwicklungszweige zu erzeugen und mit anderen Entwicklungszweigen zu verschmelzen, ist ein wichtiger Bestandteil der Arbeit mit Git. Git liefert die Werkzeuge, die dafür gebraucht werden mit. Die Arbeit mit zusätzlichen Entwicklungszweigen bietet sich immer dann an, wenn neue Funktionen in eine Anwendung eingebaut werden sollen. Solange die neuen Funktionen noch nicht stabil funktionieren, können sie getrennt von der „Hauptversion“ entwickelt werden. Die Hauptversion kann gleichzeitig weiterentwickelt werden. Wenn die neue Funktion funktioniert, kann sie in die Hauptversion übertragen werden. Alle Änderungen die in dem getrennten Entwicklungszweig durchgeführt wurden, werden dann auf den Zweig der Hauptversion übertragen.

Eine weitere Besonderheit von Git ist, dass es keinen zentralen Server gibt, auf dem die Daten der Versionsverwaltung gespeichert sind. Statt dessen hat jeder Benutzer des Versionsverwaltungssystems eine komplette Kopie des Repositorys. Diese Kopie schließt sowohl die Daten als auch die Versionsgeschichte vollständig mit ein. Technisch gibt es zwischen den verschiedenen Repositorys keinerlei Unterschied. Allerdings gibt es bei vielen Projekten ein Repository, auf das als Referenz verwiesen wird und das als *Haupt*-Repository bezeichnet werden kann.

Der Datentransfer zwischen den Repositorys kann über verschiedene Netzwerkprotokolle oder durch die Übertragung von Patch-Dateien durchgeführt werden. Die Patch-Dateien enthalten die Änderungen am Repository in einer kompakten Form und können mit entsprechenden Programmen überprüft und leicht in das Ziel-Repository übertragen werden. Für den Datentransfer über Netzwerkprotokolle kann das eigene Protokoll von Git, oder eines der gebräuchlichen Protokolle wie Secure Shell (SSH), Hypertext Transfer Protocol (HTTP) oder Hypertext Transfer Protocol Secure (HTTPS) verwendet werden.

Die Versionsgeschichte von Git ist grundsätzlich so aufgebaut, dass der Name eines einzelnen Eintrages eine Art Prüfsumme der kompletten Versionsgeschichte dieses Eintrages ist. Daher ist es nicht möglich, einzelne Einträge in der Versionsgeschichte zu ändern ohne dass sich der Name aller Einträge, die dem geänderten Eintrag nachfolgen, ändert. Zusätzlich kann ein Eintrag in der Versionsverwaltung markiert werden, um beispielsweise einen wichtigen Zwischenstand

der Software zu markieren. Diese Markierungen können zusätzlich mit GNU Privacy Guard (GPG) digital signiert werden.

Anwendung

Da Git ursprünglich für Linux-ähnliche Betriebssysteme entwickelt wurde, muss unter Windows eine entsprechende Portierung verwendet werden. Das Projekt *msysgit* hat sich diesem Problem angenommen und bietet ein leicht installierbares „Git for Windows“ an.

Ein Git-Repository befindet sich auf einem Rechner immer in einem frei wählbaren Ordner. Dabei besteht das Repository in der Regel aus zwei Teilen. Zum einen das eigentliche Repository, das sich im Unterordner `.git` befindet. Der zweite Teil ist die Arbeitskopie, die sich direkt in dem gewählten Ordner befindet. Die Dateien und Ordner, die in der Arbeitskopie erstellt, geändert oder gelöscht werden, können in das Versionsverwaltungssystem eingetragen werden. Bestimmte Dateien in der Arbeitskopie können ignoriert und niemals eingetragen werden. Das ist vor allem für Dateien sinnvoll, die automatisch erzeugt werden. Im Fall der Anwendungsentwicklung trifft das oft auf Dateien zu, die im Zuge der Kompilierung erzeugt werden. Git arbeitet grundsätzlich am besten mit Textdateien. Quellcode-Dateien können daher sehr effizient in Git gespeichert werden. Binäre Dateien, also z. B. kompilierte Programme oder Bilder dagegen, werden von Git schlechter verwaltet. Es gibt auch keine Möglichkeit später Differenzen von diesen Dateien zu betrachten, zumindest nicht mit den Mitteln die Git von Haus aus installiert hat.

Grundsätzlich wird Git immer dann verwendet, wenn ein Projekt geändert wurde, das mit dem System verwaltet wird. Diese Änderungen können eine oder mehrere Dateien umfassen. Sie sollten sich aber immer inhaltlich auf eine konkrete Änderung im Projekt beziehen und nicht mehrere Korrekturen zusammenfassen. Sonst lässt sich die Entwicklungsgeschichte schlecht nachvollziehen. Der Vorgang eine Änderung ins Versionsverwaltungssystem einzutragen nennt sich „commit“ (engl. übergeben). Jede dieser Eintragungen muss in Git mit einem Kommentar versehen werden, der grob beschreiben sollte, was mit dieser Eintragung geändert wurde.

```
git commit -am ”Kommentar-Text”
```

Quellcode 3.1: Git-Befehl zum Eintragen aller geänderter Dateien

Neue Dateien werden nicht automatisch in diese Eintragungen aufgenommen. Das Versionsverwaltungssystem wird diese Dateien als „nicht überwacht“ angeben, aber selbstständig nicht in die Versionsverwaltung aufnehmen. Wenn derartige nicht überwachte Dateien gemeldet werden, kann man diese ignorieren, oder sie in die Versionsverwaltung aufnehmen. Neue Dateien in die Versionsverwaltung aufzunehmen wird als „add“ bezeichnet. Wenn eine solche Datei hinzugefügt wird, wird nicht automatisch eine Eintragung durchgeführt. Die Datei wird nur markiert, um zu zeigen, dass sie Teil der nächsten Eintragung werden kann.

Für den Austausch mit anderen Entwicklern kann das eigene Repository mit den Repositorys anderer Entwickler zusammengeführt werden. Es ist dabei nur möglich, Repositorys zusammenzuführen,

```
git add ziel/datei.txt
```

Quellcode 3.2: Git-Befehl zum Hinzufügen einer Datei

die eine gemeinsame Versionsgeschichte haben. Beim Zusammenführen werden alle Änderungen beider Repositorys eingetragen, die seit dem letzten gemeinsamen Eintrag gemacht wurden. Wenn sich die Änderungen auf verschiedene Dateien, beziehen funktioniert das Zusammenführen automatisch. Bei Konflikten, müssen die Dateien, die in beiden Repositorys unterschiedlich geändert wurden, manuell zusammengeführt werden. Das Zusammenführen der Versionsgeschichte mit anderen Repositorys ist einer der wichtigsten Komponenten von Git. Der Vorgang läuft immer in zwei Schritten ab. Zunächst müssen die Daten des anderen Repositorys empfangen werden. Der Vorgang nennt sich „fetch“. Dabei wird die komplette Versionsgeschichte des entfernten Repositorys übertragen. Diese liegt dann als zusätzlicher Entwicklungszweig vor. Ab diesem Punkt unterscheidet sich das entfernte Repository nicht mehr von einem zweiten lokalen Entwicklungszweig. Der nächste Schritt ist das eigentliche Zusammenführen das als „merge“ bezeichnet wird. Dieser Vorgang gilt sowohl für das Zusammenführen von lokalen Entwicklungszweigen als auch für das Zusammenführen mit entfernten Entwicklungszweigen. Sobald das Zusammenführen gestartet wird, versucht Git, die Änderungen beider Entwicklungszweige automatisch zu vereinen. Wenn dabei Konflikte auftreten, bricht Git den Vorgang ab und ein manuelles Zusammenführen wird nötig. Dass ein entferntes Repository empfangen und dann mit dem eigenen zusammengeführt werden muss, geschieht bei der Arbeit mit Git sehr häufig, wenn mehrere Entwickler an einem Projekt arbeiten. Darum gibt es in Git einen kombinierten Befehl, der empfängt und die empfangenen Daten direkt mit den eigenen zusammenführt. Dieser Befehl nennt sich „pull“. Im Quellcode 3.3 ist zu lesen, wie man das Empfangen und Zusammenführen aufrufen kann. Zusätzlich steht in der ersten Zeile, wie man ein entferntes Repository zu dem Eigenen hinzufügen kann.

```
git remote add name ssh://adresse.zum:ziel/repo.git
git fetch name
git merge name/master
```

Quellcode 3.3: Git-Befehle zum Zusammenführen

Um das eigene Repository zu sichern oder auch anderen zugänglich zu machen, wird das Repository oft auf einen entsprechenden Server hochgeladen. Auf diesem Server wird das Repository vollständig gespeichert. Mit diesen entfernten Repositorys werden meistens die Daten mit anderen Entwicklern ausgetauscht. In den meisten Fällen ist kein direkter Zugriff auf das Repository möglich, das sich auf dem PC des entsprechenden Entwicklers befindet. Der Vorgang, die Änderungen des lokalen Repositorys auf einen Server zu übertragen nennt sich „push“. Dabei wird zunächst überprüft ob in dem entfernten Repository Änderungen sind, die im lokalen Repository noch fehlen. Ist das der Fall wird das Versenden abgebrochen. Diese Änderungen werden sonst überschrieben. Wenn nur das lokale Repository Änderungen hat, die nicht im entfernten Repository sind, dann werden die fehlenden Änderungen an das entfernte Repository übertragen.

Die Verwendung der Kommandozeile, um Git zu steuern, ist Geschmackssache. Es gibt unter Windows zusammen mit msysgit ein Programm mit dem Namen „GitGui“, das die Steuerung von

```
git push name master
```

Quellcode 3.4: Git-Befehle zum Versenden

Git in eine kleine Benutzeroberfläche verpackt und dabei alle gebräuchlichen Funktionen von Git problemlos abdeckt. Außerdem gibt es für viele Entwicklungsumgebungen Erweiterungen, die die Steuerung von Git direkt aus der Entwicklungsumgebung heraus erlauben.

3.3.2 Unit-Tests

Unit-Tests sind Hilfsmittel in der Programmierung, um entwickelte Programme und Bibliotheken ausführlich und automatisiert zu testen. Diese Tests ermöglichen es Probleme die durch Änderungen im Programm auftreten schnell zu finden. Der Aufwand diese Unit-Tests zu schreiben ist allerdings sehr groß.

Ein Unit-Test testet immer eine Funktion in einer Klasse. Die Rückgabewerte und Nebeneffekte der Funktion werden mit verschiedenen äußeren Bedienungen und Parametern getestet. Alle Reaktionen der Funktion die nicht vorgesehen sind, werden als Fehler gemeldet.

Unit-Tests werden ausgeführt sobald das Programm geändert wurde. Ein Programm hat normalerweise sehr viele Unit-Tests die alle Komponenten des Programms testen. Die Tests können entweder manuell ausgeführt werden oder immer dann wenn Funktionen die durch die Tests abgedeckt sind geändert werden.

Das Aufkommen von Unit-Tests hat auch das sogenannte „test driven development“ hervorgebracht. Dabei werden zuerst die Unit-Tests entwickelt und danach die entsprechenden Programmklassen, die diese Unit-Tests erfüllen müssen. Bei der konventionellen Entwicklung werden zunächst die normalen Programmbestandteile entwickelt und danach die dazu passenden Unit-Tests.

Ein einzelner Unit-Test bezieht sich dabei meistens auf den Test einer einzelnen Funktion. Diese wird im Zuge des Tests ein- oder mehrfach aufgerufen um alle oder einen Teil der möglichen Rückgabewerte oder Reaktionen der Funktion zu prüfen. Im einfachsten Fall wird die Funktion mehrfach mit verschiedenen Parametern aufgerufen, bei denen der Rückgabewert bekannt ist, den die Funktion produzieren muss. Dann wird der echte Rückgabewert mit dem erwarteten Rückgabewert verglichen. Wenn die Tests ausgeführt werden, meldet die Entwicklungsumgebung für jeden Test, ob dieser erfolgreich war oder nicht. Im Quellcode 3.5 ist eine Beispiel Funktion und ein Unit-Test dazu zu sehen.

```
Public Module Mathe
    Public Function Mul(a As Integer, b As Integer) As Integer
        Return a * b
    End Function
End Module

Public Module MatheTest
    Public Sub MulTest()
        Assert.AreEqual(1, Mul(1, 1))
        Assert.AreEqual(2, Mul(2, 1))
        Assert.AreEqual(5, Mul(1, 5))
        Assert.AreEqual(36, Mul(6, 6))
    End Sub
End Module
```

Quellcode 3.5: Funktion und Unit-Test Funktion

4 Kommunikationsschicht der neuen Software

Bei der Beschreibung der neuen Kommunikationsschicht geht es vor allem um die Änderung des Kommunikationsprotokolls zwischen ILM und PC. Außerdem wird die Implementierung des Kommunikationsprotokolls in der .NET™-Software beschrieben.

4.1 Problemdarstellung

Die Kommunikationsschicht der neuen Software muss in der Lage sein, zwei Probleme zu lösen, die in den alten Varianten der Software aufgetreten sind.

Das erste Problem ist die Kommunikation über verschiedene Anschlusssysteme. Relevant sind dabei momentan RS-232 und USB. Die Kommunikation über beide Systeme muss aus Sicht der Anwendung auf die gleiche Art funktionieren. In den alten Versionen der Messsoftware ist diese Bedingung nicht erfüllt. Das hat zur Folge, dass die Messsoftware in der Regel nur über RS-232 oder über USB kommunizieren kann, aber nicht über beide Systeme.

Das zweite Problem ist, dass die RS-232-Kommunikation nicht gegen Fehler geschützt ist. Wenn bei der Kommunikation Fehler auftreten, durch Störungen entlang der Übertragungsstrecke, dann müssen diese Fehler erkannt werden. In der momentanen Software gelingt das nicht. Dadurch kann es zu unvorhersehbaren und nicht nachvollziehbaren Fehlern kommen. USB-Verbindungen sind grundsätzlich gegen Fehler abgesichert.

4.2 Konzeptbeschreibung

Um die Forderung nach einer einheitlichen Methode zur Kommunikation mit dem ILM zu gewährleisten, muss eine Schnittstelle verwendet werden, welche die Kommunikation via USB und RS-232 auf den gleichen Nenner bringt.

Das Konzept der Kommunikationsschicht sieht vor, dass eine Anwendung über eine abstrakte Schnittstelle mit dem ILM kommunizieren kann. Diese Schnittstelle soll die eigentliche Implementierung für RS-232 und USB verbergen.

Die Fehlerkorrektur bei der RS-232-Kommunikation muss innerhalb der Schnittstelle gelöst werden. Die Anwendung soll keine Information darüber bekommen, ob über USB oder RS-232 kommuniziert wird.

4.3 Neuerungen im Kommunikationsprotokoll

Die Implementierung der fehlenden Fehlerüberprüfung für RS-232-Verbindungen ist die wichtigste Änderung am Kommunikationsprotokoll. Um sicher über RS-232 kommunizieren zu können, muss man sicher feststellen können wann ein Befehl beginnt, wie viele Daten er beinhaltet und ob die übertragenen Daten fehlerfrei sind.

Da die RS-232-Kommunikation grundsätzlich als endloser Datenstrom ohne definierten Anfang und Ende abläuft, ist die Erkennung des Anfangs eines Befehls ein Problem. Die einzige Möglichkeit besteht darin, ein Byte oder eine Byte-Folge innerhalb des Datenstroms zu suchen, die immer den Start eines Befehls markiert. Die Bedingung an dieses Byte oder die Byte-Folge ist, dass sie im normalen Betrieb selten oder nie innerhalb des Datenstroms vorkommt. Bei den Daten die zwischen PC und ILM ausgetauscht werden, kann jedes Byte vorkommen. Damit ist ein Start-Byte allein nicht ausreichend um einen sicheren Anfang eines Befehls zu markieren. Eine Startkennung, die aus zwei Bytes besteht, ist besser geeignet. Für die Kennung wird die ID des übertragenen Befehls und das Einerkomplement dieser ID verwendet. Für die aktuelle Anzahl der Befehle ergibt sich eine Wahrscheinlichkeit von $0{,}026\%$ (siehe Formel (4.1)), dass eine gültige Kombination dieser zwei Bytes zufällig im Datenstrom auftaucht. Da die Anfangskennung nur gesucht werden muss wenn die Kommunikation zwischen ILM und PC gestört wurde, kann man von einer gesicherten Übertragung ausgehen.

$$\frac{n_{Befehle}}{n_{Bytes}} \cdot \frac{1}{n_{Bytes}} = \frac{17}{256} \cdot \frac{1}{256} = 0{,}026\% \tag{4.1}$$

Für das zweite Problem, die Erkennung von Übertragungsfehlern. Eine Lösung dafür ist jedes übertragene Datenpaket mit einer redundanten Prüfsumme auszustatten. Als Prüfsumme wurde die Standard CRC-16-Prüfsumme gewählt. Der Grund für die Auswahl dieser Prüfsumme war, dass es sich um ein standardisiertes Format handelte. Die Implementierung dieser Prüfsumme war für den Mikroprozessor bereits verfügbar. Das Polynom der Prüfsumme ist in Formel (4.2) dargestellt. Die Prüfsumme wird mit allen Bytes des Befehls mit Ausnahme der Prüfsummen-Bytes selbst berechnet.

$$x^{16} + x^{15} + x^2 + 1 \tag{4.2}$$

Die Darstellung eines kompletten Befehls ist in Abbildung 4.1 zu sehen.

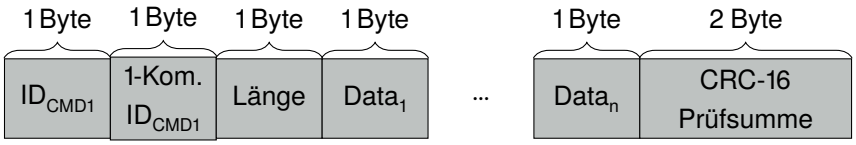

Abbildung 4.1: Aufbau eines Befehlspaketes für RS-232

4.4 Implementierung des Kommunikationsprotokolls in .NET™

Die .NET™-Implementierung der Kommunikationsschnittstelle muss in erster Linie das Problem lösen, dass zwei sehr unterschiedliche Kommunikationsprotokolle RS-232 und USB über die gleiche Schnittstelle funktionieren sollen. Abgesehen von der Tatsache, dass beide Protokolle zur Übertragung von Daten benutzt werden, gibt es keine weiteren Schnittpunkte zwischen den beiden Protokollen.

Die .NET™-Schnittstelle, die den Zugriff auf beide Protokolle ermöglicht, kann sich daher nur darauf beziehen, dass die Befehlsdaten auf irgendeine Weise zwischen PC und ILM ausgetauscht werden. Dabei kann nicht eingeschränkt werden, ob die Daten in Paketen oder in einem endlosen Datenstrom empfangen oder gesendet werden. Genauso kann nicht festgelegt werden, wie eine Verbindung zwischen dem PC und dem ILM hergestellt wird.

Um diese Probleme zu lösen habe ich die Kommunikation in einem mehrschichtigen Modell schrittweise abstrahiert. Der grundsätzliche Aufbau des Modells ist in Abbildung 4.2 zu sehen.

4.4.1 Steuerbefehle und Antwortnachrichten

Der Hauptteil der Kommunikation zwischen dem PC und dem ILM wird durch den Austausch von Nachrichten bestimmt. Die Steuerbefehle und die Antwortnachrichten sind die Komponenten der Kommunikationsschicht mit denen die Anwendung direkt arbeiten muss (siehe Abbildung 4.2).

Jeder Befehl, der an das ILM gesendet werden kann, wurde von mir in einer entsprechenden Klasse vorbereitet. Diese Befehle können mit den Parametern gefüllt und dann abgeschickt werden. Der Befehl selbst ist völlig unabhängig davon, welche Art der Datenübertragung verwendet wird. Für die Anwendung ist die Art der Übertragung nicht wichtig, solange der Steuerbefehl vom ILM empfangen wird.

In Abbildung 4.3 ist die Klassendefinition eines solchen Befehls zu sehen. Dieser Befehl erlaubt es, die Frequenz des Frequenzgenerators im ILM zu ändern. Dazu kann über die Eigenschaft `Frequency` des Befehls eine Frequenz eingestellt werden, die dann an das ILM geschickt wird.

Die Antwortnachrichten sind genauso aufgebaut wie die Steuerbefehle. Der Unterschied besteht nur darin, dass diese Objekte nicht von der Anwendung sondern von der Empfangsschnittstelle

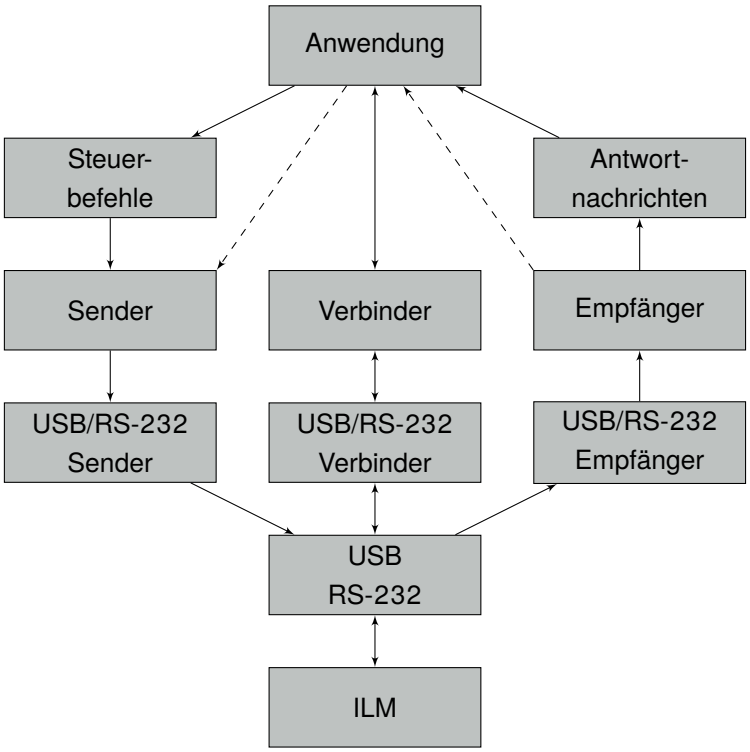

Abbildung 4.2: .NET™-Implementierung des Kommunikationsprotokolls

erzeugt werden. Jede mögliche Antwort, die das ILM senden kann, ist in einer passenden Klasse definiert. Nachdem ein Objekt dieser Klasse erzeugt und mit den empfangenen Daten gefüllt wurde, enthält es keine Informationen mehr darüber, über welche Schnittstelle es empfangen wurde. Sie enthält nur noch die Daten, die das ILM gesendet hat. Diese Daten können dann von der Anwendung nach Bedarf ausgewertet werden.

Wie in der Abbildung 4.2 dargestellt, soll eine Anwendung nur auf die vorgefertigten Befehlssätze zugreifen müssen, um mit dem Gerät zu kommunizieren. Diese Befehle übernehmen bereits die Kodierung der Daten die an das ILM gesendet werden sollen. Wenn die Befehle geändert werden müssen, die zwischen PC und ILM ausgetauscht werden, dann funktioniert das ohne Veränderung der einzelnen Anwendungen, weil die Hilfsklassen für die Befehle direkt geändert werden können. Außerdem reduziert sich durch die Verwendung dieses Modells der Entwicklungsaufwand einer neuen Anwendung, da dort nicht mehr die Kommunikation implementiert werden muss.

Mit den Steuerbefehlen und den Antwortnachrichten ist die Schnittstelle zur Kommunikation mit dem ILM für die Anwendung geschaffen.

4.4.2 Verbinder, Sender und Empfänger

Die Abbildung 4.2 stellt die zweite Schicht der Kommunikationsimplementierung mit Verbinder, Sender und Empfänger dar. Auf diese Komponenten muss die übergeordnete Anwendung in einigen wenigen Fällen zugreifen.

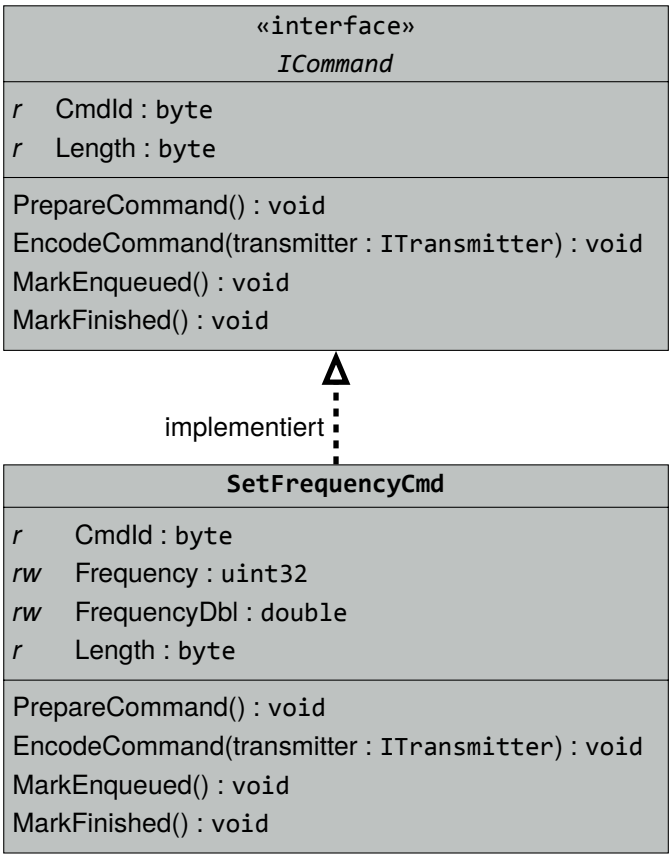

Abbildung 4.3: Klassendefinition für den Befehl zum Einstellen der Frequenz

Der Verbinder stellt die eigentliche Verbindung mit dem ILM her. Mit Hilfe dieser Klasse können die Schnittstellen des PC der Reihe nach oder auch gezielt abgefragt werden. Diese Verbinder-Klassen suchen ein ILM das mit dem PC verbunden ist. Sobald eine Verbindung hergestellt wurde, bietet diese Klasse auch die Möglichkeit, die Verbindung wieder zu trennen.

Die Sender- und Empfänger-Klassen haben zwei Aufgaben. Zum einen verbergen sie die eigentlichen Implementierungen zum Senden und Empfangen. Zum anderen erzeugen sie eine Zwischenschicht, die in getrennten Threads arbeitet. Damit verbergen diese Klassen die Eigenschaften der dahinterstehenden Kommunikationsimplementierungen. Das stellt sicher, dass das Senden und Empfangen von Daten niemals den Ablauf der übergeordneten Anwendung blockieren kann. Die eigentliche Aufgabe dieser Klassen besteht darin, die Befehle und Antworten anzunehmen, zwischenzuspeichern und dann in einem abgetrennten Thread wieder freizugeben. Das eigentliche Senden und Empfangen wird von diesen Klassen nicht durchgeführt.

Diese Zwischenschicht stellt eine klare Abtrennung der protokollabhängigen Implementierungen dar. Außerdem bietet sie der Anwendung begrenzt die Möglichkeit die Verbindungen zu steuern. Das Modell dieser drei Komponenten ist in Abbildung 4.4 dargestellt.

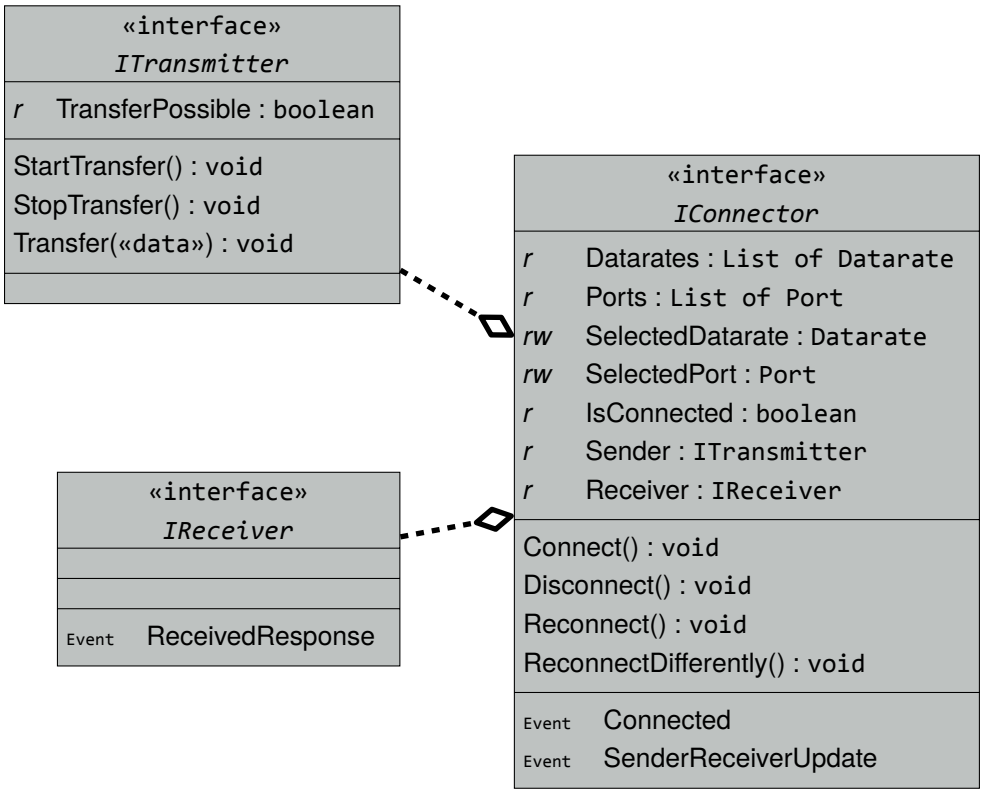

Abbildung 4.4: Abhängigkeitsdiagramm der Schnittstellen der Abstraktionsschicht

4.4.3 RS-232-Verbinder, -Sender und -Empfänger

RS-232 ist ein Übertragungsstandard, der sehr einfach aufgebaut ist und leicht in Betrieb genommen werden kann. Der Standard definiert das die Daten in Form von *Wörtern* übertragen werden müssen. Diese bestehen aus fünf bis neun Bits. Wie diese Wörter ausgewertet werden ist allerdings nicht definiert. Grundsätzlich ist ein RS-232-Datenstrom endlos. Es gibt keinen definierten Start und kein definiertes Ende. Die Kommunikation mit dem ILM läuft allerdings in Form von Datenpaketen ab. Die Lösung diese Datenpakete abzugrenzen ist in Abschnitt 4.3 auf Seite 14 beschrieben.

Aber auch mit dieser Einteilung in Pseudo-Datenpakete stellt die Kommunikation über RS-232 im Schema der Kommunikationsschicht eine Herausforderung dar. RS-232 definiert keine Übertragungsgeschwindigkeit, die verwendet werden muss. Es gibt eine Reihe von Geschwindigkeiten, die im Standard definiert sind und höhere Geschwindigkeiten die gebräuchlich sind. Wenn die Einstellung der Übertragungsgeschwindigkeit nicht bei beiden Kommunikationspartnern gleich ist, dann schlägt die Kommunikation fehl. Darüber hinaus bietet RS-232 nicht die Möglichkeit, festzustellen was für ein Gerät mit dem PC über RS-232 verbunden ist. Bei machen Implementierungen ist es nicht einmal möglich festzustellen, ob überhaupt ein Gerät mit dem RS-232-Anschluss verbunden ist oder ob der Anschluss frei ist.

Diese Einschränkungen sorgen dafür, dass es keine effiziente Lösung zum Herstellen einer Verbindung gibt. Entweder der Anwender kennt den Anschluss und die Parameter des Gerätes, das

angeschlossen ist, oder es muss ausprobiert werden. Der Aufwand, eine Verbindung herzustellen, wird mit dem Ablaufdiagramm in Abbildung 4.5 auf Seite 20 deutlich. Wenn keine genaueren Einschränkungen für die Verbindungsparameter vorliegen, muss dieser komplette Ablauf für die Suche nach einer gültigen Verbindung durchlaufen werden. Da die meisten ILM mit einer Datenrate von $57{,}6\,\mathrm{kBaud}$ arbeiten, werden für die Datenrate erst alle Ports durchsucht, bevor die nächste Datenrate versucht wird.

Der RS-232-Sender hat dann die Aufgabe eine bestehende RS-232-Verbindung zu nutzen, um die Steuerbefehle zu übertragen, die von der übergeordneten Anwendung erzeugt werden. Wie die Daten in Pakete verpackt werden, wurde in Abschnitt 4.3 bereits beschrieben. Die Datenübertragung selbst übernimmt die Implementierung des seriellen Anschlusses, die im .NET™-Framework enthalten ist.

Der RS-232-Empfänger muss ständig den Datenstrom abhören, der über die RS-232-Schnittstelle empfangen wird. Sobald er eine Anfangskennung eines Befehls erkennt, beginnt die Aufzeichnung der Daten. Wenn ein Befehl vollständig empfangen wurde, wird die Prüfsumme verglichen. Nur wenn die berechnete Prüfsumme mit der Prüfsumme die zusammen mit dem Befehl übertragen wurde übereinstimmt, wird der Befehl akzeptiert. Ansonsten werden alle empfangenen Daten verworfen. Wenn der Befehl akzeptiert wurde, wird anhand der ID die entsprechende Klasse der Antwortnachricht ausgesucht. Die empfangenen Daten, die den Inhalt der Nachricht darstellen, werden an die neu erzeugte Instanz dieser Klasse übergeben. Wenn die Instanz die Daten akzeptiert, ist die Verarbeitung des Befehls abgeschlossen. Die Antwortnachricht wird zur Veröffentlichung weitergegeben und der RS-232-Empfänger wartet auf die nächste Nachricht aus dem Datenstrom.

4.4.4 USB-Verbinder, -Sender und -Empfänger

Die komplette USB-Kommunikation setzt auf der Schnittstelle auf, die in meiner Praktikumsarbeit entstanden ist[7].

Die Herstellung einer Verbindung und damit die Arbeit des USB-Verbinders ist sehr einfach, im Vergleich zu dem Aufwand beim Herstellen einer RS-232-Verbindung. Am USB sind alle Geräte mit einer eindeutigen Hersteller- und Produkt-ID ausgestattet. Anhand dieser Kennung können alle ILM-Geräte am USB gesucht werden. Nachdem ein ILM gefunden wurde, kann eine Verbindung zu dem ILM angefordert werden. Diese Verbindung wird durch den HID-Treiber von Windows zur Verfügung gestellt. Wenn diese Verbindung getrennt wird, weil z. B. das USB-Kabel abgezogen wurde, wird die Anwendung davon automatisch informiert. Die Aufgabe des USB- Verbinders beschränkt sich im wesentlichen darauf, den USB abzufragen, ob ein entsprechendes Gerät vorhanden ist und dann die Verbindung anzufordern.

Wie erwähnt verwendet die Verbindung zum ILM via USB die HID-Klasse von USB. Der große Vorteil bei der Verwendung dieser Klasse ist, dass praktisch jedes Betriebssystem, das USB unterstützt, auch einen HID-Treiber mitliefert. Damit entfällt die aufwändige Entwicklung und Auslieferung eines

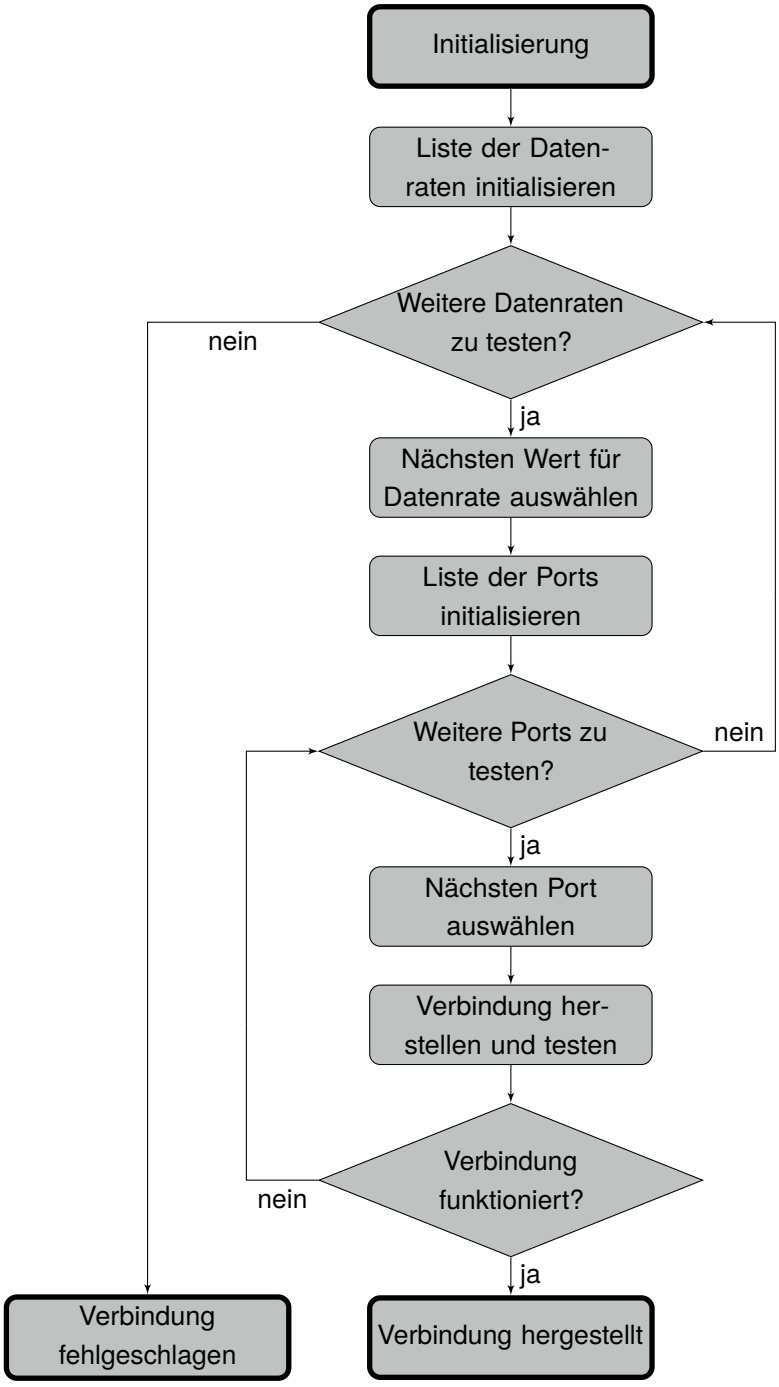

Abbildung 4.5: Ablaufdiagramm für die Suche nach einer RS-232-Verbindung

Treibers.

Für die Übertragung von Daten verwendet das HID-Protokoll Datenpakete, die mit einer festen Größe und in regelmäßigen Abständen zwischen dem PC und dem ILM ausgetauscht werden[14]. Sowohl für den USB-Empfänger als auch für den USB-Sender entsteht dadurch die Bindung an diese Paketgrößen. Diese Bindung stellt kein Problem für Datenpakete dar, die kleiner sind als die HID-Pakete. In diesem Fall wird das restliche Paket einfach mit Nullen aufgefüllt. Für Steuerbefehle oder Antwortnachrichten, die mehr Daten übertragen als in ein HID-Paket passt, müssen die Daten auf mehrere Pakete aufgeteilt werden. Diese Besonderheit ist das einzige Problem, das bei der Implementierung der USB-Anbindung zu lösen war. USB benötigt keine zusätzlichen Prüfsummen wie RS-232, weil die Verbindung vom Treiber mit Prüfsummen gesichert ist. Es ist keine exakte Startkennung notwendig, weil jeder Befehlsanfang mit dem Anfang eines HID-Paketes gleich ist.

5 Werkzeugkasten der neuen Software

5.1 Konzeptbeschreibung

Der Werkzeugkasten umfasst eine Reihe von Hilfsklassen, die die Entwicklung von neuer Messsoftware erheblich beschleunigen und vereinfachen soll. Sie enthalten bestimmte Funktionalitäten, die in vielen Programmen für die Steuerung des ILM verwendet werden.

Diese Werkzeuge sind auf bestimmte Komponenten des ILM bezogen, wie zum Beispiel auf den ADC. Andere Komponenten beziehen sich auf bestimmte Abläufe, die in vielen Programmen verwendet werden, wie das Aufzeichnen der Phasenwerte über einen Frequenzbereich.

5.2 Hilfsklassen

Zusätzlich zu den Hilfsklassen, die bereits im Zuge meiner Praktikumsarbeit [7] und meiner Bachelorarbeit [8] entwickelt wurden, gibt es vor allem für die direkte Arbeit mit dem ILM eine Reihe von neuen Hilfsklassen.

5.2.1 Physikalische Größen

Die meisten vom ILM gelieferten Werte haben einen physikalischen Bezug. Die Berechnungen die für die Arbeit mit dem ILM notwendig sind, verwenden alle diese physikalischen Größen. Um diese Berechnungen zu vereinfachen, wurden Hilfsklassen entwickelt, die je eine physikalische Einheit repräsentieren.

Diese Hilfsklassen werden vor allem in den Anwendungen verwendet, die auf die Softwareumgebung aufsetzen. Darum ist es wichtig das sich diese Klassen ähnlich verhalten, wie normale Zahlenwerte im .NET™-Framework. Dadurch werden unnötige Hürden bei der Softwareentwicklung vermieden. Außerdem müssen die Hilfsklasse zu gebräuchlichen Rechenoperationen verwendet werden können.

Jede der Klassen speichert den Wert der physikalischen Größe in der physikalischen Grundeinheit. Die Klasse, die die Stromstärke speichern kann, speichert die Stromstärke immer in Ampere (A).

Die Werte vom ILM werden meistens nicht in ihren Grundgrößen geliefert. Bei jeder Klasse, die eine physikalische Einheit repräsentiert, kann eine Einheit der Größe beim Erstellen angegeben werden. Dies sind bei den Stromwerten Ampere (A), Milliampere (mA), Mikroampere $(\mu\mathrm{A})$ und Nanoampere (nA). Für die Speicherung wird der eingegebene Wert intern dann automatisch immer in Ampere gespeichert.

Für die Ausgabe des Wertes gilt das gleiche. Auch hier kann die Einheit angegeben werden, in der der Zahlenwert benötigt wird. Die Klassen führen die entsprechenden Umrechnungen automatisch durch. Die Auswahl der Einheiten ist die gleiche wie beim Erstellen.

Die dritte Funktion der Klassen ist die Textausgabe. In vielen Fällen ist es notwendig, dass die physikalischen Größen in der Benutzeroberfläche oder im Protokoll gespeichert werden. Bei der Textausgabe kann ähnlich wie bei der Wertausgabe die Basis angegeben werden. Die Basis kann explizit oder automatisch festgelegt werden. Automatisch wählt die Klasse immer die Basis, mit der der Wert am besten dargestellt werden kann. Diese Funktion eignet sich besonders für Werte, die sich in einem großen Wertebereich, über mehrere Zehnerpotenzen hinweg, ändern können.

Zahlenwerte im .NET™-Framework sind immer vergleichbar, sortierbar und unveränderlich. Die Klassen, die die physikalischen Größen speichern, haben diese Eigenschaften ebenfalls. Damit können sie ähnlich wie `Integer`- oder `Double`-Werte benutzt werden.

Der größte Vorteil dieser Klassen besteht in ihrer Operatoren-Überladung. Dabei wurden die Rechenoperationen (Addieren, Subtrahieren, ...) für diese Klassen so erweitert, das man sinnvolle Kombinationen von Einheiten miteinander verrechnen kann. Es ist z. B. möglich, zwei Ströme zu addieren. Mit dieser Operation ist das Resultat wieder ein Strom. Einen Strom, mit einer Größe, die keine Einheit hat (`Double`), zu summieren ist dagegen nicht möglich. Bei Multiplikationen ist es nur möglich, eine einheitenlose Größe mit dem Strom zu multiplizieren, um wieder einen Strom zu erhalten. Zwei Ströme zu multiplizieren ist nicht möglich.

Auch wenn diese Berechnungen bei weitem nicht alles abdecken, was mathematisch möglich wäre, bieten sie doch eine gute Basis für die Berechnungen mit dem ILM. Sie sorgen in Berechnungen dafür, dass Größen und Einheiten klar ersichtlich sind.

Darüber hinaus bieten diese Klassen eine einfache und effektive Möglichkeit, die physikalischen Größen zu speichern und anzuzeigen, die vom ILM aufgezeichnet werden.

5.2.2 Fortschrittsüberwachung

Große Probleme bereiten in vielen Anwendungen Vorgänge, die sehr lange dauern. Wenn ein Programm nicht anzeigt, dass es einen Vorgang bearbeitet, neigt der Benutzer oft dazu, seine Eingaben zu wiederholen. Einige Vorgänge, die von dem Softwarepaket ausgeführt werden, dauern sehr lange. Deshalb ist es nötig, dass eine Fortschrittsüberwachung möglich ist.

Für diesen Zweck wurde der `ProgressTracker` geschaffen. Dieser Klasse muss regelmäßig der Fortschritt des überwachten Vorgangs gemeldet werden. Bei einer Änderung des Fortschritts sendet die Klasse ein Ereignis an die Anwendung. Mit Hilfe dieses Ereignisses besteht dann die Möglichkeit eine grafische Fortschrittsanzeige in der Anwendung anzuzeigen. Für einzelne Aufgaben ist diese Funktionalität ausreichend. Bei komplexen Aufgaben mit Teilaufgaben, die nacheinander abgearbeitet werden, ist diese Funktionalität allerdings nicht komfortabel. Für diesen Fall bietet

die Klasse die Möglichkeit, weitere Klassen des gleichen Typs als Kinderelemente einzufügen. Damit kann der „Eltern“-`ProgressTracker` nicht mehr direkt mit einem Fortschritt aktualisiert werden. Allerdings überwacht er den Fortschritt aller Kinderelemente und veröffentlicht den kompletten Fortschritt über das entsprechende Ereignis.

So deckt die Klasse in der Form alle Voraussetzungen für die Fortschrittsüberwachung der einzelnen Werkzeuge des Softwarepaketes ab. Vor allem bei der Überwachung der Profilanalyse hat sich dieses Hilfsmittel als sehr vorteilhaft herausgestellt.

5.3 Hardware-orientierte Werkzeuge

Für die Steuerung des ILM selbst wurden einige Klassen erstellt, die sich um die Steuerung der einzelnen Hardwarekomponenten kümmern. Diese Klassen werden im weiteren als Hardware-orientierte Werkzeuge bezeichnet.

Jedes dieser Werkzeuge hat die Aufgabe die Steuerung einer Hardwarekomponente des ILM zu übernehmen. Die Anforderungen an diese Klassen sind abhängig von der Hardwarekomponente. Jede Klasse prüft selbst, ob der Datenaustausch mit der Hardwarekomponente richtig funktioniert.

Wenn z. B. Übertragungsfehler auftreten und ein Befehl an das ILM verloren geht, muss die befehlsgebende Klasse diesen Befehl erneut senden, ohne dass die übergeordnete Anwendung eingreifen muss.

Dieser Ansatz hat sich bei der Analyse der Anforderungen als der beste herausgestellt. Der größte Vorteil besteht in der klaren Definition der Klassen. Jede Klasse muss sich nur um eine Hardwarekomponente kümmern. Alle Aktionen in Bezug auf diese Hardwarekomponente können über diese Klassen ablaufen.

Die Kommunikationsschnittstelle (Abschnitt 4) macht es nicht notwendig weitere Klassen für die Steuerung des ILM zu verwenden. Allerdings handelt es sich bei der Steuerung der Hardwarekomponenten um Vorgänge, die in jeder Software in gleicher Form durchgeführt werden müssen. Im Vergleich zu der alten Variante Messsoftware zu entwickeln, ist der Aufwand zum Erstellen einer Anwendung viel geringer und weniger fehleranfällig, wenn solche Werkzeuge verwendet werden können.

5.3.1 Frequenzgenerator

Das ILM verwendet in seinen verschiedenen Versionen unterschiedliche Frequenzgeneratoren, die sich in ihren Parametern unterscheiden. In den älteren Versionen des ILM werden Direct-Digital-Synthesizer (DDS) verwendet, um die notwendigen Frequenzen zu erzeugen. Diese DDS können Frequenzen bis $37{,}5\,\mathrm{MHz}$[4] erzeugen. In neueren Varianten des ILM wird eine Phasenregelschleife (PLL) verwendet um die notwendigen Frequenzen zu erzeugen. Diese Schaltkreise

arbeiten zwischen 10 MHz und 280 MHz, 810 MHz und zukünftig bis zu 1,4 GHz[11]. Die zweite Einschränkung für die Frequenz entsteht durch das SFP-Modul. Diese Module haben verschiedene maximale Datenübertragungsraten und damit verschiedene maximale Frequenzen. SFP-Module mit 1,25 Gbit/s können mit den Frequenzen, die von dem DDS oder der PLL erzeugt werden, problemlos arbeiten. Allerdings gibt es auch SFP-Module, deren maximale Datenübertragungsrate mit 100 Mbit/s angegeben ist. In diesem Fall liegt die maximal übertragbare Frequenz deutlich unter den Frequenzen, die durch die Frequenzgeneratoren erzeugt werden können. In Abbildung 5.1 sieht man die Profilkurve des SFP-Moduls LS38-A3S-TC-N-DD. Dieses SFP hat eine angegebene maximale Datenübertragungsrate von 155 Mbit/s[5]. Im Profil ist zu erkennen, dass das SFP-Modul bis etwa 400 MHz richtig arbeitet. Bei höheren Frequenzen wird die Qualität der Signalübertragung zunächst deutlich schlechter. Ab etwa 565 MHz bricht die Übertragung komplett zusammen. Als Vergleich erkennt man in Abbildung 5.2 das Profil des SFP-Moduls F413S17415-D, das mit 1,25 Gbit/s[9] angegeben ist. Dieses SFP-Modul zeigt im gleichen Frequenzbereich keine Probleme.

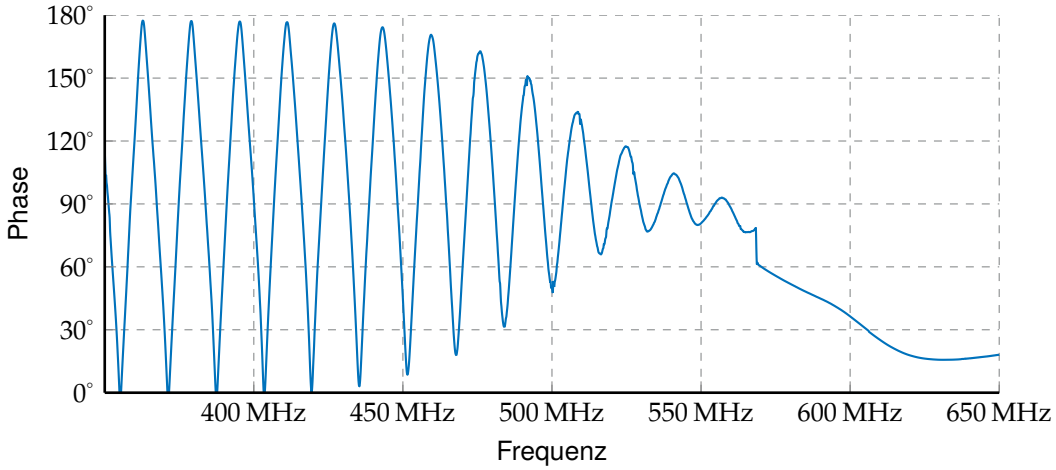

Abbildung 5.1: Profilkurve mit SFP LS38-A3S-TC-N-DD

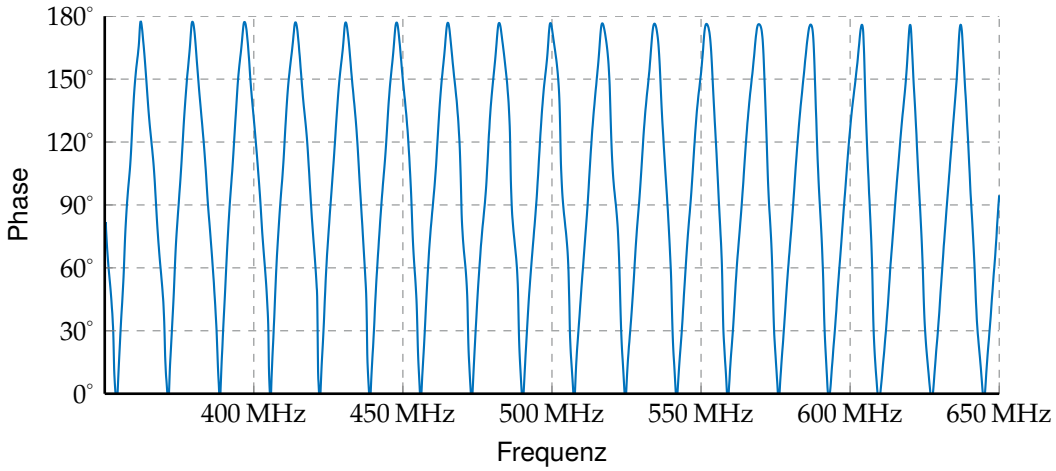

Abbildung 5.2: Profilkurve mit SFP F413S17415-D

Über die beiden genannten Einschränkungen hinaus ist die Implementierung des Werkzeugs für

die Steuerung des Frequenzgenerators sehr einfach. Das Werkzeug bietet zwei Eigenschaften an, mit denen die Grenzfrequenzen ausgelesen werden können. Eine weitere Eigenschaft setzt oder liest die aktuelle Frequenz. Das vereinfachte Klassendiagramm ist in Abbildung 5.3 zu sehen. Das Werkzeug unterstützt zusätzlich die Möglichkeit effektiv, an Komponenten der Windows Presentation Foundation (WPF) angebunden zu werden. Details dazu sind in Abschnitt 5.5 auf Seite 40 beschrieben.

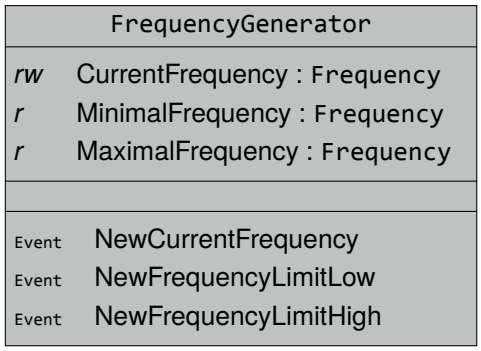

Abbildung 5.3: Klassendiagramm für Frequenzgenerator-Werkzeug

5.3.2 Analog-Digital-Wandler

Obwohl die unterschiedlichen Versionen des ILM intern verschiedene ADC verwenden, ist die Schnittstelle um die ADC immer gleich. Der gravierende Unterschied zwischen den verschiedenen ILM-Versionen ist die Referenzspannung des ADC. Beide verwendete ADC ermöglichen die Referenzspannung extern zuzuführen und skalieren den Wertebereich dann zwischen der analogen Masse und der Referenzspannung[1][12]. Alle weiteren Berechnungen verwenden die Spannungswerte, die der ADC aufzeichnet. Der ADC liefert allerdings 16-Bit-Werte. Diese Werte müssen in die entsprechenden Spannungswerte umgerechnet werden. Die Formel dafür ist in Formel (5.1) zu sehen. Wobei V_{AGND} als $0\,\mathrm{V}$ angenommen wird und N der digitale Ausgabewert des ADC ist. N_{min} und N_{max} sind die minimalen und maximalen Werte des digitalen Wertes. N_{min} ist bei den verwendeten ADC 0 und N_{max} ist $2^{16} - 1$[1][12].

$$U_{Analog} = \frac{N - N_{min}}{N_{max} - N_{min}} \cdot \left(U_{Ref} - U_{AGND}\right) + U_{AGND} \tag{5.1}$$

Der richtige Wert für U_{ref} ist in jedem ILM hinterlegt und wird zu Beginn der Verwendung des ADC-Werkzeuges automatisch vom ILM angefordert. Sobald der Wert vorliegt, werden alle $16\,\mathrm{bit}$ breiten Werte, die vom ADC empfangen werden, automatisch in die entsprechenden Spannungswerte umgerechnet. Dieser Spannungswert ist für weitere Rechnungen besser geeignet, weil er unabhängig vom ADC der im ILM verbaut wurde ist.

Weiterhin kann der ADC auf mehrere Weisen ausgelesen werden, um neben dem analogen

Spannungswert zusätzliche Informationen zu erhalten. Der Informationsgehalt der erweiterten Lesefunktionen ergibt sich erst, wenn der ADC den Auftrag erhält, mehrfach zu mitteln. Die aktuelle Implementierung im ILM unterstützt bis zu 255 Messungen des ADC. Das gemittelte Ergebnis wird zurück an den PC geschickt. Diese Methode der Mittlung ist viel schneller, als jeden Wert einzeln an den PC zu schicken, denn der größte Zeitanteil bei einer Messung geht an die Datenübertragung zwischen ILM und PC. Wenn das ILM mehrere Messungen ausführt und den Mittelwert bildet, fallen zusätzliche Informationen an. Zum einen der Rest der Mittlung und zum anderen maximale und minimale Digitalwerte, die bei den Messungen aufgezeichnet wurden. Die verschiedenen Lesemethoden des ADC-Werkzeugs erlauben es, nur den gemittelten Wert, zusätzlich den Rest der Mittlung oder den minimalen und maximalen Digitalwert auszulesen. Wenn der Rest der Mittlung mit ausgelesen wird, erhöht sich die Auflösung des ausgelesenen Wertes um $8\,\mathrm{bit}$. Die Formel zur Berechnung der analogen Spannung wird, wie in Formel (5.2) gezeigt, erweitert. Wenn der minimale und maximale Digitalwert zusätzlich übertragen wird, kann ein Bereich ausgelesen werden, in dem der analoge Wert schwankt. Damit ist eine Aussage möglich, wie stark verrauscht die analoge Spannung ist. Wenn diese Grenzwerte nicht ausgelesen werden, sind die entsprechenden Felder im ADC-Werkzeug immer gleich der gemittelten Spannung.

$$U_{Analog} = \frac{N + \frac{N_{Rest}}{256} - N_{min}}{N_{max} - N_{min}} \cdot \left(U_{Ref} - U_{AGND}\right) + U_{AGND} \tag{5.2}$$

In Abbildung 5.4 ist das vereinfachte Klassendiagramm dieses Werkzeugs zu sehen. Die Klasse erlaubt es neue Werte für jeden Kanal anzufordern und liefert entsprechende Ereignisse, sobald neue Spannungswerte für den Kanal vorliegen.

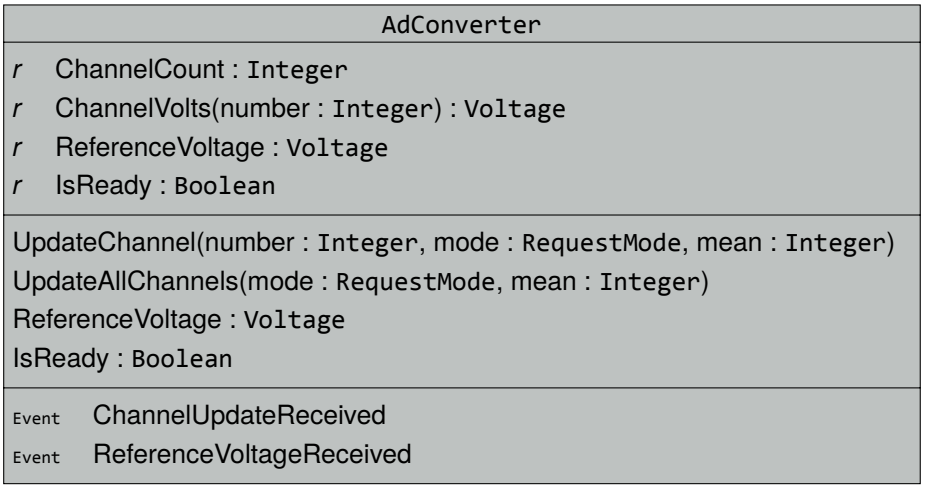

Abbildung 5.4: Klassendiagramm für ADC-Werkzeug

5.3.3 Small-Formfactor-Pluggable

Das SFP ist nur in den neusten Varianten des ILM verbaut und liefert Diagnosedaten des optischen Senders und Empfängers. Allerdings unterstützen nicht alle SFP-Module das digital diagnostics monitoring (DDM). Das Software-Werkzeug muss also feststellen, ob das ILM überhaupt ein SFP-Modul hat und auslesen ob dieses SFP-Modul DDM unterstützt. Die meisten SFP-Module haben außerdem einen Informationsspeicher, in dem Informationen zur Art des SFP-Moduls zu finden sind. Unter anderem lässt sich von dort aus auch die Seriennummer des SFP-Moduls auslesen. Mit dieser identifiziert sich das SFP-Modul eindeutig.

Die Diagnostik-Funktionen des SFP-Moduls bestehen grundsätzlich aus zwei Teilen. Das sind die 16-Bit-Werte des ADC, der im SFP integriert ist. Der zweite Teil besteht aus einer Reihe von Kompensationswerten. Diese werden verwendet um die Werte des ADC in physikalische Werte umzurechnen. Außerdem gibt es noch die Information, ob ein SFP intern oder extern kalibriert ist. Bei intern kalibrierten SFP-Modulen muss keine Umrechnung der DDM-Messwerte durchgeführt werden, weil die DDM-Komponente die Umrechnung selbstständig intern ausführt. Bei extern kalibrierten SFP-Modulen müssen die DDM-Messwerte mit den Kompensationswerten korrigiert werden.

Das DDM stellt fünf Messwerte zur Verfügung: die Stromaufnahme der Sendediode oder des Sendelasers, die optische Ausgangsleistung, die optische Eingangsleistung, die interne Temperatur und die aktuelle Versorgungsspannung [10]. Die Berechnungen für die Kompensation der Werte sind in den Formeln (5.3) bis (5.7) angegeben. Außer bei der optischen Eingangsleistung wird immer eine lineare Kompensation mit Anstieg und Offset benutzt. Bei der optischen Eingangsleistung wird ein Polynom vierten Grades verwendet.

$$I_{Bias} = \left(I_{Slope} \cdot N_{txCurrent} + I_{Offset}\right) \cdot 2 \qquad \left[I_{Bias}\right] = \left[\mu\mathrm{A}\right] \tag{5.3}$$

$$W_{Out} = \frac{W_{Out_{Slope}} \cdot N_{outPwr} + I_{Out_{Offset}}}{10} \qquad \left[W_{Out}\right] = \left[\mu\mathrm{W}\right] \tag{5.4}$$

$$W_{In} = \frac{\sum_{i=0}^{4} W_{In_i} \cdot N_{inPwr}^{i}}{10} \qquad \left[W_{In}\right] = \left[\mu\mathrm{W}\right] \tag{5.5}$$

$$T_{Internal} = \frac{T_{Slope} \cdot N_{temp} + T_{Offset}}{256} \qquad \left[T_{Internal}\right] = \left[^{\circ}\mathrm{C}\right] \tag{5.6}$$

$$U_{Supply} = U_{Slope} \cdot N_{voltage} + U_{Offset} \cdot 100 \qquad \left[U_{Supply}\right] = \left[\mu\mathrm{V}\right] \tag{5.7}$$

Diese Kompensation wird von dem Software-Werkzeug automatisch angewendet, wenn die entsprechenden Werte von der Software angefordert werden. Dazu werden alle Kompensationsdaten einmalig eingelesen. Die Messwerte des internen ADC können von der Software auf Anfrage aktualisiert oder automatisch in regelmäßigen Abständen neu angefordert werden. Sobald neue Werte empfangen wurden, löst das Werkzeug entsprechende Ereignisse aus. Das Werkzeug

unterstützt zusätzlich die Möglichkeit, effektiv an Komponenten der WPF angebunden zu werden. Details dazu sind in Abschnitt 5.5 auf Seite 40 beschrieben. Abbildung 5.5 zeigt den Aufbau der Klasse.

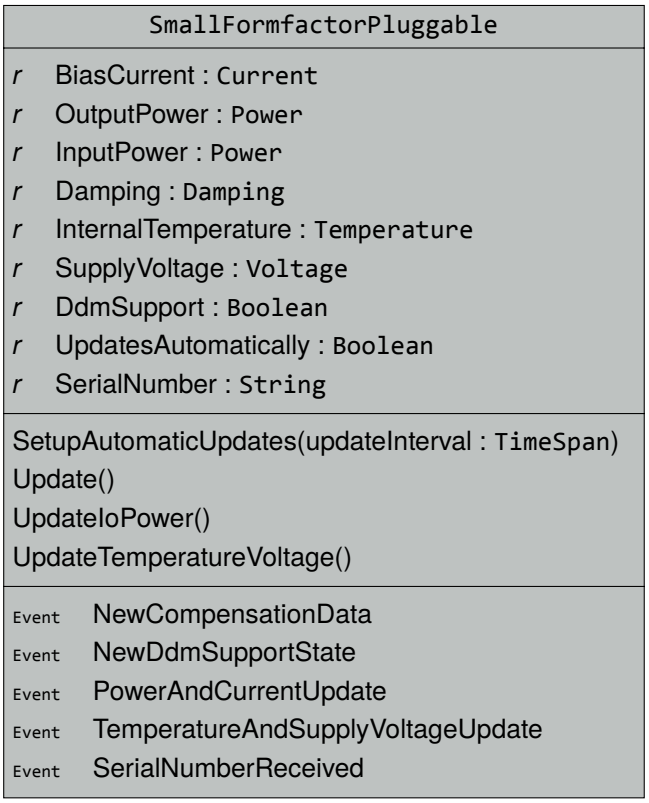

Abbildung 5.5: Klassendiagramm für SFP-Werkzeug

5.3.4 Echtzeituhr

Die Echtzeituhr (RTC) ist in den neueren Versionen des ILM verbaut und unterstützt in Zukunft eine autarke Funktionsweise. Das Software-Werkzeug liest und schreibt die aktuelle Uhrzeit. Außerdem warnt es, wenn die RTC im ILM von der Uhrzeit des PC zu sehr abweicht. Die RTC liefert neben der eigentlichen Uhrzeit die Information, ob die Uhrzeit aktuell noch gültig ist. Die Uhrzeit wird immer dann ungültig, wenn die RTC von der Stromversorgung getrennt wurde. Wenn sowohl die Hauptstromversorgung als auch die Zusatzstromversorgung ausfallen, kann die RTC nicht mehr weiterlaufen. Die Gültigkeit der Uhrzeit ist dann nicht mehr sichergestellt. Die Zusatzstromversorgung wird von einem 1-F-Kondensator geliefert.

Das Software-Werkzeug muss in der Lage sein, diese Informationen auszulesen und entsprechend zur Verfügung zu stellen. Da das RTC-Zählwerk bereits in die Einheiten der Uhrzeit unterteilt ist, ist keine zusätzliche Umrechnung nötig. Die RTC reagiert auf spezielle Daten, wie Schaltjahre, automatisch.

Wie in Abbildung 5.6 gezeigt, stellt die Klasse eine Eigenschaft zur Verfügung, über die die Zeit der RTC sowohl gelesen als auch verändert werden kann. Wenn ein neuer Zeitwert von der RTC

gelesen wird, gibt das Werkzeug diese Information über entsprechende Ereignisse weiter. Intern speichert die Klasse nicht die aktuelle Zeit der RTC, sondern die Abweichung zur lokalen Zeit. Damit kann die laufende Zeit in der Anwendung dargestellt werden, ohne dass ständig neue Werte von der RTC im ILM angefordert werden müssen. Außerdem kann eine maximale Abweichung eingestellt werden, die gültig ist. Wenn diese Abweichung überschritten wird, gibt die Klasse eine Warnung als Ereignis aus. In diesem Falle sollte die Uhr neu gestellt werden. Auch dieses Werkzeug kann an WPF-Komponenten gebunden werden. Details dazu sind in Abschnitt 5.5 auf Seite 40 beschrieben.

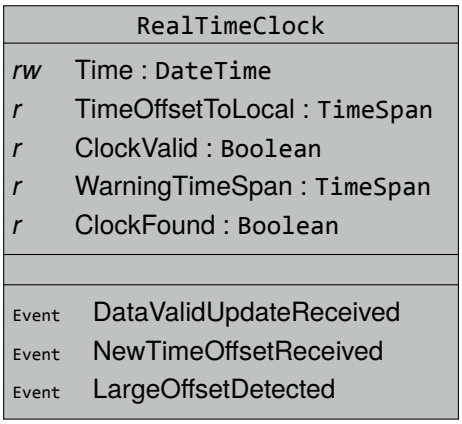

Abbildung 5.6: Klassendiagramm für RTC-Werkzeug

5.3.5 Phasendiskriminator

Der Phasendiskriminator nimmt unter den Hardware-orientierten Werkzeugen eine gesonderte Rolle ein. Diese Komponente liefert keine digitalen Daten, welche durch das ILM gelesen oder durch den PC ausgewertet werden könnten. Der Phasendiskriminator vergleicht das Ausgangs- und das Eingangssignal des ILM und gibt die Phasenabweichung als analogen Spannungswert an den ADC weiter. Bei älteren Versionen des ILM misst der Phasendiskriminator zusätzlich noch die Leistungsdifferenz zwischen den beiden Signalen und gibt diesen Dämpfungswert ebenfalls als analogen Spannungswert an den ADC. Bei den neuen Versionen enthält diese Dämpfungsmessung keine gültigen Daten mehr, weil das SFP-Modul in diesen Systemen dafür sorgt, dass der Pegel des empfangenen Signals immer gleich ist.

Die Aufgabe des Software-Werkzeugs für den Phasendiskriminator ist die Umrechnung der Spannungswerte in die vom Phasendiskriminator gemessene Phasenabweichung oder die Dämpfung. Diese Spannungswerte werden von dem Werkzeug für den ADC zur Verfügung gestellt. Die eigentliche Datenquelle für dieses Werkzeug ist damit der ADC und nicht der Phasendiskriminator direkt. Zusätzlich ist diesem Werkzeug noch die Temperaturüberwachung des Phasendiskriminators zugeordnet. Da der Phasendiskriminator sehr stark von der Temperatur beeinflusst wird, befindet sich im ILM ein Temperatursensor, der mechanisch Kontakt zum Gehäuse des Phasendiskriminators hat. Der Temperatursensor ist in allen ILM ein TMP36. Die Formel (5.8) für die Berechnung des Temperaturwertes kann dem Datenblatt entnommen werden [3]. Für die Berechnung der

Phasendifferenz und der Dämpfung können die Formeln (5.9) und (5.10) verwendet werden[2].

$$T = U_{ADC} \cdot 100\,\mathrm{K/V} - 50\,^{\circ}\mathrm{C} \tag{5.8}$$

$$\Delta\varphi = \frac{U_{ADC}}{1{,}8\,\mathrm{V}} \cdot (-180^{\circ}) + 180^{\circ} \tag{5.9}$$

$$D = \frac{U_{ADC}}{1{,}8\,\mathrm{V}} \cdot 60\,\mathrm{dB} - 30\,\mathrm{dB} \tag{5.10}$$

Diese Berechnungen führt das Software-Werkzeug automatisch durch, wenn es für den ADC neue Spannungswerte für die entsprechenden analogen Kanäle empfängt. Außerdem prüft es, ob die Dämpfungswerte über den Phasendiskriminator oder das SFP zur Verfügung stehen.

In Abbildung 5.7 ist der Aufbau der Klasse zu sehen. Die Funktionen, um neue Werte anzufordern, leiten diesen Aufruf dabei nur an den entsprechenden Kanal des ADC-Werkzeuges weiter. Die Klasse gibt entsprechende Ereignisse aus, wenn der ADC neue Spannungswerte auf einem Kanal empfangen hat, der einem der Werte des Phasendiskriminators zugeordnet ist. Auch dieses Werkzeug kann an WPF-Komponenten gebunden werden. In Abschnitt 5.5 auf Seite 40 befinden sich die Details zur Anbindung an WPF-Komponenten.

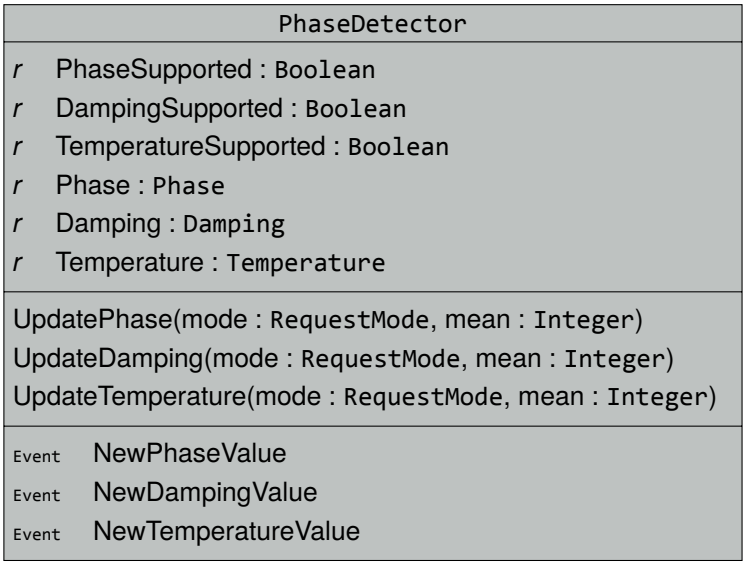

Abbildung 5.7: Klassendiagramm für Phasendiskriminator-Werkzeug

5.3.6 Lichtwellenleiter

Dem Werkzeug für den LWL kommt eine besondere Bedeutung zu. Die Eigenschaften des LWL können vom ILM nicht ausgewertet werden und müssen per Hand in die Anwendung eingetragen werden. Dieses Werkzeug verwendet die Profildaten, die mit den Funktionen erstellt wurden, die in Abschnitt 5.4 beschrieben werden. Mit Hilfe dieser Daten ist es möglich, die Länge des LWL

zu berechnen. Die Genauigkeit dieser Berechnungen ist stark davon abhängig, wie exakt die Parameter des LWL bekannt sind. Die Formel (5.11) wird zur Berechnung der Länge des LWL genutzt. Dabei ist c die Lichtgeschwindigkeit im Vakuum, n der Brechungsindex des LWL, k der Verseilfaktor, t die Periode der verwendeten Profilfunktion und t_0 die Laufzeitkorrektur, die zur Kompensation der LWL-Länge im Gerät und zur Kompensation der Laufzeit der elektrischen Signale sowie der Wandlungszeit verwendet wird. Mit Ausnahme der Periode der Profilfunktion (t) müssen alle Werte per Hand eingestellt werden. Die Qualität der kompletten Berechnung basiert auf der Richtigkeit dieser Einstellungen.

$$l = \frac{c}{n \cdot k} \cdot \left(t + t_0\right) \tag{5.11}$$

Neben der absoluten Länge (l) des LWL kann dieses Werkzeug auf Basis der aktuellen stabilisierten Frequenz die Änderung der Länge des LWL bestimmen. Die Berechnung dieser Änderung basiert auf der Differenz zwischen der aktuellen Frequenz und einer „Start"-Frequenz bei der die Längenänderung als 0 angenommen wird. Die Formel (5.12) wird zur Berechnung verwendet.

$$\Delta l = l \cdot \frac{f_{start}}{f} - l \tag{5.12}$$

Da auch die Längenänderung Δl von der Ausgangslänge l abhängig ist, haben die Einstellmöglichkeiten für die Berechnung der Ausgangslänge auch auf diese Berechnung einen drastischen Einfluss.

Die entsprechende Klasse ist in Abbildung 5.8 dargestellt. Sie ermöglicht es, die Parameter des LWL einzustellen und jederzeit die Referenzfrequenz f_{start} zu ändern, um einen Nullpunkt festzulegen. Für die Werte der Längenänderung unterstützt die Klasse die einfache Anbindung an WPF, wie sie in Abschnitt 5.5 auf Seite 40 beschrieben wird.

5.4 Software-orientierte Werkzeuge

Bei den Software-orientierten Werkzeugen handelt es sich um Hilfsmittel, die keinen direkten Bezug zu Hardware-Komponenten haben. Die existierenden Werkzeuge dieser Art sind Hilfsmittel, um das Profil des ILM einzulesen und auszuwerten.

Das Profil des ILM ist grundsätzlich eine Menge von Phasenwerten bei verschiedenen Frequenzen des internen Frequenzgenerators. Die Abhängigkeit zwischen Phasenwerten und Frequenzen wird maßgeblich durch die Laufzeit des Lichts im LWL und durch die Länge des LWL beeinflusst.

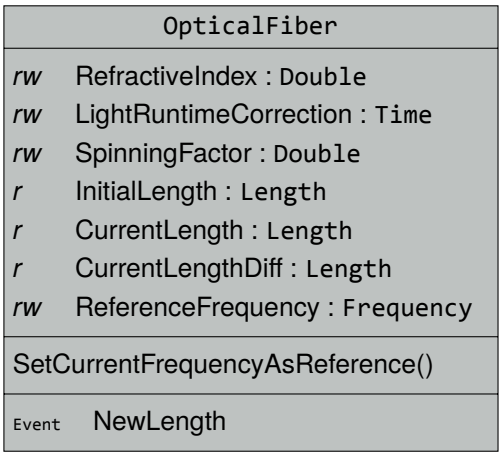

Abbildung 5.8: Klassendiagramm für LWL-Werkzeug

5.4.1 Profil einlesen

Das Werkzeug zum Einlesen der Profildaten verwendet die Werkzeuge für die Steuerung des Frequenzgenerators (Abschnitt 5.3.1) und des Phasendiskriminators (Abschnitt 5.3.5), um das Profil nach einer Strategie einzulesen. Die einfachste Strategie ist das lineare Einlesen des Profils. Dabei wird innerhalb eines Frequenzbereichs eine festlegbare Anzahl von Messwerten linear verteilt aufgenommen. Die Einstellung der Strategie ist der problematische Teil der Profil-Analyse. Wenn der Frequenzbereich mit zu vielen Schritten abgetastet wird, dauert das Ausnehmen des Profils sehr lange. Ein Profil mit 2000 Punkten aufzuzeichnen dauert etwa $90\,\mathrm{s}$. Für 500 Punkte sind es noch etwa $20\,\mathrm{s}$. Wichtig ist außerdem der Frequenzbereich, der eingelesen wird. Die Abhängigkeit zwischen Phase und Frequenz ist stark von der Laufzeit des Lichts abhängig und damit von der Länge des LWL. Bei sehr kurzen Kabellängen kann ein kompletter Phasendurchgang über einen Frequenzbereich von vielen MHz ausgedehnt sein. Bei großen Kabellängen erstreckt sich ein Phasendurchgang nur über einige kHz. Die Wahl des Frequenzbereichs und die Anzahl der Schritte muss also in Abhängigkeit von der Kabellänge getroffen werden. In Abbildung 5.9 ist der Effekt zu sehen, wenn ein Profil mit zu wenigen Messpunkten aufgenommen wird. Bei der Profilanalyse werden beide Kurven mit Sicherheit völlig unterschiedliche Werte ergeben. Aber nur die Messkurve, die mit 500 Messwerten aufgenommen wurde, kann am Ende richtige Werte liefern. Eine Profilkurve die mit zu wenigen Messpunkten aufgenommen wurde, kann nicht immer sicher erkannt werden.

Bei den Einstellungen für das Messprofil ist zwischen der Messsicherheit und der benötigten Zeit abzuwägen. Mehr Messpunkte erhöhen die Sicherheit, dass das Profil richtig aufgezeichnet wurde, brauchen aber auch deutlich mehr Zeit.

Die Klasse, die als Werkzeug für das Aufzeichnen des Profils dient, ist einfach gestaltet. Nachdem der Lesevorgang gestartet wurde, steuert die Klasse den ADC und den Phasendiskriminator automatisch, um alle Punkte aufzuzeichnen. Sobald der Vorgang beendet wurde, wird ein entsprechendes Ereignis ausgelöst. Der Aufbau der Klasse ist in Abbildung 5.10 zu sehen.

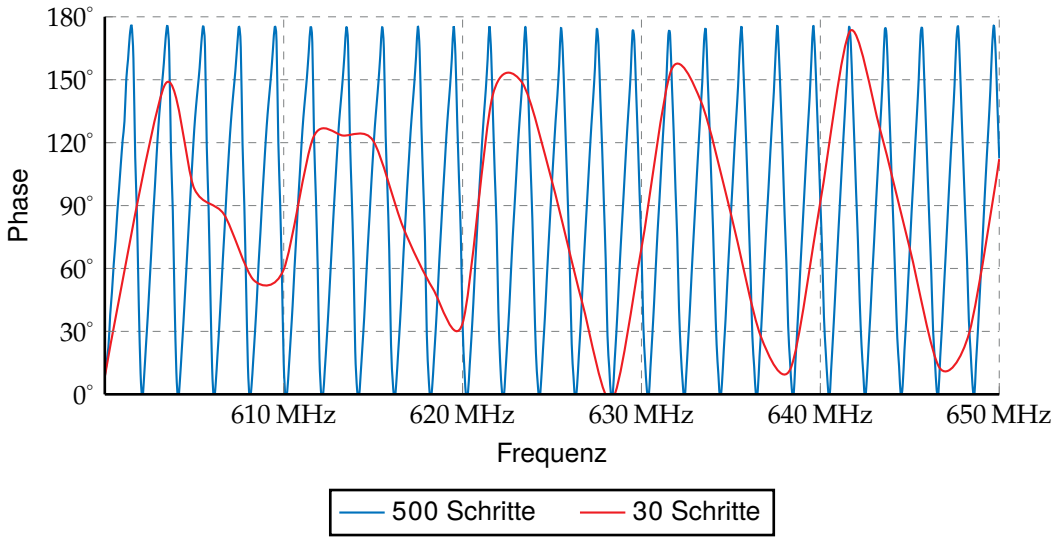

Abbildung 5.9: Profilkurve einer 50m-Strecke

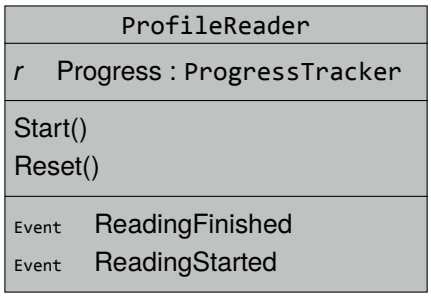

Abbildung 5.10: Klassendiagramm für Profil-lesen-Werkzeug

5.4.2 Profil-Rampen suchen

Nachdem das Profil eingelesen wurde, muss es analysiert werden. Das Ziel ist es, eine Sinus- oder eine Dreieckfunktion zu erzeugen, die möglichst wenige Abweichungen zu dem Profil aufweist. Um eine entsprechende Funktion möglichst schnell zu erzeugen, müssen zunächst die Suchparameter abgesteckt werden. Der Phasenbereich, in dem sich das Profil bewegt, ist immer zwischen 0° und 180°. Die Abhängigkeit zwischen Phasenänderung und Frequenzänderung schwankt aber abhängig vom Messaufbau stark. Eine Methode die Suchparameter für $\Delta f/\Delta\varphi$ zu bestimmen ist die Erkennung der Rampen auf der Messkurve. Die entsprechende Hilfsklasse für diesen Vorgang liest das Profil geordnet nach den Frequenzen ein. Dabei wird der aktuelle Phasenwert mit dem Vorgängerwert verglichen und die aktuelle Tendenz der Profilkurve bestimmt. Wenn die Tendenz der Profilkurve umschlägt, ist eine vollständige Rampe erkannt.

Danach kann für jede Rampe der Anstieg der Kurve in diesem Bereich berechnet werden. Das wird in einem getrennten Schritt gemacht, weil die Berechnungen für jede erkannte Rampe gleich sind und sich gegenseitig nicht beeinflussen. Unter dieser Maßgabe ist es möglich, die Berechnungen parallel ablaufen zu lassen, um die Leistung aktueller PC-Systeme entsprechend zu nutzen. Um die Werte für die Anstiege zu berechnen, wird eine lineare Regression nach der Methode der

kleinsten Quadrate ausgeführt nach der Formel (5.13).

$$M = \frac{\sum_{i=0}^{n}\left(f_i \cdot \varphi_i\right) \cdot n - \sum_{i=0}^{n}\left(f_i\right) \cdot \sum_{i=0}^{n}\left(\varphi_i\right)}{\sum_{i=0}^{n}\left(f_i^2\right) \cdot n - \left(\sum_{i=0}^{n}\left(f_i\right)\right)^2} \qquad [M] = \left[\frac{^\circ}{\mathrm{Hz}}\right] \tag{5.13}$$

Für die Bestimmung der eigentlichen Profilfunktion erzeugt diese Hilfsklasse die Suchparameter in einem Bereich vom minimalen bis zum maximalen berechneten Anstieg. Zur Sicherheit wird dieser Bereich nach oben und unten noch etwas erweitert. In Abbildung 5.11 ist das Klassendiagramm der Klasse zu sehen, die diese Funktionalitäten enthält. Nach der Analyse aller Rampen und der Berechnung der Anstiege generiert die Klasse automatisch die Suchparameter für die eigentliche Funktionsbestimmung nach den oben genannten Methoden. Die Suchparameter werden dann mit einem Ereignis an das Programm zurück übergeben.

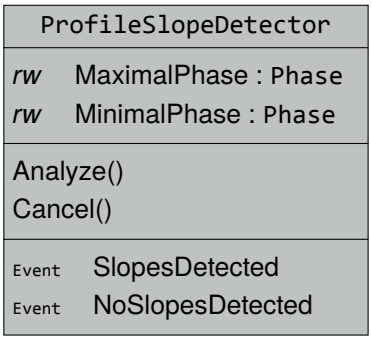

Abbildung 5.11: Klassendiagramm für Rampen-Erkennungs-Werkzeug

Dieser Zwischenschritt ist für die Bestimmung der Profilfunktion eigentlich unnötig, da die Funktion auch ohne die Informationen aus der Rampen-Suche gefunden werden kann. Allerdings ermöglicht dieser Zwischenschritt, die Suchparameter für die eigentliche Suche nach der Profilfunktion massiv einzuschränken. Die besseren Suchparameter erlauben dann eine größere Zeiteinsparung als für die Erkennung der Rampen aufgewendet wurde.

5.4.3 Profilfunktion suchen

Der letzte Schritt der Profilanalyse ist die Ermittlung einer Funktion, welche dem aufgezeichneten Profil so ähnlich wie möglich ist. Dazu werden anhand der Suchparameter alle Eigenschaften der Funktion (Mittelwert, Amplitude, Periode und φ_0) der Reihe nach solange verändert, bis die Quadrate der Abweichungen an jedem aufgenommenen Punkt des Profils möglichst gering sind. Die Funktion wird gesucht indem die Parameter der Reihe nach ausprobiert werden. Auch bei dieser Aufgabe wurden die Möglichkeiten der parallelen Programmierung genutzt. Die Differenzen zwischen den Profildaten und der zu testenden Kurve werden parallel berechnet und dann in einem synchronisierten Kontext miteinander verglichen. Die aufwändigen Berechnungen der Differenzen

laufen daher nebeneinander ab. Nur die einfachen Vergleiche der Differenzwerte zur Suche der kleinsten Differenz werden seriell verarbeitet.

Der Phasendiskriminator übersetzt den Phasenverlauf von 0° und 360° in eine Dreieckfunktion[2]. Zwischen 0° und 180° ist die steigende Rampe der Funktion, zwischen 180° und 360° fällt die Funktion von 180° zurück auf 0°. Diese Abhängigkeit ist in Abbildung 5.12 dargestellt.

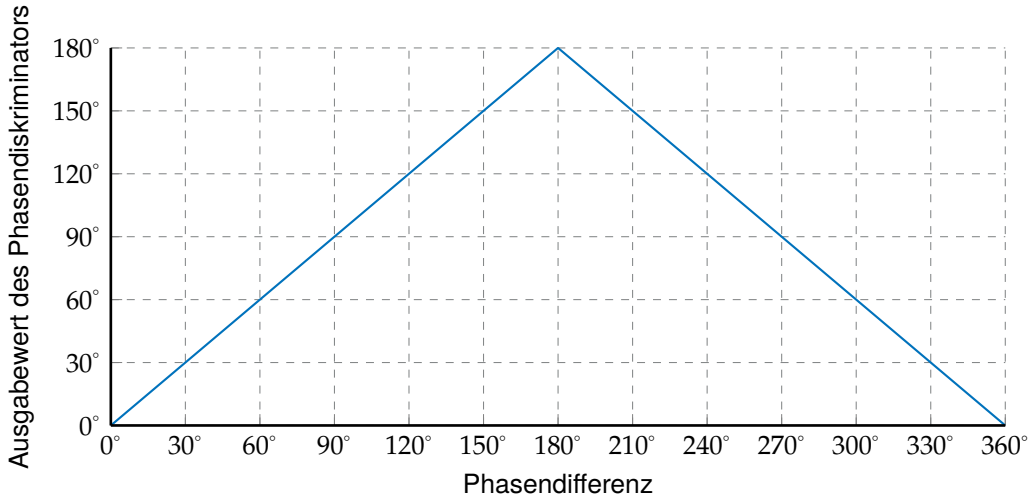

Abbildung 5.12: Abhängigkeit Phasendifferenz zu Ausgabe des Phasendiskriminators

Um eine größtmögliche Übereinstimmung mit der Profilkurve zu erreichen, muss die Profilfunktion eine Dreieck-Funktion sein. Die entsprechende Funktion ist in Formel (5.14) dargestellt.

$$\varphi = M + A \cdot \frac{2}{\pi} \cdot \arcsin\left(\sin\left(2 \cdot \pi \cdot f \cdot p + \varphi_0\right)\right) \tag{5.14}$$

Nachdem der Algorithmus die Werte mit der gringsten Abweichung für die Funktion bestimmt hat, beginnt der nächste Durchlauf. Dabei werden die Suchparameter so angepasst, dass die Grenzen des Suchbereichs jeweils einen Suchschritt über und unter dem gefundenen Idealwert sind. Die Anzahl der Schritte, in die der Bereich unterteilt wird, bleibt von Durchgang zu Durchgang gleich. Damit wird der feinere Bereich viel genauer abgesucht, um die idealen Werte zu finden. Diese Durchläufe lassen sich theoretisch beliebig oft wiederholen, um die Genauigkeit zu erhöhen. Allerdings vergrößert jeder zusätzliche Durchlauf den Zeitaufwand, um die passende Profilfunktion zu finden. Die Verbesserung wird mit jedem Durchlauf geringer. Praktisch hat sich gezeigt, dass mehr als drei Durchläufe nicht sinnvoll sind. Bei mehr Durchläufen rechtfertigt der höhere Zeitaufwand die Verbesserung nicht mehr.

Das Ergebnis ist eine Funktion, deren Werte mit den Werten des Profils fast übereinstimmen. In Abbildung 5.13 ist eine Profilkurve und die angepasste Profilfunktion zu sehen. Eine weitestgehend exakte Übereinstimmung wird bei Profilen, die ohne Probleme aufgezeichnet wurden, immer

erreicht. Profile von Strecken mit einer sehr hohen Dämpfung können zu stark abweichenden Ergebnissen führen.

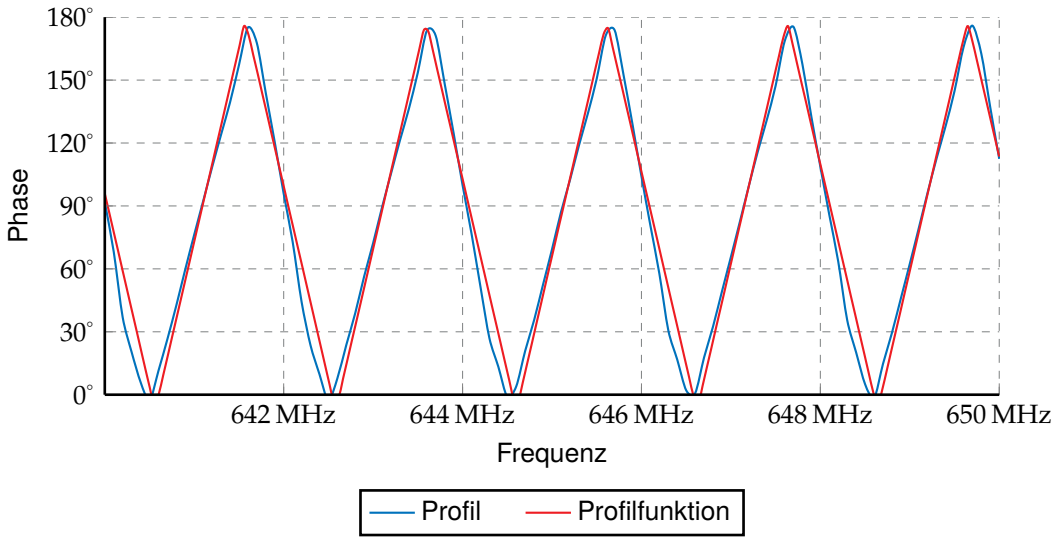

Abbildung 5.13: Profilfunktion mit angenäherter Kurve

Die Werkzeugklasse für diese Aufgabe ist in Abbildung 5.14 dargestellt und kann die Suchparameter entsprechend anpassen. Für die Suchparameter der Frequenz sollte das Resultat der Rampen-Erkennung verwendet werden. Alle anderen Parameter werden standardmäßig über ihren vollständigen gültigen Wertebereiche abgesucht. Da der Wertebereich für diese Parameter allerdings keinen so großen Schwankungen unterliegt wie der Wertebereich der Frequenz, hält sich der Aufwand dieser Suche in Grenzen. Außerdem lässt sich die Anzahl der Durchläufe frei wählen. Der Standardwert für die Durchläufe ist drei, da sich dieser Wert in bisher allen Tests bewährt hat. Es ist allerdings problemlos möglich, die Anzahl der Durchläufe nach oben oder unten zu verändern.

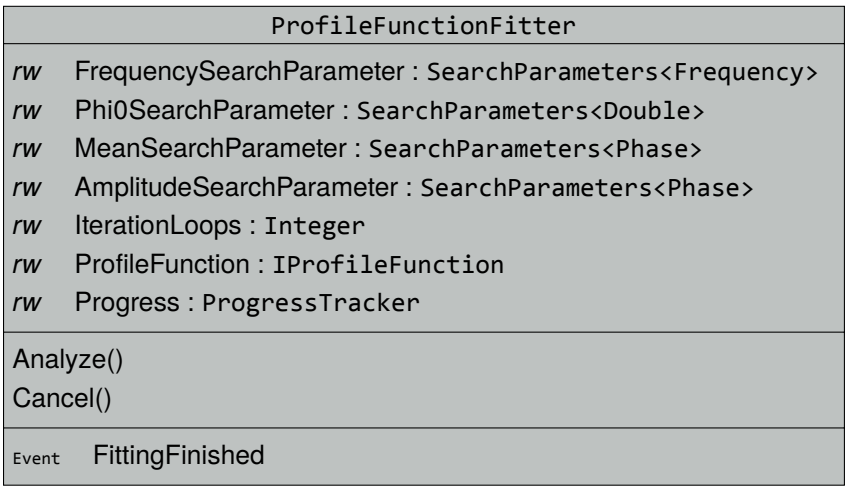

Abbildung 5.14: Klassendiagramm für das Profilfunktion-suchen-Werkzeug

5.4.4 Phasenstabilisierung

Der reguläre Messbetrieb des ILM sieht vor, dass die Phasendifferenz stabil auf einem Wert gehalten wird. Dieser Wert liegt in der Regel bei 90°, um die maximale Aussteuerung in Richtung Dehnung und in Richtung Stauchung zu ermöglichen. Ein Arbeitspunkt ist in Abbildung 5.15 dargestellt. Bei einer Veränderung der Länge des LWL kommt es zu einer Änderung der Phasenverschiebung. Die Aufgabe des Werkzeugs zur Phasenstabilisierung ist es, diesem Effekt durch eine Anpassung der Arbeitsfrequenz entgegenzuwirken. Die Veränderung der Frequenz kann im Anschluss z. B. mit dem „Lichtwellenleiter“-Werkzeug (Abschnitt 5.3.6 auf Seite 32) genutzt werden, um die Längenänderung des LWL zu berechnen.

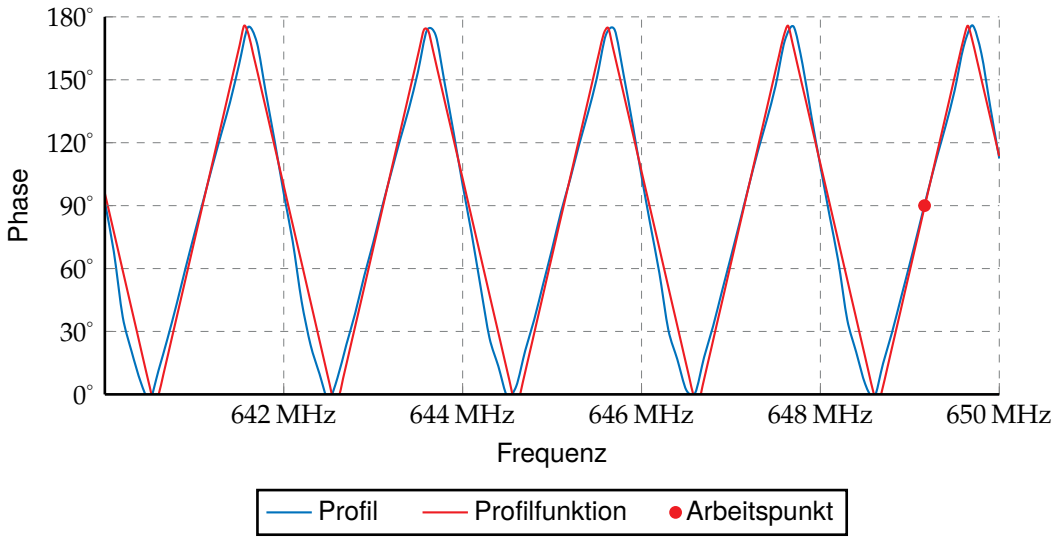

Abbildung 5.15: Profilfunktion mit angenäherter Kurve und Arbeitspunkt

Die Reglung erfolgt entlang einer Arbeitsgerade, deren Anstieg aus der Amplitude A und der Periode p der Profilfunktion berechnet wird. Für steigende Flanken wird Formel (5.15) für die Berechnung der Arbeitsgerade verwendet, für fallende Flanken wird Formel (5.16) verwendet.

$$N_{control} = 4 \cdot A \cdot p \tag{5.15}$$

$$N_{control} = -4 \cdot A \cdot p \tag{5.16}$$

$$\left[N_{control}\right] = [^\circ\mathrm{s}] = [^\circ/\mathrm{Hz}] \tag{5.17}$$

Die Berechnung der neuen Frequenz in einem Regelschritt ergibt sich aus der Phasenabweichung des gewählten Arbeitspunktes φ_{AP}, der gemessenen aktuellen Phasenabweichung φ, dem Anstieg der Arbeitsgerade $N_{control}$ und einem Verstärkerfaktor s wie in Formel (5.18) angegeben. Der Verstärkerfaktor dient dazu, die Geschwindigkeit des Reglers zu beeinflussen.

$$f = \frac{\varphi_{AP} - \varphi}{N_{control} \cdot s} \tag{5.18}$$

Diese Klasse, die zur Regelung der Frequenz verwendet wird, erhält ihre Werte automatisch von dem Phasendiskriminator-Werkzeug (Abschnitt 5.3.5). Sie steuert das Frequenzgenerator-Werkzeug (Abschnitt 5.3.1), um die Änderungen der Phase zu korrigieren. In Abbildung 5.16 ist der Aufbau der entsprechenden Klasse zu sehen. Diese Klasse korrigiert die Frequenz nur auf Befehl um die Regelgeschwindigkeit zu senken. Die Korrektur der Frequenz wird nur bei einem Aufruf der `ApplyNewFrequency`-Funktion ausgeführt. Diese muss zu einem beliebigen Zeitpunkt nach dem Ereignis `PhaseStabilized` aufgerufen werden.

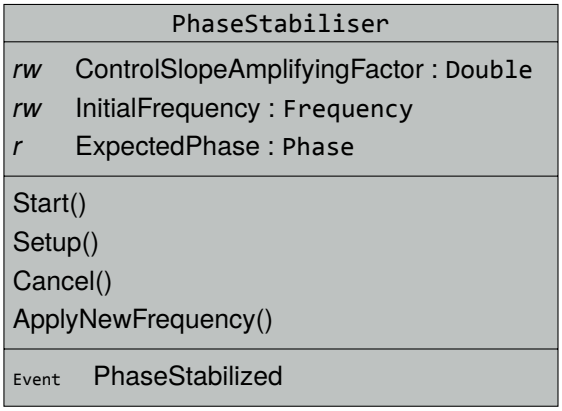

Abbildung 5.16: Klassendiagramm für das Phasen-Stabilisierungs-Werkzeug

Um die Phasenstabilisierung vorübergehend abzuschalten, wird der Verstärkerfaktor s auf den Wert 0 gesetzt. Auch wenn dieser Wert nach der Formel (5.18) ungültig ist, wird er gesondert behandelt. Das führt zu einer unveränderlichen Frequenz des Frequenzgenerators.

5.5 WPF-Anbindung

Bei der Anbindung an WPF geht es grundsätzlich darum, bestimmte Eigenschaften von Nicht-WPF-Objekten, wie z.B. von den in den vorangegangenen Abschnitten genannten Werkzeugen, an Anzeigeelemente von WPF zu koppeln. Der Vorteil dieser Anbindungen ist, dass beim Entwurf der WPF-Oberfläche die entsprechenden Verknüpfungen hergestellt werden können und WPF die Übertragung der Daten automatisch durchführt, ohne das zusätzlicher Programmieraufwand notwendig ist. Bei dieser Art der Datenverbindung werden grundsätzlich zwei Eigenschaften eines Quell- und eines Zielobjekts verbunden (Abbildung 5.17). Das Quell- und Zielobjekt können das gleiche Objekt sein.

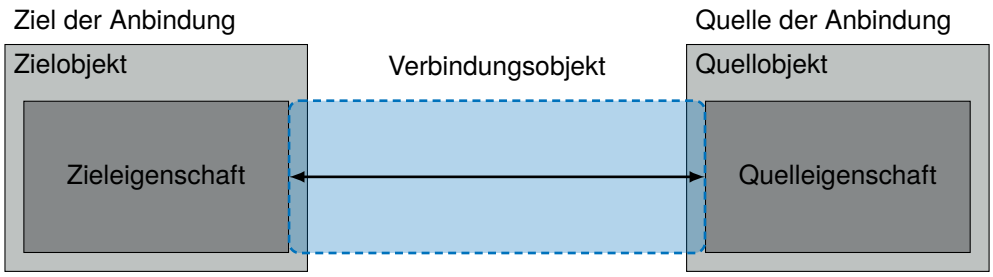

Abbildung 5.17: Objekt-Verbindung in WPF

5.5.1 Verknüpfungsmethoden

Es gibt verschiedene Methoden, um Datenquelle und Datenziel zu verknüpfen. Im Kontext dieser Arbeit sind die Datenziele immer die grafischen Steuerelemente auf der Benutzeroberfläche der Anwendung. Die Datenquellen sind die Werkzeuge des Softwarepakets, das in dieser Arbeit beschrieben wird. Für die Auswahl der richtigen Verknüpfungsmethoden ist es wichtig zu wissen, wie und wann sich die entsprechenden Daten ändern können. Es gibt grundsätzlich vier Methoden, um Datenquelle und Datenziel zu verknüpfen.

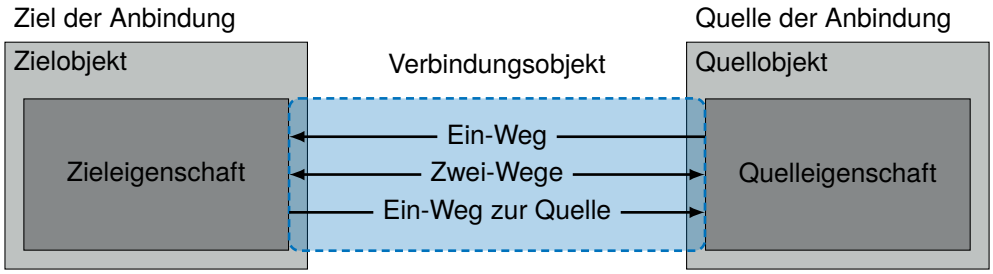

Abbildung 5.18: Verbindungsarten in WPF

Eine Einwegverknüpfung sorgt dafür, dass Datenänderungen in der Datenquelle auf das Datenziel übertragen werden. Allerdings werden Änderungen im Datenziel nicht auf die Quelle zurück übertragen. Diese Art der Verknüpfung muss immer dann verwendet werden wenn die verknüpfte Eigenschaft nur lesbar ist. Ein gutes Beispiel dafür sind die Werte des ADC-Werkzeugs. Die Spannungswerte, die von den Kanälen des ADC empfangen werden, können gelesen, aber nicht geschrieben werden.

Die Zweiwegeverknüpfung unterscheidet sich von der Einwegverknüpfung dadurch, das Änderungen am Datenziel auf die Datenquelle zurück geschrieben werden. Diese Art der Verknüpfung ist nur bei Eigenschaften gültig, die les- und schreibbar sind. Als Beispiel kann hier die aktuelle Frequenz des Frequenzgenerator-Werkzeugs genannt werden. Diese kann mit einem Einstell-Element auf der grafischen Benutzeroberfläche verknüpft werden, das bei Änderungen der Frequenz immer den aktuellen Wert anzeigt. Wird die Frequenz mit dem Einstell-Element allerdings verändert, wird diese Änderung auch an das Frequenzgenerator-Werkzeug übertragen. Sie führt damit zu einer Änderung der Frequenz, die im ILM erzeugt wird.

Die dritte Methode der Verknüpfung ist die Einwegverknüpfung zur Quelle. Diese funktioniert genau umgekehrt zur Einwegverknüpfung. Änderungen am Datenziel werden auf die Datenquelle übertragen, aber nicht andersherum.

Die letzte Methode, die verfügbar ist, ist die Einmalverknüpfungen. Diese Art der Verknüpfung übergibt den Wert der Datenquelle bei Initialisierung der Verknüpfung einmalig an das Datenziel. Alle späteren Änderungen der Datenquelle oder des Datenziels werden nicht mehr übertragen. Diese Art der Verknüpfung eignet sich zur Anzeige von Werten, die sich während der Laufzeit der Anwendung niemals ändern.

5.5.2 Auslöser für die Aktualisierung

Es gibt drei Möglichkeiten den Zeitpunkt festzulegen, wann der Datenaustausch zwischen Datenquelle und Datenziel stattfinden soll. Einmalverknüpfung sind davon ausgenommen, weil diese niemals aktualisiert werden.

Die *Explizite*-Methode führt eine Aktualisierung nur dann aus, wenn von der Anwendung eine entsprechende Funktion aufgerufen wird. Mit dieser Methode hat man die beste Kontrolle darüber, wann die Daten ausgetauscht werden. Sinnvoll ist die Verwendung dieser Art der Aktualisierung z. B. dann, wenn der neue Wert einer Eigenschaft erst übertragen werden soll, wenn der Benutzer einen „Übernehmen“-Knopf drückt.

Die *Fokus-Verloren*-Methode ist besonders bei Verknüpfungen interessant, mit denen vor allem Daten aus der Benutzeroberfläche zur Datenquelle übertragen werden. In diesem Fall werden die Daten übertragen, sobald das entsprechende Element der Benutzeroberfläche den Programm-Fokus verliert. Das passiert z. B., wenn der Textcursor ein Eingabefeld verlässt, weil er auf ein anderes Feld übertragen wird, oder weil die Eingabe mit *Enter* abgeschlossen wurde. Bei einem WPF-Textfeld ist diese Art der Verknüpfung der Standardwert für Verknüpfungen mit der *Text*-Eigenschaft.

Die dritte Methode ist die *Eigenschaft-Verändert*-Methode. Diese überträgt alle Änderungen an den Daten, sobald diese geschehen. Am Beispiel eines Textfeldes wird also jede Änderung des Textes sofort übertragen, wenn sie passiert. Wenn ein kurzer Text geschrieben wird, wird der Text in jedem Zwischenzustand mit übertragen. Diese Methode wird vor allem bei den Software-Werkzeugen verwendet, die in dieser Arbeit vorgestellt wurden. Sie lösen über ein `PropertyChanged`-Ereignis die Aktualisierung der Verknüpfung aus, sobald neue Daten vorliegen. Diese Art der Aktualisierung ist für die meisten WPF-Steuerelemente der Standardwert.

5.5.3 Konvertierung der Daten

Wenn zwei Eigenschaften mit unterschiedlichen Typen verknüpft werden sollen, dann muss die Möglichkeit bestehen, die Datentypen zu konvertieren. Dafür müssen Konvertierungsklassen zur

Verfügung gestellt werden, die in der Lage sind, die verschiedenen Datentypen, die miteinander verknüpft werden sollen, ineinander zu konvertieren.

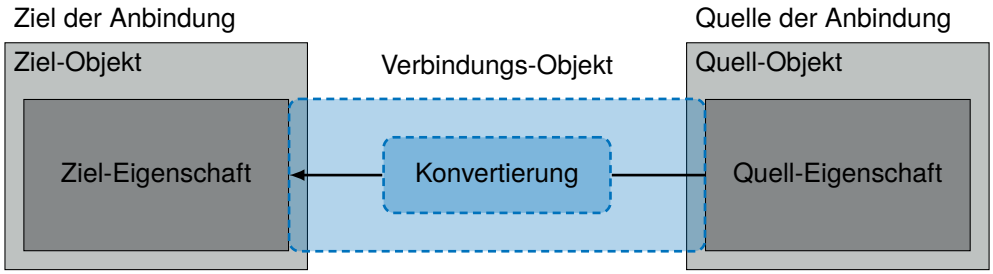

Abbildung 5.19: Konvertierung in WPF

Für diese Operationen bietet das .NET™-Framework ein passendes Interface an, mit dem eine Konvertierungsklasse erzeugt werden kann. Durch diese Konvertierungsklassen können beliebige Datentypen miteinander verknüpft werden.

Für dieses Projekt wurden Konvertierungsklassen für einige Werteklassen erstellt, die sich auf physikalische Werte (z. B. Frequenzen, Leistungen, u. a.) beziehen. Diese Klassen können verwendet werden, um die Einheiten mit Anzeige- und Steuerkomponenten von WPF zu verknüpfen.

```
<ValueConversion(GetType(Object), GetType(String))>
Public Class StringFormatConverter
    Implements IValueConverter

    Public Function Convert(value As Object, targetType As Type, param As Object,
        cult As CultureInfo) As Object Implements IValueConverter.Convert

        Dim format As String = TryCast(param, String)
        If format Is Nothing Then
            Return value.ToString
        End If
        Return String.Format(cult, format, value)
    End Function

    Public Function ConvertBack(v As Object, tT As Type, param As Object, cult As
        CultureInfo) As Object Implements IValueConverter.ConvertBack
        Return Nothing
    End Function
End Class
```

Quellcode 5.1: String-Format Konvertierungsklasse

6 Anwendungsentwicklung

6.1 Fertige Anwendungen

Bei der Entwicklung der Kommunikationsumgebung und der Werkzeuge sind einige Anwendungen entstanden, die Teil des Projekts sind. Diese Anwendungen nutzen die Möglichkeiten der neu geschaffenen Softwareumgebung und können als Referenz für neue Anwendungen betrachtet werden.

6.1.1 Diagnose

Das Diagnose-Programm ist eine einfache Anwendung, die vor allem dazu dient das ILM und die Softwareumgebung zu testen. Die Software liest kontinuierlich alle Werte, die das ILM liefern kann. Der Zweck dieser Software war ursprünglich der Test des Verbindungsaufbaus über die RS-232-Verbindung und die Überprüfung der Berechnungen der einzelnen Werkzeuge. Die Software liefert die Spannungswerte des ADC, die Werte des Phasendiskriminators und die Diagnosedaten des SFP, soweit verfügbar. Die Software erlaubt es außerdem, die aktuelle Messfrequenz des ILM innerhalb des gültigen Bereichs einzustellen.

6.1.2 Powermeter

Das *Powermeter* ist ein Programm, dass das ILM als Dämpfungsmesser benutzt. Es ist die einzige Anwendung, die auf WPF basiert. Es wurde entwickelt, um die Verbindung zwischen den Software-Werkzeugen und der WPF-Oberfläche zu testen und zu erweitern. Die Anwendung verwendet die Möglichkeiten zur Verknüpfung, die in Abschnitt 5.5 beschrieben wurden. Das Programm liest die Diagnosedaten des SFP aus und zeigt die Leistungswerte in Form eines Dämpfungswertes an.

Diese Anwendung kam bei dem Optronik-Seminar 2012 an der Northumbria University in Newcastle upon Tyne zum Einsatz und funktionierte dort ohne Probleme.

6.1.3 ILM-Messsoftware

Die ILM-Messsoftware ist in der Lage die eigentlichen Messaufgaben durchzuführen, für die das ILM gedacht ist. Es verwendet sämtliche Software -Werkzeuge, um das Profil der Messstrecke zu lesen und zu analysieren. Die Profilfunktion wird im Anschluss benutzt um die Phasendifferenz durch Reglung der Frequenz auf einem konstanten Wert zu halten. Außerdem zeichnet das Programm die Änderungen der Länge des LWL und die aktuellen Dämpfungswerte auf. Alle Werte werden mit den

Software-Werkzeugen berechnet. Dieses Programm wurde entwickelt, um das Zusammenspiel aller Software-Werkzeuge zu testen und die Stabilität der Software-Komponenten unter realen Einsatzbedingungen zu überprüfen.

6.2 Konzept für neue Software

Das komplette Softwarepaket setzt sich aus sieben Bibliotheken zusammen die in einer Anwendung benutzt werden können. Jede dieser Bibliotheken liegt als Dynamic Link Library (DLL) vor und kann damit in anderen .NET™-Programmen verwendet werden.

- Common
- Modulsystem
- Modulsystem-Serial
- Modulsystem-USB
- NativeBridge
- USBdotNET
- USBdotNET-WPFext

Common enthält als Projekt Quellcode, der von fast allen anderen Bibliotheken gebraucht wird. Die dazu gehörende DLL muss Teil von jedem Projekt sein, das auf die Komponenten dieser Softwareumgebung zurückgreift. *Modulsystem* enthält den gesamten Werkzeugkasten (Kapitel 5) sowie die grundlegende Kommunikationsschnittstelle (Kapitel 4). Außerdem enthält sie die Definition aller Kommunikationsbefehle und die Abstraktionsschicht der Kommunikation. Jede Software, die mit dem ILM arbeiten soll, muss die dazu gehörende DLL einbinden. *Modulsystem-Serial* enthält die Implementierung der Kommunikationsschnittstelle für die Kommunikation mit dem ILM über RS-232. Programme, die eine RS-232-Verbindung mit dem ILM herstellen müssen, müssen auf die entsprechende DLL verweisen. *Modulsystem-USB* enthält die Implementierung für die USB-Verbindung. Für die Kommunikation über USB werden zusätzlich die Projekte *USBdotNET* und *NativeBridge* benötigt. Beide Projekte zusammen ermöglichen den Zugriff auf die USB-Schnittstelle. Daher muss jedes Programm die Projekt-DLL einbinden, wenn über die USB-Schnittstelle kommuniziert werden soll. Das Projekt *USBdotNET-WPFext* ist eine kleine Erweiterung für *USBdotNET*, mit der die Einbindung in WPF-Anwendungen vereinfacht wird. WPF-Anwendungen, die die USB-Verbindung nutzen, sollten auf dieses Zusatzprojekt mit seiner DLL zurückgreifen.

7 Zusammenfassung und Ausblick

Die Softwareumgebung, die im Zuge dieser Arbeit entwickelt wurde, kann die Basis für jede PC-Software darstellen, die mit dem ILM arbeiten muss. Die Software ist in der Lage, alle Aufgaben zu erledigen, die im Moment abzusehen sind. Bei neuen Software-Entwicklungen kann sich auf die Präsentation der Ergebnisse und auf die Besonderheiten der einzelnen Software-Versionen konzentriert werden.

In der Zukunft kann diese Softwareumgebung durch neue Werkzeuge erweitert werden. Eine sinnvolle Erweiterung wäre, die gemessenen Daten direkt anhand der aktuellen Temperatur zu kompensieren und somit die starke Temperaturabhängigkeit der Messwerte zu reduzieren. Außerdem könnte die Kommunikation um den Controller Area Network (CAN)-Bus erweitert werden. Als gemeinsame Basis für zukünftige Anwendungen bietet diese Lösung außerdem den Vorteil, dass sich jede Verbesserung an der Softwareumgebung auf jede Anwendung auswirkt, die auf diese Softwareumgebung zurückgreift.

Die entstandene Softwareumgebung unterstützt, dass eine Anwendung mehrere ILM-Geräte gleichzeitig steuert. Eine komplette Software für diesen Anwendungsfall gibt es allerdings noch nicht. Wenn die Ergebnisse mehrerer Messgeräte exakt synchronisiert werden müssen, hätte diese Möglichkeit entscheidende Vorteile.

Literaturverzeichnis

[1] Analog Devices Inc. (Hrsg.): *AD974 Data Sheet*. Rev. A. http://www.analog.com/static/imported-files/data_sheets/AD974.pdf: Analog Devices Inc., 1999

[2] Analog Devices Inc. (Hrsg.): *AD8302 Data Sheet*. Rev. A. http://www.analog.com/static/imported-files/data_sheets/AD8302.pdf: Analog Devices Inc., Juli 2002

[3] Analog Devices Inc. (Hrsg.): *TMP35/TMP36/TMP37 Data Sheet*. Rev. F. http://www.analog.com/static/imported-files/data_sheets/TMP35_36_37.pdf: Analog Devices Inc., November 2010

[4] Analog Devices Inc. (Hrsg.): *AD9834 Data Sheet*. Rev. C. http://www.analog.com/static/imported-files/data_sheets/AD9834.pdf: Analog Devices Inc., Februar 2011

[5] APAC Opto Electornics Inc. (Hrsg.): *LS38-A3S-Tx-N Data Sheet*. Version 1.1. http://www.apacoe.com.tw/datasheet/60/LS38-A3S-Tx-N-V1.1.pdf: APAC Opto Electornics Inc., August 2011

[6] Callas, J. ; Donnerhacke, L. ; Finney, H. ; Shaw, D. ; Thayer, R.: *OpenPGP Message Format*. RFC4880. http://tools.ietf.org/html/rfc4880: The Internet Engineering Task Force, November 2007

[7] Karing, Martin: *Entwicklung eines USB Interface und Implementierung eines 16bit 8-Kanal A/D-Wandlers in ein modularisiertes optisches Längenmessgerät*. Technikumplatz 17, 09648 Mittweida, Deutschland, University of Applied Science Mittweida, Praktikumsarbeit, August 2010

[8] Karing, Martin: *Software- und Hardwareentwicklung für ein Sensorsystem*. Technikumplatz 17, 09648 Mittweida, Deutschland, University of Applied Science Mittweida, Bachelorarbeit, März 2011

[9] Liverage Technology Inc. (Hrsg.): *F413S17415-D Data Sheet*. Version A. http://www.liverage.com.tw/UploadFile/Download/201161419311319O31L13OD.pdf: Liverage Technology Inc., Dezember 2005

[10] SFF Committee (Hrsg.): *SFF-8472 Specification for Diagnostic Monitoring Interface for Optical Transceivers*. Rev 11.0. ftp://ftp.seagate.com/sff/SFF-8472.PDF: SFF Committee, September 2010

[11] Silicon Laboratories (Hrsg.): *Si570/Si571 Data Sheet*. Rev. 1.3. http://www.silabs.com/Support%20Documents/TechnicalDocs/si570.pdf: Silicon Laboratories, Juni 2011

[12] Texas Instruments Inc. (Hrsg.): *ADS8331/ADS8332 Data Sheet*. Überarbeitung Mai 2012. http://www.ti.com/lit/ds/symlink/ads8331.pdf: Texas Instruments Inc., Mai 2012

[13] Torvalds, Linus B.: Kernel SCM Saga... In: *Linux-Kernel Archive* (2005), April. – http://lkml.indiana.edu/hypermail/linux/kernel/0504.0/1540.html

[14] USB.org (Hrsg.): *Device Class Definition for Human Interface Devices (HID)*. Rev. 1.11. http://www.usb.org/developers/devclass_docs/HID1_11.pdf: USB.org, Juni 2001

Stichwortverzeichnis

.NET™ .NET™ ist eine von Microsoft® entwickelte Software-Plattform zur Entwicklung und Ausführung von Programmen. Diese Plattform setzt sich aus einer Softwaresammlung, dem .NET™-Framework und einer Laufzeitumgebung, in der die Anwendungen ausgeführt werden, zusammen.

.NET™-Framework Das .NET™-Framework ist ein wichtiger Teil der von Microsoft® entwickelten Software-Plattform .NET™. Das Framework enthält eine große Anzahl an vorgefertigten Klassen und Funktionen, die für das schnelle Entwickeln von Software zur Verfügung stehen.

Analog-Digital-Konverter Ein Analog-Digital-Konverter ist ein Bauteil, das eine analoge Spannung innerhalb eines bestimmten Bereichs in einen digitalen Wert übersetzt, der besser in einem Prozessor weiterverarbeitet werden kann als der analoge Wert.

Controller Area Network Der CAN-Bus ist ein serielles Bussystem, das für den asynchronen Austausch von Daten gedacht ist. Er gehört zu den Feldbussen. Entwickelt wurde der CAN-Bus von Bosch für die Vernetzung von Steuergeräten in Automobilen. Der CAN-Bus ist international standardisiert und definiert Layer 1 und Layer 2 des OSI-Referenzmodells.

Digital diagnostics monitoring Moderne SFP-Module unterstützen die Funktionen der digitalen Diagnose. Diese liefert Informationen zur optischen Eingangs- und Ausgangsleistung des SFP-Moduls. Außerdem liegen Werte wie die Stromaufnahme, die Betriebsspannung und die interne Temperatur vor.

Direct-Digital-Synthesizer Ein Direct-Digital-Synthesizer ist ein Baustein, der zur Erzeugung von periodischen Signalen benutzt wird. Die Besonderheit dieses Frequenzgenerators besteht in der sehr feinen Einstellbarkeit der Frequenz.

Dynamic Link Library Bibliotheken zum dynamischen Verknüpfen sind Teile von Programmen, die in zusätzlichen Dateien außerhalb der eigentlichen ausführbaren Datei gespeichert werden. Beim Start eines Programms werden die entsprechenden Bibliotheken automatisch zu den ausgeführten Programmteilen hinzugefügt. Diese Bibliotheken werden meistens für Software-Komponenten benutzt die in verschiedenen Anwendungen, in gleicher Form verwendet werden.

Echtzeituhr Eine Echtzeituhr (real-time clock) ist eine Uhr, welche die physikalische Zeit misst. Sie läuft über die Betriebszeit des Gerätes hinaus weiter.

Ereignis Ereignisse können innerhalb des Programmablaufs zu beliebigen Zeitpunkten auftreten und werden von speziellen Methoden behandelt, die das jeweilige Ereignis überwachen.

GNU Privacy Guard GNU Privacy Guard ist ein freies Kryptographiesystem, welches zum Ver- und Entschlüsseln von Dateien sowie zum Erzeugen und Prüfen von elektronischen Signaturen genutzt werden kann. Das Programm verwendet den OpenPGP-Standard nach RFC4880[6].

Human Interface Device Human Interface Device ist eine USB-Geräteklasse, die Geräte beschreibt, mit denen der Benutzer direkt interagiert. Tastaturen, Mäuse und Joysticks sind typische HID-Geräte.

Hypertext Transfer Protocol Das Hypertext Transfer Protocol ist ein auf TCP aufsetzendes Netzwerkprotokoll, mit dem Daten über das Netzwerk übertragen werden können. Die Hauptanwendung ist momentan die Übertragung von Webseiten aus dem Internet.

Hypertext Transfer Protocol Secure Das Hypertext Transfer Protocol Secure dient zum sicheren Datenaustausch über ein Netzwerk. Technisch ist dieses Protokoll dem Hypertext Transfer Protocol sehr ähnlich. Allerdings wird zwischen TCP und HTTP noch eine zusätzliche Übertragungsschicht definiert, die sich um die Verschlüsselung der Daten kümmert. Die genaue Art der Verschlüsslung ist nicht festgelegt und wird beim Aufbau der Verbindung ausgehandelt.

Integrales Längenmesssystem Das integrale Längenmessystem ist ein Messgerät das von der Lehr- und Forschungsgruppe Optronik an der Hochschule Mittweida entwickelt wird. Das Messgerät ist in der Lage, Längenänderungen an einem Lichtwellenleiter zu messen, die durch Dehnung oder Stauchung des Lichtwellenleiters entstehen. Die Längenmessung basiert auf der Messung der Laufzeit des Lichts im Lichtwellenleiter.

Lichtwellenleiter Lichtwellenleiter sind Kabel oder Leitungen die zur Übertragung von Licht verwendet werden. Das Licht wird dabei in Fasern aus Glas oder Kunststoff übertragen.

Objektorientierte Programmierung Die objektorientierte Programmierung ist ein Programmierkonzept, nach dem Computerprogramme entwickelt werden können. Das Konzept sieht vor, dass Daten und Funktionen in Objekten zusammengefasst werden.

Personal Computer Ein Einzelplatzrechner, der im Kontext dieser Arbeit als Plattform für die Messsoftware verwendet wird.

Phasenregelschleife Eine Phasenregelschleife (engl. phase-locked loop) dient dazu, die Frequenz eines veränderlichen Oszillators einzustellen. Dazu wird in einem geschlossenen Regelkreis die Phasenabweichung zwischen dem Oszillator und dem Referenzsignal möglichst konstant gehalten. Die Phasenregelschleife kann damit als Frequenzgenerator genutzt werden, bei dem die Ausgangsfrequenz exakt eingestellt werden kann und stabil gehalten wird.

Polymere optische Faser Polymere optische Fasern sind Lichtwellenleiter, die aus Kunststoff gefertigt wurden. Im Vergleich zu Lichtwellenleitern aus Glas haben die POF einen viel größeren Kerndurchmesser und geringere Produktionskosten sowie eine einfachere Verbindungstechnik. Der Nachteil gegenüber Glas-Lichtwellenleitern besteht vor allem in der höheren Dämpfung, wodurch dieser Lichtwellenleiter-Typ vor allem in Kurzstecken eingesetzt wird.

Prozedurale Programmierung Die prozedurale Programmierung ist ein Programmierkonzept, nach dem Computerprogramme entwickelt werden können. Das Konzept sieht vor, dass das Problem in Teilprobleme zerlegt wird. Diese Teilprobleme werden in abgegrenzten Funktionen bearbeitet. Die prozedurale Programmierung wird vor allem verwendet, um die

Übersichtlichkeit des Quellcodes zu verbessern und Wiederholungen des Quellcodes zu vermeiden.

Repository Ein Repository ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung von digitalen Objekten. Im Kontext dieser Arbeit ist ein Repository der Speicherplatz für ein Versionsverwaltungssystem, in dem die Daten und die Geschichte aller Dateien, die Teil der Versionsverwaltung sind, gespeichert werden.

RS-232 RS-232 ist ein Standard für eine bei älteren Computern oft vorhandene serielle Schnittstelle, die seit 1960 standardisiert ist. In neueren Rechnern ist diese Schnittstelle immer seltener eingebaut. Sie wird aber auch heute wegen der robusten mechanischen Verbindung der Stecker und wegen des einfachen Übertragungsprotokolls noch häufig verwendet. Sie eignet sich für Übertragungsstrecken von bis zu einem Kilometer.

Secure Shell Secure Shell ist ein auf TCP aufsetzendes Netzwerkprotokoll mit dem man sicher eine verschlüsselte Verbindung mit einem entfernten Gerät herstellen kann. Außerdem ist Secure Shell der Name der Programme mit denen dieses Netzwerkprotokoll genutzt werden kann.

Small Form-factor Pluggable Bei Small Form-factor Pluggable handelt es sich um kleine Module, die im Bereich der Netzwerktechnik standardisiert sind. Diese Module gibt es sowohl als optische als auch elektrische Variante mit den jeweiligen Außenanschlüssen. Sie sind für die Umsetzung von extrem schnellen Netzwerkverbindungen entwickelt worden. Im optischen Bereich gibt es Versionen für die verschiedenen Leitertypen wie Singlemode- oder Multimode-LWL oder für POF.

Thread Ein Thread ist in der Informatik ein Ausführungsstrang, der parallel zu anderen Threads arbeiten kann. Ein Thread ist immer Teil eines Prozesses oder einer Anwendung. Die Programmiertechnik, die die Verwendung mehrerer Threads in einem Programm vorsieht, nennt sich „Multithreading“. Dabei erledigt der Prozess mit Hilfe mehrer Threads verschiedene Aufgaben zur gleichen Zeit. Die Verwendung von Threads ist die einzige Programmiermethode, um die Vorteile von Mehrkern-Prozessoren effektiv zu nutzen.

Universal Serial Bus Der universelle serielle Bus ist ein in der heutigen Zeit sehr verbreitetes Anschluss- und Kommunikationssystem, das vor allem bei Heimcomputern eine sehr große Verbreitung erreicht hat. Er wird verwendet, um verschiedenste Geräte mit einem PC zu verbinden. Dazu zählen Eingabegeräte (Maus, Tastatur), Ausgabegeräte (Drucker, Lautsprecher) und viele anderen Gerätetypen.

Windows Presentation Foundation Die Windows Presentation Foundation ist ein Bestandteil des .NET™-Framework und wird zum Erstellen von Desktop- und Web-Anwendungen benutzt. Es enthält die notwendigen Komponenten, die für die Präsentation einer Anwendung gebraucht werden.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 13. November 2012