OCTane - Applying software quality-measures to bare-metal firmware

Florian Hinterleitner

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

embedded systems design

in Hagenberg

im Juni 2022

Advisor:

Langer, Rankl, Zorin

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere. This printed copy is identical to the submitted electronic version.

Hagenberg, June 15, 2022

Florian Hinterleitner

Contents

Preface

This document uses the APA citation and reference style (see Ch. **??** for details).

Abstract

To-Do: detuschen abstract ubersetzen

Kurzfassung

Die Firma RECENDT GmbH entwickelt und baut OCT-Systeme (optical coherence tomography), für die im Rahmen dieser Diplomarbeit ein Teil der Steuerung entworfen werden soll. Zur 2-dimensionalen Messung mit OCT-Systemen kommen Galvanometer-Scanner im X/Y-Betrieb zum Einsatz. Das sind hochdynamische Drehantriebe fÜr optische Anwendungen, die mit einer Rate von rund 500Hz etwa 20°vor- und rückwärts rotieren können. Sie tragen mitrotierende Spiegel um den optischen Pfad in 2 Dimensionen auszulenken und somit flächige Scans zu ermöglichen.

Die ausgewählten Galvo-Modelle benötigen Steuersignale zur Erzeugung der Scan-Muster. Typischerweise sind dies zwei synchrone Rampen-Signale, eines schnell, eines langsam. Aufgabe ist es nun, auf bestehender Mikrocontroller-Hardware einen Signalgenerator zu programmieren. Dieser soll sowohl Rampen-Signale als auch arbiträr gewählte Signalformen erzeugen können. Dies beinhaltet FW-Module für die Digital-Analog-Wandler, Trigger-Einheit für das Timing sowie Synchronisation der Kanäle. Weiters ist die USB-Kommunikation per SCPI-Protokoll zu programmieren. Die Anbindung an eine übergeordnete Steuerung des OCT-Systems erfolgt per USB. Die handelsüblichen Galvo-Scanner besitzen mechanische Trägheiten, die bei Scan-Raten ab 800Hz kein maximale Auslenkung mehr erlauben. Deshalb würde bei hohen Geschwindgkeiten der optische Messbereich eingeschränkt werden. Um auch bei höheren Scan-Raten volle Auslenkungen zu erreichen, soll versucht werden, mit adaptierten Steuersignalen die Trägheiten auszugleichen.

Chapter 1

Introduction

To-Do: "vom Allgemeinen zum Speziellen"

1.1 Motivation

The RECENDT GmbH researches, develops and produces, among other technology areas, measurement systems employing optical coherence tomography. A key element of such OCT-systems are galvanometer-scanners. These allow for analyzation of areas, instead of only point-wise measurements, by manipulating a laser-beam. This manipulation again, has to be controlled via two separate steering-voltages, one for manipulation in x- and y-axis. An existing mikrocontroller-board, providing sufficient precise and fast DACs, is to be programmed, to form the 'OCTane', a signal-generator for mentioned steering-voltages, controllable via USB. The resulting Firmware shall also incorporate a HAL, utilizing several other functionalities, the microcontroller has to offer. Optionally, adapted curves for the steering voltages shall be investigated to allow linear control over galvanometer-scanners in higher frequency ranges.

1.2 Optical coherence tomography

Optical coherence tomography (OCT) is an imaging measurement method for the analyzation of transparent and semi-opaque materials and shows similarities to the measurement processes via ultrasound or radar. The sample to be measured is subjected to an electromagnetic wave, the resulting 'echoes' are analysed with regard to their times of flight. From these run-times again, the geometric structure is determined, including the layer structure of the sample and also the maximum penetration depth of the applied wave. This creates measured point with an in-depth resolution. Usually this EM wave is of a coherent broadband light source in the visible up to the near infrared spectrum. Coherent means, that several wave-bundles of a light source must have a fixed phase relationship to eachother. This is necessary to obtain stable interference patterns. For the Detection of the echoes, however, conventional opto-detectors or cameras do not suffice, on the one hand due to the propagation speed of light, on the other hand due to the low reflected light intensities. Therefore Interferometry is used to determine the depth of penetration of a photon, being reflected from the sample. In interferometry, a

laser beam is split into two waves of half the initial optical power. One wave is sent on an optical reference path of known length, the other to the surface of the sample. The reflections, the returning waves are superimposed and, depending on the nature of the sample material, result in constructive or destructive Interference. This interference can be detected using an opto-detector[1], or a spectrometer[2] and used for further processing. A single measured point and its depth information about the material under test is called an A-scan. Aggregates a lot of A-scans along a line (X-direction) across the sample material, forms a B-scan and the aggregation of B-scans along a line in the Y direction a volume scan, i.e. a spatial, three-dimensional image of the sample material. Relevant parameters of OCT systems are the penetration depth, the axial and lateral measurement range, axial and lateral resolution and the measurement speed. While the penetration depth of ultrasound typically reaches a few centimeters and a resolution in the millimeter range, OCT allows only to look a few millimeters below the surface, but with micron resolutions. Measurable areas, or field-of-view, in ultrasound is in the order of centimeters, with OCT in the order of millimeters[1]. achievable speed al results from A-scan rates up to 100kHz. The term 'optical coherence tomography' results on the one hand from the coherent light source. The other two parts of the name, 'tomos' means slice or section, and 'graphein' stand for writing or drawing, and both come from Greek. They reflect that the resulting image is assembled from individual slices or sectional images. The manipulation of the light beam along the mentioned lines takes place with rotatably mounted mirrors, one for the X one for the Y direction. The faster this rotation is possible, the faster OCT-images can be created. One widespread technical realization, allowing very fast rotation of the mirrors called a galvanometer-mirror or -scanner.

1.3 Galvanometer-Scanners

To-Do: 'scanner' statt 'mirror'
Galvanometer scanners (colloquial: galvos) are highly dynamic opto-mechanical components, based on the classic galvanometer according to Hans Christian Oersted: A rotatable, magnetizable object, e.g. a magnetic needle, that will be deflected from its position in the proximity of a current-carrying conductor. The low sensitivity of the effect on the current is improved by a high number of windings of the electric conductor around the deflectable object, creating an electrical Inductance, a coil. The non-linear connection between current and deflection angle can be linearized to a first-order approximation, by placing the coil between a rigidly potitioned iron-cylinder inside and a permanent-magnet, which is arranged outside the coil[2]. If this classic Galvanometer is equipped with a mirror as a rotatable object, optical paths, specifically: the beam-path of a point light source, can be manipulated in one space dimension. Feaseable for technical applications is, that this manipulation can be controlled reasonably linear by the current or voltage at the galvo coil. With the galvanometer-scanner employed for this master-thesis, power-electronics for the conversion of control signals to the required coil-currents and -voltages were already included. Therefore, furthermore, only 'control signals' will be discussed, instead of currents and voltages. A combination of two galvanometer scanners in a suitable geometric arrangement, irradiated with a point laser, allows to manipulate this point in two dimensions. Such an arrangement is shown in

fig. []. At sufficient speed with which the laser point is deflected, 2-dimensional contours can be projected, resulting in a 'stationary' image for the human eye. It shall be noted, that only closed geometric figures are possible, as long as the light source itself cannot be turned off. These components are commercially available as laser scanner and used for light effects at music events, art installations and in discotheques. Applied to OCT-systems, on the other hand, galvanometer scanners are used to expand measurement from a single point of interest. Deflecting mentioned coherent light source of an area of an examined sample, allows for two-dimensional analysis of the sample. If a dedicated x- and a y-galvo are to be steered with a slow and a fast ramp, respectively, this results in a rectangle on the sample, which is the desired 'image shape' of the laser dot for OCT systems. In this way, samples can be scanned in a grid pattern.

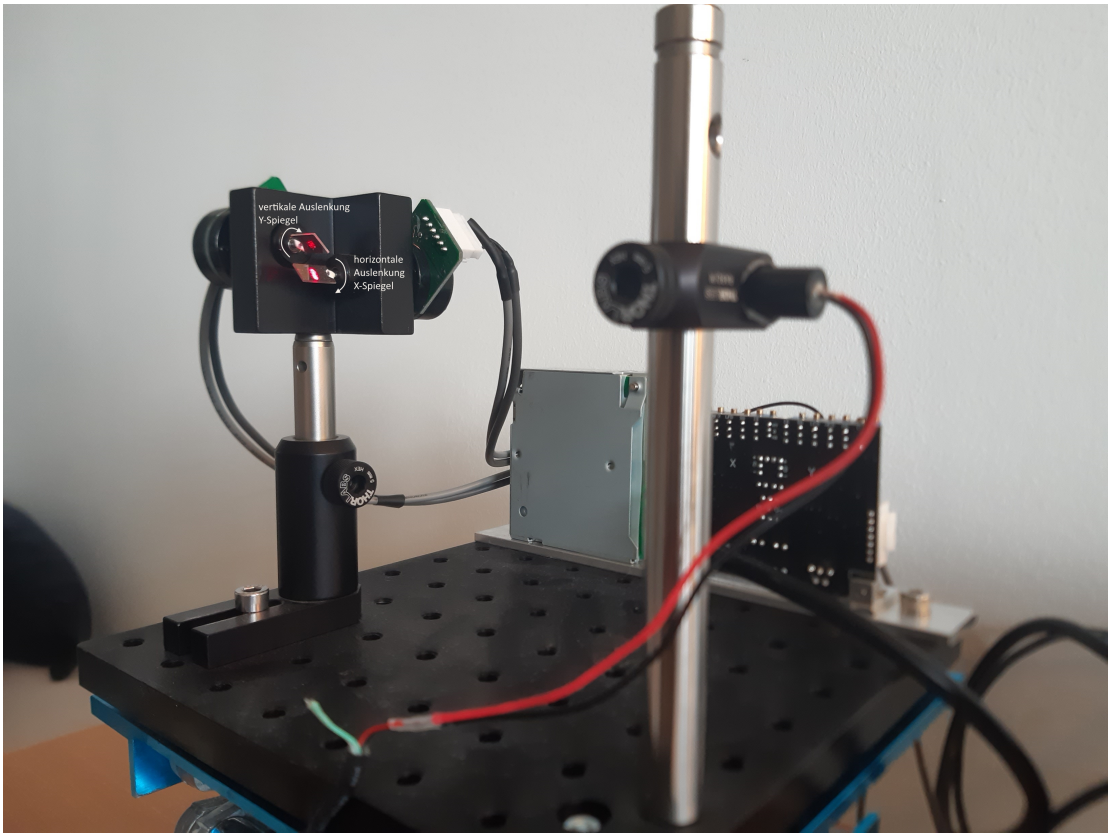

Figure 1.1: DetailGalvoOn

1.4 Control of Galvanometer-Scanners

Commercially available Galvanometer-Scanners usually allow control in the form of analog voltage inputs with a range of ± 10Volts. The angle of the rotated mirror follows that control-voltage in a linear manner for sufficiently low frequencies. The signal-forms to result in the rectangular scan-grids, as described in the previous section, are depicted in Fig. 1.3. To achieve these signals, a mikrocontroller-board was designed, including

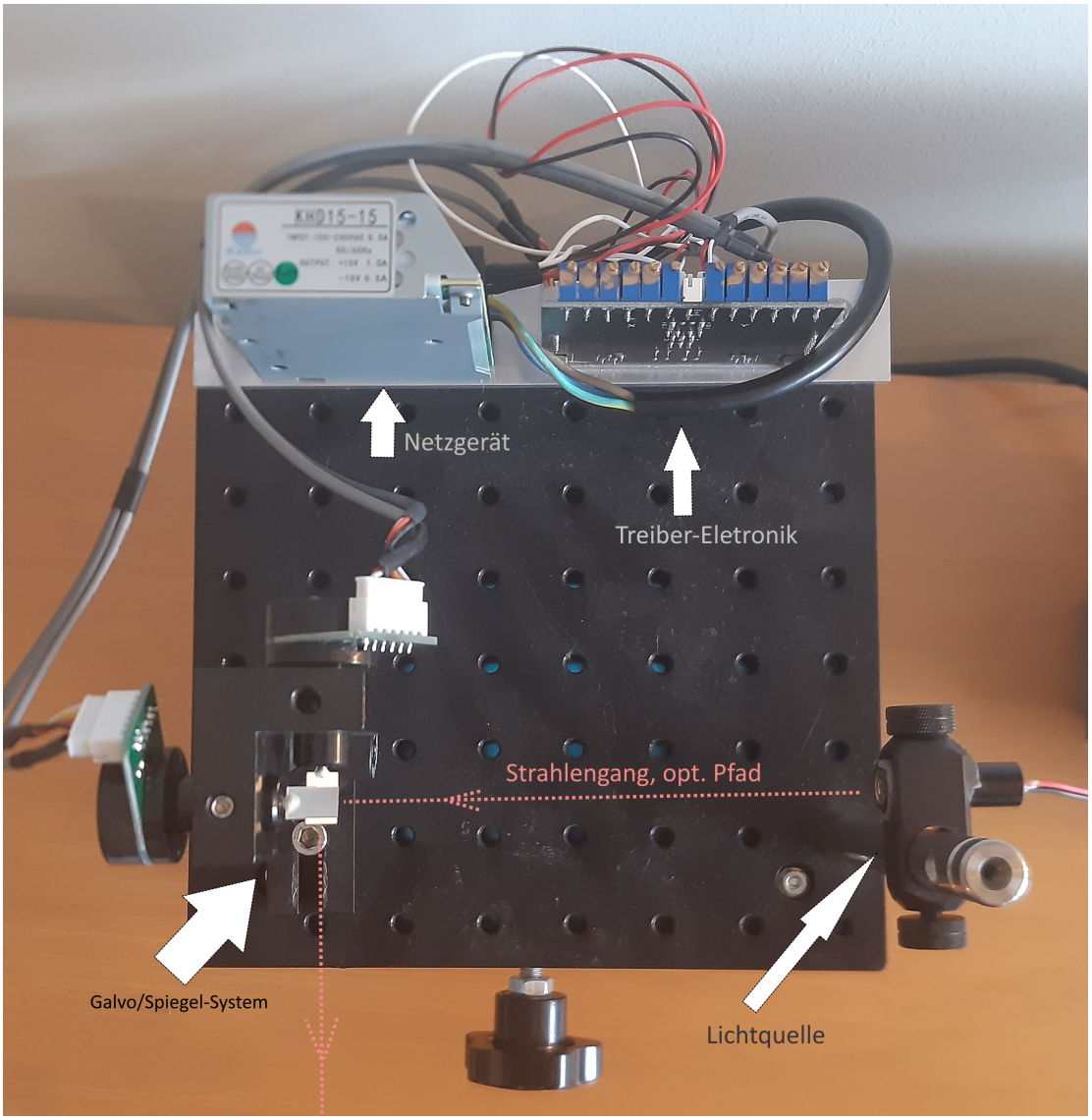

Figure 1.2: DutTop02

16-bit-DACs, USB-PHY and coaxial Trigger-IOs, among other features. To utilise these features and form an arbitrary signal generator for mentioned ramp-signals, a firmware is necessary. This should be done in a fashion employing quality-assurance during the implementation-process, as well as the verification-phase of the firmware and its supporting hardware. Aside from signal-generation and USB-connectivity, additional features are desirable, such as user-controllable Relays, utilisation of watchdog-timer, UART-, I2C- and SPI-ports as well as analog inputs. The combination of hard- and firmware will be called OCTane. **To-Do:** block-schematic: PC -> OCTane -> 2xDAC -> Galvo-Driver
This leads to the scientific problem at hand:

How can measures of quality be applied to a bare-metal firmware?

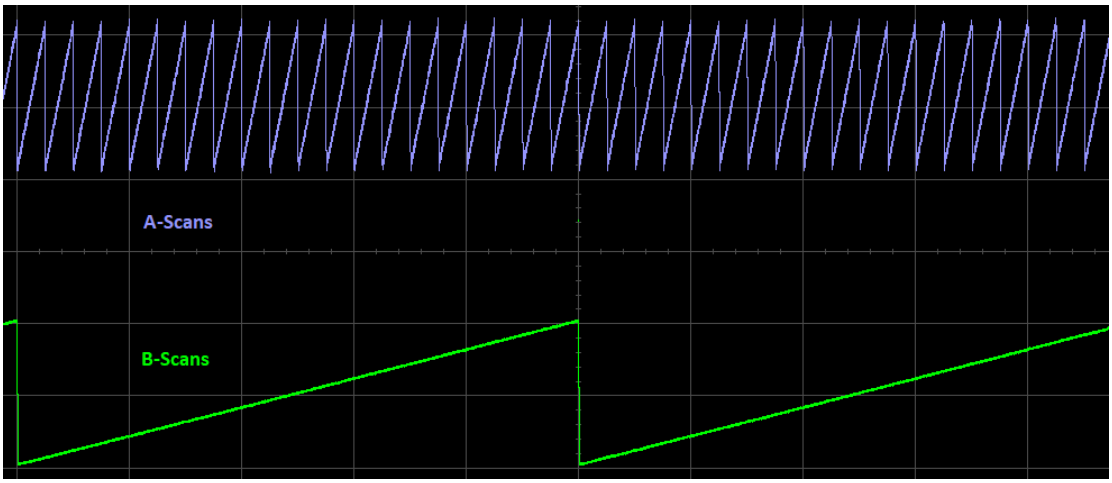

Figure 1.3: GalvoRamps01

1.4.1 optional: adapted steering curves

...fliegt wohl raus **To-Do:** Modell ausm Paper per flachheitsbasierter Reglung auf Steuerrampen draufrechnen

Chapter 2

Fundamentals

2.1 Code Quality

2.1.1 Motivation

As computers and microcontrollers are introduced into ever increasing application areas, the correct and reliable function of the software runing them becomes more important. A significantly increasing proportion of the development costs falling towards software, compared to hardware, suggests, to consider also the financial efforts put into testing and verification of said software. As the lifespan of software is significantly longer than the corresponding hardware, this seems to be a sensible investment. Nonetheless, reducing the costs for software development is also particularly economical. Investigating the dynamics of development costs leads to the result, that, the later in a project lifecycle, the higher the expenses. This means, that the largest portion of money is spent in the last phase, the maintenance of software, that is already in the market! This is a direct consequence of low quality standards applied to the product. More detailed expressed, errors are induced in the implementation, persist unnoticed through testing and verification. Finally, this erros appear as faults in operation at the customer site. Furthermore, if the project suffers from lazy documentation, insufficient structure, identifying these errors and correctiong them becomes consuming in time, cost and resources. This phenomenom is further escalated with the increasing complexity of todays software. An even worse phenomenon can arise: Insufficient understanding of a faulty piece of software at hand, can lead to introducing even more errors with additional code, that is actually intended to fix a bug. Especially when poor structuring masks hidden dependencies between modules. A general rule oh thumb here is: The earlier an error arises, for example in the phase of gathering requirements, the more extensive changes are necessary. In other words: The earlier an errors is induced and the later it is discovered, the more expensive are its consequences.

But these problems have suitable solutions at hand. It is not compulsive to plunge money, hours and employee motivation on unnecessary maintenance. Employing the right methods, implemented software can become reliable, easy to change, inexpensive to maintain and allow a more intuitive understanding. Distinguischable into two categories, these methods are either analytical or constructive. Widespread examples are functional decomposition, object-oriented development, structured design, structured analysis or

the use of the unified modeling language (UML).

functionally decomposing and object-oriented software development methods are particularly widespread. Examples are Structured Analysis (SA) /DeMarco 85/, Structured Design (SD) and the Unified Modeling Language (UML) /Booch et al. 98/. A comprehensive description of the development methods for software is contained in /Balzert 00/. Best practice is to employ a combination of constructive and analytical methods. Constructive means alone can assist in preventing errors in the intended product, but not guarantee their absence. Analytical means are capable of demonstrating the absence of errors, but not their prevention, so a large amount of preventable errors might emerge, when only analytical means are put to use. The combined use advisable would be, to employ constructive methods during every phase of a development project, and assessing the intermediate products with analytical methods by the end of each phase. This process is called the 'principle of integrated quality assurance' (cite: Liggesmeyer). If the predefined criteria for sufficient quality are not met by intermediate results, they are not transferred to the next phase and the recent phase has to further developed, until all hard criteria are met. This concept assists the development team in the early detection of errors and their removal at reasonable effort. Ideally all errors induced in a development phase are detected and eliminated by the end of the same phase. This should further help in minimizing the number of errors in a product over several phases. The described process makes it evident, that testing only a finished product is no sufficient means of ensuring high quality. Already the first intermediate result has to be investigated for deviations from the quality goals and measures have to taken for correction at an early stage. Also an interaction between constructive and analytical quality measures is required. While constructive methods are advised during the implementation activities of a phase, it should be followed by the corresponding analysis. The interlocking of design and test steps is also possible in development processes that do not provide for any explicit phases (e.g. extreme programming). A key factor in ensuring the intended quality lies in the early definition of these quality goals. It constitutes not of defining the requirements, but the specification of the desired quality features. This has to happen even before the phase of requirement-definition, as the requirements themself are affecte by aformentioned quality goals. On the other hand, testing results against quality features is also of central importance. The typical approach of every devloper is, to call a written program with a few sets of inputs and observe the program for expected, or divergent behaviour. This already constitutes for an informal danymic test. Inspecting the code after, implementing it, for structural errors is the informal equivalent of a static analysis. As these informal methods are widespread among programmers, employing formal processes of testing is rather disregarded among programmers, as well as the knowledge about their effectiveness. Ideally, testing is aimed at generating reproducible results, while following well defined procedures.

In the course of this chapter, techniques, tools, and procedures are discussed, to ensure software quality from an analytical point of view. Testing techniques are the main focus here. Additionally to the class od danymic tests, static analysis as well as formal methods are discussed. While object-oriented techniques for design, programming and analysis are state-of-the-art among server- and desktop-applications, embedded projects still are not generally written, employing OOP-methods. Nevertheless, either object-oriented and ebedded aspects are discussed with regards to testing and qual-

ity assurance. Further aspects of embedded projets often are certain levels of security and reliability to be guaranteed by the software. So these apsects are discussed in an upcoming **To-Do:** chapter/subsection.

Depending on the type of examination, supportive software-tools are necessary and will therfore be discussed. Certain techniques, like inspection and reviews are dedicated to be done wiht support ing tools, apart from editors for reading the code. Nontheless, both kinds of examination require organizational measures to be carried, defining responsibilities and procedures. The correct execution of these measures constitute the organizational framework of testing.

While hardware quality assurance often results in quantitative results, same is not the case for software, atl east not to the same extent as for hardware. But processes exist for both worlds, to ensure systematic development, as well as quality assurance. Developers of systems integrating both hardware and software have to be aware of their differences. Also, strictly separating the quality measures for software and hardware is not an advisible way to go. The quality properties have to be specified and verified for the complete system and not just its separate modules. The test results of individual modules, usually, can not be superimposed, but the correct behaviour of the whole system has to be demonstrated. Therefore, the deviating aspects of hardware and software quality assurance have to be examined.

2.1.2 Terminology and definitions of terms

Quality, quality requirement, quality feature, quality measure

- For the term 'quality', the definition according to DIN 55350 is used: The ability of a unit, or device, to fulfill defined and derived quality requirements.
- Quality requirements again, describes the aggregate of all single requirements regarding a unit or device.
- A quality feature describes concrete properties of a unit or device, relevant for the definition and assessment of the quality. While it does not make quantitative statements, or allowed values of a property, so to say, it very well may have a hierarchical structure: One quality feature, being composed of several detailed subfeatures. A differentiation into functional and non-functional features is advised. Also features may have different importance for the customer and the manufacturer. Overarching all these aspects, features may interfere with eachother in an opposing manner. As a consequence, maximizing the overall level of quality, regarding every aspect, is not a feasable goal. The sensible goal is to find a trade-off between interfering features, and achieve a sufficient level of quality for all relevant aspects. Typical features, regarding software developmant include: Safety, security, reliability, dependability, availability, robustness, efficiency regarding memory and runtime, adaptability portability, and testability.
- Finally, the quality measure defines the quantitative aspects of a quality feature. These are measures, that allow conclusions to be drawn about the characteristics of certain quality feature. Foe example, the **To-Do:** ref MTTF (mean time to failure), is a widespread measure for reliability.

error, failure, fault

- Error, the root cause of a device or unit to fail, may originate from human mistakes or from its operation outside the specification.
- Failure, or defect is the incorrect state of a unit, either on the hard- or software-side. It may cause a fault, but not necessarily.
- Fault is the incorrect behaviour of the unit, or it's complete cease of service, observalbe by the user.
- **To-Do:** Die drei Begriffe aufpaepeln mit deifitione laut Leveson/Millinger
- Whereas these three terms have differnt definitions in varying publications, the afformentioned declarations seem to be best suited for devices containing substantiable amount of software.

correctness

Correctness is the binary feature of a unit or device, loosely described as 'the absence of failures'. **To-Do:** da is noch mher beim Liggesmayer

completeness

Completeness describes, that all functionalities, given in teh specification are implemented. This includes normal intended operation, as well as the handling of error-states.

security

Reliability

Availability

Robustness

2.1.3 Coverage metrics

2.1.4 Reviews

2.1.5 Load/Fault tests

2.1.6 PM and ReqEng

2.1.7 V-Model?

2.2 Realtime and Reliability

2.2.1 soft/firm/hard

2.2.2 Jitter

2.2.3 Zeitpunkte

deadline

laxity

execution time

2.2.4 Load/Fault Hypothesis

2.3 Theory - CI/CD

2.3.1 CI/CD with open source on BareMetal

2.3.2 reliable USB-Connectivity

Chapter 3

Requirements

Allgemeine REQs an FW, RT (und evtl design-for-testability)

Chapter 4

Implementation

4.0.1 Concept

FSM

Trigger-Diagramme

4.1 Hardware

4.1.1 STM32F4

4.1.2 Wandler, Level-Shifter, HighSider

4.2 Software tools

4.2.1 CubeIDE

4.2.2 gcov

4.2.3 valgrind

4.2.4 gitlab

4.2.5 runner

4.2.6 HIL-Setup

4.3 Firmware-Requirements

4.3.1 FW-REQ

4.3.2 Traceability-Matrix

Linking Requirements by there tags, to the SW-modules, where they are fulfilled

Chapter 5

Results

5.1 Test-Res

5.2 Coverages

5.3 Review-remakrs

5.4 Gavlo-Performance

Appendix A

Technical Details

Appendix B

Supplementary Materials

List of supplementary data submitted to the degree-granting institution for archival storage (in ZIP format).

B.1 PDF Files

Path: /

thesis.pdf . . . . . Master/Bachelor thesis (complete document)

B.2 Media Files

Path: /media

*.ai, *.pdf . . . . . Adobe Illustrator files
*.jpg, *.png . . . . . raster images
*.mp3 . . . . . audio files
*.mp4 . . . . . video files

B.3 Online Sources (PDF Captures)

Path: /online-sources

Reliquienschrein-Wikipedia.pdf (**WikiReliquienschrein2020**)

Appendix C

Questionnaire

Appendix D

LaTeX Source Code

References