FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

FH Aachen

Faculty Electrical Engineering and Information Technology

Information Systems Engineering

Field of specialisation Systems and management

Master Thesis

Development of a hardware and software framework for the automated characterization of permanent magnets for low-field MRI systemsg

Submitted by	**Marcel Werner Heinrich Friedrich Ochsendorf**
	Matriculation number: **3120232**
Examiner:	Prof. Dr.-Ing. Thomas Dey
Examiner:	Prof. Dr.-Ing. Volkmar Schulz
External examiner:	extbetreuer
Date:	01.01.2024

dank

Erklärung

I hereby declare that I have prepared this thesis independently and without outside assistance. Text passages, which are based literally or in the sense on publications or lectures of other authors, are marked as such. The work has not yet been submitted to any other examination authority and has not yet been published.

Aachen, _____, _____

Abstract

In the construction of low-field MRI devices based on permanent magnets, a large number of magnets are used. In order to realize a homogeneous B0 field with these magnets, which is necessary for many setups, the magnetic properties of these magnets have to be as similar to a certain degree. Due to the complex manufacturing process of neodymium magnets, the different properties, the direction of magnetization, can deviate from each other, which affects the homogeneity of the field. To adjust the field afterwards, a passive shimming process is typically performed, which is complex and time-consuming and requires manual corrections to the magnets used. To avoid this process, magnets can be systematically measured in advance. In this methodology, the recording, data storage and subsequent evaluation of the data play an important role. Various existing open-source solutions implement individual parts, but do not provide a complete data processing pipeline from aqusation to analysis and the data storage formats of these are not compatible to each other. For this use case, the MagneticReadoutProcessing library was created, which implements all major aspects of acquisition, storage, analysis, and each intermediate step can be customized by the user without having to create everything from scratch, favoring an exchange between different user groups. Complete documentation, tutorials and tests enable users to use and adapt the FRamework as quickly as possible. The framework for the characterisation of different N45 neodymium magnets, which requires the integration of magnetic field sensors, was used for the evaluation.

Inhalt

1 Introduction

1.1 Background and Motivation

1.1.1 Low-Field MRI

1.1.2 Shimming procedure

1.2 Aim of this Thesis

1.3 Structure

2 State of the art

2.1 Opensource projects

2.2 Conceptual design

- etnwicklung eines hardware uns software framework zur einfachen aquirierung von Meagnetfelddaten
- analysetools und funktionen

3 Unified Sensor

- ziel ist es einen low cost hallsensor-interface zu entwickeln welcher möglichst
- universell
- verschienee sensoren abbilden kann
- mit verschienden magneten typen und formen nutzbar
- reproduzierbar
- 1d, 2d, 3d
- integration

3.1 Sensor selection

- list of typical low cost hall sensors
- => alle i2c in der regel
-

3.2 Mechanical Structure

- 3d druck toleranztest
- magnet halterung mit kraftloser arretierung
- motoren und andere unter umstaänden magnetische teile in der nähe des sensors
- nylon schrauben, 3d druck, 3d gedruckte klemmverbindungen
- später rausrechnen durch kalibierung

```
help
=============================================================================================
> help                          shows this message
> version                       prints version information
> id                            sensor serial number for identification purposes
> sysstate                      returns current system state machine state
> opmode                        returns 1 if in single mode
> sensorcnt                    returns found sensorcount
> readsensor x <0-senorcount>  returns the readout result for a given sensor index for X axis
> readsensor y <0-senorcount>  returns the readout result for a given sensor index for Y axis
> readsensor z <0-senorcount>  returns the readout result for a given sensor index for Z axis
> readsensor b <0-senorcount>  returns the readout result for a given sensor index for B axis
> temp                          returns the system temperature
> anc <base_id>                 perform a autonumbering sequence manually
> ancid                         returns the current set autonumbering base id (-1 in singlemode)
> reset                         performs reset of the system
> info                          logs sensor capabilities
> commands                      lists sensor implemented commamnds which can be used by hal
=============================================================================================
```

Bild **3**-1: Sensors CL-Interface

3.3 Electrical Interface

- usb, ethernet
- pps input output
- multiplexer for i2c sensors alredy implemented

3.4 Firmware

- automatic sensor detation
- serial cli support for manual mode
- sync impulse => 1 mastersensor als taktqelle
- interene mittelung und speichern der werte im buffer
- was durch den user implementiert werden muss klasse

3.4.1 CLI Interface

- einfache bedienung durch nutzer auch ohne weitere software
- configuration
- debugging

```
readsensor b 0
3279.99
```

Bild **3**-2: Query sensors b value using CLI

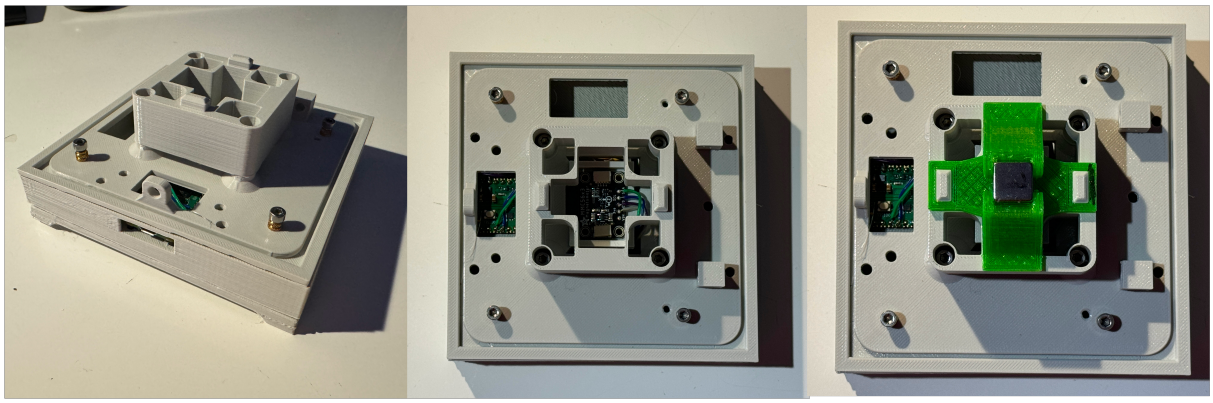

Bild **3**-3: 1D sensor contrsuction with universal magnet mount

3.5 Sensor Syncronsisation

3.6 Example Sensors

anbei werden drei erschienee sensoren für unterschiedliche anwedungfälle tablle statisch dynamisch

3.6.1 1D: Single Sensor

- einfachster aufbau rp pico + sensor

3.6.2 1D: Dual Sensor

- gleicher abstand zwei gleicher sensoren
- schnelle erkennung der plarisationsebene ggf offset vom mittelpunkt dieser

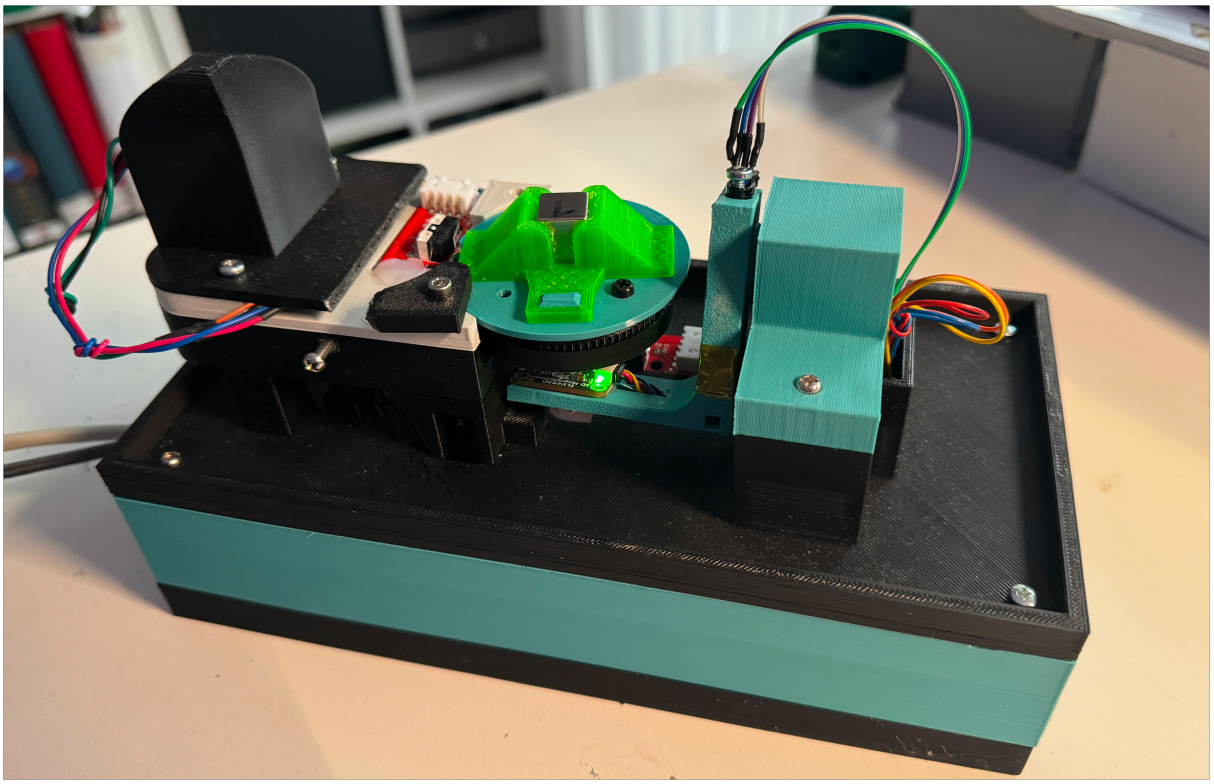

Bild **3**-4: Full-Sphere sensor implementation using two Nema17 stepper motors in a polar coordinate system

3.6.3 Full-Sphere

- komplexester aufbau sensor + mechanik
- polar mechanisches system
- full sphere sensor

3.6.4 Integration of an Professional Teslameter

- einfach anbindung professioneller teslameter
- Voltkraft

4 Software readout framework

4.1 Library requirements

4.1.1 Concepts

- beispiele für projekte welche nur einzelne schnritte implementieren
- so kann man sich auf die implementierung

4.1.2 User interaction points

- grafik zeigen
- einzelne module erleutern

HAL

- aufbau hal im grunde wird nur ein die commandos an das sensor cli weitergegeben
- alle sensoren implementieren mehr oder weniger die gleichen befehle
- hal gibt nur weiter und ist “dumm”

Visualisation

4.1.3 Export

- format import export

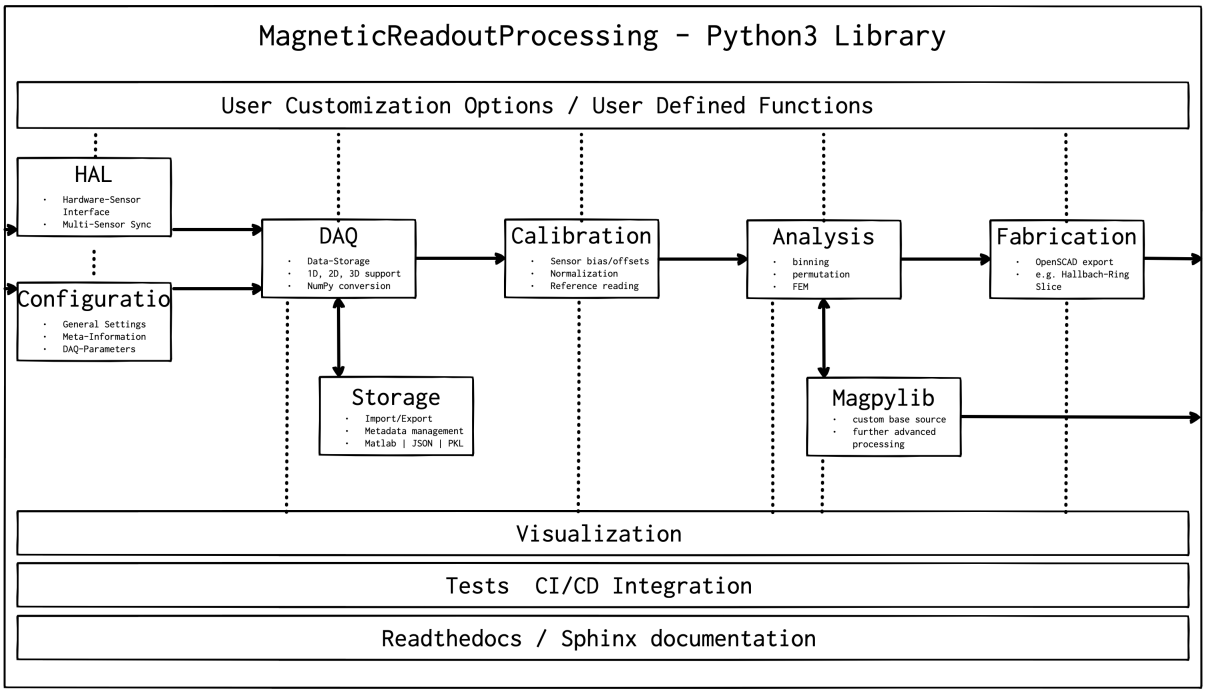

Bild **4**-1: MRPlib COMPLETE FLOW

- matlab

Meta-Data

4.1.4 Multi-Sensor setup

At the moment, it is only possible to detect and use sensors that are directly connected to the PC with the library. This has the disadvantage that there must always be a physical connection. This can make it difficult to install multiple sensors in measurement setups where space or cable routing options are limited. To make sensors connected to a small remote PC available on the network, the ‘Proxy’ module has been developed. This can be a single board computer (e.g. a Raspberry Pi). The small footprint and low power consumption make it a good choice. It can also be used in a temperature chamber. The approach of implementing this via a REST interface also offers the advantage that several measurements or experiments can be recorded at the same time with the sensors.

Another application example is when sensors are physically separated or there are long distances between them. By connecting several sensors via the proxy module, it is possible to link several instances and all sensors available in the network are available

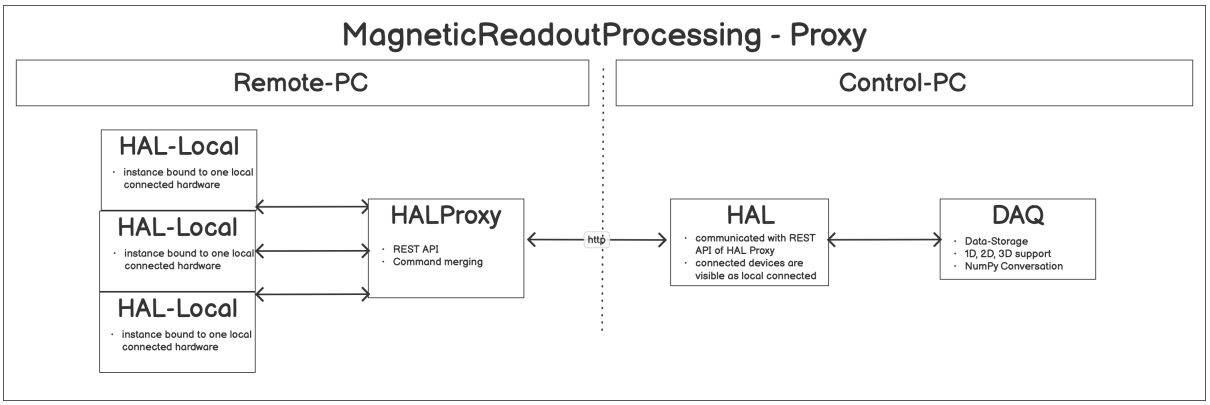

Bild **4**-2: MRPlib Proxy Module

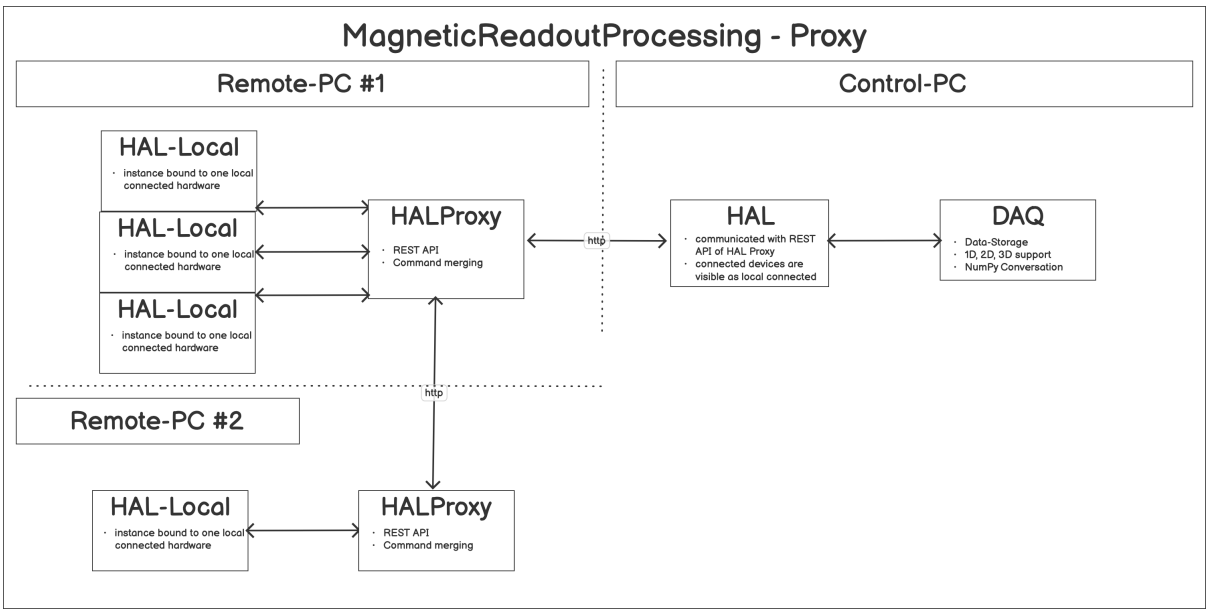

Bild **4**-3: mrp proxy multi

to the control PC.

The graphic **4**-3 shows the modified multi-proxy - multi-sensor topology. Here, both proxy instances do not communicate directly with the control PC, but remote PC #2 is connected to remote PC #1. This is then visible as a sensor opposite the Control PC, even if there are several proxy instances behind it.

Network-Proxy

The graphic **4**-2 shows the separation of the various HAL instances, which communicate with the physically connected sensors on the remote PC and the control PC side, which

communicates with the remote side via the network. For the user, nothing changes in the procedure for setting up a measurement. The proxy application must always be started on the remote PC side.

```
# START PROXY INSTNACE WITH TWO LOCALLY CONNECTED SENSORS
$ python3 mrpproxy.py proxy launch /dev/ttySENSOR_A /dev/
   ttySENSOR_B # add another proxy instance http://
   proxyinstance_2.local for multi-sensor, multi-proxy
   chain
Proxy started. http://0.0.0.0:5556/
PRECHECK: SENSOR_HAL: 1337 # SENSOR A FOUND
PRECHECK: SENSOR_HAL: 4242 # SENSOR B FOUND
Terminate  [Y/n] [y]:
```

After the proxy instance has been successfully started, it is optionally possible to check the status via the REST interface:

```
# GET PROXY STATUS
$ wget http://proxyinstance.local:5556/proxy/status
{
"capabilities":[
  "static",
  "axis_b",
  "axis_x",
  "axis_y",
  "axis_z",
  "axis_temp",
  "axis_stimestamp"
],
"commands":[
   "status",
   "initialize",
   "disconnect",
   "combinedsensorcnt",
   "sensorcnt",
   "readsensor",
   "temp"
]

# RUN A SENSOR COMMAND AND GET THE TOTAL SENSOR COUNT
$ wget http://proxyinstance.local:5556/proxy/command?cmd=
   combinedsensorcnt
{
"output":[
   "2"
]
}
}
```

The query result shows that the sensors are connected correctly and that their capabilites

have also been recognised correctly. To be able to configure a measurement on the other, only the IP address or host name of the remote PC is required:

```
# CONFIGURE MEASUREMENT JOB USING A PROXY INSTANCE
$ python3 mrpproxy.py proxy launch /dev/ttySENSOR_A /dev/
   ttySENSOR_B
```

Sensor Syncronisation

Another important aspect when using several sensors via the proxy system is the synchronisation of the measurement intervals between the sensors. Individual sensor setups do not require any additional synchronisation information, as this is communicated via the USB interface. If several sensors are connected locally, they can be connected to each other via their sync input using short cables. One sensor acts as the central clock (see chapter 3.5). However, this no longer works for long distances and a diversion must be made via the network connection.

If time-critical synchronisation is required, PTP and PPS functionality can be used on many single-board computers (such as the RaspberryPi Compute Module).

- was ptp, bild pps output
- alle clients über ptp verbunden
- dso bild von jeff gerling über rpi4 ptp

Command-Router

- nummerieierung zuerst lokale sensoren dann weitere proxy sensoren
- commando templating

4.1.5 Examples

5 Usability improvements

5.1 Commandline interface

- automatische sensor dedettion
- planung verschiedener messungen mit untersch. hardware
- zentrale abarbeitung

5.2 Programmable data processing pipeline

- datenanalyse für nicht programmieruer
- automatisierter aufbau der call-tree
- mit typcheck
- alle funktionen mit bestimmer signtur werden automatisch aus globals geladen und stehen nutzer zur verfühung

```
settings:
 enabled: true
 export_intermediate_results: false
 name: pipeline_analyse_fullsphere

stage import:
 function: import_readings
 parameters:
   IP_input_folder: ./readings/fullsphere/
   IP_file_regex: 360_(.)*.mag.json

stage import_bias_reading:
 function: import_readings
 parameters:
   IP_input_folder: ./readings/fullsphere/
   IP_file_regex: BIAS_(.)*.mag.json

```

```
18 stage apply_bias_offset:
19  function: apply_sensor_bias_offset
20  parameters:
21    bias_readings: stage import_bias_reading
22    readings_to_calibrate: stage import
23
24 stage plot_normal_bias_offset:
25  function: plot_readings
26  parameters:
27    readings_to_plot: stage apply_bias_offset
28    IP_export_folder: ./readings/fullsphere/plots/normal/
        calibrated/
29    IP_plot_headline_prefix:  Sample N45 12x12x12 magnets
        calibrated
30
31 stage plot_fullsphere:
32  function: plot_fullsphere
33  parameters:
34    readings_to_plot: stage apply_bias_offset
35    IP_export_folder: ./readings/fullsphere/plots/normal/
        calibrated/
36    IP_plot_headline_prefix:  Sample N45 12x12x12 magnets
        calibrated
```

5.2.1 Web base pipeline configuration

- drag& drop system
- automatische convertierung zum yaml system

5.3 Package management

- verteilung durch paketmanager
- docker cli beispiel

5.3.1 Documentation

6 Evaluation

6.1 Prequesites for evaluation

6.2 Evaluation confiugration

6.2.1 Sensor readout

6.2.2 Processing pipeline

6.3 Test scenarios

6.4 Results

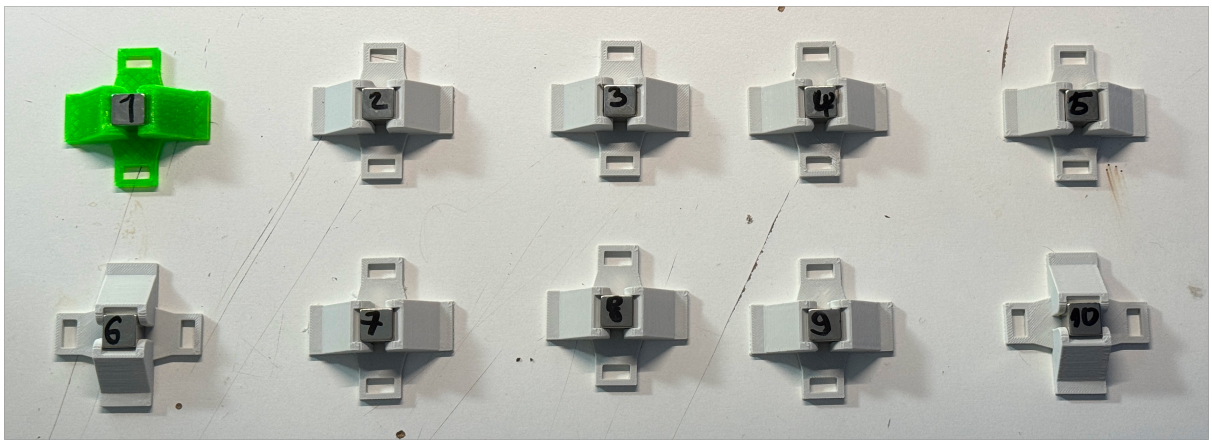

Bild **6**-1: testmagnets in holder

7 Conclusion and dicussion

7.1 Conclusion

7.2 Problems

7.3 Outlook

- magfield camera

Literaturverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis