Institut für Hochfrequenztechnik

RWTH AACHEN UNIVERSITY

Seminararbeit

Entwicklung einer grafischen Benutzeroberfläche für die Remote-Ansteuerung des Feldstärkemessgeräts SRM-3006

von Phil Träger
Matr.-Nr. 3518597

Betreut durch:
Prof. Dr. Hans-Joachim Krause
Thanh Tam Julian Ta, M.Sc.

Seminararbeit
Aachen, den
18. Januar 2024

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die Seminararbeit mit dem Thema

Entwicklung einer grafischen Benutzeroberfläche für die Remote-Ansteuerung des Feldstärkemessgeräts SRM-3006

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Name: Phil Träger

Aachen, den 18.01.2024

Unterschrift der Studentin / des Studenten

Kurzfassung

Die Seminararbeit widmet sich der Neuentwicklung einer GUI für das Selective Radiation Meter *SRM-3006*, welches am *Institut für Hochfrequenztechnik* (IHF) im Bereich der Elektromagnetischen Umweltverträglichkeit (EMVU), insbesondere im Mobilfunk eingesetzt wird. Das *SRM-3006* ist ein Feldstärke-Messgerät, welches unterschiedliche Messoptionen hat und über die GUI die Möglichkeit bieten soll, Langzeitmessungen mit einer visuellen Ausgabe der Messwerte durchführen zu können.
Für die Entwicklung einer neuen performanten GUI wurden zunächst das Messgerät und die bestehende GUI untersucht, um Limitierungen und Verbesserungsmöglichkeiten zu identifizieren. Anschließend wurde anhand der gestellten Anforderungen eine geeignete Programmiersprache gewählt und eine Softwarearchitektur geplant. Hierfür wurde das bewährte Model-View-Controller (MVC)-Muster angewendet für eine strukturierte und erweiterbare Programmentwicklung. Nach dem Definieren der GUI-Struktur wurde diese anschließend programmiert.
Die neu entwickelte GUI wurde mit der bestehenden GUI verglichen. Die Anforderungen an die GUI wurden erfüllt und es konnte eine deutliche Performanceverbesserung gezeigt werden. Zusätzlich konnten noch einige „Quality-of-Life"-Features implementiert werden.
Zusammenfassend konnte eine neue performantere GUI entwickelt werden, welche durch die gewählte Softwarearchitektur zukünftig erweiterbar ist.

Inhaltsverzeichnis

1 Motivation

Die Elektromagnetische Umweltverträglichkeit (EMVU) bezeichnet die Verträglichkeit elektromagnetischer Felder mit der Umwelt und den darin befindlichen Lebewesen, insbesondere dem Menschen. Deshalb beschäftigt sich das *Institut für Hochfrequenztechnik* (IHF) mit der Erfassung und Abschätzung von hochfrequenten elektromagnetischer Feldern mit dem Schwerpunkt auf dem Mobilfunk. Um die elektromagnetischen Felder zu messen, benutzt das IHF für seine Forschung das Feldstreckenmessgerät *SRM-3006*. Hierfür wurde am Institut eine grafische Benutzeroberfläche (GUI) in Matlab erstellt. Die Motivation für die Entwicklung einer neuen GUI für das *SRM-3006* liegt in der Einsicht, dass die derzeitige Lösung den spezifischen Anforderungen nicht gerecht wird. Die vorliegende Arbeit zielt darauf ab, eine neue grafische Benutzeroberfläche zu entwickeln, die nicht nur sämtliche Messoptionen des *SRM-3006* steuern kann, sondern auch eine Echtzeitdarstellung der Messdaten ermöglicht. Dabei ist es entscheidend, die Limitierungen der aktuellen Lösung zu überwinden und sicherzustellen, dass die neue Benutzeroberfläche den Ansprüchen an Erweiterbarkeit und Flexibilität entspricht.

2 Aktueller Stand

In diesem Kapitel werden zunächst das Messgerät und die identifizierten Limitationen erläutert sowie die aktuell bestehende GUI beschrieben.

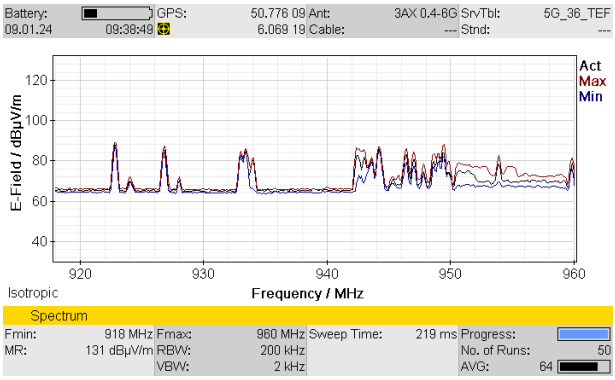
2.1 Selective Radiation Meter SRM-3006

Das Selective Radiation Meter *SRM-3006* ist ein Feldstärke-Messgerät, welches am *Institut für Hochfrequenztechnik* (IHF) für messtechnische Untersuchungen im Bereich der Elektromagnetischen Verträglichkeit der Umwelt (EMVU) genutzt wird, siehe Abb. 2.1. Aktuell liegt der Fokus im Bereich Mobilfunk, insbesondere der aktuellen Mobilfunkgeneration 5G.

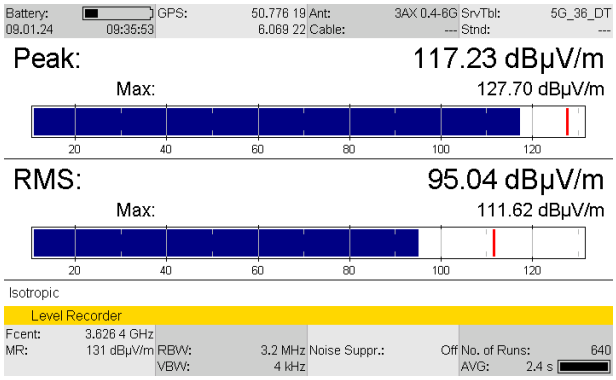
Abb. 2.1: Das Selective Radiation Meter SRM-3006

Das *SRM-3006* besitzt mehrere Messoptionen, von denen für diese Arbeit jedoch nur der „Spectrum“-Modus, der Level Recorder und die Safety Evaluation relevant sind.
Die Betriebsart „Spectrum Analysis“ bietet eine Feldstärkemessung über einen einstellbaren Frequenzbereich. Neben dem aktuellen Messwert können in diesem Modus auch durch die Max- und Min-Hold-Funktion das im gesamten Messzeitraum ermittelte Maximum und Minimum angezeigt werden. Hierbei ist zu beachten, dass die Dauer eines Messzyklus (Sweep Time) abhängig ist von der Breite des zu messenden Frequenzbereiches und anderen Parametern, auf die hier nicht näher eingegangen wird.
Die Betriebsart Level Recorder bietet eine frequenzselektive Messung bei einer definierten Frequenz. Hierbei ist der Analysator fest auf die eingestellte Frequenz abgestimmt und erfasst die Feldstärke innerhalb der eingestellten Bandbreite. Aufgrund der schmaleren messbaren Bandbreite im Gegensatz zum „Spectrum“ Modus ist die Sweep Time hier deutlich niedriger.
Der „Safety Evaluation Modus“ bietet die Möglichkeit der Bewertung der Feldstärkeexposition in einer Mehrfrequenzumgebung. Dafür wählt der Benutzer die von ihm zu messenden Dienste bzw. Frequenzbänder des Mobilfunks aus. Anschließend misst das Messgerät sukzessive die Bänder und gibt die einzelnen Feldstärkebeiträge der Bänder und die gesamte Feldexposition wieder. Eine besondere Eigenschaft dieser Betriebsart ist es, dass durch vorher eingestellte Bewertungsfaktoren die Option geboten wird, eine unmittelbare Aussage zur Einhaltung definierter Grenzwerte machen zu können. In den Abbildungen 2.2a, 2.2b und 2.2c sind die visuellen Ausgaben der drei Betriebsarten dargestellt.
Ein wichtiger Einstellparameter für alle Messoptionen ist die Measurement Range (MR). Diese bestimmt die Eingangsdämpfung und hat einen Einfluss auf die Empfindlichkeit des Systems. Je nach Messsituation muss die Eingangsdämpfung korrekt eingestellt werden, um z.B. ein Übersteuern des Gerätes durch ein sehr starkes Signal zu verhindern [Srm].

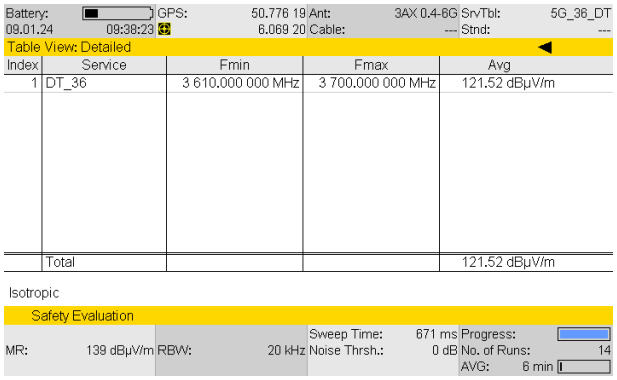
Um die gemessenen Werte auswerten zu können, bietet das *SRM-3006* die Option, diese per Knopfdruck abzuspeichern und als CSV Datei zu exportieren. Jedoch lassen sich nur die aktuell dargestellten Messergebnisse abspeichern. Für eine kontinuierliche Untersuchung der Variation der Feldstärke über einen gewissen Zeitraum ist dies von Nachteil. Um eine Langzeitmessung zu ermöglichen, lässt sich das *SRM-3006* per Remote-Ansteuerung mit einem PC verbinden, wodurch ein kontinuierliches Auslesen und Abspeichern der Messwerte möglich ist, z.B. über ein MATLAB-Skript. Bei der Nutzung im Remote-Modus zeigt das Gerät auf dem Display ausschließlich „Remoteansteuerung“ an und stellt keine Grafiken mehr dar. Dieses Problem wurde durch die Entwicklung einer in Matlab geschriebenen GUI gelöst, die die Anzeige der Graphen sowie das kontinuierliche Auslesen und Abspeichern der Messwerte ermöglicht.



(a) SRM-3006 Spectrum Analysis Modus



(b) SRM-3006 Level Recorder Modus



(c) SRM-3006 Safety Evaluation Modus

Abb. 2.2: SRM-3006 Modi

2.2 Identifizierte Limitationen

Die technischen Komponenten des *SRM-3006* sind nicht mehr auf dem aktuellen Stand der Technik und führen zu einigen Limitationen, welche in diesem Kapitel erläutert werden.
Einer der wichtigsten Faktoren ist die Sweep Time. Wenn die Sweep Time zu groß ist, ermittelt das *SRM-3006* weniger Daten in der gleichen Zeit, was zu einer schlechteren Auflösung führt. Die Sweep Time ist abhängig von der Messoption und deren Einstellungen. Durch die Remote-Ansteuerung ist die Sweep Time größer als bei der direkten Bedienung des Gerätes.
Aufgrund der unterschiedlichen Sweep Time muss für jede Messoption die Anzahl an Abfragen angepasst werden. Dies soll zum einen verhindern, dass bei zu schnellen Abfragen das *SRM-3006* keine Antwort zurückgeben kann, zum anderen soll verhindert werden, dass derselbe Messwert innerhalb einer Sweep Time mehrfach abgerufen wird.

2.3 Die Struktur der bestehenden GUI

Die GUI bietet die Möglichkeit einer visuellen Ausgabe, welche durch die Remote-Ansteuerung auf dem *SRM-3006* ausgeblendet wird. Diese wurde über die „Matlab App

Designer Toolbox“ erstellt und ist in drei Abschnitte unterteilt, siehe Abb. 2.3. Auf der rechten Seite befinden sich die Einstellungen für das *SRM-3006*. Auf der linken Seite ist der Plot dargestellt. Unter dem Plot befinden sich die Einstellungen während der Messung.

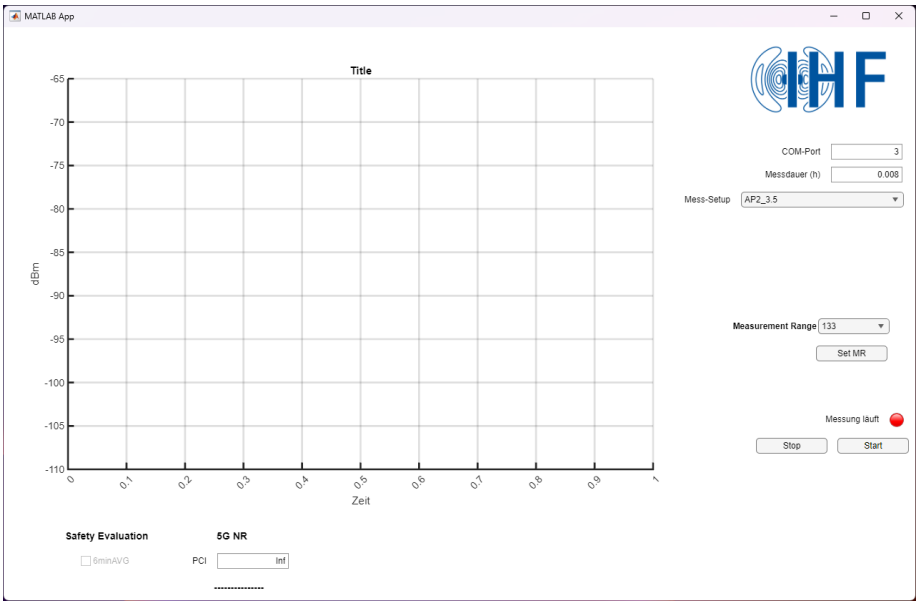


Abb. 2.3: Die Matlab GUI

Um das *SRM-3006* mit dem PC zu verbinden, muss der COM-Port ausgewählt werden. Bei der Auswahl des COM-Ports ist zu beachten, dass es keine automatische Erkennung des Ports gibt. Die COM-Port-Nummer muss vorab im Geräte-Manager des Windows-Betriebssystems eigenständig überprüft und in das Eingabefeld eingetragen werden. Darunter befindet sich die Einstellung für die Messdauer in Stunden, was für die Langzeitmessung wichtig ist. Das Measurement Setup ist eine voreingespeicherte Messoption im *SRM-3006*. Diese Einstellung kann über die GUI per Dropdown ausgewählt werden. Neue Setups müssen aber per Hand in den Code eingepflegt werden. Die Einstellung der Measurement Range ist ein wichtiger Parameter, der im vorherigen Kapitel erklärt wurde. Diese Einstellung muss situationsbedingt jederzeit einstellbar sein, um eine Übersteuerung zu verhindern. Wird während der Messung das Messgerät übersteuert, gibt die GUI eine Warnung „OVERDRIVEN“ aus. Des Weiteren ist eine Anzeige vorhanden, die darstellt, ob die Messung noch aktiv läuft. Außerdem gibt es einen Start- bzw. Stop-Button über den eine Messung gestartet oder beendet werden kann. Es gibt noch zwei optionale Einstellungen in der GUI. Die erste ist die Möglichkeit der zusätzlichen Darstellung einer Mittelung der Messwerte über sechs Minuten für den Safety Evaluation Modus. Die zweite Einstellungsoption 5G NR wird aktuell nicht mehr benötigt, weshalb dies auch in der neuen GUI nicht implementiert wird.

3 Anforderungen

Die geplante grafische Benutzeroberfläche (GUI) soll eine Steuerung der passenden Messoptionen ermöglichen und gleichzeitig die Echtzeitdarstellung der aufgezeichneten Messwerte gewährleisten. Ziel ist es, eine Analyse der Limitierungen des *SRM-3006*, und die grafische Ausgabe des Messgerätes während der Remote-Steuerung auf einem Laptop zu optimieren, insbesondere in Bezug auf die Fernsteuerung und der grafischen Ausgabe. Dies bildet die Grundlage für die Entwicklung einer Benutzeroberfläche. Es muss eine geeignete Programmiersprache für die GUI-Entwicklung gewählt werden, welche die Integration des *SRM-3006* und eine strukturierte Erweiterung des Programms ermöglichen soll.
Die geplante Benutzeroberfläche soll folgende Funktionen umfassen:

- Laden und Abspielen gespeicherter Mess-Setups.
- Speichern von Messdaten im CSV-Format während des Betriebs.
- Ändern und Anpassen der Einstellung der Measurement Range nach dem Laden.
- Bei Übersteuerung der Messung soll eine Warnung ausgegeben werden.
- Einstellbare Messdauer für Langzeitmessungen.
- Echtzeit grafische Darstellung der Messwerte über die Zeit.
- Die grafische Ausgabe sollte in Intervallen von mindestens 7 Minuten erfolgen, falls möglich auch länger.
- Zusätzliches Einblenden des sechs-minütigen Mittelwertes in der grafischen Darstellung für den Safety Evaluation Modus

Die GUI sollte so strukturiert sein, dass sie problemlos erweitert werden kann. Dies beinhaltet die Berücksichtigung möglicher zukünftiger Updates oder Ergänzungen an den Funktionen des *SRM-3006*. Die erfolgreiche Umsetzung dieser Anforderungen wird eine optimierte und benutzerfreundliche Fernsteuerung des *SRM-3006* ermöglichen, sowohl für Langzeitmessungen als auch für die Darstellung der Messdaten in Echtzeit.

4 Technologische Implementierung

4.1 Verwendete Technologien und Frameworks

Für einen strukturierten Projektverlauf wurde zunächst untersucht, welche Technologien benötigt werden, um eine „Basis“ zu identifizieren, auf der die GUI aufgebaut werden kann. Die Anforderungen deuten darauf hin, dass eine Lösung gesucht wird, die sowohl benutzerfreundlich als auch leistungsstark für die Entwicklung ist. Da bisher Matlab als Programmiersprache genutzt wurde, ist zur besseren Integration eine Technologie von Vorteil, die gut mit Matlab zusammenarbeitet. Matlab bietet eine API für die Sprachen C/C++, Fortran, Java und Python [TM]. Da Fortran eine recht alte Sprache ist und somit nur noch sehr selten in neueren Projekten benutzt wird und Java recht ressourcen-intensiv ist, standen nur noch C++ und Python zur Wahl. Nach einer sorgfältigen Abwägung zwischen C++ und Python fiel die Wahl auf Python [Lan08]. Python bietet die Vorteile einer breiten Verfügbarkeit von Ressourcen sowie der leistungsstarken PySide6-Bibliothek, die auf PyQt6 aufbaut und sich besonders für die Entwicklung einer grafischen Benutzeroberfläche (GUI) eignet [Pyq]. Da das Messgerät über eine USB-Verbindung verfügt, ist eine adäquate Lösung für die Datenkommunikation erforderlich. Hierfür wurde die PySerial-Bib genutzt, die aufgrund ihrer häufigen Verwendung für USB-Geräte und der Bereitstellung von COM-Port-Schnittstellen als geeignet erscheint [Pys].
Die Kombination aus Python, PySide6 und PySerial bietet somit eine solide Grundlage für die Entwicklung einer effizienten, benutzerfreundlichen GUI, die nicht nur den aktuellen Anforderungen gerecht wird, sondern auch Raum für zukünftige Erweiterungen und Integrationen bietet.

4.2 Softwarearchitektur und Designprinzipien

Zur Planung der Software und damit der neuen GUI wurde ein Use-Case-Diagramm erstellt, siehe Abb. 4.1. Hierbei ist zu beachten, dass das Use-Case-Diagramm nur die Interaktion zwischen dem Benutzer und dem Programm zeigt, sowie das Programm zum *SRM-3006* selber.
Im Use-Case-Diagramm sind zwei Aktoren zu sehen, das *SRM-3006* und der Benutzer. Der Benutzer kann die GUI bedienen und das *SRM-3006* kann Daten an die GUI senden. Die GUI kann die Daten vom *SRM-3006* anzeigen und die Daten abspeichern.
Nach dem Analysieren des Use-Case-Diagramms richtet sich nun der Fokus auf die Gestaltung der Softwarearchitektur, siehe Abb. 4.2. Die Softwarearchitektur nimmt einen zentralen Stellenwert in der Softwareentwicklung ein, da sie das Fundament für die Implementierung legt.

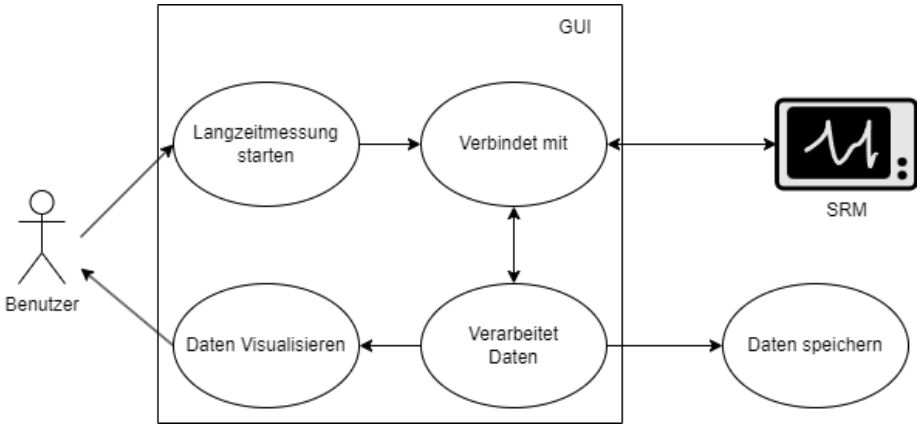


Abb. 4.1: Das Use-Case-Diagramm

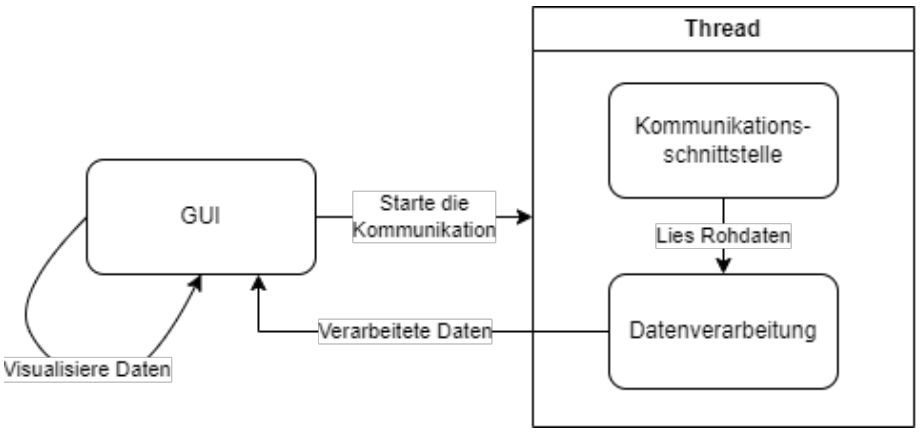


Abb. 4.2: Die Softwarearchitektur

Diese Architektur gliedert sich in drei Hauptkomponenten.
Die erste Komponente ist die GUI, welche die grafische Benutzeroberfläche repräsentiert und für die Interaktion mit dem Benutzer verantwortlich ist. Sie soll nicht nur die Kommunikationsschnittstellen und die Datenverarbeitung steuern, sondern auch die Daten in visueller Form darstellen.
Bei der zweiten Komponente handelt sich um die Kommunikationsschnittstelle, welche die Kommunikation zwischen dem PC und dem *SRM-3006* ermöglicht. Die Daten des *SRM-3006* werden von der Kommunikationsschnittstelle zusammengefasst und an die Datenverarbeitung weitergegeben.
Die dritte Komponente ist die Datenverarbeitung, welche die Daten aus der Kommunikationsschnittstelle verarbeitet und an die GUI weitergibt. Diese Daten werden in der Art verarbeitet, dass sie nicht nur einfacher für die GUI, sondern auch für die Speicherung in CSV-Dateien zu verarbeiten sind.
Die vorliegende Softwarearchitektur folgt dem bewährten Model-View-Controller (MVC)-Muster, das sich als einer der Standards in der Softwareentwicklung etabliert hat [Sta+23].
Die Komponenten sind wie folgt zugeordnet:

- Model: Datenverarbeitung
  → Sie ist für die Verarbeitung der empfangenen Daten zuständig und stellt sicher, dass die GUI mit den korrekt aufbereiteten Informationen versorgt wird.
- View: GUI (Grafische Benutzeroberfläche)
  → Sie ist für die Darstellung der grafischen Benutzeroberfläche verantwortlich und ermöglicht die Anzeige von Daten.
- Controller: Kommunikationsschnittstelle
  → Sie erleichtert die Kommunikation zwischen der GUI (View) und der Datenverarbeitung (Model) und sorgt für die Koordination des Datenaustauschs.

Die Anwendung des MVC-Musters ermöglicht eine klare Trennung der Verantwortlichkeiten zwischen Benutzeroberfläche, Anwendungslogik und Datenverarbeitung, was zu einer übersichtlichen und gut strukturierten Softwarearchitektur führt.

4.3 Datenerhebung und -verarbeitung

Um Daten zu erheben, muss zunächst die Kommunikationsschnittstelle implementiert werden. Hierfür wurde die Serial Command API des *SRM-3006* analysiert. Die API ist eine Sammlung von Befehlen, die das *SRM-3006* versteht und ausführen kann. Die Befehle sind im Handbuch des *SRM-3006* aufgelistet und beschreiben die rückgesandten Daten und deren Syntax.
Für eine Langzeitmessung ist eine kontinuierliche Abfrage der Messwerte nötig. Dies ist nur über das Versetzen des Gerätes in den Remotezustand möglich. Hierfür wird der Befehl `REMOTE ON;` verwendet. Nun ist es möglich, die Messung mit der Auswahl des Messsetups, die zuvor über das *SRM-3006* ausgelesen wird, und dem Betätigen des Startbuttons zu starten.
Um die Befehle simpler zu gestalten, wurde eine Serial Command API entwickelt, siehe Listing 4.1.

```
1  # Funktion zur Aktivierung des Remote-Modus auf dem SRM-3006
2  def set_remote_on(s: serial.Serial):
3      return command("REMOTE ON;", s)
4
5  # Funktion zur Deaktivierung des Remote-Modus auf dem SRM-3006
6  def set_remote_off(s: serial.Serial):
7      return command("REMOTE OFF;", s)
8
9  # Hauptfunktion zur Ausführung von Befehlen passend zum SRM-3006
10 def command(cmd, s: serial.Serial):
11     try:
12         # Senden des Befehls als UTF-8-kodierte Zeichenfolge an das
               Gerät
13         s.write(cmd.encode('utf-8'))
14
15         # Lesen der Antwort des SRM-3006 bis zum ersten Semikolon
16         line = s.read_until(b';').decode("utf-8")
17
```

```
18          # Bereinigen der Antwort mit bekommen des Fehlercodes
19          line = line.replace(";", "")
20          lineSplit = line.split(",")
21          errorCode = int(lineSplit[len(lineSplit) - 1])
22
23          error = None
24
25          # Prüfen, ob ein Fehlercode existiert
26          if errorCode != 0:
27              # Falls ein Fehlercode existiert, Abrufen der
                    Fehlermeldung
28              error = get_error_message(errorCode)
29
30          # Rückgabe der Antwort und der Fehlermeldung (kann None sein)
31          return line, error
32
33      # Behandlung von Ausnahmen (Exceptions)
34      except:
35          # Rückgabe eines leeren Strings und einer Meldung über
                fehlendes SRM-Gerät
36          return "", "No SRM device"
```

Listing 4.1: Serial Command API

Die Serial Command API ist eine Methode, die einen Befehl als Parameter nimmt und diesen an das *SRM-3006* sendet. Als Rückgabe werden die Informationen, wie z.B. die Setup-Werte und Messergebnisse, zurückgegeben. Wenn vorhanden, werden auch Fehlermeldungen zurückgegeben. Bei einer Messung spielt die Abfragegeschwindigkeit eine wichtige Rolle. Je nach Modus gibt es eine andere Sweep Time, welche die Abfragegeschwindigkeit bestimmt. Ist die Abfragegeschwindigkeit sehr hoch, wirft das *SRM-3006* eine Fehlermeldung aus, was vermieden werden sollte.
Nach dem Erhalt der Daten werden diese an die Datenverarbeitung weitergegeben. Die Daten kommen in Form eines Strings an und werden via Komma und Semikola unterteilt. Dieser String wird in ein Array umgewandelt. Hierfür muss zunächst überprüft werden, welcher Modus ausgewählt wurde, da die Daten in den verschiedenen Modi unterschiedlich aufgebaut sind. Nach jedem Erhalt der Daten wird ein Timestamp zugeordnet. Dieser wird von der Systemzeit genommen und in dem Array gespeichert. Anschließend wird das Array in einer Liste gespeichert. Dabei muss geprüft werden, ob Daten redundant vorkommen. Hierbei werden die Sweep-Counter untereinander verglichen, da der Counter bei jedem Sweep um eins erhöht wird.
Nachdem das Array der Liste hinzugefügt wurde, wird diese zur Speicherung weitergegeben. Aus dieser Liste werden nur der „aktueller Wert“ und der „Timestamp“ an die GUI weitergegeben.
Wie in Kapitel 3 erwähnt, soll eine optionale Einstellung für den Safety Evaluation Mode implementiert werden, der eine 6-minütige Mittelung der Messwerte ermöglicht und zusätzlich darstellt. Diese Mittelung wird in der Datenverarbeitung berechnet und an die GUI weitergegeben.

4.3.1 Abspeichern der Daten

Vor dem Starten der Messung wird eine Datei mit der Bezeichnung `Messung_Datum_Uhrzeit.csv` angelegt. Nach Erhalt des Arrays wird dieses in die CSV-Datei gespeichert. Dadurch, dass die Daten in einem Array gespeichert werden, müssen die Daten wieder in einen String im passenden CSV Format umgewandelt werden.
Durch das Abspeichern der Daten in einer CSV-Datei ist es einfacher, sie in Matlab oder Excel weiterzuverarbeiten.

4.4 Visualisierung

Die Visualisierung der Daten ist ein wichtiger Bestandteil der GUI. Hierfür wurde die Bibliothek qtgraph genutzt. Diese ist eine Erweiterung von PySide6.
Durch die Weitergabe der Daten an die GUI wird ein Live-Plot ermöglicht. Hierfür wurden die GUI und die Datenverarbeitung/-erhebung in zwei verschiedene Threads aufgeteilt, und es wurde mit der Klasse Signals gearbeitet. Signals wird für Threading genutzt, um die Kommunikation und Synchronisation zwischen verschiedenen Threads zu ermöglichen. Dies koordiniert Ausführungen und Parallelisierung von Aufgaben, wie z.B. das Weiterleiten von Daten aus dem Datenerhebungs-Thread zu dem Visualisierungs-Thread. Um nun die Daten zu visualisieren, wird der Plot mit den Daten gefüllt und aktualisiert. Große Probleme bei der Visualisierung traten im Level-Recorder-Mode auf, da hier die Daten sehr schnell aktualisiert wurden, die durch die geringe Sweep-Time des Modus zustande kam, und es zu einer Überlastung des Plots führte. Dieses Problem wurde durch eine Implementation von qtgraph gelöst, die es ermöglicht, die Daten zu puffern und erst nach einer bestimmten Zeit zu aktualisieren. Diese Zeit wird von qtgraph automatisch berechnet und angepasst.

In der Abbildung 4.3 ist ein Screenshot der neuen GUI zu sehen, welche eine Level Recorder Messung darstellt. Die Visualisierung der Daten ist ausschließlich auf den „aktueller Wert“ und den „Timestamp“ beschränkt, außer im Safety-Evolution-Mode. Hier wird zusätzlich noch der sechs-minütige Mittelwert angezeigt, wenn dieser Mittelwert gewünscht ist.

4.5 Fehlerbehandlung

Bei der Verbindung, der Auswahl des Messsetups und der Messung können Fehler auftreten. Hierfür wurde eine Fehlerbehandlung implementiert, die diese Fehler abfängt und an die GUI weitergibt.
Bevor die Messung gestartet wird, wird überprüft, ob das *SRM-3006* angeschlossen ist. Ist dies nicht der Fall, wird die Fehlermeldung `No SRM device` ausgegeben. Um die Auffälligkeit zu erhöhen, wird die Fehlermeldung mit einem roten Feld hinterlegt. Alle anderen Fehlermeldungen werden nach demselben Prinzip angezeigt. Wenn das *SRM-3006* angeschlossen ist und ein Setup geladen und die Messung gestartet wird, können ebenfalls jeweils Fehler auftreten. Die möglichen Fehler werden im Handbuch des *SRM-3006* aufgelistet und beschrieben. Dies wurde im Programm mit einer simplen Methode gelöst, die

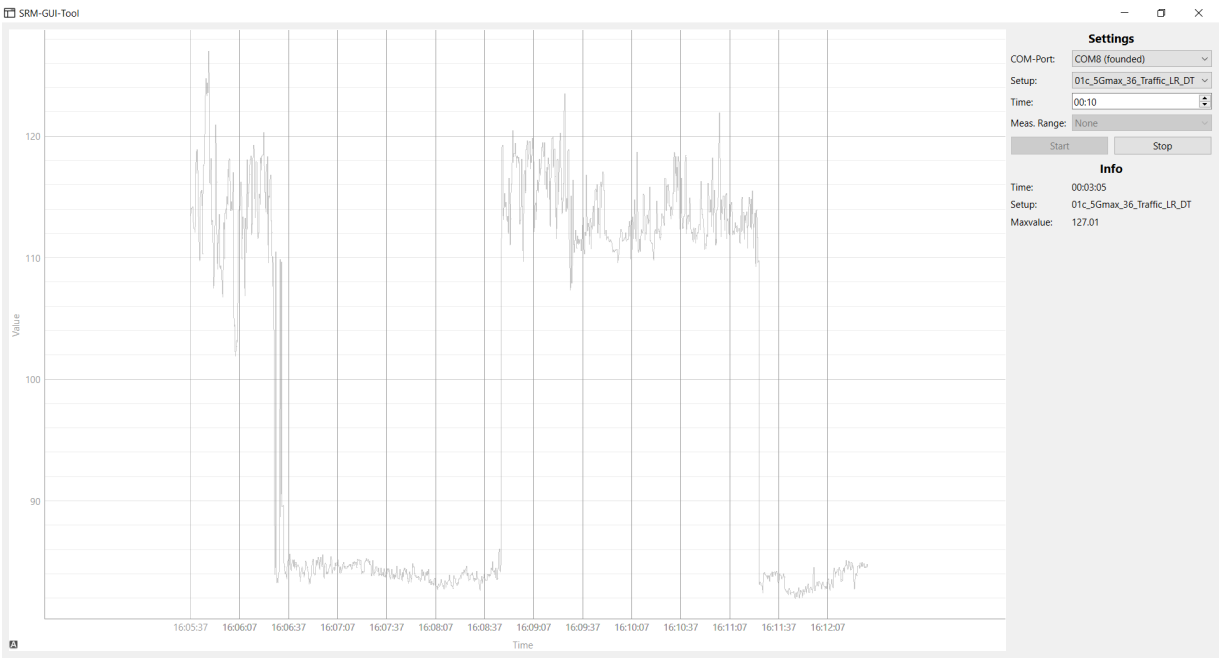


Abb. 4.3: Level Recorder Messung in der neuen Python GUI

einen Fehlercode als Parameter nimmt und die entsprechende Fehlermeldung zurückgibt, siehe Listing 4.2.

```
def get_error_message(code):
    errorMessages = {
        401: "Remote command is not implemented in the remote module",
        402: "Invalid parameter",
        403: "Invalid count of parameters",
        404: "Invalid parameter range",
        405: "Last command is not completed",
        406: "Answer time between remote module and application module is too high",
        407: "Invalid or corrupt data",
        408: "Error while accessing the hardware",
        409: "Command is not supported in this version of the application module",
        410: "Remote is not activated (please send 'REMOTE ON;' first)",
        411: "Command is not supported in the selected mode",
        412: "Memory of data logger is full",
        413: "Option code is invalid",
        414: "Incompatible version",
        415: "Subindex full",
        416: "File counter full",
        417: "Data lost",
        418: "Checksum error",
        419: "Programming not successful",
        420: "Path not found",
        421: "Break detected",
```

```
24          422: "Low battery",
25          423: "File open error",
26          424: "Data verify error"
27      }
28
29      if code in errorMessages:
30          return f"{code} {errorMessages[code]}"
31      else:
32          return f"Unknown error code: {code}"
```

Listing 4.2: Fehlerbehandlung

Wie im Kapitel 2.1 Bereits erwähnt, kann das SRM auch beim Messen übersteuern. Dies gibt das Messgerät als „OVERDRIVEN“ Parameter in den ausgelesenen Daten an. Dieser Indikator wird ausgelesen und an die GUI gesendet. Die GUI zeigt dies auch in einem rot hinterlegten Feld mit dem Text „OVERDRIVEN“ an. Um „OVERDRIVEN“ zu beheben, muss die Measurement Range angepasst werden. Dadurch wird eine neue CSV-Datei erstellt, die derzeitige Messung beendet und mit der neuen Measurement Range gestartet.

5 Ergebnisanalyse

Ziel der Arbeit ist es nun, die neue GUI mit der alten GUI zu vergleichen. Die erste Anforderung war eine Echtzeitdarstellung der aufgezeichneten Messwerte. In der alten GUI war eine flüssige Darstellung aufgrund der genutzten Plot-Funktion nicht immer gegeben. Die Hauptursache dafür war, dass bei einer niedrigen Sweep Time die Timestamps zu nah aneinander liegen, dass Matlab diese auf denselben Wert rundet. Für den Plot mussten dementsprechend die Werte in einem zeitlichen Abstand gefiltert werden, das keine Rundungsfehler auftreten. Dies hat jedoch zur Folge, dass nicht alle Messwerte dargestellt werden können und eine flüssige Darstellung aller Werte nicht möglich war. Bei der neuen GUI hat sich hier die Performance durch die Verwendung von *qtgrath* deutlich verbessert. Dadurch das hier keine Rundungsfehler mehr auftreten, werden für das Plotten keine Daten weggeworfen. Eine weitere Verbesserung der visuellen Darstellung der Werte über einen Zeitraum von sieben Minuten ist durch die Implementierung einer besseren Skalierung ersichtlich. Die alte GUI skalierte die Zeitangaben automatisch, sodass bei geringen Darstellungspunkten die Skalierung sich verschiebt, bis genügend Darstellungspunkte vorhanden sind. Bei der neuen GUI ist dies nicht der Fall aufgrund einer vordefinierten Skalierung, bei der im 30-Sekunden-Abstand eine neue Linie im Darstellungsgrid gezeichnet wird. Dies bringt eine bessere Übersicht über die dargestellten Messwerte. Eine Quality-of-Life Verbesserung in der neuen GUI ist die automatische Erkennung und Auswahl des Messgerätes, welches durch die Verwendung von `pySerial` ermöglicht wurde. Bei der alten GUI war es nötig, über den Geräte-Manager des Windows-Betriebssystems die richtige COM-Port-Nummer rauszusuchen. Die alte GUI konnte die Messdaten in Matlab-Dateien abspeichern. Durch die Verwendung von CSV-Dateien in der neuen GUI ist es möglich, die abgespeicherten Messdaten in verschiedenen Programmen leichter weiter zu verarbeiten. Des Weiteren ist es in der neuen GUI im Gegensatz zur alten GUI möglich, die Messung zu stoppen. Dies spart bei Langzeitmessungen Zeit, da diese unterbrochen werden können, falls eine zu lange Messdauer vorher eingestellt wurde. Darüber hinaus wurden beim vorzeitigen Beenden der alten GUI die Messdaten nicht gespeichert. Das Abspeichern der Messdaten in der alten GUI erfolgt erst nach dem Durchlauf der eingestellten Messdauer. In der neuen GUI werden die Messdaten kontinuierlich abgespeichert, sodass ein frühzeitiges Beenden kein Verlust der Messdaten verursacht. In seltenen Fall, dass die neue GUI während der Messung unerwartet beendet wird, kann es dazu kommen, dass die letzten Daten der CSV-Datei korrumpiert ist, da die Schreib-Funktion möglicherweise mitten drin beendet wurde. Dies verursacht jedoch nicht den Verlust aller Messdaten.

6 Fazit und Ausblick

Die Aufgabe dieser Seminararbeit war es, eine grafische Benutzeroberfläche für die Bedienung des SRM zu entwickeln, welche die in Kapitel 3 beschriebenen Anforderungen erfüllt. Als Grundlage zur Bewertung wurde die alte GUI untersucht und mit der neuen verglichen. Zusätzlich wurden viele „Qualitiy of Life“ Verbesserungen, wie z.B. das automatische Erkennen der Messgeräte, hinzugefügt. Dies vereinfacht deutlich die Bedienung der GUI und verringert die Fehleranfälligkeit. Allgemein ist die neue GUI performanter als die alte. Ein Grund dafür ist, dass die alte GUI in Matlab entwickelt wurde und keine eigenständige Software bot, so dass zum Starten immer Matlab benötigt wurde. Die neue GUI ist eigenständig und erfordert keine zusätzlichen Programmstarts. Dies hat ebenfalls den Vorteil, dass in einer Messumgebung ohne eine Internetverbindung trotzdem eine Langzeitmessung durchgeführt werden kann, da keine zusätzliche Verbindung zum Matlab-Lizenzserver benötigt wird. Des Weiteren kann sie auch auf anderen Betriebssystemen laufen. Hierzu ist lediglich die Installation des Messgerätetreibers erforderlich. Dieser ist zurzeit ausschließlich für Windows Betriebssysteme erhältlich. Durch die Speicherung der Messdaten in CSV-Dateien wird die Flexibilität der Weiterverarbeitung erhöht. Zur Weiterverarbeitung der Daten wird nicht ausschließlich Matlab benötigt, sondern Programme wie z.B. Excel oder auch selbst entwickelte Analyse-Tools können die Daten ebenfalls einfach lesen und verarbeiten. Im Gegensatz zur alten GUI wurde die neue GUI im MVC-Muster programmiert. Dies macht sie einfacher erweiterbar, ohne den Kern der GUI zu verändern.

Die GUI ist nach den Anforderungen dieser Arbeit fertiggestellt worden, jedoch sind längst nicht alle Steuerelement des *SRM-3006* einbezogen. Neben einigen Messoptionen sowie der Modi des Gerätes können auch neue Features, wie z.B. ein Wasserfalldiagramm über die Spectrumanalyse, hinzugefügt werden. Auch die Möglichkeit, nur ausgewählte Messdaten zu speichern, können der neuen GUI noch einfach hinzugefügt werden.

Abbildungsverzeichnis

Listings

Literatur

[Ins] *Institut für Hochfrequenztechnik | Historie*. URL: https://www.ihf.rwth-aachen.de/institut/historie (besucht am 13. 12. 2023).

[Lan08] Hans Petter Langtangen, Hrsg. *Python Scripting for Computational Science*. en. Bd. 3. Texts in Computational Science and Engineering. ISSN: 1611-0994. Berlin, Heidelberg: Springer, 2008. ISBN: 978-3-540-73915-9 978-3-540-73916-6. DOI: 10.1007/978-3-540-73916-6. URL: http://link.springer.com/10.1007/978-3-540-73916-6 (besucht am 13. 12. 2023).

[Pyq] *PyQt - Python Wiki*. URL: https://wiki.python.org/moin/PyQt (besucht am 13. 12. 2023).

[Pys] *Welcome to pySerial's documentation — pySerial 3.0 documentation*. URL: https://pythonhosted.org/pyserial/ (besucht am 13. 12. 2023).

[Pyt] *Welcome to Python.org*. en. Dez. 2023. URL: https://www.python.org/ (besucht am 13. 12. 2023).

[Qt2] *Qt | Tools for Each Stage of Software Development Lifecycle*. en. URL: https://www.qt.io (besucht am 13. 12. 2023).

[Srm] *SRM-3006 field strength analyzer*. URL: https://www.narda-sts.com/de/produkte/selektiv-emf/srm-3006-field-strength-analyzer/pd/pdfs/ (besucht am 13. 12. 2023).

[Sta+23] Gernot Starke u. a. *Basiswissen für Softwarearchitekten*. Sep. 2023. ISBN: 978-3-9889007-2-2. URL: https://content-select.com/de/portal/media/view/64d9ca26-17a0-4f16-abc3-48eaac1b0007?forceauth=1 (besucht am 10. 01. 2024).

[TM] Inc. The MathWorks. *MATLAB mit anderen Programmiersprachen nutzen*. de. URL: https://de.mathworks.com/products/matlab/matlab-and-other-programming-languages.html (besucht am 04. 01. 2024).

[Wil20] Joshua M. Willman. *Beginning PyQt: A Hands-on Approach to GUI Programming*. en. Berkeley, CA: Apress, 2020. ISBN: 978-1-4842-5856-9 978-1-4842-5857-6. DOI: 10.1007/978-1-4842-5857-6. URL: http://link.springer.com/10.1007/978-1-4842-5857-6 (besucht am 13. 12. 2023).

[Wus19] Matthias Wuschek. *Measuring RF electromagnetic fields at mobile communications base station and broadcast transmitter sites*. en. Issue 1. Pfullingen: Narda Safety Test Solutions GmbH, 2019. ISBN: 978-3-00-061912-0.