I M A

Intelligent Magic Audio

Jonas Lux, Szymon Banasiak, Leon Braungardt

Sommersemester 2024

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

- Ein Projekt aus Modul ESP SS24
- Eigene Idee FURZ ALARM
- Ziel des Projekts und Motivation
- Grobe Projektbeschreibung (1-2 Sätze)

2 Detaillierte Projektbeschreibung

- Umriss des Projekts, ohne zu sehr auf Details wie Regler, Potis, exakte Display-Technologie usw. einzugehen. Dazu da, um dem Leser eine Idee zu geben, worum es geht und wie wir uns die Funktionalität vorgestellt haben.
- Bild von Faceplate
- Audiosampler für Eurorack Modularsystem (was ist das?)
- Problem: Schwierig die "richtigenSSamples zu finden ⇒ Klassifizierung der Samples
- Suchfilter über Hardware-Interface (Regler)
- Anzeige der passenden Samples auf dem Display
- Auswählen/Abspielen mit Poti
- Eurorack Standard

3 Projektanforderungen

- Lastenheft: Welche Anforderungen gibt es? Falls nicht-funktionale Anforderungen existieren: in funktionale und nicht-funktionale unterteilen; LF nummeriert.
- (Pflichtenheft wird in Spezifikation integriert)

3.1 Funktionale Anforderungen

3.1.1 LF1: Steuerung des Cursors und Auswahl eines Samples

Priorität	Muss
Akteur	Benutzer
Beschreibung	**1. Cursor Bewegung:** **Aktion:** Der Benutzer dreht den Encoder. **Reaktion:** Der Cursor auf dem LCD-Display bewegt sich nach oben oder unten entsprechend den Stepperschritten des Encoders. **Details:** - Der Cursor bewegt sich um eine Position in der Liste pro Schritt des Encoders. - Der Cursor kann den Beginn oder das Ende der Liste erreichen, ohne die Liste über die Grenzen hinaus zu bewegen. Er spring von daher am Anfang oder Ende der Liste. **2. Sample Auswahl:** **Aktion:** Der Benutzer drückt den Switch des Encoders. **Reaktion:** Das aktuell ausgewählte Sample wird hervorgehoben und sein Name wird unter der Liste auf dem LCD-Display angezeigt. **Details:** - Der Name des ausgewählten Samples wird deutlich unter der Liste angezeigt. - Der Name sollte in einem Format angezeigt werden, das gut lesbar ist und keine Abkürzungen oder Unklarheiten aufweist.

3.1.2 LF2: Verhalten der Fader und des Displays

Priorität	Muss
Akteur	Benutzer
Beschreibung	**1. Anzeige der Fader-Werte:** **Aktion:** Der Benutzer bewegt die Schiebepotentiometer. **Reaktion:** Die prozentuale Angabe der Spannung, in der sich der Potentiometer befindet, wird angezeigt. **Details:** - Die prozentualen Ausgaben spiegeln die Klassen wider, wie Rhythmic, Sustained, usw., sowie die Einstellung des Threasholds - Die Prozentsätze sollen klar und deutlich angezeigt werden, ohne Verzögerung. **2. Filtern der Samples:** **Aktion:** Der Benutzer bewegt die Schiebepotentiometer. **Reaktion:** In der Liste auf dem Display werden die Samples angezeigt dessen Audio Klassen nach der Klassiefiezierung, die den Prozentsatz der Potentiometer-einstellung nicht mehr als einen bestimmten Schwellenwert überschreiten. **Details:** - Die Anzeige der Samples soll dynamisch aktualisiert werden, basierend auf den aktuellen Fader-Werten. - Der Schwellenwert kann angepasst werden, um eine präzise Filterung der Samples zu ermöglichen.

3.2 Nicht-Funktionale Anforderungen

3.2.1 Performance

/LQ1/	**Latenz**
Kategorie	Performance
Beschreibung	Die Latenzzeit zwischen der Aufnahme eines Audio-Samples und dessen Wiedergabe sollte minimal sein (z.B. ¡10ms), um eine reibungslose Benutzererfahrung zu gewährleisten.

/LQ2/	**Echtzeitverarbeitung**
Kategorie	Performance
Beschreibung	Das System muss in der Lage sein, Audio-Daten in Echtzeit zu verarbeiten, um eine sofortige Rückmeldung und Sampleklassifizierung zu ermöglichen.

/LQ3/	**Datenübertragungsrate**
Kategorie	Performance
Beschreibung	Die Middleware muss hohe Datenübertragungsraten unterstützen, um eine kontinuierliche und verlustfreie Audioübertragung zu gewährleisten.

3.2.2 Zuverlässigkeit

/LQ4/	**Fehlertoleranz**
Kategorie	Zuverlässigkeit
Beschreibung	Das System sollte fehlertolerant sein und sich bei Hardware- oder Softwarefehlern selbstständig erholen können, um Datenverluste zu vermeiden.

/LQ5/	**Systemverfügbarkeit**
Kategorie	Zuverlässigkeit
Beschreibung	Das Gerät muss eine hohe Verfügbarkeit (z.B. 99,9%) gewährleisten, besonders in Live-Performance-Umgebungen.

3.2.3 Benutzerfreundlichkeit

/LQ6/	**Intuitive Benutzeroberfläche**
Kategorie	Benutzerfreundlichkeit
Beschreibung	Die UI muss intuitiv und leicht bedienbar sein, damit Nutzer ohne umfangreiche Schulung schnell damit arbeiten können.

/LQ7/	**Feedback-Mechanismen**
Kategorie	Benutzerfreundlichkeit
Beschreibung	Das System sollte klares und sofortiges Feedback auf Benutzeraktionen geben, um Fehlbedienungen zu minimieren.

3.2.4 Sicherheit

/LQ8/	**Datenintegrität**
Kategorie	Sicherheit
Beschreibung	Die Integrität der Audio-Daten muss zu jeder Zeit gewährleistet sein, insbesondere bei der Speicherung und Übertragung.

/LQ9/	**Zugriffskontrollen**
Kategorie	Sicherheit
Beschreibung	Nur autorisierte Benutzer sollten auf bestimmte Funktionen und Daten zugreifen können.

3.2.5 Wartbarkeit

/LQ10/	**Modularität**
Kategorie	Wartbarkeit
Beschreibung	Die Software- und Hardwarekomponenten sollten modular aufgebaut sein, um einfache Wartung und Updates zu ermöglichen.

/LQ11/	**Dokumentation**
Kategorie	Wartbarkeit
Beschreibung	Umfassende und verständliche Dokumentation für Entwickler und Benutzer sollte vorhanden sein, um Wartung und Weiterentwicklung zu erleichtern.

/LQ12/	**Plattformunabhängigkeit**
Kategorie	Portabilität
Beschreibung	Die Middleware sollte auf verschiedenen Hardwareplattformen laufen können, um die Flexibilität und Wiederverwendbarkeit zu erhöhen.

/LQ13/	**Einfache Installation**
Kategorie	Portabilität
Beschreibung	Das System sollte einfach zu installieren und zu konfigurieren sein, um den Einsatz in verschiedenen Umgebungen zu erleichtern.

3.2.6 Portabilität

3.2.7 Effizienz

/LQ14/	**Ressourcennutzung**
Kategorie	Effizienz
Beschreibung	Das System sollte sparsam mit den verfügbaren Ressourcen (CPU, Speicher, Energie) umgehen, um eine lange Betriebsdauer und Stabilität zu gewährleisten.

/LQ15/	**Optimierung der Audioverarbeitung**
Kategorie	Effizienz
Beschreibung	Algorithmen für die Audioverarbeitung und Klassifizierung sollten so optimiert sein, dass sie minimale Rechenressourcen benötigen, ohne die Genauigkeit zu beeinträchtigen.

3.2.8 Kompatibilität

/LQ16/	**Schnittstellen**
Kategorie	Kompatibilität
Beschreibung	Das System sollte mit gängigen Schnittstellen und Protokollen kompatibel sein, um eine einfache Integration in bestehende Setups zu ermöglichen.

/LQ17/	**Erweiterbarkeit**
Kategorie	Kompatibilität
Beschreibung	Das Gerät sollte leicht erweiterbar sein, um zukünftige Funktionalitäten und Verbesserungen zu unterstützen.

4 Spezifikation

4.1 Architektur

- Systemarchitektur: Gesamtdarstellung des Systems, wie Komponenten zusammenarbeiten
- Unterteilung in 3 Komponenten (Audio, User Interface, NN)
- Hier SA/RT Kontextdiagramm (evtl. ein Gesamt-Diagramm und pro Komponente ein weiteres)
- Hier SA/RT Modell für Zustandsautomat? und ggf. weitere Modelle

4.2 Technische Spezifikation

- Welche Hardware wird für welchen LF und warum benötigt?
- Warum dieser Standard/Protokoll?
- Eurorack Standard

4.2.1 Hardware

Komponenten/LF1/:

(Encoder)

Elektorisch 3.3V

Mechanisch GPIO Pins PD1 PD0 PG1

Zur Auswahl der Samples und zum Navigieren durch das Menü benötigen wir einen Rotary-Encoder mit einem Switch Button.Das Empfangen der A und B Signale des Encoders erfolgt über die Pins PD0 und PD1. Der Encoder ist so konfiguriert das wen PD0 und PD1 auf high sind der Encoder den Cursor incrementiert. Wen PD0 high und PD1 low ist wird der Cursor runtergezählt. Das Drücken des Switch über den Pin PG1. Durch das drücken des Switch-Buttons wird der Pin PG1 auf high gesetzt und die Auswahl des Samples wird übernommen.

- **Präzise Steuerung:** Der Encoder ermöglicht eine präzise Steuerung des Cursors auf dem Display.
- **Benutzerfreundlichkeit:** Der Benutzer kann durch die Liste navigieren und ein Sample auswählen. Die Kombination aus Drehbewegung und Druckknopf-Funktionalität macht den Encoder intuitiv.
- **Robustheit:** Encoder sind mechanisch robust und langlebig, was sie ideal für häufigen Gebrauch in Audio-Geräten macht.

(LCD-Display)

Elektorisch 5.0V

Mechanisch GPIO Pins PF0 PF1

Zur Visualiesierung der Sampels haben wir einen Monochronen LCD-Display benutzt. Im zusammenspiel mit dem Encoder ermöglicht es eine gute Navigation durch den gewünschten Samplepool. Die Daten werden über den Output Pin PF0 an den SDA des Displays übertragen. Der Takt wird über den Pin PF1 an den SCL übertragen. Das Display wird mit 20 FPS betrieben und mit hilfe von Timer tim5 geupdated.

- **Klare Visualisierung:** LCD-Displays bieten eine klare und gut lesbare Darstellung von Text und Grafiken.
- **Energieeffizienz:** LCDs sind energieeffizient.
- **Verfügbarkeit:** LCD-Displays sind weit verbreitet und leicht zu beschaffen, was die Beschaffungskosten senkt und die Integration erleichtert.
- **Anpassbarkeit:** Sie können einfach an verschiedene Layouts und Designs angepasst werden.

Die Kombination aus Encoder und LCD-Display bietet somit eine effiziente und benutzerfreundliche Lösung für die Steuerung und Anzeige in einem Audio Sample Recorder und Playback Device.

Komponenten/LF2/:

(Schiebe-Potentiometer)

Elektorisch 3.3V

Mechanisch GPIO Pins PA0 PA3 PA4 PC0 PC3

Für die Filterfunktion benötigen wir 5 Potentiometer. Es wird zyklisch die Ausgangsspannung des Schleifers abgegriffen die die Teilspannung zwichen den VCC und dem GND darstellt. Dies erfolgt mit Hilfe vom ADC und dem DMA. Die Auswertung der Spannung erfolgt über die Pins PA0 PA3 PA4 PC0 PC3. Die Pins PA0 PA3 PA4 PC0 sind für die Klassen zuständig PC3 für den Schwellenwert an erlaubter abweichung.

- **Präzise Steuerung und feine Abstimmung:** Ein Potentiometer ermöglicht eine stufenlose und präzise Einstellung. Durch das Schieben des Potentiometers kann der Benutzer den Cursor in kleinen, genauen Schritten bewegen.
- **Einfache Bedienung und intuitive Nutzung:** Potentiometer sind einfach und intuitiv zu bedienen.
- **Robuste Mechanik und Zuverlässigkeit:** Potentiometer sind mechanisch robust und zuverlässig. Sie können eine hohe Anzahl von Schiebezyklen ohne an Genauigkeit oder Funktionalität zu verlieren.
- **Direkte visuelle Rückmeldung:** Durch die sofortige visuelle Rückmeldung auf dem LCD-Display kann der Benutzer sofort sehen, wie sich die Bewegung des Potentiometers auf die Position des Cursors auswirkt.

(ADC)

ADC (Analog Digital Converter) wird benutzt, um die Ausgangsspannung an den Schleifern zu digitalisieren und in Zahlen zu fassen. Dies erfolgt über die Pins PA0, PA3, PA4, PC0, PC3, die als Schnittstellen parallel der Reihenfolge der Channels 0, 2, 3, 10, 13 dienen. Jedes Mal, wenn an der Ausgangsspannung eines der Schleifer eine Veränderung wahrgenommen wird, startet eine Interrupt Service Routine, die auf einer Callback-Methode basiert. Diese berechnet dann einen geglätteten Endwert für die Ausgangsspannung.

(DMA)

DMA (Direct Memory Access) wird gestartet, um die Werte, die an den Pins des ADC liegen, zyklisch zu pollen. Er wird zeitbasiert mit einem Timer tim5 gestartet und in der Callback nach der Glättung der kumulierten ADC-Werte gestoppt.

Timer: Timer5, Internal Clock, One Pulse Mode
Data Width: Word Übertragung von 32-Bit-Datenblöcken
Mode: Circular

(LCD-Display)

Der LCD-Display ist der gleiche wie im /LF01/ beschrieben. Dieser dient zur Darstellung der Fader Einstellung in prozentualer Form.

4.3 Schnittstellenbeschreibung und Integration der Komponenten

- Planung der Schnittstellen zwischen den Komponenten
- Einfaches Diagramm in DrawIO:
  - Zwischen Jonas und Leon: `downsampleandread1024()`
  - Zwischen Syzmon und Jonas: `filemanager struct`, etc.
  - Zwischen Szymon und Leon: `filemanager struct`

5 Durchführung

- Implementierung der Komponenten:
  - Ansätze/Methoden: Beschreibung der Ansätze und Methoden für jedes Teilprojekt
  - Verwendete Komponenten: Detaillierte Beschreibung der verwendeten Komponenten
  - Erkenntnisse während der Implementierung: Erfahrungen und Änderungen während der Implementierung und Begründung für Alternativen
- Integration der Komponenten: Integration der Komponenten in das Gesamtsystem (aus zeitlichen Gründen nicht erfolgt)

6 Test und Validierung

- Testfälle beschreiben, wurden LF erfüllt?
- Dokumentation des Tests und der Inbetriebnahme, Testprotokoll in der Form: Erwartetes Verhalten/gemessenes Verhalten, Checklisten
- Genau so wie in der ES Dokumentation

6.1 Testprotokoll: Encoder in Verbindung mit OLED-Display

6.1.1 Stromversorgung

- **Testfall:** Überprüfung der Stromversorgung des Encoders.
- **Schritte:**
  1. Das System würde an eine stabile Stromquelle angeschlossen. Verbindung über einen ST-Link vom Laptop.
  2. Die Spannung an den Versorgungsanschlüssen des Encoders würde mithilfe eines Multimeters gemessen.
- **Erwartete Werte:**
  - Spannung: 3.3 V $\pm$ 5
  - Stromstärke: Im spezifizierten Bereich des Encoders.
- **Testergebnisse:**
  - Die gemessene Spannung betrug 3.31 V, was innerhalb des erwarteten Bereichs von 3.3 V $\pm$ 5
  - Die Stromstärke lag ebenfalls im spezifizierten Bereich des Encoders.

Vorgehensbeschreibung: Zuerst würde das System an über ST-Link ans Laptop angeschlossen, um eine konstante Spannungsversorgung zu gewährleisten. Mit einem Multimeter würde die Spannung an den Versorgungsanschlüssen des Encoders gemessen, um sicherzustellen, dass sie im erwarteten Bereich liegt.

6.1.2 Verbindung

- **Testfall:** Überprüfung der physischen Verbindung zwischen Encoder und Mikrocontroller.
- **Schritte:**
  1. Die Kabelverbindungen würden mit einem Multimeter überprüft.
  2. Es würde sichergestellt, dass alle Pins korrekt verbunden sind.
- **Erwartete Werte:**
  - Kontinuität und korrekte Verbindung an allen relevanten Pins.
- **Testergebnisse:**
  - Alle Kabelverbindungen zeigten Kontinuität.
  - Alle relevanten Pins waren korrekt verbunden und funktionsfähig.

Vorgehensbeschreibung: Um die Verbindung zu überprüfen, haben wir zunächst ein Multimeter verwendet, um die Kontinuität der Kabelverbindungen zu testen. Die Pins des Encoders würden geprüft, um sicherzustellen, dass alle Verbindungen durchgehend und korrekt mit dem Mikrocontroller verbunden sind. Alle relevanten Pins zeigten eine korrekte Verbindung.

6.1.3 Auswertung der Drehung des Encoders

- **Testfall:** Prüfung des Drehverhaltens
- **Schritte:**
  1. Der Encoder würde schnell und langsam in beide Richtungen gedreht.
  2. Dabei würde die Zählvariable `rotary_enc_count` beobachtet.
- **Erwartete Werte:**
  - Die Zählvariable zählt wie erwartet hoch und runter.
  - Nur beim Betreten der `HAL_GPIO_EXTI_Callback` soll die Variable hochgezählt werden.
  - Es traten keine Mehrfachauslösungen auf.
  - Die Zählvariable zählte korrekt und reagierte wie erwartet auf schnelle und langsame Drehungen in beide Richtungen.

Vorgehensbeschreibung: Um das Drehverhalten des Encoders zu testen, würde de Encoder an das System angeschlossen und die Funktionalität überprüft. Anschließend haben wir den Encoder in beiden Richtungen, sowohl schnell als auch langsam, gedreht und die Zählvariable Die Zählvariable `rotary_enc_count` stand dabei unter Beobachtung, um sicherzustellen, dass sie ausschließlich durch die `HAL_GPIO_EXTI_Callback` beeinflusst wird.

6.1.4 Debouncing des Push-Buttons

- **Testfall:** Prüfung der Funktionalität des Push-Buttons.
- **Schritte:**
  1. Drücken des Push-Button gedrückt.
  2. Prüfen ob der Interrupt korrekt ausgelöst wird.
  3. Das korrekte Verhalten des Entprellen würde geprüft.
- **Erwartete Werte:**
  - Der Interrupt wird bei jedem Drücken sauber und eindeutig ausgelöst.
  - Es treten kein Prellen oder Mehrfachauslösungen auf.
- **Testergebnisse:**
  - Der Interrupt wurde bei jedem Drücken zuverlässig und eindeutig ausgelöst.
  - Es traten keine Prellen oder Mehrfachauslösungen auf.

Vorgehensbeschreibung: Der Push-Button würde mehrmals gedrückt und dabei beobachtet, ob der Interrupt korrekt ausgelöst wurde. Während des Tests wird darauf geachtet, ob der Button prellte oder Mehrfachauslösungen verursachte in dem ein Zähl Variable `cnt` im Debugger beobachte würde. Die Ergebnisse bestätigten, dass der Interrupt zuverlässig und ohne Prellen funktionierte.

6.1.5 Menünavigation

- **Testfall:** Testen der Navigation im Menü mithilfe des Encoders.
- **Schritte:**
  1. Den Encoder wird in beide Richtungen gedreht und die Cursor-Bewegung beobachtet. Die cursor Variable `cursor_index` des Filemanagers würde im Debugger geprüft auf deren Zählverhalten.
  2. Der Push-Button wird gedrückt, um eine Auswahl zu treffen. Der Flag switch_push_button würde geprüft um die Validierung des korrekten Verhaltens beim drücken zu bestätigen.
- **Erwartete Werte:**
  - Der Cursor bewegt sich entsprechend der Drehrichtung des Encoders.

  - Die Auswahl wird korrekt angezeigt, wenn der Push-Button gedrückt wird.

- **Testergebnisse:**
  - Der Cursor bewegte sich korrekt und präzise entsprechend der Encoder-Drehung.
  - Die Auswahl wurde zuverlässig angezeigt, nachdem der Push-Button gedrückt wurde.

Beschreibung des Vorgehens: Der Encoder wird gedreht, um sicherzustellen, dass der Cursor im Menü korrekt bewegt wird. Die Cursor Variable `cursor_index` im FileManager Struct `fm` wird korrekt hoch und runter gezählt bei entsprechender Bewegung. Anschließend habe ich den Push-Button betätigt, um zu prüfen, ob die Auswahl entsprechend angezeigt wird. Der Flag `switch_push_button` würde ebenfalls korrekt gesetzt.

6.2 Testprotokoll: Potentiometer

6.2.1 Verbindung überprüfen

- **Testfall:** Überprüfung der Verbindung zwischen Potentiometer und Mikrocontroller.
- **Schritte:**
  1. Prüfung der Kabelverbindungen mit einem Multimeter.
  2. Ich habe sichergestellt, dass die Signale an den vorgesehenen Pins des Mikrocontrollers ankommen.
- **Erwartete Werte:**
  - Kontinuität und korrekte Verbindung an allen relevanten Pins.
  - Die gemessene Spannung betrug 3.29 V, was innerhalb des erwarteten Bereichs von 3.3 V $\pm$ 5
- **Beschreibung des Vorgehens:** Die Verbindungen zwischen dem Potentiometer und dem Mikrocontroller würde überprüft, indem die Kabel mit einem Multimeter auf Kontinuität getestet habe würden. Zudem wird verifiziert, dass die Signale korrekt an den vorgesehenen Pins des Mikrocontrollers ankommen.

6.2.2 ADC-Konfiguration überprüfen

- **Testfall:** Überprüfung der ADC-Konfiguration.
- **Schritte:**
  1. Die Auflösung, Sample-Rate und Referenzspannung des ADCs wurden überprüft.
  2. Es wurde sichergestellt, dass der ADC korrekt konfiguriert ist.
  3. Die Referenzspannung wurde mit einem Multimeter gemessen, um ihre Übereinstimmung mit den Werte im Debugger zu bestätigen.
  4. Die Sample-Rate wurde durch Analyse der ADC-Konfigurationsregister überprüft und mit den erwarteten Werten verglichen. `ADC1_CFGR` Configuration Register.
- **Erwartete Werte:**
  - Auflösung: 12-bit (0-4096 Werte).
  - Sample-Rate entspricht den Spezifikationen.
  - Referenzspannung entspricht der spezifizierten Spannung.
- **Testergebnisse:**
  - Die Auflösung des ADCs wurde korrekt auf 12-bit (0-4096 Werte) eingestellt.
  - Die Referenzspannung wurde mit einem Multimeter gemessen und entsprach der spezifizierten Spannung.
  - Die Sample-Rate wurde durch Überprüfung der ADC-Konfigurationsregister bestätigt und entsprach den angegebenen Spezifikationen.

  - Die ADC-Konfiguration war ordnungsgemäß und entsprechend den Anforderungen eingerichtet.

Beschreibung des Vorgehens: Die Konfiguration des ADCs wurde durch Überprüfung der Auflösung, Sample-Rate und Referenzspannung sichergestellt. Die Referenzspannung wurde direkt mit einem Multimeter gemessen, um ihre Übereinstimmung mit den Spezifikationen zu bestätigen. Die Sample-Rate wurde durch die Analyse der ADC-Konfigurationsregister verifiziert, indem die tatsächliche Rate mit den erwarteten Werten verglichen wurde. Die Validierung erfolgte durch den Einsatz eines Debugging-Tools, um zu gewährleisten, dass der ADC gemäß den Anforderungen konfiguriert ist.

6.2.3 DMA-Konfiguration überprüfen

- **Testfall:** Überprüfung der DMA-Konfiguration.
- **Schritte:**
  1. Es wurde sichergestellt, dass der DMA die ADC-Daten in den Puffer `currentValues` überträgt.
  2. Die Übertragungsart und die Übertragungsrate wurden überprüft.
  3. Das Datenformat wurde auf WORD (16-Bit) konfiguriert und geprüft.
- **Erwartete Werte:**
  - Korrekte Konfiguration des DMA, Übertragungsmodus auf Circular.
  - Datenformat korrekt auf WORD (16-Bit) eingestellt.
- **Testergebnisse:**
  - Der DMA überträgt die ADC-Daten wie erwartet in den Puffer `currentValues`.
  - Die Übertragungsart ist korrekt auf Circular eingestellt.
  - Das Datenformat wurde erfolgreich auf WORD (16-Bit) konfiguriert. Die Überprüfung wurde durch Einsicht in die DMA-Register bestätigt. Insbesondere wurden die Registerwerte für ‘PSIZE‘ und ‘MSIZE‘ auf 16-Bit überprüft, um die korrekte Einstellung des Datenformats zu bestätigen.

Beschreibung des Vorgehens: Die DMA-Konfiguration wurde überprüft, indem zuerst sichergestellt wurde, dass der DMA die ADC-Daten korrekt in den Puffer `currentValues` überträgt(Debugging). Danach wurde die Übertragungsart auf Circular gesetzt und die Übertragungsrate validiert. Schließlich wurde das Datenformat auf WORD (16-Bit) konfiguriert und durch Einsicht in die entsprechenden DMA-Register überprüft, insbesondere durch Überprüfung der Registerwerte für ‘PSIZE‘ und ‘MSIZE‘.

6.2.4 Daten analysieren

- **Testfall:** Analyse der aufgezeichneten ADC-Werte und Prüfung der Glättung.
- **Schritte:**
  1. Die ADC-Werte `fm.fader_Class[]`, `adcBuffer[]`, `smoothValue[]` und `currentClassPercentADC[]` wurden mit einem Debugger analysiert.
  2. Es wurde geprüft, ob eine lineare Zunahme der Werte entsprechend der Stellung des Schiebe-Potentiometers vorliegt.
  3. Der `smoothValue[]` wurde berechnet, und es wurde auf Schwankungen und plötzliche Änderungen geprüft. Glättung erfolgte mit `100, 1000, 10000, 30000, 100000` aufeinander addierten Werten.
- **Erwartete Werte:**
  - ADC-Werte entsprechen den Positionen des Potentiometers.
  - Die geglätteten Werte zeigen eine stabile Wertentwicklung.
  - Keine unerwarteten Sprünge oder signifikanten Schwankungen; Grundrauschen um wenige Volt ist normal.

- **Testergebnisse:**
  - Die ADC-Werte `fm.fader_Class[]`, `adcBuffer[]`, `smoothValue[]` und `currentClassPercentADC[]` wurden erfolgreich mit dem Debugger analysiert.
  - Die Werte zeigten eine erwartungsgemäße lineare Zunahme in Abhängigkeit von der Potentiometer-Position.
  - Der `smoothValue[]` zeigte eine stabile Wertentwicklung ohne signifikante Schwankungen oder plötzliche Änderungen.
  - Es traten keine unerwarteten Sprünge auf. Ein sehr geringes Grundrauschen wurde erstmal als normal eingestuft. 30000 aufeinader addierte Werte dessen Mittelwert berechnet würde stellten sich jedoch am als Effzietesten raus.

Beschreibung des Vorgehens: Die ADC-Werte wurden mit einem Debugger analysiert, um sicherzustellen, dass sie der Stellung des Potentiometers entsprechen. Es wurde überprüft, ob die Werte linear ansteigen und ob der geglättete Wert `smoothValue[]` stabil bleibt. Schwankungen und plötzliche Änderungen wurden geprüft, um die Effektivität der Glättung zu bewerten. Das Grundrauschen wurde als normal eingestuft.

Lösung um das Grundrauschen zu minimieren: Das Grundrauschen könnte zukünftig durch den Einsatz von Kondensatoren, z.B. 100 nF, minimiert oder beseitigt werden.

6.3 Testprotokoll: SD-Karten SPI-Schnittstelle

6.3.1 Testzielsetzung

Dieser Test überprüft die Funktionalität der Lese- und Schreibfunktionen für die SD-Karte über die SPI-Schnittstelle.

6.3.2 Testdurchführung

- **Mounten des Dateisystems:**
  - Die Funktion `f_mount` wurde aufgerufen, um das Dateisystem der SD-Karte zu mounten.
  - Der Rückgabewert (`fres`) wurde überprüft. Im Fehlerfall wurde eine Meldung ausgegeben und der Test abgebrochen.
- **SD-Karten-Statistiken:**
  - Die Funktion `f_getfree` wurde aufgerufen, um die freien Cluster, freien Sektoren und Gesamtsektoren der SD-Karte zu ermitteln.
  - Im Fehlerfall wurde eine Meldung ausgegeben und der Test abgebrochen.
  - Die Gesamt- und freien Speicherplatzwerte wurden berechnet und über `myprintf` ausgegeben.
- **Lesen einer Datei:**
  - Die Datei `test.txt` wurde mit der Funktion `f_open` im Lesemodus geöffnet.
  - Der Rückgabewert (`fres`) wurde überprüft. Im Fehlerfall wurde eine Meldung ausgegeben und der Test abgebrochen.
  - Es wurde versucht, 30 Bytes aus der Datei zu lesen (`f_gets`).
  - Gelingt das Lesen, wurde der Inhalt der gelesenen Daten über `myprintf` ausgegeben.
  - Im Fehlerfall wurde eine Meldung ausgegeben.
  - Die Datei wurde mit `f_close` geschlossen.
- **Schreiben einer Datei:**
  - Die Datei `write.txt` wurde mit der Funktion `f_open` im Schreibmodus und mit Flags zum Anlegen der Datei geöffnet.
  - Der Rückgabewert (`fres`) wurde überprüft. Im Fehlerfall wurde eine Meldung ausgegeben.
  - Ein String (`ä new file is made!"`) wurde in den Puffer `readBuf` kopiert.

  - Die Daten aus `readBuf` wurden mit `f_write` in die Datei geschrieben.
  - Die Anzahl der geschriebenen Bytes wurde über `myprintf` ausgegeben.
  - Im Fehlerfall wurde eine Meldung ausgegeben.
  - Die Datei wurde mit `f_close` geschlossen.

- **Dismounten des Dateisystems:**
  - Die Funktion `f_mount` wurde mit `NULL` aufgerufen, um das Dateisystem der SD-Karte zu dismounten.

6.3.3 Auswertung

Der Test verlief erfolgreich:

- Das Mounten und Dismounten des Dateisystems wurden erfolgreich durchgeführt.
- Die SD-Karten-Statistiken stimmten mit den Erwartungen überein.
- Das Lesen und Schreiben von .txt Dateien funktionierte einwandfrei.

Es wurde festgestellt, dass das Schreiben eines ‘struct‘ nicht möglich war. Eine zukünftige Lösung wird angestrebt.

7 Fazit

- Erkenntnisse und Gelerntes