**[*Die digitale Welt der heutigen Modelleisenbahn]***

**Anforderungsdokument**

[1. Ziel und Zweck des Projekts 3](#_Toc485744603)

[1.1. Projektbeschreibung 3](#_Toc485744604)

[1.2. Kurzbeschreibung des Projekts 3](#_Toc485744605)

[1.3. Zweck des Projekts 3](#_Toc485744606)

[1.4. Hintergrund, Problemstellung, Motivation für das Projekt 3](#_Toc485744607)

[1.5. Ziele des Projekts 3](#_Toc485744608)

[1.6. Erfolgskriterien 3](#_Toc485744609)

[2. Systemübersicht 4](#_Toc485744610)

[3. Architektur und Designentscheide 5](#_Toc485744611)

[3.1. Modell(e) und Sichten 5](#_Toc485744612)

[3.2. Daten (Mengengerüst & Strukturen) 5](#_Toc485744613)

[3.3. Entwurfsentscheide 5](#_Toc485744614)

[3.4. Randbedingungen und Einschränkungen 5](#_Toc485744615)

[4. Schnittstellen 6](#_Toc485744616)

[4.1. Externe Schnittstellen 6](#_Toc485744617)

[4.2. wichtige interne Schnittstellen 6](#_Toc485744618)

[4.3. Benutzerschnittstelle(n) 6](#_Toc485744619)

[5. Umgebungs-Anforderungen 7](#_Toc485744620)

[5.1. Technologie-Voraussetzungen 7](#_Toc485744621)

[5.2. Kooperierende Anwendungen und COTS-Komponenten 7](#_Toc485744622)

[6. Randbedingungen und Einschränkungen 7](#_Toc485744623)

[7. Entwurfsdetails 8](#_Toc485744624)

[8. Testplan 9](#_Toc485744625)

[8.1. Testziele 9](#_Toc485744626)

[8.2. Testdesign und -automatisierung 9](#_Toc485744627)

[8.3. Vorgehen beim Testen 9](#_Toc485744628)

[8.4. Testfälle, Testkriterien und Testergebnisse 9](#_Toc485744629)

[9. Projektabschluss 10](#_Toc485744630)

Versionen:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev. | Datum | Autor | Bemerkungen | Status |
| 1.0 | 22.06.2017 | Kaufmann, Reimers, Wollschläger | Abgabeversion | In Arbeit |
|  |  |  |  |  |

1. Ziel und Zweck des Projekts

## Projektbeschreibung

*<Dieser Teil ist ein fester Bestandteil eines Lasten- oder Pflichtenhefts. Möglich ist eine Übernahme der Projektbeschreibung aus den Vorträgen ergänzt um Informationen, die notwendig sind, die nachfolgenden Teile zu verstehen..>*

Unser Projekt war die Erweiterung der Software eines Fahrtreglers auf einen vierfach Fahrtregler.

## Kurzbeschreibung des Projekts

Für eine Modelleisenbahn Zentrale soll ein vierfach Fahrtregler entwickelt werden. Mit diesem Regler soll es möglich sein 4 Züge Gleichzeitig zu steuern. Über den Regler soll die Geschwindigkeit für einen Zug über einen Schieberegler gesteuert werden. Ein Taster soll die Fahrtrichtung ändern, wenn der Geschwindigkeitsregler auf 0 ist. Die DUO LED zeigt an ob der Zug vorwärts (grün) oder rückwärts (rot) fährt. Über Funktionstasten (1-4) sollen Zuggeräusche für einen der 4 Züge abgespielt werden.

## Zweck des Projekts

Der Zweck des Projekts ist die Optimierung eines bestehenden Systems durch einen vierfach Fahrtregler.

## Hintergrund, Problemstellung, Motivation für das Projekt

Zurzeit können mit der Zentrale nur 2 Züge gleichzeitig gesteuert werden. Wenn ein weiterer Zug gesteuert werden soll, muss die Zentrale umschalten. Durch das Umschalten verliert der Anwender die direkte Kontrolle über alle fahrenden Züge. Dabei soll ein vierfach Fahrtregler helfen, der 4 Züge gleichzeitig ansteuern kann und erweiterbar sein soll. Dadurch sieht der Anwender sofort ob der Zug vorwärts oder rückwärts fährt, kann die Geschwindigkeit ohne umzuschalten anpassen und direkt die Funktionstasten für einen Zug benutzen.

## Ziele des Projekts

Das Ziel des Projekts ist eine Implementierung des vierfach Fahrtregler mit den gewünschten Optionen und eine Anbindung an die Zentrale über das LocoNet.

## Erfolgskriterien

Ein funktionierender vierfach Fahrtregler mit den gewünschten Optionen.

1. Systemübersicht

*Übersicht und Begründung des gewählten Lösungsansatzes*



Eine grobe Übersicht über den Ablauf der Software soll das obere Bild darstellen. Der Benutzer interagiert mit dem Fahrtregler. Die Hauptroutine in der Software holt sich die Werte der Bauteile. Wenn alle Werte ausgelesen wurden, werden sie in Daten umgewandelt, mit der die Zentrale arbeiten kann. Danach werden sie in Pakete gesteckt und für die Übertragung vorbereitet. Wenn alles abgeschlossen ist, wird das Paket gesendet. Nachdem das Paket an der Zentrale angekommen ist, schickt sie eine Antwort an den Fahrtregler zurück. Damit ist ein Durchlauf beendet. In unserem Fall wird dieser Ablauf viermal ausgeführt (für jeden Regler). Danach beginnt der Durchlauf wieder mit dem ersten Regler.



Dieses Bild beschreibt die Funktion des Fahrtreglers.

Auf dem Fahrtregler befinden sich:

1. 4 Schieberegler (Zum Steuern der Geschwindigkeit)
2. 4 Richtungstasten (Zum Ändern der Zugrichtung)
3. 4 Shifttasten (Zum durchschalten der Modi 1 und 2)
4. 4 Funktionstasten (4 in Modus 1, 4 weitere in Modus 2)
5. 4 LED’s (Rot zum Signalisieren für Rückwärts, Grün für Vorwärts)
6. Architektur und Designentscheide

*<Beschreibung der Software-Architektur (Ist-Zustand)*

* *Zu allen Unterpunkten*
* *Motivation und Ziele*
* *Beschreibung*
* *Entscheidungen und Bewertung*
* *Ggf. Beispiel*
* *>*

## Modell(e) und Sichten

* *Architekturmodelle zur Beschreibung der Struktur der Software,*
* *Dekomposition in Subsysteme*
* *ggf dynamisches Modell,*
* *Zugangskontrolle und Security*

Um eine grobe Übersicht über den Zusammenhang der kompletten Software zu bekommen, wurde eine Dokumentation der Software erstellt.

Ein Modell der gesamten Software ist als HTML Seite verfügbar. Sie liegt mit im Abgabeordner unter: /Abgabeordner/Doxygen/html/index

Die Index Datei ist die Startseite der Dokumentation.

## Daten (Mengengerüst & Strukturen)

* *Datenmodell,*
* *Definition wichtiger Begriffe,*
* *ggf Data Dictionary*
* *Handhabung persistenter Daten*

<tbd>

## Entwurfsentscheide

Die vorhandene Software war darauf ausgelegt nur einen Zug gleichzeitig steuern zu können. In der Software gibt es einen rwslotdata (siehe Punkt 7) in denen die Informationen gespeichert werden. Für unsere Version für vier Regler mussten wir den rwslotdata auf vier erweitern, durch ein Array. Alle Funktionen die mit dem rwslotdata arbeiten, mussten angepasst werden, da die Structur viermal existiert.

## Randbedingungen und Einschränkungen

Softwareseitig hatten wir keine nennenswerten Einschränkungen. Alle Methoden, Variablen und Klassen konnten so erstellt werden, wie wir es geplant hatten.

Hardwareseitig: Der ATmega 16 Mikrocontroller hat eine begrenzte Speicherkapazität, die wir jedoch mit unserer Software nicht erreicht haben.

1. Schnittstellen

*bei allen Unterpunkten*

* *Motivation und Ziele*
* *Beschreibung*
* *Entscheidungen und Bewertung*
* *Ggf. Beispiel*

## Externe Schnittstellen

Damit die Züge später mit dem Fahrtenregler gesteuert werden können, müssen die Befehle des Fahrtenreglers an die Zentrale geschickt werden können.

Die LocoNet Schnittstelle ist die wichtigste Schnittstelle, ohne sie kann keine Kommunikation stattfinden und damit auch kein Zug gesteuert werden.

LocoNet Schnittstelle der Zentrale:

Die Zentrale wird mit einem RJ11 Kabel mit dem Fahrtenregler verbunden.

## wichtige interne Schnittstellen

Auf dem Fahrtenregler befindet sich ein Mikroprozessor der die Verarbeitung aller anfallenden Operationen übernimmt. Verarbeiten eines Tastendrucks, oder verschieben eines Reglers.

Ohne die Verarbeitung der Signale können die zugewiesenen Züge nicht ordentlich gesteuert werden.

Erste Schnittstelle: LocoNet Schnittstelle des Fahrtenreglers

Der Fahrtenregler wird mit einem RJ11 Kabel mit der Zentrale verbunden.

Zweite Schnittstelle: Programmierschnittstelle auf dem Fahrtenregler

Um den Mikrocontroller programmieren zu können, wird er mit einem Programmierboard verbunden. Über das Programmierboard wird dann die erstellte Software aufgespielt.

## Benutzerschnittstelle(n)

Der Benutzer interagiert direkt mit dem Fahrtenregler. Auf dem Fahrtenregler befinden sich Taster, Schieberegler und LED’s. Er kann Aktionen durch das Betätigen der Taster oder Regler auslösen.

Am Computer kann die Software nicht bedient werden. Nur durch das LocoNet Tool kann die Kommunikation zwischen Fahrtregler und Zentrale am Computer ausgelesen werden.

1. Umgebungs-Anforderungen

## Technologie-Voraussetzungen

*HW, BS, VM*

Hardwarevoraussetzungen: Die entwickelte Software wird auf einen Mikrocontroller geschrieben, der auf einer selbst angefertigten Platine sitzt.

Betriebssystemvoraussetzungen: Entwickelt wird die Software mit Atmel Studio 7, das nicht auf allen Betriebssystemen funktioniert. Unterstützt wird Atmel Studio 7 erst ab Windows 7.

Die Software für den Mikrocontroller ist nicht Betriebssystemabhängig, da auf dem Controller kein Betriebssystem läuft.

Virtual-Machine Voraussetzungen: In unserem Projekt werden Virtual Machines nicht verwendet.

<tbd>

## Kooperierende Anwendungen und COTS-Komponenten

*<Externe Anwendungen, mit denen die Software zusammenarbeitet oder Commercial of the Shelf (COTS) Komponenten, die verwendet werden>*

Die implementierte Software auf der Platine kommuniziert über LocoNet mit der Zentrale. Die Zentrale verwaltet und steuert die Züge die sich auf den Gleisen befinden. Ein Befehl wird von der Platine gesendet, die Zentrale empfängt und verarbeitet den Befehl und sendet eine Bestätigung an die Platine zurück. An die Zentrale kann ein PC angeschlossen werden, um die Zentrale und die LocoNet Kommunikation auszulesen. Das verwendete Tool (Panel Pro) listet die gesamte Kommunikation zwischen Zentrale und Platine auf. Damit lässt sich sehr schnell erkennen, ob Befehle die gesendet werden auch korrekt an der Zentrale ankommen.

COTS Komponenten:

Die Intellibox II (Zentrale) von Uhlenbrock ist eine von uns verwendete COTS Komponente.

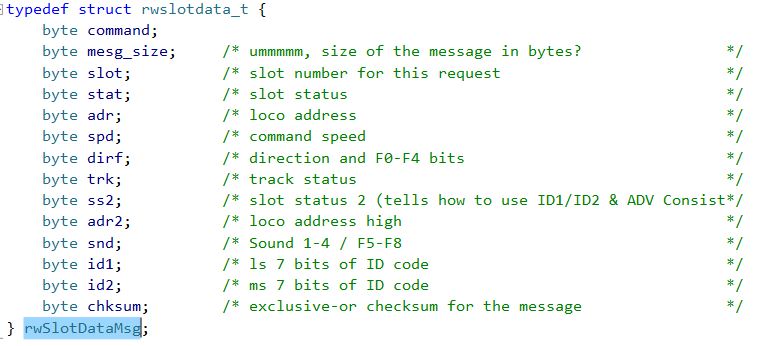
1. Randbedingungen und Einschränkungen

Einschränkungen werden nur von der Zentrale vorgegeben. Die Anzahl der Züge die gleichzeitig (mit genug Reglern) gesteuert werden können, liegt bei 128.

1. Entwurfsdetails

*Detaillierte Beschreibung wichtiger Einzelteile*

* *Pakete,*
* *Klassen,*
* *Objekte*
* *...*



Die wichtigste Struktur in unserem Programm ist die rwslotdata. In ihr werden sämtliche Informationen gespeichert, die an die Zentrale gesendet werden. Zugadresse, Geschwindigkeit, Richtung. Alles wird mit der Struktur bearbeitet. In unserem Fall existiert die Struktur viermal, für jeden Regler einen.

Danach kommen die Methoden für den kompletten Programmablauf.

1. vProcessRxLoconetMessage
   * Verarbeitet alle Informationen die von der Zentrale geschickt werden
2. vProcessADC
   * Wandelt den Wert der Schieberegler in akzeptable Werte für die Zentrale um. Eine Wandlung wird gestartet. Wenn sie abgeschlossen ist, wird sie umgewandelt und in einer Variable gespeichert. Danach wird der Input auf den nächsten Wandler gestellt und der Prozess beginnt von vorne. (Starten, Wert umwandeln, Wandler wechseln)

Der „dispatch“ Vorgang: Der sogenannte „dispatch“ ist von der Zentrale das Freigeben eines Zuges an alle Bedienelemente, die an das LocoNet angeschlossenen sind. Das gewünschte Bedienelement kann sich, durch Drücken der Funktionstaste, den Zug dann zuweisen lassen.

1. Testplan

Siehe PM-Vorlage

## Testziele

Welche Features werden getestet/nicht getestet

Getestet wird jede Funktion die programmiert wird. Ob verschieben des Reglers, das Drücken eines Tasters oder wechseln der LED Farbe. Es gibt keine Funktion die nicht getestet wird.

Ein Test gilt als erfolgreich, wenn die programmierte Funktion auf dem Mikrocontroller wie gewünscht ausgeführt wird.

## Testdesign und -automatisierung

Funktionen die direkt nach ihrer Implementierung getestet wurden, sind manuell getestet wurden.

Funktionen die erst mit der Kommunikation mit der Zentrale überprüft werden können, sind automatisiert.

## Vorgehen beim Testen

Getestet wird jede Funktion nach ihrer Implementierung. Die Funktion wird programmiert, auf den Mikrocontroller aufgespielt und getestet.

Der Ablauf zum Testen durchläuft immer die gleiche Routine.

1. Funktion schreiben
2. Funktion auf Mikrocontroller implementieren
3. Funktion direkt an der Platine testen

## Testfälle, Testkriterien und Testergebnisse

Hardwaretests sind, wie oben beschrieben, das Testen der implementierten Funktion direkt an der Platine.

Wird die Richtungstaste abgefragt, kann durch das Wechseln der LED Farbe angezeigt werden, ob die Abfrage richtig ausgeführt wurde.

Softwaretests sind Tests, bei denen man nicht direkt durch Drücken eines Tasters ein Event auslöst, sondern durch kommunizieren über das LocoNet mit der Zentrale.

Daten die im EEPROM gespeichert werden, müssen nach einem Neustart des Mikrocontrollers noch erhalten sein. Zum Beispiel die Zuweisung eines Zuges durch die Zentrale. Wird der Controller neu gestartet, muss der Zug immer noch steuerbar sein und nicht noch einmal neu zugewiesen werden. Wenn das der Fall ist, werden die Daten im EEPROM richtig abgespeichert. Damit wäre der Test (Speichern von Daten im EEPROM) erfolgreich.

1. Projektabschluss

*Ein letztes Update der Projektmanagementvorlage auf den Projekt-Schlussstand, dabei sind insb die folgenden Absätze und Kapitel zu aktualisieren*

**Absatz 2.2 Projektkontrolle und Projektsteuerung**

* *Getroffene Maßnahmen und ihre Auswirkungen im Projektverlauf.*

**PM - Anhänge**

<für den Projektabschluss sind abzugeben >

* Sprintpläne
* Sprintreview-Protokolle
* Meilensteinberichte
* **Aufgaben und Zeitaufwände aller Teammitglieder dargestellt über den Verlauf des Projekts**