

**Infrarot Übertragungssystem zur  
Steuerung von Modellfahrzeugen**

Bachelorarbeit von  
Philipp Gahtow  
1090126

Aufgabenstellung:  
Prof. K. Buchenrieder, Ph.D.

Betreuung:  
Dipl.-Medieninf. Andreas Attenberger

Abgabedatum: 23.12.2011

Universität der Bundeswehr München  
Fakultät für Informatik



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Technische Grundlagen</b>	<b>3</b>
2.1 Digital Command Control Standard (DCC) . . . . .	3
2.1.1 Telegrammaufbau . . . . .	4
2.1.2 Datenpaketformat . . . . .	5
2.1.3 Operation-Mode . . . . .	7
2.1.4 Service-Mode . . . . .	10
2.2 Arduino Entwicklungsboard . . . . .	11
<b>3 Konzept der Steuerung</b>	<b>13</b>
3.1 Funktionsweise des Faller Car-System . . . . .	13
3.2 Digitaler Funktionsaufbau . . . . .	15
<b>4 Funktionsweise der Infrarotdatenübertragung</b>	<b>19</b>
4.1 Infrarotdatenübertragung . . . . .	19
4.1.1 Frequenzen . . . . .	19
4.1.2 Datenraten . . . . .	20
4.2 Hardware . . . . .	20
4.2.1 Infrarot-Empfänger . . . . .	22
4.2.2 Infrarot-Sender . . . . .	23
4.3 Datenverarbeitung . . . . .	23
4.4 Störeinflüsse . . . . .	26
<b>5 Fahrzeugdekoder</b>	<b>27</b>
5.1 Hardware . . . . .	27
5.1.1 Schaltungsaufbau des Dekoders . . . . .	28
5.2 Software . . . . .	29
5.2.1 Energieeffizienz . . . . .	30
5.2.2 Dekodierung von IR-Signalen . . . . .	31
5.2.3 Befehlsauswertung . . . . .	31
5.2.4 Ansteuerung der Hardware . . . . .	32
5.2.5 Programmierung . . . . .	33
<b>6 Zusammenfassung</b>	<b>35</b>

<b>Abbildungsverzeichnis</b>	<b>37</b>
<b>Literaturverzeichnis</b>	<b>39</b>
<b>A Schaltungen</b>	<b>41</b>
<b>B Konfigurationsvariablen des Fahrzeugdekoder</b>	<b>43</b>

# Kapitel 1

## Einleitung

In der vorliegenden Arbeit wird eine digitale Steuerung für Miniaturmodellfahrzeuge und ein prototypischer Aufbau beschrieben. Auf modernen Modelleisenbahnanlagen werden die Züge, Signale und Weichen durch digitale Signale über das Bahngleis gesteuert. Dabei hat die digitale Technik gegenüber der analogen viele Vorteile. Digitale Modelleisenbahnanlagen benötigen eine zentrale Steuereinheit die auch Digitalzentrale genannt wird. Sie übernimmt als Master die gesamte Steuerung. Diese Zentrale erzeugt das digitale Gleissignal, welches je nach Hersteller ein anderes Datenübertragungs-Protokoll besitzt. In der vorliegenden Arbeit wird das von der NMRA<sup>1</sup> erstellte und in der aufgebauten Modellfahrzeugsteuerung eingesetzt DCC<sup>2</sup>-Protokoll ausführlich beschrieben.

Auf Modelleisenbahnanlagen finden sich neben Zugstrecken auch Straßen. Auf den Straßen sind in vielen Anlagen meist dem Maßstab entsprechend Modellfahrzeuge als Standmodelle zu finden. Da diese Fahrzeuge immer an der gleichen Stelle auf der Straße stehen, entstand die Idee diese Fahrzeuge mit der Eisenbahnsteuerung auf den Straßen fahren zu lassen. Fahrende Modellautos wurden erstmals von der Firma Faller<sup>3</sup> produziert. Anfangs wurden die Fahrzeuge analog gesteuert. Auf der Spielwarenmesse 2011 in Nürnberg wurde erstmals von der Firma Faller ein System zur digitalen Steuerung für Car-System Fahrzeuge vorgestellt [6].

Zielstellung der vorliegenden Arbeit war es, mit einer Digitalzentrale für die Fahrzeugsteuerung und einer Modelleisenbahn-Software, wie zum Beispiel Rocrail™, Railware™ oder Railroad & Co.™, Modellfahrzeuge zu steuern. Die dazu benötigte Hardware für die digitale Steuerung und Datenübertragung wird im weiteren Verlauf dieser Arbeit beschrieben. Zur Datenübertragung wird das oben genannte, weit verbreitete DCC-Protokoll eingesetzt.

Um jederzeit mit allen Modellfahrzeugen kommunizieren zu können ist ein geeignetes System für die Datenübertragung erforderlich. Dazu muss die Datenübertragung die Entfernung auf einer Modelleisenbahnanlage abdecken können und zusätzlich eine ausreichende Übertragungsgeschwindigkeit für die hohen Datenraten des DCC-Standards aufweisen. Außerdem wird eine hohe Störsicherheit der Übertragungsstrecke

---

<sup>1</sup>National Model Railroad Association [16]

<sup>2</sup>Digital Command Control [13]

<sup>3</sup>eingetragene Marke der Firma Gebr. FALLER GmbH, Kreuzstraße 9, 78148 Gütenbach, DE

gefordert. Zur Datenübertragung wird in dieser Arbeit ein Infrarotsystem eingesetzt. Alle Modellfahrzeuge müssen dabei mit einem geeigneten Infrarot (IR)-Empfänger ausgestattet werden. Dieser empfängt die digitalen DCC-Steuersignale der Digitalzentrale und gibt sie an den Fahrzeugdekoder im Fahrzeug weiter. Dieser dekodiert die gesendeten Daten und steuert Funktionen wie Blinken, Scheinwerfer, Motor und das Bremslicht.

In dieser Arbeit werden zuerst die Grundlagen einer digitalen Steuerung mit Datenübertragung mittels DCC-Protokoll erklärt. Im Weiteren wird die Hardwaregrundlage der Dekoder, die in den Modellfahrzeugen verwendet wurden, beschrieben. Um die Modellfahrzeuge in der vorgegebenen Spur zu halten, wird ein spezielles System der Firma Faller verwendet. Dazu werden die Grundlagen dieses Systems erläutert. Anschließend wird der Datenempfang und die Datenverarbeitung im Fahrzeug beschrieben. Es wird aufgezeigt, warum hier spezielle Hardware erforderlich ist. Dann wird der Dekoder mit der verwendeten Software zur Steuerung der Modellfahrzeuge beschrieben.

# Kapitel 2

## Technische Grundlagen

In diesem Kapitel werden die Grundlagen für eine Steuerung einer Modelleisenbahnanlage beschrieben. Zusätzlich wird die verwendete Hardware für den Fahrzeugdekkoder erklärt.

Eine herkömmliche analoge Modelleisenbahnsteuerung wird mit einem Eisenbahntransformator bedient. Mit diesem kann man die Gleisspannung über einen (Hand-)Regler einstellen. Dabei kann immer nur eine Lokomotive (Zug) gesteuert werden. Bei einer analogen Gleichstrombahn liegen hier je nach Größe des Modells 9 Volt bis 15 Volt auf dem Gleis. Durch eine Umpolung der Gleisspannung lässt sich die Fahrtrichtung ändern.

Diese Art der analogen Modellbahnsteuerung wurde im Laufe der Zeit in ihrem Funktionsumfang ständig erweitert. Dann aber konnten gewünschte Zusatzfunktionen mit ihr nicht mehr realisiert werden. So wurde für die Modelleisenbahn in den Achtzigerjahren die digitale Steuerung DCC entwickelt [13]. Das Ziel war es mit nur wenig Verdrahtungsaufwand auszukommen und eine unabhängige Steuerung mehrerer Lokomotiven (Züge) auf demselben Gleis zu ermöglichen. Außerdem sollte eine dauerhafte Beleuchtung der Wagen und Lokomotiven wie beim Original möglich sein.

### 2.1 Digital Command Control Standard (DCC)

Das Digital Command Control (DCC-) Protokoll wurde von der NMRA (National Model Railroad Association) als standardisiertes Datenübertragungsprotokoll speziell für digitale Modelleisenbahnen entwickelt. Es wird unter anderem für die Steuerung von Lokomotiven, Signalen, Weichen und Beleuchtung verwendet [16]. Andere bekannte Protokolle für die Datenübertragung sind das Märklin Motorola Protokoll, das Selectrix-Protokoll, das FMZ-Protokoll von Fleischmann und das mfx/Systems von Märklin.

Die Digitalisierung ermöglicht es Steuerungssignale über eine 2-Leiter-Schienentechnik<sup>1</sup>, ähnlich wie bei einem Bussystem zu übertragen. Über zwei Bahnschienen des Gleises wird die Versorgungsspannung für Lokomotiven und Wagenbeleuchtung eingespeist. In die Versorgungsspannung werden zugleich digitalen Daten eingekoppelt, indem sie auf die Versorgungsspannung aufmoduliert werden. Im Lokdekkoder werden

---

<sup>1</sup>Märklin verwendet zum Beispiel einen Mittelschleifer und hat somit eine dritte Leitung (Gleis) zur Datenübertragung oder Steuerung frei.

Versorgungsspannung und die Daten wieder getrennt. Der Dekoder generiert nach der Demodulation eine konstante Versorgungsspannung für den Mikrocontroller und alle anderen Komponenten. Alle am Dekoder angeschlossenen Komponenten werden über die ankommenden Datenbefehle angesteuert.

Die Datenübertragung für das Digitale Steuersignal DCC ist im NMRA-Standard S9 vom Juli 2004 festgelegt [16]. In der aktuellen Norm (Ausgabe 2007) der Normen Europäischer Modellbahnen wird in den NEM-Normen 670 und 671 der Standard detailliert beschrieben [10]. Die Beschreibung des Telegramms und der Befehle beziehen sich auf diese festgelegten Standards.

Die Datenübertragung erfolgt gemäß der 2-Leiter-Technik seriell. Dabei wird immer eine bestimmte Anzahl von Bits übertragen, die einen Befehl darstellen. Diese Bits werden zusammen mit der Gleisspannung übertragen. Zur Abbildung der Bits besteht die Gleisspannung aus zwei Spannungsniveaus die einen positiven und negativen Pegel besitzen. Bei einem Wechsel des Spannungsniveaus findet ein Nulldurchgang statt (siehe Abbildung 2.1). Über diese Nulldurchgänge ist es möglich, die zeitliche Länge eines Impulses zu bestimmen. So trennen zwei hintereinander folgende Nulldurchgänge mit gleicher Richtung die jeweiligen Bits voneinander. Jeder Nulldurchgang teilt ein Bit in einen ersten und einen zweiten Teil auf [11]. Ob ein Eins-Bit oder ein Null-Bit gesendet wurde, wird nur durch den zeitlichen Abstand der Nulldurchgänge bestimmt.

### 2.1.1 Telegrammaufbau

Zur Datenübertragung eines Bit, wechselt das Signal von einem negativen zu einem positiven Impuls. Jedes Bit besteht dabei aus zwei Teilen, einem negativen Impuls und einem positiven Impuls. Diese werden jeweils einen Nulldurchgang getrennt. Dabei hat jeder Impuls stets die gleiche Zeitdauer. Eine Ausnahme gibt es dabei nur bei einem Null-Bit. In der Abbildung 2.1 ist ein Teilausschnitt des DCC-Signals dargestellt. Ein Null- und ein Eins-Bit ist besonders gekennzeichnet und vergrößert.

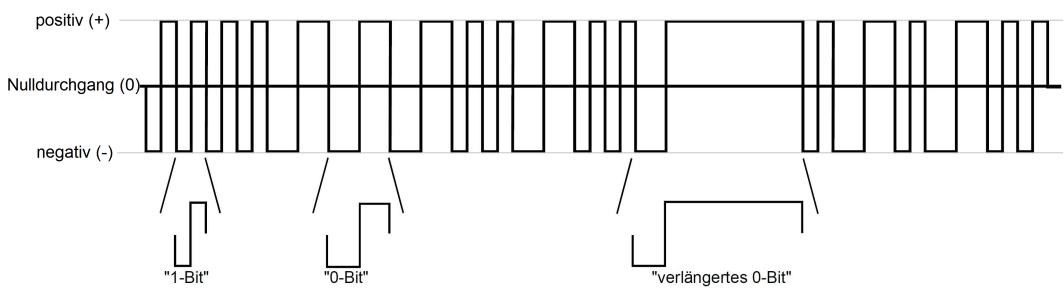


Abbildung 2.1: Aufbau eines DCC-Signals [11].

#### Erkennung eines Eins-Bit

Bei einem Eins-Bit beträgt die Dauer des positiven Impulses, wie auch des negativen Impulses jeweils 58 Mikrosekunden [11]. Somit ergibt sich für ein komplettes Eins-Bit eine Dauer von 116 Mikrosekunden. Durch Verbraucher oder äußere Störeinflüsse

können bei der Übertragung Impulslängenschwankungen auf dem Gleis auftreten. Dabei beträgt die zugelassene Toleranz eines positiven beziehungsweise negativen Impulses beim Senden plus minus drei Mikrosekunden. Die Dauer eines von der Digitalzentrale gesendeten Eins-Bit darf somit nicht unter 55 und über 61 Mikrosekunden liegen [11].

Im Gegensatz dazu sind die Toleranzen bei der Dekodierung eines gültigen Eins-Bit im Empfänger bei bis zu sechs Mikrosekunden pro Impuls. Die Abweichungen für die beiden Signalabschnitte vor und hinter dem Nulldurchgang eines Eins-Bit müssen dabei gleich gewichtet sein.

### Erkennung eines Null-Bit

Für ein korrektes Null-Bit gilt eine Dauer für beide Impulse von größer oder gleich 100 Mikrosekunden. Dabei bleiben die beiden Impulse normalerweise gleich zueinander. Die Länge der beiden Signalabschnitte lässt sich für bestimmte Zwecke, wie zum Beispiel für eine Datenpause bis zur Toleranzgrenze von maximal 9900 Mikrosekunden erweitern. Abweichungen sind hierbei bis zu einer minimalen Impulsdauer von 95 Mikrosekunden möglich. Die Gestamtdauer der jeweiligen Impulse darf dabei 12000 Mikrosekunden nicht überschreiten.

Für die Dekodierung gilt die Erkennung eines Impuls mit der Dauer von 90 bis 10000 Mikrosekunden als gültiges Null-Bit [11]. Meist wird für ein Null-Bit jedoch eine Zeit von 116 Mikrosekunden verwendet. Das ist die doppelte Länge eines Eins-Bit mit 58 Mikrosekunden. Bei der Verwendung eines speziellen verlängerten Null-Bit, sind Pausen im Sendevorgang oder die Realisierung einer Rückmeldung vom Dekoder zur Zentrale möglich.

### 2.1.2 Datenpaketformat

Laut Standard muss ein Dekoder wenigstens 95% der an ihn adressierten Datenframes als gültig erkennen [11]. Um die Gleisspannung, die auch als Energieversorgung der Dekoder dient, aufrecht zu erhalten müssen kontinuierlich Daten übertragen werden. Mittels Brückengleichrichter wird im Dekoder aus dem Datenstrom eine Gleichspannung erzeugt. Um immer ein Datensignal auf dem Gleis zu haben, werden zum Beispiel bestimmte Datenframes wiederholt oder Leerlaufframes (siehe Abschnitt 2.1.3) gesendet. Mit sich wiederholenden Befehlen wird sichergestellt, dass bei einer fehlerhaften Datenübertragung die Daten noch einmal gesendet werden. Für bestimmte Datenframes gelten standardisierte Wiederholungszeiten nach NEM671 [12].

Im Weiteren wird der Aufbau eines DCC-Basis-Datenpakets beschrieben.

### Datenpakete des DCC-Standard

Das Datenpaket besteht aus sechs Teilen. Diese sind: Synchronisation, Start-Bit, Adress-Byte, Daten-Byte, Prüf-Byte und Stop-Bit. In der Abbildung 2.2 ist ein DCC-Basis-Datenpaket mit drei Daten-Bytes (Adress-Byte, Befehls-Byte und Prüf-Byte) dargestellt.

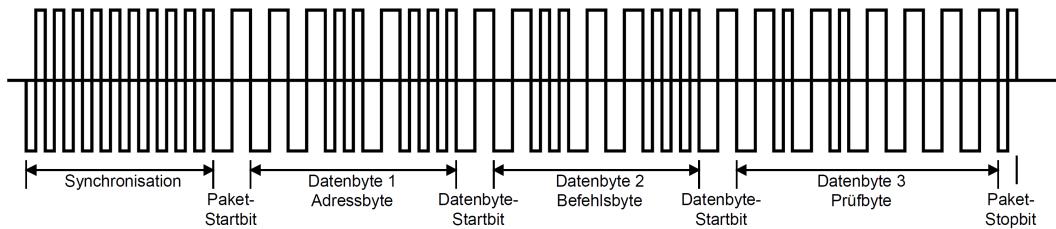


Abbildung 2.2: Aufbau eines DCC-Basis-Datenpakets mit drei Daten-Bytes [11]. Übertragen wurde hier ein Befehl für die Adresse 55, mit der Fahrstufe elf als Vorwärtsfahrt.

Um dem Dekoder zu signalisieren, dass Daten gesendet werden, wird eine Folge von mindestens zehn Eins-Bit als Synchronisation gesendet [9]. Danach folgt das Start-Bit. Es kennzeichnet als Null-Bit das Ende der Synchronisationsphase. Das Adress-Byte wird als dritter Teil des DCC-Basis-Datenpaketes gesendet und besteht normalerweise aus 8 Bit. Dann schließt sich das Daten-Byte-Startbit an, gefolgt von dem Daten-Byte mit 8 Bit Daten. Dieser Teil kann mehrmals, wenn mehr Daten vorhanden sind, gesendet werden. Vor dem Stop-Bit, welches das Ende des Datenpaketes mit einem Eins-Bit kennzeichnet, enthält das letzte Byte eine CRC (Cyclic Redundancy Check)-Prüfsumme, um Fehler bei der Übertragung zu erkennen. Sie wird durch ein Bitweises XOR<sup>2</sup> aus Adress-Byte und den vorhandenen Daten-Bytes gebildet.

### Dekoderadresse

Die Adresse des Dekoders, zu dem die jeweiligen Daten-Byte gehören, wird immer nach der Synchronisationsphase gesendet. Dafür sind die Basiswerte 0000 0000 (=0), 1111 1110 (=254) und 1111 1111 (=255) reserviert und dürfen nicht für die Dekodersteuerung verwendet werden. Weiterhin wird unterschieden zwischen der Basisadresse mit 8 Bit und der erweiterten Adresse mit 14 Bit [12]. Die erweiterten Adressen wurden eingeführt, da eine Beschränkung auf nur 255 Adressen zu gering war. Oft wird dies auch als lange Adresse bezeichnet. Das Adress-Byte beginnt mit einer führenden Null. Die folgenden 7 Bit enthalten die binär kodierte Adresse des Empfängerdekoders. Somit sind nur 127 Adressen verfügbar. Da ein Dekoder möglichst den ganzen Adressbereich unterstützen sollte, wird auch das erste Adress-Bit mit ausgewertet. Dies kann bei der Verwendung von Basis- und erweiterten Adressen zu Überschneidungen bei der Adresserkennung führen. Um lange Adressen nutzen zu können, wurde ein zusätzliches Adress-Byte eingeführt [4]. Dadurch ist es möglich 14 Bit Adressen anzusprechen. Zur Erkennung einer zwei Byte Adresse enthält das erste Byte eine Kodierung. Das erste Adress-Byte beginnt mit zwei Eins-Bit. Die Adresse wird dann aus den folgenden 6 Bit und dem zweiten Adress-Byte mit zusätzlich 8 Bit zusammengesetzt. Für das erste Adress-Byte gilt dann eine Einschränkung. Es darf nicht 1111 1111 (=255) sein. Somit sind nur 16218 Adressen möglich. Die meisten Digitalzentralen beschränken jedoch oft die maximal möglichen Adressen auf 9999 aufgrund einer beschränkten digitalen Anzei-

<sup>2</sup>Ein exklusives ODER, ist im Ergebnisbit Eins, falls die beiden zu vergleichenden Bit unterschiedlich sind, und Null, falls sie gleich sind.

ge. Bei Adressen über 127 wird meist von der Digitalzentrale aus automatisch eine lange Adresse gesendet [4]. In Dekoder ist meist wegen des Problem der Überschneidung ab Adresse 127 von Basis- und erweiterter Adresse die Eingabe einer Basisadresse auf maximal 127 beschränkt. Bei Inkonsistenzen, also einem häufigen Wechsel ab Adresse 127 zwischen Basis- und erweiterter Adresse, erkennt der Dekoder den an ihn adressierten Befehl nicht.

### Befehlsdaten

Im Daten-Byte können verschiedene Informationen übertragen werden. Hierbei handelt es sich um Fahrbefehle und Funktionsbefehle für Lokdekoder (engl. Multi Function Decoder) oder Schaltbefehle für Signal oder Weichendekoder (engl. Accessory Decoder). Durch unterschiedliche Kodierung der Befehls-Byte kann der Dekoder erkennen, welche Art von Befehl gesendet wurde. Für Multi Function Decoder gibt es wieder zwei Arten von Befehlen. Es handelt sich hierbei um die Fahr- oder Funktionsbefehle. Der genaue Befehlsaufbau wird in Abschnitt 2.1.3 dargestellt.

### Fehlererkennung per CRC

Die Prüfsumme wird durch ein bitweises XOR bestimmt. Sie besteht unabhängig davon, wie viele Adress- oder Daten-Byte gesendet wurden immer aus nur einem Byte. Für die Berechnung werden, wenn vorhanden, beide Adress-Bytes und das erste Befehls-Byte verwendet. Das Expansion-Byte, welches einen Befehl erweitert, wird somit nicht in der Prüfsumme berücksichtigt.

### 2.1.3 Operation-Mode

Im Weiteren wird der Aufbau eines Befehls-Byte beschrieben. Hierbei handelt es sich um Befehle im Operation-Mode, in welchem eine Modelleisenbahnanlage gesteuert wird. Der Aufbau ist je nach Inhalt der Daten-Bytes unterschiedlich gestaltet. Wenn sich die Digitalzenrale nicht im Operation-Mode befindet ist der Service-Mode aktiv. Dieser dient zur Dekoderprogrammierung. Dieser wird im Abschnitt 2.1.4 beschrieben.

### Fahrstufen und Fahrsteuerung für Lokomotiven

Für die Fahrsteuerung gibt es verschiedene Fahrbefehle. Unterschieden wird dabei anhand der Genauigkeit der Motoransteuerung. Das DCC-Protokoll sieht dafür Fahrstufen vor, mit denen die Motorgeschwindigkeit (Drehzahl) stufenweise gesteuert wird. Je nachdem mit welcher Anzahl von Fahrstufen der Dekoder betrieben wird, enthält der Befehl maximal 14, 28 oder 128 Fahrstufen je Fahrtrichtung. Ältere Dekoder oder Digitalzentralen können meist nur 14 oder 28 Fahrstufen realisieren. Das symbolisch dargestellte Daten-Byte für maximal 28 Fahrstufen ist wie folgt aufgebaut: 01DCSSSS. Es beginnt mit einem Null- und Eins-Bit gefolgt von dem Fahrtrichtungsbit (D). Ein Eins-Bit als Fahrtrichtungsbit steht für die Vorwärtsfahrt. Die restlichen fünf Bit (CSSSS) geben

Auskunft über die Fahrstufe [16]. Dabei ist das (C)-Bit ein zusätzliches Geschwindigkeitsbit, um die 14 Fahrstufen auf 28 zu erweitern. Die genaue Aufteilung der einzelnen Fahrstufen zu den einzelnen Binärwerten ist in der Norm festgelegt und dort in einer Tabelle dargestellt [15].

Für 128 Fahrstufen wird ein zweites Daten-Byte benötigt. Dabei enthält das erste Daten-Byte zwei Null-Bit gefolgt von sechs Eins-Bit. So kann der Dekoder erkennen, dass es sich hier um einen Fahrbefehl mit 128 Fahrstufen handelt. Das zusätzliche zweite Daten-Byte ist wie folgt aufgebaut: DSSSSSSS. Auch hier wird die Fahrtrichtung mit dem (D)-Bit übertragen. Die folgenden 7 Bit (SSSSSSS) stehen für die aktuelle Fahrstufe. Eine oder zwei Fahrstufen sind egal mit wie vielen Fahrstufen der Dekoder arbeitet für einen Notstop des Motors reserviert. Dieser ermöglicht eine sichere Abschaltung des Motors, zum Beispiel vor dem Programmieren des Dekoders. Diese ist notwendig, da im Service-Mode (siehe Abschnitt 2.1.4) keine Operation-Mode Befehle verarbeitet werden können und die Lokomotive sonst steuerungslos fahren würde.

### Funktionen für Lokomotiven

Je nach verwendetem Dekoder stehen an diesem eine bestimmte Anzahl von zusätzlichen frei verwendbaren Ausgängen zur Verfügung. Diese können im Service-Mode so programmiert werden, dass sie auf bestimmte Funktionen reagieren. Jede Digitalzentrale bietet zu jeder Lokomotivadresse zusätzlich zu den Fahrbefehlen Funktionstasten zur Steuerung von Zusatzfunktionen an. Mit diesen Funktionsbefehlen können Licht, akustische Signale oder andere Zusatzfunktionen mit dem jeweils adressierten Dekoder geschaltet werden.

Mit nur einem Daten-Byte können dabei bis zu 13 Funktionen adressiert werden. Für die Funktionen F13 bis F28 wird ein zweites Daten-Byte, das Expansion-Byte gesendet. Es beinhaltet die folgenden Funktionen [15]:

- **Grundfunktionen F0 bis F4:** Bei den Grundfunktionen F0 bis F4 besteht das Daten-Byte aus den folgenden Bit-Kombinationen 100D DDDD. Dabei steht jedes (D)-Bit für eine Funktion. Dabei stehen die hinteren vier (D)-Bit für F4 bis F1 und das vorderste erste (D)-Bit für die Lichtfunktion F0.
- **Erweiterte Funktionen F5 bis F12:** Für erweiterte Grundfunktionen beginnt das Daten-Byte mit einer anderen Bit-Kombination 101D DDDD gefolgt von den Steuerungsdaten (D). Hierbei werden die Funktionen F5 bis F12 angesteuert. Dazu gibt das erste (D)-Bit an, um welche Gruppe von Funktionen F5 bis F8 oder F9 bis F12 es sich handelt. Die restlichen (D)-Bit steuern dann die jeweilige Funktionen an.
- **Weitere Funktionen F13 bis F28:** Für die Funktionen F13 bis F28 wird ein zusätzliches Byte verwendet. Hierbei hat das erste Daten-Byte die folgende Zusammensetzung 1101 111C. Diese signalisiert, dass noch ein zweites Daten-Byte folgt. Im sogenannten Expansion-Byte stehen 8 Bit kodiert zur Steuerung der Funktionen F13 bis F20, wenn das (C)-Bit im ersten Daten-Byte ein Null-Bit ist

und die Funktionen F21 bis F28, wenn das (C)-Bit ein Eins-Bit ist. Jedes Bit im Expansions-Byte steht somit für eine eigene Funktion.

### Schaltbefehle

Diese speziellen Befehle werden genutzt um Signale, Weichen, Beleuchtung oder andere Effekte auf der Modelleisenbahnanlage zu steuern. Bei dem Entwurf des DCC-Protokolls wurden auch Befehle für die Weichensteuerung erstellt. Dabei haben die Weichen für je eine der beiden Lagen eine Magnetspule. Ein Befehl ist so ausgelegt, dass pro Adresse bis zu vier Weichen gerade oder abbiegend geschaltet werden können. In der weiteren Beschreibung stehen für die zwei Stellungen die Farben Rot (gerade) und Grün (abbiegen). Diese werden auch für die Tasten auf vielen Digitalzentralen verwendet werden.

Schaltbefehle werden normalerweise an den jeweiligen Dekoder adressiert. Dieser besitzt vier Ports die sich jeweils auf Rot oder Grün schalten lassen. Rot steht dabei für deaktiviert und Grün für aktiviert. Viele aktuelle Dekoder können pro Ausgang verschiedene Adressen und Ports verwalten. Sie sind somit vielseitig einsetzbar.

Das Adress-Byte gibt die Adresse des Befehls an und im Befehls-Byte wird der Port und der Schaltzustand kodiert. So lassen sich je Adresse bis zu vier verschiedene Ports steuern [15]. Das Adress-Byte beginnt bei Schaltbefehlen mit einem Eins-Bit gefolgt von einem Null-Bit. Die folgenden 6 Bit stehen für den ersten Teil der Adresse. Der zweite Adressteil wird mit dem ersten Datenbyte gesendet und ist wie folgt aufgebaut: 1AAA CBBR. Dadurch ergeben sich aus den 6 Bit des Adress-Byte 64 mögliche Adressen und mit den drei (A)-Bit aus dem ersten Daten-Byte, welche invertiert betrachtet werden müssen, sind somit maximal  $64 \cdot 8 = 512$  Adressen ansprechbar [4]. Das (C)-Bit im ersten Datenbyte gibt an ob der Port aktiv bei logisch Eins oder inaktiv bei logisch Null ist. Die drei letzten Bit (BBR) stehen für die einzelnen vier Ports jeder Adresse. Das letzte Bit (R) wird nur dazu verwendet, den Port auf Grün oder Rot zu schalten. Bei einer Weiche wird dadurch zum Beispiel unterschieden welche Spule für abbiegen oder gerade angesteuert wird. Das (C)-Bit dient als Steuerimpuls. Dieser ist notwendig um beispielsweise zu verhindern, dass die Spule des Weichenantriebs durchbrennt. Mit diesem System ist es möglich mit jeder Adresse vier Ports anzusteuern. Insgesamt lassen sich so bis zu  $512 \cdot 4 = 2048$  Weichen auswählen [4].

### Broadcast Pakete

Diese Pakete werden von allen Dekodern empfangen und ausgewertet [15].

- **Dekoder-Reset-Paket:** Das Dekoder-Reset-Paket enthält drei Byte. Jedes dieser drei Byte besteht aus 8 Null-Bit. Erhält ein Dekoder ein solches Paket erhält, werden alle Ausgänge auf den Ausgangszustand zurückgesetzt und der Motor abgeschaltet. Ein solches Paket wird von den Digitalzentralen vor Beginn des Service-Mode (siehe Abschnitt 2.1.4) gesendet.
- **Dekoder-Idle-Paket:** Das DCC-Protokoll sieht vor, dass ständig Daten gesendet werden. Dies ist notwendig, um die Gleisspannung, die auch als Versorgungsspan-

nung für die Dekoder dient, aufrecht zu erhalten. Ein Dekoder-Idle-Paket besteht aus drei Byte. Das erste Byte besteht aus 8 Eins-Bit, das zweite aus 8 Null-Bit und das letzte wieder aus 8 Eins-Bit. Dieses Paket wird immer dann gesendet, wenn keine anderen Daten vorhanden sind, zum Beispiel beim Einschalten der Digitalzentrale. Sie werden im Dekoder nicht weiter verarbeitet.

- **Broadcast-Stop-Paket:** Das Broadcast-Stop-Paket erfüllt eine ähnliche Funktion wie das Dekoder-Reset-Paket, allerdings bezieht sich dieses Paket ausschließlich auf die Motoransteuerung. Es werden drei Byte gesendet. Das erste Byte enthält 8 Null-Bit, das zweite Byte besteht aus (01DC000S) Bit und das letzte Byte enthält die Prüfsumme. Ist das (S)-Bit ein Null-Bit und damit nicht gesetzt, dann sollen alle Dekoder den Motor zum Halten bringen. Wenn das (S)-Bit gesetzt ist, dann sollen alle Dekoder die Motoren der fahrenden Lokomotiven ausschalten. Das (DC)-Bit dient der Fahrtrichtung. Wenn das (C)-Bit gesetzt ist, wird die Fahrtrichtungsinformation ignoriert. Wenn es nicht gesetzt ist, gibt das (D)-Bit die Fahrtrichtung an.

#### 2.1.4 Service-Mode

Im Service-Mode ist es möglich, Einstellungen der Dekoder zu ändern. Dies geschieht von der Digitalzentrale aus auf einem separaten Gleis (Programmiergleis). Dabei darf die Lokomotive auf dem Gleis nicht fahren. Alle Funktionen müssen abgeschaltet sein. Um dies sicherzustellen, wird vor dem Programmervorgang ein Broadcast-Stop-Paket und ein Dekoder-Reset-Paket gesendet. In der NMRA Norm RP 9.2.3 ist der Aufbau der zu programmierenden Register und deren Funktionsweise vorgeschrieben. Im Dekoder und bei der Programmierung werden diese Register als CV (Configuration-Variable) bezeichnet [14].

Service-Mode-Pakete werden von einem Dekoder nur im Service-Mode erkannt. Ein Dekoder wechselt in den Service-Mode, wenn er ein gültiges Service-Mode Paket gefolgt von einem Dekoder-Reset-Paket empfängt. Der Dekoder verlässt den Service-Mode, wenn er einen Befehl erkennt, der kein Service-Mode-Befehl ist oder nach dem letzten Reset 20 Millisekunden vergangen sind. Erst nachdem der Dekoder ein gültiges Operation-Mode-Paket erhält, wechselt dieser wieder in den Operation-Mode. Dies ist notwendig damit der Dekoder keine Service-Mode-Befehle als Operation-Mode-Pakete erkennt. Service-Mode-Pakete besitzen eine kurze Adresse von 112 bis 127. Während dem schreiben einer CV wird beispielsweise die Adresse 124 gesendet. In dem ersten Daten-Byte ist dann binär das CV Register kodiert und das zweite Daten-Byte enthält den neuen Registerwert.

Eine Erweiterung des Service-Mode arbeitet mit einer adressbezogenen Programmierung. Über “Programming On Main“ (POM) können Register im Dekoder während des Betriebes geändert werden. Alle Programmierbefehle werden an den jeweiligen Dekoder adressiert. Dieser erkennt die Befehle und ändert seine Registerwerte. Die eigene Dekoderadresse kann bei dieser Programmierung nicht geändert werden, da sie für die Programmierung benötigt wird. Eine Reaktion auf die Veränderung ist sofort sichtbar. Allerdings unterstützen nicht alle Digitalzentralen diese Art der Programmierung.

## 2.2 Arduino Entwicklungsboard

Arduino ist der Handelsname für ein Open Source Projekt<sup>3</sup>. Mit dieser frei verfügbaren Software und den preisgünstigen Entwicklungsboards können sehr komfortabel Anwendungen auf Mikrocontrollern realisiert werden. Die verschiedenen Entwicklungsboards sind mit Mikrocontroller der Firma ATMEL bestückt. Auf den verschiedenen Boards befinden sich Mikrocontroller unterschiedlicher Leistung und Größe. Die als Prototyp aufgebaute Modellfahrzeugsteuerung verwendet ein ArduinoUNO-Board [1]. Das Board ist mit dem Mikrocontroller Atmega 328p bestückt. Dieser Controller hat sowohl in der DIP<sup>4</sup>- als auch in der SMD<sup>5</sup>-Bauform 20 programmierbare Ein/Ausgänge. Davon können sechs als analoge Eingänge genutzt werden.

Um den Controller auf dem Arduino-Board für unterschiedliche Anwendungen nutzen zu können, wird ein Bootloader mitgeliefert. Dieser muss bei neuen Controllern zuerst über die ICSP (In Circuit Serial Programming)-Schnittstelle in den Mikrocontroller gebrannt werden. Danach erst kann über einen Level-Konverter der Chip über eine serielle Schnittstelle (zum Beispiel RS232), oder über die auf dem Board befindliche USB-Schnittstelle mit der Software auf dem Computer kommunizieren.

Die Open-Source Entwicklungsumgebung Arduino-IDE<sup>6</sup> bietet viele fertige Bibliotheken an. So ist es sehr einfach Funktionen wie Tonausgabe, EEPROM-Zugriff oder das Einlesen analoger Werte in den Hauptspeicher des Controllers vorzunehmen.

Für die Dekoder in den Miniaturfahrzeugen konnte der ArduinoUNO nicht verwendet werden. Das Entwicklungsboard ist fast doppelt so groß wie ein Modellfahrzeug. Den Fahrzeugabmessungen entsprechend musste für den Dekoder, ein spezielles Platinenlayout mit dem Entwicklungsprogramm Eagle<sup>TM7</sup> entworfen werden. Der Controller Atmega 328p ist in der SMD-Bauform auf der Dekoderplatine aufgelötet. Die Hardware auf dieser Platine ist ähnlich der auf dem ArduinoUNO verwendeten. Um Platz einzusparen wurde auf einen externen Quarz verzichtet und dafür der interne 8 MHz Quarz des Atmega 328p verwendet. Programmiert wird der Atmega-Chip über ein ArduinoUNO-Board. Dazu wird das Arduino-Board mit der USB-Schnittstelle eines Computers verbunden. Dann wird mittels eines vieradrigen Schnittstellenkabels, die Software auf den Controller aufgespielt. Der FT232R-Chip<sup>8</sup> auf dem ArduinoUNO dient hier als Level-Konverter von der USB-Schnittstelle auf die serielle Schnittstelle.

---

<sup>3</sup>Webseite: <http://www.arduino.cc>

<sup>4</sup>Dual-In-Line-Package, ein standardisiertes zweireihiges Gehäuse für elektronische Bauteile

<sup>5</sup>Surface-Mounted-Device, Gehäuse ohne Drahtanschlüsse welches dadurch sehr kompakt ist.

<sup>6</sup>Webseite: <http://arduino.cc/en/Main/Software>

<sup>7</sup>Webseite: <http://www.cadsoft.de>

<sup>8</sup>Future Technology Devices International



# **Kapitel 3**

## **Konzept der Steuerung**

Die Funktionsweise des Faller Car-System wird in diesem Kapitel erklärt. Dabei wird darauf eingegangen was benötigt wird um Modellfahrzeuge auf den Straßen einer Modelleisenbahnanlage fahren zu lassen. Da eine analoge Steuerung auch für dieses System nicht mehr zeitgemäß ist wird im Weiteren erklärt, welche Vorteile ein digitales System bietet. In dem dann folgenden Abschnitt wird die Steuerung und Funktionsweise des von der Firma Faller entwickelten Car-System dargestellt [5].

### **3.1 Funktionsweise des Faller Car-System**

Die Fahrzeuge im Modell benötigen Komponenten wie einem Motor, die Lenkung und zur Speicherung der Antriebsenergie einen Akku. Nach dem Einschalten der Motorspannung durch einen Schalter am Fahrzeugboden, setzen sich die Fahrzeuge in Bewegung. Ein kleiner Lenkschleifer mit einem Permanentmagneten auf der beweglichen Vorderachse führt das Fahrzeug auf der Straße entlang einer vorgegebenen Fahrspur.

Handelsübliche Car-System Fahrzeuge, wie die der Firma Faller, sind mit den folgenden Merkmalen ausgestattet [14]:

- Leistungsstarker Glockenanker-Motor,
- EIN/AUS-Schiebeschalter,
- Permanentmagnet auf einer dreipunktgelagerten Vorderachse,
- Ladebuchse um den Akku zu Laden,
- Antriebseinheit an der Hinterachse bestehend aus Achswelle, Antriebsschnecke und Schneckenzahnrad,
- Reed-Sensor durch den das Fahrzeug an einer Stoppstelle anhalten kann.

Die Abbildung 3.1 zeigt drei originale Faller Car-System Fahrzeuge. Auf dem linken und rechten Bild ist ein Fahrzeug von unten dargestellt. Hier kann gut der Lenkschleifer und das an der Hinterachse sitzende Schneckenzahnrad erkannt werden. Im mittleren



Abbildung 3.1: Ansicht von Faller Car-System Fahrzeugen [5].

Bild ist das Fahrzeug geöffnet. Im inneren kann man den Schalter (links), den Motor (Mitte), sowie die zwei Knopfzellenakkus (rechts) gut sehen.

Damit das beschriebene Fahrzeug sicher fahren kann, wird eine spezielle Fahrspur benötigt. In die Straße wird ein Fahrdräht oder Magnetband eingelassen. Der Fahrdräht ist wie auch das Magnetband magnetisch. Sie haben die Aufgabe den kleinen Permanentmagneten, der auf der Vorderachse des Fahrzeugs befestigt ist, anzuziehen. Dadurch wird der Lenkeinschlag der Vorderräder geändert und das Fahrzeug fährt fast unsichtbar gelenkt immer der Fahrspur nach, die durch den eingelassenen Fahrdräht oder Magnetband in der Straße vorgegeben ist. In der Abbildung 3.2 ist die Lenkung mit Lenkschleifer links auf einem Magnetband und rechts auf einem Fahrdräht dargestellt.



Abbildung 3.2: Lenkachse mit Permanentmagnet auf Magnetband (links) und Fahrdräht (rechts).

So ausgerüstet sind die Fahrzeuge bereit für den Betrieb auf einer Modelleisenbahn-Anlage. Bei gleichzeitigem Einsatz von mehreren Fahrzeugen kommt es unweigerlich zu Problemen. Die Car-System Fahrzeuge fahren aufeinander auf oder halten ruckartig an. Aus diesem Grunde besitzen Faller Car-System Fahrzeuge einen seitlich angeordneten Reed-Sensor<sup>1</sup>. Mit Hilfe von Magneten oder Magnetspulen unterhalb der Fahrbahn, wird ein Magnetfeld neben dem Magnetband oder Fahrdräht erzeugt. Der Reed-Sensor öffnet den Fahrstromkreis und das Fahrzeug bleibt stehen. Derartige Stopstellen werden

---

<sup>1</sup>Ein Reed-Sensor ist ein in Glas eingeschmolzener Kontakt, der von einem Vakuum umgeben ist und in der Nähe eines Magnetfeldes, schaltet.

vor Kreuzungen, an Haltestellen, auf Parkplätzen oder an Bahnübergängen installiert. Um auch eine Abbiegung (Abzweigung) realisieren zu können, muss der fest auf der Lenkachse sitzende Permanentmagnet entsprechend umgelenkt werden. Das Fahrzeug wechselt so auf einen anderen Fahrdraht. Bei der Verwendung von Magnetband kann eine Abzweigung mittels eines handelsüblichen Servo hergestellt werden. Der Servo übernimmt hier die Funktion einer Weiche ähnlich wie im Modellbahnbetrieb.

Für die analoge Fahrzeugsteuerung gibt es verschiedene Zubehörteile. Diese ermöglichen es beispielsweise, Busse zu erkennen und diese in Bushaltestellen einfahren zu lassen. Dazu besitzen die Busse einen zusätzlichen Magneten auf der entsprechenden Fahrbahnseite, der einen Reed-Sensor unter der Fahrbahn aktiviert. So kann der Bus in die Bushaltestelle geleitet werden alle anderen Fahrzeuge fahren vorbei [5].

## 3.2 Digitaler Funktionsaufbau

Die analoge Fahrzeugsteuerung, wie sie im Faller Car-System verwendet wird, hat Einschränkungen:

- Fahrzeuge können nur fest vorgegebene Strecken fahren. Ein Bus fährt zum Beispiel immer an die Bushaltestelle.
- Die Programmierbarkeit eines Fahrwegs in Form einer Ablaufsteuerung ist nicht möglich.
- Die Fahrzeuge können nur an bestimmten, dafür vorgesehenen Punkten angehalten werden.
- Eine Ausbaumöglichkeit durch eine zentrale Steuereinheit mit Zugriff auf alle im Gelände befindlichen Fahrzeuge ist nicht möglich.
- Zusätzliche Funktionen wie Blinken, Bremslicht oder Einschalten der Scheinwerfer bei Dämmerung sind in analoger Technik aufgrund des schaltungstechnischen Platzbedarfs in den kleinen Fahrzeugen ohne Mikrocontroller nicht realisierbar,
- die analoge Technik erfordert viel Material und Verkabelung unter der Anlage um die Fahrzeuge sicher steuern zu können.

Durch eine digitalisierte Modellfahrzeugsteuerung ergeben sich deutlich mehr Möglichkeiten. Mit dem Einsatz einer Digitalzentrale als Steuereinheit, lassen sich alle Fahrzeuge mittels eigener Adresse bedienen egal wo diese sich gerade befinden.

In der vorliegenden Arbeit fiel die Wahl auf eine Datenübertragung mittels Infrarotsignalen. Dieses System wurde gewählt, weil es wenig Platz in den Fahrzeugen benötigt. Für den Empfang und die Demodulierung der Infrarotsignale wurde ein handelsüblicher IR-Empfänger verwendet. Der Empfänger enthält dabei alle notwendigen Bauelemente in einem Gehäuse. Zum Senden der Daten werden einige Infrarotsendedioden an der Decke über der Modellfahrzeuganlage verteilt.

Das digitale Steuerungssystem kann mittels einer herkömmlichen Digitalzentrale aus dem Modellbahnbereich gesteuert werden. Alle Fahrzeuge können dabei mit einer eigenen Adresse angesprochen werden. Die Datenübertragung von der Digitalzentrale aus zu den Fahrzeugen erfolgt über das im Kapitel 2.1 beschriebene standardisierte DCC-Protokoll.

Um digitale Steuerungsbefehle im DCC-Format an die Fahrzeuge zu übertragen, benötigen diese einen geeigneten Signalempfänger und einen Mikrocontroller der als Dekoder arbeitet. Der Dekoder, ähnlich dem in der Modellbahnlokomotive verwendeten, ist das Herzstück des Fahrzeugs. Er hat die Aufgabe, die im Übertragungsprotokoll enthaltenen Befehle zu dekodieren und sie an die im Fahrzeug untergebrachten Aktronen zu leiten. Eine Übersicht über die einzelnen Komponenten ist in der Abbildung 3.3 dargestellt.

Der erwähnte IR-Empfänger benötigt auf der Sendeseite einen entsprechenden IRSender. Dieser wandelt die DCC-Steuerungsdaten der Digitalzentrale in Infrarotsignale um. Diese Signale werden dann mit einer Wellenlänge von etwa 875 Nanometer (nm) in Richtung Fahrzeug gesendet. Die Abbildung 3.4 zeigt ein Gestamtbild des digitalen Steuerungs- und Datenübertragungssystems zu den Fahrzeugen.

Um eine ausreichende hohe Versorgungsspannung für den IR-Empfänger und den Mikrocontroller im Fahrzeug zu gewährleisten, muss die Akku-Spannung der Fahrzeuge angepasst werden. Das Faller Car-System arbeitet mit nur 1,2 Volt (eine Zelle) bei kleinen Fahrzeugen oder 2,4 Volt (zwei Zellen) in größeren Fahrzeugen. Diese Spannungs niveau reicht für die analoge Steuerung gerade aus, um den kleinen Motor im Fahrzeug zu betreiben. Um den Dekoder und den IR-Empfänger zu betreiben, werden bei den hier aufgebauten Prototypen mindestens 2,7 Volt benötigt. Die Fahrzeuge bieten bei der Verwendung von kleinen Akkus ausreichend Platz. Mit drei Akkuzellen hat das Fahrzeug eine Spannung von 3,6 Volt. In der aufgebauten Testanlage fahren speziell umgerüstete Modellfahrzeuge im Maßstab 1:120. Personenfahrzeuge in dem genannten Maßstab lassen sich aufgrund des Platzmangels mit der hier beschriebenen digitalen Steuerung nicht realisieren. In Personenfahrzeugen ist es nicht möglich drei Akkuzellen und einen Dekoder unterzubringen. Deshalb wurden nur größere Fahrzeuge, wie Versorgungsfahrzeuge oder Busse im Rahmen dieser Arbeit umgerüstet.

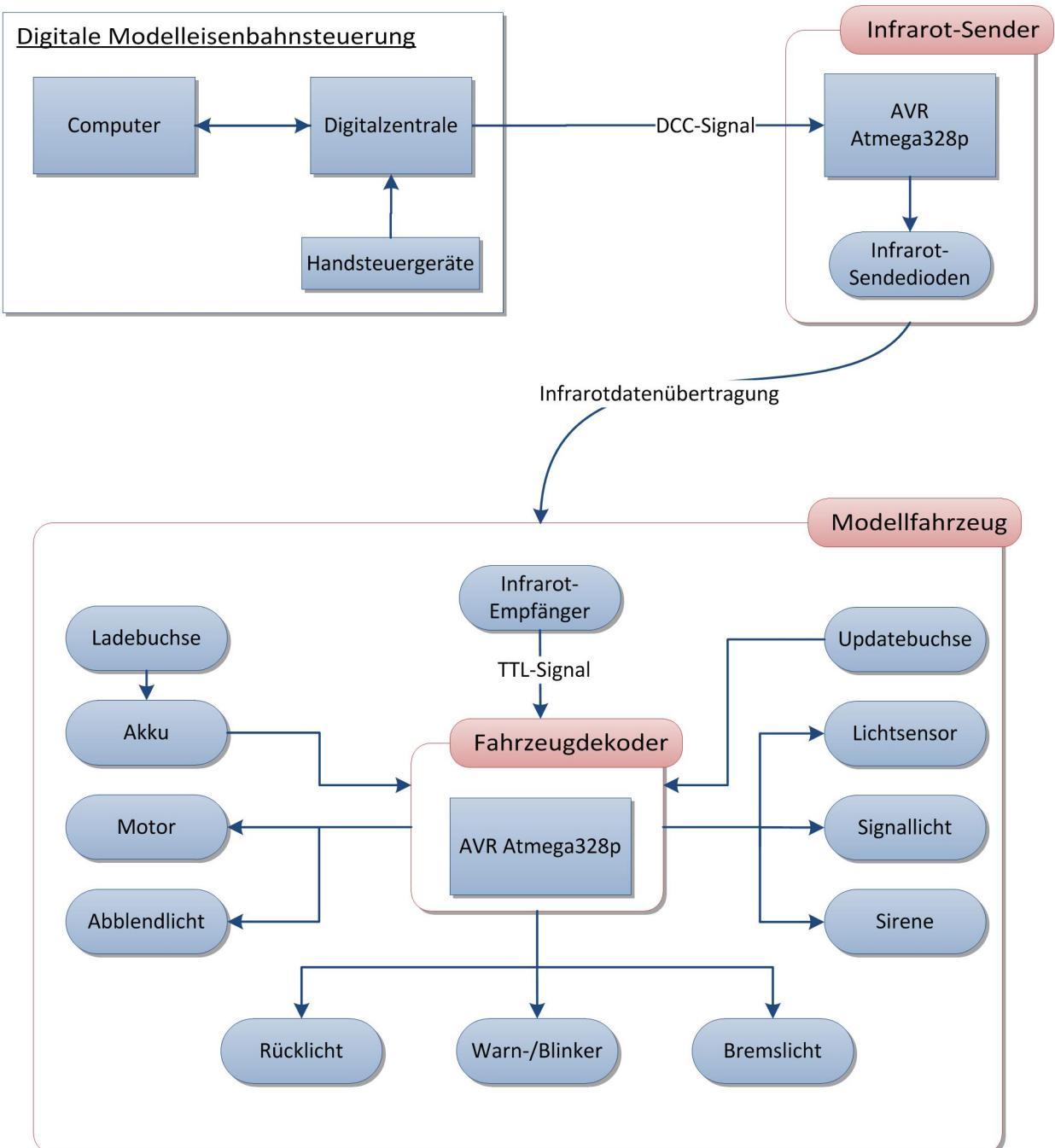


Abbildung 3.3: Digitale Fahrzeugsteuerung und seine Komponenten.

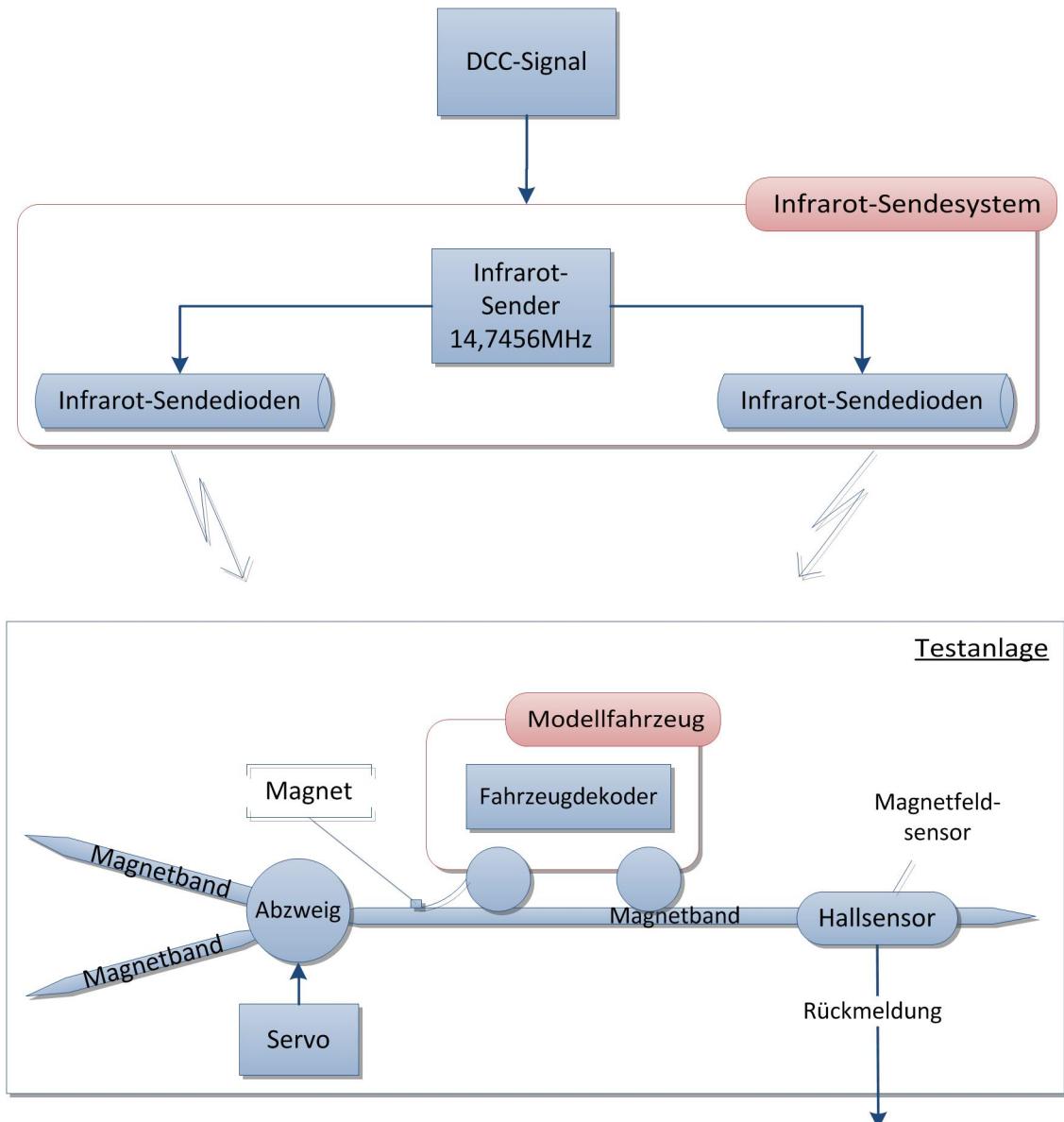


Abbildung 3.4: Blockbild des Funktionsaufbau.

# **Kapitel 4**

## **Funktionsweise der Infrarotdatenübertragung**

Hier wird erläutert, wie die Datenübertragung zu den Modellfahrzeugen funktioniert. Es wird dazu erklärt wie die Infrarotdatenübertragung arbeitet und warum eine spezielle IR-Empfänger Hardware für die Übertragung des DCC-Protokolls notwendig ist.

### **4.1 Infrarotdatenübertragung**

Bei einer Infrarotdatenübertragung werden die Daten mittels kurzwelligem Infrarotlicht übertragen. Dieses Licht ist für das menschliche Auge nicht sichtbar. Die verwendete Wellenlänge im Übertragungskanal liegt im sogenannten nahen Infrarotbereich (Near Infrared, NIR). Die NIR-Strahlung umfasst einen Bereich von 780 nm bis 1400 nm Wellenlänge, der sich direkt an den sichtbaren (roten) Bereich des Lichtspektrum von 380 nm bis 750 nm des menschlichen Auges, anschließt [3]. Gebräuchliche Infrarotsender arbeiten mit einer Wellenlänge von 850 nm bis 950 nm. IR-Empfänger finden oft dort Anwendung, wo Übertragungsstörungen keine größeren Probleme darstellen [7]. Infrarotübertragungsstecken finden sich im alltäglichen Gebrauch, um zum Beispiel einen Fernseher von Ferne mit einer Fernbedienung zu steuern. Früher wurden IR-Strecken auch zur Datenübertragung zwischen Mobilfunkgeräten oder zur Verbindung mit einem Computer verwendet. Heute wird dies meist über eine sichere und schnelle Übertragung wie Bluetooth gelöst.

#### **4.1.1 Frequenzen**

Bekannte Infrarotdatenübertragungsstecken, wie zum Beispiel die Fernbedienung des Fernsehers, verwenden je nach Hersteller für die Übertragung verschiedene Frequenzen. Zusätzlich haben die Hersteller unabhängig voneinander verschiedene Übertragungsprotokolle entwickelt. Für den Anwender hat das den Vorteil, dass mit einer TV-Fernbedienung nicht unbeabsichtigt die Stereoanlage gesteuert wird. Für die Übertragung verwenden die meisten Hersteller Trägerfrequenzen im Bereich von 36 bis 40 kHz [2]. Auf diese Frequenzen werden die Daten aufmoduliert. Der Empfänger filtert dann

aus dieser Frequenz die Daten heraus und wandelt diese in ein TTL<sup>1</sup>-Signal um. Durch die gewählte Trägerfrequenz wird eine gewisse Störsicherheit erreicht. Der verwendete IR-Empfänger besitzt einen Bandpassfilter der alle zufälligen Störsignale herausfiltert [7].

Um Daten über eine Infrarotübertragungsstecke senden zu können, müssen die Daten serialisiert vorliegen. Diese seriellen Daten bestehen aus mehreren Null- und Eins-Bit, welche wie unter Kapitel 2.1 beschrieben unterschiedliche Zeiten besitzen. Um die Daten störungssicher zu übertragen, werden sie zusätzlich auf eine sogenannte Trägerfrequenz aufmoduliert. Diese muss je nach verwendetem IR-Empfänger eine bestimmte Schwingungsdauer aufweisen. Die Schwingung gibt dabei die Dauer an, wie oft ein Impuls eines Null- oder Eins-Bit unterbrochen wird. Somit wird auch die gesendete Datenfrequenz und die Übertragungsgeschwindigkeit festgelegt.

#### 4.1.2 Datenraten

Die Datenraten einer herkömmlichen IR-Datenübertragungsstecke einer TV-Fernbedienung sind sehr gering. So kann zum Beispiel ein Signalpaket mit dem RC-5<sup>2</sup> und RC-6 Verfahren der Firma Philips, welches 14 Bit groß ist nur etwa zehnmal pro Sekunde wiederholt ausgestrahlt werden. Eine Wiederholung derselben Daten ist für eine sichere Datenerkennung zwingend erforderlich. Ein IR-Empfänger ist, weil keine bidirektionale Datenübertragung genutzt wird, nicht in der Lage dem Sender zu signalisieren, dass sein Datenpaket fehlerhaft übermittelt wurde. Das ergibt ohne Wiederholung der Datenpakte eine Übertragungsrate von gerade mal 140 Bit/s [2].

## 4.2 Hardware

Die Hardware des IR-Sender mit mehreren IR-Sendedioden ist bei dem aufgebauten Prototyp auf einer Punktrasterleiterplatte untergebracht. Die Sendesteuerung der IR-Sendedioden übernimmt ein Mikrocontroller Atmega328p-PU von der Firma AVR. Dieser besitzt einen Arduino-Bootloader (siehe Abschnitt 2.2). Die Schaltung (siehe Anhang A.2) realisiert die IR-Übertragungsstrecke mit dem für die Modelleisenbahn gängige DCC-Datenformat, wie es in Abbildung 2.1 in Kapitel 2 erläutert ist.

Das von der Digitalzentrale an den IR-Sender übertragende DCC-Signal besteht, wie dargelegt aus zwei Pegeln die durch den Nulldurchgang getrennt werden. Durch die Eigenschaft, dass jeder Signalabschnitt vor und hinter dem Nulldurchgang die gleiche Zeidauer besitzt, wird für die Übertragung der Daten über die Infrarotstecke nur der positive Signalanteil verwendet. Bei entgegengesetzter Polung, wird der negative Impuls verwendet [11]. Es entsteht dabei kein Datenverlust, da die Daten auf dem positiven und negativen Pegel identisch sind und über feste Kabelverbindungen von der Digitalzentrale an die IR-Sende-Hardware geführt werden. Dieses halbe Rechtecksignal mit einem

---

<sup>1</sup>Transistor-Transistor-Logik mit High-Pegel von 5 Volt.

<sup>2</sup>Der Code wurde von der Firma Philips entwickelt und wird bei der Infrarotsteuerung von Fernsehern oder Audiogeräten verwendet. Weitere Details zu diesem Protokoll sind unter: [http://www.stefan-buchgeher.info/elektronik/rc5/rc5\\_doku.pdf](http://www.stefan-buchgeher.info/elektronik/rc5/rc5_doku.pdf) zu finden.

Null- oder Eins-Bit, wird durch den Mikrocontroller im Sender einer Trägerfrequenz aufmoduliert.

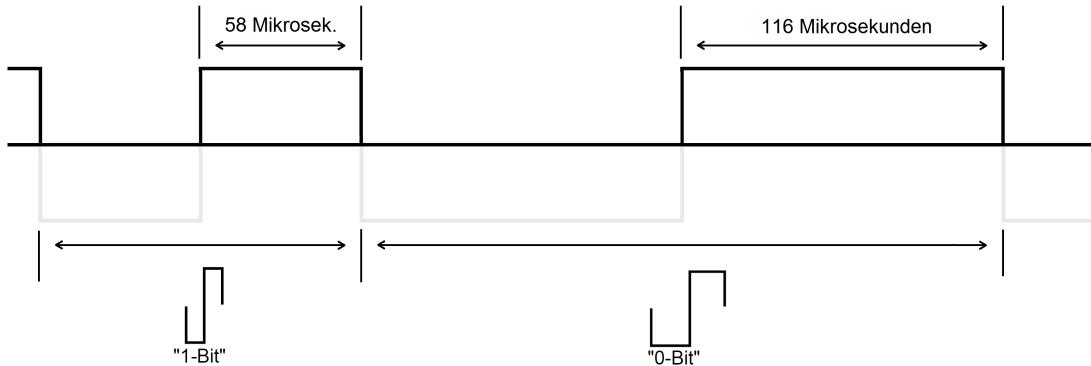


Abbildung 4.1: Oberer Teil des DCC-Signal.

In der Abbildung 4.1 ist, das DCC-Signal, wie es an dem Mikrocontroller des IR-Senders anliegt, dargestellt. Der untere, zweite Teil des Differenzensignals wird durch die Diode, auf der Schaltung im Anhang A.2 zusehen, abgetrennt und nicht an den Mikrocontroller übergeben.

Für die Datenübertragung im DCC-Format ist ein fester Rahmen vorgegeben. Nur mit einem speziellen, ausreichend schnellen IR-Empfänger kann ein Eins-Bit mit einer Länge von nur 58 Mikrosekunden übertragen werden. Die maximale Übertragungsgeschwindigkeit liegt dann bei etwa 17 kbit/s<sup>3</sup>. Der für die Modellfahrzeuge verwendete IR-Empfänger TSOP7000 von VISHAY arbeitet mit einer Trägerfrequenz von 455 kHz<sup>4</sup>. Mit dieser hohen Trägerfrequenz und einer entsprechenden IR-Wellenlänge, ist es möglich störungsfrei maximal circa 20 kbit/s auf einer Distanz bis 20 Meter zu übertragen [17]. Bei Versuchen mit den Fahrzeugen hat sich gezeigt, dass diese auch bei größeren Entfernungen von mehr als 15 Metern die gesendeten Daten erkennen.

Die einzige Aufgabe des Mikrocontrollers im IR-Sender ist es, das DCC-Datensignal auf die Trägerfrequenz zu modulieren. Das heißt der Mikrocontroller muss den IR-Sendedioden das modulierte Signal in der Form wie es die Abbildung 4.4 zeigt, bereitstellen. Um eine Schwingung mit einer Frequenz von 455 kHz zu erzeugen, muss jedes DCC-Datenbit in Impulse mit einer Länge von circa 2,2 Mikrosekunden unterteilt werden [17].

$$455 \text{ kHz} = 455000 \text{ Hz} = \frac{1}{455000 \text{ s}} \quad (4.1)$$

$$= \frac{1}{455 \text{ ms}} \quad (4.2)$$

$$= \frac{1}{0,455 \mu\text{s}} \approx 2,1978 \mu\text{s} \quad (4.3)$$

<sup>3</sup>1000 / 58 = 17,24 Kilobit pro Sekunde

<sup>4</sup>Kilohertz, entsprechen tausend Schwingungen pro Sekunde

### 4.2.1 Infrarot-Empfänger

Mit dem IR-Empfänger, der im Fahrzeug sitzt, werden die vom Sender gesendeten Daten aus dem Umgebungslicht herausgefiltert. Dazu besitzen IR-Empfänger, wie der TSOP7000, eine Verstärkungsregulierung (Automatic Gain Control (AGC)). In dem 3-poligen Gehäuse des Bausteins sind auch ein Bandpass und ein Demodulator untergebracht [17]. Die Verstärkungsregulierung dient dazu, die empfangenen Trägerfrequenz zu verstärken. Der nachgeschaltete Bandpass lässt bevorzugt die spezielle Trägerfrequenz des IR-Empfängers, dargestellt in Abbildung 4.2 passieren. Andere Frequenzen werden stark gedämpft. Deshalb besitzt der IR-Empfänger bei einer Frequenz von 455 kHz die höchste Empfindlichkeit. In diesem Bereich arbeitet die Datenübertragung am sichersten. Bei einer geringen Abweichungen nach links oder rechts der Trägerfrequenz, werden Störungen durch nicht erkennen der Trägerfrequenz immer größer. Dann kann es, bei Unterbrechung des IR-Sendesignals sogar vorkommen, dass beim Wiedereinschalten die gesendeten Daten als Störung erkannt werden. Links in der Abbildung ist dargestellt, wie die Trägerfrequenz vom IR-Empfänger in ein TTL-Signal umgewandelt wird, und welche Verschiebungen dabei auftreten können.

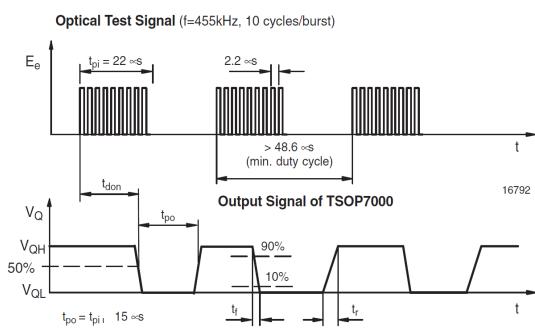


Figure 1. Output Function

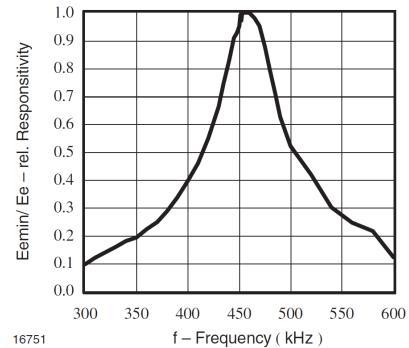


Figure 5. Frequency Dependence of Responsivity

Abbildung 4.2: Eingangsfrequenz des TSOP 7000 [17].

Ein Demodulator am Ausgang des IR-Empfängers dient dazu die Daten von der Trägerfrequenz zu trennen und in ein TTL-Signal umzuwandeln. Er hat auch die Aufgabe die maximale Anzahl der Schwingungen pro Datenbit zu beschränken. Das hat den Vorteil, dass keine Störungen durch andere IR-Strahlung mit längeren Bitzeiten auftreten können. Beim TSOP7000 muss ein Datenbit mindestens 22 Mikrosekunden Dauer haben (zehn Zyklen der Trägerfrequenz) und darf maximal 500 Mikrosekunden lange andauern. Alles Andere wird vom Empfänger als Störfrequenz herausgefiltert und erreicht nicht den Ausgang. Der IR-Empfänger TSOP7000 benötigt als Versorgungsspannung mindestens 2,7 bis 5,5 Volt und hat dabei eine Stromaufnahme von circa 2 mA. Bei einer Wellenlänge von 875 nm hat der IR-Empfänger die größte spektrale Empfindlichkeit, wie die Abbildung 4.3 zeigt. [17].

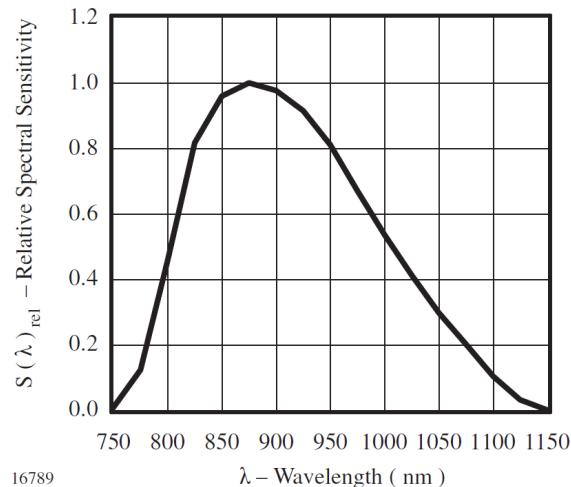


Figure 10. Relative Spectral Sensitivity vs. Wavelength

Abbildung 4.3: Spektrale Empfindlichkeit der Empfängerdiode des TSOP7000 [17].

#### 4.2.2 Infrarot-Sender

Der IR-Empfänger TSOP7000 kann IR-Signale aus einer Entfernung von bis zu 20 Metern noch sicher entschlüsseln. Damit solche Entferungen möglich sind müssen die Sende-Dioden mit der maximal möglichen Sendeleistung betrieben werden. Dazu muss eine Durchlassstrom von 200 mA durch die Dioden fließen.

Um eine erforderliche Sendeleistung zu erhalten wurden TSH6203<sup>5</sup> IR-Sende-Dioden von der Firma Vishay mit einer Wellenlänge von 875 nm eingesetzt. Diese werden über ein Darlington-Array<sup>6</sup> vom Mikrocontroller angesteuert. Die Sende Hardware ist im Anhang A.2 dargestellt.

### 4.3 Datenverarbeitung

Das DCC-Signal von der Digitalzentrale kommend, wird über einen  $10\text{ k}\Omega$  Widerstand und einen Optokoppler an den Mikrocontroller geführt. Das Eingangssignal liegt am Port D2 an. In der Arduino-IDE wird dieser Port über den Pin 2 adressiert. Dieser Port löst einen internen Interrupt aus, wenn anstehende DCC-Daten verarbeitet werden sollen. Zusätzlich befindet sich am Port B5 (Pin 13) eine LED (Light-Emitting Diode). Diese signalisiert, wenn DCC-Daten erkannt wurden und über die IR-Dioden gesendet werden.

Die Software für die Steuerung des Mikrocontrollers wurde mit Hilfe der Entwicklungsumgebung Arduino-IDE (siehe auch Kapitel 2.2) geschrieben. Die Arduino-IDE bietet die Möglichkeit, einfach und komfortabel alle Funktionen eines Mikrocontrollers

<sup>5</sup>Datenblatt: <http://pdf.dzsc.com/TSH/TSHA620.pdf>

<sup>6</sup>Leistungsverstärker ULN2003 mit sieben Transistoren und einem Ausgangsstrom mit je 500 mA.

## 24 KAPITEL 4. FUNKTIONSWEISE DER INFRAROTDATENÜBERTRAGUNG

zu überwachen und zu steuern. Der Code des Mikrocontrollers zur Steuerung des IR-Senders ist wie folgt aufgebaut:

---

```
1 #include "TimerOne.h" // Bibliothek für die Timeransteuerung einbinden

3 const int dccPin = 2;           // Pinnummer des DCC Eingangs
4 const int irPin = 9;           // Pinnummer der IR-LED
5 const int ledPin = 13;          // Pinnummer der Status LED

7 void setup()
8 {
9     pinMode(ledPin, OUTPUT);    // Signalisierung für Aktivität
10    pinMode(dccPin, INPUT);
11    attachInterrupt(0, dcldata, CHANGE); // Interrupt an Pin 2

13    Timer1.initialize(2);        // initialisiere timer1, setzte 2
14        Mikrosekunden Periode
14    Timer1.pwm(irPin, 512);    // setup PWM auf irPin, 50% duty cycle
15    Timer1.stop();
16 }

18 void dcldata() {
19     if (digitalRead(dccPin) == HIGH) { // Modulierung einschalten
20         Timer1.pwm(irPin, 512);      // PWM, 50% duty cycle
21         Timer1.start();
22         digitalWrite(ledPin, HIGH);
23     }
24     else {                      // Modulierung abschalten
25         Timer1.stop();
26         digitalWrite(irPin, LOW);
27         digitalWrite(ledPin, LOW);
28     }
29 }

31 void loop()
32 {
33 }
```

---

Listing 4.1: IR-DCC-Sender Quellcode.

In der Software wird die benötigte Trägerfrequenz von 455 kHz für den IR-Sender mittels Puls-Weiten-Modulation (PWM) erzeugt. Da die im Mikrocontroller implementierte PWM nicht mit der benötigten Frequenz zur Verfügung steht, ist eine Anpassung des Timers notwendig. Der Timer wird immer dann eingeschaltet, wenn das DCC-Signal einen ansteigenden Flanke aufweist. Bei einer fallenden Flanke wird der Timer abgeschaltet. Um die Flanken exakt zu erkennen, besitzt der Mikrocontroller, wie oben dargestellt, einen Interrupt auf dem Eingangspin. Dieser Interrupt startet die Funktion *dcldata* ausführt. Diese ermittelt in Abhängigkeit des Pegels am Eingangspin, ob die Modulation gestartet oder angehalten werden soll.

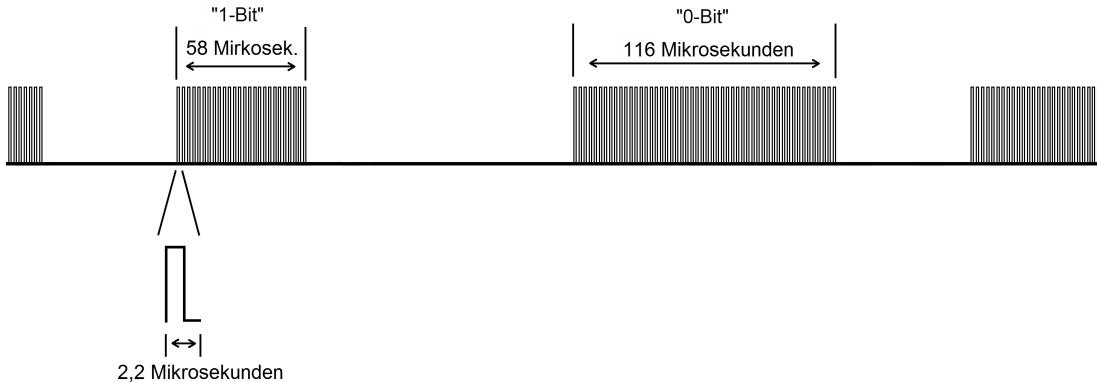


Abbildung 4.4: DCC-Signal mit Trägerfrequenz von 455 kHz.

Wie in der Abbildung 4.4 gezeigt ist, sollen die erzeugten Perioden für HIGH und LOW gleich lang sein. Um dies zu erreichen, wird das Tastverhältnis (Duty-Cycle) auf 50% eingestellt. Die Aufteilung des Tastverhältnisses wird von der Timer-Interrupt-Zeit festgelegt. Für 455 kHz ist, wie oben dargelegt, eine Periodendauer von 2,2 Mikrosekunden erforderlich. Ein Timer im Mikrocontroller arbeitet immer in Abhängigkeit von seiner eingestellten, vorhandenen Taktfrequenz. Die Taktfrequenz des eingesetzten Mikrocontroller liegt bei 16 MHz. So ergibt sich für die nächstmögliche ganzzahlige Periodendauer von zwei Mikrosekunden eine Frequenz von 500 kHz.

$$\frac{1}{f} = 2 \mu s \quad (4.4)$$

$$f = \frac{1}{2 \mu s} \quad (4.5)$$

$$f = 500 \text{ kHz} \quad (4.6)$$

Da diese Frequenz nicht im optimalen Bereich der Bandpass-Kurve des IR-Empfängers TSOP7000 liegt (siehe Abbildung 4.2), funktioniert die Datenübertragung nur so lange, wie die Infrarotstrecke zum Empfänger nicht unterbrochen wird. Bei fahrenden Modellfahrzeugen kann dies allerdings häufiger auftreten. Nach der Unterbrechung stimmen die Periodenlängen kurz nicht überein. Da es keine Sendepausen im DCC-Signal gibt, werden die gesendeten DCC-Daten vom IR-Empfänger als Störsignal erkannt.

Da im Befehlsvorrat der verwendeten Software keine weitere zeitrelevante Funktion vorhanden ist, musste eine Hardwarelösung gefunden werden. Durch Austausch des Schwingquarz, der die Taktfrequenz des Controllers liefert, konnte eine Trägerfrequenz von nahezu 455 kHz erreicht werden. Der externe 16 MHz Quarz des verwendeten Atmega 328p im IR-Sender, wurde deshalb durch einen handelsüblichen Quarz mit einer Frequenz von 14,7456 MHz ersetzt.

$$16 \text{ MHz} = 2 \mu s \rightarrow 32 \text{ MHz} = 1 \mu s \rightarrow 1 \text{ MHz} = 32 \mu s \quad (4.7)$$

$$14,55 \text{ MHz} = 2,2 \mu s \quad (4.8)$$

$$14,7456 \text{ MHz} = 2,17 \mu s \quad (4.9)$$

Durch den 14,7456 MHz Quarz ist der Mikrocontroller jetzt langsamer getaktet als zuvor mit dem 16 MHz Quarz. Somit erhöht sich die Pausenzeit der Perioden des Timers. Statt der geforderten 2,2 Mikrosekunden für exakt 455 kHz, liefert der Controller mit dem veränderten Quarz Schwingungen mit der Länge von 2,17 Mikrosekunden.

$$\frac{1}{f} = 2,17 \mu s \quad (4.10)$$

$$f = \frac{1}{2,17 \mu s} \quad (4.11)$$

$$f \approx 460,83 \text{ kHz} \quad (4.12)$$

Damit liegt die Frequenz jetzt bei circa 460 kHz. Einen Schwingquarz, der den genauen Wert der Trägerfrequenz liefert, gibt es nicht. Die einwandfreie Funktion des Bandpass im Empfänger zeigt, dass diese Abweichung durch die Filterkurve in Abbildung 4.2 toleriert wird.

## 4.4 Störeinflüsse

Die Datenübertragung über das Infrarotsignal läuft – solange Sichtkontakt herrscht – sehr stabil. Auch größere Entfernen stören die Datenübertragung nicht. Eine kurze oder längere Signalunterbrechung hat zur Folge, dass es Probleme bei der Wiederaufnahme der Verbindung gibt. Diese treten vor allem auf, wenn die Frequenz zu sehr von der durch den Bandpass des Empfängers vorgegebenen Mittelfrequenz abweicht. So kann es vorkommen, dass die gesendeten Daten vom Empfänger als Störlicht gewertet werden. Je nach Entfernung tritt dieses Verhalten stärker auf und liegt vor allem an der nicht ganz exakten Trägerfrequenz, die für die Übertragung verwendet wird. Umso genauer die Frequenz mit der des Empfängers übereinstimmt, desto weniger tritt diese Störung auf.

Ab und zu kommt es vor, dass ein fehlerhaftes Datenpaket empfangen wird. Deshalb sollten die übertragenen Daten auf jeden Fall eine Prüfsumme enthalten. So kann der Empfängerdekoder entscheiden, ob die erhaltenen Daten korrekt übermittelt wurden.

Generell sollte vom Sender eine ständige Sichtverbindung zu den Modellfahrzeugen bestehen. Dabei kann der IR-Empfänger in den Fahrzeugen auch unter Plastikmaterial verbaut werden. Helles Plastikmaterial lässt dabei noch circa 70% der Infrarotstrahlung passieren. Bei einer ausreichenden Sendeleistung über die gesamte Fahrbahnfläche ist das empfangende Signal für eine digitale Steuerung der Fahrzeuge ausreichend. Ein guter Empfang wird auch dadurch erzielt, indem die Infrarot Sende-Dioden mit ihrer vollen Leistung angesteuert werden. Zusätzliche Verbesserungen des IR-Empfang kann mit einer räumlichen korrekten Ausrichtung der Sende-Dioden erreicht werden. Mit erhöhter Sendeleistung kann sich Störlicht auf die Datenübertragung weniger auswirken.

# Kapitel 5

## Fahrzeugdekoder

Für die digital gesteuerten Modellfahrzeuge wird eine Kombination aus Hardware und Software zur Dekodierung und Steuerung benötigt. Dazu wird in diesem Kapitel beschrieben wie die Modellfahrzeugsteuerung arbeitet. Der entwickelte Fahrzeugdekoder wird vorgestellt und seine Arbeitsweise beschrieben.

### 5.1 Hardware

Die Infrarotstrahlung wird durch den IR-Empfänger TSOP7000 aufgenommen. Eine erste Version des Fahrzeugdecoders bestand nur aus dem Mikrocontroller mit externem 16 MHz Quarz und einem IR-Empfänger die mit Kupferlackdraht ohne eine Leiterplatte verkabelt waren. Da bei diesem System kein Software-Update durchgeführt werden konnte, wurde ein geeignetes Platinen-Layout mit einer seriellen Schnittstelle entworfen.

Die eigens für die Fahrzeugsteuerung entworfene Hardware realisiert alle Funktionen im Fahrzeug. Die Dekodierung übernimmt dabei wieder ein Mikrocontroller vom Typ Atmega328p-AU, der Firma AVR. Mit dem Fahrzeugdekoder wurden drei Modellfahrzeuge mit unterschiedlicher Bauform (Bus, LKW, Krankenwagen) und Größe ausgerüstet. Zusätzlich zum Fahrwerk wurde auch eine komplette Beleuchtungsanlage installiert, somit besitzen alle Fahrzeuge Licht, Rücklicht, Bremslicht und Blinker. Für die Energieversorgung werden NIMH-Akkus (Nickel-Metallhydrid-Akkumulator) in kleiner Ausführung verwendet. Jeder Akku besitzt 1,2 Volt. Um dem Controller und dem IR-Empfänger eine ausreichende Versorgungsspannung bereit zu stellen, müssen drei NIMH-Zellen im Fahrzeug untergebracht werden. So erhält man die benötigte Spannung von 3,6 Volt.

Eine geeignete Schaltung für den in SMD-Bauform verwendeten Atmega 328p ist im Anhang A.1 dargestellt. Die Platine mit der Hardware für den Dekoder besteht im Wesentlichen aus dem Mikrocontroller. Aus Platzgründen wird hierfür ein SMD-Chip, welcher funktionsäquivalent zu dem DIP-Chip auf dem ArduinoUNO-Board ist, verwendet.

Um den SMD-Chip auf der Dekoder Platine leicht mit einem neuen Softwareupdate aktualisieren zu können, ist eine vierpolige Schnittstelle vorgesehen. Diese besitzt die

normalen TX-, RX-, Reset- und Ground-Leitungen, welche für ein Update eines Mikrocontrollers mit dem Arduino-Bootloader über die serielle Schnittstelle erforderlich sind.

Aus Platzgründen wurde auf der Platine kein externer Quarz vorgesehen. Auf diese Weise konnte ein Platzgewinn auf der Dekoderplatine erzielt werden. Deshalb muss der Controller über den internen Quarz mit 8 MHz getaktet werden. Für den Einsatz des internen Quarz ist ein spezieller Arduino-Bootloader und angepasste Fuses beim erstmaligen Programmieren erforderlich.

Der TSOP7000 hat eine Stromaufnahme von weniger als 5 mA. Für Modellfahrzeuge ist eine geringe Stromaufnahme wichtig, um möglichst lange Akku Betriebszeiten zu gewährleisten.

### 5.1.1 Schaltungsaufbau des Dekoders

Wie im Kapitel 2.2 erwähnt stehen am Atmega 328p 20 Aus- und Eingänge zur Verfügung. Durch den computergestützten Entwurf des Platinen-Layouts war es möglich, die Belegung der Anschlusspins an die Bauteilbelegung anzupassen und somit eine kleinstmögliche Platine zu fertigen. Der TTL-Ausgang des IR-Empfängers ist an einen der Interrupt-Eingänge des Mikrocontrollers angeschlossen. Alle anderen Pins waren bis auf die Akkuspannungsmessung, dem Lichtsensor und der PWM-Ansteuerung des Motors frei wählbar. Der Lichtsensoreingang wird an AREF (Referenzspannung) und dem Eingang am Dekoder, der mit einem Pulldown-Widerstand bestückt ist, verbunden (siehe Schaltung im Anhang A.1). Für die Motoransteuerung reicht ein maximaler Strom von 20 mA, mit der jeder Ausgang des Mikrocontroller belastbar ist, nicht aus. Hier kommt ein Power-MOSFET-Treiber zum Einsatz. Der IRLML2402 kann eine Leistung bis zu 1,2 A schalten und benötigt keinen Kühlkörper [8]. Den Schutz des Mikrocontrollers vor Spannungsspitzen übernimmt eine Sperrdiode, die über den Motoranschlüssen liegt.

Aufgrund der geringen Größe der Modellfahrzeuge konnten nur sehr kleine Akkus eingebaut werden. In der kleinsten Ausführung hat der Akku nur eine Kapazität von 80 mAh. Bei größeren Fahrzeugen wo mehr Platz vorhanden ist, können Akkus mit einer Kapazität von 300 mAh und mehr installiert werden. Mit einer Akku-Ladung sollte ein Fahrzeug mindestens 30 Minuten fahren. Dabei nehmen der Mikrocontroller und der IR-Empfänger im Leerlauf ohne eingeschalteten Motor oder Beleuchtung etwa 11,5 mA auf.

Damit die Akkus der Fahrzeuge einfach geladen werden können, wurde ein speziell auf drei Zellen abgestimmtes digitales Ladegerät entworfen. Das Ladegerät wurde auf der Hardwaregrundlage des ArduinoUNO aufgebaut und erkennt verschiedene Kapazitäten der Akkus automatisch. Ein Schnellladevorgang dauert unabhängig des verwendeten Akkus circa eine Stunde. Dabei kann der Akku mit bis zu einem zweifachen seiner Kapazität geladen werden.

## 5.2 Software

Für eine korrekte Fahrzeugsteuerung müssen die empfangenen DCC-Daten fehlerfrei dekodiert und umgesetzt werden. Die Software dazu besteht aus circa 1800 Zeilen (Lines of Code) und gliedert sich in mehrere Phasen, welche durchlaufen werden:

1. **Initialisierung** der Software. Hier werden alle Einstellungen, die während eines Betriebs notwendig sind, vorgenommen.

- **Einlesen der Registerwerte:** Es werden Registerwerte in die vorhandenen globalen Variablen geladen. Die Werte sind dafür im EEPROM (Electrically Erasable Programmable Read-Only Memory) hinterlegt, der auch nach Abschaltung der Versorgungsspannung die Daten speichert. Diese Werte können über eine Programmierung im Service Mode des Dekoders verändert werden.
- **Port-Initialisierung:** Alle aktiven Pins des Mikrocontrollers werden dabei in den jeweiligen Arbeitsmodus gesetzt werden. Auf diese Weise wird festgelegt ob ein Pin als Ein- oder Ausgang verwendet wird. Außerdem wird die Referenzspannung umgeschaltet. Die Software benötigt zur Bestimmung der Akkuspannung eine konstante Referenzspannung von 1,1 Volt, die der Mikrocontroller intern bereitstellt.
- **Timerkonfiguration:** Ein Timer des Mikrocontroller wird zur Erkennung von DCC-Daten verwendet. Dafür wird ein Takt von 70 Mikrosekunden und eine Unterbrechungsroutine implementiert.
- **Aktivierung der seriellen Kommunikation:** Über die serielle Schnittstelle ist es möglich, Konfigurationswerte aus dem Fahrzeugdekker auszulesen. Die serielle Schnittstelle muss für eine korrekte Datenübertragung mit einer Geschwindigkeit von 9600 Baud betrieben werden.
- **Sleep-Einstellungen:** In dieser Phase werden der Watchdog-Timer und der Sleep-Modus des Mikrocontrollers konfiguriert. Dabei erhält der Watchdog-Timer eine größtmögliche Unterbrechungszeit von acht Sekunden. Dieser Timer ist dafür verantwortlich den Controller aus dem Sleep-Modus aufzuwecken. Dazu wird regelmäßig geprüft ob DCC-Daten über den IR-Empfänger erhalten werden.

2. **Programmschleife:** Hier erfolgt die Betriebssteuerung aller angeschlossenen Komponenten. Dazu wird alle  $10\text{ms}$  eine Funktion durchlaufen welche die Ausgänge setzt.

- **Setzen der Ausgänge:** In dieser Funktion werden alle Ausgänge des Mikrocontrollers je nach deren Programmierung gesetzt. Sie beinhaltet einen Zähler, der bei jeder Ausführung erhöht wird. An diesem Zähler orientieren sich alle zeitabhängigen Funktionen. Diese Abläufe sind unter Kapitel 5.2.4 genauer erklärt.

- **Serielle Kommunikation:** Diese Funktion wird aktiviert durch eine schreibende Programmierung der Konfigurationsvariable zwölf. Danach können alle Registerwerte über die serielle Schnittstelle ausgelesen werden. Nach einer Aktivierung, kann der Dekoder nicht mehr über den IR-Empfänger gesteuert werden. Dadurch wird ein einfaches Debuggen des Fahrzeugdekoders mit einem Computer möglich. Zur Deaktivierung kann die Versorgung des Fahrzeugdekoders einfach abgeschaltet werden oder über einen seriellen Befehl in den normalen Operation-Mode gewechselt werden.

3. **Timer:** Es gibt drei Hardware-Timer. Timer0 wird für die Verzögerungen im Programmablauf verwendet. Der Timer1 ist für die Erkennung der DCC-Befehle erforderlich. Mit dem Timer2 wird der programmierbare Tonausgang für eine Sirene gesteuert.

Aufgrund der vielen möglichen Funktionen ergaben sich bei der Entwicklung der Software einige Schwierigkeiten. Ein wesentliches Problem ist die Taktbeschränkung auf nur 8 MHz. Da einige Funktionen wie die DCC-Datenerkennung und die PWM-Ansteuerung mancher Ausgänge zu exakten Zeitpunkten ausgeführt werden müssen, ergeben sich Zeitprobleme. Der dabei entstehende Jitter, durch die Interrupt-Unterbrechung des Programms für zeitrelevante Programmfunctionen, darf den normalen Programmablauf nicht zu stark beeinträchtigen. Je nach Programmfunction können die Auswirkungen aber vernachlässigbar sein, beispielsweise eine nur in Mikrosekunden messbare Veränderung der Blinkfrequenz.

Die Software wurde extra für die Taktgeschwindigkeit von 8 MHz angepasst. Fahrtests brachten die Erkenntnis, dass die Software auch mit einer geringeren Taktfrequenz des Fahrzeugdekoders zuverlässig arbeitet. Durch weitere Fahrtests auf den Straßen einer Testanlage konnten viele Softwaremängel gefunden und behoben werden, zum Beispiel das anfängliche ruckartige Verhalten der Fahrzeuge während der Fahrt.

### 5.2.1 Energieeffizienz

Mit der aktuellen Software kann ein Schalter zum Ein-/Ausschalten der Fahrzeuge entfallen. Die Software besitzt verschiedene Energiesparmechanismen. Im normalen Leerlaufbetrieb ohne eingeschaltenen Motor oder LEDs fließen etwa 11,5 mA Strom. Die Abschaltung des Controllers erfolgt, wenn keine DCC-Daten erkannt werden, in zwei Phasen.

In der ersten Phase nach etwa 20 Sekunden<sup>1</sup> schaltet der Mikrocontroller in den Sleep-Modus. Dann wird von dem im Leerlaufbetrieb fließenden Strom 90% eingespart. Der Strom von noch etwa 1,15 mA wird größtenteils vom IR-Empfänger verbraucht. Wenn DCC-Daten erkannt werden, wird der Controller sofort über eine Interrupt-Unterbrechung aufgeweckt und arbeitet normal weiter.

In einer zweiten Phase, die etwa 32 Sekunden<sup>2</sup> nach der Ersten beginnt, schaltet sich der Dekoder nahezu vollständig ab. Er benötigt dann nur noch 0,03 mA, was einer

---

<sup>1</sup>einstellbar über CV#54

<sup>2</sup>einstellbar über CV#55

Einsparung von circa 99,6% entspricht. Für eine Einsparung von 99,6% ist eine angepasste Hardware notwendig. Diese ermöglicht eine zusätzliche Abschaltung des IR-Empfängers. Um ein Aufwecken des Mikrocontrollers durch ankommende DCC-Daten zu ermöglichen läuft auf dem Controller ein Watchdog Timer. Der Watchdog Timer ist ein spezieller Hardware-Zeitgeber. Durch ihn wird eine Unterbrechung nach maximal acht Sekunden ausgelöst. Der Mikrocontroller wird dadurch aufgeweckt. Für eine Zeit von 100 Millisekunden wird der IR-Empfänger eingeschaltet und geprüft ob ein korrektes DCC-Daten-Paket empfangen wurde. So dauert ein Wiedereinschalten des Dekoder aus der zweiten Phase bis zu 10 Sekunden. Ein Ein-/Ausschalter ist bei dieser Hardwieranpassung nicht mehr notwendig. Ein Fahrzeug mit einem 80 mAH Akku kann bei einer Akkubelasung von etwa 0,04 mA etwa 145 Tage eingeschaltet bleiben bis der Akku entladen ist. In dieser Zeit würde auch die Selbstentladung des Akkus beginnen und allein deshalb eine Neuaufladung vor Betriebsbeginn erforderlich sein.

### 5.2.2 Dekodierung von IR-Signalen

Für die Erkennung von DCC-Befehlen ist eine Interrupt auf dem Eingangspin notwendig. Bei einer steigenden Flanke wird ein Interrupt ausgelöst. Dieser startet dann den Timer1 des Mikrocontrolles. Der Timer läuft dann für eine fest eingestellte Zeit von 75 Mikrosekunden. Nach Ablauf dieser Zeit wird der Timer1 wieder gestoppt. Es kann nun ermittelt werden, ob am Empfänger-Eingang ein Eins- oder Null-Bit anliegt. Wenn der Eingang weiterhin ein positives Potential hat, dann wird wie im DCC-Protokoll festgelegt ein Null-Bit erkannt. Ist auf dem Eingang nach Ablauf der 75 Mikrosekunden ein negatives Potential, wird ein Eins-Bit erkannt. Während der Dekoder aktiv ist wird so der gesamte empfangende Bitstrom mitgelesen. Anhand fester Vorgaben aus dem Telegramm können die übermittelten Bitfolgen zu einem Befehl zusammengesetzt werden (siehe Kapitel 2).

### 5.2.3 Befehlsauswertung

Die am Eingang ermittelte Bitfolge wird auf die Präambel hin untersucht. Dazu wird innerhalb des Timer1 jedes Eins-Bit gezählt. Bei einem erkannten Null-Bit wird die Anzahl der Eins-Bit geprüft. Wurden mehr als zehn Eins-Bit erkannt, folgen jetzt immer byte-weise die DCC-Datenpakete. Diese werden in einem Array (Feld) bitweise hinterlegt. Um das Datenende zu ermitteln, wird nach 8 Bit das nun folgende Start- oder Stop-Bit ausgewertet. Ist dies ein Null-Bit, folgt ein weiteres Datenpaket mit einem Byte Daten. Wenn hier ein Eins-Bit erkannt wurde, dann ist dies als Paket-Stop-Bit zu werten. Nachdem das Stop-Bit erkannt wurde, werden keine weiteren Bit in dem Array gespeichert und die DCC-Auswertungsfunktion in der Software wird gestartet.

In der Auswertefunktion wird als erstes die CRC-Prüfsumme des Datenpaketes ermittelt. Da die Pakete eine variable Datenlänge besitzen, werden immer nur die letzten 8 Bit betrachtet. Durch die Bestimmung eines XOR über die Adress- und Datenbytes kann hier die Korrektheit der Daten ermittelten werden. Sind die Daten alle korrekt empfangen, werden sie ausgewertet. Dazu wird als erstes die Adresse des Datenpaketes

bestimmt. Stimmt diese mit derjenigen des Dekoders überein, wird der Inhalt der Datenbytes dekodiert und in globale Variablen abgelegt. In der Programmschleife werden diese Variablen alle zehn Millisekunden abgefragt. Je nach Inhalt werden die Ausgänge, wie im nächsten Abschnitt erklärt, angesteuert.

### 5.2.4 Ansteuerung der Hardware

Die Ansteuerung der Hardware erfolgt entsprechend den am Controller angeschlossenen Komponenten. Für die Erzeugung von Blaulichtsequenzen wurde eine zusätzliche Funktion im Quellcode implementiert, da hier der Rechenaufwand wesentlich höher ist als für die Berechnung des Blinker oder anderer Lichtansteuerungen. Bestimmte Ausgänge sind fest für Funktionsbefehle konfiguriert und bei anderen lässt sich die zugehörige Funktion auswählen. So lässt sich das Licht über die Funktion F0 und der Blinker über F2 und F3 schalten. Die Lichtautomatik lässt sich über F3 aktivieren. Für die Hardwaeansteuerung ist die Software wieder in mehrere, periodisch abzuarbeitende Aufgaben unterteilt:

#### 1. DCC Prüfung:

- Prüfen, ob ausreichend oft korrekte DCC-Daten empfangen wurden. Falls nicht, wird ein laufender Motor abgeschaltet und die Warnblinkanlage aktiviert.
- Ermitteln, ob regelmäßig DCC-Daten empfangen werden. Wenn eine gewisse Zeit keine DCC-Daten erkannt wurden, wird der Mikrocontroller in den Sleep-Modus versetzt.

#### 2. Basisfunktion Ansteuerung:

- Hier wird der Blinker gesteuert. Er kann über die Funktionen F1 (links) und F2 (rechts) ansprechen. Dazu wird zusätzlich geprüft, ob die Warnblinkanlage aktiv ist (F1 und F2). Dann werden die beiden Blinkerausgänge synchronisiert. Falls die Blinker vertauscht angeschlossen wurden, sind lässt sich über die CV#68 eine Invertierung aktivieren.
- Für die Ansteuerung des Abblendlichts ist die Funktion F0 vorgesehen. Das Abblendlicht kann in Abhängigkeit des Helligkeitssensors aktiviert werden. Dazu lässt sich der Helligkeitssensor mittels der Funktion (F3) einschalten. Wenn dieser eingeschaltet ist, wird anhand von Schwellwerten und Verzögerungszeiten, die programmierbar sind, dass Licht ein- und ausgeschaltet.
- Für die Motoransteuerung können Verzögerungen festgelegt werden. Zusätzlich wird über die aktuelle Motordrehzahl und die von der Digitalzentrale gesendete Fahrstufe das Ein- und Ausschalten des Bremslichtes berechnet.
- Hier werden die Lichtausgänge mit den oben ermittelten Daten aus Punkt 2 sowie das Bremslicht und Abblendlicht gesetzt. Dabei wird das Rücklicht, um Hardware einzusparen über ein gedimmtes Bremslicht realisiert.

- Die für die Motoransteuerung bestimmten Daten aus Punkt 2. werden hier normiert auf minimale und maximale Drehzahl beschränkt und auf den Motorausgang geschaltet.

### 3. Programmierbare Ausgänge:

- Für sechs Ausgänge ist die Funktion nicht definiert. Sie können je nach Verwendung frei programmiert werden. Über die CV-Programmierung des Fahrzeugdekoders, kann das Ausgangsverhalten dieser Ausgänge bestimmt werden. Dazu stehen für jeden Ausgang vier verschiedene Ausgangsmodi zur Auswahl:
  - Ein/Aus, für Zusatzlicht oder ähnliche Funktionen.
  - Einfacher Blitz.
  - Doppelter Blitz.
  - Dreifachblitz, wird bei neuen Einsatzfahrzeugen oft als Front oder Dachblitzer verwendet.
  - Sirene oder Martinshorn mit zusätzlicher Funktionszuweisung.
- Ein weiterer Ausgang ist fest auf Rundumlicht eingestellt. Dieses kann zum Beispiel für Müllfahrzeuge oder ältere Einsatzfahrzeuge genutzt werden. Auch hier kann mittels Programmierung die Blinkgeschwindigkeit geändert werden. Jedem Ausgang lässt außerdem über die CV-Programmierung eine Funktion von F4 bis F11 zuweisen.

#### 5.2.5 Programmierung

Viele Verhaltenseinstellungen des Fahrzeugdekoders lassen sich über eine Programmierung beeinflussen. Dazu können im Service Mode (siehe Abschnitt 2.1.4) der Digitalzentrale, in die Register des Fahrzeugdekoders, nutzerdefinierte Einstellungen abgelegt werden.

#### Konfigurationsvariablen

Über die Registerwerte lassen sich die Fahrzeugdekoder speziell auf das jeweilige Fahrzeug anpassen. So kann zum Beispiel über das Register CV#1 die Adresse festgelegt werden. Über die Register CV#2 bis CV#5 lässt sich das Motorverhalten einstellen. Mit den Registerwerten CV#59 bis CV#88 lassen sich Blinkfrequenzen und Funktionstasten für die programmierbaren Ausgängen festlegen. Um zum Beispiel alle Registerwerte wieder zurückzustellen muss nur ein beliebiger Wert in das Register CV#8 geschrieben werden. Eine erfolgreiche Programmierung wird von dem Dekoder mit einem dreifachen Aufblitzen des Abblendlichts bestätigt.

Da die Fahrzeugdekoder ihre aktuellen Registerwerte nicht, wie dies bei Lokomotiven möglich ist an die Digitalzentrale zurückmelden können, wird dazu die serielle Schnittstelle benutzt. Mit dieser können alle Register ausgelesen und auch neue Werte geschrieben werden. Damit eine Kommunikation mit dem Dekoder über die serielle

Schnittstelle möglich ist, muss vorher mit einer Digitalzentrale ein beliebiger Wert in das Register CV#12 geladen werden. Nun kann der Dekoder nur noch über einen Computer gesteuert werden. Alle anderen Funktionen sind bis zur Beendigung oder Abschaltung des Dekoders inaktiv. Eine Aktivierung der seriellen Schnittstelle signalisiert der Fahrzeugdekoder durch ein vierfach blinkendes Abblendlicht, die Deaktivierung mit dem Befehl “exit;“ über einen Doppelblitz.

Der Vorteil einer dekoderbezogenen Programmierung ist es, dass unabhängig von der eingespielten Software alle Einstellungen des Fahrzeugs erhalten bleiben und jedesmal beim Einschalten auch erneut geladen werden. So bleibt zum Beispiel auch nach einem Software-Update die Steuerungsadresse des Modellfahrzeugs noch dieselbe.

Im Anhang B dieser Arbeit ist eine Auflistung aller programmierbaren Register des Fahrzeugdecoders aufgeführt.

# Kapitel 6

## Zusammenfassung

Im Rahmen der Bachelorarbeit wurde ein System zur Steuerung von Miniaturmodellfahrzeugen zu entwerfen. Dazu wurde ein geeignetes Verfahren für die IR-Datenübertragung zur Steuerung der Modellfahrzeuge ausgewählt. Unter der Nutzung bestehender Code-Bibliotheken für die Mikrocontroller-Plattform Arduino wurde eine Software für Modellfahrzeuge entworfen. Die Steuerung der Fahrzeuge erfolgt dabei über ein für Modelleisenbahnen übliches Übertragungsprotokoll. Es ist das häufig verwendete DCC-Protokoll.

Alle Softwarekomponenten wurden in einer speziell angepassten miniaturisierten Versuchsanordnung zusammen mit den Hardwarekomponenten getestet. Zusätzlich wurden mit den umgebauten Modellfahrzeugen Fahrversuche unternommen. In die üblichen stehenden Fahrzeuge einer Modelleisenbahn, im Maßstab 1:120 wurden Akkus, Fahrzeugdekoder, Motor und Beleuchtung eingebaut. Für den Dekoder musste eine Platine mit besonders schmalen Leiterzügen für den SMD-Mikrocontroller angefertigt werden.

Die vorliegende Arbeit stellt dabei nur einen Teil der Komponenten dar, die auf einer Modelleisenbahnanlage mit einem automatischen Faller Car-System benötigt werden. Ein komplett automatisiertes Steuerungssystem enthält zusätzlich Rückmelder in der Fahrbahn. Auf diese Weise ist es möglich, Modellfahrzeuge mit einem Computer zu lokalisieren und automatisch mit vorher programmierten Abläufe oder zufällig zu steuern. Für die Datenrückmeldungen ist ein weiteres Übertragungssystem erforderlich.

Durch die Installation eines handelsüblichen Funk-Empfängermoduls in den Modellfahrzeugen kann eine bidirektionale Datenübertragung erfolgen. Dadurch ist es möglich, eine zuverlässigere Steuerung und Rückmeldung von fahrzeugspezifischen Daten zu realisieren. Kleine Funkmodule, die nur halb so groß sind wie der hier entwickelte Fahrzeugdekoder, können die IR-Steuerung ersetzen oder zusätzlich zur IR-Datenübertragung verwendet werden. Bei Problemen mit der Kommunikation über Infrarot können die Daten auch vom Funkempfänger verarbeitet werden. Außerdem besteht die Möglichkeit, Daten aus den Fahrzeugen im Betrieb an den Steuerungscomputer weiterzugeben. So können bei Absinken der Akkusspannung unter einem bestimmten Schwellwert – die Modellfahrzeuge automatisch in eine Ladestation fahren. Für ein Software-Update ist in diesem Fall außerdem keine Kabelbuchse für eine Verbindung zu einem Computer notwendig.



# Abbildungsverzeichnis

2.1	Aufbau eines DCC-Signals [11]. . . . .	4
2.2	Aufbau eines DCC-Basis-Datenpakets mit drei Daten-Bytes [11]. Übertragen wurde hier ein Befehl für die Adresse 55, mit der Fahrstufe elf als Vorwärtsfahrt. . . . .	6
3.1	Ansicht von Faller Car-System Fahrzeugen [5]. . . . .	14
3.2	Lenkachse mit Permanentmagnet auf Magnetband (links) und Fahrdräht (rechts). . . . .	14
3.3	Digitale Fahrzeugsteuerung und seine Komponenten. . . . .	17
3.4	Blockbild des Funktionsaufbau. . . . .	18
4.1	Oberer Teil des DCC-Signal. . . . .	21
4.2	Eingangsfrequenz des TSOP 7000 [17]. . . . .	22
4.3	Spektrale Empfindlichkeit der Empfängerdiode des TSOP7000 [17]. . .	23
4.4	DCC-Signal mit Trägerfrequenz von 455 kHz. . . . .	25
A.1	Schaltung IR-Fahrzeugdekoder. . . . .	41
A.2	Schaltung DCC Infrarot Sender. . . . .	42



# Literaturverzeichnis

- [1] ARDUINO TEAM: *Arduino Uno*. <http://www.arduino.cc/en/Main/arduinoBoardUno>, Stand: 20.11. 2011.
- [2] BREDENDIEK, JÖRG: *IR-Fernbedienungen - der RC-5 Code*. <http://www.sprut.de/electronic/ir/rc5.htm>, Stand: 09.12 . 2011.
- [3] BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG: *Die Erforschung der menschlichen Sinne*. [http://www.gesundheitsforschung-bmbf.de/\\_media/Sinnesforschung\\_doppelseitig.pdf](http://www.gesundheitsforschung-bmbf.de/_media/Sinnesforschung_doppelseitig.pdf), Stand: 20.12. 2011.
- [4] DIPL.-ING. SVEN BRANDT: *Digital-Bahn*. [http://www.digital-bahn.de/info\\_tech/dcc.htm](http://www.digital-bahn.de/info_tech/dcc.htm), Stand: 02.10. 2011.
- [5] GEBR. FALLER GMBH: *Faller*. <http://faller.de/App/WebObjects/XSeMIPS.woa/cms/page/pid.14.17.109/ecm.p/Car-System.html>, Stand: 14.11. 2011.
- [6] GEBR. FALLER GMBH: *Neuheiten 2011, Seite 3*. [http://www.faller.de/xs\\_db/DOKUMENT\\_DB/www/allgemeines/FALLER\\_neu2011\\_mitPreis.pdf](http://www.faller.de/xs_db/DOKUMENT_DB/www/allgemeines/FALLER_neu2011_mitPreis.pdf), Stand: 14.11. 2011.
- [7] HOLLANDT, JÖRG: *Infrarotstrahlung*. <http://www.weltderphysik.de/de/7019.php>, Stand: 09.12. 2011.
- [8] IR RECTIFIER: *IRLML2402 HEXFET Power MOSFET*. <http://www.irf.com/product-info/datasheets/data/irlml2402.pdf>, Stand: 23.11. 2011.
- [9] LINDECKE, CHRISTIAN: *Lokodex*. [http://www.lokodex.de/mo/m\\_digital\\_dccprot01.htm](http://www.lokodex.de/mo/m_digital_dccprot01.htm), Stand: 12.10. 2011.
- [10] MOROP (VERBAND DER MODELLEISENBAHNER UND EISENBAHFREUNDE EUROPAS): *Normen Europäischer Modellbahnen*. <http://www.morop.org/de/normes/index.html>, Stand: 05.10. 2011.
- [11] MOROP (VERBAND DER MODELLEISENBAHNER UND EISENBAHFREUNDE EUROPAS): *Normen Europäischer Modellbahnen: Digitales Steuersignal DCC Bitdarstellung*. [http://www.morop.org/de/normes/nem670\\_d.pdf](http://www.morop.org/de/normes/nem670_d.pdf), Stand: 05.10. 2011.

- [12] MOROP (VERBAND DER MODELLEISENBAHNER UND EISENBAHNFREUNDE EUROPAS): *Normen Europäischer Modellbahnen: Digitales Steuersignal DCC Datenpaket.* [http://www.morop.org/de/normes/nem671\\_d.pdf](http://www.morop.org/de/normes/nem671_d.pdf), Stand: 05.10. 2011.
- [13] NATIONAL MODEL RAILROAD ASSOCIATION INC.: *DCC.* <http://www.nmra.org/standards/DCC/>, Stand: 05.12. 2011.
- [14] NATIONAL MODEL RAILROAD ASSOCIATION INC.: *NMRA RP 9.2.3 (01-2006).* [http://www.nmra.org/standards/DCC/standards\\_rps/RP-923.pdf](http://www.nmra.org/standards/DCC/standards_rps/RP-923.pdf), Stand: 05.10. 2011.
- [15] NATIONAL MODEL RAILROAD ASSOCIATION INC.: *NMRA S-9.2 (07-2004).* [http://www.nmra.org/standards/DCC/standards\\_rps/S-92-2004-07.pdf](http://www.nmra.org/standards/DCC/standards_rps/S-92-2004-07.pdf), Stand: 05.10. 2011.
- [16] NATIONAL MODEL RAILROAD ASSOCIATION INC.: *nmra.org.* [http://www.nmra.org/standards/DCC/standards\\_rps/DCCStds.html](http://www.nmra.org/standards/DCC/standards_rps/DCCStds.html), Stand: 20.10. 2011.
- [17] VISHAY SEMICONDUCTORS: *TSOP7000 IR Receiver for High Data Rate PCM at 455 kHz.* <http://www.hobbyengineering.com/specs/Vishay-TSOP7000.pdf>, Stand: 18.11. 2011.

# Anhang A

## Schaltungen

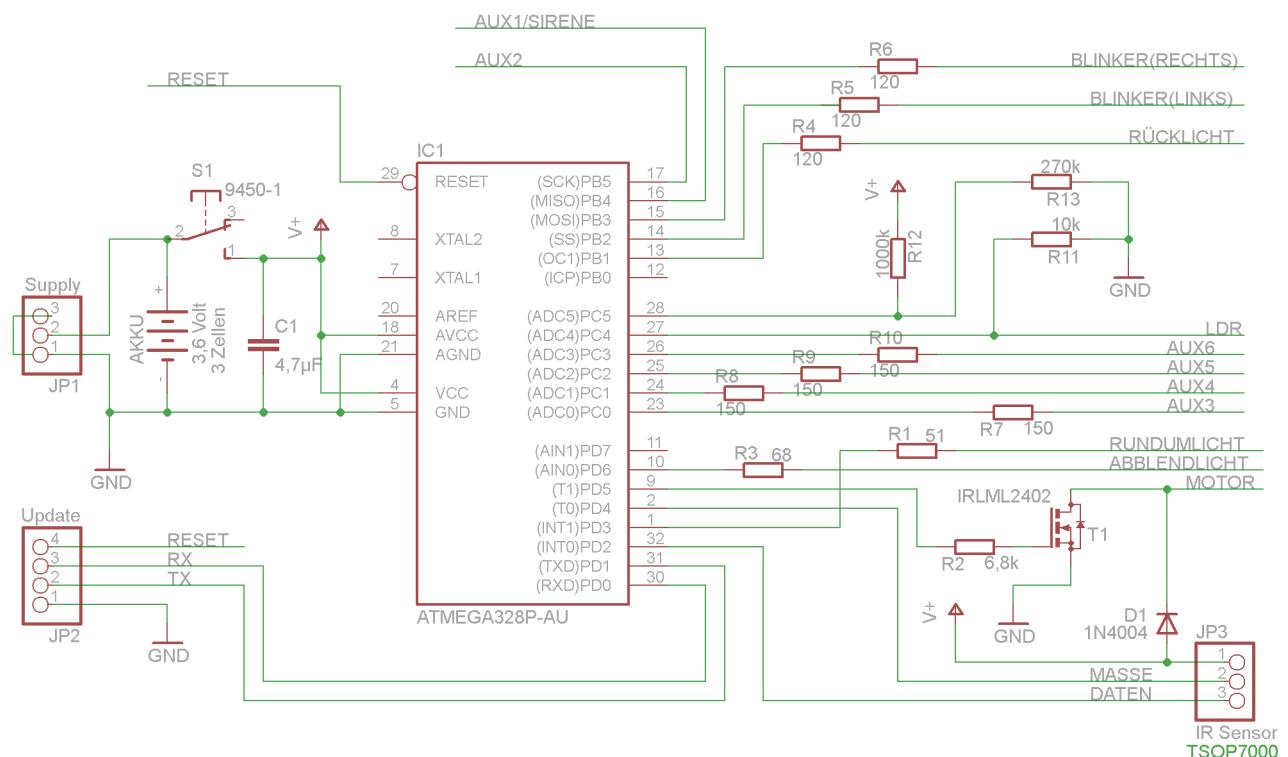


Abbildung A.1: Schaltung IR-Fahrzeugdekoder.

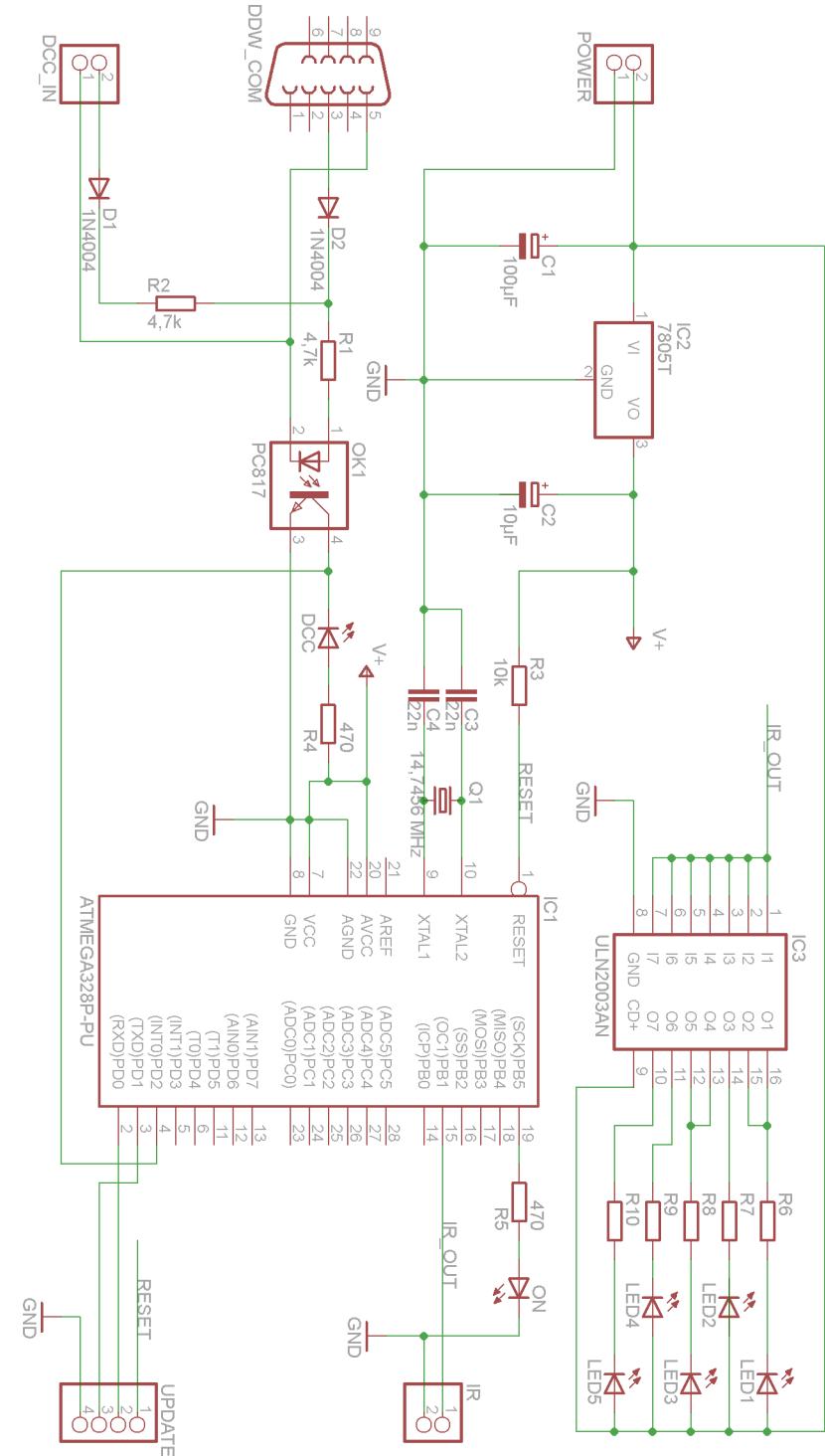


Abbildung A.2: Schaltung DCC Infrarot Sender.

## Anhang B

# Konfigurationsvariablen des Fahrzeugdekoder

In der Tabelle B sind alle Konfigurationsvariablen aufgeführt. Die Spalte "CV.Nr." enthält die Nummer der Konfigurationsvariable. Die Defaultwerte sind die Werte die nach einem Reset oder für die allererste Inbetriebnahme eingestellt werden.

Name der CV	CV-Nr.	Eingabewert (Defaultwert)	Erläuterungen und Hinweise
Basisadresse	1	1...111 (3)	Legt die Adresse fest auf welche der Dekoder reagiert. Für größere Adressen muss eine lange Adresse mit CV#17 und CV#18 programmiert werden.
Startspannung	2	0...255 (5)	Spannung, die bei Fahrstufe 1 an den Motor ausgegeben wird. Ein Wert "0" entspricht 0 Volt, der Wert "255" der max. Spannung.
Beschleunigungsrate	3	0...255 (18)	Länge der Wartezeit, die beim Beschleunigen des Fahrzeug jeweils vor dem Hochschalten zur nächst höheren Fahrstufe vergeht. Die Wartezeit wird wie folgt berechnet: $(CV\#3) \cdot 0,01sec$ .
Bremsrate	4	0...255 (5)	Länge der Wartezeit, die beim Abbremsen des Fahrzeug jeweils vor dem Herunterschalten zur nächst niedrigeren Fahrstufe vergeht. Die Wartezeit wird wie unter CV#3 beschrieben berechnet.

44 ANHANG B. KONFIGURATIONSVARIABLEN DES FAHRZEUGDEKODER

Maximalspannung	5	0...255 (255)	Spannung, die bei der höchsten Fahrstufe an den Motor ausgegeben wird. Der Wert “255“ entspricht 100% der maximalen Spannung.
Version	7	—	Nur seriell auslesbar.
Hersteller	8	—	Nur seriell auslesbar.
Reset	8	0...255	Beim Schreiben eines beliebigen Wertes werden die Einstellungen im Auslieferungszustand wieder hergestellt.
Serial	12	0...255	Beim Schreiben eines beliebigen Wertes wird die DCC Erkennung und der “Sleep Modus“ deaktiviert. Über die serielle Schnittstelle lassen sich nun alle CV Werte auslesen und ändern.
Erweiterte Adresse	17 18	192...231 0...255	Die erweiterte Adresse berechnet sich aus diesen zwei Registern. Beim schreiben einer der beiden Register wird die (CV#1) gelöscht.
Akkuwarnung	28	0...255 (178)	Steuert das Einschaltung des Warnblinker unterhalb der eingestellten Spannung. Zur Berechnung der Spannung: $Volt = \frac{(CV\#28)\cdot 2}{100}$
Blinkerferquenz	29	0...255 (40)	Länge die der Blinker ein/aus geschaltet ist. Die Zeit wird wie folgt berechnet: $(CV\#29) \cdot 0,01sec$ .
Motor Notstop	41	0...255 (50)	Länge der Wartezeit ohne erkannte DCC Daten bis der Motor gestoppt wird. Die Wartezeit wird wie folgt berechnet: $(CV\#41) \cdot 2 \cdot 0,01sec$ .
Licht ein	42	0...255 (105)	Umgebungshelligkeit die vom Dekoder gemessen wurde. Ein Wert von “255“ entspricht hell und “0“ dunkel.
Licht ein Verzögerung	43	0...255 (100)	Länge der Wartezeit bis das Licht beim unterschreiten von (CV#42) eingeschaltet wird. Die Zeit wird wie folgt berechnet: $(CV\#43) \cdot 0,01sec$ .

Licht aus	44	0...255 (145)	Umgebungshelligkeit die vom Dekoder gemessen wurde. Ein Wert von "255" entspricht hell und "0" dunkel.
Licht aus Verzögerung	45	0...255 (200)	Länge der Wartezeit bis das Licht beim überschreiten von (CV#44) abgeschaltet wird. Die Wartezeit wird wie unter CV#43 berechnet.
Bremslicht ein	46	0...255 (30)	Gibt die Fahrstufe an, unterhalb welcher das Bremslicht beim Herunterschalten sofort eingeschaltet wird.
Bremslicht ein stark	47	0...255 (4)	Bei einer starken Reduzierung der Fahrgeschwindigkeit wird das Bremslicht eingeschaltet. Die Stärke der Reduzierung berechnet sich wie folgt: $Fahrstufe + \frac{Fahrstufe}{(CV\#47)} < aktueller\_Fahrstufe$ .
Bremslicht Stand	48	0...255 (255)	Länge der Zeit, die das Bremslicht im Stand nachleuchtet. Die Zeit wird wie folgt berechnet: $(CV\#41) \cdot 0,01sec$ .
Bremslicht Fahrt	49	0...255 (80)	Länge der Zeit, die das Bremslicht nach dem Herunterschalten von Fahrstufen während der Fahrt angeschaltet bleibt. Die Zeit wird wie unter unter CV#44 berechnet.
Dimmen der Ausgänge: Rücklicht Bremslicht Blinker Abblendlicht	50 51 52 53	0...255 (70) 0...255 (255) 0...255 (255) 0...255 (110)	Reduzierung der Spannung, die am Ausgang anliegt. Der Wert "1" entspricht der kleinsten, "255" der maximalen Spannung.
Power Save	54	0...255 (20)	Wartezeit ohne DCC-Daten bis der Dekoder alle Ausgänge abschaltet. Ein Aufwecken ist sofort wieder möglich. Die Wartezeit wird für CV#54 in Sekunden angegeben.
Power Down	55	0...255 (4)	Wartezeit nach CV#54 bis der Dekoder die Hardware komplett abschaltet. Ein Aufwecken aus dem "Full Sleep" kann bis zu 8sec. dauern. Die Wartezeit berechnet sich wie folgt: $(CV\#55) \cdot 8sec$ .

## 46 ANHANG B. KONFIGURATIONSVARIABLEN DES FAHRZEUGDEKODER

Funktion für Ausgang: Sirene-Zusatzfunktion	59	0...255 (1)	Die Funktion auf welche der jeweilige Ausgang reagiert. F4 = 1
Rundumlicht	60	0...255 (1)	F5 = 2
AUX1 / Sirene	61	0...255 (1)	F6 = 4
AUX2	62	0...255 (1)	F7 = 8
AUX3	63	0...255 (1)	F8 = 16
AUX4	64	0...255 (1)	F9 = 32
AUX5	65	0...255 (1)	F10 = 64
AUX6	66	0...255 (1)	F11 = 128
Effekte <sup>1</sup> der Ausgänge AUX1 bis AUX4	67	0...255 (120)	AUX1 = (1,2) AUX2 = (4,8) AUX3 = (16,32) AUX4 = (64,128)
Effekte <sup>3</sup> der Ausgänge AUX5 und AUX6, Invertierung Blinker und Deaktivierung Akkuwarnung	68	0...255 (7)	AUX5 = (1,2) AUX6 = (4,8) Blinker inv. + 16 Akkuwarnung off + 32
Frequenz der Sirene	69	1...4 (1)	Wechsel zwischen: 440Hz/585Hz (je 0,9 sec) = “1“ 466Hz/622Hz (je 0,9 sec) = “2“ 435Hz/580Hz (je 1 sec) = “3“ 450Hz/600Hz (je 1 sec) = “4“ Hupe 400Hz = “8“
Rundumlichtfrequenz	70	0...255 (40)	Pause zwischen dem Aufleuchten. Die Zeit berechnet sich wie folgt: $(CV\#70) \cdot 0,01sec.$
Blitzlänge AUX1	71	0...255 (1)	Länge der Einschaltzeit des Ausgangs. Die Zeit berechnet sich wie in CV#70 beschrieben.
Blitzlänge AUX2	72	0...255 (3)	
Blitzlänge AUX3	73	0...255 (2)	
Blitzlänge AUX4	74	0...255 (2)	
Blitzlänge AUX5	75	0...255 (3)	
Blitzlänge AUX6	76	0...255 (20)	

<sup>1</sup>Für alle außer AUX1 gilt: (0,0) = kein Blinken, (0,1) = Einfachblitz, (1,0) = Doppelblitz, (1,1) = Dreifachblitz. Bei AUX1: (0,0) = Sirene, (0,1) = kein Blinken, (1,0) = Einfachblitz, (1,1) = Doppelblitz. Wo eine “1“ steht muss die entsprechende Zahl von oben aufaddiert werden damit die gewünschte Art der Funktion am Ausgang anliegt.

Blitzpause1 AUX1	77	0...255 (6)	Länge der Wartezeit bis zum Wiedereinschalten des Ausgangs nach dem ersten Einschalten. Die Zeit berechnet sich wie in CV#70 beschrieben.
Blitzpause1 AUX2	78	0...255 (6)	
Blitzpause1 AUX3	79	0...255 (5)	
Blitzpause1 AUX4	80	0...255 (15)	
Blitzpause1 AUX5	81	0...255 (6)	
Blitzpause1 AUX6	82	0...255 (20)	
Blitzpause2 AUX1	83	0...255 (43)	Länge der Wartezeit bis zum Wiedereinschalten des Ausgangs nach dem zweiten und/dern dritten Einschalten. Die Zeit berechnet sich wie in CV#70 beschrieben.
Blitzpause2 AUX2	84	0...255 (40)	
Blitzpause2 AUX3	85	0...255 (30)	
Blitzpause2 AUX4	86	0...255 (30)	
Blitzpause2 AUX5	87	0...255 (43)	
Blitzpause2 AUX6	88	0...255 (0)	