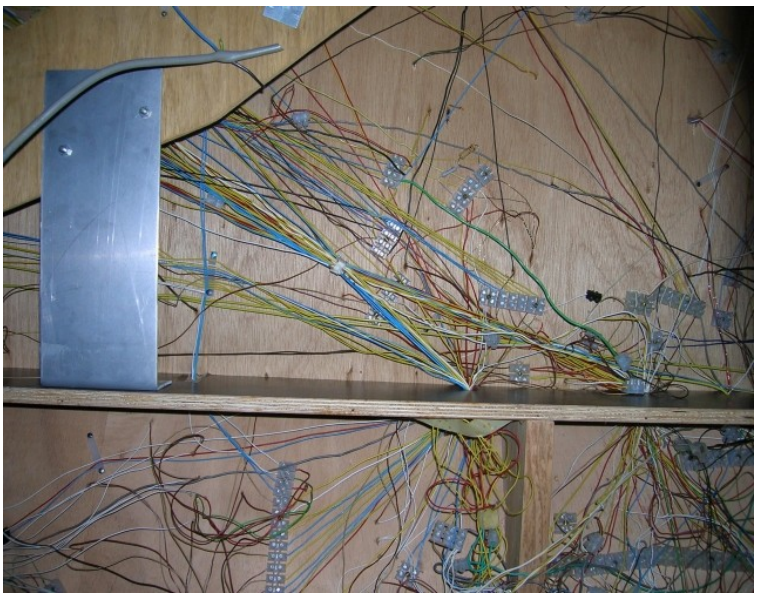
# Modell(-bau-)getriebene Eisenbahnsteuerung

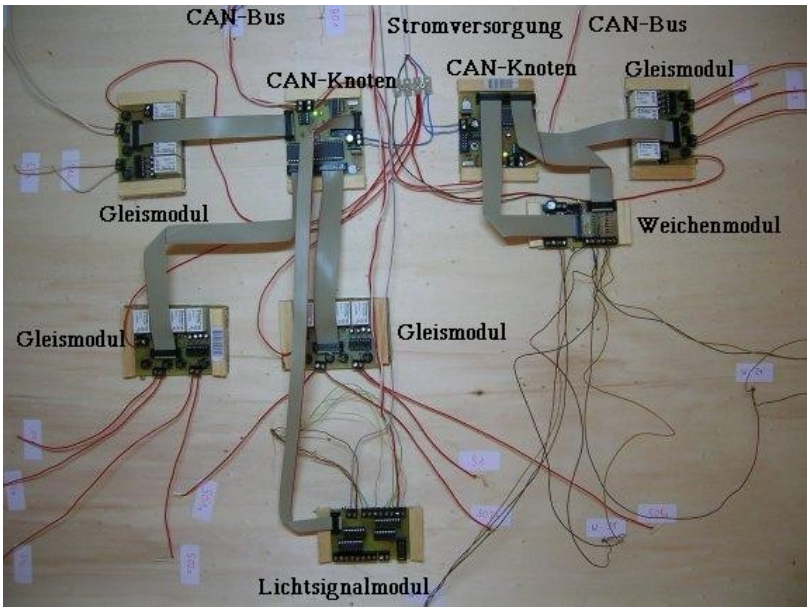
## *Motivation*

Der Bau einer Modelleisenbahnanlage ist nicht nur technisch eine Herausforderung. Viele Kabel müssen über Kopf verlegt werden. Zum Teil ist die Topologie der Anlage so, dass viele Stellen relativ unzugänglich sind. Kann dort Verkabelung reduziert werden, vereinfacht das im erheblichen Maße die Installation. Aus diesem Grunde wurde die Steuerung so konzipiert, dass möglichst viel



Verkabelungsaufwand gespart werden soll. Bei einer klassischen Anlage werden alle zu steuernden Bauteile mit dem Stellwerk verbunden. Eine simple Weiche hat drei Anschlüsse. Bei 30 Weichen sind das schon knapp hundert Kabel, die zum Teil über weite Strecken geführt werden müssen. Aus diesem Grund ist die Steuerung der Bauteile möglichst nah anzubringen. Es verbleibt nur noch die Stromversorgung und die Kommunikationsleitungen, die anlagenweit anzubringen sind. Die Steuermodule – im Folgenden CAN-Knoten genannt - sind nicht spezialisiert, das heißt, dass sie über ihre Mehrzweck-IO-Leitungen alle möglichen Erweiterungen ansteuern können. Die Gerätemodule haben keine Eigenintelligenz und dienen als Schnittstelle und Leistungsverstärker bzw. Schalter zwischen dem CAN-Knoten und dem Bauteil. Die Bauteile der Eisenbahnanlage werden direkt an die Gerätemodule angeschlossen und diese wiederum per Flachbandkabel an die Mehrzweckanschlüsse der CAN-Knoten. Jeder CAN-Knoten hat drei dieser Anschlüsse. Als zusätzliche Anschlussmöglichkeit hat jeder CAN-Knoten einen seriellen Anschluss, mit dem ein spezielles Lichtsignalmodul über ein 10-poliges Flachbandkabel angeschlossen werden kann. Die reine Verkabelung meiner eigenen Anlage mit 29 Weichen, 84 Gleisabschnitten, 45 Signalen gesteuert von 15 CAN-Knoten hat gerade mal fünf volle Tage gedauert.

Als Bussystem wurde CAN gewählt. Ursprünglich von Bosch entwickelt, wird es in fast jedem neueren Fahrzeug als Bussystem verwendet. Durch die raue Umgebung im Automobil gilt der CAN-Bus als sehr robust und findet immer mehr Verwendung auch in anderen Bereichen, etwa in der Automatisierungstechnik. Auf Grund der zwanzigjährigen Geschichte ist der CAN-Bus ausgereift und kostengünstig verfügbar. Auf die in der Modellbahnwelt benutzten Protokolle wie z.B. DCC wurde



bewusst verzichtet. Zum einen ist DCC nur eine Teillösung, weil eine Ansteuerung durch einen Rechner nicht in einer Spezifikation abgedeckt ist. Zum anderen ist die Möglichkeit der Rückmeldung begrenzt. DCC ist ursprünglich unidirektional ausgelegt. Die bidirektionale Nutzung liegt derzeit nur als Empfehlung vor. Des Weiteren haben sich andere Protokolle, die der Rückmeldung dienen etabliert. Insgesamt ist dadurch der Digitalmarkt im Modellbahnumfeld relativ unübersichtlich. Bei der Entwicklung der Steuerung ging es auch nicht darum, eine weitere Steuerung auf den Markt zu bringen und anderen Systemen Konkurrenz zum machen, sondern eine Machbarkeitsstudie für eine modellgetriebene Mikrocontroller-Firmware zu entwickeln. Aus diesem Grund wurde die Steuerung samt Rückmeldungsmöglichkeit als Ganzes völlig neu konzipiert. Zu berücksichtigen bleibt, dass die Steuerung nur den verbauten Anteil der Anlage, also Gleise, Weichen und Signale, ansteuern. Die Loks können unabhängig davon sowohl klassisch analog, als auch digital mit DCC angesteuert werden. Diese Steuerung nimmt keinen Einfluss auf das Verhalten von Lokomotiven!

## *Architektur*

Grob gesehen besteht die Eisenbahnsteuerung aus einem Thick Client, der die Fahrstraßensteuerung und die Visualisierung über eine GUI übernimmt und der zu steuernden Hardware samt ihrer Mikrocontroller, welche die Ansteuerung der Geräte übernimmt. Die Schnittstelle zwischen einem High-Level-Rechner und der Eisenbahnanlage übernimmt die RS232-Schnittstelle.

Anlagenseitig ist direkt an der RS232-Schnittstelle ein CAN-Gateway angeschlossen. Dieses setzt die über RS232 laufenden Kommandos auf den CAN-Bus um. Am CAN-Bus sind in Reihe geschaltet die CAN-Knoten. Der CAN-Bus wird mit 128 KBit/s betrieben, was eine Länge von 500 Metern mit 110 CAN-Knoten möglich macht. Der CAN-Knoten kann Kommandos wie „Schalte Weiche 19 nach rechts“ oder „Schalte Signal P3 auf Hp1“ abarbeiten. Dabei hat dieser CAN-Knoten die vollständige Kontrolle über den Schaltvorgang. Als Rückmeldung kann dann Erfolg oder Fehler gemeldet werden. Des Weiteren können vom CAN-Knoten eigenständige Meldungen erzeugt werden. Das sind Gleisbesetztmeldungen, die durch fahrende Loks entstehen und manuelle Weichenschaltvorgänge.

Rechnerseitig wird ein Eclipse-RCP-Client verwendet. Dieser kommuniziert mit der RS232-Schnittstelle mittels der RXTX-Api. Die Modelleisenbahn wird in einem EMF-Modell beschrieben. Aus dieser Beschreibung wird zum Teil die GUI generiert. Mittels der GUI können Fahrstraßen ausgewählt und geschaltet werden. Dabei wird zwischen normalen Zugfahrten und Rangierfahrten unterschieden. Rangierfahrten können einen Bahnhof nicht verlassen. Die Berechnung einer Fahrstraße berücksichtigt belegte Gleisabschnitte sowie durch Fahrstraßen blockierte Gleisabschnitte. Die Schaltung einer Fahrstraße erfolgt in einer Abfolge von drei Kommandostapeln:

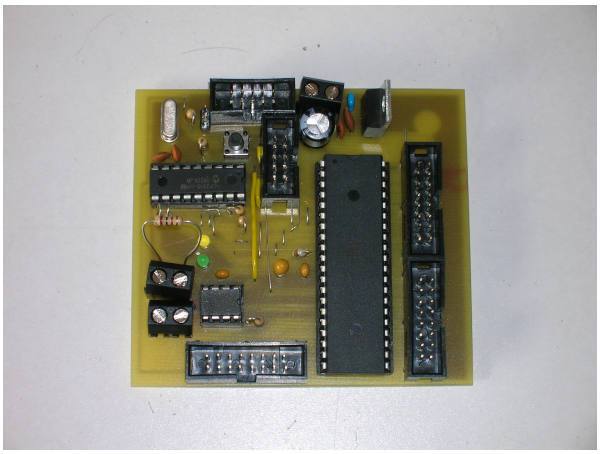
1. Schaltung der Weichen
2. Schaltung der Signale
3. Freigabe der Gleisabschnitte

Jeder Stapel muss vollständig und korrekt abgearbeitet werden, bevor der nächste Stapel bearbeitet wird. So kann verhindert werden, dass eine Weiche nicht in der korrekten Lage ist und so den Zug unter Umständen zur Entgleisung führt. Sobald alle Stapel bearbeitet sind, gilt die Fahrstraße als gesichert. Die von den CAN-Knoten erzeugten automatischen Rückmeldungen werden verwendet, um eine automatische Gleisfreigabe zu ermöglichen. Gleisabschnitte, die verlassen wurden, lassen sich so in neue Fahrstraßen einplanen. Manuelle Schalteingriffe von Weichen werden auch ausgewertet. Ist eine solche Weiche Bestandteil einer gültigen Fahrstraße, wird durch diesen Eingriff der Fahrweg verändert. Um dadurch bedingt Unfälle zu verhindern, wird die komplette

Fahrstraße aufgelöst.

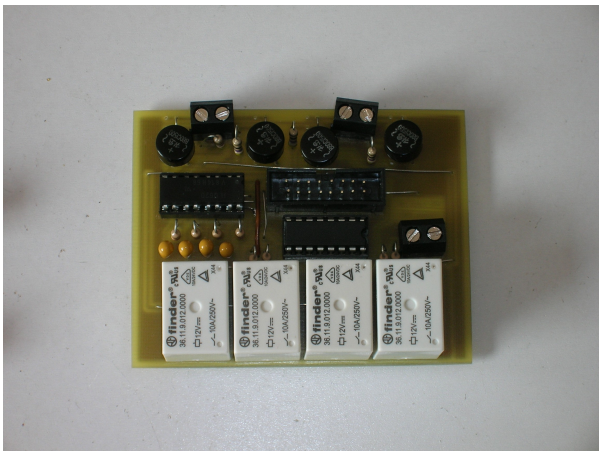
## *Die Module*

### Die CAN-Module



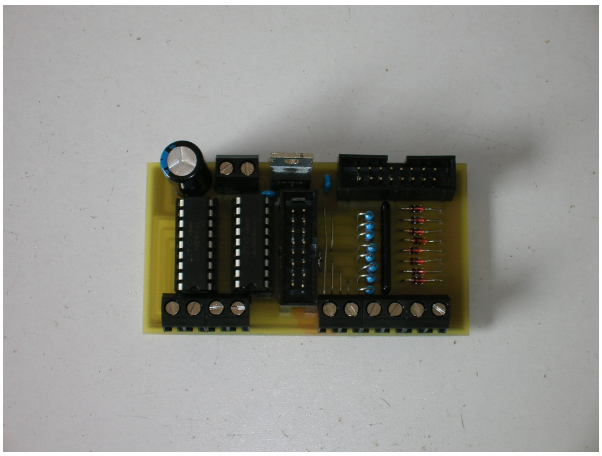
Die Aufgaben des CAN-Gateway und der CAN-Knoten wurde schon beschrieben. Beide Module basieren auf Atmels ATmega32 Mikrocontroller. Sie haben 32 KByte Flash, 1 KByte EEPROM und 2 KByte SRAM. Des Weiteren verfügt der ATmega32 über 32 Mehrzweck-IO-Leitungen. Fünf Leitungen werden für die Kommunikation mit dem CAN-Controller MCP2515 über SPI-Interface verwendet. Der CAN-Knoten verbraucht weitere drei IO-Leitungen zur Kommunikation über das serielle Interface. Die restlichen 24 IO-Leitungen stehen für Steuermodule zur Verfügung. Diese sind über drei Acht-Bit-Ports über 16-polige Flachbandstecker nach außen geführt. Über die restlichen acht Leitungen werden 5V, 12V und Masse als Stromversorgung an die externen Module geführt. Derzeit werden diese für simple Ein- und Ausgaben verwendet – es werden also nur Bits abgefragt bzw. Bits gesetzt. Ein Einsatz für die DA- und AD-Wandler stellt kein Problem dar, wird aber durch bisherige Module noch nicht verwendet. Das CAN-Gateway besitzt eine feste Firmware, die durch einen externen SPI-Stecker umprogrammiert werden kann. Die CAN-Knoten dagegen haben einen CAN-Bootloader. Über diesen kann neue Firmware über den CAN-Bus übertragen und auf dem Knoten selbst im System programmiert werden. Der Bootloader wird einmal initial über einen SPI-Anschluss geflasht. Für die Nutz-Firmware stehen so 28 KByte zur Verfügung. Der große Vorteil an diesem Verfahren ist, dass Firmwareupdates zentral in wenigen Sekunden auf alle Knoten gespielt werden können, ohne einen halben Nachmittag mit einem Programmiergerät unter der Anlage liegen zu müssen.

### Das Gleismodul



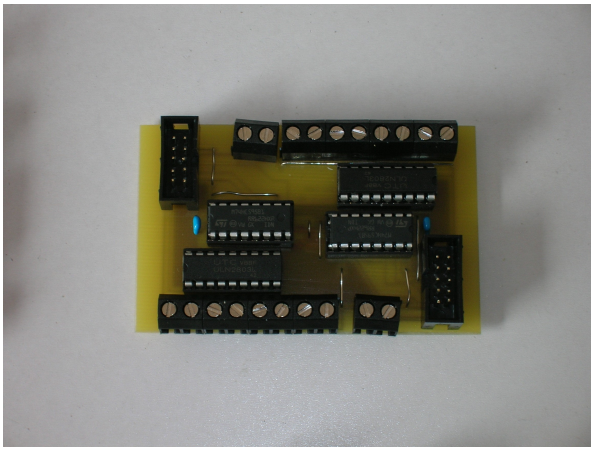
Das Gleismodul kann vier Gleisabschnitte verwalten. Ein Gleisabschnitt kann über ein Relais stromlos geschaltet werden. Unabhängig vom Zustand des Relais kann festgestellt werden, ob auf diesem Gleisabschnitt ein „Stromverbraucher“ steht. Damit können Loks auf den Gleisen detektiert werden. Auch Waggons mit Beleuchtung lassen sich damit erkennen. Um die Anwesenheit von Waggons festzustellen, müssen die Achsen der Waggons mit einem Widerstand ausgestattet werden. Die Detektion erfolgt darüber, dass zwei antiparallel geschaltete Dioden bei Stromfluss (beide Polaritäten) einen Spannungsabfall hervorrufen. Dieser steuert einen Optokoppler an und kann die Information somit an den CAN-Knoten weiterleiten. Da Fahrstrom über die Relais eingespeist werden muss, muss dieser auf das Gleismodul gelangen. Dieses geschieht über eine weitere Anschlussklemme.

## Das Magnetartikelmodul



Das Magnetartikelmodul kann Weichen und Formsignale mit Spulenantrieb ansteuern. Haben Weichen eine Endabschaltung, lässt sich darüber auch die aktuelle Weichenlage feststellen. So kann einerseits ein Schaltvorgang beschleunigt werden, andererseits lassen sich dadurch auch manuelle Schaltvorgänge melden. Es können acht Spulen angesteuert werden, was vier Weichen entspricht. Die Weichenlage wird über acht weitere Anschlüsse gemeldet. So benötigt das Magnetartikelmodul zwei der CAN-Knoten-Ports. Die Firmware der CAN-Knoten achtet darauf, dass nur eine Weiche gleichzeitig geschaltet wird, um zu hohen Stromfluss zu vermeiden. Pro Spule sollte der Stromfluss nicht über 1A liegen. Die Weichen werden mit 18V geschaltet. Da diese Spannung nicht über das Flachbandkabel geliefert wird, muss es über eine zusätzliche Anschlussklemme bereitgestellt werden. Da auf dem Modul ein 18V-Spannungsregler vorhanden ist, kann das Modul mit 18V bis 50V versorgt werden, falls nicht 18V zur Verfügung stehen.

## Das Lichtsignalmodul



Das Lichtsignalmodul steuert Lichtsignale an. Es wird über den seriellen Anschluss des CAN-Knotens angeschlossen. Die Lichtsignale können kaskadiert angeschlossen werden. Es werden jedoch nicht mehr als vier Module angesteuert. Jedes Modul hat 16 Anschlüsse. Diese können beliebig mit unterschiedlich großen Lichtsignalen gemischt werden. Ein Ausfahrsignal benötigt fünf Anschlüsse, ein Gleissperrsignal nur zwei. So lassen sich zwei Ausfahrsignale mit einem Vorsignal (vier Anschlüsse) und einem Gleissperrsignal mischen. Der CAN-Knoten berechnet zu dieser Anschlussbelegung abhängig vom anzuzeigenden Signalbild ein Bitmuster, was seriell über die Anschlüsse an die simplen CMOS-Schieberegister weitergereicht wird. Über Leistungstreiber (ULN 2803) wird dann die Lämpchen bzw. LEDs des Signals angesteuert. Die Spannungsversorgung von 12V geschieht über Flachbandkabel vom CAN-Knoten.

## *Das Protokoll*

Die Kommandos an die CAN-Knoten werden vom Rechner über RS232 auf den CAN-Bus geschickt. Dabei sind diese Kommandos schon in CAN-Frames eingebettet. Damit die Synchronisation der byteweisen Kommunikation zwischen Rechner und CAN-Gateway erhalten bleibt, wird die Übertragung mit einem Prüfsummen-Byte abgeschlossen. Die Aufträge lassen sich in drei Gruppen einteilen:

- Schaltaufträge
- Verwaltungsaufträge

- Konfigurationsaufträge

Die Struktur aller Aufträge ist gleich. Ein CAN-Telegramm kann zwischen keinem und acht Bytes beinhalten. Bei der Modellbahnsteuerung wird mindestens ein Byte transportiert: Das Kommandobyte. Je nach Kommando können weitere Bytes hinzukommen. Gerade die Konfigurationskommandos können die acht Bytes ausreizen.

Obwohl die ID eines CAN-Telegramms als Objektidentifier gemeint ist, wird bei der Modellbahnsteuerung damit der CAN-Knoten adressiert. Es können sowohl Standard Frames, als auch Extended Frames benutzt werden. Standard Frames werden für Controller-Aufgaben benutzt und Extended Frames zur Ansteuerung von Geräten. Die im Standard Frame benutzte ID adressiert somit den Controller und kann dadurch elf Bit breit sein. Alle IDs über 2000 und die „0“ sind reserviert. Die „0“ ist die Broadcast-Adresse, auf die alle CAN-Knoten hören. Des Weiteren hat das CAN-Gateway die Adresse 2047. Der restliche Raum zwischen 2000 und 2046 ist für Testzwecke reserviert. Obwohl die CAN-Knoten untereinander CAN-Messages austauschen könnten, wird bei der Modellbahnsteuerung davon derzeitig kein Gebrauch gemacht. Somit werden Kommandos nur vom CAN-Gateway verschickt. Die Antworten der CAN-Knoten laufen immer zum CAN-Gateway zurück.

Werden Schaltaufträge an Geräte gesendet, wird immer ein Extended Frame verwendet. Von den zusätzlichen 18 Bit werden „nur“ 16 verwendet und wird als Gerätenummer bezeichnet. Die restlichen Bits stehen auf 0. Somit könnten theoretisch 65536 Geräte pro CAN-Knoten angesteuert werden. Da die Konfiguration im EEPROM des CAN-Knotens hinterlegt wird, ist die Zahl der Geräte auf die Größe des EEPROMS begrenzt. Beim ATmega32 sind das ein Kbyte. Somit kann ein ATmega32 maximal 64 Geräte verwalten.

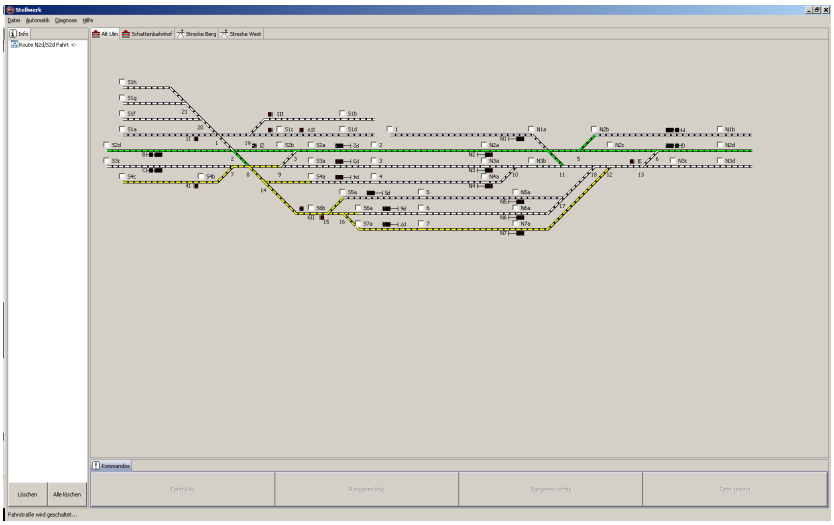
Die Antworten haben als Zieladresse immer das CAN-Gateway und liegen grundsätzlich als Extended Frame vor. Die Standard-ID ist dann immer 2047 für das Gateway und die zusätzlichen 18 Bit der Extended-ID umfassen die absendende Controller-ID. Dadurch ist eine saubere Arbitrierung des CAN-Busses möglich, wenn z.B. auf ein Broadcast-Ping alle CAN-Knoten antworten müssen. Die Antworten enthalten neben dem ausgelösten Kommando noch einen ein Byte großen Antwortcode. Zusätzlich werden vier Bytes für die Absende-ID (also Controller-ID und Gerätenummer) benötigt. Die restlichen zwei Bytes können für zusätzliche Informationen benutzt werden. Hier können Weichenlage, Gleisbesetztmeldung sowie Statusinformationen über den Mikrocontroller untergebracht werden. Ein Kommando muss nicht zwingend zu einer Antwort führen. Es kann auch sein, dass ein Kommando sogar zeitversetzt mehrere Antworten auslöst. Ein Weichenschaltauftrag meldet zuerst, dass der Auftrag eingereiht wurde. Nach erfolgreicher Ausführung wird einerseits der Erfolg gemeldet, aber auch zusätzlich die erreichte Endlage. So können auf höherer Ebene die Endlagenmeldungen unabhängig von den Schaltaufträgen verarbeitet werden.

## *Die Stellwerks-Software*

Die Stellwerks-Software ist zum Teil aus modellgetrieben generiert. Die Hauptlogik ist aber klassisch implementiert. Die GUI basiert auf Eclipse RCP/SWT. Mit ihr werden Fahrstraßen ausgewählt und der Schaltvorgang ausgelöst. Dabei wird intensiv von Multi-Threading Gebrauch gemacht. Dies wird durch das asynchrone Auftreten von Rückmeldungen nötig. Sämtliche Zustände werden sofort ausgewertet und visualisiert. Es kann sogar parallel ein automatischer Modus aktiviert werden, der aus einem Schattenbahnhof zufällig Züge auswählt und diese durch wieder zufällig ausgewählte Gleise eines Bahnhofs fahren lässt. Die Visualisierung der GUI zeigt an, ob eine Fahrstraße geschaltet wird (gelb) oder fertig geschaltet ist (grün). Besetzte Gleisabschnitte werden orange angezeigt. Auch der Schaltzustand der Signale wird dargestellt. Befindet sich noch

ein anhängiger Weichenschaltauftrag, wird die Weichennummer blinkend dargestellt. Des Weiteren befinden sich im Gleisbild Checkboxen, die zur Planung einer Fahrstraße herangezogen werden.

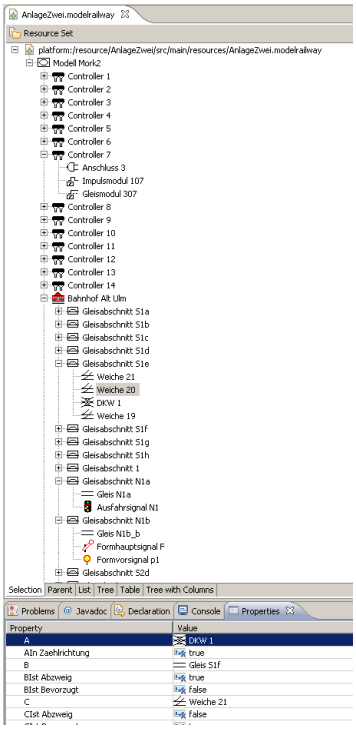
Während die GUI recht übersichtlich erscheint und nur wenig Eingriffsmöglichkeiten zu bieten scheint, sind in der zu Grunde liegenden Logik viele komplexe Vorgänge implementiert. Als erstes sei die Berechnung einer Fahrstraße aus selektierten Gleisabschnitten erwähnt. Dieses Verhalten entspricht dem Vorbild, bei dem durch Start- und Zielwahl ebenfalls eine Fahrstraße gebildet werden kann. Die Fahrstraßenberechnung berücksichtigt schon belegte Gleisabschnitte sowie durch andere Fahrstraßen verplante Gleisabschnitte. Im Modell sind an einer Weiche bevorzugte Abzweigungen modelliert. Diese werden bei der Fahrstraßenwahl zu erst gewählt. Erst, wenn ein Abzweig nicht zum Ziel führt, wird der andere Abzweig gewählt. Die Modellierung gibt auch Abzweigungen („Kurven“) an. Dadurch können Langsamfahrstellen forciert werden, die sich im Signalbild niederschlagen. Alle Signalbilder werden korrekt berechnet. Dies bezieht sich auf Langsamfahrstellen, Vorsignale, mit Hauptsignalen kombinierte Vorsignale sowie durch Rangierfahrten bedingt die Gleissperrsignale insbesondere in Verbindung mit Ausfahrsignalen.



Automatische Rückmeldungen der CAN-Knoten werden ebenfalls ausgewertet. Dies ist besonders für die automatische Gleisfreigabe nötig. Verlässt eine Lok einen Gleisabschnitt und befindet sich im darauf folgenden, dann kann dieser Gleisabschnitt stromlos geschaltet werden und für eine neue Fahrstraßenplanung zur Verfügung stehen. Die Bedingung, dass der darauf folgende Abschnitt belegt ist, ist an dieser Stelle wichtig. Gerade beim Zweileiter-Gleichstrombetrieb einer Modelleisenbahnanlage kann es zu Kontaktschwierigkeiten durch Dreck kommen. In diesem Falle verschwindet die Lok scheinbar von den Gleisen. Das darf aber nicht zu einer Gleisfreigabe führen. Hat ein Zug den Zielabschnitt erreicht, wird die Fahrstraße komplett aufgelöst. Dadurch ist kein Benutzereingriff mehr nötig.

## *Das Metamodell*

Das Metamodell beschreibt, wie allgemein eine Modelleisenbahnanlage aussehen kann. Es beinhaltet den gesamten Gleisplan, die Aufteilung der Gleisabschnitte sowie die Gruppierung in Bahnhöfe und Strecken. Des Weiteren sind unterschiedliche Signale im Modell enthalten. Weitere Eigenschaften wie bevorzugte Abzweigungen und Abzweigungen (im Sinne von „Kurven“) ergänzen die Modellbeschreibung. Das Metamodell ist als EMF-Modell mit MagicDraw erstellt worden. Durch die openArchitectureWare-Cartridge „uml2ecore“ kann das in ein ECore umgewandelt werden. Aus diesem ECore kann für Eclipse ein Editor erstellt werden, mit dem eine Eisenbahnanlage erfasst werden kann. Jedem Gleisteil kann eine Zählrichtung vorgegeben werden, sodass Fahrtrichtungen bei der Fahrstraßenschaltung berücksichtigt werden können. Die Validierung hat unter Anderem die Aufgabe zu prüfen, ob die modellierte Zählrichtung für alle Gleisteile schlüssig ist. Aus diesem Modell werden Code-Teile generiert, die eine Konfiguration der Eisenbahn enthält. Daraus wiederum können die Konfigurationskommandos erzeugt werden, die die CAN-Knoten mit den Geräteinformationen versorgt. Des Weiteren wird diese Konfigurationsinformationen für die Anzeige der GUI benötigt.




## *Modellgetriebene Firmware*

In der ersten Fassung war die Firmware der CAN-Knoten als Referenzimplementierung noch klassisch implementiert. Die Kommandoverarbeitung in der Firmware der CAN-Knoten war ein unschönes Stück Spaghetti-Code. Aus diesem Grund wurde ein Zustandsdiagramm gefertigt, das die Betriebszustände des CAN-Knoten enthält. Die eingehenden Kommandos sind als Zustandsübergänge modelliert. Aus diesem Diagramm wird mit Hilfe von openArchitectureWare der Code für die Kommandoauswertung generiert. Der generierte Code enthält zwar relativ viele Callbacks und switch/case-Anweisungen und bläht den Binärcode um ca. 1,5 KByte auf. Das stellt aber kein Problem dar, weil jetzt noch Platz für ca. 18 KByte im Flash des ATmega32 vorhanden ist. Ein weiterer Vorteil ist, dass der generierte Code wesentlich übersichtlicher aussieht und somit wartbarer wird. Änderungen und Erweiterungen der Steuerkommandos lassen sich jetzt viel leichter einpflegen. Die Integration des generierten Codes in den bestehenden Firmware-Rahmen gestaltete sich als unproblematisch, da der restliche klassisch implementierte Teil nur aus Service-Methoden zur Ansteuerung der Hardware dient.