Bedienungs- und Aufbauanleitung

V1.0 Stand 03/2017

Das SpassCAN - System

Liebe Freundinnen, liebe Freunde der gepflegten Modellbahnerei,

im Folgenden möchten wir Euch unser SpassCAN - System zur Überwachung jeg-licher Sensorik auf Modellbahnanlagen vorstellen. Da wir und das namengebende Spassbahnforum aus dem Großbahn-/Gartenbahnbereich kommen, bezieht sich alles hier Gesagte auch auf die große Spur, dennoch ist das System, eventuell mit Modifizierungen, natürlich auch in anderen Spurweiten einsetzbar.

Insgesamt handelt es sich um ein Open Source Projekt, das hier der Öffentlichkeit preisgegeben wird, will sagen: Ihr baut und setzt alle Module, Software wie Hard-ware, auf eigene Gefahr ein. Sollte sich also Eure gesamte Anlage bei Einsatz unserer Module in ein blaues Wölkchen aus Qualm und Atomstrom auflösen, dann macht uns bitte nicht dafür verantwortlich! Alles, was wir hier vorstellen, funktioniert auf unserer Anlage; das bedeutet aber nicht, dass es fehlerfrei auch bei Euch funktioniert muss (, obwohl wir natürlich schon davon ausgehen).

1. Wer braucht das SpassCAN-System?

Um es gleich vorweg zu sagen: Dies ist nichts für Analogfahrer und nichts für Modellbahner, die noch nie mit einer gewissen Automatisierung Ihrer Anlage geliebäugelt haben.

Wer jedoch bereits mit Steuerungsprogrammen wie RocRail, Traincontroller, Railware o.ä. experimentiert hat, der weiß, dass der steuernde Computer, da blind und unwissend, über die Situation auf der Anlage per Rückmeldung von besetzten Gleisen auf dem Laufenden gehalten werden muss. Da kommt das SpassCAN-System ins Spiel, nur SpassCAN kann noch wesentlich mehr ...

1.1 Warum CAN und nicht vorhandene Rückmeldesysteme?

Die Rückmeldung der Gleiszustände erfolgt heute auf den meisten Anlagen unter Einsatz des S88 Busses, bzw. über Littfinskis HSI, welches aber auch nur eine Zusammenführung von bis zu drei S88 Strängen ist.

Worin liegen aber die Vorteile des CAN-Busses?

- die Buslänge bträgt bis zu 500m, das sind Strecken, die auch im Gartenbahn-bereich schwerlich zu erreichen sind
- hohe Busgeschwindigkeit, in der langsamsten Ausführung, die wir verwenden, da sie die größte Leitungslänge erlaubt, 128Kb/Sec, was ein Vielfaches der S88 Geschwindigkeit ist
- geringe Fehleranfälligkeit, zum einen durch die Hardware, zum zweiten durch das verwendete Protokoll und zum dritten durch die Bidirektionalität, d.h. an-geschlossene Module und die Zentrale vergewissern sich, ob ihre Nachrichten angekommen sind und richtig verstanden wurden. So werden Übertragungs-fehler weitgehend vermieden
- universell einzusetzen; der SpassCAN-Bus kann nicht nur zur Gleisrück-meldung genutzt werden, sondern auch für jede andere denkbare Sensorik. Wir haben bereits einen RFID-Leser und Helligkeitssensoren integriert und denken über viele andere Anwendungsmöglichkeiten nach. Sollte jemand noch zündende Ideen haben, immer raus damit, möglich ist (fast) alles ...
- die Bauteile sind preiswert; denn da es sich um einen in der Industrie, vor allem im Automobilbau, durchgesetzten Standard handelt, sind die Komponenten reichlich vorhanden und preiswert zu erwerben

1.2 Und was ist der CAN-Bus überhaupt?

Wer es gern technisch mag, den verweise ich zu dieser Frage auf Wikipedia, für uns Ottonormalbahner reicht es zu wissen, dass der CAN-Bus aus einer Zentrale und vielen Modulen besteht, die alle hintereinander an einem Kabel hängen. An beiden Enden wird das Kabel mit einem Abschlusswiderstand begrenzt. Auf besagtem Kabel melden die Module nun also ihre Erkenntnisse an die Zentrale. Z.B. wenn ein Gleisabschnitt besetzt wurde, eine bestimmte Lok einen bestimmten Punkt passiert oder die Lichtverhältnisse langsam das Einschalten der Wagenbeleuchtung erforderlich machen. Weil wir es hier mit einem "Multimaster"-Bus zu tun haben, werden die Module nicht von der Zentrale abgefragt, sondern jedes meldet selbständig jede Änderung seines Zustands. Da nun aber den Modulen je nach Wichtigkeit eine unterschiedliche Priorität zugeteilt wird, die Wagenbeleuchtung ist nicht so wichtig wie die Verhinderung eines Zusammenstoßes, werden die Ereignisse in der Reihenfolge ihrer Wichtigkeit von der Zentrale abgearbeitet.

1.3 Was braucht man für den Einsatz auf der Modellbahn?

Zunächst braucht man natürlich einen Computer mit einem Steuerungsprogramm wie zum Beispiel RocRail. Auch andere Steuerungsprogramme wie Railware, Train Controller etc. funktionieren, wenn man den SpassCan mit der HSI Emulation betreibt. Dies macht allerdings nur den halben Spaß, da das System dann eben nur zur Gleisbesetztmeldung genutzt werden kann. Zur vollen Ausnutzung aller Möglichkeiten muss nämlich das RASCII Protokoll unterstützt werden, und das kann (logischerweise) bis jetzt nur RocRail, aber vielleicht werden ja noch ein paar andere "Softwareriesen" folgen.

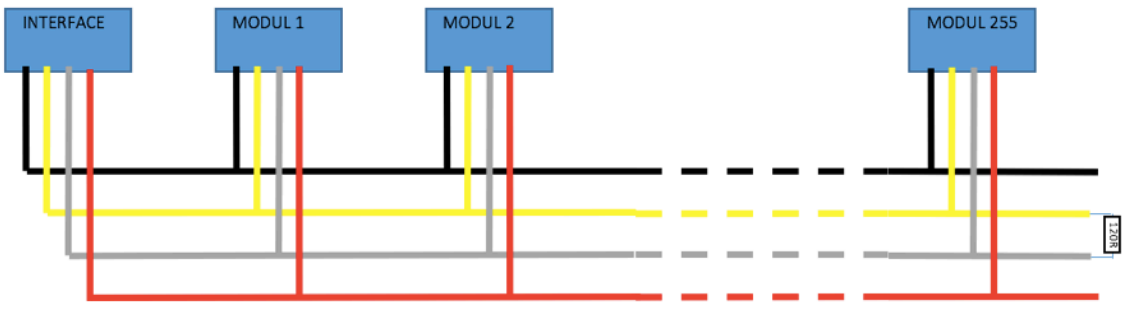
Wenn der Computer läuft, benötigen wir ein Interface, d.h. einen Dolmetscher zwischen dem CAN-Bus und dem Steuerungscomputer, kurz eine weitere Platine, bei uns ein Arduino-Nano, der für ein paar Euro (maximal zehn) bei den bekannten Quellen zu erstehen ist. Die übrige Funktionalität erbringt die Interface-Platine. Das Programm dazu gibt es bei uns, gratis, die Gewährleistung dazu nirgends, siehe die Einleitung oben, aber gemeinsam kriegen wir das Ding schon zum Laufen! Um das ganze Konstrukt mit Strom zu versorgen, ist noch ein beliebiges Netzteil/Trafo zwischen 12 und 18 V DC oder AC erforderlich.

Alle übrigen Busteilnehmer, sprich: Module, werden nach Wunsch und Bedarf ergänzt. Zur Zeit gibt es Gleisbesetztmelder für 16 bzw. 24 Meldeabschnitte, RFID-Melder und S88/CAN Adapter. Allen ist gemeinsam, dass man bei Ihnen selbst Hand anlegen muss, denn es gibt sie nur als Bausatz. Somit heißt es den Lötkolben anheizen und los geht es mit dem fröhlichen Modulebraten, selbstverständlich gemäß der Bau- und Lötanleitungen.

1.4 Welche Kabel müssen auf der Anlage zusätzlich verlegt werden?

Diese Frage ist mit einem eindeutigen "kommt drauf an" zu beantworten. Klar, man braucht das Kabel für den CAN-Bus. Beginnend beim ersten Modul, in der Regel dem Interface (muss aber nicht!) und endend beim letzten Modul. Merke: Vom Hauptkabel abzweigende Stichstrecken sind nicht erlaubt! Außerdem muss natürlich von den Modulen aus eine Verkabelung je nach Art und Zweck erfolgen. In jedem Fall muss der Verlauf der Verkabelung sorgfältig geplant werden, nachträgliche Änderungen sind zwar jederzeit möglich, doch gerade im Garten auch mit einem erheblichen Aufwand verbunden, "drum prüfe, was Du auf ewig verbindest!"

Wie bereits weiter oben erwähnt, sind an beiden Enden des Buskabels Terminierungswiderstände vorzusehen. Beginnt der Strang mit dem Interface, ist auf dieser Seite nichts weiter zu tun, denn auf der Platine sind die entsprechen-den Widerstände bereits vorhanden. Wenn man diese jedoch brückt, kann das Interface auch an beliebiger Stelle im Strang eingebunden werden.



1.5 Was für ein Kabel muss verlegt werden?

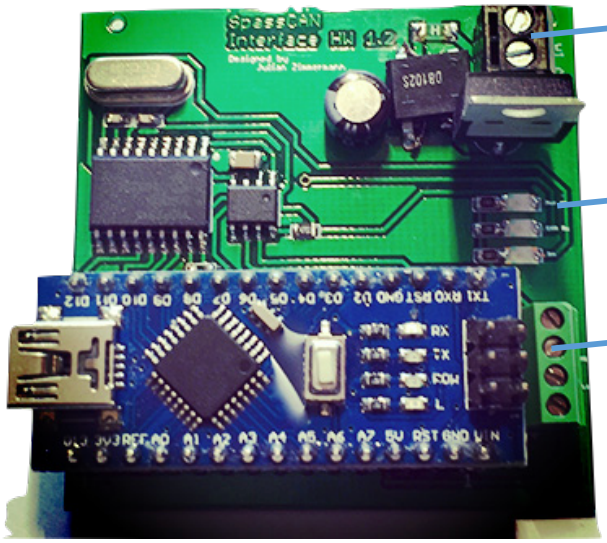
Der SpassCAN - Bus ist nicht sonderlich anspruchsvoll, was die Art der Verkabelung angeht. Wichtig ist, dass ein vieradriges Kabel verlegt wird, zwei Signalleitungen und zwei für die Stromversorgung des Busses. Wenn man möchte, kann man CAT Kabel aus der PC-Netzwerktechnik nehmen, eine Abschirmung ist jedoch nicht erforderlich, so dass auch andere Kabel problemlos zu verwenden sind.

2. Die verfügbaren Module

Hier nun eine Auflistung der verfügbaren Module und ihrer Anschlussmöglich-keiten. Aber Achtung: Alle hier vorgestellten Module sind nur als Bausatz lieferbar, es muss also selbst gelötet werden!

(to be continued...)

2.1 Das Interface V 1.2



Das Herzstück des Systems ist das Interface. Hier werden die Meldungen auf dem Bus interpretiert, zusammengefasst und an den Steuerrechner weitergeleitet.

Es gibt zwei verschiedene Softwareversionen. Zum einen die Littfinski HSI Emulation, die dann allerdings nur die Gleisbesetztmeldung unterstützt, zum anderen das SpassCAN/RASCII Protokoll, für den vollen Funktionsumfang.

Für den Steuerungsrechner wird eine Treibersoftware für den Seriell-/USB Wandler benötigt (CH340). Der entsprechende Link befindet sich auf www.spasscan.de .

Weiterhin erforderlich ist ein Netzteil, welches zwischen 0,5 und 1,0 Ampere bei 12V - 18V leisten sollte. Der Anschluss erfolgt an der oberen zweipoligen Klemme, die Polarität ist egal.

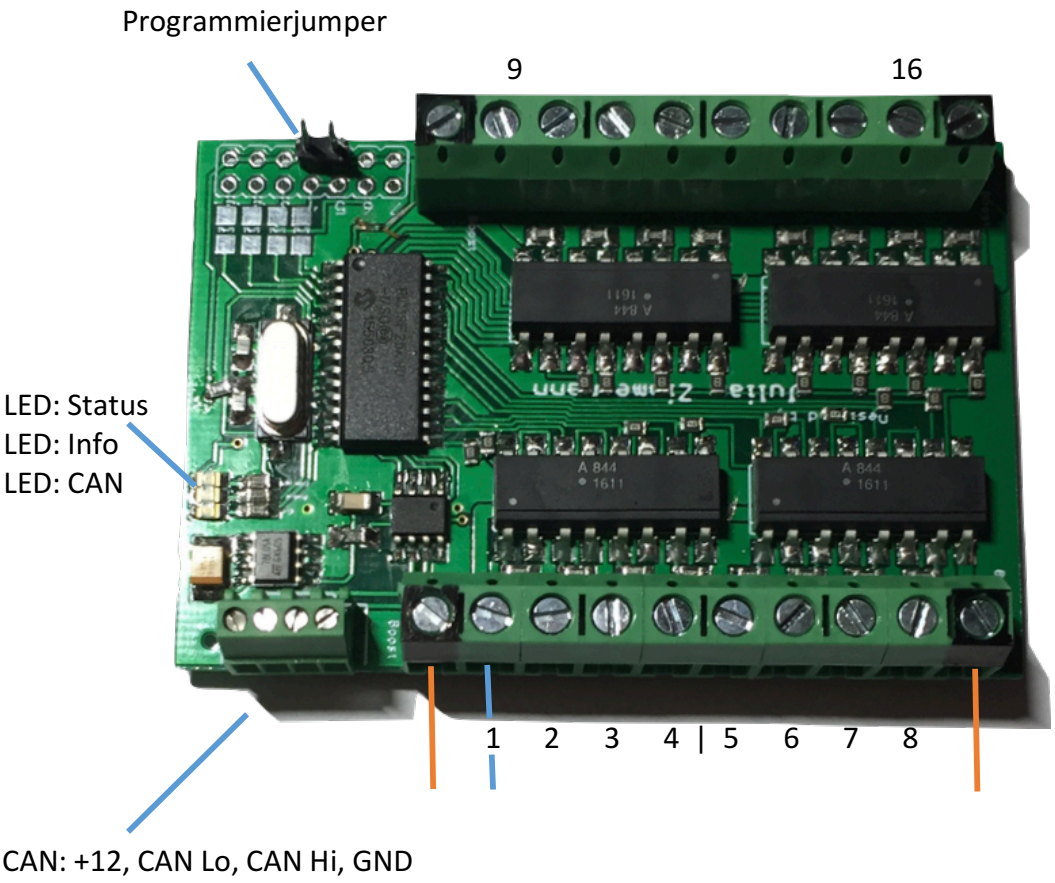
Die untere vierpolige Klemme ist der Busanschluss.

Zwischen den Anschlussklemmen befinden sich drei LEDs, die wie folgt melden:

Oben: Status - Das Interface erhält Strom
Mitte: CAN sendet
Unten: CAN empfängt

Hinter dem Busanschluss befindet sich eine Lötbrücke für den Widerstand der Busterminierung. Wenn sich das Interface nicht am Anfang des Busstrangs befindet, Ist diese zu entfernen.

2.2 **Der Melder 16 Vers. 1.1**



Der Melder 16 erlaubt es 16 (wer hätte das gedacht?) Rückmeldekontakte zu überwachen. Hierbei ist es möglich, zusammen mit einer zusätzlichen Diodenkaskade, Gleisabschnitte auf Stromverbrauch hin zu detektieren oder Impulse/Schalter zu überwachen, z.B. Schaltgleise, Hallsensoren, Reedkontakte.

Die Anschlüsse auf der Platine sind zu Fünfergruppen zusammengefasst, so dass es durchaus möglich ist, verschiedene Rückmeldetechniken mit dem gleichen Melder zu nutzen. Der äußere der fünf Anschlüsse ist der gemeinsame Leiter für die nachfolgenden vier Anschlussklemmen, an die die jeweiligen Geber, also Schaltkontakte oder Dioden angeschlossen werden. (Schaltungen Siehe 4.)

Auf der linken Platinenseite befinden sich drei LEDs, die die folgenden Zustände signalisieren:

Obere LED: Dauerlicht = Melder läuft

Blinklicht = Störung, d.h. der Melder sieht kein Interface, versucht aber dennoch, Daten zu übertragen .

Mittlere LED: Blinkt abwechselnd mit der oberen LED, wenn der Melder sich im Programmiermodus befindet.

Blinkt allein, wenn die Datenübertragung nicht erfolgreich war, versucht aber weiterhin, Daten zu übertragen.

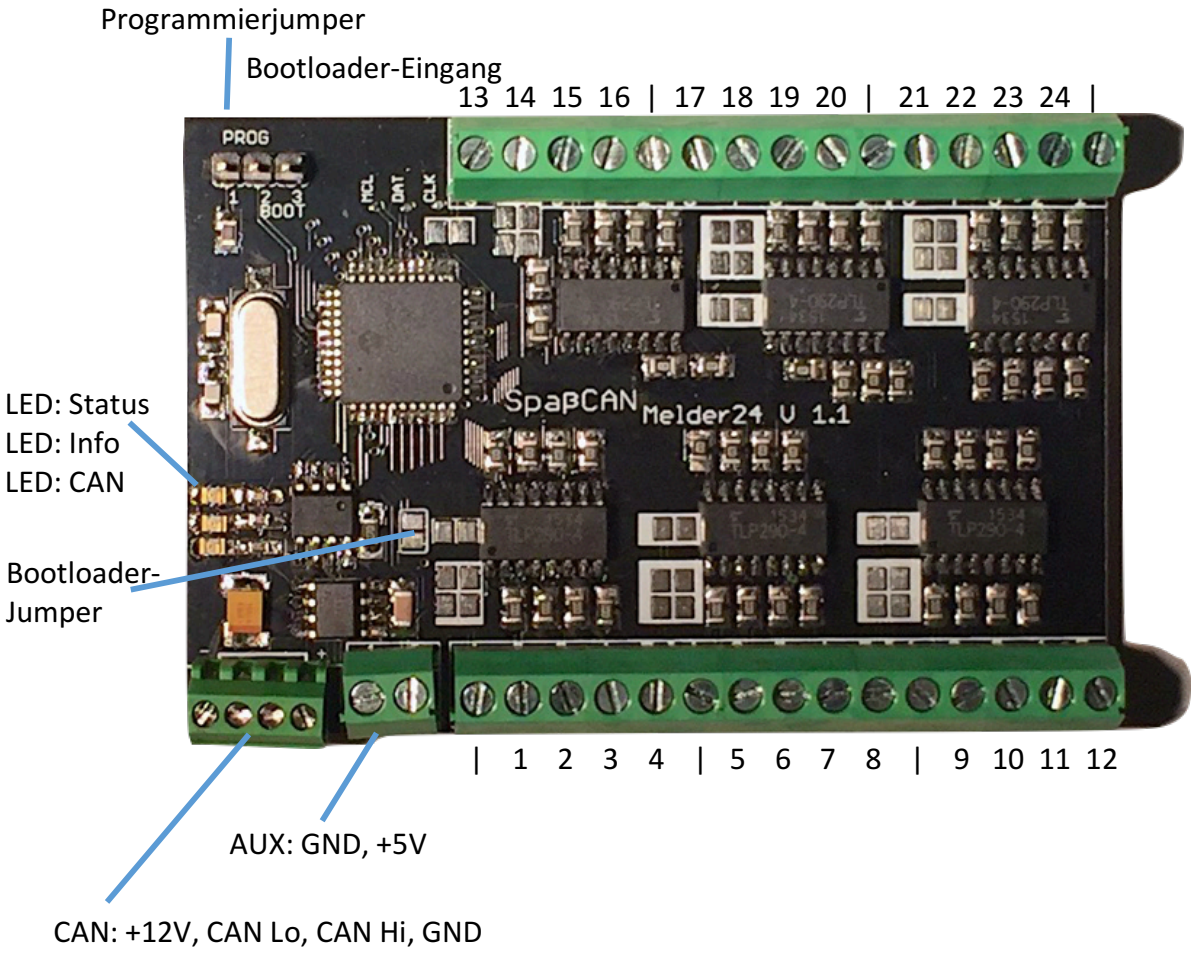
Blinkt allein, gleichzeitig ist die obere LED aus: Das Modul befindet sich im Passivmodus, weil es vom Interface wegen fortgesetzter fehlerhafter Datenübertragung gesperrt wurde.

Untere LED: Blinkt wenn Daten an den CAN Bus übergeben werden.

An der oberen Platinenkante befinden sich zwei Stifte, die mit einem Jumper geschlossen werden können, um das Modul in den Programmiermodus zu versetzen. Bei geschlossener Brücke meldet sich das Modul automatisch im Programmiertool an, woraufhin die eingegebenen Werte angezeigt und geändert werden können.

CV 16 (0X10)	Adresse	Moduladresse, aus der sich zusammen mit dem Meldertyp später die Bus-ID zusammensetzt. Werte: 1 - 254. 0 = Systemmeldung. 255 = Standardaresse (sollte vermieden werden). Standardwert: 255
CV 32 (0x20)	Entprellung	Anzahl der Auslesevorgänge, bevor die Eingangszustände bearbeitet werden. Hilfreich, wenn man mechanische Schalter verwendet, die ein Melderflackern auslösen könnten. Jeder Auslesevorgang hat ein delay von 5 ms. Werte: 1-128 Standardwert: 3
CV 33 (0x21)	Maskierung	Maskiert die Eingänge 1 - 8 und legt fest, ob diese bei einer Zustandsänderung eine Übertragung auf den Bus auslösen können. Die Eingänge werden ansonsten nur übertragen, wenn ein nicht maskierter Eingang geändert wird. Werte: 0 – 255 (Bitweise progremmierbar) Standard: 255 (alle werden gesendet)
CV 34 (0x22)	Maskierung	Wie CV 33, jedoch für Ports 9 - 16
CV 37 (0x25)	Datenformat	0: Portweise Übertragung mit 11 bit ID (Standard und richtige Option) 1: Portweise mit 29 Bit ID (CAN 2)(In Zukunft richtig) 2: Byteweise Übertragung (alte HSI Übertragung, obsolet) 3: Zimo-Datenformat (nicht getestet, nur teilweise kompatibel) 4: Universal-CAN (ALPHA-Status) 5: LIN-Bus (Zur Zeit noch mit langen Verzögerungen behaftet)
CV 255 (0xFF)	Ende	Beendet den Programmiermodus und schreibt die neuen Konfigurationen in den Speicher. Danach Jumper ziehen und Modul neu starten. Befehl wird von Programmiersoftware ausgelöst, wenn man "Abschließen" bestätigt.

2.3 Der Melder24 V1.1



Um einmal einen alten Running Gag wieder zu bemühen: Der sicherlich schönste Rückmelder, den wir je hatten, phantomschwarz und geheimnisvoll! Aber in diesem Fall ist es nicht allein so, dass 24 besser als 16 ist, es gibt ein paar subtile Unterscheidungsmerkmale zum Melder 16. Doch zunächst die Gemeinsamkeit: Gleich ist bei beiden Modulen die Zusammenfassung zu Fünfergruppen, in diesem Fall sechs Stück, aber sonst?

Direkt neben dem Busanschluss befindet sich ein 5 Volt Ausgang, belastbar bis ca. 300mA. Hier können beispielsweise Magnetsensoren mit Strom versorgt werden, was wiederum Kabelverlegung sparen kann. Aber auch andere Anwendungen sind denkbar.

Hinter dem 5V Anschluss befindet sich eine Lötbrücke. Sobald diese geschlossen ist, wird ein Boot-Loader Programm initialisiert, so dass an Pin 2 und 3 ein serielles Kabel zwecks Übertragung einer neuen Programmversion angeschlossen werden kann.

Jeder Anschlussreihe ist eine Gruppe von Lötpads zugeordnet. Diese dienen zur Konfiguration der Detektionsmethode. Werden die oberen beiden, etwas abgesetzten Pads verbunden, wird ein gemeinsamer Leiter am Optokoppler geschaltet, um so wie beim Melder 16 über eine Diodenkaskade Stromfluss zu erfassen.

Werden die beiden darunter liegenden Paare jeweils verbunden, führt der gemeinsame Leiter GND und +5V liegt auf dem Optokoppler, so dass mit einem Reed Kontakt, einem Schaltgleis o.ä. gegen Ground ein Signal ausgelöst wird. Auch hier ist es möglich, jede Gruppe unterschiedlich zu konfigurieren.

Programmiervariablen

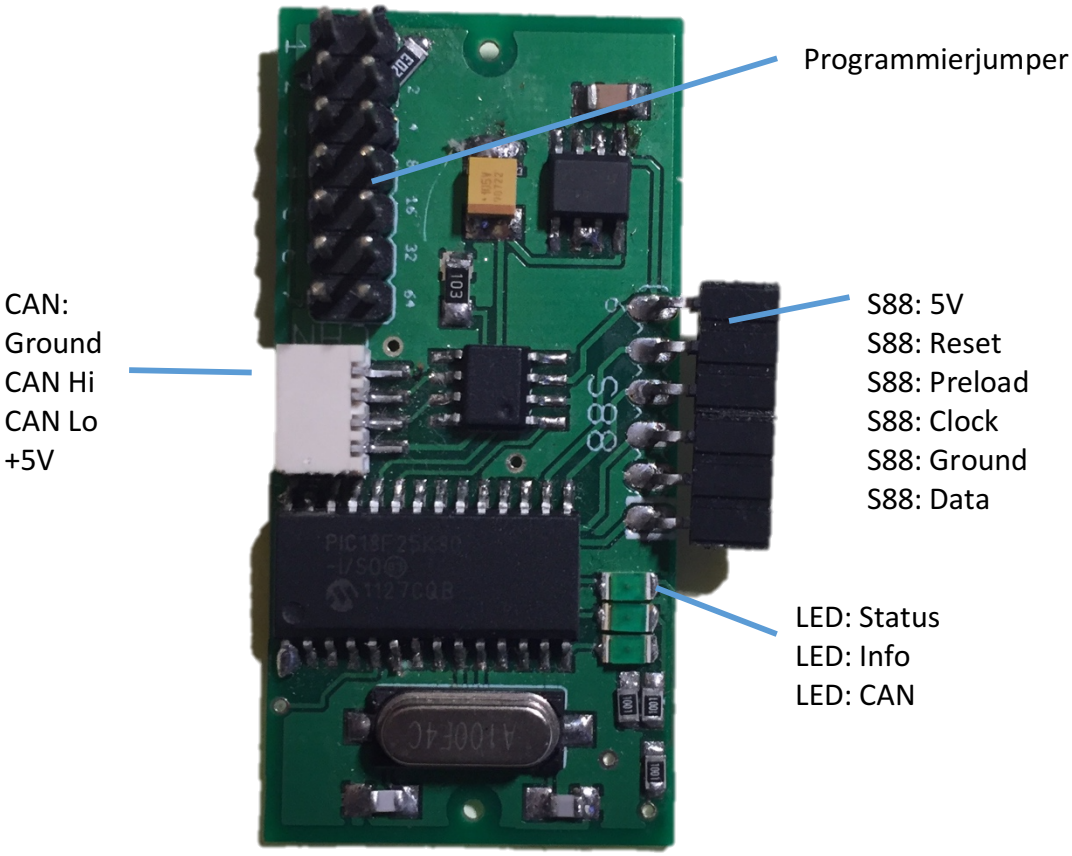
CV 16 (0X10)	Adresse	Moduladresse, aus der sich zusammen mit dem Meldertyp später die Bus-ID zusammensetzt. Werte: 1 - 254. 0 = Systemmeldung. 255 = Standardaresse (sollte vermieden werden). Standardwert: 255
CV 32 (0x20)	Entprellung	Anzahl der Auslesevorgänge, bevor die Eingangszustände bearbeitet werden. Hilfreich, wenn man mechanische Schalter verwendet, die ein Melderflackern auslösen könnten. Jeder Auslesevorgang hat ein delay von 5 ms. Werte: 1-128 Standardwert: 3
CV 33 (0x21)	Maskierung	Maskiert die Eingänge 1 - 8 und legt fest, ob diese bei einer Zustandsänderung eine Übertragung auf den Bus auslösen können. Die Eingänge werden ansonsten nur übertragen, wenn ein nicht maskierter Eingang geändert wird. Werte: 0 – 255 (Bitweise progremmierbar) Standard: 255 (alle werden gesendet)
CV 34 (0x22)	Maskierung	Wie CV 33, jedoch für Ports 9 - 16
CV 34 (0x23)	Maskierung	Wie CV 33, jedoch für Ports 17 – 24
CV 37 (0x25)	Datenformat	0: Portweise Übertragung mit 11 bit ID (Standard und richtige Option) 1: Portweise mit 29 Bit ID (CAN 2)(In Zukunft richtig) 3: Zimo-Datenformat (nicht getestet, nur teilweise kompatibel) 4: Universal-CAN (ALPHA-Status) 5: LIN-Bus (Zur Zeit noch mit langen Verzögerungen behaftet)
CV 255 (0xFF)	Ende	Beendet den Programmiermodus und schreibt die neuen Konfigurationen in den Speicher. Danach Jumper ziehen und Modul neu starten. Befehl wird von Programmiersoftware ausgelöst, wenn man "Abschließen" bestätigt.

2.4 Der SpaßCANS88 V1.0

Den S88 Adapter haben wir gebaut, um die vorhandene (teure) S88 Hardware nicht wegschmeißen zu müssen. Damit die alten Rückmelder verwendet werden können, werden sie mit diesem Adapter (einzeln) in das System integriert. Es können pro S88 Eingang jeweils 16 Eingänge ausgelesen werden.

Der Aufbau ist einfach. Der sechspolige Stecker (die 1 ist markiert) dient zum Anschluss des S88 Moduls, der vierpolige Molex-Stecker auf der anderen Seite zur Anbindung an den CAN-Bus.

Der Jumper schließlich muss zur Programmierung des Moduls gesetzt werden.

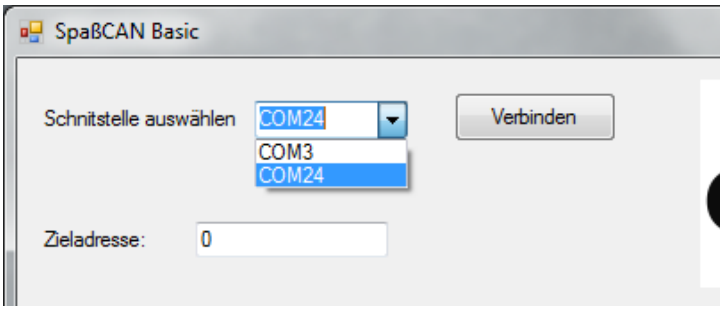


Programmiervariablen

CV 16 (0X10)	Adresse	Moduladresse, aus der sich zusammen mit dem Meldertyp später die Bus-ID zusammensetzt. Werte: 1 - 254. 0 = Systemmeldung. 255 = Standardaresse (sollte vermieden werden). Standardwert: 255
CV 32 (0x20)	Entprellung	Anzahl der Auslesevorgänge, bevor die Eingangszustände bearbeitet werden. Hilfreich, wenn man mechanische Schalter verwendet, die ein Melderflackern auslösen könnten. Jeder Auslesevorgang hat ein delay von 30 ms. Werte: 1-128 Standardwert: 2
CV 33 (0x21)	Maskierung	Maskiert die Eingänge 1 - 8 und legt fest, ob diese bei einer Zustandsänderung eine Übertragung auf den Bus auslösen können. Die Eingänge werden ansonsten nur übertragen, wenn ein nicht maskierter Eingang geändert wird. Werte: 0 – 255 (Bitweise progremmierbar) Standard: 255 (alle werden gesendet)
CV 34 (0x22)	Maskierung	Wie CV 33, jedoch für Ports 9 - 16
CV 35 (0x23)	Bit-Timing	Timingverzögerungen der Bittakte auf dem CLOCK-Signal des S88-Schieberegisters. Standard 50µs (CV-Wert: 1)
CV 36 (0x24)	Lesezeitpunkt	Zeit-Quanta auf dem S88 Signal. (Optimal am Ende des Clockpegels, vor der fallenden Clockflanke (CV-Wert = CV35 -1)
CV 255 (0xFF)	Ende	Beendet den Programmiermodus und schreibt die neuen Konfigurationen in den Speicher. Danach Jumper ziehen und Modul neu starten. Befehl wird von Programmiersoftware ausgelöst, wenn man "Abschließen" bestätigt.

3. Programmierung (SpaßCAN Basic)

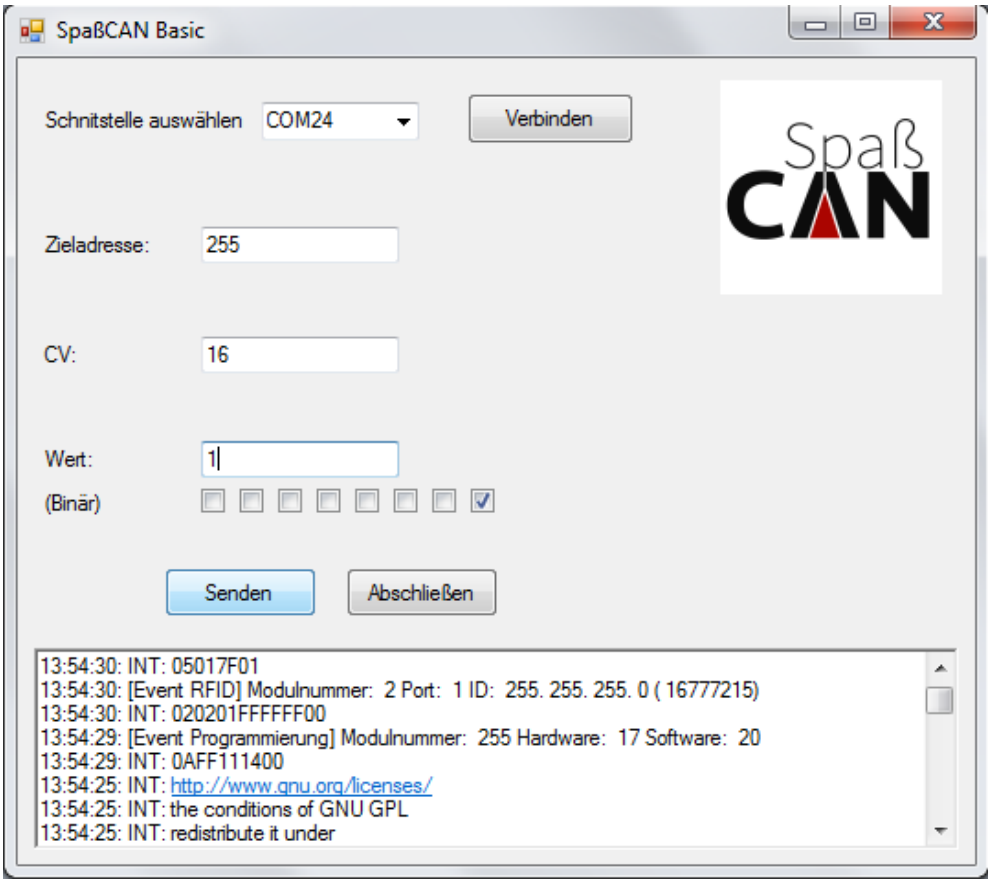
Die Programmierung der Busmodule geschieht über ein angeschlossenes Interface mit dem Tool ‚SpaßCAN Basic'. Das Programm kann auf www.spasscan.de heruntergeladen werden. Sofern bereits der Schnittstellentreiber für den Arduino Nano installiert ist (oder kein Treiber erforderlich ist), wird die serielle Schnittstelle im SpaßCAN Basic erkannt und kann ausgewählt werden (erfahrungsgemäß eine schwachsinnig hohe Portnummer). Mit einem Klick auf ‚Verbinden' meldet sich das Programm am Interface an, woraufhin das Interface mit einem Lizenztext der GNU GPL antwortet.



Jetzt zeigt SpaßCAN Basic alle Aktivitäten auf dem Bus an. Möchte man ein Modul programmieren, muss der entsprechende Programmierstecker geschlossen und im Anschluss an die Busversorgung eingeschaltet werden. Wird der Programmiermodus erkannt, meldet SpaßCAN Basic im Log:

```
[Event Programmierung] Modulnummer: 255 Hardware:17 Software 20
```

Also ist ein Modul mit der Adresse 255, der Hardware 1.7 und Softwarestand 2.0 im Programmiermodus am Bus angemeldet. Auf dem entsprechenden Modul blinken Info- und Status-LED im wechsel.



Sinnvollerweise sollte sich nur ein Modul je Adresse im Programmiermodus befinden.

Gemäß der CV-Tabellen der Melder kann man nun beginnen, die Konfigurationswerte der Busbausteine zu ändern (als erstes am besten die Adresse: CV16), indem man die akteulle Melder-ID als Zieladresse einträgt (diese ändert sich erst, wenn man die Programmierung per ‚Abschließen' bestätigt) und die zu ändernde CV mit einem neuen Wert sendet. Jede gewünschte CV muss einzend durch einen Klick auf ‚Senden' übertragen werden. Die CV-Werte können entweder dezimal oder binär eingegeben werden. Das macht für den übertragenden Wert keinen Unterschied, ist aber bei manchen CVs als Binärwert etwas anwenderfreundlicher.

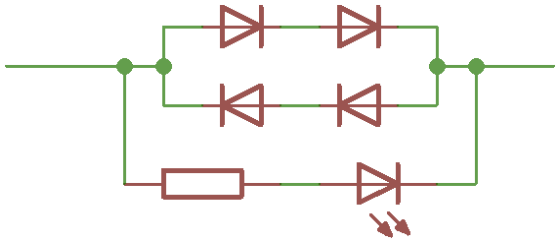
Sobald man mit den Einstellungen fertig ist, bestätigt man die Programmierung durch einen Klick auf ‚Abschließen' (es wird die CV 255 gesendet). Nun wird der Programmierjumper wieder entfernt und der Melder kann mit den neuen Einstellungen neu gestartet werden (Busversorgung trennen).

4. Rückmeldeschaltungen

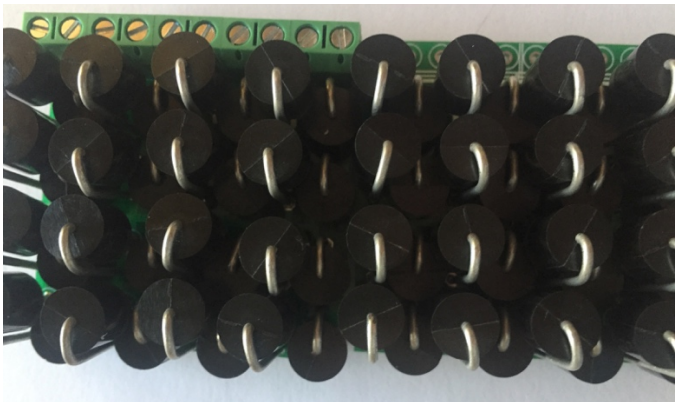
Eigentlich sind der Fantasie kaum Grenzen gesetzt. Ziel des Anschlusses der SpaßCAN Melder sollte immer sein, dass die LED im Optokoppler leuchtet (sieht man nicht). Außerdem sollte die LED auch nicht kaputt gehen, weshalb man einen Widerstand vorsehen sollte.

4.1 Die Diode

Ganz klassisch, kann man die Belegung seiner Gleise über den Stromverbrauch der Züge messen. Hierzu werden in jede Polaritätsrichtung je zwei Dioden zusammengeschaltet:



Bauartbedingt fällt je nach angelegtem Strom an der Diode eine gewisse Spannung ab, die dann durch die LED fließt und sie zum leuchten bringt. Da die abfallende Spannung an einer Diode typischerweise bei 0,7V liegt, benötigt man zwei in serie, um die LED zu betreiben. Aus diesem Konzept ist eine handliche Platine entstanden, die Dioden mit bis zu 6 Ampere Dauerstromfestigkeit aufnehmen können.



Je nach Melder ist ein Widerstand zwischen 60Ohm und 100 Ohm bereits auf der Melderplatine verbaut. Daher kann man einfach frei drauflosverkabeln. Wie auch die Melder, sind die Dioden zu vier Ausgängen zusammengeschlossen, mit einer gemeinsamen Zuleitung.

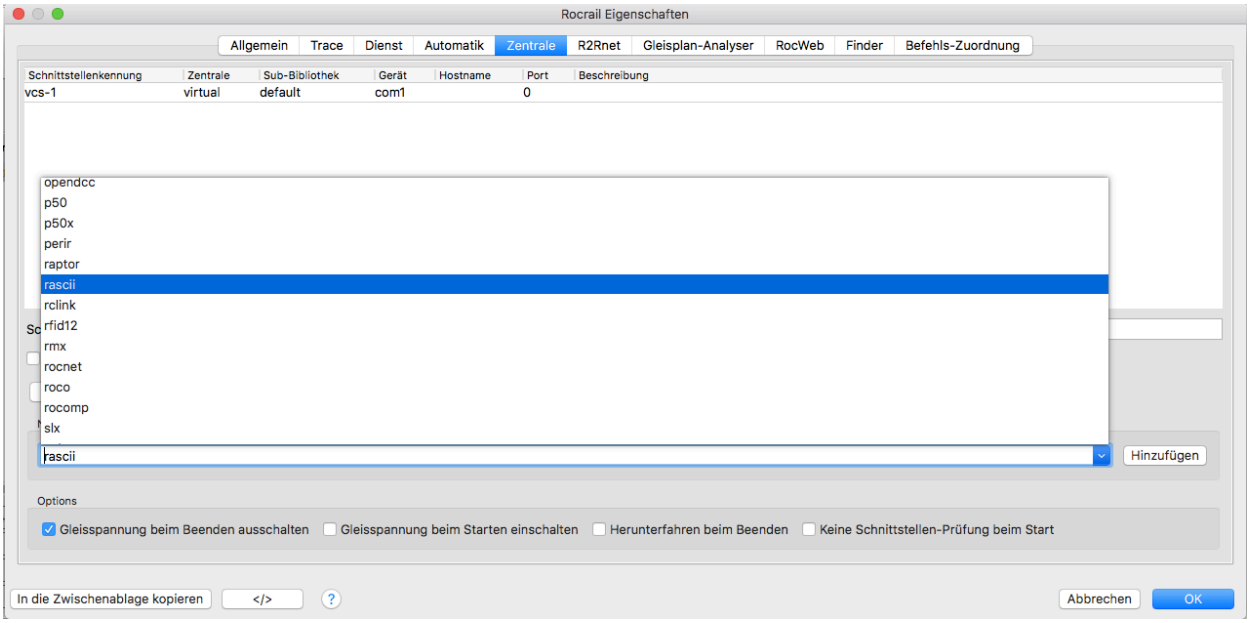
5. RocRail

Als Steuerungssoftware kann derzeitig für den vollen Funktionsumfang ausschließlich RocRail genutzt werden.

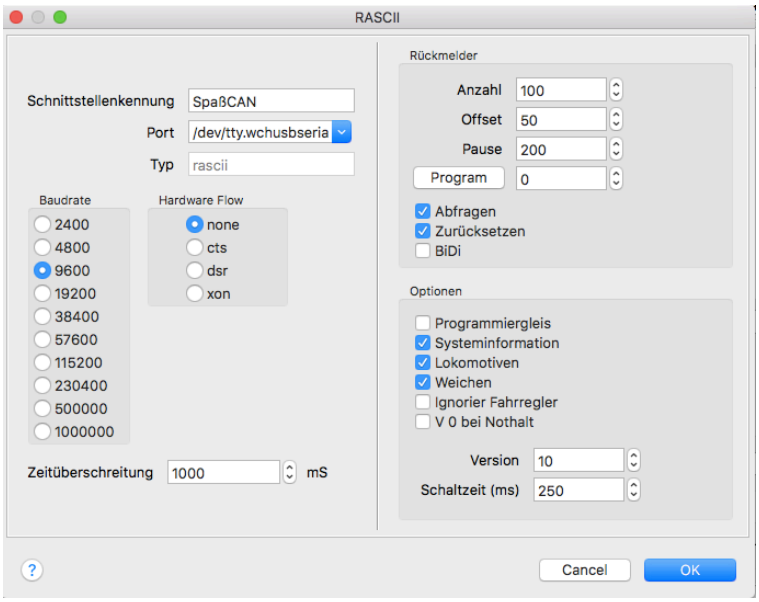
5.1 Rocrail SpaßCAN-Einrichtung

Das SpaßCAN Interface wird in RocRail als zusätzliche Zentrale in den RocRail-Eigenschaften eingerichtet. Hierfür muss der entsprechende Schnittstellentreiber installiert sein.

Datei -> RocRail Eigenschaften
Tab: Zentrale



Neue Zentrale vom Typ ‚rascii' hinzufügen



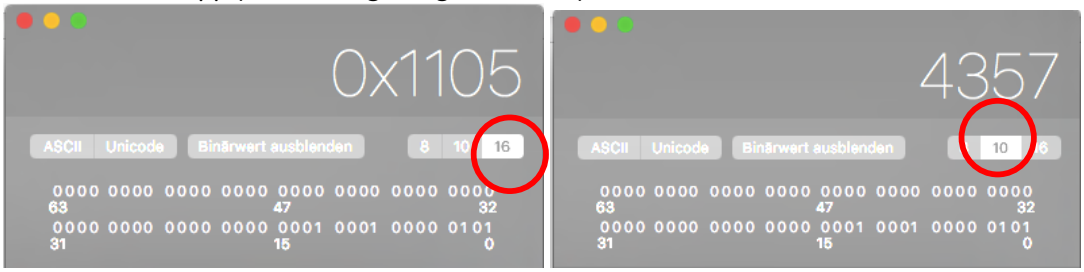
In den Eigenschaften der neuen Zentrale den korekten Port des Interfaces auswählen (bei Windows ein COM-Port) und Baudrate auf 9600bd sowie Hardwarehandshakes auf ‚none' stellen.
Bestätigen mit ‚OK' und im Anschluss RocView und RocRail Server neu starten.

5.2 RocRail: Rückmelder einrichten

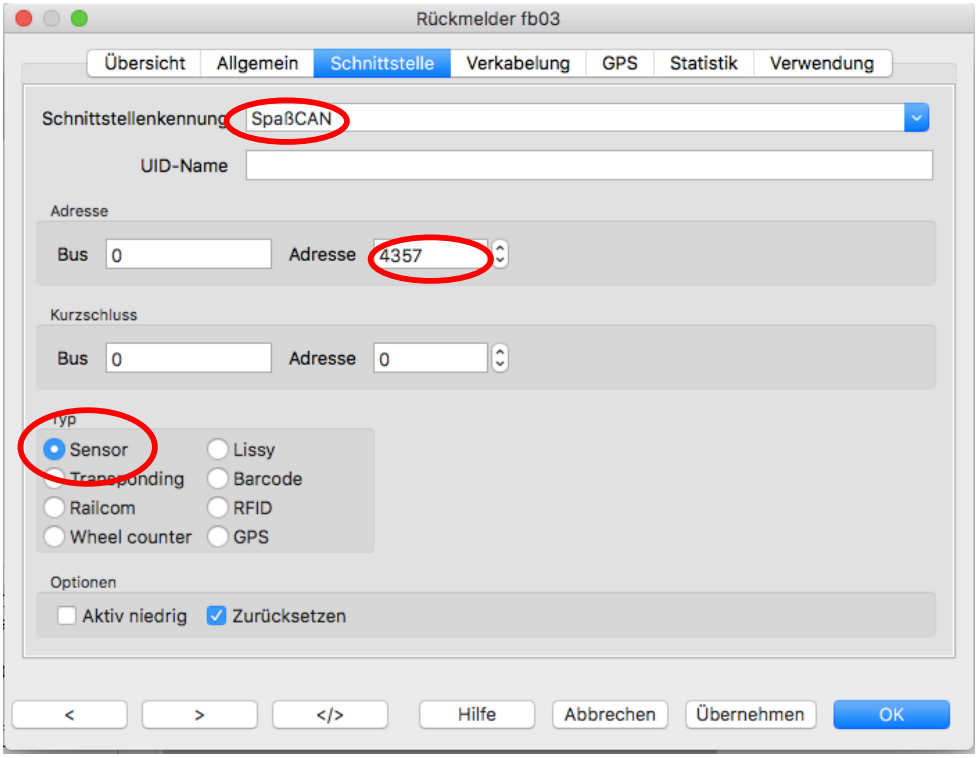
Derzeitig werden die Rückmelderadressen in zwei byte-Länge gemeldet. Hierbei entspricht die Portnummer dem unteren Byte und die Moduladresse dem oberen Byte.
Wird ein Rückmeldeereignis ausgelöst, wird in Rocrail die Bytefolge: 05 MM PP SS in hexadezimal ausgegeben. MM: Moduladresse (über CV16 zu festzulegen). PP: Portnummer (Anschluss am Rückmelder) SS: Status (00 oder 01)

In den Einstellungen eines Rückmeldekontaktes in Rocrail wird unter dem Tab ‚Schnittstelle' nun die Adresse des Ports aus den Werten MMPP eingetragen. Hier allerdings in dezimalschreibweise.

Hilfe: Rechner-App (Darstellung: Programmierer) Umrechnen zwischen Hex und Dez



Beispiel: Moduladresse: 17 (0x11). Portnummer: 5 (0x05) – Rückmelderadresse: 4357



Die Schnittstellenkennung wird auf die zuvor eingerichtete rascii-Zentrale gestellt und der Rückmeldekontakt als Sensor bzw. RFID konfiguriert.