

0 - Disclaimer und Vorwort

Alle nachstehenden Informationen sind auf der Basis eigener Erfahrungen mit meiner Märklin HO zusammengetragen und stellen nur einen winzigen Bruchteil möglicher Lösungen dar. Aber ich möchte alles nachvollziehbar dokumentieren.

Selbstverständlich kann ich dafür keine Gewährleistung bzw. Haftung übernehmen. Ich habe daher in den nachstehenden Ausführungen bei der Anbindung des Raspberry Pi alternativ zur Modifikation der Gleisbox auch die CAN-Bus-Anbindung über eine Steckverbindung mit aufgenommen (diese Variante habe ich jedoch noch nicht praktisch getestet).

Aber auch hier gilt: Verpolung der CAN-Bus-Verdrahtung kann zur Schädigung bzw. Zerstörung der Hardware führen.

Das nachstehende Dokument ist aus meiner OneNote-Sammlung zum Thema entstanden. Für daraus resultierende vielleicht ungewöhnliche Formatierungen bitte ich um Nachsicht.

So schön sich ein System aus Hardwarekomponenten auch zusammenfügt:

Auf die Software kommt es im Wesentlichen an!

An dieser Stelle meinen herzlichen Dank an Gerd Bertelsmann (can2lan) und Teddy (RailControl), die mit ihrer ganz **wesentlichen Vorarbeit** und ihren lizenzfreien Programmen dieses Projekt überhaupt erst zum Laufen bringen!

Deshalb an dieser Stelle einige für mich sehr wichtige Verweise auf Webseiten und GIT-Projekte:

<https://modellbahn.mahrer.net/>

<https://www.railcontrol.org/forum/>

<https://github.com/CANGuru-System>

<https://github.com/GBert>

<https://github.com/CANGuru-System>

<http://can-digital-bahn.com/forum/index.php>

<https://stummiformum.de/>

<http://www.skrauss.de/modellbahn/gboxcan.html>

:

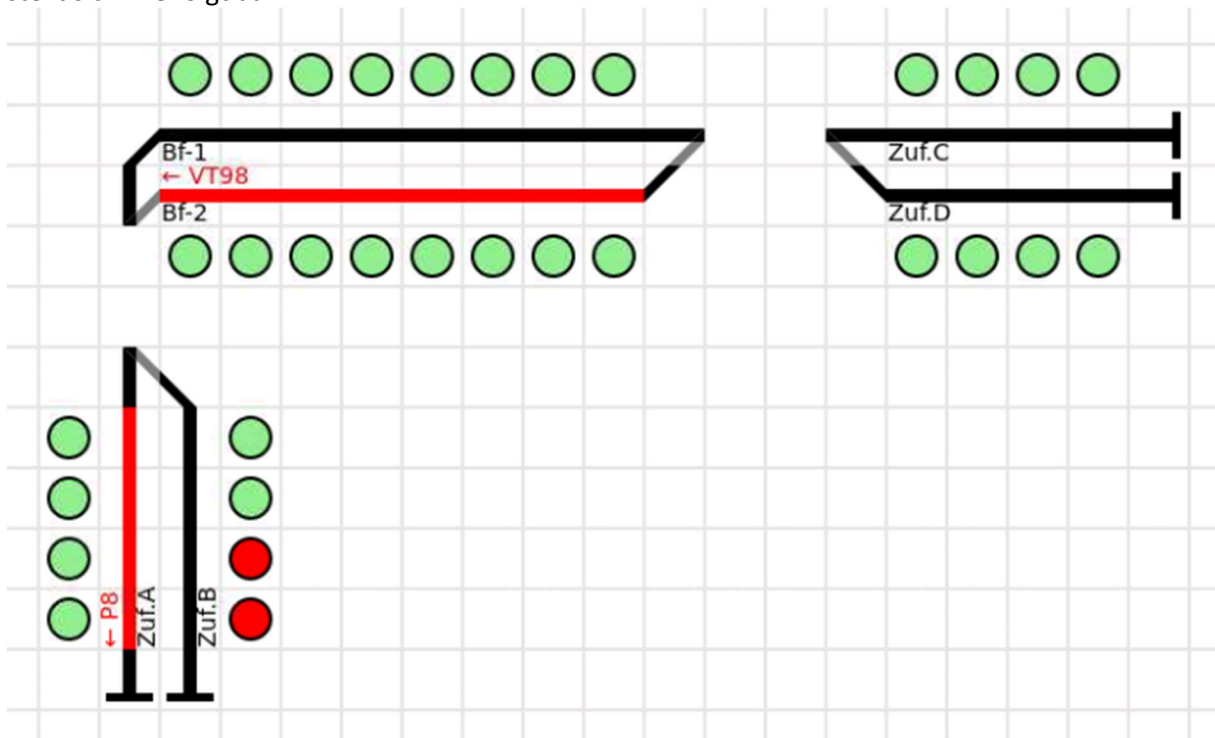
1 - Ziel der Dokumentation

Aus der schier unendlichen Auswahl von Kombinationen von Hard- und Softwarekomponenten soll einem Anfänger auch mit kleinem Geldbeutel die **einfache Möglichkeit** eröffnet werden, seine Modellbahn zu automatisieren - die Digitalisierung auch in diesem Bereich nachvollziehbar und vor allem nutzbar zu machen.

Ich habe daher meine Anlagenkonfiguration beim neuen Aufsetzen meines Raspberry Pi (hoffentlich) hinreichend detailliert dokumentiert.

Meine eigene Motivation zu diesem Aufbau war die Automatisierung des Fahrbetriebes meiner Testanlage und die Gewinnung von Erfahrungen zum Aufbau einer kleinen/mittleren digitalisierten Modellbahnanlage. Bei kleinem Budget.

Meine Testanlage 'immer an der Wand lang' mit vielfältigen Fahrmöglichkeiten im Mehrzugbetrieb stellt sich wie folgt dar:



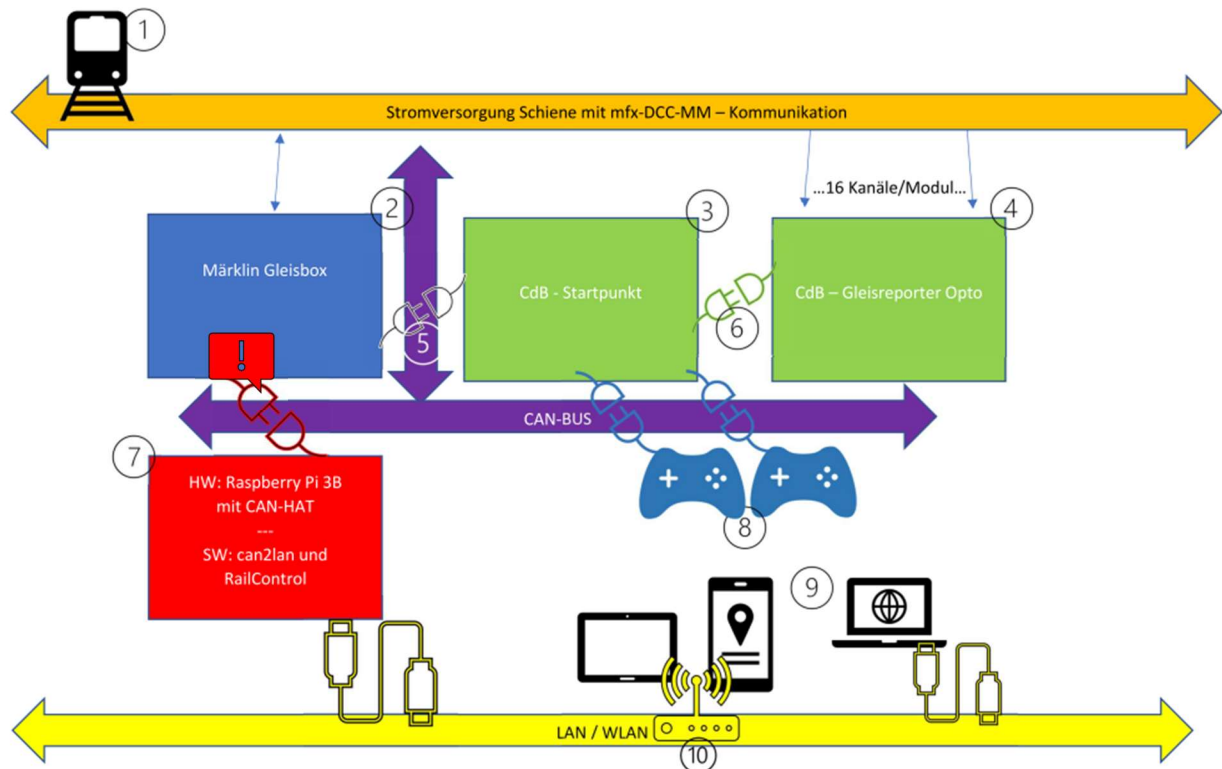
Die im Wesentlichen zu klärenden Fragen hierbei sind:

Welche Hardware wird benötigt (und mit welchen Kosten ist zu rechnen)?

Welche Software wird benötigt (und mit welchen Kosten ist zu rechnen)?

Kann ich mir die Verkabelung des CAN-Busses und Installation der Software zumuten?

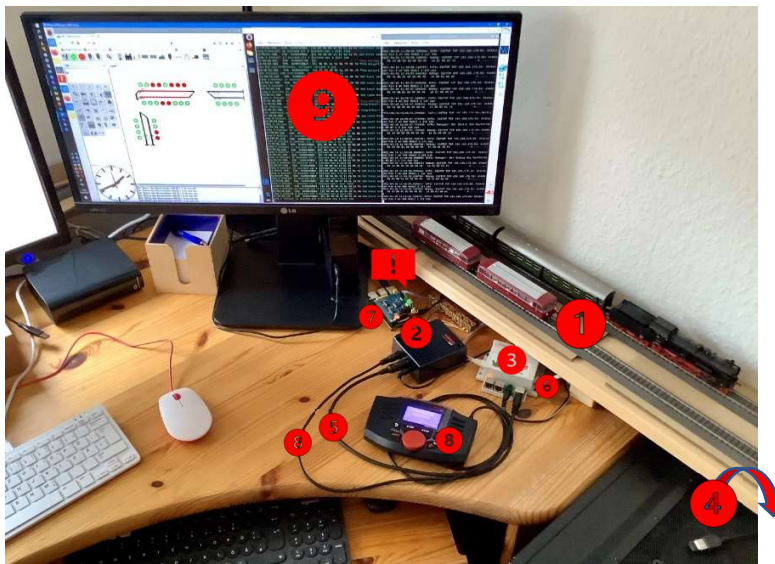
Praktisch sieht die Zusammenstellung der Komponenten dann z.B. so aus:



Systemübersicht RailPi

Legende:

- 1 – Gleise und Rollmaterial
- 2 – Märklin Gleisbox 60116
- 3 – CdB Startpunkt 2 – Art.Nr. 160302
- 4 – CdB Gleisreporter Opto – Art.Nr. 110106 mit Netzteil Art.Nr. 410009
- 5 – CdB Elektronik Anschlusskabel Art.Nr. 410007
- 6 – Ethernet Patchkabel zur Verbindung von Startpunkt und Gleisreporter
- 7 – Raspberry Pi 3B mit Waveshare 17912 2-CH CAN HAT
- ! – Märklin Ersatzanschlußkabel E146781 für MS2
- 8 – bis zu 2 MS2 in dieser Konfiguration (an den Startpunkt2) anschließbar
- 9 – Browserfähige Endgeräte (PC's – Notebooks – Tablets – Smartphones etc.)
- 10 – vorhandene Netzwerkinfrastruktur (LAN-/WLAN-Router)



Diese Doku wurde also auch dazu erstellt, damit auch ein Anfänger bzw. Neueinsteiger in die Digitaltechnik ungefähr abschätzen kann, was da auf ihn zukommt.

Dabei habe ich Wert darauf gelegt, daß dieses Dokument allein genügen muß, um zum beschriebenen Ziel zu führen.

Über Rückfragen aber auch Fehlerberichtigungen/Korrekturen freue ich mich gleichermaßen. Hoffentlich wird es keine Bauchlandung ;) - bei mir läuft's jedenfalls!

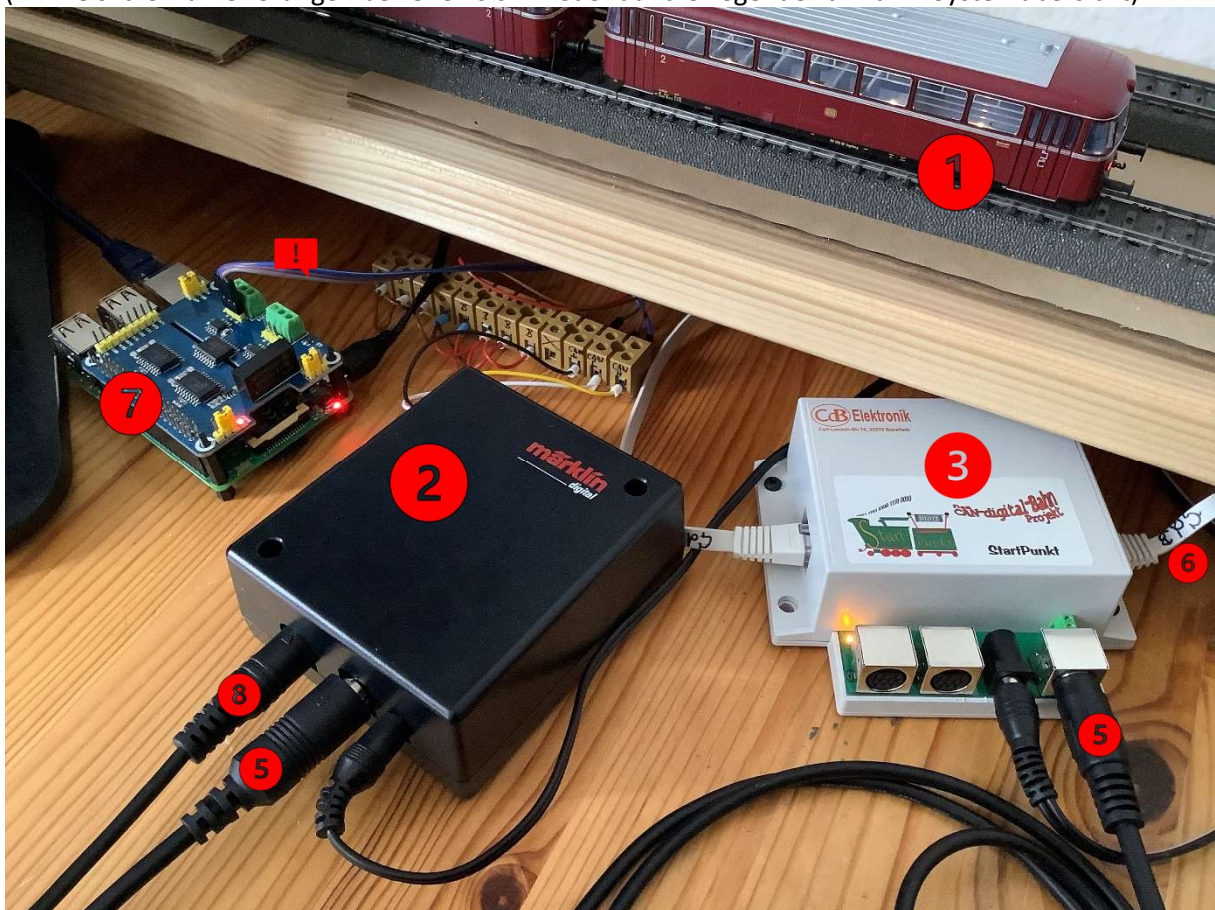
Auf geht's!

2 - Hardware - nach Lieferanten sortiert

Auf der nachstehenden Abbildung sind folgende Komponenten zu sehen:

- Mitte:
die Gleisbox von Märklin, daran angeschlossen das passende Netzteil, der StartPunkt und eine Mobile Station 2 von Märklin.
- Rechts:
der StartPunkt von CdB Elektronik - an den rechts und links sichtbaren weissen LAN-Kabeln (mit CdB von Hand beschriftet) lassen sich über den CdB-Systembus weitere CdB-Module anschließen. In diesem Fall sind das 2 Gleisbesetzmeldemodule GleisReporter Opto
Weiterhin sind ein Netzteil zur Stromversorgung sowie unten rechts das Verbindungskabel zur Gleisbox (CAN-Bussignale). Zwei weitere Buchsen zum Anschluß von CAN-Geräten nach Märklin-Standard sind noch frei - z.B. für weitere MS2
- Links:
der Raspberry Pi als RailControl-Leitrechner, RailControl-Webserver und CAN-Signalumsetzer von CAN auf LAN

(Hinweis: die Numerierungen beziehen sich wieder auf die Legende zur RailPi-Systemübersicht)





2.1 - Märklin

Ohne Märklin geht's gar nicht, denn immerhin ist die Gleisbox sozusagen der tragende Pfeiler im Konzept und bei der Modellbahnausrüstung.

Ist digitales Rollmaterial sowie das C-Gleis bereits vorhanden, könnte - wie es bei mir der Fall war - auch bei Euch das Einstiegs-Set Märklin Art.Nr. 29000 in Frage kommen:

Spur H0 - Art.Nr. 29000

Modelleisenbahn Digitaler Einstieg. 230 Volt

The image shows the packaging for the Märklin digital 29000 starter set. The box is white with red and black text. It features the Märklin logo and the word 'digital'. Text on the box includes 'Idealer Einstieg in die digitale Welt von Märklin H0', '12 gebogene Gleise', '4 gerade Gleise', 'Gleisanschlussbox', 'Schaltnetzteil', and 'Mobile Station'. A small image of the digital decoder is also visible.

169,99 €
inkl. MwSt. **Versandkosten**

● Lieferzeit 2-3 Werktage

1 ▾

In den Warenkorb

⚖ Vergleichen 📌 Merken

Artikel-Nr.: 29000

Versandkostenfrei innerhalb Deutschland ab 30,00 €

Beschreibung

Ersatzteile und Downloads

Inhalt: 12 gebogene Gleise 24230, 4 gerade Gleise 24188, 4 gerade Gleise 24172. Gleisanschlussbox, Schaltnetzteil 36 VA/230 V und Mobile Station für den digitalen Einstieg. Illustriertes Spielbuch mit vielen Tipps und Anregungen. Erweiterungsmöglichkeiten mit den C-Gleis-Erweiterungspackungen und mit dem gesamten C-Gleis-Programm.

Highlights

- Idealer Einstieg in die digitale Welt von Märklin H0.
- Aufbaufreundliche C-Gleis-Anlage.
- Gleisoval Radius R2.

Stand 04/2021 - Märklin Shop:

<https://www.maerklinshop.de/maerklin/spur-h0/packungen/startpackungen/55320/digitaler-einstieg.-230-volt>

2.2 – Welectron

Die Fa. Welectron liefert nicht nur den Raspberry Pi (eine der weiteren tragenden Säulen) sondern auch das CAN-Interface, den sogenannten CAN HAT.

Die nachstehende Auswahl bezieht sich nur auf den Raspberry Pi - es wird vorausgesetzt, daß ein PC-Netz (LAN oder WLAN) am Aufstellungsort vorhanden ist.

Ist kein PC oder Tablet vorhanden, ist der Raspberry Pi zusätzlich mit Tastatur, Maus und Monitor auszurüsten. Das wäre aber eher unwahrscheinlich, oder?

Artikel		Menge	Einzelpreis	Preis	
	Raspberry Pi 3 Modell B	1 ▾	60,00 €	60,00 €	
	ArtikelNr.: 402033-002 Lieferzeit: 1 - 3 Werktage Einplatinen-Computer, 4x1,2 GHz, 1 GB RAM, 4x USB, 40-Pin GPIO, CSI (Kamera-Port), DSI (Display-Interface), LAN, WLAN, Bluetooth <ul style="list-style-type: none">• 1x 0,10 € Raspberry Pi Pinout Karte• 1x 1,25 € Kühlkörpersatz• 1x 4,90 € SanDisk Ultra microSD Speicherkarte 16 GB• 1x 6,95 € Offizielles microUSB Netzteil (EU)• 1x 12,90 € HighPi Raspberry Pi 3 Gehäuse transparent				
	Waveshare 17912 2-CH CAN HAT	1 ▾	18,90 €	18,89 €	
	ArtikelNr.: WS1-017912 Lieferzeit: 1 - 3 Werktage 2-Channel Isolated CAN Bus Expansion HAT for Raspberry Pi, MCP2515 + SN65HVD230 Dual Chips Solution, Multi Onboard Protection Circuits				
				inkl. 19% MwSt.:	12,60 €
				Gesamtsumme:	78,89 €

(für 1,11 EUR mehr wäre diese Bestellung versandfrei - das wäre dann mit zusätzlicher Bestellung 2er Patchkabel - eines für die LAN-Anbindung des Raspberry Pi und eines für die Verbindung zwischen StartPunkt2 und GleisReporter Opto (CdB-Systembus) auch gegeben). Die Längen sollten sich nach den Erfordernissen richten - so kurz wie notwendig!



goobay®

Artnr.: 409332-010

Goobay 68693 Ethernet-Patchkabel 1,0m CAT6

CAT 6 Patchkabel, S/FTP (PiMF), schwarz, LSZH halogenfrei, Kupfer

Länge

- ☐ 0,81 € 0,25m CAT6
- ☐ 0,80 € 0,5m CAT6
- ☒ 1,00 € 1,0m CAT6
- ☐ 1,50 € 1,5m CAT6
- ☐ 1,90 € 2,0m CAT6
- ☐ 2,90 € 3,0m CAT6
- ☐ 3,90 € 5,0m CAT6
- ☐ 6,90 € 10m CAT6
- ☐ 11,90 € 20m CAT6
- ☐ 17,90 € 30m CAT6
- ☐ 24,90 € 50m CAT6
- ☐ 5,50 € 5,00m CAT6a/CAT7
- ☐ 8,90 € 10m CAT6a/CAT7
- ☐ 14,90 € 20m CAT6a/CAT7

2.3 - CdB-Elektronik

Bis zu diesem Punkt wäre die Hardware-Auswahl bereits geeignet, um einen komfortablen Handbetrieb über ein Smartphone, Tablet oder PC zu erlauben.

Als Lieferant vielfältiger Module rund um den CAN-Bus haben wie hier eine weitere Säule des Konzeptes vor uns, wenn mehr als nur komfortabler Handbetrieb an einem browserfähigen Endgerät erfolgen soll.






Denn für einen zugesteuerten Automatikbetrieb über Fahrstraßen braucht es **mindestens** auch Gleisbesetzmelder. Und hier tut sich ein ganzes Füllhorn an interessanten Modulen auf...

Homepage: <http://can-digital-bahn.com/news.php>

Shop: <https://www.spielzeug-meiners.de>

Hier ein beispielhafter Warenkorb, um einen Gleisbesetzmelder mit der Märklin Gleisbox zu verbinden

Stand 04/2021

Can-Digital Bahn und weitere				
Artikel-Nr.	Bezeichnung	Einzelpreis	Anzahl	Gesamtpreis
 410007	CdB Elektronik Anschlusskabel 0,6m CAN-digital-Bahn Lieferzeit: 2-4 Werktage ⁽¹⁾	12,00 €	<input type="text" value="1"/>	12,00 €
 410009	CdB Elektronik Netzteil 12V DC 1,5A CAN-digital-Bahn Lieferzeit: 2-4 Werktage ⁽¹⁾	17,00 €	<input type="text" value="1"/>	17,00 €
 110106	CdB Elektronik GleisReporter Opto GleisReporter Opto Can digital Bahn Lieferzeit: 2-4 Werktage ⁽¹⁾	69,00 €	<input type="text" value="1"/>	69,00 €
 160302	CdB Elektronik Startpunkt 2 Can-digital-Bahn Startpunkt 2 Lieferzeit: 2-4 Werktage ⁽¹⁾	29,95 €	<input type="text" value="1"/>	29,95 €
 410015	CdB Elektronik CAN digital Bahn Buchse für Lieferzeit: 2-4 Werktage ⁽¹⁾	7,95 €	<input type="text" value="1"/>	7,95 €
Warenwert:				135,90 €

Und hier gibt's die detaillierten Infos zum StartPunkt:

http://can-digital-bahn.com/modul.php?system=sys3&modul=47#Mod_Top (link zum Modul 'StartPunkt2')

Soll eine größere Anlage versorgt werden, so kann zusätzlich auf den ModulBooster2 zurückgegriffen werden.

Bitte informiert Euch hierzu auf http://can-digital-bahn.com/modul.php?system=sys3&modul=96#Mod_Top

Eigene Erfahrungen mit diesem Produkt liegen mir bislang noch nicht vor.

Erste Rückmeldung von Eurer Seite:

Im Lokmuseum Online Shop gibt es das MS2-Ersatzanschlußkabel E146781 von Märklin für 12,90 EUR

[Märklin E146781](https://www.lokmuseum.de/shopartikel.php?SWg1=Spur%20H0%20M%E4rklin&SWg2=Ersatzteile&SArt=10020587&SBez=E146781%20M%E4rklin%20Kabel%20m.Stecker%20u.%20Zugentlastung1)<https://www.lokmuseum.de/shopartikel.php?SWg1=Spur%20H0%20M%E4rklin&SWg2=Ersatzteile&SArt=10020587&SBez=E146781%20M%E4rklin%20Kabel%20m.Stecker%20u.%20Zugentlastung1> Kabel m.Stecker u. Zugentlastung Onlineshop lokmuseum.de Spur H0 Märklin

Tip:

Wenn man vorsichtig ,am freien Ende (auf dem Foto links) die Laschen mit einer Nadel hoch kippt, dann kann man die gekrimpten Enden direkt am CAN-Hat verschrauben.

LOKMUSEUM ONLINE SHOP

Jetzt registrieren und zusätzliche Funktionen nutzen!

Anmelden Login

Ihre Auswahl

Teile: 0 Summe: 0,00 EUR

Kasse Warenkorb

Artikelsuche

in Alle Bereiche

Suchen

ONLINE SHOP

- Fischertechnik
- Weitere Hersteller
- Spur G LGB
- Spur 1 Märklin
- Spur H0 Auhagen
- Spur H0 Fleischmann
- Spur H0 NOCH
- Spur H0 Märklin
- Dampflokomotiven
- Diesellokomotiven
- E-Lokomotiven
- Triebwagen
- Zugpackungen
- Personenwagen
- Güterwagen
- K-Gleismaterial

Artikel: E146781 Kabel m.Stecker u. Zugentlastung

Beschreibung

E146781 Märklin Kabel m.Stecker u. Zugentlastung

Kabel m.Stecker u. Zugentlastung

Nur für Erwachsene

Details und Daten

E146781 Märklin Kabel m.Stecker u. Zugentlastung

Spurweite	Spur H0 1:87
Kategorie	Spur H0 Märklin > Ersatzteile
Hersteller	Märklin
Artikelnummer	E146781
Hersteller Preis	21,00 EUR

Neuware vom Fachhändler

Neuware mit gesetzlicher Gewährleistung 14 Tage Widerrufsrecht.

Preis und Warenkorb

Inkl. 19% MwSt **12,90 EUR**

Staffelpreise		
ab 3 Stück	ab 6 Stück	ab 12 Stück
12,50 EUR	12,30 EUR	11,95 EUR

Zuzügl. 2,80 EUR Versand innerhalb Deutschland

Versand in die EU: 8,50 EUR Weltweit: 29,50 EUR

Artikel ist bestellt aber noch nicht am Lager! Liefertermin noch unbekannt! Vorbestellung/Reservierung möglich!

Artikel vorbestellen > Menge 1

Damit würde dann das 2. CdB-Anschlußkabel 0,6 m (410007) sowie der alternativ notwendige CdB Stecker (410015) entfallen.

→ s. nächste Seite, Kapitel 2.4 manuell notwendige Anpassungen (Varianten 2 und 3)

2.4 - Manuell notwendige Anpassungen

Es handelt sich bei dem vorgestellten Aufbau grundsätzlich um (leicht bestellbare) Fertigkomponenten.

Lediglich bei der Montage des Raspberry Pi's (Montage der Platinen in ein Gehäuse) und der Anbindung der CAN-Schnittstelle an die Gleisbox ist ein klein wenig Handarbeit - und Aufmerksamkeit! - gefordert.

Anbindung des Raspberry Pi an die CAN-Schnittstelle der Gleisbox:

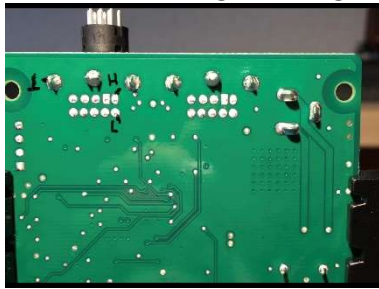
Variante 1 - wie schon im Übersichtsfoto zum Kapitel Hardware eingangs gezeigt - Lötarbeit notwendig

(Märklins Gewährleistung bzgl. der Gleisbox erlischt!)

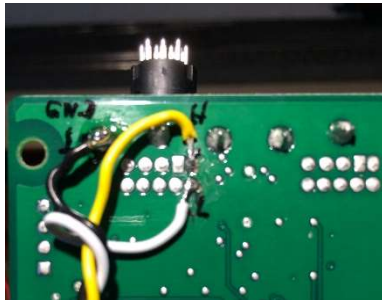
hier noch Detailfotos der modifizierten Gleisbox - würde ich zukünftig durch Variante 3 ersetzen wollen

Der im Foto zu sehende Stecker für die Gleisbox wird erst in Variante 2 interessant - hier ist er nicht notwendig

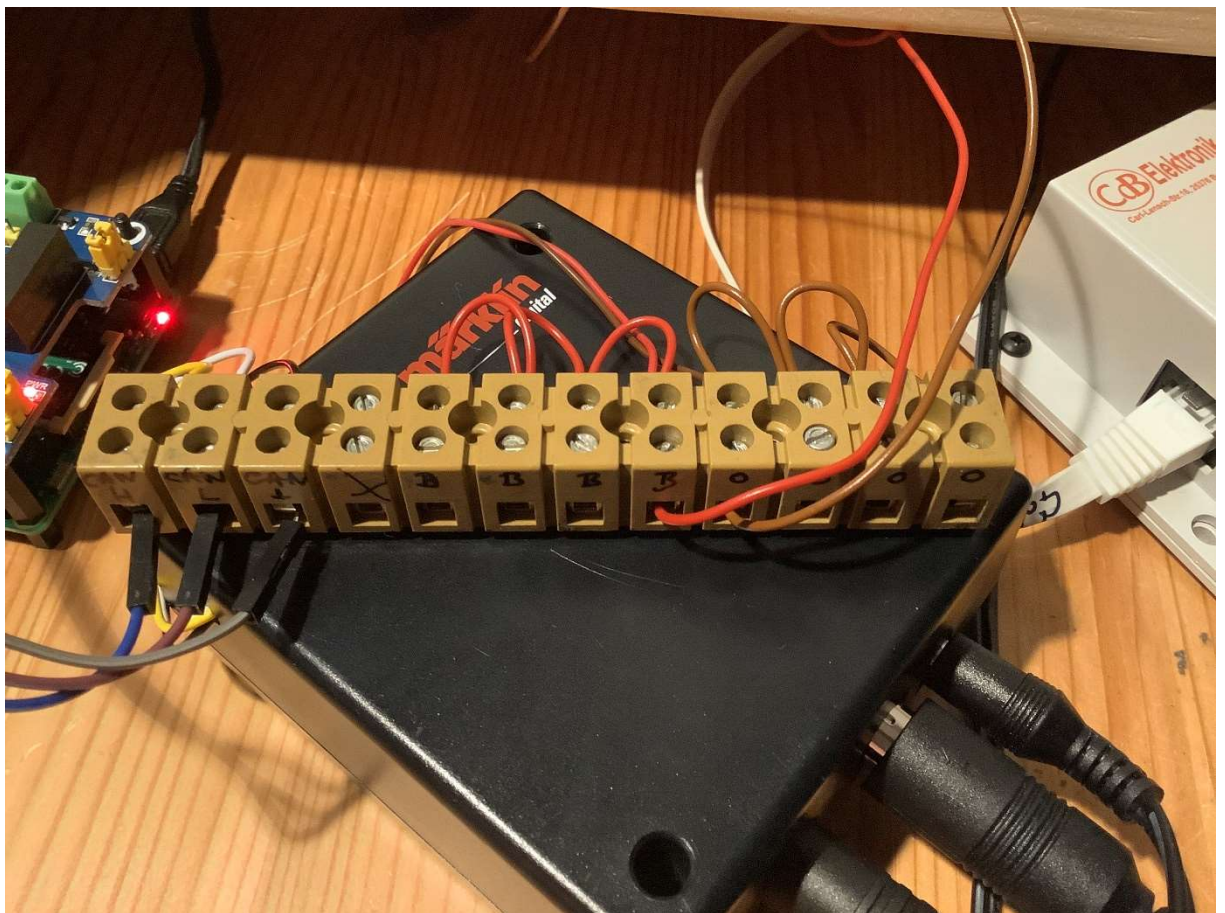
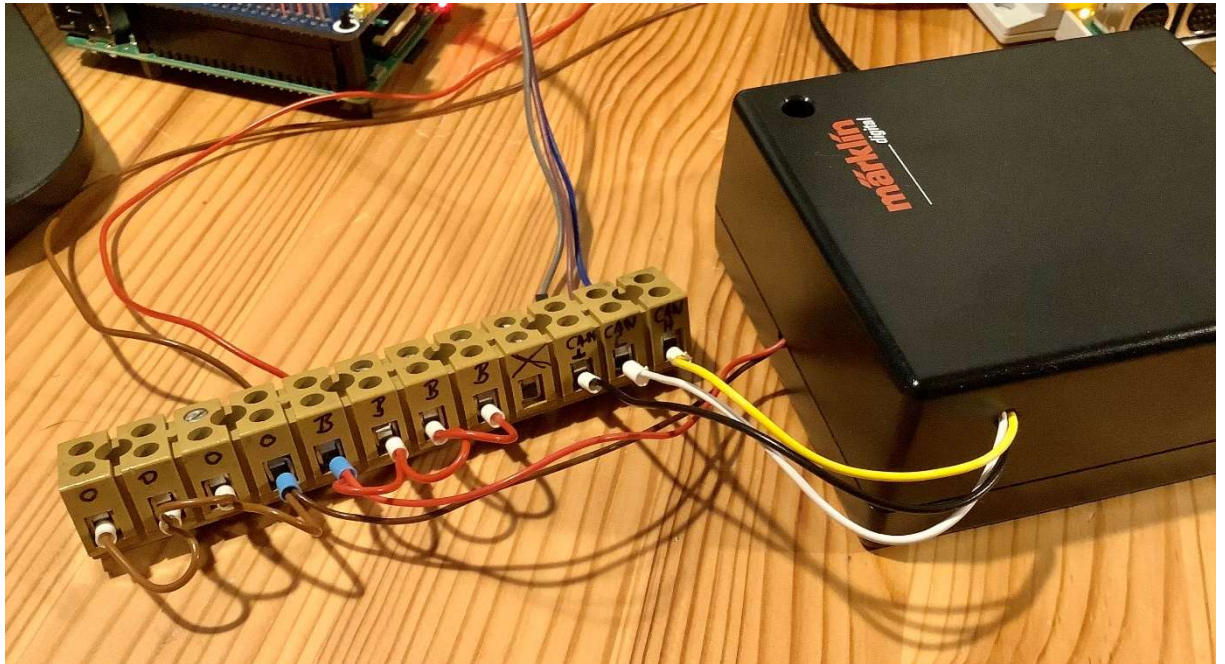
Vorher - die richtigen Lötäugen markiert: H(igh) - L(ow) - GND



Nachher - die Kabel verlötet zum Anschluß an einen CAN-Controller - in unserem Fall den CAN-HAT



und schließlich die CAN-Bus-Adern (Ge/Ws/Sw) zusätzlich zur Schienenstromversorgung (Rt/Bn) nach außen geführt:



Variante 2 - mit der 'CdB Art.Nr. 410015 - Buchse für Gleisbox' ein eigenes Kabel herstellen (fummelige Arbeit!, löten notwendig)



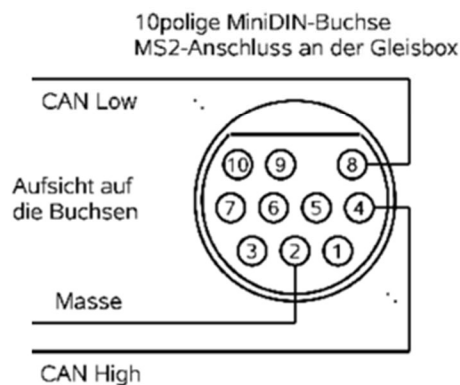
In rot angedeutet: die steckbare CAN-Verbindung zwischen Gleisbox und Raspberry Pi CAN-HAT der Varianten 2 und 3

Variante 3 - mein heutiger Favorit, wenn keine Lötarbeit vorgenommen werden soll - das Anschlußkabel 0,6m zwischen Gleisbox und StartPunkt2 muss sowieso gekauft werden. Werden sofort 2 Stück Kabel CdB Art.Nr. 410007 gekauft, läßt sich das eine Kabelende durch Abschneiden des 'Nicht-Gleisbox-Steckers' zum StartPunkt und Freilegen der notwendigen Kabeladern wunderbar auf die CAN-Klemme des CAN-Hats am Raspberry Pi auflegen.

Hier ein passender Auszug aus der Webseite von Stefan Krauß
<http://www.skrauss.de/modellbahn/gboxcan.html>

Das Tiny-CAN-Interface hat wie viele andere auch einen 9-poligen Sub-D-Stecker, dessen Anschlussbelegung genormt ist (CiA DIS-102).

Für den Anschluss an die Gleisbox werden nur drei Signalleitungen benötigt: CAN High, CAN Low und Masse. Diese müssen nach dem folgenden Schema mit den entsprechenden Pins der Gleisbox verbunden werden. Es ist egal, welche der beiden Buchsen, die eigentlich für den Anschluss einer MobileStation 2 gedacht sind, man verwendet.



VORSICHT!

Um sicher zu sein, daß die richtigen Adern freigelegt und auf die Klemmen des CAN-HATs aufgelegt werden, sollte mit einem Messgerät am modifizierten Kabel der Durchgang geprüft werden. Diese Arbeit muss sorgfältig ausgeführt werden, da bei falschem Auflegen der Adern die angeschlossenen Komponenten Schaden erleiden können.

Sehr lesenswert ist auch die Arbeit von Jörg Pleumann 'railuino':

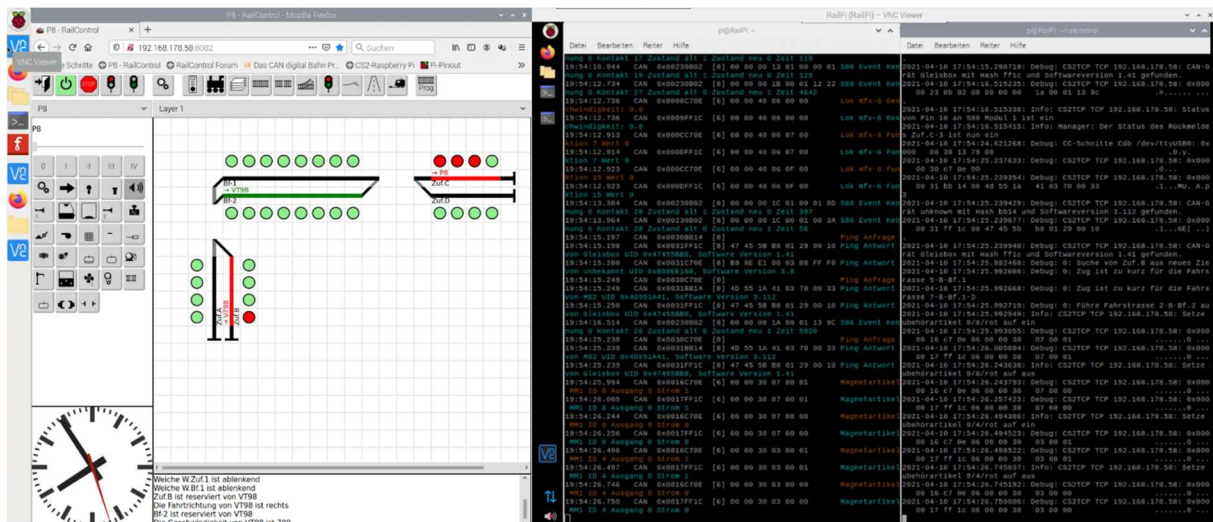
<https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/railuino/2013-08-01-Railuino-Directors-Cut.pdf>

Hier kann auch noch einmal die Steckerbelegung verifiziert werden - auch sonst ein 'must-read', wenn auch ein Arduino die Teilaufgabe unseres Raspberry Pi übernimmt. Das Prinzip dahinter ist gleich oder doch sehr ähnlich.

3 – Software

Die auf dem Bildschirm sichtbaren Fenster (von rechts nach links) zeigen die verwendeten Komponenten:

- Rechts:
Die laufende Ausgabe von RailControl im Terminalfenster auf dem Raspberry Pi (RailControl dient auf diesem Raspberry als
- Leitrechner für die Modellbahn
- Webserver für angeschlossene Geräte zur Visualisierung des Anlagenzustandes und Benutzereingaben
- Interface zur Zentrale (hier Gleisbox über die CAN2LAN-Software --> CS2-Emulation)
- Mitte:
Die Ausgabe von CAN2MONITOR im Terminalfenster auf dem Raspberry Pi
- Links:
der aktuelle Anlagenzustand auf einem beliebigen Browser im Netzwerk (also auch auf dem vorgangs erwähnten Raspberry Pi oder irgendeinem Smartphone, Tablet oder sonstigem PC).



3.1 - Installation von Pi OS

Kleine Vorbemerkung: ich installieren einen Raspberry Pi letztlich als Leitrechner für RailControl.
Daher nenne ich diesen Pi ab sofort nur noch RailPi (ist kürzer).

Auf <https://www.raspberrypi.org/software/> finden wir den 'Raspberry Pi Imager' - ein Hilfsprogramm, womit wir unter Windows / macOS / Ubuntu und selbstverständlich vom Raspberry Pi aus eine microSD-Karte mit einem ausgewählten PiOS-Umfang bespielen können, um danach mit dieser SD-Karte den Raspberry Pi zu starten.

Installieren von Raspberry Pi OS mit Raspberry Pi Imager

Raspberry Pi Imager ist die schnelle und einfache Möglichkeit, Raspberry Pi OS und andere Betriebssysteme auf einer microSD-Karte zu installieren, die mit Ihrem Raspberry Pi einsatzbereit ist. [Sehen Sie sich unser 45-Sekunden-Video](#) an, um zu erfahren, wie Sie ein Betriebssystem mit Raspberry Pi Imager installieren.

Laden Sie Raspberry Pi Imager auf einen Computer mit einem SD-Kartenleser herunter und installieren Sie es. Legen Sie die SD-Karte, die Sie mit Ihrem Raspberry Pi verwenden, in den Reader und führen Sie Raspberry Pi Imager aus.

Download für Windows

[Download für macOS](#)

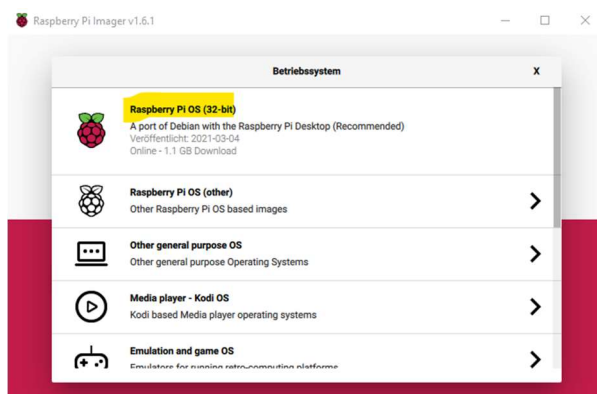
[Download für Ubuntu für x86](#)

Um auf **Raspberry Pi** OS zu installieren, geben Sie ein Terminalfenster ein. `sudo apt install rpi-imager`



Eigentlich ist die Nutzung des Programms ganz einfach:

1. 16 GB micro-SD-Karte mit dem PC verbinden
2. Rpi-Imager herunterladen (startet auf Wunsch direkt nach der Installation)
3. Umfang des zu installierenden PiOS auswählen (ich wähle wie gezeigt die 1,1 GB-Variante - mit Desktop - wird auch empfohlen und mehr ist auch nicht notwendig)



4. Danach die gesteckte SD-Karte auswählen und das Schreiben starten, nach Ausführung meldet sich das Programm wie folgt zurück:



Ich bin durch diesen Prozess recht schnell gesprungen, da ich denke, daß es hier keine Unklarheiten geben sollte.

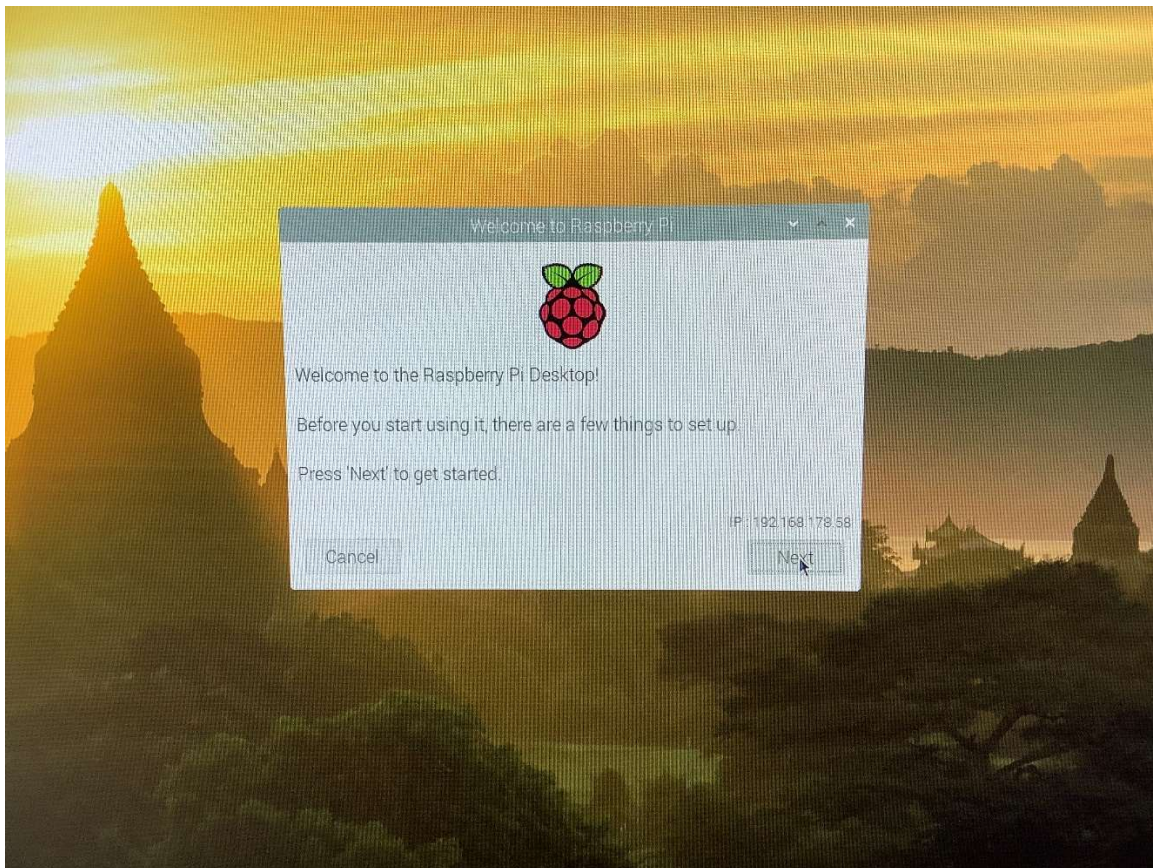
An anderen Stellen werde ich wesentlich detaillierter protokollieren. Versprochen.

5. Diese mit einem neuen PiOS formatierte Karte wird nun in den Rasperry Pi eingesteckt, der im weiteren Verlauf dann mit RailPi bezeichnet wird. Dann wird Spannung an den RailPi gelegt und er bootet.

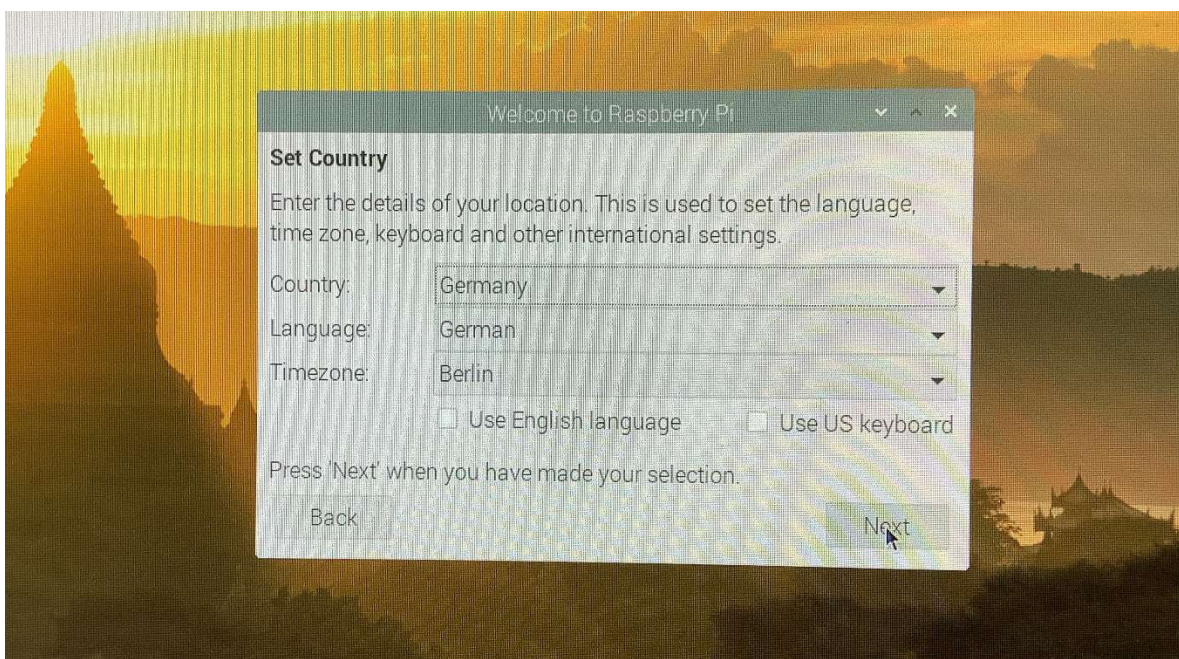
Ich hatte im Vorfeld davon gesprochen, daß der RailPi aufgrund der Verwendung von VNC keinen Monitor und keine Tastatur und Maus benötigt.

Eine Ausnahme davon ist der erste Bootvorgang, da wir den VNC-Server ja erst noch aktivieren müssen. Ich zeige nachstend hier also alle Schritte, die nach dem Erst-Start erledigt werden müssen.

6. Nach dem erfolgreichen ersten Bootvorgang meldet sich direkt die Raspberry-Pi-Konfiguration und man wird insbesondere durch die Lokalisierung des Pi geführt (hier Fotos, da noch keine Screenshots möglich sind):

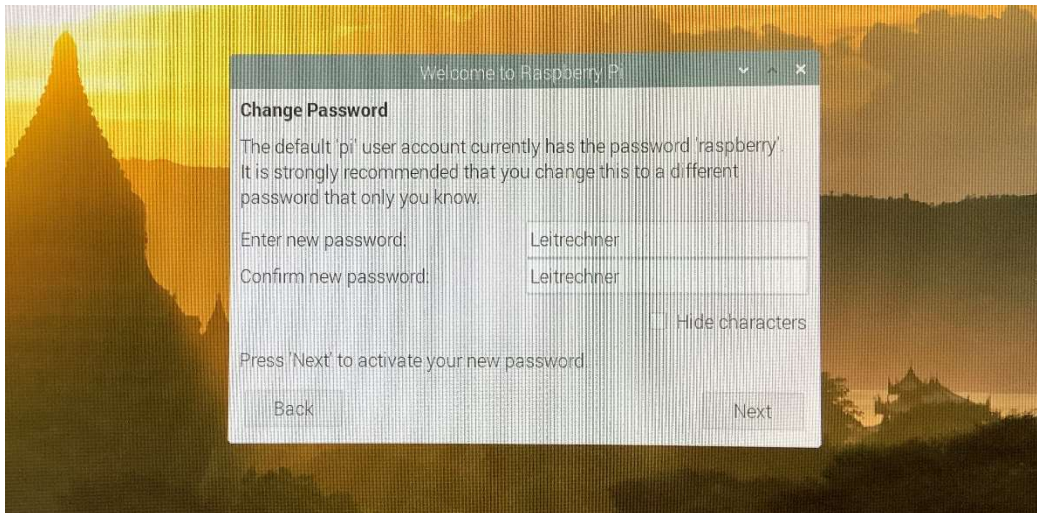


Weiter mit Next - wie oberhalb der Schaltfläche erkennbar ist, zeigt der mit dem LAN verbundene Pi bereits hier seine IP-Adresse an, die er vom DHCP-Server erhalten hat.

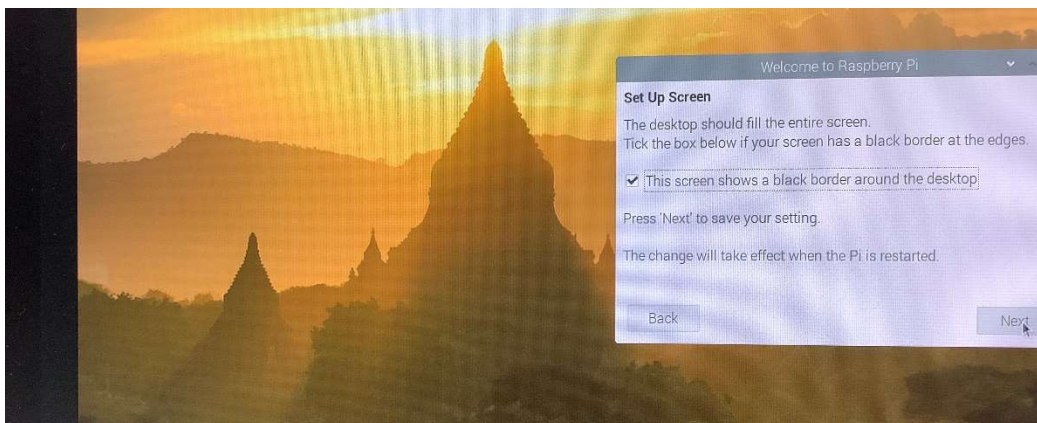


Und weiter geht es mit der Bestimmung der location: da ich in Deutschland lebe und die deutsche Sprache vorzugsweise nutze werden die entsprechenden Auswahlfelder eingestellt.

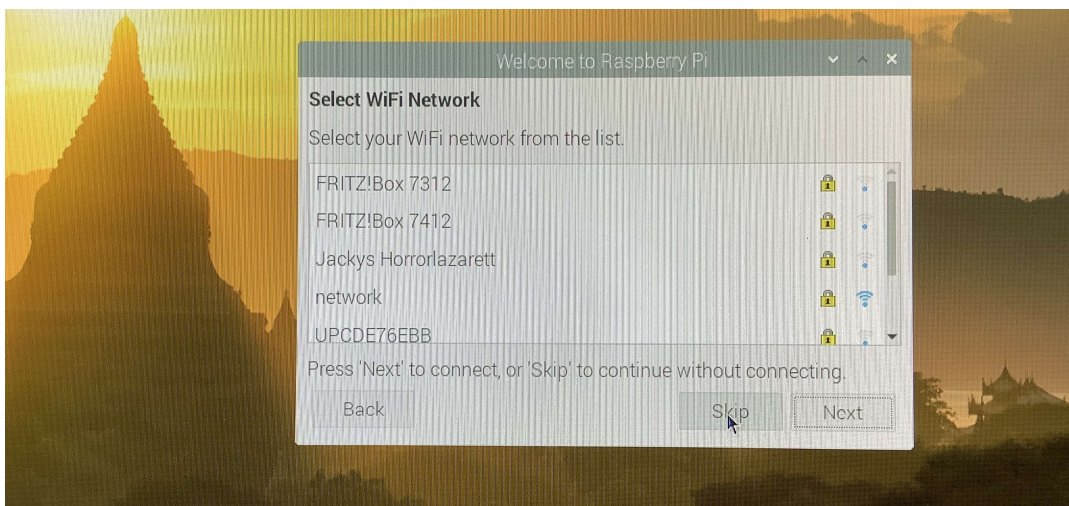
Dann muss noch das Passwort gesetzt werden - wie oben erwähnt lautet es für diese Doku 'Leitrechner'



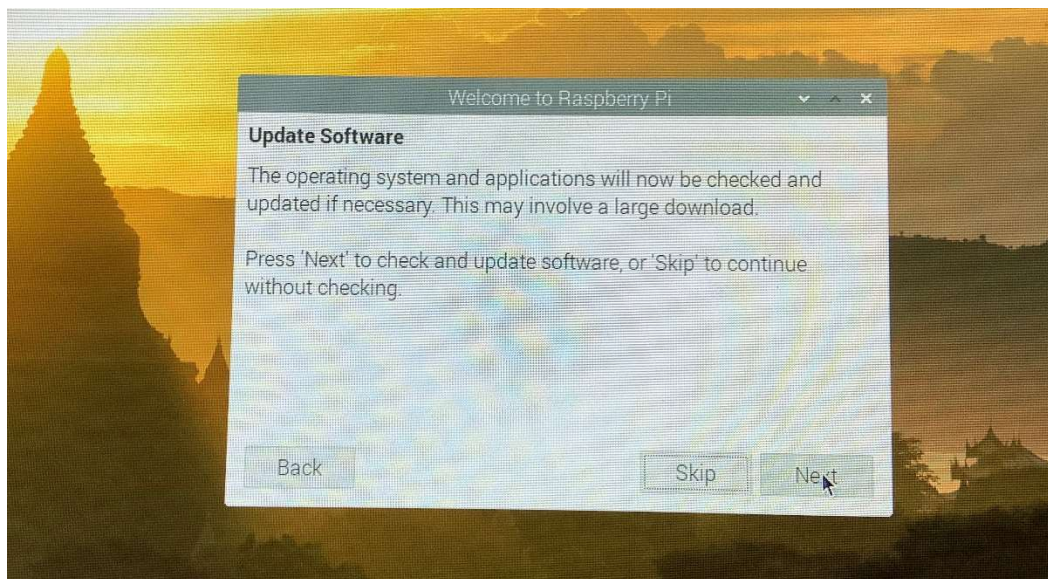
Nur der Vollständigkeit halber soll hier kurz gezeigt werden, wie präzise die Monitoranpassung hier erfolgen kann.



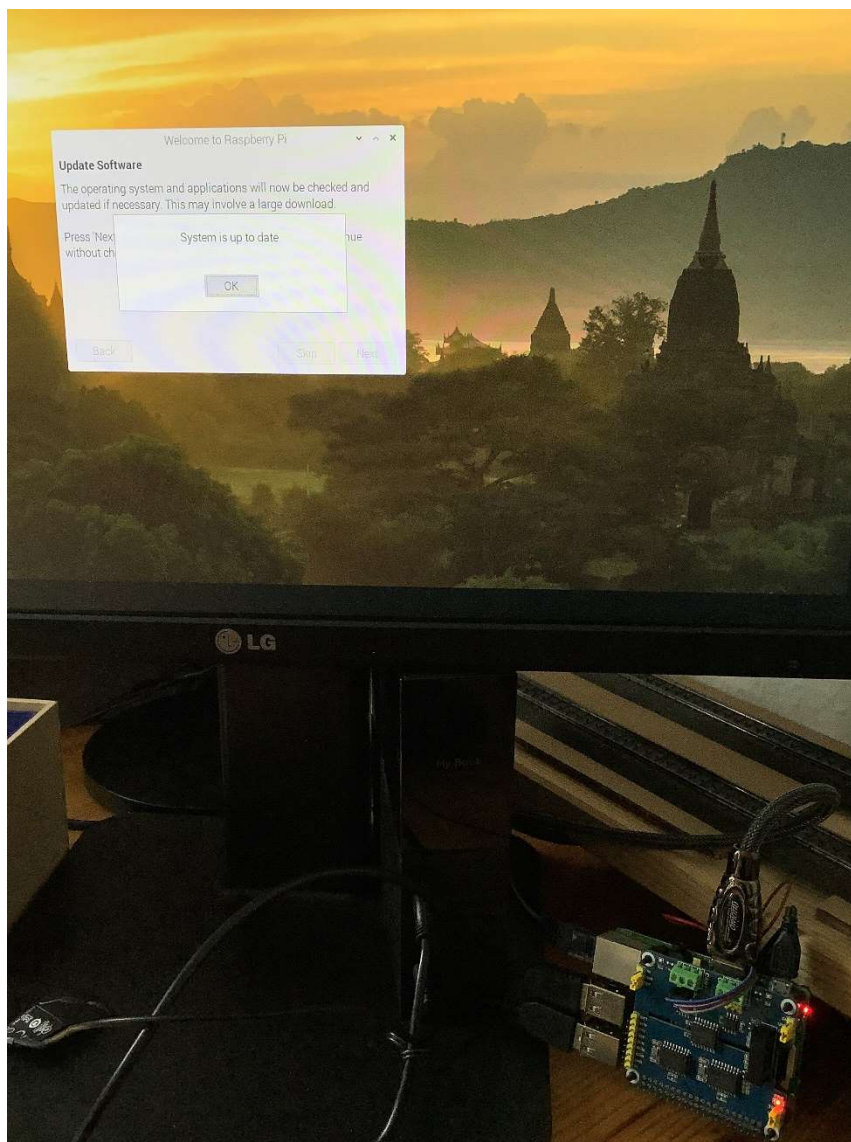
Da wir unseren Pi ausschließlich per LAN-Kabel betreiben wollen, können wir auch diesen Schritt überspringen:



Diesen Schritt überspringen wir nicht: wir aktualisieren die Software (dauert nur ein paar Minuten)

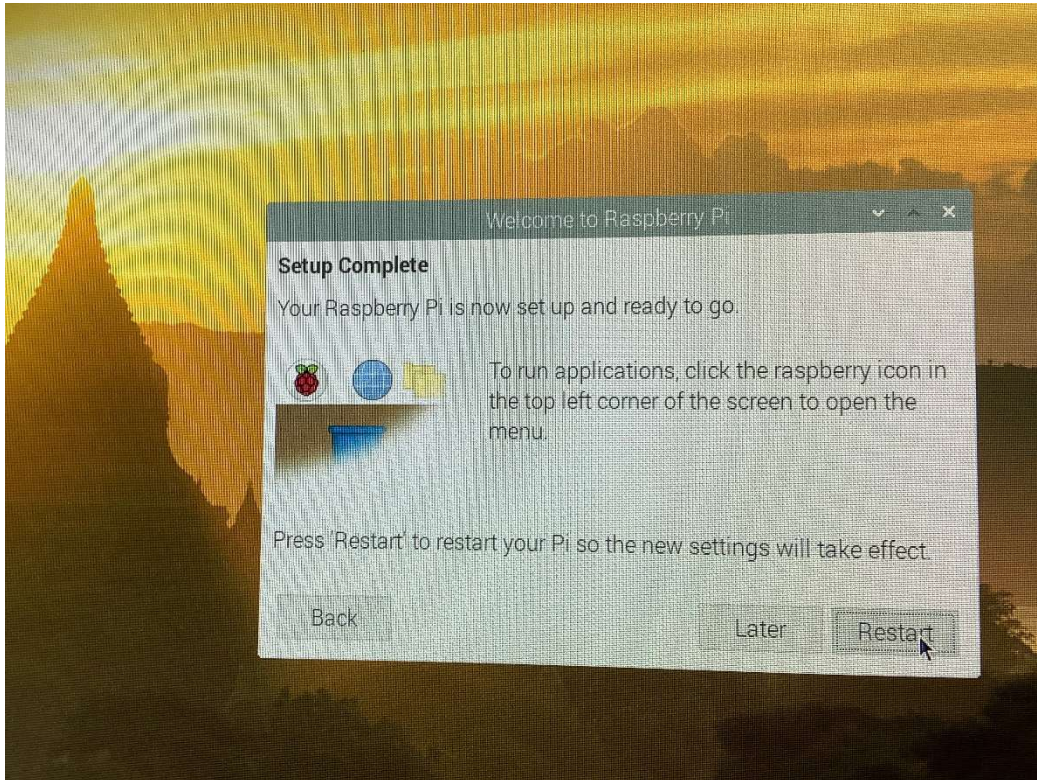


Immerhin haben wir in dieser Zeit noch Zeit für ein weiteres Foto:

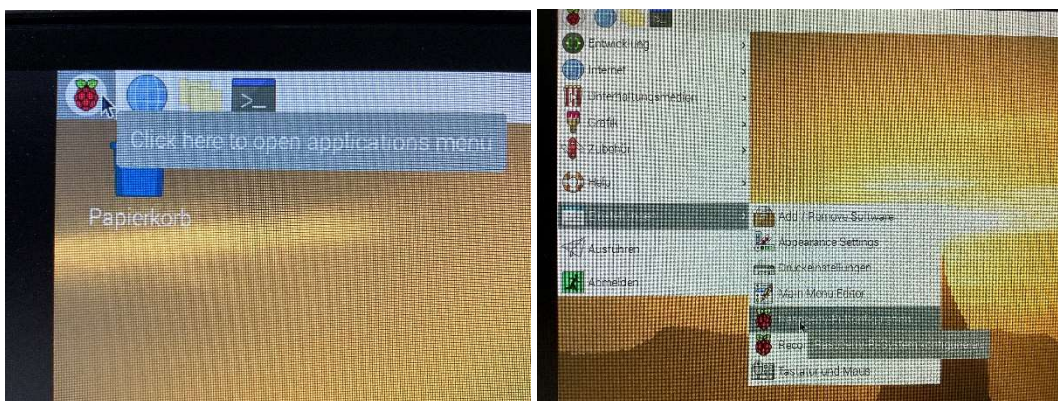


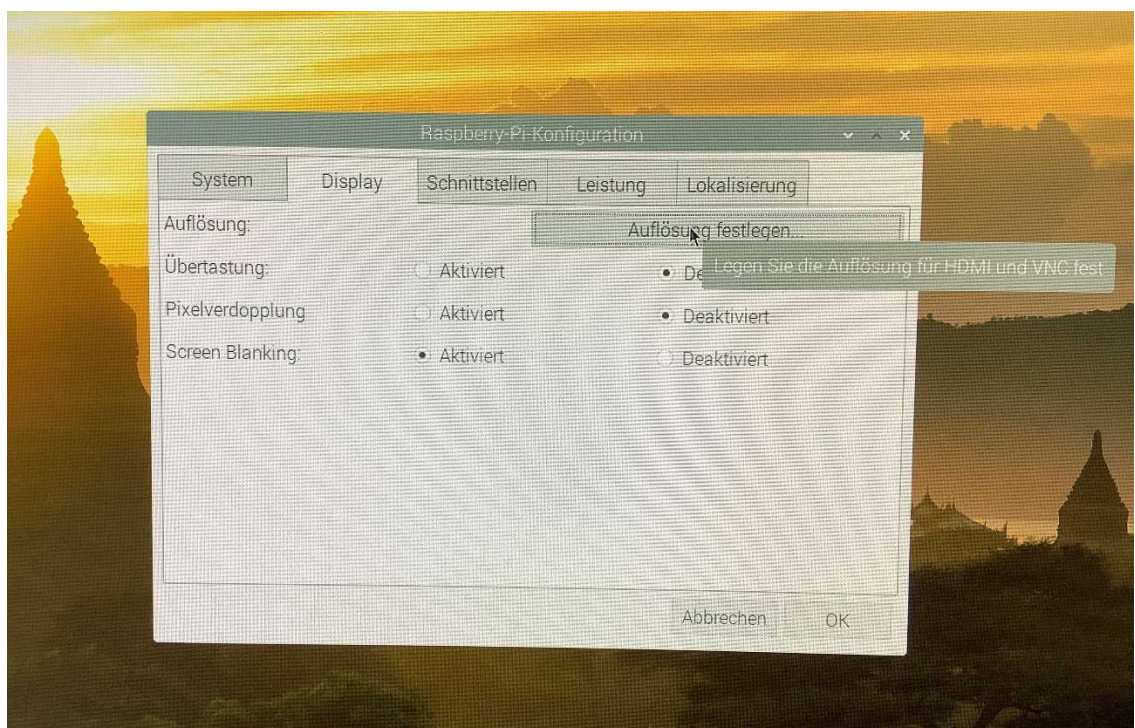
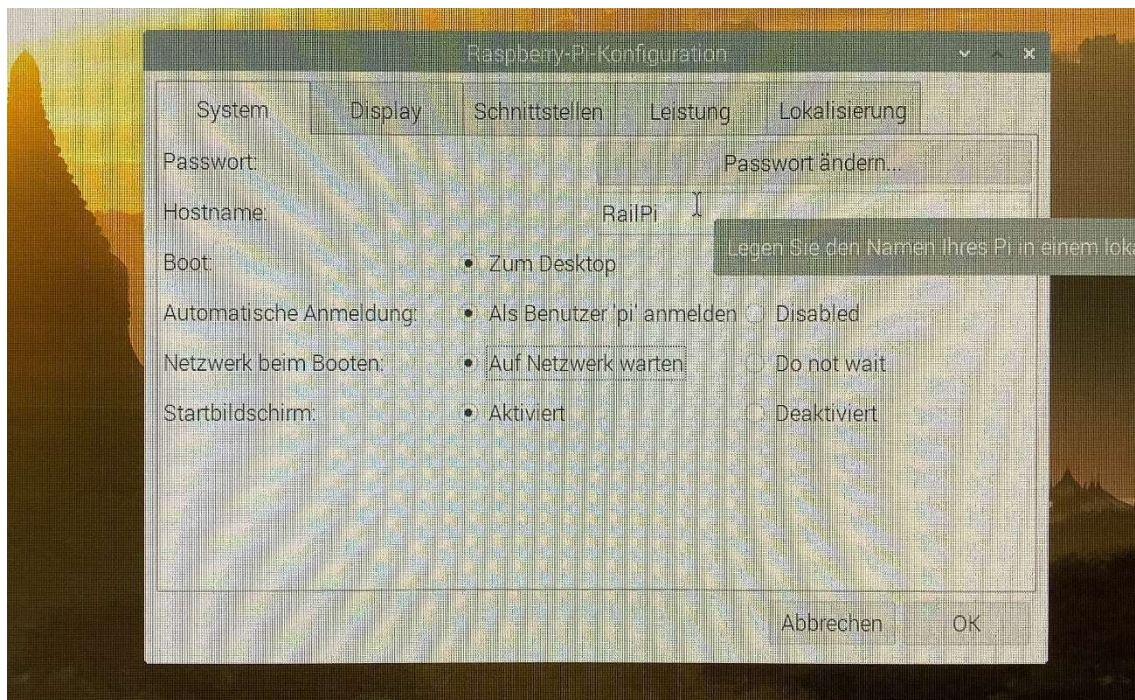
Und wie zu erkennen ist, wurde der RailPi bereits vor der Installation mit dem CAN-HAT und dem CAN-Bus verbunden. Ist ja kein USB-Gerät ;)

Und es erfolgt der erste Neustart nach Einstellung der wichtigsten Optionen:

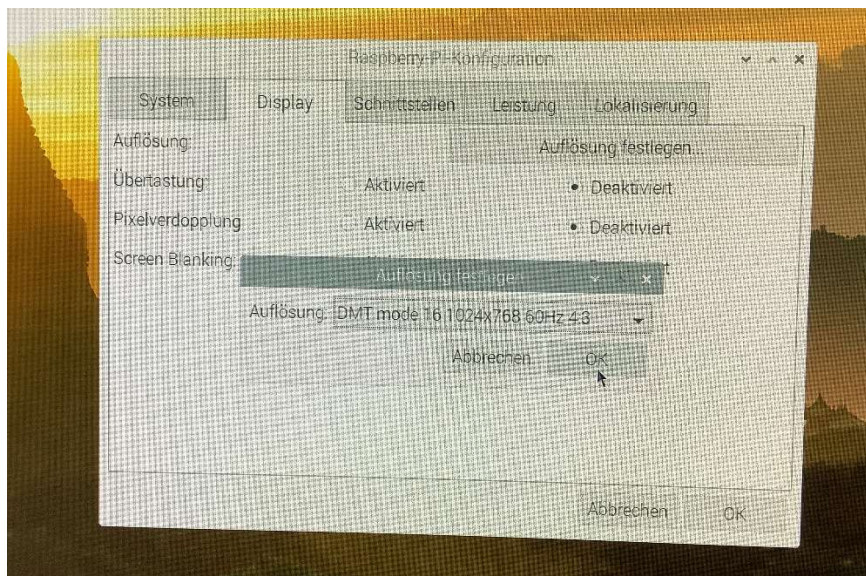


7. Nach dem Neustart gilt unser Augenmerk dem Systemmenü, denn wir müssen im Raspberry-Pi-Konfigurator noch einige weitere Einstellungen vornehmen:

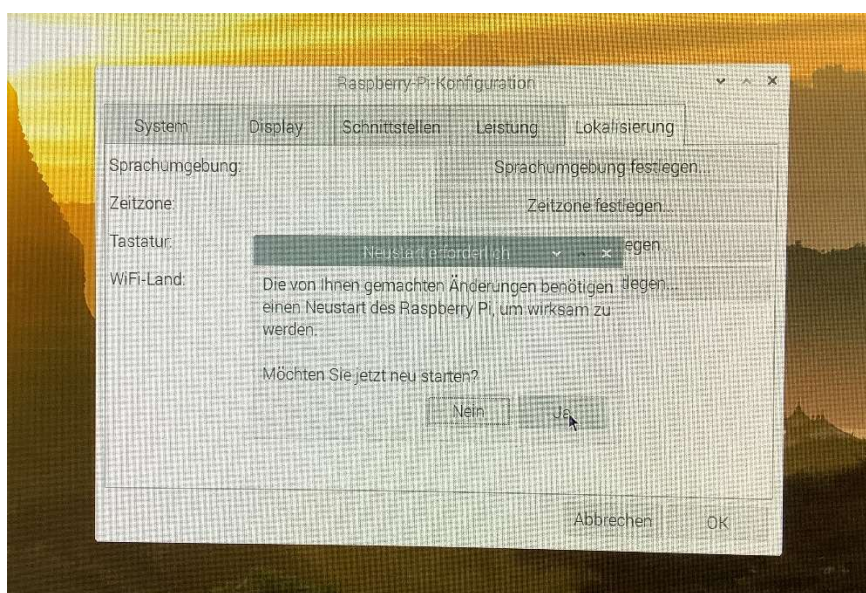
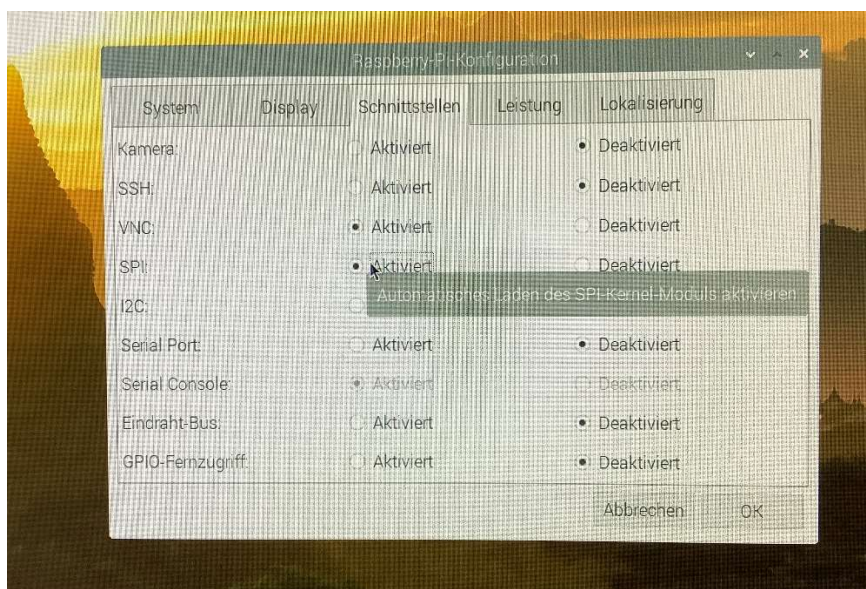




Da wir den RailPi über VNC bedienen möchten, wird erst einmal eine kleinere Auflösung eingestellt: nicht mehr als notwendig... 1024x768 genügt vollkommen. Auch das Screen-Blanking wird nicht benötigt...

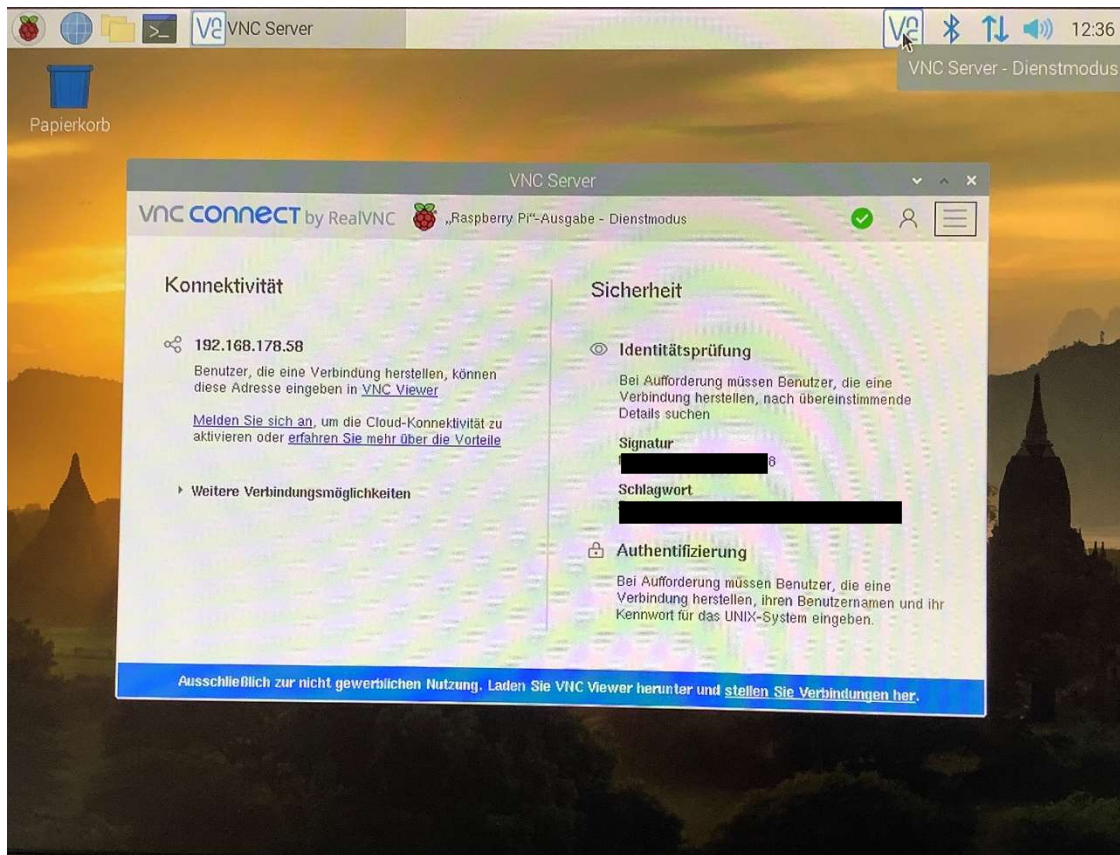


Aber die Aktivierung des VNC-Servers und des SPI-Kernel-Modus sind sehr wichtige Einstellungen für später in Kapitel 3.2



Jetzt ist unser RailPi vom eigentlichen Pi OS her vollständig vorbereitet und muss neu gestartet werden!

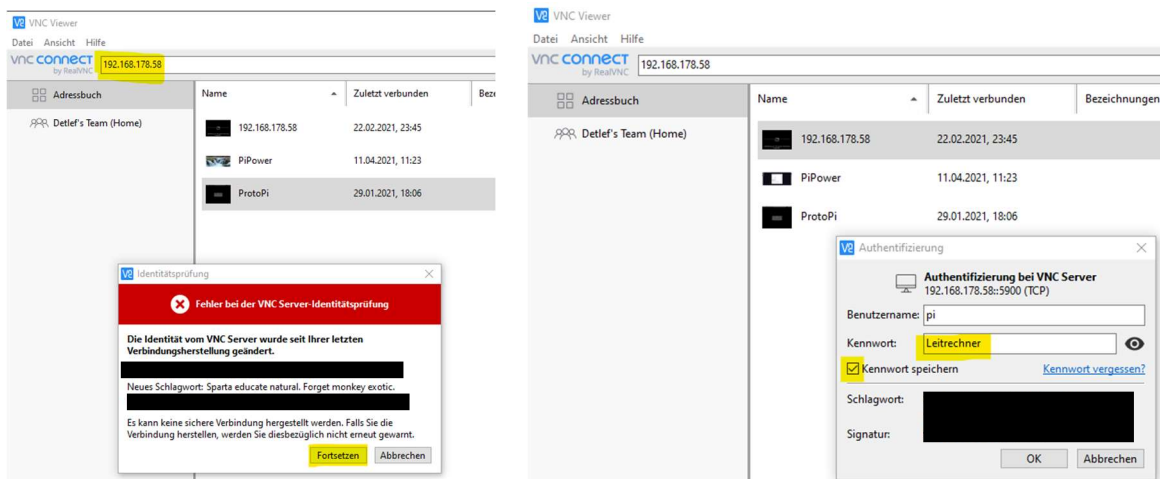
Nach dem Neustart zeigt sich der Desktop wie nachstehend:



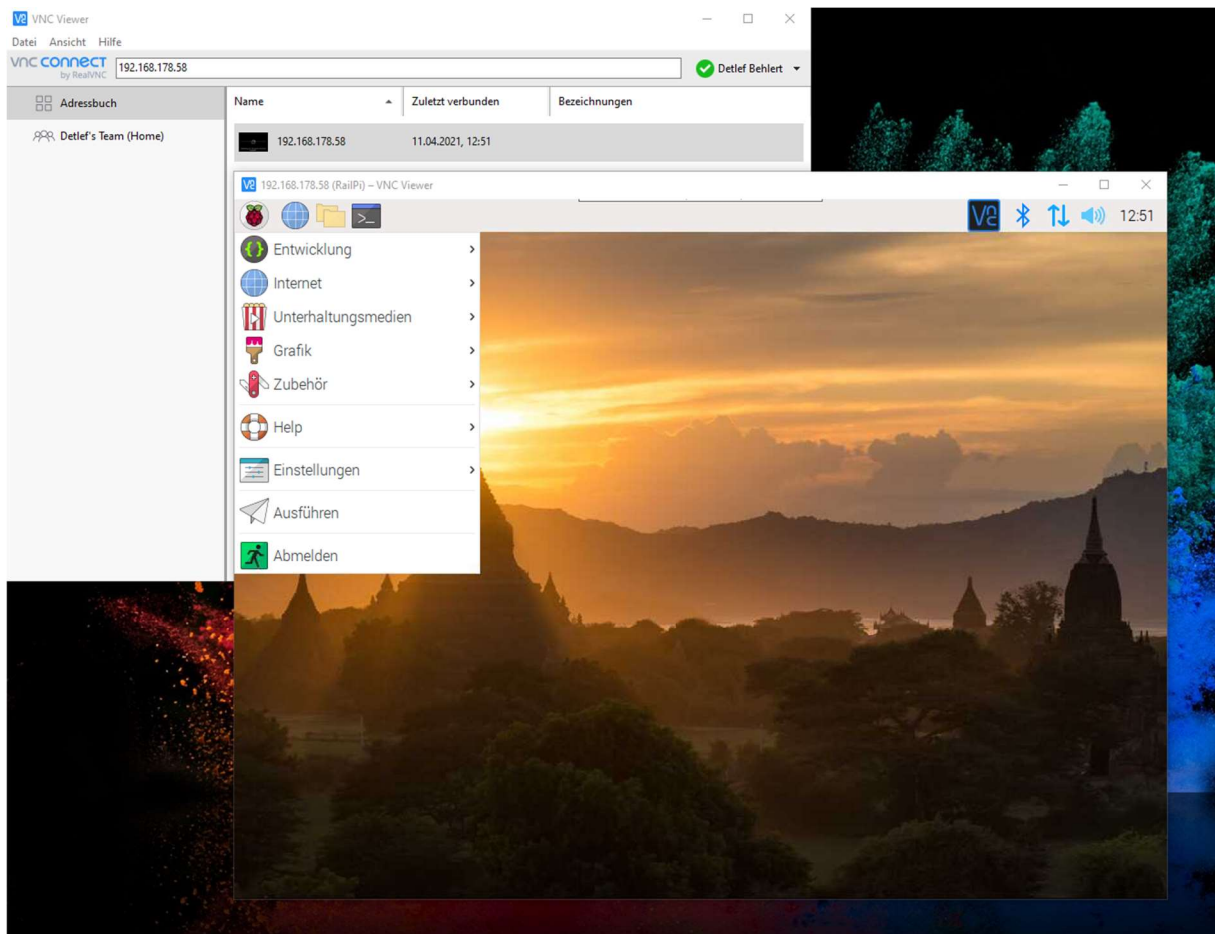
Und ein Klick auf das VNC-Icon rechts oben auf der Statusleiste zeigt, daß der VNC-Server betriebsbereit ist:

VNC kann viel mehr als wir benötigen. Daher ist eine Anmeldung, um die Cloud-Konnektivität nutzen zu können, nicht notwendig.

Das war das letzte Foto - jetzt werden Monitor, Tastatur und Maus vom RailPi getrennt und auf dem zukünftigen Visualisierungs-PC o.ä. wird der VNC-Viewer gestartet (sofern er installiert ist - dann wären Monitor & Co schon wichtig...)



Und da ist der RailPi nun in einem VNC-Windows-Fenster:



Nun kann ich auf Fotos wieder verzichten und direkt ScreenShots generieren. Der RailPi an sich ist für unsere Belange nun vollständig eingerichtet.

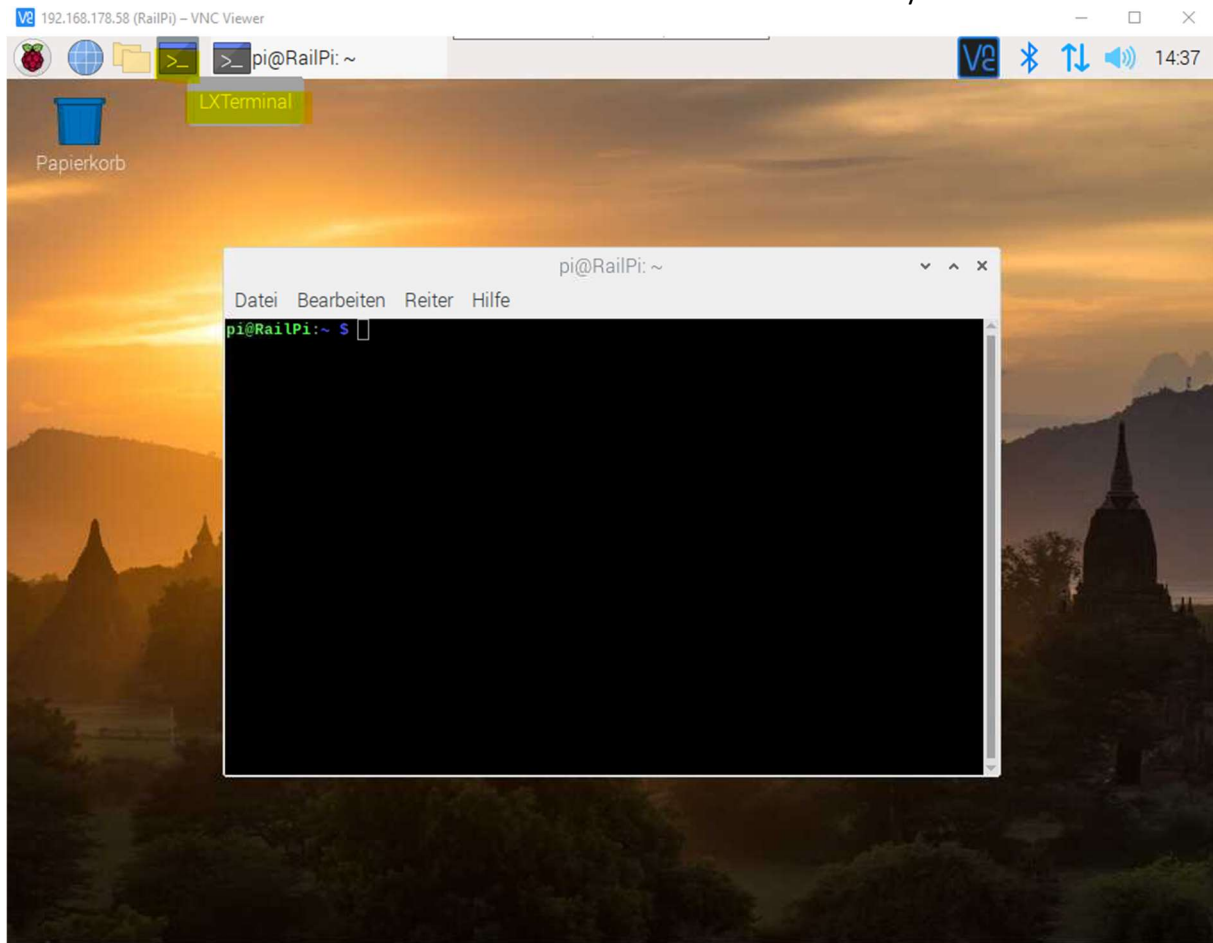
3.2 - Installation von CAN2LAN

CAN2LAN ist nicht nur ein Programm von Gerd Bertelsmann, er hat auch für Linux-Neulinge eine vorzügliche Installationsanleitung geschrieben (zu finden unter <https://github.com/GBert/misc/tree/master/RPi-MCP2515>), der ich nun folgen möchte.

Die Programme finden wir unter <https://github.com/GBert/railroad>

Und damit wir uns nicht vorab mit der Struktur und vorhandenen anderen (Hilfs-)Programmen etc. beschäftigen müssen, folgt jetzt die detaillierte Doku der Installation, natürlich klar bezogen auf unsere Hardware - siehe Beispielwarenkorb unter 2.2 - Welectron:

Zunächst starten wir im RailPi die Konsole durch Klick auf das LX-Terminal-Symbol:



Von nun an konzentrieren wir uns ausschließlich auf das Terminal-/Konsolenfenster

1. CAN Nutzung vorbereiten

Die Befehle lauten

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install can-utils
```

Und so sieht das dann in Interaktion mit dem RailPi aus (gelb die eingegebenen obigen Befehle, weiß die Antwort des RailPi's)

```
pi@RailPi:~$ sudo apt-get update
OK:1 http://archive.raspberrypi.org/debian buster InRelease
Holen:2 http://raspbian.raspberrypi.org/raspbian buster InRelease [15,0 kB]
Es wurden 15,0 kB in 1 s geholt (12,6 kB/s).
Paketlisten werden gelesen... Fertig
pi@RailPi:~$ sudo apt-get upgrade
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Paketaktualisierung (Upgrade) wird berechnet... Fertig
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
pi@RailPi:~$ sudo apt-get install can-utils
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden NEUEN Pakete werden installiert:
  can-utils
0 aktualisiert, 1 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen 103 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 430 kB Plattenplatz zusätzlich benutzt.
Holen:1 http://mirror.de.leaseweb.net/raspbian/raspbian buster/main armhf can-utils armhf 2018.02.0-1 [103 kB]
Es wurden 103 kB in 0 s geholt (245 kB/s).
Vormals nicht ausgewähltes Paket can-utils wird gewählt.
(Lese Datenbank ... 98611 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../can-utils_2018.02.0-1_armhf.deb ...
Entpacken von can-utils (2018.02.0-1) ...
can-utils (2018.02.0-1) wird eingerichtet ...
Trigger für man-db (2.8.5-2) werden verarbeitet ...
pi@RailPi:~$
```

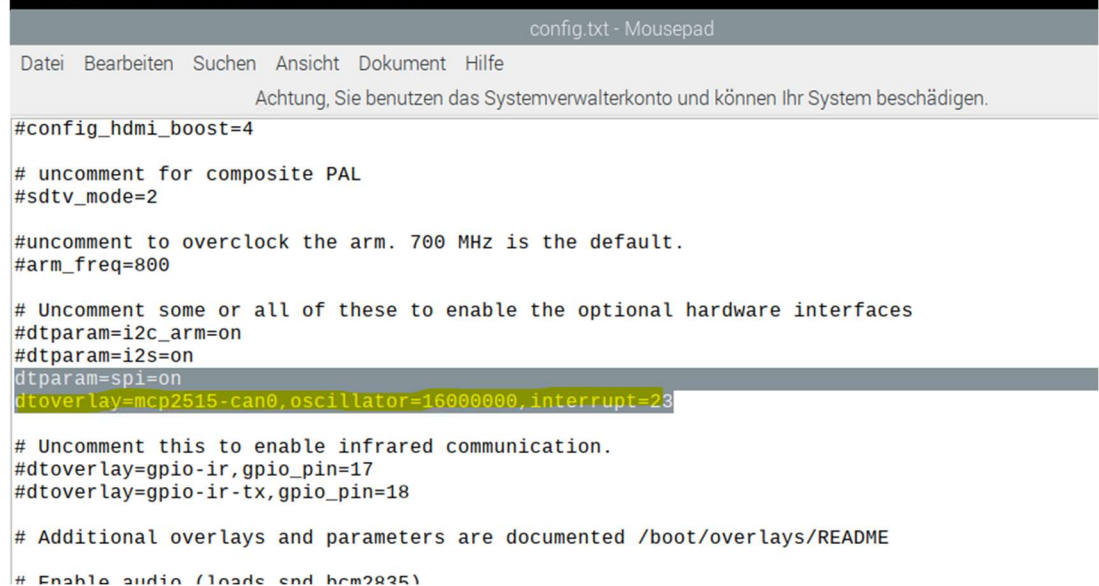
2. /boot/config.txt editieren

In der o.g. Datei config.txt muß nun die folgende Zeile unmittelbar hinter dtparam=spi=on hinzugefügt werden:

```
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=23
```

Da wir ja keine Linux-Profis sind und uns - auch beim Ändern einer Datei eher auf die grafische Oberfläche verlassen (müssen), rufen wir zur Änderung der config.txt das Programm mousepad auf und fügen die neue Zeile ein:

```
pi@RailPi:~$
pi@RailPi:~$ sudo mousepad /boot/config.txt
```



```
#config_hdmi_boost=4

# uncomment for composite PAL
#sdtv_mode=2

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
#dtparam=i2c_arm=on
#dtparam=i2s=on
dtparam=spi=on
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=23

# Uncomment this to enable infrared communication.
#dtoverlay=gpio-ir,gpio_pin=17
#dtoverlay=gpio-ir-tx,gpio_pin=18

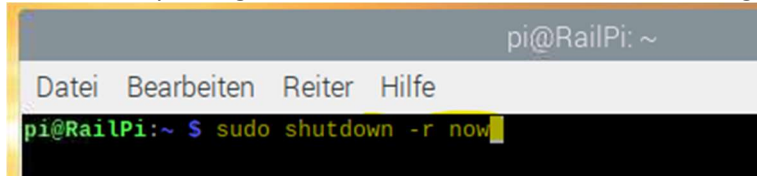
# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
```


Speichern nicht vergessen und dann:

3. reboot

Da wir 'am System geschraubt haben', muss der RailPi mit obigem Befehl neu gestartet werden:



```
pi@RailPi: ~  
Datei Bearbeiten Reiter Hilfe  
pi@RailPi:~ $ sudo shutdown -r now
```

4. CAN Interface konfigurieren

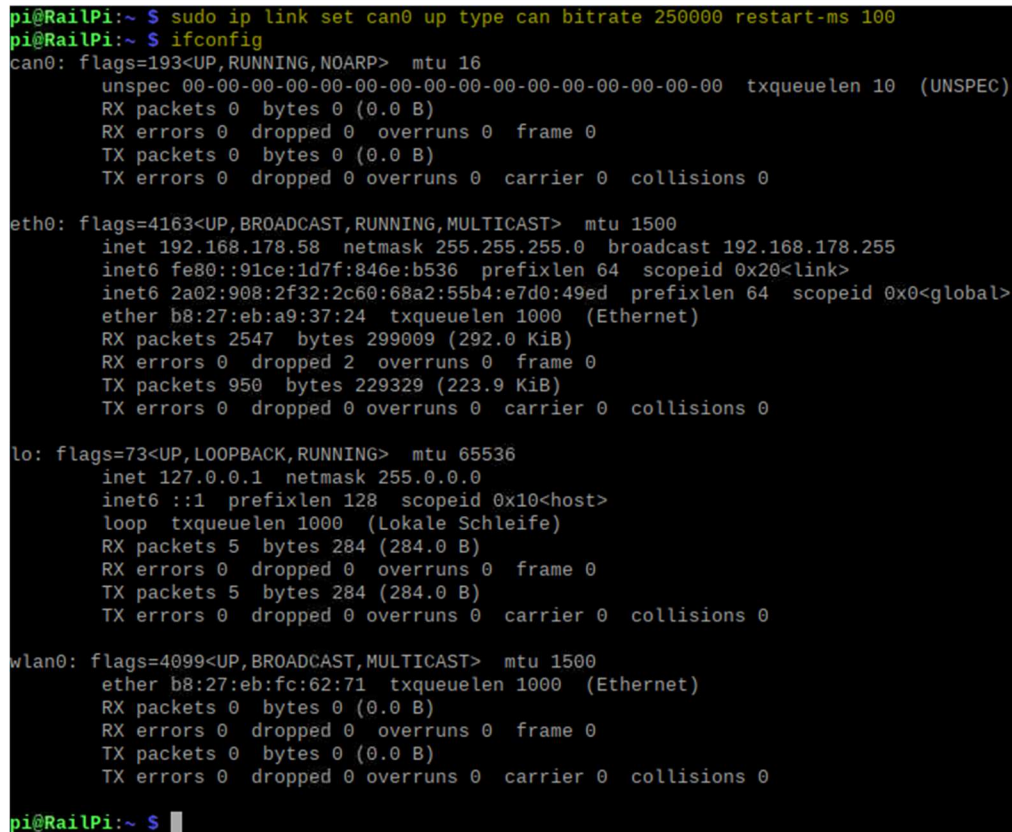
`sudo ip link set can0 up type can bitrate 250000 restart-ms 100`

bzw: die Datei `/etc/network/interfaces.d/can0` wie folgt anlegen:

```
auto can0  
iface can0 inet manual  
    pre-up /sbin/ip link set $IFACE type can bitrate 250000 restart-ms 100  
    up /sbin/ifconfig $IFACE up  
    down /sbin/ifconfig $IFACE down
```

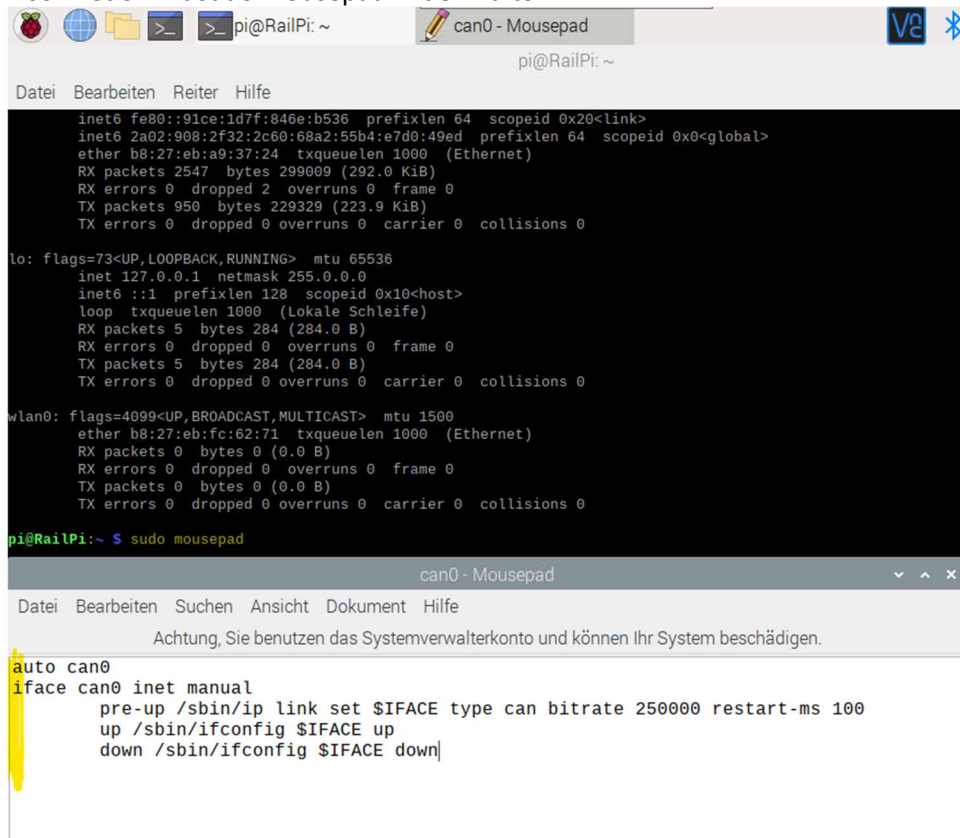
Aus <<https://github.com/GBert/misc/tree/master/RPi-MCP2515>>

Zunächst also der manuelle Konfiguration der Schnittstelle:



```
pi@RailPi:~ $ sudo ip link set can0 up type can bitrate 250000 restart-ms 100  
pi@RailPi:~ $ ifconfig  
can0: flags=193<UP,RUNNING,NOARP> mtu 16  
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 10 (UNSPEC)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
    inet 192.168.178.58 netmask 255.255.255.0 broadcast 192.168.178.255  
    inet6 fe80::91ce:1d7f:846e:b536 prefixlen 64 scopeid 0x20<link>  
    inet6 2a02:908:2f32:2c60:68a2:55b4:e7d0:49ed prefixlen 64 scopeid 0x0<global>  
    ether b8:27:eb:a9:37:24 txqueuelen 1000 (Ethernet)  
    RX packets 2547 bytes 299009 (292.0 KiB)  
    RX errors 0 dropped 2 overruns 0 frame 0  
    TX packets 950 bytes 229329 (223.9 KiB)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
    inet 127.0.0.1 netmask 255.0.0.0  
    inet6 ::1 prefixlen 128 scopeid 0x10<host>  
    loop txqueuelen 1000 (Lokale Schleife)  
    RX packets 5 bytes 284 (284.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 5 bytes 284 (284.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
    ether b8:27:eb:fc:62:71 txqueuelen 1000 (Ethernet)  
    RX packets 0 bytes 0 (0.0 B)  
    RX errors 0 dropped 0 overruns 0 frame 0  
    TX packets 0 bytes 0 (0.0 B)  
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@RailPi:~ $
```

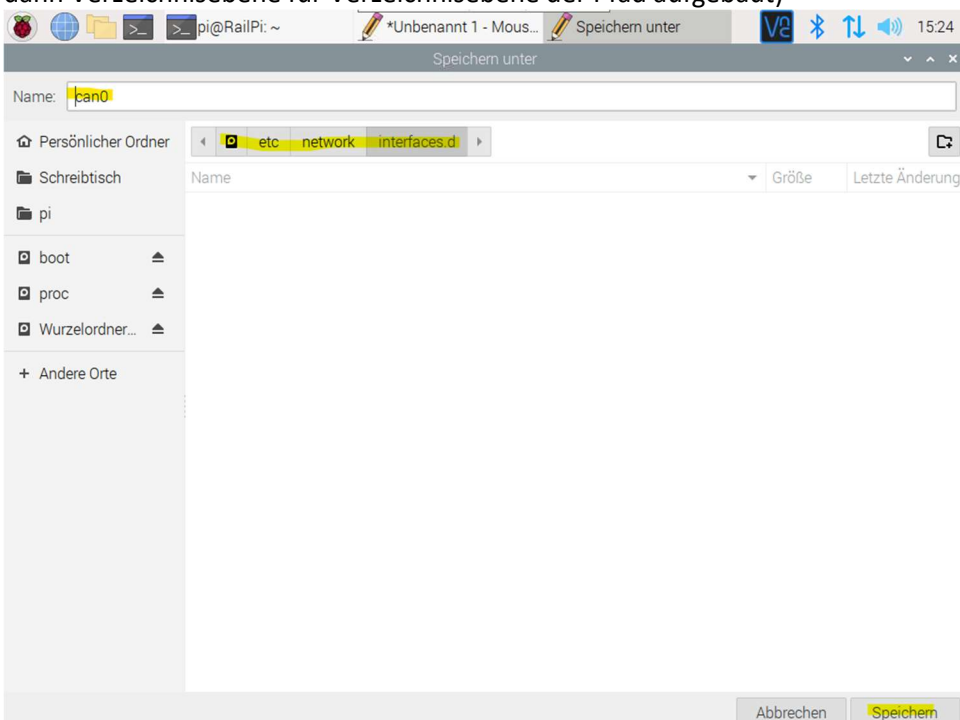
Dann wäre es ja nicht schlecht, wenn bei jedem Neustart die Schnittstelle eingebunden würde.
Also wieder mit `sudo mousepad` in den Editor:



```
pi@RailPi: ~  
Datei Bearbeiten Reiter Hilfe  
inet6 fe80::91ce:1d7f:846e:b536 prefixlen 64 scopeid 0x20<link>  
inet6 2a02:908:2f32:2c60:68a2:55b4:e7d0:49ed prefixlen 64 scopeid 0x0<global>  
ether b8:27:eb:a9:37:24 txqueuelen 1000 (Ethernet)  
RX packets 2547 bytes 299009 (292.0 KiB)  
RX errors 0 dropped 2 overruns 0 frame 0  
TX packets 950 bytes 229329 (223.9 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1000 (Lokale Schleife)  
RX packets 5 bytes 284 (284.0 B)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 5 bytes 284 (284.0 B)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
wlan0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500  
ether b8:27:eb:fc:62:71 txqueuelen 1000 (Ethernet)  
RX packets 0 bytes 0 (0.0 B)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 0 bytes 0 (0.0 B)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
pi@RailPi:~$ sudo mousepad  
can0 - Mousepad  
Datei Bearbeiten Suchen Ansicht Dokument Hilfe  
Achtung, Sie benutzen das Systemverwalterkonto und können Ihr System beschädigen.  
auto can0  
iface can0 inet manual  
    pre-up /sbin/ip link set $IFACE type can bitrate 250000 restart-ms 100  
    up /sbin/ifconfig $IFACE up  
    down /sbin/ifconfig $IFACE down|
```

Wichtig bei der Eingabe des Textes: auch die letzte Zeile muß mit <RETURN> abgeschlossen werden!
Im Bild oben steht der Cursor noch hinter dem Wort 'down' - so würde es nicht funktionieren...

Dann unter dem Pfad `/etc/network/interfaces.d` und dem Namen `can0` speichern:
(damit der Pfad beim Speichern gewählt werden kann, wird zunächst auf '**Wurzelordner**' geklickt und dann Verzeichnisebene für Verzeichnisebene der Pfad aufgebaut)



Dann den Editor mit 'Datei - Fenster schließen' beenden.

5. can2lan nutzen

```
sudo apt-get install lighttpd
# zumindest geraet.vrs unter /var/www/html/config/ anlegen:
sudo mkdir -p /var/www/html/config/
sudo cat << EOF > /var/www/html/config/geraet.vrs
[geraet]
version
.minor=1
geraet
.sernum=1
.hardvers=RPi,3
EOF
```

Aus <<https://github.com/GBert/misc/tree/master/RPi-MCP2515>>

Die Schnittstelle allein nutzt uns ja noch nichts. Also bereiten wir die Installation von CAN2LAN vor:

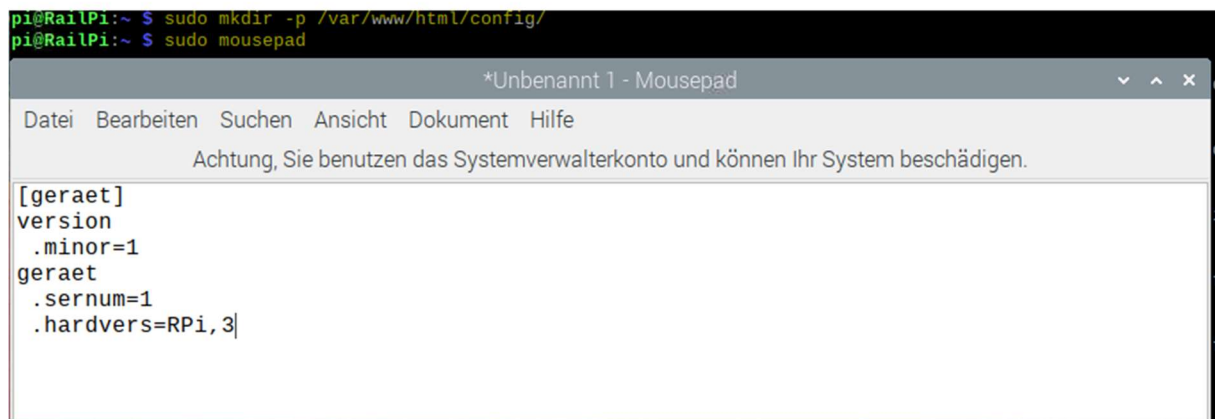
```
pi@RailPi:~$ sudo apt-get install lighttpd
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
Die folgenden zusätzlichen Pakete werden installiert:
  libfam0 lighttpd-modules-ldap lighttpd-modules-mysql spawn-fcgi
Vorgeschlagene Pakete:
  fam rrdtool php-cgi apache2-utils lighttpd-doc
Die folgenden NEUEN Pakete werden installiert:
  libfam0 lighttpd lighttpd-modules-ldap lighttpd-modules-mysql spawn-fcgi
0 aktualisiert, 5 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen 349 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 1.130 kB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n] J
Holen:1 http://ftp.halifax.rwth-aachen.de/raspbian/raspbian buster/main armhf libfam0 armhf 2.7.
Holen:2 http://ftp.gwdg.de/pub/linux/debian/raspbian/raspbian buster/main armhf lighttpd armhf 1
Holen:3 http://mirror1.hs-esslingen.de/pub/Mirrors/archive.raspbian.org/raspbian buster/main arm
  1.4.53-4+deb10u1 [8.848 B]
Holen:4 http://ftp.agdsn.de/pub/mirrors/raspbian/raspbian buster/main armhf lighttpd-modules-mys
  98 B]
Holen:5 http://ftp.halifax.rwth-aachen.de/raspbian/raspbian buster/main armhf spawn-fcgi armhf 1
Es wurden 349 kB in 1 s geholt (337 kB/s).
Vormals nicht ausgewähltes Paket libfam0:armhf wird gewählt.
(Lese Datenbank ... 98673 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../libfam0_2.7.0-17.3_armhf.deb ...
Entpacken von libfam0:armhf (2.7.0-17.3) ...
Vormals nicht ausgewähltes Paket lighttpd wird gewählt
```

: die Informationen bei Installieren können schon mal etwas länger werden...

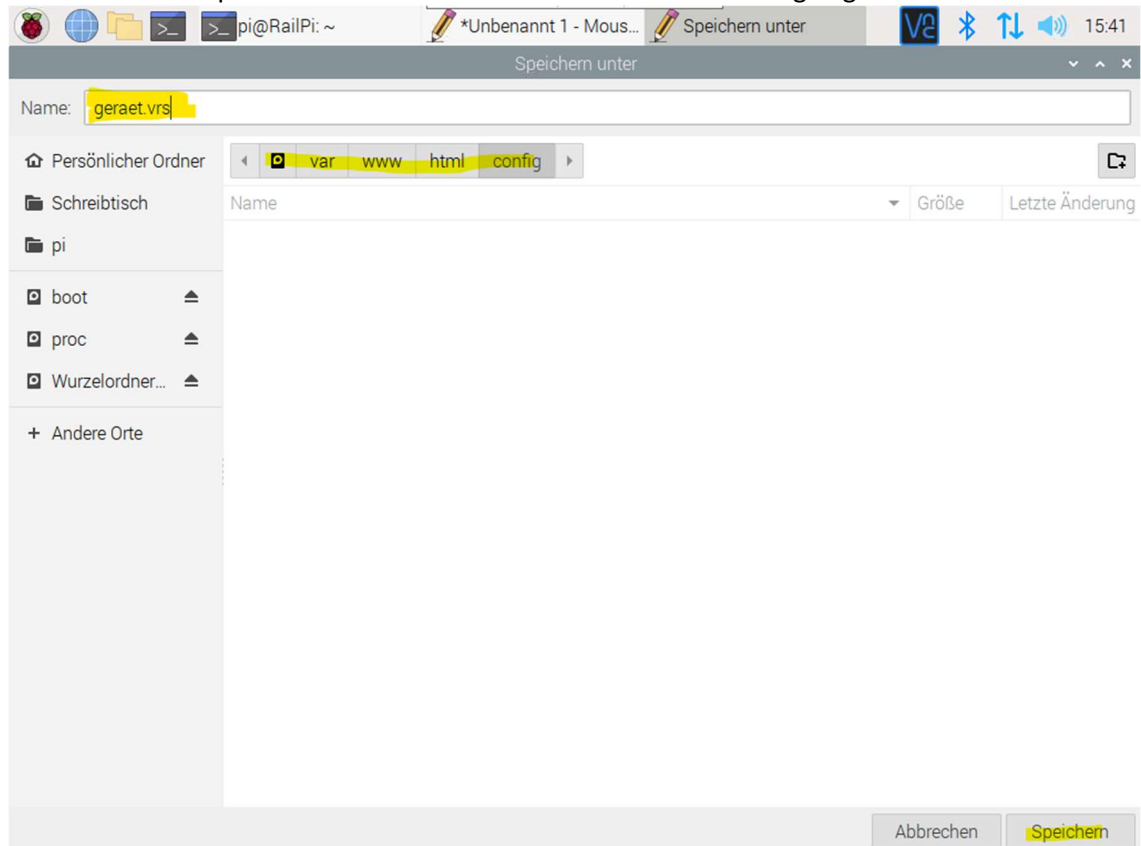
```
Vorbereitung zum Entpacken von .../libfam0_2.7.0-17.3_armhf.deb ...
Entpacken von libfam0:armhf (2.7.0-17.3) ...
Vormals nicht ausgewähltes Paket lighttpd wird gewählt.
Vorbereitung zum Entpacken von .../lighttpd_1.4.53-4+deb10u1_armhf.deb ...
Entpacken von lighttpd (1.4.53-4+deb10u1) ...
Vormals nicht ausgewähltes Paket lighttpd-modules-ldap wird gewählt.
Vorbereitung zum Entpacken von .../lighttpd-modules-ldap_1.4.53-4+deb10u1_armhf.deb ...
Entpacken von lighttpd-modules-ldap (1.4.53-4+deb10u1) ...
Vormals nicht ausgewähltes Paket lighttpd-modules-mysql wird gewählt.
Vorbereitung zum Entpacken von .../lighttpd-modules-mysql_1.4.53-4+deb10u1_armhf.deb ...
Entpacken von lighttpd-modules-mysql (1.4.53-4+deb10u1) ...
Vormals nicht ausgewähltes Paket spawn-fcgi wird gewählt.
Vorbereitung zum Entpacken von .../spawn-fcgi_1.6.4-2_armhf.deb ...
Entpacken von spawn-fcgi (1.6.4-2) ...
libfam0:armhf (2.7.0-17.3) wird eingerichtet ...
spawn-fcgi (1.6.4-2) wird eingerichtet ...
lighttpd (1.4.53-4+deb10u1) wird eingerichtet ...
Created symlink /etc/systemd/system/multi-user.target.wants/lighttpd.service → /lib/systemd/systeme
lighttpd-modules-ldap (1.4.53-4+deb10u1) wird eingerichtet ...
lighttpd-modules-mysql (1.4.53-4+deb10u1) wird eingerichtet ...
Trigger für systemd (241-7~deb10u7+rpil) werden verarbeitet ...
Trigger für man-db (2.8.5-2) werden verarbeitet ...
Trigger für libc-bin (2.28-10+rpil) werden verarbeitet ...
pi@RailPi:~$
```

Jetzt wird ein neues Verzeichnis angelegt und darin mittels mousepad eine neue Datei erzeugt:

```
pi@RailPi:~ $ sudo mkdir -p /var/www/html/config/
pi@RailPi:~ $ sudo mousepad
```



Und mit 'Datei - Speichern unter' unter dem Namen Geraet.vrs abgelegt



Mit 'Datei - Fenster schließen' beenden wir den Editor wieder.

6. (und letztens) can2lan herunter laden und starten

```
cd
sudo apt-get install git libpcap-dev
git clone https://github.com/GBert/railroad.git
cd railroad/can2udp/src
make
sudo cp can2lan /usr/sbin/
sudo cp can-monitor /usr/bin/
sudo cp ../files/maerklin/config/gleisbild.cs2 /var/www/html/gleisbild.cs2
# Debian Init Skript installieren und testen
sudo cp ../debian/can2lan.init /etc/init.d
sudo /etc/init.d/can2lan.init start
sudo /etc/init.d/can2lan.init status
sudo /etc/init.d/can2lan.init stop
sudo update-rc.d can2lan.init defaults
# oder manuell im Vordergrund start
can2lan -mvf -c /var/www/html
```

Aus <<https://github.com/GBert/misc/tree/master/RPi-MCP2515>>

```
pi@RailPi:~$
pi@RailPi:~$ sudo apt-get install git libpcap-dev
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen... Fertig
git ist schon die neueste Version (1:2.20.1-2+deb10u3).
Die folgenden zusätzlichen Pakete werden installiert:
  libpcap0.8 libpcap0.8-dev
Die folgenden NEUEN Pakete werden installiert:
  libpcap-dev libpcap0.8 libpcap0.8-dev
0 aktualisiert, 3 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
Es müssen 370 kB an Archiven heruntergeladen werden.
Nach dieser Operation werden 919 kB Plattenplatz zusätzlich benutzt.
Möchten Sie fortfahren? [J/n] J
Holen:1 http://ftp.agdsn.de/pub/mirrors/raspbian/raspbian buster/main armhf libpcap0.8 armhf 1.8.1-6 [124 kB]
Holen:2 http://ftp.halifax.rwth-aachen.de/raspbian/raspbian buster/main armhf libpcap0.8-dev armhf 1.8.1-6 [220 kB]
Holen:3 http://ftp.halifax.rwth-aachen.de/raspbian/raspbian buster/main armhf libpcap-dev armhf 1.8.1-6 [25,9 kB]
Es wurden 370 kB in 1 s geholt (474 kB/s).
Vormals nicht ausgewähltes Paket libpcap0.8:armhf wird gewählt.
(Lese Datenbank ... 98800 Dateien und Verzeichnisse sind derzeit installiert.)
Vorbereitung zum Entpacken von .../libpcap0.8_1.8.1-6_armhf.deb ...
Entpacken von libpcap0.8:armhf (1.8.1-6) ...
Vormals nicht ausgewähltes Paket libpcap0.8-dev:armhf wird gewählt.
Vorbereitung zum Entpacken von .../libpcap0.8-dev_1.8.1-6_armhf.deb ...
Entpacken von libpcap0.8-dev:armhf (1.8.1-6) ...
Vormals nicht ausgewähltes Paket libpcap-dev:armhf wird gewählt.
Vorbereitung zum Entpacken von .../libpcap-dev_1.8.1-6_armhf.deb ...
Entpacken von libpcap-dev:armhf (1.8.1-6) ...
libpcap0.8:armhf (1.8.1-6) wird eingerichtet ...
libpcap0.8-dev:armhf (1.8.1-6) wird eingerichtet ...
libpcap-dev:armhf (1.8.1-6) wird eingerichtet ...
Trigger für man-db (2.8.5-2) werden verarbeitet ...
Trigger für libc-bin (2.28-10+rpil) werden verarbeitet ...
pi@RailPi:~$
```

Jetzt wird Gerts Repository auf den RailPi geholt und anschließend ins Source-Verzeichnis gewechselt und mit MAKE die CAN2LAN-Programme übersetzt:

```
pi@RailPi:~$ git clone https://github.com/GBert/railroad.git
Klone nach 'railroad' ...
remote: Enumerating objects: 577, done.
remote: Counting objects: 100% (577/577), done.
remote: Compressing objects: 100% (382/382), done.
remote: Total 26081 (delta 366), reused 333 (delta 174), pack-reused 25504
Empfange Objekte: 100% (26081/26081), 52.80 MiB | 1.67 MiB/s, Fertig.
Löse Unterschiede auf: 100% (17187/17187), Fertig.
Checke Dateien aus: 100% (1863/1863), Fertig.
pi@RailPi:~$ cd railroad/can2udp/src
pi@RailPi:~/railroad/can2udp/src$ make
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -c -o can-bounce-test.o can-bounce-test.c
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -o can-bounce-test can-bounce-test.o
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -c -o crc-ccitt.o crc-ccitt.c
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -c -o lib.o lib.c
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -c -o can-monitor.o can-monitor.c
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -o can-monitor crc-ccitt.o lib.o can-monitor.o -lpcap
cc -g -O2 -fstack-protector-strong -Wformat -Werror=format-security -Wall -Wextra -Wmaybe-uninitialized -std=gnu99 -pedantic -rrors -o can-monitor can-monitor.o
```

Dann sorgen wir dafür, daß die für uns wichtigen Programme can2lan und can-monitor von überall her verfügbar sind (aus Gründen der Übersicht soll hier auf die ScreenShots verzichtet werden):

```
sudo cp can2lan /usr/sbin/  
sudo cp can-monitor /usr/bin/  
sudo cp ../files/maerklin/config/gleisbild.cs2 /var/www/html/gleisbild.cs2
```

```
# Debian Init Skript installieren  
sudo cp ../debian/can2lan.init /etc/init.d  
sudo /etc/init.d/can2lan.init start  
sudo /etc/init.d/can2lan.init status  
sudo /etc/init.d/can2lan.init stop
```

```
sudo update-rc.d can2lan.init defaults
```

Aus <<https://github.com/GBert/misc/tree/master/RPi-MCP2515>>

Dann wird neu gestartet (shutdown -r now)c, die Konsole aufgerufen und mit can-monitor getestet:

```
pi@RailPi:~$ can-monitor  
17:38:39.469 CAN 0x0008BB14 [6] 00 00 40 05 00 06 Lok mfx-5 Geschwindigkeit: 0.6  
17:38:39.469 CAN 0x0009FF1C [6] 00 00 40 05 00 06 Lok mfx-5 Geschwindigkeit: 0.6  
17:38:39.591 CAN 0x0008BB14 [6] 00 00 40 05 00 42 Lok mfx-5 Geschwindigkeit: 6.6  
17:38:39.592 CAN 0x0009FF1C [6] 00 00 40 05 00 42 Lok mfx-5 Geschwindigkeit: 6.6  
17:38:39.739 CAN 0x0008BB14 [6] 00 00 40 05 00 7E Lok mfx-5 Geschwindigkeit: 12.6  
17:38:39.739 CAN 0x0009FF1C [6] 00 00 40 05 00 7E Lok mfx-5 Geschwindigkeit: 12.6  
17:38:40.245 CAN 0x0008BB14 [6] 00 00 40 05 00 78 Lok mfx-5 Geschwindigkeit: 12.0  
17:38:40.245 CAN 0x0009FF1C [6] 00 00 40 05 00 78 Lok mfx-5 Geschwindigkeit: 12.0  
17:38:40.347 CAN 0x0008BB14 [6] 00 00 40 05 00 0C Lok mfx-5 Geschwindigkeit: 1.2  
17:38:40.348 CAN 0x0009FF1C [6] 00 00 40 05 00 0C Lok mfx-5 Geschwindigkeit: 1.2  
17:38:40.459 CAN 0x0008BB14 [6] 00 00 40 05 00 00 Lok mfx-5 Geschwindigkeit: 0.0  
17:38:40.460 CAN 0x0009FF1C [6] 00 00 40 05 00 00 Lok mfx-5 Geschwindigkeit: 0.0  
17:38:41.382 CAN 0x0030BB14 [0] Ping Anfrage  
17:38:41.383 CAN 0x0031FF1C [8] 47 45 58 B8 01 29 00 10 Ping Antwort von Gleisbox UID 0x47455BB8, Software Version 1.41  
17:38:42.212 CAN 0x0008BB14 [5] 00 00 00 00 00 00 System: alle Stopp  
17:38:42.213 CAN 0x0001FF1C [5] 00 00 00 00 00 00 System: alle Stopp  
17:38:42.325 CAN 0x00230B01 [8] 00 00 00 09 01 00 2D 18 S88 Event Kennung 0 Kontakt 9 Zustand alt 1 Zustand neu 0 Zeit 1154  
17:38:42.325 CAN 0x00230B01 [8] 00 00 00 0A 01 00 2D 18 S88 Event Kennung 0 Kontakt 10 Zustand alt 1 Zustand neu 0 Zeit 1154  
17:38:42.326 CAN 0x00230B01 [8] 00 00 00 0C 01 00 2D 18 S88 Event Kennung 0 Kontakt 12 Zustand alt 1 Zustand neu 0 Zeit 1154  
17:38:42.326 CAN 0x00230B01 [8] 00 00 00 0D 01 00 2D 18 S88 Event Kennung 0 Kontakt 13 Zustand alt 1 Zustand neu 0 Zeit 1154  
17:38:42.327 CAN 0x00230B01 [8] 00 00 00 0E 01 00 2D 18 S88 Event Kennung 0 Kontakt 14 Zustand alt 1 Zustand neu 0 Zeit 1154  
17:38:42.328 CAN 0x00230B01 [8] 00 00 00 0B 01 00 2D 18 S88 Event Kennung 0 Kontakt 11 Zustand alt 1 Zustand neu 0 Zeit 1154  
17:38:43.362 CAN 0x0008BB14 [7] 00 00 00 00 09 00 0B System: Neuanmeldezähler setzen UID 0x00000000 Zähler 0x000B  
17:38:43.363 CAN 0x0008BB14 [6] 00 00 00 00 08 02 System: Gleisprotokoll freischalten - MFX  
17:38:43.363 CAN 0x0001FF1C [7] 00 00 00 00 09 00 0B System: Neuanmeldezähler setzen UID 0x00000000 Zähler 0x000B  
17:38:43.364 CAN 0x0008BB14 [5] 00 00 00 00 01 System: alle Go  
17:38:43.364 CAN 0x0001FF1C [6] 00 00 00 00 08 02 System: Gleisprotokoll freischalten - MFX  
17:38:43.365 CAN 0x0001FF1C [5] 00 00 00 00 01 System: alle Go  
17:38:44.135 CAN 0x00230B01 [8] 00 00 00 09 00 01 00 B5 S88 Event Kennung 0 Kontakt 9 Zustand alt 0 Zustand neu 1 Zeit 181  
17:38:44.135 CAN 0x00230B01 [8] 00 00 00 0A 00 01 00 B5 S88 Event Kennung 0 Kontakt 10 Zustand alt 0 Zustand neu 1 Zeit 181  
17:38:44.136 CAN 0x00230B01 [8] 00 00 00 0C 00 01 00 B5 S88 Event Kennung 0 Kontakt 12 Zustand alt 0 Zustand neu 1 Zeit 181  
17:38:44.136 CAN 0x00230B01 [8] 00 00 00 0D 00 01 00 B5 S88 Event Kennung 0 Kontakt 13 Zustand alt 0 Zustand neu 1 Zeit 181
```

Damit rasch Werte dargestellt werden, kann einfach einmal an einer angeschlossenen MS2 'gespielt' werden...

Das war's - der CAN-Bus läuft und wird vom RailPi über das Programm can2lan ausgelesen und beschrieben. Auf die Werte vom und zum CAN-Bus kann über die IP-Adresse 192.168.178.58 (die IP-Adresse des RailPi's) zugegriffen werden.

Dies erfolgt durch RailControl, dessen Installation wir uns im folgenden Kapitel anschauen.

3.3 - Installation von RailControl

Teddy's RailControl stellt nicht nur einen Leitrechner zur Koordinierung des Schienenverkehrs dar, sondern unterstützt den Anwender mit einem Web-Interface zur Visualisierung der Modellbahnautomatisierung.

Gleichzeitig kommuniziert das Programm mit der von Gerd geschriebenen CS2-Emulation, die im Kapitel zuvor installiert wurde.

Kommen wir also zunächst zur Installation von RailControl.

Teddy hat auf seiner Seite den Quellcode sowie die Informationen zur Installation abgelegt:

Kompilieren unter Linux bzw. unter einer Posix-Umgebung

Auf debian-basierten System können die nötigen Entwickler-Werkzeuge in einem Terminal folgendermassen installiert werden (je nach Distribution kann dies etwas abweichen):

```
sudo apt-get install g++ binutils make git
```

Nach der Installation der Entwickler-Werkzeuge können die Sourcen geholt werden:

```
git clone https://github.com/teddych/railcontrol.git
```

Im neu erstellten Verzeichnis railcontrol kann RailControl kompiliert werden:

```
make
```

Anschliessend ist die Datei railcontrol zu starten:

```
./railcontrol
```

Aktualisieren

Ein Update kann folgendermassen durchgeführt werden:

```
git pull
```

```
make
```

Aus <<https://www.railcontrol.org/index.php/de/source-code-de>>

Und so schaut's dann auf dem RailPi aus:

Die Entwickler-Werkzeuge sind bereits installiert, wir rufen die Installation aber trotzdem auf

Hinweis:

die Ausführung des make-Befehls dauert auf einem Raspberry Pi 3 ca. 30 Minuten - Zeit für einen Screen-Shot

Während dieser Zeit wurde eine zweite Konsole geöffnet und der CAN-monitor gestartet - dies zeigt uns auch, daß 'can2lan' bereits ohne weiteres Zutun unsererseits nach dem Systemneustart arbeitet.


```
pi@RailPi: ~/railcontrol
Datei Bearbeiten Reiter Hilfe

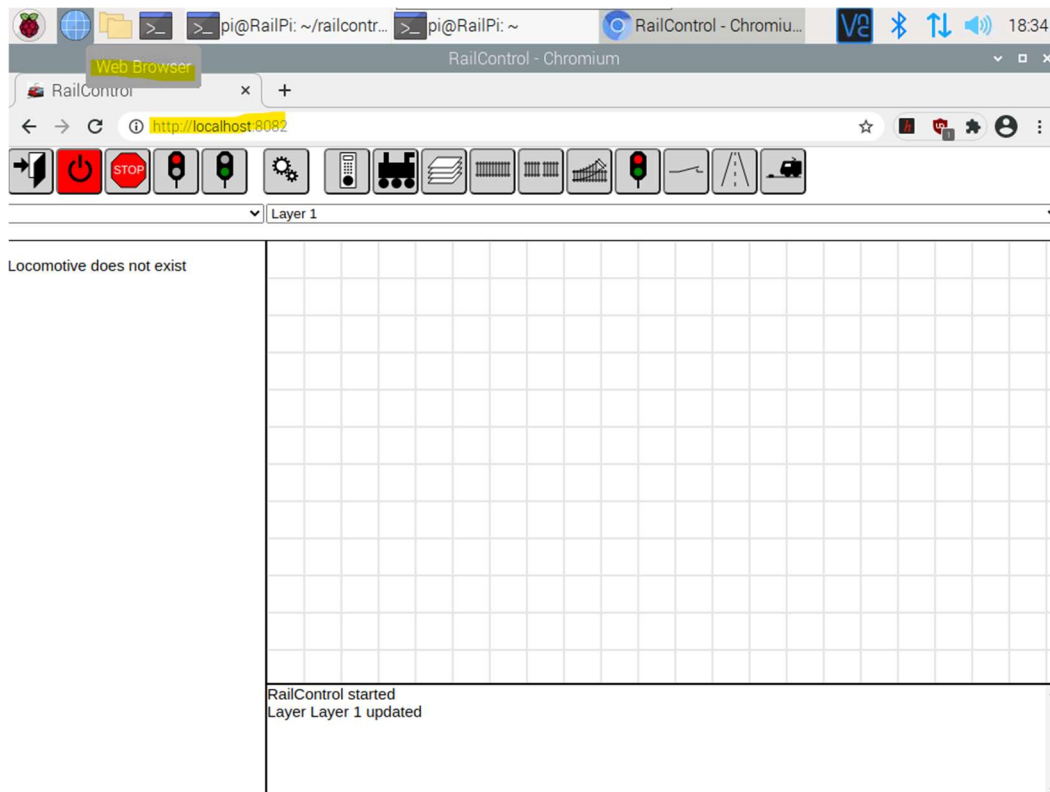
pi@RailPi:~$ sudo apt-get install g++ binutils make git
Paketlisten werden gelesen... Fertig
Abhängigkeitsbaum wird aufgebaut.
Statusinformationen werden eingelesen.... Fertig
binutils ist schon die neueste Version (2.31.1-16+rp12).
binutils wurde als manuell installiert festgelegt.
g++ ist schon die neueste Version (4:8.3.0-1+rp12).
g++ wurde als manuell installiert festgelegt.
git ist schon die neueste Version (1:2.20.1-2+deb10u3).
make ist schon die neueste Version (4.2.1-1.2).
make wurde als manuell installiert festgelegt.
0 aktualisiert, 0 neu installiert, 0 zu entfernen und 0 nicht aktualisiert.
pi@RailPi:~$ git clone https://github.com/teddych/railcontrol.git
Klone nach 'railcontrol' ...
remote: Enumerating objects: 289, done.
remote: Counting objects: 100% (289/289), done.
remote: Compressing objects: 100% (184/184), done.
remote: Total 11784 (delta 193), reused 181 (delta 105), pack-reused 11495
Empfange Objekte: 100% (11784/11784), 11.21 MiB | 1.57 MiB/s, Fertig.
Lose Unterschiede auf: 100% (9540/9540), Fertig.
pi@RailPi:~$ cd railcontrol
pi@RailPi:~/railcontrol$ make
cc -g -O2 -DSQLITE_ENABLE_FTS4 -DSQLITE_ENABLE_JSON1 -DSQLITE_ENABLE_RTREE -DHAVE_USLEEP -c -o Storage/sqlite/sqlite3.o Storage/sqlite/sqlite3.c
g++ -I. -g -O2 -Wall -Wextra -pedantic -Werror -std=c++11 -c -o ArgumentHandler.o ArgumentHandler.cpp
g++ -I. -g -O2 -Wall -Wextra -pedantic -Werror -std=c++11 -c -o Config.o Config.cpp
g++ -I. -g -O2 -Wall -Wextra -pedantic -Werror -std=c++11 -c -o Languages.o Languages.cpp
g++ -I. -g -O2 -Wall -Wextra -pedantic -Werror -std=c++11 -c -o Manager.o Manager.cpp
g++ -I. -g -O2 -Wall -Wextra -pedantic -Werror -std=c++11 -c -o RailControl.o RailControl.cpp
[...]
```

Und dann ist es so weit:

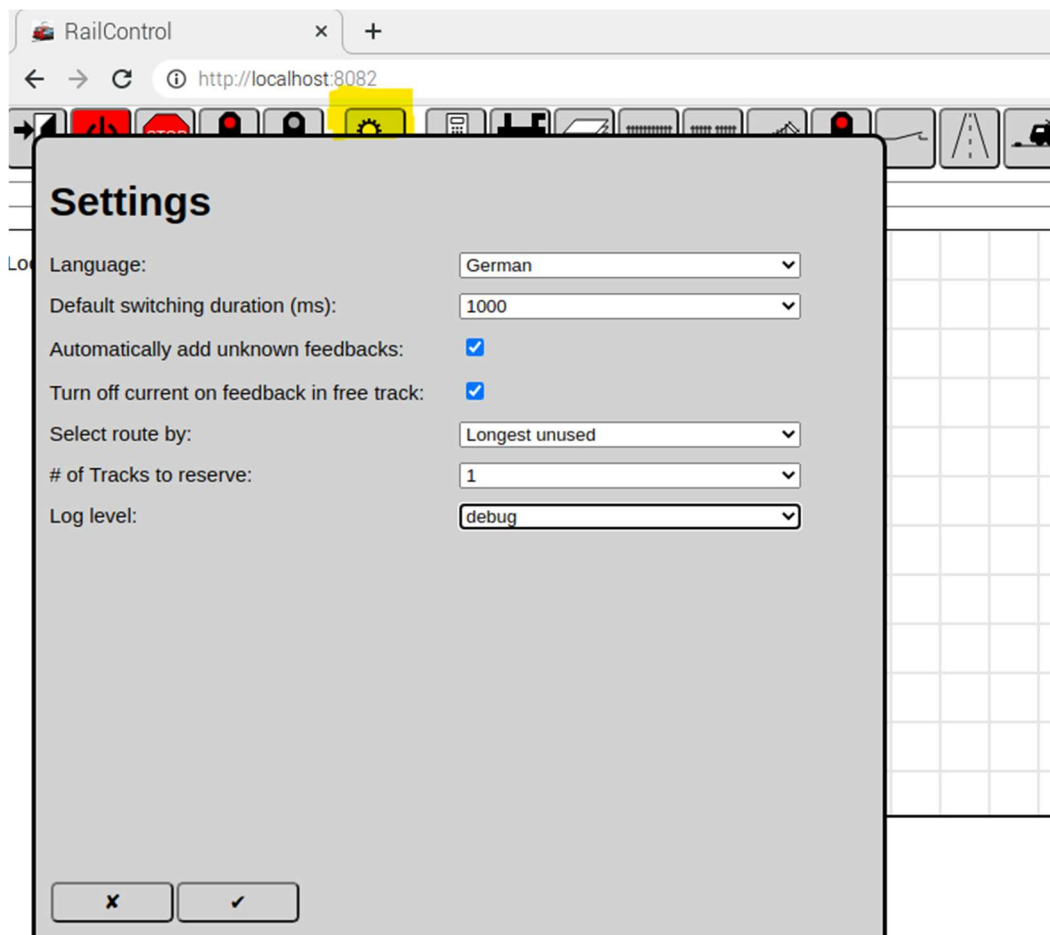
nach ca. einer halben Stunde kann der erste Aufruf von Railcontrol in der Form './railcontrol' erfolgen:

```
hardware/zlib/gzlib.o Hardware/zlib/uncompr.o Hardware/zlib/adler32.o Hardware/zlib/zutil.o Hardware/zlib/inffast.o Hardware/zlib/gzread.o Hardware/zlib/crc32.o Hardware/zlib/gzwrite.o Hardware/zlib/inflate.o Hardware/zlib/inffast.o Hardware/zlib/trees.o Hardware/zlib/inftrees.o Hardware/zlib/gzclose.o Hardware/zlib/compress.o -o railcontrol -lpthread -ldl
pi@RailPi:~/railcontrol$ ./railcontrol
2021-04-11 16:31:18.155979: Info: Main: Starting RailControl
2021-04-11 16:31:18.156861: Info: Main: Version: 19
2021-04-11 16:31:18.157046: Info: Main: Compile date: 2021-04-11 18:08:42
2021-04-11 16:31:18.157109: Info: Main: Last GIT commit hash: 2c6733f2ea2a0c338d56a7f8c9904ee1d6291b71
2021-04-11 16:31:18.157196: Info: Main: Last GIT commit date: 2021-04-11 17:47:45
2021-04-11 16:31:18.157325: Info: Main: Copying from railcontrol.conf.dist to railcontrol.conf
2021-04-11 16:31:18.163087: Info: Config: Reading config file railcontrol.conf
2021-04-11 16:31:18.163471: Info: Config: Parameter found in config file: dbfilename = railcontrol.sqlite
2021-04-11 16:31:18.163582: Info: Config: Parameter found in config file: dbkeepbackups = 10
2021-04-11 16:31:18.163698: Info: Config: Parameter found in config file: webserverport = 8082
2021-04-11 16:31:18.164237: Info: SQLite: Opening SQLite database with filename railcontrol.sqlite
2021-04-11 16:31:18.167217: Info: SQLite: Creating table hardware
2021-04-11 16:31:18.188950: Info: SQLite: Creating table objects
2021-04-11 16:31:18.215858: Info: SQLite: Creating table relations
2021-04-11 16:31:18.237641: Info: SQLite: Creating table settings
2021-04-11 16:31:18.262886: Info: WebServer: Webserver started
2021-04-11 16:31:18.263724: Info: WebServer: Please type one of the following links in your browser to connect to RailControl:
    http://localhost:8082/
    http://127.0.0.1:8082/
    http://192.168.178.58:8082/
    http://[:1]:8082/
    http://[2a02:908:2f32:2c60:68a2:55b4:e7d0:49ed]:8082/
    http://[fe80::91ce:1d7f:846e:b536]:8082/
2021-04-11 16:31:18.287622: Info: Manager: Debounce thread started
```

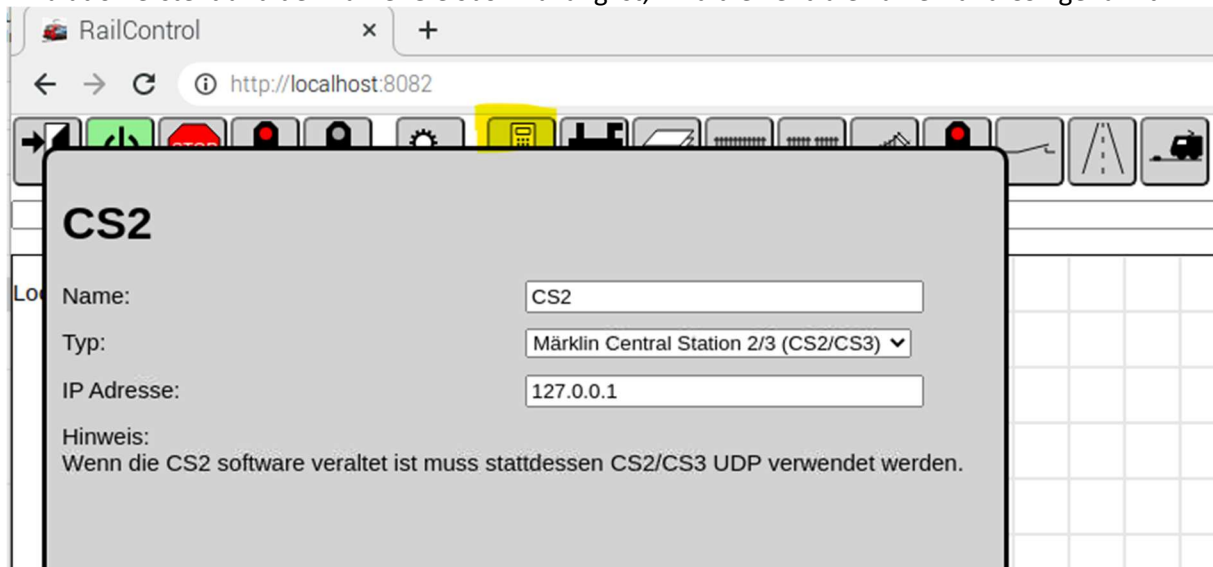

Nun kann mit dem Chromium Webbrowser und der Adresse localhost:8082 vom RailPi oder unter 192.168.178.58:8082 von jedem Browser im Netzwerk aus auf RailControl zugegriffen werden:



Und die wichtigsten Einstellungen werden gesetzt:



Und - ganz wichtig - die Verbindung zur Zentrale (Gleisbox) wird hergestellt. Da CAN2LAN eine CS2-Emulation erstellt und der Name 'Gleisbox' zu lang ist, wird die Zentrale kurzerhand CS2 genannt:



Zum Schluß noch ein Tip, wie eine Lokadresse sehr einfach abgelesen werden kann:

```
pi@RailPi: ~
Datei Bearbeiten Reiter Hilfe
18:55:25.182 CAN 0x000DFF1C [6] 00 00 40 07 08 00 Lok mfx-7 Funktion 8 Wert 0
18:55:25.182 CAN 0x000CBB14 [5] 00 00 40 07 0A Lok mfx-7 Funktion 10
18:55:25.182 CAN 0x000DFF1C [6] 00 00 40 07 09 00 Lok mfx-7 Funktion 9 Wert 0
18:55:25.183 CAN 0x000CBB14 [5] 00 00 40 07 0B Lok mfx-7 Funktion 11
18:55:25.183 CAN 0x000DFF1C [6] 00 00 40 07 0A 00 Lok mfx-7 Funktion 10 Wert 0
18:55:25.183 CAN 0x000CBB14 [5] 00 00 40 07 0C Lok mfx-7 Funktion 12
18:55:25.184 CAN 0x000DFF1C [6] 00 00 40 07 0B 00 Lok mfx-7 Funktion 11 Wert 0
18:55:25.185 CAN 0x000CBB14 [5] 00 00 40 07 0D Lok mfx-7 Funktion 13
18:55:25.185 CAN 0x000DFF1C [6] 00 00 40 07 0C 00 Lok mfx-7 Funktion 12 Wert 0
18:55:25.185 CAN 0x000CBB14 [5] 00 00 40 07 0E Lok mfx-7 Funktion 14
18:55:25.186 CAN 0x000DFF1C [6] 00 00 40 07 0D 00 Lok mfx-7 Funktion 13 Wert 0
18:55:25.186 CAN 0x000CBB14 [5] 00 00 40 07 0F Lok mfx-7 Funktion 15
18:55:25.187 CAN 0x000DFF1C [6] 00 00 40 07 0E 00 Lok mfx-7 Funktion 14 Wert 0
18:55:25.187 CAN 0x000DFF1C [6] 00 00 40 07 0F 00 Lok mfx-7 Funktion 15 Wert 0
18:55:25.234 CAN 0x000CBB14 [4] 00 00 40 06 Lok mfx-6 Abfrage Fahrstufe
18:55:25.235 CAN 0x000ABB14 [4] 00 00 40 06 Lok mfx-6 Richtung wird abgefragt
18:55:25.235 CAN 0x0009FF1C [6] 00 00 40 06 00 00 Lok mfx-6 Geschwindigkeit: 0.0
18:55:25.235 CAN 0x0008FF1C [5] 00 00 40 06 01 Lok mfx-6 Richtung: vorwärts
18:55:25.236 CAN 0x000CBB14 [5] 00 00 40 06 00 Lok mfx-6 Funktion 0
18:55:25.237 CAN 0x000CBB14 [5] 00 00 40 06 01 Lok mfx-6 Funktion 1
18:55:25.237 CAN 0x000DFF1C [6] 00 00 40 06 00 00 Lok mfx-6 Funktion 0 Wert 0
18:55:25.238 CAN 0x000CBB14 [5] 00 00 40 06 02 Lok mfx-6 Funktion 2
18:55:25.239 CAN 0x000DFF1C [6] 00 00 40 06 01 00 Lok mfx-6 Funktion 1 Wert 0
18:55:25.239 CAN 0x000CBB14 [5] 00 00 40 06 03 Lok mfx-6 Funktion 3
18:55:25.240 CAN 0x000DFF1C [6] 00 00 40 06 02 00 Lok mfx-6 Funktion 2 Wert 0
18:55:25.241 CAN 0x000CBB14 [5] 00 00 40 06 04 Lok mfx-6 Funktion 4
18:55:25.241 CAN 0x000DFF1C [6] 00 00 40 06 03 00 Lok mfx-6 Funktion 3 Wert 0
18:55:25.241 CAN 0x000CBB14 [5] 00 00 40 06 05 Lok mfx-6 Funktion 5
18:55:25.242 CAN 0x000DFF1C [6] 00 00 40 06 04 00 Lok mfx-6 Funktion 4 Wert 0
18:55:25.242 CAN 0x000CBB14 [5] 00 00 40 06 06 Lok mfx-6 Funktion 6
18:55:25.242 CAN 0x000DFF1C [6] 00 00 40 06 05 00 Lok mfx-6 Funktion 5 Wert 0
18:55:25.242 CAN 0x000CBB14 [5] 00 00 40 06 07 Lok mfx-6 Funktion 7
18:55:25.243 CAN 0x000DFF1C [6] 00 00 40 06 06 00 Lok mfx-6 Funktion 6 Wert 0
18:55:25.244 CAN 0x000DFF1C [6] 00 00 40 06 07 00 Lok mfx-6 Funktion 7 Wert 0
18:55:25.485 CAN 0x000CBB14 [4] 00 00 40 05 Lok mfx-5 Abfrage Fahrstufe
18:55:25.485 CAN 0x000ABB14 [4] 00 00 40 05 Lok mfx-5 Richtung wird abgefragt
18:55:25.486 CAN 0x0009FF1C [6] 00 00 40 05 00 00 Lok mfx-5 Geschwindigkeit: 0.0
18:55:25.486 CAN 0x0008FF1C [5] 00 00 40 05 01 Lok mfx-5 Richtung: vorwärts
18:55:25.487 CAN 0x000CBB14 [5] 00 00 40 05 00 Lok mfx-5 Funktion 0
18:55:25.487 CAN 0x000CBB14 [5] 00 00 40 05 01 Lok mfx-5 Funktion 1
18:55:25.488 CAN 0x000DFF1C [6] 00 00 40 05 00 00 Lok mfx-5 Funktion 0 Wert 0
18:55:25.488 CAN 0x000CBB14 [5] 00 00 40 05 02 Lok mfx-5 Funktion 2
18:55:25.489 CAN 0x000DFF1C [6] 00 00 40 05 01 00 Lok mfx-5 Funktion 1 Wert 0
18:55:25.489 CAN 0x000CBB14 [5] 00 00 40 05 03 Lok mfx-5 Funktion 3
18:55:25.490 CAN 0x000DFF1C [6] 00 00 40 05 02 00 Lok mfx-5 Funktion 2 Wert 0
```

Mit der MS2 werden nacheinander die gespeicherten Loks ausgewählt und ein Fahrtbefehl vorgegeben. Und schon zeigt der CAN-Monitor die passende mfx-Adresse zu dieser Lok.

Aber zu diesen Themen mehr in der eigentlichen RailControl-Doku - ein use-case von mir hierzu steht noch aus ;)

4 – Ausblick

Gern würde ich in diesem Stil eine Fortsetzung schreiben, die sich dann aber nicht mit Alternativen zu den geschilderten Möglichkeiten befaßt (davon gibt es ganz einfach beliebig viele), sondern die eine wirkliche **Fortsetzung im Hinblick auf die Nutzung von RailControl** darstellt.

Einen UseCase also im Stil: 'wie realisiere ich auf meiner Modellbahn einen sicheren und automatischen Fahrbetrieb und stelle gleichzeitig parallel dazu einen manuellen Teilbetrieb sicher'.

So ganz schnell wird's nicht gehen - aber Ihr müßt ja auch erst noch einen Raspberry Pi ertüchtigen ;)

... wobei es aber bei der Fortsetzung überhaupt nicht mehr darauf ankommt: auf welchem Boliden RailControl zum Laufen kommt: ein Raspi genügt ganz einfach - doch seht selbst, der Einsatz ist nicht hoch und lohnt sich

Sollten sich Flüchtigkeitsfehler oder auch logische und/oder sachliche Fehler in meine Beschreibung eingeschlichen haben: Bitte meldet Euch - ich stelle das dann umgehend ab.