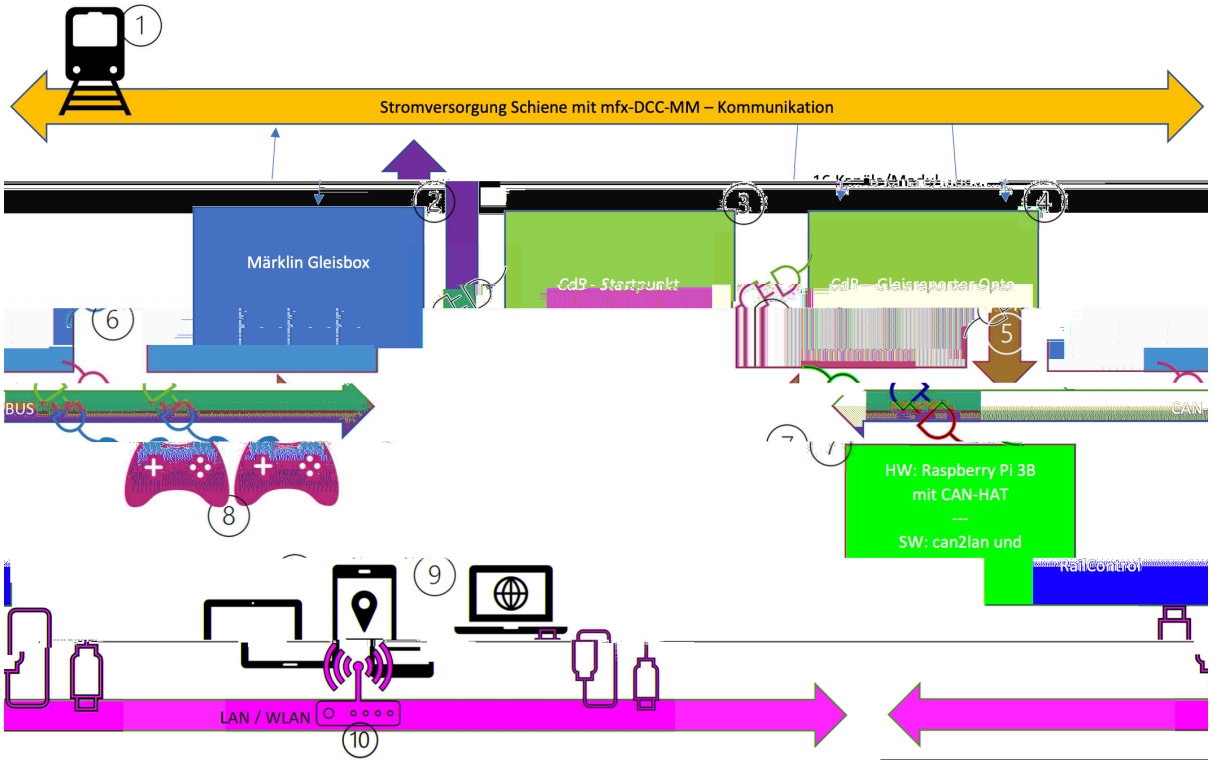
RailControl

Einrichtung des RailPi Vers. 2.1.1



Detlef Behlert

Winter 2022

RailPi basierend auf Diet-Pi

Weniger ist mehr!

Vorbemerkungen:

Dieses Dokument ist ein weiteres Dokument einer zukünftigen Reihe über RailControl auf dem RailPi und basiert auf folgendem früheren Dokument:

RailPi Vers.1.1
https://www.railcontrol.org/forum/download/file.php?id=62
(Das dortige Kapitel 3 wird durch dieses Dokument ersetzt - die Informationen um die Hardware bleiben gleich - wir steigen lediglich **von PiOS auf Diet-Pi-OS** um)

Das obige Dokument zu kennen, ist in Ordnung. Als Hintergrund-information zur eingesetzten Hardware auch wichtig. Aber auch dort steht eine Überarbeitung in Kürze an.

Bitte beachten:
Alle notwendigen Schritte, den RailPi Vers. 2.1 einzurichten, finden sich in **diesem** Dokument. **Allein in diesem Dokument.**

Bitte auch den Disclaimer am Ende dieses Dokumentes beachten!

Des weiteren:
eine schnelle 4 oder 8 GB-mikroSD-Karte genügt. Aber: schnell muss sie sein - nicht nur beim Lesen, sondern auch beim Schreiben!

Aber aufpassen: bei einer 4GB-mikroSD-Karte gibt's kaum noch Reserven!!! (für ein BackUp o.ä. beispielsweise)

Idealerweise wird also die vorhandene RailPi-Installation nicht überschrieben, sondern (sicherheitshalber) eine neue micro-SD-Karte genutzt. Überschreiben kann man später immer noch…
… wenn es funktioniert (und gefällt).

Es wird KEIN grafisches UI (UserInterface,GUI) genutzt - alles läuft maximal schnell auf dem Pi und wird vom Terminal eines PC's aus bedient - deshalb der SSH-Zugriff von einem anderen PC aus! Oder von einer SSH-App im Pad…

An dieser Stelle sei noch einmal darauf hingewiesen, daß KEINE zusätzliche Software installiert werden sollte, denn unser Pi ist ja ein **RailPi** und soll alle seine Ressourcen in das Handling der **CAN-Schnittstelle** via **can2lan** und das Steuern unserer Modellbahnen via **RailControl** über das **LAN** stecken**!**

Für einen Betrieb in einem Netzwerk mit anderweitig hoher Auslastung sollte darüber nachgedacht werden, ob man ggf. ein eigenes Sub-Netz für die Modellbahnsteuerung aufmacht.

Auf jeden Fall ist dem kabelgebundenen Netz der Vorrang vor einer Wifi-Anbindung zu geben!

Also:
Je mehr Softwarepakete wir im RailPi installieren und je mehr Netzwerkverkehr wir (gerade über Wifi) zulassen, desto eher bekommen wir u.U. später eine Betriebsstörung!

Denkt bei dieser Gelegenheit doch einfach mal darüber nach, ob Ihr nicht doch noch einen alten (wifi-)Router in der Bastelkiste liegen habt, um ein Sub-Netz für die Modellbahn aufzumachen…

> Hier findet Ihr die Verknüpfung zur DietPi - Originaldokumentation:
> https://dietpi.com/docs/install/
>
> Sowie zu einem sehr interessanten YouTube
> ‚diet-pi - Installationsvideo‘, deutsch
> https://www.youtube.com/watch?v=J5yPeJFLSO0&list=PLQIL7cyHMGboXtOzwAcX4hGPW6ECbVinp&index=8

Jetzt kann es - endlich - losgehen!

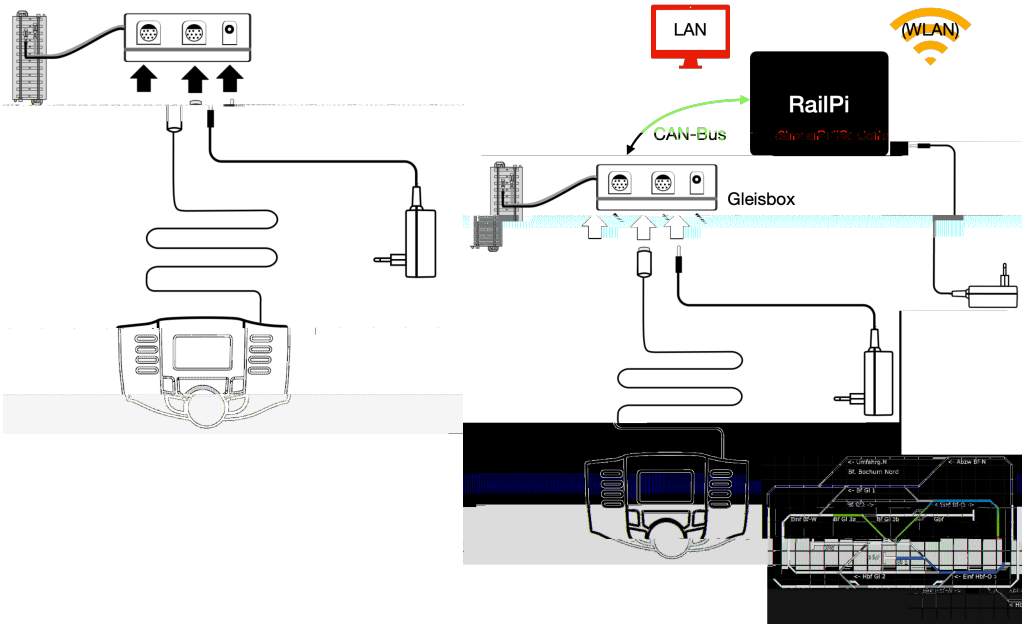
Was will ich mit diesem System (RailPi) erreichen?

Mein Ziel ist es, einen (minimalen) Rechner zu ertüchtigen, aus einer normalen Märklin-Startpackung heraus ein (fast beliebig) erweiterbares System aufzubauen, um damit entweder

- ohne weitere externe Hardware eine bedienbare Stellwerksicht

- oder - mit externer Hardware - einen Automatikbetrieb zu ermöglichen.

Links unten: Ausgangssituation - eine Märklin H0-Startpackung



Rechts oben: Das Zwischenziel, allein durch den Einsatz eines RailPi (dieses Dokument)

Auf die Hardware bezogen heißt das

HW RailPi = RaspberryPi + Welectron CAN-HAT + Märklin HO Startpackung
https://www.waveshare.com/wiki/2-CH_CAN_HAT

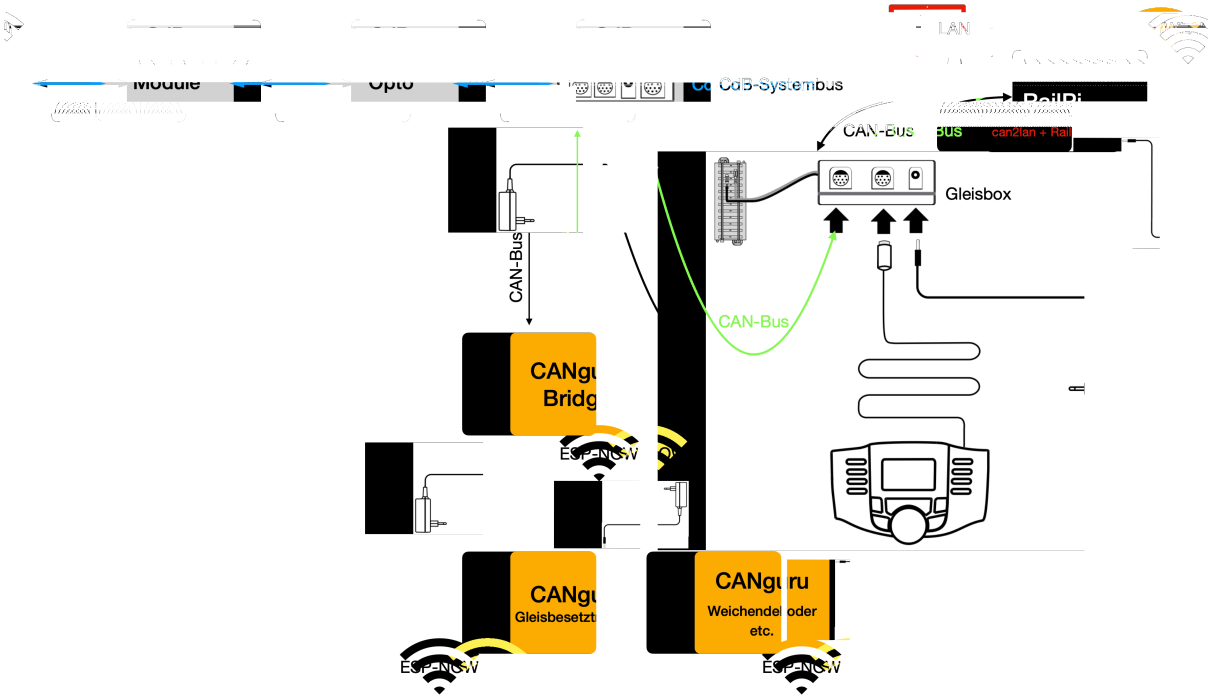
In diesem Dokument geht es vorrangig um die Installation und Konfiguration der dazu notwendigen OpenSource-Software:

SW RailPi = Debian Linux in der Ausprägung diet-pi 8.0 + can2lan + RailControl.

Das spätere Ziel wird sein:
(noch nicht Gegenstand dieses Dokumentes, aber ansatzweise bereits in https://www.railcontrol.org/forum/download/file.php?id=62 enthalten)

universelle Modellbahnsteuerung unter Nutzung weiterer kommerzieller und/oder freier Hard- und Softwarekomponenten für alle Betriebsarten

Konkret am Beispiel der Komponenten vom CAN-digital-Bahn-Projekt CdB und dem CANguru-System:

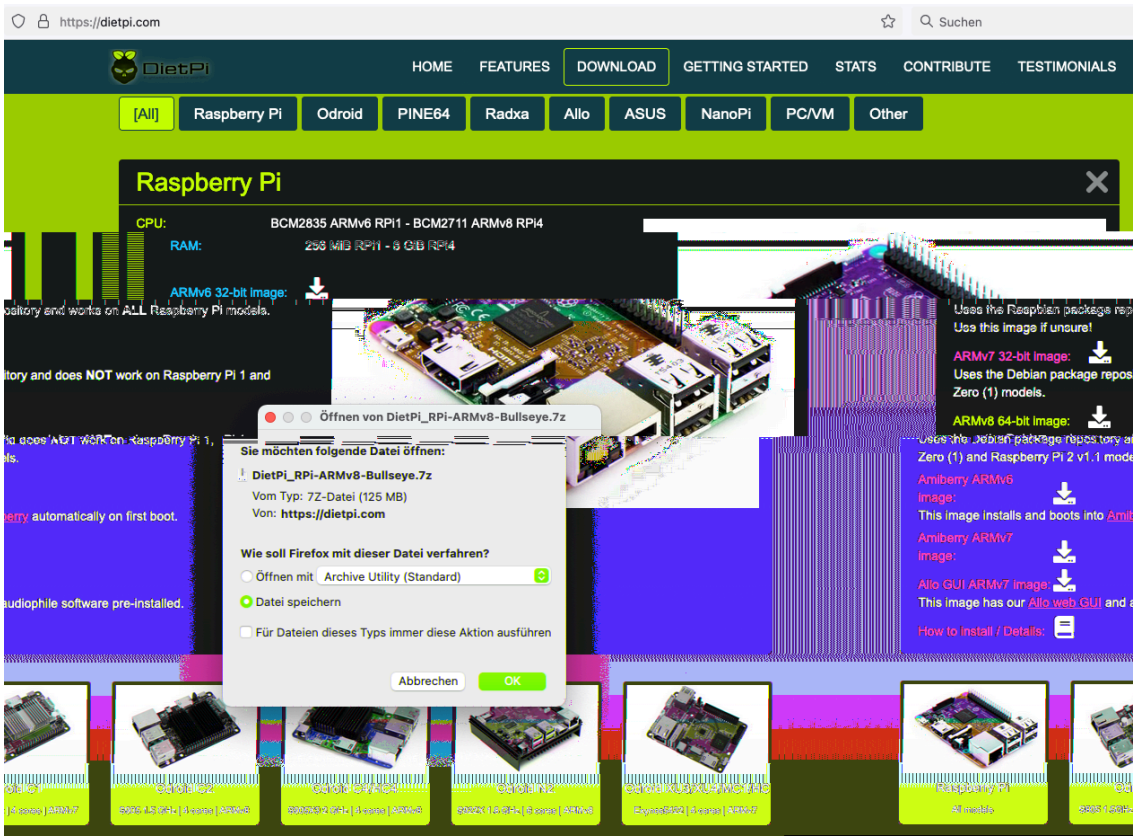


Basis-Installation des Betriebssystems DietPi

Mit einem vorhandenen PC - in meinem Fall einem Mac wird das passende Diet-Pi-Image heruntergeladen:

https://dietpi.com

Da ich einen Raspberry 3B einsetze, wähle ich das Image ARMv8 64bit-Image



Jetzt sind folgende Schritte auszuführen:

1.
diet-pi - Daten herunterladen, 7z-Datei entpacken und das Image auf SD-Karte schreiben

2.
die microSD-Karte in den Pi einlegen, den Pi mit dem LAN verbinden und einschalten

3.
über eine LAN-Verbindung sozusagen ‚vollautomatisch' den Pi mit dem PC verbinden

Wichtig:
der RaspberryPi ist mit dem LAN verbunden!
auf dem PC ist ein SSH-Client vorhanden

Die Verbindung mit dem RailPi (der ja ohne Monitor, Maus und Tastatur betrieben werden soll), geschieht über eine SSH-Verbindung.
Im folgenden Fall über einen Mac mit dem Namen ‚MBP' und den Mac-user ‚detlef'

Zu 1:
In dem jetzt vorhandenen Ordner ‚**DietPi_RPi-ARMv8-Bullseye**' kann die darin enthaltene gleichnamige .img-Datei mit einem geeigneten Tool (z.B. Balena Etcher oder auch dem vielleicht aus früherem Umgang mit dem Raspberry Pi bekannten Tool) dem ‚**Raspberry Pi Imager**' auf die SD-Karte geschrieben werden:

Zu 3:
Nach dem Bootvorgang des ‚neuen' Pi's wird ab jetzt via SSH-remote-Zugriff (z.B. vom PC aus) auf den RaspberryPi zugegriffen.

Host-Name: dietpi
login: root
password: dietpi

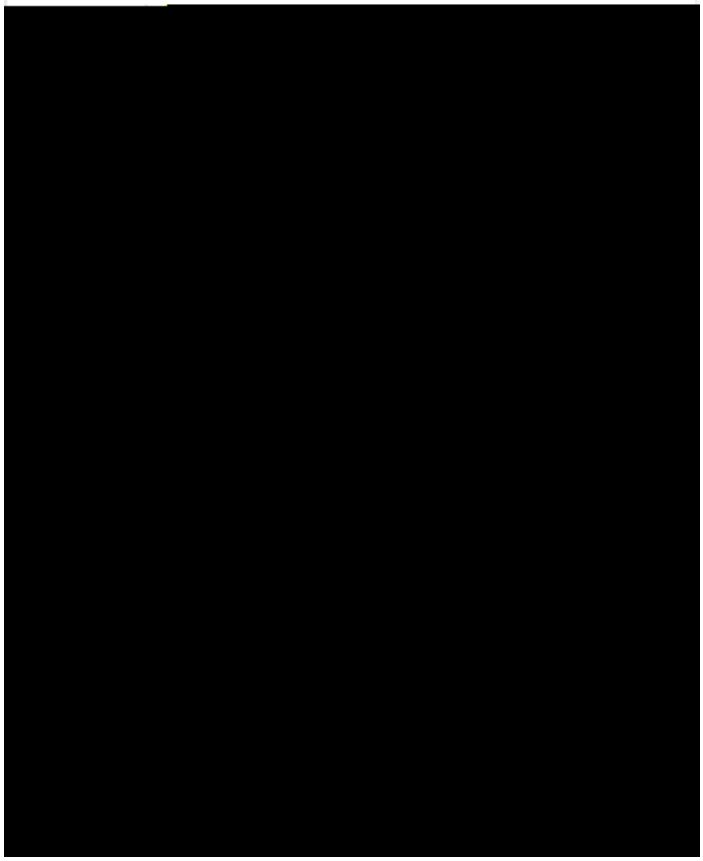
Natürlich kann der RailPi auch mit Monitor, Tastatur und Maus bestückt werden, dann kann selbstverständlich auf die SSH-Verbindung verzichtet werden.

Aber Ziel dieser Doku ist es ja, eine MINIMALE Konfiguration eines Modellbahn-Steuerungsrechners zu installieren, um für die wirklich notwendigen Programme eine MAXIMALE Performance zu erreichen!

Nach Anlegen der Versorgungsspannung melden wir uns also z.B. über SSH am künftigen RailPi an. Momentan heißt er per Voreinstellung noch ‚dietpi':

```
detlef@MBP ~ % ssh root@dietpi
Warning: Permanently added the ECDSA host key for IP address
 '192.168.178.65' to the list of known hosts.
root@dietpi's password:
```

Und los gehts:



Mit Bestätigung der Lizenzbedingungen über <TAB> und <OK> wird das DietPi bzw. das DebianOS, auf dem DietPi ja basiert, erst einmal aktualisiert:

```
 DietPi-Update
-------------------------------------------------------------
 Phase: Checking for available DietPi update

[  OK  ] DietPi-Update | Checking network connectivity
[  OK  ] DietPi-Update | Checking DNS resolver
[  OK  ] Network time sync | Completed
[ INFO ] DietPi-Update | Getting latest version from: https://raw.
githubusercontent.com/MichaIng/DietPi/master/.update/version
[  OK  ] DietPi-Update | Got valid latest version: 8.0.2
[  OK  ] DietPi-Update | Update available:
[ INFO ] DietPi-Update | Current version : v7.9.3
[ INFO ] DietPi-Update | Latest version  : v8.0.2

 DietPi-Update
-------------------------------------------------------------
 Phase: Checking for update pre-requirements

[  OK  ] DietPi-Update | DietPi-Userdata validation: /mnt/dietpi_u
serdata
[  OK  ] DietPi-Update | Free space check: path=/ | available=2701
8 MiB | required=100 MiB
[ SUB1 ] DietPi-Services > stop
[ INFO ] DietPi-Services | skip : cron (due to mask)

 DietPi-Update
-------------------------------------------------------------
 Phase: Applying pre-patches

[  OK  ] DietPi-Update | Downloading pre-patches
[  OK  ] DietPi-Update | Applying execute permission
[  OK  ] DietPi-Update | Successfully applied pre-patches

 DietPi-Update
-------------------------------------------------------------
 Phase: Upgrading APT packages

[ INFO ] DietPi-Update | APT update, please wait...
Get:1 https://deb.debian.org/debian bullseye InRelease [116 kB]
Get:2 https://archive.raspberrypi.org/debian bullseye InRelease [2
3.5 kB]
Get:3 https://deb.debian.org/debian bullseye-updates InRelease [39
```

:
Und nach wenigen Minuten
:

können wir durch Bestätigung mit ‚0: KEIN Nach-Hause-Telefonieren…'
(aber das mag jeder so entscheiden, wie er es mag)
mit der Installation es RailPi konkret anfangen.

Jetzt noch ein globales Passwort setzen:

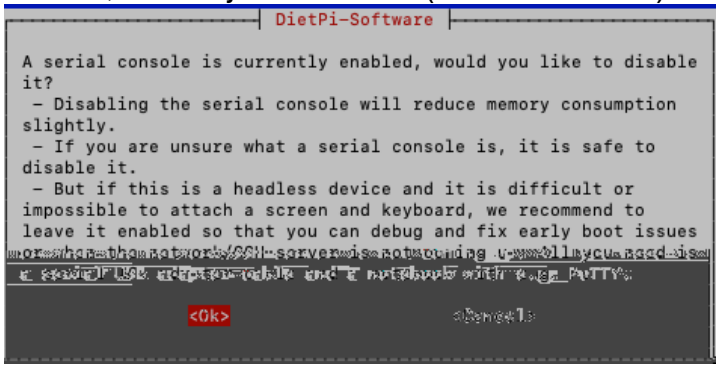
und die User-PWs für root und dietpi setzen:

Als nächstes kann die serielle Schnittstelle abgeschaltet werden.

Hinweis:
bei einem Headless-Device (so wie wir das vorhaben, kann die serielle Schnittstelle durchaus als Notfall-Zugriffsmöglichkeit dienen -> dann nicht abschalten!)

Ich für meinen Fall schalte die seriellen Schnittstellen ab. Was ich tatsächlich benötige, ist die LAN-Verbindung und die CAN-Verbindung über das CAN-HAT.

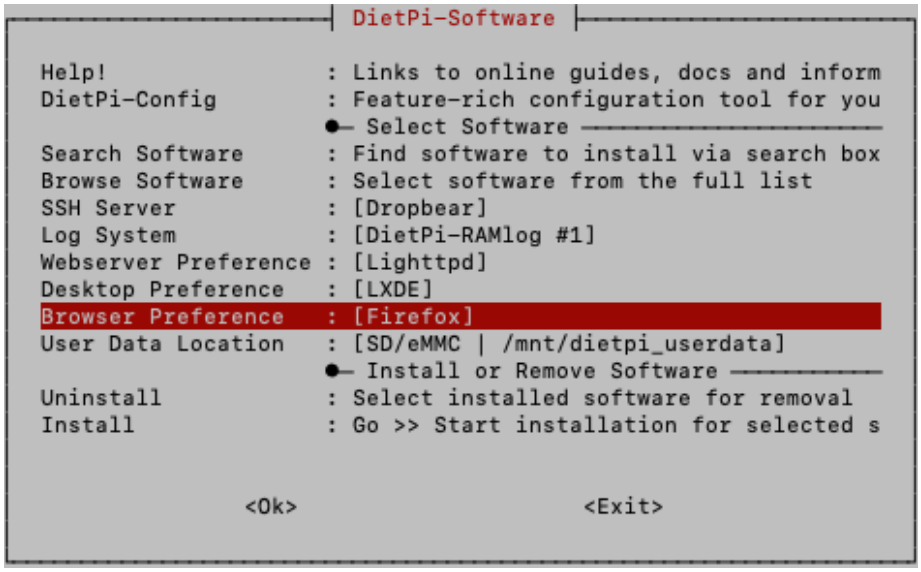
ACHTUNG:
Eine Modellbahnzentrale wie die CC-Schnitte hat dann allerdings schlechte Karten, da sie ja über USB (= serieller Port) verbunden wird!



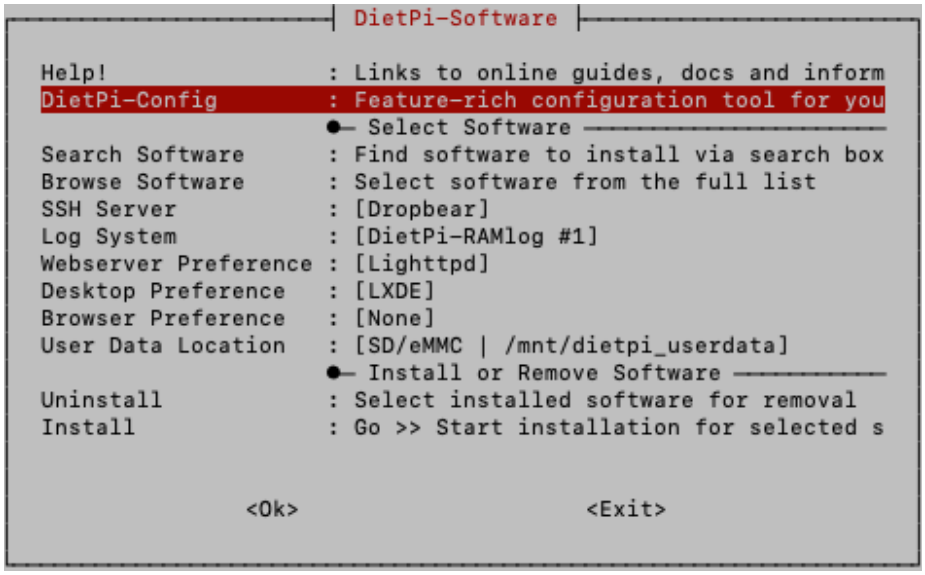
Das waren die Vorbereitungen, jetzt geht es mit der Konfiguration wirklich los. Und da man über den Befehl sudo dietpi-config jederzeit diese Konfiguration erneut aufrufen kann, beschränken wir uns auf das Wesentliche:

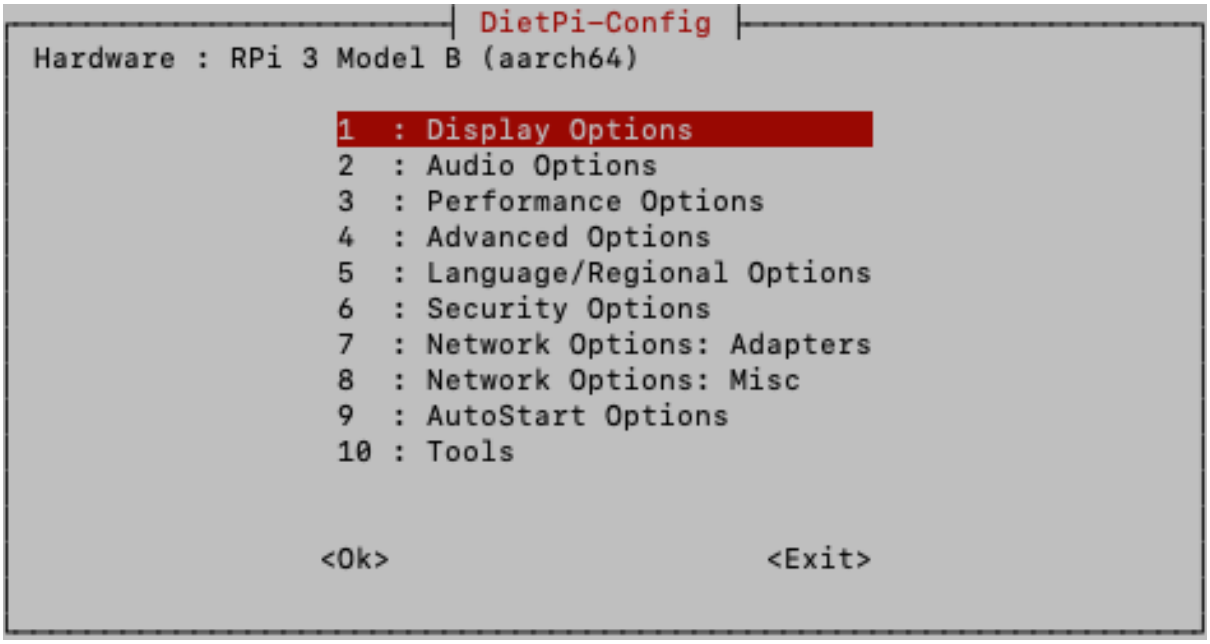
Wir benötigen keinen Desktop bzw. ein grafisches Benutzerinterface GUI. Dies können wir nur indirekt beeinflussen, indem wir kein darauf zurückgreifendes Programm installieren.

Und wir benötigen keinen Browser (RailPi u.a. als Webserver stellt ja lediglich Webseiten zur Bedienung der Modellbahn zur Verfügung und soll NICHT auch noch als die Benutzeroberfläche dazu dienen).



Also wird Firefox abgewählt und wir kümmern uns um die Konfiguration der Hardware:





1 - Display options:
- 1 - Display Resolution auf ‚Headless‘
- 7 - HDMI Boost auf 0
- 9 - RPi camera LED auf 0

3 - Performance options:
- Overclocking: Profil LOW
- ARM Temp Limit: 65°C (empfohlen)

4 - Advanced options:
- SPI state: ON. (wird für den CAN-HAT benötigt)
- (Tip: hier könnte die serielle Schnittstelle auch wieder aktiviert werden)

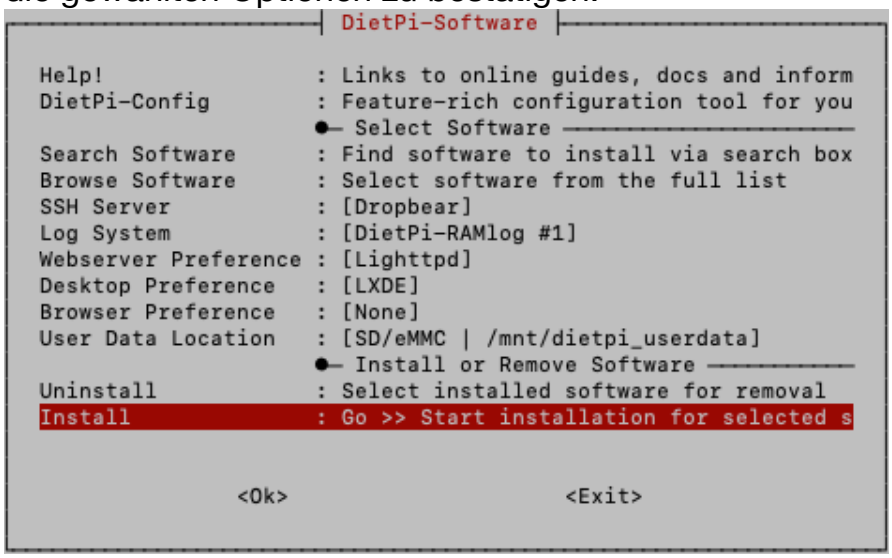
5 - Language/Regional options:
- Timezone: Europe - Berlin

6 - Security option:
- Hostname: RailPi

7 - Network options adapters:
- hier verändern wir nichts, wir nutzen ausschließlich LAN
- (Tip: wir könnten hier auch WIFI wieder einschalten - oder eine statische IP-Adresse setzen)

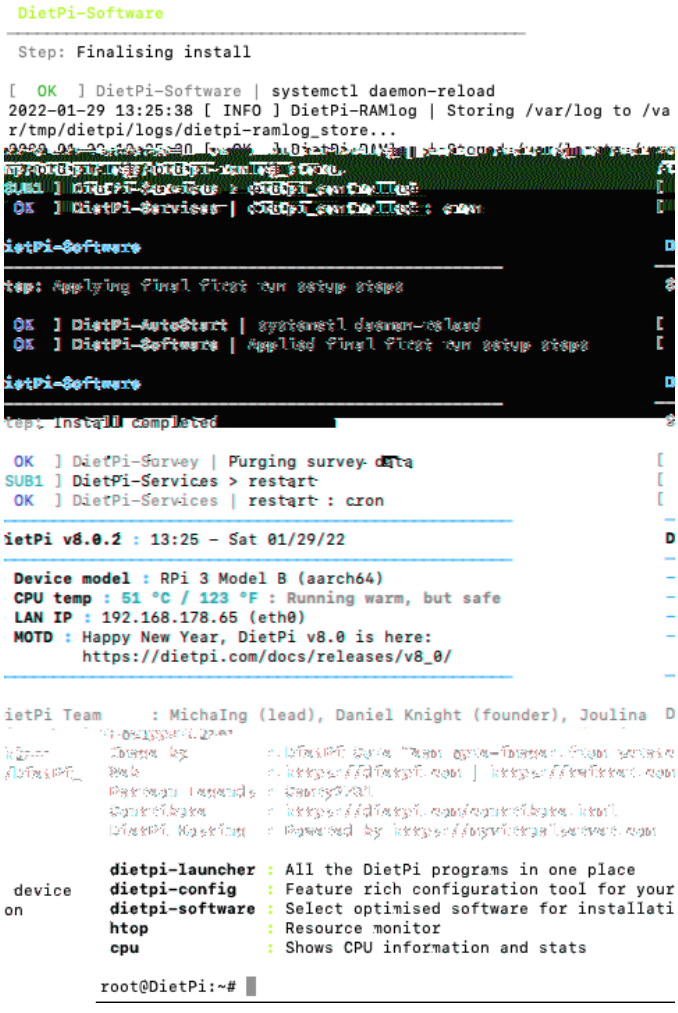
(bei den Menüpunkten, die nicht aufgeführt sind, wird erst einmal nichts geändert)

Die Konfiguration ist nun beendet und wir wählen den Menüpunkt **Install**, um die gewählten Optionen zu bestätigen:



Jetzt müssen wir nur noch bestätigen, daß wir mit einem minimalen Image weitermachen möchten. Fertig.

Die Basis-Installation des RailPi's ist abgeschlossen:



Wenn wir jetzt den RailPi neu starten und versuchen, uns wieder anzumelden, könnte es passieren, daß wir auf unserem PC erst einmal die Sicherheit für den SSH-Zugriff wiederherstellen müssen. Denn der SSH-Client auf unserem PC findet unter der IP Adresse jetzt 2 Rechner mit den Namen ‚dietpi' und ‚RailPi'.

Dazu müssen wir (auf einem Mac) in der Datei ‚known_hosts:1' den alten Eintrag für den Zugriff auf die IP 192.168.178.65 löschen (IP-Eintrag für dietpi) und können uns dann auf der gleichen Adresse mit dem neuen Namen RailPi verbinden:

```
detlef@MBP ~ % ssh root@RailPi
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: POSSIBLE DNS SPOOFING DETECTED!          @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
The ECDSA host key for railpi has changed,
and the key for the corresponding IP address 192.168.178.65
has a different value. This could either mean that
DNS SPOOFING is happening or the IP address for the host
and its host key have changed at the same time.
Offending key for IP in /Users/detlef/.ssh/known_hosts:6
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle a
ttack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:fZd4ytla875+8r9XPmjJgj6wsj469TiW0nbJdytfxpk.
Please contact your system administrator.
Add correct host key in /Users/detlef/.ssh/known_hosts to get rid of
 this message.
Offending ECDSA key in /Users/detlef/.ssh/known_hosts:1
ECDSA host key for railpi has changed and you have requested strict
checking.
Host key verification failed.
```

und kaum ist die Datei known_hosts:1 um die alte Anmeldung bereinigt, geht's auch weiter:

```
detlef@MBP ~ % ssh root@RailPi
The authenticity of host 'railpi (192.168.178.65)' can't be established.
ECDSA key fingerprint is SHA256:fZd4ytla875+8r9XPmjJgj6wsj469TiW0nbJdytfxpk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'railpi,192.168.178.65' (ECDSA) to the list of known
hosts.
root@railpi's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
 ------------------------------------------------------
 DietPi v8.0.2 : 13:55 - Sat 01/29/22
 ------------------------------------------------------
 - Device model : RPi 3 Model B (aarch64)
 - CPU temp : 44 °C / 111 °F : Optimal temperature
 - LAN IP : 192.168.178.65 (eth0)
 - MOTD : Happy New Year, DietPi v8.0 is here:
          https://dietpi.com/docs/releases/v8_0/
 ------------------------------------------------------

 DietPi Team     : MichaIng (lead), Daniel Knight (founder), Joulinar (support)
 Image by        : DietPi Core Team (pre-image: from scratch)
 Web             : https://dietpi.com | https://twitter.com/DietPi_
 Patreon Legends : Camry2731
 Contribute      : https://dietpi.com/contribute.html
 DietPi Hosting  : Powered by https://myvirtualserver.com

 dietpi-launcher : All the DietPi programs in one place
 dietpi-config   : Feature rich configuration tool for your device
 dietpi-software : Select optimised software for installation
 htop            : Resource monitor
 cpu             : Shows CPU information and stats
```

Die zuvor beschriebene Prozedur muß nicht, kann aber vorkommen.

Der RailPi ist jetzt bereits hinreichend eingerichtet, um Zentralen wie die Märklin CS2 oder CS3 oder auch die CANguru-Bridge (hierzu in einem weiteren Dokument später mehr) in Verbindung mit RailControl nutzen zu können.

Aber - wir wollen unseren RailPi ja auch DIREKT an die CAN-Schnittstelle hängen, um uns eine weitere Hardware-Zentrale sparen zu können.

Abschalten können wir später den RailPi mit dem Befehl ~~`sudo reboot -p`~~ – jetzt werden wir den RailPi einfach nur neu starten: `reboot`

Eine Routine um den RailPi herunterzufahren wird noch nachgereicht (der Diet-Pi kennt den reboot-p bzw. den shutdown-Befehl offensichtlich nicht)

Mit einer neuen Anmeldung via SSH ist jetzt der Zeitpunkt gekommen, um uns noch einige Tools zu installieren, die wir später für die Programme can2lan und RailControl benötigen.

Es handelt sich dabei um den c++ - Compiler mit notwendigen Tools und verschiedenen notwendigen Libraries:

```
root@RailPi:~#
apt-get install g++ binutils make git can-utils lighttpd zlib1g-dev libpcap-dev
```

```
[detlef@MBP ~ %                                                                    ]
[detlef@MBP ~ % ssh root@RailPi                                                    ]
[root@railpi's password:                                                           ]
[                                                                                  ]
[The programs included with the Debian GNU/Linux system are free software;         ]
 the exact distribution terms for each program are described in the
 individual files in /usr/share/doc/*/copyright.

 Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
 permitted by applicable law.
  ──────────────────────────────────────────────────────
  DietPi v8.0.2 : 17:28 - Sat 01/29/22
  ──────────────────────────────────────────────────────
  - Device model : RPi 3 Model B (aarch64)
  - CPU temp : 42 °C / 107 °F : Optimal temperature
  - LAN IP : 192.168.178.65 (eth0)
  - MOTD : Happy New Year, DietPi v8.0 is here:
           https://dietpi.com/docs/releases/v8_0/
  ──────────────────────────────────────────────────────

  DietPi Team     : MichaIng (lead), Daniel Knight (founder), Joulinar (support)
  Image by        : DietPi Core Team (pre-image: from scratch)
  Web             : https://dietpi.com | https://twitter.com/DietPi_
  Patreon Legends : Camry2731
  Contribute      : https://dietpi.com/contribute.html
  DietPi Hosting  : Powered by https://myvirtualserver.com

  dietpi-launcher : All the DietPi programs in one place
  dietpi-config   : Feature rich configuration tool for your device
  dietpi-software : Select optimised software for installation
  htop            : Resource monitor
  cpu             : Shows CPU information and stats

 root@RailPi:~#
 root@RailPi:~#
[root@RailPi:~# apt-get install g++ binutils make git can-utils lighttpd zlib1g-dev libpcap-de  ]
```

Damit ist die Installation der zusätzlichen Tools abgeschlossen und wir können uns ganz auf den RailPi konzentrieren

Einbindung der CAN-Schnittstelle

Konfiguration des SPI-Geräts ‚CAN-Hat'

Hierzu müssen wir nun die Datei config.txt im Verzeichnis /boot mit Hilfe des im System integrierten Editors **nano** ändern:

hinter der Zeile
dtparam=spi=on
muß die folgende Zeile hinzugefügt werden:
dtoverlay=mcp2515-can0,oscillator=16000000,interrupt=23

```
root@RailPi:~# nano /boot/config.txt
```

Datei speichern mit <ctrl>+x - dann y für speichern und <Enter> um den Dateinamen zu bestätigen.

Hinweis:
beim verwendeten CAN-Hat (s.S. 4 unten) wird der Interrupt 23 genutzt - für andere RPI-CAN Interfaces ist es zumeist der Interrupt 25.

Starten der CAN-Schnittstelle

zunächst einmal manuell:

```
root@RailPi:~# ip link set can0 up type can bitrate 250000 restart-ms 100
Cannot find device "can0"
```

die Antwort ‚cannot find device „can0“‘ zeigt einen Fehler an. Entweder ist die Gleisbox noch nicht mit dem CAN-HAT verbunden oder ein anderer Fehler (CAN-Verdrahtung?) liegt vor.

In diesem Fall ist der Fehler zu beheben, der RailPi neu zu starten und dann die can0 Schnittstelle ebenfalls erneut zu starten:

```
root@RailPi:~# ip link set can0 up type can bitrate 250000 restart-ms 100
root@RailPi:~#
```

—> Keine Fehlermeldung - alles i.O.

Jetzt kann für die automatische Konfiguration bei Start des RailPi mit dem **nano-Editor** die Datei **can0** im Verzeichnis **/etc/network/interfaces.d** angelegt werden:

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

Anmerkung
hinter der letzten Zeile muß noch ein Zeilenumbruch vorhanden sein (nach down)

So schaut's dann aus:

```
root@RailPi:~# nano /etc/network/interfaces.d/can0
```

Datei speichern mit <ctrl>+x - dann y für speichern und <Enter> um den Dateinamen zu bestätigen.

Um nun auch ‚echten traffic' auf dem CAN-Bus zu erzeugen bzw. zu lesen, bereiten wir das System für das Programm ‚can2lan' vor, indem wir eine weitere Datei ‚geraet.vrs' mit folgendem Inhalt anlegen:

[geraet]
version
.minor=1 geraet
.sernum=1
.hardvers=RPi,3

Anmerkung
hinter der letzten Zeile muß noch ein Zeilenumbruch vorhanden sein (nach RPi,3)

```
root@RailPi:~# nano /var/www/html/geraet.vrs
```

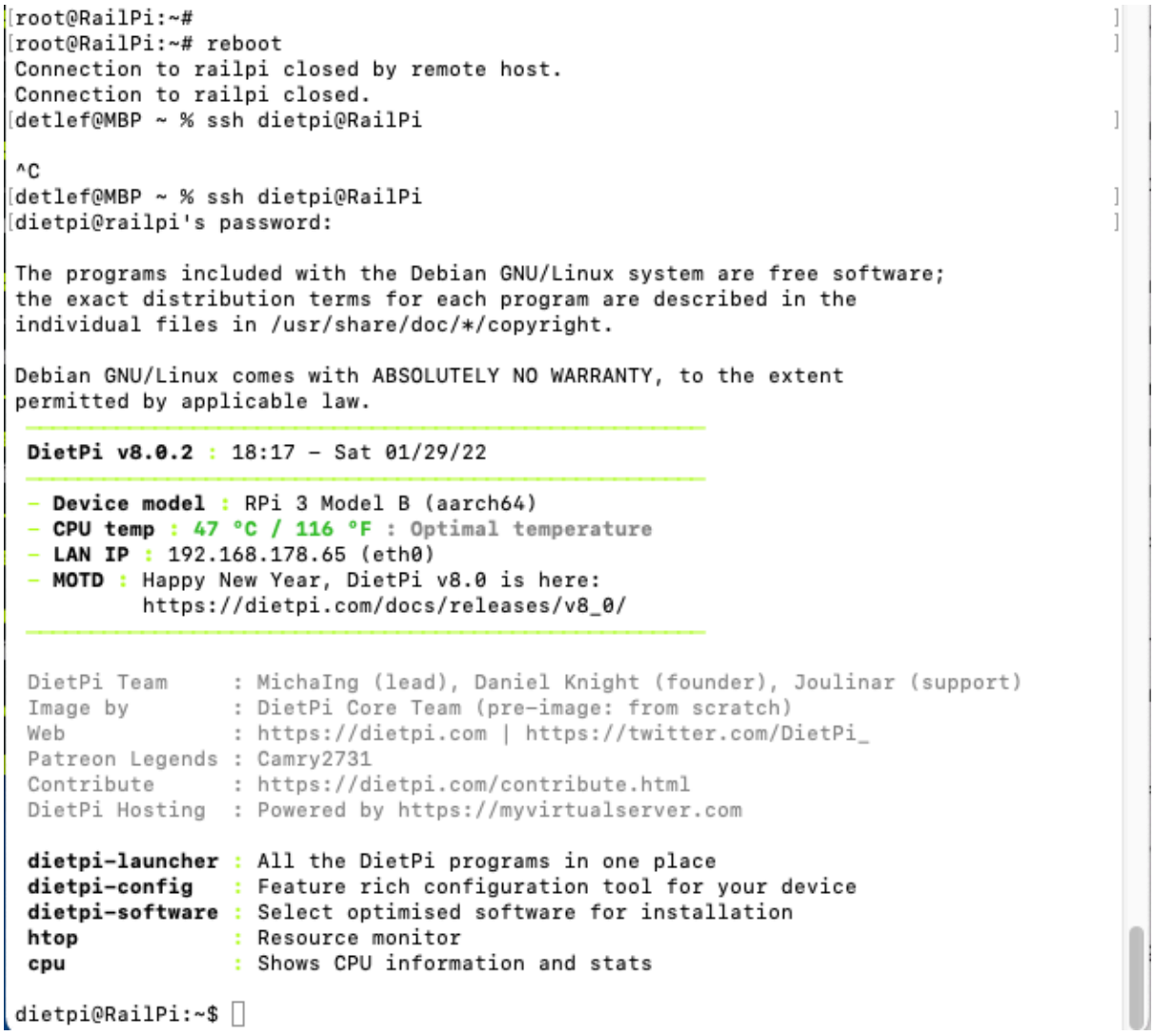
Datei speichern mit <ctrl>+x - dann y für speichern und <Enter> um den Dateinamen zu bestätigen.

Installation von can2lan

can2lan wurde von Gerd Bertelsmann entwickelt, um Daten zwischen dem CAN-Bus und der Ethernet-Schnittstelle auszutauschen. Dabei wird dann - ganz praktisch - eine Märklin CentralStation CS2 emuliert - was es dann später dem Programm RailControl ermöglicht, sich mit einer CS2 zu verbinden…

Aber zurück zur Installation von can2lan:

wir führen einen Neustart durch und loggen uns als user ‚**dietpi**‘ ein (ab jetzt immer).



Dann holen wir uns aus Gerds GIT-repository den Source-Code für can2lan, gehen ins Source-Verzeichnis und übersetzen und binden das Programm:

```
dietpi@RailPi:~$ git clone https://github.com/GBert/railroad.git
Cloning into 'railroad'...
:
dietpi@RailPi:~$ cd railroad/can2udp/src
dietpi@RailPi:~/railroad/can2udp/src$ make
:
```

Nun müssen wir noch die erzeugten Programme **can2lan** und **can-monitor** in verschiedene Verzeichnissen verteilen:

```
dietpi@RailPi:~/railroad/can2udp/src$
sudo cp ../files/maerklin/config/gleisbild.cs2 /var/www/html/config/gleisbild.cs2

dietpi@RailPi:~/railroad/can2udp/src$ sudo cp can2lan /usr/sbin/
dietpi@RailPi:~/railroad/can2udp/src$ sudo cp can-monitor /usr/bin/
```

Danach wird can2lan als Dienst im System verankert, so daß bei jedem Neustart der Dienst can2lan automatisch läuft:

```
dietpi@RailPi:~/railroad/can2udp/src$ sudo /etc/init.d/can2lan.init start
dietpi@RailPi:~/railroad/can2udp/src$ sudo /etc/init.d/can2lan.init status
dietpi@RailPi:~/railroad/can2udp/src$ sudo /etc/init.d/can2lan.init stop
dietpi@RailPi:~/railroad/can2udp/src$ sudo /etc/init.d/can2lan.init defaults
```

Der Übersicht halber wurden hier die Ausgaben des Systems nicht angezeigt, als screen-copy sieht das wie folgt aus:

Wir haben zwar kein Windows-System vor uns und könnten uns auch diesen Neustart sparen, aber: um immer einen reproduzierbaren Stand zu haben, leisten wir uns diesen Neustart ganz einfach:

```
sudo reboot
```

Installation von RailControl

Wir starten den RailPi **neu** und melden uns wieder als **user dietpi** ein.

Dann holen wir uns von GitHub den RailControl-Sourcecode von Teddy:

```
dietpi@RailPi:~$ git clone https://github.com/teddych/railcontrol.git
:
```

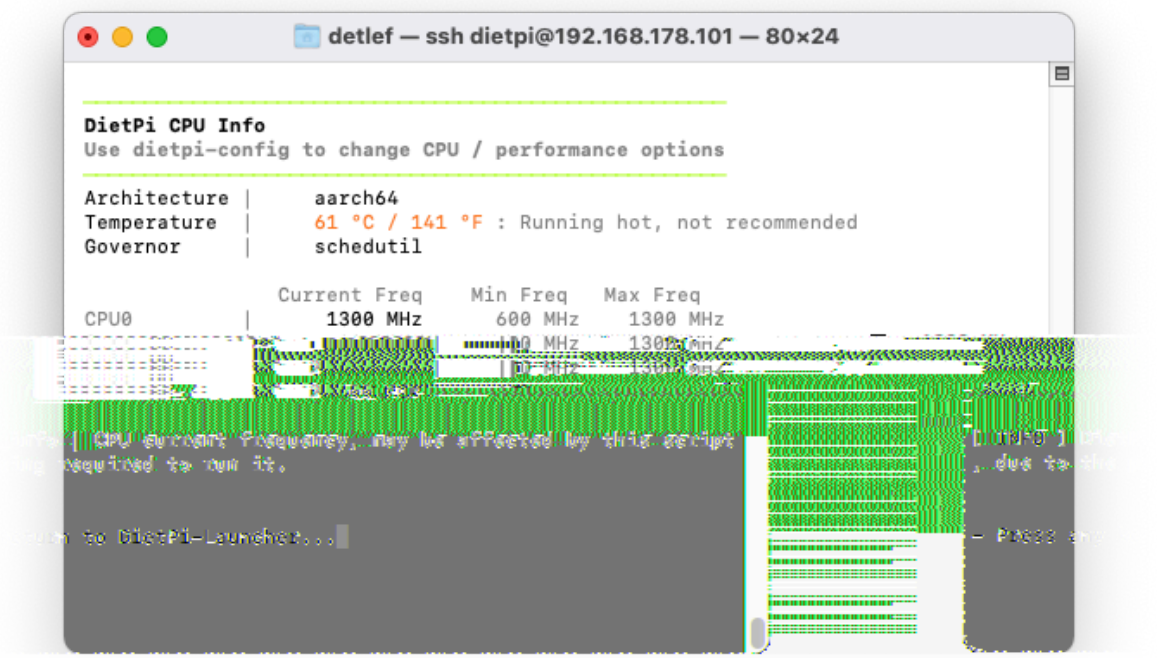
wechseln in das neu angelegte Verzeichnis

```
dietpi@RailPi:~$ cd railcontrol
```

und starten die Übersetzung des Programms:

```
dietpi@RailPi:~/railcontrol$ make
:
```

Während der Übersetzung von RailControl wurde sporadisch die CPU-Info abgerufen:



Dies zeigt, daß die 4-Kerne der CPU mit max. Taktfrequenz (übertaktet) arbeiten, aber dennoch unterhalb der empfohlenen Grenztemperatur 65°C bleiben (die Chips auf dem Raspberry sind mit den handelsüblichen Standard-Kühlkörpern versehen, passive Kühlung)

Der Übersetzungsvorgang benötigte ca. 34 min

und es sollte nach der erfolgreichen Übersetzung einem Aufruf von RailControl nichts mehr im Wege stehen:

```
dietpi@RailPi:~/railcontrol$ ./railcontrol
```

In RailControl muß jetzt noch die Zentrale eingebunden werden. In unserem Fall mit Namen ‚test' die CS2 mit UDP-Protokoll (ggf. Auch mit TCP-Protokoll)

```
dietpi@RailPi:~/railcontrol$ ./railcontrol
2022-01-29 20:39:27.565363: Info: Main: Starting RailControl
2022-01-29 20:39:27.565825: Info: Main: Version: 21
2022-01-29 20:39:27.565930: Info: Main: Compile date: 2022-01-29 19:11:49
2022-01-29 20:39:27.565973: Info: Main: Last GIT commit hash: 5dbeaab6e857fe432e68a1d66a8afcb17abbcdcb
2022-01-29 20:39:27.566030: Info: Main: Last GIT commit date: 2021-10-26 20:46:15
2022-01-29 20:39:27.566093: Info: Config: Reading config file railcontrol.conf
2022-01-29 20:39:27.566409: Info: Config: Parameter found in config file: dbfilename = railcontrol.sqlite
2022-01-29 20:39:27.566500: Info: Config: Parameter found in config file: dbkeepbackups = 10
2022-01-29 20:39:27.566572: Info: Config: Parameter found in config file: webserveraddress = any
2022-01-29 20:39:27.566647: Info: Config: Parameter found in config file: webserverport = 8082
2022-01-29 20:39:27.567129: Info: SQLite: Opening SQLite database with filename railcontrol.sqlite
2022-01-29 20:39:27.569238: Info: SQLite: relations is up to date
2022-01-29 20:39:27.570463: Info: WebServer: Webserver started
2022-01-29 20:39:27.571214: 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://192.168.178.65:8082/
   http://[::1]:8082/
   http://[2a02:908:2f32:2c60:ba27:ebff:fea9:3724]:8082/
   http://[fe80::ba27:ebff:fea9:3724]:8082/
2022-01-29 20:39:27.577446: Info: New: My UID: 5ff8cf41; My CAN hash: 8739
2022-01-29 20:39:27.577631: Info: New: Starting Maerklin Central Station 2 (CS2) UDP / New at IP 192.168.178.65
2022-01-29 20:39:27.577679: Info: New: Sender socket created
2022-01-29 20:39:27.578131: Info: New: Receiver thread started
2022-01-29 20:39:27.578148: Info: Manager: Loaded control 10: New
2022-01-29 20:39:27.578708: Info: Manager: Loaded layer 1: Layer 1
2022-01-29 20:39:27.579806: Info: Manager: Loaded switch 1: New
2022-01-29 20:39:27.580469: Info: Manager: Loaded locomotive 1: test
2022-01-29 20:39:27.580931: Info: Manager: Debounce thread started
2022-01-29 20:39:28.581009: Info: New: Setting locomotive 4/6 to speed 0
2022-01-29 20:39:28.606429: Info: New: Setting locomotive 4/6 to direction of travel left
```

Dieses Dokument endet hier - weiter geht es später mit einem anderen Dokument über die konkrete Nutzung von RailControl (in Vorbereitung)

ACHTUNG!

Alle in diesem Dokument beschriebenen Konfigurationsvorschläge basieren auf persönlichen Erfahrungen.

Für Folgen aus der Nutzung der in diesem Dokument enthaltenen Informationen kann keine Haftung übernommen werden.