

# hzrr200

## Dynamische Rücklauftemperaturbegrenzung für Einrohrheizungen.

Version 200  
Aufbauanleitung

### Kurzbeschreibung

TODO

### Änderungen:

Datum	Version	Autor	Beschreibung	Bemerkung
1.7.2020	1.00	P. Loster	erste Zusammenfassung bestehender Beschreibungen, Sensor und Motor Anschluss	
20.7.2020	1.01	P. Loster	Sensor und Motor Test	
23.7.2020	1.02	P. Loster E. Loster	Inbetriebnahme hzrr200 Module	
28.7.2020	1.03	P. Loster E. Loster	Fehler verbessert; Anleitung für Dummies erweitert	
29.7.2020	1.04	E. Loster	Korrekturen, Ergänzungen	
25.8.2020	1.05	E. Loster	Korrekturen	
26.9.2020	1.06	P. Loster	Anfang Kapitel Raspberry Pi einrichten; Bitte and AK seine Installation zu beschreiben	
27.9.2020	1.07	Andi	Ergänzung Raspberry Pi Installationen	
28.9.2020	1.08	P. Loster	Ergänzung Steckerbelegung, RPi4 USB Anschluss	

# Table of Contents

Übersicht.....	2
TODO: Blockschaltbild typ. Gesamtsystem.....	2
TODO: Kurzbeschreibung der Komponenten.....	2
Reglermodul („Modul“).....	3
Komponenten.....	3
Blockschaltbild.....	4
Anschlüsse.....	5
Adresseinstellung.....	5
Crimpen von Modular Steckverbindern.....	5
Temperatursensoren DS18B20.....	6
Aufbau und Anschluss.....	6
Eigenschaften.....	7
Stellmotoren zur Ventilbetätigung.....	7
Anschluss Motor.....	7
Testgerät für Sensoren und Motoren.....	8
Aufbau und Inbetriebnahme.....	9
Software zur Inbetriebnahme aufspielen.....	9
Modul aufbauen.....	11
Test Aufbau.....	14
Der RS485 - USB Adapter.....	14
Serielles Terminal am PC oder RPi.....	14
Inbetriebnahme Tests.....	15
LCD-Test.....	15
Taster-Test (Button-Test).....	16
Address- und Temperatursensor-Test.....	16
Motor-Test.....	17
Kommunikationstest.....	18
Abschluss Inbetriebnahme.....	19
Anwendungsprogramm Installieren und Prüfen.....	19
Einrichten der Zentrale.....	19
Herstellen der SD-Karte.....	20
Einrichten des USB-Sticks.....	31
Stromanschluss und Monitor Anschluss.....	35
Einstecken des USB-Sticks.....	35
Ablauf beim Starten.....	37

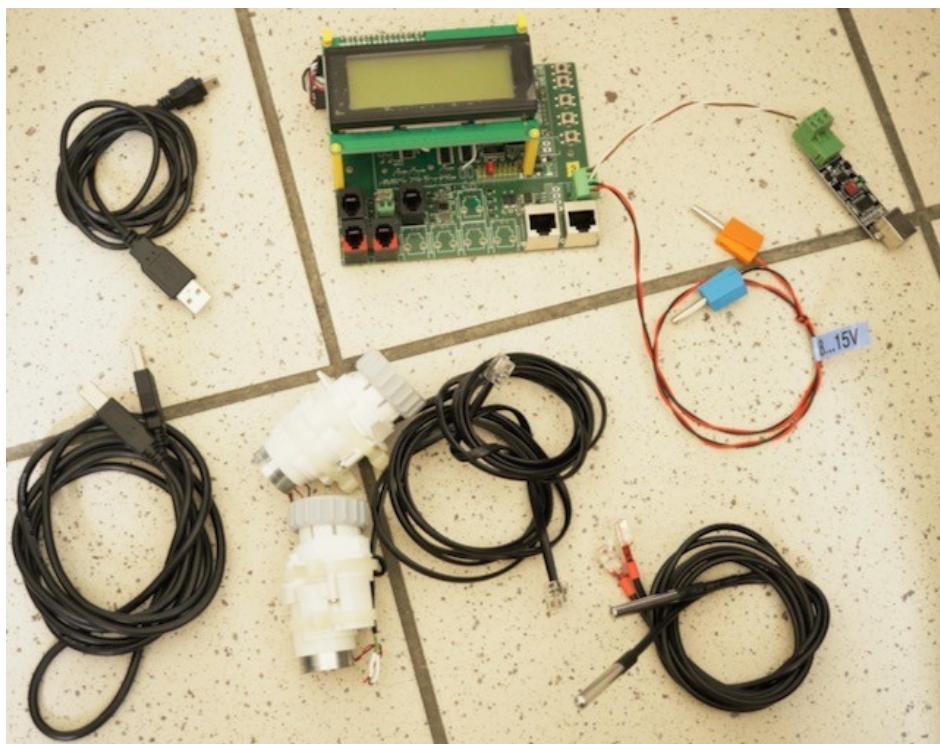
## 1 Übersicht

### 1.1 TODO: Blockschaltbild typ. Gesamtsystem

### 1.2 TODO: Kurzbeschreibung der Komponenten

## 2 Reglermodul („Modul“)

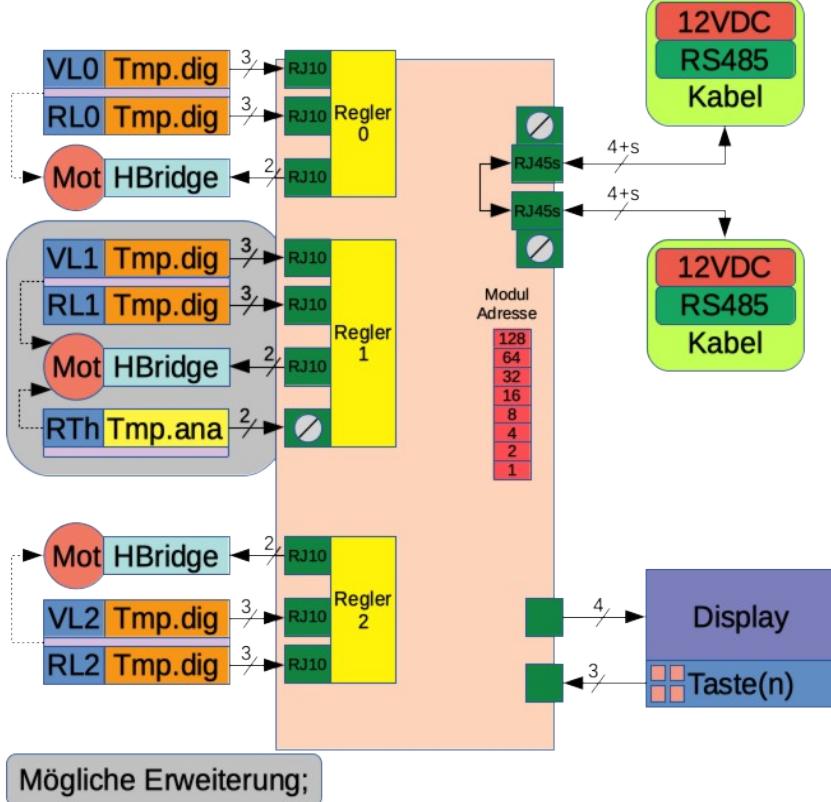
### 2.1.1 Komponenten



*Bild 1: Reglermodul oben Mitte (Prototyp); USB-Kabel oben links zum Programmieren; RS485-USB Adapter und 12V Anschluss oben rechts; USB-A-B Kabel f.RS485 Adapter unten links; zwei Stellmotoren unten Mitte und zwei Temperatursensoren unten rechts*

## 2.1.2 Blockschaltbild

### HZRR-200 Steuermodul



#### Arbeitspakete:

- Hardware:
    - +Schaltung entwerfen
    - +PCB 1 entwerfen, fertigen, testen
    - +Korrektur
    - PCB 2 fertigen, testen
    - Gehäuse Einbau, Muster
    - Stückliste, Lieferanten, Doku
  - Software programmieren:
    - +Treiber f. alle Anschlüsse, C++ Klasse(n)
    - Rücklauf (RL) Regelung
    - Raumthermostat (RT) Regelung
    - Anzeige definieren (LCD, 4x20)
    - Display + Taste progr.; Menü
    - Status + Kommandos definieren
    - RS485 Dialog definieren und progr.
    - Speicherung
    - Timing festlegen
    - Prozesse und Ablauf festlegen
    - Programmteile zusammenführen
    - Watchdog progr. und Test
    - Tests, Korrekturen
  - Dokumentation
    - i Schalt- und Bestückungsplan
    - i Stückliste (BOM)
    - i Aufbauanleitung
    - i Software Struktogramm, Abl.Diagr.
    - i Quellcode dokumentieren
    - e Installationsanleitung
    - e Anschlussbeschreibung
- => Definition und Programmierung der Zentrale

HZRR-200\_Block04.odg  
© Peter Loster, 2020-03-21

Bild 2: Blockschaltbild Reglermodul

Ein Mikrocontroller im Reglermodul (kurz: Modul) enthält bis zu drei Regler Nr. 0,1 und 2. Regler 1 kann auch als Heizkörperthermostat arbeiten. An das Modul sind eine LCD-Anzeige und Tasten angeschlossen. Die Moduladresse wird über Steckbrücken eingestellt. Die Versorgungsspannung von 12V sowie die Datenbusleitung für eine RS485 Schnittstelle werden über RJ45 Modular Stecker und geschirmte min. CAT5E Leitungen hergestellt.

1. Zwei Temperatursensoren, 'Onewire' DS18B20 über RJ10 (4P4) Modular Stecker für Vorlauf (VL) und Rücklauf (RL) Temperatur für jeden Regler Nr. 0, 1, 2 . Anschluss über drei Adern.
  2. Ein 3V Stellmotor für jeden Regler zum Verstellen des Ventils; zwei Adern.
  3. 4 Zeilen zu je 20 Zeichen LCD-Anzeige
  4. 5 Tasten und eine Reset-Taste
  5. 2 RJ45 Stecker zur Stromversorgung mit 12V (2x +12V, 3xGND) und für den RS-485 Bus (2 Pole) sowie die Abschirmung der Leitung. Alternativ können diese Leitungen auch über Steckklemmen angeschlossen werden, von denen eine bestückt ist.
- ACHTUNG:** Es können maximal 30 Module in einem RS485 Netz betrieben werden.
6. **Alternativ kann Regler 1** als Heizkörper Regler mit einem Raumsensor „RAS PT“ der Fa. „Technische Alternative“ betrieben werden. Die Onewire Temperatursensoren entfallen dann.

## 2.1.3 Anschlüsse

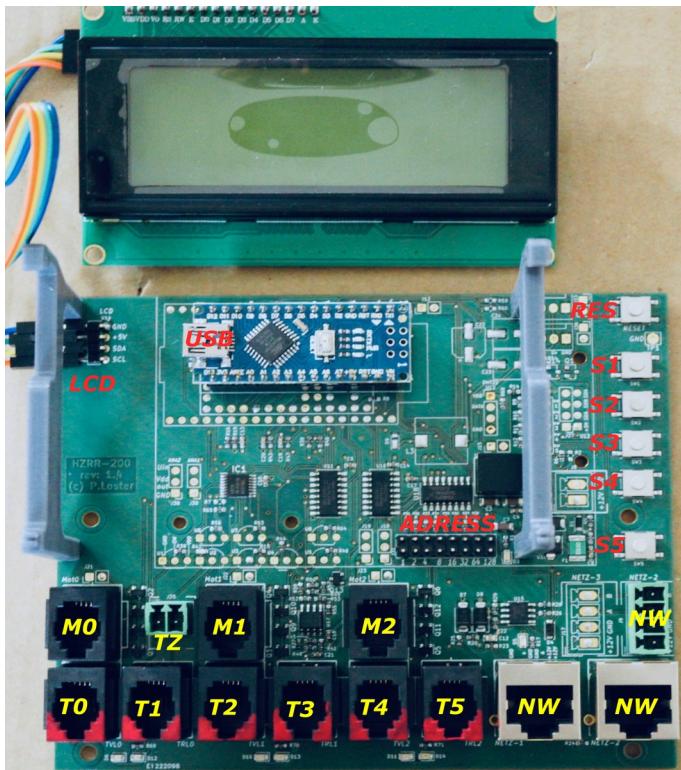


Bild 1: Anschlüsse, Schalter, Adresse

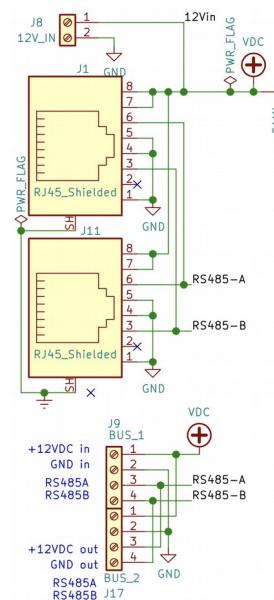


Bild 2:  
Durchverbundene  
Pins 4-Pol  
Steckklemmen und  
RJ45 Stecker

**Achtung: die RJ45 Steckverbinder haben keine Ethernet-Pegel; KANN BESCHÄDIGT WERDEN WENN AN ETHERNET ANGESCHLOSSEN WIRD**

## 2.1.4 Adresseinstellung

Die Steckbrücken ('Jumper') sind mit 1,2,4,8,16,32,64 und 128 bezeichnet. Die linken 5 Brücken dienen der Adresswahl.

Dabei addiert man die Zahl unter jeder gesteckten Brücke und erhält eine Adresse zwischen 0 (keine Brücke) und 31 (alle Brücken gesteckt).

Die Brücke 32 und 64 schalten den Regler Nr. 2 und 3 aktiv; die Brücke 128 schaltet den Regler Nr. 2 in den Raumregler Modus.

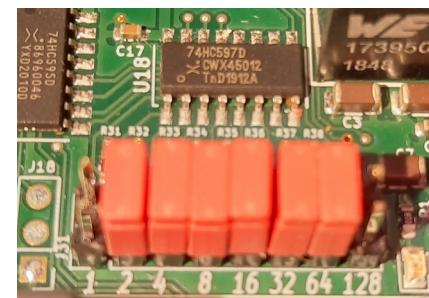


Bild 3: Steckbrücken für  
Adresse und Optionen

## 2.2 Crimpen von Modular Steckverbindern



## 2.3 Temperatursensoren DS18B20



### 2.3.1 Aufbau und



Bild 7: Sensor fertig; rot markiert

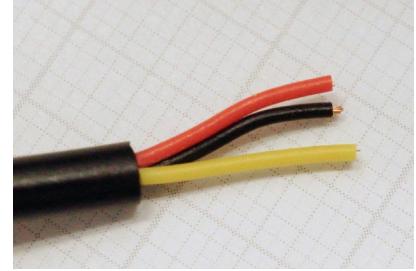
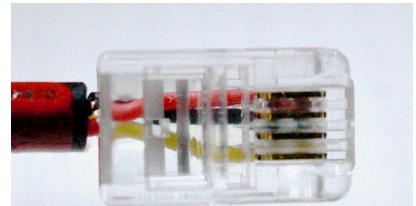


Bild 6: Vorbereitung der Anschluss



- Kabel wie in Bild 6 ca. 12mm abisolieren und Kabel nach Farbe etwas aufspreitzen
  - In den RJ10 Stecker wie in Bild 11 Kabel einführen; Pin 3 bleibt frei; dann mit Spezialzange für Modular 4P4 Stecker crimpfen
- ACHTUNG: Verpolen der Anschlüsse zerstört den Sensor !!!**
- Sensor an Testschaltung anschließen und Temperatur prüfen.
  - Stecker oder auch Kabel in Steckernähe mit rotem Lack markieren, wenn Test erfolgreich war, siehe auch Bilder;
- dies ist nötig um die Stecker von den Motorsteckern zu unterscheiden!**

### 2.3.2 Eigenschaften

Siehe Datenblatt DS18B20. Auf wenige zehntel °C genaue Sensoren mit digitaler Messwertübermittlung. Die Leitungslänge verfälscht daher die Messwerte nicht. Um eine Eigenerwärmung der Sensoren durch den Betrieb gering zu halten (vor allem bei Messungen in Luft) werden die Sensoren nur während der Messung kurzfristig mit Strom versorgt.

## 2.4 Stellmotoren zur Ventilbetätigung

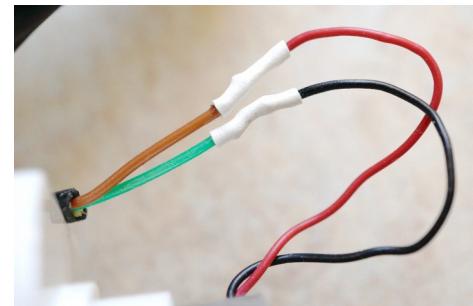
Die Stellmotoren stammen aus elektronischen Heizungsthermostaten mit 2x1,5V Versorgungsspannung. 2 Adern eines 4-poligen Spezialkabels für Modular Stecker (RJ10) 4P4 werden an den Motor angelötet und auf der anderen Seite an den Stecker gecrimpt.

### 2.4.1 Anschluss Motor

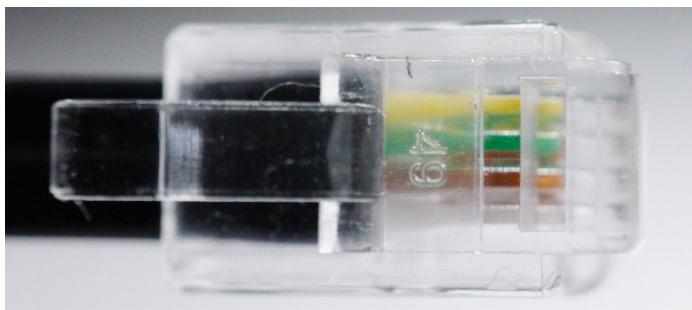
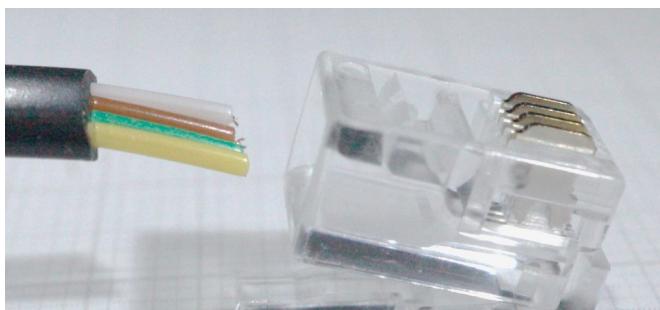
*Bild 12:*



Stellmotor für 'Heimann' Ventil



*Bild 13: Anschluss rot-braun und grün-schwarz an Motor*



*Bild 16: fertig gecrimpter Stecker*



**WICHTIG: Auf richtige Polung achten !!! Andernfalls laufen die Ventile falsch rum**

## 2.5 Testgerät für Sensoren und Motoren

Ein Reglermodul wurde zum Test der Funktion von Temperatursensoren und Motoren nach deren Montage mit einer speziellen Test-Software versehen.

Bild 18:



*Stromversorgung über 5V USB Kabel am Mikrocontroller des Sensormoduls; Temperatursensoren können unten links an den Modular Steckern (4P4) rot gekennzeichnet angeschlossen werden, darüber über 4P4 Stecker schwarz die Motoranschlüsse und dazwischen der Anschluss für den Raumtemperatur Fühler.*

Motor&Sensor test  
+ FW:0.14.2020-07-19  
HW:0.1+ 2020-03-31

Sensortest01 5=Ende  
U=25.1 R=25.5 Z=24.3  
1/2=M0 3/4=M1 Auf/Zu

Sensortest01 5=Ende  
U=25.2 R=25.2 Z=24.4  
1/2=M0 3/4=M1 Auf/Zu

Sensortest01 5=Ende  
U=25.2 R=25.0 Z=24.9  
1/2=M0 3/4=M1 Auf/Zu

Bild

Sensortest01 5=Ende  
U=25.1 R=24.6 Z=24.4  
1/2=M0 3/4=M1 Auf/Zu  
! MOTOR KURZSCHLUSS!

INITIATING  
RELOAD  
OF

23: Bei einer Inbetriebnahme kann ein Kurzschluss vorkommen. Der Motor wird sofort ausgeschaltet und für 2sec dieser Zustand angezeigt: Taste sofort auslassen, Motor abstecken und Fehler suchen!!!

## 2.6 Aufbau und Inbetriebnahme

### 2.6.1 Software zur Inbetriebnahme aufspielen

Die Software befindet sich in obigem Verzeichnis (iMac3 von pl), kann aber auf einem beliebigen Computer mit Linux/OSX/Win mit installierter Arduino IDE (Vers >= 1.8) mit den nötigen Zusatzmodulen geladen und auf dem Arduino Nano installiert werden.

[pl/Documents/\\_/projekte/hzrr200/HZRR\\_200/Software/arduino/hzrr-200/hzrr200\\_10\\_Inbetrieb/hzrr200\\_10\\_Inbetrieb.ino](http://pl/Documents/_/projekte/hzrr200/HZRR_200/Software/arduino/hzrr-200/hzrr200_10_Inbetrieb/hzrr200_10_Inbetrieb.ino)

**ACHTUNG:** Das Programm besteht aus mehreren Dateien. Sie alle werden in der Leiste über dem Text-Editor angezeigt. DIESE DATEIEN NICHT VERÄNDERN sonst funktioniert das Test-Programm nicht mehr und muss wieder repariert oder von einer Sicherung geholt werden.

**WICHTIG:** Zum Hochladen des Programms ist es unwichtig, welche der Dateien angewählt wurde, Drücken des „Hochladen“ Pfeils lädt immer das gesamte Programm.

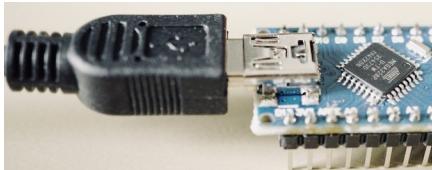
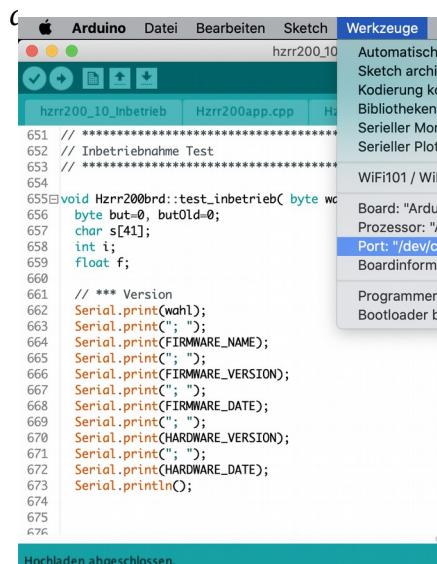


Bild 25: Arduino Nano an USB-Buchse angeschlossen



an der Arduino IDE

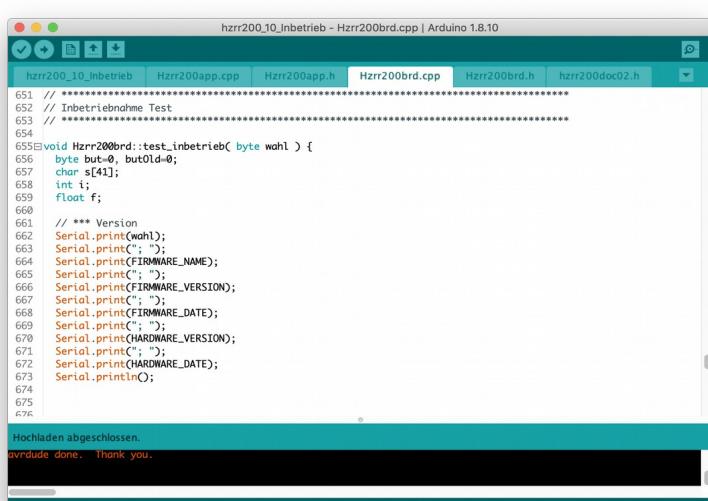


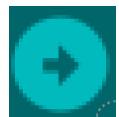
Bild 26: Arduino IDE mit geladenem Programm zur Inbetriebnahme ('Sketch')



}

Unter Werkzeuge wird eingestellt:

- Board: Arduino Nano
- Prozessor: ATmega328P (Old Bootloader)
- Port: Häckchen neben der passenden seriellen USB Schnittstelle.  
Hinweis: der lässt sich finden indem man den Arduino absteckt, in obiges Menü geht und sich die Port-Liste merkt. Dann wieder anstecken und nach etwa 10 sec die Portliste neu öffnen und anzeigen lassen. An den neuen Port ein Häckchen setzen.
- Ganz unten rechts wird zur schnellen Orientierung eine Kurzform der Einstellungen angezeigt,  
Oben in der Fensterüberschrift wird der Name des Programms und die Version der Arduino IDE angezeigt.
- Drücken der runden Schaltfläche oben links mit dem Pfeil nach rechts startet das Programmieren des Arduino Nano. Dabei wird das Programm ggf. vorher neu übersetzt.
- Nach erfolgreichem Hochladen wird unten in der Zeile über dem schwarz hinterlegten Fenster der Text „Hochladen abgeschlossen“ angezeigt.
- Die Programmierung kann auch im eingebauten Zustand bei fertigem Modul erfolgen.



## 2.6.2 Modul aufbauen

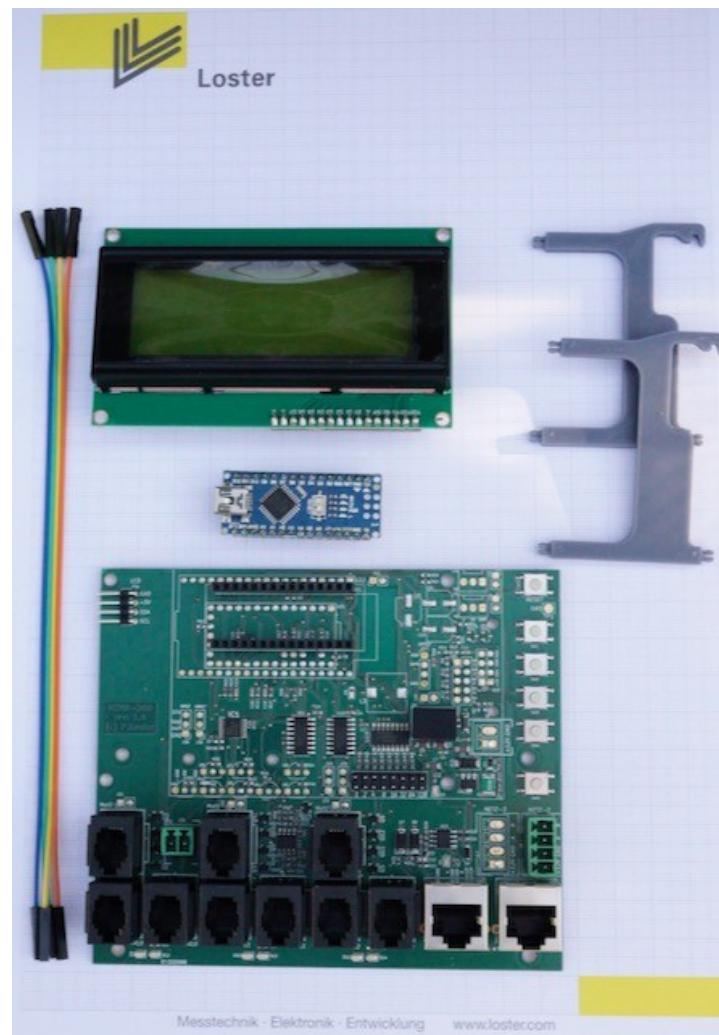
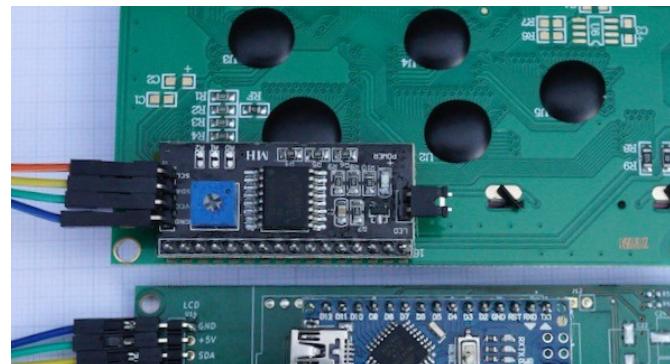
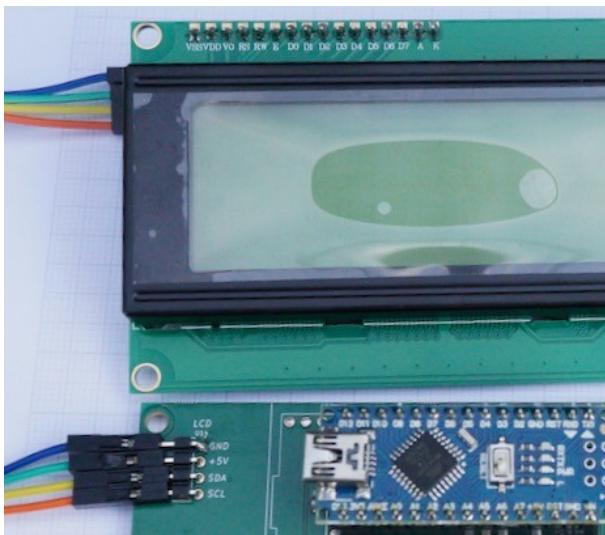
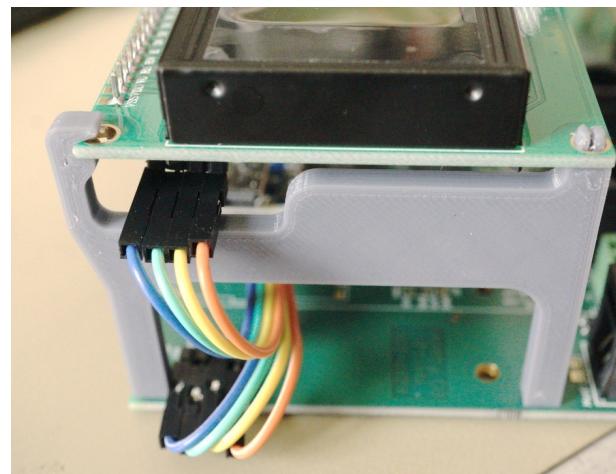
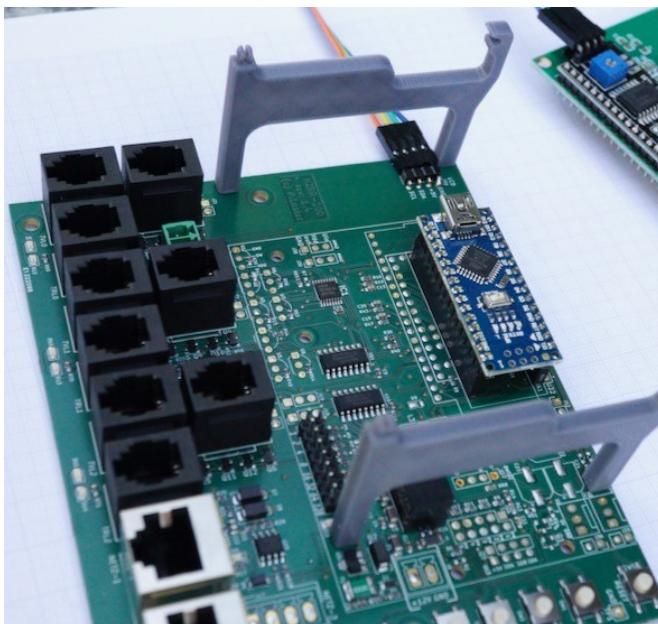


Bild 28: Komponenten zum Aufbau des Moduls

- Oben links: LCD mit montierter I2C Platine auf der Rückseite
- Oben rechts: Zwei Halter für LCD-Anzeige
- Mitte: Arduino Nano, vorzugsweise mit Inbetriebnahme Programm installiert
- Unten: Modul Platine
- linker Rand: 4-poliges Buchse-Buchse Kabel zum Anschluss der LCD-Anzeige an die Modul Platine.



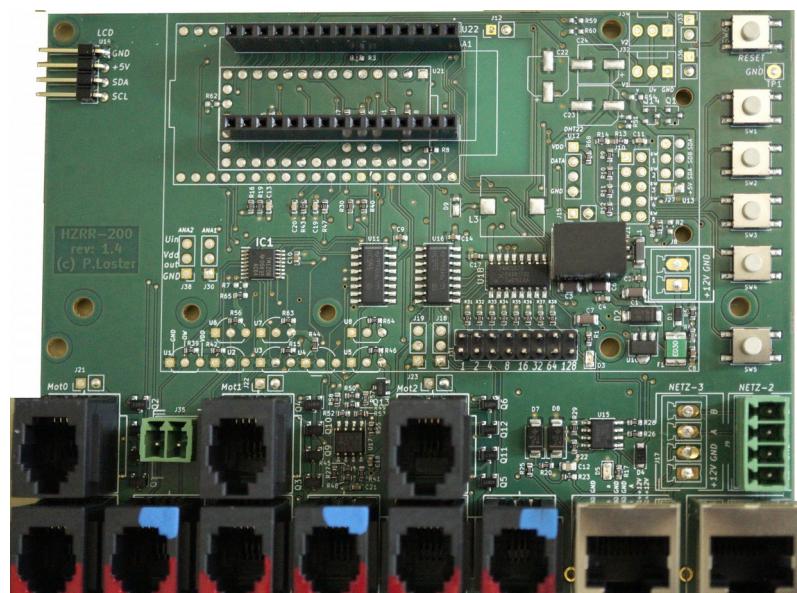
- Arduino Nano wird wie im Bild aufgesteckt. Richtung beachten! Nicht versetzt aufstecken.
- 4-poliges Kabel wie im Bild an die Stecker anschließen. POLUNG BEACHTEN !!!  
Bemerkung: Die Kabel haben jedesmal andere Farben, da sie von einem 20-poligen Flachkabelbund abgetrennt werden.





**ACHTUNG: Vorsichtig auftragen dass kein Lack in das Innere der Buchse fließt !!!**

Rücklauf-Temperatur Buchsen **blau** kennzeichnen:



### 2.6.3 Test Aufbau

Das Modul wird mit 12VDC versorgt, über den selben Stecker wird auch das serielle RS485 Netzwerk angeschlossen. Diese Leitungen sind auch 1:1 mit den RJ45 Netzwerksteckern verbunden.

**ACHTUNG: Es handelt sich NICHT UM EINE ETHERNET (Computer Netzwerk) Verbindung!! auch wenn die Modular 8P8 Buchsen mit Schirm gleich sind.**

Die Module können auch über die RJ45 8P8 verkabelt werden.



36: USB nach RS485 Konverter  
(A/+:weiß B/-:braun)



Die Netzwerkverbindung wird über einen RJ45 - USB Adapter hergestellt.

### 2.7 Der RS485 - USB Adapter

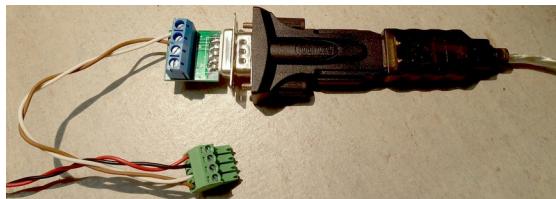


Bild 2: RS-485 Adapter von Digitus mit 4-Poligem Stecker zum Modul

Es können auch andere RS485 - USB Adapter verwendet werden. Der Vierpolige Stecker liefert 12V DC (rot + / schwarz -) sowie die RS485 A (weiss) und B (braun).

### 2.8 Serielles Terminal am PC oder RPi

Auf dem PC wird zusätzlich zur Arduino IDE ein serielles Terminal geöffnet. Als Parameter werden das USB-Interface, 115200 Baud, 8 Bit, no parity, 2 Stopbits eingestellt.

**WICHTIG: Es sind nun ZWEI USB-serielle Schnittstellen mit dem Modul verbunden, und es laufen ZWEI verschiedene serielle Terminal-Programme in jeweils einem eigenen Fenster !!!**

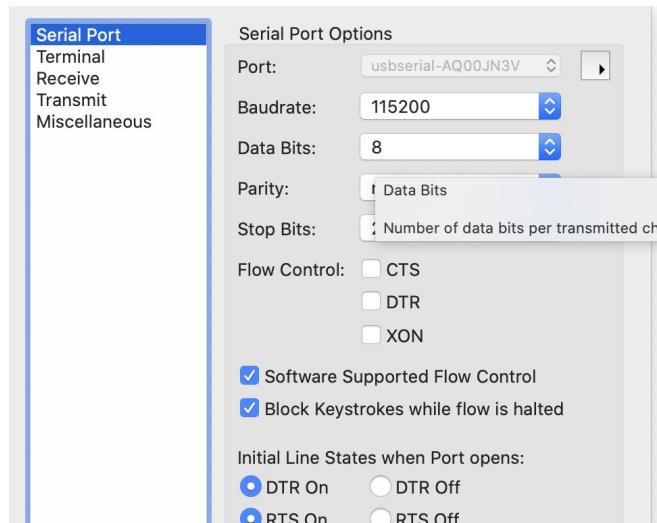
1. Über die USB-Schnittstelle des Arduino Nano, darstellbar im seriellen Terminal der Arduino IDE (Start über die Schaltfläche rechts oben in der Arduino IDE), z.B. in Bild Bild 41 oder Bild 42 und folgende.



## 2. Über den USB-RS485 Konverter Bild 36 am RS485 Bus; darstellbar über das in Bild Bild 37 und Bild 38 beschriebene serielle Terminal



37: Serielles Terminal öffnen, hier CoolTerm (Mac), alternativ GtkTerm (Linux)



### 2.8.1 Inbetriebnahme Tests

#### 2.8.1.1 LCD-Test

Nach Start des Programms wird ein Startbildschirm angezeigt, DANN nach einigen Sekunden wird der LCD-Test angezeigt:

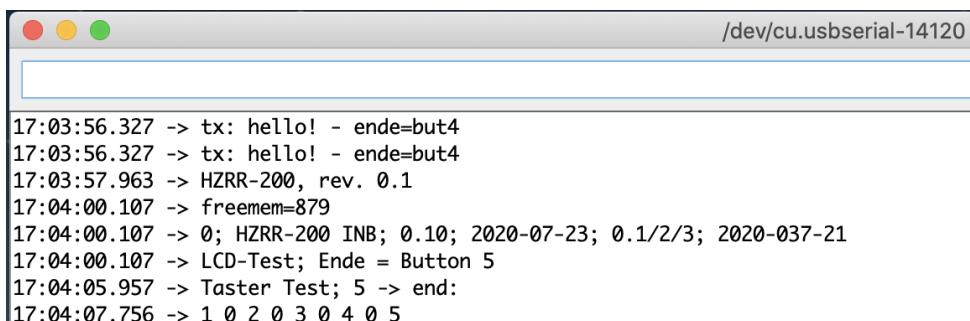
### 2.8.1.2 Taster-Test (Button-Test)

Bild 40:



Tasten: But5 Loch But4 But3 But2 But1 --- RESET

Gleichzeitig wird auf dem Arduino Terminal (Start oben rechts in der Arduino IDE) folgendes angezeigt:



```
17:03:56.327 -> tx: hello! - ende=but4
17:03:56.327 -> tx: hello! - ende=but4
17:03:57.963 -> HZRR-200, rev. 0.1
17:04:00.107 -> freemem=879
17:04:00.107 -> 0; HZRR-200 INB; 0.10; 2020-07-23; 0.1/2/3; 2020-037-21
17:04:00.107 -> LCD-Test; Ende = Button 5
17:04:05.957 -> Taster Test; 5 -> end:
17:04:07.756 -> 1 0 2 0 3 0 4 0 5
```

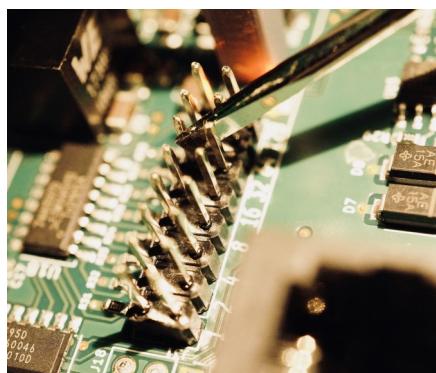
Drückt man die Tasten (Buttons) 1 bis 4 so erscheint die zugehörige Nummer, beim Loslassen wird wieder 0 angezeigt. Nach Drücken der Taste 5 wird zum nächsten Test weitergeschaltet.

### 2.8.1.3 Address- und Temperatursensor-Test

Auf dem seriellen Terminal der Arduino IDE erscheint jede Sekunde eine neue Anzeige der Form:

```
Temp4 degC=-127.00; Temp5 degC=-127.00; Temp6 degC=-127.00; Temp7 degC=-127.00dC; TZ=25.7
Moduladresse = 0x20 = 32
Temp0 degC=27.56; Temp1 degC=-127.00; Temp2 degC=-127.00; Temp3 degC=-127.00; ->But4
Temp4 degC=-127.00; Temp5 degC=-127.00; Temp6 degC=-127.00; Temp7 degC=-127.00dC; TZ=25.7
Moduladresse = 0x20 = 32
Temp0 degC=27.56; Temp1 degC=-127.00; Temp2 degC=-127.00; Temp3 degC=-127.00; ->But4
```

Der Reihe nach werden die gegenüberliegenden Pins der Adress-Stiftleiste gebrückt, z.B. mit einem blanken Schraubenzieher. **Pins nicht verbiegen!**



Auf der Ausgabe des seriellen Monitors der Arduino IDE kann dann die vom Modul erkannte Adresse abgelesen werden: Je nachdem welche Stifte gebrückt wurden:

„Moduladresse = 1“ oder 2, 4, 8, 16, 32, 64 oder 128.

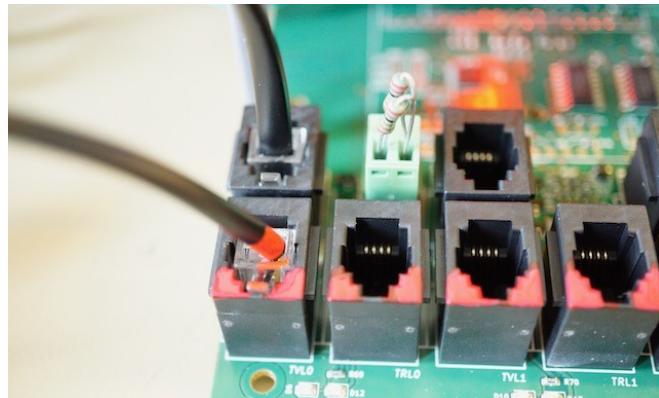


Bild 46: Raum-Thermometer Simulation mit 1100 Ohm

Als nächstes wird in jede der 6 Modular 4P4 Buchsen (auch RJ10 bezeichnet) ein Temperatursensor gesteckt. Es genügt ein Sensor der jeweils in eine Buchse gesteckt wird. Auf dem seriellen Monitor der Arduino IDE wird dann die Temperatur angezeigt, in etwa die Umgebungstemperatur des Sensors, beginnend mit Temp0 bis Temp5. Temperatur 6 und 7 haben keine Stecker.



Zuletzt wird der Eingang der Raumtemperatur geprüft.

Anstecken des Raumthermostaten von Technische Alternative oder als Ersatz einen Widerstand von 1100 Ohm ermöglicht das Prüfen der Anzeige „Tz=25.7“ - gemeint ist auch hier °C .

Mit der Taste4 (But4) geht es zur nächsten Anzeige:

#### 2.8.1.4 Motor-Test

```
19:42:38.010 -> ; Temp4 degC=-127.00; Temp5 degC=-127.00; Temp6 degC=-127.00; Temp7 degC=-127.00dC; TZ=25.7
19:42:39.086 -> ; Moduladresse = 0x0 = 0
19:42:39.194 -> ; Temp0 degC=26.37; Temp1 degC=-127.00; Temp2 degC=-127.00; Temp3 degC=-127.00; ->But4
19:42:39.194 -> ; Temp4 degC=-127.00; Temp5 degC=-127.00; Temp6 degC=-127.00; Temp7 degC=-127.00dC; TZ=25.7
19:42:40.274 -> ; Moduladresse = 0x0 = 0
19:42:40.379 -> ; Temp0 degC=26.37; Temp1 degC=-127.00; Temp2 degC=-127.00; Temp3 degC=-127.00; ->But4
19:42:40.379 -> ; Temp4 degC=-127.00; Temp5 degC=-127.00; Temp6 degC=-127.00; Temp7 degC=-127.00dC; TZ=25.7
19:42:41.456 -> Motor Test
19:42:41.456 -> Taste 1:nächster; 2:vorh.Motor /// 3:Auf; 4:Zu; Taste 5 Ende
19:42:41.456 -> Motor0 Zu
19:42:41.456 -> iMot=2.09mA;
```

```

19:38:53.191 -> iMot=1.05mA;
19:38:54.192 -> iMot=0.70mA;
19:38:55.203 -> iMot=0.00mA;
19:38:56.202 -> iMot=0.00mA;
19:38:57.206 -> iMot=0.00mA;
19:38:58.217 -> iMot=0.00mA;
19:38:59.220 -> iMot=0.00mA;
19:39:00.192 -> iMot=0.00mA;
19:39:00.407 -> Motor0
19:39:01.202 -> iMot=0.00mA;
19:39:02.207 -> iMot=0.00mA;
19:39:03.211 -> Motor0 Auf
19:39:03.211 -> iMot=2.09mA;
19:39:04.220 -> iMot=12.54mA;

```

- Taste1: nächster Motor Nr. 0, 1, 2
- Taste2: vorheriger Motor
- Taste3: Ventil auf
- Taste4: Ventil zu
- Taste5: Ende

Mit den Tasten kann der Motor gewählt und in beide Richtungen bewegt werden. Der Test ist mit einem Motor in allen drei Motor-Ausgängen durchzuführen.

Mit Taste 5 kommt man zum nächsten Test:

### 2.8.1.5 Kommunikationstest

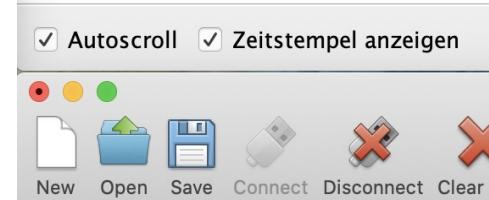
Bild 49 zeigt oben die Ausgabe der seriellen Terminals der Arduino IDE (STA1) und unten jenes das mit dem RS486-USB Wandler (STUW) verbunden ist.

- alle paar Sekunden schreibt das Regler Modul auf das STA1  
**'tx: '**  
und sendet dann den Text  
**'hello! - ende=but4'**  
über die serielle Schnittstelle und aktiviertem RS485 Sender-Treiber  
and den angeschlossenen RS495 - USB Adapter.
- Auf dem STA1 erscheint der komplette Text.
- Auf dem zweiten STUW erscheint nur der Text  
**'hello! - ende=but4'**
- Schreiben von Zeichen auf dem STUW unten wird dort nicht angezeigt, aber die Zeichen werden über RS485 an das STA1 gesendet und dort jeweils als z.B.  
**'rx=a'**  
angezeigt.

```

08:12:53.289 -> tx: hello! - ende=but4
08:12:58.265 -> tx: hello! - ende=but4
08:13:03.276 -> tx: hello! - ende=but4
08:13:08.066 -> rx=a
08:13:08.136 -> rx=s
08:13:08.172 -> rx=d
08:13:08.241 -> rx=f
08:13:08.279 -> tx: hello! - ende=but4
08:13:08.661 -> rx=w
08:13:08.661 -> rx=q
08:13:08.728 -> rx=e
08:13:08.867 -> rx=r
08:13:09.442 -> rx=1
08:13:09.545 -> rx=2
08:13:09.613 -> rx=3
08:13:09.680 -> rx=4
08:13:13.290 -> tx: hello! - ende=but4
08:13:18.284 -> tx: hello! - ende=but4
08:13:23.281 -> tx: hello! - ende=but4
08:13:28.282 -> tx: hello! - ende=but4
08:13:33.284 -> tx: hello! - ende=but4

```



```

hello! - ende=but4
g hello! - ende=but4
hello! - ende=but4
|

```

Bild 49: serial Terminals

## 2.8.2 Abschluss Inbetriebnahme

Aufkleber zur Kennzeichnung der Inbetriebnahme anbringen.



## 2.9 Anwendungsprogramm Installieren und Prüfen

TODO

## 3 Einrichten der Zentrale

Als Zentrlacomputer kann ein beliebiger PC, vorzugsweise mit Linux Betriebssystem eingesetzt werden. Die Anwendungen wurden in Python geschrieben mit Unterstützung von System- und Zeit Tasks.

Die Wahl fiel auf einen Raspberry Pi (RPi) mit dem aktuellen (2020) Raspbian Buster (auf Debian Basis). Dabei kommt ein Modell3, besser ein Modell 4 zum Einsatz.

Die SD Speicherkarten sind bei häufigem Schreiben anfällig für Ausfälle. Häufiges Schreiben ist bei dieser Anwendung nötig. Daher wird das System so eingerichtet, dass die SD-Karte im nur-lese (read-only) Modus betrieben wird. Alle Anwendungen laufen dabei im RAM des RPi. Die Anwendungsprogramme und die Daten befinden sich auf einem USB Stick, der einfach auszutauschen ist und ebenfalls einfach wiederherzustellen ist.

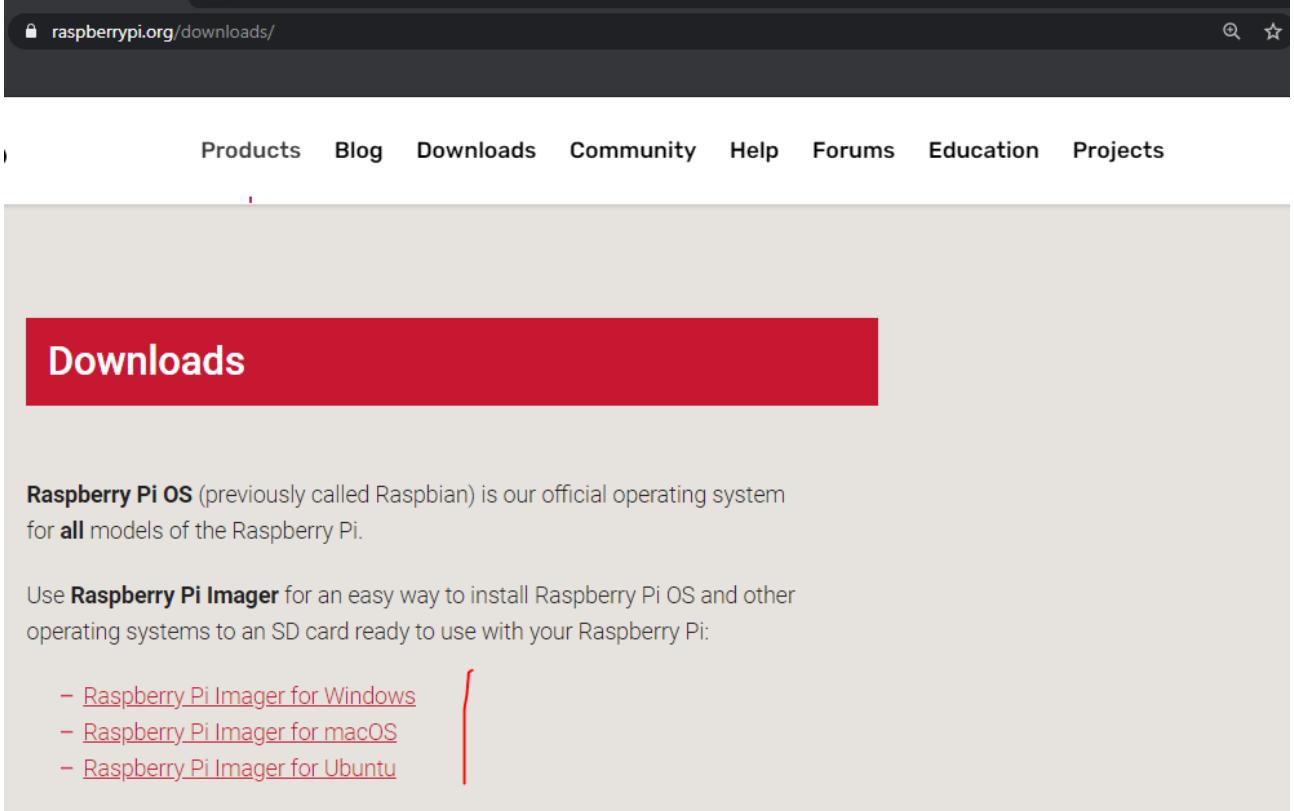
# Folgende Titel sind nur Vorschläge

## 3.1 Herstellen der SD-Karte

### Step 1

Besuche die Seite <https://www.raspberrypi.org/downloads/>.

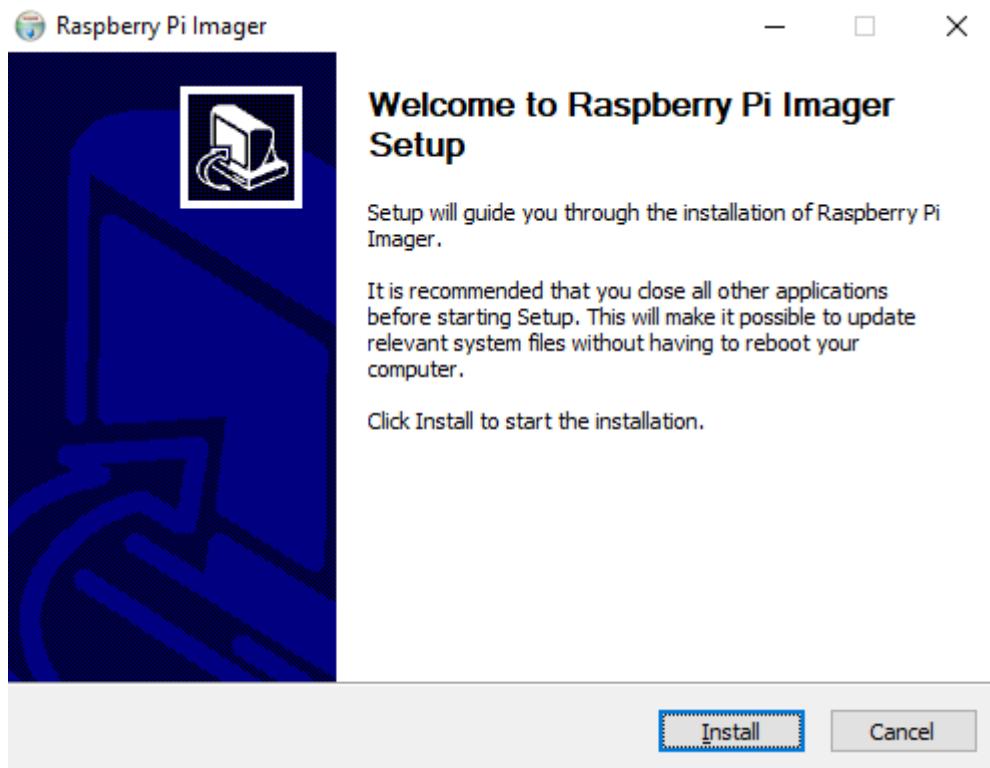
Download von dort den korrekten Imager für Ihr Betriebssystem.



The screenshot shows the official Raspberry Pi website at [raspberrypi.org/downloads/](https://www.raspberrypi.org/downloads/). The page features a navigation bar with links for Products, Blog, Downloads, Community, Help, Forums, Education, and Projects. A prominent red banner at the top contains the word 'Downloads'. Below this, a section for 'Raspberry Pi OS' is visible, stating it's the official operating system for all models. A list of download links for the Raspberry Pi Imager is provided, including links for Windows, macOS, and Ubuntu. A red vertical line is drawn to the right of the 'Downloads' banner, likely indicating a continuation of the previous section.

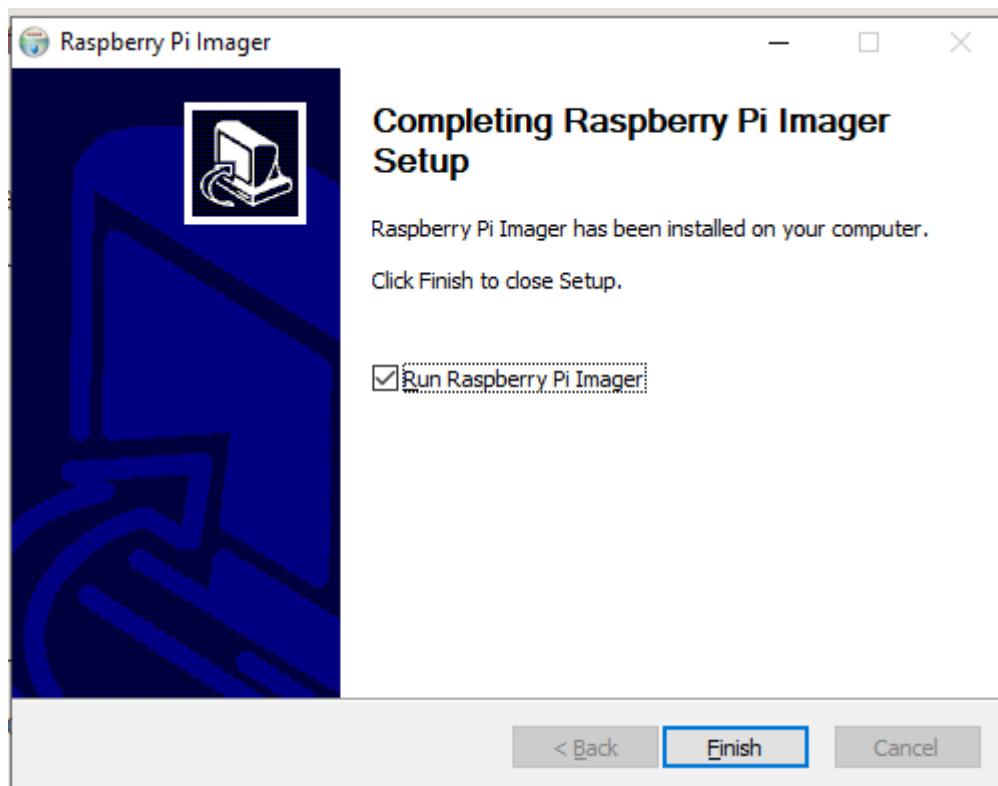
### Step 2

Installiere das Programm mit einem klick auf Install.



Der Prozess installiert sich automatisch in einen Ordner auf Laufwerk C:

Nachdem dieser abgeschlossen ist, setze den haken bei „Run Raspberry Pi Imager“



Und klicke auf Finish.

### **Step 3**

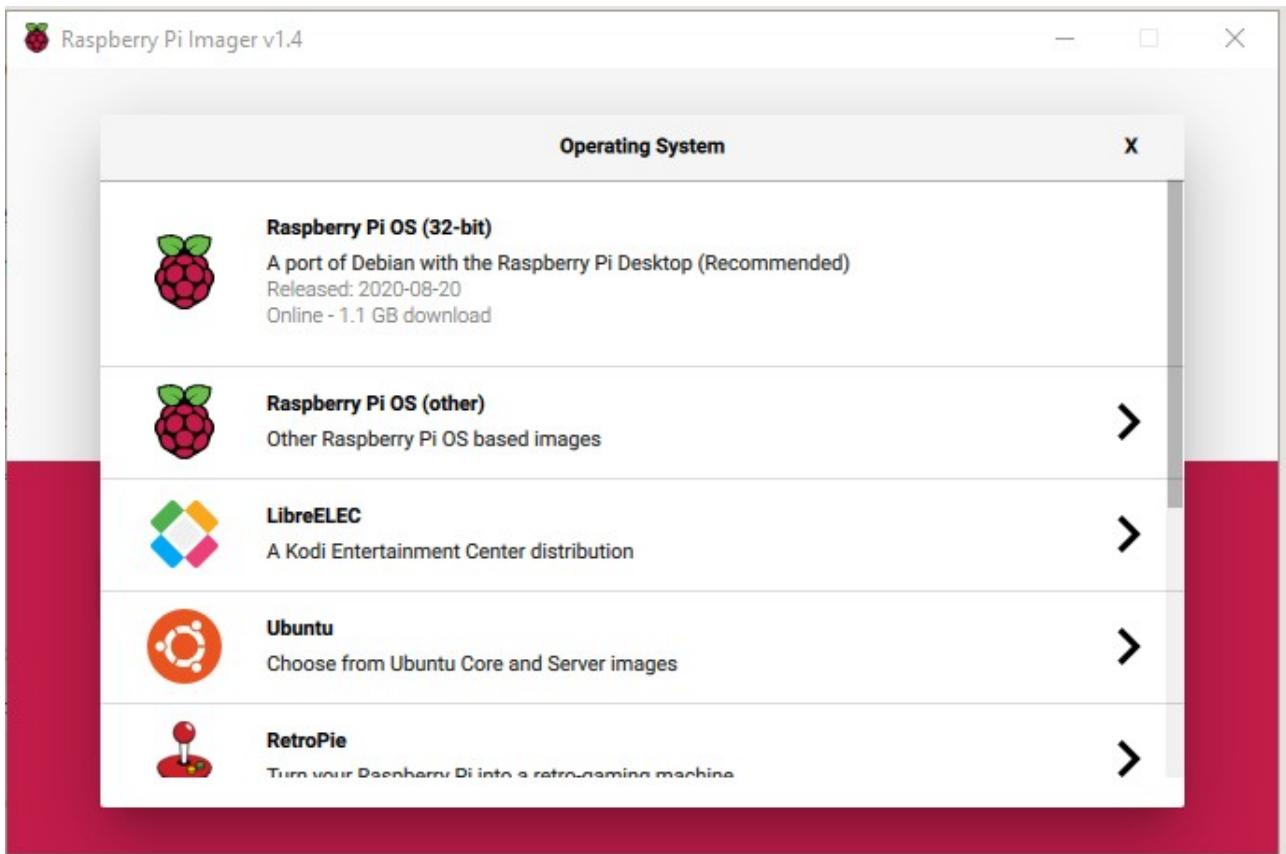
Stecke die SD-Karte nun in deinen Raspberry Pi. Bevor du den USB-Stecker zur Stromversorgung anschließt, stecke außerdem an diesen eine Ethernet-Verbindung zu deinem Router – falls dies nicht möglich ist, kann man auch später im Betriebssystem über das WLAN online gehen. Wenn du den Raspberry Pi per Remote einrichten möchtest, musst du nachdem er gestartet ist eine SSH-Verbindung aufbauen. Andernfalls stecke ebenso vor dem Start eine Tastatur via USB und einen Bildschirm via HDMI an.

### **Step 4**

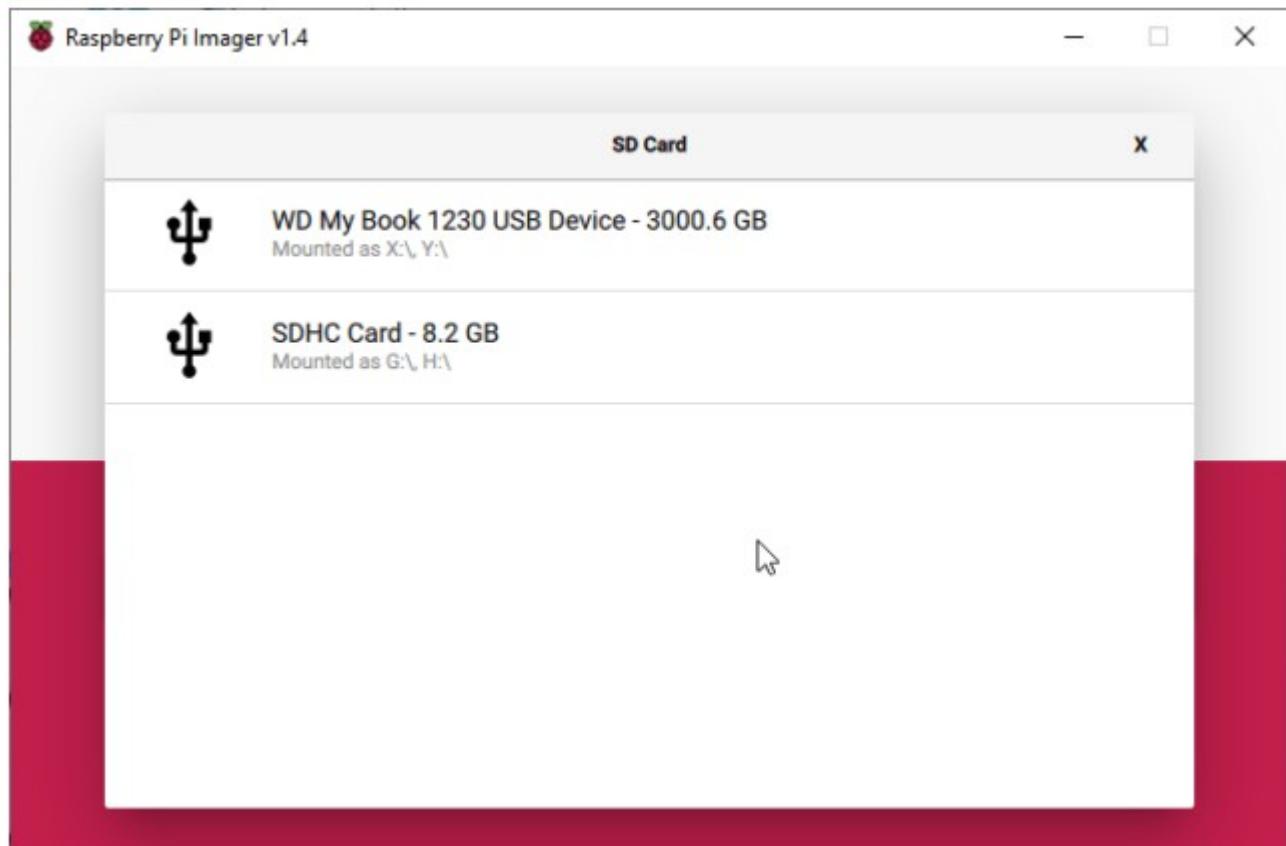
Zurück zum Raspberry Pi Imager.



Klicke auf „CHOOSE OS“ und wähle „Raspberry Pi OS(32-bit)“



Als nächstes klickt man auf „CHOOSE SD CARD“ – es öffnet sich ein Fenster in welchem die angesteckte SD Karte angezeigt werden sollte.

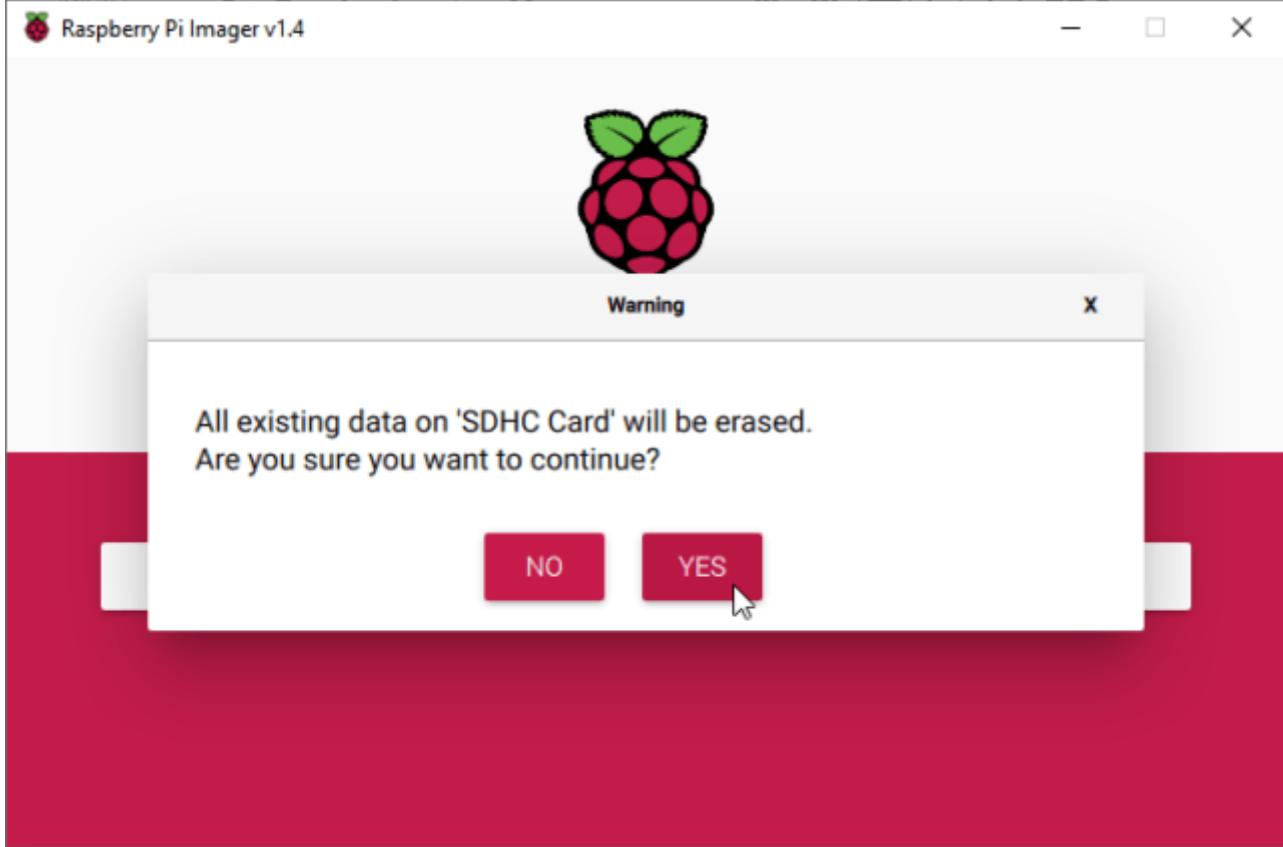


Sobald eine angesteckt wurde, erscheint diese hier.

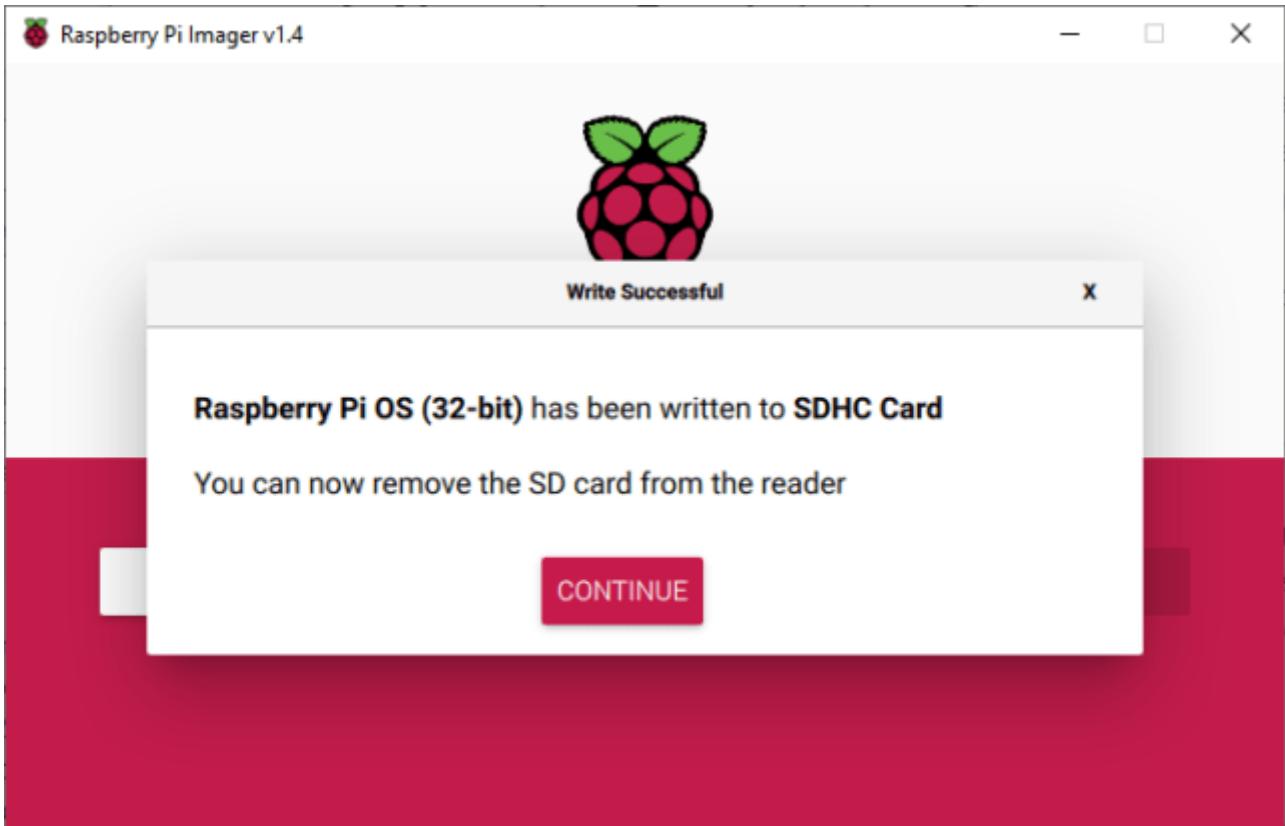
Klicke Sie an und im Haupt-Fenster klicke nun auf „WRITE“



Seid Ihr mit der Auswahl zufrieden, startet Ihr mit einem Klick auf **Write** die Installation des gewählten Systems auf die microSD-Karte. Vorher müsst Ihr noch bestätigen, dass der Inhalt gelöscht wird und dieses je nach System noch per Admin-Kennwort bestätigen.



Wartet nun ab, bis der Raspberry Pi Imager die Installation des Systems abgeschlossen hat. Da das Programm die jeweils aktuellste Version des gewählten Betriebssystem aus dem Internet herunterlädt und die Installation zudem automatisch auf Fehler überprüft, kann der Vorgang eine Weile dauern.



Nachdem die Installation samt Überprüfung abgeschlossen ist, könnt Ihr Euren Raspi mit der SD-Karte starten.

Der Vorgang schließt von alleine ab – sobald er fertig ist, stecke die SD Karte ab und Stecke sie danach in den Pi.

## Step 5

Erststart von Raspberry Pi mit Raspbian.

Nachdem einige Initiierungen zum erstmaligen Start von Raspbian getätigten wurden kommt eine grau-blaue Oberfläche samt Menü, durch welches man sich mittels der Pfeiltasten, Enter und Escape auf der Tastatur bewegen kann. In dieser werden nun einige Einstellungen durchgeführt.

Man öffne nun ein Terminal mit einen klick darauf. Dieser befindet sich meistens oben als Icon sichtbar.

Nun schreibe

```
sudo raspi-config
```

In den Terminal und bestätige es mit der „ENTER“ Taste.

Es erscheint nun dieses Konfigurationsprogramm:

hzrr200 Anleitung

26 von 37

ggf. fehlerhafte Vorläufige Version 1.08



Benutze die Pfeiltaste nach unten bis man an dem punkt „7 ADVANCED OPTIONS“ angekommen ist und bestätigt die Auswahl mit der „ENTER“ Taste.



Wähle den Punkt „A1 EXPAND FILESYSTEM“ aus und bestätige wieder mit der „ENTER“-Taste.  
Falls man gefragt wird ob man neustarten möchte, drücke auf Ja (YES),  
Falls neugestartet wurde, öffne erneut ein Terminal und schreibe:

Sudo raspi-config

Klicke wieder auf „7 ADVANCED OPTIONS“

Navigiere diesmal mit den Pfeiltasten auf den Punkt „AB OVERLAY FS“ und bestätige diesen mit der „ENTER“-Taste.

Bestätige die nächste Frage mit „JA“(YES) und die darauffolgende mit „NEIN“(NO) (Sperren der Boot Partition.)

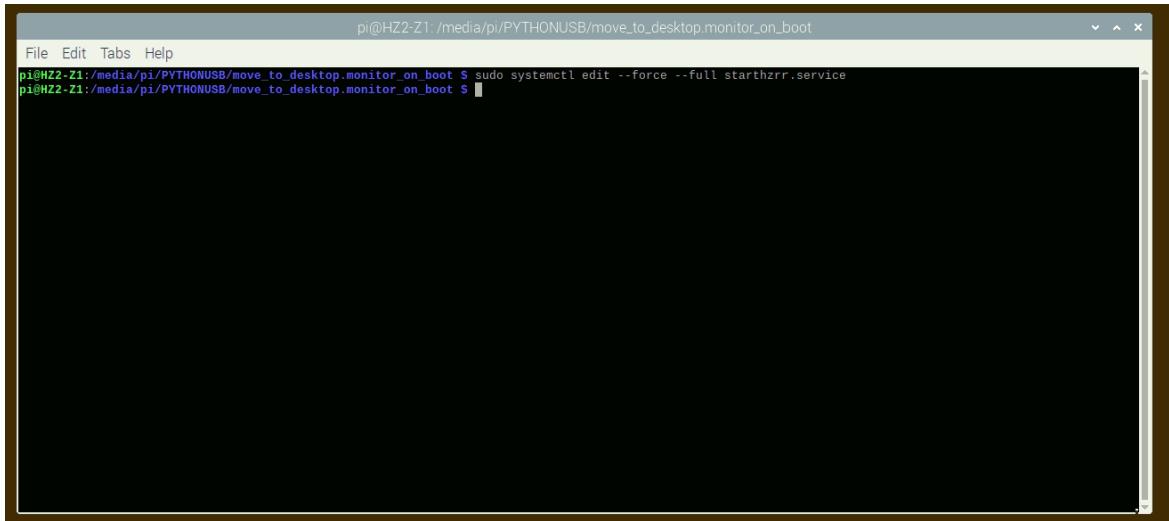
Dieser Prozess wird einige Zeit in Anspruch nehmen. (1-2 Minuten.)

## Step 6

Sobald der Vorgang abgeschlossen ist, Drücke die „ESC“ Taste um wieder zurück in das Terminal und raus aus der Raspi-Conf. Manchmal muss man 2-3 mal ESC Drücken.

Als nächstes schreiben wir in die Zeile:

```
sudo systemctl edit --force --full starthzrr.service
```



Es könnte eine Abfrage kommen, welcher Editor benutzt werden soll.

Diese einfach mit der Taste „1“ Bestätigen.

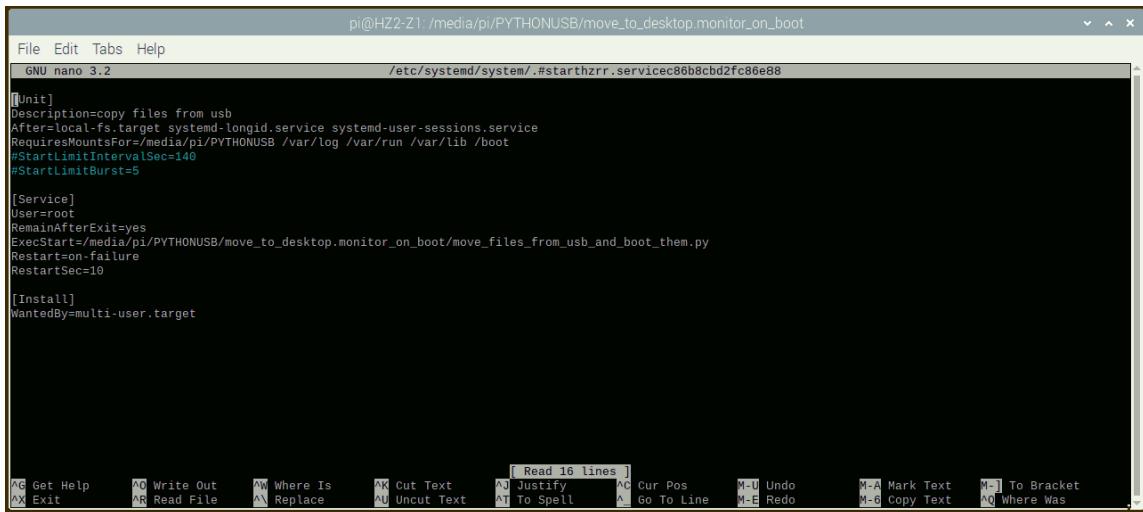
Es hat sich nun ein Editor Fenster im Terminal geöffnet.

Markiere nun den folgenden Text mit der linken Maustaste – achte darauf nichts weiteres zu markieren, denn in Linux kopiert wer markiert – also man überschreibt das gerade kopierte.

```
[Unit]
Description=copy files from usb
After=local-fs.target systemd-longid.service systemd-user-sessions.service
RequiresMountsFor=/media/pi/PYTHONUSB /var/log /var/run /var/lib /boot
#StartLimitIntervalSec=140
#StartLimitBurst=5

[Service]
User=root
RemainAfterExit=yes
ExecStart=/usr/bin/python3 /media/pi/PYTHONUSB/move_to_desktop.monitor_on_boot/
move_files_from_usb_and_boot_them.py
Restart=on-failure
RestartSec=10

[Install]
WantedBy=multi-user.target
```



pi@HZZ-Z1: /media/pi/PYTHONUSB/move\_to\_desktop.monitor\_on\_boot

GNU nano 3.2 /etc/systemd/system/.#starthzrr.servicec86b8cbd2fc86e88

```
[Unit]
Description=copy files from usb
After=local-fs.target systemd-longid.service systemd-user-sessions.service
RequiresMountsFor=/media/pi/PYTHONUSB /var/log /var/run /var/lib /boot
#StartLimitIntervalSec=140
#StartLimitBurst=5

[Service]
User=root
RemainAfterExit=yes
ExecStart=/media/pi/PYTHONUSB/move_to_desktop.monitor_on_boot/move_files_from_usb_and_boot_them.py
Restart=on-failure
RestartSec=10

[Install]
WantedBy=multi-user.target
```

Terminal menu bar: File Edit Tabs Help

Bottom status bar: ^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo M-A Mark Text M-] To Bracket ^X Exit ^R Read File ^N Replace ^U Uncut Text ^T To Spell ^C Go To Line M-E Redo M-B Copy Text ^Q Where Was

Sobald der Text Markiert wurde – beweg den Cursor nun in das Terminal Fenster mit dem offenen Editor.

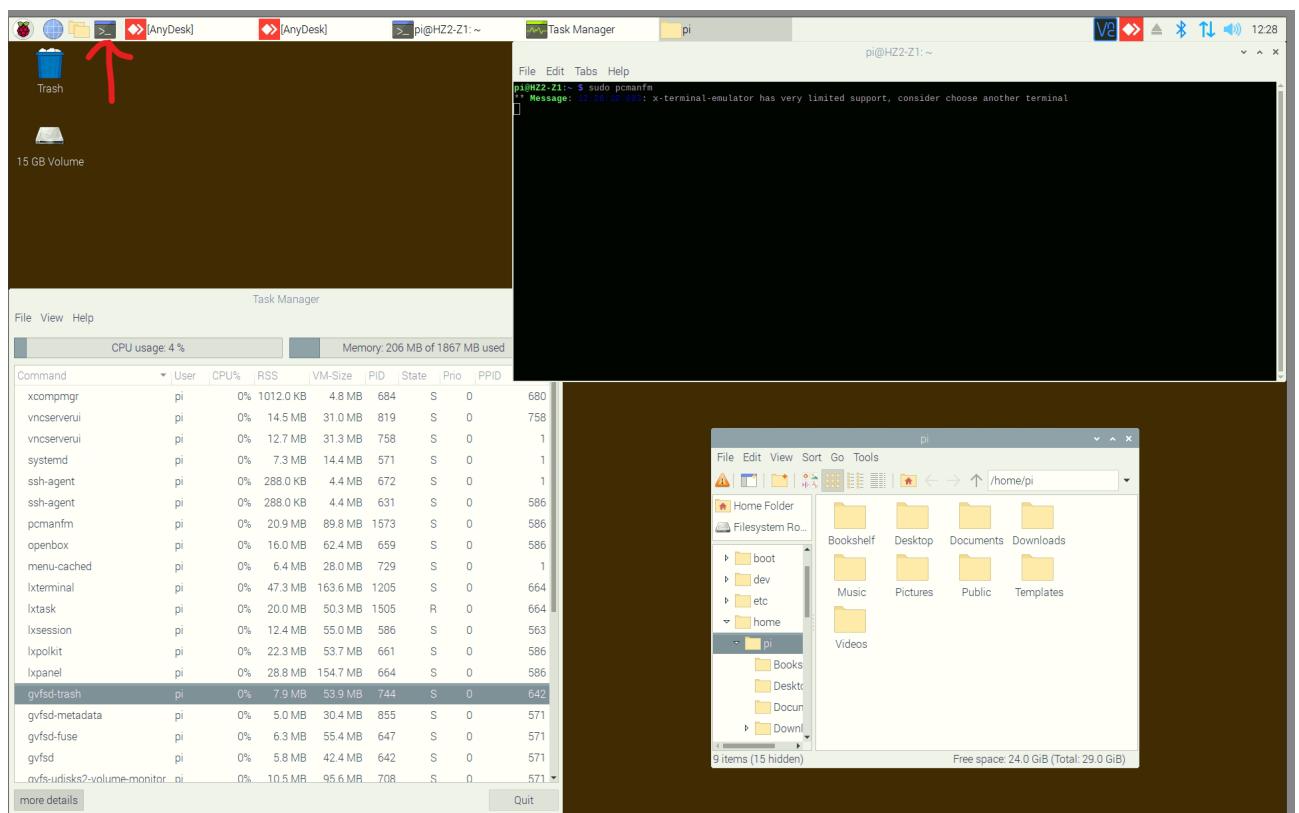
Drücke nun auf dein „MAUSRAD“ (Nicht drehen, drücken) – der Text sollte nun im Editor erschienen sein.

Drücke nun die Tastenkombination „STRG“ + „X“ und direkt darauf „J“ (Bzw. Y – abhängig von der jeweiligen Systemsprache)

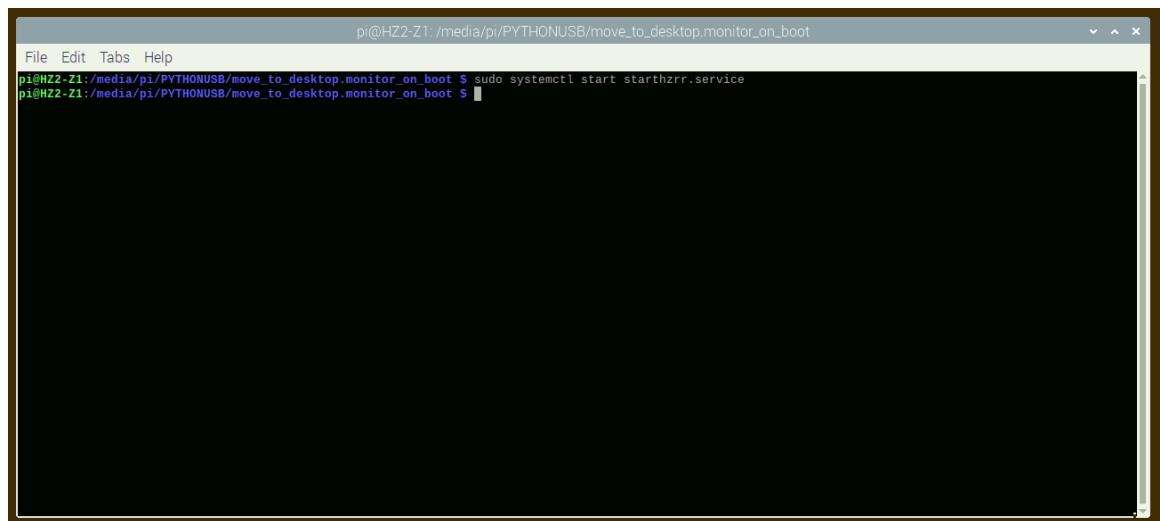
Bestätige mit „ENTER.“

## **Step 7**

Schreibe nun in das Terminal:



Sudo systemctl enable starthzrr.service



Sudo systemctl start starthzrr.service

Zum überprüfen ob der Service läuft, schreibe

systemctl status starthzrr.service

(Zum stoppen des Services, schreibe

`sudo systemctl stop starthzrr.service`

sollte aber nicht benötigt werden.)

Der Status müsste auf fail oder error gesetzt sein, da die Daten noch nicht vorhanden sind.

Schreibe nun in das Terminal

Watch `systemctl status starthzrr.service`

Dies wird alle 2 Sekunden den Befehl „`systemctl status starthzrr.service`“ ausführen und uns zeigen, was dort passiert.

Somit sehen wir auch ob es läuft und ggf. Fehler Meldungen werden uns auch angezeigt.

Nun ist dieser Part vorerst beendet und wir gehen weiter zum:

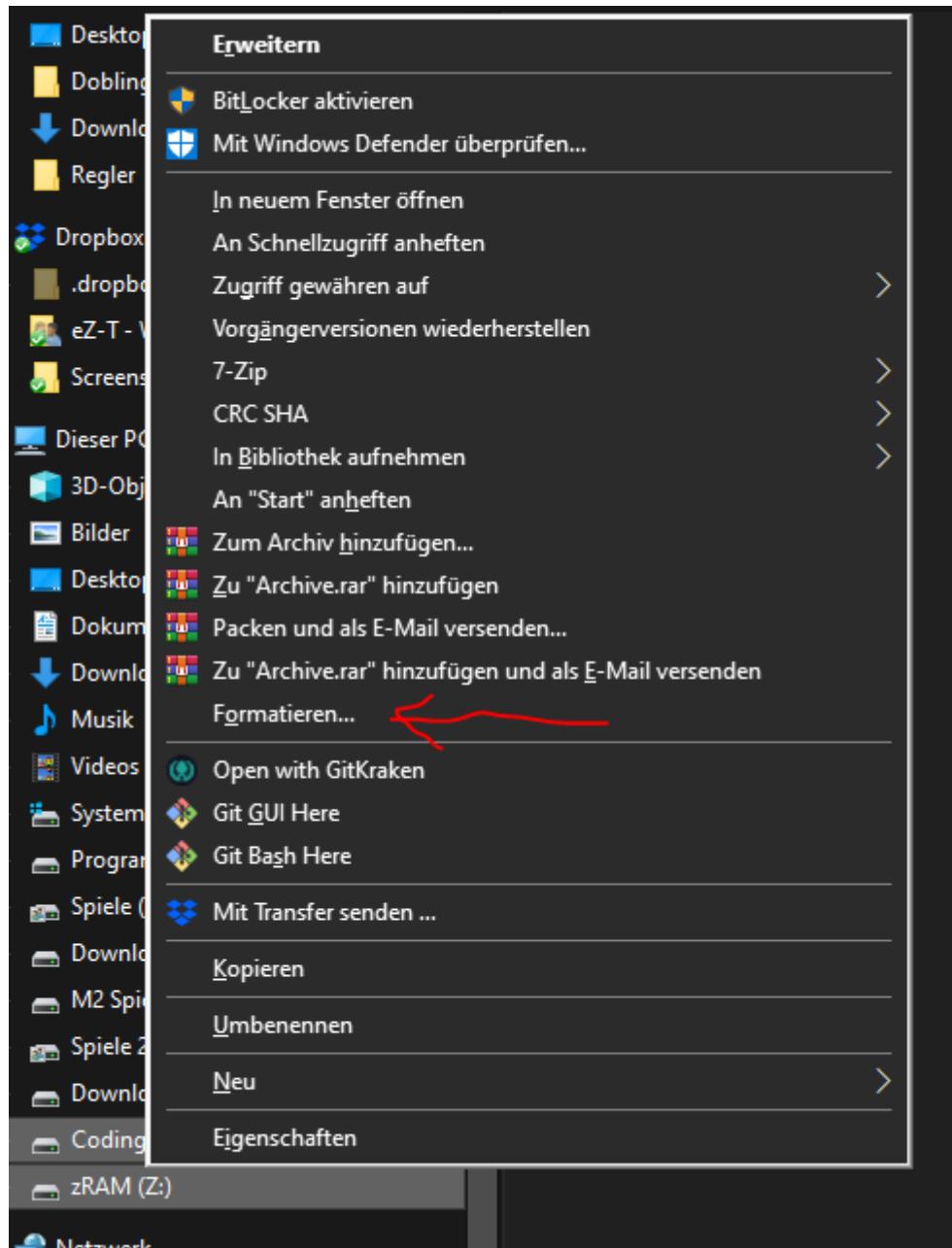
## 3.2 Einrichten des USB-Sticks

Man stecke den USB Stick mit den Daten der HZRR\_200 und stecke ihn an den Computer.

Falls der Stick nicht formatiert ist, formatiere ihn auf das NTFS Dateiformat.

Klicke dazu mir einem „RECHTSKLICK“ auf den USB-Stick.

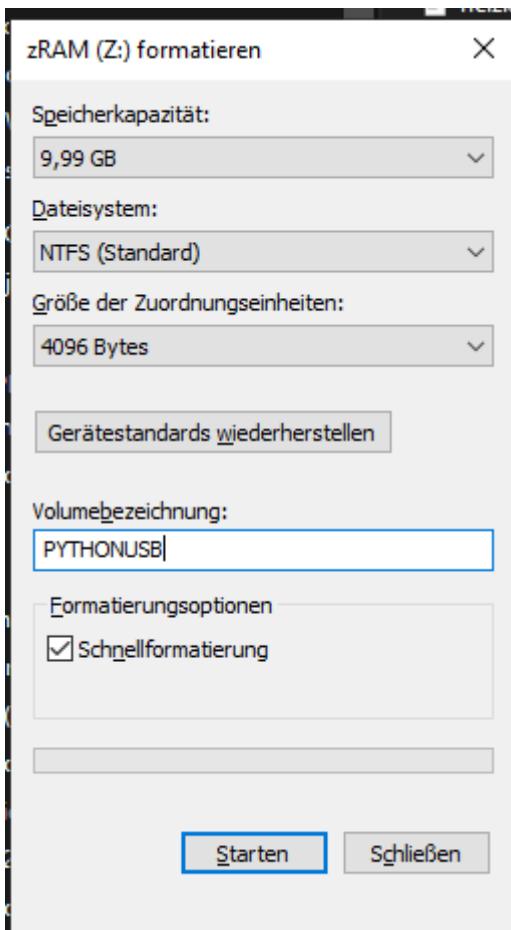
Wähle danach „FORMATIEREN“ aus.



Setze nun alle Einstellungen wie auf dem Folgenden Bild zu sehen

**WICHTIG: Der USB-Stick muss den Namen „PYTHONUSB“ haben.**

**WICHTIG: Der USB-Stick muss auf „FAT32“ Formatiert werden.**



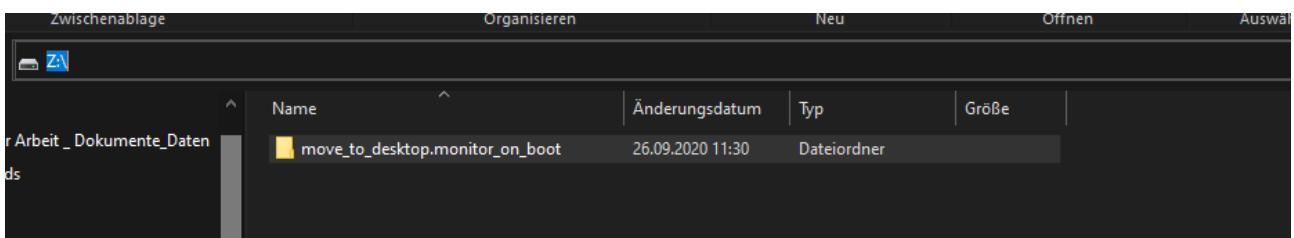
Klicke auf Starten.

Der Vorgang sollte relativ schnell beendet sein.

Öffne nun den USB-Stick im Datei-Explorer und kopiere den Ordner

„move\_to\_desktop.monitor\_on\_boot“

Auf den USB Stick.



Beginn dich nun in den Ordner auf dem USB-Stick, es müssen nun einige Dateien angepasst werden.

**Dieser Punkt wird noch verändert damit diese Informationen von einer lokalen Datei bezogen werden.**

Für den Moment müssen wir aber in den folgenden Dateien, folgende Zeilen Anpassen:

In Dateien:

```
[1] - ./usb_ser.py
[2] - ./hz_rr_log.py
```

Anpassen:

```
[1] - Z.23
```

serPort=

```
❶ select_log.py ❷ usb_ser.py
❸ move_to_desktop.monitor_on_boot > ❹ usb_ser.py > {} mb
1  # -*- coding: utf-8 -*-
2  """
3  module ser.pi
4  RS-485 Interface over USB adapter
5
6  Created on Sat Nov 19 10:37:27 2016
7
8  @author: pl
9  """
10
11 import serial
12
13 import modbus as rm
14 import vorlaut as vor
15
16 #serPort = "/dev/ttyUSB0" # USB0 might change
17 # fixed for the same adapter is always:
18 #serPort = "/dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A504YCFM-if00-port0"
19 # has problems if the adapter changes (because of serial number)
20 # so we use the USB-socket where the adapter is connected:
21 # Bottom socket next to Ethernet socket:
22 serPort = "/dev/serial/by-path/platform-3f980000.usb-usb-0:1.3:1.0-port0"
23 serPort = "/dev/serial/by-path/platform-3f980000.usb-usb-0:1.1.3:1.0-port0" #"/dev/serial/by-path/platform-3f980000.usb-usb-0:1.3:1.0-port0/move_to_desktop.monitor_on_boot/"
24
25
```

```
[2] - Z.75
```

serPort=

```
move_to_desktop.monitor_on_boot > ❶ hz_rr_log.py > {} time
50 filtFakt = 0.1    # Filter 2. Ordnung; filtFakt = Bewertung des neuen Wertes
51 v1zen = 0.0        # temperature from Zentrale Vorlauf; 0 -> no temperature
52 v1el = 0.0         # initial value
53
54
55 print("*****")
56 print("hz-rr-log")
57 print("creating logfile")
58 print("*****")
59
60 def wait(s) :
61     pass
62     #print(s)
63     #time.sleep(2.0)
64
65 wait('start')
66
67 #locPath="/home/pi/Desktop/monitor/"
68 #serPort = "/dev/ttyUSB0" # USB0 might change
69 # fixed for the same adapter is always:
70 #serPort = "/dev/serial/by-id/usb-FTDI_FT232R_USB_UART_A504YCFM-if00-port0"
71 # has problems if the adapter changes (because of serial number)
72 # so we use the USB-socket where the adapter is connected:
73 # Bottom socket next to Ethernet socket:
74 locPath = "/media/pi/pythonUSB/move_to_desktop.monitor_on_boot/log"
75 serPort = "/dev/serial/by-path/platform-3f980000.usb-usb-0:1.3:1.0-port0" #"/dev/serial/by-path/platform-3f980000.usb-usb-0:1.3:1.0-port0/move_to_desktop.monitor_on_boot"
76 # bottom left socket = 0:1.3:1.0-port0 on rasp 3b+
77 # 1.6.1"
```

Hinter „serPort=“ schreibt ihr für:

**Raspberry 3b+:**

```
"/dev/serial/by-path/platform-3f980000.usb-usb-0:1.1.3:1.0-  
port0"
```

#### Raspberry 4:

```
"/dev/serial/by-path/platform-fd500000.pcie-pci-0000:01:00.0-  
usb-0:1.4:1.0-port0"
```

Dies ist auf dem Raspberry 3 der untere linke USB-Port.

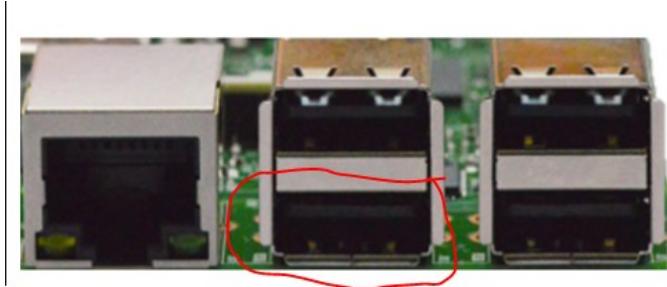


Figure 1: RPi3 USB-RS485 Adapter hier

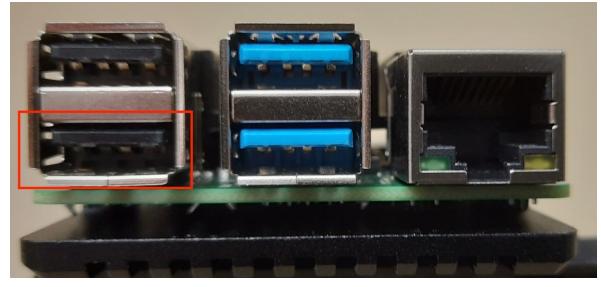


Figure 2: RPi4: USB-RS485-Adapter HIER

Dort muss unbedingt die serielle Schnittstelle eingesteckt werden.

**WICHTIG: Der „PYTHONUSB“ darf überall- aber nicht dort eingesteckt werden!**

Speichere die Dateien nun ab und ziehe den Stick ab und mache dich auf den Weg zum Raspberry.

### 3.3 Stromanschluss und Monitor Anschluss

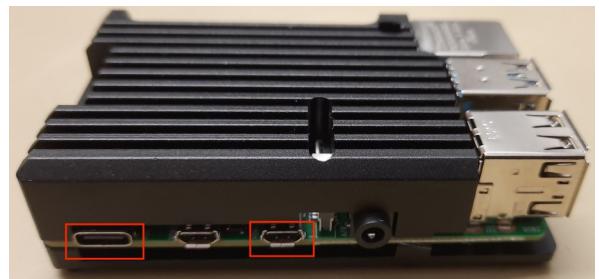


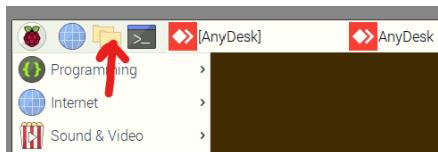
Bild 5: RPI4 USBC-5V  
Versorgungsspannung (links) und Micro-  
HDMI für Monitor (rechts)

### 3.4 Einsticken des USB-Sticks

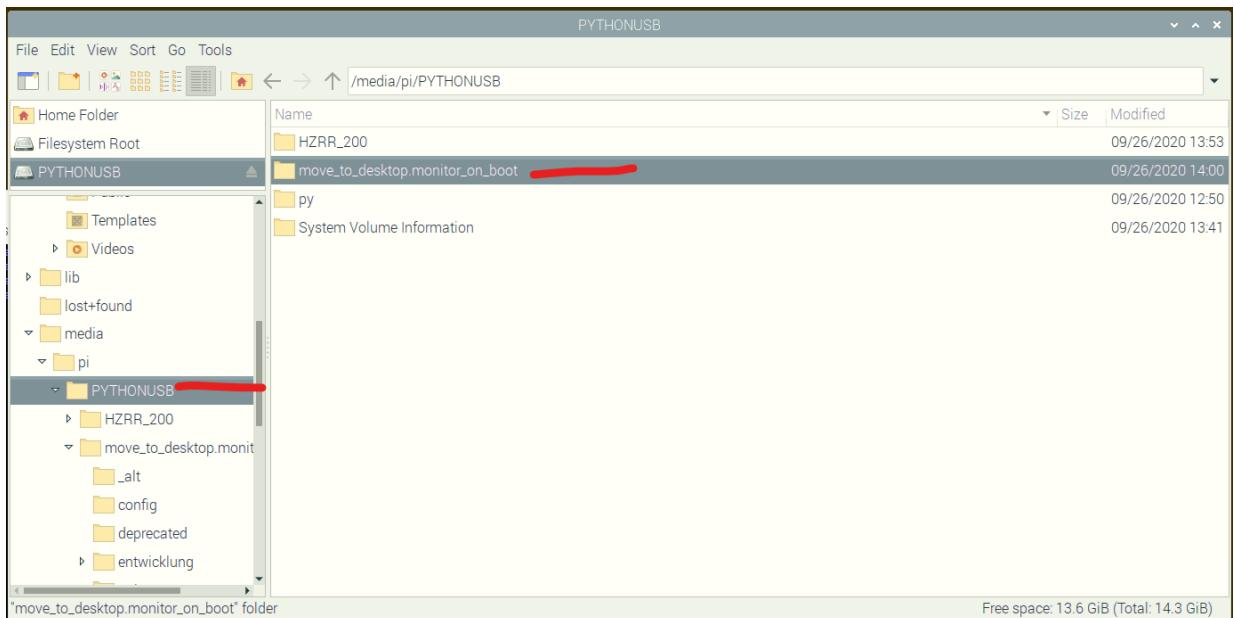
Stecke nun den USB-Stick in den Pi.

Bitte nehme nicht den USB unten links – neben dem Ethernet Adapter. Siehe Punkt 3.2.

Ist der USB angesteckt öffne deinen Datei-Explorer, dies machst du mit einem Klick auf das Gelbe Ordner Symbol oben in der Leiste.



Hier müsste nun der PYTHONUSB auftauchen.



In den nächsten 60-120 Sekunden sollte das Skript seine Arbeit erledigt haben.

Es ist keine weitere Interaktion erforderlich, der HZRR Monitor sollte nach einer kurzen Zeit erscheinen.

Zum überwachen des Scripts und zur Fehleridentifizierung, kann man im Terminal

Watch systemctl status starthzrr.service

```
pi@HZZ1: /media/pi/PYTHONUSB/move_to_desktop.monitor_on_boot
File Edit Tabs Help
Every 2.0s: systemctl status starthzrr.service
HZZ1: Sat Sep 26 14:52:02 2020
● starthzrr.service - copy files from usb
  Loaded: loaded (/etc/systemd/system/starthzrr.service; disabled; vendor preset: enabled)
  Active: activating (auto-restart) (Result: exit-code) since Sat 2020-09-26 14:51:57 CEST; 4s ago
    Process: 12762 ExecStart=/media/pi/PYTHONUSB/move_to_desktop.monitor_on_boot/move_files_from_usb_and_boot_them.py (code=exited, status=203/EXEC)
   Main PID: 12762 (code=exited, status=203/EXEC)
```

Eingeben.

Dies zeigt Rückgabewerte des Skripts im standardmäßigen 2 Sekunden Takt.

### **3.5 Ablauf beim Starten**

### **3.6**