

Ich gehe davon aus, dass Du Windows benutzt, und das nicht erst seit gestern. Also denke ich auch, dass Du Dir mit Hilfe der Suchmaschinen ein Programm holen kannst, und es auch unter Windows installiert kriegst.

Dies hier ist also nicht eine Anleitung für Leute wie meine Großmutter, die außer Kühen und Wiesen nichts kannte...

Als erstes holen wir uns mal den rpi-imager auf den Rechner und nutzen ihn um ein ganz normales Linux auf die SD-Karte zu laden. Wir nehmen dazu etwas ohne Desktop, weil den brauchen wir meistens nicht.

Wir gehen auch hier nicht darauf ein, dass Dein Netzwerk zuhause unkonventionell eingerichtet ist, sondern wir nehmen an, es hat eine übliche Fritzbox oder so und es gibt dort automatisch eine IP-Adresse...

Install Raspberry Pi OS using Raspberry Pi Imager

Raspberry Pi Imager is the quick and easy way to install Raspberry Pi OS and other operating systems to a microSD card, ready to use with your Raspberry Pi.

Download and install Raspberry Pi Imager to a computer with an SD card reader. Put the SD card you'll use with your Raspberry Pi into the reader and run Raspberry Pi Imager.

[Download for Ubuntu for x86](#)

[Download for Windows](#)

[Download for macOS](#)

To install on Raspberry Pi OS, type `sudo apt install rpi-imager` in a Terminal window.



Vorab: Ich will nen T68 über TW39 an einem österreichischen AGT verwenden: Ö-Telex Und ich will das mit Centralex machen, werde also (Stand Februar 25) das „testing“ nehmen. Und ich machs mit Ethernet, nicht mit Wlan (mit Wlan geht's auch, das wird dann im Imager schon eingestellt...).

Ich zeig hier alles mit Raspberri PI OS
Raspberry Pi OS Lite

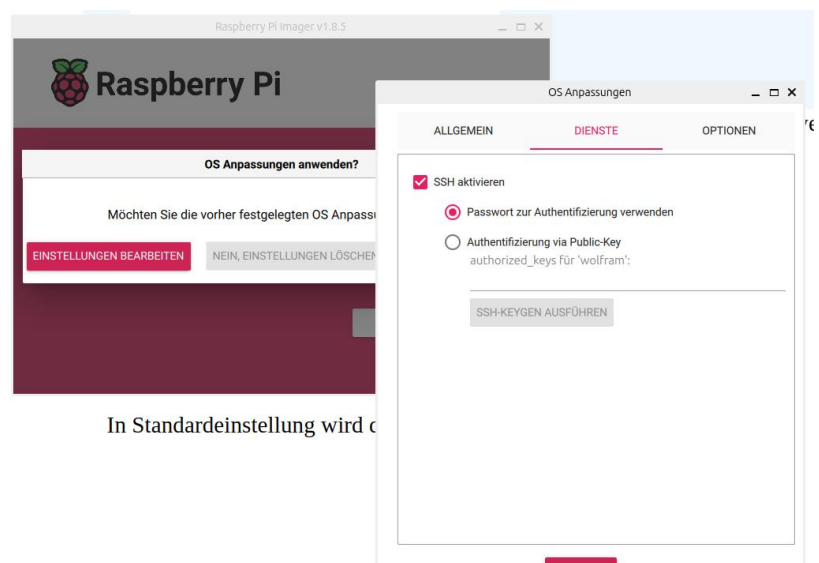
- Release date: November 19th 2024
- System: 32-bit
- Kernel version: 6.6
- Debian version: 12 (bookworm)

Wir schauen, dass wir hier schon „SSH“ aktivieren, dann brauchen wir später nicht mehr dahinterher zu laufen. Das geht wenn wir „weiter“ klicken und dann unter „Einstellungen bearbeiten“.

„SSH“ heißt „Secure Shell“ und gibt uns den Zugang zum Raspberry über Netzwerk.

Manchmal muß man jetzt bevor man weiterkommt Benutzernamen und Passwort angeben.

Wir machen es hier einfach:
Benutzername ist „pi“ und das Passwort auch.



Der Benutzername ist später noch wichtig, denn allerhand Zeug wird in einem Ordner gespeichert, der den Namen des Benutzers trägt. Das sehen wir später. Wenn Ihr das ändern wollt, macht das später einfach nochmal mit euren Daten, aber hier üben wir einfach. Und nein, ich erkläre nicht zweimal.

Ich schreibe Befehle ans System **kursiv**. Was nicht unbedingt wichtig oder zur Auswahl ist, mache ich **heller**. Du kannst die Befehle abschreiben oder sogar markieren und kopieren.

Und nochwas: „Test“ ist nicht gleich „test“ Ab sofort benehmen wir uns und beachten Groß- und Kleinschreibung. Wir sind schließlich nicht bei Windows. ;)

Bei uns wird der Ordner also „pi“ sein. Der Ordner pi liegt im „home“-Ordner und home liegt in der Wurzel „/“(Was in Windows C:\users\benutzername ist, ist hier eben /home/pi)

Nun wird das dann auf die SD-Karte geschrieben.

...stunden vergehen...

...und dann ist Imager fertig.



Jetzt wird's Zeit für den Raspberry: Karte rein, Netzkabel dran und USB-Netzteil...

Hurra, wir lernen Linux!

Nach ein, zwei Minuten kann ich mich einloggen (später bootet er viel schneller):

Oh, ja, welche IP-Adresse hat Dein Raspberry: Das verrät Dir Deine Fritzbox. Oder Du hast nen Bildschirm an den Raspberry angeschlossen. Dort zeigt ers Dir sogar...

Und wie „Putty“ funktioniert ist auch nicht schwer. Ich nutze ein Linux, daher hier meine Ansicht des Ganzen.

Mein Raspi hat hier die IP-Adresse 192.168.3.135 erhalten.

Ich weise also meine Konsole an, per SSH den Benutzer pi auf 192.168.3.135 anzurufen...

Es folgt der erste richtige Linuxbefehl:

ssh pi@192.168.3.135

Er fragt nach dem Passwort! Ich antworte **pi**

...und wie Bobele fragte: „Bin ich schon drin?“ Ja. Du bist.

```
pi@raspberrypi:~  
wolfram@wolframW530:~$  
wolfram@wolframW530:~$ ssh pi@192.168.3.135  
pi@192.168.3.135's password:  
Linux raspberrypi 6.1.21-v7+ #1642 SMP Mon Apr 3 17:20:52 BST 2023 armv7l  
  
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.  
Last login: Wed Feb 26 16:26:50 2025 from 192.168.3.14  
  
Wi-Fi is currently blocked by rfkill.  
Use raspi-config to set the country before use.  
pi@raspberrypi:~$
```

Als erstes fahre ich ein Update: Der Befehl dazu: **sudo apt update**

sudo heißt „SuperUser mach mal...“ und „apt update“ ist der Update-Befehl an „apt“, die Paketverwaltung. Der Raspi ist nicht dumm, alle Programmpakete sind im Internet und er holt sich dort alles... also dann mach mal...

Es folgt **sudo apt upgrade** um das dann auch durchzuführen.

Nun, das dauert jetzt mal ein, zwei ...öh... Minuten. Zeit für die Werbung...



...Werbung ende.

Wir starten nun ein paar kleine Dinge, die der Raspi braucht...

sudo raspi-config

Mit Pfeiltasten, enter und Leertaste navigieren wir.

Im Menu "Interface options" setzen wir „SPI“ auf disable

bei "System" wollen wir, dass er „Wait for network at boot“ enable hat (ist aber wenig relevant: Wenns nicht im Menü ist, ist's auch gut...)

In den "Localisation Options": such ich mir ein schickes „DE“ aus und aktiviere es.

Nun kommt der erste reboot. Automatisch. Später tun wir das selbst.

Wir loggen uns wieder ein, denn beim reboot sind wir aus dem System geflogen.

ssh pi@192.168.3.135

Der Befehl **sudo apt install python3 python3-commentjson python3-pigpio git mc screen byobu** weist apt an, die Programme „python3“, „python3-commentjson“, „python3-pigpio“, „git“, „screen“, „byobu“ und „mc“ zu installieren (wenn ich jetzt gleich alle reinschreib, brauch ich es nachher nimmer...).

Er fragt nett, ob er es auch wirklich tun soll, ich sage Y für Yes.

Ich benutze auch gern „mc“, den „Midnight Commander“. Wer aus DOS-Zeiten kommt, kennt seinen Vater: „Norton Commander“. Oben hatten wir es schon drin. Einzeln würde es es heißen: **sudo apt install mc**

Wenig später ist alles erledigt und die Eingabezeile ist wieder da.

Nun muss ich pigpio starten, damit der Raspi die einzelnen I/O-Pins lesen kann.

sudo systemctl start pigpiod

ob das geklappt hat? Sehen wir nach:

sudo systemctl status pigpiod

```
pi@raspberrypi:~  
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.  
Last login: Wed Feb 26 16:26:50 2025 from 192.168.3.14  
  
Wi-Fi is currently blocked by rfkill.  
Use raspi-config to set the country before use.  
  
pi@raspberrypi:~ $ sudo apt update  
Get:1 http://raspbian.raspberrypi.org/raspbian bullseye InRelease [15.0 kB]  
Get:2 http://archive.raspberrypi.org/debian bullseye InRelease [39.0 kB]  
Get:3 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf Packages [13.2 MB]  
Get:4 http://archive.raspberrypi.org/debian bullseye/main armhf Packages [329 kB]  
Get:5 http://raspbian.raspberrypi.org/raspbian bullseye/non-free armhf Packages [107 kB]  
Fetched 13.7 MB in 11s (1276 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
41 packages can be upgraded. Run 'apt list --upgradable' to see them.  
pi@raspberrypi:~ $
```

```
pi@raspberrypi:~  
Use raspi-config to set the country before use.  
  
pi@raspberrypi:~ $ sudo apt install python3 python3-commentjson python3-pigpio mc  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
python3 is already the newest version (3.9.2-3).  
python3 set to manually installed.  
The following additional packages will be installed:  
  libgpm2 libpigpio-dev libpigpio1 libpigpiod-if-dev libpigpiod-if1 libpigpiod-if2-1  
Sudo  
pi@raspberrypi:~ $  
Setting up libpigpiod-if-dev (1.79-1+rpt1) ...  
Setting up libpigpio-dev (1.79-1+rpt1) ...  
Setting up pigpio (1.79-1+rpt1) ...  
Processing triggers for libc-bin (2.31-13+rpt2+rp1+deb11u1) ...  
Processing triggers for man-db (2.9.4-2) ...  
Processing triggers for mailcap (3.69) ...  
pi@raspberrypi:~ $ sudo systemctl start pigpiod  
pi@raspberrypi:~ $ sudo systemctl status pigpiod  
● pigpiod.service - Daemon required to control GPIO pins via pigpio  
   Loaded: loaded (/lib/systemd/system/pigpiod.service; disabled; vendor preset: enabled)  
   Active: active (running) since Wed 2025-02-26 17:08:59 GMT; 25s ago  
     Process: 1146 ExecStart=/usr/bin/pigpiod -l (code=exited, status=0/SUCCESS)  
    Main PID: 1147 (pigpiod)  
      Tasks: 4 (limit: 1595)  
         CPU: 2.009s  
        CGroup: /system.slice/pigpiod.service  
               └─1147 /usr/bin/pigpiod -l  
  
Feb 26 17:08:59 raspberrypi systemd[1]: Starting Daemon required to control GPIO pins via pig  
Feb 26 17:08:59 raspberrypi systemd[1]: Started Daemon required to control GPIO pins via pig  
pi@raspberrypi:~ $
```

Tja, was soll man sagen? Läuft.

Die Anzeige beenden wir mit <CTRL> und c

Das solls nun nach jedem boot automatisch starten.

Sagen wirs ihm:

`sudo systemctl enable pigpiod`

Der Pi antwortet „Created symlink /etc/systemd/system/multi-user.target.wants/pigpiod.service → /lib/systemd/system/pigpiod.service.“ also „is recht, hab ich gemacht...“

Nun geht's ans eigentliche Telexen.

Wir brauchen die PiTelex-Programmdateien. Die liegen auf github

Um sie uns zu holen, gibt's das Programm git.

Wir haben es ganz am Anfang schon installiert. Wenn nicht, holen wir es hier nach.

`sudo apt install git`

Der sudo hat jetzt erstmal ausgedient. Wir machen nun als normaler user weiter.

`git clone https://github.com/fablab-wue/piTelex.git` --- aber HALT

Ich will testing drauf machen.

`git clone -b testing https://github.com/fablab-wue/piTelex.git`

Hier kanns passieren, dass er beim ersten mal Github nicht findet.

Einfach nochmal...

der Raspi antwortet dann brav:

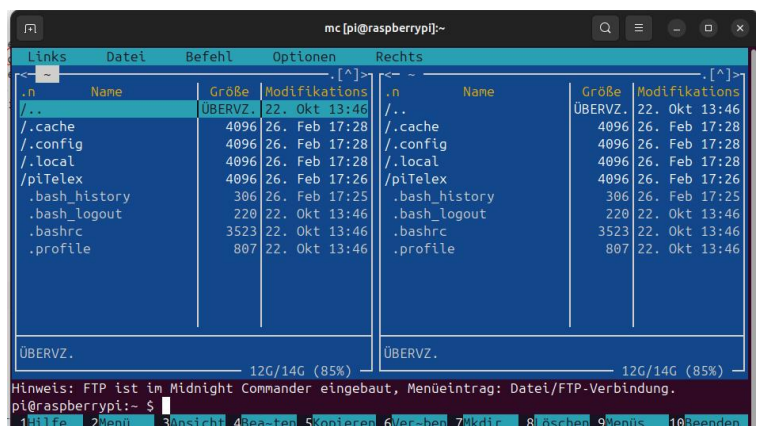
```
pi@raspberrypi:~$ git clone -b testing https://github.com/fablab-wue/piTelex.git
Klone nach 'piTelex' ...
remote: Enumerating objects: 2813, done.
remote: Counting objects: 100% (618/618), done.
remote: Compressing objects: 100% (127/127), done.
remote: Total 2813 (delta 522), reused 525 (delta 480), pack-reused 2195 (from 2)
Empfange Objekte: 100% (2813/2813), 25.72 MiB | 9.45 MiB/s, fertig.
Löse Unterschiede auf: 100% (1735/1735), fertig.
pi@raspberrypi:~$
```

Ist er nicht lieb?

Ich öffne an dieser Stelle gerne mal mc. Also geb ich einfach `mc` ein und drücke die „Enter“-Taste

Wir befinden uns weil wir der User „pi“ sind im Ordner /home/pi

Und wir sehen den Ordner /piTelex, den wir uns von github geholt haben...



rechts unten können wir nun mit der Maus den mc wieder schließen (10 Beenden)

wir wechseln den Ordner: `cd piTelex`

(für die Oldies: Das ist wie das gute alte DOS...)

mit `chmod +x telex.py` sichern wir ab, dass die Sache auch ausführbar ist

Nun kommt der erste Test.
Der GROSSE MOMENT
Wir tippen `./telex.py`

...

Die Antwort sollte nun sein:
== TELEX

```
pi@raspberrypi: ~/piTelex
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $
pi@raspberrypi:~/piTelex $ ./telex.py

== TELEX (Rev. 001d 2025-02-19 18:30) ==

<MCP:TP1><MCP:WB><piT:PULSE>
```

Wenn das soweit läuft, dann ist
erstmal alles gut.

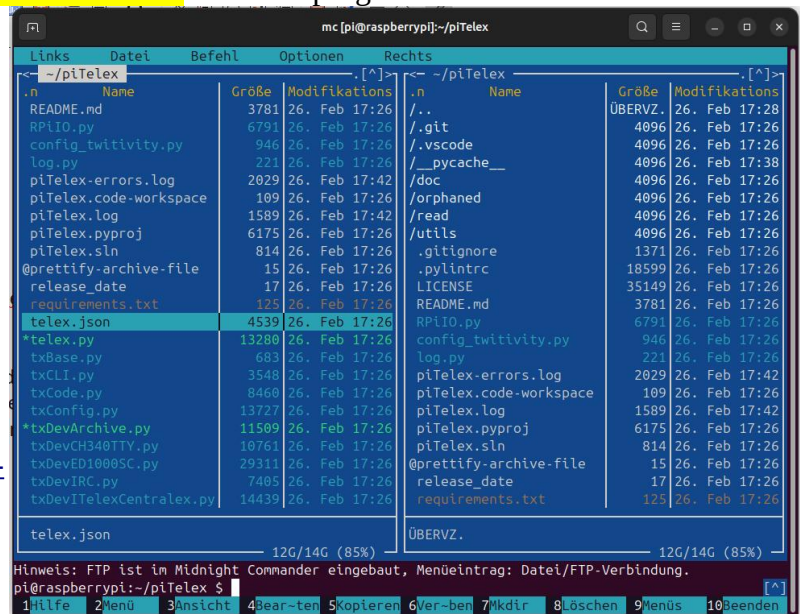
Nun kommt das Konfigurieren des Fernschreibsystems an die Reihe.
Ich stoppe mit der Tastenkombination `<CTRL>` und `c` das Telexprogramm.

Nun starte ich mc
mit den Pfeiltasten suche ich telex.json

unten ist das Schaltfeld 4 Bearbeiten...
Was macht das wohl?

Erstmal fragt es, welchen Editor ich
nutzen will. Ich machs mit dem mc-
Editor, drücke also 2

Was da genau reingeschrieben werden
muss steht in https://github.com/fablab-wue/piTelex/wiki/SW_ConfigJSON



Für mich gilt z.B.

```
# SEU-M - TW39-teletype with FSG using an  
Austrian AGT (0eAGT) with SEU-M-card as  
controller
```

```
# Note: SEU-M-card (with aRPi) is a replacement of a SEU-B ord SES-B-card
"RPiTTY_SEU-M_0e_AGT-TW39": {
  "type": "RPiTTY",
  "enable": true,
  "mode": "AGT-TW39",
  "pin_txd": 17,
  "pin_rxd": 27,
  "pin_relay": 22,
  "inv_relay": true,
  "pin_power": 9,
  "inv_power": false,
  "pin_number_switch": 10,
  "inv_number_switch": true,
  "WB_pulse_length": 60,
  "baudrate": 50,
  "coding": 0,
  "loopback": false
},
# 0=ITA2
```

Und

```
# Module type "i-Telex"

"i-Telex": {
  "type": "i-Telex",
  "enable": true,
  # "port": 2342,
```

```

"centralex": true, #True ist falsch. true wird klein geschrieben!!! WH
"centralex_srv": "tlnserv2.teleprinter.net",
"centralex_port": 49491,
"tns_dynip_number": 123456, # Subscriber number registered at TNS
"tns_pin": 12345
},

```

Man muss hier genau drauf achten, dass die Klammern stimmen.
 Ich würde einen Konfigurationsgenerator haben wollen, der das aus Ankreuzfeldern erstellt.
 Aber mit etwas Geduld kriegt man das auch hin.

Wenn `./telex.py` nun korrekt antwortet, ist es Zeit, den Autostart dafür herzustellen. Schließlich wollen wir nicht nach jedem booten uns einloggen und tausend Tasten drücken.
 Also stoppen wir `telex.py` wieder mit CTRL und c

Nun kommen noch zwei Sachen zum Installieren (aber wie das geht, wissen wir ja jetzt...):
`sudo apt install screen byobu`

Wir sind ja immernoch im Ordner `/home/pi/piTelex`
 mit `cd /home/pi/piTelex/Utils/systemd`
 wechseln wir „absolut“ ins passende Verzeichnis (wir fangen bei der root / an).
 Wir können aber auch mit `cd Utils/systemd` wechseln. Beachte: Wir setzen kein / vor `Utils`, weil wir von da aus wechseln, wo wir grad sind.
 Ist eigentlich genau wie bei Windows, hier fängts halt bei „/“ an und nicht bei „c:/“.
 Wir sind nun in `/home/pi/piTelex/Utils/systemd`
 Hier schauen wir uns (mit Hilfe von `mc`) die Datei `pitelex.service` an: User, Group und `WorkingDirectory` sind interessant.
 Hier ist wie anfangs gesagt überall „pi“ drin. Würde ich als User z.B. „karl“ gewählt haben, müsste ich User, Group und die Pfadangaben ändern auf `User=karl`, `Group=karl`, `WorkingDirectory=/home/karl/piTelex` und `ExecStart=...`

```

pi@raspberrypi: ~/piTelex/Utils/systemd
/home/pi/piTelex-/pitelex.service [----] 0 L: [ 1+ 0 1/ 20] *(0 / 389b) 0035 0x023 [*][X]
systemd service file to start piTelex

[Unit]
Description=piTelex
Requires=network.target
After=network.target
Documentation=https://github.com/fablab-wue/piTelex/wiki/

[Service]
Type=forking
User=pi
Group=pi
WorkingDirectory=/home/pi/piTelex/
ExecStart=/usr/bin/byobu-screen -d -m -S piTelex /home/pi/piTelex/telex.py
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target

```

Wie gut, dass wir zu faul waren...
 Verlassen wir wieder den `mc` und kopieren das File: `sudo cp pitelex.service /lib/systemd/system/`
 Dass er das beim booten auch starten soll, bringen wir ihm mit `sudo systemctl enable pitelex.service` bei.
 Nun booten wir mal neu... `sudo reboot -n`

wenn wir uns jetzt einloggen und einfach `byobu-screen` eingeben, empfängt uns unser frisch installiertes piTelex. Das wars. Oder?
 Es kann sein, dass der Raspberry nicht gescheit booted, wenn er auf einer SEU-M Karte steckt.
 Dann kann es helfen, mit einem USB-Netzteil als Unterstützung hochzufahren, und mit `sudo mc` den `mc` mit besonderen Rechten zu starten. Dann ganz hoch in den Ordnern. Immer „/..“ wählen bis es nimmer geht.
 Nun `/boot`, dann `/firmware` - hier suchen wir die `config.txt`. Mit F4 öffnen wir den Editor. Es kann

```

mc [root@oetelex]:/home
Links  Datei      Befehl  Optionen  Rechts
<-- /home
.n      Name          Größe  Modifikations  .n
/..     ..             ÜBERVZ. 19. Nov 14:41  /..
/pi     pi             4096    17. Mär 19:57  /pi

```

sein, dass mc fragt, welchen Editor er nehmen soll. Ich wähle hier den eingebauten mc-Editor (Nummer 2).

Mit Pfeil runter suche ich die „arm_boost=1“ und ändere die 1 zu 0.

Speichern und schließen.

Nach dem Reboot sollte es gehen. Sonst muß auf der SEU-M Karte die Spannungsversorgung geändert werden. Es empfiehlt sich, einen einstellbaren Spannungsregler zu verwenden und ihn auf 5,2 Volt einzustellen. Ich hab jetzt beim freundlichen Chinesen so ein paar Dinger bestellt...

Nochwas:

Lass das Ö-AGT nicht unnötig ohne angeschlossenen Fernschreiber laufen!

Warum? Ganz einfach: Das Gerät ist darauf ausgelegt, einen Fernschreiber als Last zu haben. Fehlt dieser, dann wird die Linienspannung über die eingebauten Lastwiderstände verbraten – und das kann richtig heiß werden!

Wie heiß genau?

Nun, wir reden hier nicht von „ein bisschen warm“, sondern von Temperaturen bis zu 160 Grad Celsius – heiß genug, um sich gewaltig die Finger zu verbrennen.

Und noch schlimmer...

Es hat schon Fälle gegeben, in denen durch den Dauerbetrieb ohne Fernschreiber der Trafo des Ö-AGT ins thermische Jenseits geschickt wurde. Der war dann so überlastet, dass er sich verabschiedet hat – und ohne Trafo läuft dann gar nichts mehr.

Was tun?

Immer einen Fernschreiber anschließen, wenn das Ö-AGT aktiv ist.

Falls der Fernschreiber mal nicht verfügbar ist, das Ö-AGT ausschalten, um Überhitzung zu vermeiden.

Wenn du nur an der Software arbeiten willst: Den Raspberry Pi kannst du problemlos ohne das Ö-AGT betreiben, indem du ihn einfach über USB mit Strom versorgst. Er darf sogar auf der SEU-M stecken und im Ö-AGT.

Ö-AGT ohne Fernschreiber laufen lassen? Richtig dumme Idee!

Die Lastwiderstände werden so heiß, dass du Spiegeleier darauf braten könntest, und im schlimmsten Fall stirbt dein Trafo den Heldentod.

Also: Immer mit Fernschreiber oder Stecker raus!

Sollten noch Fragen sein: <https://github.com/fablab-wue/piTelex/wiki> hat die komplette Doku zu piTelex, und natürlich www.telexforum.de

Und nun geh einfach zu Deinem (jetzt funktionierenden) Fernschreiber und schick einen überschwänglichen Dank an 38718 WLFHNK D