Projektbeschreibung:

http://www.ifoedit.com/RaspiCS2.html

Auf dem Raspberry läuft ein C++ Script, dass eine Märklin CS2 simuliert.

Sript lauscht ob UDP Pakete übers Wlan empfangen werden und übersetzt diese in

Can Bus Signale die von der Märklin Gleisbox verarbeitet werden.

Settings

Raspberry muss die IP Adresse 192.168.178.100 haben

PC auf dem der Rocrail Server läuft 192.168.1.144. Diese IP ist im script startscs2.sh hinterlegt.

Raspberry PI

1. Manuelles Starten der Märklin CS2 Emulation

Start Script startcs2.sh liegt im Verzeichnis /home/pi ./startcs2.sh

2. Starten der Train Controll Software in /home/pi/projects/TrainControll python server.py

Autotart via Crontab

Scripte werden per Autostart (crontab) nach dem Booten ausgeführt.

> crontab -e Job läuft unter dem Benutzer PI

@reboot python /home/pi/projects/basics/blink\_s.py &

@reboot /home/pi/startcs2.sh &

@reboot python /home/pi/projects/TrainControll/server.py &

& = Job startet im Hintergrund

crontab -e Edit crontab file, or create one if it doesn’t already exist.  
crontab -l crontab list of cronjobs , display crontab file contents.  
crontab -r Remove your crontab file.

CAN Bus Traffic anzeigen

cd /home/pi/can-utils

sudo ./candump can0

UDP Pakete auf Raspberry anzeigen

netcat -kluvw 1 192.168.178.100 1310

UDP mit Python script erzeugen

/home/pi/projects/basics/python/udp/udp\_client.sh

Traincontroller Pyth script generiert Webseite

Adresse: <http://192.168.178.100:3033/>

Source Code Verwaltung:

Python liegt auf dem Raspberry unter: /home/pi/projects/TrainControll

Github Repository

[https://github.com/MrFlexi/TrainControll](https://github.com/MrFlexi/TrainControll/tree/14798bafe9d955287ea8c48bc095c2fbb70ee9df)

Raspberry

Initiale Replikation

/home/pi/projects/Git clone <https://github.com/MrFlexi/TrainControll.git>

Änderungen abholen:

/home/pi/projects/Git pull

HTML Client mit FLASK und Socket IO

Entwicklung des Python Scripts mit PyCharm

Pakete installieren: FLASK, FLASK-SOCKETIO, eventlet

VS Code Remote SSH mit Raspberry

[https://code.visualstudio.com/docs/remote/ssh#\_getting-started](https://code.visualstudio.com/docs/remote/ssh" \l "_getting-started)

Bugfix

https://github.com/microsoft/vscode-docs/issues/3210

OLED Display

https://indibit.de/raspberry-pi-oled-display-128x64-mit-python-ansteuern-i2c/

Jetzt installieren wir noch ein paar Pakete, die wir für unser Script brauchen und starten den RPi neu:

sudo apt-get install python-smbus i2c-tools git python-pil  
sudo reboot

Test ob ob das Dislay gefunden wird

i2cdetect -y 1

Virtual Environment

source ./venv/bin/activate

pip install python-socketio==5.0.4

sudo pip install flask-socketio==4.1.1

sudo apt-get install libopenjp2-7

**Luma OLED**

Luma in einem VENV installieren

cd projects/TrainControll/

source ./venv/bin/activate

python -m pip install -U --force-reinstall luma.oled

cd luma.examples/examples

sudo python3 pi\_logo.py --interface spi --spi-device=1 --spi-port=0 --gpio-data-command 27 --gpio-reset 26 --gpio-chip-select 18 --display sh1106

sudo python3 animated\_gif.py --interface spi --spi-device=1 --spi-port=0 --gpio-data-command 27 --gpio-reset 26 --gpio-chip-select 18 --display sh1106

show version: pip3 show luma.core

Luma in einem VENV installieren

sudo pip3 install --ignore-installed luma.oled

Crontab

@reboot cd /home/pi/projects/TrainControll && venv/bin/python server.py