


RealTime (RT) Kernel

basierend auf : https://www.raspberrypi.com/documentation/computers/linux_kernel.html#building

Compilieren

- `sudo apt install git bc bison flex libssl-dev make libncurses5-dev #I think this is all the tools required`
- `mkdir kernel && cd kernel/`
- `git clone --depth=1 --branch rpi-6.12.y`
<https://github.com/raspberrypi/linux>
- `wget`
<https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/6.12/patch-6.12.39-rt11.patch.gz>
<https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/6.12/>
- `cd linux/`
- `zcat ../patch-6.12.39-rt11.patch.gz | patch -p1 --dry-run #check the patch fits`
- `zcat ../patch-6.12.39-rt11.patch.gz | patch -p1`
- `KERNEL=kernel8`
- `make ARCH=arm64 bcm2711_defconfig`
- `make ARCH=arm64 menuconfig`
 - General setup --->
 - Preemption Model (Preemptible Kernel (Low-Latency Desktop)) --->
 - [*] Fully Preemptible Kernel (Real-Time)
 - [] Preemption behaviour defined on boot
 - CPU Power Management aus !
 - Beim Exit Config Save → Yes
- `nano .config`
`CONFIG_LOCALVERSION="-V8_DrKlipper_RT"`
 -  Hier auf gar keinen Fall Leerzeichen einbauen. Gibt sonst am Ende Compile und Kopierfehler!
- `time make -j6 ARCH=arm64 Image.gz modules dtbs`
 - CPU Cores * 1,5 → `nproc`

Lokale Installation

```
echo $KERNEL sudo make modules_install sudo cp arch/arm64/boot/dts/broadcom/*.dtb /boot/ sudo cp arch/arm64/boot/dts/overlays/*.dtb* /boot/overlays/ sudo cp arch/arm64/boot/dts/overlays/README /boot/overlays/ sudo cp arch/arm64/boot/Image.gz /boot/$KERNEL.img
```

Kopier Installation

- **Der erste Part passiert auf einem Raspberry Pi 4 - Kompilierzeit ca. 2 Stunden**

- USB Stick mounten

```
sudo mkdir /mnt/copy
sudo mount -t ext4 /dev/sdb1 /mnt/copy
sudo chown -R pi:pi /mnt/copy
```

- Module kopieren

```
sudo make ARCH=arm64 INSTALL_MOD_PATH=/mnt/copy modules_install
```

- Overlays und Kernel kopieren

```
sudo mkdir -p /mnt/copy/boot/firmware/overlays
sudo cp arch/arm64/boot/Image.gz /mnt/copy/boot/firmware/$KERNEL.img
sudo cp arch/arm64/boot/dts/broadcom/*.dtb /mnt/copy/boot/firmware/
sudo cp arch/arm64/boot/dts/overlays/*.dtb*
/mnt/copy/boot/firmware/overlays/
sudo cp arch/arm64/boot/dts/overlays/README
/mnt/copy/boot/firmware/overlays/
```

- sudo umount /mnt/copy/

- **Der zweite Part dann auf dem Raspberry Pi Zero 2W**

- Libs kopieren

```
sudo cp -r /mnt/copy/lib/modules/* /lib/modules/
```

- Kernel und Overlays kopieren

```
sudo cp -n ~/usb_mount/boot/firmware/*.dtb /boot/firmware/
sudo cp -n ~/usb_mount/boot/firmware/overlays/*.dtb*
/boot/firmware/overlays/
sudo cp -n ~/usb_mount/boot/firmware/overlays/README
/boot/firmware/overlays/
sudo cp ~/usb_mount/boot/firmware/$KERNEL.img /boot/firmware/$KERNEL.img
```

''

Links

- https://www.raspberrypi.com/documentation/computers/linux_kernel.html
- Ursprung: <https://forums.raspberrypi.com/viewtopic.php?t=344994>
- <https://blog.emlid.com/raspberry-pi-real-time-kernel/>
- ISO erstellen mit RT Kernel
<https://github.com/remusmp/rpi-rt-kernel>

From:
<https://www.drklipper.de/> - **Dr. Klipper Wiki**

Permanent link:
https://www.drklipper.de/doku.php?id=projekte:sekwai:rt_kernel&rev=1753588837

Last update: **2025/07/27 06:00**

