

Ik ga in dit bestand uit van een unix-achtig systeem. Ik heb zelf gewerkt vanuit MacOS, maar alle commandos moeten ook vanuit Linux werken.

Voor deze opdracht heb ik gebruik gemaakt van de [rpi-rt-kernel](#) repository. Deze repository heb ik geforkt omdat er een probleem was met de usb driver. De patch hiervoor heb ik gekopieerd van [deze commit](#). Ik in de [Dockerfile](#) aanpassingen gemaakt zodat de patch wordt doorgevoerd voor zowel kernel versie 5 als 6. De diff van de fork is [hier](#) te zien. Daarnaast heb ik een nieuw argument toegevoegd, **NO\_RT** zodat ik met dezelfde Dockerfile ook een standaard versie van de kernel kan bouwen. Waar de versie kan in de Dockerfile worden aangepast wordt getoond in de volgende afbeelding, maar ik heb voor deze test v6.6 gebruikt.

```
Dockerfile M X
Dockerfile > ...
1 # set PLATFORM32 to something not null if you're building for older platforms, i.e. Pi 1, Pi Zero, Pi Zero W and Pi One
2 # do not define PLATFORM32 or set it to null if you're building for newer platforms, i.e. Pi 3, Pi 3+, Pi 4, Pi 400, Pi Zero 2
3 FROM ubuntu:latest
4
5 # Timezone, You may change if you want
6 ENV TZ=Europe/Amsterdam
7 RUN ln -sfn /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ > /etc/timezone
8
9 # Installing dependencies
10 RUN apt update
11 RUN apt upgrade -y
12 RUN apt install -y git make gcc bison flex libssl-dev bc ncurses-dev kmod
13 RUN apt install -y crossbuild-essential-arm64 crossbuild-essential-armhf
14 RUN apt install -y wget zip unzip fdisk nano curl xz-utils
15
16 # Linux version. Only works if the usb driver is the same as in the patch. Will check.
17 ENV LINUX_KERNEL_VERSION=6.6
18 ENV LINUX_KERNEL_BRANCH=rpi-${LINUX_KERNEL_VERSION}.y
19
```

De keuze of de realtime versie van de kernel gebruikt wordt of niet kan simpelweg gedaan worden met de commandos: **export NO\_RT=1** of **unset NO\_RT**. Wanneer **NO\_RT** gedefinieerd is dan wordt de real-time patch en de usb-driver patch niet uitgevoerd, en blijven de kernel configuraties op default. Dit wordt getoond in de volgende afbeelding.

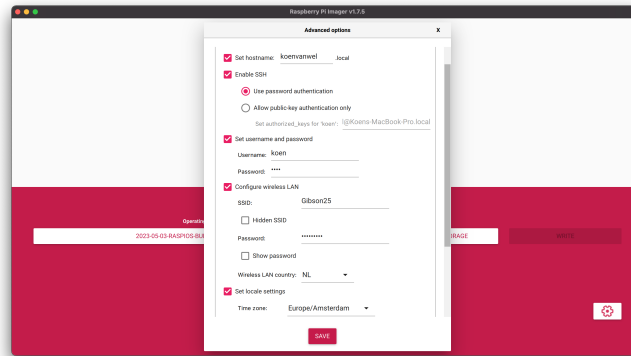
```
28 ARG NO_RT
29 # Download and apply RT patch (unless NO_RT is defined)
30 RUN [ -z "$NO_RT" ] && echo "Downloading patch $(PATCH)" || true
31 RUN [ -z "$NO_RT" ] && curl https://mirrors.edge.kernel.org/pub/linux/kernel/projects/rt/${LINUX_KERNEL_VERSION}/${PATCH}.gz
32 RUN [ -z "$NO_RT" ] && gzip -cd /rpi-kernel/linux/${PATCH}.patch.gz | patch -p1 --verbose || true
33
34 # Apply USB patch (unless NO_RT is defined)
35 ENV USB_DRIVER_SOURCE=usb_driver.patch
36 ENV USB_DRIVER_PATCH=$(USB_DRIVER_SOURCE)/patch
37 ENV USB_DRIVER_CHECK=$(USB_DRIVER_SOURCE)/check
38 ENV USB_DRIVER_TARGET=/rpi-kernel/linux/drivers/usb/host/dwc_otg/
39
40 COPY $(USB_DRIVER_SOURCE) /$(USB_DRIVER_SOURCE)
41 RUN [ -z "$NO_RT" ] && echo "Applying USB patch. Open /usb_diff.patch to see the differences.." || true
42 RUN [ -z "$NO_RT" ] && bash /$(USB_DRIVER_SOURCE)/patch.sh || true
```

Om de kernel te installeren moet docker op de computer geïnstalleerd staan. De installatie instructies zijn op de [site van Docker](#) te vinden. Om de kernel te installeren kan het volgende commando uitgevoerd worden:

```
git clone git@github.com:koenichiwa/rpi-rt-kernel.git && cd "$(basename "$_"
.git) && make Pi3
```

Ik heb enige aanpassingen gedaan in de Makefile, om de juiste argumenten aan de Docker build mee te geven. Wanneer de build is afgelopen komt er een zip bestand in de build directory te staan. Wanneer deze wordt ge-unzippt kan de gegenereerde img gebruikt worden om op de SD-kaart te installeren. De installatie heb ik zelf gedaan met [Raspberry Imager](#), maar het is ook mogelijk om **dd** te gebruiken.

Raspberry Imager heeft de optie om direct de hostname, username, ssh en wifi instellingen te doen. Al heb ik dit uiteindelijk handmatig gedaan.

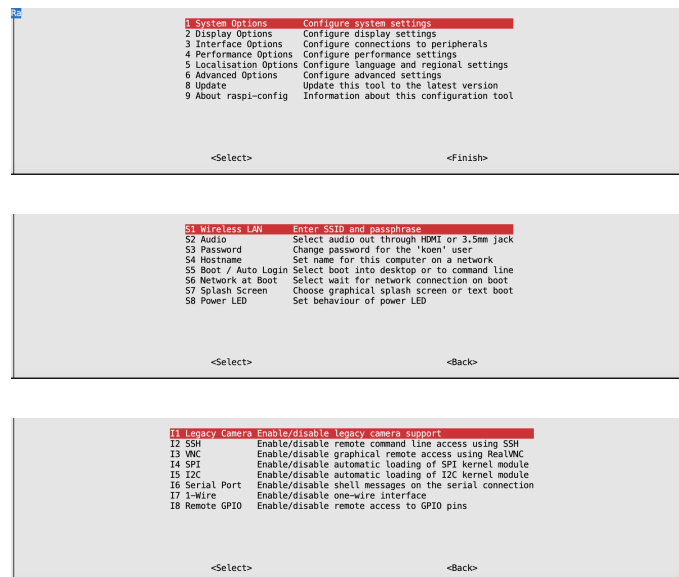


Gelukkig had ik zelf een scherm waarvan ik gebruik kon maken om de eerste instellingen op de RPI te doen. Eerst heb ik de libraries en binaries geïnstalleerd die ik nodig had voor het project:

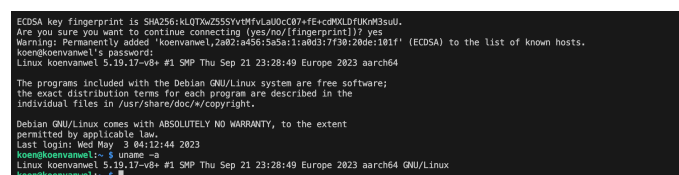
```
sudo apt update && sudo apt upgrade -y && sudo apt install git pigpio libnuma-dev gnuplot -y
```

Vervolgens heb ik de configuratie aangepast met het volgende commando.

```
sudo raspi-config
```



Dit was enkel om de wifi en hostname in te stellen en de ssh aan te zetten. Hierna heb ik op mijn eigen PC geconnect met de RPI met `ssh koen@koenvanwel` en, omdat het een gedeelte van de opdracht was, heb ik getest of `uname -a` ook mijn naam toonde. De rest van de commandos zullen op de RPI worden uitgevoerd.



Daarna moest had ik een ssh key nodig om in de git repositories te komen. De output hiervan heb ik aan mijn github keychain toegevoegd. `ssh-keygen -t rsa -f ~/.ssh/id_rsa -N "" -q && cat ~/.ssh/id_rsa.pub`

Vervolgens moest ik definiëren wat de output directory was voor de komende tests. `export OUTPUT_DIR=NO_RT6`

En ik heb mijn eigen test gecloned. De code en de uitkomst zijn te vinden in [mijn repository](#) `cd ~/ && git clone git@github.com:koenichiwa/rpi_pins_c.git && cd "$(basename "$_" .git)" && sudo make run`

```
koen@koenvanmel:~$ git clone git@github.com:koenichiwa/rpi_pins_c.git && cd "$(basename "$_" .git)" && sudo make run TEST_OUTPUT=NO_RT6
Cloning into 'rpi_pins_c'...
remote: Enumerating objects: 434, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 434 (delta 0), reused 2 (delta 0), pack-reused 430
Receiving objects: 100% (434/434), 47.85 MiB | 2.44 MiB/s, done.
Resolving deltas: 100% (9/9), done.
Updating files: 100% (204/204), done.
gcc -Wall -pthread -O3 -o build/test_delay src/main.c -lpigpio -lrt
././build/test_delay NO_RT6
Set mode.
Test 0.
Set interrupt.
Spawn thread.
```

Toen deze test klaar was heb ik versie 1.10 van `rt-tests` gebruikt om de cyclicttest te draaien. Dit heb ik gebouwd met het volgende commando:

```
cd ~/ && git clone https://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git &&
cd "$(basename "$_" .git)" && git fetch --all --tags && git checkout tags/v1.10
-b 1.10-branch && make
```

```
koen@koenvanmel:~$ git clone https://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git && cd "$(basename "$_" .git)" && git fetch --all --tags && git checkout tags/v1.10 -b 1.10-branch && make
Cloning into 'rt-tests'...
remote: Enumerating objects: 5179, done.
remote: Total 5179 (delta 0), reused 0 (delta 0), pack-reused 5179
Receiving objects: 100% (5179/5179), 1.49 MiB | 2.60 MiB/s, done.
Resolving deltas: 100% (3330/3330), done.
Fetching origin
Switched to a new branch '1.10-branch'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'distutils.sysconfig'
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'distutils.sysconfig'
gcc -D VERSION=1.10 -c src/cyclicttest/cyclicttest.c -Wall -Wno-nonnull -O2 -g -D GNU_SOURCE -Isrc/include -o bld/cyclicttest.o
```

En uitgevoerd met de volgende commandos:

```
~/rpi_pins_c/$OUTPUT_DIR/cyclic_test_output
```

```
sudo ./cyclicttest -l1000000000 -m -S -p90 -i200 -h400 -q | cat >
```

```
~/rpi_pins_c/$OUTPUT_DIR/cyclicttest_output
```

Ten slotte heb ik `create_histogram.sh` in mijn `rpi_pins_c` repository gebruikt om van de data een histogram te maken. De code ervan heb ik gebaseerd op [deze post van OSADL](#)