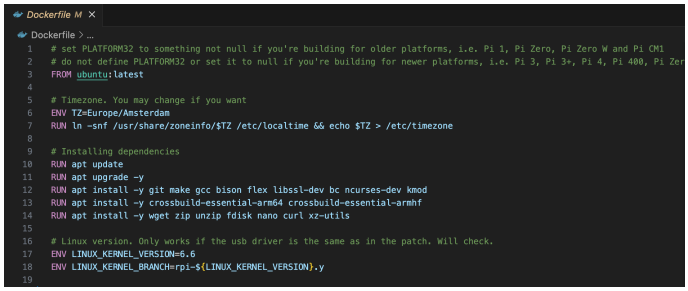
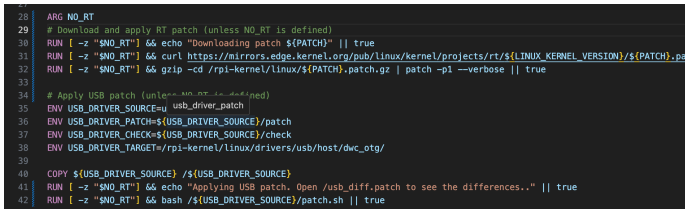
Ik ga in dit bestand uit van een unix-achtig systeem. Ik heb zelf gewerkt vanuit MacOS, maar alle commandos moeten ook vanuit Linux werken.

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.



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.

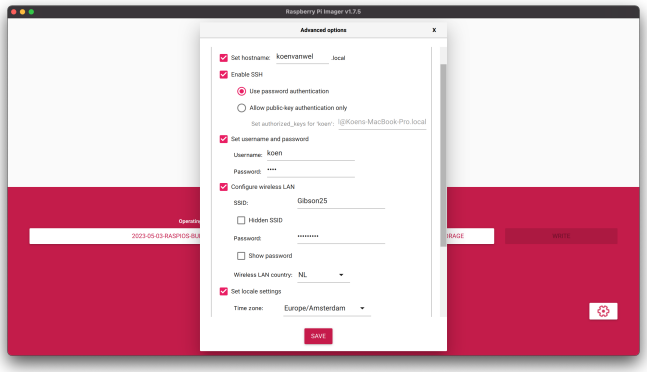


Om de kernel te installeren moet docker op de computer geinstalleerd staan. De installatie instructies zijn op de site van Docker te vinden. Om de kernel te installeren can het volgende commando uitgevoerd worden:

```
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-unzipt 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.

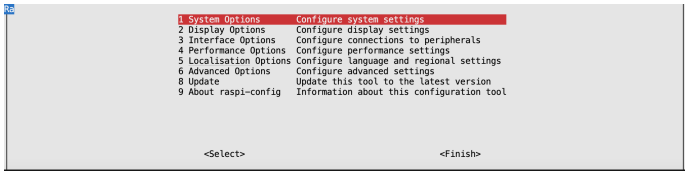


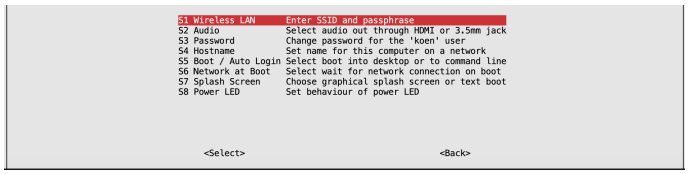
Gelukkig had ik zelf een scherm waarvan ik gebruik kon maken om de eerste instellingen op de RPI te doen. Eerst heb ik de libraries en binaries geinstalleerd die ik nodig had voor het project:

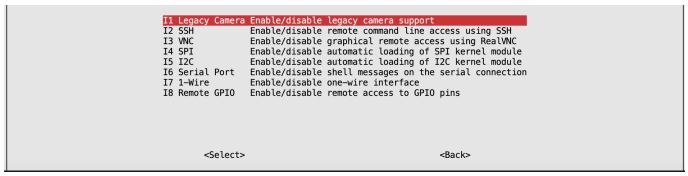
```
sudo apt update && sudo apt upgrade -y && sudo apt install git pigpio libnuma-dev gnuplot -y
```

Vervolgens heb ik de configuratie aangepast met het volgende commando.

```
sudo raspi-config
```





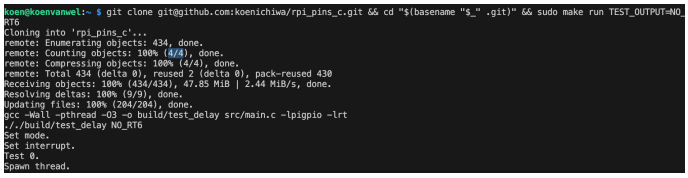


Dit was enkel om de wifi en hostname in te stellen en de ssh aan te zetten. Hierna heb ik op mijn eigen PC geconnect met de RPI met `ssh koen@koenvanwel` en, omdat het een gedeelte van de opdracht was, heb ik getest of `uname -a` ook mijn naam toonde. De rest van de commandos zullen op de RPI worden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

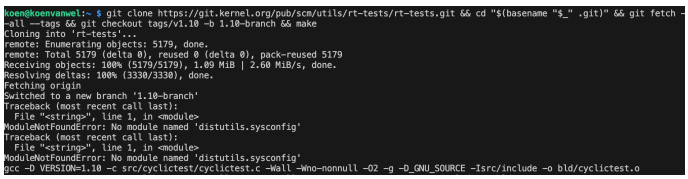
Vervolgens moest ik definieren wat de output directory was voor de komende tests. `export OUTPUT_DIR=NO_RT6`

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`



Toen deze test klaar was heb ikversie 1.10 van `rt-tests` gebruikt om de cyclictest 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`



En uitgevoerd met de volgende commandos:

`~/rpi_pins_c/$OUTPUT_DIR/cyclic_test_output`

`sudo ./cyclictest -l100000000 -m -S -p90 -i200 -h400 -q | cat > ~/rpi_pins_c/$OUTPUT_DIR/cyclictest_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