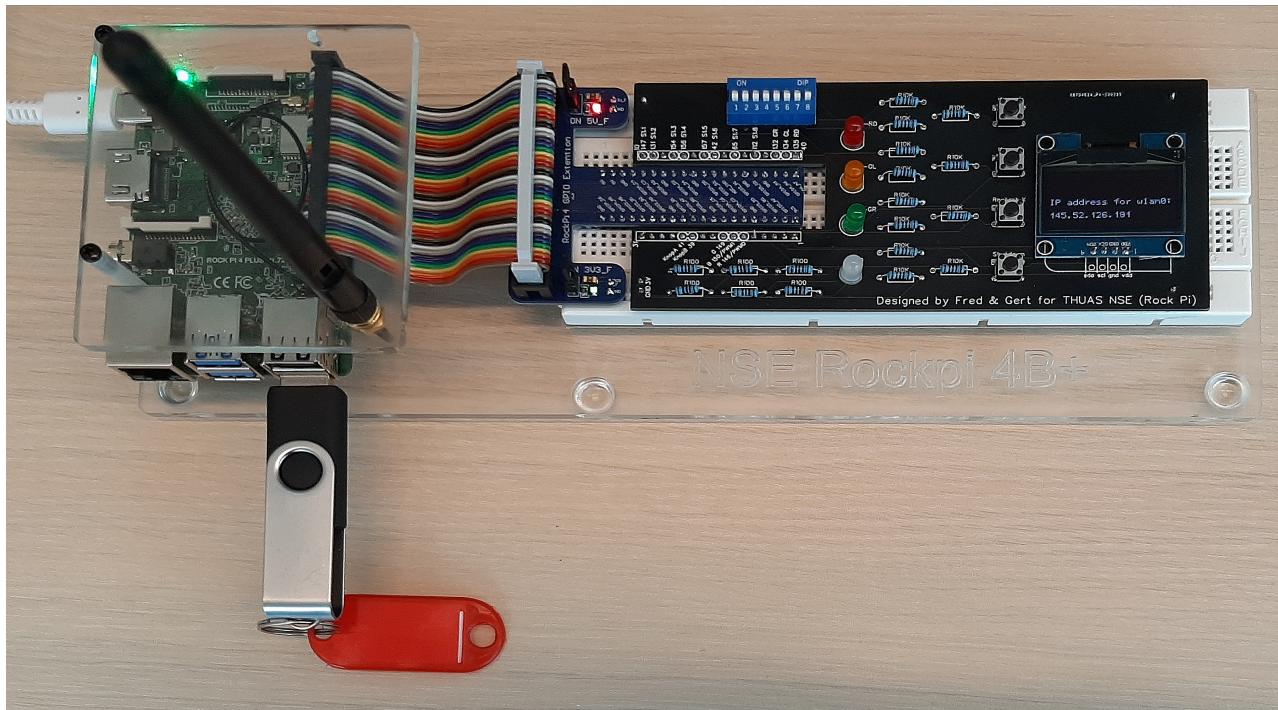


# Object Georiënteerd Programmeren

## ( Introductie )

(Practicumhandleiding inclusief de practicumopstelling voor de RockPi)

Versie 0.5



# Inhoudsopgave

<b>1 Inleiding</b>	<b>3</b>
1.1 Voorbereiding - Software installeren . . . . .	3
1.2 Eerste gebruik van de RockPi . . . . .	4
1.3 Inloggen met VNC . . . . .	5
1.4 USB Stick voorbereiden . . . . .	5
<b>2 Klasse en objecten in C++</b>	<b>8</b>
2.1 Werken met de RockPi . . . . .	8
2.1.1 Verbinding opzetten met de RockPi . . . . .	8
2.1.2 Het aansturen van een LED . . . . .	10
2.2 Het werken vanuit Visual Studio Code . . . . .	12
2.3 Het werken met een visuele debugger . . . . .	15
<b>Appendices</b>	<b>18</b>
<b>A Tips&amp;Tricks voor Linux omgeving</b>	<b>20</b>
A.1 Tips . . . . .	20
A.2 Linux commando's (zie ook tips in A.1) . . . . .	21
A.3 GPIO poorten in Linux . . . . .	22
<b>B Installeren van de practicum omgeving</b>	<b>23</b>
B.1 De VNC viewer . . . . .	23
B.2 Het gebruik van 'Visual Studio Code' . . . . .	23

# 1 Inleiding

Voor het practicum OOPR1 zijn (ongeveer) 25 RockPi's beschikbaar. Dit zijn Raspberry Pi achtige ‘Single Board Computers’ die met een Linux variant werken, in dit geval Debian 10. De RockPi's zijn aangeschaft omdat Raspberry Pi's niet leverbaar (of te duur) waren. Tijdens het practicum gebruik je een RockPi van school. Je kunt deze alleen op school gebruiken, je mag hem niet meenemen naar huis. Dat is dan ook het belangrijkste nadeel.

Het gebruik van de RockPi heeft een aantal voordelen:

- Je hoeft zelf niets aan te schaffen
- Je werkt onder gecontroleerde omstandigheden: de verstrekte RockPi bevat alles wat nodig is voor het practicum. Dat betekent dat jij- of de docent- niet eerst moet puzzelen om de omgeving aan de praat te krijgen.

Bij het practicum heb je *je eigen* USB stick nodig om je bestanden op te zetten (*geen gedeelde!*). Een USB stick van 1GB is goed genoeg (128+ MB). Groter is zonde.

- Je werkt op je eigen **USB stick** en niet op het bestandssysteem van de RockPi.
- Na afloop van het practicum lever je de RockPi in zonder daar bestanden op achter te laten.
- Ga er vanuit dat de RockPi's na een practicum gewist worden!

## 1.1 Voorbereiding - Software installeren

Zorg dat je vóór het practicum de volgende software geïnstalleerd hebt:

- Installeer Github Desktop: <https://desktop.github.com/>
- Vanuit Github Desktop, druk Ctrl+Shift+O voor ‘Clone repository’ en voeg de repository **JohnVi-hhs/oop** toe.
- Installeer ‘Visual Studio Code’ (VSC) volgens de instructies in <https://github.com/Grrtzm/OOPR1> (*moet nog aangepast worden*)
- Download en installeer de VNC viewer:  
<https://www.realVNC.com/en/connect/download/combined/>
- (Optioneel maar wel handig) Download en installeer de SSH client KiTTY  
<https://www.fosshub.com/KiTTY.html>  
KiTTY is een opvolger van PuTTY. Het grote voordeel is dat KiTTY automatisch opnieuw verbinding maakt als de verbinding even verbroken is geweest.

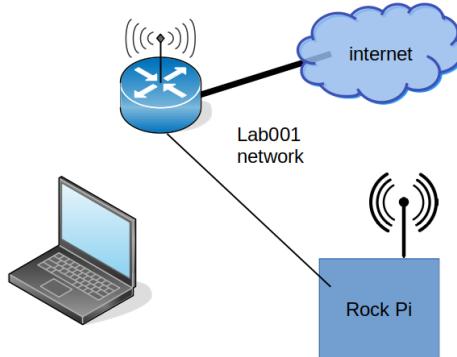
## 1.2 Eerste gebruik van de RockPi

We gaan er van uit dat je de RockPi in lokaal D2.001 of D2.003 van HHS Delft gebruikt.

De RockPi is al ingesteld voor Wi-Fi netwerk in D2.001: **Lab001**.

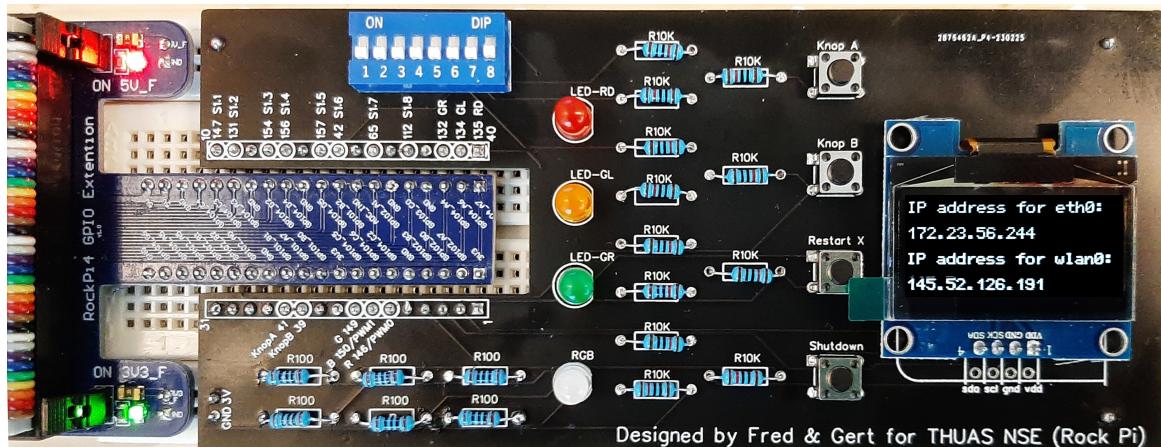
Log ook met je laptop in op dit netwerk. Het password is **Lab001WiFi**.

In Figuur 1.1 wordt weergegeven hoe de RockPi op het lab netwerk is aangesloten.



**Figuur 1.1:** De RockPi in het lab netwerk

Sluit de USB-C voedingskabel aan zodat de RockPi gaat opstarten (op de RockPi gaan een groene led branden). Verder hoef je nog niets aan te sluiten!



**Figuur 1.2:** Het uitbreidingsbord van de RockPi

**Controleer de zelftest:** Om zeker te weten dat het bovenstaande bordje OK is gaan de RockPi alle leds even aan en uitzetten.

Als je de RockPi in het lab (D2.001) aanzet, verschijnt op het oled display bij "IP address for wlan0:" een IP adres van het Lab001 Wi-Fi netwerk (zie Figuur 1.2).

Nu kun je via SSH en/of VNC een verbinding maken met dit IP adres en inloggen op de RockPi. De username is **rock**, het password is ook **rock**. Mocht er een sudo password gevraagd worden, dan is dit ook **rock**. Overigens kun je ook een ethernet kabel aansluiten zoals in Figuur 1.2 te zien is bij "IP address for eth0:".

Zorg wel dat je PC op hetzelfde netwerk is aangesloten. Als VNC het niet wil doen, maar ssh wel, dan zitten je PC en de RockPi mogelijk op verschillende Wi-Fi accesspoints. Als VNC niet werkt, maar je RockPi en laptop zitten wel op hetzelfde netwerk,

dan kun je het knopje '**Restart\_X**' indrukken, dit herstart de grafische schil op de RockPi.

Zodra je bent ingelogd met VNC, kun je je USB stick aansluiten en configureren. Het maakt niet uit welke USB poort je gebruikt.

Als je klaar bent met je practicum, moet je de RockPi netjes afsluiten. Door op het knopje '**Shutdown**' te drukken, wordt het operating system netjes afgesloten zodat het bestandssysteem niet corrupt raakt (wat kan gebeuren als je de RockPi gewoon uitzet).

'Knop\_A' kun je gebruiken om tijdens het opstarten voor volledig GPIO gebruik te kiezen; druk de knop in vóórdat je de RockPi aanzet, en houd hem ingedrukt tot de LED zelftest klaar is. Dit is zo gedaan omdat wisselen tussen PWM gebruik (voor de rode en blauwe leds van de driekleurige led) en GPIO (alleen aan en uit, geen 'analoge' waarden) niet lukt zonder opnieuw op te starten.

Aan 'Knop\_A', 'Knop\_B' en de DIP switches (blauwe blokje) zijn verder geen functies toegewezen. Daar kun je zelf code voor schrijven.

## 1.3 Inloggen met VNC

Voer het IP adres zoals getoond op het oled display in op VNC (Figuur 1.2). Log in op de RockPi en klik op het 'Terminal' icoon (rood omcirkeld in Figuur 1.3):



Figuur 1.3: Terminal icoon

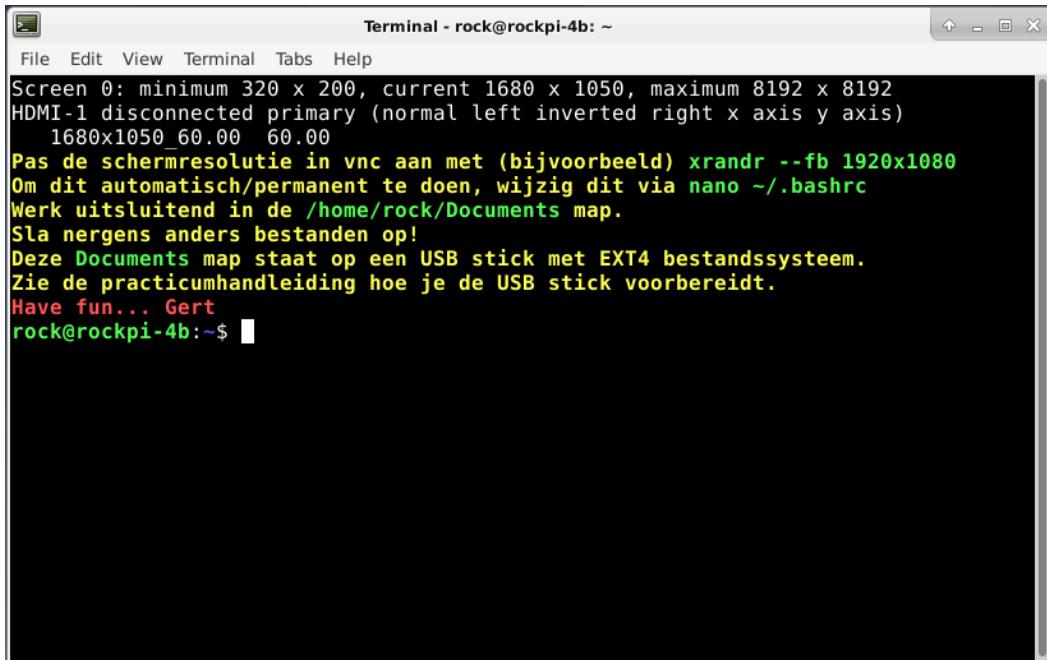
Nu verschijnt het Terminal venster. Van hieruit kun je allerlei commando's invoeren:

## 1.4 USB Stick voorbereiden

### LET OP!!

- Met onderstaande procedure verwijder je alle bestanden van je USB stick!
- Je hebt genoeg aan een [kleine USB stick](#) (ergens tussen 128MB en 4 GB).
- Je kunt hierna alléén met Linux op je USB stick kijken, niet meer met Windows!

Formatteer nu je USB stick met EXT4 filesystem. Dit is nodig omdat Linux d.m.v. attributen in het bestandssysteem aangeeft of een bestand uitvoerbaar ('executable') is. Dat is op een Windows compatible USB stick (met FAT of NTFS bestandsysteem) niet mogelijk. Kijk of je USB stick gezien wordt. Geef het commando **lsblk** en kijk of daar een device bij zit met sda in de naam (zie hieronder in de listing van lsblk). In dit geval is deze er. De partitie waar we gebruik van maken is /dev/sda1.



**Figuur 1.4:** Het Terminal scherm

```
rock@rockpi-4b:~$ lsblk
NAME      MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
sda        8:0    1    1.9G  0 disk
`-sda1     8:1    1    1.9G  0 part /media/rock/9C02-C2F1
mmcblk1   179:0   0 115.2G  0 disk
|-mmcblk1p1 179:1   0    3.9M  0 part
|-mmcblk1p2 179:2   0     4M  0 part
|-mmcblk1p3 179:3   0     4M  0 part
|-mmcblk1p4 179:4   0   512M  0 part
`-mmcblk1p5 179:5   0    7.8G  0 part /
mmcblk1boot0 179:32  0     4M  1 disk
mmcblk1boot1 179:64  0     4M  1 disk
mmcblk1rpmb 179:96  0     4M  0 disk
}
```

### Controle:

Als je het commando **mount** geeft, dan verwacht je in de uitvoer het device (de disk) /dev/sda1 te zien:

```
/dev/sda1 on /media/rock/9C02-C2F1 type vfat (rw,nosuid,nodev,relatime,
    uid=1000,gid=1000,fmask=0022,dmask=0022,codepage=936,iocharset=utf8,
    shortname=mixed,showexec,utf8,flush,errors=remount-ro,uhelper=udisks2
)
```

Hierboven zie je dat disk ge-'mount' is. Om te formatteren moet je eerst 'umount' doen:

**umount /dev/sda**

Nu USB disk formatteren met ext4 filesystem:

**sudo mkfs -t ext4 /dev/sda**

Vervolgens disklabel 'Documents' instellen:

**sudo e2label /dev/sda Documents**

Hieronder zie je de output van bovenstaande commando's:

```
umount: /dev/sda#: No such file or directory
rock@rockpi-4b:~\$ sudo mkfs -t ext4 /dev/sda
mke2fs 1.44.5 (15-Dec-2018)
/dev/sda contains a vfat file system
Proceed anyway? (y,N) y
Creating filesystem with 491520 4k blocks and 123120 inodes
Filesystem UUID: 1b2275ec-6dbd-4f3b-8d41-feaf4da0c7b7
Superblock backups stored on blocks:
32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

Als het formatteren en labelen klaar is, kun je testen of de USB stick correct gezien wordt:

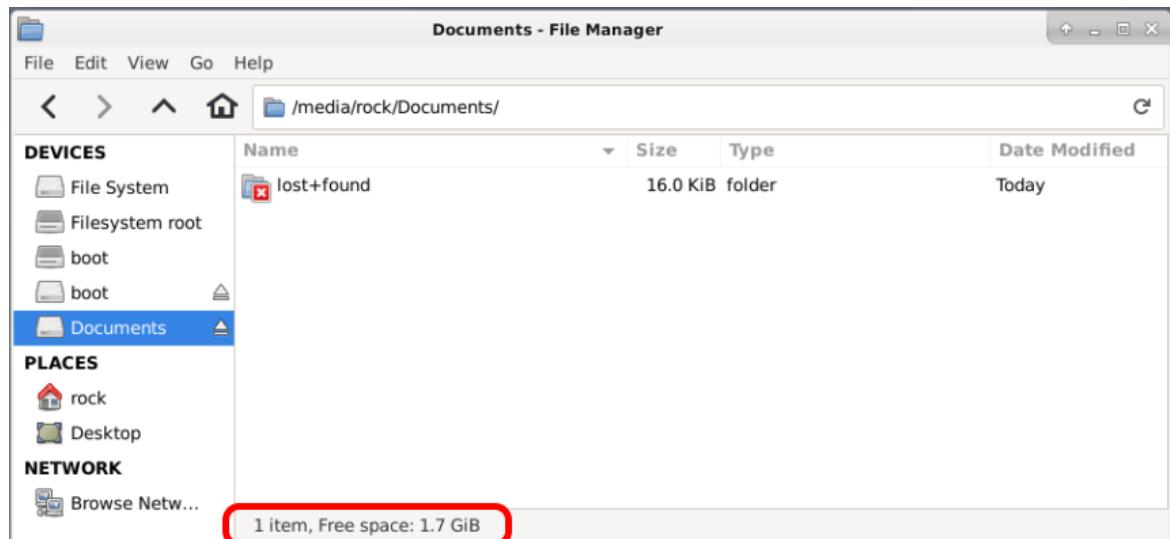
Verwijder de USB stick, wacht even en steek hem weer terug in een USB poort.  
Klik in VNC op het 'File Manager' icoon (rood omcirkeld in Figuur 1.5):



**Figuur 1.5:** Terminal icoon

Ga in de 'File Manager' naar het 'Documents' device.

Als dit werkt, dan zie je onderin de statusbalk van de 'File Manager' hoeveel vrije ruimte je USB stick heeft (rood omcirkeld in Figuur 1.5)



**Figuur 1.6:** File Manager

Met de 'File Manager' kun je in het bestandssysteem van Linux kijken.

Bij het practicum werk je alleen maar in de home directory en subdirectories van gebruiker **rock** (/home/rock).

Als je meer wilt weten over het Linux bestandssysteem, [klik dan hier](#).

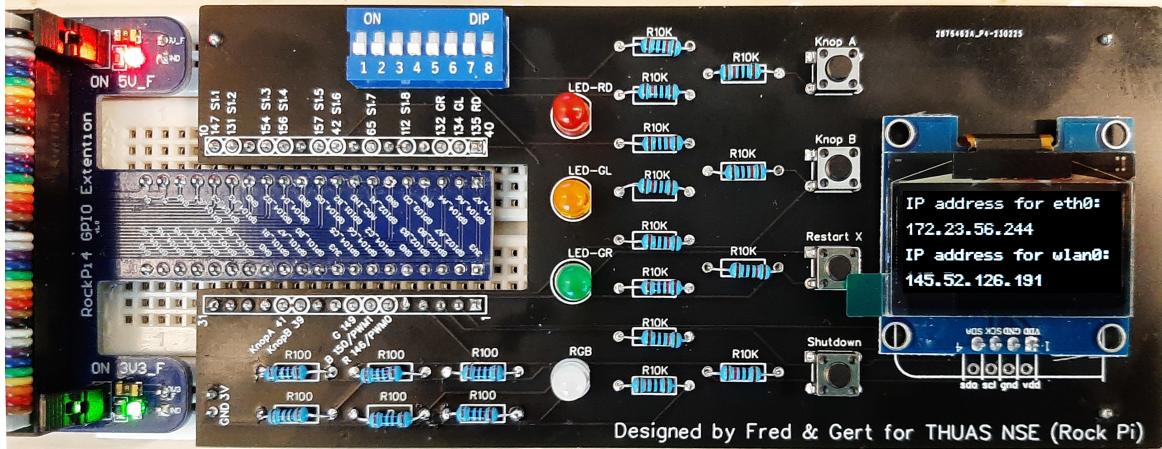
## 2 Klasse en objecten in C++

Als eerste wordt er verbinding gemaakt vanaf je laptop met de RockPi, waarna een LEDje aan- en uitgezet wordt. Vervolgens wordt door middel van een programma een aantal LEDs aangestuurd.

### 2.1 Werken met de RockPi.

#### 2.1.1 Verbinding opzetten met de RockPi

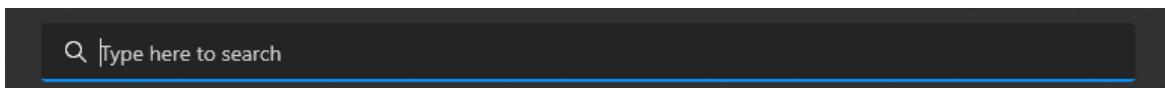
In eerste instantie wordt er verbinding gemaakt met de RockPi via het *ssh* commando. Het IP adres van de RockPi is af te lezen via het oled display zoals in Figuur 2.1 wordt



Figuur 2.1: De RockPi met het IP adres.

weergegeven. In het voorbeeld hierna wordt het IP adres *192.168.178.89* gebruikt.

Open in Windows een terminal (b.v. een PowerShell, cmd prompt of een extern programma zoals b.v. [KiTTY](#)). We gebruiken voor nu PowerShell: Ga naar Windows Search: druk Windows+S (de knop met het Windows vlaggetje + de 'S'), zie Figuur 2.2 en tik vervolgens in: *powershell*. In de PowerShell kan het *ssh* commando met de



Figuur 2.2: Zoekscherf in Windows

username en het IP adres gegeven worden, zoals in figuur 2.3 wordt weergegeven. Bij de eerste keer zal de melding komen of de host key moet worden opgeslagen, kies *yes*. Vervolgens zal om om het password gevraagd worden. Dit is *rock*.

Wanneer het inloggen gelukt is, verschijnt een verhaaltje dat eindigt met de prompt van de RockPi, zoals in figuur 2.3 te zien is. Vanaf nu werk je in een Linux terminal



```
rock@rockpi-4b: ~
PowerShell 7.3.3
PS C:\Program Files\PowerShell\7> ssh rock@192.168.178.89
The authenticity of host '192.168.178.89 (192.168.178.89)' can't be established.
ECDSA key fingerprint is SHA256:MgoCsCAA5Mb8qUwGolYLqLLTP+0gDpXzWhfaUveSATg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.178.89' (ECDSA) to the list of known hosts.
rock@192.168.178.89's password:
Linux rockpi-4b 4.4.154-116-rockchip-g86a614bc15b3 #1 SMP Mon Jan 10 12:03:08 UTC 2022 aarch64

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.

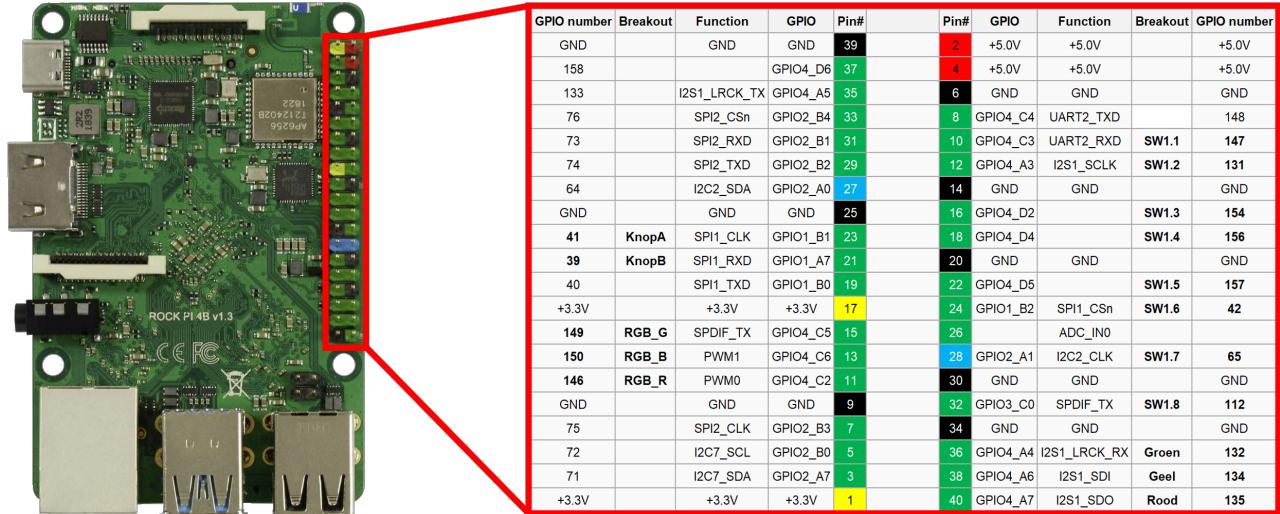
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Mar 13 11:28:39 2023 from 192.168.178.46
Pas de schermresolutie in vnc aan met (bijvoorbeeld) xrandr --fb 1920x1080
Om dit automatisch/permanent te doen, wijzig dit via nano ~/.bashrc
Werk uitsluitend in de /home/rock/Documents map.
Sla nergens anders bestanden op!
Deze Documents map staat op een USB stick met EXT4 bestandssysteem.
Zie de practicumhandleiding hoe je de USB stick voorbereidt.
Have fun... Gert
rock@rockpi-4b:~$ -
```

Figuur 2.3: Ingelogd in de RockPi

omgeving. Een listing van een directory kan opgevraagd worden met het *ls* commando, het veranderen van een directory kan gedaan worden het *cd* commando (change directory) en het maken van een directory met het *mkdir* commando (make directory). Verdere Linux commando's en tips zijn te vinden in [de Bijlage](#).

## 2.1.2 Het aansturen van een LED.

Wanneer contact is gemaakt met de RockPi, zoals beschreven in hoofdstuk 2.1.1, kunnen de LEDs aangestuurd worden. De LED's zitten aangesloten op de GPIO ('General Purpose Input Output') connector van de RockPI, deze is te zien in Figuur 2.4. De buitenste kolom geeft het GPIO nummer aan waarop deze geprogrammeerd



Figuur 2.4: GPIO connector van de RockPI

kan worden. Zo zit de rode LED aangesloten op GPIO nummer 135. De **dikgedrukte** pinnen worden gebruikt met de printplaat. In de kolom **breakout** zie je wat op die pinnen zit.

### Opdracht 1a, het eerste programmaatje op de RockPI

1. Plaats de [USB stick](#) in de RockPi.
2. Ga naar de directory van je USB stick
  - `cd Documents` (het hele pad is `/media/rock/Documents` )
3. Maak een directory `ledje` aan en ga daar na toe.
  - `mkdir ledje`
  - `cd ledje`
4. Download files `gpiofuncties.h`, `gpiofuncties.cpp`, `test.cpp` van Github. Het main programma (`test.cpp`) wordt in Listing 2.1 weergegeven
  - `git clone https://github.com/JohnVi-hhs/oop.git`
  - `cd oop`
  - `ls -l`
  - `nano test.cpp` Druk Ctrl+X om nano weer te sluiten.

```

rock@rockpi-4b:~/Documents$ cd Documents
rock@rockpi-4b:~/Documents$ mkdir ledje
rock@rockpi-4b:~/Documents$ cd ledje
rock@rockpi-4b:~/Documents/ledje$ git clone https://github.com/JohnVi-hhs/oop.git
Cloning into 'oop'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (46/46), done.
remote: Total 57 (delta 18), reused 29 (delta 6), pack-reused 0
Unpacking objects: 100% (57/57), done.
rock@rockpi-4b:~/Documents/ledje$ cd oop
rock@rockpi-4b:~/Documents/ledje/oop$ ls -l
total 16
-rw-r--r-- 1 rock rock 258 Mar 16 21:25 README.md
-rw-r--r-- 1 rock rock 1119 Mar 16 21:25 gpiofuncties.cpp
-rw-r--r-- 1 rock rock 91 Mar 16 21:25 gpiofuncties.h
-rw-r--r-- 1 rock rock 530 Mar 16 21:25 test.cpp
rock@rockpi-4b:~/Documents/ledje/oop$ nano test.cpp

```

**Figuur 2.5:** Schermuitvoer na bovenstaande opdrachten

```

#include <unistd.h>
#include <iostream>
#include "gpiofuncties.h"

using namespace std;
#define RODELED 135

int main() {

    cout<<"Hi_NSE"<<endl;
    int b=zetPinOpOutput(RODELED); //return waarde of het gelukt is.
    if(b == 0) { //if(!b) mag ook.
        cout<<"Fout je bedankt"<<endl;
        return 0;
    }
    cout<<"b=_ "<<b<<endl; //return waarde of het gelukt is.
    b=zetPinWaarde(RODELED,1); //Zet de rode LED aan.
    usleep(1000000);
    b=zetPinWaarde(RODELED,0); //Zet de rode LED uit.
    cout<<"einde"<<endl;
}

```

**Listing 2.1:** Zet LED aan en uit

## 5. Compileer en run het testprogramma.

- Compileren van het programma.

*g++ -g3 \*.cpp -o tst*

optie *-g3* heeft te maken met debug mogelijkheden.

Optie *-o* geeft een naam aan de output file (*tst*)

- Voer het zojuist gecompileerde programma *tst* uit.

*./tst*

Resultaat:

De rode LED gaat 1 seconde aan.

- Pas het programma zodanig aan, zodat eerst de groene LED aangaat, daarna de gele LED en vervolgens de rode LED.

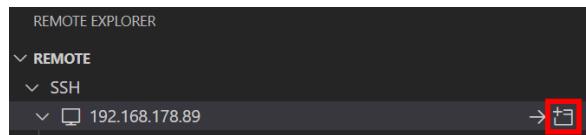
In de linux terminal, kunnen verschillende editors gebruikt worden, waarvan **nano** één van de meest gebruikte is, een ander beroemde/beruchte editor is **vim**. Als je nano gebruikt: druk Ctrl+O om op te slaan en Ctrl+X om af te sluiten. Bij vim, druk/tik **:help**

## 2.2 Het werken vanuit Visual Studio Code

Hierbij wordt vanuit 'Visual Studio Code' (VSC) gewerkt, en wordt de klasse Led geïmplementeerd. Er wordt vanuit gegaan dat VSC geïnstalleerd is zoals in appendix B.2 beschreven staat.

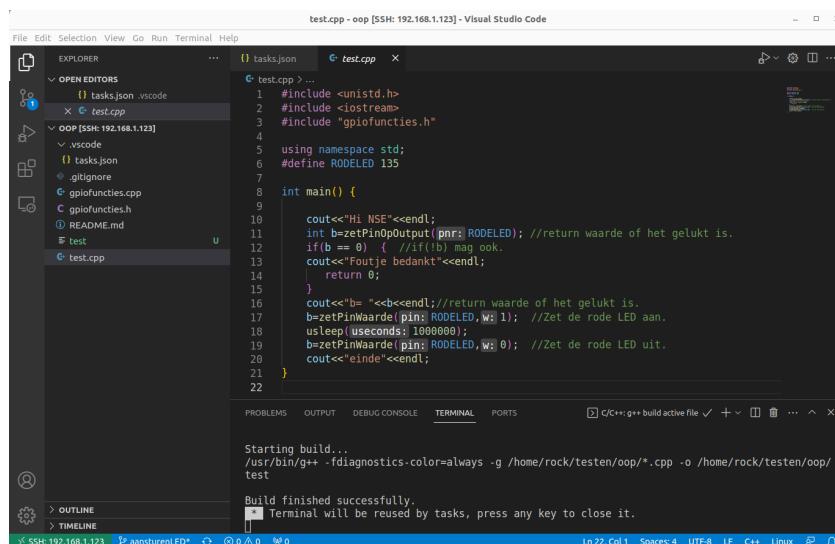
- Start VSC op en maak verbinding met de RockPi.

- Klik op remote Explorer . De IP nummers van de remote system(en) die al eerder gebruikt zijn worden zichtbaar, zoals te zien is in Figuur 2.6



Figuur 2.6: Remote Explorer van VSC

- Maak een nieuwe connectie aan, door op  te klikken (rood omkaderd).
- Nadat een verbinding gemaakt is met de RockPi, open een remote folder door op  te klikken (of druk Ctrl+K Ctrl+O) en ga naar de directory van de vorige opdracht: /home/rock/Documents/ledje/oop/
- Aan de linkerkant worden de files zichtbaar en onderaan een statusbalk met informatie. Dit is te zien in figuur 2.7



Figuur 2.7: VSCode in de gewenste directory

Klik op select folder en selecteer de folder waarin de files staan.

- Selecteer de file *test.cpp* en compileer de files<sup>1</sup> (Terminal → Run Build Task... of Ctrl+Shift+B)
- Klik links van regelnummer 10, er verschijnt een rode stip, dit is een breakpoint.
- Selecteer bij het pijltje rechtsboven de Debug optie en klik op de pijl (of Run → Start Debugging), de debugger wordt gestart en het beeldscherm zoals in figuur 2.8 verschijnt.

Figuur 2.8: Start van de debug sessie.

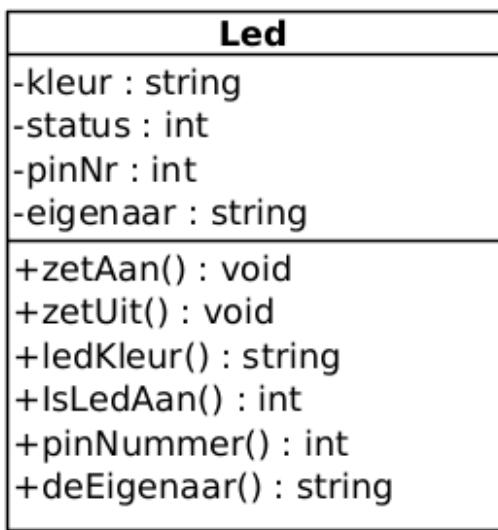
Met de debugknoppen rechtsboven kan nu stap voor stap door het programma gelopen worden.

Doorloop het programma stap voor stap, zodat de LED ook daadwerkelijk bij een stap aan- en uitgaat.

3. Bij deze opdracht wordt de eerste klasse gemaakt.

- Maak een nieuwe terminal aan (Terminal → New Terminal) of Ctrl+Shift+' en maak een nieuwe directory aan.
- clone de volgende code:  
git clone --branch opdrLedH https://github.com/JohnVi-hhs/oop.git
- De UML notatie van de klasse **Led** wordt weergegeven in figuur 2.9 en de headerfile in listing 2.2

<sup>1</sup>compileren lukt niet, controleer of C++ extension () is geïnstalleerd



**Figuur 2.9:** UML diagram van de klasse Led

```

class Led
{
public:
    Led(int);
    Led(int, string);
    Led(int, string, string);
    ~Led();
    void zetAan();
    void zetUit();
    string ledKleur() const;
    int isLedAan() const;
    int pinNummer() const;
    string deEigenaar() const;

private:
    string kleur;
    int pinNr;
    int status;
    string eigenaar;
};

```

**Listing 2.2:** LED declaratie file(.h)

**Opdracht:** Implementeer de Led.cpp zodat het hoofdprogramma van listing 2.3 zonder errors en warning gecompileerd en uitgevoerd kan worden.

```

#include <unistd.h>
#include <iostream>
#include <string>
#include "Led.h"

using namespace std;
#define RODELED 135
#define GROENELED 132
#define GELELED 134

int main() {

    cout<<"Hi_NSE"<<endl;
    Led rood(RODELED, "Rood", "Pietje_Puk");
    Led geel(GELELED, "Geel");
    Led groen(GROENELED);

    groen.zetAan();
    usleep(1000000);
    groen.zetUit();
    geel.zetAan();
    usleep(1000000);
    geel.zetUit();
    rood.zetAan();
    usleep(1000000);
    rood.zetUit();

    cout<<"einde"<<endl;
}

```

**Listing 2.3:** Hoofdprogramma om de LED uit te testen

## 2.3 Het werken met een visuele debugger.

Bij dit onderdeel van de opdracht wordt gewerkt met een grafische debugger. Hiermee wordt een grafische weergave gedaan wat een object is en wat een object van een afgeleide klasse inhoud (dit laatste komt in opgave 2 te spraken). Verder worden de associaties tussen objecten duidelijk weergegeven (dit wordt in week 4 en 5 gedaan).

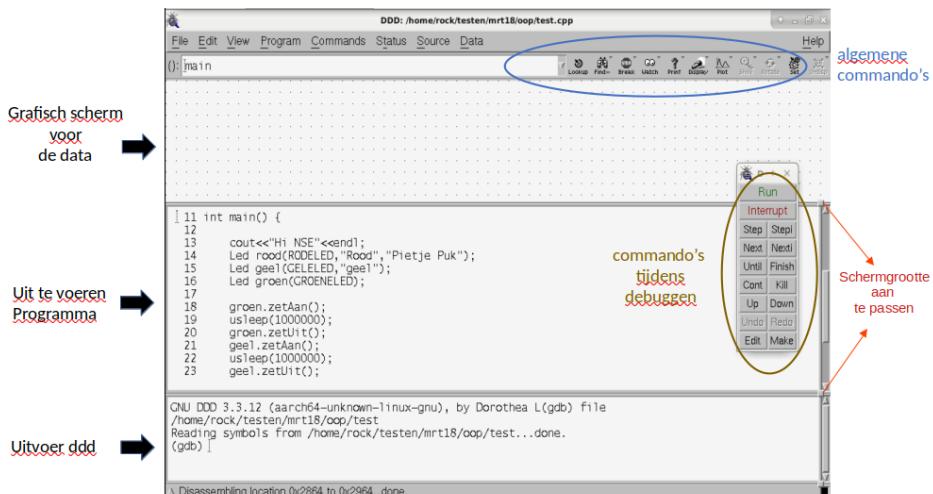
De grafische debugger die gebruikt wordt is de DDD debugger(Data Display Debugger)

Een paar handige links hierbij zijn:

- DDD manual in [html](#) en [pdf](#)
- [GNU DDD project](#)
- [taufanlubis.wordpress.com](#)
- [Swarthmore](#)
- [linuxfocus](#)

De instellingen van de DDD debugger staan in de file .ddd die staat in de home directory (`/home/rock`). Soms is het eenvoudiger om deze directory te deleten dan de instellingen aan te passen.

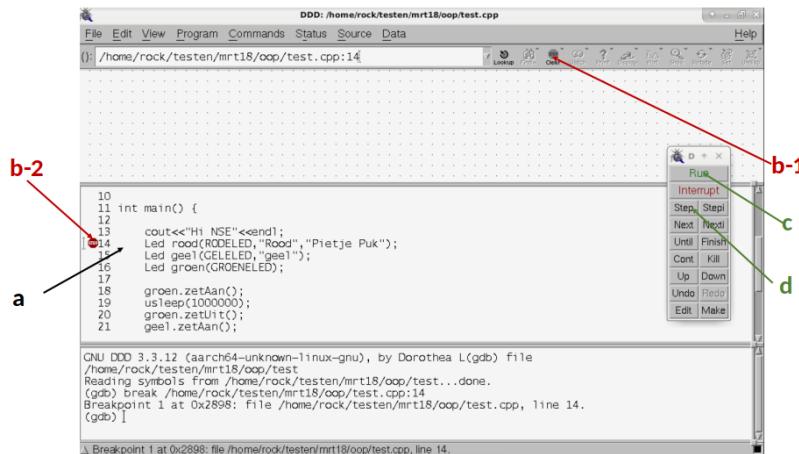
We gaan nu met de DDD debugger werken. Om met DDD te kunnen werken hebben we een grafische omgeving nodig. Een handige methode is om de VNC viewer van de host te gebruiken. Deze viewer toont de grafische uitvoer van de RockPi. Open in de VNC een terminal, ga naar de directory die in de vorige opdracht gemaakt is (waar de files test.cpp, Led.cpp en Led.h **test** staan) en start de DDD debugger op: `ddd test`, vervolgens wordt het scherm, zoals weergegeven in Figuur 2.10, getoond.



Figuur 2.10: het DDD opstartschermscherm.

**Opdracht** We gaan hierbij stap voor stap het programma doorlopen, waarbij de inhoud van de objecten van de klasse Led wordt getoond.

- a Als eerste een korte kennismaking met de DDD debugger.
- i Ga met de cursor op de regel `Led rood(RODELED, "Rood", "Pietje Puk");` (regelnummer 14)
- ii Klik op *Break* (bij algemene commando's). Voor het begin van de regel verschijnt nu een breakpoint.



Figuur 2.11: het zetten van een breakpoint.

- iii Klik op *Run* (bij debug commando's), het programma wordt uitgevoerd tot het breakpoint. Er verschijnt een groene pijl bij het breakpoint.
- iv Klik op *Next*, de regel wordt uitgevoerd.  
De groene pijl komt voor de regel `Led geel(GELELED, "Geel");` te staan.
- v Klik op step, met dit commando *Step* wordt in de constructor van de klasse Led gestapt. Doorloop het programma verder stap voor stap.
- b Het zichtbaar maken van de inhoud van de objecten.
- i Start de DDD debugger op (`ddd test`):
- ii Zet een breakpoint op de regel `groen.zetAan();`
- iii Klik met de muis op variabele `rood` en vervolgens op Display of klik met rechtermuisknop en vervolgens op Display `rood`. Het object `rood` van de Klasse `Led` wordt nu weergegeven.
- iv Doe hetzelfde met de variabele `geel` en `groen`.
- v Als het goed is ziet de debugger er ongeveer uit zoals Figuur 2.12, alleen met je **eigen naam**.
- vi Ga met het Step commando de methode `zetAan` van `groen` in.

- vii Ga met het Step commando de methode zetAan van geel in. Je ziet dat de methode hetzelfde is, alleen de attributen hebben een andere waarde.

```

DDD: /home/rock/testen/mrt18/oop/test.cpp
File Edit View Program Commands Status Source Data
0: gee1
1: rood
kleur = "Rood"
pinNr = 135
status = 0
eigenaar = "Pietje Puk"
2: geel
kleur = "geel"
pinNr = 134
status = 0
eigenaar = "Anoniem"
3: groen
kleur = "Kleurloos"
pinNr = 132
status = 0
eigenaar = "Anoniem"

6 using namespace std;
7 #define RODELED 135
8 #define GROENELED 132
9 #define GELELED 134
10
11 int main() {
12
13     cout<<"Hi NSE"<<endl;
14     Led rood(RODELED,"Rood","Pietje Puk");
15     Led geel(GELELED,"Geel");
16     Led groen(GROENELED);
17
18     groen.zetAan();
19     usleep(1000000);
20     groen.zetUit();
21     geel.zetAan();
22     usleep(1000000);

Hi NSE
Breakpoint 1, main () at /home/rock/testen/mrt18/oop/test.cpp:18
(gdb) graph display rood
(gdb) graph display geel
(gdb) graph display groen
(gdb)

```

△ Display 2: geel (enabled, scope main, address 0x7fffffff028)

**Figuur 2.12:** Weergave van twee objecten van de klasse Led.

- c Maak een screenshot die lijkt op Figuur 2.12 alleen met je **eigen naam** in plaats van "Pietje Puk" en upload deze op Brightspace.
- d Laat de opdracht aftekenen met onder andere een zichtbaar screenshot.

# Appendices



# A Tips&Tricks voor Linux omgeving

## A.1 Tips

*Je kunt de tips en trucs toepassen bij de commando's in A.2*

- Linux is in tegenstelling tot Windows ‘case-sensitive’. Het maakt dus verschil of je hoofdletters of kleine letters gebruikt. Voorbeeld: Als je map Documents heet, dan werkt cd Documents wel en cd documents niet.
- Tip bij het invoeren van map- en bestandsnamen: tik de eerste 3 letters in en druk de Tab toets in, Linux vult dan automatisch de rest van de naam in. Voordeel: je hoeft minder te tikken en het is gelijk een foutcontrole.
- Op en neer pijltjestoetsen in de terminal: door vorige commando's heen gaan.
- Nog een verschil met Windows; Windows werkt met bestandsextensies: \*.exe is een uitvoerbaar (executable) bestand. In Linux doet de extensie er niet toe, maar zet je met chmod +x het ‘executable’ bitje aan, een attribuut op het bestand.
- Hulp vragen bij commando's: <commando> --help. Voorbeeld: ls --help Uitgebreidere informatie: man <commando>. Voorbeeld: man ls Voor de bedieningstoetsen van man druk ‘h’.
- **Uitleg en gebruik van het sudo commando** ('**S**uper **U**ser **D**o'); admin rechten.
- Zoeken in het bestandssysteem: sudo find / -name <bestandsnaam> zoekt naar <bestandsnaam> vanaf de root '/' in het gehele bestandssysteem.
- **Uitleg van- en navigeren door het bestandssysteem.**
- Wil je meer uitleg, [zoek dan naar ‘Linux Command Cheatsheet’](#). Ik vind  [deze uitleg](#) wel prettig.

## A.2 Linux commando's (zie ook tips in A.1)

Commando	Uitleg
<code>ls -l</code>	Lange directory listing - Opvragen van bestanden in een directory
<code>cd</code>	change directory; <code>cd ..</code> betekent 1 niveau omhoog naar de root '/'
<code>cp</code>	copy
<code>rm</code>	remove - Verwijderen van bestanden en mappen.
<code>mv</code>	move - Verplaatsen van bestanden en mappen.
<code>cat</code>	bestand bekijken
<code>pwd</code>	In welke directoy sta ik? ('present working directory')
<code>mkdir</code>	make directory
<code>rmdir</code>	remove (lege) directory
<code>history</code>	Laat lijst zien van laatst uitgevoerde commando's
<code>sudo</code>	Zet dit vóór je commando om het met admin rechten uit te voeren.
<code>chmod</code>	(met sudo) lees/schrijfrechten wijzigen, +x maakt bestand executable.
<code>chown</code>	(met sudo) eigenaar van bestand / map wijzigen

## A.3 GPIO poorten in Linux

In Linux wordt alles gerepresenteerd als een bestand. Of het nu een volledige disk of een device is, je kunt het als een bestand openen en lezen en schrijven. Dit geldt ook voor GPIO pinnen ('General Purpose Input/Output' pinnen). Deze pinnen gebruik je op dezelfde manier als de pinnen van een microcontroller (denk aan een Arduino Uno, of een Microbit). Je kunt pinnen als uitgang zetten en hoog of laag maken, of als ingang instellen en uitlezen of een pin hoog of laag is (en nog meer, bijvoorbeeld pull-up en pull-down en reageren op events, maar dat gaan we niet behandelen).

Omdat pinnen gerepresenteerd worden door bestanden, geldt ook dat er rechten op gezet worden in het bestandssysteem. Meestal kan een standaard gebruiker (zoal 'rock') er niet bij. Alleen 'root' kan er standaard bij. Voor het practicum hebben we de rechten zo ingesteld dat gebruiker 'rock' standaard gebruik kan maken van de op de breakout board beschikbare pinnen.

Voorbeeld van de handelingen om een pin in te stellen en uit te lezen of aan te sturen, het hekje '#' aan het begin van de regel geeft aan dat het commentaar (uitleg) is:

```
# GPIO pin 135 (de rode led) aanzetten:  
echo 135 > /sys/class/gpio/export  
# GPIO pin 135 (de rode led) als uitgang instellen:  
echo out > /sys/class/gpio/gpio135/direction  
# GPIO pin 135 (de rode led) hoog maken:  
echo 1 > /sys/class/gpio/gpio135/value  
# 1 seconde wachten  
sleep 1  
# GPIO pin 135 (de rode led) laag maken:  
echo 0 > /sys/class/gpio/gpio135/value
```

Je kunt ook de hele bestandstructuur van de pin bekijken:

```
rock@rockpi-4b:~$ ls -l /sys/class/gpio/gpio135/  
total 0  
-rwSrwsr-- 1 root gpio 4096 Mar 12 18:49 active_low  
lrxwxrwxrwx 1 root gpio 0 Mar 12 18:49 device -> ../../../../../pinctrl  
-rwSrwsr-- 1 root gpio 4096 Mar 12 18:49 direction  
-rwSrwsr-- 1 root gpio 4096 Mar 12 18:49 edge  
drwsrwsr-x 2 root gpio 0 Mar 12 18:49 power  
lrxwxrwxrwx 1 root gpio 0 Mar 12 18:49 subsystem -> ../../../../../../  
    class/gpio  
-rwSrwsr-- 1 root gpio 4096 Mar 12 18:49 uevent  
-rwSrwsr-- 1 root gpio 4096 Mar 13 10:55 value
```

En een bepaalde status uitlezen (in dit geval of de pin als ingang of uitgang is ingesteld):

```
rock@rockpi-4b:~$ cat /sys/class/gpio/gpio135/direction  
out  
rock@rockpi-4b:~$
```

Tot zover de uitleg. Ik laat GPIO pin als ingang en gebruik met PWM voor nu achterwege. Als je daar meer van wilt weten, kijk dan even naar het zelftest scriptje: sudo cat /usr/local/bin/pwmtest.sh

# B Installeren van de practicum omgeving

## B.1 De VNC viewer

Installeer [VNC Viewer](#) op je laptop. Op de RockPi is VNC Server al geïnstalleerd. Start op je laptop de VNC viewer op. Vul in het scherm *VNC CONNECT*, zie Figuur B.1, het IP adres van de RockPi in.

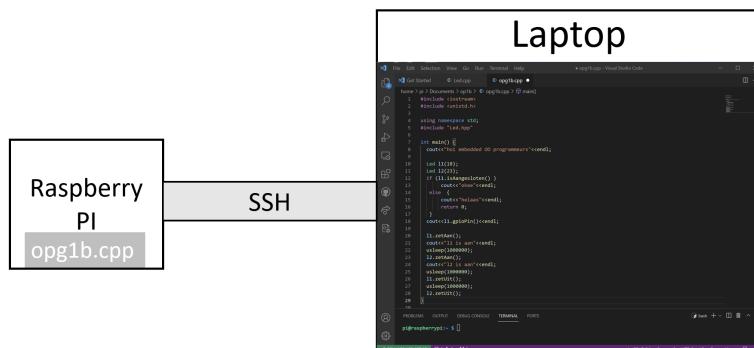


Figuur B.1: VNC verbinding met de RockPi maken.

Als het goed is zie je nu de grafische omgeving van de RockPi. Als dit niet werkt, druk dan eens op de '**Restart\_X**' knop op het uitbreidingsbordje. Dit herstart de X-Windows grafische schil op de RockPi. Probeer daarna opnieuw te verbinden.

## B.2 Het gebruik van 'Visual Studio Code'

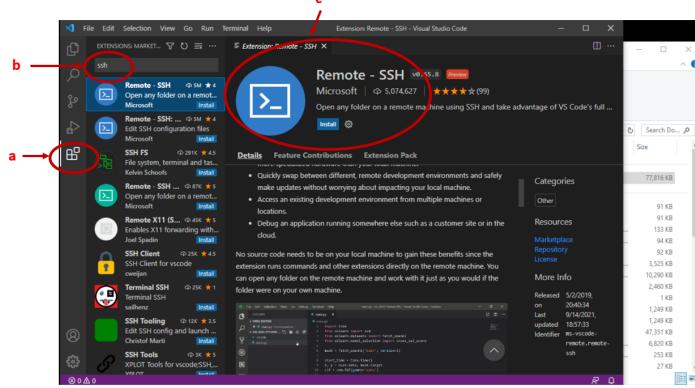
[Visual Studio Code](#) is een cross platform editor uit de Microsoft omgeving. Je kan vanuit je laptop/desktop omgeving direct files editeren op een embedded platform zoals een RockPi. Dit principe is te zien in Figuur B.2. Op de laptop draait 'Visual Studio



Figuur B.2: 'Visual Studio Code' en de RockPi.

Code' (VSC) en de source file is de file `opg1.cpp` op de RockPi. Het compileren kan via VSC, maar kan ook via de terminal in VSC of via een externe terminal b.v. Windows PowerShell, KiTTY / PuTTY of een andere.

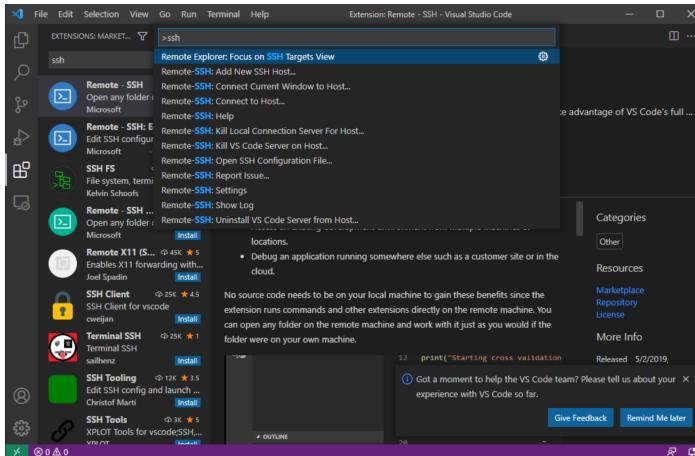
1. Download en installeer VCS [Visual studio code](#)
2. Bij Visual Studio Code kunnen meerdere extensions geïnstalleerd worden. Het installeren van de *remote SSH* extension kan als volgt gedaan worden:



**Figuur B.3:** Installeren van Remote SSH in VCS.

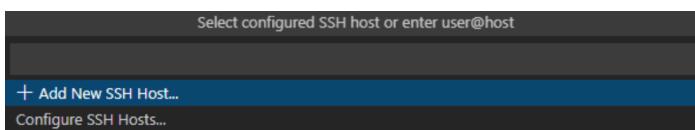
- (a) Klik op Extension of CTRL + Shift + X
  - (b) Zoek naar ssh
  - (c) klik op install
3. Verbinding maken met de RockPi.

- (a) Ga naar het 'Command Palette' (CTRL+Shift+P). Voer het command Remote-SSH:Connect to Host...



**Figuur B.4:** SSH verbinding naar de Host.

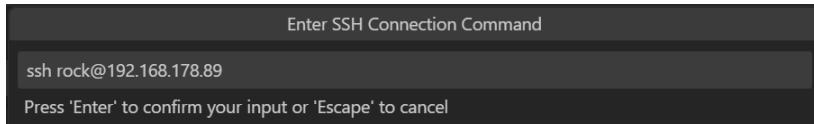
Zoals te zien is in Figuur B.4. VSC komt nu met het scherm zoals te zien is in Figuur B.5.



**Figuur B.5:** toevoegen van een host.

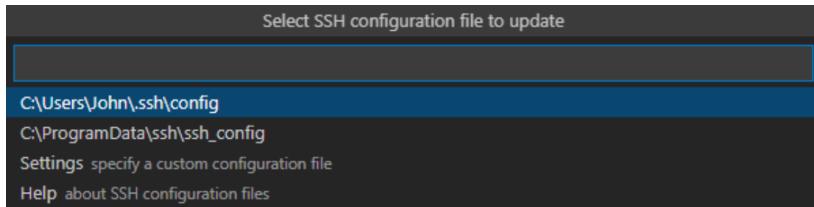
- (b) klik op *+ Add New SSH Host...*. VSC vraag vervolgens om een ssh command:

type in: ssh rock@*ip.nr.* van de host zoals te zien is in Figuur B.6. Uiteraard wel met je eigen IP nummer. VSC wil de gegevens opslaan en vraagt



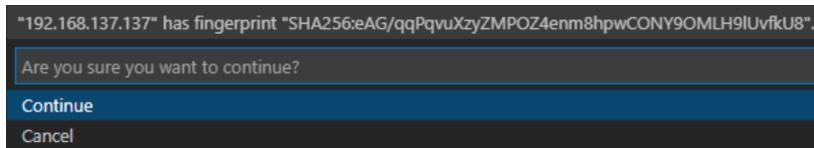
**Figuur B.6:** connectie maken met de host.

vervolgens waar deze opgeslagen moet worden, zoals te zien is in Figuur B.7.



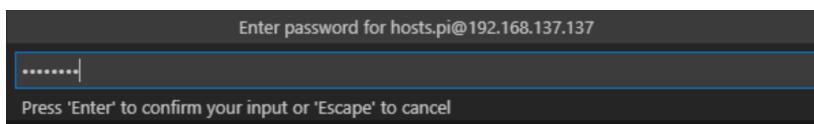
**Figuur B.7:** waar de gegevens moeten worden opgeslagen.

VSC slaat de gegevens op en geeft een bericht (rechtsonder) of de config file geopend moet worden of dat een connectie gemaakt moet worden. Maak een connectie, VSC komt vervolgens met de vraag of je zeker weet dat je doorgaat, zoals Figuur B.8 laat zien.



**Figuur B.8:** waar de gegevens moeten worden opgeslagen.

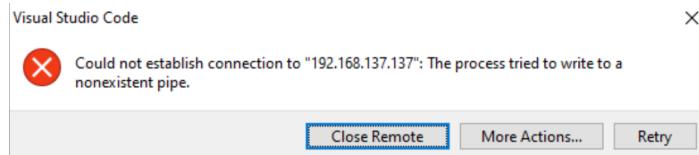
- (c) Klik op *Continue*. VSC vraagt om een password zoals Figuur B.9 laat zien. Het password is: *rock*



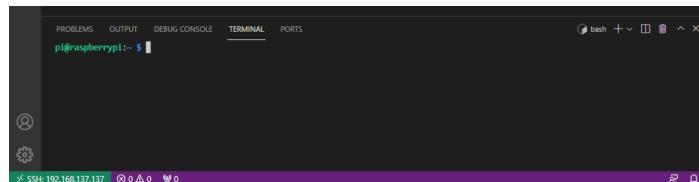
**Figuur B.9:** invullen van een password.

Het kan zijn dat een foutmelding gegeven wordt zoals b.v. in Figuur B.10 laat zien. Raak niet in paniek en probeer desgewenst weer opnieuw.

- (d) Maak een nieuwe terminal aan (Terminal → New Terminal). Als het goed is verschijnt de terminal in het onderste deel van VSC met een ssh verbinding naar de RockPi, zoals Figuur B.11 laat zien.
- Met het *ls* commando vraag je de listing op van de huidige directory.



**Figuur B.10:** een foutmelding.



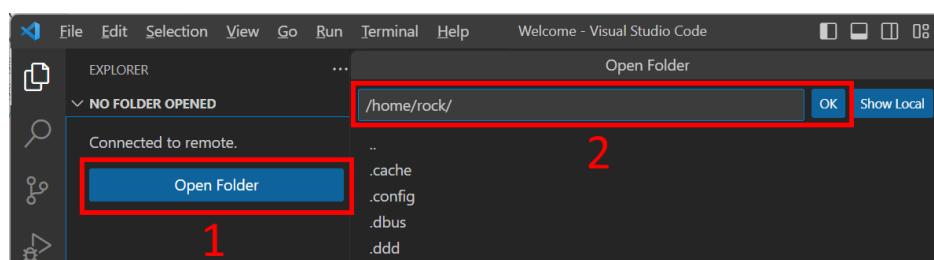
**Figuur B.11:** Een geopende terminal in VSC.

- ii. Met het *pwd* commando wordt het pad van de huidige directory zichtbaar.  
*pwd* <enter> → /home/rock.
- iii. Met het *cd* commando wordt naar een directory gegaan  
(LET OP: Ik ga er van uit dat je een **USB stick** gebruikt!):  
*cd Documents* <enter> → **rock@rockpi-4b:~/Documents \$**
- iv. Met het *mkdir* commando wordt een directory aangemaakt.  
Maak in de directory *Documents* een directory *intro* aan.

Ga naar de directory *intro* (*cd intro*) en vraagt het path op.  
*pwd* → /home/rock/Documents/intro

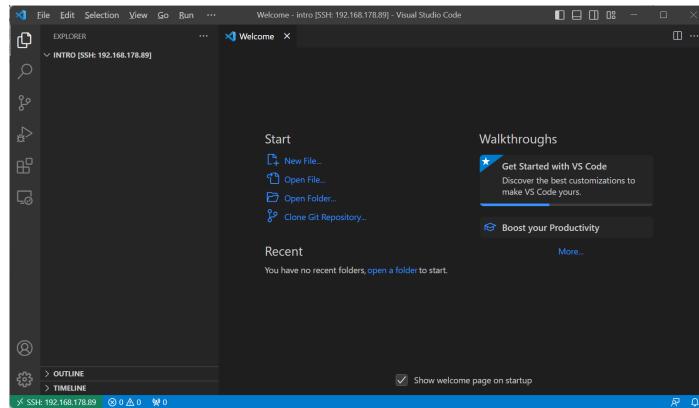
#### 4. Het eerste programma op de RockPi.

- (a) VSC werkt voornamelijk met directory's (folders). Het openen van een folder in VSC op de RockPi kan door te klikken op Explorer of Ctrl+Shift+E. VSC opent een scherm, zoiets als [1] in Figuur B.12.



**Figuur B.12:** De directory die geopend moet worden.

Voer bij [2] de directory */home/rock/Documents/intro* in en klik op 'OK'. VSC kan vragen of je de auteur vertrouwt. Klik op 'Yes'. In het *EXPLORER* veld verschijnt vervolgens de nu nog lege werkdirctory, zoals in Figuur B.13 te zien is.



**Figuur B.13:** In het EXPLORER veld is de werkdirectory te zien.

- (b) Het programma van listing B.1 kan: gecreëerd worden door een nieuwe file *hoi.cpp* aan te maken [i], gedownload worden van Brightspace [ii] of gecloned worden van Github [iii].
    - i. Klik op New File en geef de filenaam *hoi.cpp*. Type/kopieer de tekst van Listing B.1 in de file *hoi.cpp*

```
#include <stdio.h>
int main() {
    printf("Hoi_programmeurs_van_de_wereld\n");
    return 0;
}
```

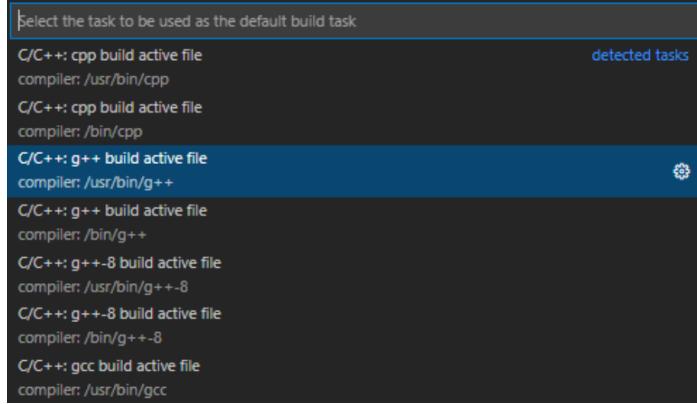
**Listing B.1:** het meest gebruikte voorbeeld programma.

    - ii. Download de file *hoi.cpp* van Brightspace en plaats deze in de directory. Het laatste kan zelfs gedaan worden door middel van slepen van de file.
    - iii. Via git: *git clone - -branch intro https://github.com/Johnny63Vi/oopr1.git*<sup>1</sup>  
Het nadeel van deze methode is dat een directory *oopr1* wordt aangemaakt waarin de file komt te staan.
  - (c) Open de bestaande terminal of open een nieuwe terminal *Ctrl+Shift+T*. VSC opent een terminal in de werkdirectory. compileer de file *hoi.cpp* met de *g++* compiler (*g++ hoi.cpp*) en run het gecompileerde programma *./a.out*.  
Uiteraard kan ook gekozen worden voor een externe terminal zoals, *Windows PowerShell*, *PuTTY*, etc.  
Delete *a.out* (*rm a.out*).
5. Compileren via VSC.  
Je kan het programma ook via VSC laten compileren.
- (a) Daarvoor moet de C++ omgeving in Visual Studio Code worden geïnstalleerd (is al gedaan bij de RockPi):

---

<sup>1</sup>bij uitvoering geen spatie tussen - -

- i. Klik op Extension  of Ctrl+Shift+X.
  - ii. Type in C++ en installeer de C++ , het Extension pack en C/C++ Themes.
- (b) Activeer het tabblad *hoi.cpp*.
- (c) Klik op *Terminal* → *Configure Default Build Task...*. VSC komt met een scherm zoals Figuur B.14. Kies voor C/C++: g++ build active



Figuur B.14: Het kiezen van de compiler.

file. VSC maakt nu een task.json file aan waarin de gegevens betreffende compiler worden opgeslagen.

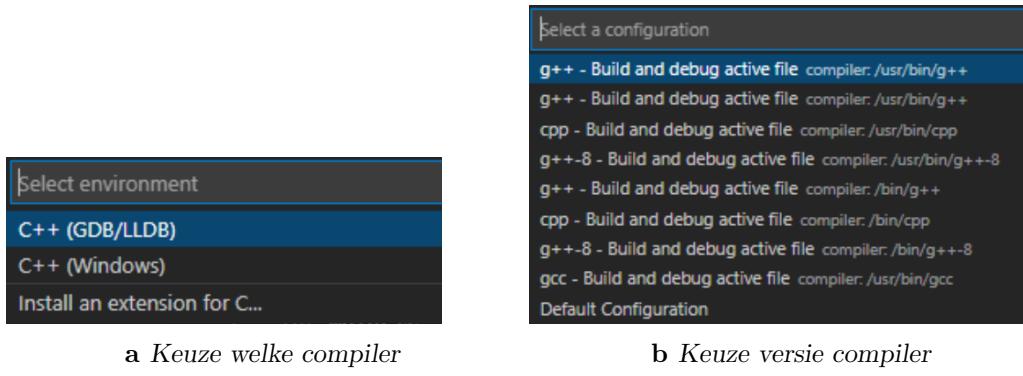
- (d) Activeer opnieuw *hoi.cpp* tab. Ga naar Terminal → Run Build Task... of Ctrl+Shift+B. VSC zal nu *hoi.cpp* compileren.
- (e) Het gecompileerde programma kan nu gerund worden in b.v. een terminal. Open een terminal of maak een nieuwe terminal aan Ctrl+Shift+' en run hoi (*./hoi*).

## 6. Run/Debug via VSC.

Je kan een programma ook direct via VSC laten runnen en Debuggen.

- (a) Activeer het tabblad *hoi.cpp*.
- (b) Klik op *Run and Debug* (Ctrl+Shift+D)  en vervolgens op **Run and Debug**. Afhankelijk van instellingen kan VSC kiezen voor Figuur B.15a. Kies hierbij C++ (GDB/LLDB), VSC laat hierna Figuur B.15b zien. Kies voor: *g++ - Build and Debug active file compiler /usr/bin/g++*. VSC maakt o.a. *launch.json* file aan. Hierin worden diverse gegevens met betrekking tot de compiler/debugger in opgeslagen. Het zou ook kunnen dat VSC de DEBUG CONSOLE openet.
- (c) Klik op tabblad *hoi.cpp* en plaats vervolgens een breakpoint voor regel 4 (linker muisklik voor regel 4) (*printf*), een rode stip verschijnt voor de 4.
- (d) Start de debugger:Run → Start Debugging of F5 of klik op . Het programma wordt op de RockPi uitgevoerd en stopt op regel 4, zoals te zien is

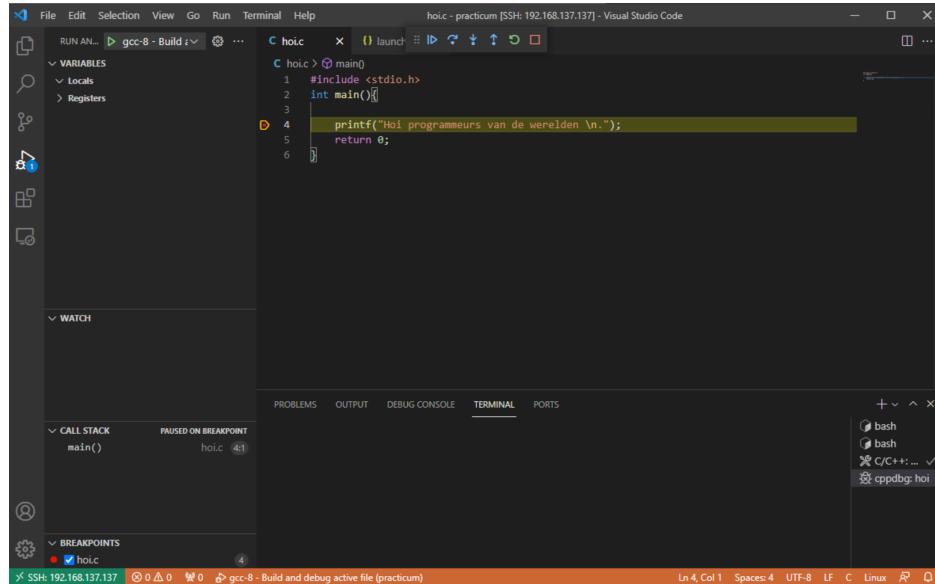
in Figuur B.16.



a Keuze welke compiler

b Keuze versie compiler

**Figuur B.15:** Keuze compiler te gebruiken door VSC



**Figuur B.16:** Debuggen met VSC.

Met behulp van de debugknoppen kan stap voor stap door het programma gelopen worden. Hieronder staan de belangrijkste:

	'Continue (F5)'	Vervolg het programma.
	'Step Over (F10)'	Stap over de volgende regel heen (ook als het een functieaanroep is).
	'Step into (F11)'	Voer de regel uit. Als het een functieaanroep is: ga de functie in. <i>Als het een systeemfunctie is, dan wordt het vaak onbegrijpelijk!</i>