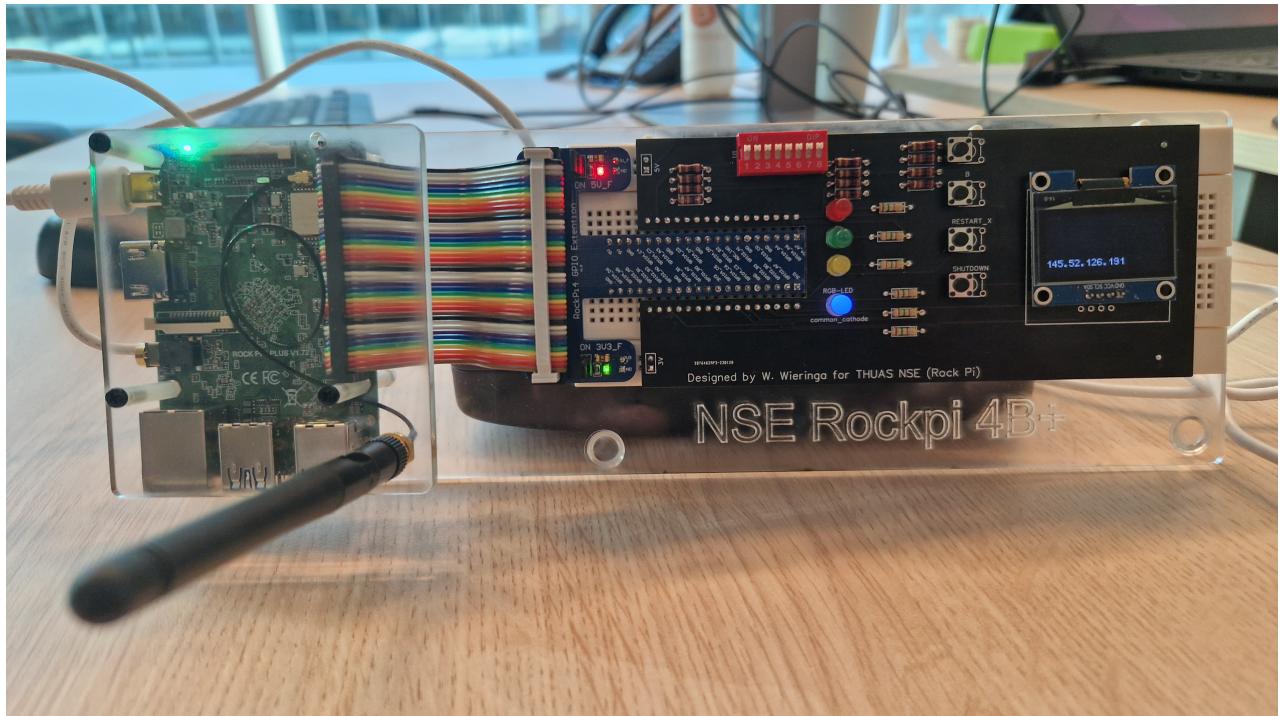


Object Georiënteerd Programmeren

(Introductie)

(practicumhandleiding inclusief de practicumopstelling voor de Rock Pi)

Versie 0.5



Contents

1 Klasse en objecten in C++	3
1.1 De eerste kennismaking met de RockPi	3
1.1.1 Contact maken met de RockPi	3
1.1.2 Het aansturen van een LED.	4
1.2 Het werken vanuit Visual Studio Code	6
1.3 Het werken met een visuele debugger.	8
Appendices	12
A Tips en tricks in een linux omgeving	14
A.1 commando's	14
A.2 tips	14
B Installeren van de practicum omgeving	15
B.1 De VNC viewer	15
B.2 Het gebruik van Visual Studio Code	15

1 Klasse en objecten in C++

Als eerste wordt er contact gelegd tussen de laptop en de RockPi, waarna een LEDje aan- en uitgezet wordt. Vervolgens wordt door middel van een programma een aantal LEDs aangestuurd.

Het practicum wordt gedaan op een RockPi dit is een single board computer dat draait in ons geval met het linux operating systeem. Met behulp van de RockPi wordt tijdens het practicum door middel van objecten diverse LEDs aangestuurd. Dit wordt gedaan door een 1 of een 0 naar naar een file te schrijven. In Figuur 1.1 wordt weergegeven hoe de RockPi op het lab netwerk is aangesloten. Dit kan via de lab-wifi of via een

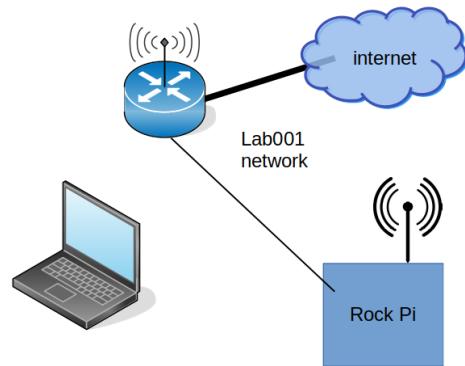


Figure 1.1: De rockPi in het lab-netwerk

UTP kabel. Het bijbehorende Ipnr. wordt getoond op het displaytje dat aangesloten is op de RockPi. Om contact te kunnen maken tussen de laptop en de RockPi moet de laptop **ook** op het **labnetwerk(Lab001)** aangesloten zijn, b.v. via de wifi.

1.1 De eerste kennismaking met de RockPi.

1.1.1 Contact maken met de RockPi

In eerste instantie wordt er contact gemaakt met de RockPi via het *ssh* commando. Het IP nummer van de rockPi is af te lezen via het schermpje, zoals in figuur 1.2 wordt

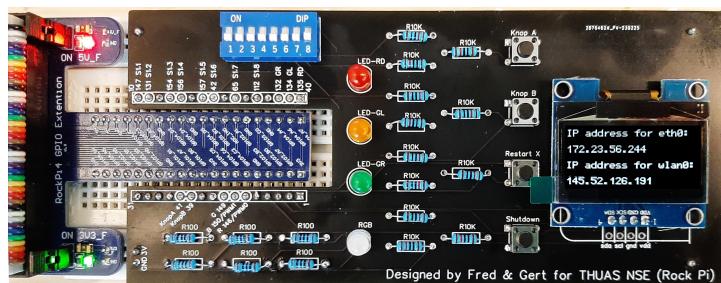
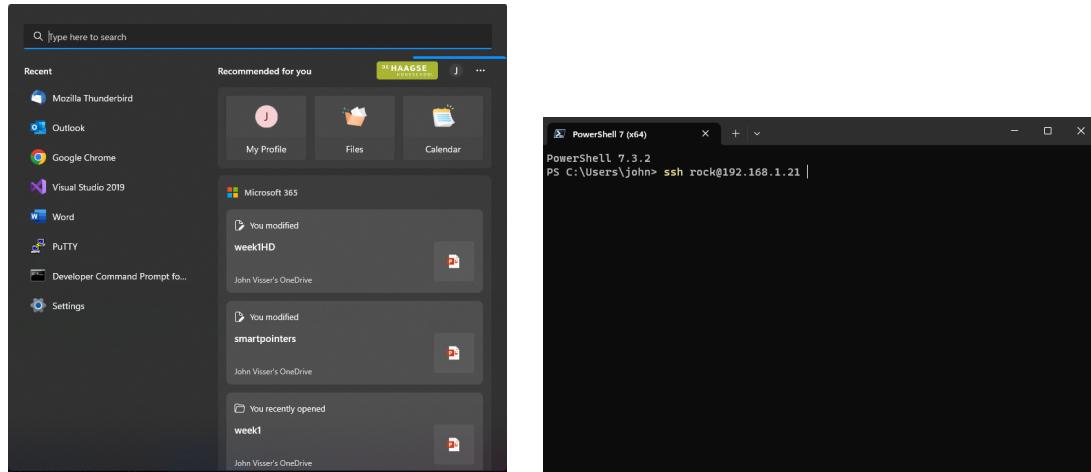


Figure 1.2: De rockPi met het IPnr.

weergegeven. In dit voorbeeld is het IP nummer *192.168.1.21*.

Open in windows een terminal (b.v. een powershell, cmd prompt of een extern programma zoals b.v. putty), b.v.powershell. Ga naar windows search, zie figuur 1.3a en type vervolgen in *powershell*. In de powershell kan het *ssh* commando met de username



a Zoekscherf in windows

b ssh verbinding maken naar de rockPI

Figure 1.3: Opstelling bij opdracht 1.

en ipnr gegeven worden, zoals in figuur 1.3b wordt weergegeven. Bij de eerste keer zal de melding komen of de host key moet worden opgeslagen, type in *yes*. Vervolgens zal om om het password gevraagd worden. Dit is *rock*.

Wanneer het inloggen gelukt is, verschijnt eerst een verhaaltje dat eindigt met de prompt van de rockPI, zoals in figuur 1.4 te zien is. Vanaf nu werk je in een linux

```

rock@rockpi-4b: ~
PowerShell 7.3.2
PS C:\Users\john> ssh rock@192.168.1.21
rock@192.168.1.21'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: Thu Feb 16 09:40:20 2023 from 192.168.1.110
xset: unable to open display ""
xset: unable to open display ""
xset: unable to open display ""
Can't open display
Can't open display
Pas de scherresolutie in vnc aan met (bijvoorbeeld) xrandr --fb 1920x1080
Om dit automatisch/permanent te doen, wijzig dit via nano ~/.bashrc
Werk in een map in de home directory (bijv ~/opdr1). Ruim dit na afloop op.
Verplaats je bestanden dan naar usb stick, te vinden onder /home/rock/usb
Have fun... Gert
rock@rockpi-4b:~$ |

```

Figure 1.4: Ingelogd in de rockPi

terminal 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 appendix A.

1.1.2 Het aansturen van een LED.

Wanneer contact is gemaakt met de Pi, zoals beschreven in hoofdstuk 1.1.1, 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 1.5. De

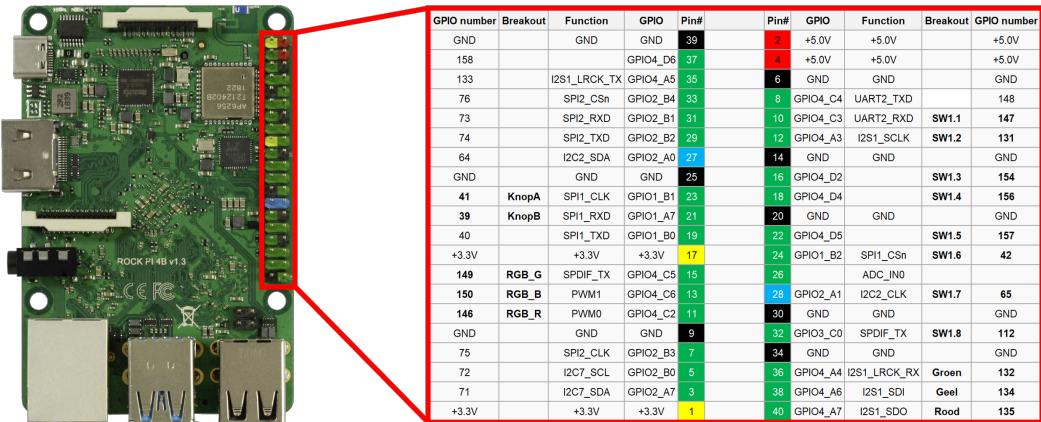


Figure 1.5: De GPIO connector van de RockPI

buitenste kolom geeft aan de GPIO nummer die gebruikt kan worden waarop deze geprogrammeerd kan worden. Zo zit de rode LED aangesloten op GPIO nummer 135.

Opdracht 1a, het eerste programmaatje op de RockPI

1. Plaats de USB stick in de Rock PI.
2. Ga naar de directory van je USB stick
 - `cd /media/rock`
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 git hup. Het main programma (`test.cpp`) wordt in Listing 1.1 weergegeven
 - `git clone https://github.com/JohnVi-hhs/oop.git`

```
#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 1.1: 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.

6. 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`

1.2 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.

1. 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 1.6

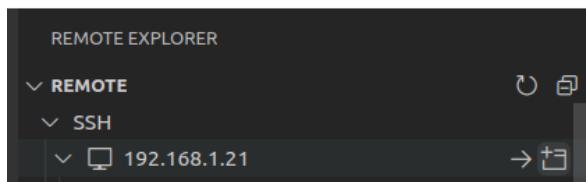


Figure 1.6: Remote Explorer van VSC

- Maak een nieuwe connectie aan, door op te klikken.
2. Nadat een verbinding gemaakt is met de RockPi, open een remote folder door op te klikken of (Cntrl + k Cntrl + o) en ga naar de directory van de vorige opdracht. Aan de linkerkant worden de files zichtbaar en onderaan een statusbalk

met informatie, indien de runnable extensie  geïnstalleerd is. Dit is te zien in figuur 1.7

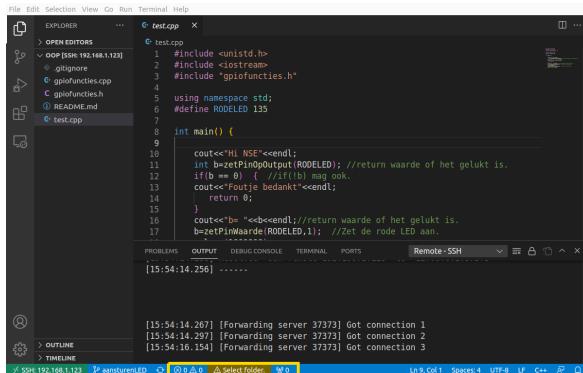


Figure 1.7: VSCode in de gewenste directory

Klik op select folder en selecteer de folder waarin de files staan. Hierna voert de Runnable extensie een aantal opdrachten uit en wordt de statusbalk uitgebreid, zoals te zien is in figuur 1.8.



Figure 1.8: De statusbalk nadat de directory is gekozen.

Klik op , de runnable extension compileert nu het programma.

- Door met de debugger te werken, kan een aantal onduidelijkheden van het programmeren verduidelijkt worden.
 - Klik links van regelnummer 10, er verschijnt een rode stip, dit is een breakpoint.
 - Klik op , de debugger wordt gestart en het beeldscherm zoals in figuur 1.9 verschijnt.

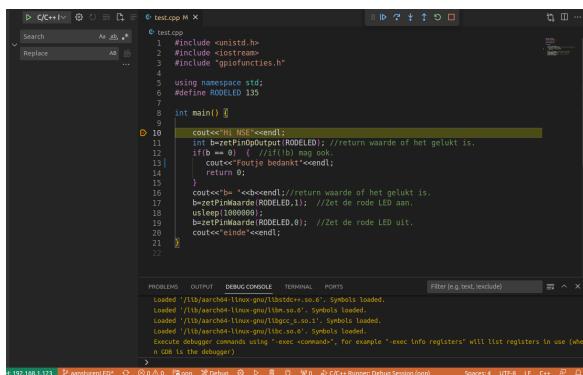


Figure 1.9: Start van de debug sessie.

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.

4. Bij deze opdracht wordt de eerst klasse gemaakt.

- Maak een nieuwe terminal aan (Terminal → New Terminal) of Cntr +Shift+` en maak een nieuwe directory aan.
- clone de volgende code:
git clone --branch opdrLedH <https://github.com/JohnVi-hhs/oop.git>
- De klasse **Led** wordt weergegeven in figuur 1.10

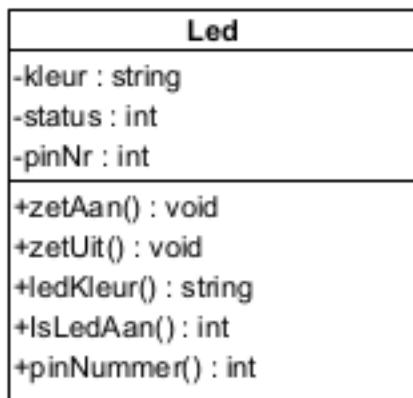


Figure 1.10: Diagram van de klasse Led

Implementeer de Led.cpp

1.3 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 3 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 opdracht ?? van hoofdstuk ?? aangemaakt is (waar de files `blink1.cpp`, `Led.cpp` en `Led.h` **blink1** staan) en start de DDD debugger op: `ddd blink1`, vervolgens wordt het scherm, zoals weergegeven in Figuur 1.11, getoond.

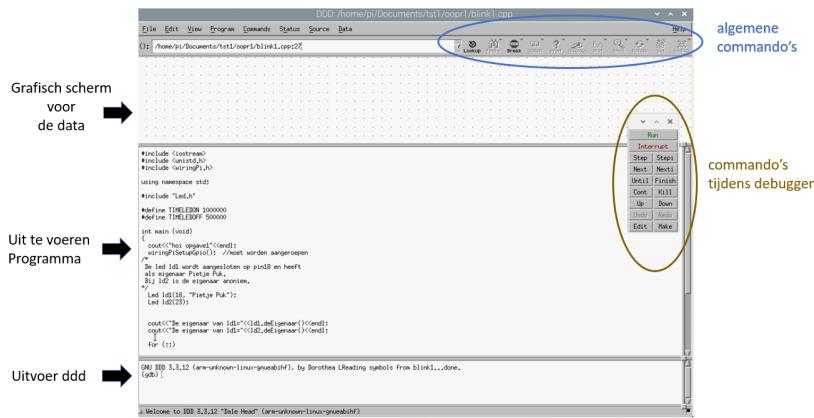


Figure 1.11: het DDD opstartschermscherm.

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

- Als eerste een korte kennismaking met de DDD debugger.
 - Ga met de cursor op de regel `Led ld1(18, "Pietje Puk");` (regelnummer 22)
 - Klik op *Break* (bij algemene commando's). Voor het begin van de regel verschijnt nu een breakpoint.
 - Klik op *Run* (bij debug commando's), het programma wordt uitgevoerd tot het breakpoint. Er verschijnt een groene pijl bij het breakpoint.
 - Klik op *Next*, de regel wordt uitgevoerd.
De groene pijl komt voor de regel `Led ld2(23);` te staan.
 - Klik op *Step*, met dit commando *Step* wordt in de constructor van de klasse `Led` gestapt. Doorloop het programma verder stap voor stap.
- Het zichtbaar maken van de inhoud van de objecten.
 - Start de DDD debugger op (`ddd blink1`):

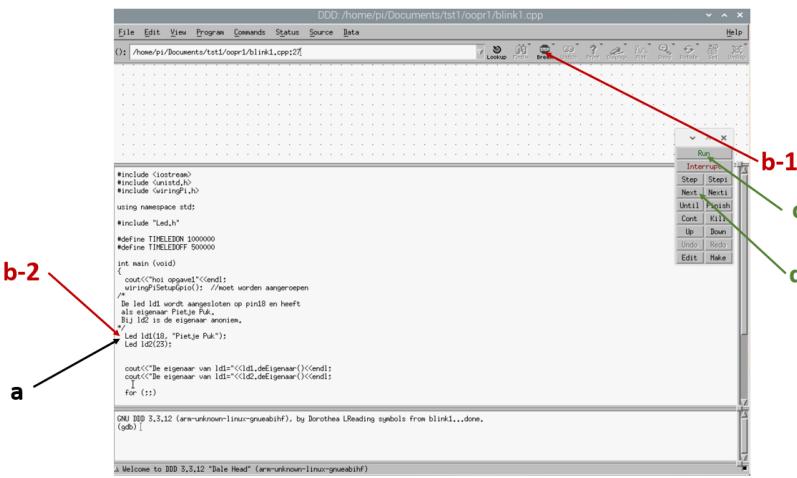


Figure 1.12: het zetten van een breakpoint.

- ii Zet een breakpoint op de regel `ld1.zetAan();`
- iii Klik met de muis op variabele `ld1` en vervolgens op Display of klik met rechtermuisknop en vervolgens op Display `ld1`.
- iv Doe hetzelfde met variabele `ld2`.
- v Als het goed is ziet de debugger er ongeveer uit zoals Figuur 1.13, alleen met je **eigen naam**.
- vi Ga met het Step commando de methode `zetAan` van `l1` in.
- vii Ga met het Step commando de methode `zetAan` van `l2` in. Je ziet dat de methode hetzelfde is, alleen de attributen hebben een andere waarde.
- c Maak een screenshot die lijkt op Figuur 1.13 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.

```

File Edit View Program Commands Status Source Data Help
() 1d2
1: 1d1
kleur = "Rood"
pinNr = 18
eigenaar = "Pietje Puk"
status = 0

2: 1d2
kleur = ""
pinNr = 23
eigenaar = "Anoniem"
status = 1

cout<<"hoi opgave1" << endl;
wiringPiSetupGpio(); //moet worden aangeroepen
/*
De led 1d1 wordt aangesloten op pin18 en heeft
als eigenaar Pietje Puk.
Bij 1d2 is de eigenaar anoniem.
*/
Led 1d1(18, "Rood", "Pietje Puk");
Led 1d2(23);

cout<<"De eigenaar van 1d1=" << 1d1.deEigenaar() << endl;
cout<<"De eigenaar van 1d1=" << 1d2.deEigenaar() << endl;

for (;;)
{
    1d1.zetAan();
    1d2.zetUit();
    usleep(TIMELEDON);
    1d1.zetUit();
}

```

Figure 1.13: Weergave van twee objecten van de klasse Led.

Appendices

A Tips en trucks in een linux omgeving

A.1 commando's

Commando's	
ls	listing
cd	change directory
mkdir	make directory

A.2 tips

B Installeren van de practicum omgeving

B.1 De VNC viewer

Het installeren van de [VNC viewer](#) gebeurt op de laptop. Je moet echter wel een standaard instelling van de PI veranderen.

1. sudo raspi-config
 - Kies 3 Interface Options.
 - P3 VNC
 - Yes
 - Kies 2 Display Option.
 - D5 VNC Resolutie of D1 Resolution
 - DM Mode 85 1280x720 of 1280x1024
2. Start de VNC viewer op, op je laptop. Vul in het scherm *VNC CONNECT*, zie figuur [B.1](#), het IP adres van de Raspberry PI in. Als het goed is krijg je de



Figure B.1: Connect met de PI maken.

grafische omgeving van de PI te zien.

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 de Raspberry Pi. Dit idee is te zien in figuur [B.2](#). Op de laptop draait visual studio code(VSC) en de source file is de file `opg1.cpp` op de PI. Het compileren kan via VSC, maar kan ook via de terminal in VCS of via een externe terminal b.v windows PowerShell, putty of een ander.

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:

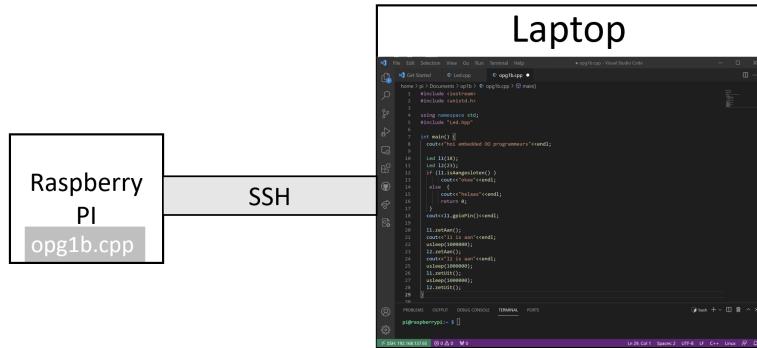


Figure B.2: Visual Studio Code en de PI.

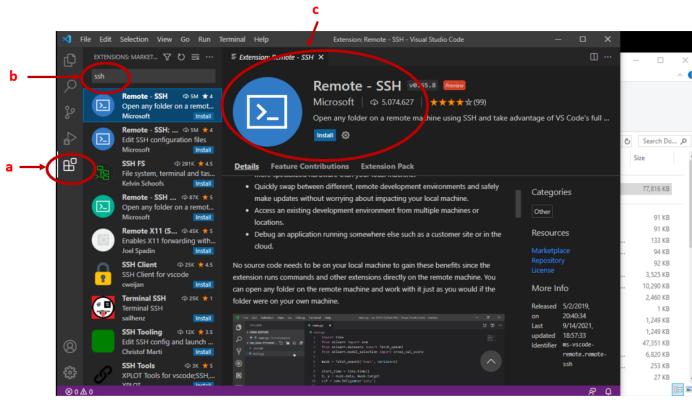


Figure B.3: Installeren van Remote SSH in VCS.

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

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

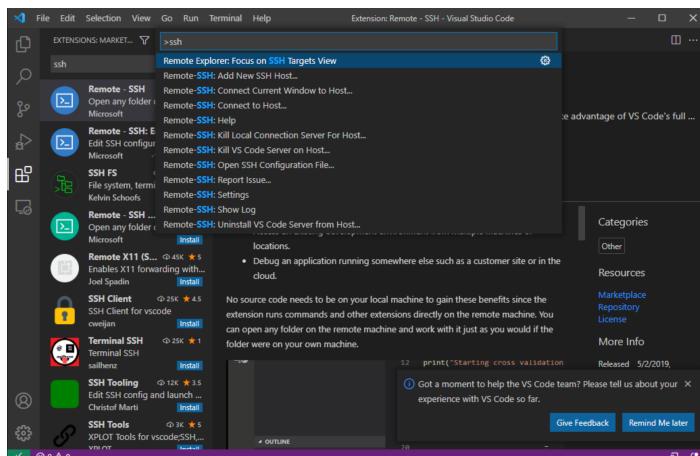


Figure B.4: SSH verbinding naar de Host.

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

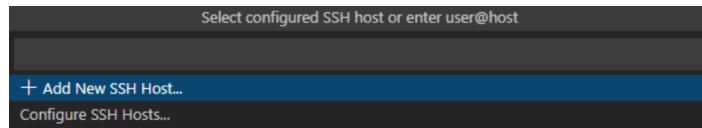


Figure B.5: toevoegen van een host.

- (b) klik op *+ Add New SSH Host*. VSC vraag vervolgens om een ssh command: type in: `ssh pi@ip.nr.` van de host zoals te zien is in figuur B.6. Uiteraard wel met je eigen IP nummer. VSC wilt de gegevens opslaan en vraagt vervolgen

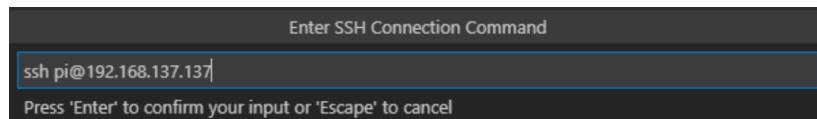


Figure B.6: een connectie maken met de host.

waard deze opgeslagen moet worden, zoals te zien is in figuur B.7.

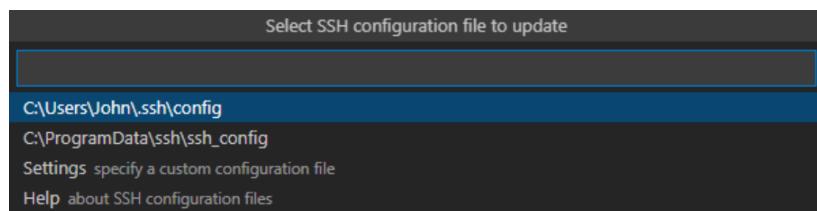


Figure B.7: waar de gegevens moeten worden opgeslagen.

VCS 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, VCS komt vervolgens met de vraag of je zeker wilt dat de doorgaat, zoals figuur B.8 laat zien.

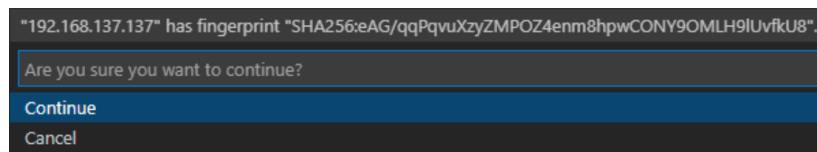


Figure B.8: waar de gegevens moeten worden opgeslagen.

- (c) Klik op *Continue*. VCS vraagt om een password zoals figuur B.9 laat zien. Het password is: *embedded*

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 VCS met een ssh verbinding naar de PI, zoals figuur B.11 laat zien.

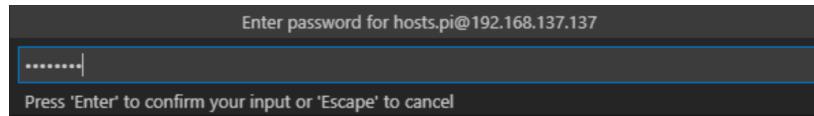


Figure B.9: invullen van een password.

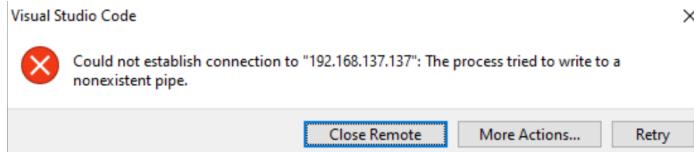


Figure B.10: een foutmelding.

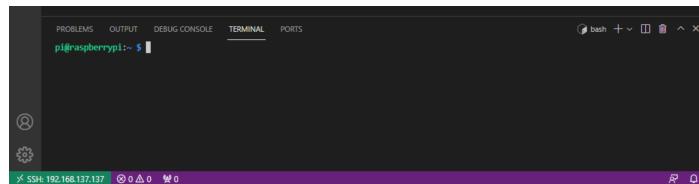


Figure B.11: Een geopende terminal in VSC.

- i. Met het *ls* commando vraag je de listing op van de huidige directory.
 - ii. Met het *pwd* commando wordt het pad van de huidige directory zichtbaar.
pwd <enter> → /home/pi.
 - iii. Met het *cd* commando wordt naar een directory gegaan.
cd Documents <enter> → **pi@raspberrypi:~/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/pi/Documents/intro
4. Het eerste programma op de PI.
 - (a) VSC werkt voornamelijk met directory's (folders). Het openen van een folder in VSC op de PI kan door te klikken op Explorer of CTRL + Shift + E. VSC opent een scherm zoets als figuur B.12. Ga naar de directory */home/pi/Documents/intro* en klik op oké. VSC kan vragen of je de auteur vertrouwt. Klik op Yes.
 In het *EXPLORER* veld verschijnt vervolgens de werkdirctory, zoals in figuur B.13 te zien is.
 - (b) Het programma van listing B.1 kan: gecreëerd worden door een nieuwe file *hoi.cpp* aan te maken, gedownload worden van Brighspace of gecloned worden van github.
 - Klik op New File en geef de filenaam *hoi.cpp*. Type/kopieer de tekst van Listing B.1 in de file *hoi.cpp*

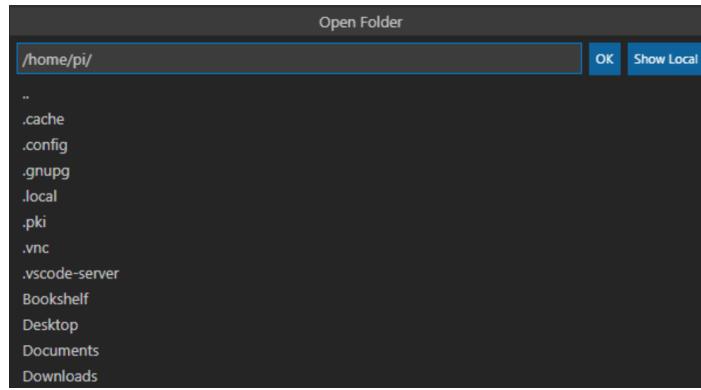


Figure B.12: De directory die geopend moet worden.

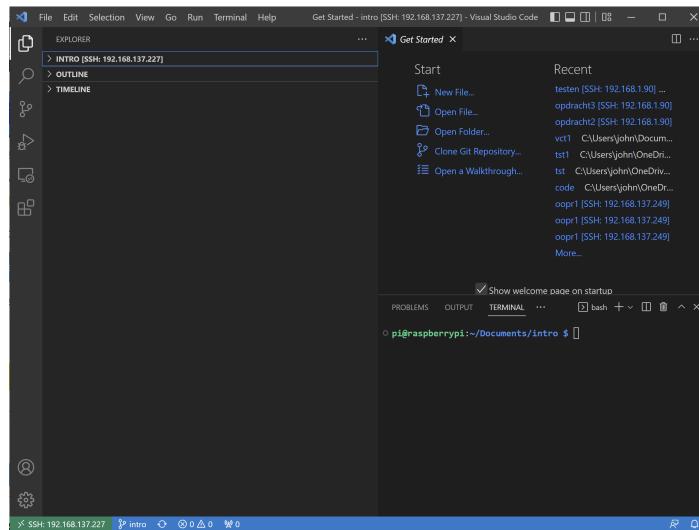


Figure B.13: In het EXPLORER veld is de werkdirctory te zien.

```
#include <stdio.h>
int main() {

    printf("Hoi_programmeurs_van_de_wereld\n");
    return 0;
}
```

Listing B.1: het meest gebruikte voorbeeld programma.

- 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.
 - Via git: `git clone - -branch intro https://github.com/Johnnny63Vi/oopr1.git1` Het nadeel van deze methode is dat een directory oopr1 wordt aange- maakt waarin de file komt te staan.
- (c) Open de bestaande terminal of open een nieuwe terminal *Cntr + Shift + ‘.* VSC opent een terminal in de werkdirctory. compileer de file *hoi.c* met de *g++* compiler(*g++ hoi.c*) en run het gecompileerde programma *./a.out*.

¹bij uitvoering geen spatie tussen - -

Uiteraard kan ook gekozen worden uit 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) Als eerste wordt de C++ omgeving in Visual Studio Code geïnstalleerd.
 - 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 tab-blad 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 file`.

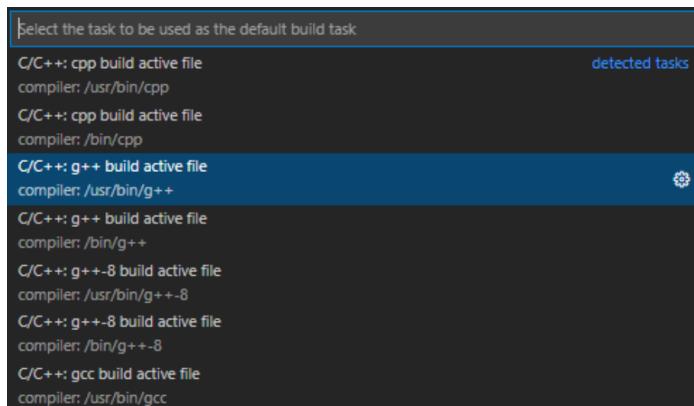


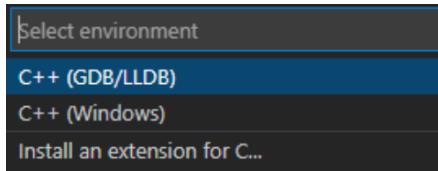
Figure B.14: Het kiezen van de compiler.

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

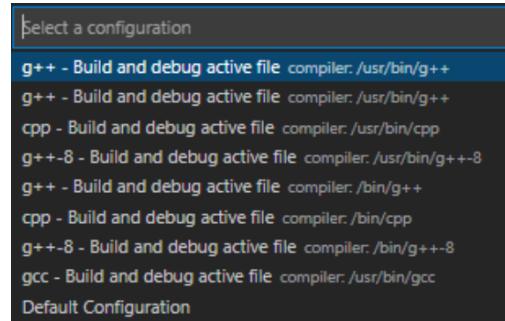
- (d) Activeer opnieuw hoi.cpp tab. Ga naar Terminal → Run Build Task... of *Ctrl+Shift+B*. VSC zal nu *hoi.cpp* compileren.
 - (e) Het gecompileerde programma kan nu gerund worden in b.v. een terminal. Open een terminal of maak een nieuwe terminal aan *Ctrl+Shift+T* en run *hoi* (`./hoi`).
- ## 6. Run/Debug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

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

- (a) Activeer he 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++`.



a Keuze welke compiler



b Keuze versie compiler

Figure B.15: Keuze compiler te gebruiken door VSC

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 opent.

- (c) Klik op tab-blad 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 PI uitgevoerd en stopt op regel 4, zoals te zien is in Figuur B.16. Met behulp van de debugknoppen kan stap voor stap door het programma gelopen worden.

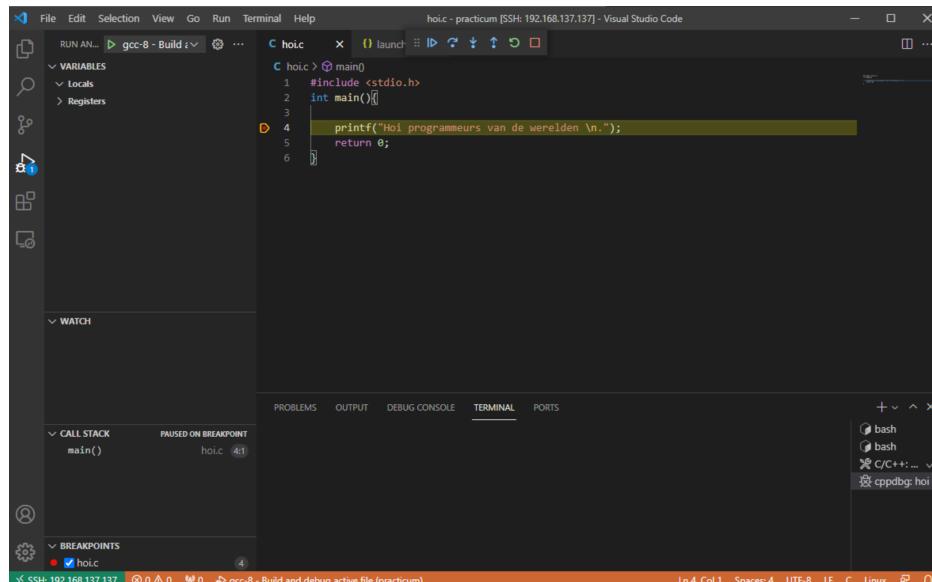


Figure B.16: Debuggen met VSC.

voor stap door het programma gelopen worden.