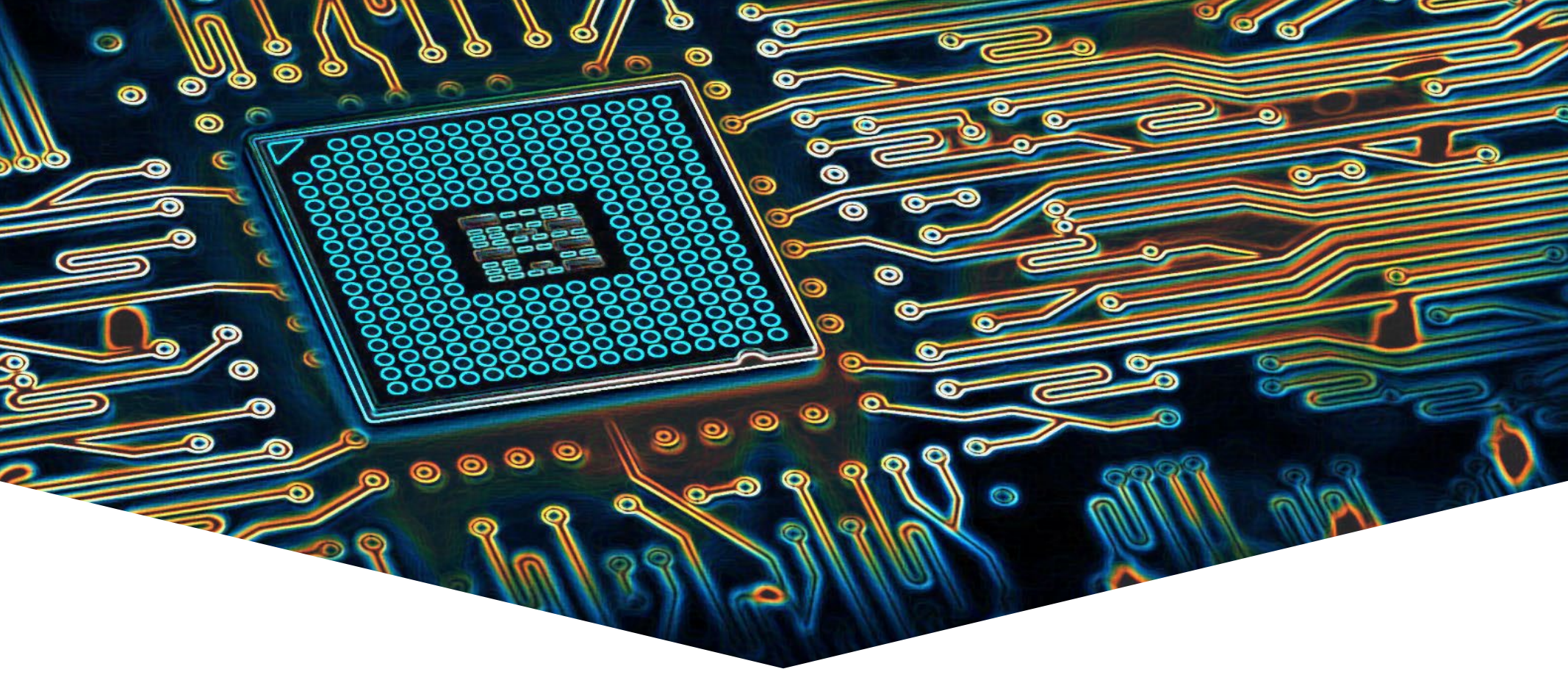


EMS31 Introductie

Embedded Systems: Jouw toekomst?

Vacature HBO Elektrotechniek (Embedded Systems):

- Bare-metal C/C++ programmeren (en documenteren) op embedded systemen;
- Communicatieprotocollen opstellen en vastleggen;
- Vertalen van (bestaande) schema's naar een pcb-ontwerp;
- Ervaring met hardware (op componentniveau) gekoppeld aan embedded systemen;
- Kunnen werken met multimeter en oscilloscoop;

Bron: https://toptech.nl/vacatures/Software/Embedded-Software-Engineer/2854

Embedded Systems: Jouw toekomst?

Vacature HBO Elektrotechniek (Embedded Systems):

- Een afgeronde HBO of universitaire opleiding bij voorkeur in Embedded Systems (Engineering) of (Technische) Informatica.
- Kennis van:
  - Meerdere programmeertalen (o.a. C, C#) op verschillende platformen (o.a.Windows, Linux) en moderne microcontrollers met of zonder RTOS;
  - Linux en C++ is een pre;
  - (Embedded) Technologieën o.a. threading
  - protocollen en (veld)bussen o.a. I^2C, SPI, RS485, CAN, TCP/IP;
  - relevante tools, zoals bijvoorbeeld Visual studio, Eclipse, Git;

- Kennis van en ervaring met verschillende ontwikkelmethoden (Agile, Scrum) en ontwerptechnieken (UML, Design Patterns, OOP) en versiebeheersystemen zijn een pre.

Bron: https://www.brunelco.nl/vacature-senior-embedded-software-engineer/

Embedded Systems: Jouw toekomst?

Vacature HBO Elektrotechniek (Embedded Systems):

- Je hebt een afgeronde technische opleiding zoals (Technische) Informatica, Computer Science, Elektrotechniek, Mechatronica of Embedded Systems op hbo- of wo niveau;
- Je hebt ervaring met en kennis van:
  - Embedded development in C en C++;
  - Embedded systems;
- Het is geen vereiste, maar je hebt een streepje voor als je kennis hebt van:
  - Scrum / Agile werken
  - (Embedded) Linux en Bare-metal systemen
  - ARM development
  - Kennis van elektronica om solide device drivers en networkinterfaces te ontwikkelen

Bron: https://www.alten.nl/career-nl/139-embedded-software-engineer/

Embedded Systems: Jouw toekomst?

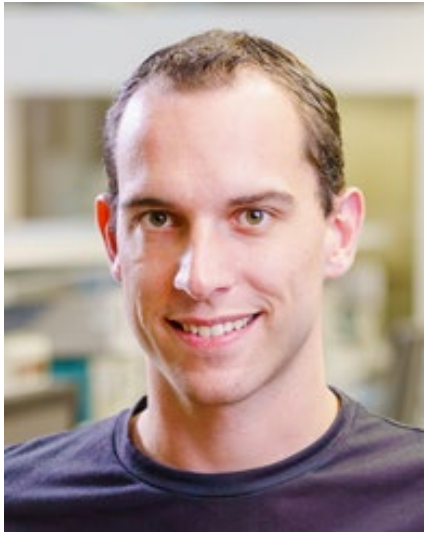
Vacature HBO Elektrotechniek (Embedded Systems):

- Master of Bachelor in Computer Science / Electronics
- Ervaring met C/C++
- Ervaring met OO design en design patterns
- Ervaring met units test / TDD
- Ervaring met Code coverage / review / analyses / version tooling
- Kennis van BSPs en driver ontwikkeling is een plus
- Kennis van RealTime / Multi-Threaded systemen is een plus
- Kennis en ervaring met Agile principes en manier van werken
- Affiniteit met hardware

Bron: https://www.dekimo.com/jobs/consultancy/experts-delft/senior-embedded-developper/

Docenten EMS31

Daniël
Versluis
VersD@hr.nl



Roy
Bakker
BaRoy@hr.nl

Harry
Broeders
BroJZ@hr.nl

Leerdoelen EMS31

Leerdoelen EMS31.

Je leert in week **kwartaal 3**:

- hoe je een betere **C**-programmeur wordt.

Je leert in week **kwartaal 4**:

- de basisbeginselen van objectgeoriënteerd programmeren in **C++**;
- de basisbeginselen van objectgeoriënteerd ontwerpen met **UML**.

Leerdoelen EMS31 kwartaal 3

#	Niveau	Weging	De student is in staat om …
1	C	5 %	… versiebeheer toe te passen (GIT) zodanig dat de student in groepsverband software revisies kan beheren.
2	C	25 %	… een dynamische datastructuur te implementeren in C met behulp van pointers en de functies `malloc` en `free`.
3	C	20 %	… de kwaliteit van code te verbeteren door het toepassen van unit testen, code coverage en coding standards.

Toetsing en studielast EMS31

Toets	Leerdoelen	Weging	Deadline
Opdracht 1	1, 2 en 3	50 %	Lesweek T3, zondag 23.59 uur
Opdracht 2	4, 5 en 6	50 %	Lesweek T4, zondag 23.59 uur

Je werkt in **tweetallen** samen m.b.v. git (*je krijgt van ons drie repositories*) en levert je werk ook in op die repositories.

Werkvorm	Omschrijving	Studielast
Theorielessen	Theorie volgen. In kwartaal 4	8 klokuren
Practicum	Begin maken met de labopdrachten.	32 klokuren
Zelfstudie	Bestuderen van het studiemateriaal. Uitwerken van alle labopdrachten.	100 klokuren

EMS31 levert je **5** studiepunten op. Dat betekent dat je 5/30 = 1/6 van de week → 40/6 = **6,5 uren / week** aan EMS31 dient te besteden.

Planning kwartaal 3

Week	Werkvorm	Beschrijving
3.1	Les Zelfstudie	Modules in C Zie paragraaf 2
3.2	Les Zelfstudie	Unittesten en Git submodules Zie paragraaf 2
3.3	Les Zelfstudie	Dynamisch geheugenallocatie Zie paragraaf 2
3.4	Les Zelfstudie	Trunk-based development met Feature flags Zie paragraaf 2
3.5	Les Zelfstudie	Memory tests en test coverage Zie paragraaf 2
3.6	Les Zelfstudie	Coding standards en static code analysis Zie paragraaf 2
3.7	Les Zelfstudie	Werken aan opdracht 1
3.8	Les Zelfstudie	Werken aan opdracht 1
T3	**Opdracht 1**	Deadline zondag 23.59 uur

Studiemateriaal vind je op de wiki:

HR_ELEKTRO / Embedded / EMS31

Wiki

Create page

Clone wiki

EMS31 / Home

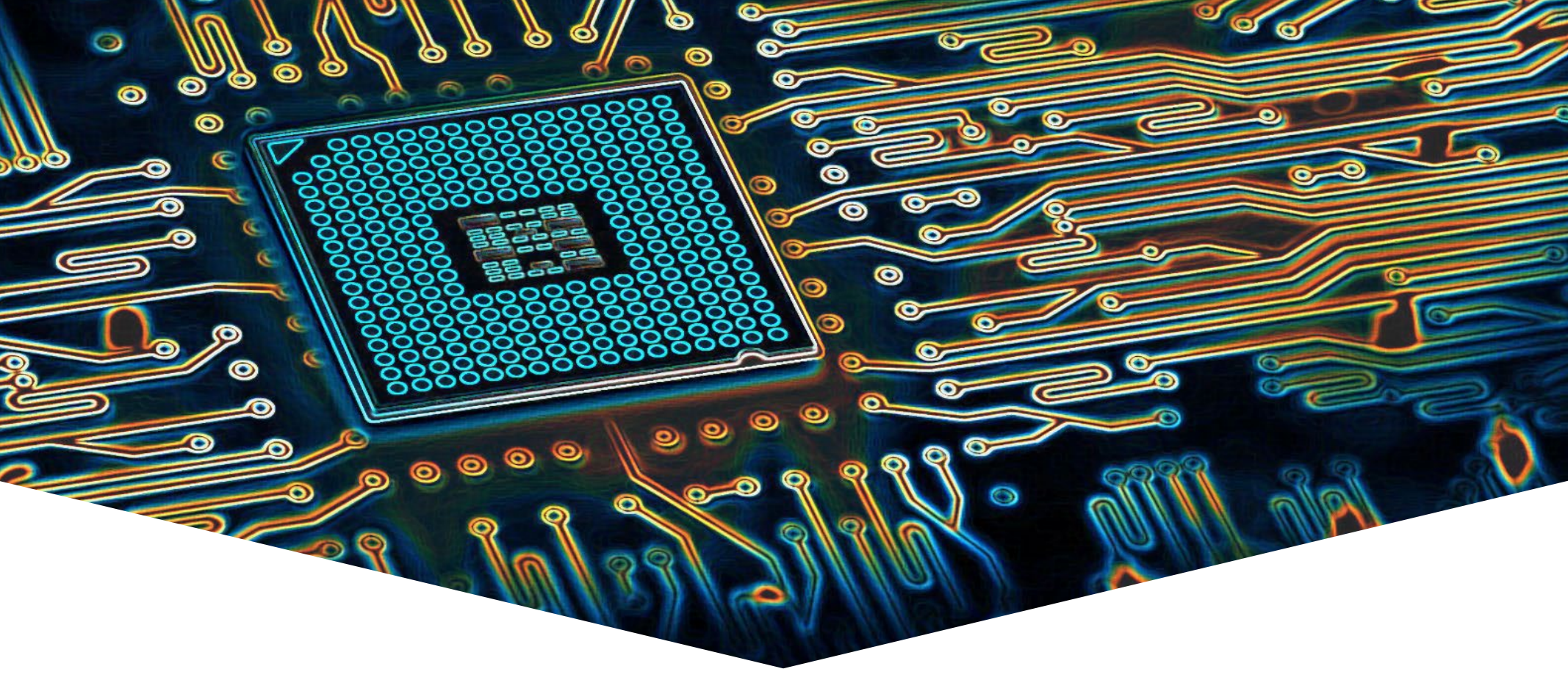
View | History | Edit

EMS31 - Embedded Systems 3

Dit repository is bedoeld voor studenten en docenten van de opleiding Elektrotechniek van de Hogeschool Rotterdam en wordt gebruikt het studiemateriaal voor de cursus "EMS31 - Embedded Systems 3" te verspreiden.

Let op! Deze wiki is nog niet volledig voor studiejaar 2023-2024.

De informatie in dit repository is zoals alle mensenwerk niet foutloos, verbeteringen en suggesties zijn altijd welkom! Maak als je ons feedback wilt geven een issue aan.



EMS31 Kwartaal 3 Week 1: Modules in C

Modules in C

Leerdoelen kwartaal 1 week 1. Je leert hoe je:

- de interface van een module in een `.h` bestand kunt declareren;
- de implementatie van een module in een `.c` bestand kunt definiëren;
- deze module kunt gebruiken en testen op een pc;

In **week 2** leer je onder andere hoe je:

- deze module kunt gebruiken en testen op een CC3220S Launchpad.

Voorbeeld van een module in C

Als voorbeeld bekijken we een module `breuk`.

- Waarom zou je breuken willen gebruiken i.p.v. `double`'s?

Module breuk

breuk.h

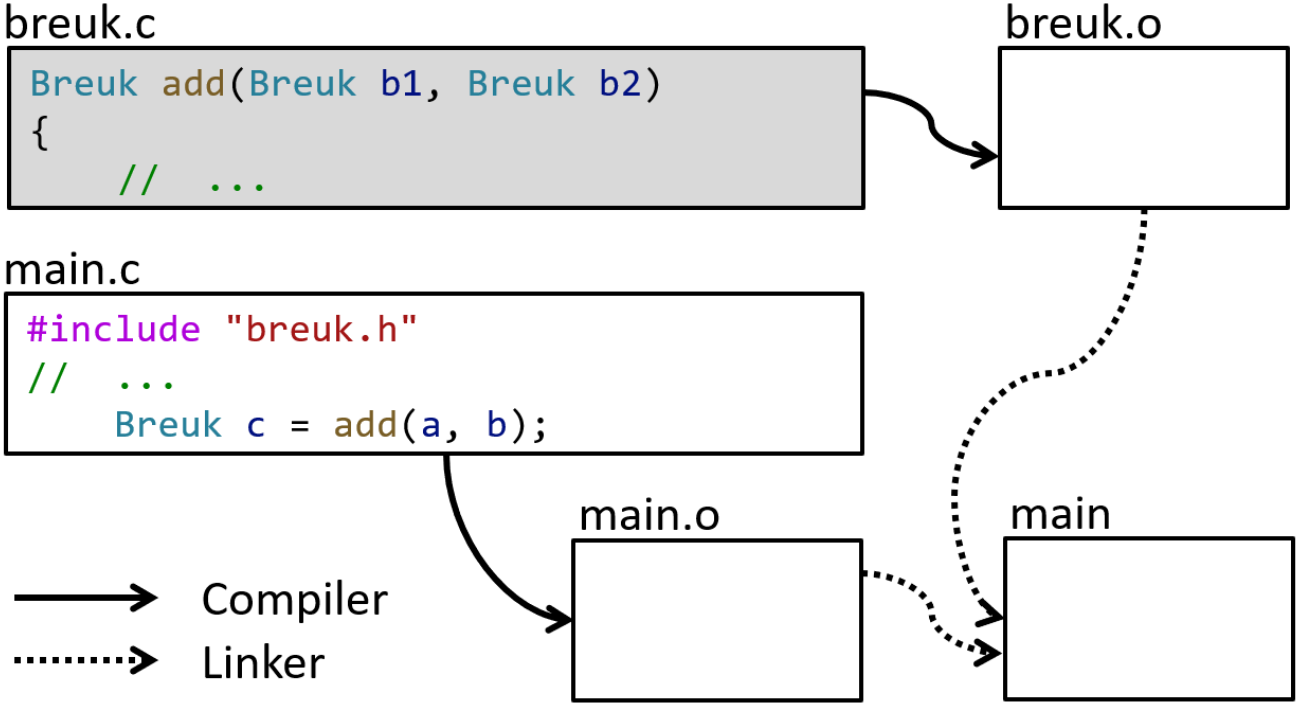
```
typedef struct {/* ... */} Breuk;
extern Breuk add(Breuk b1, Breuk b2);
extern Breuk mul(Breuk b1, Breuk b2);
```

breuk.c

```
Breuk add(Breuk b1, Breuk b2)
{
    //  ...
```

main.c

```
#include "breuk.h"
//  ...
    Breuk c = add(a, b);
```



Module breuk

We splitsen de module in een `.h` en een `.c` bestand:

- `.h` bestand bevat de definitie van alle types (`Breuk`) en de declaratie van alle functies (prototypes) (`add` en `mul`).
- `.c` bestand bevat de implementatie van alle functies.
- Het `.h` bestand is nodig om code die de module gebruikt te compileren, het `.h` bestand moet ge-include worden.
- Het `.c` bestand (of een gecompileerde versie daarvan) is nodig om code die de module gebruikt te linken (tot een executable), het gecompileerde `.c` bestand moet meegelinkt worden.

breuk.h


```c
#ifndef _HR_BroJZ_Breuk_
#define _HR_BroJZ_Breuk_

typedef struct
{
    int teller;
    int noemer;
} Breuk;

extern Breuk add(Breuk b1, Breuk b2);
extern Breuk mul(Breuk b1, Breuk b2);

#endif
```

Include guard voorkomt problemen bij meerdere keren includen in dezelfde `.c` file

breuk.c


```c
#include <assert.h>
#include "breuk.h“

static Breuk normaliseer(Breuk b)
{
    assert(b.noemer != 0);
    // ...
    return b;
}

Breuk add(Breuk b1, Breuk b2)
{
    Breuk som;
    som.teller = b1.teller * b2.noemer + b1.noemer * b2.teller;
    som.noemer = b1.noemer * b2.noemer;
    return normaliseer(som);
}
// ...
```

`static` functie is alleen in deze `.c` file zichtbaar (is dus verborgen voor gebruikers van de module Breuk)

main.c

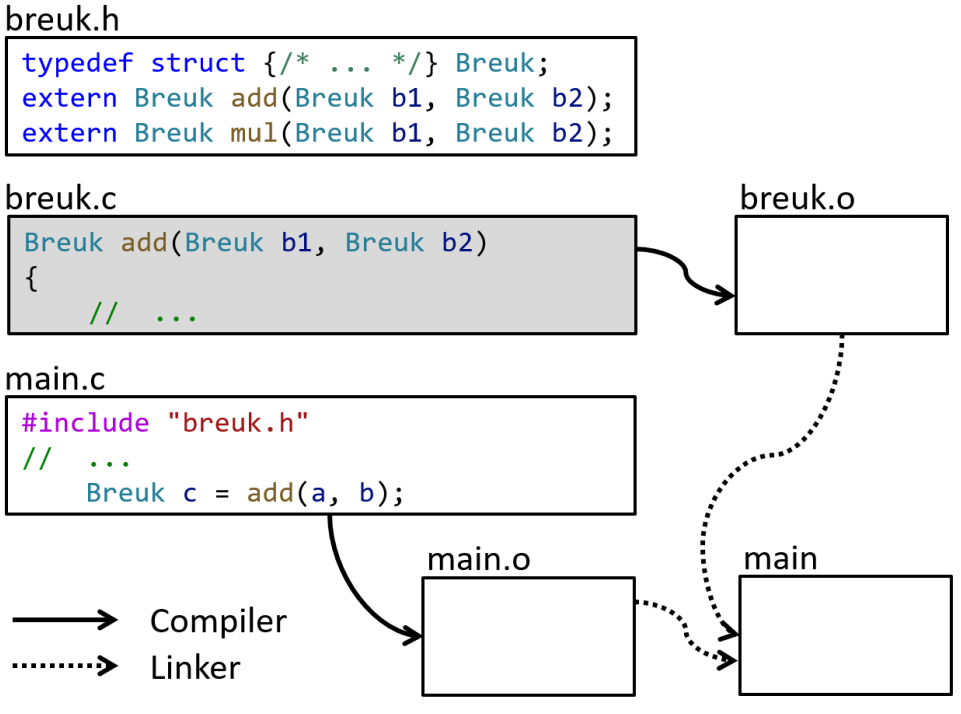

```c
#include <stdio.h>

#include "breuk.h"

int main(void)
{
    Breuk a = {-2, 4}, b = {6, -8};

    Breuk c = add(a, b);
    printf("c = %d/%d\n", c.teller, c.noemer);

    Breuk d = mul(a, b);
    printf("d = %d/%d\n", d.teller, d.noemer);

    return 0;
}
```

Output:

```
c = -5/4
d = 3/8
```

Compileren en linken


```
$ gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c breuk.c
$ gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c main.c
$ gcc breuk.o main.o -o main
$ ./main
c = -5/4
d = 3/8
```



Automatiseren build proces: make

makefile:

IDE's (CCS) gebruiken make onder de motorkap

target

dependencies

```
main : breuk.o main.o
tab gcc breuk.o main.o -o main
```

command

```
breuk.o : breuk.c breuk.h
    gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c breuk.c

main.o : main.c breuk.h
    gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c main.c
```

Veel meer mogelijkheden, zoek zelf maar uit:
http://www.gnu.org/software/make/manual/make.html

Automatiseren build proces: make

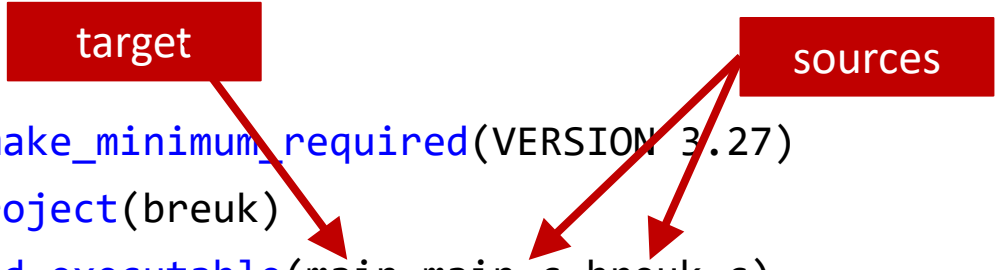

```
$ make
gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c breuk.c
gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c main.c
gcc breuk.o main.o -o main
$ touch main.c
$ make
gcc -std=c18 -Wall -Wextra -pedantic-errors -g3 -O0 -c main.c
gcc breuk.o main.o -o main
```


Zie: https://bitbucket.org/HR_ELEKTRO/EMS31/raw/master/Programmas/breuk.zip

Automatiseren build proces: CMake

Cmake genereerd makefile en bepaald dependencies

CMakeLists.txt:




```
cmake_minimum_required(VERSION 3.27)
project(breuk)
add_executable(main main.c breuk.c)
target_compile_options(main PRIVATE -std=c18 -Wall -Wextra -Wpedantic -g3 -O0)
```

Veel meer mogelijkheden, zoek zelf maar uit: https://cmake.org/

Automatiseren build proces: CMake


```
$ mkdir build
$ cd build
$ cmake ..
-- The C compiler identification is GNU 13.2.1
-- The CXX compiler identification is GNU 13.2.1
-- …
-- Build files have been written to: /home/ems/breuk/build
$ make
[ 33%] Building C object CMakeFiles/main.dir/main.c.o
[ 66%] Building C object CMakeFiles/main.dir/breuk.c.o
[100%] Linking C executable main
[100%] Built target main
```


Zie: https://bitbucket.org/HR_ELEKTRO/ems31/raw/master/Programmas/breuk.zip

CMake in Visual Studio Code

Run: Shift+F5

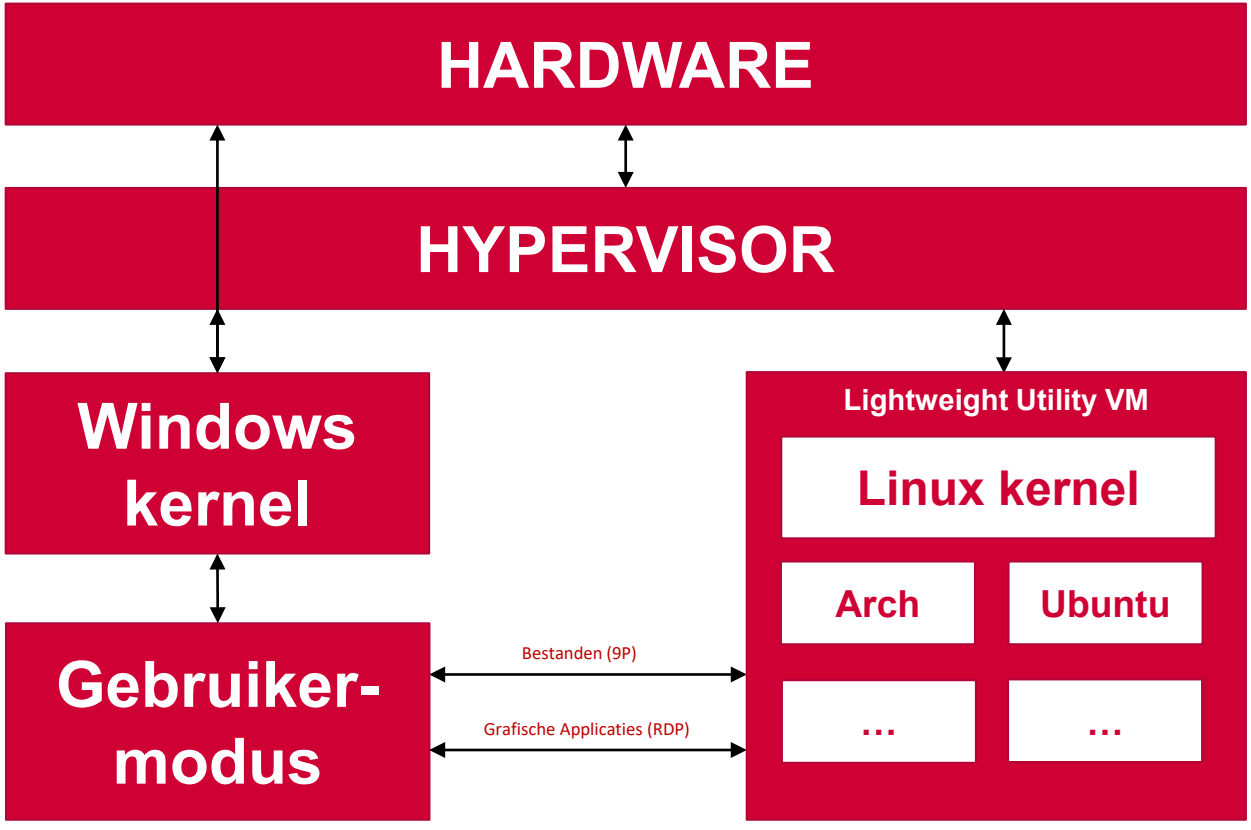
Eerst zelf een breakpoint zetten. Debug: Ctrl+F5

Build: F7

Errors en warnings: Ctrl+Shift+M

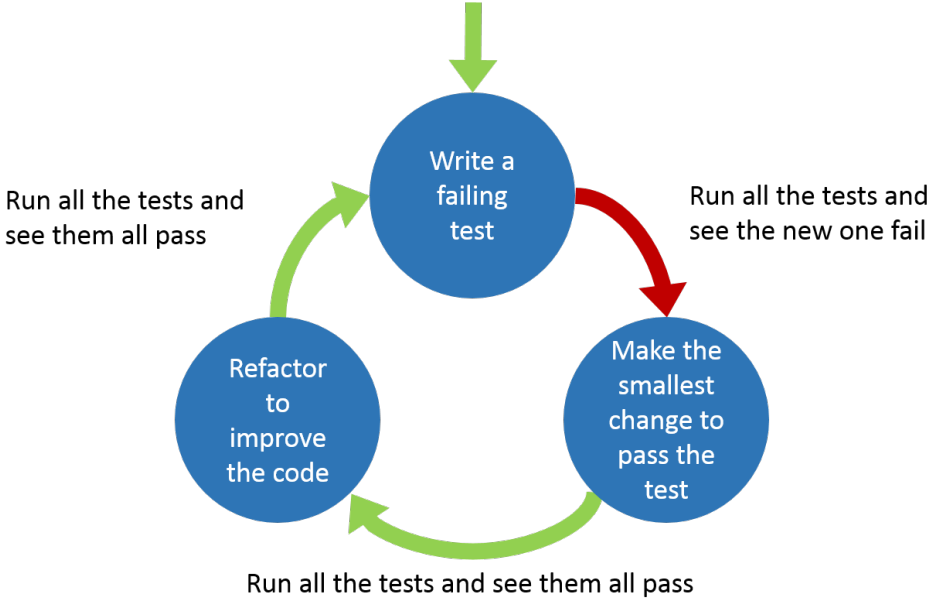
Zie: https://github.com/microsoft/vscode-cmake-tools/tree/main/docs

WSL


HARDWARE
HYPERVISOR
Windows kernel
Gebruiker-modus
Bestanden (9P)
Grafische Applicaties (RDP)
Lightweight Utility VM
Linux kernel
Arch
Ubuntu
...
...

Volgende les…

Unittesten en Git submodules



Bron: https://hiddeninplainsight.co.uk/post/embedded-tdd/

Aan de slag!

Aan de slag met [Opdrachten_Week_3.1.pdf]

HOGESCHOOL ROTTERDAM