

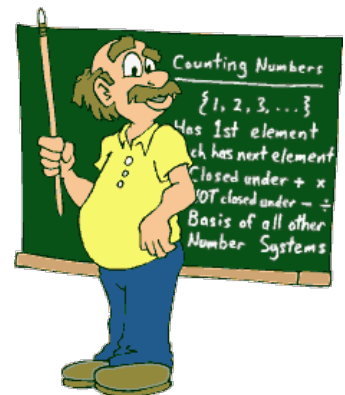


AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

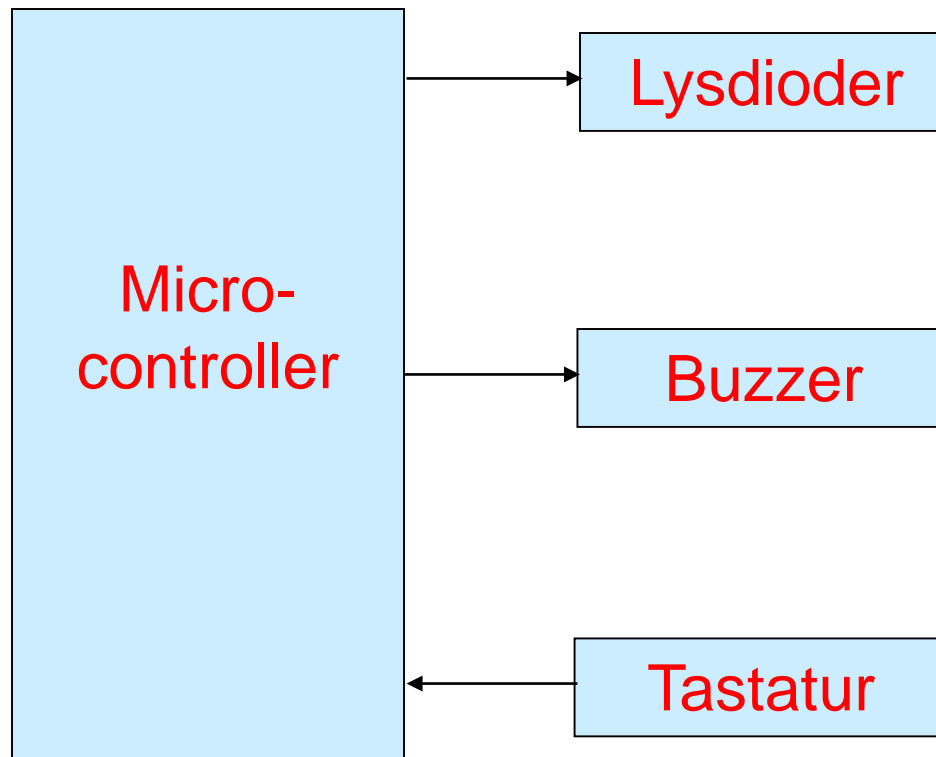
MSYS

Microcontroller Systems

Lektion 12 : Indkapsling



Design - eksempel



Ønsket funktionalitet (forenklet):

Ved tryk på taster på tastaturet, skal buzzeren afgive en tone i 1 sekund. Imens skal en bestemt lysdiode lyse.

Den ustrukturerede måde ("spaghetti")

```
void main()
{
    [Kode til klargøring af lysdioder]
    [Kode til klargøring af buzzer]
    [Kode til klargøring af tastatur]
    while(1)
    {
        if [Der er trykket på en tast]
        {
            [Start lydgiveren med den ønskede tone]
            [Tænd den ønskede lysdiode]
            [Vent 1 sekund]
            [Sluk lydgiveren]
            [Sluk alle lysdioder]
        }
    }
}
```

[...] er "pseudo"-kode



Problemet: Kode for forskellige H/W-enheder er i samme funktion og i samme fil. Rodet og uoverskuelig kode !

Bedre struktur: Opdeling i funktioner

```
void InitLysdioder()
{
    [Kode til klargøring af lysdioder]
}

void InitBuzzer()
{
    [Kode til klargøring af buzzer]
}

void InitTastatur()
{
    [Kode til klargøring af tastatur]
}

unsigned char TastTrykket()
{
    [Returner tastens kode, hvis der er trykket den]
    [- hvis der ikke nogen trykket: Returnerer 0]
}

void StartTone(unsigned char tast)
{
    [Kode, der starter tonen, som hører til "tast"]
}

void TaendLysdiode(unsigned char tast)
{
    [Kode, der tænder lysdioden, som hører til "tast"]
}

void StopTone()
{
    [Kode, der slukker lyd giveren]
}

void SlukAlleLysdioder()
{
    [Kode, der slukker alle lysdioder]
}
```

```
void main()
{
    unsigned char knap;

    InitLysdioder();
    InitBuzzer();
    InitTastatur();
    while(1)
    {
        knap = TastTrykket();
        if (knap)
        {
            StartTone(knap);
            TaendLysdiode(knap);
            [Vent 1 sekund]
            StopTone();
            SlukAlleLysdioder();
        }
    }
}
```

Gruppering efter HW

```
void InitLysdioder()
{
    [Kode til klargøring af lysdioder]
}
```

```
void InitBuzzer()
{
    [Kode til klargøring af buzzer]
}
```

```
void InitTastatur()
{
    [Kode til klargøring af tastatur]
}
```

```
unsigned char TastTrykket()
{
    [Returner tastens kode, hvis der er trykket den]
    [- hvis der ikke nogen trykket: Returnerer 0]
}
```

```
void StartTone(unsigned char tast)
{
    [Kode, der starter tonen, som hører til "tast"]
}
```

```
void TaendLysdiode(unsigned char tast)
{
    [Kode, der tænder lysdioden, som hører til "tast"]
}
```

```
void StopTone()
{
    [Kode, der slukker lydgiveren]
}
```

```
void SlukAlleLysdioder()
{
    [Kode, der slukker alle lysdioder]
}
```

```
void main()
{
    unsigned char knap;

    InitLysdioder();
    InitBuzzer();
    InitTastatur();
    while(1)
    {
        knap = TastTrykket();
        if (knap)
        {
            StartTone(knap);
            TaendLysdiode(knap);
            [Vent 1 sekund]
            StopTone();
            SlukAlleLysdioder();
        }
    }
}
```

— = Lysdioder

— = Buzzer

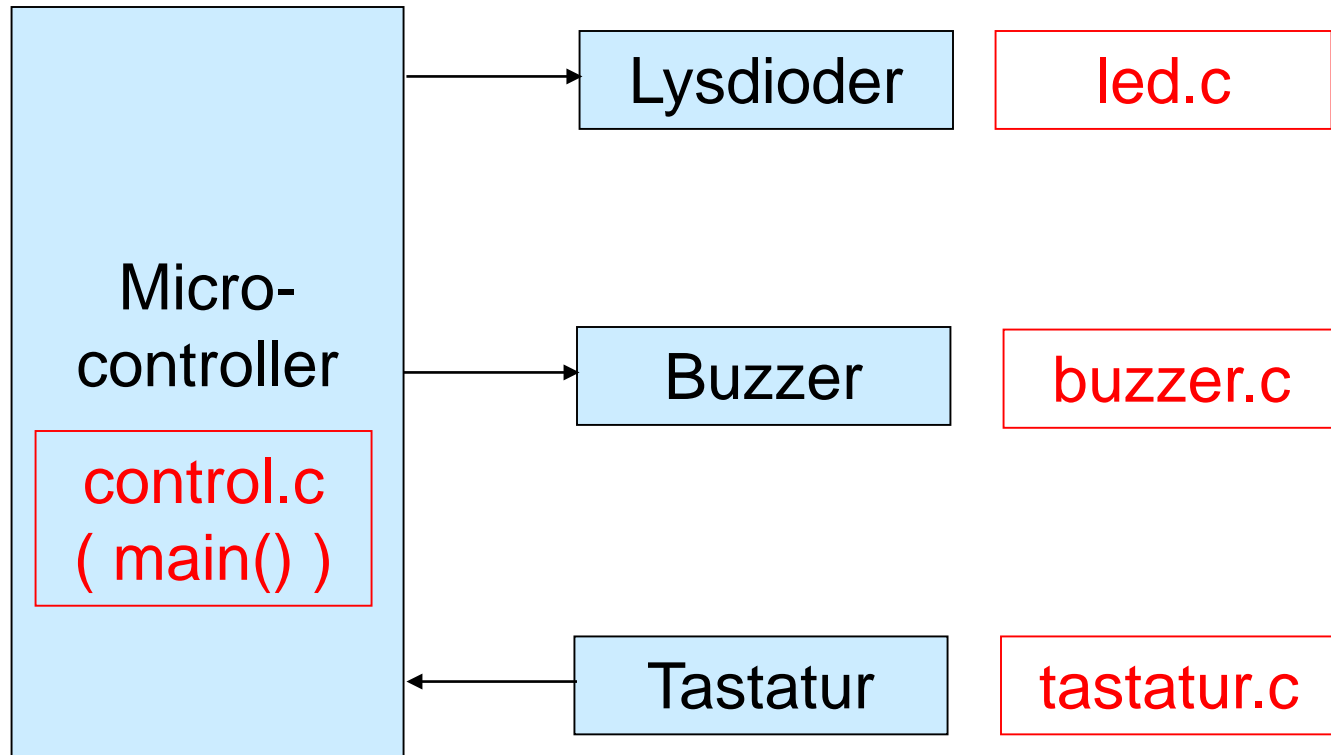
— = Tastatur



Modulering: En C-fil for hver HW-enhed

- Ved at skrive **en C-fil for hver HW-enhed** (gælder principielt også for SW-enheder), opdeler vi programmet i logiske blokke eller **moduler**.
- Bedre **overskuelighed** (= færre fejl).
- Bedre **vedligeholdelses-venlighed**.
- Mulighed for **genbrug** (en HW-enheds C-fil vil ofte kunne anvendes uforandret i et andet program). Skal derfor kun skrives (og testes) en gang for alle.
- **Indkapsling** (= beskyttelse) af data og funktioner.

Modul-opdeling (eksempel)



Lav os se på indholdet af
de 4 C-filer...

Filen "led.c"

```
void InitLysdioder()  
{  
    [Kode til klargøring af lysdioder]  
}  
  
void TaendLysdiode(unsigned char tast)  
{  
    [Kode, der tænder lysdioden, som hører til "tast"]  
}  
  
void SlukAlleLysdioder()  
{  
    [Kode, der slukker alle lysdioder]  
}
```



Filen "buzzer.c"

```
void InitBuzzer()  
{  
    [Kode til klargøring af buzzer]  
}  
  
void StartTone(unsigned char tast)  
{  
    [Kode, der starter tonen, som hører til "tast"]  
}  
  
void StopTone()  
{  
    [Kode, der slukker lydgiveren]  
}
```



Filen "tastatur.c"

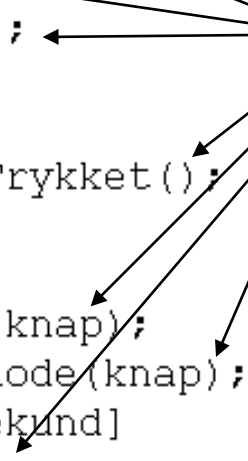
```
void InitTastatur()  
{  
    [Kode til klargøring af tastatur]  
}  
  
unsigned char TastTrykket()  
{  
    [Returner tastens kode, hvis der er trykket den]  
    [- hvis der ikke nogen trykket: Returnerer 0]  
}
```



Filen "control.c"

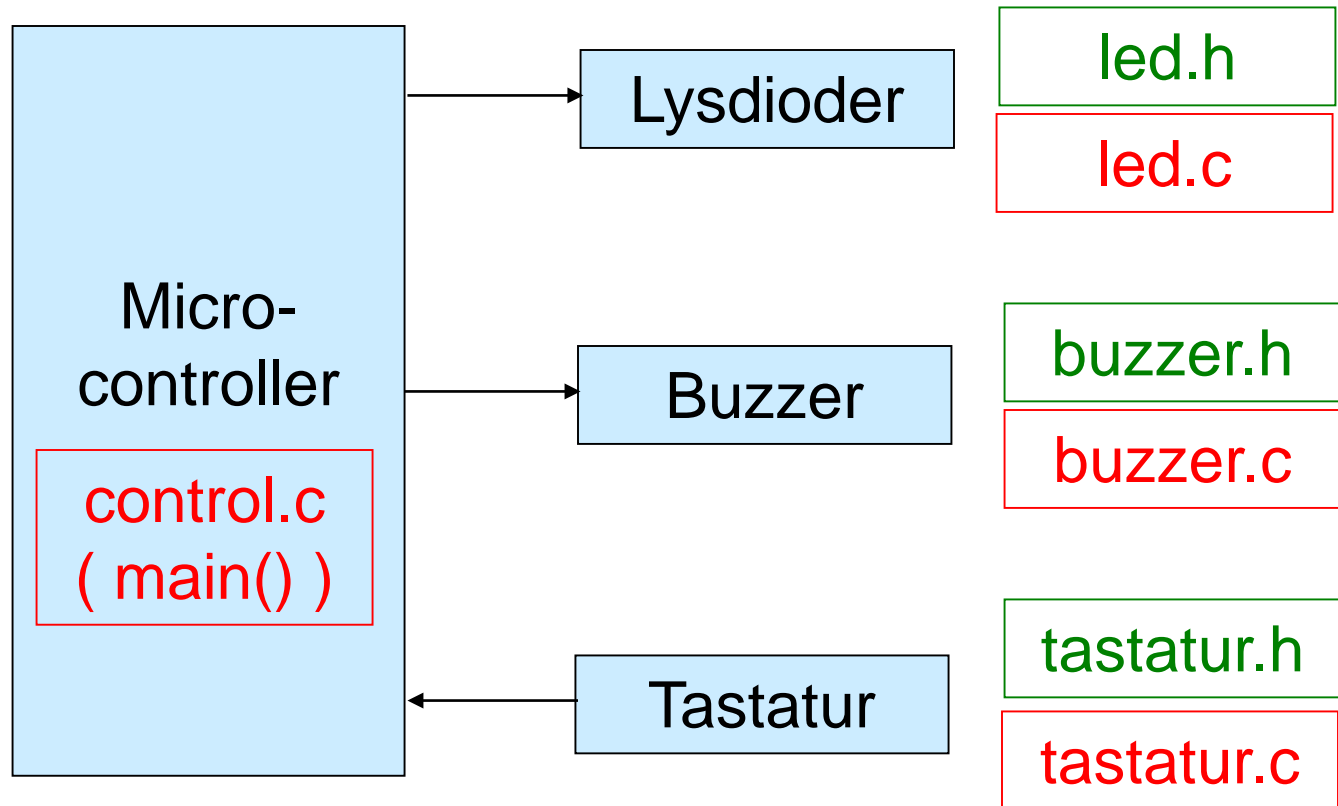
```
void main()
{
    unsigned char knap;

    InitLysdioder();
    InitBuzzer();
    InitTastatur();
    while(1)
    {
        knap = TastTrykket();
        if (knap)
        {
            StartTone(knap);
            TaendLysdiode(knap);
            [Vent 1 sekund]
            StopTone();
            SlukAlleLysdioder();
        }
    }
}
```

A red dot is positioned to the right of the code block. Five arrows originate from this dot and point to the following lines of code: 'InitLysdioder();', 'InitBuzzer();', 'InitTastatur();', 'StartTone(knap);', and 'SlukAlleLysdioder();'.

- De enkelte C-filer (her 4) **compileres en ad gangen**, hvorefter alt **"linkes"** til et program.
- **PROBLEM:** Når "control.c" oversættes, **kender** compileren **ikke prototypen** for de kaldte funktioner.
- Løsningen herpå er **"header-filer" (.h –filer).**

Header-filer (.h –filer)



Headerfilen indeholder prototyperne for de af modulets funktioner, der skal kunne kaldes fra andre moduler (her control.c).

Lav os se på indholdet af de 3 header-filer...



Indholdet af de 3 header-filer

"led.h":

```
// Prototyper for funktionerne i "led.c"  
void InitLysdioder();  
void TaendLysdiode(unsigned char tast);  
void SlukAlleLysdioder();
```

"buzzer.h":

```
// Prototyper for funktionerne i "buzzer.c"  
void InitBuzzer();  
void StartTone(unsigned char tast);  
void StopTone();
```

"tastatur.h":

```
// Prototyper for funktionerne i "tastatur.c"  
void InitTastatur();  
unsigned char TastTrykket();
```

Header-filerne
skal inkluderes
fra "control.c"....



Inkludering af headerfiler

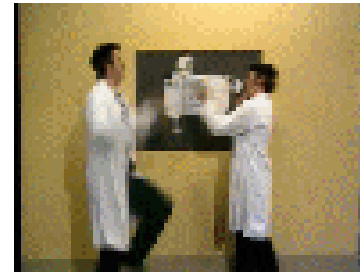
```
#include "led.h"  
#include "buzzer.h"  
#include "tastatur.h"
```

```
void main()  
{  
    unsigned char knap;  
  
    InitLysdioder();  
    InitBuzzer();  
    InitTastatur();  
    while(1)  
    {  
        knap = TastTrykket();  
        if (knap)  
        {  
            StartTone(knap);  
            TaendLysdiode(knap);  
            [Vent 1 sekund]  
            StopTone();  
            SlukAlleLysdioder();  
        }  
    }  
}
```

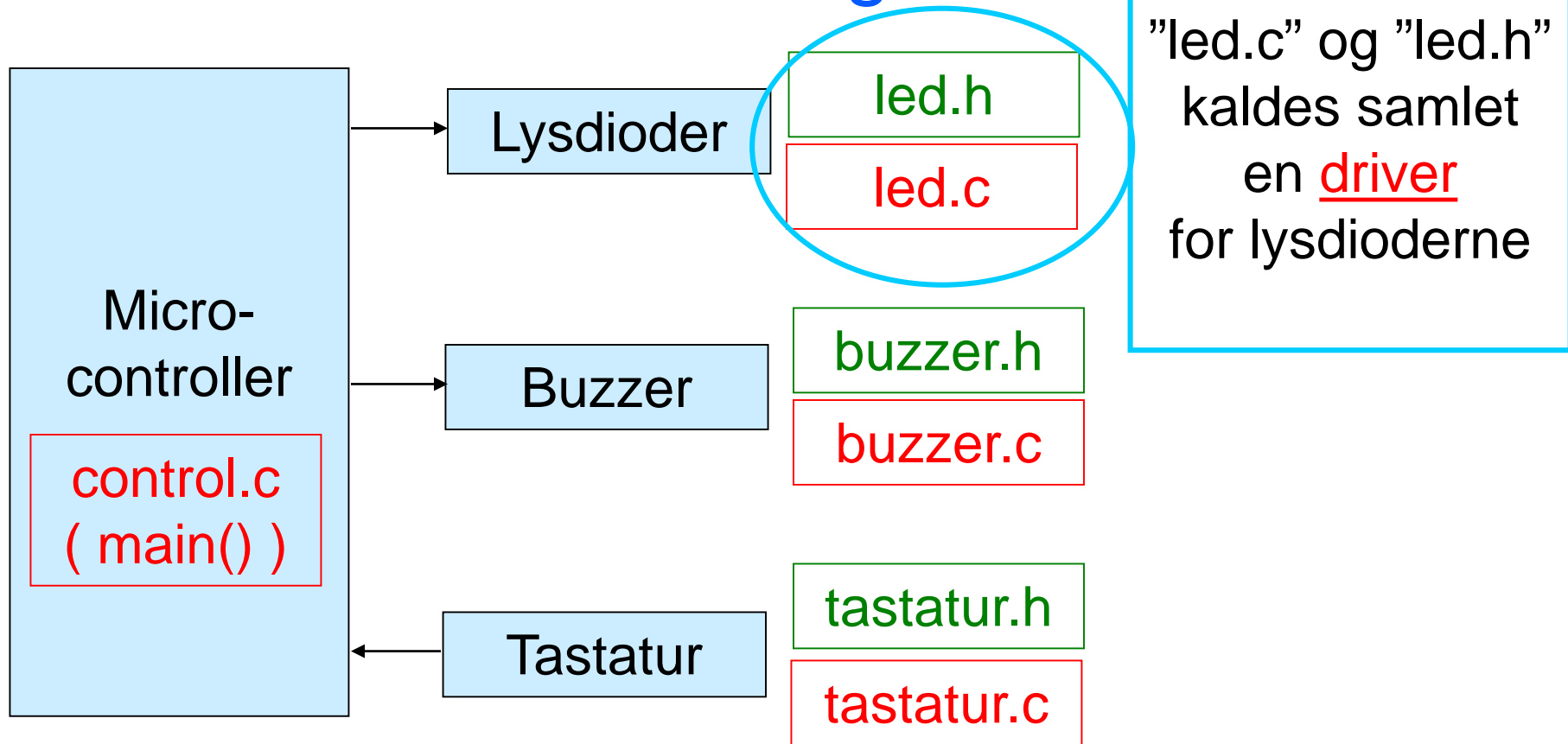
- **#include** af vores headerfiler.

Compileren kan nu oversætte "control.c".

Problemet er løst !



Driver - begrebet



"led.c" og "led.h"
kaldes samlet
en driver
for lysdioderne

"led.c" indeholder
implementeringen

"led.h" indeholder
grænsefladen

Slut på lektion 12

