

# EKSAMEN I INDLEJRET SYSTEMDESIGN 1

---

E3ISD1

Janus Bo Andersen (JA67494)

24. januar 2020 kl. 9.00

# EMNER

# 2 X 10-12 MIN. + VOTERING

---

## Hardware (Morten)

1. Start-up og memory
2. Peripheral I/O
3. Interrupts
4. Timers
5. Seriel kommunikation
6. Analog interfacing

---

## Software (Klaus)

1. C++: Klasser og OOP
2. C++: Funktioner, objekter, polymorphisme
3. C/C++: Make og Makefile
4. Beaglebone: Boot/shutdown, shell-scripts, I/O
5. Beaglebone: Udvikling, (cross-)compiling, debug
6. Beaglebone: Mikroprojekt

# HW1: STARTUP OG MEMORY

## INDHOLD

---

1. Globale og lokale variable: Forskelle på variabel scopes og levetid
  1. Memory map for KL25Z og placering af initialisering af variabler
    1. `data_init`, `bss_init`, efter `init`
  2. Brug af `volatile` qualifier
  3. Stack
    1. Stack pointer

### **1: Startup og memory**

1. Forklar forskellen på en ***global*** og en ***lokal*** variabel. Hvordan initialiseres hhv. globale og lokale variable, når processoren starter op.
2. Hvornår vil du bruge **`volatile`** sammen med oprettelsen af en variabel.
3. Hvad er **`stack`**, og hvordan bruges denne når du kalder en funktion, f.eks fra **`main()`**.

# HW1: STARTUP OG MEMORY

## GLOBALE VS LOKALE VARIABLE

---

Hvad er forskellen?

- Global (statisk) variabel:
  - Bruges til delt data, tilgængelig i alle scopes i en translation unit.
  - Brug `extern`, hvis den deklarereres i andre translation units (global på tværs).
  - Lifetime: Statisk, hele programmets levetid.
  - Brug af globale variable *kan* være "code smell", dvs. et tegn på dårligt design.
- Lokal variabel:
  - Lokal til et scope {}, fx en funktions scope `f(){...}`
  - Indkapslet, interfererer ikke med andre variable i andre scopes.
  - Lifetime: Scope (økonomisk, mindre fejlrisko).
  - Brug `static`, hvis værdien skal overleve efter scope forlades, men kun være tilgængelig fra scope.

Initialiseres forskelligt. Initialisering -> næste slides (men først kort lidt om memory map for Cortex-M0).

# HW1: STARTUP OG MEMORY

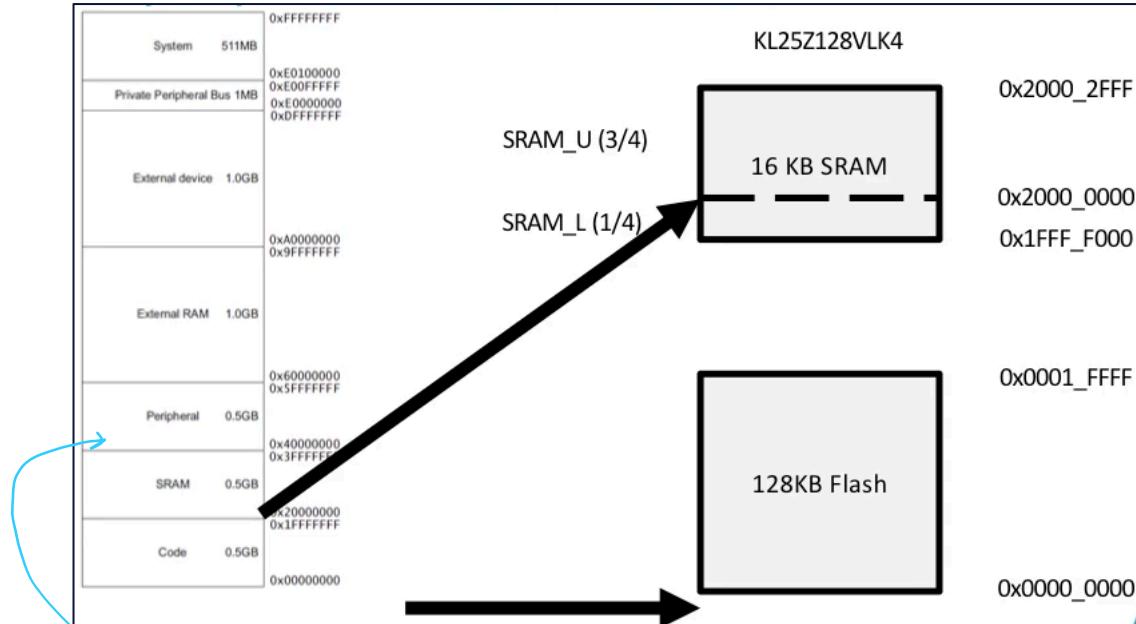
## MEMORY MAP FOR CORTEX-M0

Struktur:

- 32-bit processor ->  $2^{32}$  adresser ( $4.3e9 \rightarrow 4$  GB)
- Byte-addressable
  - Word = 32 bit => 1 word = 4 adresser
  - Endianness (!)
- Memory-mapped I/O (peripherals)

Blokke:

- "Code" består af
  - 128 KB Flash (permanent)
  - 4 KB SRAM (flygtigt)
- "SRAM" består af
  - 12 KB SRAM (rest. af de totale 16 KB)
- "Peripheral"
  - Adresser til fx GPIO-registre, osv.
- "Private Peripheral"
  - Til GPIO registre (direkte adgang ad "privat vej")



Behov for lagring:

- Kode
- RO statisk data
- RW statisk data
  - Initialiseret, 0-initialiseret, u-initialiseret.
- Stack
- Heap

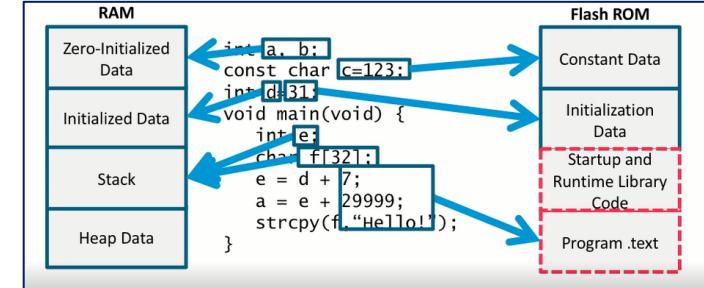
Absolute address (hex)	Register name	Width (in bits)	Access
4004_9000	Pin Control Register n (PORTA_PCR0)	32	R/W
4004_9004	Pin Control Register n (PORTA_PCR1)	32	R/W
4004_9008	Pin Control Register n (PORTA_PCR2)	32	R/W
4004_900C	Pin Control Register n (PORTA_PCR3)	32	R/W
4004_9010	Pin Control Register n (PORTA_PCR4)	32	R/W
4004_9014	Pin Control Register n (PORTA_PCR5)	32	R/W
4004_9018	Pin Control Register n (PORTA_PCR6)	32	R/W
4004_901C	Pin Control Register n (PORTA_PCR7)	32	R/W

+4

# HW1: STARTUP OG MEMORY INITIALISERING AF VARIABLE

Proces for initialisering ved processorens opstart drives af startup-filen (startup\_mkl25z4.c):

- ResetISR(), SystemInit() og så håndteres data...
- Eks.: Lokalisering følger C's memory model
  - Globale / lokale statiske:
    - Const variable er gemt i et read-only område (RO).
    - Med initial værdi -> datasektion (RW) -> håndteres med **data\_init**
    - Uden initial værdi -> bliver 0-init -> BSS-sektion (RW) -> **bss\_init**.
  - Lokale:
    - Automatiske: Stack (RW) (u/værdi)
    - Dynamiske: Heap (RW) (u/værdi)



```
1 #include <stdio.h>
2 #include <string.h>
3
4 int a, b;           // global, uninitialized
5 const char c=123;  // global const
6
7 int d=31;          // global, initialized
8
9 void main(void)
10{
11
12    int e;           // local, uninitialized
13
14    char f[32];      // local, uninitialized
15
16    e = d + 7;
17
18    a = e + 29999;
19
20    strcpy(f, "Hello!");
21
22 }
```

# HW1: STARTUP OG MEMORY

## DATA\_INIT

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int a, b;           // global, uninitialized
5 const char c=123;   // global const
6
7 int d=31;           // global, initialized
8
9 void main(void)
10 {
11     int e;          // local, uninitialized
12     char f[32];    // local, uninitialized
13     e = d + 7;
14     a = e + 29999;
15     strcpy(f, "Hello!");
16 }

```

The screenshot shows a debugger interface with several windows:

- Variables**: Shows variables romstart (2776), start (536866944), len (4), and loop (0). A red arrow points from the value 2776 in the Variables window to the value 2776 in the Memory Browser window.
- Registers**: Shows various CPU registers (r0-r12, sp, pc, xpsr, msp, psp, primask) with their current values.
- Disassembly**: Shows the assembly code for the startup routine, including the `data_init` function which copies data from flash to SRAM.
- Memory Browser**: Shows a memory dump starting at address 2776. The value 2776 is highlighted in yellow. A red arrow points from the value 2776 in the Variables window to the value 2776 in the Memory Browser window. Handwritten notes say "værdi læses fra Flash".

Handwritten annotations in green and red:

- "Skal skrives til denne adresse i RAM" (Should be written to this address in RAM)
- "4 bytes til en int (32 bit)" (4 bytes for an int (32 bit))
- "const = 31" (const = 31)

2776 = 0x00 00 0A D8 (Flash)
536866944 = 0x1F FF F0 80 (SRAM)

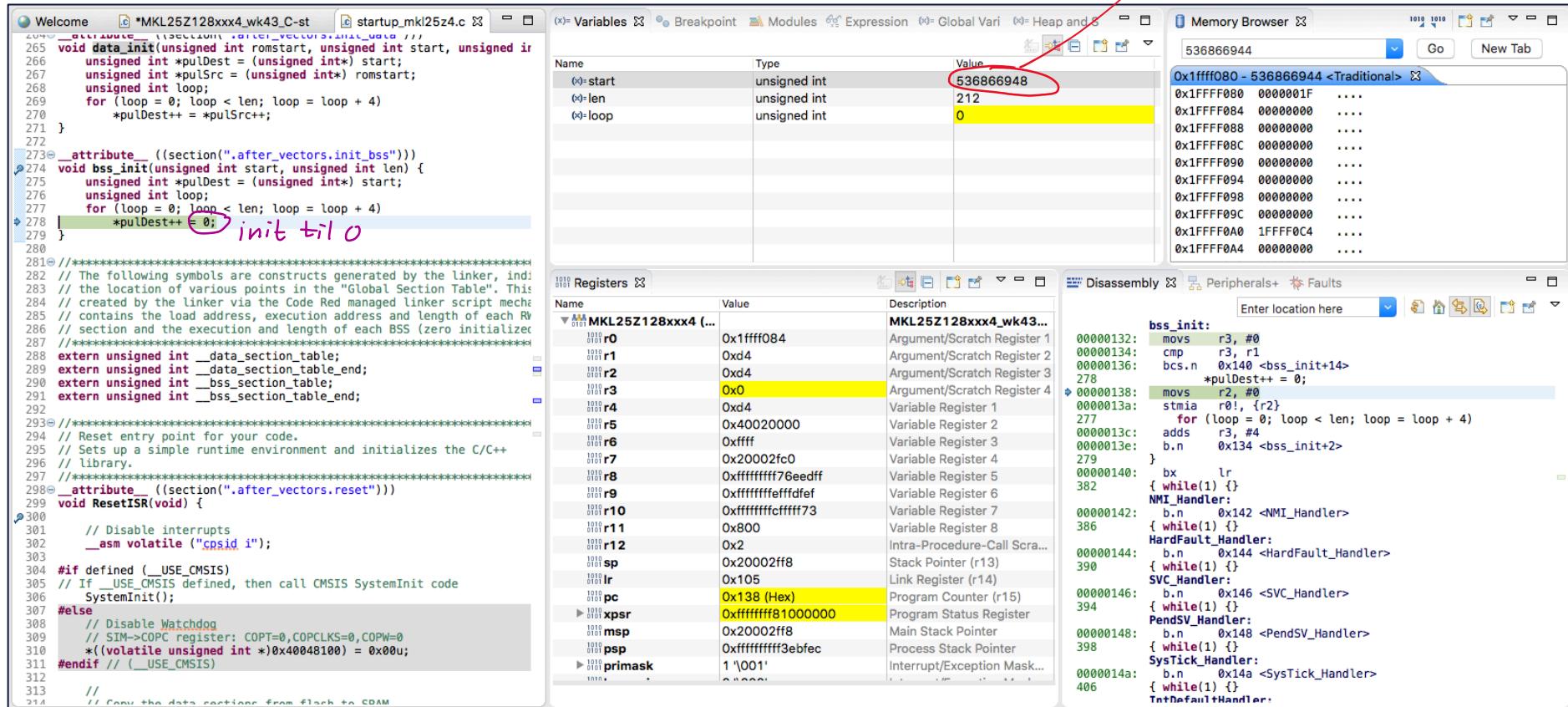
# HW1: STARTUP OG MEMORY

## BSS\_INIT

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int a, b;           // global, uninitialized
5 const char c=123;  // global const
6
7 int d=31;          // global, initialized
8
9
10 void main(void)
11 {
12     int e;           // local, uninitialized
13     char f[32];     // local, uninitialized
14     e = d + 7;
15     a = e + 29999;
16     strcpy(f, "Hello!");
17 }

```



# HW1: STARTUP OG MEMORY

## EFTER INIT

### Globale variable

Global variable

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int a, b;           // global, uninitialized
5 const char c=123;   // global const
6
7 int d=31;           // global, initialized
8
9 void main(void)
10 {
11
12     int e;           // local, uninitialized
13
14     char f[32];      // local, uninitialized
15
16     e = d + 7;
17
18 }
```

Variable	Type	Value	Address
(x)=a	int	0	0x1ffff084
(x)=b	int	536883200	0x0
(x)=c	const char	0 '\0'	0x0
(x)=d	int	31	0x1ffff080

init til 0 (bss)  
(bruges senere)

ikke lavet,  
bruger ikke

Bruges ikke  
→ ellers i flash

### Lokale variable

Lokale variable

```

1 #include <stdio.h>
2 #include <string.h>
3
4 int a, b;           // global, uninitialized
5 const char c=123;   // global const
6
7 int d=31;           // global, initialized
8
9 void main(void)
10 {
11
12     int e;           // local, uninitialized → stack
13
14     char f[32];      // local, uninitialized → Stack
15
16     e = d + 7;
17
18     a = e + 29999;
19
20     strcpy(f, "Hello!");
21
22 }
23
24 }
```

Name	Type	Value
(x)=e	int	212
(x)=f	char [32]	0x200002fc4
(x)=f[0]	char	24 '\030'
(x)=f[1]	char	241 '\n'
(x)=f[2]	char	255 '\y'
(x)=f[3]	char	31 '\037'
(x)=f[4]	char	214 '\O'
(x)=f[5]	char	10 '\n'
(x)=f[6]	char	0 '\0'
(x)=f[7]	char	0 '\0'
(x)=f[8]	char	255 '\y'
(x)=f[9]	char	255 '\y'
(x)=f[10]	char	0 '\0'
(x)=f[11]	char	0 '\0'
(x)=f[12]	char	0 '\0'
(x)=f[13]	char	0 '\0'

Name	Type	Value
(x)=e	int	38
(x)=f	char [32]	0x200002fc4
(x)=f[0]	char	72 'H'
(x)=f[1]	char	101 'e'
(x)=f[2]	char	108 'l'
(x)=f[3]	char	108 'l'
(x)=f[4]	char	111 'o'
(x)=f[5]	char	33 '!'

Variable	Type	Value	Address
(x)=a	int	30037	0x1ffff084
(x)=b	int	536883200	0x0
(x)=c	const char	0 '\0'	0x0
(x)=d	int	31	0x1ffff080

Pa stack  
→ uninitialized.

# HW1: STARTUP OG MEMORY

## BRUG AF VOLATILE QUALIFIER

---

### Volatile

- Problem:
  - Optimizer prøver at være "smart" og spare load-operationer.
  - Men værdien i hukommelsen kan have ændret sig siden sidste load.
- Løsning
  - Tving compiler til at lave load-operation før hver anvendelse af variabel vha. `volatile` qualifier.
- Anvendelse:
  - Repræsentere hardwareregistre som variable (memory-mapped)
  - Hvor ISR-rutiner kan ændre værdier asynkront
  - Multi-trådet programmering

**Kritiske sektioner** -> Atomic operations til at undgå race conditions.

- Problem: Operationer i en kritisk sektion må ikke afbrydes. Data races / race conditions på ikke-atomisk data giver risiko for "korrumpering".
- Løsning: Gøre kritiske sektioner interrupt-fri og derved "atomiske".
- Forekomst: Vigtige sammenhængende ikke-atomiske operationer (fx datastrukturer) (sænker dog programmets responsiveness).

# HW1: STARTUP OG MEMORY

## HVAD ER EN STACK

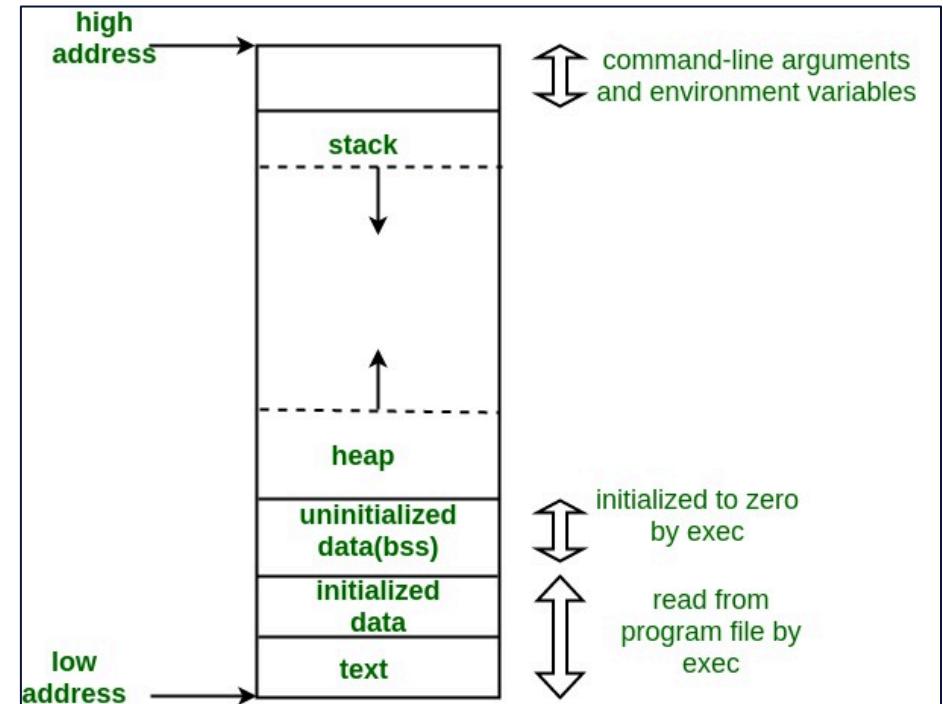
Datastruktur i RAM

Push og pop operationer

- Vokser "oppe fra og ned"
- push (data på stack) mindsker SP
- pop (data fra stack) øger SP

Bruges bl.a. til

- Funktionkald (-> stack frame/activation record med lokale variable)
- Gemme kontekst (registre, osv.), når et interrupt får CPU'en til at skifte opgave.

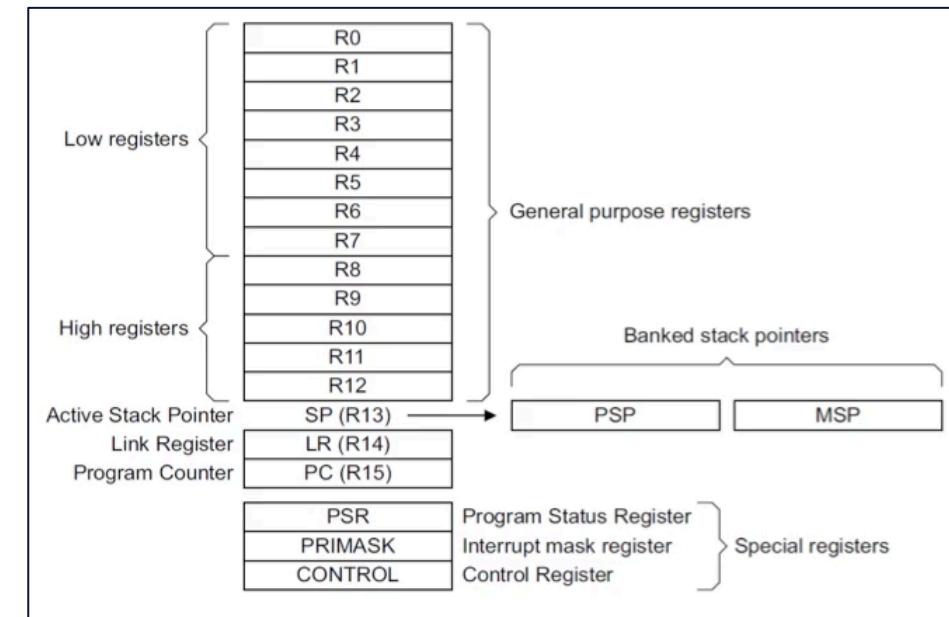


# HW1: STARTUP OG MEMORY

## STACK – SP ER ET REGISTER, DER "PEGER"

Registre i CPU (Core registers i registerfil)

- **SP:** Stack-pointer – peger ud i RAM
  - Stack: Midlertidig lagring af variabler, m.m.
    - MSP (main..) og PSP (process..)
  - 13 general purpose registers (R0-R12)
- **PC:** Program counter (R15)
- **LR:** Link (returadresse) (R14)
- Kontrolregister, bl.a.
  - PSR: APSR, IPSR, EPSR
  - PRIMASK: Masking af interrupts



# HW1: EMNE OPSUMMERING OG PERSPEKTIVERING

---

# HW2: PERIPHERAL I/O

## INDHOLD

---

1. Hvordan sættes pins op?
2. Tilgang til et register; pointers og assembly
3. Logiske operatorer

### 2: Peripheral I/O

1. Forklar hvordan et enkelt ben på processoren sættes op til en given funktion, hvis du f.eks fra en SPI driver vil bruge processorens PORTE3 som SPI1\_MISO.
2. Hvordan tilgår processoren et **register**, f.eks. TPM0\_SC, på adresse 0x40038000. Dvs. hvad oversættes dit C program til, og hvordan bruger processoren dette.
3. Hvad sker der i kode eksemplet nedenfor, hav særligt fokus på de logiske operatorer:

```
/* enable SPI1 & PORTE clock*/
SIM->SCGC4 |= SIM_SCGC4_SPI1_MASK;
SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;

/* disable SPI1 */
SPI1->C1 &= ~SPI_C1_SPE_MASK;

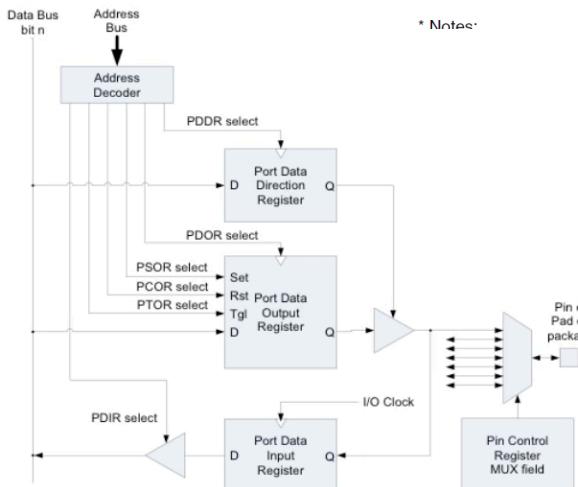
/* Setup pins */
/* E2 SPI1_SCK, alt2 */
PORTE->PCR[2] = (2 << 8);
```

# HW2: PERIPHERAL I/O

## OPSÆTNING AF PIN

Enkelt pins (ikke bus)

- Mux sætter fysisk forbindelse
- Pin control register MUX bits styrer funktion
- Forskellige pins har forskellige "muligheder" -> pinouts



Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R			0					ISF		0							
W								w1c									
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R			0					MUX		0	DSE	0	PFE	0	SRE	PE	PS
W										0	x*	0	x*	0	x*	x*	x*
Reset	0	0	0	0	0	x*	x*	x*		0	x*	0	x*	0	x*	x*	x*

\* Notes:

MUX (bits 10-8)	Configuration
000	Pin disabled (analog)
001	Alternative 1 – GPIO
010	Alternative 2
011	Alternative 3
100	Alternative 4
101	Alternative 5
110	Alternative 6
111	Alternative 7

For GPIO pins er der forskellige veje

- Normal GPIO (med DMA-adgang)
- FGPIO (FasterGPIO)
  - Sparer CPU-cycles. Lader I/O tage en kortere vej
  - Ingen DMA

FROM KL25Z Pins				KL25Z128 Pins											
On-board Usage	I/O Header & Pin Num	Arduino™ R3 Pin Name	FRDM-KL25Z Pin Name	KL25Z Pin #	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	Reset State/Function		
--	J1_20	D14	PTE0	1		PTE0		UAR71_T0	RTC_CLKOUT	GPI0_OUT	I2C1_SDA			DISABLED	
--	J1_18	D15	PTE1	2		PTE1		SP1_MOSI	UAR71_RX	SP1_MISO	I2C1_SCL			DISABLED	
--	J1_09		PTE2	3		PTE2		SP1_SCK						DISABLED	
--	J1_11		PTE3	4		PTE3		SP1_MSO						DISABLED	
--	J1_13		PTE4	5		PTE4		SP1_PCS0						DISABLED	
--	J1_15		PTE5	6		PTE5								DISABLED	
Power	--	VDD		7	VDD									VDD	
Power	--	VSS		8	VSS									VSS	
USB	--	--	USB0_DP	9	USB0_DP									USB0_DP	
2.2uf cap	--	--	USB0_DM	10	USB0_DM									USB0_DM	
USB VBUS (5V)	--	--	VOUT133	11	VOUT133									VOUT133	
--	J1_01		PTE0	13	ADC0_DR/DAC0_SE0		PTE0							ADC0_DR/DAC0_SE0	
--	J1_03		PTE1	14	ADC0_DM/DAC0_SE4a		PTE1							ADC0_DM/DAC0_SE4a	
--	J1_05		PTE2	15	ADC0_DR/DAC0_SE1		PTE2							ADC0_DR/DAC0_SE1	
--	J1_07		PTE3	16	ADC0_DM/DAC0_SE7a		PTE3							ADC0_DM/DAC0_SE7a	
Power	--	VDDA		17	VDDA									VDDA	
Power	J1_16	AREF*	VRGFH	18	VRGFH									VRGFH	
Power	--	VREFL		19	VREFL									VREFL	
Power	--	VSSA		20	VSSA									VSSA	
--	J1_09		PTE29	21	CMP0_IN5/ADC0_SE4b		PTE29							CMP0_IN5/ADC0_SE4b	
--	J1_11		PTE30	22	DAC0_OUT/ADC0_SE23/CMP0_IN4		PTE30							DAC0_OUT/ADC0_SE23/CMP0_IN4	

# HW2: PERIPHERAL I/O

## OPSÆTNING AF PIN

### Eksempel

- LCD-øvelse
- 5 keys til 5 funktioner

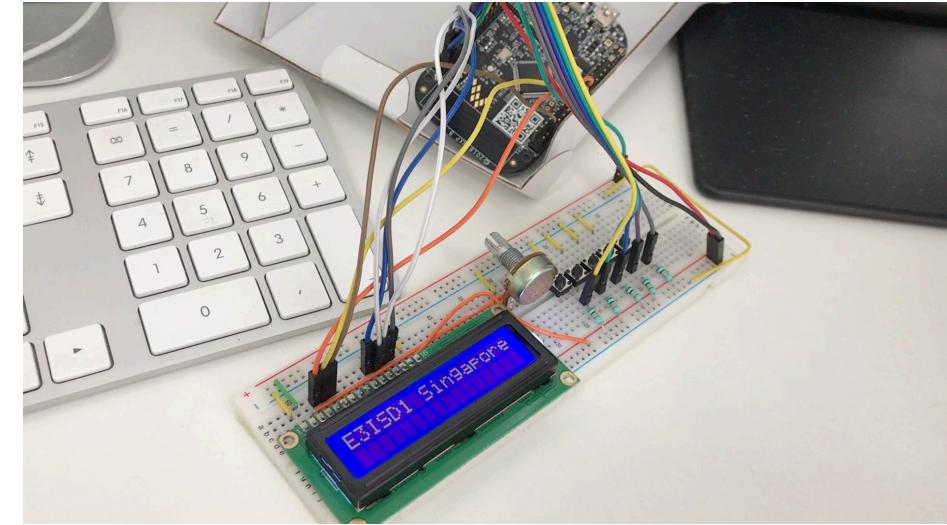
Sætter de 5 keys op

- Clock
- Clear MUX
- Sætter alternativ 1 (GPIO)
- Sætter data direction til input

Aftastning:

- Inverterer (aktiv lav)
- Tjekker ift mask
- Opdaterer array

```
18 /* All switches are on port E */
19 #define SW_UP_POS (21)
20 #define SW_DN_POS (29)
21 #define SW_LT_POS (30)
22 #define SW_RT_POS (23)
23 #define SW_CR_POS (22)
24
25 #define NUM_KEYS (5)
26
27 typedef enum {SWUp=0, SWDn, SWLt, SWRt, SWCr} sw_t;
28
29 /* Define macro to read switches */
30 #define READ_SWITCHES (PTE->PDIR)
31
32 void setup_keys(void) {
33
34     // Enable clock for GPIO keys
35     SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK; // Connect clock to Port E
36
37     // Set 5 switches pins as GPIO
38     PORTE->PCR[SW_UP_POS] &= ~PORT_PCR_MUX_MASK;
39     PORTE->PCR[SW_UP_POS] |= PORT_PCR_MUX(1);
40
41     PORTE->PCR[SW_DN_POS] &= ~PORT_PCR_MUX_MASK;
42     PORTE->PCR[SW_DN_POS] |= PORT_PCR_MUX(1);
43
44     PORTE->PCR[SW_LT_POS] &= ~PORT_PCR_MUX_MASK;
45     PORTE->PCR[SW_LT_POS] |= PORT_PCR_MUX(1);
46
47     PORTE->PCR[SW_RT_POS] &= ~PORT_PCR_MUX_MASK;
48     PORTE->PCR[SW_RT_POS] |= PORT_PCR_MUX(1);
49
50     PORTE->PCR[SW_CR_POS] &= ~PORT_PCR_MUX_MASK;
51     PORTE->PCR[SW_CR_POS] |= PORT_PCR_MUX(1);
52
53
54     /* Clear each SW to be input */
55     PTE->PDDR &= ~MASK(SW_UP_POS);
56     PTE->PDDR &= ~MASK(SW_DN_POS);
57     PTE->PDDR &= ~MASK(SW_LT_POS);
58     PTE->PDDR &= ~MASK(SW_RT_POS);
59     PTE->PDDR &= ~MASK(SW_CR_POS);
60 }
```



```
99 /* writes to the passed array */
100 void read_keys(unsigned key[NUM_KEYS]) {
101
102     // switches are active low
103     // get the 32-bit input register for PORT E and negate.
104     // Should have 1s where switches are pressed.
105     unsigned switch_code = ~READ_SWITCHES;
106
107     //Detect pressed keys
108     key[SWUp] = switch_code & MASK(SW_UP_POS) ? 1 : 0;
109     key[SWDn] = switch_code & MASK(SW_DN_POS) ? 1 : 0;
110     key[SWLt] = switch_code & MASK(SW_LT_POS) ? 1 : 0;
111     key[SWRt] = switch_code & MASK(SW_RT_POS) ? 1 : 0;
112     key[SWCr] = switch_code & MASK(SW_CR_POS) ? 1 : 0;
113
114
115 }
```

```
29 /* Define macro to read switches */
30 #define READ_SWITCHES (PTE->PDIR)
```

# HW2: PERIPHERAL I/O

## TILGÅ REGISTER

Processor-manual viser adresser i memory-mapped I/O

Defineret i MKL25Z4.h:

- TPM0\_SC status and control register er på adressen 0x40038000
  - Laver en pointer til dette sted, som har en struktur som TPM\_Type
  - Først 32 bits er SC
  - TPM0\_SC->SC er pointer dereferencing
- Assembly:
  - R3 holder ønsket adresse:
    - LDR (load with immediate offset) en værdi fra mem, pc + 80
  - Gem "1" i R2
  - STR (store with imm. offset): Gemmer registerværdi R2 i hukommelseslokation, som findes i register R3 med offset "0".

TPM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/page
4003_8000	Status and Control (TPM0_SC)	32	R/W	0000_0000h	31.3.1/552
4003_8004	Counter (TPM0_CNT)	32	R/W	0000_0000h	31.3.2/553
4003_8008	Modulo (TPM0_MOD)	32	R/W	0000_FFFFh	31.3.3/554

```

4309 /* TPM - Peripheral instance base addresses */
4310 /** Peripheral TPM0 base address */
4311 #define TPM0_BASE (0x40038000u)
4312 /** Peripheral TPM0 base pointer */
4313 #define TPM0 ((TPM_Type *)TPM0_BASE)

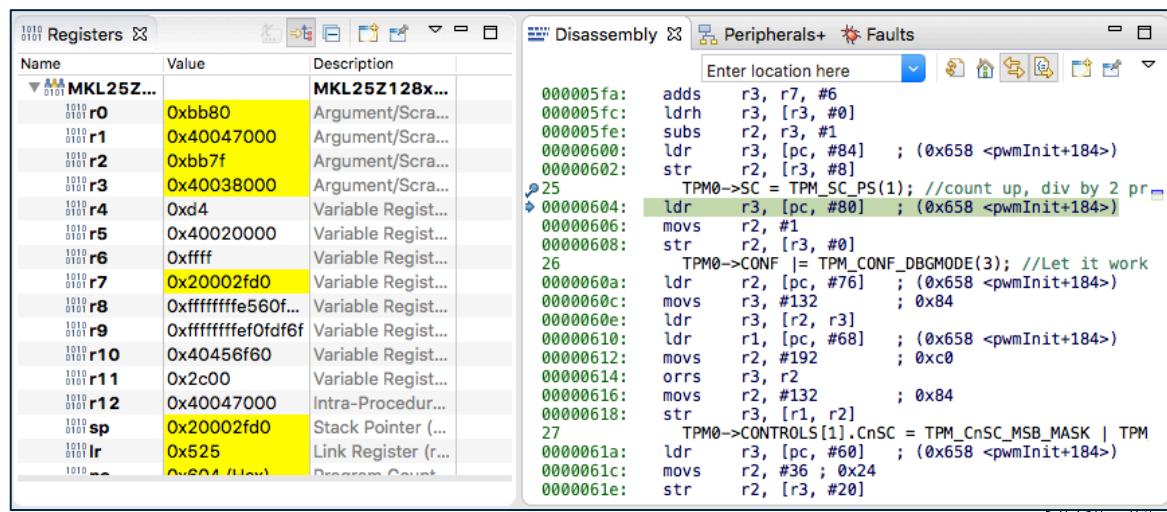
```

```

4169 /** TPM - Register Layout Typedef */
4170 typedef struct {
4171     __IO uint32_t SC;
4172     __IO uint32_t CNT;
4173     __IO uint32_t MOD;
4174     struct {
4175         __IO uint32_t CnSC;
4176         __IO uint32_t CnV;
4177     } CONTROLS[6];
4178     uint8_t RESERVED_0[20];
4179     __IO uint32_t STATUS;
4180     uint8_t RESERVED_1[48];
4181     __IO uint32_t CONF;
4182 } TPM_Type;

```

/\* Status and Control, offset: 0x0 \*/  
 /\* Counter, offset: 0x4 \*/  
 /\* Modulo, offset: 0x8 \*/  
 /\* offset: 0xC, array step: 0x8 \*/  
 /\* Channel (n) Status and Control, array offset:  
 \* /<< Channel (n) Value, array offset: 0x10, array  
 /\* Capture and Compare Status, offset: 0x50 \*/  
 /\* Configuration, offset: 0x84 \*/



# HW2: PERIPHERAL I/O

## "AFKODNING" AF EKSEMPEL

SPI1-modulet forbundet til System Clock Gating Control 4  
- Vi skal have tændt for clock til det (slukket for at spare energi)

Bitwise samtidig OR og tilskrivning  
 $hs = hs \mid vs$   
Så flipper bit højt, uanset om den var høj i forvejen

Mask makrovariabel med bit på 1 lokation

Port E pins forbundet til SCGC5  
Ligesom med SPI skal der tændes for clock for at portmodulet virker.

```
/* enable SPI1 & PORTE clock*/
SIM->SCGC4 |= SIM_SCGC4_SPI1_MASK;
SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;
```

Mask makrovariabel med bit på 1 lokation

PTE2 MUX-bits bliver sat til værdien 2 (0b010), det er alternativ 2, altså SPI1\_SCK, som er SPI1-clock...

```
/* disable SPI1 */
SPI1->C1 &= ~SPI_C1_SPE_MASK;
/* Setup pins */
/* E2 SPI1_SCK, alt2 */
PORTE->PCR[2] = (2 << 8);
```

Denne operation inverterer mask, så fx  $\sim(00..010..00) \rightarrow (11..101..11)$   
Bitwise AND med tilskrivning søger for at alt andet bliver som det var før ( $1\&1=1, 0\&1=0$ ), kun C1 i registret bliver sat til 0.

Hvorfor? Nok for at kunne måle på den via et oscilloskop / analog discovery...

Shifter værdien 2 (0b010) 8 placser "til venstre"... Svarer til eksponentiering  $2^8$ .  
Således bliver bits 10, 9 og 8 sat

Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R				0					ISF			0					
W									w1c								
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R				0					MUX	0	DSE	0	PFE	0	SRE	PE	PS
W										x*	x*	x*	x*	x*	x*	x*	x*
Reset	0	0	0	0	0	x*	x*	x*		0	x*	0	x*	0	x*	x*	x*

\* Notes:

FRDM-KL25Z Pin Name	KL25Z Pin #	ALT0	ALT1	ALT2	ALT3
PTE2	3		PTE2	SPI1_SCK	



# HW2: PERIPHERAL I/O

## OPSUMMERING OG PERSPEKTIVERING

---

# HW3: INTERRUPTS

## INDHOLD

---

1. Hvorfor interrupts? Hvorfor ikke bare polling?
2. Race conditions
3. Kort ISR – et eksempel

### **3: Interrupts**

- 1) Hvad er ***race-conditions***, og hvordan tages hensyn til disse ?
- 2) Antag at du har en digital indgang, GPIO, hvor du ønsker at koble en trykknap på. Forklar forskellen på at anvende hhv. **Polling** og **Interrupts** til at reagere på et knaptryk.
- 3) Hvorfor ønsker vi at en *Interrupt Service Routine - ISR* skal være kort / hurtig afviklet

# HW3: INTERRUPTS

## HVORFOR INTERRUPTS? IKKE POLLING?

---

### Responsive systemer

- Absolutte deadlines (sikkerhed)
- Usability og UX

Polling er busy waiting med meget høj CPU overhead, spilder bare clock cycles, energi, processor-tid på at vente

### Efficiens: Concurrency, dele processorens (tid), scheduling

- Aflaste software med hardware -> event-baseret, interrupts
  - Responsiveness (~15 clock cycles fra IRQ til ISR-afvikling i teorien)

### Løsninger:

- Hardware: Interrupts og event-triggering
- Software: Scheduling (statisk/kooperativt og dynamisk m/u task-level preemption)

# HW3: INTERRUPTS RACE CONDITIONS, DATA RACES

**Overordnet:** Flere operationer fra forskellige sektioner af et program prøver at modificere / læse på data.

- Fx en ISR, der afbryder afvikling af anden kode. Måske ikke atomisk. Måske ved den anden kode ikke, at forudsætninger er ændret.

**Løsning:** Kritiske sektioner -> Atomic operations til at undgå race conditions.

- Problem: Operationer i en kritisk sektion må ikke afbrydes. Data races / race conditions på ikke-atomisk data giver risiko for "korrumpering".
- Løsning: Gøre kritiske sektioner interrupt-fri og derved "atomiske".
- Forekomst: Vigtige sammenhængende ikke-atomiske operationer (fx datastrukturer) (sænker dog programmets responsiveness).

Volatile *kan være en anden løsning...*

- Problem:
  - Optimizer prøver at være "smart" og spare load-operationer.
  - Men værdien i hukommelsen kan have ændret sig siden sidste load.
- Løsning
  - Tving compiler til at lave load-operation før hver anvendelse af variabel vha. volatile qualifier.
- Anvendelse:
  - Repræsentere hardwareregistre som variable (memory-mapped)
  - Hvor ISR-rutiner kan ændre værdier asynkront
  - Multi-trådet programmering

# HW3: INTERRUPTS

## INTERRUPT LATENCY

Øvelse uge 40:

- Fra tryk på knap til ISR kører, ca. 690 ns.
- @48 MHz svarer det til ca. 33 clock cycles.
- Noget længere end de 15-16 cycles nævnt.

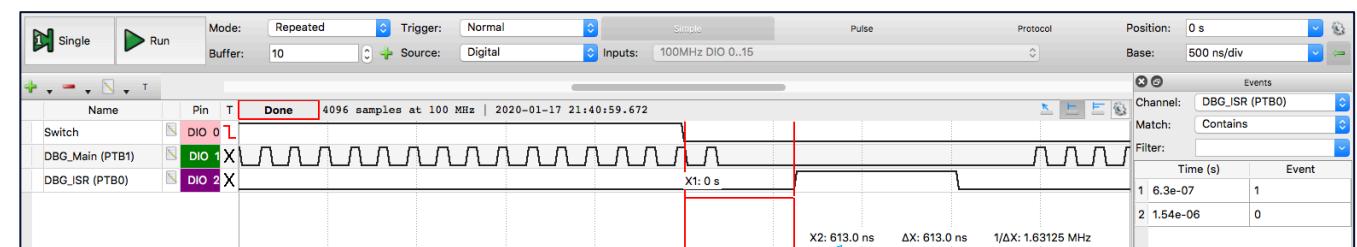
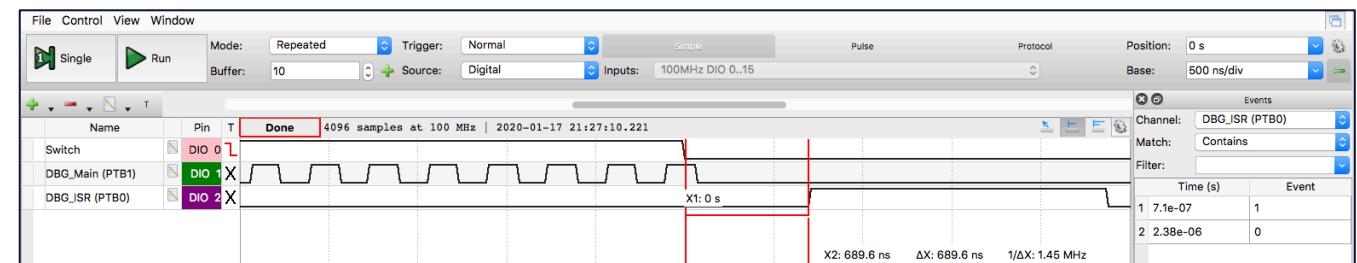
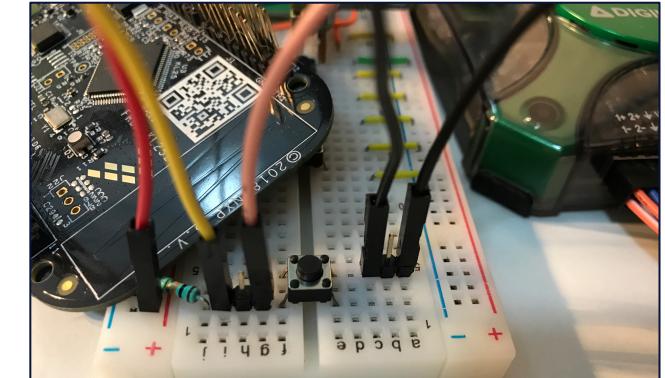
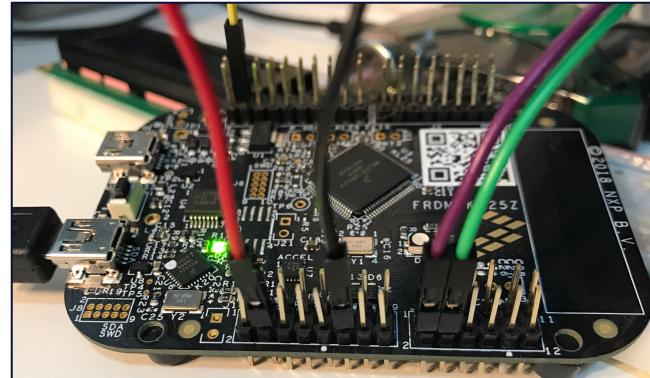
Mulige forklaringer

- Leadtime fra ISR startes til GPIO-signal aktivt.
- Ventetid på at gemme kontekst på stack.
- CPU/bus, mv. kører ikke 48 MHz, ej optimeret.

Når kode optimeres (-O3) forbedres med 77 ns.

- Bemærk også ændret frekvens på DBG\_Main og kortere ISR-tid.

Måske behøver vi ikke afbryde? Brug en hardwaretimer/counter i baggrunden til at tælle events...



Med -O3  
forbedres med  
77 ns

# HW3

## KORT ISR

### Kort ISR

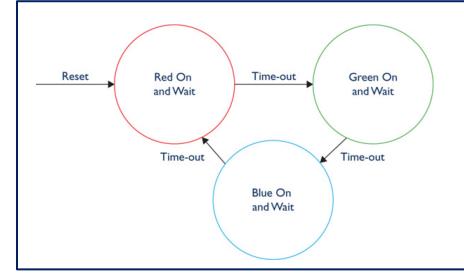
- Lang ISR sørger "responsiveness" af resten af koden
  - Plads til nye interrupts - kan forsinke start af andre ISR'er (samtidigt opståede nu i prioritetslisten)
  - Performance/responsiveness vs. complexity
  - Evt. nestede interrupts / med prioriteter.
- Bliver nemt til spaghetti-kode
  - Svært at forstå, vedligeholde, udvikle på

Eksempel: En nok lidt for lang ISR fra uge 38

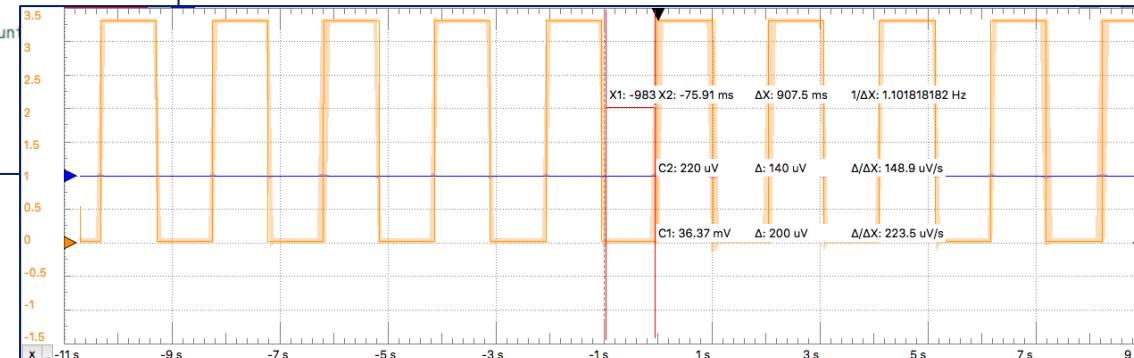
- Burde flytte noget kode ud
- Rammer næsten den ønskede frekvens, men der er lidt forsinkelse...

```

60 void SysTick_Handler(void)
61 {
62     // Variables for the state machine
63     static state_t state = RED; //Initialize as RED
64     static state_t state_next;
65     static unsigned long count = 0;
66
67     // The LEDs are connected so high=off, low=on.
68     switch (state) {
69     case RED:
70
71         PTB->PCOR = MASK(RED_LED_SHIFT); // red LED on
72         PTD->PSOR = MASK(BLUE_LED_SHIFT); // blue LED off
73
74         count++; // counter for times in case
75
76         // Run the red light for 3 cycles of 100 ms / cycle = 300 ms
77         if (count < 3) {
78             state_next = RED; // Go to this case again next
79         } else {
80             state_next = GREEN;
81             count = 0; // Reset counter
82         }
83
84         break;
85
86     case GREEN:
87         PTB->PCOR = MASK(GREEN_LED_SHIFT); // set the green LED
88         PTD->PSOR = MASK(RED_LED_SHIFT); // clear the red LED
89
90         count++;
91
92         // Run the green light for 6 cycles = 600 ms
93         if (count < 6) {
94             state_next = GREEN; // Go to this case again next
95         } else {
96             state_next = BLUE;
97             count = 0; // Reset counter
98         }
99
100        break;
101
102    case BLUE:
103        PTD->PCOR = MASK(BLUE_LED_SHIFT); // set the blue LED
104        PTD->PSOR = MASK(GREEN_LED_SHIFT); // clear the green LED
105        count++;
106
107        // Run the blue light for 9 interrupt cycles = 900 ms
108        if (count < 9) {
109            state_next = BLUE; // Go to this case again next
110        } else {
111            state_next = RED;
112            count = 0; // Reset counter
113        }
114
115        break;
116    }
117
118    state = state_next;
119 }
```



STATE	LED farve	TimeOut
RED	Rød	300[mS]
GREEN	Grøn	600[mS]
BLUE	Blå	900[mS]



# HW3: EMNE OPSUMMERING OG PERSPEKTIVERING

---



# HW4: TIMERS

## INDHOLD

---

1. Perifere enheder med timer-funktioner
2. Interrupts med en timer + håndtering
  1. SysTick-eksempel (uge 46)
  2. Eksempel med FSM-håndtering (uge 38)
3. Opsætning af TPM
  1. Oversigt over registre
  2. PWM-eksempel (uge 46)
4. Opsummering og perspektivering

### **4: Timers**

- 1) Redegør for hvordan et timermodul, som KL25's TPM (Timer/PWM Module) kan anvendes, f.eks. til **periodisk interrupt**, **PWM** eller **input capture**.
- 2) Giv et overblik over hvilke perifer-enheder (Peripherals) i KL25 processoren der indeholder timer-funktioner
- 3) Forklar hvordan en timer kan bruges til at generere periodiske interrupts, og hvordan disse kan håndteres, f.eks. hvis du skal bruge en 10ms tællevariabel.

# HW4: TIMERS

## PERIFERE TIMERENHEDER PÅ KL25Z-BOARD

Formål:

- Hardware aflaster software
- Tælle tid / events
  - Clocked synkront / asynkront
- Beregninger på signal, IRQ, PWM, osv.

Perifere enheder:

- TPM: Timer/PWM
- PIT: Intervaller
- LPTMR: Low-power TMR
- RTC: Real-time Clock
- Watchdog: Fejlddetektion. Holder øje med CPU "heartbeat" -> reset vektor
- SysTick (Cortex-M0)

# HW4: TIMERS INTERRUPTS MED EN TIMER

## SysTick driver uge 46

Bruger Cortex-M0+'s SysTick til at generere interrupts med 1500 ms interval

- initSystickTmr
- ISR, toggler
- Beregning af loadværdi
- Resten er standard / loop.

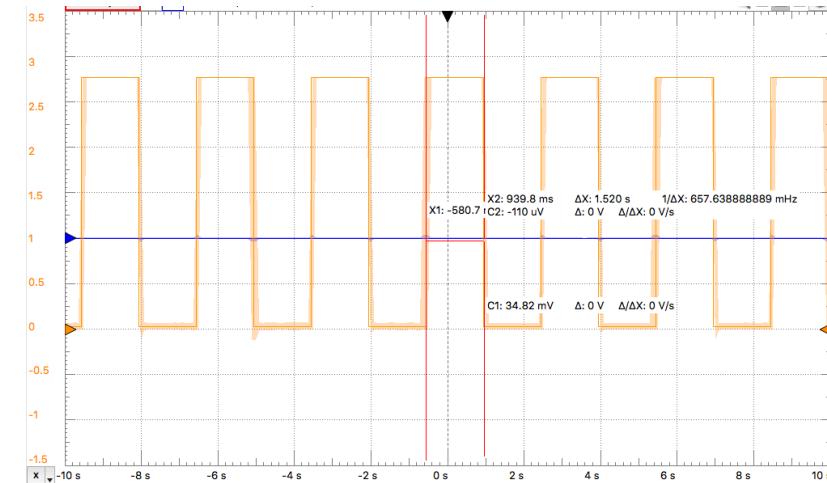
Med Analog Discovery dok. interval på 1500 ms.

10 ms? Genberegn LDV:

- $LDV = 10e-3 / 3e-6 - 1$
- $LDV = 30e3 - 1$

```
18@@/**  
2  * @file SysTick.c  
3  * @brief E3ISD1 Week 46: Set up SysTick timer to run at 3 MHz and load  
4  * compare register  
5  * @date 2019  
6  * @author Janus Bo Andersen  
7  * Function for setting up SysTick and corresponding interrupt,  
8  * also with 'ticks' valued as trigger threshold for triggering interrupt  
9  *  
10 * @note timer is started after calling this function  
11 * @param ticks number of ticks, counting at 3 MHz, before triggering input  
12 * @return 0 on success, -1 if ticks outside range  
13 *  
14 */  
15  
16 #include "SysTick.h"  
17  
18@ int initSystickTmr(unsigned int ticks) {  
19  /* Based on Dean p. 189 */  
20  
21  // Check valid input, already guaranteed unsigned, must fit in 24-bit (2^24-1)  
22  if (ticks == 0 || ticks > 16777215) {  
23      return -1;  
24  }  
25  
26  // LDV = T_target / T_counter -1.  
27  SysTick->LOAD = (ticks);  
28  
29  // Set interrupt  
30  NVIC_SetPriority(SysTick_IRQn, 3);  
31  
32  // Start at zero so it initially reloads  
33  SysTick->VAL = 0;  
34  
35  // Note: We use the alternate clock source  
36  // (48 MHz/16 = 3 MHz), so CLKSOURCE bit is 0  
37  SysTick->CTRL = SysTick_CTRL_TICKINT_Msk | // Set exception on reaching zero  
                  SysTick_CTRL_ENABLE_Msk; // Enable the SysTick  
38  
39  return 0;  
40  
41 }  
42 }
```

```
37@ void SysTick_Handler() {  
38  // Toggle the blue LED  
39  PTD->PTOR = MASK(BLUE_LED_SHIFT);  
40 }  
  
62  // LDV = T_target / T_counter -1 = 1.5s / (1 / 3MHz) = 4.5e6.  
63  const unsigned int LDV = 4500000L-1L;  
64  
65  if ( !initSystickTmr(LDV) ) {  
66      //error occurred  
67      return -1;  
68 }
```



# HW4: TIMERS

## HÅNDTERING AF TIMERINTERRUPTS MED FSM

FSM ISR uge 38

## SysTick hver 10 ms

## SysTick ISR indeholder FSM

- 3 tilstande
  - Hver tilstand har forskellig varighed
  - Designet kan forbedres!
    - Ryk kode ud af ISR...

Benyttede ikke egen driver, men den fra core\_cm0plus.h

- CLKSRC=1, så der burde benyttes processorclock (48 MHz), men den kører faktor-2 langsommere

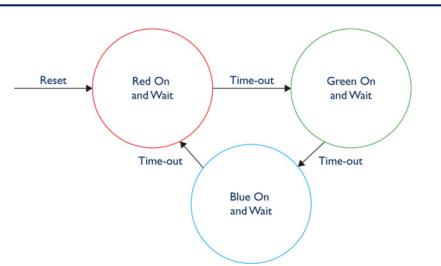
```

00 void SysTick_Handler(void)
01 {
02     // Variables for the state machine
03     static state_t state = RED; //Initialize as RED
04     static state_t state_next;
05     static unsigned long count = 0;
06
07     // The LEDs are connected so high=off, low=on.
08     switch (state) {
09     case RED:
10
11         PTB->PCOR = MASK(RED_LED_SHIFT); // red LED on
12         PTD->PSOR = MASK(BLUE_LED_SHIFT); // blue LED off
13
14         count++; // counter for times in case
15
16         // Run the red light for 3 cycles of 100 ms / cycle = 300 ms
17         if (count < 3) {
18             state_next = RED; // Go to this case again next
19         } else {
20             state_next = GREEN;
21             count = 0; // Reset counter
22         }
23
24         break;
25
26     case GREEN:
27
28         PTB->PCOR = MASK(GREEN_LED_SHIFT); // set the green LED
29         PTD->PSOR = MASK(RED_LED_SHIFT); // clear the red LED
30
31         count++; // Run the green light for 6 cycles = 600 ms
32         if (count < 6) {
33             state_next = GREEN; // Go to this case again next
34         } else {
35             state_next = BLUE;
36             count = 0; // Reset counter
37         }
38
39         break;
40
41     case BLUE:
42
43         PTD->PCOR = MASK(BLUE_LED_SHIFT); // set the blue LED
44         PTD->PSOR = MASK(GREEN_LED_SHIFT); // clear the green LED
45         count++; // Run the blue light for 9 interrupt cycles = 900 ms
46         if (count < 9) {
47             state_next = BLUE; // Go to this case again next
48         } else {
49             state_next = RED;
50             count = 0; // Reset counter
51         }
52
53         break;
54     }
55
56     state = state_next;
57 }

```

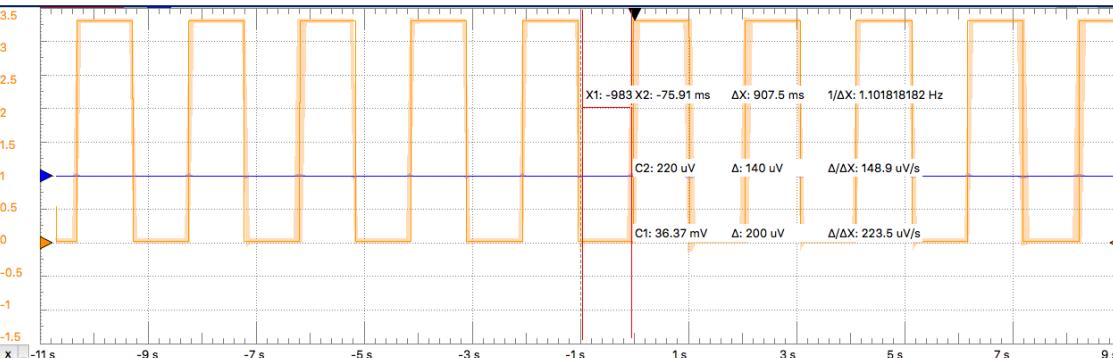
```
52 @typedef enum {
53     RED,
54     GREEN,
55     BLUE
56 } state_t;

144 /*100 ms = 10 interrupts pr. second. Fcpu / 10 = sysTickConfigValue */
145 /* Was 4800000, but there is some factor clock scaling going on here */
146 SysTick_Config(2400000);
```



STATE	LED farve	TimeOut
RED	Rød	300[mS]
GREEN	Grøn	600[mS]
BLUE	Blå	900[mS]

```
8855 STATIC_INLINE uint32_t SysTick_Config(uint32_t ticks)
886 {
887     if ((ticks - 1UL) > SysTick_LOAD_RELOAD_Msk)
888     {
889         return (1UL);                                /* Reload value impossible */
890     }
891
892     SysTick->LOAD  = (uint32_t)(ticks - 1UL);      /* set reload register */
893     NVIC_SetPriority (SysTick_IRQn, (1UL << __NVIC_PRIO_BITS) - 1UL); /* set Priority for SysTick Interrupt */
894     SysTick->VAL   = 0UL;                          /* Load the SysTick Counter Value */
895     SysTick->CTRL  = SysTick_CTRL_CLKSOURCE_Msk | /* Enable SysTick IRQ and SysTick Timer */
896                      SysTick_CTRL_TICKINT_Msk |
897                      SysTick_CTRL_ENABLE_Msk;        /* Function successful */
898     return (0UL);
899 }
900
901 #endif
```



# HW4: TIMERS

## OPSÆTNING AF TPM

Konfiguration, der skal sættes for **TPM-modulet**

- Clock source (CMOD)
- Interrupt ved overflow (TOIE ved TOF)
- Tælleretning (CPWMS)
- Prescaling (PS)
- Free-running eller modulo tæller (TPMx\_MOD)

Kanal-indstillinger – næste slide...

### TPMx\_SC (TPM0, TPM1, TPM2)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R							0		TOF							
W									DMA		TOIE		CPWMS		CMOD	
Reset	0	0	0	0	0	0	0	0	w1c	0	0	0	0	0	0	0

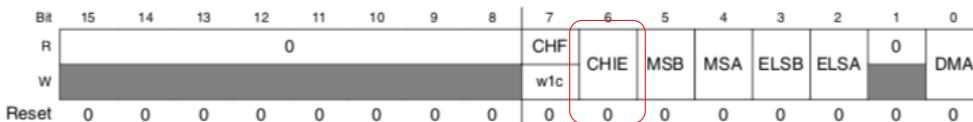
6 TOIE	Timer Overflow Interrupt Enable  Enables LPTPM overflow interrupts.  0 Disable TOF interrupts. Use software polling or DMA request. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
-----------	--

# HW4: TIMERS

## OPSÆTNING AF TPM

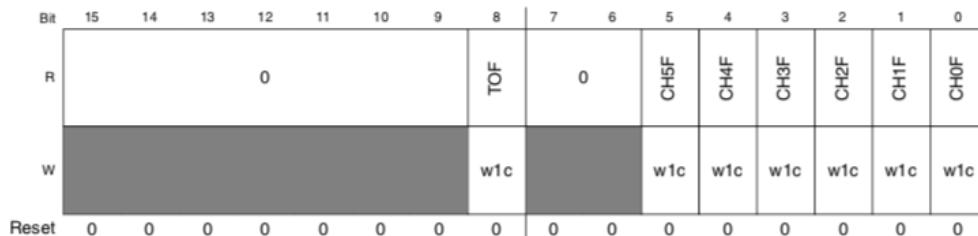
Konfiguration, der skal sættes for **kanalen**

- TPM0: 6 kanaler, TPM1 og TPM2: Hver 2 kanaler
- Register TPMx\_CnSC:



- Interrupt på kanalen (CHIE)
- Mode (MSB, MSA)
- Edge select (ELSB, ELSA)

Alle flag kan ses i TPMx\_STATUS



### Konfiguration af en kanal

CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	00	00	None	Channel disabled
X	01/10/11	00	Software compare	Pin not used for LPTPM
0	00	01	Input capture	Capture on Rising Edge Only
		10		Capture on Falling Edge Only
		11		Capture on Rising or Falling Edge
	01	01	Output compare	Toggle Output on match
		10		Clear Output on match
		11		Set Output on match
	10	10	Edge-aligned PWM	High-true pulses (clear Output on match, set Output on reload)
		X1		Low-true pulses (set Output on match, clear Output on reload)
	11	10	Output compare	Pulse Output low on match
		X1		Pulse Output high on match
1	10	10	Center-aligned PWM	High-true pulses (clear Output on match-up, set Output on match-down)
		X1		Low-true pulses (set Output on match-up, clear Output on match-down)

# HW4: TIMERS

## EKSEMPEL PÅ OPSÆTNING AF PWM

Uge 46:

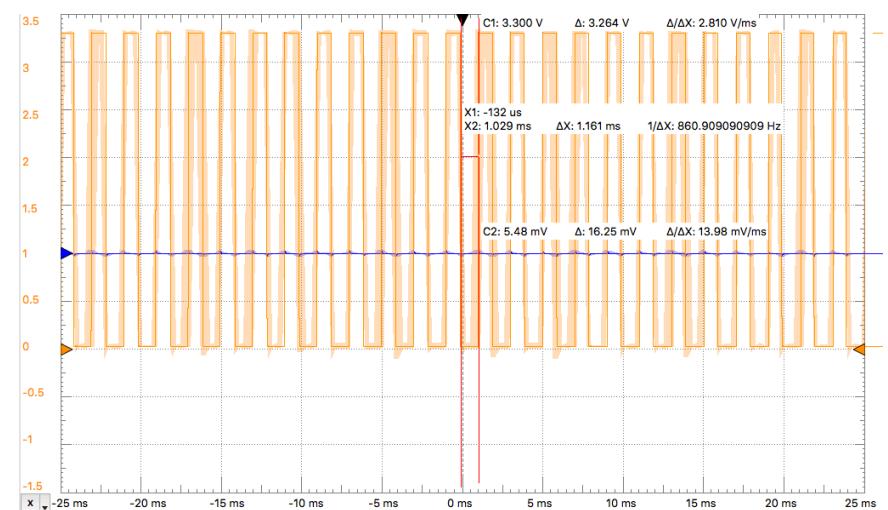
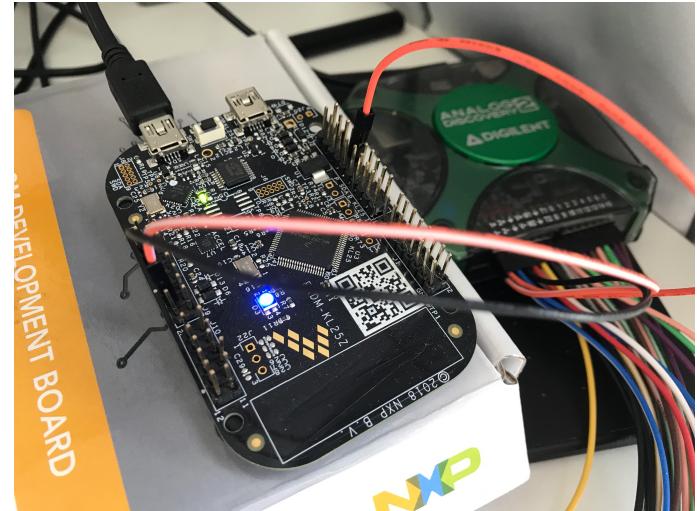
- Skrue op og ned for lysstyrke i LED
- Duty-cycle: CnV/MOD

Puls ift. edge: start

Måler med Analog Discovery på D1 (blå diode)

```
1④ /**
2 * @file pwm.c
3 * @brief E3ISD1 Week 46: Set up PWM to manage LED brightness
4 * @date E3ISD1 2019
5 * @author Janus Bo Andersen
6 * @note uint16_t because TPM MOD counter only supports 16 bit
7 * @param period number of clock ticks in a period
8 * @return 0 on success, -1 if ticks outside range
9 */
10 */
11 #include "pwm.h"
12
13④ int pwmInit(uint16_t period) {
14
15     SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK; // Clock PTD1 (LED)
16     SIM->SCGC6 |= SIM_SCGC6_TPM0_MASK; // Clock TPM
17
18     //Let's use FAST GPIO
19     PORTD->PCR[BLUE_LED_POS] &= ~PORT_PCR_MUX_MASK;
20     PORTD->PCR[BLUE_LED_POS] |= PORT_PCR_MUX(4); //Alt. 4 FGPIO
21
22     //Config TPM0
23     SIM->SOPT2 |= (SIM_SOPT2_TPMSRC(1) | SIM_SOPT2_PLLFLLSEL_MASK); //48 MHz
24     TPM0->MOD = period-1; // MOD counter mod
25     TPM0->SC = TPM_SC_PS(1); //count up, div by 2 prescaler
26     TPM0->CONF |= TPM_CONF_DBGMODE(3); //Let it work in debug mode
27     TPM0->CONTROLS[1].CnSC = TPM_CnSC_MSB_MASK | TPM_CnSC_ELSA_MASK; //Pwm edge align
28     TPM0->CONTROLS[1].CnV = 0; //initial duty cycle
29     TPM0->SC |= TPM_SC_CMOD(1); //START :
30
31     return 0;
32 }
```

```
37     pwmInit(PWM_PERIOD); //Initialize PWM using our driver
38     while(1) {
39         //First brighten the LED
40         for (i=0; i<PWM_PERIOD; i++) {
41             TPM0->CONTROLS[1].CnV = i;
42
43             //wait a while
44             for (delay=0; delay<100; delay++) {
45                 //pass
46             }
47         }
48         //Then fade the LED
49         for (i=PWM_PERIOD-1; i>0; i--) {
50             TPM0->CONTROLS[1].CnV = i;
51
52             //wait a while
53             for (delay=0; delay<100; delay++) {
54                 //pass
55             }
56     }
```



# HW4: TIMERS OPSUMMERING OG PERSPEKTIVERING

---



# HW5: SERIEL KOMMUNIKATION

## INDHOLD

---

1. Synkron vs. asynkron kommunikation
  1. Eksempel
2. Overblik over seriell kommunikation (væsentlige karakteristika ved I2C, SPI og UART)
3. Forskelle på fuld duplex og halv duplex
  1. Eksempel
4. Opsummering og perspektivering

### **5: Seriel kommunikation**

- 1) Hvad er de væsentlige forskelle der adskiller hhv. synkron/asynkron kommunikation, giv et eksempel.
- 2) Hvad er de væsentlige forskelle på full- og half- duplex, giv et eksempel.
- 3) Redegør for væsentlige karakteristika ved hhv. SPI/I2C/UART.

# HW5: SERIEL KOMMUNIKATION

## SYNKRON VS ASYNKRON

Synkron:

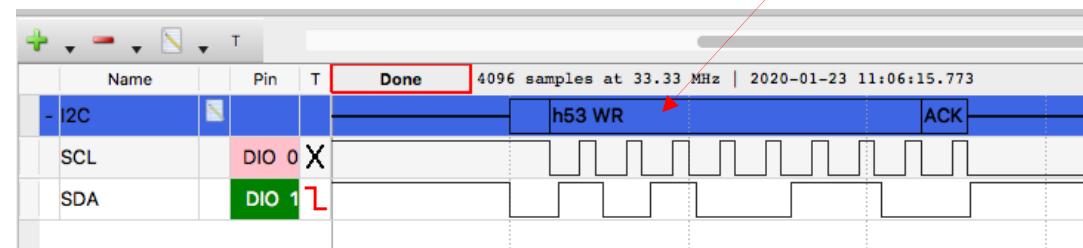
- Følles clock-signal
- Koordineret af master eller ekstern clock
- Eksempel: SPI, I2C

Eksempel: Kommunikation med et ADXL345-modul

- SCL er clock
- Master trækker SDA lav for at initiere
- 0x53 addresseres

Asynkron:

- Kommunikationshastighed aftalt på forhånd (baud)
- Krav til nogenlunde ens intern clockfrekv.
- Eller sat som standardhastighed
- Eksempel: UART (selvom der findes USART)
- Forskellige protokoller



Eksempel: Kommunikation med et ADXL345-modul

- Initiering og bit overføres
- Registre aflæses

Begge typisk aktiv lav



Eksempel: Når clock stopper og SDA går høj, så er kommunikationen slut

# HW5: SERIEL KOMMUNIKATION

## SERIEL KOMMUNIKATION

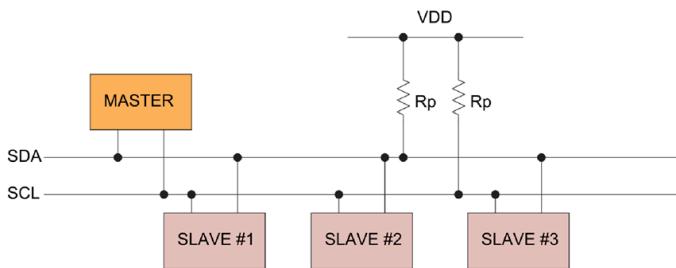
### I2C

Seriel-bus, hardware og protokol (half-duplex)

- 2 linjer: SDA, SCL (aktiv lav)
- Synkron (clocked, SCL)
- Philips / NXP specifikation
- Hurtigere end UART, langsommere end SPI

Master-slaves

- 7-bit-adresser (128 enheder, men...)
- Clock-synkronisering fra master
  - Clock-stretching fra slave
- Multi-master via "collision detection" og "arbitration".



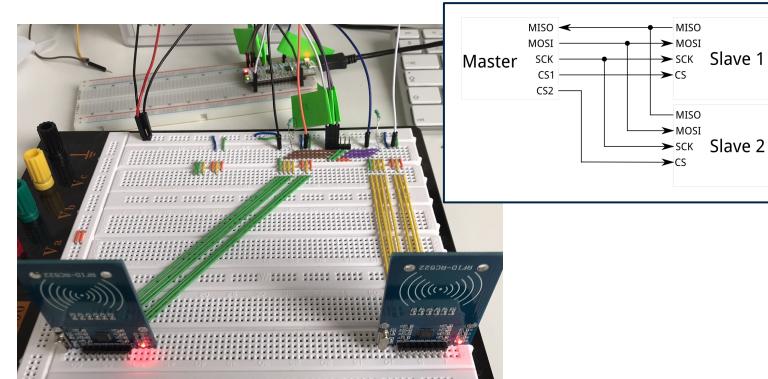
### SPI

Seriel-bus, hardware og protokol (full-duplex)

- 2+N linjer: MOSI, MISO, SCK,  $N \times$ SS/CS
- Synkron (clocked, SCK)
- Master-slave(s), kun 1 master
- Hurtigste af I2C, UART og SPI

Eksempel:

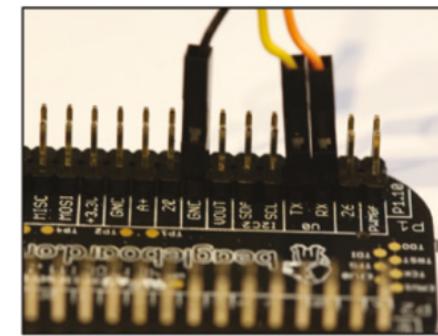
- Øvelse 20b
- Pro2, flere RFID-læsere i "samme hus"



### UART

Seriel-forbindelse (half-duplex)

- Point-to-point (UART1 <-> UART2)
  - Rx, Tx linjer på hvert punkt
- Asynkron: Ingen fælles clock (intern baud rate)
  - Hastighed (<115200 baud) skal være ens i hver ende
- Start/stop-bits, og maks. længde af databits (5-9) -> typisk 8 og paritet
- Hardware: Parallel databus -> skifteredister -> Tx -> seriel forb. -> Rx, osv...
- Oversampling til fx støjreduk.
- Kan implementere forskellige kommunikationsprotokoller, fx RS-232
- Langsommet af de tre
- Men simpel!



# HW5: SERIEL KOMMUNIKATION

## HALV / FULD DUPLEX

### Halv duplex

Kun en kanal, kan benyttes af en ad gangen

- Enten Tx eller Rx

Eksempler:

- UART med ét skifteredister:
  - Skifteredister bruges til en Tx eller Rx, FIFO-buffer afkobler
- I2C:
  - Kun en bus, og kun 1 enhed kan addresseres ad gangen

### Simplex:

- Kun en kan overføre på kanalen, den anden kan kun lytte.

### Fuld duplex

To kanaler eller forskellige frekvensbånd

- Samtidig Tx og Rx på to forskellige kanaler/bånd

Eksempel:

- UART med to skifteredistre
  - Et til Tx og et til Rx, to FIFO buffers afkobler
- SPI: Fuld-duplex, fordi der er både MOSI- og MISO-kanaler
  - Dog kun 1 slave ad gangen (kun 1 CS/SS)
- ADSL:
  - Frekvensopdeling
- Oplagt eksempel:
  - Telefoner: Både digitale/mobiler og de gamle analoge

Ikke-serielle:

- Parallel databus

# HW5: SERIEL KOMMUNIKATION

## EKSEMPEL SPI

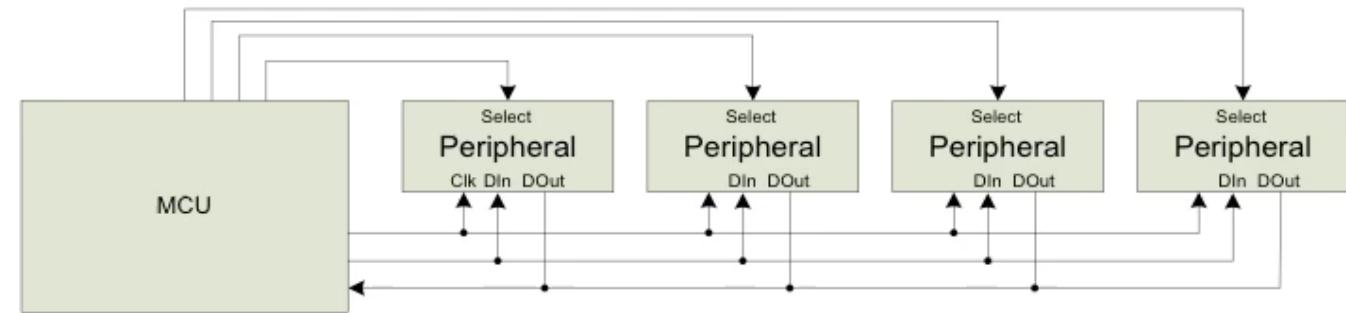
Serielt: En linje til hver dataretning (ind/ud)

Synkront (clock)

- SPI: SCK

Databus

- SPI: MISO, MOSI



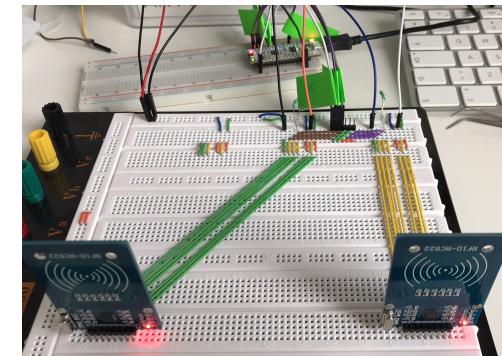
Select:

- SPI: SS/CS

Antal ben/linjer med N tilsluttede enheder:

- $3 + N$

Færre pins ud af IC end parallelforb.... Mindre pakke, mindre energi.



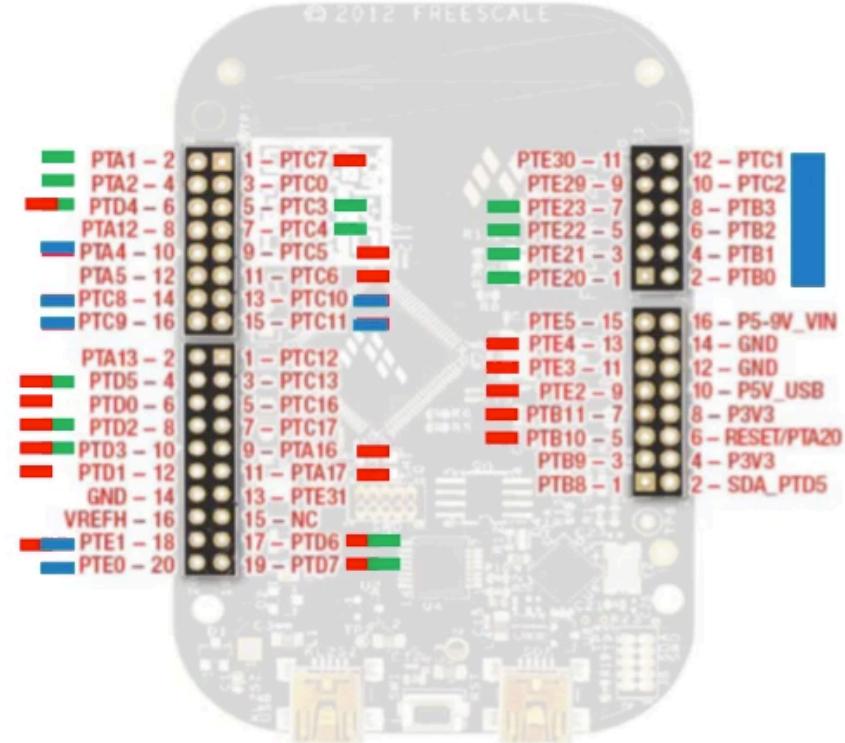
# HW5: SERIEL KOMMUNIKATION

## KL25Z

KL25Z har mulighed for UART, SPI og I<sup>2</sup>C

- 3x UART (UART0 er dog optaget)
- 2x SPI
- 2x I<sup>2</sup>C
- USB

Hvis vi skal bruge CAN, så er det på  
PocketBeagle ;)



UART  
SPI  
I<sup>2</sup>C

# HW5: SERIEL KOMMUNIKATION

## KL25Z UART INITIALISERING

UART2 clock fra SCGC4

UART2 Control register 2  
Slukker for TX (TE) og RX (RE)

Baud rate generator (prescaler)  
Clockes ift. bus clock på 24 MHz  
BDH og BDL er High Low bits i modulo divisor

UART2 data register  
-> Nu skriver vi endelig ud på linjen...

```
void Init_UART2(uint32_t baud_rate) {
    uint32_t divisor;
    // enable clock to UART and Port A
    SIM->SCGC4 |= SIM_SCGC4_UART2_MASK;
    SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;

    // connect UART to pins for PTE22, PTE23
    PORTE->PCR[22] = PORT_PCR_MUX(4);
    PORTE->PCR[23] = PORT_PCR_MUX(4);
    // ensure tx and rx are disabled before configuration
    UART2->C2 &= ~(UARTLP_C2_TE_MASK | UARTLP_C2_RE_MASK);

    // Set baud rate to 4800 baud
    divisor = BUS_CLOCK/(baud_rate*16);
    UART2->BDH = UART_BDH_SBR(divisor>>8);
    UART2->BDL = UART_BDL_SBR(divisor);

    // No parity, 8 bits, two stop bits, other settings;
    UART2->C1 = UART2->S2 = UART2->C3 = 0;

    // Enable transmitter and receiver
    UART2->C2 = UART_C2_TE_MASK | UART_C2_RE_MASK;
}
```

Vil outputte på PTE22 og PTE23, så tænder for clock til portmodul E

PTE22 alt. 4: UART2\_TX  
PTE23 alt. 4: UART2\_RX

8 høje bits og 8 lave bits i baud rate generator

UART2 Control register 2  
Tænder for TX (TE) og RX (RE) igen

```
void UART2_Transmit_Poll(uint8_t data) {
    // wait until transmit data register is empty
    while (!(UART2->S1 & UART_S1_TDRE_MASK))
        ;
    UART2->D = data;
}
```

Pos. for flaget, der går op, når data register er klar til at blive skrevet til



# HW5: SERIEL KOMMUNIKATION OPSUMMERING OG PERSPEKTIVERING

---

Sende/modtage på interrupts

- Buffer -> cirkulær FIFO til hver retning

Message parsing og protokoller/specs (fx NMEA til sonarer, TSIP til GPS-modtagere, osv.)

- endianness (network byte order big-endian)
- binær til struktureret data (ascii, utf-8, tal, osv...)

Fejlhåndtering, støjimmunitet

- Differentielt signal, højere spændinger

Andre kommunikationstyper: CAN-bus 4. semester ☺

# HW6: ANALOG INTERFACING

## INDHOLD

---

1. Formål med analogt interfacing
2. ADC'en i KL25Z
3. KL25Z eksempel på analogt input via ADC
  1. Overvejelser omkring konverteringstid
4. Hardware triggering
5. Opsætning af pins (pinmux)
6. Opsummering og perspektivering

### **6: Analog interfacing**

- 1) Giv et kort overblik over Analog til Digital Converteren (ADC) i KL25, suppler evt. med et kodeeksempel, der viser hvordan converteren anvendes.
- 2) Forklar hvordan **Hardware triggering** kan anvendes i forhold til ADC'en.
- 3) Hvordan sættes KL25 processorens Ben (Pins) op når de skal anvendes til f.eks. Analogt input eller som GPIO.

# HW6: ANALOG INTERFACING

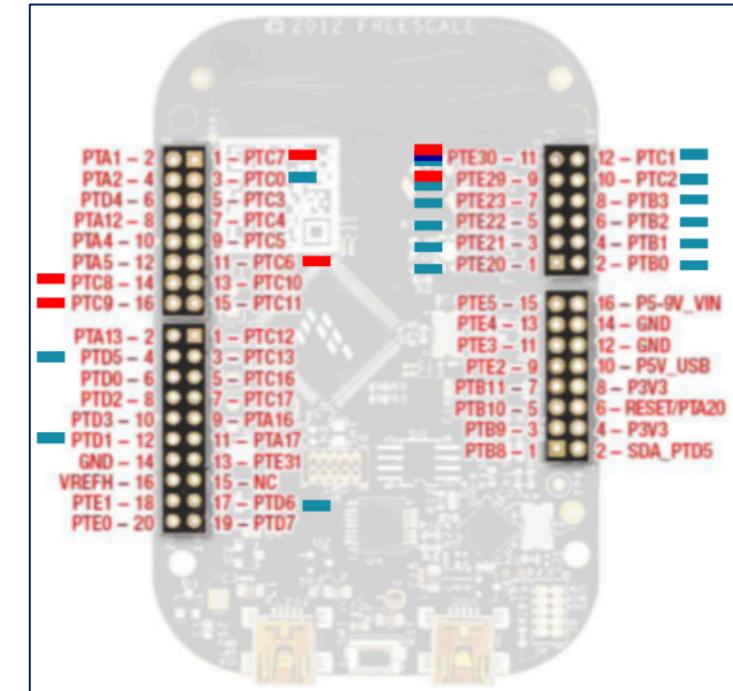
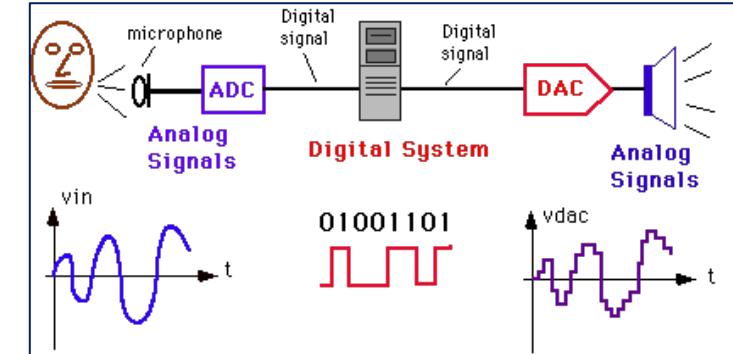
## ANALOG CONVERSION KL25 (ADC)

Formål med analog interfacing:

- Oversætte mellem digital og analog elektronik.
  - Måle på / interagere med den fysiske verden via analog elektronik
  - Fysiske fænomener som en kontinuert/analog spændingssignal (kontinuert i tid og skala):
    - Temperatur, tryk, lysstyrke/intensitet, lydtryk, acceleration/vibration, udsvingning, osv.
- Ind: Oversætte kontinuert spænding til et diskret/digitalt signal (sampling + kvantisering); ADC, CMP.
- Ud: Generere analoge signaler; DAC.

Hardware til rådighed:

- Ind: 1x **ADC** (14 kanaler: 14xSE, 4xDI), 1x Comparator (6 inputs )
- Ud: 1x DAC (1 output )



# HW6: ANALOG INTERFACING

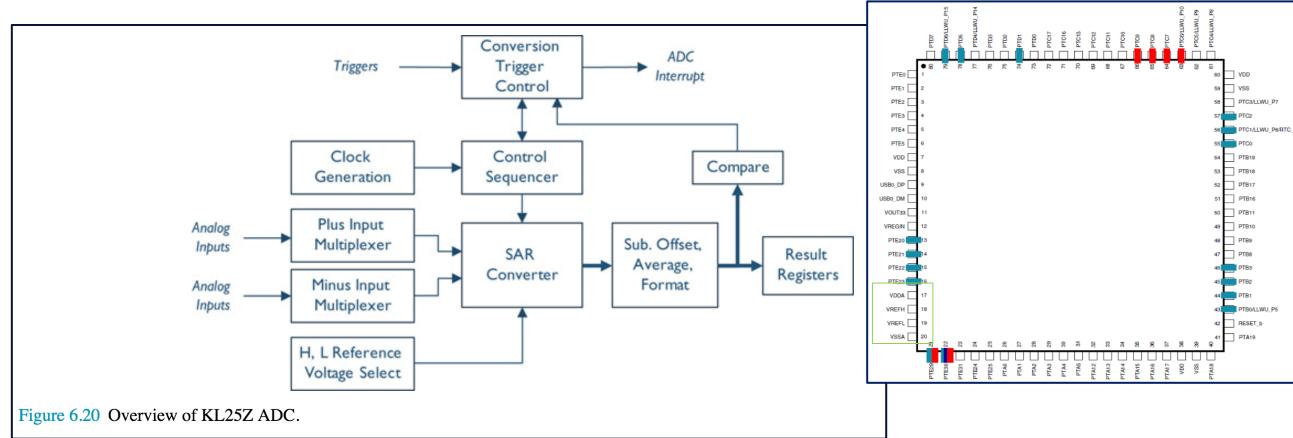
## ANALOG CONVERSION KL25 (ADC)

### Konversion A/D

- Vælg +/- referencer
- Vælg opløsning = præcision
- Vælg clockfrekvens
- Single-ended vs. differentialsig.
- Digital kode repræsenterer analogt interval
- Interpolation til at udregne det analoge interval for fysisk param.

### Andre overvejelser:

- Type er SAR-konverter så
  - Binærsgning tager N steps.
  - vs. kvantiseringsfejl sfa. N
- Samplingsfrekvens ift. maks. frek.
  - Anti-aliaseringsfiltre



#### General Equation

$n$  = converted code

$V_{in}$  = sampled input voltage

$V_{+ref}$  = upper voltage reference

$V_{-ref}$  = lower voltage reference

$N$  = number of bits of resolution in ADC

$$n = \left\lfloor \frac{(V_{in} - V_{-ref}) 2^N}{V_{+ref} - V_{-ref}} + 1/2 \right\rfloor$$

$\lfloor X \rfloor = I$       floor function: nearest integer  $I$  such that  $I \leq X$   
 $\text{floor}(x+0.5)$  rounds  $x$  to the nearest integer

$$n = \left\lfloor \frac{3.30 \text{ v } 2^{10}}{5 \text{ v }} + 1/2 \right\rfloor = 676$$

#### General Equation

$n$  = converted code

$V_{in\_min}$  = minimum input voltage for code  $n$

$V_{in\_max}$  = maximum input voltage for code  $n$

$V_{+ref}$  = upper voltage reference

$V_{-ref}$  = lower voltage reference

$N$  = number of bits of resolution in ADC

$$V_{in\_min} = \frac{n - 1}{2^N} (V_{+ref} - V_{-ref}) + V_{-ref}$$

$$V_{in\_max} = \frac{n + 1}{2^N} (V_{+ref} - V_{-ref}) + V_{-ref}$$

#### Simplification with $V_{-ref} = 0 \text{ V}$

$$V_{in\_min} = \frac{n - 1}{2^N} (V_{+ref})$$

$$V_{in\_max} = \frac{n + 1}{2^N} (V_{+ref})$$



# HW6: ANALOG INTERFACING

## ANALOG CONVERSION KL25 (ADC)

### Steps...

#### Initialisering

- Clock til porte og ADC
- Vælg generelle indstillinge
- Konfigurer conv.clock (ADCK)
- Vælg Vref
- Vælg trigger (HW/SW)
- Konfigurer inputkanal/pin
- *Burde kalibrere her...*

#### Konvertering

- SW trigger pba. mask
- Poll COCO-bit (bedre: IRQ)
- Udlæs resultat, cast til def. bitlænge

#### Resultat:

- Koder mellem ~350 (støj) og 65536 ( $2^{16}-1$ )

### Initialisering

```
11/*  
12 * @author Janus Bo Andersen  
13 * @date 2019-2020  
14 * @param channel adc_channel (corresp. to book p171 and mapg 464, pinmap)  
15 * @return void  
16 * @brief this function initializes the ADC.  
17 */  
18  
19void adcInit(adc_channel channel)  
20 {  
21     /* Clock source for port B & C */  
22     SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK;  
23     SIM->SCGC5 |= SIM_SCGC5_PORTC_MASK;  
24  
25     /* Connect clock source for ADC */  
26     SIM->SCGC6 |= SIM_SCGC6_ADC0_MASK;  
27  
28     /* Setup the converter part 1 */  
29     ADC0->CFG1 = ADC_CFG1_ADLPC_MASK |           // Low power config  
30         ADC_CFG1_DLTSMP_MASK |                  // Long sample time  
31         ADC_CFG1_MODE(3) |                      // 16-bit, single-ended  
32         ADC_CFG1_ADICLK(0);                    // ADCK is Bus Clock  
33  
34     /* Setup the converter part 2 (redundant but explicit method) */  
35     ADC0->SC2 = ADC_SC2_REFSEL(0);             // Voltage ref. is VREFH, VREFL  
36     ADC0->SC2 &= ~ADC_SC2_ADRG_MASK;           // Software trigger  
37     ADC0->SC2 &= ~ADC_SC2_ACFC_MASK;           // Compare func. disabled  
38     ADC0->SC2 &= ~ADC_SC2_DMAEN_MASK;          // DMA disabled  
39  
40     /* Set up input for ADC input */  
41     switch (channel) {  
42         case AD15: //PTC1  
43             PORTC->PCR[1] = 0x00000000;           // Totally clear PCR (precaution)  
44             PORTC->PCR[1] &= ~PORT_PCR_MUX_MASK;    // Clear MUX for PTC1  
45             PORTC->PCR[1] |= PORT_PCR_MUX(0);       // Set up for analog input  
46             break;  
47     }  
48 }
```

### Konvertér og udlæs

```
53 * @author Janus Bo Andersen  
54 * @date 2019-2020  
55 * @param channel adc_channel (corresp. to book p171 and mapg 464, pinmap)  
56 * @brief this function from a channel on ADC  
57 * @return unsigned short with the 16-bit code  
58 */  
59unsigned short adcRead(adc_channel channel)  
60 {  
61     /* Warning: This implementation requires that adcInit(channel) has been run */  
62  
63     unsigned short res;           // Result code from conversion  
64     uint32_t start_mask = 0UL;    // Mask used to trigger conversion  
65  
66     /* Set up for ADC input */  
67     switch (channel) {  
68         case AD15: //PTC1  
69             start_mask = 15UL;           // Channel 15 (0b01111)  
70             start_mask &= ~ADC_SC1_DIFF_MASK; // Single-ended mode (disable DIFF)  
71             start_mask &= ~ADC_SC1_AIEN_MASK; // No interrupts (disable AIEN)  
72             break;  
73     }  
74  
75     /* Software-triggering consists of writing to SC1A */  
76     ADC0->SC1[0] = start_mask;  
77  
78     /* Poll - busy waiting until COCO (Conv. Completed) */  
79     /* TODO: Implement IRQ approach */  
80     while (!(ADC0->SC1[0] & ADC_SC1_COCO_MASK)) {  
81         // zzzz  
82     }  
83  
84     /* result: the 32-bit register, casted to unsigned short,  
85      * as unused bits 16:31 are cleared by HW */  
86     res = (unsigned short) ADC0->R[0];  
87  
88     return res;  
89 }
```

# HW6: ANALOG INTERFACING

## OVERVEJELSER OM KONVERTERINGSTID

Som udgangspunkt:

- 16-bit for bedst præcision
- single-ended eller diff.  
afhængigt af behov
- Ekstra features efter behov

Hvis hastighed er en faktor:

- Undgå low-power conversion
- Justér præcision
- Vælg single-ended
- Undgå at bruge adders
- Undgå at bruge averaging

Precision (bits)	Mode	Base Conversion Time BCT
8	single-ended	17 ADCK cycles
9	differential	27
10	single-ended	20
11	differential	30
12	single-ended	20
13	differential	30
16	single-ended	25
16	differential	34

Eksempel: Differential, 16-bit:  
 $f_{ADC} = 12 \text{ MHz} \Rightarrow T_{ADC} = 83 \text{ ns/cycle}$   
 $T_{BCT} = 34 \text{ cycles} \times 83 \text{ ns / cycle} = 2.8 \mu\text{s}$

$2.8 \mu\text{s} \sim 136 \text{ CPU cycles @ 48 MHz.}$

# HW6: ANALOG INTERFACING

## HARDWARE TRIGGERING

Hvordan kan hardware triggering anvendes?

Trigger: udløser en ADC-måling. HW eller SW.

Muligheder for HW-signaler som triggers:

- Ekstern udløser
- Tid: TPM, PIT, RTC, LPTMR
- Comparator

Metode:

- Vælg HW-trigger i feltet ADTRG i ADC'ens Status and Control Register 2 (ADC0\_SC2)
- Vælg trigger-kilde i feltet ADC0TRGSEL fra Systems Options 7 (SIM\_SOPT7)

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									ADACT							
W									ADTRG							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6	ADTRG	Conversion Trigger Select
		Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable:
0	Software trigger selected.	• Software trigger: When software trigger is selected, a conversion is initiated following a write to SC1A.
1	Hardware trigger selected.	• Hardware trigger: When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWTn input after a pulse of the ADHWTSn input.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									ADC0TRGSEL							
W									N	0						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

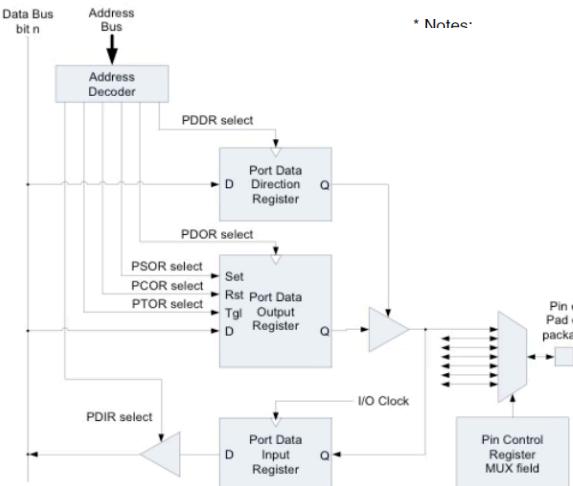
3-0	ADC0TRGSEL	ADC0 trigger select
		Selects the ADC0 trigger source when alternative triggers are functional in stop and VLPS modes. .
0000	External trigger pin input (EXTRG_IN)	
0001	CMP0 output	
0010	Reserved	
0011	Reserved	
0100	PIT trigger 0	
0101	PIT trigger 1	
0110	Reserved	
0111	Reserved	
1000	TPM0 overflow	
1001	TPM1 overflow	
1010	TPM2 overflow	
1011	Reserved	
1100	RTC alarm	
1101	RTC seconds	
1110	LPTMR0 trigger	
1111	Reserved	

# HW4: ANALOG INTERF.

## OPSÆTNING AF PIN

Enkelt pins (ikke bus)

- Mux sætter fysisk forbindelse
- Pin control register MUX bits styrer funktion
- Forskellige pins har forskellige "muligheder" -> pinouts
- Analog/ADC:
  - Alternativ 0 (0b0)



Bit	31	30	29	28	27	26	25	24		23	22	21	20	19	18	17	16
R			0					ISF		0							
W								w1c									
Reset	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0
Bit	15	14	13	12	11	10	9	8		7	6	5	4	3	2	1	0
R			0					MUX		0	DSE	0	PFE	0	SRE	PE	PS
W										x*	x*	x*	0	x*	0	x*	x*
Reset	0	0	0	0	0	x*	x*	x*		0	x*	0	x*	0	x*	x*	x*

\* Notes:

MUX (bits 10-8)	Configuration
000	Pin disabled (analog)
001	Alternative 1 – GPIO
010	Alternative 2
011	Alternative 3
100	Alternative 4
101	Alternative 5
110	Alternative 6
111	Alternative 7

FROM KL25Z Pins				KL25Z128 Pins											
On-board Usage	I/O Header & Pin Num	Arduino™ R3 Pin Name	FRDM-KL25Z Pin Name	KL25Z Pin #	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	Reset State/Function		
--	J2 20	D14	PTE0	1		PTE0		UAR71_T0	RTC_CLKOUT	CMP2_OUT	I2C1_SDA			DISABLED	
--	J2 18	PTE1	PTE1	2		PTE1		SP11_MOSI	UAR71_RX	SP11_MISO	I2C1_SCL			DISABLED	
--	J2 09	PTE2	PTE2	3		PTE2		SP11_SCK						DISABLED	
--	J2 11	PTE3	PTE3	4		PTE3		SP11_MSO						DISABLED	
--	J2 13	PTE4	PTE4	5		PTE4		SP11_PCS0						DISABLED	
--	J2 15	PTE5	PTE5	6		PTE5								DISABLED	
Power	--	VDD	VDD	7										VDD	
Power	--	VSS	VSS	8										VSS	
USB	--	USB0_DP	USB0_DP	9										USB0_DP	
2.2uf cap	--	USB0_DM	USB0_DM	10										USB0_DM	
USB VBUS (5V)	--	VOUT133	VOUT133	11										VOUT133	
--	J10 01	PTE20	ADC0_DR/DAC0_SE0	13		PTE20								ADC0_DR/DAC0_SE0	
--	J10 03	PTE21	ADC0_DM/ADC0_SE4a	14		PTE21								ADC0_DM/ADC0_SE4a	
--	J10 05	PTE22	ADC0_DR/DAC0_SE3	15		PTE22								ADC0_DR/DAC0_SE3	
--	J10 07	PTE23	ADC0_DM/ADC0_SE7a	16		PTE23								ADC0_DM/ADC0_SE7a	
Power	--	VDDA	VDDA	17										VDDA	
Power	J2 16	AREF*	VREFH	18										VREFH	
Power	--	VREFL	VREFL	19										VREFL	
Power	--	VSSA	VSSA	20										VSSA	
--	J10 09	PTE29	CMP0_IN5/ADC0_SE4b	21		PTE29								CMP0_IN5/ADC0_SE4b	
--	J10 11	PTE30	DAC0_OUT/ADC0_SE23/CMP0_IN4	22		PTE30								DAC0_OUT/ADC0_SE23/CMP0_IN4	

# HW6: ANALOG INTERFACING

## OPSUMMERING OG PERSPEKTIVERING

---

# SW1: C++ KLASSER OG OOP

## INDHOLD

---

1. Klasser (eks. fra uge 38)
2. Nedarvning
3. Access specifiers
4. Constructors/destructors
  1. Ansvar for at rydde op?

1	C++	Forklar <ul style="list-style-type: none"><li>• hvad klasser er,</li><li>• hvordan constructor og destructor fungerer og</li><li>• access specifiers (private, protected og public)</li><li>• hvorledes klasser kan arve, single inheritance,</li></ul>
---	-----	---

# SW1: C++ KLASSER OG OOP

## HVAD ER KLASSE

### Options.h (definition af type og interface)

Objektorienteret koncept: Software-objekter, der modsvarer "virkelighedens" objekter...

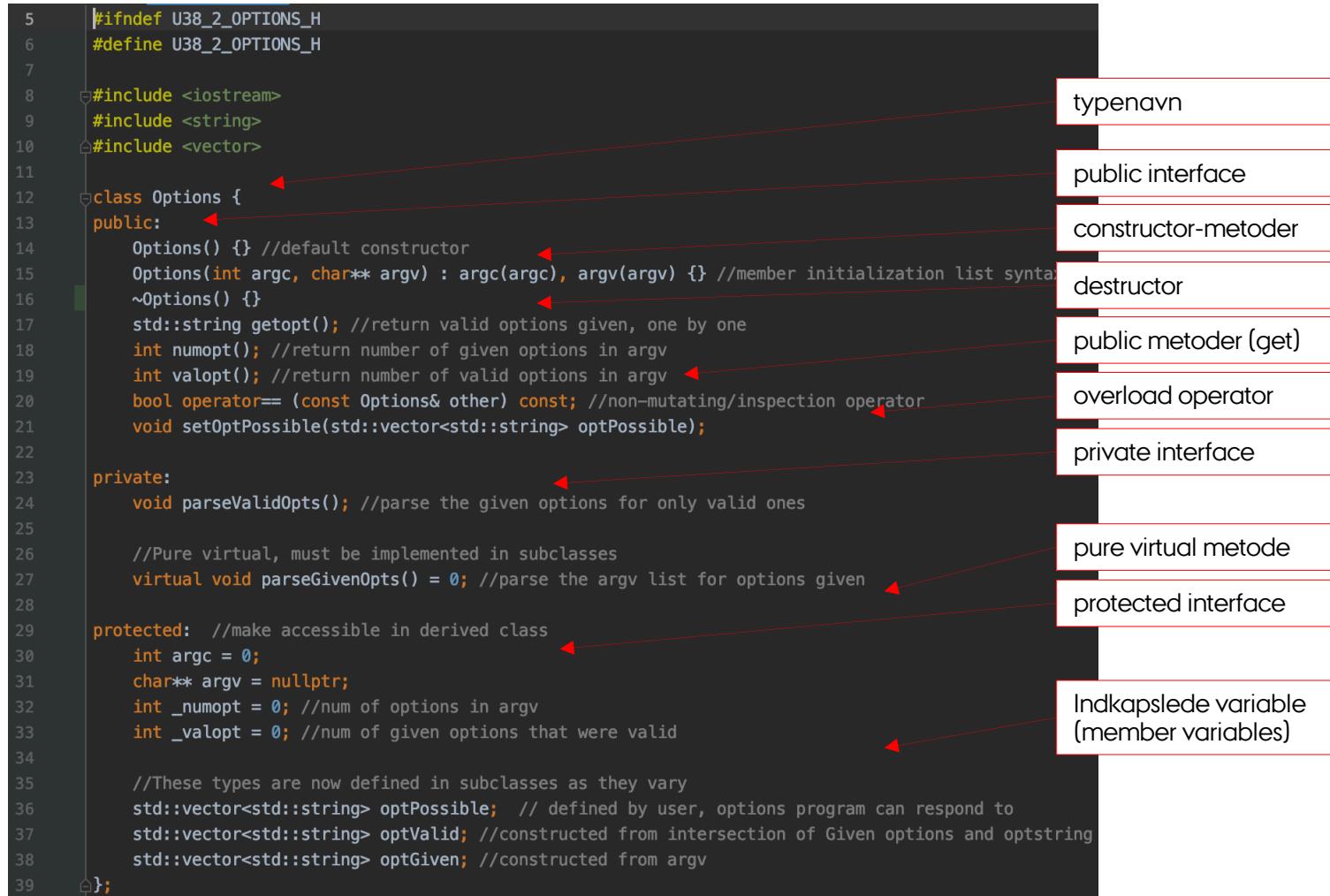
- Indkapsling, "metoder", nedarvning

C++ typesystem:

- Brugerdefinerede komplekse typer:  
Klasser og structs
- Adskille interface og implementation
- Beskytte variable/datastrukturer

Vores ArgParser "Options" fra uge 38

```
5  ifndef U38_2_OPTIONS_H
6  define U38_2_OPTIONS_H
7
8  include <iostream>
9  include <string>
10 include <vector>
11
12 class Options {
13 public:
14     Options() {} //default constructor
15     Options(int argc, char** argv) : argc(argc), argv(argv) {} //member initialization list syntax
16     ~Options() {}
17     std::string getopt(); //return valid options given, one by one
18     int numopt(); //return number of given options in argv
19     int valopt(); //return number of valid options in argv
20     bool operator== (const Options& other) const; //non-mutating/inspection operator
21     void setOptPossible(std::vector<std::string> optPossible);
22
23 private:
24     void parseValidOpts(); //parse the given options for only valid ones
25
26     //Pure virtual, must be implemented in subclasses
27     virtual void parseGivenOpts() = 0; //parse the argv list for options given
28
29 protected: //make accessible in derived class
30     int argc = 0;
31     char** argv = nullptr;
32     int _numopt = 0; //num of options in argv
33     int _valopt = 0; //num of given options that were valid
34
35     //These types are now defined in subclasses as they vary
36     std::vector<std::string> optPossible; // defined by user, options program can respond to
37     std::vector<std::string> optValid; //constructed from intersection of Given options and optstring
38     std::vector<std::string> optGiven; //constructed from argv
39 }
```



# SW1: C++ KLASSER OG OOP

## IMPLEMENTERING

### Implementering

- Implementerer ind i klassen som et namespace
- Interface skal være stabilt
- Implementering kan ændres, optimeres

Hvis man har overloadet sine constructors, så bør man implementere muligheder for at "nå til samme tilstand"

- fx Options() og Options(argc, argv)...

### Options.cpp (udklip fra implementering)

```
5  #include "Options.h"
6  #include <regex>
7  #include <algorithm>
8  #include <stdexcept>
9
10 //sets a vector that defines valid options, then calls the parser for valid options
11 void Options::setOptPossible(std::vector<std::string> optPossible) {
12     this->optPossible = optPossible; //set private member
13     parseValidOpts(); //now parse and build set of valid options
14     return; //done
15 }
16
17 //Parses given options and finds the valid ones
18 void Options::parseValidOpts() {
19
20     //iterate over the options (allowed opts), and the ones found in Given are put in Valid vector
21     std::vector<std::string>::iterator it; //iterator for find algo
22
23     for (auto option: optPossible) {
24         //check if the option was given at the command line
25         it = std::find(begin(optGiven), end(optGiven), option); //where the c was found
26         if (it != end(optGiven)) //if find did not reach end of vector, c was found
27             optValid.push_back(option); // add the option as it was found in the Given
28     }
29
30     _valopt = optValid.size(); //set the number of Given options that were Valid
31
32     return;
33 }
```

Definerer en tidligere  
erklæret public  
metode

Arbejder på private  
variable

# SW1: C++ KLASSER OG OOP

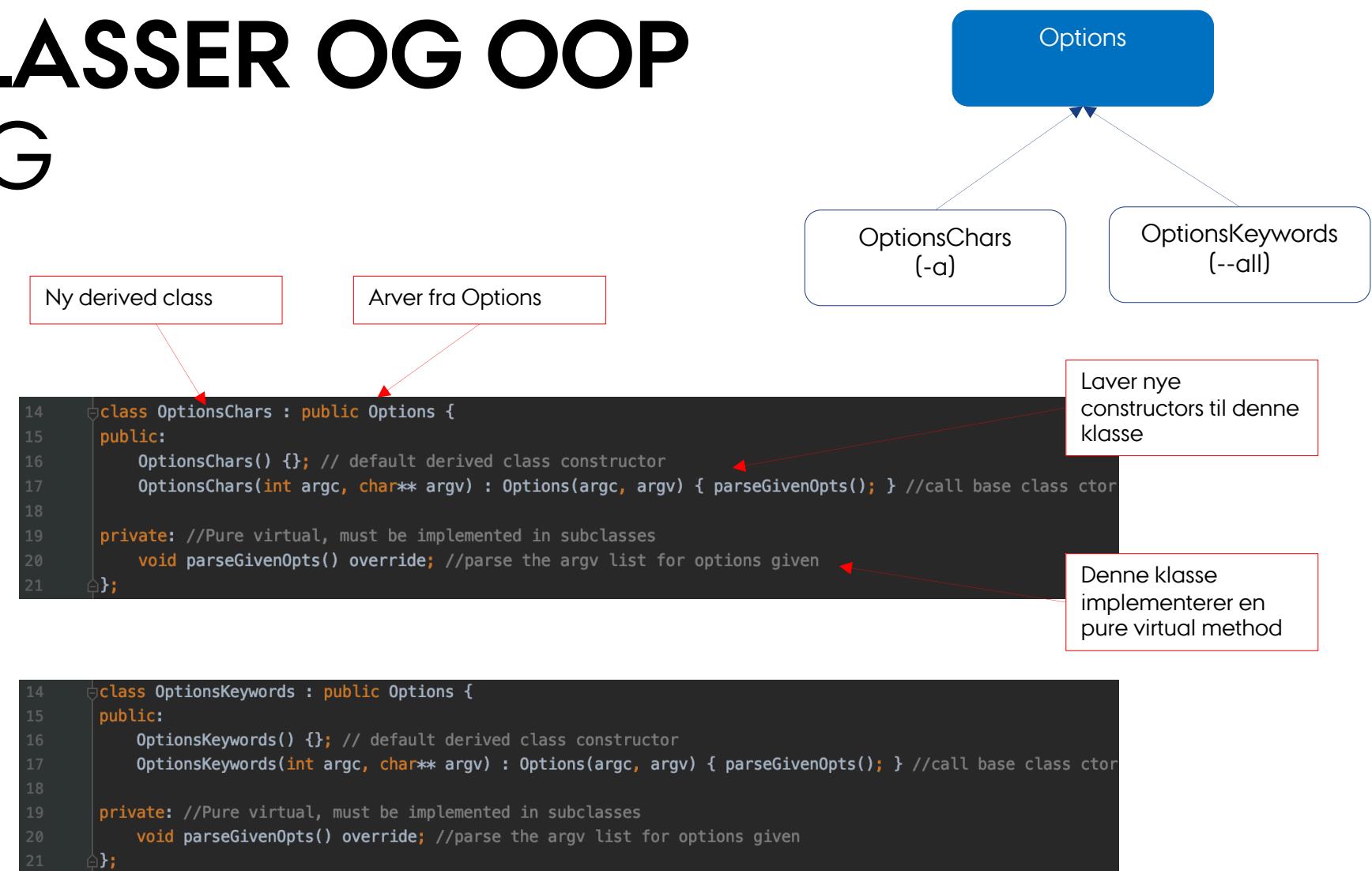
## NEDARVNING

### Nedarvning

- Single inheritance: nedarver metoder, variable fra en anden klasse
  - *Hvordan* nedarvningen sker afhænger af **nedarvningstypen** i den nye klasse (public, private, protected)
  - *Hvad der er adgang til* af variable og metoder fra base class afhænger af **access specifiers** i base class (senere slides)
- Multiple inheritance: arve fra flere klasser på en gang...

### Eksemplet:

- To specifikke options-typer arver "publically" fra Options



# SW1: C++ KLASSEER OG OOP

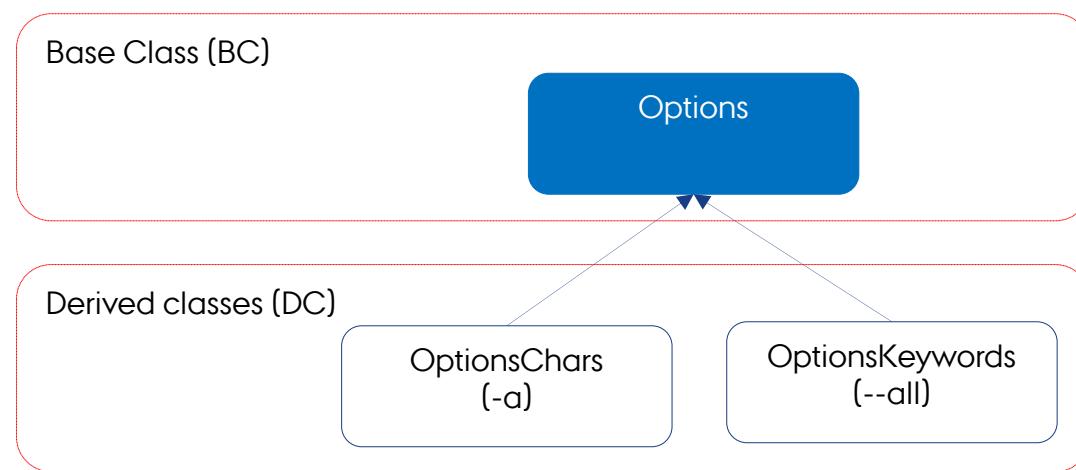
## ACCESS SPECIFIERS

Access Specifiers:

- Adgang til data/metoder i klassen
- **public**: kan tilgås af "alle"
- **protected**: kan tilgås i DC
- **private**: kan kun tilgås fra klassen selv
- Friendship ☺: En klasse defineret som `friend class <navn>` må gerne tilgå private/protected. Den behøver ikke nedarve.

Type af nedarvning:

- Hvordan den nye klasse bygges:
  - **public**: nedarver fra base class sp
    - public fra BC som public i DC
    - protected fra BC som protected i DC
    - private kan jo ikke tilgås...
  - **protected**:
  - **private**:
- public og protected fra BC bliver protected i DC.
  - public og protected fra BC bliver private i DC.



# SW1: C++ KLASSER OG OOP

## CONSTRUCTORS OG DESTRUCTORS

### Constructors:

- Definerer, hvad der skal ske, når en ny instans (objekt) af en klasse skal oprettes
- Copy constructor giver tit mening!

```
17 OptionsKeywords(int argc, char** argv) : Options(argc, argv) { parseGivenOpts(); } //call base class ctor
```

Argumenter til instantiering af nyt objekt

Initializer list:  
Benytter BC constructor til at initialisere

Kalder denne funktion så objektet færdig-initialiseres

### Destructors:

- Definerer, hvordan objektet skal afvikles:  
Oprydning, delete

```
16 ~Options() {}
```

C++'s spec/regler for hvad der sker, når et objekt destrueres, går ud af scope

### Hvornår? Lifetime, ansvar for at rydde op...

- new / delete, versus smartpointers (sharedptr, uniqueptr)...

```
43 std::vector<std::string> optChars {"a", "b", "o"}; //only accept -a -b -o options  
44 std::vector<std::string> optKeywords {"alpha", "beta", "omega"}; //only accept --alpha --beta --omega options  
45  
46 // Instantiate two different Options-derived objects (for demo)  
47 OptionsChars myOptionsChars(argc, argv);  
48 OptionsKeywords myOptionsKeywords(argc, argv);  
49  
50 // We can now access them via polymorphism  
51 std::vector<Options*> opt(2); //vector holds two pointers to options objects  
52 opt[0] = &myOptionsChars;  
53 opt[1] = &myOptionsKeywords;  
54  
55 // We are now working through an Options object pointer (base class)  
56 opt[0]->setOptPossible(optChars);  
57 opt[1]->setOptPossible(optKeywords);
```

Instantierer nye objekter

Arbejder polymorfisk på objekterne

### Kritiske systemer:

- C++ guideline/krav til flysystemer: Ingen delete/free efter takeoff...

### Embedded systemer

- Måske undgå heap/freestore, pga. hukommelsesfragmentering

# SW1: C++ KLASSER OG OOP

## OPSUMMERING OG PERSPEKTIVERING

---



# SW2: C++ OVERLOADING, OOP, POLYMORF.

## INDHOLD

---

1. Overloading af funktioner og metoder
2. Instantiering af objekter
3. Hvad betyder polymorfisme
4. Templates
5. Opsummering og perspektivering

2	C++	Forklar <ul style="list-style-type: none"><li>• function overloading,</li><li>• object instantiation</li><li>• hvad polymorphism er og hvorledes det kan udnyttes</li><li>• hvad templates er og hvordan det kan anvendes i programmeringen</li></ul>
---	-----	---

# SW2: C++ OVERLOADING, OOP, POLYMORF. OVERLOADING AF FUNKTIONER

Ideen:

- Forskellige metode-/funktionsdefinitioner for forskellige "argumentlister" med samme funktionsnavn
  - Virker for *bruger* af API som samme funktion, men forskellige typer/antal af argumenter resulterer i kald til forskellige funktioner
- Endnu bedre, hvis man også har overloadet sine operatorer

Hvis man har overloadet sine constructors, så bør man implementere muligheder for at "nå til samme tilstand"

- fx Options() og Options(argc, argv)...

Eksempler:

- Overloaded constructor
- Overloaded operator
- Vores argparser "Options" fra uge 37/38

```
11
12 class Options {
13 public:
14     Options() {} //default constructor
15     Options(int argc, char** argv) : argc(argc), argv(argv) {} //member initialization list syntax
```

To forskellige  
metoder til at  
instantiere et objekt

```
3 // Find the maximum of two integers
4 int max(int a, int b) {
5     return (a > b) ? a : b;
6 }
7
8 // Find the maximum of two doubles
9 double max(double a, double b) {
10    return (a > b) ? a : b;
11 }
12
13 int main() {
14     //Determine some maxima and give to the user
15     std::cout << max(2,1) << " and " << max(2.1,2.2) << std::endl;
16     return 0;
17 }
```

Rigtig fedt at kunne beregne max uden at skulle tænke alt for meget over hvilken type...  
Implicit/explicit casting er også muligheder

```
94 //non-mutating/inspection operator
95 //compares all data items for this and other: this==other
96 bool Options::operator== (const Options& other) const {
97
98     //comparison list is very redundant, only first three needed... Others are then invariants.
99     if (    this->argc == other(argc)           // same command line argument length
100        && this->argv == other(argv)             // same command line argument mem location
101        && this->optString == other.optString   // started with identical strings
102        && this->optGiven == other.optGiven     // command line options have been parsed (always true if above)
103        && this->optValid == other.optValid     // cmd line options have been filtered with optstring (same)
104        && this->_numopt == other._numopt       // this should always be true as a consequence of above
105        && this->_valopt == other._valopt) {      // same...
106            return true; //similar
107        } else {
108            return false; //not similar
109        }
110 }
```

Tilbage i options-  
eksemplet...  
Overloader operator

# SW2: C++ OVERLOADING, OOP, POLYMORF. INSTANTIERING

## Instantiering:

- Kald til constructor
- Valg af allokering
  - lokal/automatisk variabel
  - heap/freestore-variabel

## Scope, lifetime, memory leaks, ansvar for oprydning

- new / delete, versus smartpointers  
(`sharedptr`, `uniqueptr`)...

## Kritiske systemer:

- C/C++ guideline/krav til flysystemer: Ingen delete/free efter takeoff...

## Embedded systemer

- Måske undgå heap/freestore, pga. hukommelsesfragmentering

```
17 OptionsKeywords(int argc, char** argv) : Options(argc, argv) { parseGivenOpts(); } //call base class ctor
```

Argumenter til instantiering af nyt objekt

Initializer list:  
Benytter BC constructor til at initialisere

Kalder denne funktion så objektet færdig-initialiseres

```
16 ~Options() {} ◀
```

C++'s spec/regler for hvad der sker, når et objekt destrueres, går ud af scope

```
8 #include <iostream>
9 #include <string>
10 #include "Options.h"
11
12 int main(int argc, char* argv[]) {
13
14     std::string optString {"abo"}; //only accept -a -b -o options
15
16     //allocate object on the heap, modern C++ style
17     auto myOptions = std::make_unique<Options>(argc, argv); //make a unique smart pointer (RAII)
18
19     myOptions->setOptstring(optString); //set the allowed options in the object, also parses them
20
21     std::cout << "Number of GIVEN options in command line: " << myOptions->numopt() << std::endl;
22     std::cout << "Number of VALID options in command line: " << myOptions->valopt() << std::endl;
23
24     //popping the options one by one...
25     for (unsigned i = 0; i < myOptions->valopt(); ++i) {
26         std::cout << "Valid option was set: " << myOptions->getopt() << std::endl;
27     }
28
29     return 0; ◀
30 }
```

pointer-typen bliver også automatisk infereret

Smartptr tilgang til at instantiere med allokering på heap. Det er nu C++'s problem, at der bliver ryddet op...  
RAII-binding

RAII garanterer de-allokering her!

# SW2: C++ OVERLOADING, OOP, POLYMORF. POLYMORFISME

## Polymorfisme

- Options-klassen er polymorf til de to derived klasser (de nedarver fra Options)
  - Den kan altså *antage* form som enten det ene eller det andet.
  - Vi har implementeret pure virtual-funktioner – som skal implementeres i en derived klasse.
  - Options er i dette tilfælde en *abstract base class*. Dens interface skal implementeres fuldt ud af en derived klasse.
- Når vi tilgår Options-klassen, så benyttes nedarvnings-hierarkiet til at afgøre, hvilken "form", der er tale om...

```
43     std::vector<std::string> optChars {"a", "b", "o"}; //only accept -a -b -o options
44     std::vector<std::string> optKeywords {"alpha", "beta", "omega"}; //only accept --alpha --beta --omega options
45
46     // Instantiate two different Options-derived objects (for demo)
47     OptionsChars myOptionsChars(argc, argv);
48     OptionsKeywords myOptionsKeywords(argc, argv); ← Instantierer nye objekter som lokale variabler
49
50     // We can now access them via polymorphism
51     std::vector<Options*> opt(2); //vector holds two pointers to options objects ← Pointer til base class!
52     opt[0] = &myOptionsChars;
53     opt[1] = &myOptionsKeywords;
54
55     // We are now working through an Options object pointer (base class)
56     opt[0]->setOptPossible(optChars);
57     opt[1]->setOptPossible(optKeywords); ← Arbejder polymorfisk på objekterne
58
59
60     //Pure virtual, must be implemented in subclasses
61     virtual void parseGivenOpts() = 0; //parse the argv list for options given ← pure virtual metode i Options-klassen Skal implementeres!
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
```

/\* E3ISD1 Week 38/2 -- Exercise 8  
\* Parse and shows valid options from command line with pattern like "-h" and "--help" (GNU style)  
\*  
\* This week we work with Polymorphism and Exception handling.  
\*  
\* Polymorphism  
\* - A class that declares or inherits a virtual function is called a polymorphic class.  
\* -> So the Options class is polymorphic to its two derived classes.  
\* -> It declares pure virtual functions that are only implemented in derived classes.  
\* -> The Options base class is an abstract base class.  
\* - To demonstrate polymorphism, we access the derived classes through pointers to a base class type.

Nogen har været søde at huske at skrive en forklaring i koden... ☺

```
graph TD; Options --> OptionsChars["OptionsChars (-a)"]; Options --> OptionsKeywords["OptionsKeywords (--all)"]
```

# SW2: C++ OVERLOADING, OOP, POLYMORF. TEMPLATES

## Templates

- Generisk definition, mange anvendelser
- Både funktioner og klasser
- C++ compileren udleder hvilke funktionskopier, der skal generes

## Anvendelser

- Over det hele i moderne C++, STL er dybt afhængigt af Templates
- "Generic programming", ikke-typeafhængige algoritmer og datastrukturer
  - Fx et generisk binærtræ, der kan indeholde alt muligt...
- Don't repeat yourself...

```
43     std::vector<std::string> optChars {"a", "b", "o"}; //only accept -a -b -o options
```

Containeren std::vector er defineret generisk, vi bruger template-argumentet til at angive hvilken type/klasse der skal indeholdes

```
3  // Find the maximum of operands with > operator
4  template <typename T>
5  T max(T a, T b) {
6      return (a > b) ? a : b;
7  }
8
9  int main() {
10     //Determine some maxima and give to the user
11     std::cout << max(2,1) << ", " << max("Kaffe", "Kage") << std::endl;
12     return 0;
13 }
```

Definerer nu max-funktionen som en template

- Kan fungere for alle typer med operatoren '>' implementeret
- Operator overloading vil altså give mulighed for at anvende denne på i principippet alle typer.

C++ compileren udleder selv hvilken funktion, der skal kaldes

## Ulemper:

- Aggressiv anvendelse kan give stor kode (mange kopier af templatekode med type fyldt ud...)

# SW2: C++ OVERLOADING, OOP, POLYMORF. OPSUMMERING OG PERSPEKTIVERING

---

# SW3: MAKE

## INDHOLD

---

1. Formål og metode med Make
2. Eksempel på automatisering
3. Eksempel på mere avanceret Makefile
4. CMake til at lave cross-platform Makefiles
5. Opsummering og perspektivering

3	make	Forklar hvordan make og en makefile fungerer
---	------	--

# SW3: MAKE FORMÅL OG METODE

Ideen:

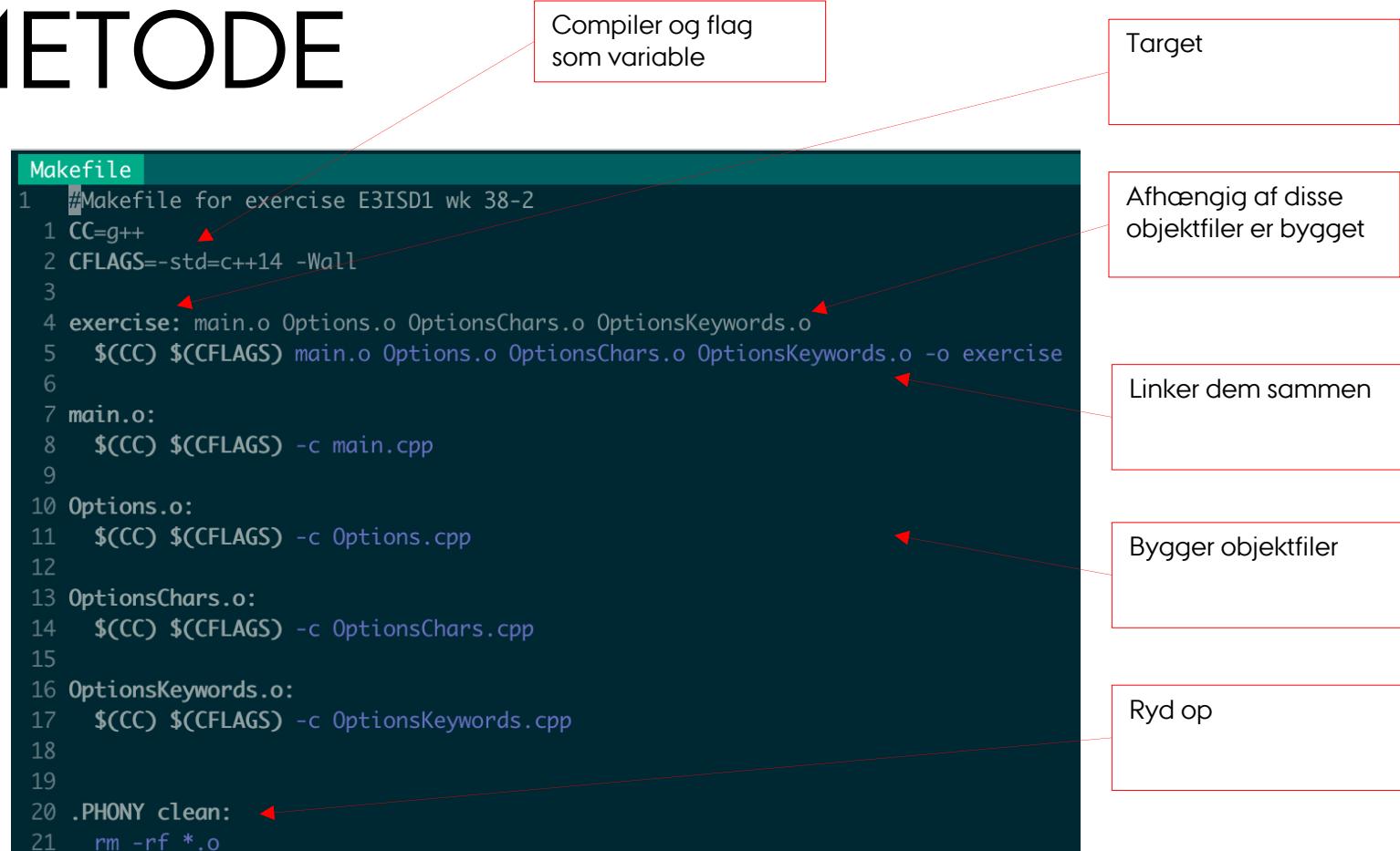
- Automatisk build
  - Compiling, linking
- Definér targets, , include libraries, compiler, compiler-flag osv. én gang
  - Definer evt. midlertidige compile targets
- Kør simpel kommando for at gentage noget kompliceret (jeg har vist det overdrevet kompliceret!)
- Nemt at dele/distribuere

Tilgang:

- make er GNU utility (Eclipse bruger også make)
- Makefile er tekst-definition på build (husk! skal være tabs!)

Anvendelser

- Behøver ikke at være compiling, kan også være opgaveafvikling...



```
~/git/E3ISD1-SW/u38_2(master*) » make
g++ -std=c++14 -Wall -c main.cpp
g++ -std=c++14 -Wall -c Options.cpp
g++ -std=c++14 -Wall -c OptionsChars.cpp
g++ -std=c++14 -Wall -c OptionsKeywords.cpp
g++ -std=c++14 -Wall main.o Options.o OptionsChars.o OptionsKeywords.o -o exercise
janusboandersen@Januss-MacBook-Pro
```

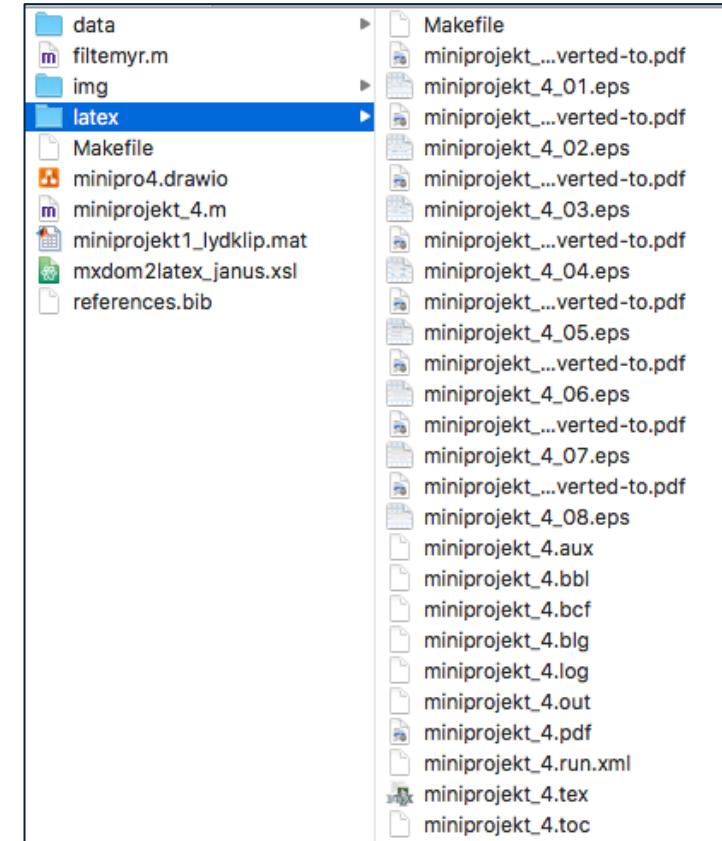
# SW3: MAKE AUTOMATISERING

## Eksemplet

- Automatisering af LaTeX afvikling
- Skal køre flere gange pga. indholdsfortegnelse, bibliografi, citater, osv.
- Makefile, der refererer til Makefile

```
Makefile
1 all:
  1 $(MAKE) -C latex all
```

```
Makefile
1 all:
  1 lualatex miniprojekt_4.tex
  2 biber miniprojekt_4.bcf
  3 lualatex miniprojekt_4.tex
  4 lualatex miniprojekt_4.tex
  5 open miniprojekt_4.pdf
  6
  7 clean:
```



# SW3: MAKE LIDT MERE AVANCERET

## Eksemplet

- Kan fuldautomatisere byggeproces
- Benytte kommandoer til at afvikle shell-funktioner
- Bruges med en mere "best-practice" biblioteksstruktur

```
Makefile
41 # ----- Target, build directory and source directory -----
40 # set target executable file. It will be compiled into the build dir
39 TARGET_EXEC ?= a.out
38
37 # Set directories for placing built files and intermediate object files
36 BUILD_DIR ?= ./build
35
34 # Set directory to search for source code
33 SRC_DIRS ?= ./src
32
31
30 # ----- Compilers and versions -----
29 # These need likely not be explicitly defined, but are done so here for stability
28 # Using GNU compilers as they seem to produce similar results across platforms
27 # On Mac OS, gcc and g++ are wrapped by Clang
26
25 # C compiler and flags
24 CC := gcc
23
22 # C++ compiler and flags
21 CXX := g++
20 # CXX_FLAGS := -Wall -Werror -std=c++11
19
18
17 # ----- Build dependency / prerequisites tree -----
16
15 # Source code filenames to map in dependency tree and to compile
14 SRCS := $(shell find ${SRC_DIRS} -name *.cpp -or -name *.c -or -name *.s)
13
12 # Object file names to map in dependency tree and to give to linker
11 # Prefixes the build_dir, and postfixes .o (main.cpp > build/main.cpp.o)
10 OJBS := ${SRCS:=%(BUILD_DIR)%.o}
9
8 # Dependency list files, one dependency file per source file (main.cpp.o -> main.cpp.d)
7 DEPS := ${OJBS:.o=.d}
6
5
4 # ----- Pre-processor -----
3 # Specify include directories to search for header files during pre-processing
2 INC_DIRS := $(shell find ${SRC_DIRS} -type d)
1
67 # Build the flag for all custom include directories ( -I<dir1> -I<dir2> -I<dir3> ... )
NORMAL Makefile                                     make
```

Prøver at have en fornuftig  
biblioteksstruktur: /src, /build osv

Variabel defineres så højresiden  
evalueres

% er wildcards  
Finder all objektfiler i build\_dir

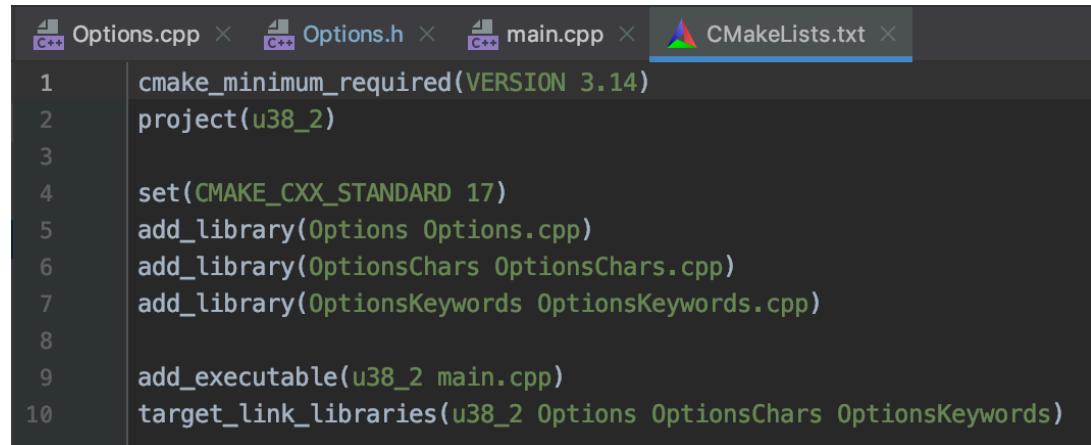
Omdøber ud fra skabelone

# SW3: MAKE CMAKE

---

CMake:

- Nyere cross-platform Makefile-maker
- Definerer projektet som et objekt
  - Sætte attributter
- Fordel:
  - Forskellige platforme kan have dependencies forskellige steder, forskellige navne, osv
  - Makroer (defineret af folkene bag Cmake og andre contributors), definerer hvordan en hver platform fungerer
- CMakeLists.txt definerer
  - -> cmake laver en Makefile
  - -> make afvikler build



```
1 cmake_minimum_required(VERSION 3.14)
2 project(u38_2)
3
4 set(CMAKE_CXX_STANDARD 17)
5 add_library(Options Options.cpp)
6 add_library(OptionsChars OptionsChars.cpp)
7 add_library(OptionsKeywords OptionsKeywords.cpp)
8
9 add_executable(u38_2 main.cpp)
10 target_link_libraries(u38_2 Options OptionsChars OptionsKeywords)
```

# SW3: MAKE

# OPSUMMERING OG PERSPEKTIVERING

---

Opsummering

Andre buildsystemer

- Rake -> Ruby-baseret automatisering
- Autotools -> GNU build-system

# SW4: BEAGLEBONE STARTUP, I/O, M.V.

## INDHOLD

---

1. Boot-up & Shutdown
2. Shell scripts
  1. Anvendelse og eksempel med GPIO
  2. Øvrigt eksempel
3. I/O
  1. Overblik over I/O muligheder på PB
  2. I/O: GPIO og PWM
  3. I/O: I2C og ADX345
  4. Overblik over seriell kommunikation
4. Opsummering og perspektivering

4	BeagleBone Black/PocketBeagle (forkortet BBB)	Forklar <ul style="list-style-type: none"><li>• hvad der sker når en BBB bootes - og når den lukkes rigtigt ned</li><li>• hvad shell-scripts er og hvor de anvendes gerne med eksempler</li><li>• hvordan skabes der adgang til IO (gpio, i2c, can, pwm, etc.)</li></ul>
---	---	--

# SW4: BEAGLEBONE STARTUP, I/O, M.V. BOOT-UP

## Hovedbudskaber

Der kører en opstartsrutine inden man kan få adgang til PB/BB via seriel/SSH

- "Dum" processer/hardware -> initialiseret hardware -> kernel loadet i hukommelse -> userspace vi kan interagere med...
- Ingen UEFI/BIOS på BB ->
- Resetvektor: AM335x's startup ROM (clock+hukom.) -> Bootloader
  - -> U-Boot bootstrapper via filer på PB's SD-kort
    - Initialisering og test af hardware (device tree\*)
    - Loader tidligt filesystem til hukommelse (ramdisk)
    - Loader kernen til hukommelsen
      - -> systemd tager så over...

Hvorfor er det interessant for os?

- Vi kan modifcere startup-proces/hardware/miljø via uEnv.txt
- Vi kan modifcere boot config for device tree (pins standardfkt.)
- Vi kan lave device tree overlays, hvis vi permanent tilføjer ny HW
- Hvis noget går galt, kan vi måske reparere det (læsbar debugsymb.)
- Vigtigste filer er i partition uden for userspace rækkevidde (fx u-boot.img)

\*) Device tree: IEEE-standard til definition af hardware. Samme kerne på flere systemer – FDT er specifik til hardware; undgår hardcoding i source.

```
janus@beaglebone:/boot$ ls -la
totalt 17840
drwxr-xr-x 4 root root 4096 aug 4 02:12 .
drwxr-xr-x 21 root root 4096 aug 4 02:10 ..
-rw-r--r-- 1 root root 161266 jul 31 00:16 config-4.14.108-ti-r113
drwxr-xr-x 3 root root 4096 aug 4 02:03 dtbs
-rw-r--r-- 1 root root 4696442 aug 4 02:12 initrd.img-4.14.108-ti-r113
-rw-r--r-- 1 root root 542 aug 4 03:27 SOC.sh
-rw-r--r-- 1 root root 3447799 jul 31 00:16 System.map-4.14.108-ti-r113
drwxr-xr-x 2 root root 4096 aug 4 02:10 uboot
-rw-r--r-- 1 debian debian 2299 aug 4 03:27 uEnv.txt
-rw-r--r-- 1 root root 9929216 jul 31 00:16 vmlinuz-4.14.108-ti-r113
```

```
janus@beaglebone:/boot/dtbs/4.14.108-ti-r113$ ls -la | grep pocket
-rw-r--r-- 1 root root 124423 jul 31 00:16 am335x-pocketbeagle.dtb
-rw-r--r-- 1 root root 119573 jul 31 00:16 am335x-pocketbeagle-gameup.dtb
-rw-r--r-- 1 root root 126463 jul 31 00:16 am335x-pocketbeagle-techlab.dtb
```

## 4 stadier i opstarten

- 1: Primary Program Loader (AM335x ROM): Gør det muligt at læse fra SD-kortet
- 2: U-Boot Secondary Program Loader: Initialiserer pin-muxing, clocks, hukommelse. Loader U-Boot.
- 3: U-Boot (regular): Filsystem, uEnv.txt, loader og giver kontrol til Linux-kernen
- 4: Linux-kernen i hukommelse, perifere enheder konfigureres (USB, osv.). Mounter endeligt filesystem.

# SW4: BEAGLEBONE STARTUP, I/O, M.V. SHUT-DOWN

## Hovedbudskaber

Lad være med bare at tage strømmen...

Brug:

- Power-knappen til "soft shutdown"
- sudo reboot eller sudo shutdown -h now

Hvorfor?

- Fordi Derek Molloy siger det...
- Kørende processer skal have en chance for afslutte, gemme
- Filsystem skal unmountes
- Power management skal opdateres

Til højre: Soft shutdown

```
beaglebone login: [0;32m OK [0m Closed Load/Save RF Kill Switch Status /dev/rfkill Watch.
Stopping Session 1 of user ISD.
[0;32m OK [0m Stopped target System Time Synchronized.
[0;32m OK [0m Stopped target Timers.
[0;32m OK [0m Stopped Daily Cleanup of Temporary Directories.
[0;32m OK [0m Stopped target Host and Network Name Lookups.
Stopping dnsmasq - A lightweight DHCP and caching DNS server...
Stopping User Manager for UID 1001...
Unmounting /run/user/1001...
[0;32m OK [0m Unmounted /run/user/1001.
[0;32m OK [0m Reached target Unmount All Filesystems.
[0;32m OK [0m Stopped target Local File Systems (Pre).
[0;32m OK [0m Stopped Remount Root and Kernel File Systems.
[0;32m OK [0m Stopped Create Static Device Nodes in /dev.
[0;32m OK [0m Reached target Shutdown.
[0;32m OK [0m Reached target Final Step.
Starting Power-Off...
[ 82.361044] reboot: Power down

[0;32m OK [0m Reached target Shutdown.
[ 8812.284326] watchdog: watchdog0: watchdog did not stop!
[ 8812.537959] reboot: Restarting system
```

# SW4: BEAGLEBONE STARTUP, I/O, M.V. SHELL-SCRIPTS

## Shell scripts

Script som afvikles af/i shell

- bash, Zsh, etc.
  - En unix posix-ting (virker kun måske i Powershell/Win)
  - Der kan være forskellig syntax i forskellige shells
- Automatisering af linux-kommandoer
- Ikke bedste valg til mere avancerede scripts (brug Python eller Lua i stedet)
  - *Kun* linux-kommandoer, ingen libraries
  - Ikke OOP, ikke test-/udviklingsværktøjer
- Spawner en ny shell (proces) ved afvikling
- Kræver execute-rettigheder (chmod ugo+x), og evt. sudo
- Kan placeres, hvor det bliver auto-executed ved opstart
  - /etc/init.d/

## Eksempel 1

```
bashLED.sh
1 #!/bin/bash
2 LED3_PATH=/sys/class/leds/beaglebone:green:usr3
3 function removeTrigger
4 {
5   echo "none" >> "$LED3_PATH/trigger"
6 }
7
8 echo "Shell-script til håndtering af diode 3"
9
10 if [ $# != 1 ]; then
11   echo "Du må kun give et argument: "
12   echo -e "bashLED <command> \n"
13   echo -e "hvor <command>=on, off, flash, status"
14   exit 2
15 fi
16 echo "Du har givet kommandoen: $1"
17
18 if [ "$1" == "on" ]; then
19   echo "Tænder diode 3"
20   removeTrigger
21   echo "1" >> "$LED3_PATH/brightness"
22 elif [ "$1" == "off" ]; then
23   echo "Slukker for diode 3"
24   removeTrigger
25   echo "0" >> "$LED3_PATH/brightness"
26 elif [ "$1" == "flash" ]; then
27   echo "Blinker med diode 3"
28   echo "timer" >> "$LED3_PATH/trigger"
29   echo "50" >> "$LED3_PATH/delay_on"
30   echo "50" >> "$LED3_PATH/delay_off"
31 elif [ "$1" == "status" ]; then
32   cat "$LED3_PATH/trigger";
33 fi
34 echo "Scriptet er slut"
```

Afvikles af ...

Variabel

Funktion

Control flow  
if/then / elif/  
else/ fi

Input-args

Exit-val (fejl 2,  
incorrect  
usage)

[ ] er en test  
substitution...  
Bash's [[ ]] er  
bedre!

Afslutning af if-  
struktur

# SW4: BEAGLEBONE STARTUP, I/O, M.V. SHELL-SCRIPTS

## Eksempel fra Pro3 - testafvikling

```
s8-tc.sh
1 #!/usr/bin/env bash
2 # This script runs test cases S8.TC-1, TC-2, TC-3 and TC-4
3 # Only tested on MacOS so far.
4 # I don't think TC-4 is compatible on Windows, and I am not sure if it works on Linux
5 # Janus 21 Nov 2019
6
7 MQTT_PORT="1883"
8 WEBSOCK_PORT="8081"
9 URL="auteam2.mooo.com"
10 TMP_FILE="./test-cases/s8-tc-tmp.txt"
11 TEST_MSG="This is a test message sent from publisher to subscriber"
12
13 # Attempting to determine the system, and set the correct commands
14 # Prepare
15 if [[ "$uname" =~ "MING" ]]; then
16   USER_OS="Windows"
17   DOCKERCOMPOSEUP="docker-compose -f docker-compose-win.yml up -d"
18   LOCALHOST="192.168.99.100"
19   PORTSCANNER="nc -z"
20   WORDSEARCH="grep -w"
21
22 elif [[ "$uname" =~ "Darwin" ]]; then
23   USER_OS="macOS"
24   DOCKERCOMPOSEUP="docker-compose up -d --build"
25   LOCALHOST="localhost"
26   PORTSCANNER="nc -z"
27   WORDSEARCH="grep -w"
28
29 elif [[ "$uname" =~ "Linux" ]]; then
30   USER_OS="Linux"
31   DOCKERCOMPOSEUP="docker-compose up -d --build"
32   LOCALHOST="localhost"
33   PORTSCANNER="nc -z"
34   WORDSEARCH="grep -w"
35
36 else
37   USER_OS="unknown"
38   DOCKERCOMPOSEUP="docker-compose up -d --build"
39   LOCALHOST="localhost"
40
41 service_running () {
42   if [[ -z "$(docker-compose ps | ${WORDSEARCH} ".mqtts.*Up.*")" ]]; then
43     false # the service mqtt does not have Up in the status
44   else
45     true
46   fi
47 }
48
49 listening_on () {
50   if [[ -z "${!PORTSCANNER[@]} $1 $2 2>&1 | ${WORDSEARCH} "succeeded" " " ]]; then
51     false #the word succeeded was not found, so the service is not listening on the port
52   else
53     true
54   fi
55 }
56
57 get_credentials () {
58   if [[ -z "${!MQUITTO_USER[@]}" || [[ -z "${!MQUITTO_PASSWORD[@]}" ]]; then
59     echo "Enter credentials for the Mosquitto server:"
60     read -p "Enter User: " MOSQUITTO_USER
61     read -s -p "Enter Password: " MOSQUITTO_PASSWORD
62     echo ""
63     export MOSQUITTO_USER
64     export MOSQUITTO_PASSWORD
65   fi
66 }
67
68 #Font colours
69 RED="\033[1;31m"          # Bold red
70 GREEN="\033[0;32m"         # Bold green
71 YELLOW="\033[0;33m"        # Bold yellow
72 HEAD="\033[1;37;44m"
73 NC="\033[0m"              # No Color
74
75 ##### Info to users #####
76 echo -e "${HEAD}Running all testcases for deliverable S8${NC}"
77
78 ##### TEST-CASE TC-1 #####
79
80 TC="S8.TC-1"
81
```

```
s8-tc.sh
40 ##### TEST-CASE TC-1 #####
41 TC="S8.TC-1"
42 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
43 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
44
45 if service_running; then
46   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
47 else
48   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
49   $DOCKERCOMPOSEUP
50
51 if service_running; then
52   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
53 else
54   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
55   exit 1
56 fi
57
58 ##### TEST-CASE TC-2.01 #####
59 TC="S8.TC-2.01"
60 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
61 echo "Verifies that a message can be sent to the Mosquitto server on $LOCALHOST port $MQTT_PORT."
62
63 # Ask and set credentials if not inherited from parent process
64 get_credentials
65 if mosquitto_pub -h $LOCALHOST -p $MQTT_PORT -t "Test/topic" -m "Test message" --id TestPublisher -u "${MQUITTO_USER}" -P "${MQUITTO_PASSWORD}"; then
66   echo -e "${TC}: ${GREEN}OK${NC}. Service can accept publish messages."
67 else
68   echo -e "${TC}: ${RED}FAILED${NC}. Service declined the operation."
69 fi
70
71 ##### TEST-CASE TC-2.02 #####
72 TC="S8.TC-2.02"
73 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
74 echo "Verifies that the Mosquitto-server is listening on $URL:$MQTT_PORT"
75
76 if listening_on "$URL" "$MQTT_PORT" && listening_on "$URL" "$MQUITTO_PORT"; then
77   echo -e "${TC}: ${GREEN}OK${NC}. Service is listening."
78 else
79   echo -e "${TC}: ${RED}FAILED${NC}. Service is not listening."
80 fi
81
82 ##### TEST-CASE TC-2.03 #####
83 TC="S8.TC-2.03"
84 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
85 echo "Verifies that the Mosquitto-server is listening on $URL:$MQUITTO_PORT"
86
87 if listening_on "$URL" "$MQUITTO_PORT" && listening_on "$URL" "$MQTT_PORT"; then
88   echo -e "${TC}: ${GREEN}OK${NC}. Service is listening."
89 else
90   echo -e "${TC}: ${RED}FAILED${NC}. Service is not listening."
91 fi
92
93 ##### TEST-CASE TC-4 #####
94 TC="S8.TC-4"
95 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
96 echo "Verifies that a topic can be subscribed to and that a message can received from the Mosquitto server on $LOCALHOST port $MQUITTO_PORT."
97
98 mosquito_sub -h $LOCALHOST -p $MQUITTO_PORT -t "Test/topic" --id TestSubscriber -u "${MQUITTO_USER}" -P "${MQUITTO_PASSWORD}" > "$TMP_FILE" & PROC_ID=$!
99
100 disown
101 sleep 1 # Wait to ensure it's running
102
103 # Send a publish message
104 mosquito_pub -h $LOCALHOST -p $MQTT_PORT -t "Test/topic" -m "${TEST_MSG}" --id TestPublisher -u "${MQUITTO_USER}" -P "${MQUITTO_PASSWORD}"
105
106 # Wait for message handling, kill the subscribe process
107 sleep 1
108 kill -1 $PROC_ID 2>/dev/null
109
110 # Read back the output from the received message and compare to what was sent
111 RCV_MSG=$(tr -d '\0' < $TMP_FILE)
112 if [ "$RCV_MSG" = "${TEST_MSG}" ]; then
113   echo "Received the message: ${RCV_MSG}"
114 else
115   echo -e "${TC}: ${GREEN}OK${NC}. Messages on subscribed topics can be received from the service."
116 fi
117
118 ##### TEST-CASE TC-5 #####
119 TC="S8.TC-5"
120 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
121 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
122
123 if service_running; then
124   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
125 else
126   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
127   $DOCKERCOMPOSEUP
128
129 if service_running; then
130   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
131 else
132   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
133   exit 1
134 fi
135
136 ##### TEST-CASE TC-6 #####
137 TC="S8.TC-6"
138 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
139 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
140
141 if service_running; then
142   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
143 else
144   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
145   $DOCKERCOMPOSEUP
146
147 if service_running; then
148   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
149 else
150   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
151   exit 1
152 fi
153
154 ##### TEST-CASE TC-7 #####
155 TC="S8.TC-7"
156 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
157 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
158
159 if service_running; then
160   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
161 else
162   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
163   $DOCKERCOMPOSEUP
164
165 if service_running; then
166   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
167 else
168   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
169   exit 1
170 fi
171
172 ##### TEST-CASE TC-8 #####
173 TC="S8.TC-8"
174 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
175 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
176
177 if service_running; then
178   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
179 else
180   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
181   $DOCKERCOMPOSEUP
182
183 if service_running; then
184   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
185 else
186   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
187   exit 1
188 fi
189
190 ##### TEST-CASE TC-9 #####
191 TC="S8.TC-9"
192 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
193 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
194
195 if service_running; then
196   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
197 else
198   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
199   $DOCKERCOMPOSEUP
200
201 if service_running; then
202   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
203 else
204   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
205   exit 1
206 fi
207
208 ##### TEST-CASE TC-10 #####
209 TC="S8.TC-10"
210 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
211 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
212
213 if service_running; then
214   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
215 else
216   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
217   $DOCKERCOMPOSEUP
218
219 if service_running; then
220   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
221 else
222   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
223   exit 1
224 fi
225
226 ##### TEST-CASE TC-11 #####
227 TC="S8.TC-11"
228 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
229 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
230
231 if service_running; then
232   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
233 else
234   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
235   $DOCKERCOMPOSEUP
236
237 if service_running; then
238   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
239 else
240   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
241   exit 1
242 fi
243
244 ##### TEST-CASE TC-12 #####
245 TC="S8.TC-12"
246 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
247 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
248
249 if service_running; then
250   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
251 else
252   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
253   $DOCKERCOMPOSEUP
254
255 if service_running; then
256   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
257 else
258   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
259   exit 1
260 fi
261
262 ##### TEST-CASE TC-13 #####
263 TC="S8.TC-13"
264 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
265 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
266
267 if service_running; then
268   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
269 else
270   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
271   $DOCKERCOMPOSEUP
272
273 if service_running; then
274   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
275 else
276   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
277   exit 1
278 fi
279
280 ##### TEST-CASE TC-14 #####
281 TC="S8.TC-14"
282 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
283 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
284
285 if service_running; then
286   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
287 else
288   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
289   $DOCKERCOMPOSEUP
290
291 if service_running; then
292   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
293 else
294   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
295   exit 1
296 fi
297
298 ##### TEST-CASE TC-15 #####
299 TC="S8.TC-15"
300 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
301 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
302
303 if service_running; then
304   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
305 else
306   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
307   $DOCKERCOMPOSEUP
308
309 if service_running; then
310   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
311 else
312   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
313   exit 1
314 fi
315
316 ##### TEST-CASE TC-16 #####
317 TC="S8.TC-16"
318 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
319 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
320
321 if service_running; then
322   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
323 else
324   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
325   $DOCKERCOMPOSEUP
326
327 if service_running; then
328   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
329 else
330   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
331   exit 1
332 fi
333
334 ##### TEST-CASE TC-17 #####
335 TC="S8.TC-17"
336 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
337 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
338
339 if service_running; then
340   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
341 else
342   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
343   $DOCKERCOMPOSEUP
344
345 if service_running; then
346   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
347 else
348   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
349   exit 1
350 fi
351
352 ##### TEST-CASE TC-18 #####
353 TC="S8.TC-18"
354 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
355 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
356
357 if service_running; then
358   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
359 else
360   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
361   $DOCKERCOMPOSEUP
362
363 if service_running; then
364   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
365 else
366   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
367   exit 1
368 fi
369
370 ##### TEST-CASE TC-19 #####
371 TC="S8.TC-19"
372 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
373 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
374
375 if service_running; then
376   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
377 else
378   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
379   $DOCKERCOMPOSEUP
380
381 if service_running; then
382   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
383 else
384   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
385   exit 1
386 fi
387
388 ##### TEST-CASE TC-20 #####
389 TC="S8.TC-20"
390 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
391 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
392
393 if service_running; then
394   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
395 else
396   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
397   $DOCKERCOMPOSEUP
398
399 if service_running; then
400   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
401 else
402   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
403   exit 1
404 fi
405
406 ##### TEST-CASE TC-21 #####
407 TC="S8.TC-21"
408 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
409 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
410
411 if service_running; then
412   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
413 else
414   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
415   $DOCKERCOMPOSEUP
416
417 if service_running; then
418   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
419 else
420   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
421   exit 1
422 fi
423
424 ##### TEST-CASE TC-22 #####
425 TC="S8.TC-22"
426 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
427 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
428
429 if service_running; then
430   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
431 else
432   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
433   $DOCKERCOMPOSEUP
434
435 if service_running; then
436   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
437 else
438   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
439   exit 1
440 fi
441
442 ##### TEST-CASE TC-23 #####
443 TC="S8.TC-23"
444 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
445 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
446
447 if service_running; then
448   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
449 else
450   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
451   $DOCKERCOMPOSEUP
452
453 if service_running; then
454   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
455 else
456   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
457   exit 1
458 fi
459
460 ##### TEST-CASE TC-24 #####
461 TC="S8.TC-24"
462 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
463 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
464
465 if service_running; then
466   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
467 else
468   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
469   $DOCKERCOMPOSEUP
470
471 if service_running; then
472   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
473 else
474   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
475   exit 1
476 fi
477
478 ##### TEST-CASE TC-25 #####
479 TC="S8.TC-25"
480 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
481 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
482
483 if service_running; then
484   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
485 else
486   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
487   $DOCKERCOMPOSEUP
488
489 if service_running; then
490   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
491 else
492   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
493   exit 1
494 fi
495
496 ##### TEST-CASE TC-26 #####
497 TC="S8.TC-26"
498 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
499 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
500
501 if service_running; then
502   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
503 else
504   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
505   $DOCKERCOMPOSEUP
506
507 if service_running; then
508   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
509 else
510   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
511   exit 1
512 fi
513
514 ##### TEST-CASE TC-27 #####
515 TC="S8.TC-27"
516 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
517 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
518
519 if service_running; then
520   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
521 else
522   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
523   $DOCKERCOMPOSEUP
524
525 if service_running; then
526   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
527 else
528   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
529   exit 1
530 fi
531
532 ##### TEST-CASE TC-28 #####
533 TC="S8.TC-28"
534 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
535 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
536
537 if service_running; then
538   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
539 else
540   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
541   $DOCKERCOMPOSEUP
542
543 if service_running; then
544   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
545 else
546   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
547   exit 1
548 fi
549
550 ##### TEST-CASE TC-29 #####
551 TC="S8.TC-29"
552 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
553 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
554
555 if service_running; then
556   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
557 else
558   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
559   $DOCKERCOMPOSEUP
560
561 if service_running; then
562   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
563 else
564   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
565   exit 1
566 fi
567
568 ##### TEST-CASE TC-30 #####
569 TC="S8.TC-30"
570 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
571 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
572
573 if service_running; then
574   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
575 else
576   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
577   $DOCKERCOMPOSEUP
578
579 if service_running; then
580   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
581 else
582   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
583   exit 1
584 fi
585
586 ##### TEST-CASE TC-31 #####
587 TC="S8.TC-31"
588 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
589 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
590
591 if service_running; then
592   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
593 else
594   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
595   $DOCKERCOMPOSEUP
596
597 if service_running; then
598   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
599 else
600   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
601   exit 1
602 fi
603
604 ##### TEST-CASE TC-32 #####
605 TC="S8.TC-32"
606 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
607 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
608
609 if service_running; then
610   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
611 else
612   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
613   $DOCKERCOMPOSEUP
614
615 if service_running; then
616   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
617 else
618   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
619   exit 1
620 fi
621
622 ##### TEST-CASE TC-33 #####
623 TC="S8.TC-33"
624 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
625 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
626
627 if service_running; then
628   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
629 else
630   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
631   $DOCKERCOMPOSEUP
632
633 if service_running; then
634   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
635 else
636   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
637   exit 1
638 fi
639
640 ##### TEST-CASE TC-34 #####
641 TC="S8.TC-34"
642 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
643 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
644
645 if service_running; then
646   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
647 else
648   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
649   $DOCKERCOMPOSEUP
650
651 if service_running; then
652   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
653 else
654   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
655   exit 1
656 fi
657
658 ##### TEST-CASE TC-35 #####
659 TC="S8.TC-35"
660 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
661 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
662
663 if service_running; then
664   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
665 else
666   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
667   $DOCKERCOMPOSEUP
668
669 if service_running; then
670   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
671 else
672   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
673   exit 1
674 fi
675
676 ##### TEST-CASE TC-36 #####
677 TC="S8.TC-36"
678 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
679 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
680
681 if service_running; then
682   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
683 else
684   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
685   $DOCKERCOMPOSEUP
686
687 if service_running; then
688   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
689 else
690   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
691   exit 1
692 fi
693
694 ##### TEST-CASE TC-37 #####
695 TC="S8.TC-37"
696 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
697 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
698
699 if service_running; then
700   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
701 else
702   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
703   $DOCKERCOMPOSEUP
704
705 if service_running; then
706   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
707 else
708   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
709   exit 1
710 fi
711
712 ##### TEST-CASE TC-38 #####
713 TC="S8.TC-38"
714 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
715 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
716
717 if service_running; then
718   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
719 else
720   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
721   $DOCKERCOMPOSEUP
722
723 if service_running; then
724   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
725 else
726   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
727   exit 1
728 fi
729
730 ##### TEST-CASE TC-39 #####
731 TC="S8.TC-39"
732 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
733 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
734
735 if service_running; then
736   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
737 else
738   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
739   $DOCKERCOMPOSEUP
740
741 if service_running; then
742   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
743 else
744   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
745   exit 1
746 fi
747
748 ##### TEST-CASE TC-40 #####
749 TC="S8.TC-40"
750 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
751 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
752
753 if service_running; then
754   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
755 else
756   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
757   $DOCKERCOMPOSEUP
758
759 if service_running; then
760   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
761 else
762   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
763   exit 1
764 fi
765
766 ##### TEST-CASE TC-41 #####
767 TC="S8.TC-41"
768 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
769 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
770
771 if service_running; then
772   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
773 else
774   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
775   $DOCKERCOMPOSEUP
776
777 if service_running; then
778   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
779 else
780   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
781   exit 1
782 fi
783
784 ##### TEST-CASE TC-42 #####
785 TC="S8.TC-42"
786 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
787 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
788
789 if service_running; then
790   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
791 else
792   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
793   $DOCKERCOMPOSEUP
794
795 if service_running; then
796   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
797 else
798   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
799   exit 1
800 fi
801
802 ##### TEST-CASE TC-43 #####
803 TC="S8.TC-43"
804 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
805 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
806
807 if service_running; then
808   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
809 else
810   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
811   $DOCKERCOMPOSEUP
812
813 if service_running; then
814   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
815 else
816   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
817   exit 1
818 fi
819
820 ##### TEST-CASE TC-44 #####
821 TC="S8.TC-44"
822 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
823 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
824
825 if service_running; then
826   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
827 else
828   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
829   $DOCKERCOMPOSEUP
830
831 if service_running; then
832   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
833 else
834   echo -e "${TC}: ${RED}FAILED${NC}. Service could not be started."
835   exit 1
836 fi
837
838 ##### TEST-CASE TC-45 #####
839 TC="S8.TC-45"
840 echo -e "${HEAD}*** Test-case ${TC} ***${NC}"
841 echo "Verifies that the Mosquitto server is successfully running in docker-compose"
842
843 if service_running; then
844   echo -e "${TC}: ${GREEN}OK${NC}. Service is running."
845 else
846   echo -e "${TC}: ${YELLOW}Service not started${NC}. Attempting to build and start.."
847   $DOCKERCOMPOSEUP
848
849 if service_running; then
850   echo -e "${TC}:
```

# SW4: BEAGLEBONE STARTUP, I/O, M.V.

## I/O: OVERBLIK

### I/O-systemet

Pins har 8 modes (mux bits 0-2)

- Brug ikke de "røde" pins i headertabellerne

Single-wire (GPIO, mode 7)

- Digital I/O
- Elekt. begrænsninger! 3.3V, src:4-6mA, snk:8mA

Seriel bus (SPI, I2C, CAN) eller UART

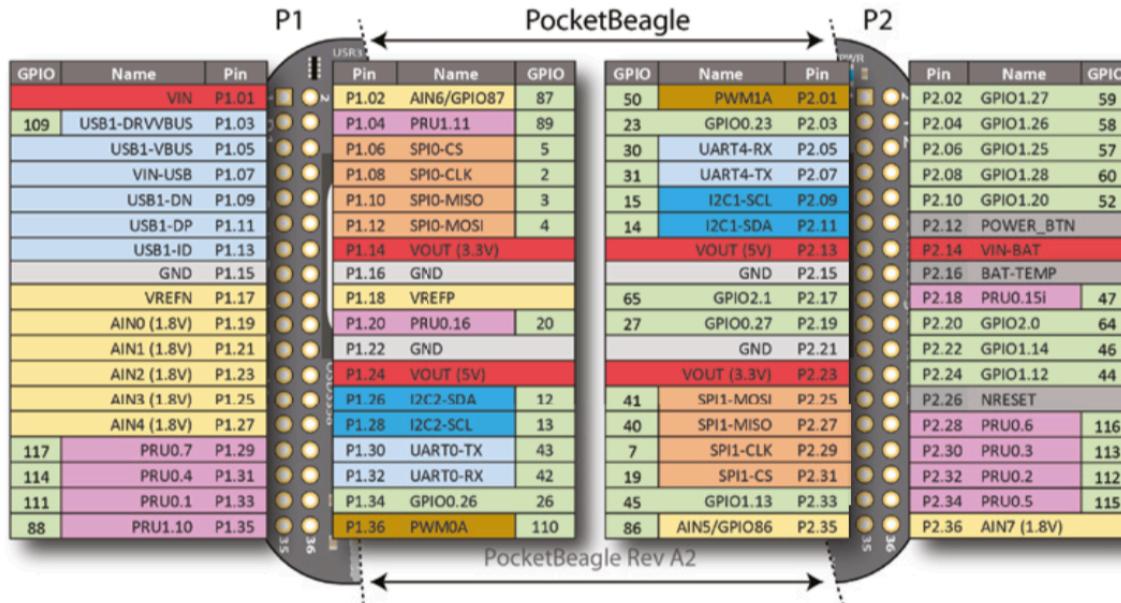
- Kommunikation til komplekse sensorer eller moduler
- Seriell kommunikation

Analog interfacing

- Ind: ADC, ud: PWM

USB-devices

- Hvis der er Linux-drivers



Pin	PB Pin	SPIN	Offset from	DIO#	Mode?	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode8	Mode9	Mode10	Mode11	Mode12	Mode13	Mode14	Mode15	Mode16	Mode17	Mode18	Mode19	Mode20	Mode21	Mode22	Mode23	Mode24	Mode25	Mode26	Mode27	Mode28	Mode29	Mode30	Mode31	Mode32	Mode33	Mode34	Mode35	Mode36	Mode37	Mode38	Mode39	Mode40	Mode41	Mode42	Mode43	Mode44	Mode45	Mode46	Mode47	Mode48	Mode49	Mode50	Mode51	Mode52	Mode53	Mode54	Mode55	Mode56	Mode57	Mode58	Mode59	Mode60	Mode61	Mode62	Mode63	Mode64	Mode65	Mode66	Mode67	Mode68	Mode69	Mode70	Mode71	Mode72	Mode73	Mode74	Mode75	Mode76	Mode77	Mode78	Mode79	Mode80	Mode81	Mode82	Mode83	Mode84	Mode85	Mode86	Mode87	Mode88	Mode89	Mode90	Mode91	Mode92	Mode93	Mode94	Mode95	Mode96	Mode97	Mode98	Mode99	Mode100	Mode101	Mode102	Mode103	Mode104	Mode105	Mode106	Mode107	Mode108	Mode109	Mode110	Mode111	Mode112	Mode113	Mode114	Mode115	Mode116	Mode117	Mode118	Mode119	Mode120	Mode121	Mode122	Mode123	Mode124	Mode125	Mode126	Mode127	Mode128	Mode129	Mode130	Mode131	Mode132	Mode133	Mode134	Mode135	Mode136	Mode137	Mode138	Mode139	Mode140	Mode141	Mode142	Mode143	Mode144	Mode145	Mode146	Mode147	Mode148	Mode149	Mode150	Mode151	Mode152	Mode153	Mode154	Mode155	Mode156	Mode157	Mode158	Mode159	Mode160	Mode161	Mode162	Mode163	Mode164	Mode165	Mode166	Mode167	Mode168	Mode169	Mode170	Mode171	Mode172	Mode173	Mode174	Mode175	Mode176	Mode177	Mode178	Mode179	Mode180	Mode181	Mode182	Mode183	Mode184	Mode185	Mode186	Mode187	Mode188	Mode189	Mode190	Mode191	Mode192	Mode193	Mode194	Mode195	Mode196	Mode197	Mode198	Mode199	Mode200	Mode201	Mode202	Mode203	Mode204	Mode205	Mode206	Mode207	Mode208	Mode209	Mode210	Mode211	Mode212	Mode213	Mode214	Mode215	Mode216	Mode217	Mode218	Mode219	Mode220	Mode221	Mode222	Mode223	Mode224	Mode225	Mode226	Mode227	Mode228	Mode229	Mode230	Mode231	Mode232	Mode233	Mode234	Mode235	Mode236	Mode237	Mode238	Mode239	Mode240	Mode241	Mode242	Mode243	Mode244	Mode245	Mode246	Mode247	Mode248	Mode249	Mode250	Mode251	Mode252	Mode253	Mode254	Mode255	Mode256	Mode257	Mode258	Mode259	Mode260	Mode261	Mode262	Mode263	Mode264	Mode265	Mode266	Mode267	Mode268	Mode269	Mode270	Mode271	Mode272	Mode273	Mode274	Mode275	Mode276	Mode277	Mode278	Mode279	Mode280	Mode281	Mode282	Mode283	Mode284	Mode285	Mode286	Mode287	Mode288	Mode289	Mode290	Mode291	Mode292	Mode293	Mode294	Mode295	Mode296	Mode297	Mode298	Mode299	Mode300	Mode301	Mode302	Mode303	Mode304	Mode305	Mode306	Mode307	Mode308	Mode309	Mode310	Mode311	Mode312	Mode313	Mode314	Mode315	Mode316	Mode317	Mode318	Mode319	Mode320	Mode321	Mode322	Mode323	Mode324	Mode325	Mode326	Mode327	Mode328	Mode329	Mode330	Mode331	Mode332	Mode333	Mode334	Mode335	Mode336	Mode337	Mode338	Mode339	Mode340	Mode341	Mode342	Mode343	Mode344	Mode345	Mode346	Mode347	Mode348	Mode349	Mode350	Mode351	Mode352	Mode353	Mode354	Mode355	Mode356	Mode357	Mode358	Mode359	Mode360	Mode361	Mode362	Mode363	Mode364	Mode365	Mode366	Mode367	Mode368	Mode369	Mode370	Mode371	Mode372	Mode373	Mode374	Mode375	Mode376	Mode377	Mode378	Mode379	Mode380	Mode381	Mode382	Mode383	Mode384	Mode385	Mode386	Mode387	Mode388	Mode389	Mode390	Mode391	Mode392	Mode393	Mode394	Mode395	Mode396	Mode397	Mode398	Mode399	Mode400	Mode401	Mode402	Mode403	Mode404	Mode405	Mode406	Mode407	Mode408	Mode409	Mode410	Mode411	Mode412	Mode413	Mode414	Mode415	Mode416	Mode417	Mode418	Mode419	Mode420	Mode421	Mode422	Mode423	Mode424	Mode425	Mode426	Mode427	Mode428	Mode429	Mode430	Mode431	Mode432	Mode433	Mode434	Mode435	Mode436	Mode437	Mode438	Mode439	Mode440	Mode441	Mode442	Mode443	Mode444	Mode445	Mode446	Mode447	Mode448	Mode449	Mode450	Mode451	Mode452	Mode453	Mode454	Mode455	Mode456	Mode457	Mode458	Mode459	Mode460	Mode461	Mode462	Mode463	Mode464	Mode465	Mode466	Mode467	Mode468	Mode469	Mode470	Mode471	Mode472	Mode473	Mode474	Mode475	Mode476	Mode477	Mode478	Mode479	Mode480	Mode481	Mode482	Mode483	Mode484	Mode485	Mode486	Mode487	Mode488	Mode489	Mode490	Mode491	Mode492	Mode493	Mode494	Mode495	Mode496	Mode497	Mode498	Mode499	Mode500	Mode501	Mode502	Mode503	Mode504	Mode505	Mode506	Mode507	Mode508	Mode509	Mode510	Mode511	Mode512	Mode513	Mode514	Mode515	Mode516	Mode517	Mode518	Mode519	Mode520	Mode521	Mode522	Mode523	Mode524	Mode525	Mode526	Mode527	Mode528	Mode529	Mode530	Mode531	Mode532	Mode533	Mode534	Mode535	Mode536	Mode537	Mode538	Mode539	Mode540	Mode541	Mode542	Mode543	Mode544	Mode545	Mode546	Mode547	Mode548	Mode549	Mode550	Mode551	Mode552	Mode553	Mode554	Mode555	Mode556	Mode557	Mode558	Mode559	Mode560	Mode561	Mode562	Mode563	Mode564	Mode565	Mode566	Mode567	Mode568	Mode569	Mode570	Mode571	Mode572	Mode573	Mode574	Mode575	Mode576	Mode577	Mode578	Mode579	Mode580	Mode581	Mode582	Mode583	Mode584	Mode585	Mode586	Mode587	Mode588	Mode589	Mode590	Mode591	Mode592	Mode593	Mode594	Mode595	Mode596	Mode597	Mode598	Mode599	Mode600	Mode601	Mode602	Mode603	Mode604	Mode605	Mode606	Mode607	Mode608	Mode609	Mode610	Mode611	Mode612	Mode613	Mode614	Mode615	Mode616	Mode617	Mode618	Mode619	Mode620	Mode621	Mode622	Mode623	Mode624	Mode625	Mode626	Mode627	Mode628	Mode629	Mode630	Mode631	Mode632	Mode633	Mode634	Mode635	Mode636	Mode637	Mode638	Mode639	Mode640	Mode641	Mode642	Mode643	Mode644	Mode645	Mode646	Mode647	Mode648	Mode649	Mode650	Mode651	Mode652	Mode653	Mode654	Mode655	Mode656	Mode657	Mode658	Mode659	Mode660	Mode661	Mode662	Mode663	Mode664	Mode665	Mode666	Mode667	Mode668	Mode669	Mode670	Mode671	Mode672	Mode673	Mode674	Mode675	Mode676	Mode677	Mode678	Mode679	Mode680	Mode681	Mode682	Mode683	Mode684	Mode685	Mode686	Mode687	Mode688	Mode689	Mode690	Mode691	Mode692	Mode693	Mode694	Mode695	Mode696	Mode697	Mode698	Mode699	Mode700	Mode701	Mode702	Mode703	Mode704	Mode705	Mode706	Mode707	Mode708	Mode709	Mode710	Mode711	Mode712	Mode713	Mode714	Mode715	Mode716	Mode717	Mode718	Mode719	Mode720	Mode721	Mode722	Mode723	Mode724	Mode725	Mode726	Mode727	Mode728	Mode729	Mode730	Mode731	Mode732	Mode733	Mode734	Mode735	Mode736	Mode737	Mode738	Mode739	Mode740	Mode741	Mode742	Mode743	Mode744	Mode745	Mode746	Mode747	Mode748	Mode749	Mode750	Mode751	Mode752	Mode753	Mode754	Mode755	Mode756	Mode757	Mode758	Mode759	Mode760	Mode761	Mode762	Mode763	Mode764	Mode765	Mode766	Mode767	Mode768	Mode769	Mode770	Mode771	Mode772	Mode773	Mode774	Mode775	Mode776	Mode777	Mode778	Mode779	Mode780	Mode781	Mode782	Mode783	Mode784	Mode785	Mode786	Mode787	Mode788	Mode789	Mode790	Mode791	Mode792	Mode793	Mode794	Mode795	Mode796	Mode797	Mode798	Mode799	Mode800	Mode801	Mode802	Mode803	Mode804	Mode805	Mode806	Mode807	Mode808	Mode809	Mode810	Mode811	Mode812	Mode813	Mode814	Mode815	Mode816	Mode817	Mode818	Mode819	Mode820	Mode821	Mode822	Mode823	Mode824	Mode825	Mode826	Mode827	Mode828	Mode829	Mode830	Mode831	Mode832	Mode833	Mode834	Mode835	Mode836	Mode837	Mode838	Mode839	Mode840	Mode841	Mode842	Mode843	Mode844	Mode845	Mode846	Mode847	Mode848	Mode849	Mode850	Mode851	Mode852	Mode853	Mode854	Mode855	Mode856	Mode857	Mode858	Mode859	Mode860	Mode861	Mode862	Mode863	Mode864	Mode865	Mode866	Mode867	Mode868	Mode869	Mode870	Mode871	Mode872	Mode873	Mode874	Mode875	Mode876	Mode877	Mode878	Mode879	Mode880	Mode881	Mode882	Mode883	Mode884	Mode885	Mode886	Mode887	Mode888	Mode889	Mode890	Mode891	Mode892	Mode893	Mode894	Mode895	Mode896	Mode897	Mode898	Mode899	Mode900	Mode901	Mode902	Mode903	Mode904	Mode905	Mode906	Mode907	Mode908	Mode909	Mode910	Mode911	Mode912	Mode913	Mode914	Mode915	Mode916	Mode917	Mode918	Mode919	Mode920	Mode921	Mode922	Mode923	Mode924	Mode925	Mode926	Mode927	Mode928	Mode929</th

# SW4: BEAGLEBONE STARTUP, I/O, M.V.

## I/O: ALTING ER EN FIL

### I/O linux tools -> alt er en fil

config-pin tool:

- mode, direction (+/- pullup/down) value

```
janus@beaglebone:~/projects/E3ISD1/ch6$ sudo usermod -a -G gpio janus
```

```
janus@beaglebone:~/projects/E3ISD1/ch6$ config-pin -l p2.08
```

```
default gpio gpio_pu gpio_pd gpio_input
```

```
janus@beaglebone:~/projects/E3ISD1/ch6$ config-pin -q p2.08
```

```
P2_08 Mode: gpio Direction: in Value: 0
```

Mapping ift. AM335x fire GPIO-banker á 32 pins:

- PB 2.08 -> GPIO1\_28 => 32+28 = GPIO60

-> Sysfs:

```
janus@beaglebone:~/projects/E3ISD1/ch6$ cat /sys/class/gpio/gpio60/direction
```

-> Alt er en fil

```
janus@beaglebone:/sys/class/gpio/gpio60$ config-pin -a p2.08 out
```

```
janus@beaglebone:/sys/class/gpio/gpio60$ echo 1 > value
```

```
janus@beaglebone:/sys/class/gpio/gpio60$ config-pin -q p2.08
```

```
P2_08 Mode: gpio Direction: out Value: 1
```

Men: Langsom metode, høj CPU laad (polling sysfs).

- Hurtig switching -> PWM!

C++ tools (gpio.h)



AARHUS  
UNIVERSITET  
INGENIØRHØJSKOLEN AARHUS UNIVERSITET

```
janus@beaglebone:~$ groups
janus dialout cdrom floppy sudo audio video plugdev users i2c spi pwm gpio
```

### PWM-eksempel

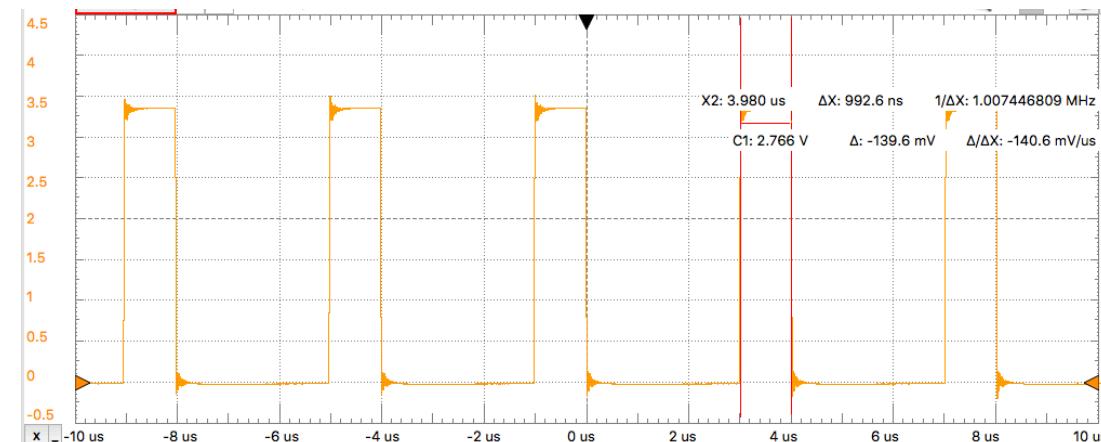
pwm.sh

```
#!/bin/bash
1 #!/bin/bash
1 PWM_PATH="/sys/class/pwm/pwm-0:0"
2 if [ "$1" == "on" ]; then
3     echo 4000 > ${PWM_PATH}/period
4     echo 1000 > ${PWM_PATH}/duty_cycle
5     echo 1 > ${PWM_PATH}/enable
6     echo "PWM Period is now $(cat ${PWM_PATH}/period) ns."
7     echo "PWM Duty-cycle is now $(cat ${PWM_PATH}/duty_cycle) ns out of total period."
8 elif [ "$1" == "off" ]; then
9     echo 0 > ${PWM_PATH}/enable;
10 fi
```

```
janus@beaglebone:~/projects/E3ISD1/pwm$ ./pwm.sh on
PWM Period is now 4000 ns.
PWM Duty-cycle is now 1000 ns out of total period.
```

BBB CHIP <sup>3</sup>	CHANNEL	BBB PINS	PB CHIP	PB PINS
pwmchip0	0A	P9_22/	pwmchip0	P1.08/
				P9_31
pwmchip0	0B	P9_21/	pwmchip0	P1.10/
				P9_29

```
janus@beaglebone:~/sys/class/pwm/pwm-0:0$ ls -la
total 0
drwxrwxr-x 3 root pwm 0 jan 1 2000 .
drwxrwxr-x 5 root pwm 0 jan 1 2000 ..
-r--r--r-- 1 root pwm 4096 jan 1 2000 capture
lrwxrwxrwx 1 root pwm 0 jan 23 00:07 device -> ../../pwmchip0
-rw-rw-r-- 1 root pwm 4096 jan 1 2000 duty_cycle
-rw-rw-r-- 1 root pwm 4096 jan 1 2000 enable
-rw-rw-r-- 1 root pwm 4096 jan 1 2000 period
-rw-rw-r-- 1 root pwm 4096 jan 1 2000 polarity
drwxrwxr-x 2 root pwm 0 jan 1 2000 power
lrwxrwxrwx 1 root pwm 0 jan 23 00:07 subsystem -> ../../../../../../class/pwm
-rw-rw-r-- 1 root pwm 4096 jan 1 2000 uevent
```



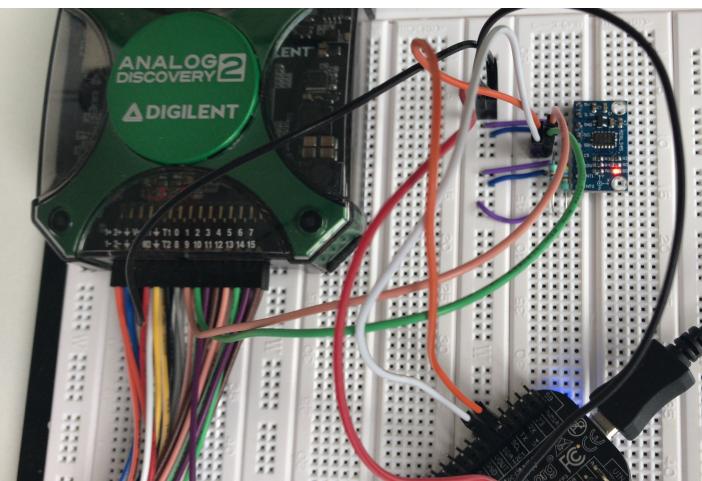
# SW4: BEAGLEBONE STARTUP, I/O, M.V.

## I/O: LINUX TOOLS, I2C - ADXL345

### ADXL345

Accelerometer, tri-axial MEMS sensor

- I2C, SPI; her forbundet via I2C (øv. 20a)
- SDA og SCL pulled-up (aktiv lav)
- Interrupts 1 og 2 ikke benyttet
- På I2C-bus på adresse 0x53, DEVID 0xe5
- Start-bit (SDA), så clock, så 7-bit adresse



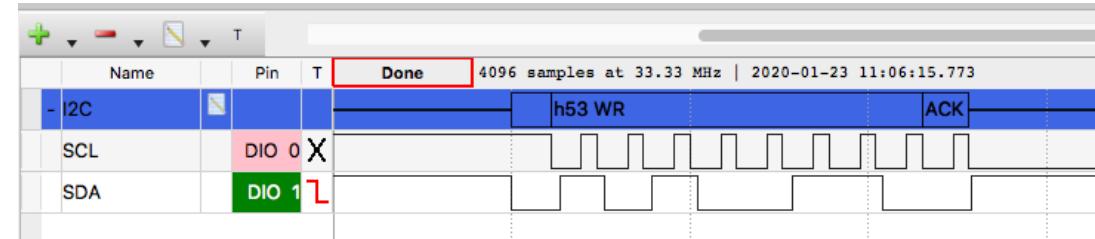
### Linux tools

```
janus@beaglebone:~$ i2cdetect -y -r 1
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: --- - - - - - - - - - - - - - - - -
10: --- - - - - - - - - - - - - - - - -
20: --- - - - - - - - - - - - - - - - -
30: --- - - - - - - - - - - - - - - - -
40: --- - - - - - - - - - - - - - - - -
50: --- 53 - - - - - - - - - - - - - - -
60: --- - - - - - - - - - - - - - - - -
70: --- - - - - - - - - - - - - - - - -
```

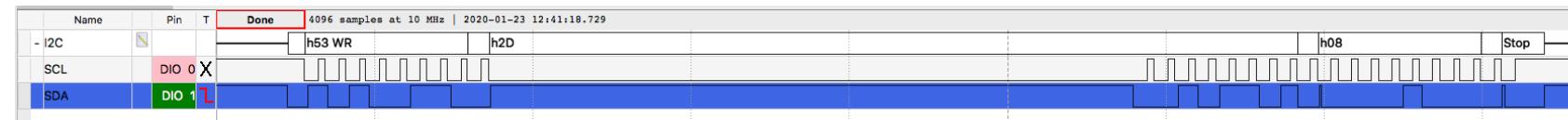
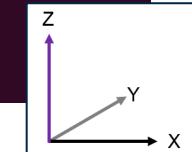
```
janus@beaglebone:~$ i2cget -y 1 0x53 0x00
0x05
```

```
janus@beaglebone:~/projects/E3ISD1/ADXL345$ i2cdump -y 1 0x53 b
 0 1 2 3 4 5 6 7 8 9 a b c d e f 0123456789abcdef
00: e5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 ?....?>...?...
10: 82 00 30 00 00 ff 01 3e 00 00 00 93 00 00 00 00 ?0...?>...?...
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....?....?...
30: 82 00 fc ff 06 00 f9 00 00 00 00 00 00 00 00 00 00 ??.??.?....?
40: e5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 ?....?>...?...
50: 80 00 30 00 00 fe 02 3e 00 00 00 93 00 00 00 00 ?0...?>...?...
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....?....?...
70: 82 00 fb ff 07 00 f8 00 00 00 00 00 00 00 00 00 00 ??.??.?....?
80: e5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 ?....?>...?...
90: 00 00 30 00 00 fe 01 3e 00 00 00 93 00 00 00 00 ..0...?>...?...
a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....?....?...
b0: 00 00 30 00 00 ff 02 3e 00 00 00 93 00 00 00 00 ..0...?>...?...
c0: e5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 ?....?>...?...
d0: 00 00 30 00 00 ff 02 3e 00 00 00 93 00 00 00 00 ..0...?>...?...
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....?....?...
f0: 02 00 fc ff 09 00 f8 00 00 00 00 00 00 00 00 00 00 ??.??.?....?
```

### Outputs



```
janus@beaglebone:~/projects/E3ISD1/ADXL345$ ./ADXL345
Starting the ADXL345 sensor application
The Device ID is: e5
The POWER_CTL mode is: 08
The DATA_FORMAT is: 00
X=-5 Y=7 Z=249 sample=24
```



# SW4: BEAGLEBONE STARTUP, I/O, M.V.

## I/O: SERIEL KOMMUNIKATION

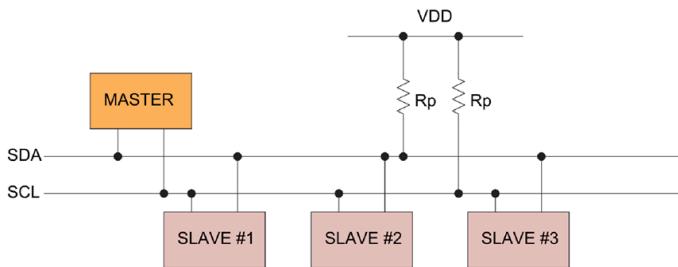
### I2C

Seriell-bus, hardware og protokol

- 2 linjer: SDA, SCL (aktiv lav)
- Synkron (clocked, SCL)
- Philips / NXP specifikation
- Hurtigere end UART, langsommere end SPI

Master-slaves

- 7-bit-adresser (128 enheder, men...)
- Clock-synkronisering fra master
  - Clock-stretching fra slave
- Multi-master via "collision detection" og "arbitration".



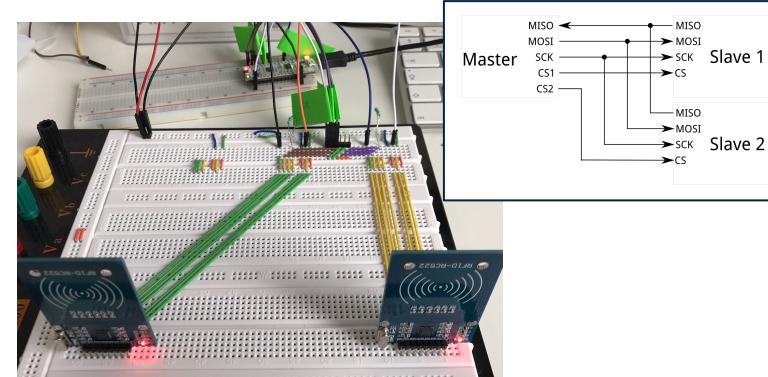
### SPI

Seriell-bus, hardware og protokol

- 2+N linjer: MOSI, MISO, SCK,  $N \times$ SS/CS
- Synkron (clocked, SCK)
- Master-slave(s), kun 1 master
- Hurtigste af I2C, UART og SPI

Eksempel:

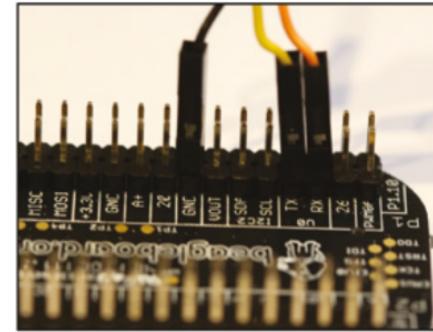
- Øvelse 20b
- Pro2, flere RFID-læsere i "samme hus"



### UART

Seriell-forbindelse, kun hardware

- Point-to-point (UART1 <-> UART2)
  - Rx, Tx linjer på hvert punkt
- Asynkron: Ingen clock
  - Hastighed (<115200 baud) skal være ens i hver ende
- Start/stop-bits, og maks. længde af databits (5-9)
- Hardware: Parallel databus -> skifteredister -> Tx -> seriel forb. -> Rx, osv...
- Kan implementere forskellige kommunikationsprotokoller, xx RS-232
- Langsommet af de tre
- Men simpel!



# SW4: BEAGLEBONE STARTUP, I/O, M.V. OPSUMMERING OG PERSPEKTIVERING

---

# SW5: BEAGLEBONE UDVIKLING

## INDHOLD

---

1. Overvejelser inden valg af sprog/teknologi
2. Scripting-eksempler
3. Cross-compiling
4. Remote debugging
5. Øvrige muligheder
6. Opsummering og perspektivering

5	Programudvikling og BBB	Forklar <ul style="list-style-type: none"><li>• hvordan man kan udvikle programmer der skal afvikles på BBB</li><li>• hvordan man kan fejlfinde et program, som kører på en BBB</li></ul>
---	-------------------------	--

# SW5: BEAGLEBONE UDVIKLING

## UDVIKLING - OVERORDNEDE VALG

### Scripting og fortolkede sprog

Stor udvalg

- Shell, Python, Lua, js/Node.js, Java

Metode:

- Hvis sproget ikke allerede er understøttet, så installér vha. debians package manager
- Installer krævede libs vha. sprogets egen package manager
- Åben en editor i terminalen (ViM, Emacs, nano, ...) og gå i gang...

Fordele:

- Kræver som regel blot at der build-tools samt at sprogets package manager kan hente pakker

Ulemper:

- Performance er sjældent lige så god som compilede sprog
- Debugging-værktøjer ikke så gode
- Koden kan læses og ændres af alle

### Compiledede sprog

Udvalg:

- C/C++, Rust, osv.

Generel fordel

- Mere "performant" kode
- Mulighed for build-system til at håndtere komplekse builds (Make, Cmake)

Metode 1: Compiling direkte på BB

- + Skriv kode og benyt toolchain/compiler i BB (GNU), samt evt. GDB og Valgrind
- - Ikke samme IDE-hjælp som med cross-compiling

Metode 2: Cross-compiling og cross-debugging

- Cross-compiler tool-chain (Linaro)
- + Grafisk IDE + Debugger (Eclipse CDT + GDB/GDBserver)
- + Debugger på den faktiske hardware + peripherals
- - Kan være vanskeligt med dynamisk linking til mange biblioteker
  - Multi-arkitektur tools og libraries...

# SW5: BEAGLEBONE UDVIKLING

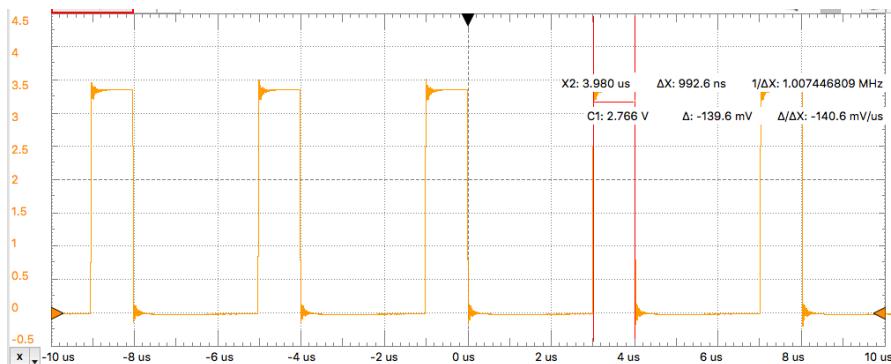
## SCRIPTING-EKSEMPLER

### Shell-script (fra øvelser)

```
bashLED.sh
1 #!/bin/bash
2 LED3_PATH=/sys/class/leds/beaglebone:green:usr3
3
4 function removeTrigger
5 {
6   echo "none" >> "$LED3_PATH/trigger"
7 }
8 echo "Shell-script til håndtering af diode 3"
9
10 if [ $# != 1 ]; then
11   echo "Du må kun give et argument: "
12   echo -e "bashLED <command> \n"
13   echo -e "hvor <command>=on, off, flash, status"
14   exit 2
15 fi
16 echo "Du har givet kommandoen: $1"
17
18 if [ "$1" == "on" ]; then
19   echo "Tænder diode 3"
20   removeTrigger
21   echo "1" >> "$LED3_PATH/brightness"
22 elif [ "$1" == "off" ]; then
23   echo "Slukker for diode 3"
24   removeTrigger
25   echo "0" >> "$LED3_PATH/brightness"
26 elif [ "$1" == "flash" ]; then
27   echo "Blinker med diode 3"
28   echo "timer" >> "$LED3_PATH/trigger"
29   echo "50" >> "$LED3_PATH/delay_on"
30   echo "50" >> "$LED3_PATH/delay_off"
31 elif [ "$1" == "status" ]; then
32   cat "$LED3_PATH/trigger";
33 fi
34 echo "Scriptet er slut"
```

```
pwm.sh
1 #!/bin/bash
2 PWM_PATH="/sys/class/pwm/pwm-0:0"
3 if [ "$1" == "on" ]; then
4   echo 4000 > ${PWM_PATH}/period
5   echo 1000 > ${PWM_PATH}/duty_cycle
6   echo 1 > ${PWM_PATH}/enable
7   echo "PWM Period is now $(cat ${PWM_PATH}/period) ns."
8   echo "PWM Duty-cycle is now $(cat ${PWM_PATH}/duty_cycle) ns out of total period."
9 elif [ "$1" == "off" ]; then
10   echo 0 > ${PWM_PATH}/enable;
11 fi
```

janus@beaglebone:~/projects/E3ISD1/pwm\$ ./pwm.sh on  
PWM Period is now 4000 ns.  
PWM Duty-cycle is now 1000 ns out of total period.



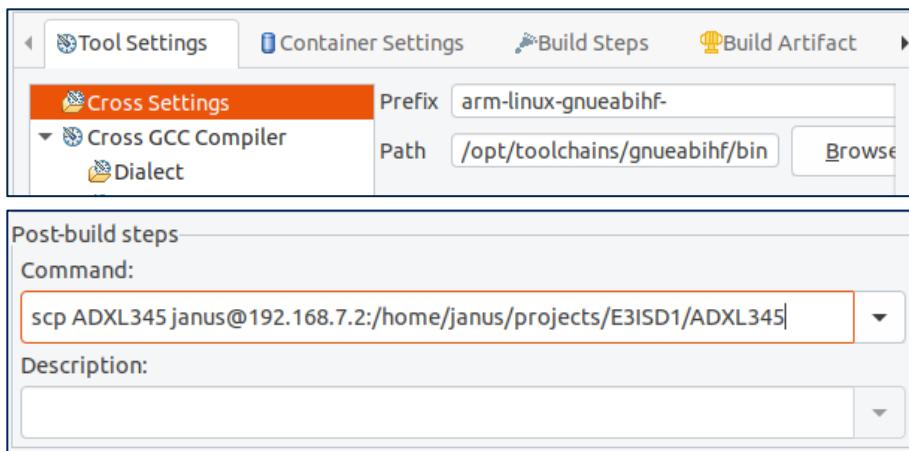
### Python (eksempel fra mikroprojekt)

```
handler.py
1 from client import MqttClient
2 import Adafruit_BBIO.GPIO as GPIO
3
4 def setup():
5   GPIO.setup("USR3", GPIO.OUT)
6
7 def on_message_callback(client, userdata, message):
8   # The received MQTT message is binary and must be decoded
9   msg = message.payload.decode("utf-8")
10  print(msg)
11  if msg == "ON":
12    GPIO.output("USR3", GPIO.HIGH)
13  elif msg == "OFF":
14    GPIO.output("USR3", GPIO.LOW)
15  else:
16    print("No action taken!")
17
18 def handler_sub_main():
19   # Define a subscriber that listens to all subjects
20   subscriber = MqttClient("MessageHandler1", on_message_callback)
21
22   # Only subscribe to relevant inbound messages
23   subscriber.subscribe("mikroprojekt/inbound")
24
25   print("Starting listening loop")
26   subscriber.loop()
27
28 def handler_pub_main():
29   #Sends off a single message and quits
30   publisher = MqttClient("MessagePublisher", on_message_callback)
31   publisher.publish("mikroprojekt/outbound", "Hello World!")
32
33 if __name__ == '__main__':
34   setup()
35   handler_sub_main()
```

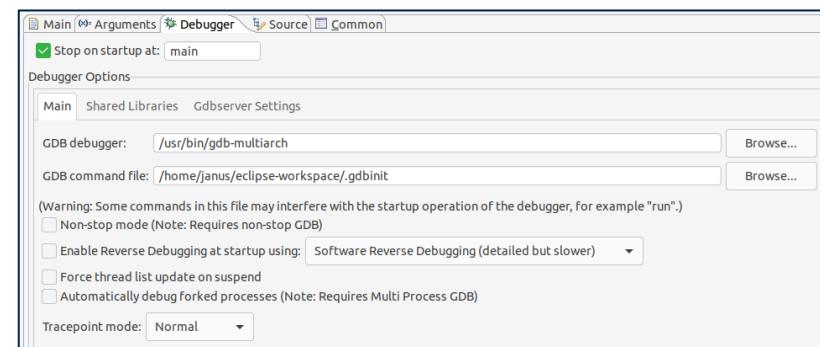
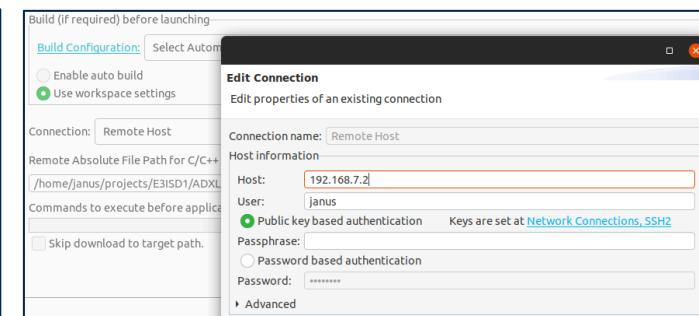
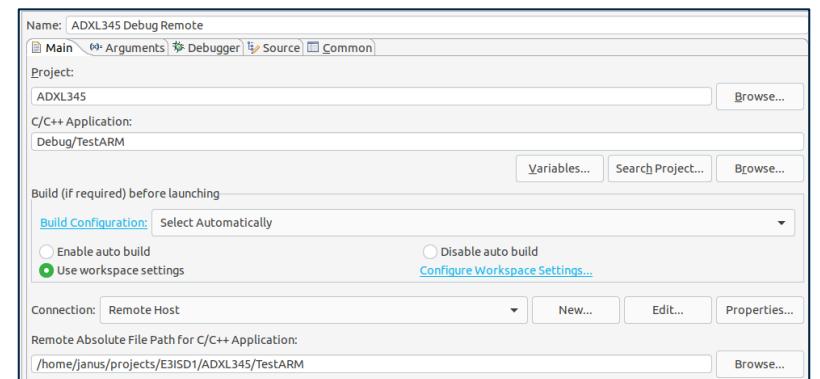
# SW5: BEAGLEBONE UDVIKLING

## CROSS-COMPILING

### Compiler-setup (ADXL345 over I2C)



### Debugger-setup (ADXL345-eksempel)



# SW5: BEAGLEBONE UDVIKLING

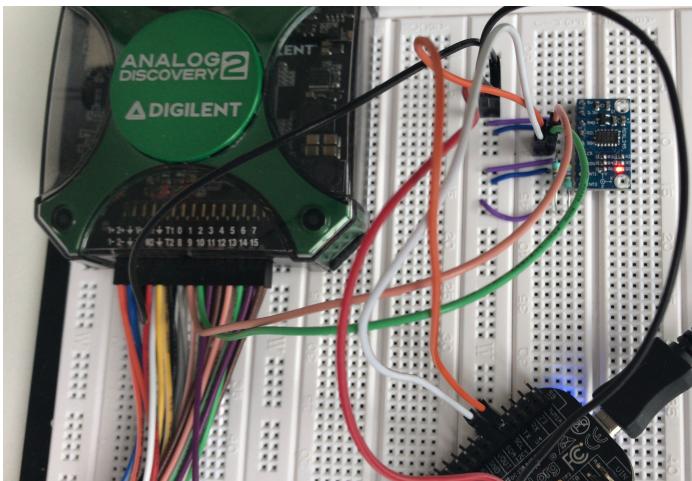
## REMOTE DEBUGGING

### Debugging-flow (ADXL345 over I2C)

The screenshot shows the Eclipse IDE interface with two panes. The left pane displays the source code for `ADXL345.cpp`, which includes comments for initializing the I2C bus, opening files for power control and data format, and reading registers. The right pane shows the variable values for `file` (int) and `count` (int), both set to 0. The console pane at the bottom shows the command to start the debugger (`gdbserver :2345`) and the resulting output from the Beaglebone, indicating successful connection and application startup.

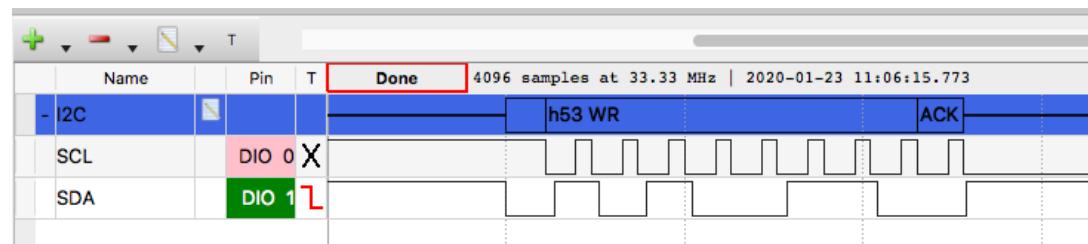
This screenshot continues the Eclipse IDE session. The code now includes a loop to read data from the sensor. The variable values remain the same. The console output shows the application running and printing sensor data to the terminal.

### Hardware



```
janus@beaglebone:~/projects/E3ISD1/ADXL345$ ./ADXL345
Starting the ADXL345 sensor application
The Device ID is: e5
The POWER_CTL mode is: 08
The DATA_FORMAT is: 00

X=-5 Y=7 Z=249 sample=24
```



# SW5: BEAGLEBONE UDVIKLING

## ØVRIGE MULIGHEDER

---

### Øvrige muligheder

- Debugging på target: gdb
- Profiling, memory-debugging: Valgrind (både på PB og via CDT)
- Emulering af ARM-processor med QEMU (qemu-static)

# SW5: EMNE OPSUMMERING OG PERSPEKTIVERING

---

# SW6: MIKROPROJEKT

## INDHOLD

**Teamprojekt: Marc, Tommy, Janus**

1. Krav / idé / formål
2. Analyse og design
3. Klargørelse af PocketBeagle
4. Implementering/løsning
5. Demo
6. Opsummering og perspektivering

6	Projektopgave	Gennemgå projektopgaven. Forklar <ul style="list-style-type: none"><li>• hvilke(t) krav du vil opfylde,</li><li>• analysen der bringer dig frem til den valgte løsning,</li><li>• det design du er nået frem til og endelig</li><li>• demonstrér din løsning</li></ul>
---	---------------	---

# SW6: MIKROPROJEKT KRAV

## Punkt

## Detaljer

Formål

- Vi vil benytte Beaglebone til at kommunikere med "Webinterface" specificeret i PRO3: Proof-of-concept, prototype og dokumentation
- Tager vores egen medicin ift. Pro3-specs...

Målsætning

Vi indsætter Beaglebone-platformen som en "Prøvestand" = klient, og på klienten implementerer vi funktionalitet som nedenstående med Python (interpreted).

Funktionalitet

Ambitionen er at implementere 1-3 i mikroprojektet

1. MQTT Publisher
  1. Klient skaber netværksforbindelse til en MQTT-broker.
  2. Klient sender "publish"-beskeder til broker.
2. MQTT Subscriber
  1. • Klient "subscriber" på et emne hos broker.
  2. • Klient modtager beskeder via broker
3. GPIO-interaktion (beskeder via MQTT benyttes til at interagere med GPIO)
  1. Tænd lysdiode på kommando

# SW6: MIKROPROJEKT ANALYSE OG DESIGN

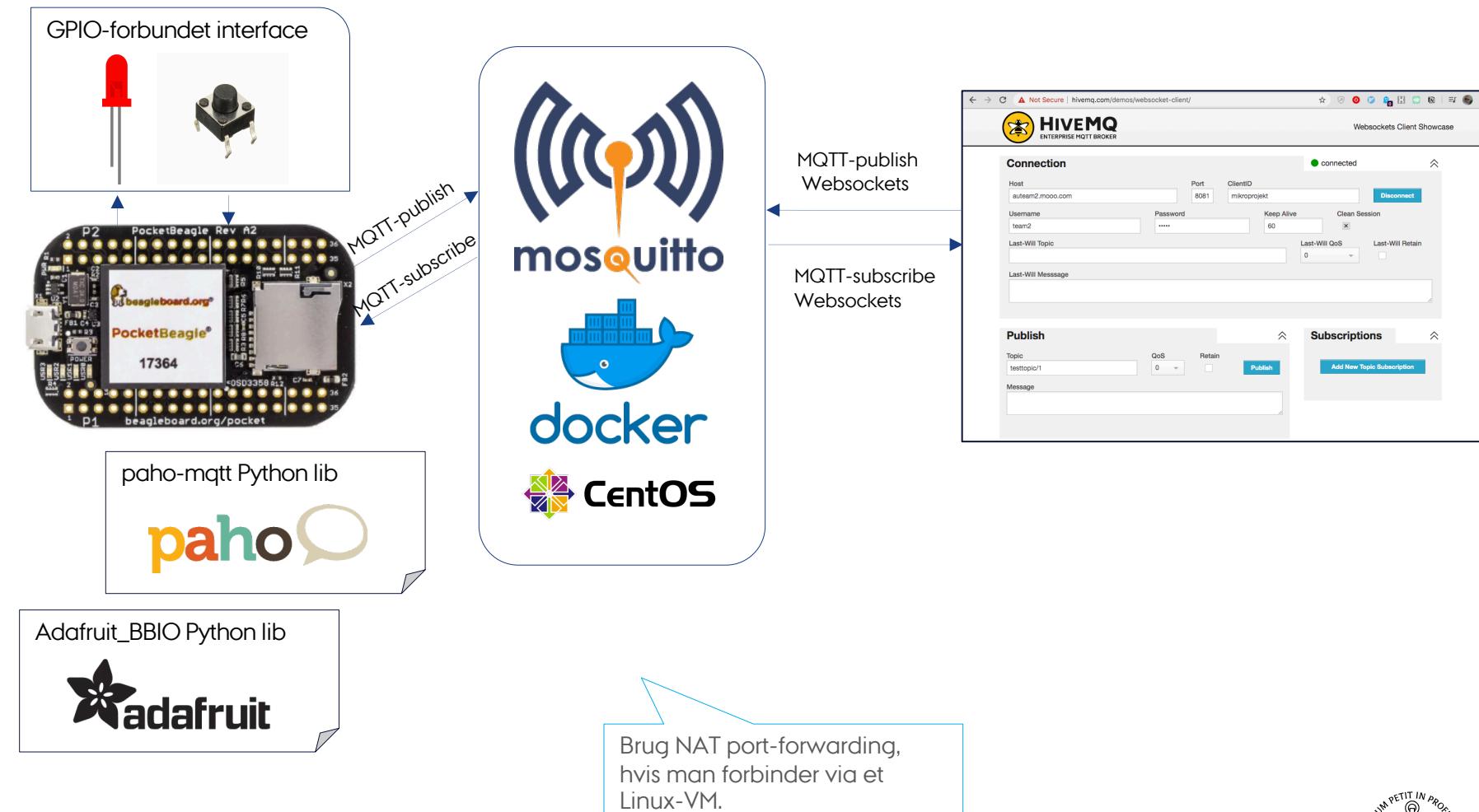
## Analyse

Idé, protokol, m.v. udvalgt i Pro3

- Vi er her "på den anden side" af vores egen Pro3-implementering.

Funktionalitet udvikles på Beaglebone (Debian) i dette mikroprojekt, så andre teams har noget at læne sig op ad.

- Kommunikerer vha. CentOS-Docker-Mosquitto stack fra Pro3



# SW6: MIKROPROJEKT ANALYSE OG DESIGN

## Design-valg

Sprog

Vi benytter Python, fordi:

- Kodegenbrug
- Ingen høje performancekrav, så Python er OK
- Ingen behov for at compile libmosquitto, eller cross-compile
- Ingen constraints ift. hvad de andre projektteams vil benytte af sprog til deres implementering
  - Portabilitet for Python er høj (nemt at få op at køre andre steder)

Interface

GPIO (men kunne også have været I2C, osv)...

- Vi vælger at blinke med en LED, fordi det er nemt at demo, og beviser at vi kan bruge GPIO
- GPIO-input ville have være næste lige så nemt

# SW6: MIKROPROJEKT

## KLARGØRELSE AF VÆRKTØJER

### Forberedelse

Build tools

```
janus@beaglebone:~/projects/E3ISD1$ sudo apt-get install build-essential python3-dev python3-setuptools  
python3-pip python3-smbus
```

Libraries

```
requirements.txt
```

```
1 paho-mqtt  
1 adafruit_BBIO
```

```
janus@beaglebone:~/projects/E3ISD1/mikroprojekt$ vim requirements.txt
```

```
janus@beaglebone:~/projects/E3ISD1/mikroprojekt$ sudo python3 -m pip install -r requirements.txt
```

Test

```
blink_pocket.py  
1 import Adafruit_BBIO.GPIO as GPIO  
1 import time  
2  
3 def setup():  
4     for i in ["USR0", "USR1", "USR2", "USR3"]:  
5         GPIO.setup(i, GPIO.OUT)  
6  
7 def pulse():  
8     setup()  
9     counter = 0  
10  
11     while True:  
12         GPIO.output("USR3", GPIO.HIGH)  
13         print("High")  
14         time.sleep(0.5)  
15         GPIO.output("USR3", GPIO.LOW)  
16         print("Low")  
17         time.sleep(0.5)  
18         counter += 1  
19  
20 if __name__ == '__main__':  
21     pulse()
```

```
janus@beaglebone:~/projects/E3ISD1/mikroprojekt$ sudo python3 blink_pocket.py  
High  
Low  
High  
Low
```

Observér, at USR3-dioden  
blinker i takt

En "svaghed" ved Adafruit\_BBIO er at  
det umiddelbart skal køres som *sudo*  
- medmindre man fikser nogle  
systemsettings...  
- udev-rules, setuid, osv.

# SW6: MIKROPROJEKT IMPLEMENTERING / LØSNING

## MQTT-klient

```
client.py
1 """
2 This module implements a custome MQTT client, which can be of two types:
3 - Subscriber: Listens for messages on a given topic
4 - Publisher: Sends messages on a given topic
5 """
6 import paho.mqtt.client as mqtt
7
8 class MqttClient():
9
10     # These should be gotten from the environment ideally
11     broker_address = "auteteam2.mooo.com"
12     broker_port = "1883"
13     username = "team2"
14     password = "team2"
15
16     # This method is the same for all instances of the class
17     @staticmethod
18     def on_connect(client, userdata, flags, rc):
19         #print("Connected with result code " + str(rc))
20         pass
21
22     # For outputting log messages to console
23     @staticmethod
24     def on_log(client, userdata, level, buf):
25         pass
26         #print("log: ", buf)
27
28     # By default, do nothing on_message
29     @staticmethod
30     def on_message_default(client, userdata, message):
31         pass
32
```

```
34     #Initialize the client, straight on create
35     def __init__(self, name, on_message, will_message="Logging off"):
36         """__init__ Handles all setup and connection when object is initialized.
37         @param: name is the name of the client, as will be shown on the server
38         @param: on_message is the callback used when this client receives a message
39         @param: will_message is the "Last Will" message sent when client loses conn
40         """
41
42         self.client = mqtt.Client(client_id=name,
43                               clean_session=True,
44                               userdata=None,
45                               transport="tcp")
46
47         self.client.username_pw_set(MqttClient.username, MqttClient.password)
48         self.client.on_connect = MqttClient.on_connect
49         self.client.on_message = on_message
50
51         # In production, let's consider disabling logging or routing to a file
52         #self.client.on_log = MqttClient.on_log
53         #self.client.enable_logger()
54
55         # This ensures, that there is some sort of goodbye on losing connection
56         self.client.will_set(name, will_message)
57
58         # Connect immediately
59         self.client.connect(MqttClient.broker_address)
60
61     def publish(self, topic, payload):
62         return self.client.publish(topic, payload)
63
64     def subscribe(self, topic):
65         return self.client.subscribe(topic)
66
67     def loop(self):
68         return self.client.loop_forever(retry_first_connection=False)
69
70     def disconnect(self):
71         return self.client.disconnect()
72
```

NORMAL client.py python utf-8[unix] 49% 34/

## Handler

```
handler.py
1  from client import MqttClient
2  import Adafruit_BBIO.GPIO as GPIO
3
4  def setup():
5      GPIO.setup("USR3", GPIO.OUT)
6
7  def on_message_callback(client, userdata, message):
8      # The received MQTT message is binary and must be decoded
9      msg = message.payload.decode("utf-8")
10     print(msg)
11     if msg == "ON":
12         GPIO.output("USR3", GPIO.HIGH)
13     elif msg == "OFF":
14         GPIO.output("USR3", GPIO.LOW)
15     else:
16         print("No action taken!")
17
18  def handler_sub_main():
19      # Define a subscriber that listens to all subjects
20      subscriber = MqttClient("MessageHandler1", on_message_callback)
21
22      # Only subscribe to relevant inbound messages
23      subscriber.subscribe("mikroprojekt/inbound")
24
25      print("Starting listening loop")
26      subscriber.loop()
27
28  def handler_pub_main():
29      #Sends off a single message and quits
30      publisher = MqttClient("MessagePublisher", on_message_callback)
31      publisher.publish("mikroprojekt/outbound", "Hello World!")
32
33
34 if __name__ == '__main__':
35     setup()
36     handler_sub_main()
```

# SW6: MIKROPROJEKT

DEMO ☺

---

<https://youtu.be/p4kX3HWeck0>

[https://github.com/AUTeam2/E3ISD1\\_mikoprojekt](https://github.com/AUTeam2/E3ISD1_mikoprojekt)

# SW6: MIKROPROJEKT

## OPSUMMERING OG PERSPEKTIVERING

---

### Mulige udvidelser

- Aftaste en knap via GPIO
  - Setup er `GPIO.setup("pin", GPIO.IN)`
  - Aftastning er fx
    - `GPIO.wait_for_edge("pin", GPIO.RISING)` eller
    - `GPIO.add_event_detect("pin", GPIO.FALLING)`, samt if `GPIO.event_detected("pin")`
    - `add_event_callback("pin", callback_func)`
- Analog interfacing via
  - Input: PocketBeagle's 12-bits ADC
  - Output: PocketBeagle's PWM
- Måle accelerometer via bus-IO I2C, SPI, osv...



AARHUS  
UNIVERSITET