

//

ELEKTRA INVITERER DEG TIL AVR-KURS



KURSHOLDER:
JØRGEN STEEN

30. MARS | 4. APRIL | 6. APRIL
KL.17:00 - 21:00 alle dagene

Pris: **150,-** for Elektra-medlemmer | **250,-** for TTS-medlemmer
300,- for ikke medlemmer

Forkunnskap: Ingen, men vi har spennende og utfordrende ekstra oppgaver!

Kurset er ment for de med liten eller ingen erfaring med AVR programmeringer
(AVR-programmering er 8 bits mikrokontrollere man programmer i C)

Brus og pizza vil bli servert i løpet av kveldene

Frist for påmelding: **Søndag 26. mars**
For mer info, følg Elektra facebook-siden/gruppa, eller QR-kodene

FB-event



I samarbeid med Microchip



Innmeldings-
skjemæt



Contents

1	Oversikt	1
1.1	Dag 1: Bitmanipulasjon, la det bli lys(emitterende diode).	1
1.2	Dag 2: Klokke, tid for PWM!	1
1.3	Dag 3: Analog til digital konvertering(ADC).	1
2	Dagen 1: Kontrollere en bit	2
2.1	Oppgavene for dag 1	2
2.2	Visuel hjelp	3
3	Dagen 2: Bruke den interne klokken	5
4	Dagen 3: Analog til digital Konvertering(ADC)	5

1 Oversikt

1.1 Dag 1: Bitmanipulasjon, la det bli lys(emitterende diode).

Timeplan:

- Spise pizza og få mikrokontroller.
- Lære om bitmanipulasjon.
- Programmere mikrokontrolleren.
- Nøkkelsbegreper: Bitmanipulasjon, bitmaske, sette bits.

1.2 Dag 2: Klokke, tid for PWM!

Timeplan:

- Spise pizza.
- Lære om den interne klokken og hvordan man bruker den.
- Programmere mikrokontrolleren.
- Nøkkelsbegreper: Funksjon, bitmaske, register, duty. cycle, prescaling, interrupt.

1.3 Dag 3: Analog til digital konvertering(ADC).

Timeplan:

- Spise pizza.
- Lære om den analog til didgtal konvertering og hvordan man brukere det
- Programmere mikrokontrolleren
- Få hjelp til egene projekter.
- Nøkkelsbegreper: ADC, referansespenning, bitmaske.

2 Dagen 1: Kontrollere en bit

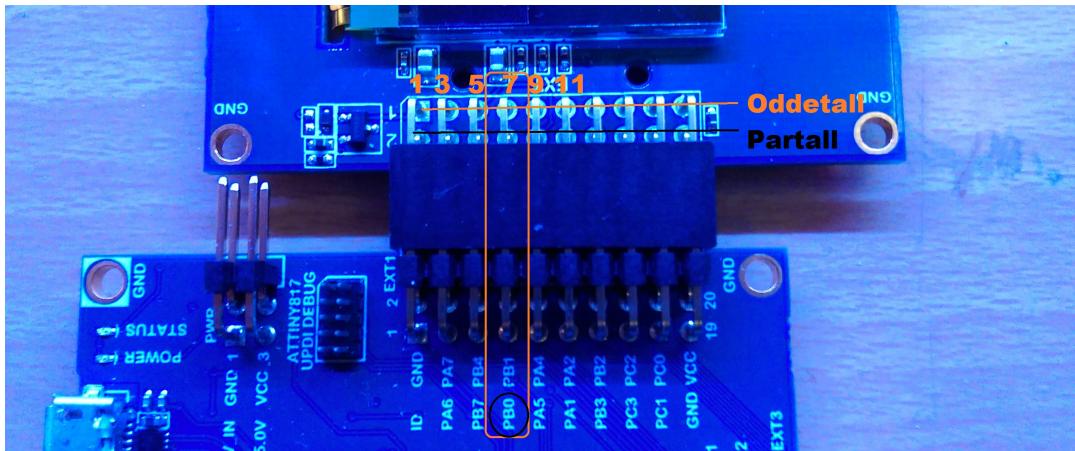
Målet for den første dagen er å forstå bitmanipulasjon og bruke denne kunnskapen til å kontrollere tilstanden til en bit. Bitmanipulasjon er en veldig viktig del av AVR-programmering; kunnskapen om dette kan dere bruke i så å si all mikrokontroller programmering. Under de 4 oppgavene eksempel kode. Huske å ta med delay biblioteket og det som skjer i main(void) skjer kun en gang, mens det som skjer i while(1) så lengde det er strøm på kontrolleren.

2.1 Oppgavene for dag 1

- Oppgave 1: La det bli LED
 - Beskrivelse:
I denne oppgaven skal du bruke det du har lært om bit-manipulasjon til å veksle tilstanden til en IO-port mellom høy og lav.
 - Hvordan:
 1. Velg LEDen du vil at skal blinke og se hvilken IO-port den er koblet til
 2. Sett denne IO-porten som en utgang
 3. Deretter sette det høyt, legge til litt venting for å så sette den lav.
 - Verktøy: Eller-opperasjon, Og-opperasjon, not-opperasjon og delay.
- Oppgave 2: Slå av og på flere LED
 - Hvordan: Velg neste LED du vil at skal blinke å gjør det samme som i oppgave 1.
 - Verktøy: Eller-opperasjon, Og-opperasjon, not-opperasjon og delay. Her kan man også bruke XOR-operasjon hvis man føler seg dristig.
- Oppgave 3: Bruke en knapp for å styre LED
 - Hvordan:
 1. Velg knappen du skal bruke og se hvilken IO-port den er koblet til.
 2. Sette denne som en utgang
 3. Slå på pull-up motstanden.
Hint: Denne finnes i et eget register PORTx.CTRLn
 4. Lag en if-løkke som får et av lysene til å gjøre noe hvis knappen er trykket.
Hint: Når man slår på pull-up motstanden har knappen en høy verdi når den ikke er trykket.
 5. Verktøy: If-løkke, og-operasjon.
 - Ekstra oppgave: Knapp som toggler kun en gang.
 - Hvordan:
 1. Første det samme som i oppgavene over.
 2. Sett opp noe som hindrer at knappen gjør det du er har bedt flere ganger hvis du holder den inne.

2.2 Visuel hjelp

Finne fram på brettet



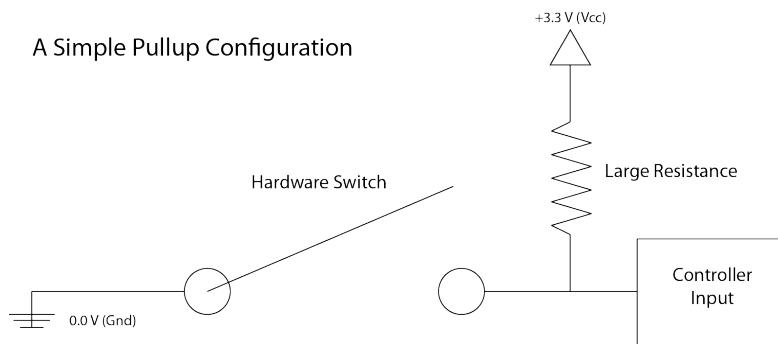
Oppgave 1

```

7  /* Det er en 20Mhz klokke på brettet, men den forhånds prescaleret med en faktor på 6. 20/6 = 3.333333Mhz
8  Dette kan forandres hvis man ønsker.
9 */
10 #define F_CPU 3333333UL
11 #include <avr/io.h> //Denne tar man alltid med med AVR
12 #include <util/delay.h> //tar med biblioteket for delay
13
14 int main(void)
15 {
16     PORTB.DIR = PORTB.DIR | 0b00000001;    //setter PB0(PORTB pin 0) som en utgang
17     while (1)
18     {
19         PORTB.OUT = PORTB.OUT | 0b00000001; // PB0 høy så leden den går av
20         _delay_ms(1000); //Venter i 1 sekund
21         PORTB.OUT = PORTB.OUT & ~0b00000001; //setter PB0 lavt så LEDen går på
22         _delay_ms(1000); //Venter i 1 sekund
23     }
24 }

```

Pull-up illustrasjon



Oppgave 3

```

7  /* Det er en 20Mhz klokke på brettet, men den forhånds prescalet med en faktor på 6. 20/6 = 3.333333Mhz
8  Dette kan forandres hvis man ønsker.
9 */
10 #define F_CPU 3333333UL
11 #include <avr/io.h> //Denne tar man alltid med med AVR
12 #include <util/delay.h> //tar med biblioteket for delay
13 #define button1_bm 0b00100000 //PA5 PORTA PIN 5 Knapp 1 på Oled ext
14
15 int main(void)
16 {
17     PORTB.DIR = PORTB.DIR | 0b00000001; //setter PB0(PORTB pin 0) som en utgang
18     PORTA_PIN5CTRL |= (1<<3); // 0b00001000 Setter den 3 pinen i ctrl registeret til høy for å aktivitere pullup.
19
20     while (1)
21     {
22         if (!(PORTA.IN & button1_bm))
23         {
24             PORTB.OUT = PORTB.OUT | 0b00000001; // PB0 høy så leden den går av
25             _delay_ms(1000); //Venter i 1 sekund
26             PORTB.OUT = PORTB.OUT & ~0b00000001; //setter PB0 lavt så LEDen går på
27             _delay_ms(1000); //Venter i 1 sekund
28         }
29     }
30 }
  
```

3 Dagen 2: Bruke den interne klokken

4 Dagen 3: Analog til digital Konvertering(ADC)

Forandringer Her er noen av forandringene fra det gamle til det ny.

Alt skrives nå som strukturer. x: Porten du bruker(A-C). n: biten du bruker(0-7)

Gammel	Ny	Forklaring
DDR _x	PORT _x .DIR	Sette inngang/utgang
PORT _x	PORT _x .OUT	Sette en bit høy/lav
PIN _x	PORT _x .IN	Lese verdien til registeret
PORT _x	PORT _x .PINCTRL _n	Sette en pull-up motstand på inngangen