AVR-KURS

KURSHOLDER:
JØRGEN STEEN

3 dagers plan

Dag 1: Bit-manipulasjon

- Sette en bit høy, lav
- Toggling av bit.
- Bruk av knapp.

Dag 2: Klokke og timer

- Bruke mikrokontrollerens klokke for å utføre operasjoner med nøyaktig tidsintervall.
- PWM, timer, interrupts, etc.

Dag 3: ADC

- Analog til digital konvertering
- Leke seg med ideer.

Atmel – AVR - Microchip

Hva er AVR?

- AVR er en familie mikrokontrollere fra Atmel.
- AVR programmering = Programmering på en Atmel chip.

Hva er Atmel?

- Det tidligere navnet på bedriften du er hos nå.
- AVR kunne gjøre mer på en klokkesyklus enn konkurentene.

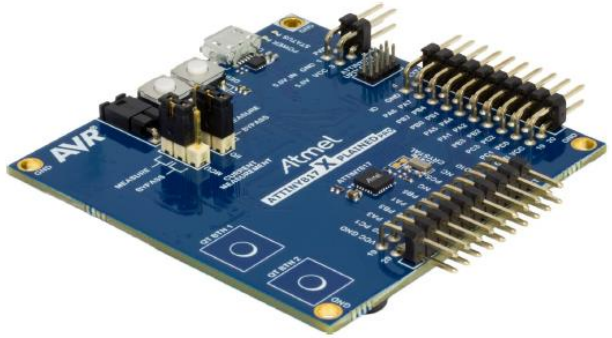
Microchip?

- Tidligere konkurrent av Atmel som også lager mikrokontrollere.
- Atmel ble kjøpt opp av Microchip, så #define Atmel Microchip

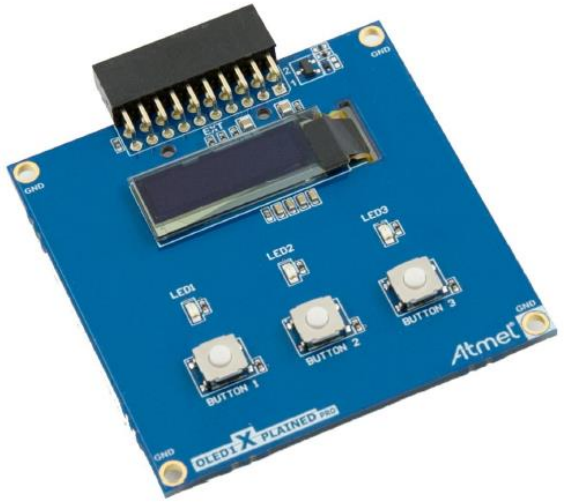
AVR-KURS

KURSHOLDER:
JØRGEN STEEN

ATtiny817 Xplained Pro

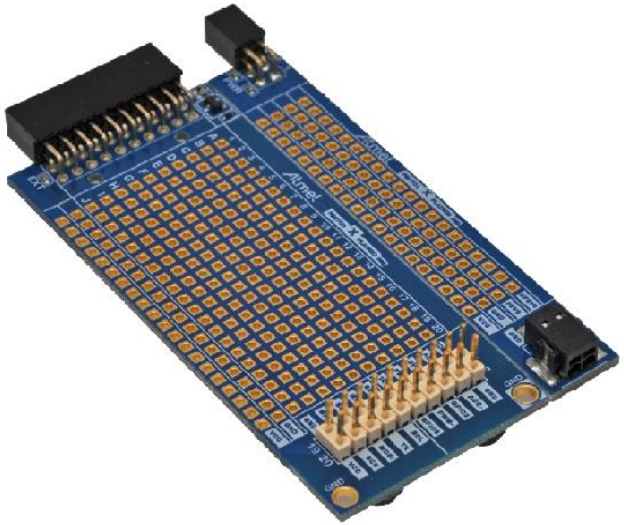


OLED1 Xplained Pro



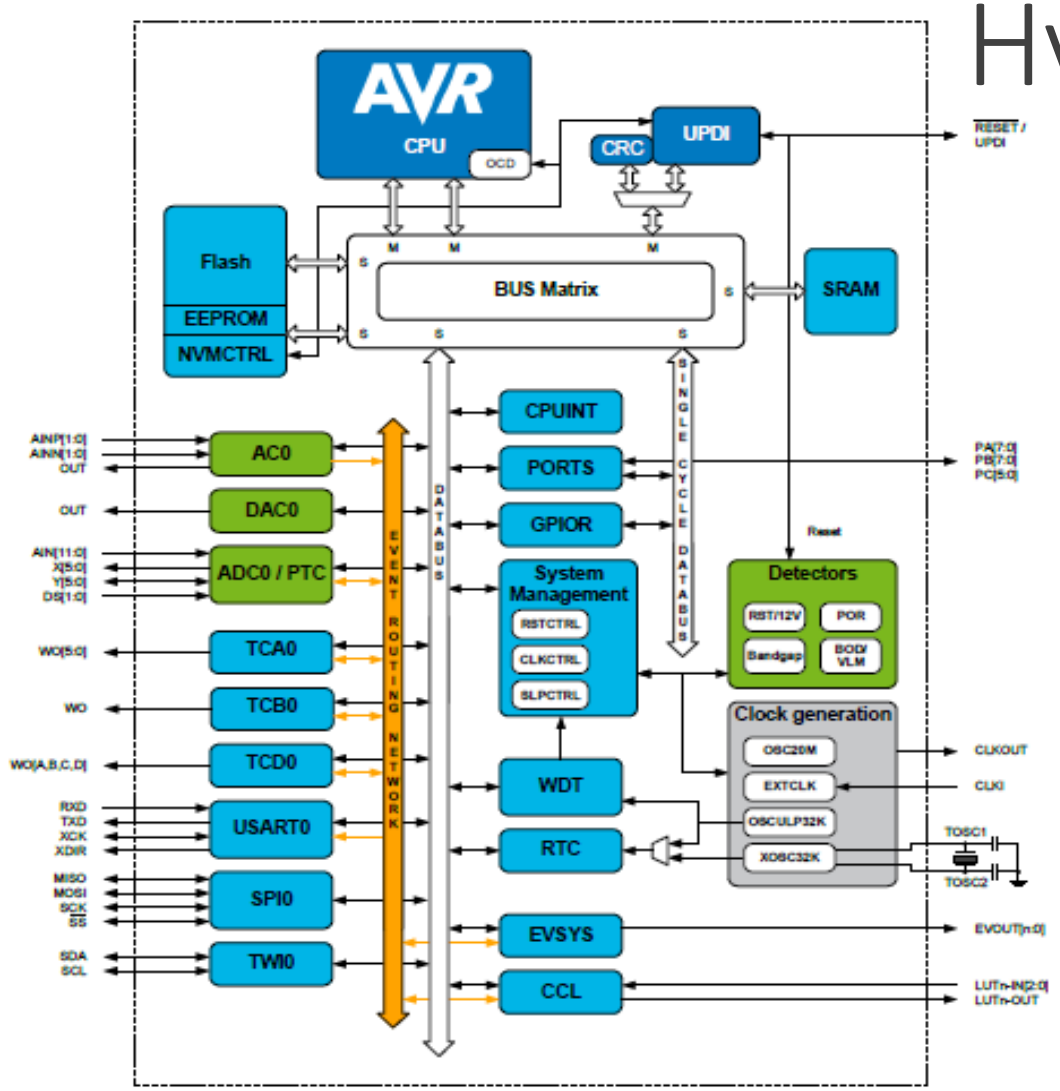
I/O1 Xplained Pro

Xplained Proto



elektra
linjeforeningen for
elektrofag ved NTNU

Hva er en mikrokontroller?



En liten datamaskin med mange forskjellige moduler.

Modulene har hver sin funksjon.

elektra
linjeforeningen for elektrofag ved NTNU

Dag 1

Hva skal vi gjøre?

- 1. Få LED til å blinke og lyse.
- 2. Kontrollere LEDs med knapper.

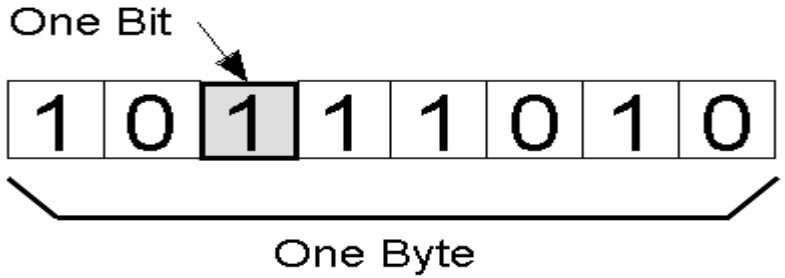
Hvordan får vi dette til?

- Sette bits høye og lave ved hjelp av bit-manipulasjon.

Hvorfor det?

- Et høyt bit representerer (1.8-5.5)V og et lavt bit er 0V

Bitmanipulasjon?



Hva er bit-manipulasjon?

- Handling å forandre en bit eller annen informasjon som er mindre enn 1 byte (8 bit).
- Bruker boolsk algebra og bit-operatorer.

Hva er boolsk algebra?

- Sant eller Usant.
- Høy eller lav.
- (1.8-5.5)V eller 0V.
- Likheter med formal logikk

Hva er bit-operatorer?

- De mest brukte er «eller», «og», «xor», «not», «leftshift» og «rightshift».
- Gjør det mulig å forandre kun 1 bit, når man skriver til en byte.

Eller-operasjon: Sette en bit høy

Resultatet blir sant hvis en av inngangen er sanne

```
0 | 0 = 0
0 | 1 = 1
1 | 0 = 1
1 | 1 = 1
```

Eller-operasjon

01000110

| 01100101

= 01100111

Hvordan det vil se ut i kodenform

```
  01000110
| 01100101
= 01100111
```

01100111 = 01100101 | 01000110

PORTx.OUT = PORTx.OUT |01000110

PORTx.OUT |= 01000110

Nye verdien til PORTx: 01100111

Og-operasjon: Sette en bit lav

Resultatet blir sant hvis begge inngangen er sanne

```
0 & 0 = 0
0 & 1 = 0
1 & 0 = 0
1 & 1 = 1
```

Og-operasjon

01000110

& 01100101

= 01000100

Hvordan det vil se ut i kodenform

01000110
& 01100101
= 01000100

01000100 = 01100101 & 01000110

PORTx.OUT = PORTx.OUT &01000110

PORTx.OUT &= 01000110

Nye verdien til PORTx: 01000100

elektra
linjeforeningen for
elektrofag ved NTNU

Sette en enkelt bit høy og så lav

Svadakode

PORTB sin verdi: 10000000

PORTB.OUT = PORTB.OUT| 00001000

PORTB sin verdi: 10001000

PORTB.OUT &= 11110111

PORTB sin verdi: 10000000

Utregning

```
  00001000
| 10000000
= 10001000

  10001000
& 11110111
= 10000000
```

elektra
linjeforeningen for elektrofag ved NTNU

Not-operasjon: Inverterer en bit

Resultatet blir det inverse av bitet sin verdi

~1 = 0

~0 = 1

Not-operasjon

~ 01000110

= 10111001

~ 00010000

= 11101111

Sette en enkelt bit høy og så lav med not

~ 00010000
= 11101111

Svadakode

PORTB sin verdi: 10000000

PORTB.OUT = PORTB.OUT| 00001000

PORTB sin verdi: 10001000

PORTB.OUT &= ~00001000

PORTB sin verdi: 10000000

Utregning

00001000
| 10000000
= 10001000

~00001000
& 10001000
= 10000000

elektra
linjeforeningen for elektrofag ved NTNU

XOR-operasjon: Toggle en bit

Resultatet blir sant hvis kun en av inngangene er høye

```
0 ^ 0 = 0
0 ^ 1 = 1
1 ^ 0 = 1
1 ^ 1 = 0
```

XOR-operasjon

01000110

^ 01100101

= 00100011

Hvordan det vil se ut i koden

```
 01000110
^01100101
=  00100011
```

00100011 = 01100101 ^ 01000110

PORTx.OUT = PORTx.OUT ^01000110

PORTx.OUT ^= 01000110

Nye verdien til PORTx: 01000011

Shift-operasjon: Flytte en bit

Tar alle verdiene og flytter de til en retning

```
  01100101
<< 00000010    (verdi: 2)
=  10010100
```

elektra
linjeforeningen for elektrofag ved NTNU

Venstre skift

```
  01100101
<< 00000010
=  10010100
```

10010100 = 01100101 << 00000010

10010100 = 01100101 << 2

Mer vanlig:

00000100 = 00000001 << 2

00000001 = 00000001 << 0

elektra
linjeforeningen for
elektrofag ved NTNU

Hvordan det vil se ut i kodenform

00000100 = 00000001 << 2

00000100 = 1<<2

01000000 = 1<<6

01100100 = 01100000 | (1<<2)

01100100 = PORTx.OUT | (1<<2)

PORTx.OUT |= (1<<2)

Nye verdien til PORTx: 01100100

elektra
linjeforeningen for
elektrofag ved NTNU

Høyere skift

00011001 = 01100101 >> 00000010

00011001 = 01100101 >> 2

00000100 = 00010000 >> 2

00000010 = 01000000 >> 5

Forskjellen på bit nivå operasjoner og større operasjoner

|, &, ^,~,<<,>> fungerer kun på bit nivå.

Skal man opp til byte nivå eller større bruker man andre tegn

| → || & → && ~ → !

Bit-operasjoner

Eller-operasjon:
- Setter en bit høy.
- Jeg holder et kurs ELLER er hjemme = sant
- Syntax: «|».

Og-operasjoner
- Setter en bit lav.
- Sammenligne to bitmasker.
- Jeg holder et kurs OG er hjemme = Usant
- Syntax: «&» .

Not-operasjoner
- Inverterer svare.
- Syntax: For bit «~» og for logikk «!».

XOR-operasjoner
- Setter bit høy hvis den er lav og omvendt(toggle)
- Er den 5V så blir den 0V, og 0V blir til 5V
- Jeg holder ENTEN et kurs ELLER er jeg hjemme = SANT
- Syntax: «^»

Bit-shifting
- Flytter bitene X antall steg i retningen man ønsker.
- Venstreskift brukes masse i kombinasjon med maskenavn for å få en god og leselig kode.
- Syntax: «<<», «>>»
- Ideellkode: #define LED1_bp 5 //PB5
- PORTB.OUT |= (1<<LED1_bp);

Hva blir den ukjente verdien «?»?.

1. ???????? = 00010000 | 00000100

2. 01000010 = 00000010 | ????????

3. ???????? = 00010100 & ~00000100

4. 00000010 = 01000010 & ????????

5. ????????? = 01000010 ^ 01000000

Oppgave med ukjente:

XXXX1XXX = XXXXXXXX | 00001000

XXXX1XXX |= 00001000

6. ???????? = XXXXXXXX | 11011111

7. ???????? |= 00100000

8. ???????? |= (1<<3)

elektra
linjeforeningen for elektrofag ved NTNU

Hva blir den ukjente verdien «?»?.

1. 00010100 = 00010000 | 00000100
2. 01000010 = 00000010 | 01000000
3. 00010000 = 00010100 & ~00000100
4. 00000010 = 01000010 & ~01000000
5. 00000010 = 01000010 ^ 01000000

Oppgave med ukjente:

XXXX1XXX = XXXXXXXX | 00001000

XXXX1XXX |= 00001000

6. 11X11111 = XXXXXXXX | 11011111
7. XX1XXXXX |= 00100000
8. XXXX1XXX |= (1<<3)

elektra
linjeforeningen for elektrofag ved NTNU

«0b» foran et tall betyr at det skal leses binært.

Hvilken binærverdi har PORTB?

1. PORTB.OUT |= 0b00100000;
2. PORTB.OUT = ~0b00100000;
3. PORTB.OUT &= 0b00100000;
4. PORTB.OUT &= ~0b00100000;
5. PORTB.OUT = 0b00100000;
6. PORTB.OUT ^= 0b00100000;
7. PORTB.OUT |= (1<<3);
8. PORTB.OUT &= ~(1<<3);

elektra
linjeforeningen for elektrofag ved NTNU

Hvilken binærverdi har PORTB?

1. PORTB.OUT |= 0b00100000

   PORTB: 0bXX1XXXXX

2. PORTB.OUT = ~0b00100000

   PORTB: 0b11011111

3. PORTB.OUT &= 0b00100000

   PORTB: 0b00X00000

4. PORTB.OUT &= ~0b00100000

   PORTB: 0bXX0XXXXX

5. PORTB.OUT = 0b00100000

   PORTB: 0b00100000

6. PORTB.OUT ^= 0b00100000

   PORTB: 0bXX «~X»XXXXX

7. PORTB.OUT |= (1<<3)

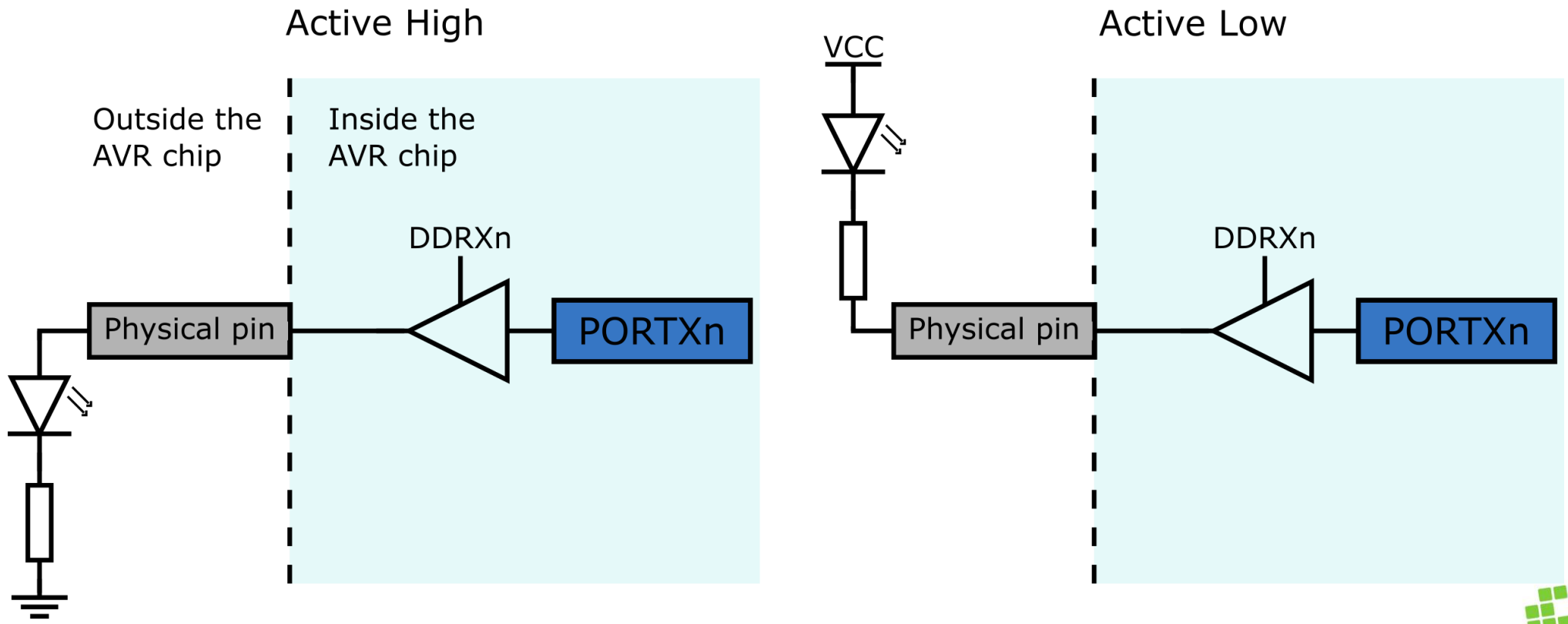
   PORTB: 0bXXXX1XXX

8. PORTB.OUT &= ~(1<<3)

   PORTB: 0bXXXX0XXX

elektra
linjeforeningen for
elektrofag ved NTNU

Slå på et LED: Skal jeg sette et bit høy eller lav.



elektra
linjeforeningen for elektrofag ved NTNU

PORT kommunikasjon

16.4 Register Summary - PORT

Offset	Name	Bit Pos.								
0x00	DIR	7:0	DIR[7:0]							
0x01	DIRSET	7:0	DIRSET[7:0]							
0x02	DIRCLR	7:0	DIRCLR[7:0]							
0x03	DIRTGL	7:0	DIRTGL[7:0]							
0x04	OUT	7:0	OUT[7:0]							
0x05	OUTSET	7:0	OUTSET[7:0]							
0x06	OUTCLR	7:0	OUTCLR[7:0]							
0x07	OUTTGL	7:0	OUTTGL[7:0]							
0x08	IN	7:0	IN[7:0]							
0x09	INTFLAGS	7:0	INT[7:0]							
0x0A ... 0x0F	Reserved									
0x10	PINCTRL0	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x11	PINCTRL1	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x12	PINCTRL2	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x13	PINCTRL3	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x14	PINCTRL4	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x15	PINCTRL5	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x16	PINCTRL6	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x17	PINCTRL7	7:0	INVEN				PULLUPEN	ISC[2:0]		

I dag: DIR, OUT, IN og PINCTRLn. Ikke tenk på resten.

Gammel syntax:

- PORTx.DIR -> DDRx
- PORTx.OUT -> PORTx
- PORTx.IN -> PINx
- PORTx.PINCTRLn ->PORTx (pull-up)

elektra
linjeforeningen for elektrofag ved NTNU

Programmerings eksempel!

Hva skal vi gjøre?

Hvordan får vi det til?

Hvordan får man det til med denne brikken?

- Datablad!

Tid for datablad!

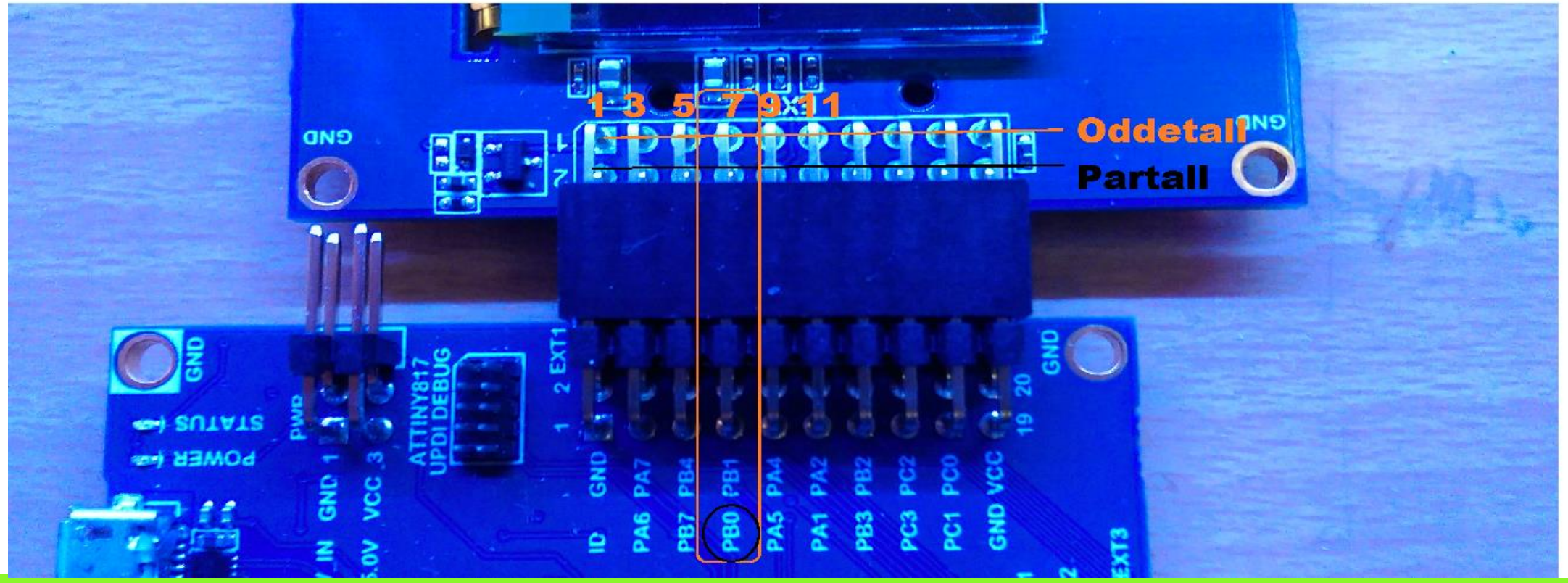
elektra
linjeforeningen for
elektrofag ved NTNU

4.3.1. LEDs

There are three yellow LEDs available on OLED1 Xplained Pro. The LEDs can be activated by driving the connected I/O line low.

Table 4-3. LED Connections

Pin on EXT connector	Silk screen marking
7	LED1
8	LED2
6	LED3

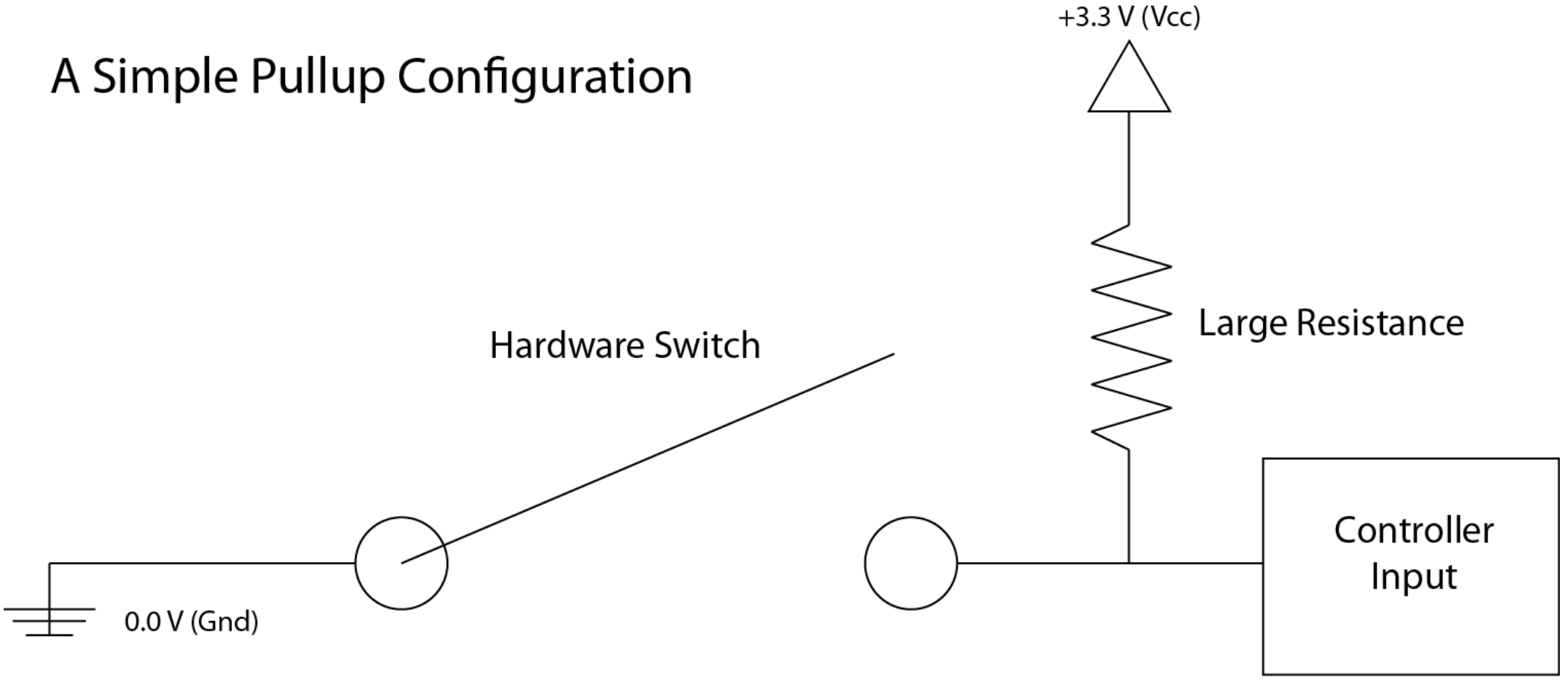


```c
/* Det er en 20Mhz klokke på brettet, men den forhånds prescalet med en faktor på 6. 20/6 = 3.333333Mhz
   Dette kan forandres hvis man ønsker.
*/
#define F_CPU 3333333UL
#include <avr/io.h> //Denne tar man alltid med med AVR
#include <util/delay.h> //tar med biblioteket for delay

int main(void)
{
    PORTB.DIR = PORTB.DIR | 0b00000001;   //setter PB0(PORTB pin 0) som en utgang
    while (1)
    {
        PORTB.OUT = PORTB.OUT | 0b00000001;  // PB0 høy så leden den går av
        _delay_ms(1000); //Venter i 1 sekund
        PORTB.OUT = PORTB.OUT & ~(0b00000001); //setter PB0 lavt så LEDen går på
        _delay_ms(1000);  //Venter i 1 sekund

    }
}
```

16.4 Register Summary - PORT

Offset	Name	Bit Pos.								
0x00	DIR	7:0	DIR[7:0]							
0x01	DIRSET	7:0	DIRSET[7:0]							
0x02	DIRCLR	7:0	DIRCLR[7:0]							
0x03	DIRTGL	7:0	DIRTGL[7:0]							
0x04	OUT	7:0	OUT[7:0]							
0x05	OUTSET	7:0	OUTSET[7:0]							
0x06	OUTCLR	7:0	OUTCLR[7:0]							
0x07	OUTTGL	7:0	OUTTGL[7:0]							
0x08	IN	7:0	IN[7:0]							
0x09	INTFLAGS	7:0	INT[7:0]							
0x0A ... 0x0F	Reserved									
0x10	PINCTRL0	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x11	PINCTRL1	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x12	PINCTRL2	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x13	PINCTRL3	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x14	PINCTRL4	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x15	PINCTRL5	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x16	PINCTRL6	7:0	INVEN				PULLUPEN	ISC[2:0]		
0x17	PINCTRL7	7:0	INVEN				PULLUPEN	ISC[2:0]		

Pull-up (og pull-down)



elektra
linjeforeningen for
elektrofag ved NTNU

Mer datablad

elektra
linjeforeningen for
elektrofag ved NTNU

```c
/* Det er en 20Mhz klokke på brettet, men den forhånds prescalet med en faktor på 6. 20/6 = 3.333333Mhz
   Dette kan forandres hvis man ønsker.
 */
#define F_CPU 3333333UL
#include <avr/io.h> //Denne tar man alltid med med AVR
#include <util/delay.h> //tar med biblioteket for delay
#define button1_bm 0b00100000 //PA5  PORTA PIN 5 Knapp 1 på Oled ext

int main(void)
{
    PORTB.DIR = PORTB.DIR | 0b00000001;   //setter PB0(PORTB pin 0) som en utgang
    PORTA_PIN5CTRL |= (1<<3); // 0b00001000 Setter den 3 pinen i ctrl registeret til høy for å aktivetere pullup.

    while (1)
    {
        if (!(PORTA.IN & button1_bm))
        {
            PORTB.OUT = PORTB.OUT | 0b00000001;  // PB0 høy så leden den går av
            _delay_ms(1000); //Venter i 1 sekund
            PORTB.OUT = PORTB.OUT & ~(0b00000001); //setter PB0 lavt så LEDen går på
            _delay_ms(1000);  //Venter i 1 sekund
        }
    }
}
```