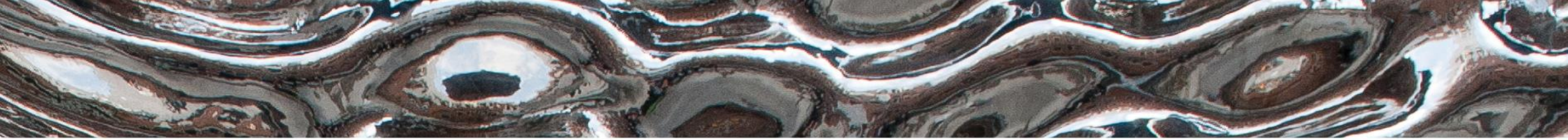


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN 1080

Mikrokontrollere, Busser
Fremgangsmåte for mekatronikkprosjekter

Yngve Hafting, 2021

Hvor står vi og hvor går vi…

Kort om emnet

- *Grunnleggende analog elektronikk*, *sensorer og* **sensor grensesnitt**, *aktuatorer*. *Programmering av mekatroniske systemer*.

Hva lærer du?

Etter å ha tatt IN1080 kan du:

- *forstå virkemåten til analoge kretser. Aktuelle begreper er: strøm, spenning, motstand, effekt, impedans, likestrøm, vekselstrøm, RCL, MOS, FET, OPAmp*
- *bruke klassiske analysemetoder basert på Kirchoff, Thevenin og Nortons teoremer*
- *forstå og anvende sensorer, signalkondisjonering og konvertering, samt noen komponent-komponent busser*
- **bygge og programmere enkle mekatroniske systemer med mikrokontroller, aktuatorer og sensorer**
- *forstå grunnleggende kontrollteori og virkemåte for PIDkontrollere*

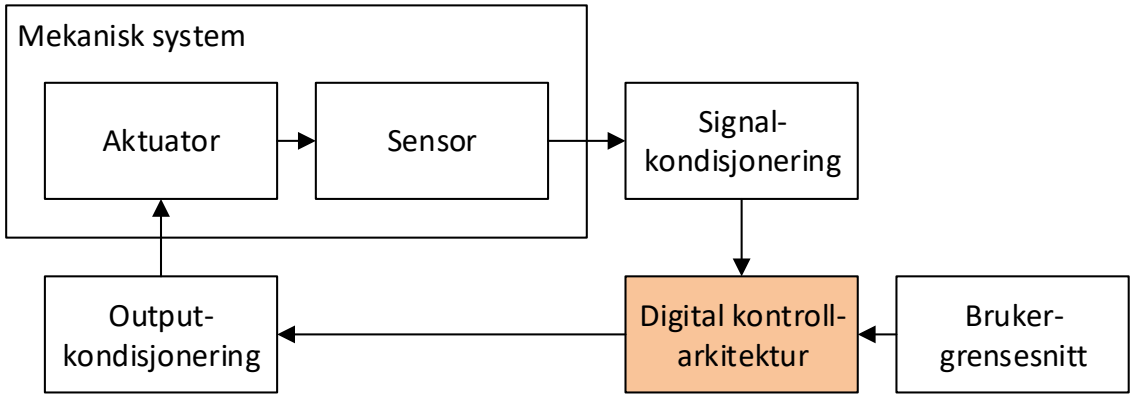
Lab

- Ikke lab pga koronavirus => jobb med oppgaver!

Forelesning

- Kunne benytte terminalprogram på PC sammen med mikrokontrollere.
- Kjenne til metode for å lage mekatroniske systemer.
  - Fremgangsmåte (step by step)
  - Litt om prosjekthåndtering
-

Systemperspektiv og oversikt



- Rest fra mikrokontrollere
  - Terminalprogrammer
  - Eksempel på mikrokontroller
- Fremgangsmåte i designprosess
  - Bittelitt om prosjekthåndtering

Recap busser

- UART
  - Universell Asynkron Receive/Transmit
  - Tx: Transmit linje
  - Rx: Receive linje
  - I utgangspunktet point to point (1-1 kommunikasjon)
  - Ingen overføring av klokke
  - Hastighet, Databit, paritet må settes på forhånd
- RS485
  - Elektrisk spesifikasjon- differensiell drevet
    - For høy hastighet, osv.
- SPI: 4 wire
  - SCK, MOSI, MISO, SS
  - Master igangsetter alt
  - Kan kjedekobles
  - Kan koble flere til samme buss
    - En Slave Select per Slave
  - Klokke overføres på bussen
  - Egnet for enkle enheter
- I2C : To wire, pullup på både SDA og SCL
  - Kan ha flere mastere, elektrisk trygg
  - Inntil 1008 enheter
  - Adressering
  - Klokkestrekking osv.
  - Egnet for «smarte» enheter

I praksis…

- Som regel holder vi på med brikker der signalprotokollen er ferdig implementert.
- For Arduino har vi ferdigskrevne klasser og HW-implementert UART og I2C osv. holder styr på timingen for oss.
  - => Vi må først og fremst sørge for at koblingene er riktige, og at vi setter opp enhetene riktig før vi begynner å bruke dem.
  - Biblioteket for I2C til arduino heter «Wire»
    - https://www.arduino.cc/en/Reference/Wire
- Kobler vi Arduino opp til en PC via USB porten, emulerer PCen en serieport (COM-port) som vi kan benytte som om det var en normal serieport.
  - DVS: Vi må sette opp baud rate, om vi bruker paritet og antall stoppbit

Terminalprogrammer

- En terminal var opprinnelig en «tynnklient» med tekstskjerm og tastatur, koblet til en server.
- Et terminalvindu åpner ikke nødvendigvis en kommandotolk, men den *kan* brukes til det.
- Når vi har koblet opp en enhet til PC via serieport (fysisk eller emulert), kan vi finne koblingen i *device manager* under «*ports*»
  - Typisk COMx, der x er et nummer gitt av fysisk tilkobling eller OS ved portemulering
  - Linux: ls /dev/tty*
- Har vi en COMport åpen, kan vi sende og motta data fra den med et terminalprogram.
- KUN ETT program får bruke èn COMport av gangen.
  - NB: Serial Monitor og programmere Arduino
- For å snakke over en COMport, må vi sette opp koblingen slik at sender og mottager benytter samme oppsett (se UART)
- Eksempler på Terminalprogram er:
  - Putty, HyperTerminal, Coolterm, Arduino Serial Monitor, TeraTerm, Screen og så videre…

https://learn.sparkfun.com/tutorials/terminal-basics/all

Eksempel mikrokontroller: Arduino

- Open source mikrokontrollerkort og åpent rammeverk for å lage kode
- Bruker «Arduino Programming language» som i praksis er C
  - NB: Man kan aksessere alle registre i arduino i koden, på måter som ødelegger funksjonen til rammeverket. Hvis går inn i spesifikasjonen til mikrokontrolleren og bruker registre derfra ukritisk, kan det hende arduino-funksjonene ikke vil virke.
- All kode inneholder
  - setup()
    - Her fyller man inn ting som skal kjøres initielt.
      - Oppsett av IO pinner, ol.
  - loop()
    - Her er hoveddelen av programmet som kjører på repeat til man skrur av

ARDUINO Leonardo: Tech specs

- Merk:
  - Vi har en begrensning i hvor mye strøm utgangene kan levere. Prøver vi å trekke mer strøm enn det, vil ikke Leonardo klare å levere en utspenning som er høy nok (og vi risikerer skade på kortet!)
- Sjekk alltid at utgangen er i stand til å drive det du setter som mottager…

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage (Recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega32u4) of which 4 KB used by bootloader
SRAM	2.5 KB (ATmega32u4)
EEPROM	1 KB (ATmega32u4)
Clock Speed	16 MHz
Lenght	68.6 mm
Width	53.3 mm
Weight	20 g

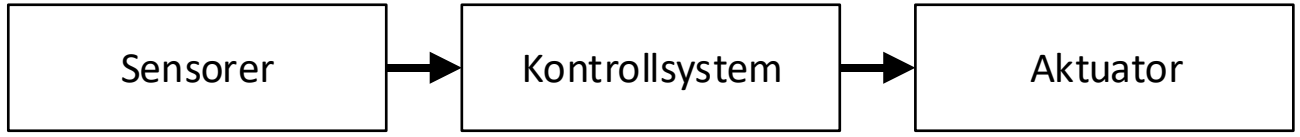
Arduino Demo

- File->Examples->Communication->SerialCallResponseASCII
- Sjekk at riktig kort er valgt (Tools->Port...)
- Get Board Info (Viser at vi har kontakt med kortet)
- Laste opp "=>"
- Tools -> Serial Monitor
- Putty-Session->Serial + velg port og hastighet

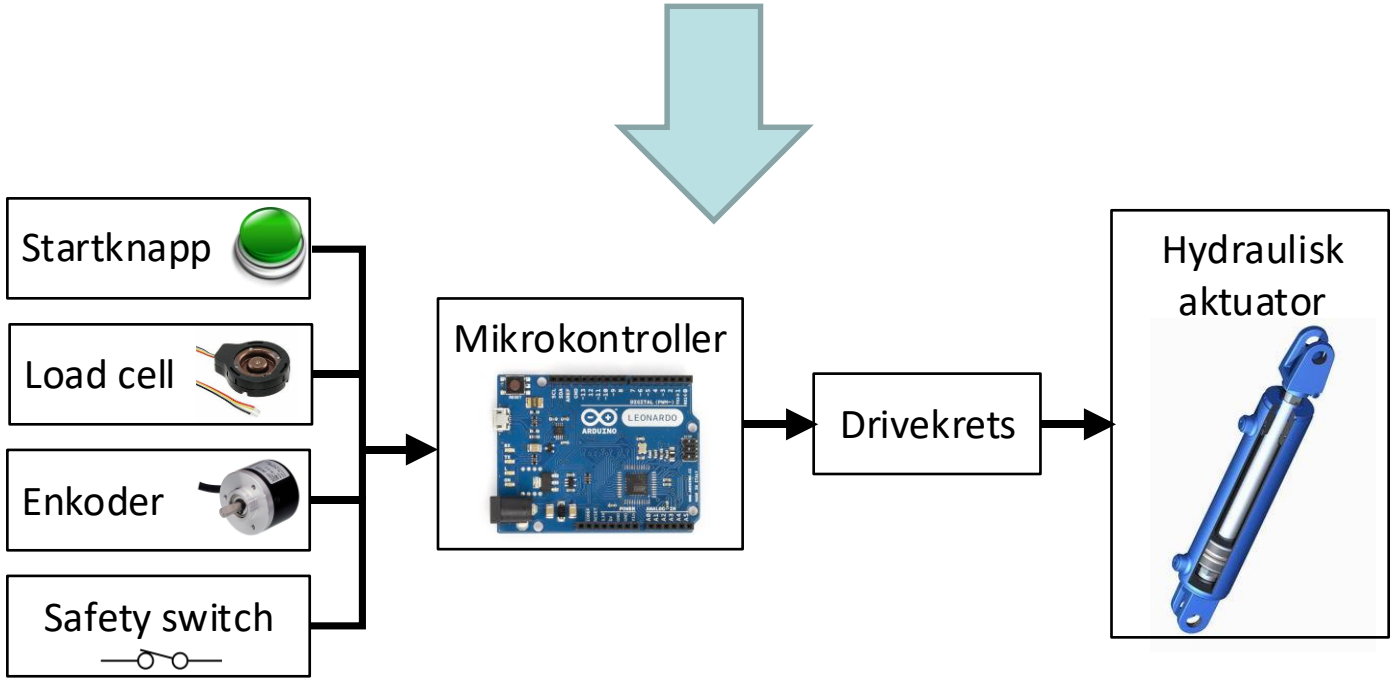
Metode for å designe mikrokontrollerbaserte systemer *Eksempel*

- `1: Definer problemet (kravspekk)`
  - EKS *jeg ønsker å lage et system for styre et et lasteplan.*
  - Raffinér problemet...
    - Etter å ha løsnet bakluka kan jeg
      - trykke startknappen => Lasteplanet heves til
        - » Enten: Lasteplanet er hevet 45 grader
        - » Eller: Jeg trykker på startknappen igjen
        - » Eller: Vekten av det som er på lasteplanet er 0.
      - Trykker jeg startknappen igjen skal lasteplanet gå ned til
        - » Lasteplanet er i vater (0 grader) eller
        - » Jeg trykker startknappen igjen

Forts.Mikro.Eks.:

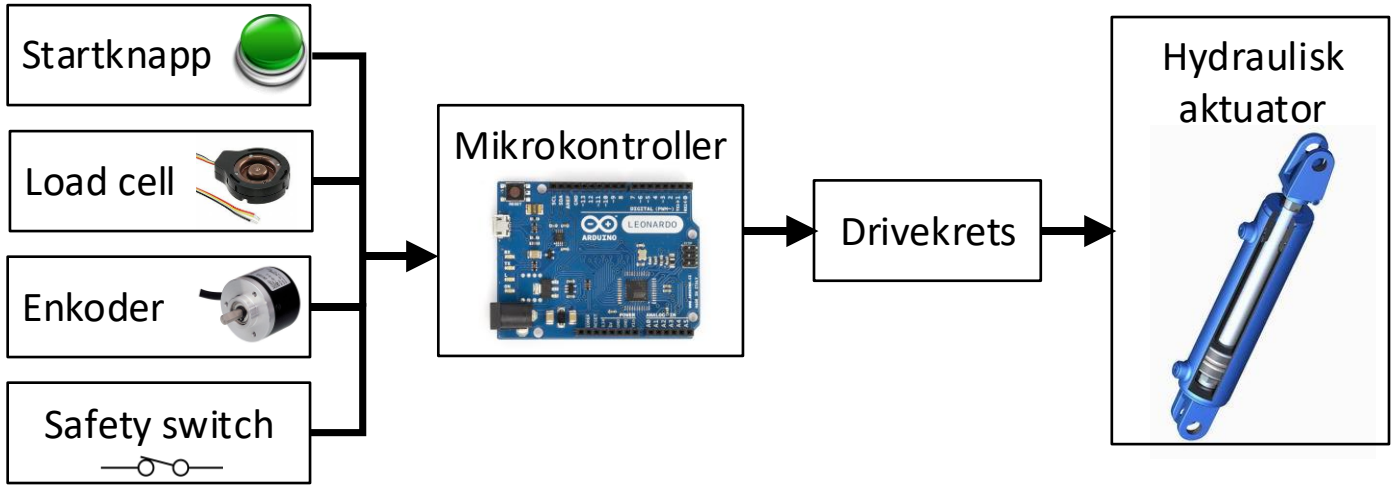


- 2: **Tegn et funksjonsdiagram**
  - Bokser med tekst, symboler eller bilder
  - Skisserer hvordan enhetene er koblet sammen (ikke detaljer)



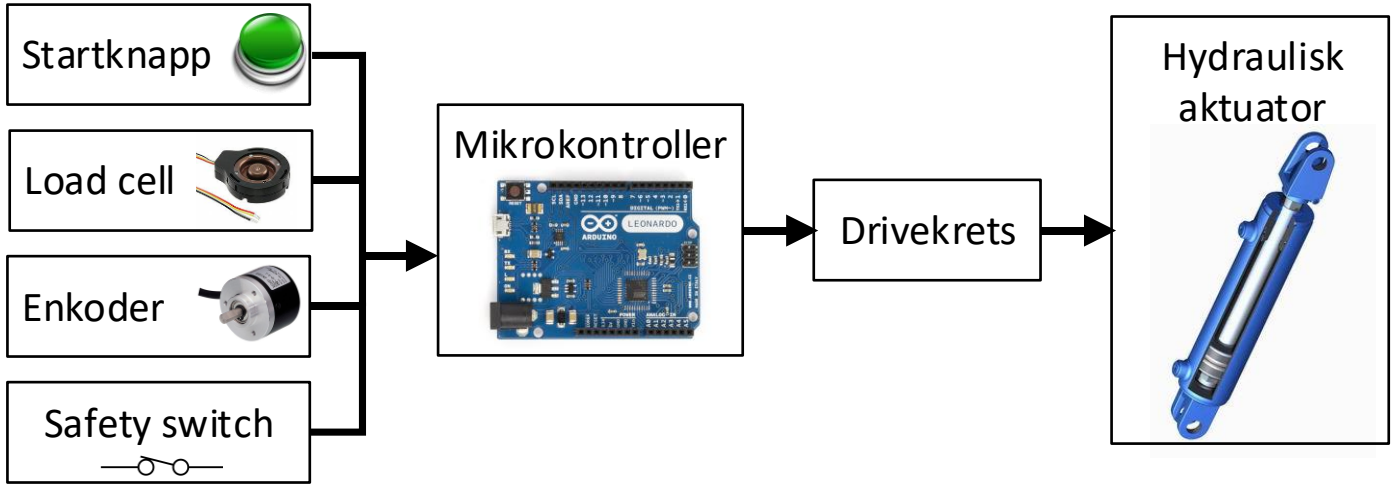
Forts.Mikro.Eks.:

- 3: **Identifiser IO krav** (herunder busser)
  - **Input**
    - Startknapp
      - 1 pinne, digital
    - Loadcell- analog
      - 1 eller 2 pinner, analog
    - Enkoder- analog eller digital
      - 1 analog input eller
      - 2 eller flere digitale input pinner
    - Sikkerhetsbryter
      - 1 pinne, digital
  - **Output**
    - 2 – 4 pinner, - *kommer an på drivekretsen...*
  - **Busser** (Både Input og Output)
    - Digitale sensorer kommer ofte med bestemte bussgrensesnitt (SPI, I2C, UART)



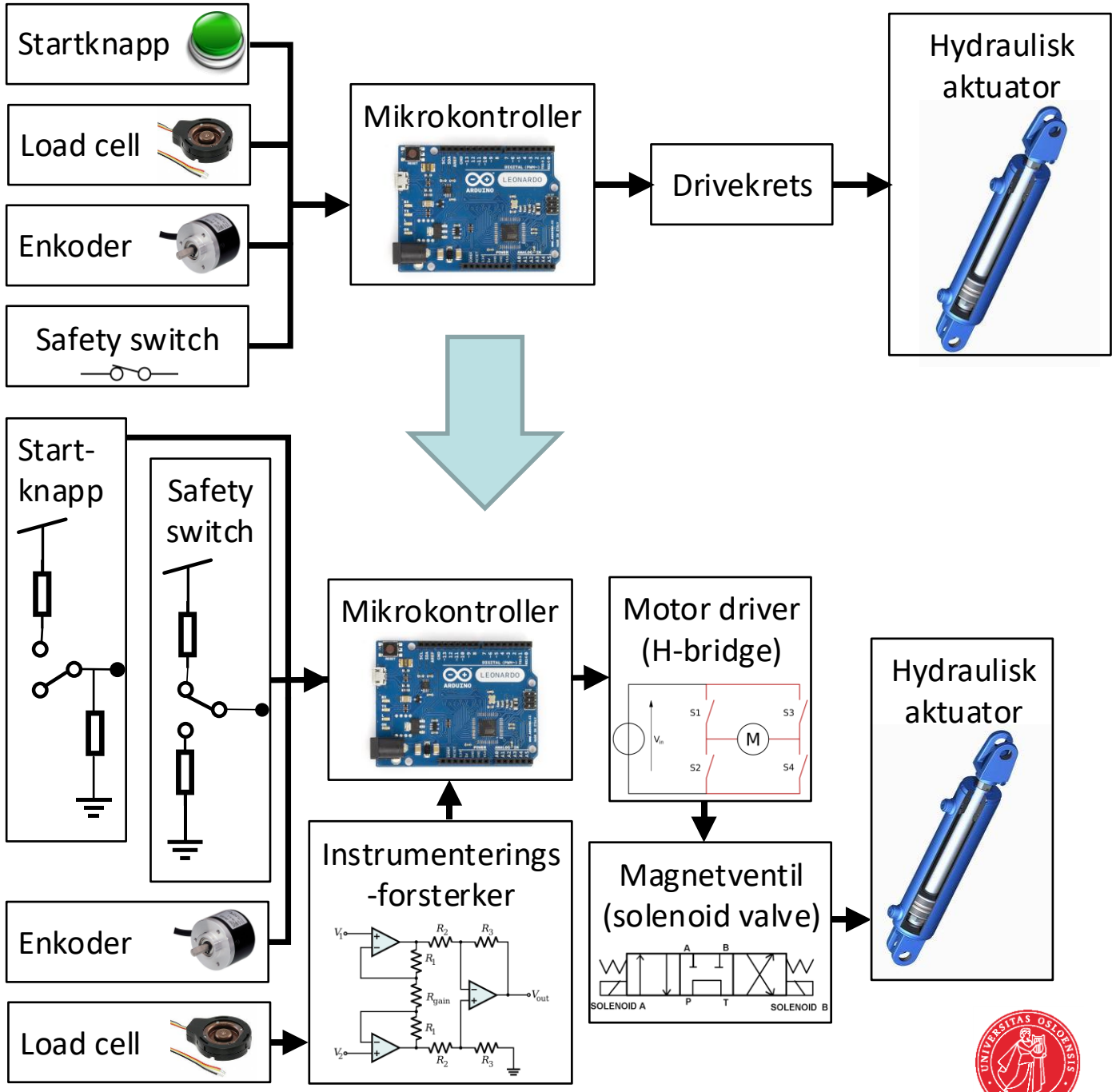
Forts.Mikro.Eks.:

- 4: **Velg mikrokontroller**
  - Nok IO
    - Digitale
    - Analoge
    - Busser
  - Nok minne
  - Rask nok
  - *Skal produktet i salg på...*
    - Massemarked
      - Minste som kan fylle krav
    - Begrenset marked/ forventet endringer
      - Det som gir deg raskest resultater
    - Kun prototype / forskning
      - Det som gir deg raskest resultater og nok handlingsrom.
      - Evt noe overspekket ifht senere produksjonsplaner.



Forts.Mikro.Eks.:

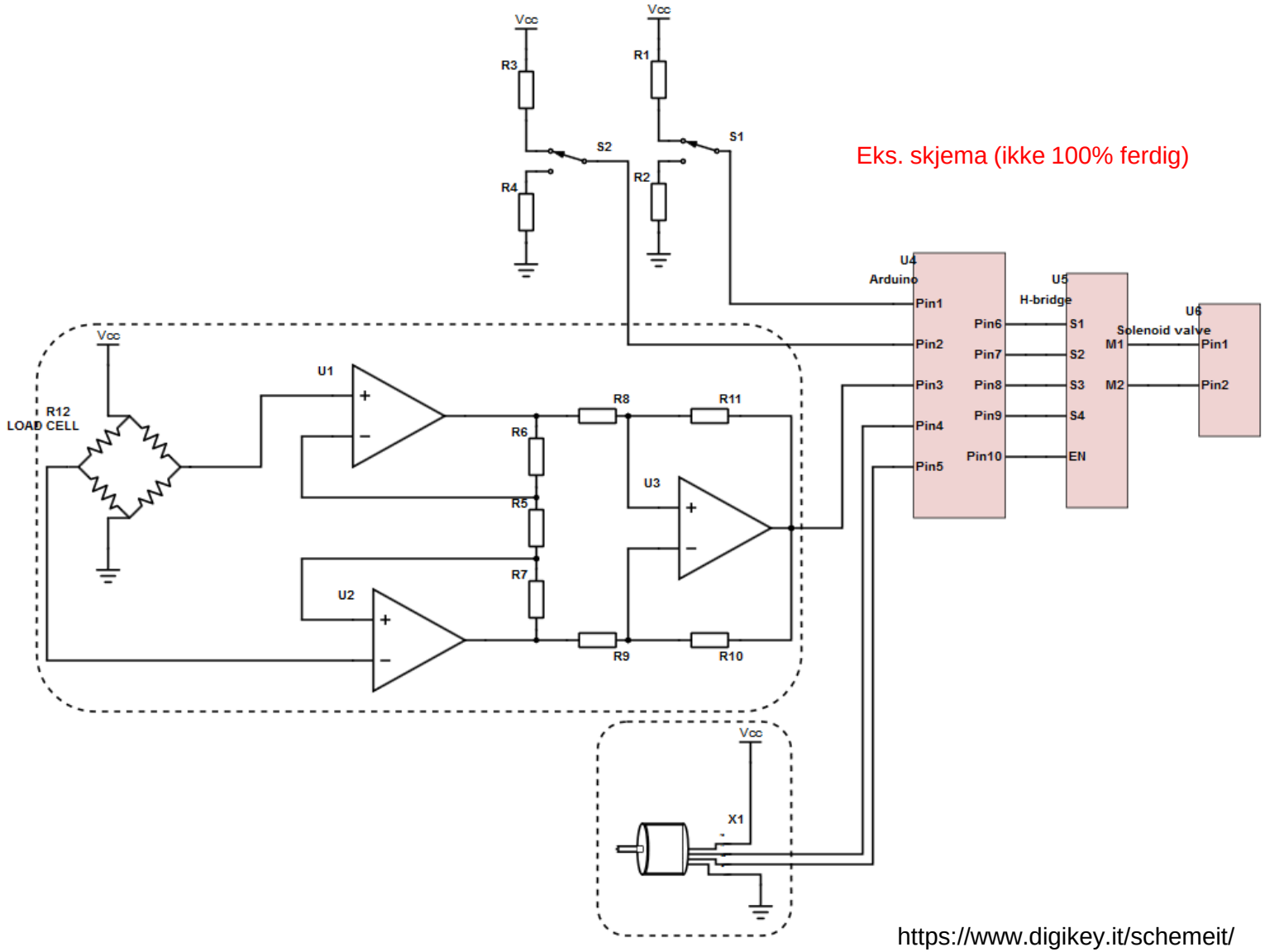
- 5: **Identifiser nødvendige interface kretser**
  - Drivekapasitet til output
  - Forsterkningsbehov for analoge input
  - Pullup/-down for brytere ol.
  - Buffere / 3V-5V konvertere,..
  - (Dekodere)



Forts.Mikro.Eks.:

- 6: **Bestem programspråk**
  - Arduino Wiring
  - C
  - Python
  - Assembler
  - HDL

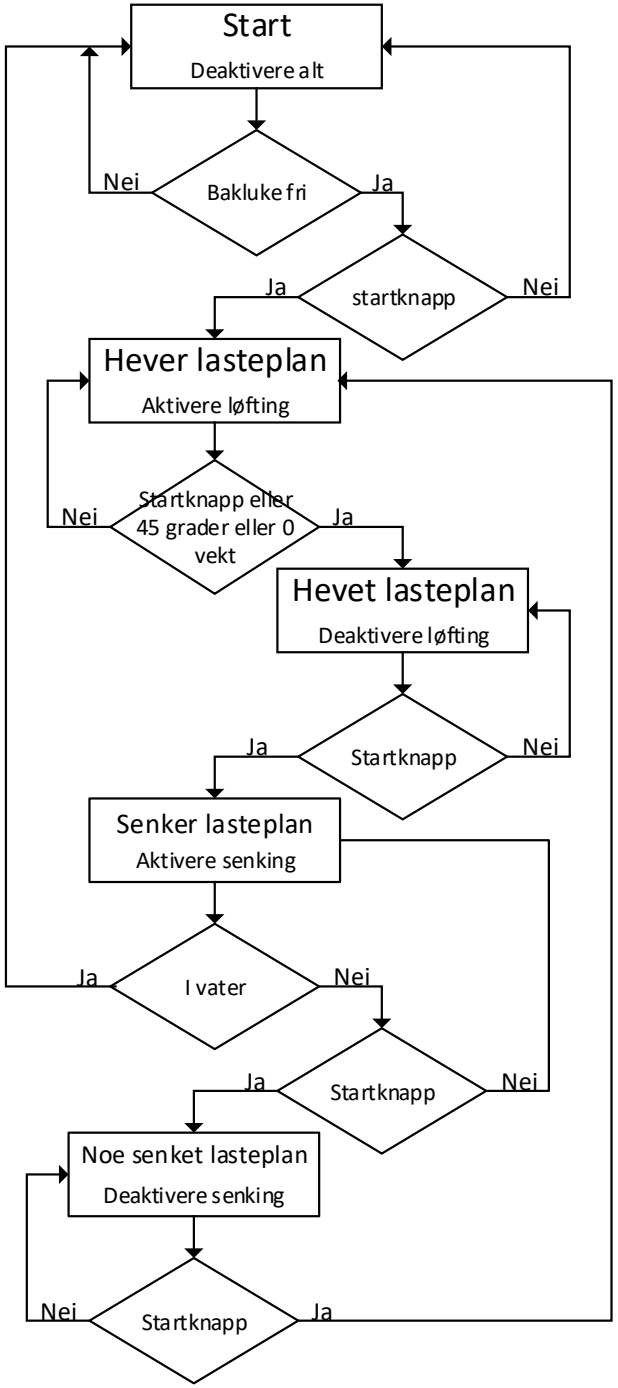
- 7: Tegn detaljert skjema
  - Ta med alt du skal koble til selv.
  - Detaljér. Ikke la pinner du skal bruke henge i lufta.



https://www.digikey.it/schemeit/

Forts.Mikro.Eks.:

- 8: **Tegn flytskjema**
  - Her: ASM diagram har bestemte regler for bruk av boksene (tema→IN3160)
    - ingen loop tilbake til decision box.
    - Tilstandsbokser har kun én utgang.
- 9: **Skriv** (*og simuler*) **koden**
  - Det er lov å modifisere eksempelkode men øvelsen blir bedre om man gjør selv!
- 10: **Bygg og test** systemet!



Begreper: Krav og spesifikasjon

- Krav, kravspesifikasjon (Requirements)
  - Forteller hva et system må kunne utføre
- Spesifikasjon, designspesifikasjon (Specification)
  - Forteller hvordan man skal oppnå kravspesifikasjon

Metode oppsummering

1. Definér problemet
2. Tegn funksjonsdiagram
3. Identifiser IO krav
4. Velg mikrokontroller
5. Identifiser interface kretser
6. Bestem programspråk
7. Tegn detaljert koblingsskjema
8. Tegn flytskjema for software
9. Skriv og simuler koden
10. Bygg og test systemet

- Regn med å gå tilbake i prosessen
  - reiterere
  - lage flere prototyper osv.

- Hvilke punkter danner..
  - Kravspekk..?
  - Designspesifikasjon..?

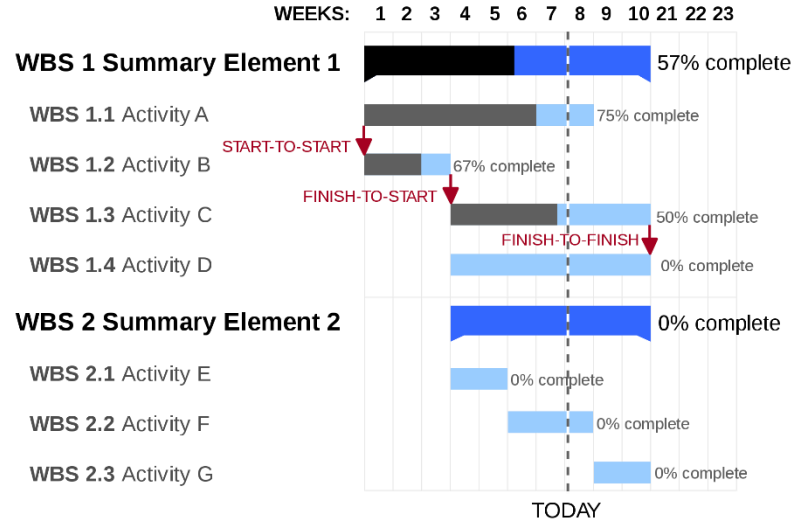
Prosjektmanagement

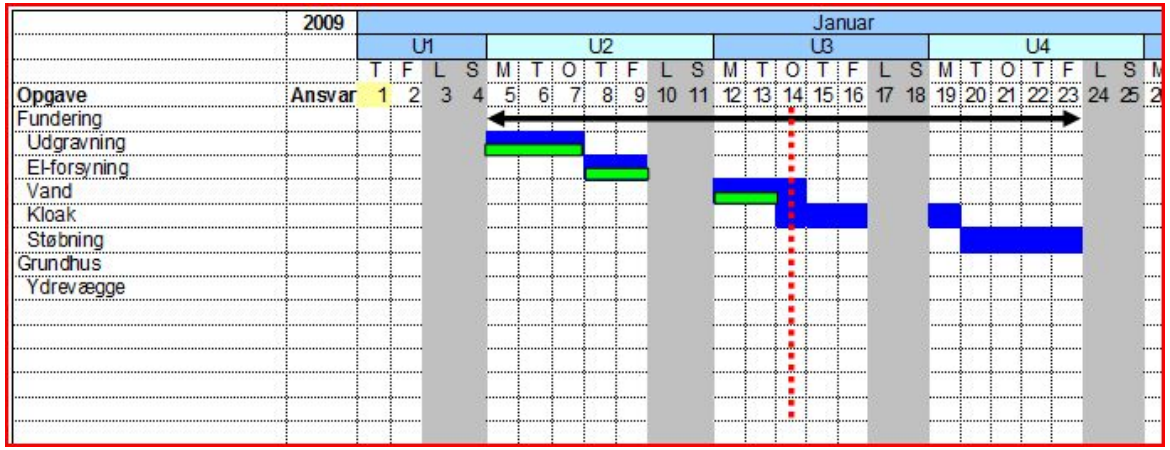
- Holde styr på
  - Oppgaver underoppgaver
  - Ressursbruk (timebruk, mennesker, innkjøp)
  - Tidsplan, milepæler (milestones)
  - Prosesser

https://en.wikipedia.org/wiki/Gantt_chart

Gantt diagram

- Tid- og prosessplan
  eventuelt ressurs-tid-plan
  - Rekkefølge
    - Tidslinje
    - Milepæler
  - Viser avhengigheter
    - Hva som er på kritisk linje (Critical path)
- Kun et verktøy for å få oversikt.
  - Ingen garanti for at det er mulig å følge opp





Wikipedia, Finn Svenning

Anbefalt lesing og oppgaver

- Lese
  - COK: 30.1-30.6
    - Fokuser på å fange opp ideene/ begrepene.
- Oppgaver
  - 30.12

  - Eksamensoppgaver tilgjengelig i canvas

Fremover...

- Øve til eksamen- eksamensoppgaver (ikke glem oblig 5)
- Neste gang: gjennomgang av deler av en eksamen og eller repetisjon?
- Repetisjon ønsker?
  - Opamper / Instrumentering?
  - Motorer?