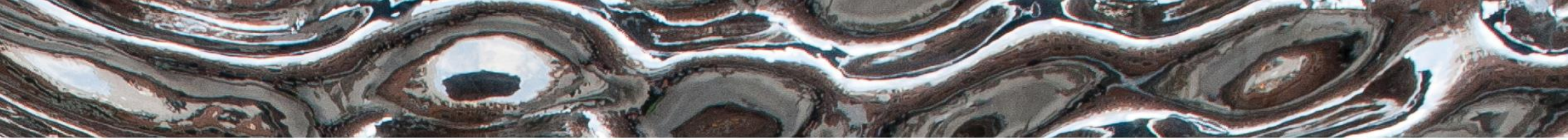


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

IN 1080

Busser og fremgangsmåte for mekatronikkprosjekter

Yngve Hafting, 2020

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

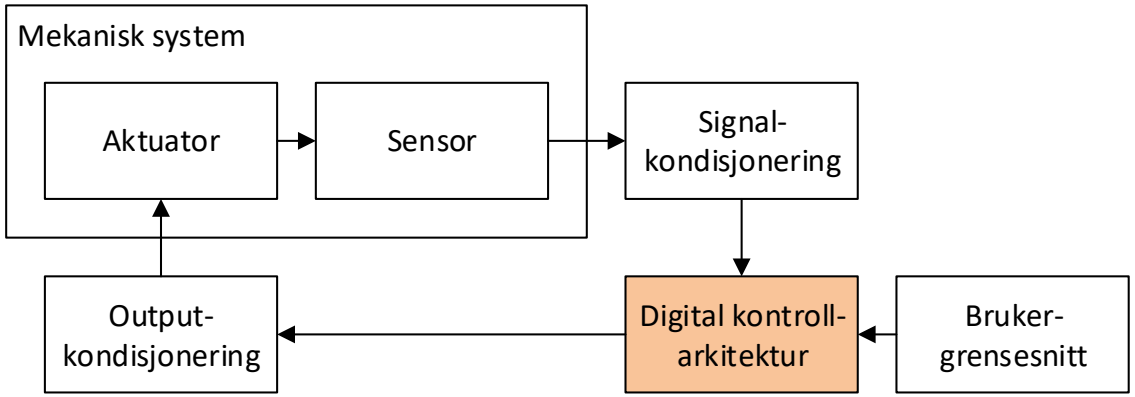
Lab

- Ikke lab pga koronavirus => jobb med oppgaver!

Forelesning

- Kunne velge sensorsystemer for mekatronikk
  - Kunne redegjøre for virkemåte til noen busser som brukes til å overføre data mellom komponenter brukt i mekatroniske systemer.
- Kunne benytte busser til kommunikasjon mellom mikrokontroller og chipbaserte sensorer
- Kunne benytte terminalprogram på PC sammen med mikrokontrollere.
- Kjenne til metode for å lage mekatroniske systemer.
-

Systemperspektiv og oversikt



- Innebygde systemer
- Mikrokontrollere
- Busser
  - RS232
  - SPI
  - I2C
  - RS 485
- Terminalprogrammer
- Fremgangsmåte for å lage kontrollarkitektur
- Eksempel på mikrokontroller

Innebygde systemer «Embedded systems»

- = Prosessor/minne + IO-periferienheter
  - Ting- biler, båter, batterisystemer, symaskiner, vaskemaskiner, osv.
  - IO-enhetene styrer andre elektriske eller mekaniske (mekatronikk) systemer
  - Prosessorsystem kan være
    - Mikrokontrollere
      - Gjerne uten OS
    - FPGAer med tilstandsmaskiner (IN3160)
    - DSPer (Digital signal processor)
    - Mikroprosessorer (IN2060)
      - ARM
      - IntelCPU?
- Kjøres ofte uten operativsystem
  - Kan ha spesiallagde real-time OS.
  - Sanntid/ «Real time» og deterministisk utfall
    - Reaksjoner på miljøet må skje fort nok
    - Reaksjoner må skje fort nok
      - Bakgrunnsrutiner = farlig

Innebygde systemer fortsetter:

- Periferienheter (peripheral devices)
  - Gjør jobben som systemet er tiltenkt
    - Skjermer, lysdioder (LEDs), drivekretser + aktuatorer, trådløs kommunikasjon...
  - Digitale
    - Mange ferdigdefinerte busser og protokoller
      - USB, SPI, I2C, UART, m.fl.
  - Analoge
    - Signaler må kondisjoneres og konverteres (AD-konvertering).
    - DA-konvertering for output
      - PWM / drivesignaler
  - Brytere ol.

Mikrokontroller og mikroprosessor

- Mikroprosessor
  - = Kun prosesseringsenhet (Trenger perifert minne, osv)
  - «Det du finner i en PC»
  - Kjører typisk et OS
  - Mer komplekse arkitekturer (IN2060)
    - flerstegs pipeline
    - Cache
- Mikrokontroller
  - «Mer alt i ett enhet»
    - Enklere/ mindre enheter
  - Ofte single-cycle prosessering
  - Inneholder gjerne minne (Flash)
  - Glidende overgang mot mikroprosessorer ifht kompleksitet
    - Eks: ARM – kjøres både med og uten OS.

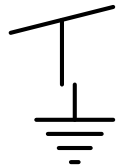
Digital kommunikasjon med mikrokontrollere

- Vi skal fordype oss i noen mye brukte varianter..

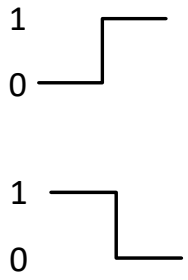
Noen begreper / notasjon

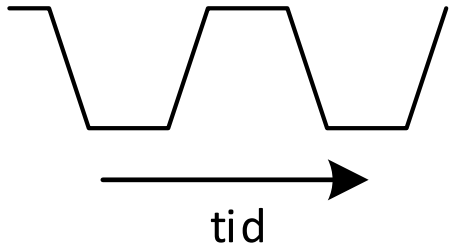
- Typiske representasjon digitale signaler

høy	true	1	VDD	VCC	3V	5V
lav	false	0	GND	VSS	0V	0V



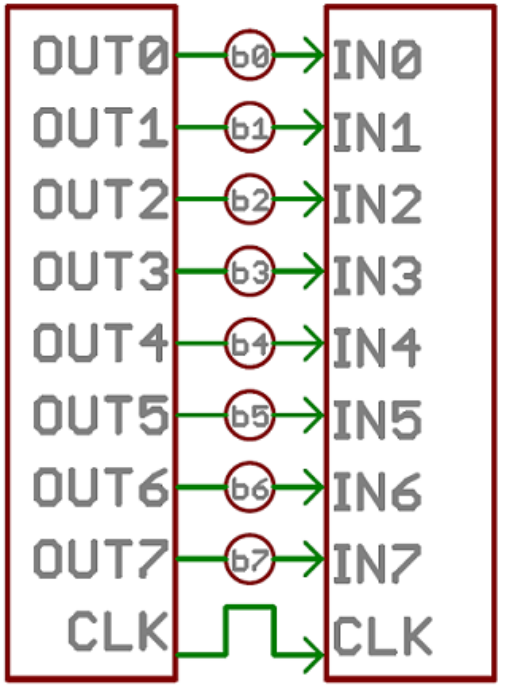
- Stigende flanke
  - transisjon fra lav til høy
- Fallende flanke
  - transisjon fra høy til lav.



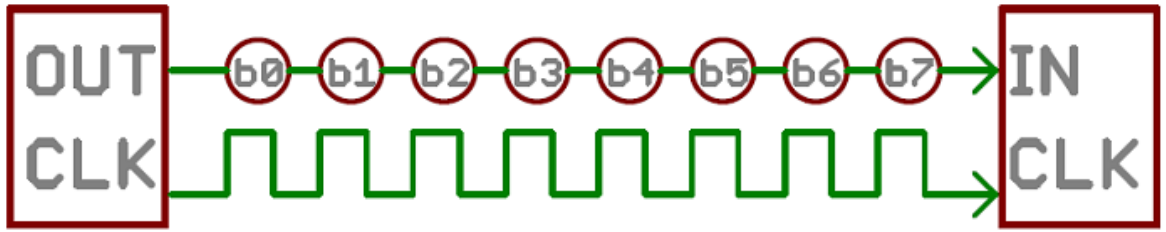


Bussterminologi

- En buss er en kobling der to eller flere komponenter kan kommunisere med hverandre.
- Seriell vs parallell buss
- Simplex (En retning)
- Full duplex: Begge retninger samtidig
- Halv-duplex: Begge retninger, men kun en av gangen
- Rx = Recieve, Tx = transmit (for RS232)



Seriell buss ↓ ,Parallell buss ↑



Bestemmer	Adlyder	Eksempel-buss
Host	Device	USB
Master	Slave	SPI, I2C
Data Terminal Equipment (DTE)	data communications equipment* (DCE)	RS-232

Eksempler Busstyper

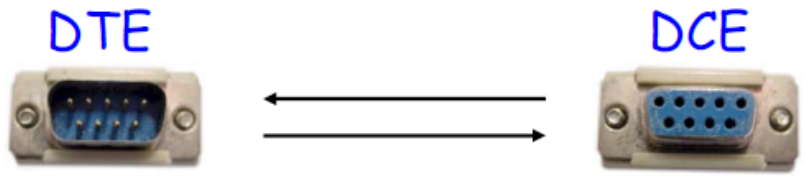
	Komponent til komponent	Kort til kort* (korte avstander)	Enhet til enhet*
Seriell	RS-232, SPI, I2C	RS232, SATA (harddisk)	RS485, USB1 og 2, Ethernet
Flere Serielle		SCSI, PCI-e	Displayport, HDMI, Thunderbolt, USB 3.x
Parallell	Minnebuss	(Parallellport), (PCI), Minnebuss	

*inndelingen etter avstand kan variere

- Utvikling: fra parallellkoblinger (tykke ledningsbunter) til mer seriell kommunikasjon
  - Overføring av mange bit samtidig = trøbbel ved høye hastigheter og store avstander
  - Differensiell seriell kommunikasjon tillater høyere hastighet på dataoverføring per linje.
    - Eks: USB har D+ og D- i tillegg til VDD og GND. D+ og D- er alltid motsatt ved dataoverføring.
- Parallelle busser brukes typisk internt i komponenter (chiper), og mellom minne og prosessor
  - Minnebuss = Data buss og Adressebuss

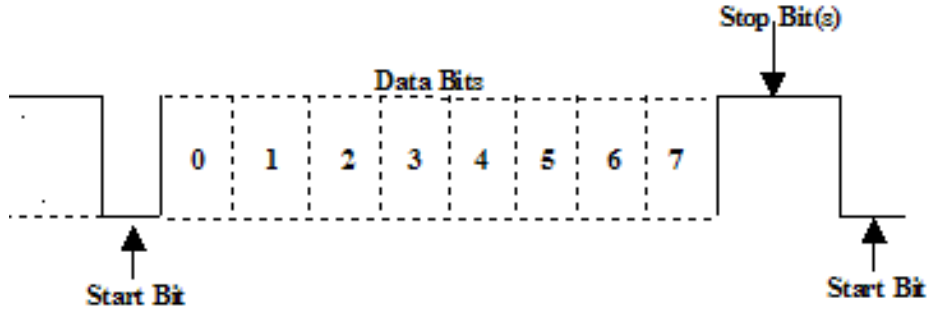
"RS-232", Asynkron Seriell kommunikasjon

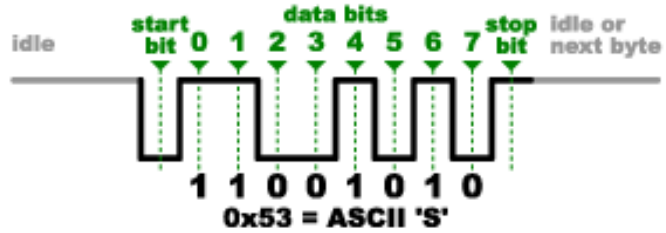
- Toveis seriell datakommunikasjon.
- Laget for kommunikasjon mellom 2 enheter (PC/terminal og modem)
- Kan kjøres halv duplex eller full duplex
  - Full duplex krever flere ledninger
- Mulighet for en rekke ekstra ledninger for dataflytkontroll (handshake/ status)
  - i prinsippet brukes bare to pinner til kommunikasjon, Rx og Tx
- opprinnelig laget på 60 tallet beregnet for kommunikasjon mellom datamaskin og modem ved +-15V…
- I dag benyttes det typisk mellom kretskort på 5V eller 3,3V
  (*TTL – transistor-transistor level*)
- Hastighet < ca 200kbps



UART protokoll

- For å overføre data med RS-232 brukes gjerne UART
  («Universal asynchronous receiver-transmitter»)
  - UART står for overføringen av data og presenterer dem parallelt i mikrokontrolleren.
  - Normalt er UART innebygd i mikrokontrollere.
- Klokkedata overføres ikke (derav asynkron).
- «baud rate» vs «bit rate»
  - Baud = symbolrate (multinivå logikk)
  - Bestemmes i forkant!
    - Typisk 9600 -115200 bps
- Data sendes i pakker som også *må defineres i forkant.*
  - Alle pakker inneholder start og stoppbit.
    - Det kan også være 2 stoppbit
  - Data kan være fra 5 til 9 bit
  - Paritetsbit er lite brukt, men kan hjelpe dersom det er mye støy
    - Typisk 1 dersom tversummen av bits er odd, 0 ved partall.
- Generelt sendes LSB før MSB, så LSB er bit 0 i data
- TX (Transmit) fra en enhet kobles til RX (Receive) i den andre
- *USART = UART med tillegg for synkronisering*

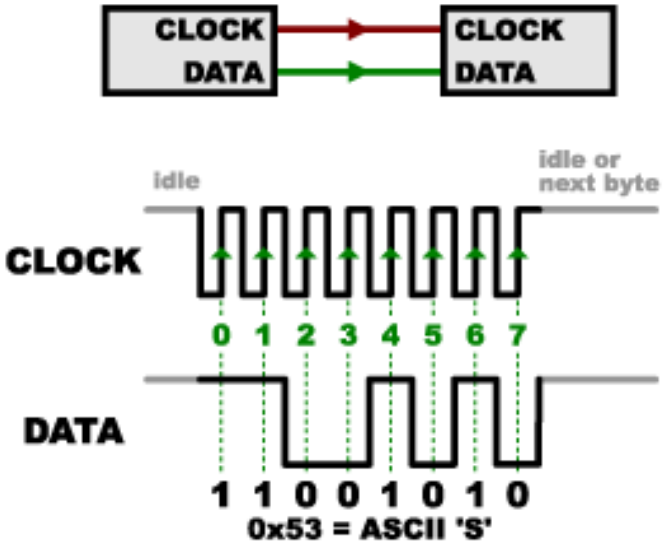




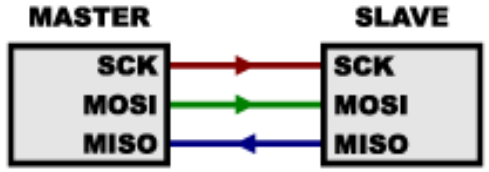
Mer/kilder:
https://learn.sparkfun.com/tutorials/serial-communication/all
https://learn.sparkfun.com/tutorials/terminal-basics/all

SPI – Serial Peripherial interface

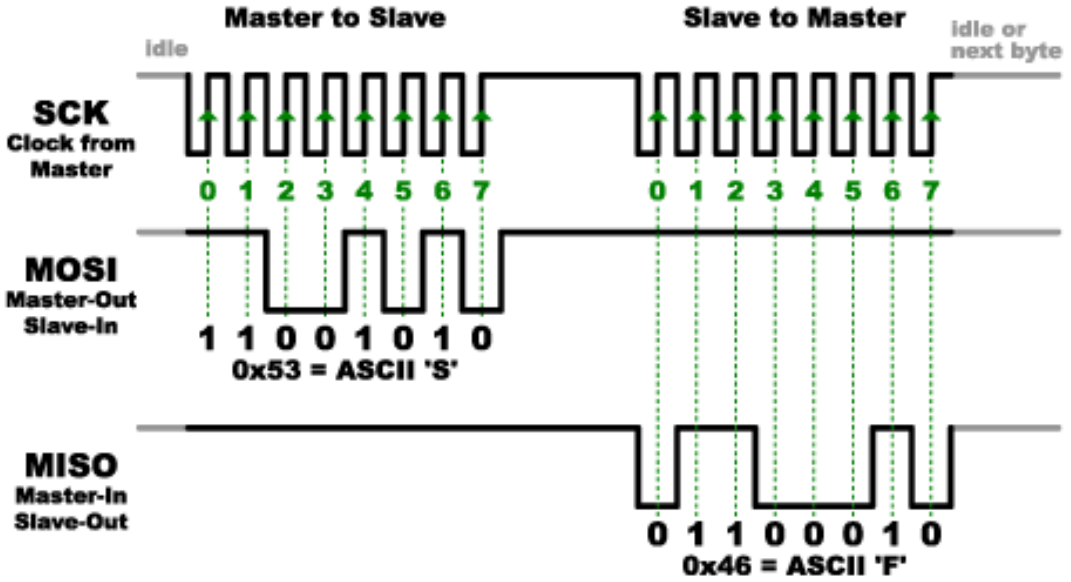
- Synkron, full duplex seriell dataoverføring
- En Master, en eller flere slaver.
- Klokke overføres via egen linje
  - slavene behøver ikke klokke
- Enklere enn RS-232.
  - Slave trenger omtrent bare et shift-register for å ta imot data
- Hastighet kbps- Mbps
- Terminologi:
  - SCK – Serial Clock (fra Master)
  - MOSI – Master Out, Slave In
  - MISO – Master In, Slave Out
  - SS – Slave Select

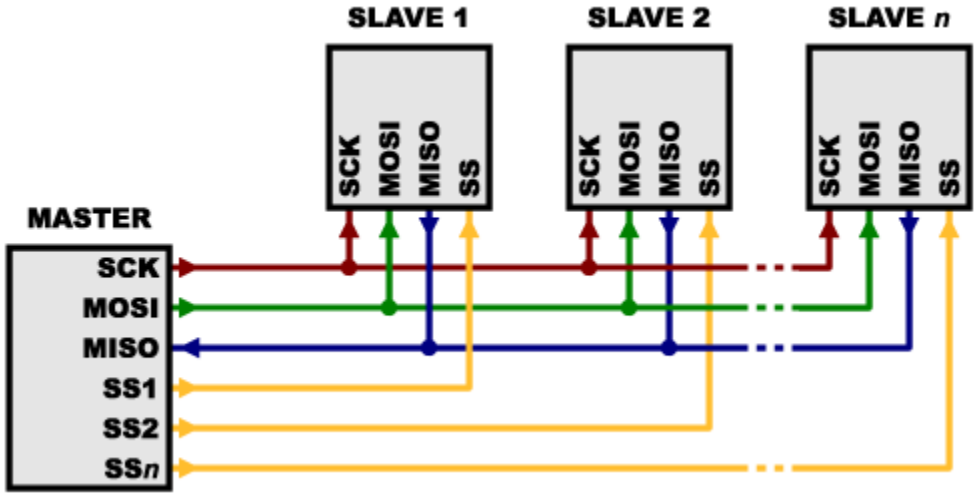


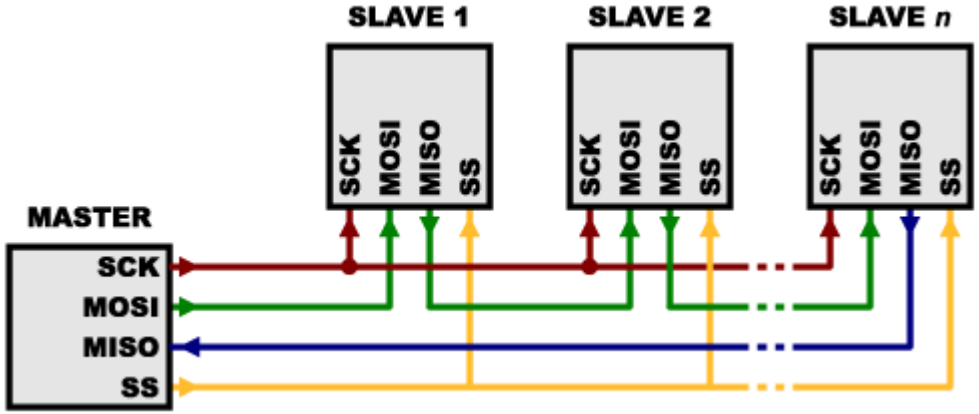
SPI- protokoll



- Master som klokker all dataoverføring, og må vite hvor mye data den skal motta fra hver slave.
- Master kan enten ha en slave-select linje for hver slave, eller så må slavene være koblet i en kjedekobling «daisy chain» (se figur)
- Slave select signalet forteller slaven at den må lytte og eller sende data.
  - Slave select er aktivt lavt, og bør ha egen pullup-motstand, for å unngå konflikt på bussen.

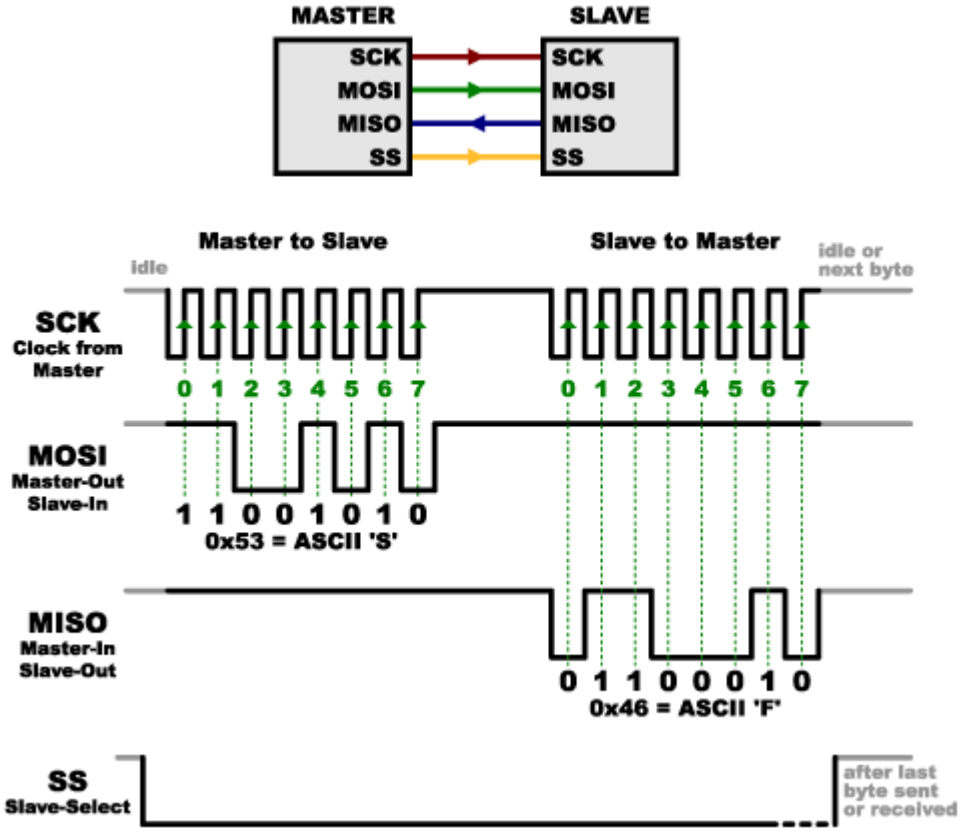


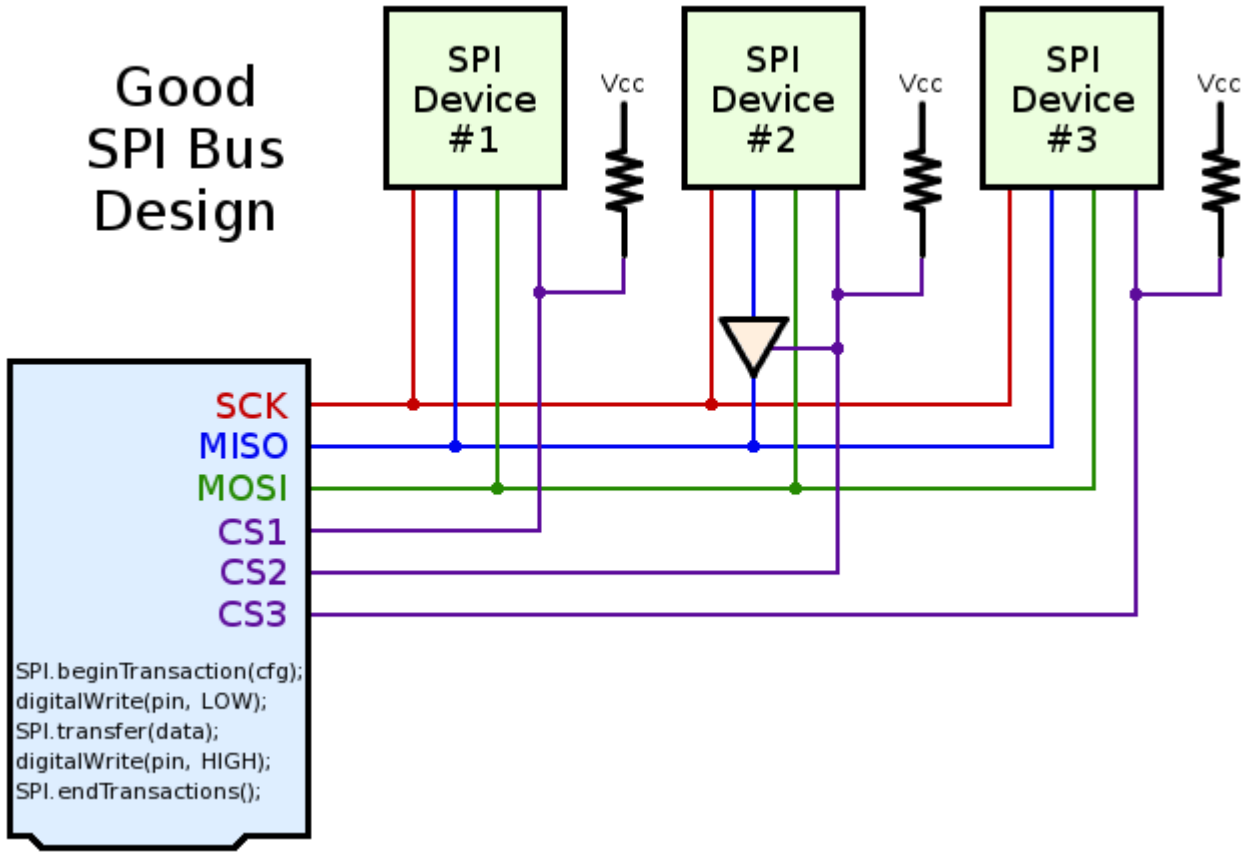




SPI kjedekobling: Merk kobling MOSI-MISO

SPI forts.

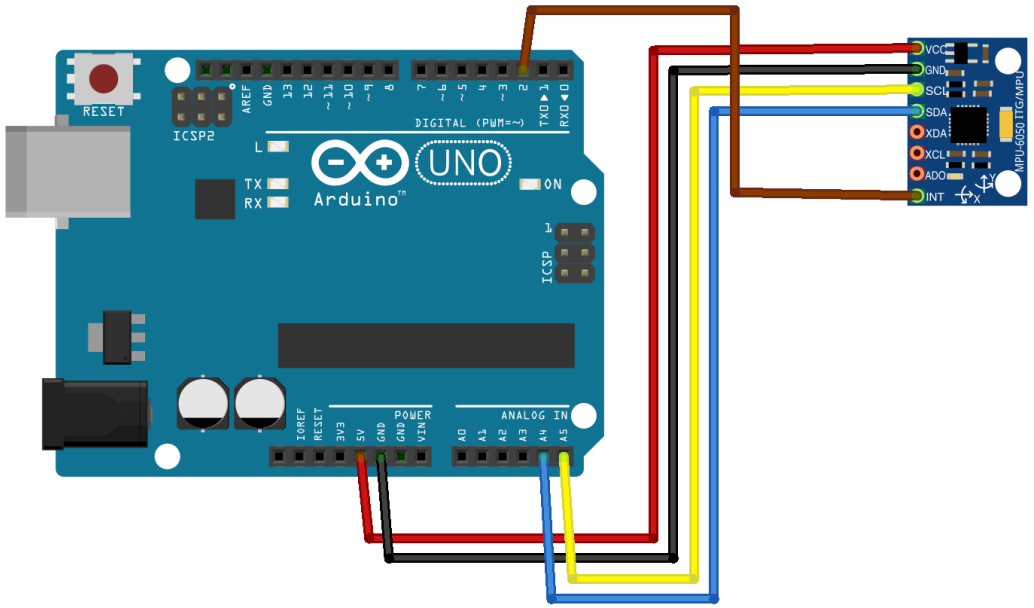




Videre lesing:
https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi
https://www.dorkbotpdx.org/blog/paul/better_spi_bus_design_in_3_steps

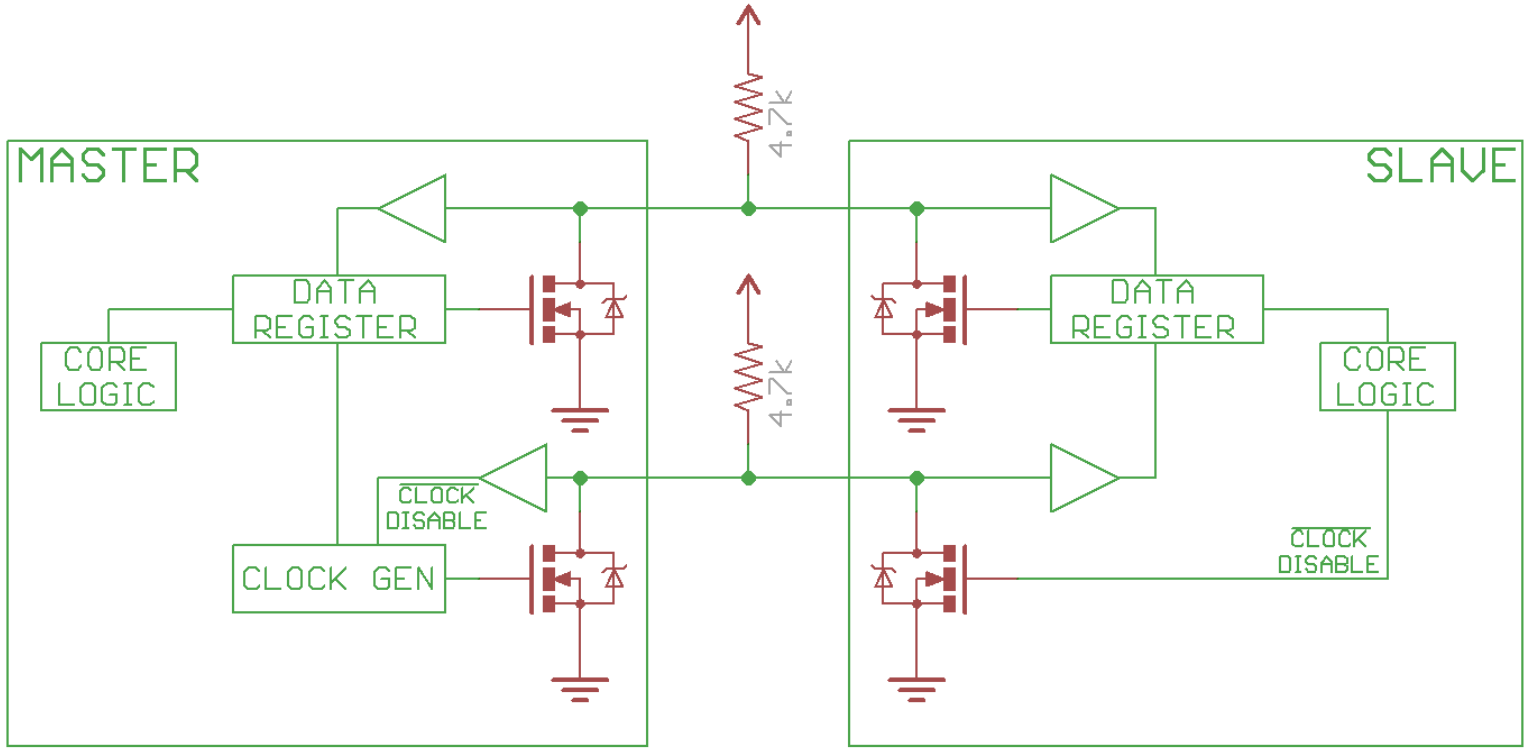
I²C «Inter-Intergrated Circuit Protocol.»

- I2C Bruker kun 2 linjer: SCL og SDA
- Kan ha svært mange enheter koblet på samme buss-
  - inntil 1008 slaver.
  - Flere enheter kan ha rollen som master
- Klokkehastighet på inntil 100kHz eller 400kHz
  - Raskere enn RS232,
  - langsommere enn SPI
- Mer kompleks logikk kreves sammenlignet med RS-232 og SPI.
- Kan fungere opp til 2-3 m lengde
- Kun halv duplex



I²C - Oppsett

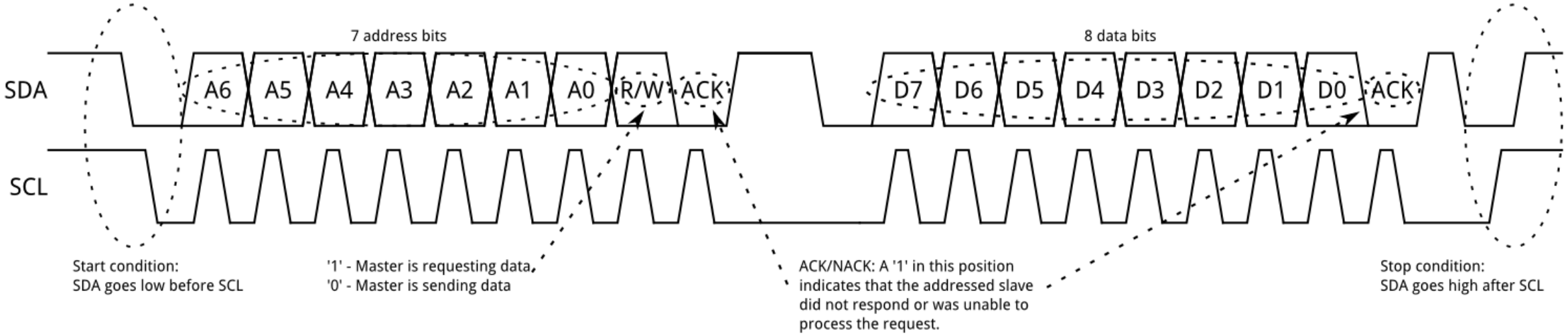
- Både SCL og SDA er open drain (open collector for BJT).
  - DVS enhetene kan bare trekke linjene til jord, aldri sende ut et høyt signal.
  - Krever pullup for å virke
    - Normalt 4,7kΩ
      - Ved flere enheter kan det kreves lavere verdier
    - Dersom man benytter forskjellige forsyningsspenninger kobler man mot den laveste (forutsatt at det er høyt nok for begge)…



I2C protokoll

- I utgangspunktet må Master trekke SDA lav for å få kontroll på bussen. Den som trekker bussen lav først får den (!)
  - SCL høy og SDA lav gjør at de andre enhetene lytter
- Videre følger en pakke med en 7 bits adresse, fulgt av et bit som forteller om Master vil ha data fra eller skrive data til enheten den adresserer.
  - Master klokker SCL, data leses på stigende flanke
- På den 9. klokkeflanken, så må enheten som ble addresert trekke SDA lav (ACK), ellers er ikke dataene å regne for lest (NACK).

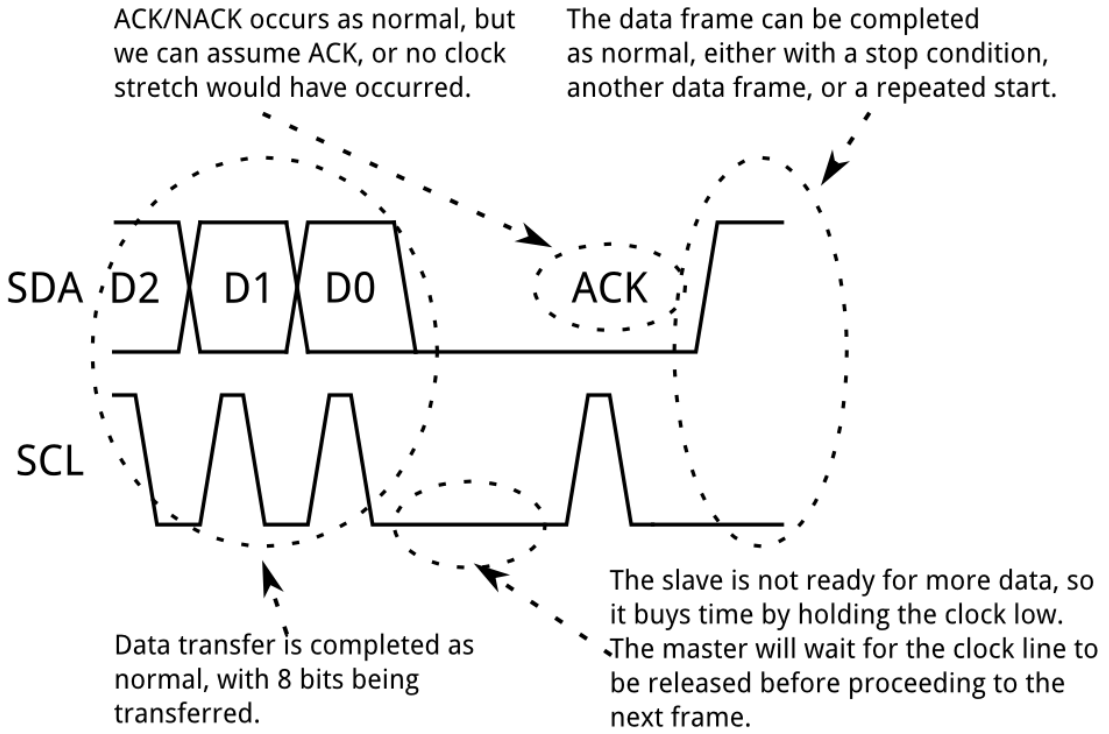
- Etter at adressedataene er sendt og akseptert, fortsetter Master å klokke SCL-linjen.
  - Avhengig av om det var en lese eller skriveoperasjon Master initierte, så vil enten Slave eller Master presentere data
- Data sendes i pakke på 8 bit av gangen, fulgt av en ny «ACK»
- Dataoverføringen fortsetter til Master sier «stopp» ved å sette SDA høy etter SCL går høy.
  - Utenom «stopp» endres aldri SDA etter SCL er gått høy for å unngå falske stoppsignal



I2C: 10 bit adresser og Strekking av klokke.

- I2C kan utvides til 10 bits adressering ved å starte med en adresse som er reservert (b11110), fulgt av de tre første (minst signifikante) bitene av 10 bits adressen.
  - Dette signaliserer at det kommer utvidede adressedata i neste pakke fra Master
- Alle enheter med korresponderende 10 bits adresse vil svare med ACK på den første pakken, mens bare den riktige vil svare etter den andre.
- Denne type adressering fungerer med 7 bits-enheter fordi ingen 7 bits adresse har lov til å ha de fire minst signifikante bit som 1 og det 5. som 0

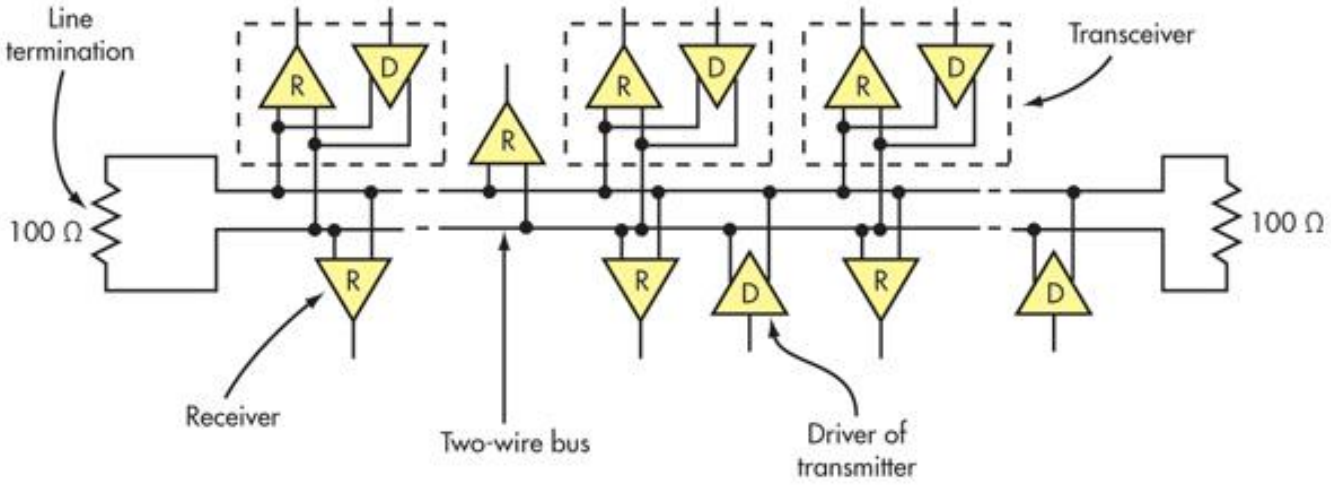
- I noen tilfeller kan Master be om data fortere enn slaven kan levere. Da kan slaven holde SCL lav slik at ACK signalet ikke klokkes. («Clock stretching»)



For detaljer, se
https://learn.sparkfun.com/tutorials/i2c

RS-485

- RS-485 er en fysisk standard laget for kommunikasjon over større avstander enn RS-232.
- Linjene drives differenselt, noe som gir mindre støy.
  - |Vd| >= 200 mV
  - Retningen på strømmen bestemmer om det er 0 eller 1 som sendes.
- Max 32 drivere og 32 mottagere.
- Kan kjøre både full og halv duplex
- Ingen definert protokoll, UART brukes ofte
- Bruker Twisted pair kabel med eller uten skjerming
- Hastigheten avtar med kabellengde
- Til sammenligning:
  - RS422: point-point
  - LVDS: point–point, + hastighet, -lengde
  - M-LVDS : flere på samme bus..



http://www.electronicdesign.com/what-s-difference-between/what-s-difference-between-rs-232-and-rs-485-serial-interfaces

KEY CHARACTERISTICS OF THE RS-232 AND RS-485 SERIAL INTERFACES		
Parameter	RS-232	RS-485
Line configuration	Single-ended	Differential
Mode of operation	Simplex or full duplex	Simplex or half duplex
Maximum cable length	50 feet	4000 feet
Maximum data rate*	20 kbits/s	10 Mbits/s
Typical logic levels	±5 to ±15 V	±1.5 to ±6 V
Minimum receiver input impedance	3 to 7 kΩ	12 kΩ
Receiver sensitivity	±3 V	±200 mV

* Maximum rate at maximum cable length

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

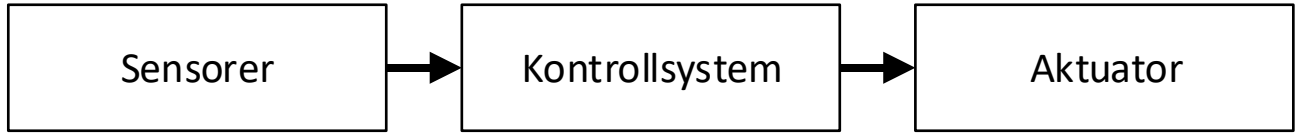
- Når vi har koblet opp en enhet til PC via serieport (fysisk eller emulert), kan vi finne koblingen i *device manager* under «*ports*»
  - Typisk COMx, der x er et nummer gitt av fysisk tilkobling eller OS ved portemulering
- Har vi en COMport åpen, kan vi sende og motta data fra den med et terminalprogram.
- KUN ETT program får bruke èn COMport av gangen.
  - NB: Serial Monitor og programmere Arduino
- 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.
- For å snakke over en COMport, må vi sette opp koblingen slik at sender og mottager benytter samme oppsett (se RS232 og UART)
- Eksempler på Terminalprogram er:
  - Putty, HyperTerminal, Coolterm, Arduino Serial Monitor, TeraTerm, og så videre…

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

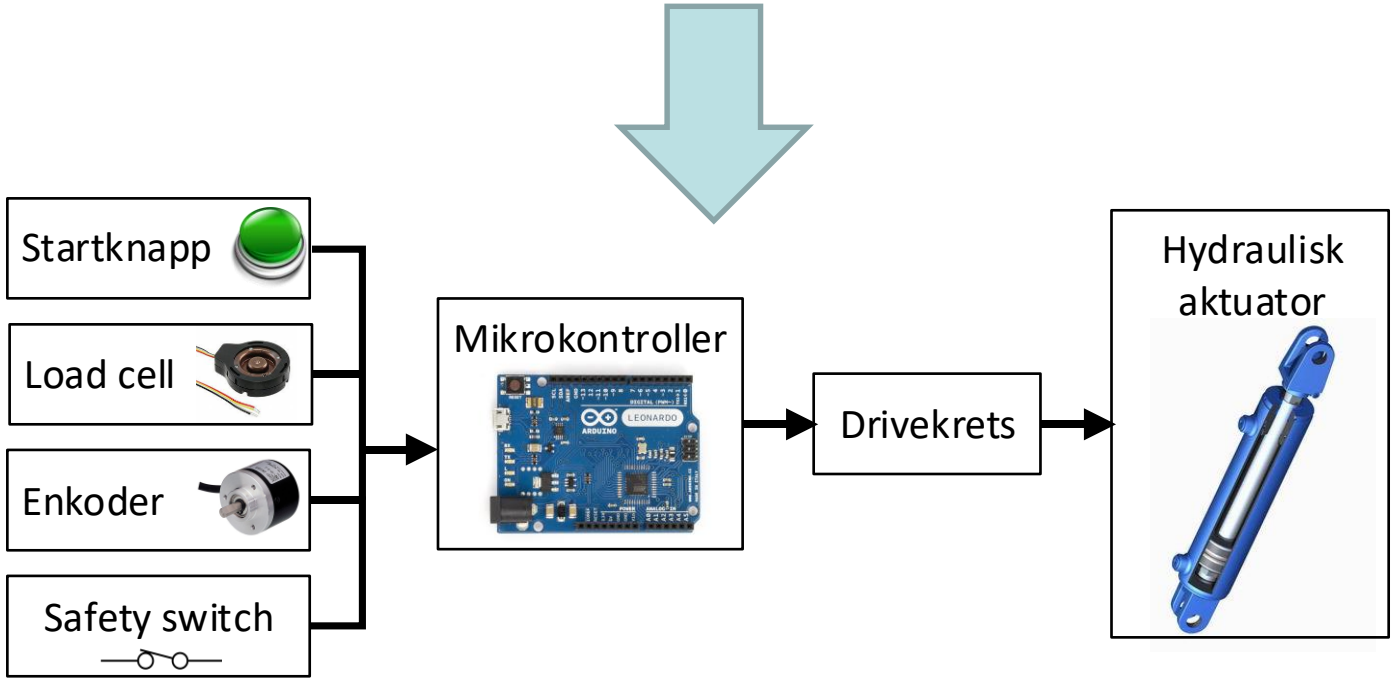
Metode for å designe mikrokontrollerbaserte systemer *Eksempel*

- `1: Definer problemet`
  - 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
        - » Lasteplanet er hevet 45 grader eller
        - » Jeg trykker på startknappen igjen
        - » 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.:

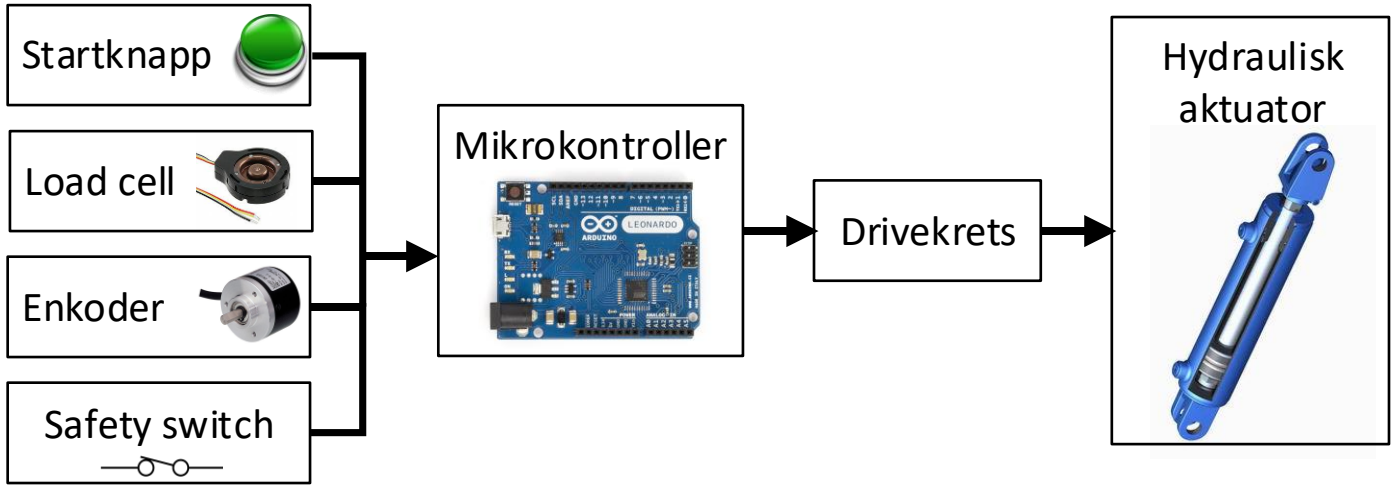


- 2: **Tegn et funksjonsdiagram**
  - Bokser med tekst, symboler eller bilder
  - Skisserer hvordan enhetene er koblet sammen (ikke detaljer)



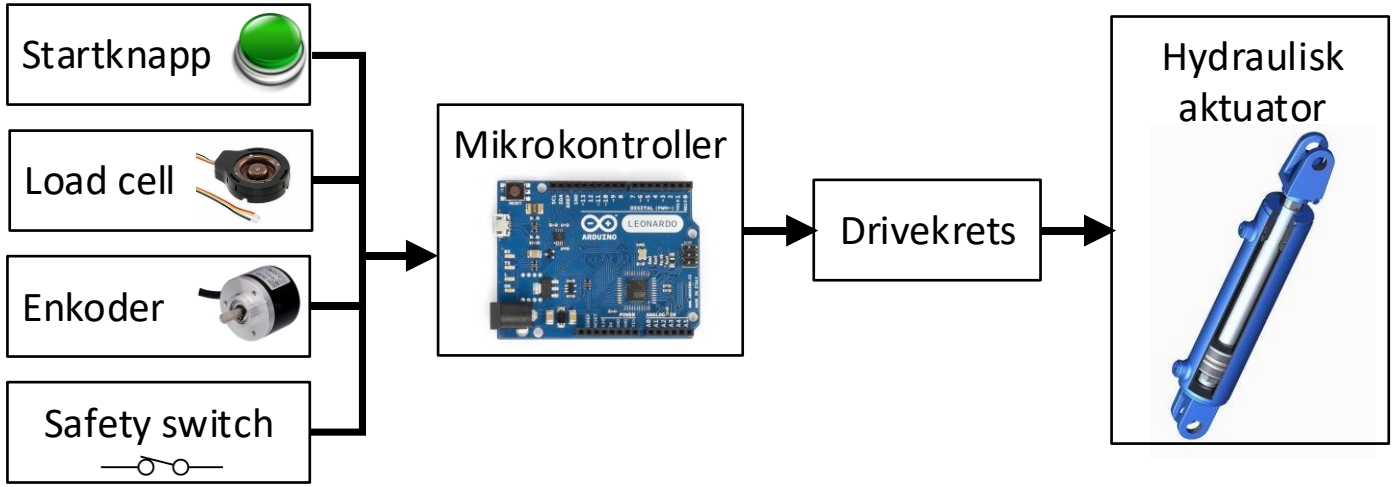
Forts.Mikro.Eks.:

- 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)



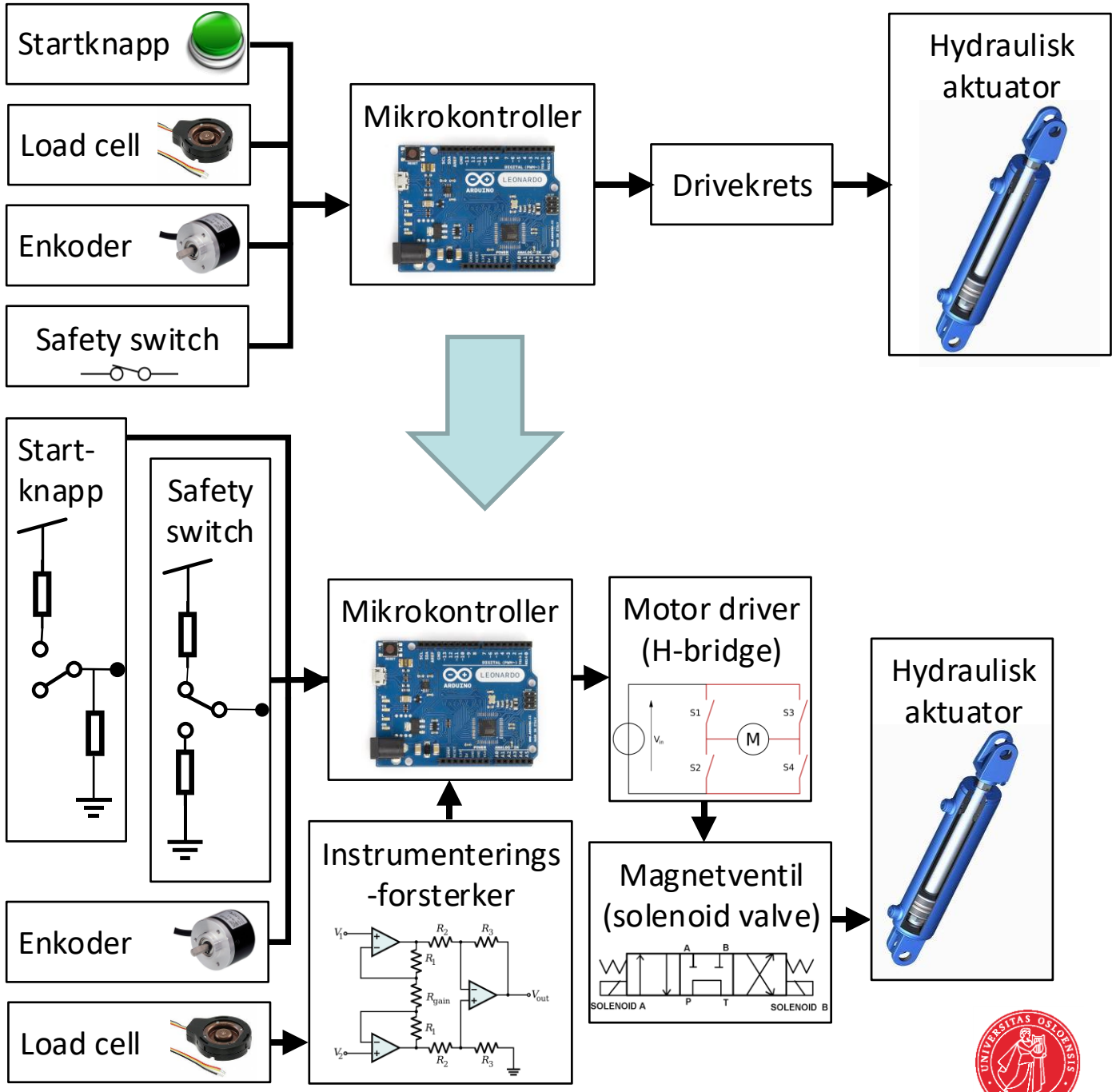
Forts.Mikro.Eks.:

- 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.:

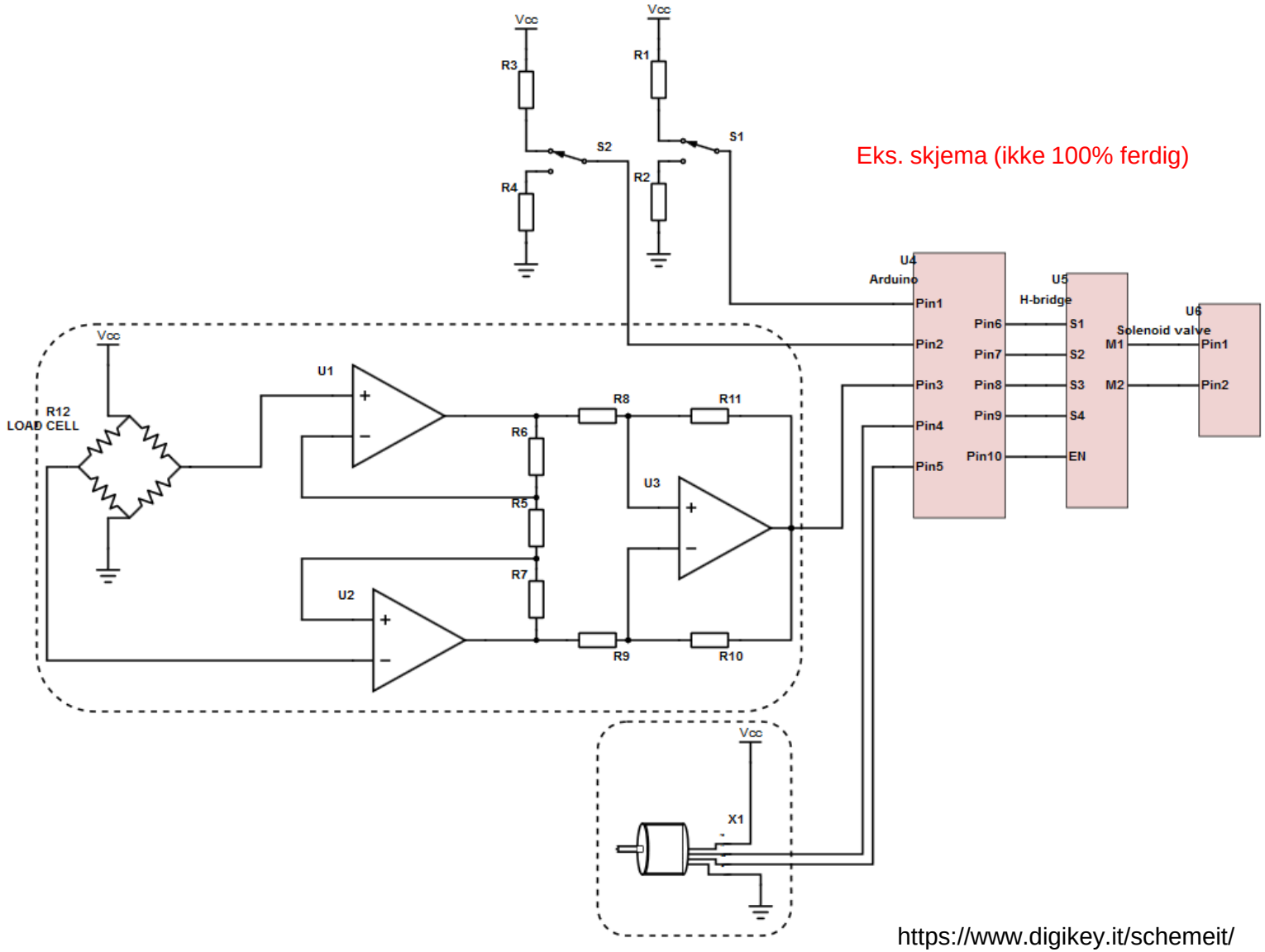
- 5: **Identifiser nødvendige interface kretser**
  - Drivekapasitet til output
  - Forsterkningsbehov for analoge input
  - Pullup/-down for brytere ol.
  - Buffere / 3V-5V konvertere,..
  - (Dekodere)



Forts.Mikro.Eks.:

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

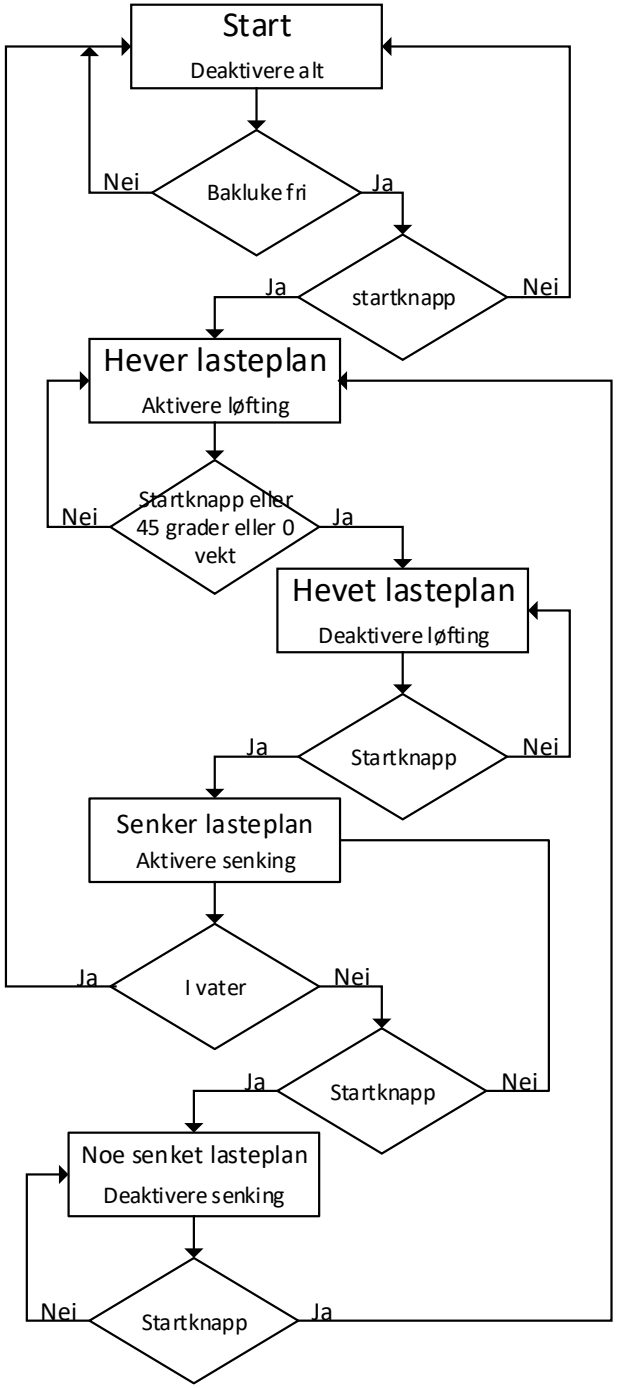
- 7: Tegn detaljert skjema
  - Ta med alt du skal koble til selv.
  - Detaljér. Ikke la pinner du skal bruke henge i lufta.



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

Forts.Mikro.Eks.:

- 8: **Tegn flytskjema**
  - 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!



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

Eksempel mikrokontroller: Arduino

- Bestemte (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
    - Tilsvarer «draw()» i Processing

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

Anbefalt lesing og oppgaver

- Lese
  - 2.1-2.4, 2.8 (2.5-2.10 ellers er orienteringsstoff- gir litt frempek til IN2060)
  - 7.1- 7.3.1.2, 7.4.1- 7.4.3
  - (30.1-30.3 orienteringsstoff, mer om hvordan organisere mekatronikkprosjekter ikke pensum).

- Oppgaver
  - 7.1, 2, 5, 6, 7, 9