



UiO • **Institutt for informatikk**
Det matematisk-naturvitenskapelige fakultet

IN 1080

Busser for innebygde systemer

Yngve Hafting, 2023



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

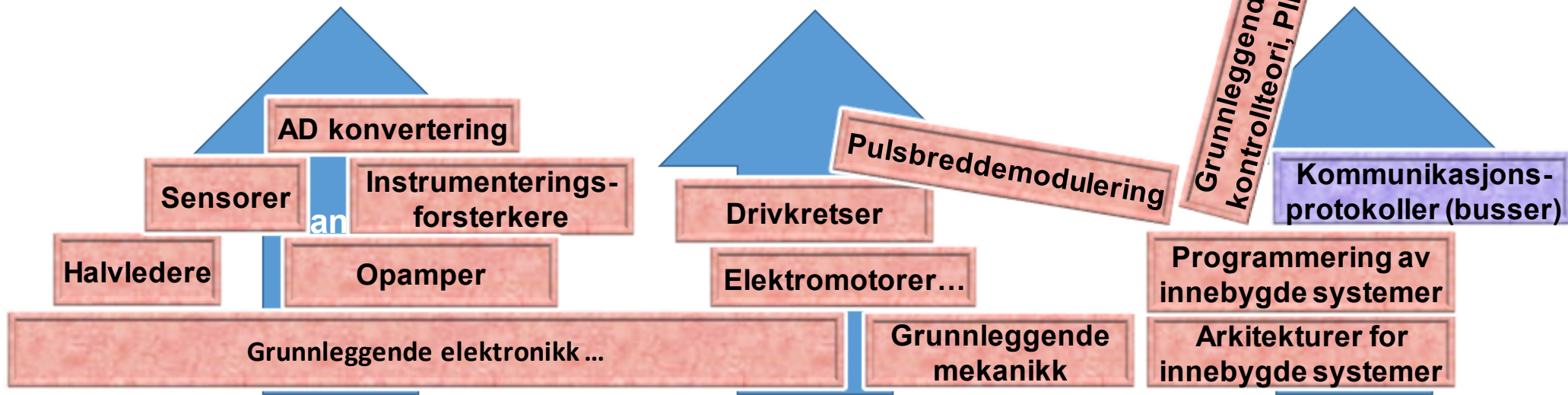
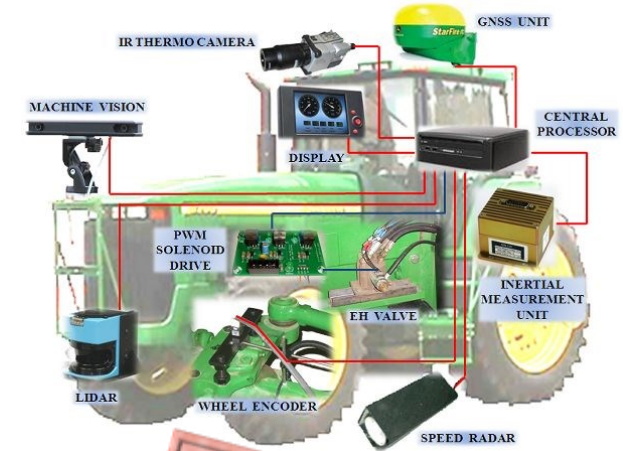
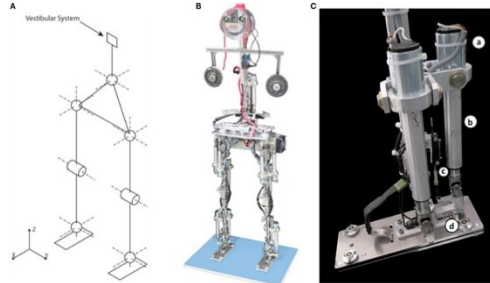
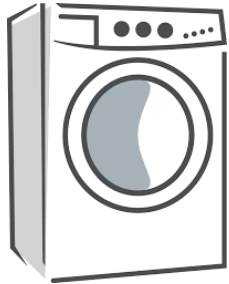
- Ikke lab pga koronavirus => jobb med oppgaver!

Forelesning

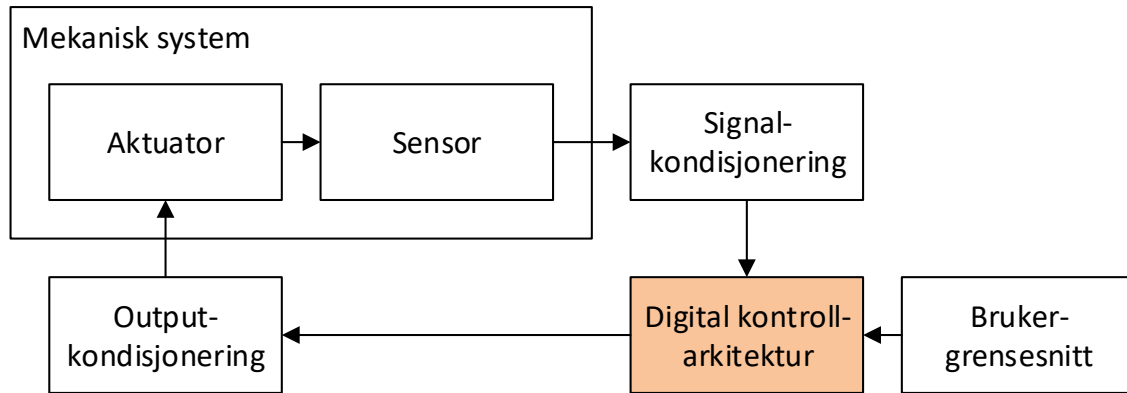
- Kjenne til hva vi mener med innebygde systemer, mikrokontrollere og mikroprosessorer
- Kunne velge sensorsystemer for mekatronikk
 - Kunne redegjøre for virkemåte til noen busser som brukes til å overføre data mellom komponenter brukt i mekatroniske systemer.
- Kunne benytte busser til kommunikasjon mellom mikrokontroller og chipbaserte sensorer
-

Hvor står vi – hvor går vi...

Formål: Å lage og programmere mekatroniske systemer



Systemperspektiv og oversikt



- Hvordan kan vi koble sammen ulike sensorer og systemer?
 - Hvilke prinsipper har vi for digitale sammenkoblinger?
 - Hvordan er disse realisert i praksis?

- Innebygde systemer og periferienheter
- Busser
 - RS232
 - SPI
 - I2C
 - RS 485

Innebygde systemer «Embedded systems»

- = Prosessor/minne + IO-periferienheter
 - Ting- biler, båter, batterisystemer, symaskiner, vaskemaskiner, osv.
 - IO-enhetene styrer andre elektriske eller mekaniske (mekatronikk) systemer
- 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
 - Hendelesbaserte programmer
 - Interruptstyring
- Prozessorsystem kan være
 - Mikrokontrollere (IN1080)
 - Harvard arkitektur
 - ikke OS
 - Single core
 - On-chip minne
 - Eks:
 - » Arduino/ Atmel AVR
 - » Arm Cortex M0
 - FPGAer med tilstandsmaskiner (IN3160)
 - DSPer (Digital signal processor)
 - Mikroprosessorer (IN2060)
 - Von Neumann arkitektur
 - Pipelining, høy klokkefrekvens (GHz)
 - Ofte flerkjerne-systemer
 - » Ofte eksternt minne
 - Ex:
 - » ARM Cortex A7, A72, m.fl
 - » Intel x86

Innebygde systemer fortsetter:

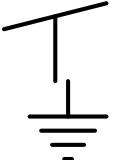
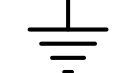
- Periferienheter (peripheral devices)
 - Gjør jobben som systemet er tiltenkt
 - Skjermer, lysdioder (LEDs), drivekretser + aktuatorer, trådløs kommunikasjon...
 - Digitale
 - Mange ferdigdefinerte busser og protokoller
 - USB, SPI, I2C, UART, m.fl.
 - Analoge
 - Signaler må kondisjoneres og konverteres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

Digital kommunikasjon med mikrokontrollere

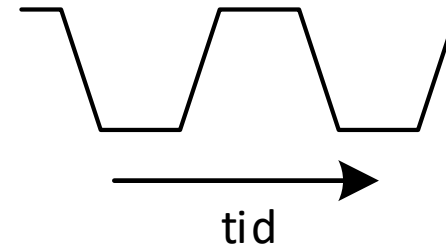
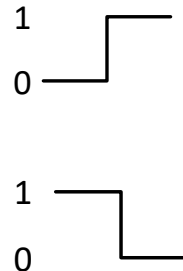
- Vi skal fordype oss i noen mye brukte varianter..

Noen begreper / notasjon

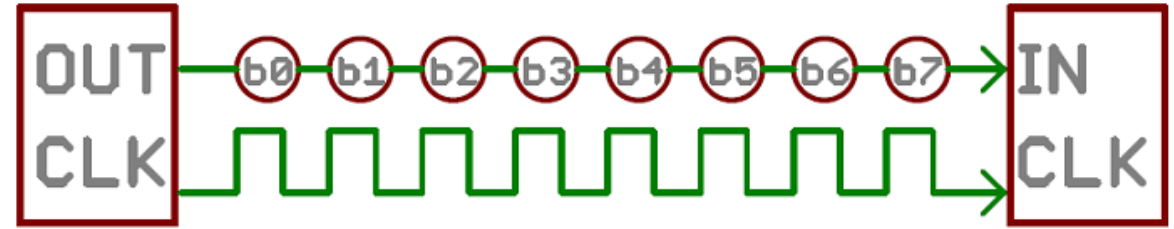
- Typiske representasjon digitale signaler

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.

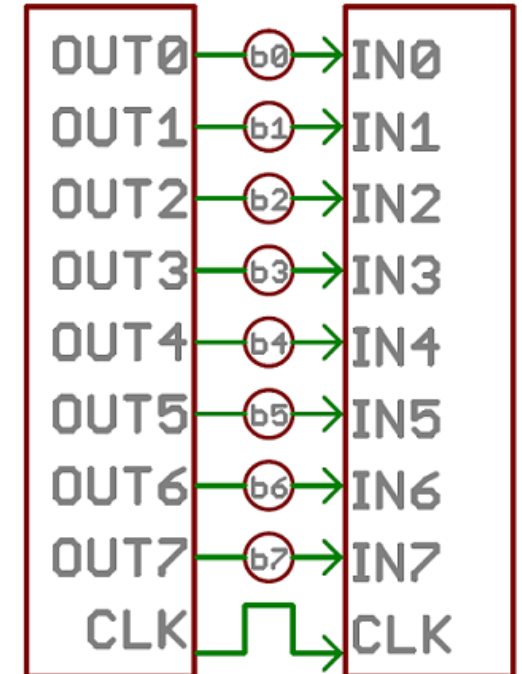


Bussterminologi



Seriell buss ↑ ,Parallell buss ↓

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



Bussterminologi

- Master vs Slave
- Host vs Device

- Det er host/master som "bestemmer" og typisk ber om eller sender data på bussen.

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 (Lengre avstander)
Seriell	UART, SPI, I2C	UART, SATA (harddisk)	RS485, USB1 og 2, Ethernet
Flere Serielle		SCSI, PCI-e	Displayport, HDMI, Thunderbolt, USB 3.x
Parallell	Minnebuss	(Parallellport), (PCI), Minnebuss	



- Utvikling: fra parallellkoblinger (tykke ledningsbunter) til mer seriell kommunikasjon
 - Overføring av mange bit samtidig = trøbbel ved høye hastigheter og store avstander
 - Signalhastighet = ca $1/3 - 2/3$ av c
 - $(1/2) * 3 * 10^8 \text{ m/s} = (3/2) * 10^8 \text{ m/s}$
 - 5GHz RAM: gir en klokkeperiode... ($f=1/T \Leftrightarrow T=1/f$)
 - $1/(5 * 10^9 \text{ Hz}) = 2 * 10^{-10} \text{ s}$
 - Avstand tilbakelagt på en klokkeperiode ($s=vt$)
 - $= (3/2) * 10^8 \text{ m/s} * 2 * 10^{-10} \text{ s} = 0,03 \text{ m} = 3 \text{ cm}$
 - \Rightarrow *Forskjellig ankomsttid på bit = trøbbel.*

Eksempler Busstyper

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



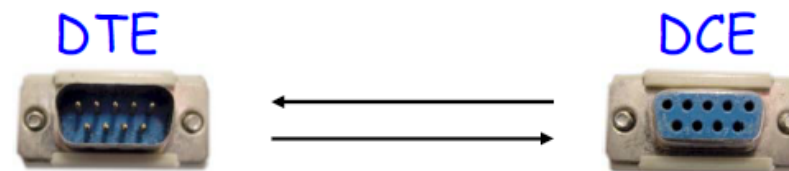
- **Differensiell seriell kommunikasjon:**
 - tillater høyere hastighet på dataoverføring per linje.
 - Lavere effektforbruk fordi pos. og neg. ladninger følger hverandre i ledningspar
 - Samtidighet er ikke et problem når vi leser 1 bit av gangen.
 - Eks: USB har D+ og D- i tillegg til VDD og GND. D+ og D- er alltid motsatt ved dataoverføring.
- **Parallele busser brukes typisk internt i komponenter (chiper), og mellom minne og prosessor**
 - Minnebuss = Data buss og Adressebuss

"RS-232" - Serieport - Asynkron Seriell kommunikasjon

- RS232 "Serieport"
 - opprinnelig laget på 60 tallet beregnet for kommunikasjon mellom datamaskin og modem ved +/- 15V...
- Halv duplex eller full duplex
 - Full duplex krever flere ledninger
- Toveis seriell datakommunikasjon.
- Kommunikasjon mellom 2 enheter (= PC/terminal og modem)
- Mulighet for en rekke ekstra ledninger for dataflytkontroll (handshake/ status)
 - i prinsippet brukes bare to pinner til kommunikasjon, Rx og Tx
- I dag benyttes serieport typisk mellom kretskort med 5V, 3,3V eller lavere spenninger (*TTL – transistor-transistor level*)
- Hastighet < ca 200kbps

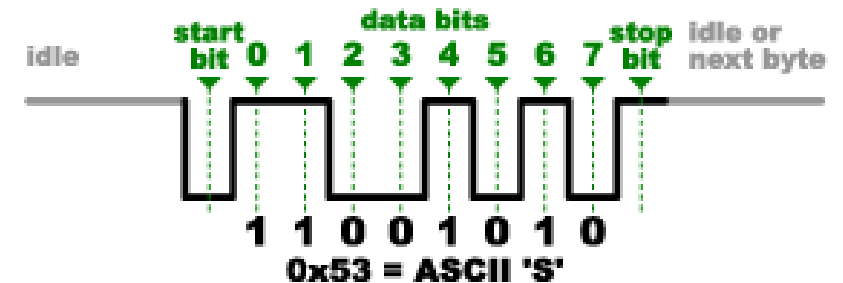
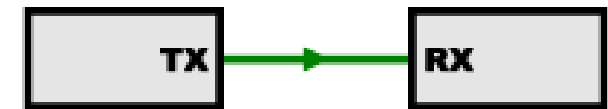
Mer på Wikipedia

<https://no.wikipedia.org/wiki/RS-232>



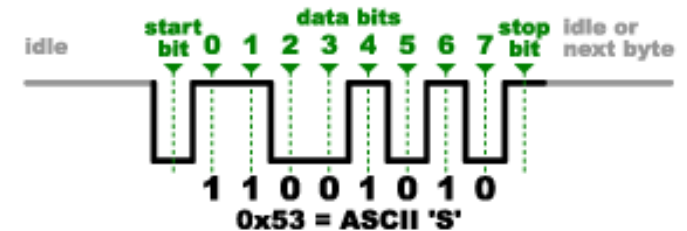
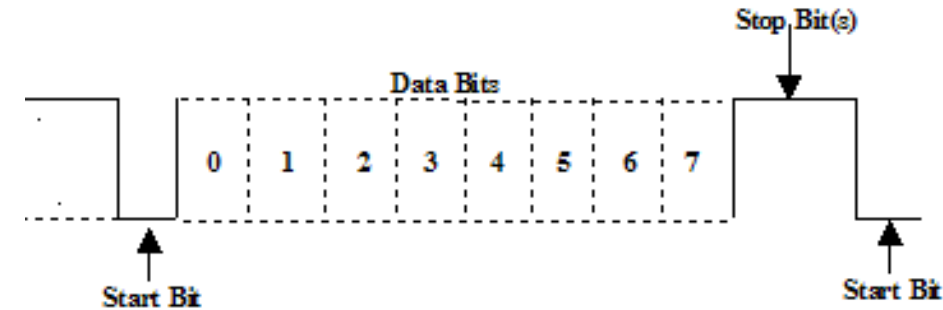
UART protokoll

- For å overføre data med serieport brukes gjerne en **UART**
«Universal asynchronous receiver-transmitter»
 - UART står for overføringen av data og presenterer dem parallelt i mikrokontrolleren.
 - Normalt er UART innebygd i mikrokontrollere.
- Klokke data overføres ikke (derav asynkron).
- «**baud rate**» vs «**bit rate**»
 - Baud = symbolrate
(opprinnelig multnivå logikk)
 - Bestemmes i forkant!
 - Typisk 9600 -115200 bps



UART protokoll

- 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*

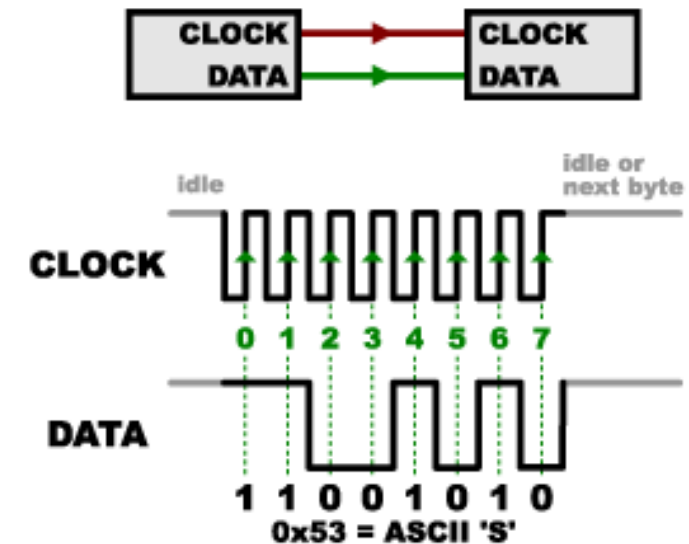


Virtuelle serieporter (COM port)

- Serieporten er gjerne tatt helt bort fra moderne PCer
- Kobler vi til en Arduino til PC gjør vi det med USB
 - Deretter setter vi opp en virtuell serieport
 - Denne må settes opp som en hvilken som helst serieport
 - Mindre behov for paritetssjekk ol. som håndteres av underliggende lag.
 - NB: Kretskort med separate USB kontrollere (egen chip) snakker gjerne serielt videre.
 - Da kan paritetsbit bli vesentlig allikevel..

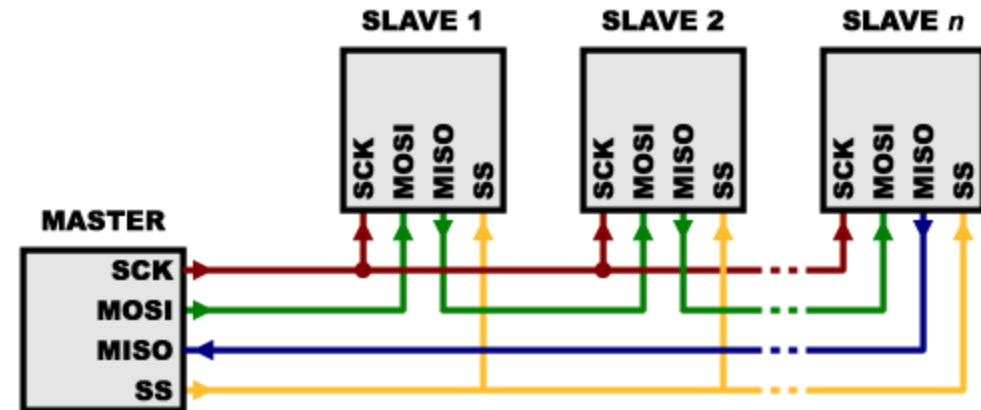
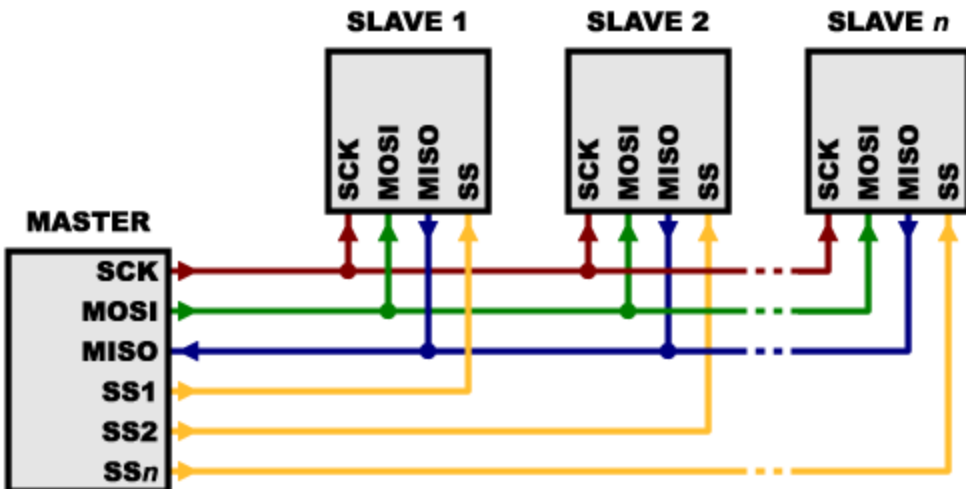
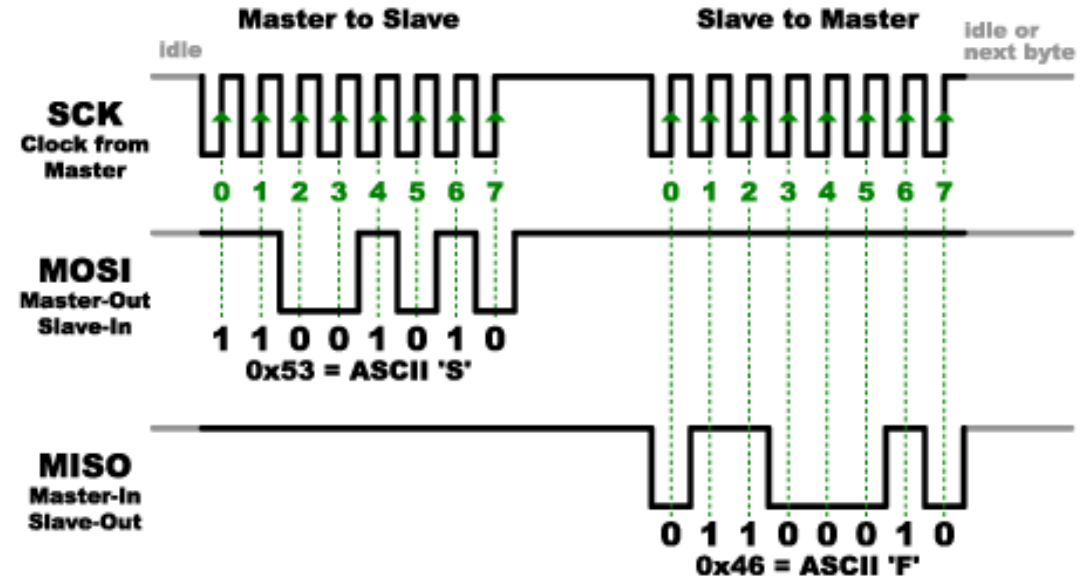
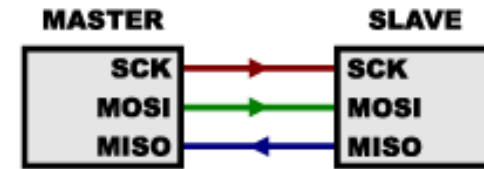
SPI – Serial Peripheral interface

- *Mye brukt fordi det er enkelt å implementere.*
- Synkron, full duplex seriell dataoverføring
- En Master, en eller flere slaver.
- Klokke overføres via egen linje
 - *slavene behøver ikke egen 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



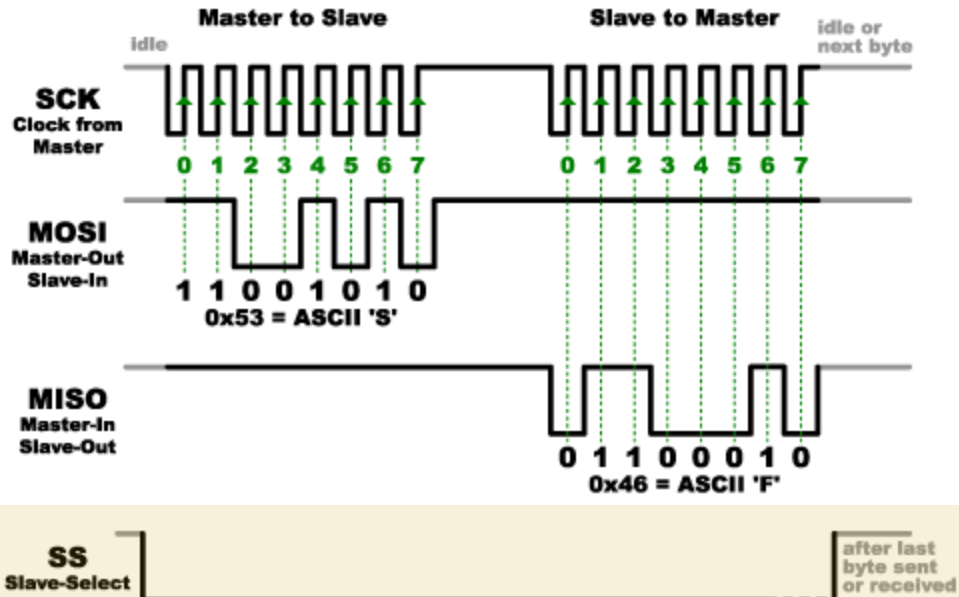
SPI- protokoll

- Master som klokker all dataoverføring
 - må vite hvor mye data den skal motta fra hver slave.
- Master kan enten ha
 - en slave-select linje for hver slave,
 - eller 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.

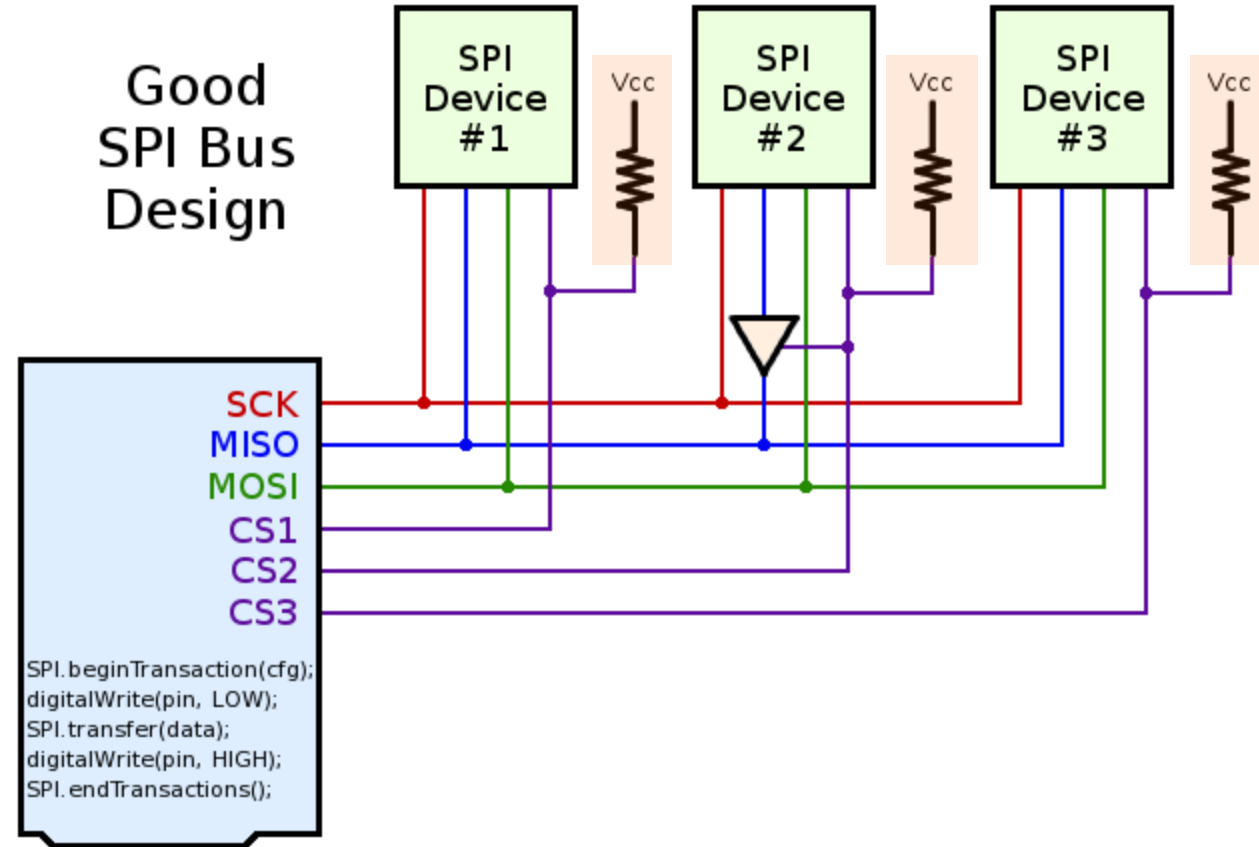


SPI kjedekobling: Merk kobling MOSI-MISO

SPI forts.

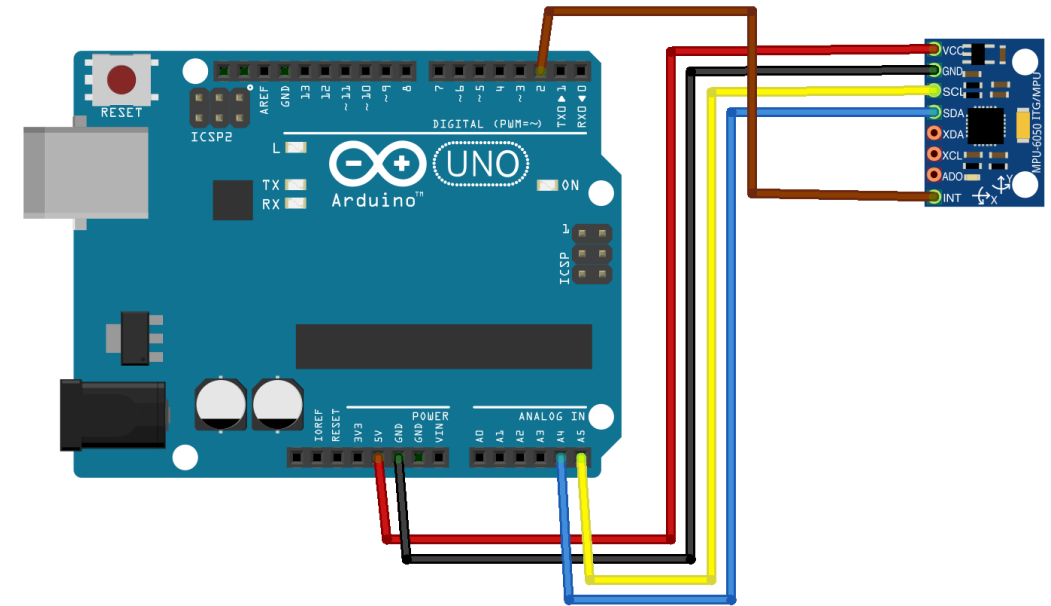


Good SPI Bus Design



I²C «Inter-Integrated Circuit Protocol.»

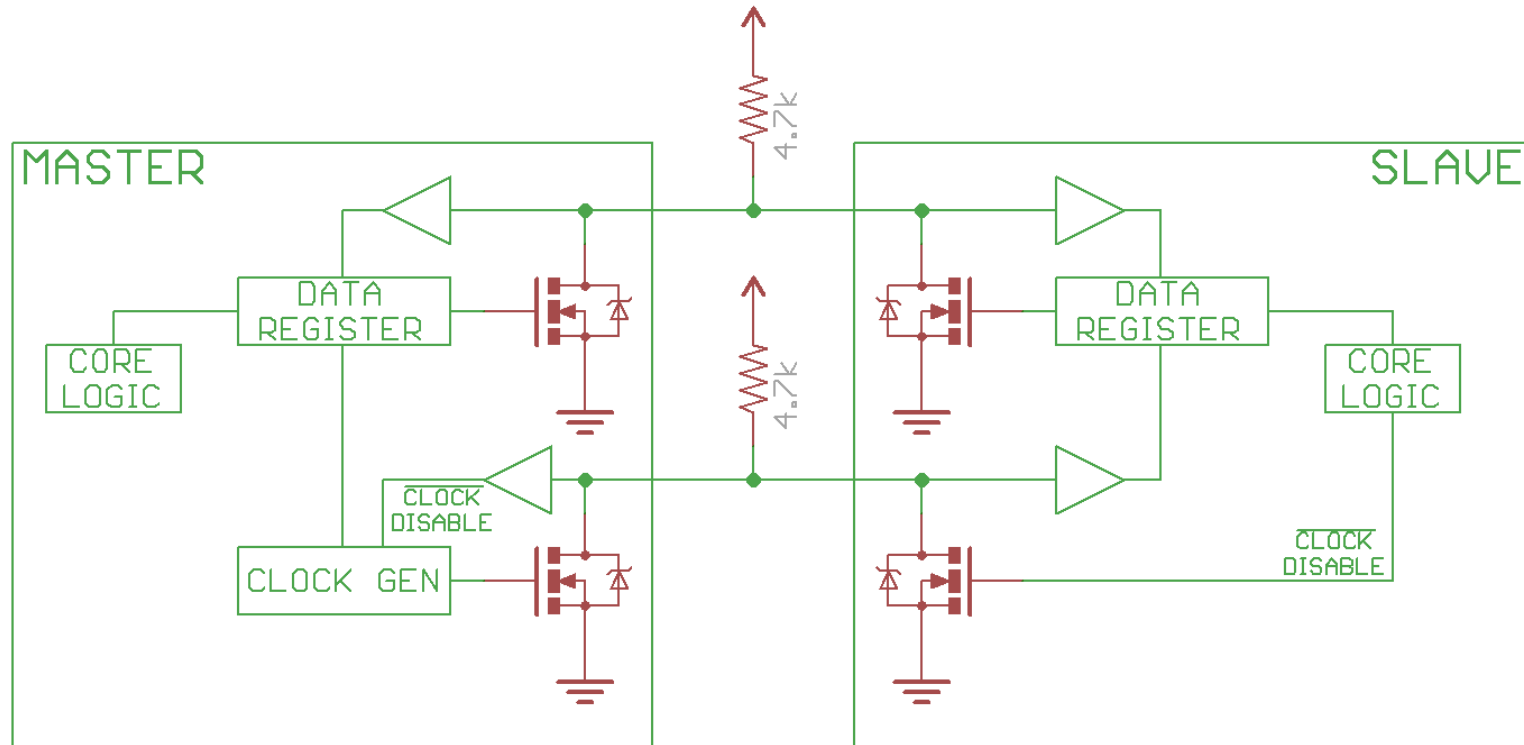
- I2C Bruker kun 2 linjer: SCL og SDA
- Kan ha svært mange enheter koblet på samme buss-
 - inntil 1008 slaver.
 - Flere enheter kan ha rollen som master
- Overføringshastigheter:
 - 100kbit/s «standard mode»
 - 400kbit/s «full speed»
 - 1Mbit/s «fast mode»
 - (3,2 Mbit/s «high speed» & 5 Mbit/s «ultra high speed»)
 - Ulike fysiske krav (intern pullup + capacitance)
- 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



fritzing

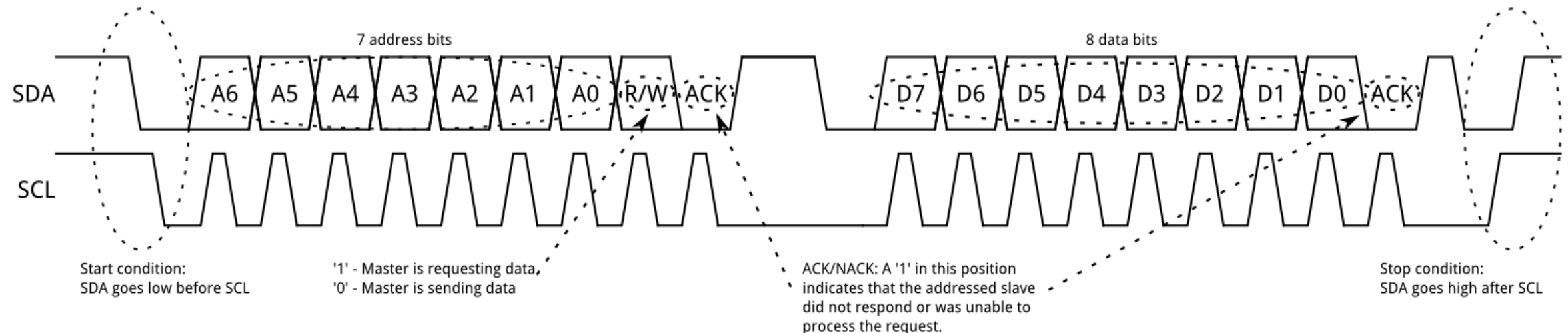
I²C – Oppsett - Mange til mange, Uten kortslutninger

- Både SCL og SDA er open drain (open collector for BJT).
 - enhetene kan bare trekke linjene til jord,
 - aldri sende ut et høyt signal.
 - **Krever pullup** for å virke
 - Normalt 4,7k Ω
 - Ved flere enheter kan det kreves lavere verdier
 - Ved ulike forsyningsspenninger kobles pullup til den laveste
 - forutsatt at nivået er høyt nok ...



I2C protokoll

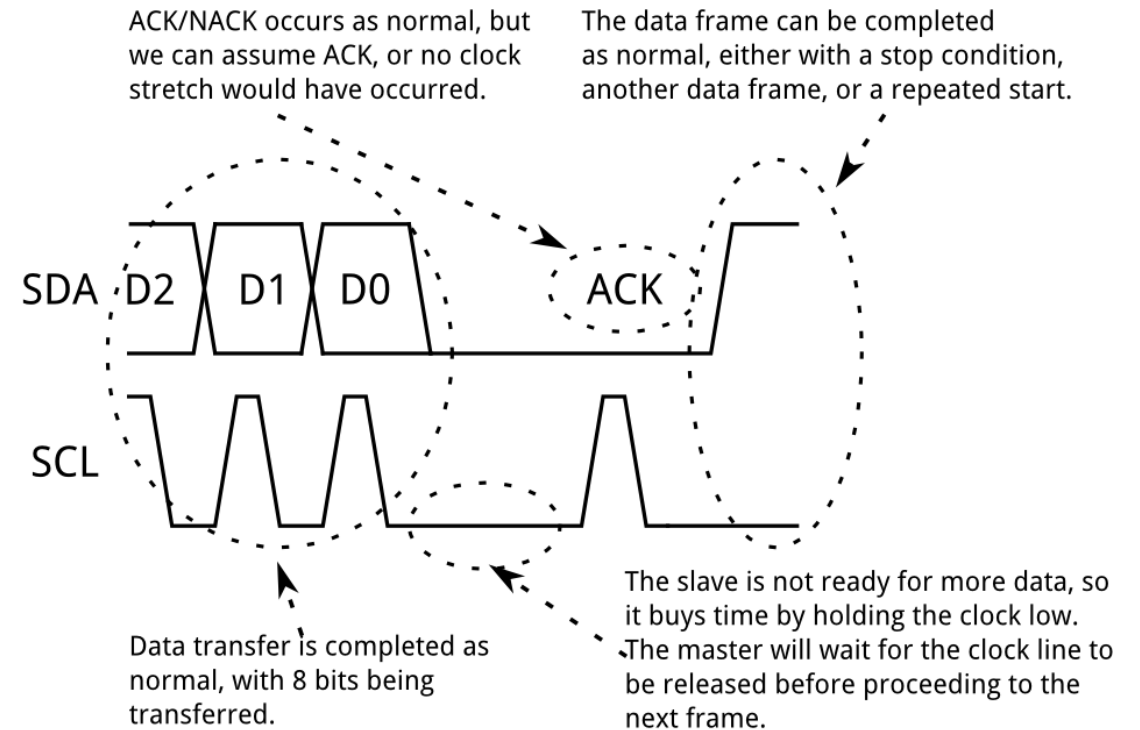
- Master trekker SDA lav for å få kontroll på bussen.
 - Den som trekker bussen lav først får den (!)
 - SCL høy og SDA lav gjør at de andre enhetene lytter
 - Videre sender master en pakke med en 7 bits adresse,
 - fulgt av et bit som forteller om Master vil ha data fra eller skrive data til enheten den adresserer.
 - Master klokker SCL, data leses på stigende flanke
 - Enheten som ble adressert trekker SDA lav (ACK),
 - Dette må skje på den 9. klokkeflanken
 - ellers er ikke dataene å regne for lest (NACK).
- Etter at adressedataene er sendt og akseptert, fortsetter Master å klokke SCL-linjen.
 - Ved «request» vil slaven sende data
 - Ved «write» sender Master data til slaven
 - 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
 - Startes med en reservert adresse (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»)

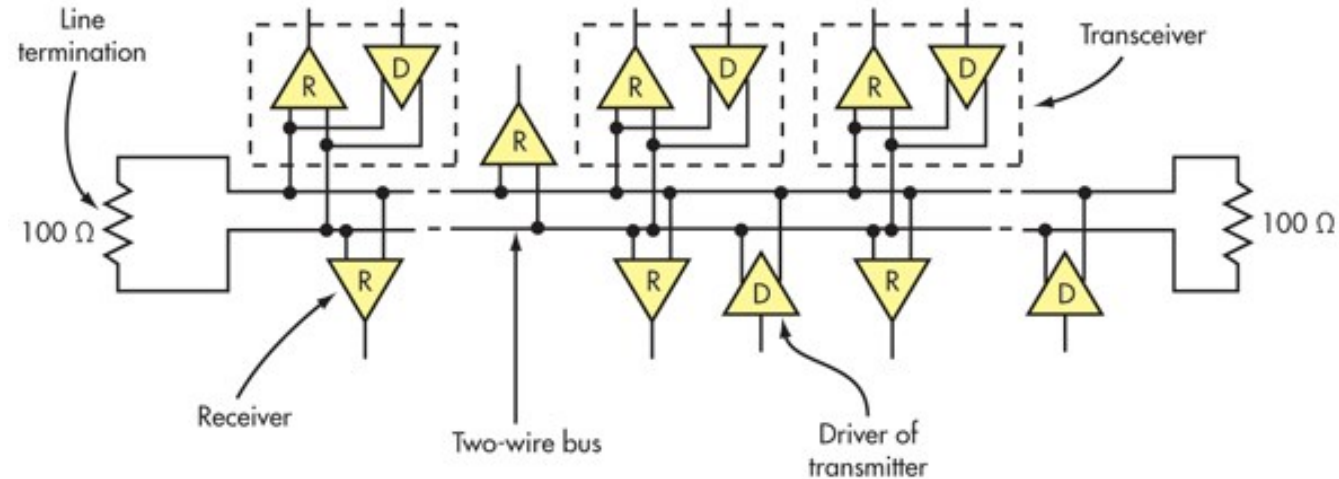


For detaljer, se

<https://learn.sparkfun.com/tutorials/i2c>

Differensiell kommunikasjon – RS485

- Fysisk standard laget for kommunikasjon over større avstander (enn RS-232).
- Linjene drives differensielt, => mindre støy.
 - $|V_d| \geq 200 \text{ mV}$
 - Retningen på strømmen bestemmer om det er 0 eller 1.
 - Termineringsmotstander viktig for å ikke få refleksjoner
 - Vi går ikke i dybden på dette
- Kan kjøre både full og halv duplex
 - Drivere i **tristate** når de ikke er i bruk. (Receiver alltid på).
- *Ingen definert protokoll*
 - UART brukes ofte
 - RS485 brukes som fysisk standard i flere andre busser
- Bruker «Twisted pair» kabel med eller uten skjerming
 - Hastigheten avtar med kabellengde
- Andre differensielle bustyper:
 - RS422
 - LVDS, M-LVDS... (Low voltage differential signaling)



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

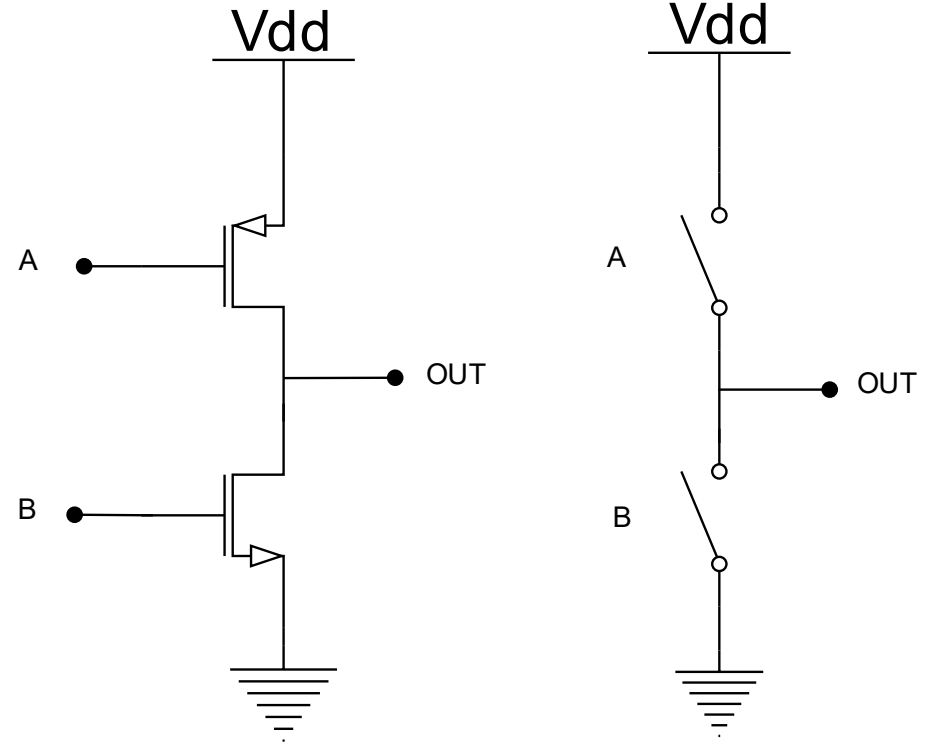
KEY CHARACTERISTICS OF THE RS-232 AND RS-485 SERIAL INTERFACES

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 $\pm 15 \text{ V}$	± 1.5 to $\pm 6 \text{ V}$
Minimum receiver input impedance	3 to 7 k Ω	12 k Ω
Receiver sensitivity	$\pm 3 \text{ V}$	$\pm 200 \text{ mV}$

* Maximum rate at maximum cable length

Tristate, Høy Impedans 'Z'

- En driver er i tristate når både høy og lav side er av.
- CMOS drivere består typisk av halvbroer
 - Ikke alltid separate tilgang til begge sidene
 - Databladet til en brikke forteller om vi kan sette utgangene i tristate.



A	B	OUT
0	0	Z
1	0	1
0	1	0
1	1	☠

I praksis...

- *Som regel* holder vi på med brikker der signalprotokollen er ferdig implementert.
 - Vi må velge brikker som kan bruke samme type buss for å snakke sammen.
 - Vi må forstå hva som kreves fysisk (pullup motstander, termineringer osv) for å koble sammen komponenter på busser.
- Arduino: ferdigskrevne klasser og HW-implementert UART og I2C osv.
 - holder styr på timingen for oss.
 - => sørge for at koblingene er riktige, og at vi setter opp enhetene riktig før bruk
 - 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

Oppsummering busser

- Fysisk sammenkobling og (eller) protokoll for dataoverføring
 - De fleste er fysiske
 - Virtuelle busser = kun protokoll- bruker gjerne en annen fysisk buss.
- Kan være 1-1 (ptp), 1-mange eller mange-mange
 - Gir forskjellige elektriske løsninger vi må kjenne til
 - Pull-up motstander
 - Termineringsmotstander
 - Tristate logikk
 - PtP-løsninger som snakker med mange bruker aktive HUBer og Switcher
 - Ex Ethernet, USB, etc.
- Protokoll tar for seg logisk kommunikasjon
 - Bruk av paritetsbit
 - Handshake
 - Flankestyring

Oppsummering

- UART
 - Protokoll for serieport
 - Brukes både virtuelt og i fysiske busser
 - Må settes opp på forhånd med baudrate, paritet osv.
- SPI
 - Én master, flere slaver
 - Kan kreve tristate implementasjon
 - Klokke + datalinjer => 4 wire (MOSI, MISO, CS, GND)
 - Slave select må ha pullup (aktiv lav)
- I2C
 - Mange til mange, alle *kan* være master
 - Ikke tristate, men pullup
 - Alle kan ta kontroll ved å trekke linjene lave
 - Klokke overføring
 - Bruker adresser
- Andre (ikke typisk for innebygde systemer):
 - RS-485
 - Kun fysisk grensesnitt
 - Differensiell
 - Termineringsmotstander
 - Tristate
 - USB (Mange standarder)
 - Både fysisk grensesnitt og protokoll
 - Differensiell
 - PtP
 - Flankestyrt "NRZI", m.fl
 - Logisk/SW protokoll
 - spesifikasjon på hundrevis av sider,..
 - Ethernet
 - Differensiell, ptp
 - LVDS ,..
 - Fysisk standard for differensiell kommunikasjon.
 - Canbus
 - Laget for kjøretøy
 - Ligner RS485, men med protokoll

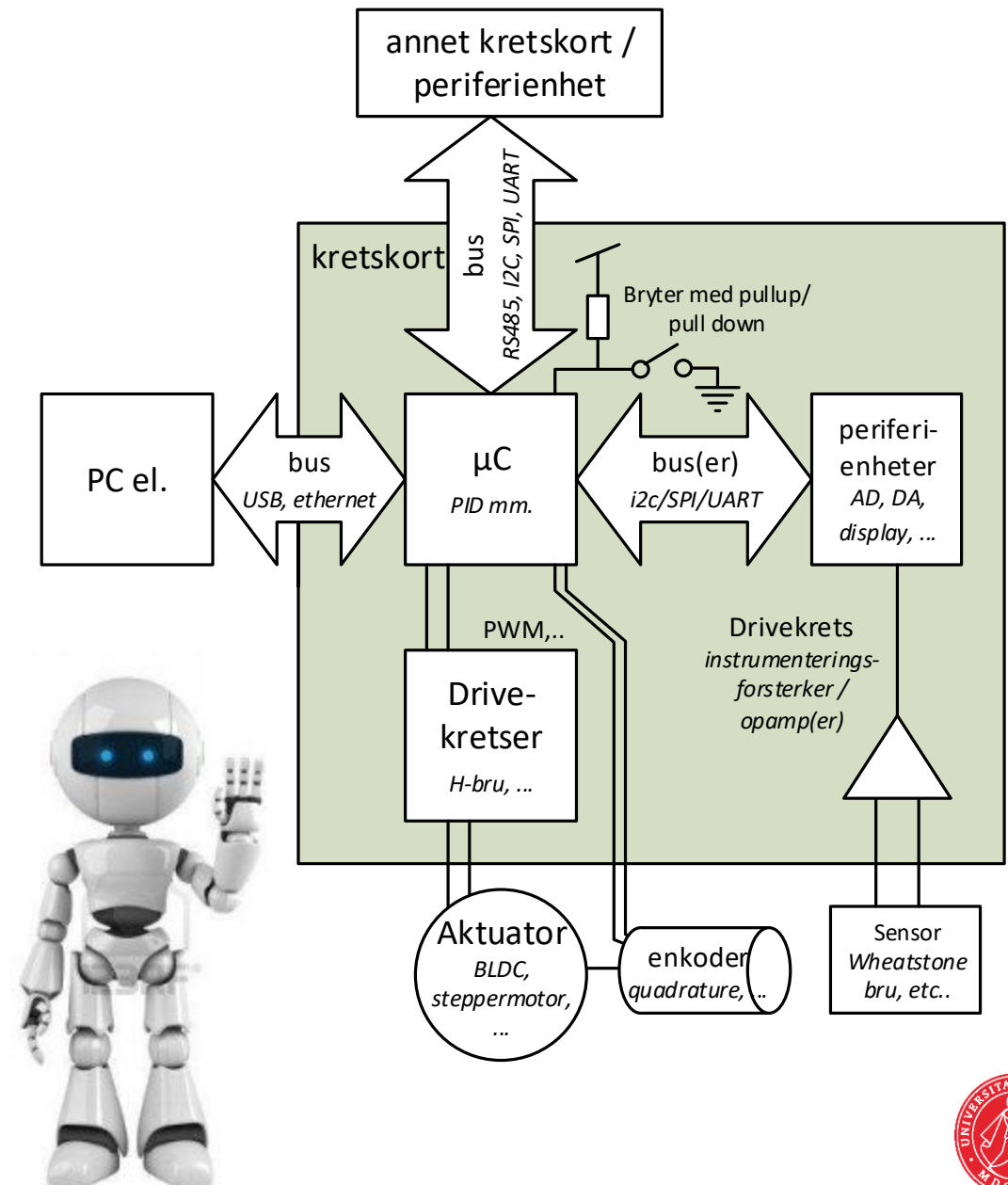


Anbefalt lesing og oppgaver

- Lese
 - 7.1- 7.3.1.2, 7.4.1- 7.4.3
- Oppgaver
 - 7.1, 2, 5, 6, 7, 9
- Eksamensoppgaver
 - 2021: 19

Mekatronikk- Hvor står vi

- 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



Veien videre (Robotikk bachelor) – Hva bygger videre:

- IN2060 Digitalteknikk og datamaskinarkitektur
 - mikroprosessorarkitektur (≠Arduino), assembler, litt c, litt om tilstandsmaskiner.
- IN3140 Introduksjon til robotikk
 - Mer om kontroll, kinematikk, styre roboter med flere akser
- IN3050 AI, ML
 - Hvordan lære(r) et system oppførsel..?
- IN3160 Digital systemkonstruksjon
 - Enda mer tilstandsmaskiner, bygge digitale kretser

...

Master i

Kybernetikk og autonome systemer eller

Robotikk og intelligente systemer

