IN 1080

F-13: ARDUINO, PROCESSING, BRYTERE OG BUSSER

Yngve Hafting, Forelesning 13

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

OVERSIKT

- Arduino og Processing
  - Hvordan komme i gang
  - Jobbe med standardbiblioteker
  - Tilgjengelig eksempelkoder
- Brytere
- Pullup/motstander
- Avkoblingskondensator
- Busser
  - RS232
  - SPI
  - I2C
  - RS 485
- Terminalprogrammer

OBLIGEN: ARDUINO OG PROCESSING

- Begge kan pakkes ut med 7zip og kjøres fra hjemmeområde
- www.arduino.cc
  - Laste ned rammeverk (ca 500MB pakket ut.)
  - Obligen bruker Arduino Leonardo.
  - Driver må installeres, og man må gi tilgang til å bruke nettverk (popup)
    - Drivere skal være installert på laben
- www.processing.org
  - Laste ned rammeverk (ca 250 MB pakket ut)
  - Processing bruker java (Python er ikke ferdig)
- Begge finnes på CADlib: \\bifrost\project0\robin\CADlib\IN1080\2018\
  - Versjon: 1.8.5.
  - Egen PC: Like raskt å laste ned fra web.
  - Inneholder både Arduino og Processing, inkludert arduino drivere.
    - Begge kan kjøres direkte fra området, men drivere må installeres når kort skal brukes.
    - NB: Det kan gå langsomt.

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…

https://store.arduino.cc/arduino-leonardo-with-headers

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

PROCESSING OG ARDUINO DEMO

- Hvordan komme i gang
- File->Examples->Libraries->Serial->SimpleWrite
  - (inneholder også arduino kode)
- Koble til arduino og finn portnavn
  - Endre portnavn i Processing
- arduino rammeverk:
  - Kopiere tekst fra processing til arduino
  - Modde LEDpin til 13.
  - Sjekke at riktig kort og port er valgt («tools» meny)
  - Kompilere og kjøre (programmere arduino)
- Processing
  - Kompilere og kjøre
    - (gi evt tilganger underveis)

- Jobbe med standardbiblioteker
- Tilgjengelig eksempelkoder

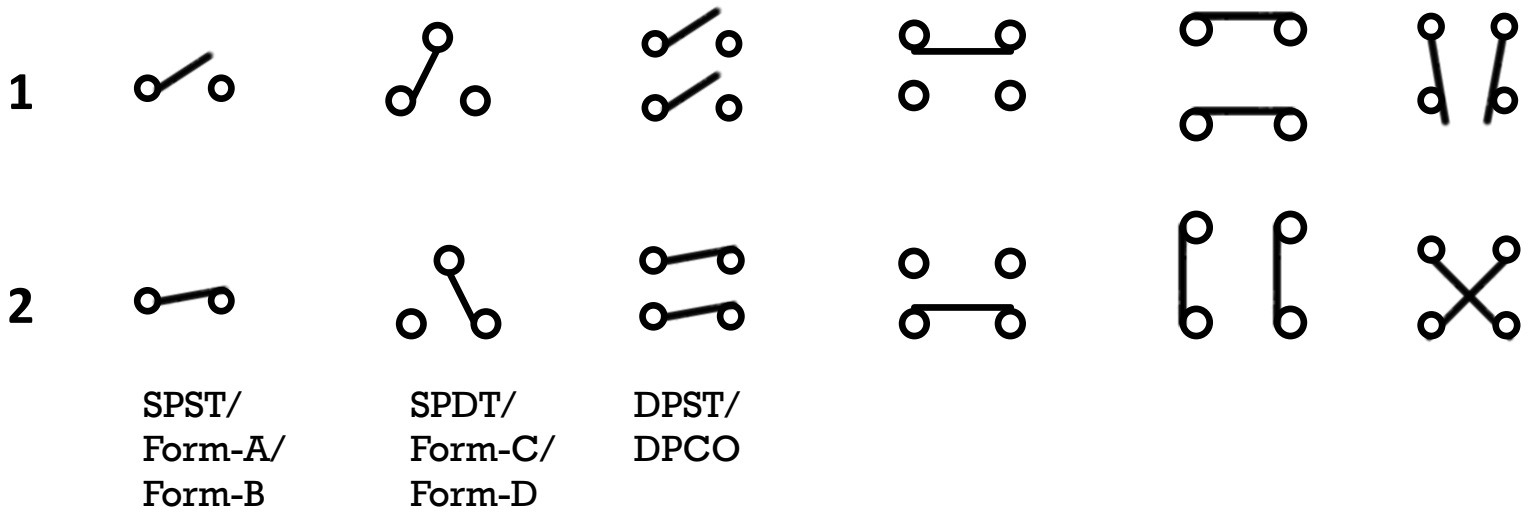
Andre relevante processing eksempler:

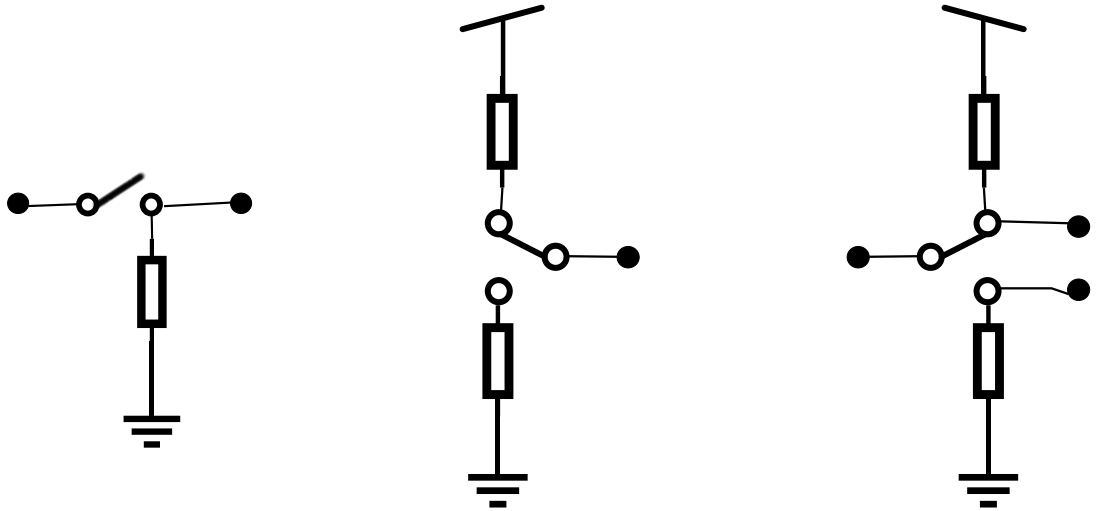
- Examples->Topics->Continous Lines
  - Tegner strek vhja mus
  - Kan oversettes til å tegne graf…
- Examples->Basics->Data->CharacterStrings

=> Se på noen eksempler, bruk en du kan tenke deg.

BRYTERE

- Bryterkonfigurasjoner
  - Lett å avsløre med multimeter
- Pullup / Pulldown
  - Brukes i forbindelse med
    - Brytere
    - Busser
    - IO- porter
- Prell
  - Vanlig i mekaniske kontakter. Digitale systemer må ta høyde for at det forekommer.
- Terminologi
  - Firebokstavers koder for brytere se https://en.wikipedia.org/wiki/Switch
  - P = «Pole» er antall brytere
  - T = «Throw» er antall muligheter per pol
  - S = «Single»
  - D = «Double»

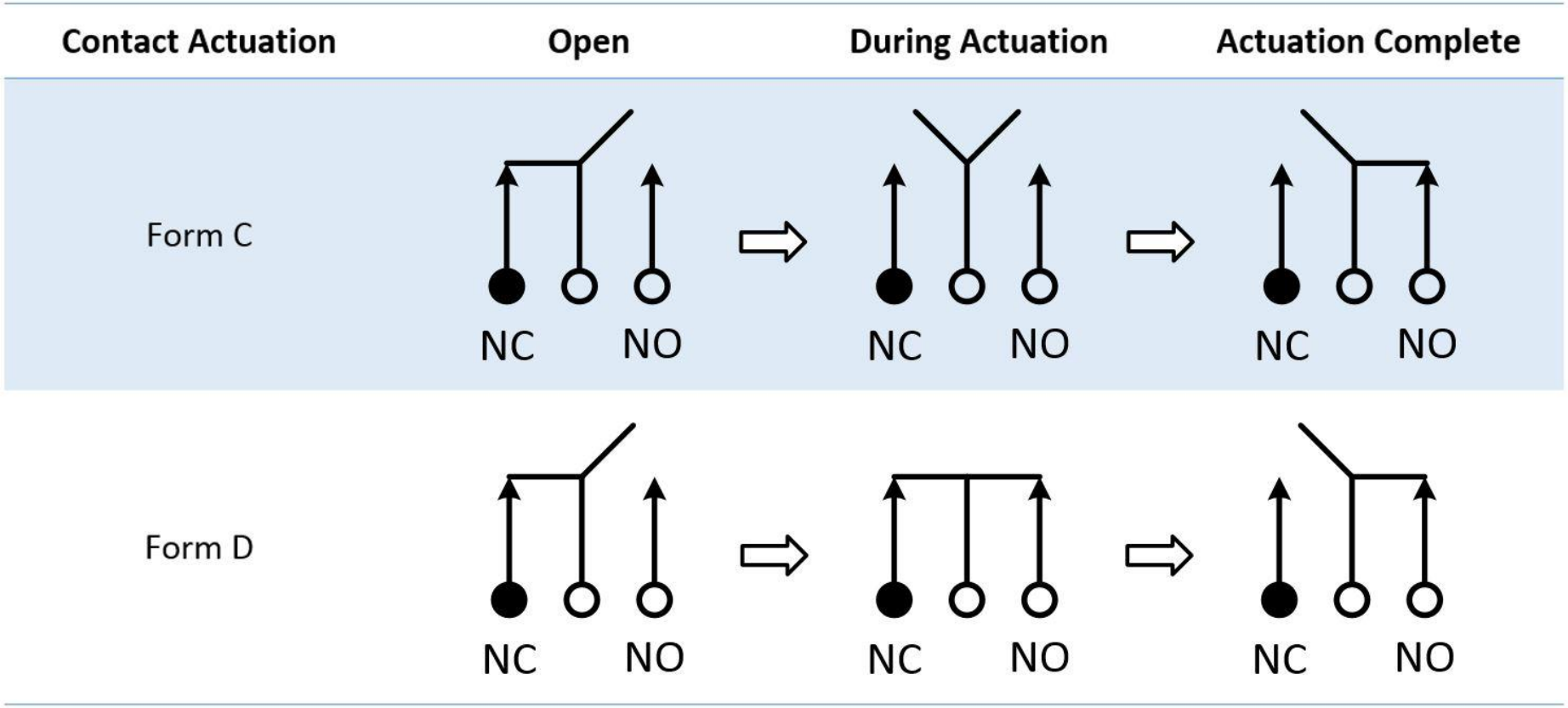




BRYTERE- TERMINOLOGI

Hver pinne kan beskrives som:
NC = Normally Closed
NO = Normally Open

MERK forskjell på form D og form C før du har fullført aktuering



http://www.ni.com/white-paper/3960/en/

OPAMP SOM DIFFERENSIELL FORSTERKER

- For å forsterke signalet til wheatstone-broen til trykksensoren må man bruke opampen som differensiell forsterker. Dette gjøres ved å konfigurere to eller tre opamper til en instrumenteringsforsterker.
  - Databladet til Opampen (MCP601) har eksempler (se figur).
- For å få en pekepinn på hvor mye forsterkning som trengs må man benytte databladet til trykksensoren.

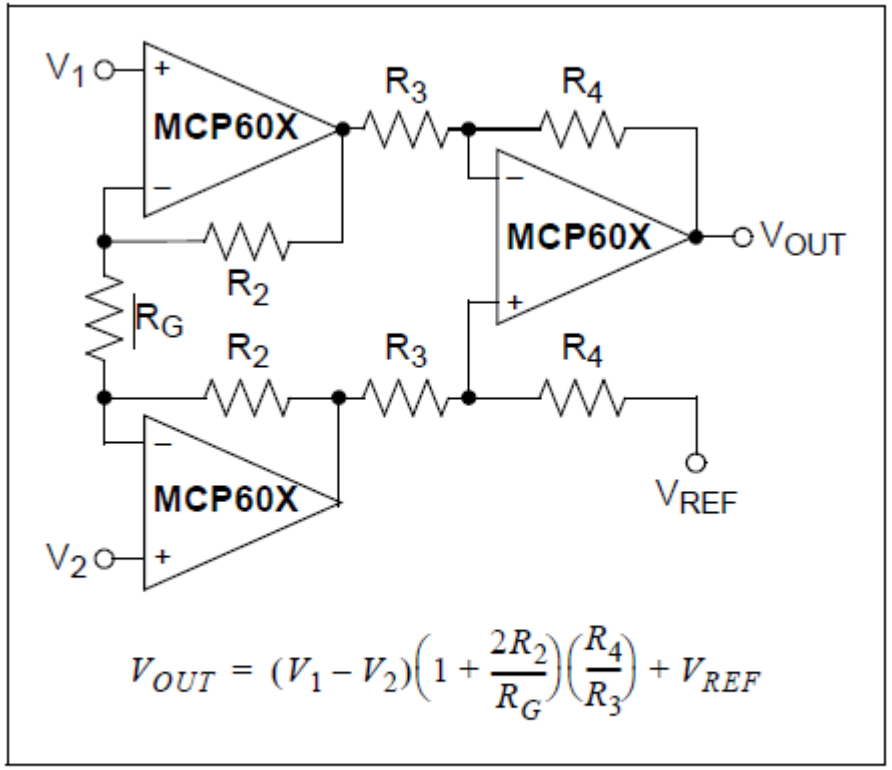
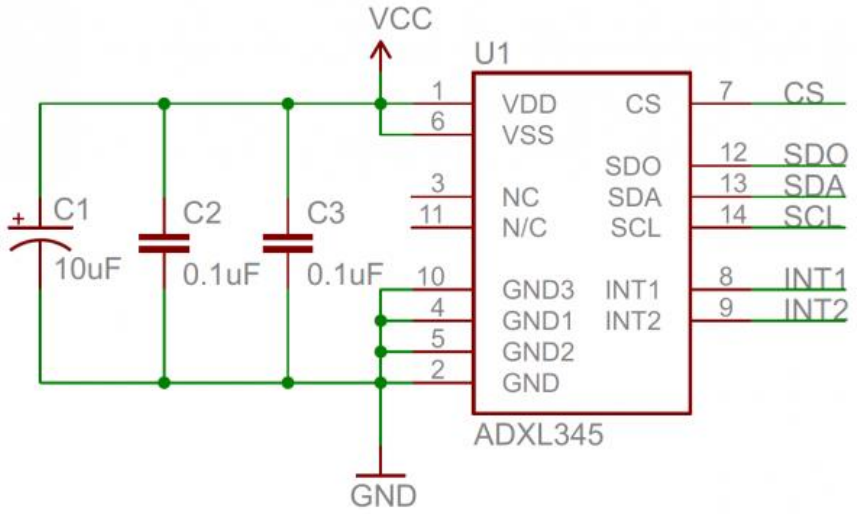


FIGURE 4-10: *Three-Op Amp Instrumentation Amplifier.*

Datablad MCP601

AVKOBLINGSKONDENSATOR



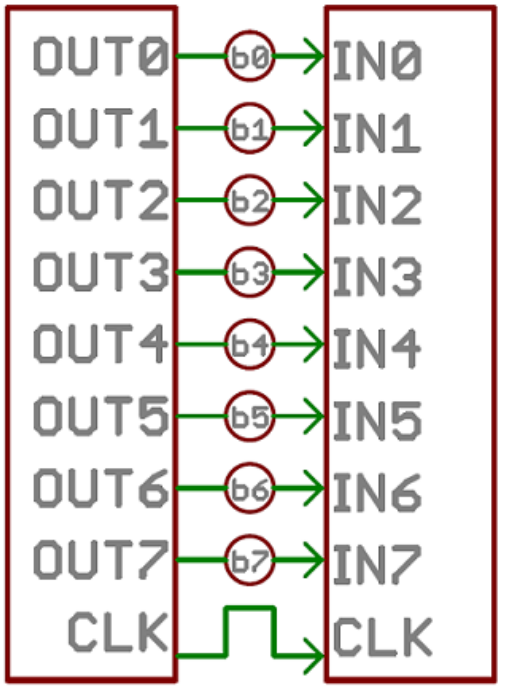
Små endringer i strømforsyning kan gi utslag i form av støy når vi holder på med sensitive komponenter.

- For å glatte ut strømmen benytter vi gjerne avkoblingskondensatorer «bypass/decoupling capacitor» rett ved strømforsyningspinnene til alle ICer.
- Størrelsen på avkoblingskondensator varierer.
  - Hurtigswitchende komponenter benytter gjerne små, raske kondensatorer omkring 100nF- 10uF
  - Strømforsyninger bruker typisk mye større kondensatorer.
- Ofte vil databladet til komponenten gi en pekepin på hva som er en fornuftig størrelse på avkoblingkondensatoren

https://learn.sparkfun.com/tutorials/capacitors/application-examples

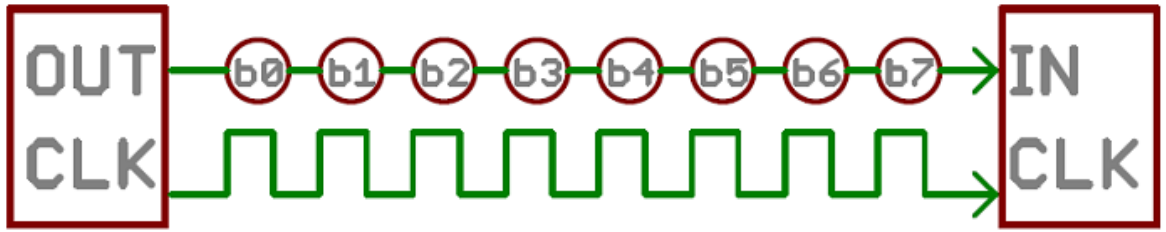
BUSSTERMINOLOGI

- 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



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

Seriell buss ↓ ,Parallell buss ↑



BUSSTYPER

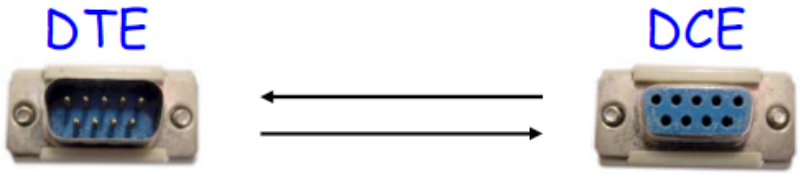
	Komponent til komponent	Kort til kort* (korte avstander)	Enhet til enhet*
Seriell	RS-232, SPI, I2C	RS232, SATA	RS485, USB1 og 2, Ethernet
Flere Serielle		SCSI, PCI-e	Displayport, HDMI, Thunderbolt, USB 3.x
Parallell	Memory BUS	(Parallellport), (PCI), (tidl SCSI)	

*inndelingen etter avstand kan variere

- Utviklingen har gått fra parallellkoblinger (tykke ledninger) til mer seriell kommunikasjon
- Differensiell seriell kommunikasjon tillater høyere hastighet på dataoverføring per linje.
- Parallelle busser brukes typisk internt i komponenter og f.eks når vi kobler ting selv.

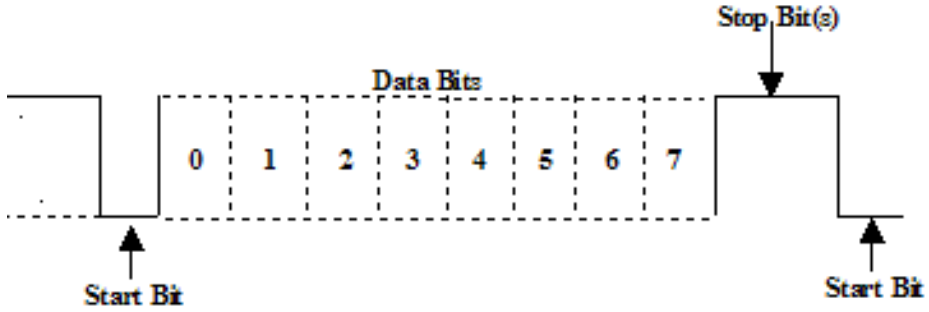
RS-232 SERIELL KOMMUNIKASJON

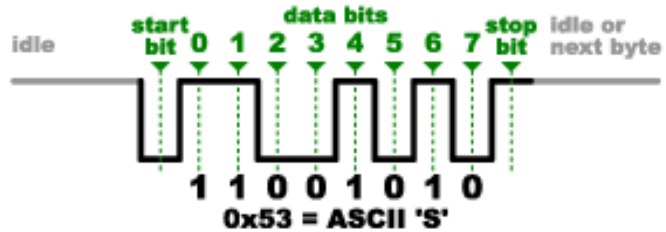
- Toveis seriell datakommunikasjon.
- Laget for kommunikasjon mellom 2 enheter (PC/terminal og modem)
- Kan kjøres halv duplex eller full duplex
- Mulighet for en rekke ekstra ledninger for handshake/ status, men I prinsippet brukes bare to pinner til kommunikasjon
- opprinnelig laget på 60 tallet beregnet 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



RS-232, OG UART PROTOKOLL

- For å overføre data med RS-232 bruker vi en 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).
- Hastigheten kalles «baud rate»
  - 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

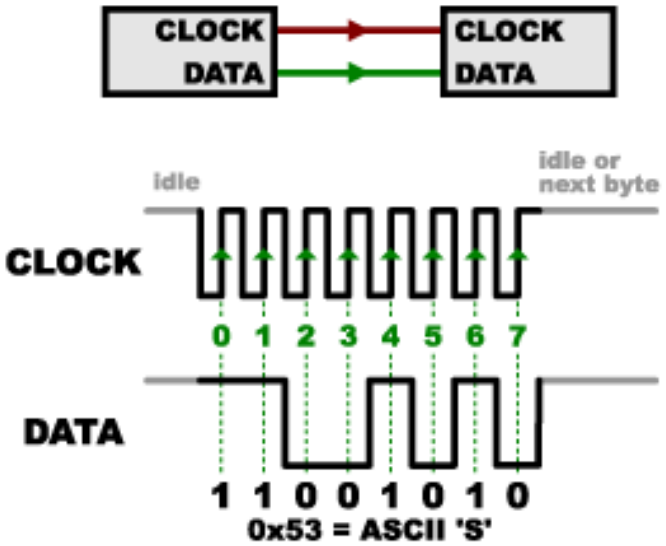




Mer/kilder:
https://learn.sparkfun.com/tutorials/serial-communication/all
https://learn.sparkfun.com/tutorials/terminal-basics/all

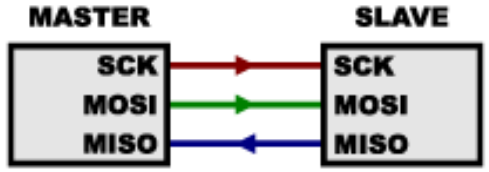
SPI – SERIAL PERIPHERIAL INTERFACE

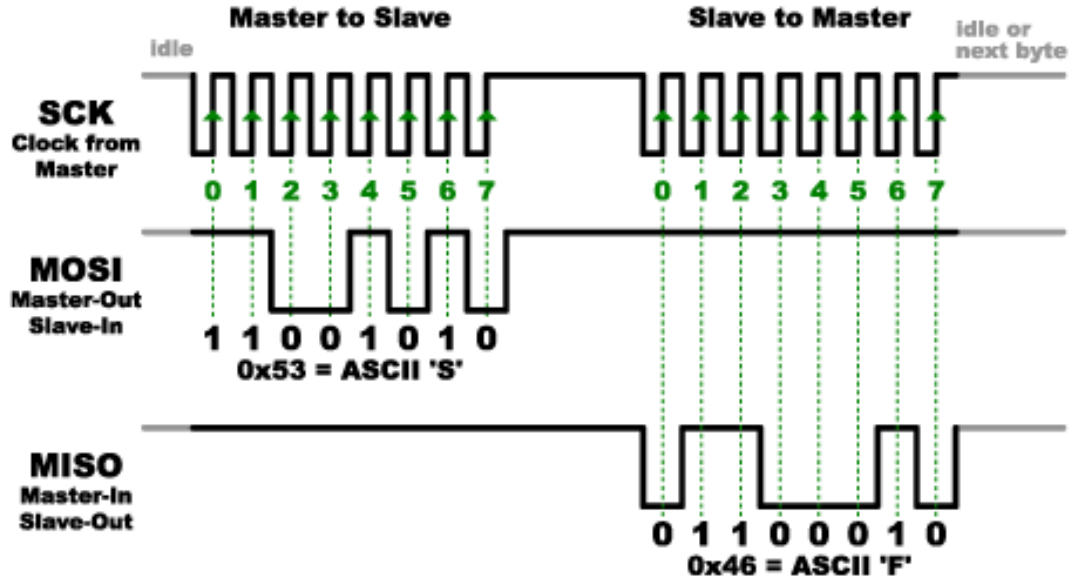
- Synkron, full duplex seriell dataoverføring
- En Master, en eller flere slaver.
- Klokke overføres via linje
  - slavene behøver ikke egen klokke
- Enklere enn RS-232.
  - Slave trenger omtrent bare et shift-register for å ta imot data
- Hastighet kbps- Mbps
- Terminologi:
  - SCK – Serial Clock (fra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

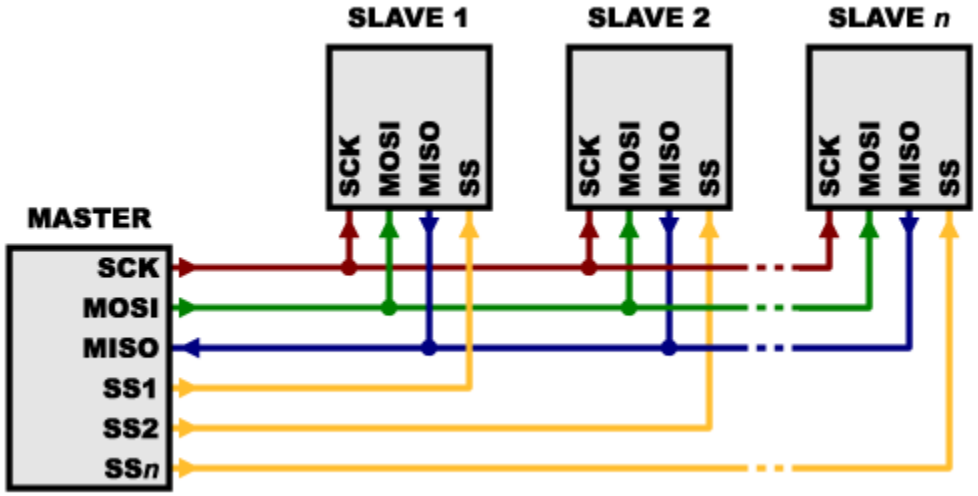


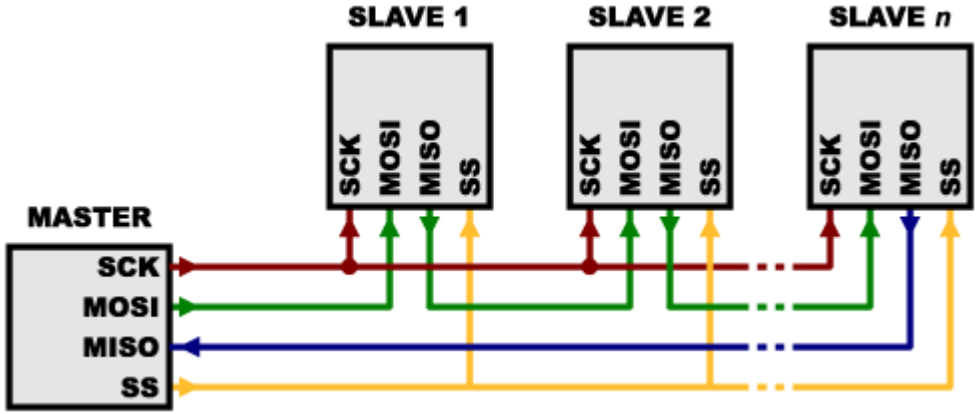
SPI- PROTOKOLL

- Master som klokker all dataoverføring, 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.



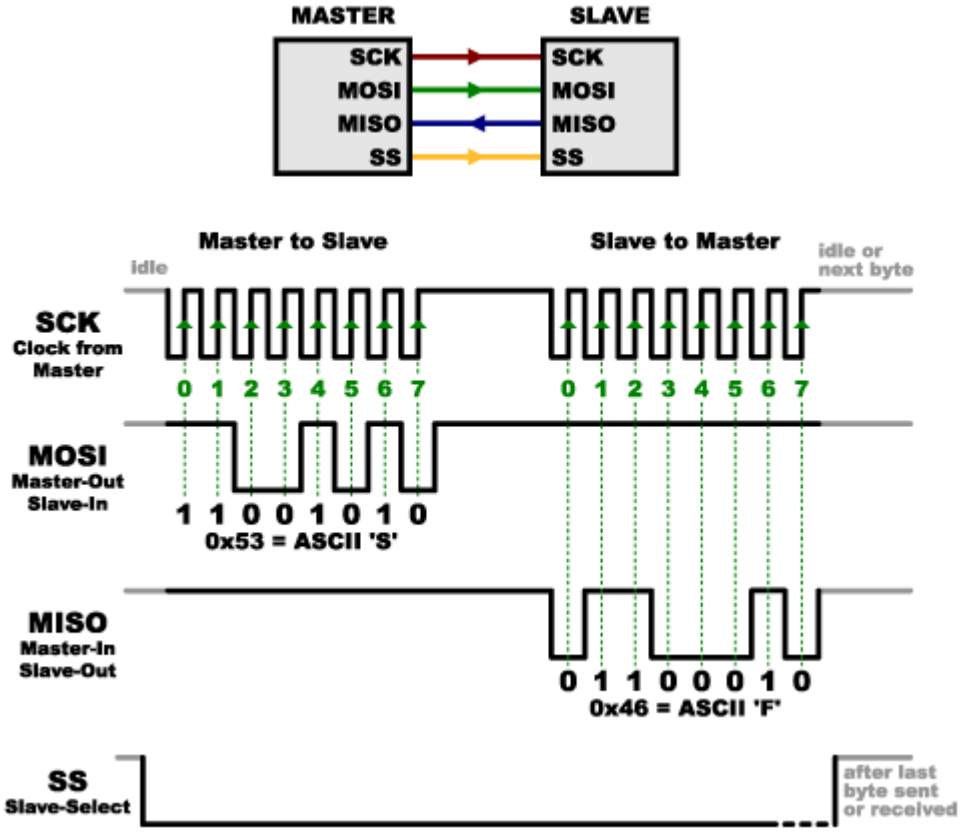


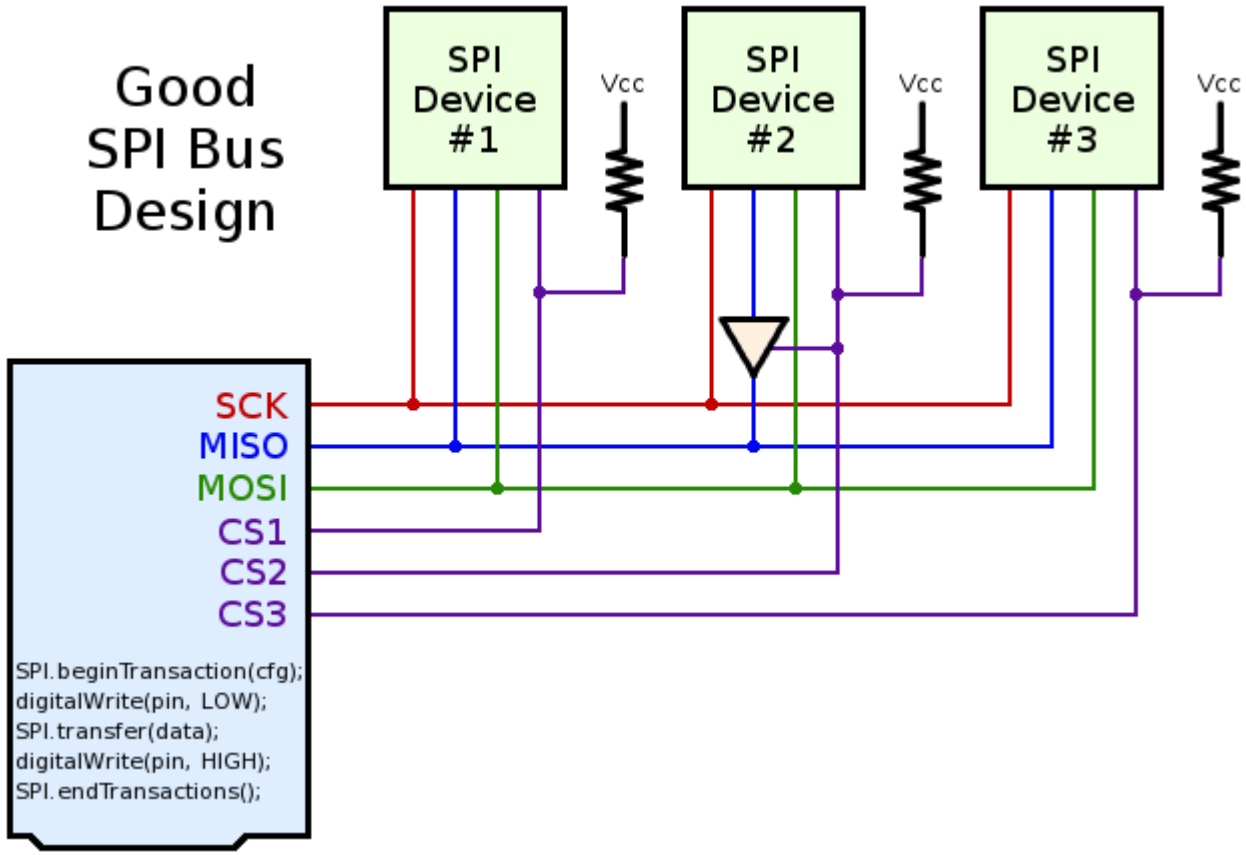




SPI kjedekobling: Merk kobling MOSI-MISO

SPI FORTS.



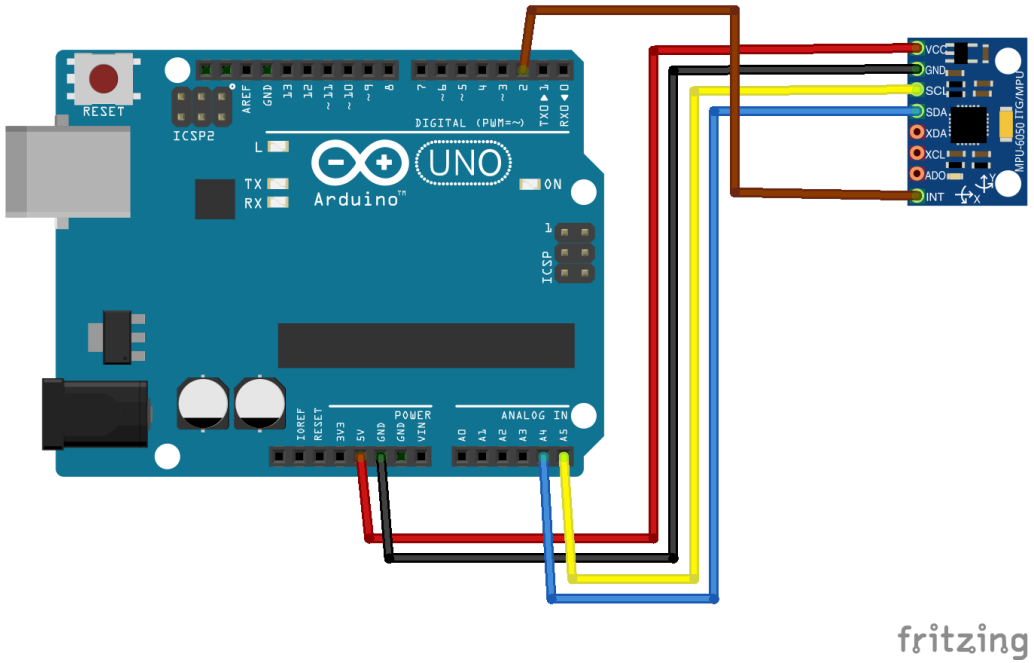


UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

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.»

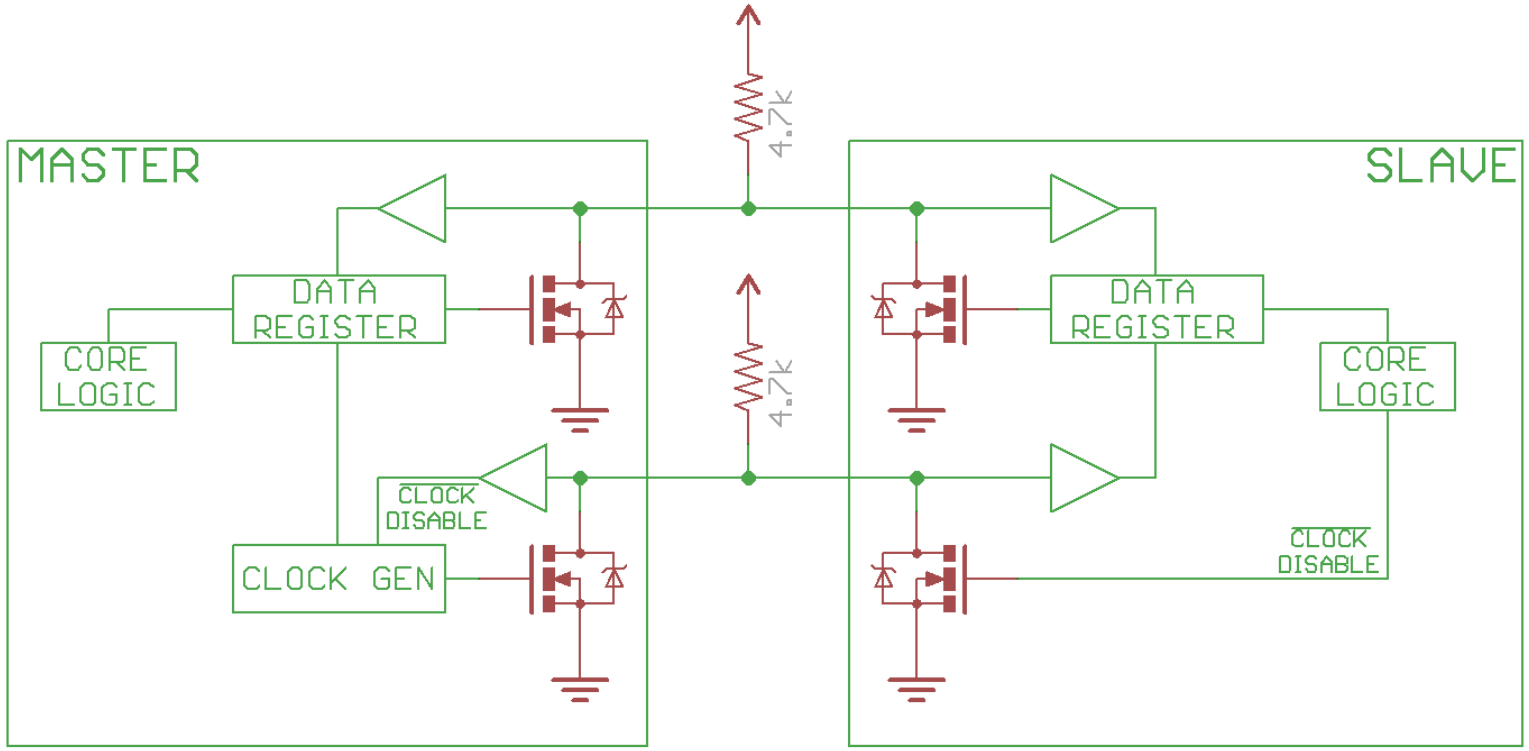
- I2C Bruker kun 2 linjer: SCL og SDA
- Kan ha svært mange enheter koblet på samme buss-
  - inntil 1008 slaver.
  - Flere enheter kan ha rollen som master
- Klokkehastighet på inntil 100kHz eller 400kHz
  - Raskere enn RS232,
  - langsommere enn SPI
- Mer kompleks logikk kreves sammenlignet med RS-232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



fritzing

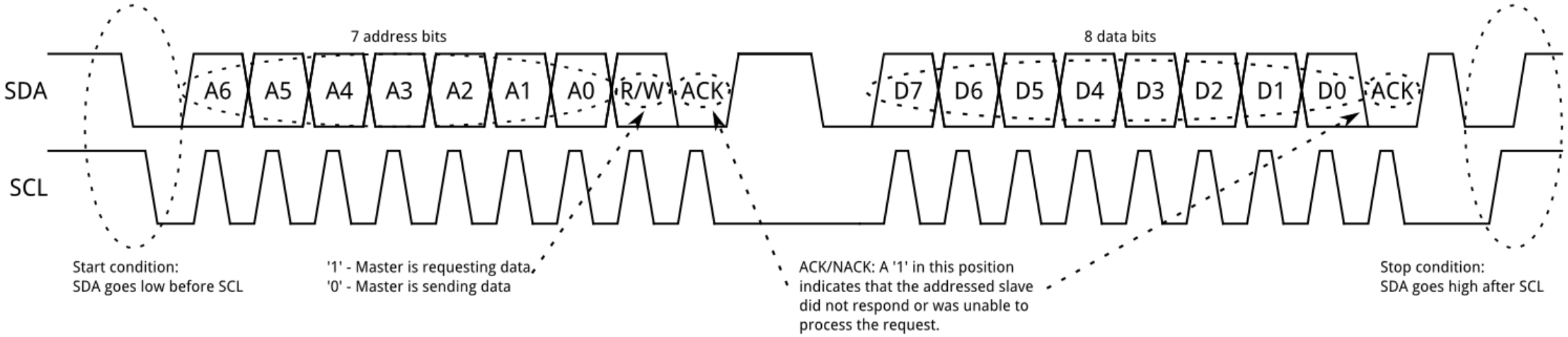
I²C - OPPSETT

- 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

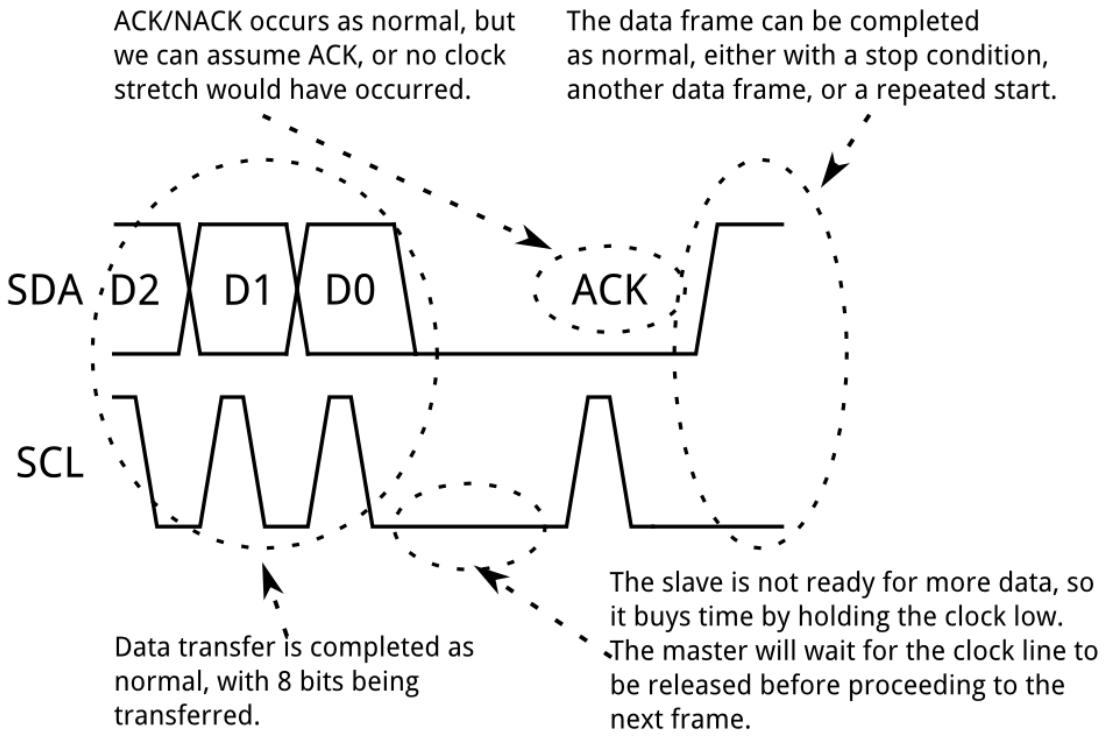


I2C: 10 BIT ADRESSER OG STREKKING AV KLOKKE.

- 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 signifikanete 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»)



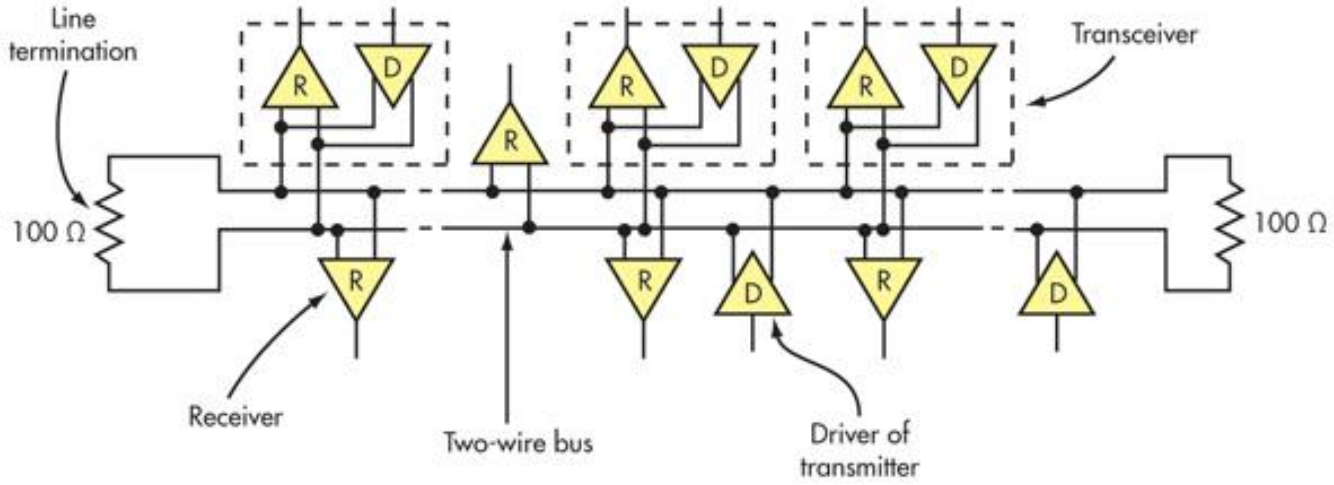
For detaljer, se
https://learn.sparkfun.com/tutorials/i2c

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, antall stoppbit hører RS-232

RS-485

- RS-485 er en fysisk standard laget for kommunikasjon over større avstander enn RS-232.
- Linjene drives differensielt, 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

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.
- 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:
  - HyperTerminal, Coolterm, Arduino Serial Monitor, TeraTerm, og så videre…

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