



4

Datamaskiner, data og nettverk

KOMPETANSEMÅL

I dette kapittelet utvikler du kompetansen din i følgende kompetansemål:

- > beskrive sentrale komponenter i datamaskiner og nettverk og gjøre rede for hvilken funksjon de har
- > gjøre rede for og vurdere hvordan ulike typer informasjon kan representeres digitalt og struktureres ved hjelp av ulike datalagringsmetoder

4.1 Datamaskiner

DIGITALT UTSTYR

Du er hele tiden omgitt av **digitalt utstyr**. Om morgenen blir du vekt av alarmen på mobiltelefonen din. På badet slår du på DAB-radioen. Du haster så av sted for å rekke bussen til skolen og kommer på at du glemte å låse døra. Fra bussen åpner du programmet som kommuniserer med dørlåsen og husalarmen og trykker på knappen som låser døra og slår på alarmen. På resten av turen sjekker du kanskje sosiale medier, spiller spill eller ser et videoklipp. Vel fremme på skolen kjøper du deg en yoghurt i kantina. I kassa skannes strekkoden på yoghurtbegeret av en strekkodeleser som er koblet til kasseapparatet. Du betaler yoghurten med Vipps-appen på mobiltelefonen din, og kassadama sier at betalingen er mottatt. I løpet av en time har du brukt mange forskjellige digitale apparater, kanskje uten at du har tenkt spesielt over akkurat det.

En datamaskin brukes kort fortalt til å utføre beregninger og til å behandle det man gir den (input), slik at den gir tilbake noe man kan bruke (output). Stasjonære datamaskiner, bærbare datamaskiner, mobiltelefoner og nettbrett er alle laget med de samme interne hovedkomponentene. Prosessor, minne, harddisk, grafikkort, lydkort og strømforsyning utgjør sammen med hovedkortet og kabinettet det vi kjenner som en datamaskin. USB-porter og trådløs kommunikasjon (wifi, blåttann) er også vanlige komponenter.

Datamaskin og PC eller computer. Et kan være sta (desktop) eller (laptop). Et r og telefoner datamaskine

En datamaskin
ulike komponenter:
venstre øver:
(CPU), minne
harddisk, gra
strømkilde o

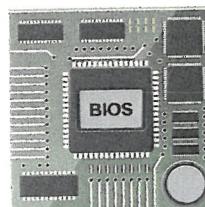
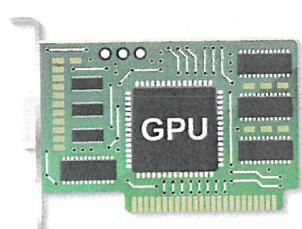
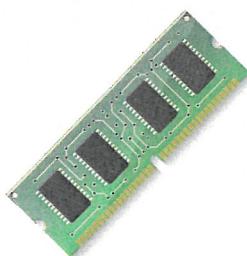
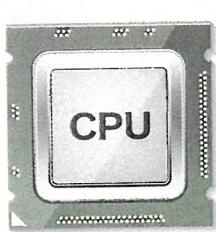
Datamaskinen kalles også PC eller personal computer. En datamaskin kan være stasjonær (desktop) eller bærbar (laptop). Et nettnett og telefoner er også datamaskiner.



Datamaskiner bygges opp av hovedkomponentene prosessor, harddisk, grafikkort, lydkort og strømforsyning. Hovedkomponentene kobles til et hovedkort og plasseres inni et kabinett.

Nesten alt digitalt utstyr fungerer på den samme måten. Når vi slår på strømmen til en datamaskin, starter et dataprogram som heter **BIOS**, Basic Input Output System. Det er et spesielt program som ligger lagret på hovedkortet. BIOS hjelper hovedkortet med å starte opp og få kontakt med prosessoren, minnebrikkene for hurtig-

En datamaskin består av ulike komponenter. Fra venstre øverst: prosessor (CPU), minnebrikke, harddisk, grafikkort, strømkilde og hovedkort.



minne, harddisken, skjermen, tastaturet og andre enheter i datamaskinen. BIOS setter også i gang det programmet som kalles **operativsystemet**, som ligger lagret på harddisken. Operativsystemet kan være ulike programmer, som Windows, Unix, Linux og Apple OS. Når operativsystemet er startet opp, overtar det kontrollen fra BIOS.

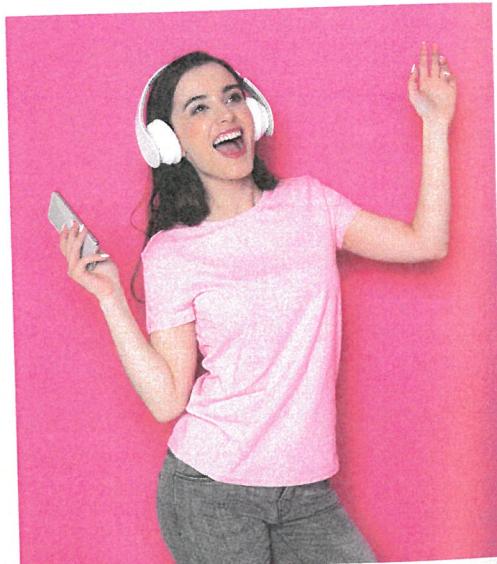
Operativsystemet har som oppgave å kontrollere og styre kommunikasjonen mellom enhetene i datamaskinen og programmene som kjører. Operativsystemet sørger for at programmene får tilgang til enhetene, og gjør kommunikasjonen sikker. Det gjør det også enklere for programmene å kommunisere i tur og orden og på riktig måte seg i mellom, og med enhetene. Egne programmer som heter **drivere**, sørger for at et operativsystem kan snakke med en enhet, for eksempel nettverkskortet. Derfor må man ofte ha forskjellige drivere for forskjellige versjoner av operativsystemet.

EKSEMPEL

Blåtann

En del digitalt utstyr, som skjerm og tastatur, kan kommunisere trådløst med hverandre ved hjelp av det som heter **blåtann** (bluetooth). Vi bruker ofte blåtann når vi skal koble trådløst utstyr til en datamaskin, eller når vi skal koble en mobiltelefon til håndfri systemet i bilen. Rekkevidden til en blåtannsender er ca. ti meter, og overføringskapasiteten (dataraten) til blåtann er på maksimalt to Mbit per sekund. Dermed er blåtann godt egnet til bærbart utstyr, for eksempel hodetelefoner, og til å sende filer fra en datamaskin til en annen i nærheten.

Mobiltelefonen
kommuniserer trådløst
med høretelefonene ved
hjelp av blåtann.



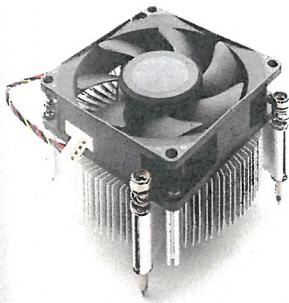
Kjøleribbe
plasseres c
prosessorer
at den ove

HOVEDKORT

Hvis du har sett inni en datamaskin, har du antakelig lagt merke til det store kretskortet som de andre komponentene er montert på eller koblet til med kabler. Hovedkortets oppgave er å sørge for kommunikasjon mellom de andre komponentene. Det er nødvendig for at maskinen skal fungere.

Interne kontakter (utvidelsesspor) på hovedkortet kobler sammen prosessor, minne, lagring, strøm og PCI Express. Prosessoren er festet rett på hovedkortet og har en kjøleribbe med vifte montert oppå for å unngå overoppheating. Minnebrikker og PCI Express-kort festes i egne spor på hovedkortet, mens strømforsyningen kobles til med en kabel.

Hovedkortet i en mobiltelefon.



Kjøleribbe og vifte plasseres ofte oppå prosessoren for å unngå at den overopphetes..

Eksterne kontakter på hovedkortet er tilgjengelige utenfra når kabinetet er lukket. Dette er vanligvis USB-kontakter, kontakter for mus/tastatur (PS/2), skjerm (HDMI, VGA, DVI, Display Port), lyd og nettverk.

DISKUTER

Hvilke eksterne kontakter har dere på datamaskinene deres?

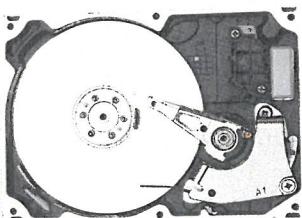
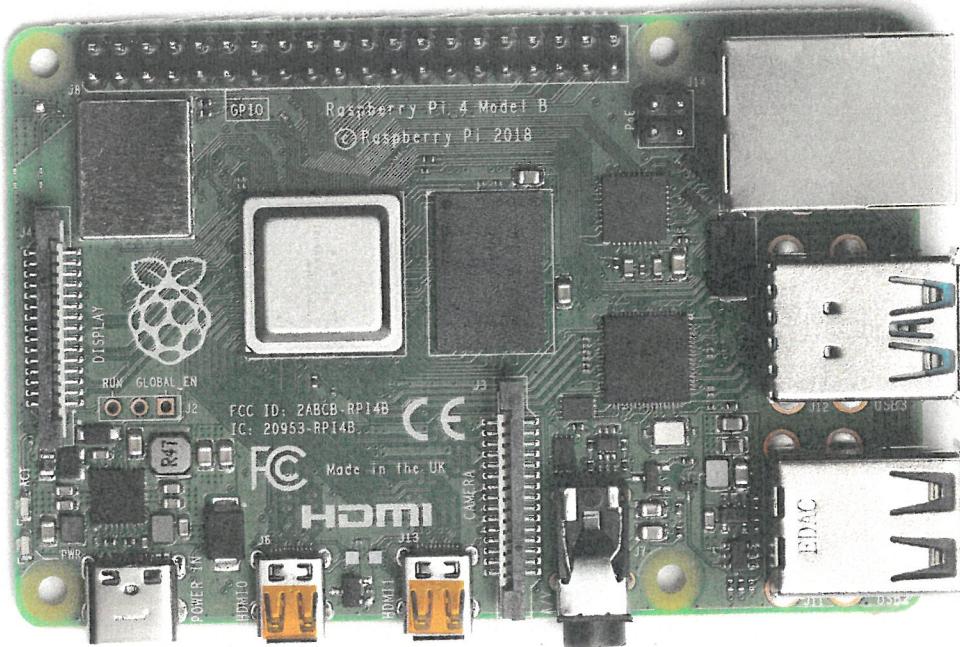
PROSESSOR (CPU)

Prosessoren, også kalt CPU (*Central Processing Unit*), styrer og utfører instruksjoner fra programmene som kjører. Den omtales derfor ofte som «hjernen» til datamaskinen. De fleste instruksjonene utfører prosessoren selv, men den kan delegerere andre spesiatiserte oppgaver til mer egnede komponenter, for eksempel grafikkortet. Prosessoren har et lite, men svært raskt tilgjengelig innebygget minnelager (cache). Der ligger en kø med ventende oppgaver så lenge prosessoren er opptatt.

Vanlige prosessorer inneholder to til åtte kjerner som alle kan utføre beregninger uavhengig av hverandre.

Tidligere ble gjerne minne, USB og grafikk håndtert i egne fysiske brikker på hovedkortet, men disse oppgavene er nå i hovedsak flyttet inn i prosessoren. Særlig i mobile enheter blir det mer vanlig med **SoC** (*System on a Chip*). Da samles flere sentrale kontrollbrikker i prosessoren og færre styringsbrikker på hovedkortet. Da sparer man plass, og datamaskinen bruker mindre strøm til overføring av signaler mellom komponentene. Dette er noe av teknologien som gjør det mulig å lage smarttelefoner med større regnekapasitet enn det stasjonære-PC-er hadde for 15 år siden, og med en brøkdel av strømforbruket.

Raspberry Pi er en datamaskin på størrelse med et bankkort. Da er alle komponentene samlet på ett kort (SoC).



HDD/magnetdisk. Data lagres på disken ved å magnetisere deler av disken. Disken roterer flere tusen ganger i minuttet.

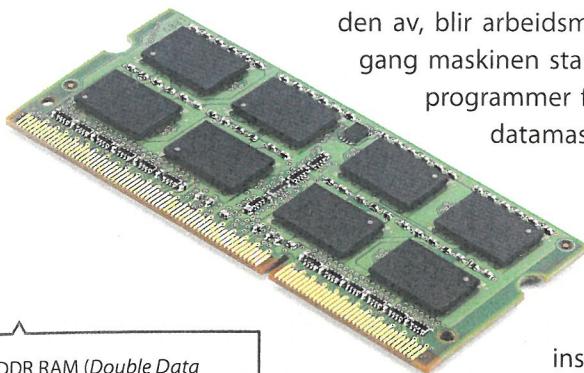
LAGRING

Harddisk brukes ofte som en fellesbetegnelse på den eller de komponentene i en datamaskin som kan lagre informasjon (filer) over tid. Da stasjonære maskiner og klumpete laptoper utgjorde det man kjente som datamaskiner, var magnetdisker (HDD – *Hard Disk Drive*) enerådende på grunn av overlegen lagringskapasitet, overføringshastighet og stabilitet. Etter hvert modnet teknologien som brukes i minnekort. Da ble SSD-teknologien (*Solid State Drive*) videreutviklet, og vanlige datamaskiner kan utnytte fordelene med høyere skrive- og lesehastigheter. I moderne datamaskiner kan nå begge typer kobles til SATA-portene (Serial Advanced Technology Attachment) på hovedkortet.

SSD gjør det mulig å lagre store mengder informasjon på små minnebrikker med et veldig lavt strømforbruk, og teknologien brukes i mange laptoper og alle mobile enheter. Tradisjonelle magnetdisker er likevel fortsatt utbredt i stasjonære datamaskiner og servere, siden HDD fortsatt er billigere per gigabyte lagringsplass og tåler flere lese- og skriveoperasjoner.

ARBEIDSMINNE (RAM)

Kode (programmer) og filer som kjøres på maskinen, blir liggende i arbeidsminnet mens de er i bruk. RAM (*Random Access Memory*) er normalt ikke større enn noen gigabyte, men det arbeider mange ganger raskere enn en harddisk og er stort nok til å lagre de viktigste filene det arbeider med. Arbeidsminnet er delt opp i en rekke «plasser» hvor lesing og skriving kan gjøres uten å påvirke dataene i de omliggende plassene. Dataene deles ofte opp over flere plasser i minnet uavhengig av hverandre. På samme måte som en kalkulator «glemmer» tallene man sist brukte, når man slår



DDR RAM (Double Data Rate Random Access Memory)

den av, blir arbeidsminnet i en datamaskin tømt når man slår av maskinen. Hver gang maskinen startes opp, må den derfor hente inn operativsystemet og kjøre programmer fra harddisken på nytt. Arbeidsminnet kan derfor sies å være datamaskinens «korttidsminne».

GRAFIKKORT

I alle datamaskiner som viser informasjon på en skjerm, fins det en form for grafikkort som effektivt tolker og behandler instrukser fra maskinen og omformer dem til et bilde som kan vises på skjermen. Prosessoren i maskinen er utstyrt for å kunne behandle et bredt spekter av instrukser og koder etter hverandre. Styrken til grafikkprosessoren ligger i å kunne behandle et stort antall skjerm- og bilderelaterte prosesser samtidig. Grafikkprosessoren inneholder flere kjerner, i likhet med datamaskinens prosessor. Selv om hastigheten til hver enkelt kjerne i en grafikkprosessor er betydelig lavere, er antallet grafikkprosessorer mye høyere. Dermed kan grafikkprosessoren behandle flere instrukser parallelt. I stasjonære maskiner er grafikkortet ofte et eget kort koblet til en PCIe-ports på hovedkortet, mens bærbare og mobile enheter i hovedsak har grafikkortet integrert i prosessoren.

NETTVERSKORT

For å kunne kommunisere med omverdenen er det nødvendig med et nettverkskort. Tradisjonelle nettverkskort var fysiske kort i et utvidelsesspor på hovedkortet. De var koblet med en nettverkskabel til en nettverkskontakt på veggen eller et modem, slik at man fikk tilgang til et nettverk. I dag har trådløse nettverk (wifi) i stor grad overtatt. Mange bærbare og mobile enheter kan kun koble seg til nettet trådløst. Samlebetegnelsen nettverkskort brukes om flere ulike grensesnitt som gjør at man kan koble seg til et nettverk. Ethernet (nettverkskontakt), wifi, mobilt Internett (4G/5G) og blåtann er forskjellige måter å koble seg til nettverk på. Du lærer mer om nettverk i delkapittel 4.7.

USB-PORT

USB (*Universal Serial Bus*) kom på slutten av 1990-årene. Målet var å erstatte tidligere varianter av kontakter, kabler, protokoller og strømforsyning på enheter man koblet til datamaskiner, med nye standarder som var enklere å bruke. For eksempel er seriellport, parallelport og PS/2 (tastatur/mus) i praksis erstattet av USB. På en datamaskin er gjerne mus og tastatur koblet til med USB, sammen med minnepinner, eksterne harddisker og kanskje et kamera og et trådløst nettverkskort. USB-C-kontakten (som brukes til lading på mange nye mobile enheter), sammen med USB 3.1, den nyeste standarden, og gjør det mulig å bruke USB-kabelen for å koble datamaskinen til en skjerm. Det er også vanlig at produsenter av datamaskiner bruker interne USB-grensesnitt i bærbare maskiner for å koble til webkamera og fingeravtrykksleser.

4.2 Binære tall – bit og byte

BIT

De aller første datamaskinene var mekaniske og kunne bestå av muttere, tannhjul og metallstenger. I overgangen fra mekanisk til elektronisk datamaskin fant man ut at den enkleste og mest pålitelige måten å **lagre informasjon** på er å ha to «tilstander»: strømmen av og strømmen på. Var strømmen i av-tilstanden, skulle det stå for tallet 0, og var strømmen i på-tilstanden, representerte det tallet 1. Hvordan kan vi så lagre noe særlig av verdi tallene 0 og 1? La oss se på et eksempel. Du har en hytte med en vimpel i flaggstangen. Naboene dine vet at hvis vimpelen er oppe, så er du på hytta. Ellers er du der ikke. På denne enkle måten kan naboene se om noen er på hytta. Da har du allerede gitt viktig informasjon ved hjelp av en enkel **av/på-tilstand** eller en **binær tilstand**.

En elektronisk datamaskin sjekker om tilstanden er av eller på, ved hjelp av strøm.



Hvordan kan vi signalisere noe av verdi ved hjelp av bare to tilstander, av eller på?

La oss så si at du har to utelamper på hytta di, en utenfor hoveddøra og en utenfor bakdøra. Så lager du et lite system med tanke på naboene dine. Hvis begge lampene er av, er du ikke på hytta. Hvis begge er på, så er du inne. Med to lamper kan du gjøre dette mer avansert. For eksempel kan du ha det slik at hvis lampa ved hoveddøra er på mens lampa ved bakdøra er av, viser det at du bare er ute en liten tur. Mens hvis lampa ved hoveddøra er av mens den ved bakdøra er på, kommer du ikke tilbake før dagen etter.

Allerede her blir det litt komplisert. Med to lamper har du fire muligheter. Lampene kan representeres som kalles bit i en datamaskin. Her har vi to lamper i av- eller

Det binære tallet 10 leser vi som én-null, ikke som ti. Og 11 leser vi som én-én, ikke som elleve.

på-tilstand. **To bit** gir fire muligheter: på-på, av-av, på-av og av-på. Hvis vi nå bruker tallet 1 for på og tallet 0 for av, får vi 11, 00, 10 og 01. Dette kaller vi **binære tall**. Hvis du øker til tre utelamper på hytta, får du åtte muligheter: 000, 001, 010, 100, 011, 101, 110, 111. Når du har tre utelamper, sier vi i at du har et system basert på **tre bit**.

To utelamper	Hoveddør-Bakdør	To bit Binært tall	Beskjed
	På-På	11	Inne på hytta
	Av-Av	00	Ikke på hytta
	På-Av	10	Ute en kort tur
	Av-På	01	Ute, tilbake etter ett døgn

Vi bruker binære tall, 1 og 0 for tilstandene av og på i en datamaskin.

To bit gir fire muligheter: 11, 00, 10, 01

Tre bit gir åtte muligheter: 000, 001, 010, 100, 011, 101, 110, 111

EKSEMPEL

Tosifrede binære tall til vanlige tall

Vi kan erstatte de forskjellige kombinasjonene av 0 og 1, altså de binære tallene, med vanlige tall. Har vi to bit, for eksempel 11, gir vi ett-tallet på plassen lengst til venstre verdien 2, og plassen til høyre får verdien 1. Er de av, altså 0, får begge plassene verdien null. Med to utelamper kan vi da telle til tre med vanlige tall. Det gjør vi slik: 00 gir $0 + 0 = 0$. Mens 01 gir $0 + 1$ altså 1. Ser vi på 10, så har sifferet 1 verdien 2 siden det står først, og det gir $2 + 0 = 2$. Tilsvarende blir $11 = 2 + 1 = 3$ siden begge er slått på og den første har verdien 2 og den neste verdien 1.

Binære tall kalles totallsystemet.
Vanlige tall kalles titallsystemet.

To bit Binært tall	Verdi: 2	Verdi: 1	Sum, vanlig tall
00	0	0	0
01	0	1	1
10	1	0	2
11	1	1	3

Som du ser, kan du regne deg frem til summen et binært tall har, ved å legge sammen verdiene ut fra om det står 1 eller 0, og hvilken plassering tallet da har.

EKSEMPEL**Tresifrede binære tall til vanlige tall**

For tre bit får ett-tallet som står først, verdi 4, den i midten får verdien 2, og den siste får verdien 1. Null får alltid verdien 0. Med andre ord er: $101 = 4 + 0 + 1 = 5$. Alle de forskjellige kombinasjonene til tre bit blir da slik:

Tre bit Binært tall	Verdi: 4	Verdi: 2	Verdi: 1	Sum, vanlig tall
000	0	0	0	0
001	0	0	1	1
010	0	1	0	2
011	0	1	1	3
100	1	0	0	4
101	1	0	1	5
110	1	1	0	6
111	1	1	1	7

Legg merke til at vi med tre bit har åtte ulike muligheter, men vi kan bare telle til sju. I dataverdenen tar vi stort sett alltid med 0 som en av mulighetene. Dette kan virke litt forvirrende siden vi i dagligtale stort sett ikke regner med 0 når vi teller. Hvis det er 25 elever i en klasse, kan vi nummerere elevene fra 1 til 25. Elev nummer 25 får tallet 25. En datamaskin vil helst nummerere dem fra 0 til 24.

ANTALL MULIGHETER**DISKUTER**

Ser du mønsteret mellom antall bit og antall muligheter?
Hvor mange muligheter har vi med 8 bit?

Fins det så noen enkel metode for å finne ut hvor mange **muligheter** vi har ved en gitt mengde av/på-tilstander, eller en gitt mengde bit? Vi fant ut at to utelamper gir fire muligheter, og at tre utelamper gir åtte muligheter. Du kan regne ut hvor mange muligheter du har, ved å ta tallet 2 og opphøye det i antallet bit. Med 2 bit blir antallet muligheter 2^2 . Det er det samme som $2 \cdot 2$, som altså gir 4. For 3 bit blir det da 2^3 , eller $2 \cdot 2 \cdot 2 = 8$. Fire bit gir 2^4 , som er 16 muligheter. Fem bit gir 2^5 , som er 32 muligheter, og slik fortsetter det. Disse tallene går ofte igjen i dataverdenen. Tabellen viser antall muligheter opp til 10 bit.

Bit	1	2	3	4	5	6	7	8	9	10
Muligheter	2	4	8	16	32	64	128	256	512	1024

Antall muligheter ved to bit finner vi slik: $2^2 = 2 \cdot 2 = 4$
 Antall muligheter ved tre bit finner vi slik: $2^3 = 2 \cdot 2 \cdot 2 = 8$
 Antall muligheter ved n bit finner vi ved å regne ut potensen 2^n

8 BIT ER EN BYTE

Øker du til flere bit som du så skal finne tallet til, dobler du verdien til hver bit. I eksempelet på forrige side hadde vi tre bit, og den høyeste verdien er 4. Hvis vi har fire bit, vil den neste biten, som legges på lengst til venstre, få verdien 8. Hva blir da det binære tallet 1101 omregnet til vanlige tall? Jo, her får vi disse verdiene bortover: 8, 4, 2, 1. Tre av de fire bitene her er «slått på», nemlig 8, 4 og 1. Så $1101 = 8 + 4 + 0 + 1 = 13$. Vi skal se litt nærmere på dette systemet og øker nå til **8 bit**. Her får vi følgende verdier bortover: 128, 64, 32, 16, 8, 4, 2, 1. Hva blir da det binære tallet 1011 1101 omregnet til vanlige tall? Seks av de åtte bitene er «slått på», så $1011 1101 = 128 + 0 + 32 + 16 + 8 + 4 + 0 + 1 = 189$.

Verdiene finner vi slik:
 $2^7, 2^6, 2^5, 2^4, 2^3, 2^2, 2^1, 2^0$

For å vise at 1110 er et binært tall, kan vi skrive det slik: 1101
 101112

Det du nå har regnet på, altså 8 bit, er det som ofte bare blir kalt **1 byte**. I mange datamaskiner er 8 bit, altså 1 byte, den minste plassen du kan sette av til bruk i minnet. Med 8 bit får vi 2 opphoyd i 8 (2^8), altså 256 mulighetene som én byte gir, er et tall som ofte er stort nok til å beskrive en farge, et lydnivå eller til å angi et tegn i et tegnsett.

8 bit blir kalt 1 byte. I mange datamaskiner er 1 byte den minste plassen du kan sette av til bruk i minnet. De 256 mulighetene som én byte gir, beskriver ofte en farge, et lydnivå eller et tegn i et tegnsett i en datamaskin.

EKSEMPEL

Vanlige tall til binære tall

Når vi skal regne om fra et vanlig tall til et binært tall, må vi først finne den høyeste verdien som er i tallet. Vi har det vanlige tallet 112 og skal skrive det som 8 bit.
 Verdiene for plassen til 1 i 8 bit er: 128, 64, 32, 16, 8, 4, 2, 1. Først finner vi ut hvor vi skal sette 1. I det vanlige tallet 112 er verdien 64 det største tallet, og vi setter 1 her.

Da har vi igjen $112 - 64 = 48$. Tallet 48 er større enn 32, og da setter vi 1 på 32-plassen. Så har vi igjen $112 - 64 - 32 = 16$. Da setter vi 1 på plassen til verdien til 16. Dermed har vi 0 igjen, og vi setter 0 på 8-, 4-, 2- og 1-plassen. Vi setter også 0 på plassen til verdien 128. Det binære tallet blir da: 01110000.

Verdi: 128	Verdi: 64	Verdi: 32	Verdi: 16	Verdi: 8	Verdi: 4	Verdi: 2	Verdi: 1
1	1	1					
0				0	0	0	0

LAGRINGSPLASS

Binært tallsystem:
1 kilobyte = 2^{10} byte =
1024 byte

Størrelsen på **lagringsplassen** i minnet og på en disk blir vanligvis oppgitt i byte, og da med et prefiks som angir hvor mange tusen eller millioner byte det er snakk om. Kilo, mega og giga er vanlige prefikser. En kilobyte er ca. 1000 byte (se margen). Det er ikke alltid helt forutsigbart når vi bruker betegnelsen bit, og når vi bruker byte. Men en tommelfingerregel er at lagringsplass oppgir vi i byte, og hastigheten i nettverk oppgir vi i bit, dvs. antallet bit per sekund.

Byte	Bit	Binært tall
Én byte	8 bit	11001100
To byte	To 8 bit	11001100 11001111
Tre byte	Tre 8 bit	11001100 11001111 11001101
Én kilobyte = 1000 byte	Tusen 8 bit	11001100 ...999 ganger
Én megabyte = 1000 000 byte	Én million 8 bit	11001100 ...999 999 ganger
Én gigabyte = 1000 000 000 byte	Én milliard 8 bit	11001100 ...999 999 999 ganger

Lagringsplass oppgis i byte, mens hastigheten på nettverk oppgir vi i bit per sekund.

EKSEMPEL

Minne

Tallene 2, 4, 8, 16, 32, 64, 128, 256, 512 og 1024 kjenner vi lett igjen fra tidligere digitale datamaskiner. I 1980-årene kom den legendariske hjemmedatamaskinen Commodore 64 på markedet. Tallet 64 forteller oss at den hadde 64 kB minne, nærmere bestemt 65 536 byte, som utgjør 524 288 bit. Minnekortet hadde altså 524 288 små

transistorer som kunne ha to strømtilstander: på eller av. I moderne mobiltelefoner er det heller ikke tilfeldig at de ofte kommer med et minne på 32, 64, 128 eller 256 gigabyte. Utgangspunktet er nå som før hvor mange bit det er tale om.

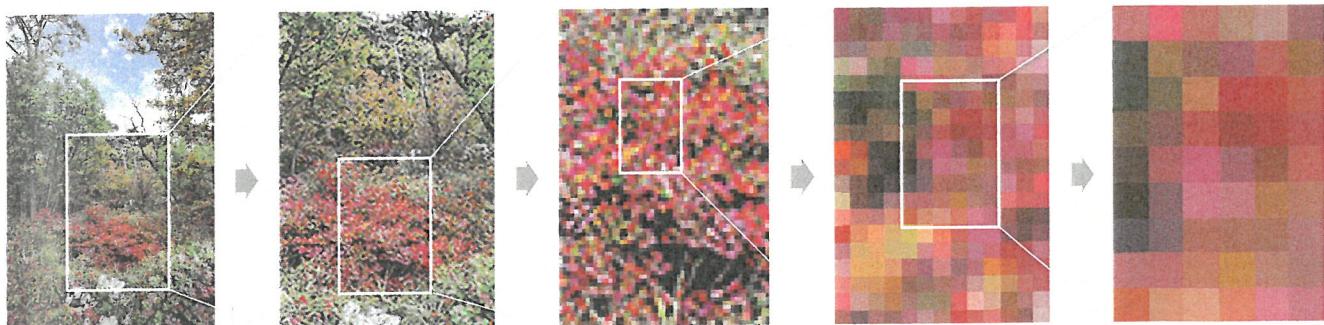
Commodore 64 hadde
64 kB minne.



4.3 Bilder består av bildepunkter

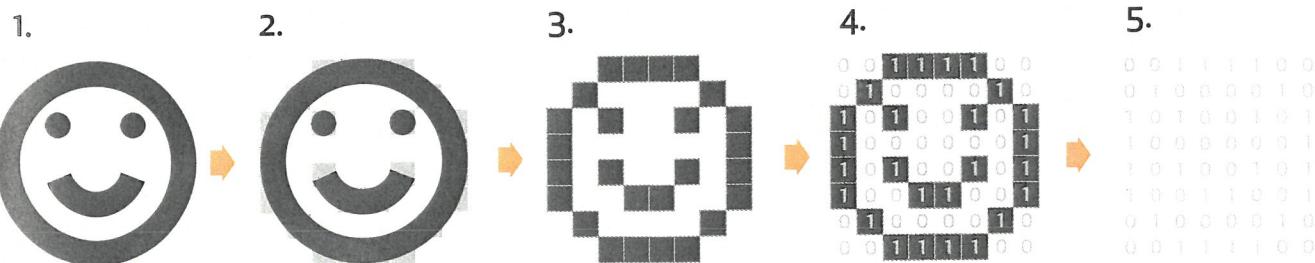
BILDEPUNKTER

Et digitalt bilde danner et godt utgangspunkt når vi skal se mer på hvordan en datamaskin fungerer. Selve bildet er tydelig og forståelig, selv om teknikken inne i maskinen kan virke komplisert. Som for det meste i en datamaskin gjelder det å bryte ting helt ned til 0 og 1. Derfor deler vi bildet inn i bitte små deler; jo flere, desto bedre. Disse delene kalles **piksler** eller **bildepunkter**. Disse bildepunktene har ikke noen fast eller standardisert størrelse. Det blir slik at jo flere punkter du deler et bilde inn i, desto høyere blir oppløsningen og dermed kvaliteten på bildet. Men da krever det også mer plass i minnet på datamaskinen.



Et bilde som er digitalisert og består av mange piksler, eller bildepunkter.

For å vise hvordan vi kan gjøre bilder og illustrasjoner om til bildepunkter, og hvordan det foregår i en datamaskin, bruker vi en enkel figur av et smilefjes, se nedenfor. Smilefjeset har tydelige, svarte streker. Det er to nivåer: svart og hvitt. Figuren av smilefjeset blir så digitalisert ved at et kamera eller en skanner leser den av og deler den inn i bildepunkter, figur 2 nedenfor. Vi bruker lav oppløsning i dette eksempelet. Maskinen plasserer de definerte bildepunktene så godt den kan. Når vi så tar bort det originale bildet, står vi igjen med en lavoppløst digital versjon, figur 3. Siden dette er et svært enkelt bilde med enten svart eller hvitt, er det ganske enkelt for datamaskinen å sette opp tall for hvert bildepunkt. Tallet 1 står for svart, og tallet 0 står for hvitt, figur 4.



Datamaskinen setter opp tall for hvert bildepunkt, 1 for svart og 0 for hvitt.

Strikkemønster er på mange måter et digitalisert bilde.

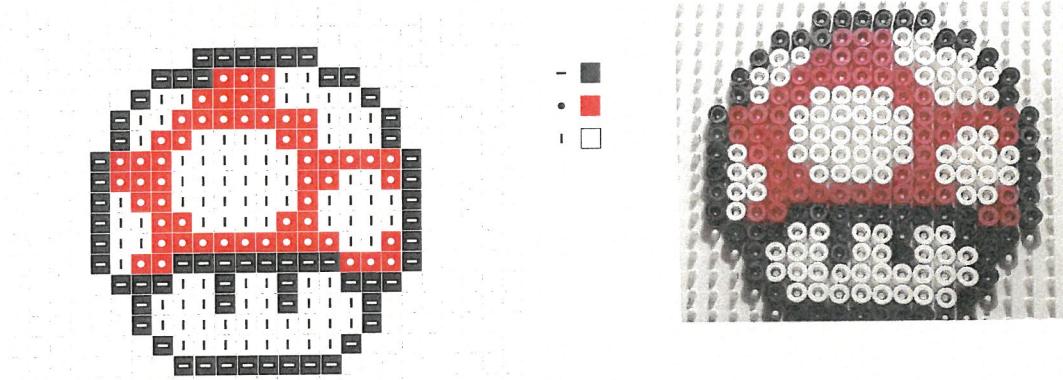


Datamaskinen trenger bare tallene og litt informasjon om hvordan de er satt opp. I dette tilfellet er tallene satt opp i et mønster med 8 punkter horisontalt og 8 punkter vertikalt, figur 5. Og så lenge maskinen vet at de skal settes opp igjen i et rutenett som er på $8 \cdot 8$ punkter, klarer den fint å gjenstape bildet fra denne tallrekken: 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0.

EKSEMPEL

Et mønster

Å gjøre bilder og illustrasjoner om til bildepunkter har ikke bare datamaskiner gjort. Ett av områdene hvor dette var vanlig lenge før datamaskinene kom i bruk, er innenfor strikking, sør og broderi. Et broderimønster er på mange måter et digitalisert bilde.



LAGRING OG MINNE

Et bilde av smilefjeset består av $8 \cdot 8$ bit, altså 64 bit til sammen. I en datamaskin er det en viss mengde plass til slike bit. Det gjelder både for **minnet** i selve maskinen og for minnet på en eventuell harddisk, eller for **lagring** på minnepinne eller CD-ROM. Dersom du vil overføre dette enkle bildet over bredbåndet ditt eller over mobilnettet, vil det legge beslag på 64 bit. I den store sammenhengen er det et svært lite bilde. Såpass enkel grafikk med så lav oppløsning krever lite både av minnet i maskinen og eventuelle linjer for overføring.

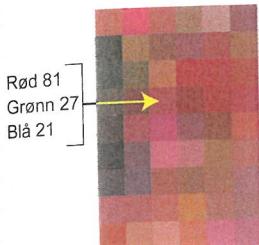
grafikk = alle ulike visningsmåter av skrift og tegning

Punktmatrise = *bitmap*

Men hvor mange ulike motiver på slike enkle bilder er det mulig å lage ved hjelp av disse 64 bitene? Svaret er 2 opphøyd i 64, altså $2^{64} = 18\,446\,744\,073\,709\,551\,616$. Du kan med andre ord lage mange ulike, enkle figurer innenfor disse rammene. Ett bilde med 64 bit krever lite plass, men kombinasjonsmulighetene innenfor 64 bit er enorme. Et bilde med alle de forskjellige kombinasjonsmulighetene kaller vi en **punktmatrise**, engelsk bitmap. Det er et slags kart over piksler.

FARGER – RGB

Det er ikke så ofte vi klarer oss med så enkel grafikk med kun svart og hvitt, så vi bruker ofte **farge** på bildepunktene våre. Datamaskinen trenger da flere tall for å kunne beskrive hvert punkt. De to vanligste måtene å beskrive farger på er ved hjelp av **RGB-verdier** eller **CMYK-verdier**. Vi bruker RGB i eksempelet videre.

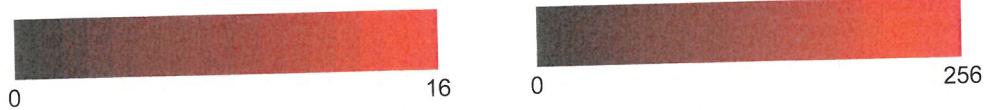


Datamaskinen setter opp tall for hvert bildepunkt, 1 for svart og 0 for hvitt.

	Bruk	Fargene
RGB	Grafikk på skjerm	Rød, Grønn og Blå
CMYK	Trykk på papir	Cyan, Magenta, Yellow og black

RGB er altså vanligst når det gjelder grafikk på skjerm. Om vi tar for oss rødt alene, har vi mange muligheter når vi skal gjengi rødt – alt fra så mørkerødt at det ender i svart, til så knallrødt at det bare er helt rødt. Bruker vi **lav oppløsning** her og sier at vi bare tar oss råd til å gjengi rødt med for eksempel 16 forskjellige **nivåer**, får vi en grov oppdeling. Øynene våre er såpass gode at vi trenger flere sjatteringer for å kunne se jevne overganger. En vanlig oppdeling er **256 nivåer** for hver farge.

Til venstre er rødt delt inn i 16 nivåer, til høyre i hele 256 nivåer.



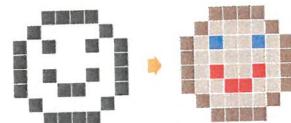
EKSEMPEL

Farger

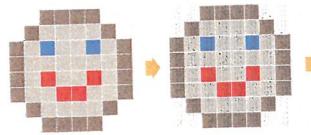
Vi går tilbake til det enkle bildet av smilefjeset, som vi nå har gitt litt farger. Så setter vi på verdier for alle fargene. Hvert bildepunkt trenger nå tre tall – ett for rødt, ett for grønt og ett for blått. Vi velger å bruke systemet med 256 forskjellige sjatteringsmuligheter for hver farge. Hver farge får da et tall fra 0 til 255.

Du ser at de hvite punktene har de tre tallene 255, 255, 255. Det er full styrke på alle fargene RGB, som gir helt hvitt. Nå er det ikke lenger helt svarte punkter i bildet, men de ville ha fått verdien 0, 0, 0. De helt røde punktene i munnen har 255, 0, 0, og ansiktsfargen 0. De helt grønne punktene i øynene har 0, 255, 0, og de helt blå punktene i håret har 0, 0, 255. Denne siste her har ganske høye verdier på alle tre tallene. Det er altså en ganske lys farge. Den har mest rødt, litt grønt og enda mindre blått. Sannsynligvis er det da en farge i retning oransje, rosa eller lysebrun.

Farge	R, G, B
Hvit	255, 255, 255
Svart	0, 0, 0
Ansiktsfarge	255, 204, 153



Smilefjeset får nå farger



Hvert bildepunkt trenger nå tre tall, ett for R, ett for G og ett for B.

LAGRING OG OVERFØRING AV FARGE Bilder

Uavhengig av hvilke farger det er snakk om, tar bildet vårt nå vesentlig mer plass enn da det var en enkel punktmatrise med null og én. Hvert bildepunkt krever nå veldig mange flere bit. For å være nøyaktig trenger vi 8 bit for å beskrive rødt (256 mulige sjatteringer av rødt), 8 bit for grønt og 8 for blått. Hvert bildepunkt trenger da 24 bit. Det er en formidabel økning, fra én bit per bildepunkt i vår enkle **punktmatrise** til 24 bit per bildepunkt i **fargebildet**. Svart-hvitt-bildet av smilefjeset, som måtte ha 64 av/på-tilstander, krever i farger $64 \cdot 24 = 1536$ slike lagringsplasser. Fargebildet vårt trenger altså 1536 bit.

En rask og moderne datamaskin har mye minneplass.



La oss si at du skal skrive opp og lagre disse to bildene som bit i en papirbok, for hånd. Du skal altså skrive alle bitene. For det enkle svarte og hvite bildet må du da skrive 64 nuller og enere. Det er en overkommelig oppgave. Men for fargebildet må du skrive 1536 ulike tall. Det tar svært lang tid og krever mange ark. Eventuelt kan du få arbeidet til å gå forttere hvis du er ekstremt rask til å skrive. Og du får plass til flere bilder om du har en veldig tykk bunke med ark. Datamaskinen har det litt på samme måte. Den må bruke mer tid og mer plass på fargebilder. En rask og moderne datamaskin med mye **minneplass** vil ha mindre problemer med det store bildet enn en treg datamaskin med lite minne.

Du skal så **overføre** disse bildene til en venn i en annen by. Du tar opp telefonen og ringer, og så leser du alle tallene mens din venn noterer i den andre enden. Det blir den samme utfordringen her som tidligere. 64 tall går greit. Men 1536 tall tar svært lang tid. For å få det til å gå forttere kan du lese forttere. Da må din venn i den andre enden skrive forttere. Og til slutt hører han ikke hva du sier. For å øke hastigheten ytterligere kan du forbedre kvaliteten på linjen. Med bedre lyd vil du kunne lese enda

fortere uten at din venn hører feil eller mister noen tall. Så mange nuller og enere som du klarer å lese på ett sekund, blir da din **datahastighet**. Den måles i bit per sekund. Leser du forttere, får du høyere **datarate** og flere bit per sekund.

Slik har datamaskinene det også. Når du skal overføre et bilde på nettet, må alle tallene komme gjennom. Et bilde med mye data tar lengre tid enn et bilde med lite. Så kan du øke hastigheten ved å sørge for at selve datamaskinen kan sende fra seg enda flere tall per sekund. Men da må datalinjen ha god nok kvalitet til å formidle dem uten at sendingen blir unøyaktig. Og datamaskinen i den andre enden må være rask nok til å ta imot.

Datahastighet måles i bit per sekund.

KOMPRESJON AV BILDER

Mediefiler tar stor plass i dataverdenen. **Mediefiler** inneholder bilder, lyd og video. De tar så stor plass at det er funnet opp måter å få datafilene til å ta mindre plass uten at de blir ødelagt, og uten at vesentlig informasjon i for eksempel et bilde går tapt. Dette kalles **kompresjon**. Det fins to hovedtyper kompresjon. Den ene kalles **tapsfri kompresjon** (*lossless*). Det er betegnelsen vi bruker når vi har gjort en fil mindre i størrelse, men likevel har nok data til å gjenskape en fil som er identisk med originalen. Den andre typen kalles **destruktiv kompresjon** (*lossy*). Den er svært utbredt når det gjelder mediefiler, for eksempel bilder.

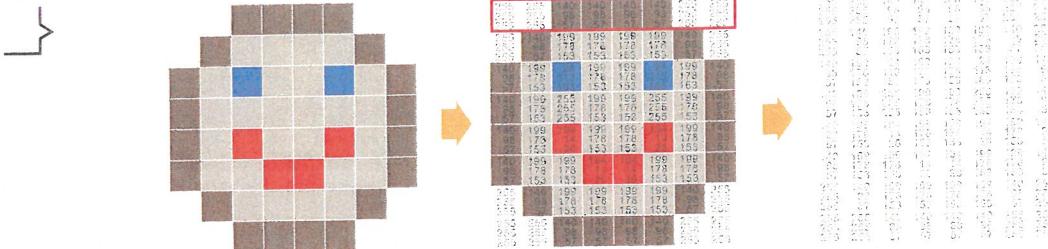
Kompresjon gjør at datafilene tar mindre plass uten at de blir ødelagt, og uten at vesentlig informasjon i for eksempel et bilde går tapt.

To typer kompresjon	Forklaring
Tapsfri kompresjon	Det er betegnelsen vi bruker når vi har gjort en fil mindre i størrelse, men likevel har nok data til å gjenskape en fil som er identisk med originalen. Kompresjon brukes ofte på illustrasjoner og figurer.
Destruktiv kompresjon	Den er svært utbredt når det gjelder mediefiler, for eksempel bilder. Kompresjonen forenkler bildet basert på kunnskapen om hva som er nødvendig å ta med.

TAPSFRI KOMPRESJON

Hvordan er det mulig å fjerne mye data fra en fil og likevel klare å gjenopprette den? For å forklare det skal vi nå gå tilbake til fargebildet av smilefjeset. Du sitter der med boka di og skal skrive inn alle tallene. Og siden du er et menneske og ikke en datamaskin, skriver du tallene for hvert bildepunkt. Det er tre tall per bildepunkt og 64 bildepunkter i alt. Her skal altså totalt 192 ulike tall skrives.

Smilefjeset består av 192 ulike tall.



Vi tar nå for oss kun den øverste linja av bildet. På papir blir denne tallrekken slik: (255, 255, 255), (255, 255, 255), (153, 102, 53), (153, 102, 53), (153, 102, 53), (153, 102, 53), (255, 255, 255), (255, 255, 255). Så langt har du skrevet 24 tall med to eller tre siffer i hvert av dem. I alt blir det 68 siffer. Men her er det rom for forbedring. Det er ikke nødvendig å drive med all denne repetisjonen. I stedet for å skrive (255, 255, 255) to ganger og (153, 102, 53) fire ganger osv., kan du heller bare skrive slik: 2, (255, 255, 255), 4, (153, 102, 53), 2, (255, 255, 255). Og de hvite bildepunktene kan du forenkle enda mer ved å skrive bare (3, 255). Da blir resultatet slik: 2, (3, 255), 4, (153, 102, 53), 2, (3, 255). Vi har redusert det til 19 siffer. Dette er under en tredjedel av de opprinnelige 68 sifrene.

Tapsfrei kompresjon brukes ofte til enkel grafikk og figurer.

Bildet vil nå bare kreve en tredjedel av plassen i boka di, og du kan lese det tre ganger raskere i telefonen. Likevel kan du siden gjenskape den opprinnelige tallrekken hundre prosent nøyaktig. Her snakker vi altså om en **tapsfrei kompresjon**. Akkurat denne måten å gjøre det på kalles **RLE**, av *Run Length Encoding*. I datamaskinen må vi sende med litt informasjon om hvilken type kompresjon som er brukt, slik at maskinen som mottar filen, vet hva den skal gjøre for å gjenskape originalen. Dette tar litt plass, og begge maskinene må bruke noe maskinkraft på å stokke om på tallene. Det som kalles **koding** og **dekoding**. Selve systemet kalles en **kodek**.

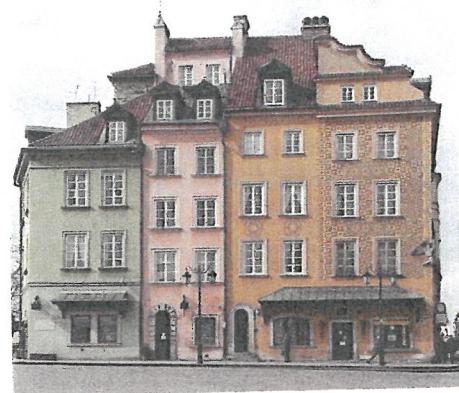
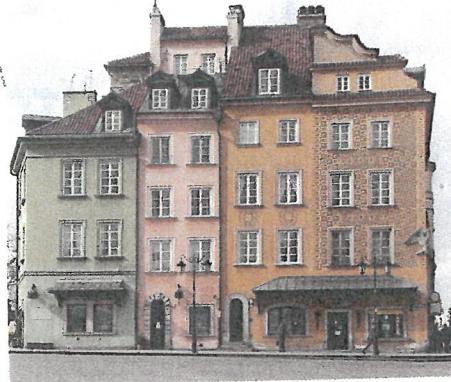
Ved tapsfrei kompresjon med RLE kan vi gjenskape den opprinnelige tallrekken hundre prosent nøyaktig. Datamaskinen som sender filen, legger ved informasjon om hvilken type kompresjon som er brukt. Dermed kan maskinen som mottar filen, gjenskape originalen. Dette kalles koding og dekoding.

DESTRUKTIV KOMPRESJON

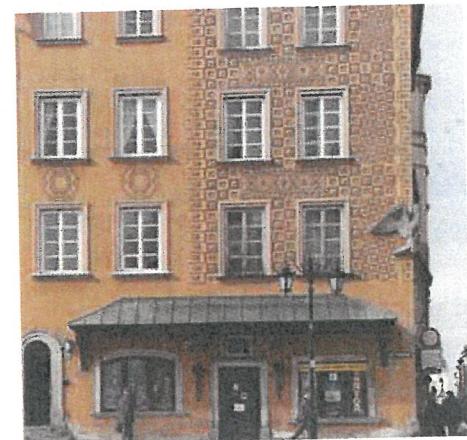
For enkel grafikk og figurer kan tapsfrei kompresjon gi en svært effektiv komprimering. Men for fotografi er det ikke så ofte at mange bildepunkter etter hverandre er helt like. Da kommer **destruktiv kompresjon** inn. Et bilde inneholder som oftest en mengde detaljer som øyet vårt ikke kan se. Hvis vi da forenkler bildet basert på kunnskapen om hva som er nødvendig å ta med, kan vi komprimere ganske kraftig.

Destruktiv kompresjon brukes ofte på fotografier.

Bildet til venstre er
ukomprimert og krever
27 megabyte plass i
datamaskinen. Bildet
til høyre er kraftig
komprimert og krever
bare 220 kilobyte.

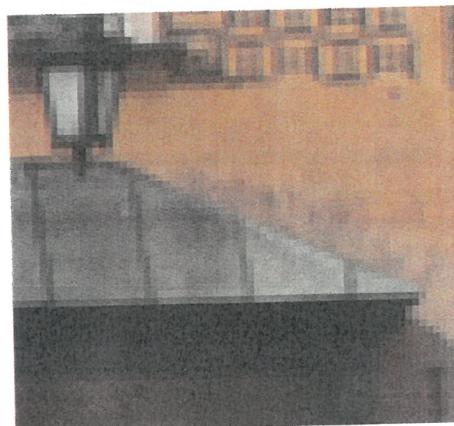


Forstørrer vi mer, er det
lettere å se hva som er
gjort. Vi ser at bildet til
høyre mangler detaljer.



Bildet til venstre nedenfor er ukomprimert. Til høyre kan du se at datamaskinen først har delt bildet inn i en serie med firkanter. Så har den prøvd å redusere antallet farger i hver rute og samtidig prøvd å tvinge så mange bildepunkter som mulig til å bli helt like. I dette eksempelet har vi komprimert så hardt at vi ikke bare reduserer størrelsen på filen, men det har også gått utover bildekvaliteten.

Forstørrer vi enda mer,
ser vi tydeligere hva
kompresjonen har ført til.



Hvis vi så ber algoritmen om å gå litt lettere til verks, har vi til slutt et bilde som det menneskelige øyet har vanskelig for å se problemer med. Men med tanke på lagring og overføring kan det likevel være problematisk.

EKSEMPEL

Filstørrelse

Alle bildene du legger ut på nettsidene dine, må lastes ned av brukerne til brukernes datamaskin. **Filstørrelsen** til bildet er av betydning når det gjelder lagring av bildefiler og overføring av bilder via Internett. Enheten for filstørrelse er byte (B). Som regel oppgir vi filstørrelsen i kilobyte (kB) eller megabyte (MB). Et viktig punkt som endrer filstørrelsen, er valg av filformat og komprimering.

I et bilderedigeringsprogram kan vi lagre bildet i et filformat som **komprimerer** filstørrelsen slik at bildet passer det vi skal bruke det til. De vanligste filformatene er BMP, JPG, PNG og TIF. De ulike filformatene lagrer informasjon om bildene på forskjellige måter, og det er

viktig å bruke riktig format i hvert tilfelle. En gyllen regel er at du bør bruke formatet JPG når du skal vise fotografier, og GIF eller PNG til mindre webgrafikk, som logoer og ikoner.

Tips for bruk av bilder på nettetsider

1. Bruk bildeformatet JPG til fotografier på nettsider.
2. Rediger kvaliteten i JPG-bildene i et bilderedigeringsprogram så mye som mulig uten at kvalitetstapet blir synlig.
3. Skaler bildene til den størrelsen (antall piksler i bredden og høyden) de skal ha på nettsiden, i et bilderedigeringsprogram.

Filformat	Egner seg til	Komprimering
JPG	Fotografier	destruktiv
PNG	Fotografier, ikoner, webgrafikk, logoer og grafikk til animasjoner og spill.	tapsfri
PNG-8	Samme som PNG. PNG-8 gir mindre filstørrelse.	tapsfri
GIF	Lite egnet. Bruk heller PNG eller PNG-8.	destruktiv eller tapsfri

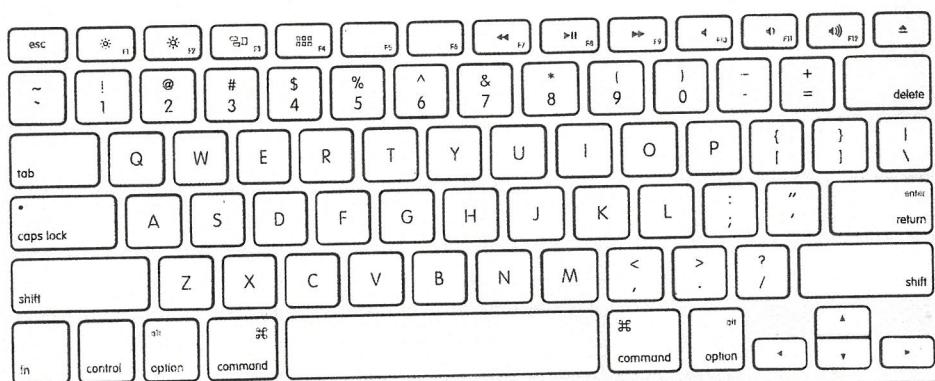
4.4 Tegnsett

UTF-8 OG ANSI

Tegnsettet i en datamaskin avgjør hvordan bokstavene i alfabetet, tall og andre skrifttegn skal vises på skjermen. Som for bilder gjelder det å bryte ned ting til 0 og 1. Derfor har hver enkelt bokstav, tall og tegn sitt binære tall. Tidligere brukte de fleste datamaskiner tegnsettet **ASCII**. Det brukte 7 bit og ga 128 muligheter for ulike tegn (0 til 127). Programvarehuset Windows laget etter hvert sitt eget tegnsett som de kalte **ANSI**. De utvidet da tegnsettet til 8 bit, slik at de blant annet fikk med bokstavene Å, Ø og Å. I dag bruker datamaskiner tegnsettet Unicode, som har 8 bit og derfor 256 mulige tegn vi kan bruke.

Du har sikkert lagt merke til at du i begynnelsen av koden skriver `<meta charset="UTF-8">`. **UTF-8** står for Unicode Transformation 8-bit og er et Unicode-tegnsett som kan vise alle tegn fra alle språk. Det som kjennetegner UTF-8, er at det bruker ulike lengder på én byte, alt fra én, til fire byte. Når du skriver HTML-kode, blir UTF-8 automatisk brukt som tegnsett.

Et tastatur har taster med ulike funksjoner.



Tegnsettet i en datamaskin avgjør hvordan bokstavene i alfabetet, tall og andre skrifttegn skal vises på skjermen. Forskjellige tegnsett har ulikt antall tegn: ASCII har tegn fra tallene 0 til 127, mens ANSI og UTF-8 har i tillegg fra tallene 128 til 255.

Binært	Heksadesimalt
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

DET HEKSADESIMALE TALLSYSTEMET

Det **heksadesimale tallsystemet** har 16 ulike tegn. Tabellen i marginen viser at det heksadesimale tallsystemet går fra sifrene 0 til 9, og så videre fra bokstaven A til F. På denne måten kan vi erstatte de fire binære tallene 0101 med tallet 5 fra det heksadesimale tallsystemet. Og vi kan erstatte 1010 med bokstaven A. Dette gjør at vi kan skrive det binære tallet 0101 1010 som 5A. En av fordelene med dette systemet er at det er lettere for oss å lese enn det binære tallet med mange nuller og ett-tall. Samtidig er det lettere for datamaskinen å oversette mellom det binære og det heksadesimale tallsystemet enn mellom det binære og det vanlige tallsystemet. Sammenhengen mellom det heksadesimale tallsystemet og det binære tallsystemet ser du i tabellen til venstre.

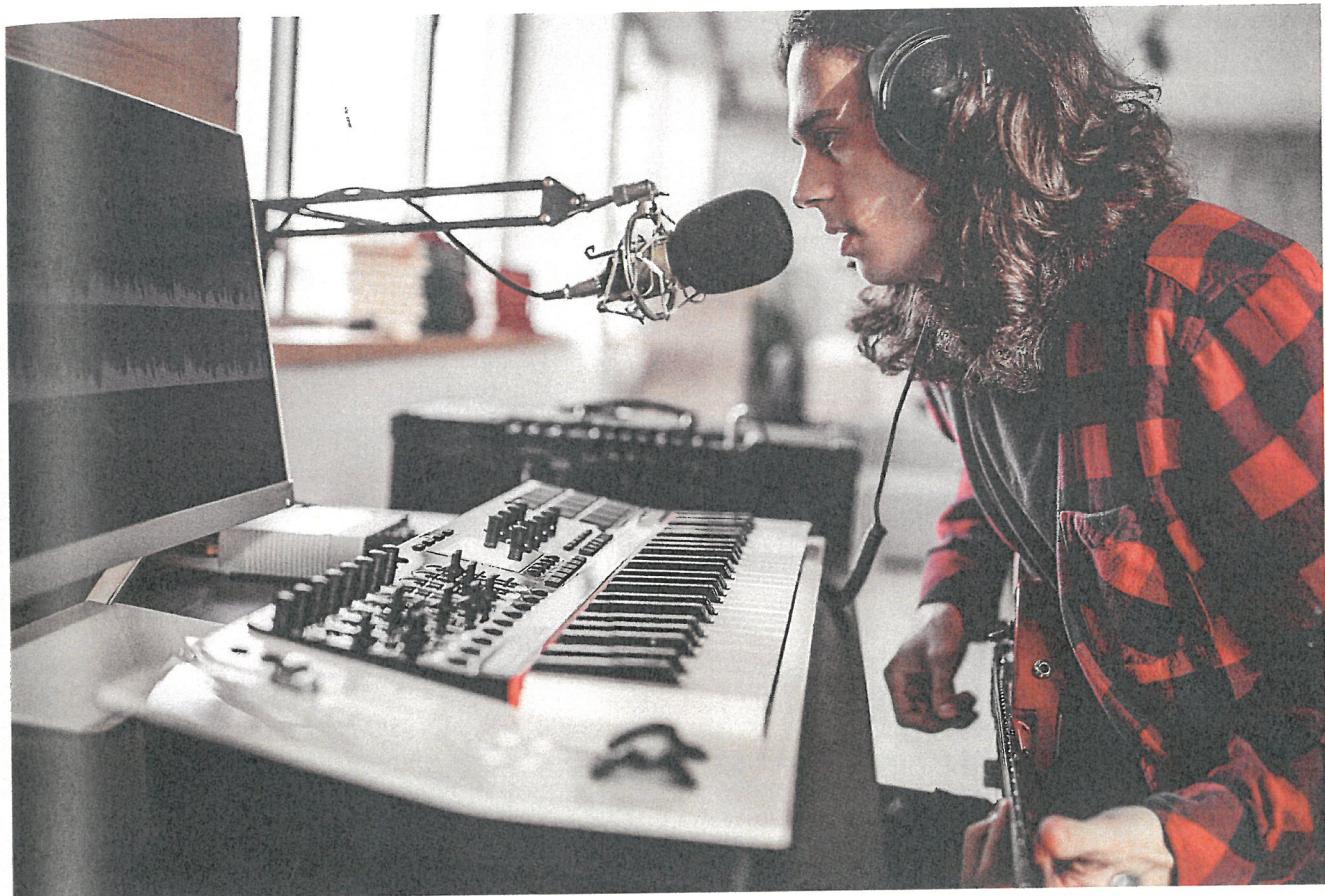
Tall i titallsystemet	Binaært tall	Heksadesimalt tall	Tegn på tastaturer
32	010 0000	20	Mellomrom
33	010 0001	21	!
34	010 0010	22	"
35	010 0011	23	#
48	011 0000	30	1
49	011 0001	31	2
50	011 0010	32	3
64	100 0000	40	@
65	100 0001	41	A
66	100 0010	42	B
67	100 0011	43	C
68	100 0100	44	D
97	110 0001	61	a
98	110 0010	62	b
99	110 0011	63	c
100	110 0100	64	d
106	110 1010	6A	j
107	110 1011	6B	k
108	110 1100	6C	l
122	111 1010	7A	z
197	1100 0101	C5	Å
198	1100 0110	C6	Æ
216	1101 1000	D8	Ø

4.5 Digitale lydopptak

OPPTAK AV LYD

Lyd er hurtige endringer i trykk, også kalt **lydbølger**. Lyd kan spre seg i både vann og faste stoffer, men vanligvis har vi å gjøre med lyd som sprer seg i luft, altså den lyden vi kan høre. Helt siden slutten av 1800-tallet har mennesker vært i stand til å gjøre opptak av lyd. Først foregikk det med mekaniske opptak på voksruller, så fikk vi elektroniske opptak på magnetbånd, og siden fulgte **digitale opptak** ved hjelp av datamaskiner.

En **mikrofon** kan gjøre lydbølger i luft om til **elektriske signaler**. Det skjer som oftest ved hjelp av en membran som setter en spole i bevegelse mot en magnet. Høyt lydvolum vil få membranen og dermed spolen til å svinge dypere. Da lages en høyere



Vi kan gjøre digitale opptak av lyd.

Lydbølger i luft kan gjøres om til elektriske signaler.

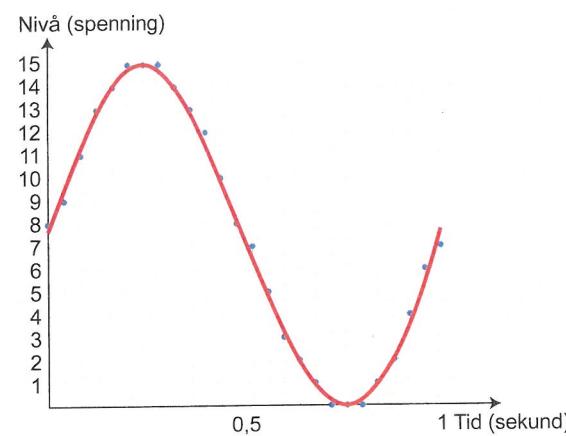
elektrisk spenning i spolen. Endringene i spenningen blir et elektrisk signal som svarer med lyden. Det gjør at vi får et elektrisk signal som svinger i takt med membranen og dermed med lyden i rommet. Dette signalet kan vi så sende videre gjennom en forsterker og ut i høyttalere, eller vi kan lagre det på for eksempel et magnetbånd. Da har vi et analogt lydopptak. Skal vi lagre og behandle lyd i en datamaskin, må det analoge lydsignalet digitaliseres. Det må altså gjøres om til tall.

Skal vi lagre og behandle lyd i en datamaskin, må lydsignalet digitaliseres. Det må altså gjøres om til tall.

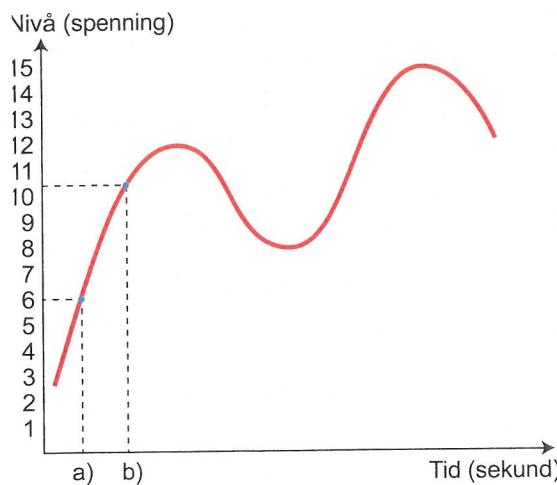
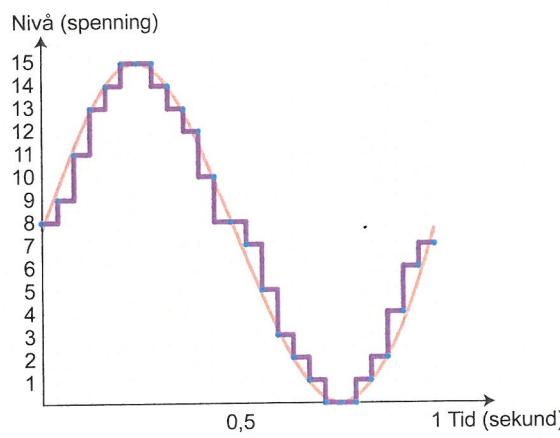
DIGITALISERING AV LYD

Lydsignalet i et analogt lydopptak er endringer i spenning over tid. Spenningen kan vi måle. Du kan fremstille den grafisk med tiden langs x-aksen og nivået på spenningen oppover langs y-aksen. Når som helst kan du gå inn og måle for å tallfeste spenningen. Når vi digitaliserer lyd, går vi inn med jevne mellomrom og måler spenningen. Hver gang tar vi vare på verdien og lagrer den i den samme rekkefølgen som målingene ble gjort. Hvor ofte og hvor nøyaktig vi måler, er med og avgjør **kvaliteten** på lyden.

Vi mäter spenningsnivået 26 ganger i sekundet.
 Én bølge i løpet av ett sekund gir oss en frekvens på 1 Hz.
 Menneskets øre oppfatter lyd med frekvens mellom 20 Hz og 20 000 Hz.



Avspillingen ville blitt hakkete.



Illustrasjonen til venstre viser at vi har målt spenningen 26 ganger i sekundet. Nivåene har vi plassert på en skala fra 0 til 15. Når vi skal rekonstruere denne lydbølgen, ser vi i den midterste figuren at dette ikke blir nøyaktig nok. Resultatet blir hakkete, og ved avspilling ville lyden fått svært **lav kvalitet**.

Hvor ofte vi gjør målingene, kalles **samplingsfrekvensen**. **Nøyaktigheten** på hver måling oppgir vi i antall bit. Dersom vi bruker en god mikrofon, får vi en meget detaljert og glatt kurve på målingene. For å klare å få til en like fin kurve med digital teknologi må vi gjøre målinger svært ofte, og hver måling må ha en svært nøyaktig angivelse. Om vi gir hver måling en lav oppløsning, som i bildene til venstre, som er på 4 bit, må målingene våre passe inn i ett av 16 tilgjengelige nivåer siden skalaen går fra 0 til 15.

I illustrasjonen til venstre treffer måling a) direkte på tallet 6. Da er alt vel. Måling b) treffer på 10,4. Skalaen vi sampler med, har 16 (heltalls-)nivåer. Vi må da være fornøyd med at denne målingen blir rundet ned til vi får heltallsverdien 10. Vi kan dermed ikke gjenskape den opprinnelige kurven nøyaktig nok.

EKSEMPEL**Lydkvalitet**

Det viktige spørsmålet er nå hvor stor nøyaktighet vi trenger i lydmålingene. Både oppløsningen i måleresultatene (i antall bit) og hvor ofte vi mäter (samplingsfrekvensen), er av betydning. Det første digitale lydproduktet som fikk stor utbredelse hos publikum, var CD-plata (*Compact Disc*). Da den standarden ble definert, ble samplingsfrekvensen satt til 44,1 kHz, altså 44 100 målinger per sekund. For oppløsningen i måleresultatene ble det valgt 16 bit, altså 65 536 mulige nivåer. I systemet blir de delt inn fra -32 768 til +32 767.

Dette gir CD-plata en lydkvalitet som er god nok for de fleste typer lytting, men i situasjoner med ekstra krav til

kvalitet benyttes enda høyere samplingsfrekvens og oppløsning. Når vi gjør profesjonelle opptak, er det viktig å ta hensyn til at lyden skal gjennom mange ledd før den når lytteren. Derfor gjøres som oftest originalopptak med mye høyere kvalitet.

Det blir ganske store datamengder når vi gjør målinger mange tusen ganger per sekund og deler inn hver måling så nøyaktig. 16 bit tar vi vare på 44 100 ganger i sekundet. Det gir en datastrøm på $16 \text{ bit} \times 44\,100 = 705\,600 \text{ bit per sekund}$. Siden de fleste opptak gjøres i stereo, trengs to kanaler (høyttalere). Da ender vi på en **bitfart** på $704\,600 \times 2 = 1\,411\,200 \text{ bit/s, eller ca. } 1,41 \text{ Mb/s}$.

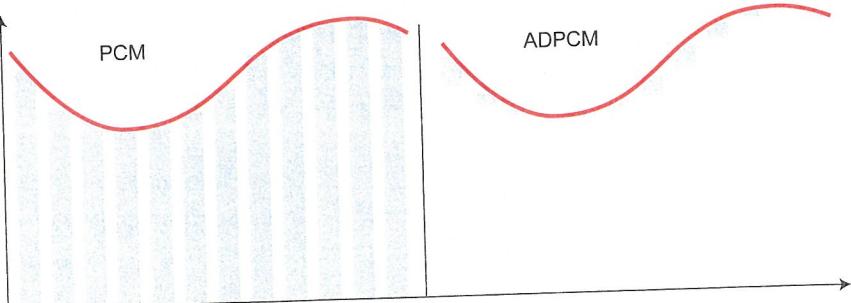
KOMPRIMERING

Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**. Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**. Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**. Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**. Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**. Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**. Når vi skal lagre eller sende lyd, krever det mye plass. Derfor bruker vi **komprimering**.

Om vi gjør 10 målinger av en PCM-lyd, kan målingene være se slik ut: 24876, 23645, 23899, 24561, 25010, 24890, 24764, 24487, 23920, 23843. Her må vi ta vare på 10 ganske store tall. Hvis vi kunne ha funnet en måte å gjøre tallene mindre på, kunne vi spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass. En av metodene som brukes til dette er å lagre en måling og deretter ha spart plass.

Med kompresjon reduserer vi hvor mange tall som skal sendes.

Til venstre er signalet målt og lagret tolv ganger. Til høyre er nivået målt og lagret én gang. Deretter måles og lagres kun endringen de neste elleve gangene.



1 kHz (kilohertz) er 1000 bølgesvingninger per sekund.

Med denne metoden kan vi minske datamengden ganske betydelig uten å få dårligere kvalitet. Dette er **tapsfri kompresjon**. De opprinnelige dataene kan rekonstrueres helt nøyaktig. For å klare å lage enda **mindre lydfiler** bruker vi mye mer avanserte metoder. Som i bildekompresjon tar vi utgangspunkt i hvordan vi mennesker faktisk fungerer. For hørselen vår er det slik at den målbare delen går fra ca. 20 Hz til 20 000 Hz. Etter hvert som vi blir eldre, mister vi mer og mer av detaljene, spesielt i de høye frekvensene.

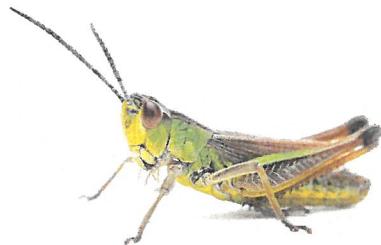
Med tapsfri kompresjon kan vi minske datamengden ganske betydelig uten å få dårligere kvalitet. De opprinnelige dataene kan rekonstrueres helt nøyaktig.

EKSTRASTOFF

Høye frekvenser

Det klassiske eksempelet her er den gamle mannen som er bekymret fordi det var så mye mer gresshopper før. Nå er de døde alle sammen. På blomsterengene der han tidligere hørte mange av dem, er det nå ingen. Sannheten er at de er der alle sammen, men når vi blir eldre, hører vi ikke lenger de høye frekvensene. Og lyden av gresshopper har stort sett svært høye frekvenser.

Gresshopper har høye frekvenser.



HØRSEL OG LYDKVALITET

Hørselen vår er aller best til å skille lyder mellom 2 kHz og 5 kHz. Det er ganske naturlig siden menneskelig tale i hovedsak foregår i dette frekvensområdet. Det er også slik at hvis du hører en høy lyd samtidig som du hører en lavere lyd med nesten samme frekvens, så dekker hørselen over denne andre lyden. Slike detaljer kan utnyttes i **kompresjonsalgoritmer** som ikke er tapsfrie. **MP3** og **AAC** er to slike **lydformer**. De bruker til tider svært avanserte matematiske funksjoner og modeller for å ta bort data, slik at filene blir mindre. Dette skjer hele tiden i et kompromiss med lydkvaliteten. Det hele blir enda mer komplisert fordi forskjellige typer lyd reagerer ulikt på forskjellige typer kompresjon. En **kompresjon** som er god nok for én type filmyd, kan være for dårlig for en annen type filmyd.

Den opplevde **lydkvaliteten** vi da sitter igjen med, blir påvirket av mange faktorer. Og vi får diskusjonen om hva virkelig god lyd er. På en gammel telefonlinje som holder seg mellom 2 og 5 kHz, hører du greit hva folk siør. Men det er ikke så flaut som du tror, at du ikke hørte forskjell på mor og datter eller far og sønn. Ofte avslører frekvensene under og over 2 kHz og 5 kHz forskjellene i ellers ganske like stemmer. Konklusjonen om at vi kan høre lyd mellom 20 Hz og 20 kHz, baserer seg stort sett på tester der du får servert lyder og svarer på om du kan høre dem eller ikke. Det betyr ikke at vi ikke lar oss påvirke av frekvenser som er høyere eller lavere.

Menneskelig tale foregår mellom frekvensområdet 2 kHz og 5 kHz.



Hvis en lydfil skal gjennom flere ledd med komprimering og dekomprimering, kan en ellers god lyd få stygge feil fordi en kompresjon som i utgangspunktet var god, blir påvirket av nye komprimerte ledd. I lydproduksjon brukes både samplingsfrekvenser og oppløsninger som kan virke unødvendig høye. Når du skal arbeide med digital lyd, er det en grei grunnregel å begynne med så høy kvalitet som mulig og beholde tapsfrie formater så lenge det lar seg gjøre. Da blir lyden din mer robust og vil holde bedre når den så legger ut på en lang vei med kompresjon og dekompresjon før den når sitt publikum.

Begynn med så høy kvalitet som mulig, og behold tapsfrie formater så lenge du kan.

Heldigvis er teknologiutviklingen på lydkvalitetens side – med bedre nettverk og kraftigere datamaskiner med større minnekapasitet. Det blir stadig mer aktuelt å benytte tapsfri håndtering hele veien hjem til lytteren. Da unngår vi enkelt og greit det uforutsigbare med for hard kompresjon.

4.6 Video og animasjon

LEVENDE BILDER



En gammeldags film kunne vise 24 bilder per sekund. Nye digitale standarder kan ha over 120 bilder per sekund.

Levende bilder krever relativt mye av en datamaskin. Store mengder data skal prosesseres (bearbeides) på kort tid. Veldig mange bit og byte skal lagres, overføres og gjøres om til bilder. Og det må skje raskt, for her skal det vises mange bilder etter hverandre per sekund. For god gammeldags film vises 24 bilder per sekund. For fjernsyn og video i de landene som bruker PAL-systemet, er det 25 bilder per sekund, og for landene som bruker NTSC-systemet, er det 30 bilder per sekund for fjernsyn og video.

Etter hvert som nye digitale standarder har tatt over for de gamle analoge, har det dukket opp formater med både 50, 60, 120 og enda flere bilder per sekund. Når du skal vise 25 flotte fargebilder med høy oppløsning hvert sekund, blir det store data-mengder. Så hvis kompresjon er viktig for stillbilder, er det enda viktigere for video og animasjon. Og siden video og animasjon består av mange stillbilder etter hverandre, kan vi benytte mye av de samme prinsippene for kompresjon av video som vi bruker for kompresjon av stillbilder.

Video og animasjon består av mange stillbilder etter hverandre. Vi kan benytte mye av de samme prinsippene for kompresjon av video som vi bruker for kompresjon av stillbilder.



Video består av mange stillbilder etter hverandre.

EKSEMPEL

HD – datarate

For å regne litt på **datamengde** kan vi ta utgangspunkt i formatet som kalles **HD**. Det har en oppløsning på 1920×1080 , altså er det 1920 bildepunkter bredt og 1080 bildepunkter høyt. Om vi gir alle bildepunktene en brukbar fargeoppløsning, får de $8 + 8 + 8$ bit, altså 24 bit. Ganger vi dette opp, betyr det at et HD-videobilde krever $1920 \times 1080 \times 24 = 49\,766\,400$ bit for å kunne lagres, altså ca. 50 megabit per bilde. Skal vi kjøre dette som jevn video, må vi ha 25 slike bilder per sekund. Da trenger vi $25 \times 49\,766\,400 = 1\,244\,160\,000$ bit hvert sekund. Dette kaller vi en **datarate**. Ukomprimert HD-video krever altså en datarate på ca. 1,24 gigabit per sekund. Skal vi lagre én time med ukomprimert HD-video, trenger vi altså $60 \times 60 \times 1\,244\,160\,000 = 4\,478\,976\,000\,000$ bit. Med andre ord skal ca. 4500 milliarder små brytere være av eller på.

En datarate oppgis i bit per sekund.

For den **dataraten** som vi nevnte i eksempelet på forrige side, overføring av data gjennom datalinjer, snakker vi om bit per sekund. En del overføringsmetoder baserer seg faktisk på overføring av **én og én bit**. Når vi lagrer filer i en datamaskin, snakker vi heller om **byte**. Så om noen spør deg hvor stor plass én time med ukomprimert video vil kreve i datamaskinen, er det mer fornuftig å gi svaret i antall byte. Det er åtte bit i en byte. Så da krever denne timen med video $4\ 478\ 976\ 000\ 000/8 = 559\ 872\ 000\ 000$ byte. Eller litt enklere sagt ca. 560 gigabyte med plass. Og det er bare for videodelen. Som oftest følger det med lyd også, gjerne stereolyd.

Vi oppgir svaret på hvor stor plass ukomprimert video vil kreve i datamaskinen, i antall byte.

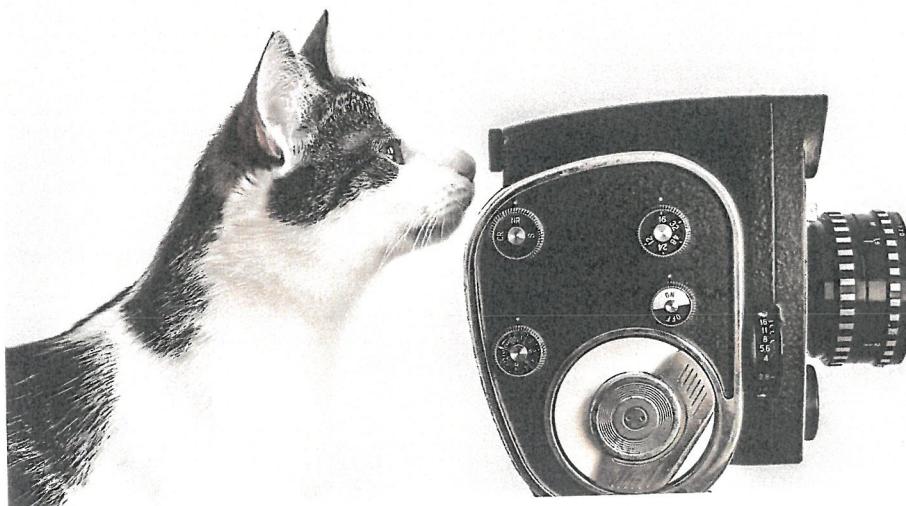
KOMPRIMERING AV VIDEO

Siden video krever så mye plass i datamaskinen og gjennom nettverksforbindelser, er det nyttig å få **komprimert videofilene**, slik at de tar mindre plass. På samme måte som for bilder og lyd har vi både kompresjon som ikke gir kvalitetstap, og kompresjon som gir kvalitetstap for video. Hvert bilde kan behandles for seg med metoden vi kalte «*run length encoding*». Da reduserer vi datamengden uten å tape kvalitet. I tillegg kan vi komprimere ~~hvert~~ bilde, f.eks. ved JPG-komprimering. Hvis vi gjør det med video, får vi en videofil som kalles MJPG, Motion-JPG. Den inneholder en rekke bilder som alle er komprimert med JPG-algoritmen.

Video krever mye plass, og det er viktig å komprimere videofilene, slik at de tar mindre plass. To metoder:

- Run length encoding: Hvert bilde behandles for seg, og datamengden reduseres uten kvalitetstap.
 - JPEG-komprimering (MJPG): Hvert bilde komprimeres med en JPG-algoritme.
-

Videokompresjon basert på et nøkkelsbilde.
All informasjon i nøkkelsbildet lagres, kun endringene i de neste bildene lagres, og så lagres all informasjon igjen når det dukker opp et nytt nøkkelsbilde.



Men **videokomprimering** blir mye mer effektiv om vi ser på mange av bildene i videoen samlet, siden det oftest er bare små forskjeller fra bilde til bilde. Filmer vi en person med 25 bilder per sekund, rekker ikke personen å gjøre store sprell mellom bildene. Dette utnytter vi i videokompresjon. Vi sammenligner bilde for bilde og tar bare vare på forskjellene. På en måte kan vi sammenligne dette litt med PCM-kodingen for lyd. Når vi bare lagrer forskjellene, analyserer **videokompresjonsprogrammet** bildene nøyne. Når programmet kommer til et klipp, altså et sted der det plutselig er veldig stor forskjell i forhold til forrige bilde, lagrer videokompresjonsprogrammet hele bildet på nytt. Det kalles et **nøkkelbilde**. På engelsk kan du finne både begrepet *keyframe* og begrepet *I-frame*.

REDIGERING AV VIDEO

De fleste kompresjonsalgoritmene baserer seg på et system som tar utgangspunkt i et nøkkelbilde. Systemet lagrer bare endringene for det neste og noen flere bilder før det igjen tar jobben med å lagre et helt nøkkelbilde, uavhengig av om det er snakk om klipp eller ikke. For alle videoer som har slik kompresjon, blir det da litt mer komplisert for datamaskinen å la deg **redigere innholdet**. Når du går inn på et gitt bilde i sekvensen og vil gjøre et klipp der – og det da ikke er et nøkkelbilde, men et bilde som bare er lagret som en forskjell i forhold til bildet før og bildet etter – må maskinen først regne seg frem til hvordan det bildet skulle ha sett ut, ved å se på bilder helt tilbake til forrige nøkkelbilde, og noen ganger også frem til neste. De fleste redigeringsprogrammene gjør dette raskt, slik at du ikke merker noe til det. Men du kan risikere at du ikke får klippe hvor du vil, eller at maskinen plutselig bruker tid på å gjøre noe som kanskje virket som en svært enkel jobb.

Innen **profesjonell videoproduksjon** benyttes ofte såkalte «I-frame only»-formater, altså komprimering som tar for seg hvert enkelt bilde. Etter hvert som kameraene er blitt bedre, er det også mer vanlig å arbeide med helt ukomprimerte filer så langt ut i produksjonen som mulig. Dette gjøres på samme måte som med lyd for å bevare kvaliteten.

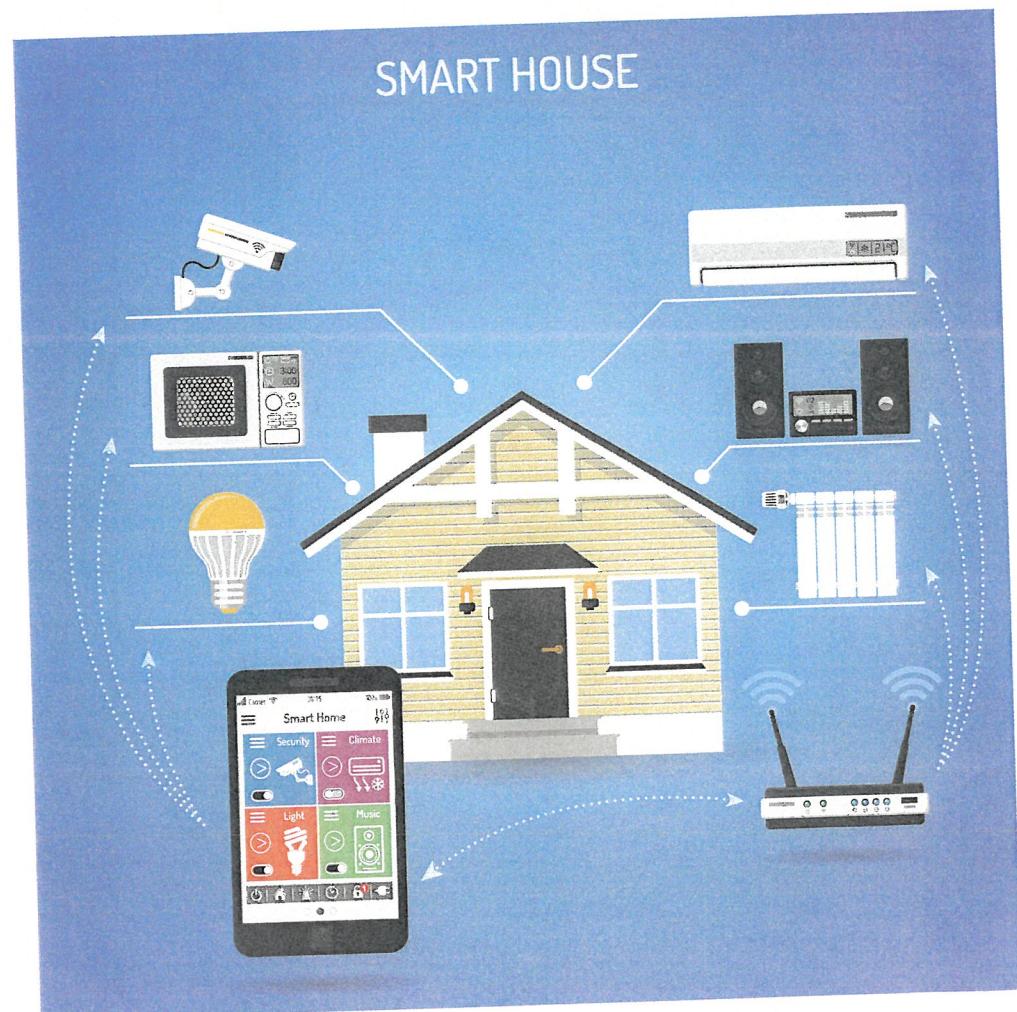
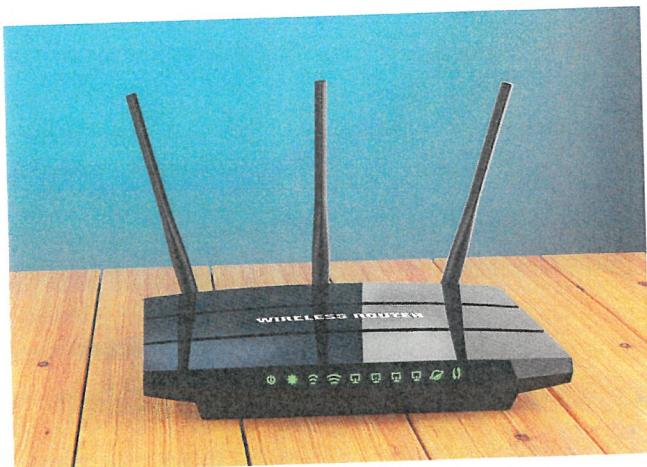
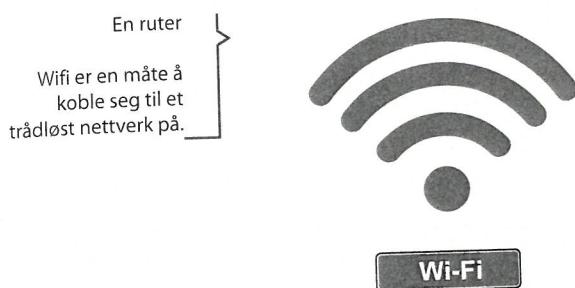
4.7 Kommunikasjon mellom digitalt utstyr

Mye av det vi bruker datamaskiner og annet digitalt utstyr til, krever at vi har tilgang til Internett, eller at utstyret kommuniserer med andre enheter. I dette kapittelet ser vi på noen av de viktigste måtene digitale enheter kan kommunisere med hverandre.

DIGITALE NETTVERK

Digitalt utstyr har ofte en egen komponent, et **nettverkskort**, som gjør at det kan kobles opp mot et nettverk av andre enheter. Dette nettverket kalles **digitale nettverk**, og kommunikasjonen gjennom nettverket kalles **nettverkskommunikasjon**.

Et nettverk kan for eksempel bestå av en datamaskin og en **ruter** i et **hjemmenettverk**. I et hjemmenettverk er det oftest trådløs kommunikasjonen mellom datamaskinen, telefoner, ruter og printere gjennom **WLAN** (Wireless Local Area Network). Det er et trådløst lokalnettverk som bruker **wifi-tilkobling** mellom datamaskinene og ruten.

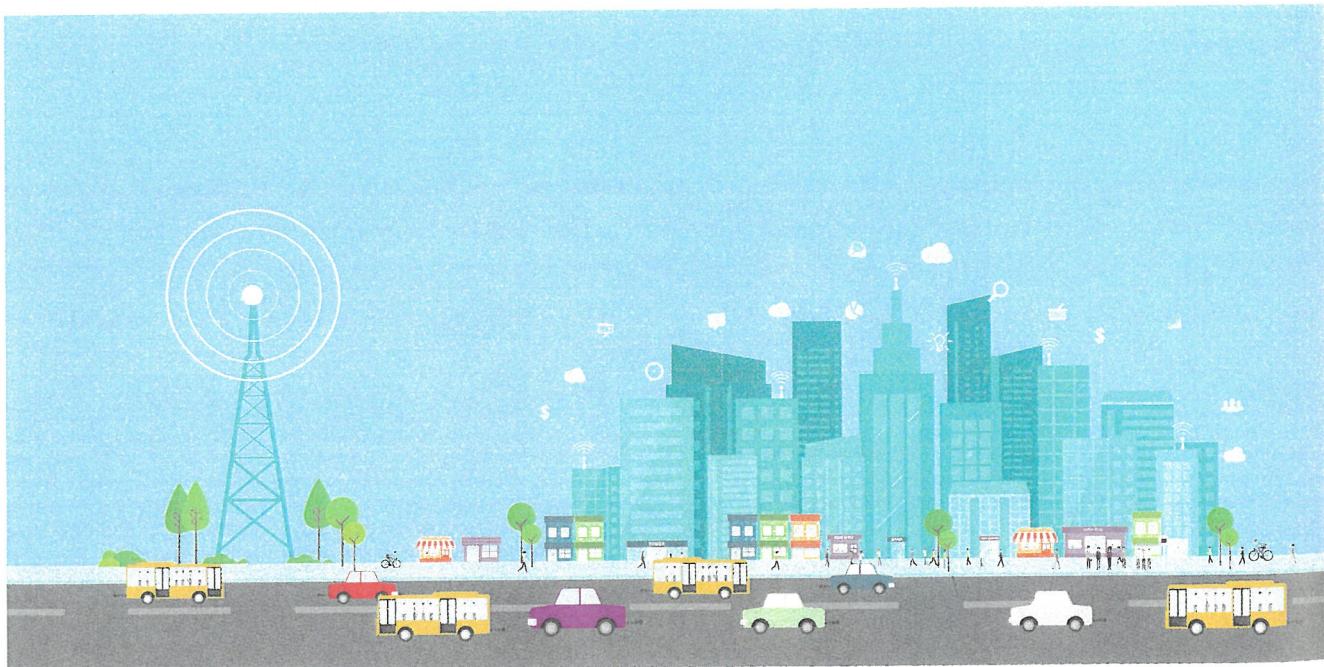


Rutere fra flere hjemmenettverk kan vi koble til **Internett**, og din og naboenes datamaskin kan da kommunisere med hverandre. Ruten er da koblet enten til en telefonledning, en kabel-TV-kabel, en radiosender eller en fiberoptisk kabel, som så er koblet til et større nettverk via andre spesielle rutere for Internett. Oppkobling til Internett kan gå via telefonledningene med en teknologi som heter **ADSL** (*asymmetric digital subscriber line*). Koblingen kan gå mellom en rute til kabel-TV-kablene eller via en fiberkabel. Noen internettleverandører tilbyr også trådløs oppkobling mot 4G/5G. Det brukes mest for mobiltelefoner.

Et selskap som tilbyr internettoppkobling, **ISP** (*Internet Service Provider*), administrerer ruten. Der kan hastigheten på datatrafikken, **datahastigheten** til internettoppkoblingen – det vil si hvor mye data som kan sendes per sekund – endres etter kapasiteten i nettet og det abonnementet vi har betalt for. Det er vanlig å snakke om flaskehalser i nettverket. Det hjelper lite å ha en god internettoppkobling via fiberkabel hvis du i hjemmenettverket ditt har en trådløs wifi-tilkobling med dårlig signal og lav kapasitet på den mengden data som kan sendes hvert sekund. Hvis datamengden blir lavere enn 0,5 megabit per sekund, vil du få problemer med for eksempel å strømme videoer. Om du skal se videoer i Ultra-HD-kvalitet, må du ha en datahastighet på 25 Mbit/s.

I en **mobiltelefon** er det også et hovedkort, som i en datamaskin. Hovedkortet i en mobiltelefon har også en radiosender og en mottaker koblet til hovedkortet, slik at den kan kommunisere med **mobilnettet**.

Hvert hus har en rute koblet til internettleverandørens ruter. Mange slike bredbåndsrutere kan sende signaler til hverandre via basestasjoner og nettverksleverandørens sentrale rutere.



Nettverkskommunikasjon	Forklaring
WLAN	Wireless Local Area Network. Trådløst lokalt nettverk.
Wifi	En standard for tilkobling til et trådløst lokalt nettverk.
LAN	Local Area Network. Lokalt nettverk som bruker nettverkskabel til å koble sammen enheter.
Ethernet	Nettverksprotokoll for kommunikasjon mellom enheter i et LAN.
Kabel-TV-kabel	Noen selskaper tilbyr internettoppkobling gjennom kabel-TV-kabelen. Bruker et kabelmodem.
Fiberoptisk kabel	Noen selskaper tilbyr internettoppkobling gjennom en fiberoptisk kabel. Den kan sende store mengder data fort. De bruker et spesielt fiberoptisk kabelmodem som oversetter digitale lyssignaler til digitale strømsignaler.
4G/5G via mobilnett	De fleste smarttelefoner kan kobles til teleleverandørenes trådløse internettoppkoblinger.

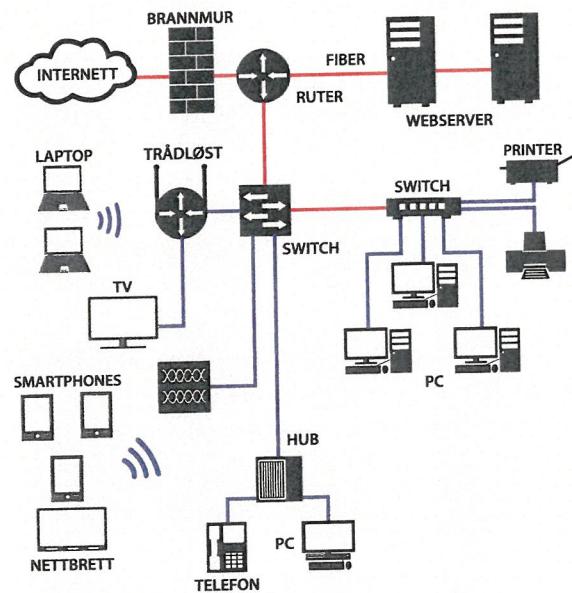
EKSEMPEL

Trafikk og ruter, switch og hub

En ruter knytter flere datamaskiner sammen i et nettverk. Datamaskiner med nettverkskabel eller trådløst nettverkskort kobler seg til ruten for å bli en del av nettverket. Ruten er igjen knyttet til internettleverandørens sentrale rutere. En ruter kjenner til om det fins alternative veier til mottakeren av kommunikasjonen og vil bruke disse alternative veiene dersom den oppdager at det er problemer med den «vanlige» ruten.

Et modem tolker og overfører data mellom ruter og datamaskin. Et modem sørger for at kommunikasjon fra et lokalt nettverk oversettes til noe internettleverandørens nettverk forstår. Bredbåndsrutere fra internettleverandørene har modem innebygget.

En switch mottar datapakker fra en tilkobling og sender dem videre til en av de andre tilkoblingene. En hub ligner på switch, men datapakker som sendes inn i en tilkobling blir sendt videre ut til alle de andre tilkoblingene, samtidig. Er det mye trafikk på nettverket, blir det mange datapakkekollisjoner og tregt nettverk.



EKSTRASTOFF**Webserver, webklienter og skytjenester**

server = tjener

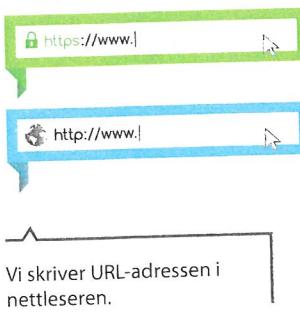
En webserver er en datamaskin som lagrer, prosesserer og leverer websider til webklienter. Webklienter kan være nettlesere, for eksempel Mozilla Firefox, Microsoft Internett Explorer eller Google Chrome. Webklientene kommuniserer med webserveren ved hjelp av protokollen HTTP. Innholdet som sendes til webklienten, er ofte HTML-dokumenter med tekst, bilder og script. En webserver kan også tilby forskjellige tjenester som andre datamaskiner kan benytte. En tjeneste kan være at webserveren tilbyr lagring av bildene dine, eller det kan være en tjeneste for å vise populære filmer. Mange webservere kan plasseres sammen i store datahaller slik at lagringskapasiteten blir større. Skytjenester kan tilby ulike webserver-tjenester. Vi kan leie plass for å lagre bildene våre eller for å bruke programvare vi selv ikke har plass til på vår egen datamaskin. Skyen kan også lagre store mengder data for firmaer.

EN NETTADRESSE TIL EN NETTSIDE

URL (*Uniform Resource Locator*) er adressen til nettsteder på Internett. URL er en kombinasjon av tall og bokstaver som gjør at nettverket finner frem til det vi ønsker, omrent slik et postbud som leverer post, må finne frem til riktig adresse. En URL kan se ut slik: <https://www.cappelendammundervisning.no/>. De ulike delene blir holdt fra hverandre ved hjelp av kolon, skråstreker og punktum.

HTTP (*hypertext transfer protocol*) er en standard måte for å sende **datapakker** med nettsteder. For å gjøre sendingen av datapakker sikrere har vi også fått **HTTPS**. **HTTPS krypterer** innholdet i datapakkene og sørger for at ingen andre kan utgi seg for å være deg eller kan forstå kommunikasjonen din når du for eksempel bruker nettbank eller handler på Internett.

Et **domene** er en adresse til et nettsted som alltid er på formen www.nettstedsnavn.no. Rutene bruker adressen når den skal sende datapakker. Først slår ruten opp i en adressebok som sier hvor det er lurt å sende datapakken for raskest å nå frem til en **webserver** som har dette domenet og det nettstedet vi er ute etter. Slike «adresse-rutere» kalles også **DNS** (*Domain Name System*), og de oversetter www.nettstedsnavn.no til den unike **IP-adressen** som er knyttet til dette domenet. Alle datamaskiner som er koblet til et nettverk, har en unik IP-adresse. IP-adressen består av fire samlinger med numre fra 0 til 255. Eksempelvis er 192.168.1.255 en IP-adresse. IP-adressen er den egentlige adressen til datamaskinen i et nettverk, som Internett. IP-adressene er vanskelige å huske, derfor hjelper domenenavnet oss, for de er lettere å huske.



Vi skriver URL-adressen i nettleseren.

DNS leser domenet bakfra. Sist står landet til nettstedet. For eksempel viser .no at dette ligger på en norsk webserver, så da kan datapakken sendes til en ruter i Norge. På denne ruten ligger det enda en adressebok som vet hvilken webserver den skal videresende datapakkene til. Adressen www.cappelendammundervisning.no er på webserveren til Cappelen Damm Undervisning. På denne webserveren er det igjen installert flere domener på nivå tre. Vi kaller dem **subdomener**. To av disse subdomene er kode.cdu.no og sinus.cdu.no, som er domenene til to ulike nettsteder. Vi behøver ikke www på subdomener.

Åpne en nettside:

1. En bruker skriver internettadressen inn i adressefeltet i nettleseren. Dette tolkes som en forespørsel om å få lese en bestemt nettside som har dette domenet som adresse. De som laget nettsiden, registrerte dette domenenavnet hos en ISP og lagret det i en DNS-ruter.
2. **Ruteren** i lokalnettverket skjønner at adressen ikke er i lokalnettverket, og sender derfor forespørselen til Internett. Et modem sørger for å sende forespørselen om å få lese nettsiden til domenet brukeren skrev til internettleverandøren.
3. Forespørselen går til en DNS som inneholder et kartotek med registrerte domener og tilhørende IP-adresser. Domenet ligger i dette kartoteket siden det ble registrert der tidligere. Se punkt 1.
4. Forespørselen blir sendt til riktig webserver og med riktig IP-adresse.
5. Webserveren behandler forespørselen og sender **datapakker** med nettsiden tilbake til den nettleseren som skrev inn domenet på adresselinjen.
6. Nettleseren får opp datapakkene med nettsiden fra webserveren og viser den.

Begrep	Forklaring	Eksempel
http	Betyr at det er en nettside på Internett.	http://www.cappelendammundervisning.no
https	Betyr at det er en sikker nettside på Internett.	https://www.cappelendammundervisning.no/
domene	En tekstbasert internettadresse som oversettes til en IP-adresse i en DNS	www.cappelendammundervisning.no
IP-adresse	En nettverksadresse med fire grupper av tall mellom 0 og 255	192.168.1.1

EKSTRASTOFF

Brannmur

En **brannmur** er et program som kan beskytte datamaskinen mot at noen med uærlige hensikter får tilgang til datafilene dine eller legger inn et spionprogram eller virus på datamaskinen din. Forskjellige typer programmer, som e-post, nettlesere eller nettverksspill, kommuniserer på litt forskjellige måter med Internett, gjennom såkalte porter. En IP-adresse kan i tillegg til selve adressen også ha med en port. Porten angis ved å sette et kolon og portnummeret etter selve adressen. Et eksempel på en URL med portnummer er <http://www.cappelendammundervisning.no:80>. En brannmur kan være med og begrense hvilke porter som skal være åpne på maskinen, og hvilke programmer som skal få lov til å snakke gjennom disse portene.

Et nettverk inneholder ofte flere brannmurer. Du har gjerne en brannmur installert på datamaskinen din. På ruteren i hjemmenettverket som du kobler deg til Internett via, finns det også en brannmur. Ofte er disse brannmurene stilt inn slik at de er svært strenge med hvilke programmer og andre datamaskiner som får lov å til å sende deg datapakker eller be om data fra datamaskinen din. Hvis en ikke-godkjent datamaskin forsøker å sende data til datamaskinen din, sier brannmuren fra og spør om du vil godkjenne kommunikasjon fra denne datamaskinen. Hvis du svarer ja, kan brannmuren også huske at du stoler på denne datamaskinen til neste gang. Da blir det ikke så mange forstyrrende sikkerhetsspørsmål fra brannmuren senere.

SAMMENDRAG

- > Nesten alt digitalt utstyr er datamaskiner som er bygget opp av de samme hovedkomponentene. I en datamaskin finner du blant annet hovedkort, harddisk (sekundærminne), prosessor (CPU), minnebrikke (hurtigminne eller RAM), grafikk-kort, nettverkskort, strømforsyning og USB-port.
- > Vi bruker binære tall, 1 og 0, for tilstandene av og på i en datamaskin.
- > To bit gir fire muligheter ($2^2 = 4$): 11, 00, 10 og 01, og tre bit gir åtte muligheter ($2^3 = 8$): 000, 001, 010, 011, 100, 101, 110 og 111.
- > Binære tall kalles totallsystemet. Vanlige tall kalles titallsystemet.
- > Binært tall omgjort til vanlig tall: 00 gir $0 + 0 = 0$. Mens 01 gir $0 + 1 = 1$. Ser vi på 10, så har sifferet 1 verdien 2 siden det står først, og det gir $2 + 0 = 2$. Tilsvarende blir $11 = 2 + 1 = 3$, siden begge er slått på og den første har verdien 2 og den neste verdien 1. Ser vi på tre bit, får vi: $101 = 4 + 0 + 1 = 5$.
- > 1 byte = 8 bit, som gir $2^8 = 256$ muligheter. Dette tallet er ofte stort nok til å beskrive en farge, et lydnivå eller til å angi et tegn i et tegnsett. Størrelsen på lagringsplassen i minnet og på en disk blir vanligvis oppgitt i byte.
- > Et bilde deles inn i bitte små deler som kalles piksler eller bildepunkter. Disse bildepunktene har ikke noen fast eller standardisert størrelse. Jo flere punkter du deler et bilde inn i, desto høyere blir oppløsningen og dermed kvaliteten på bildet.
- > Farger kan beskrives ved hjelp av RGB-verdier eller CMYK-verdier.
- > Mediefiler (bilde, lyd, video) tar stor plass. For at datafilene skal ta mindre plass uten at de blir ødelagt, og uten at vesentlig informasjon i for eksempel et bilde går tapt, bruker vi kompresjon.
- > Tapsfri kompresjon er når vi har gjort en fil mindre i størrelse, men likevel har nok data til å gjenskape en fil som er identisk med originalen.
- > Destruktiv kompresjon brukes ofte når det gjelder mediefiler, for eksempel bilder.
- > Tegnsettet i en datamaskin avgjør hvordan bokstavene i alfabetet, tall og andre skrifttegn skal vises på skjermen.
- > Lyd er hurtige endringer i trykk, også kalt lydbølger. Skal vi lagre og behandle lyd i en datamaskin, må lydsignalet digitaliseres. Det må altså gjøres om til tall.
- > Et nettverkskort gjør at datamaskiner kan kobles opp mot et nettverk av andre enheter.

Oppgaver

4.1 Datamaskiner

● 4.01

Lag en liste over hvor mange ulike typer digitalt utstyr du bruker i løpet av en dag.

● 4.02

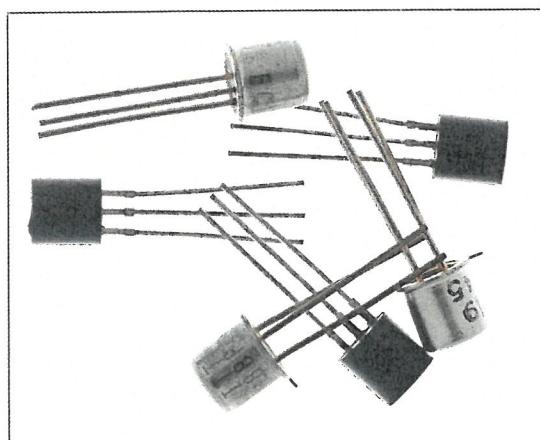
Gjør rede for hvilke hovedkomponenter vi trenger i en datamaskin.

●● 4.03

Gjør rede for hva et operativsystem er og gjør. Finn mer informasjon om operativsystemet på PC-en og mobiltelefonen din.

●● 4.04

En transistor er en komponent som kan kobles i elektriske kretser. Søk på Internett og finn ut hvordan en transistor er bygd opp, og hvordan den virker. Kanskje finner du også ut omtrent hvor mange transistorer som fins i en datamaskin?



Transistorer

● 4.05

a) Hva er forskjellen mellom en HDD (*Hard Disk Drive*) og en SSD (*Solid State Drive*)?

b) Undersøk hvilke komponenter en HDD består av, og hvordan den lagrer data.

●●● 4.06

Finn informasjon om tre–fire forskjellige datamaskiner. Sammenlign spesifikasjonene om komponentene (lagringsplass, RAM, grafikkort, eksterne porter osv.).

4.2 Binære tall – bit og byte

● 4.07

Hvilke to tall representerer tilstanden for strømmen av og strømmen på i en elektronisk datamaskin?

● 4.08

a) Hva kaller vi tallsystemet som er basert på bruken av 0 og 1?

b) Hvilket tall i titallsystemet svarer 10110 til?

● 4.09

Hvor mange bit er det i de binære tallene?

a) 01 b) 110 c) 1100 d) 00110011

● 4.10

Hvilken verdi har de ulike plassene i 8 bit?

● 4.11

a) Hvor mange muligheter har vi med 8 bit?

b) Hva er en byte?

● 4.12

Hva er den minste lagringsenheten?

4.13

Skriv av tabellen nedenfor og fyll ut det som mangler.

Binært tall	Verdi: 128	Verdi: 64	Verdi: 32	Verdi: 16	Verdi: 8	Verdi: 4	Verdi: 2	Verdi: 1	Vanlig tall
00001001	0	0	0	0	1	0	0	1	
	0	0	1	1	0	1	1	0	54
									17
11111111	1	1	1	1	1	1	1	1	
10100110	1	0	1	0	0	1	1	0	
									242
11001000	1	1	0	0	1	0	0	0	
									64

4.3 Bilder består av bildepunkter**4.14**

Hva består et bilde av? Gjør rede for hvordan datamaskinen lagrer bilder.

4.15

Forklar hva RGB står for. Hvordan kan vi angi for eksempel tekstfargen med HTML- og CSS-kode?

4.16

Hva er tapsfri kompresjon?

4.17

Hva er destruktiv kompresjon?

4.18

Bruk dine egne ord og forklar for en medelev hvordan overføring av et bilde foregår.

4.4 Tegnsett**4.19**

Hva er et tegnsett?

4.20

Hvorfor skriver vi koden

`<meta charset=" UTF-8" >`?

4.21

Hva er Unicode og ANSI?

4.22

Finn det heksadesimale tallet.

- a) 110 0001
- b) 110 0100
- c) 0111 1010

4.23

Hvilket tegn er a, b og c i oppgave 4.21?

4.5 Digitale lydopptak**4.24**

Gjør rede for hvordan vi kan digitalisere lyd.

4.25

Hvor mye plass trenger vi for å lagre én time med ukomprimert lyd? Ta da utgangspunkt i standarden som krever en samplingsfrekvens på 44,1 kHz og en oppløsning på 16 bit per sampel.

4.26

Fortell en medelev det du vet om lydkvalitet.

4.6 Video og animasjon

● 4.27

Hva er en video?

● 4.28

Hvordan regner vi oss frem til datamengden?

4.29

Forklar hvordan komprimering med nøkkelsbilder fungerer.

● 4.30

Hvorfor er komprimering av video viktig?

4.7 Nettverk

● 4.31

Hva er et hjemmenettverk, og hvordan fungerer det?

● ● 4.32

Hvordan fungerer nettverkskommunikasjon?

● ● 4.33

Forklar hvordan Internett fungerer ved å beskrive for en medelever gangen fra registreringen av et domene til at en nettside vises på din datamaskin.

● ● 4.34

- a) Gjør rede for hvordan mobilnett fungerer.
- b) Hva er forskjellen på 4G og 5G?
- c) Hvilke teknologier ble brukt før 4G og 5G?

● ● 4.35

Hvordan tror du digitalt utstyr i transportsektoren vil påvirke «smarte byer» i fremtiden? Skriv en tekst om det.

● 4.36

Hva er en IP-adresse?

Skriv IP-adressen 195.88.55.16 i nettleservinduet og trykk Enter. Hvor kommer du?

● ● ● Prosjektoppgave

Ordet "Smart" dukker stadig opp i nye sammenhenger når ny teknologi introduseres på markedet. Først var det "Smart-telefon", så kom "Smart-TV" og "Smart-hjem". Mye av det vi tenker på som smart i dag, er uløselig knyttet til kommunikasjonen mellom tingene vi eier og bruker. Hvis det er varmt i været, kan vi skru på kjøleanlegget på bilen med mobilen på forhånd, så vi slipper å bli svette. Hvis vi lurer på hvor en venn er, kan vi sjekke det på telefonen, og hvis vi har mistet telefonen, kan vi finne den igjen ved å spore den fra en annen enhet.

Terskelen for å bli sporet er synkende, men fortsatt er sporingen avhengig av at vi faktisk går rundt med telefonen i lomma. Hvis batteriet går tomt, eller telefonen faller ut av lomma, blir det straks vanskeligere å finne oss. Ville det ikke vært en god idé om vi fikk implementert en liten chip i kroppen som frigjorde oss for bindingen med telefonen, slik at uansett om batteriet er flatt eller om telefonen gikk i bakken og sluttet å virke, kunne venner og familie finne ut hvor vi var.

a) Hvilke fordeler og ulemper kunne en slik løsning ha medført?

b) Se for deg en fremtid hvor alle kan spores. Hvordan kunne vi ha utnyttet en slik teknologi til noe positivt?

c) Utforsk noe du syns er interessant fra tematikken i oppgaveteksten eller kapittelet forøvrig. Du kan lage en presentasjon, skrive en artikkel eller lage en nettside der du dokumenterer det du har funnet ut.