Zahlensysteme

- Sie kennen nun bereits vier Zahlensysteme:
  - Dezimalsystem Basis 10 Ziffern 0 – 9
  - Dualsystem Basis 2 Ziffern 0, 1
  - Oktalsystem Basis 8 Ziffern 0 – 7
  - Hexadezimalsystem Basis 16 Ziffern 0 – 9, A – F
- Das Ganze lässt sich auf beliebige Basen verallgemeinern: Basis *n* mit *n* Ziffern; z. B. Basis 32 mit Ziffern 0 – 9, A (10) – V (31)
- Beispiele: Basen 12 und 100

Das 12er-System (1/2)

- Sinnvoll in der Informatik:
  2er (Dual-), 8er (Oktal-) und 16er (Hex.-) System
- Prinzipiell sind Zahlensysteme mit beliebiger Basis möglich, z. B. 12 oder 13
- Nur zum Spaß: Das 12er-System („Duodezimalsystem“) – das niemand verwendet
  - Basis 12
  - 12 Ziffern: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (10), B (11)

Das 12er-System (2/2)

- Notation:
  $12A4_{12} = 1 \times 12^3 + 2 \times 12^2 + 10 \times 12^1 + 4 \times 12^0$
  $= 1 \times 1728 + 2 \times 144 + 10 \times 12 + 4 \times 1$
  $= 2140$
- Alle Umrechnungen (12er-System ↔ Dezimalsystem) funktionieren wie bei den übrigen Zahlensystemen
-

Besser lesbar: 100er-System

- Leichter lesbar sind Zahlen im 100er-System, das 100 Ziffern (00, 01, 02, ..., 98, 99) besitzt:
  - $01\ 34\ 26_{100} = 13426$,
  - $01_{100} = 1$, $00_{100} = 0$, $1\ 00_{100} = 100$
- Zur einfachen „Umrechnung“ zwischen 100er- und 10er-System:
  - gleiches Prinzip wie bei Umrechnung 8er ↔ 2er ↔ 16er Systeme
  - Jede Ziffer des 100er Systems über Tabelle (oder Hingucken) in Ziffern des 10er Systems umwandeln

Kodierung: Übungen (1/2)

1. Stellen Sie den folgenden Text:
   `Informatik-Grundlagen 2008/09`
   - im Dezimalformat (normales 10er-System),
   - im Hexadezimalformat,
   - im Oktalformat
   - und das erste Wort (Informatik) im Dualformat dar.

   Dazu benötigen Sie:
   - ASCII-Tabelle (1. Schritt)
   - Hex/Oktal/Dual-Tabellen (2. Schritt)

Kodierung: Übungen (2/2)

2. a) Wie viele 16-stellige Bitfolgen gibt es?
   b) Wie viele verschiedene Worte („Doppel-Bytes“) gibt es?
3. Rechnen Sie die Hexadezimalzahl $\mathbf{A01F_h}$ in Dual- und Oktaldarstellung um.
4. Wenn Sie wissen, dass $\mathbf{FF_h} = \mathbf{100_h} - \mathbf{1}$ gilt, wie können Sie dann schnell $\mathbf{FFF_h}$ und $\mathbf{FFFF_h}$ in Dezimalzahlen umrechnen?

Zahlensysteme: Fragen (1/2)

- Aus welchen Zahlensystemen kann die folgende Zahl stammen? Aus welchen nicht? Warum? **41823**
  - Berücksichtigen Sie: Dual-, Oktal-, Dezimal- und Hexadezimalsystem.
- Können Sie **2049** (dezimal) ganz schnell in eine Dualzahl umrechnen?
- Wie sieht das Wort „Übung“ in ASCII-Darstellung aus? (Achtung: Fangfrage)

Zahlensysteme: Fragen (2/2)

- Was ist größer: $\mathbf{74_h}$ oder **75** ?
- Wie viele Hexadezimalzahlen liegen zwischen $7A7_h$ und $8B8_h$? (Zählen Sie die beiden „begrenzenden“ Zahlen mit.)

Größenordnungen

Name (Symbol)	SI-konforme Bedeutung[1)]	„klassische“ Bedeutung	Unterschied
Kilobyte (kB)	10^3 Byte = 1.000 Byte	2^{10} Byte = 1.024 Byte	2,4 %
Megabyte (MB)	10^6 Byte = 1.000.000 Byte	2^{20} Byte = 1.048.576 Byte	4,9 %
Gigabyte (GB)	10^9 Byte = 1.000.000.000 Byte	2^{30} Byte = 1.073.741.824 Byte	7,4 %
Terabyte (TB)	10^{12} Byte = 1.000.000.000.000 Byte	2^{40} Byte = 1.099.511.627.776 Byte	10,0 %
Petabyte (PB)	10^{15} Byte = 1.000.000.000.000.000 Byte	2^{50} Byte = 1.125.899.906.842.624 Byte	12,6 %
Exabyte (EB)	10^{18} Byte = 1.000.000.000.000.000.000 Byte	2^{60} Byte = 1.152.921.504.606.846.976 Byte	15,3 %
Zettabyte (ZB)	10^{21} Byte = 1.000.000.000.000.000.000.000 Byte	2^{70} Byte = 1.180.591.620.717.411.303.424 Byte	18,1 %

[1)] SI: Internationales Einheitensystem (Système International d'Unités) Quelle: Wikipedia, „Byte“

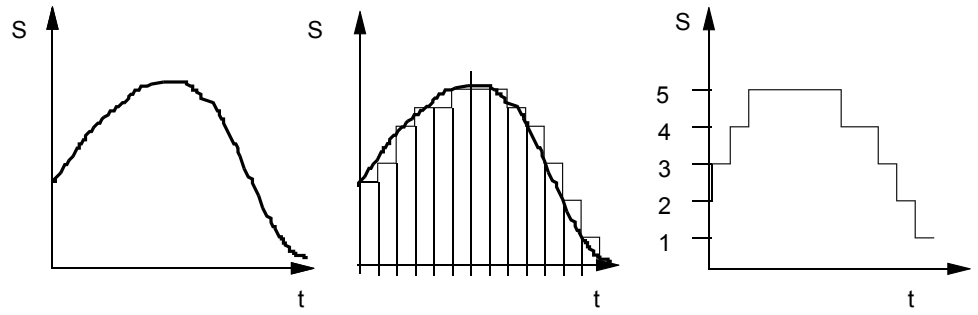
Informationen speichern

- Bisher: Einfache Texte (z. B. im ASCII- oder ISO-8559-15-Format) in Bytes gespeichert
- Was tun mit analogen Daten?
  - Wie speichert ein digitaler Fotoapparat Bilder?
  - Wie speichert ein MP3-Player Musik?
  - Was unterscheidet Schallplatte und CD?
- Digitalisierung von analogen Daten

Digitalisierung analoger Daten

- **Rasterung** (**Diskretisierung**): Abtasten (Sampling) an diskreten zeitlichen oder örtlichen Punkten
  - diskret = nur endlich viele Werte
  - Gegenteil: kontinuierlich, z.B. Werte aus $\mathbb{Q}$
- **Quantisierung** der abgetasteten Werte (runden auf wenige mögliche Werte)
- **Digitalisierung** (Kodierung): Darstellung des abgetasteten und quantisierten Signals als Digitalcode

Zeitliche Digitalisierung

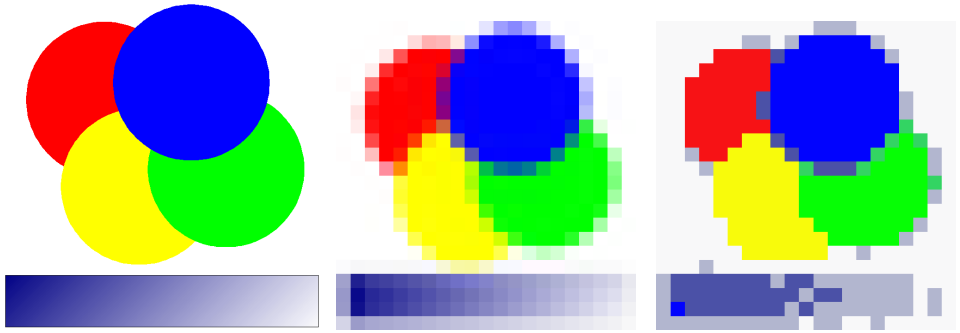


analoges Signal

„gesampelt“: Proben zu festen Zeitpunkten gezogen – noch mit exakten Werten

quantisiert: alle Messwerte auf 1, 2, 3, 4, 5 gerundet

Örtliche Digitalisierung



gleichmäßige Färbung über ganze Bereiche

nur begrenzt viele (diskrete) Farbabstufungen

Codierung von Farbinformation: Die Farbe eines Bildpunktes ist eine additive Mischung der 3 Grundfarben Rot, Grün und Blau → für Darstellung des Farbbildes wird die Rot-, Grün- und Blau-Intensität für jeden Bildpunkt separat digitalisiert.

Gebräuchliche Datenformate

- Text: ASCII/ANSI/Unicode (.txt), RTF (.rtf), Word (.doc), OpenOffice (.odt)
- Tabellen: Excel (.xls), OpenOffice (.ods)
- Dokumente: .doc, .pdf, HTML (.html / .htm), …
- strukturierte Daten: XML
- Rastergrafik: Bitmap (.bmp), Graphics Interchange Format (.gif), JPEG, TIFF, PNG
- CAD: VDA-FS, IGES, STEP
- Audiodateien: MP3 (.mp3), Ogg Vorbis (.ogg)
- Bildfolgen (Video): MPEG

Alltagsthemen der Kodierung

- Reduktion der Datenmenge durch Kompression
  - verlustfrei
  - verlustbehaftet
- Fehlerhafte Darstellung aufgrund ungleicher Zeichenvorräte bei Sender und Empfänger (z. B. bei Darstellung von nationalen Sonderzeichen)
- Fehlerhafte Übertragung
  → Codes mit Fehlererkennung, Fehlerkorrektur
- Verschlüsselung, Signatur

Kompression (1/3)

- Einfaches Verfahren:
  - „AAAAABBCCCCCCCCDD“ → „5A2B8C2D“ (kürzer)
  - aber: „ABCDAB“ → „1A1B1C1D1A1B“ (länger)
- verlustbehaftet:
  - „Informatik-Grundlagen“ → „informatik grundlagen“ (kleineres Alphabet: a-z + Leerzeichen = 27 Zeichen, weniger Bits pro Zeichen)
  - „Informatik-Grundlagen“ → „NFRMTKGRNDLGN“ (gleiches Prinzip; hier: Verzicht auf Vokale)

Kompression (2/3)

- Besseres Verfahren: Buchstabenhäufigkeiten
  - Welche Buchstaben kommen wie häufig vor? Reihenfolge („top ten“) festlegen, z. B. a, e, i, o, u, n, t, s, r, p, k, l, m, ..., q, y
- Dann Kodierung
  - von „häufigen“ Buchstaben in kurze Bitfolgen
  - und von „seltenen“ Buchstaben in lange Bitfolgen
  - z. B. a → 000, e → 001, i → 010, o → 011, u → 1000, n → 1001, t → 1010, s → 1011, r → 11000, k → 11001, l → 11010, m → 11011, ..., q → 11111001, y → 11111010

Kompression (3/3)

- Beispiele mit obiger Tabelle:
  - „nein“ → 1001 001 010 1001
  - „test“ → 1010 001 1011 1010
  - „mlml“ → 11011 11010 11011 11010
  - „qyqy“ → 11111001 11111010 11111001 11111010
- Je „unwahrscheinlicher“ ein Wort, desto länger seine Kodierung

Fehlererkennung (1/3)

- Übertragung von digitalen Daten (etwa über eine Telefonleitung oder ein Datenkabel zwischen zwei PCs) ist oft fehlerhaft:
  - einzelne Bits „kippen“ bei der Übertragung
  - einzelne Bits gehen bei der Übertragung komplett verloren
- Fehlererkennung: Übertragen von zusätzlichen Informationen, mit denen Fehler erkannt werden

Fehlererkennung (2/3)

- Einfachste Möglichkeit: Alles doppelt senden
  - Empfänger vergleicht die beiden empfangenen Infomationen, die identisch sein sollten – sind sie es nicht, hat es einen Fehler gegeben
  - Wahrscheinlichkeit, dass die Daten zweimal mit exakt demselben Fehler übertragen werden, ist klein
  - Verfahren ist aber sehr aufwendig:
    - Verdopplung der übertragenen Datenmenge
    - d. h. Halbierung der Übertragungsgeschwindigkeit

Fehlererkennung (3/3)

- Idee: Daten mit **Prüfsummen** versehen
  - einfaches Beispiel: 7 Bit (ASCII-Zeichen) um sog. **Paritäts-Bit** ergänzen:
  - Im ASCII-Code ist das höchstwertige (ganz links) stehende Bit immer 0
  - Verwende dieses Bit wie folgt:
    - Wenn die Summe der übrigen Bits ungerade ist, setze es auf 1
    - Wenn die Summe der übrigen Bits gerade ist, setze es auf 0

ASCII + Paritätsbit

	ASCII	ASCII binär (nur 7 Bit)	Anzahl Einsen	ASCII + Paritätsbit
A	65	01000001	gerade	**0**1000001
B	66	01000010	gerade	**0**1000010
C	67	01000011	ungerade	**1**1000011
D	68	01000100	gerade	**0**1000100
E	69	01000101	ungerade	**1**1000101

- In den Bytes **mit** Paritätsbit: Anzahl Einsen immer gerade
- „Kippt“ ein Bit, fällt der Fehler auf
- Kippen zwei Bits, dann nicht...

Übung: ASCII-Fehlerkorrektur

- Sie haben folgende Nachricht (binär) erhalten: 01001000, 11100001, 11101100, 11101110, 01101111
- Prüfen Sie zunächst, welche Zeichen ohne „Bit-Kipper“ übertragen wurden
  - Für alle korrekt empfangenen schlagen Sie in der ASCII-Tabelle den zugehörigen Buchstaben nach,
  - für alle fehlerhaft empfangenen setzen Sie ein Fragezeichen ein.

65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g
72	H	104	h
73	I	105	i
74	J	106	j
75	K	107	k
76	L	108	l
77	M	109	m
78	N	110	n
79	O	111	o
80	P	112	p
81	Q	113	q
82	R	114	r
83	S	115	s
84	T	116	t
85	U	117	u
86	V	118	v
87	W	119	w
88	X	120	x
89	Y	121	y
90	Z	122	z