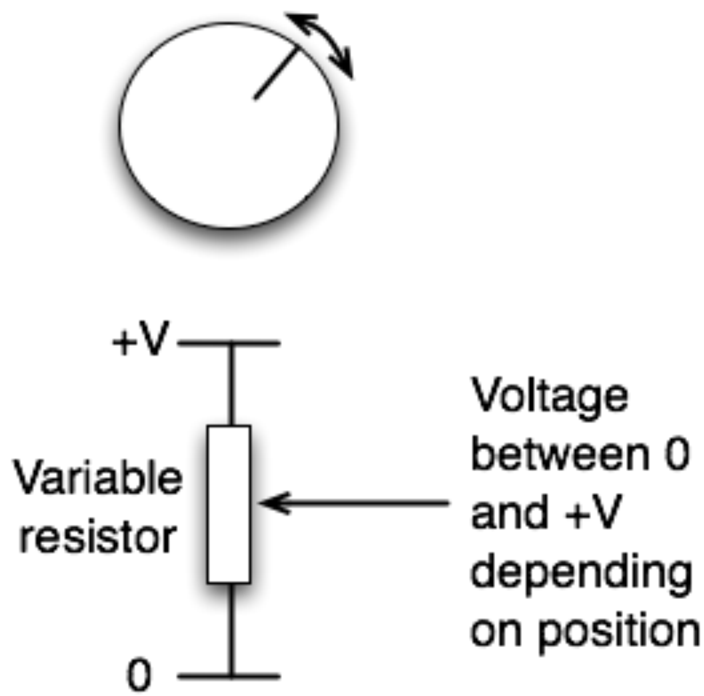
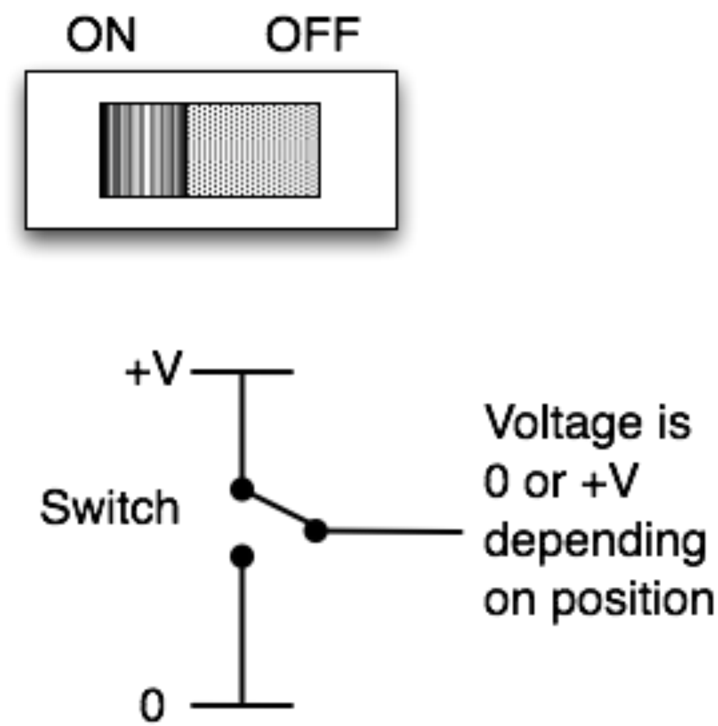
Introduction to digital systems

Juan P Bello

Analogue vs Digital (1)

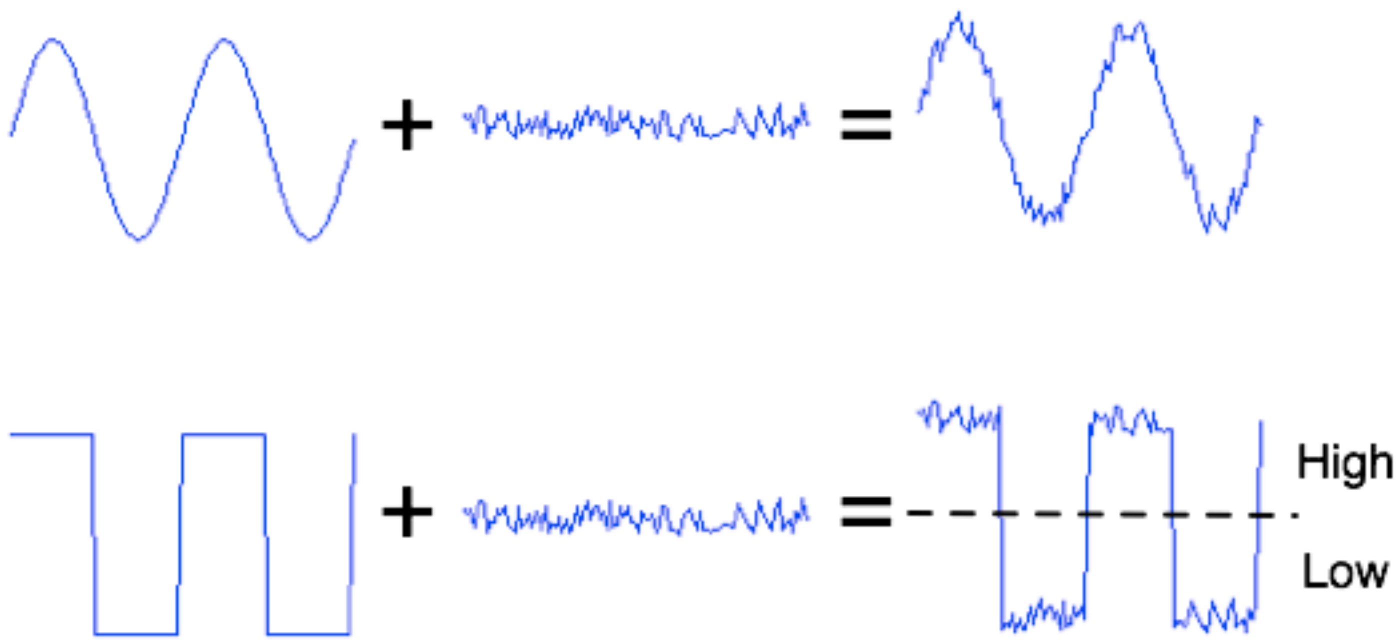
- Analog information is made up of a continuum of values within a given range
- At its most basic, digital information can assume only one of two possible values: one/zero, on/off, high/low, true/false, etc.





Analogue vs Digital (2)

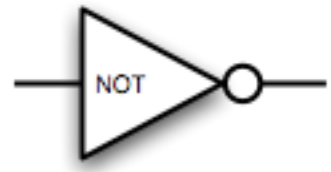
- Digital Information is less susceptible to noise than analog information
- Exact voltage values are not important, only their class (1 or 0)
- The complexity of operations is reduced, thus it is easier to implement them with high accuracy in digital form
- BUT: Most physical quantities are analog, thus a conversion is needed

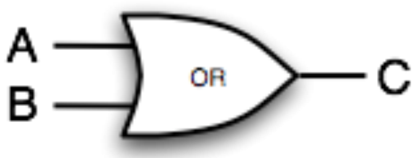


Logical operations (1)

Truth table

A	B	C
0	0	0
1	0	0
0	1	0
1	1	1



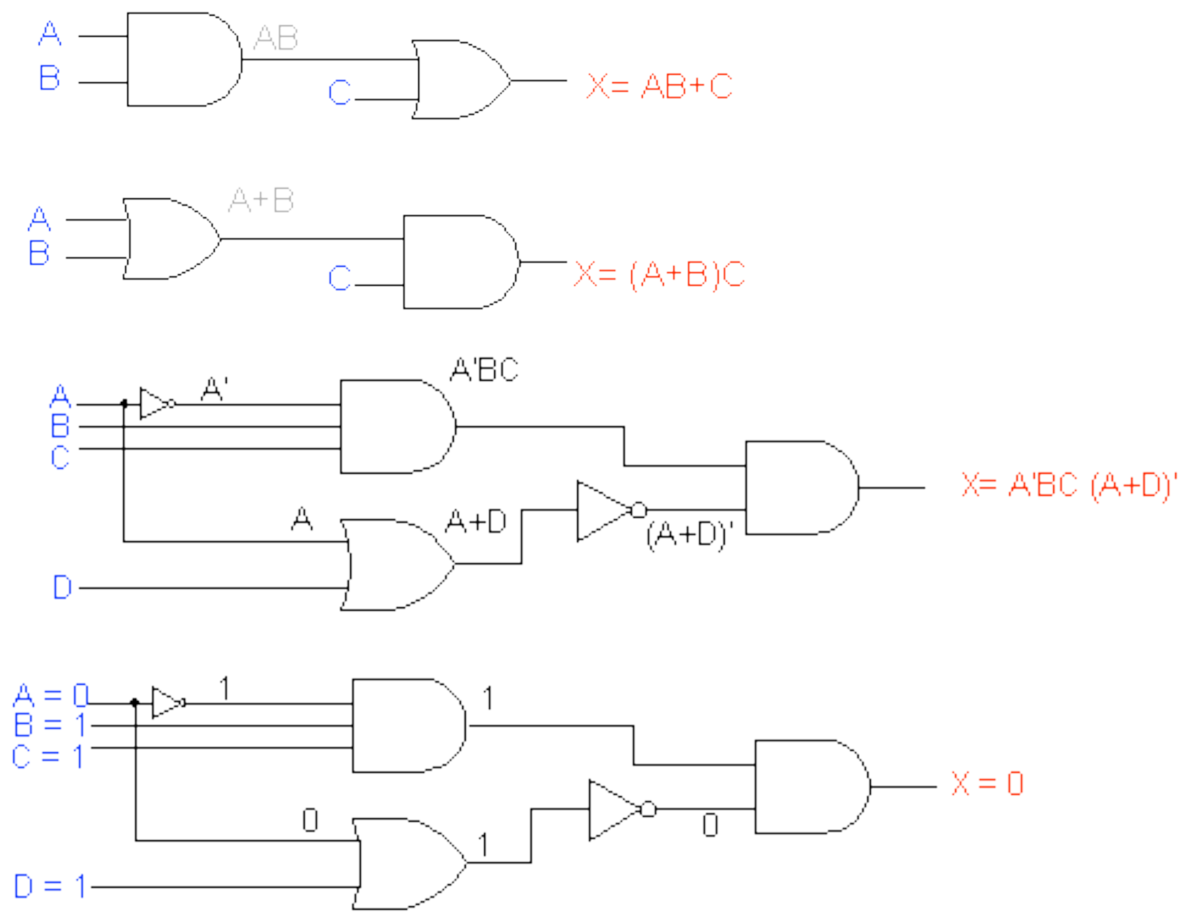


A	B	C
0	0	0
1	0	1
0	1	1
1	1	1

A	B	C
0	0	0
1	0	1
0	1	1
1	1	0

Logical operations (2)

- These logic gates are the basic building blocks of all digital systems



Numeral Systems

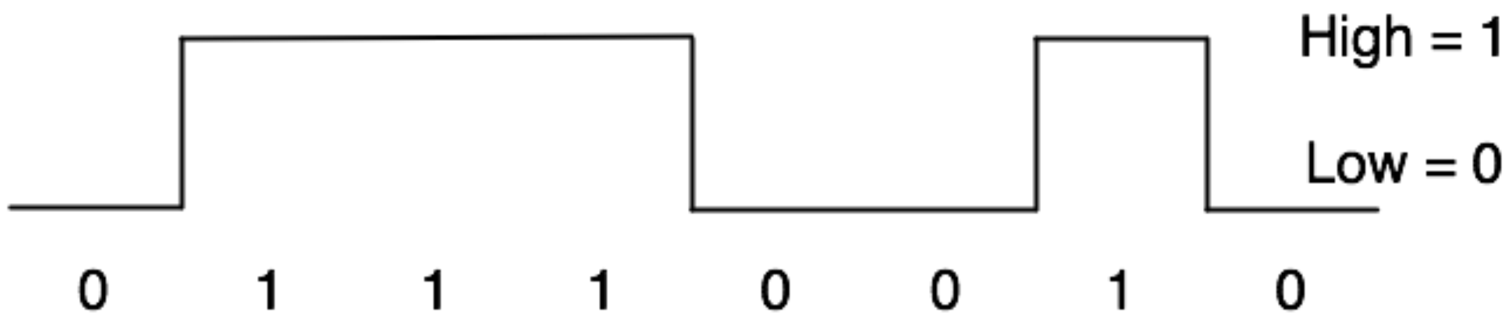
- Notation system using a limited set of symbols to express numbers uniquely
- We are most familiar with the positional, base-10 (decimal), Hindu-Arabic numeral system:

0123456789

- Least significant digit, to the right, assumes its own value. As we move to the left, the value is multiplied by the base.

Binary system (1)

- Digital systems represent information using a binary system, where data can assume one of only two possible values: zero or one.
- Appropriate for implementation in electronic circuitry, where values are characterized by the absence/presence of an electrical current flow.



- Pulse code modulation (PCM) is used to represent binary numbers electrically, as a string of high and low voltages

Binary system (2)

- The binary system represents numbers using *bi*nary digi*ts* (*bits*) where each digit corresponds to a power of two.

binary	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
Power of two	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
decimal	128	64	32	16	8	4	2	1

- The total (in decimal) is *128 + 64 + 32 + 4 + 1 = 229*
- Since we begin counting from zero, *n* bits can represent 2^n values: from 0 to 2^n-1 inclusive (e.g. 256 values, from 0 to 255, for 8 bits).
- Groups of bits form binary words

Binary system (3)

MSB x 2^7

LSB x 2^0

binary	**1**	**1**	**1**	**0**	**0**	**1**	**0**	**1**
Power of two	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
decimal	128	64	32	16	8	4	2	1

binary word

4 bits = 1 *nybble*

4 bits = 1 *nybble*

8 bits = 1 *byte (from by eight)*

Unit	Value
Kilobyte (KB)	1024 Bytes
Megabyte (MB)	1024 KBytes
Gigabyte (GB)	1024 MBytes
Terabyte (TB)	1024 GBytes

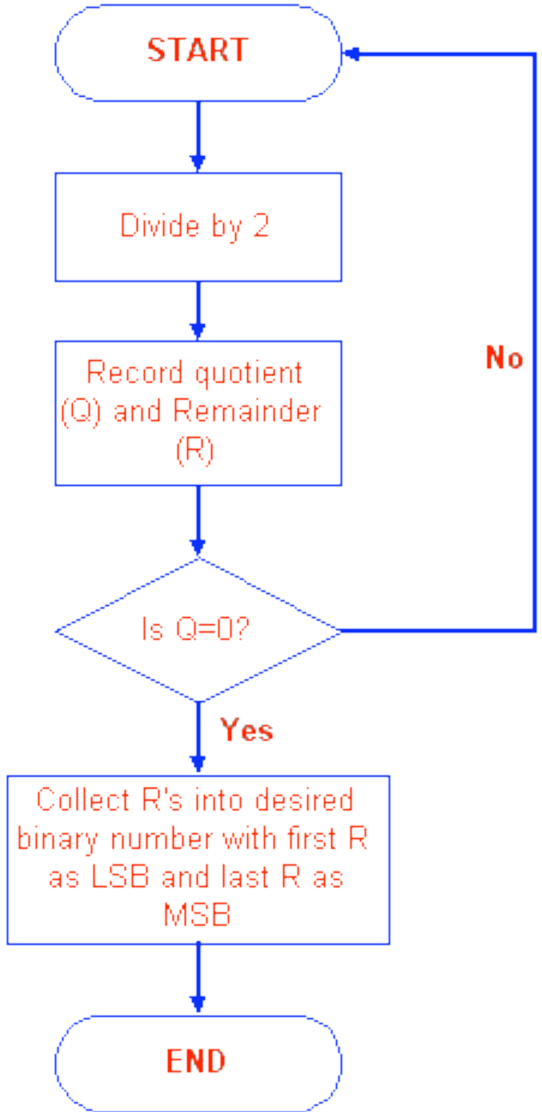
byte *byte* *byte*

MSbyte

LSbyte

Binary system (4)

- How to convert from decimal to binary?
- *Repeat division* by 2
- Example: Convert 29_{10} to binary
  - 29/2 = 14 remainder 1 (LSB)
  - 14/2 = 7 remainder 0
  - 7/2 = 3 remainder 1
  - 3/2 = 1 remainder 1
  - 1/2 = 0 remainder 1 (MSB)
- 29_{10} => 11101_2



Octal system

- To avoid writing down long binary words, it is often easier to use larger base systems. Two commonly-used systems are octal and hexadecimal.
- The octal number system is base eight, i.e. values can be represented using an 8-symbol dictionary: 0-7
- To convert from binary to octal, binary numbers are grouped on 3-bits words such that: $000_2 = 0_8$, 001 = 1, 010 = 2, 011 = 3, 100 = 4, 101 = 5, 110 = 6, and 111 = 7
- To convert from octal to decimal: $24_8 = 2x8^1 + 4x8^0 = 20_{10}$
- From decimal to octal (and from there to binary), Repeat divide by 8:
  - 20/8 = 2 remainder 4 (LSB)
  - 2/8 = 0 remainder 2 (MSB)
  - $20_{10} = 24_8$

Hexadecimal numbers

- The hexadecimal number system (AKA hex) is a base 16 notation. It is the most popular large-base system for representing binary numbers.
- Values in MIDI implementation charts are often expressed as hexadecimal numbers.
- Each symbol represents 4-bits (1 nybble), that can take one of 16 different values: the values 0-9 are represented by the digits 0-9, and the values 10-15 are represented by the capital letters A-F.
- Conversions are performed as with the other number systems.

Bin	**Hex**	**Dec**	**Bin**	**Hex**	**Dec**	**Bin**	**Hex**	**Dec**	**Bin**	**Hex**	**Dec**
0000	0	0	0100	4	4	1000	8	8	1100	C	12
0001	1	1	0101	5	5	1001	9	9	1101	D	13
0010	2	2	0110	6	6	1010	A	10	1110	E	14
0011	3	3	0111	7	7	1011	B	11	1111	F	15

A/D Conversion (1)

- The conversion of an analog (continuous) voltage x(t) into a discrete sequence of numbers x(n) is performed by an Analog-to-digital Converter (ADC)

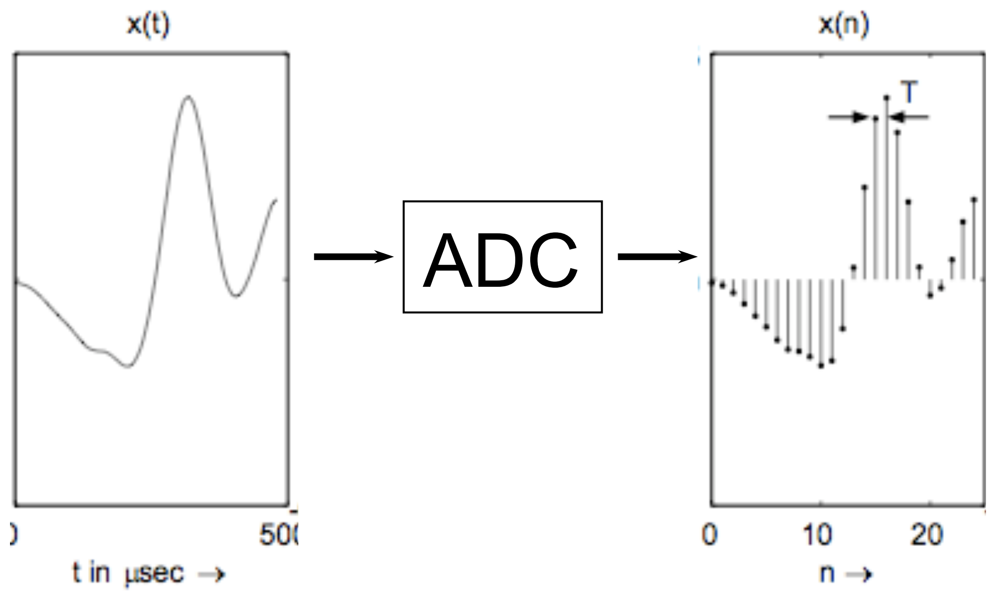
- The ADC samples the amplitude of the analog signal at regular intervals in time, and encodes (quantizes) those values as binary numbers.

- The regular time intervals are known as the sampling period (T_s) and are determined by the ADC clock.

- This period defines the frequency at which the sampling will be done, such that the sampling frequency (in Hertz) is:

$$fs = 1/T_s$$

A/D Conversion (2)

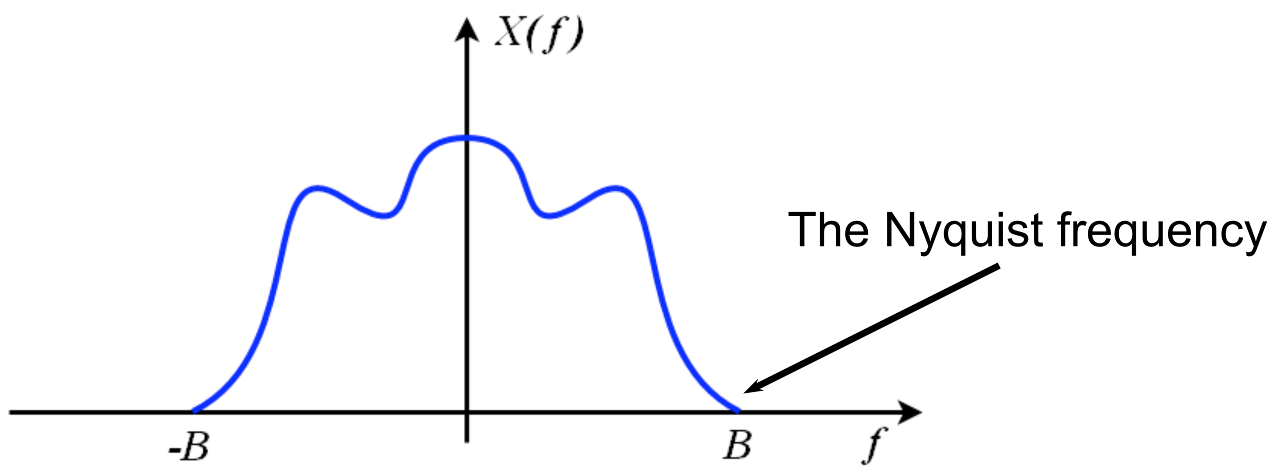


- The outgoing sequence x(n) is a discrete-time signal with quantized amplitude
- Each element of the sequence is referred to as a sample.

$$\ldots, x[n-1], x[n], x[n+1], \ldots$$

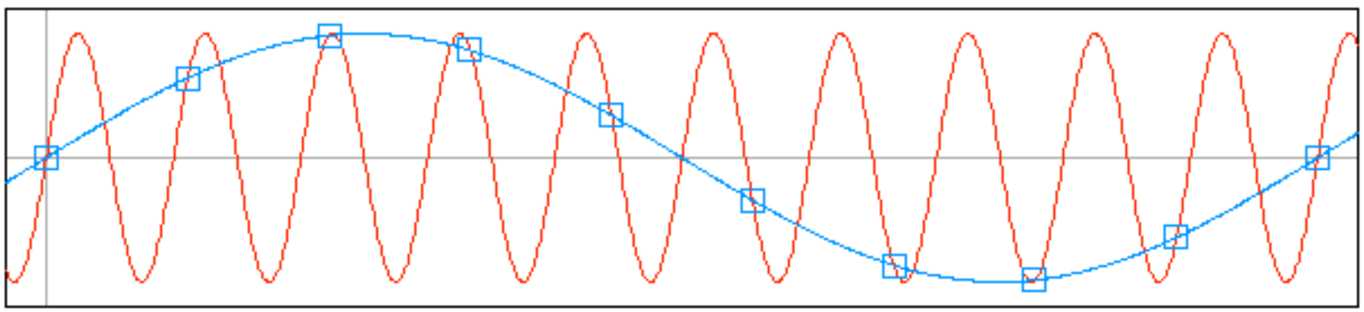
A/D Conversion (3)

- Sampling is the process of converting a continuous signal into a discrete sequence
- Our intuition tells us that we will loose information in the process
- However this is not necessarily the case and the **sampling theorem** simply formalizes this fact
- It states that “in order to be able to reconstruct a *bandlimited* signal, the sampling frequency must be at least *twice* the highest frequency of the signal being sampled” (Nyquist, 1928)

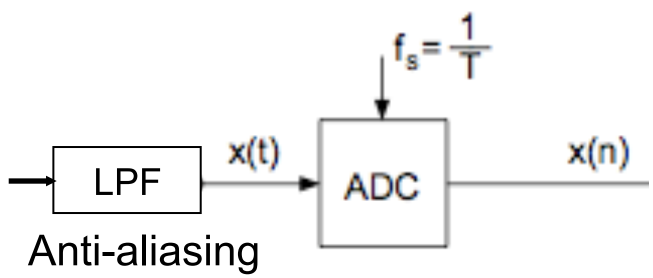


A/D Conversion (4)

- What happens when $fs < 2B$
- There is another, lower-frequency, signal that share samples with the original signal (aliasing).

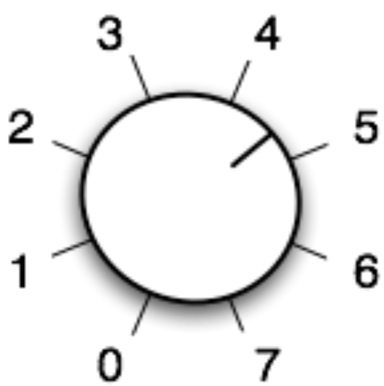


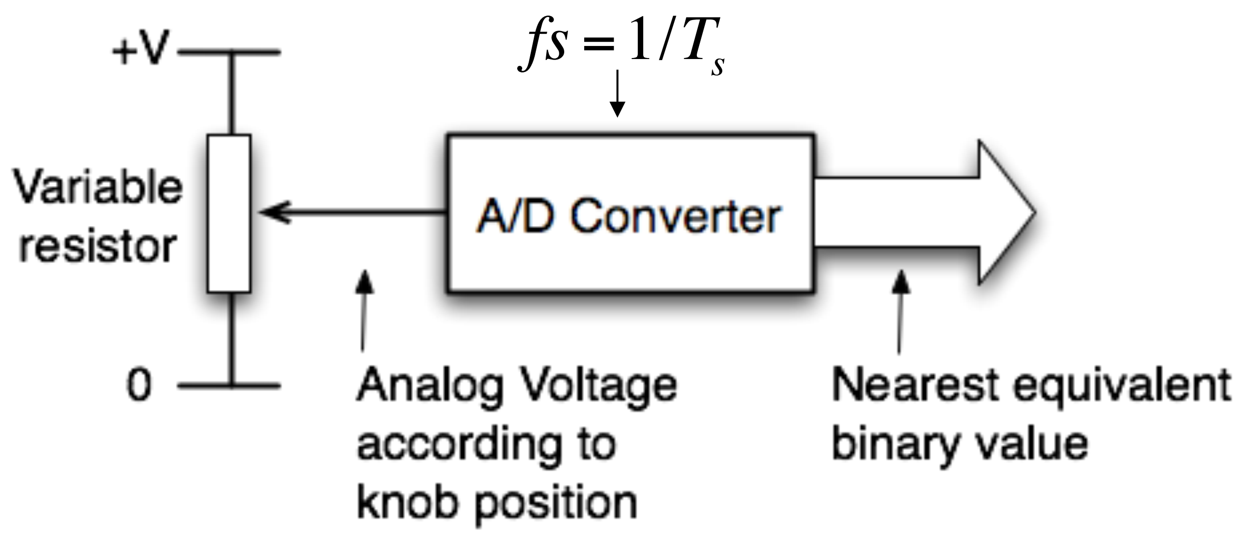
- Related to the wagon-wheel effect: http://www.michaelbach.de/ot/mot_strob/index.html



A/D Conversion (5)

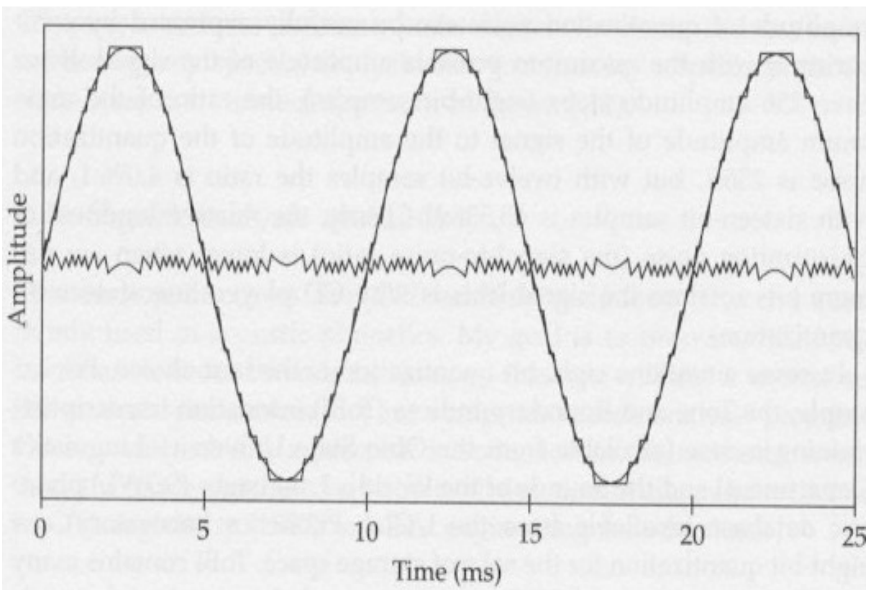
- The accuracy of the quantization depends on the number of bits used to encode each amplitude value from the analog signal.
- Example: to quantize the position of a control knob, it is necessary to determine the nearest point of the scale
- This conversion implies an error (max. half a point)

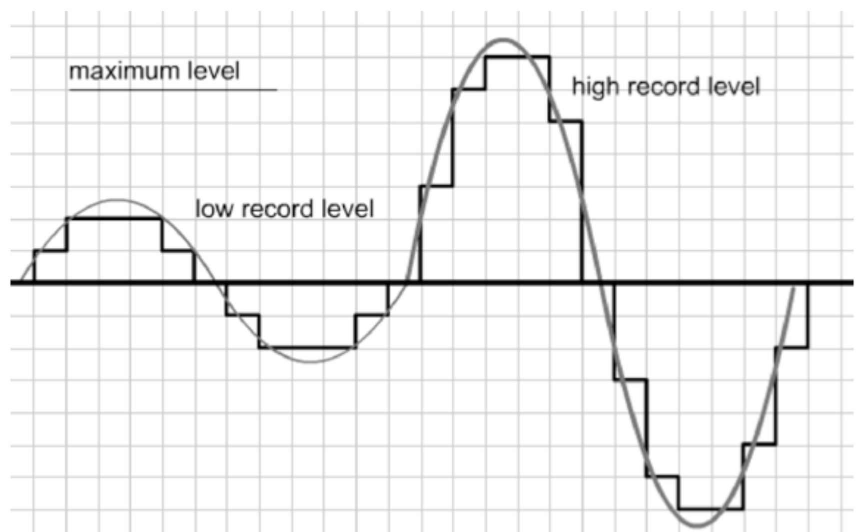




A/D Conversion (6)

- Quantization error: is the distortion produced by the rounding-up of the continuous values of the analog signal during the ADC process to the values "allowed" by the bit-resolution of each sample.
- This depends on the quantization accuracy (# of bits)

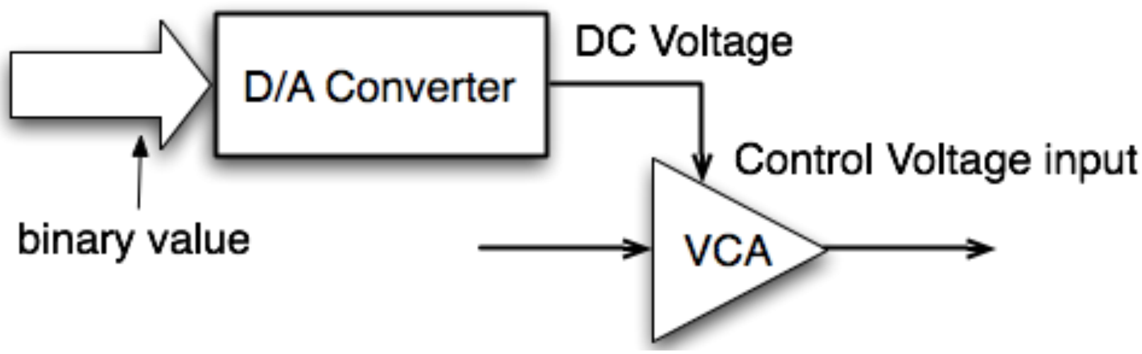


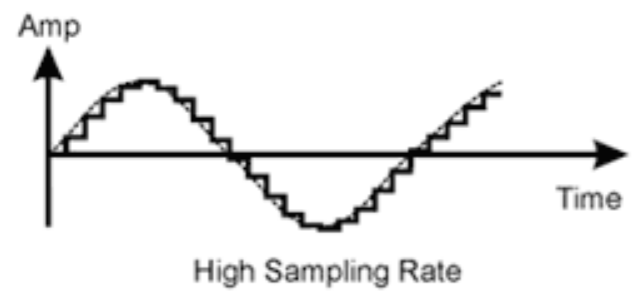


- Example: a sound with progressively worsening quantization noise:

D/A Conversion

- Just as we used an ADC to go from x(t) to x(n), we can turn a discrete sequence into a continuous voltage level using a digital-to-analog converter (DAC).





- However, the quantized nature of the digital signal produces a "Zero-Order Hold" effect that distorts the converted signal, introducing some step (fast) changes (know as *imaging)*.
- To avoid this, we use an anti-imaging filter (AKA smoothing or reconstruction filter) that smoothes out those fast changes.

Useful References

- Francis Rumsey (1994). "MIDI Systems and Control", Focal Press.
  - Chapter 1: An Introduction to Computer Systems and Terminology
- Francis Rumsey and Tim McCormick (2002). "Sound and Recording: An Introduction", Focal Press.
  - Chapter 13: MIDI
- Peter Lau (2001). "Digital System Tutorial on the Web", University of Sydney: http://www.eelab.usyd.edu.au/digital_tutorial/toc.html
- Curtis Roads (1996). "The Computer Music Tutorial", The MIT Press