CONVERSIONE ANALOGICO/DIGITALE (ADC)

La conversione analogico/digitale è il processo che permette di trasformare un segnale analogico in uno digitale. Inevitabilmente una parte delle informazioni contenute nel segnale analogico di partenza viene persa, ma, se il processo è svolto rispettando alcune condizioni, tale perdita risulta del tutto irrilevante.

La conversione A/D si compone di tre fasi:

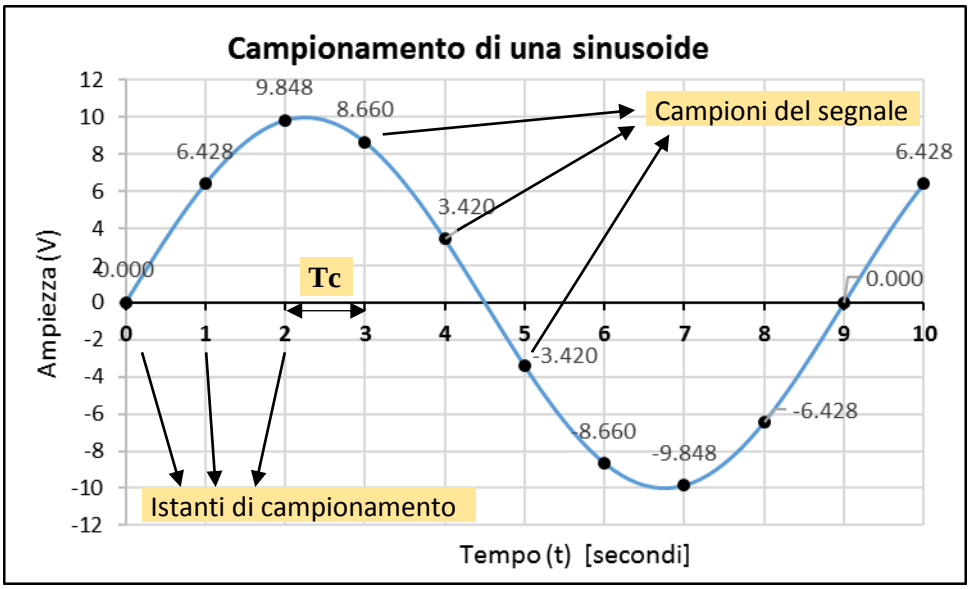
1. **Campionamento;**
2. **Quantizzazione;**
3. **Codifica.**

1. CAMPIONAMENTO (in inglese *"sampling"*)

"Campionare un segnale" significa **prelevare da esso una successione di valori** ad istanti ben precisi detti "**istanti di campionamento**".

L'intervallo di tempo tra due istanti di campionamento si chiama **periodo di campionamento** (**Tc**) ed il suo reciproco si chiama **frequenza di campionamento** (**fc**). Come già detto per i segnali:

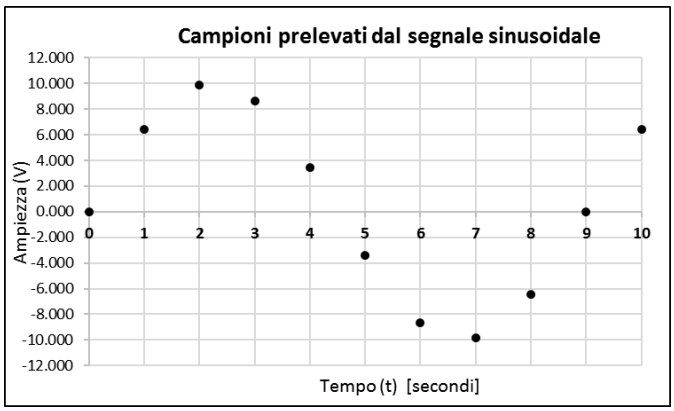
$f_c = \frac{1}{T_c}$ e quindi $T_c = \frac{1}{f_c}$



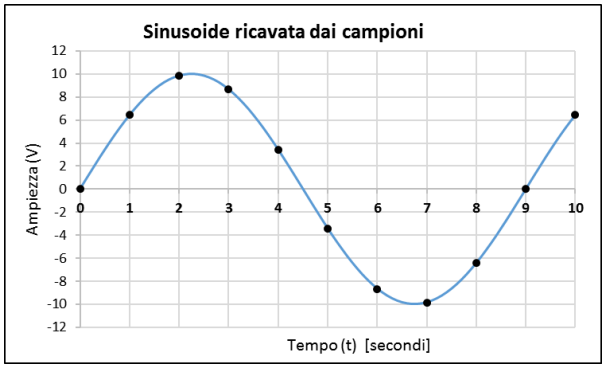
In figura è rappresentato un segnale sinusoidale campionato. Periodo del segnale sinusoidale T = 9 secondi, quindi f = 0,11 Hz. Periodo di campionamento Tc = 1s, quindi fc = 1Hz. Ad ogni periodo del segnale vengono estratti 9 campioni.

Degli infiniti valori del segnale analogico ne vengono presi solo alcuni, questo provoca una perdita di informazioni?

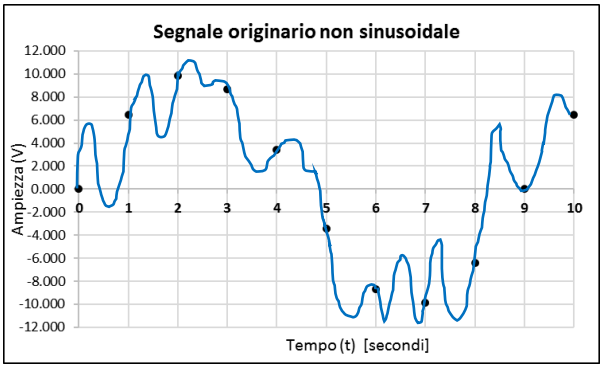
In altre parole: è possibile ricostruire esattamente il segnale analogico originale avendo solo i campioni??



SI! Se il segnale originario era questo:



NO! Se il segnale originario era questo:



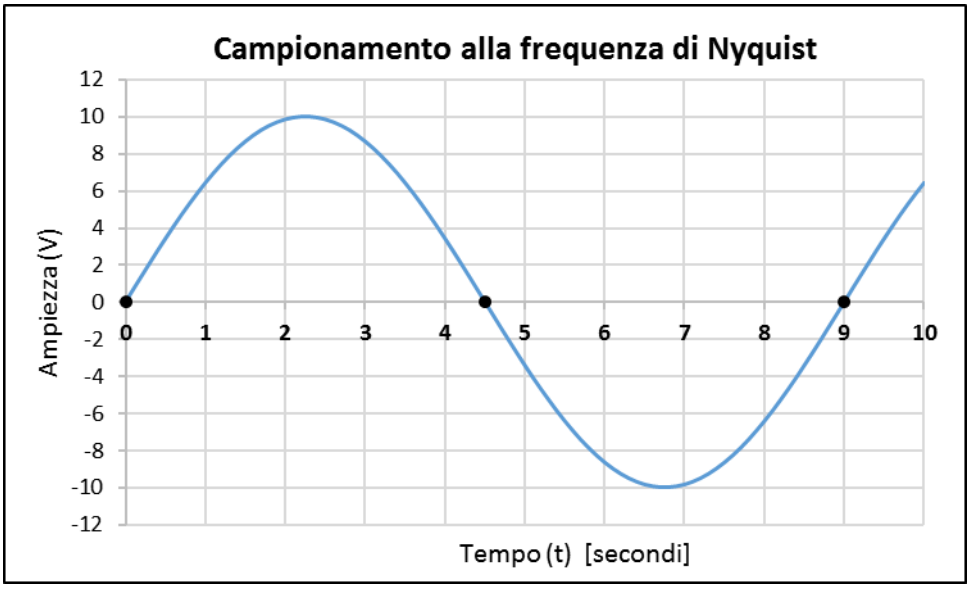
Se tra due campioni il segnale varia "troppo" è impossibile sapere com'era. Il problema riguarda la frequenza del segnale...esiste un teorema che ci permette di trovare la soluzione.

Teorema del campionamento (o di Nyquist-Shannon)

La frequenza di campionamento necessaria per evitare la perdita di dati deve essere superiore al doppio della frequenza massima del segnale analogico originario (chiamata frequenza di Nyquist).

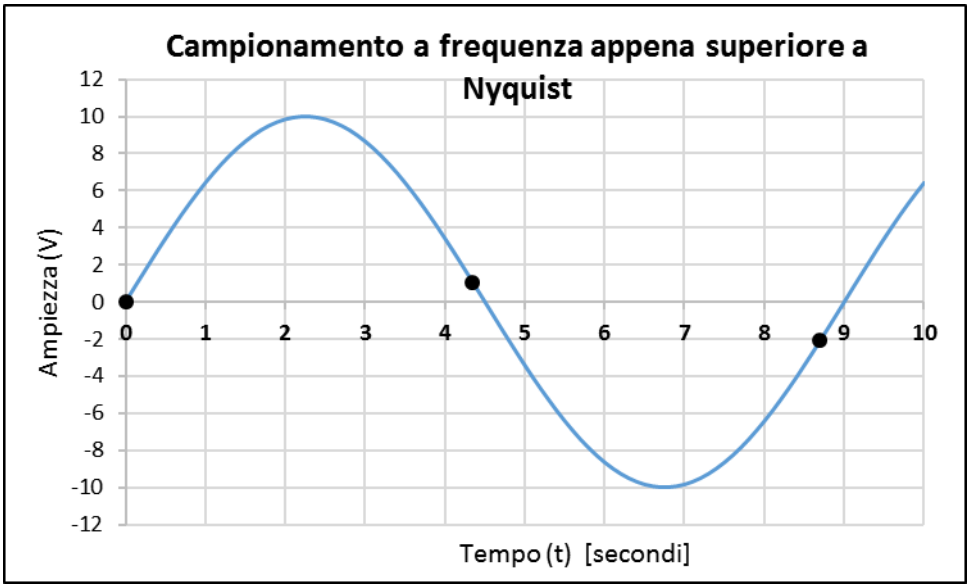
$f_c > 2f_{max}$ e quindi $f_{max} < \frac{f_c}{2}$

Quindi non si può campionare al di sotto della frequenza di Nyquist. Perché?

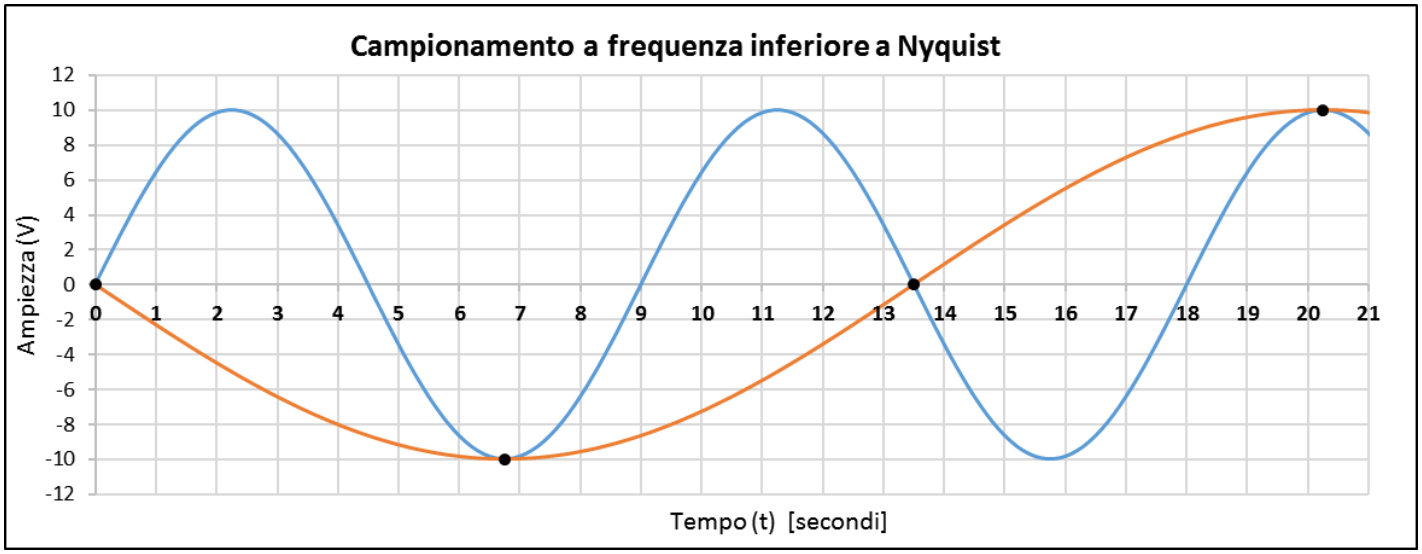


Campionare una sinusoide alla frequenza di Nyquist significa conoscere, per ogni periodo, solo i due punti in cui la sinusoide tocca l'asse zero (vedi figura sopra). Questo non permette di recuperare il segnale originario perché non sappiamo nulla della sua ampiezza. Ci sono infinite sinusoidi che possono passare per quei due punti.

Se si alza anche di poco la frequenza (vedi figura sotto) vediamo che il secondo campione avrà un valore positivo, il terzo uno negativo, questo ci permette di ricostruire perfettamente la sinusoide. In altre parole, **esiste una sola sinusoide che può passare per quei punti**, quindi è possibile trovarla senza errori. Potete provare come sfida a trovarne un'altra ma vedrete che non sarà possibile trovare un'altra sinusoide a frequenza inferiore a f_{max} che passi per quei punti.

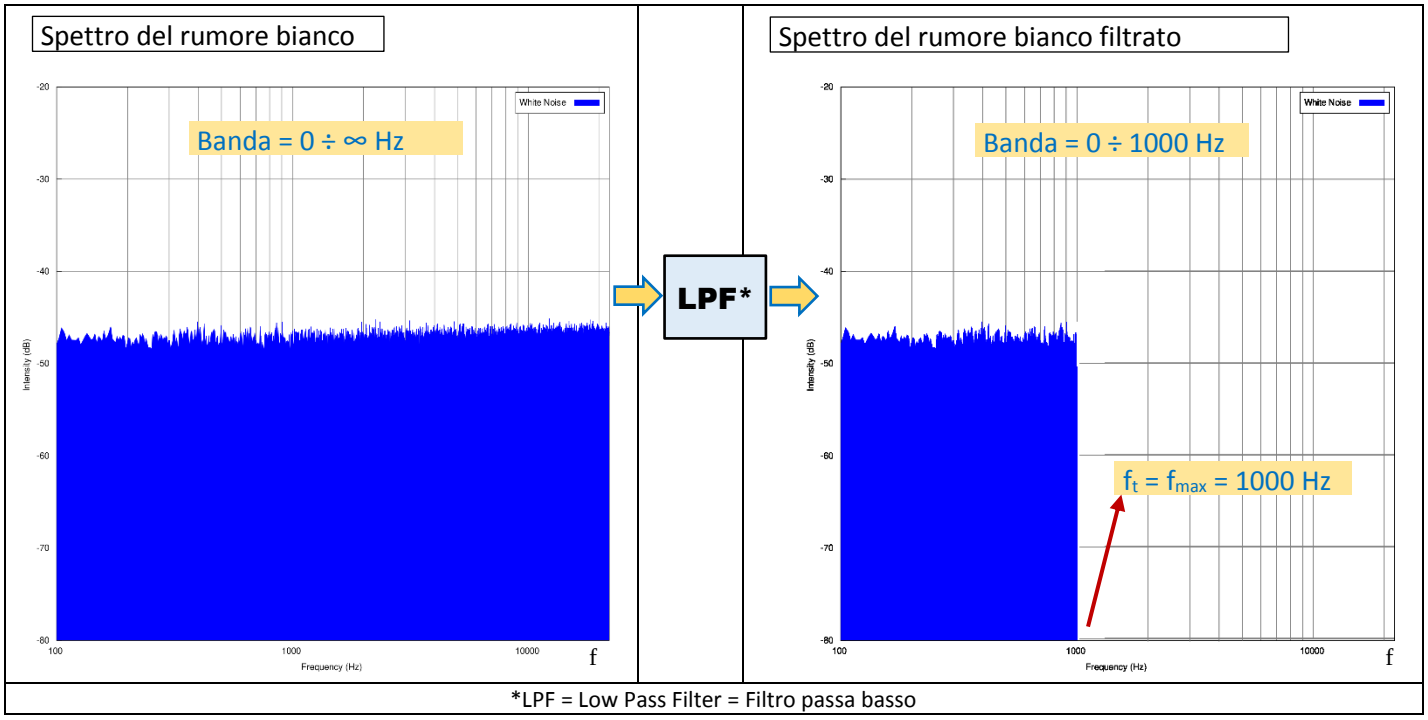


Campionando a frequenza inferiore a quella di Nyquist non sarà possibile trovare ricostruire esattamente il segnale originario. Nella figura sottostante è stata presa una frequenza di campionamento $f_c = 1{,}33\, f_{max}$ quindi inferiore alla frequenza di Nyquist. In questa condizione non siamo più in grado di individuare una sola sinusoide che passa per i punti campionati, bensì ne possiamo individuare 2. Quella originaria è la blu, mentre la seconda è quella arancione. Questo fenomeno prende il nome di "**aliasing**" e si traduce in uno sporcamento del segnale alle alte frequenze che diventa sempre più grande più la frequenza di campionamento si abbassa.



Questa è anche la ragione per cui prima di ogni campionatore è necessario un **filtro passa-basso**. Ossia un dispositivo in grado di eliminare dal segnale analogico tutte le frequenze superiori alla frequenza di Nyquist che andrebbero a sporcare il segnale. Bisogna considerare che ogni segnale è affetto da rumori a tutte le frequenze (vedi "Rumore bianco"). L'effetto di un filtro passa basso è quello di eliminare le frequenze superiori ad una particolare frequenza, chiamata **"frequenza di taglio"** f_t**.** La frequenza di taglio è anche pari alla frequenza massima, perché oltre quella non c'è più nulla $f_t = f_{max}$.

Andando a riprendere lo spettro del rumore bianco, vediamo nella figura seguente l'effetto di un filtro.



Nella pratica il filtro passa basso non è mai perfetto, nei pressi della frequenza di taglio c'è ancora un po' di rumore, perciò si consiglia di utilizzare una frequenza di campionamento molto superiore a quella di Nyquist (***oversampling***): $f_c \gg 2f_{max}$. Ad esempio $3f_{max}$.

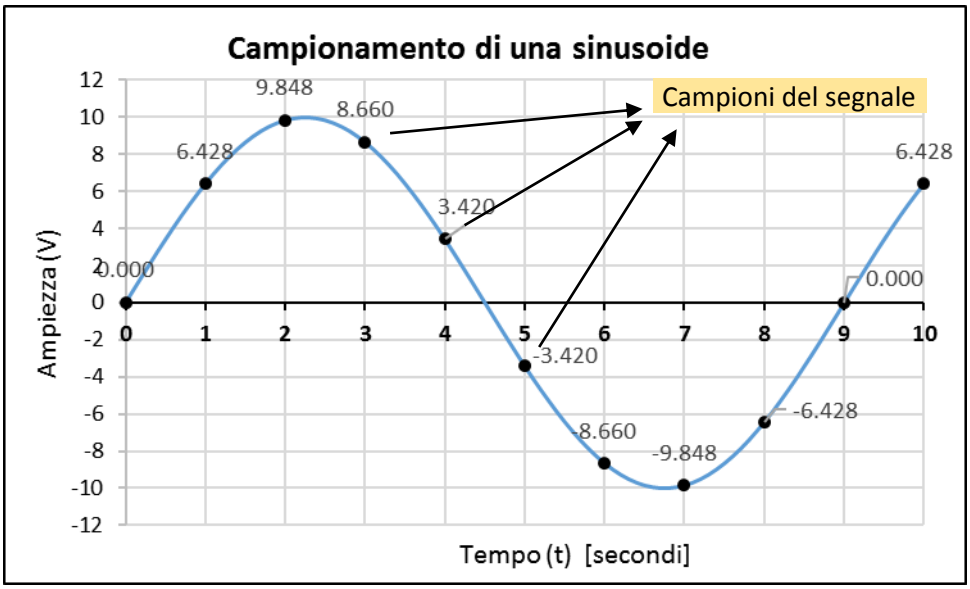
Ad esempio, Arduino (senza schede esterne) ha una frequenza di campionamento pari a 9 kHz. Quindi teoricamente è possibile campionare solo segnali analogici con frequenza inferiore o uguale a 4,5 kHz. Per avere ottimi risultati però si utilizza l'oversampling e quindi la frequenza massima scende a circa 3 kHz. Non si può usare quindi per applicazioni audio ad alta fedeltà visto che la frequenza dei suoni udibili va da 20 Hz a 20 kHz.

Domanda del secchione: perché stiamo considerando solo sinusoidi? Se il segnale di origine non fosse una sinusoide ma un qualsiasi segnale aperiodico?

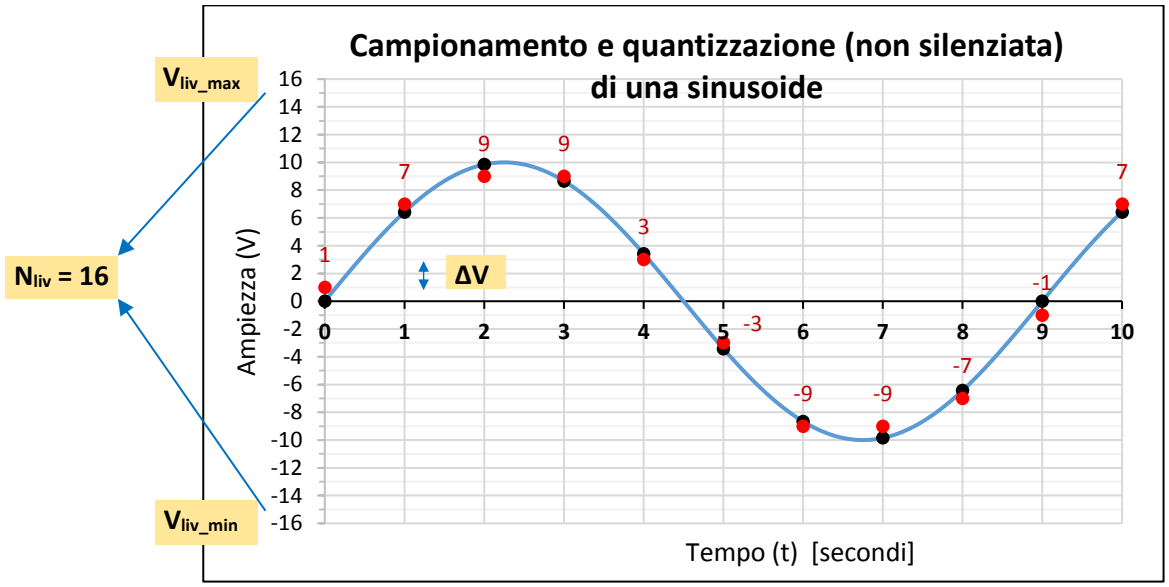
Risposta: In realtà anche il segnale aperiodico è formato da una somma di sinusoidi (vedi la lezione sui segnali). La differenza è solamente che nei segnali aperiodici le sinusoidi sono ad infinite frequenze anziché a frequenze multiple della fondamentale. A noi interessa vedere cosa succede alla massima frequenza del segnale. Quindi, se di queste infinite sinusoidi, prendiamo quella alla massima frequenza, anche per i segnali aperiodici avremo a che fare con una sinusoide.

2. **QUANTIZZAZIONE** (in inglese *"quantization"*)

I campioni del segnale possono avere infiniti valori (numeri con infinite cifre dopo la virgola). Cioè dal campionamento si ottiene un segnale discreto sull'asse dei tempi ma non sull'asse delle ampiezze.



La **quantizzazione** è il processo in cui i campioni vengono **discretizzati**, ossia approssimati e ricondotti ad un **numero finito di livelli**.



Nell'esempio in figura si è deciso di quantizzare utilizzando intervalli di 2V. Ad esempio il primo intervallo è 0 ÷ 2 V e gli viene associato il valore medio di 1 V; il secondo intervallo è 2 ÷ 4 V e gli viene associato il valore medio di 3 V e così via. Quindi i campioni vengono approssimati alle cifre intere dispari più vicine.

Questo processo introduce sempre un errore, chiamato **errore di quantizzazione**, perché non sarà possibile in nessun modo tornare esattamente al segnale originario dopo averlo quantizzato. Questo errore è chiaramente tanto più piccolo quanto più è grande il numero di livelli di quantizzazione. L'errore massimo si commette quando il campione è esattamente al centro tra due livelli, quindi è pari alla metà di un intervallo tra due livelli (ΔV). Si può anche calcolare come l'intervallo totale tra il livello più alto (V_{liv_max}) e quello più basso (V_{liv_max}) diviso il numero di livelli, diviso due.

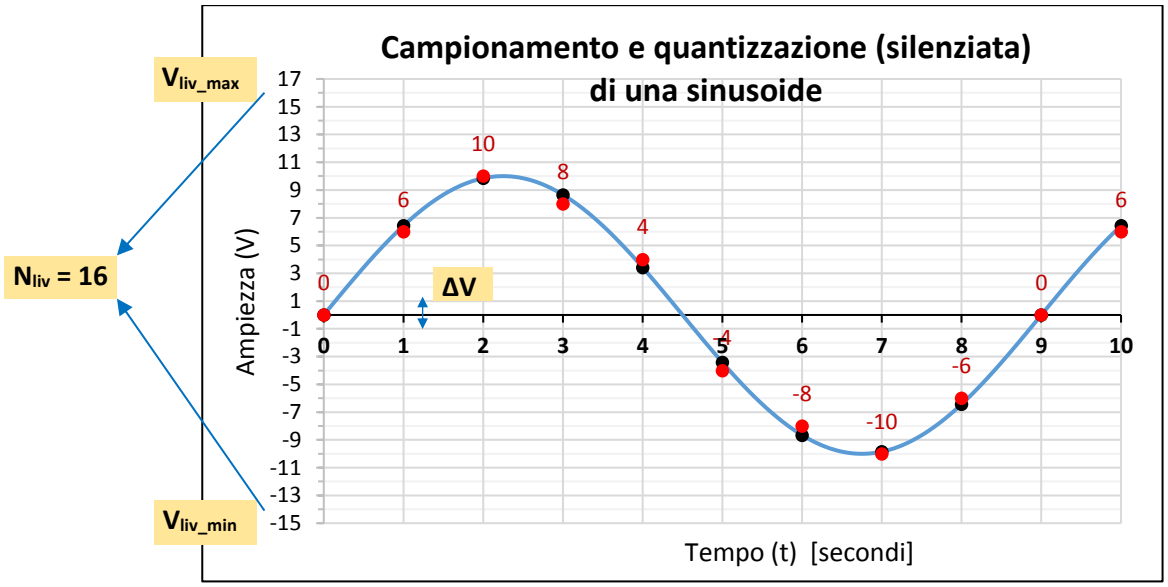
$$e_q = \pm\frac{\Delta V}{2} = \pm\frac{1}{2}\cdot\frac{V_{liv\_max} - V_{liv\_min}}{N_{liv}}$$

Per l'esempio in figura il calcolo è il seguente:

prima formula: $e_q = \pm\frac{2}{2} = \pm 1V$ oppure seconda formula: $e_q = \pm\frac{1}{2} \cdot \frac{16-(-16)}{16} = \pm\frac{1}{2} \cdot \frac{32}{16} = \pm 1V$

La tipologia di quantizzazione dell'esempio si dice "**non silenziata**" perché non esiste il livello con valore zero. Il vantaggio è che i livelli sono simmetrici rispetto allo zero (nell'esempio in figura sopra ci sono 8 livelli positivi e 8 negativi). Lo svantaggio è che quando il segnale è zero gli viene comunque attribuito un valore diverso da zero (vedi t=0s o t=9s in figura).

Se tra i livelli esiste il valore 0 allora si parla di quantizzazione "**silenziata**" (vedi figura sotto). Il vantaggio è che quando il segnale è zero o nei pressi dello zero (ad esempio rumore di sottofondo) gli viene attribuito il valore zero (vedi t=0s o t=9s in figura). Lo svantaggio è che i livelli non sono simmetrici rispetto allo zero (nell'esempio in figura sopra ci sono 8 livelli positivi, uno centrato sullo zero e 7 livelli negativi).



La quantizzazione degli esempi visti finora è detta **uniforme** perché la distanza tra i livelli è sempre la stessa. Esistono anche quantizzazioni **non uniformi** in cui i livelli non sono equispaziati. Tipicamente i livelli sono vicini tra loro per basse ampiezze e lontani per alte ampiezze. Ciò può essere utile per ridurre l'errore di quantizzazione relativo.

3. CODIFICA (in inglese *"encoding"*)

La codifica nell'associare ad ogni intervallo di quantizzazione una cifra binaria. Questo permette di ottenere un segnale digitale a tutti gli effetti e quindi permette di salvarlo o elaborarlo su un computer o trasmetterlo sulle reti digitali. Esiste una moltitudine di codifiche, ciascuna con particolari caratteristiche. Nei PC è la codifica che dà il nome al tipo di file che viene creato (esempio JPG, PNG, WAV, MP3, MP4, AVI, ecc).

Per segnali **unipolari** (solo positivi, che non scendono mai sotto l'asse zero) le codifiche più facili sono le seguenti:

a) Codice binario puro (PCM - Pulse Code Modulation)

La codifica binaria pura, chiamata anche "modulazione a impulsi codificati", associa ad ogni livello un numero binario partendo da 0, fino al valore massimo (2^N-1), dove N è il numero di bit della codifica. Per N=4 viene l'esempio in tabella.

Numero del livello	Codifica binaria pura (PCM)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Nel mondo audio i file con codifica PCM sono caratterizzati dall'estensione .WAV e si possono trovare ad esempio nei CD musicali (con qualche piccola modifica). Nel mondo delle immagini si possono trovare i file .BMP (Bitmap) che erano i primi file di immagine utilizzati e i .RAW che sono prodotti solitamente dalle fotocamere reflex.

b) Codice Gray

E' un codice in cui tra un livello e quello successivo cambia solo 1 bit. E' stato inventato per **minimizzare gli errori nel caso in cui venga mal interpretato 1 bit**. Infatti se nella lettura si sbaglia 1 bit l'errore è al massimo di un livello.

Numero del livello	Codifica Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001

15	1000

Curiosità: come si genera il codice Gray? Si devono svolgere i seguenti passi:

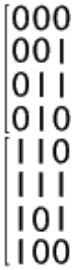
1. Si scrive 0 e 1 in colonna:

2. Si tira una riga sotto e poi si specchia rispetto a quella riga:

3. Si aggiunge uno 0 di fronte ai numeri sopra la riga a un 1 di fronte a quelli sotto al riga:

4. Si ripete il punto 2:

5. Si ripete il punto 3:



6. E così via ripetendo i punti 2 e 3.

In caso di segnali **bipolari** (segnali anche negativi che non scendono sotto l'asse zero) si pone il problema di come rappresentare il numero negativo. Le possibilità sono diverse

a) Codice binario con offset

E' identico al binario puro, in cui si associa la cifra 0000 al valore negativo più piccolo (anziché al valore 0 come fatto per il binario puro).

Numero del livello	Codifica binaria con offset
-8	0000
-7	0001
-6	0010
-5	0011
-4	0100
-3	0101
-2	0110
-1	0111
0	1000
+1	1001
+2	1010
+3	1011
+4	1100
+5	1101
+6	1110
+7	1111

b) Codice binario con segno

E' simile al binario pure ma il primo bit indica il segno: 0 se il numero è positivo, 1 se il numero è negativo. Lo zero può essere fatto in due modi (+0 oppure -0). Quindi in totale possono essere descritti (2^N-2) livelli, dove N è il numero di bit della codifica. Si perde un livello rispetto al binario puro. Per N=4 viene l'esempio in tabella.

Numero del livello	Codifica binaria con segno
-7	1111
-6	1110
-5	1101
-4	1100
-3	1011
-2	1010
-1	1001
0 (prima possibilità)	1000
0 (seconda possibilità)	0000
+1	0001
+2	0010
+3	0011
+4	0100
+5	0101
+6	0110
+7	0111

c) Codice binario complemento a 2

I numeri positivi sono identici al codice binario con segno. Quelli negativi si trovano facendo il complemento a 2 dei rispettivi valori positivi, ossia si prende il valore positivo, si invertono gli 1 con gli 0 e viceversa e infine si somma 1. Ad esempio per il valore -7 si prende +7 che vale 0111. Si girano invertono le cifre creando 1000. Si somma 1, creando 1001.

Numero del livello	Codifica binaria con segno
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
+1	0001
+2	0010
+3	0011
+4	0100
+5	0101
+6	0110
+7	0111

Il vantaggio di questo codice è di facilitare le operazioni di calcolo: le sottrazioni vengono ricondotte ad addizioni. Ad esempio per fare 6-7=-1 basta fare la somma tra 6 e (-7):
0110+
1001=
1111
Si può anche notare che il codice complemento a 2 è identico a quello con offset scambiando il primo 0 con un 1 e viceversa.

Esistono altri tipi di **codifiche molto più avanzate**, in continua evoluzione, utili a vari scopi. Ad esempio:

a) Codici correttori di errori
Esistono codici in grado di individuare e/o correggere dei piccoli errori che si possono verificare nella trasmissione dei segnali digitali. Questi sono comunemente usati in tutte le trasmissioni via internet.
Il più semplice si chiama CRC (Cyclic Redundancy Check), in italiano "controllo ciclico di ridondanza". Si basa sul calcolo del cosiddetto *checksum* (somma di controllo). In pratica, oltre ai dati viene inviata periodicamente una cifra binaria che rappresenta il risultato della somma di alcuni dei bit del segnale. Chi riceve il segnale calcola il *checksum* e lo confronta con quello inviato. Se sono diversi sicuramente c'è un errore. Il CRC non permette di correggere l'errore, quindi in caso di errore il ricevente deve chiedere nuovamente il dato errato al trasmittente.

b) **Codici crittografati**
Sono codici che sono progettati per essere decifrati solamente in possesso della cosiddetta "chiave di decodifica". Ne esistono di moltissimi tipi a vari livelli di sicurezza. Attualmente tutti i maggiori programmi di messaggistica utilizzano trasmissioni crittografate. Anche i siti web che iniziano per "https" (anzichè "http") forniscono connessioni crittografate ('s' sta per "secure"). Per maggiori informazioni cercare su internet "crittografia simmetrica", "crittografia asimmetrica" e "funzione di ash".

c) **Codici di compressione**
Sono codici utili a ridurre le dimensioni dei dati rispetto ad esempio alla classica codifica binaria pura. Queste codifiche danno il nome alle estensioni dei file che utilizziamo nei nostri PC.
Esistono due tipi di compressione:

1. **Lossless** (senza perdita di dati)
   Si basano su algoritmi che trovano nei dati delle ripetizioni che quindi possono essere "compattate". Non c'è perdita di dati quindi si può tornare esattamente al file originale. Ad esempio se in un segnale è presente una catena di 300 zeri, è possibile raggrupparli, indicando il numero di zeri anziché scriverli tutti. Fanno parte di questa categoria:

Per file generici: .ZIP .RAR .GZ .7z ecc
Per file audio: .FLAC
Per file video: praticamente non usati
Per file immagine: .TIFF

2. **Lossy** (con perdita di dati)
   Si basano su algoritmi che, oltre a fare la stessa cosa delle codifiche lossless, in più eliminano le informazioni meno importanti del segnale. Ad esempio nei file audio vengono eliminati i suoni molto deboli che sono coperti da altri suoni forti. Nelle immagini viene ridotto il numero di colori. Nei video si conservano solo i dati delle parti di schermo dove ci sono movimenti evidenti, lasciando fermi gli sfondi. E così via.
   Fanno parte di questa categoria:
   Per file generici: Non esistono! Nessuno vuole perdere dei dati di un documento o di un lavoro!
   Per file audio: .MP3 .AAC .WMA .AC3 ecc..
   Per file video: .MP4 .WMV .DivX .Xvid .Xvid ecc..
   Per file immagine: .JPG .PNG .GIF ecc...

   Qui sotto è riportato un esempio di compressone lossy. Più c'è compressione più si perde di qualità.