Digitalizzazione e campionamento: una breve introduzione

francesco.fuso@unipi.it; http://www.df.unipi.it/∼fuso/dida

(Dated: version 6 - FF, 17 novembre 2016)

È linguaggio comune distinguere le grandezze misurate e i metodi di misura tra *analogici* e *digitali*. Ormai da alcuni decenni il mondo viaggia veloce verso la diffusione capillare di metodi digitali pressoché per tutte le necessità pratiche, in particolare quelle legate allo scambio e al trattamento delle informazioni. Nel nostro piccolissimo, anche noi possiamo toccare con mano vantaggi (molti) e svantaggi (pochi, legati soprattutto alla scelta dell'hardware) dell'approccio digitale, grazie in particolare all'uso di Arduino. Questa nota intende fornire un minimo di background concettuale utile per affrontare questo approccio. Inoltre essa riporta alcune considerazioni sull'impiego di Arduino per la misura di segnali periodici, in particolare di tipo sinusoidale, come richiesto in una esperienza pratica.

I. ANALOGICO VS DIGITALE

Come ben sapete, le grandezze di interesse per l'elettricità e l'elettronica hanno generalmente valori che "appaiono" continui, cioè non discreti, né quantizzati [1]. La misura di grandezze di questo tipo si dice talvolta *analogica*. Un buon esempio di misura analogica è quella che si fa con uno strumento a lancetta, dove idealmente la lancetta può muoversi assumendo con continuità qualsiasi posizione angolare.

Ci sono poi delle grandezze la cui misura implica un conteggio, cioè l'ottenimento di un valore rappresentato da un *numero intero* (detto anche, con le opportune precisazioni che faremo in seguito, *parola digitale*), per esempio quando si contano gli atomi contenuti in un campione, i fotoni emessi per unità di tempo da una sorgente, gli elettroni presenti in un pezzo di materiale, e così via. La misura di queste grandezze si dice talvolta *digitale*. Un buon esempio di misura digitale è il conteggio dei cicli di oscillazione, o periodi, di un oscillatore (non necessariamente elettronico: potete considerare per esempio il conteggio del passaggio per una determinata posizione di una pendola).

È ovvio che dal punto di vista concettuale esistono notevoli differenze tra le due tipologie di misura: l'analogica consente di ottenere un valore idealmente accurato, cioè con tante cifre significative quante sono consentite dalla sensibilità, o accuratezza, dello strumento: cambiando strumento, cioè migliorandone la sua sensibilità, è possibile abbattere l'incertezza fino a valori virtualmente trascurabili. La misura digitale fornisce un valore intero, normalmente con un'incertezza pari all'unità: idealmente questa incertezza non dipende dallo strumento di misura (purché idoneo per la misura che si intende eseguire).

Dal punto di vista pratico, anche una qualsiasi misura analogica può dare luogo a un risultato intero (opportunamente dimensionato), per esempio quando essa, codificata in modo opportuno (per esempio scegliendo l'unità di misura in modo che non ci siano cifre significative decimali) viene trascritta su un foglio, o inserita in un computer per il successivo trattamento. La misura digitale, però, ha tali caratteristiche "in natura", e non richiede ulteriori passaggi per essere ricondotta a un numero intero.

A. Digitalizzazione e campionamento

I meccanismi di digitalizzazione, di cui daremo un breve esempio concettuale in seguito, sono inerentemente accompagnati da processi di *campionamento*. Campionare significa prendere un pezzettino, in senso generalmente temporale, della grandezza da misurare, che quindi viene analizzata solo per questo pezzettino. In pratica, la digitalizzazione avviene in un intervallo temporale finito e non nullo, cioè essa richiede del tempo per essere portata a termine, per cui è inevitabile che la conversione da analogico a digitale avvenga per pezzettini del segnale sotto analisi, di durata non nulla, benché generalmente piccola. Dunque digitalizzazione e campionamento sono concetti intimamente legati l'un l'altro.

Rifrasando ancora: nella descrizione convenzionale di un qualsiasi strumento di misura si fa riferimento alla "prontezza" come alla capacità di seguire e registrare le variazioni temporali della grandezza misurata. Qualsiasi strumento ha una prontezza finita: in un digitalizzatore l'analogo del limite di prontezza è dettato proprio dal tempo (minimo) di campionamento.

Il legame tra digitalizzazione e campionamento è evidente anche in settori apparentemente diversi dalla misura di grandezze elettriche. Facciamo un solo esempio, valido anche per intuire i progressi permessi dai metodi digitali: la riproduzione di musica è notoriamente passata da tecniche analogiche (fonografo e poi giradischi, dove, per intenderci, il movimento della puntina sul solco del disco, opportunamente amplificato con metodi meccanici e poi elettronici, provocava uno spostamento analogico della membrana dell'altoparlante, e viceversa per la registrazione) a metodi digitali a tutti voi ben noti. Sono altrettanto noti i vantaggi che se ne sono ricavati, essenzialmente costituiti da un miglioramento della qualità audio (estensione dinamica e in frequenza nettamente più ampie) e dalla drastica riduzione dei disturbi, ovvero rumori, dovuti alla soppressione dei segnali "spurii" generati in fase di registrazione e riproduzione. I più appassionati

di musica tra voi sapranno che la capacità dinamica, intesa come *profondità di digitalizzazione* (espressa in bits, come vedremo in seguito) e la rapidità (ovvero *rate o frequenza di campionamento*) sono figure di merito rilevanti nella riproduzione musicale. Esse sono anche caratteristiche distintive nella misura digitale di segnali elettrici qualsiasi.

II. FUNZIONAMENTO DI BASE DI UN DIGITALIZZATORE MODELLO

Grazie alla vasta diffusione della digitalizzazione, motivata soprattutto dalla diffusione della consumer electronics, sono state messe a punto diverse tecnologie specifiche, molto efficaci e spesso complicate da descrivere. Tuttavia è possibile identificare un meccanismo base in grado di spiegare come sia possibile digitalizzare un segnale analogico. In sostanza, quello che qui descriviamo è il principio di funzionamento di un modello di *convertitore analogico-digitale* (ADC - Analog to Digital Converter, o convertitore A/D), che può essere considerato il cuore di ogni dispositivo che permetta di convertire in numero intero una grandezza elettrica, per esempio una d.d.p., analogica in natura.

Gli ingredienti fondamentali del metodo sono:

1. un *clock*, ovvero un dispositivo in grado di generare impulsi, di durata virtualmente trascurabile, equispaziati nel tempo, così come fa un orologio;

2. un *contatore*, cioè un dispositivo in grado di contare gli impulsi di cui sopra, dotato di un circuito di reset (di azzeramento) opportunamente comandabile;

3. un'*interfaccia digitale*, cioè, per intenderci, un processore in grado di trattare (registrare, visualizzare, etc.) il conteggio prodotto dal contatore creando la parola digitale, anch'esso dotato della possibilità di registrare, visualizzare, etc. in istanti comandabili dall'esterno;

4. un *generatore di rampa*, cioè un dispositivo, opportunamente triggerabile, cioè con partenza comandabile dall'esterno, in grado di fornire una d.d.p. crescente linearmente con il tempo [2];

5. un *comparatore*, cioè un dispositivo in grado di comparare i livelli (le d.d.p., riferite ovviamente alla stessa linea di massa) di due ingressi e fornire in uscita un segnale dipendente dall'esito della comparazione (per intenderci, zero o uno a seconda che prevalga uno o l'altro dei due segnali in ingresso).

La Fig. 1 illustra lo schema a blocchi, semplificato, del sistema [pannello (a)] e mostra uno schemino della temporizzazione (timing) del circuito, ovvero della sequenza temporale delle operazioni da esso compiute.

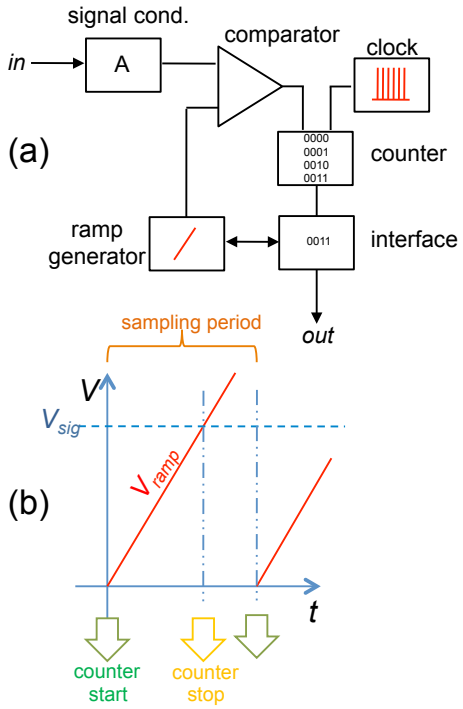

Figura 1. Schema a blocchi semplificato (a) e schema del timing di funzionamento (b) di un digitalizzatore modello; V_{sig} e V_{ramp} indicano rispettivamente la d.d.p. del segnale (incognita da misurare) e la d.d.p. prodotta dal generatore di rampa, che aumenta linearmente nel tempo. Lo schema a blocchi non riporta esplicitamente, per semplicità tipografica, lo stadio di sample-and-hold brevemente citato nel testo.

Per capire il funzionamento di questo digitalizzatore modello cominciamo subito con il notare che il conteggio degli impulsi di clock è un'operazione inerentemente digitale, dato che il suo esito è sicuramente un numero intero. Infatti, almeno in linea di principio, ogni operazione di misura di intervalli temporali può essere ricondotta al conteggio dei cicli di oscillazione di un orologio. Dunque, se si riesce a "collegare" la misura del tempo con quella della d.d.p. in ingresso, si è sulla buona strada per costruire un digitalizzatore.

Agli ingressi del comparatore sono inviati il segnale (d.d.p.) da misurare e l'uscita (d.d.p.) del generatore di rampa. Supponiamo poi che la partenza della rampa, cioè il trigger del generatore che produce la rampa, scatti contemporaneamente al reset del contatore del tempo: dunque il contatore segna zero all'inizio del processo, quando il valore della rampa è pure zero (o altro livello noto). Infine, supponiamo che l'uscita del comparatore vada a comandare l'interfaccia digitale, cioè sia in grado di "bloccare" il conteggio. Il contatore conta (1, 2, 3,..) fin quando il segnale incognito in ingresso è inferiore alla rampa. Quando invece la rampa supera il segnale incognito, il processo si interrompe e l'interfaccia digitale acquisisce ("blocca") il conteggio. Grazie alla linearità della rampa con il tempo, questo conteggio è proporzionale al

valore del segnale in ingresso, che quindi risulta digitalizzato, cioè convertito in un numero intero, realizzando di fatto il "collegamento" tra grandezze che abbiamo prima ipotizzato. Un'opportuna calibrazione consente di ricondurre il conteggio al valore di d.d.p. (in unità fisiche) del segnale incognito entro una determinata incertezza di calibrazione. Inoltre è ovvio che, nelle implementazioni pratiche, il segnale incognito, prima di arrivare all'ingresso del comparatore, viene "condizionato", cioè opportunamente amplificato o attenuato, eventualmente cambiato di segno e traslato rispetto allo zero. In assenza di questo condizionamento non sarebbe possibile, per esempio, misurare con il nostro digitalizzatore modello dei segnali negativi. Anche il circuito di condizionamento soffre di incertezze di calibrazione, delle quali tenere conto nella determinazione dell'errore complessivo di misura.

Poiché la conversione da analogico a digitale è, di fatto, una misura di tempo, è ovvio che essa richiede del tempo per essere completata. Dunque un digitalizzatore non può operare continuamente, per cui viene confermata l'affermazione precedente sulla necessità che il segnale digitalizzato sia sempre e inevitabilmente campionato in un (generalmente piccolo) pezzettino di tempo.

In questo intervallo di tempo sarebbe opportuno che la d.d.p. incognita all'ingresso del comparatore risultasse costante e, dato che questo non si verifica necessariamente (non si verifica affatto nella maggior parte delle situazioni sperimentali di interesse), è necessario applicare un'ulteriore forma di condizionamento. Questo condizionamento viene normalmente attuato da un circuito, detto *sample-and-hold*, che mantiene costante, al valore che ha all'inizio della digitalizzazione, il segnale in ingresso al comparatore per tutta la durata della digitalizzazione stessa. Un semplice modello di sample-and-hold è un condensatore che viene caricato in maniera praticamente istantanea, mantenendosi a un potenziale costante per la durata della digitalizzazione. Al termine, cioè quando il contatore del tempo viene bloccato, il condensatore deve essere scaricato, in maniera altrettanto istantanea, per permettere una nuova digitalizzazione, e quindi occorre un opportuno circuito di controllo in grado di compiere tali operazioni in maniera sincrona con l'operazione del digitalizzatore [3].

La descrizione del meccanismo di digitalizzazione mette in evidenza immediatamente quali siano le due principali figure di merito di un digitalizzatore:

- il massimo rate di campionamento, che è evidentemente dipendente dalla rapidità di risposta del comparatore e dalla ripidità della rampa prodotta; esso è normalmente espresso in Sa/s (samples per secondo) e qualche volta in Hz (eventi di campionamento per secondo);

- la dinamica, o profondità di digitalizzazione, cioè il massimo numero di conteggi che può essere registrato (ovviamente nel tempo necessario alla misura), normalmente espresso in *bit*, dato che nel mondo digitale vige la codifica binaria.

Entrambe queste figure di merito hanno ripercussioni sulla qualità della misura. Il massimo rate di campionamento influenza direttamente la possibilità di seguire le variazioni di segnali rapidamente dipendenti dal tempo. La dinamica, o profondità di digitalizzazione, ha a che fare con la portata e la sensibilità della misura. Inoltre è evidente che, a prescindere da ogni altra eventuale causa di errore legata alla calibrazione, esiste un'*incertezza di digitalizzazione* che vale (almeno) un bit. Infatti la capacità dinamica finita implica che il comparatore scatti quando il segnale si trova all'interno di un intervallo la cui ampiezza è pari a quella di un singolo bit, per cui il valore del segnale può essere determinato solo all'interno di questo intervallo [4]. È prassi comune, e motivata da considerazioni tecniche e concettuali, largheggiare nella definizione di questa incertezza. Nella misura, essa diventa di fatto una barra di errore, e in genere si considera una (semi-)barra di errore di un bit. Nel caso in cui il singolo bit costituisca l'unità di digitalizzazione, come nel digitalizzatore modello esaminato sopra, oppure nell'uso pratico di Arduino, allora, e usando una nomenclatura che già conosciamo, si può anche dire che l'incertezza di lettura è *un digit*, con ovvio significato del termine (il digit in questione è la cifra meno significativa della lettura effettuata).

Facciamo qualche esempio di tipiche figure di merito, mantenendoci nell'ambito degli strumenti di misura. A partire dalla loro prima comparsa nel mercato, intorno a metà degli anni '80, gli oscilloscopi digitali hanno progressivamente soppiantato gli oscilloscopi analogici (ma noi continuiamo a usarli in laboratorio). Oggi un oscilloscopio digitale economico (più economico dei rari e preziosi strumenti analogici ancora a catalogo) può avere un rate di campionamento (*sampling rate*) di diverse centinaia di MSa/s. Oscilloscopi di classe più elevata campionano a un rate di parecchi GSa/s su tutti i canali (tipicamente quattro) di cui sono dotati. Un elevato sampling rate è necessario per seguire l'andamento temporale di segnali che variano molto rapidamente nel tempo. Per esempio, nel caso di segnali periodici una ricostruzione fedele della forma d'onda richiede che venga acquisito il segnale corrispondente a diversi istanti nell'ambito del periodo. Dunque un sampling rate di 1 GSa/s garantisce un'agevole ricostruzione di segnali periodici che hanno una frequenza dell'ordine di alcune centinaia di MHz (sempre che la *banda passante* dello stadio di ingresso dello strumento lo consenta).

Normalmente la profondità di digitalizzazione per un oscilloscopio è di 8 bit, che significa che sono accessibili $2^8 = 256$ livelli distinti, ovvero che il massimo conteggio di digitalizzazione può andare da zero a 255 unità. Esistono anche strumenti in cui tale profondità arriva a 12 bit (ovvero $2^{12} = 4096$ livelli) e certamente alcuni dispositivi (per esempio, schede di acquisizione da inserire negli slot dei computer e strumenti simili) permettono di arrivare a 16 bit ($2^{16} = 65536$ livelli) o addirittura 24 bit (2^{24} oltre 16 milioni di livelli) , comprensibilmente a spese del rate di campionamento, limitato, in questo caso, alle

centinaia di kHz. La risoluzione, ovvero la sensibilità, di un digitalizzatore a 8 bit può non essere particolarmente esaltante. Supponendo di avere un segnale la cui ampiezza massima (se volete, picco-picco) è 10 V, allora la misura fatta a 8 bit avrà un'incertezza di digitalizzazione di 10 V/256 $\simeq$ 40 mV, che sarà dunque la minima variazione che potrà essere letta nella misura. Se preferite, tutto questo vuol dire che l'incertezza di digitalizzazione ha un valore relativo dello 0.4% del fondo scala, che non è un dato eccellente per le misure che si possono fare in un laboratorio, specie considerando che ad essa si devono generalmente aggiungere gli errori sistematici di calibrazione.

III. CARATTERISTICHE PRINCIPALI DI ARDUINO

Come ben sapete, in laboratorio facciamo uso di Arduino per diversi scopi sperimentali, il principale dei quali è quello di digitalizzare delle d.d.p. generalmente dipendenti dal tempo usando le porte analogiche disponibili (6 ingressi distinti, marcati `A0 - A5`). La dinamica di digitalizzazione, che vale 10 bit (cioè 1024 livelli distinti, da 0 a 1023), è un dato ben noto. Il massimo rate di campionamento, invece, non è riportato in maniera chiara nei datasheets, dove è indicato un ampio intervallo, evidentemente dipendente dalle condizioni di operazione, che mostra come il tempo minimo necessario per concludere una singola digitalizzazione sia di almeno 13 μs. Nelle nostre analisi sperimentali, in particolare attraverso la ricostruzione della distribuzione temporale degli intervalli di campionamento Δt, abbiamo individuato un intervallo tipico Δt_{dig}, compreso tra 10 e 20 μs, che possiamo identificare con tale tempo minimo.

Il massimo rate di campionamento dipende, ovviamente, da questo tempo, essendo in sostanza pari al suo reciproco. Tuttavia, poiché la digitalizzazione richiede l'esecuzione di altre operazioni (la scrittura nella memoria interna, in particolare), il massimo rate di campionamento, anche nelle condizioni di "overclock" tipiche delle nostre esperienze, presenta ulteriori limitazioni. Di fatto, Arduino diventa instabile quando si cerca di campionare a un rate superiore a oltre 20 kSa/s, che corrisponde a un intervallo di campionamento (nominale) $\Delta t \sim 50$ μs. La scelta conservativa da noi effettuata in molte situazioni (non tutte) è quella di non spingersi al di sotto di $\Delta t = 100$ μs, che corrisponde a un rate di campionamento di 10 kSa/s. Di conseguenza, usando Arduino è difficile seguire segnali che variano in maniera significativa su scale temporali inferiori al centinaio di microsecondi.

Dato che qui siamo a riassumere le principali caratteristiche di Arduino, ricordiamone anche delle altre, cominciando con quelle di carattere elettrico. La resistenza di ingresso del digitalizzatore, come dichiarata nei datasheets, è molto alta (100 Mohm nominali). Nel caso di segnali rapidamente variabili nel tempo, è possibile che il circuito di sample-and-hold dia luogo a una "resistenza apparente" (è un'*impedenza*, come tratteremo diffusamente nel seguito del corso) minore e dipendente dalla rapidità di variazione del segnale stesso. È sempre buona norma verificare che la resistenza di ingresso sia trascurabile nelle specifiche condizioni di uso, verifica che può essere facilmente compiuta collegando l'ingresso di Arduino in parallelo a un altro strumento di misura (per esempio un oscilloscopio, che però ha una resistenza di ingresso di solo 1 Mohm) e osservando le variazioni al segnale misurato da questo strumento quando Arduino viene collegato o scollegato dal circuito. Se la resistenza si mantiene sufficientemente alta, la perturbazione dovuta alla presenza di Arduino è trascurabile, e il segnale misurato dallo strumento non cambia apprezzabilmente di ampiezza (entro la propria barra di errore).

Oltre agli ingressi analogici, Arduino ha la possibilità di controllare ben 14 porte digitali, configurabili come input o output. Queste porte possono assumere o leggere un valore binario (zero o uno) a seconda che siano a livello "basso" (ovvero, in rappresentazione binaria, 0, in unità fisiche circa 0 V) o "alto" (ovvero 1, tipicamente pari alla tensione di riferimento del digitalizzatore, cioè $V_{ref} \simeq 5$ V). Infatti il significato dell'aggettivo digitale ad esse attribuito significa proprio che hanno un comportamento binario (se usate come output, sono accese o spente, se utilizzate come input stabiliscono che in ingresso c'è un segnale acceso o spento). Configurate come uscite, queste porte sono utili per "controllare" l'esperimento, come per esempio nella carica/scarica del condensatore. La massima corrente che può essere ottenuta è, nominalmente, di 20 mA (50 mA in totale se se ne usa più di una). Naturalmente accensione e spegnimento delle porte non sono operazioni realmente istantanee, cioè la commutazione tra livello alto e basso non può che avvenire in un tempo finito: si può verificare facilmente, con una semplice misura all'oscilloscopio, che soprattutto la fase di spegnimento segue un andamento esponenziale, con un tempo caratteristico di alcune decine di microsecondi (paragonabile al tempo minimo di campionamento, che quindi stabilisce una sorta di "limite di velocità" per Arduino).

Sei tra queste porte digitali, quelle marcate con un tilde nelle serigrafie, possono operare come output in una modalità detta `PWM` (*Pulse-Width Modulation*). In questa modalità le porte generano treni di impulsi a frequenza fissa (e piuttosto bassa, cioè minore di 1 kHz) con duty-cycle regolabile via software (può essere aggiustato su 8 bit, cioè 256 distinti valori) [5]. Come vedremo in una futura esperienza, supponendo di collegare a queste uscite un circuito integratore con frequenza di taglio opportuna, la regolazione del duty-cycle del treno di impulsi permette di "simulare" un'uscita analogica (regolabile tra qualcosa di prossimo a 0 V e qualcosa di prossimo a V_{ref}), essendo la d.d.p. in uscita al circuito integratore proporzionale alla frazione di tempo in cui l'impulso rimane al suo livello alto.

Infine, avremo sicuramente modo di usare alcune di queste porte come ingressi digitali, per esempio secondo quanto descritto in Appendice. In termini generali, la

lettura digitale significa attribuire un livello basso o alto a seconda che la d.d.p. letta sia al di sotto o al di sopra di una certa soglia. Nel caso di Arduino, la distinzione segue la convenzione TTL, per cui la soglia è attorno a 3.5 V (con grande tolleranza). Concettualmente, questa operazione di lettura corrisponde a una digitalizzazione con una dinamica di un solo bit. Ridurre la dinamica implica abbattere i tempi necessari a completare la digitalizzazione, per cui normalmente tali porte operano a un rate corrispondente al massimo rate di campionamento.

Ci sono poi vari altri aspetti rilevanti per il funzionamento e l'uso di Arduino, di cui ci occuperemo eventualmente in sede di presentazione e discussione delle singole esperienze.

IV. SOTTO- (E SOVRA-)CAMPIONAMENTO

I processi di campionamento possono facilmente condurre a degli artefatti, cioè alla comparsa di andamenti apparenti che non sono presenti nel segnale originale. Il problema che qui vogliamo brevemente illustrare è molto ampio e complesso, e, in un certo senso, anche ubiquo, poiché si presenta non solo in ambito sperimentale ma anche quando si affrontano simulazioni numeriche basate su calcoli al computer. In qualche forma, di questo siete già al corrente, visto che proprio per evitare una sorta di sotto-campionamento create degli array equispaziati logaritmicamente come variabile indipendente per il calcolo delle curve di fit nei casi in cui la rappresentazione dei grafici è su scala logaritmica.

In questa sezione ci limitiamo ad affrontare il problema realizzando un semplice esperimento in cui acquisiamo con Arduino il segnale prodotto da un generatore di funzioni impostato per generare un'onda sinusoidale di frequenza, ampiezza e offset opportunamente regolati. In particolare:

- dato che Arduino può accettare solo d.d.p. positive (rispetto alla linea di massa, o terra, ovvero la boccola `GND`, che va debitamente collegata alla massa del generatore), aggiustiamo la manopola `OFFSET` del generatore di funzioni in modo che il segnale da esso prodotto sia *sempre positivo*: questa condizione va attentamente verificata all'oscilloscopio prima di collegare Arduino;

- dato che Arduino accetta solo d.d.p. al di sotto del massimo valore $V_{ref} \simeq 5$ V (in condizioni tipiche e senza optare, ovviamente, per l'uso del riferimento interno a 1.1 V nominali), verifichiamo anche che il segnale *non superi mai* i 5 V di ampiezza: eventualmente agiamo sulle regolazioni di `OFFSET` e `AMPL` affinché anche questa condizione sia soddisfatta alla visualizzazione con l'oscilloscopio.

Visto che, almeno per il momento, non siamo interessati a sincronizzare l'acquisizione dati con un qualche evento, possiamo ragionevolmente impiegare la strategia (sketch e script) che abbiamo già usato per la misura di d.d.p. continue. Nello script possiamo selezionare l'intervallo di campionamento in passi di 100 μs, tra 100 μs e 900 μs e qui scegliamo $\Delta t = 500$ μs (nominali). In queste condizioni, essendo il record composto da 256 dati, la durata complessiva dell'acquisizione è $\Delta t_{tot} \sim 0.13$ s (inclusa una grossolana stima del tempo minimo di digitalizzazione). Naturalmente non c'è alcun motivo per convertire i valori digitalizzati in unità fisiche, visto che siamo interessati a valutare gli andamenti temporali e non a conoscere l'effettivo valore della d.d.p. in un determinato istante. Quindi non applicheremo alcuna calibrazione, usando direttamente i dati in unità arbitrarie di digitalizzazione (digit).

La Fig. 2 mostra un esempio di dati acquisiti per quattro diverse impostazioni della frequenza f del generatore di funzioni: osservate che, per esigenze di chiarezza tipografica, i punti corrispondenti ai dati, rappresentati con le barre di errore (incertezze "convenzionali", cioè $\delta V = \pm 1$ digit, $\delta t = \pm 4$ μs), sono stati collegati con una linea continua blu, che quindi rappresenta una "guida per gli occhi" (e non un best-fit).

La figura mostra in maniera chiara, almeno dal punto di vista visivo, gli effetti dei problemi che abbiamo citato. I dati dovrebbero seguire l'andamento di una sinusoide, ma questo andamento è evidente solo nel terzo pannello dall'alto, quello acquisito a $f \simeq 0.03$ kHz, corrispondenti a un periodo $T \sim 30$ ms. In queste condizioni in un singolo periodo vengono campionate oltre 60 misure, per cui l'oscillazione può essere fedelmente riprodotta. Se, per esempio, la frequenza viene moltiplicata per un fattore 10, cioè nel caso $f \simeq 0.3$ kHz, il numero di campionamenti in un periodo scende ben sotto la decina: visivamente diventa difficile distinguere la forma sinusoidale da una forma, per esempio, triangolare. Pertanto questi dati potrebbero essere difficilmente impiegati se lo scopo della misura fosse quello di discriminare fra triangolare e sinusoidale, anche se alcune informazioni, come il periodo o l'ampiezza, potrebbero ancora essere dedotte con ragionevole accuratezza. In altre parole, il *sotto-campionamento* inizia a produrre i suoi effetti quando, nel caso sinusoidale che stiamo esaminando, ci sono meno di una decina di punti per ogni periodo [6].

Quando il sotto-campionamento è particolarmente "spinto", come nel pannello superiore, acquisito a $f \simeq 3$ kHz, possono comparire facilmente degli artefatti: l'andamento che si intuisce guardando la figura è tipico di un battimento, invece che di una semplice oscillazione. Il periodo del battimento è presumibilmente legato, in un modo non sempre facilmente individuabile, alla differenza tra periodo del segnale e tempo di campionamento, ovvero fra loro multipli e sottomultipli. Il risultato indica in maniera chiara che sotto-campionare, cioè scegliere intervalli di campionamento maggiori del periodo del segnale che si vuole ricostruire (in questo caso $\Delta t \sim 2T$), conduce a conclusioni erronee. Nell'ambito della teoria del campionamento, si parla spesso di *aliasing* quando gli artefatti assumono delle forme specifiche (in questo

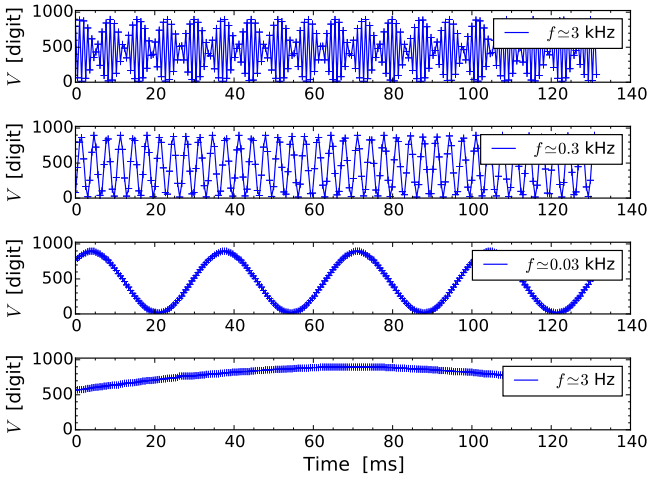

Figura 2. Acquisizione di un segnale sinusoidale, di opportuna ampiezza e offset, e diversa frequenza, come indicato in legenda, eseguita da Arduino. Il campionamento è effettuato su 256 punti intervallati da $\Delta t = 500$ μs nominali ed è totalmente asincrono. I punti corrispondenti ai dati sperimentali sono uniti da una linea continua di colore blu, che quindi costituisce una guida per l'occhio e non un best-fit.

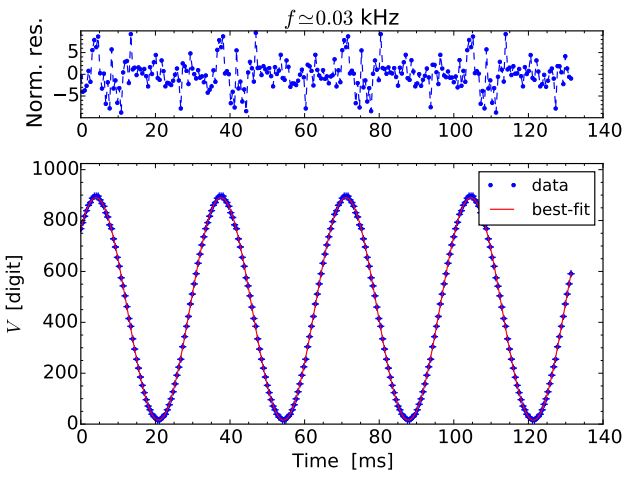

Figura 3. Stessi dati di Fig. 2, terzo pannello dall'alto, con sovrapposta la curva ottenuta con il best-fit discusso nel testo. Il pannello superiore mostra il grafico dei residui normalizzati.

caso, tipo battimento), che non corrispondono al segnale "genuino".

Dall'altra parte, esiste anche il rischio di cadere nel problema "opposto", che qui chiamiamo sovracampionamento: il pannello inferiore, acquisito a $f \simeq 3$ Hz, cioè per $T > \Delta t_{tot}$ (il periodo è maggiore della durata totale dell'acquisizione), mostra solo una porzione di oscillazione sinusoidale, senza dare alcuna ovvia evidenza del carattere periodico del segnale.

Infine, come ultimo commento alla figura, notiamo che l'"inizio" della registrazione avviene a istanti diversi nei quattro pannelli, cioè che le quattro forme d'onda registrate hanno un fattore di fase costante diverso fra loro. Questa è un'ovvia conseguenza del carattere *asincrono* dell'acquisizione, che parte in un istante (determinato dallo script di Python) che non corrisponde ad alcun evento fisico di rilievo.

Per i soli dati acquisiti in cui si distingue una bella sinusoide, cioè, in questo esempio, per quelli corrispondenti a $f \simeq 30$ Hz, può valere la pena di eseguire un best-fit secondo la funzione modello

$$V(t) = V_0 \sin(2\pi f t + \Phi) + c \,, \tag{1}$$

dove l'ampiezza V_0, la frequenza f, il termine di fase costante Φ e l'offset costante c (tiene conto dell'offset introdotto dal generatore ed eventualmente di quello del digitalizzatore) sono lasciati parametri liberi: ci sono dunque ben quattro parametri liberi.

La Fig. 3 mostra i dati con, sovrapposta, la curva del best-fit, e il grafico dei residui normalizzati. Si osserva come i residui abbiano un andamento specifico, in cui valori particolarmente significativi (molto maggiori dell'unità) si ripetono quasi periodicamente. L'interpretazione di questo andamento è immediata tenendo conto delle imperfezioni di campionamento in Arduino: esse, infatti, sono prevalentemente concentrate attorno ad alcuni valori di conteggio che, essendo la funzione periodica, vengono raggiunti periodicamente. Per gli scopi di questa nota non è necessario cercare strategie che tengano eventualmente in conto questi effetti sistematici (in sostanza, accettiamo di sottostimare l'errore sistematico associato ad alcuni campionamenti).

I risultati del best-fit, eseguito usando l'opzione `absolute_sigma = True` (dunque attribuendo un'origine statistica alle incertezze convenzionali) e trascurando le incertezze δt, sono

$$V_0 = (437.1 \pm 0.1) \text{ digit} \tag{2}$$
$$f = (29.85 \pm 0.08) \text{ Hz} \tag{3}$$
$$\Phi = (806 \pm 4) \text{ mrad} \tag{4}$$
$$c = (452.7 \pm 0.1) \text{ digit} \tag{5}$$
$$\chi^2/\text{ndof} = 3275/252 \tag{6}$$

(si omette l'indicazione della covarianza normalizzata tra i parametri per esigenze di spazio [7]).

Come breve commento, osserviamo che il valore della frequenza f è in accordo con la misura eseguita dal frequenzimetro del generatore di funzioni: $f = (29.860 \pm 0.004)$ Hz, dove l'errore è dominato dalla fluttuazione delle cifre meno significative [8]. I parametri V_0 e c sono ovviamente in accordo con il grafico (essi dipendono dall'ampiezza e dall'offset usati in questo esempio), mentre il termine di fase costante Φ, anch'esso in ovvio accordo con il grafico, dipende in maniera random dalla specifica acquisizione.

APPENDICE: SINCRONIZZAZIONE

Come probabilmente diventerà evidente in qualche prossima esperienza, con i segnali periodici ci sono diversi motivi per cui è spesso preferibile realizzare delle acquisizioni *sincrone*, che cioè partono quando si verifica un determinato evento. Dal punto di vista concettuale, l'obiettivo è simile a quello che si consegue quando un oscilloscopio è correttamente "triggerato", ovvero quando la sweep parte in corrispondenza di una certa condizione: in Arduino, infatti, la partenza della sweep è concettualmente equivalente alla partenza dell'acquisizione dei dati.

Ci sono numerose modalità per triggerare un oscilloscopio, ma quella più frequentemente usata si basa sul monitoraggio del segnale in ingresso al trigger, normalmente corrispondente al segnale che si vuole visualizzare: quando questo segnale attraversa, in salita o in discesa a seconda del segno della *slope*, un certo livello, regolato tramite un'apposita manopolina, la sweep parte. Questa strategia potrebbe, in linea di principio, essere implementata pari pari in Arduino, infilando nello sketch un ciclo che viene abbandonato solo quando il segnale digitalizzato supera, o scende al di sotto di, una soglia prefissata.

Dal punto di vista pratico questa implementazione è molto poco ragionevole, se non altro perché essa introduce dei ritardi, anche di carattere aleatorio, tra quando la condizione viene effettivamente verificata da Arduino e quando l'acquisizione ha effettivamente inizio. Infatti la digitalizzazione comporta un tempo finito per essere completata e la condizione che vogliamo verificare, attraverso il confronto tra lettura digitalizzata e valore di soglia, richiede l'esecuzione di istruzioni software, con ulteriori ritardi in parte casuali (latenze del microcontroller). Se, poi, la soglia predisposta cadesse dalle parti dei valori che mandano in crisi il digitalizzatore (normalmente vicino a potenze di 2), allora molto probabilmente si avrebbero forti instabilità di funzionamento.

Per scopi di sincronismo è sempre preferibile usare dei segnali digitali, che in genere sono campionati a rate più elevati e che sono virtualmente privi di fluttuazioni (nel senso che essi possono valere o zero o uno, per cui non c'è bisogno di fare complicate operazioni matematiche di comparazione). Fortunatamente i nostri generatori di funzioni producono un segnale "digitale" (standard TTL) che si accende, o si spegne, periodicamente e automaticamente in sincrono con il segnale (la forma d'onda) generato. A questo segnale di sincronismo si accede attraverso un connettore (lo standard è BNC, ne riparleremo) posto o sul frontale o sul retro dello strumento e indicato come `TTL CMOS OUTPUT`. Nella pratica, e supponendo di produrre onde sinusoidali, questo connettore fornisce una tensione nulla (rispetto alla linea di massa, o terra) finché la sinusoide non raggiunge il proprio valore massimo, per poi assumere un valore attorno a 5 V che resta inalterato finché la sinusoide non raggiunge il proprio valore minimo (il tutto avviene con una qualche ritardo, del tutto trascurabile per i nostri scopi). Questo dà luogo a un segnale di sincronismo che ha la forma di un'onda quadra di ampiezza compresa tra 0 e 5 V (circa), che è sempre *in fase* con la forma d'onda prodotta.

Arduino può facilmente leggere questo segnale usando una delle sue porte *digitali*, ovviamente configurata via sketch come input. In questo esempio, la porta prescelta è quella collegata al pin ∼`5` (boccola verde). Nello sketch, disponibile in rete e sui computer di laboratorio con il nome `synclong2016.ino` [9], si trova una serie di istruzioni software che serve a creare un'attesa che dura finché non viene raggiunta la condizione posta, cioè che il segnale di sincronismo prodotto dal generatore e letto da Arduino tramite la sua porta ∼`5` (nello sketch indicata come `sincPin`) non passa da alto a basso. Per chi è interessato, le istruzioni specifiche sono le seguenti:

```
sinc = digitalRead(sincPin);//legge sincPin
   while (sinc==HIGH) // ciclo di attesa iniziale per sincronizzazione, attende che sincPin vada basso
     {sinc = digitalRead(sincPin);} //legge sincPin
   while (sinc==LOW) // ciclo di attesa iniziale per sincronizzazione, attende che sincPin vada alto
     {sinc = digitalRead(sincPin);}//legge sincPin
```

Record "lunghi"

Come già anticipato, avremo modo in diverse occasioni di apprezzare l'utilità di eseguire acquisizioni dati in maniera sincrona con la forma d'onda prodotta dal generatore. Infatti l'acquisizione sincronizzata permette di costruire dei record "lunghi", che risultano dall'unione di tanti cicli di acquisizione ognuno dei quali, per noi, è composto di 256 misure di tempo e di d.d.p. digitalizzata. In termini molto generali, aumentare la dimensione di un record consente di incrementare la quantità, o qualità, delle informazioni fisiche che i dati acquisiti possono dare.

Questa affermazione trova conforto in un'esperienza che abbiamo già svolto: per costruire dei campioni di una d.d.p. continua siamo già passati attraverso la registrazione di record costituiti appendendo le misure effettuate in diverse acquisizioni, compiute sequenzialmente dentro un ciclo. Questo ci ha permesso, almeno in linea di principio, di accrescere la qualità e quantità delle informazioni, poiché abbiamo potuto determinare media e deviazione standard della d.d.p., e anche (tentare di) analizzare la

distribuzione del campione.

Nell'esperienza già svolta, la d.d.p. misurata era continua, cioè costante nel tempo, e quindi non si poneva alcun problema di sincronizzazione. Quando invece il segnale ha uno specifico andamento temporale, che qui è molto semplice da descrivere (è periodico), allora è necessario preoccuparsi che tutti i cicli di acquisizione avvengano a partire da un istante *determinato* rispetto alla periodicità del segnale stesso. La ricchezza di strategie di acquisizione che comporta la sincronizzazione ci sarà chiara andando più avanti.

Qui, come primo e semplicissimo esempio, ci prefiggiamo l'obiettivo di costruire dei record in cui la durata complessiva dell'intervallo temporale analizzato è maggiore di quanto consentito da una singola acquisizione. In pratica, vogliamo costruire dei record in cui i dati (tempo e valore digitalizzato) dei vari cicli di acquisizione sono attaccati in sequenza, in modo che con una sequenza di acquisizioni sia possibile ricostruire il segnale analizzato su una scala temporale maggiore.

Tutto questo è realizzato nel già citato sketch `synclong2016.ino`: al suo interno viene realizzato un ciclo di acquisizioni (sono 8 per default, per cui il campione risulta costituito da $256 \times 8 = 2048$ coppie di dati, tempo e valore digitalizzato). Ogni acquisizione viene fatta partire con un certo ritardo rispetto alla precedente, e questo certo ritardo è, entro l'incertezza, pari all'istante temporale in cui è avvenuta l'ultima misura dell'acquisizione precedente. In altre parole, nella prima acquisizione del ciclo le misure partono con un certo ritardo (posto pari a zero, entro l'incertezza ed escludendo un paio di misure "a vuoto", utili per ridurre gli artefatti iniziali) rispetto al segnale di sincronismo e terminano dopo un certo intervallo di tempo $\Delta t_{tot,1} \sim 256 \times \Delta t$ (qui si omette per brevità il tempo di digitalizzazione). La seconda acquisizione del ciclo partirà con un ritardo $\Delta t_{tot,1} + \Delta t$ rispetto al segnale di sincronismo, e così via fino a concludere il ciclo [10].

L'acquisizione viene gestita attraverso uno specifico script di Python (`synclong2016.py`), al solito disponibile in rete e nei computer. L'utilità di questa strategia di acquisizione è in questa esperienza piuttosto limitata; però vedremo più avanti che disporre di un record "lungo" apre la possibilità di analizzare i dati con una tecnica, detta *FFT*, che è spesso di grande interesse.

La Fig. 4 mostra un esempio dei risultati: confrontando con la Fig. 2, si nota subito come l'intervallo temporale totale analizzato sia maggiore (di un fattore 8). La grande densità dei punti impedisce di distinguere granché nei primi due pannelli dall'alto: anche questo, volendo, può essere interpretato come un effetto dovuto a un errato campionamento, solo che in questo caso il (sotto-)campionamento è quello eseguito dalla stampante, che imprime un numero finito di punti per unità di superficie. Usando una visualizzazione espansa, cioè facendo lo zoom delle figure, non si notano sostanziali differenze con i dati di Fig. 2. Lo stesso si può senz'altro affermare per il terzo pannello, che mostra una bella forma sinusoidale, ripetuta per un numero di periodi 8 volte maggiore rispetto al corrispondente pannello di Fig. 2. Invece per quanto riguarda il pannello inferiore, quello relativo a $f \simeq 3$ Hz, la differenza è ben visibile: ora, infatti, il problema che avevamo chiamato di sovra-campionamento non esiste più, poiché la lunghezza complessiva del record è tale da permettere un'adeguata ricostruzione della forma d'onda anche in questo caso.

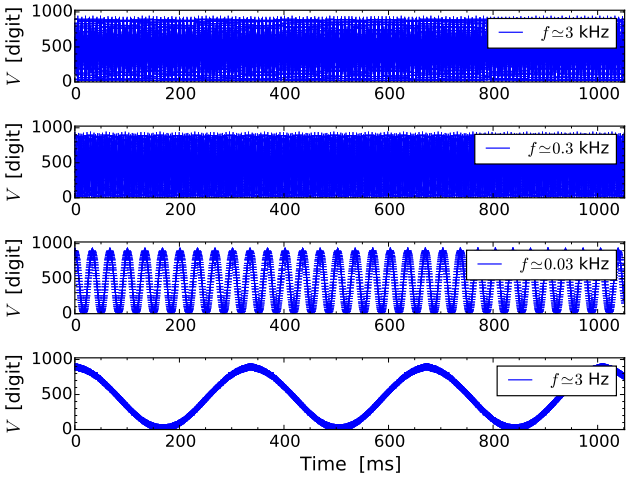

Figura 4. Analogo di Fig. 2 costruito, però, usando acquisizioni su record "lunghi", contenenti un totale di 2048 coppie di dati, secondo quanto descritto nel testo.

---

[1] Naturalmente ci sono importantissimi controesempi a questa affermazione, dovuti in particolare alla natura discreta della carica elettrica. Provate per esempio a trovare il legame tra d.d.p. e carica accumulata in un condensatore di capacità piccolissima, dell'ordine dell'aF. Tuttavia le grandezze di interesse pratico nel mondo macroscopico hanno sicuramente un carattere non discreto.

[2] Un circuito di questo tipo, cioè un generatore di rampa triggerabile, è anche impiegato negli oscilloscopi, dove serve per generare la *sweep* (o spazzata temporale).

[3] È evidente che il sample-and-hold e tutto l'ambaradan necessario al suo funzionamento sono componenti cruciali di un digitalizzatore, dovendo operare su scale temporali particolarmente brevi (a questo fa riferimento l'aggettivo "istantaneo"). Probabilmente questi componenti sono i principali responsabili del comportamento "erroneo" che si riscontra quando Arduino viene impiegato per digitalizzare segnali variabili nel tempo (scalettature e andamenti irregolari).

[4] Questa incertezza di digitalizzazione ha molto a che vedere con l'incertezza di lettura di uno strumento digitale, per esempio il tester. Tuttavia, nel tester digitale l'incer-

tezza di lettura può anche essere influenzata da caratteristiche specifiche del digitalizzatore e della circuiteria elettronica in generale che vi è installata. Per esempio, è noto che, almeno per il modello in uso in laboratorio, essa può dare luogo a fluttuazioni maggiori di un singolo digit, dunque più grandi della cifra meno significativa che si legge sul display.

[5] Il duty-cycle rappresenta, in valore relativo o percentuale, la quantità di tempo in un singolo ciclo in cui il segnale si trova allo stato alto. Un segnale (periodico) con duty-cycle del 50%, o di 0.5, in uscita da una porta `PWM` di Arduino rappresenta di fatto un'onda quadra *simmetrica*.

[6] Prendete questa affermazione con ampio beneficio di inventario, tenendo conto che, in questa esperienza, siamo interessati a una ricostruzione della forma d'onda che sia "visivamente" fedele. Scoprirete (e, probabilmente, scopriremo insieme, almeno in parte) che in certi casi il sotto-campionamento non preclude una dettagliata analisi del segnale, eseguita facendo uso di strumenti (*sviluppo di Fourier*) a cui faremo brevi cenni nel futuro. Semplificando diversi concetti matematici avanzati ed estremizzando le conclusioni, si potrebbe dire che, secondo un teorema chiamato, spesso, *di Nyquist*, per analizzare una forma d'onda periodica attraverso questi strumenti è sufficiente acquisire solo più di due punti per periodo. Quindi è chiaro che il sotto-campionamento di cui ci occupiamo in questa nota riguarda esclusivamente la necessità di distinguere visivamente fra diverse forme d'onda (per esempio, registrando tre punti per periodo siamo tutti d'accordo che sarebbe impossibile stabilire a occhio se la forma d'onda è sinusoidale, triangolare, o di altro tipo).

[7] L'analisi delle covarianze normalizzate mostra che il parametro c è generalmente poco correlato con gli altri, e quasi per niente correlato con V_0. Questa è un'ovvia conseguenza di come è scritta la funzione modello: il termine costante è infatti "influenzato" dal termine di fase Φ (e da V_0). Dunque la scelta dei quattro parametri, anche se conveniente dal punto di vista dell'interpretazione, è non propriamente corretta, poiché la stessa informazione, il valore del termine costante, dipende da diversi parametri, alcuni pressoché scorrelati tra di loro. In alcuni casi, in funzione dei valori numerici effettivamente registrati, può verificarsi che l'algoritmo di minimizzazione usato per il best-fit stenti a convergere proprio per la presenza di parametri pressoché scorrelati tra di loro.

[8] Il fatto che ci sia compatibilità tra la frequenza determinata dal best-fit e quella misurata con il frequenzimetro del generatore è quanto ci si può attendere sulla base di considerazioni banali (la grandezza è la stessa, e quindi il suo valore dovrebbe risultare lo stesso, entro le incertezze). Tuttavia in diversi casi sperimentali si verifica una sostanziale incompatibilità: le due "misure" di f portano a valori "simili", ma non compatibili entro le incertezze. Questo può essere dovuto a diverse circostanze. In primo luogo, è possibile che l'incertezza sul parametro determinato dal best-fit esca sottostimata a causa dell'uso dell'opzione `absolute_sigma = True`, basata sull'affermazione (arbitraria) che le incertezze sui dati sperimentali sono tutte correttamente stabilite e dotate di carattere prevalentemente statistico. C'è poi un'altra motivazione, ben più forte: i due valori di frequenza che confrontiamo tra loro sono ottenuti con due distinti strumenti. La misura della frequenza è direttamente legata alla misura dei tempi (per esempio, per determinare la frequenza si possono contare i cicli del segnale all'interno di un dato intervallo di tempo), per cui, nel confrontare le due "misure", assumiamo che entrambi gli strumenti siano correttamente calibrati rispetto alla misura *assoluta* dei tempi. Questa affermazione, visto il livello di accuratezza (il numero di cifre significative) con cui riusciamo a "misurare" la frequenza delle oscillazioni, può non essere corretta. Per intenderci e semplificare: sappiamo che Arduino misura i tempi con una certa accuratezza, per esempio quella "convenzionale" di ± 4 μs, ma non sappiamo a cosa corrisponde "in assoluto" il singolo microsecondo da lui misurato. Un hint su questo tipo di incertezza, che ha carattere prevalentemente sistematico, può essere trovato leggendo cosa è scritto sul cristallo in quarzo che fornisce la base dei tempi (l'orologio) di Arduino. La scritta è 16.000 MHz, e il numero di cifre significative lascia intuire che la misura dei tempi sia calibrata, in termini assoluti (cioè in riferimento a un qualche standard della misura dei tempi), con un'accuratezza dell'ordine di una parte su 10^4 (o un po' meglio). Lo stesso ragionamento può poi essere applicato al frequenzimetro del generatore di funzioni: dal manuale si capisce che la misura dei tempi è calibrata entro 20 ppm (parti per milione), corrispondenti a 2 parti su 10^5, supponendo che lo strumento sia correttamente "termalizzato". A prevalere è l'incertezza (assoluta e sistematica) di Arduino, e in effetti le discrepanze relative riscontrate tra le due "misure" di frequenza sono spesso dell'ordine di una parte su 10^4.

[9] Allo scopo di velocizzare, per quanto possibile, l'esecuzione dei cicli di acquisizione necessari, lo sketch prevede di ridurre al minimo le fasi di attesa tra un'acquisizione e la successiva, e anche di impiegare una velocità di trasferimento dati attraverso porta seriale un po' più alta rispetto a quella utilizzata in altre circostanze. È stato riscontrato sperimentalmente come questa velocità (impostata in origine a 76800 Baud, cioè 76800 bit/s, ovviamente sia nello sketch che nello script di controllo) risulti eccessiva per alcuni dei computer disponibili in laboratorio. Questo è probabilmente da attribuire, almeno in parte, a una scadente implementazione dell'emulazione di porta seriale via USB eseguita nei sistemi operativi.

[10] Se vi prenderete la briga di analizzare lo sketch, troverete alcune stranezze, o complicazioni, nel definire il ritardo a cui devono partire i vari cicli di acquisizione. Queste stranezze sono dovute a una limitazione di Arduino, che non consente di ottenere ritardi espressi in microsecondi di durata arbitrariamente grande. Per questo motivo il ritardo è prima impartito via software in unità di millisecondi e quindi di microsecondi per la parte rimanente. Naturalmente questa necessità comporta un'ulteriore fonte di incertezza per la determinazione dei tempi tra la fine di un ciclo e l'inizio di quello successivo, che per il momento possiamo ritenere trascurabile per i nostri scopi.