Curva caratteristica del diodo con Arduino

francesco.fuso@unipi.it

(Dated: version 12 - Francesco Fuso, 3 dicembre 2022)

Questa nota discute alcuni aspetti di interesse per l'esperimento di registrazione della curva caratteristica I-V di un diodo bipolare a giunzione p-n condotta in laboratorio usando Arduino.

I. INTRODUZIONE

L'esperimento considerato in questa nota è piuttosto semplice dal punto di vista concettuale. Arduino è usato come scheda di input/output (I/O) del computer in una modalità che permette di automatizzare la presa dati e raccogliere un numero sufficiente di punti per ricostruire con buon dettaglio una curva sperimentale. La curva in questione è la cosiddetta *curva caratteristica I-V* di un diodo a giunzione bipolare (p-n) in silicio.

Un diodo a giunzione si comporta come un componente che ha una risposta decisamente non ohmica. Infatti la corrente di intensità I che attraversa la giunzione non è linearmente proporzionale alla differenza di potenziale ($\Delta V = V_A - V_K$, con V_A e V_K potenziali elettrici di anodo e catodo rispetto allo stesso riferimento) applicata ai terminali del componente. Secondo il modello di S., la legge che descrive il comportamento di una giunzione bipolare in un materiale semiconduttore è

$$I = I_0 \left[\exp\left(\frac{\Delta V}{\eta V_T} \right) - 1 \right] , \qquad (1)$$

con I_0 *corrente di saturazione inversa*, tipicamente qualche nA per i diodi usati in laboratorio, dunque molto piccola, η parametro costruttivo di valore tipico $\eta \lesssim 2$ per gli ordinari diodi al silicio, V_T differenza di potenziale, talvolta definita *termica*, legata alla temperatura di operazione T, alla carica elementare e e alla costante di Boltzmann k_B attraverso la relazione $eV_T = k_B T$. Poiché $k_B T \simeq 1/40$ eV a temperatura ambiente, si ha $V_T \simeq 26$ mV (per $T \simeq 300$ K). L'Eq. 1 è assunta come funzione modello per la curva I-V del diodo.

Ricostruire sperimentalmente la curva richiede di avere un generatore di d.d.p. variabile, di misurare la d.d.p. ΔV applicata al diodo e la corrispondente intensità di corrente I che vi fluisce. Per automatizzare la procedura sono possibili due strade: (i) campionare e digitalizzare con Arduino i dati necessari in presenza di una d.d.p. che varia nel tempo, ad esempio una rampa (o una forma d'onda triangolare), prodotta da un generatore esterno; (ii) utilizzare Arduino anche per creare la d.d.p. variabile. La seconda strada è probabilmente più affascinante dal punto di vista tecnico, poiché consente di applicare praticamente un approccio piuttosto interessante e di eseguire l'intera misura senza bisogno di generatori esterni.

Ingredienti principali della configurazione sperimentale sono:

1. ottenere una d.d.p. variabile, ovvero una rampa di tensione che evolve nel tempo in modo controllato, con tanti piccoli gradini;

2. campionare e digitalizzare il valore della d.d.p. applicata al diodo per ogni gradino della rampa;

3. registrare la corrispondente intensità di corrente che circola nel diodo; poiché un digitalizzatore consente solo misure di d.d.p., questo punto richiede di "convertire" l'intensità di corrente in una opportuna tensione, cosa che può facilmente essere realizzata a posteriori, cioè lavorando sui dati grezzi acquisiti e sfruttando la definizione di resistenza.

II. MODALITÀ PWM DI ARDUINO E INTEGRATORE/LIVELLATORE

Come ben sappiamo, Arduino contiene al suo interno un certo numero di convertitori A/D (analogico/digitali), ben 12 nel caso di Arduino Due. Per l'esperimento che abbiamo in mente ci piacerebbe molto avere anche dei convertitori D/A in grado di produrre una determinata d.d.p. attraverso opportuni comandi inseriti nello sketch. Purtroppo realizzare un "buon" convertitore D/A è piuttosto complicato dal punto di vista tecnologico: Arduino Due dispone di due porte di uscita analogiche, che però sono limitate sia in termini di corrente che può essere fornita, insufficiente per i nostri scopi, che per intervallo di valori (nominalmente compresi tra 0.55 e 2.75 V).

Sono inoltre disponibili numerose (ben 54 in Arduino Due!) porte digitali che possono essere configurate individualmente come input o output: in quest'ultimo caso, esse, quando poste a livello "alto", forniscono una d.d.p. V_{max} nominalmente di poco inferiore a 3.3 V rispetto alla linea di massa, o terra. Su alcune (12) di queste porte è disponibile un'interessantissima opzione: esse possono operare in modalità *Pulse Width Modulation* (PWM). Questo significa che in uscita si può trovare un'*onda quadra con duty cycle variabile* da zero (onda quadra "spenta", cioè nessun segnale in uscita) al massimo (onda quadra "sempre accesa", cioè segnale continuo pari a V_{max}). Il duty cycle dell'onda quadra, o treno di impulsi, così formata è aggiustabile in maniera digitale agendo su un carattere, cioè su un byte: dunque sono possibili $2^8 = 256$ livelli diversi di duty cycle che possono essere scelti attraverso un'opportuna istruzione dello sketch. Un software concepito in modo da includere un

loop in cui il duty cycle viene sequenzialmente incrementato e acquisire i segnali rilevanti in ogni passo della sequenza è il punto di partenza per realizzare l'esperimento. Per i nostri scopi è tuttavia necessario convertire l'onda quadra con duty cycle variabile in una d.d.p. continua, o *quasi-continua*.

Un treno di impulsi è ovviamente una forma d'onda *non* alternata: variando il duty cycle si modifica, in sostanza, il valore medio del segnale. Quindi usando un integratore RC opportunamente dimensionato, cioè con un tempo caratteristico $\tau >> T$, dove T è il periodo del treno di impulsi, è possibile ottenere un segnale quasi-continuo, di valore variabile virtualmente tra 0 (treno di impulsi sempre spento) e V_{max}.

Come illustrato nel seguito, l'integratore RC può anche essere realizzato usando *solo* un condensatore di capacità C sufficientemente grande, ovviamente posto "in parallelo" al segnale, cioè tra segnale e linea di massa, o terra. Infatti una qualche resistenza, per esempio la resistenza interna di uscita (resistenza di Thévenin) della porta di Arduino, si troverà di sicuro a monte del condensatore. Anche se la costante tempo dell'integratore potrebbe non essere nota con sufficiente accuratezza (ma, volendo, la resistenza di Thévenin potremmo anche misurarla!), il circuito si comporterebbe sempre da integratore, a patto di soddisfare la condizione $\tau >> T$. Il treno di impulsi prodotto da Arduino ha una frequenza nominale di circa 1 kHz (essa può essere aumentata, ma non faremo uso di questa opzione), per cui un tempo caratteristico dell'ordine delle decine di ms dovrebbe già garantire un'efficace integrazione temporale. Supponendo (in analogia con Arduino Uno, per il Due non lo so) una resistenza di Thèvenin dell'ordine della decina, o decine, di ohm, si potrebbe ipotizzare una capacità dell'ordine dei mF: benché essa appaia esagerata per componenti ordinari, la richiesta può essere soddisfatta usando condensatori *elettrolitici*. Un'analisi nel dominio delle frequenze di un integratore che opera in condizioni simili a quelle del circuito che stiamo descrivendo è riportata in Appendice A: potrete ispirarvi al suo contenuto per realizzare una parte della famosa relazione/esercizio obbligatorio durante la pausa invernale.

Sicuramente un'analisi qualitativa del circuito può anche essere eseguita in analogia con quella che si adotta per i *livellatori* normalmente implementati nei raddrizzatori/livellatori a singola semionda. Si può infatti vedere il condensatore come un serbatoio di cariche che provengono da Arduino (quando l'onda quadra è a livello alto) e che vengono fornite al carico, cioè al diodo, quando l'onda quadra è a livello basso. Il risultato è una d.d.p. quasi-continua a causa della presenza di un *ripple*, cioè di oscillazioni che avvengono attorno a un valore simile alla media temporale dell'onda quadra. Presenza ed eventuale quantificazione del ripple sono verificabili durante l'esperimento, osservando con l'oscilloscopio i segnali rilevanti.

A. Configurazione circuitale

Dobbiamo a questo punto definire il metodo di misura dell'intensità di corrente I che fluisce nel diodo. La soluzione più semplice consiste nell'inserire una resistenza R_D in serie al diodo: infatti, per definizione di resistenza, è semplicemente $I = \Delta V_{RD}/R_D$, dove ΔV_{RD} è la caduta di potenziale misurata ai capi di questa resistenza.

Dal punto di vista pratico, dato che le porte analogiche di Arduino misurano d.d.p. relative alla linea di massa o terra, la misura ΔV_{RD} deve essere eseguita per differenza, $\Delta V_{RD} = V_1 - V_2$, tra le d.d.p. (riferite a massa o terra) che si misurano a monte (V_1) e a valle (V_2) di R_D. La d.d.p V_2 è inoltre quella che di fatto si trova ai capi del diodo, cioè corrisponde a ΔV nella simbologia di Eq. 1, per cui la misura indipendente di queste due d.d.p. consente, assieme alla conoscenza di R_D, di trovare tutte le grandezze necessarie per ricostruire la curva I-V.

In definitiva, lo schema circuitale adottato è quello di Fig. 1: in esso si prevede l'impiego della porta digitale PWM `5` (boccola verde) e delle porte analogiche `A0` e `A2` (boccole blu e gialla), rispettivamente per la misura di V_1 e V_2. È inoltre necessario collegare debitamente la boccola nera di Arduino, in modo che le d.d.p. siano correttamente riferite alla linea di massa, o terra.

Il dimensionamento di R_D va stabilito cum grano salis. Da un lato essa dovrebbe essere sufficientemente grande da permettere un'adeguata sensibilità nella misura di I. Infatti, come discuteremo ulteriormente in seguito, la sensibilità del digitalizzatore di Arduino è finita e vale circa 3 mV. Se volessimo, ad esempio, misurare variazioni di corrente dell'ordine di qualche μA, poniamo 3 μA, R_D dovrebbe essere dell'ordine del kohm, o superiore.

Dall'altro lato, al crescere del valore di R_D aumenta la caduta di potenziale ai capi della resistenza. Pur avendo un benefico effetto in termini di limitazione della corrente richiesta alla porta digitale di Arduino (il suo valore limite, da datasheet, è 15 mA, oltre il microcontroller si può danneggiare), un'eccessiva caduta di potenziale potrebbe rendere troppo piccola la massima ΔV applicata al diodo e, di conseguenza, non permettere di ricostruire una curva caratteristica significativa. Il consiglio pratico è quello di provare diversi valori di R_D scelti, ad esempio, nel range $0.33 - 3.3$ kohm e vedere come viene fuori la curva.

Per quanto riguarda il condensatore, benché una capacità molto grande favorisca, in linea di principio, il livellamento, usare valori eccessivi può rendere estremamente lenta la variazione di d.d.p. applicata al diodo. Osservate come, a patto che la misura di V_1 e V_2 sia simultanea, ovvero avvenga con un ritardo trascurabile, la presenza di ripple non è elemento che pregiudica la ricostruzione della curva. Tipicamente è sufficiente utilizzare $C \simeq 100 - 470$ μF (valori nominali, negli elettrolitici la tolleranza è tradizionalmente molto alta, tipo $\pm 50\%$!). Volendo, potete fare prove con diversi condensatori (stando bene attenti a collegarli nel modo giusto!) e scegliere quello che permette di ricostruire la curva I-V più "bella".

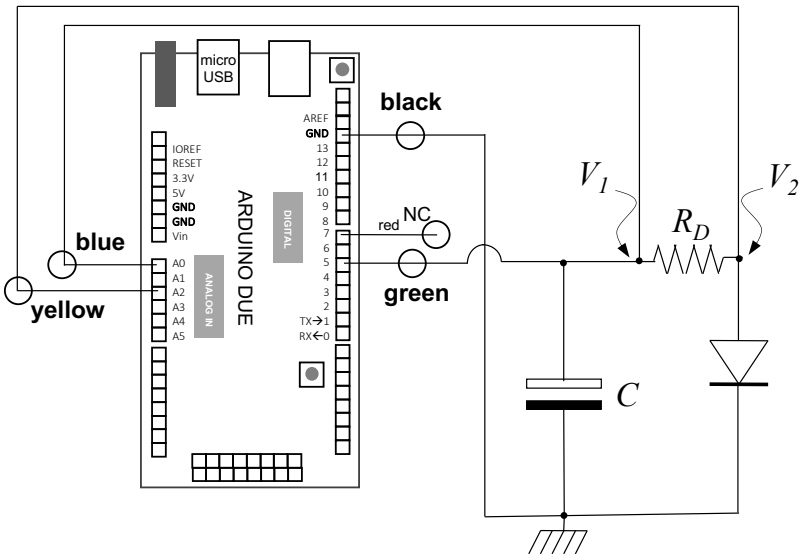

Figura 1. Schema circuitale dell'esperimento con indicate le quattro connessioni da effettuare alla scheda Arduino Due (NC significa non collegato).

B. Calibrazione e incertezze

Lo scopo dell'esperimento è quello di costruire una curva I-V da cui, oltre all'andamento, sia possibile dedurre i valori caratteristici in gioco, I_0 e ηV_T in Eq. 1, espressi nelle debite unità fisiche. Pertanto è necessario convertire in [V] le grandezze digitalizzate (misurate nativamente in [digit]).

Questo obiettivo richiede di calibrare in qualche modo i digitalizzatori di Arduino. Date le finalità dell'esperimento è possibile limitarsi alla cosiddetta calibrazione "alternativa": essa si basa sull'ipotesi che i digitalizzatori abbiano risposta lineare (senza offset), che tutte le porte analogiche si comportino allo stesso modo e che la massima d.d.p. digitalizzabile sia pari alla massima d.d.p. V_{max} prodotta in uscita su una delle porte digitali. Poichè nella configurazione in uso la dinamica di digitalizzazione è di 10 bit, cioè sono possibili $2^{10} = 1024$ valori distinti, il fattore di calibrazione ξ (in [V/digit]) risulta semplicemente dalla divisione $\xi = V_{max}/1023$: essendo $V_{max} \simeq 3.3$ V, si ottiene tipicamente $\xi \sim 3$ mV. Dato che al termine dello sketch la porta `5` si trova a livello alto (e ci rimane finché l'acquisizione non viene fatta ripartire), per conoscere V_{max} è sufficiente misurare con il multimetro digitale, al *termine* delle acquisizioni, la d.d.p. fra tale porta digitale e la linea di massa, o terra, cioè tra la boccola verde e quella nera di Arduino. È ovvio che questa misura deve essere fatta *a circuito aperto*, cioè scollegando quanto si trova di seguito alla porta. Infatti, come già affermato, nel diodo può circolare corrente di intensità tutt'altro che trascurabile, in grado di provocare una rilevante caduta di potenziale sulle resistenze.

La calibrazione ottenuta è ovviamente affetta da un'incertezza che, supponendo valide le assunzioni che abbiamo fatto, è pari, in termini relativi, all'incertezza con cui si misura V_{max}. Per determinare l'incertezza nativa delle misure digitalizzate possiamo usare il consueto approccio, attribuendo l'errore *convenzionale* di ± 1 digit alla lettura digitalizzata. Poiché in questo esperimento convertiamo la lettura digitalizzata in unità fisiche, dovremo sommare (eventualmente in quadratura) a questo errore quello dovuto alla calibrazione di Arduino. Questa operazione è sufficiente a determinare l'incertezza sulla grandezza che sta sull'asse orizzontale del grafico che intendiamo costruire, cioè $\Delta V = V_2$.

Invece la grandezza che compare sull'asse verticale del grafico che vogliamo produrre, cioè l'intensità di corrente I, è valutata attraverso la relazione

$$I = \frac{V_1 - V_2}{R_D} \,, \tag{2}$$

dove tutte le grandezze sono già state definite in precedenza. Per stimarne l'incertezza dovremo fare debito uso delle regole di propagazione dell'errore, tenendo conto anche dell'incertezza sulla misura di R_D, da fare con il multimetro.

Anche se l'argomento dovrebbe esservi stra-noto, può essere utile esprimere l'incertezza δI in funzione di quella sulle grandezze misurate. Seguendo la regola generale che prevede, nel caso di propagazione, di partire da una espressione "minimale" allo scopo di evitare sovrastime dell'errore, riscriviamo l'Eq. 2 in termini dei valori digitalizzati (cioè espressi in [digit]) $\tilde{V}_1$ e $\tilde{V}_2$:

$$I = \frac{\tilde{V}_1 - \tilde{V}_2}{R_D}\xi = \frac{\tilde{V}_{12}}{R_D}\xi \,, \tag{3}$$

con $\tilde{V}_{12} = \tilde{V}_1 - \tilde{V}_2$.

Assumendo per il momento la propagazione dell'errore massimo δI_{max}, si ha

$$\delta I_{max} = \delta\tilde{V}_{12}\frac{\partial I}{\partial \tilde{V}_{12}} + \delta\xi\frac{\partial I}{\partial \xi} + \delta R_D\frac{\partial I}{\partial R_D} = \quad (4)$$

$$= \delta\tilde{V}_{12}\frac{\xi}{R_D} + \delta\xi\frac{\tilde{V}_{12}}{R_D} + \delta R_D\frac{\tilde{V}_{12}\xi}{R_D^2}\,, \quad (5)$$

con ovvio significato dei simboli.

Poiché in alcune delle misure, quelle corrispondenti a una intensità di corrente minore della sensibilità della misura, $\tilde{V}_{12} = 0$, può verificarsi che gli ultimi due addendi all'ultimo membro di Eq. 4 siano nulli. Il primo addendo, però, non è mai nullo a causa dell'errore di digitalizzazione. Infatti $\delta\tilde{V}_{12} = \delta\tilde{V}_1 + \delta\tilde{V}_2 = 1 + 1 = 2$ digit, dove abbiamo usato l'incertezza convenzionale ± 1 digit per ognuno dei due valori digitalizzati. Osservate che, essendo le porte analogiche di Arduino multiplexate, lo stesso digitalizzatore viene usato sequenzialmente per i due canali, per cui le misure di V_1 e V_2 non possono essere considerate indipendenti e quindi la somma in quadratura non è formalmente giustificata.

Invece può essere opportuno sommare in quadratura i termini di Eq. 3, ottenendo

$$\delta I = \sqrt{\left(\delta\tilde{V}_{12}\frac{\xi}{R_D}\right)^2 + \left(\delta\xi\frac{\tilde{V}_{12}}{R_D}\right)^2 + \left(\delta R_D\frac{\tilde{V}_{12}\xi}{R_D^2}\right)^2}\,. \quad (6)$$

Banali passaggi matematici portano a riscrivere l'Eq. 6 nella forma

$$\delta I = \sqrt{\left(\frac{\delta\tilde{V}_{12}}{\tilde{V}_{12}}I\right)^2 + \left(\frac{\delta\xi}{\xi}I\right)^2 + \left(\frac{\delta R_D}{R_D}I\right)^2}\,, \quad (7)$$

cioè, come ovvio, l'incertezza relativa è data dalla somma (in quadratura) delle incertezze relative dei tre fattori in Eq. 3. Il punto chiave, però, è che nel primo addendo di Eq. 7 sia I che $\tilde{V}_{12}$ possono andare a zero contemporaneamente e l'implementazione dell'errore relativo nello script di Python dedicato al best-fit può facilmente portare a incertezze nulle o infinite, a seconda di come si scrivono le funzioni. Oltre a essere errata dal punto di vista fondamentale, la presenza di valori nulli o infiniti nell'array passato alla routine di minimizzazione del χ^2 ne impedisce la convergenza, rendendo impossibile eseguire il best-fit.

Dunque state sempre attenti a valutare correttamente le incertezze, specialmente quando dovete applicare le formuline della propagazione dell'errore a funzioni che contengono differenze.

C. Esempio di misura e commenti

Un esempio di misura effettuata in laboratorio didattico è mostrato in Fig. 2. In essa sono state compiute tutte le operazioni necessarie a dimensionare fisicamente le grandezze e a determinare l'incertezza secondo quanto prima stabilito. La figura riporta anche il best-fit secondo la funzione modello di Eq. 1 (parametri liberi I_0 e ηV_T), e il grafico dei residui.

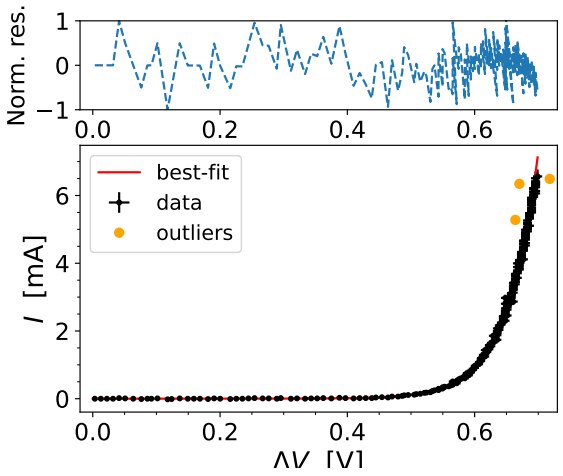

Figura 2. Esempio di misura condotta in laboratorio didattico secondo quanto descritto nel testo; la linea rossa continua rappresenta il best-fit condotto secondo l'Eq. 1 (risultati nel testo), i marker arancioni individuano i potenziali outliers, secondo quanto specificato nel testo. Il pannello superiore riporta il grafico dei residui normalizzati depurati dagli outliers.

I risultati del best-fit, eseguito considerando gli errori efficaci e con l'opzione `absolute_sigma = False` in ossequio alla presenza di errori non statistici dovuti alle calibrazioni, sono

$$I_0 = (3.8 \pm 0.2)\ \text{nA} \quad (8)$$

$$\eta V_T = (48.4 \pm 0.2)\ \text{mV} \quad (9)$$

$$\chi^2/\text{ndof} = 62/254 \quad (10)$$

$$\text{Norm. cov.} = 0.997\,. \quad (11)$$

Per divertimento è stata anche eseguita una variante di best-fit in cui sono stati (arbitrariamente) battezzati come outliers quei punti sperimentali distanti oltre 3 barre di errore dalla previsione del best-fit (i punti corrispondenti, solo tre nell'esempio qui considerato, sono marcati con pallini arancioni). I risultati del best-fit fatto escludendo gli outliers sono

$$I_0 = (3.5 \pm 0.2)\ \text{nA} \quad (12)$$

$$\eta V_T = (48.1 \pm 0.2)\ \text{mV} \quad (13)$$

$$\chi^2/\text{ndof} = 32/251 \quad (14)$$

$$\text{Norm. cov.} = 0.997\,. \quad (15)$$

Non si riporta qui una discussione su questi risultati, che comunque seguono qualitativamente le aspettative. In particolare, la curva mostra come il diodo entri in conduzione, cioè faccia passare una rilevante intensità di corrente, quando ΔV è sufficientemente alto. In gergo tecnico si individua spesso una d.d.p. detta *tensione di*

soglia, che indichiamo con V_{thr}: si può grossolanamente affermare che il diodo va in conduzione per $\Delta V \simeq V_{thr}$. In generale si accetta che, per un ordinario diodo al silicio, sia $V_{thr} \simeq 0.6 - 0.7$ V; la definizione di V_{thr} in effetti non è univoca. Spesso si dice che V_{thr} è la d.d.p. che corrisponde a un'intensità di corrente dell'ordine di 1/10 la massima corrente permessa (a causa della dissipazione per effetto Joule, il diodo si brucia a correnti più elevate di un qualche massimo). Il diodo usato in laboratorio dovrebbe assomigliare a un modello 1N914, per il quale il datasheet riporta un valore massimo di corrente di 300 mA, per cui V_{thr} dovrebbe corrispondere a $I \sim 30$ mA. Questa corrente è fuori dalle possibilità del nostro esperimento, però, visto l'andamento molto ripido del grafico, possiamo aspettarci a occhio che essa verrebbe raggiunta per $\Delta V \sim 0.7$ V.

Sottolineiamo poi un paio di peciuliarità della misura. In primo luogo i dati risultano evidentemente non equispaziati lungo l'asse orizzontale. Questa è un'ovvia conseguenza del comportamento non ohmico del diodo: mano a mano che l'acquisizione procede, cioè che l'onda quadra tende a rimanere accesa per tempi maggiori, l'intensità di corrente I aumenta in modo non lineare e la caduta di potenziale su R_D aumenta di conseguenza. Quindi V_2, ovvero ΔV, tende a un valore asintotico che si attesta attorno a V_{thr}.

Inoltre, anche se nell'esempio di Fig. 2 il problema non è particolarmente evidente, si possono verificare delle "fluttuazioni" che rendono non biunivoca la corrispondenza tra ΔV e I: queste fluttuazioni possono essere dovute a diversi meccanismi, per esempio il verificarsi di *spikes*, o altri "rumori", nei dati digitalizzati, il non simultaneo campionamento di V_1 e V_2 (le porte analogiche sono *multiplexate*, cioè lo stesso digitalizzatore viene usato sequenzialmente per più porte), la sensibilità finita (e discreta) del digitalizzatore, etc.

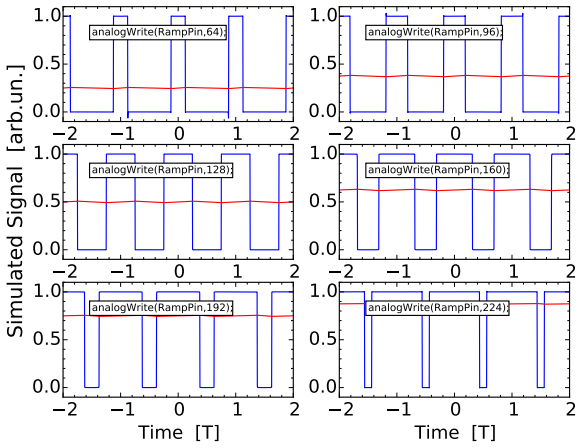

Figura 3. Esempi di simulazione di treni di impulsi con duty cycle variabile (curve blu) e di uscita dall'integratore (curve rosse), come discusso nel testo. I vari grafici si riferiscono a diverse scelte del duty cycle: in legenda è riportata l'istruzione software usata nello sketch di Arduino.

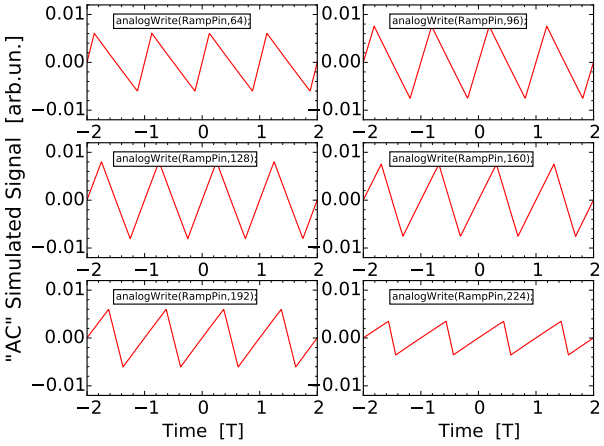

Figura 4. Analogo di Fig. 3 per le sole tracce rappresentative del segnale in uscita dall'integratore: per questa figura, a tale segnale è stato sottratto il valore medio nel tempo, simulando, in pratica, un'osservazione con oscilloscopio accoppiato in AC.

APPENDICE A: SERIE DI FOURIER PER IL TRENO DI IMPULSI

È sicuramente interessante ricostruire numericamente il comportamento di un integratore al cui ingresso abbiamo un treno di impulsi, ovvero un'onda quadra con un certo duty cycle. Lo strumento concettuale che consente di eseguire la simulazione è basato sulla serie di Fourier: è infatti sufficiente esprimere il treno di impulsi in termini dei suoi coefficienti dell'espansione di Fourier, e quindi applicare alle varie componenti, cioè alle varie armoniche, la funzione di trasferimento [guadagno $G(f)$ e sfasamento $\Delta\phi$] di un filtro passa-basso RC.

Userete lo strumento concettuale in una prossima relazione/esercizio obbligatorio, e in quella sede potrete apprezzare il contenuto di questa appendice.

Due osservazioni preliminari:

- nella simulazione considereremo un tempo caratteristico ragionevole per le effettive condizioni sperimentali e trascureremo gli effetti della presenza del diodo, cioè ci limiteremo a descrivere la d.d.p. quasi-continua in uscita dall'integratore, o livellatore;
- il treno di impulsi in ingresso all'integratore non sarà mai alternato: infatti, anche per un'onda quadra simmetrica (duty cycle pari al 50%), Arduino fa sì che la d.d.p. prodotta oscilli tra zero e il valore massimo, cioè sia sempre positiva, per cui la sua media non è mai nulla.

Supponendo per semplicità un treno di impulsi rappresentato da una funzione $g(t)$ *pari* nel tempo (cioè l'istante $t = 0$ si trova a metà strada della parte alta di un im-

pulso) e di ampiezza unitaria (valore minimo 0, valore massimo 1, in unità arbitrarie), si ha che $g(t)$ può essere rappresentata dalla seguente serie di *coseni*:

$$g(t) = \delta + \sum_{k=1}^{n} \frac{2}{k\pi} \sin(k\pi\delta) \cos(\omega_k t) \ , \qquad (16)$$

dove $\delta = \tau'/T$ rappresenta il duty cycle variabile tra 0 e 1 (τ' è qui la durata della parte alta dell'impulso e T è il periodo del treno di impulsi) e $\omega_k = k\omega$ è la frequenza angolare dell'armonica k-esima. Poiché talvolta il duty cycle si esprime come percentuale D (la percentuale di tempo in cui l'impulso si trova a livello alto rispetto al periodo), scriviamo l'ovvia conversione $D = \delta \times 100$. Infine, tenendo conto del fatto che in Arduino il duty cycle può essere impostato via software attraverso l'intero `i` variabile tra 0 e 255 (vedi lo sketch), l'altrettanto ovvia conversione che lega δ a `i` recita $\delta =$ `i`$/256$.

Come noto, per determinare la funzione $g_{out}(t)$ che rappresenta l'uscita dell'integratore le varie armoniche di frequenza $f = \omega_k/(2\pi)$ vanno moltiplicate per la funzione che esprime il guadagno del passa-basso, $G(f) = 1/\sqrt{1 + (f/f_T)^2}$, e sfasate di $\Delta\phi = \arctan(-f/f_T)$. La Fig. 3 riporta un esempio dei risultati per vari valori del duty cycle (in legenda si riporta per chiarezza proprio l'istruzione da usare nell'eventuale sketch di Arduino): i grafici mostrano in blu il treno di impulsi simulato e in rosso l'uscita simulata dell'integratore. In questo esempio il rapporto tra tempo caratteristico τ dell'integratore e periodo T del treno di impulsi è $\tau/T \simeq 16$, che dovrebbe essere ragionevole dal punto di vista sperimentale. Per chiarezza, la Fig. 4 mostra il segnale in uscita dall'integratore a cui è stato sottratto il valore medio nel tempo: dunque i pannelli riportati in questa figura rappresentano una sorta di simulazione di quanto si vedrebbe usando un oscilloscopio con ingresso accoppiato in AC, che permette di apprezzare la discrepanza rispetto a un segnale idealmente costante con grande sensibilità.

Il risultato è in accordo con le attese: effettivamente all'uscita dell'integratore si ritrova un livello quasi-continuo che dipende linearmente dal duty cycle impostato. Il carattere quasi-continuo è dovuto alla frequenza di taglio dell'integratore che abbiamo supposto finita, cioè diversa da zero: essa dà luogo a un ripple (oscillazione attorno al valore medio) che non riesce a essere integrato in maniera completa e che, come risulta chiaro da Fig. 4, è particolarmente rilevante per i valori intermedi del duty cycle.

APPENDICE B: SCRIPT DI PYTHON E SKETCH DI ARDUINO

Come nostro solito, ci serviamo di uno script di Python per gestire la partenza delle operazioni di Arduino e per gestire il trasferimento dei dati da questo al computer tramite porta seriale USB. Sketch e script sono disponibili in rete (nome `diodo2016`) e pre-installati nei computer di laboratorio.

Nello script è possibile fornire l'intervallo nominale di campionamento (tempo tra un campionamento e il successivo) in unità di 10 ms: il default è 10 ms, ma il parametro non è affatto critico.

Il testo dello script è il seguente:

```
import serial # libreria per gestione porta seriale (USB)
import time   # libreria per temporizzazione

print('Apertura della porta seriale\n') # scrive sulla console (terminale)
ard=serial.Serial('/dev/ttyACM0',9600)  # apre la porta seriale /dev/ttyACM0
time.sleep(2)   # aspetta due secondi
ard.write(b'1')#intervallo (ritardo) in unita' di 10 ms <<<< questo si puo' cambiare (default 10 ms)
 print('Start!\n') # scrive sulla console (terminale)
Directory='/home/studenti/dati_arduino/'   # nome directory dove salvare i file dati
FileName=(Directory+'diodo.txt') # nomina il file dati <<<< DA CAMBIARE SECONDO GUSTO
outputFile = open(FileName, "w+" ) # apre file dati in scrittura

# loop lettura dati da seriale (sono 256 righe, eventualmente da aggiustare)
for i in range (0,256):
    data = ard.readline().decode() # legge il dato e lo decodifica
    if data:
            outputFile.write(data) # scrive i dati nel file

outputFile.close() # chiude il file dei dati
ard.close() # chiude la comunicazione seriale con Arduino
print('end') # scrive sulla console (terminale)
```

Il testo dello sketch di Arduino è il seguente:

```
const unsigned int RampPin = 5;  //pin 5 uscita pwm per generare la rampa
const unsigned int analogPin_uno=0;  //pin A0 per lettura V1
const unsigned int analogPin_due=2;  //pin A2 per lettura V2
unsigned int i=0;  //variabile che conta gli step durante la salita della rampa
int V1[256];  //array per memorizzare V1 (d.d.p, letta da analogPin_uno)
int V2[256];  //array per memorizzare V2 (d.d.p, letta da analogPin_due)
int delay_ms;  //variabile che contiene il ritardo tra due step successivi (in unita' di 10 ms)
int start=0;  //flag per dare inizio alla misura

//Inizializzazione
void setup()
{
  pinMode(RampPin, OUTPUT);  //pin pwm RampPin configurato come uscita
  Serial.begin(9600);  //inizializzazione della porta seriale
  Serial.flush(); // svuota il buffer della porta seriale
}

//Ciclo di istruzioni del programma
void loop()
{
    if (Serial.available() >0) // Controlla se il buffer seriale ha qualcosa
      {
        delay_ms = (Serial.read()-'0')*10; // Legge il byte e lo interpreta come ritardo (unita' 10 ms)
        Serial.flush(); // Svuota il buffer della seriale
start=1; // Pone il flag start a uno
      }
  if(!start) return // solo se il flag e' a uno parte l'acquisizione
    delay(500); // attende 0.5s per evitare casini
    if (start==1)
      analogWrite(RampPin,0); // all'inizio pone a 0 la RampPin per favorire scarica condensatore
    delay(1500); // attende 1.5s per scaricare condensatore
    for(i=0;i<256;i++)  //il valore che definisce il duty cycle dell'onda quadra e' scrivibile su 8 bit
                        //cioe' assume valori da 0-->duty cycle = 256
      {
       analogWrite(RampPin, i); //incrementa il duty cycle di uno step
       delay(delay_ms);   //aspetta il tempo impostato
       V1[i]=analogRead(analogPin_uno);   //legge il pin analogPin_uno
       V2[i]=analogRead(analogPin_due);   //legge il pin analogPin_due
       delay(delay_ms);   //aspetta il tempo impostato
      }
     for(i=0;i<256;i++)  //nuovo ciclo che scorre gli array di dati e li scrive sulla seriale
     {
       Serial.print(V1[i]);
       Serial.print(" ");
       Serial.println(V2[i]);
     }
    start=0; // Annulla il flag
    Serial.flush(); // svuota il buffer della porta seriale
  }
```