Curva caratteristica del diodo con Arduino

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

(Dated: version 10 - Lara Palla e Francesco Fuso, 9 dicembre 2016)

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

I. INTRODUZIONE

L'esperienza considerata in questa nota è piuttosto semplice dal punto di vista concettuale. Arduino è usato come scheda di input/output (I/O) in una modalità che permette di automatizzare la presa dati e di 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 (ΔV) applicata ai terminali del componente. Secondo il modello di Shockley, la legge che descrive il comportamento 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* (del valore tipico dell'ordine di $1 - 10$ nA per i diodi usati in laboratorio, dunque molto piccola), η parametro costruttivo di valore tipico $\eta \simeq 1.5 - 2$ 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.

Tracciare la curva in questione richiede, in una semplice configurazione sperimentale, di poter disporre di un generatore di d.d.p. variabile, di misurare il valore effettivo ΔV applicato e di misurare la corrispondente intensità di corrente I che fluisce nel diodo. Tale semplice configurazione sperimentale è descritta schematicamente in Fig. 1(a) in cui si suppone implicitamente di poter trascurare gli effetti di tutte le resistenze interne (quella del generatore e del misuratore di corrente, ritenute trascurabili, e quella del misuratore di d.d.p., ritenuta così grande da non sottrarre corrente al resto del circuito).

In laboratorio non disponiamo di un generatore di d.d.p. variabile: potremmo realizzarne facilmente uno, per esempio costruendo un partitore di tensione con una resistenza variabile (potenziometro). Tuttavia, per avere una ricostruzione significativa della curva I-V [rappresentata per esempio in Fig. 1(b)] occorre registrare i dati su un numero relativamente elevato di punti corrispondenti a piccole variazioni del valore di ΔV, cosa non semplice dal punto di vista pratico.

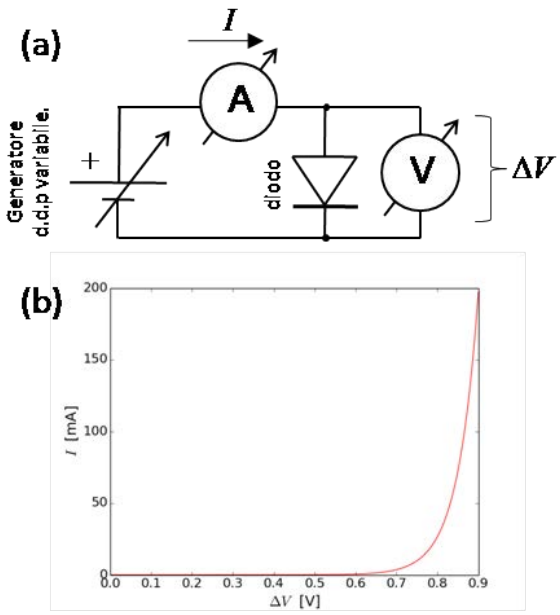

Figura 1. Schema concettuale di un'esperienza di di ricostruzione della curva caratteristica I-V di un diodo (a) e curva calcolata secondo Eq. 1, supponendo $\eta = 2$, $V_T = 26$ mV e $I_0 = 3$ nA: per chiarezza è mostrato il solo ramo corrispondente a valori positivi della d.d.p. applicata, nel solo intervallo $\Delta V = 0 - 0.9$ V.

La scheda Arduino, opportunamente programmata e con l'aggiunta di pochi elementi circuitali esterni, può permettere un'acquisizione *automatizzata* via computer, cioè ottenere un file contenente un numero di punti sperimentali sufficiente per le ulteriori analisi (grafico, best-fit). Gli ingredienti necessari sono:

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

2. registrare il valore della d.d.p. applicata al diodo per ogni gradino della rampa in modo automatico;

3. registrare il valore della corrispondente intensità di corrente che circola nel diodo; poiché le porte analogiche di Arduino consentono 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, sfruttando la legge di Ohm.

II. LA MODALITÀ PWM DI ARDUINO

Abbiamo più volte fatto riferimento ad Arduino come a una scheda I/O. Nell'esperienza che vogliamo svolgere ci sono sicuramente delle grandezze analogiche *in ingresso* (input) a Arduino che vogliamo leggere, cioè digitalizzare e acquisire con il computer. Queste grandezze analogiche sono quelle rappresentative di ΔV e I. Ci piacerebbe molto avere anche una grandezza analogica (variabile in maniera quasi continua e controllata) *in uscita* (output), cioè la d.d.p. che deve essere applicata al diodo.

Purtroppo, dal punto di vista tecnologico ottenere una grandezza analogica in uscita da una scheda I/O è più complicato che non eseguire la digitalizzazione di una grandezza analogica in ingresso. Il microcontroller di Arduino, infatti, non ha la possibilità di creare una d.d.p. di valore determinato a partire da un'istruzione digitale.

Arduino ha invece la possibilità di accendere o spegnere delle porte digitali in uscita, cioè di porle a potenziale nullo (entro l'incertezza) o massimo (entro l'incertezza), dove il massimo, come discusso altrove, si riferisce a un valore legato, nel nostro caso, alla tensione di alimentazione della scheda (tipicamente $V_{ref} \sim 5$ V).

C'è un'interessantissima ulteriore opzione: alcune delle porte digitali di Arduino, quelle marcate sulla scheda con un simbolo tilde, 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 al massimo, come per le altre porte digitali quando vengono poste a "livello alto"). Il duty cycle 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 via software, attraverso un'opportuna istruzione dello sketch. Più propriamente, questa onda quadra si chiama *treno di impulsi*.

A. Implementazione software

Un aspetto molto importante di Arduino è che il treno di impulsi prodotto è gestito "direttamente" dal microcontroller, cioè esso non risulta da cicli inseriti nel programma dello sketch. Dunque le sue caratteristiche non risentono di eventuali latenze del microcontroller e, una volta definite attraverso l'istruzione relativa, rimangono nominalmente costanti nel tempo (entro l'incertezza).

Come controparte, la frequenza del treno di impulsi è determinata internamente e non può essere variata facilmente. Infatti essa è agganciata alla frequenza del contatore, cioè dell'orologio interno al microcontroller. Inoltre, tale frequenza è, purtroppo, piuttosto bassa, in analogia con la generale "lentezza" del microcontroller usato in Arduino. Infatti essa vale nominalmente $f = 976$ Hz per le porte digitali PWM ∼`5` e ∼`6`, e 488 Hz per le porte digitali PWM ∼`3`, ∼`9`, ∼`10`. Spulciando tra le specifiche, si vede come in realtà la frequenza possa essere modificata, in particolare aumentata per multipli di 2, ma di questa possibilità, che implica di ritoccare in qualche modo l'orologio interno del microcontroller, non faremo uso.

L'istruzione software da mettere nello sketch per creare un treno di impulsi con un certo duty cycle è molto semplice e auto-esplicativa. Chiamata `RampPin` la variabile che punta al numero della porta digitale PWM da impiegare, che è la ∼`5` nel nostro caso (e infatti nello sketch comparirà la dichiarazione `const unsigned int RampPin = 5;`), si dovrà inizializzare questa porta come uscita attraverso il comando, da mettere nel `void setup` dello sketch, `pinMode(RampPin, OUTPUT);`. A questo punto, un'onda con duty cycle corrispondente al livello `i` (con `i` intero compreso tra 0 e 255) sarà ottenuta con il semplice comando `analogWrite(RampPin, i);`.

B. Integratore

Abbiamo dunque capito come creare un treno di impulsi, cioè un'onda quadra con un certo duty cycle. Siamo ancora lontani dall'avere una d.d.p. continua variabile attraverso semplici istruzioni software, però abbiamo disponibile tra le nostre conoscenze un metodo che permette di ottenere una tensione *quasi continua* a partire dal treno di impulsi. È infatti evidente che il *valore medio* di un'onda quadra con duty cycle variabile dipende dal duty cycle stesso. Sappiamo poi che l'operazione di media è (a meno di coefficienti) equivalente all'*integrazione temporale* e ci è ben noto come costruire un integratore, per esempio facendo uso di un filtro passa-basso RC.

Naturalmente il circuito RC che ci proponiamo di realizzare dovrà avere una costante tempo RC sufficientemente alta, ovvero una frequenza di taglio $f_T = 1/(2\pi RC)$ sufficientemente bassa, in modo da permettere un'efficace integrazione temporale. In particolare ci aspettiamo che debba essere $f_T << f$; negli esempi ("simulati") riportati nel seguito supporremo di avere $f_T = 10$ Hz, cioè $f/f_T \sim 10^2$.

D'altra parte è ovvio che, aumentando il tempo di integrazione, cioè diminuendo la *banda passante* del sistema, dovremo introdurre degli opportuni tempi di attesa nell'operazione di Arduino, necessari affinché, dopo aver cambiato il duty cycle del treno di impulsi, il segnale in uscita dall'integratore possa raggiungere una nuova condizione stazionaria. Dunque avremo un'acquisizione automatizzata che, però, richiederà un po' di tempo per essere compiuta.

C. Serie di Fourier per il treno di impulsi

È sicuramente interessante "simulare" il comportamento di un integratore al cui ingresso abbiamo un treno di impulsi, ovvero un'onda quadra con un certo duty cycle. Possediamo già lo strumento concettuale che consente di eseguire la simulazione: è sufficiente esprimere il treno

di impulsi in serie di Fourier, cioè conoscerne i coefficienti dell'espansione di Fourier, e quindi applicare alle varie componenti, cioè alle varie armoniche, la funzione di trasferimento [attenuazione $A(f)$ e sfasamento $\Delta\phi$] del passa-basso RC.

Due osservazioni preliminari:

- nella simulazione considereremo una situazione "ideale", per cui trascureremo le resistenze interne, per esempio quella del generatore (la resistenza della porta digitale di Arduino) e di tutto il resto, cioè del circuito contenente il diodo e il collegamento alle porte analogiche di Arduino necessarie per la misura; vedremo nel seguito che queste ipotesi possono non essere del tutto verificate nell'esperimento;

- 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 impulso) 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*:

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

dove $\delta = \tau/T$ rappresenta il duty cycle variabile tra 0 e 1 (τ è 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 sopra), 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, $A(f) = 1/\sqrt{1 + (f/f_T)^2}$, e sfasate di $\Delta\phi = \arctan(-f/f_T)$, esattamente come si fece per l'esercizio sulla "pinna di squalo". La Fig. 2 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. Come specificato sopra, il treno di impulsi ha una frequenza $f = 976$ Hz, mentre la frequenza di taglio dell'integratore è supposta $f_T = 10$ Hz. Per chiarezza, la Fig. 3 mostra

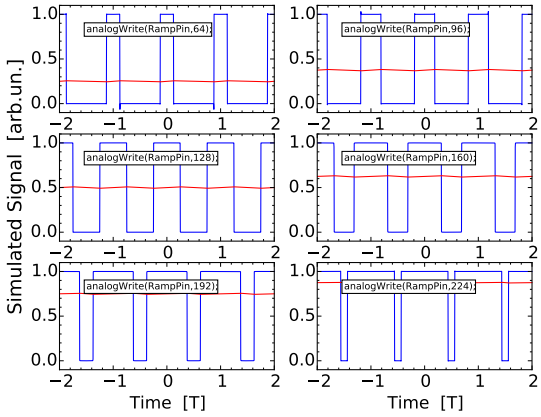

Figura 2. Esempi di simulazione di treni di impulsi con duty cycle variabile (curve blu) e di uscita dall'integratore (curve rosse). I vari grafici si riferiscono a diverse scelte del duty cycle: in legenda è riportata l'istruzione software usata nello sketch di Arduino. Per il calcolo si è supposto un integratore RC con frequenza di taglio $f_T = 10$ Hz; il periodo T con cui si misurano i tempi vale $T = 1/f$, con $f = 976$ Hz.

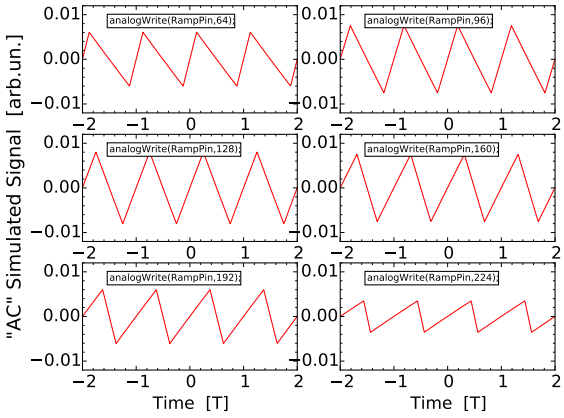

Figura 3. Analogo di Fig. 2 per le sole tracce rappresentative del segnale in uscita dall'integratore: per questa figura, a tale segnale è stato sottratto il valore medio nel tempo, simulando, in pratica, un'osservazione con oscilloscopio "accoppiato in AC".

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 "accoppiato in AC", che permette di apprezzare la discrepanza rispetto a un segnale idealmente costante.

Il risultato simulato 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 una sorta di

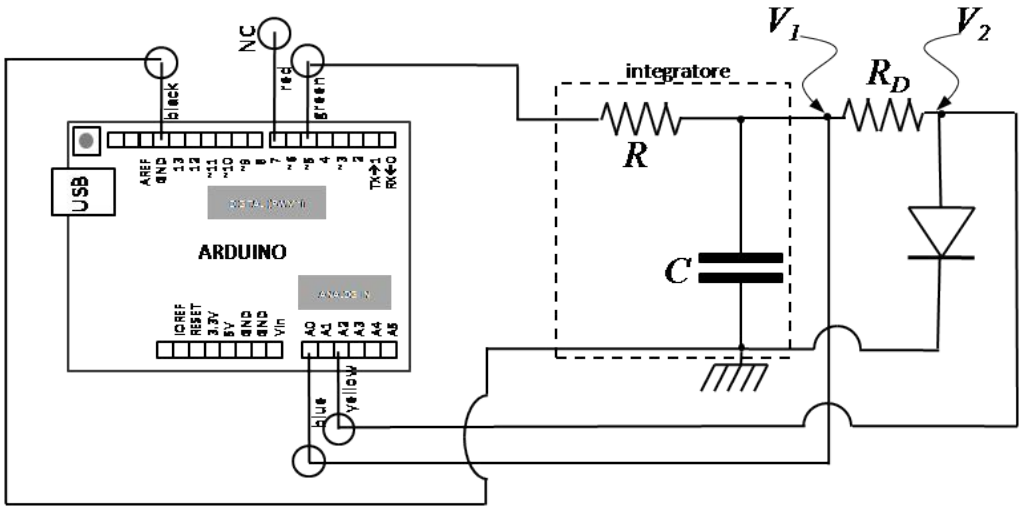

Figura 4. Schema circuitale dell'esperienza con indicate le quattro connessioni da effettuare alla scheda Arduino (NC significa non collegato).

ripple (piccola modulazione) che non riesce a essere integrato in maniera completa e che, come risulta chiaro da Fig. 3, è particolarmente rilevante per i valori intermedi del duty cycle. In linea di principio, l'entità del ripple, che è generalmente "piccola" rispetto al livello quasi-continuo, potrebbe renderne la presenza ininfluente per gli scopi del nostro esperimento: su questo argomento torneremo in seguito.

III. MISURA DELLA CORRENTE

Dobbiamo a questo punto definire il metodo che consente di misurare l'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 la legge di Ohm 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 terra, la misura ΔV_{RD} deve essere eseguita per differenza, $\Delta V_{RD} = V_1 - V_2$, tra le d.d.p. (riferite a terra) che si trovano all'"ingresso" (V_1) e all'"uscita" (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 di trovare tutte le grandezze necessarie per ricostruire la curva I-V.

Oltre alla determinazione di I, la resistenza R_D ha anche un altro ruolo. Quando si trova in conduzione, cioè per ΔV superiore a un valore di *soglia* V_{thr} (tipicamente compreso nell'intervallo $0.45-0.65$ V, per diodi al silicio), il diodo sostiene il passaggio di correnti potenzialmente molto intense [si veda Fig. 1(b)], incompatibili con le possibilità di Arduino. La resistenza in serie al diodo limita la corrente massima richiesta.

Supponendo di porre tale limite sotto alla decina di mA (le porte digitali di Arduino sono progettate per fornire un massimo di 20 mA, secondo datasheet, ma è meglio tenersi al di sotto), e tenendo conto che la d.d.p. erogata è al massimo attorno a 5 V, R_D deve essere dimensionata nell'ordine delle centinaia di ohm, o superiore. Ponendo, ad esempio, $R_D = 680$ ohm, si ha una richiesta di corrente massima inferiore a 8 mA, dove per semplicità si sono trascurate la resistenza interna effettiva del diodo e la resistenza R dell'integratore.

In definitiva, lo schema circuitale adottato è quello di Fig. 4: 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 delle d.d.p. V_1 e V_2.

IV. 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.

Lo script sfrutta tutte le particolarità già discusse a proposito dell'esperienza sulla carica/scarica del condensatore; esso si trova in rete con il nome di `diodo2016.py` e, ovviamente, è già caricato sui computer di laboratorio. Anche in questa esperienza lo script invia un'informazio-

ne ad Arduino attraverso un carattere (byte) che rappresenta il ritardo tra una coppia di misure e la successiva, necessario per permettere all'integratore di raggiungere condizioni stazionarie. Il carattere, che può essere impostato tra 1 e 9, rappresenta il ritardo in unità di 10 ms, e per default è regolato a 50 ms. Poiché nello sketch di Arduino ci sono due istruzioni successive di ritardo, di fatto, con le impostazioni di default, occorrono $50 \times 2 = 100$ ms per acquisire un singolo punto della curva I-V. Tenendo conto di ulteriori ritardi (necessari per la scarica del condensatore e per evitare impallamenti nella comunicazione seriale), l'acquisizione della curva richiede un tempo stimabile in diverse decine di secondi.

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'5')#intervallo (ritardo) in unita' di 10 ms <<<< questo si puo' cambiare (default 50 ms)
 print('Start!\n') # scrive sulla console (terminale)
Directory='../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)
```

Anche lo sketch di Arduino, che si può trovare in rete con il nome `diodo2016.ino`, è molto semplice e pressoché auto-esplicativo. Poiché in questo esperimento non è necessario spingere al massimo possibile il rate di campionamento, non vengono incluse le istruzioni necessarie all'"overclock" del microcontroller (tra l'altro, esse sono incompatibili con le altre operazioni richieste ad Arduino in questa esperienza).

Il testo 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 Ã¨ 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 0% a 256-->duty cycle 100%
      {
       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
  }
```

V. CALIBRAZIONE E INCERTEZZE

Lo scopo dell'esperienza è quello di costruire una curva I-V da cui, oltre all'andamento, sia anche possibile dedurre i valori caratteristici in gioco, espressi nelle debite unità fisiche. Pertanto è opportuno che le grandezze digitalizzate ([digit]) siano convertite in unità [V].

A questo scopo è possibile impiegare la procedura di calibrazione "alternativa", che consiste nella misura di V_{ref} e nella determinazione del fattore di calibrazione $\xi = V_{ref}/1023$ (le sue unità di misura sono [V/digit], o, ancora meglio, [mV/digit]). Come già discusso in altra sede, V_{ref} è atteso coincidere, almeno approssimativamente, con il massimo valore della tensione fornita dalle uscite digitali di Arduino. Poiché al termine dello sketch la porta ∼5 si trova al livello "alto" (e ci rimane finché l'acquisizione non viene fatta ripartire), per conoscere V_{ref} 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 operazione di misura deve essere condotta *a circuito aperto*, cioè scollegando quanto si trova di seguito alla porta (almeno il diodo). Infatti, come già affermato, nel diodo può circolare corrente di intensità tutt'altro che trascurabile, che quindi provoca una caduta di tensione sulla serie di resistori R e R_D. Nell'eseguire l'esperimento può essere istruttivo verificare qual è la differenza nella lettura di V_2 quando il diodo viene collegato o scollegato (si ricorda, en passant, che scollegare un componente a due fili significa scollegare uno o tutti e due i fili, e basta). Dunque dalla singola misura è possibile ricavare il fattore di calibrazione e l'incertezza ad esso associata, dovuta all'errore (calibrazione e lettura, sommate in quadratura) della misura di V_{ref} con il multimetro.

Come sappiamo da precedenti esperienze, la calibrazione "alternativa" trascura completamente la non linearità del digitalizzatore, in particolare l'eventuale presenza di un offset. Per questa esperienza, e a meno di non eseguire una calibrazione "completa" della scheda Arduino (che richiede tempo e sforzi), possiamo disinteressarci di questo aspetto, tenendo conto che l'Eq. 1, che useremo per interpretare i dati, contiene già, di fatto, un termine costante. Pertanto non avrebbe molto senso aggiungere un ulteriore offset all'equazione stessa, con lo scopo di migliorare l'accordo tra dati e best-fit.

Per determinare l'incertezza delle misure possiamo usare il consueto approccio per cui attribuiamo un errore *convenzionale* (di lettura) di ±1 digit alla lettura digi-

talizzata. Poiché in questa esperienza convertiamo la lettura digitalizzata in unità fisiche, dovremo sommare (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 $\Delta V = V_2$ (chiameremo il suo errore δV).

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

$$I = \frac{V_1 - V_2}{R_D} \ , \tag{3}$$

dove tutte le grandezze sono già state definite in precedenza. Per stimarne l'incertezza δI dovremo fare debito uso delle regole di propagazione dell'errore. Poiché al numeratore di Eq. 3 compare una differenza tra grandezze affette dalla stessa incertezza di calibrazione, dovremo porre attenzione a non sovrastimare l'errore della differenza (questo argomento dovrebbe essere nelle vostre conoscenze di analisi dati).

VI. FUNZIONAMENTO EFFETTIVO DELL'INTEGRATORE

Ora che abbiamo specificato più nel dettaglio come intendiamo effettuare le misure, possiamo tornare all'analisi del funzionamento dell'integratore. In buona sostanza, esso si comporta come il *livellatore* di un alimentatore basato su trasformatore e raddrizzatore a semionda. Nell'esaminare questa tipologia di circuiti si ottiene che il ripple può dipendere dal "carico": in particolare, esso tende ad aumentare quando il carico (supposto resistivo) diminuisce, cioè quando viene richiesta una maggiore corrente.

Nell'analisi di Sez. II B abbiamo di fatto *trascurato* il carico, e tutti gli eventuali problemi connessi: abbiamo infatti supposto di trascurare tutte le resistenze al di fuori di quella inserita nell'integratore. Invece, nella realtà del nostro circuito al "livellatore", cioè al condensatore C, viene richiesta corrente per tutta il tempo che intercorre tra due impulsi successivi del treno. La corrente fluisce attraverso R_D e quindi nel diodo, quando la d.d.p. ai suoi capi è tale da portarlo in conduzione; ovviamente, in seguito a questo processo il condensatore perde parte della carica che ha accumulato in precedenza, quando l'impulso si trovava a livello "alto", per cui il meccanismo di livellamento tende a perdere di efficacia.

L'effetto del carico potrebbe essere analizzato sulla base di un opportuno modello (questo potrebbe essere tentato, ad esempio, in una relazione semestrale). Qui ci limitiamo a notare che, nelle condizioni effettive di funzionamento, il ripple della d.d.p. applicata al diodo può rivelarsi sensibilmente superiore rispetto a quanto predetto in Fig. 3: siete invitati a verificare sperimentalmente la sua entità, per esempio monitorando con l'oscilloscopio il segnale V_2 *durante* la misura.

La presenza del ripple, combinata con la circostanza che le digitalizzazioni di V_1 e V_2 non sono simultanee fra loro (fra di esse intercorre un tempo almeno pari a quello necessario per la singola digitalizzazione), può produrre delle "fluttuazioni" nella misura di V_2 e di I. È evidente che tale problema potrebbe essere mitigato aumentando R_D. Tuttavia, così facendo si determinerebbe una rilevante caduta di potenziale su questo componente, che finirebbe per limitare il range di valori di ΔV esplorati (tutto questo diverrà più chiaro quando saranno presentati i risultati esempio). D'altro canto, come già sottolineato, questo componente serve anche per limitare la richiesta di corrente alle porte di Arduino, per cui R_D non può nemmeno essere scelta troppo bassa (un limite minimo ragionevole può essere il centinaio di ohm).

Una possibilità alternativa consiste nello "spianare" al meglio il treno di impulsi, cioè scegliere una frequenza di taglio f_T particolarmente bassa, ovvero, rifrasando, usare R e C di valore alto. Poiché R è di fatto in serie a R_D e al diodo, conviene che anche questa resistenza non abbia valore troppo alto (si consiglia R dell'ordine delle poche centinaia di ohm). Quindi l'opzione più opportuna può essere quella di impiegare un condensatore C di capacità elevata.

Per mantenere dimensioni fisiche accettabili in condensatori di elevata capacità, è in genere necessario servirsi di componenti che sfruttano tecnologie diverse rispetto a quella dei dispositivi a poliestere (detti anche "a carta") che siamo soliti usare. In particolare, nell'esperienza potrebbe essere consigliabile servirsi di *condensatori elettrolitici*, che, a parità di dimensioni fisiche, consentono capacità nettamente superiori.

A causa delle loro caratteristiche costruttive (che siete fortemente invitati a studiare), i condensatori elettrolitici sono componenti a due fili *polarizzati*, cioè il loro collegamento deve essere effettuato solo in un senso. Il "polo" negativo del componente, normalmente collegato alla carcassa metallica del suo involucro esterno, deve essere collegato alla linea di massa, o terra.

VII. ESEMPIO DI MISURA E COMMENTI

L'esperimento esempio qui riportato è stato compiuto usando un integratore realizzato con $R = 330$ ohm (nominali, tolleranza 5%) e $C = 100$ μF (nominali, tolleranza 20%), capacità ottenuta proprio con un condensatore elettrolitico. La frequenza di taglio nominale è quindi $f_T = 4.8$ Hz, con tolleranza dominata da quella sulla capacità. La resistenza in serie al diodo è stata misurata, ottenendo $R_D = (682 \pm 6)$ ohm. La tensione di riferimento, misurata a circuito aperto con la procedura descritta in precedenza, è risultata $V_{ref} = (5.02 \pm 0.03)$ V. Il fattore di calibrazione è quindi $\xi = (4.91 \pm 0.03)$ mV/digit. La somma delle resistenze $R + R_D$ è relativamente grande, oltre 1 kohm, e quindi la richiesta di corrente alla porta di Arduino è relativamente piccola, limitata nominalmente a circa 5 mA. Come intervallo di tempo tra un'acquisizio-

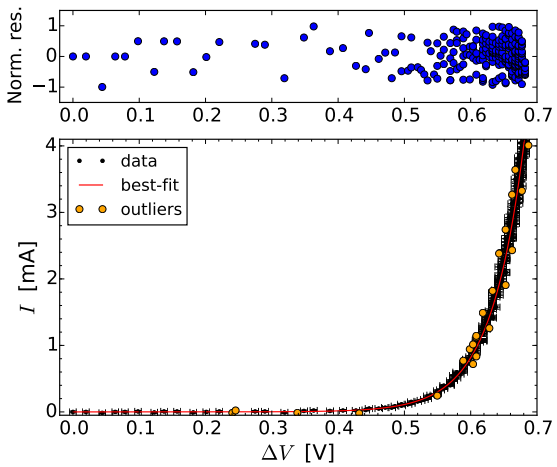

Figura 5. Esempio di misure ottenute con la scelta dei parametri e dei componenti circuitali indicati nel testo; la linea rossa continua rappresenta il best-fit condotto secondo l'Eq. 1 (commenti e risultati nel testo), i marker arancioni individuano i potenziali outliers, secondo quanto specificato nel testo. Il pannello superiore riporta il grafico dei residui normalizzati (depurati dagli outliers).

ne e la successiva è stato mantenuto il valore di default di 50 ms (la scelta non è critica).

La Fig. 5 riporta un esempio delle misure: nella determinazione delle barre di errore sono state seguite le istruzioni specificate in precedenza. La figura mostra anche, con una linea continua rossa, il risultato di un best-fit dei dati secondo l'Eq. 1, eseguito lasciando come parametri liberi del fit I_0 e ηV_T, il corrispondente grafico dei residui normalizzati (pannello superiore), e i potenziali *outliers*, identificati come sarà discusso nel seguito e indicati con pallini arancioni sovrapposti al grafico dei dati sperimentali.

Facciamo subito alcune osservazioni preliminari sui dati: in primo luogo l'andamento qualitativo è quello atteso, cioè, grazie anche alla scala lineare di rappresentazione (potrebbe essere opportuno impiegare una rappresentazione semi-logaritmica, che però darebbe un risultato visualmente meno immediato), si intuisce una crescita pressoché esponenziale della corrente I in funzione di ΔV, in particolare per $\Delta V \gtrsim 0.5$ V.

Accanto a questa osservazione in linea con le aspettative, ci sono alcuni aspetti apparentemente meno ovvi. Per esempio, si nota che la spaziatura dei valori sull'asse orizzontale, che in principio ci aspetteremmo sempre uguale (approssimativamente pari $V_{ref}/256 \sim 20$ mV, essendo 256 i livelli su cui viene aggiustato il duty cycle del treno di impulsi), non è affatto uniforme. Per bassi valori di ΔV possiamo attribuire la circostanza all'incertezza con la quale viene generata la d.d.p. e al tempo di risposta finito dell'integratore, ma certamente l'addensamento dei punti di misura a partire da $\Delta V \sim 0.5$ V richiede una spiegazione. Inoltre è anche evidente che il range spazzato nella misura è molto inferiore rispetto alle aspettative: infatti il ΔV massimo esplorato non raggiunge 0.7 V, mentre invece ci aspetteremmo, sulla base di un ragionamento molto naïf, valori prossimi a V_{ref}.

Il motivo può essere facilmente intuito ricordando le caratteristiche di funzionamento del diodo bipolare a giunzione e osservando come, per $\Delta V \gtrsim 0.6$ V, l'intensità di corrente I salga a valori dell'ordine di alcuni mA. In queste condizioni la caduta di potenziale sulla serie $R+R_D$ è tutt'altro che trascurabile (essa vale oltre 4 V per $I = 4$ mA), per cui la d.d.p. ΔV effettivamente applicata al diodo si riduce considerevolmente. Rifrasando, potremmo affermare che il generatore che alimenta il diodo ha una resistenza interna ("a la Thévenin") tutt'altro che trascurabile, stimabile proprio come la somma $R + R_D$ (se si suppone praticamente nulla la resistenza interna della porta digitale di Arduino).

In buona sostanza, come evidenziato anche dall'andamento della curva caratteristica I-V, vedi Fig. 1(b), attorno o sopra il valore di soglia V_{thr} (ricordiamo che esso è tipicamente compreso tra 0.45 e 0.65 V, per un ordinario diodo al silicio), la d.d.p. ΔV tende proprio a stabilizzarsi attorno a tale valore, indipendentemente (o "poco dipendentemente") dalla corrente. Questa affermazione è naturalmente da prendere con le molle, se non altro perché V_{thr} non è definito dal punto di vista matematico. La definizione pratica che ne è talvolta data, come la d.d.p. applicata al diodo tale che in esso fluisca una corrente di intensità pari a 1/100 dell'intensità massima tollerata (300 mA, supponendo che il diodo impiegato sia il modello 1N914 al cui datasheet si fa riferimento), conduce a $V_{thr,prat} \simeq 0.66$ V, dove il pedice *prat* indica che si tratta di un valore riferito a una definizione pratica. Esso è leggermente al di fuori del range generalmente accettato, e la (piccola) discrepanza può essere interpretata come dovuta alla caduta di potenziale sulle resistenze che si trovano in serie alla giunzione, cioè i pezzi di semiconduttore drogato p e n, gli elettrodi, i conduttori di collegamento, che non sono considerate quando si descrive il modello di un diodo a giunzione "ideale"; la presenza di queste resistenze non è ovviamente considerata nell'equazione modello (Eq. 1). Notate in ogni caso che esistono anche altre definizioni pratiche di soglia, che possono condurre a valori leggermente diversi.

Allora, l'infittirsi dei punti sperimentali rispecchia la circostanza che, a partire da un certo valore del duty cycle, V_2, cioè ΔV in Fig. 5, tende ad aumentare di pochissimo da una misura alla seguente. Poiché, inoltre, I è costruita a partire dalla differenza $V_1 - V_2$, anche a causa del ripple residuo in uscita dall'integratore può verificarsi che i dati acquisiti non siano ordinati monotonamente. Tutto questo dà luogo a una sorta di "fluttuazioni" dei dati graficati.

Chiariti questi aspetti, torniamo ad occuparci del best-fit. Esso è stato condotto tenendo in conto anche l'incertezza δV sulla grandezza riportata in asse orizzontale, che l'estensione delle barre di errore suggerisce non trascurabile. Allo scopo si è determinato l'"errore equivalente" at-

traverso propagazione dell'errore applicata all'equazione modello (Eq. 1), che è poi stato sommato in quadratura con δI (a sua volta determinato con attenzione, secondo quanto specificato in precedenza). Naturalmente la procedura ha richiesto inizialmente di individuare i parametri del best-fit (I_0 e ηV_T) usando solo δI, esattamente come abbiamo già fatto in altre circostanze e come tutti ben sapete da lungo tempo.

I risultati del best-fit, per il quale è stata usata l'opzione `absolute_sigma = False` (le incertezze hanno sicuramente un carattere non prevalentemente statistico, vista la presenza di errori sistematici di calibrazione), sono

$$I_0 = (5.2 \pm 0.2)\ \text{nA} \tag{4}$$

$$\eta V_T = (50.1 \pm 0.3)\ \text{mV} \tag{5}$$

$$\chi^2/\text{ndof} = 102/254 \tag{6}$$

$$\text{Norm. cov.} = 0.997\ . \tag{7}$$

L'elevato valore della covarianza normalizzata è conseguenza del modello, Eq. 1: in esso, se si considerano punti sperimentali tali che il termine esponenziale prevalga sull'unità (praticamente tutti quelli acquisiti per ΔV superiore a poche decine di mV), c'è una correlazione pressoché totale tra I_0, che sta a moltiplicare l'esponenziale, e ηV_T, che sta a dividere nell'argomento dell'esponenziale.

In termini assoluti, i valori trovati per I_0 e per ηV_T sono grossolanamente compatibili con le aspettative. In particolare, facendo riferimento al datasheet del diodo 1N914 (un "paradigma" dei diodi al silicio di piccola potenza), si osserva come valori tipici per I_0 siano compresi tra circa 4 e circa 6 nA (la misura del datasheet è in realtà riferita a una polarizzazione inversa $\Delta V = -1$ V) e, in ogni caso, correnti di saturazione inversa di alcuni nA, e fino a oltre 10 nA, sono ampiamente ragionevoli. Tenendo poi conto che η, per un ordinario diodo al silicio, è generalmente compreso tra 1.5 e 2, anche il valore di ηV_T è perfettamente in linea con le aspettative, considerando che la misura è eseguita a temperatura ambiente, almeno finché si considera trascurabile il riscaldamento dovuto a effetto Joule del diodo (un'evoluzione dell'esperimento potrebbe comprendere misure condotte a diverse temperature).

Infine, nonostante i dati presentati in questo esempio non richiedano particolari "attenzioni" di tipo cosmetico, è stata applicata una procedura volta ad identificare eventuali *outliers*, che qui definiamo, in accordo con una precedente esperienza, come dati che distano dalla previsione del best-fit per oltre una certa soglia. Questa procedura, che ha sempre un carattere *arbitrario*, può essere utile per isolare dati affetti da errori di tipo prevalentemente sistematico, legati ad aspetti specifici del funzionamento di Arduino o dell'esperimento. Come in precedenti esperienze, potrebbe infatti verificarsi che la digitalizzazione di segnali variabili nel tempo avvenga in maniera imperfetta; inoltre le "fluttuazioni" dei dati, le cui origini sono state accennate in precedenza, e la possibilità di registrare segnali spuri, legati ad esempio a spikes (impulsi) che circolano all'interno del microcontroller per cause o stocastiche, o dovute a qualche operazione transiente a carico del microcontroller stesso (accensione o spegnimento di porte, o altro), possono anche dare luogo ad artefatti potenzialmente rilevanti.

In questo esempio, è stato arbitrariamente scelto di considerare potenziali outliers i dati che si discostavano per oltre una barra di errore (comprensivo dell'"errore equivalente") dalle previsioni del best-fit: i dati così identificati, in numero totale di 23, sono marcati con un pallino arancione in Fig. 5. Come si può facilmente intuire, in questo esempio la rimozione degli outlier dal set di dati considerato nel best-fit non produce modifiche sostanziali ai risultati riportati in precedenza, a parte un'ovvia diminuzione del χ^2 (che scende a 52 per ndof = 231).