Curva caratteristica del diodo con Arduino

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

(Dated: version 6 - Lara Palla e Francesco Fuso, 14 dicembre 2015)

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 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 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

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

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 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, operazione non semplice dal punto di vista pratico.

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" questa 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 (di input/output). 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 quelle che permettono di determinare I. Ci piacerebbe molto avere anche una grandezza analogica variabile *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 (circa 5 V).

C'è un'interessantissima ulteriore opzione: alcune delle porte digitali di Arduino, quelle marcate sulla scheda

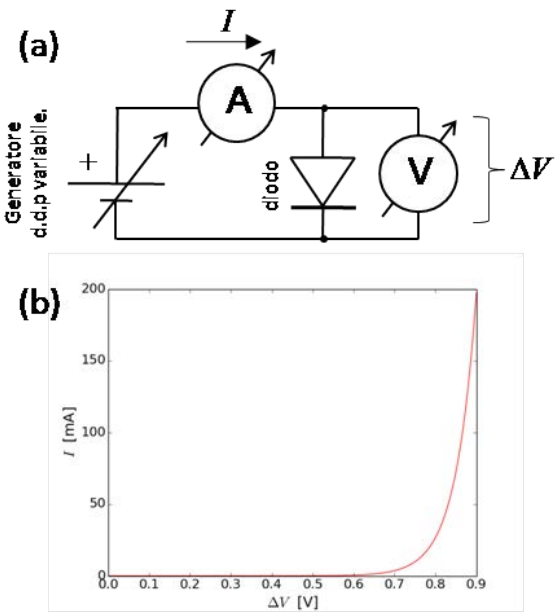

Figura 1. Schema concettuale di un'ipotetica esperienza di ricostruzione della curva caratteristica $I - V$ di un diodo (a) e curva calcolata secondo Eq. 1, supponendo $\eta = 2$, $V_T = 26$ mV e $I_0 = 3$ nA: per chiarezza è mostrato il solo ramo corrispondente a valori positivi della d.d.p. applicata,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

con un simbolo tilde, possono operare in modalità *Pulse Width Modulation* (PWM). Questo significa che in uscita si trova 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 accese). 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 con duty cycle variabile 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 con Arduino un treno di impulsi, cioè un'onda quadra con un certo duty cycle controllato via software. Siamo ancora lontani dall'avere una d.d.p. continua variabile, però abbiamo disponibile tra le nostre conoscenze un metodo che permette di ottenere una tensione *quasi continua* a partire dal treno di impulsi. Osserviamo che l'onda quadra prodotta da Arduino è certamente *non* alternata: essa infatti oscilla tra il valore minimo (zero, o circa zero) e quello massimo (circa 5 V), senza diventare mai negativa. Il *valore medio* (nel tempo) di un'onda quadra fatta in questo modo e con duty cycle variabile è non nullo e 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, da permettere un'efficace integrazione temporale. In particolare ci aspettiamo che debba essere $f_T << f$. In generale, un integratore si comporta bene se la sua frequenza di taglio è circa un centinaio di volte minore della frequenza del segnale che deve essere integrato, cioè se $f/f_T \sim 10^2$, che nel nostro caso corrisponde a $f_T \sim 10$ Hz. Vedremo nel seguito che questa scelta può non essere sufficiente per lo specifico esperimento che vogliamo eseguire, circostanza che ci porterà a scegliere $f_T \sim 1$ Hz.

Inoltre 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 dal-

l'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, affermazione che, come illustreremo nel seguito, è tutt'altro che verificata sperimentalmente;

- come già affermato, il treno di impulsi in ingresso all'integratore non è alternato, cioè la sua media temporale è diversa da zero (sempre positiva, o nulla per duty cycle nullo).

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*:

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

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 già affermato, 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 il guadagno, o attenuazione, 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 (si riporta per chiarezza proprio l'istruzione da usare nello 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.

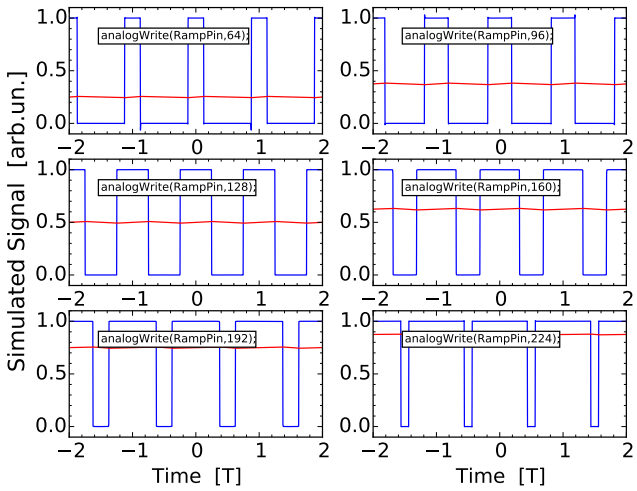

Figura 2. Esempi di simulazione di treni di impulsi con duty cycle variabile (curve blu) e di uscita dall'integratore (curve rosse). I vari grafici si riferiscono a diverse scelte del duty cycle: in legenda sono riportate le istruzioni software che dovrebbero essere usate nello sketch di Arduino. Per 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.

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: esso dà luogo a una sorta di *ripple* (piccola modulazione) che non riesce a essere integrato temporalmente in maniera completa. La presenza del ripple è, in linea di principio, trascurabile per la nostra esperienza se si suppone che *tutte* le misure che servono per generare la curva caratteristica siano acquisite simultaneamente, circostanza non completamente verificata (tra due misure di Arduino intercorre sempre un intervallo di tempo non nullo). Pertanto l'entità effettiva del ripple può essere rilevante nel determinare eventuali fluttuazioni delle misure.

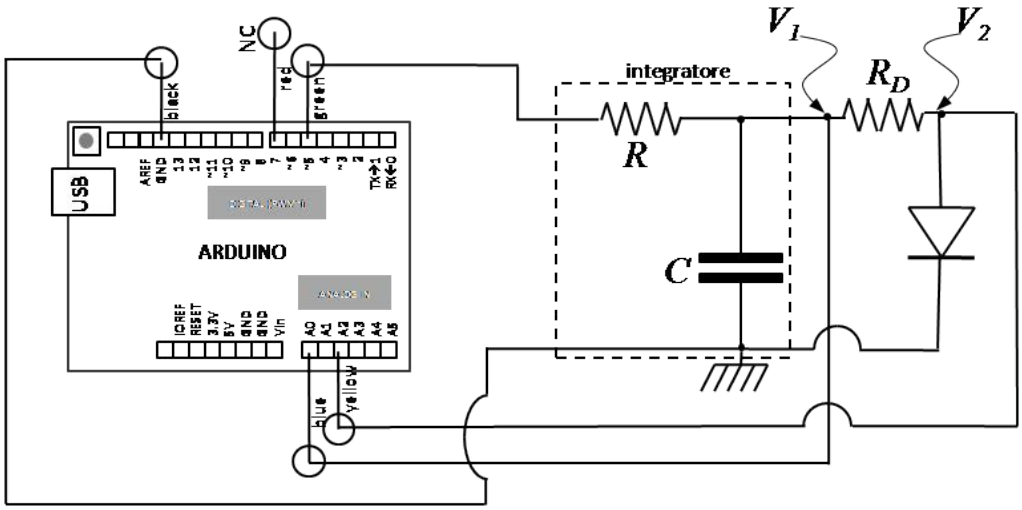

Figura 3. Schema circuitale dell'esperienza con indicate le quattro connessioni da effettuare alla scheda Arduino (NC significa non collegato). Nell'esperienza pratica è possibile impiegare come condensatore C un dispositivo *elettrolitico*, che va collegato in modo che il suo elettrodo negativo sia connesso alla linea di massa, o terra.

III. LIMITAZIONE DELLA CORRENTE E SUA MISURA

Come si può facilmente intuire dall'Eq. 1 e dalla sua rappresentazione di Fig. 1(b), il diodo usato in polarizzazione diretta è in grado di far passare una corrente molto intensa, ovvero la sua "resistenza effettiva" (rapporto tra ΔV e I) può diventare trascurabile. Ci sono diversi motivi che suggeriscono di *limitare* il valore massimo ammissibile per questa intensità di corrente: i principali sono che il diodo, al di sopra di un certo valore di corrente (~ 100 mA, per i diodi in uso in laboratorio) si distrugge per surriscaldamento; inoltre, e più importante, Arduino non è progettato per generare correnti superiori a poche decine, di mA (nominalmente 40 mA).

Per limitare la corrente massima che può fluire nel diodo è sufficiente inserire un resistore R_D in serie al diodo stesso. Supponendo trascurabile la resistenza interna del diodo nelle condizioni "peggiori" che stiamo esaminando, la corrente massima sarà limitata al valore $\Delta V/R_D$. Se vogliamo che questa corrente non superi, poniamo precauzionalmente, alcuni mA, e se teniamo conto che al massimo ΔV può valere circa 5 V nel nostro esperimento, possiamo dimensionare R_D nell'ordine dei kohm.

Tornando allo schema di Fig. 1(a), sappiamo che dobbiamo ancora occuparci della misura dell'intensità di corrente I, che va necessariamente "convertita" in una d.d.p. affinché sia digeribile per Arduino. Possiamo allo scopo sfruttare ancora R_D, cioè il resistore in serie al diodo: infatti per la legge di Ohm si ha semplicemente $I = \Delta V_{RD}/R_D$, dove ΔV_{RD} indica la caduta di potenziale sulla resistenza R_D.

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 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.

In definitiva, lo schema circuitale adottato è quello di Fig. 3: in esso si prevede l'impiego della porta digitale PWM $\sim$`5` (boccola verde) e le porte analogiche `A0` e `A2` (boccole blu e gialla), rispettivamente per la misura delle d.d.p. V_1 e V_2.

IV. FUNZIONAMENTO EFFETTIVO DELL'INTEGRATORE

Torniamo ora all'analisi del funzionamento dell'integratore. Il suo scopo è fondamentalmente quello di *livellare* un segnale oscillante, il treno di impulsi, allo scopo di ottenere una d.d.p. (quasi) continua. Questa d.d.p. viene poi applicata alla parte di circuito costituita da R_D e dal diodo, che in un certo senso costituisce il "carico" del nostro integratore. Nell'analisi svolta sopra abbiamo completamente trascurato la presenza del carico, operazione non necessariamente giustificata. Dal punto di vista sperimentale, il ruolo della parte di circuito comprendente R_D e il diodo può essere facilmente messa in luce. Infatti è sufficiente osservare con l'oscilloscopio il segnale in uscita dall'integratore, marcato con V_1 in Fig. 3, per rendersi conto se e di quanto esso si modifichi quando il diodo è collegato, o meno, al circuito.

Per stimare gli "effetti del carico", cioè di R_D e del

diodo, sul comportamento dell'integratore si possono seguire due strade. Una prevede di riscrivere le equazioni del circuito nel dominio delle frequenze tenendo conto di tutti i componenti coinvolti, e di ottenere di conseguenza le espressioni corrette per $A(f)$ e $\Delta\phi$: questa strada è un po' complicata e non verrà seguita in questa nota (potete affrontarla per esercizio).

L'altra possibilità, più semplice, consiste nel valutare qualitativamente il comportamento del circuito nel dominio dei tempi. Supponiamo di essere nelle condizioni in cui il diodo si trova "in conduzione", cioè la sua resistenza effettiva è trascurabile. In queste condizioni il condensatore C vede in parallelo a se stesso la resistenza R_D. Per effetto del treno di impulsi creato da Arduino, possiamo distinguere due diverse situazioni che si verificano periodicamente nel tempo: infatti il segnale all'ingresso dell'integratore può trovarsi a "livello alto" (circa 5 V) o a "livello basso" (circa zero). Nel primo caso il condensatore viene caricato, nel secondo esso si scarica. Se il diodo è in conduzione, la scarica avviene proprio attraverso R_D. Questo dimostra che la parte di circuito contenente R_D e il diodo non può essere trascurata a priori.

Per rendere costante il più possibile la d.d.p. applicata al diodo durante le misure è conveniente scegliere una capacità C piuttosto elevata. Nell'esempio che sarà discusso in seguito si è scelto $C = 220\ \mu$F (nominali). Valori così alti di capacità sono generalmente ottenibili con condensatori *elettrolitici*, che sfruttano un materiale dielettrico in grado di esaltare la densità di carica elettrica sulle armature. Questi condensatori hanno la particolarità di essere "polarizzati", cioè il collegamento dei loro due elettrodi non è invariante rispetto alle polarità del circuito. In sostanza, facendo riferimento allo schema di Fig. 3, occorre che l'elettrodo negativo sia collegato alla linea di massa, o terra, e quello positivo sia collegato a R e R_D (e anche V_1). In caso contrario la capacità offerta risulta minore e, soprattutto, si possono verificare danneggiamenti al condensatore e al circuito.

V. SCRIPT DI PYTHON E SKETCH DI ARDUINO

Come di consueto quando si usa Arduino, anche in questo caso ci serviamo di uno script di Python per gestire la partenza delle operazioni della scheda e per consentire il trasferimento dei dati da questo al computer tramite porta seriale USB.

Lo script sfrutta tutte le particolarità già discusse a proposito di altre esperienze; esso si trova in rete con il nome di `diodo_v1.py` e, ovviamente, sarà già caricato sui computer di laboratorio. Anche in questo caso lo script invia un'informazione 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 20 ms. A differenza di altre esperienze, qui la scelta del ritardo *non* è critica, ma siete ovviamente invitati a verificarlo facendo diverse prove.

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
time.sleep(2)   # aspetta due secondi
ard.write(b'2')#intervallo (ritardo) in unita' di 10 ms <<<< modificabile
print('Start!\n') # scrive sulla console (terminale)
Directory='../dati_arduino/'   # nome directory dove salvare i file dati
FileName=(Directory+'dati.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 `diod.ino`, è molto semplice e pressoché autoesplicativo, oltre che concettualmente e strutturalmente molto simile a quello usato per altre esperienze. Il testo

è il seguente:

```
const unsigned int RampPin = 5;  //pin 5 uscita pwm per generare la rampa.
                                 // Questo pin viene collegato all'integratore
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 è a uno parte l'acquisizione
    delay(2000); // attende 2s
    for(i=0;i<256;i++)  //il valore che definisce il duty cycle dell'onda quadra è scrivibile su 8 bit
                        //cioè assume valori da 0 a 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
  }
```

Notate che l'intervallo di ritardo `delay_ms` viene applicato due volte all'interno del ciclo di misura, in modo da minimizzare la presenza di disturbi all'atto della misura. Di conseguenza l'esecuzione completa del ciclo richiede un tempo che, per default, è superiore a una decina di secondi.

VI. SENSIBILITÀ E INCERTEZZE

Come per ogni esperimento in cui si adotta una procedura automatizzata di presa dati, è necessario discutere alcuni aspetti relativi alla sensibilità della misura e all'incertezza da attribuire ai dati.

Innanzitutto notiamo che la d.d.p. prodotta da Arduino seguito dall'integratore è variabile in modo necessariamente discreto, dato che discreta (digitale) è l'impostazione del duty cycle del treno di impulsi. Supponendo di avere un valore massimo di tensione attorno a 5 V, e tenendo conto che l'impostazione avviene con una dinamica di 8 bits, cioè sono possibili $2^8 = 256$ livelli diversi, il "quanto" della discretizzazione vale 5 V/256 $\simeq$ 20 mV.

Osserviamo poi che la ricostruzione dell'intensità di corrente I che fluisce nel diodo richiede di eseguire un calcolo in cui compare una differenza:

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

dove tutte le grandezze sono già state definite in precedenza. Dato che la dinamica delle porte analogiche è 10 bits, cioè sono possibili $2^{10} = 1024$ livelli, la lettura di V_1 e V_2 ha una sensibilità, sempre supponendo che la tensione di riferimento effettiva sia (circa) 5 V, di 5 V/1023 $\sim$ 5 mV. Ponendo per esempio $R_D = 3.3$ kohm nominali, la sensibilità nominale nella misura di I è approssimativamente 5 mV/3.3 kohm $\simeq 2$ μA.

Discutiamo ora le incertezze delle nostre misure. Al solito, esse sono in generale causate da motivi statistici, che convenzionalmente danno luogo all'incertezza ± 1 unità arbitrarie di digitalizzazione, che chiamiamo anche digit, e da errori sistematici, per esempio dovuti alla calibrazione. A rigore, in questa esperienza tutte le misure provengono dalla digitalizzazione dei segnali alle porte `A0` e `A2`. Esse appartengono alla stessa scheda Arduino, per cui dovrebbero (o, meglio, potrebbero, non conosciamo i dettagli circuitali del microcontroller di Arduino in maniera sufficientemente dettagliata per esser certi di questa affermazione) essere affette in ugual modo dagli errori sistematici. Dunque in linea di principio potremmo essere autorizzati a trascurare l'incertezza di calibrazione, che sappiamo essere spesso dominante nelle nostre misure, nell'esecuzione del best-fit.

Tuttavia in questa esperienza vogliamo produrre un grafico dell'intensità di corrente I che fluisce nel diodo in funzione della d.d.p. ΔV ad esso applicata, e inoltre vogliamo eseguirne un best-fit in cui i parametri, per essere confrontati con le attese, devono essere espressi in unità fisiche di corrente e tensione. Di conseguenza, per motivi che sono soprattutto di ordine pratico, preferiamo in questa nota seguire un approccio nel quale le grandezze digitalizzate vengono tutte convertite in unità fisiche prima della loro analisi, tenendo quindi conto delle incertezze di calibrazione nella conversione. Siete naturalmente invitati, tempo e voglia permettendo (per esempio, per una relazione semestrale), a utilizzare un approccio nel quale, invece, il contributo sistematico all'incertezza viene considerato solo dopo l'analisi dei dati, cioè dopo aver realizzato il best-fit, e quindi mantenuto separato da quello statistico.

A. Procedura di calibrazione e determinazione dell'errore

Per esprimere le misure in unità fisiche occorre innanzitutto determinare il fattore di conversione ξ (in [V/digit]), cioè conoscere il valore della tensione di riferimento V_{ref} usata internamente dal digitalizzatore di Arduino. Come già discusso altrove, questa tensione dipende dall'alimentazione, che avviene tramite porta USB e grossolanamente, cioè trascurando altre, più sottili, possibili cause di errore, essa equivale al valore massimo di tensione che può essere fornito da una porta digitale.

Nello sketch di Arduino si può osservare come, alla fine delle operazioni, la porta digitale PWM $\sim$`5` venga a trovarsi al valore di duty cycle corrispondente al 100%, dato che alla fine del ciclo di misura l'indice `i` che stabilisce il duty cycle ha raggiunto il suo valore massimo (255). In queste condizioni la porta fornisce un valore *continuo* che corrisponde al valore massimo possibile, e quindi grossolanamente a V_{ref}. La misura di V_{ref} può allora essere condotta in maniera molto semplice: è sufficiente collegare il multimetro digitale tra il punto indicato nel circuito con V_1 e la terra *alla fine delle operazioni* per poter leggere il suo valore. La misura deve naturalmente essere eseguita *a diodo scollegato*: infatti la presenza del diodo, che al termine del ciclo si trova ovviamente in conduzione, determina una rilevante richiesta di corrente al circuito costituito da Arduino, dall'integratore e dalla resistenza R_D. Questo circuito ha, chiaramente, una "resistenza interna" (à la Thévenin) tutt'altro che trascurabile, producendo una caduta di tensione tutt'altro che trascurabile. Sperimentalmente ve ne potrete rendere facilmente rendere conto osservando la differenza nella lettura del multimetro quando il diodo è scollegato o collegato al circuito. Si sottolinea a questo proposito che *scollegare significa scollegare*: almeno una delle connessioni al diodo va rimossa e non va fatta alcuna altra modifica al circuito, in particolare non si deve rimpiazzare il diodo con alcunché (men che meno con un cortocircuito).

Noto V_{ref} potrà essere determinato il fattore di conversione $\xi = V_{ref}/1023$. Dato che V_{ref} è stata misurata, e quindi il valore ha un'incertezza (l'errore del multimetro), anche ξ dovrà avere un'incertezza che possiamo interpretare in maniera diretta come *incertezza di calibrazione*.

Alla fine, l'incertezza sulle misure di d.d.p. potrà essere ottenuta sommando in quadratura l'errore statistico dovuto alla digitalizzazione (convenzionalmente preso pari a quella di lettura, cioè ± 1 digit) e quello sistematico attribuito alla calibrazione: state attenti al fatto che in nessun caso l'incertezza sulle misure di d.d.p. potrà essere nulla, anche quando è nulla la corrispondente lettura, dato che l'incertezza statistica è *sempre* diversa da zero.

Naturalmente nel contesto di questo esperimento trascuriamo altre possibili cause di incertezza, come per esempio quelle legate alla non linearità del digitalizzatore e alla conseguente presenza di un *offset* di lettura, che qui trascuriamo. Inoltre, visto che la misura prevede un notevole ritardo tra un'acquisizione e la successiva, potrebbe essere implementata nello sketch di Arduino una misura mediata su tante acquisizioni che permetterebbe di individuare l'errore statistico come deviazione standard di un campione (per noi l'errore statistico è *convenzionalmente* rappresentato da quello di lettura, affermazione che a rigore non è corretta). Tuttavia anche questa possibilità non è sfruttata, almeno al presente stato di evoluzione dell'esperimento (e solo per questioni di praticità).

Infine, per quanto riguarda la misura di I, l'incertezza va valutata attraverso propagazione dell'errore sull'Eq. 3. Ovviamente occorre in questo caso tenere conto anche dell'incertezza con la quale si determina il valore di R_D, misurata tipicamente con il multimetro digitale. È probabile che il valore di I risulti nullo (sotto la soglia di sensibilità) per alcune misure: anche in questo caso, l'incertezza associata non potrà in alcun modo essere nulla.

VII. ESEMPIO DI MISURA E COMMENTI

L'esperimento qui presentato è stato compiuto usando un integratore realizzato con $R = 680$ ohm (nominali, tolleranza 5%) e $C = 220$ μF (nominali, tolleranza 20%, il condensatore è elettrolitico). La frequenza di taglio nominale è quindi $f_T \simeq 1$ Hz, con tolleranza dominata da quella sulla capacità. La resistenza in serie al diodo è stata misurata di valore $R_D = (3.28 \pm 0.03)$ kohm, dove l'incertezza tiene conto dell'errore di calibrazione del multimetro nella scala utilizzata ($\pm 0.8\%$) e del suo proprio errore di lettura (± 1 digit). La tensione di riferimento, misurata con la procedura descritta sopra, è risultata $V_{ref} = (4.89 \pm 0.03)$ V. La somma delle resistenze $R + R_D$ è relativamente grande, circa 4 kohm, e quindi la richiesta di corrente alla porta di Arduino è relativamente piccola, dell'ordine del mA; in simili condizioni la porta digitale di Arduino approssima bene un generatore di d.d.p. ideale.

La Fig. 4 riporta un esempio delle misure: nella determinazione delle barre di errore sono state seguite le istruzioni specificate nella sezione precedente, cioè esse comprendono errore di lettura, di calibrazione ed eventuale propagazione.

Facciamo subito alcune osservazioni sui dati: intanto si nota che la spaziatura dei valori sull'asse orizzontale, che in principio ci aspetteremmo sempre pari a circa 20 mV, secondo quanto stabilito sopra, 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 circa 0.45-0.5 V richiede una spiegazione. Inoltre è anche evidente che il range spazzato nella misura potrebbe sembrare molto inferiore rispetto alle aspettative (all'uscita dell'integratore, come si può verificare facilmente osservando all'oscilloscopio il segnale V_1 a diodo scollegato, si ritrova una d.d.p. costante, o quasi-costante, che arriva fino a circa 5 V).

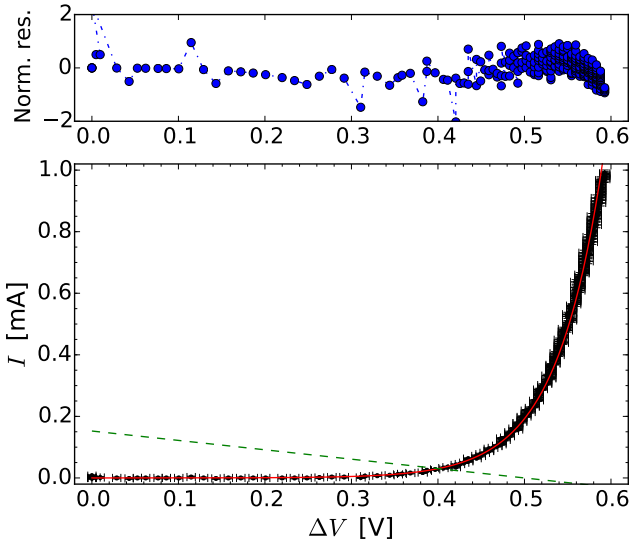

Figura 4. Esempio di misure ottenute con la scelta dei parametri e dei componenti circuitali indicati nel testo; la linea rossa continua rappresenta il best-fit condotto secondo l'Eq. 1 (commenti e risultati nel testo); il pannello superiore mostra il grafico dei residui normalizzati, dove la normalizzazione è effettuata con l'incertezza σ definita in Eq. 4, che tiene conto dell'errore sulla misura di I e di ΔV. La linea verde tratteggiata sovrapposta al grafico del pannello inferiore rappresenta la *retta di carico* per $V_1 = \bar{V}_1 = 0.5$ V, determinata come discusso nel testo.

Il motivo può essere facilmente intuito ricordando le caratteristiche di funzionamento del diodo bipolare a giunzione. Come evidenziato dall'andamento della curva caratteristica I-V, vedi Fig. 1(b), al di sopra di una certa V_{thr}, detta *tensione di soglia*, che è proprio tipicamente collocata tra circa 0.45 e circa 0.65 V per un ordinario diodo al silicio, la caduta di potenziale sul diodo si stabilizza grosso modo a questo valore, o appena al di sopra. Infatti in queste condizioni la resistenza della giunzione diventa praticamente trascurabile, a parte un (piccolo) contributo dovuto alla resistenza dei materiali che costituiscono l'anodo e il catodo, cioè i due elementi della giunzione stessa.

Nel nostro esperimento il diodo è alimentato da Arduino (attraverso la porta digitale PWM utilizzata) seguito da integratore e resistenza R_D. Considerato tutto assieme, il "generatore" che stiamo impiegando ha una resistenza interna tutt'altro che trascurabile. Impiegando l'approccio di Thévenin e trascurando la resistenza interna della porta di Arduino, la resistenza interna del nostro generatore è almeno pari a quella della serie $R + R_D$,

cioè, nel caso analizzato, dell'ordine di alcuni kohm. Per $\Delta V \gtrsim V_{thr}$ essa è dunque molto maggiore della resistenza effettiva del diodo, per cui c'è una forte caduta di potenziale sulla serie delle resistenze, mentre quella ai capi del diodo si attesta su valori prossimi, o poco superiori, a V_{thr}.

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. Poiché l'incertezza sulla misura di ΔV è non trascurabile rispetto a quella su I, per il fit è stata eseguita la propagazione dell'incertezza su ΔV nella determinazione di I, che è quindi stata sommata in quadratura all'incertezza ΔI. A questo scopo si è quindi definita l'incertezza σ usata nella routine di minimizzazione del χ^2 come

$$\sigma = \sqrt{(\Delta I)^2 + \left(\Delta(\Delta V)\frac{\partial I(\Delta V)}{\partial(\Delta V)}\right)^2} = \tag{4}$$

$$= \sqrt{(\Delta I)^2 + \left(\Delta(\Delta V)\frac{I + I_0}{\eta V_T}\right)^2}, \tag{5}$$

dove $\Delta(\Delta V)$ rappresenta l'incertezza in ΔV, tutti gli altri simboli sono ovvi, o già definiti, e si è calcolata la derivata di Eq. 1 rispetto a ΔV.

Notiamo che il fit riproduce in maniera non perfetta i dati sperimentali, come evidenziato dal grafico dei residui normalizzati riportato nel pannello superiore di Fig. 4, dove, notate bene, la normalizzazione è stata eseguita per l'incertezza σ definita sopra. Ci sono vari motivi per questo, il principale dei quali è che l'equazione di Shockley nasce da un modello che non è atteso descrivere in modo rigoroso la complessità dei fenomeni di trasporto elettronico in un diodo a giunzione in silicio di uso pratico. Inoltre, come si verifica spesso in acquisizioni automatizzate, alcune misure possono essere affette da disturbi sotto forma di spikes, legati all'acquisizione erronea di segnali transienti (rumori "ad alta frequenza"). La presenza di questi spikes può generalmente essere individuata semplicemente esaminando i dati sperimentali (prima di eseguire il best-fit). In caso essi siano chiaramente individuati, è possibile rimuoverli dal best-fit (ma non dal grafico), specificando in modo chiaro che si è compiuta questa operazione.

I risultati del best-fit sono i seguenti:

$$I_0 = (5.8 \pm 0.1)\ \text{nA} \tag{6}$$

$$\eta V_T = (54.0 \pm 0.2)\ \text{mV} \tag{7}$$

$$\chi^2/\text{ndof} = 55/254 \tag{8}$$

$$\text{Norm. cov.} = 0.99\ . \tag{9}$$

Il basso valore del χ^2_{rid} può essere legato alla sovrastima delle incertezze conseguente alla considerazione dell'errore sistematico (di calibrazione). In termini generali, sia la corrente di saturazione inversa I_0 che il valore di V_T sono in buon accordo con le aspettative per il diodo impiegato. Infine è evidente che il modello, Eq. 1, fornisce una forte covarianza normalizzata tra i parametri, dovuta alla circostanza che, per molti dei punti sperimentali analizzati, l'esponenziale prevale nettamente rispetto al termine -1. In queste condizioni le variazioni di I_0 e ηV_T sono fortemente correlate tra loro (quando uno diminuisce l'altro aumenta, da cui il segno positivo della covarianza). Di conseguenza, l'incertezza sui parametri ottenuti dal best-fit è probabilmente sottostimata.

VIII. RETTA DI CARICO

Questa sezione riguarda un aspetto "accessorio" dell'esperienza e del trattamento dati. Infatti può essere interessante tracciare la cosiddetta *retta di carico* sovrapposta al grafico della curva caratteristica I-V. Questa operazione, che nel contesto dell'esperimento qui considerato non ha grande significato pratico, viene tradizionalmente eseguita per determinare a priori, in modo grafico, il *punto di lavoro* di una giunzione bipolare in un circuito che contiene (almeno) la giunzione stessa e una resistenza. Il metodo prevede di partire dalla conoscenza di una tensione, per esempio quella di alimentazione, di tutto il circuito.

Immaginiamo allora di voler determinare il punto di lavoro del nostro diodo, cioè la sua corrente di lavoro I_q e la sua tensione di lavoro ΔV_q, quando il circuito costituito dal diodo stesso e dalla resistenza R_D è sottoposto a una certa d.d.p. $\bar{V}_1$ (applicata ovviamente tra il punto indicato con V_1 nello schema di Fig. 3 e la linea di massa, o terra). Il circuito da considerare è realizzato dalla serie di diodo e resistenza, per cui deve essere

$$\bar{V}_1 = R_D I + \Delta V\ . \tag{10}$$

Questa equazione corrisponde, nel piano I, ΔV della curva caratteristica, a una retta di pendenza negativa e intercette con asse verticale e orizzontale rispettivamente pari a $\bar{V}_1/R_D$ e $\bar{V}_1$. La retta in questione disegnata per lo specifico valore $\bar{V}_1 = 0.5$ V, è riportata come linea tratteggiata nel grafico di Fig. 4. Il punto di lavoro del diodo, cioè i valori di I_q e V_q nelle condizioni specificate, sono ottenuti dall'intercetta tra la curva caratteristica I-V e la retta di carico.