Laboratorio di Elettronica 2
Relazione Lezione 1

Federico Fabrizi, Pietro Pennestrì

8 aprile 2018

Indice

1 Sommario

In questo report sono riportate le simulazioni eseguite durante la prima lezione del Laboratorio di Elettronica 2. I circuiti proposti sono stati simulati in laboratorio attraverso `PSpice`. Le stesse simulazioni vengono qui eseguite

utilizzando software open source. In particolare per la creazione degli schematici è stato utilizzato `gEDA`[1], mentre le simulazione dei circuiti sono state eseguite con `ngspice`[2].

In qualche caso è stato utilizzato `python` come front end. Da ultimo è stato realizzato un apparato sperimentale.

2 Curve d'Uscita del BJT BFQ67W/PLP

2.1 Primary Sweep V1

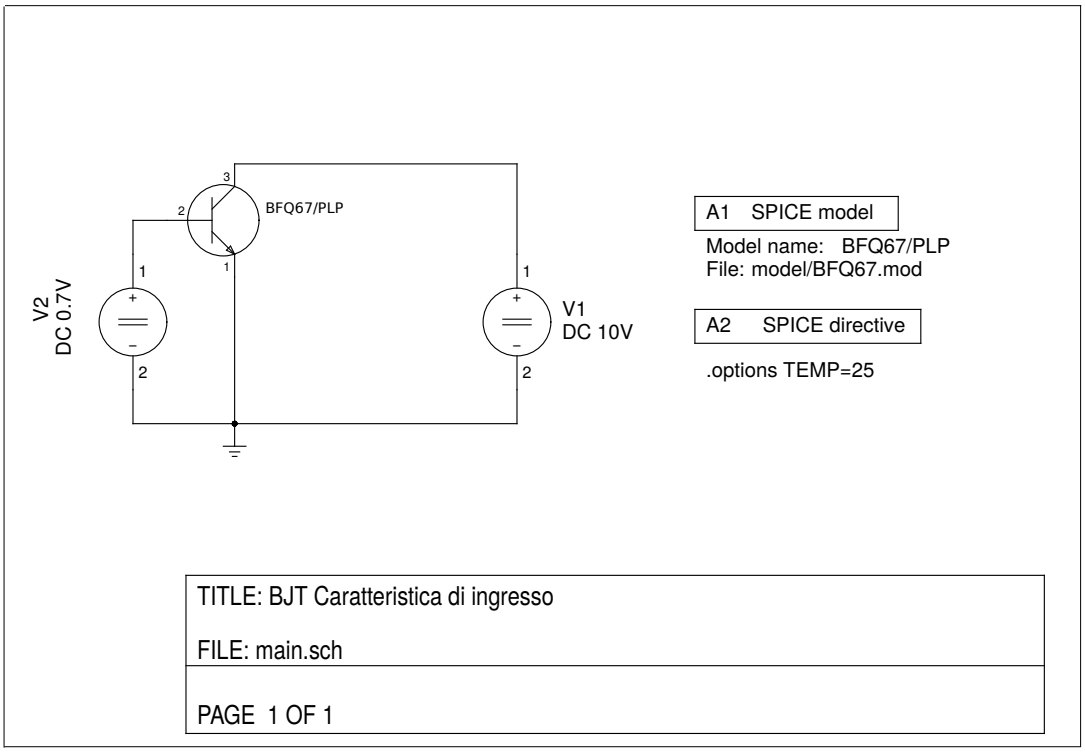


Figura 1: Schema del Circuito Simulato

Di seguito riportiamo la *netlist* ottenuta da terminale con il comando

[1] http://www.geda-project.org/
[2] http://ngspice.sourceforge.net

```
gnetlist -g spice-sdb -o circ.net main.sch .
```

```
*****************************************************
* Spice file generated by gnetlist *
* spice-sdb version 4.28.2007 by SDB -- *
* provides advanced spice netlisting capability. *
* Documentation at http://www.brorson.com/gEDA/SPICE/ *
*****************************************************
* BFQ67W SPICE MODEL
* PHILIPS SEMICONDUCTORS
* Date : September 1995
*
* PACKAGE : SOT323 DIE MODEL : BFQ65
* 1: COLLECTOR; 2: BASE; 3: EMITTER
.SUBCKT BFQ67W/PLP 1 2 3
Q1 6 5 7 7 BFQ65
* SOT323 parasitic model
Lb 4 5 .6n
Le 7 8 .6n
L1 2 4 .34n
L2 1 6 .1n
L3 3 8 .34n
Ccb 4 6 100f
Cbe 4 8 2f
*
* PHILIPS SEMICONDUCTORS Version: 1.0
* Filename: BFQ65.PRM Date: Feb 1992
*
.MODEL BFQ65 NPN
+ IS = 5.56440E-016
+ BF = 1.70000E+002
+ NF = 9.94874E-001
+ VAF = 4.80334E+001
+ IKF = 9.18182E-001
+ ISE = 1.04797E-014
+ NE = 1.47947E+000
+ BR = 1.42108E+002
+ NR = 9.94125E-001
```

```
+ VAR = 2.55564E+000
+ IKR = 9.63277E+000
+ ISC = 4.38252E-016
+ NC = 1.08948E+000
+ RB = 1.00000E+001
+ IRB = 1.00000E-006
+ RBM = 1.00000E+001
+ RE = 6.55965E-001
+ RC = 2.00000E+000
+ EG = 1.11000E+000
+ XTI = 3.00000E+000
+ CJE = 1.13769E-012
+ VJE = 6.00000E-001
+ MJE = 2.49462E-001
+ TF = 1.19796E-011
+ XTF = 2.59987E+001
+ VTF = 1.22309E+000
+ ITF = 1.97376E-001
+ PTF = 1.00348E+001
+ CJC = 5.15977E-013
+ VJC = 1.55871E-001
.ENDS



*========== Begin SPICE netlist of main design =======
.options TEMP=25
V2 2 0 DC 0.802V
V1 1 0 DC 10V
x1 1 2 0 BFQ67W/PLP
* DC analysis to sweep v1 from 0 to 10
.dc V1 0.1 10 0.001
.control
run
plot V(1)
plot -i(V1)
print -i(V1)
.endc
```

```
.end
```

Sembra opportuno precisare che è stato utilizzato per la simulazione il transistor `BFQ67/PLP`. Nella *netlist* sopra riportata, tale transitor viene caratterizzato per la simulazione come un *subcircuit*, ovvero un insieme di dispositivi noti di default al simulatore o precedentemente definiti nella *netlist* medesima. Un *subcircuit* viene identificato nella *netlist* con il prefisso **x**.

Nella *netlist* è stato inoltre incorporato il comando per la simulazione richiesta

```
.dc V1 0.1 10 0.001
```

ovvero una dc sweep di `V1` da 0 a 10 V con passo di 0.001V.

2.2 Secondary Sweep di V2

Durante l'esercitazione è stato richiesto oltre ad uno sweep di V1, un secondary sweep di V2. Per gestire in maniera automatica l'esecuzione dello sweep, nonostante `ngspice` ne permettesse la gestione diretta, si è preferito sviluppare un apposito script `python` che ne controllasse l'esecuzione. Tale scelta è giustificata dai seguenti motivi:

- Possibilità di indirizzare l'output dei vari sweep su file diversi, senza eccessive complicazioni della *netlist*;
- migliore resa grafica dei plot grazie alla libreria `matplotlib`;
- volontà di sperimentare python quale front end di `ngspice`.

Alleghiamo di seguito il listato dello script di `python`

Script python per la gestione dello sweep secondario

```
import string
import numpy as np
import matplotlib.pyplot as plt
import os

def secondary_sweep(inputfile_name,param,start,stop,step,plot,
    ↪ plot_title,x_label,y_label):
```

```
n=sum(c != ' ' for c in param)
tempfilename=string.split(inputfile_name , '.')[0]+'
    ↪ _temp'+'.net'
init_start=start
#start plot
plt.subplot(223)
plt.title(plot_title)
plt.grid()
while start<stop:
        inputfile = open(inputfile_name,'r')
        tempfile = open(tempfilename,'w')
        for line in inputfile:
                if param==line[:n]:
                        linesplit=string.split(line , ' ')
                        len_linesplit=len(linesplit)
                        linesplit[len_linesplit-1]=str(
                            ↪ start)
                        joinline=''
                        for words in linesplit:
                                joinline=joinline+' '+words

                        tempfile.write(joinline+'\n')
                elif 'run' in line:
                        tempfile.write(line)
                        tempfile.write('wrdata '+str(start
                            ↪ )+param+' '+plot + '\n')
                else:
                        tempfile.write(line)


        tempfile.close()
        inputfile.close()
        #os.system('ngspice -b '+tempfilename+' -o '+
            ↪ str(start)+param+'_res.txt')
        os.system('ngspice -b '+tempfilename)
        datafile = open(str(start)+param+'.data','r') #
            ↪ leg file risultati e plotta
        x_array=[]
```

```
                y_array=[]

                for sweep_data in datafile:
                        x=string.split(sweep_data ,'␣␣')[0]
                        y=string.split(sweep_data ,'␣␣')[1]
                        x=float(x)
                        y=float(y)
                        x_array=np.append(x_array, [[x]])
                        y_array=np.append(y_array, [[y]])

                plt.plot(x_array, y_array,label=param+':␣'+str(
                    ↪ start)+'V')
                datafile.close()
                start=start+step
        plt.xlabel(x_label)
        plt.ylabel(y_label)
        plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
            ↪ borderaxespad=0.)
        plt.savefig('thegraph.pdf')



#sweep(inputfile_name,param,start,stop,step,plot,plot_title,
    ↪ x_label,y_label)

secondary_sweep('mycir.net','V2',0.7,1,0.05,'-i(V1)','
    ↪ Caratteristica␣di␣uscita␣BFQ67W/PLP', 'V1␣[V]' ,'
    ↪ Corrente␣di␣collettore␣[A]')
```

Lo script produce i risultati riassunti nella Figura 2 :

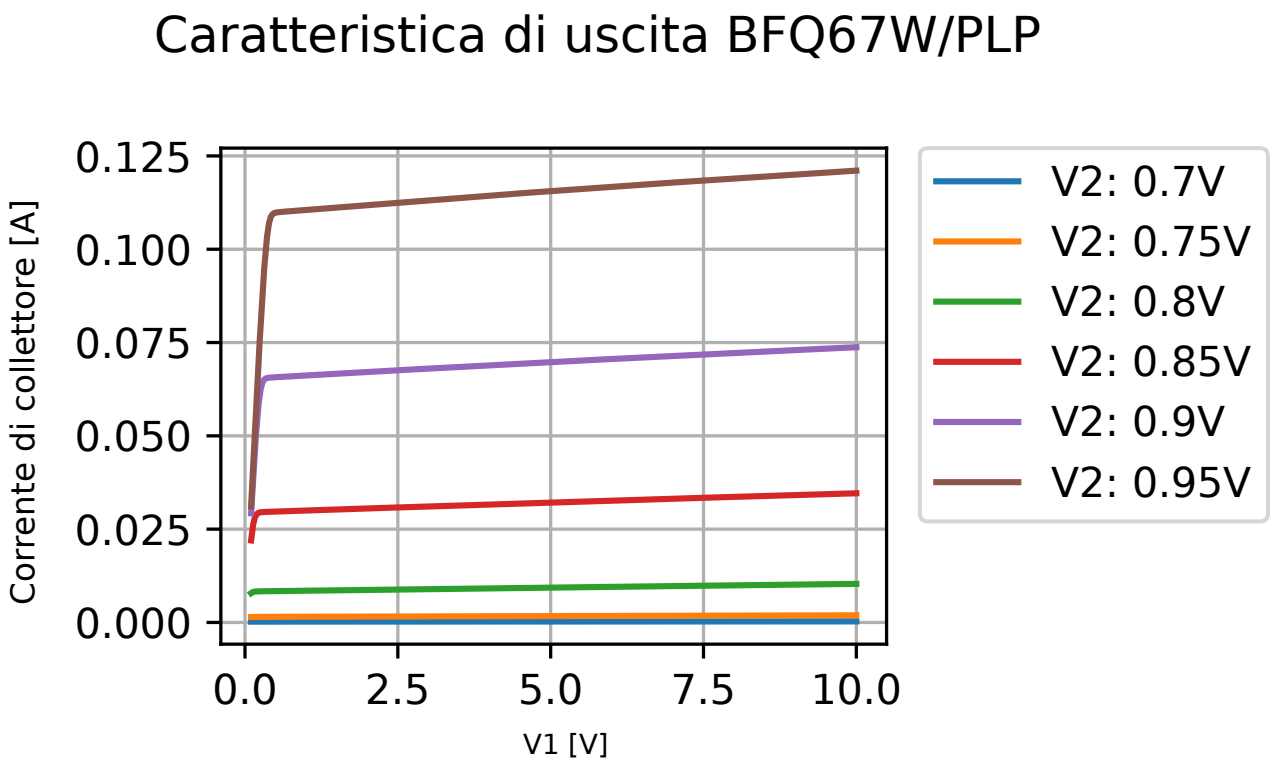


Figura 2: Simulazione del circuito schematizzato in Figura 1

3 Curve d'Ingresso del BJT BFQ67W/PLP

La *netlist* utilizzata per la caratterizzazione della caratteristica d'uscita è stata modificata per ottenere la caratteristica d'ingresso sostituendo il comando di simulazione come segue:

```
.dc V2 0 0.9 0.0001
```

Il risultato ottenuto è riassunto in Figura 3:

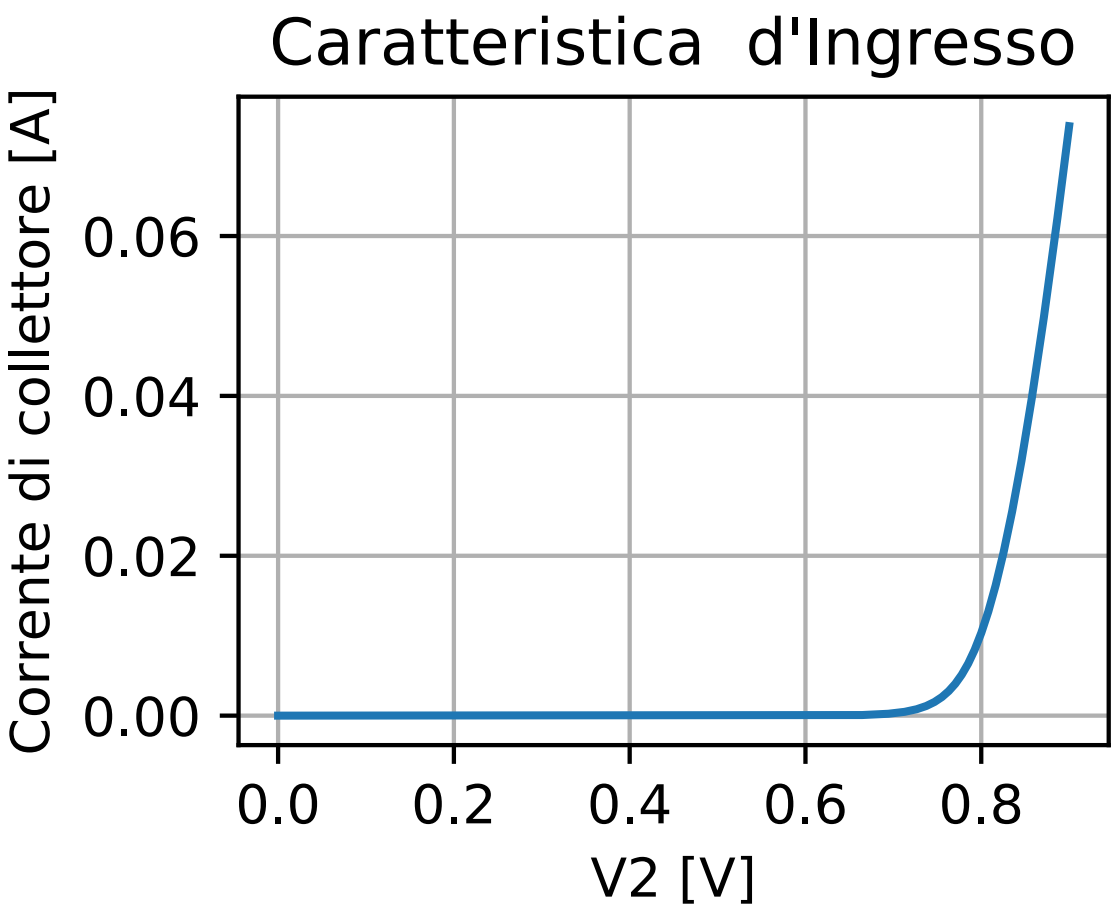


Figura 3: Simulazione caratteristica d'ingresso

4 Transcaratteristica statica del BJT BFQ67W/PLP a emettitore comune

Il circuito preso in considerazione nella seconda parte dell'esercitazione è mostrato in Figura 4.

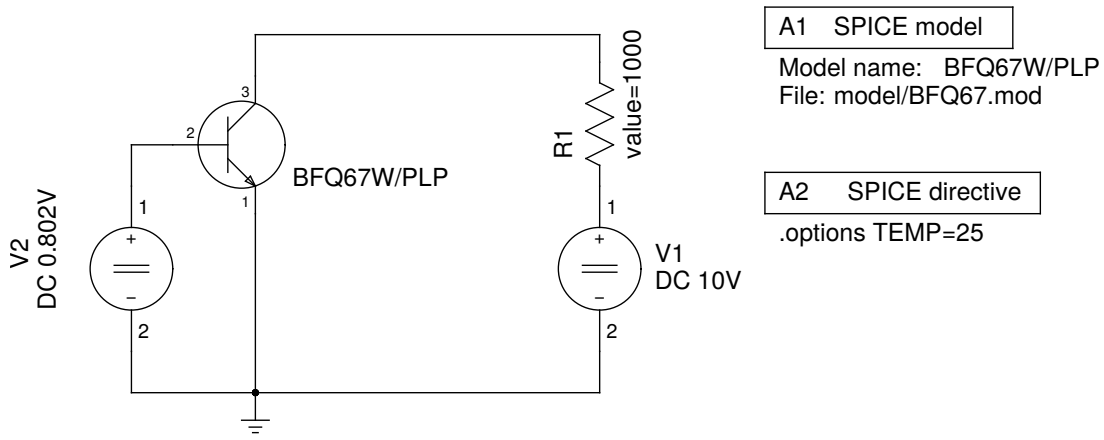


Figura 4: Emettitore Comune

```
* gnetlist -g spice-sdb -o main.net main.sch
************************************************************
* Spice file generated by gnetlist *
* spice-sdb version 4.28.2007 by SDB -- *
* provides advanced spice netlisting capability. *
* Documentation at http://www.brorson.com/gEDA/SPICE/ *
************************************************************
*vvvvvvvv Included SPICE model from model/BFQ67.mod vvvvvvvv
* BFQ67W SPICE MODEL
* PHILIPS SEMICONDUCTORS
* Date : September 1995
*
* PACKAGE : SOT323 DIE MODEL : BFQ65
* 1: COLLECTOR; 2: BASE; 3: EMITTER
.SUBCKT BFQ67W/PLP 1 2 3
```

```
Q1 6 5 7 7 BFQ65
* SOT323 parasitic model
Lb 4 5 .6n
Le 7 8 .6n
L1 2 4 .34n
L2 1 6 .1n
L3 3 8 .34n
Ccb 4 6 100f
Cbe 4 8 2f
*
* PHILIPS SEMICONDUCTORS Version: 1.0
* Filename: BFQ65.PRM Date: Feb 1992
*
.MODEL BFQ65 NPN
+ IS = 5.56440E-016
+ BF = 1.70000E+002
+ NF = 9.94874E-001
+ VAF = 4.80334E+001
+ IKF = 9.18182E-001
+ ISE = 1.04797E-014
+ NE = 1.47947E+000
+ BR = 1.42108E+002
+ NR = 9.94125E-001
+ VAR = 2.55564E+000
+ IKR = 9.63277E+000
+ ISC = 4.38252E-016
+ NC = 1.08948E+000
+ RB = 1.00000E+001
+ IRB = 1.00000E-006
+ RBM = 1.00000E+001
+ RE = 6.55965E-001
+ RC = 2.00000E+000
+ EG = 1.11000E+000
+ XTI = 3.00000E+000
+ CJE = 1.13769E-012
+ VJE = 6.00000E-001
+ MJE = 2.49462E-001
+ TF = 1.19796E-011
```

```
+ XTF = 2.59987E+001
+ VTF = 1.22309E+000
+ ITF = 1.97376E-001
+ PTF = 1.00348E+001
+ CJC = 5.15977E-013
+ VJC = 1.55871E-001
.ENDS
*^^^^^^^^ End of included SPICE model from model/BFQ67.mod
    ↪ ^^^^^^^^
*
*============== Begin SPICE netlist of main design ============
R1 3 1 1000
.options TEMP=25
V2 2 0 DC 0.802V
V1 3 0 DC 10V
x1 1 2 0 BFQ67W/PLP
.end
```

4.1 Sweep di V2

Eseguendo uno sweep del generatore `V2`:

```
.dc V2 0.1 0.9 0.001
```

otteniamo il grafico di Figura 5

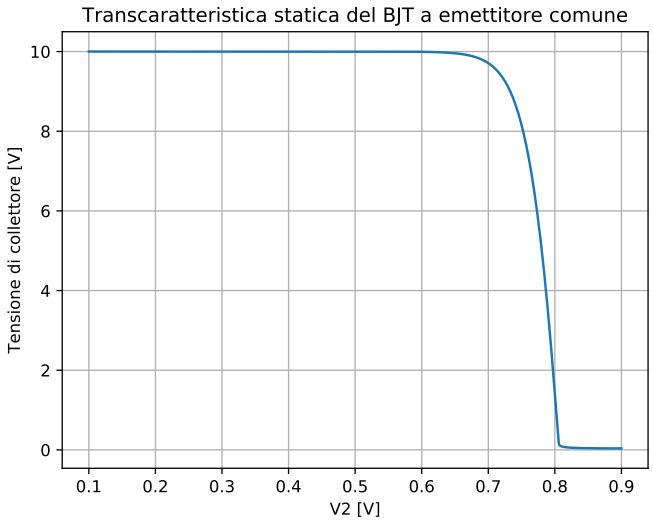


Figura 5: Risultato simulazione

4.2 AC Sweep

Consideriamo il circuito in Figura 6

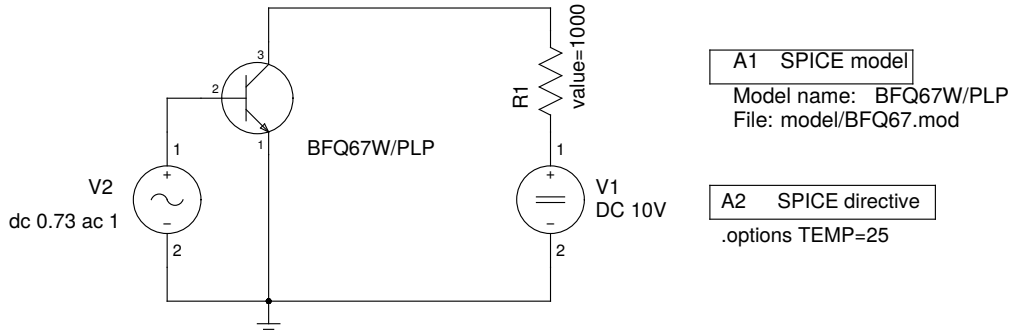


Figura 6: AC Sweep

eseguendo uno sweep in frequenza tramite il comando

```
.ac dec 100 10000 1000000000
```

otteniamo il risultato riportato in Figura 7.

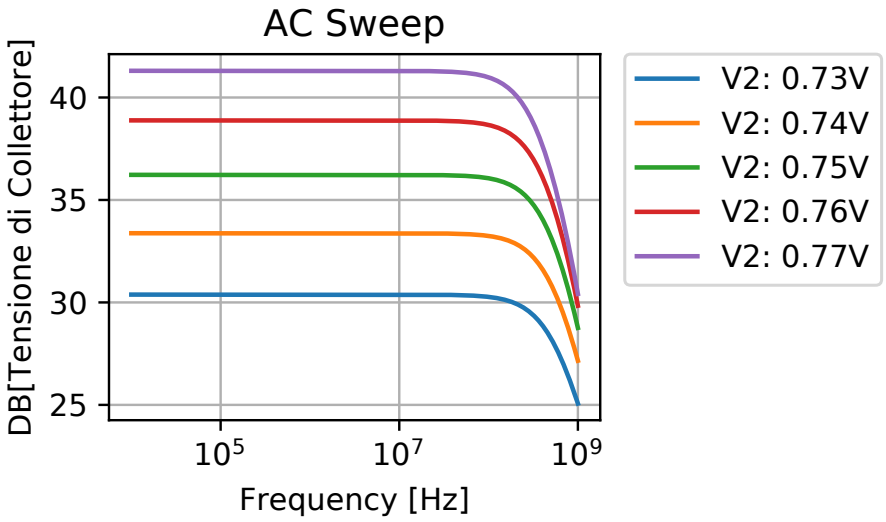


Figura 7: AC Sweep

La gestione del secondary sweep è ottenuta attraverso il seguente script `python`.

```
import string
import numpy as np
import matplotlib.pyplot as plt
import os

def secondary_sweep(inputfile_name,param,start,stop,step,plot,
    ↪ plot_title,x_label,y_label):
        n=sum(c != ' ' for c in param)
        tempfilename=string.split(inputfile_name , '.')[0]+'
            ↪ _temp'+'.net'
        init_start=start
        #start plot
        ax=plt.subplot(223)
        ax.set_xscale("log", nonposx='clip')
        plt.title(plot_title)
        plt.grid()
        while start<stop:
                inputfile = open(inputfile_name,'r')
                tempfile = open(tempfilename,'w')
                for line in inputfile:
                        if param==line[:n]:
                                tempfile.write('V2 2 0 dc '+str(
                                    ↪ start)+' ac 1 \n')
                        elif 'run' in line:
                                tempfile.write(line)
                                tempfile.write('wrdata '+str(start
                                    ↪ )+param+' '+plot + '\n')
                        else:
                                tempfile.write(line)


                tempfile.close()
                inputfile.close()
                #os.system('ngspice -b '+tempfilename+' -o '+
                    ↪ str(start)+param+'_res.txt')
```

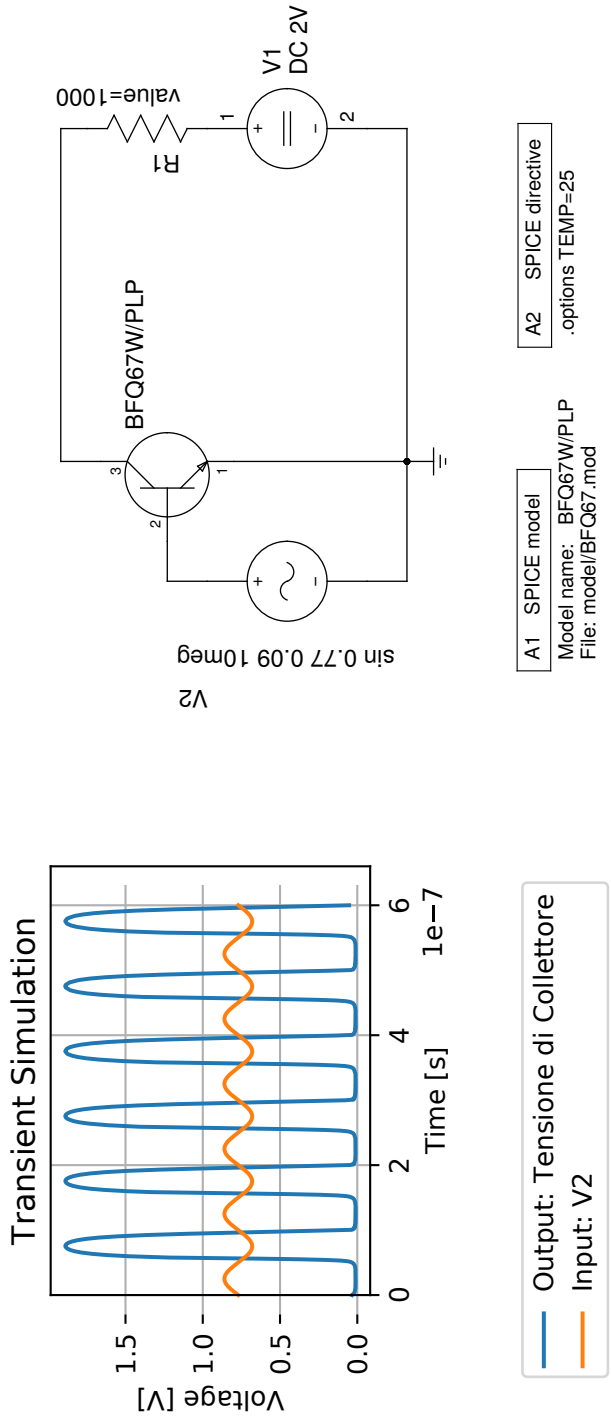
```
            os.system('ngspice␣-b␣'+tempfilename)
            datafile = open(str(start)+param+'.data','r') #
               ↪ leg file risultati e plotta
            x_array=[]
            y_array=[]

            for sweep_data in datafile:
                    x=string.split(sweep_data ,'␣␣')[0]
                    y=string.split(sweep_data ,'␣␣')[1]
                    x=float(x)
                    y=float(y)
                    x_array=np.append(x_array, [[x]])
                    y_array=np.append(y_array, [[y]])

            plt.plot(x_array, y_array,label=param+':␣'+str(
               ↪ start)+'V')
            datafile.close()
            start=start+step
        plt.xlabel(x_label)
        plt.ylabel(y_label)
        plt.legend(bbox_to_anchor=(1.05, 1), loc=2,
           ↪ borderaxespad=0.)
        plt.savefig('thegraph.pdf')

secondary_sweep('main.net','V2',0.73,0.78,0.01,'vdb(1)','AC␣
   ↪ Sweep', 'Frequency␣[Hz]' ,'DB[Tensione␣di␣Collettore]')
```

4.3 Transient Analysis



5 Apparato Sperimentale

Pur non avendo a disposizione il transistor preso in esame nel report, abbiamo realizzato un apparato sperimentale sulla base dello schema di Figura 4. Il transistor utilizzato è il `NPN 2N222`, con una resistenza `R`=2k. L'apparato è mostrato in Figura 8.

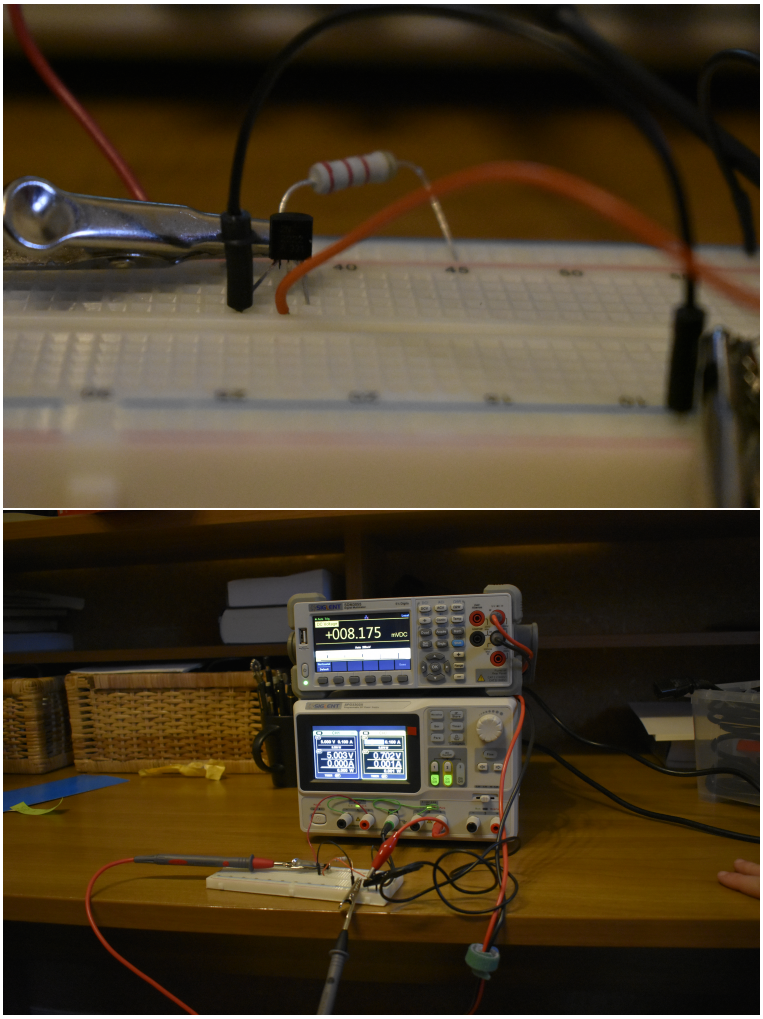


Figura 8: Apparato sperimentale

Le misure realizzate sono visibili nelle seguenti Figure 9

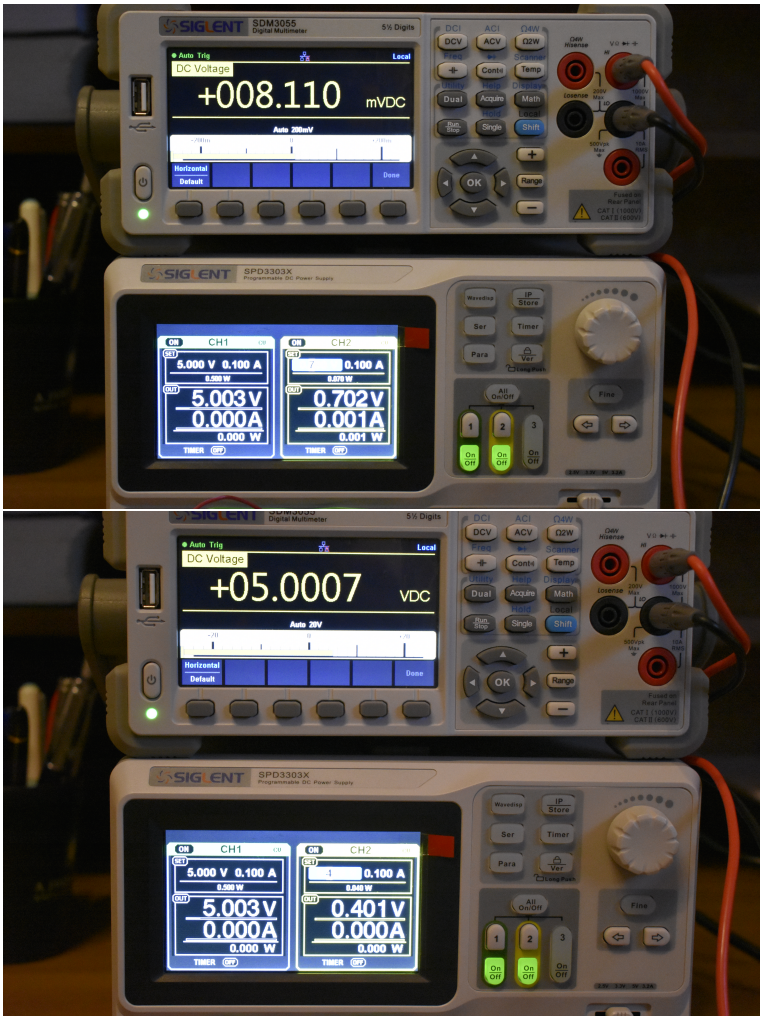


Figura 9: Catena di Misura: Generatore di Tensione e Multimetro

Il comportamento sperimentalmente riprodotto è quello di Figura 5.