

Laboratorio di Segnali e Sistemi

- Esercitazione -9 -

DFT con Arduino



Claudio Luci
SAPIENZA
UNIVERSITÀ DI ROMA

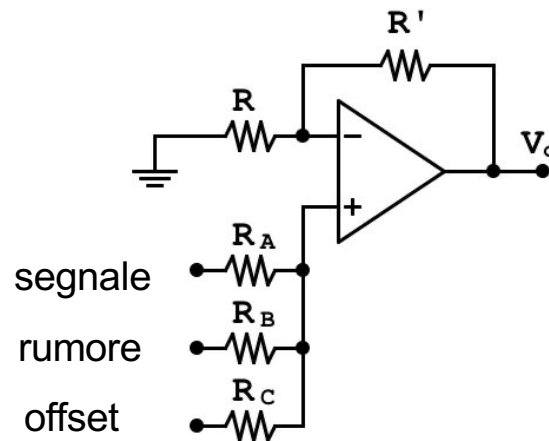
last update : 070117

Argomenti dell'esercitazione:

- DFT di un segnale sinusoidale:
 - Al di sotto della frequenza di Nyquist
 - Al di sopra della frequenza di Nyquist → alias
- DFT di un'onda quadra
- DFT del rumore
 - Con e senza filtro
- DFT di un segnale impulsivo
 - Aggiunta del rumore
 - Rumore filtrato

Scopo dell'esercitazione

- ❖ Lo scopo di questa esercitazione è quello di effettuare con Arduino, con l'ausilio dell'applicazione Processing, un'analisi in frequenza di segnali, periodici o impulsivi, in assenza o in presenza di rumore.
- ❖ Per i segnali periodici possiamo utilizzare il generatore di funzioni, mentre per i segnali impulsivi utilizzeremo lo stesso Arduino.
- ❖ Potremo poi aggiungere del rumore (prodotto dal generatore già costruito) utilizzando il sommatore a tre ingressi già costruito.



Occorre un sommatore a tre ingressi per aggiungere un offset al rumore quando il segnale è zero (studio dello spettro di frequenza del rumore).

- ◆ Analisi in frequenza: calcolo della trasformata di Fourier discreta
- ◆ Segnali periodici: sinusoidi a diverse frequenze, onda quadra
- ◆ Segnali impulsivi: "scarica" di un condensatore
- ◆ Inoltre vedremo come cambia lo spettro in frequenza con l'aggiunta del filtro passa-basso

Software

- ❖ Per Arduino utilizzeremo lo sketch `adc_read_5_2019` (che potete scaricare dal mio sito), scritto per lavorare in congiunzione con lo sketch di Processing `adc_arduino_5.pde`, che trovate sul mio sito.
- ❖ L'interfaccia grafica dello sketch Processing ci consente di:
 1. Ordinare ad Arduino di avviare la presa dati. Il segnale verrà campionato 800 volte ad una frequenza di circa 10 kHz (finestra di campionamento di 80 ms) e i dati verranno poi inviati a Processing.
(Da notare che Arduino Due consentirebbe un campionamento fino ad una frequenza massima di circa 200 kHz, ma il software utilizzato è stato ottimizzato per lavorare con Arduino Uno, la cui frequenza massima è di circa 9 kHz).
 2. Calcolare la trasformata di Fourier discreta del campione e la sua antitrasformata.
 3. Visualizzare il campione e il suo spettro in frequenza.
 4. Salvare su file le misure e i risultati del calcolo, per successive analisi offline.

- abbiamo impostato `analogRead()` in modo da aver bisogno di circa 100 μ s per essere completata; di conseguenza la massima frequenza di campionamento f_c è di circa 10 kHz
- Quindi la frequenza di Nyquist è di circa 5 kHz

N.B. In questa esercitazione dovrete usare Processing (e Arduino) come una black-box. Non c'è nessun software da scrivere, come avete fatto nell'esercitazione precedente, ma ci sono tante misure da fare



Processing

Download

Documentation

Learn

Teach

About

Donate

Search



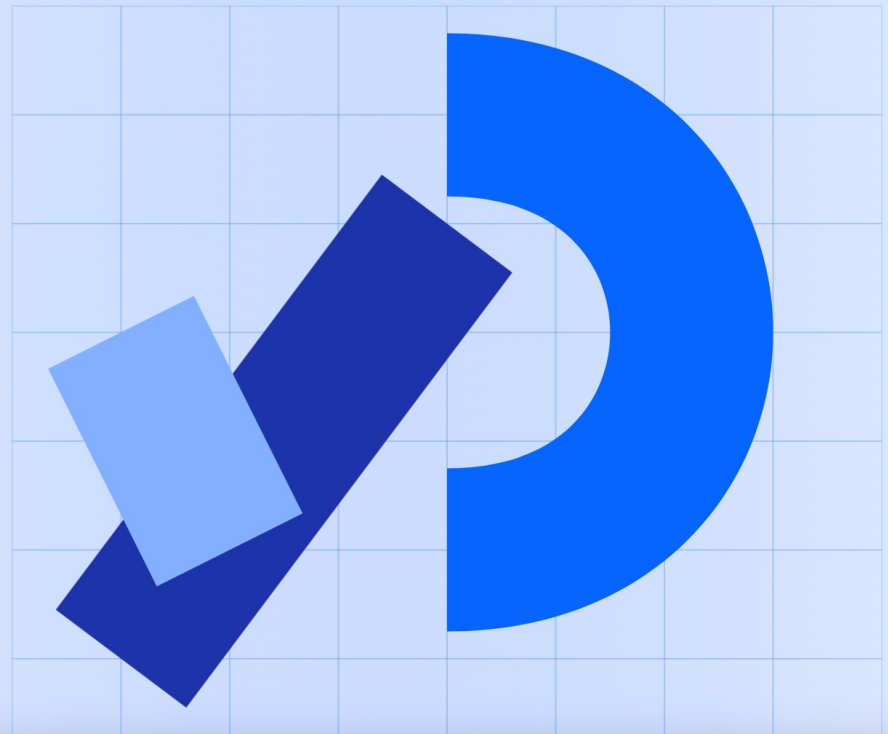
Welcome to Processing!

Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts. Since 2001, Processing has promoted software literacy within the visual arts and visual literacy within technology. There are tens of thousands of students, artists, designers, researchers, and hobbyists who use Processing for learning and prototyping.

Download

Reference

Donate



Noi non ci occupiamo di **visual arts**, quindi perché usiamo Processing?

Arduino e Processing

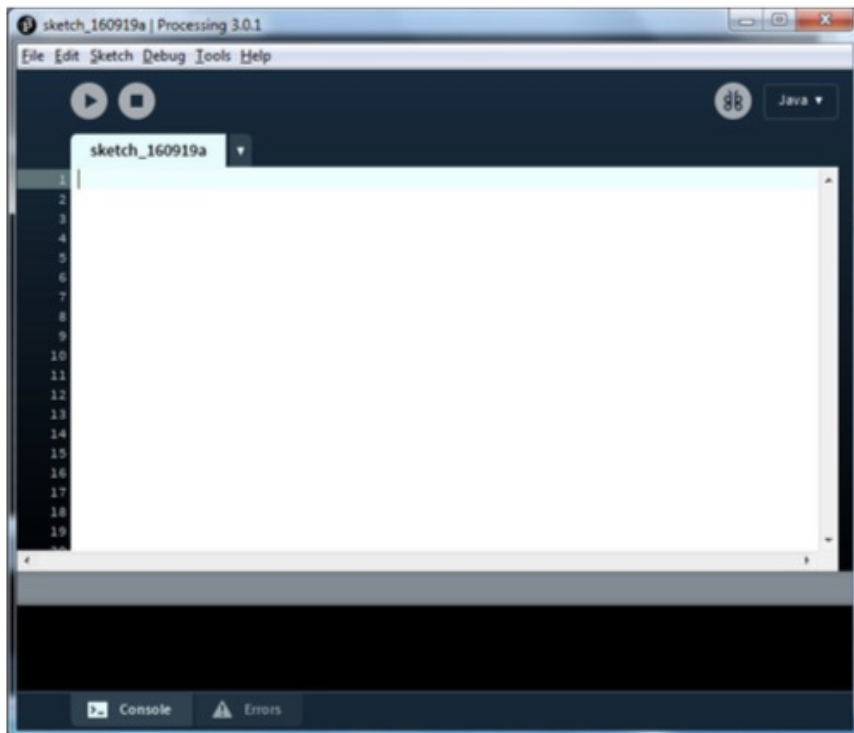
- ❖ Processing è un'applicazione che consente di sviluppare contenuti interattivi. Eredita la sintassi e i comandi di programmazione a oggetti (object oriented program) dal linguaggio Java ma in aggiunta mette a disposizione numerose funzioni ad alto livello per gestire gli aspetti grafici e multimediali.
- ❖ È distribuito sotto la licenza libera GNU General Public License, scaricabile gratuitamente dal sito ufficiale www.processing.org; è supportato dai sistemi operativi Linux, Mac OS e Windows.
- ❖ **Processing può interagire con la porta seriale di Arduino tramite USB, consentendo quindi di analizzare sul PC i dati raccolti dal microcontrollore.**

Lo sketch *adc_arduino_5.pde* di Processing fa (molto meglio) quello voi avete fatto “a mano” la volta scorsa nell'acquisire i vari punti di una data forma d'onda; in pratica è un piccolo sistema di acquisizione dati completo (DAQ)

- 1) Dice a arduino (sketch *adc_read_5_2019*) di leggere i valori dell'adc ad una frequenza di 10 kHz
- 2) arduino scrive i valori (800 punti) in un buffer interno
- 3) alla fine del ciclo di lettura (~ 80 ms) arduino manda tutti i valori a processing via porta seriale
- 4) Processing calcola la DFT con l'algoritmo della Fast Fourier Transform
- 5) Fa il grafico della forma d'onda acquisita e della trasformata di Fourier discreta
- 6) Salva tutti questi valori su un file per permettere ulteriori analisi off-line

Avvio dell'applicazione processing

- ❖ Al primo uso dell'applicazione è necessario eseguire alcune operazioni preliminari:
- ❖ Avviate Processing.
(Se non lo trovate sul Desktop potete cercare in C:/Programmi/processing-3.0.1/processing)
In questo modo viene creata automaticamente una cartella Processing nella vostra cartella Documenti e si apre una finestra con uno sketch vuoto come in figura.
- ❖ Senza scrivere nulla salvate lo sketch vuoto con un nome qualunque dentro questa cartella.
Notare che, come Arduino, Processing salva ogni programma in una cartella con lo stesso nome.
- ❖ **Poi chiudete Processing.**



- ❖ Processing ha bisogno della libreria *ControlP5*.
Se non fosse installata scaricate dal mio sito il file *ControlP5.zip*, decomprimetelo e installatelo nella cartella Documenti/Processing/Libraries.
- ❖ Voi avete bisogno dello sketch di Processing *adc_arduino_5.pde*. Nel caso non fosse già disponibile nel vostro PC, potete scaricare dal mio sito il file *adc_arduino_5.zip*.
Decomprimetelo e installate la cartella *adc_arduino_5* nella cartella Documenti/Processing.
Essa contiene il file in questione.
- ❖ Se non fosse già presente, scaricate anche lo sketch di Arduino *adc_read_5_2019*.

Uso del programma `adc_arduino_5`

- ❖ Dopo aver collegato la scheda Arduino Due e avviato su di essa il programma `adc_read_5_2019`, avviate `adc_arduino_5` (cliccandoci sopra): si aprirà la finestra mostrata in figura.
- ❖ Questo programma è pronto per essere utilizzato: premendo il bottone di avvio si apre la finestra grafica di interfaccia (vedi figura successiva).
- ❖ A questo punto dovete verificare che `adc_arduino_5` sia connesso alla stessa porta USB a cui è connesso Arduino: nella finestrella di controllo in basso di Processing (vedi figura) appare l'elenco di tutte le porte USB disponibili; per default il programma è connesso alla porta 1 (nell'esempio corrisponde a COM4).
Se necessario modificate la variabile `portNumber` nel codice `adc_arduino_5` e salvate la modifica.
- ❖ Ora potete avviare la presa dati con il bottone START.
Viene inviato al microcontrollore il comando di inizio campionamento e i dati raccolti vengono restituiti a `adc_arduino_5` che produce due grafici, quello del campione e quello della trasformata. Come esempio potete vedere un'onda quadra con frequenza di 500 Hz.
- ❖ È possibile salvare i dati su un file di testo inserendo un nome di file (per esempio `dati.txt`) e premendo il bottone SAVE. Il file verrà salvato nella cartella Documenti/Processing/`adc_arduino_5` e potrà essere successivamente analizzato con un qualunque altro programma (OpeOffice, Python, etc..)

Finestra del programma adc_arduino_5

The screenshot shows the Processing IDE interface for the program 'adc_arduino_5'. The code editor contains the following code:

```
1 import processing.serial.*;
2 import controlPS.*;
3 import processing.pdf.*;
4
5 /* Analizza il campione raccolto con Arduino con il progr adc_read_5 */
6
7 /*****
8  int portNumber = 1; /* change the 0,1,2 etc. to match your port */
9  *****/
10 PrintWriter output;
11 ControlPS cp5;
12 Serial myPort;
13 PFont font1;
14 PFont font2;
15 int buttonColor = color(255,0,0);
16 int captionColor = color(0,0,0);
17 String portName;
18 int portSpeed = 9600;
19 int flag=0;
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

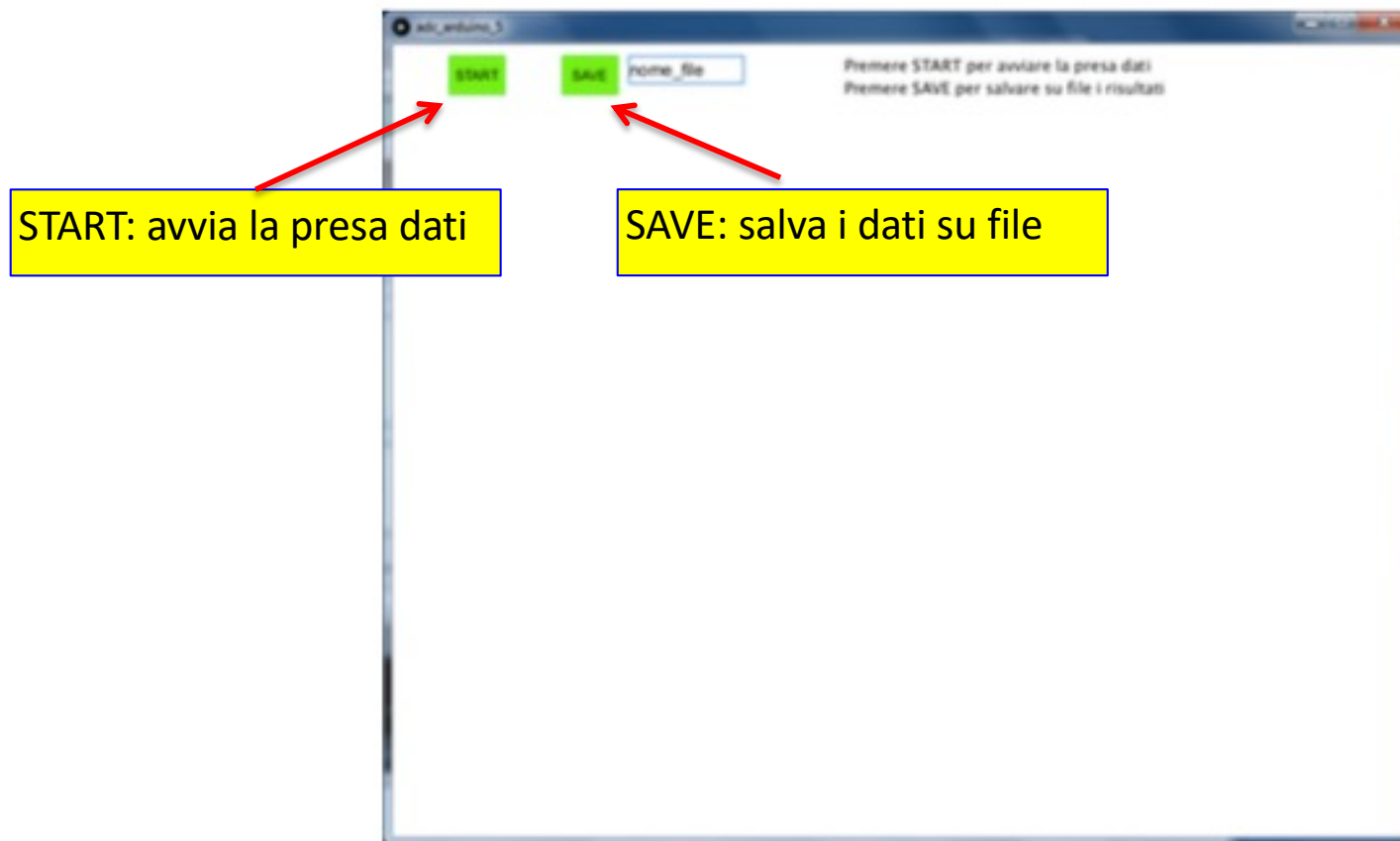
The console output shows the following text:

```
[0] *COM3*
[1] *COM4*
ControlPS 2.2.5 Infos, comentarios, questions at http://www.sojamo.de/libraries/controlPS
```

Annotations in the image:

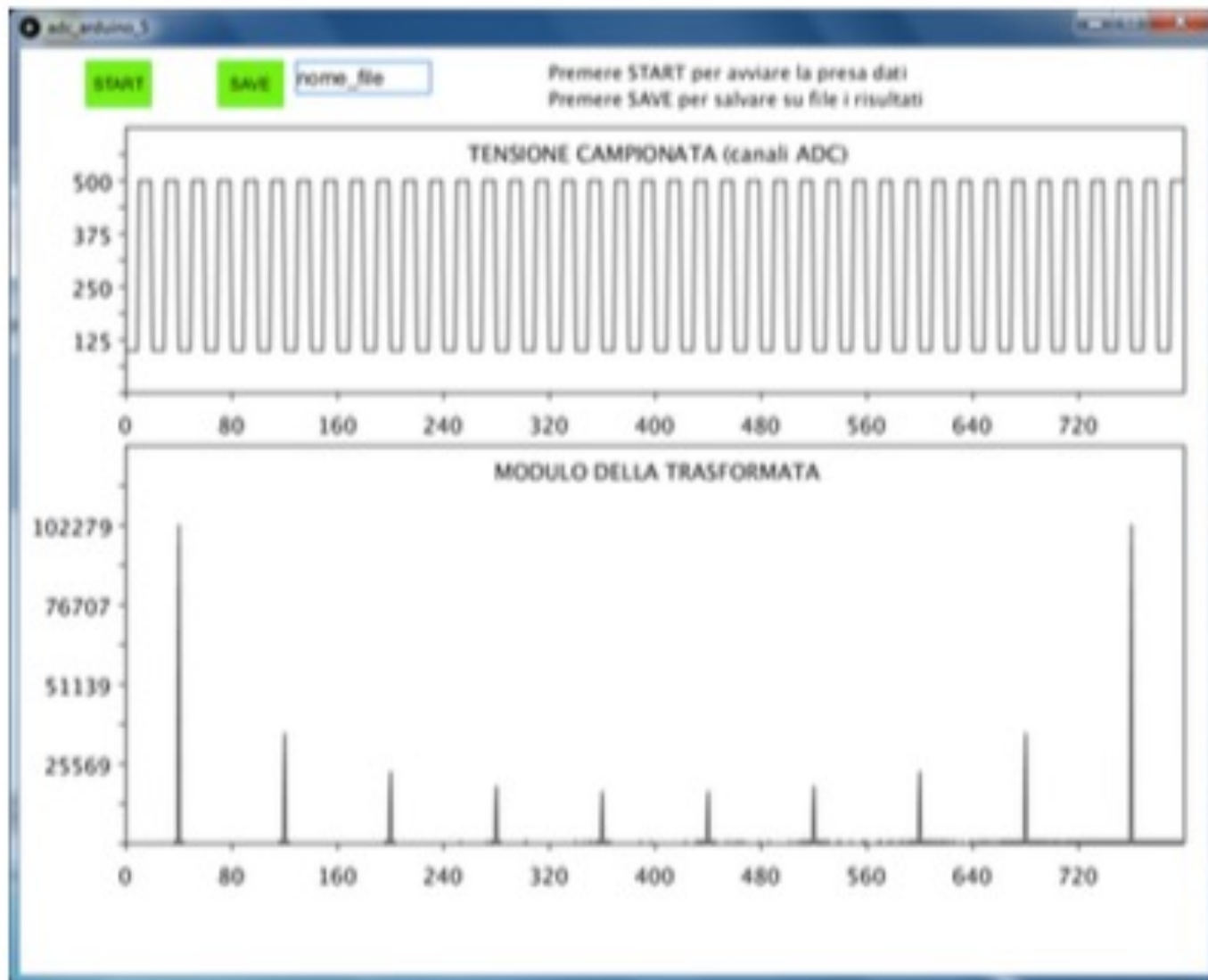
- Avvio del programma**: Points to the green play button in the top toolbar.
- Arresto del programma**: Points to the square stop button in the top toolbar.
- Variabile da modificare se necessario**: Points to the line `int portNumber = 1;` in the code editor.
- All'avvio del programma vengono mostrate le porte USB disponibili**: Points to the console output showing available USB ports.

Interfaccia grafica di adc_arduino_5



- ❖ Il programma `adc_arduino_5`, quando è in esecuzione, prende il controllo della porta USB designata e questo impedisce ad altri processi di utilizzarla.
- ❖ Quindi, se ad esempio è necessario ricaricare il programma sul microcontrollore (tramite la finestra dell'applicazione Arduino), occorre arrestare `adc_arduino_5`: questo può essere ottenuto premendo il bottone di arresto del programma (vedi figura precedente).

Esempio di interfaccia grafica



Onda quadra a 500 Hz

adc_read_5_2019 (arduino)

```
int ledPin = 13; //Collegato al led sulla scheda  
int signalPin = 12; //M. Raggi 13/07/2017
```

È acceso quando il programma di campionamento di arduino sta runnando

```
digitalWrite(ledPin, HIGH);
```

Va prelevato il segnale da mandare al filtro RC passa-basso

```
/* Legge ADC */
```

```
int ii = 0;  
for (int i = 0; i < npoint; i++)  
{  
  if(i==100){  
    digitalWrite(signalPin,HIGH); // Apre un gate sul pin 12 digitale all'interno della finestra di lettura ADC  
  }  
  if(i==100+Delay)digitalWrite(signalPin,LOW); //chiude il gate sul pin 12  
  //crea un segnale di lunghezza sufficiente a caricare il condensatore del filtro RC 100KOhm 100nF  
  // la tipica lunghezza e' di 110us per ogni lettura di analogread quindi con 20 letture circa 2ms
```

```
int value = analogRead(3);  
data[ii] = highByte(value);  
data[ii+1] = lowByte(value);  
ii = ii +2;
```

Il segnale analogico del generatore o dell'uscita del sommatore o del filtro Butterworth va mandato al pin analogico 3

```
delayMicroseconds(95); // to go to 10000Hz sampling (take into account 5 us time transmission)  
/* Serial.println(value); */
```

analogRead di Arduino Uno impiega circa 110 micros per fare una lettura, mentre quello di Arduino Due impiega solo 5 microsecondi. Quindi abbiamo messo un ritardo di 95 microsecondi per avere le stesse prestazioni

Come prima cosa verificate che il pin analogico 3 di arduino funzioni. Per far questo usate lo sketch che avete usato per la calibrazione dell'ADC: mandate una tensione continua in input e leggete il responso di Arduino sulla seriale. Se non funziona, mandate il segnale su un altro pin, ma poi dovete modificare di conseguenza il programma adc_read_5_2019. Fatto ciò caricate il programma adc_read_5_2019 e non toccate più arduino.

Iniziamo le misure

Segnale sinusoidale

- ❖ Possiamo utilizzare un segnale sinusoidale (con una frequenza inferiore alla frequenza di Nyquist), senza rumore, per verificare il buon funzionamento del sistema. Lo spettro in frequenza ottenuto sperimentalmente ci consente anche di calibrare con precisione la nostra scala di frequenza.
- ❖ Possiamo poi studiare il fenomeno dell'*aliasing* utilizzando segnali sinusoidali con frequenza superiore a quella di Nyquist, verificando la validità della relativa formula teorica.

Onda quadra

- ❖ Studiate poi lo spettro in frequenza di un'onda quadra (con una frequenza inferiore alla frequenza di Nyquist) e verificate che esso corrisponda a quanto atteso teoricamente, individuando i picchi delle armoniche e quelli dovuti all'*aliasing*.

Ricordate che l'ADC di Arduino converte solo i segnali compresi tra 0 e 3.3 V; aggiungete quindi un offset al segnale, sul generatore, in modo che esso ricada in questo intervallo

Segnale sinusoidale

- ❑ Misurate con Arduino e Processing, senza aggiungere il rumore, lo spettro di una sinusoide generata con il generatore di segnali:
 - Collegate il BNC del generatore alla breadboard
 - Utilizzate un filo per connettere il segnale dalla bradboard al ANALOG IN A3 di Arduino
 - Collegate il GND della beadboard a quello di Arduino
- ❑ **Inizialmente scegliete una frequenza inferiore a quella di Nyquist dell'ADC (10 kHz/2 = 5.0 kHz)**
- ❑ **Studiate poi il fenomeno dell'aliasing utilizzando segnali sinusoidali con frequenza superiore a quella di Nyquist.**
- ❑ **Effettuare lo stesso studio con un'onda quadra.**



Sinusoide a 200 Hz

- Caratteristiche Arduino

- ▶ 1024 conteggi, 0-3.3 V
- ▶ 800 campionamenti a circa 10 kSPS

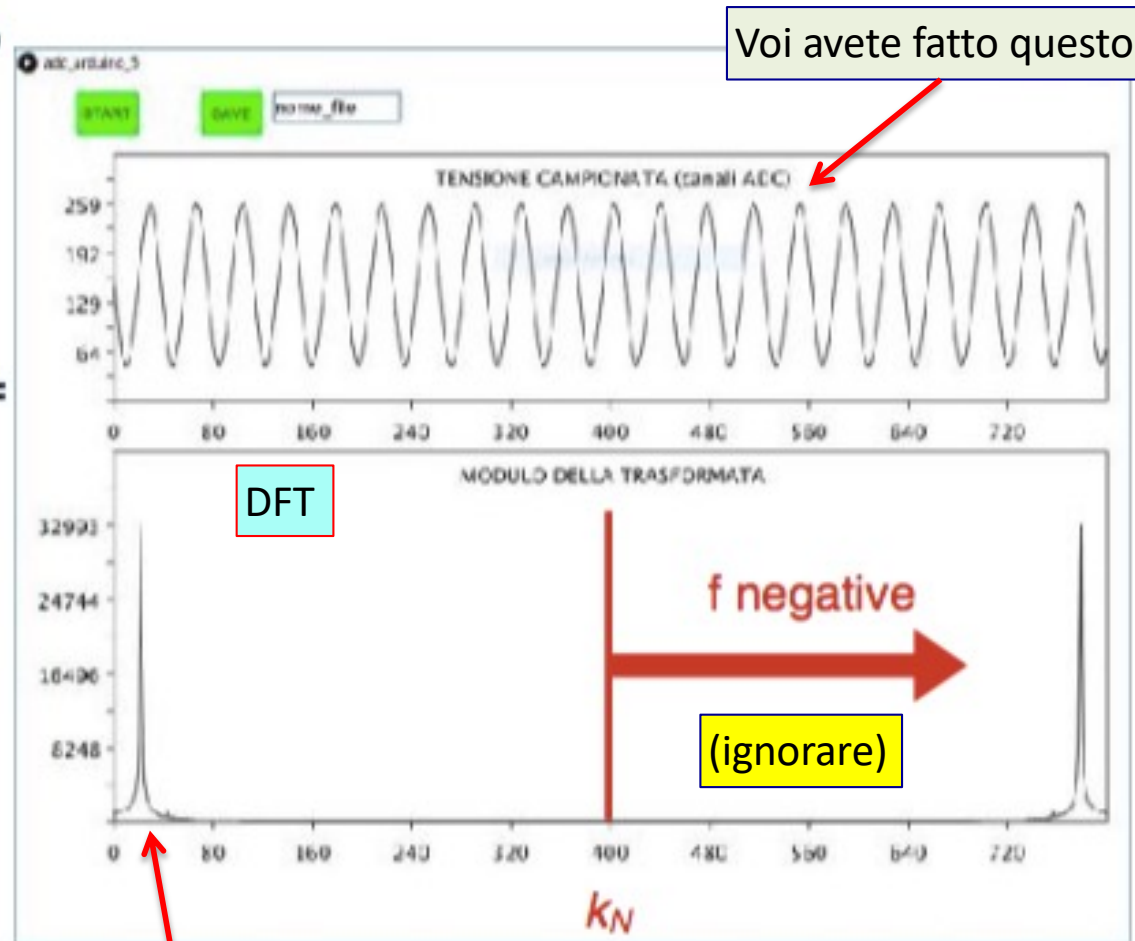
Sample Per Second

- Pertanto

- ▶ Volt / count = $3.3 / 1024 = 3.22 \text{ mV / count}$

10 bit

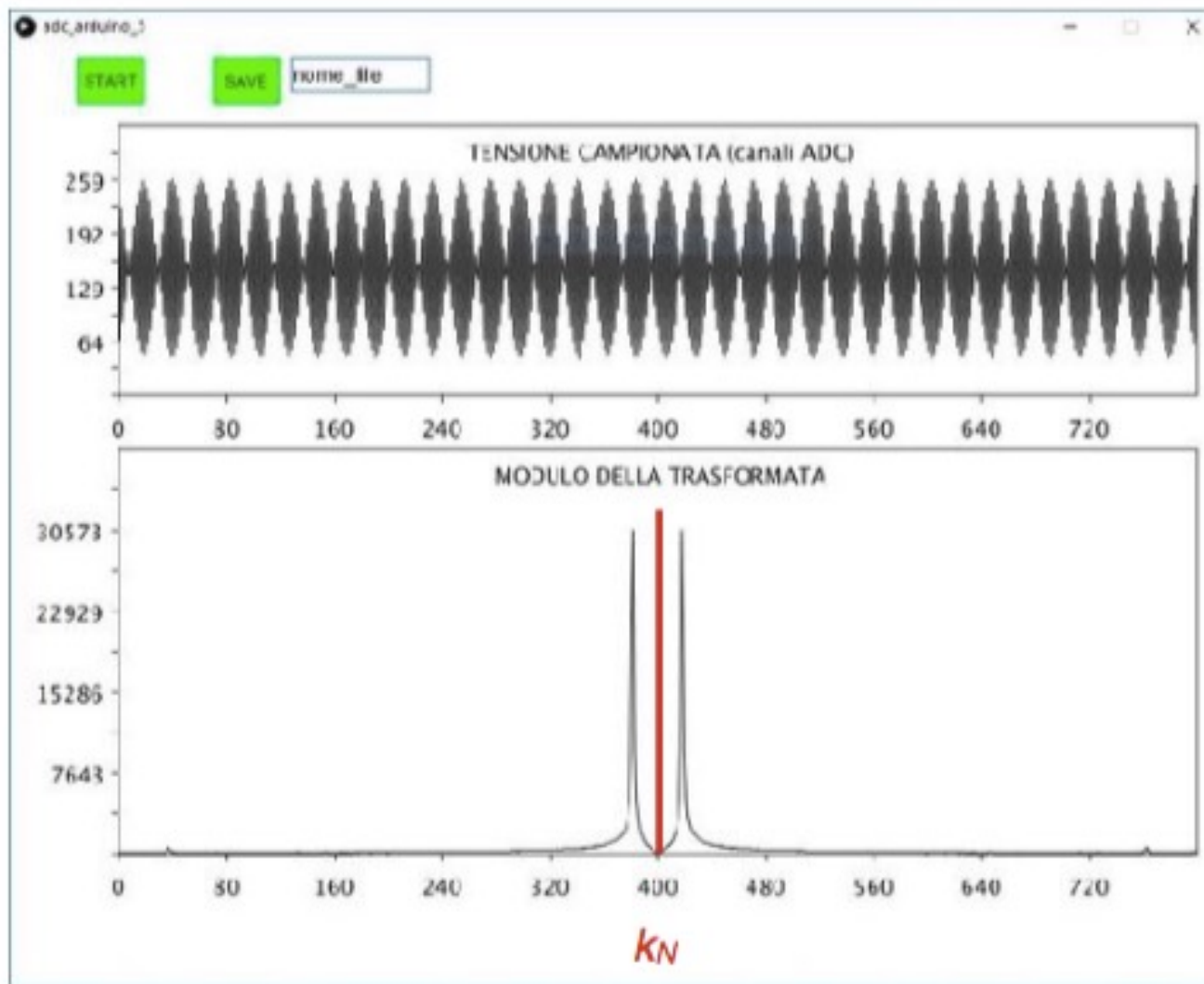
- ▶ $N = 800$
- ▶ $f_s = 10 \text{ kHz}$
- ▶ $T = N / f_s = 80 \text{ ms}$
- ▶ $\Delta t = 1 / f_s = 100 \mu\text{s}$
- ▶ $\Delta f = f_s / N = 12.5 \text{ Hz}$
- ▶ $f_N = 5 \text{ kHz}$
- ▶ $k_N = 400$



- $k_{\text{segnale}} = 200 \text{ Hz} / 12.5 \text{ Hz} = 16$

$f = k \times 12.5 \text{ Hz}$

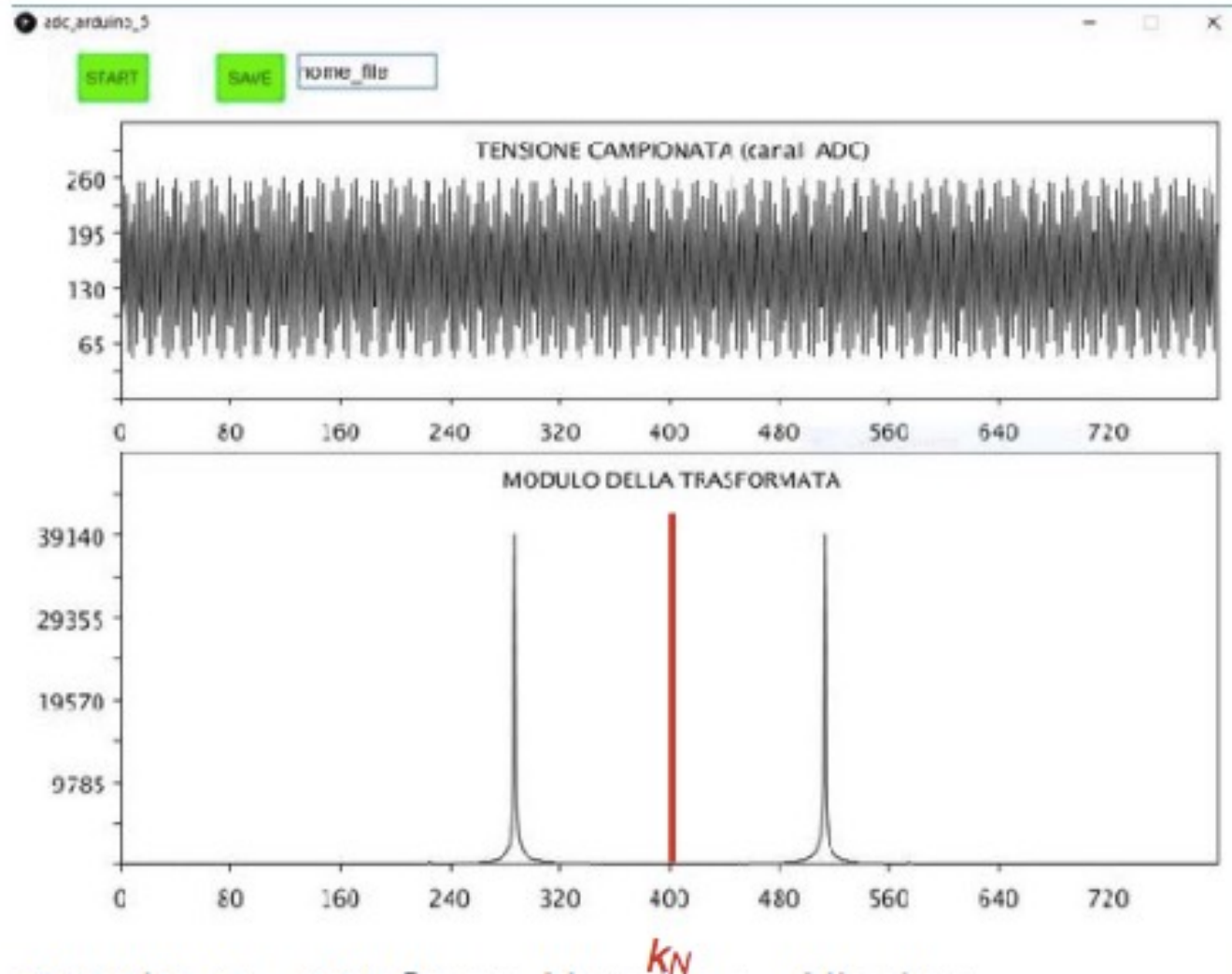
Sinusoide a 4.2 kHz



- $k_{\text{segnale}} = 4200 / 12.5 \text{ Hz} = 336$

Fare questa misura per 3-4 valori di frequenze minori della frequenza di Nyquist e per 3-4 valori al di sopra

Segnale a 6 kHz

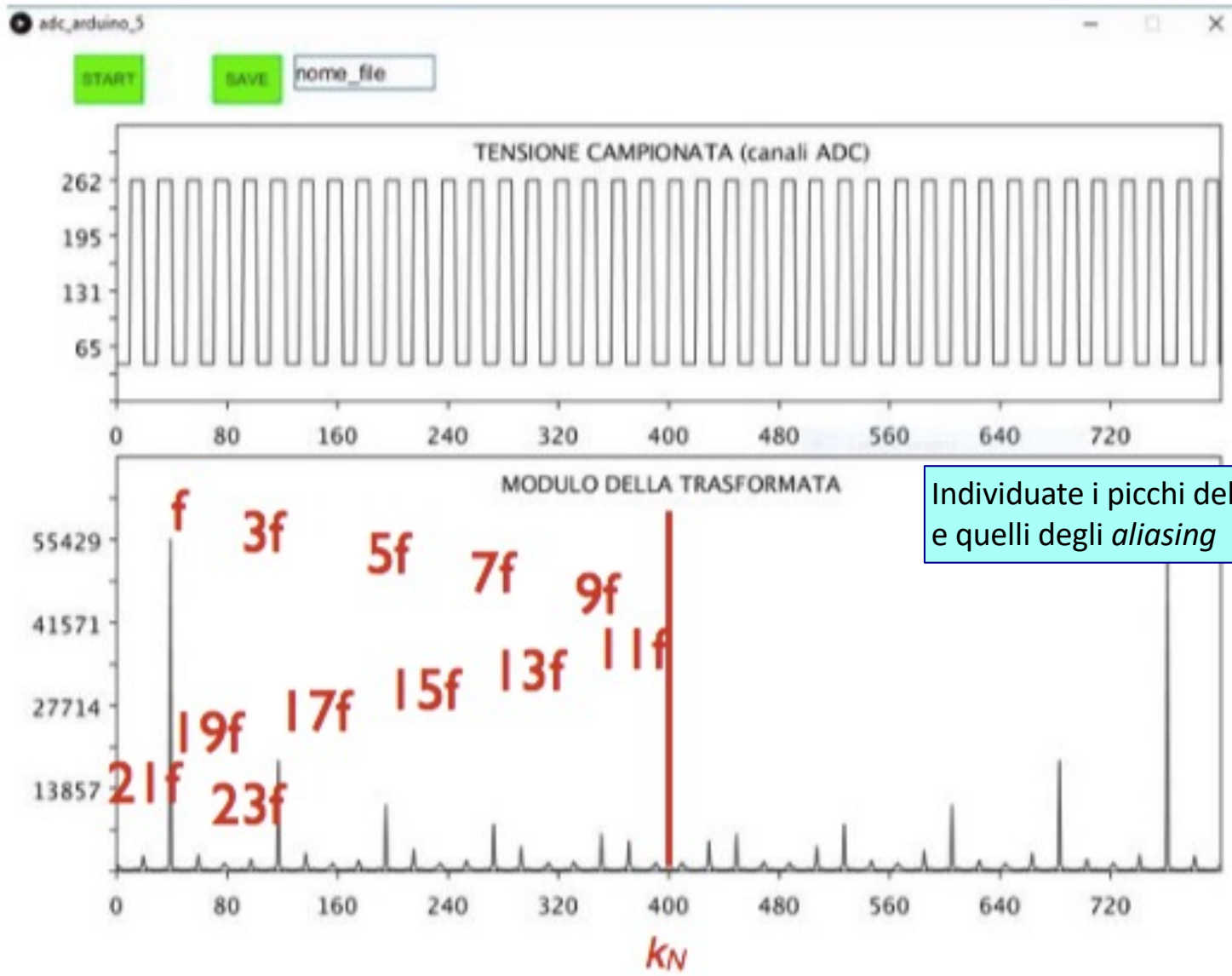


$$f_a = N \times f_N - f_s$$

$$N=2$$

- $k_{\text{segnale}} = 6000 / 12.5 \text{ Hz} = 480$ Sopra Nyquist \rightarrow Aliasing
- $k_{\text{apparente}} = 400 - 80 = 320$ (si tratta come un rimbalzo su k_N)

Onda quadra a 450 Hz



Individuate i picchi delle armoniche e quelli degli *aliasing*

Provate a far passare il segnale attraverso il filtro Butterworth prima di mandarlo ad arduino. Se taglia tutte le armoniche ($f_c=1$ kHz), provate con un'onda quadra di 150 Hz (con e senza filtro)

Spettro di frequenza del rumore

- ❖ È interessante studiare lo spettro di frequenza del rumore generato dal circuito che avete costruito.
- ❖ Data la sua natura casuale è necessario prendere una decina di campioni per poi ricavare lo spettro di rumore S facendo la media dei moduli quadri della trasformata:

$$S_k = \sqrt{\frac{\sum_{m=0}^{M-1} |X_k|^2}{M}}$$

Spettro di potenza del rumore.
Componente k -esima della DFT.
 K va da 1 a 800

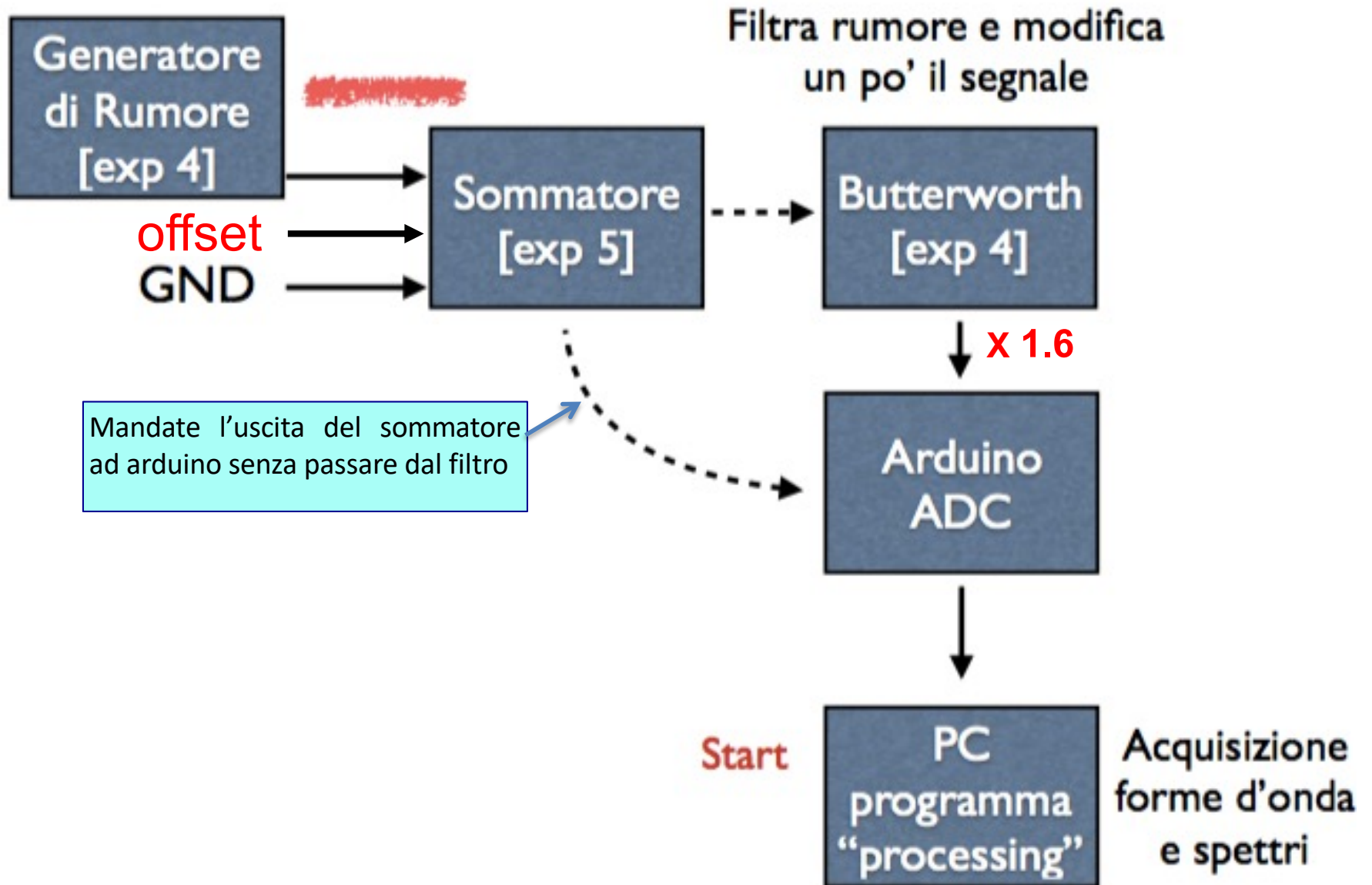
dove M è il numero di segnali acquisiti (ovvero il numero dei file che avete salvato con Processing).

- ❖ Si può procedere salvando ogni campione su un file diverso in formato txt e poi, off-line, fare la media (con un programma in C o in Python).
- ❖ Il formato dei dati salvato da Processing, per ogni punto, è:
 - k , $V(k)$, $\text{Re}(k)$, $\text{Im}(k)$, $\text{mod}(k)$; $\text{mod}(k)$ è il modulo $|X_k|$ della frequenza k -esima
- ❖ Si dovrebbe verificare che questo spettro è sostanzialmente piatto.
- ❖ Collegate poi il segnale al filtro e confrontate i risultati ottenuti.

Il segnale ha una componente negativa, quindi utilizzate il sommatore per aggiungere un offset al rumore e portarlo nel range 0 – 3.3 V.

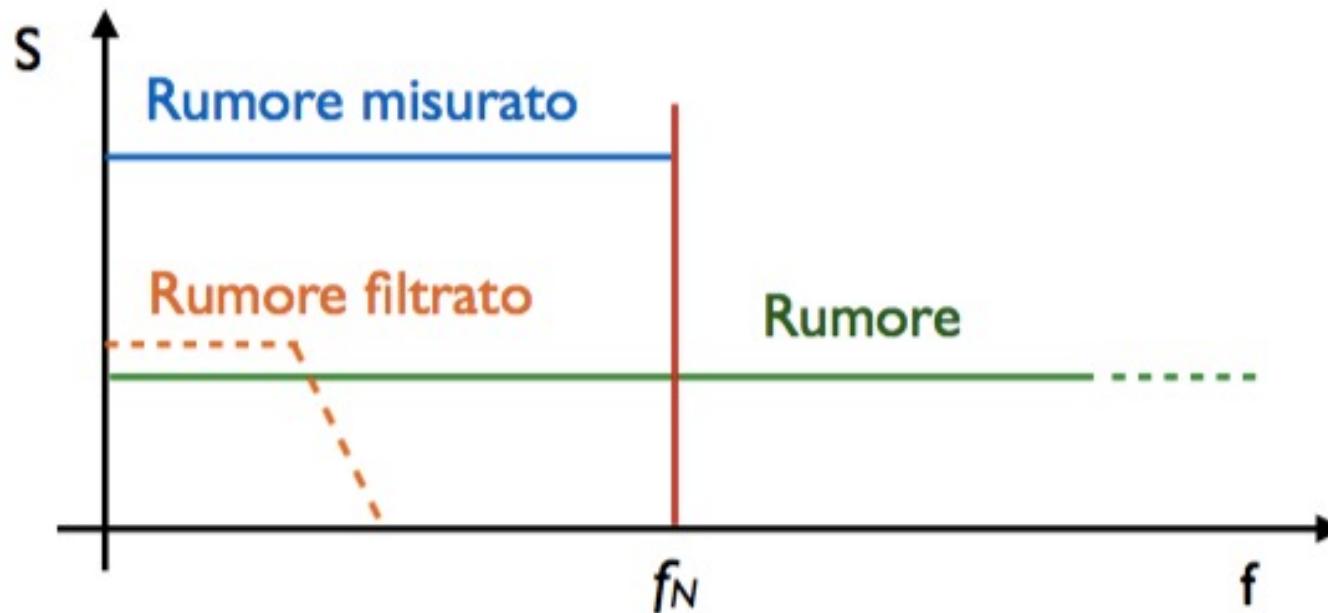
Tenete anche presente che il filtro di Butterworth moltiplica il segnale per 1.6

Misura del rumore



Misura del rumore

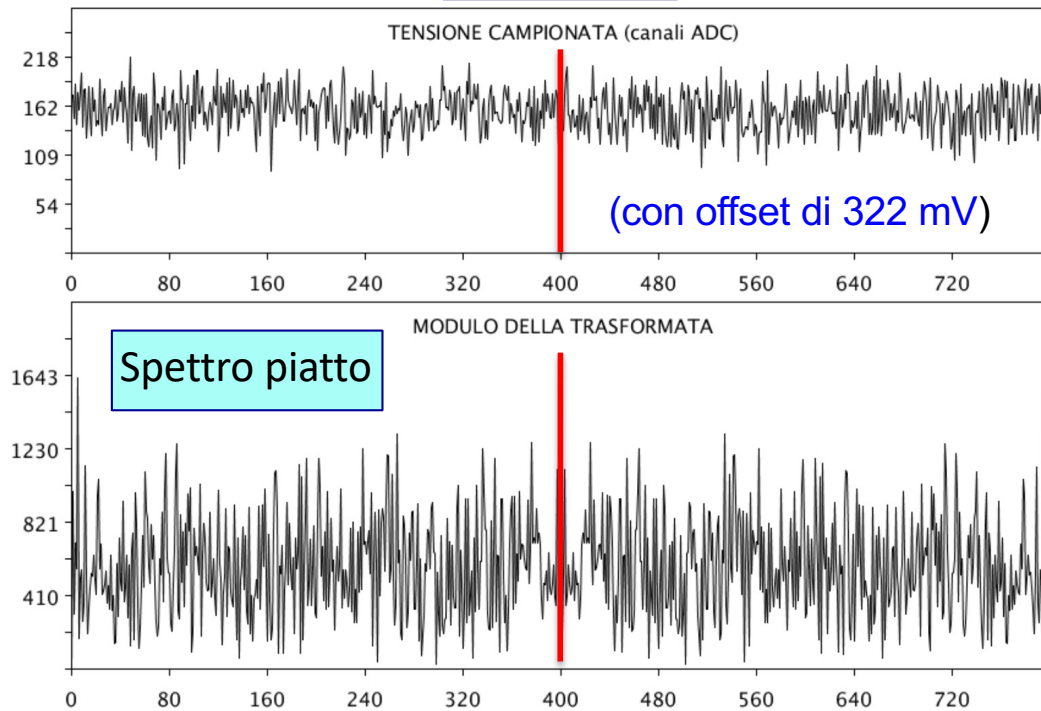
Cosa aspettarsi



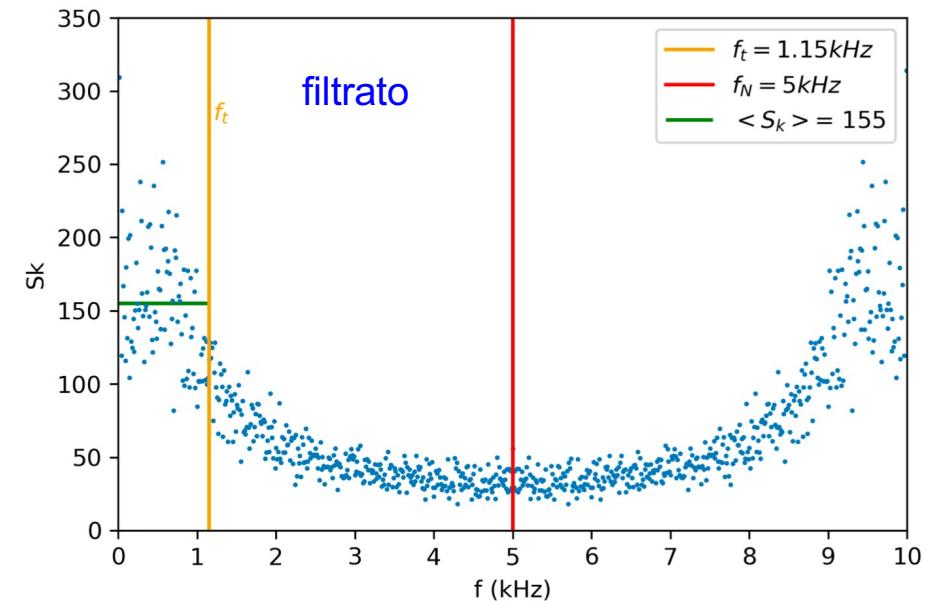
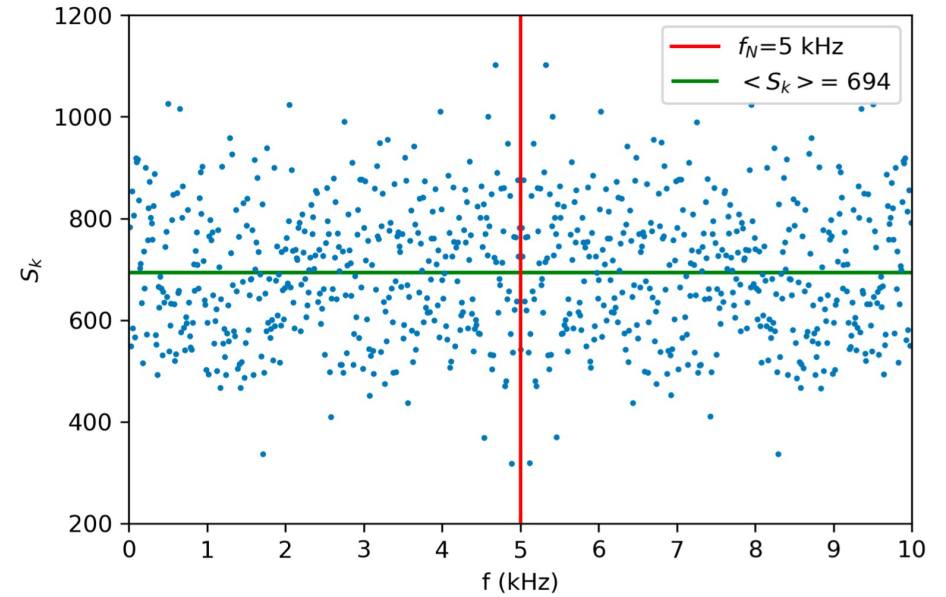
- Perché il rumore misurato ha un livello più alto di quello originale?
- Se avete tempo convertite i grafici da indice k a frequenza, in questo modo riconoscerete facilmente il taglio del Butterworth.

Misura del rumore

Senza filtro



$$S_k = \sqrt{\frac{\sum_{m=1}^M |mod(k)|_m^2}{M}} \quad M = 7$$



Misura del rumore

Senza filtro

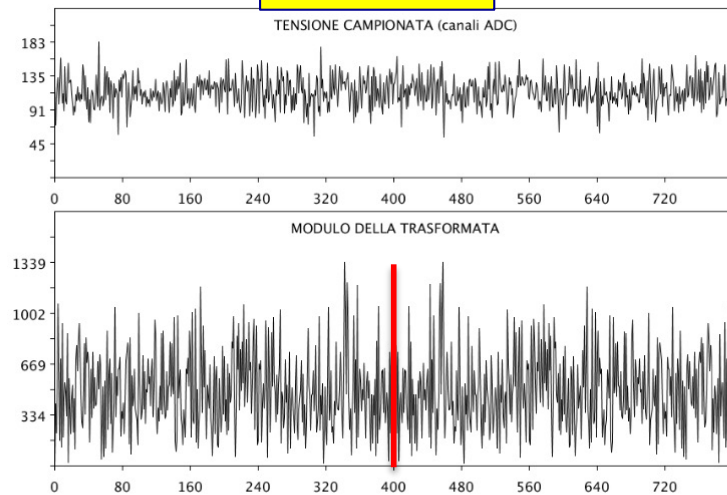


Figura 12: Esempio di rumore ricevuto dall'ADC.

Con filtro

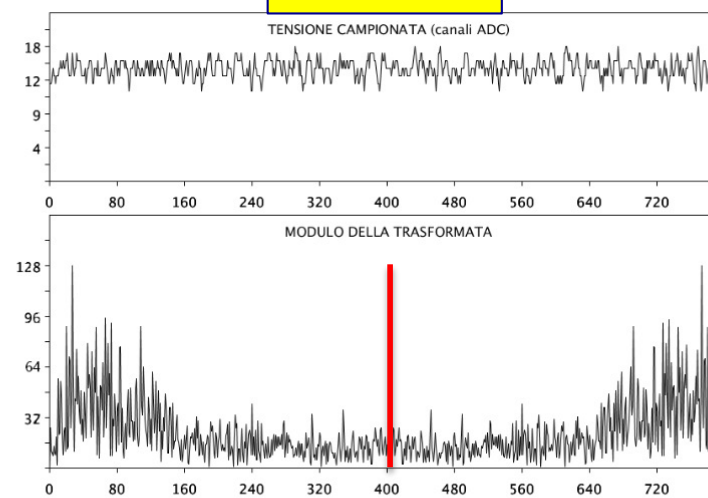


Figura 13: Esempio di rumore filtrato.

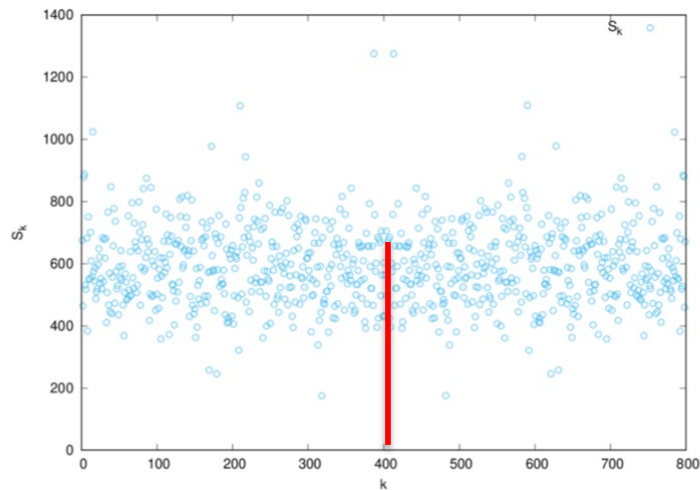


Figura 14: RMS di M campionamenti.

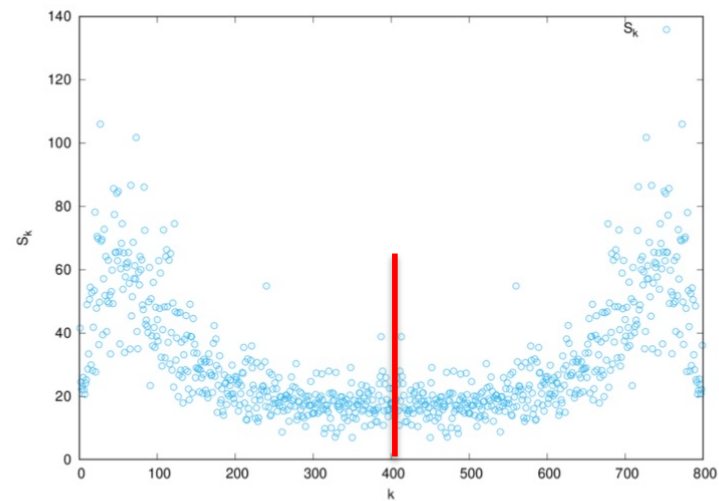
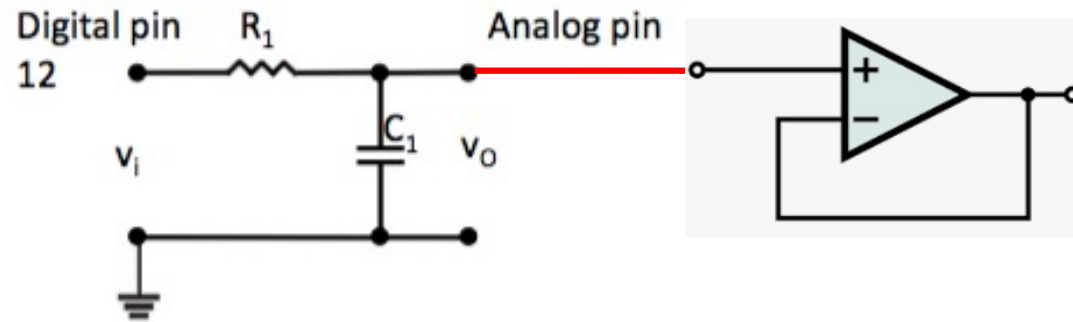


Figura 15: RMS di M campionamenti con rumore filtrato.

Segnale impulsivo

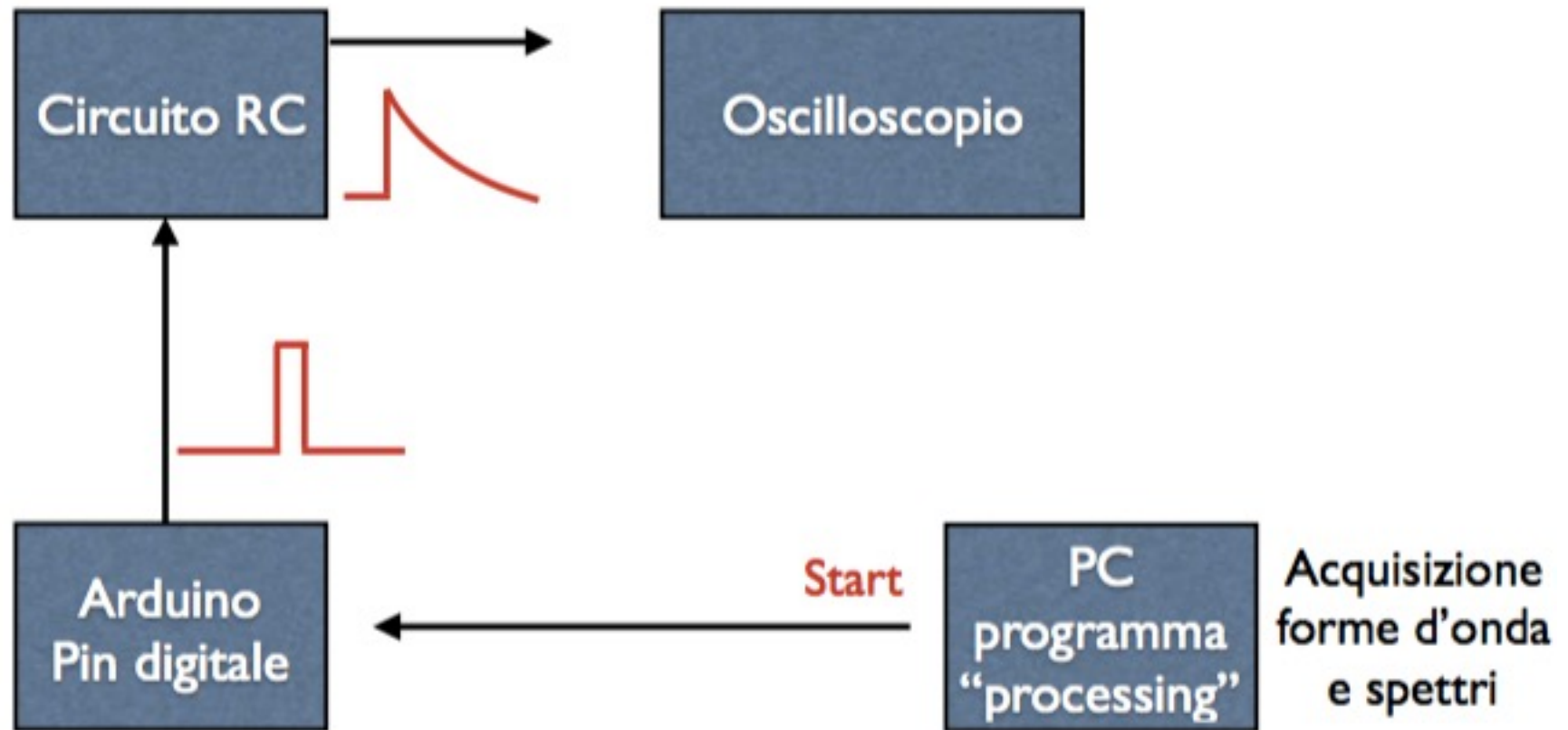
R=100 kΩ
C=100 nF
tau=10 ms



Analog Pin di Arduino
oppure sommatore

- ❖ Utilizzeremo Arduino per produrre un segnale impulsivo digitale che invieremo ad un circuito RC con costante di tempo di circa 10 ms.
- ❖ Per non alterare la costante di tempo del circuito quando lo si connette al sommatore, è consigliato collegare l'uscita di questo circuito ad un emitter follower realizzato con un op-amp (potete utilizzare lo stesso circuito integrato del sommatore)
- ❖ `adc_read_5_2019` genera un segnale impulsivo di forma quadrata della durata di circa 2 ms
- ❖ **Analizziamo ora il responso del circuito utilizzando i programmi `adc_read_5_2019` e `adc_arduino_5`:**
 1. Osserviamo dapprima il segnale prodotto da Arduino collegando il pin di uscita digitale 12 ad uno degli ingressi analogici (pin 3 di default)
 2. Successivamente colleghiamo l'uscita del circuito RC ad Arduino ed osserviamo la risposta del circuito ed acquisiamo una forma d'onda di riferimento.
 3. Infine aggiungeremo il rumore al segnale e cercheremo di ridurlo utilizzando il filtro

Creazione di un segnale impulsivo



- Controllare il segnale in uscita dal pin digitale di Arduino.
- Controllare il segnale in uscita dal circuito RC ($100 \text{ k}\Omega \times 100 \text{ nF} = 10 \text{ ms}$)

Ricordatevi di collegare il ground di Arduino al ground della basetta

Forma attesa del segnale impulsivo

- Da Arduino esce un'onda quadra molto stretta, circa $\Delta T = 2$ ms, con ritardo di circa 10 ms rispetto allo start dell'acquisizione ADC.
- Possiamo rappresentarlo come la sovrapposizione di 2 onde quadre e applicare Laplace:

$$f(t) = A[\theta(t) - \theta(t - \Delta t)] \quad f(s) = A \left[\frac{1}{s} - \frac{1}{s} e^{-s\Delta t} \right]$$

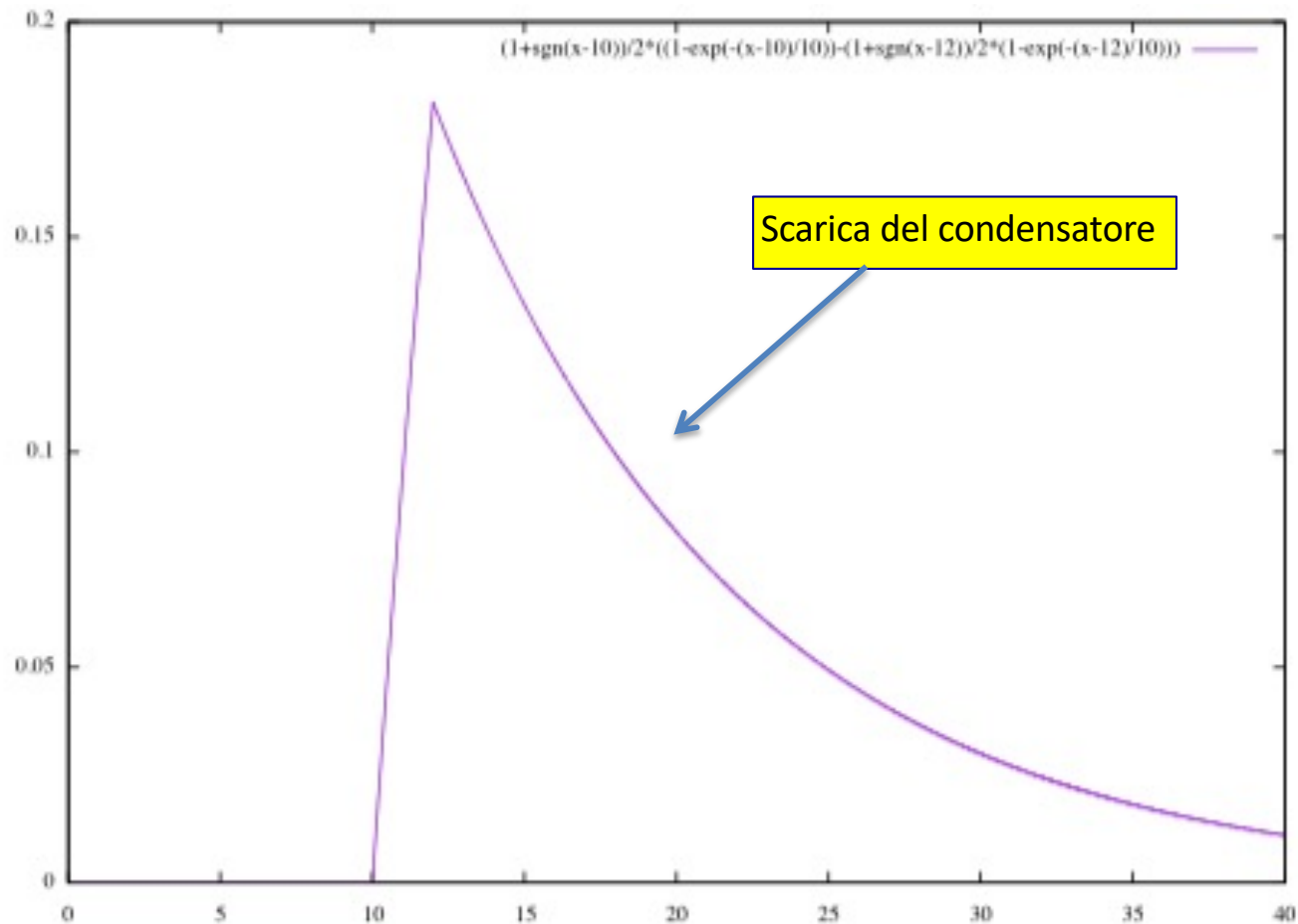
- Moltiplicando per la funzione di trasferimento del passa basso.

$$P(s) = f(s) \frac{1}{1 + s\tau}$$
$$P(t) = A\theta(t)[1 - e^{-t/\tau}] - A\theta(t - \Delta t)[1 - e^{-(t-\Delta t)/\tau}]$$

- Quindi in pratica l'impulso è:

$$P(t) = \begin{cases} A[1 - e^{-t/\tau}], & \text{if } t < \Delta t \\ A[e^{\Delta t/\tau} - 1]e^{-t/\tau}, & \text{if } t \geq \Delta t \end{cases}$$

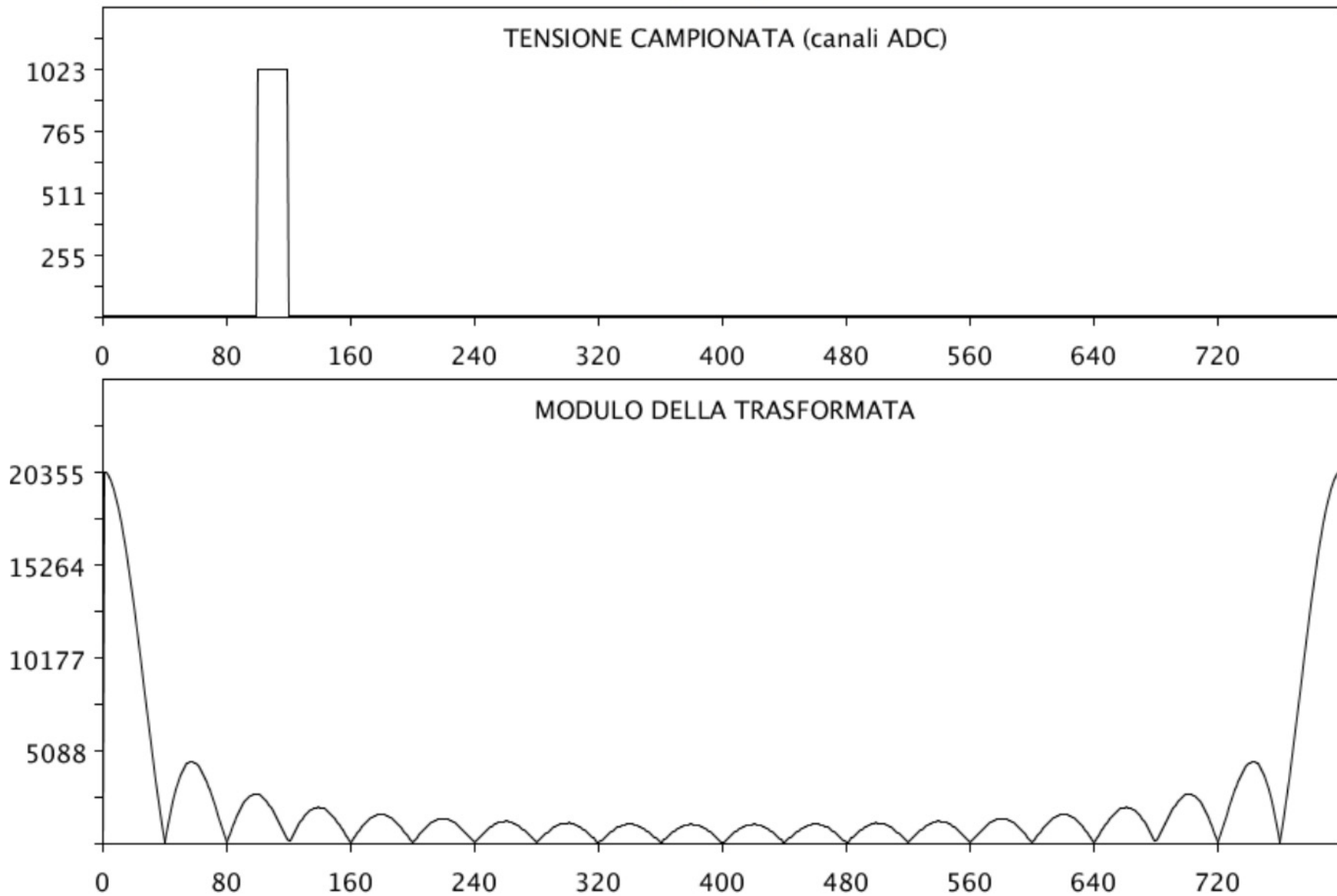
segnale impulsivo simulato



- Il massimo vale $A [1 - e^{-t/\Delta t}] = 0.18 A$

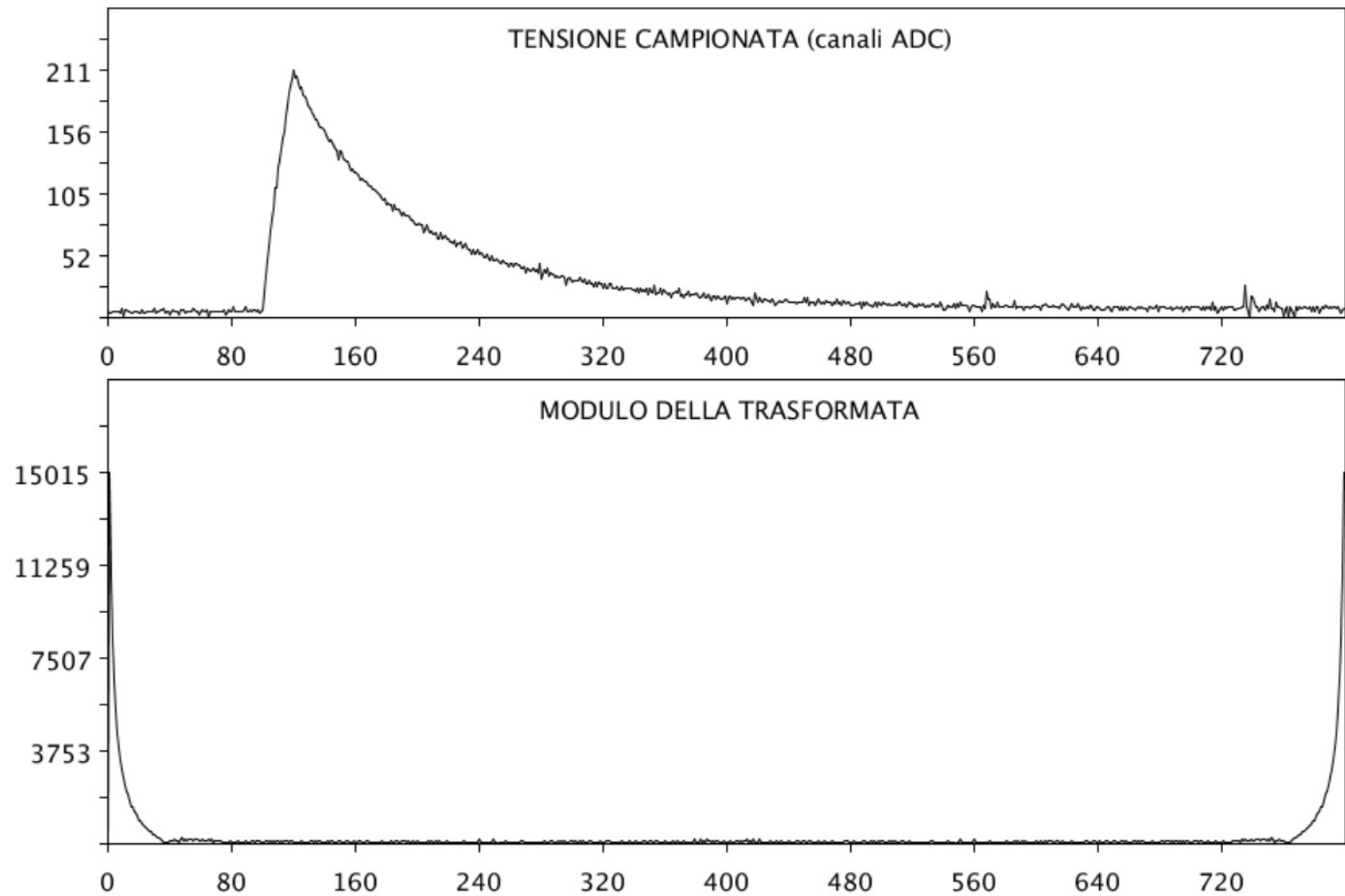
Segnale impulsivo

Onda rettangolare impulsiva



Segnale all'uscita del circuito RC

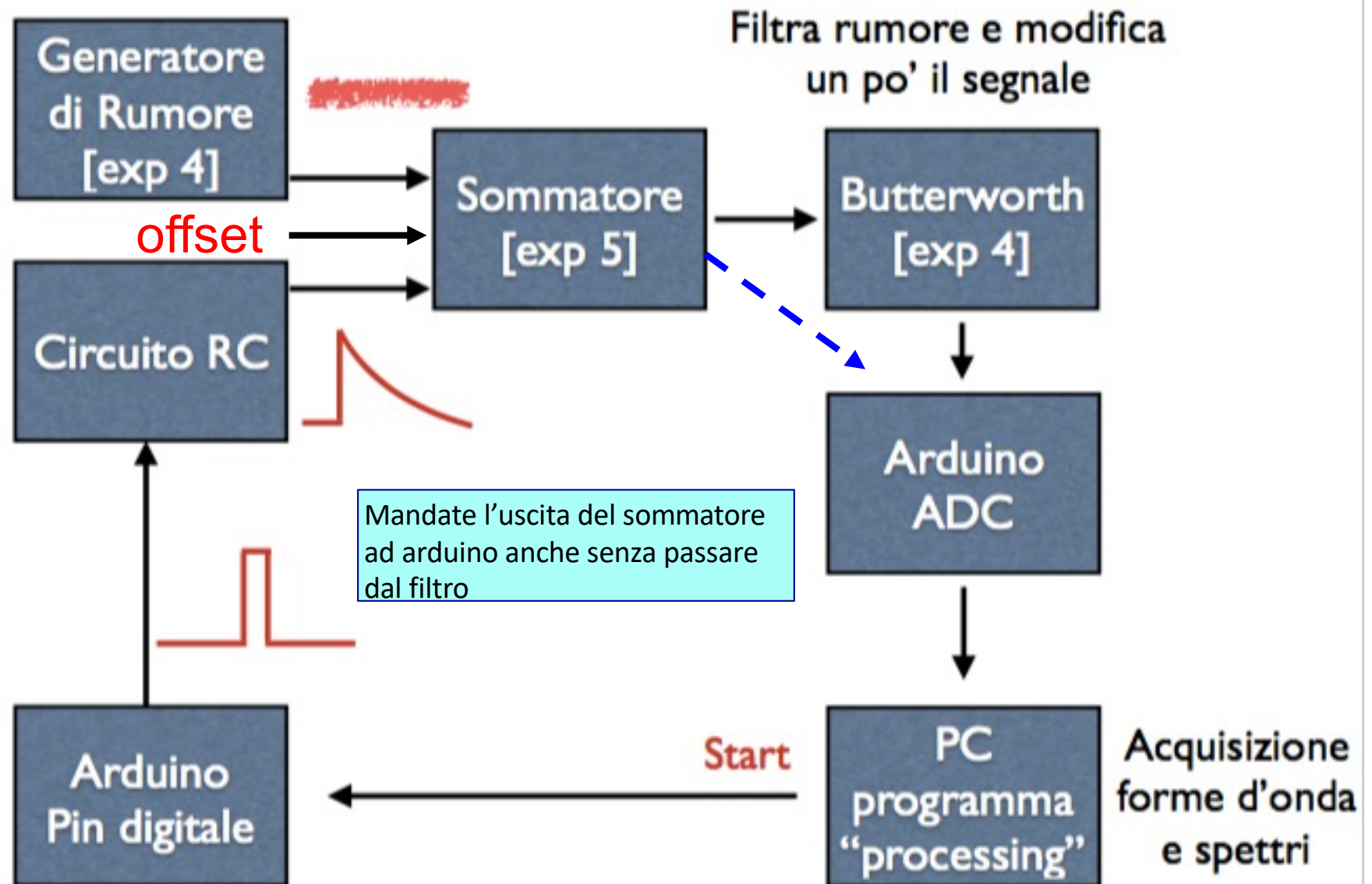
Impulso con filtro RC



Misure segnale + rumore

- ❖ Possiamo ora simulare una situazione reale aggiungendo al segnale impulsivo una certa quantità di rumore utilizzando il sommatore a 3 ingressi (se il rumore aggiunto vi sembra troppo basso potete aumentare il suo effetto agendo sui pesi (resistenze) del sommatore) e studiare lo spettro in frequenza che ne risulta
- ❖ Nelle situazioni reali si cerca di ridurre il più possibile il rumore cercando di alterare il meno possibile la forma del segnale, trovando il giusto compromesso tra queste due esigenze, mediante fitri, che vengono progettati studiando appunto lo spettro in frequenza del rumore e quello del segnale. Possiamo cercare di fare lo stesso inserendo nel circuito il filtro passa basso VCVS che abbiamo costruito.
- ❖ Osservando il segnale all'oscilloscopio possiamo valutare il risultato che otteniamo. Colleghiamo poi il circuito ad Arduino e osserviamo il risultato della misura.

Circuito completo

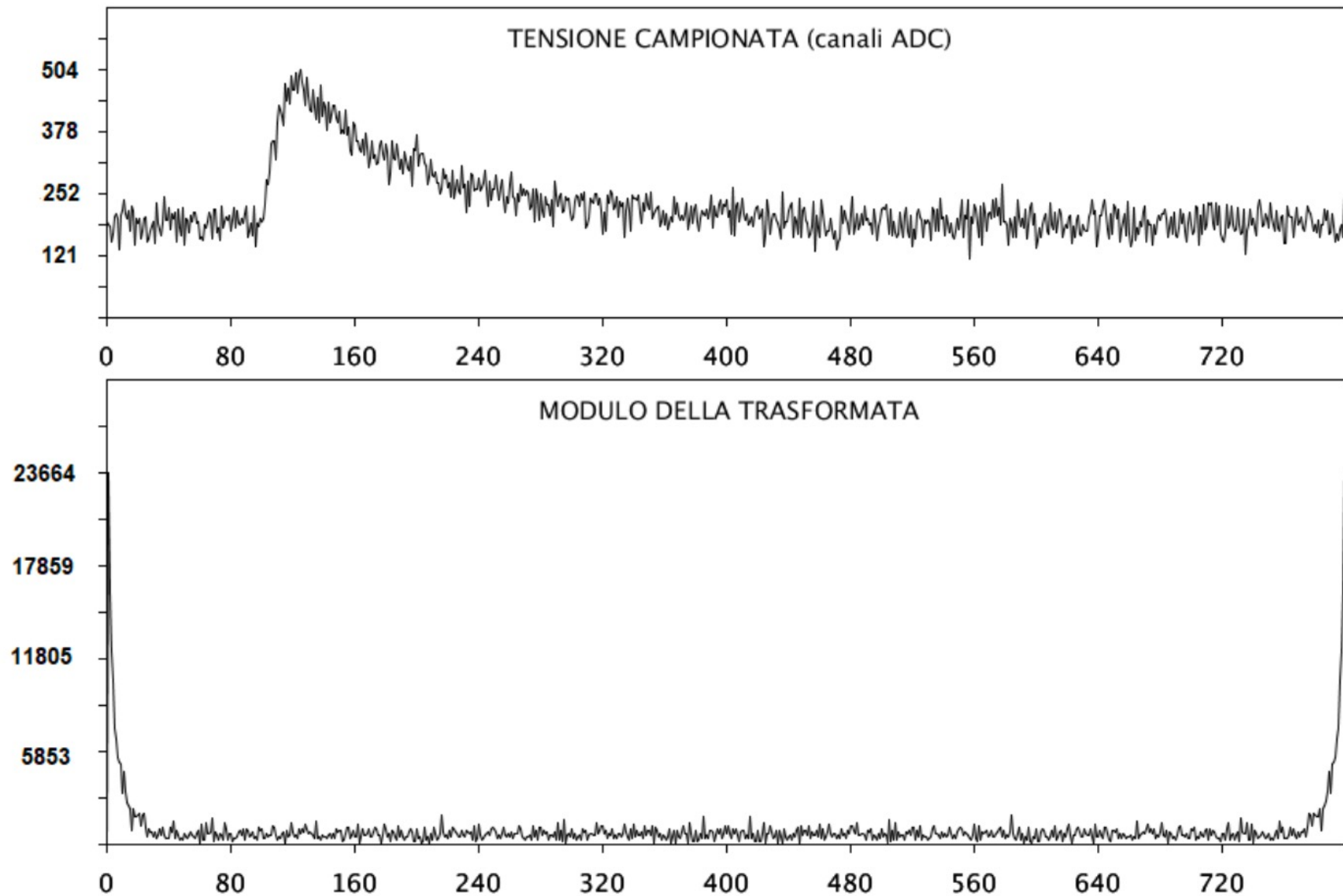


Misure segnale + rumore

- ❖ Analogamente a quanto fatto per il rumore, studiate lo spettro in frequenza del segnale da solo, staccando il rumore di sommatore e mettendolo a massa:
 - Effettuate questa misura con e senza filtro passa basso.
- ❖ Attaccate poi segnale e rumore al sommatore, con l'eventuale offset per avere sempre il segnale nel range 0-3.3 V dell'ADC di Arduino:
 - Controllate l'uscita del sommatore con l'oscilloscopio
 - Acquisite le forme d'onda con Arduino
 - Effettuate questa misura con e senza filtro passa basso.

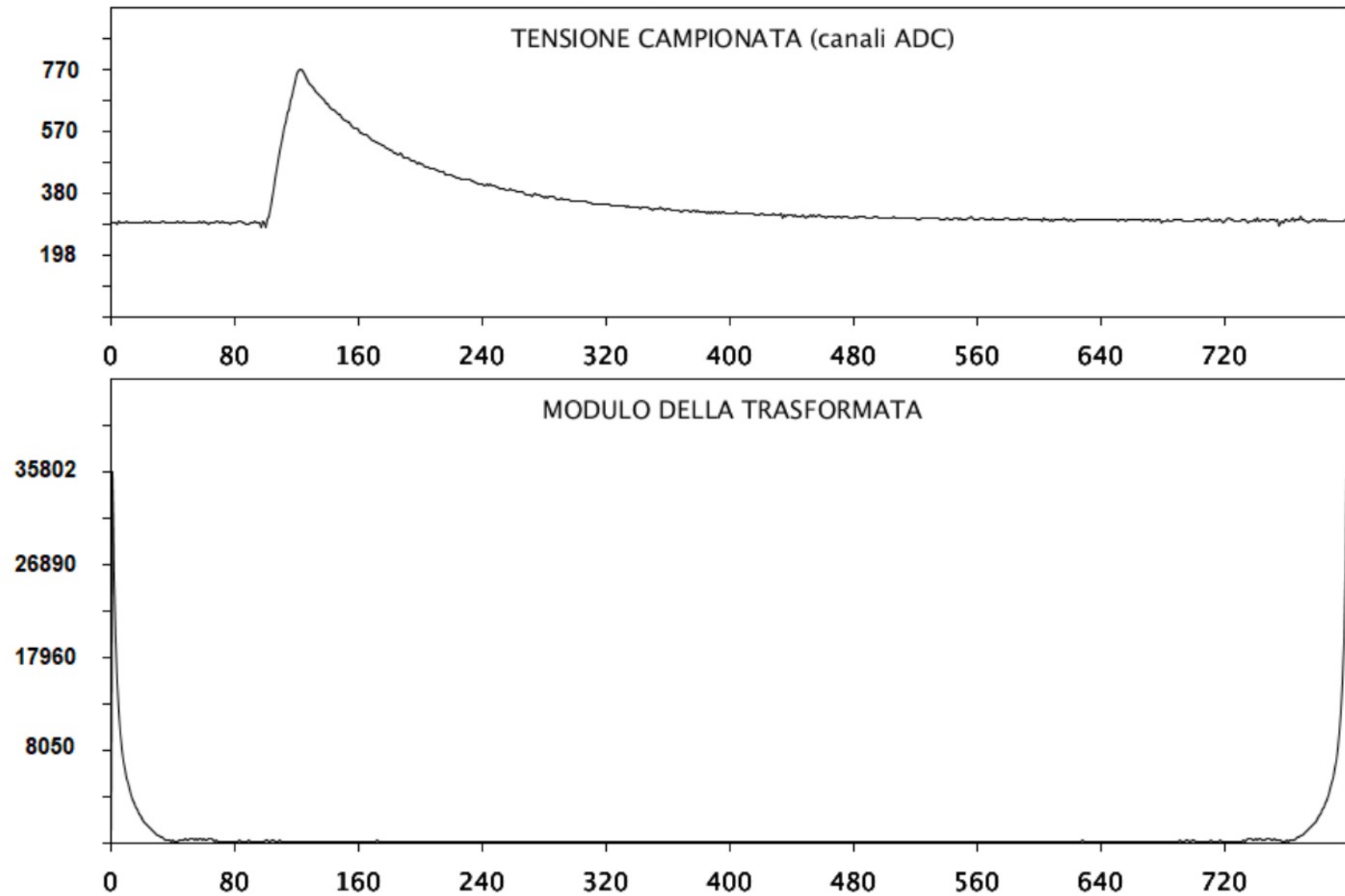
Segnale impulsivo con rumore non filtrato

Segnale impulsivo con rumore non filtrato

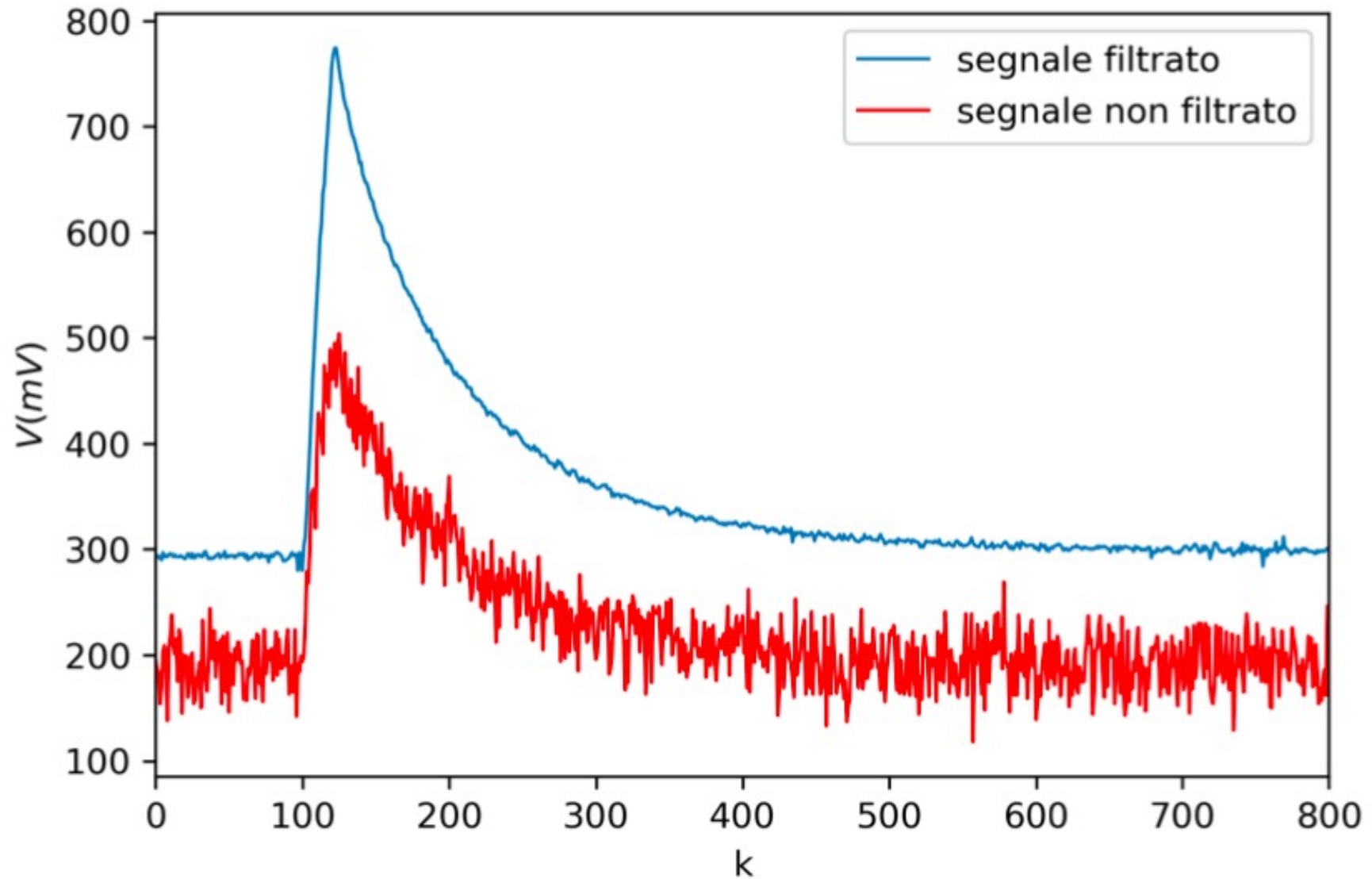


Segnale impulsivo con rumore filtrato

Segnale impulsivo con rumore filtrato



Segnale impulsivo con rumore



Memento: il segnale filtrato è moltiplicato per 1.6



SAPIENZA
UNIVERSITÀ DI ROMA

Fine esercitazione 9