Laboratorio di Segnali e Sistemi
- Esercitazione -9 -

DFT con Arduino

Claudio Luci

SAPIENZA
UNIVERSITÀ DI ROMA

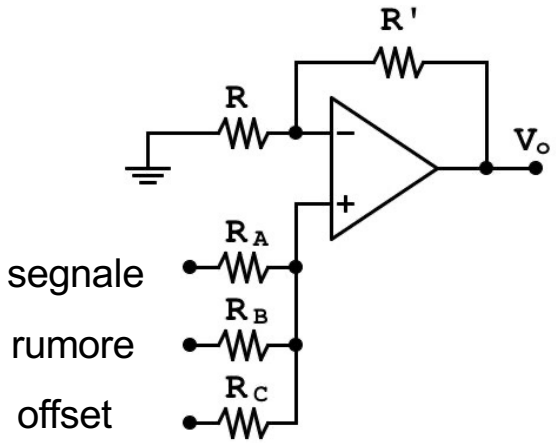
last update : 070117

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

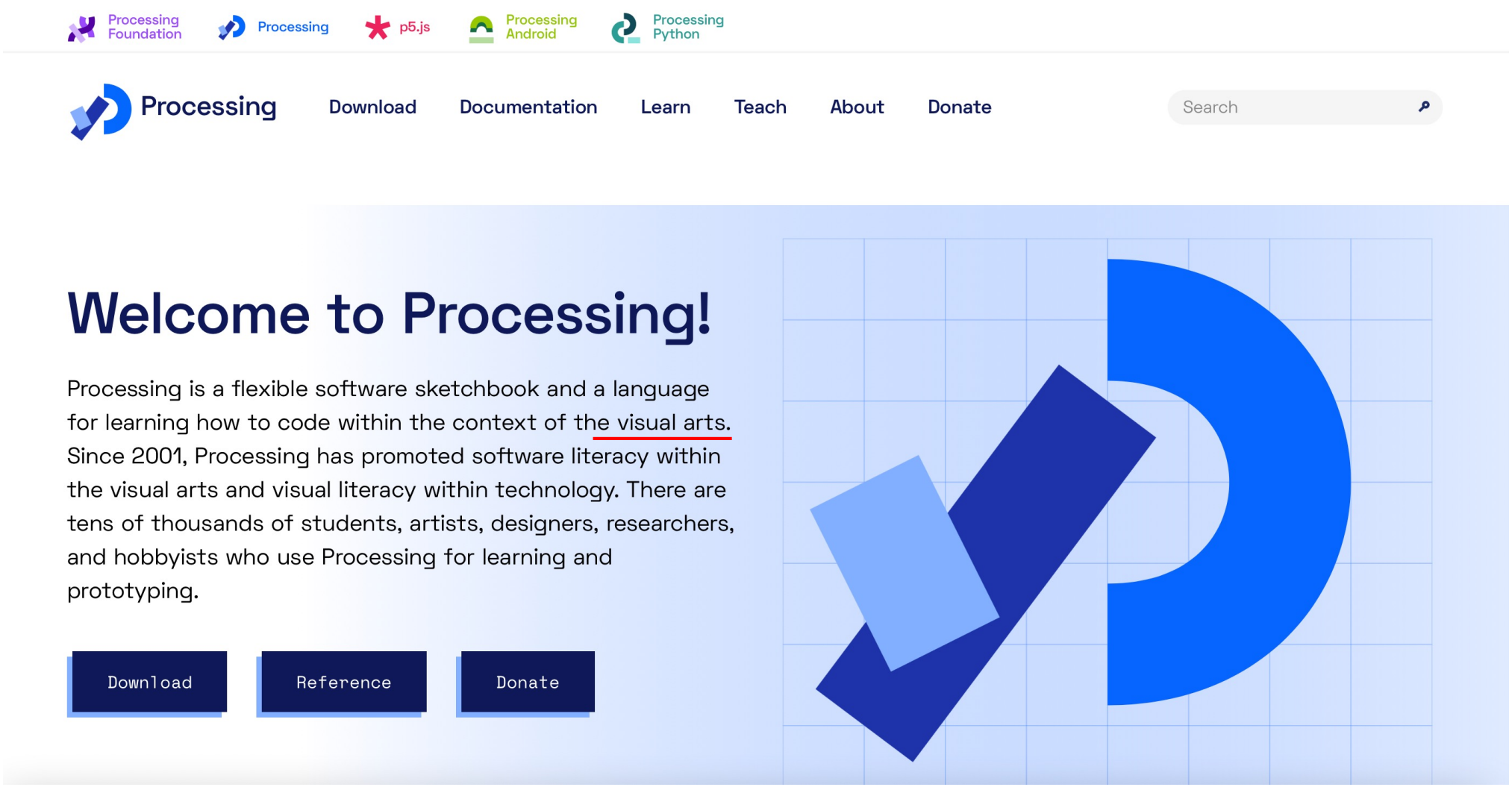
- Per Arduino utilizzeremo lo sketch *adc_read_5_2019* (che potete scaricare dal mio sito), scritto per lavorare in congiunzione con lo sketch di Processing *adc_arduino_5.pde*, che trovate sul mio sito.

- 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 verrano 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 µs 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

https://processing.org



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

Arduino e Processing

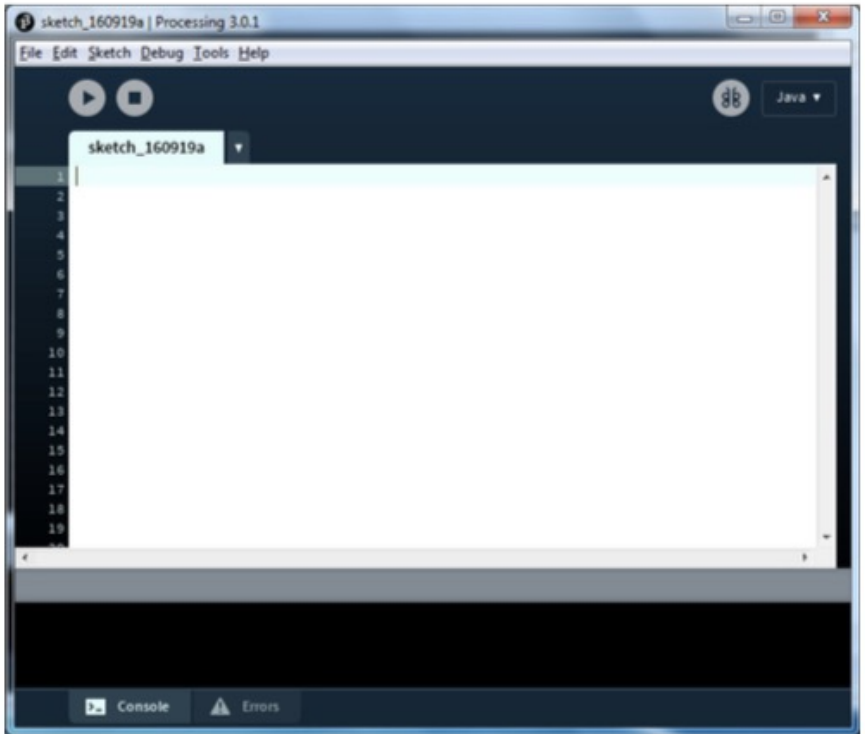
- Processing è un'applicazione che consente di sviluppare contenuti interattivi. Eredita la sintassi e i comandi di programmazione a oggetti (object oriented program) dal linguaggio Java ma in aggiunta mette a disposizione numerose funzioni ad alto livello per gestire gli aspetti grafici e multimediali.
- 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 trovare 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 istallata scaricate dal mio sito il file *ControlP5.zip*, decomprimetelo e istallatelo nella cartella Documenti/Processing/Libraries.
- Voi avete bisogno dello sketch di Processing *adc_arduino_5.pde*. Nel caso non fosse gia disponibile nel vostro PC, potete scaricare dal mio sito il file *adc_arduino_5.zip*.
  Decomprimetelo e istallate la cartella *adc_arduino_5* nella cartella Documenti/Processing.
  Essa contiene il file in questione.
- Se non fosse gia 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.

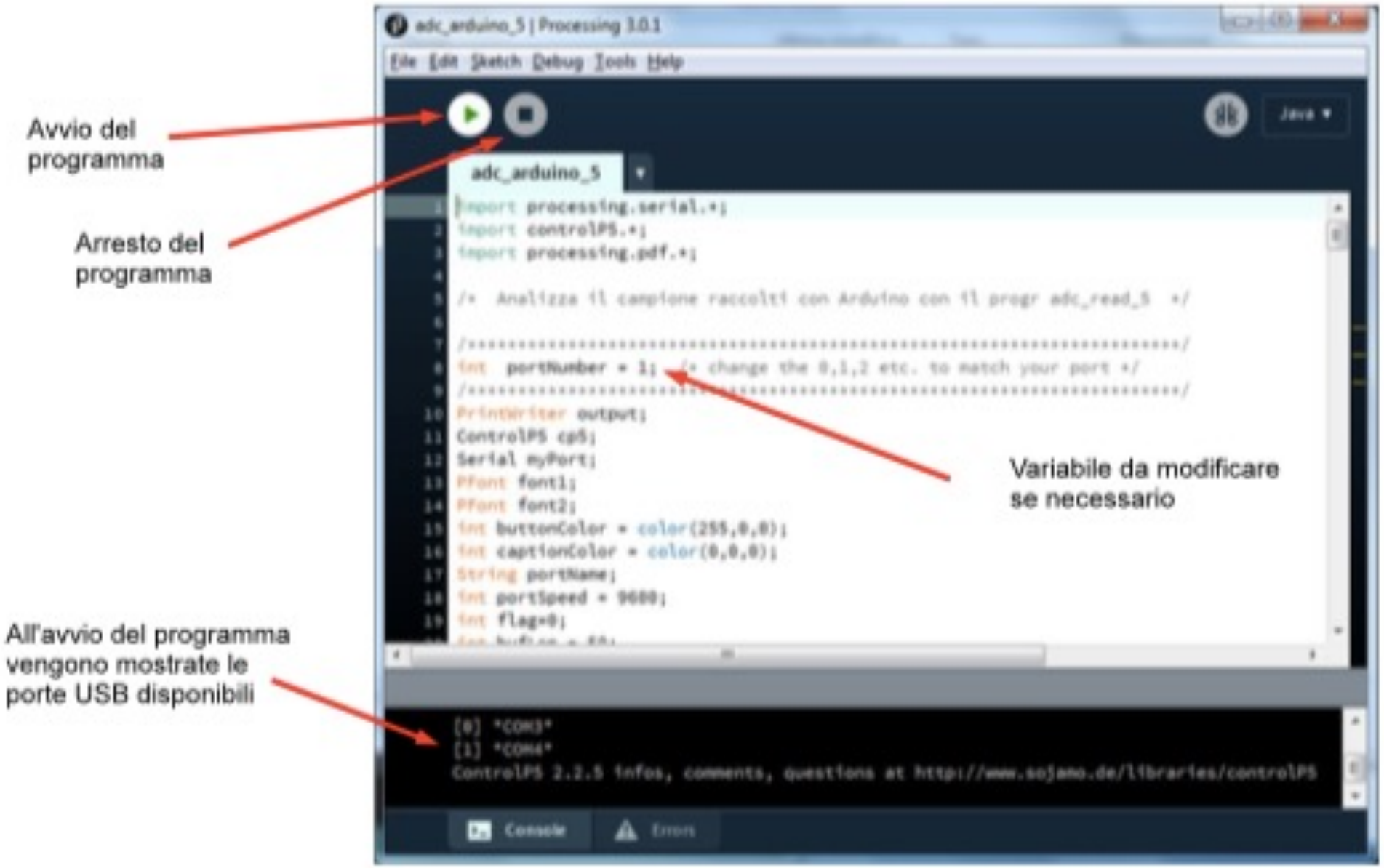
- Questo programma è pronto per essere utilizzato: premendo il bottone di avvio si apre la finestra grafica di interfaccia (vedi figura successiva).

- 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.

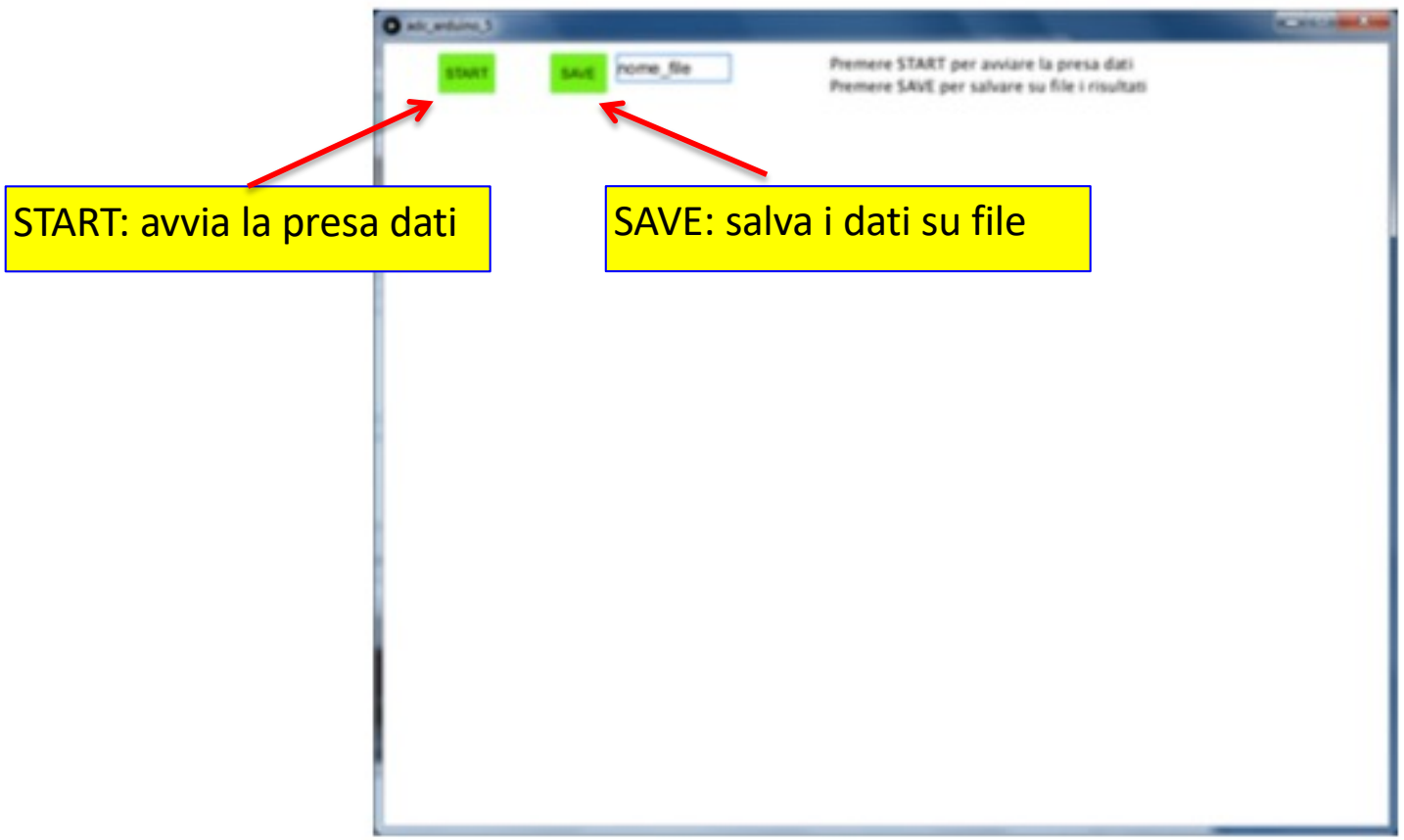
- Ora potete avviare la presa dati con il bottone START.
  Viene inviato al microcontrollore il comando di inizio campionamento e i dati raccolti vengono restituiti a *adc_arduino_5* che produce due grafici, quello del campione e quello della trasformata.
  Come esempio potete vedere un'onda quadra con frequenza di 500 Hz.

- È possibila 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

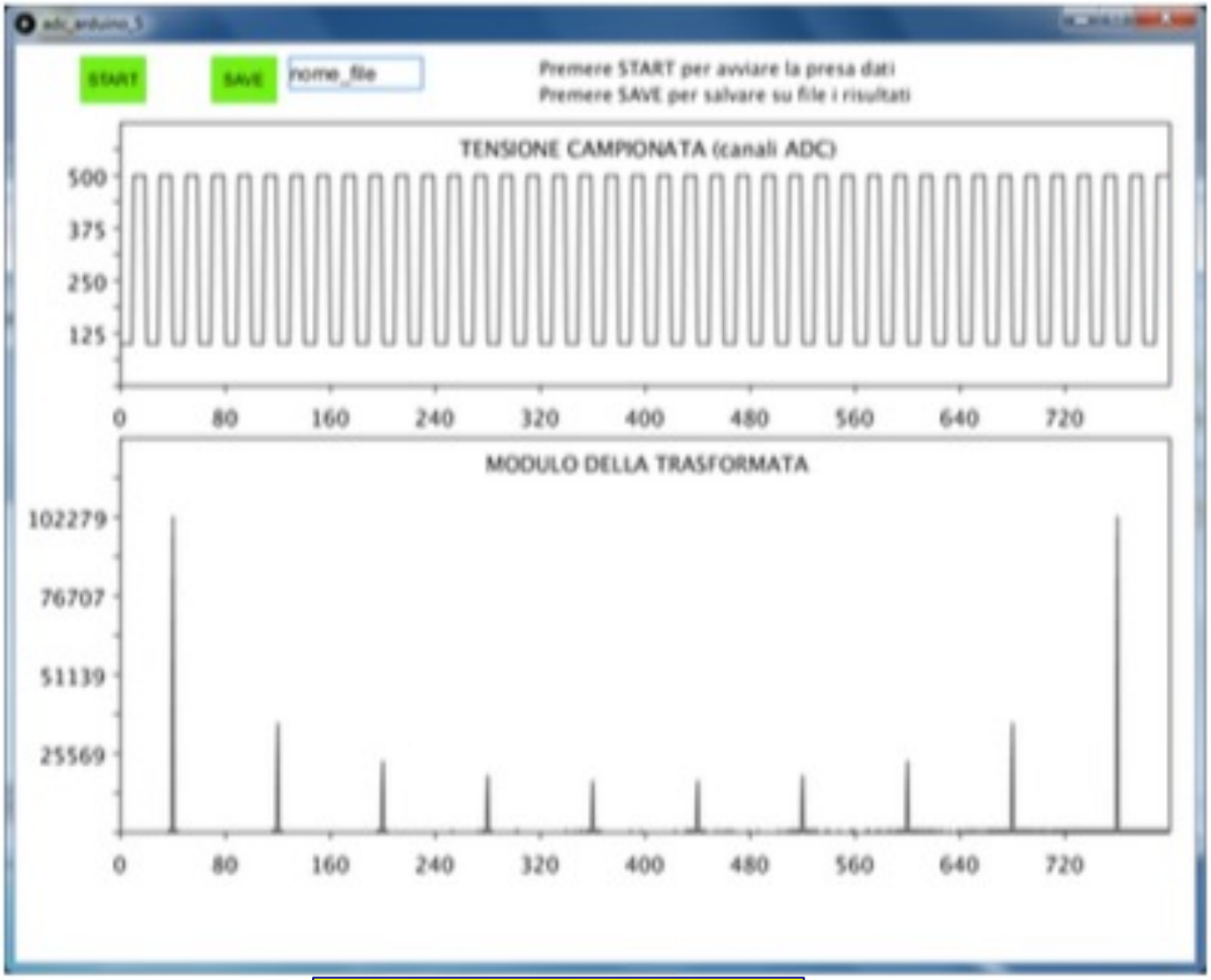


Interfaccia grafica di adc_arduino_5



- 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;
       delayMicroseconds(95);   // to go to 10000Hz sampling (take into account 5 us time transmission)
        /* Serial.println(value); */
       }
/* Invia i dati a Processing */
```

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

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 fenomento dell'*aliasing* utilizzando segnali sinusoidali con frequenza superiore a quella di Nyquist, verificando la validità della relativa formula teorica.

Onda quadra

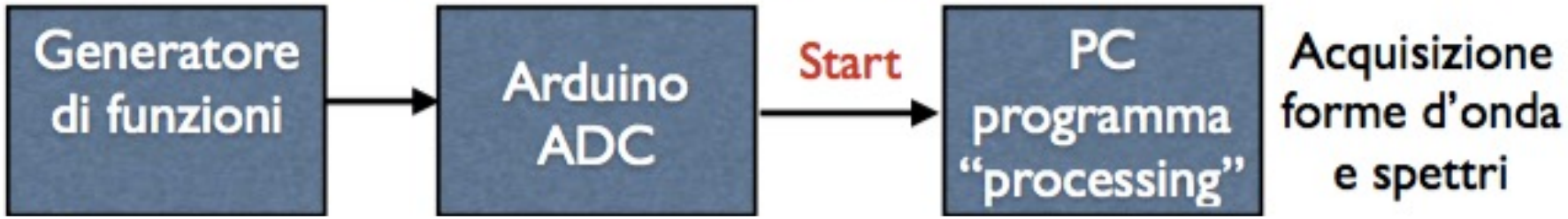
- Studiate poi lo spettro in frequenza di un'onda quadra (con una frequenza inferiore alla frequenza di Nyquist) e verificate che esso corrisponda a quanto atteso teoricamente, individuando i picchi delle armoniche e quelli dovuti all'*aliasing*.

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.



Lo start viene dato da processing

Sinusoide a 200 Hz

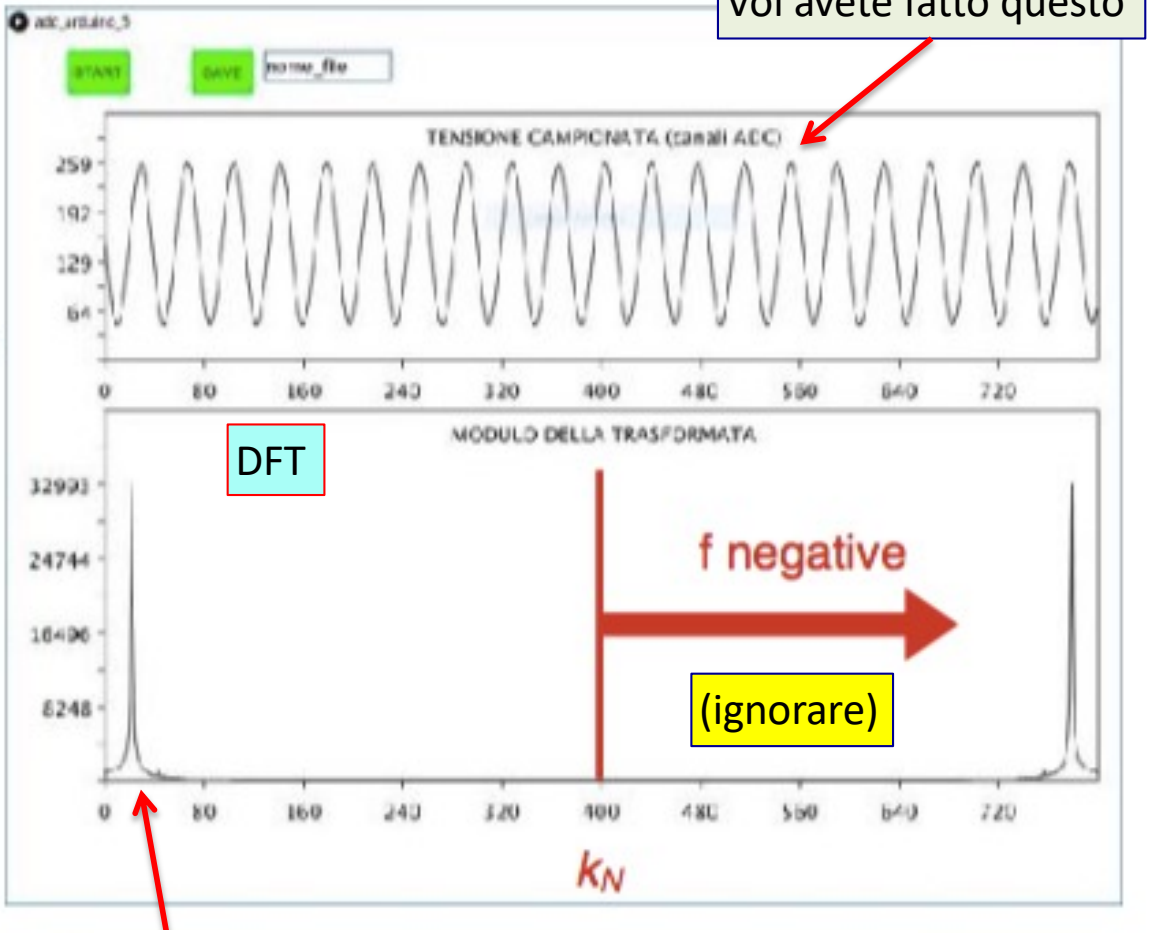
- Caratteristiche Arduino
  - 1024 conteggi, 0-3.3 V
  - 800 campionamenti a circa 10 kSPS

Sample Per Second

- Pertanto

10 bit

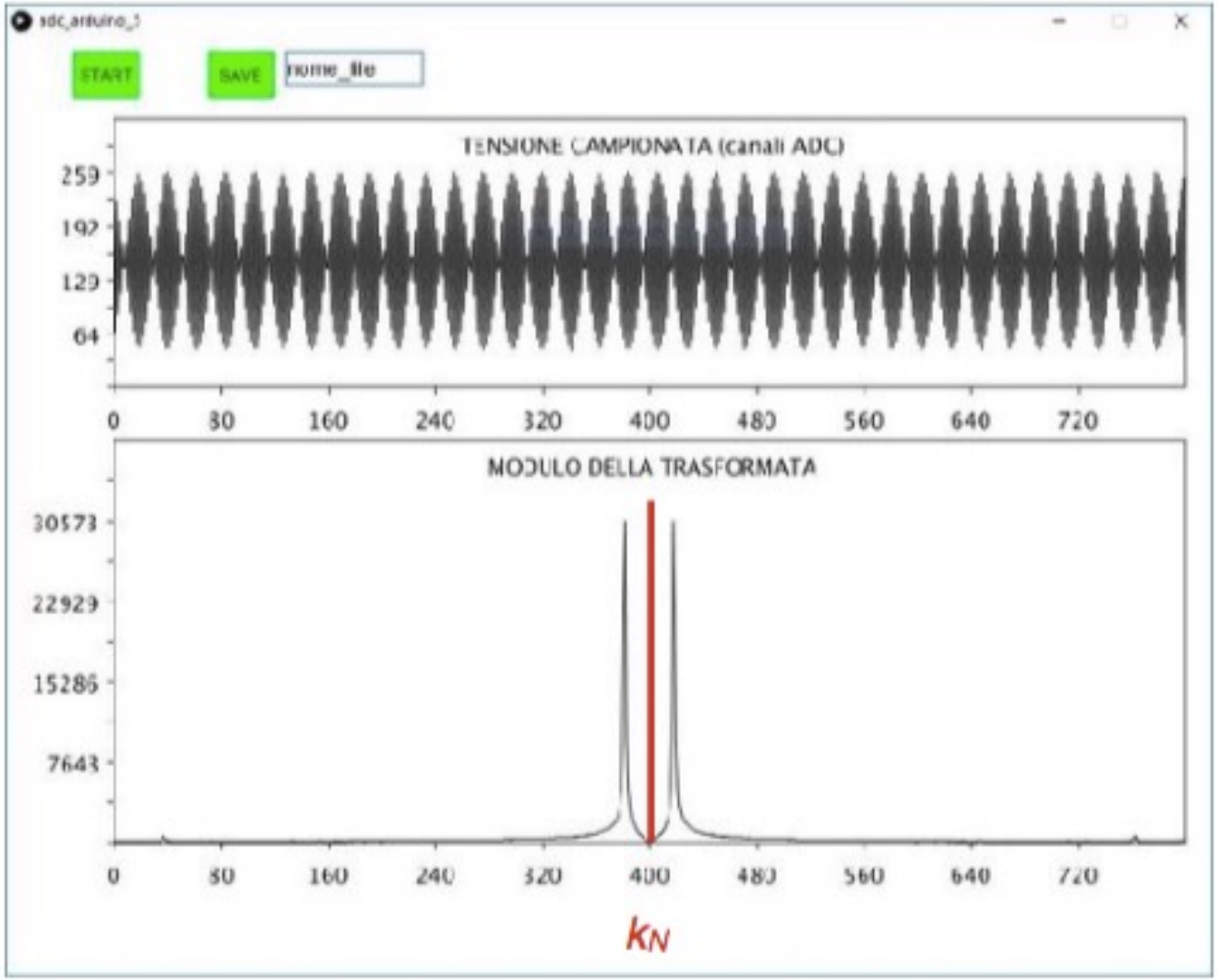
  - Volt / count = 3.3 / 1024 = 3.22 mV / count
  - N = 800
  - f_s = 10 kHz
  - T = N / f_s = 80 ms
  - $\Delta t = 1 / f_s$ = 100 µs
  - $\Delta f = f_s / N$ = 12.5 Hz
  - f_N = 5 kHz
  - k_N = 400



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

f=K × 12.5 Hz

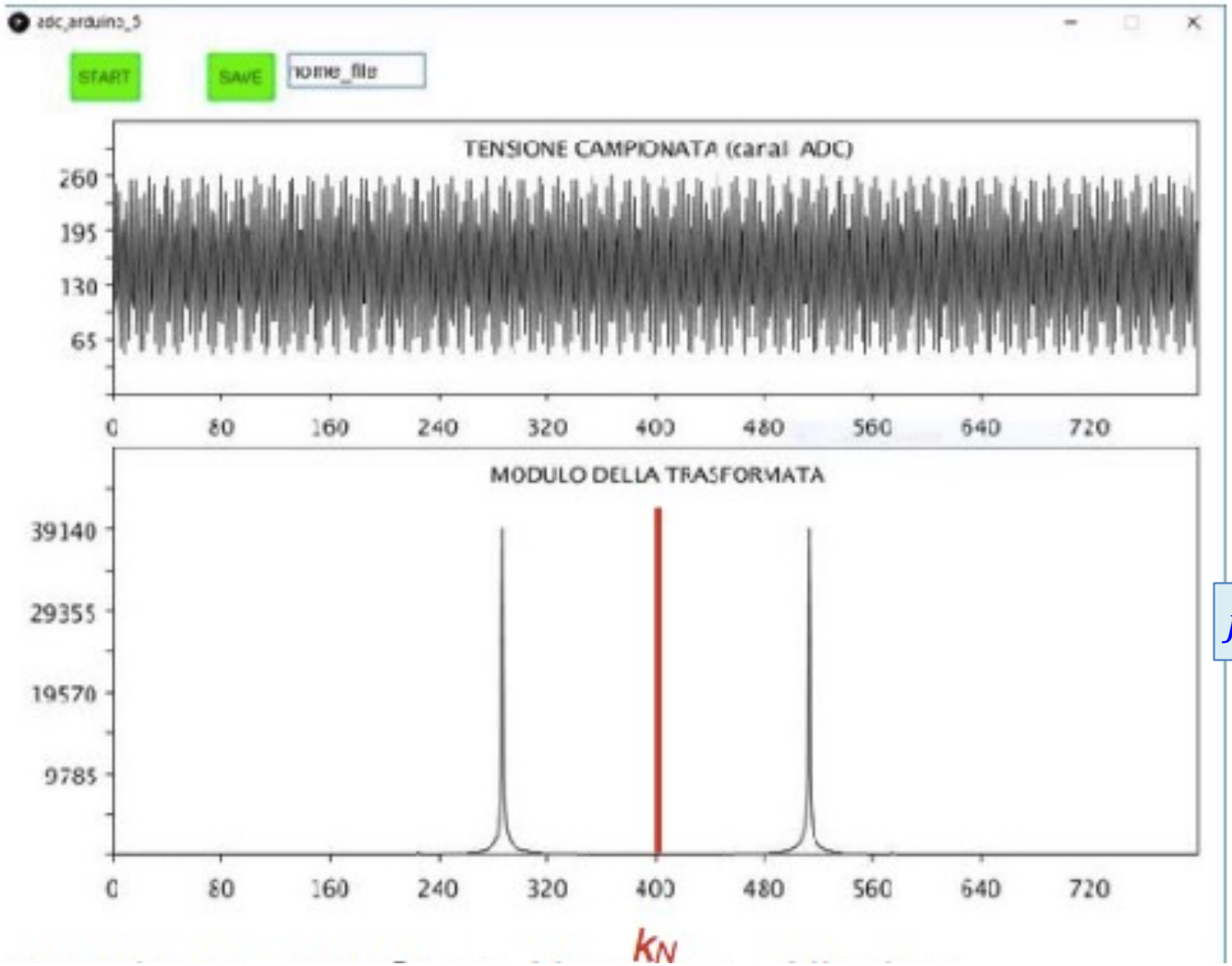
Sinusoide a 4.2 kHz



- $k_{segnale}$ = 4200 / 12.5 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

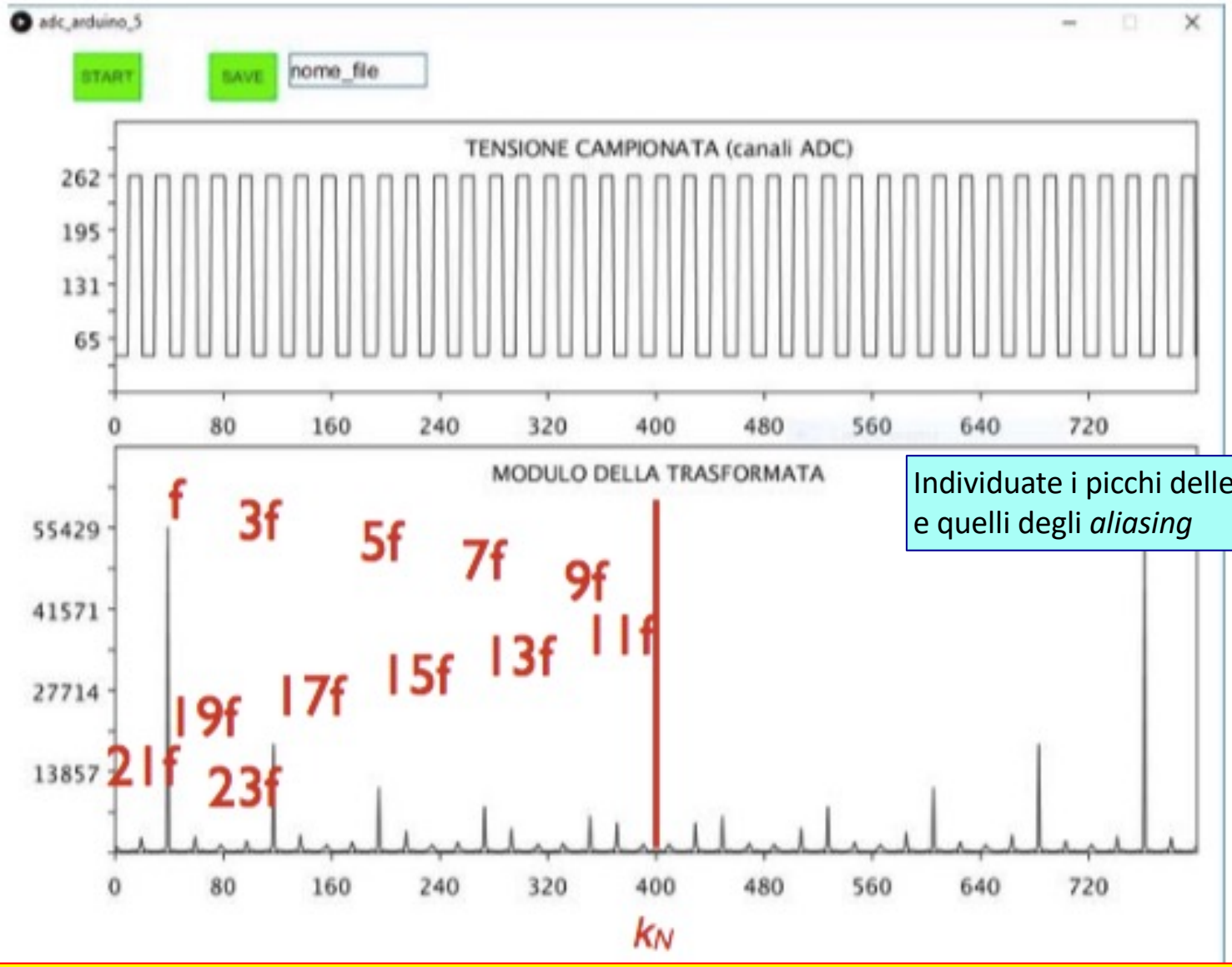


$$f_a = N \times f_N - f_s$$

N=2

- $k_{segnale}$ = 6000 / 12.5 Hz = 480 Sopra Nyquist → Aliasing
- $k_{apparente}$ = 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_t=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).

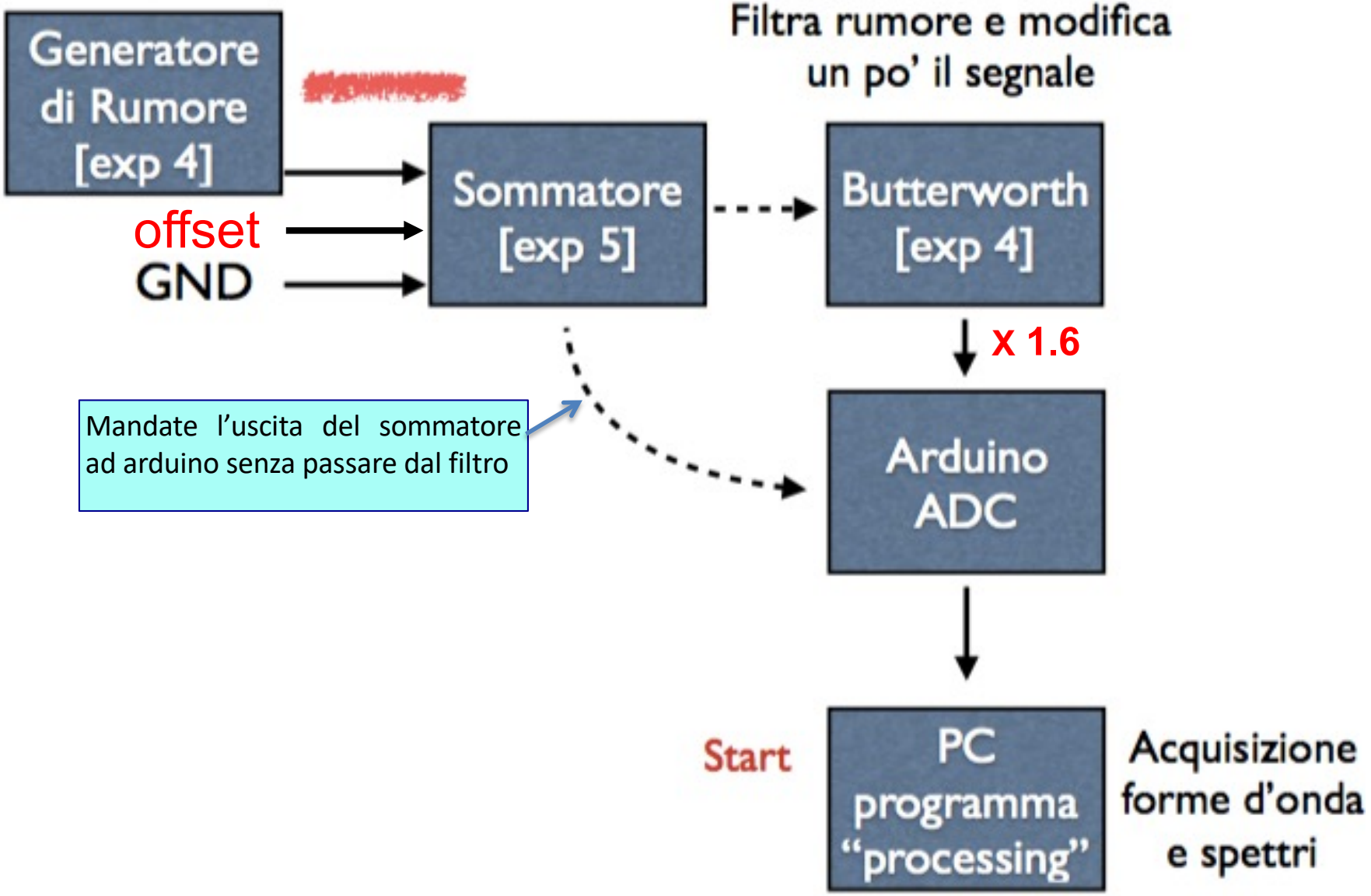
- Si può procedere salvando ogni campione su un file diverso in formato txt e poi, off-line, fare la media (con un programma in C o in Python).
- Il formato dei dati salvato da Processing, per ogni punto, è:
  - **k, V(k), Re(k), Im(k), mod (k)** ; *mod(k) è il modulo $|X_k|$ della frequenza k-esima*

- Si dovrebbe verificare che questo spettro è sostanzialmente piatto.
- Collegate poi il segnale al filtro e confrontate i risultati ottenuti.

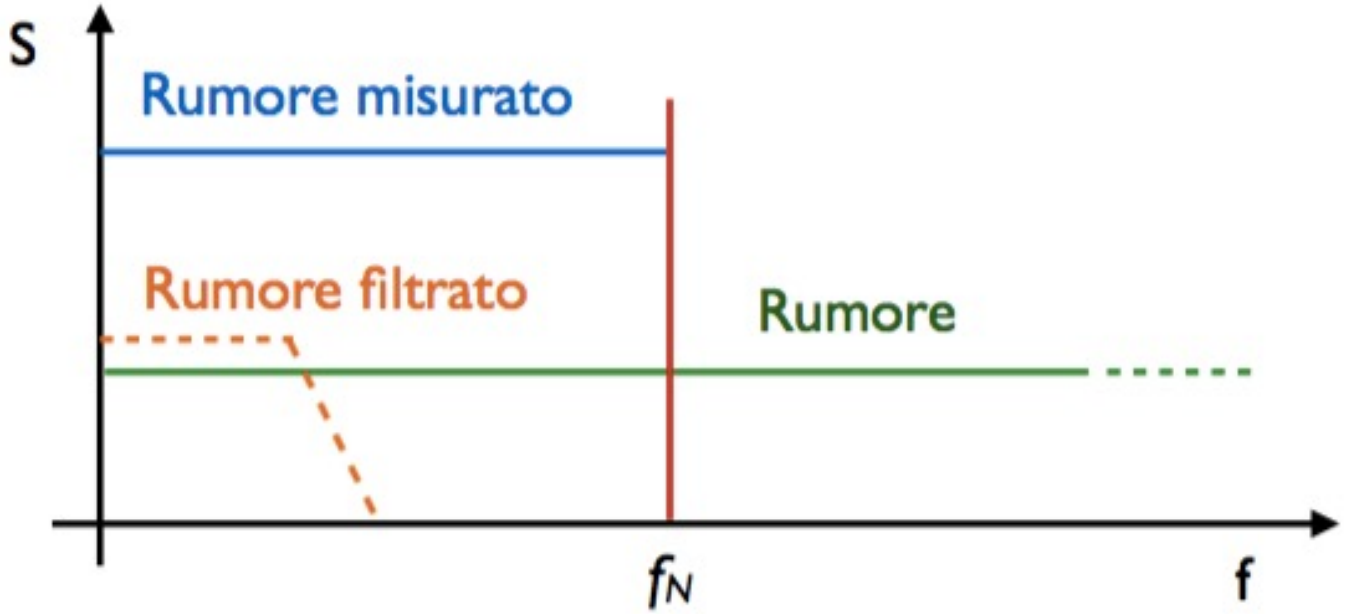
Il segnale ha una componente negativa, quindi utilizzate il sommatore per aggiungere un offset al rumore e portarlo nel range 0 – 3.3 V.

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

Misura del rumore



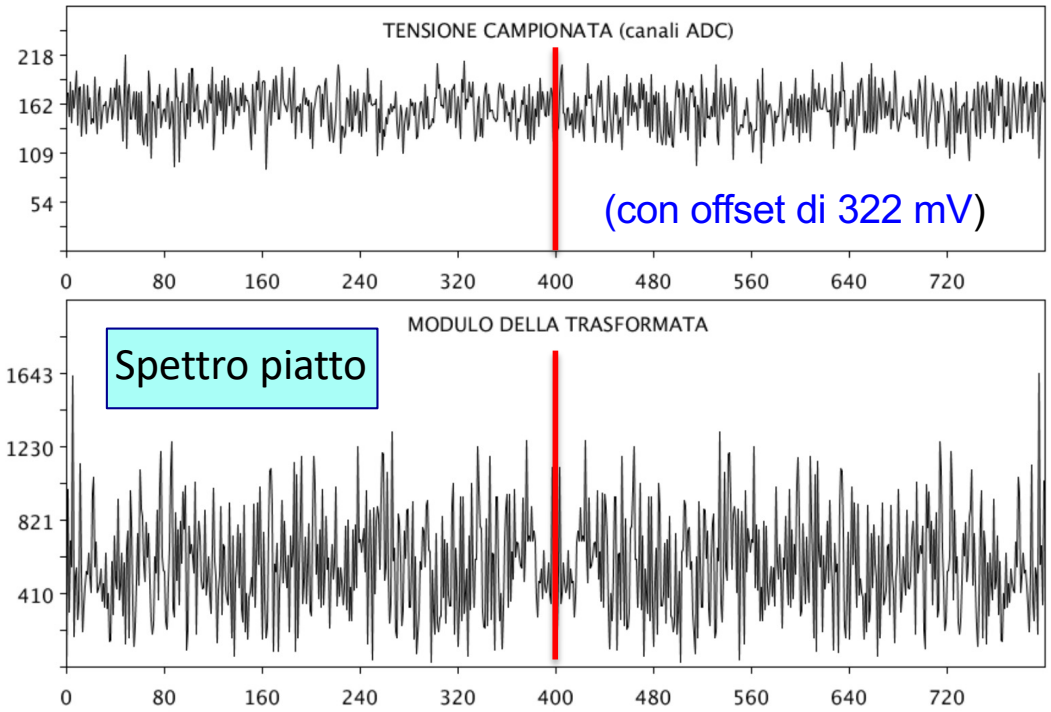
Cosa aspettarsi



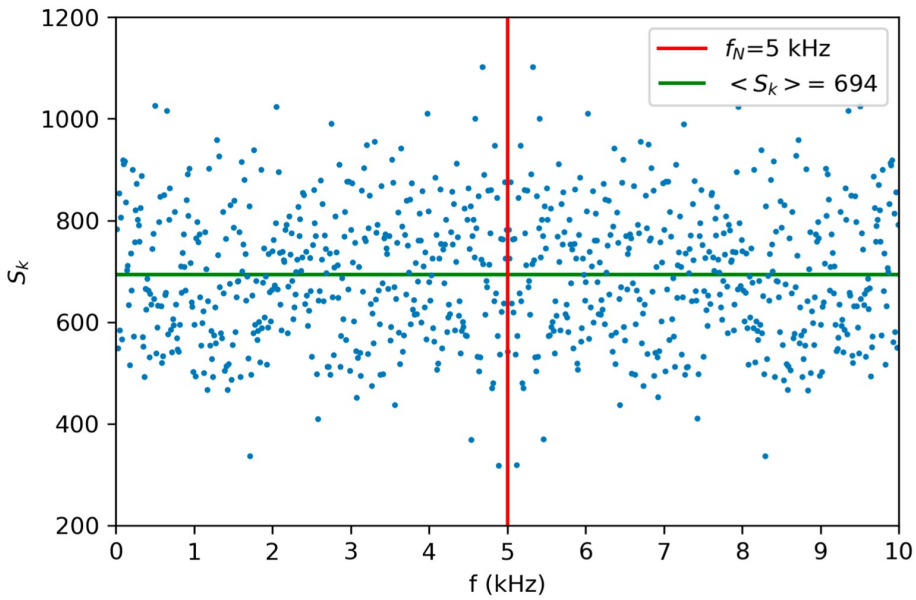
- Perché il rumore misurato ha un livello più alto di quello originale?
- Se avete tempo convertite i grafici da indice k a frequenza, in questo modo riconoscerete facilmente il taglio del Butterworth.

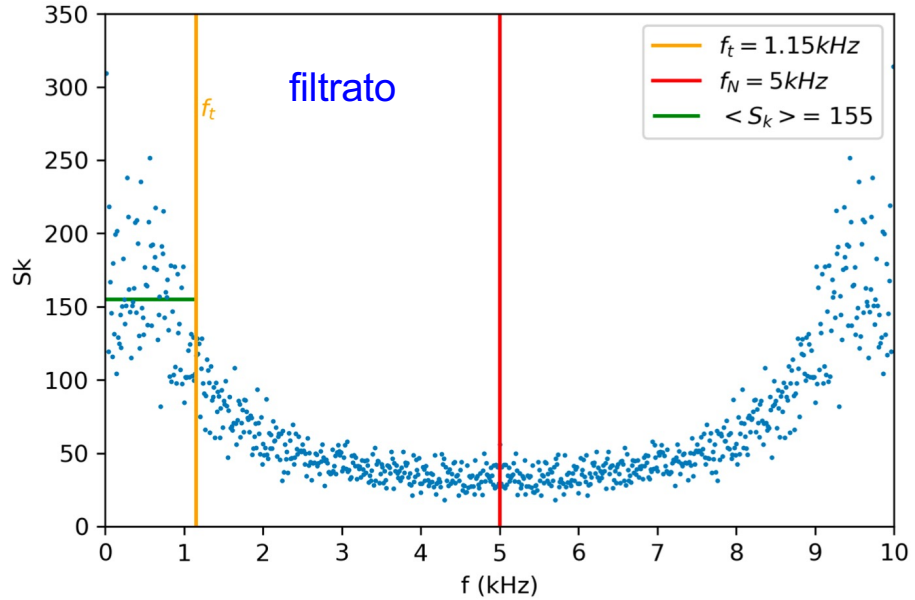
Misura del rumore

Senza filtro



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





Misura del rumore

Senza filtro

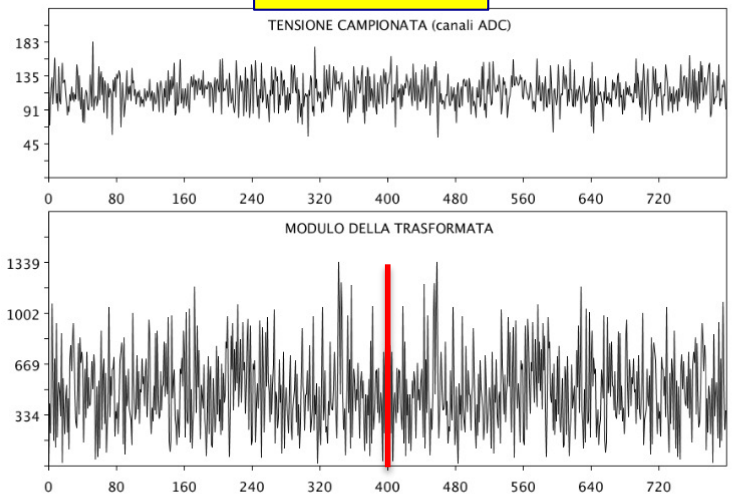


Figura 12: Esempio di rumore ricevuto dall'ADC.

Con filtro

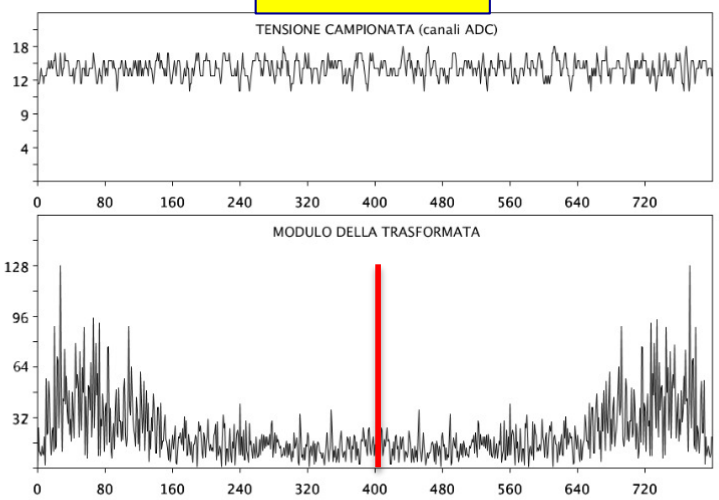


Figura 13: Esempio di rumore filtrato.

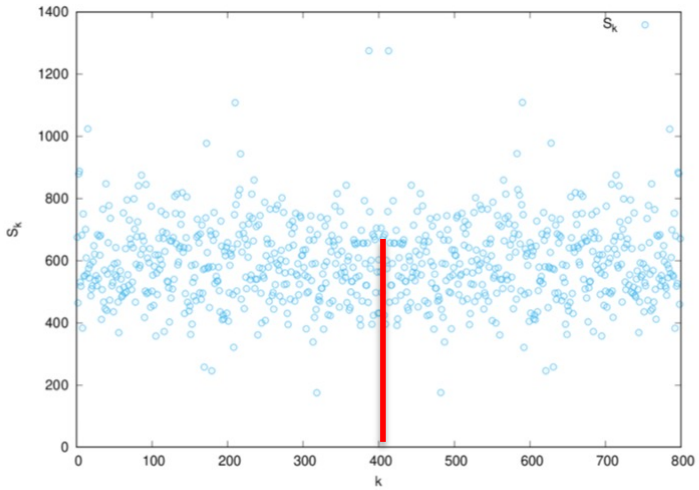


Figura 14: RMS di M campionamenti.

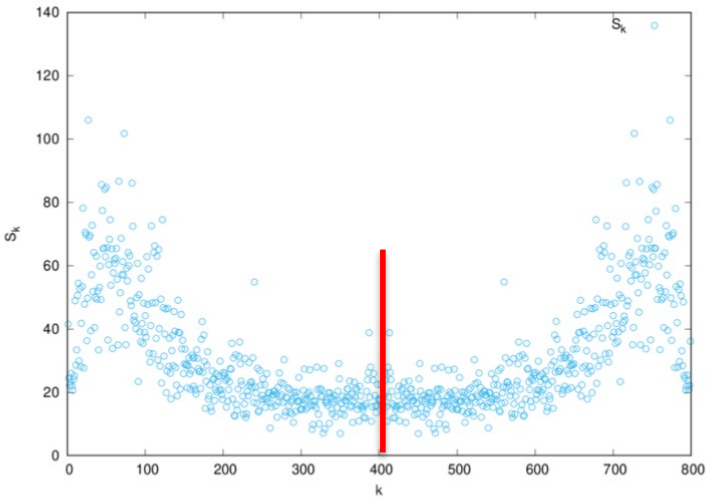
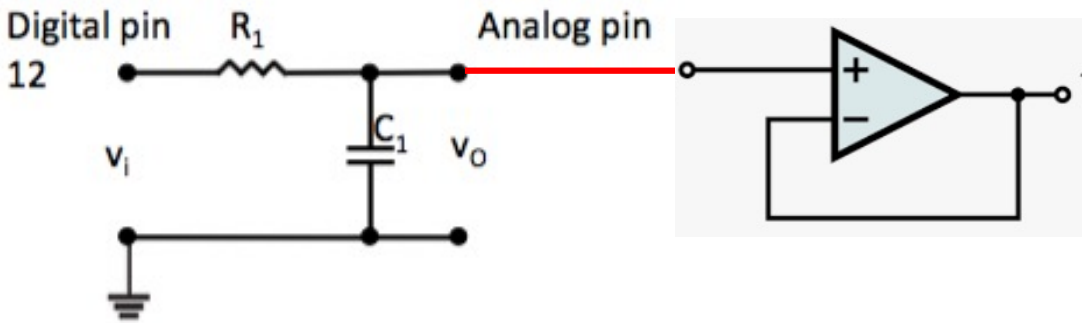


Figura 15: RMS di M campionamenti con rumore filtrato.

Segnale impulsivo

R=100 kΩ
C=100 nF
tau=10 ms

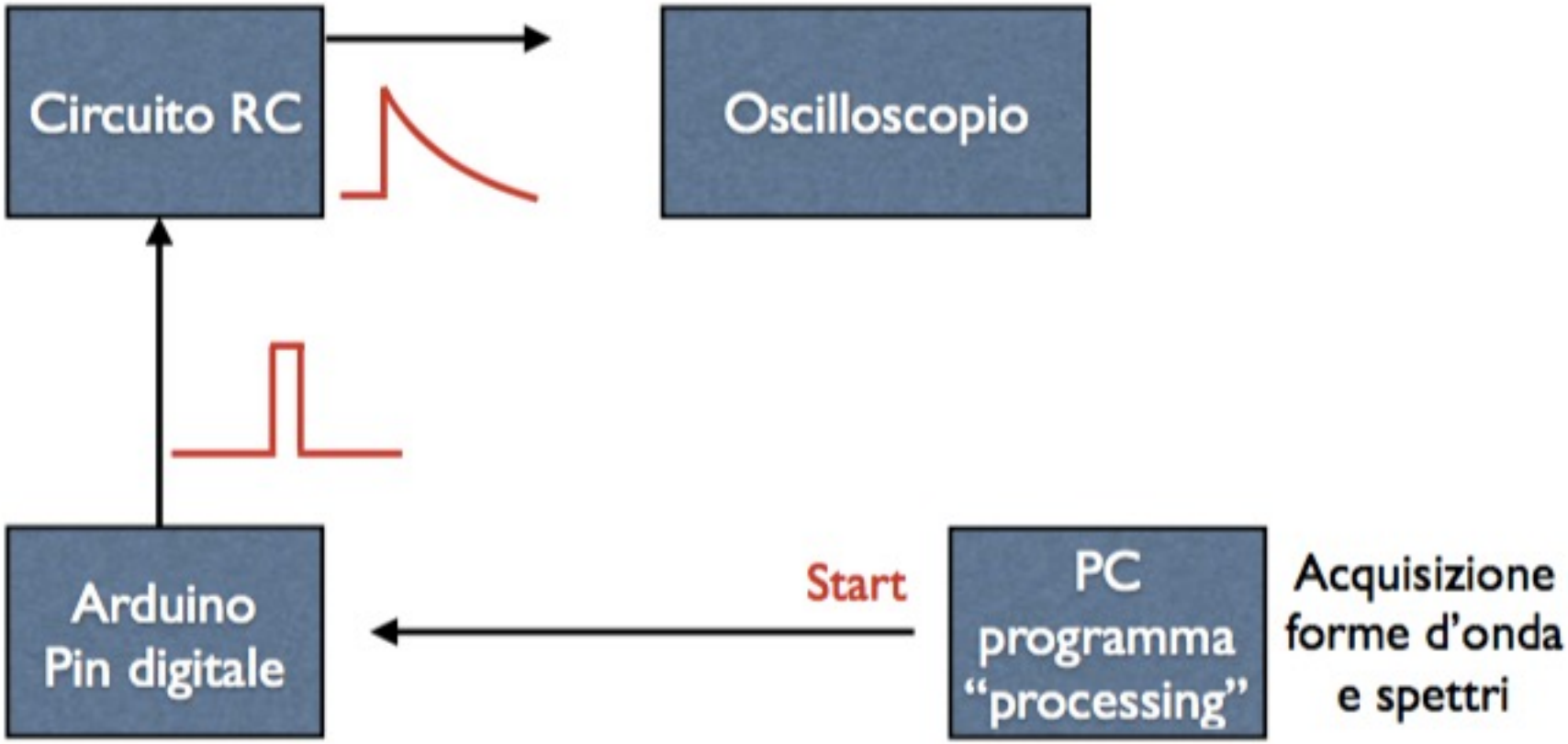


Analog Pin di Arduino oppure sommatore

- 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 sommattore, 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 kΩ x 100 nF = 10 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 ΔT = 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:

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

- Moltiplicando per la funzione di trasferimento del passa basso.

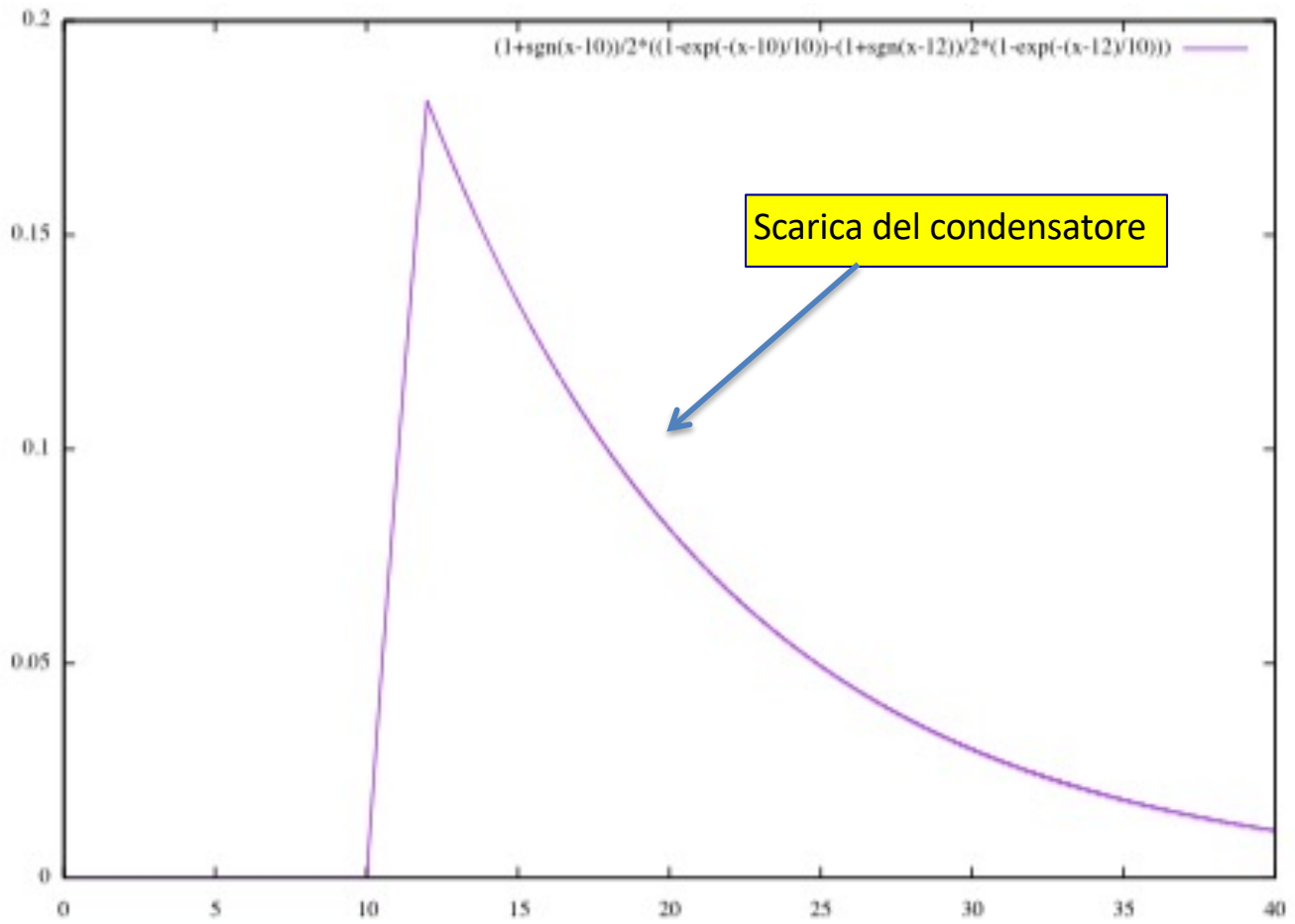
$$P(s) = f(s)\frac{1}{1 + s\tau}$$

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

- Quindi in pratica l'impulso è:

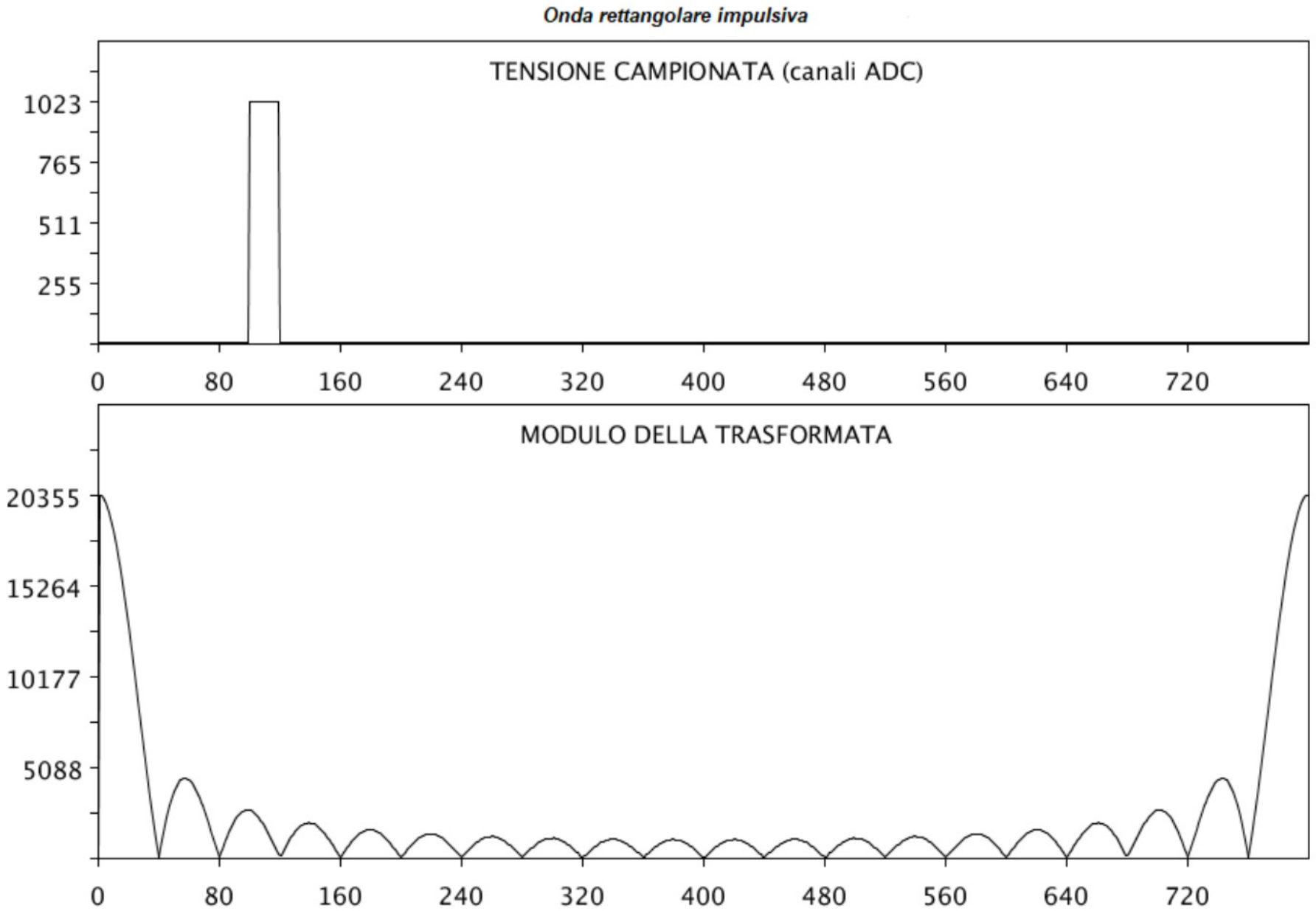
$$P(t) = \begin{cases} A[1 - e^{-t/\tau}], & \text{if } t < \Delta t \\ A[e^{\Delta t/\tau} - 1]e^{-t/\tau}, & \text{if } t >= \Delta t \end{cases}$$

segnale impulsivo simulato

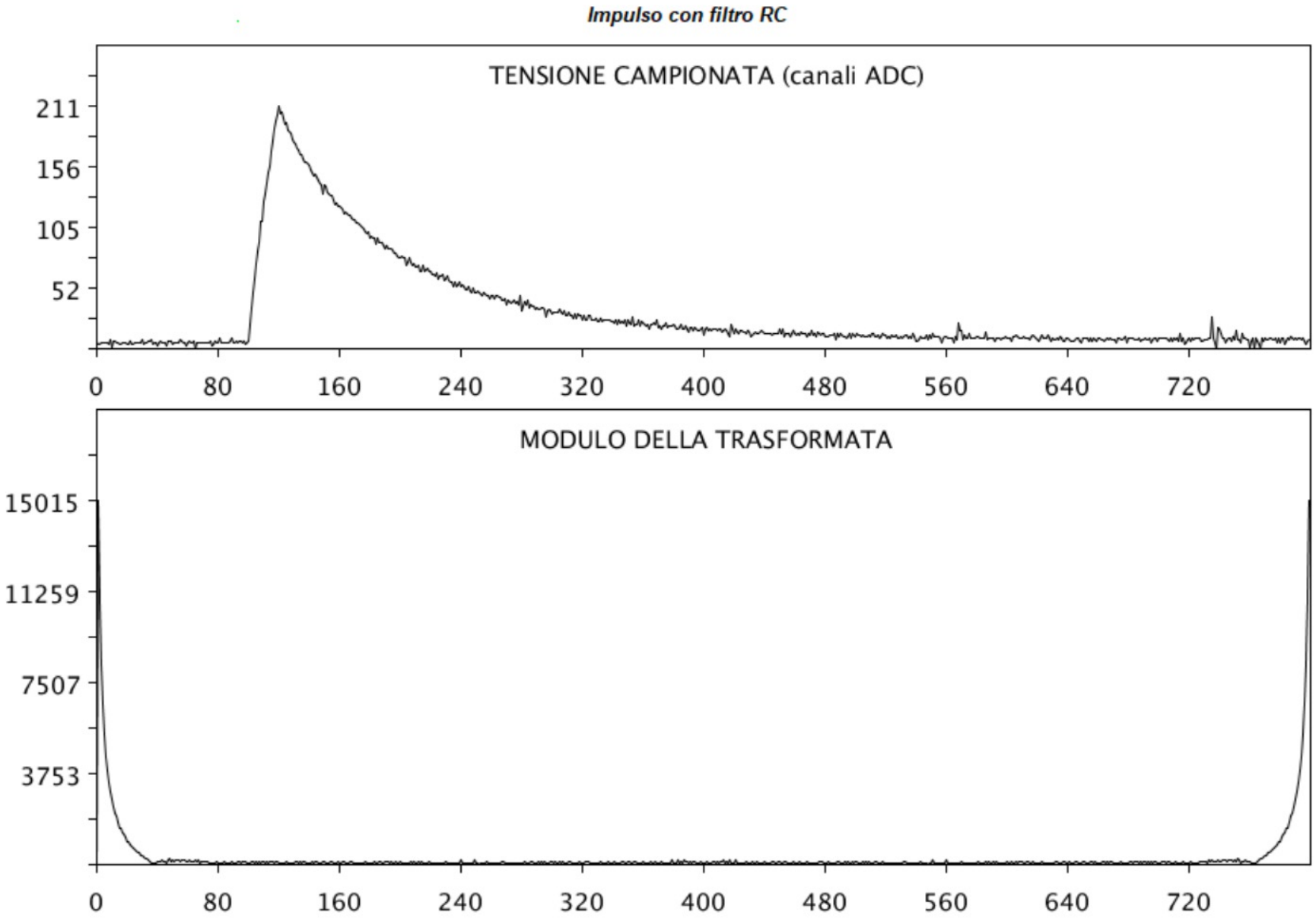


- Il massimo vale A $[1-e^{-t/\Delta t}]$ = 0.18 A

Segnale impulsivo



Segnale all'uscita del circuito RC



Misure segnale + rumore

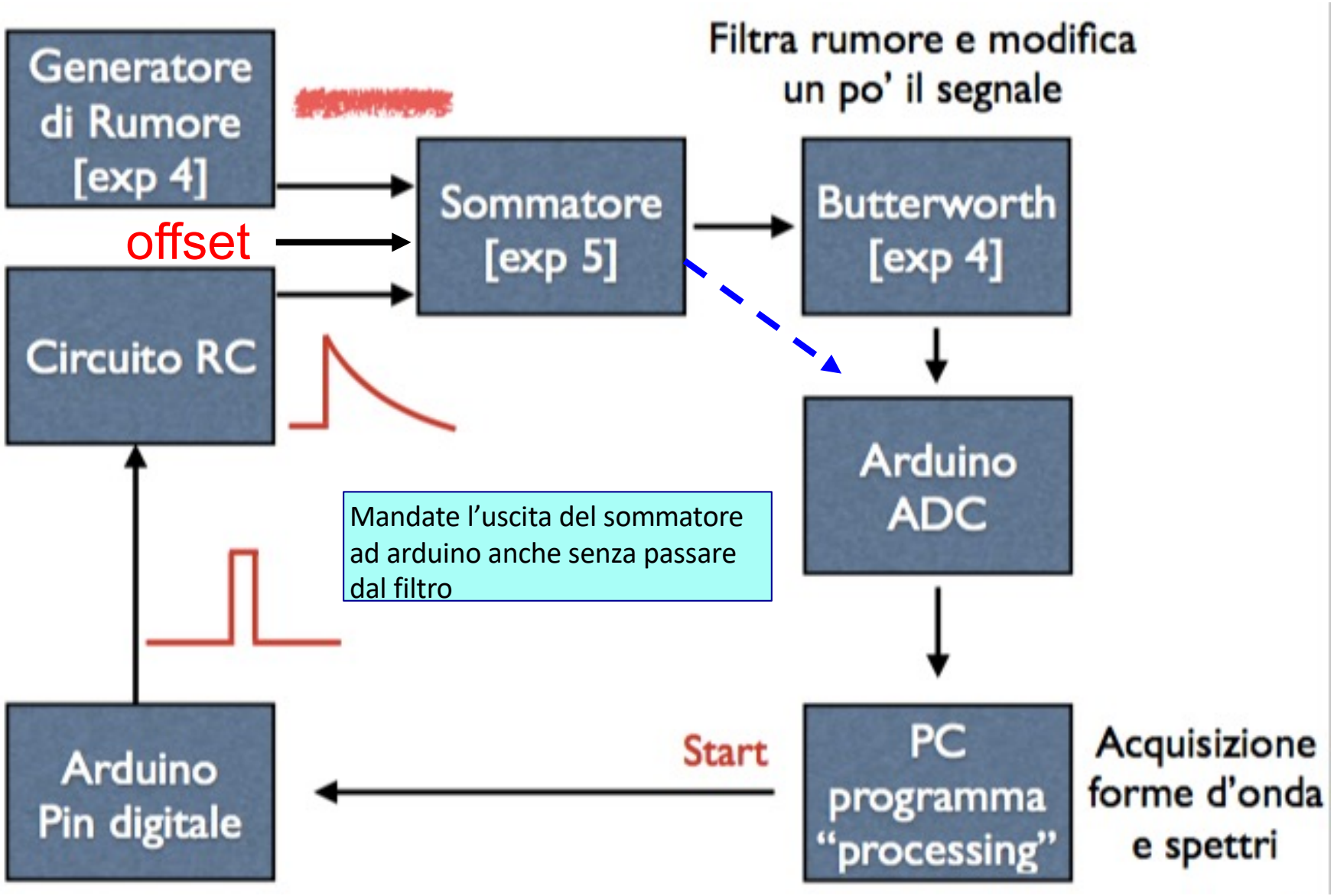
- Possiamo ora simulare una situazione reale aggiungendo al segnale impulsivo una certa quantità di rumore utilizzando il sommatore a 3 ingressi (se il rumore aggiunto vi sembra troppo basso potete aumentare il suo effetto agendo sui pesi (resistenze) del sommatore) e studiare lo spettro in frequenza 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.

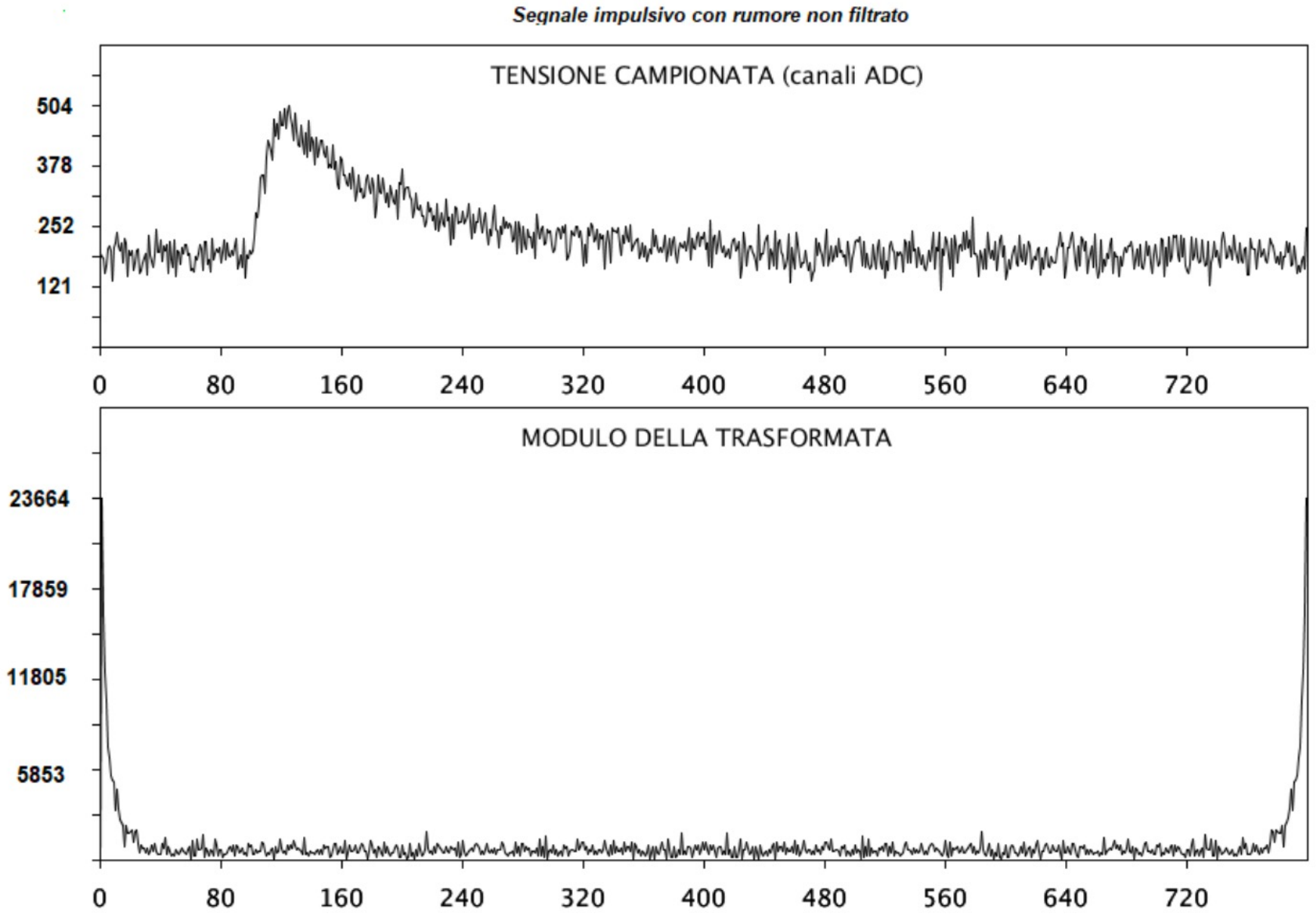
Circuito completo



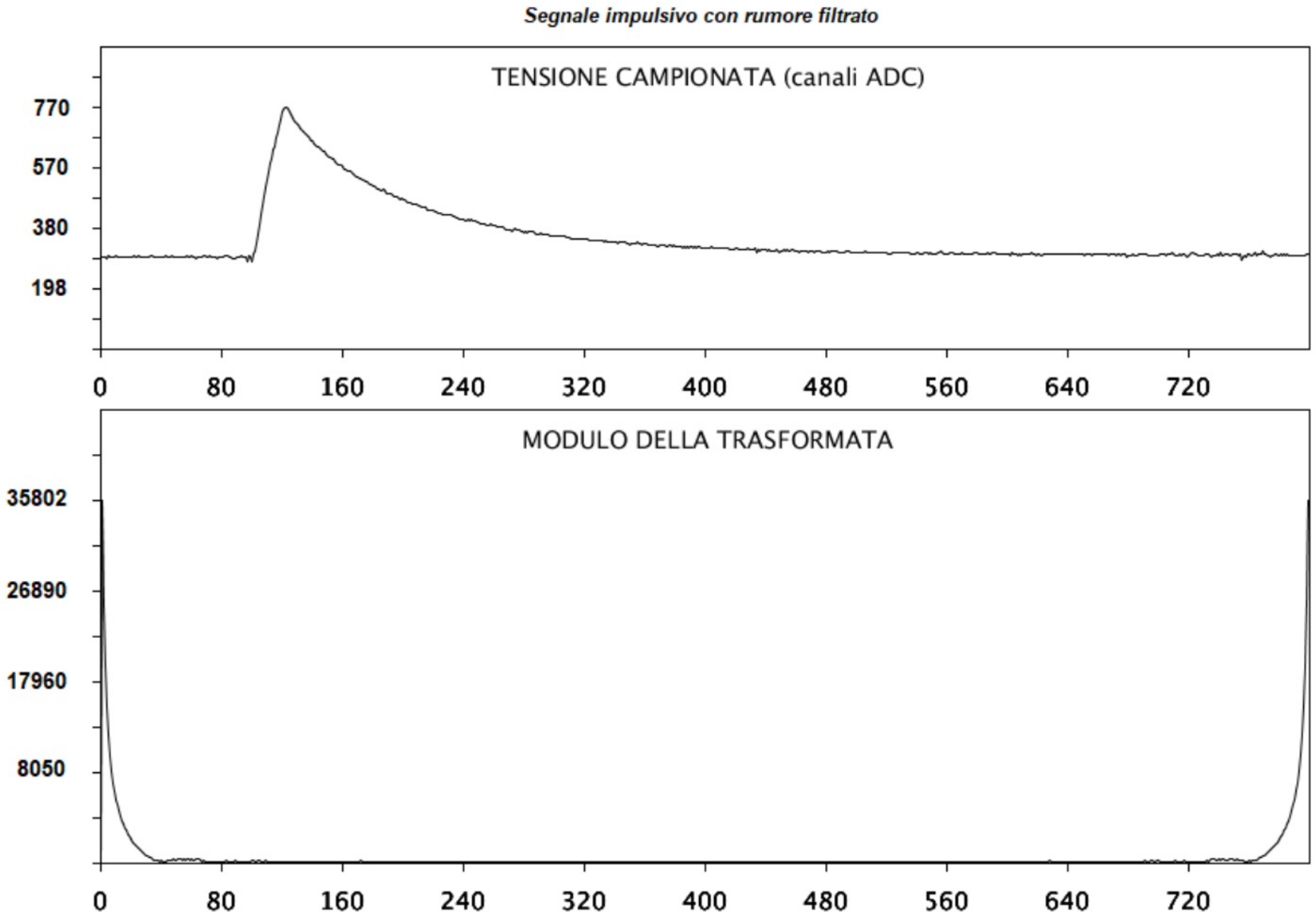
Misure segnale + rumore

- Analogamente a quanto fatto per il rumore, studiate lo spettro in frequenza del segnale da solo, staccando il rumore dl 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.

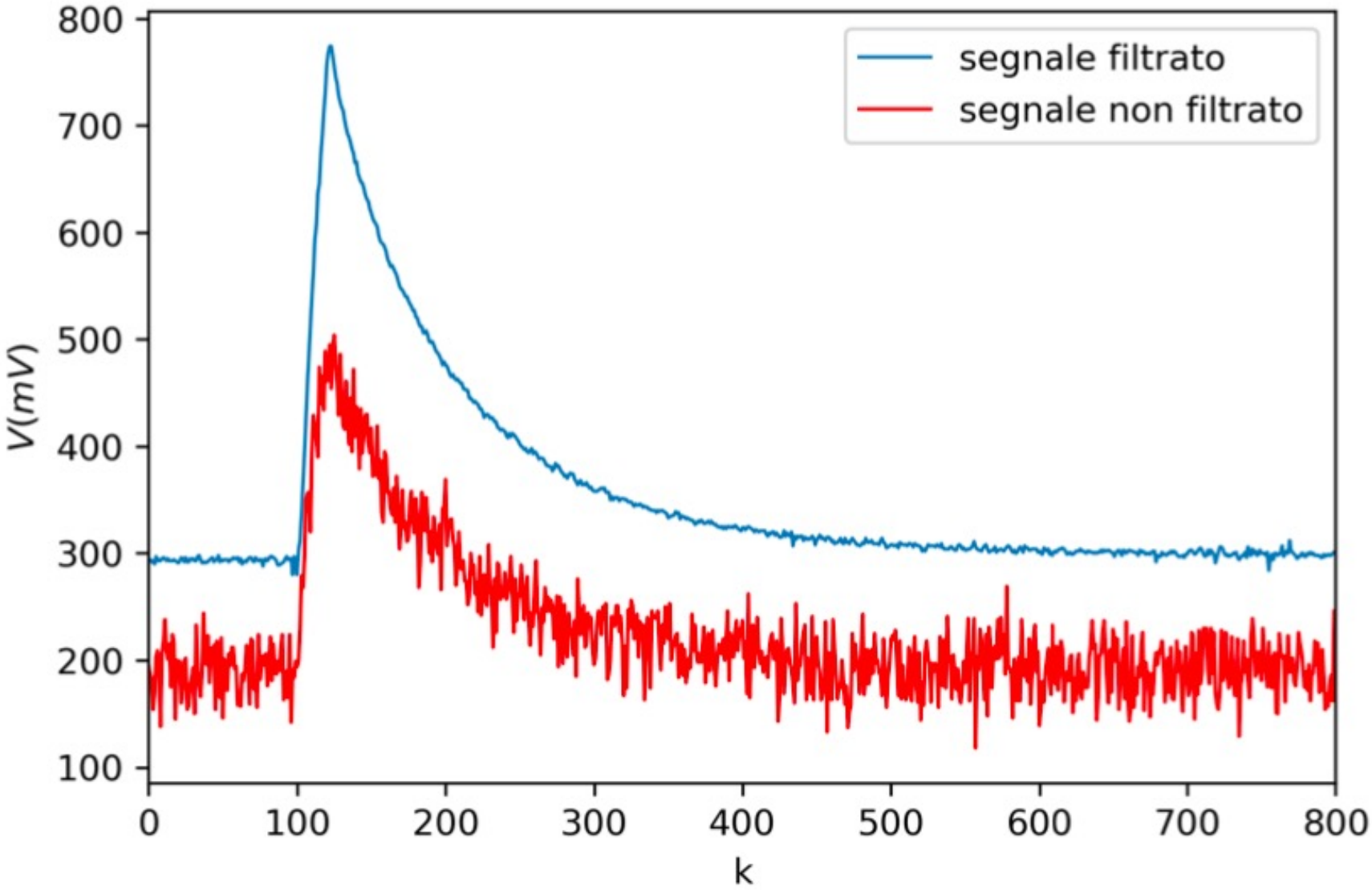
Segnale impulsivo con rumore non filtrato



Segnale impulsivo con rumore filtrato



Segnale impulsivo con rumore



Memento: il segnale filtrato è moltiplicato per 1.6

SAPIENZA
UNIVERSITÀ DI ROMA

Fine esercitazione 9