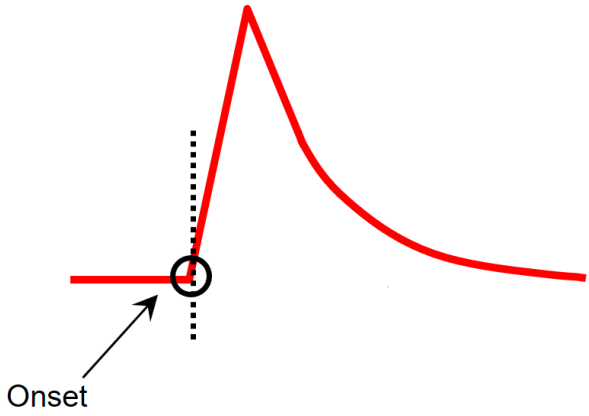
Onset Detection, La Corte Lorenzo

Contesto

Un onset è la fase antecedente al picco di ampiezza di un segnale sonoro:



Nella pubblicazione scientifica *evaluating the online capabilities of onset detection methods* ci vengono illustrati molti metodi per la *onset detection* (identificazione di onset), che si dividono principalmente in:

- Sviluppo di una **Onset Detection Function (ODF)**: sviluppo di una curva che descriva il cambiamento di intensità sonora tra frame vicini,
- Identificazione dei picchi (**Peak-Peaking**) sulla ODF.

Tutti i moderni algoritmi di onset detection usano una rappresentazione del segnale per frequenze, ottenuta grazie alla STFT, da cui poi si deriva lo spettrogramma. Su di esso viene calcolato lo **spectral flux**, che risulta essere la tecnica di ODF piu' moderna ed efficace. Sullo spectral flux viene poi implementato il **peak-peaking**, ovvero l'identificazione dei picchi.

Provo quindi a costruire la mia *Onset Detection* basandomi su questi macroscopici passi, nel dettaglio faccio riferimento alle formule della pubblicazione scientifica, cercando di ottenere un risultato quanto piu' vicino possibile a quello reale.

Obiettivi

- Creare un metodo di *onset detection*, diviso in:
  - costruzione dello *spectral flux*,
  - identificazione dei picchi.
- Confrontare i risultati rispetto a librosa, trovando le criticità dell'implementazione e quelle eventuali della libreria.

Metodo Implementato

Importo una serie di librerie che verranno usate nell'implementazione del metodo e nella fase di

analisi dei risultati:

In [1]:
```
# librerie principali
import numpy as np
import matplotlib.pyplot as plt
import IPython
from statistics import mean
import warnings
import scipy
warnings.filterwarnings("ignore")
%matplotlib inline

# librosa
import librosa
import librosa.display

# Parametri Principali
DURATION = 20
hop_length = 512
```

Calcolo dello *Spectral Flux*

Implemento una funzione che calcola lo *spectral flux* a partire dallo spettrogramma derivante dal segnale.

Calcolo quindi una sommatoria, per ogni frame temporale, di tutte le intensità dello spettro sottratte a quelle del frame precedente. Se il risultato della differenza è negativo lo rendo nullo *(half-wave rectifier function)* in modo da avere una funzione a valori positivi.

La funzione risultante sarà **indicativa di come cambiano le intensità dello spettro tra un frame temporale e quello precedente**.

$$SF(n) = \sum_{k=1}^{k=\frac{N}{2}} H(|X(n,k)| - |X(n-1,k)|)$$

$$\text{with } H(x) = \frac{x+|x|}{2} \text{ being the half-wave rectifier function.}$$

In [2]:
```
def implemented_onset_strength(S, sr, hop_length):
    odf_sf = []

    # normalizzazione dello spectral flux
    norm = np.linalg.norm(S.ravel() , ord=2)

    # sommatoria sulle differenze di intensità tra frame vicini
    for n in range(0,S.shape[1]-1):
        sum = 0
        for k in range(0,S.shape[0]-1):
            sum += abs(S[k,n]) - abs(S[k,n-1])

        odf_sf.append(sum/norm)

    # Applico la half-wave rectifier function eliminando i negativi
    odf_sf = np.maximum(0.00, odf_sf)
```

```
    return odf_sf
```

Identificazione dei Picchi sullo *Spectral Flux*

Sviluppo adesso una funzione che identifichi i picchi sullo *spectral flux*, che essendo i punti di massimo cambiamento sono gli onset.

Identifichiamo con *n* il frame temporale e con *ODF(n)* il valore dello *spectral flax* in quel frame. *ODF(n)* è definito picco se ci sono tre condizioni:

- se *ODF(n)* = ***max****(ODF(n-w1:n+w2))*
- se *ODF(n)* >= ***mean****(ODF(n-w3:n+w4))* + *delta*
- se *n* > (***n_last_onset***) + *w5*

con:

- *w1* (***pre_max***): quanti frame temporali antecedenti a quello corrente vengono considerati nel calcolo del massimo;
- *w2* (***post_max***): quanti frame temporali successivi a quello corrente vengono considerati nel calcolo del massimo;
- *delta* (***threshold***): soglia che, sommata alla media in un dato intervallo di tempo, deve essere superata per la considerazione di un picco;
- *w3* (***pre_avg***): quanti frame temporali antecedenti a quello corrente vengono considerati nel calcolo della media;
- *w4* (***post_avg***): quanti frame temporali successivi a quello corrente vengono considerati nel calcolo della media;
- *w5* (***wait***): i minimi frame temporali che devono passare da un picco a quello successivo.

In [3]:
```
def implemented_peak_pick(odf_sf, w1, w2, w3, w4, delta, w5):
    onsets = []
    n_last_onset = 0

    # scorro tutti i frame di ODF(n)
    for n, odf_value in enumerate(odf_sf):

        isPeak = True # se resta True allora è un picco

        # per evitare errori non considero frame che porterebbero a intervalli di te
        if (n-w1) < 0 or (n-w3) < 0:
            isPeak = False
            continue

       # un odf_value ODF(n) è definito picco se ci sono tre condizioni:

        # 1. se ODF(n) = odf_value = max(ODF(n-w1:n+w2))
        if odf_value != max(odf_sf[n-w1:n+w2]):
                isPeak = False

        # 2. se ODF(n) = odf_value >= mean(ODF(n-w3:n+w4)) + delta
        if odf_value < (mean(odf_sf[n-w3:n+w4]) + delta):
                isPeak = False
```

```
        # 3. se n > (n_last_onset) + w5
        if(n < (n_last_onset + w5)):
            isPeak = False

        if isPeak:
            onsets.append(n)
            n_last_onset = n

    return onsets
```

Funzione Principale

Definisco ora una funzione che racchiude i vari passi per l'identificazione degli onset, che quindi:

- Calcola lo spettrogramma, che al suo interno integra operazioni *preprocessing*,
- Calcola lo *spectral flux*,
- Identifica i picchi su di esso.

In [4]:

```
def my_onset_detect(y, sr):

    if y is None:
        raise ParameterError("Assenza di un segnale in input.")

    # Parametri per il calcolo dello spettrogramma
    n_mels = 138
    fmin = 27.5
    fmax = 16000.

    global S
    S = librosa.feature.melspectrogram(y, sr=sr, hop_length=hop_length, fmin=fmin, f

    # Onset Detection Function (ODF) basata sullo spectral flux
    global odf_sf
    odf_sf = implemented_onset_strength(S, sr, hop_length)

    # Peak-Peaking sullo spectral flux

    # pre_max: 3 valore ottimale secondo l'articolo, librosa gli da 1
    w1 = 3

    # post_max: 0-1 valori ottimali secondo l'articolo, librosa gli da 1
    w2 = 1

    # pre_avg: valore ottimale tra 4 e 12, librosa gli da 4
    w3 = 4

    # post_avg: 0-1 valori ottimali, librosa gli da 1
    w4 = 1

    # è la soglia di threshold, librosa gli da 0.07, impostata piu' bassa a causa di
    delta = 0.003

    # wait: 3 valore ottimale, librosa gli da 1
    w5 = 3

    onsets = implemented_peak_pick(odf_sf, w1, w2, w3, w4, delta, w5)

    return onsets
```

Analisi ed Esperimenti

Eseguo adesso una serie di test sul metodo implementato; utilizzo come valori per il confronto quelli dati in output dalla libreria Librosa, anche se non è detto (come vedremo) che essa identifichi tutti e soli gli onset corretti. Analizzo in particolare tre caratteristiche del metodo implementato:

- la **precisione**:
  - quanto si discosta in termini di numero di onset,
  - quanto si discosta in termini di posizione precisa identificata per l'onset.
- la **scalabilità**: quanto peggiora o migliora l'implementazione all'aumentare della durata della traccia.
- l'**adattabilità**: quanto peggiora o migliora l'implementazione al variare della traccia.

Risultati Generali

Per dare un primo sguardo generale ho creato un programma in Python *(allegato alla consegna)* che esegue il programma su tutte le 13 tracce esistenti nella libreria librosa e su diverse durate per le tracce: da 10 a 120 secondi.

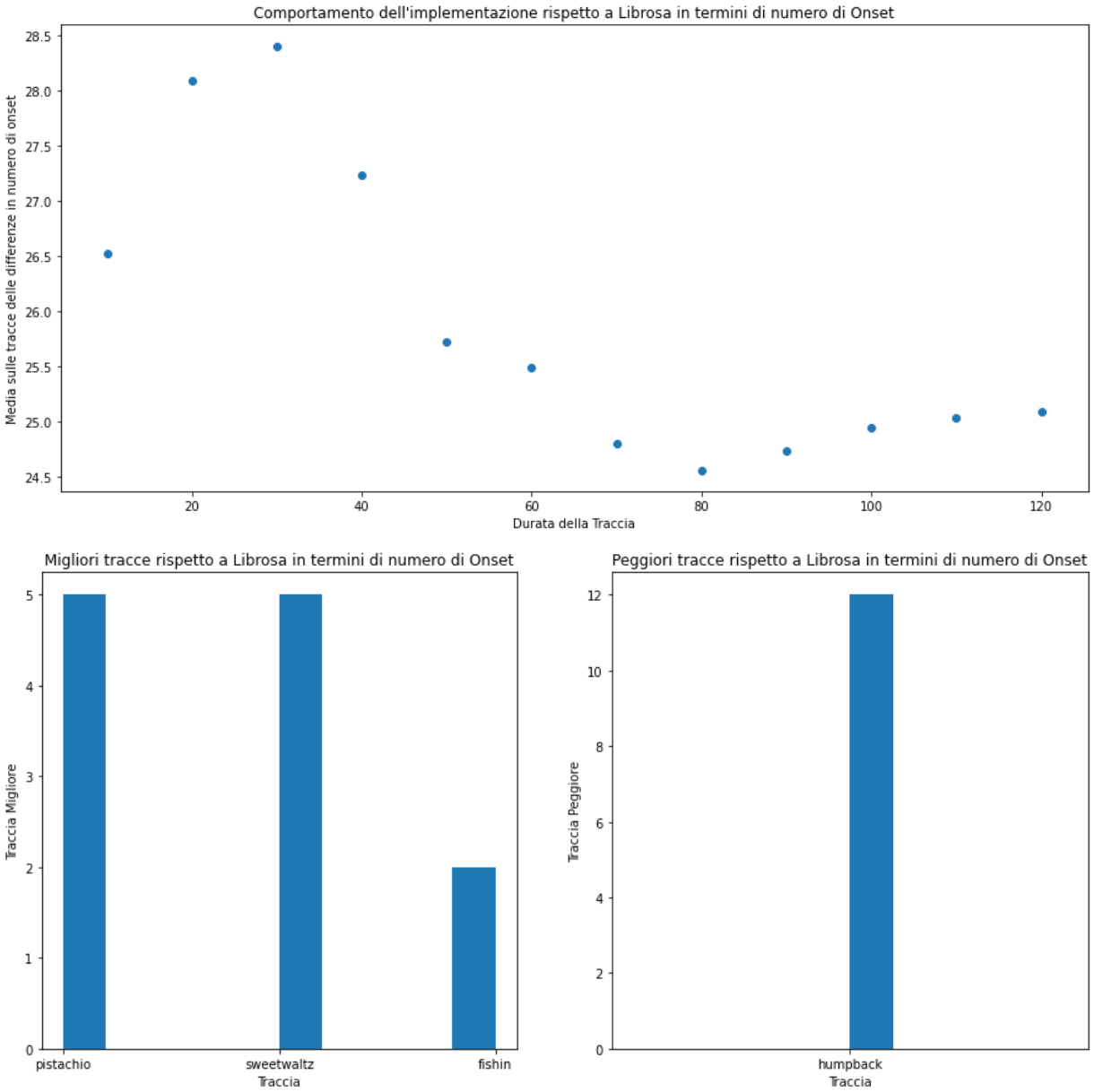
Trovo in questo modo:

- un media sulle tracce di quanto l'implementazione si discosta da librosa per quella determinata durata in secondi;
- un'indicazione di quale sia la traccia che si comporta meglio rispetto a librosa;
- un'indicazione di quale sia la traccia che si comporta peggio rispetto a librosa.

In [5]:

```
durations_list = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]

# Lista delle medie generali sulle differenze in numero di onset rispetto a Librosa
list_avg = [26.519301204884357, 28.087228002943796, 28.40186818561876, 27.2335900907

plt.figure(figsize=(15,7))
plt.title("Comportamento dell'implementazione rispetto a Librosa in termini di numer
plt.xlabel("Durata della Traccia")
plt.ylabel("Media sulle tracce delle differenze in numero di onset")
plt.scatter(durations_list, list_avg)
plt.show()

# Lista delle migliori traccie sul numero di onset rispetto a Librosa
best_tracks = ['pistachio', 'pistachio', 'pistachio', 'pistachio', 'pistachio', 'swe

# Lista delle migliori traccie sul numero di onset rispetto a Librosa
worst_tracks = ['humpback', 'humpback', 'humpback', 'humpback', 'humpback', 'humpbac

fig, axes = plt.subplots(ncols=2, figsize=(15, 7))

axes[0].title.set_text("Migliori tracce rispetto a Librosa in termini di numero di O
axes[0].set_xlabel("Traccia")
axes[0].set_ylabel("Traccia Migliore")
axes[0].hist(best_tracks)

axes[1].title.set_text("Peggiori tracce rispetto a Librosa in termini di numero di O
axes[1].set_xlabel("Traccia")
```

```
axes[1].set_ylabel("Traccia Peggiore")
axes[1].hist(worst_tracks);
```



Esempi Interessanti

Analizzo ora gli esempi piu' interessanti, in particolare:

- *humpback*, dove il confronto con librosa ha i risultati peggiori;
- *pistachio* e *sweetwaltz*, dove i confronti con librosa danno i risultati migliori;

Per farlo creo una funzione che utilizzerò per stampare informazioni salienti e grafici di confronto tra Librosa e l'implementazione:

In [14]:
```
def study_example(y, sr):

    # Applico sia la funzione implementata che quella di librosa

    onset_frames = my_onset_detect(y, sr=sr)
    onset_times = librosa.core.frames_to_time(onset_frames, sr=sr, hop_length=hop_le

    librosa_onset_frames = librosa.onset.onset_detect(y, sr=sr, hop_length=hop_lengt
    librosa_onset_times = librosa.onset.onset_detect(y, sr=sr, units='time', hop_len
```

```
print("\nNumero di Onset di Librosa: " + str(len(librosa_onset_times)))
print("Numero di Onset dell'Implementazione: " + str(len(onset_frames)))


# Creo il primo grafico di comparazione tra gli onset,
# che mostra il segnale e gli onset rappresentati su di esso

plt.figure(figsize=(14.4,10))

librosa.display.waveshow(y)

plt.axvline(x=onset_times[0], color='y', lw=3, label="Onset dell'Implementazione
plt.axvline(x=librosa_onset_times[0], color='r', label="Onset di Librosa")
plt.title("Visualizzazione degli Onset dell'Implementazione e di Librosa sul Seg
plt.xlabel("Tempo")
plt.ylabel("Ampiezza del Segnale")

for ot in onset_times:
    plt.axvline(x=ot, color='y', lw=2)

for ot in librosa_onset_times:
    plt.axvline(x=ot, color='r', lw=1)

plt.legend(loc="upper left")
plt.show()


fig, axes = plt.subplots(nrows=2, figsize=(100, 10))

# Creo il grafico di dettaglio per la comparazione tra gli onset,
# utilizzando lo spectral flux implementato
print("\n(dal notebook: cliccare due volte sull'immagine per ingrandirla e anali

times = librosa.times_like(odf_sf, sr=sr, hop_length=hop_length, n_fft=1024)

axes[0].plot(times, odf_sf)
axes[0].title.set_text("Visualizzazione Dettagliata sullo Spectral Flux Implemen
axes[0].set_xlabel("Tempo")
axes[0].set_ylabel("Altezza Normalizzata dello Spectral Flux Implementato")

for ot in onset_times:
    axes[0].axvline(x=ot, color='y', lw=3)

# Creo il grafico di dettaglio per la comparazione tra gli onset,
# utilizzando lo spectral flux di librosa

librosa_odf_sf = librosa.onset.onset_strength(y, sr)
librosa_times = librosa.times_like(librosa_odf_sf, sr=sr, hop_length=hop_length,

axes[1].plot(librosa_times, librosa_odf_sf)
axes[1].title.set_text("Visualizzazione Dettagliata sullo Spectral Flux di Libro
axes[1].set_xlabel("Tempo")
axes[1].set_ylabel("Altezza Normalizzata dello Spectral Flux di Librosa")

for ot in librosa_onset_times:
    axes[1].axvline(x=ot, color='r', lw=3)
```

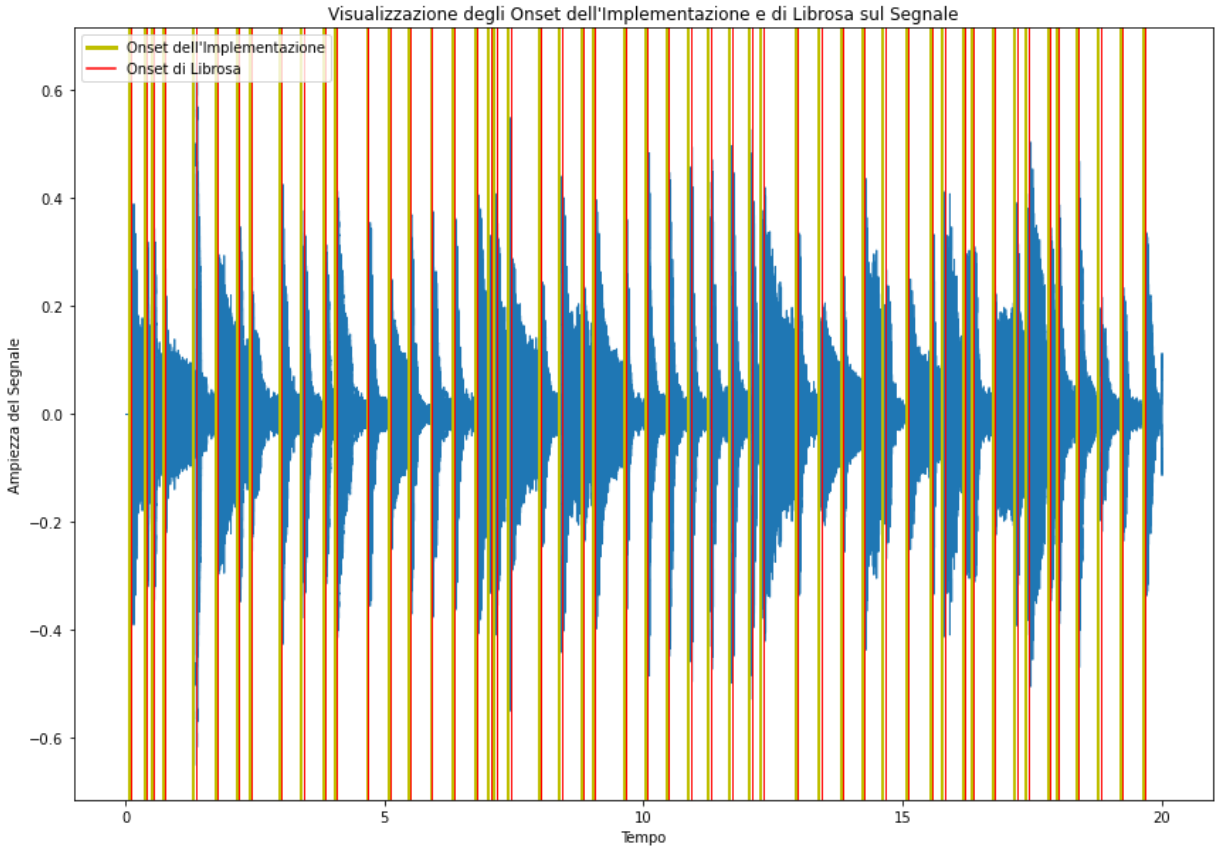
Le Tracce Migliori nel Confronto con Librosa: *pistachio* e *sweetwaltz*

In [11]:

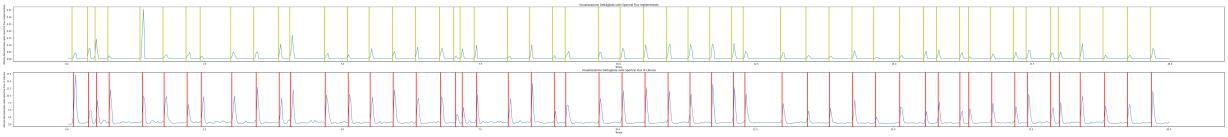
```
filename = librosa.example('pistachio')
```

```
y, sr = librosa.load(filename, duration=DURATION)
study_example(y, sr)
```

```
Numero di Onset di Librosa: 52
Numero di Onset dell'Implementazione: 52
```



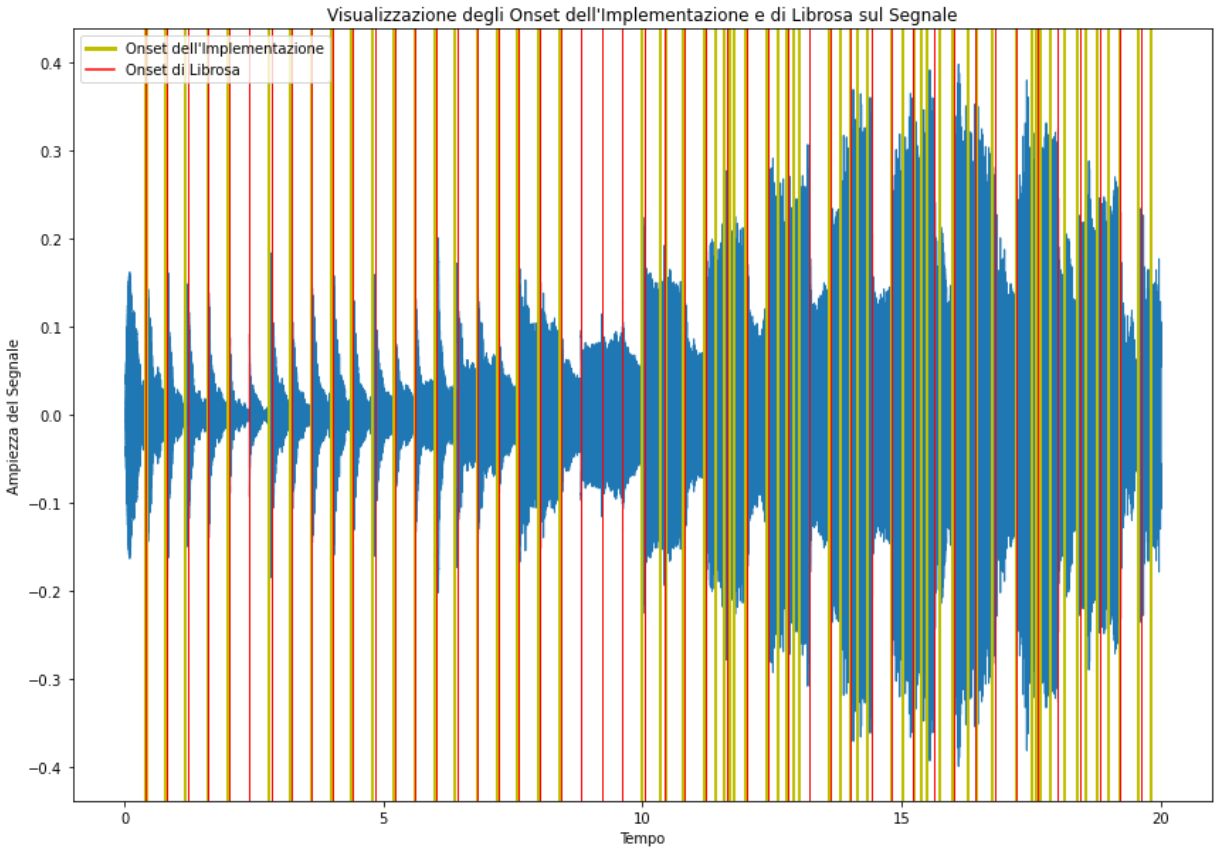

```
--- dal notebook: cliccare due volte sull'immagine per ingrandirla e analiz
zarla nel dettaglio ---
```



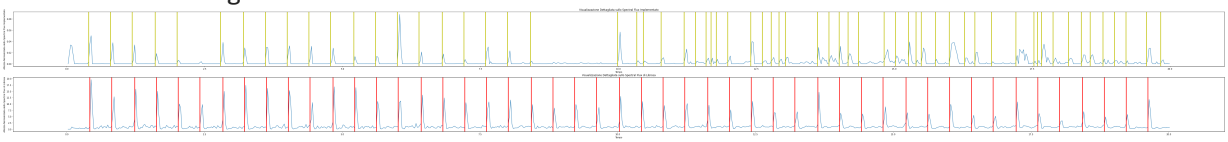
In [12]:

```
filename = librosa.example('sweetwaltz')
y, sr = librosa.load(filename, duration=DURATION)
study_example(y, sr)
```

```
Numero di Onset di Librosa: 49
Numero di Onset dell'Implementazione: 63
```




```
--- dal notebook: cliccare due volte sull'immagine per ingrandirla e analiz
zarla nel dettaglio ---
```

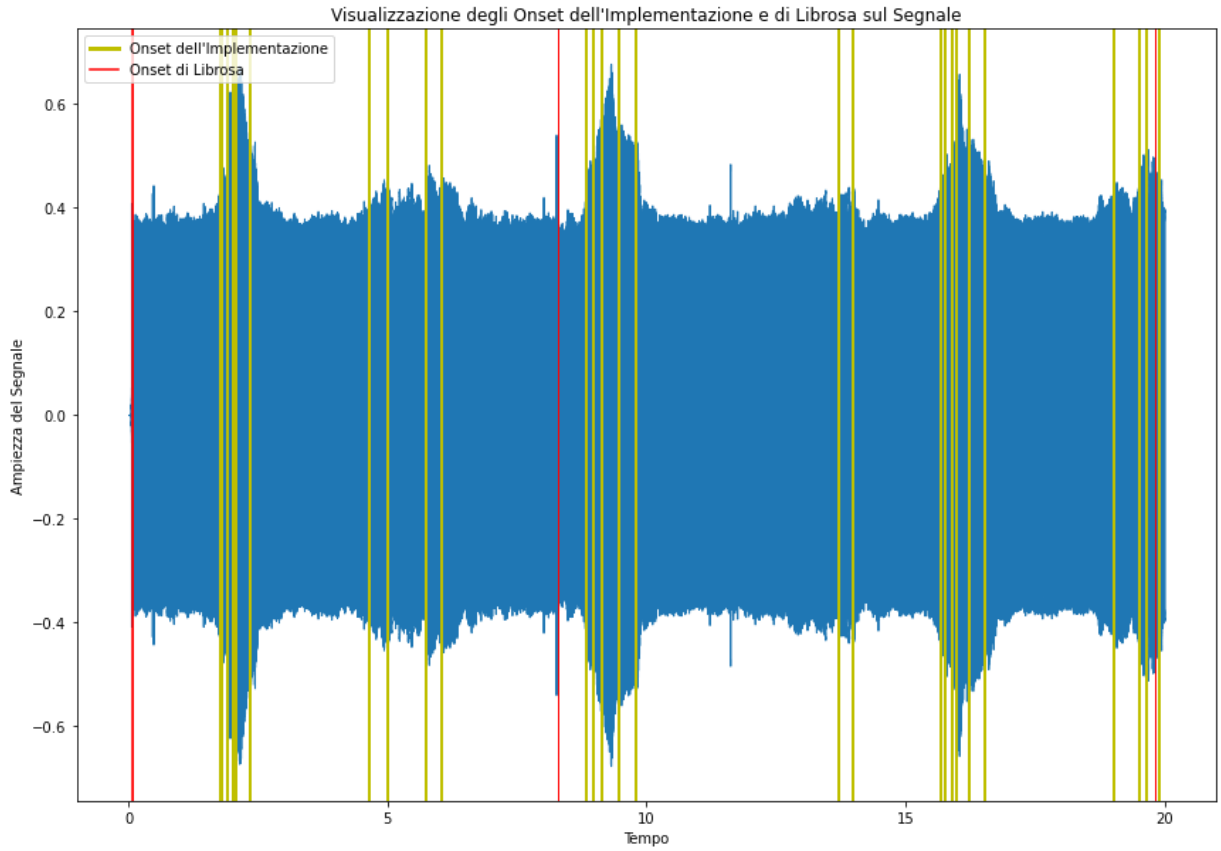


Notiamo come, a seconda del metodo utilizzato, cambino sia lo *spectrul flux* (in blu), sia gli onset trovati (in giallo ed in rosso). Lo *spectral flux* descritto dall'implementazione ha picchi piu' bassi ed è molto piu' piatto per le basse intensità, nonostante cio' la rilevazione degli onset sembra buona in entrambi i casi.

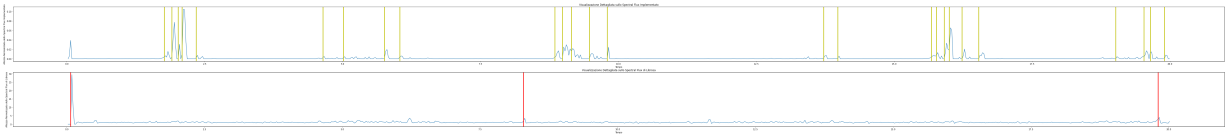
La Traccia Peggiore nel Confronto con Librosa: *humpback*

In [13]:
```
filename = librosa.example('humpback')
y, sr = librosa.load(filename, duration=DURATION)
study_example(y, sr)
```

```
Numero di Onset di Librosa: 3
Numero di Onset dell'Implementazione: 26
```




```
--- dal notebook: cliccare due volte sull'immagine per ingrandirla e analiz
zarla nel dettaglio ---
```



In questo caso notiamo sorprentemente che lo *spectral flux* di librosa sembra essere molto piatto. Probabilmente l'implementazione della libreria soffre segnali molto rumorosi, mentre l'implementazione si adatta meglio ad essi.

Una criticità dell'implementazione è visibile per il primo picco, non considerato dalla funzione di peak-peaking poiché l'intervallo di tempo in questione avrebbe considerato dei frame temporali negativi.

Conclusioni

I risultati ottenuti mostrano punti di forza e di debolezza dell'implementazione:

- essa risulta discreta rispetto a librosa, con differenze in numero di onset intorno al 25%;
- allo stesso tempo però notiamo che **in alcuni casi librosa ha delle criticità**, come nel caso di forte rumore, dove il calcolo dello *spectral flux* restituisce risultati molto approssimativi; in questi casi, calcoli e normalizzazioni semplici come quelle implementate mostrano risultati piu' sensati.

Fonti

- Pubblicazione Scientifica: *evaluating the online capabilities of onset detection methods*, Sebastian Bock, Florian Krebs and Markus Schedl: https://ismir2012.ismir.net/event/papers/049_ISMIR_2012.pdf

- *Librosa*, in particolare il codice sorgente delle funzioni:
  - librosa.onset.onset_detect: https://librosa.org/doc/main/generated/librosa.onset.onset_detect.html,
  - librosa.onset.onset_strength: https://librosa.org/doc/main/generated/librosa.onset.onset_strength.html,
  - librosa.util.peak_pick: https://librosa.org/doc/main/generated/librosa.util.peak_pick.html.