Creazione di un demodulatore di segnale FM simulato in software

Lena Giovanni Leonardo

15 Febbraio 2022

1 Introduction

1.1 Segnale FM

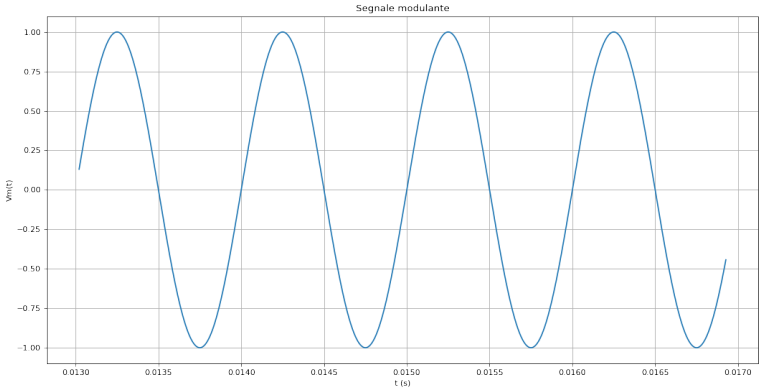
La modulazione FM è una modulazione che consente di codificare un segnale detto modulante all'interno di un altro segnale di frequenza più alta detto portante. L'informazione del segnale modulante è contenuto nella variazione di frequenza. Il segnale modulato ha quindi un'ampiezza costante ma una frequenza che varia. La modulazione FM è usata spesso quando si vuole trasmettere un segnale a bassa frequenza attraverso onde radio.

1.2 Tecnica DSP

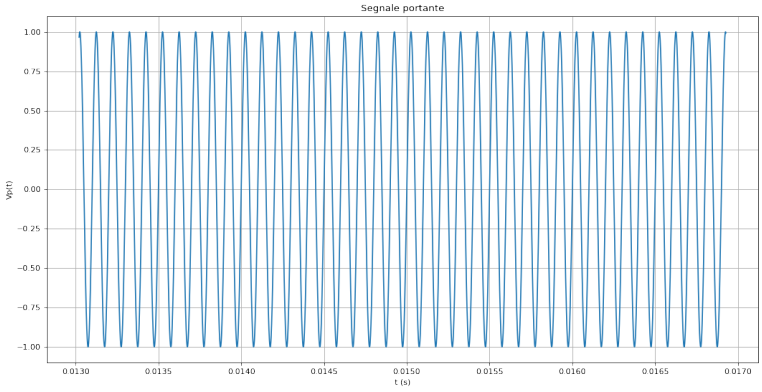
DSP o Digital Signal Processing consiste nell'elaborazione di un segnale utilizzando tecniche software, spesso emulando un circuito. La comodità di questa tecnica è che un computer può emulare qualsiasi circuito ed è quindi molto più flessibile.

2 I segnali di partenza

La prima simulazione è realizzata in Python con il software Jupyter Lab che permette di eseguire parti di codice e visualizzarne immediatamente i risultati tramite grafici e tabelle. Il codice scritto in Python sarà poi importato all'interno di LabVIEW.
Prima di demodulare un segnale FM è necessario avere un segnale FM, il quale si crea a partire da un segnale modulante ed uno portante. In questo esempio i due segnali sono entrambi sinusoidali, quello modulante ha una frequenza $f_m = 1kHz$ mentre quello portante ha una frequenza di $f_m = 10kHz$.
Avvalendosi degli strumenti di visualizzazione di Jupyter Lab, è possibile visualizzare il grafico dei due segnali:

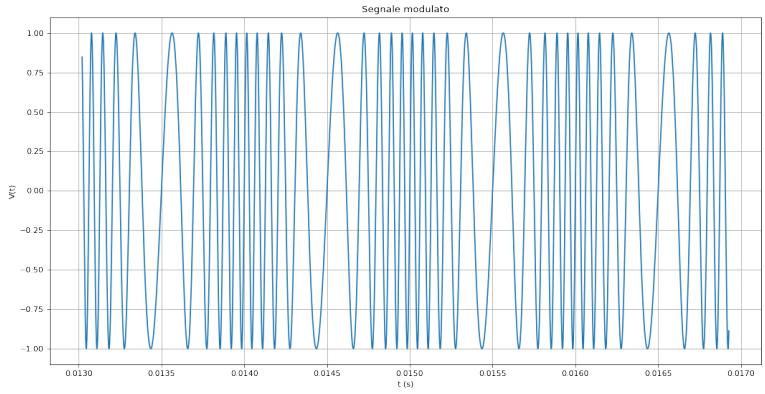


Il segnale modulante, $f = 1kHz$



Il segnale portante, $f = 10kHz$

Che uniti vengono modulati nel seguente segnale.



3 Implementazione in DSP

3.1 Passa basso

Per demodulare un segnale FM è necessario applicare un filtro passa-basso. In hardware si potrebbe implementare con un circuito R-C, mentre in software è sufficiente scrivere un algoritmo che discretizza il circuito.
L'implementazione più semplice richiede un ciclo for e si può scrivere in Python con la seguente funzione:

```
def low_pass_filter(signal: List[float], tau: float) -> List[float]:
    output = np.zeros_like(signal)

    output[0] = tau * signal[0]
    for i in range(1, len(modulated)):
        output[i] = output[i - 1] + tau * (signal[i] - output[i - 1])

    return output
```

La funzione prende in input un segnale e una tau, e ritorna il segnale filtrato.

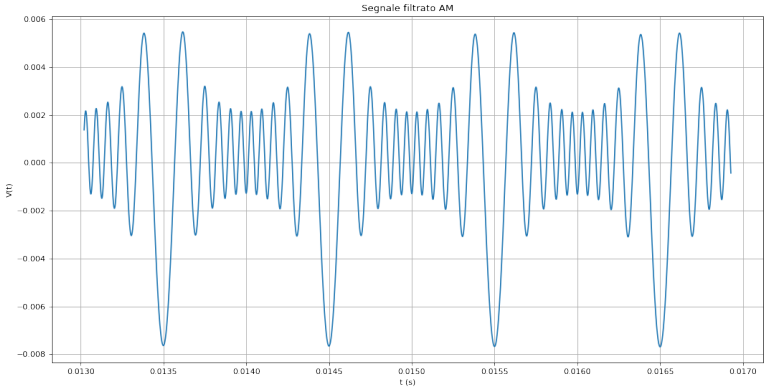
3.2 Diodo

4 Demodulazione in Python

4.1 Filtro passa basso

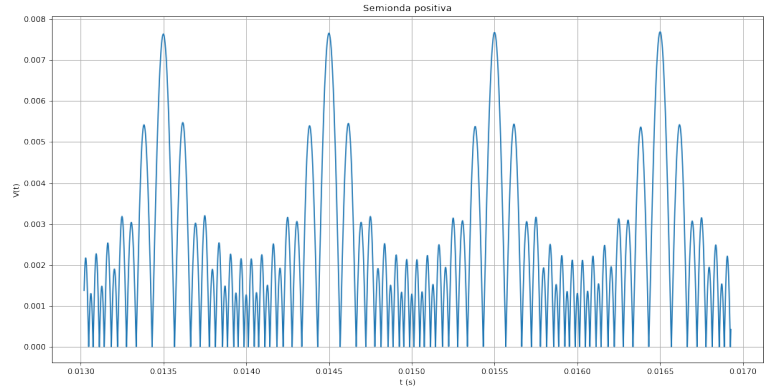
Per poter tornare al segnale modulante di partenza, il primo passo è quello di applicare un filtro passa-basso. Nel caso concreto, la frequenza di taglio del

filtro è pari al 70% della frequenza della portante, che in questo caso è pari a $7kHz$.
Applicando la funzione creata in precedenza, si ottiene in output un segnale pseudo-AM.



4.2 Semionda positiva

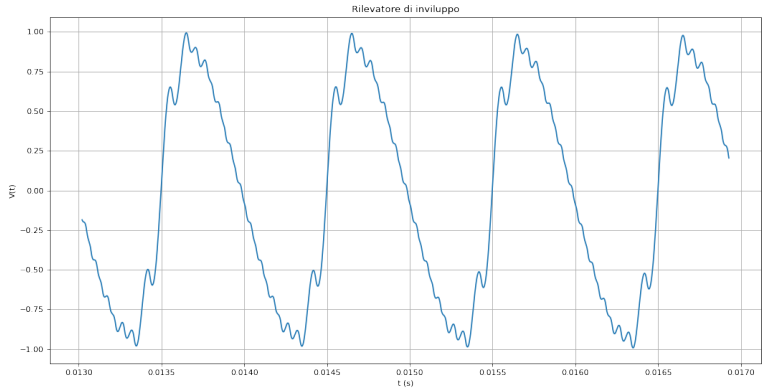
Il secondo passaggio consiste nell'ottenere solamente la semionda positiva del segnale AM. Per far ciò è sufficiente applicare a tutto il segnale la funzione matematica di modulo, che in python si chiama abs (da absolute).



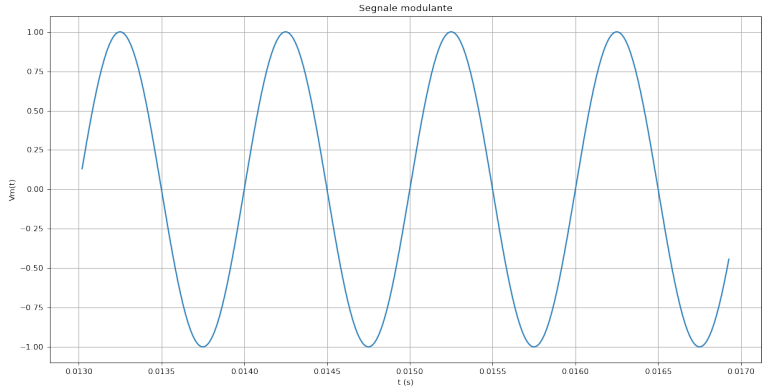
4.3 Rilevatore di inviluppo

Per ottenere indietro il segnale modulante, si applica un cosiddetto rilevatore di inviluppo, che non è altro che un filtro passa basso che segue i "picchi" del

segnale, togliendo tutta la cosiddetta seghettatura.
La frequenza di taglio del filtro passa basso in questo caso è pari a tre volte la frequenza modulante, ovvero $3kHz$. Avendo già la funzione pronta per il passa basso, è sufficiente chiamarla passando come parametri il segnale della semionda positiva e la tau relativa a $3kHz$.
Sottraendo poi un segnale DC per centrare lo zero del segnale ed avere una parte positiva ed una negativa, si ottiene finalmente un'onda che assomiglia molto alla sinusoide di partenza.



Il segnale demodulato.



Il segnale modulante di partenza.

Come si può facilmente notare confrontando i due segnali, il primo è sfasato di 180 gradi rispetto al secondo. Si tratta probabilmente del risultato dei due filtri passa-basso di primo ordine, che oltre ad attenuare il segnale applicano anche uno sfasamento.

5 Implementazione in LabVIEW

LabVIEW permette di eseguire codice Python, per cui è possibile utilizzare tutto il codice