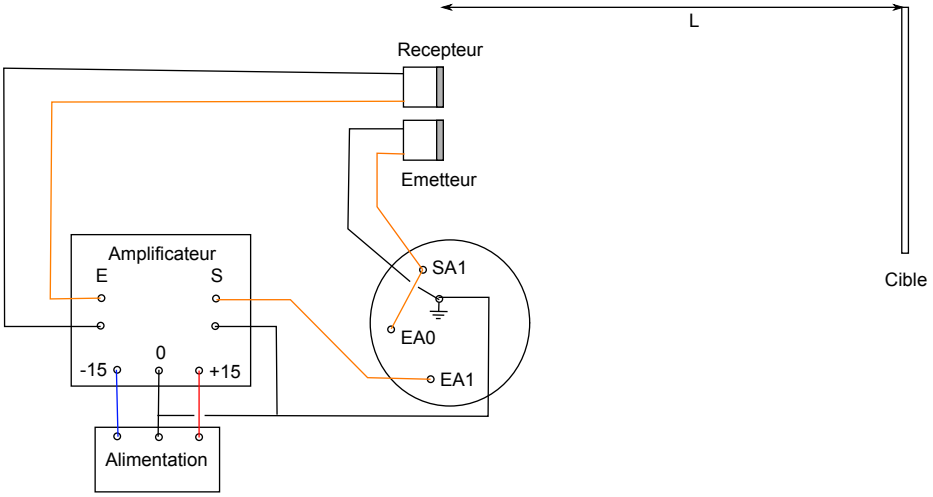
Mesure de la vitesse du son et télémètre à ultrason

1. Expérience

L'émetteur et le récepteur dont deux transducteurs piézoélectriques, disposés parallèlement, très proche l'un de l'autre (quelques millimètres). La cible est une plaque d'environ $10\ cm$ de large fixée verticalement à une distance de l'émetteur et du récepteur inférieure à 1 mètre.



Le signal comportant le paquet d'onde est délivré par la sortie SA1 de la carte SysamSP5. Un amplificateur à gain réglable permet d'augmenter l'amplitude du signal délivré par le récepteur. Une acquisition est faite simultanément sur les voies EA0 et EA1. La première reprend le signal délivré par la sortie SA1. La seconde est reliée à la sortie de l'amplificateur.

Le code suivant génère un paquet d'onde d'enveloppe rectangulaire. La fréquence d'oscillation est $40\ kHz$.

```
import pycanum.main as pycan
from matplotlib.pyplot import *
import numpy
import math
import time
```

On commence par définir la demi-largeur du paquet et la durée totale de l'expérience. L'unité est la période d'oscillation.

```
sigma = 5
T = 1000.0
```

La fonction suivante définit le signal :

```
def signal(t):
    u=t-T/2
    if abs(u)<sigma:
        return math.cos(2*math.pi*t)
    else:
        return 0.0
```

Voici le calcul des échantillons et de la période d'échantillonnage :

```
N = 20000 # nombre d'echantillons
t = numpy.arange(N)*T/N
x = numpy.zeros(N)
for i in range(N):
    x[i] = signal(t[i])
f = 40000.0
periode = 1.0/f
fe = N/(periode*T)
te = 1.0/fe
```

La période d'échantillonnage doit être supérieure à un dizième de microseconde.

Voici la programmation de la carte SysamSP5. L'acquisition de fait en parallèle avec l'émission du signal sur la sortie. La voie EA0 reprend le signal envoyé et la voie EA1 enregistre le signal délivré par le récepteur. Comme l'amplitude en réception est très faible, on utilise un calibre de 0.1 volts. Le temps et les tensions sont enregistrés dans un fichier pour un traitement ultérieur.

```
can = pycan.Sysam("SP5")
can.config_entrees([0,1],[10.0,0.1])
can.config_echantillon(te*1e6,N)
can.ecrire(1,0.0,1,0.0) # on applique 0 V sur la sortie pendant 1 seconde
time.sleep(1)
can.acquerir_avec_sorties(x,numpy.zeros(N))
t=can.temps()
u=can.entrees()
can.fermer()

figure()
plot(t[0],u[0])
plot(t[1],u[1])
numpy.savetxt("paquet-1.txt",[t[0],u[0],u[1]])

show()
```

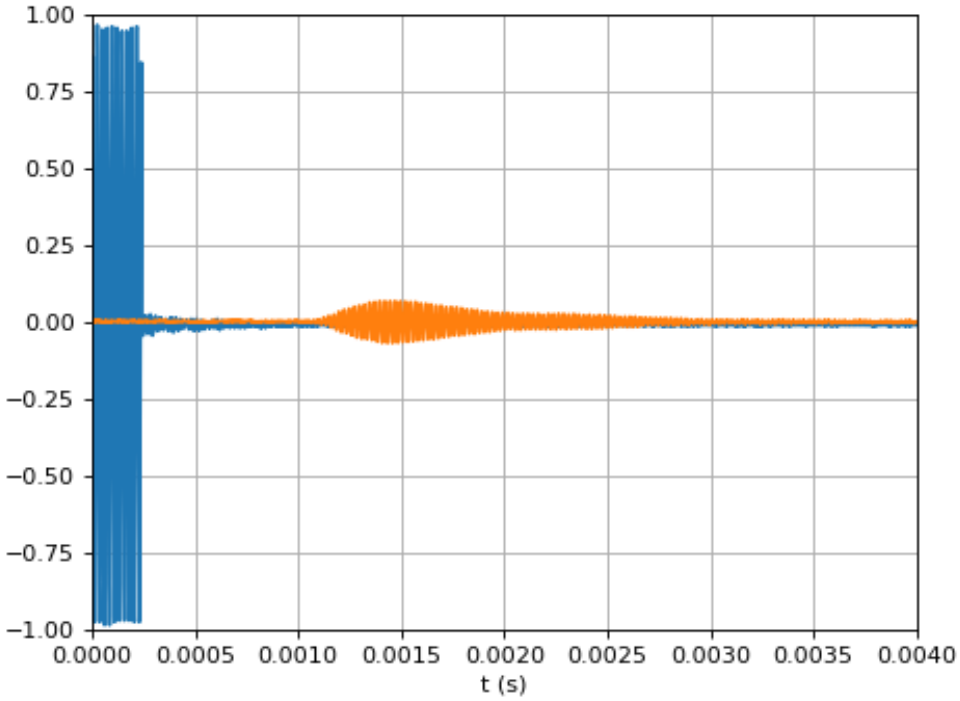
2. Traitement des données

```
from matplotlib.pyplot import *
import numpy
import math
```

Voyons le signal envoyé à l'émetteur et le signal transmis par le récepteur. On enlève les points situés avant le début du paquet.

```
[t,x,y] = numpy.loadtxt("paquet-10.txt")
debut = 9900
x=x[debut:]
y=y[debut:]
y=y-y.mean()
T = t[t.size-1]-t[0]
te=t[1]-t[0]
t = numpy.arange(x.size)*te

figure()
plot(t,x)
plot(t,y)
xlabel('t (s)')
axis([0,0.004,-1,1])
grid()
```



On voit que le signal reçu est plus étalé que le signal envoyé à l'émetteur. La montée en amplitude est progressive, ce qui rend difficile la détection du début du paquet.

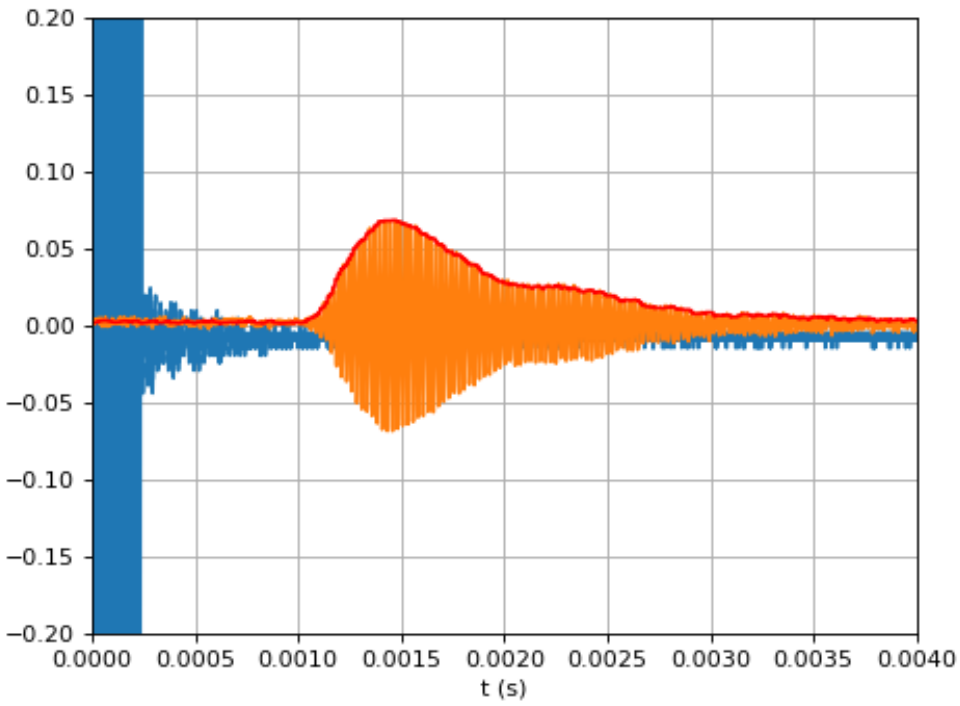
Pour obtenir le décalage entre le paquet reçu et le paquet émis, on peut rechercher le maximum d'amplitude de ce dernier. La fonction suivante renvoit l'amplitude calculée sur une fenêtre :

```
def amplitude(x,i,l):
    y = x[i:i+l]
    return y.max()-y.min()
```

On définit une largeur pour la fenêtre, et on la fait glisser sur la totalité du signal reçu, ce qui permet d'obtenir une courbe d'amplitude :

```
largeur = 30
taille = t.size
amp = numpy.array([])
temps = numpy.array([])
for i in range(largeur//2,taille-largeur//2):
    amp = numpy.append(amp,amplitude(y,i-largeur//2,largeur))
    temps = numpy.append(temps,i*te)

plot(temps,amp/2,'r')
axis([0,0.004,-0.2,0.2])
```



L'instant du maximum est obtenu en recherchant le maximum :

```
tau = te*numpy.argmax(amp)
```

```
print(tau)
--> 0.0014352
```

Voici une fonction qui effectue tout le traitement pour un fichier donné :

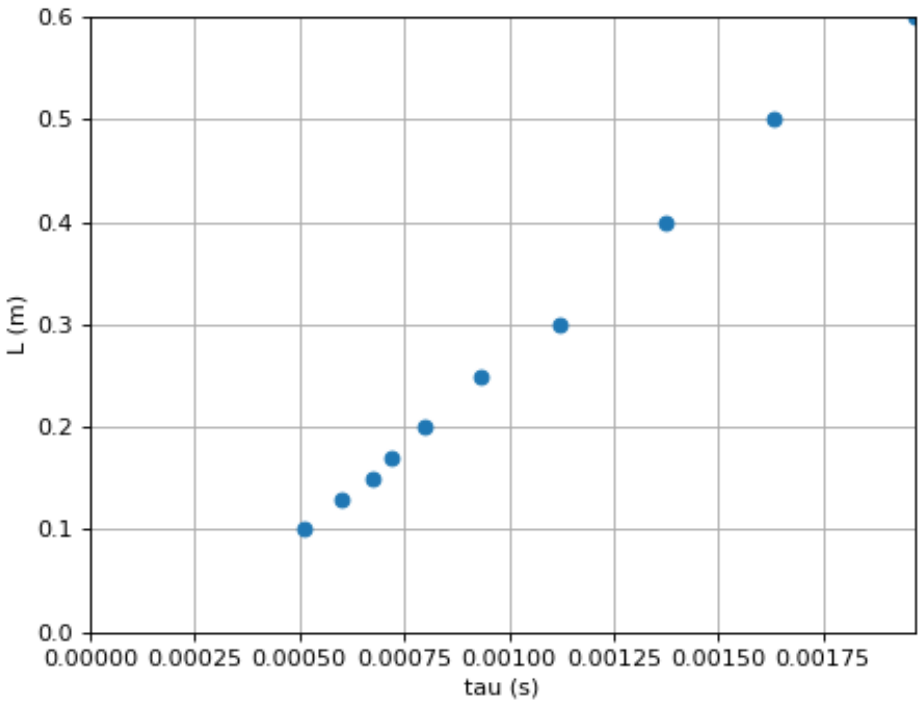
```
def traitementPaquet(numero):
    [t,x,y] = numpy.loadtxt("paquet-%d.txt"%numero)
    debut = 9900
```

```
    t=t[debut:]
    x=x[debut:]
    y=y[debut:]
    y=y-y.mean()
    te=t[1]-t[0]
    largeur = 30
    taille = t.size
    amp = numpy.array([])
    temps = numpy.array([])
    for i in range(largeur//2,taille-largeur//2):
        amp = numpy.append(amp,amplitude(y,i-largeur//2,largeur))
        temps = numpy.append(temps,i*te)
    tau = te*numpy.argmax(amp)
    return tau
```

Voici le résultat pour différentes distances :

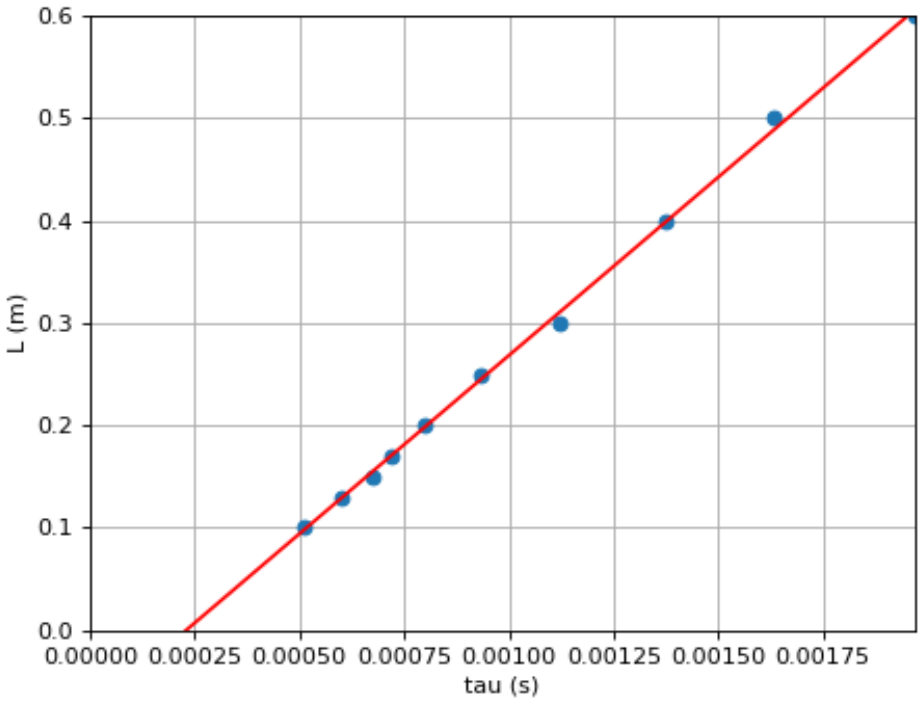
```
L = numpy.array([100,130,150,170,200,250,300,400,500,600])*1e-3
numeros = [7,8,9,10,11,12,13,14,15,16]
tau = []
for n in numeros:
    tau.append(traitementPaquet(n))
tau = numpy.array(tau)/2

figure()
plot(tau,L,'o')
xlabel('tau (s)')
ylabel('L (m)')
grid()
axis([0,tau.max(),0,L.max()])
```



La fonction `scipy.optimize.leatsq` permer de faire une régression linéaire :

```
from scipy import stats
a, b, r_value, p_value, std_err = stats.linregress(tau,L)
tau_m = tau.max()
plot([0,tau_m],[b,a*tau_m+b],'r')
```




```
print((a,b,std_err))
--> (348.15425580631586, -0.0799427589073411, 4.580711760692226)
```

Le premier paramètre donne la vitesse du son. Le troisième est l'écart-type de ce paramètre. La vitesse du son dans l'air à pression atmosphérique est fonction de la température :

$$c = c_0\sqrt{\frac{T}{273}} \tag{1}$$

où T est la température absolue en Kelvin et $c_0 = 331,6\ m/s$ la vitesse du son à zéro degrés celcius. L'expérience est réalisée à environ 22 degrés, ce qui fait une vitesse de $345\ m/s$. La valeur obtenue par ce traitement est légèrement plus grande (écart de moins de 1 pour cent). La validité de la méthode repose sur l'hypothèse d'une durée fixe entre le début du paquet et son maximum. En refaisant le traitement avec des valeurs différentes de la largeur de fenêtre (10,20,40), on obtient la même vitesse aux décimales près.

La connaissance des paramètres de la droite permet d'utiliser le dispositif comme télémètre.