ANNEXE : Programme à découper

a) **Afficher le graphe distance en fonction du temps.**

```
celerite.plot(liste_temps,liste_distance)
```

b) **Traiter les données enregistrées sur les voies gauche et droite.**
Afficher les instants auxquels le son est reçu sur chaque voie.

```
tL=celerite.traitement_gauche()
print("Le signal gauche est reçu à l'instant tL=", tL,'s')
tR=celerite.traitement_droit()
print("Le signal droit est reçu à l'instant tR=", tR,'s')
```

c) **Démarrer la lecture du bip.**

```
celerite.lecture(''bip.wav'')
```

d) **Demander à l'utilisateur si la mesure doit être gardée.**
Si la réponse est non, retirer les valeurs de distance et de temps.
Sinon, ajouter 1 à la valeur de j.

```
print('garder mesure? o ou n')
eff=input()
if eff=='n':
    liste_temps.remove(temps)
    liste_distance.remove(distance)
    print("mesure effacée")
else:
    j=j+1
```

e) **Tant que j inférieur ou égal au nombre de mesures :**

```
While(……..) :        # À compléter
```

f) **Enregistrer pendant 1s les voies gauche et droite.**

```
celerite.enregistre("1")
```

g) **Déclarer les listes contenant les valeurs de durées et de distances.**
Définir le nombre n de mesures et initialiser la variable j à 1 qui définit le compteur de mesures.

```
liste_temps=[]
liste_distance=[]
n=…..    # À compléter
j=1
```

h) **Calculer la vitesse et afficher sa valeur**

```
vitesse=…..    # À compléter
print(……..)
```

i) **Demander à l'utilisateur d'entrer la distance d puis ajouter la valeur à la liste correspondante.**

```
distance=celerite.entrer_distance(j)
liste_distance.append(distance)
```

j) **Calculer la durée s'écoulant entre les instants de réception sur chaque voie puis ajouter la valeur à la liste correspondante. Afficher sa valeur.**

```
temps= ……    # À compléter
liste_temps.append(temps)
print("La différence entre les deux:",temps,'s')
```