Thème: Ondes et signaux : Produire un son

Niveau: Seconde

TI-83 Premium CE

python™

Fiche méthode

Eric Tixidor

Référentiel, compétences

Capacités exigibles :

- Utiliser un dispositif comportant un microcontrôleur pour produire un signal sonore.
- Réaliser une série de mesures.

Commentaires de l'auteur

On ajoutera pour cette séance la création d'un télémètre à ultra-sons qui s'ajoutera au dispositif sonore.

- Capacités numériques : Utiliser les entrées et sorties d'un microcontrôleur, en langage python.
- Capacités mathématiques : définir et programmer une fonction affine et une fonction racine carrée.

Materiel

- Calculatrice TI-83 Premium CE Edition Python
- Le microcontrôleur TI-Innovator™ Hub et son haut-parleur intégré.
- Un émetteur-récepteur à ultra-sons avec connectique groove, qui sera branché sur le port **`IN 1`** du Hub.

Enoncé

On souhaite créer un dispositif autonome de type « radar de recul ». Lorsque la distance à un obstacle diminue, ce dispositif devra non seulement émettre un son de fréquence plus aiguë, mais aussi avec des bips plus rapprochés.

Enfin, il serait préférable que ces variations de fréquences sonores soient plus marquées pour des obstacles se situant à moins de 1,0m (c'est à dire, entre la distance de contact et 1,0m). Les variations sonores pourront être moins importantes pour de plus grandes distances.

Prérequis

Le lecteur de cette fiche devra être déjà familiarisé avec l'environnement python de la calculatrice.

Proposition de résolution

Mesurer une distance avec un télémètre à ultra-sons

On utilisera le capteur de type émetteur-récepteur à ultra-sons, que l'on branchera sur le port **IN1** du hub-innovator.

Dans l'éditeur python de la calculatrice : créer un nouveau script que l'on appellera : **SON**
Importer les librairies nécessaires pour :

- Utiliser le capteur à ultra-son :
  Modules **6:ti_hub** puis **2:dispositifs d'entrée 2:Ranger** ce qui renvoie : **from ranger import ***
- Ecrire une boucle non bornée sur la fonction escape( ) :
  Modules **4:ti_system** puis 1:**from ti_system import ***
- Ecrire dans une fenêtre graphique sur l'écran de la calculatrice :
  Modules **5:ti_plotlib** puis 1:**import ti_plotlib as plt**

```
from ti_system import *
import ti_plotlib as plt
from ranger import *
```

On définit une fonction **screen** qui permettra un affichage dans la fenêtre graphique à l'aide de la librairie **ti_plotlib**.
Cette fonction prendra en paramètre :

- ✓ **y** : Le numéro de la ligne à laquelle le message s'affichera
- ✓ **val** : la valeur numérique qui complète le texte affiché
- ✓ **msg** : le texte à afficher, qui comprend une règle de formatage lui permettant de remplacer les caractères **%.3f** (réel) ou **%.0f** (entier) par **val**.

```
def screen(y,val,msg):
    plt.text_at(y,msg%val,
    "center")
```

Saisir ensuite les instructions qui permettront :

- ✓ de déclarer l'objet associé au port groove **IN 1**.
- ✓ d'effacer l'écran

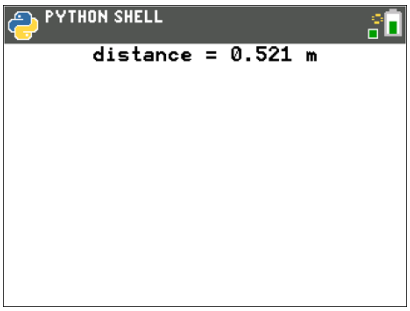
```
sr=ranger("IN 1")
plt.cls()
```

On complète enfin le corps du programme avec une boucle non bornée qui s'exécute tant que l'utilisateur n'appuie pas sur la touche 

.
Dans cette boucle, le programme permet de relever les mesures de distance, en continu, et avec un affichage sur l'écran de la calculatrice.

```
while not escape( ):
    d = sr.measurement( )
    msg="distance = %3.f m"
    screen(1,d,msg)
    sleep(0.2)
```

Brancher l'émetteur-capteur à ultra-sons sur le port **`IN 1`** du Hub.
Diriger vers un obstacle et mesurer la distance.
La calculatrice affiche sa distance en mètre.



La plage de mesure annoncée par la notice du capteur est de 2cm – 400cm avec une résolution de 1cm.

Les valeurs retournées semblent conformes.

Remarque : Il peut y avoir parfois quelques *exceptions*, et la valeur renvoyée peut être soudainement égale 10,0m.
Il faudra considérer ce point pour le calcul de la fréquence sonore.

Fonction de calcul de la fréquence sonore

Le haut-parleur intégré émet des sons de fréquences 200 à 4500 Hz environ. On cherche une fonction qui fait correspondre les valeurs de distances avec celles des fréquences, jusqu'à d = 1,0m.
Les obstacles seront jugés préoccupant lorsque la distance est inférieure à 1m.

Cette fonction doit être décroissante si l'on veut une ***fréquence plus élevée*** (son plus aigu) lorsque ***l'obstacle est rapproché.***

- Une première idée serait de programmer une fonction **freq** affine à partir des correspondances suivantes :

d(cm)	0	100
f(Hz)	4500	200

On pourrait alors définir la fonction suivante :

```
def freq(d):
    return 4500-4300*d
```

Et, afin d'éviter de retourner des valeurs négatives pour **`f`** (voir la remarque plus haut sur les *exceptions* qui génèrent des valeurs de d>1), on ajoutera la ligne suivante :

```
if d>1 : d = 1
```

- Une deuxième idée serait de définir une fonction **freq**, dont les *variations* seraient plus importantes pour des plus petites valeurs de d. Cela créé un effet *critique* lorsque l'obstacle se rapproche.

Ce serait alors une fonction non linéaire en **d** (c'est-à-dire dont la représentation graphique n'est pas une droite). Une possibilité est d'utiliser la racine carrée de d :

```
def freq(d):
    if d>1 : d=1
    return 4500-4300*d**0.5
```

On pourra saisir cette deuxième fonction à la suite des imports de librairies dans le programme.

Programmer la sortie sonore

Dans l'éditeur du script, importer la librairie nécessaire pour :

- Utiliser haut-parleur intégré : **Modules 6:ti_hub 1:dispositifs intégrés du Hub** puis **3:Sound**

```
import sound
```

On ajoutera alors dans la boucle non bornée :

- L'instruction qui permet de calculer la fréquence sonore en fonction de la distance mesurée :

```
f = freq(d)
```

- La chaine formatée associée à la variable msg2 :

```
msg2 = "freq = %.0f Hz"
```

- L'appel de la fonction **screen** qui permettra d'afficher la fréquence sonore jouée par le Hub.

```
screen(2,f,msg2)
```

- L'instruction qui permet de jouer un son :

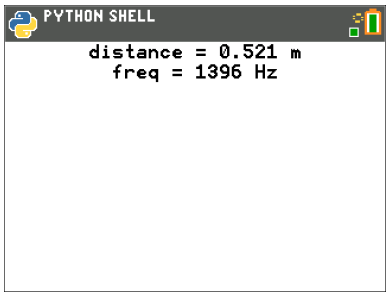
```
sound.tone(frequence,duree)
```

La fonction **sound.tone** prend 2 paramètres :

- ✓ la fréquence sonore.
- ✓ la durée du son. Avec une durée de 0.1s, le son ressemble plus à un Bip.

Enfin, on modifiera le paramètre de la fonction **sleep** en écrivant **sleep(d/2) :** c'est une valeur croissante avec la distance. Cela rajoutera à l'effet « *critique* » lorsque l'obstacle se rapproche, les Bips étant alors plus rapprochés.

Le programme mesure alors la distance, joue un son (ou plutôt une série de Bips sonores), et affiche la distance et fréquence de la note



jouée.
La durée qui sépare deux Bips sonores (c'est à dire la période d'émission des Bips sonores) est plus courte lorsque l'obstacle est plus proche.

Script complet

ÉDITEUR : SON
LIGNE DU SCRIPT 0001

```
from ti_system import *
import ti_plotlib as plt
from ranger import *
import sound

def screen(y,val,msg):
  plt.text_at(y,msg%val,"center"
      )
def freq(d):
  if d>1:d=1
  return 4500-4300*d**0.5

sr=ranger("IN 1")
plt.cls()

while not escape():
  d=sr.measurement()
  msg="distance = %.3f m"
  screen(1,d,msg)
  f=freq(d)
  msg2="freq = %.0f Hz"
  screen(2,f,msg2)
  sound.tone(f,0.1)
  sleep(d/2)
```

Fns… | a A # | Outils | Exéc | Script

Documents et script à télécharger à l'adresse :
https://education.ti.com/fr/physique-chimie