7
Mission InSIGHT - CCP 2019 Sciences industrielles MP

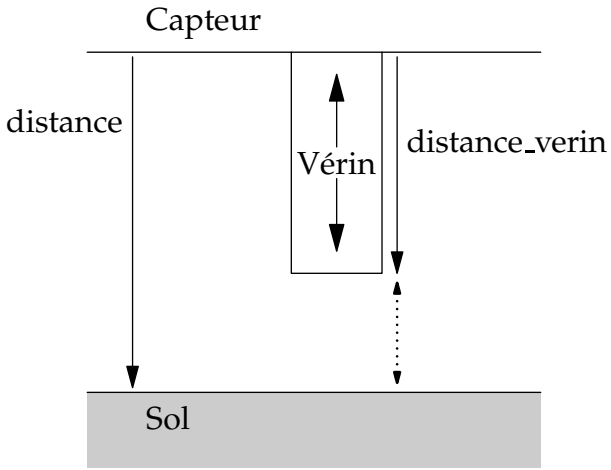
I

Analyse de la loi de commande

1. Comme d'habitude, la principale difficulté est d'arriver à lire vite et très précisément le sujet. Il est utile de prendre des notes en lisant et de s'aider de schémas, pour bien comprendre ce qu'on doit faire.

2. **Q26**

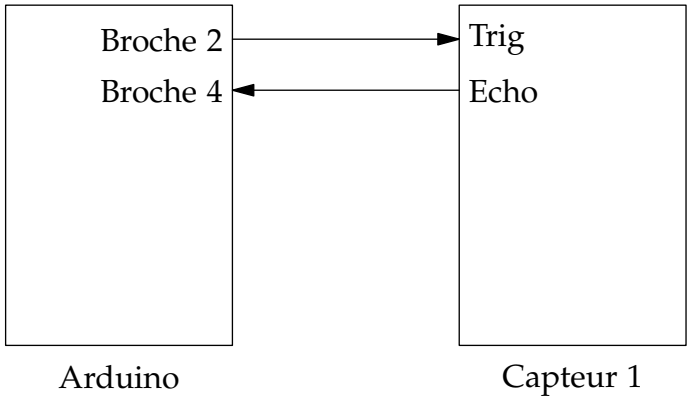
Les deux arguments sont exprimés en *centimètres*, donc la distance de référence, donnée en *millimètres*, doit être convertie en centimètres.




```
def consigne(distance, distance_verin):
    ecart = distance - distance_verin
    if (ecart > 1.0):
        return rapide
    else:
        return lente
```

3. Q27

En lisant bien l'énoncé, on constate que la broche 2 est une sortie de la carte Arduino (reliée à une entrée du capteur 1) alors que la broche 4 est une entrée (reliée à une sortie du capteur 1).



Il n'y a pas d'autre difficulté dans cette question...

```
def setup():
    pinMode(2, OUTPUT)    # vers Trig
    pinMode(4, INPUT)     # depuis Echo
    digitalWrite(2, LOW)
```

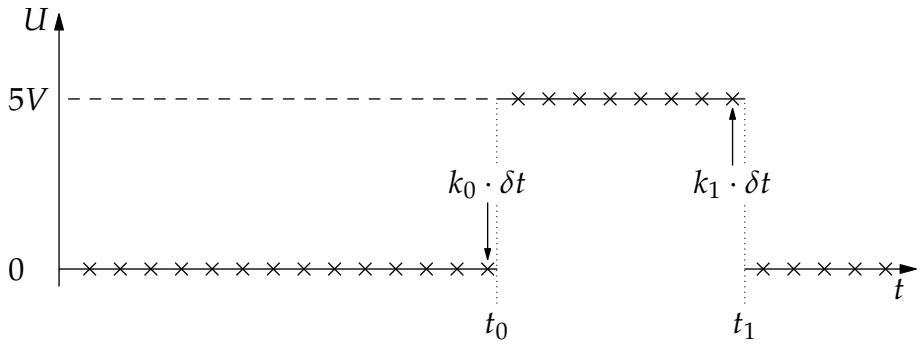
4. Q28

Il faut deviner que la commande `digitalWrite` agit comme un interrupteur : seule une seconde commande `digitalWrite` peut modifier l'effet d'une première commande `digitalWrite`.
La durée de 20 microsecondes doit être convertie en secondes pour être donnée en argument à la fonction `time.sleep`.

```
def impulsion(S):
    digitalWrite(S, HIGH)   # la tension imposée à l'entrée du capteur passe à 5V
    time.sleep(2e-5)        # attente de 20 microsecondes
    digitalWrite(S, LOW)    # la tension revient à 0V (fin de l'impulsion)
```

5. Q29

Il faut comprendre que les deux boucles `while` effectuent en alternance deux opérations : mesurer la tension sur la broche E (avec `digitalRead`) et noter la date (avec `time`). On peut considérer que la durée nécessaire à la réalisation de ces deux opérations est constante, égale à δt.



Il s'agit donc de mesurer la durée $(t_1 - t_0)$ de l'impulsion en réalisant des mesures aux instants $k \cdot \delta t$.

5.1 La première boucle

```
while (digitalRead(E)==0):
    pulse_start = time.time()
```

note la date (en secondes depuis *the Epoch*, c'est-à-dire le 1er janvier 1970) tant que la tension mesurée sur la broche E est nulle (`LOW`) : la variable `pulse_start` prend successivement les valeurs δt, $2\delta t$, …, $k_0 \cdot \delta t$.
À l'instant $(k_0 + 1) \cdot \delta t$, la tension mesurée est passée à $5V$ et on sort alors de la boucle. La valeur définitive de `pulse_start` est donc $k_0 \cdot \delta t$, qui est une approximation à δt près par défaut de la date t_0 de début de réception d'un signal `HIGH`.

5.2 On entre aussitôt dans la deuxième boucle qui fonctionne sur le même principe.

```
while (digitalRead(E)==1):
    pulse_end = time.time()
```

Tant que la broche reçoit une tension `HIGH`, on note la date. La variable `pulse_end` prend donc successivement les valeurs $(k_0 + 1) \cdot \delta t$, …, $k_1 \cdot \delta t$.
À l'instant $(k_1 + 1) \cdot \delta t$, la tension mesurée est revenue à `LOW` et on sort de la boucle. La valeur définitive de `pulse_end` est donc $k_1 \cdot \delta t$, approximation à δt près par défaut de la date t_1 de fin de réception du signal `HIGH`.

5.3 La différence entre ces deux dates

```
pulse_duration = pulse_end - pulse_start
```

exprime donc (en secondes) la durée pendant laquelle la broche a reçu une tension `HIGH`, c'est-à-dire la durée de l'impulsion. (Il est toujours utile de connaître un peu d'anglais.)

5.4 La différence entre la durée réelle $t_1 - t_0$ et la durée mesurée $(k_1 - k_0) \cdot \delta t$ est de l'ordre de δt. Il est donc important que la durée δt soit de l'ordre de la microseconde.

5.5 Enfin, la valeur

```
distance = pulse_duration*17150
```

donne, selon les indications de l'énoncé, la distance mesurée par le capteur.

5.6 D'après l'énoncé, c'est la broche 4 de la carte Arduino qui doit recevoir le signal. Il faut donc choisir `E=4`.

5.7 Si le facteur de proportionnalité 17 150 vous intrigue, il suffit de se souvenir qu'il s'agit d'une vitesse exprimée en centimètres par seconde, soit 171,5 mètres par seconde.
La vitesse du son dans l'air est voisine de 340 mètres par seconde comme chacun sait, soit environ le *double* de la vitesse précédente.
Il est facile d'en déduire que le capteur impose un signal `HIGH` pendant toute la durée qui sépare l'émission des 8 impulsions ultrasonores de leur réception. Pendant cette durée, les ultrasons ont en effet parcouru deux fois la distance qui sépare l'émetteur-récepteur de l'obstacle sur lequel le signal s'est réfléchi.

6. Q30

On doit demander à la broche 2 d'émettre une impulsion vers l'entrée `Trig` du capteur, puis de calculer la distance qui sépare le capteur de l'obstacle en fonction du signal reçu par la broche 4.

```
def mesure():
    impulsions(2)
    mes = calcul_distance(4)
    return mes
```

II

Exploitation de la base de données du système de mesures

7. Q31

Dans chaque table, toutes les mesures sont numérotées et c'est le numéro de chaque mesure qui l'identifie. Les clés primaires des tables **Sismique**, **LargeBande** et **CourteBande** sont respectivement `Mesure`, `Id_LB` et `Id_CB`.
S'il n'y a pas plus d'une mesure par minute, on pourrait aussi se servir du champ **Date** comme d'une clé primaire.

8. Q32

La requête

```
SELECT DATE FROM CourteBande
ORDER BY MCBx
```

renvoie les dates des mesures qui se trouvent dans la table **CourteBande** classées par ordre croissant selon la valeur de **MCBx**.

9. Q33

Champ à extraire : **MCBz**.
Table à explorer : **CourteBande**.
Filtre à appliquer : on ne retient que les valeurs supérieures à $0,2$.
Classement des données : par ordre croissant selon le champ **Date**.
Cela nous donne la requête suivante.

```
SELECT MCBz FROM Courtebande
WHERE MCBz>0.2
ORDER BY DATE
```

10. Q34

Les trois champs **MLB*** se trouvent dans la table **LargeBande** et le champ **Temperature** se trouve dans la table **Sismique**. Il faut donc opérer une jointure entre ces deux tables, en fonction du seul champ commun : **Date**.

```
SELECT LB.MLBx, LB.MLBy, LB.MLBz FROM LargeBande AS LB
JOIN Sismique AS S ON S.DATE=LB.DATE
WHERE S.Temperature>-30.
```

Puisque le champ sur lequel s'effectue la jointure porte le même nom dans les deux tables (et que c'est le seul champ dans ce cas), on pourrait aussi écrire le code suivant.

```
NATURAL JOIN Sismique AS S
```