TP5 : Contrôler la position, fiche outil de l'activité 3

COMMENT CHOISIR LES TYPES DE VARIABLES EN PROGRAMMATION (ARDUINO, PYTHON, C…) ?

Pour programmer, il faut utiliser plusieurs sortes de données, les **«TYPES », p**ar exemple :

- pour incrémenter un compteur, une variable de type entier convient,
- pour calculer la moyenne d'une série de données fournies par un capteur, il faut un nombre réel,
- pour tester une condition, on peut utiliser un simple booléen.

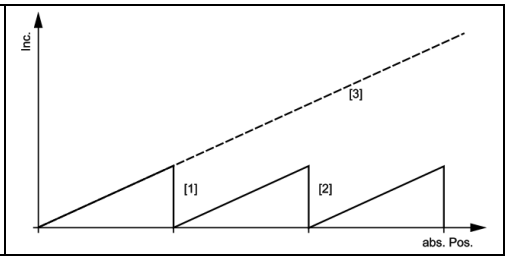
Les principaux types :

- **void** : mot clef utilisé pour spécifier qu'il n'y a **pas de variable**, par exemple une fonction qui ne renvoie pas de paramètre (ou qui n'en prend pas)
- **boolean** : variable logique qui ne prend que **deux états : false, true**. Sert à **tester des conditions**, des **états binaires**.
- **char** : traditionnellement, le **char** est utilisé pour les caractères. En fait, il s'agit d'une variable codée sur **un octet**, qui peut donc prendre **256 valeurs** (2^8) différentes. Les caractères ayant longtemps été indexés sur une table à 256 entrées, mais elle peut être utilisée pour toute variable dans l'**intervalle [-128 ; 127]**.
- **unsigned char** : le mot **unsigned** signifie que l'on considère des **valeurs positives** uniquement. Ainsi, **char** permet de représenter des valeurs allant de -128 à +127, et **unsigned char** utilise quant à lui l'intervalle **[0, 255]**.
- **byte** : le **byte** est un synonyme du **unsigned char**, c'est-à-dire qu'il se code sur **un seul octet** et prend des valeurs pouvant aller **de 0 à 255**. Il est particulièrement **adapté pour les sorties PWM**.
- **int** : un **int** (pour **integer**) est un **entier relatif** codé sur **deux octets**, ce qui signifie qu'il possède **2^{16} valeurs différentes**. Il peut prendre des valeurs comprises dans l'**intervalle [-32,768 ; 32,767]**.
- **unsigned int** : c'est un **entier positif** codé sur **deux octets**, prenant ses valeurs dans l'**intervalle [0 ; 65,535]**.
- **word** : synonyme de **unsigned int**.
- **long** : **entier relatif** codé sur **4 octets**, pouvant donc prendre des valeurs allant de **-2 147 483 648 à 2 147 483 647**.
- **unsigned long** : **entier positif** codé sur **4 octets**, prenant dès lors ses valeurs dans l'**intervalle [0 ; 4 294 967 295]**.
- **float** : il s'agit d'une variable de **type réel**, codée sur **4 octets**. Sa précision n'excède pas **6 ou 7 décimales**.
- **double** : traditionnellement, le type **double** offre une précision deux fois supérieure à celle du **float**. Dans le cas du microcontrôleur Arduino, les deux types sont synonymes : **double = float**.
- **String** : le type **String** n'est plus un type, mais un **Objet** (programmation orientée objet). Il s'agit d'une **chaîne de caractères**, dans laquelle on stocke des mots, des phrases... **String** permet de **traiter, comparer, manipuler**, ... les chaînes de caractères.
- **Array** : ce n'est pas un type à proprement parler. Il est possible de déclarer **pour chaque type** un **tableau de données** de ce type, un tableau d'entiers par exemple pour stocker des états, ou un tableau de booléens, pour caractériser une série de données. On utilisera pour cela les **crochets [ ]** placés après le nom de la variable.

QUE SIGNIFIE DEBORDEMENT D'UN COMPTEUR (OVERFLOW) ?

Il y a dépassement de la capacité du compteur, lorsque le compteur passe par exemple en hexadécimal pour un octet de FF_h à 00_h soit 1111 1111 à 0000 0000 en binaire (débordement positif ou supérieur).
Conséquence pour un codeur de position (figure ci-contre)

(1) Premier dépassement codeur
(2) Second dépassement codeur
(3) Position visible par l'utilisateur

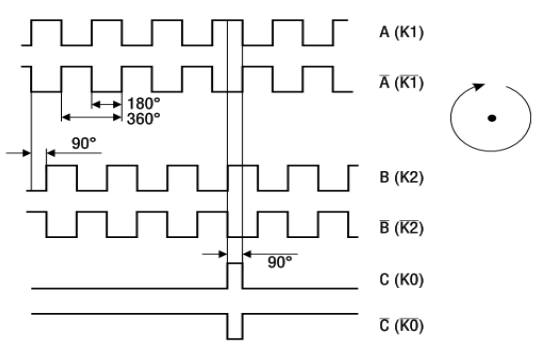


COMMENT TRAITER DU SENS DE ROTATION D'UN CODEUR INCREMENTAL ?

Un codeur de position incrémental possède au moins 2 voies A et B décalées d'un quart de période entre elles.

Quand il s'agit d'un codeur industriel, il possède en plus une voie Z (zéro) qui délivre une impulsion par tour et le complément des voies A, B, Z soit A/, B/ et Z/.

C'est le traitement de ces signaux qui permet d'augmenter la résolution du codeur et de connaitre son sens de rotation.



Détection du sens de rotation,

Extrait de la routine « Encoder » (sous-programme) appelée par l'instruction « include Encoder » en début de programme.

Toutes les combinaisons possibles sont traitées, ainsi lorsque l'arbre de rotation oscille autour d'un point (vibrations) il n'y a aucun risque de mauvaise information.

```
//
//                Pin1 ______|        |________|        |______ Pin1
// negative <---                _______          _______
//         --> positive
//                Pin2 __|        |________|        |________|   Pin2

          //    new   new   old   old
          //    pin2  pin1  pin2  pin1  Result
          //    ----  ----  ----  ----  ------
          //    0     0     0     0     no movement
          //    0     0     0     1     +1
          //    0     0     1     0     -1
          //    0     0     1     1     +2  (assume pin1 edges
only)
          //    0     1     0     0     -1
          //    0     1     0     1     no movement
          //    0     1     1     0     -2  (assume pin1 edges
only)
          //    0     1     1     1     +1
          //    1     0     0     0     +1
          //    1     0     0     1     -2  (assume pin1 edges
only)
          //    1     0     1     0     no movement
          //    1     0     1     1     -1
          //    1     1     0     0     +2  (assume pin1 edges
only)
          //    1     1     0     1     -1
          //    1     1     1     0     +1
          //    1     1     1     1     no movement
/*
```

COMMENT NE PAS RATER D'IMPULSION A COMPTER EN UTILISANT LES ENTREES D'INTERRUPTION ?

Comme son nom l'indique, **une entrée d'interruption** interrompt le déroulement des calculs sur le microcontrôleur pour effectuer un traitement spécifique, en l'occurrence la mise à jour du compteur d'impulsions, avant de rendre la main à la boucle principale. Ceci permet de prendre en compte des évènements fugitifs d'une durée très inférieure à toutes boucles de programme.

- Sur une carte Arduino MEGA ATK, il y a plusieurs lignes d'**Interruptions Externes** : Broches 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), et 21 (interrupt 2).
- Pour une Arduino UNO il y en a deux numérotées 0 et 1.

Ces broches peuvent être configurées pour déclencher une interruption sur une valeur basse, sur un front montant ou descendant, ou sur un changement de valeur.

C'est l'instruction attachInterrupt qui sera associée à ces interruptions dans le programme.

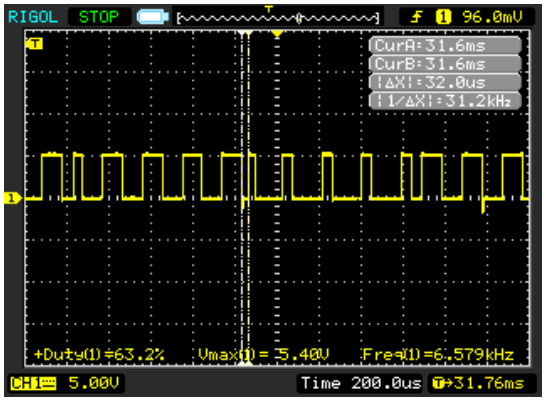
COMMENT OBTENIR UNE VITESSE AVEC UN CODEUR INCREMENTAL ?

La variation de vitesse d'un codeur incrémental se traduit par une variation instantanée de la période entre 2 impulsions d'une voie (figure ci-contre).

Sachant que la vitesse angulaire Ω est la dérivée par rapport au temps de la position angulaire, il suffit de se fixer un temps de comptage invariable (base de temps Tc) à l'intérieur duquel on compte les impulsions issues du codeur.

Si la vitesse est élevée il y aura beaucoup d'impulsions dans le temps Tc (N >>). Si le système est arrêté, il y aura aucune impulsion, N=0 soit Ω = 0.

Un compteur de vélo utilise ce principe (1 impulsion par tour de roue), le système ABS automobile également.



Exemple pour un codeur ayant 200 impulsions par tour, et une base de temps Tc = 0,1s :

- une vitesse du codeur est de 1tr/s ou 60tr/min. donne N = 20,
- la vitesse minimale mesurable correspond à 1 impulsion en 0,1s soit 360 impulsions = 1tour en 36s ou (60/36) tr/min = 1,67 tr/min
- La vitesse maximale est seulement limitée par la fréquence maxi acceptable par l'entrée de comptage.