TP C++ sur PC
Durée: 6h + 1h de préparation

TP 6 - Calcul de consigne de barre

Objectif :

- Développer un programme destiné à récupérer des informations issues d'un encodeur incrémental.
- Utiliser les classes personnalisées de l'extension PiFace sur Raspberry .
- Analyser la notice technique constructeur afin de concevoir une nouvelle classe objet CEncInc dérivée de la classe PiFaceIO.

But :

Créer la partie calculateur qui lit la consigne de barre issue du codeur incrémental. Cette application doit donc effectuer la mise à jour de la consigne. Cette variable est incrémentée ou décrémentée selon le sens de rotation du codeur incrémental.

Les 2 signaux issus du codeur sont reliés sur les entrées I1 et I2 de la carte PiFace du Raspberry.

Remarque : On peux simuler les signaux en utilisant les boutons poussoirs S1 et S2 de la carte PiFace.

Critères d'évaluation:

Vous serez évalués sur les critères suivants:

* Votre autonomie à résoudre les différents problèmes énoncés.
* Votre capacité à mettre en oeuvre le travail étudié.
* Votre capacité à utiliser le matériel et la notice mis à votre disposition.
* La rédaction de l'algorithme et du programme C++ de mise en oeuvre.

Travail demandé

Développer un programme en C++ qui permet d'afficher, en hexadécimale dans un objet QLabel, la valeur présente sur les entrées de la carte PiFace du Raspberry.

Capteurs numériq

Codeur_incrémental.pdf
Codeur_absolue.pdf

1) Calcul de la consigne de barre :

Cmc1, Cmc2: signaux fournis par le codeur incrémental câblés respectivement sur les entrées I1 et I2 de la carte PiFace du Raspberry connectée. Le déphasage de l'un par rapport à l'autre permet d'identifier le sens de la rotation du codeur et donc le signe de la modification à apporter à la consigne de barre. L'évolution de cette modification est déterminée en comptant le nombre de périodes.

Sortie : Consigne mot binaire de 8 bits dont la valeur est représentative de la consigne de barre.

0 : barre en butée à bâbord
127 : barre dans l'axe du bateau
255 : barre en butée à tribord

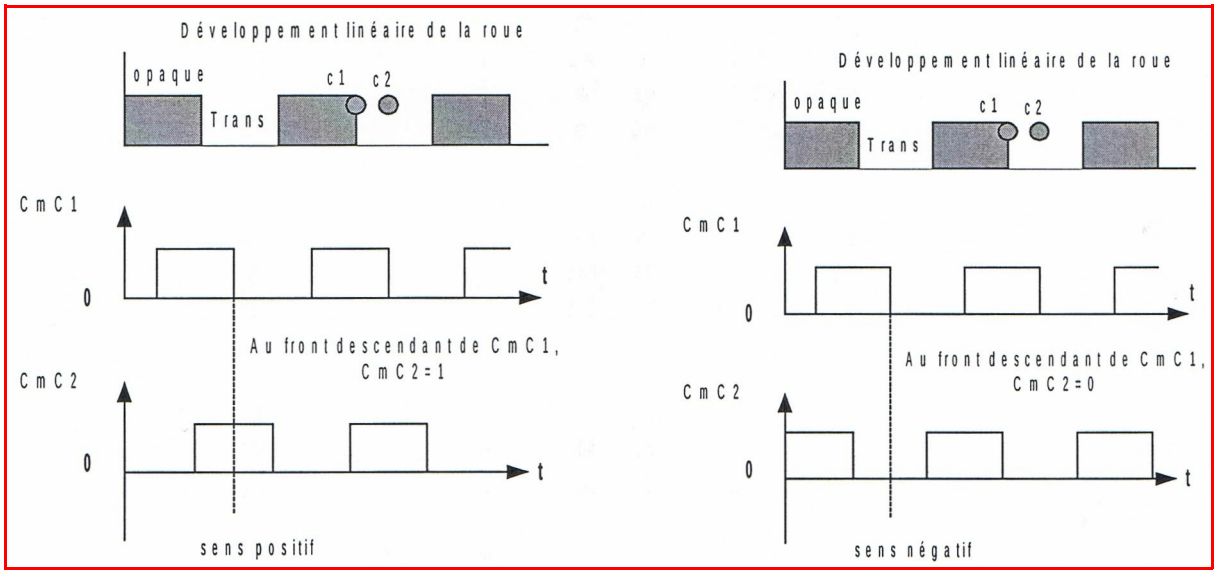
2) Fonctionnement :

A chaque front descendant de Cmcl, on doit lire le niveau présent sur Cmc2.

Si Cmc2 = 1 la consigne est incrémentée.
Si Cmc2 = 0 la consigne est décrémentée.

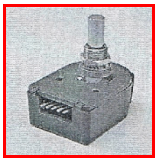
La valeur de la consigne doit rester comprise entre 0 et 255.

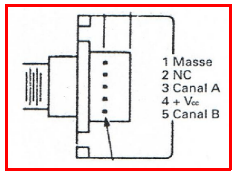
Lors de la mise sous tension, la consigne doit permettre de ramener la barre dans l'axe du bateau.



2) Spécifications techniques et montage:

a) Encodeur incrémental optique de panneau pour montage sur circuits imprimés, 500 cycles par révolution. Vitesse de rotation: 2000 tr/mn max. Tenue aux vibrations: 20 G (20 Hz à 2 kHz) Température d'utilisation: -20°C à +85°C max. Durée de vie: 12 x 106 révolutions.





b) Relier la carte encodeur à la carte PiFace du Raspberry tel que :

Encodeur incrémentale optique	VSS (DB25: patte 25)	cmc1 (DB25: patte 10)	cmc2 (DB25: patte 11)
PiFace	0v (Gnd)	I1	I2

3) Programme:

a) Créer une nouvelle classe **"CEncInc"** qui dérive de la classe "**PiFaceIO**".

Pour cela , créer, sous Qt Creator un nouveau projet "Application graphique Qt, ajouter les fichiers **piface.h** et **piface.cpp** ainsi que les fichiers **.h** et **.cpp** de la classe **PiFaceIO** puis créer la nouvelle classe **CEncInc** contenant les données membres privées:

unsigned char consigne ; // qui mémorise la valeur consigne
bool cmc1, cmc2 ; // valeurs de sortie de l'encodeur incrémenteur

et des méthodes membres publiques suivantes :

CEncInc ( ) ; // constructeur qui met la valeur de la consigne à celle qui correspond à l'axe du bateau.

unsigned char getConsigne (void) ; // getter permettant d'obtenir la valeur consigne.

bool Lit_Sens ( ) ; // méthode qui retourne un "0" si l'axe tourne à bâbord , un "1" si l'axe tourne à tribord. ("0" pour sens décroissant et "1" pour sens croissant)

bool Lit_Rotation ( ) ; // effectue la lecture des entrées I1 et I2 et si nécessaire mettre à jour la consigne et la mise à jour de cmc1. Retourne "1" si une rotation du codeur a eu lieu, "0" dans le cas contraire.

- Ecrire l'application sous Qt permettant de tester toutes les méthodes de la classe **"CEncInc"** et qui affiche la consigne de barre, en temps réel. On utilise un objet QTimer pour lire le plus souvent possible, les entrées I1 et I2 de la carte PiFace.

b) Améliorer votre programme en remplaçant l'objet QTimer par un Thread (application de fond de tâche qui fonctionne en parallèle) pour détecter en permanence les changements des sorties de l'encodeur incrémentale optique.

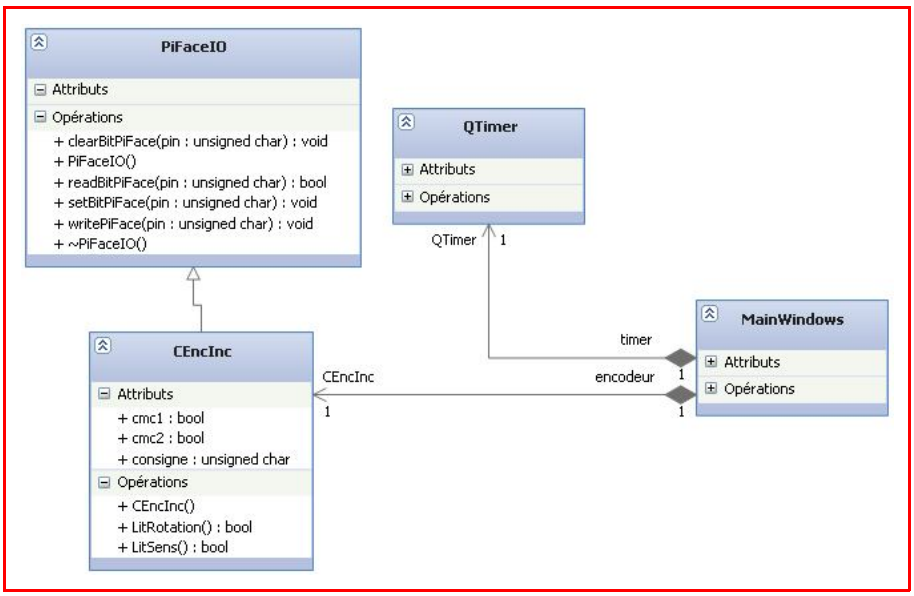
Qu'est qu'un thread ?

- C'est une partie de code qui s'exécute en parallèle au code principal de notre application.
- Le système d'exploitation (sous les systèmes monoprocesseur) "partage" le temps processeur entre les différents threads en donnant régulièrement accès à chacun d'eux au processeur pendant un très court instant, ce qui nous donne l'impression que les deux parties de code s'exécutent simultanément. Cela peut être utile pour exécuter des processus lents sans bloquer l'application.

Utiliser la classe **ClassThread** et des exemples sur internet pour mettre en oeuvre votre thread.

Diagrammes de Classe:

- Programme avec timer.



- Programme avec avec une application de fond (Thread).