

**Réalisation d’un prototype**

**Première partie :**

Vérin électrique asservi

**Seconde partie :**

Pilote automatique de péniche

Le travail s’effectue par îlots de 3 élèves :

* Un chef de projet qui coordonne le travail, récupère les résultats et rédige le compte rendu.
* Un expérimentateur qui programme la carte Arduino
* Un Modélisateur qui élabore les modèles Scilab et effectue le réglage du gain

# Compétences attendues

**ANALYSER (S1-S2-S3)**

|  |  |  |
| --- | --- | --- |
| DC1 Systèmes et informations | **Architecture générale d'un produit** · Analyse structurelle et comportementale ; · Chaîne d'information, chaîne d'énergie. | Analyser un système d’un point de vue structurel et comportemental |
| DC1 Systèmes et informations | **Transmetteurs de puissance** · Caractéristiques ; · Domaines d'application. | Analyser une solution de transmission de puissance |
| DC4 Systèmes linéaires continus invariants | **Structure des systèmes asservis** · Définition et structure d'un système asservi : chaîne directe (ou chaîne d'action), chaîne de retour (ou chaîne d'acquisition), comparateur et écart ; · Consigne, perturbation ; · Régulation, poursuite ; · Définition des performances : rapidité, précision et stabilité. | Justifier la nécessité d’un asservissement (analyse du couple performances/perturbations) |
| DC1 Systèmes et informations | **Association de pré actionneurs et d’actionneurs** · Caractéristiques ; · Domaines d'application. | Analyser une association de pré actionneurs et d'actionneurs |
| DC1 Systèmes et informations | **Commandes programmables** · Fonctions ; · Composants programmables. | Identifier les caractéristiques de la commande, E/S analogiques, numériques |
| DC1 Systèmes et informations | **Description fonctionnelle des systèmes de traitement de l'information** Architecture générale de la chaîne d'information. | Identifier et décrire les composants associés au traitement de l’information |
| DC1 Systèmes et informations | **Information** · Définition et nature, information et support d'information; · Information discrète (TOR et numérique), codage ; · Information analogique. | Identifier la nature et le support d’information |
| DC1 Systèmes et informations | **Capteurs** · Fonctions ; · Nature des grandeurs physiques d'entrées et de sorties; · Nature du signal, support de l’information. | Caractériser un capteur (grandeur physique observée et utilisable, transducteur) |

**MODELISER (S1,S3,S4)**

|  |  |  |
| --- | --- | --- |
| DC4 Systèmes linéaires continus invariants | **Systèmes linéaires continus et invariants** · Modélisation par équations différentielles ; · Représentation par fonction de transfert (formalisme de Laplace) ; · Modèles canoniques 1er et 2ème ordre. | Identifier le comportement d’un système pour l’assimiler à un modèle canonique, à partir d’une réponse temporelle ou fréquentielle Établir un modèle de comportement à partir de relevés expérimentaux |
| DC4 Systèmes linéaires continus invariants | **Systèmes linéaires continus invariants asservis** · Représentation par schémabloc ; · Fonction de transfert en boucle ouverte et en boucle fermée ; · Classe d’un système. | Écrire un schéma bloc du système Déterminer les fonctions de transfert du système en boucle ouverte et en boucle fermée |
| DC4 Systèmes linéaires continus invariants | **Systèmes non linéaires** · Modèle de non linéarité (hystérésis, saturation, seuil, retard) ; · Linéarisation du comportement des systèmes non linéaires continus. | Identifier les non linéarités Identifier le point de fonctionnement pour la linéarisation du modèle du système non linéaire |
| DC4 Systèmes linéaires continus invariants | **Modélisation des systèmes asservis** · Stabilité : - définition, nature de l’instabilité (apériodique, oscillatoire), - contraintes technologiques engendrées, - interprétation dans le plan des pôles, - critère du revers, - marges de stabilité, - dépassement. | Caractériser la stabilité (marges de stabilité) |
| DC4 Systèmes linéaires continus invariants | **Modélisation des systèmes asservis** · Performances et réglages ; · Précision d’un système asservi en régime permanent pour une entrée en échelon, une entrée en rampe, une entrée en accélération ; · Rapidité d’un système asservi : - temps de réponse, - bande passante. | Justifier une simplification du modèle Déterminer l’influence du gain et de la classe de la fonction de transfert en boucle ouverte sur la précision et la rapidité |
| DC4 Systèmes linéaires continus invariants | **Modélisation des systèmes asservis** · Amélioration des performances d’un système asservi ; - critères graphiques de stabilité dans les plans de Black, Bode, marges de stabilité ; - influence et réglage d’une correction proportionnelle, intégrale, dérivée ; - prise en compte d’une perturbation constante, créneau ou sinusoïdale. | Mener une démarche de réglage d’un correcteur pour obtenir les performances attendues |
| DC4 Systèmes linéaires continus invariants | **Systèmes asservis** · Point de fonctionnement ; · Non-linéarités (hystérésis, saturation, seuil…). | Vérifier la cohérence du modèle choisi avec les résultats d’expérimentation ou/et de simulation |

**REALISER (S4)**

|  |  |  |  |
| --- | --- | --- | --- |
| DC8 ANALYSER CONCEVOIR et REALISER | **Réalisation d’un prototype** | Réaliser tout ou partie d’un prototype Valider l’architecture fonctionnelle et structurelle Valider les choix des composants vis-à-vis des performances attendues | *L’acquisition de savoir-faire professionnels est exclue. On se limitera à : - réaliser une pièce prototype ; - implanter (alimenter, paramétrer) une carte de commande dans son environnement matériel ; - programmer à partir d’outils graphiques. Les langages de programmation ne donnent pas lieu à évaluation.* |

**Vérin électrique asservi**

Partie commune aux 3 élèves : Questions 1 à 4 et 14

Travail de l’expérimentateur : Questions 5 à 7

Travail du modélisateur : Questions 8 à 13

# Description du système et de sa commande

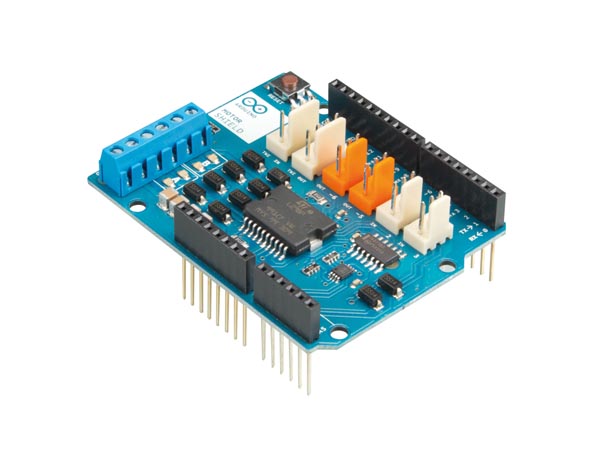
## Description de l’actionneur :

Il est animé par un petit moteur à courant continu, avec vis trapézoïdale et butées de fin de course. La vitesse peut être réduite par une diminution de la tension appliquée. La modification du sens se fait par permutation de la polarité. Un potentiomètre renvoie une tension proportionnelle à la longueur de tige sortie.

Caractéristiques techniques :

|  |  |
| --- | --- |
| Fabricant N° | 16012114CR |
| Référence | DLA-12-10-A-200-POT-IP65 |
| Démultiplication | 10:1 |
| Indice de protection | IP65 |
| Type | Vérin électrique |
| Spécificité | Avec potentiomètre intégré |
| Vitesse | 27 mm/s |
| Effort maximum | 250 N |
| Longueur de course | 200 mm |
| Tension nominale | 12 V/DC |
| Matériau | alliage en zinc (boîtier) et aluminium (tube extérieur et tige). |

## Description de l’interface de puissance :

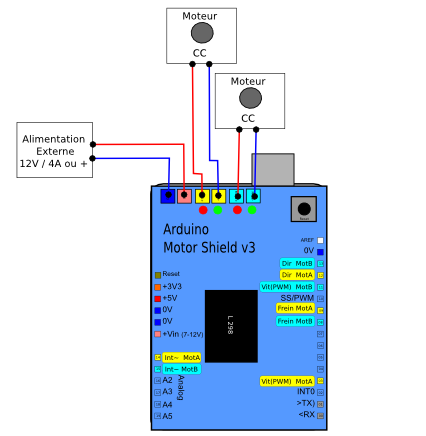


L’Arduino® motor shield rev3 est basé sur le L298, un double pont en H pour piloter des composants inductifs comme des moteurs à courant continu.   
  
Ce shield a deux canaux séparés: A et B. Ils utilisent chacun 4 broches pour:

* le sens de rotation
* la vitesse de rotation
* le freinage rapide
* la mesure de la valeur absolue du courant dans le moteur

Vous pouvez utiliser chaque canal séparément pour alimenter deux moteurs à courant continu. Le shield peut fournir 2 ampères par canal, pour un total de 4 ampères au maximum.

Ce shield est directement enfiché sur une carte Arduino Uno, ce qui évite les câblages.



A

B

pin 13 Dir MotB

pin 12 Dir MotA

pin PWM 11 Vit MotB

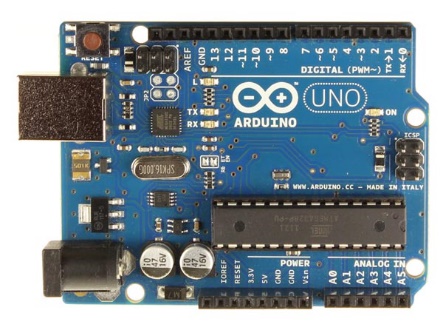
pin 9 Frein MotA

pin 8 Frein MotB

pin analogique A0 Courant MotA

pin analogique A1 Courant MotB

pin PWM 3 Vit MotA



## Description de la partie commande :

Il s’agit d’une Arduino UNO.

Lancez le logiciel Arduino version 1.7.7 ou supérieure et connectez l’Arduino au port USB de l’ordinateur.

Cliquez :

Outils ; Type de carte ; Arduino/Genuino Uno

Outils ; Port ; et sélectionnez le port série sur lequel est connectée l’Arduino.

## Description des câblages :

La carte Motor Shield est alimentée par une alimentation externe 12V

Le moteur du vérin électrique est branché sur les bornes du moteur B

Le potentiomètre donnant la position de la tige du vérin est câblé de la façon suivante :

* fil jaune alimenté en 5V
* fil blanc est relié à GND (0V)
* fil bleu relié à A2

# Partie commune aux 3 élèves : Mise en œuvre du vérin

Objectifs de cette partie : Faire fonctionner l’actionneur dans les deux sens

Manipulation :

* **Démontez l’actionneur de la maquette**
* Copiez le programme Pilote\_Auto\_Vide dans votre espace de travail.
* Ouvrez ce fichier avec le logiciel Arduino version 1.7.7 ou supérieure.
* Enregistrez régulièrement votre travail en changeant le nom du fichier

Dans la partie 1 du programme sont définies toutes les variables.

Pour commencer, nous n’utiliserons que les variables vérin :

// Variables vérin

int Vit; //Vitesse moteur (Pin 11 0-255)

int Dir; //Sens rotation moteur (Pin 13 0-1)

int Pot\_V; //Potentiomètre Vérin (Pin A2 0-1023)

int I; //Courant moteur (Pin A1 0-1023)

float IA; //Courant moteur (Ampères)

int IMax=1; //Courant Max moteur (Ampères) Ne pas dépasser cette valeur

int arretMot=0; //Demande d'arrêt moteur (0-1)

long temps=0; //Temps depuis début du programme (milli sec)

int signifie que la variable est un entier (integer)

long signifie que la variable est un entier de grande longueur (elle occupe plus de mémoire)

float signifie que la variable est un flottant (nombre avec virgule)

Dans la partie 2 du programme sont définies la vitesse de la sortie série (115200 baud ou bits/s) et le mode de fonctionnement des pins. Pour commencer, nous n’utiliserons que les pin 13 (sortie Dir) et 11 (sortie Vit)

Serial.begin(115200); // Initialisation du port série à la vitesse de 115200 baud

pinMode(13, OUTPUT); // initialise pin 13 comme sortie (Direction moteur)

pinMode(11, OUTPUT); // initialise pin 11 comme sortie (Vitesse moteur)

Dans la partie 3 se trouve le programme que nous allons taper. Il est divisée en rubriques.

La rubrique « Commande du moteur » est déjà programmée. Elle envoie la direction et la vitesse au moteur.

// Commande du moteur

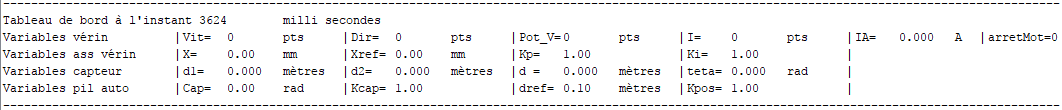
if (arretMot==1 or millis()>60000) {Vit=0;} // Arrêt moteur si surcharge ou aprés 60s

digitalWrite(13,Dir); // Envoie du sens rotation moteur

analogWrite(11, min(abs(Vit),255)); // Envoie de la vitesse avec saturation à 255

A la fin du programme se trouve un tableau de bord qui affiche l’état de toutes les variables en temps réel.

Il est accessible en cliquant  après avoir télé versé le programme avec le bouton .



**Important : Régler la vitesse de la liaison série à 115200 bits/s et arrêter le défilement automatique dès que l’expérience est terminée. Remonter la barre de défilement (à droite) pour revenir à l’instant 0**

Tapez les lignes suivantes dans la partie 3 du programme à la rubrique : // Asservissement en position du vérin

Dir=1;

Vit=200;

Allumez l’alimentation.

Télé versez le programme avec le bouton . Affichez le tableau de bord en cliquant 

* La commande  
  if (arretMot==1 or millis()>60000) {Vit=0;} //Arrêt moteur si surcharge ou après 60s  
  arrête le moteur automatiquement après 60s ou si la variable arretMot est à 1
* La commande digitalWrite(13,Dir); envoie la valeur 0 ou 1 sur le pin 13 de l’Arduino, et la carte motor shield rev3 convertit cette valeur en une tension positive ou négative aux bornes du moteur (1=sortie de tige et 0=rentrée de tige).
* La commande analogWrite(11, min(abs(Vit),255)); envoie la valeur absolue de la variable Vit sur le pin PWM 11 de l’Arduino. Un valeur comprise entre 0 et 255 est convertit par la carte motor shield rev3 en une tension comprise entre 0 et 12V en valeur absolue.

Si Vit dépasse 255, on envoie 255. Cette non linéarité s’appelle ***la saturation***.   
Si Vit est trop faible, le moteur ne démarrera pas. Cette non linéarité s’appelle ***le seuil***.

Donnez des valeur à Vit. Pensez à changer la valeur de Dir à chaque expérience.

Télé versez le programme avec le bouton .

***Question  :***

Déterminez la tension de saturation du moteur.

Déterminez le seuil du moteur (valeur seuil de Vit et tension seuil du moteur).

La sortie du potentiomètre est branchée sur le pin A2 de la carte Arduino. Le convertisseur analogique numérique transforme une tension comprise entre 0 et 5V en un entier compris entre 0 et 1023.

Tapez la ligne suivante dans la partie 3 du programme à la rubrique : // Asservissement en position du vérin

Pot\_V=analogRead(2);

Télé versez le programme avec le bouton . Affichez le tableau de bord en cliquant 

***Question  :*** En faisant fonctionner le vérin à Vit=255, relevez les valeurs Pot\_V\_mini et Pot\_V\_maxi. Relevez la course du vérin en faisant le différence (tige sortie-tige rentrée)

On définit la variable X (position de la tige). Xmini=0 et Xmaxi=course

Tapez la ligne suivante dans la partie 3 du programme à la rubrique : // Asservissement en position du vérin

Utilisez vos propres valeurs numériques.

X=map(Pot\_V, Pot\_V\_mini,Pot\_V\_maxi,Xmini,Xmaxi);

Télé versez le programme avec le bouton . Affichez le tableau de bord en cliquant 

Attention, car la fonction map renvoie un entier. Pour plus de précision, tapez :

X=map(Pot\_V, Pot\_V\_mini,Pot\_V\_maxi,Xmini,10\*Xmaxi)/10.0;

Tapez bien 10.0 et pas 10, pour que la division donne un flottant. Sinon, ce sera un entier, avec une précision moins bonne.

Le potentiomètre du vérin est très précis, mais la numérisation engendre une erreur de **quantification** qui est le déplacement correspondant à la mesure brute d’un point.

***Question  :*** Quelle est l’erreur de quantification sur la position du vérin ?

En dessous du calcul de X, calculez Dir automatiquement avec la ligne suivante :

Dir=((Xref-X)>0); Explication : Dir=1 si Xref>X (test vrai) Dir=0 si Xref<X (test faux)

Testez le fonctionnement en entrant une valeur pour Xref : Xref=100 ;

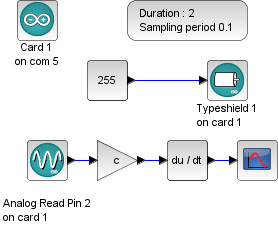
Le vérin atteint la position Xref mais il est instable. Heureusement, un arrêt automatique est prévu après 60s.

***Identification de la fonction de transfert du moteur***

Sauvegardez le programme précédent puis télé versez le programme tollbox\_arduino\_V3 disponible sur le réseau dans le dossier :

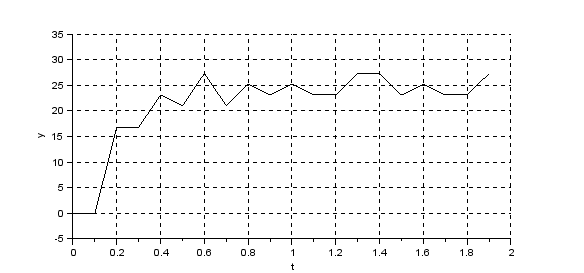
CL\_2PT\Ressources\TP Arduino

Lancez Scilab Xcos et construisez le modèle ci-dessous à partir des palettes « Arduino », « CPGE » et « Systèmes à temps continu » :



V (mm/s)

temps (s)



On envoie 255 à l’interface de puissance du moteur.

On lit le pin 2 (potentiomètre). « c » est le gain du potentiomètre qui transforme les points en mm

On dérive et on affiche le résultat en mm/s.

Cliquez sur Typeshield 1 et précisez bien qu’il s’agit du motor shield rev3 Moteur B.

***Question  :*** Identifiez la fonction de transfert du moteur. Donnez la valeur de son gain K avec unités et de sa constante de temps T. Donnez ces valeurs au modélisateur.

# Travail de l’expérimentateur

Vous pouvez ensuite fermer Scilab/Xcos et reprendre votre programme Arduino.

Pour stabiliser le vérin, il faut créer une boucle d’asservissement pour moduler la vitesse On choisit un correcteur proportionnel de gain Kp=10. Entrez cette valeur de Kp dans la partie 1 du programme.

En dessous du calcul de X, calculez Vit comme ci-dessous dans la partie 3 du programme à la rubrique :

// Asservissement en position du vérin

Vit=Kp\*(Xref-X);

Entrez une autre valeur de Xref pour tester le fonctionnement.

***Question  :***

* Pour quelles valeurs de  se produisent la saturation et le seuil ?
* Faites des expériences pour déterminer l’écart statique de cet asservissement en position.
* D’où provient cet écart statique et comment peut-on la réduire ?

***Filtrage du courant :***

La mesure du courant peut se faire sur le pin A1 de l’Arduino et s’affiche sur le moniteur série.

Tapez la ligne suivante dans la partie 3 du programme à la rubrique : // Calcul du courant

I=analogRead(1);

Donnez une consigne et lisez la valeur brute du courant en points sur le tableau de bord. Vous remarquez qu’avant l’arrêt du vérin, le courant n’est pas stable du tout. Nous allons donc le filtrer avec un filtre passe-bas. Modifiez le calcul du courant avec la ligne suivante :

I=0.95\*I+0.05\*analogRead(1);

0.95\*I signifie que l’on enlève 5% sur la valeur du courant précédent

+0.05\*analogRead(1) signifie que l’on ajoute 5% de valeur fraiche au courant.

Ainsi, la valeur du courant se rafraîchit progressivement et les variations brusques du courant (hautes fréquences) n’interviennent que pour 5% de la valeur totale. Modifier en prenant les valeurs 0.9 et 0.1. Le rafraîchissement sera plus rapide mais le filtrage moins efficace. La somme doit toujours faire 1.

Nous allons maintenant calculer la valeur IA du courant en Ampères sachant que le gain du capteur de courant est de 1,65 V/A, et sachant que le convertisseur analogique-numérique de l’Arduino transforme 5V en 1023 points.

***Question  :*** Tapez une ligne de code pour calculer le courant IA en Ampères.

Testez le fonctionnement. Le courant doit être inférieur à 1A sauf lors des accélérations et décélérations.

***Question  :*** L’alimentation supporte une charge nominale de 3A, le moteur 2,3A et la carte Motor Shield v3 2A. Pour des raisons de sécurité, on décide de limiter le courant IMax à 1A (voir Imax dans la partie 1 du programme)

Dans la partie 3 du programme à la rubrique : // Programmation de l’arrêt moteur en cas de surcharge, écrivez la ligne suivante et faites des tests en bloquant manuellement la tige du vérin lors du déplacement :

if (IA>IMax) {arretMot=1;}

Vous pouvez reprendre le mouvement en appuyant sur la touche « Reset » de la carte Motor Shield v3.

Cette commande risque de nous poser des problèmes, car à chaque démarrage, le courant est fort et le moteur risque de s’arrêter aussitôt.

La fonction millis() donne le temps absolu en ms depuis le lancement du programme.

En écrivant temps=millis(); vous relevez l’instant présent dans la variable temps.

Modifiez la ligne de code ci-dessus pour couper la vitesse du moteur (arretMoteur=1) si le courant dépasse IMax=1A pendant plus de 500 ms consécutives. On éliminera ainsi les surcharges dues au démarrage.

Vérifiez le fonctionnement en bloquant manuellement la tige du vérin. Il doit s’arrêter en cas de surcharge.

Ce type de commande est souvent utilisé pour arrêter les moteurs électriques, par exemple pour les vitres électriques des voitures, qui s’arrêtent automatiquement en cas de blocage.

Pour la suite, pour éviter les arrêts du vérin, dans la partie 1 du programme, portez la valeur de IMax à 2A.

# Travail du modélisateur : Optimisation de la boucle d’asservissement

***Cahier des charges :***

* Ecart statique nul, même en présence de perturbations (tolérance ±0.2 mm)
* Marge de phase Mϕ≥50°
* Temps de réponse à 5 % inférieur à 3s pour un échelon de 10 mm
* Premier dépassement inférieur à 4 mm quelle que soit l’amplitude de la consigne échelon

***Objectif*** : Détermination des paramètres du correcteur PI par la méthode de l’optimum symétrique et simulation avec Scilab Xcos

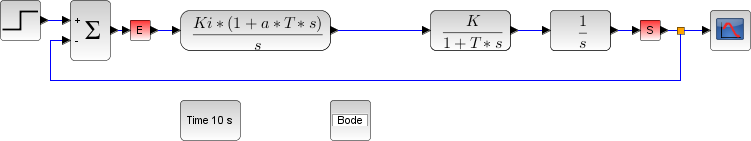
Construire sous Scilab Xcos et la palette CPGE le schéma bloc de la boucle d’asservissement de position avec un correcteur proportionnel intégral de fonction de transfert :

 *s*  étant la variable de Laplace.

On note  la constante de temps du correcteur.

L’entrée est un échelon de 10 mm

E et S se trouvent dans la palette CPGE/Analyses (grandeurs physiques)



Cliquez : Simulation ; Modifier le contexte et entrez les valeurs numériques des différents paramètres.

Prendre les valeurs Ki=10 et a=1 pour commencer l’étude.

Ki=10;

a=1;

T=\_ma\_constante\_de\_temps;

K=\_mon\_gain;

***Question  :*** D’après les expériences précédentes et d’après le cahier des charges, justifiez la présence d’un correcteur comportant une intégration.

***Question  :*** Montrez qu’il n’est pas possible d’utiliser un correcteur intégral pur de fonction de transfert .

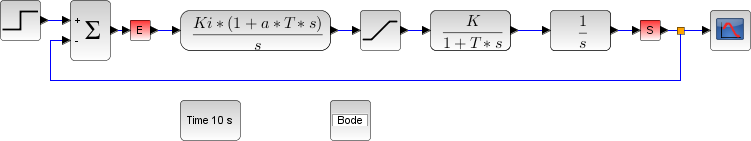
Pour appuyer votre réponse, faites une simulation avec a=0.

***Question  :*** Montrez que « a » doit être plus grand que 1. Pour appuyer votre réponse, faites une simulation avec a=0,5.

***Question  :*** En faisant des simulations avec Scilab, déterminez la valeur de « a » qui donne une remontée de phase jusque -130°.

***Question  :*** En faisant des simulations avec Scilab, déterminez la valeur de «*Ki* » qui donne une marge de phase Mϕ conforme au cahier des charges. Calculez la valeur de « *Kp »* et donnez les valeurs de « *Kp »* et « *Ki »* à l’expérimentateur.

***Question  :*** Le système réel comporte une non linéarité puisque l’Arduino motor shield rev 3 reçoit une consigne comprise entre -255 et +255. Modifiez le modèle Scilab en ajoutant cette non linéarité et vérifiez la conformité du modèle au cahier des charges pour un échelon de 10 mm puis pour un échelon de 100 mm.



# Travail commun et vérification expérimentale

Le correcteur proportionnel est déjà en place. Il reste à programmer le correcteur intégral.

Dans la partie 1 : // Variables asservissement en position du vérin, entrez vos valeurs de « *Kp »* et « *Ki ».*

Dans la partie 3 // Asservissement en position du vérin, ajoutez les lignes de code suivantes pour calculer   (intégration par la méthode des rectangles).

tInt=micros(); // temps absolu depuis le démarrage de la carte (en µs)

if (abs(Vit)<=255) {integral=integral+(Xref-X)\*((tInt-tIntPrec)/1000000);}

tIntPrec=tInt;

Dans notre cas, la période d’échantillonage Te=tInt-tIntPrec est recalculée à chaque boucle.

La bibliothèque SimpleTimer (pas utilisée ici) permet de définir une période d’échantillonnage constante.

***Astuce :*** **Il ne faut pas intégrer pendant la durée de la saturation**, c'est-à-dire quand Vit>255, car le dépassement deviendrait très très important pour les grands échelons. Vous pouvez le vérifier dans le modèle Scilab avec saturation en donnant un échelon de 100 mm ou dans le programme Arduino en enlevant le test : if (abs(Vit)<=255)

Enfin, modifiez la ligne de code calculant Vit : Vit=Kp\*(Xref-X)+Ki\*integral;

***Question  :*** Vérifiez le cahier des charges et déterminez les écarts entre les performances attendues, les performances simulées et les performances mesurées.

|  |  |  |
| --- | --- | --- |
| Performances attendues | Performances mesurées | Performances simulées |
| Ecart statique nul, même en présence de perturbations (tolérance ±0.2 mm) |  |  |
| Marge de phase Mϕ≥50° |  |  |
| Temps de réponse à 5 % inférieur à 3s pour un échelon  de 10 mm |  |  |
| Premier dépassement inférieur  à 4 mm quelle que soit  l’amplitude de la  consigne échelon |  |  |

**Asservissement en cap d’un pilote automatique de pénichette**

Travail de l’expérimentateur : Questions 15

Travail du modélisateur : Questions 16

Partie commune aux 3 élèves : Question 17-18

***Objectif :*** Créer un prototype fonctionnel de pilote automatique pour petite péniche asservi en cap.

Deux capteurs à ultrason situés sur les côtés de la péniche, au raz de l’eau mesurent l’angle θ entre la péniche et la berge du canal. Un asservissement permettra de maintenir cet angle égal à la consigne définie par le timonier.

Cette consigne est le Cap. Logiquement, il faudrait Cap=0 pour rester parallèle aux berges du canal.

***Hypothèses :***

Les angles seront considérés petits.

La péniche se déplace à la vitesse V=5 km/h=1.39 m/s

La fonction de transfert de la péniche est :  avec *s* la variable de Laplacec

Justification : si l’on incline la barre de α=0.1 rad (5.7°) à la vitesse V=5 km/h, la pénichette tourne à la vitesse  (soit un tour en 1 minute), avec un temps de réponse de 3×0.5s.

Berge du canal

Barre

Capteur à ultrasons 2

Capteur à ultrasons 1

d2

d1

e

b

Pilote

automatique

Vitesse V

X

α

θ

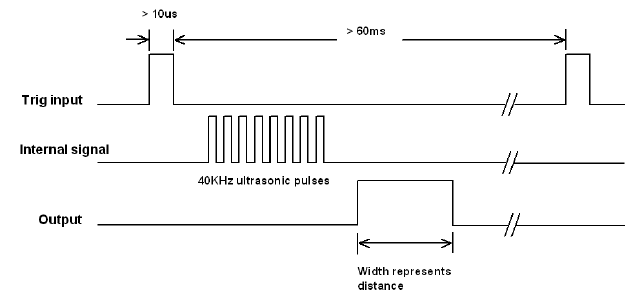
# Travail de l’expérimentateur

## [Résultat de recherche d'images pour "capteur à ultrasons"](https://www.google.fr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwijwqDcyKnVAhVEvBQKHbzmBfYQjRwIBw&url=http://www.framboise314.fr/mesure-de-distance-par-ultrasons-avec-le-raspberry-pi/&psig=AFQjCNEAkYq10JjE-fBdplJDUFcJ1jrDlA&ust=1501248527004636)Mise en œuvre des capteurs à ultrasons :

A part son alimentation 5V (broche VCC) et la masse (broche GND), le capteur à ultrasons fonctionne à partir des deux pins Trig et Echo, qui doivent être connectés à deux pins d’un microcontrôleur Arduino.

|  |  |
| --- | --- |
| Capteur à ultrasons 1 :  Trig1 relié au pin 4  Echo1 relié au pin 5 | Capteur à ultrasons 2 :  Trig2 relié au pin 6  Echo2 relié au pin 7 |

* La demande de la mesure s’effectue en maintenant la broche Trig à l’état 1 pendant 10 μs.
* Le capteur émet alors de courtes impulsions sonores à haute fréquence. Ces impulsions se propagent dans l’air à la vitesse du son (345 m/s). Lorsqu’elles rencontrent un objet, elles se réfléchissent et reviennent sous forme d’écho au capteur. Celui-ci calcule alors la distance le séparant de la cible sur la base du temps écoulé entre l’émission du signal et la réception de l’écho.
* Le capteur donne le résultat de sa mesure en maintenant la broche Echo à l’état 1 pendant le temps qu’à mis le son pour aller jusqu’à la cible et revenir. Le cycle dure un peu plus de 60ms, soit une fréquence de rafraichissement de 16 Hz environ.

[](https://www.google.fr/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwjbx4eGyanVAhXB1hQKHXyfDP0QjRwIBw&url=https://itechnofrance.wordpress.com/2013/03/12/utilisation-du-module-ultrason-hc-sr04-avec-larduino/&psig=AFQjCNEAkYq10JjE-fBdplJDUFcJ1jrDlA&ust=1501248527004636)

durée

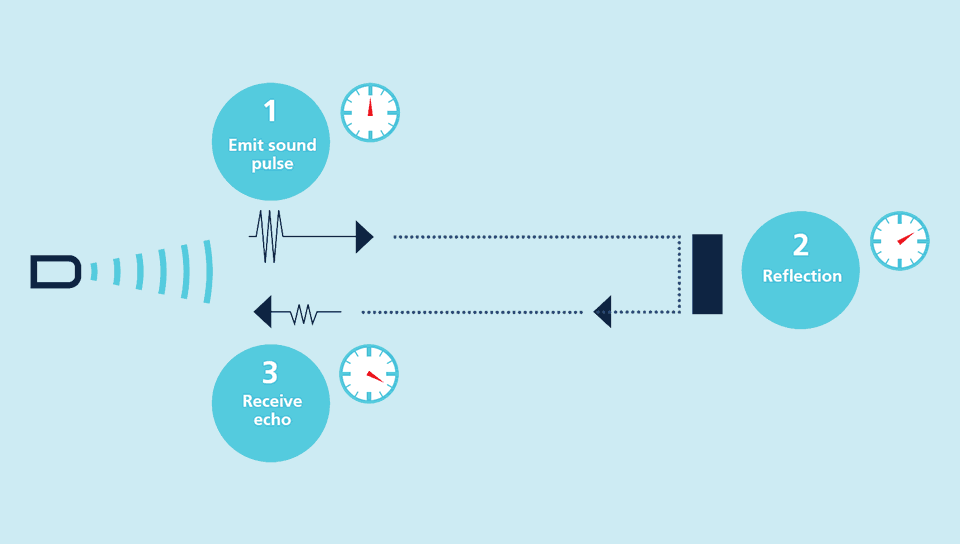
Largeur représentant la distance

Impulsions ultrason à 40 kHz

Signal interne

Echo

Trig



Les capteurs à ultrasons microsonic détectent tout type d’objet et sont disponibles pour des portées de 20 mm à 10 m et, du fait même de leur principe, donnent la valeur mesurée au millimètre près.

Lancez le logiciel Arduino version 1.7.7 ou supérieure et connectez l’Arduino au port USB de l’ordinateur.

Ouvrez le fichier Pilote\_Auto2\_Verin

Ce fichier comporte l’asservissement en position du vérin fait dans le TP précédant.

Cliquez :

Outils ; Type de carte ; Arduino/Genuino Uno

Outils ; Port ; et sélectionnez le port série sur lequel est connectée l’Arduino.

Dans la partie 1 du programme se trouvent les // Variables capteur :

unsigned long duree1=0; // Temps aller-retour capteur 1/rive (micro sec)

unsigned long duree2=0; // Temps aller-retour capteur 2/rive (micro sec)

float d1=0; // Distance capteur 1/rive (métres)

float d2=0; // Distance capteur 2/rive (métres)

float d=0; // Distance péniche/rive (métres)

float teta=0; // Cap péniche/rive (rad)

Dans la partie 2 du programme, on a initialisé les pins des capteurs :

pinMode(4, OUTPUT); // Initialise pin 4 comme sortie (Trig 1)

pinMode(5, INPUT); // Initialise pin 5 comme entrée (Echo 1)

pinMode(6, OUTPUT); // Initialise pin 6 comme sortie (Trig 2)

pinMode(7, INPUT); // Initialise pin 7 comme entrée (Echo 2)

Dans la partie 3 du programme à la rubrique : // Programmation du capteur, écrivez (sans les commentaires) :

digitalWrite(4, HIGH); //Broche Trig1 à l'état haut

delayMicroseconds(10); //on attend 10 µs

digitalWrite(4, LOW); //Broche Trig1 à l'état bas

// On lit la durée d’état haut sur la broche Echo1 en micro secondes

duree1 = pulseIn(5, HIGH);

//d1 est filtrée aves 90% de valeurs anciennes et 10% de valeurs fraiches

//si durée1 > 30000 micro secondes, l'onde est perdue

if(duree1 < 30000) {d1=0.9\*d1+0.1\*duree1\*345.0/2000000.0;}

//on multiplie par la vitesse, d1=duree1\*v/2

Attention, car duree1 est un entier. Tapez bien 345.0 pour obtenir un flottant. Sinon, d1 sera toujours nulle.

Téléversez le programme avec le bouton . Affichez le tableau de bord en cliquant .

Placez l’écran à 20 centimètres du capteur et vérifiez que la mesure de d1 est juste. Eventuellement, vous pouvez ajuster la célérité du son pour obtenir un résultat plus précis (355 à la place de 345 m/s)

***Question  :*** Elaborez un programme qui renvoie l’angle teta=(d2-d1)/b (avec l’hypothèse d’angle petit) et la distance moyenne d=(d1+d2)/2. Mesurez b sur la maquette. Testez le fonctionnement.

# Travail du modélisateur

***Objectif :*** Déterminer le gain du correcteur de la boucle d’asservissement en cap.

Ouvrez le fichier « Modèle Vérin élec.zcos » qui contient la correction du TP Vérin électrique Asservi.

A partir du modèle Scilab Xcos non linéaire du vérin électrique asservi en position, réalisez le modèle ci-dessous de l’asservissement en Cap (c'est-à-dire de l’asservissement angulaire).

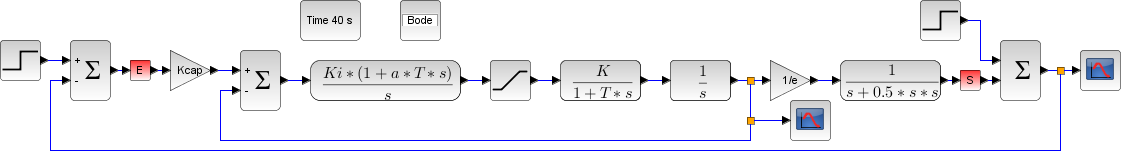
La consigne est un échelon, instant 0, valeur initiale 0, valeur finale 0.1 rad.

La perturbation est un échelon, instant 20s, valeur initiale 0, valeur finale 0.1 rad.

Cliquez : Simulation ; Modifier le contexte et entrez les valeurs de Kcap et e :

Kcap=1000 ;

e= 360mm ; (longueur de la barre, voir page 11)



***Cahier des charges :***

* Ecart statique nul, même en présence de perturbations (tolérance ±0.01 rad)
* Marge de phase : Mϕ≥50°
* Temps de réponse à 5%  inférieur à 12s pour une consigne en échelon de 0.1 rad
* Dépassement sur la réponse indicielle < 5%

***Question  :***

* En faisant des simulations successives, déterminez la plus grande valeur de Kcap qui donne des performances conformes au cahier des charges.
* Comment expliquez-vous que certaines valeurs de Kcap donnent un comportement instable sur la réponse indicielle et stable sur le diagramme de Bode ?

# Travail commun aux 3 élèves

Entrez votre valeur de Kcap dans la partie 1 du programme Arduino à la rubrique :

// Variables asservissement en cap du pilote automatique

Tapez dans la partie 3 du programme à la rubrique // Asservissement en cap du pilote automatique

Cap=0 ; //pour demander un déplacement parallèle à la berge

Xref=100-Kcap\*(Cap-teta) ; //pour que le vérin soit en position centrale (X=100) quand Cap=teta

Téléversez le programme avec le bouton . Affichez le tableau de bord en cliquant .

Testez le fonctionnement. Quand la barre tourne, vous devez tourner le réflecteur pour simuler le mouvement de la pénichette.

***Question  :*** Vérifiez le cahier des charges et déterminez les écarts entre les performances attendues, les performances simulées et les performances mesurées (remplir le tableau ci-dessous).

|  |  |  |
| --- | --- | --- |
| Performances attendues | Performances mesurées | Performances simulées |
| Ecart statique nul, même en présence de perturbations (tolérance ±0.01 rad) |  |  |
| Marge de phase Mϕ≥50° |  |  |
| Temps de réponse à 5%  inférieur à 12s pour une consigne en échelon de 0.1 rad |  |  |
| Dépassement sur la réponse indicielle : 0 % |  |  |

**Asservissement en position d’un pilote automatique de pénichette**

Travail de l’expérimentateur : Questions 18

Travail du modélisateur : Questions 19

Partie commune aux 3 élèves : Question 20

***Objectif :*** Créer un prototype fonctionnel de pilote automatique pour petite péniche asservi en position.

Deux capteurs à ultrason situés sur les côtés de la péniche, au raz de l’eau mesurent la distance d et l’angle θ entre la péniche et la berge du canal. Un asservissement permettra de maintenir cette distance d égale à la consigne dref=0.2 mètres.

***Hypothèses :***

Les angles seront considérés petits.

La péniche se déplace à la vitesse V=5 km/h=1.39 m/s

La fonction de transfert de la péniche est :  avec *s* la variable de Laplacec

Justification : si l’on incline la barre de α=0.1 rad (5.7°) à la vitesse V=5 km/h, la pénichette tourne à la vitesse  (soit un tour en 1 minute), avec un temps de réponse de 3×0.5s.

Berge du canal

Barre

Capteur à ultrasons 2

Capteur à ultrasons 1

d2

d1

e

b

Pilote

automatique

Vitesse V

X

α

θ

# Travail de l’expérimentateur

Le timonier donne comme consigne la distance par rapport à la berge : dref=0.2 mètres

Dans la partie 1 du programme, on trouve les différentes variables :

// Variables asservissement en position du pilote automatique

float dref=0.2; //Consigne de position (mètres)

float Kpos=1; //Gain correcteur de position

Dans la partie 3 du programme, rubrique // Asservissement en cap du pilote automatique

Effacer la ligne Cap=0 ; car la cap va être calculé automatiquement

***Question  :***

Tapez dans la partie 3 du programme, rubrique // Asservissement en position du pilote automat

Cap=-Kpos\*( ?????????);// Le signe – peut être nécessaire si la barre se déplace dans le mauvais sens

Testez le fonctionnement en déplaçant l’écran réflecteur. Vérifiez que le bateau se stabilise à dref=0.2 mètres de cet écran.

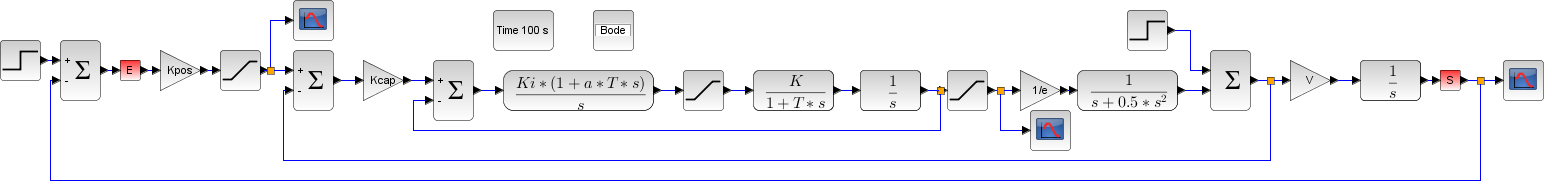
# Travail du modélisateur

***Cahier des charges :***

* Ecart statique nul, même en présence de perturbations
* Marge de phase : Mϕ≥50°
* Temps de réponse à 5%  inférieur à 30s pour une consigne en échelon de 0.2 m
* Dépassement sur la réponse indicielle : <20 % pour un échelon de 0.2 m

***Question  :*** A partir du modèle Scilab Xcos de l’asservissement en cap, réalisez le modèle de l’asservissement de position. Déterminez les valeurs des saturations en cap (voir ci-dessous).

En faisant des simulations successives, déterminez la plus grande valeur du correcteur proportionnel Kpos pour que le système soit conforme au cahier des charges.



# Travail commun aux 3 élèves

***Question  :*** Pour mieux visualiser le fonctionnement, conservez la valeur Kpos=1 sur la maquette. Vérifiez le cahier des charges et déterminez les écarts entre les performances attendues et les performances simulées (remplir le tableau ci-dessous).

|  |  |  |
| --- | --- | --- |
| Performances attendues | Performances mesurées | Performances simulées |
| Ecart statique nul, même en présence de perturbations |  |  |
| Marge de phase Mϕ≥50° |  |  |
| Temps de réponse à 5%    inférieur à 30s pour une  consigne en  échelon de 0.2 m |  |  |
| Dépassement sur la réponse indicielle : <20 %  pour un échelon de 0.2 m |  |  |