



MODULE M4 - PAGE WEB DYNAMIQUE

CYCLE INGÉNIEUR - UV 2.7

Le robot DART

Remplacement du robot JOG en service depuis 2010 ...

... par le robot DART

Robot prototype réalisé par les étudiants de 2A promo 2016 dans le cadre des UV projets 3.4 et 4.4 :

- D : Brian **D**etourbet
- A : Fahad **A**l Shaik
- A : Elouan **A**utret
- R : Corentin **R**ifflart
- R : Clément **R**odde
- T : Rémi **T**errien

DAARRT simplifié en DART

DART vs. JOG



Le robot DART

Le contrôle du robot DART se fait par l'acquisition des informations issues des capteurs et la commande des actionneurs.

Architecture du robot (voir tableau)

Plusieurs méthodes d'accès aux capteurs et actionneurs cohabitent sur le robot DART :

- entrées sorties numériques (pseudo fichiers linux)
- liaison série (RS 232)
- bus (I2C)

Contrôle du robot DART

Les fonctions de bas niveau d'accès aux pilotes, à la mémoire et aux bus sont fournies

Il faudra écrire les fonctions de haut niveau, par exemple, pour la centrale de navigation connectée au port série RS-232 :

- nous disposons de la liste des commandes à envoyer
- nous disposons du format des réponses (accéléros, gyros, cap mag.)
- nous disposons du module serial de Python pour accéder au port série
- nous devons écrire, par exemple, une fonction lectureCompas() en envoyant la bonne commande sur la liaison série, en récupérant l'information et en la mettant au format qui nous intéresse

En fin de cette partie de TD, vous disposerez d'un ensemble de fonctions Python vous permettant d'acquérir les informations capteurs et de piloter les actionneurs du robot DART

Gestion du temps

importer le module time de Python :
import time

acquérir le temps (donné par l'horloge de la carte)
t0 = time.time() # valeur en secondes (float)
st0 = time.asctime() # chaîne de caractères

dormir pendant un temps de t1 secondes (float)
time.sleep(t1)

Simulateur - Mise au point

Le simulateur est écrit en python à l'aide du module pygame. Le code est entièrement disponible sous MOODLE.

En principe vous utiliserez le même programme sur le simulateur et sur le robot réel. Le programme détecte si il est exécuté sur le robot, sinon il lance le simulateur)

Le simulateur est en 2D, il permet la commande des moteurs et l'acquisition des capteurs (sonar, IMU, odomètres)

voir démo simulateur

Contrôle-Commande

Contrôle-commande du robot DART :

- Définition d'une mission
- Formalisme de la machine à états finis (FSM : Finite State Machine)
- Pilotage du DART : reprise de la FSM du module 1 (pilotage au clavier de NAO)

Exécution de la mission : démarrage à l'état START et fin à l'état END et exécutant tous les Δt ms :

- Acquisition des informations capteurs
- Estimation des erreurs entre le comportement souhaité et le comportement réellement exécuté
- Définition des corrections
- Commande des actionneurs (moteurs)

Méthodes de Régulation

Appliquer une correction en fonction de l'erreur entre :

- le comportement souhaité : **la consigne**
- le comportement réel : **la mesure**

Mode "Bang-Bang" :

- Utilisation du signe de l'erreur pour la correction
- Module de la correction constant

Mode Proportionnel :

- Utilisation du signe de l'erreur pour la correction
- Module de la correction est une proportion de l'erreur
- Coefficient de proportionnalité K_P

Régulateur Proportionnel

Soit **headOk** le cap désiré exprimé dans un repère géographique (0° = Nord)

Mesure du cap à l'aide de la boussole avec une résolution de 0.1°

headActual=compass.getWordHeading()

Calcul de l'erreur de cap :

headErr = deltaCap (headOk,headActual)

Commande des moteurs :

speedLeft = **speedLinear** + **Kp*****HeadErr**

speedRight = **speedLinear** - **Kp*****HeadErr**

Détermination empirique de **Kp**, si **Kp** faible le robot met plus de temps pour atteindre le bon cap, si **Kp** élevé le robot risque de d'osciller autour du bon cap ...

Attention, le résultat du calcul de la différence de cap, fonction deltaCap(), doit être compris entre -180 et $+180$ degrés

Remarque : si **speedLinear** = 0, le robot tourne sur lui-même, si **speedLinear** est non nul le robot régule son cap en avançant ...

Régulateur PID

La correction est la somme de trois termes :

- un terme K_P proportionnel à l'erreur
- un terme K_D proportionnel à la dérivée de l'erreur
(variation de l'erreur)
- un terme K_I proportionnel à l'intégrale de l'erreur
(dérive de l'erreur)

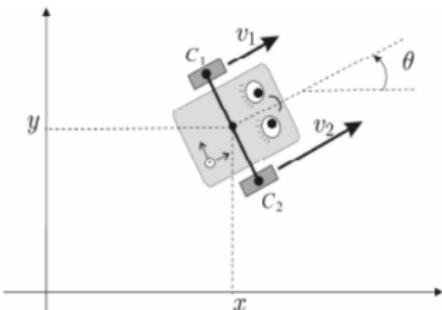
Définition des gains K_P , K_D et K_I

```
previous_error = setpoint - actual_position
integral = 0
start:
    error = setpoint - actual_position
    integral = integral + (error*dt)
    derivative = (error - previous_error)/dt
    output = (Kp*error) + (Ki*integral) + (Kd*derivative)
    previous_error = error
    wait(dt)
    goto start
```

Rappel Modèle Char

JOG : modèle de type char

$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = w \end{cases}$$



r Rayon des roues

l Distance entre roues

Modèle similaire pour DART et JOG (modèle utilisable aussi sur un catamaran !!)

Rappel Modèle Char

$$v = \frac{v_1 + v_2}{2}$$

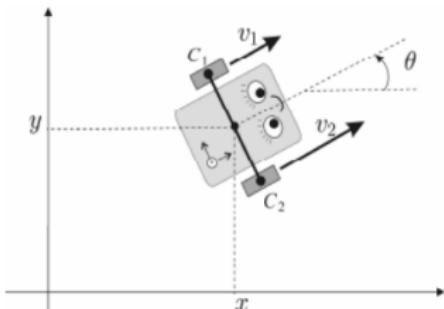
$$\omega = \frac{v_2 - v_1}{l}$$

$$\omega_1 = \alpha_1 u_1$$

$$\omega_2 = \alpha_2 u_2$$

$$v_1 = r\omega_1$$

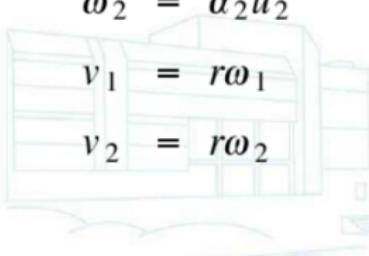
$$v_2 = r\omega_2$$



$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = w \end{cases}$$

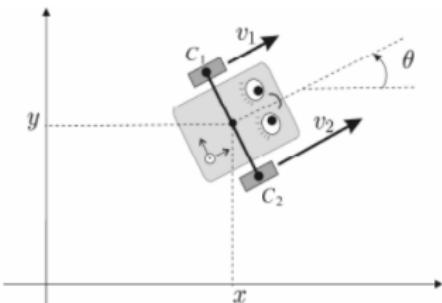
r Rayon des roues

l Distance entre roues



Rappel Modèle Char

$$\begin{cases} \dot{x} = \frac{r\alpha_1 u_1 + r\alpha_2 u_2}{2} \cos \theta \\ \dot{y} = \frac{r\alpha_1 u_1 + r\alpha_2 u_2}{2} \sin \theta \\ \dot{\theta} = \frac{r\alpha_2 u_2 - r\alpha_1 u_1}{l} \end{cases}$$



$$v = \frac{v_1 + v_2}{2}$$

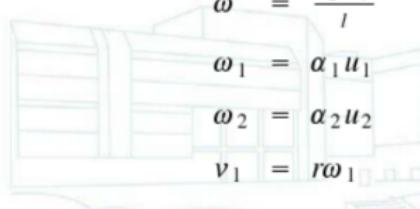
$$\omega = \frac{v_2 - v_1}{l}$$

$$\omega_1 = \alpha_1 u_1$$

$$\omega_2 = \alpha_2 u_2$$

$$v_1 = r\omega_1$$

$$v_2 = r\omega_2$$



$$\begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = w \end{cases}$$

r Rayon des roues

l Distance entre roues

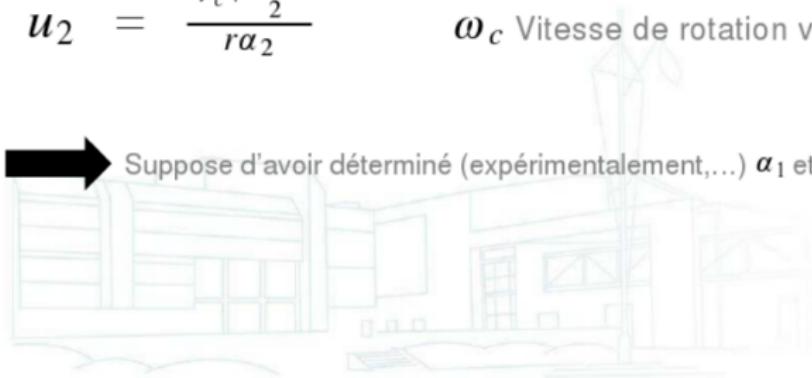
Vitesse Tangentielle

$$u_1 = \frac{v_c - \frac{\omega cl}{2}}{r\alpha_1} \quad v_c \text{ Vitesse tangentielle voulue}$$

$$u_2 = \frac{v_c + \frac{\omega cl}{2}}{r\alpha_2} \quad \omega_c \text{ Vitesse de rotation voulue}$$



Suppose d'avoir déterminé (expérimentalement,...) α_1 et α_2



Trajectoire Circulaire

On a la relation: $v_c = R\omega_c$

v_c Vitesse tangentielle voulue

R Rayon du cercle à suivre

En remplaçant dans la formule de la commande par vitesse tangentielle et de rotation:

$$u_1 = \left(1 - \frac{l}{2R}\right) \frac{v_c}{r\alpha_1}$$

$$u_2 = \left(1 + \frac{l}{2R}\right) \frac{v_c}{r\alpha_2}$$

Trajectoire Linéaire

Vu que $v = \frac{d}{t}$

En imposant d et t cela revient à une commande en vitesse tangentielle (avec vitesse de rotation nulle pour aller tout droit)



Odomètres - Position Angulaire

$$\theta_1 = \beta_1 odo_1$$

$$\theta_2 = \beta_2 odo_2$$

Avec β_1 et β_2 à déterminer (expérimentalement,...)



Odomètres - Vitesse Angulaire

$$\omega_{m1} = \frac{\beta_1 \delta odo_1}{dt}$$

$$\omega_{m2} = \frac{\beta_2 \delta odo_2}{dt}$$

Avec β_1 et β_2 à déterminer (expérimentalement,...)



Odomètres - Régulation

Boucle faisant:

$$u_1 = K_1(\omega_{c1} - \omega_{m1})$$

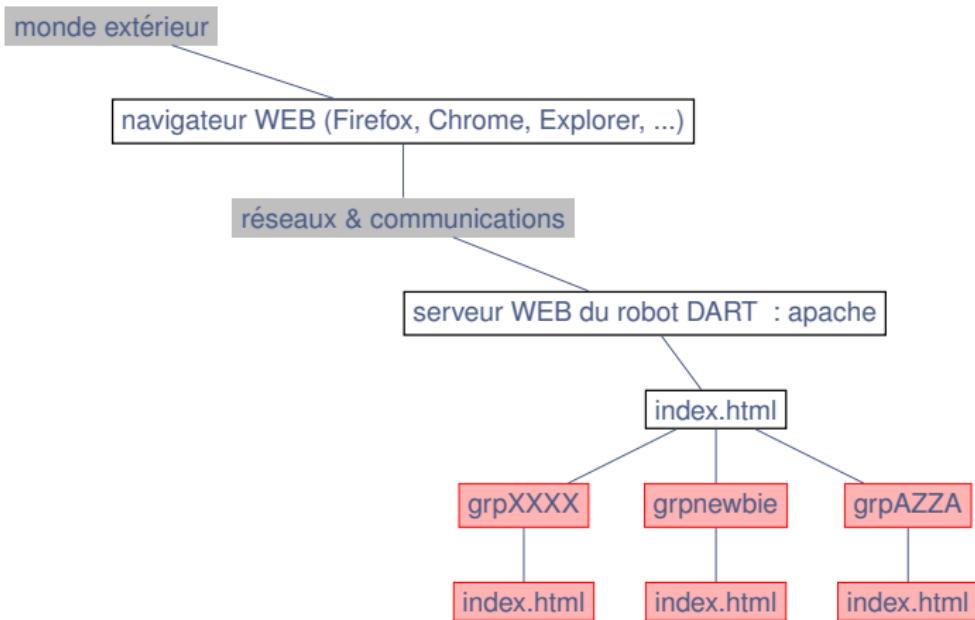
$$u_2 = K_2(\omega_{c2} - \omega_{m2})$$

→ Commande grand gain (mais risque d'être saccadée)

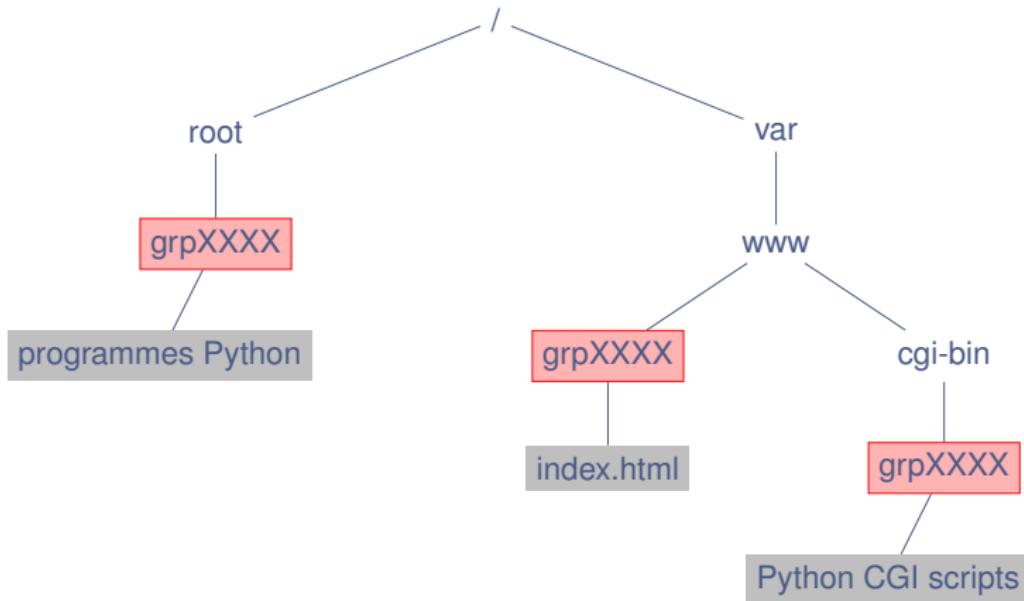
→ Ceci n'est pas vraiment une régulation du char, mais plutôt de ses moteurs

Suppose qu'on a β_1 et β_2 mais pas α_1 et α_2

Accès WEB au robot DART



Robot DART - Répertoires Utiles



Rappel - Répertoires Utiles

Après vous être connectés au robot DARTX par la console, vous êtes en principe dans le répertoire /root (sinon placez vous dans ce répertoire en tapant **cd /root**)

Créez votre répertoire par **mkdir grpXXXX**, n'oubliez pas le préfixe **grp** (**XXXX** étant le pseudo de votre groupe).

Pour les modules M1 et M4 répétez l'opération dans les dossiers /var/www et /var/www/cgi-bin, soit :

- **cd /var/www**
- **mkdir grpXXXX**
- **chmod 755 grpXXXX**
- **cd /var/www/cgi-bin**
- **mkdir grpXXXX**
- **chmod 755 grpXXXX**

Canevas d'une page WEB en HTML

HTML : Hyper Text Markup Language

Le texte est mis en forme à l'aide de balises (ou "tags")

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

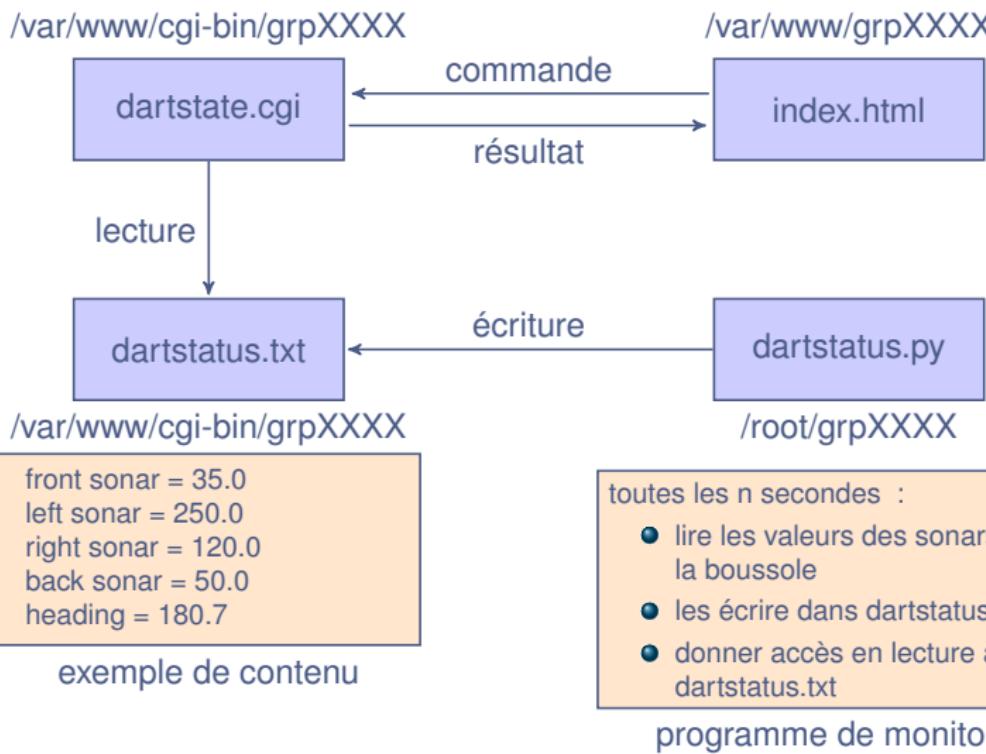
Raffraîchissement Automatique

Raffraîchissement manuel d'une page WEB : touche F5 ou commande "actualiser"

Utilisation de la balise < META > pour simuler toutes les 30 secondes l'appui sur la touche F5

```
<meta http-equiv="Refresh" content="30 ; url=/grpXXX/index.html">
```

Architecture

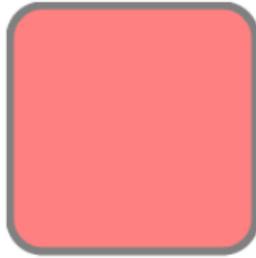


SVG - Scalable Vector Graphics

```
<!DOCTYPE html>
<html>
<body>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
    <rect x="50" y="20" rx="20" ry="20"
          width="150" height="150"
          style="fill :red ;stroke :black ;stroke-width :5 ;opacity :0.5" />
</svg>

</body>
</html>
```



Python - Format d'écriture des textes

Pour la création de documents HTML ou SVG par scripts CGI en python, il est utile de formatter précisément les chaînes de caractères.

Chaîne de caractères définissant le format et une liste de valeurs

```
stout = "valeur i= % 4d, h= % 0x4.4x, v= % 7.2f, st=%s"%(i1,i1,x,str)
```

Si `i1=255`, `x=41.999`, `str="Hi guys"`, `print stout` nous donne :

```
'valeur i = 255, h = 0x00ff, v = 42.00, st = Hi guys'
```

```
st="valeur i = %4d, h = 0x%4.4x, v = %7.2f, st = %s"  
i1=255  
x=41.999  
stw="Hi guys"  
stout = st%(i1,i1,x,stw)  
print stout
```