

Réaliser un dispositif “tête-cou” qui permet de suivre un stimulus visuel.

Arduino C Makefile Servo-Moteurs

Asservissement

(PID)

Architectures de contrôle

Traitement d'image et vision simple

B&W -- couleur

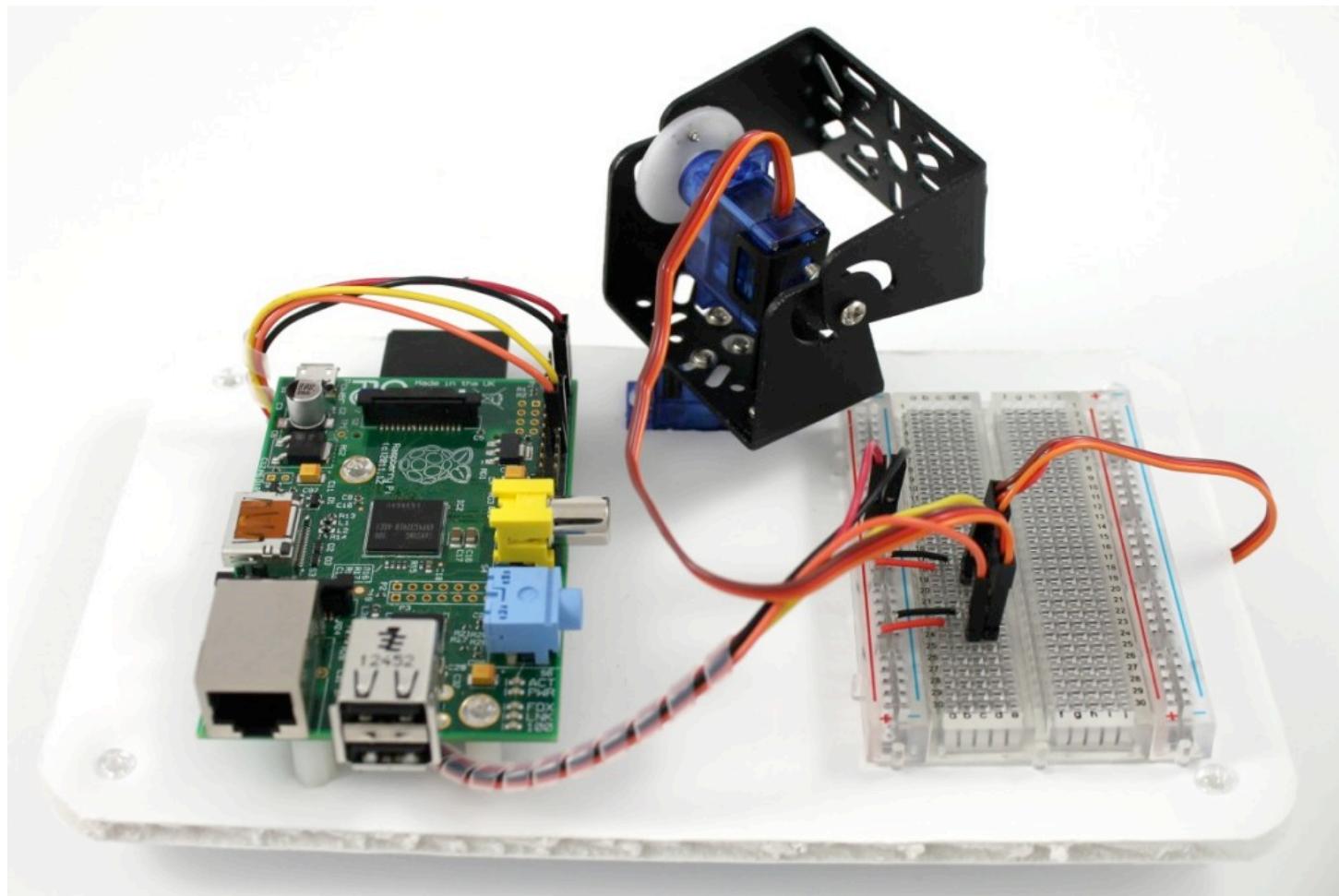
Temps réel

Intelligence Artificielle

dispositif :



dispositif :

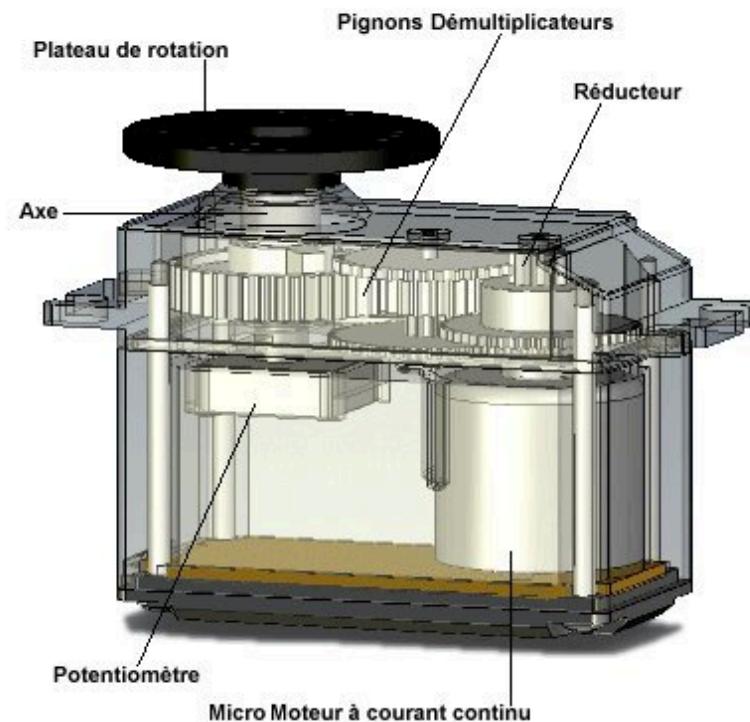


dispositif :



Les servo-moteurs

source : <http://eskimon.fr/287-arduino-602-un-moteur-qui-de-la-tete-le-servo-moteur>



Les servo-moteurs

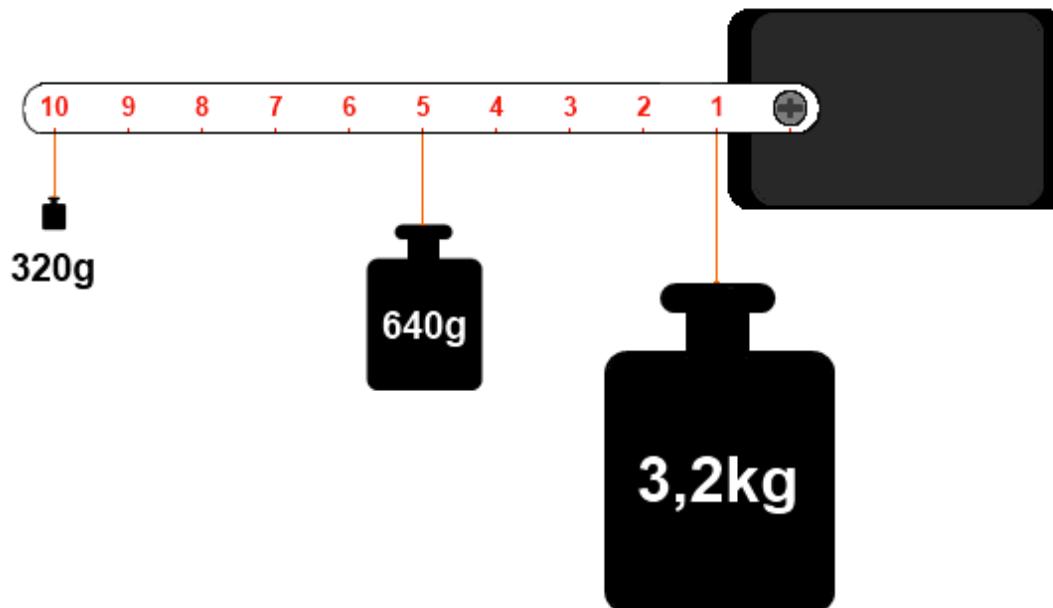
Eric G

CONNECTIQUE

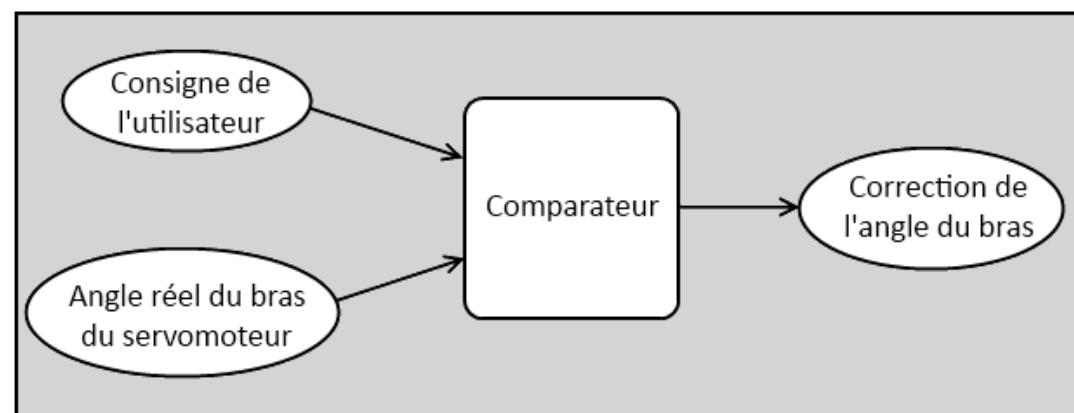
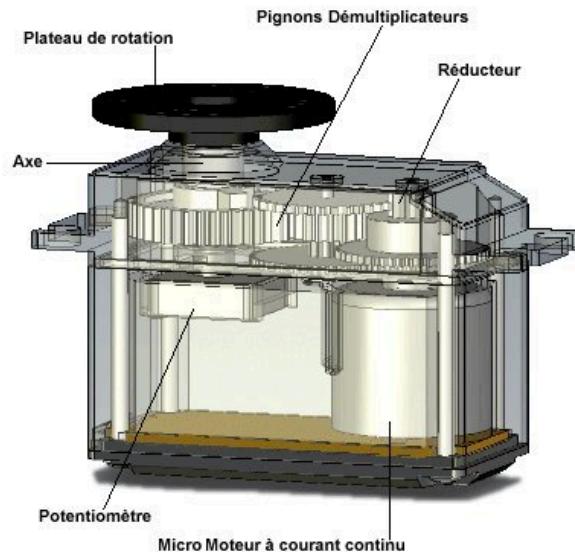
Le servomoteur a besoin de trois fils de connexion pour fonctionner. Deux fils servent à son alimentation, le dernier étant celui qui reçoit le signal de commande :

- **rouge** : pour l'alimentation positive (4.5V à 6V en général)
- **noir ou marron** : pour la masse (0V)
- **orange, jaune, blanc, ...** : entrée du signal de commande

Le couple :



Les servo-moteurs



Synoptique de fonctionnement de l'asservissement du servomoteur

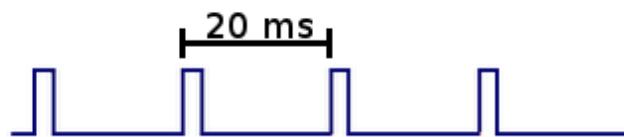
Les servo-moteurs

Le signal de commande

La consigne envoyée au servomoteur n'est autre qu'un signal électronique de type PWM. Il dispose cependant de deux caractéristiques indispensables pour que le servo puisse comprendre ce qu'on lui demande. À savoir : une fréquence fixe de valeur 50Hz (comme celle du réseau électrique EDF) et d'une durée d'état HAUT elle aussi fixée à certaines limites. Nous allons étudier l'affaire.

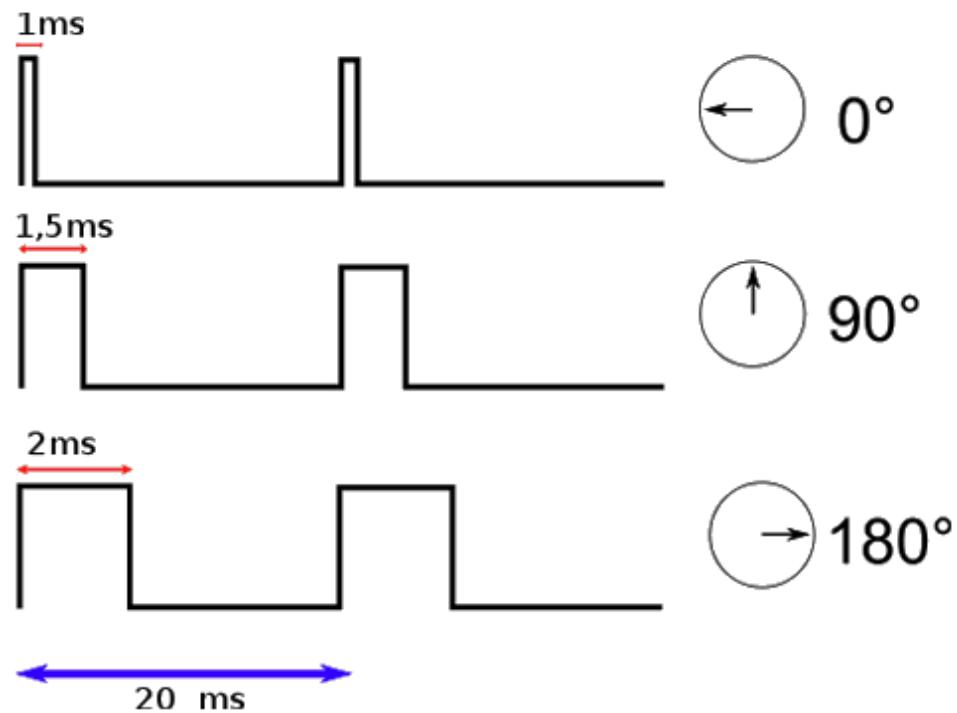
LA FRÉQUENCE FIXE

Le signal que nous allons devoir générer doit avoir une fréquence de 50 Hz. Autrement dit, le temps séparant deux fronts montants est de 20 ms. Je rappelle la formule qui donne la relation entre la fréquence (F) et le temps de la période du signal (T) : $F = \frac{1}{T}$

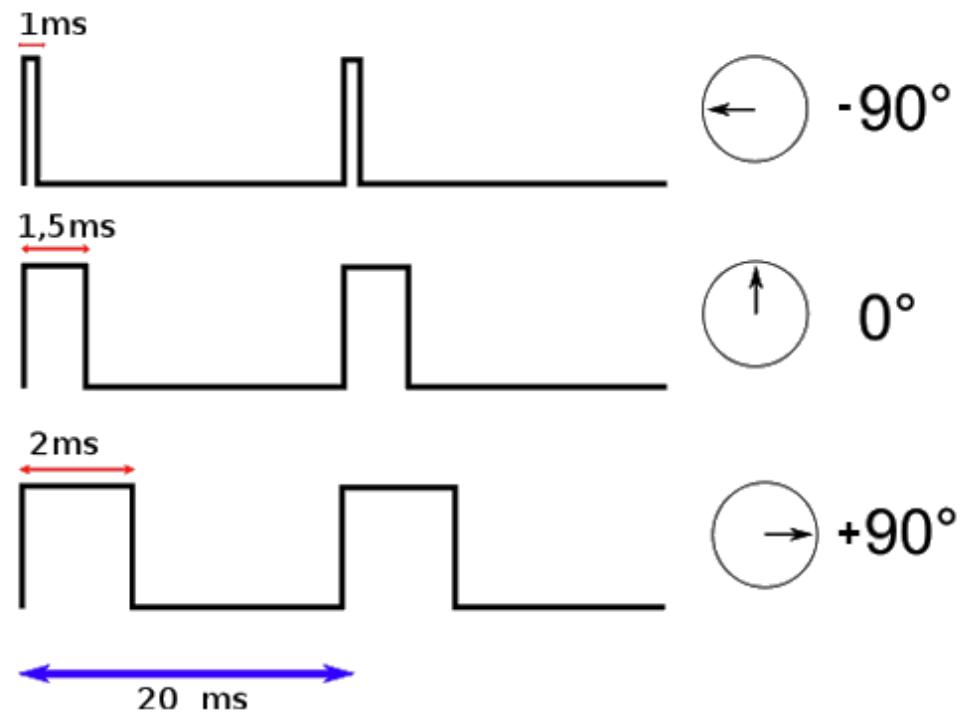


Malheureusement ,la fonction analogWrite() de Arduino ne possède pas une fréquence de 50Hz, mais dix fois plus élevée, de 500Hz environ. On ne pourra donc pas utiliser cette fonction.

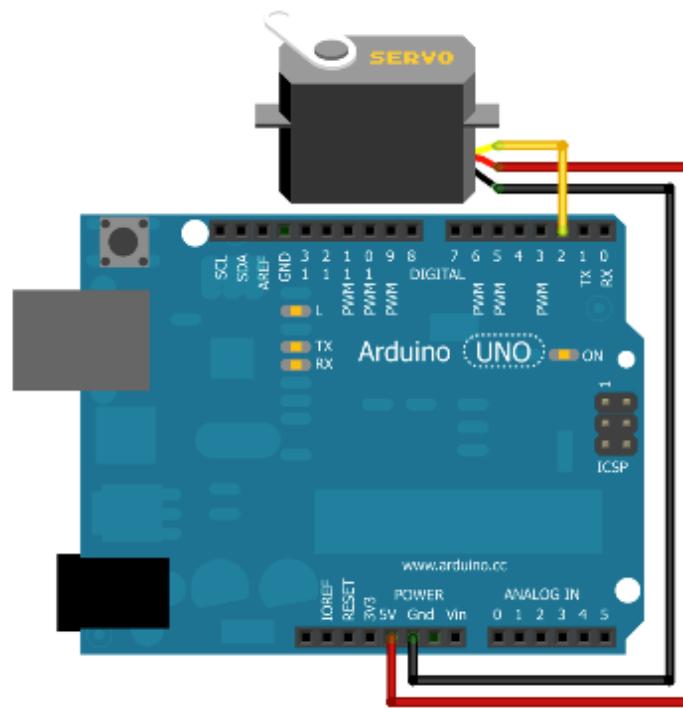
Les servo-moteurs



Les servo-moteurs



Les servo-moteurs



Les servo-moteurs

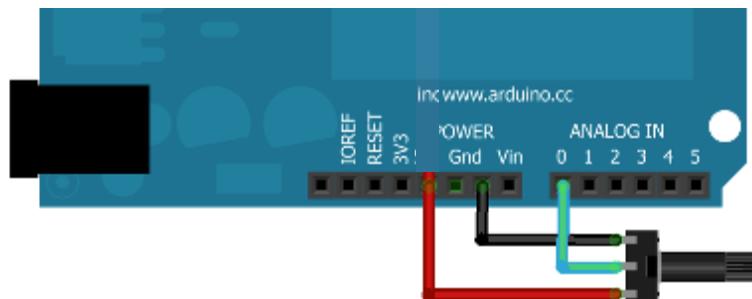
```
1 #include <Servo.h>
2
3 Servo monServo;
4
5 void setup()
6 {
7     monServo.attach(2, 1000, 2000);
8 }
```

```
1 #include <Servo.h>
2
3 Servo monServo;
4
5 void setup()
6 {
7     monServo.attach(2, 1000, 2000);
8     monServo.write(90);
9 }
10
11 void loop()
12 {
13 }
```

Ajout d'un potentiomètre ?



- doit être alimenté
- retourne un courant compris entre 0 et 5v
- doit être connecté à une entrée analogique de l'arduino



```
3 //notre potentiomètre
4 const int potar = 0;
```

```
1 void loop()
2 {
3     //on lit la valeur du potentiomètre
4     int val = analogRead(potar);
```

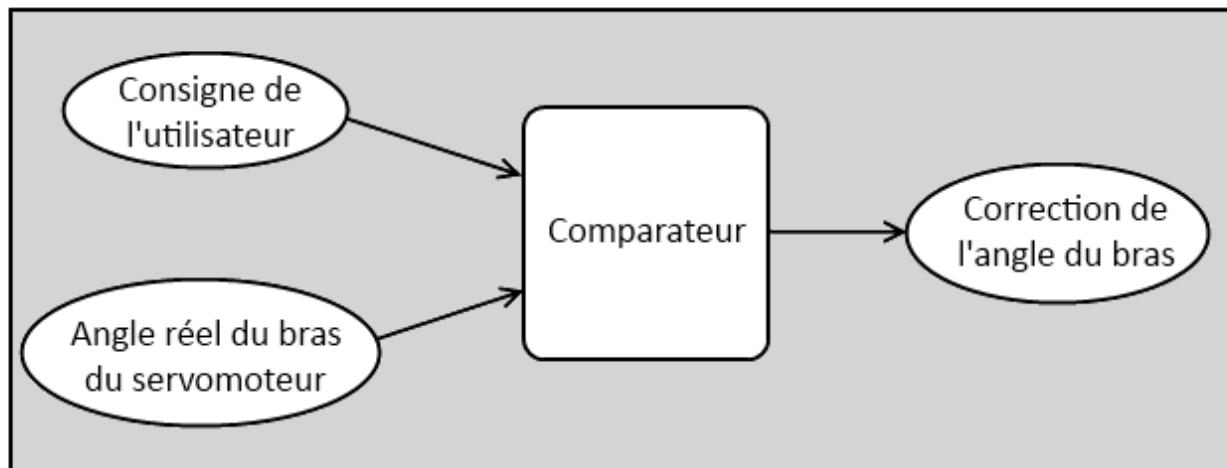
Ajout d'un potentiomètre ?

- connectez un potentiomètre
- lisez et affichez les valeurs. Notez les valeurs min et max
- convertissez les valeurs en degrés et commandez le servo-moteur pour qu'il tourne comme le potentiomètre

- enlevez le potentiomètre
- pilotez votre servo-moteur depuis un programme en C:

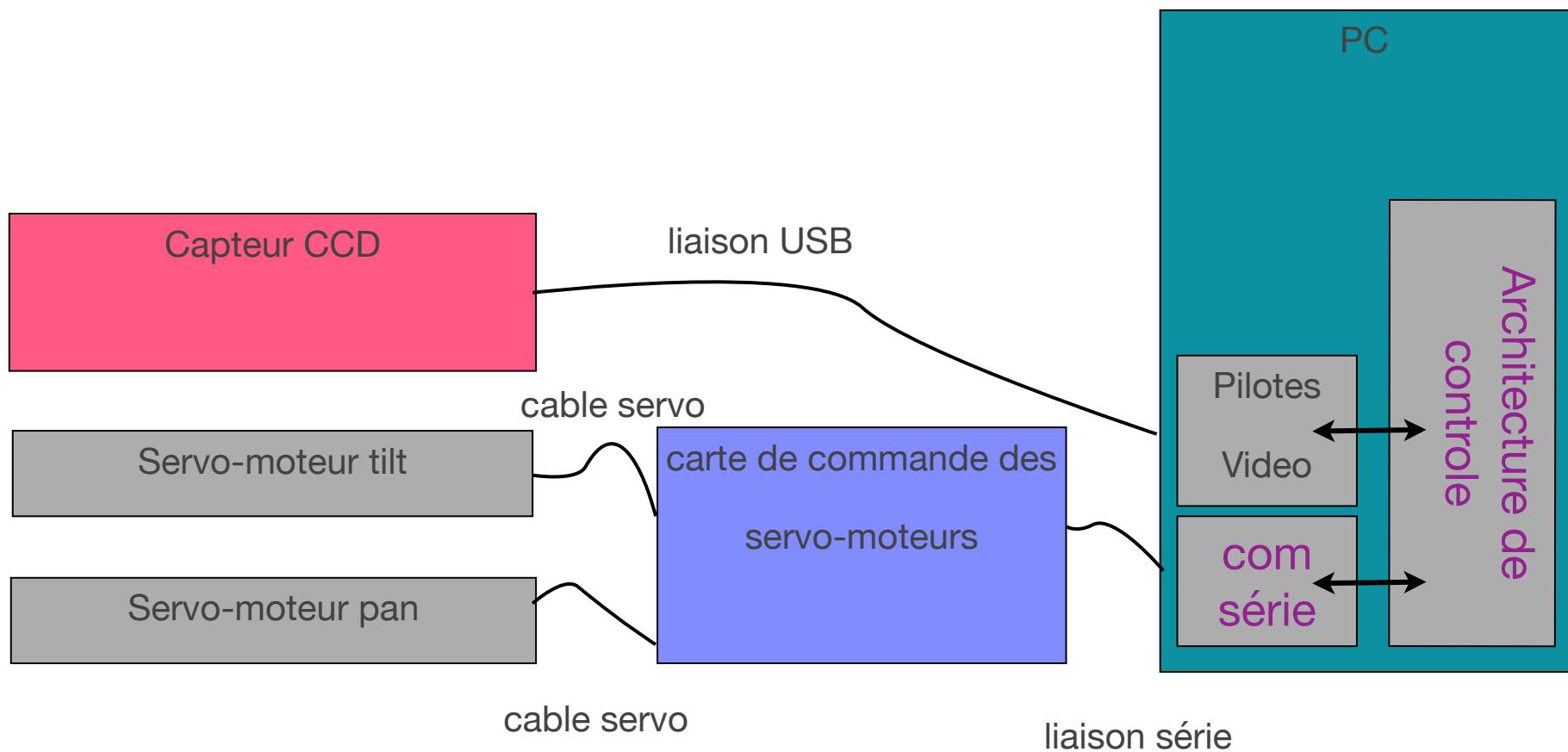
‘e’ signifie aller vers le haut
‘f’ la droite
‘s’ la gauche
‘c’ le bas

Les servo-moteurs



Synoptique de fonctionnement de l'asservissement du servomoteur

dispositif :



Les grandes étapes (1)

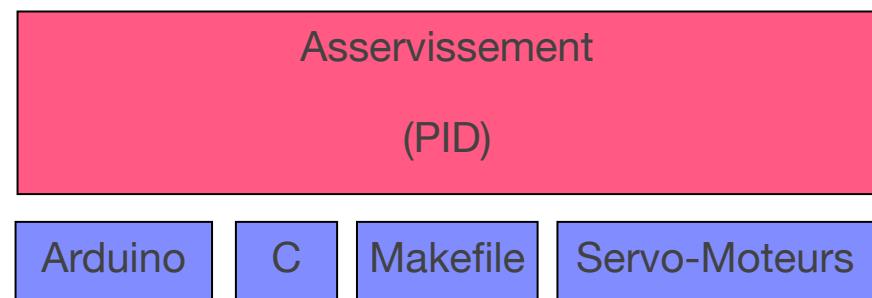
Maitriser le contrôle moteur

monter les servo-moteurs

choisir la carte de développement

établir la liaison série (programme en C)

vérifier le contrôle en position



Les grandes étapes (2)

Maitriser l'acquisition video

routine de vérification des pilotes

installer openCV

comprendre l'espace pixel

Traitement d'image simple

sélection d'une couleur

détection du mouvement

Traitement d'image et vision simple

B&W -- couleur

C

Makefile

Les grandes étapes (3)

Concevoir l'architecture de contrôle

“relier” code video et code de contrôle des servo-moteurs

dans quel espace travailler ?

quelles opérations faire ?

convergence du système ?

maitrise des espaces

maitrise des équations

maitrise du temps

Architectures de contrôle

C

Makefile

Temps réel

Maitriser le controle moteur

Tracking/poursuite

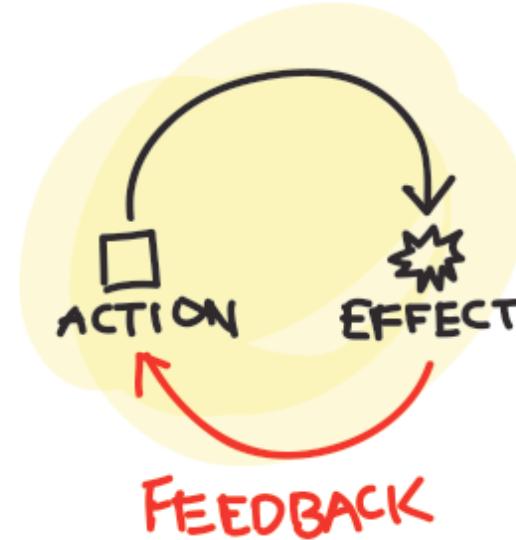
Maitriser le contrôle moteur

Tracking/poursuite : rejoindre la cible visuelle

asservissement : pour atteindre la cible : réduire l'écart entre la valeur actuelle et la valeur de consigne.

notion d'erreur.

boucle fermée.



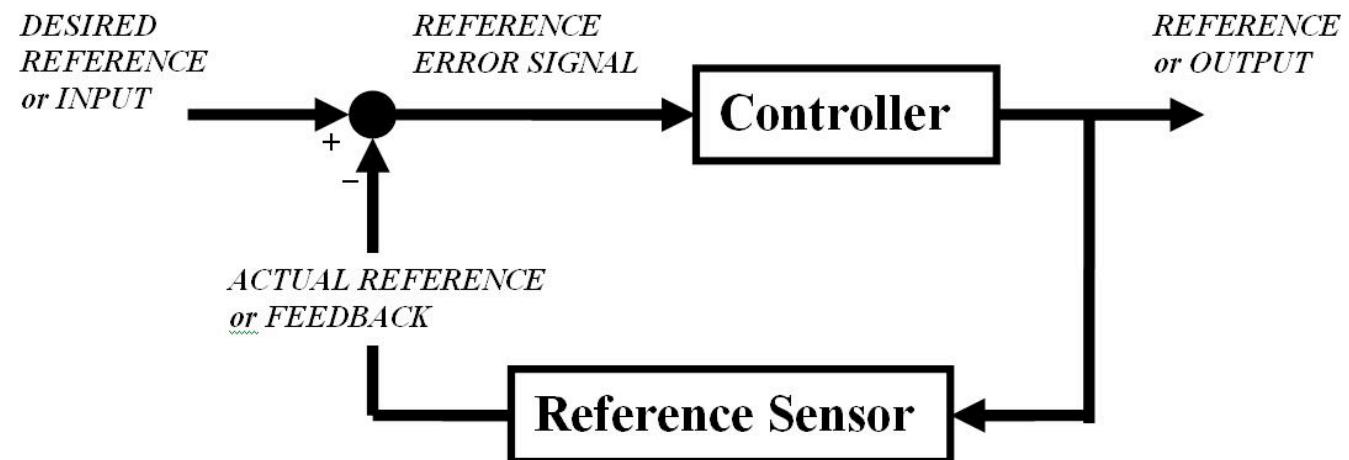
Maitriser le contrôle moteur

Tracking/poursuite : rejoindre la cible visuelle

asservissement : pour atteindre la cible : réduire l'écart entre la valeur actuelle et la valeur de consigne.

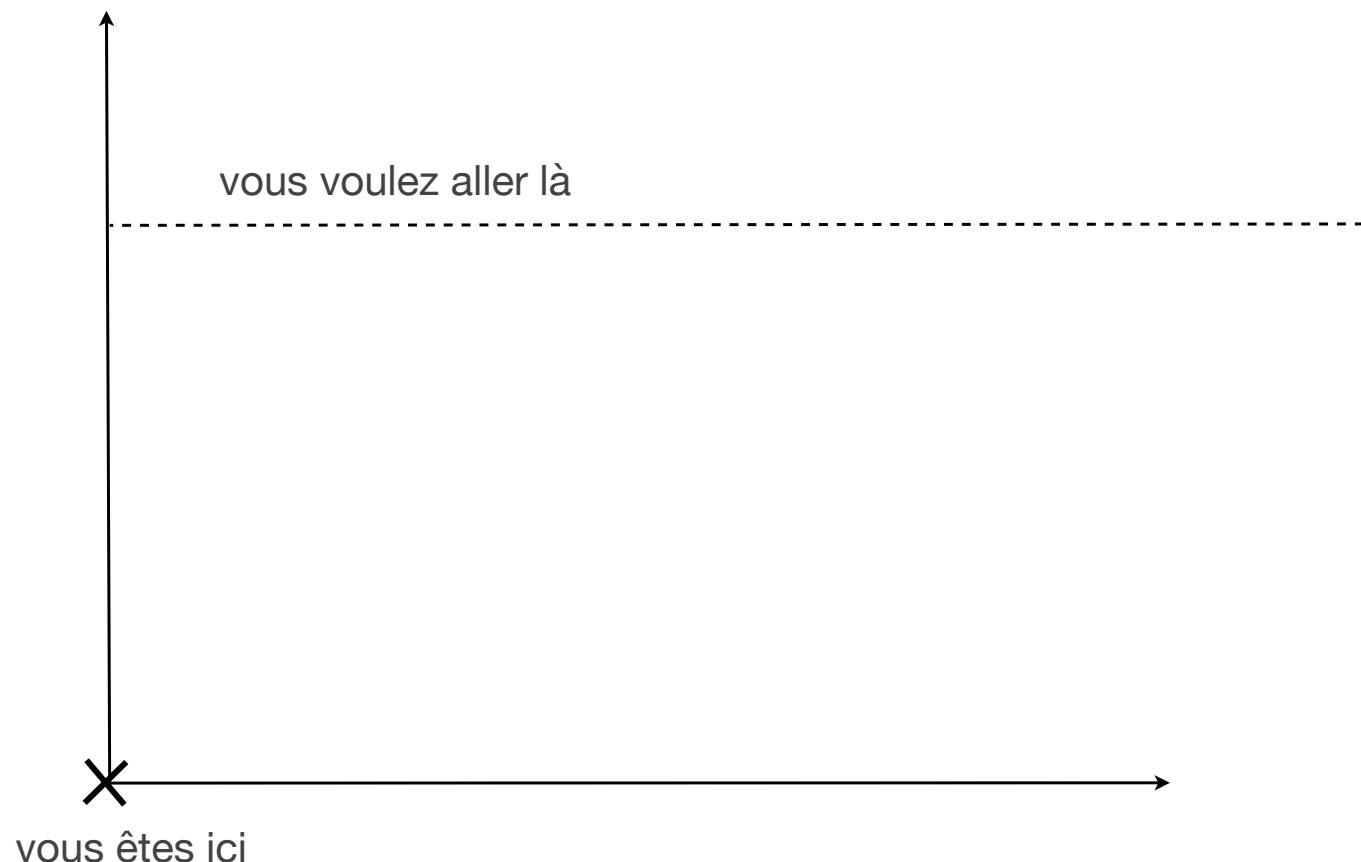
notion d'erreur.

boucle fermée.



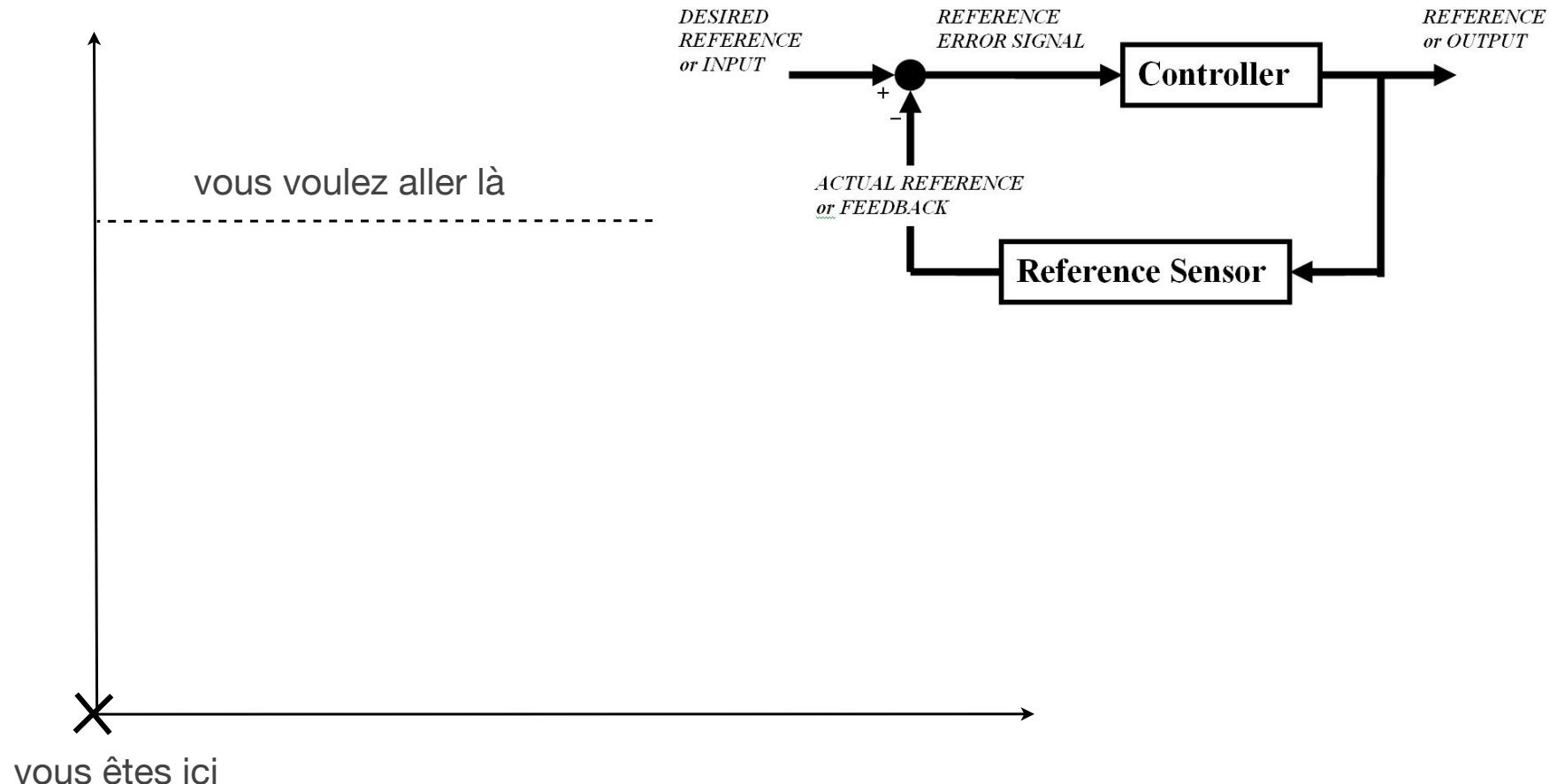
Maitriser le contrôle moteur

Tracking/poursuite : rejoindre la cible visuelle



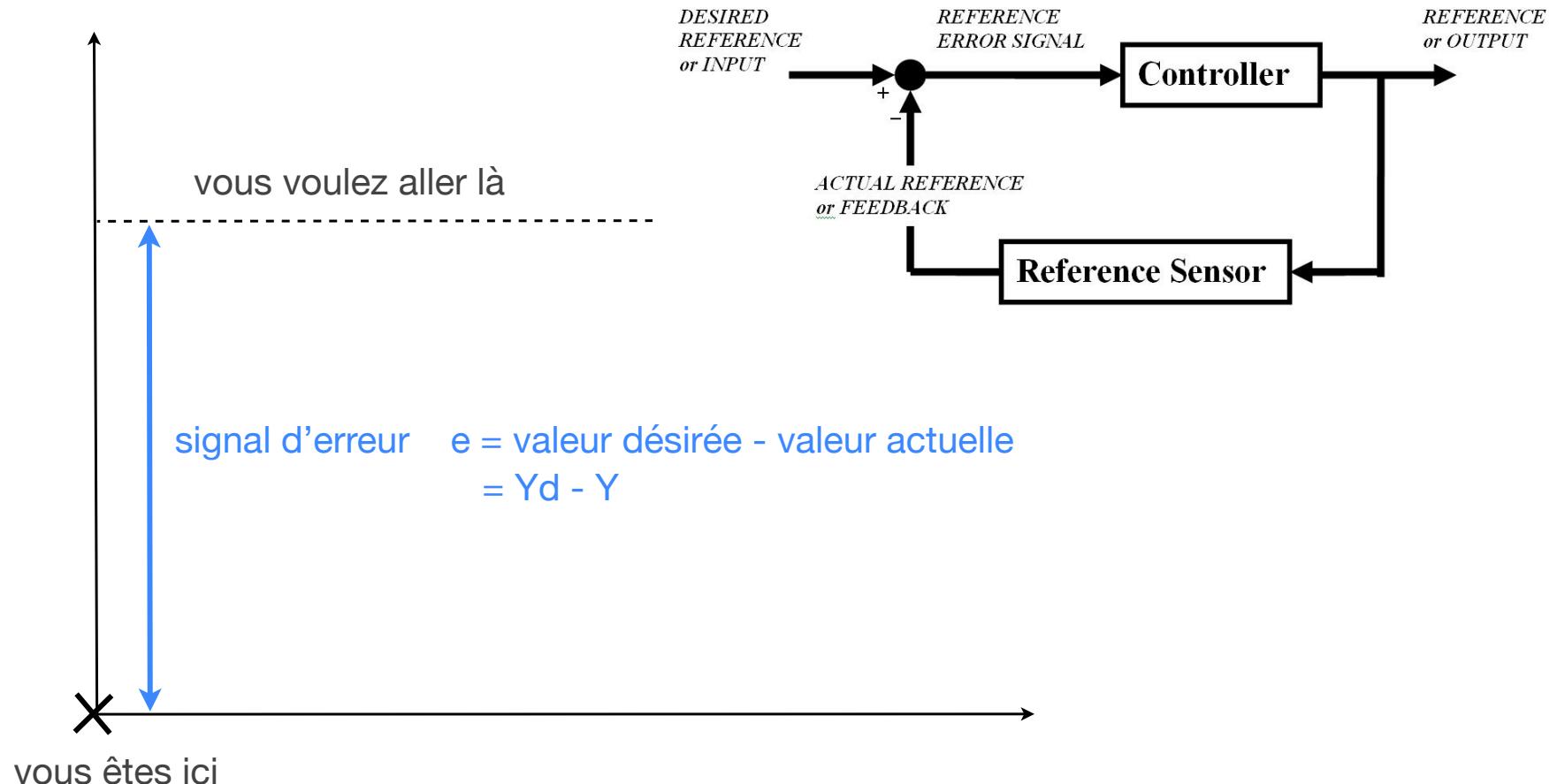
Maitriser le contrôle moteur

Tracking/poursuite : rejoindre la cible visuelle



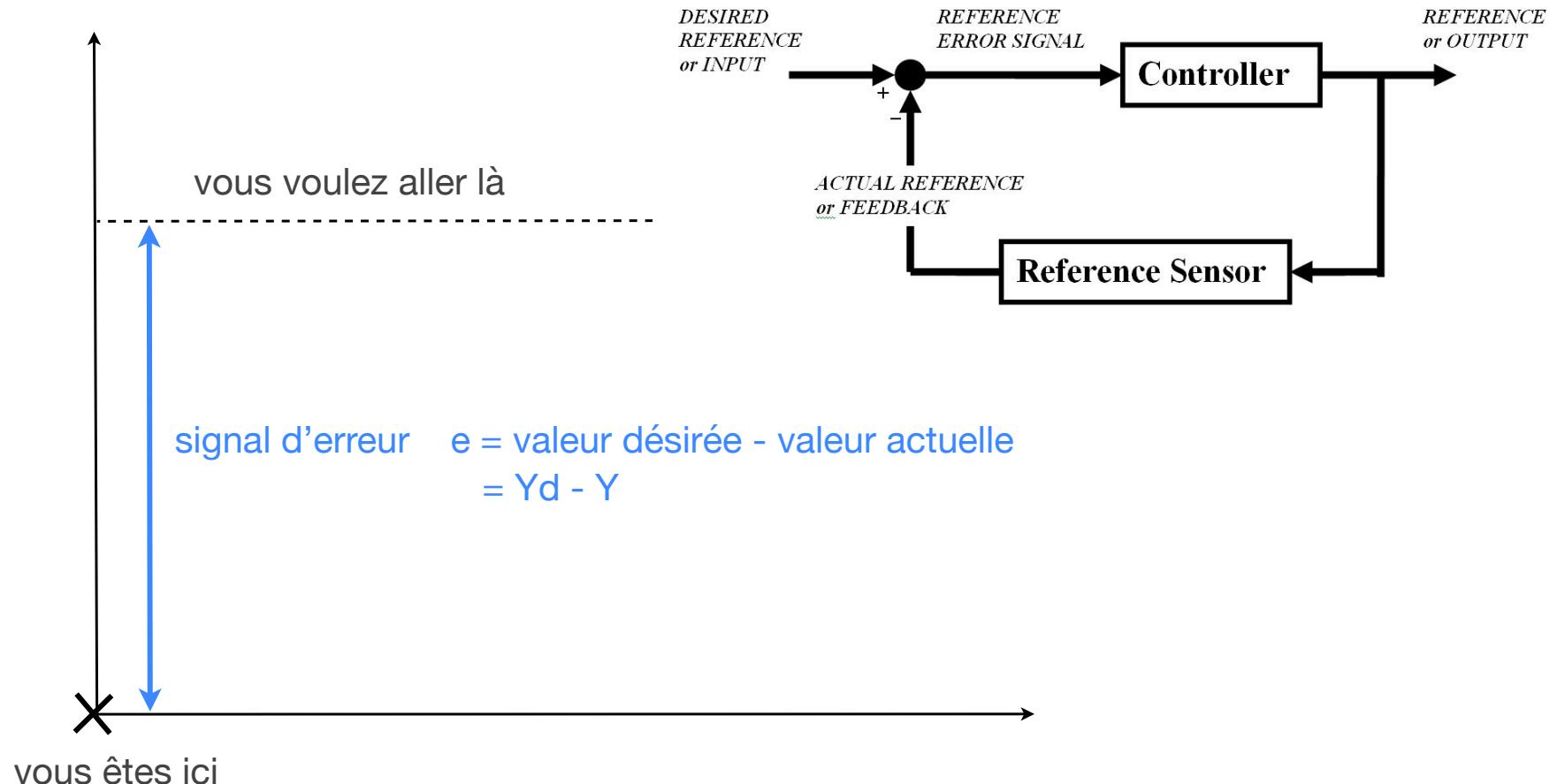
Maitriser le contrôle moteur

Tracking/poursuite : rejoindre la cible visuelle



Maitriser le contrôle moteur

1 ère solution : on envoie au contrôleur la totalité de l'erreur



Maitriser le contrôle moteur

1 ère solution : on envoie au contrôleur la totalité de l'erreur:

$$\text{Output} = e(t) \cdot$$

avec $e(t)$, l'erreur calculée à l'instant t .

-simple.

- inconvénients ?

rien ne dit que l'erreur ne soit calculée dans l'espace moteur (il y a une transformation souvent complexe à faire). Donc rien ne dit que l'erreur ne corresponde à la commande à faire (une force, un temps, une pression, un courant, etc...).

même si la transformation d'espace est faite, donner une consigne égale à la totalité de l'erreur = grand mouvement = paramètres forts.

Ne marche que si la cible a rejoindre est immobile, où si l'effecteur est très rapide

inapproprié au contrôle en vitesse

Maitriser le contrôle moteur

Contrôle en position : la consigne est la position à atteindre.

Contrôle en vitesse : la consigne est une vitesse

Contrôle en force : la consigne est une force, une tension, une pression

Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec $e(t)$, l'erreur calculée à l'instant t .

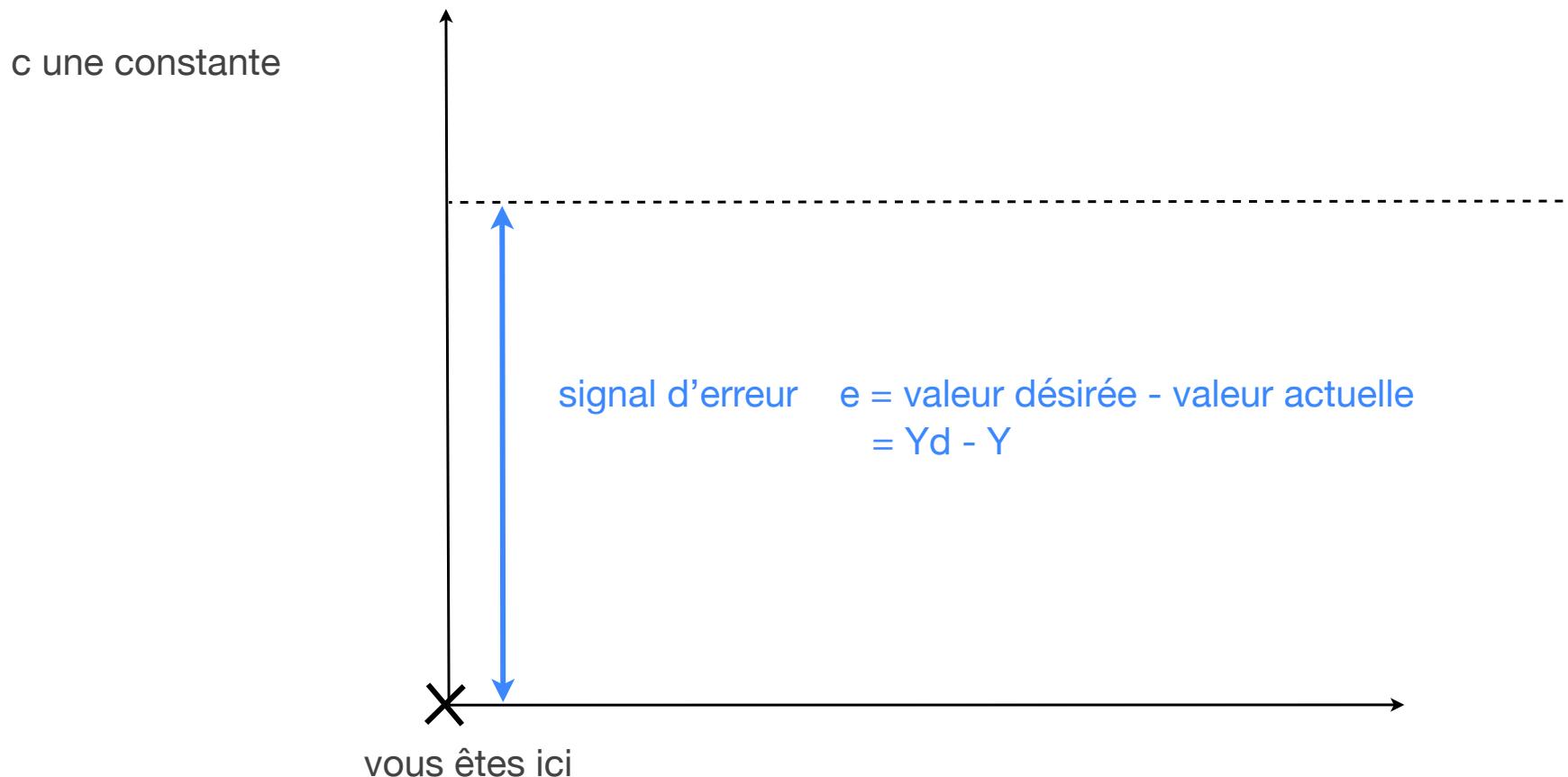
c une constante

Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec $e(t)$, l'erreur calculée à l'instant t .

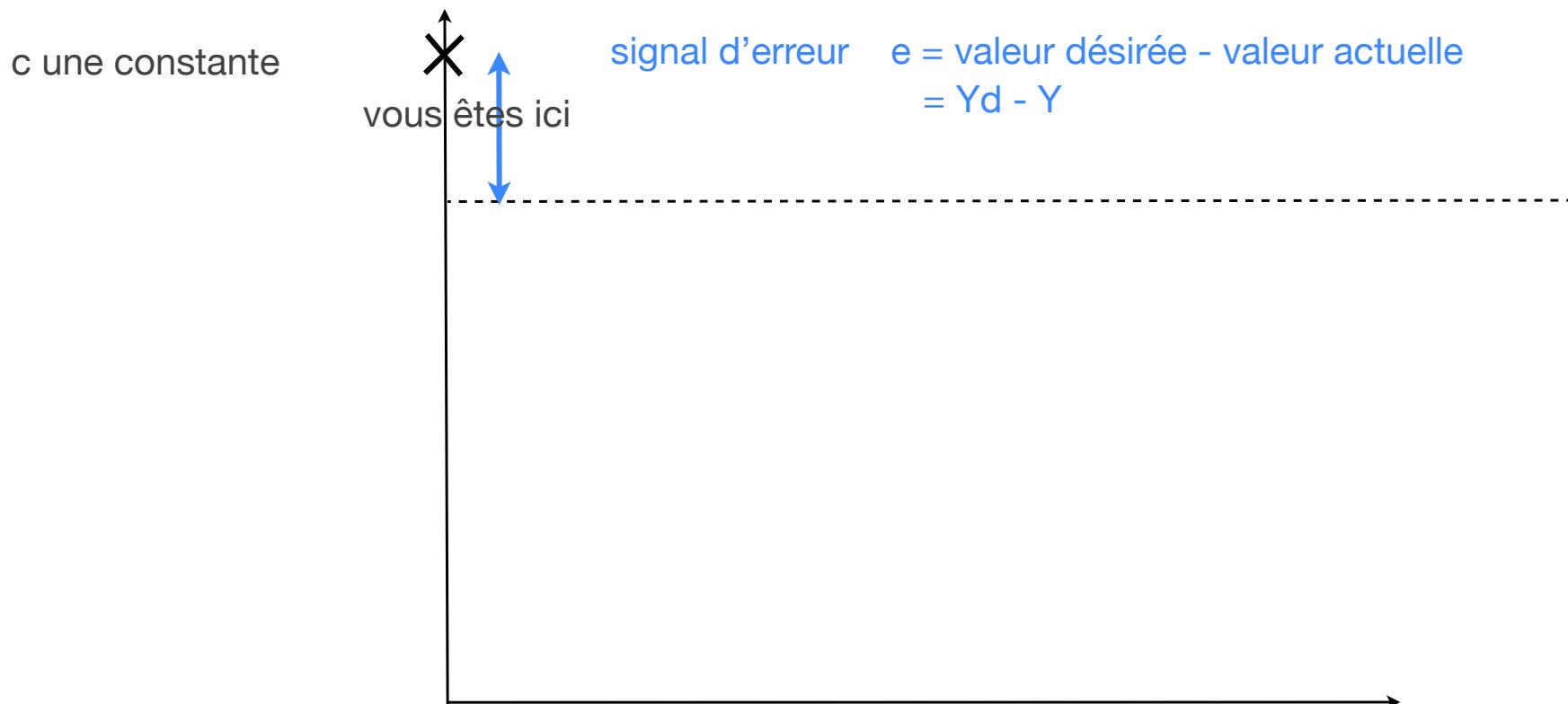


Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

avec $e(t)$, l'erreur calculée à l'instant t .



Maitriser le contrôle moteur

2 ème solution : on envoie au contrôleur une quantité fixe selon le signe de l'erreur

$$\text{Output} = c \cdot \text{sign}(e(t))$$

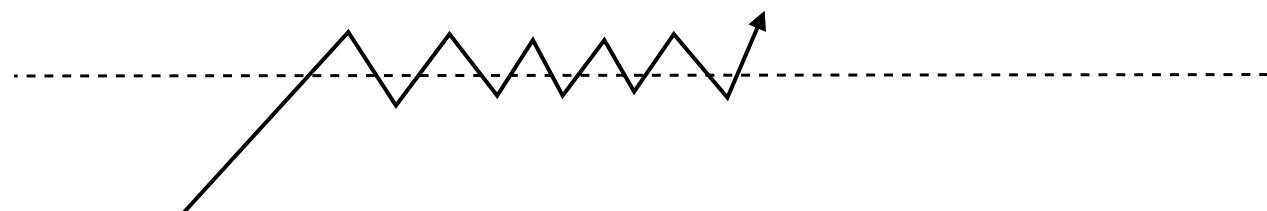
avec $e(t)$, l'erreur calculée à l'instant t .

c une constante

-simple.

- inconvénients ?

c étant une constante, si l'erreur est inférieure a c, on ne converge pas vers la solution



Maitriser le contrôle moteur

3 ème solution : on envoie au contrôleur une quantité proportionnelle à l'erreur

$$\text{Output} = K_p e(t)$$

avec $e(t)$, l'erreur calculée à l'instant t .

K_p un coefficient multiplicateur : facteur proportionnel à l'erreur

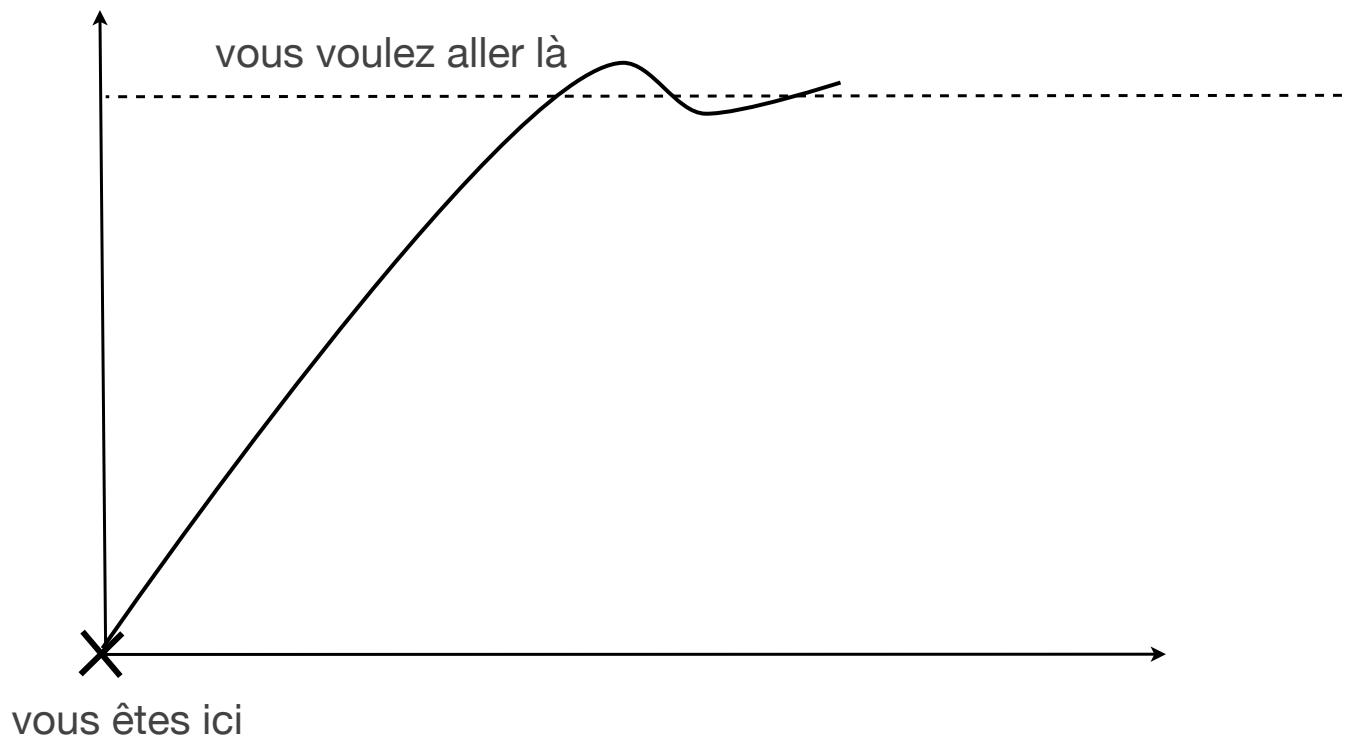
Maitriser le contrôle moteur

3 ème solution : on envoie au contrôleur une quantité proportionnelle à l'erreur

$$\text{Output} = K_p e(t)$$

avec $e(t)$, l'erreur calculée à l'instant t .

K_p un coefficient multiplicateur : facteur proportionnel à l'erreur



Maitriser le contrôle moteur

3 ème solution : on envoie au contrôleur une quantité proportionnelle à l'erreur

$$\text{Output} = K_p e(t)$$

avec $e(t)$, l'erreur calculée à l'instant t .

K_p un coefficient multiplicateur : facteur proportionnel à l'erreur

-simple.

- inconvénients ?

la convergence peut prendre du temps (amortissement vers la solution)

si le système rencontre une force opposée (frottement sec, frottement, objet), il restera empêché (la commande ne changera jamais et ne gagne pas en intensité).

Maitriser le contrôle moteur

si le système rencontre une force opposée (frottement sec, frottement, objet), il restera empêché (la commande ne changera jamais et ne gagne pas en intensité).

Nous allons prendre en compte l'accumulation de l'erreur : intégrale de l'erreur

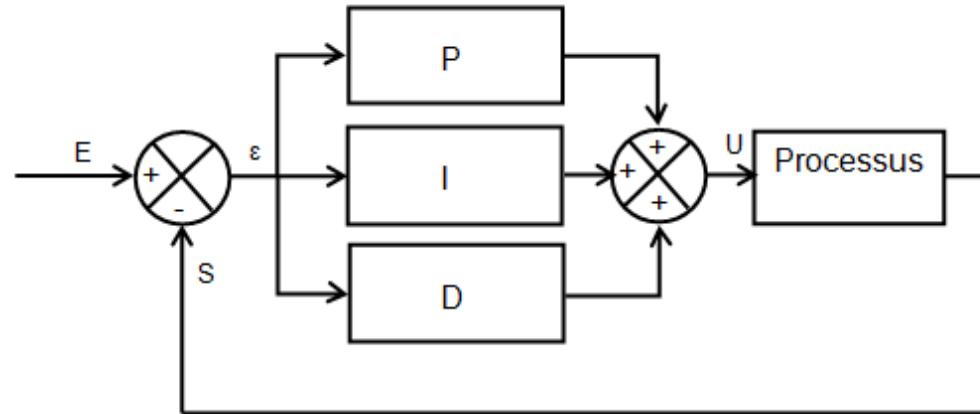
$$K_I \int e(t) dt$$

la convergence peut prendre du temps (amortissement vers la solution)

Nous allons prédire la direction du mouvement : dérivée de l'erreur

$$K_D \frac{d}{dt} e(t)$$

Maitriser le contrôle moteur



$$\text{Output} = K_P e(t) + K_I \int e(t) dt + K_D \frac{d}{dt} e(t)$$

Where : $e = \text{Setpoint} - \text{Input}$

K_p : Proportional gain, a tuning parameter

K_i : Integral gain, a tuning parameter

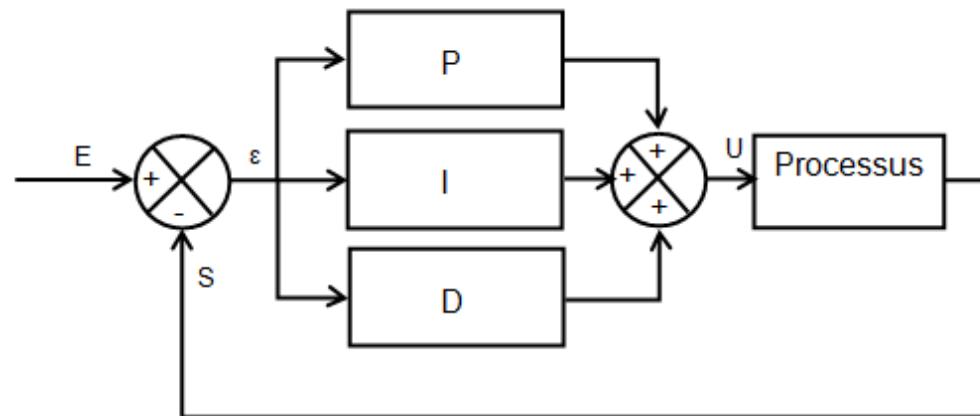
K_d : Derivative gain, a tuning parameter

e : Error = $SP - PV$

t : Time or instantaneous time (the present)

τ : Variable of integration; takes on values from time 0 to the present t .

Maitriser le contrôle moteur



$$\text{Output} = K_P e(t) + K_I \int e(t) dt + K_D \frac{d}{dt} e(t)$$

Where : $e = \text{Setpoint} - \text{Input}$

K_p : Proportional gain, a tuning parameter

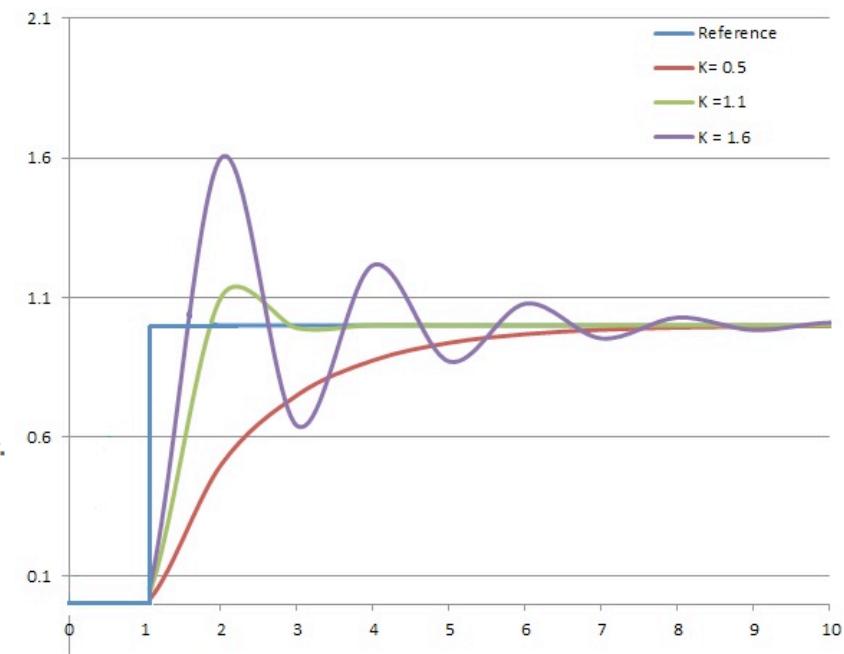
K_i : Integral gain, a tuning parameter

K_d : Derivative gain, a tuning parameter

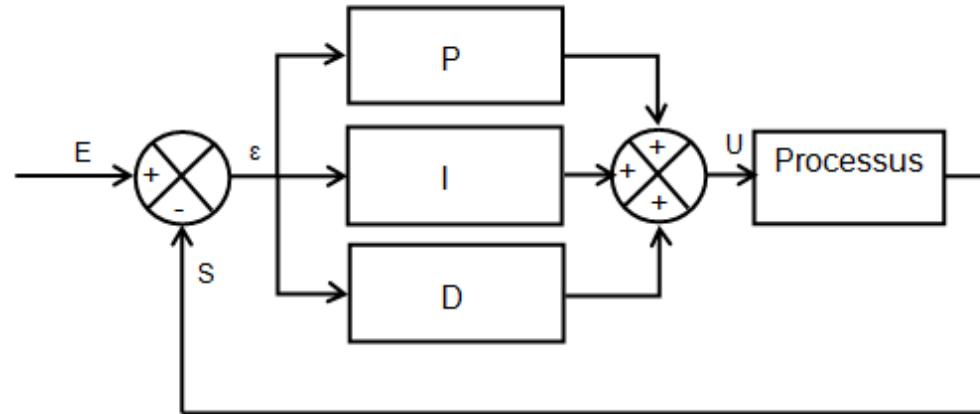
e : Error = $SP - PV$

t : Time or instantaneous time (the present)

τ : Variable of integration; takes on values from time 0 to the present t .



Maitriser le contrôle moteur



$$\text{Output} = K_P e(t) + K_I \int e(t) dt + K_D \frac{d}{dt} e(t)$$

Where : e = Setpoint - Input

K_p : Proportional gain, a tuning parameter

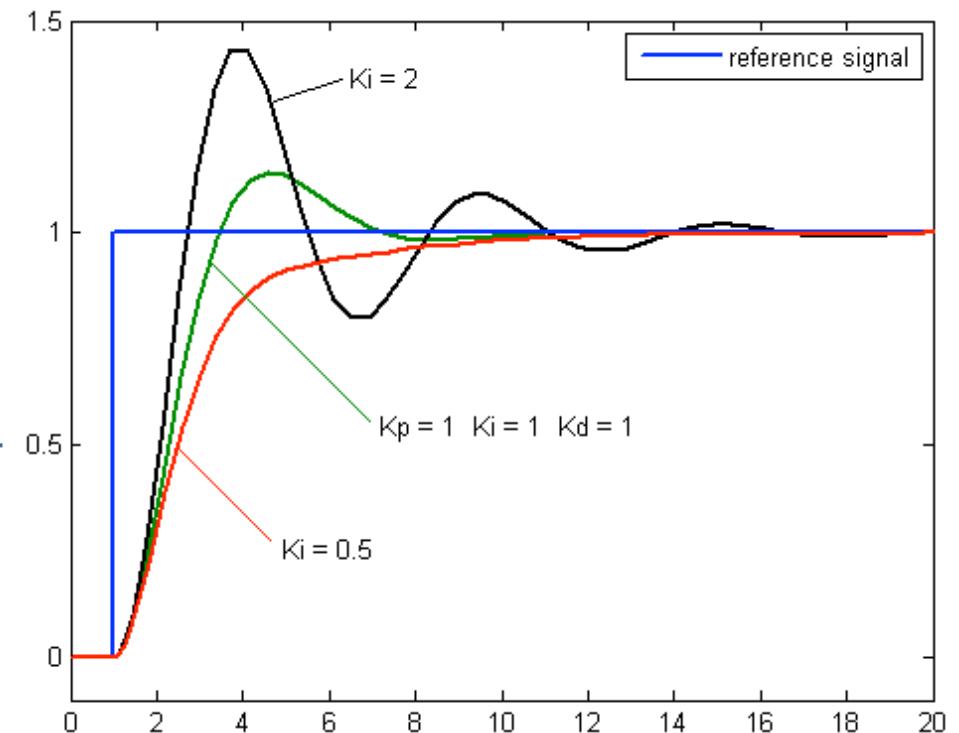
K_i : Integral gain, a tuning parameter

K_d : Derivative gain, a tuning parameter

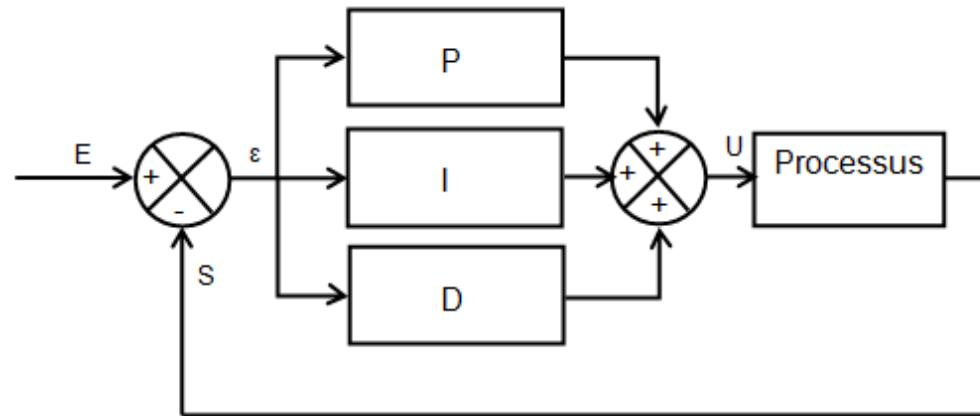
e : Error = $SP - PV$

t : Time or instantaneous time (the present)

τ : Variable of integration; takes on values from time 0 to the present t .



Maitriser le contrôle moteur



$$\text{Output} = K_P e(t) + K_I \int e(t) dt + K_D \frac{d}{dt} e(t)$$

Where : $e = \text{Setpoint} - \text{Input}$

K_p : Proportional gain, a tuning parameter

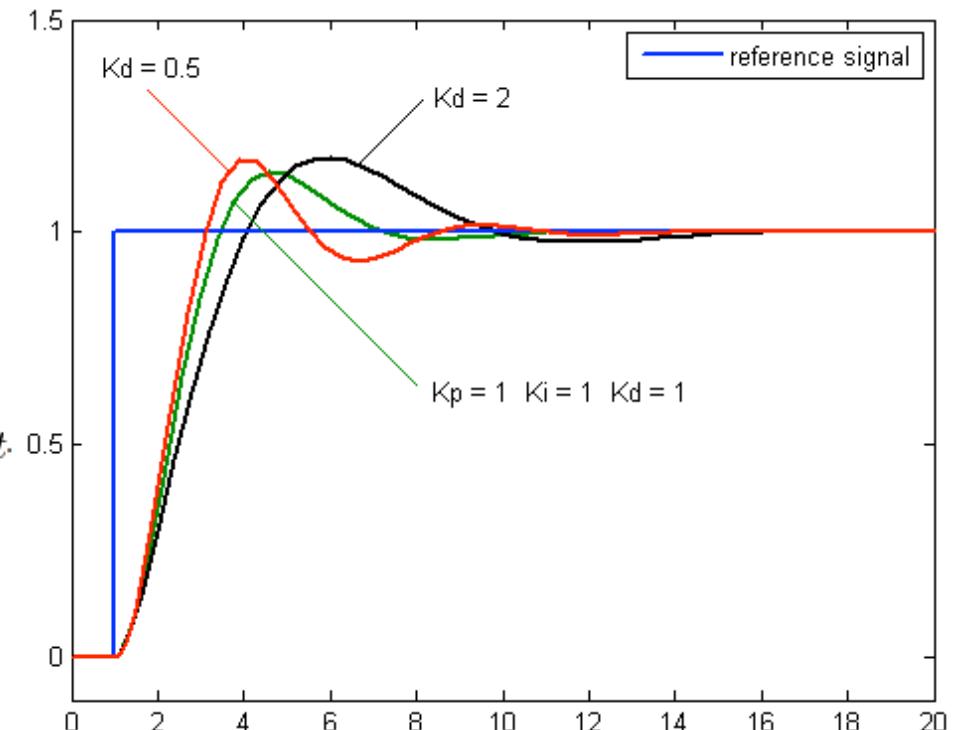
K_i : Integral gain, a tuning parameter

K_d : Derivative gain, a tuning parameter

e : Error = $SP - PV$

t : Time or instantaneous time (the present)

τ : Variable of integration; takes on values from time 0 to the present t .



Image



Image



Centre de l'image

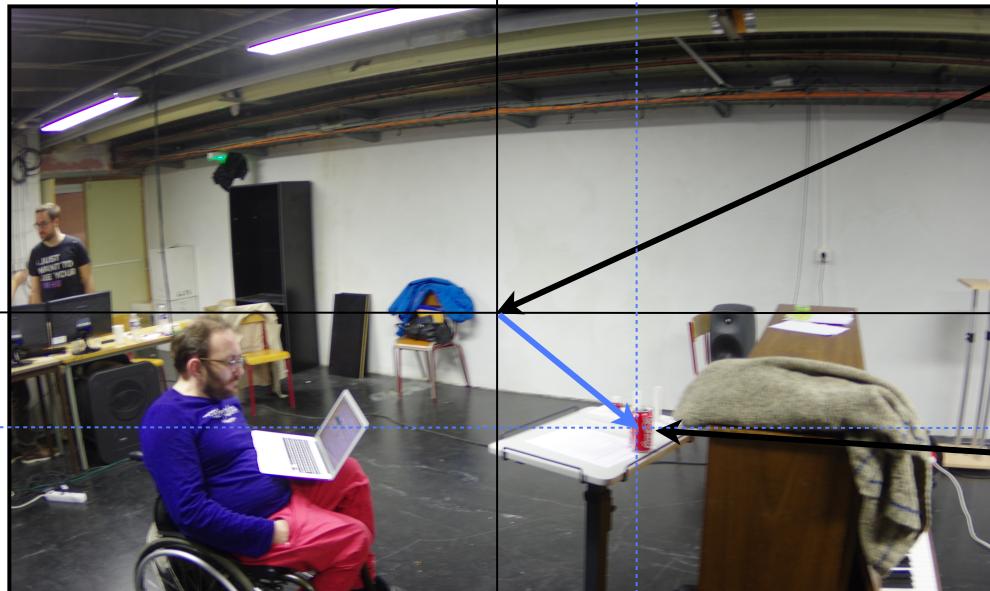
Image



Centre de l'image

cible

Image

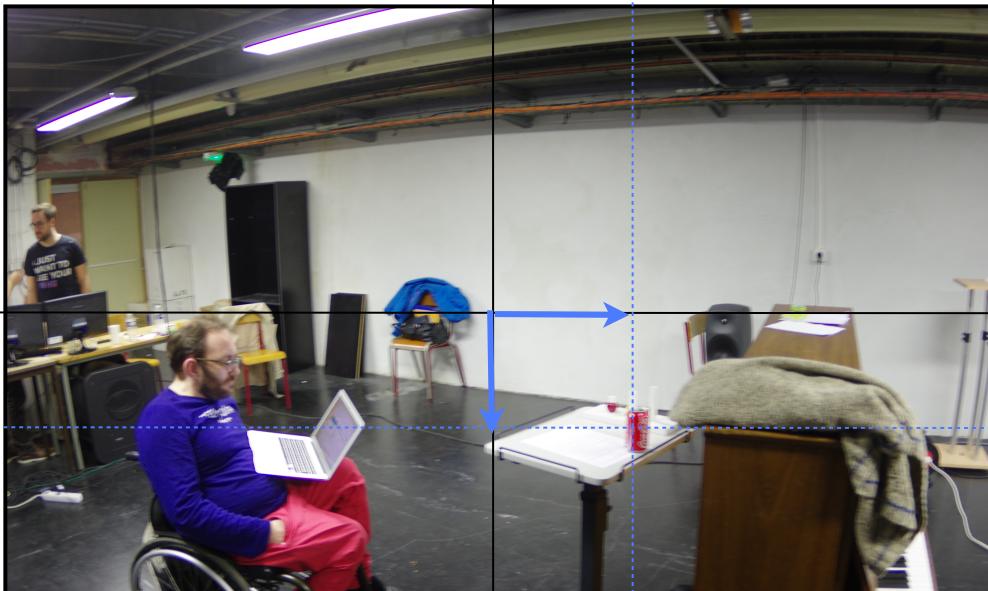


Centre de l'image

cible

vecteur de
déplacement

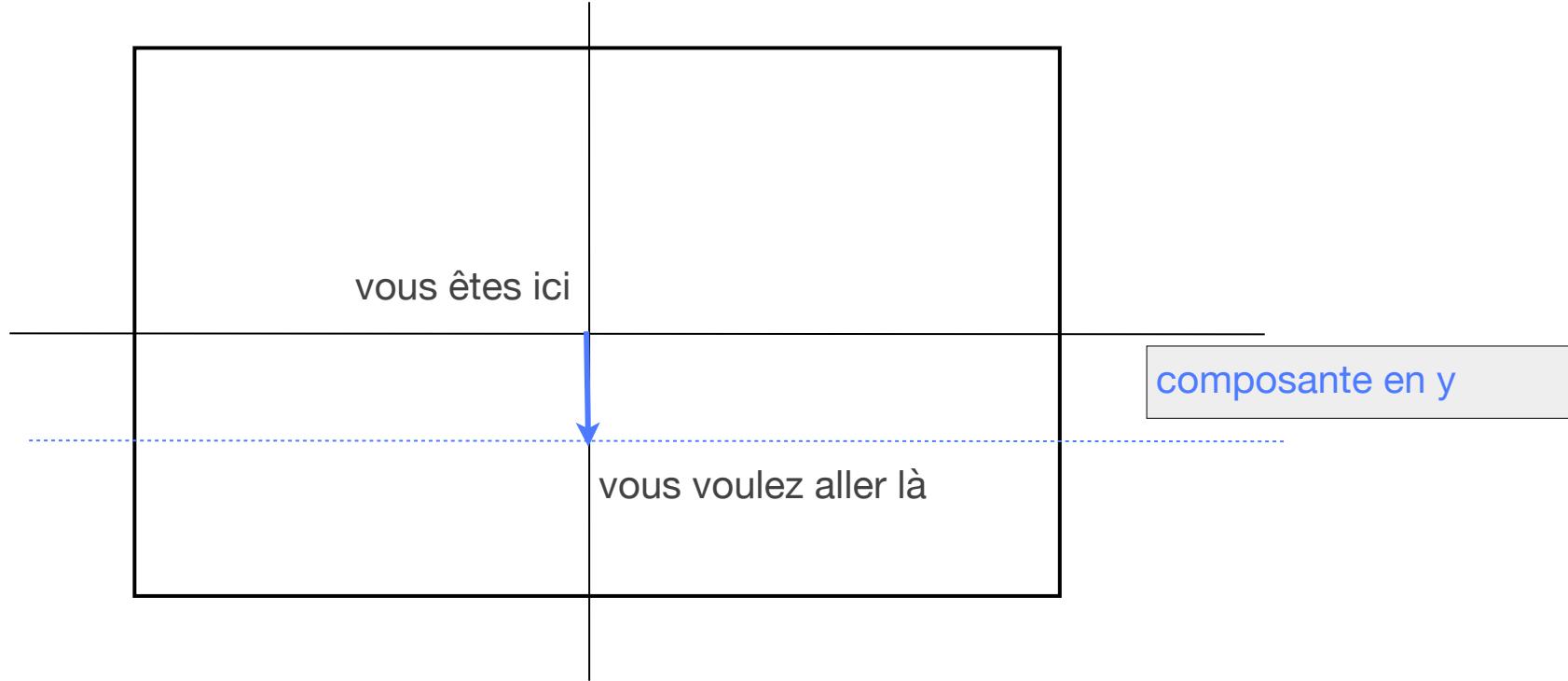
Image



composante en y

composante en x

Image

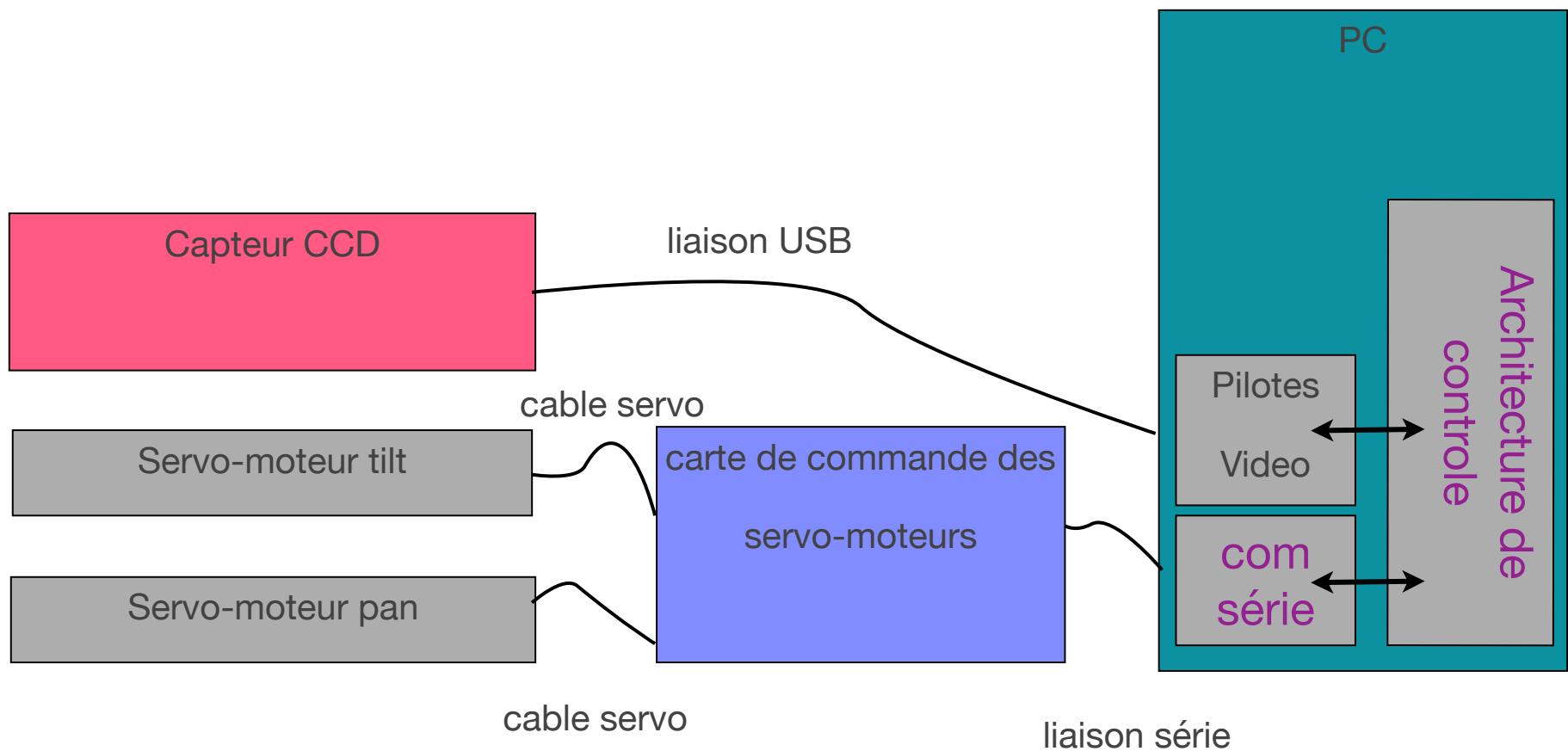


Dans quel espace est calculé la distance entre l'origine et la cible (l'erreur) ?

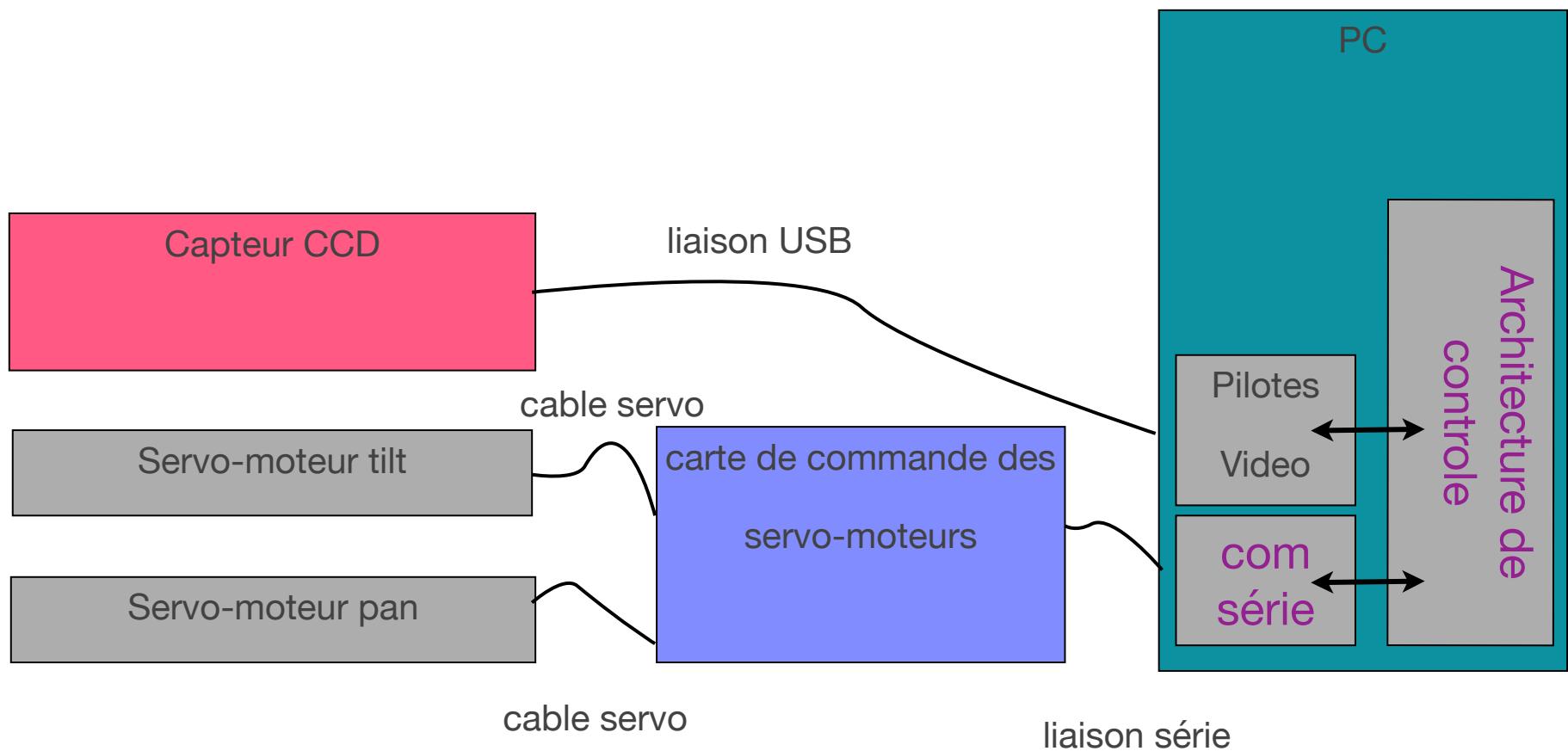
Dans quel espace est exprimé la commande ?

Architectures de contrôle

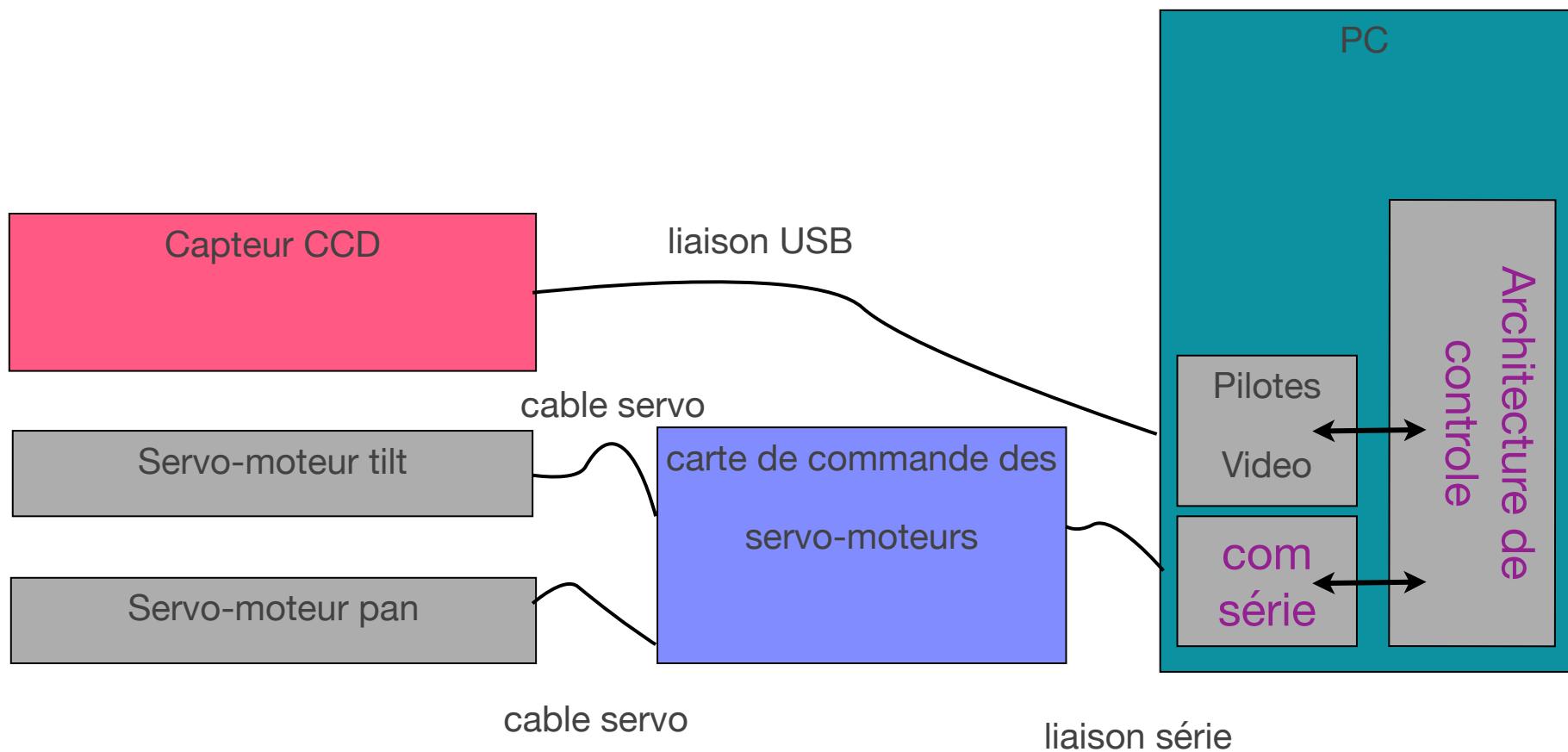
dispositif :



dispositif :



dispositif :



Image

```
namedWindow("MyCam",1);

cap.set(CV_CAP_PROP_FRAME_WIDTH,320); //taille de la fenetre
cap.set(CV_CAP_PROP_FRAME_HEIGHT,240); //au dela de 320*240
```

```
while(1){
    if(cap.read(frame)){// get a new frame from camera

        count = 0;
        for( it = frame.begin<Vec3b>(), end = frame.end<Vec3b>(); it != end; ++it)
        {
            // (*it) [0] = /*accès au R */
            // (*it) [1] = /* le G */
            // (*it)[2] = /* le B*/
            count++;
        }

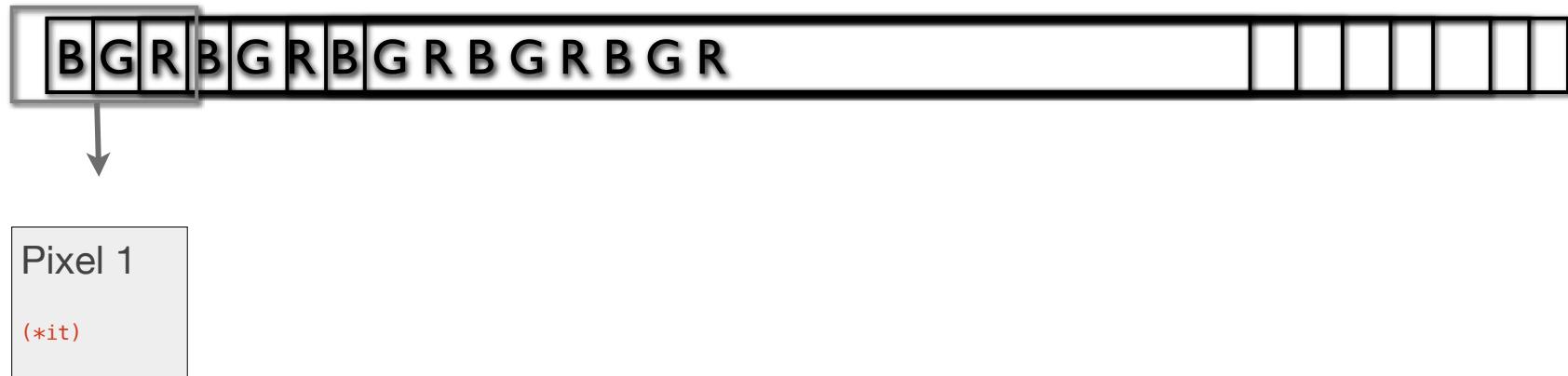
        printf("count : %d\n",count);

        imshow("MyCam", frame);
    }
}
```

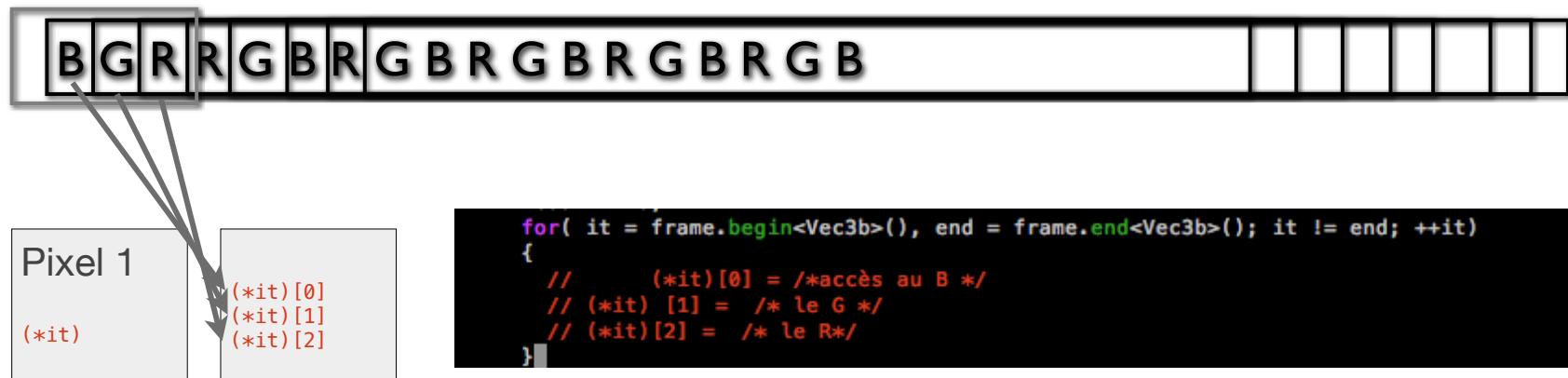
```
count : 76800
```

On a bien $320 \times 240 = 76\ 800$ pixels

Image



Image



Déetecter les pixels rouges

