1. **Asservissement**
   1. **Principe de l’asservissement**

Lorsque l’on souhaite faire avancer un robot, on ne peut transmettre à celui-ci qu’une seule consigne. Idéalement, si le système est parfait et que l’on sait exactement une telle tension donne une telle vitesse, une seule consigne peut être suffisante pour avancer le robot. Mais la réalité, en raison de la perturbation et de l’imperfection du système, les erreurs sont sûrement garanties. Afin de résoudre ce problème et de réduire les erreurs produites, il y a besoin d’avoir un retour sur la consigne transmise. On a donc besoin des capteurs pour mesurer la sortie et envoyer le retour, qui permet d’ajuster la consigne correspondant pour corriger les erreurs.

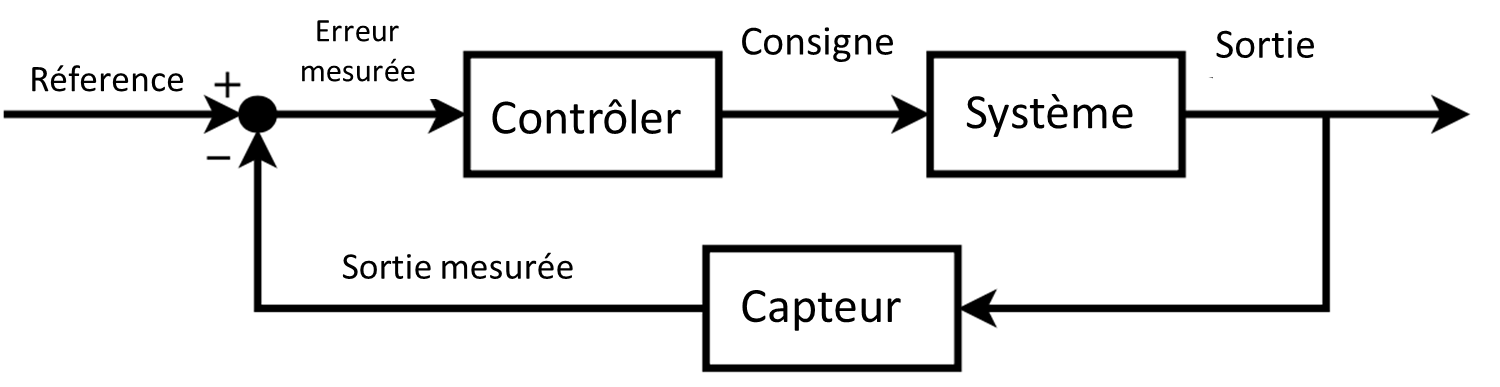
****

Figure - Pincipe d'un asservissement

Dans notre cas, on souhaite contrôler les deux moteurs du robot avec la carte Arduino et les encodeurs qui permet de mesurer la position du robot. Notre intérêt est de faire avancer le robot tout droit et de faire tourner le robot.

* 1. **Présentation des matériels**
     1. **La carte Arduino**

Nous utilisons la carte Arduino MEGA afin d’asservir le déplacement du robot car elle dispose de très nombreux pins d’entrée-sortie. En effet, elle dispose de six pins digitaux qui permettent d’utiliser la fonction « Interrupt » utilisée pour faire tourner tout le temps les encodeurs, alors que l’Arduino UNO ne dispose que deux de ces pins. Il nous donc reste de ces pins pour servir les autres capteurs qui ont besoin.

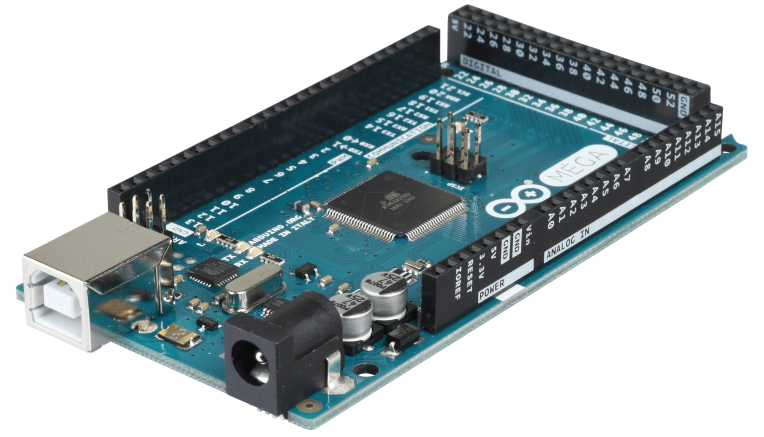
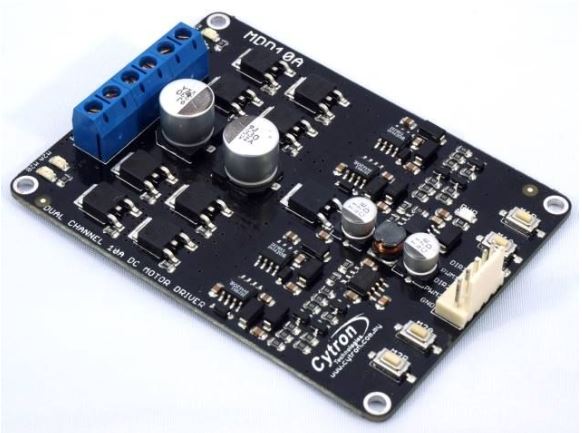


Figure : la carte puissance MDD10A

Figure : Arduino MEGA

On se rend compte d’utiliser la carte Arduino pour réaliser l’asservissement, mais la sortie PWM de l’Arduino de 0 à 5V est très faible pour contrôler le moteur de 12 V. On a donc besoin un circuit intermédiaire pour augmenter la tension sortie et aussi prévenir le courant induit par le moteur. Le circuit mentionné n’a pas été fait car on a trouvé une autre solution plus simple, c’est d’utiliser une carte puissance (MDD10A) de Cytron Technologies (figure 2).

* + 1. **Moteurs**

Le bon choix des moteurs est un point important dans la réalisation d'un robot. En effet si les moteurs n'ont pas assez de couple, il sera très difficile pour le robot de se mouvoir à faible vitesse, de démarrer ou même de freiner efficacement. Si la vitesse de rotation est trop faible, le robot sera lent, si le moteur est sous-dimensionné il sera utilisé en surrégime, ce qui peut être bénin pour une utilisation de courte durée comme critique si l'utilisation est en fait trop violente et entraîner la fonte de la gaine du bobinage et donc la destruction du moteur.

Après étudier les caractéristiques du moteur de la marque suisse « Maxon Motors » qui est réputé pour sa robustesse, le groupe Méca a décidé de l’utiliser pour notre robot.



Figure : Encodeur



Figure : Moteur Maxon

* + 1. **Encodeur séparé**

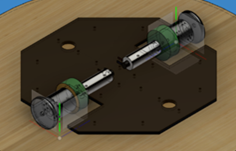
 Pour optimiser l’odométrie, nous avons décidé rapidement de séparer la motricité de l’encodage du mouvement comme la figure ci-dessous pour s’affranchir du compromis adhérence-précision (la roue motrice veut avoir une grande surface de contact tandis qu’une roue codeuse optimale souhaite un contact ponctuel).

Figure 6 : La structure de la base du robot

* 1. **Asservissement**
     1. **Localisation**

À part de la caméra, le positionnement peut se faire à partir des encodeurs. C’est le positionnement relatif, on met à jour la position du robot à chaque petit pas du temps, à partir de la position précédente du robot.

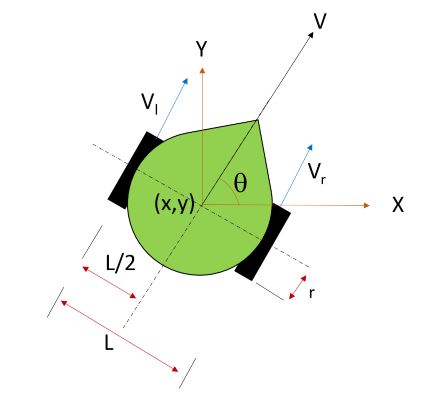
 Le modèle qu’on a utilisé, c’est le modèle du robot mobile à roues différentielles.

Figure 7 : modèle du robot

et

Il nous donc donne la position et l’orientation du robot par :

Comme est petit, et sont le déplacement de la roue droite et gauche respectivement ().

On a donc

= le déplacement par « tick » un écart de nombre de ticks en

L’implémentation du positionnement du robot n’est pas très compliquée, il suffit de mettre à jour les équations à chaque petit pas du temps.

* + 1. **Implémentation du PID**

Notre asservissement est un asservissement en position, en effet, nous ne cherchons pas à contrôler la vitesse de déplacement du robot mais uniquement sa position afin qu’il parcoure exactement la distance souhaitée en ligne droite et qu’il tourne exactement vers l’orientation souhaitée.

Le principe du PID (Proportionnel, intégrale, dérivée) est de comparer à la fois la sortie et l’entrée (reference) afin d’observer l’évolution de l’erreur et de le corriger avec la consigne qui est formée par trois composantes :

* Composante proportionnel : elle génère une valeur proportionnelle aux erreurs mesurées. Si l’erreur est importante, il faut aussi appliquer une correction importante afin de converger vers la valeur voulue. Mais quand on demande plus de rapidité (composante proportionnel est grande), le système est sensible à l’oscillation et il ne converge pas vers la valeur voulue.
* Composante dérivée : elle permet d’améliorer la stabilité du système en diminuant les oscillations. En effet, elle donne une valeur proportionnelle à l’évolution de l’erreur au cours du temps.
* Composante intégrale : elle permet de gagner en précision en empêchant la formation d’une erreur statique. Lorsqu’il n’y a pas de changement de l’erreur au cours du temps, les composantes proportionnelle et dérivée donnent uniquement zéro. Afin d’éviter ce phénomène, la somme des erreurs au cours du temps nous permet de pousser la convergence du système vers la valeur voulue.

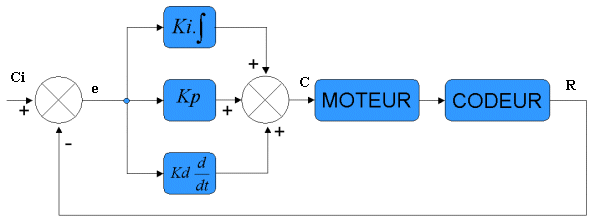


Figure : Schéma PID

Comme indiqué plus haut, le modèle du robot nous donne :

ou bien

et sont la vitesse angulaire du moteur droite et gauche.

On a donc

À partir de ces équations, les consignes envoyées au moteur droite et gauche sont déterminées par la vitesse de déplacement et la vitesse angulaire du robot.

* Pour faire tourner le robot à l’angle souhaité, la vitesse de déplacement du robot doit être zéro. Il ne nous reste que de gérer la vitesse angulaire.

C’est à partir d’ici, on va implémenter le PID. On va suivre ces étapes :

* + Mesurer l’orientation du robot
  + Mesurer l’erreur en angle par rapport à l’angle désiré
  + Former une correction de trois composantes

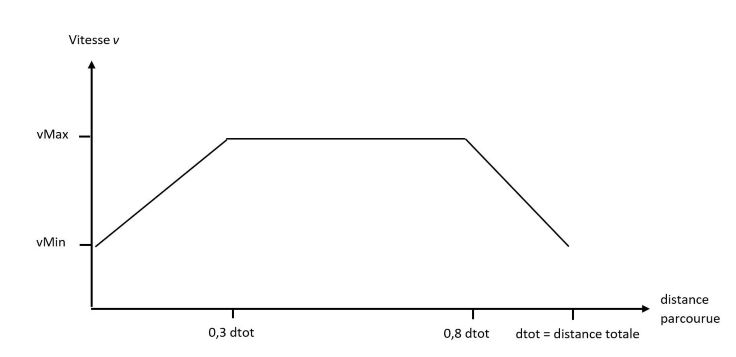
* Pour faire avancer le robot en ligne droite, l’orientation du robot doit être zéro ; est déterminée par le PID de l’angle souhaité zéro. Cependant, la vitesse de déplacement est déterminée de la manière indiquée dans la figure ci-dessous, elle augmente en rampe au début, descend en rampe à la fin et reste constant au milieu du trajet.

Figure 9 : la fonction de vitesse au cours du temps

**Réglage des constantes**

On va se servir la méthode de Ziegler-Nichols pour régler les constantes du PID. Afin d’obtenir , il nous faut d’abord cherche le gain maximal en mettant et en zéro et en réglant jusqu’à ce que le système s’oscille de manière stable. Ensuite, on repère la période d’oscillation . On a donc le réglage correspondant :