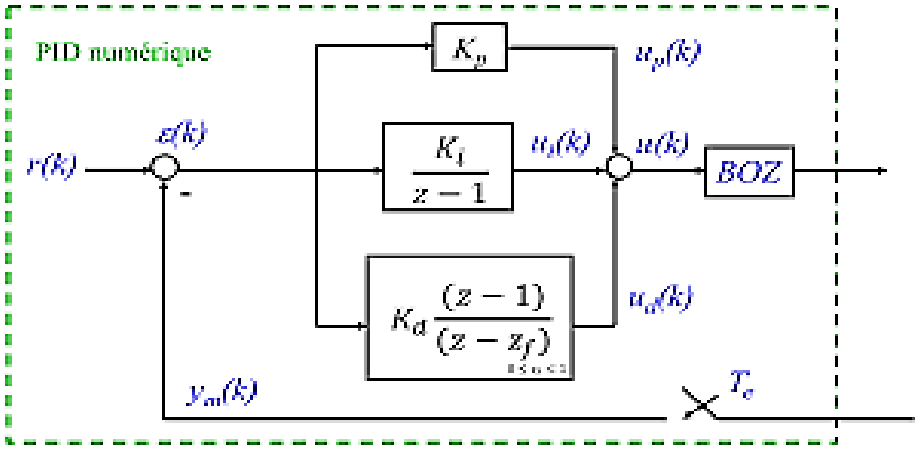
Implémentation du correcteur PID sur Arduino :

Nous verrons dans ce document comment implémenter le correcteur PID sur une carte Arduino.

Pour commencer nous détaillerons en premier lieu les différents éléments d'un correcteur PID.

Correcteur PID :

Nous verrons dans cette partie comment implémenter et régler efficacement les paramètres d'un correcteur PID numérique réel. Contrairement à un PID théorique, le PID numérique dispose d'un filtre passe-bas en HF afin de limiter les perturbations. En revanche, la bande passante du filtre est directement liée à l'efficacité de l'effet dérivateur. Autrement dit, la stabilité du système.



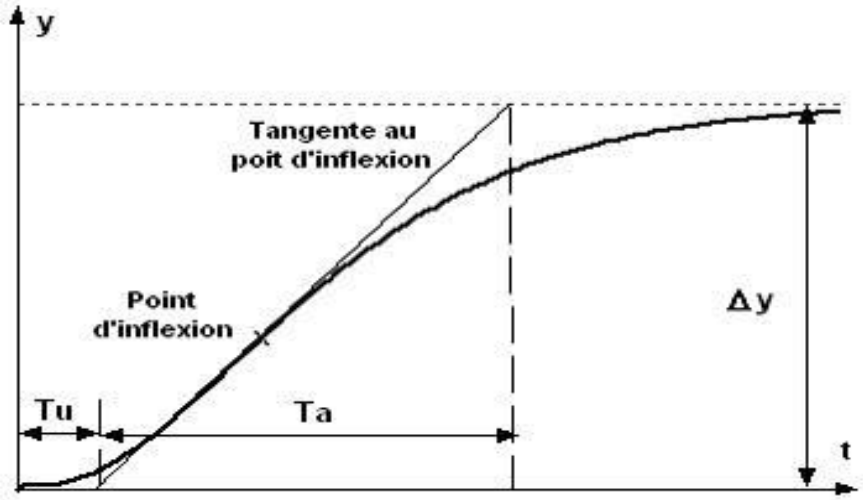
Comment régler les paramètres du correcteur PID ?

Il existe plusieurs techniques de réglage des paramètres d'un correcteur PID, en particulier la méthode de **Ziegler-nichols.**

Ziegler et Nichols ont proposé deux approches expérimentales destinées à fixer rapidement les paramètres des régulateurs P, PI et PID. **La première nécessite l'enregistrement de la réponse indicielle du système à régler**, alors que **la deuxième exige d'amener le système en boucle fermée à sa limite de stabilité.**

- **Méthode de Ziegler&Nichols en boucle ouverte**

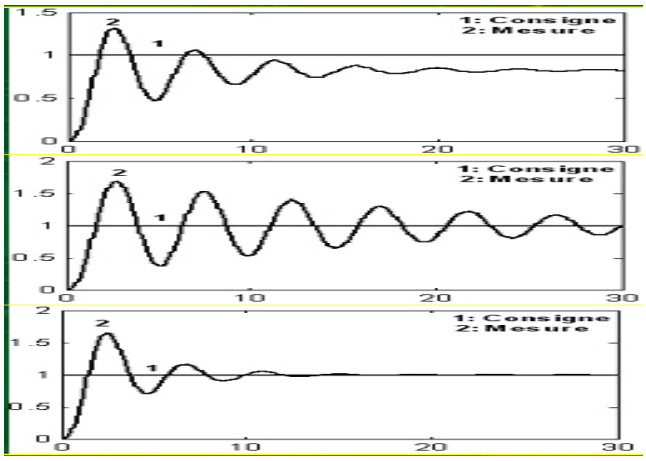
Sur l'enregistrement de la réponse indicielle, on trace le mieux possible la tangente au point d'inflexion Q de la courbe. On mesure ensuite le temps T_u correspondant au point d'intersection entre l'axe des abscisses et la tangente ainsi que le temps T_a « temps de montée de la tangente ».



Réglage du régulateur PID

Ziegler&Nichols proposent de calculer les paramètres du régulateur P, PI ou PID à l'aide des recommandations suivantes :

Réglage des paramètres			
Régulateur	K_p	T_i	T_d
P : $R(p) = K_p$		*	*
PI : $R(p) = Kp(1 + \frac{1}{T_i p})$	$\frac{Ta}{Tu}\frac{0.9}{K}$	3.33 Tu	*
PID : $R(p) = K_p(1 + T_d p + \frac{1}{T_i p})$	$\frac{Ta}{Tu}\frac{1.2}{K}$	2.0 Tu	0.5 Tu



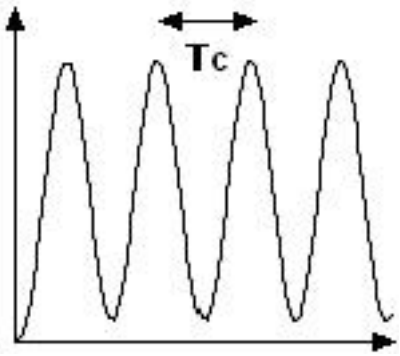
Régulation P

Régulation PI

Régulation PID

- **Méthode de Ziegler&Nichols en boucle fermée**

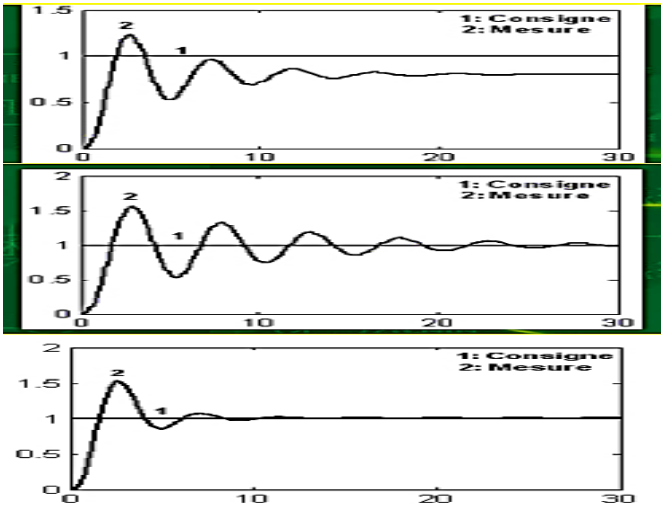
On relève le gain limite (Kpc) conduisant au pompage de la boucle et la période des oscillations T_c correspondant à ce fonctionnement à partir de n'importe quel point d'observation (sortie du régulateur, sortie du procédé..).



Réglage du régulateur PID

Ziegler&Nichols proposent de calculer les paramètres du régulateur P, PI ou PID à l'aide des recommandations suivantes :

Réglage des paramètres			
Régulateur	K_p	T_i	T_d
P : $R(p) = K_p$	0.5 Kpc	*	*
PI : $R(p) = Kp(1 + \frac{1}{T_i p})$	0.45 Kpc	0.83 Tc	*
PID : $R(p) = K_p(1 + T_d p + \frac{1}{T_i p})$	0.6 Kpc	0.5 Tc	0.125 Tc



Régulation P

Régulation PI

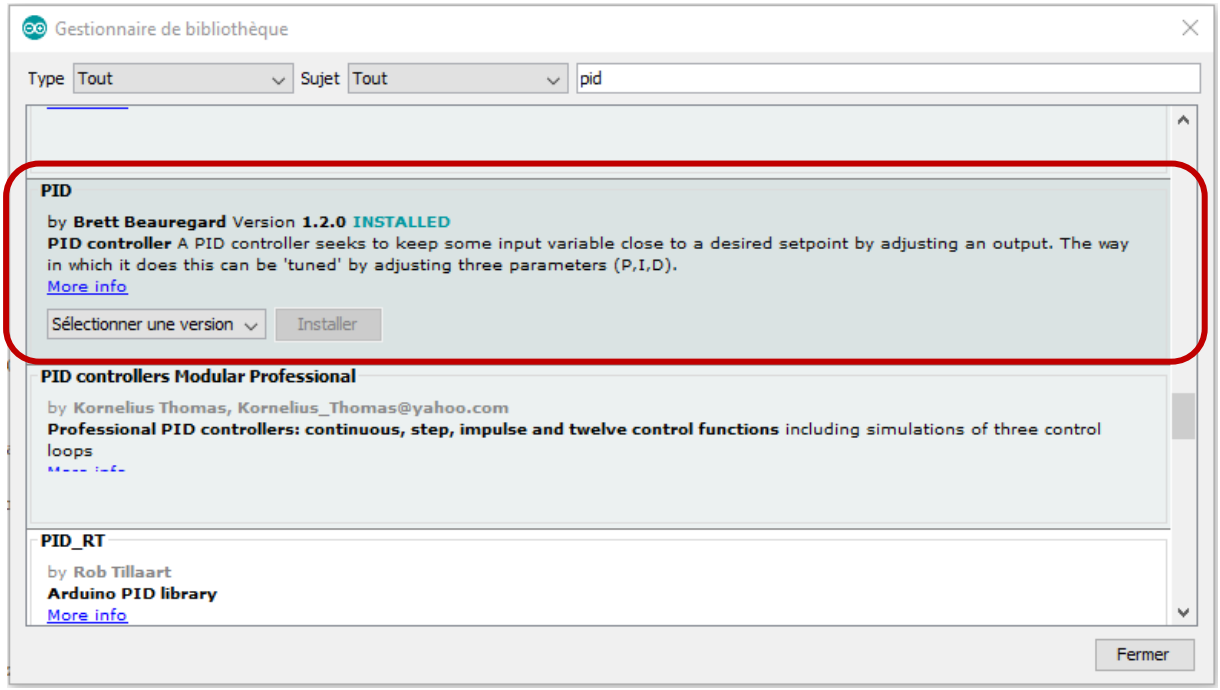
Régulation PID

Implémentation sur Arduino :

Dans cette partie vous aurez les détails de l'implémentation du correcteur PID sur une carte arduino.

Etape 1 :

Télécharger la bibliothèque PID :



Etape 2 :

Implémenter le code suivant, tout en lisant les différents commentaires explicatifs :

```
#include <PID_v1.h>

double ValeurD; //Valeur Désirée
double ValeurE; //Valeur Entrée
double ValeurS; //Valeur Sortie

//Parametres PID
double Kp=10, Ki=10, Kd=10; // les parametres sont en fonction du systèmes

//Création de l'instance PID
PID monPID(&ValeurE, &ValeurS, &ValeurD, Kp, Ki, Kd, DIRECT);
```

```
void setup()
{
  Serial.begin(9600);
  //Choix de la valeur désirée
  ValeurD = 80;
  //Allumer le PID
  monPID.SetMode(AUTOMATIC);
  //Ajustement des Valeurs du PID
  monPID.SetTunings(Kp, Ki, Kd);
}
```

```
void loop()
{
  /*Lire l'entré sur le pin 5 par exemple, Analog input : entre 0
  et 1024, on la passe à une valeur entre 0 et 255 vu qu'on
  l'utilise sur une fonction PWM.*/
  ValeurE = map(analogRead(5), 0, 1024, 0, 255);
  //Calcul PID
  monPID.Compute();
  //Ecrire la sortie sur le pin 3 par exemple
  analogWrite(3, ValeurS);
  //Envoie des valeurs par SerialPort pour la visualisation
  Serial.print(ValeurE);
  Serial.print(" ");
  Serial.println(ValeurS);
  Serial.print(" ");
  Serial.println(ValeurD);
  // Ajouter un delay si vous voulez
}
```

CODE :

```
#include <PID_v1.h>

double ValeurD; //Valeur Désirée
double ValeurE; //Valeur Entrée
double ValeurS; //Valeur Sortie
//Parametres PID
double Kp=10, Ki=10, Kd=10; // les parametres sont en fonction du systèmes
//Création de l'instance PID
PID monPID(&ValeurE, &ValeurS, &ValeurD, Kp, Ki, Kd, DIRECT);
void setup() {
  Serial.begin(9600);
  //Choix de la valeur désirée
  ValeurD = 80;
  //Allumer le PID
  monPID.SetMode(AUTOMATIC);
  //Ajustement des Valeurs du PID
  monPID.SetTunings(Kp, Ki, Kd);
}
void loop() {
  //Lire l'entré sur le pin 5 par exemple, Analog input : entre 0 et 1024, on la passe à une valeur entre 0 et 255 vu qu'on
l'utilise sur une fonction PWM.
  ValeurE = map(analogRead(5), 0, 1024, 0, 255);
  //Calcul PID
  monPID.Compute();
  //Ecrire la sortie sur le pin 3 par exemple
  analogWrite(3, ValeurS);
  //Envoie des valeurs par SerialPort pour la visualisation
  Serial.print(ValeurE);
  Serial.print(" ");
  Serial.println(ValeurS);
  Serial.print(" ");
  Serial.println(ValeurD);
  // Ajouter un delay si vous voulez
}
```