Sommaire

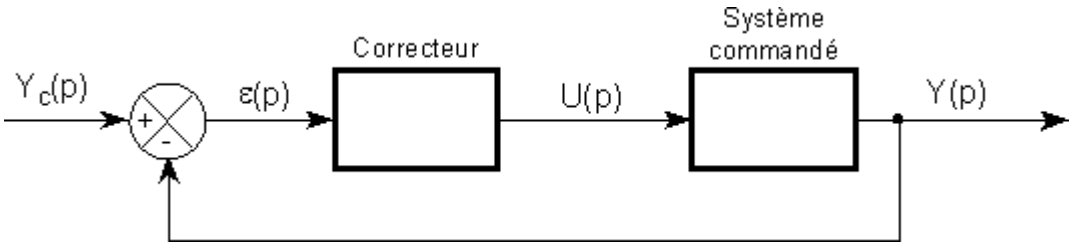
Objectifs

1. Implémentation d'un système du 2nd ordre en boucle fermée (BF)
2. Analyse d'un système en BF
3. La réponse indicielle/rampe en BF
4. Avantage et inconvénient d'un système en boucle fermée à retour unitaire sans correcteur La réponse à une rampe d'un système en 2nd ordre
5. Etc.

> *Voir le tuto d'avant pour plus des détails*

Fonctionnement

Le tuto a pour objectif d'introduire l'architecture d'asservissement avec Arduino. En particulier la boucle fermée avec le retour unitaire et sans correcteur. On va analyser la réponse indicielle et une rampe du système en BF en comparaison avec la boucle ouverte et en déduire les avantages et incontinents des deux stratégies de commande. L'architecture introduite dans ce tuto sera utilisé dans les tutos prochains avec les divers types des correcteurs numériques (P, PI, PD, PID, ...). Voir le tuto pour plus des détails. Ci-dessous le code Arduino commenté.



Les paramètres du système

```
#define   Fn      10.00

#define   Zeta    0.70710678118

#define   K       1.0

#define   T_ms    2
```

Les paramètres de l’échelon (ou rampe)

```
#define   A_step  10.0 // Amplitude

#define   c_step  200  // Période = 2*c_step*T_ms

unsigned long c=0; // Compteur (période)

bool Step=false;
```

Les variables en BO et BF

```
double x_nn=0.0;    // Consigne (entrée)

double y_n_BF=0.0;  // Sortie en BF

double y_n_BO=0.0;  // Sortie en BO

double eps_n=0.0;   // Erreur

double y_capt=0.0;  // Sortie du capteur

double y_corr=0.0;  // Sortie du correcteur
```

Les variables internes des systèmes

```
double x1[2], y1[3]; // Système en BF

double x2[2], y2[3]; // Système en BO
```

Les étapes de l’implémentation de la boucle fermée avec Arduino

La consigne (l’entrée) x(n)

```
c++; c=c%c_step;

if(!c)

{

  Step=!Step;

  c=0;

}

x_nn=A_step*(double)Step; // Réponse à un échelon x(n)=cte

//x_nn=(double)c;           // Réponse à une rampe x(n)=n
```

Sortie du capteur: Retour unitaire

```
y_capt=y_n_BF;
```

Soustracteur: Calcul de l’erreur eps(n)

```
eps_n=x_nn-y_capt;
```

Correcteur

```
y_corr=eps_n;
```

Calcul de la sortie en BF

```
y_n_BF=Sys2All(y_corr, x1, y1, Zeta, Wn, K, T_s);
```

Calcul de la sortie en BO

```
y_n_BO=Sys2All(x_nn, x2, y2, Zeta, Wn, K, T_s);
```

Affichage des signaux

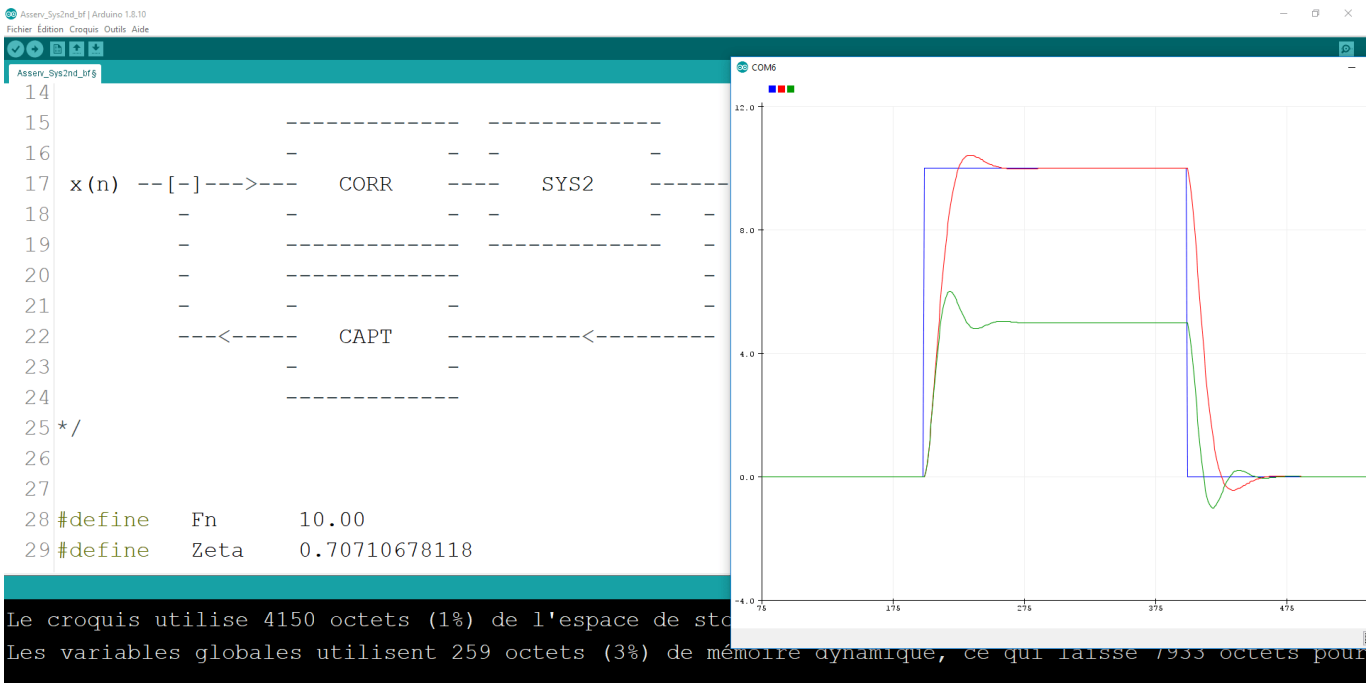
```
Serial.print(x_nn); Serial.print(",");

Serial.print(y_n_BO); Serial.print(",");

Serial.println(y_n_BF);
```

Implémentation d’un système du

second ordre

```
double Sys2All(double x_nn, double *x, double *y, double zeta, double wn, double k, double T)

{

  // Paramètre du système

  double a1=2.0*zeta/wn;

  double a2=1.0/(wn*wn);



  const double b0=(a1/(2.0*T))+(a2/(T*T));

  const double b1=-2.0*a2/(T*T);

  const double b2=(-1.0*a1/(2.0*T))+(a2/(T*T));

  const double b[3]={b0,b1,b2};



  // Variables de l'entrée et la sortie

  double y_nn=0.0;



  // Calcul de la nouvelle sortie

  y_nn= -(y[0]*(1.0+b[1]))-(y[1]*b[2])+(k*x[0]); // y[1]: y(n-2), y[0]: y(n-1)

  y_nn/=b[0];
```

```
  // Mise à jour de la sortie

  y[1]=y[0];

  y[0]=y_nn;



  // Mise à jour de la sortie

  x[0]=x_nn;



  // Renvoie du résultat

  return y_nn;

}
```



Le code complet

```
/*
 * 1. Implémentation d'un système du 2nd ordre en boucle fermée (BF)
 * 2. Analyse d'un système en BF
 * 3. La réponse indicielle/rampe en BF
 * 4. La réponse à une rampe d'un système en 2nd ordre
 * 5. Etc.
 *
                 -------------
                 -           -
 x(n) ------->---   SYS2     ------->-- y(n): BO
                 -           -
                 -------------
                 -------------  -------------
                 -           -  -           -
 x(n) --[-]--->---   CORR    ----   SYS2    ------->-- y(n): BF
         -       -           -  -           -   -
         -       -------------  -------------   -
         -       -------------                  -
         -       -           -                  -
         ---<-----   CAPT    ----------<---------
                 -           -
                 -------------
*/


#define   Fn      10.00
#define   Zeta    0.70710678118
#define   K       1.0
#define   T_ms    2

#define   A_step  10.0 // Amplitude
#define   c_step  200  // Période = 2*c_step*T_ms

double Wn=2.0*PI*Fn;
double T_s=(double)T_ms/1000.0;


double x_nn=0.0;    // Consigne (entrée)
double y_n_BF=0.0;  // Sortie en BF
```

```
double y_n_BO=0.0;  // Sortie en BO
double eps_n=0.0;   // Erreur
double y_capt=0.0;  // Sortie du capteur
double y_corr=0.0;  // Sortie du correcteur

// Variables internes des systèmes
double x1[2], y1[3]; // Système en BF
double x2[2], y2[3]; // Système en BO


// Paramètres de l'échelon
unsigned long c=0; // Compteur (période)
bool Step=false;

void setup()
{
  // Port série de la réponse du système
  Serial.begin(9600);
}

void loop()
{
  // 1. La consigne (l'entrée) x(n)
  c++; c=c%c_step;
  if(!c)
  {
    Step=!Step;
    c=0;
  }
  x_nn=A_step*(double)Step; // Réponse à un échelon x(n)=cte
  //x_nn=(double)c;           // Réponse à une rampe x(n)=n
  // 2. Sortie du capteur: Retour unitaire
  y_capt=y_n_BF;
  // 3. Soustracteur: Calcul de l'erreur eps(n)
  eps_n=x_nn-y_capt;

  // 4. Correcteur
  y_corr=eps_n;
  // 5. Calcul de la sortie en BF
  y_n_BF=Sys2All(y_corr, x1, y1, Zeta, Wn, K, T_s);

  // 6. Calcul de la sortie en BO
  y_n_BO=Sys2All(x_nn, x2, y2, Zeta, Wn, K, T_s);
  // Affichage des signaux
```

```
  Serial.print(x_nn); Serial.print(",");
  Serial.print(y_n_BO); Serial.print(",");
  Serial.println(y_n_BF);

  // Période d'échantillonnage
  delay(T_ms);
}



double Sys2All(double x_nn, double *x, double *y, double zeta, double wn, double k,
double T)
{
  // Paramètre du système
  double a1=2.0*zeta/wn;
  double a2=1.0/(wn*wn);

  const double b0=(a1/(2.0*T))+(a2/(T*T));
  const double b1=-2.0*a2/(T*T);
  const double b2=(-1.0*a1/(2.0*T))+(a2/(T*T));
  const double b[3]={b0,b1,b2};

  // Variables de l'entrée et la sortie
  double y_nn=0.0;
  // Calcul de la nouvelle sortie
  y_nn= -(y[0]*(1.0+b[1]))-(y[1]*b[2])+(k*x[0]); // y[1]: y(n-2), y[0]: y(n-1)
  y_nn/=b[0];
  // Mise à jour de la sortie
  y[1]=y[0];
  y[0]=y_nn;

  // Mise à jour de la sortie
  x[0]=x_nn;
  // Renvoie du résultat
  return y_nn;
}
```

Accueil Asservissement avec Arduino