Boucle à verrouillage de phase

1. Introduction

La boucle à verrouillage de phase est un dispositif permettant de produire une oscillation synchrone avec un signal périodique. Après avoir étudié son principe de fonctionnement, nous réaliserons une boucle à verrouillage de phase. En seconde partie, on abordera une application, la démodulation d'un signal modulé en fréquence.

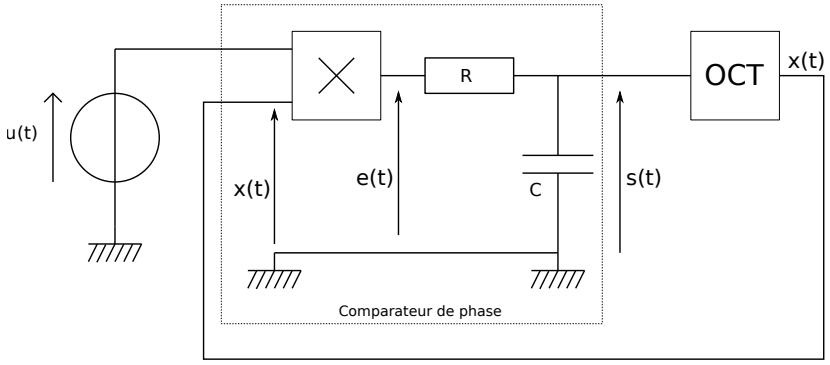
Matériel :

- ▷ Générateur de fonctions SIGLENT SDG1025.
- ▷ Multiplieur analogique (AD633).
- ▷ Carte Arduino MEGA.
- ▷ Résistances $2{,}2\,\mathrm{k\Omega}$ et $10\,\mathrm{k\Omega}$.
- ▷ Condensateurs $1\,\mathrm{nF}$ et $1\,\mu\mathrm{F}$.

2. Boucle à verrouillage de phase

2.a. Fonctionnement

La figure suivante montre le schéma d'une boucle à verrouillage de phase :



Le signal à analyser est la tension $u(t)$. La boucle est constituée d'un *comparateur de phase* et d'un *oscillateur commandé en tension* (OCT).

L'OCT délivre une tension sinusoïdale $x(t)$ dont la fréquence est commandée par la tension d'entrée $s(t)$.

Le comparateur de phase effectue la comparaison des signaux $u(t)$ et $x(t)$. Il est constitué d'un multiplieur analogique et d'un filtre RC passe-bas. Le multiplieur délivre le signal :

$$e(t) = u(t)x(t) \tag{1}$$

Supposons que le signal $u(t)$ soit sinusoïdal, de pulsation ω :

$$u(t) = U\cos(\omega t) \tag{2}$$

Nous faisons une étude simplifiée de la boucle à verrouillage de phase, dans laquelle nous supposons qu'elle est verrouillée (voir l'annexe pour un traitement plus général). On admettra que le signal en sortie de l'OCT est alors synchrone avec $u(t)$, c'est-à-dire de même fréquence mais déphasé :

$$x(t) = X\cos(\omega t + \phi) \tag{3}$$

Le signal en sortie du multiplieur a alors la forme suivante :

$$e(t) = \frac{1}{2}UX\left(\cos\phi + \cos(2\omega t + \phi)\right) \tag{4}$$

Le filtre RC passe-bas est conçu pour atténuer suffisamment la composante de fréquence 2ω. Il reste donc en sortie du filtre (approximativement) :

$$s(t) = \frac{1}{2}UX\cos\phi \tag{5}$$

Cette tension est constante lorsque la fréquence de $u(t)$ est constante et lorsque la boucle est verrouillée, mais en général elle dépend du temps. La tension $s(t)$ commande la pulsation de l'OCT en la faisant varier autour d'une pulsation centrale ω_o selon la relation :

$$\omega' = \omega_o + as(t) \tag{6}$$

où a est une constante positive s'exprimant en Hz par V.

Lorsque la boucle est verrouillée, cette pulsation est identique à celle de $u(t)$, c'est-à-dire $\omega' = \omega$. Si la fréquence de $u(t)$ change légèrement, la boucle s'adapte automatiquement pour maintenir une fréquence en sortie de l'OCT égale à la fréquence de $u(t)$. Il s'en suit que le signal $s(t)$ qui commande l'OCT reproduit les variations de fréquence de $u(t)$. Pour que la boucle reste toujours verrouillée, il faut cependant que la pulsation de $u(t)$ reste proche de ω_0.

2.b. Réalisation

Le signal sinusoïdal $u(t)$ est généré sur la voie CH1 du générateur de fonction SIGLENT. Le signal sinusoïdal $x(t)$ est généré sur la voie CH2. Pour réaliser la commande de fréquence, activer la fonction Mod (modulation) sur CH2 puis sélectionner FM mod (modulation de fréquence) et une source de modulation externe. La tension de commande $s(t)$ doit être envoyée sur l'entrée *Modulation In* située à l'arrière de l'appareil.

Pour commencer, on prendra $f = 5000\,\text{Hz}$ pour la fréquence de $u(t)$ et $f_o = 5000\,\text{Hz}$ pour la fréquence centrale de l'OCT.

Le filtre RC passe-bas est réalisé avec la résistance $R = 10\,\text{k}\Omega$ et le condensateur de capacité $C = 1\,\mu\text{F}$.

[1] Calculer la fréquence de coupure de ce filtre.

[2] Réaliser la boucle à verrouillage de phase. Visualiser sur l'oscilloscope les tensions $u(t)$ et $x(t)$.

[3] Modifier légèrement la fréquence f. Constater que $x(t)$ reste synchrone avec $s(t)$.

[4] Observer la tension $s(t)$ sur l'oscilloscope. Comment cette tension évolue-t-elle lorsque la fréquence f varie ? La valeur de cette tension est-elle en accord avec le déphasage observé entre $x(t)$ et $u(t)$.

[5] Constater que si l'on impose une variation de f trop rapide et trop grande, la boucle perd le verrouillage.

[6] Considérer le cas d'un signal $u(t)$ de forme carrée. La boucle fonctionne-t-elle toujours ? Expliquer à l'aide de la série de Fourier.

[7] Délivrer un signal $u(t)$ sinusoïdal mais comportant un bruit important. Quel est alors l'intérêt du signal $x(t)$?

2.c. Annexe : verrouillage de la boucle

On présente dans cette annexe une théorie simplifiée du verrouillage de la boucle, dans l'hypothèse où elle reste relativement proche du verrouillage. On suppose que le déphasage de $x(t)$ par rapport à $u(t)$ reste proche de $\pi/2$, ce qui permet d'écrire :

$$x(t) = X\cos\left(\omega t + \frac{\pi}{2} + \epsilon(t)\right) = -X\sin\left(\omega t + \epsilon(t)\right) \tag{7}$$

Lorsque la boucle est verrouillée, ϵ est constant mais très petit. Si un changement de la pulsation ω survient, la boucle n'est plus verrouillée et $\epsilon(t)$ évolue. La tension en sortie du multiplieur est :

$$e(t) = u(t)x(t) = -\frac{UX}{2}\left(\sin(2\omega t + \epsilon(t)) + \sin(\epsilon(t))\right) \tag{8}$$

Dans l'hypothèse où les variations de $\epsilon(t)$ sont assez lentes pour franchir le filtre passe-bas, on a en sortie du filtre :

$$s(t) = -\frac{UX}{2}\sin(\epsilon(t)) \tag{9}$$

Ce signal est utilisé pour définir la pulsation ω' du signal $x(t)$. Or la pulsation est la dérivée par rapport au temps de la phase, soit :

$$\omega' = \frac{d}{dt}\left(\omega t + \frac{\pi}{2} + \epsilon(t)\right) = \omega + \frac{d\epsilon(t)}{dt} \tag{10}$$

Si l'on suppose que la pulsation en sortie de l'OCT s'ajuste instantanément selon la relation (6), on a l'équation différentielle suivante :

$$\omega + \frac{d\epsilon(t)}{dt} = \omega_0 - a\frac{UX}{2}\sin(\epsilon(t)) \tag{11}$$

qui s'écrit, si ϵ reste très petit :

$$\frac{d\epsilon(t)}{dt} + \frac{aUX}{2}\epsilon(t) = \omega_0 - \omega \tag{12}$$

Cette équation différentielle montre que, si le coefficient a est positif, $\epsilon(t)$ tend vers une valeur constante avec un temps caractéristique $\tau = \frac{2}{aUX}$. Lorsque ϵ est constant on a $\omega' = \omega$, c'est-à-dire que la boucle est à nouveau verrouillée. On a alors :

$$s = -\frac{UX}{2}\epsilon = \frac{\omega - \omega_0}{a} \tag{13}$$

La valeur de s fournit donc le décalage entre la fréquence du signal analysé $u(t)$ et la fréquence centrale de l'OCT.

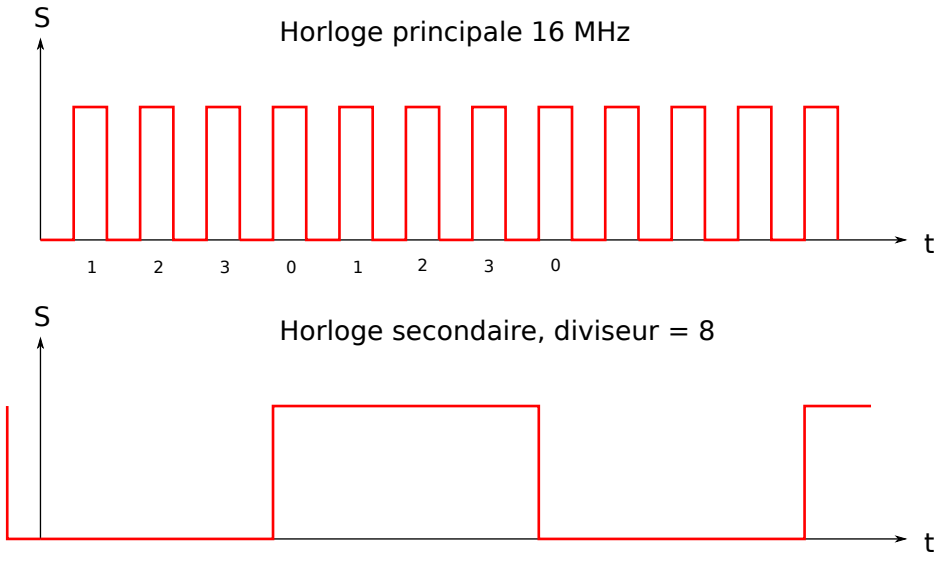
3. Modulation de fréquence

3.a. Génération d'un signal binaire

Un signal binaire ne contient que deux valeurs de tension, par exemple 0 et 5 V pour un signal TTL. La génération d'un signal binaire périodique se fait au moyen de compteurs.

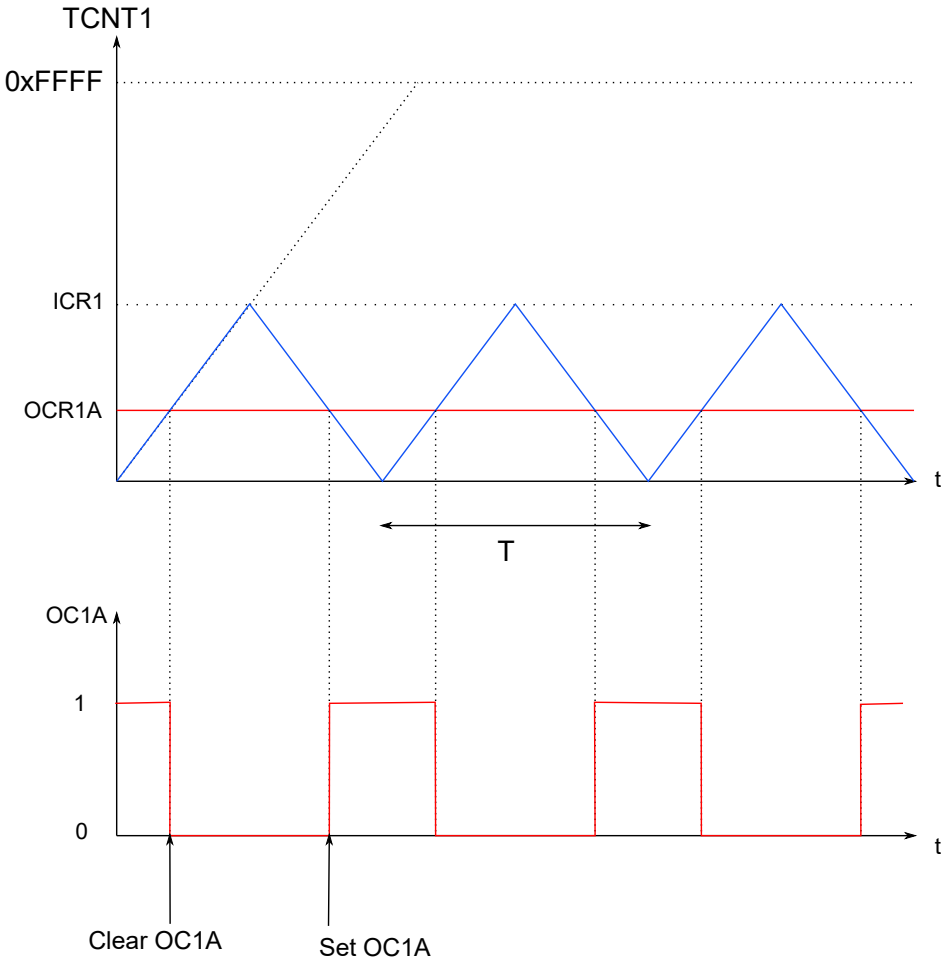
L'élément central de la carte Arduino MEGA est un microcontrôleur (ATmega 2560), c'est-à-dire un dispositif qui intègre un microprocesseur capable d'exécuter une suite d'instructions programmées et différents phériphériques, en particulier des compteurs permettant de générer des signaux binaires.

La carte Arduino MEGA possède une horloge à quartz de fréquence 16 MHz (horloge primaire), qui délivre un signal binaire à cette fréquence. L'horloge à quartz sert à cadencer l'exécution des instructions et toutes les opérations d'entrée-sortie, en particulier la génération de signaux périodiques. Cette fréquence étant trop grande pour certains usages, chaque périphérique comporte un diviseur d'horloge, qui génère un signal d'horloge secondaire, dont la fréquence est 16 MHz divisée 8, 64, 256 ou 1024. La division de fréquence se fait au moyen d'un compteur 10 bits, qui est incrémenté à chaque front montant du signal d'horloge (à la fréquence de 16 MHz). Si la division par 8 est choisie, le compteur s'incrémente de la valeur 0 jusqu'à la valeur 3 (11 en binaire), puis est réinitialisé à 0. À chaque fois que la valeur revient à 0, la sortie de l'horloge secondaire change d'état logique.



Afin de générer des signaux binaires de fréquence quelconque et de rapport cyclique variable, le microcontrôleur possède des périphériques appelés *Timers*. L'ATmega 2560 possède 4 timers 16 bits. On s'intéresse au Timer 1. Celui-comporte un compteur 16 bits, incrémenté à chaque front montant de l'horloge secondaire. La valeur du compteur est accessible par un registre nommé TCNT1. Elle peut aller de 0 à 0xFFFF (notation hexadécimale). On s'intéresse à un mode particulier de fonctionnement du compteur, dans lequel sa valeur est incrémentée jusqu'à une valeur maximale, puis décrémentée jusqu'à zéro. Le registre ICR1 contient la valeur maximale. La période T des cycles du compteur est égale à la période de l'horloge secondaire mutlipliée par deux fois ICR1. Le registre OCR1A contient une valeur qui est comparée à la valeur du registre après chaque front montant de l'horloge secondaire. Cette comparaison

permet de faire la bascule du registre binaire OC1A (1 bits), lequel sert à générer la tension de sortie binaire. Lorsque ICR1 passe au-dessus de OCR1A, OC1A passe à 0. Lorsque ICR1 passe en dessous de OCR1A, OC1A passe à 1. La valeur de OCR1A permet donc de contrôler le rapport cyclique du signal généré.



Le programme Arduino suivant permet de générer un signal binaire de fréquence $5000\,\mathrm{Hz}$ sur la sortie 11 de la carte Arduino MEGA.

generationSignal.ino

```
#include "Arduino.h"
uint16_t icr_0, ocra_0;

void setup() {
DDRB |= 1 << PORTB5; // borne 11 configurée en sortie
/* configuration du Timer 1 avec une horloge à 2 MHz (diviseur 8) */
uint16_t diviseur[6] = {0,1,8,64,256,1024};
uint8_t d=2; // diviseur 8
cli(); // désactivation des interruptions
TCCR1A = 0;
TCCR1A |= (1 << COM1A1) ;// clear OC1A on compare match when up-counting, reset OC1A on
TCCR1B = 1 << WGM13; // phase and frequency correct pwm mode, top = ICR1
```

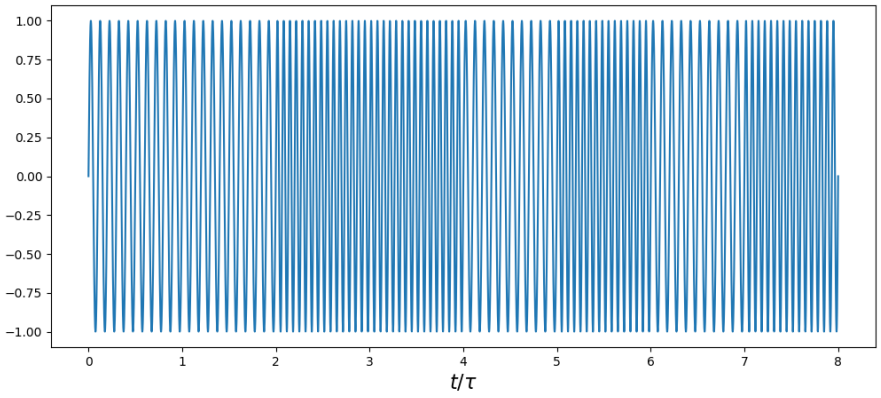
```
uint32_t period = 200; // période en microsecondes
icr_0 = (F_CPU/1000000*period/2/diviseur[d]); // valeur maximale du compteur, pour défir
ocra_0 = icr_0/2; // rapport cyclique = 1/2
ICR1 = icr_0;
OCR1A = ocra_0;
TCCR1B |= d; // déclenchement du compteur avec choix du diviseur d'horloge
sei();// activation des interruptions
}

void loop() {


}
```

[8] Télécharger ce fichier et l'enregistrer dans un dossier nommé `generationSignal`. Brancher la carte Arduino sur un port USB. Ouvrir le fichier avec l'IDE Arduino. Dans le menu `Outils`, sélectionner le type de carte (Arduino Mega) puis le port COM sur lequel la carte est branchée. Téléverser le programme sur la carte.

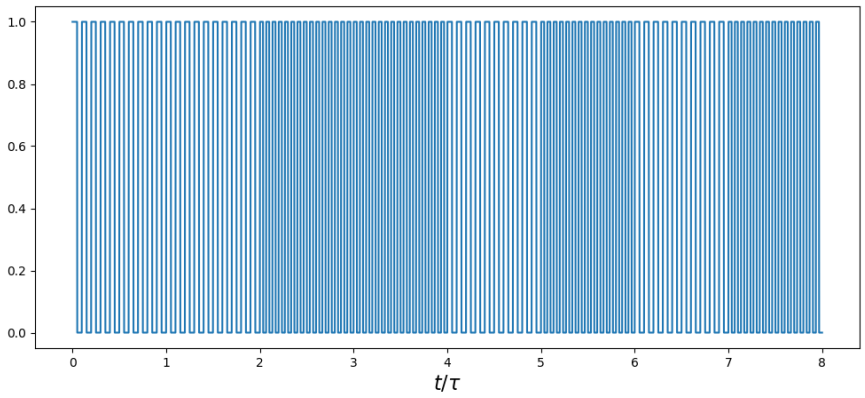
[9] Observer la sortie 11 de la carte à l'oscilloscope et vérifier la fréquence, qu'on pourra modifier en changeant la période.

3.b. Modulation de fréquence FSK

La modulation de fréquence FSK (Frequency Shift Key) est une technique de transmission numérique consistant à donner deux fréquences légèrement différentes à la porteuse (f_0 et f_1). Chacune de ces deux fréquences est appliquée pendant une durée fixée, notée τ. L'information transmise est un signal numérique, c'est-à-dire une suite de 0 et de 1. La fréquence f_0 représente un 0, la fréquence f_1 représente un 1. Par exemple, pour transmettre le nombre 53, dont la représentation en binaire sur 8 bits est 00110101, on module la fréquence comme représenté sur la figure suivante (pour une porteuse sinusoïdale) :



Dans le cas présent, où la porteuse est de forme carrée, la modulation FSK a l'aspect suivant :



Le programme Arduino suivant permet de faire une modulation FSK élémentaire (alternance de 0 et de 1).

generationFSK.ino

```
#include "Arduino.h"
uint16_t icr_0, ocra_0;
uint16_t icr_1, ocra_1;


void setup() {
DDRB |= 1 << PORTB5; // borne 11 configurée en sortie
/* configuration du Timer 1 avec une horloge à 2 MHz (diviseur 8) */
uint16_t diviseur[6] = {0,1,8,64,256,1024};
uint8_t d=2; // diviseur 8
cli(); // désactivation des interruptions
TCCR1A = 0;
TCCR1A |= (1 << COM1A1) ;// clear OC1A on compare match when up-counting, reset OC1A on
TCCR1B = 1 << WGM13; // phase and frequency correct pwm mode, top = ICR1
uint32_t period = 200; // période en microsecondes
icr_0 = (F_CPU/1000000*period/2/diviseur[d]); // valeur maximale du compteur pour défini
ocra_0 = icr_0/2; // rapport cyclique = 1/2
ICR1 = icr_0;
OCR1A = ocra_0;
TCCR1B |= d; // déclenchement du compteur avec choix du diviseur d'horloge
sei();// activation des interruptions
icr_1 = icr_0*0.9; // seconde période 10% plus petite (180 microsecondes)
ocra_1 = icr_1/2;

}

void loop() {
  // changement périodique de la fréquence
  delay(100); // attente en millisecondes
  ICR1 = icr_0; // première période (200 microsecondes)
  OCR1A = ocra_0;
  delay(100);
  ICR1 = icr_1; // seconde période (180 microsecondes)
  OCR1A = ocra_1;

}
```

La première période ($200\,\mu$s) est appliquée pendant $100\,$ms, la seconde période ($180\,\mu$s) est appliquée pendant $100\,$ms puis on recommence.

[10] Téleverser ce programme sur l'Arduino MEGA et observer le signal généré à l'oscilloscope.

[11] Afin de réduire la pente des fronts du signal, on place un filtre RC passe-bas, avec $2{,}2\,\mathrm{k}\Omega$ et $C = 1\,$nF. Réaliser le filtre et observer le signal en sortie.

Le signal ainsi obtenu est noté $u(t)$. La boucle à verrouillage de phase peut être utilisée pour faire la démodulation.

[12] Réaliser le branchement de la boucle à verrouillage de phase. Observer à l'oscilloscope les signaux $u(t)$ et $x(t)$ afin de vérifier que la boucle suit bien les variations de fréquence de la modulation FSK. Si la boucle perd le verrouillage, il faut réduire la différence entre les deux fréquences f_0 et f_1.

[13] La fréquence de la porteuse est obtenue par le signal $s(t)$ (tension de commande de l'OCT). Observer ce signal. Quel est le temps de réponse de la boucle à verrouillage de phase ?

[14] La fréquence f_0 étant maintenue à $5000\,$Hz, augmenter la fréquence f_1 jusqu'à la valeur maximale permettant de maintenir le verrouillage de la boucle. Quel est l'intérêt d'augmenter cette fréquence ?

[15] Jusqu'à quelle valeur peut-on abaisser la durée τ ? Quel est le débit d'information (en bits par seconde) ? Que faudrait-il modifier dans le dispositif pour diminuer encore cette durée et ainsi augmenter le débit d'information.

[16] Modifier le programme Arduino afin qu'il génère la transmission d'un nombre entier de 8 bits.