Sommaire

Objectifs

1. Comprendre la forme d'onde des signaux de la commande décalée
2. Savoir l'importance de la commande décalée
3. Savoir générer une comme symétrique et décalée avec Arduino
4. Savoir contrôler le module BTS7960 Avec Arduino
5. Voir la vidéo pour plus de détails

Projet Annexe

> *Voir la première partie ICI pour en savoir plus sur la commande symétrique et les caractéristiques du pont H complet BTS7960*

Importance de la commande décalée

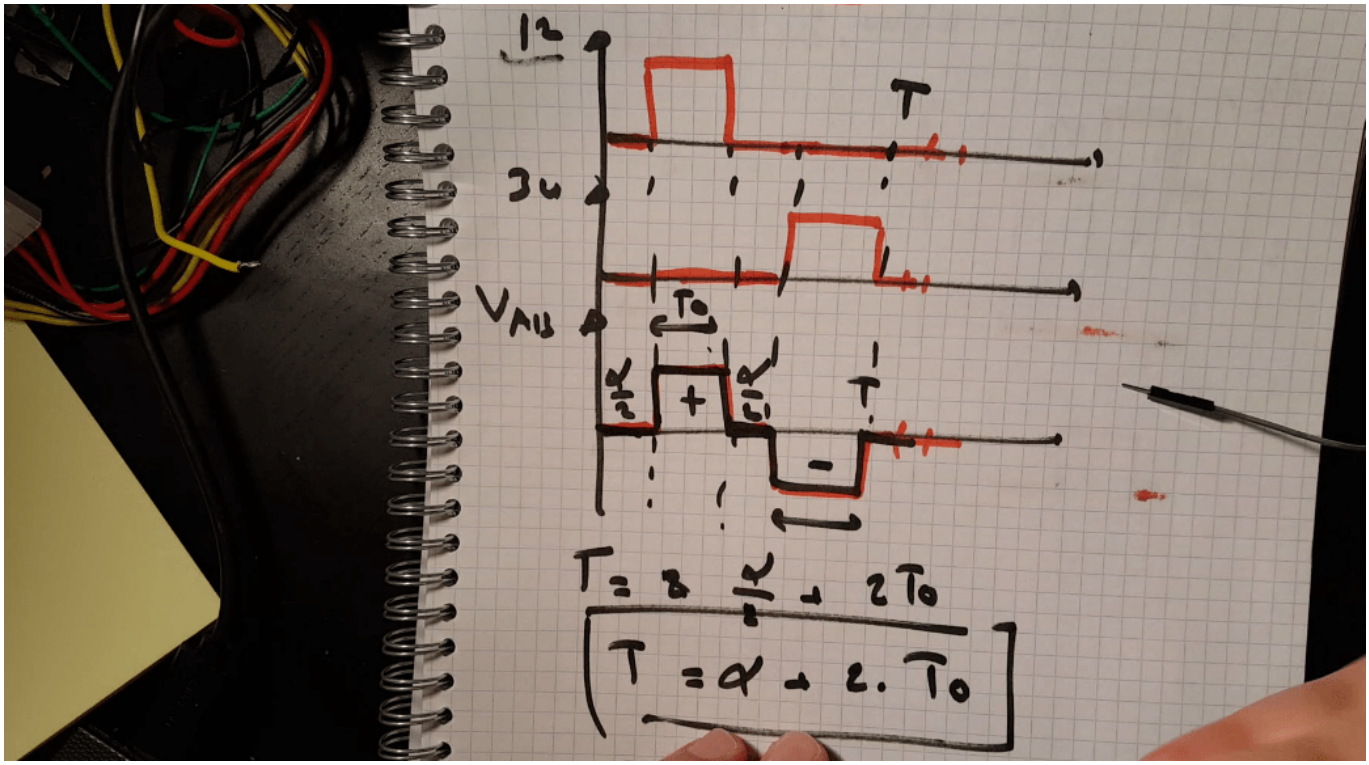
La commande symétrique alterne entre la fermeture de deux transistors à la fois en utilisant le pont H complet (sens 1 et sens 2) ou bien un demi-point à base de deux transistors MOS. Durant la commutation du sens 1 vers le sens 2 ou vis versa (la transition 0 ó1 des commandes), les transistors se trouvent court-circuités par la batterie. Ce phénomène réduit la durée de vie des transistors, ainsi celle de la batterie et induit le réchauffement des transistors. En particulier, si la fréquence de commutation est importante. La commande symétrique n'est pas recommandée en pratique. Sans oublie les pertes dues aux harmoniques. On verra dans la suite du projet l'analyse spectrale pour les diverses stratégies de la commande.

La commande décalée permet d'introduite un temps mort durant l'alternance positive et négative de la commande. Afin d'éviter le court-circuit des transistors. De plus la l'angle du temps mort permet de faire varier la valeur efficace au borne de la charge. Autrement dit, un variateur de tension alternative de 0 à Vbat. Lorsque le temps mort est nul, la commande décalée sera équivalent à la commande symétrique. La tension est nulle à la borne de la charge lorsque l'angle est égale 360°.

Commande décalée

> *Ci-dessous les chronogrammes d'une commande décalée (voir la vidéo pour plus de détails). Ci-dessous la formule de la valeur efficace en fonction de l'angle mort.*

$$V_{eff}=Vbat \cdot \sqrt{1- \alpha/T}$$



Implémentation de la commande symétrique avec Arduino

La commande d'un onduleur monophasé en pont H complet nécessite deux signaux carrés complémentaires C1 et C2 avec une fréquence identique. On peut générer le signal C1 puis en déduire C2 avec C2=not(C1). La fréquence du signal carrée est celui de l'onduleur (voir la vidéo pour plus des détails).

Le programme permet de générer deux sorties logiques de commandes du pont H dans les pins 5 et 6 de la carte Arduino. La période en milliseconde est définit par le paramètre T_ms dans le programme (inverse de la fréquence). Le paramètre 1/T_ms est la fréquence de l'onduleur. Elle doit être supérieure ou égale 4 ms.

```
#define T_ms    20     // 20 ms => 50 Hz
```

> *La variable Type dans le programme permet de définir le type de la commande:*
>
> - Type=0 : Commande symétrique
> - Type=1 : Commande décalée (voir la suite)

Implémentation de la commande décalée avec Arduino

> *Contrairement à la commande symétrique, la commande décalée dispose de 4 phases de fonctionnement*

- Phase 1 : Mise en arrêt du pont H pendant alfa/2 (C1=0 & C2=0)
- Phase 2 : Activation du sens 1 pendant T0 (durée de l'alternance positive) (C1=1 & C2=0)
- Phase 3 : Mise en arrêt du pont H pendant alfa/2 (C1=0 & C2=0)
- Phase 4 : Activation du sens 2 pendant T0 (durée de l'alternance négative) (C1=0 & C2=1)

La fonction HbridgeCrl()

> *void HbridgeCrl(byte PontHcmd)*

La fonction HbridgeCrl() permet de positionner les sorties logiques 5 et 6 (commande du pont H) de la carte Arduino en fonction du paramètre PontHcmd

- PontHcmd =0 : Pont H en Arrêt (C1=0 & C2=0)
- PontHcmd =1 : Pont H dans le sens 1(C1=1 & C2=0)
- PontHcmd =2 : Pont H dans le sens 2 (C1=0 & C2=1)
- PontHcmd =Autres : Pont H en Arrêt (C1=0 & C2=0)

```
void HbridgeCrl(byte PontHcmd)
{
switch(PontHcmd)
{
// Désactivation des sorties
case 0:
digitalWrite(R_PWM,LOW);
digitalWrite(L_PWM, LOW);
break;

// Sens 1
case 1:
digitalWrite(R_PWM, HIGH);
digitalWrite(L_PWM, LOW);
break;

// Sens 2
case 2:
digitalWrite(R_PWM, LOW);
digitalWrite(L_PWM, HIGH);
break;

default :
digitalWrite(R_PWM, LOW);
digitalWrite(L_PWM, LOW);
}
}
```

La fonction OnduleurCmd()

> *void OnduleurCmd(byte type, unsigned int t_ms, unsigned int t_0, unsigned int t_alf)*

La fonction OnduleurCmd() est la fonction principale qui permet de générer la commande symétrique ou décalée en fonction des paramètres d'entrées

- type : Commande symétrique ou décalée
- t_ms : Période en milliseconde de l'onduleur
- t_0 : durée de l'alternance positive/négative (voir la vidéo pour plus dé détails)
- t_alf : durée de l'angle alfa (en ms)

```
void OnduleurCmd(byte type, unsigned int t_ms, unsigned int t_0, unsigned int t_alf)
{
// Commande symétrique
if(type==0)
{
// Sens 1
HbridgeCrl(1);
delay(t_ms>>1);

// Sens 2
HbridgeCrl(2);
delay(t_ms>>1);
}

// Commande décalée
else
{
// Arret
HbridgeCrl(0);
delay(t_alf>>1);

// Sens 1
```

```
HbridgeCrl(1);
delay(t_0);

// Arret
HbridgeCrl(0);
delay(t_alf>>1);

// Sens 2
HbridgeCrl(2);
delay(t_0);
}
}
```

Le Programme Complet

```
#define R_PWM   5       // Commande 1 du pont H
#define L_PWM   6       // Commande 2 du pont H

#define T_ms    20      // Période de l'onduleur inverse de la fréquence en ms
// Supérieure à 4ms > 4 ms
#define Alfa    3*PI/2  // Angle Alfa (commande symétrique) [0, 2*PI]




unsigned long T_alfa;
unsigned long T_0;
byte Type=0;



void setup()
{
// Init sortie
pinMode(R_PWM, OUTPUT);
pinMode(L_PWM, OUTPUT);

// Calcul des périodes
T_alfa=round((double)T_ms*Alfa/(2.0*PI));
```

```
T_0=(T_ms-T_alfa)>>1;

// Désactivation des sorties
HbridgeCrl(0);

// Port série (Affichage des périodes si besoin)
Serial.begin(9600);
}

void loop()
{

// Commande Symétrique ou Décalée
Type=1;
OnduleurCmd(Type, T_ms, T_0, T_alfa);


// Affichage des périodes
/*
Serial.print(T_ms); Serial.print("\t");
Serial.print(T_alfa); Serial.print("\t");
Serial.println(T_0);
delay(1000);
*/
}



void HbridgeCrl(byte PontHcmd)
{
switch(PontHcmd)
{
// Désactivation des sorties
case 0:
digitalWrite(R_PWM,LOW);
digitalWrite(L_PWM, LOW);
break;

// Sens 1
case 1:
digitalWrite(R_PWM, HIGH);
digitalWrite(L_PWM, LOW);
break;
```

```
// Sens 2
case 2:
digitalWrite(R_PWM, LOW);
digitalWrite(L_PWM, HIGH);
break;

default :
digitalWrite(R_PWM, LOW);
digitalWrite(L_PWM, LOW);
}
}


void OnduleurCmd(byte type, unsigned int t_ms, unsigned int t_0, unsigned int t_alf)
{
// Commande symétrique
if(type==0)
{
// Sens 1
HbridgeCrl(1);
delay(t_ms>>1);

// Sens 2
HbridgeCrl(2);
delay(t_ms>>1);
}

// Commande décalée
else
{
// Arret
HbridgeCrl(0);
delay(t_alf>>1);

// Sens 1
HbridgeCrl(1);
delay(t_0);

// Arret
HbridgeCrl(0);
delay(t_alf>>1);

// Sens 2
HbridgeCrl(2);
```

```
delay(t_0);
}
}
```

Nous avons utilisé la fonction delay() pour la synthèse des périodes pour des raisons de simplification et pour faciliter la compréhension des commandes. D'où la limitation en 4ms (250 Hz) de la période pour la commande décalée. Concernant la commande symétrique, la période peut être égale à 2ms (500 Hz max).

Vous pouvez utiliser la fonction delayMicroseconds(T_us) pour augmenter largement la fréquence à quelques dizaines de kHz. Attention, la fréquence du pont H BTS7960 est limitée à 25KHz ! En cas de dépassement de la fréquence, vous engendrez un court-circuit de la batterie (ou alimentation du pont H) et détérioration du pont H voir la carte Arduino.

> *On verra dans la suite du projet comment la commande sPWM d'un onduleur et des techniques pour perfectionner la commande.*

Tout les Projets Arduino

Click to rate this post!
[Total: 1 Average: 5]