



République Tunisienne  
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique  
Université de Tunis El Manar  
École Nationale d'Ingénieurs de Tunis



## Département Génie Électrique

### Module Réseaux Electriques

# GÉNÉRATION PWM VARIABLE

**Réalisé par :**

Wael Guesmi  
Maha Romdhani

**Encadré par :**

Mr. Khaled Jlassi

**Classe :**

2<sup>ème</sup> Année GE3

**Année universitaire 2025/2026**

# Introduction

L'évolution rapide des systèmes embarqués et de l'électronique de puissance rend indispensable la maîtrise des techniques de commande numérique, notamment la **Modulation de Largeur d'Impulsion (PWM)**. Cette technique permet de contrôler efficacement la puissance moyenne délivrée à une charge, avec flexibilité et rendement élevé, et est largement utilisée dans la commande de moteurs, la variation de vitesse, l'éclairage à intensité variable et les systèmes embarqués.

Les microcontrôleurs de la famille **STM32**, et en particulier le **STM32F407**, se distinguent par leurs performances, la richesse de leurs périphériques intégrés (timers avancés, ADC haute résolution, interfaces série) et constituent une plateforme idéale pour l'implémentation de systèmes PWM.

L'objectif de ce projet est de concevoir un système embarqué capable de générer un signal PWM à fréquence et rapport cyclique réglables en temps réel, réglés via l'ADC, avec suivi du fonctionnement assuré par UART et indicateurs lumineux. Ce projet permet également de renforcer la compréhension du fonctionnement des timers, de l'ADC et de la gestion des périphériques dans un environnement embarqué.

# Chapitre 1

## Elaboration du projet

### 1.1 Requis théoriques

#### Signal MLI ou PWM

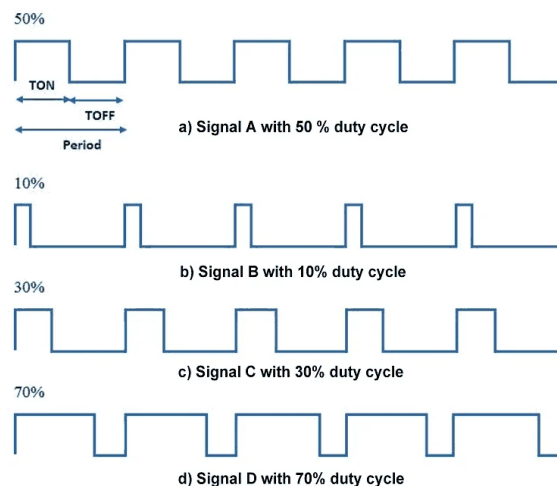
La **PWM (Pulse Width Modulation)** est une technique permettant de contrôler la puissance moyenne transmise à une charge en modulant le *rapport cyclique* d'un signal rectangulaire à fréquence constante. Elle repose sur la comparaison continue entre :

- **Signal modulant**  $V_{ref}(t)$  : tension de référence ou consigne.
- **Signal porteur**  $V_{porteur}(t)$  : signal périodique triangulaire ou en dents de scie.

Le signal PWM  $s(t)$  est défini par :

$$s(t) = \begin{cases} V_{max} & \text{si } V_{ref}(t) > V_{porteur}(t) \\ 0 & \text{si } V_{ref}(t) \leq V_{porteur}(t) \end{cases} \quad (1.1)$$

#### Signal MLI



## Paramètres principaux

### Période et fréquence

La fréquence de commutation est :

$$f_{PWM} = \frac{1}{T}$$

### Rapport cyclique

Le rapport cyclique  $\alpha$  (Duty Cycle) :

$$\alpha = \frac{T_{ON}}{T} = \frac{T_{ON}}{T_{ON} + T_{OFF}}, \quad V_{moy} = \alpha V_{max}, \quad P_{moy} = \alpha P_{max} = \alpha \frac{V_{max}^2}{R}$$

### Résolution

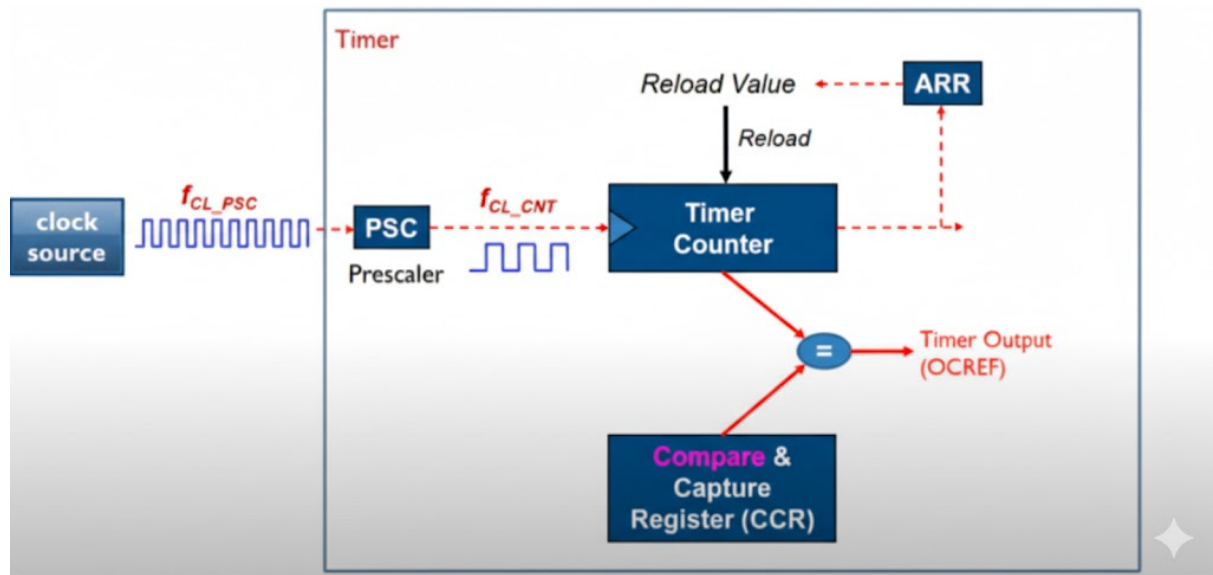
Pour un PWM numérique à  $n$  bits :

$$N_{niveaux} = 2^n, \quad \Delta\alpha_{min} = \frac{1}{2^n} \times 100\%$$

## 1.2 Les TIMERS

### Architecture et composition d'un TIMER

Architecture du timer



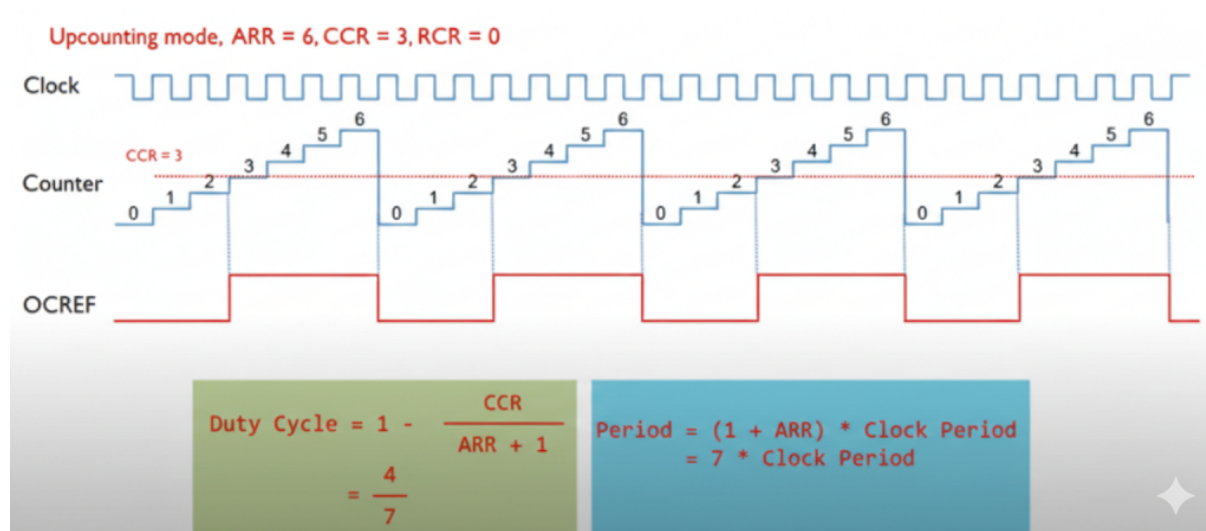
### Fonctionnement du Timer en mode PWM (MLI)

Le mode de **Modulation de Largeur d'Impulsion (MLI)** permet de générer un signal numérique dont la durée de l'état haut est contrôlable. Ce principe est largement utilisé pour le pilotage de moteurs, la variation de puissance ou la commande d'actionneurs.

Le fonctionnement de l'architecture MLI repose sur les étapes suivantes :

- **Division de l'horloge :** Le signal d'horloge source  $f_{CL\_PSC}$  est appliqué au prescaler (PSC), qui divise cette fréquence afin d'obtenir une fréquence de comptage adaptée notée  $f_{CL\_CNT}$ .
- **Comptage cyclique :** Le compteur (CNT) incrémente à chaque impulsion de l'horloge  $f_{CL\_CNT}$  jusqu'à atteindre la valeur définie dans le registre d'auto-rechargement (ARR). Une fois cette valeur atteinte, le compteur est réinitialisé à zéro, ce qui définit la période du signal MLI.
- **Comparaison et génération de la sortie (OCREF) :** La valeur instantanée du compteur est comparée en temps réel à celle du registre de capture/ comparaison (CCR) :
  - Le signal de sortie reste à l'état **bas** tant que  $CNT < CCR$ .
  - Le signal de sortie passe à l'état **haut** dès que  $CNT \geq CCR$ .

Ainsi, en modifiant la valeur du registre CCR, il est possible d'ajuster le rapport cyclique du signal MLI, et par conséquent de contrôler la puissance délivrée à la charge.



## Les Formules Utilisées

$$\text{TIM Clock} = \frac{\text{APB TIM Clock}}{\text{Prescaler}}$$

$$\text{Fréquence} = \frac{\text{TIM Clock}}{\text{ARR}}$$

$$\text{Duty Cycle (\%)} = \frac{\text{CCR}_x}{\text{ARR}} \times 100$$

## Conversion Analogique-Numérique (ADC)

### Principes fondamentaux

L'ADC (Analog-to-Digital Converter) convertit les signaux analogiques continus en valeurs numériques discrètes. Pour ce projet :

- **Résolution** : 12 bits (4096 valeurs : 0 à 4095)
- **Canaux utilisés** : 2 (CH0 et CH1)
- **Mode opération** : Scan avec polling
- **Plage d'entrée** : 0V à 3.3V

### Formules de conversion ADC

Pour convertir la valeur ADC (0-4095) en fréquence (1000-10000 Hz) :

$$f = 1000 + \frac{\text{ADC\_value} \times 9000}{4095} \quad (1.2)$$

Pour convertir en rapport cyclique (0-100%) :

$$\text{Duty\%} = \frac{\text{ADC\_value} \times 100}{4095} \quad (1.3)$$

# Chapitre 2

## Fonctionnement général

### Fonctionnalités principales

#### Génération d'un signal PWM à paramètres variables

Le système met en oeuvre la génération d'un signal PWM dont la fréquence et le rapport cyclique peuvent être ajustés dynamiquement. Les principales caractéristiques de cette génération sont résumées ci-dessous :

- **Timer utilisé** : TIM1 (Timer avancé)
- **Canal de sortie** : Canal 1 associé à la broche PE9
- **Plage de fréquence** : de 1 kHz à 10 kHz, sélectionnée via les entrées analogiques
- **Fréquence de l'horloge système** : 168 MHz
- **Prescaler** : égal à 0 afin de garantir une précision maximale
- **Résolution** : 12 bits, correspondant à 4096 niveaux distincts

La mise à jour des paramètres du signal PWM est réalisée en temps réel à partir des valeurs converties par l'ADC, sans interruption du fonctionnement global du système.

#### Conversion analogique-numérique (ADC)

La conversion analogique-numérique est assurée par le module ADC1 configuré pour la lecture de deux canaux distincts. Les paramètres de configuration sont les suivants :

- **Mode SCAN** : activé pour permettre la lecture séquentielle de plusieurs canaux
- **Canal 0 (PA0)** : utilisé pour le réglage de la fréquence PWM (1 kHz à 10 kHz)
- **Canal 1 (PA1)** : dédié au contrôle du rapport cyclique (0% à 100%)
- **Résolution** : 12 bits
- **Mode de lecture** : polling, garantissant une acquisition synchrone des deux canaux
- **Temps déchantillonnage** : 84 cycles afin d'assurer une mesure stable et fiable

## Commande des LEDs indicatrices

Trois LEDs connectées aux broches GPIO PD12, PD13 et PD14 sont utilisées pour fournir une indication visuelle du niveau du rapport cyclique du signal PWM. Leur fonctionnement est décrit dans le tableau suivant :

**Fonctionnement des LEDs indicatrices**

LED	Broche	Condition d'activation
LED1	PD12	Allumée si le rapport cyclique est supérieur ou égal à 33%
LED2	PD13	Allumée si le rapport cyclique est supérieur ou égal à 66%
LED3	PD14	Allumée si le rapport cyclique est supérieur ou égal à 95%

## Communication UART pour le suivi du système

Une communication série UART est mise en place afin de superviser l'état du système en temps réel. Les paramètres de cette communication sont définis comme suit :

- **Interface utilisée** : UART2
- **Débit de transmission** : 115200 bauds
- **Périodicité d'affichage** : toutes les 2 secondes, en mode non bloquant
- **Format des données** : affichage ASCII sous forme tabulaire avec jauge visuelle
- **Informations transmises** : fréquence du signal PWM, rapport cyclique et état des LEDs

**Exemple de trame UART :**

```
+--- ETAT DU SYSTEME -----+
| Frequence   : 5000 Hz          |
| Rapport Cy  : 50 %   [|||||.....] |
| Etat LEDs   : [1:ON ] [2:OFF] [3:OFF] |
+-----+
```

## Architecture matérielle

### Configuration des ressources du microcontrôleur STM32F407

Le tableau ci-dessous présente la configuration des principales ressources matérielles utilisées dans ce projet :

**Configuration des ressources du STM32F407**

Ressource	Configuration	Commentaires
Horloge système	PLL à 168 MHz	HSE 8 MHz → PLL (M=8, N=336, P=2)
TIM1	Mode PWM, Prescaler = 0	Sortie sur le canal 1 (PE9)
ADC1	Mode Scan, 2 canaux	Canaux 0 et 1, lecture par polling
UART2	115200 bauds	TX sur PA2, RX sur PA3
GPIO	3 sorties numériques	LEDs sur PD12, PD13 et PD14

### Schéma de connexion des périphériques

Le tableau suivant récapitule les connexions entre les périphériques externes et le microcontrôleur :

**Récapitulatif des connexions matérielles**

Périphérique	Broche STM32	Fonction associée
Potentiomètre 1	PA0 (ADC_CH0)	Réglage de la fréquence du signal PWM
Potentiomètre 2	PA1 (ADC_CH1)	Réglage du rapport cyclique du PWM
Sortie PWM	PE9 (TIM1_CH1)	Génération du signal PWM en sortie
LED1	PD12	Indication visuelle du niveau 1
LED2	PD13	Indication visuelle du niveau 2
LED3	PD14	Indication visuelle du niveau 3

# Chapitre 3

## Implémentation logicielle

### Architecture générale du code

Le programme principal suit cette structure :

#### **Phase 1 : Initialisation (une seule fois)**

- Configuration de l'horloge système
- Initialisation des périphériques (GPIO, ADC, TIM, UART)
- Affichage du message de bienvenue sur UART
- Démarrage du Timer1 PWM

#### **Phase 2 : Boucle principale (infinie)**

- Lecture des deux canaux ADC (polling)
- Calcul de la fréquence et du duty cycle
- Mise à jour des registres Timer (ARR, CCR)
- Contrôle des LEDs selon le duty
- Envoi périodique des données UART (toutes les 2 secondes)
- Délai de stabilisation (5ms)

## Points clés de l'implémentation

### Gestion non-bloquante du polling ADC

```
1 // Lecture des deux canaux ADC en mode scan
2 HAL_ADC_Start(&hadc1);
3
4 // Attente conversion Canal 1 (Fréquence)
5 if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK) {
6     adc_freq_raw = HAL_ADC_GetValue(&hadc1);
7
8     // Attente conversion Canal 2 (Duty Cycle)
9     if (HAL_ADC_PollForConversion(&hadc1, 10) == HAL_OK) {
10         adc_duty_raw = HAL_ADC_GetValue(&hadc1);
11     }
12 }
13 HAL_ADC_Stop(&hadc1);
```

Cette approche garantit que le système ne se bloque pas si une conversion échoue.

### Conversion ADC vers valeurs utiles

```
1 // Convertir ADC(0-4095) en Frequence (1000-10000 Hz)
2 frequency_hz = FREQ_MIN +
3     ((uint32_t)adc_freq_raw * (FREQ_MAX - FREQ_MIN)) / 4095;
4
5 // Convertir ADC(0-4095) en Duty (0-100%)
6 duty_percent = ((uint32_t)adc_duty_raw * 100) / 4095;
7
8 // Securites
9 if (frequency_hz < FREQ_MIN) frequency_hz = FREQ_MIN;
10 if (duty_percent > 100) duty_percent = 100;
```

## Mise à jour dynamique du PWM

```

1 // Calcul des registres Timer
2 // ARR = (Clock / Freq) - 1
3 timer_arr = (SYSCLOCK / frequency_hz) - 1;
4
5 // CCR = (ARR + 1) * Duty / 100
6 timer_ccr = ((timer_arr + 1) * duty_percent) / 100;
7
8 // Mise a jour immediate du PWM
9 __HAL_TIM_SET_AUTORELOAD(&htim1, timer_arr);
10 __HAL_TIM_SET_COMPARE(&htim1, TIM_CHANNEL_1, timer_ccr);

```

La modification des registres ARR et CCR en temps réel permet de changer la fréquence et le duty instantanément.

## Gestion du temps pour UART

```

1 // Delai de 2 secondes (2000ms) pour avoir le temps de lire
2 if (HAL_GetTick() - last_uart_tick >= 2000) {
3     // Construction et envoi du message
4     sprintf(uart_buffer,
5         "+--- ETAT DU SYSTEME -----+\r\n"
6         "| Frequence   : %-5lu Hz           |\r\n"
7         "| Rapport Cy  : %-3lu %%    [%s]    |\r\n"
8         "| Etat LEDs   : [1:%s] [2:%s] [3:%s] |\r\n"
9         "+-----+\r\n",
10        frequency_hz, duty_percent, jauge,
11        (duty_percent >= 33) ? "ON " : "OFF",
12        (duty_percent >= 66) ? "ON " : "OFF",
13        (duty_percent >= 100) ? "ON " : "OFF");
14
15    HAL_UART_Transmit(&huart2, (uint8_t*)uart_buffer,
16                      strlen(uart_buffer), 100);
17    last_uart_tick = HAL_GetTick();
18 }

```

Utilisation d'un compteur basé sur les ticks système pour éviter l'utilisation d'inter-  
rptions bloquantes.

## Fonction de contrôle des LEDs

```
1 void Update_LEDs(uint32_t duty) {  
2     // Eteindre tous les LEDs d'abord  
3     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);  
4     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);  
5     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);  
6  
7     // Allumer les LEDs selon le duty cycle  
8     if (duty >= 33) {  
9         HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);  
10    }  
11    if (duty >= 66) {  
12        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);  
13    }  
14    if (duty >= 95) {  
15        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);  
16    }  
17 }
```

# Chapitre 4

## Résultats et tests

### Fonctionnement nominal

Le système fonctionne comme prévu avec les caractéristiques suivantes :

- **Fréquence PWM** : Ajustable de 1 kHz à 10 kHz via potentiomètre
- **Rapport cyclique** : Ajustable de 0% à 100% via potentiomètre
- **Réactivité** : Mise à jour instantanée de la sortie PWM
- **Monitoring UART** : Affichage correct toutes les 2 secondes
- **Contrôle LEDs** : Allumage progressif selon le duty cycle

### Tests effectués

#### Résumé des tests effectués

Test	Résultat	Observations
Génération PWM	Succès	Fréquence correcte, duty variable
Lecture ADC	Succès	Conversion stable et rapide
Mise à jour dynamique	Succès	Changements instantanés
Affichage UART	Succès	Format correct, jauge visuelle fonctionnelle
Contrôle LEDs	Succès	Transitions correctes aux seuils

### Observations et performance

#### Réactivité du système

La réactivité du système est excellente. Les modifications des potentiomètres sont détectées et appliquées instantanément sans latence observable.

## **Stabilité de l'ADC**

L'ADC fournit des valeurs stables même avec des entrées légèrement bruyantes. Les conversions sont rapides et fiables.

## **Charge CPU**

Avec un délai de 5ms dans la boucle principale, l'utilisation du CPU reste modérée et permet d'autres tâches si nécessaire.

## **Affichage UART**

L'affichage toutes les 2 secondes offre un bon compromis entre :

- Information suffisante pour suivre les changements
- Charge CPU minimale (pas de saturation série)
- Lisibilité (chaque bloc reste visible longtemps)

# Conclusion

Ce projet a permis de concevoir et de réaliser un système embarqué basé sur le micro-contrôleur **STM32F407**, dédié à la génération d'un signal **PWM** à fréquence et rapport cyclique réglables. L'objectif principal, visant la maîtrise de la PWM avec intégration d'entrées analogiques et de mécanismes de supervision, a été pleinement atteint.

Sur le plan théorique, le travail a renforcé la compréhension du fonctionnement des timers, de l'ADC et de l'influence du rapport cyclique sur la puissance délivrée. Sur le plan pratique, l'utilisation du **timer TIM1** a permis de générer un signal précis et stable, dont les paramètres sont ajustés dynamiquement via des potentiomètres, avec un suivi en temps réel grâce à l'UART et aux LEDs indicatrices.

Les tests effectués ont validé la fiabilité, la réactivité et la stabilité du système, confirmant la pertinence des choix techniques matériels et logiciels. Ce projet constitue ainsi une base solide pour des applications industrielles telles que le contrôle de moteurs, la variation de puissance et les systèmes de commande embarqués.