République Tunisienne

Ministère de L'Enseignement Supérieur et de la recherche Scientifique
Université de Tunis

National Higher School of Engineers

of Tunis
Département Génie Electrique

Code : GPE-RP-02-00
Date de création : 16-06-2023

Ref: PFA2-24-25.

Rapport de Projet de Fin d'Année

Deuxième année en Génie Electrique

Présenté et soutenu publiquement le 27/05/2025

Par

Ben Jemaa Aymen

Hammami Nour

Jammeli Mohamed Yassine

Système de Contrôle pour Véhicule Électrique

Composition du jury

Monsieur	**A.Zaafouri**	**Président**
Monsieur	**F.Farhani**	**Encadrant**

Dédicaces

A mes chers parents

Pour avoir sacrifié vos vies afin de donner un meilleur goût aux nôtres. Vous étiez et êtes toujours à mes côtés pour me prodiguer des conseils, me soutenir et me donner envie de persévérer pour un lendemain meilleur. Que Dieu leurs procure bonne santé et longue vie.

A mes frères et ma famille

Pour l'amour, la confiance et le respect que vous m'avez toujours octroyés.

A mes chers amis et connaissances

Pour le soutien moral, les encouragements dont vous avez fait preuve. Ce travail n'aura jamais pu voir le jour sans leurs soutiens indéfectibles et sans limite.

Merci d'être toujours là pour moi.

Aymen

Dédicaces

À mes chers parents

Pour avoir tant donné de vous-mêmes afin d'embellir ma vie. Vous avez toujours été présents pour me conseiller, me soutenir et m'encourager à avancer avec espoir vers un avenir meilleur. Que Dieu vous accorde santé et longue vie.

À mes frères et à toute ma famille

Merci pour votre amour, votre confiance et le respect que vous m'avez constamment témoigné.

À mes amis et à toutes les personnes qui m'entourent

Votre soutien moral, vos encouragements et votre présence constante ont été précieux tout au long de ce parcours. Ce travail n'aurait jamais pu aboutir sans votre appui indéfectible.

Nour

Dédicaces

A celui qui m'a indiqué la bonne voie

En me rappelant que la volonté fait toujours Les grands Hommes. . .

A ma source de fierté, à mon père

A celle qui a attendu avec patience les fruits de sa bonne éducation

A mon adorable mère. . .

A mes frères pour leurs aides et soutiens

A mes frères

A mes chers

Il n'y pas de mots,

Pour décrire cet amour,

Pour décrire ce sentiment,

Qui est plus grand que l'infini.

Je dédie ce travail

Yassine

Remerciements

Avant tout, je remercie notre encadreur pédagogique **Mr. F.FARHANI** enseignant à L'ENSIT, pour toutes ses contributions, ses encouragements et ses conseils assez utiles et fructueux qui nous ont aidés à la réalisation de ce travail. Qu'elle trouve ici l'expression de notre plus profonde reconnaissance.

Nous tenons à exprimer notre profonde gratitude à notre professeur **Mr A. ZAAFOURI** chef département génie électrique au sein de l'Ecole Nationale Supérieur d'Ingénieur de Tunis, pour son aide et ses conseils malgré ces nombreuses charges.

Nous ne manquerons pas non plus de remercier tous les membres du jury qui nous ont honorés en acceptant de juger ce travail et de nous faire part de leurs remarques, surement pertinents, qui contribueront, sans nul doute, au perfectionnement du présent travail.

Sommaire

Liste des figures

Liste des tableaux

Liste des abréviations

IHM : Interface Homme-Machine

Introduction Générale

Dans un monde confronté à de profonds bouleversements environnementaux, le secteur des transports joue un rôle central dans la transition énergétique. L'augmentation des émissions de gaz à effet de serre, la pollution de l'air et l'épuisement des ressources fossiles poussent les gouvernements, les industriels et les citoyens à repenser les modes de déplacement. C'est dans ce contexte que la voiture électrique apparaît comme une solution d'avenir, alliant innovation technologique et respect de l'environnement.

Contrairement aux véhicules thermiques, les voitures électriques utilisent un ou plusieurs moteurs alimentés par une batterie rechargeable, généralement au lithium-ion. Elles présentent de nombreux avantages : absence d'émissions directes de CO_2, réduction de la pollution sonore, rendement énergétique supérieur et entretien simplifié. Ces caractéristiques en font une alternative de plus en plus adoptée à l'échelle mondiale.

Le marché des véhicules électriques connaît une croissance rapide, portée par des politiques incitatives, une sensibilisation accrue aux enjeux climatiques et des progrès constants dans les domaines des batteries, de l'électronique de puissance et de l'intelligence embarquée. Toutefois, leur généralisation pose encore plusieurs défis, tels que l'autonomie limitée, le temps de recharge, le coût élevé des batteries et la nécessité de développer une infrastructure de recharge adaptée.

Dans ce contexte, il est essentiel d'adopter une démarche d'optimisation continue : chaque choix technologique doit viser la performance, la durabilité et l'efficacité énergétique. Ce projet s'inscrit pleinement dans cette logique en analysant les différents sous-systèmes des voitures électriques et en identifiant des solutions techniques pertinentes pour accompagner l'évolution vers une mobilité plus propre, plus intelligente et plus durable.

Chapitre 1

Présentation du projet

I. Introduction

Ce chapitre est dédié à la présentation de notre projet intitulé « Système de Contrôle pour Véhicule Électrique ». Ce projet s'appuie sur plusieurs théories de commande avancées et englobe divers domaines techniques, allant de l'électronique de puissance à l'automatique, en passant par l'électrotechnique et le traitement de signal.

II. Présentation du projet

Ce projet consiste en la conception et la réalisation d'un système de contrôle pour une véhicule électrique, dans le cadre d'une participation à la compétition internationale **Shell Eco-marathon**. L'objectif principal est d'optimiser la gestion énergétique du véhicule afin de maximiser son autonomie et son efficacité. Le système intègre des stratégies de commande avancées, adaptées aux contraintes de performance et de consommation. Ce projet allie innovation, rigueur technique et engagement pour une mobilité durable.

Figure 1:Voiture électrique

Le diagramme présenté illustre l'architecture complète d'un système de propulsion électrique destiné à un véhicule à motorisation BLDC (Brushless DC Motor). Il met en évidence l'interconnexion des différents composants nécessaires au fonctionnement, à la commande et à la protection du moteur, à partir d'une source d'énergie électrique rechargeable. Ce système repose sur une batterie lithium-ion de 48V, un onduleur DC/AC, un contrôleur de véhicule, ainsi que plusieurs capteurs et dispositifs de sécurité. L'objectif de ce schéma est d'assurer une gestion efficace de l'énergie tout en garantissant la sécurité, la surveillance en temps réel et le contrôle précis du moteur électrique.

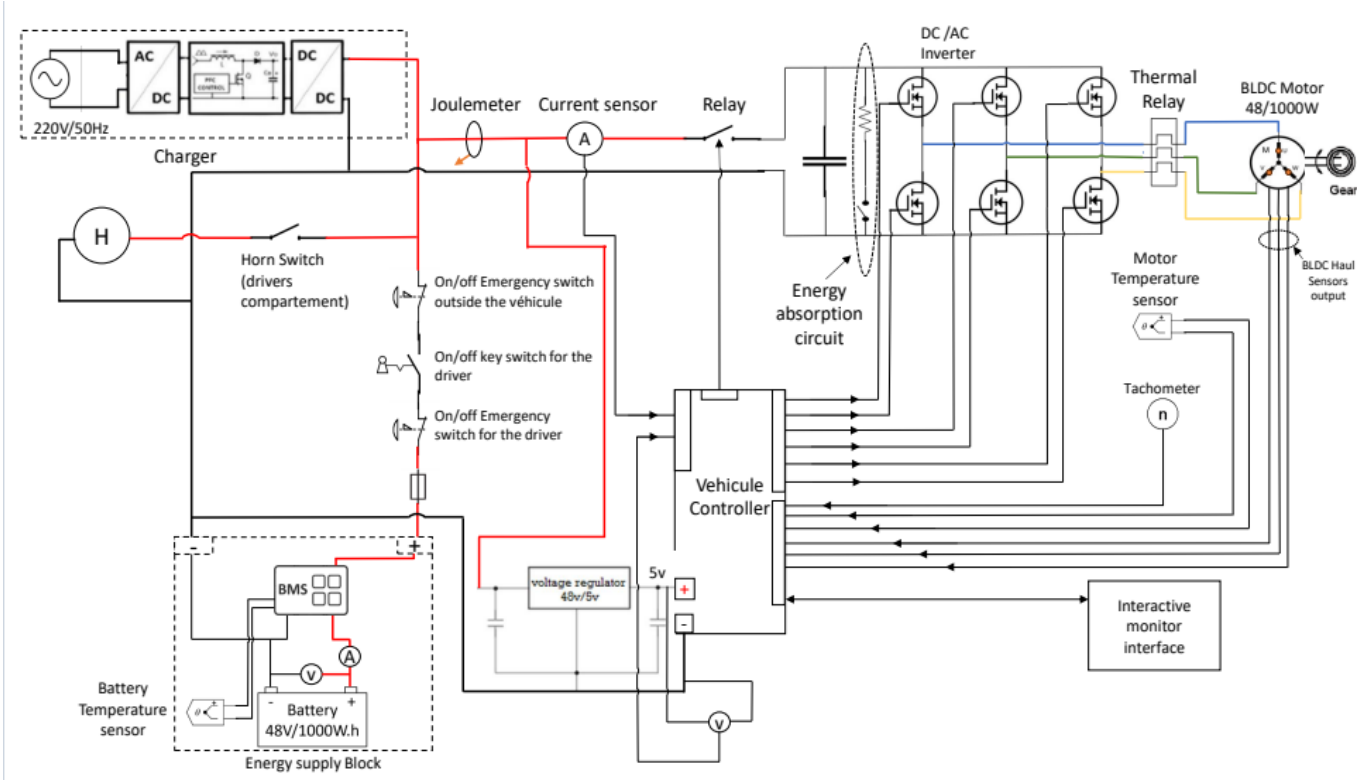


Figure 2:diagramme synoptique

III. Objectif

- Développer l'algorithme de commande et de régulation du moteur, ainsi que la supervision de l'état du véhicule
- Réaliser l'unité centrale de traitement du véhicule et mettre en œuvre l'interface de supervision sur l'IHM
- Élaborer et réaliser les tests en conditions de fonctionnement réelles à bord du prototype du véhicule

IV. Cahier de charge

Le projet consiste à concevoir et développer une unité centrale de commande pour un véhicule Shell Eco-marathon. Cette unité embarquée aura pour rôle de centraliser les données issues des capteurs (tension, courant, température, capteurs Hall, etc.), de superviser l'état du véhicule, et de générer les signaux PWM nécessaires au pilotage d'un moteur brushless via un ESC. Elle devra également assurer la communication avec une interface homme-machine (IHM) pour

l’affichage en temps réel des informations critiques. Le système sera basé sur un microcontrôleur et devra fonctionner en temps réel avec une faible consommation énergétique. Le cahier des charges comprend le développement de l’algorithme de commande et de régulation du moteur, l’intégration matérielle des capteurs, l’IHM et du contrôleur, ainsi que les tests en conditions réelles à bord du prototype. L’ensemble du système devra être robuste, fiable et conforme aux contraintes de la compétition, avec une documentation complète à livrer en fin de projet.

Conclusion

Ce premier chapitre a permis de définir les bases de notre projet en détaillant les objectifs, les fonctionnalités principales, ainsi que les étapes clés du développement de la **carte de commande pour un moteur brushless** destiné à une voiture électrique. Nous avons également mis en évidence les caractéristiques techniques, notamment l'architecture matérielle et logicielle de la carte, ainsi que les critères d’évaluation qui guideront la réalisation du projet. La mise en place de ce cahier des charges constitue une étape essentielle pour assurer la conformité du système aux exigences de performance, de fiabilité et de sécurité, en particulier dans le cadre de la compétition **Shell Eco-marathon**. Les prochaines étapes consisteront à concrétiser cette conception à travers des simulations, des tests pratiques et une validation continue des résultats obtenus.

Chapitre 2

Choix des Composantes

I. Introduction

Dans ce chapitre, nous présentons les différents composants matériels sélectionnés pour la réalisation du système de contrôle d'un véhicule électrique. Le choix de chaque élément a été effectué en fonction des exigences techniques du cahier des charges, de la compatibilité entre les modules, ainsi que de la fiabilité et de la disponibilité des composants sur le marché. L'objectif principal est de construire une architecture cohérente, performante et sécurisée, capable d'assurer les fonctions essentielles du véhicule : la gestion du moteur, la surveillance de la batterie, la communication entre les sous-systèmes et l'interface avec l'utilisateur. Chaque composant sera donc justifié en tenant compte de ses spécifications, de son rôle dans le système, et de son adéquation avec les contraintes du projet (puissance, précision, communication, sécurité, coût...).

II. Choix des composantes

1. Composantes utilisées

Le tableau ci-dessous présente l'ensemble des composants matériels sélectionnés pour la réalisation du système de propulsion électrique. Chaque composant répondre aux exigences techniques du cahier des charges, en assurant la fiabilité, la sécurité et la performance globale du système. Cette sélection couvre les différents sous-systèmes du véhicule : commande, puissance, détection, sécurité et interface utilisateur.

Tableau 1:– tableau descriptif composantes utilisées

N°	Composant	Référence / Exemple	Fonction
1	Microcontrôleur	STM32F407VGT6	Unité centrale de contrôle (VCU)
2	Capteurs effet Hall (moteur)	Intégrés dans le moteur	Détection de la position rotor pour la commutation
3	Capteur de température (moteur)	NTC ou thermistance intégrée	Surveillance thermique du moteur
4	Capteur de courant	ACS712 / INA219	Mesure du courant consommé
5	Tachymètre / Vitesse	Encodeur magnétique / optique	Mesure des RPM du moteur
6	Convertisseur DC-DC	LM2596 (48V → 5V)	Alimentation 5V pour STM32 et capteurs
7--	Interface NEXTION		Communication avec BMS, moteur, etc.

2. Analyse et Justification du Choix des Composants

Afin d'assurer le bon fonctionnement du système de propulsion électrique, chaque composant doit être sélectionné avec rigueur selon des critères techniques, fonctionnels et économiques. Dans cette partie, nous présentons une analyse détaillée des différents éléments matériels utilisés dans notre projet. Pour chaque composant, nous justifions notre choix en mettant en avant ses avantages, sa compatibilité avec l'architecture globale, et sa capacité à répondre aux exigences du cahier des charges. Lorsque cela est pertinent, des comparaisons avec d'autres alternatives disponibles sont proposées pour démontrer la pertinence des choix retenus.

2.1. Microcontrôleur – STM32F407VGT6

a. Description

Le microcontrôleur joue le rôle de **cerveau central du système**. Il assure :

- Le contrôle moteur avec des fréquences PWM élevées
- La lecture des capteurs (température, vitesse, courant…)
- La communication avec le BMS via UART
- Le dialogue avec l'interface utilisateur via UART

b. Avantages

- Cœur Cortex-M4 cadencé à 168 MHz avec unité FPU (calculs flottants rapides)
- Interfaces multiples : CAN, **UART**, SPI**, I2C**, **ADC**, PWM
- Large support logiciel (STM32CubeIDE, HAL, RTOS, bibliothèques open source)

c. Justification du choix

Ce microcontrôleur offre un excellent compromis entre puissance, bonne gestion de l'interruption, robustesse de programmation, interfaces de communication multiples et nombres des pins E/S suffisant, modularité **et coût** qui l'a rend le choix le plus optimal afin de mettre en œuvre l'unité de traitement de données

Tableau 2: Comparatif avec d'autres microcontrôleurs

Microcontrôleur	Fréquence	Mémoire Flash / RAM	Niveau	Avantages / Inconvénients
STM32F407VGT6	168 MHz	1MB / 192KB	⭐ ⭐ ⭐ ⭐	Puissant, fiable, bien documenté
STM32F103C8T6 (Bluepill)	72 MHz	64KB / 20KB	⭐ ⭐	Trop limité pour ce projet
ESP32	240 MHz	4MB / 520KB	⭐ ⭐ ⭐	Bon en WiFi , mais pas fait pour contrôle temps réel
Arduino Mega 2560	16 MHz	256KB / 8KB	⭐	Trop lent

2.2. Capteurs à effet Hall (détection de position rotor)

a. Description

Les capteurs à effet Hall intégrés dans le moteur BLDC permettent de **détecter la position du rotor**. Cette information est indispensable pour synchroniser la commutation des phases dans un contrôle BLDC.

b. Avantages

- **Sans contact** → très bonne durabilité
- Détection rapide et fiable
- Intégrés dans la majorité des moteurs BLDC → pas besoin d'installation externe
- Le moteur BLDC choisi intègre déjà ces capteurs, ce qui simplifie l'architecture. Ils sont essentiels pour **assurer un bon fonctionnement du contrôle moteur**, notamment en basse vitesse où la méthode sensorless devient instable.

c. Comparatif de méthodes de détection de position

Tableau 3:tableau comparatif

Méthode	Précision	Coût	Complexité	Fiabilité	Remarque
Effet Hall intégré	⭐ ⭐ ⭐	Faible	Faible	Élevée	Meilleur rapport simplicité/prix
Encodeur optique	⭐ ⭐ ⭐ ⭐	Élevé	Élevée	Élevée	Très précis mais coûteux
Contrôle sensorless	⭐ ⭐	Aucun	Moyen	Moyenne	Inutilisable à basse vitesse

2.3. BMS (Battery Management System)

a. Description

Le **BMS** est un élément indispensable qui surveille et protège la batterie. Il mesure :

- La **tension**, le **courant**, la **température**
- Le **SOC** (State of Charge)
- Équilibre les cellules lors de la charge

b. Avantages

- Protège la batterie contre surcharge, décharge profonde, surchauffe
- Communication via **CAN** (important pour l'intégration avec la STM32)
- Assure la **longévité et la sécurité** de la batterie

c. Justification du choix

Un **Smart BMS** comme **JBD ou Daly** avec interface **CAN** est choisi pour sa compatibilité avec les microcontrôleurs STM32 et pour son intégration simple dans un réseau embarqué. Il évite d'avoir à développer un système de protection maison complexe.

d. Comparatif de BMS

Tableau 4:tableau comparatif

BMS	Communication	Équilibrage	Tension supportée	Remarque
JBD Smart BMS	CAN, UART	Actif	48V	Adapté VE, précis
Daly Smart BMS	UART	Passif	48V	Moins cher, pas de CAN
BMS générique	Aucun	Passif	12–24V	✕ Pas compatible avec 48V

2.4. Capteur de température

a. Description

Capteur analogique (NTC, LM35…) permettant de mesurer la température en temps réel.

b. Avantages

- Simple à interfacer avec STM32 (entrée analogique)
- Permet la mise en sécurité en cas de surchauffe
- Peu coûteux, fiable

c. Justification du choix

Indispensable pour protéger la batterie et le moteur contre les surchauffes qui pourraient endommager les cellules ou le bobinage.

d. Comparatif capteurs de température

Tableau 5:tableau comparatif

Capteur	Type	Plage	Précision	Interface	Remarque
LM35	Analogique	-55 à 150°C	±0.5°C	ADC	Linéaire et facile à lire
NTC Thermistor	Analogique	Varie	Moyenne	ADC	Non linéaire, bon marché
DS18B20	Numérique	-55 à 125°C	±0.5°C	1-Wire	Précis, mais nécessite protocole

2.5. Convertisseur DC/AC (MOSFET + IR2101 ou VESC)

a. Description

Convertit le courant continu (DC) de la batterie en courant triphasé (AC) pour alimenter le moteur BLDC.

b. Avantages

- Contrôle précis du moteur (vitesse, couple)
- Intègre souvent protections surchauffe/surtension
- Le VESC permet aussi le retour d'énergie (freinage régénératif)

c. Justification du choix

Pour des raisons pédagogiques et économiques, le pont MOSFET (IR2101 + IRF540N) est un bon choix. Pour une solution clé en main et professionnelle : **VESC**.

d. Comparatif onduleurs

Tableau 6:tableau comparatif

Solution	Commande PWM	CAN	Facilité d'intégration	Remarque
IR2101 + MOSFET	✓	✗	Moyen	Nécessite circuit fait maison
VESC	✓	✓	✓ Facile	Très utilisé en VE open-source
ESC RC hobby	Partiel	✗	✓ Facile	✗ Pas adapté aux charges continues

Chapitre 3

Réalisation pratique

I. Introduction

Dans ce chapitre, nous présentons la mise en œuvre concrète du système étudié précédemment. Après avoir défini l'architecture générale et les choix de composantes dans les chapitres précédents, il s'agit ici de détailler les étapes de la réalisation pratique, depuis la conception du schéma fonctionnel jusqu'à l'implémentation matérielle et logicielle. Cette phase comprend l'assemblage des composants, la programmation éventuelle des microcontrôleurs, ainsi que les vérifications nécessaires pour valider le bon fonctionnement du système. Elle constitue une étape essentielle, car elle permet de confronter les hypothèses théoriques aux contraintes réelles de terrain.

II. Intégration du capteur de courant CJMCU-758

Pour assurer la surveillance du courant circulant entre la batterie lithium et l'onduleur, nous avons intégré un capteur de courant isolé de type **CJMCU-758**, basé sur le circuit **ACS758**. Ce capteur repose sur l'effet Hall, ce qui lui permet de mesurer des courants continus ou alternatifs entre la ligne de puissance et le circuit de traitement. Il présente une excellente linéarité et une réponse rapide, adaptée aux applications de puissance.

La carte CJMCU-758 est équipée de connecteurs à vis pour le passage direct du courant haute intensité (via les bornes IP+ et IP−), et de broches de connexion pour l'alimentation et la sortie analogique (Vcc, GND, OUT). Le signal analogique en sortie est proportionnel au courant mesuré et centré autour de 2.5 V (pour 0 A), ce qui facilite son acquisition par un **convertisseur analogique-numérique (ADC)** du microcontrôleur.

Ce capteur joue un rôle crucial dans la protection du système : il permet de détecter rapidement toute **surcharge**, **court-circuit** ou **dérive de fonctionnement**, et peut ainsi être utilisé pour déclencher des mécanismes de sécurité ou d'arrêt d'urgence.

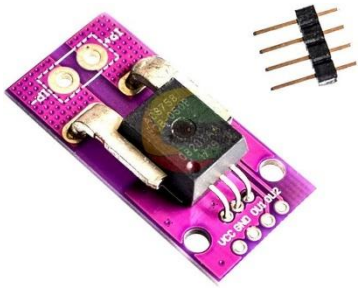


Figure 3:capteur température

1. Programme développé

Le programme développé sur la carte STM32F4 a pour objectif de mesurer l'intensité du courant à l'aide d'un capteur analogique, de traiter les données acquises, puis d'afficher le résultat sur un écran Nextion via une interface UART (USART2). Après l'initialisation des périphériques (GPIO, ADC1 et UART), une phase de calibration est effectuée afin de déterminer la tension de référence correspondant à un courant nul (QOV). Ensuite, dans la boucle principale, le microcontrôleur effectue une acquisition répétée de la tension en sortie du capteur à l'aide de l'ADC (avec un moyennage sur 10 échantillons pour améliorer la stabilité de la mesure), puis convertit cette tension en une valeur de courant en appliquant le coefficient de sensibilité du capteur (40 mV/A). Si le courant mesuré dépasse un seuil prédéfini (1 A), il est affiché ; dans le cas contraire, un message indiquant l'absence de courant est généré. L'ensemble des données est ensuite transmis à l'écran Nextion grâce à une fonction dédiée qui assure le formatage et l'envoi des informations via l'interface série. Ce processus est exécuté de manière périodique, avec un intervalle de 500 ms entre chaque cycle de mesure.

-

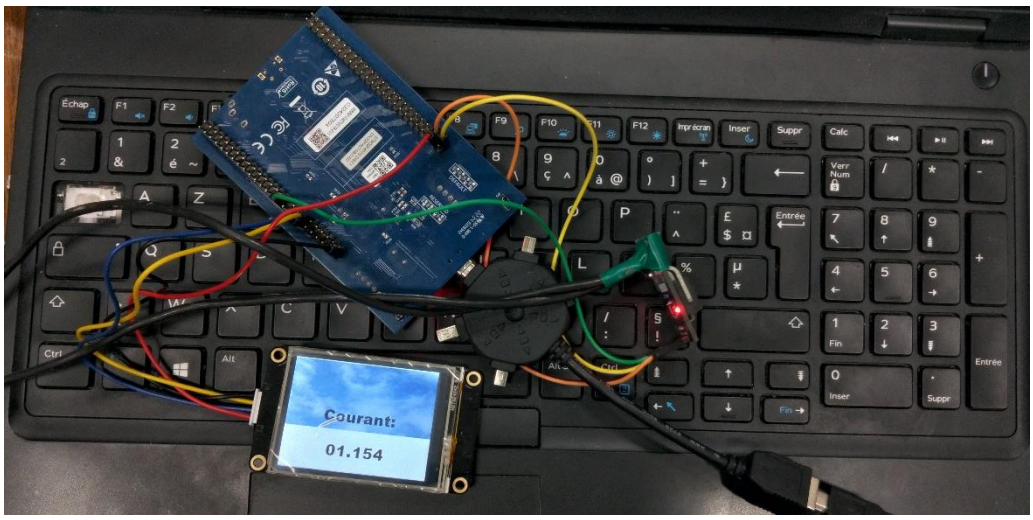


Figure 4:cablage capteur température

2. Test effectué

Figure 5:test capteur de courant

3. Application utilisée (XiaoXiangElectri)

L'application mobile présentée permet la **surveillance en temps réel** d'un système de gestion de batterie (BMS), en se connectant sans fil (généralement via Bluetooth) à un module BMS compatible. L'interface utilisateur est intuitive et visuellement claire, avec une jauge centrale affichant l'état de charge (**SOC : State of Charge**) de la batterie.

L'application fournit également des **informations électriques détaillées** telles que :

- la **tension totale** (Total Volt : 50.53 V),
- le **courant instantané** (Electric : 0.00 A),
- la **puissance** (Power : 0.00 W),
- les tensions maximale, minimale et moyenne des cellules,
- la **différence de tension** entre cellules (PressureDiff),
- et un **index de cycle** (Cycle-Index), ici à 15.

Elle affiche également l'état de plusieurs fonctionnalités comme la **charge** (ChgMos), la **décharge** (DisMos), l'**équilibrage** des cellules (Balance), et les **protections activées**. Des capteurs internes permettent de visualiser en temps réel la **température des MOSFETs**, la température ambiante (T1, T2) et l'humidité (si disponible).

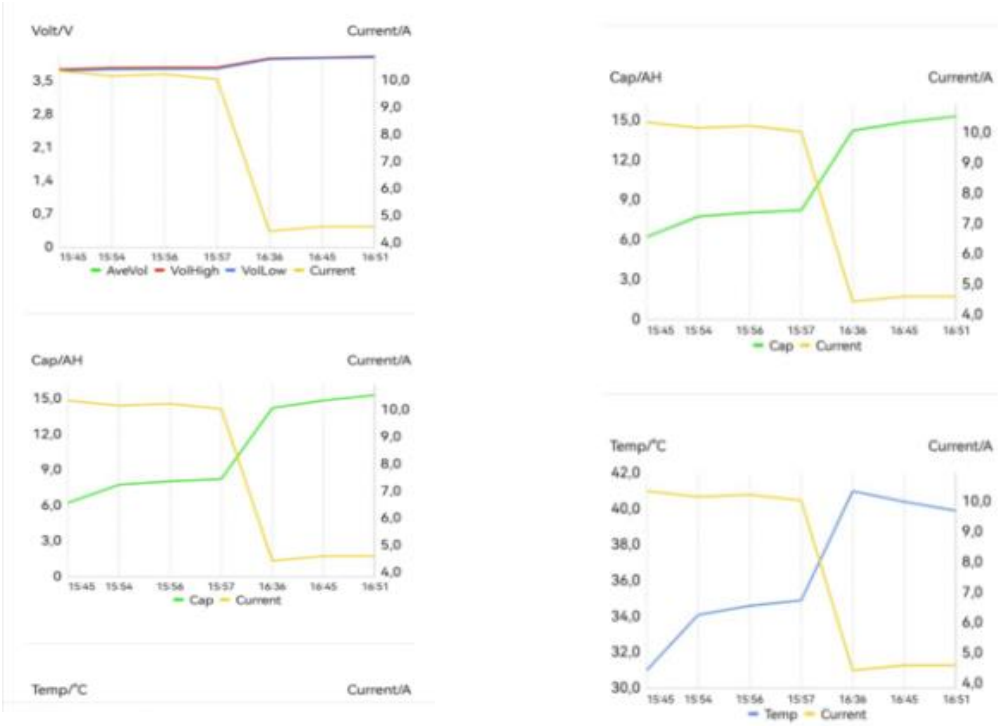


Figure 6:Extrait des courbes de charge et décharge

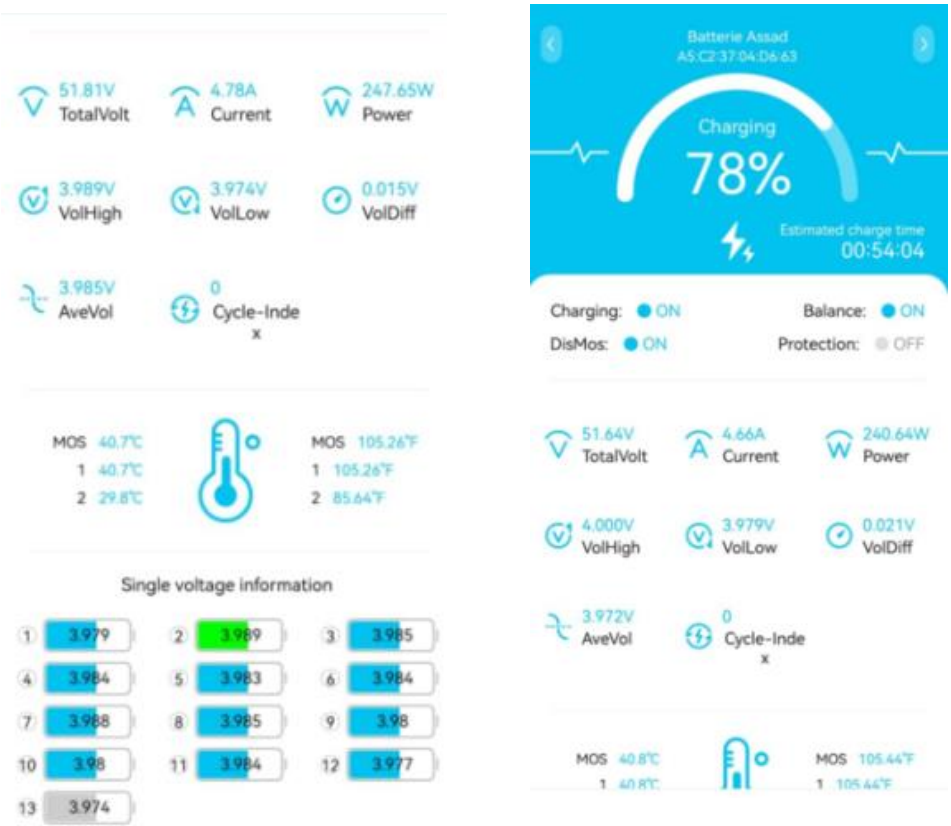


Figure 7:Application utilisée (XiaoXiangElectri)

b. Interface NEXTION

L'IHM Nextion est programmée à l'aide du logiciel dédié **Nextion Editor**, un environnement de développement graphique fourni par le fabricant. Ce logiciel permet de concevoir facilement des interfaces utilisateur à partir d'éléments visuels tels que des textes, jauges, boutons, images et autres composants interactifs. Chaque élément peut être associé à une variable ou à une instruction série, ce qui permet de recevoir ou d'envoyer des données via le port **UART** du microcontrôleur. Grâce à Nextion Editor, l'ensemble de l'interface est exécuté de manière autonome sur l'écran, évitant ainsi au microcontrôleur (comme le STM32) de gérer l'affichage ou la logique de l'IHM. **La figure ci-dessous illustre le processus de débogage visant à afficher la variable de température via le logiciel Nextion Editor.**

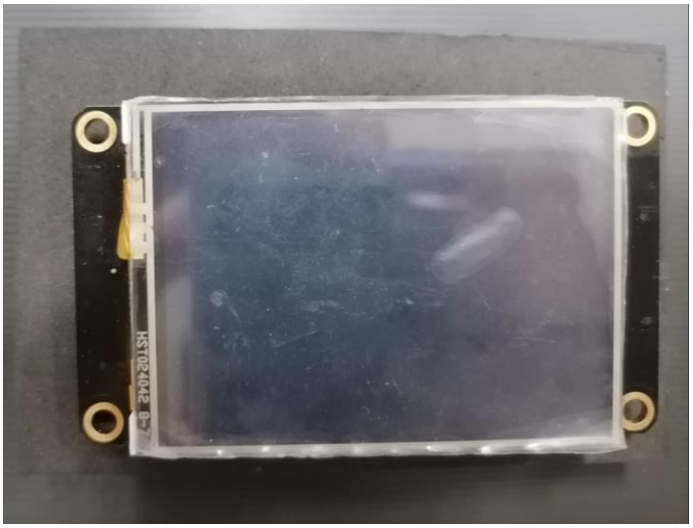


Figure 8: Interface NEXTION

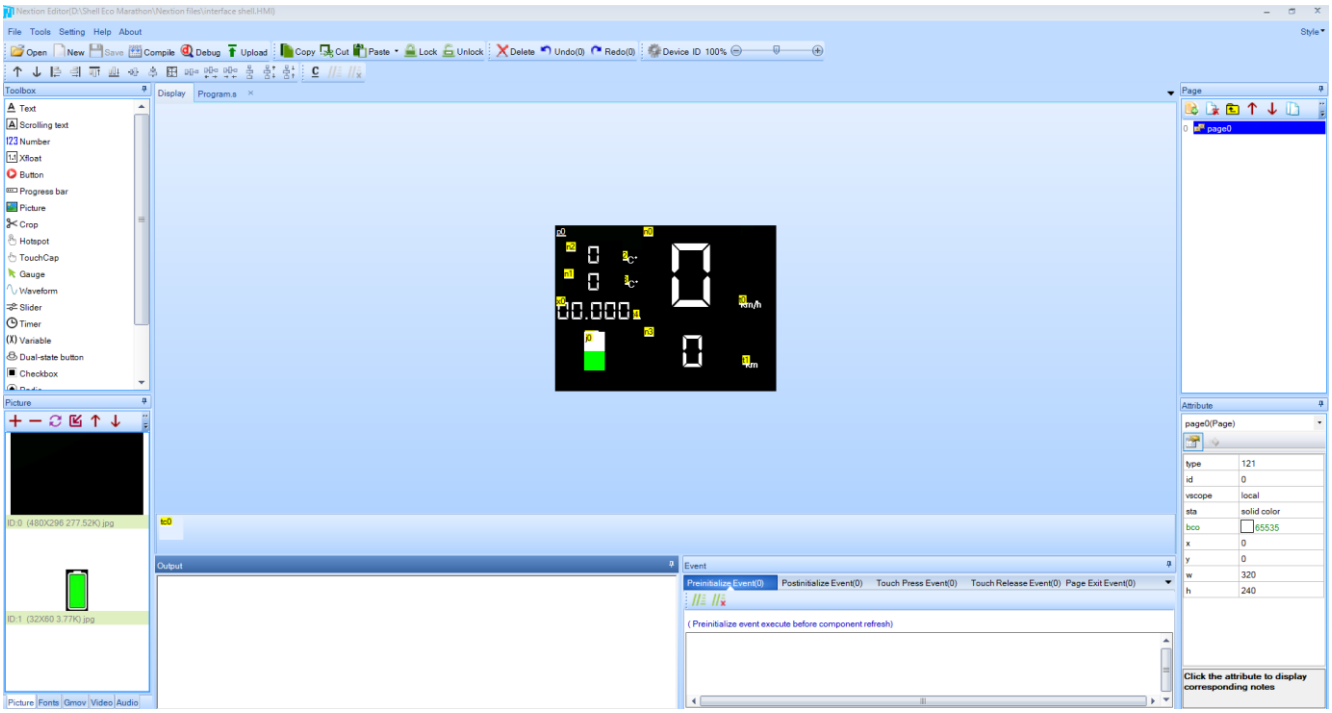


Figure 9:Nextion Editor

III. Algorithme de commande

1. Génération de signaux PWM logiciels sur STM32

Ce programme a pour objectif principal de générer des signaux de commande PWM de manière logicielle à l'aide d'un microcontrôleur STM32. Ces signaux sont destinés à piloter des actionneurs tels que des moteurs, en suivant une séquence de commutation précise. Le code ne fait appel à aucun timer matériel, mais repose plutôt sur des boucles de temporisation pour simuler le comportement d'un signal PWM. Chaque signal est appliqué successivement sur différentes broches GPIO de la carte, accompagné d'un signal de commande fixe sur une autre broche, selon un ordre déterminé. Ce fonctionnement cyclique permet, par exemple, de contrôler un système triphasé ou un ensemble de moteurs de manière séquentielle. Ce type d'implémentation est particulièrement utile pour des tests, des démonstrations ou des applications où la précision temporelle n'est pas critique, tout en mettant en œuvre une logique de commande organisée et modulaire.

2. Visualisation des signaux de commande via logic analizer

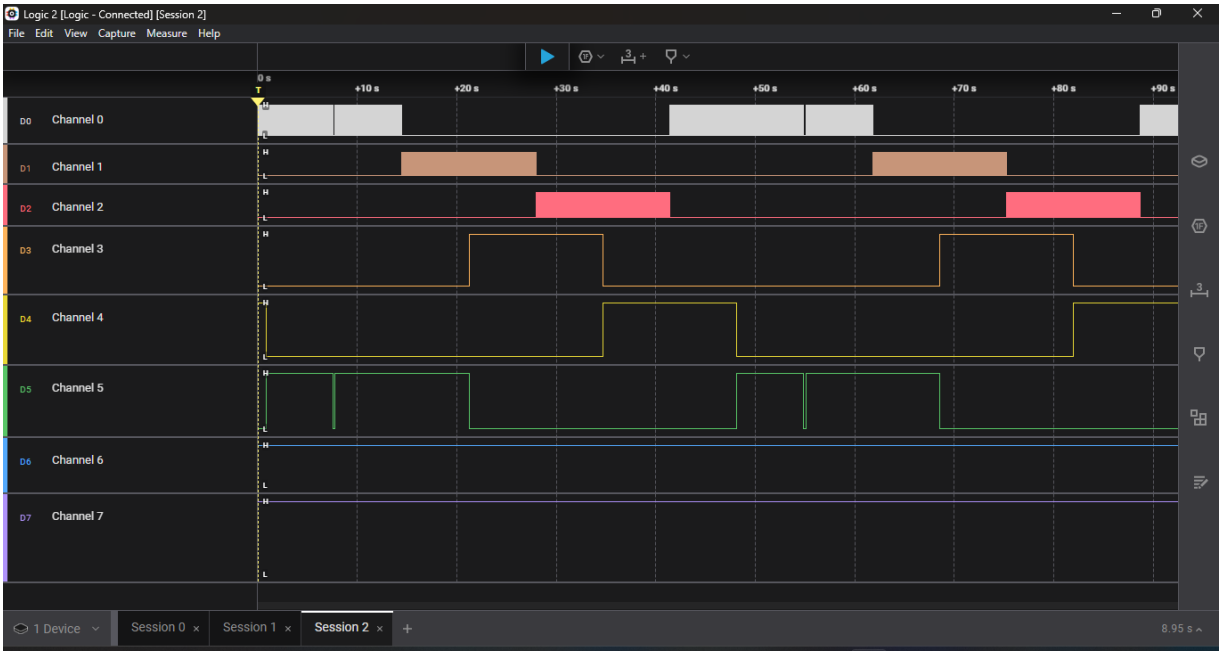
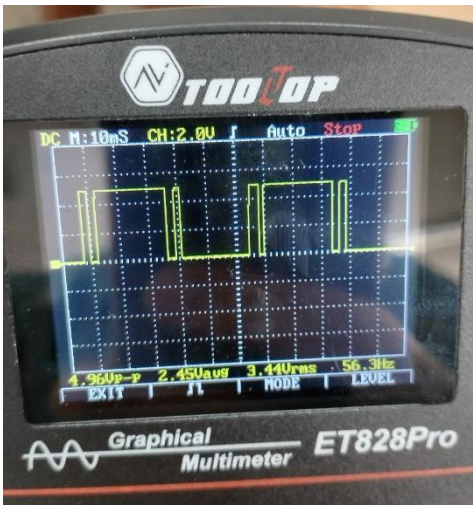


Figure 10:signaux de commande(PWM)



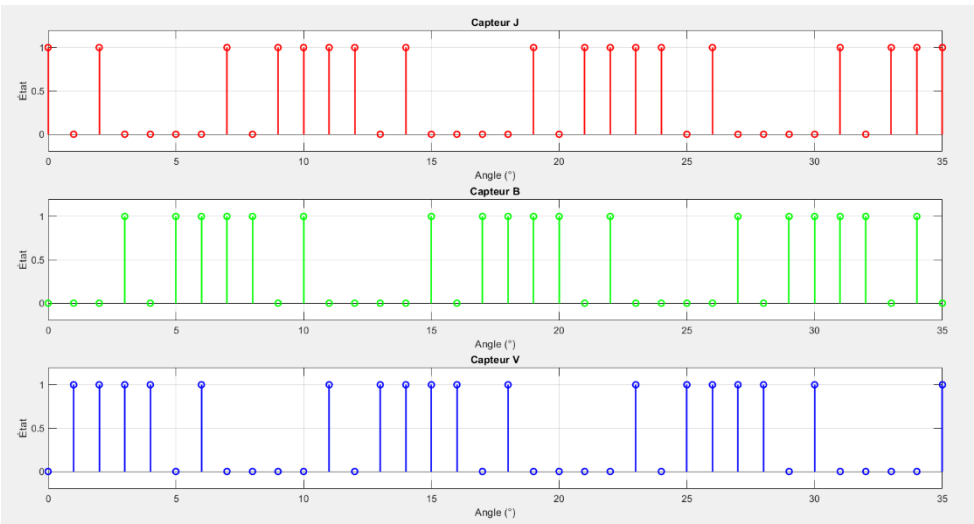


Figure 11 : Courbe de la réponse du capteur Effet HAUL intégré au BLDC

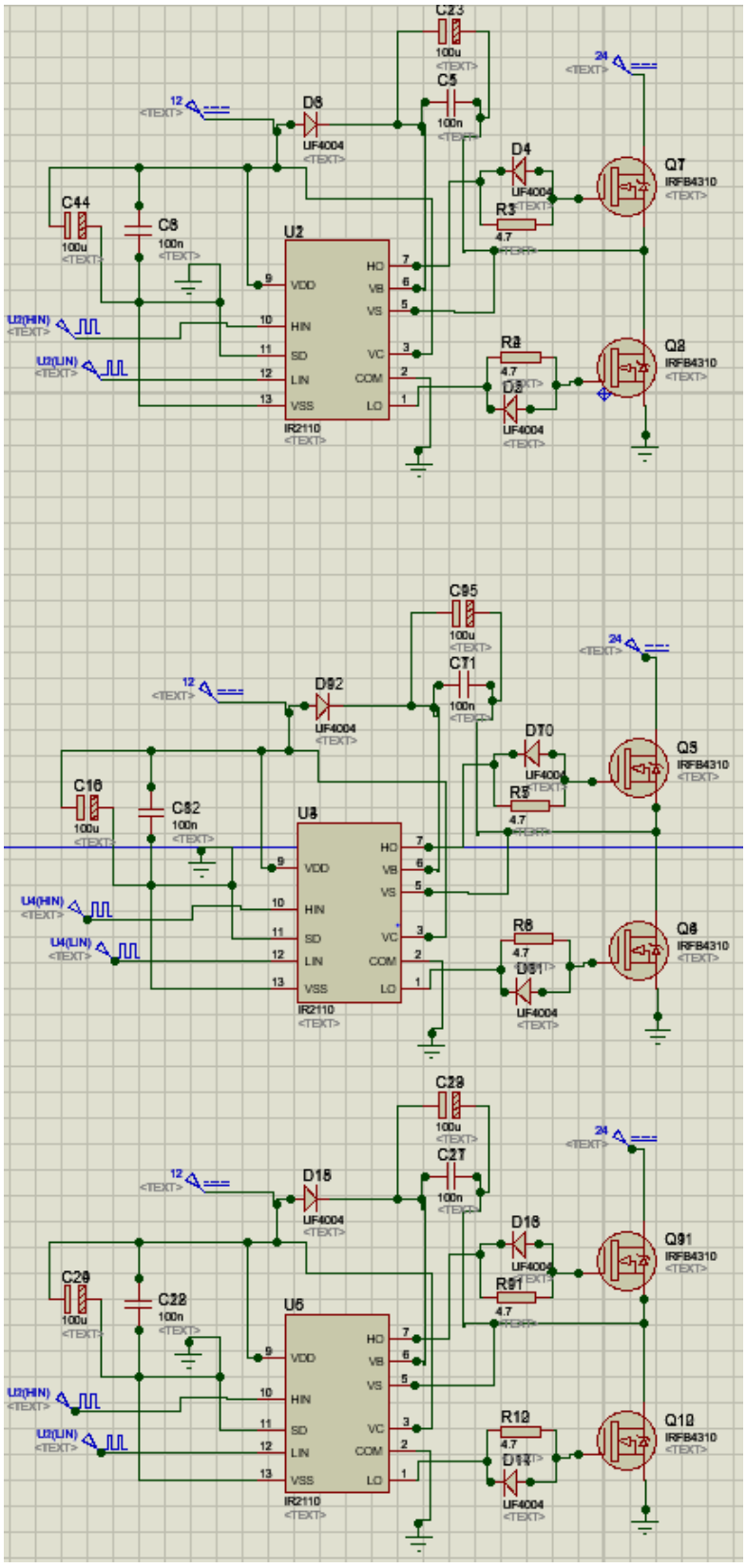


Figure 12:Conception électronique de l'ESC

3. Commande Trapézoïdale des Moteurs BLDC

3.1 Principe de fonctionnement

La commande trapézoïdale est l'une des méthodes les plus couramment utilisées pour le pilotage des moteurs Brushless à courant continu (BLDC). Elle tire son nom de la forme des tensions ou courants appliqués aux enroulements du moteur, qui suivent une **forme trapézoïdale** au lieu d'une forme sinusoïdale.

Le fonctionnement repose sur une **commutation en six étapes** (appelée "six-step commutation") au cours de chaque cycle électrique. À chaque étape :

- Deux des trois phases du moteur sont activées : l'une est alimentée positivement (HIGH), l'autre est reliée à la masse (LOW),
- La troisième phase est laissée en **état flottant** (non connectée),
- Cette commutation est synchronisée avec la position angulaire du rotor, généralement mesurée à l'aide de **capteurs à effet Hall** intégrés au moteur.

Ainsi, le couple est généré par l'interaction entre le champ magnétique du rotor et celui des enroulements statoriques commutés. La **simplicité** de cette méthode en fait un choix populaire dans les systèmes où **l'efficacité, le coût et la rapidité de mise en œuvre** sont des critères importants.

3.2 Mise en œuvre

Pour implémenter la commande trapézoïdale :

- Le contrôleur lit en temps réel l'état des capteurs à effet Hall pour déterminer la position du rotor,
- En fonction de cette position, il applique la séquence de commutation appropriée aux transistors du pont en H qui alimente le moteur,
- Un **rapport cyclique de PWM** peut être utilisé pendant chaque étape pour contrôler la vitesse du moteur ou ajuster le couple.

Ce type de commande est particulièrement adapté aux **microcontrôleurs** (comme les STM32) car il nécessite peu de ressources de calcul, et les signaux PWM peuvent être générés via des timers standards.

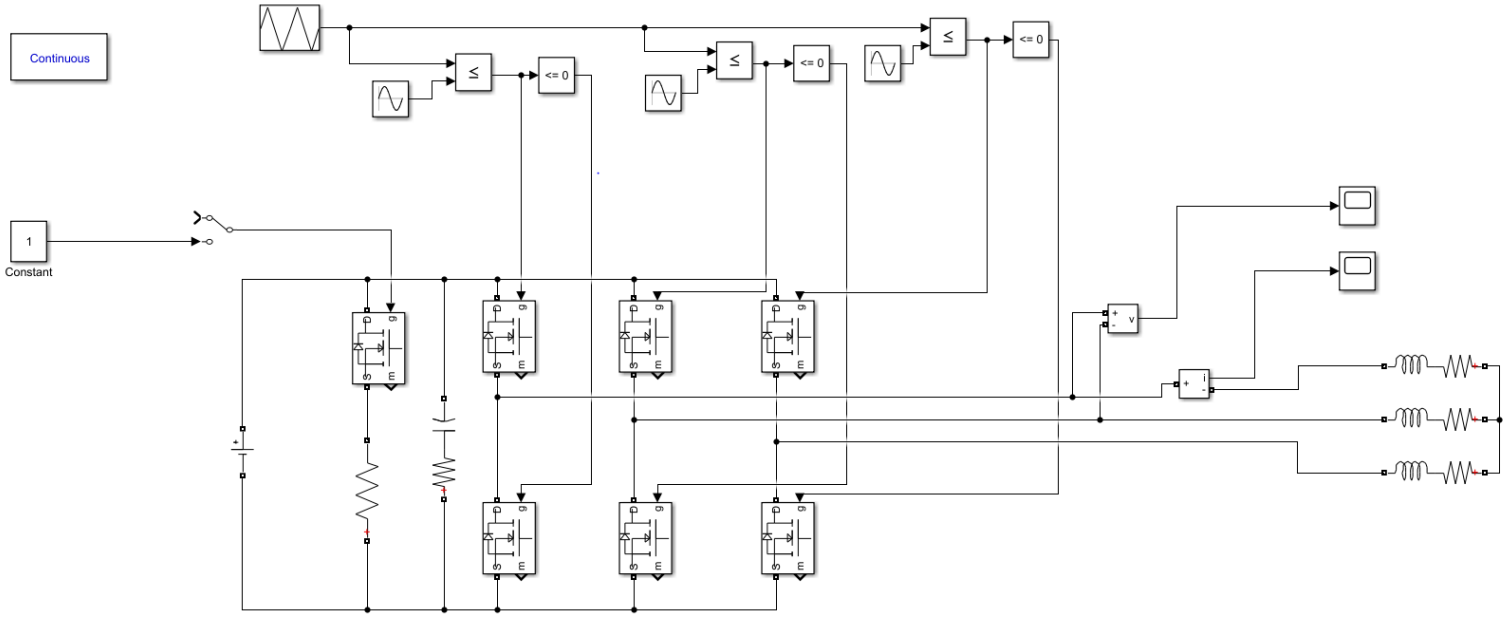


Figure 13: Modélisation (Matlab) de Bloc de commande BLDC

Et de simulation de là loin de commande

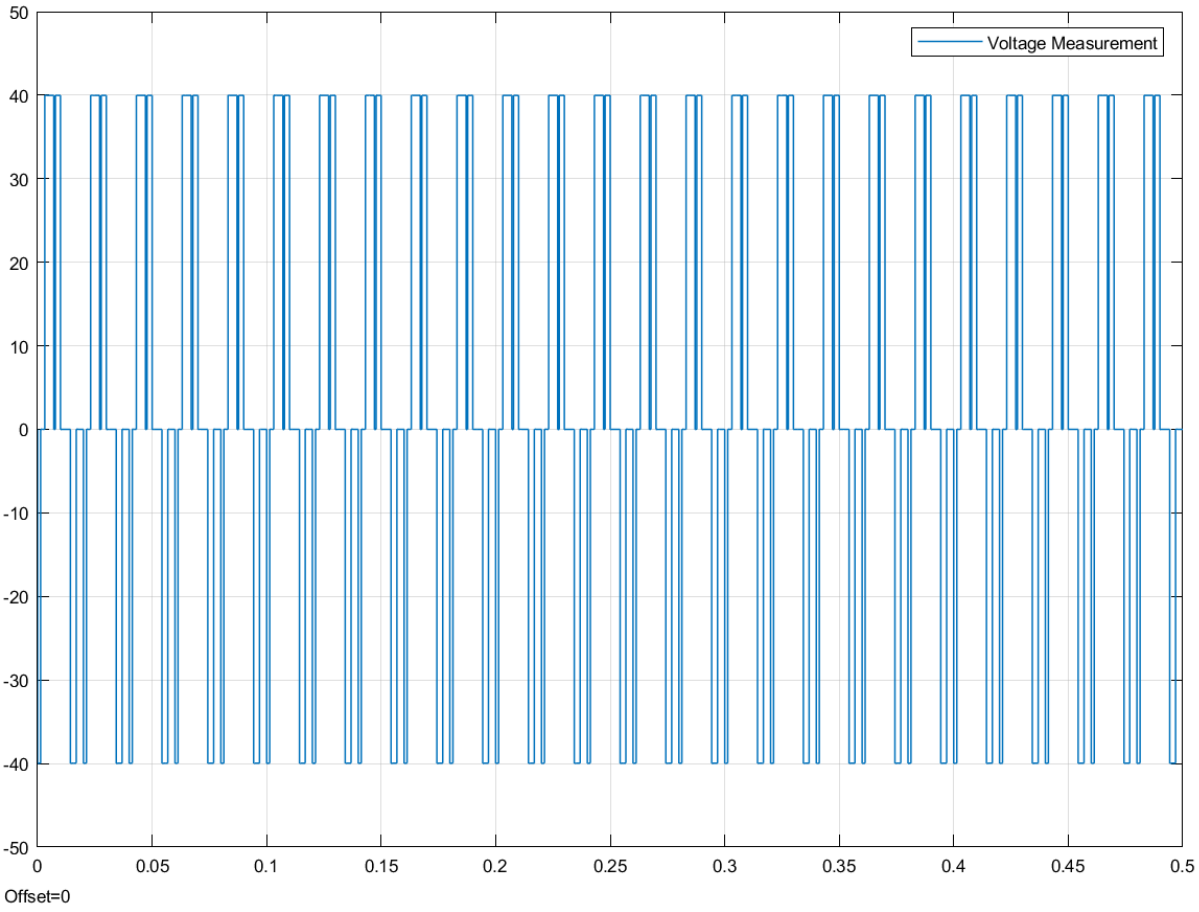


Figure 14: Courbe de mesure de tension entre phase du convertisseur de fréquence

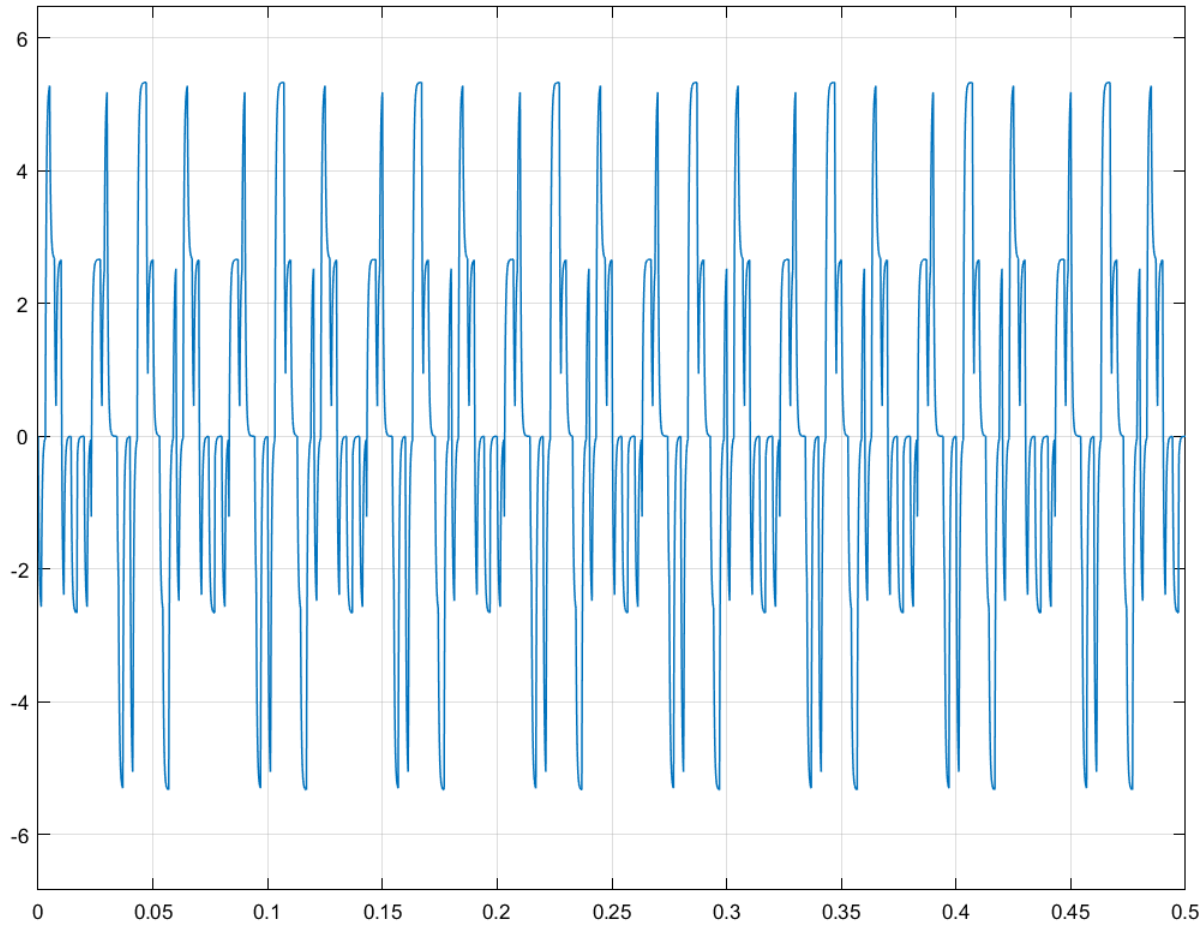


Figure 15: Courbe de mesure de Courant dans une phase du convertisseur de fréquence

3.3 Avantages

- **Simplicité de conception** : ne nécessite pas de transformations complexes (comme Clarke ou Park),
- **Facile à implémenter** avec des microcontrôleurs classiques,
- **Coût réduit** : idéal pour les applications à faible budget,
- **Réactivité** : la commutation à six étapes permet une réponse rapide.

3.4 Inconvénients

- **Ondulations de couple**: le couple n'est pas parfaitement constant, ce qui peut causer des vibrations,
- **Moins précis** que les commandes sinusoïdales ou vectorielles,
- **Moins efficace** à haute vitesse ou dans des applications exigeant une grande douceur de fonctionnement.

3.5 Comparaison avec d'autres méthodes de commande

Tableau 7:tableau comparatif

Critère	Commande Trapézoïdale	Commande Sinusoïdale	**Commande Vectorielle (FOC)**
Forme du courant	Trapézoïdale	Sinusoïdale	Sinusoïdale (en quadrature)
Complexité	Faible	Moyenne	Élevée
Capteurs requis	Capteurs Hall (ou capteurless)	Capteurs Hall ou encodeur	Encodeur ou capteurless
Ondulation du couple	Moyenne à élevée	Faible	Très faible
Douceur de fonctionnement	Moyenne	Bonne	Excellente
Rendement énergétique	Moyen	Bon	Très bon
Utilisation typique	Petits robots, ventilateurs	Véhicules légers, drones	Moteurs hautes performances, robotique de précision

Analyse

- **Par rapport à la commande sinusoïdale**, la commande trapézoïdale est plus simple mais génère davantage d'ondulations dans le couple. Elle peut être suffisante pour des applications comme les ventilateurs, pompes ou véhicules simples, où un couple parfaitement constant n'est pas indispensable.
- **Par rapport à la commande vectorielle (FOC)**, elle est beaucoup plus simple mais bien moins performante. Le FOC est réservé aux applications nécessitant un contrôle

très précis du couple et de la vitesse, comme les servomoteurs industriels, les drones professionnels ou les véhicules électriques haut de gamme.

Conclusion

La commande trapézoïdale constitue un excellent compromis entre **simplicité**, **coût** et **efficacité** pour les applications standards impliquant des moteurs BLDC. Bien qu'elle présente certaines limites en termes de précision et de douceur de rotation, elle reste largement suffisante pour de nombreux systèmes embarqués et projets industriels de moyenne complexité. Toutefois, pour les applications exigeant une grande finesse de contrôle, il est recommandé de se tourner vers des solutions plus avancées comme la commande vectorielle.

Conclusion Générale

Ce projet s'inscrit dans une démarche d'innovation technologique et de transition vers une mobilité durable. À travers la conception et la réalisation d'un système de contrôle pour véhicule électrique, nous avons pu explorer des domaines clés de l'ingénierie tels que l'électronique de puissance, l'automatique, l'embarqué et la gestion énergétique.

La mise en œuvre du microcontrôleur STM32F407VGT6 a permis de centraliser efficacement le traitement des données issues des capteurs, le pilotage du moteur BLDC via la commande trapézoïdale, ainsi que l'interface utilisateur à travers un écran Nextion. Le système a été conçu pour fonctionner en temps réel, tout en respectant les contraintes de consommation, de fiabilité et de sécurité imposées par la compétition Shell Eco-marathon.

L'intégration de capteurs de courant, de température, et d'un BMS intelligent a renforcé la capacité du système à surveiller et protéger les composants critiques du véhicule. La génération logicielle de signaux PWM et la gestion de la communication série illustrent la flexibilité offerte par la plateforme STM32 dans des applications embarquées.

En conclusion, ce projet a permis non seulement de mettre en pratique des compétences multidisciplinaires, mais aussi de proposer une solution fonctionnelle et évolutive pour le contrôle d'un véhicule électrique. Il ouvre la voie à de futures améliorations, notamment l'optimisation de la commande moteur (passage à la commande vectorielle), l'intégration de nouvelles interfaces de communication (CAN, Bluetooth), ou encore l'exploitation de l'intelligence artificielle pour la gestion intelligente de l'énergie embarquée.

Annexe1

Programme développé

```
/* Private define ----------------------------/* USER CODE BEGIN Header */
/**
******************************************************************************
* @file : main.c
* @brief : Main program body
******************************************************************************
* @attention
*
* Copyright (c) 2025 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*
******************************************************************************
*/
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include <stdio.h>
#include <math.h>
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

------------------------------*/
#define VCC 5.0f // Tension d'alimentation du capteur (5V)
#define SENSITIVITY 0.040f // Sensibilité 40mV/A
#define ADC_MAX 4095.0f // Résolution ADC 12 bits
#define CURRENT_CUTOFF 1.0f // Seuil minimal de courant à afficher
#define NUM_SAMPLES 10 // Nombre d'échantillons pour moyennage
#define ADC_REF 3.3f // Tension de référence ADC
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
float current = 0.0f;
float voltage = 0.0f;
float voltage_raw = 0.0f;
float QOV = VCC / 2.0f; // Tension à courant nul (2.5V)
uint16_t readValue = 0;

ADC_HandleTypeDef hadc1;
```

```c
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART2_UART_Init(void);
void CalibrateSensor(void);

/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
uint8_t Cmd_End[3] = {0xff,0xff,0xff};
void CalibrateSensor(void) {
float sum = 0.0f;

for (int i = 0; i < NUM_SAMPLES; i++) {
HAL_ADC_Start(&hadc1);
if (HAL_ADC_PollForConversion(&hadc1, 100) == HAL_OK) {
sum += HAL_ADC_GetValue(&hadc1);
}
HAL_Delay(10);
}

QOV = (ADC_REF / ADC_MAX) * (sum / NUM_SAMPLES);
printf("Calibration terminee. Offset: %.3f V\r\n", QOV);
}
void NEXTION_SendFloat (char *obj, float num, int dp)
{
// convert to the integer
int32_t number = num*(pow(10,dp));

uint8_t *buffer = malloc(30*sizeof (char));
int len = sprintf ((char *)buffer, "%s.vvs1=%d", obj, dp);
HAL_UART_Transmit(&huart2, buffer, len, 1000);
HAL_UART_Transmit(&huart2, Cmd_End, 3, 100);


len = sprintf ((char *)buffer, "%s.val=%ld", obj, number);
HAL_UART_Transmit(&huart2, buffer, len, 1000);
HAL_UART_Transmit(&huart2, Cmd_End, 3, 100);
free(buffer);
}
/* USER CODE END 0 */

/**
* @brief The application entry point.
* @retval int
*/
int main(void)
{
/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/* MCU Configuration--------------------------------------------------------*/
```

```c
/* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ADC1_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
CalibrateSensor();

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{/* ADC Reading with averaging */
float sum = 0.0f;
for (int i = 0; i < NUM_SAMPLES; i++) {
HAL_ADC_Start(&hadc1);
if (HAL_ADC_PollForConversion(&hadc1, 100) == HAL_OK) {
sum += HAL_ADC_GetValue(&hadc1);
}
HAL_Delay(10);
}
readValue = sum / NUM_SAMPLES;

/* Convert to voltage */
voltage_raw = (ADC_REF / ADC_MAX) * readValue;
voltage = voltage_raw - QOV;

/* Calculate current */
current = voltage / SENSITIVITY;

/* Display results */
if (fabsf(current) > CURRENT_CUTOFF) {
printf("Courant mesure: %.2f A\r\n", current);
} else {
printf("Courant nul detecte\r\n");
}
NEXTION_SendFloat("x0", current, 3);

HAL_Delay(500);


/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
```

```
* @brief System Clock Configuration
* @retval None
*/
void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Configure the main internal regulator output voltage
*/
__HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

/** Initializes the RCC Oscillators according to the specified parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
|RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
{
Error_Handler();
}
}

/**
* @brief ADC1 Initialization Function
* @param None
* @retval None
*/
static void MX_ADC1_Init(void)
{

/* USER CODE BEGIN ADC1_Init 0 */

/* USER CODE END ADC1_Init 0 */

ADC_ChannelConfTypeDef sConfig = {0};

/* USER CODE BEGIN ADC1_Init 1 */

/* USER CODE END ADC1_Init 1 */

/** Configure the global features of the ADC (Clock, Resolution, Data Alignment and number
of conversion)
*/
hadc1.Instance = ADC1;
hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
hadc1.Init.Resolution = ADC_RESOLUTION_12B;
```

```
hadc1.Init.ScanConvMode = DISABLE;
hadc1.Init.ContinuousConvMode = ENABLE;
hadc1.Init.DiscontinuousConvMode = DISABLE;
hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc1.Init.NbrOfConversion = 1;
hadc1.Init.DMAContinuousRequests = DISABLE;
hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
if (HAL_ADC_Init(&hadc1) != HAL_OK)
{
Error_Handler();
}

/** Configure for the selected ADC regular channel its corresponding rank in the sequencer
and its sample time.
*/
sConfig.Channel = ADC_CHANNEL_9;
sConfig.Rank = 1;
sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN ADC1_Init 2 */

/* USER CODE END ADC1_Init 2 */

}

/**
* @brief USART2 Initialization Function
* @param None
* @retval None
*/
static void MX_USART2_UART_Init(void)
{

/* USER CODE BEGIN USART2_Init 0 */

/* USER CODE END USART2_Init 0 */

/* USER CODE BEGIN USART2_Init 1 */

/* USER CODE END USART2_Init 1 */
huart2.Instance = USART2;
huart2.Init.BaudRate = 9600;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart2) != HAL_OK)
{
Error_Handler();
}
/* USER CODE BEGIN USART2_Init 2 */

/* USER CODE END USART2_Init 2 */

}

/**
```

```c
* @brief GPIO Initialization Function
* @param None
* @retval None
*/
static void MX_GPIO_Init(void)
{
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
* @brief This function is executed in case of error occurrence.
* @retval None
*/
void Error_Handler(void)
{
/* USER CODE BEGIN Error_Handler_Debug */
/* User can add his own implementation to report the HAL error return state */
__disable_irq();
while (1)
{
}
/* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
/**
* @brief Reports the name of the source file and the source line number
* where the assert_param error has occurred.
* @param file: pointer to the source file name
* @param line: assert_param error line source number
* @retval None
*/
void assert_failed(uint8_t *file, uint32_t line)
{
/* USER CODE BEGIN 6 */
printf("Wrong parameters value: file %s on line %d\r\n", file, line);
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Annexe2

Algorithme de commande

```
#include "main.h"

void SystemClock_Config(void);
static void MX_GPIO_Init(void);

// Fonction PWM logiciel
void generatePWM(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, uint32_t duration_ms,
uint32_t period_us, float duty_cycle) {
    uint32_t high_time = (uint32_t)(period_us * duty_cycle);
    uint32_t low_time = period_us - high_time;
    uint32_t cycles = (duration_ms * 1000) / period_us;

    for (uint32_t i = 0; i < cycles; i++) {
        HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PIN_SET);
        for (volatile uint32_t j = 0; j < high_time * 10; j++);
        HAL_GPIO_WritePin(GPIOx, GPIO_Pin, GPIO_PIN_RESET);
        for (volatile uint32_t j = 0; j < low_time * 10; j++);
    }
}

int main(void)
{
    HAL_Init();
    SystemClock_Config();
    MX_GPIO_Init();

    while (1)
    {
        // Étape 1 : PWM0 (PC0) + PWM5 (PA1)
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_0, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

        // Étape 2 : PWM1 (PC1) + PWM5 (PA1)
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_1, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

        // Étape 3 : PWM1 (PC1) + PWM3 (PC3)
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_1, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_1 | GPIO_PIN_3, GPIO_PIN_RESET);

        // Étape 4 : PWM2 (PC2) + PWM3 (PC3)
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_3, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_2, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2 | GPIO_PIN_3, GPIO_PIN_RESET);
```

```
        // Étape 5 : PWM2 (PC2) + PWM4 (PA0)
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_2, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);

        // Étape 6 : PWM0 (PC0) + PWM4 (PA0)
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_0, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0, GPIO_PIN_RESET);

        // Étape 7 : PWM0 (PC0) + PWM5 (PA1)
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_SET);
        generatePWM(GPIOC, GPIO_PIN_0, 500, 100, 0.5f);
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_1, GPIO_PIN_RESET);

        HAL_Delay(200); // petite pause visible
    }
}

void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSIState = RCC_HSI_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
    HAL_RCC_OscConfig(&RCC_OscInitStruct);
    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_SYSCLK
                                | RCC_CLOCKTYPE_PCLK1 | RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
    HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0);
}

static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_0|GPIO_PIN_1, GPIO_PIN_RESET);

    GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

```
    GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
}

void Error_Handler(void)
{
    __disable_irq();
    while (1) {}
}
```

Bibliographie

[1] :https://www.google.com/search?q=STM32F407VGT6&oq=STM32F407VGT6&gs_lcrp=EgZjaHJvbWUyCQgAEEUYORiABDIHCAEQABiABDIHCAIQABiABDIGCAMQABgeMgYIBBAAGB4yBggFEAAYHjIGCAYQABgeMgYIBxAAGB4yBggIEAAYHjIGCAkQABge0gEJMjQ1M2owajE1qAIIsAIB8QXas6NUwvYzuA&sourceid=chrome&ie=UTF-8

[2]:https://clubdomotiqueairbuspalays.wordpress.com/wp-content/uploads/2018/02/nt403_nextion.pdf

[3] :https://2btrading.tn/accueil/3868-module-capteur-effet-hall-acs758-50a-pour-arduino-cjmcu-758.html?srsltid=AfmBOoo9X5kXaiOgF79TVSJIj_boW40kzAJw-K2T_oR2BgwVliQCmvO-

[4] : https://xiaoxiangelectric.updatestar.com/fr

[5] : https://nextion.tech/editor_guide/