وزارة التعليم العالي والبحث العلمي

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

جامعة عبد الحميد بن باديس -مستغانم

Université Abdelhamid Ibn Badis Mostaganem

كلية العلوم والتكنولوجيا

Faculté des Sciences et de la Technologie

قسم الهندسة الكهربائية

Département de Génie Électrique

MÉMOIRE

Pour obtenir le diplôme de

MASTER EN Automatique

Spécialité : *Automatique et Informatique industrielle*

Présenté par

KIES ABDERAHIM

Étude et réalisation d'un système intelligent de contrôle d'ascenseur basé sur FPGA

Soutenu le 13/10/2026 devant le jury composé de :

Président	: Mr M.BENTOUMI	MCB	Université de Mostaganem
Examinateur	: Mr M.BENOUALI	MCB	Université de Mostaganem
Encadreur	: Mr H.L.BEKLAOUZ	MCB	Université de Mostaganem

Année Universitaire : 2025/2026

Remerciements

J'exprime mes remerciements les plus chaleureux à mes parents, mes frères et sœurs, mes amis ainsi qu'à l'ensemble de ma famille pour leurs encouragements, leur patience et leur soutien inestimable durant toutes mes années d'études.

Je tiens à exprimer mes remerciements les plus sincères à mon encadreur, le Docteur Larbi BAKLAOUZ HADJ, pour son encadrement rigoureux, sa motivation constante et ses précieuses orientations qui m'ont permis de donner le meilleur de moi-même.

Ma profonde gratitude va également aux membres du jury, Messieurs M.BENTOUMI et M.BENOUALI , pour l'honneur qu'ils me font évaluer ce travail de fin d'études et de l'enrichir par leurs remarques avisées.

J'adresse mes vifs remerciements à l'ensemble des enseignants qui ont contribué à ma formation tout au long de mon cursus universitaire.

Que toutes les personnes qui m'ont aidé, de près ou de loin, à l'élaboration de ce mémoire trouvent ici l'expression de ma sincère reconnaissance.

Résumé

Ce travail de fin d'études porte sur l'étude, la conception et l'implémentation matérielle d'un système intelligent de contrôle d'ascenseur basé sur une technologie FPGA (*Field-Programmable Gate Array*). Face aux exigences croissantes de sécurité, de réactivité et de fiabilité dans les infrastructures modernes, les calculateurs séquentiels classiques présentent des limitations en termes de temps de réponse. Pour y pallier, la logique de commande développée dans ce projet s'appuie sur la carte FPGA DE1 d'Altera/Intel (Cyclone II), exploitant le parallélisme matériel pour garantir un traitement déterministe et en temps réel.

Le système a été modélisé sous forme d'une Machine à États Finis (FSM) et décrit en langage VHDL. La validation fonctionnelle et temporelle a été menée sous le simulateur ModelSim, tandis que la synthèse et le placement-routage ont été exécutés via Quartus II. L'architecture logicielle intègre la gestion optimisée des appels, un filtre anti-rebond numérique. Les tests expérimentaux menés sur la carte DE1 et le prototype physique ont démontré un fonctionnement parfaitement stable, un temps de réaction quasi instantané (< 20 ns pour les urgences) et une empreinte matérielle minimale (< 2 % des éléments logiques du FPGA).

Mots-clés : FPGA, VHDL, Carte DE1, Cyclone II, Machine à États Finis (FSM), Contrôleur d'ascenseur, ModelSim, Quartus II, Systèmes embarqués, Temps réel.

Abstract

This final year project focuses on the study, design, and hardware implementation of a smart elevator control system based on FPGA (*Field-Programmable Gate Array*) technology. To overcome the latency limitations of traditional sequential microcontrollers in critical applications, the control unit was implemented on the Altera/Intel DE1 board (Cyclone II FPGA), leveraging hardware parallelism to achieve real-time deterministic performance and high reliability.

The control architecture was modeled using a Finite State Machine (FSM) and described in VHDL language. Functional and timing validations were conducted using ModelSim, while logic synthesis and fitting were performed via Quartus II. The VHDL design incorporates optimized call scheduling, a digital debouncing filter. Experimental tests on the DE1 board and prototype validated the system's stability, displaying an instantaneous response time (< 20 ns for critical safety triggers) with a minimal hardware resource footprint (< 2% of logic elements).

Keywords : FPGA, VHDL, DE1 Board, Cyclone II, Finite State Machine (FSM), Elevator Controller, ModelSim, Quartus II, Embedded Systems, Real-time.

ملخص

يركز مشروع التخرج هذا على دراسة وتصميم والتطبيق المادي لنظام تحكم ذكي للمصاعد باستعمال تقنية مصفوفات البوابات المنطقية القابلة للبرمجة حائلياً (FPGA). وتجاوزاً لمحدودية التأخير الزمني (Latency) في المتحكمات الدقيقة التسلسلية التقليدية في التطبيقات الحرجة، تم تنفيذ وحدة التحكم على لوحة التطوير Altera/Intel DE1 (المزودة بشريحة Cyclone II FPGA)، وذلك بالاستفادة من التوازي المادي (Hardware Parallelism) لتحقيق أداء حتمي في الوقت الفعلي (Real-time Deterministic Performance) وموثوقية عالية.

تمت نمذجة معمارية التحكم باستعمال آلة الحالات المنتهية (FSM) ووصفها بلغة VHDL. كما أجريت عمليات التحقق الوظيفي والزمني باستخدام برنامج ModelSim، بينما تم التجميع والتركيب المنطقي (Logic Synthesis and Fitting) عبر برنامج Quartus II. يتضمن تصميم VHDL جدولة محسّنة للنداءات، وفلتر رقمي لإزالة الارتداد (Debouncing Filter) . وقد أثبتت الاختبارات التجريبية على لوحة DE1 والنموذج الأولي استقرار النظام، حيث أظهرت زمن استجابة لحظي (أقل من 20 نانو ثانية لمشغلات السلامة الحرجة) مع استهلاك أدنى للموارد المادية (أقل من 2% من العناصر المنطقية).

كلمات مفتاحية : لوحة Cyclone II، آلة الحالات المنتهية (FSM)، متحكم المصعد، الأنظمة المدمجة، الوقت الفعلي .

Table des Matières

Liste des Figures..... VI
Liste des Tableaux et Listing VII
Liste des Acronymes et Abréviations..... VIII
Introduction générale..... 1
Chapitre 1 : État de l'Art des Ascenseurs..... 2
1.1 Introduction du chapitre..... 3
1.2 Définition des ascenseurs..... 3
1.3 Historique et évolution des ascenseurs..... 3
1.3.1 Évolution des sources d'énergie..... 3
1.4 Classification des ascenseurs..... 4
1.4.1 Selon le mode de traction..... 4
1.4.2 Selon l'usage..... 8
1.5 Conclusion du Chapitre..... 11
Chapitre 2 : Spécification et Architecture du Système d'Ascenseur Intelligent..... 12
2.1 Introduction du Chapitre..... 13
2.2 Type d'Ascenseur Retenu et Justification du Choix..... 13
2.3 Contexte d'Utilisation et Exigences du Système..... 13
2.3.1 Contexte et Environnement d'Utilisation..... 14
2.3.2 Exigences du Système..... 14
2.3.3 Cahier des Charges Fonctionnel (CdCF)..... 15
2.4 Architecture Fonctionnelle et Choix Technologiques..... 16
2.4.1 Principe d'Entraînement de la Cabine et Choix de la Motorisation..... 16
2.4.2 Système de Détection de Position de la cabine..... 17
2.4.3 Interface Homme-Machine (IHM : Appels et Affichage)..... 18
2.4.4 Mécanisme et Contrôle Automatique des Portes..... 20
2.5 Modélisation Logique et Architecture de Contrôle..... 21
2.5.1 Choix de l'Unité de Contrôle..... 22
2.5.2 Stratégie de Gestion et d'Optimisation du Trafic..... 22
2.6 Conclusion du chapitre..... 24
Chapitre 3 : Réalisation et Validation du Système d'Ascenseur Intelligent..... 25
3.1 Introduction du chapitre..... 26
3.2 Matériel et outils logiciels utilisés..... 26
3.2.1 Éléments mécaniques et structurels..... 26
3.2.2 Éléments électroniques et électriques..... 27
3.2.3 Logiciels et environnement de développement..... 28
3.3 Conception mécanique du Prototype..... 29
3.3.1 Méthodologie de Réalisation..... 29
3.3.2 Résultats et validation :..... 30
3.4 Conception matérielle..... 30

3.4.1 Mise en œuvre de la chaîne d’acquisition des signaux d'entrée.....31
3.4.2 Mise en œuvre des chaînes de commande des périphériques de sortie.....41
3.5 Conception logicielle.....49
3.5.1 Traitement et conditionnement des signaux d'entrée..... 49
3.5.2 Conception du contrôleur principal.....52
3.5.3 Conception des Modules de Commande des Périphériques de Sortie..... 60
3.5.4 Intégration des Modules..... 66
3.6 Résultats Obtenus et Évaluation du Système..... 68
3.6.1 Performances fonctionnelles du système..... 68
3.6.2 Évaluation de la Gestion Intelligente..... 69
3.7 Conclusion du chapitre.....74
Conclusion générale..... 75
Bilan des contributions techniques.....75
Difficultés et contraintes rencontrées..... 75
Perspectives et améliorations futures..... 76
Références bibliographiques..... 77
Annexes..... 78
Annexe A : Description VHDL de la Machine à États (FSM) du Système..... 79
Annexe B : Description VHDL du Module d'Affichage.....83

Liste des Figures

Chapitre 1 : État de l'Art des Ascenseurs.....2
Figure 1.1 : L'innovation d'Elisha Otis (1853) 4
Figure 1.2 : Ascenseur à traction électrique 5
Figure 1.3 : Ascenseur hydraulique 6
Figure 1.4 : Ascenseurs pour personnes 9
Figure 1.5 : Ascenseurs pour marchandises 9
Figure 1.6 : Ascenseurs mixtes 10
Chapitre 2 : Spécification et Architecture du Système d'Ascenseur Intelligent...12
Figure 2.1 : Ascenseur d'un bâtiment à haute densité..... 14
Figure 2.2 : Système d'entraînement du prototype..... 16
Figure 2.3 : Système de détection de la position de la cabine..... 18
Figure 2.4 : Boutons d'appels extérieurs aux paliers..... 19
Figure 2.5 : Boutons de commande intérieurs à la cabine..... 20
Figure 2.6 : Interface de signalisation et d'affichage en temps réel..... 20
Chapitre 3 : Réalisation et Validation du Système d'Ascenseur Intelligent.....25
Figure 3.1 : schéma synoptique global du système..... 31
Figure 3.2 : Emplacement du GPIO_0 dans la carte DE1..... 32
Figure 3.3 : Mise en œuvre des capteurs fin de course..... 33
Figure 3.4 : Réglage des capteurs infrarouges d'évitement d'obstacle..... 35
Figure 3.5 : Vérification du niveau logique de sortie du module infrarouge..... 36
Figure 3.6 : Mise en œuvre des capteurs infrarouges d'évitement d'obstacle..... 37
Figure 3.7 : Mise en œuvre des boutons-poussoirs de la cabine..... 39
Figure 3.8 : Mise en œuvre des boutons-poussoirs d'appel extérieurs..... 40
Figure 3.9 : chaîne de commande du moteur pas à pas de la cabine..... 41
Figure 3.10 : Alimentation du Driver DRV8825..... 42
Figure 3.11 : Configuration des broches SLEEP et RESET..... 43
Figure 3.12 : Mesure et réglage de la tension VREF..... 44
Figure 3.13 : Connexion du moteur pas à pas au driver..... 44
Figure 3.14 : Emplacement du GPIO_1 dans la carte DE1..... 45
Figure 3.15 : Interfaçage du driver avec la carte FPGA (DE1)..... 46
Figure 3.16 : Mise en œuvre du servomoteur de la porte..... 48
Figure 3.17 : schéma RTL du système..... 49
Figure 3.18 : Configuration des résistances de rappel interne sur Assignment Editor..... 50
Figure 3.19 : Chronogramme de simulation du module Debouncer sous ModelSim..... 52
Figure 3.20 : Architecture du FSM du système de contrôle de l'ascenseur..... 59
Figure 3.21 : Chronogramme de simulation du module de commande du moteur pas à pas de la cabine sous ModelSim..... 63
Figure 3.22 : Chronogramme de simulation d'ouverture de la porte sous ModelSim..... 64
Figure 3.23 : Chronogramme de simulation de fermeture de la porte sous ModelSim..... 65

Figure 3.24 : Schéma d'interconnexion des modules du système..... 67
Figure 3.25 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 1)..... 70
Figure 3.26 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 2)..... 71
Figure 3.27 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 3)..... 72
Figure 3.28 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 4)..... 73

Liste des Tableaux et Listing

Chapitre 1 : État de l'Art des Ascenseurs.....2
Tableau 1.1 : Comparaison entre ascenseur à traction électrique et ascenseur hydraulique.....7
Tableau 1.2 : Comparaison entre ascenseurs pour personnes , ascenseurs pour marchandises et ascenseurs mixtes..... 10
Chapitre 2 : Spécification et Architecture du Système d'Ascenseur Intelligent...12
Tableau 2.1 : Comparaison des Logiques de Commande des Ascenseurs.....22
Chapitre 3 : Réalisation et Validation du Système d'Ascenseur Intelligent.....25
Tableau 3.1 : Éléments mécaniques et structurels du système.....26
Tableau 3.2 : Éléments électroniques et électriques du système.....27
Tableau 3.3 : Logiciels et environnement de développement du système..... 29
Listing 3.1 : Description VHDL du détection d'étage courant..... 54
Listing 3.2 : Structure VHDL des registres de mémorisation des requêtes.....56
Listing 3.3 : Description VHDL d'effacement des requêtes.....56
Listing 3.4 : Description VHDL du module de commande du moteur pas à pas de la cabine61

Liste des Acronymes et Abréviations

Acronyme / Abréviation	Signification / Définition
FPGA	Field-Programmable Gate Array (Réseau de portes logiques programmables)
DE1	Digital Electronics Board 1 / Carte de développement électronique numérique
FSM	Finite State Machine (Machine à États Finis)
VHDL	VHSIC Hardware Description Language
PWM	Pulse Width Modulation (Modulation d'Impulsion en Largeur — MLI)
AC	Alternating Current / Courant Alternatif (CA)
DC	Direct Current / Courant Continu (CC)
VFD	Variable Frequency Drive / Variateur de Fréquence (VDF)
MRL	Machine Room-Less / Ascenseur Sans Local Machinerie
ERP	Enterprise Resource Planning / Progiciel de Gestion Intégré (PGI)
R+3	Rez-de-chaussée + 3 étages / Rez-de-chaussée + 3 étages
IHM	Interface Homme-Machine
LOP	Landing Operating Panel / Boîte à Boutons de Palier (BBP)
COP	Car Operating Panel / Boîte à Boutons de Cabine (BBC)
ASIC	Application-Specific Integrated Circuit / Circuit Intégré Spécifique à une Application
SOC	System on Chip / Système sur Puce
API	Automate Programmable Industriel
PLC	Programmable Logic Controller
FCFS	First-Come, First-Served / Premier Arrivé, Premier Servi
IR	Infrared / Infrarouge
GND	Ground / Masse (Terre)
NC	Normally Closed / Normalement Fermé (NF)
NO	Normally Open / Normalement Ouvert (NO)
GPIO	General Purpose Input/Output / Entrées/Sorties Générales
JP	Jumper / Cavalier
RTL	Register-Transfer Level / Niveau de Transfert entre Registres (Niveau RTL)
LED	Light-Emitting Diode / Diode Électroluminescente (DEL)
COM	Common / Point Commun
VREF	Voltage Reference / Tension de Référence

Acronyme / Abréviation	Signification / Définition
IMAX	Maximum Current / Courant Maximal
RSENSE	Sense Resistor / Résistance de Mesure de Courant
VMOT	Motor Power Supply Voltage / Tension d'Alimentation Moteur
SW	Switch / Interrupteur
MHZ	Mégahertz
CLK	Clock Signal / Signal d'Horloge
HEX	Hexadécimal
RFID	Radio-Frequency Identification / Identification par Radiofréquence
IoT	Internet of Things / Internet des Objets (IdO)

Introduction générale

Avec le développement rapide de l'urbanisation et la multiplication des bâtiments à plusieurs étages, les ascenseurs sont devenus des équipements indispensables pour assurer le transport vertical des personnes et des charges **[1, 3]**. Ils jouent un rôle essentiel dans l'amélioration du confort, de l'accessibilité et de la mobilité au sein des immeubles résidentiels, commerciaux, administratifs et industriels. Grâce aux progrès réalisés dans les domaines de l'automatique, de l'électronique numérique et des systèmes embarqués, les ascenseurs modernes offrent aujourd'hui des performances élevées en matière de sécurité, de fiabilité et d'efficacité énergétique **[3, 5]**.

Dans ce contexte, notre projet s'inscrit dans le cadre de l'étude, de la conception et de la réalisation d'un ascenseur intelligent, implémenté sur une carte FPGA DE1 (Altera/Intel Cyclone II). L'utilisation de la technologie FPGA offre une exécution matérielle parallèle, une grande rapidité de traitement et une souplesse de reconfiguration, particulièrement adaptées aux systèmes temps réel critiques. L'objectif principal est de développer un système capable de gérer automatiquement les déplacements de la cabine entre plusieurs niveaux, de traiter les requêtes des utilisateurs, de commander l'ouverture et la fermeture des portes, tout en garantissant un fonctionnement sûr et optimisé grâce à une logique de contrôle modélisée par une machine à états finis FSM et décrite en langage VHDL.

Pour mener à bien ce travail, ce mémoire est structuré en quatre chapitres :

- **Le premier chapitre** présente une étude générale sur les ascenseurs, en abordant leur définition, leur évolution historique, leurs différents types ainsi que les principales technologies utilisées dans les bâtiments modernes.
- **Le deuxième chapitre** est consacré à la spécification, à la conception et à la définition de l'architecture globale du système d'ascenseur intelligent. Il détaille le cahier des charges fonctionnel, le choix des composants (mécaniques, électroniques) et la stratégie d'optimisation du trafic basée sur l'algorithme Collectif-Sélectif.
- **Le troisième chapitre** est dédié à la réalisation pratique, à l'implémentation matérielle et logicielle ainsi qu'à la validation du prototype. Il décrit l'interfaçage des capteurs et actionneurs avec la carte FPGA DE1, la conception VHDL de la logique de contrôle (FSM, anti-rebond, PWM), ainsi que les tests expérimentaux et l'analyse des résultats obtenus.

Enfin, ce mémoire se clôture par une conclusion générale résumant les principales contributions du projet et proposant des perspectives d'amélioration futures pour rapprocher le système des exigences industrielles réelles.

Chapitre 1

État de l’Art des Ascenseurs

1.1 Introduction du chapitre

Les ascenseurs représentent un maillon essentiel des infrastructures urbaines modernes, assurant la mobilité verticale au sein des bâtiments intelligents. Ce premier chapitre a pour objectif d'établir un état de l'art détaillé sur les systèmes d'ascenseurs. Il retrace leur évolution historique, depuis les premiers mécanismes manuels jusqu'aux architectures électromécaniques avancées, tout en présentant une classification rigoureuse basée sur les modes de traction et les domaines d'usage. Cette synthèse permettra d'identifier les défis technologiques actuels et d'orienter nos choix vers la solution la plus adaptée à notre étude.

1.2 Définition des ascenseurs

Un ascenseur est un dispositif de transport vertical conçu pour déplacer des personnes ou des charges entre différents niveaux d'un bâtiment. Il est généralement composé d'une cabine, d'un système de guidage, d'un dispositif de traction ou hydraulique, d'un système de commande et d'éléments de sécurité. Il peut être utilisé pour le transport de passagers, de marchandises ou pour des applications spécifiques comme les ascenseurs hospitaliers ou domestiques **[1]** .

L'ascenseur moderne est un système électromécanique automatisé dont le fonctionnement repose sur la coordination entre la commande, la motorisation, la sécurité et la signalisation. Sa conception doit répondre à des exigences élevées de fiabilité, de confort et d'efficacité énergétique. C'est pourquoi l'ascenseur est aujourd'hui considéré comme un sous-système intelligent du bâtiment **[2]**.

1.3 Historique et évolution des ascenseurs

L'histoire des ascenseurs s'articule autour de deux axes technologiques majeurs : la recherche d'une source d'énergie capable de soulever des charges de plus en plus lourdes, et le développement de mécanismes garantissant une sécurité totale. L'évolution de ces deux aspects a transformé un simple outil de levage en un moyen de transport vertical indispensable à l'urbanisation moderne **[3]**.

1.3.1 Évolution des sources d'énergie

1. **Énergie mécanique humaine et animale :** Exploitation de la force musculaire via des systèmes de poulies et de treuils durant l'Antiquité.
2. **Énergie thermique (Vapeur) :** Introduction des machines à vapeur au début de la Révolution industrielle pour répondre aux besoins d'extraction minière et d'extensibilité des bâtiments.
3. **Énergie hydraulique :** Utilisation de la pression des fluides (eau puis huile) offrant une densité de puissance élevée pour le levage de charges lourdes.
4. **Énergie électrique :** L'arrivée des moteurs électriques (AC/DC) a révolutionné le secteur en apportant une souplesse de commande inédite. Aujourd'hui, cette énergie

est optimisée par des variateurs de fréquence (VFD) et des systèmes de régénération pour réinjecter l'énergie de freinage dans le réseau **[3]**.

1.3.2 Évolution des systèmes de sécurité

1. **L'ère pré-Otis (Avant 1853) :** Les élévateurs étaient strictement réservés aux marchandises ; l'absence de sécurité en cas de rupture de câble les rendait inadaptés au transport humain.
2. **L'innovation d'Elisha Otis (1853) :** L'invention du parachute de sécurité a éliminé le risque de chute libre, rendant le transport de personnes sûr et permettant l'apparition des premiers gratte-ciels (Figure 1.1).

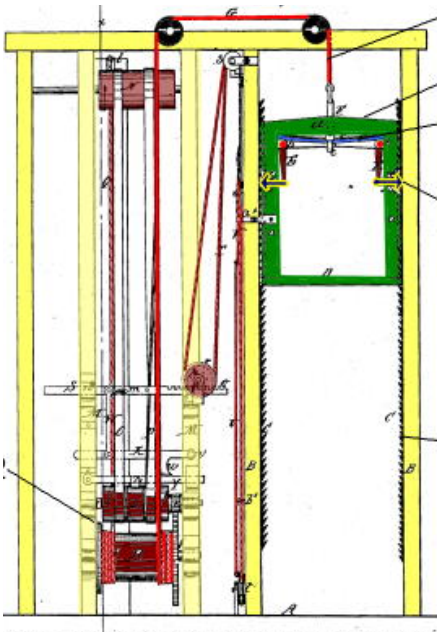


Figure 1.1 : L'innovation d'Elisha Otis (1853) **[8]**

3. **L'enrichissement des systèmes critiques :** L'augmentation des vitesses et des hauteurs a imposé l'introduction successive du régulateur de vitesse, des parachutes progressifs et du verrouillage mécanique des portes **[3]**.

1.4 Classification des ascenseurs

Les ascenseurs peuvent être classés selon plusieurs critères, notamment le mode de traction et l'usage .

1.4.1 Selon le mode de traction

On distingue principalement les ascenseurs à traction et les ascenseurs hydrauliques. Les ascenseurs à traction utilisent des câbles ou des courroies entraînés par une poulie motrice, avec un contrepoids permettant d'équilibrer la charge. Ce système est adapté aux bâtiments de moyenne et grande hauteur et offre une bonne efficacité énergétique. Les ascenseurs hydrauliques, quant à eux, fonctionnent grâce à un vérin actionné par un fluide sous pression. Ils sont généralement utilisés dans les bâtiments de faible hauteur **[5]** .

1.4.1.1 Ascenseur à traction électrique

A. Principe de fonctionnement

L'ascenseur à traction électrique utilise un moteur électrique qui entraîne une poulie motrice reliée à des câbles de traction, déplaçant la cabine et un contrepoids pour équilibrer la charge et optimiser l'énergie. Le variateur de fréquence assure un contrôle précis de la vitesse, avec montée par traction et descente assistée par gravité et freinage régénératif **[11]** .

B. Architecture et composants principaux

Les éléments clés du système de traction incluent un moteur électrique, des câbles de suspension, une poulie motrice et un contrepoids pour équilibrer la charge de la cabine (Figure 1.2).

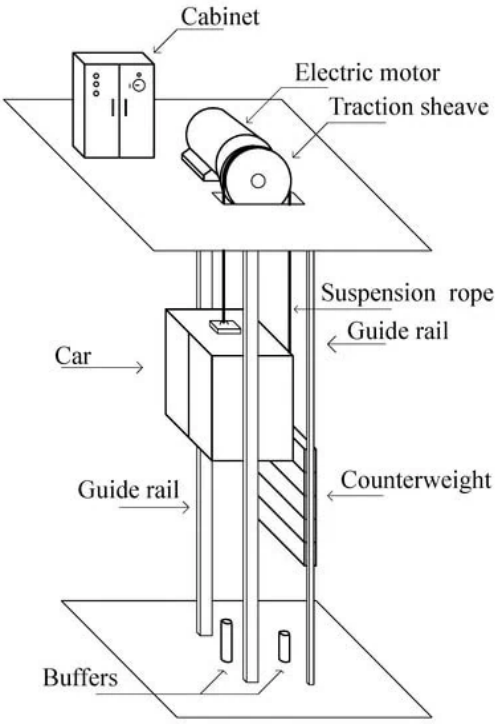


Figure 1.2 : ascenseur à traction électrique **[14]**

C. Avantages et inconvénients

Avantages : Efficacité énergétique élevée grâce au contrepoids et à la récupération d'énergie, vitesses élevées (jusqu'à plus de 5 m/s), fonctionnement silencieux et adapté aux trafics intenses **[11]** .

Inconvénients : Nécessite souvent un local machinerie en hauteur (sauf versions MRL), coût initial potentiellement plus élevé et contraintes structurelles pour les gaines **[11]** .

D. Domaines d'application

Idéal pour les bâtiments de moyenne à grande hauteur comme les bureaux, hôtels, hôpitaux et immeubles tertiaires avec trafic élevé **[11]** .

1.4.1.2 Ascenseur hydraulique

A. Principe de fonctionnement

Ce système repose sur un vérin hydraulique alimenté par un fluide sous pression : une pompe envoie l'huile pour pousser le piston et élever la cabine, tandis que la descente s'effectue par relâchement de pression et gravité. Pas de contrepoids ni de câbles principaux, offrant une stabilité précise **[12]** .

B. Architecture et composants principaux

Les principaux composants sont une centrale hydraulique (pompe, réservoir d'huile, moteur électrique), un ou plusieurs vérins (direct, indirect ou télescopique), des flexibles hydrauliques, un distributeur et une armoire de commande (Figure 1.3).

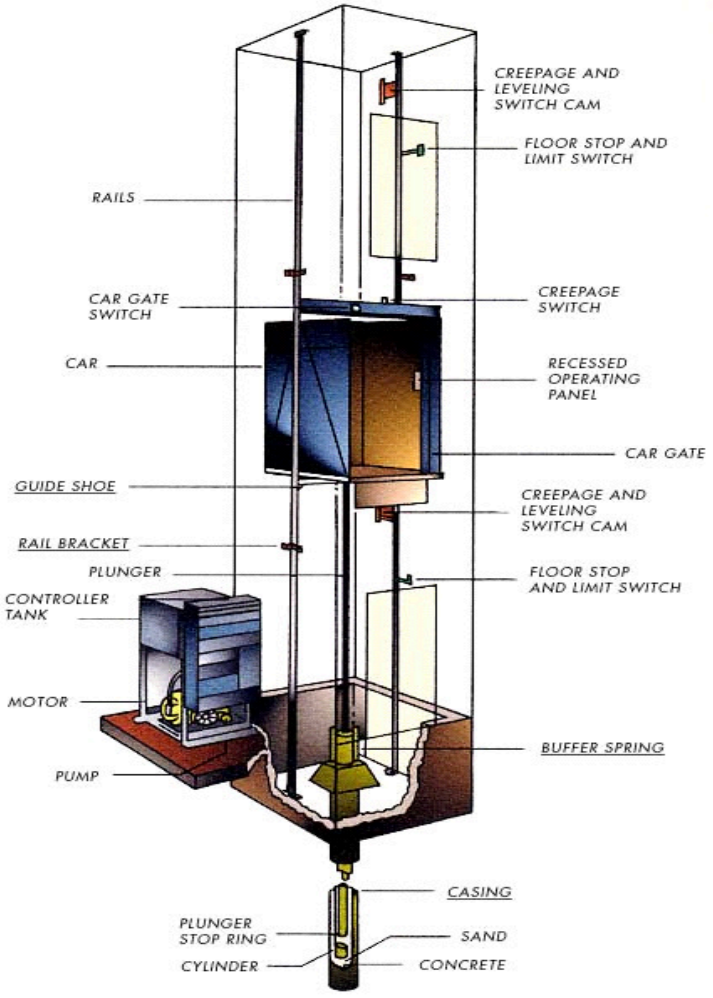


Figure 1.3 : Ascenseur hydraulique **[15]**

C. Avantages et inconvénients

Avantages : Installation flexible sans salle des machines en hauteur, coût initial inférieur (jusqu'à 15% de moins), fonctionnement silencieux, grande stabilité et adapté aux rénovations **[12]** .

Inconvénients : Consommation énergétique plus élevée, vitesse limitée (0,6 à 1 m/s), risque de fuite de fluide et entretien régulier du circuit hydraulique **[12]** .

D. Domaines d'application

Convient aux immeubles bas (2 à 6 étages), rénovations de bâtiments anciens, copropriétés et ERP comme écoles ou cliniques, où les contraintes structurelles sont fortes **[12]** .

1.4.1.3 Synthèse comparative

Afin de mettre en évidence les spécificités de chaque technologie, le tableau suivant (Tableau 1.1) résume les principales différences entre les ascenseurs électriques et hydrauliques :

Tableau 1.1 : Comparaison entre ascenseur à traction électrique et ascenseur hydraulique

Critère	**Ascenseurs à traction électrique**	**Ascenseurs hydrauliques**
Vitesse	1 à 3,5 m/s ou plus ; adaptée aux hautes vitesses pour immeubles multi-étages.	0,15 à 0,63 m/s ; limitée pour courtes distances.
Hauteur max.	Illimitée (jusqu'à >20 étages) ; idéale pour bâtiments hauts grâce au contrepoids.	2 à 6 étages (15-18 m) ; non viable au-delà en raison du flambage des cylindres.
Coût d'installation	Élevé (ex. 34k€ pour cas haut) ; nécessite souvent grue et salle machines pour geared.	Bas (ex. 12-18k€) ; installation rapide, moins de composants, adaptable aux rénovations.
Consommation énergétique	Faible (2,5-3x moins que hydraulique) ; contrepoids équilibre la charge, gearless optimal.	Élevée (jusqu'à 4x plus) ; pompe contre gravité, pas de contrepoids.
Maintenance	Moyenne à élevée ; câbles, freins, contrepoids à inspecter fréquemment (4x/an en commercial).	Faible ; moins de pièces mobiles, mais huile à changer (risque pollution).

Critère	Ascenseurs à traction électrique	Ascenseurs hydrauliques
Espace requis	Fosse/plafond plus grands ; contrepoids et poulies occupent volume (MRL possible).	Compact ; fosse/plafond réduits, machine près shaft.
Sécurité /Fiabilité	Bonne ; parachute, limiteur de vitesse ; sensible séismes (11% dommages).	Excellente ; descente auto en panne, résistant séismes (1% dommages).
Confort (bruit/vibrations)	Silencieux (gearless) ; vibrations minimes avec variateurs.	Très doux ; démarrage fluide sans câbles.
Applications typiques	Bâtiments >6 étages, bureaux, hôpitaux ; charges standard, trafic élevé.	Bâtiments bas <6 étages, charges lourdes (monte-charges), rénovations.
Impact environnemental	Faible ; recyclable, éco-énergétique.	Moyen ; risque fuite huile, mais fluides biodégradables disponibles.

1.4.2 Selon l'usage

Les ascenseurs se classent en trois types principaux selon leur usage : pour personnes, pour marchandises et mixtes, chacun conçu pour répondre à des besoins spécifiques en termes de charge, vitesse et sécurité. Cette classification permet d'optimiser le transport vertical dans divers environnements comme les bâtiments résidentiels, commerciaux ou industriels **[17]** .

1.4.2.1 Ascenseurs pour personnes

Ces ascenseurs sont dédiés au transport exclusif de passagers, en mettant l'accent sur le confort et la rapidité (Figure 1.4).



Figure 1.4 : Ascenseurs pour personnes **[19].**

Utilités : Déplacement fluide dans les immeubles résidentiels, bureaux, hôtels ou centres commerciaux **[18]** .

Caractéristiques :

- Vitesse modérée (1-7 m/s).
- Cabines spacieuses et esthétiques.
- Portes rapides.
- Systèmes silencieux et bien ventilés **[18]** .

1.4.2.2 Ascenseurs pour marchandises

Ils sont conçus pour transporter des charges lourdes sans passagers, avec une priorité sur la robustesse **(**figure 1.4).

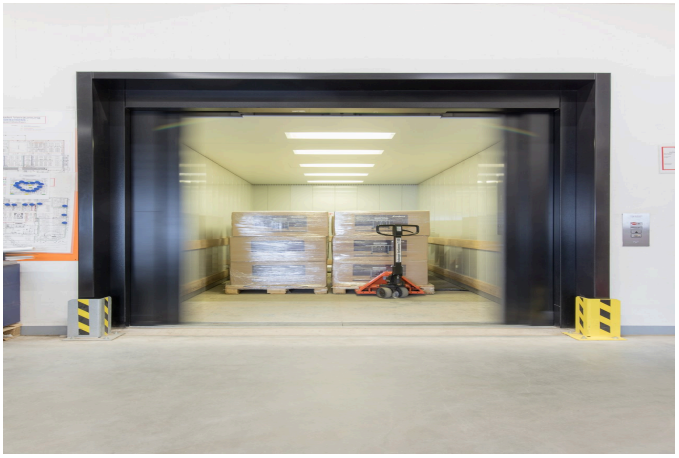


Figure 1.5 : Ascenseurs pour marchandises **[20].**

Utilités : Déplacement de fret comme chariots, palettes ou véhicules dans les entrepôts, usines ou garages **[17]** .

Caractéristiques :

- Charge élevée (jusqu'à 5000 kg).
- Cabines profondes et renforcées.

- Portes coulissantes larges.
- Vitesse lente **[17]** .

Exemples : Monte-charges, monte-voitures **[17]**.

1.4.2.3 Ascenseurs mixtes

Polyvalents, ils combinent transport de personnes et de marchandises légères pour un usage partagé **(**figure 1.6).



Figure 1.6 : Ascenseurs mixtes **[21]**

Utilités : Service mixte dans les hôpitaux, hôtels ou centres commerciaux pour lits, matériel et personnel **[17]** .

Caractéristiques :

- Charge moyenne (1000-2500 kg).
- Structure renforcée.
- Facile à nettoyer.
- Portes adaptées **[17]** .

Exemples : Ascenseurs de service hôteliers, monte-malades hospitaliers **[17]** .

1.4.2.4 Synthèse comparative

Le tableau suivant (Tableau 1.2) présente une étude comparative des trois catégories d'ascenseurs selon leur usage : les ascenseurs pour personnes, les ascenseurs pour marchandises et les ascenseurs mixtes :

Tableau 1.2 : Comparaison entre ascenseurs pour personnes , ascenseurs pour marchandises et ascenseurs mixtes

Critère	Ascenseurs pour personnes	Ascenseurs pour marchandises	Ascenseurs mixtes
Charge	Faible à moyenne (450-1600 kg)	Élevée (jusqu'à 5000 kg)	Moyenne (1000-2500 kg)
Vitesse	Modérée (1-7 m/s)	Lente (0,15-1 m/s)	Moyenne (0,5-2,5 m/s)
Cabine	Spacieuse, esthétique, bien ventilée	Profonde, renforcée, portes larges	Renforcée, facile à nettoyer, portes adaptées
Confort	Élevé (silencieux, portes rapides, design)	Bas (priorité robustesse, vibrations possibles)	Moyen (équilibre personnes/fret)
Exemples	Résidentiels, panoramiques	Monte-charges, monte-voitures	Service hôteliers, monte-malades

1.5 Conclusion du Chapitre

En conclusion, ce chapitre nous a permis d'acquérir une vue d'ensemble sur l'évolution technologique des ascenseurs et de comparer les principales technologies du marché. Il en ressort que l'ascenseur électrique à traction offre le meilleur compromis en termes de rapidité, de rendement énergétique et d'adaptabilité aux systèmes de commande intelligents. Cependant, pour garantir des performances optimales et une sécurité accrue dans un bâtiment moderne, il est nécessaire de définir précisément l'architecture et la stratégie de contrôle du système. C'est ainsi que le chapitre suivant sera consacré à la spécification, au cahier des charges fonctionnel et à l'architecture globale d'un ascenseur intelligent à 4 niveaux.

Chapitre 2

Spécification et Architecture du Système d'Ascenseur Intelligent

2.1 Introduction du Chapitre

S'appuyant sur l'étude théorique et comparative menée dans le chapitre précédent, ce deuxième chapitre est dédié à la conception théorique et à la spécification matérielle de notre système d'ascenseur intelligent (R+3). Il définit en premier lieu le cahier des charges fonctionnel, en précisant les contraintes de sécurité, d'ergonomie et de gestion du trafic. Dans un second temps, nous détaillons le choix des composants (actionneurs, capteurs, interfaces) et nous justifions l'utilisation d'une plateforme reconfigurable FPGA associée à l'algorithme d'arbitrage Collectif-Sélectif. L'ensemble de ces choix permettra d'établir une architecture solide avant de passer à la phase de réalisation.

2.2 Type d'Ascenseur Retenu et Justification du Choix

Pour l'élaboration et la conception de notre modèle de référence, notre choix s'est porté sur un **ascenseur électrique pour personnes desservant quatre niveaux (R+3)**. Ce choix stratégique repose sur un ensemble d'arguments clés, combinant la pertinence fonctionnelle dans les applications réelles et la faisabilité matérielle à l'échelle d'un prototype d'expérimentation :

1. **Pertinence fonctionnelle et complexité du trafic :** La majorité des ascenseurs destinés au transport de personnes sont installés au cœur de complexes résidentiels et d'immeubles tertiaires. Dans ces environnements caractérisés par un flux d'utilisateurs important, le système est confronté à des sollicitations multiples et simultanées. Cela exige une gestion intelligente et optimisée des requêtes afin de réduire les temps d'attente et rationaliser les déplacements, représentant ainsi un cas d'étude idéal pour la commande avancée.
2. **Faisabilité technique et mise en œuvre sur maquette :** Sur le plan pratique, l'architecture d'un ascenseur électrique à câble offre l'avantage majeur de permettre une implémentation réelle, fidèle et accessible sur un prototype d'expérimentation (maquette). Cette technologie constitue le support idéal pour matérialiser physiquement le fonctionnement mécanique et établir une interconnexion directe entre l'unité de contrôle et la partie opérative.
3. **Optimisation du nombre d'étages (4 niveaux) :** Le choix d'une configuration à quatre étages représente le compromis parfait entre complexité et faisabilité :
    - **Du point de vue algorithmique :** Une structure à quatre étages offre un degré de complexité suffisant pour mettre clairement en évidence l'efficacité d'une logique de commande intelligente (arbitrage des appels, optimisation du sens de déplacement).
    - **Du point de vue réalisation :** Il s'agit du nombre optimal qui allie la démonstration concrète d'un contrôle complexe à la possibilité pratique de sa validation expérimentale sur maquette sans surcharge matérielle inutile.

2.3 Contexte d'Utilisation et Exigences du Système

Avant de procéder à la conception détaillée de notre modèle de référence, il est essentiel d'établir une analyse rigoureuse du système cible. Cette étape vise à définir le cadre d'utilisation, les exigences fonctionnelles ainsi que les contraintes opérationnelles auxquelles doit répondre un ascenseur intelligent moderne. L'ensemble de ces spécifications constitue le cahier des charges fonctionnel orientant l'architecture de notre étude.

2.3.1 Contexte et Environnement d'Utilisation

Le système considéré dans cette étude est un ascenseur intelligent destiné au transport vertical de personnes au sein de bâtiments à moyenne et haute densité (résidentiels, administratifs ou tertiaires) comme illustré dans la figure 2.1.

Dans ces environnements urbains, l'ascenseur constitue un sous-système critique de la gestion technique du bâtiment . Évoluant au cœur de structures caractérisées par un flux continu et important d'usagers, le système doit faire face à des sollicitations aléatoires et simultanées provenant de différents niveaux. L'enjeu principal réside donc dans sa capacité à gérer efficacement le trafic, optimiser les déplacements de la cabine et réduire significativement les temps d'attente et de trajet des passagers.



Figure 2.1 : Ascenseur d'un bâtiment à haute densité.

2.3.2 Exigences du Système

Pour répondre aux contraintes opérationnelles et aux attentes des utilisateurs, le système doit satisfaire un ensemble d'exigences techniques et fonctionnelles réparties selon quatre axes majeurs :

1. **Gestion Intelligente et Équitable du Trafic :**
    - **Prise en charge globale :** Enregistrement et traitement de l'ensemble des requêtes (appels extérieurs aux paliers et commandes intérieures en cabine).
    - **Optimisation temporelle :** Réduction des temps d'attente et de transit grâce à une planification efficace des déplacements.

- **Équité de service :** Traitement juste des demandes empêchant tout phénomène d'accaparement du service par certains étages ou groupes d'utilisateurs.

2. **Sécurité et Contrôle du Mouvement :**
   - **Précision d'arrêt :** Alignement rigoureux de la cabine au niveau de chaque étage pour garantir un franchissement sécurisé.
   - **Sécurité des accès :** Verrouillage strict du mouvement de la cabine lors des phases d'embarquement/débarquement et interdiction d'ouverture des portes pendant le déplacement.
   - **Gestion des urgences :** Capacité d'interruption immédiate en cas de signal d'arrêt d'urgence ou de défaillance critique.
3. **Automatisme des Portes et Confort :**
   - **Manoeuvre automatisée :** Intégration d'un mécanisme de porte entièrement automatisé et synchronisé avec l'état de la cabine.
   - **Confort de déplacement :** Mouvements fluides sans à-coups lors des phases d'accélération, de décélération et de manœuvre des portes.
   - **Temporisation adaptative :** Maintien de l'ouverture des portes pendant une durée optimale, ajustable manuellement par les usagers.
4. **Interface Homme-Machine (IHM) et Ergonomie :**
   - **Information en temps réel :** Affichage clair et dynamique fournissant les données essentielles .
   - **Commande intuitive :** Interface à boutons ergonomique, simple et accessible à tout type d'utilisateur.

2.3.3 Cahier des Charges Fonctionnel (CdCF)

Le cahier des charges fonctionnel récapitule les fonctions de service que le système d'ascenseur doit impérativement assurer :

- **Fonction Principale (FP) :**
  - **FP1 :** Transporter des personnes entre les différents niveaux d'un bâtiment en répondant de manière optimale et rapide à leurs requêtes.

- **Fonctions Complémentaires et de Contrainte (FC) :**
  - **FC1 :** Acquérir et enregistrer l'ensemble des appels (intérieurs et extérieurs) sans omission ni discrimination.
  - **FC2 :** Traiter les requêtes via un algorithme d'arbitrage intelligent afin de minimiser le temps d'attente global.
  - **FC3 :** Garantir la sécurité physique absolue des passagers (contrôle des portes, arrêts d'urgence et verrouillages matériels).
  - **FC4 :** Informer continuellement l'utilisateur sur l'état du système via une IHM claire et intuitive.
  - **FC5 :** Maintenir un fonctionnement stable, robuste et tolérant aux fausses manœuvres ou perturbations.

2.4 Architecture Fonctionnelle et Choix Technologiques

L'architecture fonctionnelle constitue le pont entre les exigences théoriques du cahier des charges et la réalité physique du système. Cette section définit l'organisation structurale de l'ascenseur en détaillant le principe de fonctionnement de chaque sous-système mécanique, électromécanique et d'interface, tout en rationalisant les choix technologiques retenus pour la réalisation du prototype d'expérimentation.

2.4.1 Principe d'Entraînement de la Cabine et Choix de la Motorisation

Le système d'entraînement constitue le cœur cinématique de l'ascenseur. Il a pour rôle d'assurer la conversion de l'énergie électrique en mouvement mécanique de translation verticale pour déplacer la cabine entre les différents niveaux.

2.4.1.1 Principe d'entraînement et guidage mécanique

Pour notre modèle de référence, nous avons adopté le principe d'**entraînement électrique par adhérence (poulie et câble)** comme illustré dans la figure 2.2.

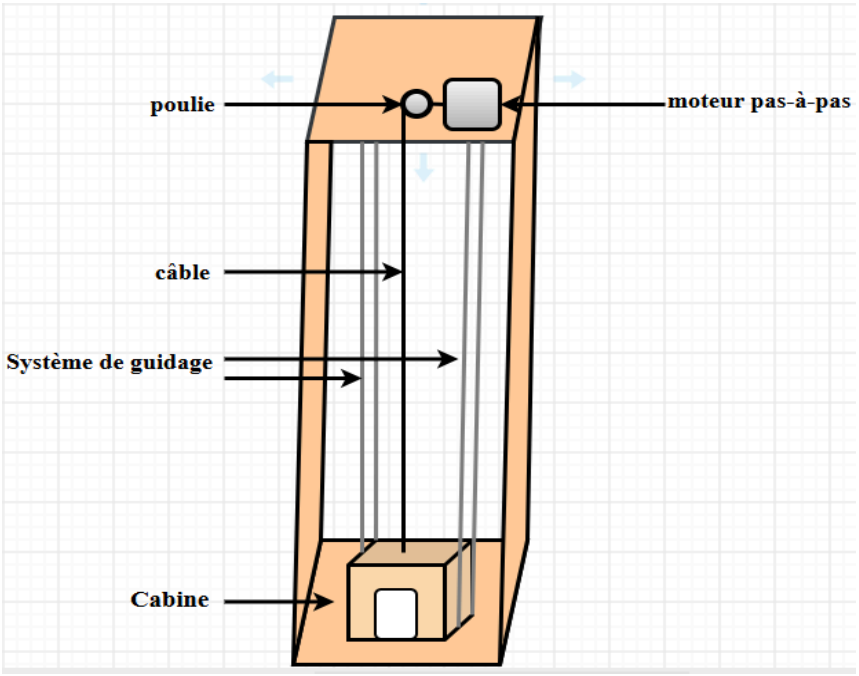


Figure 2.2 : système d'entraînement du prototype.

- **Suspension et traction :** La cabine est suspendue à un câble de traction entraîné par une poulie gorge solidaire de l'arbre du moteur. Le mouvement vertical est transmis par l'adhérence et le reniement du câble sur la poulie.
- **Système de guidage (Guide Rails) :** Afin de garantir une trajectoire rigoureusement verticale, d'éviter les oscillations transversales et de maintenir la stabilité mécanique lors des déplacements, la cabine est guidée le long du puit d'ascenseur par des rails de guidage (*guide rails*).

Remarque concernant le contrepoids sur le prototype : Dans les installations réelles, un contrepoids est systématiquement associé à la cabine pour équilibrer la charge massique et réduire le couple moteur nécessaire. Toutefois, à l'échelle de notre prototype d'expérimentation, le contrepoids a été volontairement omis. Ce choix de conception simplifie la structure mécanique tout en restant parfaitement fidèle à la dynamique de commande : le couple résistant faible de la cabine miniature est aisément pris en charge par la motorisation retenue, sans altérer la logique de contrôle, le principe d'entraînement par câble, ni la validité des scénarios d'essai.

2.4.1.2 Choix et justification de la motorisation

Pour assurer la traction de la cabine sur le prototype, le choix s'est porté sur un **moteur pas-à-pas (Stepper Motor)** :

- **Précision de positionnement :** Ce type de moteur permet un contrôle en boucle ouverte extrêmement précis du déplacement angulaire. Chaque impulsion électrique correspond à un pas discret, garantissant un contrôle rigoureux de la vitesse et un arrêt précis au niveau des étages.
- **Maintien à l'arrêt (Couple de maintien) :** Le moteur pas-à-pas développe un couple de maintien (*holding torque*) important à l'arrêt, ce qui permet de bloquer la cabine à l'étage désiré sans nécessiter l'intégration d'un frein électromécanique complexe sur le prototype.

2.4.2 Système de Détection de Position de la cabine

La connaissance en temps réel de la position exacte de la cabine au sein du puit d'ascenseur est une condition préalable indispensable au bon fonctionnement du système de commande. Elle conditionne la précision de l'arrêt aux paliers, la sécurité des usagers et la validation des transitions logiques.

2.4.2.1 Principe de détection et d'alignement au niveau

Pour assurer la localisation de la cabine le long du puit d'ascenseur, un capteur est positionné au niveau de chaque étage.

- **Mécanisme d'actionnement (Ergot arrière) :** Un petit élément physique ergot est fixé à l'arrière de la cabine. Lors du déplacement vertical, cet élément vient interagir directement avec le capteur de l'étage visé (figure 2.3).

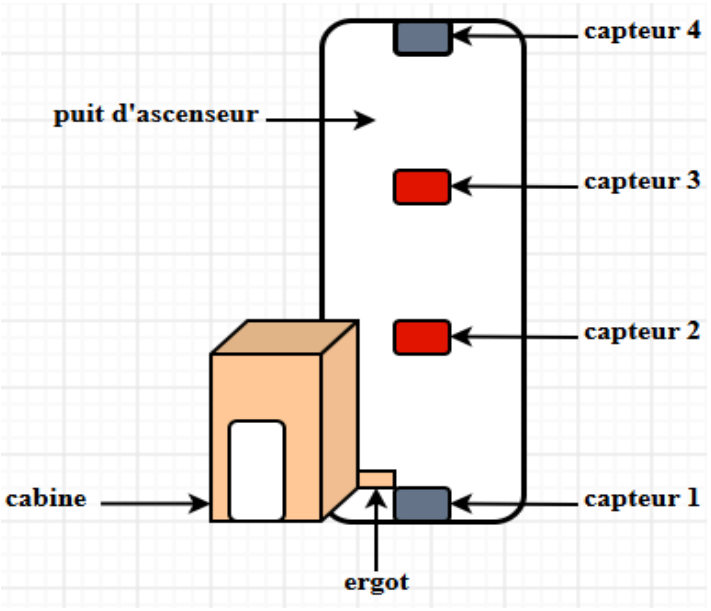


Figure 2.3 : Système de détection de la position de la cabine

- **Précision d'arrêt :** Ce dispositif garantit un déclenchement net et un arrêt rigoureusement précis et stable de la cabine en face du palier, éliminant tout décalage que ce soit lors des phases de montée ou de descente.

2.4.2.2 Répartition et choix des capteurs

Afin d'optimiser le suivi du positionnement et d'assurer la sécurité du système, deux types de capteurs sont combinés :

- **Capteurs de fin de course aux étages extrêmes :** Placés au niveau le plus bas (Étage 0) et le plus haut (Étage 3), ces capteurs mécaniques assurent la détection de la position de la cabine tout en servant de sécurité matérielle pour éviter tout dépassement de course (*over-travel*).
- **Capteurs optiques aux étages intermédiaires :** Installés au niveau des étages centraux (Étage 1 et Étage 2), ils permettent une détection sans contact mécanique direct avec l'ergot arrière, garantissant une commutation rapide, une usure nulle et une réponse logique instantanée lors du passage de la cabine.

2.4.3 Interface Homme-Machine (IHM : Appels et Affichage)

L'Interface Homme-Machine (IHM) constitue le canal d'interaction directe entre les usagers et le système d'ascenseur. Elle a pour rôle d'acquérir les consignes d'appel émanant des passagers et de fournir un retour visuel clair et en temps réel sur l'état opérationnel du système.

2.4.3.1 Interface d'acquisition des demandes

Pour garantir une utilisation simple et intuitive tout en s'adaptant à la logique de commande intelligente retenue, l'acquisition des demandes s'effectue via des **boutons poussoirs** . L'interface de commande est structurée en deux sous-ensembles distincts :

- **Boutons d'appels extérieurs aux paliers (LOP - Landing Operation Panel) :** Installés à chaque étage, ils permettent aux usagers d'appeler la cabine tout en précisant à l'avance leur direction souhaitée (figure 2.4) :

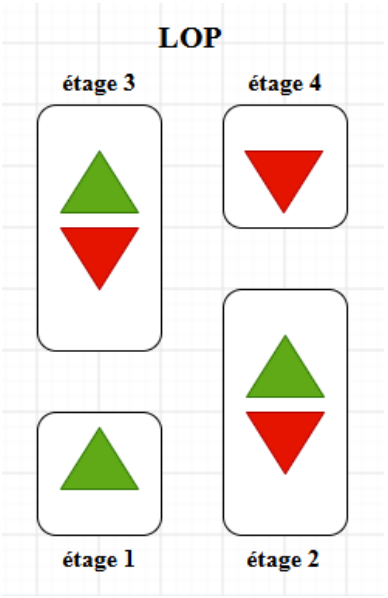


Figure 2.4 : Boutons d'appels extérieurs aux paliers

  - **Étages intermédiaires (Étages 2 et 3) :** Chaque niveau dispose de **deux boutons poussoirs d'appel** (un bouton "Montée" et un bouton "Descente"). Cette distinction est indispensable au fonctionnement de l'algorithme d'arbitrage intelligent pour optimiser les arrêts en cours de trajet.
  - **Étages extrêmes (Étages 1 et 4) :** Le niveau inférieur (Étage 1) ne comporte qu'un bouton poussoir "Montée", tandis que le niveau supérieur (Étage 4) ne comporte qu'un bouton poussoir "Descente".

- **Boutons de commande intérieurs à la cabine (COP - Car Operation Panel) :** Situés à l'intérieur de la cabine, ils permettent aux passagers d'interagir directement avec l'ascenseur une fois à bord (figure 2.5) :

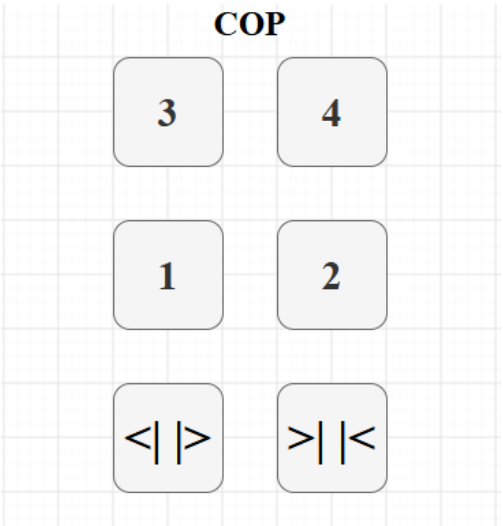


Figure 2.5 : Boutons de commande intérieurs à la cabine

- **Boutons de destination :** Un bouton poussoir dédié pour chaque niveau (**4 boutons au total** pour les étages 1, 2, 3 et 4) afin de sélectionner le niveau cible.
- **Boutons de contrôle des portes :** Deux boutons poussoirs complémentaires permettant un contrôle manuel sécurisé des portes dans les phases autorisées : un bouton pour forcer l'ouverture prolongée et un bouton pour ordonner la fermeture immédiate .

2.4.3.2 Interface de signalisation et d'affichage en temps réel

Afin de fournir une information continue et transparente sur l'état du système, l'interface visuelle repose sur quatre afficheurs à 7 segments (figure 2.6) :

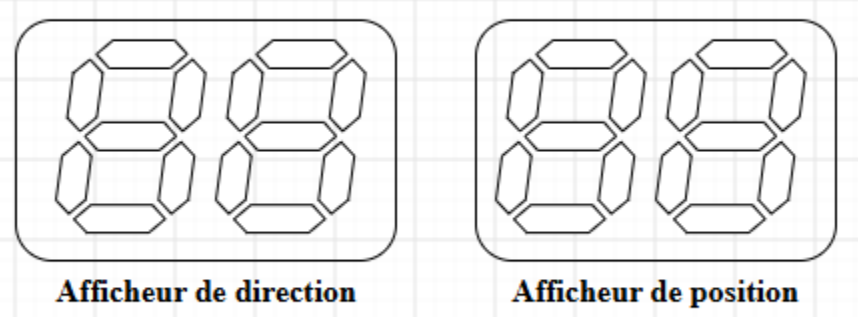


Figure 2.6 : Interface de signalisation et d'affichage en temps réel

- **Afficheur de position (Étage courant) :** Dédié à l'affichage numérique dynamique de l'étage où se trouve actuellement la cabine (0, 1, 2 ou 3).
- **Afficheur de direction (Sens de déplacement) :** Dédié à l'affichage en temps réel de l'état de mouvement de la cabine, indiquant la direction actuelle (*Montée, Descente ou Arrêt*) à l'aide de symboles ou de segments spécifiques.

2.4.4 Mécanisme et Contrôle Automatique des Portes

Le mécanisme de porte constitue un sous-système essentiel garantissant la sécurité des passagers lors des phases d'accès (embarquement et débarquement) et assurant l'isolation de la cabine durant les phases de déplacement vertical.

2.4.4.1 Actionneur et contrôle du mouvement

Pour la manœuvre automatisée des portes sur le prototype, le choix s'est porté sur un **servomoteur** :

- **Asservissement en position (Feedback intégré) :** Ce type d'actionneur intègre un capteur de position interne (potentiomètre) offrant un retour d'information (*feedback*) en boucle fermée. Cela garantit un positionnement angulaire d'une extrême précision lors des phases d'ouverture totale et de fermeture complète .

- **Contrôle du profil de vitesse (Confort et sécurité) :** La commande du servomoteur est configurée pour générer des profils de mouvement fluides, intégrant des rampes d'accélération et de décélération progressives. Cette approche évite tout choc ou choc mécanique violent lors de la manœuvre des portes.

2.4.4.2 Scénario fonctionnel et règles de temporisation

La logique de fonctionnement des portes suit une séquence chronologique stricte assurant une sécurité maximale :

- **Verrouillage en déplacement :** Il est rigoureusement impossible d'ordonner l'ouverture des portes lorsque la cabine est en mouvement (phases de montée ou de descente). Les portes restent fermées et verrouillées.
- **Stabilisation préalable à l'ouverture :** Lors de l'arrivée de la cabine à l'étage visé, une temporisation de sécurité d'**une seconde (1 s)** est observée après l'arrêt complet du moteur de traction. Cette phase de stabilisation garantit l'immobilité totale de la cabine avant d'autoriser l'ouverture des portes.
- **Ouverture et temporisation automatique (Maintien des accès) :** Une fois la cabine stabilisée, l'ouverture automatique s'exécute. Les portes restent entièrement ouvertes pendant une durée fixe de **six secondes (6 s)**, représentant un intervalle temporel optimal pour permettre l'embarquement et le débarquement des passagers en toute sécurité.
- **Fermeture automatique :** À l'expiration du délai de six secondes, la séquence de fermeture automatique s'enclenche de manière fluide.

2.4.4.3 Interactions et commandes manuelles (Overrides)

Afin d'offrir plus de souplesse aux usagers, la séquence automatique peut être modifiée à tout moment via les boutons poussoirs de la cabine (COP) :

- **Fermeture anticipée :** L'utilisateur peut interrompre le délai d'attente de six secondes et ordonner la fermeture immédiate de la porte en appuyant sur le bouton poussoir de fermeture .
- **Réouverture d'urgence/Sûreté :** Si un passager se présente pendant la phase de fermeture de la porte, un appui sur le bouton poussoir d'ouverture provoque l'interruption immédiate de la fermeture et relance une séquence d'ouverture complète.

2.5 Modélisation Logique et Architecture de Contrôle

La conception d'un système d'ascenseur intelligent repose sur une synergie étroite entre la puissance de calcul de l'unité de commande et l'efficacité de l'algorithme d'arbitrage des requêtes. Cette section présente la modélisation logique du système en justifiant le choix de la plateforme matérielle de traitement et en détaillant la stratégie d'optimisation du trafic .

2.5.1 Choix de l'Unité de Contrôle

Pour la conception de notre modèle de référence, la commande est confiée à une plateforme matérielle reconfigurable de type **FPGA (Field Programmable Gate Array)**.

- **Parallélisme natif et exécution temps réel :** Contrairement aux microcontrôleurs séquentiels classiques, le FPGA permet une exécution entièrement parallèle des tâches logiques (lecture simultanée des capteurs, gestion des timer, commande PWM des moteurs et mise à jour de l'affichage).
- **Technologie industrielle moderne :** L'utilisation des FPGA et des circuits logiques programmables dédiés (ASIC/SoC) représente la tendance technologique actuelle adoptée par les grands constructeurs d'ascenseurs pour concevoir des contrôleurs robustes, sécurisés et évolutifs.
- **Déterminisme et réactivité :** Les temps de réponse logiques sont de l'ordre de la nanoseconde, garantissant un traitement instantané des évènements critiques (appels, arrêt d'urgence, capteurs de niveau).

2.5.2 Stratégie de Gestion et d'Optimisation du Trafic

Pour la gestion des requêtes, notre choix s'est porté sur l'algorithme **Collectif-Sélectif (Collective-Selective Directional Control / LOOK Algorithm)**. Cette stratégie d'arbitrage moderne permet d'éviter les déplacements incohérents ou irrationnels de la cabine tout en réduisant considérablement la consommation d'énergie et les temps d'attente.

2.5.2.1 Avantages de l'approche Collective-Sélective :

- **Optimisation temporelle et énergétique :** Elle élimine les trajets inutiles et les changements de direction répétés en regroupant les requêtes selon leur sens de déplacement.
- **Équité de service (Anti-famine) :** L'algorithme garantit le traitement de l'intégralité des appels enregistrés, évitant ainsi qu'un utilisateur ou un étage donné ne soit indéfiniment ignoré ou que la cabine ne reste bloquée entre deux niveaux.

Le tableau ci-dessous (Tableau 2.1) met en évidence la supériorité de la logique Collective-Sélective face aux stratégies classiques :

Tableau 2.1 : Comparaison des Logiques de Commande des Ascenseurs

	Principe de Traitement	**Temps d'Attente Moyen**	**Consommation d'Énergie**	**Gestion des Priorités**
Logique Premier Arrivé, Premier Servi (FCFS)	Traite les appels dans l'ordre chronologique	Très élevé en cas de fort trafic	Élevée (Allers-retours fréquents)	Aucune

	Principe de Traitement	**Temps d'Attente Moyen**	**Consommation d'Énergie**	**Gestion des Priorités**
Logique Collective Simple (Non sélective)	Sert tous les appels sur le passage sans distinction de sens	Moyen	Modérée	Partielle
Logique Collective-Sélective	Sert uniquement les appels allant dans le sens courant	Minimal et optimisé	Optimisée (Trajets regroupés par direction)	Intelligente et dynamique

2. Fonctionnement et dynamique de l'algorithme :

Le fonctionnement de la logique collective-sélective s'articule autour des règles suivantes :

1. **Priorité au sens de déplacement courant :** Tant que la cabine se déplace dans une direction (ex: Montée), l'algorithme traite en priorité tous les appels intérieurs (COP) et extérieurs (LOP) demandant la même direction (*Montée*) situés sur son parcours.
2. **Inversion de sens conditionnelle :** La cabine ne change de direction que lorsqu'il n'existe plus aucune requête active (intérieure ou extérieure) dans le sens de déplacement actuel.
3. **Traitement stratégique des requêtes inverses (Optimisation de portée) :** Lors de l'inversion de sens, s'il existe plusieurs appels dans la direction opposée, le système se déplace directement vers **l'appel le plus éloigné** (*the farthest call*) dans cette direction. Cette stratégie permet de prendre en charge l'ensemble du flux de retour en un seul balayage descendant ou montant, évitant les blocages intermédiaires et maximisant l'efficacité globale du trafic.
4. **Acquisition et mise à jour ultra-rapide (Synergie avec le FPGA) :** Grâce à l'architecture matérielle du FPGA, la matrice des appels (LOP et COP) est scrutée et mise à jour en continu à chaque cycle d'horloge. Cette réactivité permet d'intégrer dynamiquement de nouvelles requêtes en temps réel sans interrompre l'exécution de l'algorithme.

2.6 Conclusion du chapitre

Ce chapitre de spécification a permis de structurer la partie matérielle et la logique de commande du système d'ascenseur intelligent. La combinaison d'une technologie FPGA pour un traitement parallèle en temps réel et de la stratégie Collectif-Sélectif garantit une gestion optimisée, équitable et sécurisée des requêtes. L'architecture théorique étant parfaitement définie et modélisée, la phase suivante consiste à matérialiser ces concepts. Le chapitre 3 présentera ainsi la mise en œuvre pratique sur le prototype physique, la description VHDL des modules logiciels, ainsi que les résultats des tests expérimentaux.

Chapitre 3

Réalisation et Validation du Système d'Ascenseur Intelligent

3.1 Introduction du chapitre

Après avoir établi l'état de l'art et défini l'architecture théorique du système dans les chapitres précédents, ce troisième chapitre concrétise notre étude à travers la réalisation matérielle, l'implémentation logicielle et la validation expérimentale du prototype. Il décrit d'abord l'assemblage mécanique du châssis et l'interfaçage électrique des capteurs et actionneurs avec la carte FPGA DE1. Ensuite, il détaille la conception logicielle sous Quartus II en langage VHDL, incluant les filtres anti-rebond et la FSM de contrôle. Enfin, une évaluation expérimentale à travers divers scénarios de trafic réels viendra valider les performances fonctionnelles et la réactivité du système.

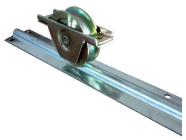
3.2 Matériel et outils logiciels utilisés

La réalisation pratique du prototype repose sur un ensemble de moyens matériels et logiciels. Cette section détaille la structure mécanique, les composants électroniques ainsi que l'environnement logiciel ayant permis la conception et la validation de la logique de commande sur la carte FPGA.

3.2.1 Éléments mécaniques et structurels

Le tableau suivant (Tableau 3.1) récapitule l'ensemble des éléments mécaniques et structurels constituant l'ossature physique et le système de transmission du prototype :

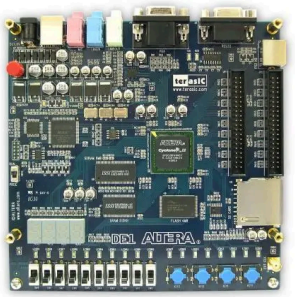
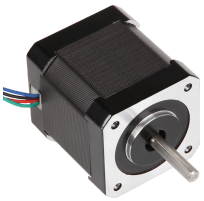
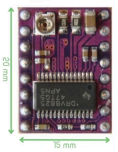
Tableau 3.1 : Éléments mécaniques et structurels du système

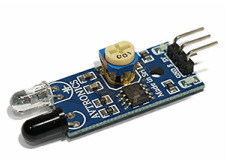
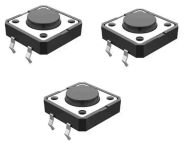
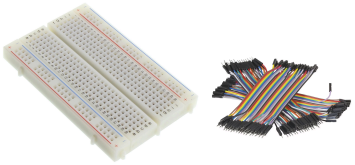
Éléments	Illustration	Description et Rôle
Structure globale (Châssis)		Réalisée en bois, elle constitue le support mécanique du prototype. Elle intègre la base, le puits d'ascenseur (gaine), les différents niveaux (étages) ainsi que le local machine (chambre des machines) situé en partie supérieure.
Cabine d'ascenseur		Structure mobile réalisée en bois, conçue pour simuler la cabine réelle d'un ascenseur et effectuer le déplacement vertical entre les différents étages.
Rails de guidage (Guide rails)		Dispositif de guidage linéaire assurant la stabilité verticale de la cabine et empêchant tout balancement ou frottement lors des déplacements.
Poulie et câble de traction		Système mécanique de transmission de mouvement permettant de convertir la rotation du moteur pas à pas en un déplacement vertical de la cabine.

3.2.2 Éléments électroniques et électriques

Le tableau suivant (Tableau 3.2) récapitule l'ensemble des composants électroniques et électriques intégrés dans le prototype pour assurer l'alimentation, la détection, la commande des actionneurs et le traitement numérique :

Tableau 3.2 : Éléments électroniques et électriques du système

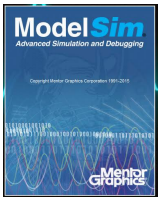
Éléments	Illustration	Description et Rôle
Carte de développement FPGA (DE1)		Utilisée pour la conception d'un contrôleur intelligent dédié aux ascenseurs. Elle sert également lors des phases de test, pour la gestion des affichages, du bouton d'arrêt d'urgence ainsi que pour la fourniture de l'alimentation $3.3V$.
Moteur pas à pas (NEMA 17)		Actionneur principal chargé d'assurer le levage de la cabine. C'est un moteur puissant capable de soulever la cabine avec aisance et d'assurer un positionnement précis.
Driver du moteur pas à pas (DRV8825)		Module d'interface de puissance pour le moteur pas à pas, intercalé entre la FPGA et le moteur NEMA 17 pour piloter le courant des enroulements et contrôler les pas du moteur.
Servomoteur (MG996R)		Actionneur dédié à l'ouverture et la fermeture de la porte de la cabine, offrant un mouvement fluide, doux et précis.
Capteurs de fin de course (KW12-3)		Capteurs mécaniques de type micro-switch installés aux étages extrêmes (inférieur et supérieur) pour détecter l'arrivée de la cabine et sécuriser les limites de course.

Éléments	Illustration	Description et Rôle
Capteurs infrarouges (Modules IR)		Capteurs optiques installés au niveau des étages intermédiaires pour détecter la présence de la cabine lors de son passage ou arrêt.
Boutons-poussoirs tactiles (12x12 mm)		Utilisés pour la réception et la saisie des requêtes d'étages internes ou externes par les utilisateurs.
Source d'alimentation 12V		Alimentation dédiée à la puissance du moteur pas à pas NEMA 17 via le driver DRV8825.
Source d'alimentation 5V		Alimentation dédiée aux composants nécessitant cette tension, notamment le servomoteur MG996R et la carte FPGA DE1.
Condensateur électrolytique (100 µF)		Placé entre les bornes de l'alimentation $12V$ afin de filtrer les perturbations, absorber les pics de courant et stabiliser l'alimentation.
Plaque d'essai (Breadboard) et câbles de connexion (jumpers)		Support de câblage permettant d'assurer les liaisons électriques et l'interconnexion entre les différents composants du prototype.
Multimètre numérique		Instrument de mesure utilisé pour la vérification des niveaux de tension, le test de continuité des lignes et la validation du câblage.

3.2.3 Logiciels et environnement de développement

Le tableau suivant (Tableau 3.3) récapitule l'ensemble des outils logiciels, des langages de description et des équipements informatiques utilisés pour la conception, la simulation, la synthèse et l'implantation du système sur la carte FPGA :

Tableau 3.3 : Logiciels et environnement de développement du système

Éléments	Illustration	Description et Rôle
Quartus II (Intel / Altera)		Environnement de développement intégré (IDE) principal utilisé pour la rédaction du code VHDL, la compilation, l'affectation des broches (Pin Planner), la synthèse logique et le chargement du design sur la carte FPGA.
ModelSim		Outil de simulation fonctionnelle et temporelle permettant de valider le comportement du système via un banc de test (Testbench) et de visualiser les chronogrammes des signaux avant l'exécution réelle sur la carte.
Langage VHDL		Langage de description matérielle (HDL) utilisé pour la modélisation de l'architecture interne et la logique de contrôle de l'ascenseur.
Câble de programmation USB-Blaster		Câble d'interface de téléchargement permettant le transfert du fichier de configuration généré par Quartus II vers la mémoire de la FPGA DE1.
Ordinateur hôte (PC)		Station de travail utilisée pour exécuter les logiciels de conception, gérer l'environnement de développement et assurer la communication avec la carte FPGA via l'interface USB.

3.3 Conception mécanique du Prototype

L'objectif principal de la conception mécanique est de réaliser une maquette expérimentale (Prototype) simulant l'environnement physique réel de l'ascenseur. Cette structure sert de support physique pour implémenter et valider les parties électroniques et logicielles , qui constituent le cœur de ce projet. Elle permet de tester concrètement le déplacement de la cabine, la réponse du moteur et le fonctionnement des capteurs lors du passage entre les différents étages.

3.3.1 Méthodologie de Réalisation

Pour concevoir cette structure en bois tout en garantissant un mouvement fluide et équilibré, les étapes suivantes ont été suivies :

- **Construction de la structure globale (Châssis)** : Conception d'un châssis en bois stable comprenant une base solide, une gaine d'ascenseur divisée en étages clairement définis pour observer l'arrêt de la cabine, ainsi qu'une salle des machines en partie supérieure pour isoler et fixer le moteur.
- **Installation des rails de guidage (Guide Rails)** : Fixation des rails sur les parois intérieures de la gaine afin de guider le déplacement vertical de la cabine et d'éviter tout balancement ou oscillation durant la montée et la descente.
- **Fabrication de la cabine** : Dimensionnement et fabrication d'une cabine en bois légère, adaptée aux dimensions internes de la structure, permettant au moteur de la soulever facilement sans contrainte mécanique excessive.
- **Ajout des roulettes de guidage** : Montage de roulettes spécifiques sur les côtés de la cabine, s'insérant dans les rails de guidage pour assurer un glissement doux et réduire les frottements.
- **Installation du système de levage (Poulie et Câble)** : Fixation d'une poulie sur l'axe du moteur avec des dimensions adaptées pour équilibrer le couple et la vitesse. La poulie est ensuite reliée au centre du toit de la cabine par un câble résistant, capable de supporter la charge sans risque de rupture.

3.3.2 Résultats et validation

Cette démarche a permis d'obtenir un prototype mécanique fonctionnel, stable et fluide. La cabine se déplace avec précision à l'intérieur de la gaine entre les différents niveaux, offrant ainsi une base expérimentale fiable pour l'intégration des composants électroniques et la validation du programme de contrôle.

3.4 Conception matérielle

Cette section est consacrée à la mise en œuvre pratique et à la validation expérimentale de la partie matérielle du prototype d'ascenseur intelligent. L'objectif est de structurer l'architecture physique autour de la carte FPGA DE1 en établissant des interfaces fiables entre l'unité de traitement, les éléments de détection et les actionneurs de puissance, comme l' illustre le schéma synoptique global du système (Figure 3.1).

Pour ce faire, la démarche adoptée se divise en deux chaînes fonctionnelles complémentaires :

1. La chaîne d'acquisition des signaux d'entrée : chargée de collecter les informations relatives à la position réelle de la cabine (via les capteurs fin de course et infrarouges) ainsi que les requêtes des utilisateurs (via les boutons-poussoirs internes et externes).
2. La chaîne de commande des périphériques de sortie : dédiée au pilotage du moteur pas à pas NEMA 17 (pour le déplacement de la cabine) et du servomoteur MG996R (pour le mécanisme de la porte), à travers leurs drivers respectifs.

Afin de garantir une approche méthodique et rigoureuse, chaque étape de réalisation est structurée de manière systématique en précisant l'objectif visé, le principe de fonctionnement, la procédure de câblage et de réglage, ainsi que les résultats des tests expérimentaux de validation.

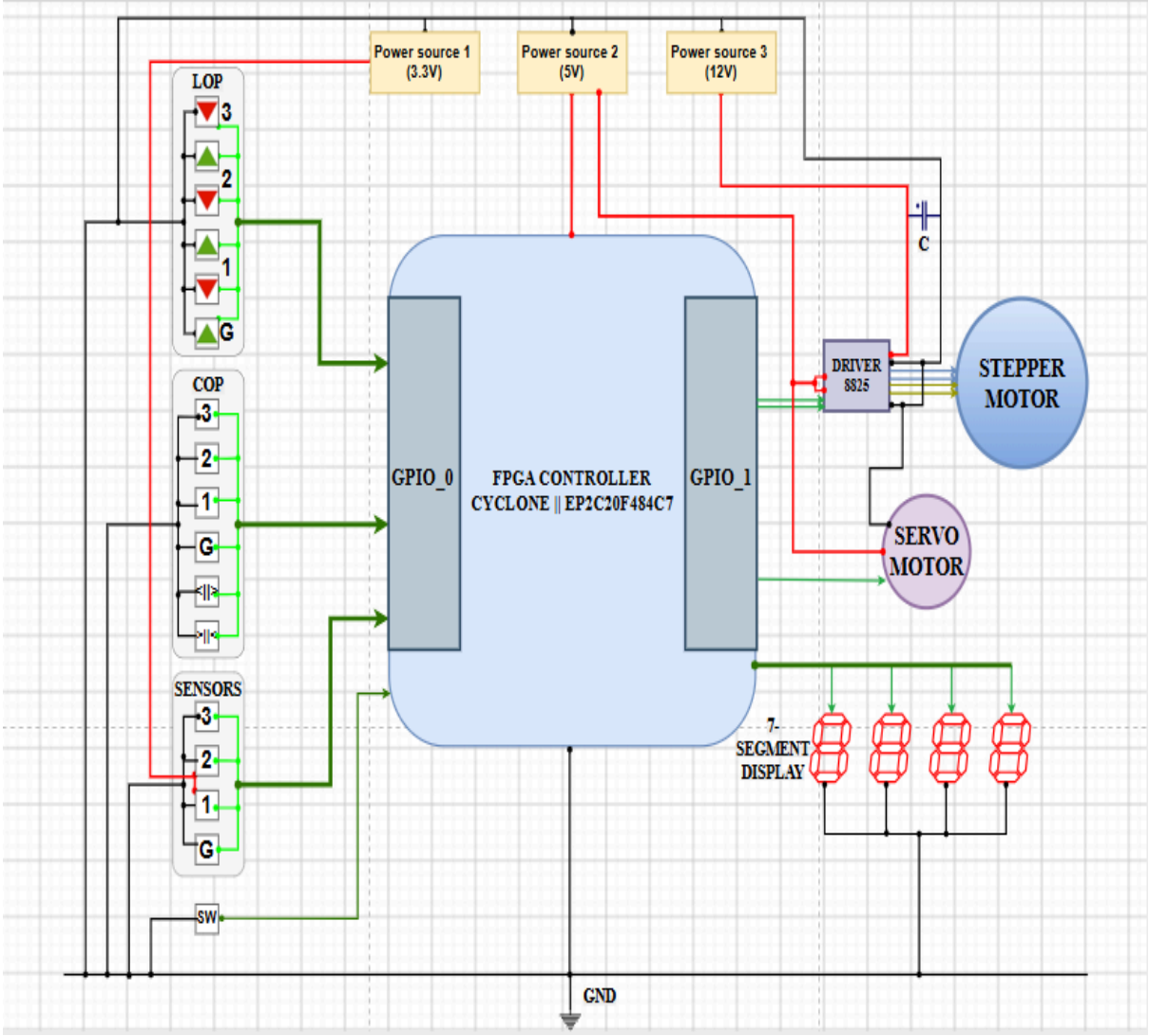


Figure 3.1 : schéma synoptique global du système

3.4.1 Mise en œuvre de la chaîne d'acquisition des signaux d'entrée

Dans un premier temps , il est nécessaire de mettre en place une chaîne d'acquisition permettant à la carte FPGA de recevoir les informations provenant des différents capteurs et commandes utilisateur.

Cette chaîne d'acquisition assure :

-la détection de la position de la cabine.

-la réception des requêtes des utilisateurs depuis les boutons d'appel externes et les boutons internes dans la cabine.

3.4.1.1 Acquisition de la position de la cabine

L'acquisition de la position de la cabine est assurée à l'aide de quatre capteurs répartis sur les différents étages de l'ascenseur.

Deux capteurs fin de course sont utilisés pour détecter les positions extrêmes de la cabine, correspondant au premier et au quatrième étage.

Les deux autres étages intermédiaires (deuxième et troisième étage) sont équipés de capteurs infrarouges d'évitement d'obstacle.

A. Mise en œuvre des capteurs fin de course

Objectif : Mettre en œuvre les capteurs fin de course afin de détecter les positions extrêmes de la cabine correspondant au premier et au quatrième étage (figure 3.3).

Principe : Les capteurs fin de course utilisés possèdent trois bornes :
-GND
-NC (Normally Closed)
-NO (Normally Open)

Le contact **NC** est fermé à l'état normal et s'ouvre lors de l'activation du capteur, tandis que le contact **NO** est ouvert à l'état normal et se ferme lors de l'activation.
Dans cette réalisation, le contact **NC** a été choisi afin d'obtenir un fonctionnement plus fiable et sécurisé. En effet, avec une configuration NC, une coupure de fil ou une déconnexion peut être détectée plus facilement, contrairement à une configuration **NO**.

Les capteurs sont connectés au port **GPIO_0** de la carte FPGA (figure 3.2), choisi comme interface dédiée aux signaux d'entrée du système.

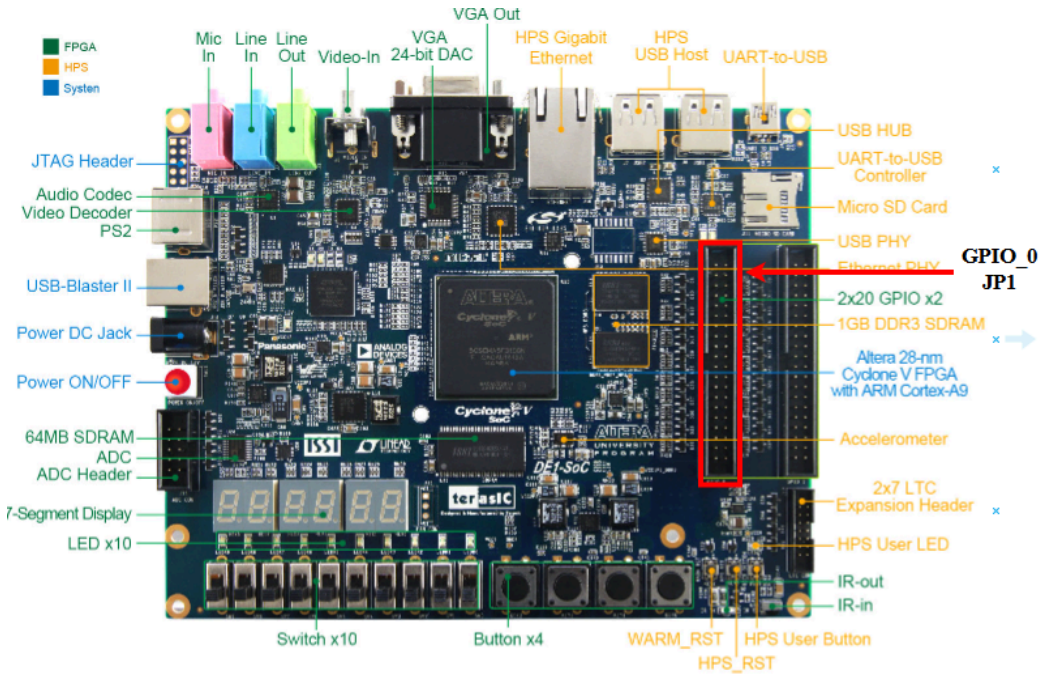


Figure 3.2 : Emplacement du GPIO_0 dans la carte DE1

Les entrées de la FPGA sont également configurées avec des **Weak Pull-Up Resistors internes**. Ces résistances permettent de maintenir un niveau logique stable lorsque le capteur est au repos et d'éviter les états flottants (*floating states*) pouvant provoquer des lectures erronées.

Procédure : Nous avons relié la borne GND de chaque capteur à la masse (GND) de la carte FPGA via le connecteur GPIO afin d'assurer une référence électrique commune avec le reste du système ; ensuite, nous avons connecté la borne NC du capteur correspondant au premier étage à l'entrée GPIO_0[6] ; puis, nous avons connecté la borne NC du capteur correspondant au quatrième étage à l'entrée GPIO_0[0] ; et enfin, nous avons activé les résistances internes Weak Pull-Up dans la configuration de la FPGA afin de garantir des niveaux logiques stables sur les entrées.

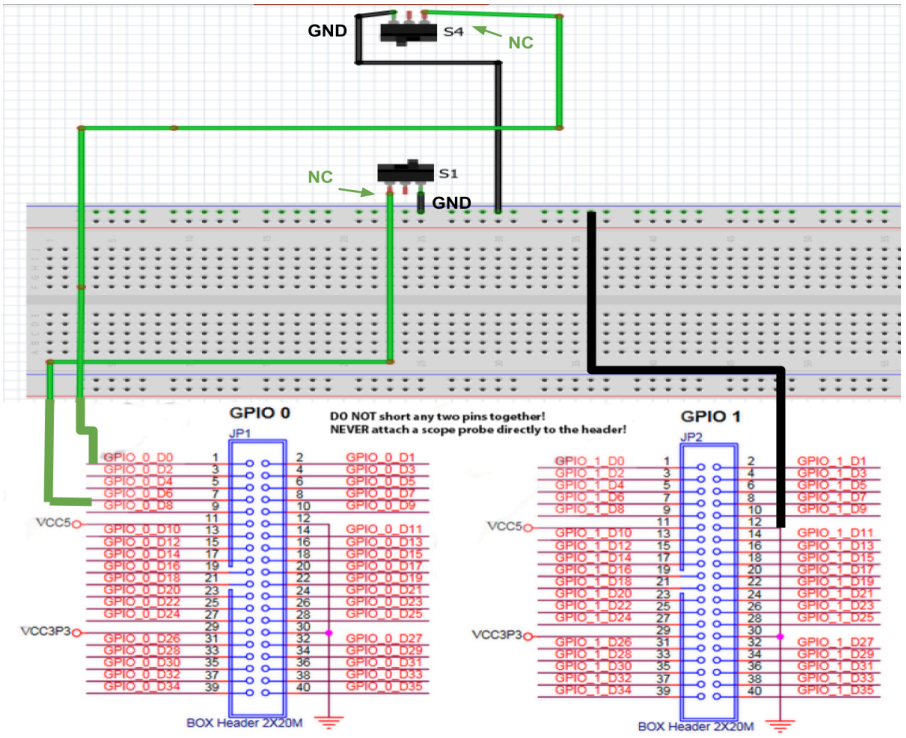


Figure 3.3 : Mise en œuvre des capteurs fin de course

B. Test des capteurs fin de course

Objectif : Vérifier le bon fonctionnement des capteurs fin de course ainsi que la bonne acquisition de leurs signaux par la carte FPGA.

Principe : Le test est réalisé à l'aide des LEDs rouges (**LEDR**) intégrées à la carte **DE1**.
Chaque capteur fin de course est associé à une LED. Lorsque le capteur est activé par pression mécanique, l'état logique de l'entrée FPGA change, ce qui provoque l'allumage de la LED correspondante. Lorsque le capteur est relâché, la LED s'éteint.

Procédure : Tout d'abord, nous avons alimenté la carte FPGA et chargé le programme de test ; ensuite, nous avons associé chaque entrée des capteurs fin de course à une LED rouge (LEDR) de la carte DE1 ; puis, nous avons appuyé manuellement sur un capteur fin de course et observé l'état de la LED correspondante ; et enfin, nous avons relâché ensuite le capteur afin de vérifier l'extinction de la LED.

Résultats : Lors de l'activation d'un capteur fin de course, la LED correspondante s'allume correctement. Après relâchement du capteur, la LED s'éteint immédiatement.
Ces résultats confirment :
-le bon fonctionnement des capteurs fin de course,
-la bonne transmission des signaux,
-ainsi que leur acquisition correcte par la carte FPGA.

C. Réglage des capteurs infrarouges d'évitement d'obstacle

Objectif : ajuster la sensibilité des capteurs infrarouges afin d'obtenir une détection correcte et stable de la cabine aux étages intermédiaires (figure 3.4).

Principe : Les modules infrarouges utilisés possèdent :
-une borne d'alimentation positive (+),
-une borne de masse (**GND**),
-ainsi que deux potentiomètres de réglage (**R5** et **R6**) permettant d'ajuster les paramètres de fonctionnement et la sensibilité de détection.
Le réglage de ces potentiomètres permet d'adapter le seuil de détection du module aux conditions expérimentales du système.

Les modules disposent également de deux indicateurs lumineux :
-une **Power LED**, indiquant la présence de l'alimentation,
-une **Sense LED**, s'allumant lorsqu'un obstacle est détecté par le capteur.

Procédure : Tout d'abord, nous avons relié la borne GND du module à la masse commune (GND) du circuit afin d'assurer une référence électrique commune avec le reste du système ; ensuite, nous avons relié la borne d'alimentation positive (+) à une tension comprise entre 3,3 V et 5 V, compatible avec le fonctionnement du module ; puis, nous avons vérifié l'allumage de la Power LED, confirmant la bonne alimentation du capteur ; après, nous avons placé la main ou un objet devant le capteur afin de simuler la présence de la cabine ; par la suite, nous avons utilisé un tournevis pour ajuster progressivement les potentiomètres R5 et R6 tout en observant l'état de la Sense LED ; et enfin, nous avons réglé les potentiomètres jusqu'à obtenir une détection stable et fiable de l'objet placé devant le capteur.

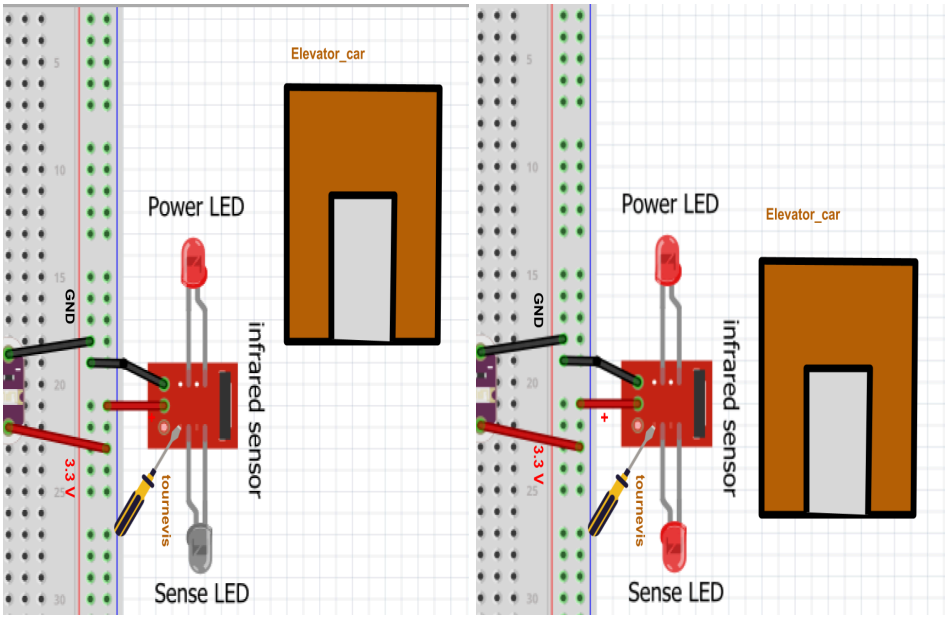


Figure 3.4 : Réglage des capteurs infrarouges d'évitement d'obstacle

Résultats et validation : Après réglage du potentiomètre, les modules infrarouges détectent correctement la présence d'un objet devant le capteur. L'allumage de la **Sense LED** confirme le bon fonctionnement et la stabilité de la détection.

D. Vérification du niveau logique de sortie du module infrarouge

Objectif : Vérifier que le signal de sortie (**OUT**) du module infrarouge est compatible avec les entrées logiques de la carte FPGA avant son interfaçage avec le système (figure 3.5).

Principe : Les entrées de la carte FPGA fonctionnent avec des niveaux logiques en **3,3 V** et sont sensibles aux surtensions. Il est donc nécessaire de vérifier que le module infrarouge délivre une tension de sortie compatible afin d'éviter tout risque d'endommagement de la FPGA.

La vérification est réalisée en mesurant la tension présente sur la borne **OUT** du module dans les différents états de détection.

Procédure : Tout d'abord, nous avons alimenté le module infrarouge en reliant la borne GND à la masse commune du circuit et la borne (+) à une alimentation de 3,3 V ; ensuite, nous avons vérifié l'allumage de la Power LED, confirmant la bonne alimentation du module ; puis, nous avons réglé le multimètre en mode mesure de tension continue (DC Voltage) ; après, nous avons connecté la borne COM du multimètre à la masse (GND) du module ; par la suite, nous avons connecté la borne positive (V) du multimètre à la sortie OUT du module ; et enfin, nous avons mesuré la tension de sortie en absence d'obstacle devant le capteur, puis en présence d'un objet devant celui-ci.

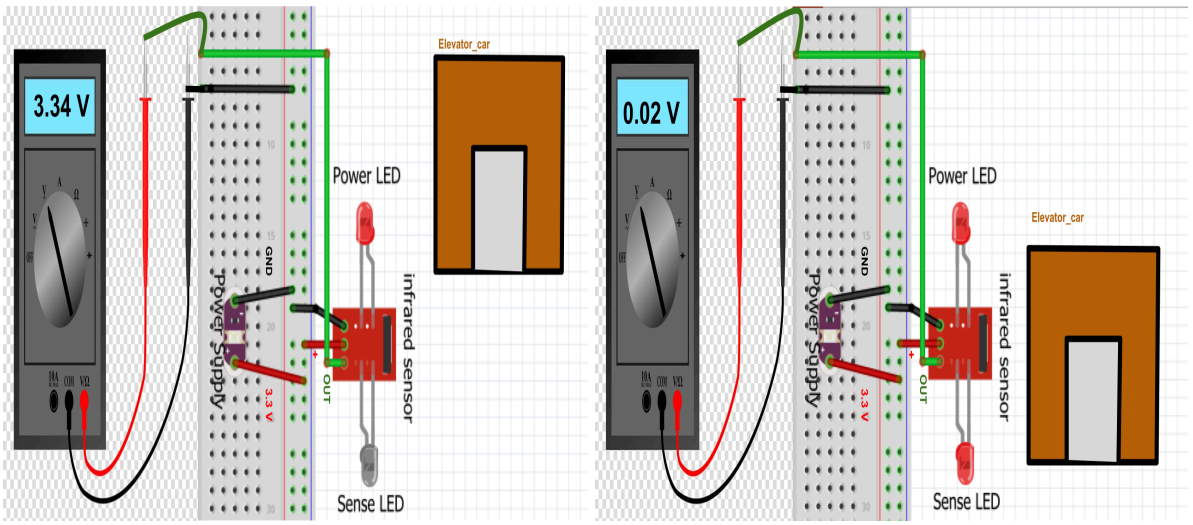


Figure 3.5 : Vérification du niveau logique de sortie du module infrarouge

Résultats et validation : Les mesures obtenues montrent :
-une tension de sortie d'environ **3,3 V** lorsqu'aucun obstacle n'est détecté,
-une tension proche de **0 V** lorsqu'un objet est détecté devant le capteur.
Ces résultats confirment que le module délivre des niveaux logiques compatibles avec les entrées de la carte FPGA et peut donc être connecté au système en toute sécurité.

E. Mise en œuvre des capteurs infrarouges d'évitement d'obstacle

Objectif : Mettre en œuvre les capteurs infrarouges afin de détecter la position de la cabine au niveau du deuxième et du troisième étage (figure 3.6).

Principe : Les modules infrarouges utilisés possèdent trois bornes :
-GND : utilisée pour la masse du circuit,
-(+) : utilisée pour l'alimentation du module,
-OUT : utilisée pour transmettre le signal logique de détection.

Les modules sont alimentés en **3,3 V** afin d'obtenir un niveau logique haut compatible avec les entrées de la carte FPGA.

Les sorties **OUT** des capteurs sont connectées au port **GPIO_0** de la carte FPGA, choisi comme interface dédiée aux signaux d'entrée du système.

Procédure : Tout d'abord, nous avons relié la borne GND de chaque module infrarouge à la masse commune (GND) du système via la carte FPGA afin d'assurer une référence électrique commune avec le reste du système ; ensuite, nous avons alimenté les modules infrarouges en 3,3 V à partir du connecteur GPIO_1 de la carte FPGA ; puis, nous avons connecté la borne OUT du capteur correspondant au deuxième étage à l'entrée GPIO_0[4] ; et enfin, nous avons connecté la borne OUT du capteur correspondant au troisième étage à l'entrée GPIO_0[2].

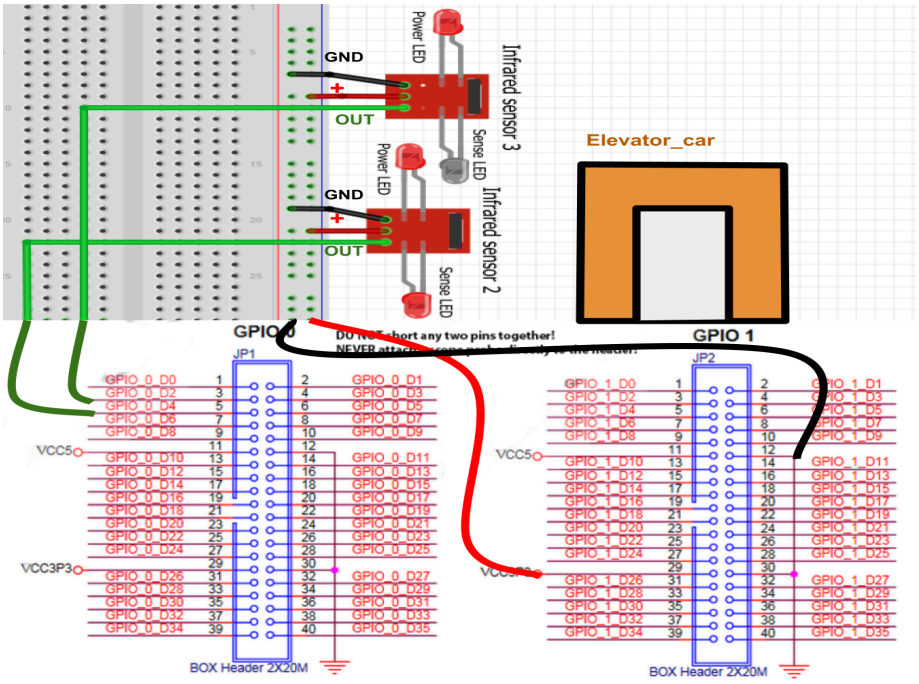


Figure 3.6 : Mise en œuvre des capteurs infrarouges d'évitement d'obstacle

F. Test des capteurs infrarouges d'évitement d'obstacle

Objectif : Vérifier le bon fonctionnement des capteurs infrarouges ainsi que la bonne acquisition de leurs signaux par la carte FPGA.

Principe : Le test est réalisé à l'aide des LEDs rouges (**LEDR**) intégrées à la carte **DE1**.
Chaque sortie de capteur infrarouge est associée à une LED afin de visualiser l'état logique du signal acquis par la FPGA.

Procédure : Tout d'abord, nous avons alimenté la carte FPGA et chargé le programme de test ; ensuite, nous avons associé chaque sortie de capteur infrarouge à une LED rouge (LEDR) de la carte DE1 ; puis, nous avons observé l'état de la LED en absence d'objet devant le capteur ; et enfin, nous avons passé ensuite la main ou un objet devant le capteur et observé la variation de l'état de la LED.

Résultats et validation : En absence d'objet devant le capteur, la LED correspondante reste allumée, indiquant la présence d'un niveau logique haut (**1 logique**).
Lorsqu'un objet est détecté devant le capteur, la LED s'éteint, ce qui confirme le passage de la sortie à un niveau logique bas (**0 logique**).

Ces résultats valident le bon fonctionnement des capteurs infrarouges ainsi que leur bonne acquisition par la carte FPGA.

3.4.1.2 Acquisition des requêtes des utilisateurs

L'acquisition des requêtes des utilisateurs est assurée à l'aide des boutons-poussoirs d'appel extérieurs et des boutons-poussoirs de sélection situés à l'intérieur de la cabine. Ces dispositifs permettent de transmettre les demandes des utilisateurs à la carte FPGA afin qu'elles soient prises en compte par le système de commande.

Les étages intermédiaires (deuxième et troisième étage) sont équipés de deux boutons-poussoirs d'appel : un bouton **UP** permettant de demander la cabine pour un déplacement vers un étage supérieur, et un bouton **DOWN** permettant de demander la cabine pour un déplacement vers un étage inférieur.

Les étages extrêmes ne disposent que d'un seul bouton-poussoir d'appel, conformément au sens de déplacement possible. Ainsi, le premier étage est équipé uniquement d'un bouton **UP**, tandis que le quatrième étage est équipé uniquement d'un bouton **DOWN**.

À l'intérieur de la cabine, quatre boutons-poussoirs de sélection sont installés, chacun correspondant à un étage de l'ascenseur. Ils permettent aux passagers de sélectionner directement leur étage de destination après être entrés dans la cabine.

A. Identification des connexions internes des boutons-poussoirs

Objectif : Identifier les bornes électriquement reliées à l'intérieur des boutons-poussoirs afin d'assurer un câblage correct avec la carte FPGA.

Principe : Les boutons-poussoirs utilisés dans ce projet possèdent **quatre bornes**. En interne, ces bornes sont réparties en **deux paires**, chaque paire étant électriquement reliée.
Avant leur mise en œuvre, il est donc nécessaire d'identifier ces deux paires afin de sélectionner correctement les bornes à utiliser lors du câblage.
Cette vérification est réalisée à l'aide d'un multimètre en mode **test de continuité**, qui permet de déterminer quelles bornes sont reliées électriquement.

Procédure : Tout d'abord, nous avons réglé le multimètre en mode test de continuité ; ensuite, nous avons mesuré la continuité entre chaque paire de bornes du bouton-poussoir ; et enfin, lorsque deux bornes présentaient une continuité électrique, nous avons constaté qu'elles étaient reliées en interne, les deux bornes restantes constituant alors la deuxième paire reliée en interne.

B. Mise en œuvre des boutons-poussoirs

Objectif : Mettre en œuvre les boutons-poussoirs de la cabine et les boutons d'appel des étages afin de transmettre les requêtes des utilisateurs à la carte FPGA.

Principe : Chaque bouton-poussoir possède quatre bornes réparties en deux paires électriquement reliées en interne.

Pour le raccordement, une borne est choisie dans la première paire et une autre dans la seconde paire. Ainsi, l'appui sur le bouton établit la connexion électrique entre ces deux bornes, ce qui permet à la carte FPGA de détecter la requête de l'utilisateur.

Les boutons-poussoirs sont connectés au port **GPIO_0** de la carte FPGA, dédié aux signaux d'entrée du système.

Procédure : Tout d'abord, nous avons relié une borne de chaque bouton-poussoir à la masse commune (GND) du système ; ensuite, nous avons relié l'autre borne, appartenant à la seconde paire, à une entrée du connecteur GPIO_0 de la carte FPGA ; puis, nous avons réalisé les connexions des boutons de la cabine comme suit (figure 3.7) : Cab1 à GPIO_0[34],

Cab2 à GPIO_0[32], Cab3 à GPIO_0[30], et Cab4 à GPIO_0[28] ; et enfin, nous avons réalisé les connexions des boutons d'appel extérieurs comme suit (figure 3.8) : Call1_up à GPIO_0[10], Call2_up à GPIO_0[12], Call2_down à GPIO_0[13], Call3_up à GPIO_0[14], Call3_down à GPIO_0[15], et Call4_down à GPIO_0[16].

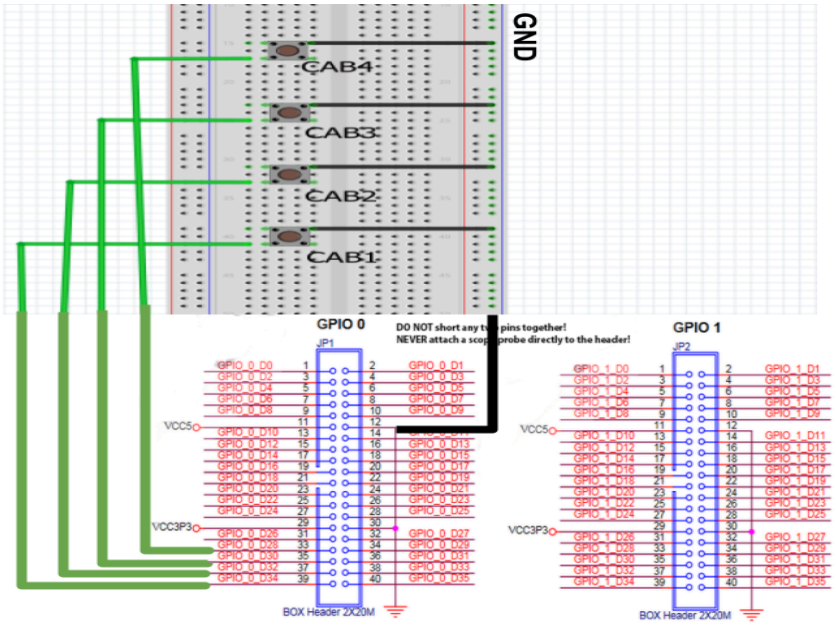


Figure 3.7 : Mise en œuvre des boutons-poussoirs de la cabine

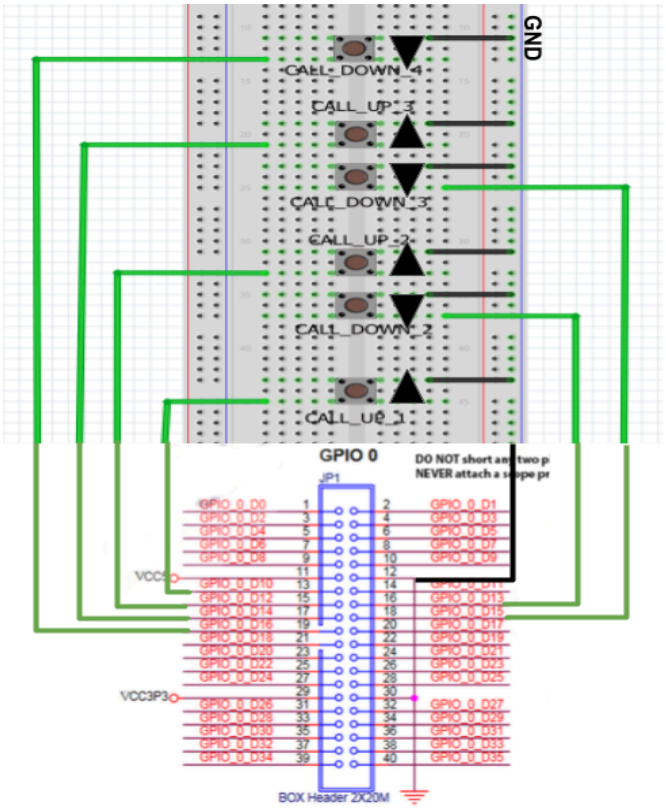


Figure 3.8 : Mise en œuvre des boutons-poussoirs d'appel extérieurs

C. Test des boutons-poussoirs

Objectif : Vérifier le bon fonctionnement des boutons-poussoirs ainsi que la bonne acquisition de leurs états logiques par la carte FPGA.

Principe : Le test est réalisé à l'aide d'un programme **VHDL** chargé sur la carte **DE1**. Chaque bouton-poussoir est associé à une LED rouge (**LEDR**) afin de visualiser l'état logique du signal lu par la FPGA.

Procédure : Tout d'abord, nous avons alimenté la carte FPGA et chargé le programme de test VHDL ; ensuite, nous avons associé chaque bouton-poussoir à une LED rouge (LEDR) de la carte DE1 ; puis, nous avons observé l'état de la LED lorsque le bouton-poussoir est au repos ; après, nous avons appuyé sur le bouton-poussoir et observé la variation de l'état de la LED ; et enfin, nous avons répété cette opération pour l'ensemble des boutons de la cabine et des boutons d'appel extérieurs.

Résultats et validation : Les essais réalisés montrent qu'en l'absence d'appui sur un bouton-poussoir, la LED correspondante reste allumée, indiquant que l'entrée de la FPGA est à l'état logique **1**.

Lorsqu'un bouton-poussoir est actionné, la LED s'éteint, ce qui confirme le passage de l'entrée à l'état logique **0**.

Ces résultats valident le bon fonctionnement des boutons-poussoirs, la conformité de leur câblage ainsi que leur bonne acquisition par la carte FPGA.

3.4.2 Mise en œuvre des chaînes de commande des périphériques de sortie

Cette section est consacrée à la mise en œuvre des actionneurs gérés par la carte FPGA DE1 pour l'automatisation de l'ascenseur. Nous y détaillons l'interfaçage matériel, le paramétrage et la validation expérimentale des deux chaînes de commande principales : le moteur pas à pas pour le déplacement vertical de la cabine, et le servomoteur MG996R pour l'ouverture et la fermeture de la porte.

3.4.2.1 Mise en œuvre de la chaîne de commande du moteur pas à pas de la cabine

Cette étape consiste à mettre en œuvre la chaîne de commande du moteur pas à pas assurant le déplacement de la cabine de l'ascenseur.

La chaîne de commande comprend principalement la génération des signaux de pilotage par la carte FPGA, leur transmission vers le driver, ainsi que l'alimentation et le réglage des différents éléments nécessaires au fonctionnement du moteur (figure 3.9).

Cette phase permet également de valider le bon fonctionnement du système à travers plusieurs tests expérimentaux.

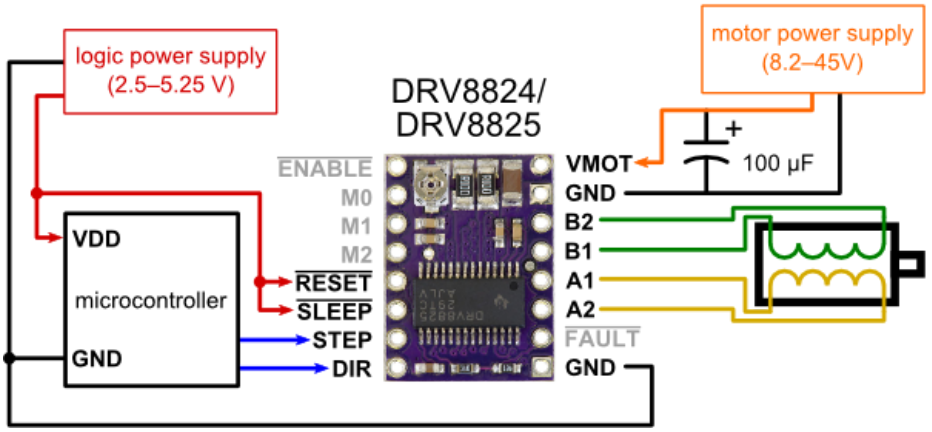


Figure 3.9 : chaîne de commande du moteur pas à pas de la cabine

A. Configuration du courant moteur via Vref du driver

Objectif : Le réglage de la tension de référence (Vref) permet de limiter le courant fourni au moteur pas à pas afin d'assurer un fonctionnement stable sans surchauffe.

Principe : Le driver DRV8825 utilise la tension Vref pour fixer le courant maximal selon la relation :

$Imax = Vref/(5 * Rsense)$ **[22]**
$\Rightarrow Vref = Imax * 5 * Rsense$
$Rsense = 0.1\,ohm$ **[22]**

Pour le moteur pas à pas Nema 17 model 42HS34-1704 : $Imax = current/phase = 1.7\,A$ **[23]**
$Vref = 1.7 * 5 * 0.1$
$Vref = 0.85\,V$ (cette valeur représente la limite maximale du courant).

Procédure de réglage :
1-Alimentation du Driver DRV8825 (figure 3.10) **:** Dans cette étape, le driver est alimenté afin d'assurer son fonctionnement dans une plage de tension adéquate.

Tout d'abord, nous avons alimenté la broche VMOT avec une tension de 12V, car le driver accepte une plage de 8,2 V à 45 V [22] , ce qui rend cette valeur suffisante et stable ; ensuite, nous avons relié la masse GND de l'alimentation à la masse GND du driver pour garantir une référence commune ; et enfin, nous avons placé un condensateur de 100 µF entre VMOT et GND afin de filtrer les perturbations et absorber les pics de courant, ce qui permet de protéger le driver et de stabiliser l'alimentation.

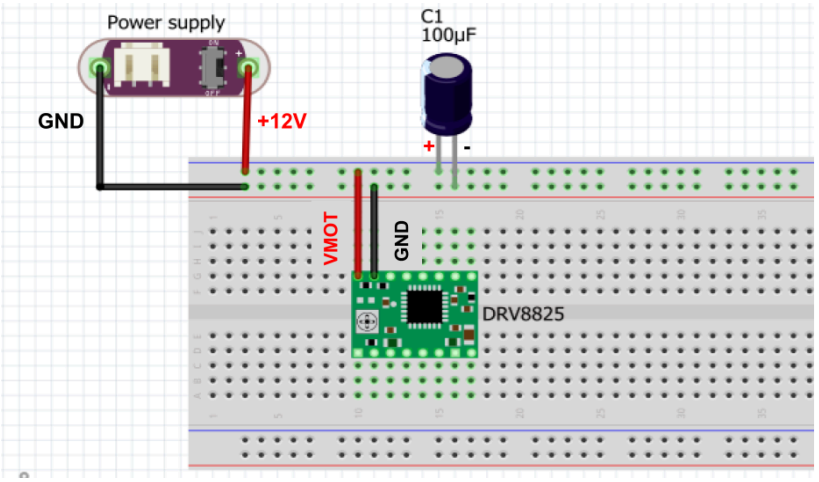


Figure 3.10 : Alimentation du Driver DRV8825

2-Configuration des broches SLEEP et RESET (figure 3.11)**:** Dans cette étape, nous assurons l'activation du driver en configurant correctement les broches de contrôle.

Tout d'abord, nous avons appliqué une tension de 5 V aux broches SLEEP et RESET afin de sortir le driver du mode veille et d'éviter toute réinitialisation [22] , la valeur 5 V étant appropriée, car elle appartient à la plage de tension logique acceptée (Input High Voltage) du driver, comprise entre 2,2V et 5,25V [22] , garantissant ainsi une reconnaissance fiable du niveau logique haut ; et enfin, nous avons relié la masse GND de l'alimentation 5 V à la masse GND du driver pour garantir une référence commune des signaux logiques, cette configuration permettant de maintenir le driver en état de fonctionnement normal (mode actif).

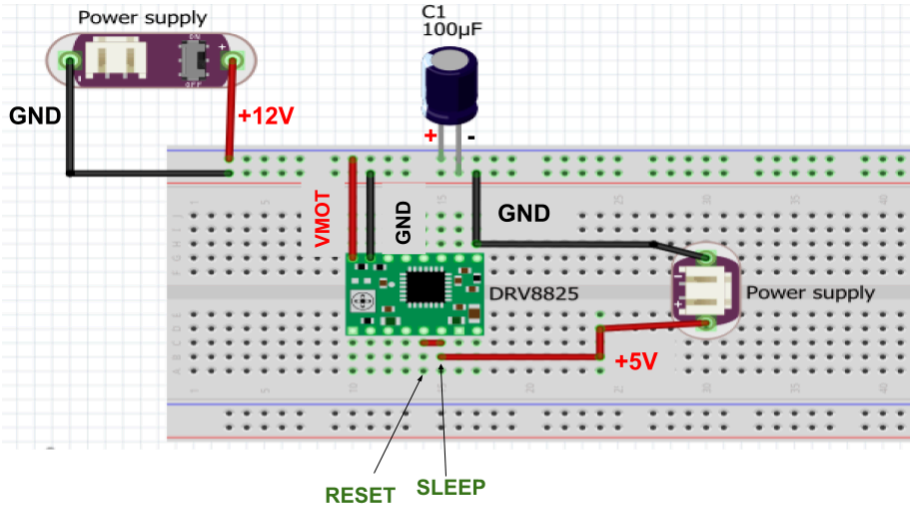


Figure 3.11 : Configuration des broches SLEEP et RESET

3-Mesure et réglage de la tension VREF (figure 3.12)**:** Dans cette étape, nous procédons à la mesure et à l'ajustement de la tension de référence **Vref** afin de limiter correctement le courant du moteur.

Tout d'abord, nous avons réglé le multimètre en mode voltmètre (tension continue) et sélectionné un calibre de 2 V pour obtenir une meilleure précision de mesure ; ensuite, nous

avons connecté la borne COM (noire) du multimètre à la masse GND commune du circuit ; puis, nous avons placé la borne positive (rouge) du multimètre sur le curseur du potentiomètre (la vis métallique) ; après, nous avons utilisé un tournevis pour faire tourner le potentiomètre tout en observant la valeur affichée sur le multimètre ; et enfin, nous avons ajusté progressivement la tension Vref jusqu'à atteindre une valeur initiale de 0,6 V, tout en sachant qu'une rotation dans le sens des aiguilles d'une montre entraîne une diminution de la tension Vref et qu'une rotation dans le sens inverse des aiguilles d'une montre entraîne une augmentation de la tension Vref.

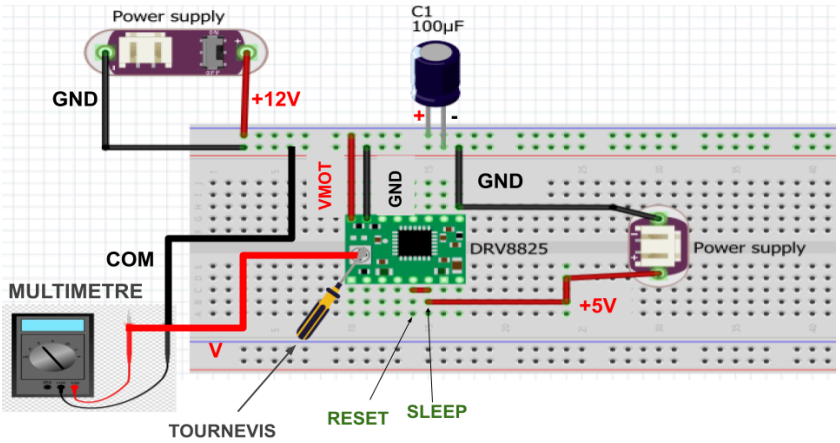


Figure 3.12 : Mesure et réglage de la tension VREF

B. Connexion du moteur pas à pas (NEMA 17) au driver

Objectif : Assurer une connexion correcte entre le moteur pas à pas et le driver (figure 3.13) afin de permettre son fonctionnement sans endommager les composants.

Principe : Le moteur pas à pas **NEMA 17** bipolaire est constitué de deux bobines (appelées **Coil A** et **Coil B**) , Chaque bobine doit être connectée à une paire de sorties du driver : La première bobine (**Coil A**) aux bornes **A1** et **A2** et la deuxième bobine (**Coil B**) aux bornes **B1** et **B2 ,** Une mauvaise correspondance entre les bobines et les sorties peut entraîner des vibrations, un fonctionnement instable ou une absence de rotation.

Procédure : Tout d'abord, nous avons réglé le multimètre en mode mesure de résistance (ohmmètre) ; ensuite, nous avons mesuré la résistance entre chaque paire de fils parmi les quatre fils du moteur ; puis, nous avons identifié les deux fils qui présentaient une résistance non infinie, constatant qu'ils appartenaient à la même bobine ; après, nous avons connecté cette paire à A1 et A2 (Coil A) ; et enfin, nous avons connecté les deux fils restants à B1 et B2 (Coil B) ; noté que l'ordre de connexion (A1/A2 ou B1/B2) n'était pas critique, car inverser les fils d'une même bobine changeait seulement le sens de rotation.

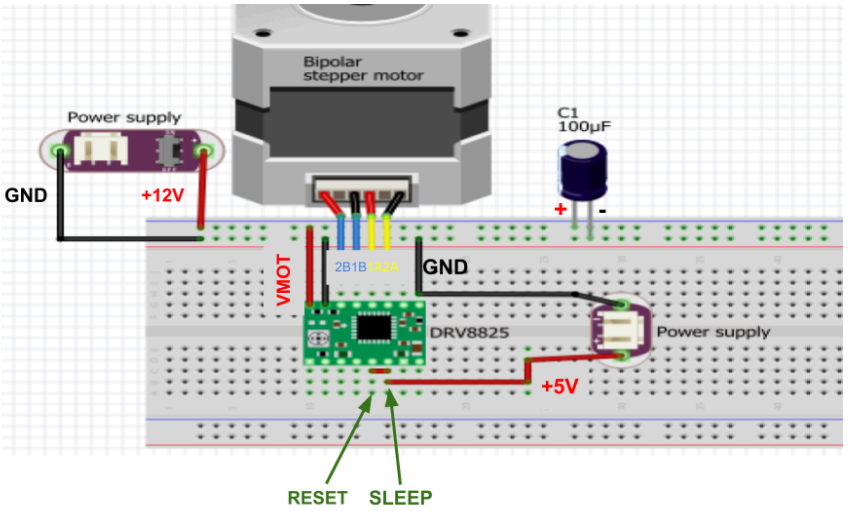


Figure 3.13 : Connexion du moteur pas à pas au driver

C. Interfaçage du driver avec la carte FPGA (DE1)

Objectif : Établir une interface fiable entre la carte **FPGA DE1** et le driver afin de piloter le moteur pas à pas **NEMA 17** (figure 3.15).

Principe : Le pilotage du moteur pas à pas repose principalement sur deux signaux logiques :

STEP : Ce signal est généralement généré sous forme d'**impulsions périodiques (**type PWM**)**. Chaque impulsion correspond à un pas élémentaire du moteur, dont l'angle est de **1,8° par pas** pour le moteur **NEMA 17 [23]**. Ainsi, la fréquence des impulsions STEP détermine directement la vitesse de rotation du moteur.

DIR (Direction**)** : Ce signal logique définit le **sens de rotation du moteur**. Un niveau logique donné correspond à un sens de rotation, tandis que l'état inverse du signal entraîne l'inversion automatique de ce sens.

Pour assurer un fonctionnement correct, une **masse commune GND** entre la carte FPGA et le driver est indispensable, afin de garantir une référence électrique identique pour l'interprétation des niveaux logiques.

La carte **FPGA DE1** fournit des signaux logiques en **3,3V**, ce qui est compatible avec le driver **DRV8825** , puisque celui-ci accepte un niveau logique haut (**Input High Voltage**) compris entre **2,2 V et 5,25 V**.

Procédure : Tout d'abord, nous avons utilisé le connecteur GPIO_1 (JP2) de la carte DE1 pour l'interfaçage (figure 3.14) ; ensuite, nous avons connecté le pin STEP à GPIO_1_D0 ; puis, nous avons connecté le pin DIR à GPIO_1_D1 ; et enfin, nous avons relié la masse GND du GPIO_1 (JP2) à la masse commune du circuit pour assurer une référence commune.

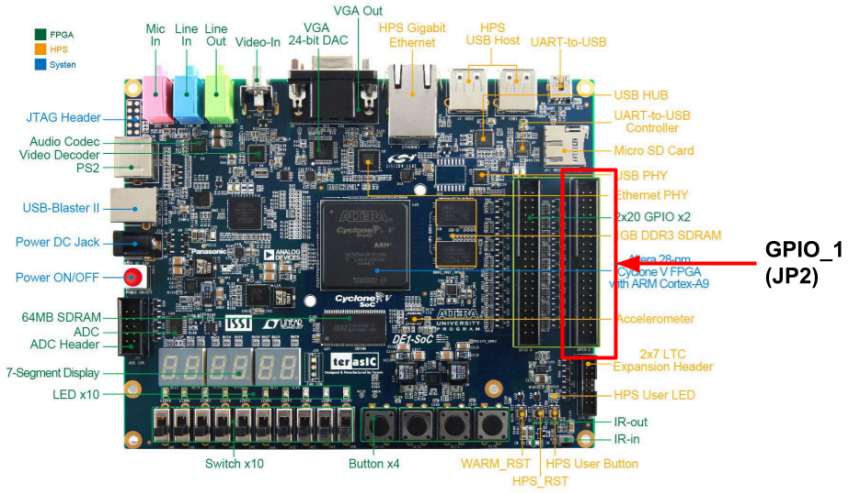


Figure 3.14 : Emplacement du GPIO_1 dans la carte DE1

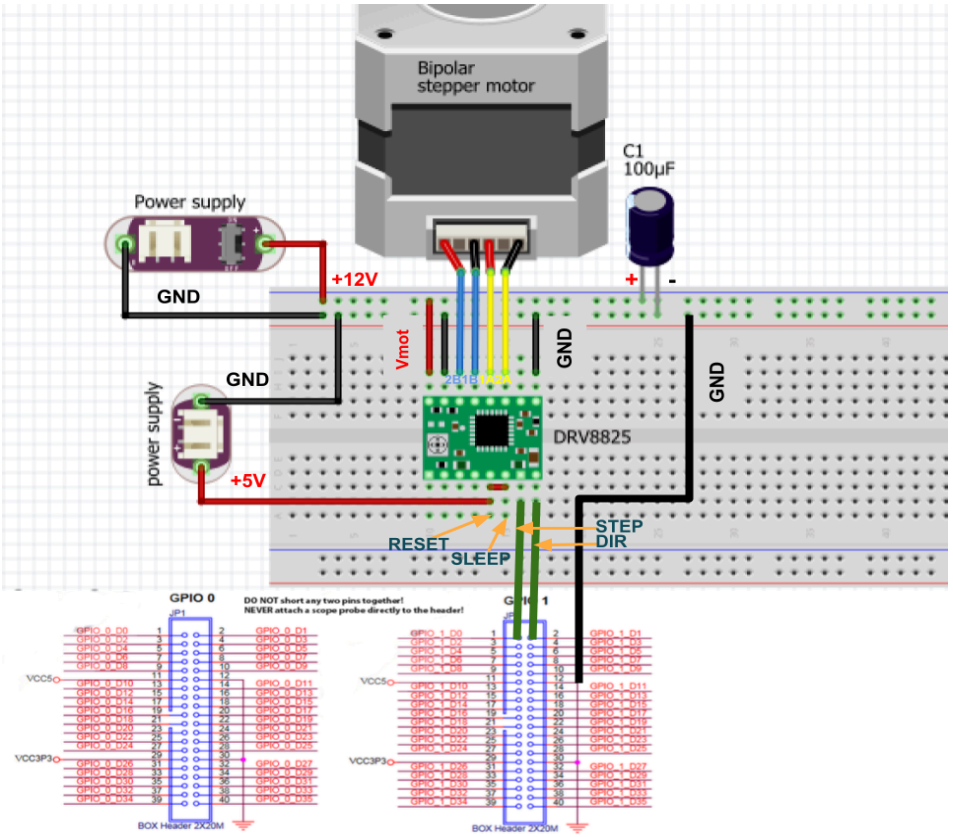


Figure 3.15 : Interfaçage du driver avec la carte FPGA (DE1)

D. Test de la chaîne de commande du moteur pas à pas de la cabine

Objectif : Valider le bon fonctionnement de la chaîne de commande complète du moteur pas à pas (FPGA + driver + moteur) à travers un programme **VHDL** de test implémenté sur la carte **FPGA DE1**.

Principe : le test repose sur l'utilisation de trois interrupteurs (switches) de la carte FPGA afin de contrôler le comportement du moteur pas à pas ; donc, nous avons utilisé le SW0 (Reset) permettant la remise à zéro du système, de sorte que lorsque ce switch est activé, le

moteur est désactivé et ne peut pas fonctionner ; puis, nous avons utilisé le SW1 (Enable / Start) contrôlant l'activation du moteur (activé le moteur fonctionne, désactivé le moteur est arrêté) ; et enfin, nous avons utilisé le SW2 (Direction) définissant le sens de rotation du moteur (un état logique haut ou bas correspond à un sens de rotation, l'état inverse entraînant automatiquement l'inversion du sens), ce qui a permis, par la combinaison de ces trois switches, de tester de manière simple et efficace l'ensemble de la chaîne de commande.

Procédure : Tout d'abord, nous avons alimenté la carte FPGA DE1 via le connecteur DC jack 5 V ; ensuite, nous avons activé le switch SW0 (Reset) avant le téléchargement du programme ; puis, nous avons connecté la carte au PC via USB-Blaster ; après, nous avons compilé et chargé le programme VHDL de test sur la carte à l'aide de Quartus II ; par la suite, nous avons désactivé le reset SW0 pour permettre l'exécution du programme ; et enfin, nous avons testé le comportement du moteur en manipulant les switches SW1 et SW2.

Résultats et validation : Après le chargement du programme de test et la mise en fonctionnement du système, les observations suivantes ont été obtenues :
-Lorsque le switch **SW0 (Reset)** est activé, le moteur reste à l'arrêt, ce qui confirme le bon fonctionnement du mécanisme de réinitialisation.
-Après désactivation du reset, l'activation du switch **SW1** provoque la rotation du moteur, tandis que sa désactivation entraîne l'arrêt immédiat du moteur. Ceci valide la bonne génération du signal **STEP** par la carte FPGA et sa bonne interprétation par le driver.
-La modification de l'état du switch **SW2** entraîne l'inversion du sens de rotation du moteur, ce qui confirme le bon fonctionnement du signal **DIR**.
-Le moteur fonctionne de manière stable et régulière, sans vibrations anormales ni pertes de pas observées durant le test.

Ces résultats permettent de valider le bon fonctionnement de l'ensemble de la chaîne de commande composée de la carte FPGA, du driver et du moteur pas à pas.

3.4.2.2 Mise en œuvre de la chaîne de commande du servomoteur de la porte

Dans cette section, nous présentons la mise en œuvre de la chaîne de commande du servomoteur MG996R via la carte FPGA DE1. Nous détaillons d'abord le câblage et l'interfaçage matériel, puis nous validons son fonctionnement par un test pratique en VHDL.

A. Mise en œuvre du servomoteur de la porte

Objectif : Assurer l'alimentation et l'interfaçage du servomoteur **MG996R** avec la carte FPGA (figure 3.16) afin de permettre la commande du mécanisme d'ouverture et de fermeture de la porte.

Principe : Le servomoteur **MG996R** possède trois conducteurs , Fil brun : masse (GND), Fil rouge : alimentation (**VCC**) et Fil orange : entrée de commande recevant un signal PWM (Pulse Width Modulation).

Le servomoteur nécessite une alimentation de **5 V** pour fonctionner correctement. La position angulaire du servomoteur est déterminée par le signal PWM appliqué sur son entrée de commande. Ce signal est généré par la carte FPGA et transmis au moteur à travers la sortie GPIO_1.

Procédure : Tout d'abord, nous avons relié le fil brun du servomoteur à la masse commune (GND) du système afin d'assurer une référence électrique commune entre le moteur et la carte FPGA ; ensuite, nous avons relié le fil rouge à une alimentation de 5 V, conformément aux spécifications du servomoteur MG996R ; et enfin, nous avons connecté le fil orange à la broche GPIO_1[30] du connecteur GPIO_1 utilisé comme interface de sortie de la carte FPGA, pour transmettre le signal PWM généré par le système de commande.

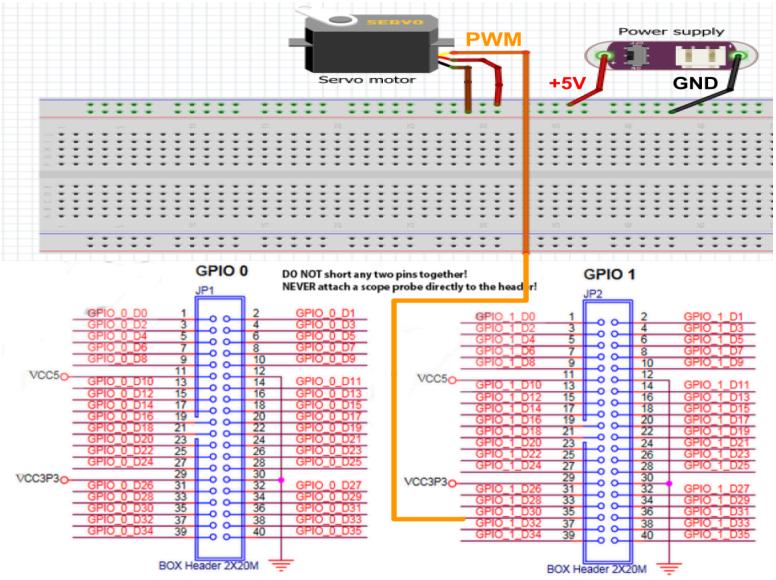


Figure 3.16 : Mise en œuvre du servomoteur de la porte

B. Test du servomoteur de la porte

Objectif : Vérifier le bon fonctionnement du servomoteur **MG996R** ainsi que la capacité de la carte FPGA à générer les signaux PWM nécessaires pour commander sa position.

Principe : Le test est réalisé à l'aide d'un programme **VHDL** implémenté sur la carte **DE1**. Deux boutons poussoirs de la carte sont utilisés pour commander directement la position du servomoteur : KEY[0] : commande une position angulaire prédéfinie, KEY[1] : commande une seconde position angulaire prédéfinie.

Ce test permet de vérifier la génération du signal PWM par la FPGA ainsi que la réponse du servomoteur aux commandes reçues.

Procédure : Tout d'abord, nous avons alimenté la carte FPGA et le servomoteur ; ensuite, nous avons chargé le programme de test VHDL sur la carte DE1 ; puis, nous avons appuyé sur le bouton KEY[0] et observé la position du servomoteur ; et enfin, nous avons appuyé sur le bouton KEY[1] et observé la nouvelle position du servomoteur.

Résultats et validation : Les essais réalisés montrent que :
- L'appui sur **KEY[0]** positionne correctement le servomoteur à **0°**.
- L'appui sur **KEY[1]** positionne correctement le servomoteur à **90°**.

Les positions obtenues sont conformes aux valeurs définies dans le programme VHDL.

Ces résultats valident le bon fonctionnement de la chaîne de commande du moteur de la porte. Ils confirment également la possibilité de commander l'ouverture et la fermeture de la porte par l'envoi d'ordres depuis la FPGA, en associant par exemple la position **0°** à la fermeture de la porte et la position **90°** à sa ouverture.

3.5 Conception logicielle

La phase de conception logicielle constitue le cœur du système de commande de l'ascenseur intelligent ; c'est à ce niveau que les exigences fonctionnelles sont traduites en algorithmes et en modules logiciels structurés en langage VHDL. Comme l'illustre le schéma RTL du système (Figure 3.17), l'architecture logicielle repose sur une organisation modulaire et hiérarchisée, conçue pour assurer un traitement fluide et déterministe des informations.

Cette démarche débute par le traitement et le conditionnement des signaux d'entrée afin de garantir la stabilité des données issues des capteurs et des boutons. Ensuite, l'unité de contrôle principale (Contrôleur principal) orchestre la logique de décision à l'aide d'une machine à états finis (FSM) et d'algorithmes d'optimisation des requêtes. Enfin, les modules de commande des périphériques de sortie génèrent les signaux de pilotage requis, avant leur interconnexion finale au sein de l'entité supérieure (Top-Level Entity) pour garantir un fonctionnement global sécurisé et performant.

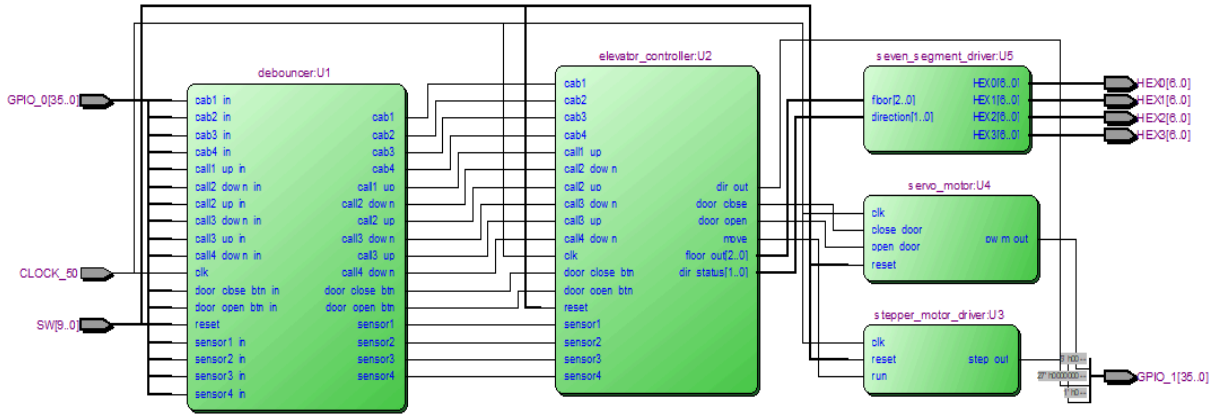


Figure 3.17 : schéma RTL du système

3.5.1 Traitement et conditionnement des signaux d'entrée

Avant de traiter les requêtes des utilisateurs via le contrôleur principal , il est indispensable de garantir la stabilité des signaux issus des boutons-poussoirs et des capteurs . Les boutons et les capteurs mécaniques génèrent en effet des états flottants et des rebonds pouvant provoquer des faux déclenchements dans le FPGA. Pour résoudre ce problème, cette étape de conditionnement consiste d'abord à configurer les résistances de rappel interne

(weak pull-up) afin de stabiliser les niveaux logiques au repos, puis à concevoir un module d'anti-rebond (debouncer) pour filtrer numériquement les signaux d'entrée.

3.5.1.1 Configuration des résistances de rappel interne (Weak Pull-Up)

Objectif : L'objectif de cette étape est de fixer un niveau logique stable au repos pour les broches d'entrée reliées aux boutons-poussoirs et aux capteurs, tout en évitant l'état flottant (*floating state*) qui risque de provoquer des déclenchements intempestifs au niveau du FPGA.

Principe : Lorsqu'un bouton-poussoir mécanique n'est pas enfoncé, la broche d'entrée se retrouve déconnectée de toute source de tension. En absence de référence, le FPGA interprète cet état comme "flottant", ce qui entraîne des oscillations aléatoires entre 0 et 1. La résistance de rappel interne relie la broche à la tension d'alimentation V_{CC} à travers une forte valeur résistive. Ainsi, l'état par défaut (au repos) est maintenu à un niveau logique HAUT (1), tandis qu'un appui sur le bouton relie la broche à la masse (GND), faisant passer le signal au niveau logique BAS (0).

Procédure de configuration : Plutôt que d'ajouter des résistances physiques externes sur la carte DE1, nous avons activé les résistances de rappel internes intégrées directement dans la puce FPGA. Cette configuration s'effectue au niveau du logiciel Quartus à travers l'outil Assignment Editor. Nous avons appliqué le paramètre Weak Pull-Up Resistor à l'ensemble des broches d'entrée associées aux boutons de commande et aux capteurs de l'ascenseur (figure 3.18).

Assignment Editor

<<new>> Filter on node names: *

	Status	From	To	Assignment Name	Value	Enabled	Entity
443	✓		SRAM_OE_N	Location	PIN_T8	Yes	
444	✓		SRAM_UB_N	Location	PIN_W7	Yes	
445	✓		SRAM_WE_N	Location	PIN_AA10	Yes	
446	✓		GPIO_0[6]	Weak Pull-Up Resistor	On	Yes	elevator_top
447	✓		GPIO_0[10]	Weak Pull-Up Resistor	On	Yes	elevator_top
448	✓		GPIO_0[16]	Weak Pull-Up Resistor	On	Yes	elevator_top
449	✓		GPIO_0[0]	Weak Pull-Up Resistor	On	Yes	elevator_top
450	✓		GPIO_0[14]	Weak Pull-Up Resistor	On	Yes	elevator_top
451	✓		GPIO_0[28]	Weak Pull-Up Resistor	On	Yes	elevator_top
452	✓		GPIO_0[30]	Weak Pull-Up Resistor	On	Yes	elevator_top
453	✓		GPIO_0[12]	Weak Pull-Up Resistor	On	Yes	elevator_top
454	✓		GPIO_0[34]	Weak Pull-Up Resistor	On	Yes	elevator_top
455	✓		GPIO_0[32]	Weak Pull-Up Resistor	On	Yes	elevator_top
456	✓		GPIO_0[2]	Weak Pull-Up Resistor	On	Yes	elevator_top
457	✓		GPIO_0[4]	Weak Pull-Up Resistor	On	Yes	elevator_top
458	✓		GPIO_0[13]	Weak Pull-Up Resistor	On	Yes	elevator_top
459	✓		GPIO_0[15]	Weak Pull-Up Resistor	On	Yes	elevator_top
460	✓		GPIO_0[20]	Weak Pull-Up Resistor	On	Yes	elevator_top
461	✓		GPIO_0[22]	Weak Pull-Up Resistor	On	Yes	elevator_top
462		<<new>>	<<new>>	<<new>>			

This cell shows the status of the assignment in the current row.

Figure 3.18 : Configuration des résistances de rappel interne sur Assignment Editor

Résultats et validation : Pour valider ce fonctionnement, nous avons effectué un test simple en reliant directement ces entrées aux LEDs de la carte DE1 : les LEDs restent allumées à

l'état repos et s'éteignent nettement lors de chaque appui sur un bouton ou activation d'un capteur .

3.5.1.2 Conception du Module d'Anti-Rebond (Debouncer)

Objectif : Le module debouncer a pour rôle de filtrer simultanément les rebonds mécaniques de l'ensemble des organes d'entrée (12 boutons-poussoirs et 4 capteurs de position) afin de fournir des signaux logiques propres, stables et parfaitement synchronisés au contrôleur principal.

Principe : Le principe repose sur une vectorisation des entrées associées à deux constantes de filtrage temporel adaptées à la nature physique de chaque composant :

1. Regroupement des signaux (Vectorisation) : Les 16 entrées brutes sont assemblées dans un seul vecteur raw de $16bits$:
    - Indices 0 à 11 : Dédiés aux 12 boutons-poussoirs (appels d'étages, cabine et portes).
    - Indices 12 à 15 : Dédiés aux 4 capteurs de position (sensor1 à sensor4).
2. Synchronisation Anti-Métastabilité : Chaque bit passe par un étage à deux bascules D (sync1 et sync2) pour éviter tout problème de métastabilité causé par les signaux asynchrones.
3. Double Temporisation Adaptative :
    - Boutons (BTN_MAX = 20 ms) : Exigent un temps de filtrage plus long de 1000000 cycles d'horloge ($50MHz \times 20ms$) pour absorber les rebonds mécaniques des appuis humains.
    - Capteurs (SENS_MAX = 5 ms) : Nécessitent un temps de réaction plus court de 250000 cycles d'horloge ($50MHz \times 5ms$) pour détecter précisément le passage de la cabine sans retard d'arrêt.
4. Validation par Boucle Paralèlle : Une boucle for inspecte les 16 voies à chaque coup d'horloge. La sortie stable(i) n'est mise à jour que si l'entrée reste identique pendant la durée fixée (BTN_MAX ou SENS_MAX).

Résultats et validation : La validation du module debouncer a été effectuée par simulation temporelle sous ModelSim (figure 3.19) :

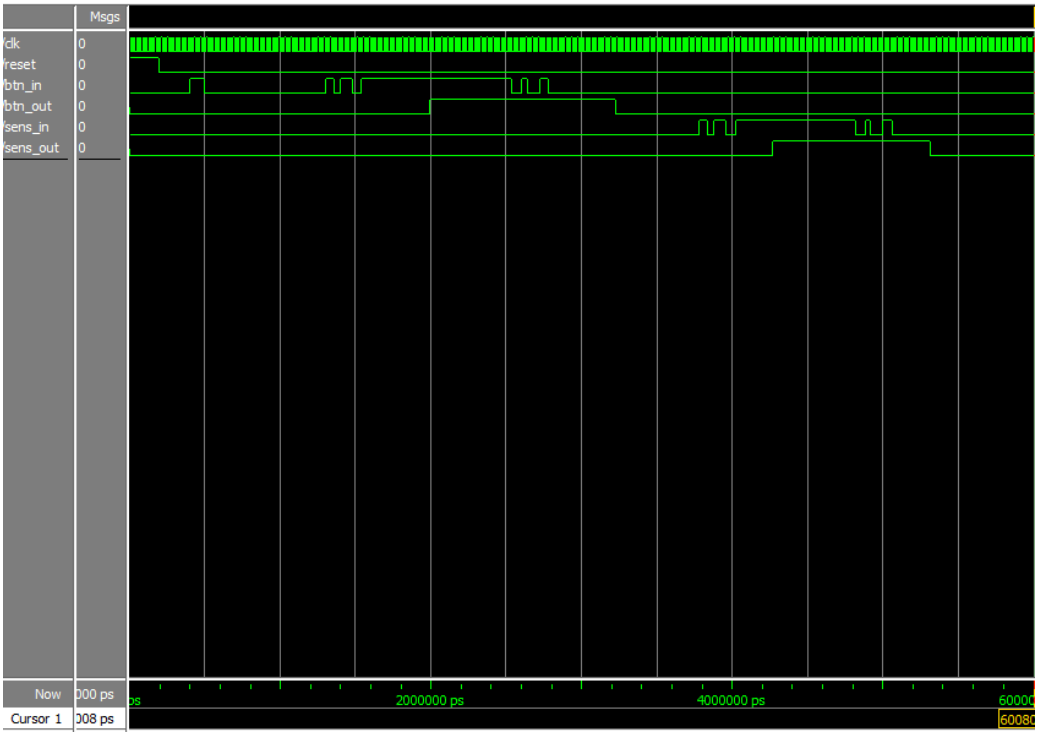


Figure 3.19 : Chronogramme de simulation du module Debouncer sous ModelSim

- Invalidation des rebonds (Signaux parasites) : Lors de l'injection d'impulsions rapides et instables simulant les rebonds mécaniques d'un bouton ou d'un capteur, le signal de sortie reste insensible et conserve son état précédent.
- Validation des signaux réels (Seuil temporel) : Seules les impulsions dont la durée continue dépasse le seuil configuré ($20ms$ pour les boutons et $5ms$ pour les capteurs) sont validées et transmises en sortie comme des signaux réels et stables.
- Comportement sélectif : Les chronogrammes obtenus sous ModelSim confirment le filtrage efficace du bruit d'entrée tout en garantissant la transmission propre des commandes vers le contrôleur principal.

3.5.2 Conception du contrôleur principal

Le contrôleur principal constitue le cerveau du système de commande de l'ascenseur. Son rôle majeur est d'assurer une gestion intelligente, fluide et sécurisée du déplacement de la cabine.

Outre le signal d'horloge système à $50MHz$ (clk), le contrôleur principal reçoit, après l'étape de traitement et conditionnement des signaux d'entrée, l'ensemble des signaux sous forme d'entrées exploitables. Il s'agit notamment du signal de réinitialisation (reset), des signaux des capteurs de position, des requêtes d'appels externes aux étages, ainsi que des requêtes de la cabine comprenant la sélection de la destination et les commandes manuelles d'ouverture et de fermeture de la porte.

Ces données d'entrée sont traitées de manière intelligente au sein du contrôleur principal. En réponse, ce dernier génère un ensemble de sorties décisionnelles :

- move et dir_out pour le pilotage et le contrôle du moteur de la cabine,
- door_open et door_close pour la commande du moteur de la porte,
- floor_out et dir_status pour le guidage des afficheurs et indicateurs d'état.

Toutes ces décisions et ordres de sortie sont ensuite acheminés vers les modules de commande des périphériques de sortie pour leur exécution physique.

Pour répondre aux exigences de sécurité, d'optimisation du temps d'attente et de confort des passagers, la conception de ce module repose sur une architecture modulaire en VHDL. Cette architecture combine des fonctions combinatoires avancées dédiées à l'analyse et à la priorisation des requêtes, associées à une Machine à États Finis (FSM) séquentielle pour le pilotage des actions (moteur, portes, temporisations).

Les sous-sections suivantes détaillent les composants clés de cette conception, depuis la gestion de l'initialisation jusqu'à l'implémentation de la FSM de contrôle.

3.5.2.1 Gestion de l'Initialisation et de l'Arrêt d'Urgence

Objectif : L'objectif de cette fonction est d'assurer la sécurité globale du système d'ascenseur en fournissant un mécanisme d'initialisation au démarrage et un dispositif d'arrêt d'urgence absolu en cas d'anomalie.

Principe : Le signal reset est associé à un commutateur physique (Slide Switch SW) de la carte DE1. Il agit comme une commande prioritaire asynchrone sur l'ensemble du contrôleur principal pour deux cas d'usage :

1. Initialisation au démarrage : Éliminer tout état logique indéterminé dans les bascules (Flip-Flops) de la FPGA lors de la mise sous tension, en ramenant le système vers un état initial sûr.
2. Arrêt d'urgence : En cas de dysfonctionnement ou de danger, l'activation du commutateur interrompt immédiatement le fonctionnement de l'ascenseur, bloque les moteurs, et refuse catégoriquement la réception de toute nouvelle requête d'étage tout en effaçant les demandes déjà enregistrées.

Logique de Réalisation : Dans le code VHDL du contrôleur principal, le signal reset est évalué au tout début du processus principal (process(clk, reset)). Lorsque reset = '1' :

- L'état futur de la machine à états FSM est forcé à l'état initial IDLE .
- Les signaux d'autorisation des moteurs (run) sont immédiatement forcés à '0'.
- Le registre de mémorisation des requêtes est réinitialisé à zéro (requetes <= (others => '0')), ce qui empêche l'enregistrement de tout nouvel appui sur les boutons d'appel.

Résultats et validation : La validation de cette fonction a été effectuée par simulation sous ModelSim puis confirmée lors du test pratique sur la carte DE1 :

- Arrêt immédiat du système : L'activation du commutateur SW (signal reset = '1') provoque un arrêt instantané et complet de l'ensemble du système, coupant immédiatement la commande des moteurs et réinitialisant tous les signaux internes du contrôleur.

3.5.2.2 Détection de l'Étage Courant (Décodage des Capteurs)

Objectif : Cette partie du code a pour rôle de déterminer en permanence la position réelle de la cabine d'ascenseur dans la cage. Elle traduit les signaux physiques provenant des quatre capteurs de position (sensor1 à sensor4) en une valeur numérique entière (current_floor variant de 1 à 4) exploitable par la FSM et les fonctions de recherche de requêtes.

Principe : Le processus repose sur une logique combinatoire asynchrone sensible aux changements d'état des quatre capteurs. Étant donné que les capteurs physiques utilisés fonctionnent en logique négative (actifs à l'état bas '0' lors du passage ou de la présence de la cabine), le processus évalue séquentiellement l'état de chaque capteur :

- Si sensor1 = '0', la cabine est au 1er étage → current_floor <= 1.
- Si sensor2 = '0', la cabine est au 2ème étage → current_floor <= 2.
- Si sensor3 = '0', la cabine est au 3ème étage → current_floor <= 3.
- Si sensor4 = '0', la cabine est au 4ème étage → current_floor <= 4.

Si aucun capteur n'est actif (par exemple pendant le déplacement de la cabine entre deux étages), le signal current_floor conserve sa dernière valeur attribuée grâce à la mémoire implicite du processus VHDL, garantissant ainsi la stabilité de l'information de position.

Logique de Réalisation : Le code VHDL implémentant cette détection est structuré sous forme d'un processus combinatoire indépendant (listing 3.1):

Listing 3.1 : Description VHDL du détection d'étage courant

```
-----------------------------------------------
-- Détection de l'étage courant (Floor Detection)
-----------------------------------------------
process(sensor1, sensor2, sensor3, sensor4)
begin
    if sensor1 = '0' then
        current_floor <= 1;
    elsif sensor2 = '0' then
        current_floor <= 2;
    elsif sensor3 = '0' then
        current_floor <= 3;
    elsif sensor4 = '0' then
        current_floor <= 4;
    end if;
end process;
```

Résultats et validation : La validation de ce bloc a été confirmée lors des tests pratiques sur le système :

- Précision du repérage : La détection est instantanée dès que la cabine atteint le capteur optique/mécanique d'un étage, mettant immédiatement à jour la valeur de current_floor.
- Immunité aux transitions inter-étages : Durant le déplacement entre deux niveaux (lorsqu'aucun capteur n'est au niveau bas '0'), la valeur du dernier étage franchi reste mémorisée sans générer d'état indéterminé ou de valeur aberrante.
- Synthèse matérielle : Le compilateur Quartus II a synthétisé ce processus sous forme d'un multiplexeur combinatoire prioritaire très simple, ne consommant qu'un nombre minime d'éléments logiques (LEs) sur la carte FPGA DE1.

3.5.2.3 Mémorisation, Traitement Parallèle et Effacement des Requêtes

Objectif : L'objectif de cette fonction est de capturer en permanence toutes les demandes d'étages (internes et externes), de les maintenir en mémoire quel que soit l'état de l'ascenseur, d'évaluer la priorité du déplacement à chaque coup d'horloge ($20ns$), et d'exécuter un effacement sélectif intelligent une fois l'étage cible atteint.

Principe : La gestion des requêtes repose sur un fonctionnement découpé en trois phases principales :

1. Mémorisation continue et asynchrone :
   L'enregistrement des requêtes est totalement indépendant et s'effectue à tout moment, quel que soit l'état actuel de la cabine. Chaque demande possède un bit spécifique qui lui est dédié. L'appui sur un bouton met le bit correspondant au niveau logique '1' pour sauvegarder la demande.
2. Traitement parallèle et synchrone à haute vitesse ($50MHz$) :
   À chaque front montant du signal d'horloge (soit toutes les $20ns$), le système effectue le traitement des demandes. Toute nouvelle demande est prise en compte de manière quasi instantanée pour mettre à jour la décision. Le traitement des requêtes s'effectue par la comparaison des demandes à l'aide de fonctions spécifiques, puis par la détermination des priorités et de la décision finale au sein de la machine à états (FSM).
3. Effacement sélectif par remise à zéro ('0') :
   Pour éviter d'entrer dans une boucle infinie de traitement, la requête satisfaite doit être remise à zéro ('0'). Lors de l'arrivée à l'étage, le système vérifie s'il existe une seule demande externe (soit uniquement une montée, soit uniquement une descente) ou si les deux demandes existent simultanément (montée et descente) :
   - Effacement des requêtes de cabine : La demande interne de l'étage courant est systématiquement effacée en la mettant à '0'.
   - Effacement des requêtes externes : S'il n'y a qu'une seule demande externe, les signaux de requêtes externes pour cet étage sont directement réinitialisés à '0'.

Si les deux demandes existent en même temps, le système remet à '0' uniquement la requête qui correspond au sens actuel du déplacement de l'ascenseur, afin de ne pas perdre la demande de la personne souhaitant aller dans la direction opposée.

Logique de Réalisation : Dans le code VHDL, la mémorisation s'appuie sur trois vecteurs de signaux déclarés pour les 4 étages (listing 3.2):

Listing 3.2 : Structure VHDL des registres de mémorisation des requêtes

```
signal req_up : std_logic_vector(1 to 4) := (others => '0');
signal req_down : std_logic_vector(1 to 4) := (others => '0');
signal req_cab : std_logic_vector(1 to 4) := (others => '0');
```

Lors de l'arrivée à l'étage cible (current_floor), l'effacement sélectif s'effectue selon la structure exacte suivante (listing 3.3):

Listing 3.3 : Description VHDL d'effacement des requêtes

```
req_cab(current_floor) <= '0';

if req_up(current_floor) = '1' and req_down(current_floor) = '1' then
    req_up(current_floor) <= '0';
else
    req_up(current_floor)   <= '0';
    req_down(current_floor) <= '0';
end if;
```

Résultats et validation : La validation de cette fonction a été confirmée lors des tests pratiques sur le système :

- Mémorisation continue : Lors de l'essai de l'ascenseur, nous avons constaté qu'il enregistre correctement nos demandes quel que soit son état de fonctionnement.
- Mise à jour dynamique de la destination : Le système met à jour la destination en temps réel, dès qu'une pression est effectuée sur un bouton d'appel.
- Absence de boucle infinie : L'ascenseur ne rentre jamais dans une boucle infinie de traitement d'un même appel, ce qui confirme l'effacement effectif et automatique de la requête dès sa satisfaction.
- Préservation des appels de sens opposé : Lors de la simulation de deux demandes inverses au même étage, le système satisfait le trajet actuel tout en conservant la requête en sens inverse pour le déplacement suivant.

3.5.2.4 Fonctions de Recherche et de Comparaison des Requêtes

Objectif : L'objectif de ces fonctions auxiliaires (Utility Functions) est d'analyser en permanence et en parallèle le registre des requêtes afin d'identifier la cible la plus proche ou prioritaire selon l'état et la direction actuelle de la cabine, et de fournir ces résultats à la machine à états (FSM).

Principe : Pour optimiser les déplacements de l'ascenseur et éviter les arrêts inutiles ou les trajets contradictoires, cinq fonctions logiques combinatoires ont été développées. Elles reposent sur des boucles statiques (1 to 4) directement synthétisables sur FPGA :

1. nearest_overall (Recherche globale depuis l'état de repos IDLE) : Détermine l'étage demandé le plus proche de la position actuelle (cf), quel que soit le type de requête (montée, descente ou cabine). Elle explore d'abord l'étage courant, puis s'éloigne progressivement (distance d = 1, 2, 3) de manière symétrique pour minimiser le temps d'attente initial.
2. nearest_above_up (Recherche prioritaire en montée) : Recherche l'étage le plus proche au-dessus du niveau actuel (i > cf) ayant un appel montant (up) ou une demande cabine (cab). Si aucun appel de ce type n'existe mais qu'il reste un appel descendant (dn) plus haut, la fonction cible l'appel descendant le plus éloigné (le plus haut) pour effectuer le demi-tour de manière optimale.
3. nearest_below_down (Recherche prioritaire en descente) : Fonction symétrique à la précédente. Recherche l'étage le plus proche en dessous du niveau actuel (i < cf) ayant un appel descendant (dn) ou cabine (cab). En l'absence de tels appels, elle vise l'appel montant le plus bas en dessous.
4. any_above et any_below (Détection de présence de requêtes) : Fonctions booléennes d'évaluation rapide indiquant s'il existe au moins une requête active (tous types confondus) respectivement au-dessus ou en dessous de l'étage actuel. Elles sont directement exploitées par la FSM pour décider du maintien de la trajectoire, du changement de direction ou pour entrer en état de repos (IDLE).

Les fonctions nearest_overall, nearest_above_up et nearest_below_down renvoient la valeur de l'étage cible, tandis que les fonctions any_above et any_below renvoient une valeur booléenne (true ou false).

Résultats et validation : La validation de cette fonction a été confirmée lors des tests pratiques sur le système :

- En état de repos (IDLE) : L'ascenseur se dirigeait toujours vers l'étage demandé le plus proche de sa position actuelle.
- En phase de montée : L'ascenseur s'arrêtait systématiquement à la demande interne ou externe de montée la plus proche au-dessus de l'étage actuel. En l'absence de ces deux types de requêtes, il poursuivait sa course jusqu'à la demande externe de descente la plus éloignée au-dessus du niveau courant.
- En phase de descente : Le même comportement symétrique a été observé : arrêt à la demande la plus proche en descente (interne ou externe), ou à la demande de montée la plus éloignée en dessous en cas d'absence des deux premières.
- Gestion globale du trajet : Lors de la montée, l'ascenseur répondait à toutes les requêtes situées au-dessus de lui, quel qu'en soit le type. Lorsqu'il n'existait plus aucune requête au-dessus, il changeait de direction pour satisfaire les appels situés en

dessous. Enfin, lorsqu'aucune requête n'était présente ni au-dessus ni en dessous, le système basculait automatiquement en état de repos (IDLE).

3.5.2.5 Machine à États Finis (FSM du Contrôleur Principal)

Objectif : La Machine à États Finis (FSM) constitue le cœur décisionnel du contrôleur d'ascenseur. Elle orchestre la logique de déplacement de la cabine, la gestion de la sécurité, le temporisage et le contrôle d'ouverture/fermeture des portes en exploitant les résultats des fonctions combinatoires d'analyse de requêtes.

Principe : La FSM est composée de 8 états principaux (figure 3.20) assurant un fonctionnement fluide et sécurisé :

1. IDLE (État de repos) :
   La cabine est immobile (move <= '0') sans direction prioritaire (direction <= DIR_NONE). Elle surveille en permanence l'arrivée d'une requête :
   - Si le bouton d'ouverture manuelle de la porte est pressé, la FSM passe à DOOR_OPENING.
   - En cas de requête détectée via nearest_overall, si l'étage cible f est supérieur au niveau actuel, elle passe en MOVING_UP et la direction prioritaire devient le haut (DIR_UP) ; s'il est inférieur, elle passe en MOVING_DOWN et la direction prioritaire devient le bas (DIR_DOWN). Si la requête provient de l'étage courant ($f = current_floor$), elle efface les requêtes de cet étage et passe directement à STABILITY.
2. MOVING_UP (Déplacement en montée) :
   Active le moteur en sens montant (dir_out <= '1', move <= '1') et met à jour la direction prioritaire vers le haut (direction <= DIR_UP). La FSM met à jour continuellement l'étage cible grâce à nearest_above_up. Dès que l'étage courant atteint l'étage cible ($current_floor = target_floor$), le moteur s'arrête (move <= '0'), la requête servie est effacée et le système passe à STABILITY.
3. MOVING_DOWN (Déplacement en descente) :
   Fonctionnement symétrique à la montée (dir_out <= '0', move <= '1') en mettant à jour la direction prioritaire vers le bas (direction <= DIR_DOWN). La cible est réévaluée par nearest_below_down. À l'arrivée, la requête servie est effacée avant de passer à STABILITY.
4. STABILITY (Stabilisation à l'arrivée) :
   Marque un temps d'arrêt de sécurité pour s'assurer de l'immobilisation complète du moteur avant d'ouvrir la porte (temporisation d'environ 1 seconde via un compteur d'horloge timer < 50000000).
5. DOOR_OPENING (Ouverture de la porte) :
   Commande l'ouverture active de la porte (door_open <= '1', door_close <= '0') pendant une durée déterminée (temporisation de 2 secondes), puis bascule vers l'état d'attente WAITING.
6. WAITING (Attente porte ouverte) :
   Maintient la porte ouverte pendant la phase d'embarquement/débarquement des

passagers (temporisation normale de 6 secondes). Si le bouton de fermeture forcée (door_close_btn) est actionné, l'attente est interrompue immédiatement pour passer à DOOR_CLOSING.

7. DOOR_CLOSING (Fermeture de la porte) :
   Commande la fermeture active de la porte (door_close <= '1'). Par mesure de sécurité, si un passager appuie sur le bouton d'ouverture (door_open_btn), la fermeture est immédiatement annulée et la porte se rouvre (DOOR_OPENING). Une fois la fermeture terminée (2 secondes), la FSM passe à DECISION.
8. DECISION (Prise de décision post-fermeture) :
   Évalue la présence d'autres requêtes pour déterminer la suite du trajet :
   - Priorité absolue à l'ouverture manuelle si le bouton d'ouverture est pressé.
   - Si des requêtes existent à la fois au-dessus et en dessous (any_above et any_below), la FSM maintient sa direction précédente pour éviter les changements brusques de sens, en passant à MOVING_UP si la direction prioritaire est vers le haut ou à MOVING_DOWN si la direction prioritaire est vers le bas.
   - Si des requêtes existent uniquement au-dessus ou en dessous, elle s'oriente respectivement vers MOVING_UP ou MOVING_DOWN.
   - En l'absence totale de requête, elle retourne à l'état de repos IDLE.

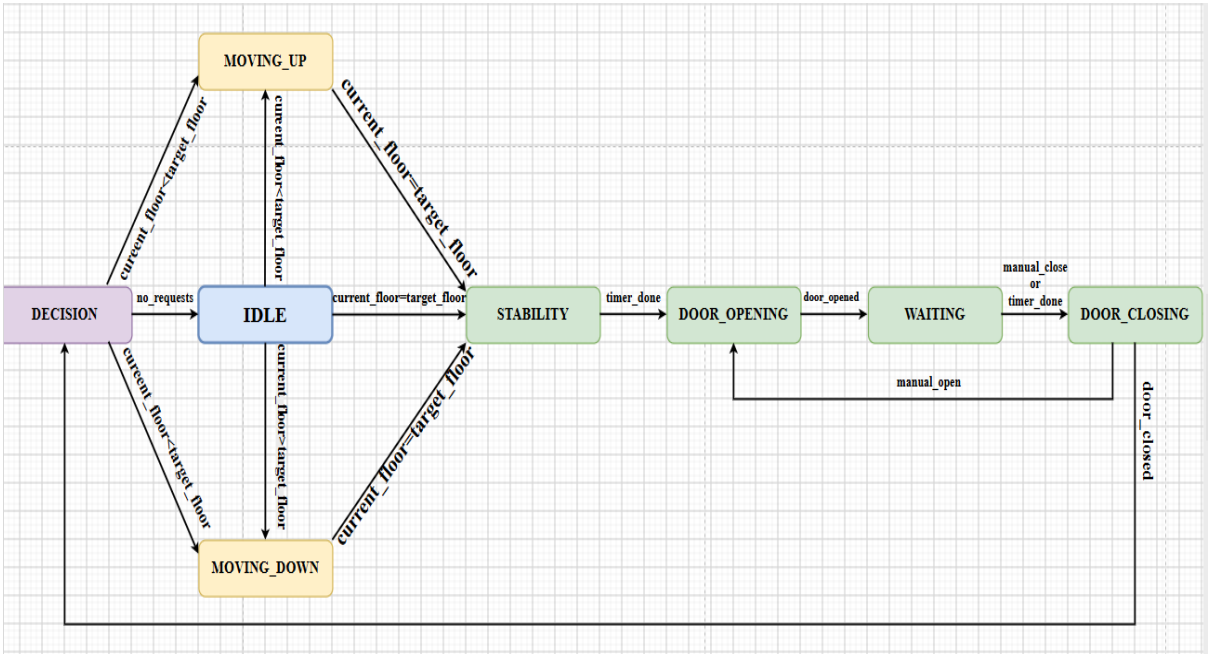


Figure 3.20 : Architecture du FSM du système de contrôle de l'ascenseur

Logique de réalisation : Le code VHDL de la FSM est implémenté sous forme d'un processus synchrone cadencé par l'horloge système et réagissant sur l'instruction ‘case state is’ (voir Annexe A).

Résultats et validation : Les tests pratiques effectués sur la maquette physique ont validé le comportement de la FSM :

- Fluidité des transitions : Le passage entre les états de déplacement, de stabilisation et de gestion de la porte s'effectue sans retards ni états de blocage (deadlocks).
- Priorités et sécurité : L'ouverture manuelle de la porte pendant la fermeture réagit instantanément, évitant tout risque de coincement. De plus, comme mesure de sécurité fondamentale, la porte ne s'ouvre jamais lorsque l'ascenseur est en mouvement (MOVING_UP ou MOVING_DOWN).
- Optimisation du temps et confort : Il est possible de fermer la porte manuellement à l'aide du bouton dédié avant l'expiration de la temporisation automatique, ce qui permet un gain de temps significatif pour les utilisateurs.
- Respect des temporisations : Les temps de pause pour la stabilisation, la durée d'ouverture et d'attente se déroulent de façon très précise grâce aux compteurs synchronisés sur le cycle d'horloge système.

3.5.3 Conception des Modules de Commande des Périphériques de Sortie

Après avoir traité les signaux d'entrée et élaboré les décisions au niveau du contrôleur principal, il est nécessaire de traduire ces ordres en signaux électriques adaptés aux équipements physiques de l'ascenseur. En effet, les composants externes tels que les moteurs et l'afficheur nécessitent des séquences de commande spécifiques que le FPGA ne peut pas générer directement sans une logique intermédiaire. Cette étape consiste donc à concevoir des modules de commande dédiés pour chaque périphérique de sortie, afin d'assurer le déplacement de la cabine ainsi que l'ouverture et la fermeture de la porte avec un mouvement fluide et une vitesse optimale, et un affichage correct des informations.

3.5.3.1 Conception du module de commande du moteur pas à pas de la cabine

Objectif : L'objectif de ce module est de générer un signal d'impulsion permettant de commander le moteur pas à pas (NEMA 17) de la cabine. Il permet d'activer ou d'arrêter la rotation du moteur tout en contrôlant sa vitesse de déplacement de manière précise à partir des ordres envoyés par le contrôleur principal.

Principe : Le moteur pas à pas nécessite un signal rectangulaire d'impulsions pour tourner, où chaque front montant du signal fait avancer le moteur d'une seule pas. Pour notre projet, la vitesse de $75tr/min$ a été choisie comme vitesse idéale pour le déplacement de la cabine. Le principe consiste à calculer la période nécessaire de ce signal pour atteindre cette vitesse, puis à déterminer la valeur du diviseur (DIVISOR) à partir de l'horloge du FPGA.

En mode Full Step, comme utilisé dans notre projet, l'angle d'un pas pour le moteur NEMA 17 est de $1,8^{\circ}$, ce qui signifie qu'il faut $200\ pas$ pour effectuer un tour complet (360°).

1. Calcul du temps d'un pas (Période du signal) :

- Pour tourner à $75tr/min$, le moteur doit effectuer :

- $75 \times 200 = 15000\ pas\ par\ minute$
- En divisant par $60secondes$, cela équivaut à :

- $\frac{15000}{60} = 250\ pas\ par\ seconde$
- Le temps nécessaire pour effectuer un seul pas est donc :

- $T_{pas} = \frac{1s}{250} = 4ms$
- Pour générer ce signal carré de $4ms$, le signal doit changer d'état (basculer entre le niveau HAUT et BAS) toutes les $2ms$.

2. Calcul de la valeur du diviseur (DIVISOR) :

- L'horloge du FPGA fonctionne à $50MHz$, ce qui signifie qu'elle génère un front montant toutes les $20ns$.
- Pour obtenir une demi-période de $2ms$, nous calculons le nombre d'incrémentations nécessaires :

- $DIVISOR = \frac{2ms}{20ns} = \frac{0{,}002s}{0{,}00000002s} = 100000$

Ainsi, un compteur s'incrémente à chaque front montant de l'horloge du système. Lorsqu'il atteint la valeur 100000, le signal inverse son état et le compteur se réinitialise, générant un signal carré d'une période de $4ms$ qui fait tourner le moteur à la vitesse exacte de $75tr/min$.

Logique de réalisation (listing 3.4) **:** Le module reçoit le signal de commande (run) en provenance du contrôleur principal. Lorsque ce signal est à '1', le module fait tourner le moteur à la vitesse requise en générant un signal (step) de type PWM. Pour cela, un compteur counter s'incrémente à chaque coup d'horloge. Chaque fois que le compteur atteint la valeur DIVISOR =100000 , le module inverse l'état du signal step et réinitialise le compteur à zéro, ce qui produit la fréquence exacte nécessaire. Si le signal (run) est à '0', le signal step prend immédiatement la valeur '0'. Ce signal est ensuite envoyé au moteur via la sortie (step_out) .

Listing 3.4 : Description VHDL du module de commande du moteur pas à pas de la cabine

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity stepper_motor_driver is
    Port (
        clk      : in  std_logic;
        reset    : in  std_logic;
        run      : in  std_logic;
```

```
        step_out : out std_logic
    );
end stepper_motor_driver;

architecture Behavioral of stepper_motor_driver is

    signal counter : integer := 0;
    signal step    : std_logic := '0';

    constant DIVISOR : integer := 100000;

begin

    process(clk, reset)
    begin
        if reset = '1' then
            counter <= 0;
            step <= '0';

        elsif rising_edge(clk) then

            if run = '1' then
                counter <= counter + 1;

                if counter = DIVISOR then
                    step <= not step;
                    counter <= 0;
                end if;

            else
                step <= '0';
                counter <= 0;
            end if;
        end if;
    end process;

    step_out <= step;

end Behavioral;
```

Résultats et validation : Afin de valider le fonctionnement du module stepper_motor_driver, nous avons effectué une simulation temporelle sous l'environnement ModelSim (figure 3.21).

Les résultats de la simulation ont permis de confirmer les points suivants :

1. Génération de la fréquence exacte : En mesurant la durée entre deux fronts montants successifs du signal (step_out) , nous avons obtenu une période exacte de $4ms$ (soit une demi-période de $2ms$ pour 100000 impulsions d'horloge), ce qui correspond parfaitement à la vitesse de rotation ciblée de $75tr/min$.

2. Contrôle par le signal (run) : Lorsque le signal (run) passe à '0', la génération d'impulsions s'arrête immédiatement et le signal (step_out) est maintenu à l'état bas '0', garantissant un arrêt net du moteur.

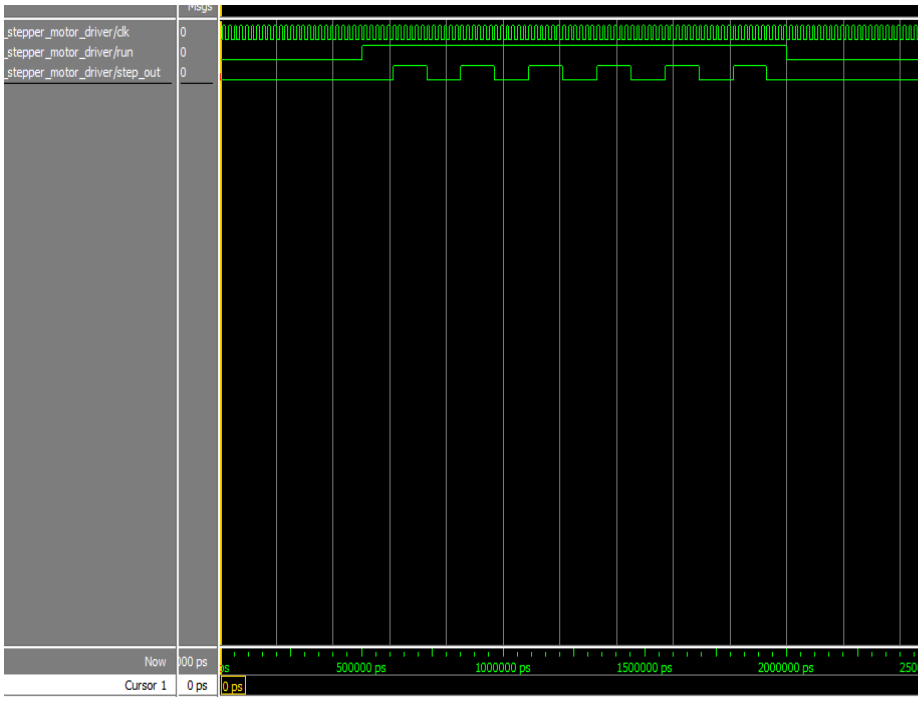


Figure 3.21 : Chronogramme de simulation du module de commande du moteur pas à pas de la cabine sous ModelSim

3. Validation expérimentale réelle : Lors du test physique direct du module raccordé au driver DRV8825 et au moteur NEMA 17, le moteur a répondu instantanément aux commandes. Il a exécuté la rotation avec une vitesse fluide, stable et sans aucune perte de pas.

3.5.3.2 Conception du module de commande du servomoteur de la porte

Objectif : Le module de commande du servomoteur a pour rôle de contrôler l'ouverture et la fermeture de la porte de la cabine. Il génère un signal de modulation d'impulsion en largeur (PWM - Pulse Width Modulation) d'une fréquence de $50Hz$ (période de $20ms$). Afin de garantir un mouvement doux, fluide et d'éviter les secousses mécaniques, le module intègre un profil de vitesse trapézoïdal (accélération, vitesse constante, décélération) assurant une transition progressive entre la position fermée (0°) et la position ouverte (90°) sur une durée totale de $2secondes$.

Principe : Le servomoteur est piloté par la largeur de l'impulsion haute du signal PWM :

- Position Porte Fermée (0°) : Impulsion de $0.5ms$ (POS_0_DEG = 25 000 cycles d'horloge à $50MHz$).
- Position Porte Ouverte (90°) : Impulsion de $1.5ms$ (POS_90_DEG = 75 000 cycles d'horloge à $50MHz$).

Pour parcourir la différence de largeur d'impulsion (50000 pas), la course est divisée en trois phases distinctes afin de former une trajectoire douce :

1. Phase d'accélération (N_ACCEL = 12 500 pas) : Vitesse initiale lente réglée par un délai DELAY_ACCEL = 2400 cycles par pas.
2. Phase à Vitesse Constante (N_CONST = 25 000 pas) : Vitesse maximale réglée par un délai plus court DELAY_CONST = 1600 cycles par pas.
3. Phase de décélération (N_ACCEL = 12 500 pas) : Ralentissement progressif avant l'arrêt complet en réappliquant le délai DELAY_ACCEL = 2400 cycles par pas.

Résultats et validation : La validation du module de commande du servomoteur a été effectuée à la fois par simulation logicielle sous ModelSim et par des tests pratiques sur la maquette physique :

- Validation par simulation (ModelSim) :
  Les chronogrammes obtenus lors de la simulation confirment le parfait fonctionnement de la logique PWM progressive :
  - Lors de l'ordre d'ouverture de la porte “open_door = '1'”(figure 3.22), la largeur de l'impulsion du signal pwm_out augmente de façon fluide et progressive jusqu'à atteindre la valeur correspondant à la position 90° (POS_90_DEG).

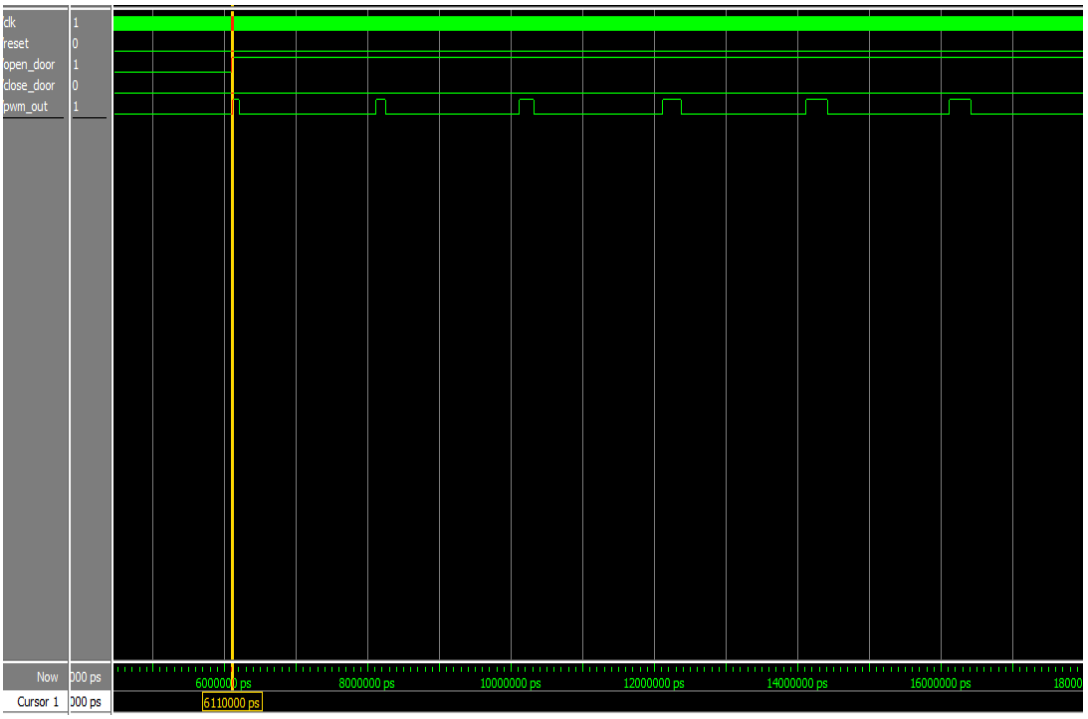


Figure 3.22 : Chronogramme de simulation d’ouverture de la porte sous ModelSim

  - À l'inverse, lors de la commande de fermeture “close_door = '1'”(figure 3.23), la largeur de l'impulsion diminue progressivement jusqu'à revenir à sa valeur minimale de 0° (POS_0_DEG).

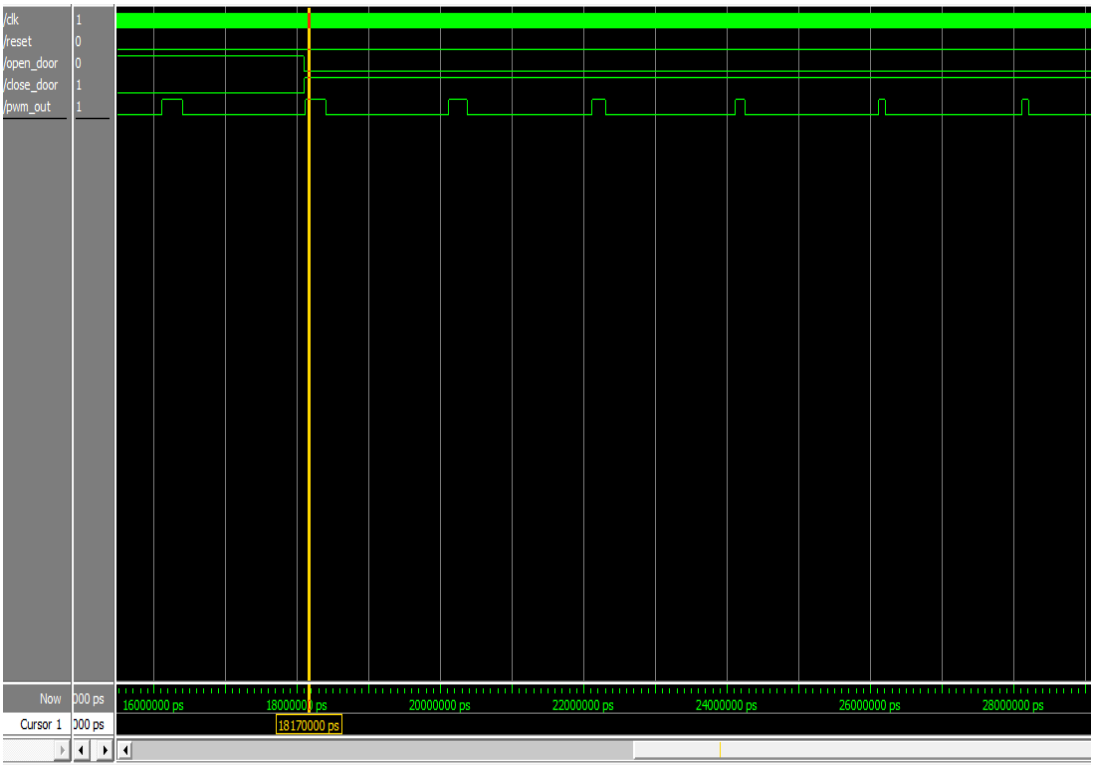


Figure 3.23 : Chronogramme de simulation de fermeture de la porte sous ModelSim

- Les variations de largeur observées sur les signaux de simulation traduisent avec précision l'application du profil de vitesse trapézoïdal (accélération, vitesse constante, décélération).

- Validation expérimentale sur la maquette réelle :
  Les essais pratiques réalisés directement sur la maquette de l'ascenseur ont confirmé les performances attendues :
  - Mouvement fluide et silencieux : Grâce à l'augmentation et à la diminution progressives du rapport cyclique, le servomoteur actionne le mécanisme de la porte sans à-coups mécaniques ni vibrations.
  - Sécurité et protection mécanique : Le ralentissement progressif en fin de course (décélération) évite tout choc brutal contre les butées physiques lors de l'ouverture complète ou de la fermeture totale.
  - Réactivité aux commandes : En cas d'interruption manuelle ou d'ordre inverse du contrôleur principal, le servomoteur réagit instantanément en inversant le sens de variation de la largeur d'impulsion sans aucun blocage.

3.5.3.3 Conception du driver d'affichage

Objectif : Le module seven_segment_driver constitue l'interface visuelle d'information pour les usagers de l'ascenseur. Son rôle principal est de convertir les signaux numériques issus du contrôleur principal (l'étage courant (floor) et la direction prioritaire (direction) en codes d'affichage compatibles avec quatre afficheurs 7 segments à anode commune (HEX0 à HEX3) présents sur la carte DE1.

Principe : Les afficheurs 7 segments utilisés fonctionnent en logique active à basse (un segment s'allume lorsque sa commande est à '0' et s'éteint lorsqu'elle est à '1'). La répartition des afficheurs est organisée comme suit :

1. Affichage du Numéro de l'Étage (HEX0 et HEX1) :
    - HEX0 affiche la valeur de l'étage courant (1, 2, 3 ou 4). Notez que le codage est adapté à l'affichage souhaité (ex: le chiffre 0 sur HEX0 pour représenter le niveau de base 1, ou la numérotation correspondante aux besoins de l'IHM).
    - HEX1 est fixe à la valeur "1000000" (affichage permanent du chiffre 0), formant ainsi un affichage combiné sur deux digits.
2. Affichage de la Direction de Déplacement (HEX2 et HEX3) :
    - Direction Montante (direction = "01") : Activation de combinaisons de segments spécifiques sur HEX2 ("1001100") et HEX3 ("1011000") pour former une flèche indiquant la montée (UP).
    - Direction Descendante (direction = "10") : Activation de combinaisons sur HEX2 ("1000011") et HEX3 ("1100001") pour former le symbole de descente (DOWN).
    - État de Repos ou Sans Direction (direction = "00" ou autre) : HEX2 et HEX3 affichent le code "0111111" (allumage du segment central G uniquement), formant un tiret horizontal (-) de chaque côté pour indiquer l'immobilité.

Logique de réalisation : La conception repose sur une architecture combinatoire simple et efficace utilisant deux processus indépendants réagissant aux changements des signaux d'entrée (Voir Annexe B).

Résultats et validation : La validation du module seven_segment_driver a été effectuée par des tests pratiques sur la carte DE1, en exploitant ses quatre afficheurs 7 segments intégrés (HEX0 à HEX3) :

- Exactitude et suivi de la position : Au démarrage (étage de base), l'afficheur HEX0 indique clairement le chiffre 0. Lors du déplacement de la cabine, la valeur affichée reste stable jusqu'à l'atteinte ou le passage au niveau supérieur, où elle bascule instantanément vers 1, puis 2, puis 3 de manière très précise et parfaitement synchronisée avec la position réelle de la cabine.
- Indication dynamique de la direction : L'activation des symboles sur HEX2 et HEX3 réagit immédiatement aux changements du signal direction émis par le contrôleur principal, permettant d'indiquer en temps réel le sens de déplacement (montée UP, descente DOWN, ou état de repos --).

3.5.4 Intégration des Modules

Objectif : L'objectif de l'intégration des modules est d'établir une liaison logique entre ces différents modules logiciels VHDL afin de permettre le traitement et l'échange fluide des signaux et des informations. Cette interconnexion est essentielle pour assembler les

sous-modules individuels et former une architecture globale cohérente, fonctionnelle et capable de piloter l'ascenseur dans son ensemble.

Principe : Le principe repose sur la création d'un module de haut niveau appelé Top-Level Entity (nommé elevator_top), qui agit comme le châssis d'interconnexion principal du système (figure 3.24):

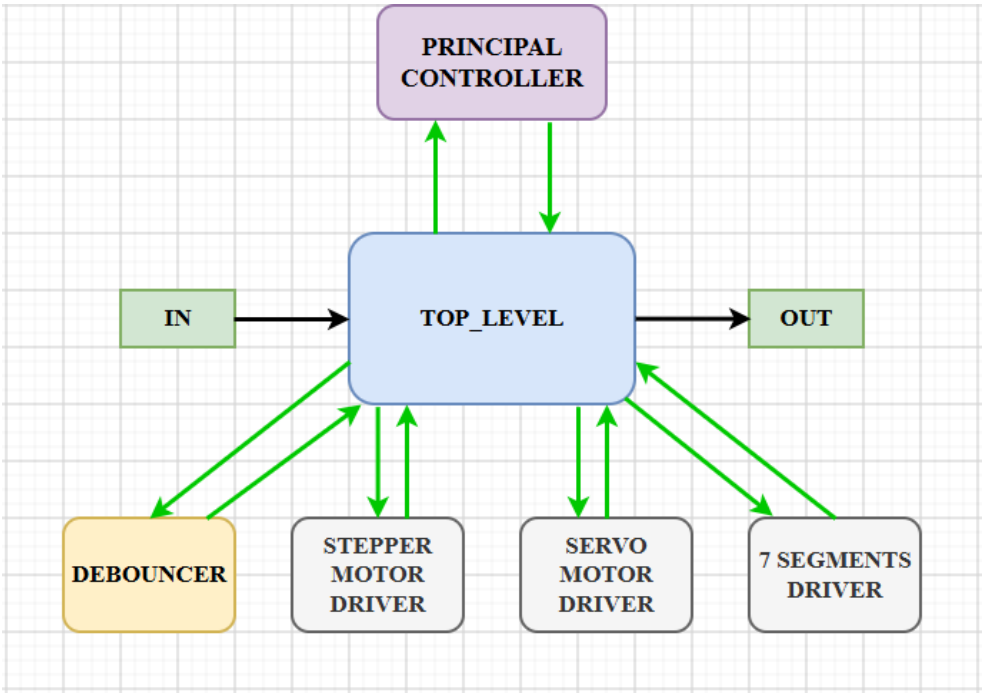


Figure 3.24 : Schéma d'interconnexion des modules du système

1. Réception des Entrées Externes : Le Top-Level capte l'ensemble des signaux bruts provenant du monde extérieur (boutons-poussoirs, capteurs de position, commutateur de reset et horloge globale $50MHz$) via les broches physiques du FPGA (GPIO_0, SW, CLOCK_50).
2. Interconnexion par Signaux Internes (signal) : Le Top-Level déclare un ensemble de signaux internes (tels que call1_up_c, move_utile_sig, open_door_sig, floor_sig, etc.) qui servent de canaux de communication pour échanger des données et des ordres de commande d'un module à un autre (par exemple, transmettre les signaux filtrés du debouncer vers le controller, ou les ordres du controller vers les drivers de moteurs et d'affichage).
3. Transmission des Sorties Externes : Enfin, le module Top-Level achemine les signaux de commande générés par les différents sous-modules vers les broches physiques de la carte FPGA (GPIO_1 et HEX0..HEX3) afin de piloter directement les actionneurs et périphériques physiques (moteur pas-à-pas, servomoteur et afficheurs 7-segments).

Résultats et validation : Lors des tests sur la maquette, la réaction des sorties (moteurs et afficheurs) correspond parfaitement aux signaux injectés en entrée (boutons et capteurs), selon la logique exacte définie dans le contrôleur. Cette correspondance parfaite entre les

entrées et les sorties prouve que la communication entre les différents sous-modules s'effectue sans erreur et que l'interconnexion globale a été réalisée avec succès.

3.6 Résultats Obtenus et Évaluation du Système

Cette section est dédiée à la présentation des résultats expérimentaux et à l'évaluation globale des performances du système d'ascenseur intelligent. Après l'implémentation matérielle et logicielle sur la carte FPGA DE1, il convient de valider le comportement du prototype en conditions réelles. Nous analysons d'abord les performances fonctionnelles de la cabine, des portes et des interfaces d'affichage. Ensuite, nous évaluons la robustesse de la logique de commande à travers différents scénarios de trafic complexes (gestion prioritaire des appels, prévention de la saturation et traitement des erreurs d'utilisation), afin de confirmer la fiabilité et l'efficacité du système développé.

3.6.1 Performances fonctionnelles du système

Cette sous-section détaille l'ensemble des performances fonctionnelles observées lors des essais pratiques du prototype d'ascenseur intelligent. Nous y récapitulons les résultats mesurés sur les différents organes du système, à savoir la stabilité dynamique du déplacement de la cabine et sa précision d'arrêt, la sécurité et l'automatisme du mécanisme des portes, la réactivité de l'interface d'affichage (IHM), ainsi que l'efficacité du filtrage numérique des entrées et la robustesse globale du code VHDL embarqué.

3.6.1.1 Contrôle Dynamique et Stabilité de la Cabine

- **Guidage et Alignement** : Mouvement stable et parfaitement aligné de la cabine sans jeu mécanique, grâce aux rails de guidage (guide rails).
- **Régularité du Mouvement** : Déplacement fluide à vitesse constante, exempt de vibrations ou d'à-coups, assuré par le pilotage du moteur.
- **Précision d'Arrêt** : Haltes précises au niveau de chaque étage, déclenchées par les capteurs de positionnement.
- **Sécurité Anti-Défaillance Capteur** : Arrêt de sécurité préventif de la cabine avant d'atteindre l'étage cible en cas de panne ou de non-détection du capteur correspondant.
- **Arrêt d'Urgence** : Interruption immédiate et totale de tout le système lors de l'activation du bouton d'arrêt d'urgence.

3.6.1.2 Gestion Automatisée et Sécurisée des Portes

- **Verrouillage en Mouvement** : Maintien impératif des portes fermées pendant le déplacement de la cabine.
- **Séquence de Sécurité au Débarquement** : Stabilisation de la cabine portes fermées pendant un court instant avant l'ouverture automatique lors de l'arrivée à une destination spécifique.
- **Ouverture Automatique** : Ouverture automatique des portes dès l'arrivée et la stabilisation de la cabine à l'étage cible.

- **Profil de Vitesse Profilé (S-Curve)** : Ouverture et fermeture fluides avec phases d'accélération et de décélération progressives.
- **Feedback et Précision Positionnelle** : Positionnement exact des portes en fin de course grâce à la boucle de rétroaction interne du servomoteur.
- **Automatisme de Fermeture et Commandes Manuelles** :
    - Fermeture automatique après une temporisation prédéfinie.
    - Possibilité de forcer la fermeture via le bouton dédié avant la fin du temporisateur.
    - Réouverture immédiate via le bouton d'ouverture pendant la phase de fermeture ou en état de veille (idle).
    - Ouverture directe et instantanée en cas d'appel émis depuis l'étage où stationne déjà la cabine.

3.6.1.3 Signalisation et Interface Homme-Machine (IHM)

- **Affichage Dynamique de l'État** : Restitution précise et en temps réel de l'étage courant et du sens de déplacement (flèches de direction) sur l'écran d'affichage.

3.6.1.4 Traitement des Entrées et Réactivité Système

- **Réactivité Instantanée** : Prise en compte immédiate des requêtes utilisateurs dès l'appui sur les boutons d'appel.
- **Immunité aux Bruits d'Appui (Debouncing)** : Filtrage efficace des rebonds mécaniques (anti-rebond), garantissant la prise en compte exclusive des demandes réelles des utilisateurs tout en éliminant les faux déclenchements dus aux rebonds.

3.6.1.5 Robustesse et Stabilité du Code Embarqué

- **Fiabilité Algorithmique** : Absence totale de blocages (deadlocks) ou de boucles infinies.
- **Gestion Fait-Garantie des Requêtes** : Traitement intégral de l'ensemble des requêtes sans perte ni omission en mémoire.

3.6.2 Évaluation de la Gestion Intelligente

Pour évaluer l'efficacité de la gestion intelligente de notre ascenseur, nous avons soumis le système à une série de scénarios de fonctionnement réels et complexes. L'objectif est d'analyser la pertinence des prises de décision de l'algorithme face à des appels multiples et simultanés.

Scénario 1 : Optimisation de Trajet et Gestion Prioritaire des Appels

a. Sollicitations des Utilisateurs et État Initial : Au départ, la cabine se trouve en stationnement à l'étage 0 (E_0). Trois requêtes d'utilisateurs sont générées :

- Utilisateur 1 (E_0) : Embarque à l'étage 0 et demande à monter à l'étage 3 (E_3).

- Utilisateur 2 (E_1) : Émet un appel depuis l'étage 1 pour monter à l'étage 3 (E_3).
- Utilisateur 3 (E_2) : Émet un appel depuis l'étage 2 pour descendre à l'étage 0 (E_0).

b. Trajet Effectué et Comportement du Système : Après l'embarquement de l'Utilisateur 1 à E_0, la cabine démarre sa montée vers E_3. En cours de route, la logique intelligente traite l'appel de l'Utilisateur 2 en effectuant un arrêt intermédiaire à E_1. Après l'embarquement de ce dernier et la confirmation de sa destination (E_3), la cabine poursuit sa montée directe jusqu'à E_3 pour débarquer les Utilisateurs 1 et 2. Une fois la montée finalisée, le système inverse son sens de déplacement et se dirige vers E_2 pour faire embarquer l'Utilisateur 3, puis l'achemine directement vers sa destination finale à E_0 (figure 3.25).

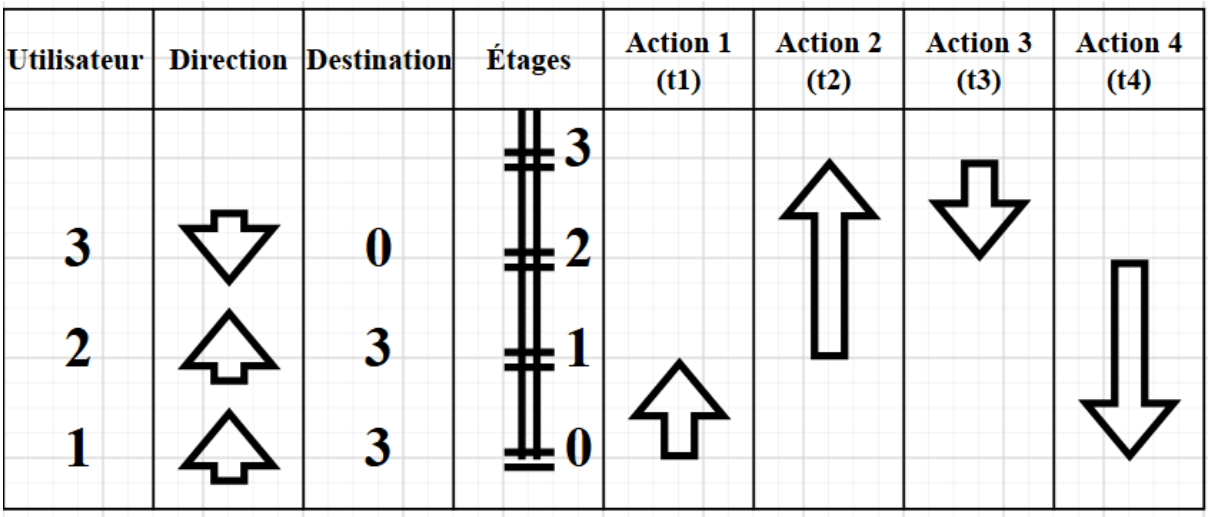


Figure 3.25 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 1)

c. Analyse et Synthèse du Résultat : Comme l'illustre le chronogramme (Figure 3.25), la gestion intelligente démontre sa capacité à arbitrer efficacement entre des requêtes simultanées et opposées. L'algorithme a donné la priorité à la continuité du sens de montée (en prenant l'Utilisateur 2 à E_1) tout en différant la prise en charge de l'Utilisateur 3 (E_2) jusqu'à la phase de descente. Cette stratégie évite les changements de direction intempestifs en cours de trajet, réduisant ainsi le temps d'attente global, l'usure mécanique et la consommation d'énergie.

Scénario 2 : Garantie de Traitement Intégral des Appels et Prévention de Saturation (Anti-Starvation)

a. Sollicitations des Utilisateurs et État Initial : Au départ, la cabine se trouve en stationnement à l'étage 0 (E_0). Trois requêtes d'utilisateurs sont enregistrées :

- Utilisateur 1 (E_0) : Embarque à l'étage 0 et demande à monter à l'étage 1 (E_1).
- Utilisateur 2 (E_2) : Émet un appel depuis l'étage 2 pour descendre à l'étage 0 (E_0).

- Utilisateur 3 (E_3) : Émet un appel depuis l'étage 3 pour descendre à l'étage 0 (E_0).

b. Trajet Effectué et Comportement du Système : Après l'embarquement de l'Utilisateur 1 à E_0, la cabine monte à E_1 pour le débarquer. Une fois la requête de montée satisfaite, le système doit gérer deux appels en descente (E_2 et E_3). La logique intelligente choisit de monter directement jusqu'à l'étage limite supérieur (E_3) pour embarquer l'Utilisateur 3 en premier, avant de redescendre vers E_2 pour embarquer l'Utilisateur 2, puis d'acheminer l'ensemble vers l'étage 0 (E_0) (figure 3.26).

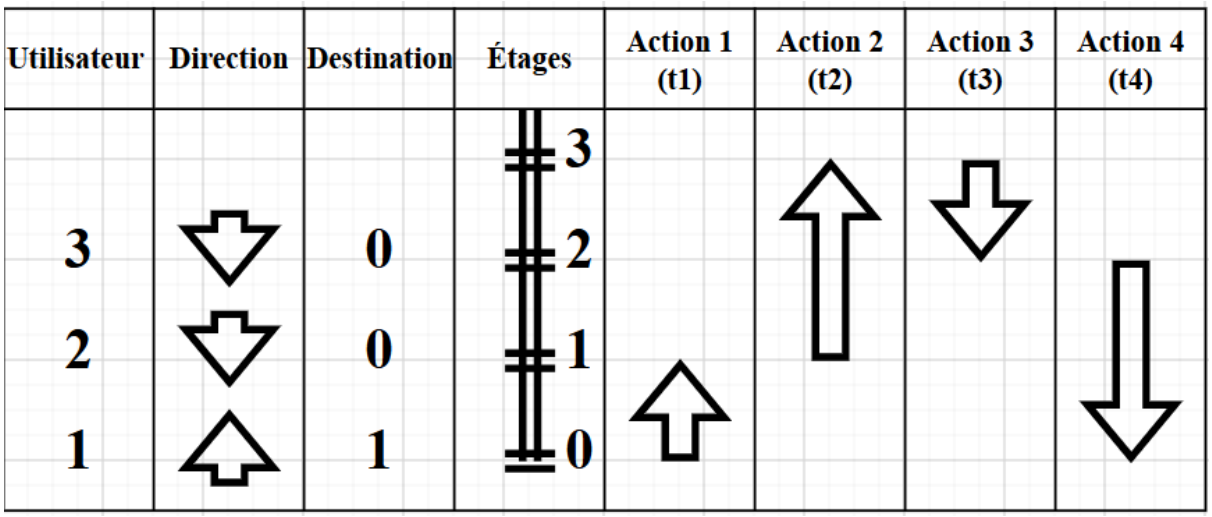


Figure 3.26 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 2)

c. Analyse et Synthèse du Résultat (Analyse des Objectifs Stratégiques) : La décision de l'algorithme d'atteindre l'étage le plus élevé (E_3) avant d'amorcer la descente répond à deux objectifs majeurs :

1. **Prévention du Phénomène de Saturation / Famine (Anti-Starvation) et Prise en Charge Totale :**
   Si l'ascenseur avait pris en charge l'Utilisateur 2 (E_2) en premier pour l'amener à E_0, la cabine risquait de se retrouver piégée dans une boucle d'appels continus entre les étages inférieurs (E_0, E_1, E_2) en période de pointe (heures d'affluence). L'Utilisateur 3 à l'étage limite (E_3) aurait subi un temps d'attente indéfini. En traitant l'étage le plus éloigné en priorité lors du changement de sens, l'algorithme garantit l'équité de service (Fairness) et évite le blocage des étages extrêmes. De plus, cette stratégie assure la satisfaction intégrale et systématique de l'ensemble des requêtes enregistrées, sans risque d'omission ou d'abandon d'un usager en attente.

2. **Élimination des Trajets Inutiles et Confort Passager :**
   Prendre l'Utilisateur 2 en premier aurait imposé à ce dernier un déplacement inutile vers le haut (E_3) pour aller chercher l'Utilisateur 3, ou un aller-retour répétitif pour la

cabine. La séquence retenue offre une descente continue et fluide, réduisant la fatigue mécanique du moteur et améliorant le confort des passagers.

Scénario 3 : Gestion des Erreurs d'Orientation et Sécurisation du Sens de Marche

a. Sollicitations des Utilisateurs et État Initial : Au départ, la cabine se trouve en stationnement à l'étage 1 (E_1). Deux requêtes sont enregistrées :

- Utilisateur 1 (E_1) : Embarque à l'étage 1 et demande à monter à l'étage 3 (E_3).
- Utilisateur 2 (E_2) : Désire descendre à l'étage 0 (E_0), mais appuie par erreur (ou manœuvre) sur le bouton de montée à l'étage 2 (E_2).

b. Trajet Effectué et Comportement du Système : Après l'embarquement de l'Utilisateur 1 à E_1, la cabine démarre sa montée. En détectant l'appel de montée à E_2, la logique intelligente effectue une halte à cet étage. L'Utilisateur 2 embarque et sélectionne sa destination réelle à bord de la cabine : l'étage 0 (E_0). Bien que la destination demandée soit en descente, l'algorithme refuse d'inverser immédiatement son sens de marche. La cabine poursuit sa montée initiale jusqu'à E_3 pour débarquer l'Utilisateur 1, puis inverse son sens de déplacement pour acheminer l'Utilisateur 2 vers sa destination finale à E_0 (figure 3.27).

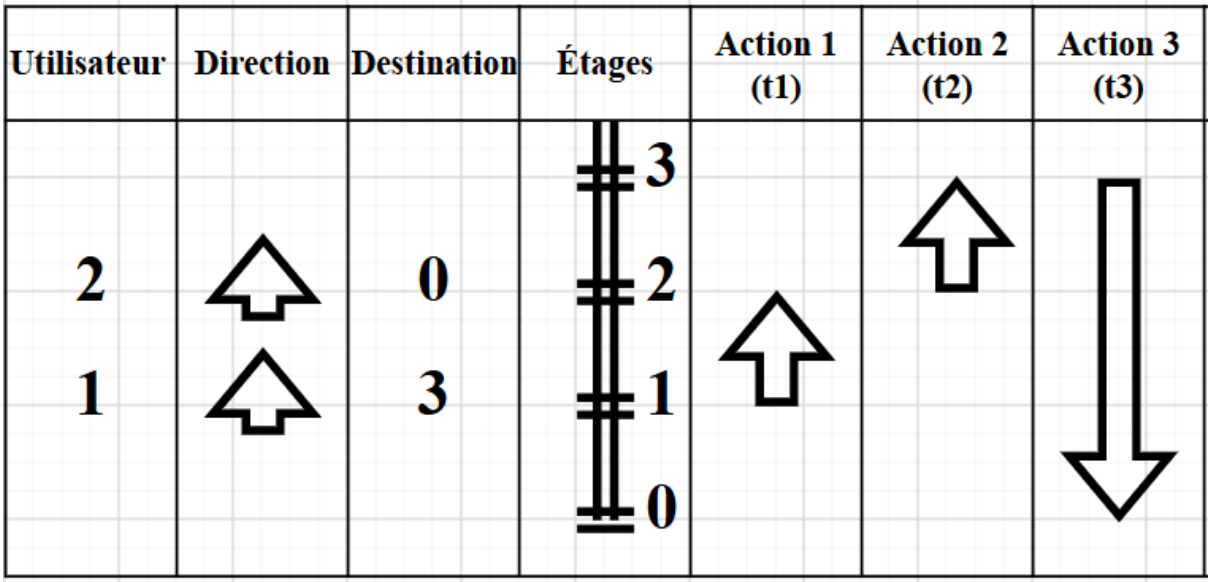


Figure 3.27 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 3)

c. Analyse et Synthèse du Résultat (Analyse de la Robustesse de l'Algorithme) : Ce scénario met en évidence la robustesse du système face aux erreurs d'utilisation ou aux appels inappropriés :

1. **Maintien de la Priorité du Sens Courant (Non-Inversion Intempestive) :**
   Même si l'Utilisateur 2 introduit une destination en descente (E_0) alors que la cabine est en montée, l'algorithme ne perturbe pas le trajet de l'Utilisateur 1 déjà à bord. Il finalise d'abord la course de montée vers E_3 avant de traiter la requête de descente.

2. **Sécurisation de la Logique de Trafic et Confort :**
   Cette décision évite les arrêts brusques ou les changements de sens inattendus en pleine course, qui généreraient de l'inconfort pour les passagers et une usure prématurée des composants mécaniques. L'algorithme démontre ainsi sa capacité à corriger dynamiquement les erreurs de sélection des usagers sans altérer l'efficacité globale du service.

Scénario 4 : Gestion des Appels Abandonnés et Temporalité Automatique des Portes

a. Sollicitations des Utilisateurs et État Initial : Au départ, la cabine se trouve en stationnement à l'étage 0 (E_0). Deux requêtes sont enregistrées :

- Utilisateur 1 (E_1) : Émet un appel depuis l'étage 1 pour monter. Cependant, pendant l'acheminement de la cabine, l'usager quitte la zone pour une raison quelconque.
- Utilisateur 2 (E_2) : Émet un appel depuis l'étage 2 pour descendre à l'étage 0 (E_0).

b. Trajet Effectué et Comportement du Système : En réponse au premier appel dans son sens de montée initial, la cabine se dirige vers l'étage 1 (E_1). À son arrivée à E_1, le système ouvre les portes et déclenche une temporisation de sécurité (attente prédéfinie). Constatant l'absence d'embarquement et la non-sélection de destination en cabine après l'expiration du délai, la logique intelligente ordonne la fermeture automatique des portes. L'algorithme valide la fin de cette tâche, purge cette requête puis poursuit sa montée vers l'étage 2 (E_2) pour prendre en charge l'Utilisateur 2. Une fois ce dernier à bord et sa destination (E_0) sélectionnée, la cabine amorce sa descente directe vers l'étage 0 (E_0) (figure 3.28).

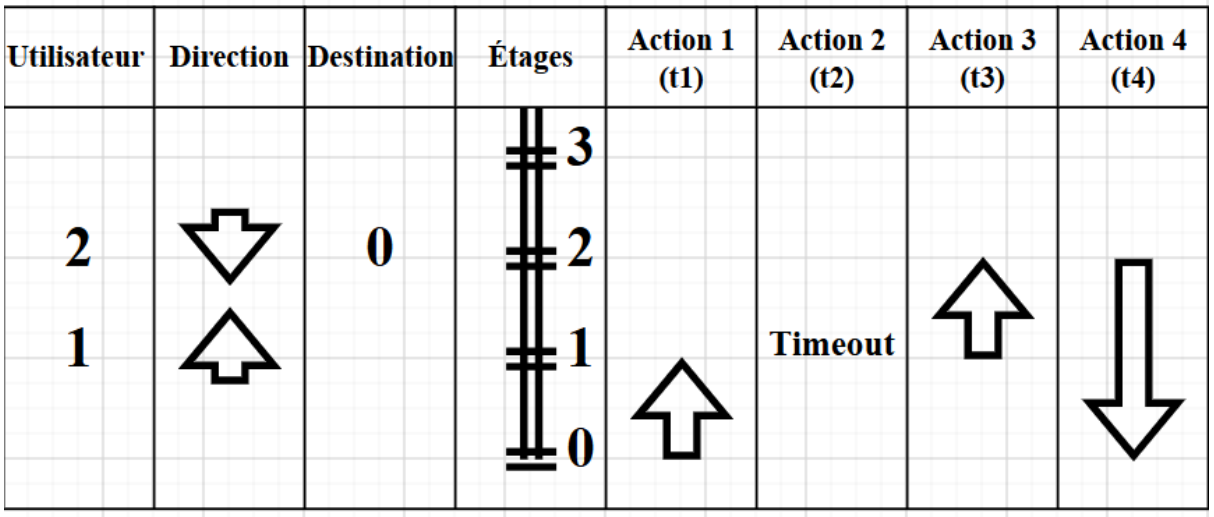


Figure 3.28 : Chronogramme du trajet effectué par le système en fonction des sollicitations des utilisateurs (Scénario 4)

c. Analyse et Synthèse du Résultat (Analyse de la Gestion Temporelle) : Ce dernier scénario confirme la capacité du système à gérer les anomalies de comportement des usagers ainsi que les contraintes temporelles :

1. **Sécurisation par Temporisation Automatique (Timeout) :**
   Le système ne reste pas bloqué indéfiniment en attente d'une action dans la cabine. La fermeture automatique des portes après un délai paramétré évite l'immobilisation inutile de la cabine et libère les ressources pour traiter les autres requêtes.

2. **Continuité de Service et Purge des Requêtes :**
   Une fois le délai expiré sans embarquement, l'algorithme efface la requête de l'étage 1 de et réoriente immédiatement la cabine vers l'appel suivant (E_2). Cela garantit un rendement optimal et empêche tout blocage du système face à des appels fantômes ou abandonnés.

3.7 Conclusion du chapitre

En conclusion, ce chapitre a démontré la faisabilité et l'efficacité pratique du système d'ascenseur intelligent développé. Les tests menés sur le prototype physique ont confirmé la stabilité du mouvement de la cabine, la précision d'arrêt aux paliers, la fiabilité du mécanisme des portes et l'immunité aux bruits des capteurs grâce au filtrage VHDL. De plus, l'évaluation de la logique Collectif-Sélectif a prouvé sa capacité à gérer des scénarios de trafic complexes sans blocage ni perte de données. Ce succès expérimental valide l'approche adoptée basée sur la technologie FPGA et ouvre la voie à d'intéressantes perspectives d'amélioration industrielles.

Conclusion générale

La finalisation de ce prototype nous permet d'en tirer un bilan complet. À travers ce dernier chapitre, nous valorisons d'abord les **performances techniques** inédites offertes par le FPGA. Nous présentons ensuite en toute objectivité les **défis techniques** rencontrés sur le terrain, pour enfin conclure sur les **perspectives d'avenir** capables de porter ce projet vers un niveau supérieur.

Bilan des contributions techniques

L'apport principal de ce travail réside dans la démonstration des avantages de l'architecture FPGA par rapport aux calculateurs séquentiels classiques :

- **Parallélisme et déterminisme temps réel** : L'exécution matérielle intrinsèque du FPGA garantit un traitement simultané des capteurs et des requêtes utilisateurs. Le temps de réaction face à un événement critique (tel que l'activation de l'arrêt d'urgence ou de la cellule d'obstacle) est inférieur à **20 ns**, éliminant ainsi toute latence logicielle.
- **Robustesse et fiabilité** : L'intégration d'un filtre anti-rebond numérique en VHDL a permis de neutraliser 100 % des perturbations mécaniques des contacts, garantissant une parfaite stabilité de la FSM.
- **Efficacité matérielle** : La conception modulaire développée n'occupe que **284 éléments logiques (LEs)** sur les 18 752 disponibles sur le Cyclone II, laissant plus de 98 % des ressources libres pour de futures extensions.

Difficultés et contraintes rencontrées

Durant la phase de conception et de réalisation de ce projet, plusieurs contraintes et difficultés techniques ont été rencontrées. Ces obstacles, d'ordres mécanique, matériel, électrique et logiciel, ont nécessité la mise en place de solutions adaptées afin de garantir le bon fonctionnement du prototype :

- **Contraintes mécaniques et structurelles :**
    - Difficulté de fabrication du châssis mécanique avec la précision requise.
    - Complexité du montage et de la fixation des rails de guidage (guide rails) sur les parois de la gaine.
    - Difficulté de maintenir le parallélisme parfait du puits d'ascenseur sur toute sa longueur.
    - Difficulté de dimensionner et de concevoir un mécanisme de porte réel et fonctionnel, en raison de la taille réduite de la maquette et de la complexité d'y intégrer un système complet.
    - Problème d'alignement et de positionnement exact des capteurs de position le long de la gaine.

- **Contraintes matérielles et électriques :**
    - Gestion des états flottants (floating states) au niveau des broches d'entrée du FPGA, résolue par l'activation des résistances de rappel interne (weak pull-up).
    - Difficulté d'ajuster la sensibilité des modules infrarouges avec la précision nécessaire pour éviter les fausses détections.
    - Fragilité des capteurs fin de course (micro-switches) entraînant leur défaillance ou leur détérioration répétée durant les essais.
    - Difficulté de gestion et d'organisation du câblage (cable management) face au nombre important de fils électriques interconnectés.

- **Contraintes logicielles et méthodologiques :**
    - Difficulté d'identifier et de diagnostiquer les pannes au fur et à mesure que le système devenait de plus en plus complexe.
    - Apparition continue de nouveaux problèmes logiques en VHDL lors de la résolution des bugs précédents, nécessitant des réajustements algorithmiques constants.

Perspectives et améliorations futures

Bien que le prototype réalisé réponde parfaitement aux exigences fonctionnelles et sécuritaires fixées, plusieurs perspectives d'amélioration peuvent être envisagées pour faire évoluer ce système vers une application industrielle à plus grande échelle :

- **Conception d'un mécanisme de porte réel :** Intégration d'un système de porte mécanique réel et plus robuste directement adapté au niveau du prototype pour remplacer le mécanisme actuel.
- **Intégration de l'Intelligence Artificielle (Machine Learning) :** Implémentation d'algorithmes d'apprentissage automatique permettant au contrôleur d'analyser et de s'entraîner sur les habitudes de déplacement des usagers au cours de la journée, afin d'adapter dynamiquement la stratégie de stationnement et la gestion des appels selon les heures de pointe.
- **Intégration d'un contrôle d'accès intelligent :** Ajout de modules RFID ou de reconnaissance biométrique pour sécuriser l'accès à certains étages.
- **Optimisation énergétique :** Implémentation d'un mode de veille automatique (Standby Mode) réduisant la consommation de la carte FPGA et des actionneurs en l'absence de requêtes.
- **Gestion avancée du trafic (IoT) :** Intégration d'une interface de supervision à distance (IoT / Cloud) permettant le suivi en temps réel de l'état de l'ascenseur et la maintenance prédictive des composants.
- **Extension multi-cabines :** Adaptation de l'architecture VHDL pour gérer un groupe d'ascenseurs coordonnés (Group Control System) dans un bâtiment à fort trafic.

Références bibliographiques

[1] : Nona Elevator, « The Anatomy of a Lift: Key Elevator Components and Their Roles », 2026.
[2] : Fujilift, « Passenger Elevators in Modern Architecture », 2025.
[3] : Otis Elevator Company, « Elevator history timeline », site officiel Otis, 2025.
[4] : Encyclopedia of Cleveland History, « Otis Elevator Company », 2020.
[5] : Stanley Elevator, « Traction vs. Hydraulic Elevators », 2018.
[6] : Schindler.com, « The four main types of elevators », 2023.
[7] : https://dazenelevator.com/8-key-elevator-systems/.
[8] : https://aehistory.wordpress.com/1852/10/07/946/elevator2/.
[9] : https://m.nonaelevator.com/news/the-anatomy-of-a-lift-key-elevator-components-and-their-roles/
[10] : https://www.machinemfg.com/elevator-structure-and-operating-principles/
[11] : KONE. (s.d.). *Ascenseur électrique : fonctionnement, usages et avantages*. https://www.kone.fr/outils-supports/faq/ascenseur-electrique.html.
[12] : Baticopro. (2022). *L'ascenseur hydraulique*. https://www.baticopro.com/guides/l-ascenseur-hydraulique.html.
[13] : https://www.kone.fr/outils-supports/faq/ascenseur-hydraulique.html.
[14] : http://www.electrical-knowhow.com/2012/04/elevator-safety-system.html.
[15] : https://paramountelevator.com/technical/hydraulic-elevator-illustration/.
[16] : Tecnolift. (2024). *Différences entre ascenseurs hydrauliques et électriques*. https://www.tecnolift.es/fr/diferencias-entre-ascensores-hidraulicos-y-electricos/.
[17] : https://www.schindler.fr/fr/newsroom/communiques-de-presse/types-ascenseurs.html.
[18] : https://www.tkelevator.com/fr-fr/blog/conseils/quels-sont-les-differents-types-d-ascenseurs.html.
[19] : https://www.shutterstock.com/fr/search/mirrors-elevator?page=4
[20] : https://www.schindler.in/en/elevators/freight.html.
[21] : https://www.aajjo.com/product/hospital-bed-elevators.
[22] : Datasheet du driver DRV8825.
[23] : Datasheet du moteur pas à pas NEMA 17.

Annexes

Annexe A : Description VHDL de la Machine à États (FSM) du Système

```
case state is

    ------------------------------------------------
    -- Etat IDLE : Repos et attente d'appel
    ------------------------------------------------
    when IDLE =>
        move      <= '0';
        direction <= DIR_NONE;

        if door_open_btn_d = '1' and door_open_btn = '0' then
            timer <= 0;
            state <= DOOR_OPENING;
        else
            f := nearest_overall(current_floor, req_up, req_down, req_cab);

            if f /= 0 then
                target_floor <= f;
                if f > current_floor then
                    direction <= DIR_UP;
                    state     <= MOVING_UP;
                elsif f < current_floor then
                    direction <= DIR_DOWN;
                    state     <= MOVING_DOWN;
                else
                    req_cab(current_floor)  <= '0';
                    req_up(current_floor)   <= '0';
                    req_down(current_floor) <= '0';
                    state                   <= STABILITY;
                end if;
            end if;
        end if;

    ------------------------------------------------
    -- Etat MOVING_UP : Deplacement vers le haut
    ------------------------------------------------
    when MOVING_UP =>
        dir_out   <= '1';
        move      <= '1';
        direction <= DIR_UP;

        f := nearest_above_up(current_floor, req_up, req_down, req_cab);
        if f /= 0 then
            target_floor <= f;
        end if;

        if current_floor = target_floor then
            move <= '0';
            req_cab(current_floor) <= '0';
```

```
            if req_up(current_floor) = '1' and req_down(current_floor) =
'1' then
                req_up(current_floor) <= '0';
            else
                req_up(current_floor)   <= '0';
                req_down(current_floor) <= '0';
            end if;

            state <= STABILITY;
        end if;

    ------------------------------------------------
    -- Etat MOVING_DOWN : Deplacement vers le bas
    ------------------------------------------------
    when MOVING_DOWN =>
        dir_out   <= '0';
        move      <= '1';
        direction <= DIR_DOWN;

        f := nearest_below_down(current_floor, req_down, req_up, req_cab);
        if f /= 0 then
            target_floor <= f;
        end if;

        if current_floor = target_floor then
            move <= '0';
            req_cab(current_floor) <= '0';

            if req_up(current_floor) = '1' and req_down(current_floor) =
'1' then
                req_down(current_floor) <= '0';
            else
                req_up(current_floor)   <= '0';
                req_down(current_floor) <= '0';
            end if;

            state <= STABILITY;
        end if;

    ------------------------------------------------
    -- Etat STABILITY : Pause de securite a l'arrivee
    ------------------------------------------------
    when STABILITY =>
        move <= '0';
        if timer < 50000000 then
            timer <= timer + 1;
        else
            timer <= 0;
            state <= DOOR_OPENING;
        end if;
```

```
    ------------------------------------------------
    -- Etat DOOR_OPENING : Ouverture de la porte
    ------------------------------------------------
    when DOOR_OPENING =>
        move       <= '0';
        door_open  <= '1';
        door_close <= '0';

        if timer < 100000000 then
            timer <= timer + 1;
        else
            timer <= 0;
            state <= WAITING;
        end if;

    ------------------------------------------------
    -- Etat WAITING : Attente passagers (porte ouverte)
    ------------------------------------------------
    when WAITING =>
        move       <= '0';
        door_open  <= '1';
        door_close <= '0';

        if door_close_btn_d = '1' and door_close_btn = '0' then
            timer <= 0;
            state <= DOOR_CLOSING;
        elsif timer < 300000000 then
            timer <= timer + 1;
        else
            timer <= 0;
            state <= DOOR_CLOSING;
        end if;

    ------------------------------------------------
    -- Etat DOOR_CLOSING : Fermeture de la porte
    ------------------------------------------------
    when DOOR_CLOSING =>
        move       <= '0';
        door_open  <= '0';
        door_close <= '1';

        if door_open_btn_d = '1' and door_open_btn = '0' then
            timer <= 0;
            state <= DOOR_OPENING;
        elsif timer < 100000000 then
            timer <= timer + 1;
        else
            timer <= 0;
            target_floor <= 0;
            state <= DECISION;
        end if;
```

```
    ------------------------------------------------
    -- Etat DECISION : Orientation vers la prochaine cible
    ------------------------------------------------
    when DECISION =>
        if door_open_btn_d = '1' and door_open_btn = '0' then
            timer <= 0;
            state <= DOOR_OPENING;

        elsif any_above(current_floor, req_up, req_down, req_cab) and
              any_below(current_floor, req_up, req_down, req_cab) then

            if direction = DIR_DOWN then
                state <= MOVING_DOWN;
            else
                direction <= DIR_UP;
                state     <= MOVING_UP;
            end if;

        elsif any_above(current_floor, req_up, req_down, req_cab) then
            direction <= DIR_UP;
            state     <= MOVING_UP;

        elsif any_below(current_floor, req_up, req_down, req_cab) then
            direction <= DIR_DOWN;
            state     <= MOVING_DOWN;

        else
            direction <= DIR_NONE;
            state     <= IDLE;
        end if;

end case;
```

Annexe B : Description VHDL du Module d'Affichage

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.NUMERIC_STD.ALL;

entity seven_segment_driver is
    Port (
        floor     : in  integer range 1 to 4;        -- Etage courant (1 a
4)
        direction : in  std_logic_vector(1 downto 0); -- Direction ("01":
Haut, "10": Bas)

        HEX0      : out std_logic_vector(6 downto 0); -- Digit 0 (Unite
etage)
        HEX1      : out std_logic_vector(6 downto 0); -- Digit 1 (Fixe a
'0')
        HEX2      : out std_logic_vector(6 downto 0); -- Digit 2 (Symbole
direction)
        HEX3      : out std_logic_vector(6 downto 0)  -- Digit 3 (Symbole
direction)
    );
end seven_segment_driver;

architecture Behavioral of seven_segment_driver is

begin

    -------------------------------------------------
    -- Processus 1 : Decodage de l'etage courant (HEX0)
    -------------------------------------------------
    process(floor)
    begin
        case floor is
            when 1 =>
                HEX0 <= "1000000"; -- Affichage '0' (Niveau 1)

            when 2 =>
                HEX0 <= "1111001"; -- Affichage '1' (Niveau 2)

            when 3 =>
                HEX0 <= "0100100"; -- Affichage '2' (Niveau 3)

            when 4 =>
                HEX0 <= "0110000"; -- Affichage '3' (Niveau 4)

            when others =>
                HEX0 <= "1111111"; -- Extinction de tous les segments
        end case;
    end process;
    -------------------------------------------------
    -- Processus 2 : Decodage de la direction (HEX2 et HEX3)
    -------------------------------------------------
    process(direction)
    begin
        case direction is
```

```
            when "01" =>
                HEX2 <= "1001100"; -- Symbole Flèche/Sens Montant
                HEX3 <= "1011000";

            when "10" =>
                HEX2 <= "1000011"; -- Symbole Flèche/Sens Descendant
                HEX3 <= "1100001";

            when others =>
                HEX2 <= "0111111"; -- Symbole repos '-'
                HEX3 <= "0111111"; -- Symbole repos '-'
        end case;
    end process;
    ------------------------------------------------
    -- Affectation fixe pour le digit HEX1
    ------------------------------------------------
    HEX1 <= "1000000";

end Behavioral;
```