Projet Tactera

Chaque cube contient une idée, chaque forme une sensation

Delaforge Maxyme
Ingold Martin
Caron Corentin
1ère Générale
Lycée Galilée FRANQUEVILLE-ST-PIERRE

13 mars 2026

Résumé

En résumé, le projet TACTERA est un dispositif mécanique modulaire interactif qui illustre concrètement comment l'ingénierie peut enrichir l'expérience artistique et la rendre plus accessible. Grâce à un système de reliefs physiques mobiles, notre solution permet de matérialiser des formes, motifs et animations de manière tangible, perceptible par la vue et par le toucher. Ce projet répond à un enjeu d'inclusion majeur, notamment pour les plus de 2 millions de Français atteints de déficiences visuelles, en proposant une interaction intuitive ne nécessitant aucun apprentissage complexe. À travers TACTERA, nous démontrons que la technologie, lorsqu'elle est pensée au service de l'humain, peut rapprocher innovation, accessibilité et création artistique, et ainsi contribuer à construire la vie de demain plus inclusive et plus interactive.

Avant-propos

Ce projet a été réalisé dans le cadre du Concours C-Génial (2025-2026) et des Olympiades de Sciences de l'Ingénieur (2025-2026-2027). Ces concours ont pour objectif de valoriser la créativité, la rigueur scientifique et la démarche d'ingénierie des élèves à travers la conception d'un projet innovant.

Tactera est un dispositif mécanique modulaire qui matérialise des formes, motifs et animations par des reliefs physiques mobiles. Il rend l'art et l'information accessibles de manière intuitive, sans apprentissage complexe, en s'appuyant uniquement sur la vue et/ou la perception tactile du relief.

Ce projet est le fruit d'un travail d'équipe, mettant en œuvre des phases de réflexion, de simulation, d'expérimentation, de modélisation et de prototypage. À travers celui-ci, nous avons souhaité proposer une approche technologique originale, alliant électronique, mécanique et programmation, afin de transformer une information numérique en une expérience physique tangible.

De nombreuses compétences non scientifiques ont également été mobilisées, telles que la communication (réseaux sociaux), le montage vidéo, les demandes de partenariats ainsi que les capacités d'organisation, rendant ce projet d'autant plus enrichissant et formateur.

Enfin, ce projet s'inscrit dans une démarche écoresponsable : 80 % des composants utilisés proviennent de matériaux de récupération, et le plastique employé est issu d'une filière de recyclage.

Remerciements

Nous tenons tout d'abord à remercier nos professeurs de Sciences de l'Ingénieur, Gonzalez Frédérique et Frouin George, sans qui ce projet n'aurait jamais vu le jour. Nous les remercions pour leurs conseils, leur accompagnement et leur soutien tout au long de ce travail.

Nous souhaitons également remercier le concours C-Génial ainsi que les Olympiades des Sciences de l'Ingénieur, qui nous ont donné un objectif et une motivation tout au long de ce projet.

Nous adressons également nos sincères remerciements à nos partenaires et sponsors :

— La boutique informatique de Rouen, qui nous a fourni d'anciens appareils dont nous avons pu récupérer de nombreux composants.

— Matlab, qui met à disposition un logiciel précieux pour la réalisation de nos premières simulations mathématiques.

— PCBWay, qui nous a offert la production de nos cartes de développement.

— Le Musée des Beaux-Arts de Rouen ainsi que la Métropole Rouen Normandie, qui nous ont ouvert leurs portes pour le tournage de notre vidéo de présentation.

Nous souhaitons également remercier toutes les personnes ayant participé comme figurants dans notre vidéo de présentation.

Enfin, nous remercions notre établissement, le lycée Galilée de Franqueville-Saint-Pierre, pour l'espace de travail mis à notre disposition ainsi que pour son soutien financier.

Table des matières

Première partie

Introduction

Chapitre 1

Introduction

1.1 Contexte du projet

L'accès à l'information, à l'art et aux contenus pédagogiques repose aujourd'hui majoritairement sur des supports numériques visuels. Cette évolution technologique offre de nombreuses possibilités mais crée également de nouvelles formes d'exclusion pour certains publics.

Selon les données de l'INSEE (2023), environ 2 millions de personnes en France sont atteintes de déficiences visuelles, dont près de 207 000 personnes aveugles. Pour ces utilisateurs, l'accès aux contenus visuels reste fortement limité, malgré les progrès des technologies d'assistance.

Par ailleurs, les dispositifs actuels reposent très largement sur des interfaces planes (écrans, images, vidéos), qui ne permettent pas toujours une exploration sensorielle complète des informations.

1.2 Problématique

Cette situation soulève plusieurs difficultés importantes :

— une accessibilité limitée aux contenus visuels et artistiques pour les personnes malvoyantes ou aveugles ;

— un manque d'outils pédagogiques interactifs permettant une exploration tactile des informations ;

— une dépendance forte aux interfaces numériques planes, réduisant les possibilités d'interaction physique avec les contenus.

Il devient ainsi pertinent d'explorer de nouvelles formes d'interface permettant de représenter des informations de manière physique et interactive.

Ainsi, nous explorerons dans ce projet, comment transformer une image en une surface perceptible par le toucher.

1.3 Objectifs du projet

Le projet *Tactera* vise à développer un dispositif capable de matérialiser des formes et des informations sous forme de reliefs physiques dynamiques.

L'objectif est de concevoir une surface interactive composée d'éléments mécaniques modulaires capables de générer des reliefs contrôlés.

Ce système permettrait :

— de rendre certains contenus visuels accessibles par le toucher ;

— de proposer de nouvelles formes d'exploration artistique et pédagogique ;

— d'offrir une interface physique intuitive ne nécessitant pas d'apprentissage complexe.

Le dispositif pourrait ainsi être utilisé dans des contextes variés, tels que la médiation culturelle, l'éducation scientifique ou les démonstrations publiques.

Deuxième partie

Cahier des charges

Chapitre 2

Fonctions principales

2.1 Objectifs généraux

Objectif	Description
Rendre l'art accessible	Proposer des créations artistiques perceptibles par relief mécanique, adaptées aux malvoyants et au grand public.
Allier technologie et culture	Montrer comment la mécanique, l'électronique et la programmation peuvent enrichir une expérience artistique inclusive.
Favoriser l'inclusion	Concevoir un système simple, transportable et lisible par relief pour tous les publics.
Soutenir la pédagogie	Offrir un support concret pour illustrer des concepts techniques et scientifiques.

2.2 Spécifications techniques

2.2.1 Fonctions principales

Le système doit assurer plusieurs fonctions essentielles afin de remplir son objectif de génération de reliefs physiques interactifs. Ces fonctions constituent le cœur du cahier des charges fonctionnel.

— **Génération de reliefs dynamiques** : produire des variations de hauteur à l'aide d'une matrice d'éléments mécaniques mobiles afin de représenter des formes ou des informations spatiales.

— **Lecture tactile et visuelle** : permettre à l'utilisateur d'observer et de percevoir physiquement les reliefs générés, afin de faciliter l'interaction avec le dispositif.

— **Interface de pilotage intuitive** : offrir un moyen simple de contrôler le système (sélection de presets, commandes logicielles ou interface utilisateur dédiée).

— **Sécurité et robustesse** : garantir un fonctionnement fiable du dispositif tout en assurant la sécurité des utilisateurs lors de la manipulation ou des démonstrations.

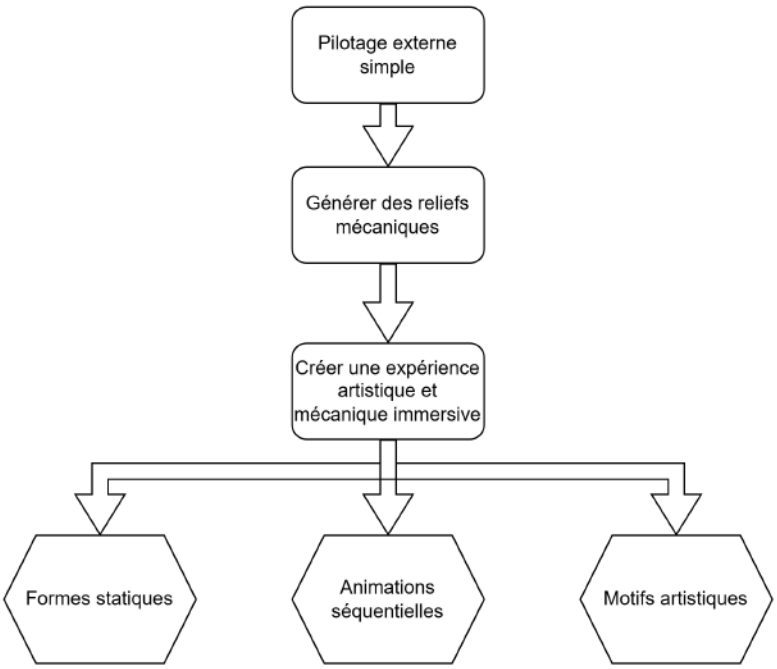


FIGURE 2.1 – Diagramme fonctionnel représentant les principales fonctions du système Produit Minimum Viable

Chapitre 3

Exigences techniques & Contraintes

3.1 Exigences techniques

Les exigences techniques traduisent les fonctions du système en spécifications mesurables permettant de vérifier que le dispositif répond correctement au cahier des charges.

Fonction	**Spécifications techniques**	**Critères de validation**
Génération des reliefs	Matrice modulaire d'éléments mécaniques mobiles permettant de produire des variations de hauteur contrôlées.	Mouvements précis et stables ; maintien de la position sans consommation permanente excessive.
Pilotage	Contrôle du système par microcontrôleur (ex. ESP32) avec architecture sujet-maitre pour gérer un grand nombre d'actionneurs.	Fonctionnement continu pendant au moins 30 minutes sans bug, blocage ou redémarrage du système.
Alimentation	Alimentation sur secteur 220 V avec conversion vers les tensions nécessaires aux composants électroniques et moteurs.	Sécurité électrique validée ; absence de surchauffe ou de comportement instable.
Transportabilité	Structure mécanique permettant le déplacement du dispositif pour des démonstrations ou des présentations.	Masse totale inférieure à 15 kg ; transport possible par deux personnes.
Sécurité	Protection des parties mobiles et des composants électriques afin d'éviter tout risque pour l'utilisateur.	Aucun composant dangereux accessible ; tests en conditions réelles sans incident.
Robustesse	Conception mécanique et électronique capable de supporter un usage répété lors de démonstrations.	Aucun dysfonctionnement majeur après au moins 50 cycles d'utilisation.

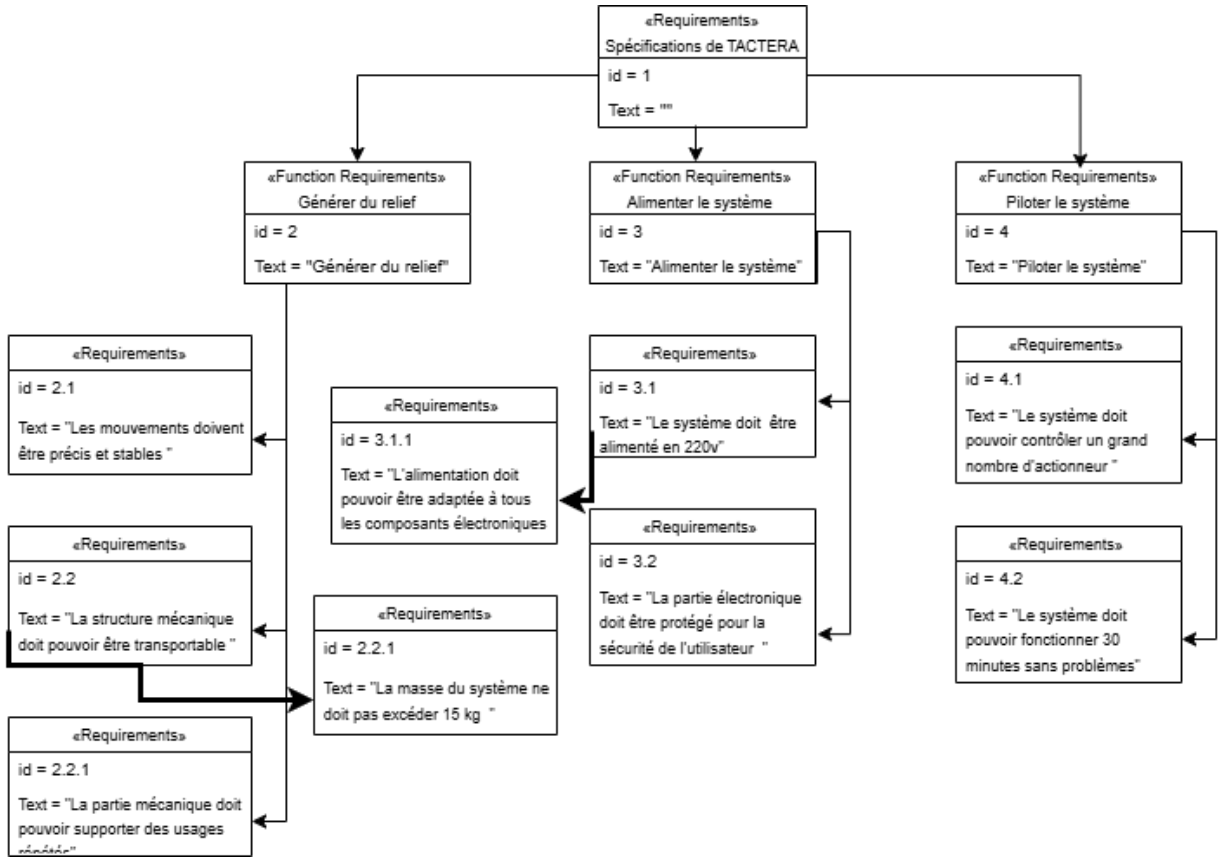


FIGURE 3.1 – Diagramme des exigences techniques

3.2 Contraintes à respecter

Le développement du système doit également respecter plusieurs contraintes liées au contexte de réalisation du projet.

— **Contraintes matérielles** : utilisation prioritaire de composants disponibles au lycée, à faible coût ou issus de récupération.

— **Contraintes d'utilisation** : l'interface de pilotage doit rester simple et intuitive, avec des commandes prédéfinies (presets logiciels) ne nécessitant pas de manipulation complexe.

— **Contraintes de conception** : le système doit être modulaire afin de faciliter la maintenance, la réparation et les évolutions futures du dispositif.

— **Contraintes esthétiques** : l'apparence du prototype doit rester soignée et adaptée à des démonstrations publiques (concours, présentations, expositions).

3.3 Risques et solutions de contournement

Risques	**Impacts**	**Solutions**
Perte d'alimentation	Arrêt du système	Indicateur de branchement
Bug logiciel	Gel du prototype	Version stable, watchdog, tests unitaires
Surchauffe	Endommagement matériel	Ventilation, capteurs thermiques
Fragilité mécanique	Casse au transport	Support renforcé, pièces modulaires
Dépassement de budget	Achat impossible	Alternatives low-cost, priorisation
Non-conformité sécurité	Refus de validation	Respect des normes, isolation

Chapitre 4

Budget & Planning

4.1 Planning prévisionnel

La réalisation du projet s'inscrit dans une démarche de gestion de projet structurée. Le développement du système est ainsi réparti en plusieurs phases successives permettant de passer progressivement de la conception théorique à la réalisation du prototype fonctionnel.

Les principales étapes du projet sont les suivantes :

— **Conception mécanique et électronique** : étude de l'architecture du système, modélisation des pièces mécaniques et conception des circuits électroniques.

— **Prototypage et assemblage** : fabrication des premiers modules, impression 3D des éléments mécaniques et réalisation des circuits imprimés.

— **Programmation et tests** : développement des logiciels de contrôle, mise en place des algorithmes et validation du fonctionnement des différents modules.

— **Corrections et optimisation** : amélioration des performances du système, correction des problèmes identifiés lors des phases de test et optimisation de la fiabilité.

— **Préparation de la soutenance** : rédaction du rapport scientifique, préparation de la démonstration du prototype et élaboration du support de présentation.

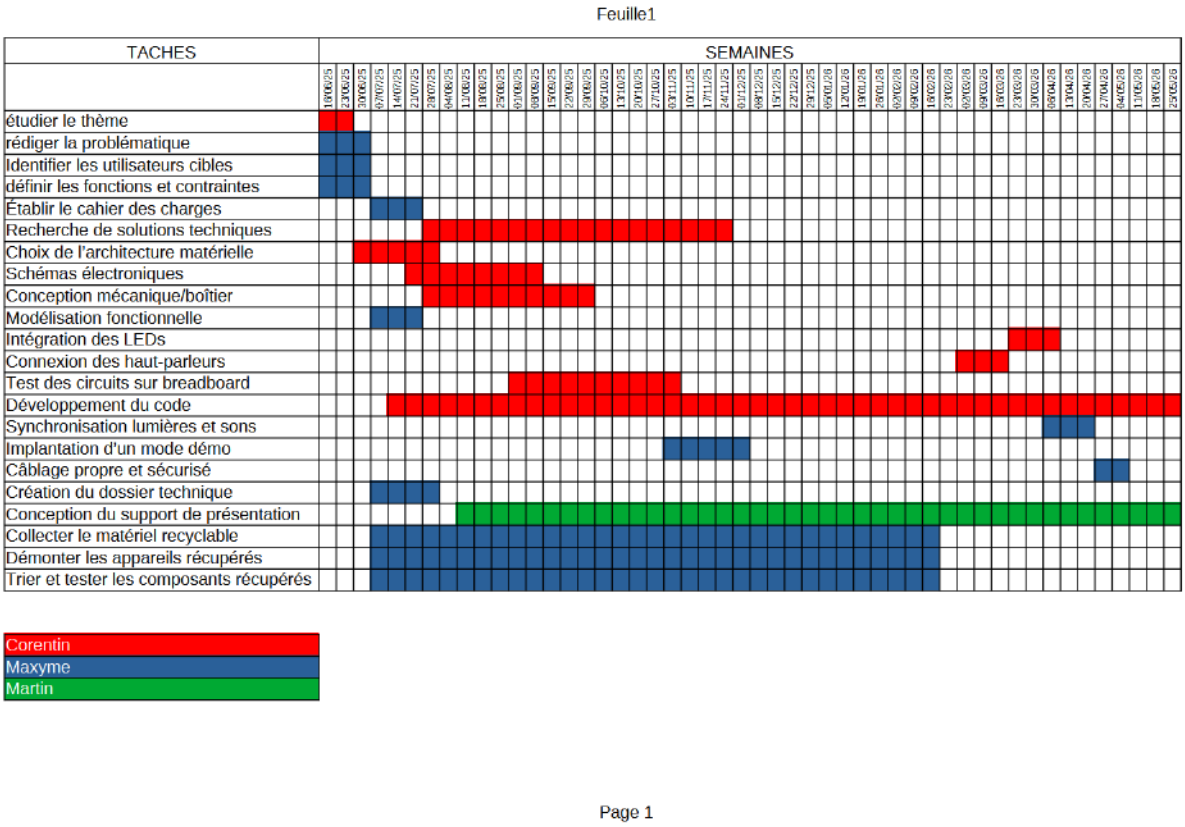


FIGURE 4.1 – Planning prévisionnel des différentes phases du projet

4.2 Budget prévisionnel

Le budget prévisionnel regroupe les principaux composants nécessaires à la réalisation du prototype. Une attention particulière est portée à l'utilisation de composants à faible coût ou disponibles au lycée afin de limiter les dépenses.

Poste de dépense	**Quantité**	**Prix unitaire (€)**	**Coût estimé (€)**	**Coût réel (€)**
Fraises 0.1 mm	10	0.40	7.99	7.99
Plaques de cuivre double face (FR4)	20	1.38	27.59	27.59
Pilotes L293D	5	1.37	6.85	6.85
PLA	1 kg	19.90	19.90	19.90
Câblage et connecteurs	120	0.08	9.99	9.99
Tiges métalliques 3 mm	12	0.75	8.99	8.99
PCB	10	5.05	25.75	0
Moteurs pas à pas à vis fin	50	2	100	0
Total			**217.05**	**81.31**

TABLE 4.1 – Budget prévisionnel du prototype

Ce budget reste relativement faible grâce à l'utilisation de composants accessibles et à la récupération de matériel. Ceci étant possible grâce à "La boutique informatique de Rouen", nous ayant directement

soutenus suite à notre demande de partenariat. Ce partenariat nous a permis de récupérer de nombreux lecteurs CD/DVD, dans lesquels nous avons récupéré nos actionneurs. Mais aussi avec le soutien de l'entreprise PCBWay, nous ayant gracieusement envoyé nos cartes PCB suite à notre demande. Cette approche permet de démontrer qu'un prototype fonctionnel peut être réalisé avec des moyens limités tout en conservant une bonne qualité de conception.

Chapitre 5

Critères d'évaluation pour les Concours

Afin d'orienter le développement du projet et de s'assurer de sa pertinence dans le cadre des concours scientifiques auxquels il est destiné, il est important d'identifier les principaux critères d'évaluation utilisés par les jurys. Ces critères permettent d'évaluer à la fois la qualité scientifique du travail, la pertinence de la démarche d'ingénierie et la capacité des élèves à communiquer leurs résultats.

5.1 Olympiades de Sciences de l'Ingénieur

Les Olympiades de Sciences de l'Ingénieur évaluent principalement la démarche de conception et la qualité de la réalisation technique. Les critères principaux sont les suivants :

- **Innovation** : caractère original du projet et capacité à proposer une solution nouvelle ou améliorée à un problème technique.
- **Approche scientifique** : utilisation rigoureuse de méthodes scientifiques pour analyser le problème, formuler des hypothèses et valider les solutions proposées.
- **Réalisation technique** : qualité de la conception matérielle et logicielle, fiabilité du prototype et pertinence des choix technologiques.
- **Démarche de projet** : organisation du travail, gestion des différentes phases de conception et capacité à documenter le processus de développement.
- **Communication** : clarté des explications, qualité du rapport et efficacité de la présentation orale.

5.2 Concours CGénial

Le concours CGénial met particulièrement l'accent sur la démarche expérimentale et la rigueur scientifique du projet.

- **Originalité et innovation** : caractère novateur du projet et capacité à explorer une idée scientifique ou technologique originale.
- **Qualité expérimentale** : mise en œuvre d'expériences pertinentes permettant de tester les hypothèses et de valider les résultats.
- **Rigueur scientifique** : précision des méthodes utilisées, analyse des résultats et capacité à discuter les limites du travail réalisé.
- **Pluridisciplinarité** : mobilisation de différentes disciplines scientifiques (physique, informatique, électronique, mécanique, etc.).
- **Présentation et communication** : qualité du dossier scientifique et capacité à présenter clairement les objectifs, la méthode et les résultats du projet.

5.3 European Union Contest for Young Scientists (EUCYS)

Le concours EUCYS récompense des projets scientifiques présentant un haut niveau de créativité, de rigueur scientifique et d'analyse critique.

— **Originalité et créativité** : capacité à proposer une idée nouvelle ou à aborder un problème scientifique sous un angle original.

— **Qualité scientifique** : profondeur de l'analyse, solidité du raisonnement et pertinence des méthodes employées.

— **Analyse critique** : capacité à interpréter les résultats obtenus, à identifier les limites du projet et à proposer des améliorations futures.

— **Présentation écrite et orale** : clarté du rapport scientifique, qualité de l'argumentation et efficacité de la communication avec le jury.

L'identification de ces critères permet d'orienter le développement du projet afin de répondre au mieux aux attentes des jurys. Une attention particulière est ainsi portée à l'innovation, à la rigueur scientifique et à la qualité de la présentation des résultats.

5.4 Partenaires

La réalisation de ce projet a été rendue possible grâce au soutien de plusieurs partenaires qui ont chacun apporté une contribution concrète à son développement. **La Boutique Informatique de Rouen** nous a fourni d'anciens appareils, dont la récupération de nombreux composants nous a permis d'expérimenter et de construire nos premiers prototypes. **PCBWay** a pris en charge la fabrication de nos cartes de développement, une étape essentielle pour concrétiser nos conceptions électroniques. Nous avons également pu utiliser **Matlab** pour certaines simulations. Enfin, le **Musée des Beaux-Arts de Rouen et la Métropole Rouen Normandie** nous ont ouvert leurs portes pour le tournage de notre vidéo de présentation, offrant un cadre idéal pour mettre en valeur notre projet.

Troisième partie

Étude théorique

5.5 Introduction générale

La partie *Étude théorique* a pour objectif de définir les bases scientifiques et mathématiques sur lesquelles repose le projet *TACTERA*. Avant toute réalisation pratique, il est indispensable de comprendre et de modéliser les phénomènes physiques, mécaniques et électriques impliqués, ainsi que les contraintes liées à la perception humaine.

Cette section fournit un cadre formel permettant de :

— modéliser la grille d'actionneurs, ses hauteurs et positions, ainsi que la conversion d'une image en relief tactile ;

— analyser le comportement électrique et dynamique des moteurs, leur alimentation et les interactions entre actionneurs ;

— évaluer les erreurs et limitations induites par la discrétisation de la surface et la résolution des moteurs ;

— identifier les contraintes liées à la perception tactile, afin de garantir une expérience utilisateur optimale.

L'approche adoptée combine :

— une modélisation mathématique précise, pour formaliser les relations entre les variables du système ;

— une analyse physique des composants, pour anticiper les performances et les limitations techniques ;

— une prise en compte des aspects perceptuels, afin de relier les données théoriques à l'expérience réelle de l'utilisateur.

En résumé, cette partie établit un cadre rigoureux et cohérent, qui servira de référence pour les études expérimentales et les développements pratiques décrits dans les parties suivantes du rapport. Elle permet de garantir que toutes les décisions de conception reposent sur des principes solides et vérifiables.

Chapitre 6

Principe Physique

La modélisation physique et notamment la modélisation électronique nous a permis de dimensionner les composants et les sous-systèmes de notre architecture. Nous avons pu, notamment grâce à ces différents modèles, estimer la puissance requise à l'alimentation optimale du système et l'intensité du courant traversant nos composants (mais aussi nos câbles), nous permettant un dimensionnement sécuritaire.

6.1 Modélisation électrique de la Grille d'Actionneurs

La grille d'actionneurs est composée de n moteurs bipolaires [moteur bipolaire]. Chaque moteur peut être modélisé par une inductance L en série avec la résistance de son bobinage R_m. La grille est alimentée par une source de tension idéale u_G, et la résistance des câbles est modélisée par R_c.

Dans des conditions idéales, la tension aux bornes de la grille est constante et la résistance de câbles est faible mais non nulle.

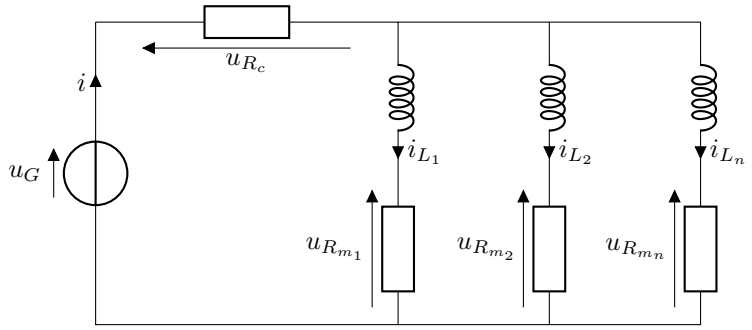


FIGURE 6.1 – Schéma électrique modélisant l'alimentation des moteurs bipolaires

6.1.1 Courant dans une inductance

Le courant traversant un moteur modélisé par une inductance L et une résistance R est donné par la relation fondamentale :

$$u_L(t) = L\frac{di_L(t)}{dt} + Ri_L(t)$$

On peut également écrire le courant sous forme intégrale :

$$i_L(t) = \frac{1}{L}\int_0^t u_L(\tau)\,d\tau + i_L(0)$$

où :

— u_L est la tension aux bornes de l'inductance,

— τ est la variable d'intégration,

— $i_L(0)$ est le courant initial dans l'inductance.

Dans le cas d'une tension constante u_L appliquée à une inductance idéale (sans résistance), le courant évolue linéairement :

$$i_L(t) = \frac{u_L}{L}\, t$$

Pour un circuit réel RL, la constante de temps est définie par :

$$\tau = \frac{L}{R}$$

Le courant dans ce cas évolue selon :

$$i_L(t) = \frac{U}{R}\left(1 - e^{-t/\tau}\right)$$

et atteint une valeur maximale :

$$I_{\max} = \frac{U}{R}$$

6.1.2 Puissance et énergie

La puissance instantanée consommée par un moteur est :

$$P_L(t) = u_L \cdot i_L(t)$$

Dans le cas d'une tension constante appliquée à une inductance idéale, on obtient :

$$P_L(t) = \frac{u_L^2}{L}\, t$$

L'énergie accumulée dans l'inductance (ou stockée dans le champ magnétique) est :

$$E_L(t) = \frac{1}{2} L i_L^2(t) = \frac{u_L^2}{2L} t^2$$

6.1.3 Moteurs en parallèle

Pour n moteurs identiques branchés en parallèle, le courant total fourni par la source est :

$$i_{\text{total}}(t) = \sum_{k=1}^{n} i_{L_k}(t)$$

et la puissance totale consommée :

$$P_{\text{total}}(t) = \sum_{k=1}^{n} P_{L_k}(t) = \sum_{k=1}^{n} \frac{u_{L_k}^2}{L_k} t$$

Chapitre 7

Modélisation mathématique

La modélisation mathématique vise à poser des bases théoriques puissantes, afin de permettre le bon déroulement du projet. Tous les concepts mathématiques utilisés dans notre projet ne sont pas explicités ici. Toutefois ce chapitre contient les éléments nécessaires à la logique de ce projet. Il constitue aussi une base de références utilisée au cours de ce rapport.

7.1 Modélisation de la grille

La surface tactile est modélisée par une matrice $H \in \mathbb{R}^{n \times n}$ représentant les hauteurs des actionneurs.

$$H = \begin{pmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,9} \\ h_{2,1} & h_{2,2} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ h_{9,1} & h_{9,2} & \cdots & h_{9,9} \end{pmatrix}$$

où :
$h_{i,j}$ = hauteur du pixel mécanique
(i, j) = position dans la grille

7.2 Modélisation de la conversion

7.2.1 Modélisation de la conversion

Une image que l'on souhaite convertir et afficher sur la surface tactile est une fonction d'intensité :

$$I(x, y)$$

avec :

— I : intensité lumineuse $(0 \to 255)$

— (x, y) : coordonnées dans l'image

Chaque cellule de la grille correspond à une zone de l'image.
La hauteur est donc définie par :

$$h_{i,j} = \frac{1}{|S_{i,j}|} \sum_{(x,y) \in S_{i,j}} I(x, y)$$

où :

— $S_{i,j}$: zone de pixels sur l'image associée au pixel mécanique (i, j)

— $|S_{i,j}|$: nombre de pixels dans cette zone

Autrement dit :

$$\text{la hauteur} = \text{moyenne de luminosité dans la zone}$$

Ensuite, on applique une normalisation :

$$h_{i,j}^{\text{norm}} = h_{\max} \cdot \frac{h_{i,j}}{255}$$

avec :

— $h_{\max}$: hauteur maximale du moteur

7.2.2 Approximation d'une surface continue

La surface tactile idéale peut être modélisée par une fonction continue :

$$z = f(x, y)$$

où :

— $f(x, y)$ représente la hauteur réelle du relief

La grille mécanique réalise un échantillonnage discret de cette surface :

$$f(i\Delta x, j\Delta y) \approx h_{i,j}$$

avec :

— Δx : largeur (côté) d'une cellule de la grille

— Δy : longueur (côté) d'une cellule de la grille

La surface générée par le dispositif correspond donc à une approximation discrète de la surface continue.

7.2.3 Chaîne de transformation

Le fonctionnement global du système peut être décrit par la transformation suivante :

$$I(x, y) \to H_{i,j} \to h_{i,j}^{norm} \to M_{i,j}$$

où :

— $I(x, y)$: image d'entrée

— $H_{i,j}$: matrice de hauteur calculée

— $h_{i,j}^{norm}$: hauteur normalisée

— $M_{i,j}$: position mécanique de l'actionneur

7.3 Erreur de discrétisation

L'utilisation d'une grille discrète introduit une erreur entre la surface réelle et la surface reconstruite (erreur de discrétisation).

On peut définir l'erreur locale :

$$e_{i,j} = f(x_i, y_j) - h_{i,j}$$

L'erreur moyenne sur toute la surface peut être définie par :

$$E = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |e_{i,j}|$$

où :

— n est la largeur de la grille carrée

Cette erreur dépend principalement :

— de la résolution de la grille

— de la complexité de la surface représentée

Plus la résolution n est grande, plus l'approximation de la surface réelle est précise. Ainsi la résolution est un réelle objectif pour ce projet, réduisant en conséquence l'erreur.

7.4 Résolution spatiale

La résolution spatiale de la surface tactile dépend du nombre d'actionneurs et de la taille physique de la surface.

Si la surface totale est carrée et possède une largeur L , la distance entre deux actionneurs est :

$$\Delta = \frac{L}{a}$$

où :

— L est la dimension physique de la surface

— a est le nombre d'actionneurs de la grille

Cette distance correspond à la résolution spatiale du dispositif.

7.5 Limite de perception tactile

La perception tactile humaine impose une limite physique à la résolution utile du dispositif (comme déterminé dans la section expérimentation).

La distance minimale perceptible entre deux points tactiles sur le doigt est approximativement :

$$d_{min} \approx 2 \text{ mm}$$

Ainsi, pour qu'un relief soit perceptible, la résolution du dispositif doit vérifier :

$$\Delta \geq d_{min}$$

Une résolution trop élevée n'apporte donc pas nécessairement une meilleure perception pour l'utilisateur.

7.6 Lissage de la surface

Afin d'éviter des variations brutales entre les cellules de la grille, un filtrage peut être appliqué à la matrice des hauteurs.

On peut par exemple utiliser un filtrage de moyenne locale :

$$h'_{i,j} = \frac{1}{8} \sum_{k=-1}^{1} \sum_{l=-1}^{1} h_{i+k,j+l}$$

Ce filtrage permet de lisser la surface tactile en remplaçant la valeur de chaque point par la moyenne des 8 points environnants et ainsi d'améliorer la continuité du relief perçu.

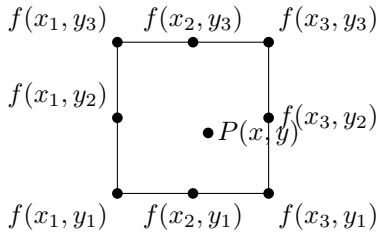


FIGURE 7.1 – Principe de moyenne local où $P(x, y)$ prend la valeur moyenne des 8 points environnants

Chapitre 8

Hypothèse

Dans le cadre de la modélisation théorique du projet *TACTERA*, certaines hypothèses sont posées afin de simplifier l'analyse et de rendre les calculs et simulations réalistes tout en restant accessibles. Ces hypothèses concernent la grille mécanique, les moteurs, l'alimentation électrique, et la perception tactile.

8.1 Hypothèses mécaniques

- Les actionneurs sont parfaitement alignés et équidistants.
- Chaque actionneur peut atteindre sa hauteur maximale $h_{\max}$ sans limitation mécanique ou perte de précision.
- Les frottements, vibrations ou déformations mécaniques sont négligés.
- La surface tactile est rigide et ne se déforme pas sous la pression de l'utilisateur.

8.2 Hypothèses électriques

- La source de tension u_G est idéale et constante dans le temps.
- La résistance des câbles R_c est faible et peut être considérée constante.
- Les moteurs sont identiques et parfaitement linéaires (modélisés par une inductance L et une résistance R_m).
- Les effets parasites (capacitances, interférences électromagnétiques) sont négligés.

8.3 Hypothèses de modélisation mathématique

- La surface réelle peut être représentée par une fonction continue $z = f(x, y)$.
- La conversion de l'image en hauteur se fait par moyenne locale des pixels dans chaque cellule.
- La discrétisation en matrice H est suffisamment fine pour représenter les reliefs significatifs.
- Les variations rapides ou très fines en hauteur peuvent être ignorées si elles ne sont pas perceptibles tactilement.

Ces hypothèses permettent de créer un cadre cohérent pour la modélisation et les calculs, tout en restant réaliste pour la conception expérimentale et la validation future du dispositif.

Quatrième partie

Conception Hardware

Chapitre 9

Introduction & Objectifs

9.1 Introduction

La partie **Hardware** est l'essence du projet *TACTERA*. C'est la passerelle entre un monde numérique essentiellement en deux dimensions, et notre monde physique.

Ce projet prend la forme d'une grille de 81 pixels répartis dans un carré de 9×9 pixels. Chaque pixel contient un moteur permettant leur translation et donc par association la représentation volumique. Nous verrons alors dans cette partie comment cette grille parvient à déplacer précisément chacun de ses pixels. Cette partie de part son aspect physique à demandé de nombreuse expérimentations et itérations amenant à la version actuelle.

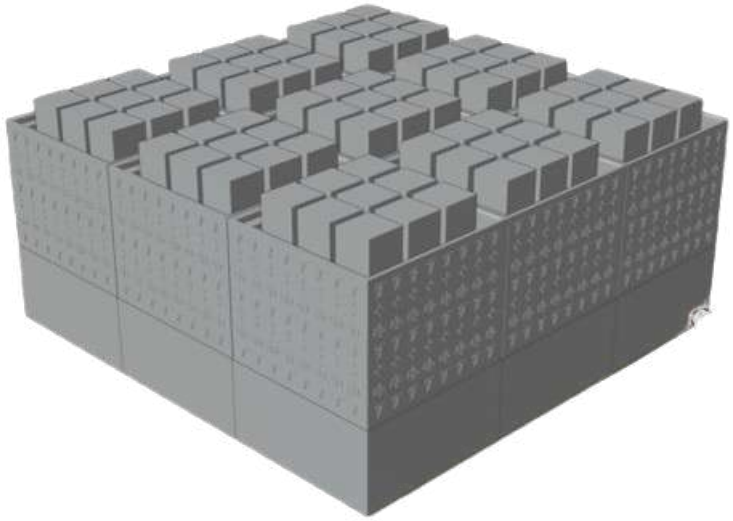


FIGURE 9.1 – Modélisation de la grille TACTERA complète*

*L'espace entre les modules provient des options d'assemblage, ceci est négligeable pour la résolution de la grille.

Chapitre 10

Architecture & Principe de fonctionnement

L'architecture globale du projet organise l'ensemble des composants qui le constituent. Elle a été remaniée à de nombreuses reprises afin de répondre au mieux aux critères de rapidité, d'optimisation et de réduction des coûts.

Le cœur de cette architecture (en excluant les contrôles provenant de l'ordinateur) est le master. Les données envoyées par le *master* données sont reçues par chaque sujet.

Ces *sujets* correspondent aux `modules` physiques contenant les pixels amovibles. Ce sont eux qui reçoivent les instructions du *master* et les transforment en mouvements mécaniques grâce aux moteurs présents dans chaque pixel. Dans notre grille de 81 pixels, un `module` est un bloc carré de 9 pixels ($3 * 3$).

L'architecture du *master* ne sera pas décrite car elle reste optionnelle sur le premier prototype pouvant être émulé par l'ordinateur ou le sujet lui même.

10.1 Première architecture

Avant d'arriver à l'architecture actuelle, présentée dans la suite de ce rapport, nous avons exploré plusieurs configurations du système, dont celle qui est décrite dans cette section.

Cette première architecture considère chaque actionneur/pixel comme un ensemble individuel et relativement indépendant. Elle est composée de 81 actionneurs, chacun disposant de son propre pilote (DRV8825), commandé par deux câbles de signal et alimenté par trois câbles d'alimentation.

Afin de pouvoir piloter chaque actionneur individuellement, nous avions conceptualisé un système de multiplexage dans lequel les deux câbles de signal doivent être activés simultanément pour déclencher le fonctionnement de l'actionneur. L'architecture prend alors la forme d'une grille de câbles de signal, dans laquelle chaque actionneur constitue un nœud permettant son activation individuelle.

On y trouve alors un démultiplexeur (DEMUX) prenant en entrée le signal nécessaire au fonctionnement de l'actionneur et l'envoyant sur la ligne correspondant au moteur ciblé. Nous pouvons représenter ce système comme une grille dans laquelle l'actionneur est identifié à l'aide de deux coordonnées indiquant sa position.Le tout étant piloté par un microcontrôleur.

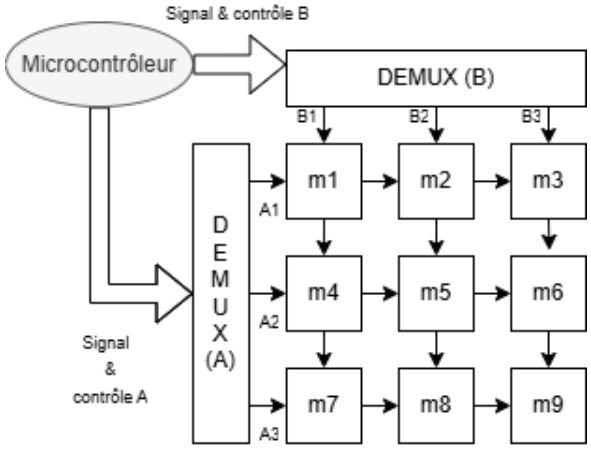


FIGURE 10.1 – Schéma de la première architecture. Pour activer le moteur 1 (m1), il faut activer le signal A1 et B1

Toutefois, cette architecture présentait un défaut majeur : un manque de modularité, ainsi qu'une latence induite par ce système. En effet, il n'était pas possible de faire se lever simultanément plusieurs moteurs appartenant à des lignes ou des colonnes différentes. Face à ces limitations, nous avons décidé de changer d'architecture afin d'adopter celle décrite dans les sections suivantes.

10.2 Architecture Sujet

Les modules sujet jouent un rôle majeur, devant assurer la récupération et traitement des données, ainsi que le mouvement des moteurs. Comme évoqué précédemment, chaque module contient 9 pixels et donc 9 moteurs bipolaires, permettant un contrôle précis de la montée. Ces moteurs nécessitent une électronique individuelle (`driver moteur`) pour assurer leur fonctionnement. Nous obtenons alors cette liste de critères :

— Chaque moteur nécessite son système de contrôle unique.
— Chaque module doit contenir un système de réception et de traitement des données.
— Chaque module doit être interconnectable (capable de se connecter avec les autres modules).

Nous avons donc abouti à l'architecture suivante :

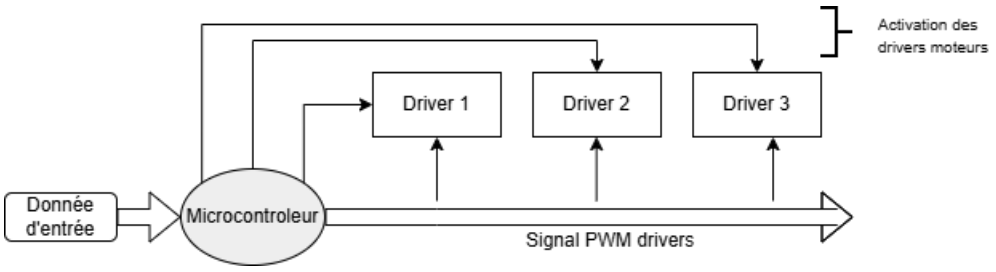


FIGURE 10.2 – Diagramme représentant l'architecture sujet

Nous pouvons voir sur le diagramme précédent que l'unité de traitement et de réception est un microcontrôleur. Celui ci génère tout les signaux `PWM` (*pulse with modulation*) permettant le contrôle des drivers moteurs. Ce choix de partage du signal découle de la limite de port de sortie disponible sur le microcontrôleur ce que nous détaillerons dans la partie électronique. D'autre part, notre objectif étant d'avoir des moteurs ayant les capacités de fonctionner indépendamment des autres, chaque driver moteur est équipé d'une entrée permettant son activation (et donc leur individualité).

10.3 Architecture globale

L'architecture globale permet de contrôler et de faire fonctionner les différents modules ensemble. Pour cela, nous utilisons le processus de communication I2C (*Inter-Integrated Circuit*) permettant une liaison suffisamment rapide entre chaque module et ne nécessitant que de seulement deux sorties/entrées de microcontrôleur. De plus, il convient pour la transmission de signal de courte portée (environ un mètre) et appartient aux protocoles fréquemment intégrés nativement aux microcontrôleurs modernes. Ainsi, chaque module possède son adresse, lui servant à interagir avec le contrôleur principal (master).

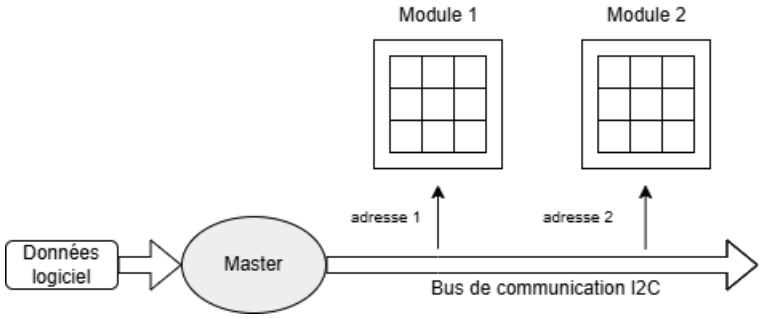


FIGURE 10.3 – Diagramme représentant l'architecture globale

Chapitre 11

Mécanique & Modélisation

11.1 Introduction & Objectifs

L'objectif de la partie mécanique dans ce projet, est de créer un mouvement de translation ascendant ou descendant pour chaque pixel. Et cela en adoptant une précision maximale. Dès lors, il nous fallait trouver une solution technologique mêlant précision et simplicité d'utilisation. Nous avons alors choisi l'énergie électrique comme moteur de notre translation, et pour cela l'exploitation d'un moteur `pas à pas` (qui a une résolution d'un pas de vis). Dans notre cas d'une résolution de 18° par pas.

La mécanique du pixel s'est donc construite autour de cet objectif.

11.2 Mécanique module sujet

11.2.1 Mécanique Pixel

La mécanique pixel assurant la translation verticale de celui-ci, doit faire face à des contraintes que sont la gestion des déformations, le dégagement thermique du moteur, et la simplicité d'assemblage.

Pour effectuer le mouvement vertical, le moteur est équipé d'une vis fin, transformant le mouvement rotatif en mouvement linéaire. Le moteur est alors fixé verticalement au châssis du module, tandis qu'une partie supérieure (la partie amovible) lui est reliée à l'aide de dents biaisées. Cette solution technologique nous semble la plus efficace de part sa simplicité d'utilisation et son efficacité.

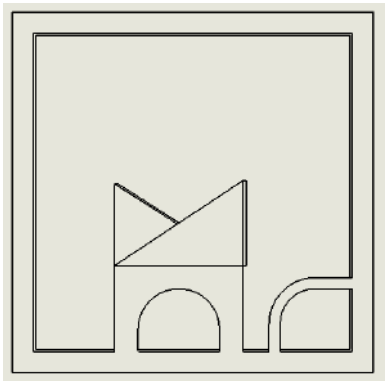


FIGURE 11.1 – Vue de haut du système de liaison entre le pixel amovible et le moteur

Pour autoriser un degré de liberté seulement selon l'axe du moteur, il est nécessaire d'utiliser deux tiges en acier permettant à la partie supérieure du pixel de coulisser et maintenant le contact entre la partie coulissante du pixel et la vis du moteur. La partie supérieure du pixel doit allier légèreté et résistance

pour permettre la plus grand facilité d'ascension, elle pèse alors environ trois grammes utilisant du PLA (polyester thermoplastique semi-cristallin) comme matériau. Il mesure 70 mm de haut pour 15.5 mm de coté, et admet une translation maximale de 40 mm.

Le `pixel` est alors constitué de deux parties individuelles :

— Le `support moteur`, maintenant le moteur et assurant la fixation sur le module (fixe).

— La `coque du pixel`, assurant la liaison avec le moteur et se déplaçant (mobile).

11.2.2 Modélisation pixel

La modélisation a pour objectif de produire les pièces mécaniques, pour ainsi les étudier et itérer sur notre *design*.

Elle a ainsi permis de réduire les effets de la déformation sur le `support moteur` nécessitant une liaison parfaite avec la `coque du pixel`. Cette déformation est due à la technologie d'impression *3D* que nous utilisons pour produire le `support moteur` qui générait une flèche (déformation) entre la vis du moteur et le système d'accroches de la `coque du pixel`. Provoquant alors une perte de hauteur du pixel, voir même un blocage du moteur. Il est alors renforcé aux endroits les plus susceptibles de se déformer. Il est aussi équipé d'entailles pour la dissipation thermique passive.

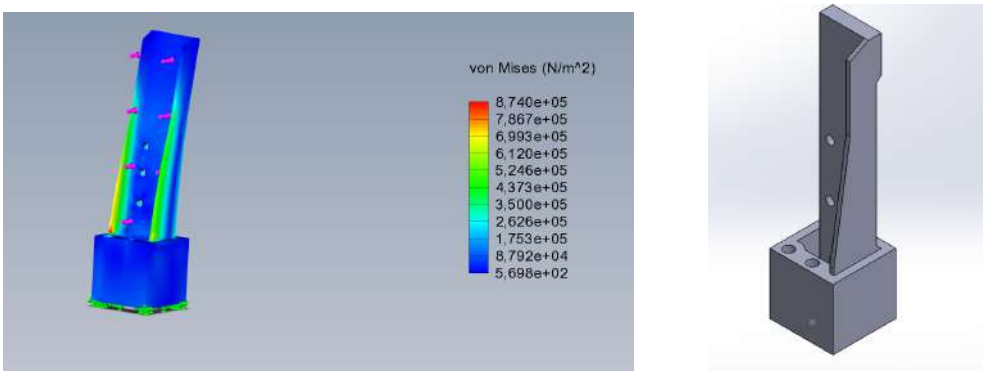


FIGURE 11.2 – Simulation de déformation sous 1N et modèle du support moteur

Nous avons aussi utilisé de nombreux assemblages pour tester les interactions entre les différents composants mécanique, permettant de réduire le nombre de prototypes. Toutefois un des défis majeur est resté la détermination optimale du système de liaison entre le moteur et la coque du pixel.

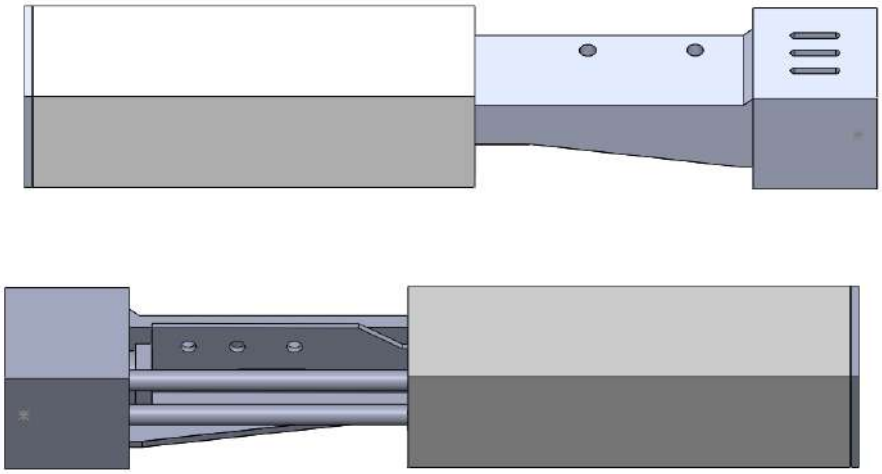


FIGURE 11.3 – Différentes vues du pixel assemblé

11.2.3 Modélisation Module

Le module complet ne possède pas d'élément mécanique à proprement parler, il vise à assembler ensemble les 9 pixels et à contenir l'électronique (3 cartes) lui étant destiné.

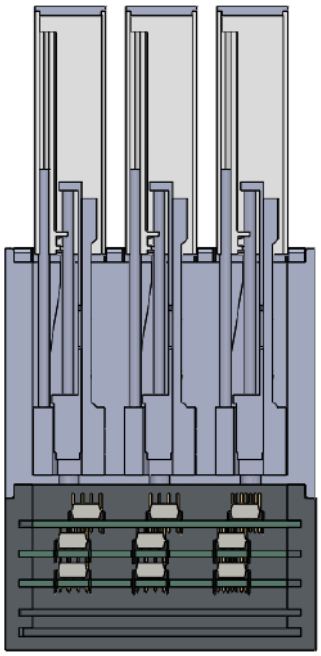


FIGURE 11.4 – Vue en coupe du module sujet assemblé

Il mesure 68*68 mm, et possède deux majeures parties, le `compartiment électronique` et le `compartiment pixel` s'assemblant à l'aide d'une liaison complète. Dans le compartiment pixel, chaque pixel possède quatre perçages, lui permettant d'être fixé de façon rigide à l'aide de vis.

Chapitre 12

Électronique

12.1 Introduction & Objectifs

Dans le projet *TACTERA*, l'électronique joue un rôle clé étant donné son architecture demandant le pilotage de nombreux moteurs (représentant une tache complexe) mais aussi le traitement de nombreuses commandes de contrôle.

D'autre part, les circuits électroniques conçus doivent être accessibles à bas coût et/ou usinable à l'aide d'une machine à commandes numériques.

L'architecture électronique du système s'organise autour de plusieurs circuits principaux :

— le circuit des modules d'actionneurs, chargé du pilotage individuel des moteurs ;

— le circuit d'alimentation, assurant la distribution de l'énergie électrique au sein du dispositif ;

— le circuit de commande principal (*master*), responsable du traitement des commandes et de la coordination globale du système.

L'objectif de cette partie est donc de présenter la conception de ces différents circuits, leurs principes de fonctionnement ainsi que leur intégration dans l'architecture globale du projet.

12.2 Électronique Module Sujet

12.2.1 Pilotage des moteurs

Un des principaux défis auxquels nous avons dû faire face en électronique est le pilotage des moteurs. Les moteurs que nous utilisons sont des moteurs bipolaires, ceux-ci contiennent alors deux bobines reliées à quatre câbles. Pour faire tourner ces moteurs, il est alors nécessaire de leur appliquer une tension déterminant son sens de rotation, et cela pour chaque bobine. Pour pouvoir contrôler le sens de déplacement du courant dans chacune des bobines , nous utilisons un `double H-bridge` (double pont en H) permettant de piloter le sens du courant dans l'inductance (représentant une bobine) et donc le sens de rotation comme illustrée ici :

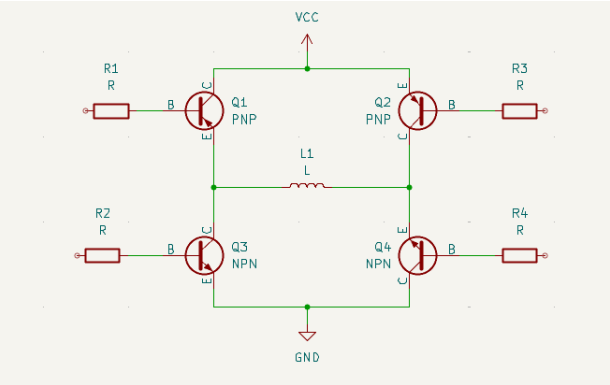


FIGURE 12.1 – Principe de commande d'une bobine par un pont en H. L'activation croisée des transistors (Q1–Q4 ou Q2–Q3) permet d'inverser le sens du courant dans l'inductance L, déterminant ainsi le sens de rotation du moteur.

Cette opération de pilotage (driver) est effectuée par le circuit intégré `L293D` intégrant un double pont en H permettant ainsi de piloter un moteur. De plus, ce circuit intégré comporte des entrées pour l'activation ou la désactivation du circuit, ainsi nous pouvons à l'aide d'un simple câble activer ou désactiver un moteur en lui attribuant un état haut ou un état bas.

12.2.2 Circuits & PCB

Le circuit assurant le fonctionnement d'un module suit une architecture électronique spécifiquement conçue pour le pilotage des actionneurs. On y retrouve un bus de communication permettant la transmission des signaux de contrôle vers les doubles ponts en H associés à chaque moteur. Chaque moteur dispose également d'une ligne d'activation dédiée permettant d'autoriser ou d'interdire son fonctionnement.

Le circuit comporte plusieurs entrées d'alimentation distinctes :

— une alimentation moteur de `5.0 V / 3 A`, destinée à fournir l'énergie nécessaire aux actionneurs ;

— une alimentation logique de `5.0 V / 500 mA`, utilisée pour les circuits de commande ;

— une masse commune assurant la référence électrique de l'ensemble du système.

L'ensemble du module est conçu pour être piloté par un microcontrôleur de développement. Dans le cadre de ce projet, une carte *Arduino Uno* a été utilisée pour assurer le contrôle des différents modules.

Afin d'optimiser l'encombrement et de faciliter l'intégration mécanique, chaque module est constitué de trois cartes électroniques distinctes. Chacune de ces cartes intègre trois circuits pilotes de moteurs de type *L293D*, permettant ainsi de contrôler plusieurs actionneurs tout en conservant une architecture compacte et modulaire.

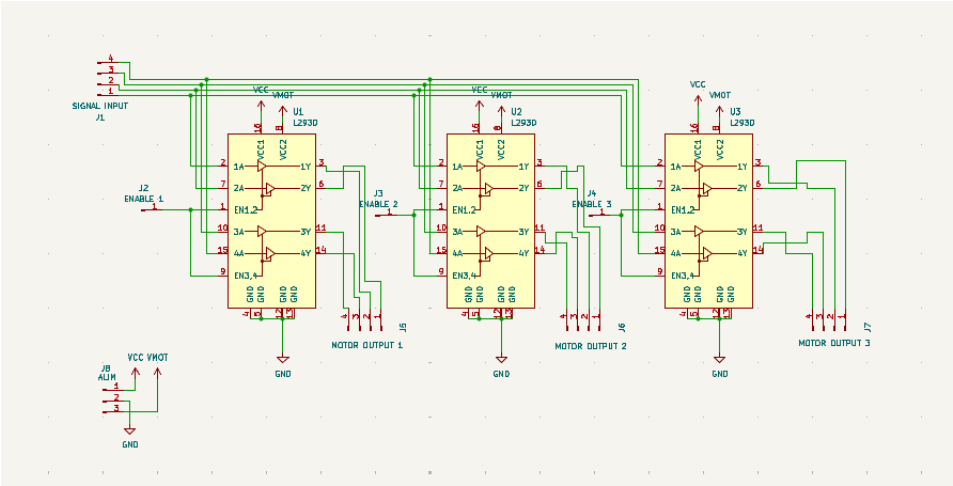


FIGURE 12.2 – Schéma électronique du module sujet

La recherche de simplicité dans la production de la carte électronique (`PCB`), nous a amenés à une carte ne possédant que deux couches de cuivre, facilitant ainsi le prototypage.

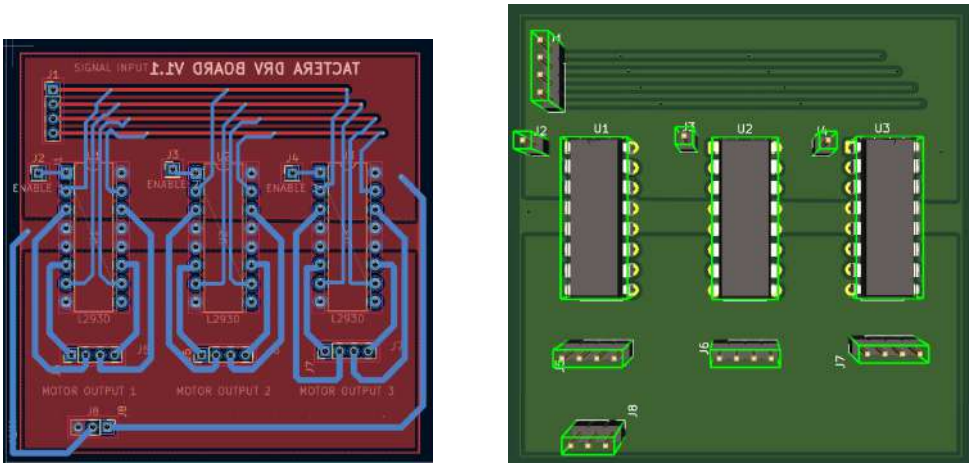


FIGURE 12.3 – Circuit imprimé du module sujet

12.2.3 Simulation

Avant la production, nous avons à de nombreuses reprises testé et simulé numériquement le fonctionnement de nos circuits électriques à l'aide du logiciel `Proteus`, nous permettant de garantir un fonctionnement optimal.

Ainsi dans les figures ci-dessous, nous pouvons observer une simulation impliquant deux cartes *Arduino Uno*, 5 drivers/pilotes, 5 moteurs bipolaires ainsi que de multiples outils de mesure tels que des terminaux, un oscilloscope et un détecteur de communication pour le protocole `I2C`. Leurs branchements respectent l'architecture du module précédemment évoqué, incluant un microcontrôleur en tant que `master` et un autre en tant que `sujet`. Cette phase de simulation nous a permis de valider une nouvelle fois notre architecture.

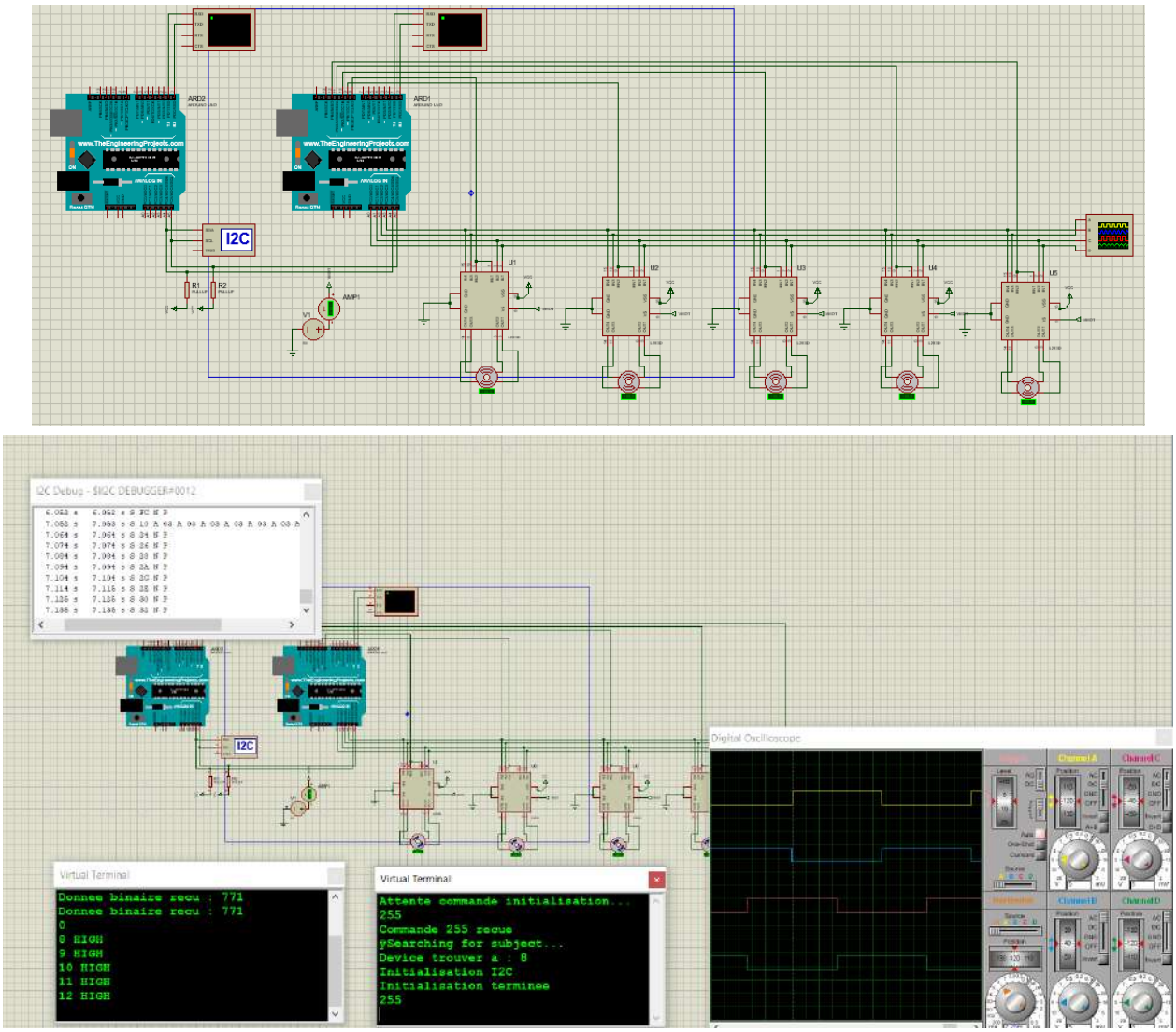


FIGURE 12.4 – Simulation sur le logiciel `Proteus` de l'architecture des modules

Chapitre 13

Assemblage

13.1 Assemblage

Étant donné le nombre important de pixels et de modules nécessaires pour atteindre les dimensions finales du dispositif, il a été nécessaire de mettre en place un véritable processus de fabrication. Celui-ci vise à rendre la production des différents éléments du système à la fois simple, rapide et reproductible.

Dans cette optique, plusieurs outils spécifiques ont été conçus afin de faciliter l'assemblage des différentes pièces constituant un pixel mécanique. Ces outils permettent notamment de positionner et d'assembler avec précision les éléments suivants : le support moteur, les tiges de guidage en acier, le moteur ainsi que la coque du pixel.

Par ailleurs, des outils de test ont également été développés afin de vérifier le bon fonctionnement des pixels avant leur intégration dans le prototype final. Ces dispositifs de test permettent d'alimenter et de piloter les actionneurs individuellement, garantissant ainsi un contrôle qualité préalable et limitant les risques de défaillance une fois les modules assemblés dans la structure complète.

FIGURE 13.1 – Système d'assemblage composé d'une partie supérieure pour enfoncer les tiges ainsi qu'une partie inférieure de maintien

Cinquième partie

Développement Software

Chapitre 14

Introduction & Objectifs

14.1 Introduction

Cette partie du rapport se concentre sur l'explication détaillée des principes de fonctionnement du logiciel, en mettant en lumière la manière dont il traite et interprète les données numériques. Elle décrit également les différentes méthodes employées pour transformer ces données abstraites en reliefs physiques tangibles, en abordant à la fois les algorithmes de conversion, les modèles de simulation utilisés, et les techniques de modélisation appliquées. Enfin, cette section présente les choix techniques qui ont été opérés pour garantir un fonctionnement à la fois efficace et fiable du système, en prenant en compte les contraintes matérielles, les exigences de précision, ainsi que les stratégies mises en œuvre pour optimiser la performance globale.

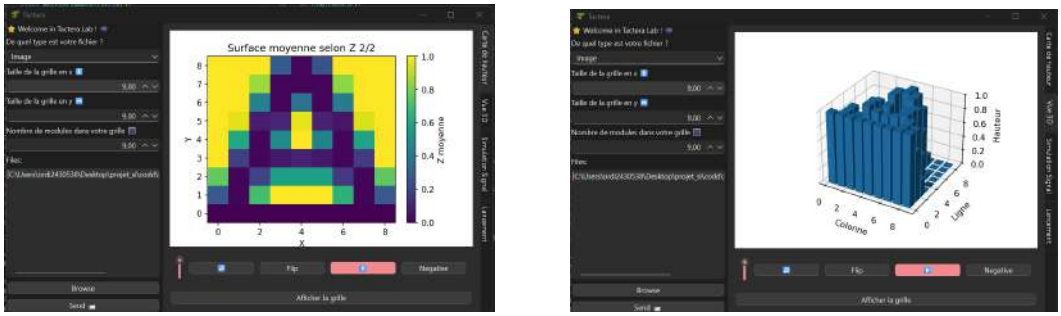


FIGURE 14.1 – Logiciel permettant le contrôle de la grille

Chapitre 15

Conversion

15.1 Introduction

Un des objectif majeurs de ce projet étant de permettre la représentation 3D d'objets, d'œuvres n'y étant pas destinées à l'origine, une grande partie du présent programme se doit de *convertir* des fichiers au format divers (image, objets 3D, vidéos...) et de résolution supérieure en quelque chose de compréhensible pour notre grille d'actionneurs : la hauteur de chacun de ses pixels. Pour réaliser cette tache, deux programmes seront décrits : le programme de conversion d'images vers des reliefs suivi de celui sur la conversion d'objets 3D en reliefs.

15.2 Images

15.2.1 Présentation

Cette section présente le principe de conversion d'une image numérique en une représentation volumique adaptée à la grille d'actionneurs du dispositif *TACTERA*.

L'objectif de cet algorithme est de transformer une image d'entrée de dimension $i \times j$ en une grille de dimensions $n \times n$, correspondant à la disposition physique des pixels mécaniques du dispositif. Chaque cellule de la grille de sortie représente une hauteur proportionnelle à la luminosité moyenne de la zone correspondante dans l'image originale.

Cette transformation permet ainsi de générer une surface discrète dont les reliefs sont cohérents avec les variations d'intensité de l'image. La hauteur maximale attribuable à chaque cellule est limitée par la course maximale des actionneurs mécaniques.

La conversion d'une image en surface discrète est une opération classique du traitement d'image, reposant sur des techniques de discrétisation et de réduction de résolution (*downsampling*) [2]. Dans le cadre de ce projet, cette étape constitue le lien entre la représentation numérique de l'image et sa restitution physique sous forme de relief tactile.

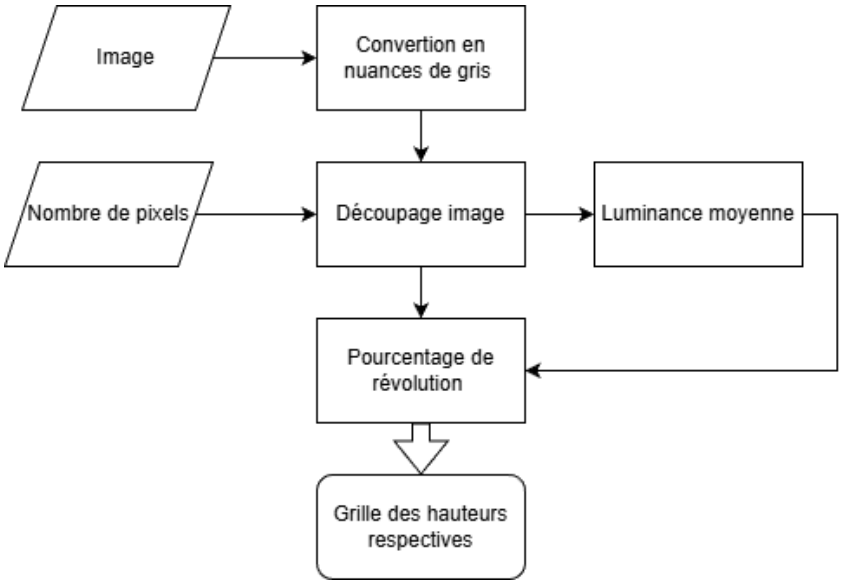


FIGURE 15.1 – Diagramme représentant le processus de conversion d'une image en grille de reliefs mécaniques

15.2.2 Conversion en nuances de gris

Cette première partie essentielle se base sur la librairie *Pillow*, permettant d'ouvrir l'image dans un premier temps mais surtout de la convertir en nuance de gris en utilisant une transformation de luminance ITU-R 601-2 décrite par la formule suivante :

$$L = R \times 0,2989 + G \times 0,5870 + B \times 0,1140$$

Ainsi, nous transformons notre image en une fonction d'intensité lumineuse $((0 \to 255))$: $I(x, y)$

Listing 15.1 – Conversion en nuance de gris

```
from PIL import Image
# ouvrir l'image et la convertir en nuance de gris
image = Image.open(fileName).convert('L')
```

15.2.3 Découpe de l'image en tuiles

Dans cette fonction, nous divisons l'image d'entrée en $n \times n$ tuiles correspondant aux $n \times n$ actionneurs. Chaque tuile est définie comme un ensemble de pixels appartenant à une zone rectangulaire de l'image

Pour une image de dimensions $i \times j$, et une grille comportant $n \times n$ tuiles (organisées en $n \times n$), chaque tuile $T_{k,v}$ ($T_{k,v}$ correspond à $S_{i,j}$ dans le modèle mathématique) est définie par :

$$T_{k,v} = \{I(x, y) \mid k\Delta_i \leq x < (k+1)\Delta_i,\ v\Delta_j \leq y < (v+1)\Delta_j\}$$

avec :

$$\Delta_i = \frac{i}{n}, \qquad \Delta_j = \frac{j}{n}$$

et

$$k \in [0, n-1]\,, \qquad v \in [0, n-1]\,.$$

où : $T_{k,v}$: la tuile de position (k, v) regroupant l'ensemble des pixels de cette zone de l'image.

Listing 15.2 – Algorithme de découpe

```
    # stockage des dimensions
    W, H = image.size[0], image.size[1]

    # calcul de la largeur d'une tuile
    w = W/cols

    # calcul de la hauteur d'une tuile selon l'echelle
    h = w/scale

    # calcul du nombre de colonnes
    rows = int(H/h)
    pict = np.zeros((cols,rows))

    for j in range(rows):
            y1 = int(j*h)
            y2 = int((j+1)*h)

            # corriger la derniere tuile
            if j == rows-1:
                y2 = H

            for i in range(cols):

                # decoupe de l'image en tuiles
                x1 = int(i*w)
                x2 = int((i+1)*w)

                # correction de la derniere tuile
                if i == cols-1:
                    x2 = W

                # decoupe de l'image pour en extraire les tuiles
                img = image.crop((x1, y1, x2, y2))
```

15.2.4 Calcul luminescence moyenne

La fonction `getAverageL` prend en entrée une tuile de pixels en nuances de gris précédemment découpée et retourne la luminescence moyenne de cette tuile selon l'expression suivante :

$$L_{moy} = \frac{1}{n^2} \sum_{(x,y) \in T_{k,v}} I(x,y)$$

Listing 15.3 – Fonction de calcul de luminescence

```
1
2  def getAverageL(image):
3
4      """
5     Etant donne une image PIL, renvoyer la valeur moyenne des niveaux de gris.
6      """
7      # obtenir l'image sous forme de tableau numpy
8      im = np.array(image)
9
10     # obtenir la forme
11     w,h = im.shape
12
13     # obtenir la moyenne
14     return np.average(im.reshape(w*h))
```

15.2.5 Course modules

Cette dernière partie de notre programme de conversion a pour but de retourner le pourcentage de la course maximale (h_{max}) que l' actionneur devra parcourir (défini sur $z \in [0; 1]$) :

$$z = \frac{avgL}{255}$$

Si la translation maximale d'un actionneur est de 40mm et que le pourcentage de la course maximale (lui étant attribué) est 0.50, alors le moteur devra parcourir 20mm.

Cette valeur est ensuite insérée à l'emplacement $(k, v$, dans le code $i, j)$ lui étant dédié inséré ensuite à l'emplacement lui correspondant dans la grille.

Listing 15.4 – Pourcentage de course

```
1  pict[i, j] = round((avg / 255), 4)
```

15.3 Objets 3D

15.3.1 Présentation

La présente section décrit le traitement des objets tridimensionnels dans le cadre du projet *TACTERA*. Le logiciel prend en entrée un objet 3D au format *stéréolithographique* (STL), qui est défini comme un ensemble de triangles dont chacun est caractérisé par les coordonnées (x, y, z) de ses sommets.

Pour assurer une représentation physique fidèle sur la grille d'actionneurs, il est recommandé d'utiliser des modèles avec une maille suffisamment dense (haute résolution). Une résolution élevée permet non seulement de simplifier les transformations matricielles appliquées aux triangles, mais également d'augmenter le réalisme du relief reproduit sur la surface tactile.

Le processus consiste ensuite à discrétiser l'objet 3D afin de calculer les hauteurs correspondantes pour chaque cellule de la grille, en tenant compte de la résolution spatiale et des limites mécaniques des actionneurs.

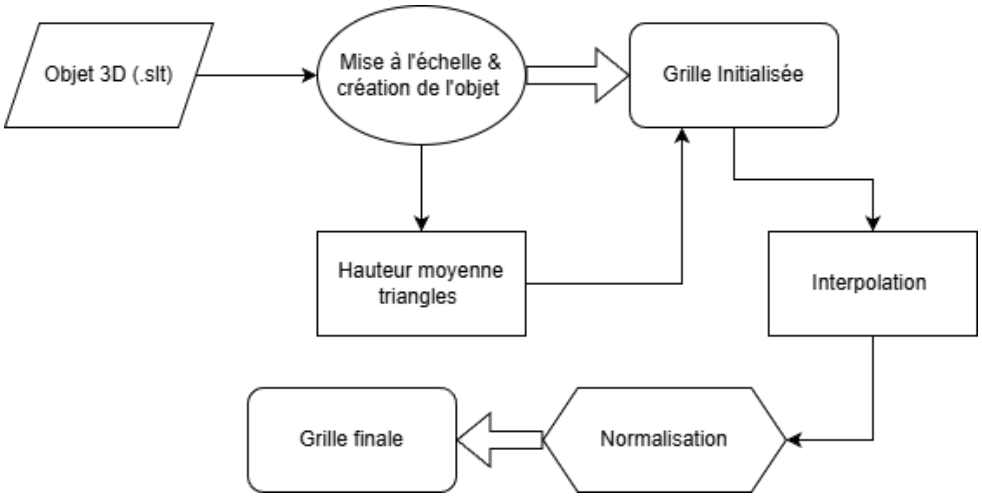


FIGURE 15.2 – Diagramme représentant le processus de conversion des objets tridimensionnels en grilles de reliefs

15.3.2 Mise à l'échelle & création de l'objet

Le processus suivant a pour objectif d'accepter un fichier entrant du type *stéréolithographique* (.stl) et les dimensions d'une grille $n \times n$ où n^2 est le nombre d'actionneur, puis de retourner un modèle aux dimensions de notre grille interprétable par notre programme.

Pour cela, il est nécessaire de :

— Transformer le fichier `.stl` importé en un `mesh` sur lequel nous pourrons utiliser les fonctions fournies par la bibliothèque **stl**, mais aussi les transformations possibles proposées par la bibliothèque **numpy** ;

— Déterminer les dimensions du modèle importé à l'aide de la fonction utilitaire `find_mins_maxs` pour pouvoir le mettre à l'échelle grâce à la fonction `translate` fournie par la bibliothèque stl. ;

Listing 15.5 – Fonction utilitaire de calcul de dimensions

```
def find_mins_maxs(obj):
    minx = obj.x.min()
    maxx = obj.x.max()
    miny = obj.y.min()
    maxy = obj.y.max()
    minz = obj.z.min()
    maxz = obj.z.max()
    return minx, maxx, miny, maxy, minz, maxz
```

Pour déterminer la nouvelle échelle translation nous utilisons une matrice de translation générée en calculant la différence entre le produit de l'échelle et de la valeur maximale de position et la valeur de position suivant la relation suivante :

Soit notre objet de dimensions $x \times y \times z$ on définira les dimensions de notre matrice de translation par :

$$(x_{max} \cdot \text{échelle} - x_{max}) \times (y_{max} \cdot \text{échelle} - y_{max}) \times (z_{max} \cdot \text{échelle} - z_{max})$$

Listing 15.6 – Fonction de mise à l'échelle & de création de l'objet

```
def scale_object(file, gridx, gridy):

    # Chargement du fichier STL
    cube_mesh = mesh.Mesh.from_file(file)

    minx, maxx, miny, maxy, minz, maxz = find_mins_maxs(cube_mesh)

    scale = gridx/maxx

    cube_mesh.translate(np.array([((maxx*scale)-maxx),((maxy*scale)-maxy),((maxz
        *scale)-maxz)]))

    return cube_mesh
```

15.3.3 Initialisation de la grille

Ce processus est primordial initialisant une grille où chaque position contient une liste (1D) vierge. Elle est destinée à accueillir la hauteur (coordonnée en `z`) de chaque triangle se trouvant dans la zone assignés à cette position (ce système est comparable à celui des *tuiles*, dans la section sur la conversion des images).

Les étapes pour parvenir à cette initialisation sont les suivantes :

— Nous initialisons un tableau de dimensions $n \times n \times t$ où la dimension t représente la liste initialisé pour accueillir la position en z des triangles.À l'initialisation nous avons $t = 1$.

— De la même manière que pour la conversion des images, nous créons des zones rectangulaires (définies par des intervalles appartenant à $[x_{min}; x_{max}]$ et $[y_{min}; y_{max}]$), dans lesquelles toutes les valeurs collectées seront ajoutées à la liste correspondante de notre tableau. Ces zones rectangulaires sont stockées dans les variables `x_bin` et `y_bin`.

Listing 15.7 – Fonction d'initialisation de la grille

```
def grid_init(cube_mesh, n_x, n_y):

    minx, maxx, miny, maxy, minz, maxz = find_mins_maxs(cube_mesh)

    x_bins = np.linspace(minx, maxx, n_x + 1)
    y_bins = np.linspace(miny, maxy, n_y + 1)

    grid = np.empty((n_x, n_y), dtype=object)

    # Initialise chaque case avec une liste vide
    for ix in range(n_x):
        for jy in range(n_y):
            grid[ix, jy] = []

    return grid, x_bins, y_bins
```

15.3.4 Hauteurs moyennes

Nous nous intéressons dorénavant au cœur du processus de conversion : la détermination de la hauteur moyenne des triangles.

— Dans la première boucle `for`, nous récupérons chaque sommet de triangles puis le plaçons dans la liste correspondant à leur position sur la grille (selon les rectangles définis par `x_bin` et `y_bin`) leur hauteur en z.

— La seconde partie du processus consiste à faire la moyenne de toutes les hauteurs de triangle conservé dans chaque liste. Pour cela nous faisons de nouveau appel à un processus itératif parcourant chaque liste pour effectuer cette moyenne.

15.3.5 Interpolation & Normalisation

Cette partie consiste en un outil puissant pour pallier les problèmes que nous pose la transformation de modèles 3D au format `.stl` en une grille de résolution différente :

— La résolution du modèle influence le nombre de triangles et donc la répartition dans l'espace de leurs sommets ce qui pose un problème dans le calcul de moyenne car certains des *pixels* de notre grille se retrouvent sans valeur.

— Cette même résolution crée des moyennes très différentes sur l'ensemble de la grille, n'étant pas compatibles avec le pourcentage de revolution de nos moteurs.

Pour pallier cela nous utilisons deux outils mathématiques que sont l'interpolation bilinéaire (estimer la valeur à partir des pixels voisins) et la normalisation pour transformer nos moyennes très différentes en valeurs réelles comprises entre 0 et 1.

Pour l'interpolation nous pourrions résoudre linéairement l'équation à quatre inconnues en fonction de ses voisins :

$$\begin{cases} f(x_1, y_1) = ax_1 + by_1 + cx_1y_1 + d \\ f(x_2, y_1) = ax_2 + by_1 + cx_2y_1 + d \\ f(x_1, y_2) = ax_1 + by_2 + cx_1y_2 + d \\ f(x_2, y_2) = ax_2 + by_2 + cx_2y_2 + d \end{cases}$$

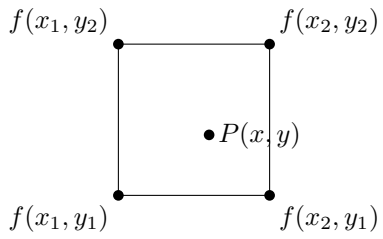


FIGURE 15.3 – Principe de l'interpolation bilinéaire : la valeur en $P(x, y)$ est estimée à partir des quatre points voisins.

Mais pour une plus grande flexibilité nous utilisons le module `ndimage` de la bibliothèque `scipy` fonctionnant de la même manière selon les voisins proches des chaque pixel.

Listing 15.8 – Interpolation

```
1
2 filled = ndimage.generic_filter(z_grid, np.nanmean, size=3, mode='nearest')
```

Pour la normalisation nous calculons simplement comment chaque valeur se positionne par rapport à la valeur la plus haute et la plus basse selon la formule suivante :

$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

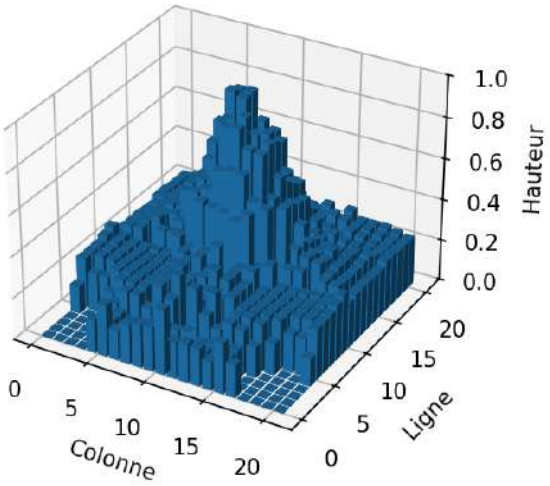


FIGURE 15.4 – Résultat du processus de conversion d'objet 3D, ici pour le Panthéon

15.4 Divers

15.4.1 Contrôle Manuel

Le logiciel possède aussi une fonction *dessin*, permettant à son utilisateur d'interagir manuellement avec la grille, indiquant la hauteur de chaque élément sur celle-ci. Ce système permet aussi la création d'animation, en permettant de créer les différentes grilles s'enchainant.

FIGURE 15.5 – Représentation de la partie du logiciel attribuant à chaque pixel une couleur correspondant à une hauteur (pourcentage de la hauteur maximale)

Chapitre 16

Simulation

16.1 Introduction & objectifs

La partie de simulation de notre logiciel est majeure, c'est elle qui permet l'étude et l'amélioration de notre système en simulant la communication et l'envoi de données à chaque module ainsi que leurs interactions entres eux, permettant ainsi une boucle de retour dans l'envoi et la génération des signaux de commande.

Le système de simulation, du fait de sa robustesse et dans l'objectif d'obtenir une qualité de simulation des plus précises, est utilisé par le système de contrôle de la grille. Ainsi, son rôle est clé dans la démarche de ce projet.

Dans cette section, nous utiliserons de nombreuses *queue* permettant de simuler une communication de série entre chaque élément, laissant leurs processus et leurs boucles d'éxecution indépendantes. Ces *queue* sont aussi utilisées pour la surveillance de chaque processus.

16.1.1 Détermination de la variation

Précédemment nous avons créé une liste de grille de dimension $n \times n$, nous laissant avec un tableau en trois dimensions contenant toutes nos grilles de la forme $n \times n \times d$. Notre grille physique (Hardware) nécessite des instructions sur le nombre de pas nécessaires pour faire correctement avancer ses moteurs, il nous est alors nécessaire de lui indiquer comment ces moteurs se déplacent au fil des figures y étant affichées.

Pour cela il nous faut calculer la variation entre chaque grille, se traduisant alors par les différents mouvements que devront faire les moteurs au cours du temps.

Pour des questions d'optimisation, ce processus s'effectue de manière vectorielle, permettant une simulation instantanée et évitant un nombre de boucles trop excessif.

— Nous créons tout d'abord un masque, contenant tous les éléments d'une grille différents de celle précédente

— Ce masque est ensuite utilisé pour indiquer quels éléments nécessitent le calcul de variation suivant : $var = valeur_{actuelle} - valeur_{precedente}$

— Les valeurs de variation sont placées sur une grille de dimensions identiques à celles d'origine où sont aussi placées les valeurs invariantes.

Listing 16.1 – Fonction de calcul de la variation

```
mask = last_layer != prev_layer
delta = np.zeros_like(last_layer)
delta[mask] = last_layer[mask] - prev_layer[mask]
```

16.1.2 Découpe de la grille en modules

Maintenant que nous avons déterminé la variation entre chaque grille, il nous faut découper cette même grille de variation en morceaux correspondant à chaque modules de notre grille physique. Actuellement le carré est la seule forme utilisée pour la grille physique, ainsi le nombre de modules `sujet` la constituant est un carré parfait.

Dès lors que nous avons le nombre, et les dimensions du module, il nous suffit de découper la grille en utilisant une boucle doublement imbriquée présente dans le code suivant :

Listing 16.2 – Extrait fonction de découpe

```
nx = x // n
ny = y // n
tils = []
    for i in range(n):
            for j in range(n):
         tile = grid[i*nx:(i+1)*nx, j*ny:(j+1)*ny].ravel(order='C')
         tils.append(tile)
```

16.1.3 Émulation Master

Le `master` détient un rôle clé dans notre architecture, il est le nœud recevant la trame principale envoyée par l'ordinateur contrôleur et la diffusant à chaque module `sujet`. Ainsi sa simulation est primordiale, étant à la tête de notre chaine de communication le `worker` dans lequel son programme sera exécuté en boucle (pour simuler le fonctionnement du microcontroleur) sera initialisé en dernier laissant les `worker` des modules `sujet` d'initialiser. Nous remontons notre chaine de communication pour vérifier l'intégrité et l'exécution de chaque programme avant l'exécution de la simulation.

Initialisation

La première étape consiste à vérifier l'intégrité de la trame reçue. Pour cela nous attendons la récupération de la trame complète du bit de commencement à celui de fin. Le modèle de la trame d'initialisation envoyé au `master` est la suivante :

1	**nombre de modules**	**nombre de pixels par module**	**vitesse des moteurs**	**0**

FIGURE 16.1 – Trame de configuration reçue par le master

Une fois cette vérification effectuée, nous traitons ces données en convertissant les données binaires (simulées) codées sur 2 octets signés en entier naturel à l'aide du code suivant :

Listing 16.3 – Convertion en entiers naturels

```
int.from_bytes(value, 'big', signed=True)
```

Une fois les données reçues, nous envoyons les données de vitesse à chaque module `sujet` à travers leurs *queue* respectives.

Traitement des données

Une fois le système initialisé, nous démarrons sa boucle de traitement (en dernier dans notre cycle de démarrage).

Celui-ci itère sur le nombre de modules, collectant la trame individuelle de chaque module, et l'envoyant à travers la *queue* destinée à ce module. La trame transmise est quant à elle de la forme suivante, contenant les pas issus du découpage des modules et du calcul de leur variation :

1	taille de la trame	liste des pas	0

FIGURE 16.2 – Trame transmise par le master

16.1.4 Émulation Sujet

Le sujet possède le rôle de l'exécutant dans notre processus. Le code de simulation lui étant dédié ne contient ni la gestion des moteurs ni l'architecture de gestion qui en découle. Son objectif est d'intercepter les données envoyées par le `master`, les décoder et de les renvoyer via de nouveau une *queue* au programme de simulation principal lequel reconstruira à travers une représentation graphique les morceaux de grilles reçus.

16.1.5 Reconstruction & Représentation Graphique

Lors de l'exécution des différents éléments de simulation énoncés précédemment, le programme de simulation va récupérer les données d'exécution de la part des sujets, qu'il va pouvoir utiliser pour reconstruire la grille obtenue en utilisant le procédé inverse à celui de découpe en module abordé précédemment.

En parallèle de la reconstruction, nous affichons la progression à travers une carte de hauteur de cette même grille.

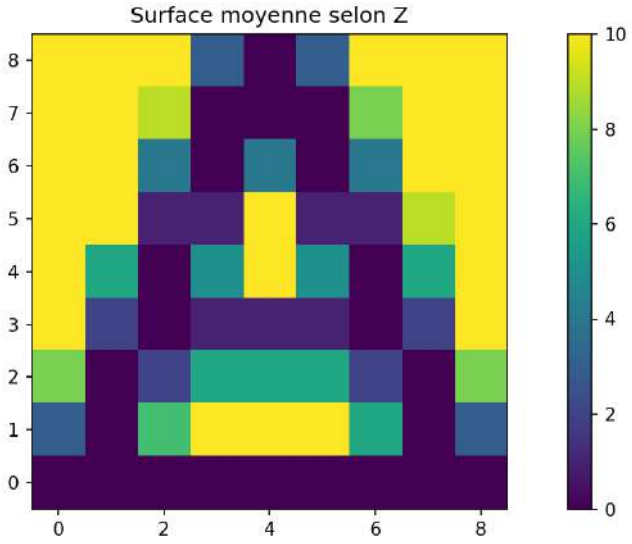


FIGURE 16.3 – Représentation graphique lors de la simulation

Une fois le graphique statique, la simulation est achevée, mettant fin à ce chapitre.

Chapitre 17

Fonctionnement sur Prototype Physique

17.1 Introduction & objectifs

Une fois le système entrainé et fonctionnel, il nous est nécessaire de déterminer la logique de traitement des modules `sujet` qui elle n'a pas été traitée précédemment.Cette logique doit respecter l'architecture adoptée par les cartes de chaque module (`driver`), énoncée précédemment (hardware). Nous rendrons aussi compte dans ce chapitre des différents essais physiques, impliquant le programme (software).

17.2 Fonctionnement Module Sujet

Comme énoncé dans la partie `hardware` l'activation des modules se fait de façon généralisée créant le même signal pour tous les drivers (l293d). Partons d'un exemple pour décrire ce système. Imaginons un module avec simplement 3 moteurs, m_1, m_2, m_3. À ces moteurs nous associons la liste suivante de pas à effectuer de dimensions 3 : $[10, 20, 30]$.

— Nous commençons par activer et faire tourner de `10` pas les moteurs m_1, m_2, m_3. Lorsque cette instruction est terminée nous obtenons la liste suivante : $[0, 10, 20]$

— Le moteur m_1 n'ayant plus de pas à exécuter, nous pouvons faire tourner de `10` pas les moteurs m_2, m_3. Nous laissant la liste suivante : $[0, 0, 10]$

— Il nous suffit finalement de faire tourner de `10` pas le moteur m_3, pour achever la liste de pas initiale.

Pour plus de précisions sur les techniques matérielles utilisées veuillez vous référer à la partie Hardware.

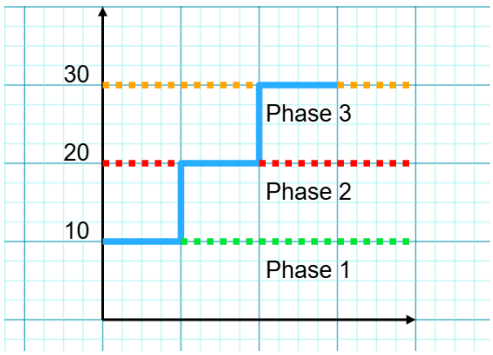


FIGURE 17.1 – Schéma des phases d'exécution

Plus formellement, pour une liste de pas positif, nous trions cette liste par ordre croissant, puis activons tous les modules pendant n-pas où n est le nombre minimal de pas dans la liste. Nous soustrayons ce nombre à tous les autres de la liste à la suite de l'exécution des premiers pas. Ce processus est itéré jusqu'à ce que plus aucun moteur ne soit activé.

Les fonctions du logiciel des modules `sujet` sont alors : de recevoir la trame lui étant destinée (déjà traité dans la simulation), d'associer chaque moteur à ses pas, de les trier par ordre croissant et finalement de gérer l'exécution des différentes phases (activation successive des moteurs).

Association & Tri

Pour ne pas perdre les pas originellement associés à chaque moteur, de part l'ordre dans lesquels ils arrivent dans la trame, nous devons les associer dans le programme du module sujet. Pour cela nous créons simplement un tableau dans lequel nous associons le `pin` de contrôle du moteur avec le nombre de pas lui étant destiné.

Le tri quant à lui est assez simple, il consiste à vérifier pour chaque élément de la liste si le suivant est plus petit, si oui, ils inversent leur place tandis que sinon, ils restent à leur position.

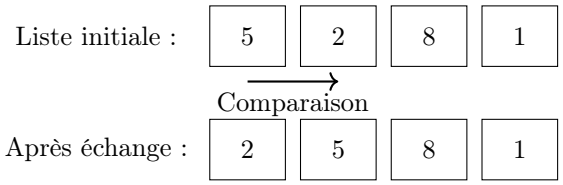


FIGURE 17.2 – Illustration du tri des pas dans la liste

Cette tache est effectuée par le code suivant (tout code dédié au matériel est en C++) :

Listing 17.1 – Tri des pas

```
void sort_step_list(int step_pin[][2], int size){
  for ( int j=0; j<size - 1; j++){
    for ( int i=0; i<size - 1; i++) {
      int temp[2];
      if (step_pin[i][0] > step_pin[i+1][0]){
        temp[0] = step_pin[i][0];
        temp[1] = step_pin[i][1];
        step_pin[i][0] = step_pin[i+1][0];
        step_pin[i][1] = step_pin[i+1][1];
        step_pin[i+1][0] = temp[0];
        step_pin[i+1][1] = temp[1];
      };
    }
  }
}
```

Exécution phases moteur

Comme expliqué précédemment, chaque moteur est progressivement désactivé au cours des phases. Ce qui est très simple à faire grâce au tri précédent. Pour cela, nous utilisons ce code itérant sur les différents éléments de la liste en désactivant les `pin` des moteurs une fois que leur valeur de pas est dépassée.

Listing 17.2 – Exécution phases moteur

```
for (int i=0; i<abs(step_pin[size-1][0]); i++){ // Selon le plus grand element
    signal_generator.step(increment);
            for ( int j = start_place; j < size; j++){
            if (step_pin[j][0] == i){
              digitalWrite(step_pin[j][1], LOW);
            }
          }
    delay(5);
}
```

Sixième partie

Expérimentation

Chapitre 18

Perception Tactile

18.1 Introduction & Objectifs

Les mains humaines constituent des capteurs sensoriels particulièrement sophistiqués. La détermination de leur résolution tactile est donc essentielle afin de comprendre les limites de la perception humaine et d'optimiser l'expérience utilisateur dans le cadre de ce projet.

L'étude de cette résolution vise à établir les bases théoriques nécessaires à la conception du dispositif, ainsi qu'à garantir son adéquation avec les capacités perceptives des utilisateurs.

18.2 Expérimentation

Nous définissons la résolution tactile de la main comme la distance minimale entre deux points de pression que le sujet est capable de percevoir comme deux stimuli distincts.

L'expérimentation consiste donc à appliquer simultanément une pression sur deux pointes séparées d'une distance d. Cette pression est appliquée par la main dominante de l'individu. Un point de contact est placé sous la pulpe de l'index, tandis que le second est appliqué au centre de la paume. Cette variation de position permet d'étudier deux niveaux de résolution tactile : une résolution fine au niveau de l'index et une résolution plus grossière au niveau de la paume.

Les pointes utilisées pour l'expérience correspondent aux deux extrémités supérieures d'un pied à coulisse électronique, offrant une précision de mesure de 0,01 mm. Les mesures sont réalisées à intervalles réguliers, avec un espacement progressif de 0,50 mm entre les deux pointes.

18.3 Résultats

Cette expérience a été réalisée sur quatre individus d'âges et de sexes différents (16 à 47 ans), permettant d'obtenir une estimation de la sensibilité tactile représentative de plusieurs profils.

Les mesures obtenues indiquent une distance de discrimination moyenne de 1,64 ± 0,01 mm au niveau de la pulpe de l'index, tandis que celle mesurée au centre de la paume est de 5,77 ± 0,01 mm.

Les résultats détaillés pour un individu sont présentés dans le tableau suivant à titre d'exemple.

Mesures cibles (mm)	0,50	1,00	1,50	2,00	2,50	3,00	10,00	8,00	7,00	6,00
Mesures réelles (mm)	0,52	1,02	1,53	2,00	2,49	3,00	10,13	8,09	7,05	5,99
Index (pulpe)	N	N	N	O	O	O	O	O	O	O
Paume (centre)	N	N	N	N	N	N	O	O	O	O

TABLE 18.1 – Mesure effectuée sur un individu droitier. Les "N" correspondent à la perception d'une seule pointe, tandis que les "O" correspondent à la perception de deux pointes distinctes.

Chapitre 19

Prototypage

19.1 Introduction & Objectifs

Le prototypage s'inscrit dans la chaine d'expérimentation nécessaire à la réalisation de ce projet. C'est une étape primordiale, nous permettant de vérifier la théorie, établir les caractéristiques de nos architectures, et finalement d'obtenir un projet physique. Il se décline en plusieurs catégories :

— Le prototypage électronique à l'aide de cartes de test et de *breadboard* (carte perforée d'expérimentation).

— Le prototypage mécanique, permettant la modification des pièces imprimées en `3D`.

— Le prototype fonctionnel, assurant les fonctionnalités du PMV (produit minimum viable) énoncé dans le cahier des charges.

19.2 Prototypage électronique

Avant son implémentation dans le projet, chaque sous système s'est vu soumis à de nombreux tests et prototypes. Cette phase nous a été nécessaire dans un premier temps pour vérifier si ces sous systèmes étaient fonctionnels. Puis dans un second temps, pour connaitre les atouts, mais aussi les faiblesses de ces sous systèmes.

	Moteur	**Driver/pilote**	**Microcontroleur**
Courant	X	X	X
Vitesse maximale	X		
Couple	X		
Précision	X	X	X
Vitesse de réponse		X	X
Surchauffe	X	X	X

FIGURE 19.1 – Tableaux indiquant par une croix les tests effectués sur chaque sous système électronique

Afin d'effectuer les expérimentations sur l'électronique, nous avons dans un premier temps testé tout les sous systèmes en utilisant des grilles de prototypage, tel que celui du pont en H (*H-bridge*). Ainsi nous avons pu garantir une precision des expérimentations correcte.

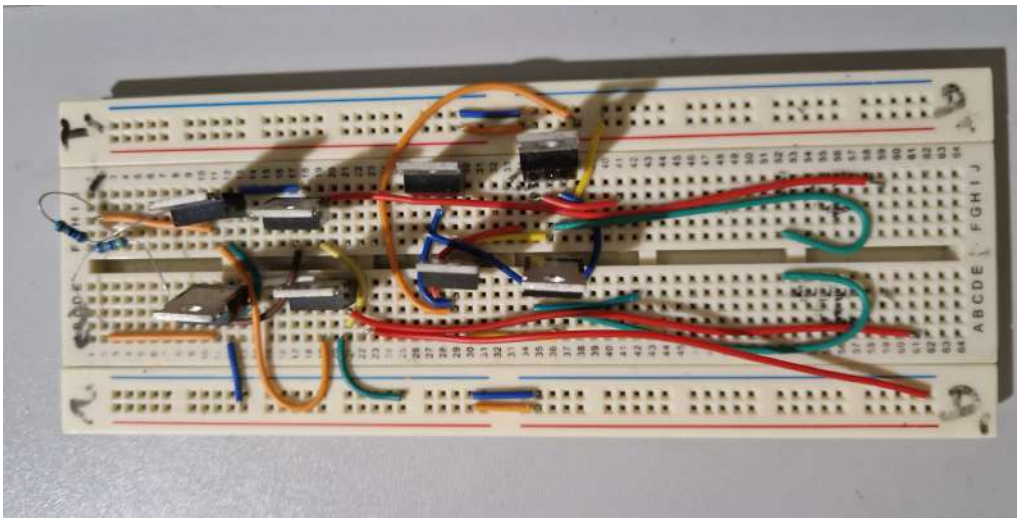


FIGURE 19.2 – Représentation de la grille de prototypage du pont en H

Une fois les tests individuels passés, nous avons décidé de produire dans un premier temps nos propres circuits imprimés supportant et connectant les différents éléments électriques (PCB) à l'aide d'une machine à commande numérique, ce qui a de nouveaux induit de nombreux tests afin de prendre en main ce système de production. Toutefois, la prise en main de cette machine nous a obligés à optimiser l'organisation de notre carte `PCB`, ce qui nous a permis d'obtenir une version de meilleure qualité de la carte mais aussi d'économiser beaucoup de temps (livraison PCB produit $\approx$ 2 semaines).

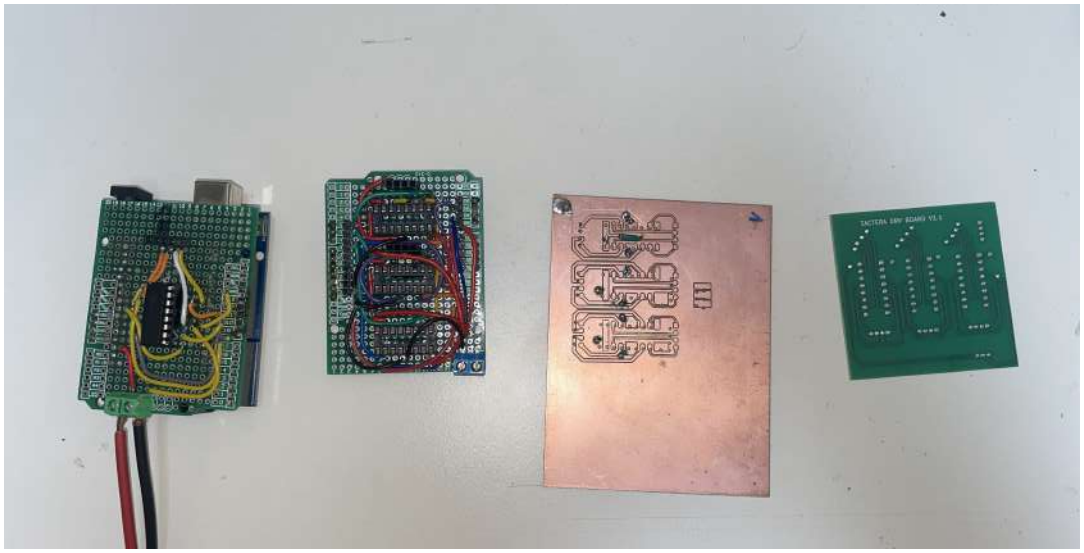


FIGURE 19.3 – Évolution des prototypes électroniques au cours du projet

19.3 Prototypage mécanique

Le prototypage mécanique a quant à lui représenté une très bonne technique dans l'élaboration du système de liaison le plus efficace avec le moteur. Ces multiples itérations nous ont permis de corriger et contrebalancer les défauts que peuvent induire l'impression 3D (distorsion des perçages, affaissement des structures verticales, friction...).

FIGURE 19.4 – Évolution des prototypes mécaniques au cours du projet

19.4 Prototype final

Le **prototype final** constitue une version réduite du système complet, limitée à un seul module. Ce module est lui-même composé de neuf pixels actifs, représentant une unité fonctionnelle élémentaire de la grille d'actionneurs.

Ce prototype permet de réaliser l'ensemble des expérimentations nécessaires à cette étape du développement, notamment l'étude de la *réactivité du système*, la validation des *protocoles de communication* entre les différents éléments ainsi que l'évaluation des capacités de *perception sensorielle*.

Il répond ainsi aux exigences d'un **Produit Minimum Viable (PMV)**, en offrant une plateforme expérimentale suffisamment représentative du système final tout en conservant une complexité maîtrisée.

Cet outil expérimental joue également un rôle essentiel dans le processus d'itération du projet. Il permet notamment d'identifier, d'analyser et de corriger différents problèmes techniques, en particulier ceux liés aux communications entre modules. De ce fait, il constitue une étape intermédiaire indispensable avant la réalisation d'un prototype à plus grande échelle.

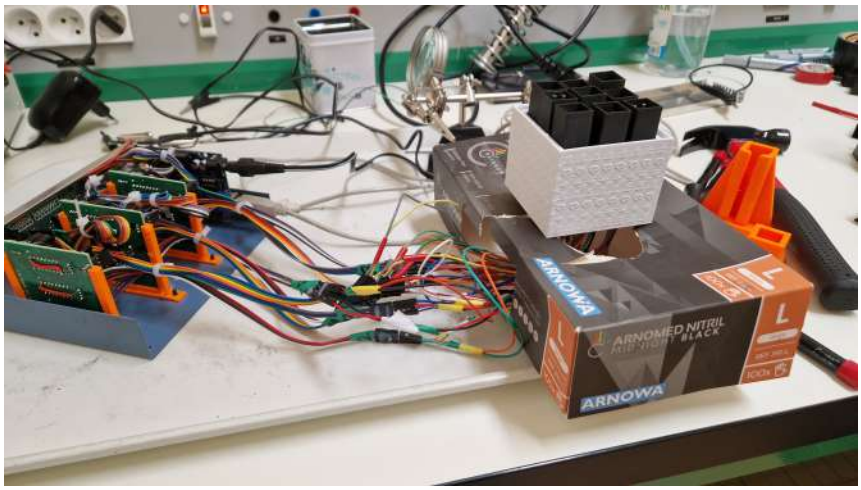


FIGURE 19.5 – Vue latérale du prototype final

Mesures

À cette étape de notre processus d'expérimentation, nous avons pu comparer nos estimations, issues du modèle physique et mathématiques, avec les mesures prises directement sur notre prototype final.

Une des premières étapes pour les estimations théoriques consiste en la mesure des valeurs physiques de base de nos éléments tels que la résistance de nos moteurs ainsi que leur inductance.

La valeur de l'inductance s'obtient grâce à la mesure de sa valeur de résonance dans un circuit composé d'une résistance, d'un condensateur et d'une inductance (circuit RLC), décrite par :

$$f_0 = \frac{1}{2\pi\sqrt{LC}} \implies L = \frac{1}{(f_0 2\pi)^2 C}$$

où :

— f_0 : la fréquence de résonance de l'inductance

— L : l'inductance

— C : la capacitance du condensateur

Ainsi, après plusieurs essais, nous obtenons une valeur de $L = 7.1mH \pm 0.3mH$ par bobine (utilisée pour le courant instantané) avec $R_{\text{bobine}} = 12.0\Omega$ et $U_{\text{moteur}} = 5.0v$. Dès lors, nous pouvons calculer toutes nos valeurs théoriques, que l'on comparera avec les valeurs mesurées.

Nous prendrons $P_{\text{moy}} \approx I_{\text{max}} \times U$. Ainsi qu'une grille contenant le même nombre d'actionneurs que le prototype final (9).

Ce tableau propose une comparaison entre les valeurs mesurées et les valeurs estimées. Les valeurs instantanées ont été volontairement exclues afin de mettre en évidence une représentation globale et moyenne du fonctionnement du système.

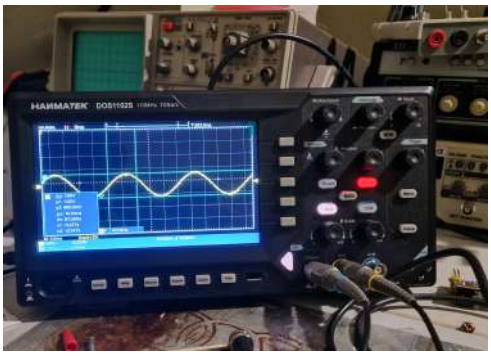
	Valeur estimée	**Valeur mesurée**
courant bobines moteur	$I_{\text{max}} = 0.41A$	$0.33A$
Puissance moteur	$2.05W$	$1.65W$
Puissance totale des moteurs	$18.4W$	$15.0W$

FIGURE 19.6 – Tableau comparant les mesures prises sur le prototype final avec celles estimées

Le prototype étant alimenté par un transformateur convertissant une tension de $220\,V$ vers $5\,V$ avec un courant maximal de $5\,A$, nous avons mesuré les différentes caractéristiques de sa consommation à l'aide d'un wattmètre.

Les mesures obtenues indiquent un courant de $0.12\,A$ et une puissance de $17W$.

(a) Mesure de la consommation avec un wattmètre



(b) Mesure de l'inductance du moteur à l'aide d'un oscilloscope

FIGURE 19.7 – Mesures expérimentales réalisées sur le prototype

Septième partie

Conclusion

19.5 Résultats

Les résultats obtenus sur le prototype final permettent d'évaluer les mêmes critères que ceux définis pour chaque sous-système (partie Hardware, section prototypage électronique).

La transmission des données a initialement posé problème, entraînant des échecs lors des premiers tests. Ces difficultés ont conduit à une révision de la communication entre les sous-systèmes, ce qui a permis de stabiliser la transmission et d'atteindre un fonctionnement correct.

En ce qui concerne la partie *software*, les performances se sont révélées très satisfaisantes. Le temps moyen de conversion pour une grille de 9×9 a été mesuré à environ `500 ms` en conditions réelles. Pour la réponse du module aux commandes du contrôleur, le temps total jusqu'à l'élévation complète des pixels est de $2,50$ s, incluant un temps d'initialisation de $2,00$ s.

19.6 Limites

Malgré ces résultats positifs, certaines limites ont été identifiées :

— La complexité et le volume de la structure électronique actuelle rendent difficile l'intégration complète de tous les composants dans le module ;

— Le bruit généré par le mouvement des modules reste important et peut être gênant lors des démonstrations ;

— Les fonctionnalités actuelles du logiciel sont limitées à la conversion et à la commande basique de la grille, sans assistance pour les utilisateurs ayant des besoins spécifiques.

19.7 Comparaison avec un système de référence : inFORM

19.7.1 Objectif de la comparaison

Afin de mieux situer le prototype *TACTERA* par rapport aux technologies existantes, nous avons choisi de le comparer au système *inFORM*, développé par le MIT Tangible Media Group.

Cette comparaison permet de mettre en évidence les choix de conception réalisés pour TACTERA, ainsi que les compromis techniques associés. Elle permet également d'évaluer les différences entre les deux systèmes en termes de résolution, de complexité matérielle, de coût et de domaines d'application.

19.7.2 Tableau comparatif

Critère	**inFORM (MIT)**	**TACTERA**
Objectif principal	Démonstrateur de *shape-display* haute résolution destiné à la recherche, à la téléprésence et à la visualisation dynamique de formes 3D.	Prototype modulaire conçu principalement pour l'accessibilité, la médiation scientifique et les démonstrations pédagogiques.
Résolution et échelle	Très grand nombre d'actionneurs (plusieurs centaines voire milliers), permettant de créer une surface physique presque continue.	Grille physique de 9×9 pixels (81 pixels). La résolution est volontairement limitée afin de réduire la complexité technique et le coût du système.
Dimensions d'un pixel	Actuateurs linéaires compacts disposés à forte densité.	Pixel d'environ 15,5 mm de côté, hauteur totale d'environ 70 mm et translation maximale d'environ 40 mm.
Actionneurs et mécanique	Actionneurs linéaires individuels contrôlés électroniquement, souvent avec un retour de position.	Moteurs pas à pas bipolaires associés à un mécanisme de vis permettant de transformer la rotation du moteur en translation verticale.
Architecture électronique	Système électronique dédié capable de piloter individuellement un très grand nombre d'actionneurs.	Architecture modulaire composée d'un contrôleur maître et de modules 3×3 gérant chacun 9 pixels.
Drivers moteurs	Drivers industriels spécialisés adaptés aux actuateurs linéaires.	Drivers L293D (pont en H) pilotés par des microcontrôleurs de type Arduino.
Alimentation	Alimentations puissantes nécessaires pour gérer le grand nombre d'actionneurs.	Alimentation typique de 5 V / 3 A pour les moteurs et 5 V / 500 mA pour l'électronique de contrôle.
Conversion données $\rightarrow$ relief	Pipeline logiciel avancé permettant de convertir des surfaces 3D complexes en relief physique dynamique.	Conversion d'images ou de fichiers STL en une grille de hauteurs : niveaux de gris, moyenne par cellule, normalisation puis interpolation bilinéaire.
Applications principales	Recherche en interaction homme–machine, design interactif et téléprésence physique.	Accessibilité pour les personnes malvoyantes, médiation culturelle, démonstration scientifique et projets éducatifs.
Coût et industrialisation	Prototype de recherche relativement coûteux et complexe à industrialiser.	Prototype à faible coût utilisant des composants accessibles et souvent issus de récupération.
Scalabilité	Architecture théoriquement très scalable, mais au prix d'une complexité et d'un coût importants.	Scalabilité modulaire possible par ajout de modules 3×3, tout en conservant une architecture relativement simple.

Table 19.2 – Comparaison technique entre le système inFORM et le prototype TACTERA

19.7.3 Analyse comparative

Le système *inFORM* représente une approche très avancée des *shape-displays* physiques. Grâce à une forte densité d'actionneurs, il est capable de reproduire des formes tridimensionnelles complexes avec une grande précision.

Le prototype *TACTERA* adopte quant à lui une approche plus pragmatique et accessible. Le choix d'une grille de 9×9 pixels et d'une architecture modulaire permet de réduire fortement la complexité mécanique et électronique du système, tout en conservant la possibilité de représenter des formes en relief perceptibles au toucher.

Ce choix implique toutefois un compromis important. La résolution de TACTERA est plus faible que celle d'inFORM, ce qui limite la finesse des formes représentées. En revanche, cette simplicité rend le dispositif plus facile à reproduire, à transporter et à utiliser dans des contextes pédagogiques ou de médiation scientifique.

19.8 Améliorations envisagées

Plusieurs axes d'amélioration sont envisagés pour l'évolution du prototype :

— **Logiciel** : intégration d'un système d'assistance vocale afin de rendre le contrôle de la grille plus accessible, et développement d'un module d'intelligence artificielle permettant la génération automatique de figures ou motifs à représenter physiquement.

— **Prototype physique** : modification de l'architecture électronique pour concentrer tous les composants dans la partie basse de chaque module, réduction du bruit produit par les mouvements, et optimisation générale de la structure pour faciliter la maintenance et l'évolutivité.

Ces améliorations visent à accroître la performance, l'accessibilité et la polyvalence du dispositif, tout en préparant le prototype à une utilisation dans des contextes pédagogiques ou artistiques plus larges.

Bibliographie

Bibliographie

[1] INSEE. *Le handicap visuel en France*. Institut National de la Statistique et des Études Économiques, 2023.

[2] R. C. Gonzalez, R. E. Woods. *Digital Image Processing*. Pearson Education, 4th edition, 2018.

[3] Arduino. *Arduino Documentation*. https ://docs.arduino.cc

[4] Numpy. *Numpy Documentation*. https ://numpy.org/doc/

Glossaire

Glossaire

actionneur Dispositif convertissant une énergie (souvent électrique) en mouvement mécanique. 26

bus de communication Système de transmission permettant à plusieurs composants électroniques d'échanger des données. 39

circuit RLC circuit composé d'une résistance, d'un condensateur et d'une inductance. 64

constante de temps Paramètre caractérisant la vitesse d'évolution d'un système dynamique vers son régime permanent. 24

driver moteur Circuit électronique chargé de piloter un moteur en contrôlant courant, direction et activation. 33

démultiplexeur Circuit logique permettant d'acheminer un signal d'entrée vers une sortie spécifique parmi plusieurs. 32

erreur de discrétisation Différence entre une surface continue réelle et son approximation par un ensemble discret de points. 26

filtrage Technique mathématique permettant de réduire les variations brusques ou le bruit dans un ensemble de données. 27, 28

flèche (déformation) Distance maximale entre la position initiale d'un élément structurel et sa position déformée sous l'effet d'une charge. 36

grille discrète Surface décrite par des points placés à intervalles réguliers. 26

I2C Protocole de communication série utilisant deux fils pour relier plusieurs circuits intégrés. 34, 40

inductance Propriété d'un circuit électrique qui s'oppose aux variations du courant et stocke de l'énergie dans un champ magnétique. 23, 24, 38

interpolation Méthode mathématique permettant d'estimer des valeurs intermédiaires entre plusieurs points connus. 51, 68

master sous-système centralisant les données et les renvoyant à chaque sous-système contrôlant les actionneurs. 32

matrice de hauteur Tableau de valeurs représentant les hauteurs des éléments d'une surface discrète. 26

moteur bipolaire Type de moteur pas à pas possédant deux bobines dont le sens du courant doit être inversé pour changer la direction. 23

multiplexage Technique permettant de transmettre plusieurs signaux différents sur un même canal de communication. 32

normalisation Transformation de valeurs numériques afin de les ramener dans un intervalle défini. 26, 51, 68

PCB circuits imprimés supportant et connectant les différents éléments électriques. 62

pont en H Circuit électronique permettant d'inverser le sens du courant dans une charge comme un moteur. 38, 39, 61

PWM Pulse Width Modulation : technique de modulation permettant de contrôler la puissance délivrée par un signal. 33

résolution spatiale Distance minimale séparant deux points indépendants pouvant être distingués dans un système. 27, 48

sujet sous-système contrôlant les actionneurs. 32

translation Mouvement rectiligne d'un objet sans rotation. 31, 35