



Institut National des Sciences Appliquées et de Technologie (INSAT)

Projet Personnel Professionnel

Spécialité : Instrumentation et Maintenance Industrielle

Niveau : 1^{ère} année cycle ingénieur

Conception Simulation et Réalisation d'un Bras Robotique Humanoïde

Réalisé par : Borhen Cherni

Encadré par : Mme Marwa Ben Chamekh

Membres du jury :
Mme Marwa Ben Chamekh – Encadrante
Mme Inès Mahouachi – Examineur

Date de soutenance : 9 juin 2025

Année Universitaire : 2024/2025

Remerciements

Nous tenons à exprimer notre sincère et profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet.

Avant tout, nous remercions chaleureusement notre encadrante, Mme Marwa Ben Chamekh, pour sa précieuse orientation, son accompagnement constant et ses retours constructifs qui ont grandement influencé la direction et l'aboutissement de ce travail.

Nous adressons également nos remerciements les plus sincères aux membres de l'équipe **Aerobotix**, dont l'apport généreux en matériel technique, ressources et conseils pratiques a joué un rôle déterminant dans le développement de notre bras robotique humanoïde. Leur disponibilité, leur expertise et leur esprit collaboratif ont été des atouts majeurs à chaque étape : conception, tests et résolution de problèmes.

Nous exprimons une reconnaissance particulière aux membres du jury d'évaluation du projet, qui ont consacré de leur temps pour examiner notre travail et nous faire bénéficier de leurs retours éclairés. Leurs observations nous ont permis d'améliorer et de mieux structurer nos solutions.

À toutes les personnes qui nous ont soutenus, de près ou de loin : vos contributions, grandes ou modestes, ont marqué ce projet, et nous vous en sommes profondément reconnaissants.

نُهدي هذا العمل بتواضع إلى أرواح الشهداء ، إلى كل من تمسك بالأمل في
أرض تُنتهك حقوقها يومياً و إلى ذلك الشعب البطل الذي يقَدِّم كل يوم دروساً في
الصمود، والإيمان، والتضحية . فلسطين ستظل منارة للكرامة والحق.
في الأخير ما نجاحاتنا إلا إهداء لروح المقاومة

Sommaire

I. Introduction Générale	1
1.1 Contexte Général	1
1.2 Applications Industrielles et Médicales	1
1.3 Motivation et Objectifs	2
1.4 Cadre Technologique	2
1.5 Défis et Perspectives	3
II. Conception Mécanique et Électronique	4
2.1 Vue d'ensemble mécanique	4
2.2 Base et structure du Bras	4
2.3 Structure du bras et mécanismes articulés	5
2.4 Mécanisme de préhension et des doigts	6
2.5 Stratégie d'assemblage et d'intégration mécanique	6
2.6 Adaptation à la tâche de construction	7
2.7 Vue d'ensemble électronique	7
2.8 Système d'alimentation	7
2.9 Architecture de commande	8
2.10 Cartes électroniques personnalisées	8
2.11 Pilotage des actionneurs	10
2.12 Conclusion	11
III. Simulation et Cinématique du Bras Humanoïde	12
4.1 Introduction à la Simulation	12
4.2 Exportation depuis SolidWorks	12
4.3 Génération du fichier XML	13
4.4 Importation du modèle dans MATLAB	14
4.5 Résultats de la simulation	14
4.6 Introduction à la Cinématique Inverse	15
4.7 Hypothèses et Configuration du Bras	15
4.8 Modélisation Géométrique	16
4.9 Simulation 2D avec Pygame	17
4.10 Visualisation et Interprétation	18
4.11 Conclusion	18
IV. Architecture de Contrôle du Bras Humanoïde	19
6.1 Introduction	19
6.2 Vue Générale de l'Architecture	19

6.3 Fonctionnement des Modes de Commande	20
6.4 Rôle de l'ESP32 dans l'Architecture	20
6.5 Schéma Fonctionnel de l'Architecture	21
6.6 Avantages de l'Architecture Proposée	21
6.7 Conclusion	22
V. Contrôle en Temps Réel via WebSocket	23
7.1 Introduction	23
7.2 Présentation du Protocole WebSocket	23
7.3 Architecture Générale de Communication	23
7.4 Interface Web : Page d'Accueil	24
7.5 Mode 1 : Commande Manuelle par Interface Web	25
7.6 Mode 2 : Commande par Vision (ml5.js)	26
7.7 La commande envoyée	27
7.8 Interprétation côté ESP32	28
7.9 Conclusion	30
VI. Conclusion Générale	31

Liste des figures

1	Exemple d'Applications Industrielles et Médicales	2
2	Modèles CAO SolidWorks de la base et de la rotation autour de l'axe Z	5
3	Modèle SolidWorks montrant les emplacements des servomo- teurs MG995 et SG90	6
4	Modèle SolidWorks montrant le mouvement des doigts avec les servomoteurs SG90	6
5	Pinout pour ESP32 NODEMCU-32S [1]	8
6	Shématique de la Carte électronique	9
7	PCB et routage de la carte	10
8	Vue 3D de la carte PCB illustrant le routage des signaux de commande	11
9	Intégration du modèle dans MATLAB	12
10	Modèle CAO du bras humanoïde sous SolidWorks	13
11	Génération des fichiers XML	13
12	Importation vers simulink	14
13	Génération de la bras sur Simulink MATLAB	14
14	Code de la cinématique inverse en Python	16
15	Simulation 2D du mouvement du Bras	17
16	Génération de la bras sur Simulink MATLAB	21
17	Architecture de communication WebSocket	24
18	Page d'accueil	25
19	Page de la Commande Manuelle	26
20	Page de la Commande Manuelle	26
21	Code pour la détection du mouvement du bras	27
22	Exemples de la commande envoyée au ESP32	28
23	Partie du code de la communication et de la connexion au serveur	29
24	Réponse après la connexion	30

I. Introduction Générale

1.1 Contexte Général

La robotique est une discipline interdisciplinaire qui s'est imposée comme un pilier fondamental des technologies modernes. Depuis les premiers automates jusqu'aux robots intelligents actuels, les systèmes robotiques n'ont cessé d'évoluer, combinant mécanique, intelligence artificielle et perception environnementale. Parmi ces systèmes, les bras robotiques humanoïdes représentent un domaine en pleine expansion, cherchant à reproduire les mouvements complexes et la polyvalence du bras humain.

1.2 Applications Industrielles et Médicales

Les bras robotisés humanoïdes trouvent déjà des applications concrètes dans divers domaines, notamment l'industrie, la médecine et la recherche. Dans le secteur industriel, ils sont largement utilisés pour l'automatisation de tâches nécessitant une grande précision, telles que la soudure, l'assemblage de composants ou encore la peinture de surfaces complexes. Leur capacité à répéter des gestes avec exactitude et fiabilité en fait des outils essentiels pour améliorer la productivité et la qualité.

Dans le domaine médical, ces bras sont intégrés dans des systèmes de chirurgie assistée, permettant aux praticiens d'effectuer des interventions complexes avec une précision accrue. Ils sont également utilisés dans les programmes de rééducation fonctionnelle, où ils peuvent accompagner les patients dans leurs exercices, ainsi que dans la conception de prothèses intelligentes capables d'imiter les mouvements naturels du corps humain.

Enfin, dans le cadre de la recherche scientifique, les bras humanoïdes servent de plateformes expérimentales pour le développement et le test d'algorithmes de contrôle avancés et d'intelligence artificielle. Leur utilisation dans ce contexte permet de faire progresser les connaissances en robotique tout en explorant de nouvelles interactions entre l'homme et la machine.



Figure 1: Exemple d'Applications Industrielles et Médicales

1.3 Motivation et Objectifs

La capacité à manipuler des objets avec précision, souplesse et adaptabilité est cruciale dans de nombreux domaines. Le développement d'un bras humanoïde vise à répondre à des besoins concrets dans l'industrie, la médecine, ou encore la recherche. Ce projet a pour objectif :

- La conception et la réalisation d'un bras robotique inspiré de l'anatomie humaine.
- L'implémentation de plusieurs modes de commande (web, IA, MQTT).
- L'intégration de composants mécaniques, électroniques et logiciels.
- L'optimisation de l'ergonomie et de la performance du système.

1.4 Cadre Technologique

Le développement du bras repose sur l'utilisation de technologies avancées :

- **Conception mécaniques** : modélisation mécanique sur SolidWorks.
- **Conception** : schémas électroniques et modélisation de cartes électronique (PCB) sur EasyEDA .
- **Commande** : interfaces web, protocoles de communication (MQTT), algorithmes IA.
- **Simulation** : environnement URDF pour tests virtuels.

1.5 Défis et Perspectives

Le développement de ce bras robotique humanoïde présente plusieurs défis techniques et fonctionnels. L'un des principaux obstacles réside dans la coordination simultanée de plusieurs moteurs, nécessaire pour assurer des mouvements fluides et réalistes. De plus, la fiabilité des composants mécaniques et électroniques est essentielle, notamment lorsqu'ils sont soumis à des cycles répétés de fonctionnement, ce qui peut engendrer une usure prématurée ou des défaillances. Un autre défi concerne la synchronisation entre les différents modes de commande du bras et la réponse effective du système, qui doit être à la fois rapide et précise. Enfin, le réalisme du comportement du bras robotisé, comparé à celui d'un bras humain, demeure un objectif ambitieux, notamment en termes de gestuelle et de souplesse de mouvement.

Ces défis, bien qu'importants, constituent également des opportunités pour des perspectives d'amélioration. Des pistes intéressantes peuvent être envisagées, telles que l'intégration d'un retour haptique permettant une interaction plus naturelle, l'exploitation de techniques d'apprentissage par renforcement pour affiner les mouvements de manière autonome, ou encore l'ajout de capteurs avancés afin d'améliorer la perception de l'environnement et d'enrichir l'interaction homme-machine.

II. Conception Mécanique et Électronique

2.1 Vue d'ensemble mécanique

La structure mécanique de notre bras robotique humanoïde a été conçue et réalisée sur SolidWorks, en l'adaptant spécifiquement aux exigences multifonctionnelles du projet. De la manipulation précise d'objets à la rotation fluide et à l'actionnement coordonné, chaque composant a été conçu de manière itérative et validé par simulation. Les assemblages modulaires permettent des améliorations ciblées et une intégration fluide avec les systèmes électroniques et de commande par intelligence artificielle.

2.2 Base et structure du Bras

Le bras robotique est monté sur un châssis rigide assurant une stabilité mécanique tout en permettant une rotation horizontale à 360° grâce à un moteur pas-à-pas. Caractéristiques principales :

- **Base** : Structure rigide en PLA avec supports anti-vibrations en MDF.
- **Rotation** : Moteur pas-à-pas NEMA17(17HS2408) assurant la rotation autour de l'axe Z.
- **Modularité** : Toutes les pièces mécaniques ont été conçues sur SolidWorks et imprimées en 3D pour une itération rapide.

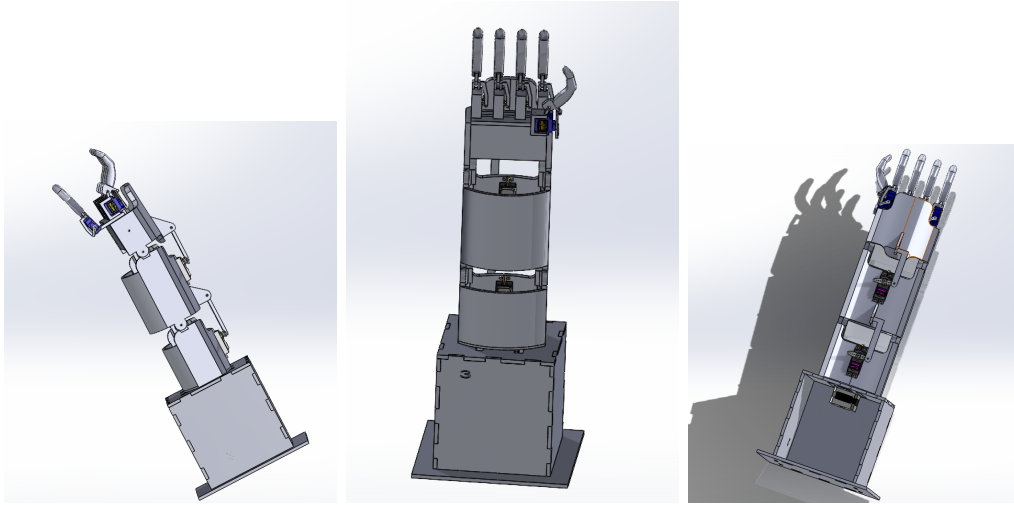


Figure 2: Modèles CAO SolidWorks de la base et de la rotation autour de l'axe Z

2.3 Structure du bras et mécanismes articulés

Le bras humanoïde se compose de segments articulés commandés par différents moteurs sélectionnés en fonction des exigences mécaniques. Les articulations majeures, comme l'épaule et le coude, sont actionnées par des servomoteurs MG995, reconnus pour leur puissance et leur capacité à supporter des charges importantes tout en assurant des mouvements fluides. Le poignet est équipé de micro-servomoteurs SG90, plus compacts, offrant un positionnement précis indispensable pour les mouvements fins. Quant à la base, elle est animée par un moteur pas-à-pas assurant la rotation autour de l'axe Z, ce qui permet une orientation stable du bras dans l'espace.

Toutes les articulations ont été simulées dans SolidWorks pour anticiper les efforts mécaniques subis pendant le fonctionnement. Ces simulations ont permis de vérifier l'absence d'interférences entre composants, de valider les plages de mouvement, et d'assurer la robustesse de la structure.

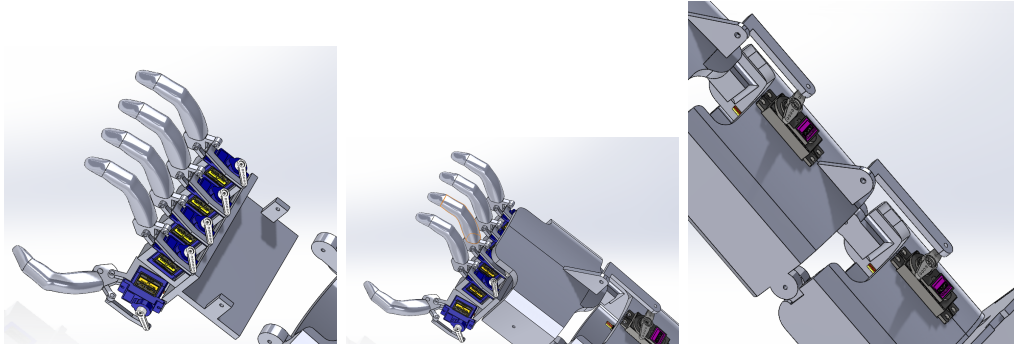


Figure 3: Modèle SolidWorks montrant les emplacements des servomoteurs MG995 et SG90

2.4 Mécanisme de préhension et des doigts

Chaque doigt du bras humanoïde est commandé de manière indépendante par un micro-servomoteur SG90, ce qui permet un contrôle précis lors des actions de saisie. Cette architecture confère au système une grande adaptabilité, notamment pour des tâches nécessitant une prise minutieuse, comme l'empilement de blocs. Chaque doigt possède un degré de liberté, suffisant pour accomplir les fonctions de préhension visées.

Les doigts ont été réalisés en PLA, un matériau léger et résistant, imprimé avec un taux de remplissage de 20 %. Ce choix offre un bon compromis entre flexibilité et solidité, garantissant la durabilité des pièces en conditions réelles d'utilisation. La stratégie de préhension adoptée privilégie la précision et l'ajustement dynamique à la forme des objets, permettant une manipulation efficace d'éléments de tailles variées.

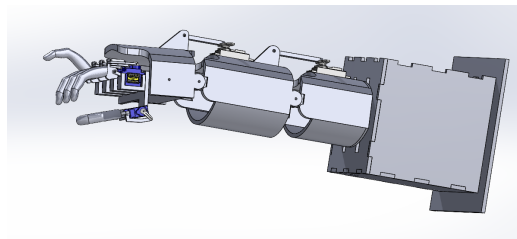


Figure 4: Modèle SolidWorks montrant le mouvement des doigts avec les servomoteurs SG90

2.5 Stratégie d'assemblage et d'intégration mécanique

La conception mécanique du bras repose sur une approche modulaire, facilitant l'assemblage et l'intégration des éléments électroniques. Chaque pièce

structurale, imprimée en PLA, intègre des canaux internes conçus pour le passage des câbles, assurant ainsi une organisation propre et réduisant les risques d'enchevêtrement ou de dégradation. Les servomoteurs sont fixés dans les articulations à l'aide d'un système d'emboîtement ajusté, renforcé par des inserts thermiques qui garantissent la stabilité mécanique à long terme.

Le tout est relié à une carte de contrôle montée sur un circuit imprimé personnalisé, assurant une interconnexion fiable entre le système mécanique et les modules électroniques. Cette stratégie permet une maintenance simplifiée et une évolutivité du système pour des itérations futures.

2.6 Adaptation à la tâche de construction

L'architecture du bras humanoïde a été spécifiquement pensée pour répondre aux exigences de la tâche de construction de tribunes lors du match. La saisie des blocs est assurée par les doigts, chacun contrôlé indépendamment via un servomoteur SG90, permettant de s'adapter à la forme et à la taille de chaque bloc. Une fois saisi, le bloc est orienté correctement grâce à la rotation sur l'axe Z, commandée par un moteur pas-à-pas situé à la base.

L'élévation du bras pour atteindre les différents niveaux est rendue possible grâce à la coordination des servomoteurs MG995 au niveau de l'épaule et du coude. Cette coordination assure la stabilité de l'ensemble lors du mouvement vertical. Enfin, une fois la position souhaitée atteinte, le mécanisme de préhension desserre progressivement sa prise, permettant un relâchement contrôlé du bloc sans perturber l'équilibre de la structure déjà en place.

2.7 Vue d'ensemble électronique

Le système électronique a été conçu à l'aide d'outils professionnels de conception assistée par ordinateur (CAO), assurant ainsi un haut niveau de fiabilité, de précision et de reproductibilité. Plus précisément, le logiciel EasyEDA a été utilisé pour l'élaboration des schémas électroniques ainsi que pour le routage du circuit imprimé (PCB). [2]

2.8 Système d'alimentation

Le système est alimenté par une batterie LiPo de 12V. Deux modules abaisseurs de tension (XL4016 PWM) sont utilisés pour générer les tensions nécessaires :

- Une sortie réglée à 12V, utilisée pour alimenter le moteur pas à pas.

Cette structure garantit une distribution d'énergie stable et indépendante entre les éléments de puissance et les éléments logiques.

2.9 Architecture de commande

Le cœur du système est un microcontrôleur ESP32, qui joue le rôle de centre de traitement principal. Il est chargé de :

- Gérer les entrées/sorties numériques,
- Piloter les moteurs et servomoteurs,
- Assurer la communication sans fil via Wi-Fi/MQTT,
- Exécuter la logique embarquée (IA, interface web, etc.).

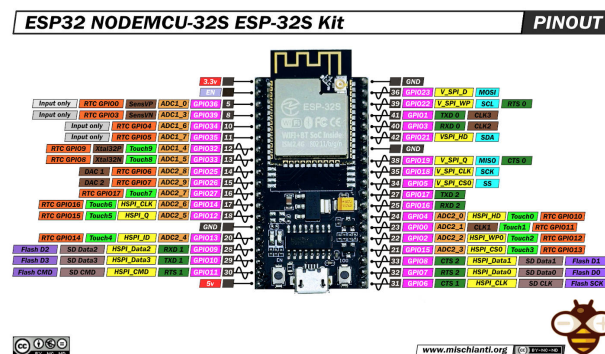


Figure 5: Pinout pour ESP32 NODEMCU-32S [1]

L'ESP32 est protégé électriquement par sept optocoupleurs, qui assurent l'isolation galvanique entre la partie logique et la partie puissance.

Le moteur pas-à-pas (axe Z du bras robotique) est contrôlé via un driver A4988, permettant une gestion précise des micropas.

2.10 Cartes électroniques personnalisées

Nous avons conçu une carte électronique personnalisée afin de centraliser les connexions entre l'unité de commande, les actionneurs et les modules d'alimentation, tout en garantissant une isolation électrique adéquate. La

carte intègre des connecteurs spécifiquement adaptés au microcontrôleur ESP32, facilitant ainsi son intégration physique et fonctionnelle. Pour assurer la protection des circuits logiques contre les perturbations électromagnétiques ou les retours de courant indésirables, sept optocoupleurs de type 4N25 ont été utilisés, permettant une isolation galvanique efficace des signaux de commande.

Les sorties de commande sont connectées à des servomoteurs de type SG90 et MG995 via des connecteurs standard, assurant un câblage propre et fiable. Le pilotage du moteur pas-à-pas est assuré par un driver A4988, reconnu pour sa capacité à fournir un courant allant jusqu'à 2 A avec un système de micro-pas intégré, offrant un contrôle précis de la position. En ce qui concerne l'alimentation, deux modules abaisseurs de tension (buck converters) XL4016 ont été intégrés afin de réguler les tensions d'alimentation des différents sous-systèmes, chacun capable de fournir une intensité maximale de 8 A avec une tension de sortie réglable. Cette architecture matérielle permet une distribution d'énergie stable et sécurisée à l'ensemble du système.

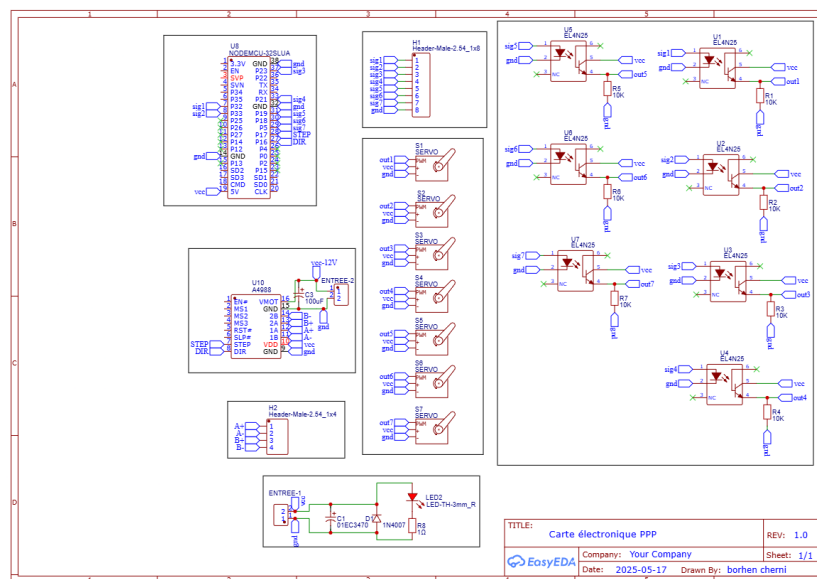


Figure 6: Shématique de la Carte électronique

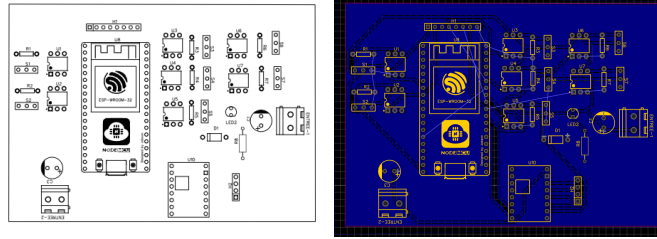


Figure 7: PCB et routage de la carte

2.11 Pilotage des actionneurs

Le pilotage des actionneurs constitue un volet essentiel dans le fonctionnement du bras robotique humanoïde. Il repose sur une gestion rigoureuse des signaux de commande, assurée par le microcontrôleur ESP32, et s'articule autour de trois familles de moteurs : les servomoteurs SG90 et MG995, ainsi que le moteur pas-à-pas dédié à la rotation de la base.

Les servomoteurs SG90, utilisés pour le contrôle des doigts, et les MG995, affectés aux articulations majeures comme l'épaule et le coude, sont commandés par des signaux PWM générés directement par l'ESP32. Afin d'assurer une isolation électrique entre la carte de contrôle et les actionneurs, chaque signal de commande traverse un optocoupleur. Cette précaution permet de protéger le microcontrôleur contre les retours de courant indésirables, tout en garantissant une transmission fiable des impulsions de commande.

Le moteur pas-à-pas, responsable de la rotation de la base autour de l'axe Z, est piloté à l'aide d'un driver A4988. Celui-ci reçoit, depuis l'ESP32, deux types de signaux distincts : l'un pour la direction, l'autre pour l'avancement angulaire sous forme d'impulsions. Ce type de commande offre une excellente précision dans le positionnement, ce qui est particulièrement utile dans des tâches nécessitant un alignement rigoureux de la base du bras.

D'un point de vue électrique, plusieurs mesures ont été mises en œuvre pour garantir la fiabilité et la sécurité du système. Chaque ligne de commande PWM a été isolée à l'aide d'un optocoupleur afin d'éviter les perturbations électromagnétiques et les court-circuits potentiels. Des condensateurs de découplage ont également été placés à proximité des moteurs pour atténuer les fluctuations de tension susceptibles d'affecter la stabilité du système. De plus, les masses des circuits de puissance et de logique ont été volontairement séparées pour éviter toute interférence entre l'alimentation des moteurs et celle du microcontrôleur.

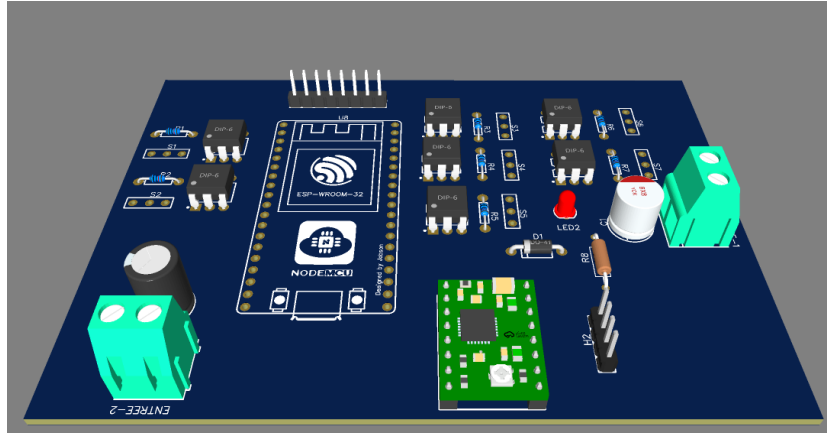


Figure 8: Vue 3D de la carte PCB illustrant le routage des signaux de commande

2.12 Conclusion

La conception intégrée mécanique et électronique assure une robustesse et une modularité optimales. Les simulations CAO et les conceptions PCB valident la faisabilité du système. Le chapitre suivant présentera les aspects logiciels et d'intelligence artificielle.

III. Simulation et Cinématique du Bras Humanoïde

4.1 Introduction à la Simulation

Avant d'implémenter la commande sur le système physique, une phase de simulation est essentielle pour analyser les mouvements, détecter d'éventuelles collisions, et valider la configuration cinématique. Le modèle mécanique conçu sous SolidWorks a donc été converti pour être exploité dans des environnements de simulation tels que **MATLAB/Simulink** ou **ROS**. Pour les aspects mécaniques et dynamiques, la bibliothèque **Simscape Multibody** permet une modélisation réaliste des composants physiques. [3, 4] [5]



Figure 9: Intégration du modèle dans MATLAB

Cette étape passe par l'exportation du modèle au format **XML** via l'outil **Simscape Multibody Link**, puis la conversion vers le format **URDF (Unified Robot Description Format)**.

4.2 Exportation depuis SolidWorks

Le bras humanoïde a été modélisé sous SolidWorks, intégrant toutes les articulations, servomoteurs, et degrés de liberté.

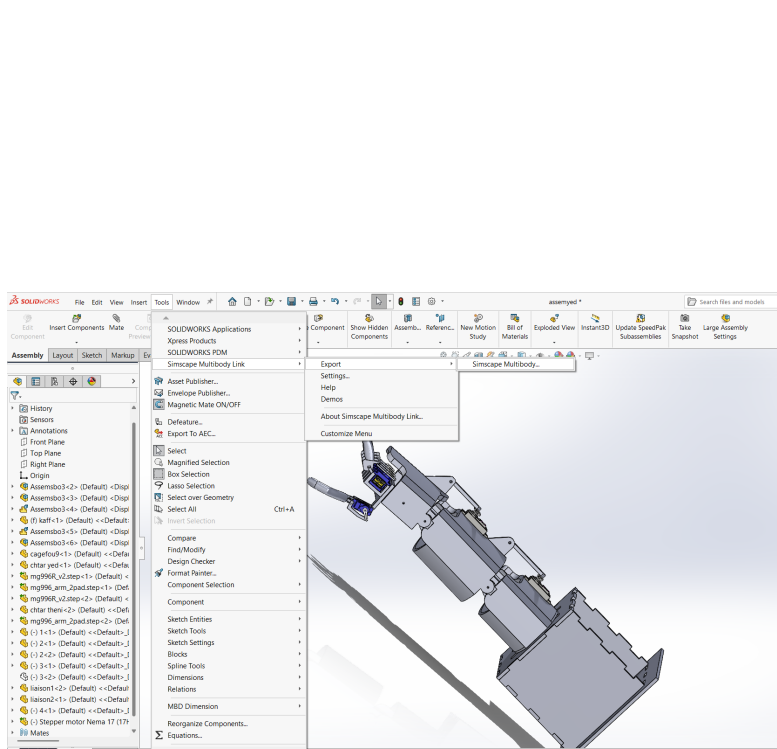


Figure 10: Modèle CAO du bras humanoïde sous SolidWorks

Une fois la modélisation terminée, le plugin Simscape Multibody Link a été utilisé pour exporter le modèle.

4.3 Génération du fichier XML

L'exportation génère :

- Un fichier `robot.xml` décrivant la structure du bras (joints, liens, coordonnées relatives, etc.)
- Un dossier contenant les fichiers `.stl` représentant les composants 3D du bras



Figure 11: Génération des fichiers XML

Ces fichiers sont ensuite importés dans MATLAB ou Simulink pour la simulation de la dynamique du bras via Simscape Multibody.

4.4 Importation du modèle dans MATLAB

1. Exporter le modèle SolidWorks au format `.xml` à l'aide de l'extension Simscape Multibody Link.
2. Importer ce fichier dans MATLAB à l'aide de la commande `smimport('nom du fichier.xml')`, ce qui permet de générer automatiquement un modèle Simscape contenant les corps rigides, les articulations, et les contraintes mécaniques.

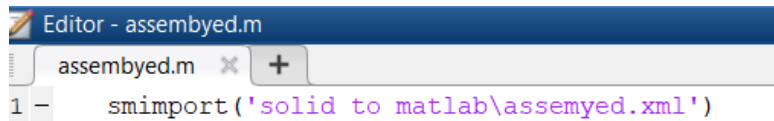


Figure 12: Importation vers simulink

3. Visualiser et affiner les paramètres de simulation tels que la masse, l'inertie, les frottements ou encore les limites angulaires directement dans l'environnement Simulink.

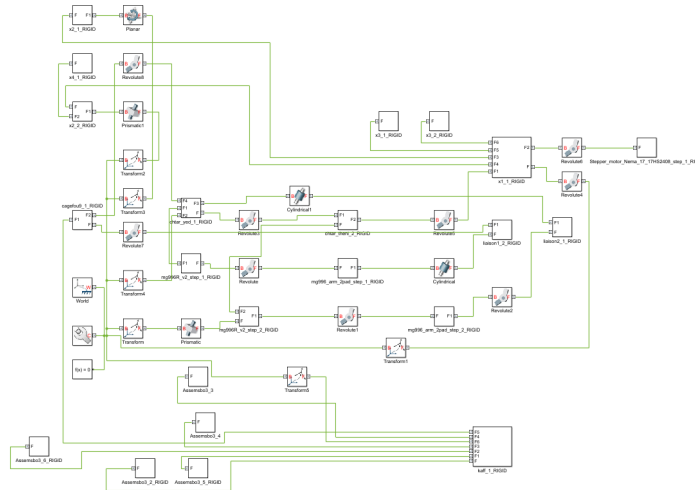


Figure 13: Génération de la bras sur Simulink MATLAB

4.5 Résultats de la simulation

Une fois le modèle du bras humanoïde entièrement intégré dans l'environnement MATLAB/Simulink, une première simulation de mouvement a été effectuée. L'objectif de cette étape était de valider le comportement cinématique et dynamique du système avant toute expérimentation

physique. Pour ce faire, chaque articulation a été soumise à un signal d'entrée simulé, reproduisant le fonctionnement d'un signal PWM réel. Les vitesses angulaires générées en réponse à ces signaux ont été enregistrées afin d'évaluer la réactivité du système.

Les résultats de cette simulation ont permis de confirmer la liberté de mouvement de chaque articulation, conformément aux degrés de liberté définis dans la conception. Aucun blocage ni interférence mécanique n'a été détecté au cours des séquences de mouvement simulées, ce qui témoigne de la cohérence du modèle mécanique et de l'efficacité de l'assemblage. Par ailleurs, la réponse dynamique du bras a été jugée satisfaisante : les mouvements simulés respectent les consignes imposées, avec des temps de réponse et des amplitudes compatibles avec les exigences de la tâche de manipulation envisagée.

Cette simulation constitue ainsi une validation préliminaire essentielle, garantissant que le système est mécaniquement opérationnel et prêt à être commandé dans un environnement réel sans risques majeurs d'instabilité ou de collisions internes.

4.6 Introduction à la Cinématique Inverse

La cinématique inverse constitue une étape fondamentale dans le contrôle des bras robotiques, permettant de déterminer les angles des articulations à partir de la position souhaitée de l'effecteur. Dans ce chapitre, nous décrivons l'approche géométrique adoptée pour notre bras humanoïde, et nous présentons une simulation 2D illustrant le fonctionnement du modèle.

4.7 Hypothèses et Configuration du Bras

Le bras robotique modélisé comprend :

- Un moteur pas-à-pas assurant la rotation selon l'axe Z (non traité dans l'analyse planaire XY).
- Deux servomoteurs MG995 assurant les mouvements dans le plan 2D (plan XY) : l'épaule (angle θ_1) et le coude (angle θ_2).
- Les longueurs des segments : $L_1 = 10$ cm (entre l'épaule et le coude), et $L_2 = 12$ cm (entre le coude et l'effecteur).

Les servomoteurs des doigts sont négligés dans cette analyse.

4.8 Modélisation Géométrique

Le but est de déterminer θ_1 et θ_2 à partir d'une position cible (x, y) de l'effecteur.

4.8.1 Distance totale

$$D = \sqrt{x^2 + y^2} \quad (1)$$

Si $D > L_1 + L_2$, on ramène la cible au bord du cercle d'accessibilité du bras.

4.8.2 Angle du coude (θ_2)

$$\cos(\theta_2) = \frac{x^2 + y^2 - L_1^2 - L_2^2}{2L_1L_2} \quad (2)$$

$$\theta_2 = \arccos(\cos(\theta_2)) \quad (3)$$

4.8.3 Angle de l'épaule (θ_1)

$$k_1 = L_1 + L_2 \cos(\theta_2)$$

$$k_2 = L_2 \sin(\theta_2)$$

$$\theta_1 = \arctan 2(y, x) - \arctan 2(k_2, k_1)$$

```
def inverse_kinematics(x, y):
    dx = x - origin[0]
    dy = y - origin[1]
    D = math.hypot(dx, dy)

    if D > L1 + L2:
        dx *= (L1 + L2) / D
        dy *= (L1 + L2) / D
        D = L1 + L2

    cos_theta2 = (dx**2 + dy**2 - L1**2 - L2**2) / (2 * L1 * L2)
    theta2 = math.acos(max(-1, min(1, cos_theta2)))
    k1 = L1 + L2 * math.cos(theta2)
    k2 = L2 * math.sin(theta2)
    theta1 = math.atan2(dy, dx) - math.atan2(k2, k1)

    return theta1, theta2
```

Figure 14: Code de la cinématique inverse en Python

4.9 Simulation 2D avec Pygame

Afin de valider visuellement les résultats des calculs d'inverse kinematics, une simulation bidimensionnelle a été développée en langage Python à l'aide de la bibliothèque `pygame`. Cette interface graphique permet une visualisation en temps réel du mouvement du bras artificiel dans un environnement simplifié.[6]

La scène de simulation intègre une grille de fond avec les axes x et y numérotés, facilitant le repérage spatial. L'utilisateur peut déplacer un curseur à l'aide de la souris, ce qui définit dynamiquement la position cible à atteindre. Les coordonnées correspondantes s'affichent à l'écran. En réponse à cette cible, les angles θ_1 et θ_2 nécessaires au positionnement du bras sont automatiquement calculés et également affichés. Le bras artificiel est représenté graphiquement par deux segments articulés, connectés entre eux au niveau du coude, reproduisant fidèlement la structure cinématique du système réel.

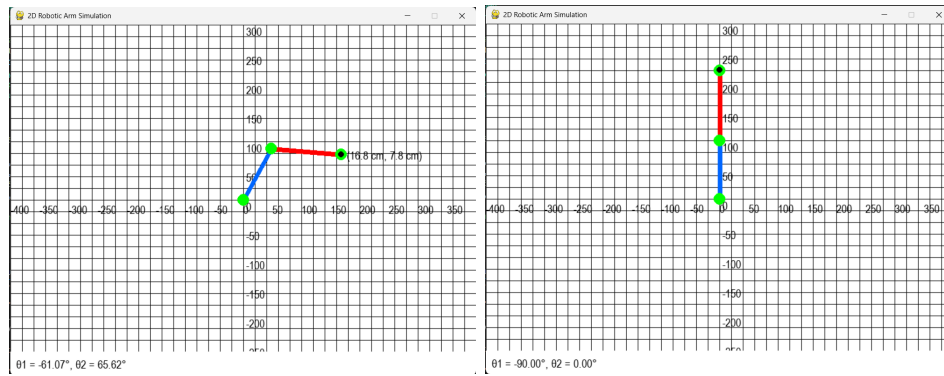


Figure 15: Simulation 2D du mouvement du Bras

La figure 15 illustre la simulation 2D du mouvement du bras humanoïde à deux articulations. Le bras est représenté par deux segments rigides articulés, modélisant respectivement l'avant-bras et le bras supérieur. Le point d'origine représente l'épaule, tandis que l'extrémité libre du second segment simule la pince ou le bout des doigts.

Cette représentation dynamique permet de visualiser comment le bras s'oriente pour atteindre une position cible définie par l'utilisateur à l'aide de la souris. Les coordonnées de la cible sont visibles dans l'interface, et les angles articulaires nécessaires pour atteindre cette position sont automatiquement calculés via un modèle d'inverse kinematics, puis utilisés pour afficher la nouvelle configuration du bras.

La simulation offre une perspective claire sur le comportement du bras dans un espace cartésien, permettant de confirmer la validité des angles cal-

culés. On observe que, quel que soit le point cible choisi dans la zone atteignable, le bras parvient à adopter une configuration réaliste, sans collision ni distorsion visuelle. L'interface permet également de détecter visuellement les zones inaccessibles par le bras, notamment les régions en dehors de son rayon d'action maximal.

Ce type de visualisation constitue donc un outil essentiel pour tester l'efficacité des algorithmes de commande avant leur implémentation sur le système réel. Il facilite également la détection d'éventuelles incohérences dans les limites angulaires ou les équations de transformation directe et inverse.

4.10 Visualisation et Interprétation

La simulation permet de visualiser avec précision les angles calculés par le modèle, qui correspondent aux commandes envoyées aux servomoteurs dans l'application réelle. La présence de la grille et des axes facilite grandement le débogage et la vérification manuelle des résultats. Elle offre également un environnement expérimental rapide pour tester différentes configurations de positionnement, sans avoir à interagir physiquement avec le système réel. Cette étape représente ainsi un outil de validation intermédiaire essentiel dans le processus de développement du bras humanoïde.

4.11 Conclusion

Cette analyse combinée des simulations 3D sous MATLAB et 2D avec Pygame a permis de valider intégralement le modèle mécanique du bras humanoïde tout en vérifiant la robustesse des algorithmes de contrôle développés. Les résultats obtenus constituent une base solide pour l'implémentation physique du système, ayant démontré la cohérence entre les modèles théoriques et le comportement simulé. Par ailleurs, cette phase intensive de simulation a mis en évidence plusieurs axes d'amélioration, notamment en termes d'optimisation des trajectoires et de gestion des singularités, qui feront l'objet des développements ultérieurs. Ces travaux préliminaires assurent ainsi une transition fiable vers la phase expérimentale tout en minimisant les risques techniques.

IV. Architecture de Contrôle du Bras Humanoïde

6.1 Introduction

L'architecture de contrôle constitue le noyau central du fonctionnement du bras humanoïde robotique. Elle garantit l'intégration harmonieuse entre le matériel embarqué (ESP32, servomoteurs, drivers, etc.) et les logiciels de commande (interfaces web, vision par ordinateur, et modules de communication). Ce chapitre décrit l'architecture adoptée, les différents modules de commande, les protocoles de communication utilisés, ainsi que leur rôle dans le système global.

6.2 Vue Générale de l'Architecture

L'architecture repose sur une communication multiprotocole basée sur :

- **WebSocket** pour la communication en temps réel entre l'interface utilisateur Web et le serveur local
- **MQTT** pour la communication entre un client distant (ex. : interface QtPython) et le microcontrôleur ESP32
- **ESP32** comme unité centrale de traitement et d'exécution des commandes vers les servomoteurs du bras.

Trois modes de commande sont intégrés dans le système :

1. **Commande Web** : L'utilisateur spécifie la position cible du bras via une interface graphique simple, accessible depuis un navigateur.
2. **Commande par Vision** : Une caméra connectée au PC détecte des gestes de la main, analysés en temps réel par un modèle d'intelligence artificielle qui génère des commandes vers le bras.
3. **Commande MQTT (QtPython)** : Un programme développé en QtPython permet de publier des commandes sur un topic MQTT, auxquelles l'ESP32, agissant comme `subscriber`, répond.

6.3 Fonctionnement des Modes de Commande

6.3.1 Interface Web

L'interface Web, construite avec HTML/CSS/JavaScript, permet à l'utilisateur de saisir des coordonnées cibles ou de cliquer sur des boutons de positionnement. Les données sont transmises au serveur via WebSocket, puis envoyées en temps réel à l'ESP32 pour exécution.

6.3.2 Commande par Vision

Ce mode utilise la caméra du PC pour capter l'image de la main de l'utilisateur. Un modèle d'apprentissage profond (par exemple basé sur **MediaPipe** ou **OpenCV**) traite l'image et détecte les gestes. Selon le geste reconnu, une commande appropriée est générée et transmise à l'ESP32 via WebSocket.

6.3.3 Commande via MQTT (QtPython)

Dans ce mode, une interface développée en QtPython permet de simuler des mouvements ou d'envoyer des consignes précises. L'interface publie les messages sur un **topic** MQTT. L'ESP32, en tant que **subscriber**, écoute ce topic et exécute les commandes reçues. Cette approche permet un contrôle distant du bras via le réseau local ou Internet.

6.4 Rôle de l'ESP32 dans l'Architecture

Le microcontrôleur **ESP32** constitue l'unité de traitement embarquée. Il est chargé de :

- Recevoir les commandes depuis les différentes sources (WebSocket, MQTT)
- Interpréter le message reçu (position, vitesse, type de mouvement)
- Contrôler les servomoteurs via PWM et le moteur pas-à-pas via le driver A4988 ;
- Gérer la sécurité (limites angulaires, alimentation, interruptions).

L'ESP32 communique avec les différents composants électroniques, notamment les optocoupleurs, les convertisseurs buck et les drivers moteurs.

6.5 Schéma Fonctionnel de l'Architecture

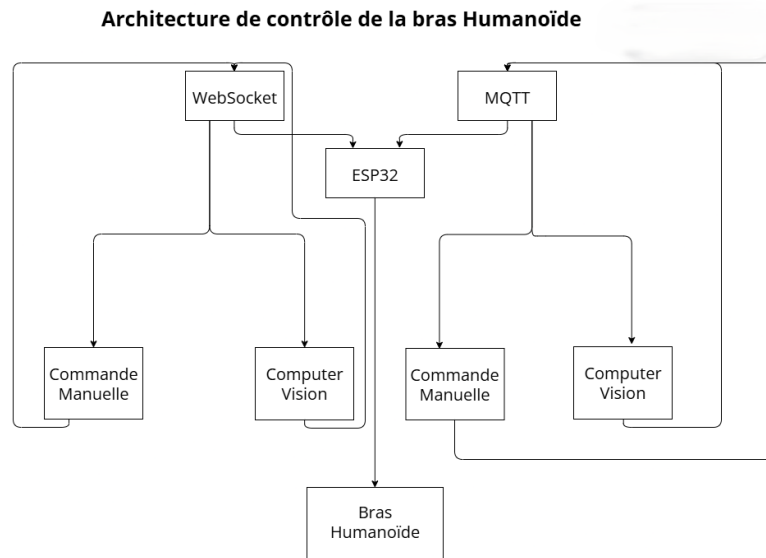


Figure 16: Génération de la bras sur Simulink MATLAB

Ce schéma illustre les interactions entre les trois interfaces de commande, les protocoles de communication, le module ESP32, et les actionneurs du bras robotique.

6.6 Avantages de l'Architecture Proposée

L'architecture développée dans ce projet se distingue par sa modularité, qui permet à chaque mode de commande (manuel, vision par ordinateur, MQTT, etc.) d'être conçu, modifié ou amélioré de manière indépendante sans affecter les autres composants du système. Cette approche modulaire facilite la maintenance, la réutilisabilité du code et l'intégration progressive de nouvelles fonctionnalités.

Un autre avantage majeur réside dans la flexibilité de l'architecture. En effet, l'utilisateur final peut librement basculer entre les différents modes de commande en fonction du contexte opérationnel ou des préférences d'utilisation. Cette adaptabilité rend le système pertinent dans une grande variété d'environnements et de scénarios, que ce soit en laboratoire, en compétition ou en démonstration.

L'architecture est également évolutive. Elle permet d'envisager l'ajout futur de nouveaux modes de commande, tels que la commande vocale ou

l'intégration d'une intelligence artificielle embarquée, sans nécessiter une refonte globale du système existant. Cette capacité d'extension est essentielle pour garantir la pérennité de la plateforme.

Sur le plan des performances, la communication entre le client et le bras via WebSocket assure une faible latence, ce qui est crucial pour les applications interactives ou en temps réel. Enfin, l'usage du protocole MQTT garantit une interopérabilité efficace avec des systèmes distants ou des services cloud, ouvrant la voie à une intégration aisée dans des infrastructures plus larges, notamment dans des environnements domotiques ou industriels.

6.7 Conclusion

L'architecture de contrôle du bras humanoïde repose sur une conception distribuée, robuste et réactive, intégrant intelligemment les nouvelles technologies de communication et de traitement. Elle assure une flexibilité maximale et prépare le système à des évolutions futures, telles que l'intégration de capteurs, l'autonomie complète par intelligence artificielle, ou le contrôle vocal.

V. Contrôle en Temps Réel via WebSocket

7.1 Introduction

Pour contrôler un bras robotique humanoïde de manière fluide et réactive, il est essentiel de disposer d'une communication bidirectionnelle rapide entre l'utilisateur et la carte ESP32. Le protocole **WebSocket** est parfaitement adapté à ce besoin, car il permet des échanges continus de données entre un navigateur web et une plateforme embarquée [7].

Dans ce projet, nous avons développé deux modes de commande :

1. **Commande manuelle** : l'utilisateur saisit ou sélectionne des positions cibles en X, Y et Z via une interface graphique web ;
2. **Commande par vision** : les gestes de la main de l'utilisateur sont capturés via la webcam et interprétés par un modèle IA basé sur **ml5.js** [8].

Dans les deux cas, les données sont transmises en temps réel à l'ESP32 via WebSocket.

7.2 Présentation du Protocole WebSocket

WebSocket est un protocole défini par la **RFC 6455** qui établit une connexion persistante entre le client et le serveur, permettant l'échange de données en temps réel dans les deux sens. Contrairement au protocole HTTP, il évite les délais de requêtes successives et s'avère idéal pour les applications interactives et robotiques [7].

7.3 Architecture Générale de Communication

L'architecture générale de la communication repose sur le protocole WebSocket, qui permet l'établissement d'une connexion bidirectionnelle entre le client et le serveur. Contrairement aux protocoles traditionnels basés sur des requêtes, WebSocket offre un canal de communication en temps réel, réduisant la latence et améliorant la réactivité du système. Dans notre application, le client (interface web) initie la connexion avec le microcontrôleur ESP32, jouant le rôle de serveur WebSocket. Une fois la connexion établie,

les données peuvent être échangées de manière asynchrone, permettant un contrôle fluide et interactif du bras .

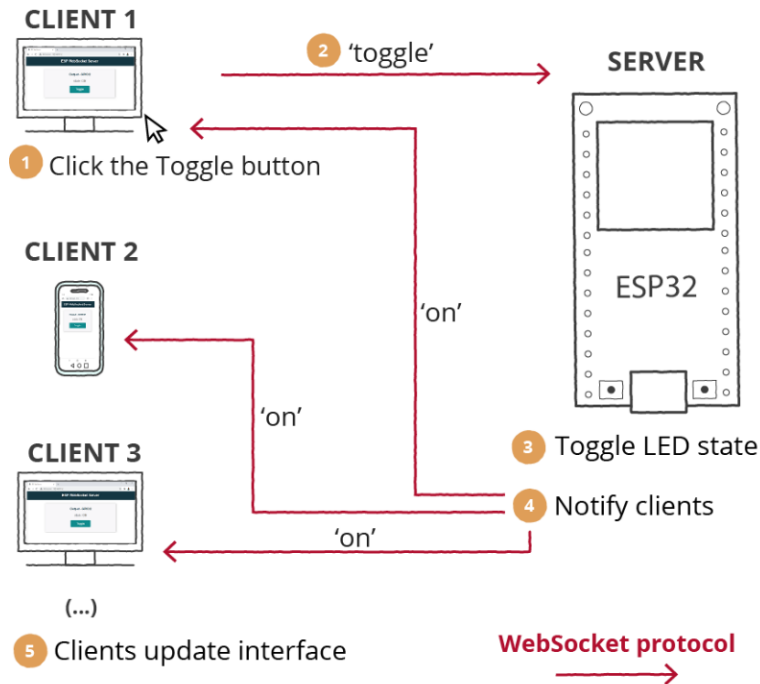


Figure 17: Architecture de communication WebSocket

7.4 Interface Web : Page d'Accueil

L'interface web constitue le point central d'interaction entre l'utilisateur et le bras robotique. Accessible depuis n'importe quel navigateur connecté au même réseau que l'ESP32, cette page d'accueil est conçue pour offrir une navigation intuitive, claire et réactive en proposant deux boutons principaux permettant de sélectionner le mode de commande souhaité:

Dans le **mode manuel**, l'utilisateur est redirigé vers une interface dédiée où il peut renseigner manuellement les coordonnées numériques X , Y et Z , définissant ainsi la position cible du bras robotique. Cette approche permet un positionnement précis basé sur des valeurs explicites. En revanche, le **mode vision** active automatiquement la caméra du dispositif, initiant ainsi le processus de détection des gestes. Ce mode repose sur l'analyse en temps réel des mouvements capturés, permettant une commande intuitive du bras à partir d'actions gestuelles interprétées par l'intelligence artificielle embarquée.

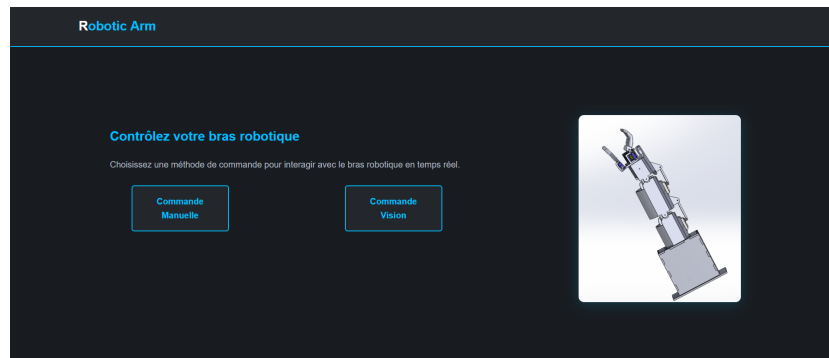


Figure 18: Page d'accueil

7.5 Mode 1 : Commande Manuelle par Interface Web

Dans ce mode, l'utilisateur saisit les coordonnées désirées dans trois champs dédiés à X, Y et Z, ou utilise des sliders pour ajuster la position du bras robotique. Ces données sont immédiatement envoyées via WebSocket sous forme de message JSON :

```
{  
  "mode": "manuel",  
  "x": 12.5,  
  "y": 7.0,  
  "z": 5.2  
}
```

L'ESP32 reçoit les coordonnées de la position cible, applique un algorithme de cinématique inverse afin de déterminer les angles nécessaires à chaque articulation, puis transmet les signaux de commande correspondants aux servomoteurs concernés. Ce mode de pilotage présente l'avantage d'offrir un contrôle précis et déterministe du bras, permettant une grande fiabilité dans l'exécution des mouvements. Il est particulièrement bien adapté aux scénarios où les positions à atteindre sont fixes ou prédéfinies, telles que les tâches de manipulation répétitive ou le placement structuré d'objets.

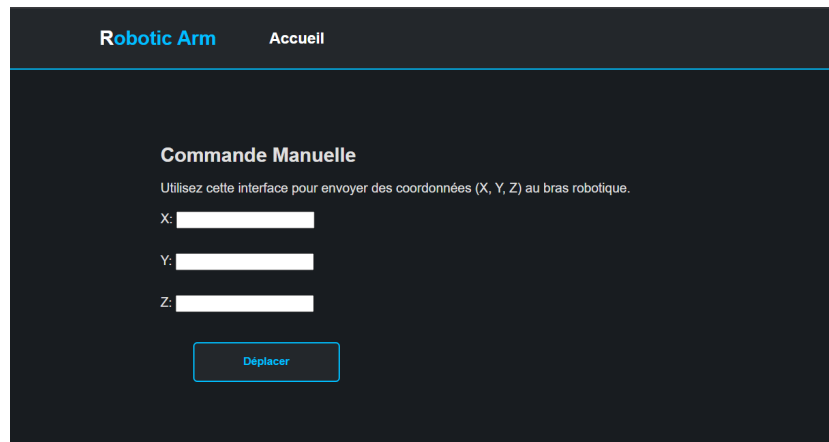


Figure 19: Page de la Commande Manuelle

7.6 Mode 2 : Commande par Vision (ml5.js)

Le second mode repose sur la reconnaissance de gestes de la main à l'aide de la bibliothèque **ml5.js**, basée sur TensorFlow.js . Le modèle **Handpose** détecte automatiquement 21 points clés de la main via la webcam. Ces informations sont analysées pour déterminer le geste effectué (main ouverte, poing fermé, direction pointée, etc.). [8, 9]

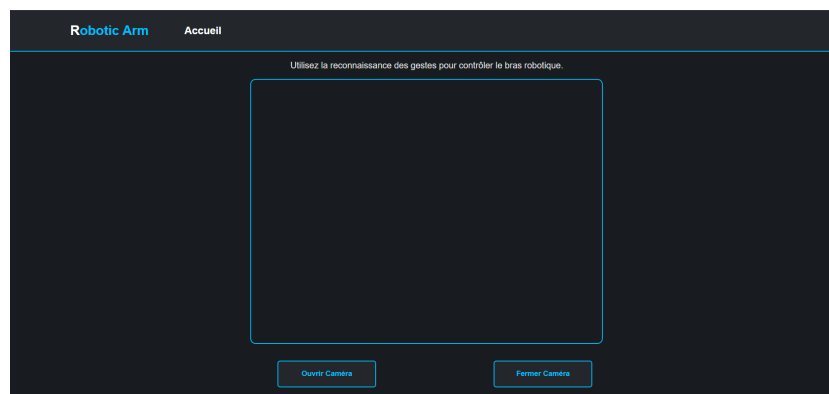


Figure 20: Page de la Commande Manuelle

Chaque geste reconnu déclenche l'envoi d'une commande structurée au format JSON à l'ESP32.

```

let handPose;
let video;
let hands = [];
let modelLoaded = false;
let started = false;

function preload() {
  // Load the handPose model
  handPose = ml5.handPose();
}

function setup() {
  createCanvas(640, 480).parent('canvasContainer');
  background(0);
}

function draw() {
  if (video && video.loadedmetadata) {
    image(video, 0, 0, width, height);

    for (let i = 0; i < hands.length; i++) {
      let hand = hands[i];
      console.log(JSON.stringify(hand.keypoints));
      if (connected) {
        socket.send(JSON.stringify(hand.keypoints));
      }

      for (let j = 0; j < hand.keypoints.length; j++) {
        let keypoint = hand.keypoints[j];
        fill(0, 255, 0);
        noStroke();
        circle(keypoint.x, keypoint.y, 10);
      }
    }
  }
}

```

Figure 21: Code pour la détection du mouvement du bras

7.7 La commande envoyée

La commande envoyée à l'ESP32 est sous la forme suivante : `{"x": , "y": , "z": }`

Le code suivant illustre comment, à partir des gestes détectés par la caméra, on peut extraire la position cible du bras robotique et transmettre les coordonnées correspondantes à l'ESP32 via WebSocket, en respectant le format JSON attendu.


```

[{"x": -4.46167757025008, "y": 417.25817070994344, "name": "wrist"},
{"x": 36.205588279034615, "y": 418.9747817016203, "name": "thumb_cmc"},
{"x": 78.03698223077406, "y": 380.9082501380545, "name": "thumb_mcp"},
{"x": 98.385811247487, "y": 344.3233952188695, "name": "thumb_ip"},
{"x": 93.7711948632413, "y": 320.63896547892404, "name": "thumb_tip"},
{"x": 69.15831281141638, "y": 318.59703645021114, "name": "index_finger_mcp"},
{"x": 84.38935616370983, "y": 292.47361246907514, "name": "index_finger_pip"},
{"x": 84.4435811813315, "y": 298.00569336367823, "name": "index_finger_dip"},
{"x": 73.12972625608468, "y": 325.52712792716034, "name": "index_finger_tip"},
{"x": 37.365958752221225, "y": 300.404050960069, "name": "middle_finger_mcp"},
{"x": 43.57462256560615, "y": 248.49323301596698, "name": "middle_finger_pip"},
{"x": 46.61023558436872, "y": 243.3787904274222, "name": "middle_finger_dip"},
{"x": 42.26595869633138, "y": 269.5688626540319, "name": "middle_finger_tip"},
{"x": 4.85745690021262, "y": 297.31009103546296, "name": "ring_finger_mcp"},
{"x": 2.2148018315211893, "y": 250.48096419543882, "name": "ring_finger_pip"},
{"x": -0.3533686054460716, "y": 248.06377422910936, "name": "ring_finger_dip"},
{"x": -3.9898825795286452, "y": 267.68824372327373, "name": "ring_finger_tip"},
{"x": -27.229630065595583, "y": 306.21063738702657, "name": "pinky_finger_mcp"},
{"x": -37.858038143268274, "y": 274.75858267481465, "name": "pinky_finger_pip"},
{"x": -40.84085366639785, "y": 270.6496198794054, "name": "pinky_finger_dip"},
{"x": -42.42153566587652, "y": 281.9535306067641, "name": "pinky_finger_tip"}]

```

Figure 22: Exemples de la commande envoyée au ESP32

7.8 Interprétation côté ESP32

L'ESP32 agit comme un serveur WebSocket. Lorsqu'un message JSON est reçu, il est analysé pour extraire le mode et la commande. [10]

En mode **manuel**, les coordonnées X , Y et Z reçues sont exploitées pour calculer les angles articulaires nécessaires à l'orientation du bras, à l'aide d'un algorithme de cinématique inverse. Ce calcul permet d'atteindre précisément la position cible spécifiée par l'utilisateur. En revanche, en mode **vision**, les données reçues correspondent à des commandes symboliques, chacune étant associée à une séquence de mouvements prédéfinie. Ce fonctionnement permet d'exécuter rapidement des gestes appris ou planifiés à partir d'une reconnaissance visuelle, sans nécessiter de recalcul en temps réel des trajectoires.

```

#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>

#define SSID "TOPNET_VSKC"
#define PASSWORD "a47qhmlwxy"

AsyncWebServer server(80);
AsyncWebSocket ws("/ws");

unsigned long lastTime = 0;
unsigned long timerDelay = 500;

void handle_recieved_msg(String message){
  Serial.println(message);
}

void initWiFi(char* ssid, char* password) {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
}

```

```

void initWiFi(char* ssid, char* password) {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

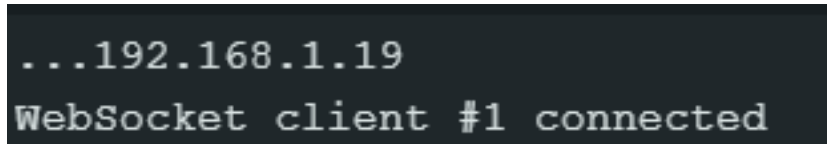
void onEvent(AsyncWebSocket *server, AsyncWebSocketClient *client, AwsEventType type, void *arg, uint8_t *data, size_t len){
  switch (type)
  {
    case WS_EVT_CONNECT:
      Serial.printf("WebSocket client #%u connected\n", client->id());
      break;
    case WS_EVT_DISCONNECT:
      Serial.printf("WebSocket client #%u disconnected\n", client->id());
      break;
    case WS_EVT_DATA:
      AwsFrameInfo *info = (AwsFrameInfo *)arg;
      if (info->final && info->index == 0 && info->len == len && info->opcode == WS_TEXT)
      {
        data[len] = 0;
        String message = (char*)data;
        handle_recieved_msg(message);
      }
      break;
  }
}

void send_msg(String msg){
  ws.textAll(msg);
}

```

Figure 23: Partie du code de la communication et de la connexion au serveur

Le traitement se fait en temps réel, avec un retour de réponse possible (accusé de réception, message d'erreur, etc.).

A terminal window with a dark background and light green text. The first line shows the IP address "...192.168.1.19". The second line shows the message "WebSocket client #1 connected".

```
...192.168.1.19  
WebSocket client #1 connected
```

Figure 24: Réponse après la connexion

7.9 Conclusion

L'intégration du protocole WebSocket avec les deux modes de commande — manuel par coordonnées et automatique par vision — permet une grande flexibilité et interactivité dans le contrôle du bras humanoïde. L'approche web multiplateforme et sans fil ouvre la voie à des applications plus évoluées comme la téléopération à distance, le suivi de gestes ou l'intégration avec la réalité augmentée.

VI. Conclusion Générale

Ce projet de conception et de réalisation d'un bras robotique humanoïde a permis de combiner plusieurs disciplines de l'ingénierie moderne, telles que la modélisation mécanique, l'électronique embarquée, la programmation web et l'intelligence artificielle. L'objectif principal était de concevoir un système interactif, flexible et évolutif, capable d'être contrôlé en temps réel à travers différentes interfaces, tout en assurant une réponse fluide et précise.

Grâce à l'intégration du protocole WebSocket, la communication bidirectionnelle entre l'utilisateur et le bras robotique est devenue instantanée, ouvrant la voie à un contrôle plus naturel. L'utilisation de la vision par ordinateur, via la bibliothèque `m15.js`, a ajouté une dimension intuitive à la commande du bras, permettant l'interaction par gestes. Par ailleurs, les phases de simulation avec MATLAB, Simulink, et Pygame ont été essentielles pour valider la cinématique avant toute implémentation matérielle.

Sur le plan électronique, l'usage de composants fiables, comme l'ESP32, ainsi que la conception de circuits sur EasyEDA ont renforcé la robustesse du système. Du côté logiciel, une architecture modulaire a été adoptée afin de faciliter les futures évolutions du projet, comme l'ajout de capteurs, de nouveaux modes de commande, ou encore l'intégration dans un environnement ROS.

En somme, ce projet a permis non seulement de développer un prototype fonctionnel, mais aussi de poser les bases d'un système robotique intelligent, interactif et extensible, répondant aux exigences actuelles de la robotique personnelle et industrielle. Il constitue une excellente plateforme d'apprentissage pour des projets plus ambitieux mêlant robotique, intelligence artificielle et systèmes embarqués.

References

- [1] R. Mischianti, “Esp32 nodemcu 32s (esp-32s) kit high resolution pinout, datasheet and specs,” 2020. Consulté le 1 Juin 2025.
- [2] EasyEDA Team, “Easyeda - online pcb design circuit simulator,” 2024. Consulté le 1 Juin 2025.
- [3] I. The MathWorks, *MATLAB - R2023b*. MathWorks, Natick, Massachusetts, United States, 2023. <https://www.mathworks.com/products/matlab.html>.
- [4] I. The MathWorks, *Simulink - R2023b*. MathWorks, Natick, Massachusetts, United States, 2023. <https://www.mathworks.com/products/simulink.html>.
- [5] I. The MathWorks, *Simscape Multibody - R2023b*. MathWorks, Natick, Massachusetts, United States, 2023. <https://www.mathworks.com/products/simscape-multibody.html>.
- [6] pygame community, “Pygame - python game development.” <https://www.pygame.org/>, 2023. Version 2.5.0.
- [7] I. Fette and A. Melnikov, “The websocket protocol.” <https://datatracker.ietf.org/doc/html/rfc6455>, 2011. RFC 6455.
- [8] ml5.js contributors, “ml5.js: Friendly machine learning for the web.” <https://ml5js.org/>, 2023.
- [9] TensorFlow.js Team, “Tensorflow.js: Machine learning for the web.” <https://www.tensorflow.org/js>, 2023.
- [10] R. Santos, “Esp32 websocket server using arduino ide.” <https://randomnerdtutorials.com/esp32-websocket-server-arduino/>, 2023.