# Hygie

Projet de robotique et IA appliqué au domaine de la santé

Ministère du bien vivre en bonne santé & Epitech Digital School

Laetitia Granjon

# Analyse opérationnel

**Brief** : Le client a besoin d'un robot assistant pour répondre à la pénurie de médecin dans les hôpitaux. Il dispose déjà d'un robot.

Cette solution se doit d'être réactive, précise et fiable, intégrant des fonctionnalités avancées de reconnaissance vocale, de traitement du langage naturel, et de navigation autonome. Elle doit aussi être suffisamment automne en termes de batterie pour pouvoir se déplacer d'un point A à un point B et revenir à la base.

Voici une synthèse des besoins exprimés par le client, structurée autour de plusieurs axes clés :

## Contexte Opérationnel et Utilisation

- **Environnement d'utilisation** : Hôpital, spécifiquement les chambres des patients. Il faut noter que le robot sera aussi amené à évoluer dans les couloirs de l'hôpital pour se rendre d'une chambre à une autre par exemple. L'aménagement des couloirs d'hôpitaux changent souvent en fonction des besoins (lit dans le couloir, réaménagement de cloisons, …), c'est à prendre en compte dans le déplacement du robot.
- **Utilisateurs principaux** : Personnel soignant (médecins et infirmiers) qui devront répondre aux demandes résumées par le robot,  téléopérateurs via une interface de téléopération, patients.

## Fonctionnalités Clés du Robot

- **Reconnaissance vocale et capacité d'écoute en direct** : Capable de comprendre et d'exécuter des commandes vocales simples pour se déplacer vers les chambres des patients et écouter en direct les besoins de ces derniers une fois dans la chambre.
- **Autonomie de navigation** : Naviguer de manière autonome vers l'emplacement spécifié (la chambre du patient) en réponse à une commande vocale.

- **Gestion du silence (exigence de robustesse)** : Identifier et réagir adéquatement aux différents types de silence (normal vs. anormal), incluant la capacité de signaler si aucun son n'est détecté après une période d'attente définie (une fois à l'arrêt par exemple).
- **Traitement des demandes** : Capacité à filtrer et trier les demandes des patients, en différenciant les requêtes pertinentes de celles non liées aux besoins immédiats (ex. demandes familiales, distractions). Capacité à gérer efficacement les situations où plusieurs patients se trouvent dans la même chambre, nécessitant une segmentation vocale précise

## Formation et Briefing des Utilisateurs

- Informer les patients sur le fonctionnement du robot et les comportements attendus de leur part pour faciliter la communication et l'interaction efficaces.
- Informer les médecins sur le fonctionnement du robot et la manière de recevoir les informations récoltées pour en faciliter et optimiser le traitement.

## Matrice des parties prenantes

| Parties Prenantes | Intérêts | Influence | Attentes | Interactions attendues |
|---|---|---|---|---|
| **Manager de l'hôpital** | Efficacité opérationnelle, rentabilité, sécurité, innovation | Élevée : décisions stratégiques et budgétaires | Succès du projet, ROI positif, amélioration de la satisfaction des patients | Probablement aucune si ce n'est de la supervision. |
| **Médecins et infirmiers** | Amélioration des soins, efficacité, support aux tâches | Élevée : utilisateurs principaux, feedback sur l'usage | Facilité d'utilisation, fiabilité, amélioration de la charge de travail | Doivent avoir accès aux retours des patients, traités par le robot, sur une interface dédiée. |
| **Téléopérateur du robot** | Efficacité opérationnelle, ergonomie de l'interface | Moyenne : interaction directe avec le système | Interface intuitive, réponse rapide du système, fiabilité | Interaction orale directive. Ce sont eux qui donnent les ordres au robot. |
| **Observateurs extérieurs** | Sécurité, accessibilité, non-perturbation | Faible à Moyenne : impact indirect | Coexistence sûre avec le robot, respect de la confidentialité, minimisation des obstructions | Normalement pas d'interactions. Il est possible qu'il existe des interactions de curieux. Le robot devra les ignorer. |

| | | | | |
|---|---|---|---|---|
| **Patients** | Qualité des soins, réactivité, confort | Élevée : bénéficiaires directs du service | Interaction simple, respect de la vie privée | Interaction orale : les patients doivent parler au robot et lui dire ce dont ils ont besoin |
| **Famille des patients** | Bien-être et soins aux patients, communication | Moyenne : impact émotionnel et soutien aux patients | Assurance de la qualité des soins, accessibilité à l'information sur l'état du patient | Pas d'intéractions directes |
| **Développeuse** | Simplicité et rapidité de développement Facilité de maintenance | Elevée : Développement + Maintenance et support une fois le projet livré | Système fonctionnel, adaptable, réplicable et qui demande peu de maintenance. | Interactions orales et virtuelles avec le système et les composants lors de la création. |

# Spécification système

Voici les spécifications techniques clés du Raspberry Pi 4 Model B :

## Processeur

- CPU : Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC
- Fréquence : 1.5GHz

## Mémoire

- RAM : Options de  8GB LPDDR4-3200 SDRAM

## Connectivité

- WiFi : 2.4 GHz et 5.0 GHz IEEE 802.11ac
- Bluetooth : Bluetooth 5.0, BLE
- Ethernet : Gigabit Ethernet
- GPIO : 40-pin GPIO header, entièrement compatible avec les modèles précédents

## Ports d’Entrée/Sortie et Connecteurs

- USB : 2 ports USB 3.0, 2 ports USB 2.0
- Vidéo et son :
    - 2 ports micro-HDMI (jusqu'à 4kp60 pris en charge)
    - Port de sortie audio/vidéo composite 3,5 mm
- Stockage : Lecteur de cartes micro-SD pour le système d'exploitation et le stockage des données
- Alimentation :
    - Connecteur USB-C pour l'entrée d'alimentation (5V 3A minimum)
    - Power over Ethernet (PoE) capable (avec PoE HAT séparé)

## Autres Caractéristiques

- Vidéo : Prise en charge de la décodification HEVC 4K60, prise en charge de la vidéo H.264 et de la vidéo 1080p60 H.264 encode
- Multimédia : OpenGL ES 3.0 graphics

- Système d’exploitation : MicroSD card slot pour charger l'OS et le stockage des données
- Alimentation recommandée : 5.1V 3A via connecteur USB-C ou GPIO

## Dimensions

- Dimensions physiques : 85.6 mm × 56.5 mm

# Architecture définition du système : schéma du robot

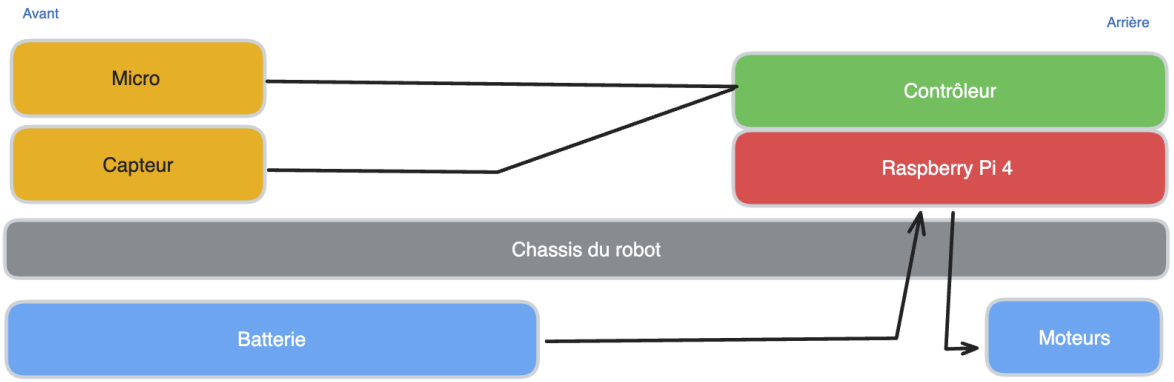
## Conception du Système

- **Choix des composants**: Sélection de composants compatibles avec le Raspberry Pi 4, comme un module Wi-Fi qui peut se connecter facilement à la carte GPIO ou via USB, des moteurs avec des pilotes correspondants, un microphone compatible avec les entrées du Raspberry Pi, et un capteur à ultrasons qui peut être connecté aux pins GPIO.
- **Disposition des composants**: Planification de la disposition physique des composants sur la plateforme robotique pour garantir l'accessibilité et la fonctionnalité. Par exemple, le capteur à ultrasons doit être placé à l'avant du robot pour détecter les obstacles, tandis que le microphone doit être positionné pour capter clairement le son sans obstruction.
- **Alimentation électrique**: Utilisation d'une batterie (Li-Po ou Li-ion), avec un circuit de gestion de l'alimentation + puissance e
- **Intégration des composants**: Intégration des composants électriques avec des câbles et des connecteurs adaptés, en suivant les spécifications du Raspberry Pi 4 pour les connexions GPIO et USB.

## Assemblage Physique

- **Structure du robot**: Commencez par assembler la structure physique ou le châssis du robot qui supportera tous les composants.
- **Installation des moteurs**: Montez les moteurs sur le châssis et connectez-les au contrôleur de moteur, qui sera ensuite connecté au Raspberry Pi.
- **Montage du raspberry Pi**: Fixez le Raspberry Pi 4 sur la plateforme, en le protégeant avec un boîtier adapté si nécessaire.
- **Connexion des capteurs:** Attachez le capteur à ultrasons à l'avant du robot et connectez-le aux pins GPIO appropriés du Raspberry Pi pour l'entrée de données.
- **Configuration du microphone:** Installez le microphone de manière à capter efficacement la voix des utilisateurs et connectez-le au Raspberry Pi.
- **Module Wi-Fi**: Le Raspberry Pi 4 est déjà équipé du Wi-Fi intégré.

Schéma du système




## Configuration Logicielle

- **Système d'exploitation:** Installation d'un système d'exploitation compatible, comme Raspberry Pi OS, et configuration pour démarrer avec les paramètres nécessaires.
- **Pilotes et bibliothèques**: Installation des pilotes nécessaires pour les moteurs, le capteur à ultrasons, et le microphone.

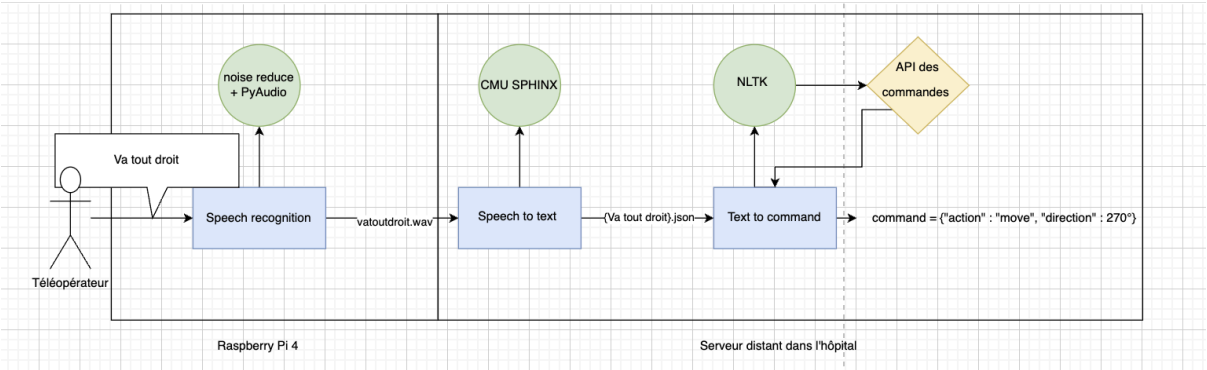
  Spécifier plus pilote et bibliothèques à utiliser

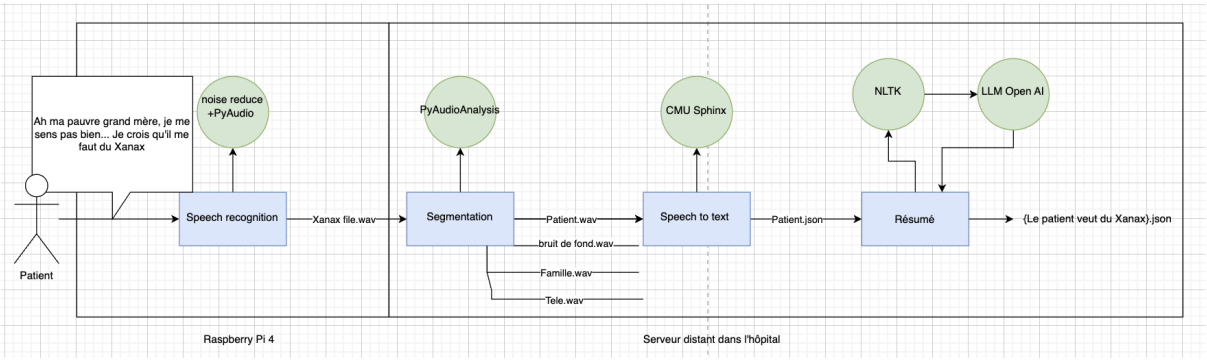
## Intégration d'un module de traitement de la voix

Nous avons besoin dans ce projet d'avoir une écoute en direct des patients et des commandes. Il faut le moins de latence possible. Le Raspberry Pi 4 n'a pas assez de puissance de calcul pour supporter les demandes d'un LLM complet. Cela dit,  il n'est pas envisageable que le module de traitement de la voix soit hors du robot. Il faut donc trouver un moyen de l'intégrer dans le raspberry.

Pour cela, on peut faire appel à un modèle allégé optimisé pour les appareils à faible puissance comme le nôtre. On va utiliser PyAudio, une librairie python qui sert à faire de la reconnaissance vocale et génère des fichiers audio.

# Spécification et architecture des modèles




Lien vers les  graphiques:
https://app.diagrams.net/#G13yGByn-I3sYy7bmmXvADrR7hZatyXvSw#%7B%22pageId%22%3A%22SvPJUJCPhPammDTppkqC%22%7D

## Justification de la répartition des librairies

Ce choix d'architecture et de répartition des tâches entre le Raspberry Pi et un serveur distant repose sur une réflexion stratégique alignée avec les exigences techniques et opérationnelles du projet.

**PyAudio et noisereduce sur le Raspberry Pi:**

- **Immédiateté et confidentialité** : Enregistrer et effectuer une réduction initiale du bruit directement sur le Raspberry Pi permet de commencer

immédiatement le traitement audio sans délai de transmission. Cela assure également que les données sensibles (comme les voix des patients) ne quittent pas le dispositif avant d'être anonymisées ou résumées, ce qui est crucial dans un contexte hospitalier pour respecter la confidentialité des patients. Anonymisation est en option (pas un attendu)
- **Réduction de la charge du réseau** : Traiter l'audio localement réduit la quantité de données nécessitant un transfert réseau, ce qui est particulièrement pertinent dans un environnement hospitalier où la bande passante peut être un facteur limitant.

**PyAudioAnalysis, CMU Sphinx, NLTK, et LLM OpenAI sur un ordinateur:**

- **Puissance de calcul** : Les tâches de segmentation, de reconnaissance vocale, d'analyse de texte, et de génération de résumé sont gourmandes en ressources. Un serveur distant, probablement plus puissant que le Raspberry Pi, peut traiter ces tâches plus efficacement, offrant une meilleure précision et des temps de réponse réduits pour les analyses complexes.
- **Scalabilité** : Un serveur offre la flexibilité de scaler les ressources selon les besoins, ce qui est essentiel pour gérer des volumes variables d'interactions patient-robot ou pour déployer des mises à jour et des améliorations sans interrompre les services.
- **Maintenance et mise à jour** : Centraliser les bibliothèques et outils les plus complexes sur un serveur facilite la maintenance et les mises à jour, permettant d'améliorer le système sans nécessiter une intervention physique sur chaque robot.

**Création d'une API de commandes de références:**

- **Standardisation des interactions** : Une API de commandes de références standardise la manière dont les commandes sont interprétées et exécutées par le robot, ce qui améliore la cohérence et la fiabilité des interactions avec les utilisateurs.
- **Intégration et extension facilitées :** Avec une API, il devient plus facile d'intégrer d'autres systèmes ou dispositifs dans l'écosystème du robot, et de développer de nouvelles fonctionnalités ou services sans perturber le fonctionnement existant.
- **Déploiement et diagnostic** : L'API sert de couche d'abstraction entre les commandes utilisateur et l'exécution des actions par le robot, facilitant le déploiement de correctifs, la surveillance des performances, et le diagnostic des problèmes.

## Justification des choix de librairies/outils :

Les choix de bibliothèques et outils pour ce projet sont principalement guidés par la compatibilité avec le matériel, la spécificité de la tâche, la facilité d'utilisation, et la performance. Voici une justification pour chaque choix de bibliothèque :

**PyAudio**

- **Compatibilité avec le Raspberry Pi** : PyAudio est bien supporté sur le Raspberry Pi, ce qui en fait un choix fiable pour l'enregistrement audio directement depuis le dispositif.
- **Simplicité d'utilisation** : Offre une interface simple pour l'enregistrement et la lecture audio, ce qui est essentiel pour démarrer rapidement le développement sans se heurter à une courbe d'apprentissage abrupte.
- **Flexibilité** : Permet un contrôle précis sur les paramètres d'enregistrement audio, ce qui est crucial pour optimiser la qualité des enregistrements dans un environnement hospitalier variable.

**noisereduce**

- **Spécialisation dans la réduction de bruit** : Cette bibliothèque est spécifiquement conçue pour la réduction de bruit, offrant des algorithmes optimisés pour améliorer la qualité de l'audio enregistré, une étape préliminaire importante avant la reconnaissance vocale.
- **Intégration facile avec PyAudio** : Peut être utilisée en conjonction avec PyAudio pour traiter l'audio juste après l'enregistrement, simplifiant le flux de travail de prétraitement.

**PyAudioAnalysis**

- **Large éventail d'analyses audio** : Propose des fonctionnalités complètes pour l'analyse audio, y compris la segmentation de la parole, qui est essentielle pour isoler les segments pertinents avant la reconnaissance vocale.
- **Indépendance du langage** : Contrairement à certains outils spécialisés dans une langue spécifique, PyAudioAnalysis fournit des analyses basées sur les caractéristiques sonores, permettant une application plus large.

**CMU Sphinx (PocketSphinx)**

- **Reconnaissance vocale hors ligne** : Offre la capacité de transcrire la parole en texte sans nécessiter une connexion Internet, un atout important pour garantir la fonctionnalité continue dans un hôpital.
- **Légèreté et performance** : Comparativement à d'autres moteurs de reconnaissance vocale, CMU Sphinx ne consomme pas énormément de ressources.

**NLTK**

- **Richesse des fonctionnalités NLP** : NLTK est une des bibliothèques les plus complètes pour le traitement du langage naturel en Python, offrant des outils pour la tokenisation, l'analyse syntaxique, l'extraction d'entités nommées, et plus encore.
- **Communauté et support** : En tant que bibliothèque établie, NLTK bénéficie d'une large communauté d'utilisateurs et d'une documentation exhaustive, facilitant le développement et la résolution de problèmes.

**LLM OpenAI** (budget 10 €)

- **Capacités de compréhension et de génération de langage avancées** : Les modèles de langage d'OpenAI sont à la pointe de la technologie NLP, capables de comprendre le contexte complexe et de générer des résumés cohérents.
- **Continuelle amélioration et suppor**t : OpenAI continue d'améliorer ses modèles et offre un support technique, ce qui garantit que le projet peut bénéficier des avancées les plus récentes en IA.

**GPIO Zero**

- Interface **simple et intuitive**
- **Contrôle facile des composants** tels que les moteurs, les LED, les boutons, etc.
- **Orienté objet**.

Il me semblait important de choisir des solutions éprouvées, fiables et bien supportées, offrant un équilibre entre performance, facilité d'utilisation, et adaptabilité aux contraintes spécifiques du projet.

# Data Engineering

## Flux de Données

Le système du robot va générer et traiter un flux de données conséquent, issu principalement de la reconnaissance vocale et des interactions avec le personnel médical et les patients.

## Volumes de Données

Les volumes de données échangés seront significatifs, étant donné que chaque interaction avec un patient ou un membre du personnel médical sera enregistrée, analysée et stockée temporairement pour traitement. On estime que chaque interaction générera environ 0.5 Mo de données audio, avec des centaines d'interactions par jour par robot.

## Format des Données

1. **Données Audio** : Capturées en temps réel et stockées initialement au format WAV pour assurer une haute fidélité sonore.
2. **Données Textuelles** : Les transcriptions issues de la reconnaissance vocale seront stockées au format JSON pour une intégration facile avec les systèmes d'analyse et de réponse.
3. **Données de Navigation** : Informations de localisation et de mouvement du robot, capturées en format de logs structurés pour une analyse de performance et d'optimisation de l'itinéraire.

# Validation

## Librairies - Protocole de test

**Objectif**

Vérifier que les librairies de reconnaissance vocale et de traitement audio (telles que PyAudio, noisereduce, PyAudioAnalysis, CMU Sphinx) reconnaissent correctement les commandes vocales et les traitent conformément aux instructions définies dans les fichiers JSON.

**Équipements et Logiciels Requis**

- Fichiers audio des commandes vocales (format M4A)
- Fichiers JSON correspondants décrivant les intentions des commandes.
- Environnement de développement Python configuré avec les librairies nécessaires (PyAudio, CMU Sphinx, etc.).
- Ordinateur avec microphone et haut-parleurs pour tests de lecture et d'enregistrement.

**Méthodologie de Test**

1. *Préparation*
    - Installer et configurer toutes les librairies nécessaires sur l'environnement de développement.
    - S'assurer que les fichiers audio et les fichiers JSON sont accessibles depuis l'environnement de test.

2. *Tests de Reconnaissance Vocale*
    - Pour chaque fichier audio :
        1. Jouer le fichier audio via un haut-parleur pour simuler une commande vocale naturelle.
        2. Utiliser la librairie de reconnaissance vocale pour capturer et transcrire la commande.
        3. Comparer la transcription obtenue avec la commande du fichier JSON correspondant.
        4. Vérifier que l'intention et les paramètres capturés correspondent à ceux spécifiés dans le JSON.

3. *Tests de Traitement Audio*
    - Utiliser PyAudioAnalysis pour analyser les caractéristiques audio de chaque commande (par exemple, fréquence, amplitude).
    - Appliquer noisereduce pour évaluer l'efficacité de la réduction de bruit sur les commandes vocales enregistrées dans un environnement bruyant.

4. *Validation des Réponses*
    - Simuler la réponse du système du robot basée sur l'intention et les paramètres décodés.
    - Enregistrer si le robot aurait exécuté l'action correcte basée sur les données décodées.

5. *Rapport et Analyse*
    - Documenter les résultats de chaque test, y compris les réussites et les échecs de reconnaissance.
    - Analyser les causes des échecs pour identifier des améliorations possibles des librairies ou des paramètres de configuration.

**Critères de Réussite**

- Les transcriptions des commandes vocales doivent correspondre exactement au ‘commandText’ pour au moins 95% des tests.
- Les intentions et paramètres décodés doivent correspondre exactement aux valeurs dans les fichiers JSON pour toutes les commandes.

**Documentation**

- Créer un rapport détaillé comprenant les configurations utilisées, les résultats des tests, et les recommandations pour l'amélioration du système de reconnaissance vocale du robot.

# Composant ML - Protocole de test

### Objectif

S’assurer que les librairies de reconnaissance vocale et de traitement audio fonctionnent correctement sur le Raspberry Pi.

### Équipements et logiciels requis :

- Le Raspberry Pi configuré avec le système d'exploitation Raspberry Pi OS.
- Les librairies audio préalablement testées installées sur le Raspberry Pi.
- Un microphone connecté au Raspberry Pi.
- Un environnement de développement pour accéder et contrôler le Raspberry Pi (par exemple, SSH ou moniteur direct avec clavier).

### Méthodologie de Test

1. *Configuration Initiale*
    - Installer et configurer le Raspberry Pi avec toutes les librairies nécessaires et les dépendances.
    - S’assurer que le microphone est correctement configuré et reconnu par le système.

2. *Tests de Capture Audio*
    - Exécuter un script Python utilisant PyAudio pour enregistrer un échantillon audio via le microphone et sauvegardez-le sur le Raspberry Pi.

3. *Tests de Reconnaissance Vocale*
    - Utiliser des scripts utilisant les librairies de reconnaissance vocale pour transcrire des phrases ou commandes simples prononcées dans le microphone.
    - Comparer les transcriptions obtenues avec les phrases attendues pour évaluer la précision de la reconnaissance.

4. *Tests de Traitement Audio Avancé*
    - Appliquer des filtres ou des traitements (comme la réduction du bruit via noisereduce) sur des enregistrements audio pour tester la capacité de traitement audio du Raspberry Pi.
    - Évaluer la qualité de l'audio traité pour détecter toute dégradation ou altération non désirée.

5. *Tests de Performance et de Stabilité*
    - Exécuter les librairies de manière continue sur plusieurs heures pour identifier toute fuite de mémoire, surchauffe du Raspberry Pi, ou autres problèmes de stabilité.
    - Contrôler et surveiller l'utilisation des ressources (CPU, mémoire) pour s'assurer qu'elles restent dans des limites acceptables.

**Critères de Réussite**

- Le Raspberry Pi doit réussir à capturer des enregistrements audio.
- Les transcriptions réalisées par les librairies de reconnaissance vocale doivent atteindre une précision minimale de 90%.
- Le système doit rester stable et performant sous charge prolongée.

**Documentation**

- Documenter chaque étape du test, les résultats obtenus, et les problèmes rencontrés.
- Rédiger un rapport de test détaillé comprenant des recommandations pour toute amélioration nécessaire.

## Intégration du composant ML sur le robot - Protocole de test

### Objectif

Vérifier que le robot peut avancer correctement en réponse aux commandes délivrées par le module ML.

### Équipements et Logiciels Requis

- Robot équipé d'un Raspberry Pi.
- Espace de test dégagé avec marquages au sol pour mesurer la distance et la direction du déplacement.
- Caméra pour enregistrer visuellement le parcours du robot.
- Logiciel ou script pour envoyer des commandes au Raspberry Pi et enregistrer les réponses du robot.

### Méthodologie de Test

1. *Configuration Initiale*
    - S'assurer que le Raspberry Pi est correctement installé et configuré sur le robot.
    - Établir un périmètre de test clair.

2. *Tests de réactivités à des commandes*
    - Commande d'avancement simple :
        - Placer le robot à un point de départ.
        - Envoyer une commande de déplacement simple ("avancer de X mètres").
        - Observer le parcours du robot.

    - Séquences de déplacement :
        - Programmer une séquence de mouvements qui inclut des changements de direction.
        - Exécuter la séquence vérifier que le robot suit précisément les instructions données.

3. *Évaluation de la Stabilité*
    - Envoyer des commandes de déplacement en continu pendant une période prolongée pour tester la stabilité du robot.
    - Noter toute déviation, arrêt ou autre problème de performance.

4. *Tests de Sécurité*
    - S’assurer que le robot s'arrête comme prévu en réponse à des obstacles.

**Critères de Réussite**

- Le robot doit exécuter correctement toutes les commandes de déplacement, avec une marge d'erreur de moins de 5% par rapport aux distances et directions prévues.
- Le système doit rester stable et réactif sous une charge opérationnelle continue.

**Documentation**

- Documenter tous les résultats de test, les anomalies, les mesures correctives prises, et les observations générales.
- Préparer un rapport final de test qui inclut des recommandations pour des ajustements ou des améliorations du système de contrôle du robot.

## Opérationnel - Protocole de test

### Objectif

Valider la fonctionnalité, la performance et la sécurité du robot dans un environnement hospitalier, en s'assurant qu'il répond aux besoins opérationnels et aux attentes en matière de soins aux patients. Ce protocole couvrira les tests de reconnaissance vocale, de navigation autonome, de gestion des interactions patient-robot, et de sécurité des données.

### Équipements Requis

- Le robot.
- Un système de contrôle à distance.
- Environnement de test reproduisant un cadre hospitalier avec des obstacles et des bruits de fond variés.

### Méthodologie de Test

1. *Tests de Reconnaissance Vocale*
    - **Objectif** : S'assurer que le robot peut correctement reconnaître et répondre aux commandes vocales dans un environnement bruyant.
    - **Procédure** :
        - Placer le robot dans différents lieux de l'hôpital (couloirs, chambres des patients, zones de repos).
        - Effectuer des commandes vocales standardisées et enregistrez la précision de la reconnaissance et du traitement des requêtes.
        - Varier le volume et le ton des commandes pour tester la robustesse du système de reconnaissance vocale.

2. *Tests de Navigation Autonome*
    - **Objectif** : Confirmer que le robot peut naviguer de manière autonome et éviter les obstacles.
    - **Procédure** :
        - Définir un parcours d'obstacles simulant les conditions d'un hôpital (lits, chariots, personnes).
        - Laisser le robot naviguer dans le parcours et enregistrer les incidents de collision ou déviation de trajectoire.

3. *Tests d'Interaction Patient-Robot*
    - **Objectif** : Vérifier que le robot gère les interactions avec les patients.
    - **Procédure** :

- Simuler des scénarios d'interaction avec le robot où des "patients" posent des questions ou font des demandes spécifiques.
- Évaluer la capacité du robot à comprendre, traiter et répondre correctement à ces interactions.
- Tester la capacité du robot à gérer simultanément plusieurs patients dans une même chambre.

**Critères de Réussite**

- Le robot doit réussir à comprendre et exécuter au moins 95% des commandes vocales correctement.
- Le robot doit naviguer sans collision dans 90% des essais sur le parcours d’obstacles.
- Le robot doit réussir à gérer les interactions avec les patients avec une précision de réponse de 90%.

**Documentation**

- Enregistrer tous les résultats des tests, les observations et les incidents dans un rapport de test détaillé.
- Documenter les zones nécessitant des améliorations ou des ajustements.