PROJET D'AIDE À LA NAVIGATION POUR AVEUGLES

SPÉCIFICATIONS

SUIVI DES MODIFICATIONS DU DOCUMENT			
Version	**Date**	**Validation**	**Commentaire**
A	14/10/2025	Gilles VENTURINI	Rédaction initiale

REDACTEUR : Alexandre BOURCIER	***VALIDATEUR :*** Gilles VENTURINI
CLIENT : Gilles VENTURINI	**ENCADRANT :** Frédéric CHAUVIN

Sommaire

Objet du document 3

1. Introduction 4

2. Description du système 5

2.1. Fonctionnement 5

2.2. Environnement matériel 5

2.3. Environnement logiciel 5

3. Besoins fonctionnels 6

3.1. MUST HAVE 7

3.2. SHOULD HAVE 9

3.3. COULD HAVE 9

3.4. WON'T HAVE 10

4. Interfaces 11

4.1. Interfaces matérielles 11

4.2. Interfaces logicielles 11

4.3. Interface audio 11

5. Tests de validation 12

6. Glossaire 13

Objet du document

Le présent document a pour objectif de définir les spécifications fonctionnelles et techniques relatives aux évolutions logicielles du système d’aide à la navigation pour aveugles.

Il précise les modifications à apporter aux programmes existants, les contraintes techniques à respecter, ainsi que les objectifs de simplification, d’homogénéisation et de fiabilisation du code.

L’ensemble de ces spécifications servira de base à la conception, au développement et à la validation des améliorations à travers une série de tests fonctionnels.

1. Introduction

- **Contexte du projet** : Reprise d'une maquette existante de canne intelligente développée dans un ancien projet POLYTECH.
- **Objectif général** : Améliorer la détection, la fiabilité et l'ergonomie du système.
- **Périmètre** : Le projet couvre le logiciel embarqué (Linux + YOLO + capteurs), la communication audio, et la robustesse matérielle.
- **Livrables attendus** : Code fonctionnel, maquette améliorée, manuel utilisateur, rapport, démonstration.

2. Description du système

2.1. Fonctionnement

- Description du fonctionnement global (capteurs → traitement → sortie vocale).
- Modes de fonctionnement :
    1. **Mode Marche** : Ultrason seul → alerte distance.
    2. **Mode Exploration** : YOLO seul → annonce objets reconnus.
    3. **Mode Mixte** : Ultrason + YOLO → description + distance.

2.2. Environnement matériel

- Carte embarquée : Jetson Nano ARM A57 - Maxwell128 cœurs CUDA
- Capteurs : caméra IMX219-200, capteur ultrason A02YYUW, bouton poussoir
- Actionneurs : haut-parleur , vibreur
- Alimentation : Batterie Lithium-Ion rechargeable, 5000 mAh

2.3. Environnement logiciel

- OS : Linux (Ubuntu 18.1)
- Librairies : Jetson, DetectNet, PyTorch (YOLO), drivers capteurs
- Langages : Python
- Outils : VS Code, GitLab

3. Besoins fonctionnels

Chaque besoin est numéroté (ex : BF1, BF2, etc…)

Catégorie	ID	Besoin fonctionnel	Description	Commentaire
MUST HAVE	BF1	Détection d'obstacles	Le capteur ultrason doit mesurer la distance d'un objet et déclencher une alerte si celui-ci est trop proche.	Fonction cœur du système, base de sécurité.
	BF2	Annonce audio	Le système doit informer vocalement l'utilisateur de la présence et de la distance des objets.	Élément essentiel d'accessibilité.
	BF3	Détection et reconnaissance d'objets	Le module YOLO doit identifier les objets ou personnes visibles dans le champ de la caméra.	Permet la description de l'environnement.
	BF4	Démarrage automatique	Le système doit se lancer au démarrage de l'appareil sans intervention complexe.	Améliore l'ergonomie et la fiabilité.
SHOULD HAVE	BF5	Sélection du mode de fonctionnement	L'utilisateur doit pouvoir choisir entre les modes *Marche*, *Exploration* et *Mixte*.	Ajoute de la flexibilité selon la situation.
	BF6	Annonce directionnelle	Le système devrait préciser la direction de l'obstacle (gauche/droite/centre).	Apporte une meilleure précision d'usage.
COULD HAVE	BF7	Vibration en complément audio	Ajouter un retour haptique pour signaler les obstacles et/ou la sélection des modes.	Améliore l'accessibilité sans bruit.
	BF8	Enregistrement des logs	Stocker les détections et événements pour analyse.	Utile pour le débogage et la validation.
	BF9	Description textuelle de la scène	Décrire entièrement la scène dans le mode exploration.	Utile pour comprendre son environnement.
WON'T HAVE	BF10	Repenser la maquette	Complètement repenser la maquette, imprimer un support en 3D.	Hors du périmètre.
	BF11	Connexion réseau	Transmission des données à un serveur externe.	En dehors du périmètre PRI, complexité inutile.

3.1. MUST HAVE

3.1.1. BF1 : Détection d'obstacles

Objectif : Mesurer en continu la distance des obstacles devant l'utilisateur.

Composant concerné : Capteur ultrason A02YYUW.

Principe de fonctionnement :

- Le microcontrôleur envoie un signal trigger.
- Le capteur renvoie un signal echo dont la durée est proportionnelle à la distance.

Entrées / Sorties :

- Entrée : GPIO Trigger
- Sortie : GPIO Echo

3.1.2. BF2 : Annonce audio

Objectif : Informer vocalement l'utilisateur de la présence et de la distance des obstacles.

Composant concerné : Haut-parleur.

Principe de fonctionnement :

- Le système reçoit des messages texte (ex. "Obstacle à 1 mètre").
- Le module TTS (Text-to-Speech) convertit ce texte en audio (via Espeak FR).
- L'audio est diffusé immédiatement.

Contraintes techniques :

- Volume potentiellement ajustable.
- Vitesse de parole ajustable (< 1s entre détection et annonce).

Cas d'usage typiques :

- "Obstacle à 1,2 mètre devant."
- "Personne détectée à gauche."

3.1.3. BF3 : Détection et reconnaissance d'objets

Objectif : Identifier les objets dans le champ de vision de la caméra.

Composants concernés : Caméra IMX219-200, Jetson Nano (GPU), modèle YOLO.

Principe de fonctionnement :

- Capture d'image en continu ou frame by frame (à réestimer plus tard dans le projet).
- Traitement des frames via le modèle DetectNet ou YOLO (réseaux neuronaux pré-entraînés).
- Extraction des classes détectées (personne, voiture, chaise, etc.) et de leur position dans l'image.
- Transmission de la liste des objets détectés au module audio avec leur direction estimée.

Performances attendues :

- Taux de détection > 60 % dans un environnement intérieur normal.
- Frame rate minimum : 2/3 fps.

Interfaces logicielles :

- Python + DetectNet pour l'acquisition.
- PyTorch pour l'inférence YOLO.

3.1.4. BF4 : Démarrage automatique

Objectif : Le système doit se lancer automatiquement à l'allumage de la carte.

Principe de fonctionnement :

- Script startup.sh exécuté au boot (via systemd).

Critères de réussite :

- Aucun besoin d'intervention manuelle après mise sous tension.

3.2. SHOULD HAVE

3.2.1. BF5 : Sélection du mode de fonctionnement

Objectif : Permettre de choisir entre trois modes :

- **Marche** (ultrason seul)
- **Exploration** (YOLO seul)
- **Mixte** (ultrason + YOLO)

Implémentation technique :

- Commutateur physique bouton poussoir.
- Indication sonore lors du changement de mode "Mode Marche".

3.2.2. BF6 : Annonce directionnelle

Objectif : Préciser la direction (gauche, centre, droite) des obstacles détectés.

Principe de fonctionnement :

- Pour YOLO : calcul de la position.
- Génération d'un message vocal du type : "Obstacle à droite à 1,5 mètre."

3.3. COULD HAVE

3.3.1. BF7 : Vibration en complément audio

Objectif : Ajouter un retour haptique proportionnel ou non à la proximité d'un obstacle.

Matériel : Moteur vibreur.

Principe de fonctionnement :

- PWM contrôlée par GPIO selon distance mesurée.
- Fréquence ou intensité de vibration inversement proportionnelle à la distance.
- Vibration courte lors du changement de mode.

3.3.2. BF8 : Enregistrement des logs

Objectif : Sauvegarder les événements de détection pour analyse et validation.

Principe de fonctionnement :

- Création de fichiers .csv contenant :
  - Date/heure, type d'objet (ou pas d'objet), distance, direction, mode actif.
- Sauvegarde locale sur le disque de la Jetson Nano.

3.3.3. BF9 : Description textuelle de la scène

Objectif : Décrire tous les objets qui nous entourent de manière précise avec les différentes interactions entre eux s'il y en a.

Implémentation technique :

- Indication sonore de la scène.

3.4. WON'T HAVE

3.4.1. BF10 : Repenser la maquette

Aucune refonte complète du châssis ou impression 3D prévue.

Seules des améliorations mineures de fixation de capteurs / actionneurs sont prévues.

3.4.2. BF11 : Connexion réseau

Pas de communication avec serveur externe ni de cloud.

Le système reste 100 % local, sans dépendance réseau.

4. Interfaces

4.1. Interfaces matérielles

- Connexions : GPIO, USB, alimentation.

4.2. Interfaces logicielles

- Scripts Python / C++ → modules YOLO, capteurs, audio.
- Interface utilisateur (interaction avec le bouton poussoir).

4.3. Interface audio

- Messages types :
    - "Objet 2m."
    - "Personne 1."
    - "Obstacle 2m."
- Essayer de les raccourcir au minimum (ne pas spécifier l'unité et avoir toujours la meme référence, accélérer, …)

5. Tests de validation

Définir les tests attendus par fonctionnalité :

Test	Objectif	Méthode	Résultat attendu
T1	Vérifier détection ultrason	Approcher un obstacle	Détection < seuil + message audio
T2	Vérifier détection YOLO	Montrer objets connus	Reconnaissance correcte
T3	Vérifier annonces vocales	Simuler détection	Message vocal correct
T4	Test mixte	Activer mode mixte	Fusion cohérente des infos

6. Glossaire

- YOLO : You Only Look Once (algorithme de détection en temps réel)
- Jetson Nano : Carte de calcul Nvidia GPU embarquée
- HP : Haut-Parleur