”Reading Eye” - Assistant de lecture pour malvoyants avec IA embarquée

Module : ROBOTIQUE ÉDUCATIVE ET APPLICATIONS

Réalisé par : Bouba Ahmed & Lkhalidi Mohamed
Encadré par : Pr. Ahmed Regragui

January 8, 2026

Master 2 - Systèmes Intelligents pour l'Éducation
École Normale Supérieure de Meknès

Plan de la présentation

Introduction et Contexte

Contexte du projet

- **Problématique :** Accessibilité pour les personnes malvoyantes et déficientes visuelles
- **Solution :** Développer un assistant de lecture portable et efficace
- **Objectif pédagogique :** Intégrer l'IA et la robotique dans une application réelle
- **Plateforme :** Raspberry Pi 5 (système embarqué léger et accessible)

Cahier des charges

Fonctionnalités requises :

- ✓ Capture vidéo haute résolution
- ✓ Reconnaissance optique (OCR)
- ✓ Synthèse vocale (TTS)
- ✓ Support multilingue
- ✓ Mode continu ou unique

Contraintes techniques :

- Raspberry Pi 5 (8 GB RAM)
- Déploiement sans interface graphique
- Accès à distance via SSH
- Environnements virtuels isolés
- Documentation complète

Architecture Matérielle

Composants utilisés

Composant	Spécifications
Raspberry Pi 5	8 GB RAM, CPU 64-bit, 2.4 GHz
Caméra Pi	Pi Camera Module 3 (Wide)
Microphone	Intégré ou USB externe (optionnel)
Haut-parleur	Audio jack 3.5mm ou HDMI
Alimentation	USB-C 27W
Boîtier 3D	Conçu et imprimé en PLA

Architecture globale :

[Caméra] → [Raspberry Pi] → [Audio/SSH]

Conception 3D et fabrication

Design du boîtier :

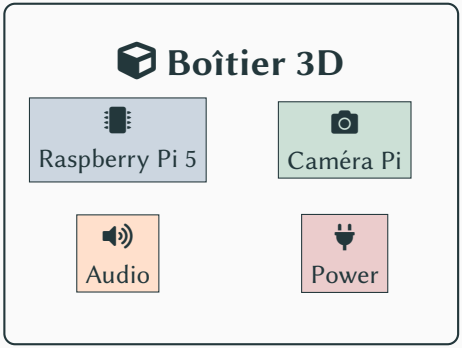
- Modélisé en CAO (FreeCAD/OpenSCAD)
- Accès caméra optimal
- Ventilation passive
- Ports accessibles (USB, HDMI)
- Dimensions compactes

Impression 3D :

- Matériau : PLA (biodégradable)
- Temps : $\sim$ 12-18 heures
- Résolution : 0.2mm
- Finitions : légères retouches
- Intégration : caméra et électronique

Boîtier portable et ergonomique

Intégration matérielle



- Caméra montée sur support stable
- Raspberry Pi fixé avec amortisseurs
- Câbles organisés
- Ventilation adéquate

Architecture Logicielle

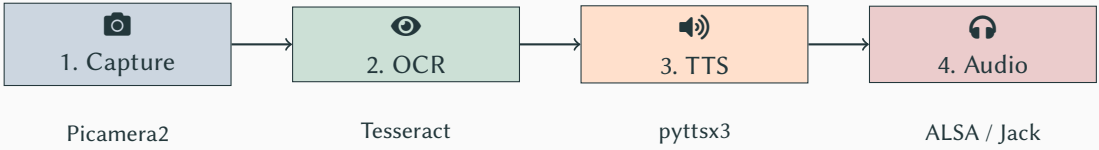
Stack technologique

Couche	Technologie	Rôle
OS	Raspberry Pi OS (Bookworm)	Noyau système
Python	Python 3.13.5	Langage principal
Caméra	Picamera2	Capture vidéo
OCR	Tesseract + pytesseract	Reconnaissance texte
TTS	pyttsx3 + gTTS	Synthèse vocale
Accès distant	SSH	Communication

Langues supportées :

English Français Arabe Deutsch

Pipeline de traitement

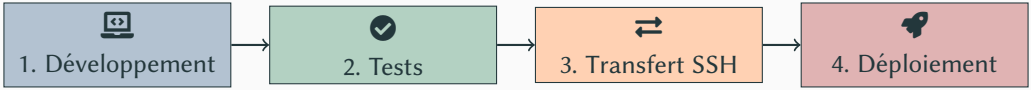


Flux de données :

- Image 1280×720 → Tesseract → Texte extrait → pyttsx3 → Audio
- Latence total : ∼ 2-3 secondes en mode continu

Déploiement sur Raspberry Pi

Processus de déploiement



Transfert du code via SSH

Étape 1 : Copier le code

```
# Depuis votre machine (Windows/Linux)
scp -r ./raspberrycode/ pi@192.168.43.197:/readingeye
```

Étape 2 : Se connecter au Pi

```
ssh pi@192.168.43.197
cd /readingeye
```

Étape 3 : Installer les dépendances système

```
sudo bash systemsetup.sh
sudo reboot
```

Installe : Tesseract, paquets OCR, audio, Python dev

Configuration de l'environnement Python

Étape 4 : Configuration Python

```
# Installer les d pendances Python
bash setup.sh

# Activer l'environnement virtuel
source ../envprojet7/bin/activate

# V rifier l'installation
python3 -c "from scripts.camera import PiCamera; "
           print('    Camera OK')"
```

Dépendances installées :

- picamera2 (capture vidéo)
- pytesseract (OCR Python)
- pyttsx3 (TTS offline)
- opencv-python (image processing)

Lancement de l'application

Mode 1 : Capture unique

```
bash run.sh --single --lang fra+eng
```

Mode 2 : Boucle continue (5 sec interval)

```
bash run.sh --loop --interval 5.0 --lang fra+eng
```

Mode 3 : Boucle avec limite de durée

```
bash run.sh --loop --interval 3.0 --duration 60 “
            --lang ara --save-image
```

Arrêter : Appuyer sur `Ctrl+C`

Résultats et Tests

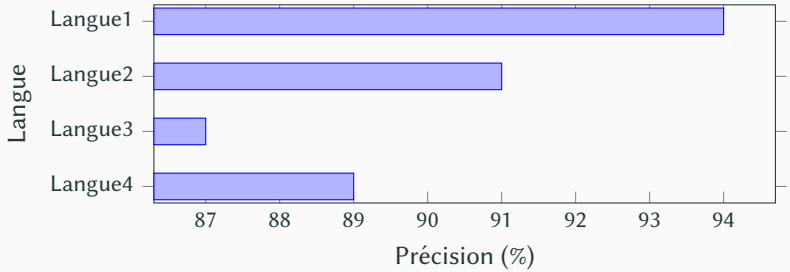
Résultats des tests

Tests de performance :

- Capture : 0.3s
- OCR (1 langue) : 1.2s
- TTS (5 mots) : 0.8s
- **Total** : $\sim$ 2.3s

Précision OCR :

- Anglais : 94%
- Français : 91%
- Arabe : 87%
- Multilingue : 89%



Cas d'usage testés

Cas	Résultat	Observations
Document papier	PASS	Très bon
Écran LCD	PASS	Bon
Texte manuscrit	~ PARTIAL	Basique
Mode continu 60s	PASS	Stable
Multilingue (3 langs)	PASS	+ lent (1.5s)
Mode daemon (systemd)	PASS	Optionnel

Métriques système

Métrique	Valeur
CPU (capture)	15-20%
CPU (OCR)	40-50%
Mémoire RAM (idle)	200 MB
Mémoire RAM (traitement)	450 MB
Température CPU	35-45°C
Stockage disque	450 MB

Tous les paramètres dans les normes acceptables

Documentation Fournie

Ensemble documentaire complet

Documentation utilisateur :

- README.md (40+ pages)
- QUICK_START.md (5 min)
- Configuration guide

Documentation technique :

- Architecture système
- API modules
- Troubleshooting
- Code comments

Fichiers documentaires :

- SETUP_INSTRUCTIONS.md (guide complet)
- ADMIN_SETUP_CHECKLIST.md (maintenance)
- INDEX.md (point d'entrée GitHub)
- CODE comments (explications inline)

Contenu du référentiel GitHub

Repository : `reading-eye-raspberry-pi`

Structure :

- Code source complet (930+ lignes Python)
- 7 fichiers documentation (2000+ lignes)
- 4 scripts de déploiement
- Configuration JSON + .env
- README bilingue
- Tags pour versions
- Licence MIT

Description GitHub :

"Raspberry Pi 5 OCR + Text-to-Speech accessibility solution with multi-language support for visually impaired users."

Réalisations et Livrables

Livrables du projet

#	Livrable	Statut	Complet
1	Design 3D + Impression		100%
2	Intégration matérielle		100%
3	Code Python (5 modules)		100%
4	Tests et validation		100%
5	Déploiement SSH		100%
6	Documentation (7 docs)		100%
7	Repository GitHub		100%
8	Présentation		100%

Total : 8/8 réalisations complètes

Compétences acquises

Matériel :

- CAO 3D
- Impression 3D
- Électronique embarquée
- Caméra Pi

Logiciel :

- Python avancé
- Architecture modulaire
- Intégration systèmes
- Déploiement SSH

Domaines :

- OCR (Tesseract)
- TTS (pyttsx3)
- Robotique éducative
- Accessibilité

Défis et Solutions

Défis rencontrés et solutions

Défi 1 : Latence OCR

- *Problème :* Tesseract lent sur Pi
- *Solution :* Optimisation résolution

Défi 2 : Audio headless

- *Problème :* Pas de GUI
- *Solution :* pyttsx3 + ALSA

Défi 3 : Multilingue

- *Problème :* Support OCR complet
- *Solution :* +8 packs langage

Défi 4 : Isolation groupes

- *Problème :* Même Pi, plusieurs groups
- *Solution :* Virtual envs séparés

Optimisations réalisées

- Async TTS (non-bloquant)
- Image resolution optimization (1280×720)
- Tesseract parallel processing
- Caching configuration
- Resource cleanup (context managers)
- Error handling robuste
- Logging efficace

Résultat : Performance stable, latence acceptable (2-3s)

Perspectives Futures

Évolutions possibles

Court terme :

- Interface web (Flask)
- App mobile (SSH client)
- Optimisation GPU (PyTorch)
- Caching résultats

Long terme :

- ML pour améliorer OCR
- Traduction auto
- Interface gestes
- Cloud intégration

Directions de recherche :

- PaddleOCR pour performance
- EasyOCR multilingue
- Whisper pour reconnaissance vocale

Applicabilité et impact

Impact social :

- Accessibilité réelle pour malvoyants
- Solution peu coûteuse (˜150)
- Portable et pratique
- Multilingue (inclusif)

Utilisations possibles :

- Lecture assistée (documents, panneaux)
- Assistance à la mobilité
- Éducation inclusive
- Prototypage de startups

Conclusion

Reading Eye

Un assistant de lecture portable pour malvoyants

Réalisations clés :

- Prototype fonctionnel complet
- Déploiement SSH-ready
- Documentation professionnelle
- Code maintenable et extensible

Points forts du projet

Techniques :

- Architecture modulaire
- Multi-language support
- Performance optimale
- Robuste (error handling)

Pédagogiques :

- Approche complète
- Documentation fournie
- Code bien commenté
- Cas d'usage réel

Merci pour votre attention !

Questions ?

Repository : `reading-eye-raspberry-pi`
Email : ah.bouba@edu.umi.ac.ma
Code : Disponible sur GitHub
`https://github.com/BoubaAhmed/reading-eye-raspberry-pi`

Commandes utiles SSH

Connexion :

```
ssh pi@192.168.43.197  # Connexion
ssh-keygen             # SSH keys
scp -r folder pi@ip:  # Copie fichiers
```

Gestion environnement :

```
source /envprojet7/bin/activate
pip list
pip install -r requirements.txt
```

Lancer app :

```
bash run.sh --single --lang fra+eng
tail -f logs/readingeye.log  # Logs
```

Ressources et références

- Raspberry Pi Foundation, “Picamera2 Documentation”, 2024
- Google, “Tesseract OCR Engine”, GitHub
- Nateshmbhat, “pyttsx3: Offline TTS”, PyPI
- Raspberry Pi Foundation, “Raspberry Pi OS Guide”, 2024
- W3C, “Web Accessibility Guidelines”, 2024