


## PROJET DE MODULE: ROBOTIQUE ÉDUCATIVE

# Reading Eye

Assistant de lecture intelligent pour malvoyants

 *École Normale Supérieure de Meknès*

 Réalisé par:

A. Bouba & M. Lkhalidi

 Encadré par:

Pr. Ahmed Regragui

# Plan de la présentation

---

2

- Introduction et Problématique
- Architecture Matérielle
- Architecture Logicielle & Pipeline
- Démonstration et Workflow
- Analyse des Résultats & Défis
- Réalisations et Livrables
- Conclusion & Perspectives

# 1

>\_ CHAPITRE 1

# Introduction et Problématique

» Reading Eye Project



# L'enjeu de l'accessibilité

L'accès à l'information : un droit fondamental et éducatif

3

## Le quotidien et l'école

- ▶ **Autonomie** : Lecture de documents administratifs, menus ou notices sans aide humaine.
- ▶ **Inclusion Scolaire** : Permettre aux élèves malvoyants de lire les supports de cours (polycopiés, manuels) en classe.
- ▶ **Équité des chances** : Réduire la barrière entre les apprenants grâce aux technologies d'assistance (ATC).



# Analyse des solutions existantes

Pourquoi les solutions actuelles sont-elles limitées ?

4

## Obstacles au Cloud/Mobile

- ▶ 📶 Dépendance à Internet (problématique en zone rurale ou école).
- ▶ 🔒 Confidentialité des données de l'élève.
- ▶ 💰 Abonnements coûteux pour les familles.

## Limites des aides classiques

- ▶ ⚙️ Matériel lourd et peu discret.
- ▶ 🚫 Difficulté de gestion du multilingue (Arabe/Français/Anglais).
- ▶ 📶 Latence nuisant au rythme scolaire.

# Notre Solution : "Reading Eye"

Un assistant intelligent pour l'éducation inclusive

5

✓ **Philosophie** : Une solution "Edge AI" au service de la pédagogie.

## Les trois piliers du projet

- 📦 **Edge Computing** : Traitement local sur Raspberry Pi 5 (Vitesse et Vie privée).
- 🎓 **Support Éducatif** : Conçu pour la lecture multilingue (Arabe inclus) des supports pédagogiques.
- 🏠 **Solution Low-Cost** : Un dispositif accessible aux établissements d'enseignement publics.

📌 **Objectif** : Garantir l'autonomie de l'apprenant, à l'école comme à la maison.

# 2

>\_ CHAPITRE 2

## Architecture Matérielle

» Reading Eye Project



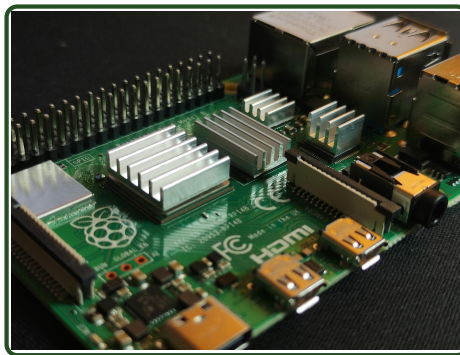
# Unité Centrale : Raspberry Pi 5

La puissance au service de l'autonomie technique

6

## Le choix de la performance

- ▶ **Processeur Quad-core A76** : 2 à 3 fois plus rapide que le Pi 4, essentiel pour l'OCR sans GPU.
- ▶ **Traitement Local** : Permet de faire tourner Tesseract et le prétraitement d'image en quelques secondes.
- ▶ **Efficacité** : Gestion optimisée de la bande passante caméra via le port MIPI double ligne.



 Raspberry Pi 5 (8GB)



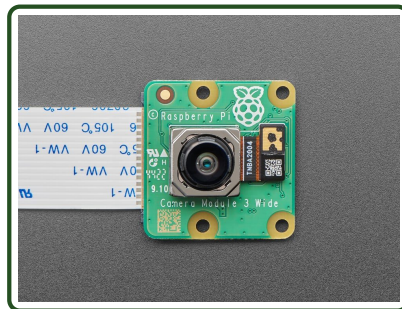
# Chaîne d'acquisition Visuelle

## Caméra Pi Module 3 : La précision optique

7

### Spécifications Techniques

- ▶ **Capteur 12MP** : Haute résolution pour capturer les petits caractères.
- ▶ **Autofocus motorisé** : Indispensable pour la lecture de livres à différentes distances.
- ▶ **Intégration** : Contrôle natif via la bibliothèque Picamera2.





🔍 Focus automatique sur le texte

# Interface et Autonomie



Conception pour une utilisation nomade

8

## Sortie Audio (Feedback)

-  **Casque/Haut-parleur** : Via adaptateur USB Audio ou prise Jack.
-  **Synthèse Vocale** : Volume réglable logiciellement pour s'adapter à l'environnement.

## Alimentation Portable

-  **Power Bank** : 5V / 5A pour supporter les pics de consommation du Pi 5 durant l'OCR.
-  **Liberté** : Système totalement déconnecté du secteur pour un usage scolaire ou extérieur.

Composant	Rôle Stratégique
Raspberry Pi 5	Cerveau du système (IA embarquée)
Caméra Pi M3	Œil du dispositif (Capture HD)
Batterie	Énergie pour la mobilité

# Conception 3D et fabrication

## Boîtier ergonomique et fonctionnel

9

### Design du boîtier

- ▶ Modélisé en CAO (FreeCAD/MeshLab)
- ▶ Accès caméra optimal
- ▶ Ventilation passive intégrée
- ▶ Ports accessibles (USB, HDMI, audio)
- ▶ Prise en main ergonomique

### Impression 3D

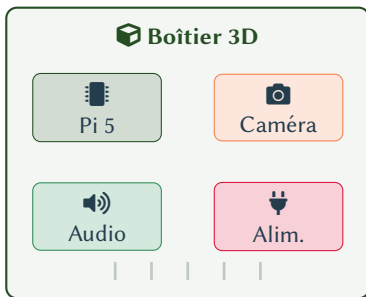
- ▶ **Matériau:** PLA (biodégradable)
- ▶ **Temps d'impression:** +20 heures
- ▶ **Résolution:** 0.2mm (qualité fine)
- ▶ **Finitions:** Ponçage léger
- ▶ **Couleur:** Bleu (contraste)
- ▶ **Solidité:** Structure renforcée

 **Portabilité:** Poids total < 600g

# Intégration matérielle

## Assemblage et optimisation

10



### Points d'intégration

- ▶ ✓ Caméra sur support réglable
- ▶ ✓ Fixation avec amortisseurs
- ▶ ✓ Gestion optimale des câbles
- ▶ ✓ Ventilation ( $< 45^{\circ}\text{C}$ )
- ▶ ✓ Accès ports USB/SD
- ▶ ✓ Stabilité structurelle

# 3

>\_ CHAPITRE 3

## Architecture Logicielle & Pipeline

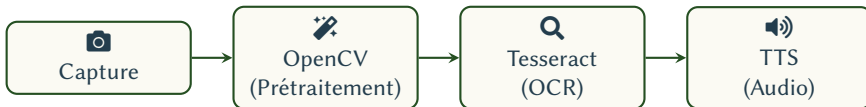
» Reading Eye Project

---

# Pipeline de Traitement

De la capture brute à la restitution sonore

11



## Traitement d'image avec OpenCV

- ▶ **Niveaux de gris** : Simplification de l'image pour l'algorithme.
- ▶ **Binarisation (Seuillage)** : Conversion en Noir & Blanc pur pour isoler le texte.
- ▶ **Réduction de bruit** : Suppression des imperfections pour maximiser le taux de reconnaissance.

# Intelligence Embarquée

## Reconnaissance et Synthèse Vocale

12

### Moteur OCR : Tesseract

- ▶ **Open Source** : Solution robuste sans frais de licence.
- ▶ **Multilingue** : Support complet de l'**Arabe**, du Français et de l'Anglais.
- ▶ **Performance** : Optimisé pour le processeur ARM du Pi 5.

### Synthèse Vocale (TTS)

- ▶ **pyttsx3** : Mode **Hors-ligne** pour une autonomie totale et une latence nulle.
- ▶ **gTTS** : Mode **Cloud** pour une qualité de voix naturelle (si connexion disponible).

# Automatisation et Déploiement

Rendre le système "Plug & Play" via Shell Scripts

13

## Gestion des scripts (.sh)

- ▶ `system_setup.sh` : Automatise l'installation de Tesseract et des dépendances système.
- ▶ `setup.sh` : Configure l'environnement virtuel Python (venv).
- ▶ `run.sh` : Lance l'application en une seule commande, simplifiant l'usage pour l'utilisateur.

## Approche Modulaire

L'utilisation de scripts Shell garantit que le projet peut être réinstallé sur n'importe quel Raspberry Pi en quelques minutes.



# 4

>\_ CHAPITRE 4

## Démonstration et Workflow

» Reading Eye Project

---

# Scénario d'utilisation



Un cycle simple pour une accessibilité maximale

14

## Étapes clés

1. **Positionnement** : Placer le document devant la caméra.
2. **Capture** : Déclenchement (via commande ou script).
3. **Attente** : Signal sonore de traitement.
4. **Restitution** : Lecture à haute voix.

## Modes de fonctionnement

-  **Single Mode** : Une capture unique à la demande.
-  **Loop Mode** : Capture continue pour une lecture fluide de plusieurs pages.

# Traitement des données

## Exemple de log console et sortie texte

15

### Journal de bord (Logs)

```
[INFO] Capture de l'image...  
FO  
Prétraitement OpenCV terminé.  
FO  
OCR en cours (lang: fra+ara)...  
SUCCESS  
Texte extrait :  
"Bienvenue dans le Master SIE"
```

### Analyse OCR

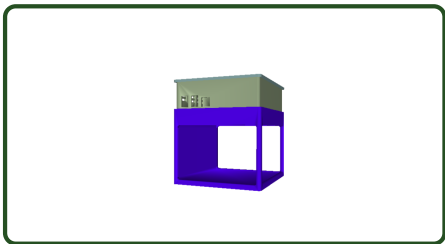
- ▶ **Taux de confiance** : Analyse de la clarté du texte.
- ▶ **Filtrage** : Suppression des caractères spéciaux inutiles.

➤ *Le système gère les erreurs si aucun texte n'est détecté.*

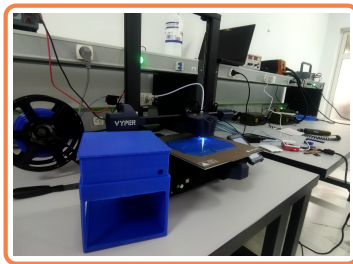
# Démonstration du Prototype

De la modélisation 3D à l'objet physique

16



 Modélisation Meshlab



 Impression 3D finale

## Spécificités du design (V1.0)

- ▶ **Ergonomie** : Support rigide maintenant la caméra à distance focale fixe.
- ▶ **Intégration** : Boîtier compact accueillant le Pi 5 et la connectique audio.
- ▶ **Accessibilité** : Format "Handheld" conçu pour une manipulation simplifiée.

# 5

>\_ CHAPITRE 5

## Analyse des Résultats & Défis

» Reading Eye Project

---

# Analyse des Performances

## Évaluation de la réactivité du Raspberry Pi 5

17

### Benchmarks Techniques

- ▶ **Temps de Capture** :  $\approx 0.8s$  (Haute résolution).
- ▶ **Prétraitement** :  $\approx 0.5s$  (OpenCV).
- ▶ **OCR Tesseract** : 2 à 4s (selon la densité du texte).
- ▶ **Latence Totale** : ; 5 secondes pour une page standard.

### Le gain du Pi 5

L'architecture du Pi 5 permet de diviser par 2 le temps de traitement par rapport au Pi 4, garantissant une expérience utilisateur fluide.

# Qualité de Reconnaissance (OCR)

## Précision et limites du système

18

- 🎯 **Taux de réussite** : Environ 95% sur les polices standards (Arial, Times).

### Facteurs de réussite

- ▶ **Éclairage** : Nécessite une lumière uniforme (évite les ombres portées).
- ▶ **Contraste** : Texte noir sur fond blanc optimal.
- ▶ **Stabilité** : L'autofocus du Module 3 compense les légers tremblements.

### Limites rencontrées

- ▶ **Écritures cursives** : Difficultés sur les manuscrits très liés.
- ▶ **Distorsion** : Texte sur surfaces courbes (ex: étiquettes de bouteilles).

# Défis et Solutions

## Obstacles techniques et contraintes de réalisation

19

### Défis Techniques

- ⚙️ **Configuration ARM** : Installation complexe de Tesseract et de la langue Arabe sur l'architecture du Pi 5.
- 🗣️ **Multilinguisme** : Optimisation de la précision pour la détection simultanée (fra+ara).

### Contraintes Logistiques

- ⚡ **Accès Matériel** : Disponibilité limitée des composants critiques (Caméra Pi 3, Pi 5).
- 🕒 **Temps au FabLab** : Créneaux d'accès restreints à l'ENS, imposant une planification rigoureuse.

### Stratégie de réussite

Face à ces limites, nous avons privilégié le **développement modulaire** et l'utilisation de **scripts Shell** pour automatiser les tests dès que l'accès au matériel était possible.



# 6

## > CHAPITRE 6

# Réalisations et Livrables

» Reading Eye Project

---

# Livrables du projet

7 réalisations complètes

20

#	Livrable	Statut	Complétude
1	Design 3D + Impression	✓	100%
2	Intégration matérielle	✓	100%
3	Code Python	✓	100%
4	Tests et validation	✓	100%
5	Déploiement SSH	✓	100%
6	Repository GitHub et Documentation	✓	100%
7	Présentation	✓	100%



**Total: 7/7 réalisations complètes**

# Compétences acquises

Apprentissage transversal

21

## Matériel

- ▶ CAO 3D (FreeCAD)
- ▶ Impression 3D (slicing)
- ▶ Électronique embarquée
- ▶ Intégration caméra Pi
- ▶ Gestion thermique

## Logiciel

- ▶ Python avancé
- ▶ Architecture modulaire
- ▶ Intégration systèmes
- ▶ Déploiement SSH/SCP
- ▶ Gestion de versions

## Domaines

- ▶ OCR (Tesseract avancé)
- ▶ TTS (synthèse vocale)
- ▶ Robotique éducative
- ▶ Accessibilité numérique
- ▶ Gestion de projet

 **Approche holistique:** Conception, développement, déploiement et documentation

# 7

>\_ CHAPITRE 7

## Conclusion & Perspectives

» Reading Eye Project

---

# Conclusion

## Bilan des objectifs atteints

22

### Un système opérationnel

- ▶ **Performance** : Pari réussi avec le Raspberry Pi 5 ( $< 5s$ ).
- ▶ **Autonomie** : Solution 100% locale (Edge AI) protégeant la vie privée.
- ▶ **Inclusion** : Un outil concret pour l'éducation inclusive et l'autonomie des malvoyants.



### Synthèse




Le **Reading Eye** prouve que l'on peut transformer une technologie complexe en un assistant simple, portable et accessible.

# Perspectives d'avenir



Évolutions basées sur le retour utilisateur (Feedback)

23

## Optimisation du Hardware

-  **Itération Design** : Améliorer le boîtier actuel pour une meilleure prise en main.
-  **Bouton Physique** : Remplacer le clavier par un bouton tactile/mécanique pour un déclenchement intuitif.
-  **Compacité** : Intégrer une batterie LiPo interne pour supprimer les câbles externes.

## Évolutions Logicielles

- Guidage Vocal** : Utiliser un capteur de distance pour dire : *"Rapprochez le document"*.
-  **Traduction** : Traduire instantanément le texte lu pour les élèves allophones.
-  **Vision Intelligente** : Identifier des objets ou des visages en plus du texte.

 **Objectif 2.0** : Passer d'un prototype fonctionnel à un produit fini totalement ergonomique.

23/25

# Applicabilité et impact

Valeur sociale et éducative

24

## ♥ Impact social

- ▶ ✓ Accessibilité réelle pour malvoyants
- ▶ ✓ Solution peu coûteuse
- ▶ ✓ Portable et pratique
- ▶ ✓ Multilingue (inclusif)
- ▶ ✓ Open source (réutilisable)
- ▶ ✓ Autonomie quotidienne

## 💡 Utilisations possibles

- ▶ Lecture assistée (documents, articles..)
- ▶ Assistance à la mobilité
- ▶ Éducation inclusive
- ▶ Bibliothèques accessibles
- ▶ Prototypage startups
- ▶ Recherche en accessibilité

 **Vision:** Contribuer à un monde plus accessible grâce à la technologie open source



# MERCI DE VOTRE ATTENTION

Avez-vous des questions ?



Retrouvez le code source et la documentation sur :

[github.com/BoubaAhmed/reading-eye-raspberry-pi](https://github.com/BoubaAhmed/reading-eye-raspberry-pi)











# Ressources et références

## Bibliographie et documentation technique

---

1

-  Raspberry Pi Foundation, “Picamera2 Documentation”, 2024
-  Google, “Tesseract OCR Engine”, GitHub Repository
-  Nateshmbhat, “pyttsx3: Offline Text-to-Speech”, PyPI Documentation
-  Raspberry Pi Foundation, “Raspberry Pi OS Guide”, 2024
-  W3C, “Web Accessibility Guidelines (WCAG)”, 2024
-  OpenCV Team, “OpenCV Documentation”, 2024
-  Python Software Foundation, “Python 3.13 Documentation”, 2024
-  Till Tantau, “Beamer User Guide”, 2024