



RÉPUBLIQUE TUNISIENNE  
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE  
UNIVERSITÉ DE TUNIS EL MANAR  
ECOLE NATIONALE D'INGÉNIEURS DE TUNIS



# Systeme d'Accès Parking Intelligent

**Élaboré par :** Dkhil Alaa & Hentati Farah

**Encadré par :** Mr. Khaled Jelassi

**3AGE 1**

Année Universitaire : 2025/2026

# PLAN

---

1.INTRODUCTION

2. Objectifs du Système

3. L'Écosystème Technologique

4. Le Processus Opérationnel

5.CONCLUSION

# Introduction

---

- Dans un monde où l'efficacité opérationnelle est devenue primordiale, les entreprises font face à un défi majeur : sécuriser et fluidifier l'accès à leurs parkings. Les méthodes traditionnelles manquent souvent de réactivité et nécessitent une présence humaine constante.
- Ce projet propose une solution innovante : un Système d'Accès Parking Intelligent qui réalise un véritable voyage de la détection à la décision. Notre vision repose sur la conception d'un système hybride capable d'identifier et d'autoriser les véhicules en temps réel, sans intervention humaine. En combinant la réactivité d'un microcontrôleur STM32 avec la puissance de calcul d'un PC, nous visons à garantir une fluidité de trafic optimale tout en assurant une sécurité proactive

# Le Défi : Sécuriser et Fluidifier l'Accès Parking

- Les entreprises exigent des systèmes d'accès à la fois **sécurisés** et **automatisés** pour gérer efficacement leurs parkings.
- Le défi est de créer une solution qui contrôle l'accès sans intervention humaine, garantissant une fluidité optimale du trafic.



## Sécurité

Prévenir les accès non autorisés de manière proactive et fiable.



## Fiabilité

Assurer un fonctionnement continu et résister aux perturbations environnementales.



## Rapidité

Minimiser le temps d'attente pour les employés afin d'optimiser la fluidité.

# Notre Vision : Un Système Intégré pour Détecter, Identifier et Agir

---

L'objectif est de concevoir un système hybride qui identifie et autorise les véhicules en temps réel en combinant microcontrôleur et puissance PC.

## 01

### DÉTECTER

Utilisation d'un capteur ultrasonique piloté par STM32 pour repérer la présence d'un véhicule à l'entrée.

## 02

### IDENTIFIER

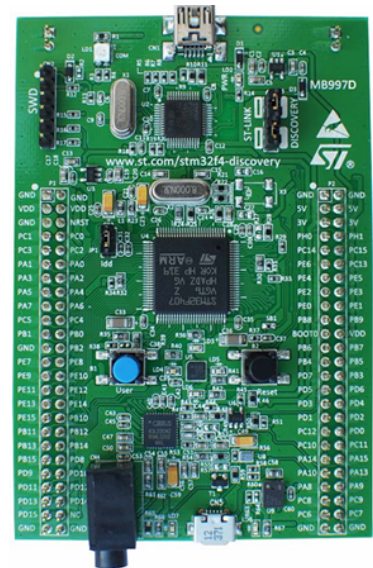
Capture d'image via Webcam PC, extraction du numéro d'immatriculation via OCR local haute performance.

## 03

### AGIR

Vérification instantanée dans la base de données Firebase et commande de l'ouverture de la barrière par le STM32.

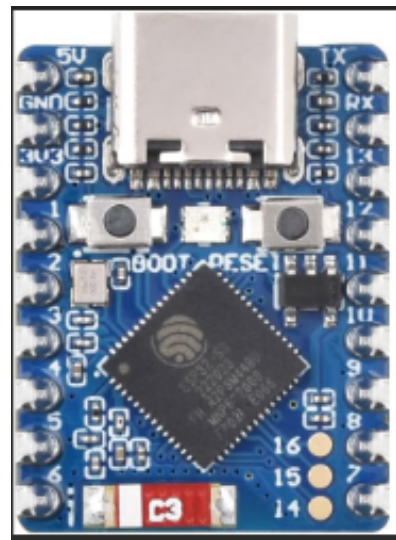
# L'Écosystème Technologique : L'Union du Matériel et du Logiciel



**STM32F407**

**Le Cerveau Temps Réel**

OS : FreeRTOS  
Gère la réactivité critique du système.  
Capteurs : HC-SR04  
Actionneurs : SG90



**ESP32S3**

**Le pont entre les 2 systèmes**

-Reçoit l'ordre de capture du stm32  
-Reçoit la réponse du serveur Python



**Webcam PC**

**L'Oeil du système**

Capture d'image haute résolution.

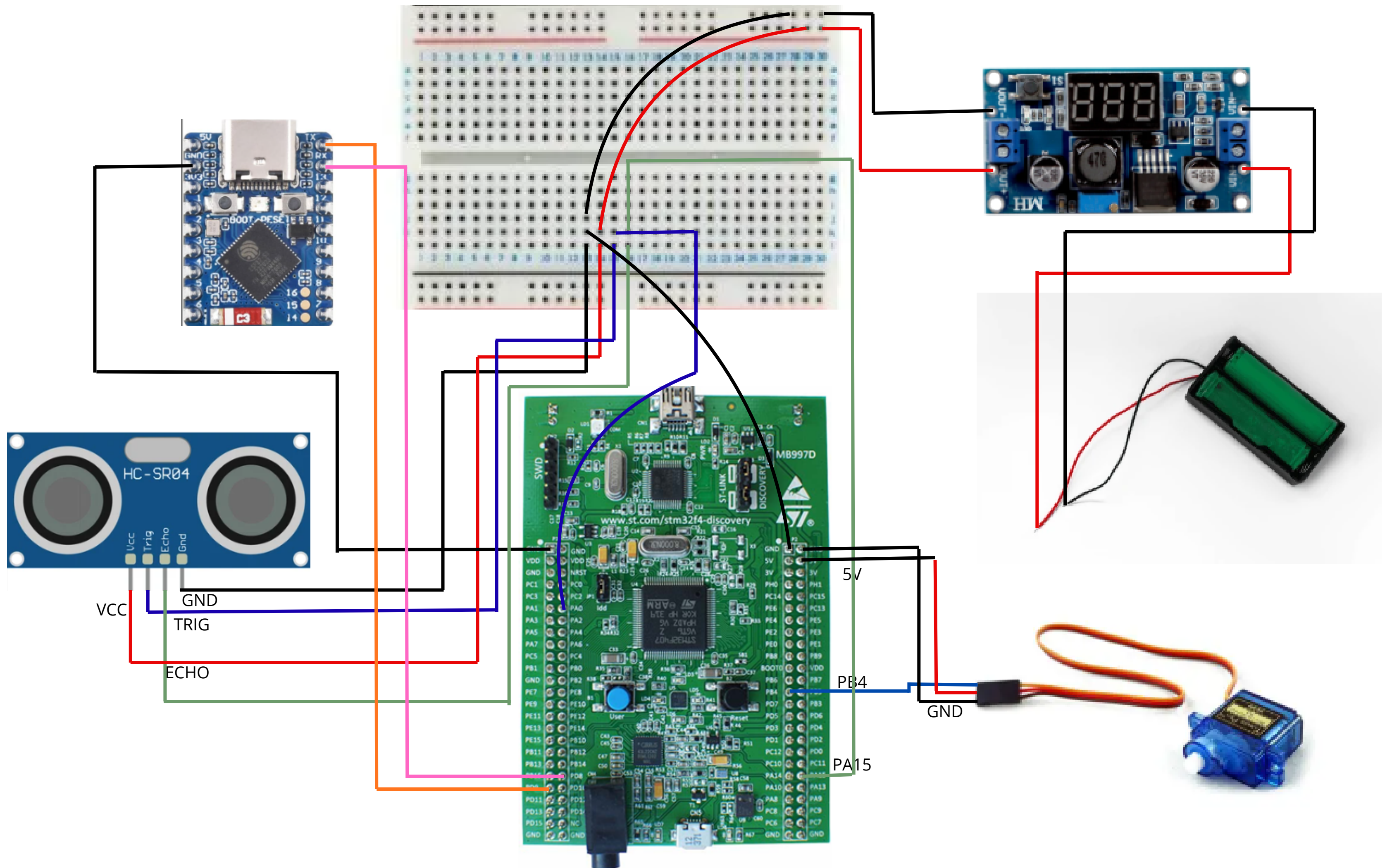


**Python & Firebase**

**L'Intelligence Centrale**

Centre de calcul local et cloud.  
OCR : EasyOCR (Local)  
Vision : OpenCV  
Base de données  
Firebase





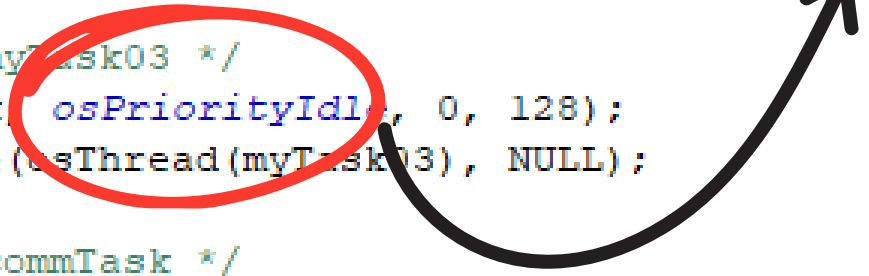
# Les tâches FreeRTOS de STM32:

## 3 tâches principales:

```
/* definition and creation of myTask02 */
osThreadDef(myTask02, UltrasonTask, osPriorityNormal, 0, 256);
myTask02Handle = osThreadCreate(osThread(myTask02), NULL);

/* definition and creation of myTask03 */
osThreadDef(myTask03, ServoTask, osPriorityIdle, 0, 128);
myTask03Handle = osThreadCreate(osThread(myTask03), NULL);

/* definition and creation of commTask */
osThreadDef(commTask, CommunicationTask, osPriorityNormal, 0, 256);
commTaskHandle = osThreadCreate(osThread(commTask), NULL);
```



Priorité la plus basse

### ✓ UltrasonTask:

Détection véhicule

- Lecture toutes les 500ms.
- Si < 20cm : Capture

### ✓ Communication Task:

Dialogue ESP32

- Envoie
- Reçoit JSON
- Parse

### ✓ ServoTask:

Contrôle barrière

- Ouvre si autorisé
- Attend départ (3s)
- Ferme après départ



# Tâche 1 : La Détection de véhicule

## CAPTEUR HC-SR04



## Principe de Fonctionnement

- Le STM32 envoie une impulsion de **10 $\mu$ s** sur la broche TRIGGER (PA0) et mesure la durée du retour sur la broche ECHO (PA15) via le timer **TIM2**.

## Logique de Déclenchement

- Le processus d'identification est initié lorsque la distance calculée est inférieure au seuil critique de **20 cm**.
- Ce seuil garantit que le véhicule est correctement positionné pour la capture par la webcam.

# Tâche 2 : Communication avec ESP32 : Appel à l'Intelligence

## Synchronisation Matériel-Logiciel

### Liaison Série UART

- Communication directe via **USART3** configurée à **115200 bauds**.
- STM32 (PD8/PD9) ↔ ESP32-S3 (GPIO43/44)
- ESP32-S3 agit comme pont WiFi

### Commande "CAPTURE"

Dès que le véhicule est détecté, la CommunicationTask envoie la commande "Capture" via UART3.

STM32 → ESP32 : UART

ESP32 → Serveur.py : HTTP

### Rôles des Modules

#### STM32F407: Contrôleur temps-réel

- Détection, communication et barrière

#### ESP32S3: Pont WiFi + Client HTTP

UART <-> HTTP

#### Serveur Python: Partie IA

- Webcam + OCR
- Firebase

# L'Appel à l'Intelligence : Le STM32 Notifie le serveur

## Communication entre STM32 et ESP32S3:

```
recette reponse ESP32...
>>> Demande capture ESP32...
Attente reponse...
Reponse ESP32: E{"authorized":true,"plate":"1234TU5678","success":true}

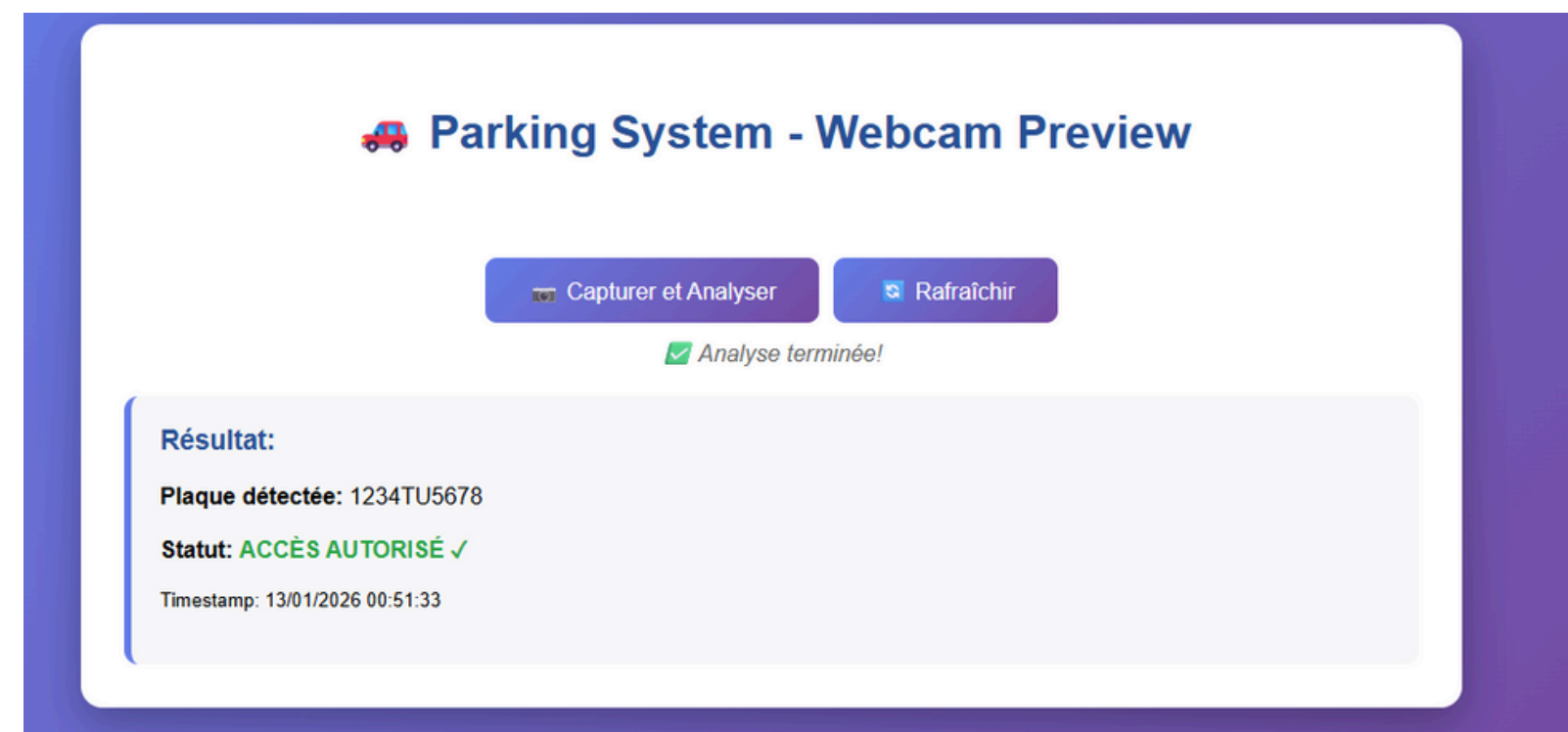
Matricule: 1234TU5678
>>> ACCES AUTORISE <<<

*** BARRIERE OUVERTE ***
Matricule: 1234TU5678
```

```
>>> RX FROM STM32: CAPTURE
=== Processing Capture Request ===
Requesting: http://192.168.247.162:5000/capture
HTTP Response Code: 200
Response: {"authorized":true,"plate":"1234TU5678","success":true}

>>> TX TO STM32: {"authorized":true,"plate":"1234TU5678","success":true}
```

## Résultat OCR obtenu sur: [http://127.0.0.1:5000/video\\_feed](http://127.0.0.1:5000/video_feed)



# L'Appel à l'Intelligence : Le STM32 Notifie le serveur

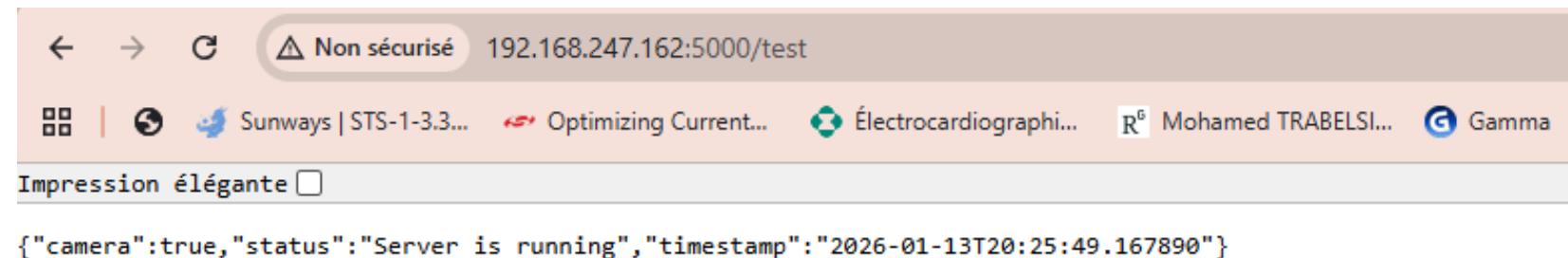
Serveur HTTP (Flask sur port 5000):

```
=====
PARKING SYSTEM - Python OCR Server
=====
Server starting on http://0.0.0.0:5000
Endpoints:
- GET / : Webcam preview page
- GET /video_feed : Video stream
- GET /capture : Capture image and run OCR
- GET /test : Test if server is running

🌐 Open your browser: http://localhost:5000
=====

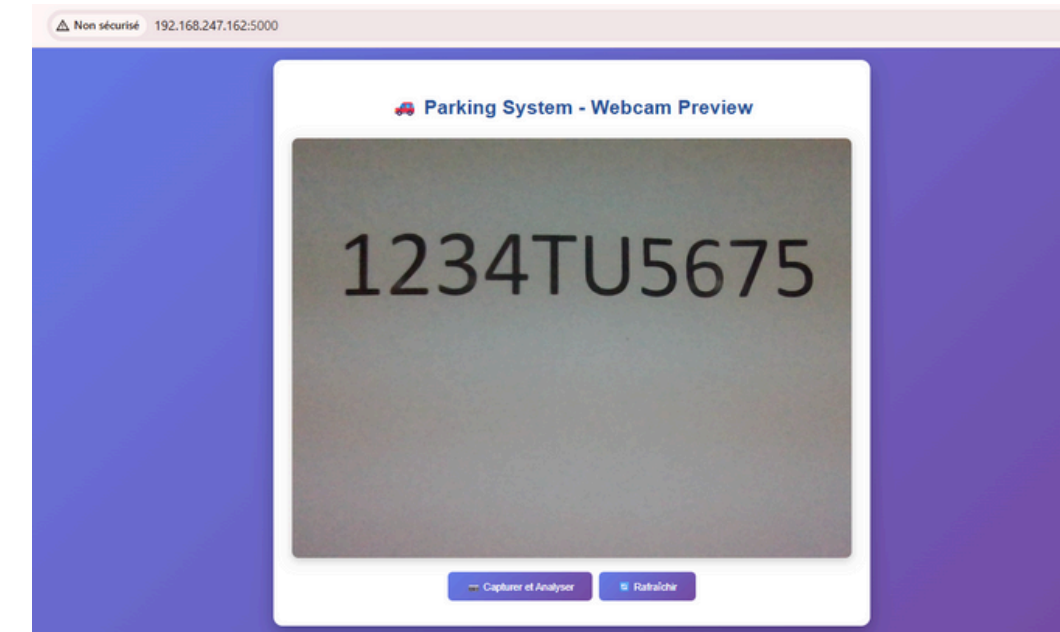
* Serving Flask app 'ocr_server'
* Debug mode: off
- - -
```

<http://127.0.0.1:5000/test>:



The screenshot shows a web browser window with the address bar displaying '192.168.247.162:5000/test'. The page content shows a JSON response: `{"camera":true,"status":"Server is running","timestamp":"2026-01-13T20:25:49.167890"}`. The browser's developer tools are open, showing the network tab with the request and response details.

[http://127.0.0.1:5000/video\\_feed](http://127.0.0.1:5000/video_feed):



<http://127.0.0.1:5000/capture>:

```
=== CAPTURE REQUEST ===
Image captured: (480, 640, 3)
Saved: captured_images/original_20260113_202658.jpg
Saved processed: captured_images/processed_20260113_202658.jpg
Running OCR...
  Detected: '1234TU5675' (confidence: 0.97)
✓ Final plate: '1234TU5675'
Plate '1234TU5675' not found in database
Authorization check: False
```

# Traitement Local : Le Prétraitement, OCR Précis

Avant la reconnaissance, le script Python local applique une série de traitements pour augmenter le contraste. **Cette étape triple le taux de reconnaissance.**



---

Ces étapes transforment une image brute en une donnée exploitable pour l'algorithme de reconnaissance local.



# Traitement Local

## 1. Extraction OCR Locale

`reader.readtext(enhanced)` s'exécute sur le PC, offrant une vitesse de traitement supérieure pour l'extraction du texte.

## 2. Nettoyage & Formatage

Normalisation instantanée des données.

Ex: "1234-TU-5678" → **"1234TU5678"**

## 3. Vérification Firebase

Le script Python interroge la base de données en temps réel pour valider l'accès du véhicule identifié.

## 4. Décision & Transmission

Le PC génère le verdict **Boolean** et le transmet au STM32 via le port série.



# Tâche 3 : Contrôle de barrière

Servo-moteur SG90



Attente

Ouverture: PlateAuthorized → PWM 2000 us

Fermeture: Attente 3s → PWM 1000 us

# Conclusion

- Le système présenté démontre l'efficacité de l'union entre le matériel et le logiciel pour répondre à des problématiques de sécurité complexes. Grâce à l'utilisation d'un capteur ultrasonique HC-SR04 pour la détection précise à moins de 20 cm et d'un algorithme OCR local haute performance via Python, le système assure un traitement rapide et fiable des données.
- En conclusion, cette architecture permet de transformer une simple image capturée par une webcam en une décision d'accès instantanée via une vérification dans le cloud avec Firebase. Ce dispositif remplit les trois piliers essentiels définis au départ : une sécurité robuste, une fiabilité face aux perturbations et une rapidité minimisant le temps d'attente pour les usagers. C'est une étape concrète vers des infrastructures urbaines plus intelligentes et autonomes.

**MERCI POUR VOTRE  
ATTENTION**