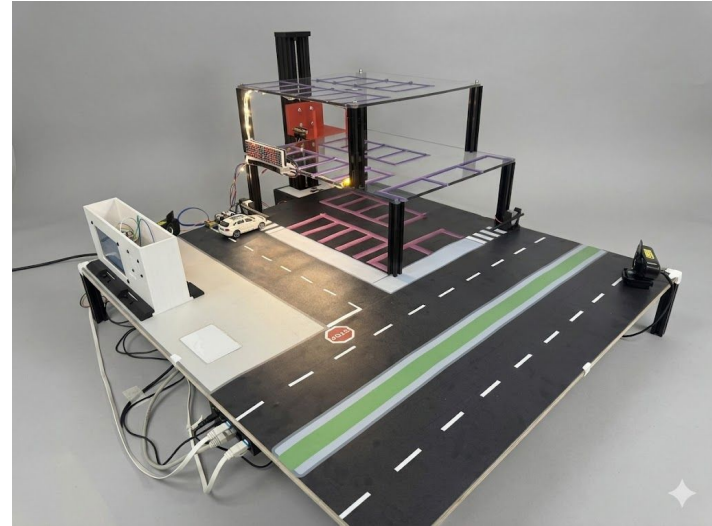




# Projet **P**ARKING **S**YSTÈME **M**ODULABLE

FALDA Andy  
CAUQUIL Vincent  
ES-SRIEJ Youness  
CLERVILLE Annabelle

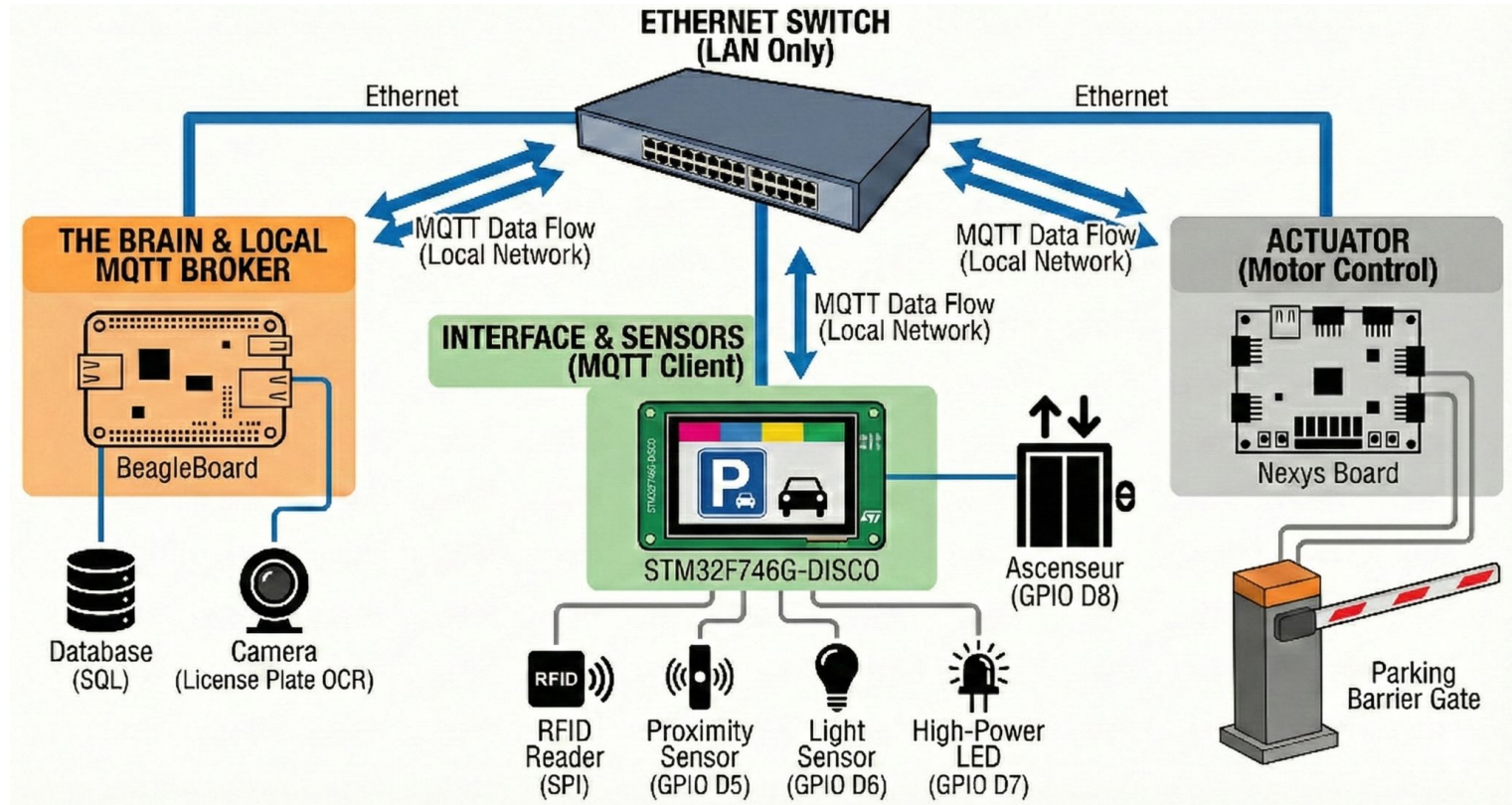
- I. Introduction
- II. Beagle
- III. STM32
- IV. FPGA
- V. Puissance
- VI. Conception 3D
- VII. Conclusion





# **Introduction**

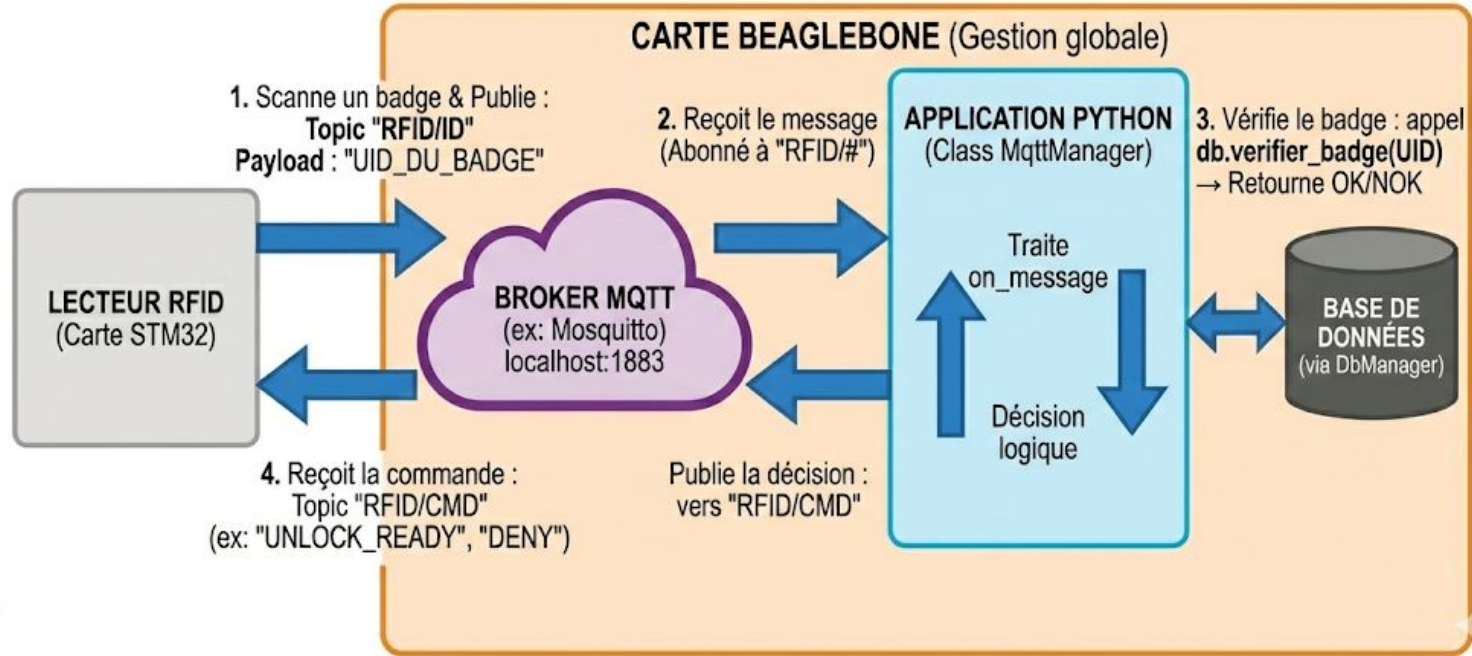
# Architecture

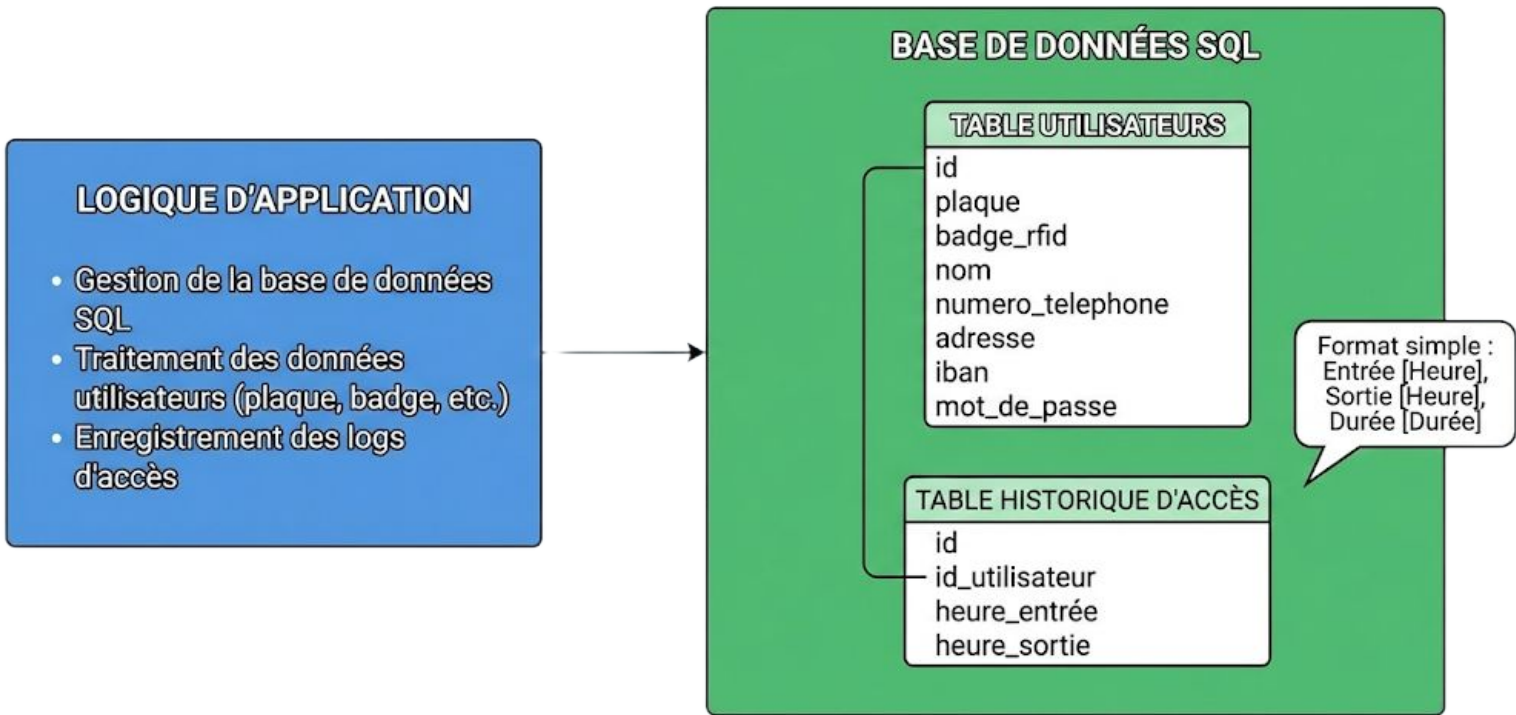


# **BeagleY-AI**

## ***Le cerveau***







# Caméra : reconnaissance des plaques

## Traitement d'Image (OpenCV *Open Source Computer Vision Library*)

- **Localisation** : Recherche des rectangles dans l'image
- **Recadrage** : Découpage l'image pour ne garder que la plaque
- **Nettoyage** : Passage de l'image en Noir & Blanc pour la lisibilité

## Lecture Intelligente (Tesseract)

- Moteur OCR Tesseract (Reconnaissance Optique de Caractères)
- Modèle d'IA déjà entraîné pour la reconnaissance des plaques
- **Extraction** : Il analyse les formes géométriques.
- **Traduction** : Il convertit les formes en texte "A", "B", "1"...

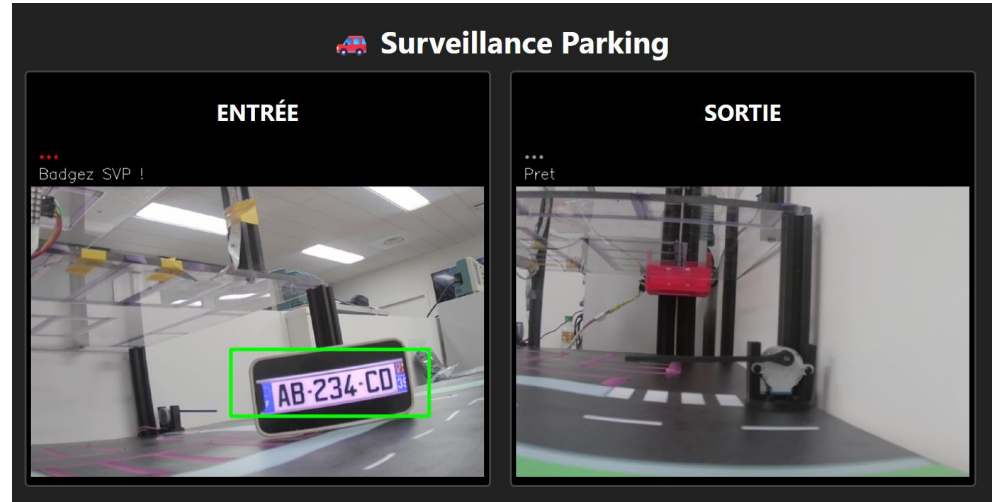
## Fiabilité & Correction

- Correction Logique : Vérification format (AA-123-AA) & correction erreurs (ex: 5 → S)
- Système de Vote : Validation sur **3 images consécutives**



## Interface de Gestion Web (Dashboard)

- **Hébergement** : Serveur Web embarqué sur la BeagleBone.
- **Frontend** : Interface développée en HTML/CSS pour le monitoring.
- **Fonctionnalités** :
  - Supervision du parking en temps réel.
  - Retour vidéo des caméras (Streaming Entrée/Sortie).
  - Visualisation directe de la détection des plaques.



## 1. Supervision & Monitoring

- **Console MQTT** : Visualisation du trafic en temps réel.
- **Journalisation** : Historique des passages validés (Logs).

## 2. Architecture & Déploiement

- **Architecture Web** : Basée sur HTML, légère et standard.
- **Flexibilité** : Déploiement simple (Serveur Web).
- **Gestion à distance** : Administration accessible de partout.

## 3. Perspectives d'évolution

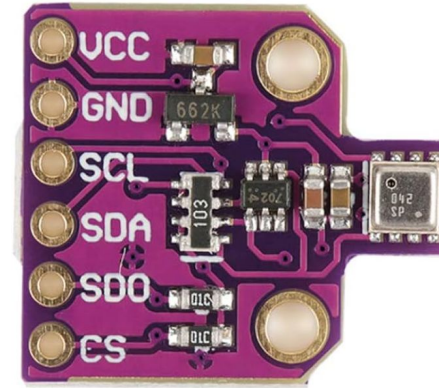
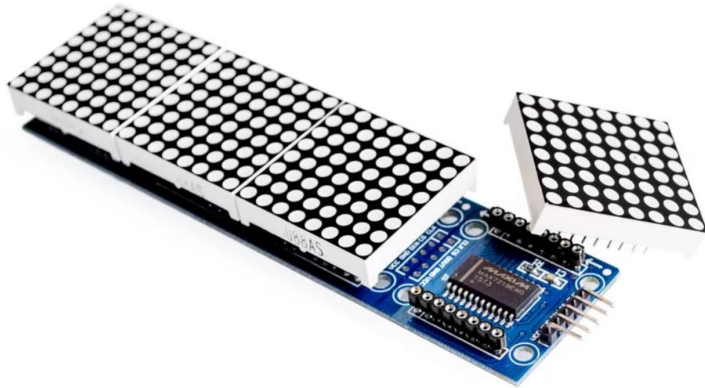
- **Gestion BDD** : Interface pour visualiser et modifier la base de données.
- **Contrôle d'urgence** : Commandes manuelles (ex : ouverture forcée des barrières).
- **Résolution d'incidents** : Déblocage utilisateur en cas de panne capteur.



The screenshot displays the 'Console MQTT' interface. At the top, there's a header 'Console MQTT' with a small icon. Below it, a large black area contains the text 'Chargement des logs...' in green. Below this, there's a section titled 'Historique des Passages' with a calendar icon. This section contains a table with the following data:

ID	Plaque	Entrée	Sortie	Etat
69	EV-666-DG	2026-01-21 18:26:50.017507	-	ENTRE
68	AA-000-AA	2026-01-21 18:13:49.129676	-	ENTRE
67	BA-234-CA	2026-01-21 18:10:04.292292	-	ENTRE

# Affichage Leds & Capteur T°C





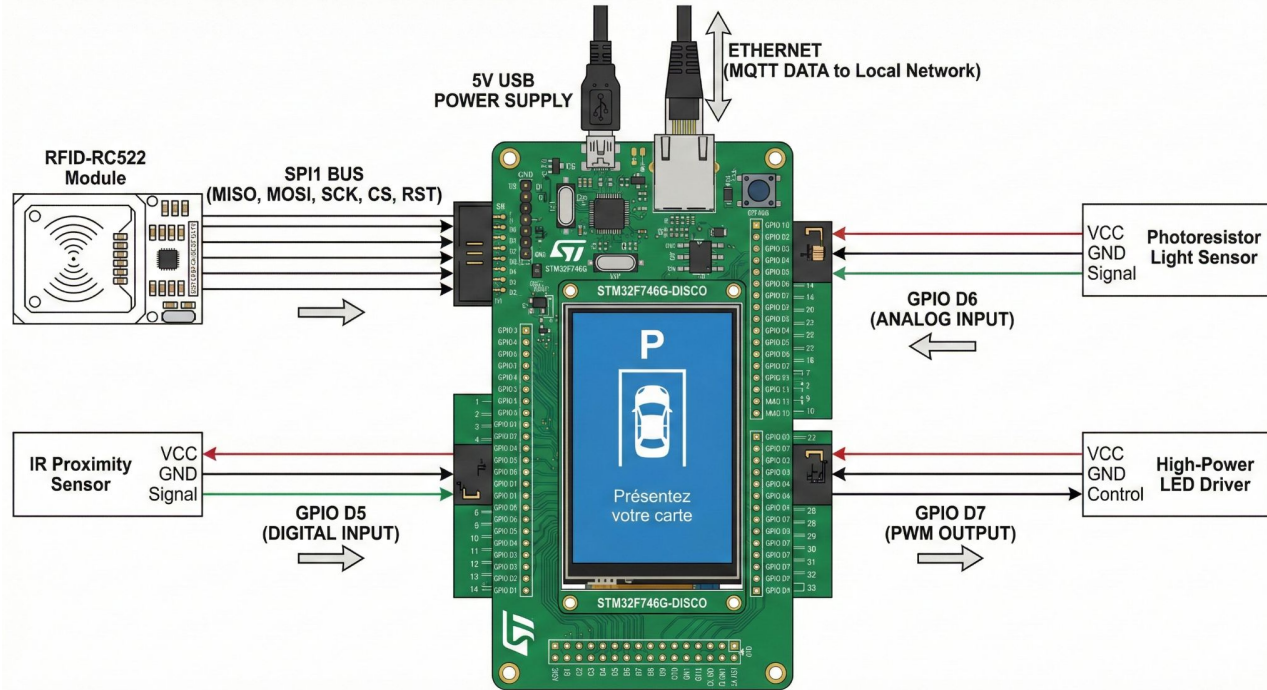
# **STM32** **RFID**

# Synoptique de la STM32

La carte STM32 est le **point d'entrée physique** du parking.

Les 4 missions principales :

1. **Identifier** : Lire les badges RFID.
2. **Interagir** : Afficher les messages via l'écran tactile.
3. **Sentir** : Détecter la présence voiture et la luminosité.



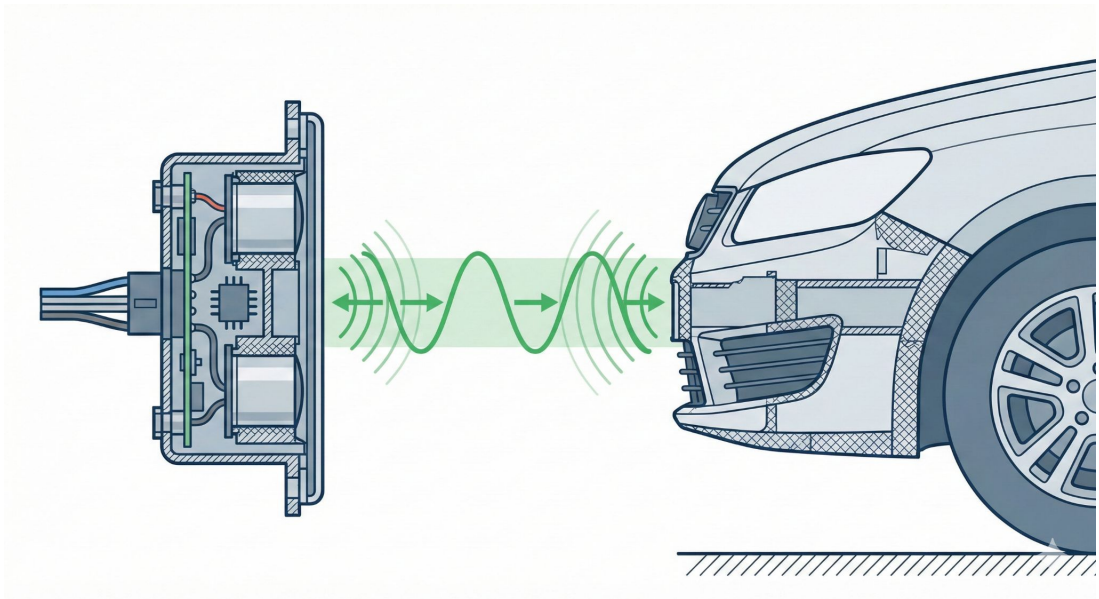
# Détection de Présence

## Fonctionnement

- Un capteur de proximité est connecté sur le **GPIO I0**.
- Il surveille en permanence si un véhicule est devant la barrière.

## Interaction

- **Si Voiture = 1** : Réveil de l'écran + Envoi MQTT **RFID/PRESENCE = 1**.
- **Si Voiture = 0** : Mise en veille de l'écran (Mode Eco) + Envoi MQTT **RFID/PRESENCE = 1**.

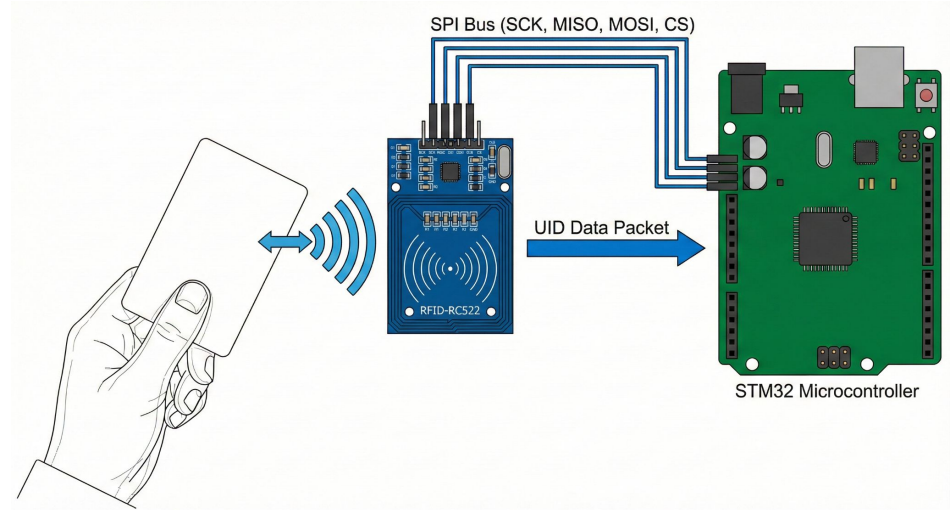


## Le Lecteur RC522 (SPI)

- La STM32 est équipée d'un lecteur RFID connecté via le bus **SPI1**.

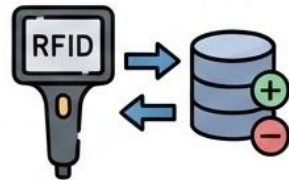
## Communication MQTT

- Pour chaque badge détecté, l'UID est envoyé au serveur :  
`mqtt_publish("RFID/ID", uid);`
- Le serveur (Beagle) répond alors par :
  - `UNLOCK` (ouvrir)
  - `DENY` (refuser).



## Le Badge Maître ou IT

- Un badge spécial est codé en dur dans la STM32, offrant des privilèges critiques :



### Déverrouillage Mode Éco :

Même sans voiture (écran éteint), passer ce badge force le **réveil immédiat** de l'écran.

### Mode Maintenance :

Donne accès à un menu caché sur l'écran tactile.

### Ajout / Suppression :

Permet de scanner un badge client pour l'ajouter (Touche 4) ou le supprimer (Touche 5) de la base SQL via MQTT .

# Gestion Automatique de l'Éclairage

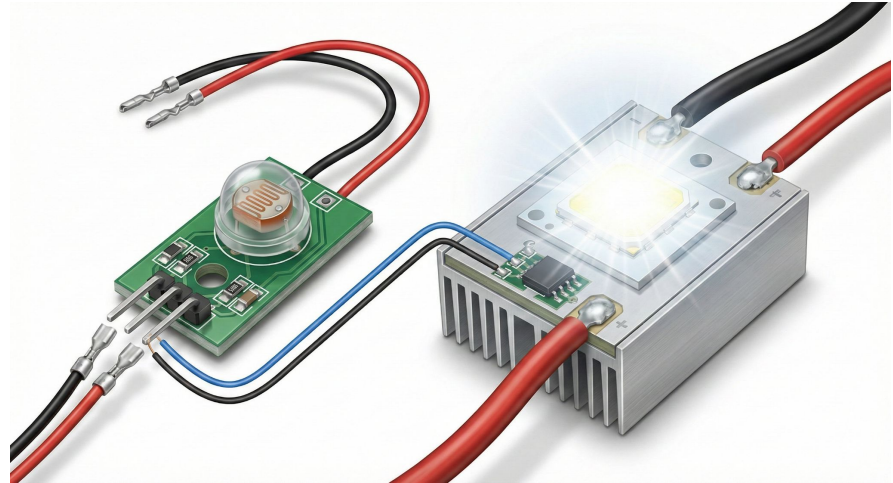
## Le Capteur D6

Une photo-résistance (sur **GPIO H6**) surveille la luminosité ambiante.

## Le Pilotage de Puissance (Relais)

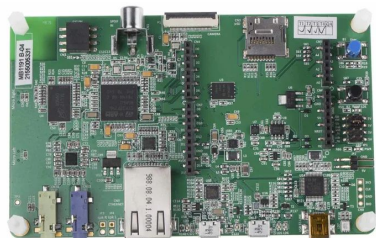
La STM32 ne pouvant pas alimenter directement des spots puissants, elle utilise un signal logique pour activer un **RELAIS**.

- **Logique** : Si Sombre -> Relais ON (Fermé).
- **Puissance** : Le relais laisse passer le **12V externe** pour allumer les LEDs.

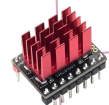




# **STM32** **Ascenseur**



STM32  
(zephyr)



Driver



Alimentation  
externe

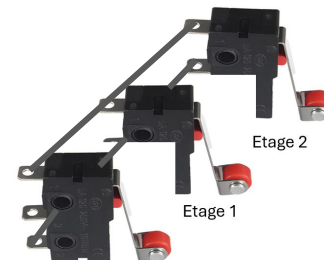
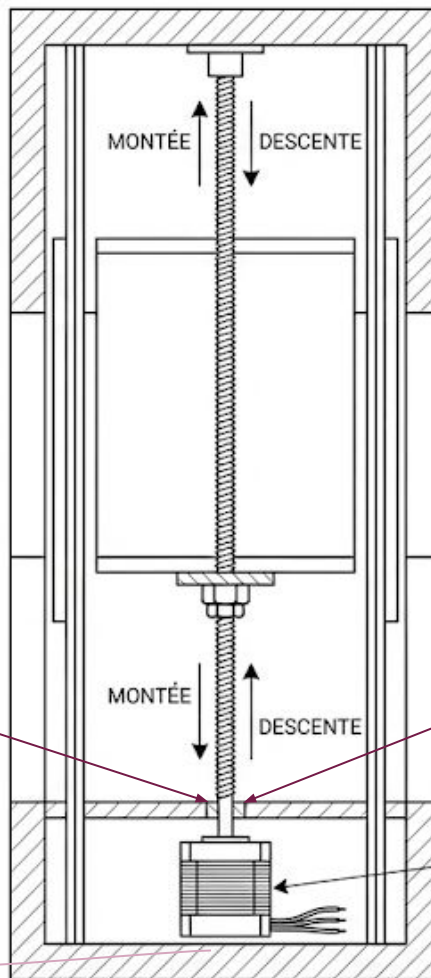


Capteur de distance

ÉTAGE 2

ÉTAGE 1

RDC



Boutons ascenseur



Capteur de fin de course



MOTEUR PAS À PAS



# FPGA

# Pilotage des Barrières

## Finalité :

- Commander les drivers moteurs pas à pas (portails entrées/sorties)

## Interface :

- **Ethernet + MQTT** (ordres reçus du “cerveau”)

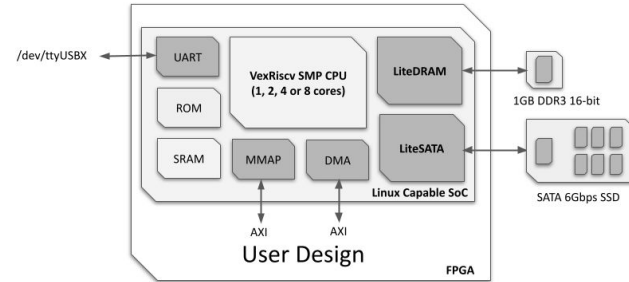
## Action :

- Traduction des commandes open/close en signaux moteur.



Le SoC custom permet une adaptation précise à l'architecture :

- Processeur Risc-V (Linux ready).
- Bus AXI permettant la liaison des périphériques :
  - (I/O moteur, liaison UART, connectivité Ethernet...).



Multi-Core Linux SoC on Acorn baseboard

EnjoyDigital

Built with  LiteX  
Build your hardware, easily!

<https://github.com/litex-hub/linux-on-litex-vexriscv/tree/master?tab=readme-ov-file>

- Le programme Linux mappe les registres via mmap.
- Le pilotage s'effectue par l'écriture de commandes dans l'IP moteur via MQTT.
- Cette approche ouvre la voie à une potentielle réalisation en ASIC.

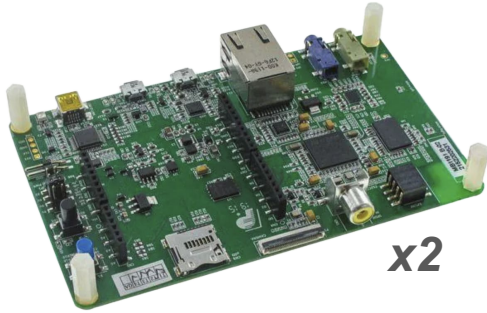


- 
- The diagram illustrates the MicroBlaze system architecture, showing the interconnections between various components:
- MBDEBUG\_0-0**: A debug module connected to the **Debug\_SYS\_Rst** block.
  - MicroBlaze Debug Module (MDM)**: Contains the **axi\_pcie\_0** block, which is connected to the **MicroBlaze** block.
  - MicroBlaze**: The central processing unit, featuring an **Interrupt** block and a **Debug** block. It is connected to the **microblaze\_0\_local\_memory** block.
  - microblaze\_0\_local\_memory**: Local memory block connected to the **MicroBlaze** block.
  - rst\_axi\_pcie\_0\_62M**: A reset block connected to the **MicroBlaze** block.
  - Processor System Reset**: A block connected to the **rst\_axi\_pcie\_0\_62M** block.
  - microblaze\_0\_axi\_periph**: An AXI Interconnect block connected to the **MicroBlaze** block.
  - microblaze\_0\_axi\_intc**: An AXI Interrupt Controller block connected to the **microblaze\_0\_axi\_periph** block.
  - Concat**: A block connected to the **microblaze\_0\_axi\_periph** block.



# **Puissance & Alimentation**

# Étages d'alimentation

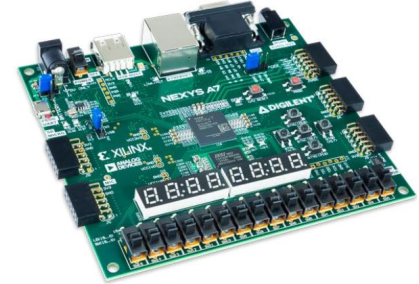


x2

*stm32f746 discovery*



*BeagleY-AI*



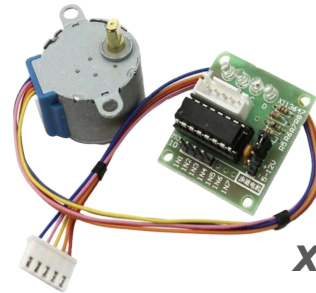
*Nexys A7 100T*



20 V  
1 A



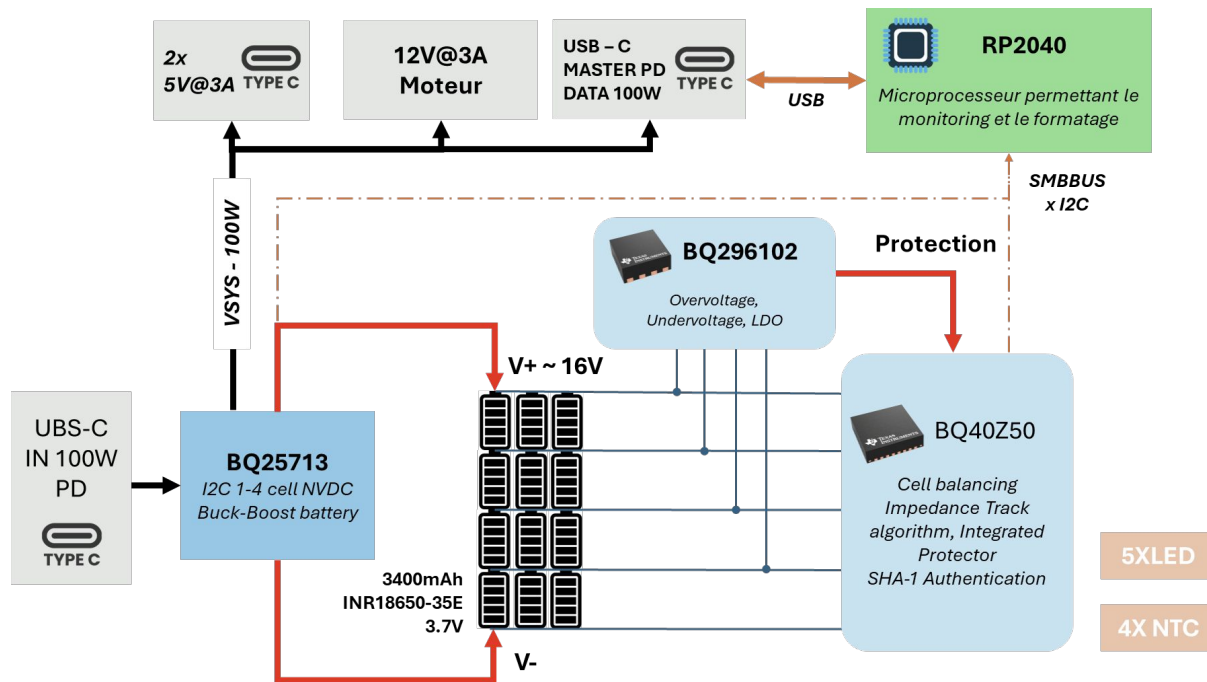
12 V  
0.5 A



x2

5 V  
3 A

# Étages d'alimentation





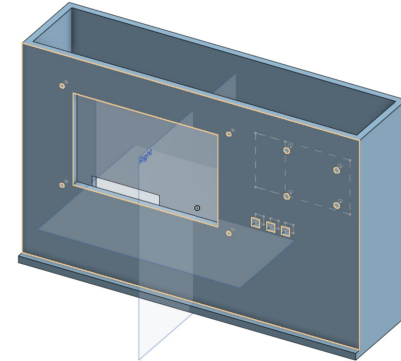
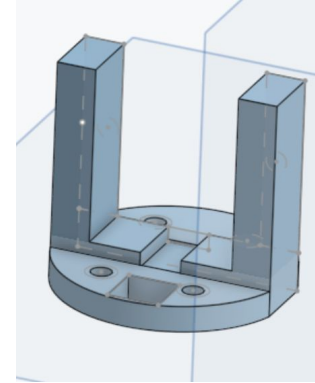
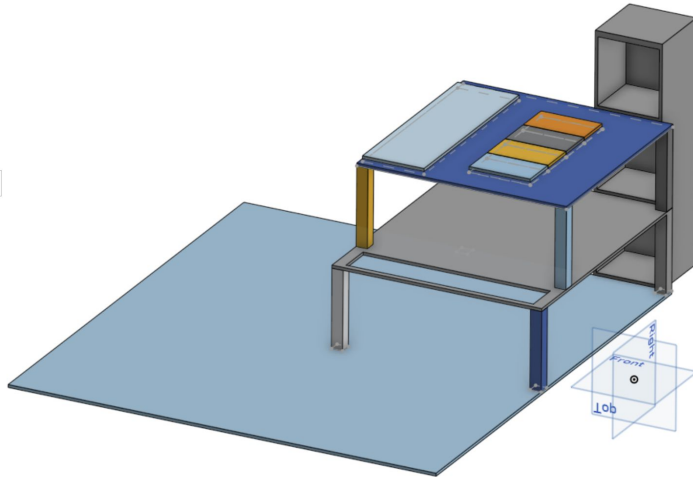
# **CONCEPTION**

## **3D + Design**

# Conception via Onshape

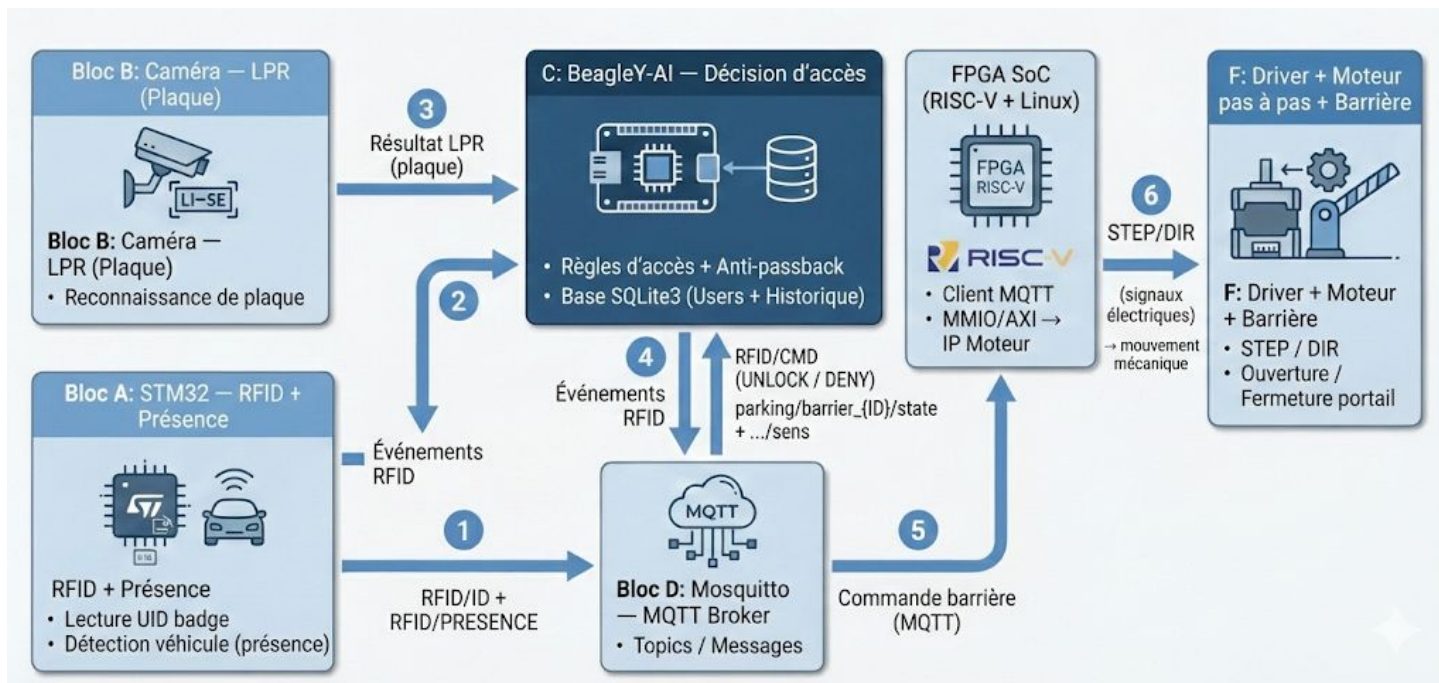
**Réalisation d'une maquette fonctionnelle  
mécaniquement pour intégrer les différents modules du  
système :**

- Barrière motorisée
- Supports de modules RFID, FPGA...





# **C**ONCLUSION





**Merci**  
**de Votre**  
**Attention**

## SYSTÈME DE GESTION DE PARKING : ARCHITECTURE BASE DE DONNÉES & APPLICATION

