المدرسة الوطنية للعلوم التطبيقية بتطوان
ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES DE TÉTOUAN

# SYSTÈME DE CONTRÔLE D'ÉCLAIRAGE INTELLIGENT

Capteurs & Instrumentations

2025-2026

# CADRE ACADÉMIQUE

## Encadré par :

PR. ZAKARIA ERRACHIDI

## Réalisé par :

MOUAD EL BEKKALI

AHMED AYMEN TIBTANI

AMINE BELAMINE

SAAD EL GUELYOUY

## Classe :

GCSE 2

# SOMMAIRE

# INTRODUCTION

Ce projet académique porte sur le développement d'un système d'éclairage intelligent basé sur un ESP32-S3. Il intègre un mode automatique utilisant des capteurs PIR et LDR pour gérer l'éclairage selon le mouvement et la luminosité, ainsi qu'un mode manuel permettant le suivi de l'état de la LED via une interface web. Le projet met en évidence les compétences en systèmes embarqués, IoT, communication réseau et intégration capteurs-actionneurs.

# PROBLÉMATIQUE

**Comment optimiser l'éclairage d'un espace pour réduire le gaspillage énergétique tout en maximisant le confort et l'autonomie du système ?**

# OBJECTIFS DU PROJET

- **Axe 1 : “Efficience énergétique”**

**Cet axe vise à réduire le gaspillage électrique en assurant une consommation d’énergie uniquement en présence humaine et en conditions de faible luminosité, grâce à un système d’extinction automatisé.**

- **Axe 2 : “Intelligence et adaptabilité”**

**Cet axe consiste à concevoir un système contextuel capable de interpréter son environnement en combinant les données des capteurs PIR et LDR, afin d’éviter les déclenchements inutiles, notamment en conditions de forte luminosité.**

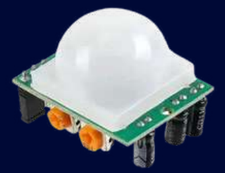
- **Axe 3 : “Connectivité et Flexibilité (IoT) “**

**L'enjeu est de sortir du matériel figé pour offrir un contrôle logiciel. La question est : Comment permettre à l'utilisateur de surveiller l'historique et de recalibrer les seuils de sensibilité à distance via une interface web et une base de données ?**

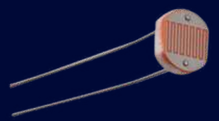
# MATÉRIELS UTILISÉS

01 **ESP32-S3**

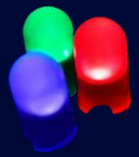
02 **Capteur de mouvement (PIR)**



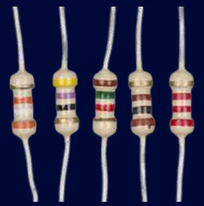
03 **Capteur de lumière (LDR)**



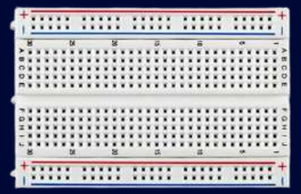
04 **LED**



05 **Résistances**



06 **Half size breadboard**



# LOGICIELS UTILISÉS

01 **Wokwi**

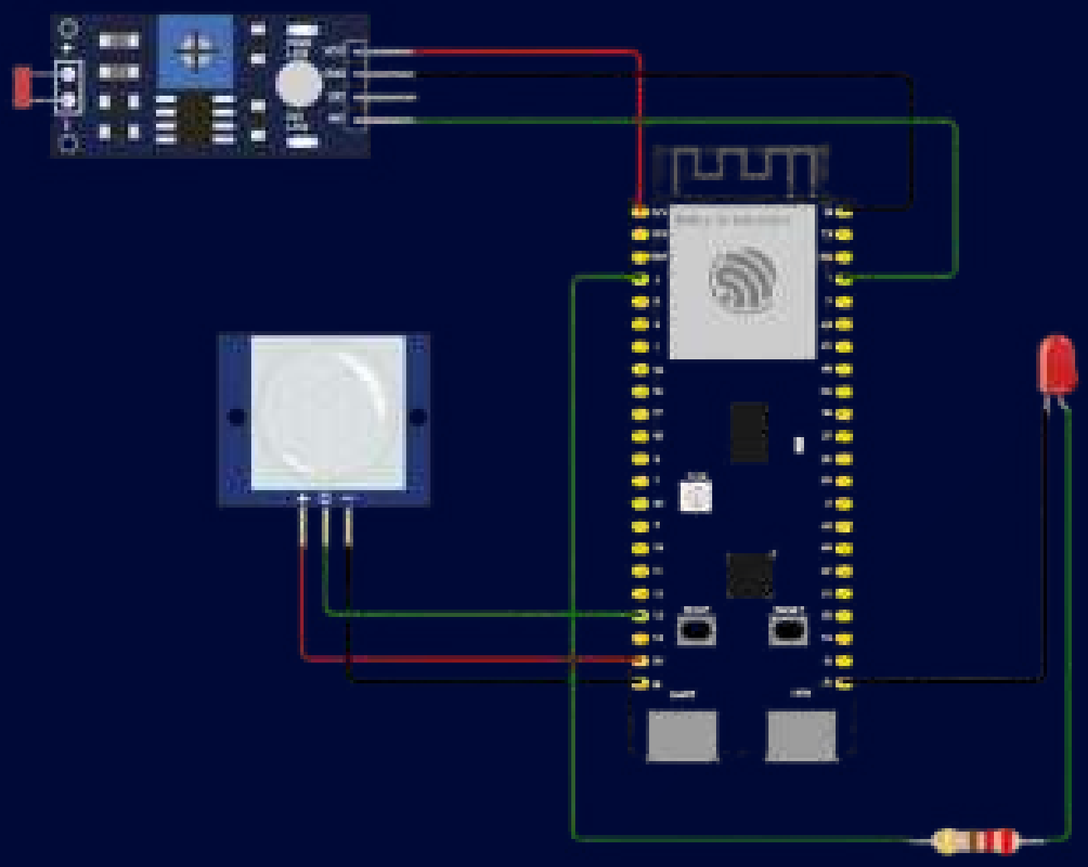
02 **Visual Studio Code**

03 **PlatformIO**

04 **SQLite**


03 **Flask**

# SCHÉMA DE CÂBLAGE



01

Le prototype utilise l'ESP32-S3 comme unité de contrôle

02

## Entrées (Capteurs) :

- Le capteur de mouvement PIR est connecté à l'entrée digitale GPIO 13
- Le capteur de luminosité LDR est connecté à l'entrée analogique GPIO 1

03

## Sortie (Lumière) :

- La LED est connectée à la sortie digitale GPIO 4 pour le contrôle de la lumière. Elle est protégée par une résistance série

04

## Alimentation :

- Les alimentations des composants sont gérées par les broches 3.3V de l'ESP32-S3
- Toutes les masses (GND) des composants sont reliées à la broche GND de l'ESP32-S3

# LOGIQUE DE FONCTIONNEMENT

**Le système se comporte selon la logique suivante :**

**01**

**Si un mouvement est détecté et que la luminosité est faible (< Threshold), la LED s’allume**

**02**

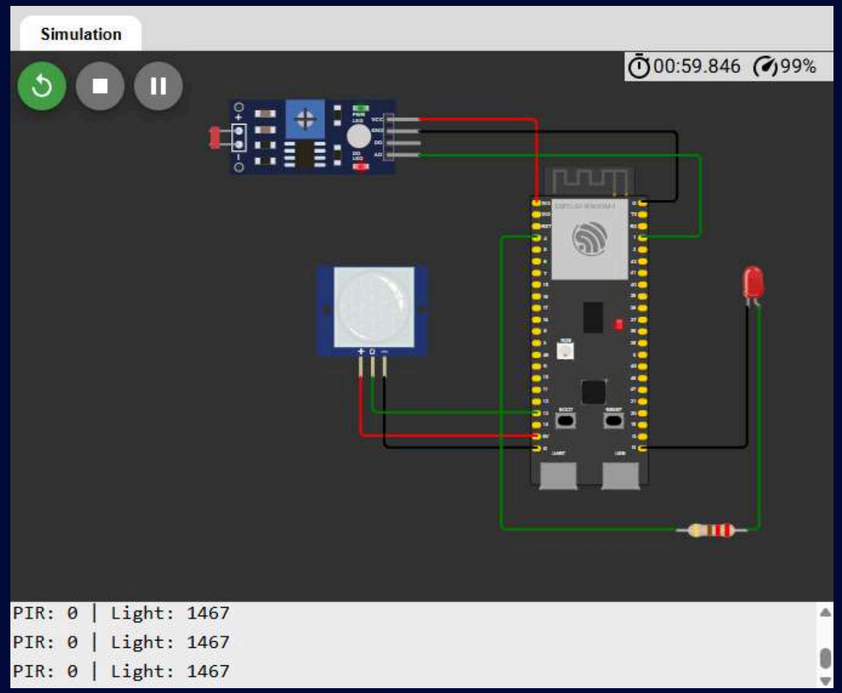
**Dans tous les autres cas (pas de mouvement ou luminosité suffisante), la LED reste éteinte**

| Mode de Contrôle (lightMode) | Mouvement (PIR) | Lumière faible (LDR < Seuil) | Sortie LED (LED_PIN) | Description de l'État |
|---|---|---|---|---|
| **AUTO (0)** | 0 | X | **0 (OFF)** | Immobile : La lumière est éteinte. |
| **AUTO (0)** | 1 | 0 | **0 (OFF)** | Mouvement, mais luminosité suffisante. |
| **AUTO (0)** | 1 | 1 | **1 (ON)** | Mouvement ET Obscurité (La condition est remplie). |

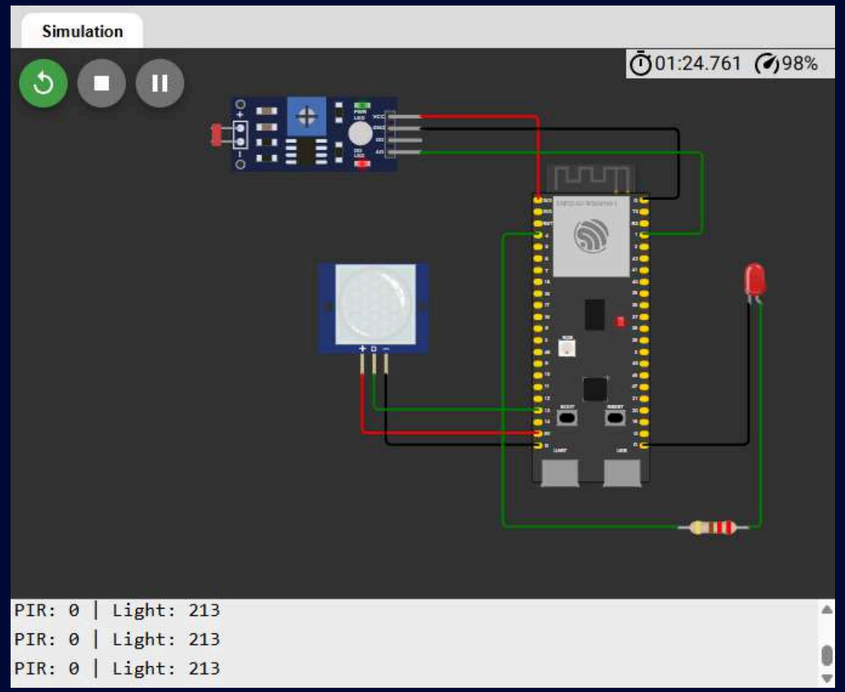
# SIMULATION SUR WOKWI

**Pour tester le fonctionement de la logique du système, on a utilisé WOKWI pour ne pas affecter les matériaux au cas d'une erreure**

**01** **Si PIR = 0 & LDR = X :**



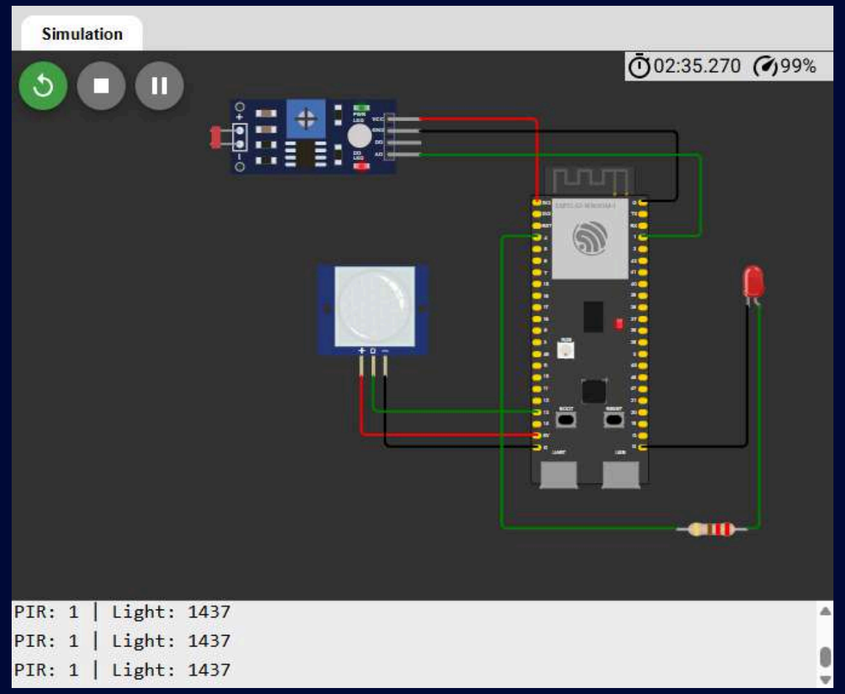

**La LED reste éteinte**




**La LED reste éteinte**

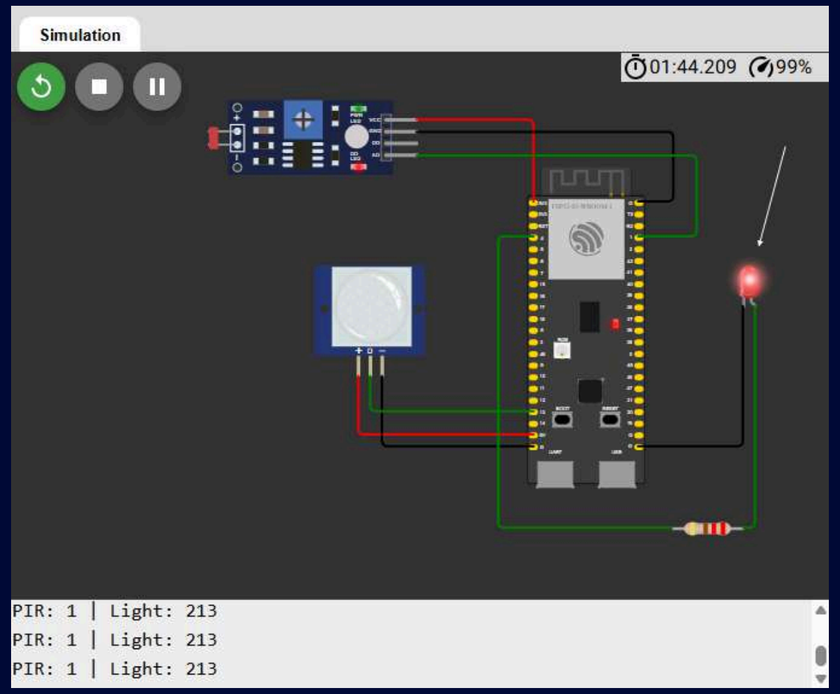
# SIMULATION SUR WOKWI

**02** **Si PIR = 1 & LDR > 1000 :**




**La LED reste éteinte**

**03** **Si PIR = 1 & LDR < 1000 :**




**La LED s’allume**

# EXPLICATION DU CODE

Le cœur de ce projet repose sur le framework Flask pour le serveur web, car il est léger et idéal pour les applications IoT

01

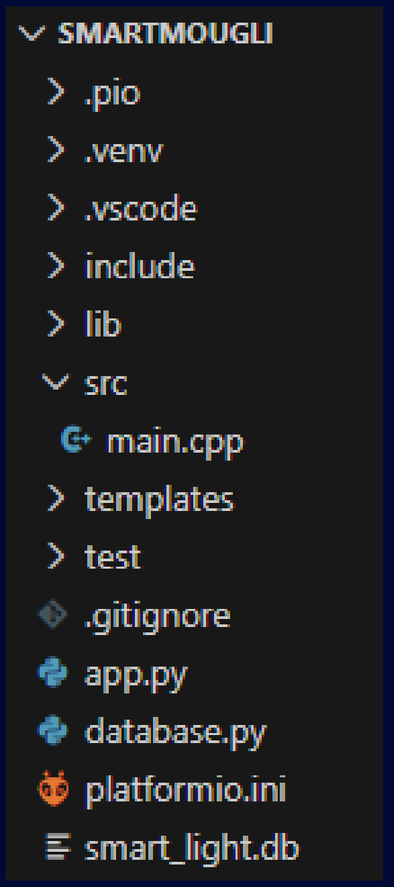
## Fichier Principal du Serveur : “app.py”

Ce fichier est le point d'entrée de l'application web. Il définit les routes URL, gère la logique de la page d'accueil, et assure la communication bidirectionnelle entre le navigateur (l'utilisateur) et la base de données (le matériel)

02

## Gestion de la Base de Données : “database.py”

Ce fichier est responsable de toutes les interactions avec le fichier de base de données smart_light.db



# EXPLICATION DU CODE

03

## Fichier de Configuration : “platformio.ini”

**Ceci le fichier de configuration de l'outil PlatformIO. Il indique à PlatformIO quel microcontrôleur est utilisé (par exemple, esp32dev), quelles bibliothèques C++ sont nécessaires, et comment compiler et téléverser le code du capteur**

```
11  [env:esp32-s3-devkitc-1]
12  platform = espressif32
13  board = esp32-s3-devkitc-1
14  framework = arduino
15  monitor_speed = 115200
16  upload_speed = 921600
17  lib_deps =
18      me-no-dev/ESPAsyncWebServer@^1.2.3
19      me-no-dev/AsyncTCP@^1.1.1
20      bblanchon/ArduinoJson@^7.0.4
21
22  build_flags =
23      -DCORE_DEBUG_LEVEL=0
```

# EXPLICATION DU CODE

04

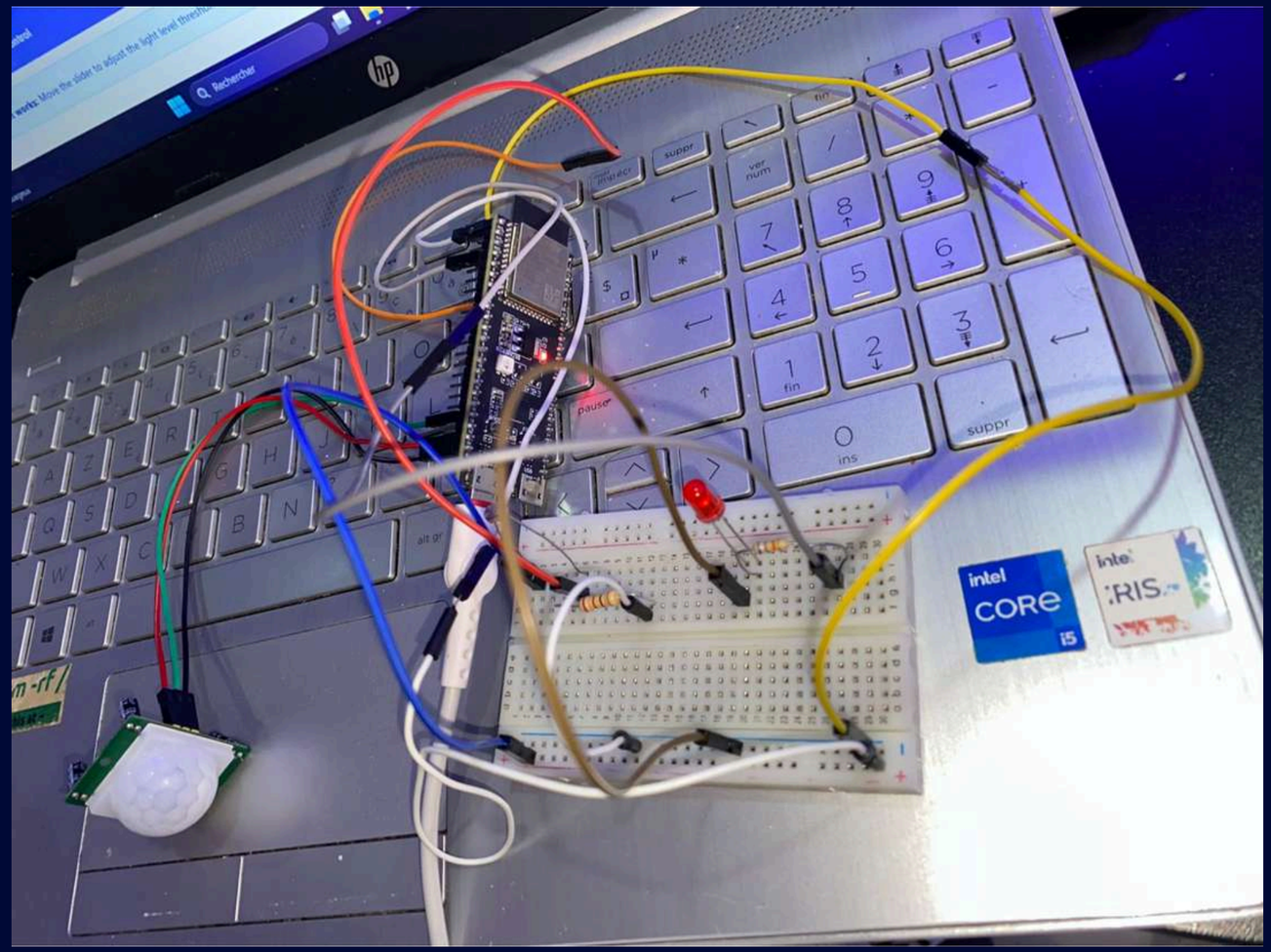
## Partie Embarquée : “src/main.cpp”

**Le fichier src/main.cpp, écrit en C++ et compilé via PlatformIO, constitue le cerveau décisionnel de l'ESP32-S3. Il assure trois missions critiques :**

- **L'acquisition de données : lecture en temps réel des signaux du capteur de mouvement (PIR) et de luminosité (LDR)**
- **La logique métier : exécution des algorithmes de contrôle (comparaison du seuil, gestion de la temporisation et déclenchement de la LED)**
- **La connectivité IoT : gestion de la pile Wi-Fi pour héberger le serveur web local et synchroniser les données avec la base de données SQLite**

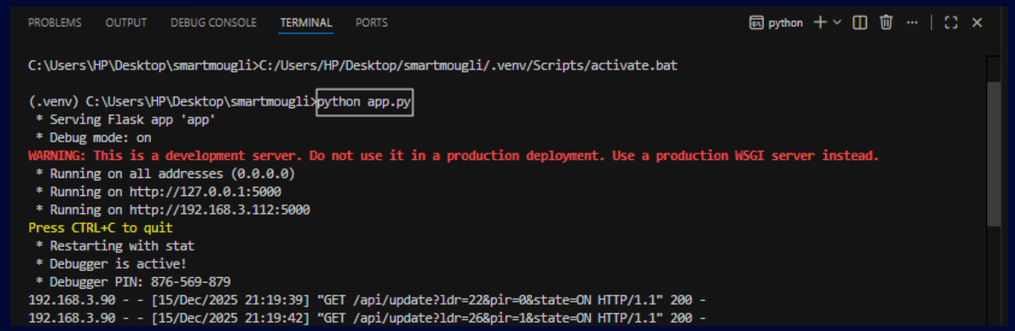
# MISE EN ŒUVRE MATÉRIELLE

Le microcontrôleur ESP32 a été câblé conformément au schéma d'architecture, connectant le capteur de luminosité LDR et le capteur de mouvement PIR aux broches définies. Après l'installation du pilote USB (CP210x), la carte a été reconnue par le système. Cette étape a permis le téléversement du code embarqué (main.cpp) via PlatformIO et l'établissement de la communication série et réseau essentielle pour l'interaction avec le serveur Python.

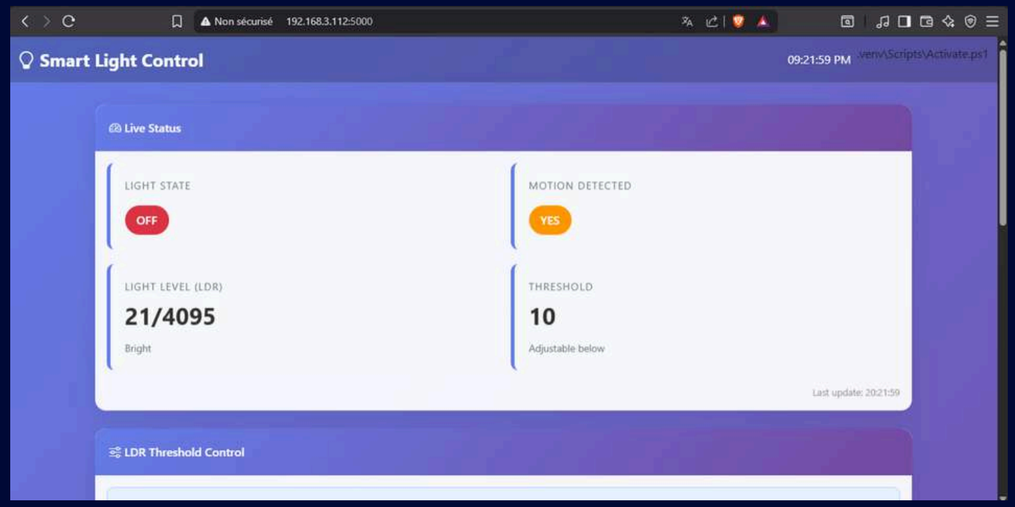


# SIMULATION RÉELLE

- Ouvrez le terminal et exécutez la commande python app.py (Python et son environnement doivent être installés)
- Copiez ensuite l'URL générée et collez-la dans un navigateur web



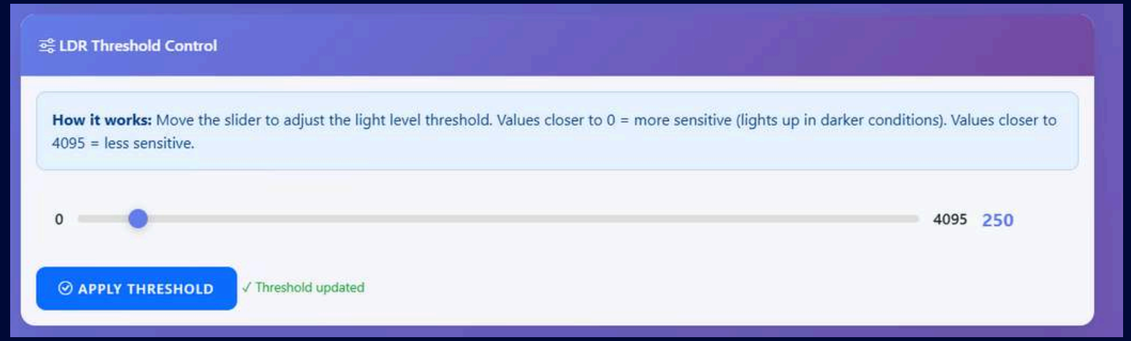

- Le site web "Smart Light Control" est l'interface utilisateur complète du système



# SIMULATION RÉELLE

**La structure de l’interface web est :**

- **Configuration (LDR Threshold Control) : Permet à l'utilisateur d'ajuster dynamiquement le seuil de luminosité via un curseur. Cette valeur est enregistrée pour modifier la sensibilité de l'algorithme de contrôle automatique**




- **Statut en Temps Réel (Live Status) : Affiche immédiatement l'état opérationnel du système, y compris si la lumière est ON/OFF, si un Mouvement est détecté, le Niveau de Lumière mesuré et le Seuil de déclenchement actuellement configuré**

# SIMULATION RÉELLE

- **Analyse des Données (Sensor Data Trends) : Fournit des outils d'historique, notamment un graphique de tendance du niveau de lumière des capteurs au fil du temps, avec la ligne du seuil superposée pour une analyse visuelle des événements de déclenchement**

- **Une section permet également la gestion et la purge des données historiques**

# SIMULATION RÉELLE

- Configuration du SSID et du mot de passe du réseau sans fil
- Copiez l'adresse IP du serveur web et collez-la dans le fichier src/main.cpp

```
8   // --- WiFi Configuration ---
9   const char* ssid = ".CAFE AZMIZAM 3";
10  const char* password = "123456789";
11
12  // --- Flask Server Configuration ---
13  const char* pythonServerHost = "192.168.3.112";
14  const int pythonServerPort = 5000;
```

- Téléversez le code en cliquant sur l'icône upload représentant une flèche orientée vers la droite, puis patientez jusqu'à la fin de l'opération

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

*  Executing task: C:\Users\HP\.platformio\penv\Scripts\platformio.exe run --target upload
Writing at 0x000bf36f... (85 %)
Writing at 0x000c4847... (88 %)
Writing at 0x000ccd5c... (91 %)
Writing at 0x000d53fd... (94 %)
Writing at 0x000da9b3... (97 %)
Writing at 0x000e0050... (100 %)
Wrote 866624 bytes (566567 compressed) at 0x00010000 in 8.1 seconds (effective 856.7 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
========================================= [SUCCESS] Took 23.11 seconds =========================================
*  Terminal will be reused by tasks, press any key to close it.

python
Upload...
```

- Ouvrez le monitor serie afin de visualiser les communications série en temps réel, en cliquant sur l'icône représentant une prise.
- L'ESP32-S3 se connecte alors au réseau Wi-Fi et les capteurs commencent à acquérir les valeurs mesurées

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

*  Executing task: C:\Users\HP\.platformio\penv\Scripts\platformio.exe device monitor
=== SMART LIGHT CONTROL ===
Calibrating PIR sensor (15 seconds)...
✓ PIR ready
Connecting to WiFi....
✓ WiFi connected!
IP: 192.168.3.90

System ready!
Commands: ON | OFF | AUTO | STATUS
AUTO | PIR: MOTION | LDR: 23 (Threshold: 200, DARK) | LED: ON
✓ Threshold updated from server: 10
AUTO | PIR: MOTION | LDR: 0 (Threshold: 10, DARK) | LED: ON
AUTO | PIR: IDLE | LDR: 0 (Threshold: 10, DARK) | LED: ON
AUTO | PIR: MOTION | LDR: 25 (Threshold: 10, BRIGHT) | LED: OFF

python
Upload Task
Monitor Task
```

# SIMULATION RÉELLE

**Nous vous invitons à observer la simulation en conditions réelles projetée sur le vidéoprojecteur**

# CONCLUSION

Ce projet aboutit à une solution d'éclairage autonome qui optimise la consommation électrique par une détection croisée de présence et de luminosité. La réussite de ce système repose sur l'alliance entre le matériel (capteurs et microcontrôleur) et le logiciel (interface web et base de données), offrant à l'utilisateur un contrôle total sur ses réglages. C'est une preuve concrète de l'utilité de l'IoT pour créer des environnements plus durables et réactifs aux besoins réels.

# MERCI DE
# VOTRE ATTENTION