جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

ENSAM
المدرسة الوطنية العليا للفنون و المهن
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS

Rapport du projet IOT

De l'objet connecté au réseau scolaire : ESP32 sur Wokwi & IoT sur Packet Tracer

---

Réalisé par: **Maha El Allam.**

Encadré par : **Brahim Bakkas.**

Table des matières

Liste des figures

I. Introduction générale

Avec l'évolution rapide des technologies de l'Internet des Objets (IoT), les bâtiments intelligents deviennent un élément clé dans l'optimisation de la gestion énergétique, de la sécurité et du confort des usagers. Dans le domaine de l'éducation, l'intégration de solutions IoT permet de transformer les établissements scolaires en environnements connectés, capables de surveiller et d'automatiser plusieurs services en temps réel.

Ce projet s'inscrit dans cette démarche et vise à concevoir un système de **Smart School** permettant la gestion intelligente de deux salles de classe et d'un couloir. Le système repose sur la collecte de données environnementales (température, humidité, mouvement, fumée) et sur l'automatisation des équipements (éclairage, ventilation, climatisation, alarme, porte et fenêtre).

La première partie du projet est consacrée à la programmation d'un **ESP32** à l'aide de la plateforme de simulation **Wokwi**. Elle permet de simuler le fonctionnement réel d'un microcontrôleur connecté à différents capteurs et actionneurs, tout en mettant en œuvre une logique embarquée structurée, non bloquante et orientée objet. Les données collectées sont formatées et transmises sous forme de trames JSON, facilitant leur exploitation dans un contexte IoT.

Cependant, les limitations de connectivité externe de Wokwi ont conduit à l'intégration de **Cisco Packet Tracer** dans la seconde partie du projet. Cette plateforme permet de simuler une infrastructure réseau IoT complète, incluant une Home Gateway, des objets connectés, des règles d'automatisation et une interface de supervision accessible depuis des terminaux mobiles. Elle offre ainsi une vision globale de l'interconnexion et de la gestion réseau des objets IoT au sein d'un établissement scolaire.

À travers ce projet, l'objectif est de démontrer la faisabilité d'une solution IoT éducative complète, alliant programmation embarquée, automatisation intelligente et architecture réseau, tout en respectant les principes fondamentaux des systèmes connectés modernes.

II. Partie 1 : programmer un ESP32 avec des capteurs /Actionneurs (simulation Wokwi).

1. Objectif du projet :

- Découvrir l'ESP32 (brochage, entrées/sorties).
- Lire des capteurs (température, mouvement, humidité)
- Commander des actionneurs (LED, board).
- **Création d'un système de gestion intelligente pour un établissement (2 classes et 1 couloir).**

2. Création d'un système de gestion intelligente pour un établissement (2 classes et 1 couloir).

a. ESP32 (Microcontrôleur)

• **Définition :** C'est le "cerveau" du projet. C'est une carte de développement puissante équipée du Wi-Fi et du Bluetooth.

• **Fonctionnement :** Il exécute le code qu'on a écrit, lit les données envoyées par les capteurs et envoie des ordres aux actionneurs (LEDs, Écran OLED).

b. DHT22 (Capteur d'humidité et de température)

• **Définition :** Un capteur numérique capable de mesurer la chaleur ambiante et le taux d'humidité dans l'air.

• **Fonctionnement :** Il utilise un élément capacitif pour l'humidité et une thermistance pour la température. Il envoie ces données à l'ESP32 via un protocole série à un seul fil (bus digital).

c. PIR (Capteur de mouvement infrarouge)

• **Définition :** Un capteur qui détecte le rayonnement infrarouge émis par des corps chauds (humains ou animaux) en mouvement.

• **Fonctionnement :** Lorsqu'une personne passe devant le capteur, la différence de rayonnement est détectée. Le capteur envoie alors un signal "Haut" (1) à l'ESP32 sur la broche de sortie.

d. OLED SSD1306 (Écran d'affichage)

• **Définition :** Un petit écran graphique monochrome de 128x64 pixels.

• **Fonctionnement :** Il reçoit les informations textuelles ou graphiques de l'ESP32 via le protocole de communication **I2C** (broches SDA et SCL). Il permet de visualiser les mesures sans avoir besoin d'un ordinateur.

e. LED (Diodes Électroluminescentes)

• **Définition :** Des composants qui émettent de la lumière lorsqu'un courant électrique les traverse.

• **Fonctionnement :** Dans mon projet, elles simulent l'éclairage des pièces. L'ESP32 les allume en envoyant une tension de 3.3V sur la broche correspondante dès qu'un mouvement est détecté ou qu'il fait trop sombre.

f. Résistances

• **Définition :** Des composants passifs utilisés pour limiter le passage du courant électrique.

• **Fonctionnement :** Elles sont placées en série avec les LEDs pour éviter que celles-ci ne reçoivent trop de courant et ne grillent. Elles protègent ainsi les composants et l'ESP32.

3. Description Fonctionnelle

L'objectif de ce projet est de concevoir un système de **Smart Building** (Bâtiment Intelligent) capable de superviser deux salles de classe et un couloir. Le système assure trois fonctions principales : le monitoring environnemental, l'automatisation de l'éclairage et la communication des données.

a. Surveillance de l'Environnement (Monitoring)

Le système surveille en permanence la qualité de l'air et le confort thermique dans les zones équipées :

- **Mesure de la Température et de l'Humidité :** Grâce aux capteurs **DHT22**, l'ESP32 récupère les données climatiques des classes 1 et 2. Cela permet de vérifier si les conditions sont optimales pour les étudiants.

b. Gestion Automatisée de l'Éclairage

Le système optimise la consommation énergétique en n'allumant les lumières que lorsque cela est nécessaire :

- **Détection de Présence :** Les capteurs de mouvement **PIR** surveillent chaque zone.
- **Logique de Commande :** * Dès qu'un mouvement est détecté (étudiant entrant dans une classe ou passant dans le couloir), l'ESP32 active la **LED** correspondante à cette zone.
    - L'éclairage reste actif tant qu'une présence est détectée, évitant ainsi le gaspillage d'électricité dans les salles vides.

c. Interface et Retour Visuel

- **Affichage Local (OLED) :** Un écran **SSD1306** affiche les données en temps réel directement sur le boîtier de contrôle. L'affichage alterne entre les différentes zones pour offrir une vue d'ensemble sans intervention humaine.

4. Architecture Logicielle (Software)

L'architecture logicielle du projet repose sur une approche **modulaire et structurée**, développée en C++ (Arduino). Elle a été conçue pour être facilement évolutive (possibilité d'ajouter des zones sans réécrire le code).

a. Structuration des données (Modèle par Zone)

Au lieu de manipuler des variables isolées, le programme utilise une **structure (struct)** nommée Room. Cette structure regroupe toutes les caractéristiques d'une zone :

- **Entrées/Sorties :** Pins pour le DHT, le PIR et la LED.
- **Données :** Variables pour stocker la température, l'humidité et l'état du mouvement
- **Objets :** Pointeurs vers les bibliothèques spécifiques (DHT, HardwareSerial).

Cette méthode permet de traiter la "Classe 1", "Classe 2" ou le "Couloir" avec les mêmes fonctions, garantissant une grande clarté du code.

b. Algorithme de fonctionnement (La boucle principale)

Le logiciel suit un cycle de fonctionnement en trois étapes répétées à haute fréquence :

1. **Acquisition :** Lecture séquentielle de chaque capteur.
2. **Logique métier :** Traitement des seuils (ex: allumage de la LED si le mouvement est détecté).
3. **Sortie :** Mise à jour de l'affichage OLED et envoi des trames de données.

c. Gestion du Multitâche simulé

Pour éviter que l'affichage de l'écran n'interfère avec la lecture des capteurs, le code utilise la fonction millis() au lieu de delay(). Cela permet de :

- Mettre à jour l'écran OLED en mode "carrousel" (alternance des zones).
- Envoyer les données JSON toutes les 2 secondes sans bloquer la réactivité des capteurs de mouvement.

d. Formatage et Exportation des données (JSON)

L'une des forces du logiciel est l'utilisation de la bibliothèque **ArduinoJson**. Toutes les informations sont encapsulées dans un objet JSON structuré :

- **Interopérabilité :** Ce format est le standard du web (IoT), permettant au projet d'être connecté à des plateformes comme Node-RED, Thingspeak ou un serveur Python.
- **Scannabilité :** Les données sont envoyées via le port série à une vitesse de **115200 bauds**.

e. Bibliothèques utilisées

Le projet s'appuie sur des bibliothèques robustes pour la gestion du matériel :

- Adafruit_SSD1306 & Adafruit_GFX : Gestion de l'affichage graphique.
- DHT sensor library : Protocole de communication avec les capteurs DHT22.
- Wire : Communication via le bus I2C.

5. Schéma de Câblage

Cette section présente l'organisation physique du circuit simulé sur Wokwi. Le montage est conçu pour séparer les flux de données afin d'éviter les collisions sur le bus I2C.

a. Capture du montage

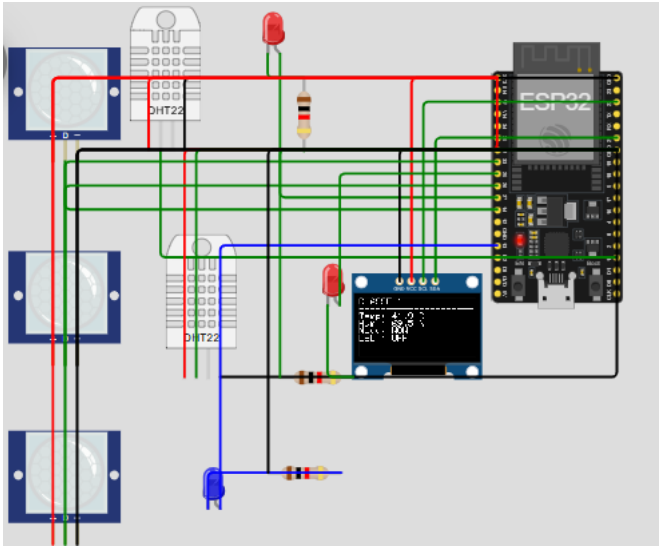

Figure 1 :montage sur wokwi

b. Liste des connexions principales

Pour assurer le fonctionnement simultané des périphériques, les broches (GPIO) de l'ESP32 ont été attribuées comme suit :

Composant	Type de Pin	Broches ESP32 (GPIO)
Écrans OLED (I2C)	SDA / SCL	**21** / **22** (Bus 1), **19** / **18** (Bus 2), **32** / **33** (Bus 3)
Capteurs PIR	Digital Input	**14** (Classe 1), **33** (Classe 2), **26** (Couloir)
Capteurs DHT22	Digital I/O	**15** (Classe 1), **32** (Classe 2)
LEDs (Éclairage)	Digital Output	**27** (Rouge), **25** (Rouge), **13** (Bleue)

6. Communication de Données (Console Serial)

La communication est un pilier central de ce projet IoT. L'ESP32 ne se contente pas d'afficher les données localement ; il les exporte vers l'extérieur de manière structurée.

a. Exemple de sortie console

Toutes les 2 secondes, le système génère une trame série au format **JSON**. Voici un exemple réel extrait de la console de simulation :

{"c1":{"temp":41.9,"hum":69.5,"motion":0},"c2":{"temp":-40.0,"hum":0.0,"motion":0},"couloir":{"motion":0}}

b. Intérêt du format JSON

L'utilisation de ce format standardisé présente plusieurs avantages pour un système professionnel :

- **Interopérabilité :** Ce format est directement lisible par des services Cloud et des tableaux de bord (Dashboards) tels que **Adafruit IO**, **Blynk** ou **Node-RED**.
- **Légèreté :** La structure clé-valeur réduit la taille des messages, ce qui est idéal pour les transmissions sans fil (Wi-Fi/MQTT).
- **Analyse de données :** Ces trames peuvent être stockées dans une base de données pour analyser l'évolution de la température ou les pics de présence dans l'école sur une année complète.

7. Extension IoT : Dashboard Blynk (Conceptuel)

Bien que la simulation actuelle fonctionne en local, une interface de supervision à distance a été conçue sur la plateforme **Blynk.cloud**. Ce tableau de bord permettrait à un administrateur de l'école de surveiller l'établissement depuis un smartphone ou un ordinateur.

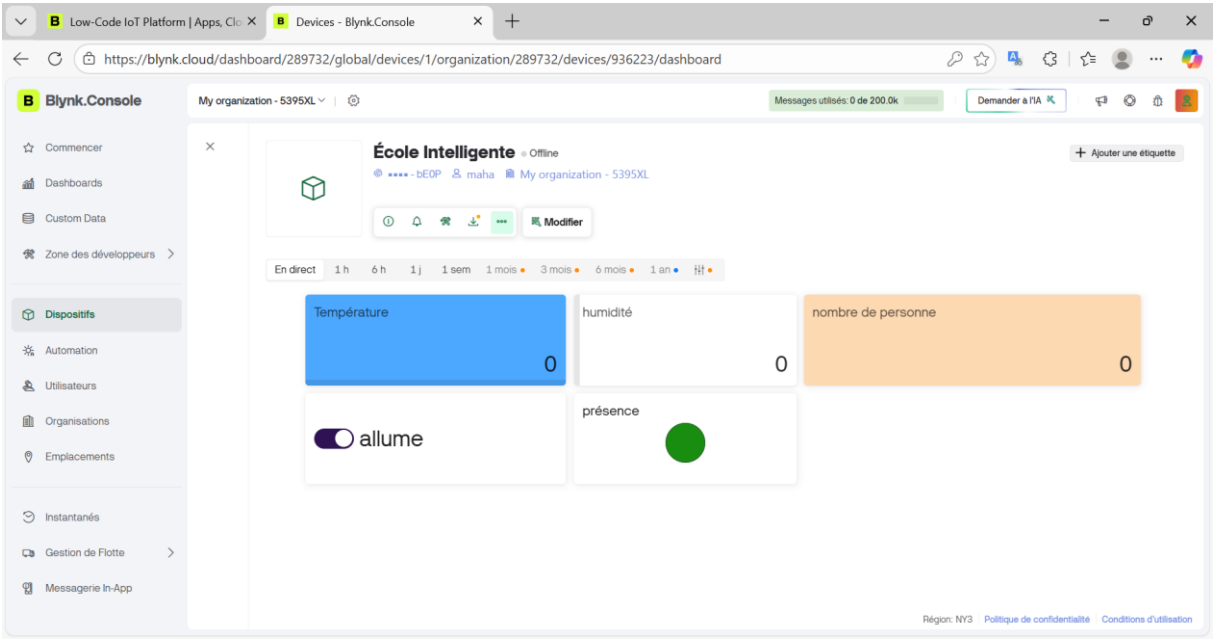

Figure 2 : interface blynk

a. Interface de Supervision

Comme illustré dans la capture d'écran du dashboard, l'interface comprend :

- **Indicateurs Numériques :** Affichage en temps réel de la **température**, de l'**humidité** et du **nombre de personnes** .

- **Contrôle à distance :** Un bouton "Allumer" pour forcer l'éclairage si nécessaire.
- **Indicateur de présence :** Un voyant visuel signalant si une zone est occupée.

b. Limitation Technique et Solution

Dans l'environnement de simulation **Wokwi**, l'accès au réseau externe (Internet) pour envoyer des données vers Blynk est impossible. Et pour cela j'ai migré vers **Cisco Packet Tracer** .

III. Partie 2 : Simulation de l'infrastructure réseau IoT (Cisco Packet Tracer)

1. Objectif de la migration

Suite aux limitations de connectivité externe de la version gratuite de Wokwi , l'utilisation de **Cisco Packet Tracer** permet de simuler une infrastructure réseau complète pour l'établissement. L'objectif est de valider l'interconnexion des objets connectés via une passerelle domestique (Home Gateway) et de simuler une supervision sur un réseau local.

2. Schéma global du réseau IoT

La figure suivante présente la topologie complète du réseau IoT simulé sous Cisco Packet Tracer. Elle illustre l'interconnexion entre les capteurs, les actionneurs, la passerelle domestique (Home Gateway) et les terminaux de supervision.

L'ensemble des objets connectés communique via un réseau local sans fil (Wi-Fi), permettant une gestion centralisée et une supervision en temps réel de l'établissement scolaire.

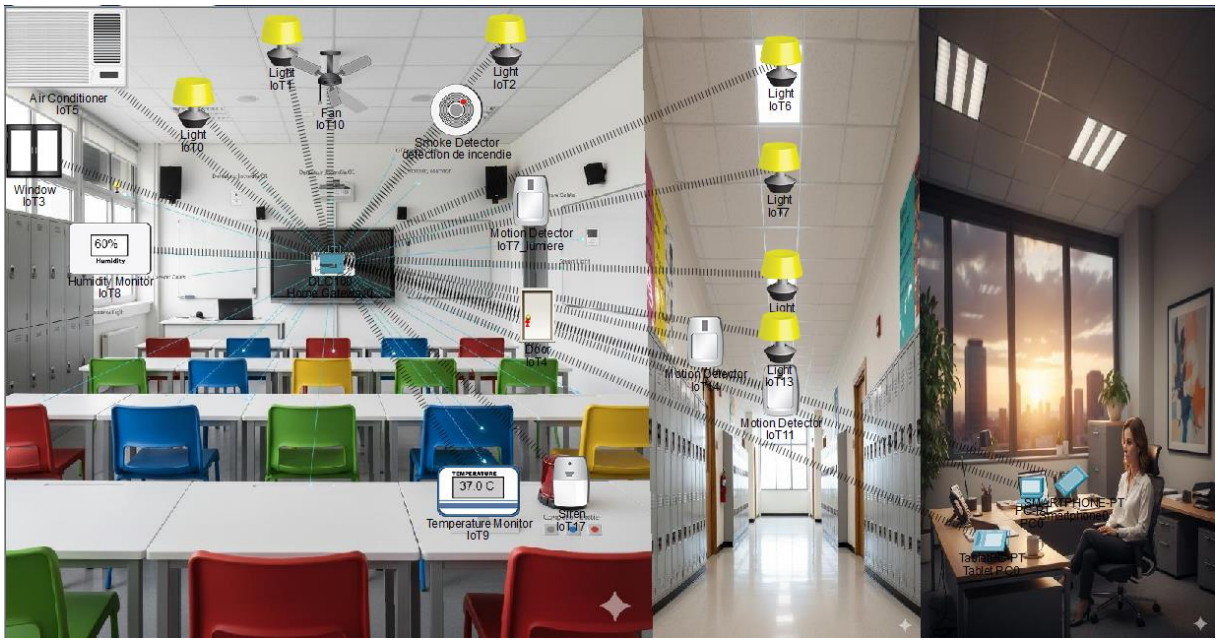

Figure 3 : composant cisco packet tracer

3. Architecture du réseau Smart School (Cisco Packet Tracer)

Cette partie détaille l'infrastructure réseau utilisée pour pallier les limitations de Wokwi. Contrairement à la simulation de composants électroniques, Packet Tracer se concentre sur l'interconnexion réseau et les services serveurs.

a. Définition des composants du réseau

D'après le montage réalisé, voici les rôles de chaque élément :

- **Home Gateway (Passerelle Domestique - DLC100) :** C'est le centre névralgique du réseau. Elle agit comme routeur, point d'accès Wi-Fi et serveur IoT local. Elle attribue les adresses IP et héberge l'interface de contrôle.
- **Capteurs (Sensors) :**
    - **Motion Detector (Détecteur de mouvement) (IoT7, IoT11, IoT14):** Capteur binaire qui détecte un passage dans les classes ou le couloir.
    - **Temperature Monitor (IoT9):** Capteur analogique mesurant la chaleur en degrés Celsius (°C) pour simuler le confort thermique.
    - **Smoke Detector (IoT8) :** Capteur de sécurité mesurant la densité de fumée pour prévenir les incendies.
- **Actionneurs (Actuators) :**
    - **Light (Luminaires) (IoT0, IoT1, IoT2, IoT6, IoT12, IoT13):** Plafonniers intelligents pouvant être activés à distance ou par condition.
    - **Siren (Alarme) :** Dispositif sonore activé en cas d'urgence (fumée ou intrusion).
    - **Porte (IoT4) & Fenêtre (IoT3) :** Éléments de structure motorisés.
    - **Fan (Ventilateur) (IoT10) / AC (Climatiseur) (IoT5) :** Systèmes de régulation thermique activés selon la température mesurée.
- **Terminaux de contrôle :**
    - **Smartphone / Tablet PC :** Appareils clients permettant à l'administrateur de visualiser l'état des capteurs via une application "IoT Monitor".

b. Programmation des Scénarios et Automatisation (Scripts IoT)

Pour valider le fonctionnement des actionneurs et la réactivité du réseau, j'ai développé des scripts personnalisés en **JavaScript** pour chaque capteur. Ces programmes remplacent les données environnementales passives par un **cycle de simulation dynamique** qui force le passage des seuils critiques définis dans les conditions de la Gateway.

(i) Simulation du Détecteur d'Incendie (Smoke Detector)

Le code implémente un cycle d'alerte automatique toutes les 5 secondes pour tester la sécurité de l'établissement.

- **Logique du code :** Le script utilise une variable ALARM_LEVEL fixée à 40. Toutes les 5 secondes, il bascule le niveau à **50**, dépassant ainsi le seuil de sécurité.

- **Impact sur les conditions :** Cette valeur force la condition "**il y a un incendie**", ce qui active immédiatement la sirène (IoT17).
- **Phrase technique du script :** setLevel(ALARM_LEVEL + 10); // Passage à 50 pour forcer l'alarme.

(ii) Simulation du Capteur d'Humidité (Humidity Sensor)

Ce script simule des variations météorologiques cycliques toutes les 15 secondes pour tester la protection des locaux.

- **Logique du code :** Le programme alterne entre un état normal (**60%**) et un état de pluie intense (**80%**).
- **Impact sur les conditions :** Lorsque l'humidité atteint **80%**, la condition "**fermer la fenetre et la porte s'il ya du pluie**" est validée car elle dépasse le seuil de **70%** programmé sur le serveur.
- **Résultat observé :** La fenêtre (IoT3) se ferme et la porte (IoT4) se verrouille automatiquement.

(iii) Simulation du Moniteur de Température (Temperature Monitor)

Ce script est le plus complet car il gère trois paliers de confort thermique sur un cycle total de 9 secondes, avec un changement d'état toutes les **3 secondes**.

- **Étape 1 (29.0°C) :** Température de référence. À ce niveau, les systèmes de refroidissement restent éteints car les seuils d'activation ne sont pas atteints.
- **Étape 2 (37.0°C) :** Température élevée. Cette valeur injectée force la validation simultanée de deux règles : **"allumer le ventilateur"** (seuil >= 20.0 °C) et **"allumer la clime"** (seuil >= 35.0 °C).
- **Étape 3 (16.0°C) :** Température de refroidissement. Cette valeur est utilisée pour simuler le retour à un état frais, déclenchant les conditions d'arrêt des systèmes de climatisation (**"eteindre la clime"** et **"eteindre ventilateur"** car < 35.0 °C).
- **Phrase technique du script :** updateTemperature(newValue); // Appeler la fonction de mise à jour avec la nouvelle valeur forcée

c. Tableau de Synthèse des Conditions d'Automatisation (Cisco Packet Tracer)

Paramètre (x)	**Condition Logique**	**Actionneur (y) : Nom [Code]**	**Action exécutée**
Mouvement	IoT7_lumiere is True	**Lampes Classe 1** [IoT0, IoT1, IoT2]	Allumer (On)
Mouvement	IoT7_lumiere is False	**Lampes Classe 1** [IoT0, IoT1, IoT2]	Éteindre (Off)

Paramètre (x)	Condition Logique	Actionneur (y) : Nom [Code]	Action exécutée
Température	>= 20.0 °C (+ Mvmt)	**Ventilateur** [IoT10]	Mode Rapide (High)
Température	>= 35.0 °C (+ Mvmt)	**Climatiseur** [IoT5]	Démarrer (On)
Température	< 35.0 °C	**Climatiseur** [IoT5]	Arrêter (Off)
Humidité	>= 70 %	**Fenêtre** [IoT3]	Fermer (On to False)
Mouvement	IoT14 is True	**Porte** [IoT4]	Déverrouiller (Unlock)
Mouvement	IoT11 is True	**Lampes Couloir** [IoT12, IoT13]	Allumer (On)
Fumée	Level >= 50	**Sirène d'alarme** [IoT17]	Activer (On)
Fumée	Level <= 50	**Sirène d'alarme** [IoT17]	Désactiver (Off)

Actions	Enabled	Name	Condition	Actions
Edit Remove	Yes	allumer la lumiere	IoT7_lumiere On is true	Set PTT0810U97T Status to 2 Set IoT0 Status to On Set IoT1 Status to On
Edit Remove	Yes	allumer le ventilateur	Match all: • IoT9 Temperature >= 20.0 °C • IoT7_lumiere On is true	Set IoT10 Status to High
Edit Remove	Yes	allumer la clime	Match all: • IoT9 Temperature >= 35.0 °C • IoT7_lumiere On is true	Set IoT5 On to true
Edit Remove	Yes	eteindre la lumiere	IoT7_lumiere On is false	Set IoT1 Status to Off Set IoT0 Status to Off Set PTT0810U97T Status to 0
Edit Remove	Yes	eteindre ventilateur	IoT9 Temperature < 35.0 °C	Set IoT10 Status to Off
Edit Remove	Yes	eteindre la clime	IoT9 Temperature < 35.0 °C	Set IoT5 On to false
Edit Remove	Yes	fermer la fenetre s'il ya du pluie	IoT8 Humidity >= 70 %	Set IoT3 On to false
Edit Remove	Yes	ouvrire la port si il ya un mouvement et allumer les lampes des couloires	Match all: • IoT14 On is true • IoT8 Humidity >= 0 %	Set IoT4 Lock to Unlock Set IoT7 Status to On Set IoT6 Status to On
Edit Remove	Yes	fermer la porte s'il ya pas de mouvement et etteindre les lampes du couloire	Match all: • IoT14 On is false • IoT8 Humidity >= 70 %	Set IoT4 Lock to Lock Set IoT7 Status to Off Set IoT6 Status to Off
Edit Remove	Yes	allumer les lampes du clouloire s'il ya une personne	IoT11 On is true	Set IoT12 Status to On Set IoT13 Status to On
Edit Remove	Yes	eteindre les lampe du couloir s'il n'y a personne	IoT11 On is false	Set IoT12 Status to Off Set IoT13 Status to Off
Edit Remove	Yes	Il ya un incendie	PTT0810078R Level >= 50	Set IoT17 On to true
Edit Remove	Yes	pas d'incendie	PTT0810078R Level <= 50	Set IoT17 On to false

Figure 4 : Conditions d'Automatisation

4. Configuration réseau et adressage IP

La Home Gateway joue le rôle de routeur et de serveur DHCP pour l'ensemble du réseau IoT.
Elle attribue automatiquement des adresses IP aux différents objets connectés ainsi qu'aux terminaux de contrôle (smartphone et tablette).

Cette configuration garantit :

- Une communication fluide entre les capteurs et les actionneurs
- Une intégration rapide de nouveaux équipements
- Une gestion simplifiée sans configuration IP manuelle

Tous les dispositifs appartiennent au même réseau local (LAN), ce qui permet une supervision centralisée et fiable.

5. Rôle de la Home Gateway dans l'architecture IoT

La Home Gateway constitue le cœur du système IoT. Elle assure plusieurs fonctions essentielles :

- Serveur IoT centralisant l'ensemble des objets connectés
- Moteur de règles pour l'automatisation (conditions et actions)
- Point d'accès Wi-Fi pour les capteurs et actionneurs
- Interface de supervision accessible depuis un terminal mobile

Elle agit comme un **middleware IoT**, reliant les capteurs aux actionneurs tout en appliquant les règles logiques définies par l'administrateur.

6. Interface de supervision (IoT Monitor)

L'administration du système est réalisée via l'application **IoT Monitor**, accessible depuis un smartphone ou une tablette connectée au réseau.

Cette interface permet :

- La visualisation en temps réel des capteurs (température, mouvement, fumée, humidité)
- Le contrôle manuel des actionneurs (lampes, sirène, porte, fenêtre, climatisation)
- La vérification de l'état global de l'établissement

Ainsi, l'administrateur peut intervenir rapidement en cas d'incident ou laisser le système fonctionner de manière totalement automatisée.

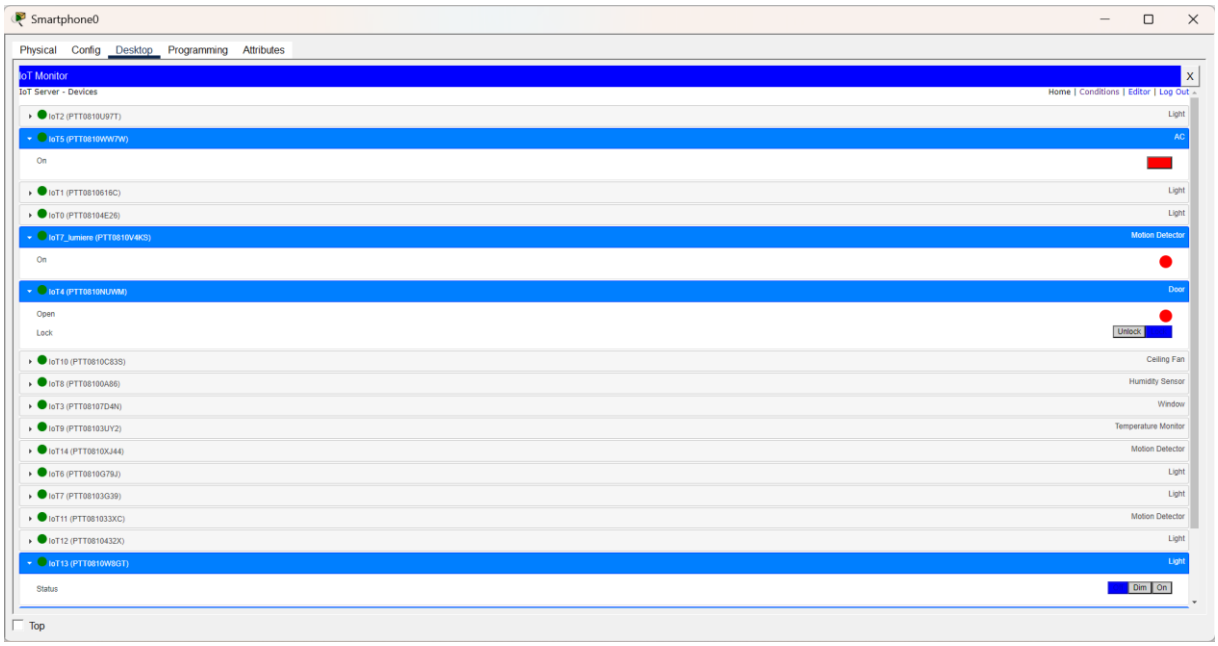

Figure 5 :Interface de supervision (IoT Monitor)

4. Conclusion

Pour surmonter les limitations de connectivité cloud de Wokwi, la migration vers **Cisco Packet Tracer** a permis de simuler une infrastructure réseau locale complète et autonome pour l'établissement. Ce système repose sur une **Home Gateway (DLC100)** agissant comme serveur IoT central, où j'ai configuré une logique métier précise pilotant des actionneurs via des identifiants uniques (IoT ID). L'intelligence du bâtiment est assurée par des scripts JavaScript intégrés aux capteurs qui forcent des cycles de données dynamiques : le détecteur de fumée active la sirène (IoT17) dès que le niveau atteint **50**, le capteur d'humidité verrouille les portes et fenêtres (IoT3, IoT4) au-delà de **70%** pour simuler la pluie, et le moniteur de température régule le ventilateur (IoT10) et la climatisation (IoT5) selon trois paliers critiques (29°C, 37°C et 16°C). Cette architecture garantit une gestion automatisée du confort et de la sécurité, supervisée en temps réel par l'administrateur via une interface mobile (Smartphone/Tablette), validant ainsi la viabilité technique d'une "Smart School" indépendante d'Internet.

IV. Conclusion générale

Ce projet a permis de concevoir et de simuler un système IoT intelligent dédié à la gestion d'un établissement scolaire, intégrant à la fois la surveillance environnementale, l'automatisation des équipements et la supervision centralisée. À travers une approche progressive, le travail réalisé couvre l'ensemble de la chaîne IoT, depuis la programmation embarquée jusqu'à l'infrastructure réseau.

La plateforme **Wokwi** a été utilisée pour valider la partie matérielle du système. Elle a permis de simuler un **ESP32 réel**, connecté à des capteurs physiques et à des actionneurs, et d'implémenter une logique embarquée efficace en **C++**. Cette étape a permis de vérifier le bon fonctionnement des lectures de capteurs, la gestion des seuils, l'automatisation locale

ainsi que le formatage et l'exportation des données sous forme de trames **JSON**, conformément aux standards de l'IoT.

En complément, **Cisco Packet Tracer** a été utilisé pour simuler une **infrastructure réseau IoT complète** à l'échelle d'un établissement scolaire. Contrairement à Wokwi, cette plateforme repose sur des **objets IoT abstraits** et des **scripts en JavaScript**, permettant de définir des règles d'automatisation, de gérer les communications réseau et de superviser l'ensemble du système via une Home Gateway et une interface utilisateur dédiée.

Ainsi, les deux outils utilisés dans ce projet se révèlent parfaitement complémentaires :

Wokwi	**Cisco Packet Tracer**
Simulation matérielle	Simulation réseau
ESP32 réel	Objets IoT abstraits
Programmation en C++	Scripts en JavaScript
Logique embarquée	Logique d'automatisation
Capteurs physiques	Infrastructure IoT

Wokwi permet de valider le fonctionnement interne du microcontrôleur et des capteurs, tandis que Cisco Packet Tracer permet de modéliser et de tester l'intégration réseau, la communication entre les objets et la supervision globale du système. Cette complémentarité offre une vision complète et cohérente d'un projet IoT, depuis le niveau matériel jusqu'au niveau applicatif et réseau.

En conclusion, ce projet démontre la faisabilité et l'efficacité d'une solution IoT pour une école intelligente, tout en mettant en évidence l'importance de combiner des outils de simulation adaptés pour couvrir l'ensemble des aspects techniques d'un système connecté moderne.