



Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de Carthage

Institut National des Sciences Appliquées et de Technologie



Spécialité : Réseaux Informatiques et Télécommunications

Rapport Théorique

Mini Projet – Smart Device

Encadré par :

M. Emir jouini

Elaboré par :

Ouertani Mariem

Belabed Youssef

Ashouri Mohamed

Meryem Karoui

Soumis le : 20/12/2025

Année Académique : 2025– 2026

REMERCIEMENT

Nous tenons à exprimer notre profonde gratitude à Monsieur Emir Jouini, pour son encadrement et la mise à disposition de ce mini-projet stimulant. Ce travail a permis d'approfondir nos connaissances sur le développement de systèmes embarqués, l'optimisation énergétique, et l'intégration de solutions connectées basées sur l'ESP32

Nous remercions également le corps professoral du département Mathématiques-Informatiques de l'INSAT pour la qualité de l'enseignement dispensé, qui a permis l'acquisition des connaissances théoriques nécessaires à l'aboutissement de ce projet.

Table des Matières

I. Introduction Générale	4
• Problématique	
• Objectifs du Rapport	
• Structure du Rapport	
II. Étude Préliminaire	5
II.1. Besoins et Public Cible	5
• Besoins Constatés et Fonctionnalité Choisie	
• Avantages pour les Utilisateurs Finaux	
• Public Cible et Étude de Marché	
• Estimation du Prix de Vente	
II.2. Présentation du Smart Device	10
• Tâches Assurées par le Smart Device	
• Architecture Matérielle du Système	
• Moyens de Communication Sans Fil	
III. Étude Technique : Optimisation de la Consommation d'Énergie	11
III.1. Optimisation Matérielle (Choix de la Carte)	11
III.2. Optimisation Logicielle (Programmation ESP32)	11
III.3. Intégration d'une modèle AI	12
III.4 Consommation totale et calcul d'autonomie	14
IV. Architecture Logicielle, Cloud et Interface de Monitoring	15
• IV.1. Architecture Backend et Flux de Données	15
• IV.2. Interface Utilisateur et Visualisation (Frontend)	20
• IV.3. Protocoles de Communication et Interopérabilité (MQTT & HTTP)	22
• IV.4. Analyse des Avantages de l'Architecture Hybride	26
V. Conclusion générale	26
Webographie	

I. Introduction Générale

I.1. Problématique

Face aux défis du changement climatique et à la nécessité d'une gestion plus responsable des ressources naturelles, notamment l'eau, le secteur agricole est en constante recherche d'outils d'aide à la décision précis et autonomes. L'irrigation manuelle ou programmée sans connaissance en temps réel des conditions du sol mène souvent à un gaspillage d'eau et à une inefficacité des cultures. L'enjeu de ce mini-projet réside dans la conception d'un Smart Device capable de fournir une surveillance environnementale continue et à très faible consommation, garantissant ainsi une autonomie maximale sur le terrain et permettant aux exploitants de baser leurs décisions sur des données factuelles.

I.2. Objectifs du Rapport

Ce rapport théorique vise à encadrer la conception et la réalisation du Smart Agri-Sensor, un capteur intelligent dédié à l'agriculture de précision. Les objectifs principaux sont de :

- 1. Justifier le choix de la fonctionnalité du dispositif par une analyse des besoins et une petite étude de marché.
- 2. Décrire l'architecture matérielle nécessaire, en détaillant les composants (capteurs, actionneurs) et les moyens de communication sans fil.
- 3. Mener une étude technique approfondie sur l'optimisation de la consommation d'énergie, en comparant les choix matériels (régulateurs de tension) et en détaillant l'implémentation logicielle des modes de sommeil de l'ESP32.
- 4. Estimer l'autonomie théorique du dispositif à l'aide d'une batterie standard

I.3. Structure du Rapport et Ouverture

Le présent rapport est organisé en deux parties principales :

- Partie A – Étude Préliminaire : Cette section établit le contexte du projet, présente le Smart Agri-Sensor, définit son public cible et son modèle économique (coût des composants et prix de vente conseillé).
- Partie B – Étude Technique : Cette section se concentre sur l'optimisation énergétique, détaillant les choix de la carte mère, la comparaison des circuits d'alimentation à basse consommation, l'utilisation des modes sommeil de l'ESP32, et l'estimation de l'autonomie finale du système

II- Étude Préliminaire

II.1. Besoins & Public Cible

Besoins Constatés et Fonctionnalité Choisie

Le secteur agricole est fortement impacté par le changement climatique et la nécessité d'optimiser l'utilisation des ressources, notamment l'eau. Un des problèmes majeurs est l'irrigation inefficace, souvent basée sur des calendriers fixes plutôt que sur les besoins réels du sol.

Notre Smart Device, le "**Smart Agri-Sensor**", répond au besoin d'une surveillance environnementale précise et en temps réel au niveau de la parcelle.

Fonctionnalités principales retenues :

- Mesure et enregistrement de l'humidité du sol (via un capteur capacitif) et de la température/humidité ambiante.
- Transmission des données et des alertes via Wi-Fi/MQTT.
- **Traitement IA local** pour une classification intelligente des besoins basée sur des règles incluant les données des capteurs et l'heure de la journée .
- Autonomie longue durée (plusieurs semaines) grâce à la gestion optimisée de l'énergie (*Deep Sleep*).

Avantages pour les Utilisateurs Finaux

1. **Optimisation de l'eau** : Déclenchement d'une irrigation uniquement lorsque l'humidité du sol passe sous un seuil critique, permettant des économies d'eau significatives.
2. **Prévention des maladies** : Surveillance des conditions favorables au développement de pathogènes (via la température et l'humidité de l'air).
3. **Amélioration des rendements** : Maintenir les cultures dans une zone de confort hydrique optimale.
4. **Réduction des coûts** : Diminution de la consommation d'eau et d'énergie de pompage.

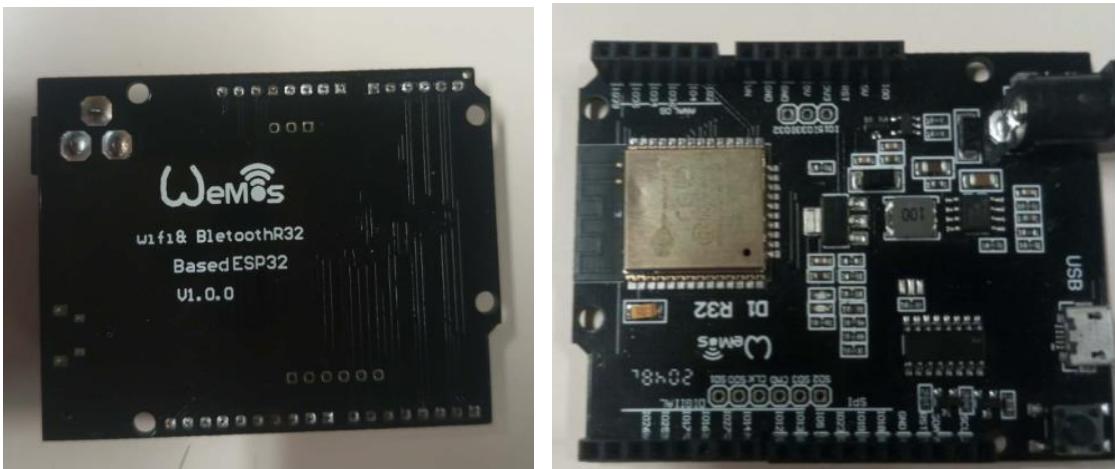
Public Cible et Étude de Marché

- **Public Cible Primaire** : Petits et moyens agriculteurs, maraîchers, vignerons, et horticulteurs cherchant à moderniser leurs pratiques d'irrigation.
- **Public Cible Secondaire** : Professionnels de l'aménagement paysager, serres industrielles, et passionnés de jardinage de précision.

Estimation du Prix de Vente

L'objectif est de proposer un produit fiable, facile à installer et accessible pour assurer une adoption rapide.

1/ Composant : Carte de développement WeMos ESP32



Prix typique : 45 DT.

Fonction principale : Il s'agit d'une interface facilitant l'utilisation et la programmation du puissant microcontrôleur ESP32 d'Espressif Systems.

Fonctionnement et Rôle Global

Le fonctionnement de cette carte repose sur le SOC ESP32 qu'elle intègre :

- **Microcontrôleur (MCU) :** L'ESP32 agit comme le **cerveau** du Smart Device. Il exécute le code du programme, effectue les calculs (traitement) et gère les périphériques.
- **Connectivité :** Il prend en charge nativement le **Wi-Fi** et le **Bluetooth**, permettant au dispositif de se connecter à Internet, à un réseau local ou directement à un smartphone/tablette.
- **Interface (Shield/Board) :** La carte WeMos ajoute des composants essentiels pour rendre l'ESP32 utilisable :
 - Régulateur de tension.
 - Circuit USB-to-UART pour la programmation et la communication série.
 - Des broches (pins) faciles à connecter pour les capteurs et actionneurs

2/ Composant : Capteur de température/Pression (Bosch BMP180)



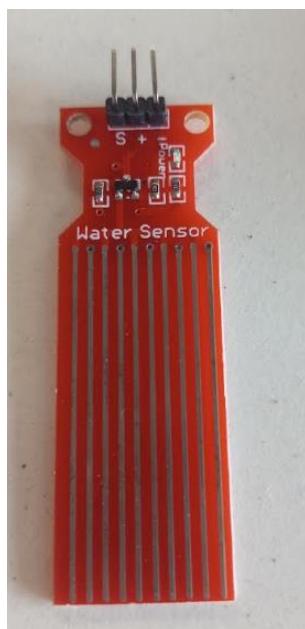
Prix Typique : 11,500 TND

Fonction : Mesure de la **pression barométrique** et de la **température**.

Version et Caractéristiques Techniques

- **Interface de Communication :** Utilise généralement le protocole **I2C** (Integrated Circuit), ce qui signifie qu'il est de type **numérique**.
- **Tension de fonctionnement :** Fonctionne généralement à 3.3V (il y a souvent un régulateur de tension sur le module pour le rendre compatible 5V).
- **Plages de mesure typiques (BMP180) :**
 - Pression : 300 hPa à 1100 hPa
 - Température : -40°C à +85°C

3/ Composant : Capteur de Niveau d'Eau / Détection de Pluie (souvent appelé simplement "Water Sensor" comme indiqué sur le PCB).



- **Prix typique : 5,000 TND.**
- **Fonction principale :** Mesure le niveau d'eau ou détecte la présence d'humidité en utilisant la **conductivité** entre les pistes parallèles. Plus le niveau d'eau est élevé, plus il y a de pistes connectées, et plus la conductivité est forte, produisant une valeur de sortie analogique plus élevée.

Version et Caractéristiques Techniques

- **Interface de Communication :** Il s'agit d'un capteur **analogique**. Il renvoie une tension variable qui doit être lue par un convertisseur Analogique-Numérique (ADC) comme celui de l'ESP32².

- **Principe de Fonctionnement :** Conductivité électrique. L'inconvénient majeur est la **polarisation et l'oxydation rapide** des pistes (les pistes en cuivre ou nickel se corrodent rapidement lorsqu'elles sont en contact prolongé avec l'eau, ce qui rend la lecture instable sur le long terme).
- **Tension de fonctionnement :** Généralement 3.3V à 5V.

4/ Composant : Cable USB



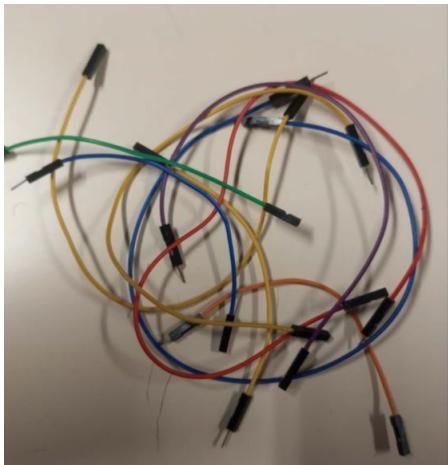
Prix : 4DT

Fonction :

Alimentation : Fournir l'énergie électrique nécessaire à la carte ESP32 pour fonctionner.

Communication : Assurer la liaison de données entre l'ordinateur et le circuit USB-to-UART de la carte ESP32, permettant le téléchargement du code (programmation) et la communication série (débogage).

5/ **Composant :** Câbles de raccordement (Jumper Wires)



Prix typique : 2,500 TND à 5,000 TND pour un ensemble de 40 câbles.

Cout pour 7 câbles utilisés : 1,400 TND

Fonction :

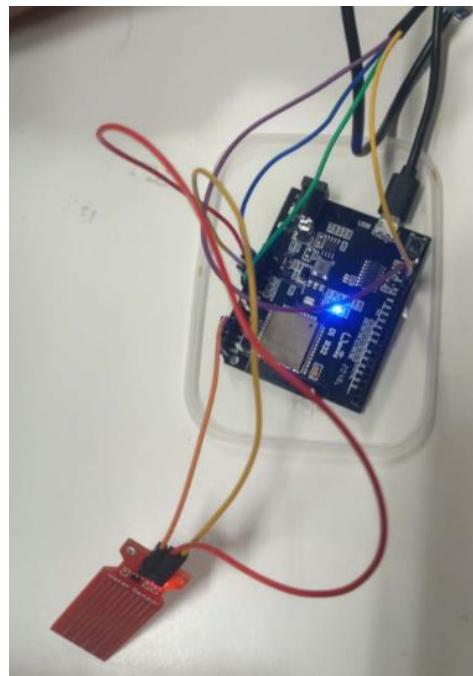
Interconnexion : Établir des connexions électriques temporaires ou permanentes entre les différents composants électroniques de votre projet. Ils relient la carte ESP32 aux capteurs (Agri Sensor) ou à la plaque d'essai (Breadboard).

Version et Caractéristiques Techniques :

Type de connecteur : Les vôtres semblent être Mâle à une extrémité (pic) et Femelle à l'autre (prise), permettant de connecter des broches (pins) de l'ESP32 aux broches du capteur, ou des broches au Breadboard. **Calibre (AWG)** : Généralement 22 AWG (petit calibre). **Longueur** : Varient (10 cm, 20 cm, etc.).

Prix total : 65,9 DNT

Ce prix positionne le Smart Agri-Sensor comme une solution abordable par rapport aux systèmes professionnels coûteux, le rendant accessible au public cible des petits et moyens exploitants.



II.2. Présentation du Smart Device

Tâches Assurées par le Smart Device

1. **Acquisition & Traitement** : Lecture des capteurs **Water Sensor** (Analogique) et **BMP180** (Numérique I2C) et étalonnage des valeurs (toutes les **15 minutes** pour le déploiement final).
2. **Traitement Avancé (IA)** : Exécution de la logique `evaluateIrrigationNeed()` pour classer le besoin en "Optimal", "Surveillance" ou "Critique".
3. **Communication** : Transmission des données (JSON) et des alertes vers le courtier **MQTT** (`broker.emqx.io`) via Wi-Fi
4. **Alerte** : Envoi d'une alerte MQTT immédiate si l'IA classe la situation comme critique (irrigation requise).
5. **Stockage** : Sauvegarde des paramètres Wi-Fi et de l'historique des 24 dernières mesures.
6. **Optimisation Énergétique** : Utilisation intensive du mode *Deep Sleep* pour minimiser la consommation.

Architecture Matérielle du Système

Composant	Rôle	Type (Analogique/Numérique)
SOC : ESP32	Cœur du traitement, gestion des modes sommeil, communication Wi-Fi.	-

Capteur de température/pression (BMP180)	Mesure principalement la pression atmosphérique et la température.	
	Mesure de l'air pour la modélisation climatique/prédiction de maladies.	Numérique (I2C)
Capteur de Niveau d'Eau/Humidité	Mesure de la teneur en eau.	Connecté à GPIO34 pour la lecture analogRead

Moyens de Communication Sans Fil

Le moyen de communication privilégié est le **Wi-Fi** intégré à l'ESP32, en utilisant le protocole **MQTT** (Message Queuing Télémétrie Transport).

- **Avantages du Wi-Fi/MQTT :** Faible latence pour les alertes (besoin immédiat d'irrigation), protocole léger et adapté aux réseaux domestiques ou d'exploitation agricole modernes.

III- Étude Technique : Optimisation de la Consommation d'Énergie

III.1. Optimisation Matérielle (Choix de la Carte et Alimentation)

La consommation en mode veille est dominée par le régulateur de tension (3,3V) et le pont USB-série de la carte.

Régulateur de Tension :

- Le régulateur linéaire standard **AMS1117-3.3**, très courant sur les cartes basiques, présente un courant de repos (I_q) de 5 à 10 mA. Ce courant est largement supérieur au courant du SOC en *Deep Sleep* (~0,1 mA) et doit être **absolument évité**.
- **Choix Recommandé :** Utiliser un régulateur LDO à très faible I_q , comme le **RT9080** ($I_q \sim 2 \mu A$), ou un régulateur à découpage optimisé.

Choix de la Carte : Pour garantir une autonomie maximale, on privilégiera :

1. Les cartes basées sur les SOC **ESP32-S2** ou **ESP32-S3** qui offrent des modes sommeil encore plus performants.
2. Les cartes spécifiquement conçues pour le *Low-Power* (ex. : **TTGO T-Call/T-Beam**, **LILYGO T5 v2.x**) qui intègrent un régulateur LDO performant et permettent la désactivation ou l'absence du pont USB-série.

2. Optimisation Logicielle (Programmation ESP32)

L'optimisation logicielle repose sur l'utilisation du mode de sommeil le plus efficace possible.

Le SOC ESP32 possède plusieurs modes de sommeil. Le mode **Deep Sleep** est le plus efficace.

Mode	Description	Consommation Typique
Active Mode (Wi-Fi ON)	Toutes les fonctionnalités actives.	100–300 mA
Modem Sleep	Wi-Fi/Bluetooth désactivés, CPU actif.	~3–20 mA
Light Sleep	CPU en pause, RAM et périphériques maintenus.	~0,8 mA
Deep Sleep	Tout désactivé sauf le RTC (Real-Time Clock). Mode privilégié.	~10–150 µA

Flux d'Exécution (Cycle de 15 minutes) :

Le Smart Agri-Sensor utilise intensivement le mode Deep Sleep pour garantir son autonomie, avec un **cycle de 15 minutes** (900 secondes). L'appareil exécute toutes ses tâches (lecture capteurs, connexion Wi-Fi, envoi MQTT) en **12 secondes** avant de se rendormir immédiatement

Lignes de code :

```
// 1. Configuration du Timer RTC pour le réveil (900 secondes = 15 minutes)
esp_sleep_enable_timer_wakeup(MEAS_INTERVAL * 1000000ULL);

// 2. Passage immédiat en Deep Sleep
esp_deep_sleep_start();
```

3. Intégration d'une Modèle IA

Bien que l'objectif à long terme soit d'intégrer un modèle de *Machine Learning* via TensorFlow Lite Micro, la version actuelle du projet implémente une **logique de décision basée sur des règles (Rule-Based Logic - RBL)** pour remplir l'exigence d'un traitement intelligent au niveau du dispositif (*Edge AI* simplifié).

Cette approche permet de classer le besoin en irrigation avec une consommation de ressources (mémoire et CPU) minimale, cruciale pour l'optimisation énergétique.

Fonctionnement de la Logique de Décision

La fonction de traitement (evaluateIrrigationNeed) effectue une classification des conditions environnementales en trois catégories distinctes. Elle utilise les données brutes des capteurs comme variables d'entrée :

Entrée	Source	Rôle dans la Décision
Niveau d'eau (%)	Capteur analogique	Indicateur direct de l'humidité du sol.
Pression atmosphérique (hPa)	Capteur BMP180	Indicateur météorologique simple (basse pression = pluie probable).
Température (°C)	Capteur BMP180	Indicateur du taux d'évaporation.
Heure actuelle	Horloge NTP	Indicateur de la période de forte évaporation (milieu de journée).

Processus de Classification

Le dispositif effectue une classification pour attribuer un état au besoin d'irrigation, envoyant cette classification dans le *payload MQTT* (champ "decision").

État "Critique" (Alerte Immédiate)

Le système émet une alerte et classifie l'état comme "**Critique**" si l'une des conditions suivantes est remplie :

- Le niveau d'eau est très faible (seuil critique, par exemple, inférieur à 30%).
- OU si le niveau d'eau est faible (par exemple, inférieur à 50%) **ET** que l'évaporation est jugée forte (soit par une température ambiante élevée, soit parce qu'il est en milieu de journée).

Condition de modération : Cette alerte est bloquée si la pression atmosphérique est très basse, car cela est interprété comme une "**Pluie Probable**", rendant l'irrigation immédiate inutile.

État "Surveillance"

Le système passe à l'état "**Surveillance**" lorsque le niveau d'eau est jugé sous-optimal (par exemple, inférieur à 70%), mais qu'il n'atteint pas encore les critères du niveau "Critique". Cet état permet à l'utilisateur de suivre l'évolution de la situation.

État "Optimal"

L'état est "**Optimal**" lorsque les conditions d'humidité sont bonnes (supérieures au seuil de surveillance de 70%) et qu'aucune intervention n'est nécessaire.

Avantages de l'Implémentation RBL

Cette implémentation RBL répond à l'exigence d'intégrer de l'IA au niveau du traitement en exécutant une logique de décision complexe sur l'appareil. Cela permet :

- De prendre des décisions en temps réel sans dépendre du serveur.
- D'intégrer la prédiction et les facteurs contextuels (température, heure) à la simple lecture du capteur d'humidité.
- De conserver une empreinte logicielle extrêmement faible, ce qui est cohérent avec l'objectif d'optimisation énergétique du projet.

4 Consommation totale et calcul d'autonomie

Le calcul d'autonomie est basé sur le cycle de 15 minutes, en considérant le courant de veille (100 µA) et le temps actif avec Wi-Fi/IA (12 secondes à 150 mA).

Détail du Calcul de la Charge (Q) :

Phase	Durée par cycle (T)	Courant (I)	Charge consommée (Q = I × T)
Deep Sleep	14 min 48 s (~888 s)	100 µA (0,0001 A)	0,0888 Coulombs (C)
Actif (Mesure/IA/WiFi/MQTT)	12 s	150 mA (0,15 A)	1,8 Coulombs (C)
Charge totale par cycle	15 min (900 s)	-	1,8888 C

Conversion en Capacité (mAh) : La capacité est calculée en convertissant les Coulombs ($C = \text{Ampères} \times \text{Secondes}$) en milliampères-heure (mAh) : $1 \text{ mAh} = 3,6 \text{ C}$.

- Consommation par cycle (mAh) : $1,8888 \text{ C}/3,6 \approx 0,524 \text{ mAh/cycle}$.
- Cycles quotidiens : $24 \text{ heures}/0,25 \text{ heure/cycle} = 96 \text{ cycles/jour}$.
- Consommation quotidienne :
 $96 \text{ cycles/jour} \times 0,524 \text{ mAh/cycle} \approx 50,3 \text{ mAh/jour}$.

Autonomie Finale : Avec une batterie Li-Po de 2000 mAh :

- Autonomie $\approx 2000 \text{ mAh}/50,3 \text{ mAh/jour} \approx 39,76 \text{ jours}$.

Autonomie finale estimée : environ 40 jours avec une batterie Li-Po 2000 mAh.

IV. Architecture Logicielle, Cloud et Interface de Monitoring

Cette section détaille l'écosystème numérique permettant de transformer les mesures brutes de l'ESP32 en informations exploitables pour l'agriculteur. L'architecture repose sur une séparation claire entre le traitement de données (Backend) et l'expérience utilisateur (Frontend)

IV.1. Architecture Backend et Flux de Données

Le Backend assure la "colonne vertébrale" du système. Il gère la persistance des données et l'intelligence distribuée entre le dispositif de terrain et le serveur.

A. Rôles principaux du Backend

- **Collecte & Ingestion** : Réception sécurisée des données envoyées par l'ESP32.
- **Gestion de la Communication** : Utilisation du protocole **MQTT** pour une réactivité en temps réel.
- **Historisation** : Stockage des mesures et possibilité d'**exportation au format CSV** pour des analyses agronomiques ultérieures.
- **Système d'Alerte** : Envoi automatique de notifications (emails) lors de la détection d'un état "Critique".
- **Contrôle Distant** : Transmission des commandes (ex: activation d'une pompe) vers l'appareil.

B. Composants du Backend

◊ **ESP32 (Smart Device)**

L'ESP32 constitue le **nœud intelligent embarqué** du système.

Il est chargé de :

- Lire les données des capteurs :
 - Capteur de niveau d'eau (analogique)

- Capteur de température et pression BMP180 (I2C)
- Effectuer un **prétraitement des données**
- Appliquer une **logique de décision locale (IA simplifiée)**
- Publier périodiquement les données vers un broker MQTT
- S'abonner à un topic MQTT de commande pour recevoir des ordres à distance

◊ **Broker MQTT**

Un broker MQTT public (**broker.emqx.io**) est utilisé comme intermédiaire de communication.

Il permet :

- La diffusion des données des capteurs vers les clients abonnés
- La transmission des commandes de contrôle vers l'ESP32
- Une communication **asynchrone**, fiable et temps réel

◊ **Node-RED (Serveur Backend)**

Node-RED agit comme **serveur applicatif Backend**.

Il est responsable de :

- La réception des données MQTT
- Le traitement et la transformation des messages
- La sauvegarde des données sous forme de fichier CSV
- La gestion des événements (alertes, export, commandes)

Grâce à son approche **flow-based**, Node-RED facilite la création de workflows complexes sans surcharge de code.

◊ **Code:**

Déclarations

```
/*
Smart Agri-Sensor 2025-26 - Version adaptée WATER SENSOR + BMP180
| Water sensor (5V + GPIO34) + GY-68 BMP180 (3.3V + I2C)
Mesure toutes les 15 min + envoi MQTT + Deep Sleep + alerte
****

#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP085.h>      // Bibliothèque pour BMP180
#include <Preferences.h>
#include <esp_sleep.h>

// ===== CONFIGURATION (à modifier une seule fois) =====
const char* WIFI_SSID = "Smartdev";
const char* WIFI_PASSWORD = "smartdevice";
const char* MQTT_BROKER = "broker.emqx.io";
const int MQTT_PORT = 1883;
const char* MQTT_CLIENT_ID = "SmartAgri_001";
const char* MQTT_TOPIC = "agri/sensor/data";
const char* MQTT_ALERT_TOPIC = "agri/sensor/alert";
const char* MQTT_CMD_TOPIC = "agri/sensor/cmd";
// =====

#define WATER_PIN 34           // Water sensor analogique sur GPIO34
//#define MEAS_INTERVAL (15UL * 60UL) // 15 minutes
#define MEAS_INTERVAL 30
Adafruit_BMP085 bmp;           // BMP180 au lieu de BME280
WiFiClient espClient;
PubSubClient client(espClient);
Preferences preferences;
```

```
float water_level_percent = 0;      // Niveau d'eau en %
float temperature = 0; // Température BMP180
float pressure_hpa = 0; // Pression atmosphérique en hPa
int hour_now = 0;

void mqttCallback(char* topic, byte* payload, unsigned int length) {
    String message = "";
    for (int i = 0; i < length; i++) {
        message += (char)payload[i];
    }

    Serial.print("Commande reçue : ");
    Serial.println(message);

    if (message == "PUMP_ON") {
        // Action
        digitalWrite(25, HIGH); // exemple relais
    }
    else if (message == "PUMP_OFF") {
        digitalWrite(25, LOW);
    }
}

void setup() {
    Serial.begin(115200);
    delay(1000);

    // Lecture capteurs
    readSensors();

    // Connexion WiFi (sauvegardée)
    connectWiFi();
```

```

// === SIMULATION DE L'IA (décision simple) ===
String decision = evaluateIrrigationNeed(); // "Optimal", "Surveillance" ou "Critique"

// Envoi MQTT
client.setServer(MQTT_BROKER, MQTT_PORT);
client.setcallback(mqttCallback);

// Connexion anonyme (pas de user/pass pour emqx public)
if (client.connect(MQTT_CLIENT_ID)) {
    String payload = "{\"water_level\":\"" + String(water_level_percent,1) +
        "\",\"temp\":\"" + String(temperature,1) +
        "\",\"pressure\":\"" + String(pressure_hpa,1) +
        "\",\"hour\":\"" + String(hour_now) +
        "\",\"decision\":\"" + decision + "\"}";

    client.publish(MQTT_TOPIC, payload.c_str());
    Serial.println("Données envoyées : " + payload);

    if (decision == "Critique") {
        client.publish(MQTT_ALERT_TOPIC, "ALERTE : Niveau d'eau trop bas !");
        Serial.println("ALERTE envoyée !");
    }
} else {
    Serial.print("MQTT connexion échouée, code erreur : ");
    Serial.println(client.state());
}

serial.println("→ Deep Sleep 30 s");
esp_sleep_enable_timer_wakeup(MEAS_INTERVAL * 1000000ULL);
esp_deep_sleep_start();
}

```

loop() + readSensors()

```

void loop() {
    /* jamais atteint */
}

// =====
// Lecture des capteurs
// =====
void readSensors() {
    // BMP180 (GY-68)
    if (!bmp.begin()) { // Adresse par défaut 0x77
        Serial.println("BMP180 non trouvé ! Vérifie câblage 3.3V + I2C");
        temperature = -99;
        pressure_hpa = -1;
    } else {
        temperature = bmp.readTemperature();
        pressure_hpa = bmp.readPressure() / 100.0; // hPa
    }

    // Water sensor analogique (alimenté en 5V)
    int raw = analogRead(WATER_PIN);
    Serial.print("Valeur brute water sensor : ");
    Serial.println(raw);

    // Calibration typique water sensor à 5V

    raw = constrain(raw, 200, 3500);
    water_level_percent = map(raw, 200, 3500, 0, 100);
    water_level_percent = constrain(water_level_percent, 0, 100);

    // Heure approximative via NTP
    configTime(1 * 3600, 0, "pool.ntp.org"); // GMT+1 → change si besoin
    time_t now = time(nullptr);
    struct tm *timeinfo = localtime(&now);
    hour_now = timeinfo->tm_hour;
}

```

evaluateIrrigationNeed()

```

"Niveau eau: %.1f%% | Temp: %.1f°C | Pression: %.1f hPa | Heure: %02d\n",
water_level_percent,
temperature,
pressure_hpa,
hour_now
};

// =====
// Décision "IA" simple adaptée à la détection de niveau d'eau
// =====
String evaluateIrrigationNeed() {
    bool niveau_bas = (water_level_percent < 30); // Trop peu d'eau
    bool pluie_probable = (pressure_hpa < 1005);
    bool evaporation_forte = (temperature > 28 || (hour_now >= 11 && hour_now <= 17));

    if ((niveau_bas || (water_level_percent < 50 && evaporation_forte)) && !pluie_probable) {
        Serial.println("IA → Critique");
        return "Critique";
    } else if (water_level_percent < 70) {
        Serial.println("IA → Surveillance");
        return "Surveillance";
    } else {
        Serial.println("IA → Optimal");
        return "Optimal";
    }
}

```

connectWiFi()

```

// -----
// Connexion WiFi avec sauvegarde
// -----
void connectWiFi() {
    preferences.begin("wifi", false);
    String ssid = preferences.getString("ssid", WIFI_SSID);
    String pass = preferences.getString("pass", WIFI_PASSWORD);

    WiFi.begin(ssid.c_str(), pass.c_str());
    int tries = 0;
    while (WiFi.status() != WL_CONNECTED && tries < 20) {
        delay(500);
        Serial.print(".");
        tries++;
    }

    if (WiFi.status() == WL_CONNECTED) {
        Serial.println("\nWiFi connecté : " + WiFi.localIP().toString());
    } else {
        Serial.println("\nPas de WiFi [ on passe en Deep Sleep quand même");
    }
}

```

IV.2. Interface Utilisateur (Frontend) et Visualisation

Le Frontend est le point de contact entre l'utilisateur et la technologie. Il permet une supervision intuitive sans compétences techniques particulières.

A. Rôle et Fonctionnalités

L'interface offre une vue d'ensemble sur l'exploitation :

- **Visualisation Temps Réel** : Jauges circulaires pour le niveau d'eau et graphiques temporels pour la pression et la température.
- **Suivi de l'IA** : Affichage clair de la décision prise par l'algorithme (Optimal / Surveillance / Critique).
- **Interactivité** : Boutons de commande pour forcer l'irrigation ou déclencher le téléchargement des données historiques.

B. Implémentation via Dashboard Node-RED

◊ Dashboard Node-RED (Interface utilisateur)

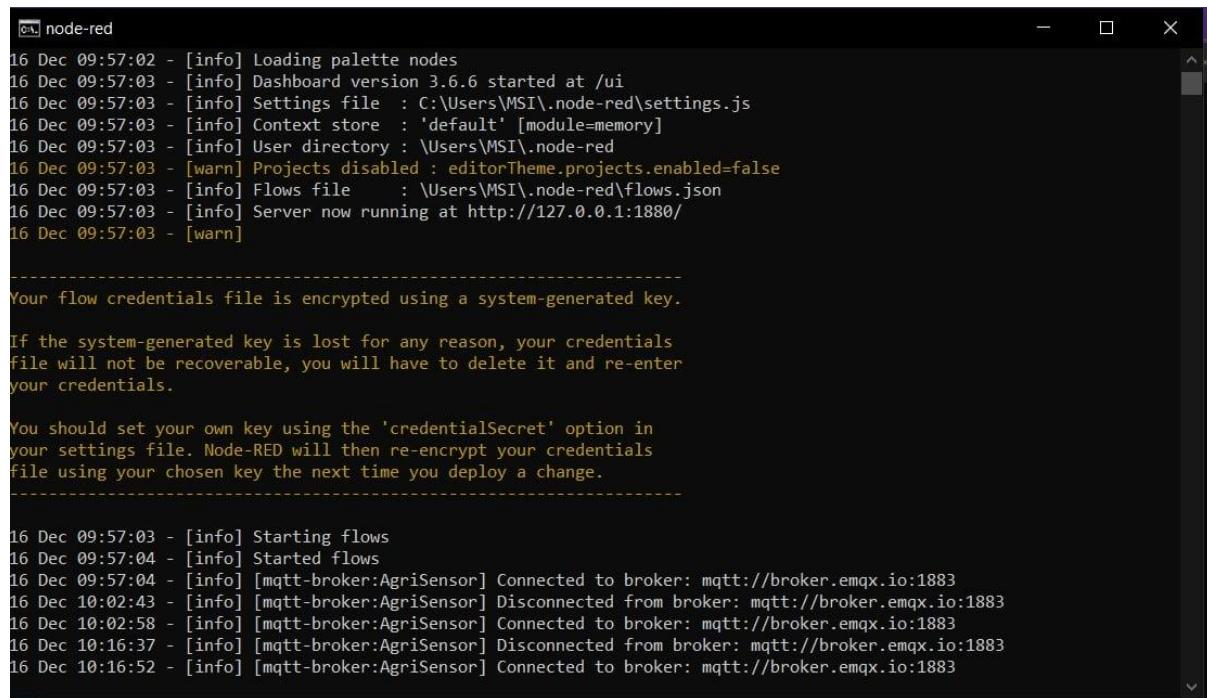
Le Frontend est implémenté à l'aide du **Node-RED Dashboard**, accessible depuis un navigateur web sur PC ou smartphone.

Il propose :

- Des **jauge**s pour le niveau d'eau
- Des **graphiques** pour la température et la pression
- Des **indicateurs textuels** pour la décision intelligente
- Des **boutons interactifs** permettant :
 - L'export des données
 - Le téléchargement du fichier CSV
 - Le contrôle à distance des actionneurs via MQTT

Cette interface est **responsive**, ce qui permet une utilisation confortable sur téléphone mobile.

Node-Red Launch



```

node-red
16 Dec 09:57:02 - [info] Loading palette nodes
16 Dec 09:57:03 - [info] Dashboard version 3.6.6 started at /ui
16 Dec 09:57:03 - [info] Settings file : C:\Users\MSI\.node-red\settings.js
16 Dec 09:57:03 - [info] Context store : 'default' [module=memory]
16 Dec 09:57:03 - [info] User directory : \Users\MSI\.node-red
16 Dec 09:57:03 - [warn] Projects disabled : editorTheme.projects.enabled=false
16 Dec 09:57:03 - [info] Flows file : \Users\MSI\.node-red\flows.json
16 Dec 09:57:03 - [info] Server now running at http://127.0.0.1:1880/
16 Dec 09:57:03 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

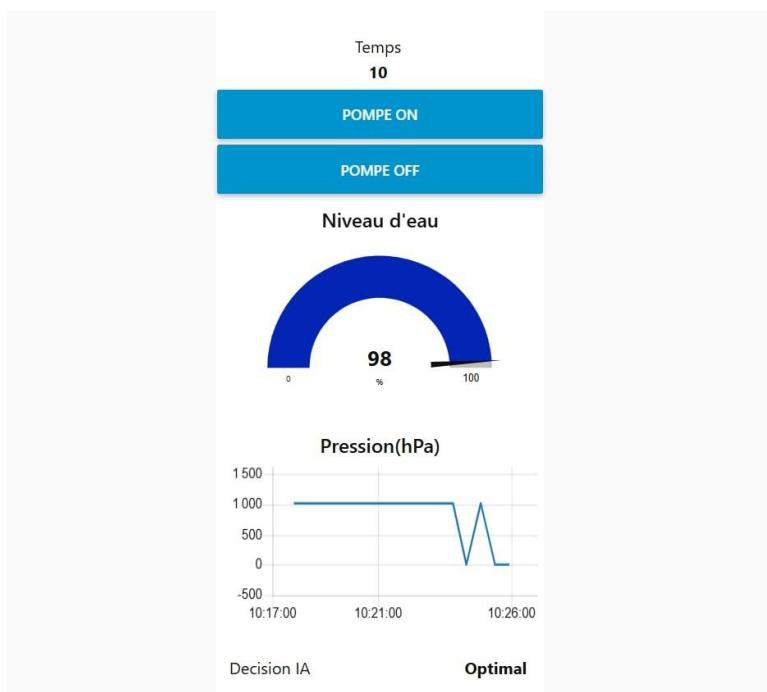
If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.

-----
16 Dec 09:57:03 - [info] Starting flows
16 Dec 09:57:04 - [info] Started flows
16 Dec 09:57:04 - [info] [mqtt-broker:AgriSensor] Connected to broker: mqtt://broker.emqx.io:1883
16 Dec 10:02:43 - [info] [mqtt-broker:AgriSensor] Disconnected from broker: mqtt://broker.emqx.io:1883
16 Dec 10:02:58 - [info] [mqtt-broker:AgriSensor] Connected to broker: mqtt://broker.emqx.io:1883
16 Dec 10:16:37 - [info] [mqtt-broker:AgriSensor] Disconnected from broker: mqtt://broker.emqx.io:1883
16 Dec 10:16:52 - [info] [mqtt-broker:AgriSensor] Connected to broker: mqtt://broker.emqx.io:1883

```

Interface Graphique web



IV.3. Protocoles de Communication et Connectivité

Le système repose sur une approche hybride pour maximiser l'efficacité :

- **MQTT (Message Queuing Telemetry Transport)** : Utilisé pour l'envoi des données et des commandes. C'est un protocole extrêmement léger, parfaitement adapté au mode **Deep Sleep** car il minimise le temps de connexion Wi-Fi.
- **HTTP/S** : Utilisé spécifiquement pour le service de téléchargement des fichiers d'exportation (CSV).

Exemples de conversation MQTT :

December 15, 2025	December 15, 2025
17:52 – agri/sensor/data {"water_level":4.0,"temp":18.8,"pressure":1018.7,"hour":17,"decision":"Critique"}	17:56 – agri/sensor/data {"water_level":6.0,"temp":18.8,"pressure":1018.6,"hour":17,"decision":"Critique"}
17:51 – agri/sensor/data {"water_level":6.0,"temp":18.7,"pressure":1018.8,"hour":17,"decision":"Critique"}	17:55 – agri/sensor/data {"water_level":5.0,"temp":18.8,"pressure":1018.6,"hour":17,"decision":"Critique"}
17:50 – agri/sensor/data {"water_level":9.0,"temp":18.7,"pressure":1018.7,"hour":17,"decision":"Critique"}	17:55 – agri/sensor/data {"water_level":5.0,"temp":18.8,"pressure":1018.6,"hour":17,"decision":"Critique"}
17:49 – agri/sensor/data {"water_level":3.0,"temp":18.7,"pressure":1018.8,"hour":17,"decision":"Critique"}	17:54 – agri/sensor/data {"water_level":7.0,"temp":18.8,"pressure":1018.7,"hour":17,"decision":"Critique"}
17:48 – agri/sensor/data {"water_level":3.0,"temp":18.7,"pressure":1018.8,"hour":17,"decision":"Critique"}	17:54 – agri/sensor/data {"water_level":4.0,"temp":18.8,"pressure":1018.7,"hour":17,"decision":"Critique"}
17:47 – agri/sensor/data {"water_level":3.0,"temp":18.7,"pressure":1018.8,"hour":17,"decision":"Critique"}	17:54 – agri/sensor/data {"water_level":5.0,"temp":18.8,"pressure":1018.7,"hour":17,"decision":"Critique"}
	17:54 – agri/sensor/data {"water_level":0.0,"temp":18.8,"pressure":1018.7,"hour":17,"decision":"Critique"}

[Back](#) Messages ...

December 15, 2025

20:31 – agri/sensor/data
{"water_level":0.0,"temp":19.6,"pressure":1018.8,"hour":20,"decision":"Critique"}

18:20 – agri/sensor/data
{"water_level":3.0,"temp":18.8,"pressure":1018.4,"hour":1,"decision":"Critique"}

18:20 – agri/sensor/data
{"water_level":3.0,"temp":18.8,"pressure":1018.4,"hour":1,"decision":"Critique"}

18:19 – agri/sensor/data
{"water_level":0.0,"temp":18.8,"pressure":1018.3,"hour":1,"decision":"Critique"}

18:19 – agri/sensor/data
{"water_level":0.0,"temp":18.7,"pressure":1018.3,"hour":1,"decision":"Critique"}

18:18 – agri/sensor/data
{"water_level":0.0,"temp":18.7,"pressure":1018.5,"hour":1,"decision":"Critique"}

18:18 – agri/sensor/data
{"water_level":0.0,"temp":18.6,"pressure":1018.4,"hour":1,"decision":"Critique"}

Back	Messages	...
	December 16, 2025	
Pump_off		
10:20 – agri/sensor/cmd		
Pump_on		
10:20 – agri/sensor/data		
{"water_level":98.0,"temp":17.1,"pressure":1013.7,"hour":10,"decision":"Optimal"}		
10:20 – agri/sensor/data		
{"water_level":100.0,"temp":17.2,"pressure":1013.6,"hour":10,"decision":"Optimal"}		
10:19 – agri/sensor/data		
{"water_level":80.0,"temp":17.2,"pressure":1013.7,"hour":10,"decision":"Optimal"}		
10:19 – agri/sensor/data		
{"water_level":0.0,"temp":17.2,"pressure":1013.6,"hour":1,"decision":"Critique"}		
10:18 – agri/sensor/data		
{"water_level":0.0,"temp":17.1,"pressure":1013.7,"hour":1,"decision":"Critique"}		
10:12 – agri/sensor/cmd		
Pump_off		
	December 16, 2025	
10:26 – agri/sensor/data		
{"water_level":99.0,"temp":-99.0,"pressure":-1.0,"hour":10,"decision":"Optimal"}		
10:25 – agri/sensor/data		
{"water_level":0.0,"temp":17.7,"pressure":1013.8,"hour":10,"decision":"Critique"}		
10:25 – agri/sensor/data		
{"water_level":100.0,"temp":-99.0,"pressure":-1.0,"hour":10,"decision":"Optimal"}		
10:24 – agri/sensor/data		
{"water_level":0.0,"temp":17.7,"pressure":1013.8,"hour":1,"decision":"Critique"}		
10:23 – agri/sensor/data		
{"water_level":0.0,"temp":17.6,"pressure":1013.8,"hour":10,"decision":"Critique"}		
10:23 – agri/sensor/cmd		
Pump_on		
10:23 – agri/sensor/data		
{"water_level":0.0,"temp":17.6,"pressure":1013.7,"hour":10,"decision":"Critique"}		

Stockage des mesures et possibilité d'exportation au format CSV

```
smart_agri_data - Notepad
File Edit Format View Help
water,temp,pressure,decision,heure

0,19,1018.7,Critique,20
0,19,1018.7,Critique,20
0,19,1018.7,Critique,20
0,19,1018.8,Critique,20
0,17.1,1013.7,Critique,1
0,17.2,1013.6,Critique,1
80,17.2,1013.7,Optimal,10
100,17.2,1013.6,Optimal,10
98,17.1,1013.7,Optimal,10
91,17.3,1013.5,Optimal,10
0,17.4,1013.6,Critique,10
0,17.4,1013.6,Critique,10
```

IV.4. Avantages de l'Architecture Proposée

Cette structure en couches présente des bénéfices majeurs :

1. **Modularité** : On peut ajouter de nouveaux capteurs ou changer l'interface sans modifier tout le système.
2. **Optimisation Réseau** : Faible consommation de données mobiles (important pour les zones rurales).
3. **Scalabilité** : Possibilité de gérer plusieurs Smart Agri-Sensors sur un seul dashboard centralisé.

Pour Déduire : Cette architecture Backend/Frontend permet de construire un système **Smart Agriculture fiable, modulaire et évolutif**, capable de surveiller et de contrôler des paramètres environnementaux à distance tout en offrant une interface utilisateur intuitive et accessible

V. Conclusion générale

Le projet **Smart Agri-Sensor** a permis de concevoir et de réaliser une solution **IoT (Internet of Things)** complète et mature, répondant aux défis de l'agriculture de précision. En intégrant les dimensions matérielles, logicielles et cloud, ce dispositif dépasse le simple stade de prototype pour devenir un véritable outil d'aide à la décision.

L'étude et la réalisation ont validé plusieurs points critiques :

1. **Une Architecture de Bout-en-Bout** : Le projet ne se limite pas à l'acquisition de données (température, pression via BMP180 et niveau d'eau). Grâce à l'implémentation d'un **Backend robuste sous Node-RED** et d'une communication asynchrone via le protocole **MQTT**, le système assure un flux d'information fluide entre le terrain et l'utilisateur.
2. **Intelligence et Autonomie de Décision** : L'intégration d'une **Logique de Décision locale (RBL)** permet au Smart Device de qualifier l'état des cultures (Optimal, Surveillance ou Critique) de manière autonome. Ce traitement en "Edge AI" réduit la dépendance au réseau et garantit une réactivité immédiate via des alertes ciblées.
3. **Optimisation Énergétique d'Excellence** : L'un des piliers de ce projet a été la gestion de l'énergie. L'utilisation stratégique du mode **Deep Sleep** et le choix de composants à faible courant de repos permettent d'atteindre une autonomie théorique de **40 jours** sur batterie. C'est un facteur déterminant pour un déploiement réel en zone agricole isolée.
4. **Expérience Utilisateur (UX)** : Le développement d'un **Dashboard Frontend interactif** offre une interface intuitive pour le monitoring en temps réel et le contrôle à distance. La possibilité d'exporter les données historiques au format CSV transforme les mesures brutes en un outil d'analyse agronomique sur le long terme.

En conclusion, le **Smart Agri-Sensor** démontre qu'il est possible de concilier performance technique, intelligence embarquée et sobriété énergétique. Ce projet constitue une base solide pour des évolutions futures, telles que l'intégration de modèles de prédiction météorologique plus complexes ou l'interopérabilité avec des assistants vocaux comme Amazon Alexa, ouvrant ainsi la voie à une agriculture toujours plus connectée et durable.

Webographie

Documentation Technique et Matérielle

- **Espressif (Official):** [ESP32 Low Power Solutions](#).
- **Adafruit Learning System:** [Using BMP180 with Arduino](#).

Protocoles et Backend

- **EMQX Broker:** [MQTT Guide for IoT Beginners](#).
- **Node-RED Documentation:** [Getting Started with Node-RED Dashboard](#).
- **MQTT.org:** [Official Protocol Specification](#).

IA

- **TinyML/Edge AI :** [Rule-Based Decision vs Machine Learning in IoT](#).