|  |
| --- |
| Gestion d'Arrosage Automatique |

Date de création – le 2 mai 2025.

Date de fin – le 2 juin 2025.

Réalisateur du projet - Sulejmani Mefail-CIN4B-ETML.

Chef de projet Monsieur Costa Lopes.

Expert 1 : Monsieur Bertino.

Expert 2 : Monsieur Albert.

Table des matières

[1 Analyse préliminaire 4](#_Toc198305256)

[1.1 Introduction 4](#_Toc198305257)

[*1.2* Objectifs 5](#_Toc198305258)

[*1.3* Planification initiale 6](#_Toc198305259)

[1.4 Méthodologie du projet 8](#_Toc198305260)

[2 Analyse / Conception 9](#_Toc198305261)

[2.1 SWOT 9](#_Toc198305262)

[*2.2* Concept 9](#_Toc198305263)

[2.3 Architecture du Système 12](#_Toc198305264)

[2.4 Stratégie de test 12](#_Toc198305265)

[2.5 Risques techniques 13](#_Toc198305266)

[2.6 Matériel et logiciels à disposition 13](#_Toc198305267)

[2.6.1 ESP32-WROOM 14](#_Toc198305268)

[2.6.2 Ecran OLED 16](#_Toc198305269)

[2.6.3 Module de capteur d’humidité du sol 16](#_Toc198305270)

[2.6.4 SPIFFS 19](#_Toc198305271)

[2.6.5 JSON 19](#_Toc198305272)

[2.6.6 CRUD 19](#_Toc198305273)

[2.6.7 Arduino IDE 19](#_Toc198305274)

[2.6.8 GIT 19](#_Toc198305275)

[2.7 Maquette 19](#_Toc198305276)

[2.7.1 Interface web 19](#_Toc198305277)

[3 Réalisation 20](#_Toc198305278)

[3.1 Dossier de réalisation 20](#_Toc198305279)

[3.1.1 Liaison du matériel 22](#_Toc198305280)

[3.1.2 Configuration du système 22](#_Toc198305281)

[3.1.3 Chemin du code 22](#_Toc198305282)

[3.1.4 Installation et configuration Arduino IDE 23](#_Toc198305283)

[3.1.5 Mesure de l’humidité 25](#_Toc198305284)

[3.1.6 Connexion sur le Wifi IOT 26](#_Toc198305285)

[3.1.7 Mise en œuvre du serveur HTTP 26](#_Toc198305286)

[3.1.8 Arrosage automatique 26](#_Toc198305287)

[3.1.9 Arrosage manuel 27](#_Toc198305288)

[3.1.10 Design Interface web 27](#_Toc198305289)

[3.1.11 Stockage des données 27](#_Toc198305290)

[3.2 Description des tests effectués 27](#_Toc198305291)

[3.3 Problèmes rencontrés 28](#_Toc198305292)

[*3.4* Erreurs restantes 28](#_Toc198305293)

[*3.5* Liste des documents fournis 28](#_Toc198305294)

[4 Conclusions 28](#_Toc198305295)

[4.1 Bilan des fonctionnalités 30](#_Toc198305296)

[4.2 Objectifs atteints 30](#_Toc198305297)

[4.3 Difficultés rencontres 30](#_Toc198305298)

[4.4 Bilan personnelle 30](#_Toc198305299)

[4.5 Résultat de stratégie de test 30](#_Toc198305300)

[4.6 Suites du projet 30](#_Toc198305301)

[5 Annexes 30](#_Toc198305302)

[*5.1* Résumé du rapport du TPI / version succincte de la documentation 30](#_Toc198305303)

[5.2 Sources – Bibliographie 30](#_Toc198305304)

[5.2.1 Installations 30](#_Toc198305305)

[5.2.2 Tutoriels 30](#_Toc198305306)

[5.2.3 Librairies 31](#_Toc198305307)

[5.2.4 Achats 31](#_Toc198305308)

[5.3 Lexique 31](#_Toc198305309)

[5.4 Utilisation de l’intelligence artificielle 31](#_Toc198305310)

[5.5 Journal de travail 31](#_Toc198305311)

[5.6 Rapport 31](#_Toc198305312)

[*5.7* Manuel d'Installation 31](#_Toc198305313)

[*5.8* Manuel d'Utilisation 31](#_Toc198305314)

[*5.9* Archives du projet 32](#_Toc198305315)

**Tableau des légendes**

# Analyse préliminaire

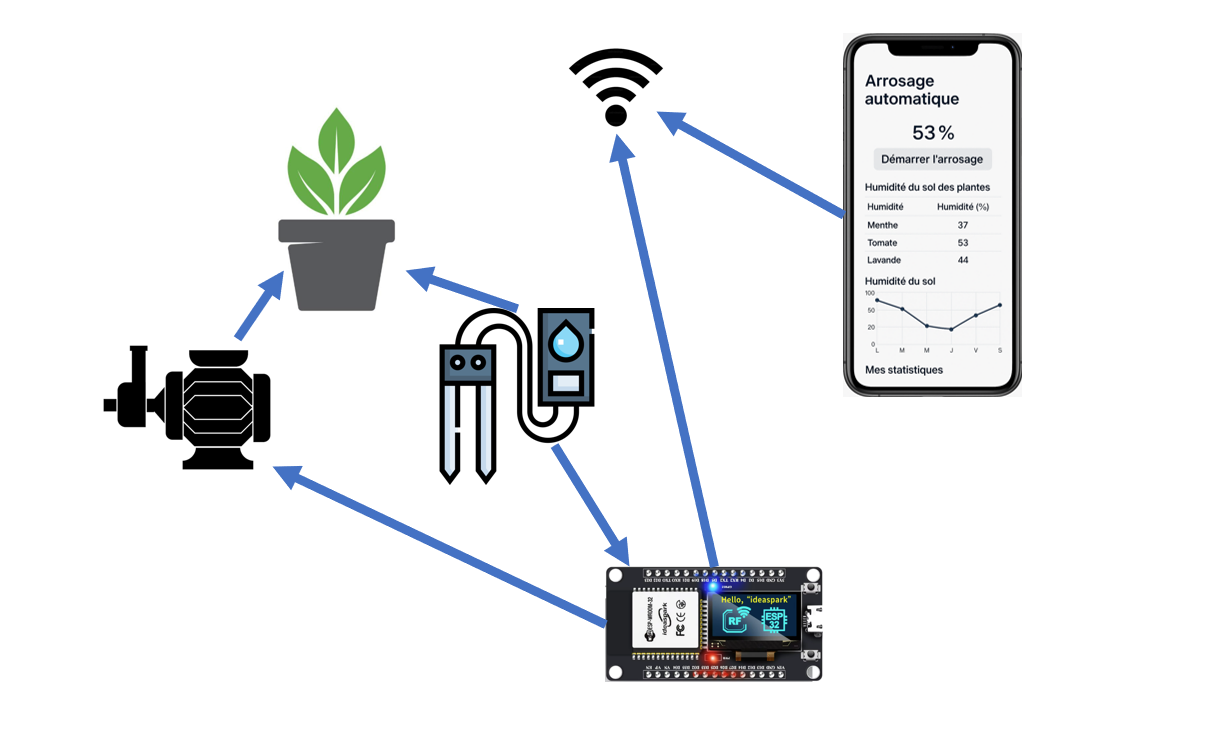
La recherche préliminaire établit les bases sur lesquelles le projet est fondé et ses objectifs principaux.

L'objectif de ce projet est de créer un système d’arrosage automatique ou manuellement géré via une interface Web.

## Introduction

Développer un système d’arrosage automatique basé sur un ESP32, un capteur d’humidité du sol et un relais, avec une plateforme de gestion permettant à un utilisateur non technique de configurer les paramètres d’arrosage via une interface web.

Ce projet a pour objectif la prise en main d’un ESP32-WROOM équipé d’un écran OLED intégré. L’ESP32 se connectera au réseau Wi-Fi IoT afin d’héberger un serveur web. Un capteur d’humidité est installé dans une plante pour mesurer son taux d’humidité. Les données relevées seront envoyées à l’ESP32, qui pourra alors déclencher automatiquement une pompe pour arroser la plante si nécessaire. Il sera également possible de lancer l’arrosage manuellement via l’interface web, qui servira de système de contrôle. Enfin, l’historique des arrosages sera enregistré dans la mémoire de l’ESP32 pour assurer un suivi complet.

Ici, on peut voir un graphique simple montrant le résultat que je dois obtenir à la fin de ce projet.**

## Objectifs

La grille d’évaluation définit les critères généraux selon lesquels le travail du candidat sera évalué (documentation, journal de travail, respect des normes, qualité, …).

En plus de cela, le travail sera évalué sur les 14 points spécifiques suivants

1. Connecter automatiquement l’ESP32 au réseau Wi-Fi local (réseau IOT dédié).
2. Développer une interface web locale pour la consultation et la configuration du système d’arrosage.
3. Lire la valeur d’humidité du sol depuis un capteur analogique, avec affichage en valeur brute ou en pourcentage.
4. Implémenter un contrôle automatique de l’arrosage basé sur un seuil d’humidité configurable.
5. Déclencher l’arrosage si la valeur mesurée est inférieure au seuil défini par l’utilisateur.
6. Permettre l’activation manuelle de l’arrosage via un bouton dans l’interface web.
7. Permettre le réglage de la durée d’arrosage (par exemple de 5 à 60 secondes).
8. Sauvegarder les paramètres de configuration dans une mémoire persistante (EEPROM émulée ou fichier JSON via LittleFS/SPIFFS).
9. Enregistrer l’historique des arrosages avec les dates, durées et seuils utilisés.
10. Mettre en place un serveur HTTP sur l’ESP32 répondant aux requêtes suivantes :
11. GET /data : renvoie les données actuelles du système (état, humidité, paramètres).
12. Implémenter la commande POST /manual pour activer manuellement l’arrosage via l’interface web.
13. Implémenter la commande POST /add pour ajouter une plante avec ses paramètres (nom, type, seuil).
14. Héberger les fichiers HTML et CSS de l’interface web dans le système de fichiers embarqué (SPIFFS ou LittleFS).

## Planification initiale

La planification initiale a pour but d'estimer le temps à consacrer à chaque phase du projet afin d’organiser efficacement le travail. Elle permet également, une fois le projet terminé, de comparer les estimations avec le temps réellement utilisé et d’analyser les écarts. Cette réflexion contribue à améliorer la gestion du temps et la planification pour de futurs projets.

Le temps prévu pour la réalisation de ce projet est de 89 heures et 55 minutes. Le projet débute le 2 mai 2025 et se termine le 2 juin 2025. Les tâches prévues dans cette planification sont basées sur le cahier des charges fourni par le chef de projet.

Voici les tâches génériques qui englobent les étapes fondamentales du projet, notamment les quatre plus importantes :

**Analyse** : cette tâche repose sur une phase de recherche, de réflexion, de vision du produit, d’imagination, de premiers tests et de compréhension du fonctionnement global.

**Implémentation** : cette tâche consiste à développer les fonctionnalités demandées dans le cahier des charges.

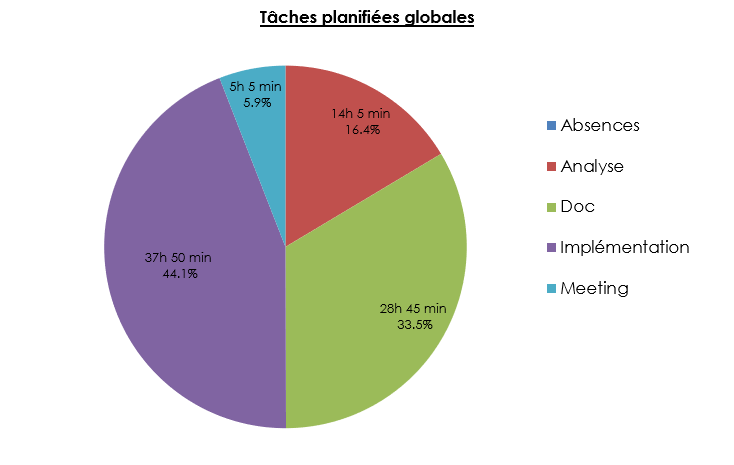
**Doc** : cette tâche comprend la rédaction du rapport, du journal de travail et des annexes.

**Meetings** : cette tâche concerne les échanges avec les experts et le chef de projet pour faire le point sur l’avancement et valider les choix techniques.

|  |
| --- |
| Absences - Imprévus |
| Analyse ‑ Cahier des charges  Analyse - Mise en place de la méthode de la gestion du projet  Analyse - Planification initiale  Analyse - Recherche  Analyse ‑ Composants  Analyse ‑ Tests |
| Implémentation - Commentaires code  Implémentation - Connexion réseau IOT  Implémentation - ESP32  Implémentation - GIT  Implémentation - Interface web  Implémentation - Mise en place des branches ESP32  Implémentation - Stockage |
| Doc - Journal de travail  Doc - Rapport |
| Meeting - Chef du projet  Meeting - Expert 1  Meeting - Expert 2 |

La différence entre le pourcentage de temps de planification que j’avais prévu et celui indiqué dans le CDC s’explique par mon estimation personnelle ; en raison de mes difficultés en français, je compte donc consacrer davantage de temps à la rédaction du rapport, comme l’illustre l’image ci‑dessus.





## Méthodologie du projet

Pour le déroulement de ce projet sur le ESP32, j’ai choisi d’utiliser la méthode de 6 pas : Informer, Planifier, Décider, Réaliser, Contrôler, Évaluer.

La principale raison de ce choix pour le projet est que je suis seul à le réaliser et aussi une meilleure réflexion sur chaque étape de mon point de vue.

La méthode des 6 étapes me permettra de planifier étape par étape mon projet de construction. Elle m’aidera à rester concentré et à ne rien oublier. Ce travail de fond m'aidera à éviter certains soucis et erreurs dans mon projet de construction. Dans mon cas, me permet aussi de décomposer mes étapes hardware avec l’ESP32 et les composants pour permettre l’arrosage des plantes, développement web, gestion du réseau et stockage etc.

Les 6 étapes organisées dans mon projet :

* **Informer** : Je me suis documenté sur l’ensemble des éléments nécessaires au bon déroulement du projet, notamment sur le matériel (ESP32, capteurs d’humidité, relais), les contraintes du cahier des charges, les outils de développement (Arduino IDE, Git, SQLite, JSON) et les fonctionnalités attendues comme le CRUD et l’interface web statique. Cette phase m’a permis de bien cerner les objectifs et le contexte global du projet.
* **Planifier** : J’ai établi une planification structurée, en répartissant les tâches selon quatre axes : analyse, implémentation, documentation et finalisation. Cette organisation m’a permis de mieux gérer mon temps et de suivre l’avancement du projet de manière efficace.
* **Décider** : J’ai fait des choix techniques en fonction des besoins et des contraintes : utilisation d’un fichier JSON pour le stockage des données, implémentation d’un serveur HTTP pour gérer la communication entre l’ESP32 et l’interface, conception de l’interface en HTML/CSS, et adoption de la méthode des 6 étapes pour structurer l’ensemble du projet.
* **Réaliser** : J’ai développé le code progressivement, en finalisant chaque fonctionnalité indépendamment (lecture des capteurs, contrôle du relais, gestion des données, etc.) avant de les intégrer dans l’ensemble du système. Cela m’a permis de tester chaque partie de manière isolée avant de les réunir.
* **Contrôler** : J’ai effectué des tests manuels pour vérifier le bon fonctionnement de chaque fonctionnalité (lecture de l’humidité, déclenchement automatique ou manuel de l’arrosage, ajout de plantes, etc.). J’ai également testé tous les composants électroniques pour m’assurer de leur fiabilité.
* **Évaluer** : En fin de projet, j’ai comparé les résultats obtenus avec les objectifs fixés initialement. Grâce aux tests réalisés dès les premières étapes, j’ai pu identifier les points forts et les éventuelles améliorations à envisager pour une future version du système.

Pourquoi je n’ai pas choisi les autres méthodes ?

Méthode Agile : adaptée aux projets évolutifs avec des changements fréquents, ce qui n’est pas le cas ici car mon projet a un objectif bien défini aux départs.

Cycle en V : plus utilisé pour des projets complexes nécessitant de longues phases de validation et de tests stricts.

# Analyse / Conception

Ce chapitre prépare la réalisation du projet en identifiant ce qu’il faut faire, avec quels outils, et selon quelle méthode, afin de garantir une progression claire et cohérente. À partir des besoins, on détermine les composants à utiliser, les logiciels nécessaires, ainsi que les principes de fonctionnement à respecter.

L’analyse permet d’éviter les erreurs en anticipant les difficultés techniques et les imprévus. On y examine les risques, on envisage des solutions de contournement, et on s’assure que chaque choix technique repose sur une logique fondée.

## SWOT

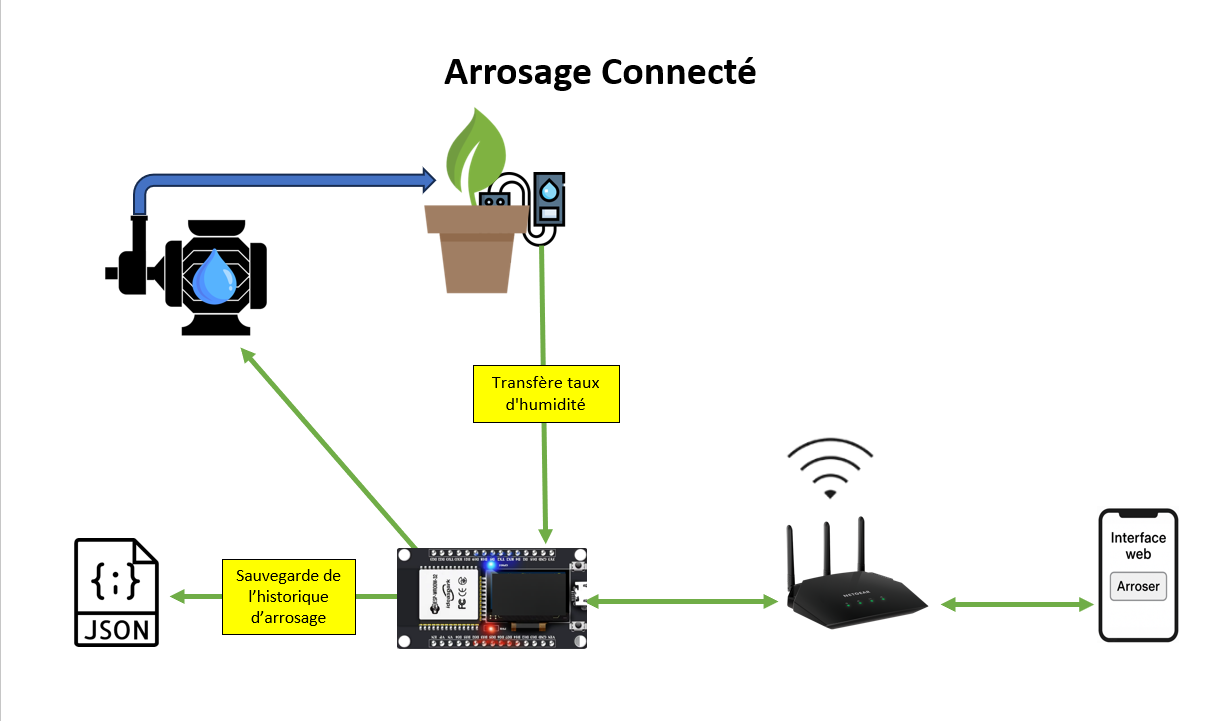
Avant de commencer le projet, j’ai pris une décision simple : analyser ma stratégie.  
Pourquoi ?

Pour mieux comprendre mes points forts, mais aussi mes faiblesses.  
Le but : voir ce qui pourrait me freiner, et ce qui pourrait m’aider.  
Cette démarche me permet aussi d’anticiper les menaces possibles,  
et, surtout, de repérer les opportunités que ce projet peut offrir.

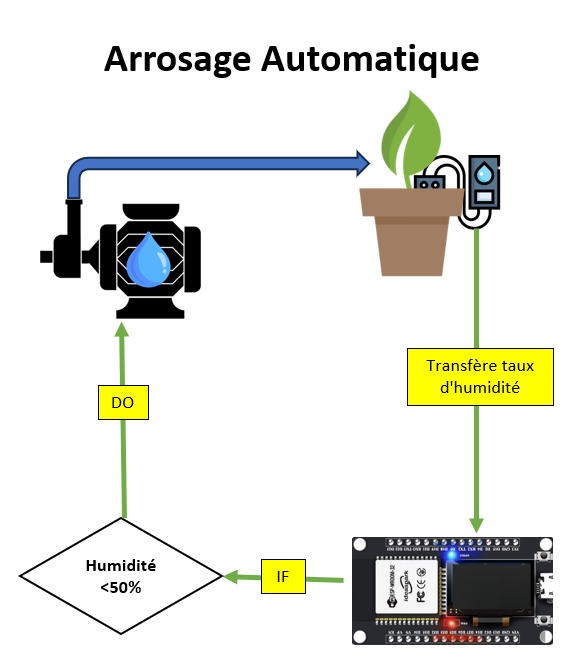
|  |  |
| --- | --- |
| **Forces** | **Faiblesses** |
| * Connaissance dans la plateforme Arduino IDE. * Connaissance en C#. | * Manque d'expérience avec SPIFFS * Perte de temps significative lors de la rédaction du cahier des charges en raison de difficultés avec la langue française. * Connaissances en électronique et systèmes embarqués. |
| **Opportunités** | **Menaces** |
| * Projet réutilisable dans un cadre personnel ou professionnel. * Projet valorisable pour un portfolio. * Apprentissage concret du développement embarqué. | * Incertitude sur la compatibilité et les limites de la mémoire SPIFFS. * Difficulté potentielle à gérer correctement les fichiers JSON. |

## Concept

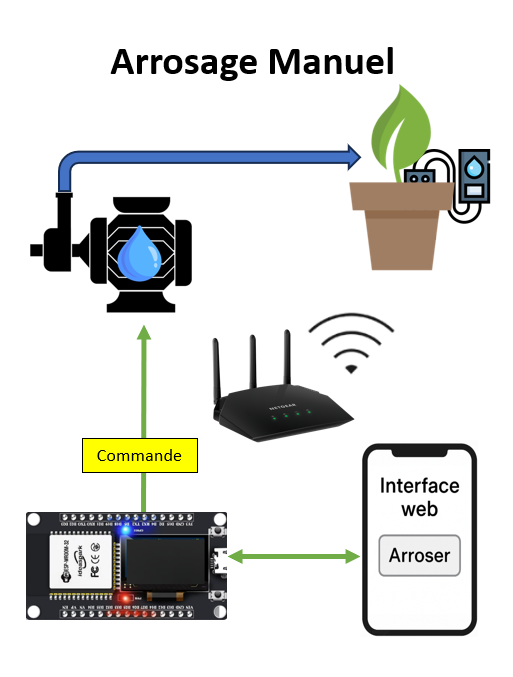
Ce projet vise à automatiser l’arrosage d’une plante sans intervention manuelle, offrant une solution pratique et connectée pour la maison. L’utilisateur peut surveiller et déclencher l’arrosage à distance, directement depuis une interface web accessible via son smartphone, sans avoir à se déplacer. Le cœur du système repose sur un module ESP32, qui reçoit les données d’un capteur d’humidité placé dans le sol. Lorsque le taux d’humidité est trop bas, l’ESP32 active une pompe à eau qui alimente la plante. Ce même microcontrôleur héberge une interface web accessible en Wi-Fi, permettant à l’utilisateur de contrôler manuellement l’arrosage. Chaque action, qu’elle soit automatique ou manuelle, est enregistrée dans un fichier JSON local afin de conserver un historique. L’ensemble du dispositif constitue une infrastructure simple, efficace et parfaitement adaptée à un projet de domotique fait maison.

**

Dans ce chemin ci-dessus, le capteur inséré dans la plante mesure l’humidité du sol et envoie la donnée au microcontrôleur ESP32. Ce dernier vérifie une condition liée au niveau d’humidité. Si cette condition est remplie, l’ESP32 envoie une commande pour activer automatiquement la pompe, qui arrose alors la plante. Ce processus se déroule sans intervention humaine, assurant un arrosage autonome selon les besoins détectés par le capteur.

**

Dans ce chemin ci-dessus, l’utilisateur doit d’abord se connecter au réseau Wi-Fi local généré par l’ESP32. Une fois connecté, il accède à l’interface web hébergée directement sur le microcontrôleur. Depuis cette interface, il peut déclencher l’arrosage en appuyant sur un bouton. Cette commande est alors transmise à l’ESP32, qui active la pompe pour arroser la plante. Ce fonctionnement permet à l’utilisateur de contrôler manuellement le système à distance, sans intervention physique sur le matériel.

**

## Architecture du Système

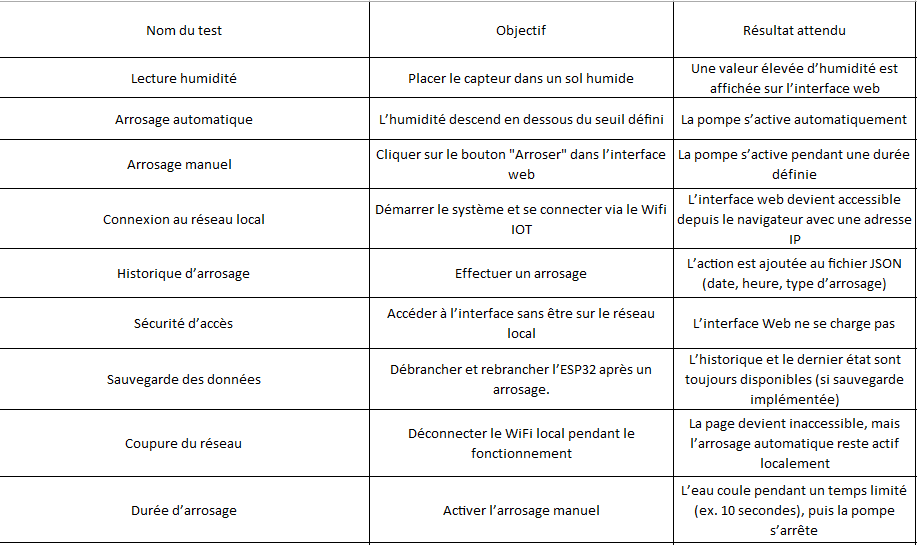
Chemin électrique

## Stratégie de test

Ce chapitre sur la stratégie de test sert à définir les tests à envisager avant même de commencer le projet. Le but est d’anticiper les vérifications nécessaires à chaque étape du développement, pour s’assurer que toutes les fonctionnalités importantes soient bien prises en compte.

À la fin du rapport, on trouvera les résultats de ces tests ainsi qu'une remarque d’évaluation personnelle, qui permet de réfléchir à ce qui pourrait être amélioré dans de futurs projets.

Le tableau présenté regroupe donc des tests prévisionnels, pensés avant le début concret du projet. Cela signifie que la précision est parfois approximative, car le système n’est pas encore construit. Malgré cela, ce tableau est utile pour se baser sur les fonctions de base que l'utilisateur devra utiliser ou rencontrer.



## Risques techniques

Le chapitre sur les risques techniques présente les problèmes que je pourrais rencontrer dans le projet, afin de les anticiper et de mieux les planifier à l’avance.

Les risques identifiés sont les suivants :

* Dysfonctionnement du matériel électronique : Bien que je dispose de composants de rechange, un appareil défectueux pourrait entraîner une perte de temps lors des tests et diagnostics.
* Mauvaise compréhension du cahier des charges : Une interprétation incorrecte des objectifs ou des contraintes du projet pourrait conduire à un développement non conforme.
* Nombre insuffisant de GPIO disponibles sur l’ESP32 : Cela limiterait la possibilité d’ajouter plusieurs plantes à contrôler individuellement.
* Mémoire limitée de l’ESP32 : Elle peut poser un problème pour le stockage des données, notamment les fichiers JSON utilisés pour les logs.
* Erreur d’arrosage en cas de déconnexion Wi-Fi : Si l’ESP32 ne parvient pas à se connecter au réseau, certaines fonctionnalités pourraient devenir inaccessibles.
* Impossibilité de charger les pages web ou de sauvegarder les données JSON :Cela pourrait compromettre l’interface utilisateur et la traçabilité des actions.
* Problèmes de communication entre le serveur web et l’ESP32 : Commandes non reçues ou affichage de valeurs erronées.

## Matériel et logiciels à disposition

* 2 microcontrollers ESP32(ex. ESP32-WROOM-32) avec Écran OLED 0,96 128×64 px (driver SSD1306, bus I²C).
* 2 capteurs d'humidité du sol FC-28.
* 2 relais 5V SRD-05VDC-SL-C pour contrôler une pompe ou électrovanne.
* 1 ordinateur type PC Windows.
* Câbles Dupont.
* 2 mini planche 45,72x35,56mm.
* Câble USB-A à micro USB.
* Environnement de développement Arduino IDE pour programmer l'ESP32 en C.
* SQLite (ou fichier JSON pour simplicité) pour la plateforme de gestion.
* Logiciel de contrôle de version Git Desktop.
* Microsoft Office pour la documentation.
* Connexion Wifi local (IOT).
* Compte GitHub.

### ESP32-WROOM

Le module ESP32-WROOM utilisé dans ce projet constitue le cœur opérationnel de mon système d'arrosage automatique connecté. C'est un microcontrôleur puissant et polyvalent qui intègre à la fois la gestion Wi-Fi et Bluetooth, idéal pour des projets IOT.

#### Caractéristiques Techniques

* **Microcontrôleur** : ESP32 dual-core Xtensa LX6
* Wifi : 2.4GHz
* **Mémoire** : 520 Ko de SRAM interne, 4 Mo de Flash.
* **Connectivité :** Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR/EDR et BLE (Bluetooth Low Energy).
* **Interfaces intégrées : GPIO, I2C, SPI, UART, ADC, DAC, PWM, I2S.**

#### Utilisation dans le projet

L'ESP32-WROOM remplit plusieurs fonctions essentielles dans ce projet :

* **Acquisition de données** : Grâce à ses entrées analogiques (GPIO analogiques), l'ESP32 mesure le taux d'humidité du sol via un capteur connecté à la broche "VP" (GPIO36, input only), permettant ainsi de surveiller en temps réel l'état de la plante.
* **Contrôle de la pompe** : En fonction des données reçues du capteur d'humidité, l'ESP32 actionne une pompe à eau via une broche GPIO numérique configurée en sortie, assurant un arrosage précis selon les besoins réels de la plante.
* **Serveur Web embarqué** : Le module ESP32 est configuré comme serveur Web local, rendant possible le contrôle manuel de l'arrosage depuis une interface web simple et intuitive accessible sur n'importe quel appareil connecté au même réseau local.
* **Connectivité IoT** : L'ESP32-WROOM se connecte via Wi-Fi au réseau domestique afin de faciliter l’interaction utilisateur/système sans nécessité de connexion physique ou filaire supplémentaire.

#### Justification du choix

Le choix de l’ESP32-WROOM est motivé par :

* Ma connaissance approfondie sur l'utilisation et la programmation des modules ESP, ainsi que ma maîtrise de l'application Arduino IDE.
* Mon expérience préalable avec ce type de dispositif à travers plusieurs projets similaires réalisés avec succès.
* Sa polyvalence en gestion simultanée des fonctions Wi-Fi et des entrées/sorties GPIO.

#### Différences majeures

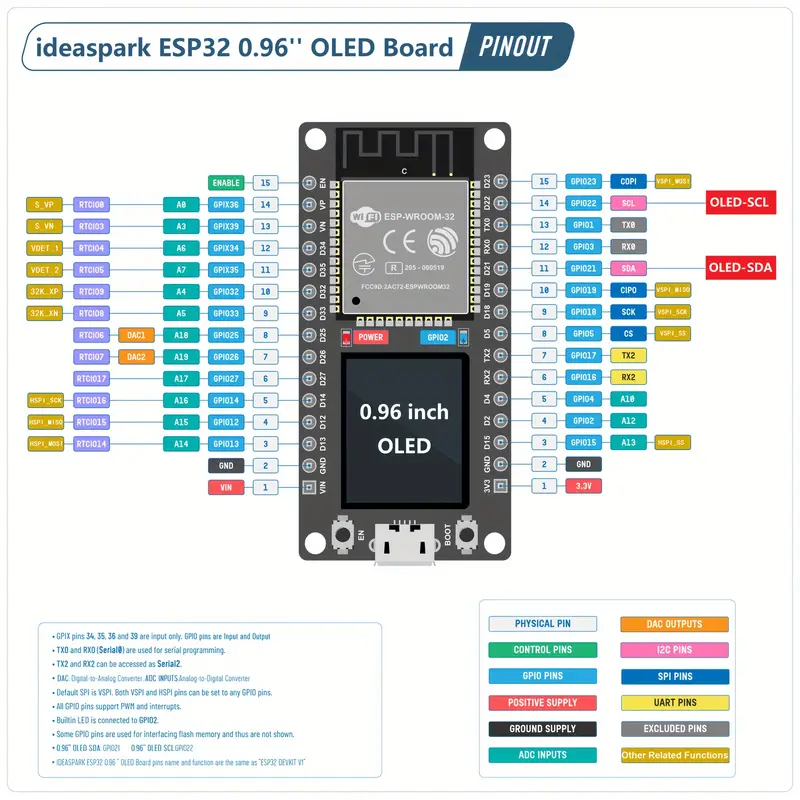
**Ordinateur vs ESP32** : L’ordinateur est puissant et polyvalent, l’ESP32 est léger et économe, idéal pour les tâches simples des systèmes embarqués.

**Raspberry Pi vs ESP32** : Le Raspberry Pi est un mini-ordinateur qui peut exécuter un système d'exploitation complet, idéal pour des applications nécessitant une interface utilisateur complexe ou une grande capacité de stockage. De son côté, l'ESP32 est plus adapté aux applications légères, temps réel et autonomes où la consommation d'énergie et le coût sont critiques.

#### GPIO

Les GPIO (General Purpose Input/Output) sont des broches polyvalentes sur l'ESP32 permettant de connecter divers capteurs et actionneurs. Elles peuvent être configurées soit en entrée pour lire les données provenant des capteurs, soit en sortie pour activer ou désactiver

Les couleurs utilisées sur les broches GPIO indiquent leurs fonctions spécifiques :

****

* **Broches bleu clair – GPIO (Entrée/Sortie numérique)**

Broches polyvalentes configurables en entrée ou sortie. Utiles pour capteurs, LEDs, relais. Ex. : GPIO2, GPIO4, GPIO12 à 27.

* **Broches vertes – ADC (Convertisseur Analogique-Numérique)**

Mesurent des tensions variables (capteurs). Convertissent le signal analogique en numérique. Ex. : GPIO32 à GPIO39**.**

* **Broches orange – DAC (Convertisseur Numérique-Analogique)**

Génèrent une tension analogique à partir de données numériques. Ex. : GPIO25, GPIO26**.**

* **Broches jaunes – SPI (Serial Peripheral Interface)**

Protocole rapide pour écrans, cartes SD, etc. Utilise plusieurs fils. Ex. : GPIO18, 19, 23, 5**.**

* **Broches violettes – Contrôle système**

Broches internes comme EN (activation). Ne pas utiliser pour des périphériques**.**

* **Broches noires – Masse (GND)**

Référence électrique commune pour tous les composants**.**

* **Broches rouges – Alimentation (Power)**

3.3V pour alimenter les modules ; VIN accepte une source externe (souvent 5V)**.**

* **Broches beige – UART (communication série)**

Communication avec d’autres modules (Bluetooth, PC). Ex. : GPIO1, 3, 16, 17**.**

* **Broches jaune clair – I2C (Inter-Integrated Circuit)**

Protocole simple, deux fils : SDA (GPIO21), SCL (GPIO22). Pour écran OLED, capteurs**.**

* **Broches gris foncé – à éviter**

Broches réservées à la mémoire flash. Ne pas utiliser : GPIO6 à GPIO11**.**

#### Ecran OLED

Sur la carte ESP32-WROOM, un petit écran OLED est directement intégré. Sa taille : 0.96 pouce, soit environ 2,44 centimètres. L’écran affiche en deux zones : en haut, le tiers supérieur, souvent en jaune ; en bas, les deux tiers restants, généralement en bleu.

L’écran offre une résolution de 128x64 pixels, suffisante pour afficher du texte, des icônes, ou des informations de capteurs avec clarté. Il fonctionne avec le protocole I2C, simple et efficace.

Pour le programmer avec Arduino IDE, deux bibliothèques sont nécessaires :

* Adafruit SSD1306, pour gérer le contrôleur de l’écran ;
* Adafruit GFX, pour le rendu graphique.

La connexion matérielle est directe :

* SDA est relié à GPIO21,
* SCL est relié à GPIO22.

Ces deux broches assurent toute la communication.

L'écran, une fois initialisé, peut afficher des messages, des valeurs numériques ou des alertes visuelles.

### Module de capteur d’humidité du sol

Le capteur d’humidité FC-28 permet de mesurer la teneur en eau du sol. Il fonctionne en détectant la conductivité électrique entre deux sondes métalliques : plus le sol est humide, plus la conductivité est élevée.

#### Fonctionnement

* Lorsque le sol est sec, la résistance est élevée → faible signal.
* Lorsque le sol est humide, la résistance est basse → signal élevé.
* Le module fournit :
* Une sortie analogique (AO) : valeur de l’humidité.
* Une sortie numérique (DO) : niveau haut ou bas selon un seuil réglable.

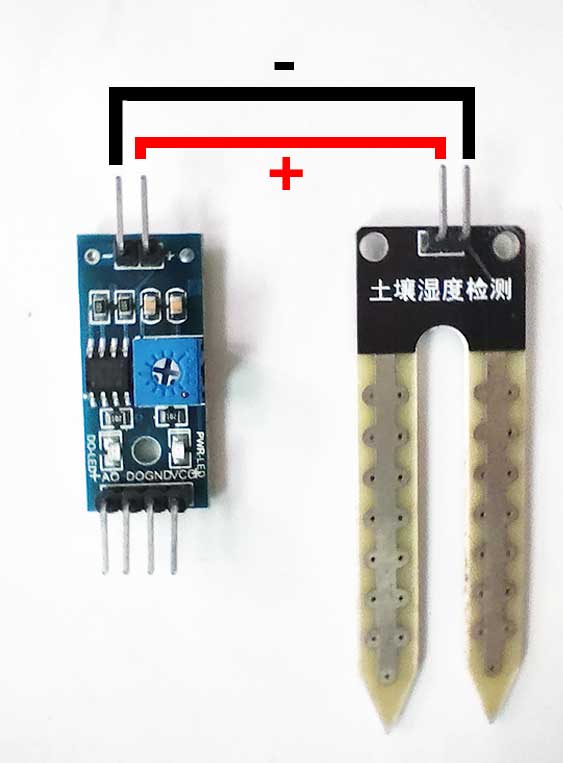
#### Avantages

* Facile à utiliser avec Arduino ou ESP32
* Double sortie : analogique (précise) et numérique (seuil réglable)
* Coût faible

#### Inconvénients

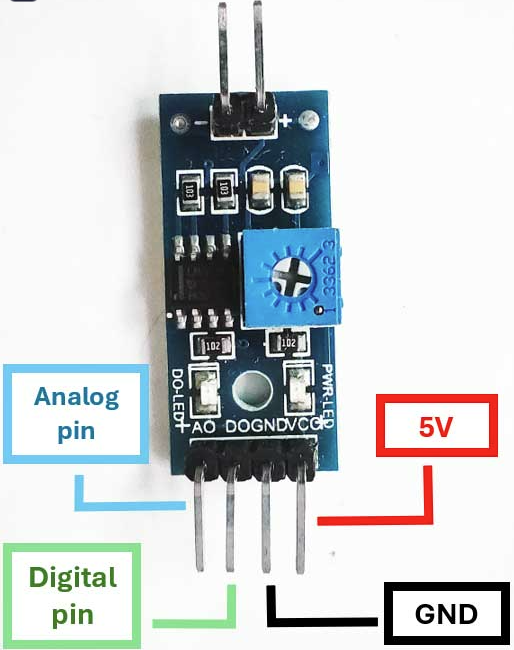
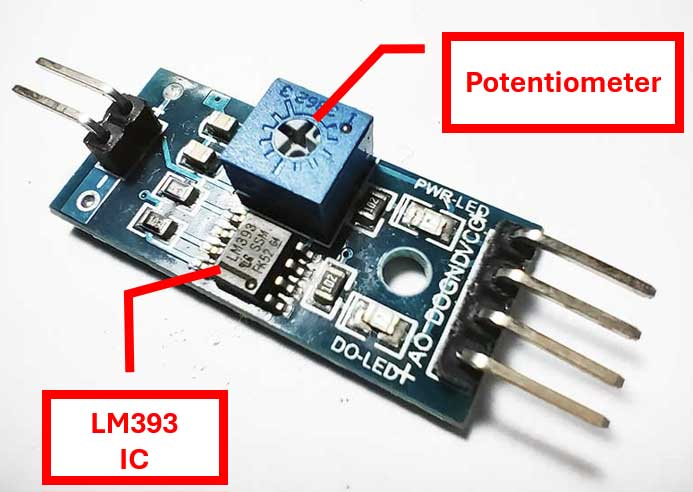
* Pas toujours précis

#### Module

Le capteur d’humidité du sol mesure la quantité d’eau en détectant la résistance entre deux sondes. Plus le sol est humide, plus la résistance est faible. Le module utilise un LM393, un comparateur qui convertit ce signal en sortie numérique selon un seuil défini.

Sur l’image ci-dessus, on observe que ce projet utilise uniquement la sortie analogique, la broche de tension (VCC) et la masse (GND). La sortie numérique n’est pas nécessaire car l’objectif est d’obtenir un pourcentage précis d’humidité, et non une simple valeur binaire (0 ou 1). Le capteur permet également d’ajuster la sensibilité via le potentiomètre pour affiner la précision des mesures.

Le **potentiomètre**, visible en bleu sur l’image, permet de **régler la sensibilité du capteur**.  
En le tournant, on ajuste le **seuil de détection** du module : plus il est élevé, plus le capteur réagira à un changement faible de signal.  
Cela permet d’adapter la réponse du capteur selon les besoins du projet.



### SPIFFS

SPIFFS, un système de fichiers léger intégré à la mémoire flash de l’ESP32, permet de lire, écrire et stocker localement des fichiers comme des pages HTML, CSS, JavaScript ou des logs d’humidité avec date et heure, ce qui rend possible dans ton projet l’affichage d’une interface web et la sauvegarde automatique des données sans carte SD ni connexion Wi-Fi, tout en garantissant la persistance des fichiers même après un redémarrage.

#### SPIFFS dans mon projet

* Stocker des fichiers HTML, CSS ou JavaScript pour créer une interface web embarquée sur l'ESP32.
* Sauvegarder des données de capteurs ou des journaux sans avoir besoin d'une carte microSD.

### JSON

Un fichier JSON est un format léger de stockage de données, permet de sauvegarder l’historique des arrosages, en enregistrant le nom de la plante, son type, la date de l’arrosage, le taux d’humidité à ce moment-là ainsi que le temps d’arrosage ; un autre fichier JSON est utilisé pour la création des plantes, et ce format est particulièrement adapté à l’ESP32 car il est très léger, ce qui permet d’optimiser l’utilisation de la mémoire limitée du microcontrôleur.

Le choix d’un fichier JSON s’impose car il offre un format structuré, léger et parfaitement adapté à la mémoire de l’ESP32.

### CRUD

Le choix du modèle CRUD (Create, Read, Update, Delete) s’impose car il permet de gérer efficacement les données des plantes et de l’historique d’arrosage en offrant une méthode simple et structurée pour créer, lire, modifier ou supprimer les informations directement depuis l’interface web, ce qui rend l’interaction avec les fichiers JSON intuitive et complète.

* Create : je vais l’utiliser pour créer de nouvelles plantes.
* Read : je vais l’utiliser pour lire les plantes enregistrées dans le fichier JSON.
* Update : je vais l’utiliser pour mettre à jour les informations d’une plante, comme son nom ou son type.
* Delete : je vais l’utiliser pour permettre la suppression d’une plante.

### Arduino IDE

J’ai choisi d’utiliser Arduino IDE comme environnement de développement car c’est celui avec lequel j’ai commencé à apprendre la programmation sur microcontrôleurs, ce qui me permet de gagner du temps en évitant de devoir me former sur un nouveau logiciel.

#### Avantages :

* Je maîtrise déjà ce logiciel, ce qui m’évite une phase d’apprentissage supplémentaire.
* Son interface est simple, épurée et bien adaptée aux débutants.
* Il est directement compatible avec l’ESP32 grâce aux bibliothèques disponibles.

#### Inconvénients

* La compilation est lente, ce qui rallonge le temps de test et de débogage.
* Les erreurs de compilation ne sont souvent détectées qu’après le flash, ce qui fait perdre du temps.

#### Alternatives possibles

* Visual Studio Code avec PlatformIO, compilation plus rapide et une meilleure gestion des fichiers.

### GIT

Git est un système de gestion de versions qui permet de suivre les modifications apportées à des fichiers dans un projet. Il est principalement utilisé pour faciliter la collaboration entre développeurs, sécuriser les données et restaurer les versions antérieures au besoin.

Dans le cadre de ce projet, Git a été employé pour organiser les documents et enregistrer quotidiennement toutes les données pertinentes (journal de travail, rapports, code source, bibliothèques, etc.), tout en assurant une structure précise des répertoires pour éviter toute perte d'informations.

Pour la gestion du dépôt Git, l'outil GitHub Desktop a été privilégié pour une facilité d'envoi des fichiers vers le répertoire GitHub.

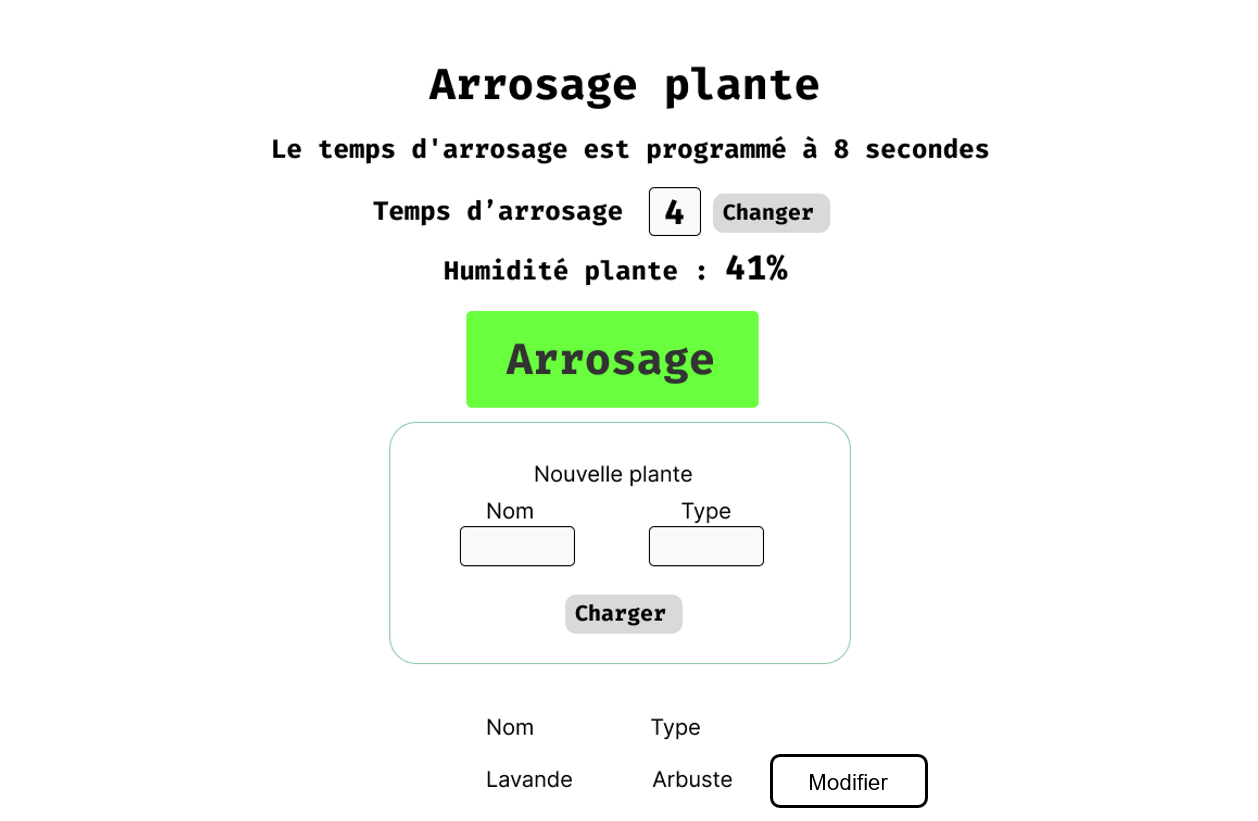
## Maquette Interface web

Pour concevoir la maquette, j’ai retenu Figma : je recherchais un outil à la fois intuitif, facile à piloter et rapide à mettre en œuvre, d’autant que le projet ne comprend qu’une seule interface.

L’utilisateur peut se connecter directement, depuis un navigateur, au réseau Wi‑Fi IOT via son adresse IP. Dans cette interface ci‑dessus, on peut voir :

* Un titre « Arrosage plante » qui indique clairement la fonction du site.
* Un texte qui affiche la durée d’arrosage programmée.
* Un champ est prévu sur le site web pour régler la durée d’arrosage, exprimée en secondes. Cette durée peut être modifiée directement depuis l’interface web.
* La mesure en temps réel de l’humidité du sol, présentée en pourcentage (0–100 %) pour être plus parlante qu’un simple nombre brut.
* Un bouton « Arroser » qui déclenche l’arrosage pour la durée définie.
* Un formulaire permet d’enregistrer une nouvelle plante en saisissant son nom et son type. Deux champs de texte sont prévus à cet effet, et un bouton «Charger » valide l’envoi des informations
* Un champ permet également d’indiquer le nom et le type de la plante, afin d’identifier facilement les différentes plantes présentes.

Ainsi, l’interface offre à la fois une vue d’ensemble de l’état de la plante et un contrôle manuel de l’arrosage.

**

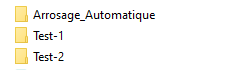
# Réalisation

Ce chapitre représente toute la partie « Pratique » de ce projet. Ce chapitre

Permet de comprendre le fonctionnement de chaque étape, les erreurs rencontrées et restantes, des descriptions des tests effectués, une solution apportée pour attendre le résultat souhaité, pour que l’ESP32-CAM ait un environnement fonctionnel et soit configuré comme demandé dans le cahier des charges.

## Dossier de réalisation

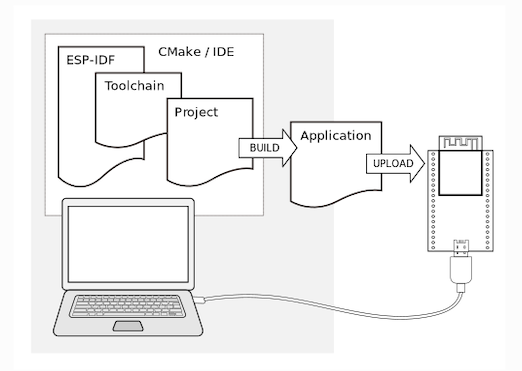
Pour le développement de ce projet, j’ai adopté une méthode de travail structurée basée sur l’utilisation de trois espaces de code, comme illustré dans l’image ci-dessous. Les fichiers nommés 'Test-1' et 'Test-2' sont utilisés comme environnements de test pour développer et valider différentes fonctionnalités du code. Une fois ces fonctionnalités testées avec succès, elles sont intégrées dans le dossier principal 'Arrosage\_Automatique', qui contient l’architecture complète et définitive du projet.



L’image ci-dessus illustre le processus de chargement du code sur l’ESP32. Ce processus se divise en plusieurs étapes :

1. Les pilotes, bibliothèques et outils nécessaires sont regroupés dans l’environnement de développement ESP-IDF (Espressif IoT Development Framework).
2. Les compilateurs et utilitaires de compilation, appelés Toolchain, permettent de convertir le code source en langage machine compréhensible par l’ESP32.
3. L’interface de développement, dans mon cas l’Arduino IDE, utilise CMake pour gérer et compiler les projets.
4. La partie "projets" correspond au code que j’ai écrit pour ce système d’arrosage.

Une fois le code prêt, il est compilé en fichier binaire. Ce fichier est ensuite envoyé à l’ESP32. Si aucune erreur n’est détectée, le microcontrôleur l’exécute et le système devient opérationnel.

**

*Code*

*Dans le code j’ai utilisé la norme under camel case*

*Décrire la réalisation "physique" de votre projet*

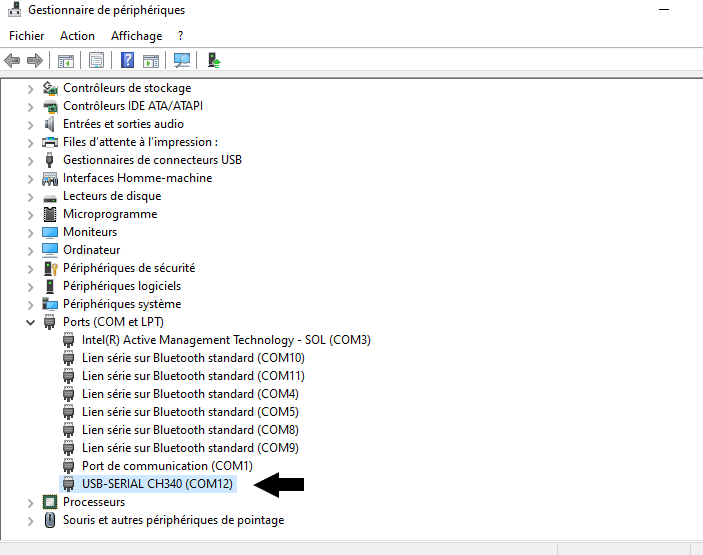
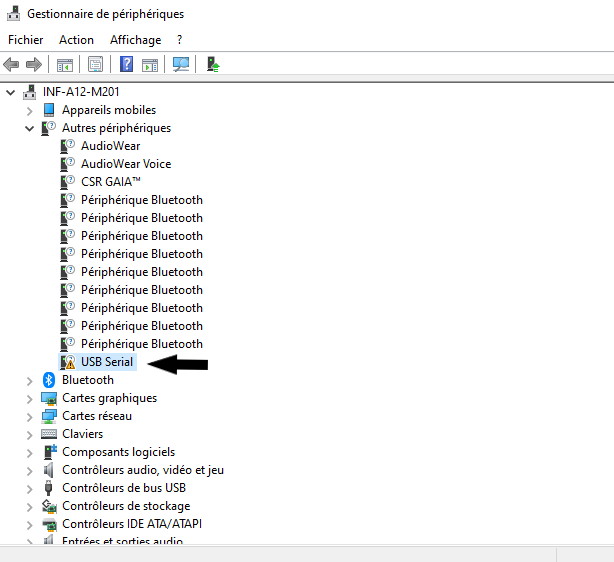
* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *programmation et scripts: librairies externes, dictionnaire des données, reconstruction du logiciel - cible à partir des sources.*

*NOTE : Evitez d’inclure les listings des sources, à moins que vous ne désiriez en expliquer une partie vous paraissant importante. Dans ce cas n’incluez que cette partie…*

### Liaison du matériel

### Configuration du système

J’ai dû télécharger le pilote CH340 (lien chapitre 5.2) sur Windows 10 afin que l’ordinateur puisse reconnaître l’ESP32. En revanche, sur Windows 11 (testé à domicile), le support est natif. Ci-dessous, une première image montre l’état de la connexion sans le pilote installé, et une seconde image illustre la détection correcte de l’ESP32-CAM une fois le pilote CH340 installé.



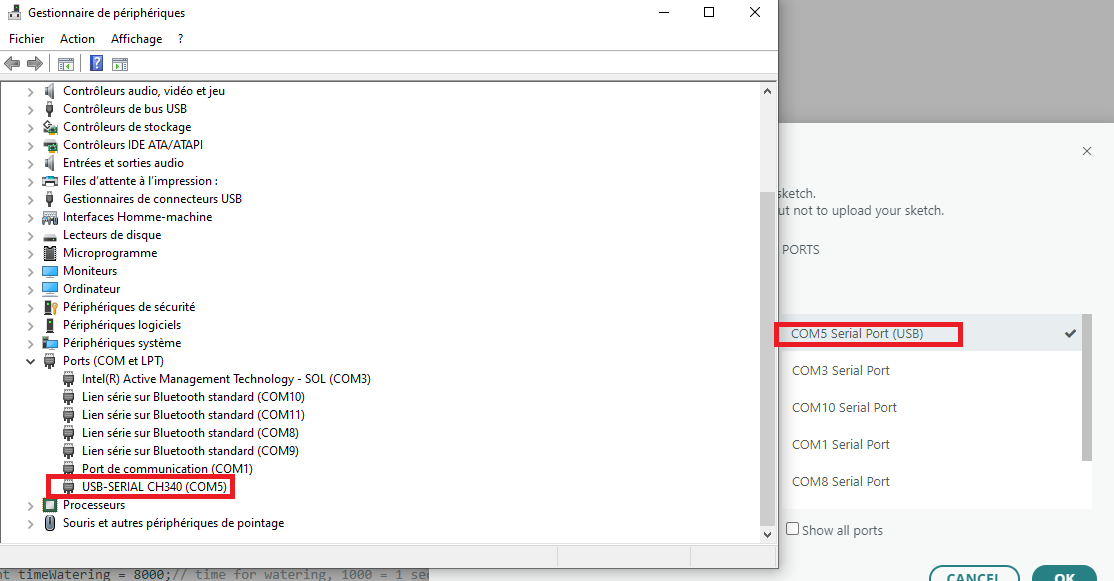
### Chemin du code

### Installation et configuration Arduino IDE

Afin de développer le projet, l’environnement de développement Arduino IDE version 2.3.4 a été installé. « Lien pour l’installation Chapitre 5.2»

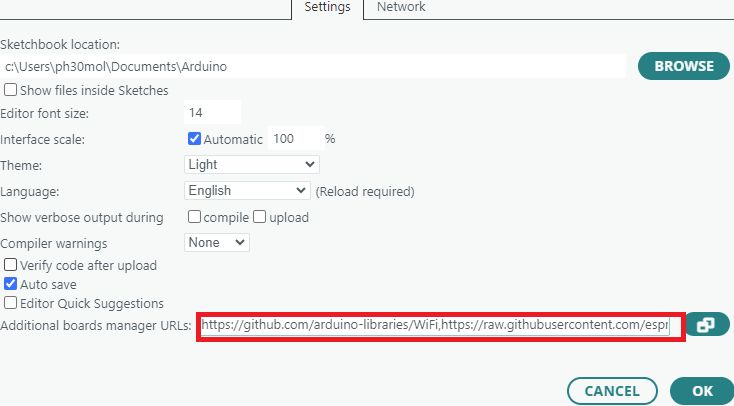
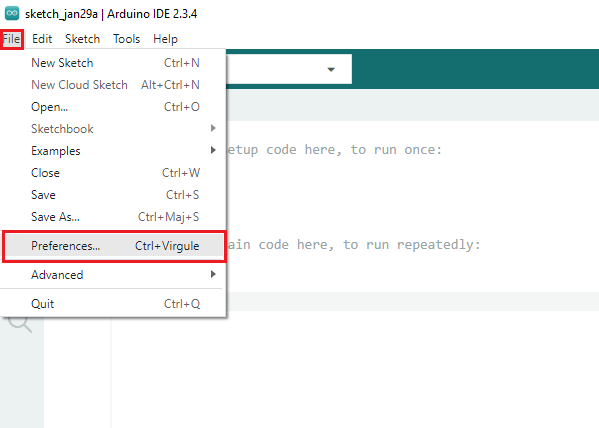
#### Configuration du port

L’image ci-dessous illustre le choix du port de communication (COM) à effectuer dans le gestionnaire de périphériques de Windows. Ce port est indispensable pour permettre la communication entre l’ordinateur et la carte ESP32 via l’IDE Arduino.



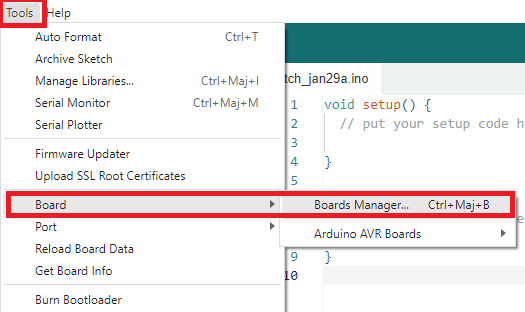
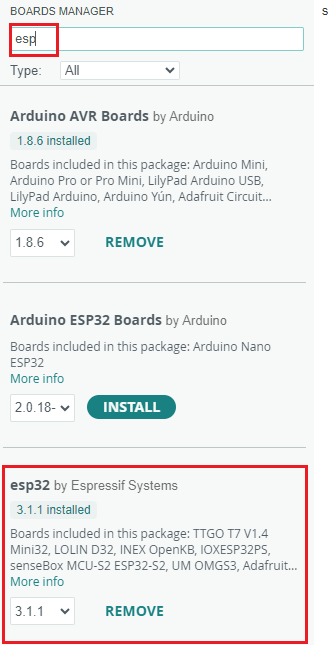
#### URL ESP32

Par la suite, il a été nécessaire d’**ajouter la carte ESP32** dans l’environnement Arduino. Pour cela, on accède aux **préférences** de l’IDE via le menu **File > Préférences**, comme le montre la deuxième image. L’URL suivante a été ajoutée dans le champ « Additional Board Manager URLs ». «Url chapitre 5.2 ».



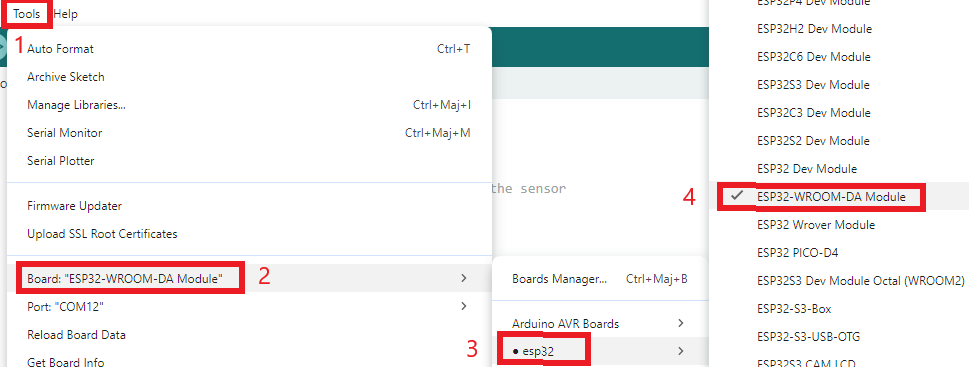
#### Téléchargement librairie ESP32

Comme le montre l’image ci-dessus, l’installation de la carte ESP32 s’effectue à partir du menu Tools > Board > Boards Manager. Dans la barre de recherche du gestionnaire de cartes, on tape "esp32", puis on installe le paquet nommé esp32 by Espressif Systems (version 3.1.1 dans ce cas).



#### Module de programmation

Une fois l’installation terminée, on retourne dans le même menu des cartes, on fait défiler les options jusqu’à la section ESP32, on sélectionne le modèle utilisé ici, ESP32-WROOM-DA Module, et dès que cette carte est cochée, le code peut être compilé et téléversé sans problème.



Ce sont les 4 manières les plus importantes de programmer et de travailler sur l’ESP32.

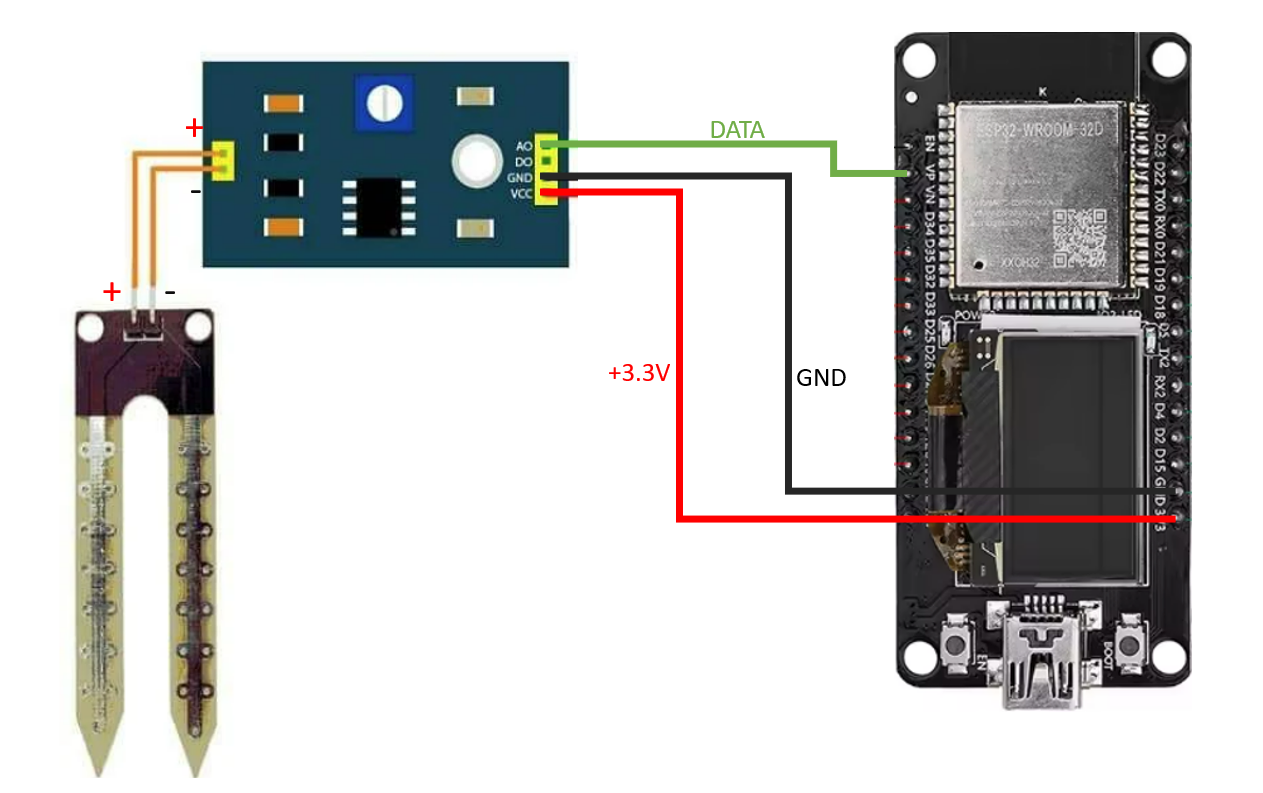
### Mesure de l’humidité

À cette étape, il faut mesurer l’humidité captée par la sonde. On utilise pour cela un capteur d’humidité relié à un convertisseur analogique‑numérique (ADC) qui transmet les données à l’ESP32 dans un format exploitable. Aucune bibliothèque n’est requise : un calcul simple (décrit ci‑dessous) convertit la valeur brute en pourcentage, où 0 % signifie un sol totalement sec et 100 % un sol saturé d’eau.

**Valeur capteur = Une valeur brute comprise entre 0 et 4095  
pourcent Humidité = (Valeur capteur × 100) / 4095**

Le GPIO VP a été choisi pour recevoir les données du capteur, car il permet une lecture analogique vers l’ESP32, sans retour de signal vers le capteur. Ce choix permet d’économiser d’autres broches GPIO pour les besoins restants du projet.

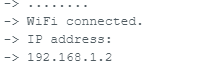
J’ai utilisé l’alimentation de 3,3 V sur l’ESP32 car, après plusieurs essais comparatifs, elle m’a donné des résultats plus stables et cohérents avec le capteur.

L'image ci-dessus indique le chemin à emprunter pour assurer le bon fonctionnement du capteur et de l'ESP32-WROOM.

### Connexion sur le Wifi IOT

Pour la connexion au wifi j’ai utilisé la librairie WiFi.h , pour effectuer la connexion j’ai créé 2 constants chars afin de mettre le SSID du réseau IOT Une image contenant texte, Police, blanc, ligne

Le contenu généré par l’IA peut être incorrect.

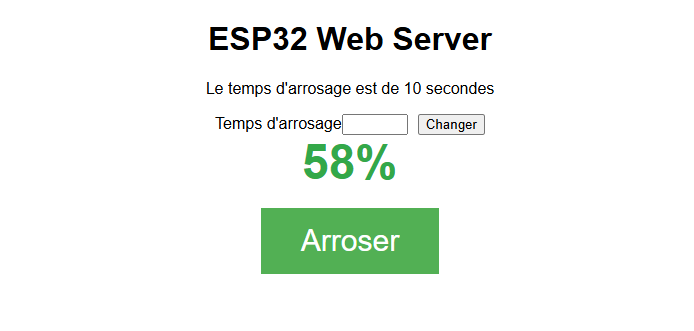


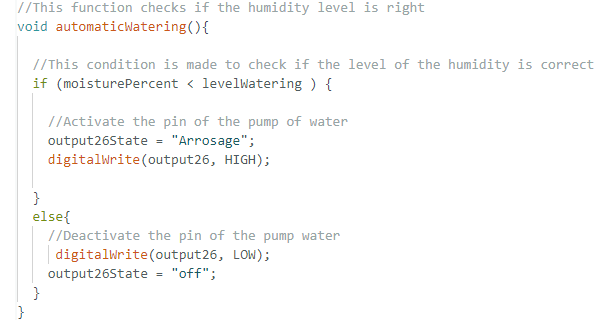
### Mise en œuvre du serveur HTTP

Une balise importante pour convertir le code html ccs dans l’Arduino

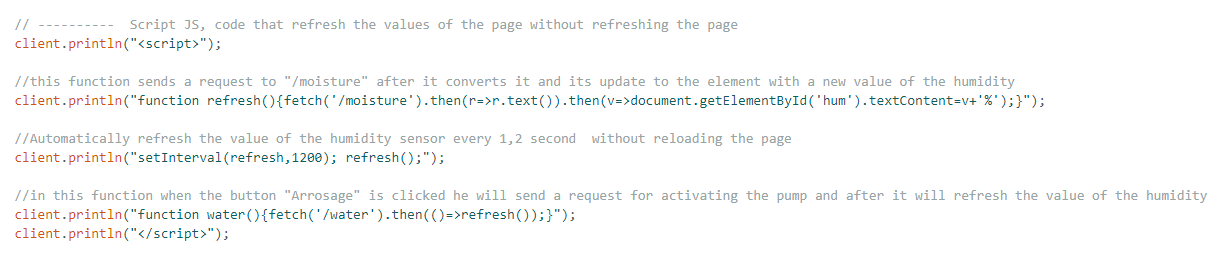
### Arrosage automatique

Cette partie du code était une de plus simple a réaliser ou il consiste a utiliser une condition afin de contrôler si l’humidité est plus bas que la valeur émise dans la condition du code la pin de la pompe vas s’activer afin d’arroser la plante .





### Arrosage manuel



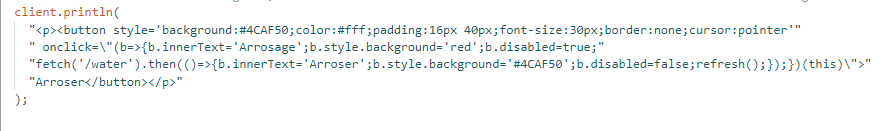
### Design Interface web

L’interface web est effectué avec du HTML CSS et java script explique pourquoi ? et pourquoi pas choisi comme ca ?

Affichage de l’historique uniquement le dernier arrosage doit s’afficher

Affichage de bouton grâce un script javascript :ce code a été généré par l’intelligence artificielle afin de pouvoir changer la couleur du bouton sans devoir actualisée la page

Explique que que avec du html et css en peux pas actualiser les boutons sans actualiser la page et c’est pour ca j’utilise du java script



### Stockage des logs

Cette section a pour objectif d’enregistrer, dans la mémoire de l’ESP32, les informations suivantes : la date, le type et le nom de la plante arrosée. Ces données sont ensuite affichées dans l’interface web afin que l’utilisateur puisse savoir quelle plante a été arrosée, à quel moment.

#### Objectifs principales

* Récupération de l’heure actuelle (via le réseau Wi-Fi)
* Écriture des données dans un fichier JSON
* Affichage du dernier historique d’arrosage uniquement dans l’interface web

#### Bibliothèques utilisées

* **SPIFFS.h** : permet de gérer le système de fichiers interne de l’ESP32 (stockage des fichiers JSON)
* **ArduinoJson.h** : utilisée pour créer, modifier et lire des fichiers au format JSON
* **time.h** : permet de récupérer l’heure depuis un serveur NTP via le Wi-Fi

#### Contenu du fichier JSON

Le fichier contiendra les champs suivants :

* « nom » : nom de la plante
* « type » : type de plante
* « annee, mois, jour, heure, minute »
* « humidite » : pourcentage d’humidité mesuré au moment de l’arrosage

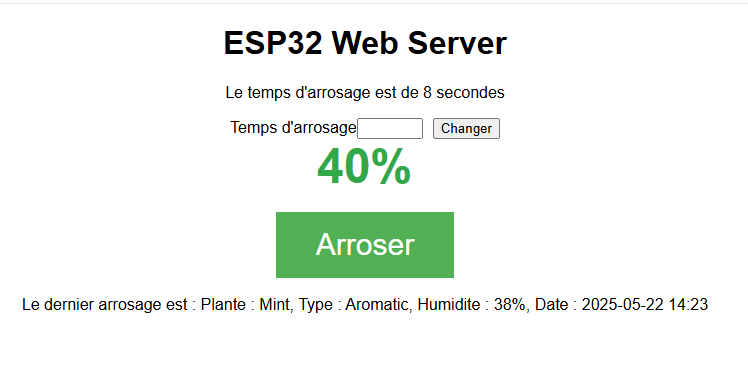
#### Structure du code

1. Récupération de l’heure
2. Ouverture du fichier JSON
3. Écriture des données (nom, type, date, humidité)
4. Fermeture du fichier
5. Affichage du dernier arrosage dans l’interface web

Le fichier contient uniquement le dernier enregistrement, ce qui permet de ne pas surcharger la mémoire de l’ESP32. À chaque arrosage, la nouvelle valeur remplace l’ancienne.

Le projet utilise deux capteurs d’humidité, donc il peut gérer deux plantes. Pour chaque plante, un fichier JSON est créé (log\_1.json et log\_2.json). Chaque fichier enregistre uniquement le dernier arrosage de la plante correspondante (nom, type, date, humidité). Cela permet de suivre séparément l’arrosage des deux plantes malgré le nombre limité de capteurs.

Et expliquer que après que la plante est arrosé il doit refrechir la page afin de voir le dernier enregistrement



### CRUD Plantes

## Description des tests effectués

* Après avoir raccordé le capteur à l’ESP32, j’ai vérifié que le terminal affichait bien une valeur et qu’aucun message d’erreur n’apparaissait.
* Pour contrôler la mesure d’humidité, j’ai immergé partiellement le capteur dans l’eau ; la variation immédiate de la lecture a confirmé le bon fonctionnement du capteur.
* Après établissement de la connexion Wi‑Fi, j’ai vérifié que l’ESP32 recevait bien une adresse IP libre par le serveur DHCP.
* Pour vérifier la création du site web, je me suis d'abord connecté au réseau, puis à l'interface web, où j'ai ajouté un texte tel que 'TEST' afin de m'assurer que l'affichage fonctionnait correctement.
* Pour vérifier l'affichage de l’humidité sur le site web, j’ai immergé le capteur dans l’eau afin d’observer si la valeur affichée se mettait à jour.
* Par la suite, j’ai choisi d’utiliser une LED afin de tester le fonctionnement du bouton d’activation et de désactivation. Je l’ai connectée entre le GND et le 3,3 V pour vérifier son bon fonctionnement avant de l’intégrer au contrôle via l’interface web.
* Le test du bouton d’arrosage a été réalisé à l’aide d’une LED. Lorsque je cliquais sur le bouton depuis l’interface web, la LED s’allumait pendant un certain temps, puis s’éteignait automatiquement.
* Le test du bouton d’arrosage a été réalisé à l’aide d’une LED afin de mieux visualiser le fonctionnement du système. Lorsque je cliquais sur le bouton depuis l’interface web, la LED s’allumait pendant un certain temps, puis s’éteignait automatiquement.
* Pour vérifier si l’arrosage a bien été activé via le bouton de l’interface web, celui-ci devient rouge pendant l’arrosage, puis redevient vert une fois l’arrosage terminé.

## Problèmes rencontrés

* Problème capteur d’humidité
* Problème actualisation de l’interface web
* Problème de connaissance fichier JSON
* Problème pompe
* Problème changement de couleur du buton sans actualiser la page

## Erreurs restantes

*S'il reste encore des erreurs:*

* *Description détaillée*
* *Conséquences sur l'utilisation du produit*
* *Actions envisagées ou possibles*

## Liste des documents fournis

*Lister les documents fournis au client avec votre produit, en indiquant les numéros de versions*

* *le rapport de projet*
* *le manuel d'Installation (en annexe)*
* *le manuel d'Utilisation avec des exemples graphiques (en annexe)*
* *autres…*

# Conclusions

*Développez en tous cas les points suivants:*

* *Objectifs atteints / non-atteints*
* *Points positifs / négatifs*
* *Difficultés particulières*
* *Suites possibles pour le projet (évolutions & améliorations)*

## Bilan des fonctionnalités

## Objectifs atteints

## Difficultés rencontres

## Bilan personnel

## Résultat de stratégie de test

## Suites du projet

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

## Sources – Bibliographie

### Installations

* Installation du Driver pour permettre de détecter l'ESP32 sur l’ordinateur :

<https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all>

* Installation de Arduino IDE 2.3.4 en version Windows 10 : <https://www.arduino.cc/en/software>

### Tutoriels

* System d’arrosage Automatique : <https://esp32io.com/tutorials/esp32-automatic-irrigation-system>
* Caractéristiques ESP32-WROOM :
* <https://www.espressif.com/sites/default/files/documentation/esp32-wroom32_datasheet_en.pdf>
* <https://www.youtube.com/watch?v=QUNKY87Da7A&t=445s>
* <https://www.circuits-diy.com/how-to-use-oled-displays-with-esp32-boards/>
* <https://www.youtube.com/watch?v=QUNKY87Da7A>
* GIT : <https://about.gitlab.com/fr-fr/blog/2024/10/08/what-is-git/>
* Capteur FC-28 : <https://www.arduinointro.com/articles/projects/arduino-soil-moisture-sensor-project-code-setup-and-tips-for-beginners>
* <https://components101.com/modules/soil-moisture-sensor-module>
* Affichage humidité su capteur : <https://www.electronicwings.com/esp32/soil-moisture-sensor-interfacing-with-esp32>
* SPIFFS : <https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/#:~:text=Introducing%20SPIFFS,-SPIFFS%20lets%20you&text=You%20can%20read%2C%20write%2C%20close,Create%20configuration%20files%20with%20settings>;
* JSON: <https://www.w3schools.com/js/js_json_intro.asp>
* PlatformIO : <https://www.youtube.com/watch?v=QMYhVqjBhKQ>
* Mise en place du serveur WEB : <https://www.electronicwings.com/esp32/soil-moisture-sensor-interfacing-with-esp32>

### Librairies

ESP32 : <https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json>

* Adafruit SSD1306, pour gérer le contrôleur de l’écran ;
* Adafruit GFX, pour le rendu graphique.

### Achats

## Intelligence artificielle

Face à certaines difficultés, notamment liées à la langue française ou à des blocages techniques ponctuels, j’ai utilisé l’intelligence artificielle de manière ciblée, parfois pour reformuler une phrase, parfois pour corriger une tournure, et d’autres fois pour m’aider à réfléchir à une solution quand je ne trouvais pas immédiatement d’issue, tout en gardant une approche personnelle, raisonnée et limitée dans son usage.

## Lexique

## Utilisation de l’intelligence artificielle

## Journal de travail

Le journal de travail doit être envoyé chaque lundi et vendredi en dernière heure, au format PDF, sous le nom :

"**2025.XX.XX-Sulejmani-Journal-Travail**".

## Rapport

Le rapport doit être envoyé chaque vendredi en dernière heure, au format PDF, sous le nom :

"**2025.XX.XX-Sulejmani-Rapport".**

## Manuel d'Installation

## Manuel d'Utilisation

## Archives du projet

*Media, … dans une fourre en plastique*