|  |
| --- |
| 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](#_Toc197680691)

[1.1 Introduction 4](#_Toc197680692)

[*1.2* Objectifs 5](#_Toc197680693)

[*1.3* Planification initiale 6](#_Toc197680694)

[1.4 Méthodologie du projet 8](#_Toc197680695)

[2 Analyse / Conception 9](#_Toc197680696)

[2.1 SWOT 9](#_Toc197680697)

[*2.2* Concept 9](#_Toc197680698)

[2.3 Architecture du Système 12](#_Toc197680699)

[2.4 Stratégie de test 12](#_Toc197680700)

[2.5 Risques techniques 13](#_Toc197680701)

[2.6 Matériel et logiciels à disposition 13](#_Toc197680702)

[2.6.1 ESP32-WROOM 14](#_Toc197680703)

[2.6.2 Ecran OLED 14](#_Toc197680704)

[2.6.3 Module de capteur d’humidité du sol 14](#_Toc197680705)

[2.6.4 SPIFFS 14](#_Toc197680706)

[2.6.5 JSON 14](#_Toc197680707)

[2.6.6 CRUD 14](#_Toc197680708)

[2.6.7 Arduino IDE 14](#_Toc197680709)

[2.6.8 GIT 14](#_Toc197680710)

[2.7 Maquette 14](#_Toc197680711)

[3 Réalisation 15](#_Toc197680712)

[*3.1* Dossier de réalisation 16](#_Toc197680713)

[3.1.1 Liaison du matériel 16](#_Toc197680714)

[3.1.2 Connexion sur le Wifi IOT 17](#_Toc197680715)

[3.1.3 Mise en œuvre du serveur HTTP 20](#_Toc197680716)

[3.1.4 Mesure de l’humidité 17](#_Toc197680717)

[3.1.5 Interface web 20](#_Toc197680718)

[3.1.6 Stockage des données 20](#_Toc197680719)

[*3.2* Description des tests effectués 20](#_Toc197680720)

[*3.3* Erreurs restantes 20](#_Toc197680721)

[*3.4* Liste des documents fournis 21](#_Toc197680722)

[4 Conclusions 21](#_Toc197680723)

[4.1 Bilan des fonctionnalités 22](#_Toc197680724)

[4.2 Objectifs attaints 22](#_Toc197680725)

[4.3 Difficultés rencontres 22](#_Toc197680726)

[4.4 Bilan personnelle 22](#_Toc197680727)

[4.5 Résultat de stratégie de test 22](#_Toc197680728)

[4.6 Suites du projet 22](#_Toc197680729)

[5 Annexes 22](#_Toc197680730)

[*5.1* Résumé du rapport du TPI / version succincte de la documentation 22](#_Toc197680731)

[5.2 Sources – Bibliographie 22](#_Toc197680732)

[5.2.1 Installations 22](#_Toc197680733)

[5.2.2 Tutoriels 22](#_Toc197680734)

[5.2.3 Librairies 22](#_Toc197680735)

[5.2.4 Achats 23](#_Toc197680736)

[5.3 Lexique 23](#_Toc197680737)

[5.4 Utilisation de l’intelligence artificielle 23](#_Toc197680738)

[5.5 Journal de travail 23](#_Toc197680739)

[5.6 Rapport 23](#_Toc197680740)

[*5.7* Manuel d'Installation 23](#_Toc197680741)

[*5.8* Manuel d'Utilisation 23](#_Toc197680742)

[*5.9* Archives du projet 23](#_Toc197680743)

**Tableau des légendes**

# Analyse préliminaire

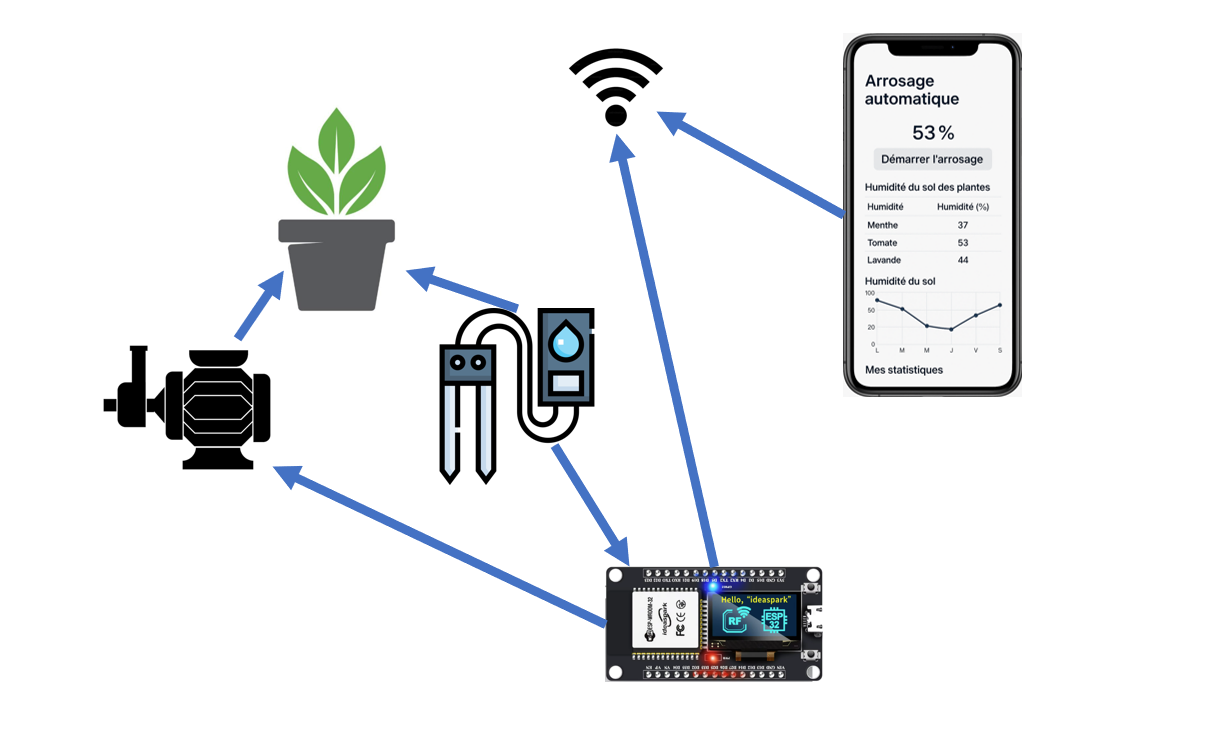
La recherche préliminaire établit les bases sur lesquelles le projet est fondé et ses objectifs principaux.

L'objectif de ce projet est de créer un système d’arrosage automatique ou manuelle gère 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.

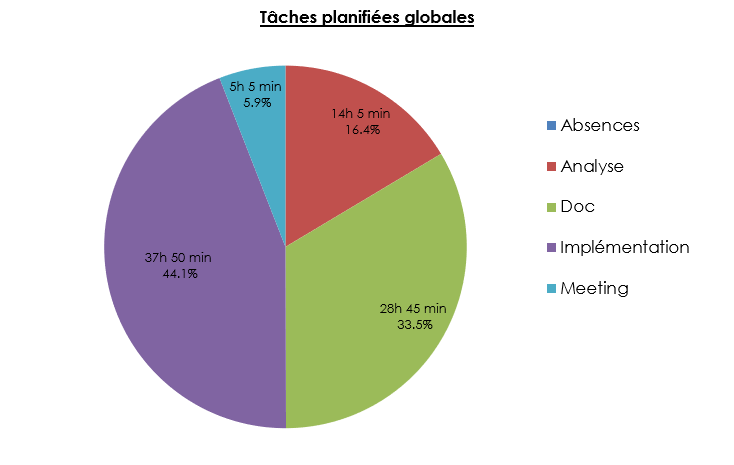
**Doc** : cette tâche comprend la rédaction du rapport, du journal de travail et des annexes.

**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 components 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.

Une autres

# 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

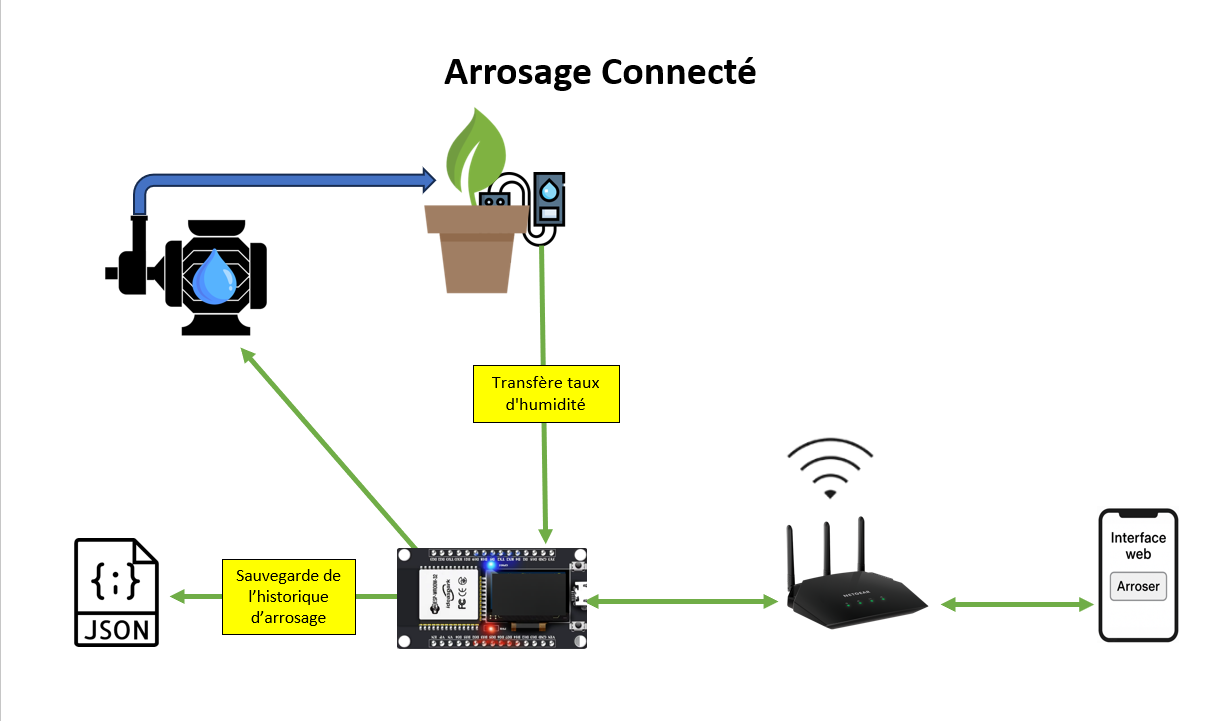
Avant de commencer le projet, j’ai pris une décision simple : analyser ma stratégie.  
Pourquoi ?

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.

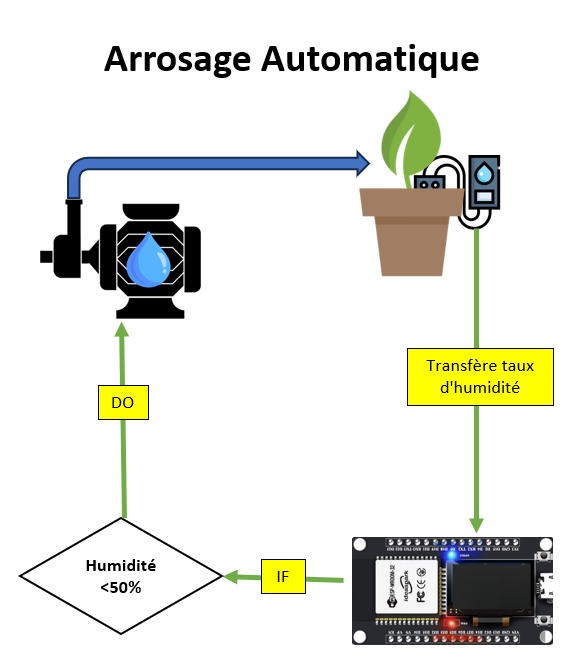
|  |  |
| --- | --- |
| **Forces** | **Faiblesses** |
| * Connaissance dans la plateforme Arduino IDE. * Connaissance en C#. | * Manque d'expérience avec SPIFFS * Perte de temps significative lors de la rédaction du cahier des charges en raison de difficultés avec la langue française. * Connaissances 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

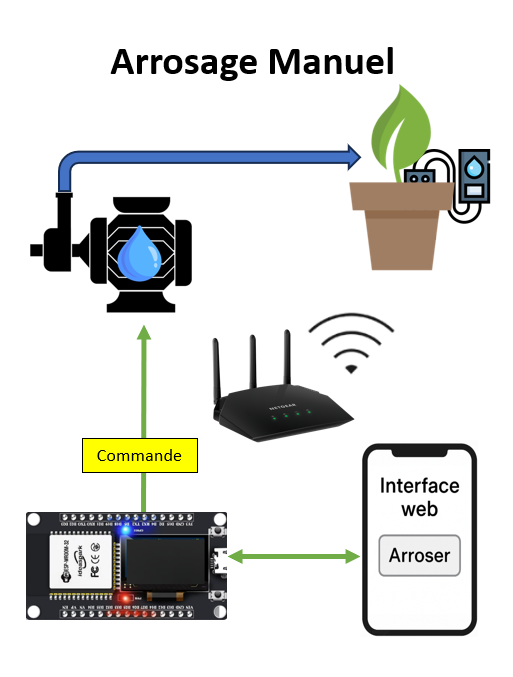
Ce projet vise à automatiser l’arrosage d’une plante sans intervention manuelle, offrant une solution pratique et connectée pour la maison. L’utilisateur peut surveiller et déclencher l’arrosage à distance, directement depuis une interface web accessible via son smartphone, sans avoir à se déplacer. 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

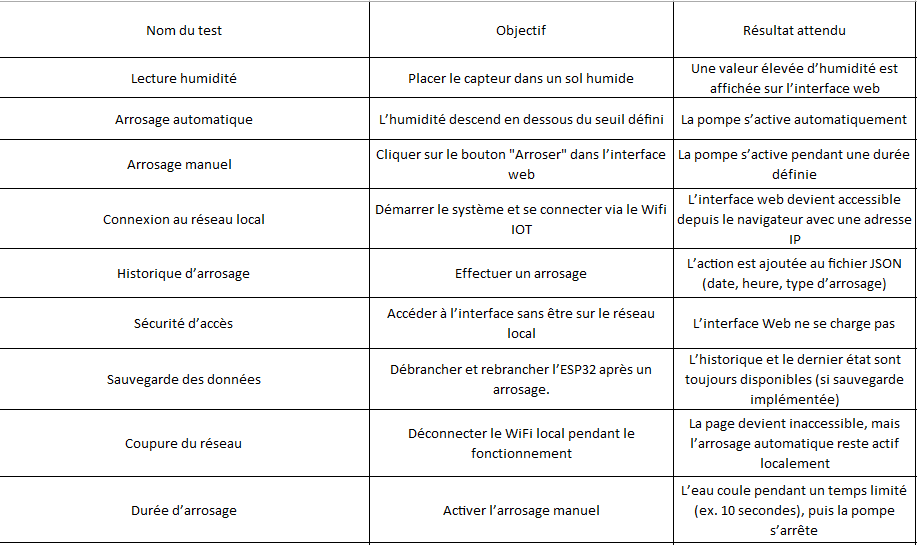
Chemin électrique

## 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

## Matériel et logiciels à disposition

* 1 microcontrôleur ESP32(ex. ESP32-WROOM-32)
* Écran OLED 0,96 128×64 px (driver SSD1306, bus I²C)
* Capteur d'humidité du sol FC-28
* Relais 5V SRD-05VDC-SL-C pour contrôler une pompe ou électrovanne
* 1 ordinateur type PC Windows
* Câbles Dupont
* 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
* Driver pour la reconnaissance de l’ESP32 CH340
* 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
* **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.
* **Affichage OLED** : L’écran OLED intégré au module est utilisé pour afficher clairement des informations utiles telles que le pourcentage d’humidité actuel, l’état de la pompe (activée/désactivée), et l’adresse IP pour la connexion web.
* **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.

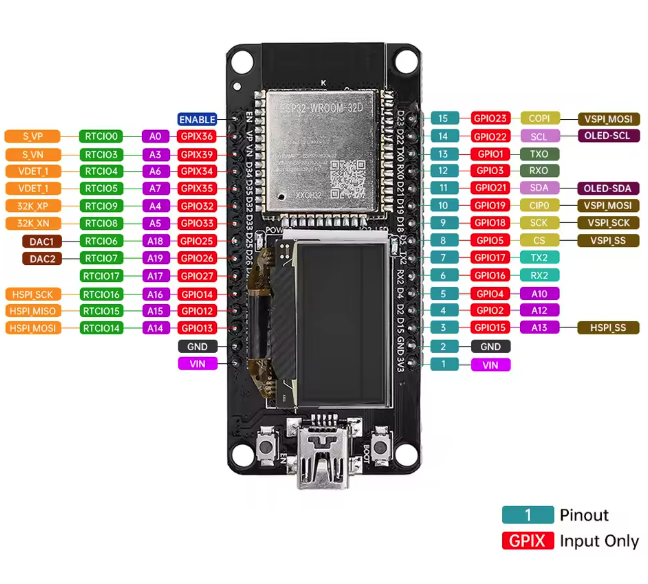
#### Différences majeures

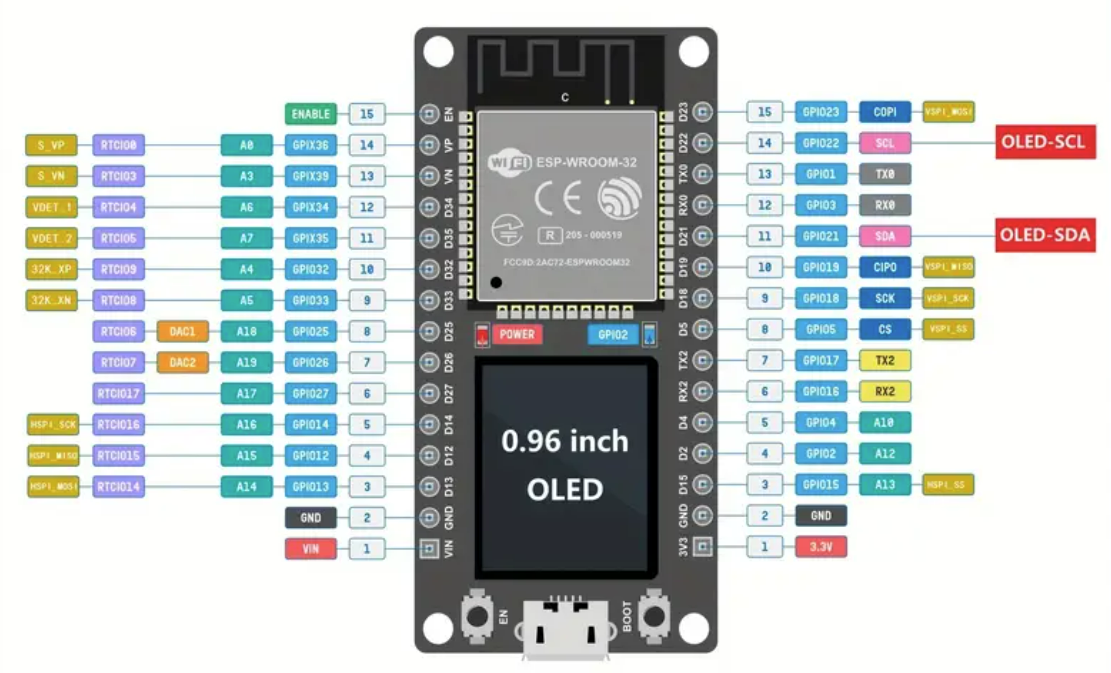
**Ordinateur vs ESP32** : Un ordinateur possède une puissance de calcul et une mémoire largement supérieure, adaptée pour des tâches complexes, tandis que l'ESP32 est un microcontrôleur optimisé pour des tâches spécifiques à faible consommation énergétique, idéal pour des systèmes embarqués et connectés.

**Raspberry Pi vs ESP32** : Le Raspberry Pi est un micro-ordinateur capable d'exécuter un système d'exploitation complet (Linux), convenant à des applications nécessitant une interface utilisateur complexe ou une grande capacité de stockage. En comparaison, 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 : 

****

* **Rouge** : Entrées/sorties numériques.
* **Orange** : Entrées analogiques et fonctions RTC (Real-Time Clock).
* **Vert foncé** : Communication I2C pour périphériques (OLED intégré).
* **Jaune** : Communication SPI (standard VSPI).
* **Marron** : SPI rapide (HSPI).
* **Violet** : Fonctions spéciales (DAC, entrées analogiques spécifiques).
* **Noir** : Masse électrique (GND).
* **Rose** : Alimentation externe (VIN).
* Ces couleurs facilitent l'identification rapide des fonctions lors du montage du projet.

### Ecran OLED

Pour faire fonctionner l’écran oled sur l’application Arduino il faut utiliser la bibliothèque SSD1306 d’Adafruit

Les pins qu’elle occupe sont les pins GPIO11 et GPIO14

### 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

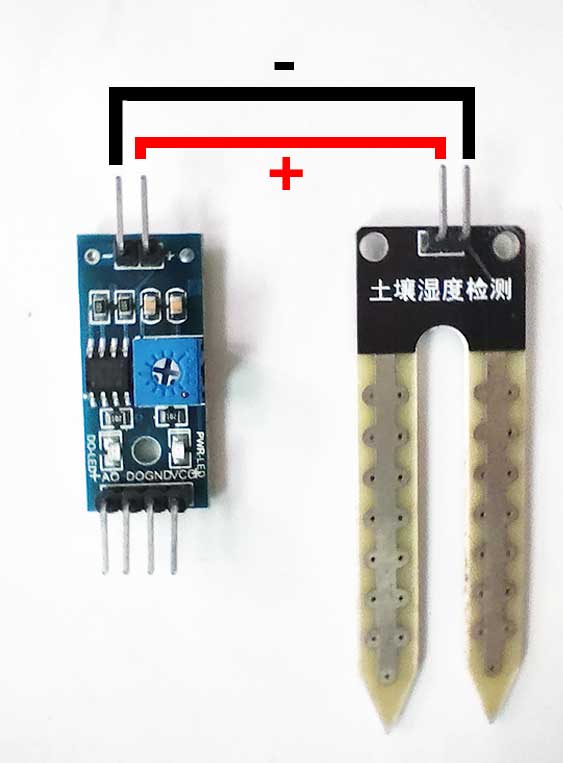
* Lorsque le sol est sec, la résistance est élevée → faible signal.
* Lorsque le sol est humide, la résistance est basse → signal élevé.
* Le module fournit :
* Une sortie analogique (AO) : valeur de l’humidité.
* Une sortie numérique (DO) : niveau haut ou bas selon un seuil réglable.

#### Avantages

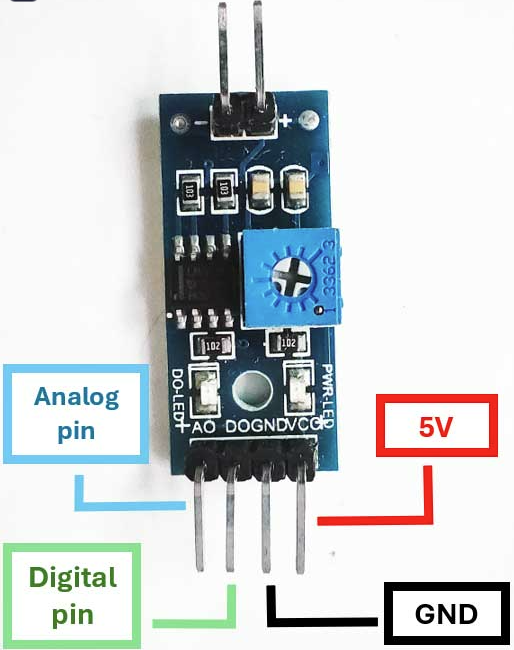
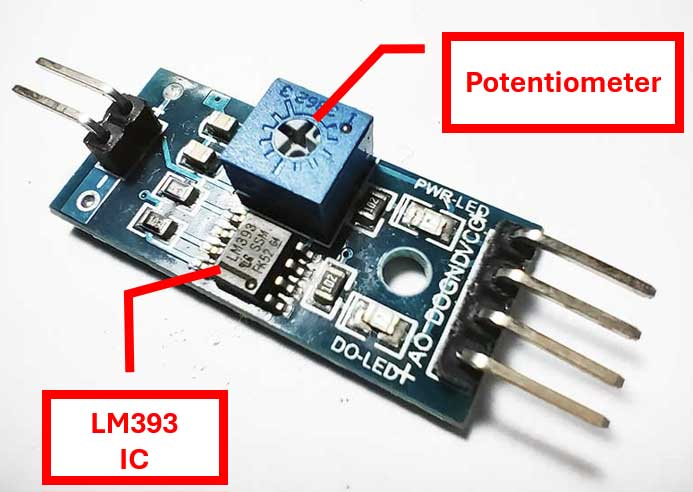
* Facile à utiliser avec Arduino ou ESP32
* Double sortie : analogique (précise) et numérique (seuil réglable)
* Coût faible

#### Inconvénients

* Pas toujours précis

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.



### SPIFFS

### JSON

### CRUD

### Arduino IDE

### GIT

## Maquette

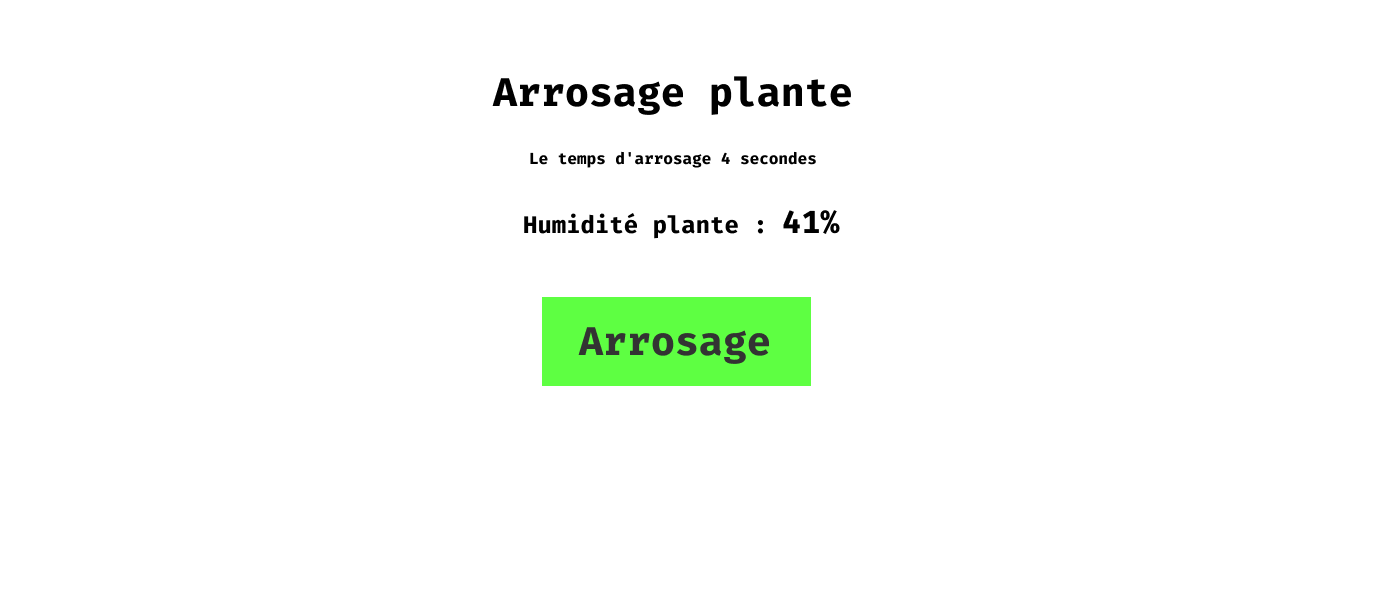
Pour concevoir la maquette, j’ai retenu Figma : je recherchais un outil à la fois intuitif, facile à piloter et rapide à mettre en œuvre, d’autant que le projet ne comprend qu’une seule interface.

### Interface web

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 champ pour régler la durée d’arrosage, toujours exprimée en secondes et limitée à 60 s pour éviter de dépasser une minute ; toute modification dans le code se reflète instantanément à l’écran.
* 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.

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

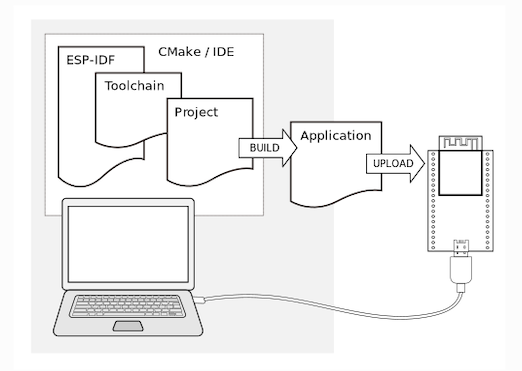


# Réalisation

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

**

*Décrire la réalisation "physique" de votre projet*

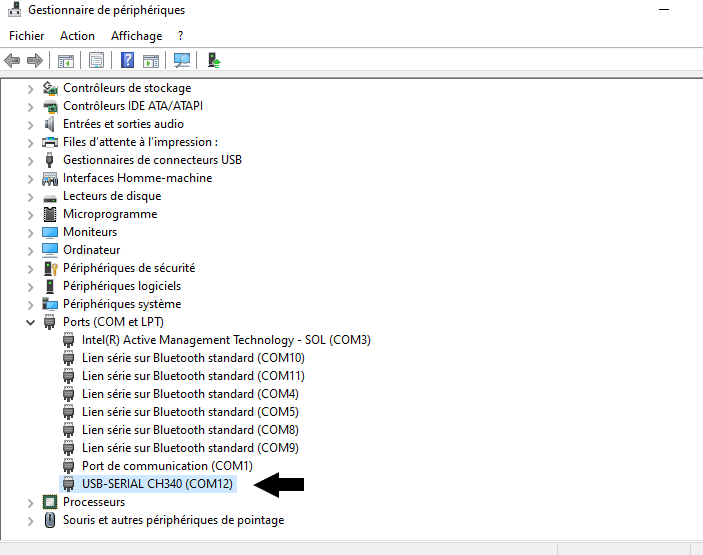
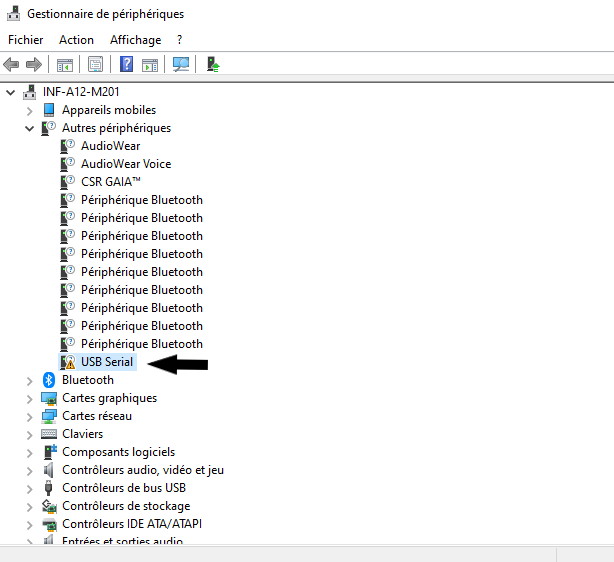
* *les répertoires où le logiciel est installé*
* *la liste de tous les fichiers et une rapide description de leur contenu (des noms qui parlent !)*
* *les versions des systèmes d'exploitation et des outils logiciels*
* *la description exacte du matériel*
* *le numéro de version de votre produit !*
* *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…*

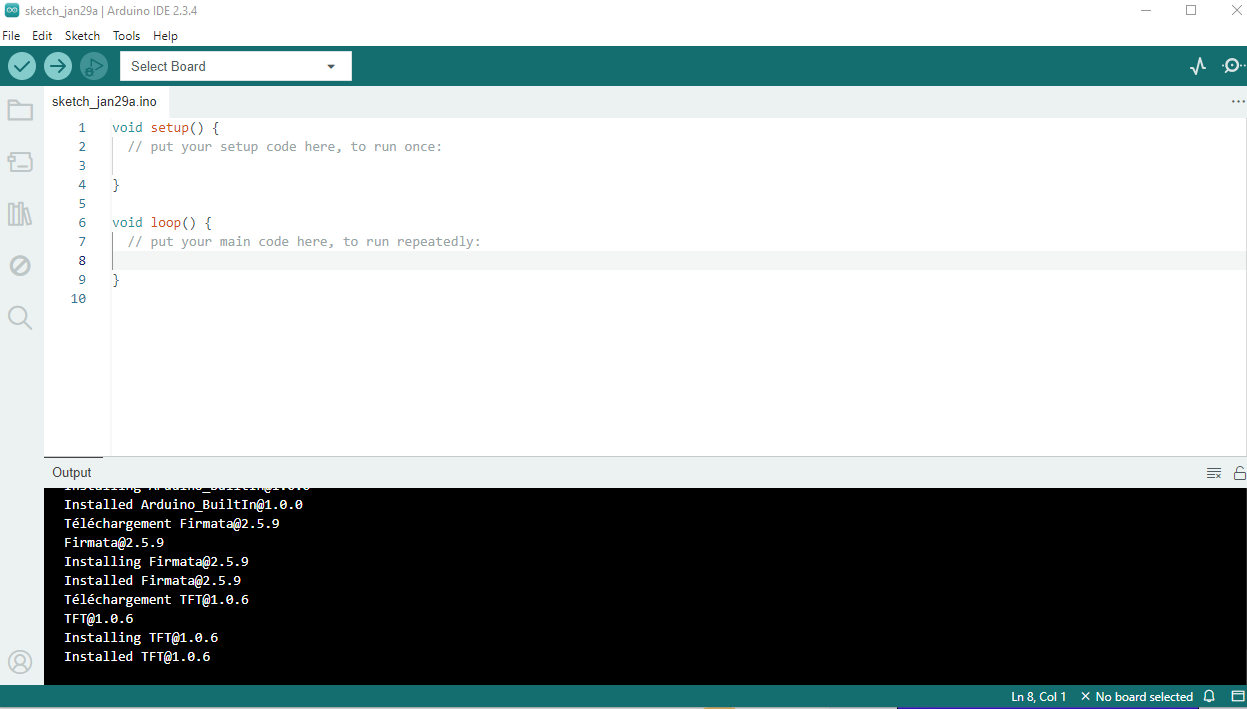
### Liaison du matériel

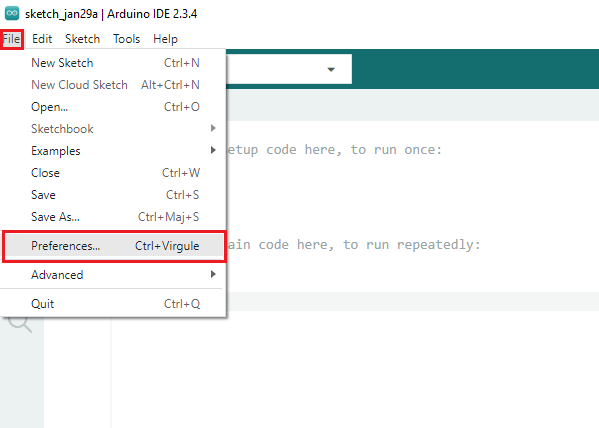
### Configuration du système

* J’ai dû télécharger *le driver CH340 « Chapitre 5.2 » sur Windows 10 pour que l'ordinateur puisse reconnaître l'ESP32.Sur Windows 11 testé chez moi, c’est natif. Ici une photo sans le driver après la connexion de l’ESP32-CAM et une autre avec le driver installé.*

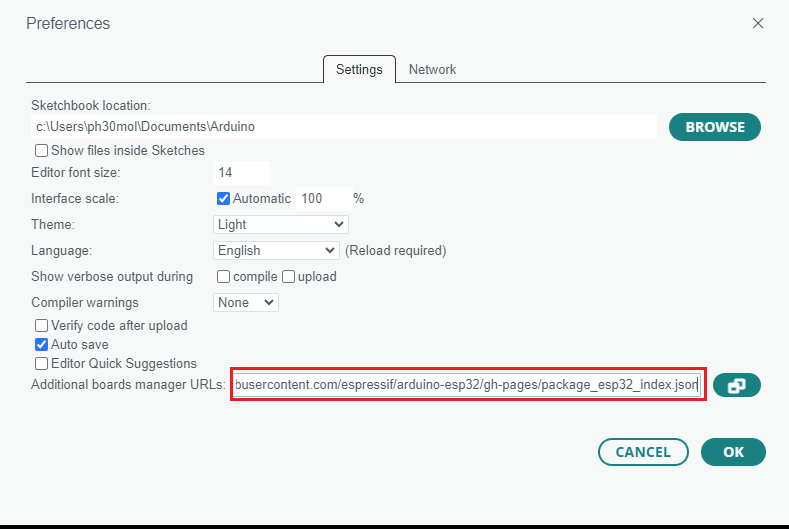
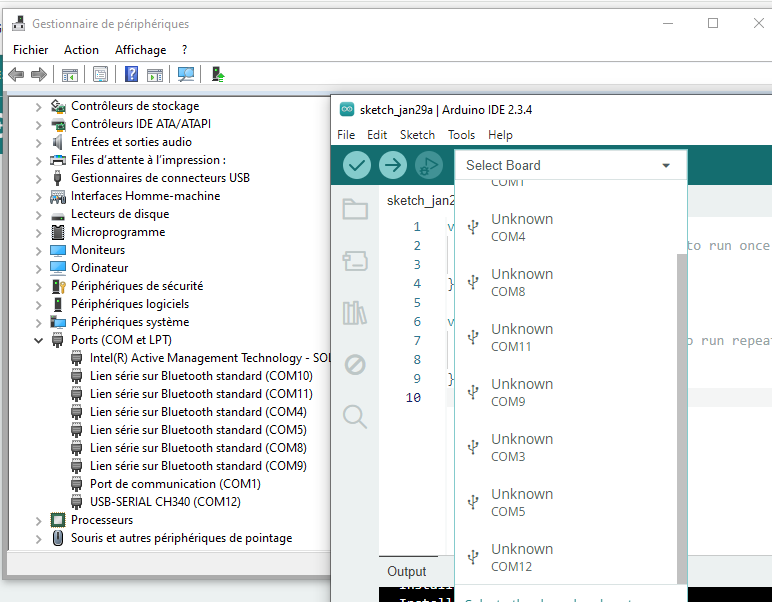


### Installation et configuration Arduino IDE

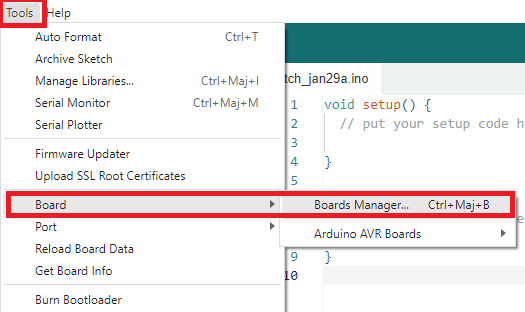
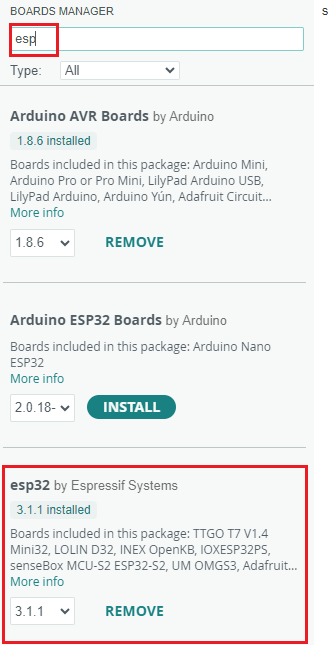
* J’ai installe l’application Arduino IDE 2.3.6« Chapitre 5.2 » qui ma permis de coder l’environement du projet.
* Contrôle de port à choisir pour la communication avec l'ESP32.
* URL qui permet d’ajouter la librairie de base pour le ESP32.

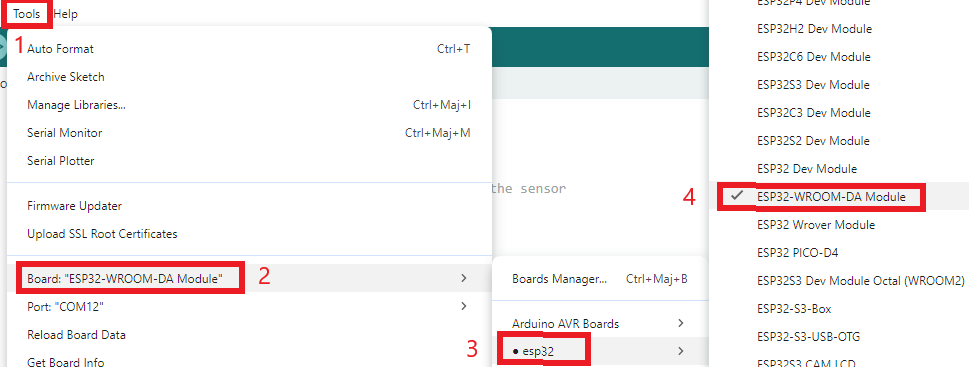
****

* *« Chapitre 5.2 »*



* Il faut installer le gestionnaire des cartes « esp32» pour permettre d'interagir avec le ESP32.

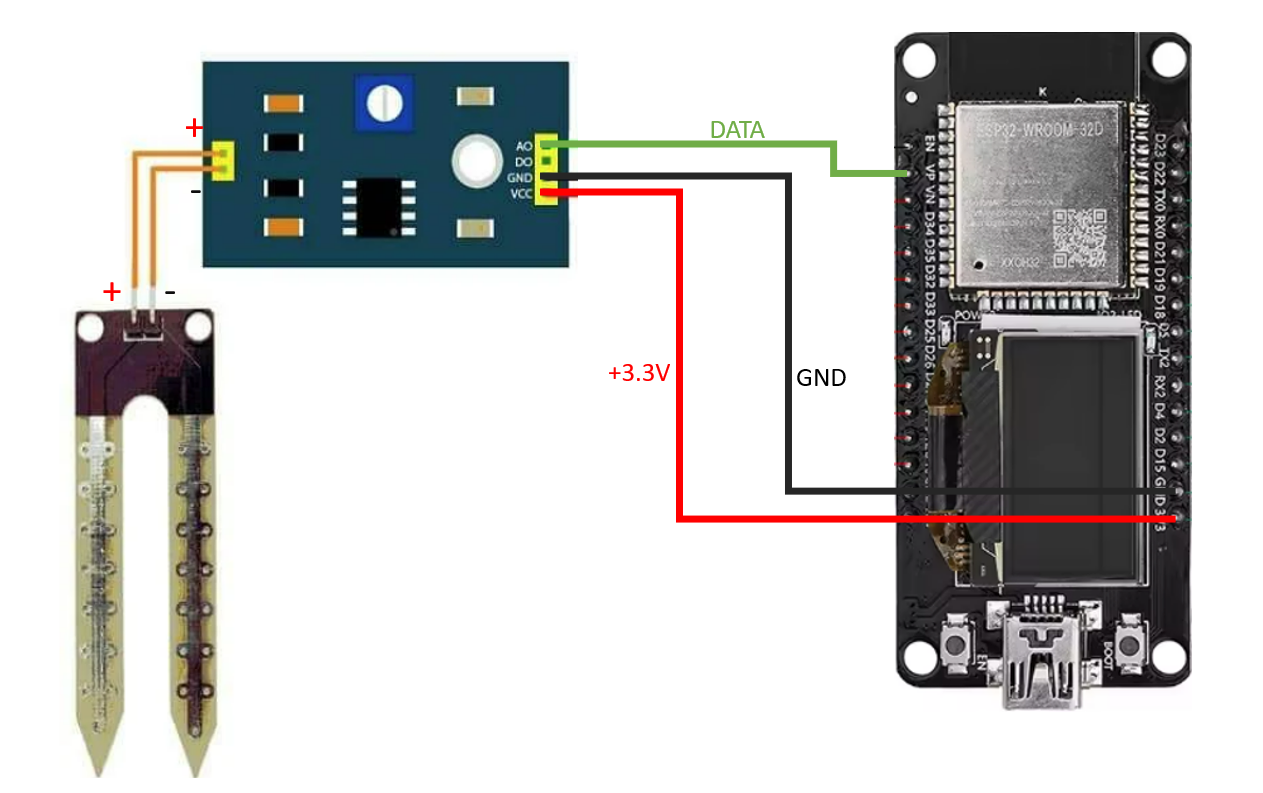




### Mesure de l’humidité

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**

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

### Mise en œuvre du serveur HTTP

### Arrosage automatique

### Arrosage manuel

### Design Interface web

### Stockage des données

## 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 chaque partie testée de votre projet, il faut décrire:*

* *les conditions exactes de chaque test*
* *les preuves de test (papier ou fichier)*
* *tests sans preuve: fournir au moins une description*

## 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 attaints

## Difficultés rencontres

## Bilan personnelle

## 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-wroom-32_datasheet_en.pdf>
* <https://www.youtube.com/watch?v=QUNKY87Da7A&t=445s>
* 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>
* 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>

### Achats

## 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*