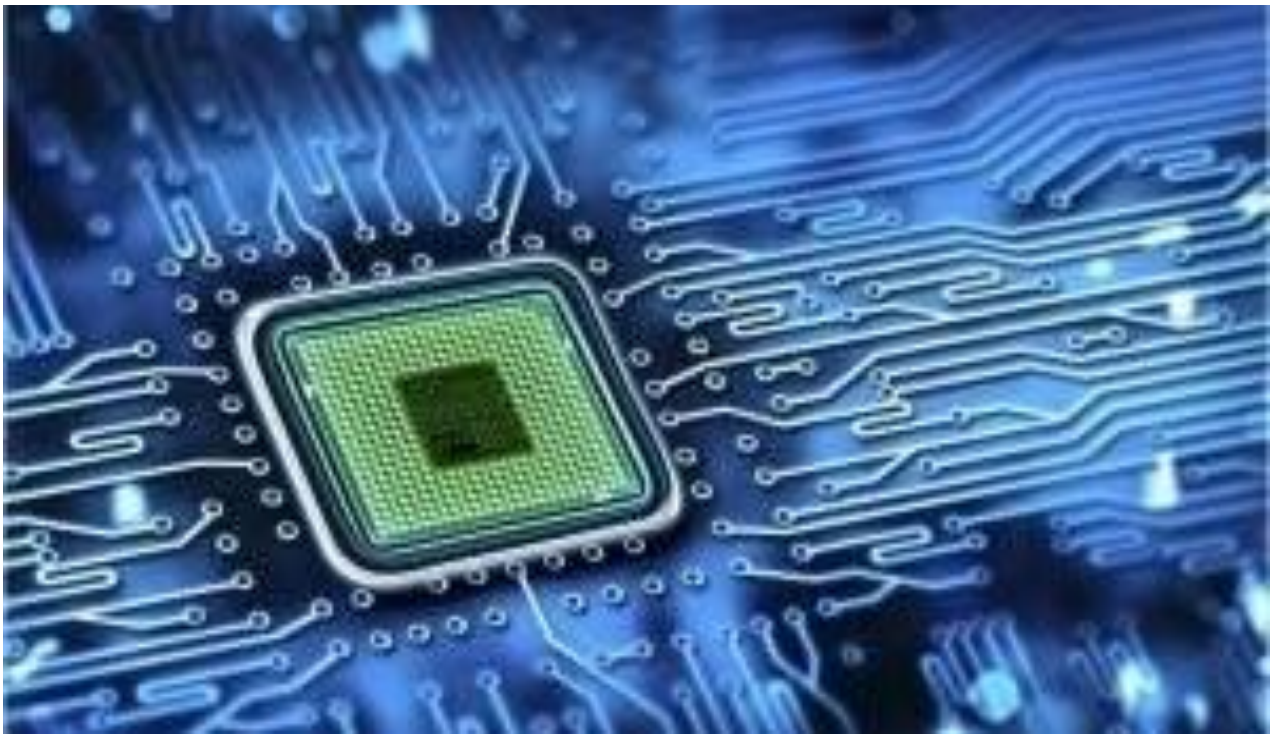
Royaume du Maroc
Université Sidi Mohammed Ben Abdellah
EcoleNationale desSciences Appliquées de Fès

المملكة الوطنية
جامعة سيدي محمد بن عبد الله
المدرسة الوطنية للعلوم التطبيقية فاس

Projet Soft Embarqué



Réalisé par :

- **BIYADI Wissal**

Année universitaire 2022/2023

Introduction générale

Les systèmes embarqués sont de plus en plus utilisés dans notre vie quotidienne. Ils sont présents dans les voitures, les téléphones portables, les appareils ménagers et même dans les équipements médicaux. Un système embarqué est un système électronique qui est intégré dans un autre système plus grand et qui est conçu pour effectuer une tâche spécifique.

Le développement de systèmes embarqués nécessite une compréhension approfondie des architectures matérielles et logicielles, ainsi qu'une connaissance approfondie des langages de programmation, des outils de développement et des protocoles de communication.

Dans ce rapport, nous présentons le développement d'un système embarqué basé sur un Raspberry Pi, qui a pour but de contrôler un système d'arrosage automatique. Ce système est composé d'un capteur d'humidité de sol, d'un capteur de température et d'humidité, d'une pompe et de plusieurs LED.

Le rapport présentera les différentes étapes de développement du système embarqué, du choix des composants matériels à la programmation du logiciel. Nous expliquerons également comment nous avons intégré ces différents composants pour créer un système d'arrosage automatique capable de mesurer l'humidité du sol et d'arroser les plantes en conséquence.

Enfin, nous discuterons des résultats de nos tests et des améliorations possibles pour le système d'arrosage automatique.

Table des matières

Chapitre 1 : Contexte générale du projet

Introduction

Le développement de systèmes client-serveur est devenu essentiel dans le monde de la technologie actuelle. Les applications client-serveur permettent une communication et une interaction entre des machines distantes. Le serveur fournit des services aux clients, qui peuvent être des ordinateurs ou des appareils mobiles. Le client envoie des requêtes au serveur, qui répond avec les résultats demandés. Dans ce rapport, nous allons explorer la conception et la mise en œuvre d'une application client-serveur pour contrôler et surveiller des dispositifs à distance.

1. Contexte

Le projet d'arrosage automatique avec les capteurs que vous avez utilisés repose sur une architecture client-serveur. Le client est une interface web en PHP qui permet de contrôler les différents composants du système d'arrosage tels que la pompe, les LED et les capteurs. Le serveur est un Raspberry Pi qui gère les différents composants physiques du système.

Les capteurs utilisés, tels que le capteur de sol et le capteur de température et d'humidité (DHT11), permettent de mesurer le niveau d'humidité du sol et de l'air, ce qui permet au système de déterminer quand arroser les plantes. Le capteur ultrasonique est également utilisé pour mesurer la quantité d'eau dans le sol, ce qui permet au système de fournir la quantité d'eau nécessaire pour maintenir un niveau d'humidité approprié.

Le système utilise une combinaison de code C et Python pour gérer les différents composants et les capteurs. Les données collectées par les capteurs sont stockées dans une base de données et peuvent être consultées en temps réel via l'interface web.

Dans l'ensemble, le projet vise à fournir un système d'arrosage automatique efficace et facile à utiliser qui permet de maintenir un niveau d'humidité approprié pour les plantes en utilisant des capteurs et une interface web conviviale.

2. Cahier de charges

a. Problématique

Lorsqu'il s'agit d'entretenir des plantes, le bon arrosage est crucial pour leur survie. Cependant, il peut être difficile de maintenir un niveau d'humidité optimal sans avoir à y consacrer beaucoup de temps et d'efforts. C'est pourquoi la mise en place d'un système d'arrosage automatique peut être très utile. Cependant, pour que ce système soit efficace, il est important de pouvoir contrôler précisément le niveau d'eau dans le sol. Comment peut-on concevoir un tel système et l'interfacer avec des capteurs de manière à ce qu'il puisse répondre aux besoins des plantes de manière autonome ? Quelle est la meilleure architecture client-serveur à adopter pour ce type de système ?

b. Objectifs

Les objectifs de l'arrosage automatique dans votre projet peuvent inclure :

- ✓ Optimiser la croissance des plantes en leur fournissant la quantité d'eau adéquate en fonction de leur besoin.
- ✓ Économiser de l'eau en évitant les arrosages excessifs et en utilisant uniquement la quantité nécessaire pour chaque plante.
- ✓ Réduire la main-d'œuvre nécessaire pour l'arrosage manuel en automatisant le processus.
- ✓ Améliorer la santé des plantes en évitant les fluctuations de l'humidité du sol.
- ✓ Réduire le risque de maladies des plantes en évitant les arrosages excessifs qui peuvent favoriser la croissance des champignons et des bactéries.
- ✓ Prévenir le gaspillage d'eau en arrêtant automatiquement l'arrosage lorsqu'il pleut ou lorsque l'humidité du sol est suffisante.
- ✓ Offrir un moyen efficace de gérer l'arrosage de plusieurs plantes en même temps, en particulier pour les grandes surfaces de cultures ou les jardins.
- ✓ Permettre un arrosage régulier et constant, même pendant les absences prolongées des propriétaires, ce qui est particulièrement utile pour les vacances ou les déplacements professionnels.

Ces objectifs sont importants car ils permettent d'optimiser la croissance des plantes tout en économisant de l'eau et en réduisant la main-d'œuvre nécessaire pour l'arrosage manuel. De plus, l'automatisation de l'arrosage offre des avantages en termes de santé des plantes, de prévention des maladies, de gestion efficace de l'arrosage de plusieurs plantes en même temps, et de facilité de gestion lors des absences prolongées.

3. Méthodologie

a. Aspects techniques

Le projet d'arrosage automatique avec des capteurs nécessite la mise en place d'un système embarqué permettant de contrôler la quantité d'eau dans le sol. Pour cela, plusieurs aspects techniques doivent être pris en compte, notamment le choix des capteurs les plus adaptés pour mesurer l'humidité du sol et la température de l'environnement.

Il est également nécessaire de mettre en place une interface utilisateur conviviale permettant de contrôler l'actuateur (la pompe à eau) et de visualiser les données collectées par les capteurs.

Le développement de l'application nécessite également la mise en place d'une communication en temps réel entre le client et le serveur embarqué, ainsi que le stockage de l'historique des actions dans une base de données pour permettre une analyse ultérieure.

Enfin, la mise en place de la sécurité de l'ensemble du système est essentielle pour éviter tout accès non autorisé ou toute altération des données collectées.

b. Aspects économiques et financiers

Dans ce projet d'arrosage automatique, les aspects économiques et financiers sont importants à prendre en considération. En effet, l'utilisation de capteurs de qualité et de pompes fiables peuvent avoir un coût élevé. Cependant, cette dépense peut être compensée par les économies réalisées en évitant un arrosage excessif ou insuffisant, qui peut entraîner des pertes de récolte ou une surconsommation d'eau.

De plus, la mise en place d'un système d'arrosage automatique peut améliorer l'efficacité du travail, réduire les coûts de main-d'œuvre et améliorer la qualité de la récolte, ce qui peut entraîner une augmentation des revenus. Il est donc important d'évaluer les coûts initiaux ainsi que les économies potentielles à long terme pour déterminer la viabilité économique du projet.

c. Suivi selon le cycle en V

Dans le contexte de ce projet d'arrosage automatique, le suivi selon le cycle en V pourrait inclure les étapes suivantes :

Analyse des besoins : identification des besoins fonctionnels et non fonctionnels du système, analyse de l'existant et des contraintes liées à la mise en œuvre.

Conception : définition de l'architecture du système, conception des différentes fonctionnalités, choix des technologies et des outils à utiliser.

Développement : mise en place de l'infrastructure du système, développement des différents modules, tests unitaires, intégration et tests de validation.

Tests : tests fonctionnels et non fonctionnels, tests de performance, tests de sécurité, tests de validation avec les utilisateurs finaux.

Mise en production : déploiement du système sur le terrain, mise en place de la maintenance et de la documentation associées.

Maintenance : maintenance corrective et évolutive du système, suivi des indicateurs de performance, analyse des incidents et des retours d'expérience pour améliorer la qualité du système.

Chapitre 2 : Conception et modélisation du projet

Introduction

Dans ce chapitre, nous aborderons la phase de conception et de modélisation du projet d'arrosage automatique. Nous expliquerons comment nous avons identifié les besoins du système, spécifié les fonctions qu'il doit remplir et modélisé le système pour mieux comprendre son fonctionnement. Nous présenterons également les choix de conception que nous avons faits et les raisons derrière ces choix. Enfin, nous expliquerons comment nous avons testé et validé le système pour nous assurer qu'il remplit bien les fonctions pour lesquelles il a été conçu.

1. Environnement technique du projet :

Notre solution repose sur une combinaison de langages et de technologies soigneusement sélectionnés pour répondre aux exigences de notre système embarqué. Voici un aperçu de notre environnement technique :

Interface web : Nous avons choisi d'utiliser PHP comme langage de programmation pour notre interface web, combiné avec le serveur web open source lighttpd. Cette combinaison nous a permis de concevoir une interface web rapide, performante et légère, capable de répondre aux demandes des utilisateurs avec une faible latence.

Base de données : Pour stocker les actions effectuées par l'utilisateur, nous avons utilisé SQLite3, un système de gestion de bases de données léger et portable, qui est bien adapté aux applications embarquées et mobiles.

Développement des drivers et d'acquisition : Pour le développement de nos drivers et d'acquisition, nous avons opté pour le langage de programmation C. C'est un langage populaire dans les applications embarquées en raison de sa performance et de sa flexibilité.

Contrôle : Pour le contrôle, nous avons choisi Python, un langage de programmation polyvalent et populaire dans le domaine de l'analyse de données et de l'automatisation. Cette combinaison nous a permis de concevoir une solution logicielle complète et facilement extensible pour notre système embarqué.

En résumé, notre environnement technique repose sur une combinaison de langages et de technologies soigneusement sélectionnés pour offrir une solution performante, légère et facilement extensible pour notre système embarqué

2. Architecture générale du projet

Dans la partie serveur, on retrouve les différents composants matériels tels que la LED, le capteur DHT11, le capteur de sol, le capteur ultrasonique et la pompe. Le rôle de ces composants est de mesurer les données environnementales (température, humidité, niveau d'eau, etc.) et de contrôler l'arrosage des plantes en fonction de ces données.

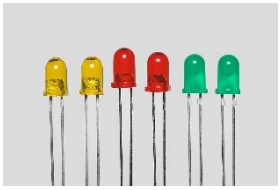
Le serveur héberge également une page web, développée en PHP et exécutée via le serveur web lighttpd. Cette page web permet à l'utilisateur de surveiller l'état des plantes et de contrôler le système d'arrosage en temps réel. Les données collectées par les capteurs sont stockées dans une base de données SQLite3 pour une utilisation ultérieure et l'analyse de ces données.

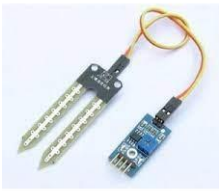
Dans la partie client, on retrouve l'interface utilisateur, qui permet à l'utilisateur de visualiser les données environnementales et de contrôler le système d'arrosage. Cette interface utilisateur est développée en HTML, CSS et JavaScript et est accessible depuis un navigateur web.

Les deux parties (client et serveur) communiquent entre elles à l'aide d'une API RESTful développée en Python, qui permet de transmettre les données de l'utilisateur et de contrôler les composants matériels du serveur en temps réel.

- **Serveur**

- **Client**



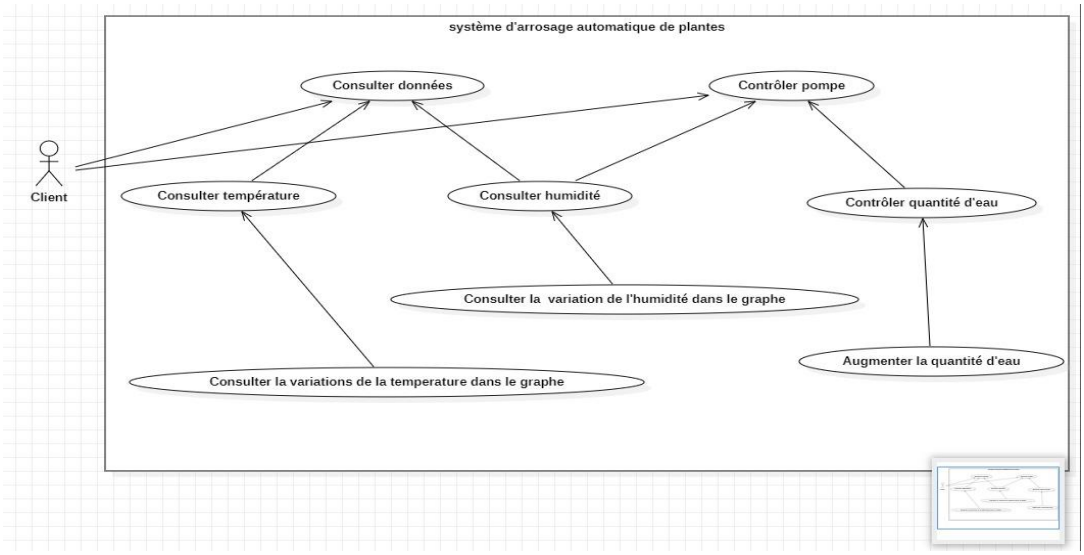



- **Interaction entre client et serveur**

3. Conception détaillée du projet

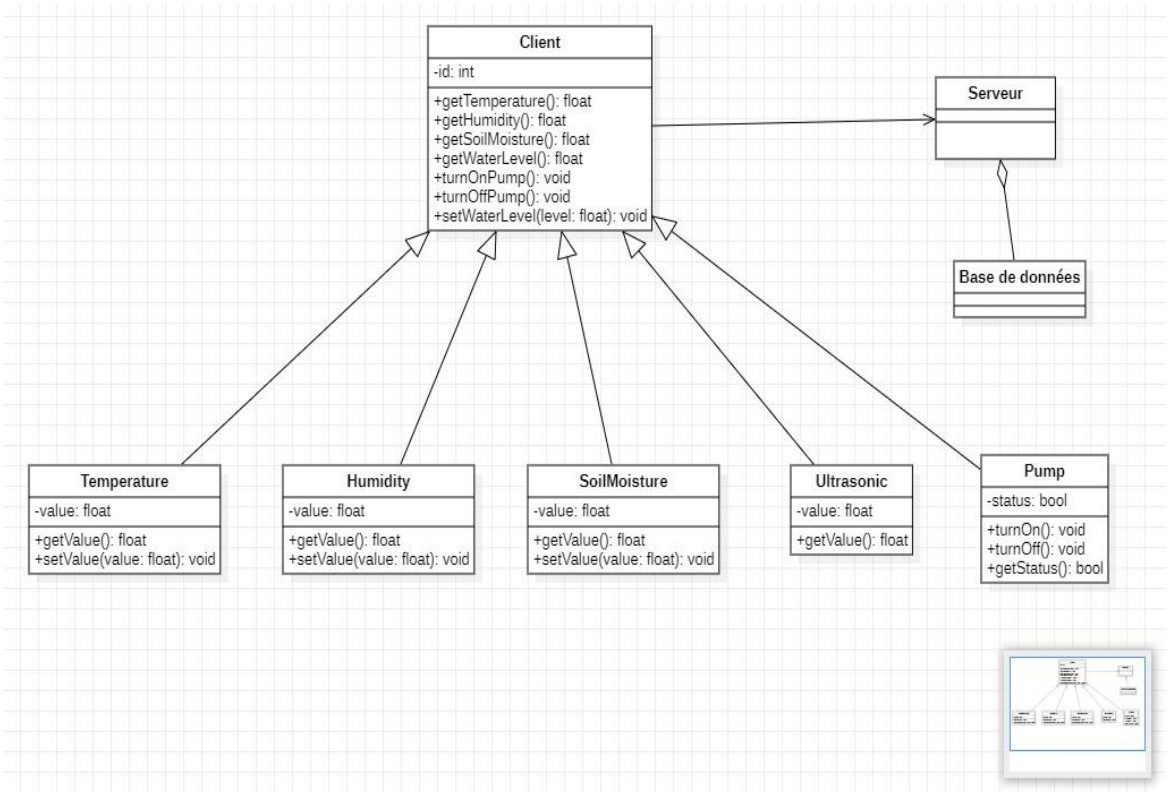
L'utilisation de diagrammes UML dans notre projet s'est avérée être un atout majeur pour le développement de notre système embarqué. UML, pour Unified Modeling Language, est un langage de modélisation graphique standardisé utilisé pour visualiser, spécifier, concevoir et documenter les différents aspects d'un système logiciel. Dans notre cas, les diagrammes UML nous ont permis de modéliser efficacement l'architecture de notre système, d'identifier les différentes composantes et leur interaction, ainsi que de définir les interfaces entre les différentes parties. Cette approche a facilité la collaboration entre les membres de l'équipe de développement, en offrant une vision globale du système, ainsi que des détails techniques spécifiques à chaque composante. Dans ce rapport, nous allons détailler les différents types de diagrammes UML que nous avons utilisés, ainsi que leur utilité respective dans la conception et la réalisation de notre système.

a. Diagramme des cas d'utilisation



Le diagramme de cas d'utilisation de notre projet permet de visualiser de manière synthétique les différentes actions que peut effectuer l'utilisateur et les différentes fonctionnalités que notre système offre. Grâce à ce diagramme, nous avons pu identifier les différents acteurs du système et leurs interactions avec celui-ci. Il a également permis de définir les fonctionnalités principales de notre application et les relations entre les différents cas d'utilisation. Le diagramme de cas d'utilisation a donc été un outil clé dans la définition des besoins du projet et dans la communication avec les différents acteurs impliqués.

b. Diagramme de classes



Le diagramme de classes de notre projet offre une vue d'ensemble de l'architecture de notre système. Il permet de visualiser les différentes classes qui composent notre application, leurs attributs et leurs méthodes, ainsi que les relations entre elles. Grâce à ce diagramme, nous avons pu concevoir une structure logique solide et cohérente pour notre système, en identifiant les différentes classes nécessaires à la mise en oeuvre de ses fonctionnalités. Le diagramme de classes a également facilité le développement de notre application en offrant une vue d'ensemble de l'ensemble des composants du système, permettant ainsi une gestion efficace des différentes fonctionnalités et des interactions entre les classes

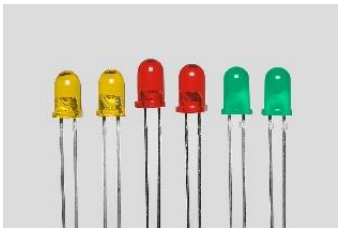
Chapitre 3 : Réalisation et test du projet

Introduction

Dans ce chapitre, nous allons nous intéresser à la réalisation concrète du projet et à sa mise en œuvre. Nous allons notamment détailler les étapes de conception et d'implémentation du système, en nous appuyant sur les spécifications définies dans le chapitre précédent. Nous verrons également comment nous avons réalisé les différents composants du projet, qu'il s'agisse du serveur web, des capteurs ou de la pompe. Enfin, nous aborderons les tests que nous avons menés pour valider le bon fonctionnement de l'ensemble du système, ainsi que les résultats obtenus.

1. Réalisation du projet

a. Leds :

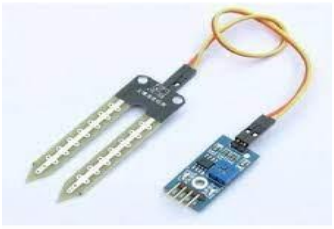


Les fichiers :

- ✓ web_led7.php
- ✓ test1.c
- ✓ io.c
- ✓ io.h

Ces codes permettent de contrôler des LEDs. Il utilise du PHP pour exécuter des commandes en fonction des boutons cliqués. Les LEDs sont représentées sous forme de boutons ON/OFF dans des conteneurs. Lorsqu'un bouton est cliqué, la page envoie une requête GET avec les paramètres correspondants pour allumer ou éteindre la LED correspondante.

b. Capteur Fc-28



Les fichiers :

- ✓ test1.php
- ✓ sol.py
- ✓ sol.db

test1.php : s'agit d'un code HTML et PHP pour une page Web qui affiche les 10 dernières valeurs d'un capteur de sol dans un tableau et dans un graphique. La page comprend également un bouton d'actualisation qui recharge la page pour mettre à jour les données toutes les 5 secondes.

Le code PHP se connecte à une base de données SQLite et récupère les 10 dernières valeurs d'une table appelée capteur_sol. Il crée ensuite deux tableaux, un pour les horodatages et un pour les valeurs des capteurs, qui sont utilisés pour créer un graphique de tracé.

sol.py : Ce code permet de capturer les valeurs d'un capteur de sol (FC-28) pendant 10 secondes, d'enregistrer ces valeurs dans une base de données SQLite, puis de tracer un graphique des 10 dernières valeurs enregistrées

c. Capteur DHT11

Les fichiers :

- ✓ test2.php
- ✓ temperature_humdite.py
- ✓ temperature_humidite.db

test2.php : ce code HTML est une page web qui affiche les dernières valeurs enregistrées par un capteur de température et d'humidité. Il se connecte à une base de données SQLite pour récupérer ces données et les affiche dans un tableau. Il crée également un graphique interactif en utilisant la bibliothèque JavaScript Plotly.

temperature_humdite.py : Ce code collecte 10 lectures de température et d'humidité à partir d'un capteur DHT11, stocke ces valeurs dans une base de données SQLite, puis extrait les 10 dernières valeurs de température et d'humidité pour tracer des graphiques avec matplotlib.

d. Pompe

Les fichiers :

- ✓ pompe.py
- ✓ pompe.php

pompe.php : Ce code est une page HTML qui affiche un formulaire pour contrôler une pompe. Le formulaire contient un champ pour entrer la durée d'activation de la pompe en secondes, ainsi que deux boutons pour démarrer et arrêter la pompe. La page utilise également du code PHP pour afficher un message indiquant si la pompe est active ou éteinte, en fonction de l'action sélectionnée dans le formulaire.

pompe.py : Ce code Python utilise la bibliothèque serial pour communiquer avec un dispositif de pompage via un port série. Il demande à l'utilisateur la durée d'activation de la pompe, envoie la commande à la pompe, attend la réponse de la pompe, affiche l'état de la pompe et demande à l'utilisateur s'il souhaite continuer. Le code se termine en fermant le port série.

e. Capteur ultrasonic

Les fichiers :

- ✓ water_sensor.py
- ✓ water_sensor.php

water_sensor.php : Ce code est une page HTML qui affiche une barre de niveau d'eau avec la hauteur déterminée par un script Python. Le script Python est exécuté en utilisant la fonction "exec" de PHP et le résultat est stocké dans une variable. La hauteur de l'eau est calculée en multipliant le résultat par 20, et la page affiche également la hauteur en centimètres.

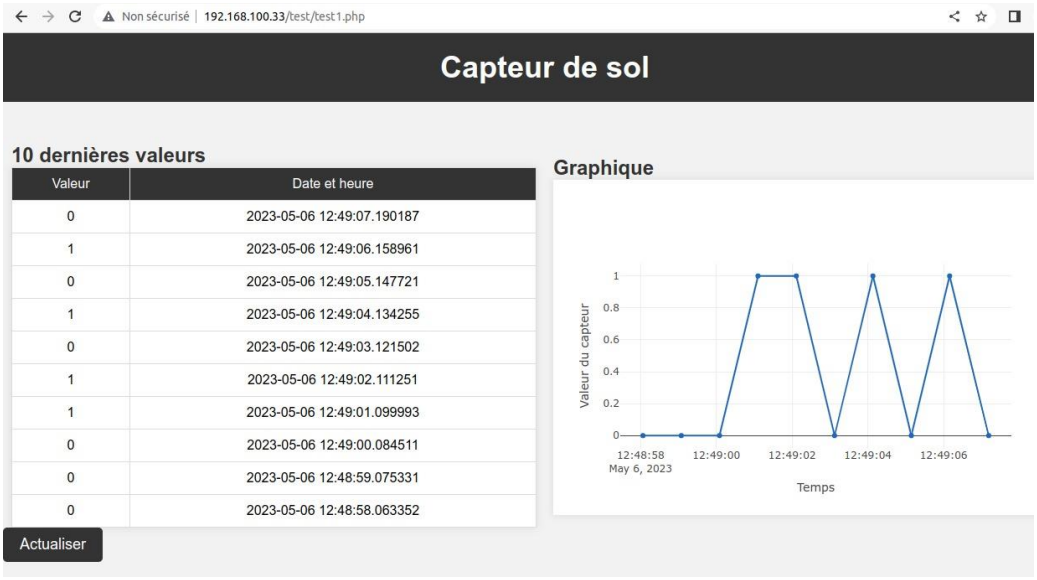
water_sensor.py : Il mesure la distance à l'aide d'un capteur ultrasonique connecté aux broches TRIG et ECHO. La fonction mesure_distance() calcule la distance en centimètres et la boucle principale affiche cette distance toutes les secondes. Le code est encapsulé dans un bloc try-except pour intercepter les interruptions clavier et nettoyer les broches GPIO avant de quitter.

2. Tests unitaires de l'application

a. Test LEDs

b. Test Capteur de sol

```
pi@raspberrypi:~/Wissal/sol $ sqlite3 sol.db
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite> SELECT * FROM capteur_sol;
1|0|2023-05-06 12:47:14.527165
2|0|2023-05-06 12:47:15.534904
3|0|2023-05-06 12:47:16.546488
4|1|2023-05-06 12:47:17.558776
5|0|2023-05-06 12:47:18.568571
6|0|2023-05-06 12:47:19.580527
7|1|2023-05-06 12:47:20.617808
8|0|2023-05-06 12:47:21.629444
9|1|2023-05-06 12:47:22.649171
10|0|2023-05-06 12:47:23.663346
11|0|2023-05-06 12:48:58.063352
12|0|2023-05-06 12:48:59.075331
13|0|2023-05-06 12:49:00.084511
14|1|2023-05-06 12:49:01.099993
15|1|2023-05-06 12:49:02.111251
16|0|2023-05-06 12:49:03.121502
17|1|2023-05-06 12:49:04.134255
18|0|2023-05-06 12:49:05.147721
19|1|2023-05-06 12:49:06.158961
20|0|2023-05-06 12:49:07.190187
sqlite>
```



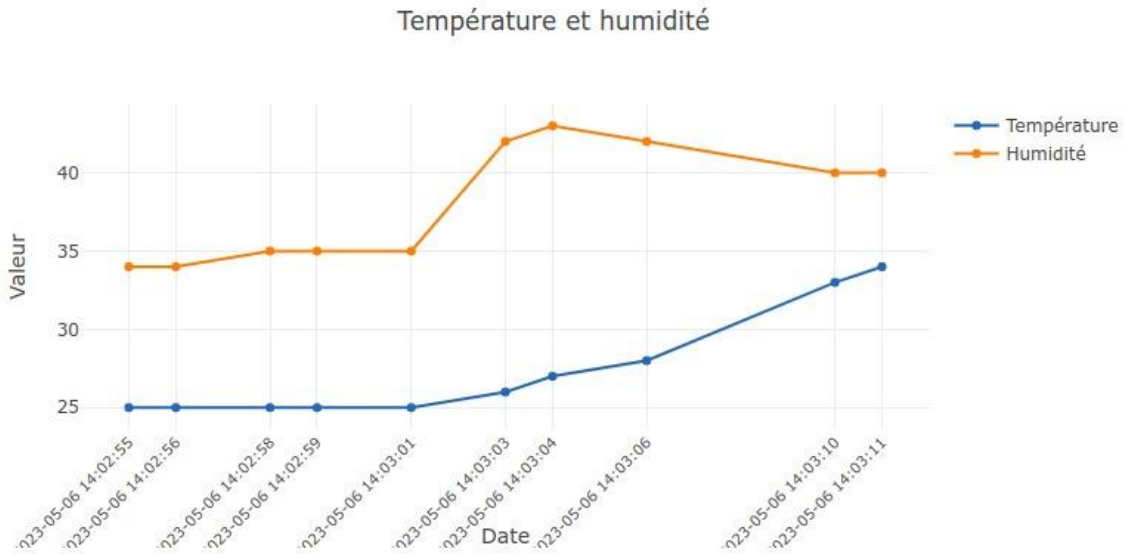
c. Capteur de temperature

```
pi@raspberrypi:/www $ sqlite3 temperature_humidite.db
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite> SELECT * FROM temperature_humidite;
1|24.0|39.0|2023-05-07 12:08:08
2|24.0|39.0|2023-05-07 12:08:09
3|24.0|39.0|2023-05-07 12:08:09
4|24.0|39.0|2023-05-07 12:08:10
5|24.0|39.0|2023-05-07 12:08:10
6|24.0|39.0|2023-05-07 12:08:11
7|25.0|40.0|2023-05-07 12:08:11
8|25.0|41.0|2023-05-07 12:08:12
9|25.0|41.0|2023-05-07 12:08:12
10|26.0|42.0|2023-05-07 12:08:13
11|32.0|29.0|2023-05-07 12:10:03
12|32.0|30.0|2023-05-07 12:10:04
13|32.0|30.0|2023-05-07 12:10:04
14|32.0|30.0|2023-05-07 12:10:05
15|32.0|30.0|2023-05-07 12:10:05
16|32.0|30.0|2023-05-07 12:10:06
17|32.0|30.0|2023-05-07 12:10:06
18|32.0|30.0|2023-05-07 12:10:07
19|32.0|30.0|2023-05-07 12:10:07
20|32.0|31.0|2023-05-07 12:10:08
21|24.0|37.0|2023-05-07 13:12:16
22|24.0|37.0|2023-05-07 13:12:16
23|24.0|37.0|2023-05-07 13:12:17
24|24.0|38.0|2023-05-07 13:12:17
25|24.0|38.0|2023-05-07 13:12:18
26|24.0|38.0|2023-05-07 13:12:18
27|25.0|39.0|2023-05-07 13:12:19
28|25.0|39.0|2023-05-07 13:12:19
29|26.0|40.0|2023-05-07 13:12:20
30|27.0|42.0|2023-05-07 13:12:20
sqlite>
```

Capteur de température et d'humidité

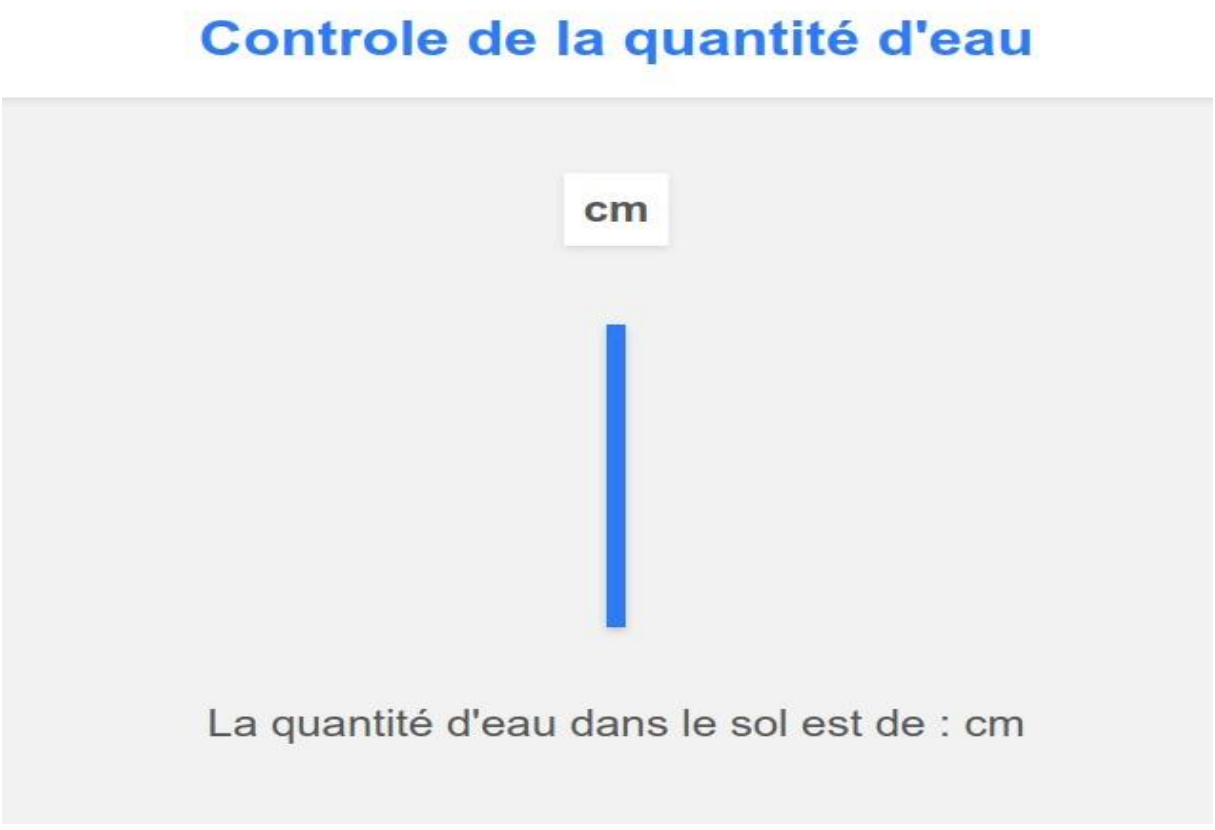
Dernières valeurs enregistrées

Date	Température	Humidité
2023-05-06 14:03:11	34.0 °C	40.0 %
2023-05-06 14:03:10	33.0 °C	40.0 %
2023-05-06 14:03:06	28.0 °C	42.0 %
2023-05-06 14:03:04	27.0 °C	43.0 %
2023-05-06 14:03:03	26.0 °C	42.0 %
2023-05-06 14:03:01	25.0 °C	35.0 %
2023-05-06 14:02:59	25.0 °C	35.0 %
2023-05-06 14:02:58	25.0 °C	35.0 %
2023-05-06 14:02:56	25.0 °C	34.0 %
2023-05-06 14:02:55	25.0 °C	34.0 %



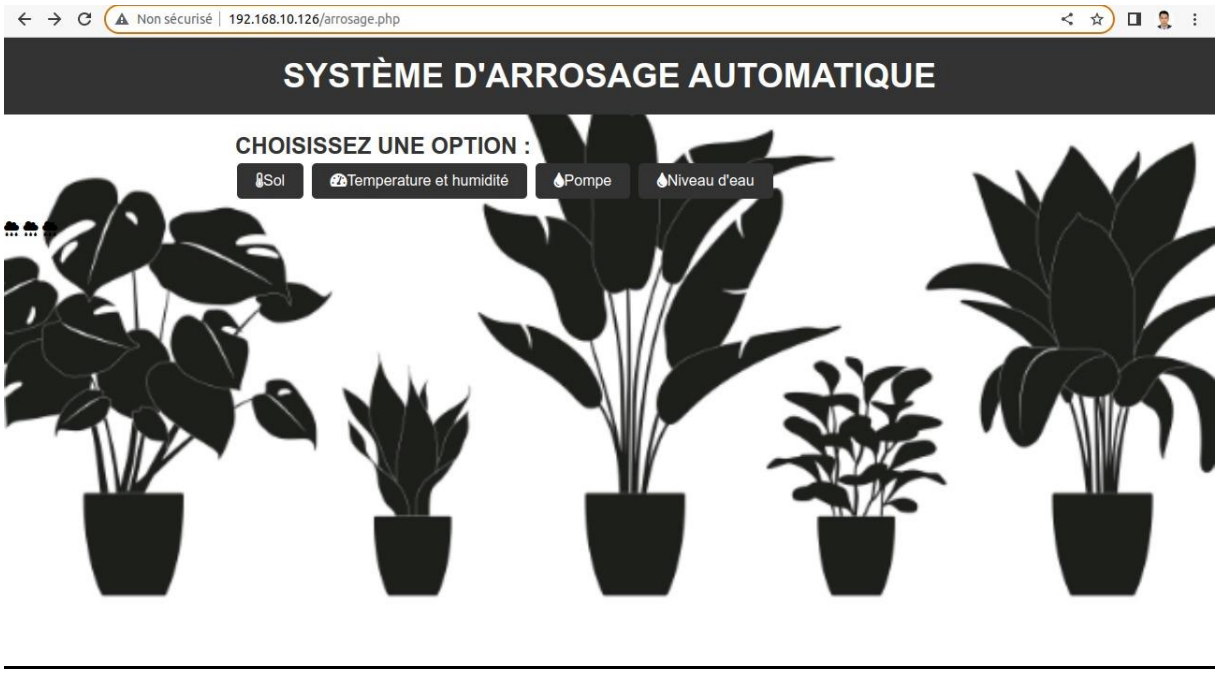
d. Pompe

e. Capteur Ultrasonic



3. Test de régression de l’interface :

```
pi@raspberrypi:/www $ nano login.php
pi@raspberrypi:/www $ sqlite3 sol1.db
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite> SELECT * FROM capteur_sol;
1|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:35:38.756508
2|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:35:39.764303
3|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:40.775624
4|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:41.787149
5|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:42.797142
6|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:43.809622
7|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:44.829172
8|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:35:45.844141
9|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:35:46.856022
10|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:47.869967
11|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:58.419865
12|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:35:59.432595
13|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:00.445297
14|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 15:36:01.455876
15|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:02.469400
16|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:03.496954
17|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:04.511500
18|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:05.523868
19|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:06.543940
20|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 15:36:07.558778
21|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:01:51.370841
22|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:01:52.383357
23|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 16:01:53.393618
24|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:01:54.410468
25|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 16:01:55.422665
26|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:01:56.436111
27|0|0|La LED est allumée|0|La pompe est désactivée|2023-05-08 16:01:57.448153
28|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:01:58.462338
29|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:01:59.476997
30|1|1|La LED est éteinte|1|La pompe est activée pendant une minute|2023-05-08 16:02:00.491107
sqlite>
```



Conclusion

En conclusion, le développement de cette application client-serveur pour le système d’arrosage automatique a été une expérience stimulante mais enrichissante. En utilisant une combinaison de PHP, Python et C, nous avons pu créer une interface Web qui permet le contrôle en temps réel de divers actionneurs, ainsi que l'acquisition et l'affichage des données des capteurs.

Notre utilisation de la carte Raspberry Pi comme serveur nous a permis d'incorporer des fichiers pilotes et d'émuler les capteurs et actionneurs, démontrant ainsi notre compréhension des systèmes embarqués. De plus, le stockage des actions historiques dans une base de données fournit un outil utile pour l'analyse et l'optimisation du système.

Dans l'ensemble, ce projet nous a fourni une expérience précieuse dans le développement Web, la gestion de bases de données et les systèmes embarqués, et nous sommes fiers d'avoir produit une application fonctionnelle et pratique qui peut potentiellement être appliquée dans des scénarios réels.