Institut National de Radioélectricité et de Cinématographie

Enseignement technique secondaire de qualification

Accès aux études supérieures

Avenue Jupiter, 188

1190 Forest

Serre connecté

Projet de fin d'étude

Projet personnel de Karakus Ravza 6TQJ

Pour l'obtention du certificat de qualification

Technicien(ne) en informatique

Année scolaire : 2025-2026

Remerciement

Avant de débuter la présentation de ce travail de fin d'études, il me tient à cœur d'adresser mes sincères remerciements aux personnes qui ont contribué, de près ou de loin, à sa réalisation.

Je tiens tout d'abord à remercier chaleureusement mon professeur principal et professeur du cours de projet, Monsieur Kajjoua, pour ses conseils avisés, son encadrement tout au long du projet ainsi que ses idées qui m'ont permis d'améliorer mon travail et de lui donner un caractère professionnel.

Je tiens à remercier sincèrement Ali Sebbahi en 5J qui m'a été d'une grande aide tout au long de ce projet. Son soutien constant et ses conseils m'ont permis de surmonter les nombreux obstacles rencontrés. Lorsque j'étais sur le point d'abandonner, il m'a convaincu de persévérer, m'a redirigé et m'a remis les idées en place pour que je puisse continuer.

Je remercie également Monsieur Janah, qui m'a été d'une aide précieuse dans les moments où je me sentais perdu. Son accompagnement et sa disponibilité m'ont permis de surmonter les difficultés techniques rencontrées au fil du projet.

Enfin, je ne saurais oublier mes parents, qui m'ont soutenu tant moralement que financièrement tout au long de cette année. Leur présence et leurs encouragements m'ont aidé à garder le cap dans les moments de doute et à avancer avec confiance jusqu'à l'aboutissement de ce travail.

Table des matières

Introduction 3
1. Analyse des besoins 4
1.2 Description du problème 4
1.3 Public cible / utilisateur 5
1.4 Cahier des charges 5
1.4.2 Matériels utilisés 5
Raspberry Pi 4 16
DHT11 17
Clé USB 17
Buzzer 18
Capteur d'humidité de sol 18
Tuyau polychlorure 19
Comet Pompe Submersible 20
Câbles 20
ADS1115 21
Breadboard 22
1.4.3 Logiciels utilisés 23
Raspberry Pi OS 23
Python 24
Visual Studio Code (VS Code) 25
Thonny IDE 25
GitHub 25
1.5 Base de données 27
Mariadb 27
Firebase 28
Que ce que Firebase ? 28
1.6 Fonctionnalité attendue 28
1.6.2 Mesurer les conditions de la serre 29
1.6.3 Contrôle automatique des équipements 29
1.6.4 Permettre à l'utilisateur de surveiller et contrôler 29
1.6.5 Alerter en cas de problème 29

2. Conception du projet 30
2.2 Architecture du projet.....31
2.3 Topologie.....31
2.4 Schéma mécanique 32
2.5 Maquette de la serre.....33
3. Déploiement et mise en ligne : Mon domaine karakusravza.be 34
4. Explication de mon code principal (serre_main.py) 35
5. Base de données 35
1. La gestion des comptes (Firebase Authentication) 35
2. Le stockage des données (Cloud Firestore).....36
6. Architecture des fichiers 37
7. Tests et validations.....37
Problèmes rencontrés.....37
1. Configuration de l'accès à distance.....37
2. Installer les librairies et faire fonctionner les capteurs 37
3. Choisir entre Flask et Tkinter pour mon application 37
Je ne savais pas quelle interface utiliser pour mon application : 38
4. Le Raspberry Pi qui s'éteignait tout seul 38
5. Les retards pour acheter les composants 38
6. Problème pour me connecter à MariaDB 38
8. Fonctionnement.....39
8.1 Introduction 39
8.2 Les Capteurs.....39
8.3 Le Traitement par le Raspberry Pi 39
8.4 Les Actions Automatiques.....39
8.5 Le Stockage des Données.....39
8.6 L'Interface Utilisateur.....40
8.7 Conclusion.....40
Réalisation.....41
1ère partie : Démarrer le système 41
2ème partie : Interface utilisateur 43
Page de Connexion.....43
Tableau de Bord Principal (Données des capteurs).....44

Conclusion..44
Bibliographie 45
Annexes...47

Introduction

Aujourd'hui, s'occuper des plantes et d'une serre demande beaucoup d'attention et de temps. Cette tâche est particulièrement ardue pour les personnes âgées, malades ou très occupées. Il faut constamment vérifier la température, l'humidité, l'arrosage, voire la ventilation. Une simple erreur ou un oubli peut endommager les plantes ou provoquer des maladies, entraînant de graves conséquences.

J'ai pensé à tous les efforts que fournissent les jardiniers, les agriculteurs, etc. J'ai donc réfléchi à un moyen de faciliter leur tâche et me suis posé la question suivante :

Comment créer une serre entièrement automatisée, tout en permettant à l'utilisateur de la contrôler à distance ?

La réponse à cette question a été de construire une serre connectée dont l'élément principal est un Raspberry Pi 4 doté de 8 Go de RAM. Celui-ci est connecté à une application permettant de contrôler et surveiller facilement la serre, ainsi que d'accéder à toutes les informations nécessaires. J'ai conçu ce projet pour que le système puisse gérer plusieurs paramètres simultanément : l'arrosage automatique, la température, l'humidité et la ventilation.

L'objectif de ce projet est de proposer un système facile à gérer, efficace et minimisant au maximum les interventions, notamment pour les personnes dans le besoin.

J'ai imaginé un système où l'arrosage s'active automatiquement lorsque l'humidité du sol est faible. J'ai également intégré un système de ventilation automatique qui s'active lorsque l'humidité et la température dépassent certains seuils.

1. Analyse des besoins

1.2 Description du problème

De nombreuses personnes âgées souhaitent continuer à jardiner ou cultiver leur potager, mais des contraintes physiques, le manque de temps ou les imprévus (sécheresse, gel, maladies) rendent cette activité difficile à gérer. Parallèlement, les agriculteurs et jardiniers amateurs recherchent des solutions pour optimiser leur travail et éviter les pertes liées à une mauvaise gestion de l'arrosage, de la température ou de l'humidité. Notre objectif est donc de concevoir un système automatisé et connecté qui simplifie la gestion des cultures, réduire les risques d'échec et permette un contrôle, afin de rendre l'agriculture plus accessible et efficace.

1.3 Public cible / utilisateur

Dans un contexte pareil, les utilisateurs finaux sont principalement :

- Les jardiniers (amateurs ou expérimentés) souhaitant optimiser leur temps et réduire les risques d'échec des cultures.
- Les agriculteurs (petites ou moyennes exploitations) cherchant à automatiser certaines tâches et à sécuriser leurs récoltes.
- Les passionnés de culture (légumes, plantes aromatiques, fleurs, etc.) désirant un suivi précis et un contrôle à distance.

1.4 Cahier des charges

1.4.2 Matériels utilisés

Voici l'équipement que j'ai utilisé afin de construire mon projet, ainsi que le coût total de celui-ci.

Equipement	Prix	Quantité	Source
DHT11	2,24€	1	Aliexpress
Clé USB	9.95€	1	Aliexpress
Buzzer	2,18€	1	Aliexpress
Capteur de pluie	1,70€	1	Aliexpress
Capteur d'humidité de sol	1,55€	1	Aliexpress
Tuyau polychlorure	0,87€	1	Aliexpress
Comet Pompe Submersible	1.10€	1	Aliexpress
Câble mâle vers mâle, femelle vers femelle, mâle vers femelle	4,59€	1	Aliexpress
Raspberry Pi 4 8 Go	175€	1	Kiwi-electronics
Transistors	7,09€	200	Amazon
ADS1115	1,56€	1	Aliexpress
Breadboard	3,14€	1	Aliexpress
Serre intérieur	34,99€	1	IKEA
Écran lcd	8,955€	1	Amazon
Le prix total consacré à mon projet est de :	248,89€.		

Raspberry Pi 4

Le Raspberry Pi 4 est un nano-ordinateur de la taille d'une carte de crédit, conçu par la fondation Raspberry Pi. Malgré sa petite taille, il est capable d'effectuer des tâches complexes comme un ordinateur classique. Il a été créé à la base pour aider les jeunes à apprendre la programmation et l'électronique, mais aujourd'hui il est utilisé dans de nombreux projets professionnels et scolaires à travers le monde.

Ce qui rend le Raspberry Pi 4 spécial, c'est qu'il regroupe dans un format très compact tout ce dont on a besoin pour faire fonctionner un système informatique complet. Il possède un processeur, de la mémoire RAM, une connexion réseau, du Wi-Fi, du Bluetooth et des ports GPIO qui permettent de le connecter à des composants électroniques externes comme des capteurs ou des moteurs.

Ses caractéristiques principales :

- 8 Go de mémoire RAM (modèle utilisé dans ce projet)
- Connexion Wi-Fi 2.4 GHz et 5 GHz intégrée
- Bluetooth 5.0 intégré
- 2 ports USB 3.0 et 2 ports USB 2.0
- 2 ports Micro-HDMI pour connecter un écran
- 40 broches GPIO pour connecter des composants électroniques

Pourquoi j'ai choisi le Raspberry Pi 4 ?

J’ai choisi le Raspberry Pi 4 comme le cerveau de mon projet. Le Raspberry Pi a pour rôle de collecter les données (humidité, température), de contrôler les actionneurs (pompe, ventilation), et de stocker et analyser les données localement.

J’ai également fait ce choix car j’ai déjà quelques connaissances sur cet outil : j’ai suivi un an de cours de projet sur le Raspberry Pi et je me suis dit que cela pourrait être une bonne opportunité de me perfectionner et d’apprendre davantage sur cet appareil.

DHT11

Le DHT22 est un capteur électronique qui permet de mesurer deux choses en même temps : la température et l'humidité de l'air. C'est l'un des capteurs les plus utilisés dans les projets électroniques car il est simple à connecter, peu coûteux et donne des résultats précis.

Ses caractéristiques principales :

- Mesure de température entre -40°C et +80°C
- Précision de température de ±0,5°C
- Mesure d'humidité entre 0% et 100%
- Temps de réponse d'environ 2 secondes entre chaque mesure
- Alimentation entre 3,3V et 5V
- Communication via un signal numérique directement lisible par le Raspberry Pi

Pourquoi j'ai utilisé le DHT11 ?

Il est essentiel de contrôler la température et l'humidité dans la serre pour assurer une bonne croissance des plantes. Le capteur DHT11 me permet de surveiller ces deux paramètres en temps réel. J'ai conçu mon code de sorte que le système de ventilation se déclenche automatiquement dès que la température ou l'humidité dépasse un seuil prédéfini.

Clé USB

Pour installer le système d'exploitation sur la carte SD du Raspberry Pi, j'ai utilisé une clé USB. La clé USB a servi à transférer les fichiers nécessaires à cette installation.

Buzzer

Le buzzer est un petit composant électronique qui produit un son lorsqu'il reçoit un signal électrique. Il est très souvent utilisé dans les projets électroniques pour émettre des alertes ou des signaux sonores. Il existe deux types de buzzers : le buzzer actif qui produit un son fixe dès qu'il est alimenté, et le buzzer passif qui nécessite un signal variable pour produire différents sons.

Ses caractéristiques principales :

- Alimentation entre 3,3V et 5V
- Compatible directement avec les broches GPIO du Raspberry Pi
- Émission sonore instantanée des réceptions d'un signal
- Petit et facile à intégrer dans un projet

Pourquoi j'ai utilisé un buzzer ?

J’utilise un buzzer dans ma serre pour un système d’alerte sonore. Lorsqu’un problème survient, comme une température trop élevée ou un niveau d’eau insuffisant dans le réservoir, le Raspberry Pi envoie un signal au buzzer, qui émet alors un son pour alerter l’utilisateur. Cela permet d’intervenir rapidement avant que les plantes ne subissent des dommages ou que la pompe ne fonctionne à vide.

Capteur d'humidité de sol

Le capteur d'humidité de sol est un composant électronique qui mesure le taux d'humidité présent dans la terre. Il est composé de deux parties : une sonde avec deux électrodes métalliques que l'on plante directement dans le sol, et un module électronique qui analyse le signal et l'envoie au Raspberry Pi.

Ses caractéristiques principales :

- Alimentation entre 3,3V et 5V
- Sortie analogique et numérique
- Sensibilité réglable grâce à un potentiomètre sur le module
- Compatible avec le Raspberry Pi via l'ADS1115

Pourquoi j'ai utilisé un capteur d'humidité de sol ?

Ce capteur, c’est un peu comme le "super pouvoir" de ma serre connectée ! Il vérifie tout le temps si la terre est assez humide pour mes plantes. Si la terre devient trop sèche, mon Raspberry Pi le détecte et allume automatiquement la pompe pour arroser. Comme ça, je ne gaspille pas d’eau et mes plantes ont toujours juste ce qu’il leur faut, ni trop, ni pas assez.

Tuyau polychlorure

Le tuyau polychlorure, plus connu sous le nom de tuyau PVC, est un tuyau en plastique solide et résistant utilisé pour transporter des liquides. Il est très courant dans les systèmes d'irrigation.

Ses caractéristiques principales :

- Résistant à l'humidité et à la corrosion
- Léger et facile à couper et à assembler
- Disponible en différents diamètres
- Étanche et durable dans le temps

Pourquoi j'ai utilisé un tuyau PVC ?

Ce tuyau établit la connexion entre la pompe submersible et les plantes de la serre. Il garantit un transfert d'eau efficace et sans fuite à chaque cycle d'arrosage automatique.

Comet Pompe Submersible

La pompe submersible est une pompe conçue pour fonctionner immergée dans l'eau. Elle aspire l'eau du réservoir et la pousse à travers le tuyau PVC jusqu'aux plantes.

Ses caractéristiques principales :

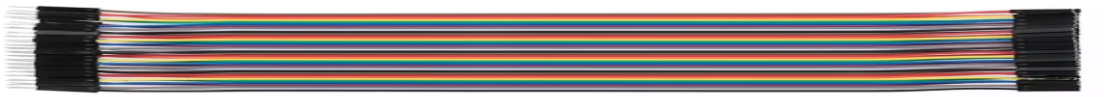
- Fonctionne totalement immergée dans l'eau
- Alimentation électrique contrôlable via un relay
- Débit suffisant pour irriguer les plantes de la serre
- Silencieuse et compacte

Pourquoi j'ai utilisé une pompe submersible ?

Cette pompe submersible est installée dans le réservoir d'eau de la serre. Elle est activée automatiquement par le Raspberry Pi via un relais lorsque le capteur d'humidité du sol détecte un assèchement du substrat, assurant ainsi l'arrosage des plantes.

Câbles

Les câbles sont des fils électriques qui permettent de relier tous les composants entre eux. Dans un projet électronique, il en existe plusieurs types selon les besoins.



Les types de câbles utilisés :

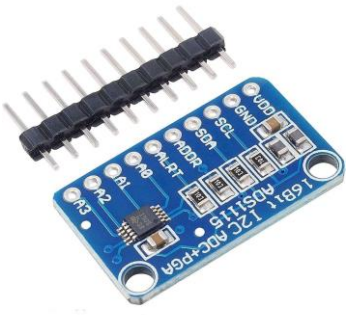
- Mâle vers mâle : pour relier deux composants avec des broches
- Femelle vers femelle : pour connecter deux modules avec des connecteurs
- Mâle vers femelle : pour combiner les deux types de connexions

Pourquoi j'ai utilisé ces câbles ?

Ces câbles m'ont servi à relier tous les éléments de ma serre (Raspberry Pi, capteurs, pompe, etc.) entre eux, sans avoir besoin de souder. C'est pratique parce que ça permet de tout modifier ou réparer facilement si besoin.

ADS1115

L'ADS1115 est un convertisseur analogique-numérique. Son rôle est de transformer des signaux électriques analogiques en données numériques que le Raspberry Pi peut lire et interpréter.



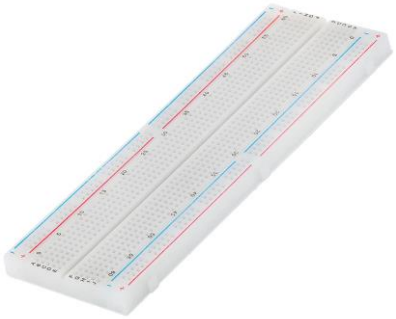
Ses caractéristiques principales :

- 4 canaux d'entrée analogique indépendants
- Précision de conversion sur 16 bits
- Communication via le protocole I2C avec le Raspberry Pi
- Alimentation entre 2V et 5,5V

Pourquoi j'ai utilisé l'ADS1115 ?

Le Raspberry Pi, contrairement à l'Arduino, ne dispose pas de convertisseur analogique-numérique intégré pour lire directement les signaux analogiques. Or, certains capteurs comme celui d'humidité du sol ou de luminosité produisent ce type de signaux. L'ADS1115 intervient alors comme un module intermédiaire en convertissant ces signaux analogiques en données numériques exploitables par le Raspberry Pi.

Breadboard



Une breadboard, aussi appelée plaque de prototypage, est une plaque plastique percée de trous reliés entre eux par des conducteurs internes. Elle permet de monter des circuits électroniques rapidement sans avoir besoin de souder les composants.

Ses caractéristiques principales :

- Connexion sans soudure, donc réversible et modifiable
- Rangées de trous reliées entre elles en interne
- Compatible avec tous les composants électroniques standards
- Réutilisable à l'infini

Pourquoi j'ai utilisé une breadboard ?

La breadboard (plaque d'essai) a été utilisée pour assembler et tester rapidement l'ensemble des composants électroniques durant la phase de développement. Ce système de connexion sans soudure a permis de modifier les branchements en temps réel, facilitant ainsi le dépannage et réduisant les risques d'endommager les éléments du circuit.

1.4.3 Logiciels utilisés

Nom	Description	Prix

Raspberry pi OS	Le système qui fait tourner tout le reste	0 €
Python	Le langage qui écrit la logique de ma serre	0 €
Visual studio code	Mon éditeur principal depuis ton PC	0 €
Thonny IDE	Mon outil de test rapide sur le Raspberry Pi	0 €
GitHub	GitHub est une plateforme d'hébergement et de collaboration pour projets informatique.	0 €

Raspberry Pi OS

Définition : C'est le système d'exploitation du Raspberry Pi — comme Windows sur un PC, mais conçu spécialement pour ce mini-ordinateur. C'est lui qui fait tourner tous les programmes et gère le matériel connecté.

Caractéristiques principales

- Basé sur Linux (Debian), gratuit et open source
- Léger, optimisé pour le Raspberry Pi
- Inclut un bureau graphique, un terminal, et des outils de développement
- Gère nativement les broches GPIO (les connecteurs physiques du Raspberry Pi)

Rôle dans mon projet : C'est la base de tout. Sans lui, le Raspberry Pi ne démarre pas. Il permet à Python de communiquer avec tes capteurs (DHT11, LDR, ADS1115…), de contrôler le relais (pompe, ventilateur), d'afficher l'interface Tkinter sur l'écran, et de faire tourner Flask pour ton site web.

Python

Définition : C'est un langage de programmation — c'est-à-dire une façon d'écrire des instructions que l'ordinateur comprend et exécute. Python est réputé pour être lisible, presque comme de l'anglais courant.

Caractéristiques principales

- Syntaxe simple et claire, idéale pour débuter
- Très utilisé en électronique, automatisation et data
- Des milliers de bibliothèques disponibles (extensions prêtes à l'emploi)
- Compatible nativement avec le Raspberry Pi

Rôle dans mon projet : C'est le langage de tout mon code. C'est Python qui lit les capteurs, décide d'activer la pompe ou le ventilateur, écrit dans la base de données MariaDB, affiche les données sur le LCD et l'interface Tkinter, et expose l'API Flask pour ton site web.

Visual Studio Code (VS Code)

Définition : C'est un éditeur de code — un logiciel pour écrire, organiser et déboguer tes programmes.

Caractéristiques principales

- Gratuit, léger et très populaire
- Coloration syntaxique (le code est coloré pour mieux le lire)
- Extensions pour Python, Git, connexion SSH à distance…
- Permet de se connecter directement au Raspberry Pi depuis ton PC

Rôle dans mon projet : Je l'utilise sur ton PC (Windows/Mac) pour écrire et organiser tout le code Python de ma serre à distance, via SSH sur le Raspberry Pi. C'est mon atelier de développement principal pour les fichiers Flask, la logique des capteurs, et le site web HTML/CSS/JS.

Thonny IDE

Définition : C'est aussi un éditeur de code Python, mais pensé pour les débutants. Il est installé directement sur le Raspberry Pi et permet de tester du code simplement, ligne par ligne.

Caractéristiques principales

- Interface très simple, sans fioritures
- Débogueur visuel : on voit exactement où le programme s'arrête et pourquoi
- Exécute le code directement sur le Raspberry Pi
- Idéal pour tester rapidement un capteur ou un actionneur

Rôle dans mon projet C'est un outil de test rapide sur le terrain. Quand je branche un nouveau capteur (DHT11, ADS1115…), j'ouvres Thonny directement sur le Raspberry Pi pour vérifier que tout fonctionne avant d'intégrer le code dans le projet complet.

GitHub

Définition : GitHub est une plateforme de gestion de développement logiciel basée sur le cloud. Elle utilise le système de contrôle de version **Git** pour permettre aux développeurs de stocker leur code, d'en suivre l'historique des modifications et de collaborer avec d'autres personnes, partout dans le monde.

C'est, en résumé, le "réseau social" et l'espace de travail principal des développeurs.

Caractéristiques principales

Définition : GitHub est la plus grande plateforme web au monde dédié à l'hébergement et à la gestion de projets informatiques. Elle s'appuie sur le système Git, un outil de "contrôle de version" qui enregistre chaque modification apportée à un fichier. Concrètement, GitHub agit

comme un immense espace de travail collaboratif où les développeurs stockent leur code dans des dépôts (repositories) pour le sauvegarder, le partager et l'améliorer à plusieurs.

Caractéristique : GitHub, c'est comme un site internet géant où les gens qui créent des applications rangent leur travail. C'est un peu un "Cloud" qui se souvient de tous les changements qu'on fait sur un fichier.

On y crée des dossiers pour partager son code et travailler à plusieurs sans tout casser. On peut aussi s'entraider pour corriger des erreurs ou proposer des idées. En résumé, c'est le réseau social des informaticiens pour construire des projets ensemble et montrer ce qu'ils savent faire.

1.5 Base de données

Nom	Description
MariaDB	Base de données open source et compatible MySQL.
Firebase	Firebase est un service Google qui gère l'authentification des utilisateurs.

Mariadb

Que ce que Mariadb ?

MariaDB est un système de gestion de base de données relationnelle open source, créé par les fondateurs originaux de MySQL. Très répandu dans le monde du développement, il est intégré par défaut dans la majorité des distributions Linux et proposé dans la plupart des services cloud. Il se distingue par ses performances élevées, sa stabilité et son engagement total envers l'open source.

Rôle dans mon projet :

Dans mon projet de serre connectée, la base de données MariaDB a pour rôle de stocker toutes les mesures relevées par les capteurs tels que la température, l'humidité et la luminosité. Elle permet d'enregistrer les données en temps réel provenant du Raspberry Pi et de conserver un historique complet des conditions dans la serre. Ces données sont ensuite récupérées pour être affichées sous forme de graphiques sur l'interface web, et permettent également de déclencher des alertes lorsque les valeurs mesurées dépassent les seuils définis.

Firebase

Que ce que Firebase ?

Firebase est une plateforme de développement d'applications proposée par Google, qui fournit un ensemble de services cloud comme l'authentification, la base de données en temps réel, le stockage et l'hébergement, permettant aux développeurs de créer des applications web et mobiles plus rapidement.

Rôle dans mon projet

Dans mon projet de surveillance des plantes, j'utilise Firebase pour gérer l'authentification des utilisateurs. Concrètement, lorsqu'un utilisateur souhaite accéder à l'interface de gestion, j'utilise Firebase pour vérifier son identité en gérant l'inscription et la connexion via un email et un mot de passe. Cela me permet de sécuriser l'accès à mon application et de m'assurer que seules les personnes autorisées peuvent consulter ou modifier les données des plantes.

1.6 Fonctionnalité attendue

Pour une serre complète et efficace, le système doit remplir plusieurs fonctions. Voici ce que le système doit faire, organisé en 4 grandes parties.

1.6.2 Mesurer les conditions de la serre

i. Mesurer la température et l'humidité de l'air

Ça sert à savoir si l'air est trop chaud, trop froid, trop sec ou trop humide pour les plantes. Le capteur DHT11 est capables de mesurer tous ces fonctions et envoie les données au Raspberry Pi

ii. Mesurer l'humidité du sol

Ça va servir à vérifier si la terre est assez humide ou trop sèche. Le capteur d'humidité de sol envoie cette information au Raspberry Pi.

iii. Détecter le niveau d'eau

Savoir si le réservoir d'eau plein ou presque vide pour éviter que la pompe fonctionne à vide.

1.6.3 Contrôle automatique des équipements

i. Arrosage automatique

La pompe s'allume toute seule quand la terre est trop sèche. Le système évite d'arroser trop souvent pour ne pas gaspiller et éviter l'excès d'eau pour les plantes.

ii. Ventilation automatique

Le ventilateur se met en marche si l'air devient trop humide (Raspberry reçoit les données du capteur DHT11). Il s'allume progressivement pour ne pas abîmer les plantes.

iii. Gérer les seuils critiques

Le système surveille en permanence les valeurs et agit immédiatement si quelque chose ne va pas.

1.6.4 Permettre à l'utilisateur de surveiller et contrôler

i. Afficher les données en temps réel

L'utilisateur pourrait voir sur un écran ou un téléphone les valeurs de température, humidité, etc., sous forme de tableau.

ii. Contrôle manuel

L'utilisateur pourrait allumer ou éteindre la pompe, le ventilateur à la main depuis l'interface.

1.6.5 Alerter en cas de problème

i. Notification visuelle

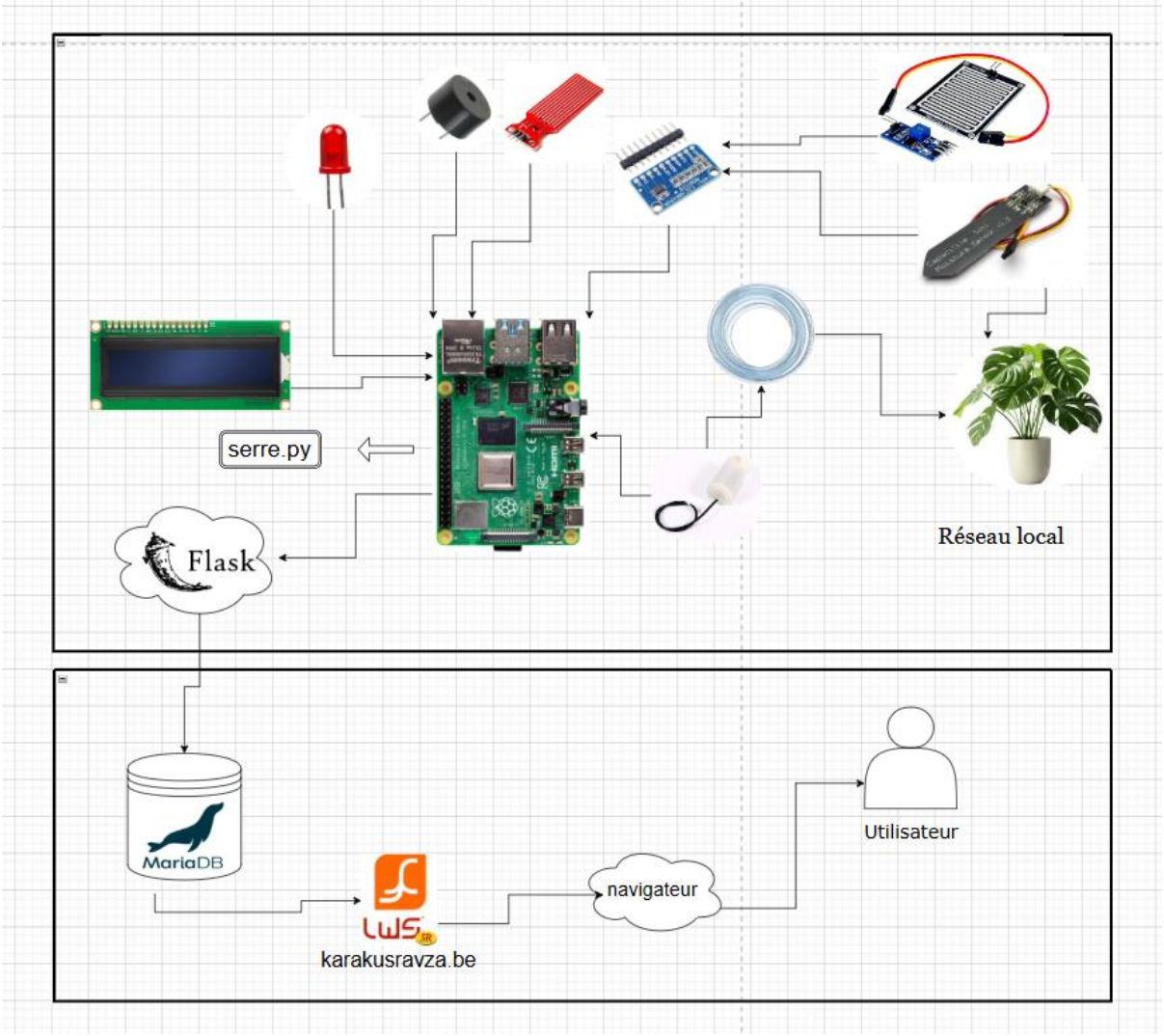
Une LED rouge s'allume et un écran affiche un message si quelque chose ne va pas.

ii. Alertes sonores

Un buzzer sonne si la température dépasse un certain seuil, si le réservoir est vide, ou si un capteur ne fonctionne pas.

2. Conception du projet

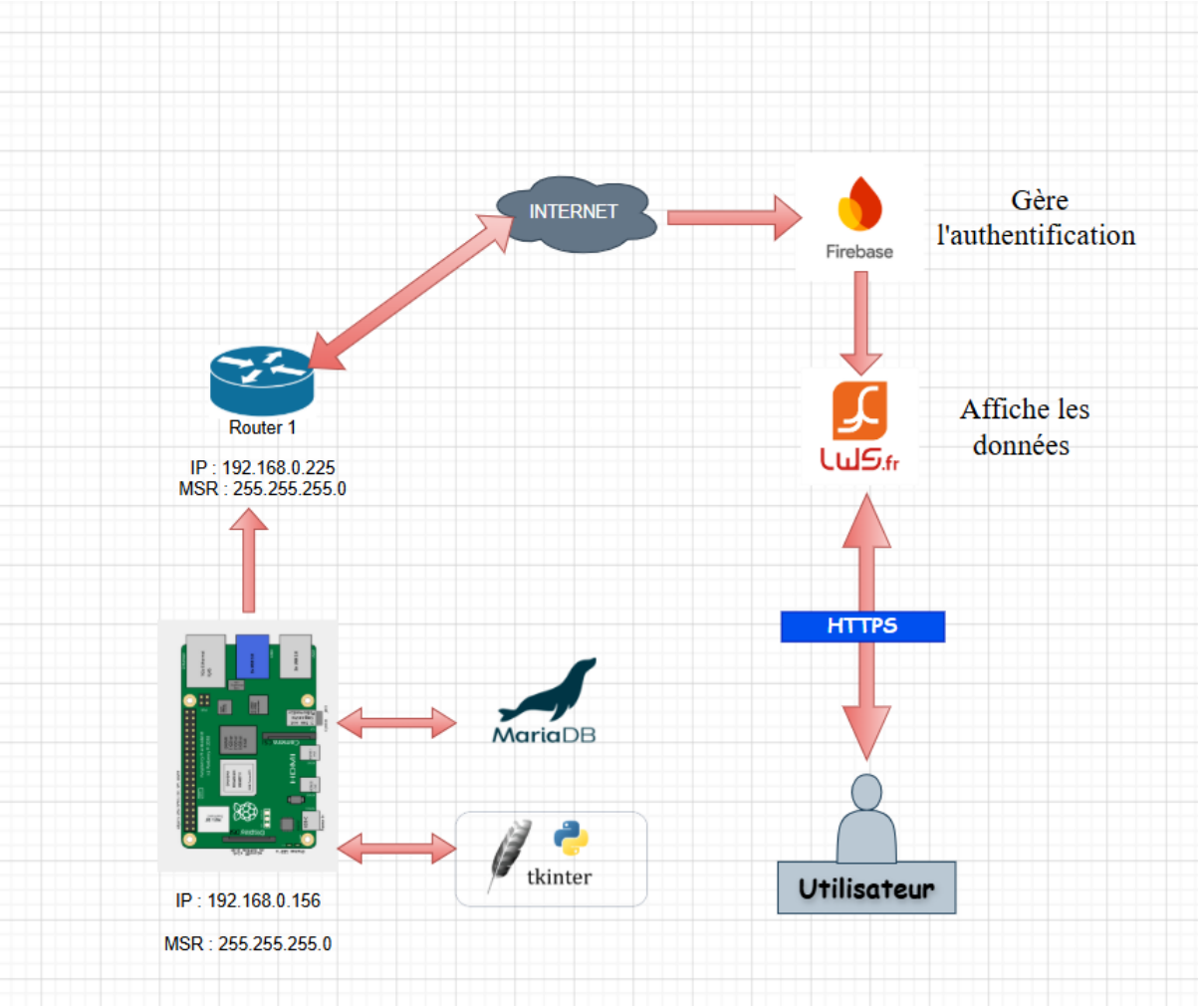
2.2 Architecture du projet



Explication de mon projet

Comme vous pouvez le constater, tous mes capteurs sont reliés directement au Raspberry Pi, à l'exception du capteur d'humidité du sol et du capteur de pluie. Ces deux capteurs passent par le module ADS1115, car le Raspberry Pi ne possède pas d'entrées analogiques. L'ADS1115 se charge donc de la conversion analogique → numérique. J'ai un fichier serre.py qui contient le code principal de lecture des capteurs. Les valeurs sont ensuite envoyées vers l'application Flask, qui les stocke dans la base de données MariaDB. Ces données sont accessibles depuis mon domaine web, permettant à l'utilisateur de surveiller sa serre non seulement via l'application Flask, mais également depuis n'importe où sur Internet.

2.3 Topologie

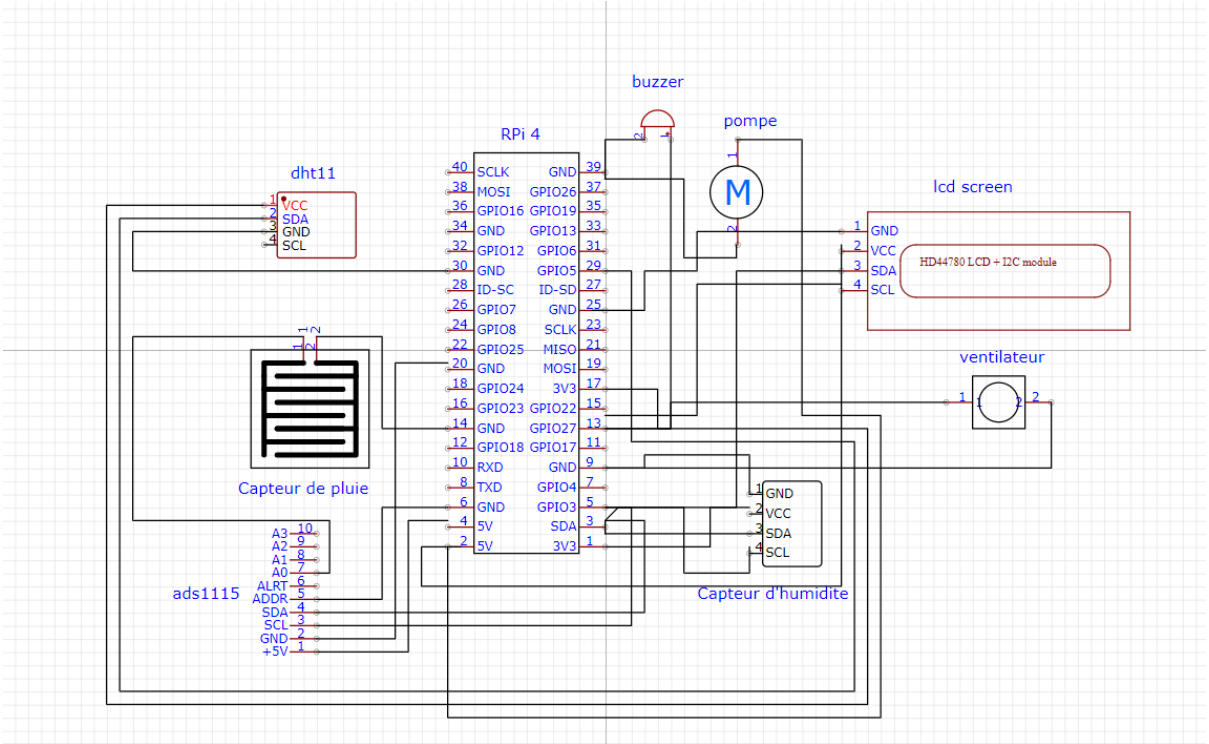


La topologie montre comment les différents éléments du projet sont connectés entre eux.

Le Raspberry Pi est connecté au routeur, ce qui lui permet d'envoyer les données vers Internet. Il communique aussi avec MariaDB pour stocker les données en local, et avec l'application Tkinter pour les afficher.

Les données passent ensuite par Internet pour arriver sur Firebase, qui gère la connexion des utilisateurs. Une fois connecté, l'utilisateur peut voir les données sur le site web hébergé sur LWS, via une connexion sécurisée HTTPS.

2.4 Schéma mécanique



Ce schéma montre un système automatique pour arroser des plantes. Au centre, on trouve un petit ordinateur appelé Raspberry Pi 4, c'est lui qui gère tout.

Autour de lui, plusieurs capteurs lui envoient des informations. Le DHT11 mesure la température et l'humidité de l'air. Le capteur de pluie détecte s'il pleut dehors. Le capteur d'humidité regarde si la terre est sèche ou mouillée. Comme certains capteurs ont besoin d'aide pour communiquer avec le Raspberry Pi, une pièce appelée ADS1115 s'occupe de cette tâche.

Quand le Raspberry Pi reçoit toutes ces informations, il prend des décisions. Si la terre est trop sèche et qu'il ne pleut pas, il allume la pompe pour arroser les plantes. Il peut aussi activer le ventilateur pour faire bouger l'air.

L'écran LCD affiche les données comme la température ou l'humidité, pour que la personne qui regarde puisse savoir ce qui se passe. Le buzzer lui, fait du bruit pour prévenir si quelque chose ne fonctionne pas bien.

Toutes ces pièces sont connectées entre elles par des fils et travaillent ensemble pour que les plantes reçoivent de l'eau au bon moment, sans intervention humaine.

2.5 Maquette de la serre

Pour mon projet de serre connectée j'ai seulement eu besoin d'une serre que j'ai acheté à IKEA. Voici la serre utilisée dans le cadre de ce projet :

Les composants sont installés à l'intérieur de la serre de la façon suivante :

- Plantes 2/3
- Le DHT22 est fixé à l'intérieur pour mesurer l'air
- Le capteur d'humidité du sol est planté directement dans la terre
- Le capteur de pluie est placé sur le toit de la serre
- La pompe est reliée à un réservoir d'eau extérieur

3. Déploiement et mise en ligne : Mon domaine karakusravza.be

J’ai acheté mon nom de domaine sur LWS, un hébergeur français spécialisé dans la vente de noms de domaine et l’hébergement web. Ce site propose des solutions simples et accessibles, idéales pour les projets comme le mien.

J’ai fait ce choix pour deux raisons principales.

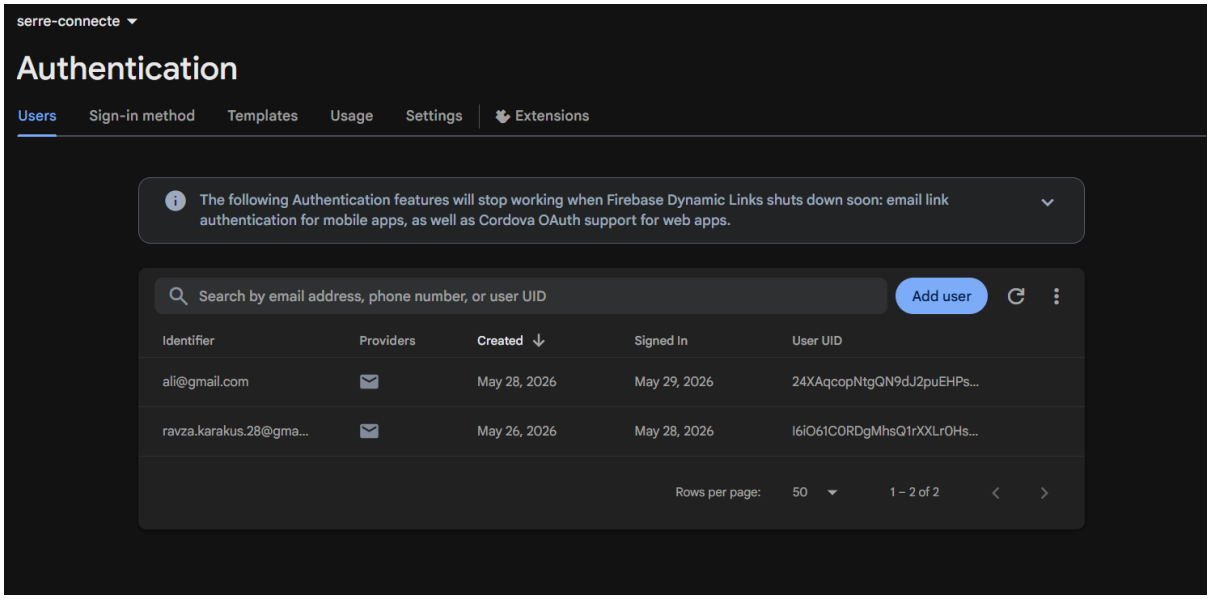
D’abord, les professeurs nous ont demandé d’utiliser un support en ligne clair et accessible pour présenter notre travail. Cela permet aux jurys et aux enseignants de consulter facilement mon projet à tout moment.

Ensuite, avoir un nom de domaine personnalisé rend le projet plus professionnel. Cela montre que j’ai pris le temps de soigner la présentation, ce qui peut faire une bonne impression sur le jury.

4. Base de données

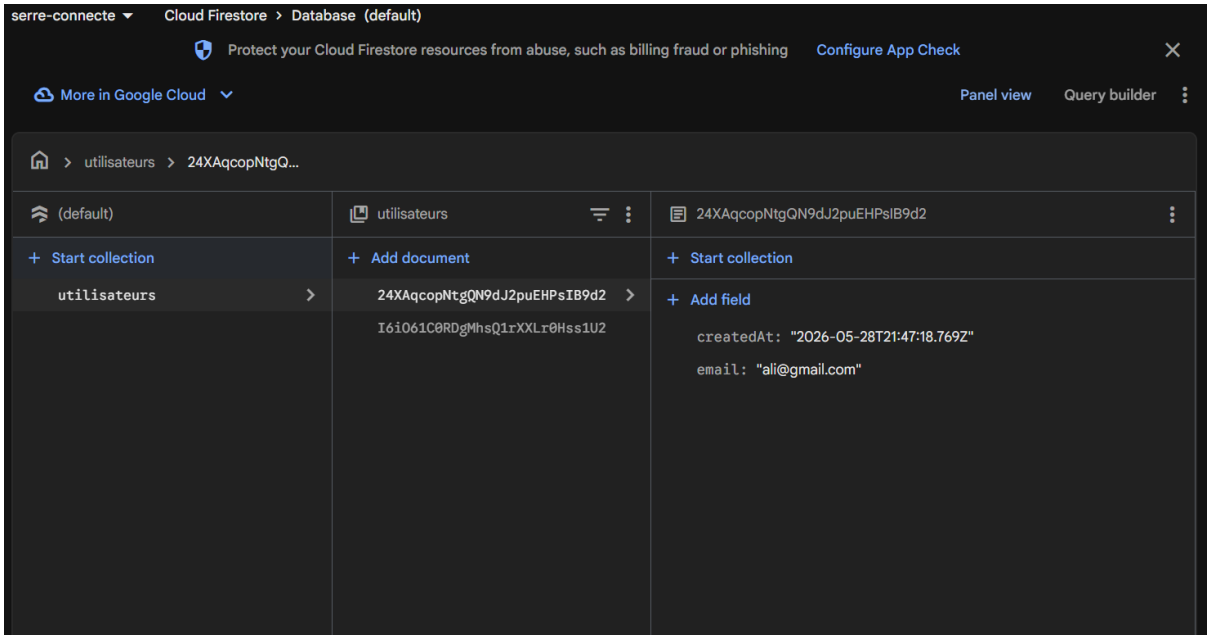
1. La gestion des comptes (Firebase Authentication)

Cette image montre le système qui gère les inscriptions et les connexions. C'est ici que l'on peut voir la liste des utilisateurs inscrits. Le tableau affiche leur adresse e-mail, la date de leur inscription, et génère un numéro d'identification unique pour chaque personne (le "User UID"). Cela permet de gérer les membres de l'application de manière sécurisée.



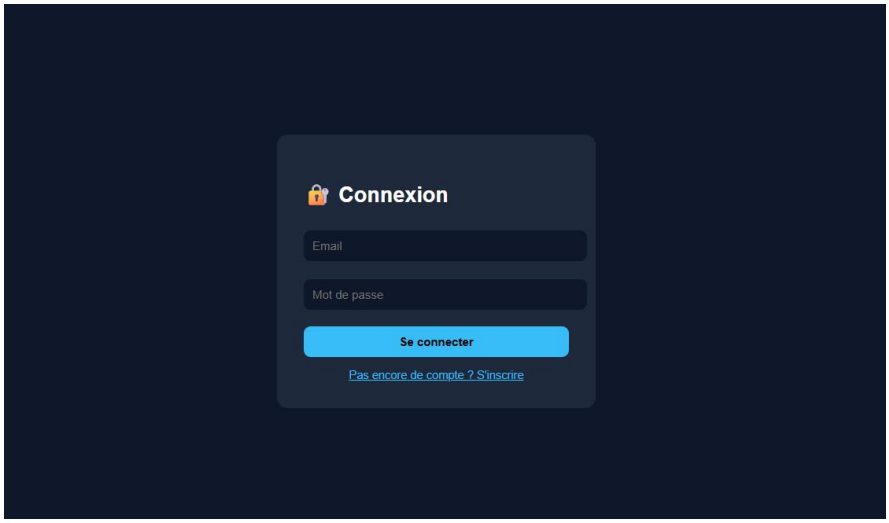
Lorsqu'un utilisateur s'inscrit ou se connecte sur l'application, le processus se déroule en deux étapes automatiques :

1. **La validation de la connexion :** L'utilisateur entre son adresse e-mail et son mot de passe



sur l'application. Le système d'authentification vérifie ces informations, valide la connexion et crée un accès sécurisé. Il attribue alors à la personne un identifiant unique (le User UID).

2. **L'enregistrement dans la base de données :** Une fois la connexion ou l'inscription validée, l'application va automatiquement sauvegarder l'adresse e-mail de l'utilisateur dans la base de données. Dans la section "utilisateurs", un nouveau dossier est créé au nom de l'identifiant unique de la personne, et son e-mail y est stocké de manière sécurisée.



2. Le stockage des données (Cloud Firestore)

Cette image montre l'interface de la base de données NoSQL de Firebase, appelée Cloud Firestore. Contrairement à une base de données classique sous forme de tableaux, les informations y sont organisées de manière flexible en “collections“ et “documents”. Sur l'image, on observe la collection d’utilisateurs. À l'intérieur de celle-ci, un document spécifique (souvent lié à l'UID de l'utilisateur) est sélectionné. On peut y voir les champs de données qui lui sont associés, comme son adresse e-mail et sa date de création. C'est cette structure qui permet à l'application de lire et d'écrire des informations de manière rapide et en temps réel.

5. Architecture des fichiers

```
serre_venv/
│
├── ADS1115/
│   └── ads1115.py              # Lecture humidité sol + pluie
│
├── Buzzer/
│   └── buzzer.py               # Gestion buzzer GPIO23
│
├── Capteur de pluie/
│   └── capteur_pluie.py        # Lecture capteur pluie
│
├── DHT11/
│   └── dht11.py                # Température + humidité air
│
├── Humidité du sol/
│   └── humidite_sol.py         # Lecture humidité sol
│
├── LCD/
│   └── lcd.py                  # Affichage LCD
│
├── Pompe/
│   └── pompe.py                # Activation pompe GPIO18
│
├── Ventilateur/
│   └── ventilateur.py          # Activation ventilateur
│
├── serre_main.py               # Lance tout les capteurs
│
├── config.py                   # Paramètres généraux
├── alertes.py                  # Système d'alertes
├── base_donnees.py             # Firebase + MariaDB
│
└── requirements.txt            # Dépendances
```

6. Tests et validations

Problèmes rencontrés

1. Configuration de l'accès à distance

Dès le début du projet, j'ai souhaité mettre en place un accès à distance à mon Raspberry Pi pour gagner en flexibilité, car je ne pouvais pas toujours travailler devant un écran dédié.

Solutions et ajustements :

J'ai d'abord tenté une connexion en SSH, mais j'ai rencontré un problème de configuration lié à ma clé d'authentification. Après avoir réinitialisé cette clé, j'ai temporairement utilisé RPi Connect pour accéder à mon appareil.
Par la suite, je suis passé à RealVNC Viewer, qui s'est avéré plus stable et mieux adapté à mes besoins.

2. Installer les librairies et faire fonctionner les capteurs

J’ai eu beaucoup de mal à installer les librairies nécessaires, surtout pour le capteur ADS1115 et le bus I2C.

Les problèmes que j’ai eus :

Le Raspberry Pi ne reconnaissait pas le capteur et mon utilisateur n’avait pas le droit d’utiliser le bus I2C.

Comment j’ai corrigé ça :

J’ai donné les droits nécessaires pour utiliser le bus I2C et J’ai vérifié que tout était bien branché.

3. Choisir entre Flask et Tkinter pour mon application

Je ne savais pas quelle interface utiliser pour mon application :
Flask : Permettait d’y accéder depuis internet, mais c’était plus compliqué à configurer.

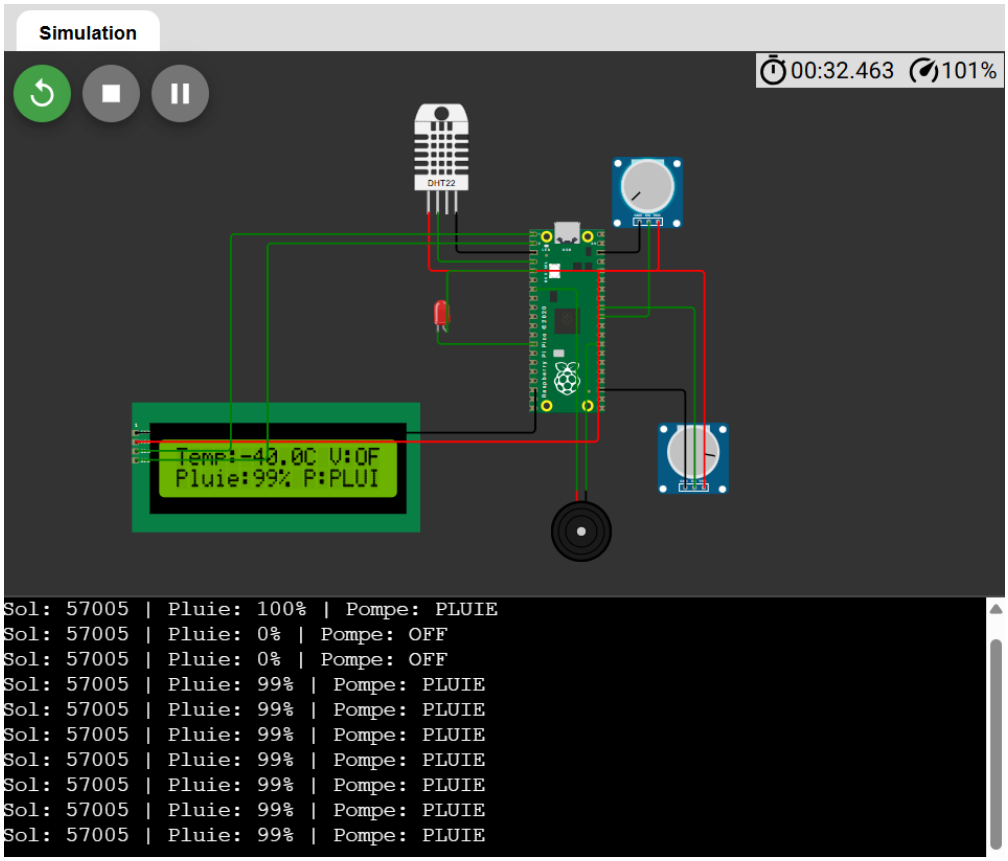
Tkinter : Plus simple, mais seulement pour un usage local.

Mon choix final : J’ai pris Tkinter, car c’était plus facile à mettre en place pour moi.

4. Le Raspberry Pi qui s’éteignait tout seul

Pendant l’installation des librairies, mon Raspberry Pi a planté plusieurs fois. L’écran devenait noir et affichait des erreurs, et après un redémarrage, il restait bloqué sur un message "Mode veille". Je suis allé voir mon professeur de sml, M.Janah, et il m’a expliqué que les fils positif et négatif du breadboard se touchaient, ce qui faisait planter le Raspberry. J’ai aussi dû réinitialiser ma clé USB, mais le message de veille persistait. Pour régler le problème, j’ai vérifié tous les branchements pour éviter les courts-circuits et j’ai changé les réglages pour que le Raspberry ne se mette plus en veille tout seul.

Solutions : Comme j'étais bloqué sur le Raspberry Pi, j'ai continué à travailler sur Wokwi pour tester et faire avancer mon code.



5. Les retards pour acheter les composants

Au début de l'année, j'ai aussi eu des retards pour acheter tous les composants dont j'avais besoin. Ça a ralenti mon travail, mais j'ai fini par tout recevoir.

6. Problème pour me connecter à MariaDB

J'ai eu du mal à me connecter à la base de données MariaDB. C'était un peu embêtant, mais j'ai réussi à régler le problème sans que ça bloque tout le projet.

8. Problème d'écran noir sur le raspberry pi

À la dernière minute, l'écran de mon Raspberry Pi est devenu noir et ne s'allume plus. Je n'ai donc pas pu intégrer dans mon TFE (Travail de Fin d'Études) l'interface graphique de mon application.

7. Fonctionnement

7.1 Introduction

La serre connectée est un système automatisé qui permet de surveiller et de gérer les conditions de croissance des plantes sans intervention humaine constante. Le principe est

simple : des capteurs récoltent des informations, le Raspberry pi les analyse, et des actionneurs réagissent en conséquence.

7.2 Les Capteurs

Tout commence par les capteurs. Le capteur DHT11 est chargé de mesurer en permanence la température et l'humidité de l'air à l'intérieur de la serre. En parallèle, un capteur d'humidité du sol est enfoncé dans la terre pour détecter si les plantes ont besoin d'eau. Enfin, un capteur de niveau d'eau surveille le réservoir pour éviter que la pompe tourne à vide.

7.3 Le Traitement par le Raspberry Pi

Le Raspberry Pi est le cerveau du système. Toutes les quelques secondes, il récupère les valeurs envoyées par les capteurs et les compare à des seuils prédéfinis. Par exemple, si la température dépasse 25°C, il considère que la serre est trop chaude. Si l'humidité du sol descend en dessous d'un certain niveau, il juge que les plantes ont soif.

7.4 Les Actions Automatiques

Une fois l'analyse faite, le Raspberry Pi agit sans attendre l'intervention de l'utilisateur. Si le sol est trop sec, il active la pompe à eau pour arroser les plantes. Si la température ou l'humidité de l'air est trop élevée, il déclenche le ventilateur pour aérer la serre. En cas de situation critique, comme un réservoir vide ou une température dangereuse, le buzzer se met à sonner et la LED rouge s'allume pour alerter l'utilisateur.

7.5 Le Stockage des Données

Toutes les mesures collectées sont enregistrées dans une base de données MariaDB installée sur le Raspberry Pi. Cela permet de garder un historique complet des conditions dans la serre et de pouvoir consulter l'évolution de la température, de l'humidité ou des arrosages sur plusieurs jours ou semaines.

7.6 L'Interface Utilisateur

L'utilisateur dispose de deux façons de surveiller sa serre. En local, une interface graphique développée avec Tkinter s'affiche directement sur un écran connecté au Raspberry Pi. À distance, il peut se connecter depuis n'importe où via le site karakusravza.be, sécurisé par une

authentification Firebase, pour consulter les données en temps réel et contrôler les équipements manuellement si besoin.

7.7 Conclusion

En résumé, la serre connectée fonctionne comme un cycle continu : les capteurs lisent, le Raspberry Pi réfléchit, les actionneurs agissent, les données sont sauvegardées et l'utilisateur peut tout surveiller à distance. C'est un système autonome qui réduit considérablement le travail manuel tout en garantissant de meilleures conditions de croissance pour les plantes.

8. Réalisation

8.1 - 1ère partie : Démarrer le système

```
try:
    while True:  # Boucle infinie, le programme tourne en permanence

        # Lecture temperature et humidite
        try:
            temp = dht.temperature  # On lit la temperature
            hum = dht.humidity      # On lit l'humidite
            temp_last = temp        # On sauvegarde au cas ou ca plante
            hum_last = hum
        except Exception:
            # Si ca plante, on garde les anciennes valeurs
            temp = temp_last
            hum = hum_last

        # --- Lecture des capteurs de pluie et de sol ---
        try:
            valeur_pluie = capteur_pluie.value   # Valeur brute du capteur pluie
            valeur_sol = capteur_sol.value        # Valeur brute du capteur sol

            # On transforme la valeur du sol en pourcentage (0% = sec, 100% = mouille)
            pourcentage_sol = round((1 - valeur_sol / 32767) * 100, 1)

            # Si la valeur est basse, c'est qu'il pleut
            pluie = valeur_pluie < 26500

            # On sauvegarde les valeurs
            sol_last = pourcentage_sol
            pluie_last = pluie
```

```
        except Exception:
            # Si ca plante, on essaie de relancer les capteurs
            try:
                capteur_pluie, capteur_sol = init_ads()
            except Exception:
                pass
            # On garde les anciennes valeurs
            pourcentage_sol = sol_last
            pluie = pluie_last

        # --- Gestion de la pompe ---
        # On allume la pompe si le sol est trop sec (moins de 30%) ET qu'il ne pleut pas
        if pourcentage_sol is not None and pourcentage_sol < 30 and not pluie:
            pompe.value = True
            etat_pompe = "ON"
        else:
            pompe.value = False
            etat_pompe = "OFF"

        # --- Gestion du buzzer ---
        # Si la temperature depasse 28°C, on fait biper le buzzer
        if temp is not None and temp > 28:
            buzzer.duty_cycle = 65535  # Buzzer ON
            time.sleep(0.5)            # On attend 0.5 secondes
            buzzer.duty_cycle = 0      # Buzzer OFF
        else:
            buzzer.duty_cycle = 0      # Pas chaud = pas de bip
```

```
    # --- Affichage dans le terminal ---
    print("=" * 30)
    print(f"Temp    : {temp}C" if temp else "Temp    : --")
    print(f"Humidite: {hum}%" if hum else "Humidite: --")
    print(f"Sol     : {pourcentage_sol}%" if pourcentage_sol else "Sol     : --")
    print(f"Pluie   : {'OUI' if pluie else 'NON'}")
    print(f"Pompe   : {etat_pompe}")
    print("=" * 30)

    # --- Affichage sur l'ecran LCD ---
    lcd.clear()  # On efface l'ecran avant d'ecrire

    if ecran == 0:
        # Page 1 : temperature et humidite
        lcd.write_string(f"T:{temp:.1f}C" if temp else "T:--")
        lcd.cursor_pos = (1, 0)  # On passe a la 2eme ligne
        lcd.write_string(f"Hum:{hum:.1f}%" if hum else "H:--")

    elif ecran == 1:
        # Page 2 : humidite du sol et pluie
        lcd.write_string(f"Sol:{pourcentage_sol}%" if pourcentage_sol else "Sol:--")
        lcd.cursor_pos = (1, 0)
        lcd.write_string("PLUIE!" if pluie else "Pas de pluie")

    elif ecran == 2:
        # Page 3 : etat de la pompe
        lcd.write_string(f"Pompe:{etat_pompe}")
        lcd.cursor_pos = (1, 0)
        lcd.write_string(f"T:{temp:.1f}C S:{pourcentage_sol}%" if temp and pourcentage_sol else "--")

    # On passe a la page suivante
    ecran = (ecran + 1) % 3

    time.sleep(2)
```

Ce code fait tourner le système d'arrosage automatique des plantes en continu.

Au début, il branche tous les composants, l'écran, le capteur de température, la pompe, le buzzer et les capteurs de pluie et de sol.

Ensuite il tourne en boucle sans jamais s'arrêter. A chaque tour de boucle il lit la température et l'humidité de l'air, il lit si le sol est sec ou mouillé en transformant la valeur en pourcentage, et il regarde s'il pleut dehors.

Avec ces informations il prend deux décisions. Si le sol a moins de 30% d'humidité et qu'il ne pleut pas, il allume la pompe pour arroser. Si la température dépasse 28 degrés, il fait biper le buzzer pour prévenir.

Il affiche aussi les informations en même temps, dans le terminal de l'ordinateur et sur l'écran LCD. Sur l'écran il alterne entre trois pages, une pour la température et l'humidité de l'air, une pour l'humidité du sol et la pluie, et une pour l'état de la pompe.

Si un capteur a un problème et ne répond plus, le code garde les dernières valeurs connues pour ne pas planter complètement.

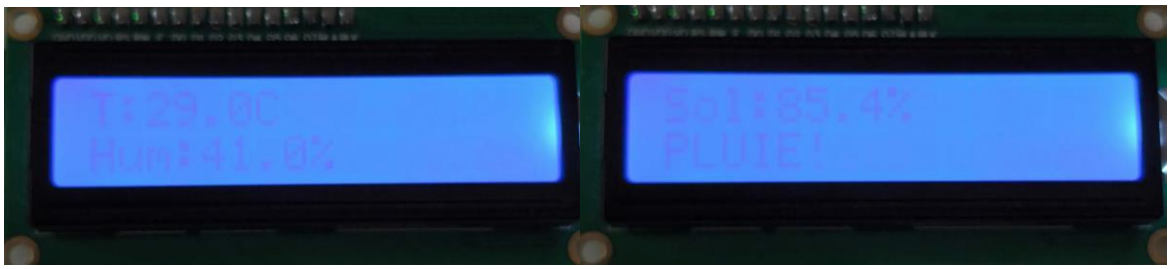
Quand on arrête le programme, il éteint proprement la pompe, le buzzer et l'écran avant de se fermer.

Test :

```
Temp    : 29.0C
Humidite: 41%
Sol     : 6.9%
Pluie   : NON
Pompe   : ON
============================
============================
Temp    : 29.7C
Humidite: 65%
Sol     : 6.9%
Pluie   : NON
Pompe   : ON
============================
============================
```

Lorsque je lance le code (python3 serre_main.py) la pompe s'active, et lorsque la terre est arrosé la pompe se désactive. Les données sont affichées dans le terminal.

L'écran LCD :



On peut voir dans l'écran les données affichées : la température, l'humidité, l'humidité du sol, s'il pleut ou pas et si la pompe est allumée ou éteinte.

8.3 - 2ème partie : Interface utilisateur

Page de Connexion

Explication : La page de connexion est la première fenêtre qui s'affiche au lancement de l'application. L'utilisateur doit entrer son adresse e-mail et son mot de passe pour accéder à la

serre. L'authentification est gérée par Firebase via la bibliothèque Pyrebase. Quand on clique sur "Se connecter", les identifiants sont vérifiés directement sur Firebase. Si c'est correct, on arrive sur le tableau de bord. Si c'est faux, une notification d'erreur s'affiche. Il y a aussi un onglet inscription pour créer un nouveau compte. On peut prouver que ça fonctionne en allant dans la console Firebase dans l'onglet Authentication, on y voit les comptes créés et les dates de connexion.

Tableau de Bord Principal (Données des capteurs)

Explication : C'est l'écran principal après la connexion. Il affiche en temps réel la température, l'humidité de l'air, l'humidité du sol et la détection de pluie, toutes ces valeurs viennent des capteurs connectés au Raspberry Pi. Il y a trois boutons pour contrôler manuellement la pompe, l'humidificateur et le ventilateur. On peut aussi ajouter des plantes (maximum 3) avec des seuils personnalisés, et chaque plante a une représentation visuelle qui change selon son état. Si la température dépasse 30°C, une alerte rouge s'affiche et le buzzer se déclenche. Les données se mettent à jour automatiquement toutes les 3 secondes grâce à la méthode after() de Tkinter sans avoir besoin de recharger quoi que ce soit.

Conclusion

Ce projet m'a permis de créer une serre connectée et automatisée capable de gérer intelligemment l'arrosage, la ventilation et la surveillance des plantes. Le système est accessible via un site web à l'adresse karakusravza.be, permettant un suivi en temps réel de l'environnement de la serre.

Durant le développement, j'ai rencontré plusieurs erreurs techniques importantes (problèmes de connexion à la base de données, mise en veille du Raspberry Pi, retards d'approvisionnement) qui m'ont presque fait abandonner le projet. Heureusement, grâce au soutien constant de mes professeurs et de mes amis, particulièrement Ali Sebbahi en 5J qui m'a convaincu de ne pas abandonner et m'a remis les idées en place, j'ai persévéré jusqu'au bout.

À l'avenir, j'aimerais enrichir le système en ajoutant l'accès à distance via une meilleure application web et mobile, la gestion automatique de la luminosité selon les besoins spécifiques des plantes, et des alertes en temps réel sur le téléphone en cas de problème. Ce projet m'a démontré l'importance de la persévérance face aux défis.

Bibliographie

Rdagger. (n.d.). *Raspberry Pi INA219 Tutorial | Rototron*. https://www.rototron.info/raspberry-pi-ina219-tutorial/

Nikodem Bartnik. (2022, July 13). *Automatic garden watering system* [Video]. YouTube. https://www.youtube.com/watch?v=s-xkdfNeIVw

monraspberry.com. (n.d.). *Jardinage intelligent avec un Raspberry Pi - MONRASPBERRY*. MonRaspberry. https://monraspberry.com/jardinage-intelligent-avec-un-raspberry-pi/

Maker Giovanni. (2025, June 29). *Building a 200€ product with 30€! ESP32 automatic watering (NO WATER TAP needed!)* [Video]. YouTube. https://www.youtube.com/watch?v=uRrnA6kK3FU

DomoticX. (2024, October 1). *DHT11/DHT22 – Raspberry Pi - DomoticX*. https://domoticx.net/docs/dht11dht22-raspberry-pi/

monraspberry.com. (n.d.-b). *Raspberry Pi et IoT : le guide pour comprendre leur rôle dans l'Internet des Objets - MONRASPBERRY*. MonRaspberry. https://monraspberry.com/raspberry-pi-iot-guide/

Solstice84. (2022, April 14). *Arrosage automatique avec un Raspberry Pi et Python* [Video]. YouTube. https://www.youtube.com/watch?v=biYWZP0aRR8

Santos, S., & Santos, S. (2024, January 17). *Raspberry Pi: DHT11/DHT22 Temperature and Humidity (Python) | Random nerd tutorials*. Random Nerd Tutorials. https://randomnerdtutorials.com/raspberry-pi-dht11-dht22-python/

Warin, S. (2018, July 29). *FriendLeaf : la serre connectée grâce à Constellation*. Constellation. https://developer.myconstellation.io/tutorials/friendleaf-la-serre-connectee-grace-a-constellation/

SensorKit - SensorKit. (n.d.). https://sensorkit.joy-it.net/fr/

GitHub · Change is constant. GitHub keeps you ahead. (2026). GitHub. https://github.com/

Annexes

```
# -*- coding: utf-8 -*-

# On importe les outils dont on a besoin
import time # pour faire des pauses dans le programme
import board # connait les noms des broches du rpi
import busio # gere la communication I2C entre le PI et les composants
import digitalio # controle les broches en ON/OFF
import adafruit_dht # lit les donnees du DHT11
import adafruit_ads1x15.ads1115 as ADS # pilote du convertisseur ADS1115
from adafruit_ads1x15.analog_in import AnalogIn # lit la valeur d'un capteur analogique branche sur l'ADS
from RPLCD.i2c import CharLCD # controle l'ecran lcd
from pwmio import PWMOut # envoie un signal rapide pour controler le buzzer

print("Initialisation...")

# On branche l'ecran LCD (adresse 0x27, 16 colonnes, 2 lignes)
lcd = CharLCD('PCF8574', 0x27, cols=16, rows=2)

# On branche le capteur de temperature/humidite sur la broche D17
dht = adafruit_dht.DHT11(board.D17)

# On branche la pompe sur la broche D18 et on dit que c'est une SORTIE
pompe = digitalio.DigitalInOut(board.D18)
pompe.direction = digitalio.Direction.OUTPUT

# On branche le buzzer (sonnerie) sur la broche D27
buzzer = PWMOut(board.D27, frequency=40000, duty_cycle=0)

# Fonction pour demarrer les capteurs de pluie et de sol
def init_ads():
    i2c = busio.I2C(board.SCL, board.SDA)   # On ouvre la connexion I2C
    ads = ADS.ADS1115(i2c)                   # On branche le convertisseur ADS1115
    return AnalogIn(ads, 0), AnalogIn(ads, 1) # On retourne les 2 capteurs
```

```
# On lance les capteurs de pluie (canal 0) et de sol (canal 1)
capteur_pluie, capteur_sol = init_ads()
print("Tout initialise!")

# On cree des variables pour se souvenir de l'ecran actuel et des dernieres valeurs
ecran = 0
temp_last = None
hum_last = None
sol_last = None
pluie_last = False

try:
    while True:  # Boucle infinie, le programme tourne en permanence

        # Lecture temperature et humidite
        try:
            temp = dht.temperature  # On lit la temperature
            hum = dht.humidity      # On lit l'humidite
            temp_last = temp        # On sauvegarde au cas ou ca plante
            hum_last = hum
        except Exception:
            # Si ca plante, on garde les anciennes valeurs
            temp = temp_last
            hum = hum_last
```

```
    # --- Lecture des capteurs de pluie et de sol ---
    try:
        valeur_pluie = capteur_pluie.value   # Valeur brute du capteur pluie
        valeur_sol = capteur_sol.value       # Valeur brute du capteur sol

        # On transforme la valeur du sol en pourcentage (0% = sec, 100% = mouille)
        pourcentage_sol = round((1 - valeur_sol / 32767) * 100, 1)

        # Si la valeur est basse, c'est qu'il pleut
        pluie = valeur_pluie < 26500

        # On sauvegarde les valeurs
        sol_last = pourcentage_sol
        pluie_last = pluie

    except Exception:
        # Si ca plante, on essaie de relancer les capteurs
        try:
            capteur_pluie, capteur_sol = init_ads()
        except Exception:
            pass
        # On garde les anciennes valeurs
        pourcentage_sol = sol_last
        pluie = pluie_last

    # --- Gestion de la pompe ---
    # On allume la pompe si le sol est trop sec (moins de 30%) ET qu'il ne pleut pas
    if pourcentage_sol is not None and pourcentage_sol < 30 and not pluie:
        pompe.value = True
        etat_pompe = "ON"
    else:
        pompe.value = False
        etat_pompe = "OFF"
```

```
    # --- Gestion du buzzer ---
    # Si la temperature depasse 28°C, on fait biper le buzzer
    if temp is not None and temp > 28:
        buzzer.duty_cycle = 65535  # Buzzer ON
        time.sleep(0.5)            # On attend 0.5 secondes
        buzzer.duty_cycle = 0      # Buzzer OFF
    else:
        buzzer.duty_cycle = 0      # Pas chaud = pas de bip

    # --- Affichage dans le terminal ---
    print("=" * 30)
    print(f"Temp    : {temp}C" if temp else "Temp    : --")
    print(f"Humidite: {hum}%" if hum else "Humidite: --")
    print(f"Sol     : {pourcentage_sol}%" if pourcentage_sol else "Sol     : --")
    print(f"Pluie   : {'OUI' if pluie else 'NON'}")
    print(f"Pompe   : {etat_pompe}")
    print("=" * 30)

    # --- Affichage sur l'ecran LCD ---
    lcd.clear()  # On efface l'ecran avant d'ecrire

    if ecran == 0:
        # Page 1 : temperature et humidite
        lcd.write_string(f"T:{temp:.1f}C" if temp else "T:--")
        lcd.cursor_pos = (1, 0)  # On passe a la 2eme ligne
        lcd.write_string(f"Hum:{hum:.1f}%" if hum else "H:--")

    elif ecran == 1:
        # Page 2 : humidite du sol et pluie
        lcd.write_string(f"Sol:{pourcentage_sol}%" if pourcentage_sol else "Sol:--")
        lcd.cursor_pos = (1, 0)
        lcd.write_string("PLUIE!" if pluie else "Pas de pluie")
```

```
        elif ecran == 1:
            # Page 2 : humidite du sol et pluie
            lcd.write_string(f"Sol:{pourcentage_sol}%" if pourcentage_sol else "Sol:--")
            lcd.cursor_pos = (1, 0)
            lcd.write_string("PLUIE!" if pluie else "Pas de pluie")

        elif ecran == 2:
            # Page 3 : etat de la pompe
            lcd.write_string(f"Pompe:{etat_pompe}")
            lcd.cursor_pos = (1, 0)
            lcd.write_string(f"T:{temp:.1f}C S:{pourcentage_sol}%" if temp and pourcentage_sol else "--")

        # On passe a la page suivante
        ecran = (ecran + 1) % 3

        time.sleep(2)

#  Arret propre du programme
finally:
    pompe.value = False       # On eteint la pompe
    pompe.deinit()            # On libere la broche
    buzzer.duty_cycle = 0     # On eteint le buzzer
    lcd.clear()               # On efface l'ecran
    print("Systeme arrete")
```

Ceci est le code pour le contrôle des capteurs

Vous pouvez consulter l'ensemble des fichiers sources de mon projet de fin d'études via le lien suivant vers mon dépôt GitHub : https://github.com/Anzai0/tfe-R