Rapport de Projet d'évaluation

Soft embarqué

Smart Temperature Control

Réalisé par : ILYAS NHASSE sous la supervision de Pr. ANASS MANSOURI

Génie Systèmes Embarqués et Informatique Industrielle.

École Nationale des Sciences Appliquées de Fès.

2022/2023.

Table des matières :

Introduction Générale :

Le contrôle de température ambiante est un enjeu crucial dans de nombreuses applications où le confort des personnes est en jeu, tels que les bâtiments, les véhicules, les équipements électroniques, les laboratoires et bien d'autres. Un système de contrôle de température efficace et fiable peut contribuer de manière significative à la réduction des coûts énergétiques, à l'optimisation de la performance et à la durabilité des équipements.

Cependant, la mise en place d'un tel système nécessite une expertise approfondie dans le développement de logiciels embarqués, qui est souvent une tâche complexe et coûteuse.

Ainsi, ce projet vise à relever ce défi en proposant une solution innovante basée sur l'architecture Client-Serveur et en adoptant une méthode de développement en cycle en V. Nous allons analyser en détail le logiciel embarqué et développer notre compétence dans ce domaine en mettant l'accent sur la qualité, la sécurité et la maintenance du système.

En implémentant cette solution, nous espérons non seulement résoudre les problèmes actuels de contrôle de température ambiante, mais aussi offrir des opportunités pour améliorer les performances et la flexibilité des systèmes futurs en introduisant des systèmes intelligents.

Chapitre 1 : Contexte Général du Projet *(le répertoire /Specs)*

Contexte :

l'application vise à créer un environnement ambiant pour l'utilisateur en apprenant comment l'utilisateur agit lorsque certaines conditions de température à certains intervalles de temps, la solution est donc de développer un système qui permet à l'utilisateur de modifier la vitesse d'un moteur pour contrôler la température et Grâce à la base de données que nous accumulons, nous pouvons exporter un modèle d'intelligence artificielle qui aide à automatiser le contrôle, ce qui améliore considérablement l'expérience de l'utilisateur.

Pour aboutir à développer ce système avec efficacité et avec une grande performance il fallait suivre une méthode de développement qui donne valeur à des métriques non fonctionnelles de système. Il fallait encore avoir un cahier des charges détaillé. les exigences fonctionnelles du système doivent être précises et pointues pour être capable de bien tester notre système.

Cahier de Charges :

Les objectifs de ce projet sont les suivants :

- Analyser les besoins en matière de contrôle de température ambiante dans différents contextes et applications.
- Développer un système de contrôle de température ambiante basé sur l'architecture client-serveur.
- Adopter une méthode de développement en cycle en V pour garantir la qualité, la sécurité et la maintenance du système.
- Introduire des systèmes intelligents pour améliorer les performances et la flexibilité des systèmes futurs.
- Fournir des pistes concrètes pour résoudre les problèmes actuels de contrôle de température ambiante.
- Réduire les coûts énergétiques, optimiser la performance et la durabilité des équipements.

Le cahier des charges doit également inclure les exigences en termes de fonctionnalités, de performances, de fiabilité, de sécurité et de convivialité de l'interface utilisateur.

L’ensemble des exigences du système sont définies dans le document : spécification de projet” sous le répertoire nommé “Exigences”.

Méthodologie :

La méthodologie adoptée pour ce projet est le cycle en V, une méthode de développement de logiciels qui permet de garantir la qualité et la fiabilité du système en mettant l'accent sur la vérification et la validation à chaque étape du processus.

Le projet se divise en plusieurs phases, à commencer par l'analyse des besoins et la rédaction du fichier Exigences, qui permettra de définir les spécifications du système et les objectifs à atteindre. Le document contient l'ensemble des des exigences fonctionnelles, non fonctionnelles matérielles et logicielles.

Cette phase sera suivie par la phase de conception, au cours de laquelle l'architecture client-serveur sera mise en place et les fonctionnalités du système seront détaillées.

La phase de développement viendra ensuite, où les différentes fonctionnalités seront programmées, les drivers et les applications soit serveur coté ou bien client coté. et les tests unitaires seront réalisés pour s'assurer de la qualité du code. En parallèle, la phase de tests d'intégration et de validation sera menée pour garantir que le système fonctionne comme prévu.

La phase de déploiement permettra d'installer le système sur la plateforme cible et de réaliser les derniers tests de validation. L'ensemble de ces étapes sera réalisé en suivant des processus stricts de gestion de projet, avec un suivi régulier de l'avancement et des indicateurs de performance clés pour s'assurer que les objectifs sont atteints dans les délais impartis et avec la qualité requise.

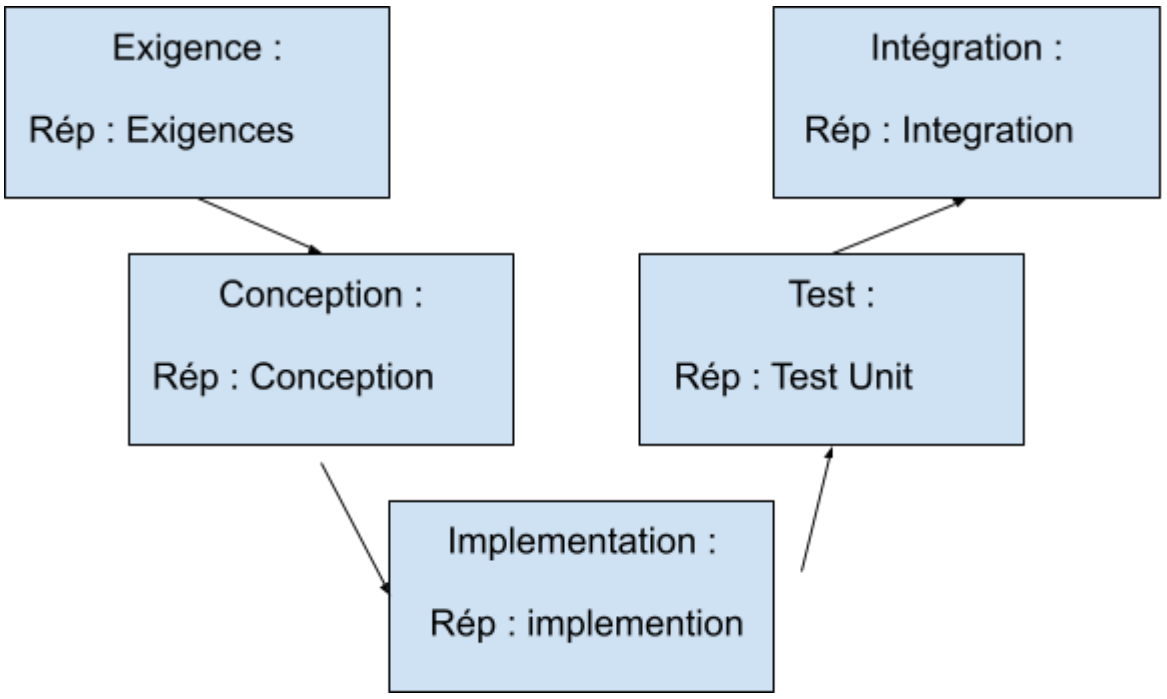

Figure 1 : processus de développement

l'organisation des fichiers sources et documents sera comme suit :

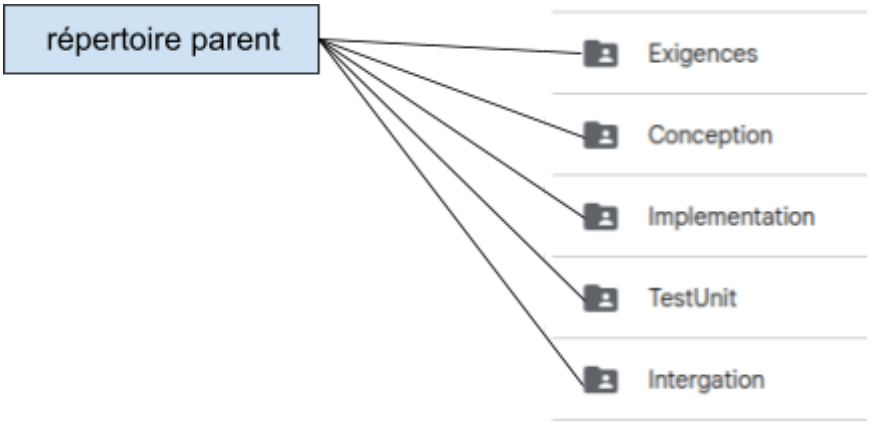

Figure 2 : Organisation fichier

Chapitre 2 : Conception et Modélisation du projet *(le répertoire /Conception)*

Environnement technique du projet :

Le développement de notre projet a été réalisé en utilisant un ensemble d'outils et de technologies adaptés pour garantir la qualité, la sécurité et la fiabilité de notre solution.

Nous avons utilisé les langages de programmation C et Python avec l'environnement de développement intégré (IDE) VS Code pour le développement logiciel. Le langage C a été utilisé pour la programmation du microcontrôleur et le développement des logiciels embarqués. Ce langage est largement utilisé pour le développement de systèmes embarqués, offre une bonne efficacité en termes de performance et de mémoire, et permet une programmation de bas niveau. tandis que Python a été utilisé pour la programmation de scripts et d'outils complémentaires. Nous avons également utilisé le Raspberry Pi 4 comme plateforme de développement pour notre projet.

Pour la partie matérielle, nous avons utilisé un capteur de température DS18B pour mesurer la température ambiante avec précision, ainsi qu'un moteur à courant continu (DC) pour le contrôle de l'équipement de chauffage ou de refroidissement.

En ce qui concerne l'architecture logicielle, nous avons opté pour une architecture client-serveur pour notre système de contrôle de température ambiante. Cette architecture permet de séparer la logique applicative et les fonctions d'interface utilisateur. Le serveur se charge du traitement des données de température et du contrôle des équipements de chauffage ou de refroidissement, tandis que le client permet aux utilisateurs de surveiller et de contrôler le système de contrôle de température à distance. Pour la communication entre le client et le serveur,.

En résumé, notre environnement technique comprend les langages de programmation C et Python, l'IDE VS Code, le Raspberry Pi 4, un capteur de température DS18B, un moteur à courant continu (DC), une architecture client-serveur

Nous utilisons la version 2021 de Raspbian Lite et l'algorithme Random Forest Classifier pour la partie intelligence artificielle de projet. Raspbian Lite est une version légère du système d'exploitation Raspbian spécialement conçue pour les projets Raspberry Pi qui ne nécessitent pas d'interface graphique utilisateur. L'utilisation de Raspbian Lite permet de maximiser les performances du Raspberry Pi et d'optimiser l'utilisation des ressources disponibles.

L'algorithme Random Forest Classifier est une méthode d'apprentissage automatique (machine learning) largement utilisée pour la classification de données. Il s'agit d'un algorithme supervisé qui utilise un ensemble d'arbres de décision pour effectuer la classification. L'utilisation de l'algorithme Random Forest Classifier permet de traiter

efficacement les données collectées par les capteurs et d'obtenir des résultats précis et fiables pour la classification des action user.

Architecture générale du projet :

L'architecture générale de notre solution comprend une partie client, une partie serveur, une base de données et une partie d'intelligence artificielle.

La partie client consiste en une application web pour les utilisateurs finaux qui leur permet de se connecter et de gérer leurs systèmes domestiques connectés. L'application permet également aux utilisateurs de visualiser les données de capter par le DS18B en temps de connexion et de contrôler les niveaux de vitesse de moteur ventilateur.

La partie serveur est constituée de 4 interfaces une pour l'authentification (loginpanel.php) cette interface nous assure de la sécurité. La deuxième interface est une interface d'affichage des valeurs capter et de contrôler l'actionneur; moteur DC (ctrpanel.php), ainsi qu'une interface pour l'affichage de la base de données. La page concernant la prédiction faite par le modèle d'intelligence artificielle va être développée seule pour avoir une réponse plus ou moins rapide.

La base de données stocke les informations relatives aux utilisateurs, et aux actions de l'utilisateur. La base de données est hébergée sur le serveur et est accessible pour l'utilisateur connecté.

La partie d'intelligence artificielle est utilisée pour prédire l'action d'utilisateur futurs concernant la vitesse de moteur, ce qui permet de mieux planifier et gérer l'utilisation de l'actionneur. L'algorithme de machine learning Random Forest Classifier est utilisé pour prédire la variable en fonction des données historiques d'acquisition de données et les valeurs de température au même point.

Voici un schéma qui résume l'architecture générale de notre solution :

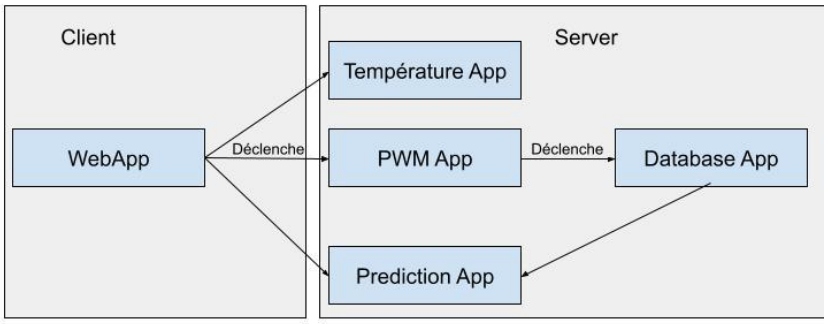

Figure 3 : Architecture Client-Serveur

Conception détaillée du projet :

L'architecture du projet se compose de plusieurs composants interconnectés pour assurer le fonctionnement du système. Le système est composé d'un capteur de température DS18B20, d'un Raspberry Pi 4 Model B, d'un module PWM pour contrôler le moteur CC du ventilateur.

Le système sera contrôlé à distance à l'aide d'une page web pour laquelle un serveur web sera installé sur le Raspberry Pi. La page web fournira une interface utilisateur graphique pour interagir avec le système. La base de données sera utilisée pour stocker les données de température et de paramètres du système pour une utilisation ultérieure.

L'architecture de ce projet vise à fournir une solution simple et efficace pour réguler la température de l'environnement en utilisant des composants abordables et largement disponibles.

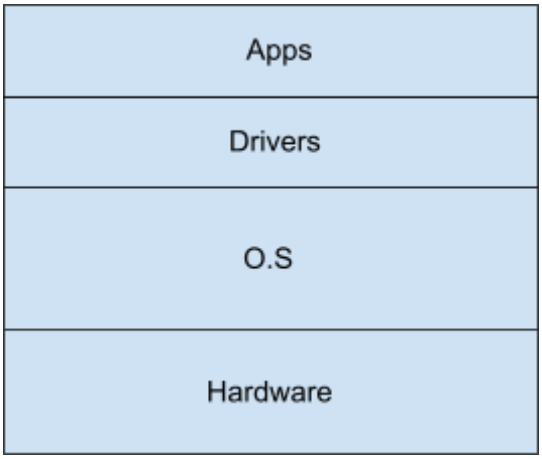

Figure 3 : Organisation d'Architecture de Systeme

La couche Apps:

Cette couche va contenir l’ensemble des applications développées pour atteindre le but du projet ex : les codes C et les scriptes Python3 qui lire et écrire dans les diapositives.

La couche Divers:

Cette couche va contenir l’ensemble des programmes C développées pour contrôler l'accès vers les fichiers d’os, ce qui va nous donner une autre couche d’abstraction pour contrôler le hardware.

La couche O.S:

O.S qu’on va utiliser n’est autre que le Raspberry Pi OS. Le système d’exploitation est basé sur le système d’exploitation Debian. ce qui nous donne une abstraction du hardware en un ensemble des fichiers (système a accès fichier).

La couche Hardware:

Cette couche concerne tous les composants matériels utilisés dans le projet, tels que le capteur de température DS18B20, le moteur avec ventilateur, le module PWM et le Raspberry Pi 4 Model B. Cette couche implique la configuration, l'installation et la connexion des composants.

Organigramme de l'application :

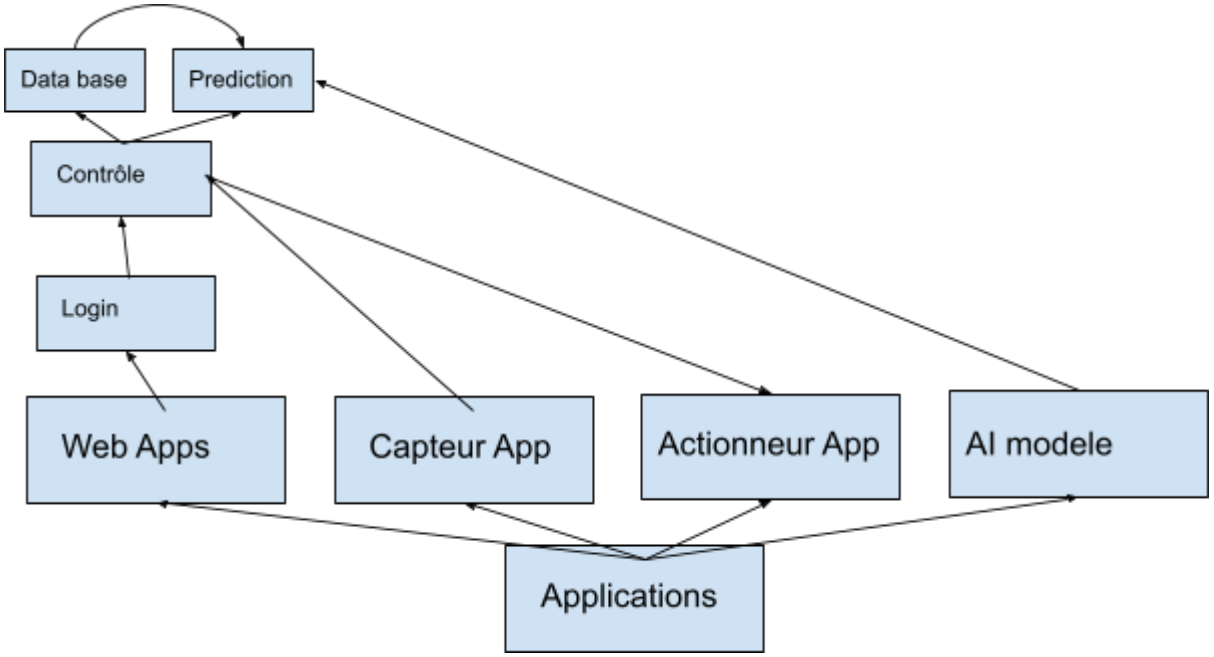

Figure 4 : Organisation d'application

la couche application se compose de 4 applications chaque application avec une tache déterminé ;

Web Application : c'est l'ensemble des code PHP qui permet au client de contrôler les actionneurs et de consulter les valeurs captées.

Capteur Application : c'est une application basé sur un Driver pour le capteur DS18B20

Actionneur Application : en exécutant ce code qui va être déclenchée par une commande Client, le moteur va tourner avec une vitesse fixée par l'utilisateur.

AI modèle : permet d'avoir une prédiction si l'utilisateur va maximiser ou minimiser la vitesse de moteur en fonction du temps et de température actuelle.

Chapitre 3 : Réalisation et test du projet *(le répertoire /Implementation)*

Réalisation du projet :

Partie Client : se compose de 4 page web; login panel se base sur le tableau users de base de donnée et après le login l'utilisateur est dirigé vers ctrpanel ou degrés de température est affiché et il peut contrôler le moteur ou d'aller vers une autre page web.

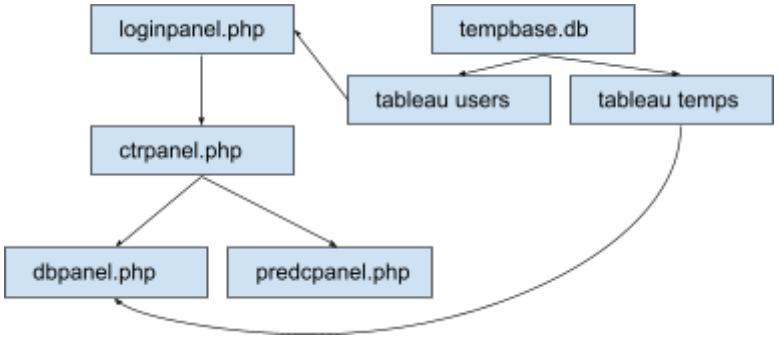

Figure 5 : Réalisation de Partie Client

Partie Serveur :

notre système utilise 2 Drivers; un pour l'acquisition de température et l'autre pour le contrôle PWM les deux sont liés à la page web ctrpanel.

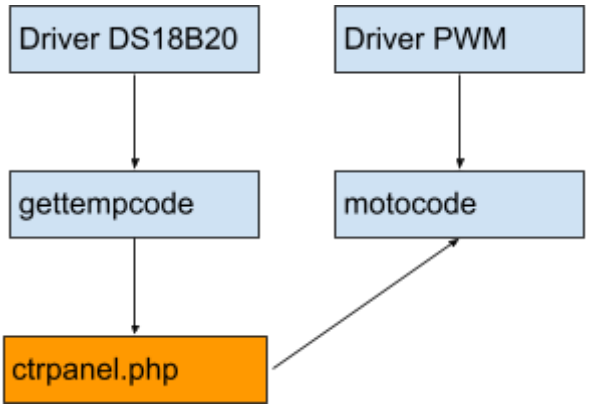

Figure 6 : Réalisation de Partie Serveur, Drivers

le modèle d'intelligence artificielle est basé sur l'algorithme de random forest classifier, pour avoir une précision élevée, après l'entraînement de model in export le modèle dans un fichier pkl en utilisant le package pickle de Python et à base de ce model on construit l'application perfectioncode qui va être sur la plateforme cible

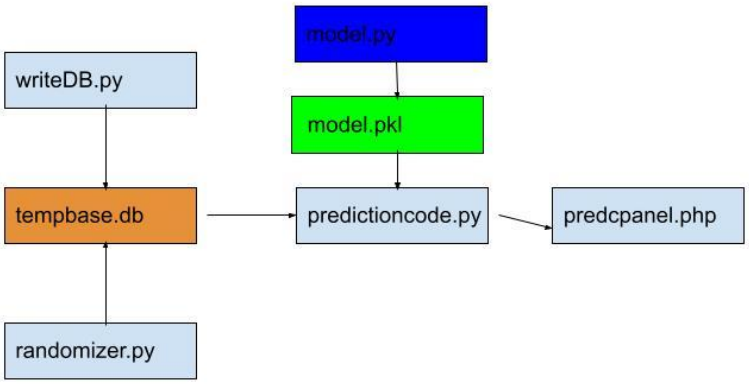

Figure 7 : Réalisation de Partie Serveur, Model AI

Test de l'application :

(les vidéos de test sont valable sous le répertoire /livrable/video_presentation et les codes des test unitaires sous le répertoire /testUnit)

Pour tester et valider la solution proposée, plusieurs scénarios peuvent être envisagés. Voici quelques exemples de scénarios de test qui couvrent différents aspects de l'application :

Scénario de connexion et test d'interface web:

- Ouvrir l'application sur un appareil Android.
- Saisir les informations d'identification (nom d'utilisateur et mot de passe) dans les champs appropriés.
- Appuyer sur le bouton de connexion.
- Vérifier si l'application se connecte au serveur avec succès.
- Après la connexion, accéder à l'écran principal de l'application.
- Vérifier si les données sont affichées correctement dans l'interface utilisateur de l'application.

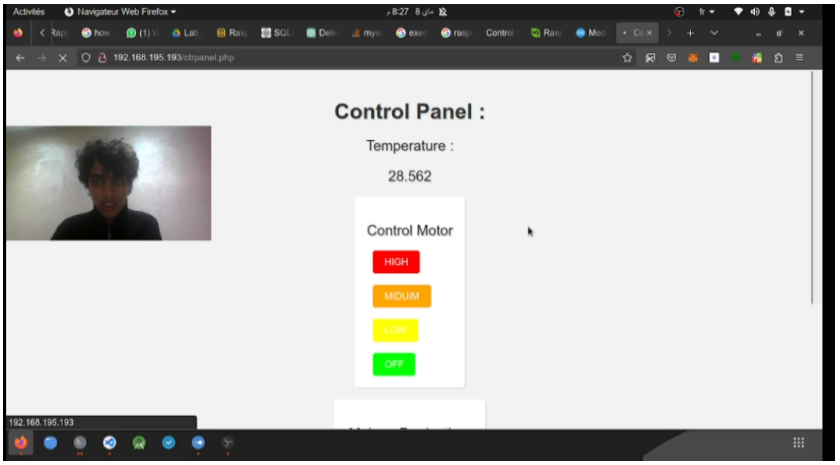

Figure 8 : Test Interface Web Ordinateur

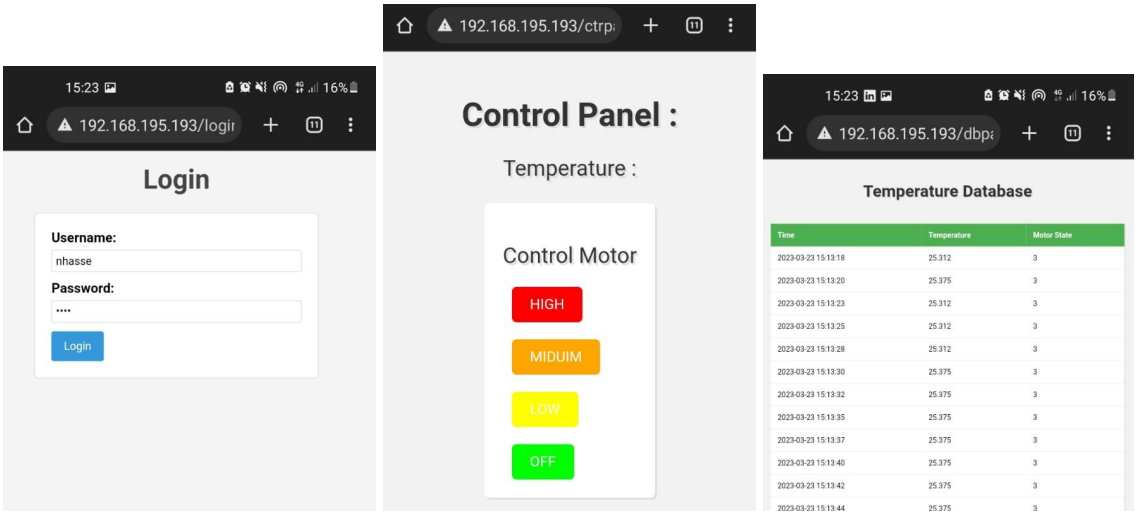

Figure 9 : Test Interface Web SmartPhone

Test des code serveur :

- exécuter les codes d'acquisition de température
- exécuter les codes de contrôle de moteur (puis que je n'avais pas un moteur DC on a utilisé une led pour modéliser la vitess de moteur)
- exécuter les codes d'alimentation de base de donnée
- entraîner le modèle AI sur un base de donnée (randomized) chaque fois pour bien tester est ce que la prédiction est précise

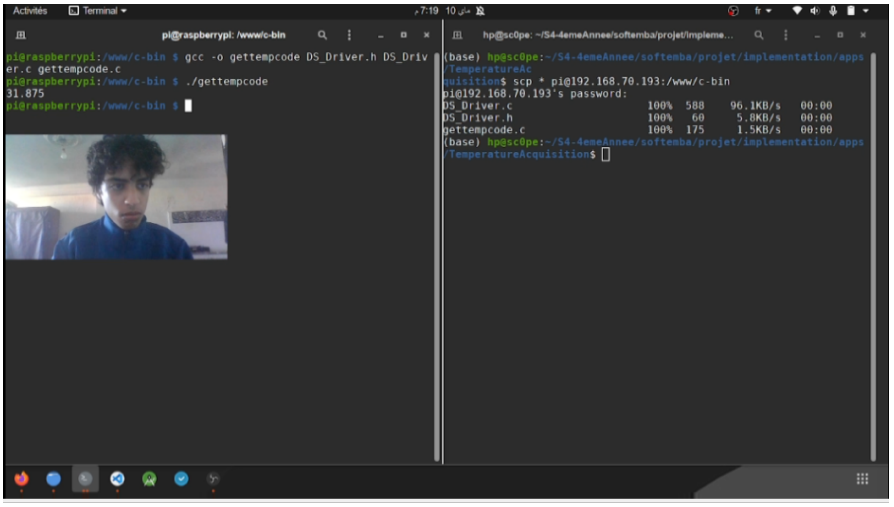

Figure 10 : Test Temperature Acquisition

Le rest des test unitaires sont capturés par des vidéos, sous le répertoire /projet/6-livrable/Video_Presentation/VodsTests

Conclusion Générale :

En somme, ce projet est motivé par la nécessité de développer un système de contrôle de température ambiante efficace et fiable pour diverses applications. Nous avons proposé une solution innovante basée sur l'architecture Client-Serveur et adopté une méthode de développement en cycle en V pour atteindre notre objectif. Tout au long de notre travail, nous avons analysé en détail le logiciel embarqué, en mettant l'accent sur la qualité, la sécurité et la maintenance du système. Notre implémentation de cette solution vise non seulement à résoudre les problèmes actuels de contrôle de température ambiante, mais aussi à offrir des opportunités pour améliorer les performances et la flexibilité des systèmes futurs, en introduisant des systèmes intelligents. Le résultat final de ce projet sera donc un système de contrôle de température ambiante fiable et efficace qui contribuera à la réduction des coûts énergétiques, à l'optimisation de la performance et à la durabilité des équipements.

Références :

https://www.tutorialspoint.com/sqlite/

https://www.waveshare.com/wiki/Raspberry_Pi_Tutorial_Series:_1-Wire_DS18B20_Sensor

https://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/