

# **Rapport de Projet**

**Lukas DELALANDE - Sylvain QUENAULT - Quentin LHOTE**

## **ENERGY MONITORING**

# Partie Commune

## Introduction

Le projet "Energy Monitoring" a été initié pour répondre au besoin crucial de surveillance de la consommation énergétique au sein de notre établissement. Avec une emphase sur la durabilité et l'efficacité énergétique, ce projet vise à développer une solution technique permettant de mesurer et d'analyser la consommation d'énergie en temps réel. Dès la première réunion, les objectifs ont été clairement définis et les rôles attribués, marquant le début d'une aventure collective vers une gestion énergétique optimisée. Lukas s'est vu attribuer le rôle de premier technicien, chargé de créer le Module Mesure (sélection des composants et codage de l'application). Sylvain a reçu le rôle de deuxième technicien, en charge de la mise en place du Module Relais, du Serveur Web et de la création de la carte électronique. Quentin, quant à lui, a endossé le rôle de troisième technicien, chargé de créer l'IHM (Interface Homme-Machine).

## Expression du Besoin

Face à l'augmentation constante des coûts énergétiques et à l'impératif écologique, notre établissement a identifié le besoin crucial de surveiller sa consommation énergétique. Le cahier des charges détaille ce besoin en spécifiant la nécessité d'une solution capable de fournir des mesures précises et en temps réel de la consommation d'énergie, ainsi que d'offrir des analyses permettant d'identifier des pistes d'optimisation.

## Choix Techniques et Contraintes de Développement

Pour répondre à ce besoin, nous avons sélectionné l'ESP32 pour sa polyvalence et sa capacité à communiquer via le protocole RS232, offrant ainsi une solution fiable et économique. Les contraintes de développement comprenaient l'intégration de librairies spécifiques, la compatibilité avec différents OS, et la mise en œuvre d'une solution robuste capable de fonctionner dans divers environnements de production.

Le PZEM a été choisi en raison de son prix, de sa capacité à fonctionner avec une pince à effet Hall et enfin car durant nos recherches sur des installations similaires et Open Source, nous avons constaté l'emploi de ces deux matériels et par conséquent, leur compatibilité en terme de communication. De plus, de nombreuses bibliothèques sont disponibles pour le PZEM.

D'autre part, l'emploi d'une communication en WiFi entre le Module Mesure et un Module Relais nous est apparu comme une évidence du fait du manque de proximité entre les prises réseau (RJ45) et les armoires électriques du lycée.

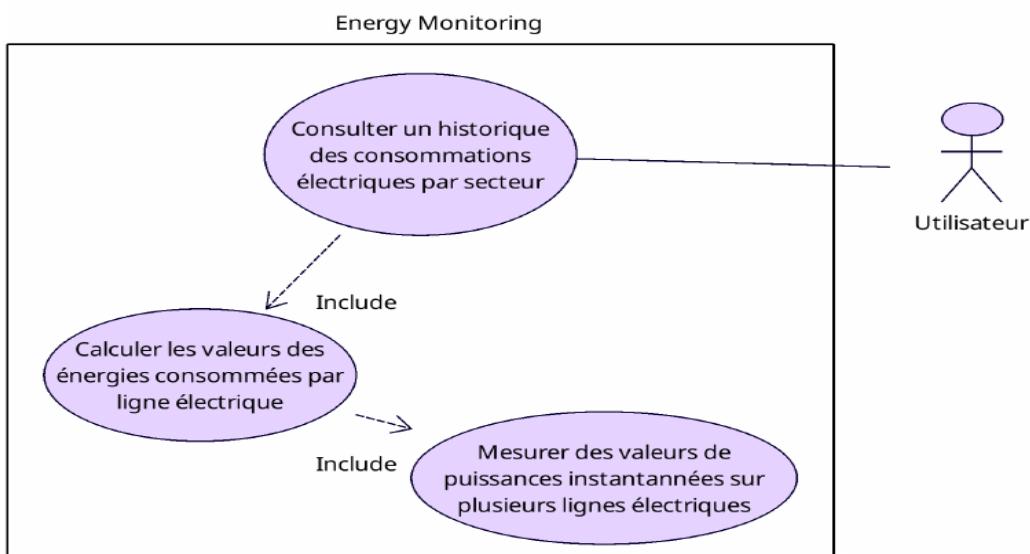
## Coûts du Projet

Article	Quantité	Prix unitaire	Prix total
PCB	5	10,00 €	50,00 €
Transformateur	5	5,20 €	26,00 €
ESP32 DevKitC V4	5	12,99 €	64,95 €
PZEM 004T v3	5	22,99 €	114,95 €
Embout USB C	6	2,00 €	12,00 €
ESP WT32 eth 01	2	19,99 €	39,98 €
Totaux	28		307,88 €

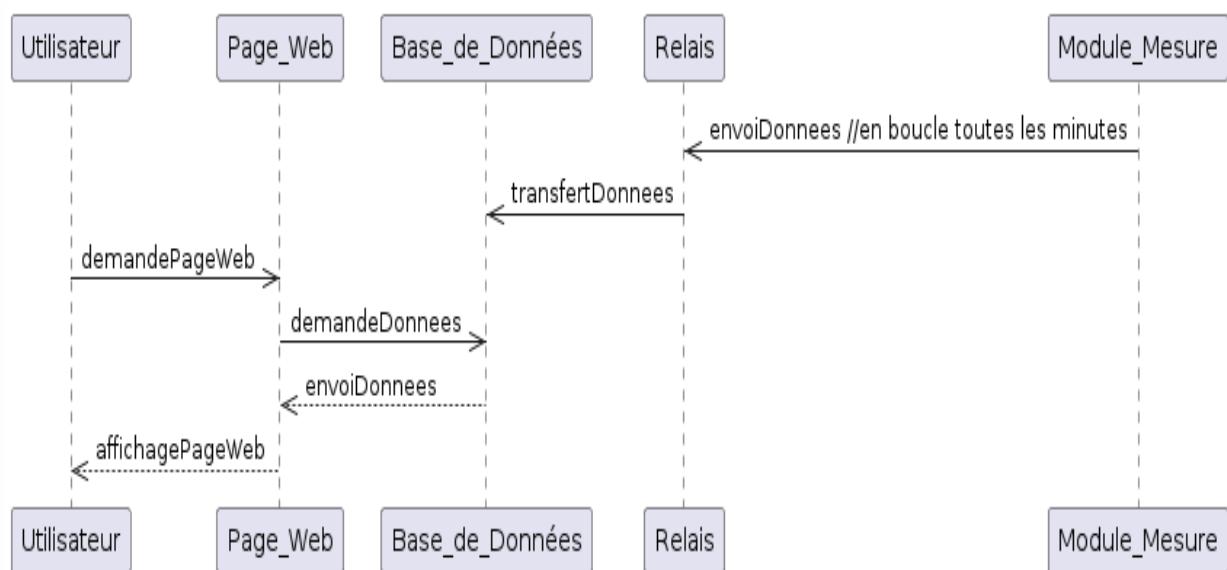
## Analyse UML

Les analyses UML ont été cruciales pour la conceptualisation de notre solution. Dans ce cadre, nous avons établi :

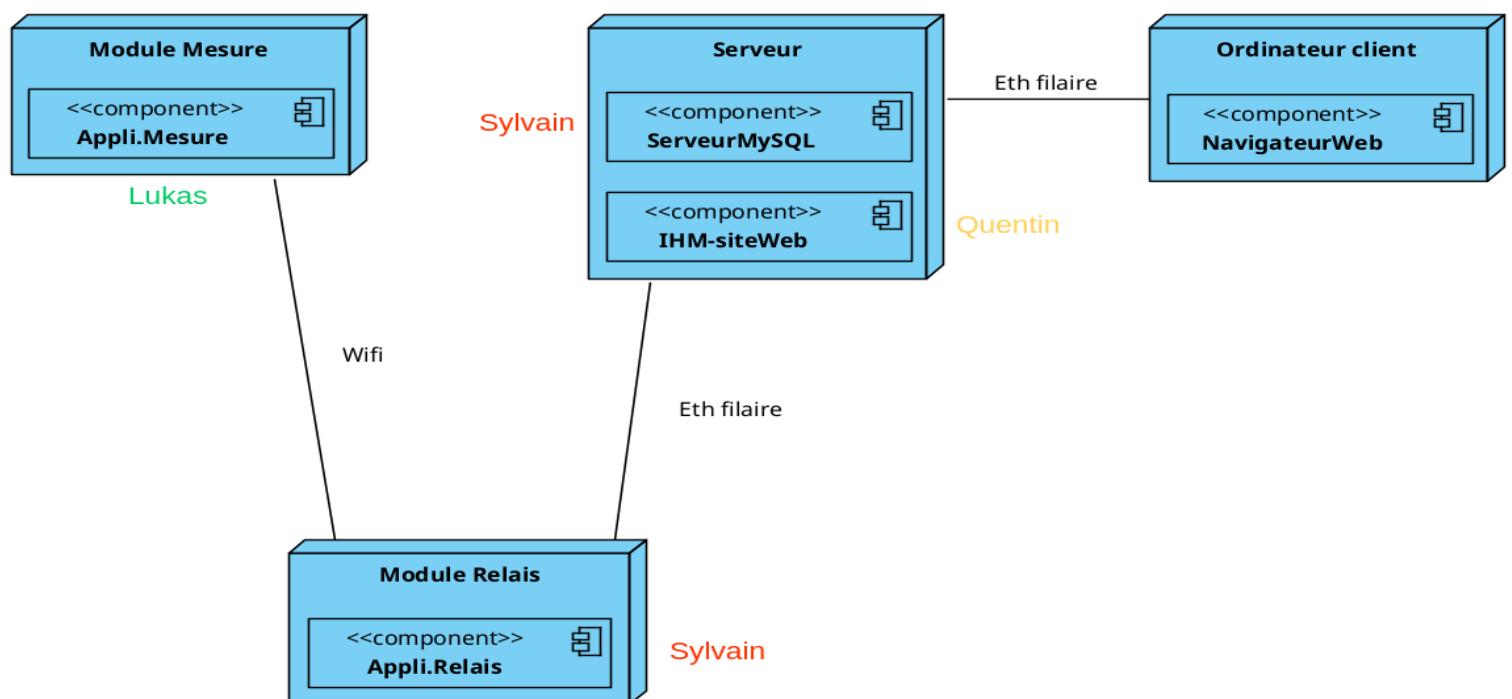
**Cas d'Utilisation** pour identifier les interactions entre les utilisateurs et l'application.



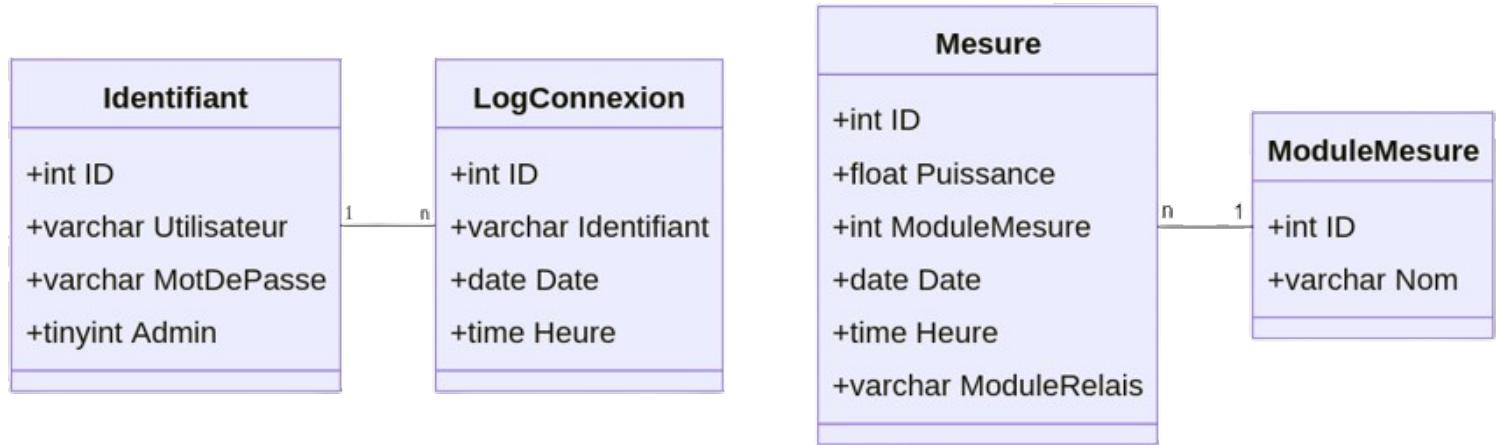
## Diagrammes de Séquence pour visualiser les processus opérationnels.



## Diagramme de Déploiement pour planifier l'intégration système.



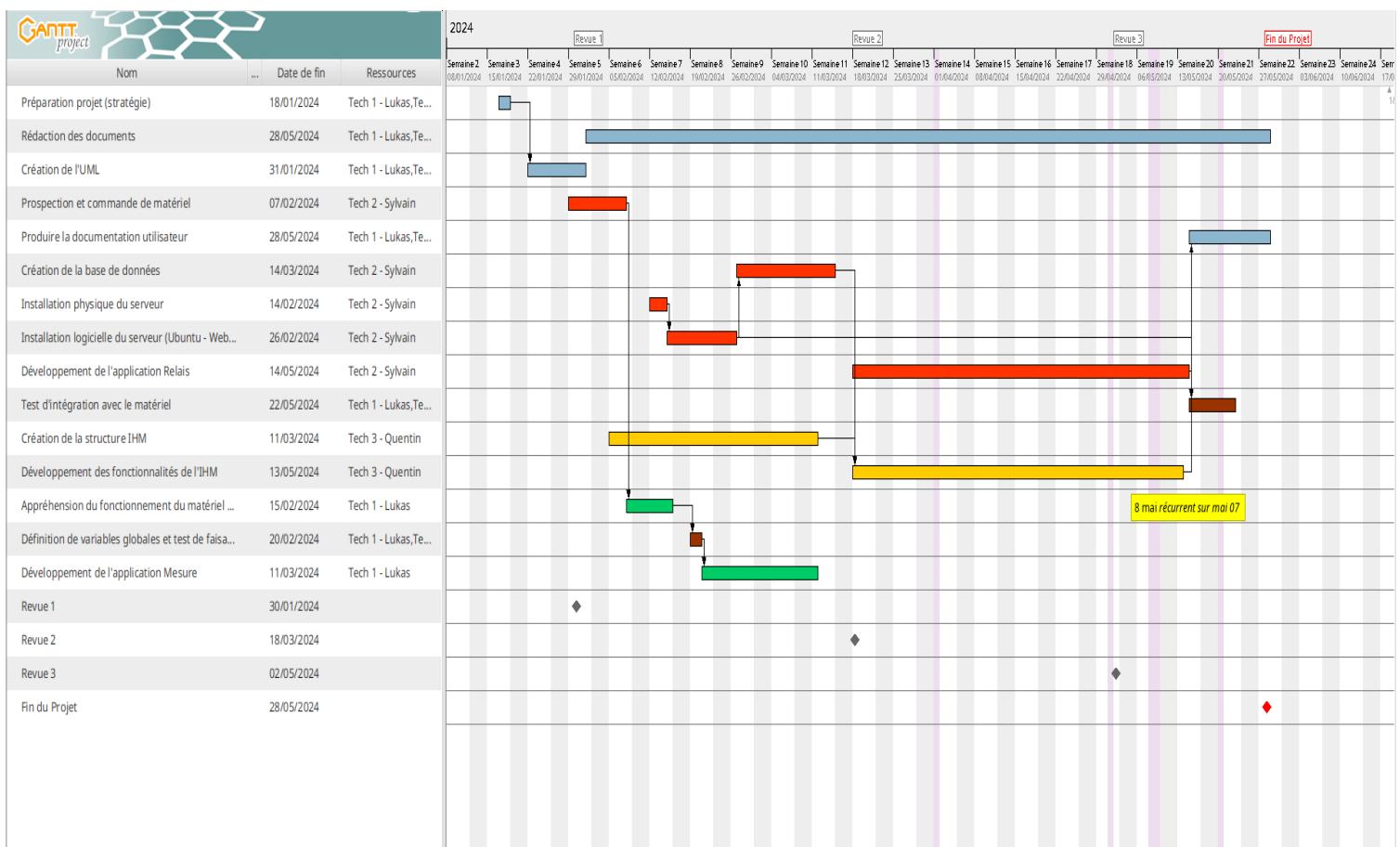
# Modèle Conceptuel de Données



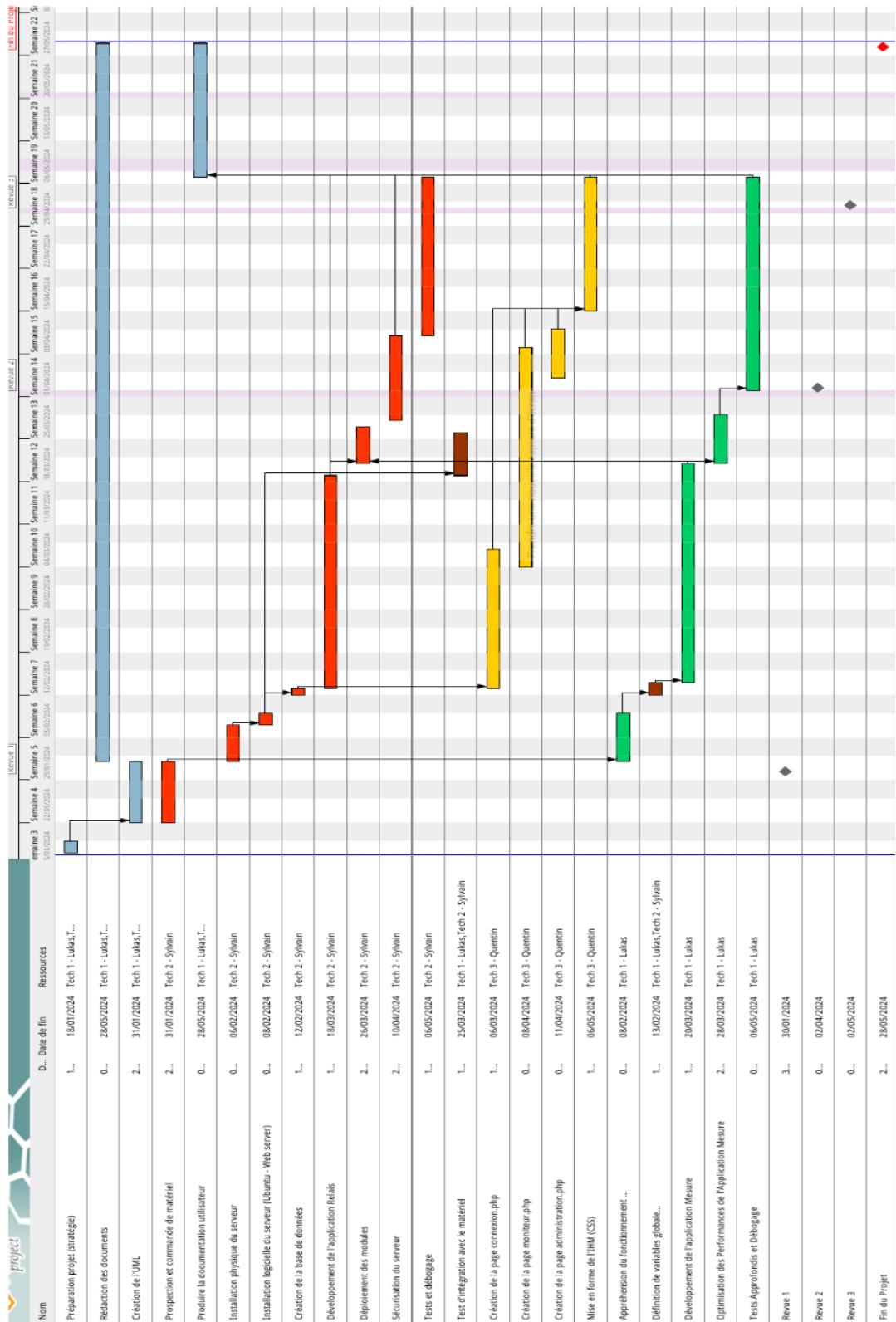
## Répartition des Tâches et Plannings

La répartition des tâches a été effectuée en fonction des compétences de chaque membre, garantissant ainsi une efficacité optimale dans la réalisation du projet. Les plannings prévisionnels et réels ont été établis, offrant une vue d'ensemble claire de l'avancement du projet et de ses différentes étapes.

## Diagramme de Gantt prévisionnel



## Diagramme de Gantt définitif



## **Recette Globale**

La phase de recette a confirmé l'adéquation de notre solution avec les besoins de l'entreprise stipulées dans le Cahier des Charges. À savoir, pouvoir visualiser les consommations en électricité de certaines zones du Lycée Grandmont via la prise de mesure sur les armoires électriques correspondant à ces zones.

À travers des tests approfondis, nous avons validé la fiabilité des mesures, la robustesse de l'application et son interface intuitive sous la forme d'un site internet.

## Partie Individuelle: Lukas

### Introduction

Dans le cadre du projet “Energy Monitoring”, ma contribution principale a consisté à concevoir, développer et tester “l’Application Mesure”, un composant essentiel destiné à la collecte et à l’analyse des données de consommation énergétique. Ce travail a impliqué une série de défis techniques, notamment la sélection de l’hardware, le développement de logiciels embarqués, et l’intégration de fonctionnalités réseau pour permettre une surveillance en temps réel.

### Problème à résoudre :

- Choix du capteur :



**- Impulsion, Gonflement, Creux Mineurs et Majeurs :** Capable de détecter et de réagir à des variations brusques de tension comme des surtensions ou des baisses de tension.

**- Boutons de Commande :** Pour tester ou réinitialiser l’appareil manuellement.

**- Paramètres Mesurables :** Tension nominale, peut-être la fréquence du réseau électrique.

**- Installation :** Montage sur rail DIN pour une installation facile dans des panneaux de distribution ou des armoires électriques.

**Impulsion, Gonflement, Creux Mineurs et Majeurs :** Capable de détecter et de réagir à des variations brusques de tension comme des surtensions ou des baisses de tension.

**Usage :** Adapté pour des environnements industriels ou commerciaux où la stabilité de l’alimentation est cruciale.



**Mesure de Puissance** : Capable de mesurer non seulement le courant mais également la puissance utilisée, souvent exprimée en watts.

**Communication** : Peut inclure des interfaces pour la communication numérique (probablement basées sur des standards comme I2C, SPI, ou UART) pour l'intégration avec d'autres systèmes.

**Usage** : Utilisé dans des applications résidentielles, commerciales, ou industrielles pour le suivi de la consommation d'énergie, efficace pour la gestion énergétique et la réduction des coûts.

- Échange des données :

### Protocole de Communication :

- **Choix du Protocole** : L'utilisation de WiFi intégré dans l'ESP32 a permis de choisir UDP pour l'échange de données en raison de sa simplicité et de son efficacité en termes de latence et de bande passante, surtout dans un environnement local.
- **Mise en Place** : Le code sur l'ESP32 configure une connexion UDP pour envoyer les données mesurées au serveur ou à la base de données centralisée.

### Sécurité et Fiabilité :

- **Gestion des Erreurs** : Le système inclut des mécanismes pour détecter et répondre à des erreurs de transmission, telles que des tentatives de retransmission en cas de paquets perdus.
- **Sécurisation des Transmissions** : Si nécessaire, des méthodes de sécurisation des données, comme le chiffrement ou l'utilisation de tunnels VPN, peuvent être ajoutées pour protéger les données sensibles.

- Format des données :

### Structuration des Données :

- **Définition du Format** : Les données sont structurées en paquets pour faciliter l'intégration avec divers systèmes de gestion de base de données et applications. Cela permet également une extension facile pour inclure d'autres mesures ou informations supplémentaires.
- **Exemple de Format** : Chaque paquet de données peut inclure des champs tels que l'identifiant du module, la tension, le courant, la puissance. Par exemple :  
{} "moduleID": "0001", "power" {}.

### Validation et Tests :

- **Tests de Format** : Des tests automatiques vérifient que les données générées respectent le format attendu et que les systèmes récepteurs (serveur, base de données) peuvent correctement parser et stocker ces informations sans erreurs.
- **Donnée reçue** : Les données reçues proviennent directement du capteur PZEM qui récupère les informations grâce la pince à effet Hall.

## Phase de Recherche et Sélection Technologique

Au début du projet, j'ai réalisé une évaluation approfondie des technologies disponibles, optant finalement pour l'ESP32 en raison de sa connectivité WiFi intégrée, de sa compatibilité avec divers capteurs et de sa communauté d'utilisateurs active. Le choix du capteur PZEM004Tv30 a été motivé par sa précision, sa facilité d'intégration et sa capacité à mesurer plusieurs paramètres électriques.

## Conception Préliminaire et Prototypage

J'ai élaboré un schéma initial de l'application, définissant les principales fonctionnalités, l'interface utilisateur, et la logique de communication. Un prototype de base a été développé pour tester ces concepts, permettant d'identifier les premiers défis techniques et de valider l'approche choisie.

## Schéma de Fonctionnement

### Matériel Utilisé :

1. **ESP32** : Microcontrôleur avec connectivité WiFi intégrée.
2. **PZEM004Tv30** : Capteur de mesure de la tension, du courant et de la puissance électrique.
3. **Pince à effet Hall** : Capteur de courant basé sur l'effet Hall, utilisé pour mesurer le courant sans contact direct avec le conducteur principal.
4. **Source de Puissance Électrique Connue** : Pour calibrer et vérifier les mesures.
5. **Câblage** : Pour les connexions électriques et les communications série.

### Connexions :

#### 1. Pince à effet Hall au PZEM004Tv30 :

- La pince à effet Hall est connectée aux bornes du PZEM004Tv30 pour mesurer le courant circulant dans le conducteur.

#### 2. PZEM004Tv30 à l'ESP32 :

- **Broche TX du PZEM004Tv30** connectée à la **broche RX (GPIO16)** de l'ESP32.
- **Broche RX du PZEM004Tv30** connectée à la **broche TX (GPIO17)** de l'ESP32.

#### 3. Source d'Alimentation :

- Alimentation de l'ESP32 via une source d'alimentation 5V ou USB.
- Alimentation du PZEM004Tv30 via la source de mesure.

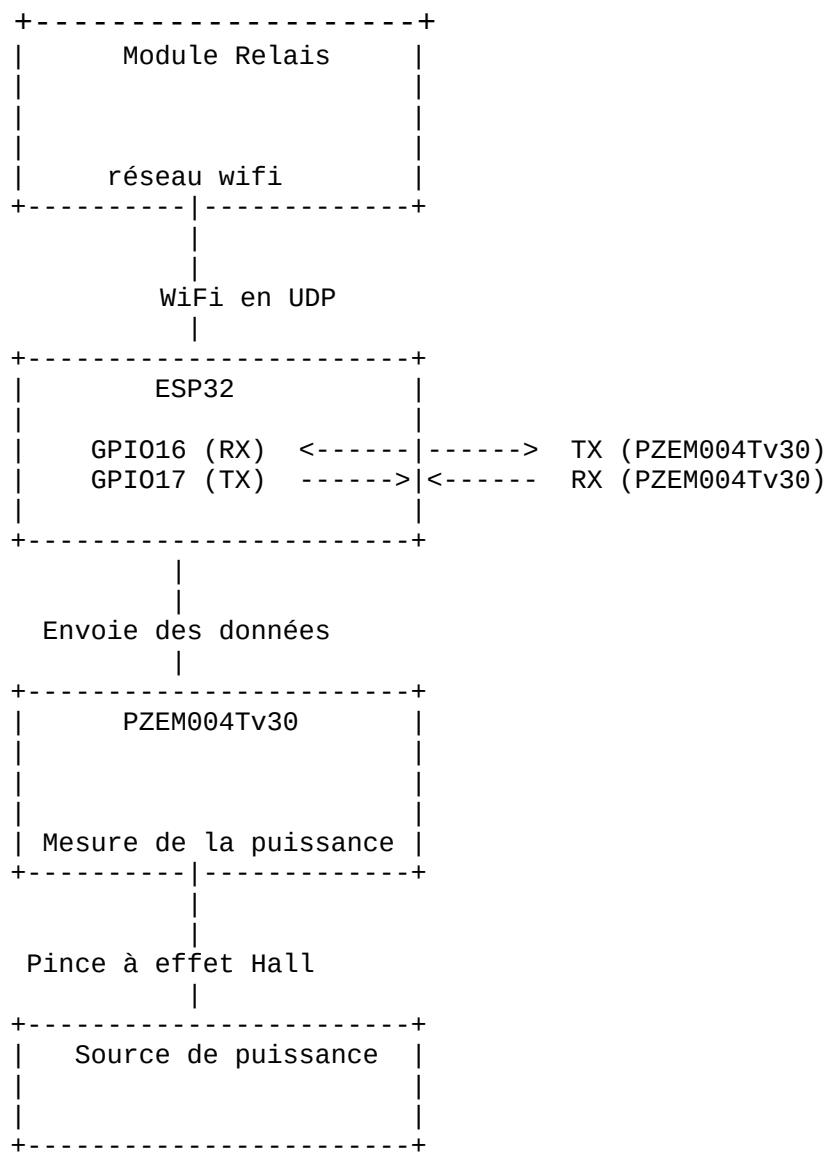
#### 4. Connexion Réseau :

- L'ESP32 se connecte au réseau WiFi configuré pour envoyer les données collectées à un serveur distant.

## Visualisation du Schéma

Pour visualiser le schéma, voici une description qui peut être utilisée pour dessiner un schéma électronique :

- **ESP32** connecté au **PZEM004Tv30** avec les broches RX/TX.
- **PZEM004Tv30** connecté à la **pince à effet Hall** pour la mesure de la puissance.
- L'ESP32 connecté à un module relai disposant d'un point d'accès **WiFi** pour envoyer les données à un **serveur distant**.
- **Alimentation 230v** connectés au **PZEM004Tv30** pour la mesure de la puissance.



## Fonctionnement :

### 1. Mesure du Courant :

- La pince à effet Hall mesure le courant dans le conducteur principal sans contact direct, et transmet cette mesure au PZEM004Tv30.

### 2. Traitement des Données :

- Le PZEM004Tv30 collecte les données de tension, de courant et de puissance.
- Ces données sont transmises à l'ESP32 via une communication série.

### 3. Transmission des Données :

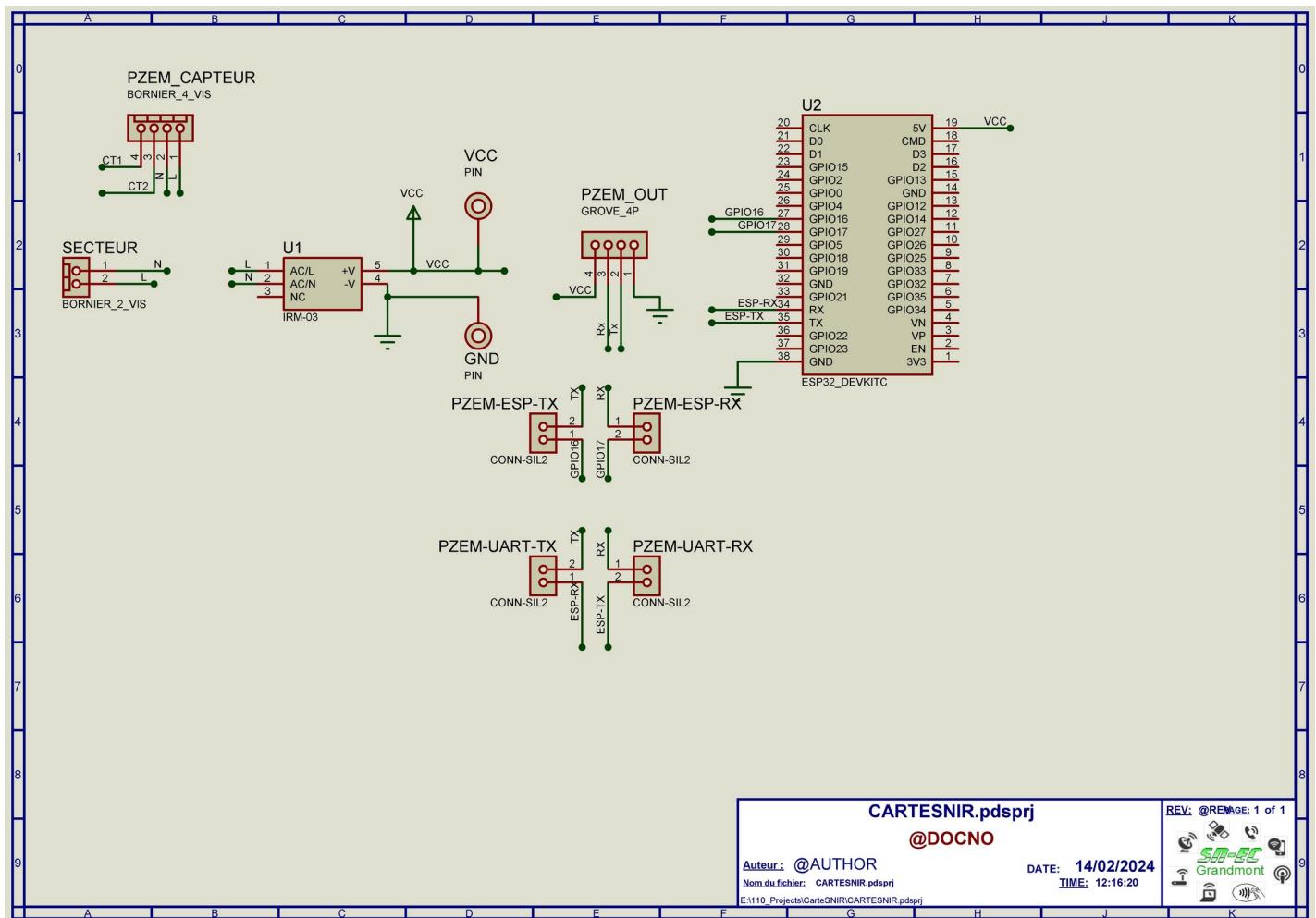
- L'ESP32 reçoit les données du PZEM004Tv30, les traite et les formate en paquets JSON.

- Les paquets de données sont ensuite envoyés à un serveur distant via une connexion WiFi en utilisant le protocole UDP.

#### 4. Surveillance et Analyse :

- Les données reçues par le serveur peuvent être analysées pour surveiller la consommation énergétique, détecter des anomalies et prendre des décisions basées sur ces analyses.

### Schéma électrique du capteur PZEM004Tv30 :



## **Conception, Configuration, Réalisation**

### **Configuration de l'Environnement de Développement**

La configuration initiale comprenait la mise en place de l'environnement IDE pour ESP32, l'installation des librairies nécessaires (comme celles pour la gestion du WiFi et la communication UDP), et la préparation du capteur pour les tests.

### **Développement de l'Application**

Le cœur du développement a porté sur la programmation de l'ESP32 pour qu'il lise les données du capteur et les transmette via le réseau à un serveur d'analyse. Ce processus a impliqué l'écriture de fonctions pour la collecte de données, la gestion des erreurs, et l'envoi des données.

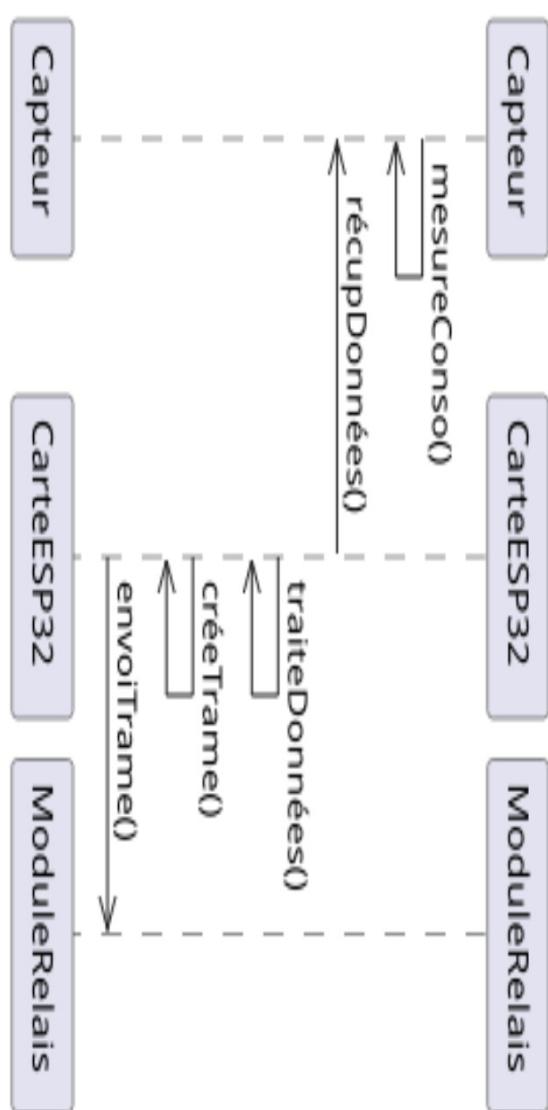
### **Intégration et Tests**

Après la phase de développement, j'ai procédé à l'intégration de l'application avec le hardware et réalisé une série de tests pour assurer la fiabilité et l'exactitude des données collectées. Cela incluait des tests unitaires pour chaque fonctionnalité et des tests d'intégration pour évaluer l'application dans son ensemble.

### Cahier de Recette

### **Tests Unitaires et d'Intégration**

Les tests unitaires ont été menés sur des composants individuels de l'application, tels que la lecture des données du capteur et la communication réseau. Les tests d'intégration ont ensuite évalué le système complet, garantissant une interaction fluide entre tous les composants.



## Fiche de test

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale**  
 **Test de faisabilité**  
 **Test unitaire**  
 **Test d'intégration**

**Nom de l'étudiant :** Lukas Delalande

**Date :** 08/04/2024

**Objectif du test :** Vérifier l'exactitude des mesures de puissance électrique recueillies par l'application Mesure.

### Conditions de réalisation

- **Besoins matériels :**
  - **Carte ESP32** : Doit être opérationnelle et configurée préalablement.
  - **Capteur de puissance électrique PZEM004Tv30** : Connecté à la carte ESP32.
  - **Source de puissance électrique connue** : Pour calibrer et vérifier les mesures de l'application.
- **Besoins logiciels :**
  - **Firmware sur la carte ESP32** : Inclut les bibliothèques pour la lecture des données du capteur PZEM004Tv30 et l'envoi des données via WiFi.
  - **Application serveur** : Pour recevoir et enregistrer les données transmises par la carte ESP32.
- **Noms des fichiers utilisés** : ApplicationMesure.ino : Contient le code source pour la configuration et le fonctionnement de l'ESP32.

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Connectez la source de puissance électrique connue au capteur PZEM004Tv30.  Alimentez la carte ESP32 et attendez l'établissement de la connexion WiFi.  Observez les mesures affichées par l'application et comparez-les aux valeurs attendues de la source de puissance.	<b>Source de puissance connue</b> : Puissance spécifique en watts.  <b>Configuration initiale du capteur</b> : Assurez-vous que le capteur est correctement calibré selon les spécifications techniques.	<b>Mesures précises</b> : Les valeurs mesurées par l'application doivent correspondre étroitement aux valeurs connues de la source de puissance	OK

2			
3			

**Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale**

**Test de faisabilité**

**Test unitaire**

**Test d'intégration**

**Nom de l'étudiant :** Lukas Delalande

**Date :** 08/04/2024

**Objectif du test :** Vérifier la capacité de l'ESP32 à se connecter au réseau WiFi et à communiquer avec le serveur.

### Conditions de réalisation

- **Besoins matériels :**
  - **Carte ESP32** : Configurée avec le dernier firmware.
  - **Ordinateur ou serveur** : Pour recevoir les données.
  -
- **Besoins logiciels :**
  - **Firmware de l'ESP32** : Incluant des bibliothèques WiFi.
  - **Serveur de réception de données** : Logiciel en cours d'exécution pour collecter les données.
- **Noms des fichiers utilisés** : ApplicationMesure.ino

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	<b>Allumez l'ESP32.</b> <ol style="list-style-type: none"> <li>1. L'ESP32 tente de se connecter au réseau WiFi configuré.</li> <li>2. Vérifiez si l'adresse IP est attribuée à l'ESP32.</li> <li>3. Envoyez une charge de données test au serveur.</li> </ol>	<b>SSID et mot de passe du réseau WiFi :</b> Correctement configurés sur l'ESP	<b>Connexion réussie :</b> L'ESP32 doit recevoir une adresse IP valide et envoyer des données au serveur. <ul style="list-style-type: none"> <li>• <b>Échec de connexion :</b>              Si la connexion échoue, vérifiez la portée du signal WiFi, les identifiants réseau, et redémarrez l'ESP32.           </li> </ul>	OK

2				OK
3				

**Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale**

**Test de faisabilité**

**Test unitaire**

**Test d'intégration**

**Nom de l'étudiant :** Lukas Delalande

**Date :** 08/04/2024

**Objectif du test :** Évaluer la stabilité et la fiabilité de l'application Mesure sur de longues périodes.

### Conditions de réalisation

- Besoins matériels : Carte ESP32 avec application Mesure :** En fonctionnement continu.
- Besoins logiciels : Logiciel de surveillance :** Pour surveiller la performance de l'application.
- Noms des fichiers utilisés :** ApplicationMesure.ino

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	<b>Laissez l'application Mesure fonctionner continuellement pendant une période prolongée (ex. 24 heures).</b> 1. Surveillez la performance et reportez tout comportement anormal.	<b>Application Mesure en cours d'exécution :</b> Sans interruption.	<b>Stabilité de l'application :</b> L'application doit fonctionner sans erreurs ou interruptions majeures. Documentez tout comportement anormal ou défaillance système.	OK

## **Problèmes Rencontrés et Solutions**

Plusieurs défis sont survenus durant le projet, notamment des problèmes de connectivité WiFi intermittente et des lectures inexactes du capteur sous certaines conditions. Pour résoudre ces problèmes, j'ai optimisé le code pour une meilleure gestion des erreurs et ajusté les paramètres du capteur pour améliorer la précision.

## **Conclusion Personnelle**

Ce projet a été une expérience enrichissante, me permettant de développer mes compétences en programmation de systèmes embarqués et en gestion de projets technologiques. Les défis rencontrés m'ont appris l'importance de la persévérance et de l'adaptabilité dans le domaine du génie logiciel. Les possibilités d'amélioration incluent l'ajout de fonctionnalités d'analyse prédictive et l'amélioration de l'interface utilisateur pour une meilleure expérience. Ce projet m'a également montré l'importance cruciale de la surveillance énergétique dans la réduction des coûts et la préservation de l'environnement, soulignant le rôle que la technologie peut jouer dans la réalisation d'un avenir durable.

## **ANNEXES**

- Manuels d'Installation et d'Utilisation** : Guide détaillé pour une mise en place et une utilisation efficace de l'Application Mesure.
- Code Source** : Inclus le code développé pour le projet, fournissant une base solide pour des améliorations futures.

**Installation Arduino IDE** : Guide d'installation de Arduino IDE dans un environnement Linux.

# **Partie Individuelle: Sylvain**

## **Introduction**

La tâche qui m'a été confiée était de créer un module Relais qui créer un réseau wifi et qui transmet les informations à une application Server qui devait ensuite communiquer avec la base de données. Je devais aussi installer le serveur (Apache2 , mysql , phpmyadmin & toute la sécurité du serveur, ...).

J'ai donc mis en place des solutions collant au plus près à ces exigences et en accord avec le travail de mes deux collègues techniciens.

## **Développement et Solutions Envisagées**

Afin de réaliser ce projet, j'ai choisi un serveur HP DL360 G7 qui est équipé de deux processeurs puissants ainsi que 64 GB de ram et de deux disques HDD 7200 rpm de deux tera chacun.

Pour le module Relai j'ai décidé de créer un boîtier imprimé en 3d afin d'y mettre un ESP WT32 ( ESP32 qui a une carte wifi et un port rj45 ) et j'ai rajouter un port USB C pour pouvoir l'alimenter.

### **▪ Sélection Technologique et Phase de Recherche**

Pour la mise en place du serveur j'ai utilisé comme OS : UBUNTU Server car il est optimisé , stable et gratuit. Pour le serveur web et la base de données j'ai décidé avec mes collègues d'utiliser Apache2 ainsi que MariaDb et PhpMyAdmin car c'est très utilisé donc la documentation en ligne est très fournis, nous avons aussi décidé de l'utiliser car c'est des installations que nous avons déjà effectuer en cours et que nous connaissons. Toujours pour le serveur et dans une optique de sécurité j'ai mis les deux HDD en RAID1.

Pour le module Relai j'ai décidé d'utiliser un ESP WT32 car il possède une antenne wifi et un port RJ45, une autre option aurait été d'utiliser un ordinateur style RaspberryPi , cela aurait très bien fait le travail mais c'était une option bien plus coûteuse et qui consomme plus d'électricité .

En conclusion nous avons retenu le WT32 car il est moins cher, moins énergivore, petit et fiable.

Pour l'application Serveur je voulais utiliser du C++ car c'est ce que j'ai vu en cours et que l'utilisation devait se limiter à recevoir des données et à les rangées dans la base de données.

### **▪ Conception Préliminaire et Prototypage**

Au départ je comptais faire une application serveur mais a force de réflexion et de test je me suis rendu compte que depuis le ESP WT32 je pouvais directement communiquer avec la base de données, donc lui envoyer des trames , J'ai donc décider

de fusionner le module relai avec l'application serveur.

En définitive , j'ai donc un module relais qui range directement et automatiquement les données reçues dans la base de données , c'est optimiser et ça évite d'avoir une application en plus sur le serveur.

## Conception, Réalisation

Pour commencer j'ai dû me mettre d'accord sur comment le technicien 1 m'envoie les données , à force de discussion nous avons décidé que j'allais créer un réseaux wifi qui distribue des adresses IP automatiquement au module Mesure qui se connecte dessus et qui attend simplement des trame UDP , nous avons normer les trames de la manières suivante : "N= le numéro du module mesure P= la puissance calculé \n"

exemple "N=0001P=70.81\n"

donc dans l'exemple c'est le module 0001 et il a mesuré 70.81 Watts

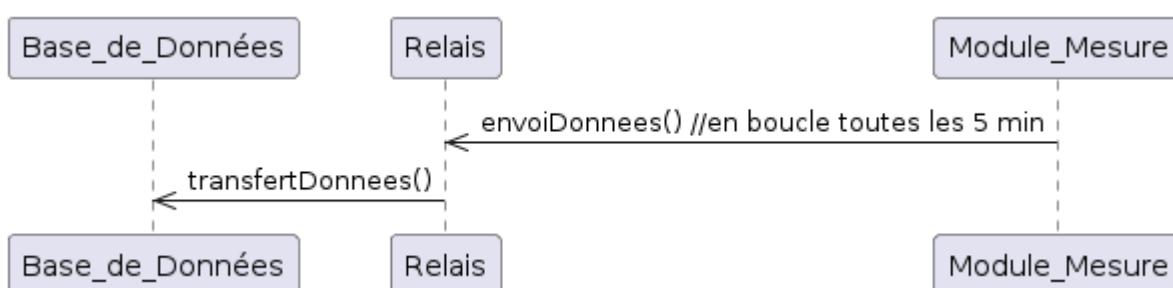
une fois ceci effectué on a pu commencer a communiquer

Par la suite j'ai dû m'occuper de ranger les données dans la BDD j'ai donc ranger le numéro du module mesure dans une variable ainsi que la puissance calculé et j'ai commencé à remplir une requête SQL qui contient le numéro du module mesure, la puissance réceptionnée précédemment , la date ( CURDATE() ) suivi de l'heure ( CURTIME() ) et enfin le nom du module relais .

Avec cette trame je peux me passer de l'application serveur et limité le nombre d'endroits ou des erreurs peuvent se produire et ça permet aussi de gagner du temps.

## Cahier de Recette

- **Diagramme de Séquence Technicien 2 – réception des données et communication avec la base de données**



- **Tests Unitaires et Fiche de Recette**

## Fiche de Recette

### Fiche de test

**Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale**

**Test de faisabilité**

**Test unitaire**

**Test d'intégration**

**Nom de l'étudiant :** Sylvain Quénault

**Date :** 22/05/2024

**Objectif du test :** S'assurer du bon fonctionnement du module relais et de la BDD

#### Conditions de réalisation

Besoins matériels

- un PC
- un Serveur local
- ESP WT32 (Module Relais)
- ESP32 (Module Mesure)

Besoins logiciels (nom et version)

- wireshark
- une Base de Données MySQL

#### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Brancher le module relais	aucune	led rouge qui s'allume	OK
2	Brancher le module mesure	aucune	led rouge qui s'allume	OK
3	Création d'un réseau wifi	SSID / Mot de passe	voir le réseau wifi sur le PC	OK
4	Réceptionner des données	Module mesure envoie des données	Avec Wireshark vérifier la bonne diffusion des données sur le réseau wifi	OK
5	Vérifier les données	WT32 brancher sur le réseaux en RJ45	ajout d'entrée dans la BDD	OK

**Pour vérifier les trames sur le réseaux wifi j'ai utiliser wireshark et le module mesure envoie les données sur le broadcast**

**Pour vérifier que des nouvelles entrées sont bien dans la BDD , j'ai simplement vérifié avec PhpMyAdmin.**

## Fiche de Test 1

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

**Nom de l'étudiant :** Quénault Sylvain

**Date :** 22/05/24

**Objectif du test :** Vérification de la création du réseau wifi via le module relais

### Conditions de réalisation

Besoins matériels

- un PC
- ESP WT32

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Brancher l'ESP WT32	aucune	Led rouge qui s'allume	OK
2	Attendre	SSID / PASSWORD	Création d'un réseau wifi	OK
3	Vérification du réseau wifi	aucune	Voir le réseau wifi dans la liste des réseaux	OK
4	Connexion au réseaux wifi	SSID / PASSWORD	Vérifier bonne connexion et l'attribution d'une IP	OK

## Fiche de Test 2

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

**Nom de l'étudiant :** Quénault Sylvain

**Date :** 22/05/24

**Objectif du test :** Test de vérification des trames reçus par le module mesure

### Conditions de réalisation

Besoins matériels

- Module Mesure
- Module Relais
- un PC

Besoins logiciels (nom et version)

- Wireshark

Scénario				
ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Mettre en marche les modules	SSID / PASSWORD	Création d'un réseau wifi et connexion du module mesure sur le réseau	OK
2	Se connecter sur le réseau wifi avec le PC	SSID / PASSWORD	Attribution d'une adresse IP	OK
3	Lancer wireshark	aucune	Voir les trame qui défile sur le réseau wifi & vérifier le respect de la norme mis en place	OK

### Fiche de Test 3

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

Nom de l'étudiant : Quénault Sylvain

Date : 15/02/24

Objectif du test : Communication WT32 / MySql

#### Conditions de réalisation

Besoins matériels

- un PC
- un Serveur MySql
- un WT32

Besoins logiciels (nom et version)

- un navigateur

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	téléverser un programme qui envoie des données de test dans une BDD	string = "TEST" DbName = "TEST" Login = "TEST" MDP = "TEST" IP = "172.16.150.1"	aucun	OK
2	Créer une base de données et la mettre sur le réseau	DbType = "TEST" Login = "TEST" MDP = "TEST" IP = "172.16.150.1"	création de la base de données et vérification dans PhpMyAdmin	OK
3	vérifier les entrées dans la		Voir "TEST" qui s'ajoute dans la	OK

	base de données via port série et via PhpMyAdmin	BDD et voir les messages de Debug sur le port série du WT32	
--	--	---	--

## ▪ Problèmes Rencontrés et Solutions

L'ESP WT32 est un ESP géniale mais assez peu utilisé, il est donc vraiment mal fourni en librairie et en code exemple .

J'ai trouver du code exemple pour réceptionner des trames UDP mais pour un ESP32 normal qui avec quelque modification a fini par fonctionner sur le WT32 , mais ensuite j'ai du faire la partie RJ45 et j'ai longuement cru qu'il était impossible d'utiliser les deux en même temps. A chaque fois que je démarrais le port RJ45 le Wifi de l'ESP se coupait. c'est alors que je me suis rendu compte qu'il fallait utiliser une librairie spéciale pour que le port RJ45 se lance en second sur un autre cœur .

Le deuxième problème que j'ai rencontré c'est pour téléverser le code sur l'ESP WT32 qui ne possède ni port USB B ni port USB C il faut donc venir avec un programmeur pour lui envoyer du code. Cependant je pensais qu'il suffisait de brancher le 5v-5v, gnd-gnd, tx-tx & rx-rx mais en fait il y a toute une procédure pour pouvoir le mettre en mode écoute sur le port serial. la procédure étant introuvable 100% fonctionnelle en ligne j'ai dû y aller au pifomètre jusqu'à trouver les bon réglage , Malheureusement un module a rendu l'âme dans cette opération nécessaire (RIP). le protocol pour passer en mode écoute est de mettre le 5v - 5v , gnd - en0 , tx-rx0 & rx-tx0 puis de faire contact entre la broche GND et io0 , et enfin téléverser le code , puis par la suite il faut remettre 5v-5v & gnd-gnd pour pouvoir le démarrer.

En troisième j'ai eu un léger soucis pour l'adresse IP du serveur , étant dans une section et voulant que ca fonctionne partout je voulais mettre un domaine local du genre : "ENERGIE.LOCAL" mais je n'es pas réussie et le prof a mis une IP static dans la section.

En quatrième j'ai eu un problème avec l'OS que je voulais utiliser ( DEBIAN serveur ) car bien que très utilisée elle supporte très mal le multi processeur , j'ai lors de l'installation remarque que seulement la moitié de la ram qui est détecté ainsi que la moitié des coeurs des processeurs , ce qui indique donc qu'un processeur n'est pas détecté . Apres verification sur des forums cela vient de Debian et de certain serveur qui ne sont pas compatible , pour régler ça il n'y a que deux option :

1. Changer de serveur

clairement pas envisageable car trop cher et surtout inutile.

2. Changer d'OS

C'est la solution que j'ai choisie car c'est gratuit et que je connais aussi bien ubuntu serveur donc c'est pas problématique.

## **Conclusion Personnelle**

En conclusion ce projet m'a permis d'approfondir mes compétences de travail en équipe , d'électronique avec la conception de la carte électronique , de la micro informatique avec l'utilisation d'esp 32 ainsi que l'installation de serveur , j'ai pu aussi en profiter pour revoir de la conception et de l'impression 3D et ainsi travailler avec différente filières dans le lycée.

je remercie mes professeurs d'informatique qui durant mes deux années mon permis d'acquérir les compétences nécessaire pour mener à bien ce projet et je remercie aussi mon professeurs de physique qui m'a lui aussi permis de mener à bien la conception de la carte électronique et enfin je remercie les professeurs des autres filières qui mon imprimer ma conception 3D et qui on créer la carte électronique qui serait resté à l'état de schémas sans eux.

Ce projet a vraiment été très intéressant et j'ai pris un grand plaisir à travailler avec mes camarades pour le mener à son terme , ça a été pour moi une expérience enrichissante et vraiment plaisante.

# Partie Individuelle: Quentin

## Introduction

La tâche qui m'a été confiée était de créer une page web sur laquelle l'utilisateur peut visualiser périodiquement, grâce à un historique de mesures, la puissance (en Watts) ou la consommation électrique (en kW/h).

J'ai donc mis en place des solutions collant au plus près à ces exigences et en accord avec le travail de mes deux collègues techniciens.

## Développement et Solutions Envisagées

Afin de réaliser ce projet, j'ai choisi de créer un site internet comprenant trois pages: une première page de connexion permettant de sécuriser l'accès à une deuxième page de consultation des mesures récupérées et visualisables sous forme d'un graphique. J'ai également créé une troisième page afin de pouvoir gérer les différents utilisateurs du site (ajout, retrait et modification) et de pouvoir consulter le journal des connexions liées à ces utilisateurs.

### ▪ Sélection Technologique et Phase de Recherche

J'ai utilisé les langages HTML (HyperText Markup Language), PHP («Personal Home Page» ou encore Préprocesseur d'Hypertexte) pour mettre en place la plupart des fonctionnalités de mon site web.

J'ai aussi utilisé du JavaScript car j'ai trouvé une bibliothèque permettant de générer des graphiques sous ce langage. Ce type d'affichage m'a paru être la façon la plus adaptée (de par sa très bonne lisibilité) dans le but transmettre les données à l'utilisateur. Le JavaScript a également servi à rendre plus intuitive la gestion des utilisateurs dans la page Administration.

Enfin, j'ai utilisé le CSS (Cascading Style Sheets) afin de rendre plus agréable la consultations des pages de mon site internet.

De plus, on peut aisément trouver de la documentation concernant ces différents langages, ce qui a facilité ma phase de recherche, étant au commencement du projet, un néophyte en terme de webmastering.

Après avoir prospecté puis testé plusieurs styles de graphiques en Php sans que cela soit satisfaisant. J'ai donc finalement choisi la bibliothèque Chart.js car elle permet de générer un graphique visuellement plus beau, plus lisible et de mettre des dates sur l'axe des abscisses.

L'outil que j'ai utilisé pour le développement des pages web est Visual Studio Code.

### ▪ Conception Préliminaire et Prototypage

Au départ du projet, j'avais eu l'idée de faire un graphique qui montre les valeurs sur une unique page nommée Moniteur. On s'est ensuite mis d'accord sur la création supplémentaire d'une page de connexion afin de limiter l'accès aux seuls utilisateurs autorisés ainsi que d'une page d'administration pour gérer ces derniers et leur statut (administrateur ou non).

## Conception, Réalisation

### ▪ Développement de l'Interface Homme Machine (ou IHM) sous forme de page web

- ◆ Pour commencer la page de Connexion, j'ai créé deux zones de saisie: «Utilisateur» et «Mot de Passe» dont j'ai transféré le contenu grâce à la méthode POST (pour que le contenu des variables ne s'affichent pas dans la barre d'adresse du navigateur lors du transfert) vers la page Moniteur.



- ◆ La page Moniteur commence par vérifier le contenu des variables envoyées par la page Connexion grâce à une requête SQL qui interroge les champs Utilisateur et MotDePasse de la table Identifiant de la Base de Données (ou BDD) afin de vérifier la validité des informations. Dans le cas d'identifiants non-valides, l'utilisateur est renvoyé sur la page connexion qui affiche un message d'erreur. Dans le cas contraire, une autre vérification s'effectue dans la BDD afin de savoir si l'utilisateur possède le statut d'administrateur ou pas. Si l'utilisateur a les droits administrateur, cela rajoute un bouton «Administrer» en haut à droite de la page Moniteur afin de pouvoir accéder à la page Administration. J'ai également créé deux listes déroulantes et deux calendriers pour pouvoir choisir le type de données et le module mesure désirés, sur une plage temporelle définie par une date de début et une date de fin. Après validation de ces données en cliquant sur le bouton «Envoyer», elles s'affichent sous la forme d'un graphique. J'ai aussi décidé de conserver les choix de l'utilisateur dans les listes déroulantes et calendriers de la page lors du rechargement de la page car au départ, ils se réinitialisaient, ce qui n'était pas pratique en terme d'utilisation.





- ◆ La Page d'Administration permet de gérer les utilisateurs, mots de passe et statuts administrateur. Ces trois plages de données sont affichées sous forme de tableau dans lequel il est possible de sélectionner une ligne qui remplit les zones de saisies modifiables situées en dessous afin de modifier, pour un utilisateur donné, le mot de passe qui lui est associé ou son statut (administrateur ou non) en cliquant sur le bouton «Ajouter/Modifier» ou de le supprimer en cliquant sur le bouton «Supprimer». Dans le cas où on veut créer un nouvel utilisateur, il suffit de remplir les trois zones de saisie et cliquer sur le bouton «Ajouter/Modifier». Cette page permet également de visualiser le journal des connexions du site (nom d'utilisateur, date et heure), de la plus récente à la plus ancienne.

Retour		
Nom	Mot de Passe	Administrateur
admin	mdp	Oui
Ted	Ted	Non
User	User	Non

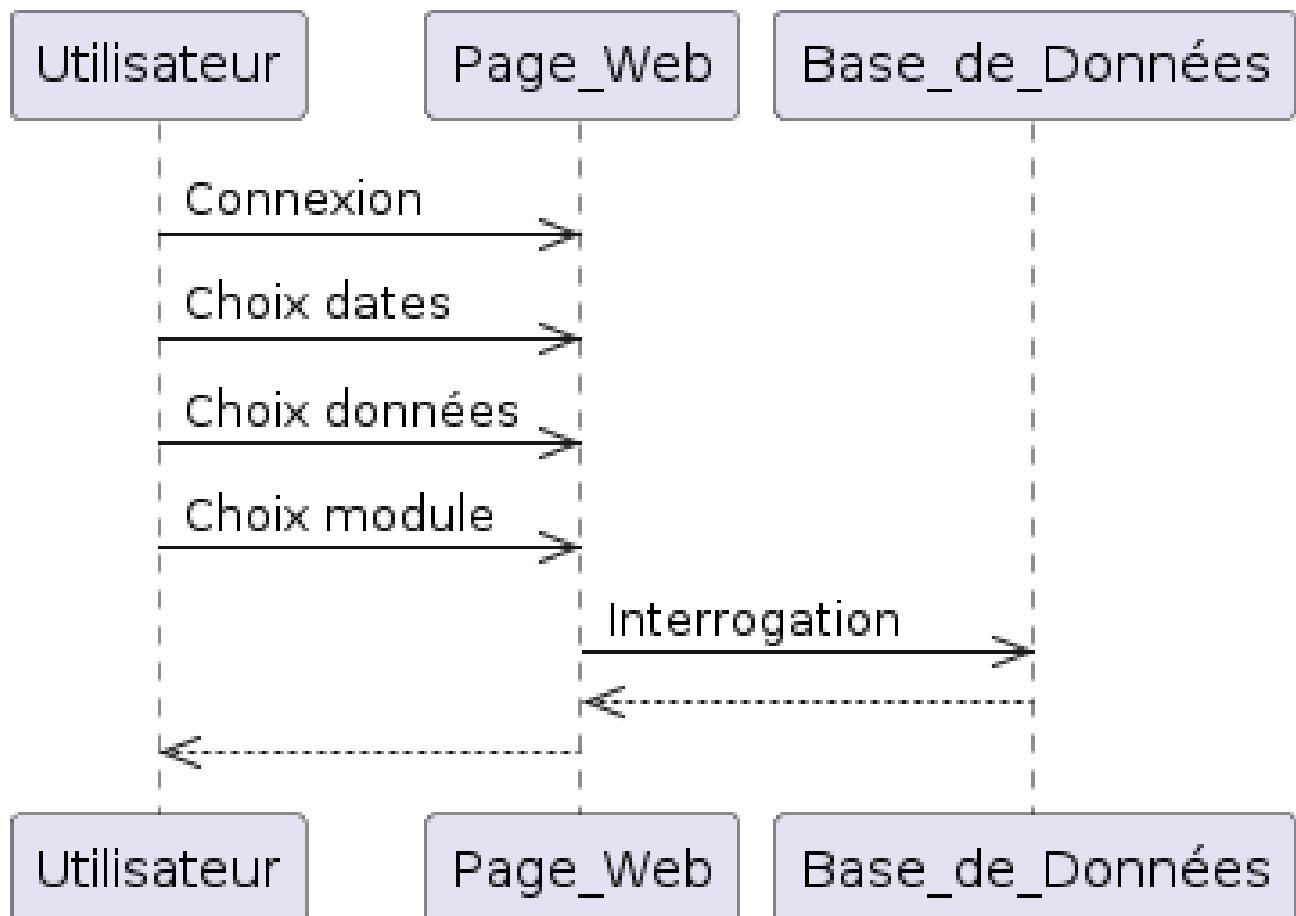
Nom	Mot de Passe	Administrateur	Supprimer
admin		Oui	
admin		Non	
admin		Non	
Ajouter/Modifier	Mot de Passe	Administrateur	
		<input type="radio"/> Oui <input type="radio"/> Non	
			Prendre une nouvelle capture d'écran

Nom	Date	Heure
admin	2024-05-13	11:54:57
admin	2024-05-13	10:57:45
admin	2024-05-13	10:51:19
Ted	2024-05-13	10:49:40
admin	2024-05-13	10:48:48
admin	2024-05-13	10:02:50
admin	2024-05-13	09:51:51
admin	2024-05-13	09:36:46
admin	2024-05-13	09:28:45
admin	2024-05-13	09:28:41
admin	2024-05-13	09:23:18
admin	2024-04-17	11:45:06
admin	2024-04-17	11:33:43

## Cahier de Recette

- Diagramme de Séquence Technicien 3 – Consultation d'un historique des consommations électriques par module mesure



▪ **Fiche de recette, Test de faisabilité et Tests unitaires**

**Fiche de Recette**

<input checked="" type="checkbox"/> <b>Recette globale</b> <input type="checkbox"/> <b>Test de faisabilité</b> <input type="checkbox"/> <b>Test unitaire</b> <input type="checkbox"/> <b>Test d'intégration</b>				
<b>Nom de l'étudiant :</b> Quentin LHOTE <b>Date :</b> 14.02.2024				
<b>Conditions de réalisation</b> <ul style="list-style-type: none"> <li>• Besoins matériels           <ul style="list-style-type: none"> <li>- un module mesure</li> <li>- un module relais</li> <li>- un PC</li> <li>- un Serveur local</li> </ul> </li> <li>• Besoins logiciels (nom et version)           <ul style="list-style-type: none"> <li>- un navigateur</li> <li>- une Base de Données MySQL</li> </ul> </li> </ul>				
<b>Scénario</b>				
ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Ouvrir un navigateur web (Chrome, Firefox, Edge...)	Avoir installé le module mesure, la Base de données et avoir conçu la page web		
2	Saisir l'IP (ici, <a href="http://172.16.150.1">http://172.16.150.1</a> ) et valider	IP correcte	La page de connexion s'affiche	
3	Rentrer ses identifiants (utilisateur et mot de passe) et cliquer sur envoyer	Identifiants valides	Une nouvelle page s'affiche sur laquelle on peut consulter la puissance (en Watt) et les consommations d'électricité (en kWh à la journée) sur une période désirée (grâce à des calendriers) et par module, sous forme de graphique	

## Fiche de Faisabilité

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale  
 Test de faisabilité  
 Test unitaire  
 Test d'intégration

Nom de l'étudiant : Quentin LHOTE

Date : 01.04.2024

Objectif du test : Vérifier que le graphique trouvé corresponde bien aux attentes du projet

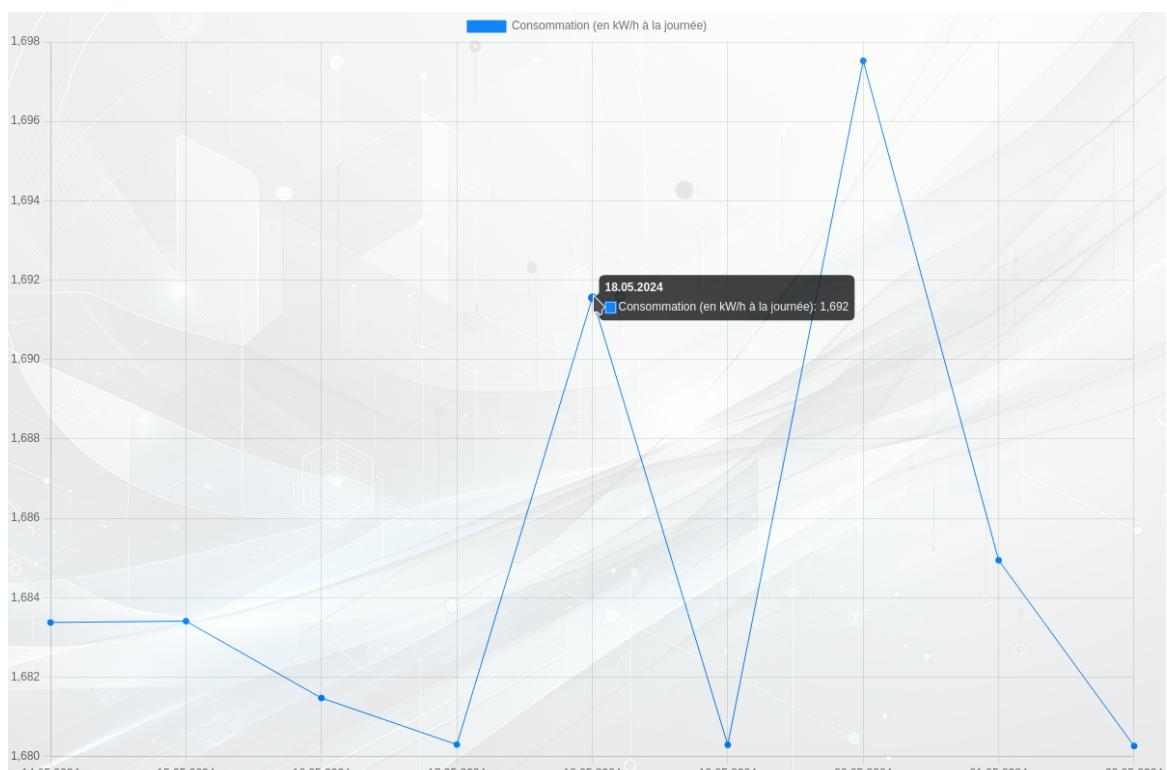
### Conditions de réalisation

- Besoins matériels
  - un PC
  - un Serveur local
- Besoins logiciels (nom et version)
  - un navigateur
  - une Base de Données MySQL
  - la bibliothèque Chart.js
- Noms des fichiers utilisés
  - moniteur.php

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Prospection sur internet, recherche de bibliothèques pour le graphique	Avoir bien défini les besoins en terme d'affichage des données	Obtenir un visuel des données pour pouvoir les transmettre à l'utilisateur	OK
2	Effectuer un affichage, s'assurer qu'il puisse s'adapter aux besoins du projet	Avoir trouvé des bibliothèques permettant de créer des graphiques sur une page web et effectué un premier tri	Que l'outil affiche les données de façon très comprehensible pour l'utilisateur et qu'il soit adaptable au projet (on doit pouvoir mettre des dates à l'axe des abscisses)	OK

### Annexes (captures d'écran et schémas)



## Fiche de Test 1

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

Nom de l'étudiant : Quentin LHOTE

Date : 06.02.2024

Objectif du test : Vérification de la bonne récupération des données saisies dans le formulaire de la page connexion.html

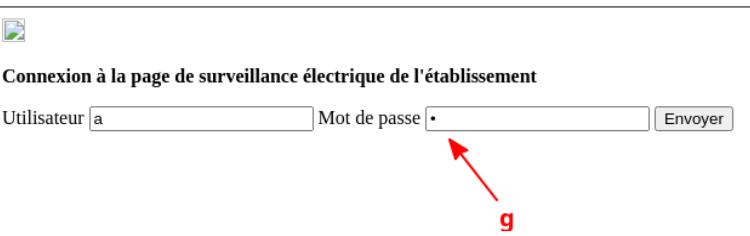
### Conditions de réalisation

- Besoins matériels
  - un PC
  - un Serveur local
- Besoins logiciels (nom et version)
  - un navigateur internet
- Noms des fichiers utilisés
  - connexion.html
  - moniteur.php

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Remplissage du formulaire	Nom utilisateur et mot de passe		OK
2	Passage de la page connexion à la page moniteur avec le bouton envoyer	Cliquer sur le bouton Envoyer	Changement de page	OK
3	Utilisation de la fonction echo pour afficher les données		Affichage sur la page moniteur des données du formulaire	OK

### Annexes (captures d'écran et schémas)



Connexion à la page de surveillance électrique de l'établissement

Utilisateur: a Mot de passe: g Envoyer

Dans le formulaire précédent, vous avez fourni les informations suivantes :

Utilisateur: a  
Mot de passe : g

## Fiche de Test 2

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

Nom de l'étudiant : Quentin LHOTE

Date : 07.02.2024

Objectif du test : Test d'interrogation de la BDD MySQL et vérification d'identifiants de connexion

### Conditions de réalisation

- Besoins matériels
  - un PC
  - un Serveur local
- Besoins logiciels (nom et version)
  - un navigateur
  - une Base de Données MySQL
- Noms des fichiers utilisés
  - connexion.html
  - moniteur.php

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Cliquer sur le bouton Envoyer	Données du formulaire de connexion valides ou erronées	Changement de page et interrogation de la BDD	OK
2	Utilisation de la fonction echo		Affichage de l'Utilisateur et de son mot de passe à l'écran en cas d'identification réussie	OK

### Annexes (captures d'écran et schémas)

ID	Utilisateur	MotDePasse	Admin
1	admin	mdp	1
2	b	b	0

- adminmdp

Dans le formulaire précédent, vous avez fourni les informations suivantes :

Utilisateur: admin  
Mot de passe : mdp

## Fiche de Test 3

- Recette globale : d'un cas d'utilisation et/ou d'une fonctionnalité globale
- Test de faisabilité
- Test unitaire
- Test d'intégration

Nom de l'étudiant : Quentin LHOTE

Date : 15.04.2024

Objectif du test : Vérification des dates choisies par l'utilisateur

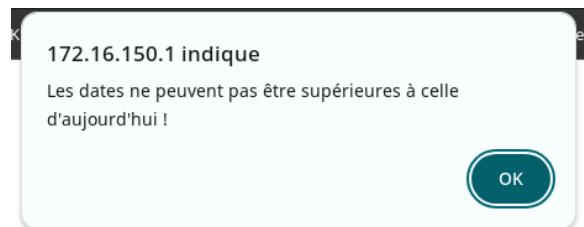
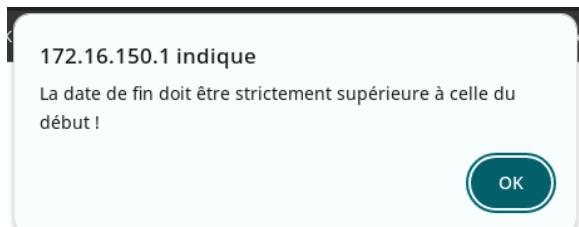
### Conditions de réalisation

- Besoins matériels
  - un PC
  - un Serveur local
- Besoins logiciels (nom et version)
  - un navigateur
  - une Base de Données MySQL
- Noms des fichiers utilisés
  - moniteur.php

### Scénario

ID	Démarche / Opération	Données en entrée / Condition initiale	Comportement / Résultat attendu	Status OK / NOK
1	Choix des dates de début et de fin pour l'affichage des données dans le graphique.	Les dates doivent être choisies dans les 2 formulaires de type calendrier.	Affichage d'un message pop-up d'erreur si la date de fin est inférieure ou égale à celle du début ou si une des dates est supérieure à celle du jour de consultation.	OK

### Annexes (captures d'écran et schémas)



## ▪ Problèmes Rencontrés et Solutions

### Conversion de variables Php en variables Javascript

Le graphique que j'utilise dans la page Moniteur est généré en JavaScript, donc côté client (sur l'ordinateur de l'utilisateur), alors que les données nécessaires à son affichage sont stockées dans la BDD, donc côté serveur et sont transmises en réponse à une requête SQL sous la forme d'un dictionnaire (donc clé et valeur) en langage Php grâce à la fonction mysqli\_fetch\_assoc. Il a donc fallu que je convertisse ces données en Json par la fonction json\_encode puis qu'elles soient stockées dans des variables JavaScript grâce à l'instruction echo pour qu'elles soient exploitées dans le graphique. (voir le code de l'annexe 3.2.2 moniteur.php, l.181 et l.207)

### Affichage des dates dans le graphique

Au départ, les étiquettes de l'axe des abscisses ne prenaient en compte que des chiffres mais j'avais besoin de caractères pour pouvoir afficher les dates de mes mesures. Pour cela, j'ai créé une fonction liste Dates\_fr qui stocke dans un tableau toutes les dates entre celle de début et celle de fin, choisies par l'utilisateur, grâce à une boucle qui utilise la bibliothèque strtotime. Ensuite, j'ai utilisé la même méthode que ci-dessus afin de convertir mon tableau du Php vers le JavaScript. Cela a permis d'avoir une bonne lisibilité du graphique. (voir le code de l'annexe 3.2.2 moniteur.php, l.116 à 126, l.182, l.208 et l.216)

### Récupération de données moyennes

Le module mesure envoie, via le relais, une nouvelle mesure toutes les minutes. Il était donc impossible de toutes les afficher car cela rendait le graphique complètement illisible... J'ai donc décidé, en accord avec mon équipe, de limiter l'affichage à une donnée par jour en faisant une moyenne (que ce soit pour la Puissance ou la Consommation). Pour ce faire, j'ai utilisé l'expression AVG dans ma requête SQL afin d'obtenir les données jour par jour. J'ai donc obtenu des valeurs exprimées en moyennes journalières. (voir le code de l'annexe 3.2.2 moniteur.php, l.167 à 168)

### Erreurs liées aux utilisateurs

Il a fallu gérer les erreurs liées aux utilisateurs afin que la date de début ne soit jamais supérieure à la date de fin (cela n'aurait aucune logique) ni supérieure ou égale à la date d'aujourd'hui (vu que le graphique affiche des moyennes journalières, il faut donc que la journée soit terminée pour obtenir une moyenne pertinente). J'ai donc décidé de bloquer ces choix en utilisant des alertes sous forme de pop-up JavaScript. (voir le code de l'annexe 3.2.2 moniteur.php, l.142 à 153)

Afin de prévenir l'utilisateur en cas d'erreur d'identification, j'ai mis en place l'apparition d'un message d'erreur sur la page de connexion. Pour ce faire, j'ai utilisé le principe de session en utilisant des variables super globales (\$\_SESSION['id\_correct'] et \$\_SESSION['jeton']). J'ai utilisé ce procédé pour pouvoir migrer une variable des pages Connexion à Moniteur et inversement. (voir les codes des annexes 3.2.1 connexion.php, l.3 et l.26 à 31 et 3.2.2 moniteur.php, l.4 à 29)

## **Conclusion Personnelle**

Après deux années scolaires où j'ai appris à coder principalement en langage C++ lors de mes cours et en Python lors de mon stage, ce projet m'a permis de découvrir trois nouveaux langages: l'HTML, le Php et le CSS. J'ai aussi découvert les univers de l'élaboration de site internet et du travail en équipe avec mes deux collègues techniciens. La principale difficulté rencontrée a donc été de partir de mon niveau de débutant dans ces langages pour arriver à finaliser le projet qui m'a été confié.

Je tiens à remercier Lukas et Sylvain pour l'implication et la bonne humeur dont ils ont fait preuve tout au long de ce projet ainsi que mes professeurs pour leur écoute et le soutien qu'il m'ont apporté.

## ANNEXE 1.1

# Documentation Technique pour l'Application de Mesure Énergétique

Cette documentation est conçue pour guider n'importe quel utilisateur dans la configuration et l'utilisation de l'application de mesure énergétique utilisant l'ESP32 et le capteur PZEM004Tv30. Le but est de mesurer et envoyer les données de puissance énergétique via un réseau WiFi à un serveur distant.

### 1. Matériel Requis

- **Carte ESP32** : Un microcontrôleur avec capacités WiFi intégrées.
- **Capteur PZEM004Tv30** : Capteur de mesure de la tension, du courant et de la puissance électrique.
- **Câbles de connexion** : Pour connecter le capteur à l'ESP32.
- **Source d'alimentation** : Pour l'ESP32 et le capteur.

### 2. Configuration de l'Environnement

- **Ordinateur** avec un IDE compatible (ex. Arduino IDE) pour programmer l'ESP32.
- **Connexion Internet** pour télécharger les bibliothèques nécessaires et pour la connexion WiFi de l'ESP32.

### 3. Bibliothèques Nécessaires

- **PZEM004Tv30.h** : Pour interagir avec le capteur de puissance.
- **HardwareSerial.h** : Pour la communication série entre l'ESP32 et le capteur.
- **WiFi.h** et **WiFiUdp.h** : Pour les fonctionnalités WiFi et la transmission des données par UDP.

```
#include <PZEM004Tv30.h>
• #include <HardwareSerial.h>
• #include <WiFi.h>
• #include <WiFiUdp.h>
```

### 4. Installation et Configuration

#### 1. Installation des Bibliothèques :

- Ouvrez l'Arduino IDE et allez dans le gestionnaire de bibliothèques.
- Installez les bibliothèques **PZEM004Tv30**, **WiFi**, et **WiFiUDP**.

#### 2. Connexion des Composants :

- Connectez le capteur PZEM004Tv30 à l'ESP32 en utilisant les broches RX et TX spécifiées (16 et 17 dans ce cas).

- Assurez-vous que toutes les connexions sont sécurisées pour éviter tout dysfonctionnement.
- ```
#define RS232_RX_PIN 16
#define RS232_TX_PIN 17
PZEM004Tv30 pzem(Serial2, RS232_RX_PIN, RS232_TX_PIN);
```

## Configuration de l'Environnement de Développement

- **Installation de l'Arduino IDE** : Téléchargez et installez l'Arduino IDE depuis le site officiel Arduino. Cette plateforme sera utilisée pour programmer l'ESP32.
- **Configuration de l'IDE** : Ajoutez le support pour l'ESP32 à l'IDE Arduino en suivant les instructions disponibles sur [le GitHub ESP32](#). Cela inclut l'ajout de l'URL de gestionnaire de cartes et l'installation du package ESP32 via le gestionnaire de cartes.

## 5. Configuration du Programme

- **Configurer les Identifiants WiFi** : Remplacez les valeurs des variables `ssid` et `password` avec les identifiants de votre réseau WiFi.
- **Configurer l'Adresse du Serveur** : Modifiez `serverIP` et `serverPort` pour correspondre à votre configuration de serveur.

```
WiFiUDP udp;
const char* ssid = "ProjetEnergie"; // Nom du réseau WIFI
const char* password = "ProjetEnergie"; // Mot de passe du réseau WIFI
String moduleID = "0001"; // Identifiant du module de mesure
IPAddress serverIP(192.168.1.255); // Adresse IP du serveur
unsigned int serverPort = 12345; // Port du serveur
```

## 6. Explication du Code

- **Fonction setup()** :
  - Initialisation des communications série pour les diagnostics et pour communiquer avec le capteur.
  - Connexion au réseau WiFi et gestion des tentatives de connexion.
- **Fonction loop()** :
  - Lecture des mesures de tension, courant, et puissance via le capteur.
  - Validation des mesures pour s'assurer qu'elles ne sont pas erronées.
  - Préparation et envoi des données au serveur via UDP, avec gestion des erreurs de transmission.

7.

## **Compilation et Téléversement**

- Connectez l'ESP32 à votre ordinateur via USB.
- Sélectionnez le bon type de carte et le port dans l'Arduino IDE.
- Compilez et téléversez le programme sur l'ESP32.

## **8. Surveillance et Dépannage**

- **Surveillance** : Utilisez le moniteur série de l'Arduino IDE pour observer les logs et vérifier que l'application fonctionne correctement.
- **Dépannage** :
  - Si la connexion WiFi échoue, vérifiez les identifiants et la portée du signal WiFi.
  - Si les mesures sont erronées, vérifiez les connexions du capteur et recalibrez si nécessaire.

## ANNEXE 1.2

### Code Source Complet

- Pour intégrer votre projet et permettre une reproduction fidèle de l'application, le code source complet est fourni ci-dessous avec des commentaires détaillés pour faciliter la compréhension et la modification selon les besoins spécifiques de chaque utilisateur.
- ```
#include <PZEM004Tv30.h>          // Bibliothèque pour le capteur PZEM-004T v3.0
```
- ```
#include <HardwareSerial.h>      // Bibliothèque pour la communication série
#include <WiFi.h>                  // Bibliothèque pour la gestion du WiFi
#include <WiFiUdp.h>                // Bibliothèque pour les communications UDP
```
- ```
// Définition des broches pour la communication RS232
#define RS232_RX_PIN 16
#define RS232_TX_PIN 17
```
- ```
// Crédit d'un objet PZEM en utilisant le Serial2
PZEM004Tv30 pzem(Serial2, RS232_RX_PIN, RS232_TX_PIN);
WiFiUDP udp; // Objet pour la communication UDP
```
- ```
// Informations du réseau WiFi
const char* ssid = "ProjetEnergie"; // Nom du réseau WIFI
const char* password = "ProjetEnergie"; // Mot de passe du réseau WIFI
```
- ```
String moduleID = "0001";           // Identifiant du module de mesure (peut être modifier)
IPAddress serverIP(192, 168, 1, 255); // Adresse IP du serveur
unsigned int serverPort = 12345;       // Port du serveur
```
- ```
void setup() {
    Serial.begin(115200); // Commence la communication série pour le débogage
    Serial2.begin(9600, SERIAL_8N1, RS232_RX_PIN, RS232_TX_PIN); // Configuration de la communication série avec le PZEM
```
- ```
// Tentative de connexion au réseau WiFi
    WiFi.begin(ssid, password);
    int maxAttempts = 20; // Nombre maximal de tentatives de connexion
    while (WiFi.status() != WL_CONNECTED && maxAttempts-- > 0) {
        // Attendre la connexion
        delay(500);
        Serial.print(".");
    }
    if (WiFi.status() == WL_CONNECTED) {
        Serial.print("Connecté au WiFi, adresse IP attribuée: ");
        Serial.println(WiFi.localIP()); // Affiche l'adresse IP si connecté
    } else {
        Serial.println("\nÉchec de la connexion au WiFi. Vérifiez vos identifiants.");
        return; // Sortie de la fonction si échec de connexion
    }
    Serial.println("\nConnecté au WiFi!");
```
- ```
void loop() {
    // Lecture des valeurs de tension, courant et puissance
```

```

•     float tension = pzem.voltage();
•     float courant = pzem.current();
•     float puissance = pzem.power();
•
•     // Vérification des données reçues
•     if (isnan(tension) || isnan(courant) || isnan(puissance)) {
•         Serial.println("Erreur de lecture du capteur!");
•         delay(5000); // Pause avant la prochaine lecture
•         return;
•     }
•
•     // Préparation des données à envoyer
•     String dataString = "N=" + moduleID + "P=" + String(puissance) + "\n";
•     Serial.println("Envoi en cours");
•     if (udp.beginPacket(serverIP, serverPort) == 1) {
•         udp.print(dataString);
•         if (udp.endPacket() != 1) {
•             Serial.println("Erreur d'envoi des données via UDP.");
•         } else {
•             Serial.println("Données envoyées via UDP: " + dataString);
•         }
•     } else {
•         Serial.println("Erreur de début de paquet UDP.");
•     }
•
•     delay(60000); // Délai entre les envois de données de 60 secondes
• }
```

Ce code constitue une base complète pour la mise en place d'une station de mesure de la puissance électrique utilisant l'ESP32 et le capteur PZEM004Tv30, avec des fonctionnalités de communication réseau via WiFi et UDP. Les utilisateurs sont encouragés à modifier les paramètres de réseau et les broches selon leur configuration spécifique pour garantir la compatibilité et la performance optimales de leur application.

## ANNEXE 1.3

**Pour installer Arduino IDE sur un système Linux et configurer un environnement prêt pour le développement, voici un guide détaillé étape par étape.**

### Prérequis

Avant de commencer l'installation, assurez-vous que votre système est à jour. Ouvrez un terminal et exécutez les commandes suivantes pour mettre à jour votre système :

```
sudo apt update  
sudo apt upgrade
```

### Étape 1: Téléchargement de l'Arduino IDE

- Visitez le Site Officiel** : Rendez-vous sur le site officiel d'Arduino à cette adresse :  
<https://www.arduino.cc/en/software>.
- Sélectionnez la Version pour Linux** : Vous trouverez des options pour différentes versions de Linux (32 bits, 64 bits, ARM). Choisissez celle qui correspond à votre architecture système.
- Téléchargez le fichier tar** : Cliquez sur le lien de téléchargement pour votre version spécifique et enregistrez le fichier `.tar.xz` dans votre dossier de téléchargements.

### Étape 2: Installation de l'Arduino IDE

#### 1. Extraction de l'Archive :

- Ouvrez un terminal et naviguez vers le dossier où vous avez téléchargé l'archive. Typiquement, ce sera `~/Téléchargements` ou `~/Downloads` :
- `cd ~/Downloads`
- Extrayez l'archive en utilisant la commande tar :  
`tar -xf arduino-x.x.x-linux64.tar.xz`

Remplacez `arduino-x.x.x-linux64.tar.xz` par le nom de fichier que vous avez téléchargé.

- **Déplacement du Dossier Arduino :**
- Il est recommandé de déplacer le dossier extrait dans un emplacement approprié. /opt est un bon choix pour les applications logicielles sur de nombreux systèmes Linux :

- `sudo mv arduino-x.x.x /opt/arduino`

Remplacez `arduino-x.x.x` par le nom du dossier extrait.

- **Exécution du Script d'Installation :**

- Changez de répertoire pour aller dans le dossier Arduino :

- `cd /opt/arduino`

- Exécutez le script d'installation :

- `sudo ./install.sh`

## Étape 3: Installation des Dépendances

L'Arduino IDE nécessite l'installation de Java et de certains pilotes pour communiquer avec les cartes Arduino :

1. **Installation de Java :**

- `sudo apt install default-jdk`

- **Ajout de l'Utilisateur au Groupe Dialout** (pour permettre à l'IDE d'accéder aux ports série) :

`2. sudo usermod -a -G dialout $USER`

3. **Configuration des Permissions UDEV** (si nécessaire pour des cartes spécifiques comme les modèles Arduino Nano) :

- Vous pouvez trouver les règles UDEV nécessaires sur le forum Arduino ou dans la documentation spécifique à votre modèle de carte.

## Étape 4: Lancement de l'Arduino IDE

- Vous pouvez maintenant lancer l'Arduino IDE directement depuis le terminal en exécutant :
- `/opt/arduino/arduino`

## **Étape 5: Configuration Post-Installation**

Après le lancement de l'Arduino IDE :

### **1. Installer des Bibliothèques et des Cartes Supplémentaires :**

- Utilisez le Gestionnaire de Cartes (dans le menu Outils > Type de carte > Gestionnaire de cartes) pour installer le support pour des cartes supplémentaires.
- Utilisez le Gestionnaire de Bibliothèques (dans le menu Croquis > Inclure une bibliothèque > Gérer les bibliothèques) pour installer des bibliothèques nécessaires à vos projets.

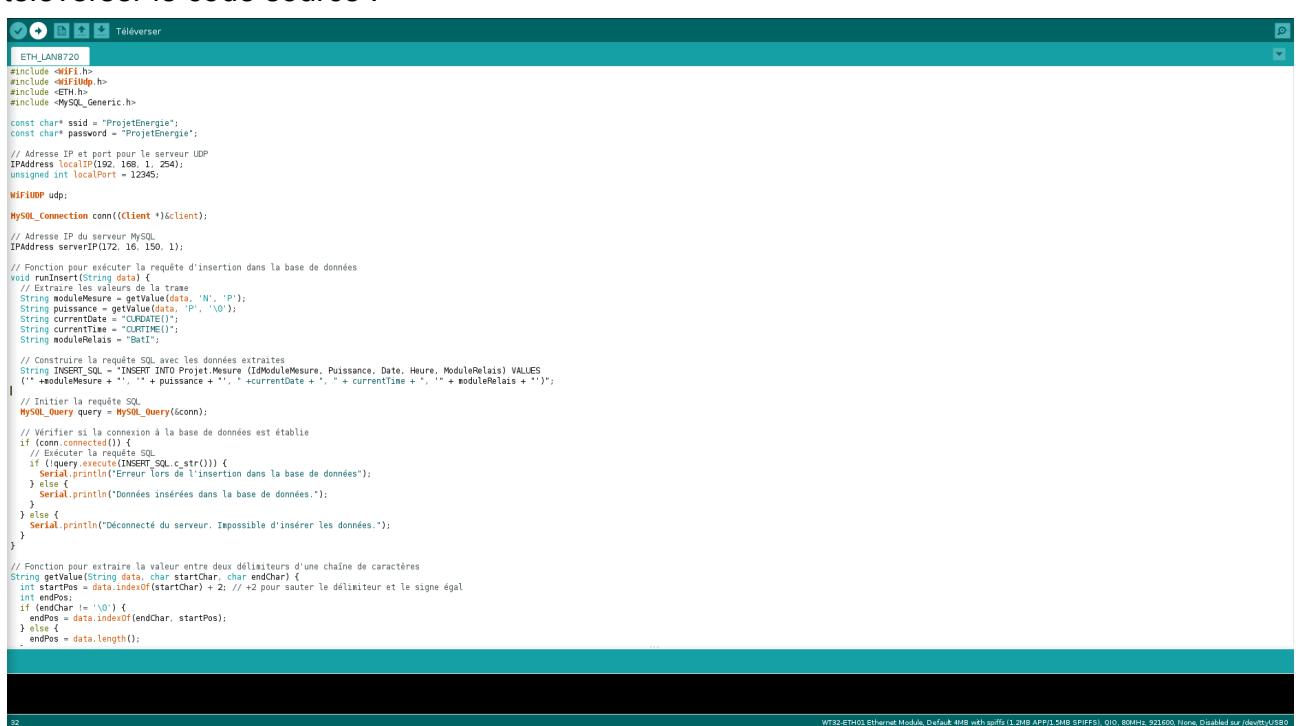
## 2.0

- **Manuel d'Utilisation :**

### Création du module relais :

1. Pour commencer il faut le connecter au programmeur afin de pouvoir téléverser le code. C'est expliqué plus haut. (Annexe 2.1)
2. Il faut changer l'information suivante avant de continuer :  
String moduleRelais = "Bati" ;  
changer "Bati" par le nom de votre module Relais .  
/!\ Avoir 2 modules relais ne pose pas de problèmes technique mais peut être plus compliqué à gérer en cas d'erreur

### 3. téléverser le code source :



```
#include <WiFi.h>
#include <WiFiUdp.h>
#include <ETH.h>
#include <MySQL_General.h>

const char* ssid = "ProjetEnergie";
const char* password = "ProjetEnergie";

// Adresse IP et port pour le serveur UDP
IPAddress localIP(192, 168, 1, 254);
unsigned int localPort = 12345;

WiFiUDP udp;

MySQL_Connection conn((Client *)&client);

// Adresse IP du serveur MySQL.
IPAddress serverIP(172, 16, 150, 1);

// Fonction pour exécuter la requête d'insertion dans la base de données
void runInsert(String data) {
    // Extraire les valeurs de la trame
    String moduleRelais = getValue(data, 'N', 'P');
    String puissance = getValue(data, 'P', '\0');
    String currentDate = "NOW()";
    String currentTime = "CURRENTTIME";
    String moduleRelais = "Bati";

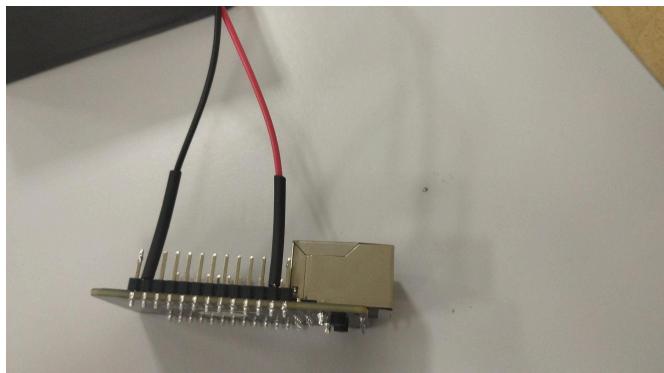
    // Construire la requête SQL avec les données extraites
    String INSERT_SQL = "INSERT INTO Projet.Mesure (IDModuleMesure, Puissance, Date, Heure, ModuleRelais) VALUES ";
    INSERT_SQL += moduleRelais + ", " + puissance + ", " + currentDate + ", " + currentTime + ", " + moduleRelais + ")";

    // Initier la requête SQL
    MySQL_Query query = MySQL_Query(&conn);

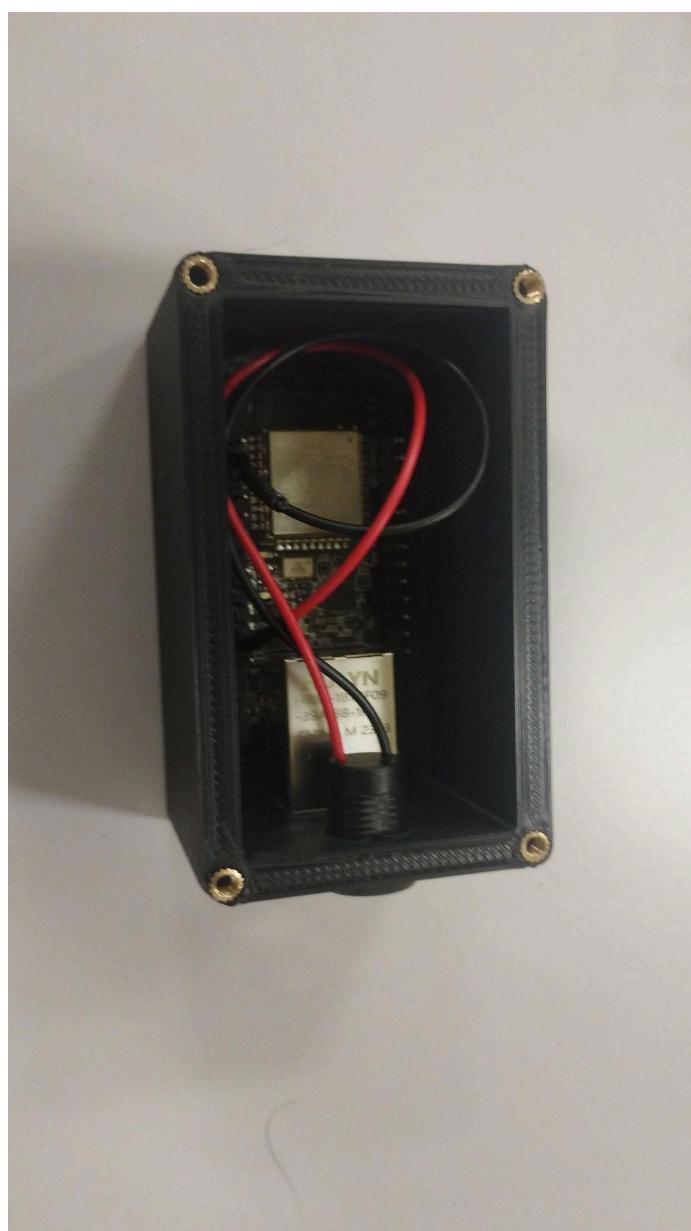
    // Vérifier si la connexion à la base de données est établie
    if (conn.isOpened()) {
        // Exécuter la requête SQL
        if (query.execute(INSERT_SQL.c_str())) {
            Serial.println("Erreur lors de l'insertion dans la base de données");
        } else {
            Serial.println("Données insérées dans la base de données.");
        }
    } else {
        Serial.println("Déconnecté du serveur. Impossible d'insérer les données.");
    }
}

// Fonction pour extraire la valeur entre deux délimiteurs d'une chaîne de caractères
String getValue(String data, char startChar, char endChar) {
    int startPos = data.indexOf(startChar) + 2; // +2 pour sauter le délimiteur et le signe égal
    int endPos;
    if (endChar != '\0') {
        endPos = data.indexOf(endChar, startPos);
    } else {
        endPos = data.length();
    }
}
```

4. Souder la prise USB C sur le WT32 après l'avoir mis dans le boîtier : (Annexe 2.2)



5. mettre les inserts fileté :



6. fermer le boîtier :

7. brancher :

Pour le brancher il suffit de mettre et dans cet ordre de préférence , le câble RJ45 et ensuite l'usb C.

8. vérifier le fonctionnement :

Pour vérifier le bon fonctionnement il suffit de prendre votre téléphone et de vérifier que le réseau wifi est bien créé et visible.

## installation serveur :

1. installation physique :

Pour l'installation physique il ma suffit de le glisser dans la baie et de mettre le câble RJ45 et les doubles alimentations .

2. installation OS :

Pour installer l'OS j'ai téléchargé une ISO de Ubuntu Server et aussi Rufus , j'ai ensuite créer un clé bootable et ensuite j'ai simplement suivi les indication à l'écran , mais au moment de choisir si on veut une installation "FULL" ou "MINIMIZED" j'ai choisi la deuxième option car elle est plus légère et que j'ai absolument pas besoin de tout ce qui est installé dans la première option. (Annexe 2.3)

3. installation applications :

Pour installer le serveur WEB j'ai simplement suivi le protocole d'installation fournie en cours et présent en annexe.

Pour le surplus j'ai eu besoin de "UFW" qui sert de pare-feu et aussi de "nano" qui me permet en interface ligne de commande de modifier des fichiers texte .

4. création BDD :

Pour créer la base de données nous avons juste regarder chaque type de valeurs que nous voulions stocker ensuite nous leur avons créer une colonne chacun en fonction de quelle type de données il s'agissait et voilà la BDD . Dans une question de rapidité et de praticité j'ai utilisé PhpMyAdmin pour la créer , je me suis donc passer des commandes SQL.

## 5. IP fixe :

Pour le bon fonctionnement du système nous avons besoin que le serveur possède un ip fixe , seulement étant limité dans le réseau de la section , c'est le prof qui nous l'attribue directement sur le DHCP.

### 2.1 code source

```
#include <WiFi.h>
#include <WiFiUdp.h>
#include <ETH.h>
#include <MySQL_Generic.h>

const char* ssid = "ProjetEnergie";
const char* password = "ProjetEnergie";

// Adresse IP et port pour le serveur UDP
IPAddress localIP(192, 168, 1, 254);
unsigned int localPort = 12345;

WiFiUDP udp;

MySQL_Connection conn((Client *)&client);

// Adresse IP du serveur MySQL
IPAddress serverIP(172, 16, 150, 1);

// Fonction pour exécuter la requête d'insertion dans la base de données
void runInsert(String data) {
    // Extraire les valeurs de la trame
    String moduleMesure = getValue(data, 'N', 'P');
```

```

String puissance = getValue(data, 'P', '\0');

String currentDate = "CURDATE()";

String currentTime = "CURTIME()";

String moduleRelais = "Bat1";

// Construire la requête SQL avec les données extraites

String INSERT_SQL = "INSERT INTO Projet.Mesure (IdModuleMesure, Puissance, Date, Heure,
ModuleRelais) VALUES
(" +moduleMesure + ", " + puissance + ", " +currentDate + ", " + currentTime + ", " + moduleRelais
+ "")";

// Initier la requête SQL

MySQL_Query query = MySQL_Query(&conn);

// Vérifier si la connexion à la base de données est établie

if (conn.connected()) {

    // Exécuter la requête SQL

    if (!query.execute(INSERT_SQL.c_str())) {

        Serial.println("Erreur lors de l'insertion dans la base de données");

    } else {

        Serial.println("Données insérées dans la base de données");

    }

} else {

    Serial.println("Déconnecté du serveur. Impossible d'insérer les données.");

}

}

// Fonction pour extraire la valeur entre deux délimiteurs d'une chaîne de caractères

String getValue(String data, char startChar, char endChar) {

```

```

int startPos = data.indexOf(startChar) + 2; // +2 pour sauter le délimiteur et le signe égal

int endPos;

if (endChar != '\0') {

    endPos = data.indexOf(endChar, startPos);

} else {

    endPos = data.length();

}

return data.substring(startPos, endPos);

}

void setup() {

Serial.begin(115200);

// Créer le point d'accès Wi-Fi

WiFi.softAP(ssid, password);

WiFi.softAPConfig(localIP, localIP, IPAddress(255, 255, 255, 0));



// Démarrer le serveur UDP

udp.begin(localPort);




Serial.print("Adresse IP du point d'accès Wi-Fi: ");

Serial.println(localIP);




// Initialiser et démarrer le port Ethernet

if (!ETH.begin()) {

    Serial.println("Échec de l'initialisation de l'Ethernet.");

    return;

}

```

```

Serial.println("Ethernet démarré.");

// Connexion au serveur MySQL

if (conn.connect(serverIP, 3306, "CodePhp", "CodePhp")) {

    Serial.println("Connexion à la base de données MySQL établie.");

    } else {

        Serial.println("Échec de la connexion à la base de données MySQL.");

    }

}

void loop() {

    // Vérifier la réception de données UDP

    int packetSize = udp.parsePacket();

    if (packetSize) {

        // Lire les données UDP dans un tampon

        char packetBuffer[255];

        udp.read(packetBuffer, packetSize);

        packetBuffer[packetSize] = '\0'; // Ajouter le caractère de fin de chaîne

        // Afficher les données reçues sur la console série

        Serial.print("Données UDP reçues: ");

        Serial.println(packetBuffer);

        // Insérer les données dans la base de données

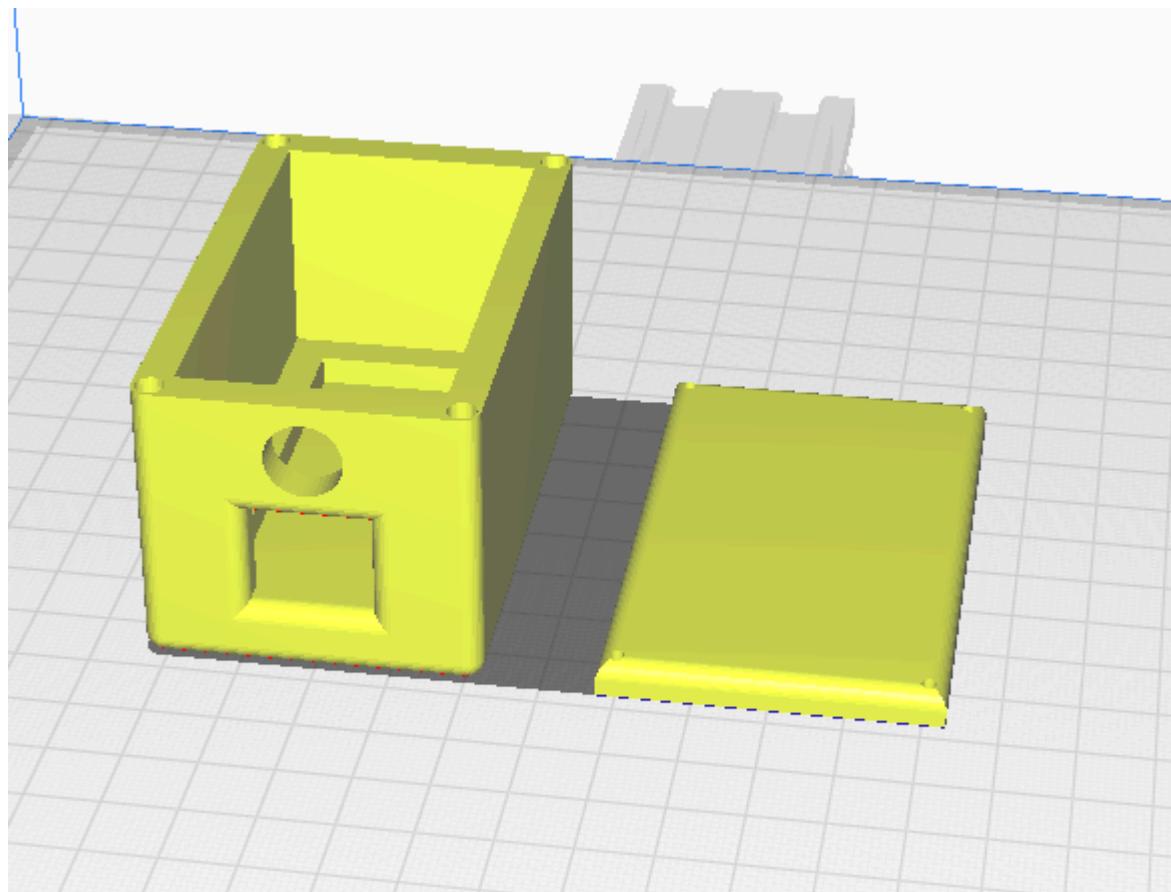
        runInsert(packetBuffer);

    }

}

```

## 2.2 Module Relais



## 2.3 Installation serveur web

## Installer les services apache, php et mariadb sur Debian

Durée : 2 h

Pré-requis : Maîtriser les commandes de base sous Linux.

Testé avec :  
RPI OS du 11janv. 2021  
Debian 11 et 12 (09-2023)



### Installation d'Apache

Installer le paquet :

```
# apt update
# apt install apache2
```

Pour voir le numéro de la version installée :

```
# apache2 -v
```

S'assurer que le service est fonctionnel :

```
# systemctl status apache2 ou bien      # service apache2 status
```

S'assurer que le serveur écoute sur le port 80 :

```
$ ss -prunat
```

Tester l'accès au port 80 (en local ou depuis une machine distante) :

```
$ nc -z adresseIP 80; echo $? # doit renvoyer 0
```

Si problème, vérifiez que le pare-feu autorise bien le port 80 en TCP.

Si on dispose d'un navigateur web, tester l'accès local :

<http://127.0.0.1/>

Si on est en ligne de commande, utiliser la commande :

```
$ wget -o recu.html http://127.0.0.1
```

Puis, observer le contenu du fichier *recu.html*

Tests facultatifs

Donner les droits qui permettront de facilement modifier les fichiers hébergés :

```
# chown -R user:www-data /var/www/html/
# chmod -R 770 /var/www/html/
```

On remplace *user* par le nom d'utilisateur.

Vérifier que l'utilisateur concerné est bien membre du groupe www-data :

```
$ groups user
```

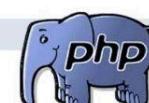
Si ce n'est pas le cas, ajouter l'utilisateur comme membre du groupe www-data, puis redémarrer :

```
# usermod -aG www-data user
# reboot
```

Pour que le service soit lancé à chaque démarrage de l'OS (c'est normalement déjà le cas par défaut) :

Si SystemV utilisé :      # systemctl enable apache2

Si InitV utilisé :      # update-rc.d apache2 defaults



### Installation de php

Installer (il est en version 8.2.7 en septembre 2023) :

```
# apt install php php-fpm
```

php est un méta paquet qui installe notamment libapache2-mod-php8.2

Pour voir le numéro de la version installée :

```
# php -v
```

Activer PHP FPM dans Apache, en exécutant les 2 commandes suivantes :

```
# a2enmod proxy_fcgi setenvif           # a2enconf php8.2-fpm
```

Puis, relancer le service apache2 :

```
# systemctl restart apache2 ou bien      # service apache2 restart
```

Remplacer le fichier *index.html* par un fichier nommé *index.php* (toujours dans /var/www/html/).

Placer le contenu suivant dans le fichier *index.php* : <?php phpinfo(); ?>

Dans le navigateur, recharger la page web. Les informations de php doivent apparaître.



## Installation de MariaDB (le fork communautaire de MySQL)

Installer les paquets :

```
# apt install mariadb-server php-mysql
```

*mariadb-server est le serveur du SGBD.  
php-mysql permet de faire le lien  
entre php et mysql.*

Relancer Apache et démarrer le serveur *mariadb* :

```
# systemctl restart apache2    ou bien    # service apache2 restart  
# systemctl start mariadb     ou bien    # service mariadb start
```

Améliorer la sécurité en utilisant l'outil suivant :

```
# mariadb-secure-installation  
Enter current password for root (enter for none):  
Taper Entrée, car il n'y a pas encore de mot de passe root.  
Switch to unix_socket authentication [Y/n] n  
Non, car on veut un accès par mot de passe natif, pas par unix_socket.  
Change the root password? [Y/n] Y  
Oui, car ça permet de définir un mot de passe root.  
$ mariadb -u root -p devient possible.  
Remove anonymous users? [Y/n] Y  
Oui, car on ne veut pas autoriser l'accès à n'importe qui.  
Disallow root login remotely? [Y/n] Y  
Oui, pour empêcher l'accès root distant.  
Remove test database and access to it? [Y/n] n  
Non, sauf si on veut supprimer la base test.  
Reload privilege tables now? [Y/n] y  
Oui, pour recharger la table des priviléges (prendre en compte les modifications).
```

Vérifier que *mariadb* fonctionne localement en se connectant au serveur :

```
# mariadb
```

On remarque que le prompt a changé : *MariaDB [(none)]>*

On peut afficher les comptes d'utilisateurs : *SELECT host,user,password FROM mysql.user ;*

Il doit y avoir un compte *root* qui permet un accès local seulement.

On peut afficher les bases de données : *SHOW DATABASES ;*

On peut sortir de *mariadb* : *exit*

Attention, ne pas faire d'erreur de saisie !

## Créer un nouveau compte utilisateur avec accès distant

Dans la console MySQL (précédemment ouverte) :

On recrée un nouvel utilisateur 'bob' en personnalisant le mot de passe (ici 'pass') :

```
MariaDB [(none)]> CREATE USER 'bob'@'%' IDENTIFIED BY 'pass';
```

Remarque : On peut limiter à un accès local en remplaçant % par localhost

Chaque commande  
doit répondre par  
Query OK

On donne à l'utilisateur tous les priviléges sur toutes les bases (\*.\*) :

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON *.* TO 'bob'@'%' WITH GRANT OPTION;
```

On recharge la table des priviléges, avant de quitter :

```
MariaDB [(none)]> FLUSH PRIVILEGES;
```

```
MariaDB [(none)]> quit;
```

Tester l'accès local :

```
$ mariadb -u username -p
```

Plus besoin d'avoir les droits administrateur  
pour lancer cette commande.

## Autoriser l'accès distant au serveur mariadb

Modifier le fichier de configuration du serveur :

```
# nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

Explications et autres méthodes ici :

<https://mariadb.com/kb/en/configuring-mariadb-for-remote-client-access/>

Dans la section [mysqld], chercher la ligne *bind-address* et remplacer :

```
bind-address = 127.0.0.1 par bind-address = 0.0.0.0
```

Redémarrer le service *mariadb* :

```
# systemctl restart mariadb.service    ou bien    # service mariadb restart
```

Tester la connexion distante :

```
$ nc -z adresseIP 3306 ; echo $? # doit renvoyer 0
```

```
$ mariadb -h adresseIP -u username -p
```

Si problème, vérifiez que le pare-feu  
autorise bien le port 3306 en TCP.

### Choisir un outil graphique pour manipuler les bases de données :

- Phpmyadmin
- Adminer



### Installer phpMyAdmin

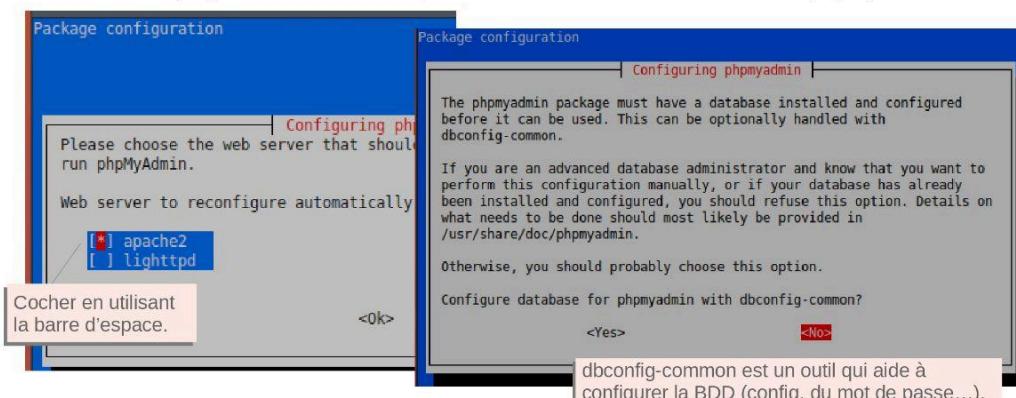
PHPMyAdmin est une application développée en php, et qui vise à fournir une interface simplifiée pour MySQL.

Installer : # apt install phpmyadmin

Dans certaines distri. Linux, le paquet ne sera disponible qu'en activant les backports dans /etc/apt/sources.list.d/debian.list

Attention, choisir le bon serveur http : sélectionner Apache2 (mettre une \* devant apache2).

À l'écran suivant, répondre « NON » à la question relative à la création d'une BDD phpmyadmin.



Si vous vous êtes trompé de réponses lors de l'installation de phpmyadmin, exécuter :

# dpkg-reconfigure phpmyadmin

Puis, redémarrer le serveur web :

# systemctl restart apache2    ou bien    # service apache2 restart

Tester un accès local à l'adresse suivante : <http://127.0.0.1/phpmyadmin>

Tester un accès distant (depuis une autre machine).



### Installer adminer

Installer le paquet :	# apt install adminer
Modifier la conf. d'apache :	# a2enconf adminer.conf
Redémarrer Apache :	# systemctl restart apache2    ou bien    # service apache2 reload

Tester un accès local à l'adresse suivante : <http://127.0.0.1/adminer>

Tester un accès distant (depuis une autre machine).

### **3.1 Manuel d'Utilisation :**

#### Connexion au site web

Pour commencer, il faut ouvrir un navigateur internet (chrome, firefox, edge ou autre) et se connecter sur le serveur. Pour ce faire, dans la barre d'adresse du navigateur, soit on rentre l'adresse IP (ici, 172.16.150.1) puis sur la page atteinte, on clique sur le fichier connexion.php, soit on marque directement 172.16.150.1/connexion.php. Vous arrivez sur la «page de Connexion».

Ensuite il faut remplir les zones de texte «Utilisateur» et «Mot de passe» avec vos identifiants. Initialement, il existe un identifiant avec les droits administrateur (Utilisateur: admin et Mot de passe: mdp) et un avec les droits utilisateur (Utilisateur: user et Mot de passe: user). Si les champs remplis sont corrects alors vous accédez à la «page du Moniteur», sinon vous demeurez sur la page de Connexion et un message d'erreur s'affiche pour vous demander de recommencer.

#### Visualisation des données

Une fois sur la page du Moniteur, vous devez sélectionner, dans la liste déroulante titrée «Données désirées», le type de données à afficher (soit «Puissance» en Watts, soit «Consommation» en kW/h à la journée). Ensuite, vous devez choisir une «Date de début» et une «Date de fin» (grâce aux deux calendriers prévus pour cet effet) pour définir la période sur laquelle vous souhaitez consulter les données, vous devez également choisir, grâce à la liste déroulante «Choix du module», l'appareil de mesure qui vous intéresse. Enfin, il ne vous reste plus qu'à cliquer sur le bouton «Envoyer» pour faire apparaître le graphique vous permettant de visualiser les données désirées.

#### Gestion des utilisateurs

Dans le cas où vous êtes connecté en tant qu'administrateur, un bouton supplémentaire intitulé «Administre» s'affiche en haut à droite de la page du Moniteur. Celui-ci vous permet d'accéder à la «Page d'Administration» afin de gérer les utilisateurs et consulter le «Journal des connexions».

La partie «Gestion des utilisateurs» permet de voir dans un tableau quels sont les utilisateurs existants. Vous pouvez soit en créer un nouveau, soit en supprimer un ou encore modifier le mot de passe et/ou le statut administrateur (Oui/Non) déjà attribués :

- Créer un nouvel utilisateur: Remplissez les deux zones de texte (Utilisateur et Mot de passe) et sélectionnez si la personne aura le statut administrateur (cochez Oui) ou pas (cochez Non). Puis cliquez sur le bouton «Ajouter/Modifier». Le nouvel utilisateur apparaît désormais dans le tableau au-dessus !
- Supprimer un utilisateur: Pour commencer, dans le tableau, cliquez sur la ligne de l'utilisateur que vous désirez supprimer. Ses identifiants et son statut apparaissent dans les zones de texte et cases à cocher en dessous. Cliquez sur le bouton «Supprimer». La ligne disparaît du tableau, l'utilisateur n'existe plus !

-Modifier un utilisateur: Pour commencer, dans le tableau, cliquez sur la ligne de l'utilisateur que vous désirez modifier. Ses identifiants et son statut apparaissent dans les zones de texte et cases à cocher en dessous. Vous pouvez changer le mot de passe dans la zone de texte ou cocher une case différente pour le statut. Puis cliquez sur le bouton «Ajouter/Modifier». L'utilisateur apparaît maintenant dans le tableau avec ses modifications !

### Journal des connexions

Le journal de connexions est présenté sous la forme d'un tableau comportant 3 colonnes: Le nom d'utilisateur, la date et l'heure de connexion. Ce tableau est agencé de façon décroissante, c'est à dire que les connexions les plus récentes apparaissent en premier. Les résultats sont affichés par séries de 20 connexions sur chaque page. En bas du tableau se trouve une liste déroulante qui affiche le numéro de la page actuelle et où vous pouvez sélectionner la page à afficher.

## 3.2 Code Source :

### 3.2.1 connexion.php

```
❶ connexion.php
1  <?php
2  session_start();
3  $_SESSION['Jeton'] = false; //création d'un jeton
4  ?>
5
6  <!DOCTYPE html>
7  <html lang="fr">
8      <head>
9          <meta charset="utf-8">
10         <title>Page de Connexion</title>
11         <link rel="stylesheet" type="text/css" href="connexion.css">
12     </head>
13     <body>
14         <!-- Logo at the top left corner -->
15         <div style="position:absolute; top:10px; left:10px;">
16             
17         </div>
18
19         <!-- Centered form container -->
20         <div class="form-wrapper" style="display:flex; flex:1;
21             align-items:center; justify-content:center; min-height:100vh;">
22             <form action="moniteur.php" method="post" style="width:50%; max-width:400px;">
23                 Utilisateur <input type="text" name="login" required>
24                 Mot de passe <input type="password" name="psswrd" required>
25             <?php
26             $id_correct = $_SESSION['IdCorrect'];
27             if ($id_correct == false && isset($_SESSION['IdCorrect']))
28             {
29                 echo '<p style="color:#FF0000;"><b>Utilisateur et/ou Mot de passe incorrect(s).
30                     <br>Veuillez recommencer!</b></p>';
31             }
32             ?>
33                 <input type="submit" value="Envoyer">
34             </form>
35
36
37
38         </div>
39
40     </body>
41 </html>
```

### 3.2.2 moniteur.php

```
moniteur.php
1  <?php
2
3 // ***** Partie Connexion *****
4 session_start();
5 $_SESSION['IdCorrect'] = true;
6 $liaison = mysqli_connect("localhost", "root", "root", "Projet");
7 $requete_credenciais = mysqli_query($liaison, "SELECT Utilisateur, MotDePasse, Admin FROM Identifiant");
8
9 if ($_SESSION['Jeton'] == false)
10 {
11 while ($enreg = $requete_credenciais->fetch_assoc())
12 {
13     if($_POST['login'] == $enreg['Utilisateur'] && $_POST['psswrd'] == $enreg['MotDePasse'])
14     {
15         $_SESSION['IdCorrect'] = true;
16         $_SESSION['Jeton'] = true;
17         $_SESSION['Admin'] = $enreg['Admin'];
18         mysqli_query($liaison,"INSERT INTO LogConnexion (Identifiant, Date, Heure)
19                                VALUES (\\"". $_POST['login']. "\", CURDATE(), CURTIME())");
20     }
21 }
22 }
23
24 //vérification du jeton
25 if ($_SESSION['Jeton'] == false)
26 {
27 $_SESSION['IdCorrect'] = false;
28 header ("Location:connexion.php");
29 }
30
31 ?>
32 <!DOCTYPE html>
33 <html lang="fr">
34 <head>
35     <title>Page du Moniteur</title>
36     <link rel="stylesheet" type="text/css" href="moniteur.css">
37 </head>
38 <body>
39     <?php if ($_SESSION['Admin'] == 1): ?>
40     <div class="admin-button-container">
41         <form action="administration.php" method="post">
42             <input type="submit" value="Administrer" />
43         </form>
44     </div>
45     <?php endif; ?>
```

```

46
47 <div class="form-container">
48 <form method="post" action="moniteur.php">
49     <h3>Données désirées: </h3>
50     <select name="Donnees" id="donnees" required>
51         <option <?php if(isset($_POST['Donnees']) && $_POST['Donnees'] == 'Puissance') ?>
52             |         echo 'selected'; ?>>Puissance</option>
53         <option <?php if(isset($_POST['Donnees']) && $_POST['Donnees'] == 'Consommation') ?>
54             |         echo 'selected'; ?>>Consommation</option>
55     </select>
56
57     <h3>Période: </h3>
58     <div class="date-container">
59         <div class="date-group">
60             <h4>Date de début:</h4>
61             <input type="date" id="date_debut" name="Date_debut" min="2024-02-17" max="2150-12-31"
62                 value=<?php echo isset($_POST['Date_debut']) ? $_POST['Date_debut'] : ''; ?>">
63         </div>
64         <div class="date-group">
65             <h4>Date de fin:</h4>
66             <input type="date" id="date_fin" name="Date_fin" min="2024-02-17" max="2150-12-31"
67                 value=<?php echo isset($_POST['Date_fin']) ? $_POST['Date_fin'] : ''; ?>">
68         </div>
69     </div>
70
71     <h3>Choix du module: </h3>
72     <?php
73         // création de la requête pour récupérer les noms des modules dans la BDD
74         $req_nom_module = "SELECT Nom FROM ModuleMesure";
75
76         // Récupération des noms
77         $requete_nom_module = mysqli_query($liaison, $req_nom_module);
78     ?>
79
80     <!-- Parcourir les résultats de la requête et afficher les données dans la liste déroulante -->
81     <select id="nom_module" name="Nom_module" required>
82         <?php
83             while ($row = mysqli_fetch_assoc($requete_nom_module)) {
84                 $selected = '';
85                 if(isset($_POST['Nom_module']) && $_POST['Nom_module'] == $row['Nom']) {
86                     $selected = 'selected';
87                 }
88                 echo '<option value="' . $row['Nom'] . '" ' . $selected . '>' . $row['Nom'] . '</option>';
89             }
90         ?>
91     </select>
92

```

```

93     <div class="submit-button-container">
94         |   <input type="submit" value="Envoyer" name = "BtValide">
95     </div>
96   </form>
97   </div>
98 </body>
99 </html>
100
101 <?php
102
103 if (isset ($_POST['BtValide'])) //lance la suite seulement après validation du formulaire
104 {
105 $date_debut = $_POST['Date_debut'];
106 $date_fin = $_POST['Date_fin'];
107
108 $req_module_id = mysqli_query($liaison, "SELECT ID FROM ModuleMesure
109 |   |   | WHERE Nom='". $_POST['Nom_module']."' ");
110
111 $row_module_id = mysqli_fetch_assoc($req_module_id);
112
113 // ***** Récupération de la liste des dates par jour *****
114
115 // au format français
116 function listeDates_fr($date_debut, $date_fin) {
117     $dates_fr = array();
118     $dateDebut = strtotime($date_debut);
119     $dateFin = strtotime($date_fin);
120
121     while ($dateDebut <= $dateFin) {
122         $dates_fr[] = date('d.m.Y', $dateDebut);
123         $dateDebut = strtotime('+1 day', $dateDebut);
124     }
125     return $dates_fr;
126 }
127
128 // au format américain
129 function listeDates_us($date_debut, $date_fin) {
130     $dates_us = array();
131     $dateDebut = strtotime($date_debut);
132     $dateFin = strtotime($date_fin);
133
134     while ($dateDebut <= $dateFin) {
135         $dates_us[] = date('Y-m-d', $dateDebut);
136         $dateDebut = strtotime('+1 day', $dateDebut);
137     }
138     return $dates_us;
139 }
```

```

140 // Vérification de la cohérence des dates
141 if ($date_debut >= date('Y-m-d') || $date_fin >= date('Y-m-d'))
142 {
143     echo "<script> alert (\"Les dates ne peuvent pas être supérieures
144     | ou égales à celle d'aujourd'hui !\"); </script>";
145     die();
146 }
147
148 if ($date_debut > $date_fin)
149 {
150     echo "<script> alert (\"La date de fin doit être strictement supérieure à celle du début !\");
151     | </script>";
152     die();
153 }
154
155 // **** Partie Récupération des Mesures ****
156
157 // Utilisation de la fonction count() pour obtenir le nombre de cases du tableau de dates
158 $nombreDeCases = count(listeDates_fr($date_debut, $date_fin));
159
160 // Créer un tableau pour stocker les données récupérées
161 $donnees = array_fill(0, $nombreDeCases, null);
162
163 // Requête pour récupérer les données moyennes par jour
164 $dates_copy = listeDates_us($date_debut, $date_fin);
165 for ($i = 0; $i < $nombreDeCases; $i++)
166 {
167     $req = "SELECT AVG(Puissance) AS Moyenne_mesure FROM Mesure WHERE Date = \"\$dates_copy[\$i]\""
168     | AND IdModuleMesure = \".$row_module_id['ID'].\"";
169     $requete_mesure = mysqli_query($liaison, $req);
170     $row = mysqli_fetch_assoc($requete_mesure);
171     $donnees[$i] = $row['Moyenne_mesure'];
172     if ($_POST['Donnees'] == "Consommation")
173     {
174         $donnees[$i] = $donnees[$i]*24/1000; //conversion des kW en kWh
175     }
176 }
177
178 // ***** Partie Conception du Graphique *****
179
180 // Convertir les tableaux PHP en JSON pour pouvoir l'utiliser en JavaScript
181 $donnees_json = json_encode($donnees);
182 $liste_dates_json = json_encode(listeDates_fr($date_debut, $date_fin));
183 if ($_POST['Donnees'] == "Consommation")
184 {
185     $donnees_mesure_json = json_encode($_POST['Donnees'] . " (en kWh à la journée)");
186 }
187 else
188 {
189     $donnees_mesure_json = json_encode($_POST['Donnees'] . " (en Watts)");
190 }
191 ?>
192
193 <html>
194 <head>
195     <meta charset="UTF-8">
196     <meta name="viewport" content="width=device-width, initial-scale=1.0">
197     <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
198 </head>
199 <body>
200 <div class="graph-container"><canvas id="myChart" class="graph-center" style="max-width: 1200px;
201 | max-height: 800px; background-color: white; opacity: 90%"></canvas></div>
202
203

```

```

204 <script>
205
206     // Récupérer les données JSON et les convertir en tableau JavaScript
207     var donnees = <?php echo $donnees_json; ?>;
208     var dates = <?php echo $liste_dates_json; ?>;
209     var label_courbe = <?php echo $donnees_mesure_json; ?>;
210
211     // Créer un graphique
212     var ctx = document.getElementById('myChart').getContext('2d');
213     var myChart = new Chart(ctx, {
214         type: 'line',
215         data: {
216             labels: dates,
217             datasets: [
218                 {
219                     label: label_courbe,
220                     data: donnees,
221                     backgroundColor: 'rgba(0, 127, 255, 1)',
222                     borderColor: 'rgba(0, 127, 255, 1)',
223                     borderWidth: 1
224                 }
225             ],
226             options: {
227                 scales: {
228                     yAxes: [
229                         {
230                             ticks: {
231                                 beginAtZero: true
232                             }
233                         }
234                     ];
235                 }
236             });
237         </script>
238     </body>
239     </html>
240     <?php
241     ?
242     ?>
```

### 3.2.3 administration.php

```
administration.php
1  <?php
2  session_start();
3
4  //vérification du jeton
5  if ($_SESSION['Jeton'] == false)
6  {
7      $_SESSION['IdCorrect'] = false;
8      header ("Location:connexion.php");
9  }
10 ?>
11
12 <!DOCTYPE html>
13 <html lang="fr">
14 <head>
15     <meta charset="utf-8">
16     <meta name="viewport" content="width=device-width, initial-scale=1.0">
17     <title>Page d'Administration</title>
18     <link rel="stylesheet" type="text/css" href="administration.css">
19
20     <form action="moniteur.php" method="post">
21         <input type="submit" value="Retour à la page du Moniteur" />
22     </form>
23
24     <h2>Gestion des utilisateurs: </h2>
25
26     <script>
27         function remplirChamps(nom, motDePasse, admin)
28     {
29         document.getElementById('nom_modifiable').value = nom;
30         document.getElementById('mot_de_passe_modifiable').value = motDePasse;
31         if (admin == '1')
32         {
33             document.getElementById('admin_oui').checked = true;
34         }
35         else
36         {
37             document.getElementById('admin_non').checked = true;
38         }
39     }
40     </script>
41
42     <style>
43         .ligne-modifiee
44         {
45             background-color: lightgreen;
46         }
47     </style>
48 </head>
```

```

49 <body>
50
51 <?php
52 // Fonction pour se connecter à la base de données
53 function connecterBaseDeDonnees()
54 {
55     $serveur = "localhost";
56     $utilisateur = "CodePhp";
57     $motDePasse = "CodePhp";
58     $baseDeDonnees = "Projet";
59
60     $connexion = new mysqli($serveur, $utilisateur, $motDePasse, $baseDeDonnees);
61
62     if ($connexion->connect_error) {
63         die("Échec de la connexion : " . $connexion->connect_error);
64     }
65
66     return $connexion;
67 }
68
69 // Affichage du tableau des utilisateurs
70
71 function afficherTableauUtilisateurs($connexion, $derniereModification)
72 {
73     $sql = "SELECT Utilisateur, MotDePasse, Admin FROM Identifiant";
74     $resultat = $connexion->query($sql);
75
76     if ($resultat->num_rows > 0) {
77         echo "<table border='1'>";
78         echo "<tr><th>Nom</th><th>Mot de Passe</th><th>Administrateur</th></tr>";
79         while ($ligne = $resultat->fetch_assoc())
80         {
81             $classeLigne = '';
82             if ($derniereModification === $ligne["Utilisateur"])
83             {
84                 $classeLigne = 'ligne-modifiee';
85             }
86             echo "<tr class='$classeLigne' onclick=\"remplirChamps('" . $ligne["Utilisateur"]
87             . "', '" . $ligne["MotDePasse"] . "', '" . $ligne["Admin"] . "')\">";
88             echo "<td>" . $ligne["Utilisateur"] . "</td>";
89             echo "<td>" . $ligne["MotDePasse"] . "</td>";
90             echo "<td>" . ($ligne["Admin"] == '1' ? 'Oui' : 'Non') . "</td>";
91             echo "</tr>";
92         }
93         echo "</table>";
94     }
}

```

```

95     else
96     {
97         echo "0 résultats";
98     }
99 }
100
101 // Traitement de l'ajout ou de la modification d'utilisateur
102 if (isset($_POST['ajouter_modifier']))
103 {
104     $nom = $_POST['nom'];
105     $motDePasse = $_POST['mot_de_passe'];
106     $admin = isset($_POST['admin']) ? $_POST['admin'] : '0';
107
108     $connexion = connecterBaseDeDonnees();
109
110     // Vérifier si l'utilisateur existe déjà dans la base de données
111     $sql = "SELECT * FROM Identifiant WHERE Utilisateur='$nom'";
112     $resultat = $connexion->query($sql);
113
114     if ($resultat->num_rows > 0)
115     {
116         // L'utilisateur existe, effectuer une mise à jour
117         $sql = "UPDATE Identifiant SET MotDePasse='$motDePasse', Admin='$admin' WHERE Utilisateur='$nom'";
118     }
119     else
120     {
121         // L'utilisateur n'existe pas, effectuer une insertion
122         $sql = "INSERT INTO Identifiant (Utilisateur, MotDePasse, Admin) VALUES ('$nom', '$motDePasse', '$admin')";
123     }
124
125     if ($connexion->query($sql) === FALSE) // Vérification erreur de connexion
126     {
127         echo "Erreur lors de l'opération : " . $connexion->error;
128     }
129
130     $derniereModification = $nom;
131
132     $connexion->close();
133 }
134

```

```

135 // Traitement de la suppression d'utilisateur
136 if (isset($_POST['effacer']))
137 {
138     $nom = $_POST['nom'];
139
140     $connexion = connecterBaseDeDonnees();
141     $sql = "DELETE FROM Identifiant WHERE Utilisateur='$nom'";
142
143     if ($connexion->query($sql) === FALSE) // Vérification erreur de connexion
144     {
145         echo "Erreur lors de la suppression de l'utilisateur : " . $connexion->error;
146     }
147
148     $derniereModification = $nom;
149
150     $connexion->close();
151 }
152
153 // Afficher le tableau avec la dernière modification
154 $derniereModification = isset($derniereModification) ? $derniereModification : '';
155 afficherTableauUtilisateurs(connecterBaseDeDonnees(), $derniereModification);
156 ?>
157
158 <form method="post" action="" class="admin-form">
159     <div class="form-row">
160         <div class="form-group">
161             <label for="nom_modifiable"><b>Nom</b></label>
162             <input type="text" id="nom_modifiable" name="nom">
163         </div>
164         <div class="form-group">
165             <label for="mot_de_passe_modifiable"><b>Mot de Passe</b></label>
166             <input type="text" id="mot_de_passe_modifiable" name="mot_de_passe">
167         </div>
168         <div class="form-group">
169             <label><b>Administrateur</b></label>
170             <div class="radio-group">
171                 <input type="radio" id="admin_oui" name="admin" value="1">
172                 <label for="admin_oui">Oui</label>
173                 <input type="radio" id="admin_non" name="admin" value="0">
174                 <label for="admin_non">Non</label>
175             </div>
176         </div>
177         <div class="form-actions">
178             <input type="submit" name="ajouter_modifier" value="Ajouter/Modifier">
179             <input type="submit" name="effacer" value="Supprimer">
180         </div>
181     </div>
182 </form>

```

```

183
184 <!-- ***** Partie Journal de Connexion ***** -->
185
186 <h2>Journal des connexions: </h2>
187
188 <?php
189 // Connexion à la base de données
190 $serveur = "localhost";
191 $utilisateur = "CodePhp";
192 $motDePasse = "CodePhp";
193 $baseDeDonnees = "Projet";
194
195 $conn = new mysqli($serveur, $utilisateur, $motDePasse, $baseDeDonnees);
196
197 // Vérifier la connexion
198 if ($conn->connect_error)
199 {
200     die("Connection failed: " . $conn->connect_error);
201 }
202
203 // Pagination
204 $rows_per_page = 20;
205 $page = isset($_GET['page']) && is_numeric($_GET['page']) ? $_GET['page'] : 1;
206 $start = ($page - 1) * $rows_per_page;
207
208 // Récupérer les données de la base de données
209 $sql = "SELECT Identifiant, Date, Heure FROM LogConnexion ORDER BY ID DESC LIMIT $start, $rows_per_page";
210 $result = $conn->query($sql);
211
212 // Afficher les données dans un tableau
213 echo "<table border='1'>";
214 echo "<tr><th>Nom</th><th>Date</th><th>Heure</th></tr>";
215
216 if ($result->num_rows > 0)
217 {
218     // Afficher les données ligne par ligne
219     while ($row = $result->fetch_assoc())
220     {
221         echo "<tr>";
222         echo "<td>" . $row['Identifiant'] . "</td>";
223         echo "<td>" . $row['Date'] . "</td>";
224         echo "<td>" . $row['Heure'] . "</td>";
225         echo "</tr>";
226     }
227 }

```

```
204 <script>
205
206     // Récupérer les données JSON et les convertir en tableau JavaScript
207     var donnees = <?php echo $donnees_json; ?>;
208     var dates = <?php echo $liste_dates_json; ?>;
209     var label_courbe = <?php echo $donnees_mesure_json; ?>;
210
211     // Créer un graphique
212     var ctx = document.getElementById('myChart').getContext('2d');
213     var myChart = new Chart(ctx, {
214         type: 'line',
215         data: {
216             labels: dates,
217             datasets: [
218                 {
219                     label: label_courbe,
220                     data: donnees,
221                     backgroundColor: 'rgba(0, 127, 255, 1)',
222                     borderColor: 'rgba(0, 127, 255, 1)',
223                     borderWidth: 1
224                 }
225             ],
226             options: {
227                 scales: {
228                     yAxes: [
229                         {
230                             ticks: {
231                                 beginAtZero: true
232                             }
233                         }
234                     ]
235                 }
236             });
237     </script>
238 </body>
239 </html>
240 <?php
241 ?>
```