

BADIN JEFFREY

BTS SYSTÈMES NUMÉRIQUES / INFORMATIQUE ET  
RÉSEAU

# RAPPORT PROJET

POSTUROMÈTRE  
EPREUVE E62



ANNÉE 2020-2021

## Sommaire :

<b>Introduction</b>	<b>3</b>
<b>Algorithmes</b>	<b>4</b>
<b>Répartition des tâches</b>	<b>9</b>
<b>Plannification du projet</b>	<b>10</b>
<b>Conception</b>	
- <b>Classe Posturomètre</b>	<b>11</b>
- <b>Diagramme de séquence</b>	<b>12</b>
- <b>Acquisition de la masse des charges</b>	<b>14</b>
- <b>Stockage des données en local</b>	<b>15</b>
- <b>Transfert des données vers la base de données</b>	<b>16</b>
<b>Cahier de recettes</b>	<b>19</b>
<b>Installation et Mode d'emploi</b>	<b>20</b>
<b>Schéma de câblage</b>	<b>22</b>
<b>Maintenance</b>	<b>23</b>
<b>Conclusion</b>	<b>24</b>
<b>Annexes</b>	<b>25</b>

# INTRODUCTION

Lors de ce projet, il était demandé de concevoir un système capable d'alerter un employé dans le cas où il ne garde pas le dos droit lors de ses manipulations. Il permet également d'estimer les charges portées et enfin il archive toutes ces informations afin de pouvoir en tirer des statistiques de pénibilité du travail. Une corrélation devait également pouvoir être faite avec le poste de travail occupé afin d'identifier les postes qui semblent causer le plus de problèmes.

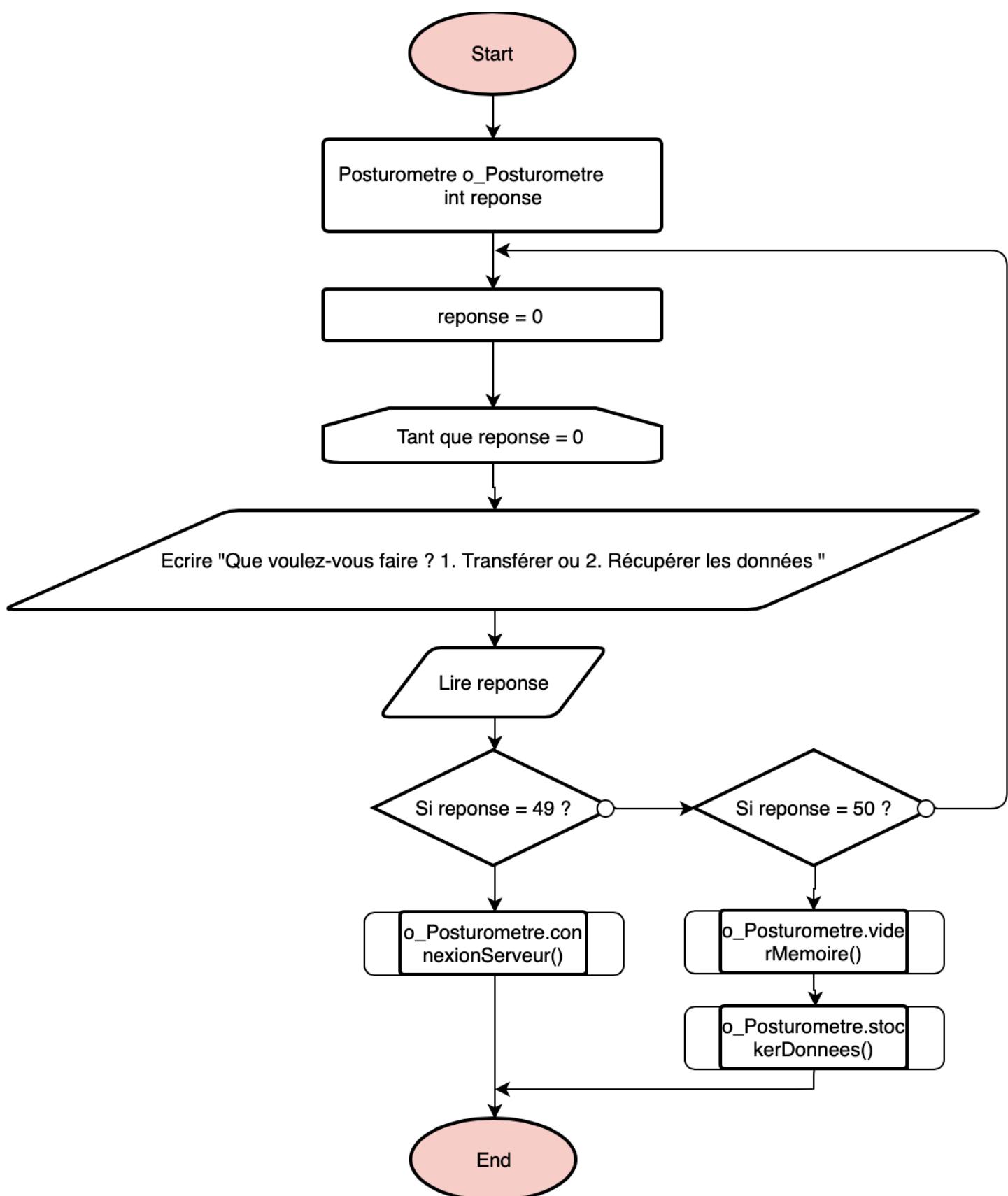
Ma participation dans ce projet a été de travailler sur la partie d'acquisition de masse des charges à l'aide de capteur de pression, de stocker les données et les transférer à la base de données.

Pour commencer, j'ai étudié la transmission d'informations par liaison Ethernet pour comprendre comment fonctionne cette transmission et les différentes bibliothèques Arduino que j'ai utilisé.

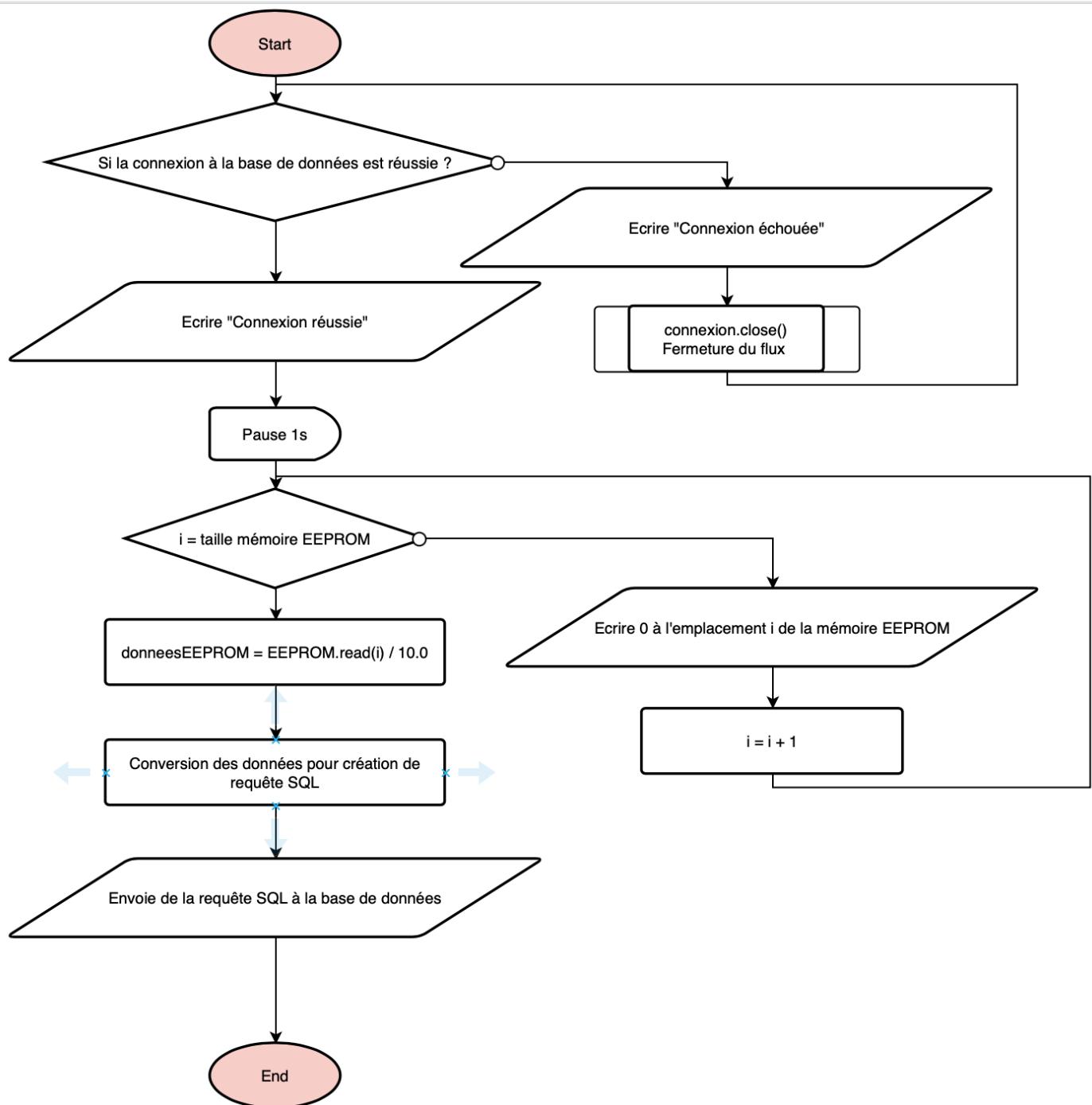
Ensuite, j'ai cherché différents capteurs mis à ma disposition afin de choisir celui qui correspondait au mieux à mon cahier des charges.

J'ai donc commencé à travailler sur la liaison Ethernet avec le Shield Ethernet de la carte Arduino Uno Rev3. Cette transmission remplit totalement mon cahier des charges concernant la facilité d'utilisation et la possibilité de changer pour une liaison Wifi. Cette transmission peut également alimenter la carte Arduino en PoE dans le cas d'un transfert de données. Ensuite, j'ai travaillé sur le stockage des données de masse, je stocke des nombres à virgule dans la mémoire EEPROM, le stockage en local des données en attente du transfert en fin de journée. Puis finalement, j'ai travaillé sur l'acquisition de la masse des charges à l'aide des capteurs de pression MXP5700AP.

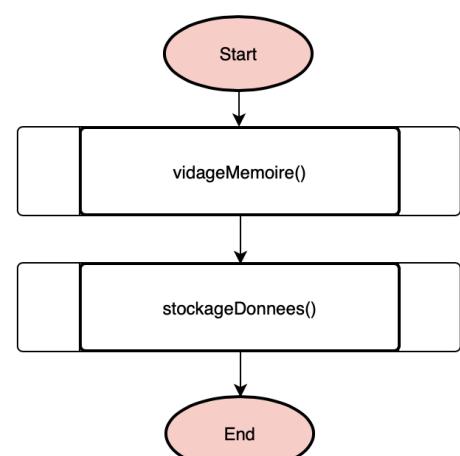
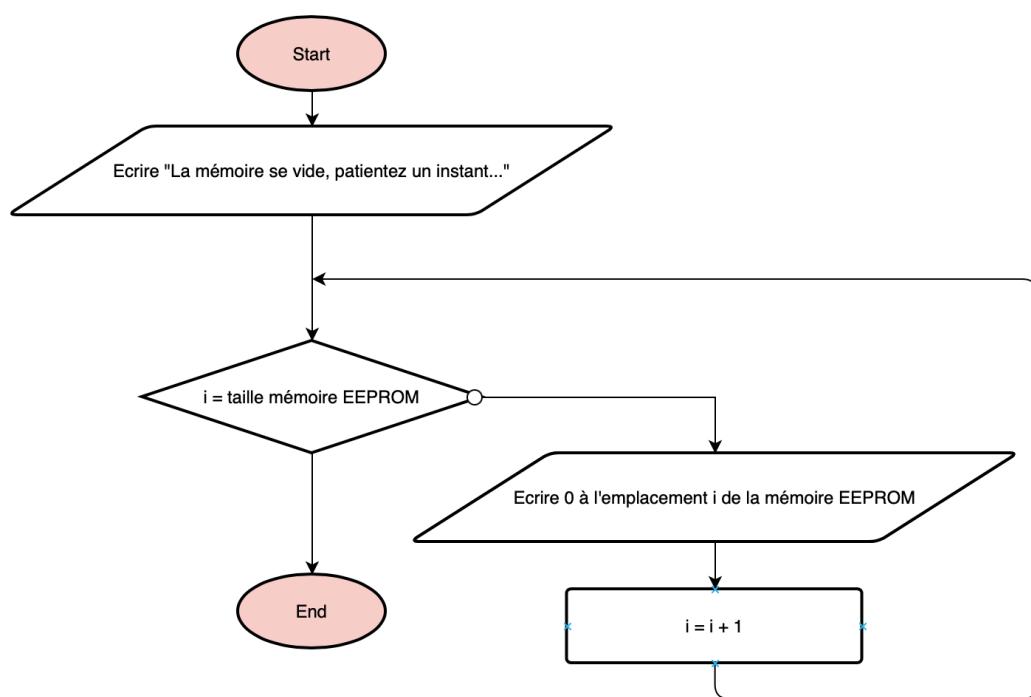
# ALGORITHME POSTUROMÈTRE



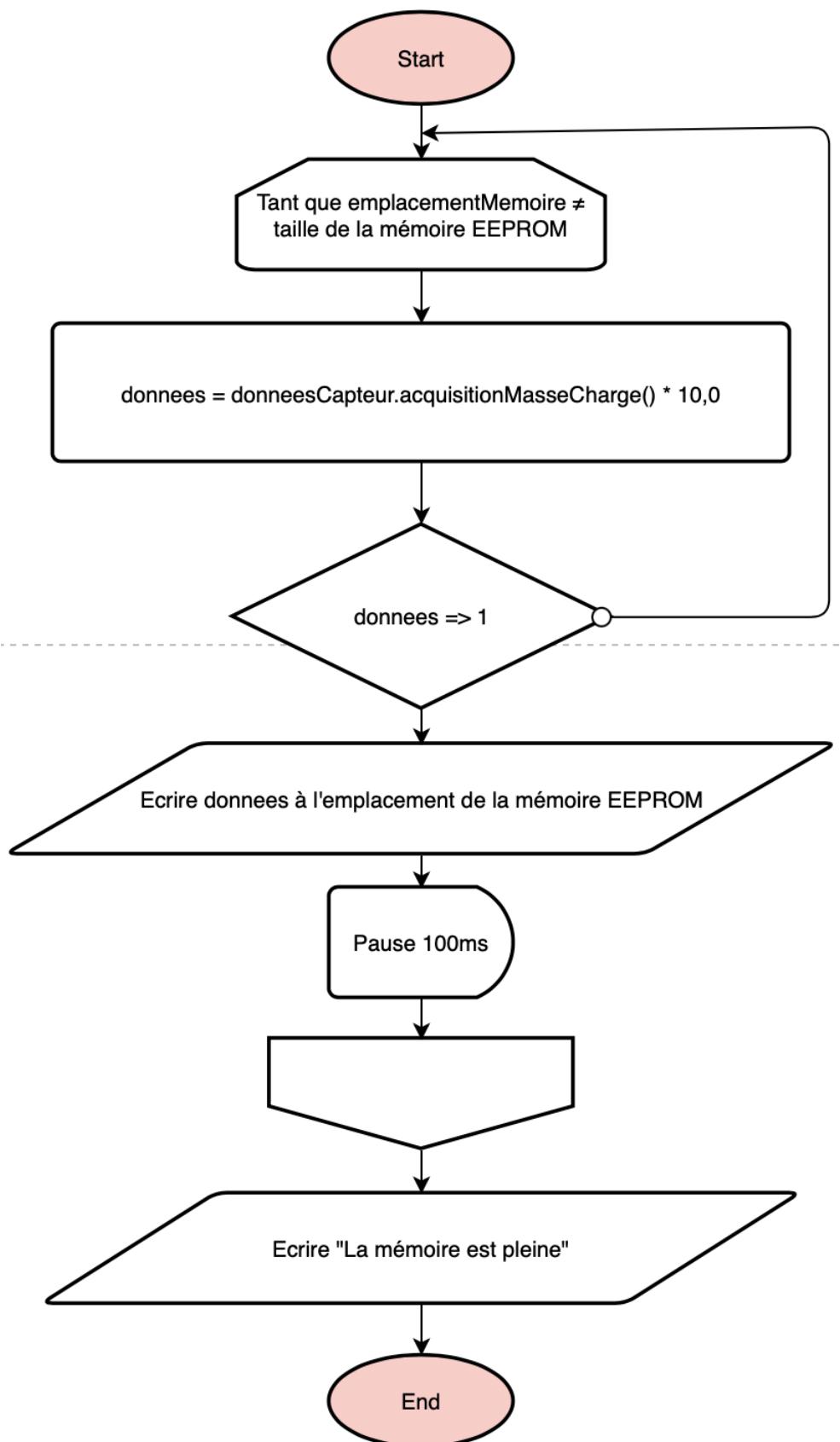
# ALGORITHME LIAISON ETHERNET



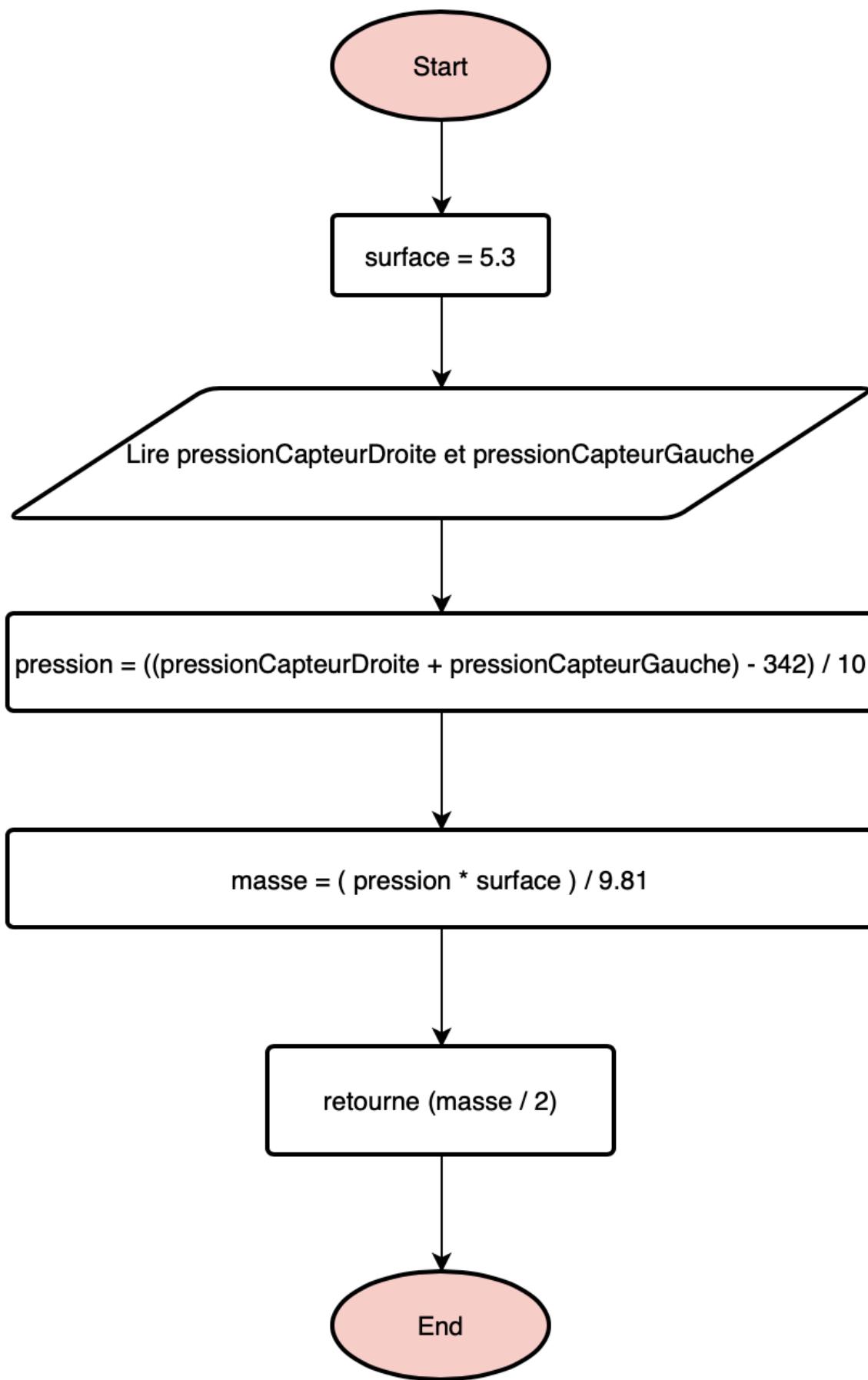
# ALGORITHME STOCKAGE



## ALGORITHME STOCKAGE



## ALGORITHME ACQUISITION DE MASSE

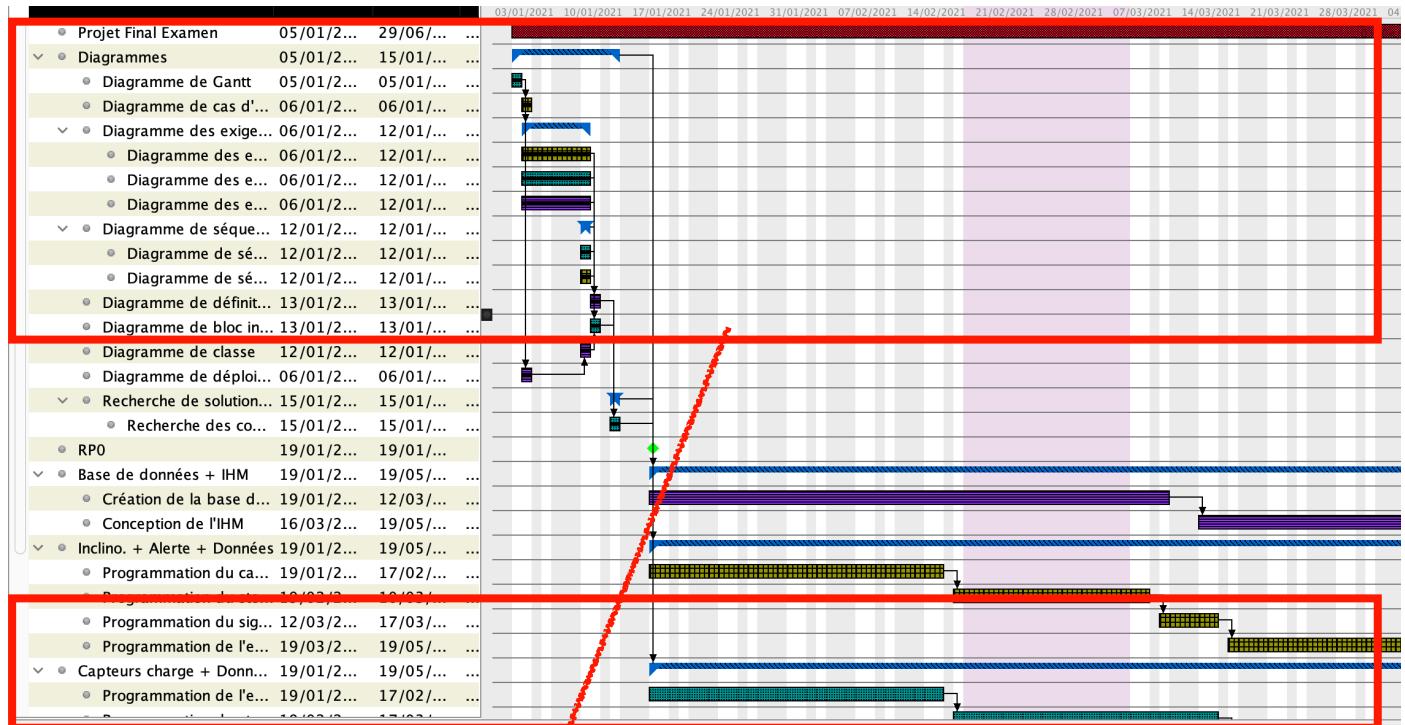


# RÉPARTITION DES TÂCHES

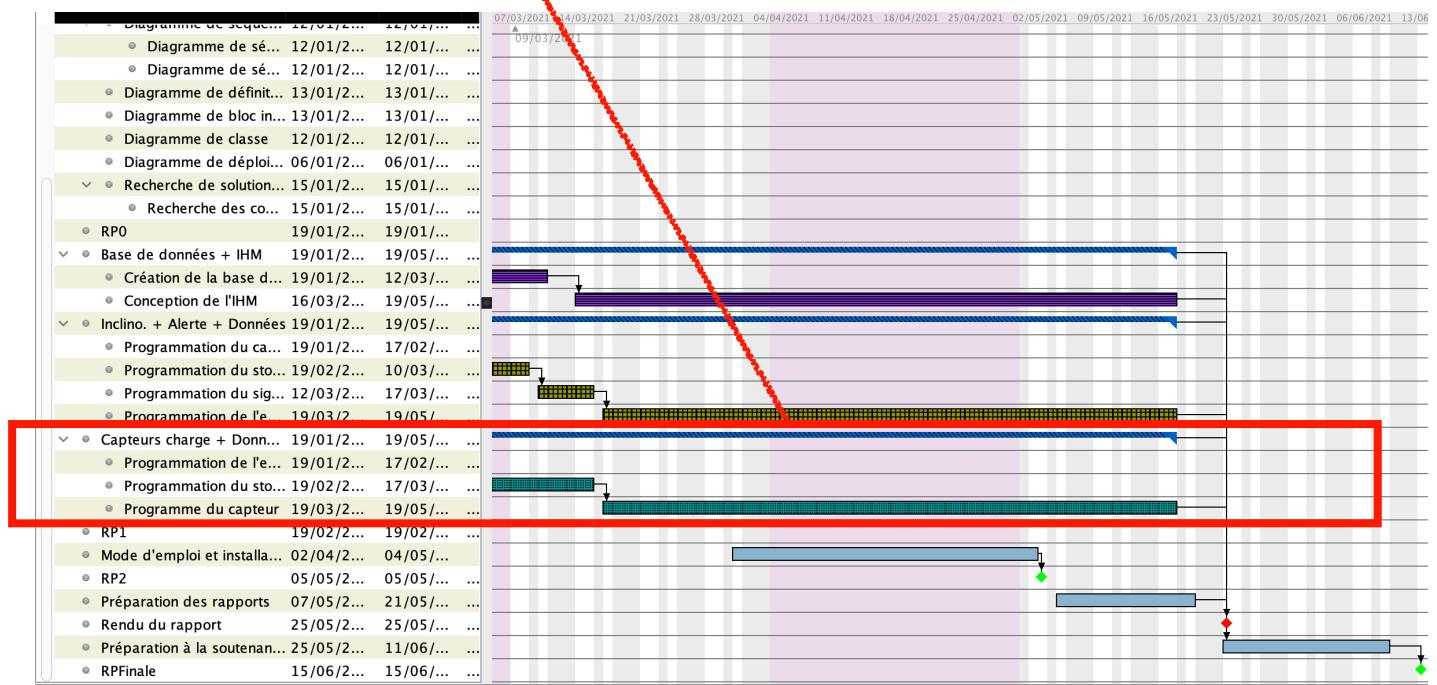
## I. LES TÂCHES EFFECTUÉES :

- Recherche et choix des capteurs.
- Recherche d'information sur la transmission avec la liaison Ethernet et le Shield Ethernet.
- Programmation de la classe LiaisonEthernet.
- Effectué des tests pour établir une connexion entre la base de données et la carte Arduino.
- Recherche d'information sur la mémoire EEPROM de la carte Arduino Uno Rev3.
- Programmation de la classe Stockage.
- Effectué des tests de stockage de données au bon format.
- Recherche d'information sur les capteurs de pression et le lien avec la masse.
- Programmation de la classe Masse.
- Fusion de toutes les classes en sous-classe généralisé par la classe Posturomètre.
- Optimisation du programme dans la carte.
- Assemblage de toutes les parties du projet.

# PLANNING DU PROJET

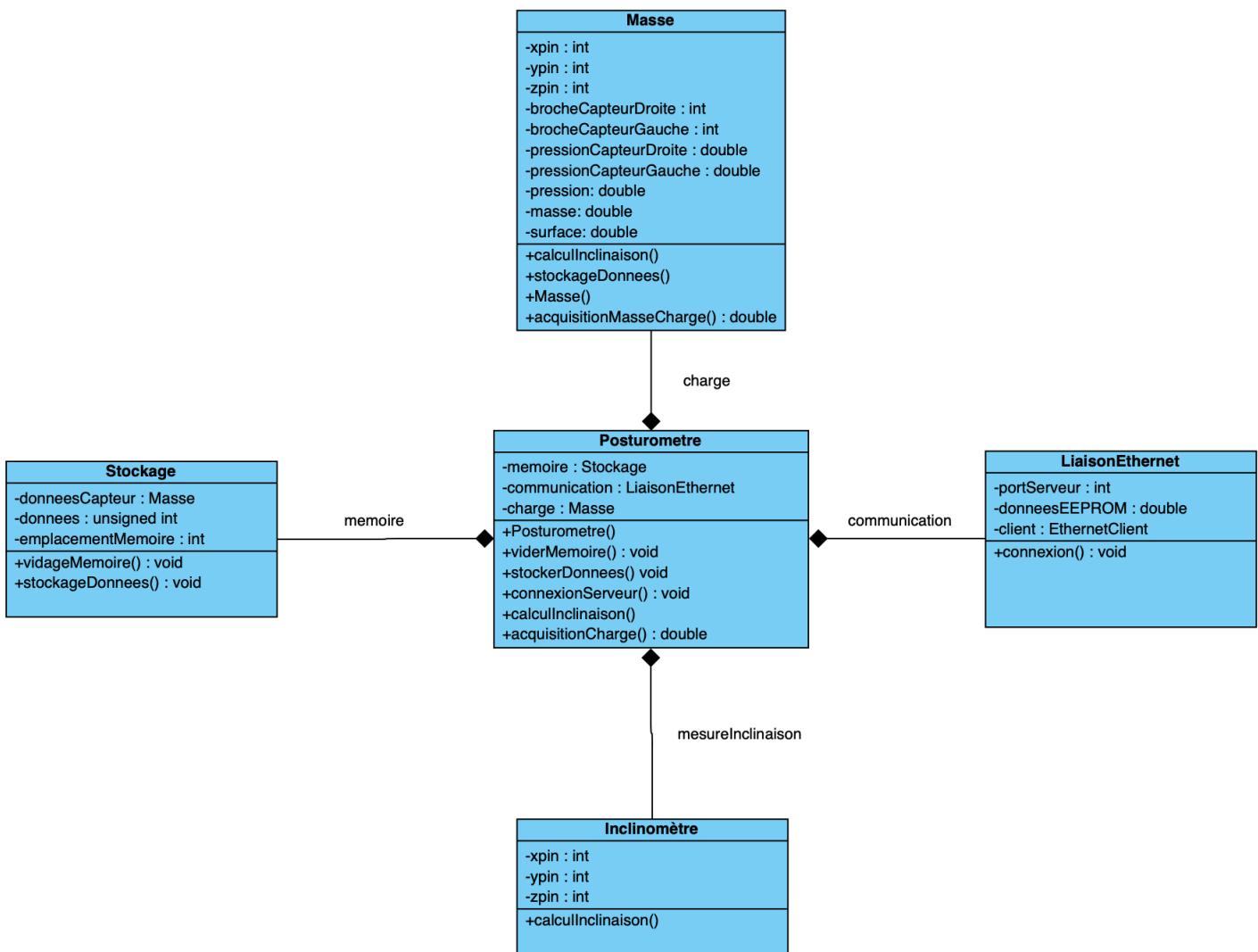


## Mes tâches



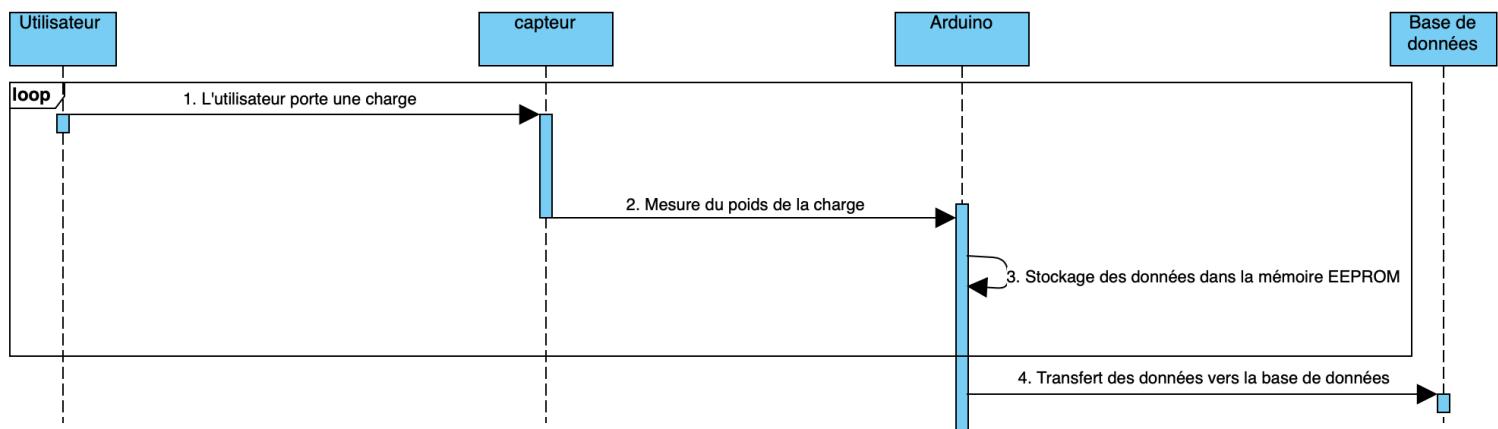
# CONCEPTION

## I. CLASSE POSTUROMÈTRE



# CONCEPTION

## II. DIAGRAMME DE SÉQUENCE :



## CONCEPTION

### III. ACQUISITION DE LA MASSE DE CHARGES

L'acquisition de la masse des charges se fait à l'aide de capteur de pression. Le lien entre la masse est la pression est régi par la formule :

m : masse en kg

$$m = (P \times S) / g$$

avec P : pression en kPa

S : surface en cm<sup>2</sup>

g : Accélération sur Terre en N/kg

Cela permet de ressortir la masse d'une charge avec une précision de plus ou moins 0,1 kg.

### IV. STOCKAGE DE DONNÉES EN LOCAL

Le stockage des données devant être effectué en local avant le transfert vers la base de données, on utilise la mémoire EEPROM de la carte Arduino Uno Rev3. Celui peut enregistrer jusqu'à 1024 octets soit 1024 valeurs.

De plus le stockage des valeurs se fait sous une condition, il faut que qu'il y ait une variation de masse suffisante (soit plus d'1kg) et que celle ci ne dépasse pas 15kg.

Également, une option a été ajoutée pour pouvoir vider la mémoire avant utilisation pour éviter les problèmes de données qui se superposeraient.

### V. TRANSFERT DE DONNÉES VERS LA BASE DE DONNÉES

Le transfert de données vers la base se fait via liaison Ethernet, pour cela on utilise le principe de client-serveur. On donne une adresse IP et une adresse MAC à la carte Arduino et l'on effectue la connexion à la base de données à l'aide d'une commande de la bibliothèque « **MySQL\_Connection.h** ». Puis une fois connecté, une requête SQL est créée et envoyée si elle respecte les normes de type de données imposées. (double)

# ACQUISITION DE LA MASSE DES CHARGES

Posturometre	Masse.cpp	Masse.hpp	Posturometre.cpp	Posturometre.hpp	Stockage.cpp
--------------	-----------	-----------	------------------	------------------	--------------

```

#include "Arduino.h"
#include "Masse.hpp"

Masse::Masse(){
    brocheCapteurDroite = 0;
    brocheCapteurGauche = 1;
    surface = 5.3; //Surface sur laquelle s'appuie la masse de la charge
}

double Masse::acquisitionMasseCharge(){

    pressionCapteurDroite = analogRead(brocheCapteurDroite);
    pressionCapteurGauche = analogRead(brocheCapteurGauche);

    pression = ((pressionCapteurDroite + pressionCapteurGauche) - 342.00)/10;

    Serial.print(pression);
    Serial.print("\t");

    masse = (pression * surface)/ 9,81; //Formule de la masse à l'aide d'une pression en N/m^2
    Serial.println(masse/2);

    return masse/2;
}

```

Main Page	Classes ▾	Files ▾
-----------	-----------	---------

**Masse Class Reference**

## Public Member Functions

### Masse ()

Constructeur de la classe **Masse**.

### double acquisitionMasseCharge ()

Cette méthode sert à gérer l'acquisition des données de masse des charges portées par l'utilisateur. [More...](#)

## Member Function Documentation

### • acquisitionMasseCharge()

```
double Masse::acquisitionMasseCharge ( )
```

Cette méthode sert à gérer l'acquisition des données de masse des charges portées par l'utilisateur.

#### Returns

Cette méthode renvoie un double

The documentation for this class was generated from the following files:

- [Masse.hpp](#)
- [Masse.cpp](#)

# STOCKAGE DES DONNÉES EN LOCAL

```

Posturometre Masse.cpp Masse.hpp Posturometre.cpp Posturometre.hpp Stockage.cpp
#include "Masse.hpp"
#include "Stockage.hpp"
#include <EEPROM.h>
#include <Arduino.h>

Stockage::Stockage(){
    //Implémentation du constructeur...
    emplacementMemoire = 0;
}

void Stockage::vidageMemoire(){
    //Implémentation de la fonction de remise à zéro de la mémoire...
    Serial.println("La mémoire se vide, patienter un instant... ");
    for<int i = 0; i < EEPROM.length(); i++>{
        EEPROM.write(i, 0);
    }
}

void Stockage::stockageDonnees(){
    //Implémentation de la fonction de stockage des données dans la mémoire...
    while (emplacementMemoire != EEPROM.length()){
        donnees = donneesCapteur.acquisitionMasseCharge()*10.0;
        if(donnees >= 1){
            EEPROM.update(emplacementMemoire, donnees);
            emplacementMemoire++;
            delay(100);
        }
    }
    Serial.println("Mémoire pleine");
}

```

Main Page

Classes ▾

Files ▾

## Stockage Class Reference

### Public Member Functions

**Stockage ()**
Constructeur de la classe **Stockage**.
**void vidageMemoire ()**
Cette méthode sert à gérer la suppression des données précédemment récupérées. [More...](#)
**void stockageDonnees ()**
Cette méthode sert à gérer le stockage des données localement dans la mémoire EEPROM de carte Arduino. [More..](#)

### Member Function Documentation

**◆ stockageDonnees()**

void Stockage::stockageDonnees ( )

Cette méthode sert à gérer le stockage des données localement dans la mémoire EEPROM de carte Arduino.

**Returns**

Cette méthode ne renvoie rien

**◆ vidageMemoire()**

void Stockage::vidageMemoire ( )

Cette méthode sert à gérer la suppression des données précédemment récupérées.

**Returns**

Cette méthode ne renvoie rien

The documentation for this class was generated from the following files:

- [Stockage.hpp](#)
- [Stockage.cpp](#)

# TRANSFERT DES DONNÉES VERS LA BASE DE DONNÉES

Posturometre	Masse.cpp	Masse.hpp	Posturometre.cpp	Posturometre.hpp	Stockage.cpp	Stockage.hpp	liaisonEthernet.cpp §	liaisonE
--------------	-----------	-----------	------------------	------------------	--------------	--------------	-----------------------	----------

```

#include <EEPROM.h>
#include <MySQL_Connection.h>
#include <MySQL_Cursor.h>
#include <MySQL_Encrypt_Sha1.h>
#include <MySQL_Packet.h>
#include <Arduino.h>
#include "Ethernet.h"
#include "SPI.h"
#include "LiaisonEthernet.hpp"
#include "Masse.hpp"

LiaisonEthernet::LiaisonEthernet(){
    //Implémentation du constructeur:
    byte adresseMAC[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
    IPAddress ipArduino(192,168,1,4);
    portServeur = 3306;
}

void LiaisonEthernet::connexion(){
    //Implémentation de la fonction de vérification de la connexion :
    byte adresseMAC[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
    IPAddress ipArduino(192,168,1,4);
    IPAddress ipServeur(192,168,1,186);
    Ethernet.begin(adresseMAC, ipArduino);

    MySQL_Connection connexion((Client *) &client);
    MySQL_Cursor * cur_mem = new MySQL_Cursor(&connexion); //Gestion de la véracité de la trame lors de l'envoie des données

    char donneesCapteur [10];
    char requete[128];

    char utilisateur[] = "admincharge";
    char mot_de_passe[] = "Admincharge";

    char INSERT_SQL[] = "INSERT INTO statistique.charge (id_capteur, masseCharge, dateCharge, utilisateur_id) VALUES ('1',%s,NOW(),'2')";
    //Requête permettant l'inclusion de données dans la base de données

    if(connexion.connect(ipServeur, portServeur, utilisateur, mot_de_passe)){
        Serial.println("Connexion réussie.");
        delay(1000);
    }else{
        //Si la connexion n'aboutie pas alors on ferme le flux
        Serial.println("Connexion echouee.");
        connexion.close();
    }

    for (int i = 0; i < EEPROM.length(); i++){
        donneesEEPROM = EEPROM.read(i)/10.0;
        dtostrf(donneesEEPROM, 5, 1, donneesCapteur); //Conversion d'un double en string par la fonction dtostrf
        sprintf(requete, INSERT_SQL, donneesCapteur); //Écriture de la chaîne de caractère créée précédemment dans la requête SQL

        Serial.println(requete);
        delay(500);
        cur_mem->execute(requete);
    }
}

```

# TRANSFERT DES DONNÉES VERS LA BASE DE DONNÉES

Main Page	Classes ▾	Files ▾
-----------	-----------	---------

## LiaisonEthernet Class Reference

### Public Member Functions

[LiaisonEthernet \(\)](#)

Constructeur de la classe [LiaisonEthernet](#).

**void connexion ()**

Cette méthode sert à gérer la connexion à la base de données. [More...](#)

### Member Function Documentation

#### ◆ connexion()

`void LiaisonEthernet::connexion ( )`

Cette méthode sert à gérer la connexion à la base de données.

#### Returns

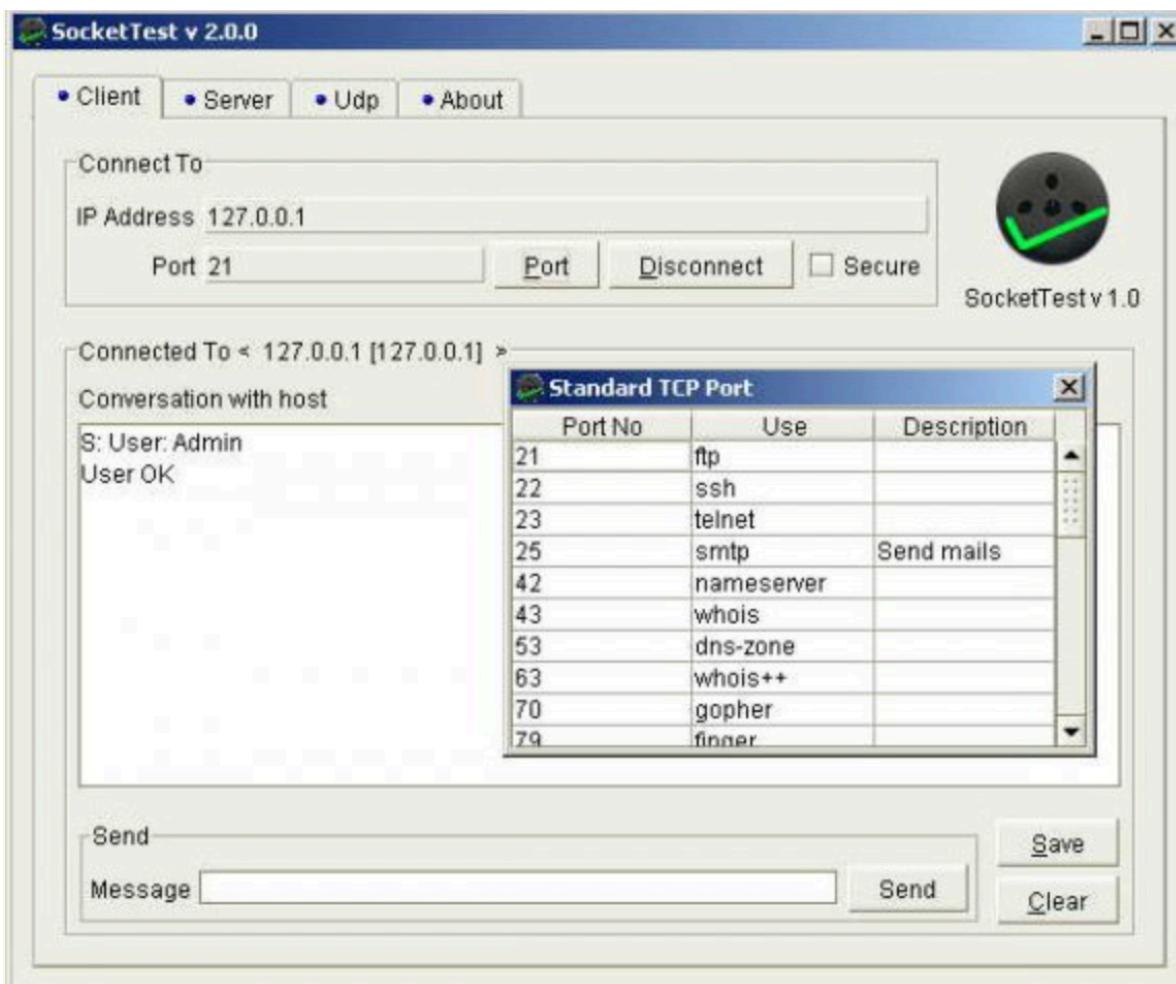
Cette méthode renvoie un caractère

The documentation for this class was generated from the following files:

- [liaisonEthernet.hpp](#)
- [liaisonEthernet.cpp](#)

# TRANSFERT DES DONNÉES VERS LA BASE DE DONNÉES

## TEST EFFECTUÉ :



J'ai utilisé le logiciel SocketTest pour pouvoir tester la connexion entre la carte Arduino et la base de données se trouvant sur un autre poste. Pour cela je mettais le logiciel sur le poste sur lequel la base de données était installé et j'envoyé des données aléatoirement pour voir si la communication fonctionnait ou non.

## CAHIER DE RECETTES

16	Les capteurs MXP5700 se situe dans les chaussures de l'utilisateur
17	Gestion d'erreur de connexion à la base de données
18	Utilisation d'un identifiant et d'un mot de passe pour sécuriser l'accès
19	Contrôle du type de données envoyées lors de la transmissions
20	Pression atmosphérique non tenu en compte
21	Possibilité de demande à l'utilisateur un nom d'utilisateur, mot de passe et l'adresse IP

OK NOK

Les capteurs sont intégrés aux chaussures de l'utilisateur pour une utilisation simple.	X	
Si la connexion échoue le moniteur série envoie un message d'erreur.	X	
Empêcher les personnes non autorisés d'accéder aux données de la base.	X	
Seul le type double est envoyé dans la base de données.(masse avec valeur contenant une virgule)	X	
On supprime la valeur de la valeur de la pression atmosphérique pour avoir un capteur absolu et non relatif.	X	
Demander à l'utilisateur de rentré ses identifiants.		X

Ce cahier de recettes regroupe les éléments du cahier des charges qui ont pu être réalisé et ce qui a été contraint d'abandonner. Chaque ligne regroupe le besoin du cahier des charges, un commentaire expliquant comment l'exigence est ou n'est pas satisfaite et une case cochée d'un OK pour la réalisation de la solution ou un NOK de l'abandon de la solution.

# INSTALLATION ET MODE D'EMPLOI

## PARTIE RÉCOLTE DE DONNÉES :

Pour pouvoir installer le Posturomètre, il vous suffit de revêtir les chaussures équipées des capteurs de pression.



Ensuite il faut connecter la carte Arduino à la batterie. Et il vous suffit plus qu'à attendre quelques secondes avant de commencer vos manipulations.



Et accrocher la carte Arduino à votre torse à l'aide du harnais, comme le montre l'image suivante :



(En cas de problème de capteurs de pression mal connectés, veuillez vous référer au schéma de câblage.)

# INSTALLATION ET MODE D'EMPLOI

## PARTIE TRANSFERT DE DONNÉES :

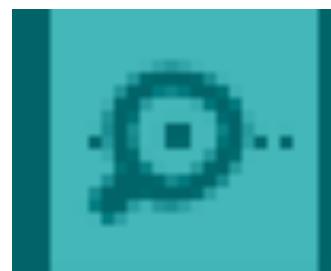
Pour pouvoir transférer les données récoltées grâce au Posturomètre, il vous suffit de connecter la carte Arduino à un ordinateur. Pour cela commencer par raccorder la carte Arduino à l'aide du câble USB-B pour l'alimenter.



Ensuite, pour que la carte Arduino soit connectée à la base de données il faut raccorder un câble RJ45 à celle-ci.



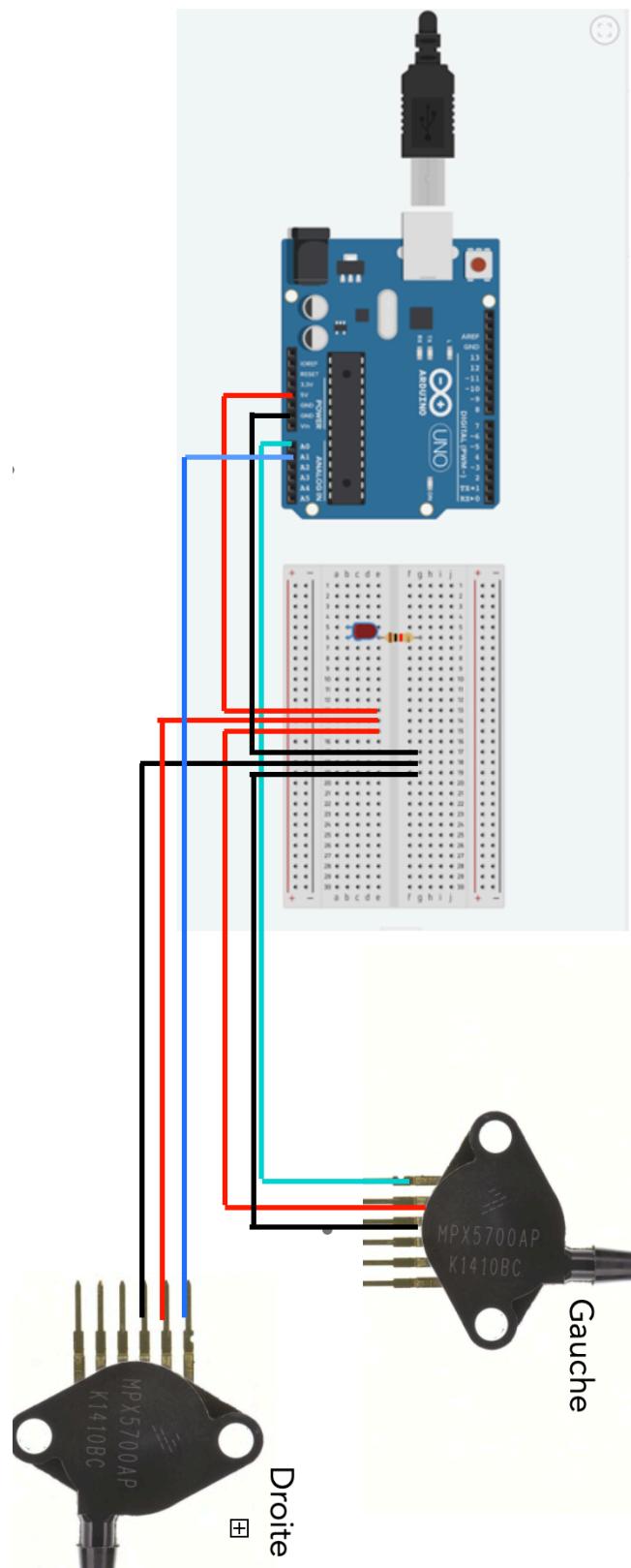
Puis il vous faut ouvrir le fichier « Posturometre.ino » dans le programme Arduino. Ouvrir le moniteur série avec le symbole en haut à droite de la fenêtre :



Une fois ceci fait, vous aurez une phrase à l'écran : « 1.Récupération de donnees. ou 2.Transfert. ». Écrivez « 2 » puis faites « Entrée ». La carte Arduino transférera les données récoltés dans la base de données.

(S'il y a un problème référez-vous au dossier de maintenance.)

## SCHÉMA DE CÂBLAGE



## MAINTENANCE

Si la carte Arduino ne s'allume pas, veuillez vérifier si tous les branchements d'alimentation sont bien faits. Dans le cas d'une récolte de données, vérifier les branchements de la batterie et dans le cas d'un transfert de données, vérifier les branchements du câble USB-B à la carte Arduino.



Si la récolte de données ne fonctionne pas, vérifier les branchements des capteurs de pression à la carte Arduino en vous référant au schéma de câblage.

Lors d'un transfert de données, si la connexion n'opère pas ou s'il y a une erreur de connexion, commencer par vérifier le branchement du câble RJ45 au Shield Ethernet de la carte Arduino.



Si cela ne fonctionne toujours pas vérifier les adresses IP de la carte Arduino et du serveur sur lequel se trouve la base de données. Pour cela, vérifier dans le fichier « liaisonEthernet.cpp » si les adresses IP correspondent à celle de la carte et du serveur.

(Si vos problèmes persistent veuillez contacter le service après vente.)

## CONCLUSION

Donc lors de ce projet, j'ai pu améliorer mes compétences en tant que technicien. J'ai pu découvrir de nouvelles méthodes de programmations tel que la Programmation Orientée Objet (POO) mais également des outils comme phpMyAdmin ou Adminer pour la base de données, SocketTest pour gérer la connexion à un serveur depuis la carte Arduino ou bien encore l'IDE d'Arduino et ces bibliothèques.

Mon travail s'est concrétisé par une classe Posturomètre qui est composé d'autres classes Masse, LiaisonEthernet et Stockage que j'ai conçu. Ces classes ont pour but de gérer ma partie du Posturomètre et donc à collecter les données de masse de charges portées par l'utilisateur, de les stocker localement et de les transférer en fin de journée.

Pour le Posturomètre j'avais des idées d'évolutions si plus de temps m'était accordé, par exemple l'alimentation de la carte Arduino par PoE (Power Over Ethernet), la connexion par liaison Wifi qui permettrait de gérer le transfert des données en temps réel et éviterait l'utilisation de la mémoire EEPROM de la carte Arduino et donc la perte de données. Également j'aurais voulu améliorer la capacité de l'utilisateur à choisir ses identifiants (son nom d'utilisateur et son mot de passe) directement via le moniteur série de l'IDE Arduino.

En conclusion, le projet répond au besoin en ce qui concerne ma partie, le Posturomètre peut mesurer la masse d'une charge d'un maximum de 15kg avec une précision de plus ou moins 0,1 kg. Il peut également stocker localement les données avant le transfert en fin de journée et le transfert de données est sécurisé par identification via nom d'utilisateur et mot de passe, avec une vérification des données transférées et une gestion des erreurs de connexions.

## ANNEXES

- ANNEXES\_01.pdf : Broches du capteur de pression (Datasheet MXP5700AP)
- ANNEXES\_02.pdf : Désignation des broches du capteur de pression (Datasheet MXP5700AP)
- ANNEXES\_03.pdf : Courbe de transfert (Datasheet MXP5700AP)
- ANNEXES\_04.pdf : Broches de la carte Arduino Uno Rev3 (Datasheet Arduino Uno Rev3)
- ANNEXES\_05.pdf : Caractéristiques techniques capteurs de pression (Datasheet MXP5700AP)